SlipStream Documentation Documentation

Release v3.52

SixSq

Mar 29, 2019

Contents

Documentation for users, administrators, and developers of the SlipStream Cloud Application Management Platform from SixSq.

CHAPTER 1

SlipStream Features

Developed by SixSq, **SlipStream is a multi-cloud application management platform**. It automates the full application management lifecycle, including the deployment, testing, certification and optimization of the application, within Infrastructure as a Service (IaaS) cloud infrastructures.

The software's features appeal to a wide variety of different users:

- *End-users* will use SlipStream as an **application provisioning engine** to deploy virtual machines as well as full services, such as data analysis clusters. Users will find that SlipStream's multi-cloud support makes deploying resources into different cloud infrastructures uniform and transparent.
- *Service providers* can use SlipStream to define customized images and full systems (batch clusters, LAMP applications, analysis platforms, etc.), creating a **rich catalog of services** that end-users can deploy with the click of a button.
- *Software developers* will be thrilled with SlipStream's ability to provide dynamic, near-production environments for **realistic testing of full software systems**. Multi-cloud support takes this testing to the next level, allowing the behavior of systems to be tested over the wide area network.
- *DevOps practitioners* can take advantage of SlipStream's automation and scaling capabilities to **deploy, maintain, and optimize production applications** in multiple cloud infrastructures.

CHAPTER 2

SlipStream Releases

Results from each development cycle are packaged and released. The process occurred approximately once every two weeks.

SlipStream version v3.71 is the **last planned release for SlipStream**. It is the only version of SlipStream supported at this time. The evolution of SlipStream will be called Nuvla 2.0. Browse the (preliminary) Nuvla 2.0 documentation to follow its development.

2.1 Supported Releases

2.1.1 v3.71 - 29 March 2019

Release v3.71 contains a single bug fix for the webui.

For Everyone:

- Fix issue with webui where the redirects when deleting resources pointed to non-existant paths.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- WebUI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

2.2 Older SlipStream Releases

These releases are no longer supported for production deployments.

2.2.1 2018 Releases

v3.70 - 01 March 2019

Release v3.70 contains a couple of bug fixes and a new notification resource.

For Everyone:

- Fix compatibility problem with using ExternalObject resources against a Minio S3 server.
- Addition of a new notification resource that will allow more flexible management of user notifications.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- WebUI

- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.69 - 15 February 2019

Release v3.69 is essentially for minimal bugfixes

For Everyone:

- Fix regression when registering via exoscale coupon

For Dave

- Fix for incorrect metadata reference
- Use kebab-case for resource attributes

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- WebUI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.68 - 01 February 2019

Release v3.68 improves the metadata for use in autogenerated forms and reorganizes some UI features

For Everyone:

- Welcome page and sidebar menu is cleaned from CIMI references
- Links to API documentation point to internal documentation
- Deployment card link is improved
- UI is now embedded into the GSSC Wordpress environment via an iframe element
- A confirmation message is added before stopping or deleting a deployment

For Dave

- Update of the metadata for use in autogenerated forms.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- WebUI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.67 - 18 January 2019

Release v3.67 fixes the known issues from the previous release and improve the dataset information description.

For Everyone:

- Fix regression when downloading reports

For Dave

- Update of schema of the Deployment resource to save dataset information for data objects.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- WebUI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.66 - 21 December 2018

Release v3.66 improves the deployments workflow and user interface when datasets are included. It also introduces public external objects

For Everyone:

- UI improvements regarding datasets information to deployments
- External objects can be made public and downloadable

For Dave

- Add datasets to deployment service offers
- Support alpine for the bootstrap (needed primarily for containers).

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- WebUI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.65 - 07 December 2018

Release v3.65 introduces data volume options on deployments and improvements on S3 objects management

For Everyone:

- S3 objects metadata are collected and stored in the resource
- Deleting the last S3 object of a bucket also deletes the bucket
- Better error handling when interacting with S3
- Full support for data volume options
- Mounting volumes in containers feature
- Data page implements a full text search
- Deployment dialog is refined (progress as steps and jobs are detailled)
- use of SlipStream state machine state only when in started/stopped states

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI

- WebUI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.64 - 23 November 2018

Release v3.64 focused on introducing new CIMI resource for deployments and upgrade of clojurescript libraries.

For Everyone:

- Implementation of an initial set of deployment resources that will allow the CIMI modules to be deployed
- Added events on deployment state change and on execution state change

For Dave:

- Module actions for the application panel
- The module migration script is updated to comply with the latest changes of the schema.
- Fixed compiler warnings coming from latest ClojureScript release (1.10.439).
- Upgrade to the latest shadow-cljs (2.7.2) and closure (v20181028) releases.
- Moved metadata utilities to make them accessible to other submodules.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- WebUI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI

- SlipStreamJobEngine

v3.63 - 9 November 2018

Release v3.63 introduced the Metadata resource, added improvements on UI and bugfixes in some resources schema.

For Everyone:

- Upgrade of datepicker in the UI
- Added copy to clipboard
- Appstore new UI based on deployment template and cimi deployment
- Visualization for Metadata
- Fix external-object authorizations for actions like Download
- Attributes added (*acceptProtocols* and *acceptContentTypes*) to the modules resources
- Attribute added (*data*) to deployment resource
- Update on the schema for the ServiceOffer resource to allow both a fully-qualified connector identifier and the current abbreviated one.

For Dave:

- Using shadow-cljs for dependencies
- The latest version of the CIMI specification has introduced "parent", therefore fixes were required when this introduced conflicts
- Fix in email resource schema
- Automatic resource metadata generation (using ephemeral storage)

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- WebUI
- Connectors
- Client
- SlipStreamClojureAPI

- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.62 - 26 October 2018

Release v3.62 focused on bugfixes, cleaning up of dead/deprecated code and update of the Slipstream API documentation

For Everyone:

- Removal of Electron as part of the UI
- Removal of deprecated 'vms' resource
- Implementation of the CIMI ResourceMetadata resource
- Update of API documentation (https://ssapi.sixsq.com) e.g vm, storage, metering and the new Quota resource
- Fixed bug when displaying the reset password dialog

For Dave:

- Upgrade of clojure to 1.10.0-beta3.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- WebUI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.61 - 12 October 2018

Release v3.61 focused on improving the UI (reset password, chart rendering)and upgrading the features for CIMI deployments

For Everyone:

- Fix Charts rendering in UI
- CIMI deployments
- Add a reset password link next to "Login with Nuvla Account"

For Dave:

- For Docker connector, publish endpoint instead of internal IP
- Improvements regarding CIMI deployment ports mappings for container

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- WebUI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.60 - 28 September 2018

Release v3.60 focused on improving the reliability of the SlipStream jobs and the performance of some WebUI pages

For Everyone:

- Improvement for Deployments as CIMI resource
- WebUI Improved performances (dashboard refresh, animations for charts)
- WebUI page for NuvlaBox (pagination added)

- Credential api key secret bug fix in claims edition

For Dave:

- Improved monitoring of VMs, including error handling
- Docker connnector now part of the upgrade process

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- WebUI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.59 - 14 September 2018

Release v3.59 has focused on enhanced performance of the WebUI and the introduction to a docker connector

For Everyone:

- Fix missing display of reports for users with long usernames
- Added about / welcome pages
- Enhanced display for metric charts

For Bob :

- Better display of billable resource
- Updated filter on the Usage page

For Dave:

- Additions to the Administrator Guide regarding “How to link Authentications to a User Account”
- Added priority support for job resource

- Support of a *disabledMonitoring* attribute for cloud connectors (performance optimisation)
- Implementation of a docker connector
- Fix on the describe instance command for connectors

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- WebUI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.58 - 31 August 2018

Release v3.58 has focused on the reliability of the resource usage data and its visualization.

For Everyone:

- Partially fix a problem with showing the usage data (Disk size)
- Optimize database for deletions
- Have Exoscale compatible with CIMI deployments

For Bob:

- Have a distinction in usage between compute and S3 storage
- Fix bug on UI for deployment panel causing blank page

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.57 - 17 August 2018

Release v3.57 has focused on improving the reliability of the resource usage data and its visualization. A number of bug fixes and improvements related to the usage data (e.g. job engine to collect information) have also been applied.

For Everyone:

- Partially fix a problem with showing the deployment reports.
- Fix user identifier issue when registering using OIDC servers.
- Fix a problem with the visualization of errors when blank values are provided in the login forms.
- Improve the loading times and accessibility support in the WebUI interface.
- Fix occasional unresponsive pages when viewing resource details with the WebUI.
- Force consistent initialization state of the WebUI to avoid spurious errors being displayed.

For Bob:

- Improve the reliability of the resource usage data.
- Improve the prototype visualization of this resource usage data in the WebUI.
- Add prototype metering resources for object storage.

For Dave:

- Allow the administrator to configure authentication of users directly with OIDC tokens.
- Fixes for deadlocked threads when treating jobs in the job engine.
- Support the SIGUSR1 signal in the job engine to retrieve thread stacktraces.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.56 - 3 August 2018

Release v3.56 has focused on improving the resource monitoring system within SlipStream to improve its coverage, precision, and reliability. In parallel, the usage dashboard has been improved to allow users to search, visualize, and download the usage information more efficiently.

In addition, a number of bugs were fixed and other enhancements have been rolled in.

For Everyone:

- The usage dashboard available from the newer web interface has been significantly enhanced to provide better search (and sort) capabilities, to view data more efficiently, and to allow download of the report data.
- As a result of the work on the usage dashboard, the newer web interface has been cleaned up, with more visual consistency between elements and many small interaction bugs corrected.
- Fixed a bug that prevented the deployment reports from being shown in the web interfaces.
- Fixed that caused user registration with a username/password to fail.

For Bob:

- Recovery of quota information from cloud service providers (starting with Exoscale) has been put in place to allow synchronization between SlipStream and provider quotas.
- Fixed the schema of the quotas to allow for zero limits, effectively blocking access to a particular resource.
- Metering has been improved to ensure that the correct people have access to the records and that the information is more precise.

For Dave:

- The logging for the job executor has been significantly improved. It now uses its own log file (rather than logging to syslog) and all messages have a consistent format and reasonable logging level.
- Support for both MITREid server and token authentication has been improved to allow a shared configuration of both authentication methods.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

- The login and sign up dialogs are not properly centered from the SlipStream welcome page. See GitHub Issue 789 for a description of the problem and the fix.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.55 - 21 July 2018

Two features dominated the work for this release:

- Enhancing the authentication process to allow users to authenticate with multiple methods for a single account and
- Refining the monitoring infrastructure to provide more accurate and better overviews of resource usage.

In addition, a number of bugs were fixed and other enhancements have been rolled in.

For Everyone:

- Fix a problem where external users making use of shared credentials could not terminate deployments.
- Allow for user registration with an Exoscale voucher that automatically creates an Exoscale account and configures the Nuvla account for all Exoscale regions.
- Change external authentication via MITREid (OIDC) servers to use unique identifier rather than the MITREid username.
- Fix the user registration workflow for browser-based clients.
- Fix an issue where specifying multiple SSH keys on an OpenStack deployment could prevent the key pair from being created.
- Simplify the user login and user sign up modals.

For Clara:

- Add full text search capabilities for the description attribute of CIMI resources. (Alpha feature subject to change.)

- Add CIMI-based modules (images, components, applications) to the server. (Alpha feature subject to change.)

For Bob:

- Add the concept of "credential managers" to allow for managers to have an overview of all resource usage related to the credential.
- Add disk size monitoring for virtual machine resources.

For Dave:

- Enhance the Exoscale connector to use a separate parameter for the root disk size, rather than relying on separate images with different default disk sizes.
- Fix a minor (and rare) problem with the job engine where there was a missing format in exception handling that affected the logs.
- Fix a problem with the handling of credentials when creating the monitoring resources for virtual machines.
- Allow multiple identities per user account. (See the migration instructions below concerning this change.)
- The self-registration template is not added by default. This must be added by the administrator to authorize self-registration of users.
- The problem with the slow start of the CIMI server was caused by insufficient entropy. It is recommended to always run the "haveged" service to avoid this problem. This has been added to the standard SlipStream installation.
- Multiple fixes and additions to the WebUI interface.

Alice, Bob, Clara, and Dave can be found here.

Migration

Migration of external users is required. See the usage instructions in the README on GitHub.

Known issues

- The deployment reports are not shown in the standard UI. See GitHub Issue 181 for resolution of this.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.54 - 30 June 2018

This release focused on improving the performance and reliability of the SlipStream server. This included identifying and fixing problems with the monitoring subsystem, properly catching exceptions to ensure relevant error messages for users, and improving the startup time for the CIMI server.

For Everyone:

- Improve monitoring subsystem to ensure that monitoring information for applications is not lost.
- Streamlined user registration with an Exoscale coupon. Creates accounts on SlipStream and Exoscale and automatically includes credentials in SlipStream.
- Ensure the user receives relevant error messages and status codes by catching exceptions within the server related to invalid input and resource conflicts.

For Bob:

- Fix UI issues for resource usage that would result in a blank page being presented.

For Dave:

- Updated workflow to ensure that all user information is included in user registrations from OpenID Connect (OIDC) identity providers based on MITREid.
- Provide simple job statistics on the WebUI to identify problems with the job subsystem.
- The CIMI server was starting slowly because of insufficient entropy for cryptographic actions. The SlipStream installation script now installs the "haveged" daemon. This is recommended for all installations.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required for this release.

Known issues

- The self-registration template is not added by default as before. The sign up form will not be visible in the WebUI, unless it is added manually.
- The CIMI server takes an extremely long time to start. The cause of this is being investigated.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.53 - 16 June 2018

The main changes for this release concern the user authentication and registration features. Those people using external identity providers must first register with SlipStream; previously accounts were created automatically. This release expands support for OIDC servers and lays the groundwork for linking multiple authentication methods to a single account.

For Everyone:

- Users identified via external identity providers must now explicitly register with SlipStream before being able to log into the service.
- OpenID Connect (OIDC) support has been expanded to support the MITREid Connect implementation (in addition to the existing Keycloak support), allowing more external identity providers to be used.
- Links to the Terms and Conditions document have been updated to those reflecting changing coming from the recent GDPR legislation.

For Alice:

- Fix an issue for the `ss-module-download` utility that caused it to fail when the module contained non-ASCII characters.
- Allow the `ss-module-download` utility to continue when errors (e.g. access permissions) occur.

For Dave:

- The OpenStack connector now contains an option to use and reuse floating IP addresses from an allocated pool. (Patch provided by IFB.)
- Fix issue where the NuvlaBox connector description would prevent the server from starting.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required for this release.

Known issues

- The self-registration template is not added by default as before. The sign up form will not be visible in the WebUI, unless it is added manually.
- The CIMI server takes an extremely long time to start. The cause of this is being investigated.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI

- SlipStreamJobEngine

v3.52 - 2 June 2018

This releases fixes the known issues from the previous release and refactors the authentication processes to make them more robust and more easily maintainable. For end-users, the primary changes are that the CYCLONE authentication method is no longer supported and SlipStream accounts are not created automatically for external logins (e.g. via GitHub or OpenID Connect).

For Everyone:

- Remove CYCLONE authentication support. Users who were using that authentication method must use another one (e.g. username/password).
- Fix display of version in footer.

For Dave:

- Refactor authentication processes to use explicit callback resource.
- Remove test dependencies leaking into production deployments.
- Ensure that deployment-specific API key/secret credentials are cleaned up even when a deployment is aborted.
- Fix job engine to use correct database index.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required for this release.

Known issues

- The parameter description for the NuvlaBox connector (if installed) prevents the SlipStream server from starting (see GitHub issue 165).

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.51 - 18 May 2018

This is a major release that updates the version of Elasticsearch and changes the mapping of resources to Elasticsearch indices. This impacts mainly SlipStream administrators. **All upgrades require a full migration of the database.**

For Dave:

- Allow different database bindings to be configured for the server.
- Provide alpha release for Elasticsearch binding based on its REST API.
- Simplify dependencies by taking SlipStream version from code rather than the service configuration.
- Upgrade to Elasticsearch 6, separating indices for resources and providing explicit mappings. This should improve performance and make management easier.
- Change session resource expiry date to make it possible to clean up expired sessions with simple Elasticsearch queries.
- Allow default ordering of events to be overridden through the API.

Alice, Bob, Clara, and Dave can be found here.

Migration

The version requires a full migration of the Elasticsearch database.

Both the old and new Elasticsearch clusters must be accessible during the migration process. You must run the upgrade process from a machine that can access both the old and new Elasticsearch clusters. Normally, this is the machine running the SlipStream services and we refer to this as the "SlipStream machine" below.

If you've not done so already, install a new Elasticsearch 6 cluster. Use the health checks to ensure that the cluster is functioning correctly before starting the migration process. This must be on a different machine from the one running your current production Elasticsearch cluster.

The first step is to download and setup the migration tools.

- On the SlipStream machine, install the Leiningen build tool. This will be used to download the dependencies required by the migration tools and then to run them.
- Ensure that Leiningen works by running `lein --help`. If it doesn't work, check the troubleshooting information on the Leiningen website.
- Download the SlipStreamMigration tarball that contains the migration tools.
- Unpack these tools in a convenient location on the SlipStream machine. The command to use is `tar zxf SlipStreamMigration-3.51.tar.gz`.
- **From the root of the unpacked tarball**, execute the command `lein with-profile +dbinit run -- --help`. Apologies for the tortured syntax.

This last command should download a large number of dependencies and end with usage information for the command. If it does not, verify that you are in the correct directory and that everything has been setup correctly. Contact support if you cannot resolve the issues.

The next step is to initialize the database with the indices and mappings for the SlipStream resources. **This must be done before any documents are migrated from the old database.** Execute the following commands:

```
$ export ES_HOST=es6-01.example.com
$ export ES_PORT=9300
$ lein with-profile +dbinit,+community,+enterprise run
```

Replace the hostname with your Elasticsearch 6 host. The "+community" and "+enterprise" initialize the database for the Community Edition and Enterprise Edition cloud connectors, respectively. Leave out those terms if they are not appropriate for your SlipStream installation.

Review the output from the dbinit tool. You should see the successful initialization of a large number of CIMI resources. You can ignore the zookeeper error concerning the initialization of the Job resource.

You can check the initialization by looking at the indices in Elasticsearch:

```
$ curl "http://$ES_HOST:9200/_cat/indices?v"
```

This should return a listing like the following:

```
health status index                                  uuid                   pri rep␣
↪docs.count docs.deleted store.size pri.store.size
green  open   slipstream-email                       Vy-Jjm4xQZaSyqTR3efRXQ   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-cloud-entry-point           tSxKHYdARhC4oZMZce-sPA   3   0  ␣
↪        1            0      7.2kb          7.2kb
green  open   slipstream-session-template            N4tSpCoASRKRmSUG7ktMxg   3   0  ␣
↪        1            0     10.4kb         10.4kb
green  open   slipstream-service-attribute-namespace rbQfhMpUQOy0OwvSGnRDQw   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-metering                    db9dnHslR-eHPDthFQVsVA   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-service-benchmark           yqGaNj78TKaXtucljKQ7mA   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-service-attribute           78PBD90cRRWVqr0d0URz5w   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-configuration               9vsI538_QnCScw-RF4LNbQ   3   0  ␣
↪        1            0     18.9kb         18.9kb
green  open   slipstream-job                         Iu6e2DGWQU2TZAntV_Ukxw   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-session                     J5CGY_SyREOTY9Rhm1JPOg   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-virtual-machine             s9b6i0tbRFO45S4UT_Vkcg   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-virtual-machine-mapping     1X_Fn6n2RhiKLgXdnMGzjw   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-user                        G9362RHRRgmjR_ZrrLvvKA   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-connector                   DMfNpYSATKKTbDFMzUISfQ   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-quota                       PWxlyO-zRb-c0R8EeQT8Aw   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-callback                    kqxw-TdaS2ORXg7_XuImsA   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-credential                  gQ-Ti6OnTKuKRpfoGxOBgw   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-service-offer               Qmoxk_5qT-GtcuJVbG1bVw   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-user-param                  Zxq2XAYjRyy9xnk-i7VTPw   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-event                       K5dYKP1nRkGWLAA6GKzAmw   3   0  ␣
↪        0            0       690b           690b
green  open   slipstream-external-object             oCe09WZeQb2jnL0_-iB3DQ   3   0  ␣
↪        0            0       690b           690b
```

The database should be empty except for the CloudEntryPoint, a SessionTemplate, and a Configuration. This command can be rerun without problems if you have trouble.

To avoid conflicts with the migration, we will remove those documents that have been created automatically. Execute the following commands:

```
$ curl -XDELETE http://$ES_HOST:9200/slipstream-cloud-entry-point/_doc/cloud-entry-
↪point?pretty=true
$ curl -XDELETE http://$ES_HOST:9200/slipstream-session-template/_doc/internal?
↪pretty=true
$ curl -XDELETE http://$ES_HOST:9200/slipstream-configuration/_doc/slipstream?
↪pretty=true
```

This removes those autogenerated documents, which will be replace during the migration process.

Now that the new Elasticsearch database has been prepared, you are ready to migrate documents from the old database to the new one. **To ensure that you have a coherent, all of the SlipStream services must be shutdown.** Verify that this is the case.

The organization of the documents in Elasticsearch has changed. In ES5, all the document types were stored in a single index. In ES6, each document type is in a separate index. Because of this, the migration of documents from the old database to the new one will be done document type by document type.

To reduce the repetition, you may want to create a script to make the process easier:

```
#!/bin/bash -x

DOC_TYPE=$1

if [ -n "$DOC_TYPE" ]; then
  time lein with-profile +dbcopy run -- \
       --src-host es5-01.example.com \
       --src-type $DOC_TYPE \
       --dest-host es6-01.example.com \
       --dest-index slipstream-$DOC_TYPE
fi
```

Be sure to replace the hostnames in the script with your hostnames. You can then just provide the type argument to migrate a given class of documents. We call this script `dbcopy.sh` and set execution permission with `chmod a+x dbcopy.sh`.

Now to migrate the user resources, do the following:

```
$ ./dbcopy.sh user
```

When the command finishes, you should see a message like the following:

```
18-05-15 07:14:04 ...  - finished copy documents from ["resources-index" "user" :_
↪search] - [788 788 788]
```

showing the number of documents copied. (The script will also show the elapsed time.) The numbers in the tuple should all be the same.

Repeat this process for all of the resource types in your listing of Elasticsearch indices above. You can skip some document types: for example, do not copy the “session” resources if you do not want to maintain open sessions or do not copy the “metering” resources if you do not care about past usage information.

Once the migration is complete, you can upgrade your SlipStream installation and configure the services to use the new database.

Known issues

- SlipStream version number is not correctly displayed in page footer. (See https://github.com/slipstream/SlipStreamUI/pull/783.)

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.50 - 30 April 2018

This is primarily a bug fix release that makes improvements for SlipStream administrators.

For Everyone:

- Fix resource usage page calculations to provide correct values
- Allow displaying more than 10 cloud names in the WebUI on the resource page
- Improved documentation regarding data management with ExternalObject resources
- Fix bug with state management when uploading ExternalObject resources
- Correct the ACLs on run reports

For Dave:

- Ensured presence of Python 2 in generated images.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is necessary.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.49 - 13 April 2018

External Object now has two types: generic and report. The latter one is used for storing the deployment reports. The `generic` one can be used by anyone willing to store data on clouds' Object Store. For details see.

For Everyone:

- Fix access to a metering resource details by its identifier
- CIMI connector collection is now searchable by users
- Fix User interface issues related to long usernames in logout button, breadcrumbs, and session information panel.
- CIMI filter interface: fix cursor position into input when using controlled value
- Usage page: default period, sorting of results
- Login button: separated from dropdowns for federated logins

For Dave:

- Fix number of taken entries in zookeeper which should always be equal to number of threads used by job executors
- Fix deletion of api key/secret
- Fix User registration callback when validating an email
- Service configuration is dynamically refreshed on Configuration singleton from backend
- Specify the version of nginx to be installed (in order to prevent a conflict with configuration files)

Alice, Bob, Clara, and Dave can be found here.

Migration

This release moves the configuration of the S3 backend for reports from `/opt/slipstream/server/.credentials/object-store-conf.edn` file to the `configuration/slipstream` resource.

The following migration steps are required.

1. After the upgrade of the packages make sure that elasticsearch service is running: `systemctl start elasticserach`
2. Create the following JSON file:

```
# cat configuration-slipstream.edn
{
  :id "configuration/slipstream"
  :slipstreamVersion "3.49"
  :reportsObjectStoreBucketName "<bucket-name>"
  :reportsObjectStoreCreds      "credential/<CHANGE-ME-UUID>"
  }
```

The value for `<bucket-name>` should either be taken from your previous configuration file `/opt/slipstream/server/.credentials/object-store-conf.edn` (where it is defined as `:reportsBucketName`) or you can define a new name. Note, that according to the S3 standard, the bucket name should be unique on the S3 endpoint.

The value for `:reportsObjectStoreCreds` should be the URI of the credential that you intend to be used for storing the reports of the SlipStream users. Because each credential refers to a connector, you have to make sure that the connector (and, hence, IaaS cloud) behind the credential implements and actually exposes S3 endpoint. All the connectors were updated to provide an extra configuration option `:objectStoreEndpoint`. It has to be set to a valid S3 endpoint before the persistence of the user deployment reports can be done.

3. After the configuration file is ready, run the following command to actually configure the service:

```
# ss-config configuration-slipstream.edn
#
```

4. Delete the previous configuration file:

```
# rm -f /opt/slipstream/server/.credentials/object-store-conf.edn
#
```

The configuration can always be updated via web UI by going to `https://<ss-host>/webui/cimi/configuration/slipstream` resource and editing the configuration document there.

Known issues

Due to this bug, the credential chosen for persisting the user reports should be shared with all the users of the SlipStream instance. This should be avoided though. Thus, either do not upgrade to v3.49 or apply the patch as describe below.

How to patch SS instance: Check this patch release https://github.com/slipstream/SlipStreamServer/releases/tag/v3.49.1. It provides a patched jar with the issue #1480 fixed. Please see the details on how to patch your instance there.

Next release *v3.50* will contain the fix.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI

- SlipStreamJobEngine

v3.48 - 23 March 2018

This is primarily a bug fix release that makes improvements for SlipStream administrators.

For Everyone:

- A usage page is gradually replacing the automatic usage report email. The page is internationalized.

For Clara:

- The CIMI externalObject resource has been extended to include an optional `filename` attribute, making downloads of the referenced objects easier.

For Dave:

- Add compatibility with Python 2.6 to the SlipStream bootstapping code so that images like Centos6 can be deployed.
- Fixed bug where the OpenStack connector always tried to get a floating IP even when the feature was disabled.
- When logged in as an administrator, the pages now load much more quickly.

Alice, Bob, Clara, and Dave can be found here.

Migration

Since reports are stored on S3, credentials should temporarily be set manually in `/opt/slipstream/server/.credentials/object-store-conf.edn` file, following the below format:

```
{:key                    "<KEY>"
 :secret                 "<SECRET>"
 :objectStoreEndpoint "<ENDPOINT>"
 :reportsBucketName   "<REPORTS_BUCKET_NAME>"}
```

Note that the location and format of the file have changed since the previous release.

Known issues

- When opening the usage page, the default time period will not be set until the `filter` is opened and the calendar objects are initialized.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI

- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.47 - 9 March 2018

This is primarily a bug fix release that makes improvements for SlipStream administrators.

For Everyone:

- The size of the application deployments are limited as described in the scaling guidelines.
- Fixed a problem where new users had to edit their profiles before the account could be used.

For Clara:

- The CIMI externalObject resource has been extended to include an optional `content-type` attribute, making downloads of the referenced objects easier.
- The editing process for resources through the new browser interface has been improved.

For Dave:

- The documentation has a new section about using a Docker container for SlipStream builds.
- Fixed an issue with the Job executor where it would send large numbers of useless requests to the CIMI server.
- The Nashorn library replaces the (now deprecated) PhantomJS for clojurescript unit tests.
- User roles are added to the request for API key/secret generation when provisioning VMs.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is necessary.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.46 - 23 February 2018

This release contains a few foundational features have been added (external objects, Docker connector, credential sharing) that will improve cloud resource management in the future. It also includes changes to the way machines within a deployment access the server and how deployment reports are stored. Both require administrator attention during upgrades. (See migration section.) The release also contains a number of bug fixes.

For Everyone:

- User resource implementation was changed to allow credential sharing between users and groups with ACLs.
- The login dialog was changed to avoid it being obscured on mobile devices.
- The default ACL for Connector resources was changed to allow all authentication users to see them.
- The bootstrap script has been corrected to avoid an issue where machine deployments on Ubuntu 16 machines would fail.
- The prototype for the new web browser UI has been improved to provide better editing capabilities with forms and JSON, to plot server metrics, and to render `href` attributes as links to other resources.
- Styles of cubic (new web browser UI) have been normalized to provide a consistent look and feel.

For Clara:

- The login methods of the Python API have been improved to cache credentials to make managing access easier.
- Improved the CIMI support in the Python API to allow CIMI actions to be called.
- The Python API is now part of the SlipStream RPM packages.
- A utility method was added to the Python API to retrieve deployment events.
- A function was added to the Clojure(Script) API to allow the server metrics to be retrieved.
- A prototype "cloud" connector (alpha) for Docker infrastructures is now available.

For Dave:

- The "machine" cookies that were used by VMs within a deployment to interact with SlipStream have been replaced by an API key/secret pair. These can be revoked if necessary.
- An "external object" CIMI resource has been created to allow links to external files and resources, such as report, data files, etc. Reports are now handled with these resources. (See migration below.)
- The server organization has been more finely segmented to allow for wider reuse of the servers and to make containerization easier.
- Package dependencies have be rationalized and corrected (including the `cheshire.jar` verson in the pricing service). More work on this will occur in the future to reduce the servers' footprints.
- SlipStream package dependency on `slipstream-client-clojure` (no longer created) has been removed.

Alice, Bob, Clara, and Dave can be found here.

Migration

API key/secret pairs are now being used to manage access to the server from deployed machines. For non-scalable deployments, this change will have no effect. However, scalable deployments will lose access to the server. They need to be terminated and restarted.

Below is the migration procedure to enable the view of the connector instances by users of your SlipStream instance. From now on this is required for the deployments to succeed.

- login to SlipStream instance as super user
- go to https://<slipstream>/webui/cimi/connector
- click on *magnifying glass* pictogram (this will fetch all connector config instances)
- click on a connector name link
- click on *update* button
- in the edit window add the following into the list under *"acl"* -> *"rules"*:

```
{ "principal": "USER", "right": "VIEW", "type": "ROLE" }
```

- click on *update* button to persist the configuration
- repeat this for each connector.

The method of storing reports has changed with this release. They are now stored in S3 rather than on the server's disk. This requires that the administrator have access to an S3 instance and migration of the existing reports to S3.

You must provide an S3 configuration file `/opt/slipstream/server/.aws/credentials` with the following contents:

```
aws_secret_access_key=<KEY>
aws_access_key_id=<SECRET>
aws_endpoint=<S3ENDPOINT>
```

Note that the name of the bucket is not configurable. It is set to "slipstream-reports" and must be created before being used.

Known issues

- The switch to using API key/secret pairs will only have an effect on running scalable deployments. These will need to be stopped and redeployed.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.45 - 4 February 2018

This is primarily a bug fix release, but also includes a prototype for a new web interface. Feedback on that prototype is welcome.

For Everyone:

- An SSH configuration bug that blocked SSH logins on machines without pre-existing `.ssh` directories was fixed.
- A bug with the Exoscale connector that caused deployments to fail was corrected.
- A prototype user interface has been included in the release, which is available by default on the `/webui` relative URL.

For Dave:

- The configuration for the Job Engine has been added to the quick installation script.
- CIMI resources for NuvlaBox registrations have been added.
- Unnecessary dependencies have been removed from services and packages have been cleaned up.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is necessary.

Known issues

- When upgrading rename the `/etc/default/ssclj` file to `/etc/default/cimi` if you've made changes to the configuration file.
- If you've made changes to the nginx configuration files, you will need to remove the reference to `authn.block` in `/etc/nginx/conf.d/slipstream.params`.
- The wrong version of `cheshire.jar` was included in the RPM package for the `ss-pricing` service. Replace `/opt/slipstream/ss-pricing/lib/cheshire.jar` with version 5.8.0 that can be found at `clojars.org`.
- The RPM package `slipstream-client-clojure` was not generated for this release. The v3.44 version works fine.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI

- SlipStreamJobEngine

v3.44 - 24 January 2018

This is primarily a bug fix release that makes improvements for SlipStream administrators.

For Everyone:

- Fix bug in the deployment garbage collection that caused the clean up to fail.

For Dave:

- Extend OpenNebula and NuvlaBox connectors to allow the vCPU/CPU ratio to be defined. The default value is 0.5.
- Correct Logstash and Filebeat configurations when using the installation script to avoid having Logstash logs fill with errors.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is necessary.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.43 - 22 January 2018

For Everyone:

- Remove deprecated basic authentication and related parameters from the Python API and Command Line Client.
- Fix concurrency issue with cookie handling in the Python API and Command Line Client.

For Dave:

- Mark `/etc/default/slipstream` as a configuration file to avoid having the configuration overwritten on upgrades.
- Improve template handling for the Exoscale connector so that the most recent templates are used by default. Avoids a problem with running in the DK region.
- Rationalize and reduce the default logging of the services to allow for better discovery and debugging of problems.
- Optimize the loading of the user resources from the database and provide metrics.
- Minimize connections requests from connectors on initialization.
- Refactor the deployment garbage collector to reduce object creation and database churn.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is necessary.

Known issues

- Missing dependency for the ssclj server blocks start of service.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.42 - 12 January 2018

This release brings the following changes.

The persistence of the user entities was moved from HSQLDB to Elasticsearch and from now on will be managed through CIMI server. On already deployed systems this assumes that a migration is required. Check *Migration* section below.

The following connectors were removed and are no longer supported

- StratusLab
- Azure
- NativeSoftlayer
- VCloud

Fixes and improvements:

- fixed and improved VMs information collection service.

Migration

The steps below perform migration of users from HSQLDB to Elasticsearch (via CIMI server).

Download migration script:

```
$ wget https://raw.githubusercontent.com/slipstream/SlipStreamServer/master/rpm/src/
↪main/migrations/020_migrate_users_to_cimi.py
$ chmod +x 020_migrate_users_to_cimi.py
$ yum install python-lxml
$ # or
$ pip install lxml
```

Dump users with:

```
$ export SLIPSTREAM_USERNAME=super
$ export SLIPSTREAM_PASSWORD=<password>
$ ss-login --endpoint https://<slipstream>
$ ./020_migrate_users_to_cimi.py --endpoint https://<slipstream> --get users-3.41/
```

Perform the upgrade:

```
$ yum upgrade -y
$ systemctl restart hsqldb ss-pricing ssclj slipstream \
    slipstream-job-distributor@vms_collect \
    slipstream-job-distributor@vms_cleanup \
    slipstream-job-distributor@jobs_cleanup \
    slipstream-job-executor \
    elasticsearch logstash filebeat kibana
```

In *https://<slipstream>/configuration -> SlipStream Basics -> java class names* remove any instances of the following connectors: nativesoftlayer, stratuslab, stratuslabiter, azure, vcloud. Save the configuration.

Push users back to SlipStream:

```
$ ./020_migrate_users_to_cimi.py --endpoint https://<slipstream> --put users-3.41/
```

Known issues

- No known issues.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client

- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

2.2.2 2017 Candidate Releases

v3.41 (candidate) - 2 December 2017

This release v3.41 provides more reliable information via the dashboard for everyone and improves the Clojure and Python APIs for developers.

New features and bug fixes

For Everyone:

- Dashboard has been updated to take virtual machine information from new monitoring subsystem, that is both more scalable and more reliable.
- Fix ACLs on virtual machine records to make them visible to the correct users.

For Clara:

- Updates to the Python API to allow the cloud image identifier to be obtained, to get all parameters of a modules, and to update a module.
- Updates to the Clojure API to respect the insecure? flag with the pricing, modules, and run resources, allow start/termination of runs, and to remove unnecesary XML processing.

For Dave:

- Provide a better mechanism for configuring the user migration script.
- Remove code related to old, unused authentication methods.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

- No known issues.

Commits

- SlipStream
- Server
- UI
- Connectors

- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.40 (candidate) - 10 November 2017

This release v3.40 add new job actions implementation to the new job resource as well as provides various bug fixes.

New features and bug fixes

For Everyone:

- Directly from SlipStream, create an Exoscale account or add trial credit to an existing one by using a coupon code.

For Clara:

- Add an ElasticSearch client to SlipStreamJobEngine.

For Dave:

- Add cleanup job and vms job distribution and implementation.
- Fix bad directory location of slipstream connectors configuration jar.
- Fix type of service offer attribute in Virtual machine mapping resource.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

- No known issues.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI
- SlipStreamJobEngine

v3.39 (candidate) - 4 November 2017

The v3.39 release includes a number of underlying improvements to improve the scalability and resilience of the SlipStream service. One major improvement is the deployment of a new monitoring infrastruture that will allow more rapid feedback on resource utilization. This includes a new Job resource that will allow many tasks from the API to be performed asynchronously.

New features and bug fixes

For Everyone:

- New monitoring infrastructure based on asynchronous Job resource to provide faster feedback on resource utilization and to improve reliability.
- Fixed issues with certificate validation and packaging that caused some deployments to fail.

For Bob:

- Added quota enforcement algorithm.
- Fixed missing usage collection script that causes resource usage information to not be collected.
- Fixed exception when collecting metering information that caused some information to be lost.

For Clara:

- Added support for CIMI aggregations to the Python API and refactored for obtaining credential resources.
- Ensure that operations for credential resources are correct. (The edit operation is not allowed.)

For Dave:

- Updated clojure dependencies to ensure that bug and security fixes are included.
- Remove hardcoded endpoint in Java server configuration to allow for more flexible SlipStream deployments.
- Use common application server to reduce duplicated code between servers to reduce service footprint.

Alice, Bob, Clara, and Dave can be found here.

Migration

- A migration of the user credentials is required to run the new collector service. This is currently optional but will be required in upcoming releases.
  - **Add the followings:** `DBMIGRATION_USER=<username>`

    `DBMIGRATION_PASSWORD=<password>`

    `DBMIGRATION_ENDPOINT`: e.g `http://localhost:8201/api/cloud-entry-point`

    `DBMIGRATION_OPTIONS`: defaults to `{:insecure? false}` and can be set to `{:insecure? false}`

    `DBMIGRATION_CONFIGFILE` : (optional) path to specific migration configuration file

The optional migration file is a EDN formatted file looking like:

```
{
:my-category       {:connectors    #{"my-connector1", "myconnector2", ....}
                      :template-keys [:key1 :key2 :key3 ....]}
...
}
```

where the list of keys in `:template-keys` must match the credential-template corresponding to your connectors category

- Set the *CLASSPATH* to:

  ```
  export CLASSPATH=/opt/slipstream/ring-container/lib/*:/opt/slipstream/
  ↪ssclj/lib/*
  ```

- Launch the migration script:

  ```
  java -cp $CLASSPATH com/sixsq/slipstream/ssclj/migrate/user_cred
  ```

- Install a zookeeper server (needed for the new CIMI job resource)
- A migration of CIMI server default configuration is needed */etc/default/ssclj*.
  - **Add followings:** `SLIPSTREAM_RING_CONTAINER_INIT=com.sixsq.slipstream.ssclj.app.server/init` `SLIPSTREAM_RING_CONTAINER_PORT=<SSCLJ_PORT>` `ZK_ENDPOINTS=<ZK_SERVER_IP>:<ZK_SERVER_PORT>`
  - **Remove following:** `SSCLJ_PORT`

Known issues

- Connector jar files are installed in the wrong directory. Copy links from `/opt/slipstream/ssclj/lib/ext` to `/opt/slipstream/ssclj/lib` or add the previous path to the service deployment file to work around the issue.
  - Issue SixSq/SlipStreamConnectors#115
  - Issue slipstream/SlipStreamConnectors#179
- `ss-config` command fail to connect to Elasticsearch.
  - Issue slipstream/SlipStreamServer#1285
- Insert a `virtual-machine-mapping` with a service offer fail.
  - Issue slipstream/SlipStreamServer#1287

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI

- SlipStreamPythonAPI

v3.38 (candidate) - 13 October 2017

New features and bug fixes

For Alice:

- Fix issue with credentials for users that have long usernames from External authentication methods (e.g. eduGAIN or Elixir).
- Treat entitlements coming from the Elixir AAI federation as SlipStream roles.

For Bob:

- Attach pricing information to metering resource to allow approximate cost estimates to be provided.

For Clara:

- Provide links in the "reference" section to the specific API documentation (Clojure, Python, and Libcloud) in the main SlipStream documentation.
- Improve the documentation for using the API key/secret pairs through the API.
- Allow users to supply their own server token when using the Clojure API.
- Support Debian for package installation.
- Add an `ss-terminate` command and ensure that `ss-login` and `ss-logout` are packaged.

For Clara and Dave:

- Initial implementation of Job resource to allow for asynchronous actions on the server.

For Everyone:

- Fix an issue with updating the internal ACL representation when editing resources, which affected the accuracy of search requests.
- Fix an issue with some cloud connectors to avoid collisions (and failures) when creating SSH key resources.
- Add support for private network addresses for the Open Telekom Cloud.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

- No known issues.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI

v3.37 (candidate) - 1 October 2017

New features and bug fixes

For Clara:

- Improve the error messages when trying to upload CIMI resources that do not follow the defined resource schema.
- Provide a ServiceBenchmark resource that allows users to post performance and reliability information concerning cloud resources and services.
- Adapt language-specific libraries to use “Session” resources for authentication, allowing also the use of API key/secret pairs.
- Provide a Libcloud driver for SlipStream. See the documentation for details.

For Bob:

- Implement new resource usage and metering scheme to provide flexible mechanism for usage and billing reports. This involves the new VirtualMachine, Quota, and Metering resources.
- Provide an initial implementation of a cloud credentials resource that will eventually permit sharing of credentials between users.

For Everyone:

- Add a wait for a routable IP address in the node executor to avoid unnecessary deployment failures.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

- No known issues.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI

v3.36 (candidate) - 8 September 2017

New features and bug fixes

For Clara:

- **Server:**
  - First version of the CIMI VirtualMachines resource

For Dave:

- **Client:**
  - The node executor now start only after a valid network configuration is available on OS with SystemD.
- **Server:**
  - Install Zookeeper together with SlipStream

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

- The new CIMI VirtualMachines resource will not be populated by the server because of a schema issue.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client

- SlipStreamClojureAPI
- SlipStreamPythonAPI

v3.35 (candidate) - 25 August 2017

New features and bug fixes

For Alice:

- **UI:**
  - The usage page is now rendered correctly
  - Weekly and monthly usages have been removed

For Clara:

- **Server:**
  - Implementation of the *$aggregation* query parameters on CIMI resources

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

There are no known issues with this release.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI

v3.34 (candidate) - 11 August 2017

New features and bug fixes in v3.34

For Alice:

- **UI:**
  - Fixed minor typo in help hint

- **Connectors:**
  - Fixed the disk resizing for VMs in the OTC provider

For Dave:

- **Client:**
  - Use Python's "requests" lib instead of "httplib2"
  - Allow deployments from users having usernames with special characters
- **Server:**
  - Reduce memory consumption of ElasticSearch if it is installed locally
  - On deploy, do not use service-offer if it is empty
  - Fixed memory leak when using ElasticSearch client
  - Improve error logging
  - Implementation of API key credentials

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

There are no known issues with this release.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI

v3.32 (candidate) - 14 July 2017

New features and bug fixes in v3.32

For Alice:

- **UI:**
  - On the "Deploy" dialog, the Cloud you selected will stay selected after a service offers refresh even if it's not the cheapest one

  - Allow to enter the amount of RAM as a float

For Clara:

- **CIMI resources:**
  - `$orderby` query parameter now support sorting by fields containing `:` (colon) character

For Dave:

- **Client:**
  - Service Offers scrapers delete only obsolete service offers of type `VM`
- **Server:**
  - Allow to internally create account with special characters in username (for external auth)
  - PRS now use one query per Node per Cloud. Each query return maximum one element. Queries are threaded.
  - PRS only search service offers of type `VM`

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

There are no known issues with this release.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI

v3.31 (candidate) - 30 June 2017

New features and bug fixes in v3.31

For Alice:

- **UI:**
  - If PRS is available use by default the fields CPU, RAM, Disk to find the most appropriate service offers

    - CPU, RAM, Disk values can be changed from the deployment dialog

- **Server:**
    - Various enhancements to the PRS service
    - Improved authentication with federated identity
    - Separated OIDC and Cyclone authentication methods

For Clara:

- **Python API:**
    - Improved error handling of CIMI resources in SlipStreamPythonAPI

For Dave:

- **Client:**
    - Reduced the size of the SlipStreamClient tarball
- **Connectors:**
    - Added service offers scraper to connectors
- **Server:**
    - Improved logging for CIMI resources
    - Added ability to start a deployment with service offers

Alice, Bob, Clara, and Dave can be found here.

Migration

Service offers schema has been changed. To use PRS, please delete all service offers and regenerate them with `*-service-offers` commands (eg: `openstack-service-offers`)

Known issues

There are no known issues with this release.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI

v3.30 (candidate) - 23 June 2017

This release is an internal release. Please look at the release notes of v3.31.

v3.28 (candidate) - 21 May 2017

New features and bug fixes in v3.28

For Alice and Clara:

- **Server:**
  - Various updates and fixes around authentication: adding sessions, fixing eduGAIN workflow and OIDC.
- **Client:**
  - Fix: accept parameter values containing = sign.
- **Client API:**
  - Added functions for CIMI resources.
- **Connectors:**
  - EC2: added support for extra disk.
  - **OpenStack:**
    - fixed leaking of private IPs.
    - improved retrieval of IPs on OpenStack.

For contributors:

- Improved contributor documentation around setting up Python environment.
- Improved conditional building of RPMs on systems with no rpmbuild installed.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

- OpenStack connector fail to deploy a component (application deployment work).

Commits

- SlipStream
- Server
- UI
- Connectors

- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI

v3.27 (candidate) - 8 May 2017

New features and bug fixes in v3.27

This release v3.27 improves the implementation of the internal SlipStream inter-service communication implementation, unifies the implementation of the users' authentication code as well as provides various bug fixes.

For Dave:

- Introduced installation of Metricbeat with SlipStream. This provides the OS level monitoring and storage of the metrics to Elasticsearch for later visualization with Kibana.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

There are no known issues with this release.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI

v3.26 (candidate) - 21 April 2017

New features and bug fixes in v3.26

This release v3.26 brings updates to EC2 connector. The release also includes a couple of other fixes and clean ups of the underlying code. Read below for more details.

For Clara and Alice:

- EC2 connector: added new instance types and regions; added support for extra disk; updated to the latest version of *boto*.

- OCCI connector was removed.
- Updated help messages and fixed an issue with HTTP redirection in SlipStream CLI.
- Fixed outdated links in the Web UI Tour.

For Dave:

- Installation of SlipStream server installs full ELK stack for collection of the logs from different components of the service.
- SlipStream server logs were moved to /var/log/slipstream/server.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

When accessing the v3.26 server with an old cookie that is still valid, the server responds with an internal server error (500). See old cookie causes internal server error. The issue is already fixed in master branch and the updated RPM with the fix is available from *SlipStream-Snapshots-** repos. The next release will contain the fix by default.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI

v3.25 (candidate) - 7 April 2017

New features and bug fixes in v3.25

Version v3.25 fixes a problem where the server could effectively hang when accessing resources in the underlying database. The release also include a couple other fixes and clean ups of the underlying code.

For everyone:

- Diagnose and fix an issue with the underlying database that caused the service to hang.
- Fix broken links in the deployment dialog (to SSH configuration) and in the tour (to external documentation).
- Fix the Kubernetes deployment in the App Store.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

There are no known issues with this release.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI

v3.24 (candidate) - 26 March 2017

New features and bug fixes in v3.24

Version v3.24 allows the OpenNebula cloud connector to resize the root disk of virtual machines and enhances the user management capabilities of the SlipStream Python API. Several bugs have been fixed and some foundational changes have been made to improve consistency and reliability.

For everyone:

- Improve the SlipStream OpenNebula and NuvlaBox cloud connectors to allow them to resize the root disk of a virtual machine.
- Fix a problem where the pricing service would hang, causing the deployment dialog to wait for a timeout.
- Ensure that the SlipStream client is only installed under Python 2.7+, not under Python 3.x (which isn't supported).
- The SlipStream client can now the use the "disk" generic cloud parameter.

For SlipStream administrator [Dave]:

- Improve management of users through SlipStream Python API.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

There are no known issues with this release.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI
- SlipStreamPythonAPI

v3.23 (candidate) - 14 March 2017

New features and bug fixes in v3.23

Version v3.23 makes some foundational changes for future improvements, improves the OpenNebula connector, makes the bootstrap process more reliable, and fixes a few bugs.

For everyone:

- Improve the SlipStream VM bootstrap process to better handle environments where Python 3 is the default (e.g. Ubuntu 16.04).
- Improve the OpenNebula connector to allow both OpenNebula native contextualization and cloud-init contextualization.
- Fix hybrid cloud option in the deployment dialog which would prevent the deployment of the application.
- Made foundational changes on the server and UI that will allow a workflow more focused on cloud service provider offers in the future.

For SlipStream administrator [Dave]:

- Improve handling of certificates for generating authentication tokens.
- Fix startup failure of Riemann server.
- Add missing file in the server backup RPM package.

Alice, Bob, Clara, and Dave can be found here.

Migration

1. IMPORTANT. Certificates for generation of authentication tokens are no longer password-protected. The new unencrypted certificates will be generated under `/etc/slipstream/auth` as part of post-install script of `slipstream-ssclj` RPM. Next time when RPM gets updated the files will not be overwritten. You can update them at your will (check */opt/slipstream/ssclj/bin/generate-auth-keys.sh*). Only one service `ssclj.service` requires private key for encrypting the authentication token. All other services require only public key for decryption. Locations of both can be configured in their respective `systemd` configuration files or in the respective `/etc/default/<service>` files.

2. The schema for the server configuration has changed. You will need to remove the "PRS Endpoint" and "PRS Enabled" parameters from the configuration before starting the updated service. First, save the current configuration into a file:

   ```
   ss-config -r configuration/slipstream > config-ss.edn
   ```

   Edit `config-ss.edn` and delete `:prsEndpoint` and `:prsEnable` key/value pairs from the configuration file. Then, upload the updated configuration back to DB with:

   ```
   ss-config config-ss.edn
   ```

Known issues

There are no known issues with this release.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI

v3.22 (candidate) - 24 February 2017

New features and bug fixes in v3.22

Version v3.22 provides improvements aimed primarily at SlipStream administrators. The major change being an upgrade from Elasticsearch 2.x to 5.x.

For SlipStream administrator [Dave]:

- Upgrade of Elasticsearch to v5.x to take advantage of database improvements.
- Fix broken packaging for OTC and Azure connectors that prevented upgrades.
- Refactor placement and pricing service (PRS) to simplify the service and to improve the logging of errors.

Alice, Bob, Clara, and Dave can be found here.

Migration

The version of Elasticsearch being used by SlipStream has changed to Version 5.

Migration of SlipStream database for Elasticsearch 5 is NOT required.

Manual upgrade of Elasticsearch plugins is required. Here it's shown on an example of S3 snapshot plugin:

```
systemctl stop elasticsearch
/usr/share/elasticsearch/bin/elasticsearch-plugin remove cloud-aws
echo y | /usr/share/elasticsearch/bin/elasticsearch-plugin -s install repository-s3
systemctl start elasticsearch
```

Known issues

There are no known issues with this release.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI

v3.21 (candidate) - 10 February 2017

New features and bug fixes in v3.21

Version v3.21 is primarily a bug fix release.

For everyone:

- FIX: Failure when installing packages should abort deployment.
- FIX: Fix missing dependency for pricing and ranking service that caused the service not to start.
- FIX: Problem with user interface changes that caused deployments to fail.

For application developers [Clara]:

- Move Riemann server package, used for autoscaling applications, to the Community Edition.

For SlipStream administrator [Dave]:

- Simplify the organization of Community and Enterprise releases to make building and deploying SlipStream easier.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

The packages for the OTC and Azure connectors to not upgrade cleanly. You can work around this by deleting the connector packages and then installing the new packages after the rest of the system has been updated.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI

v3.20 (candidate) - 28 January 2017

New features and bug fixes in v3.20

Version v3.20 allows better management of SlipStream from other services as well as bug and security fixes.

For everyone:

- Add m2.2xlarge instance type for the Amazon cloud service.
- Add checkbox to highlight option for multi-cloud deployment.

For application developers [Clara]:

- Allow managers to create and to manage a group of users.
- FIX: Default is now taken into account when saving nodes in deployment

For SlipStream administrator [Dave]:

- Bug and security fixes.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI

v3.19 (candidate) - 16 January 2017

New features and bug fixes in v3.19

Version v3.19 is a maintainence release that incorporates dependency upgrades with bug and security fixes.

Migration

No migration is required.

Known issues

No known issues.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI

2.2.3 2017 Stable Releases

v3.27 (stable) - 23 May 2017

Features

Below is the list of the main features and improvements done in releases 3.15 to 3.27.

- Introduced a different approach to the service configuration. Now the configuration can be managed with the help of the extra tooling directly in the Elasticsearch DB.
- Service Catalog displays prices consistently in EUR.

- Added support of CPU/RAM/Disk server side as module parameters.
- Introduced new Python CLI and API to SlipStream service like nuv.la.
- Made Service Catalog available in the community version.
- Added new connector for Open Telecom Cloud https://cloud.telekom.de/ named OTC.
- Introduced a number of improvements to editing of Application module and JSON rendering of all module types and run.
- Improved OpenNebula connector and allow it to resize the root disk of VM.
- Improved EC2 connector.
- Removed OCCI connector.
- Improved the implementation of the internal SlipStream inter-service communication implementation, unified the implementation of the users' authentication code.

The detailed change log is given below. For brevity bug fixes have not been included, see the change logs for the intermediate releases for the full set of changes and fixes.

For application users and developers [Alice, Clara]:

- Users can now enter CPU/RAM/Disk sizes for the component instances in the generic Cloud Configuration -> Cloud section on the components. Depending on the cloud (working with t-shirt sizes or directly with CPU/RAM/Disk), these values will be mapped either directly to the corresponding CPU/RAM/Disk or the closest match to the t-shirt size will be made. The mapping is done using service offers defined the Service Catalog.
- New Python CLI and API were released to be used with SlipStream services like nuv.la. For more details please see CLI and API.
- Add m2.2xlarge instance type for the Amazon cloud service.
- Add checkbox to highlight option for multi-cloud deployment.
- Improve the SlipStream VM bootstrap process to better handle environments where Python 3 is the default (e.g. Ubuntu 16.04).
- Improve the OpenNebula connector to allow both OpenNebula native contextualization and cloud-init contextualization.
- Made foundational changes on the server and UI that will allow a workflow more focused on cloud service provider offers in the future.
- Improve the SlipStream OpenNebula and NuvlaBox cloud connectors to allow them to resize the root disk of a virtual machine.
- EC2 connector: added new instance types and regions; added support for extra disk; updated to the latest version of *boto*.
- OCCI connector was removed.

For application developers [Clara]:

- Enabled editing of Pre/Post-Scale scripts in *Application Workflows* tab of components. For details, please see Scalability Workflow Hooks section of the SlipStream tutorial on running scalable applications.
- Improved modification of application component.
- Allowed the possibility to edit the description and category of input/output parameters on components.
- Added JSON rendering for module type resources (project, component, application) and run.
- Allow managers to create and to manage a group of users.

For administrators [Dave]:

- New way of managing the service configuration via configuration files and *ss-config* utility. See documentation.
- Introduced installation of Metricbeat with SlipStream. This provides the OS level monitoring and storage of the metrics to Elasticsearch for later visualization with Kibana.

For organization manager and SlipStream administrator [Bob and Dave]:

- New connector named OTC for Open Telecom Cloud.

The Alice, Bob, Clara, and Dave personae can be found on the SixSq website.

Migration

Multiple migrations are required between 3.14 and 3.27:

1. 3.14 -> 3.15
2. 3.15 -> 3.16
3. 3.21 -> 3.22
4. 3.22 -> 3.23

For details see the release notes for each of the corresponding candidate releases. The migrations should be applied in the order defined above. Upgrades of the SlipStream packages should be carried out carefully step by step from the release to release that require migration. To accomplish this, one has to explicitly define the version numbers of the packages. For example:

```
# List available version numbers
$ yum --showduplicates list slipstream-server
# Select the version number you are upgrading to and run
$ yum upgrade slipstream-*3.15-0.el7
# Apply migration.
# Repeat.
```

Starting from release v3.21, you should explicitly exclude packages with *enterprise* and *community* in their names. E.g.,:

```
$ yum upgrade slipstream-*3.21-0.el7 --exclude=*-enterprise \
  --exclude=*-community
```

Commits

- Server
- UI
- Client
- Connectors
- Documentation

2.2.4 2016 Candidate Releases

v3.18 (candidate) - 17 december 2016

New features and bug fixes in v3.18

v3.18 is a maintenance release.

Migration

No migration is required.

Known issues

Instance type chosen by placement and ranking service (based on the component global CPU/RAM/Disk definition) and displayed in the component Deploy dialog is ignored, and the instance type defined for the cloud on the component is used instead.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI

v3.17 (candidate) - 09 december 2016

New features and bug fixes in v3.17

Version v3.17 brings new connector for Open Telecom Cloud https://cloud.telekom.de/ named OTC, a number of improvements to editing of Application module and JSON rendering of all module types and run.

For application developers [Clara]:

- Improved modification of application component.
- Now it's possible to edit the description and category of input/output parameters on components.
- Added JSON rendering for module type resources (project, component, application) and run.
- CIMI filter can now handle "!=" operator.
- Various minor improvements in the code organization for OpenStack connector and SlipStream Client.

For organization manager and SlipStream administrator [Bob and Dave]:

- New connector named OTC for Open Telecom Cloud.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is required.

Known issues

Instance type chosen by placement and ranking service (based on the component global CPU/RAM/Disk definition) and displayed in the component Deploy dialog is ignored, and the instance type defined for the cloud on the component is used instead.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI

v3.16 (candidate) - 21 november 2016

New features and bug fixes in v3.16

The main features of the release v3.16 are addition of the support of CPU/RAM/Disk server side as module parameters and introduction of new Python CLI and API to SlipStream service like nuv.la. Service Catalog was made available in the community version.

For application users and application developers [Alice, Clara]:

- Users can now enter CPU/RAM/Disk sizes for the component instances in the generic Cloud Configuration -> Cloud section on the components. Depending on the cloud (working with t-shirt sizes or directly with CPU/RAM/Disk), these values will be mapped either directly to the corresponding CPU/RAM/Disk or the closest match to the t-shirt size will be made. The mapping is done using service offers defined the Service Catalog.
- New Python CLI and API were released to be used with SlipStream services like nuv.la. For more details please see CLI and API.

Alice, Bob, Clara, and Dave can be found here.

Migration

Upgrading to v3.16 requires each connector to be described by a corresponding service offer. To insert the service offer for a new connector, use the REST API to post on this resource. For example, for a connector named *connector-name1*, if ssh access to API server is available: - *curl -X POST -H "slipstream-authn-info: username role" -H "content-type: application/json" http://localhost:8201/api/service-offer -d@service-offer.json*

The service-offer.json should have the following structure:

```
#
{
  "connector" : {
    "href" : "connector-name1"
  },
  "schema-org:flexible" : "true",
  "acl" : {
    "owner" : {
      "type" : "ROLE",
      "principal" : "ADMIN"
    },
    "rules" : [ {
      "principal" : "USER",
      "right" : "VIEW",
      "type" : "ROLE"
    }, {
      "principal" : "ADMIN",
      "right" : "ALL",
      "type" : "ROLE"
    } ]
  },
  "resourceURI" : "http://sixsq.com/slipstream/1/ServiceOffer"
}
#
```

Without SSH access to the API, the same command can be re-written with

- *curl -X POST -H "content-type: application/json" http[s]://slipstream-endpoint/api/service-offer -d@service-offer.json* -b token.txt

(see SlipStream API documentation on how to obtain an authentication token).

It is possible to check that a given connector named *connector-name-x* is described by a service offer by querying the Service offer resource with the following command: *curl -H "slipstream-authn-info: super ADMIN" "http://localhost:8201/api/service-offer?$filter=connector/href='connector-name-x'"*

Known issues

Instance type chosen by placement and ranking service (based on the component global CPU/RAM/Disk definition) and displayed in the component Deploy dialog is ignored, and the instance type defined for the cloud on the component is used instead.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI

v3.15 (candidate) - 24 october 2016

New features and bug fixes in v3.15

Version v3.15 changes the approach to the service configuration bringing SlipStream closer to an ability to run the service in a distributed mode by decoupling the service state (including service's bootstrap configuration) from the processes running the business logic.

For application users and application developers [Alice, Clara]:

- Fixes and improvements in displaying placement and pricing information in application Deploy dialog.
- Improved retrieval of VM instance ID and IP to provide VM to run mapping in failed runs.
- Fixed issue with linking to output parameter of parent image.
- Consistently display prices in Service Catalog in EUR.

For application developers [Clara]:

- Enabled editing of Pre/Post-Scale scripts in *Application Workflows* tab of components. For details, please see Scalability Workflow Hooks section of the SlipStream tutorial on running scalable applications.

For administrators [Dave]:

- New way of managing the service configuration via configuration files and *ss-config* utility. See documentation.

Alice, Bob, Clara, and Dave can be found here.

Migration

Migration is needed from v3.14 to v3.15. As the result of the migration the service and cloud connectors configuration information will be moved from HSQLDB to Elasticsearch.

1. Declare downtime.
2. Let SlipStream service running.
3. Download the service configuration as XML:

```
$ curl -k -s -D - https://<slipstream>/auth/login -X POST -d \
    "username=super&password=<PASS>" -c cookie-user.txt
$ curl -k -b cookie-user.txt 'https://<slipstream>/configuration?media=xml' \
    -H "Accept: application/xml" -o configuration.xml
```

4. Update ssclj and connector packages:

```
$ yum update slipstream-ssclj-enterprise
$ yum update slipstream-connector-*
```

5. Perform the migration of service configuration:

```
$ export ES_HOST=localhost
$ export ES_PORT=9300
$ ss-config-migrate -x configuration.xml -m 3.14=3.15
$ # Use -m old=new to update values of the parameters if needed.
$ # Example: -m localhost=127.0.0.1 -m smtp.gmail.com=smtp.example.com
```

Now you are ready to upgrade other SlipStream packages:

```
$ yum update --disablerepo=* --enablerepo=SlipStream-<release>-<kind>
```

Substitute `<release>` and `<kind>` according to your installation.

Check `/opt/slipstream/server/etc/default.slipstream.rpmsave` file for your custom configurations and merge them with the new ones coming with `/opt/slipstream/server/etc/default.slipstream`.

Restart services:

```
$ systemctl restart hsqldb elasticsearch ssclj slipstream
```

Known issues

On enterprise edition, due to a bug in the UI part of the deployment placement and ranking, the Deploy dialog (for application or component) may display a certain choice of the cloud/price offer, but after clicking the Deploy button, the application/component may be deployed to a different cloud. This was fixed in 3.16.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClientAPI

v3.14 (candidate) - 7 october 2016

New features and bug fixes in v3.14

Version v3.14 adds the delete all versions for a module, and fixes some issues related to connectors.

For application users [Alice]:

- Add the delete all versions for a module

For application developers [Clara]:

- Fix ssh private key management to build image on StratusLab connector
- Selection of specific network for opennebula connector

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is needed from v3.13 to v3.14.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClojureAPI

v3.13 (candidate) - 28 September 2016

New features and bug fixes in v3.13

Version v3.13 fixes a bug in build image creation, and brings minor improvement in REST API.

For application users and developers [Alice, Clara]:

For application users [Alice]:

- Fix a bug for Safari users that prevented display of some pages with pagination
- Fix a bug in StratusLab connector that prevented the build of an image

For application developers [Clara]:

- Add USER and ANON roles for logged in users (used to query REST api)
- Refactor the parsing of running instances

For administrators [Dave]:

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is needed from v3.12 to v3.13.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClientAPI

v3.12 (candidate) - 13 September 2016

New features and bug fixes in v3.12

Version v3.12 improves build system and fixes some stability issues.

For application users and developers [Alice, Clara]:

- Improves readability of failing unit tests
- Increase the allowed maximum size of a report
- Fix incorrect identifier for configuration resources
- Fix: Use namespaced attributes for Riemann monitoring of connectors
- Pass SNI information to backend services

For administrators [Dave]:

- Unify build system with boot for clojure code
- Fix usage consolidations (adaptation of build configuration following boot adoption)
- Fix collector async job when only users with no connectors configures online

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is needed from v3.11 to v3.12.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClientAPI

v3.11 (candidate) - 26 August 2016

New features and bug fixes in v3.11

Version v3.11 is a preparatory release that provides much of the groundwork for future improvements. The emphasis has been on preparing new server-side resources for cloud connectors and service configuration; these will improve the management of these resources in the future. There has also been significant work done to streamline the code organization, packaging, and release process. This should speed development of new features.

For application users and developers [Alice, Clara]:

- Fix issue with pricing server that prevented prices from being calculated.

- Alpha versions of connector and configuration resources. These are available through the API and will be integrated into the web interface in a future release.

For administrators [Dave]:

- Upgrade to the latest production libraries for all server dependencies, improving the robustness of the server (in particular Aleph, Buddy, and ClojureScript).
- Correct the systemd configuration for the ssclj service so that successful shutdowns are not marked as failures.
- Clean up and reorganize the packaging for the pricing service. Logging information will now appear in the standard OS directory.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is needed from v3.10 to v3.11.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClientAPI

v3.10 (candidate) - 13 August 2016

New features and bug fixes in v3.10

Version v3.10 provides a complete set of resources for the Service Catalog (Enterprise Edition), allowing policy (and priced) based placement of virtual machines. This release also provides a complete Clojure and ClojureScript API for the SlipStream CIMI resources.

For application users and developers [Alice, Clara]:

- Provide complete set of service catalog resources (serviceOffer, serviceAttribute, and serviceAttribute-Namespace) to allow policy-based placement using the service catalog information. (Enterprise Edition)
- Provide clojure/clojurescript API for SlipStream CIMI resources. The API provides asynchronous and synchronous implementations of all SCRUD actions. Filtering and subsetting are provided for search operations.
- Use larger modal dialog to avoid truncating long parameter or component names in run dialog.

For administrators [Dave]:

- Modify service dependencies to ensure cleaner start up of all SlipStream services on boot.
- Improve the collection of virtual machine state information (used in the dashboard) to make it more efficient and reliable. Put in additional logging to make debugging easier.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is needed from v3.9 to v3.10.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClientAPI

v3.9 (candidate) - 3 August 2016

New features and bug fixes in v3.9

Version v3.9 is an incremental release that further improves the functionality of the placement and ranking service. This is an alpha-level Enterprise feature. This release also contains improvements and fixes for both the Community and Enterprise Editions.

For application users and developers [Alice, Clara]:

- Provide pricing along with a filtered set of connectors on the run dialog. (Enterprise Edition, alpha)
- Resolve an issue with the CIMI filter grammar that caused the parsing to take several seconds. After the fix, the parsing takes a few milliseconds.
- Improve the bootstrapping process to avoid having the process hang on CentOS 6 systems.
- Fix a regression that prevented run tags from being saved.
- Fix an issue where ghost nodes would appear in the run if their names matched the regex for a node instance.
- Fix an issue with redirects on authentication that prevented logging in.

For application users [Alice]:

- Provide a better message when a cloud quota has been exceeded. The message now includes the quota, number of running VMs, and number of requested VMs.

For application developers [Clara]:

- Allow application developers to specify a placement policy for application components, for example, limiting the places where a component can run. (Enterprise Edition, alpha)
- Improve the error messages reported to users of the SlipStream client API, providing more information about the underlying cause of a problem.

For administrators [Dave]:

- Streamline the installation of SlipStream with a packaged version of PhantomJS and with a package for the Elasticsearch repositories.

Alice, Bob, Clara, and Dave can be found here.

Known Issues

- The process that collects information abouts users' virtual machines can become saturated, resulting in the loss of this information for most users. When this issue appears, the slipstream service can be restarted to return it to a normal state.

Migration

No migration is needed from v3.8 to v3.9.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClientAPI

v3.8 (stable) - 15 July 2016

New features and bug fixes in v3.8

Version v3.8 is a consolidation release that fixes some issues regarding packaging and installation. It also brings some enhancements to bootstrap mechanism.

For application users and developers [Alice, Clara]:

- Inherited output parameters are visible to the users, allowing an input parameter to be mapped to an inherited output parameter.
- The SlipStream bootstrap process is now able to run on operating system with only Python 3 installed. The robustness of the bootstrapping process has also been improved.
- Display prices for running components and applications and certain clouds in the run dialog.
- Exoscale: Add support for Mega and Titan instances.
- OpenStack: Added support for Floating IPs.
- OpenNebula: Added default values for image parameters

For application developers [Clara]:

- Allow the client API to be used for test instances of SlipStream that use a self-signed certificate.

For administrators [Dave]:

- Fix an issue with the SlipStream installation process where connector installations would fail because of package name matching in the yum repository.

Alice, Bob, Clara, and Dave can be found here.

Known Issues

- The process that collects information abouts users' virtual machines can become saturated, resulting in the loss of this information for most users. When this issue appears, the slipstream service can be restarted to return it to a normal state.

Migration

No migration is needed from v3.7 to v3.8.

Commits

- SlipStream
- Server
- UI
- Connectors
- Client
- SlipStreamClientAPI

v3.7 (candidate) - 1 July 2016

New features and bug fixes in v3.7

Version v3.7 is a consolidation release that fixes some issues regarding packaging. It also brings enhancement to bootstrap mechanism.

For application users and developers [Alice, Clara]:

- FIX: Correct a problem where components could not be selected during application creation
- Make the bootstrap mechanism more reliable over low-quality networks (e.g. satellite connections)

For administrators [Dave]:

- Avoid dependency version conflicts by removing hard-coded dependencies for the PRS-lib component.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is needed from v3.6 to v3.7.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v3.6 (candidate) - 21 June 2016

New features and bug fixes in v3.6

The primary goal of v3.6 is to fix known issues of v3.5. It also puts in place the infrasture required for Placement and Ranking service.

For everyone [Alice, Bob, Clara, Dave]:

For application users and developers [Alice, Clara]:

- FIX: Fix the mapping resolution between a VM and a Run
- FIX: Cloud usages are now visible in the web server.
- FIX: Consolidation and daily sending of usage emails.
- FIX: Service catalog uses the service-offer resource.
- Allow to define relative and absolute path for module logo

For application developers [Clara]:

- FIX: CloudEntryPoint resource is now accessible.
- FIX: Correct CIMI edit responses

For administrators [Dave]:

- Rationalize logging and logging levels
- Remove http-kit support
- Cleanup unused libraries

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is needed from v3.5 to v3.6.

Commits

- Server
- UI
- Client
- Connectors

- Documentation

v3.5 (candidate) - 3 June 2016

New features and bug fixes in v3.5

The primary feature for v3.5 is the introduction of Elasticsearch for data persistency. This should make the service more stable and drastically improve response times for retrieving event and usage information.

For everyone [Alice, Bob, Clara, Dave]:

- Provide a top-level support link for users, if the system administrator has set a support email address.
- In the Enterprise Edition, improve the visualization of the Service Catalog entries and allow more than one entry per cloud connector.
- FIX: Community Edition dashboard no longer displays errors related to service catalog entries.
- FIX: Ensure build execution scripts (pre-install, packages, and post-install) only run during the build phase.
- FIX: Ensure OpenStack connector works correctly on cloud infrastructures that done use the "default" domain.

For application users and developers [Alice, Clara]:

- Improve the retry mechanism for the SlipStream clients to make them behave more uniformly and to be more robust.

For application developers [Clara]:

- Update API documentation for cookie authentication. Cookie authentication is now the preferred method; basic authentication is deprecated.
- Add a command to allow the reports from a run to be retrieved.

For administrators [Dave]:

- Make the installation script more robust concerning RPM package names.
- Improve the configuration of the nginx configuration to enhance the security of the service.
- FIX: Ensure that all services are enabled in systemd so that they restart on reboot.
- FIX: Missing file in Riemann service that caused startup to fail.
- FIX: Mark `/etc/hsqldb.cfg` as a configuration file to avoid losing local changes.
- FIX: Reducing reliance on hsqldb should reduce instabilities when running the `ssclj` service.

Alice, Bob, Clara, and Dave can be found here.

Known Issues

- Configuration files are required to build software. (GitHub Issue 277)
- Logs for the ssclj service are in the wrong location. (GitHub Issue 737)
- CloudEntryPoint resource is not accessible. (GitHub Issue 738)
- The `/usage` resource hangs. (GitHub Issue 618)
- The admin users `/usage` does not render on Safari (GitHub Issue 619)

Migration

Elasticsearch is now required for the SlipStream service. When upgrading, Elasticsearch will need to be installed, configured, and started by hand. Start by adding the Elasticsearch repository:

```
$ yum install slipstream-es-repo-community
```

Use "community" or "enterprise" as appropriate for you installation.

Install Elasticsearch:

```
$ yum install elasticsearch
$ systemctl daemon-reload
$ systemctl enable elasticsearch.service
```

Update the configuration:

```
$ cd /etc/elasticsearch/
$ mv elasticsearch.yml elasticsearch.yml.orig
$ cat > elasticsearch.yml <<EOF
network.host: 127.0.0.1
EOF
```

And finally start the service:

```
$ systemctl start elasticsearch.service
```

You can test that Elasticsearch is running correctly with:

```
$ systemctl status elasticsearch.service
$ curl http://localhost:9200/_cluster/health?pretty=true
```

The first should show that the service is running and the second should provide the health of the Elasticsearch cluster. It should contain one node and be in a "green" state.

For data persistency, SlipStream is moving from hsqldb, a Java-based SQL relational database, to Elasticsearch, a high-performance, document-oriented data store. The migration from one to the other will be incremental, so during the transition, both databases will be used. This is the first release where Elasticsearch is used.

Before starting the migration procedure, please make sure that `slipstream` and `ssclj` are not running. Both databases (hsqldb and Elasticsearch) must be running.

Then you can migrate the resources with the following commands:

```
$ export ES_HOST=localhost
$ export ES_PORT=9300
$ java -cp /opt/slipstream/server/webapps/slipstream.war/WEB-INF/lib/clojure-1.8.0.
↪jar:/opt/slipstream/ssclj/lib/ssclj.jar com.sixsq.slipstream.ssclj.migrate.script
```

Resources are migrated (from hsqldb to elastic search) by batches of 10‘000 documents. Example of output of this script:

```
...
Creating ES client
Index resetted
Will create korma database with db-spec
...
Migrating  usage , nb resources = XXX
```

(continues on next page)

(continued from previous page)

```
Migrating usage 0  ->  9999
...
Migrating  usage-record , nb resources = XXX
Migrating usage-record 0  ->  9999
...
Migrating  event , nb resources = XXX
Migrating event 0  ->  9999
...
```

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v3.4 (candidate) - 23 May 2016

New features and bug fixes in v3.4

NOTE: This release provides a fix for v3.3 and introduces the previously rolled back features and bug fixes of v3.3. For the details of v3.3 release please see the corresponding announcement section below.

For everyone [Alice, Bob, Clara, Dave]:

- The main feature of 3.4 release is introduction of on/off-line status reporting for NuvlaBox.

For application users and developers [Alice, Clara]:

- Fixed disk size unit in describe instance action in OpenNebula connector.

For application developers [Clara]:

- Please follow the migration procedure on SlipStream Enterprise for NuvlaBox connectors.
- DELETE on API resources now returns 200 instead of 204.
- API documentation was updated to match the latest API implementation.

Alice, Bob, Clara, and Dave can be found here.

Known Issues

- Riemann service jar is missing `service_offer.clj` which causes startup to fail. (GitHub Issue 5)
- Local changes to the file `/etc/hsqldb.cfg` will be lost because it isn't marked as a configuration file in the RPM package. (GitHub Issue 37)
- Build execution scripts (pre-install, packages, and post-install) are re-executed even when an image has been built, causing deployment failures. (GitHub Issue 274)
- Instabilities when running the `ssclj` service with the hsqldb database. This may cause the SlipStream service to stop responding and restart of the hsqldb database may not be possible. (GitHub Issue 725)

- OpenStack connector does not properly deploy applications on OpenStack cloud infrastuctures that do not use the "default" domain. (GitHub Issue 107)
- Community Edition dashboard displays errors when trying to access the (Enterprise-only) service catalog. (GitHub Issue 615)
- Configuration files are required to build software. (GitHub Issue 277)

Migration

The following migration is required on SlipStream Enterprise instance.

In this release the Riemann service was introduced. It is intended to be used with NuvlaBox product.

If you are using or intending to start using NuvlaBoxes with SlipStream Enterprise, please follow the migration procedure below. After following this procedure you will be able to see the connection status of the NuvlaBoxes on the SlipStream dashboard.

1. Make sure that NuvlaBox connector is installed on the SlipStream instance. If not, install it with:

   ```
   yum install slipstream-connector-nuvlabox-enterprise
   ```

   Restart SlipStream service on the current instance:

   ```
   systemctl restart slipstream
   ```

2. Add and configure NuvlaBox connector (e.g. *nuvlabox-james-chadwick:nuvlabox*) on the SlipStream instance. See NuvlaBox documentation for the details. The name of the connector should match the name under which the added NuvlaBox will be publishing its metrics.

3. Connect NB to SS for publication of availability metrics:

   ```
   /root/nuvlabox-register-mothership \
      -U nuvlabox-<NB-name> \
      -S "ssh-rsa <ssh-key> root@nuvlabox-<NB-name>"
   ```

   Add the following configuration parameters before first *Match* section in */etc/ssh/sshd_config*:

   ```
   ClientAliveInterval 15
   ClientAliveCountMax 2
   ```

   Restart *sshd*:

   ```
   systemctl restart sshd
   ```

4. Populate Service Offer resource with the information on the NuvlaBox. This step has to be manually done each time when a new NuvlaBox needs to be made available on the SlipStream instance via the NuvlaBox connector.

   Add NuvlaBox info into the service offer:

   ```
   curl -u super:<super-password> -k -s \
     -D - https://<ss-ip>/api/service-offer -d @nuvlabox.json \
     -H "Content-type: application/json"
   ```

   with the following content in *nuvlabox.json*:

```
{
  "connector" : {"href" : "nuvlabox-<nb-name>"},

  "state": "nok",

  "acl" : {
    "owner" : { "principal" : "ADMIN",
                "type" : "ROLE"},
    "rules" : [
      { "principal" : "USER",
        "type" : "ROLE",
        "right" : "VIEW"}
    ]
  }
}
```

5. Run the following to install and configure the Riemann service.

   The command below is required to be ran if you are upgrading an existing SlipStream instance. You don't need to run the command below if you've just installed SlipStream from scratch:

   ```
   curl -LkfsS https://raw.githubusercontent.com/slipstream/SlipStream/candidate-
   ↪latest/install/ss-install-riemann.sh | bash
   ```

   Edit */etc/sysconfig/riemann* and export the following environment variables:

   ```
   export SLIPSTREAM_ENDPOINT=https://127.0.0.1
   export SLIPSTREAM_SUPER_PASSWORD=change_me_password
   ```

   Restart Riemann service:

   ```
   systemctl restart riemann
   ```

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v3.3 (candidate) - 12 May 2016

New features and bug fixes in v3.3

Because of a serious authentication bug that was introduced, this release has been removed from the YUM package repository.

For application users and developers [Alice, Clara]:

- Added a field in the dashboard run list that indicates how many active VMs are associated with the run.

For application developers [Clara]:

- Use readable names for downloaded deployment scripts to make debugging easier.
- Move deployment scripts out of `/tmp` to avoid them disappearing on reboots.
- Ensure that parameter values starting with a dash do not disrupt the application deployment.
- Fix GET action of ss:groups parameter.

For SlipStream administrators [Dave]:

- Fixed module download/upload cycle so that migration of modules between servers works.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is needed from v3.2 to v3.3.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v3.2 (candidate) - 21 April 2016

New features and bug fixes in v3.2

For application users and developers [Alice, Clara]:

- Rename service catalog offers (service-offer) and attribute (service-attribute) resources for consistency.
- Fix problem with application component scale up from an initial multiplicity of 0.
- REST API more strictly validates its inputs on scale up/down requests.
- Add functions to the clojure client API to launch and terminate applications.

For SlipStream administrators [Dave]:

- Improve logging by providing full URIs of application components.
- Fix error in script that prevented the service from being started.
- Install service catalog by default (Enterprise Edition).

For application users, developers, and SlipStream administrators [Alice, Clara, Dave]:

- Remove the save button on the service catalog when user isn't authorized to make changes.
- Add a "+" to dashboard to make it easier to configure new cloud connectors.
- Make application thumbnails clickable in the App Store.
- Add terminated icon to terminated VMs in the dashboard.
- Fix serialization and calculation of usage information.

- Fix vCloud connector so that node multiplicity works correctly.
- Fix navigation and inactive run filter on the run page.
- Fix refresh for the list of runs on application and application component pages.
- Fix client-side code for sanitizing tags provided by users.
- Fix presentation of the gauges in the dashboard.
- Fix a problem where non-pending VMs were mistakenly marked as pending.

Alice, Bob, Clara, and Dave can be found here.

Migration

No migration is needed from v3.1 to v3.2.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v3.1 (candidate) - 2 April 2016

New features and bug fixes in v3.1

For managers and super users [Bob]:

- Cloud managers can now see an overview of the activity on their cloud from all users.

For SlipStream administrators [Dave]:

- Allow direct proxying of the two SlipStream services through nginx to provide more efficient and reliable system.
- Improved installation and testing scripts.
- Fix virtual machine state mapping for the OpenNebula connector.
- Fix build image functionality for the OpenStack connector.
- Fix various server-side exceptions to avoid “internal server error” responses.
- Remove unnecessary logging to make the server activity easier to understand.

For application users and developers [Alice, Clara]:

- Application component definitions now inherit configuration scripts from their parents, facilitating reuse of existing application components.
- Updated dashboard provides more detailed information about virtual machine states and to which run they belong.
- User profile now provides visual clues as to which cloud connectors are configured and which are not.

- The command line client and API now use nuv.la as the default endpoint for the SlipStream service.
- An early alpha clojure(script) API is now available. It contains functions for scaling runs and for the CRUD actions on CIMI-like resources. Feedback on the API is welcome.
- Restarting an aborted run (through `ss-abort --cancel` now generates an event in the run's event log.
- Expand SlipStream bootstrap mechanism to more operating systems (notably SuSE and OpenSuSE 11-13).
- Improve the logs for machines deployed with SlipStream.

For application users, developers, and SlipStream administrators [Alice, Clara, Dave]:

- Update the general and API documentation to consistently use "scalable" runs for those that can be dynamically scaled while running.

Alice, Bob, Clara, and Dave can be found here.

Migration

NB! Because SlipStream v3 requires the CentOS 7 operating system, an upgrade from the SlipStream v2 series to the SlipStream v3 series requires a complete database migration from the old machine to a new one running CentOS 7.

In addition, the names for the service catalog resources have changed. Follow the migration instructions for those resources before migrating the database, if you are running the service catalog.

Below are the full migration instructions.

Installation of SlipStream

Install SlipStream on CentOS 7 following Administrators Guide. Please note that for installation of SlipStream Enterprise edition you will have to (re-)use the client certificate to be able to access SlipStream Enterprise YUM repository. The certificates are usually installed as */etc/slipstream/yum-client.**. On the existing SlipStream installation this can be checked by:

```
# grep sslclient /etc/yum.repos.d/slipstream.repo
sslclientcert=/etc/slipstream/yum-client.crt
sslclientkey=/etc/slipstream/yum-client.key
...
```

When installing cloud connectors, it's important to ensure that the list of the connectors to be installed matches the one configured on the previous SlipStream instance as we are going to fully migrate DB containing the complete service configuration of the current SlipStream instance to the new one. The list of the installed connectors can be obtained on the current SlipStream by:

```
# rpm -qa | \
      grep slipstream-connector | \
      grep -v python | \
      cut -d'-' -f3 | \
      tee installed-connectors.txt
cloudstack
ec2
opennebula
openstack
nuvlabox
nativesoftlayer
stratuslab
```

(continues on next page)

(continued from previous page)

```
azure
exoscale
#
```

After installation of SlipStream and connectors on CentOS 7, verify that the service is properly up and running by accessing the main page of the service.

Migration of Service Catalog Resources

Following renaming of resources linked to Service Catalog, a script needs to be executed. Please contact support to obtain this script with information on how to run it.

Migration of DB, reports and logs

On the current CentOS 6 machine running SlipStream take the following steps.

1. Stop the following services:

```
$ service nginx stop
$ service slipstream stop
$ service ssclj stop
```

2. Restart hsqldb to checkpoint the DB (this will trigger replay of the WAL log):

```
$ service hsqldb restart
```

3. Stop hsqldb:

```
$ service hsqldb stop
```

4. Archive SlipStream DB, deployment reports, service logs, nginx configuration:

```
$ tar -zc /opt/slipstream/SlipStreamDB \
     /opt/slipstream/server/logs \
     /var/log/slipstream/ssclj \
     /var/tmp/slipstream/reports \
     /etc/nginx/{ssl/,conf.d/} \
     --dereference \
     -f ~/SlipStream-backup.tgz
```

5. Copy the archive to the new CentOS 7 machine that will be hosting SlipStream.

On the new CentOS 7 machine, after installing SlipStream from scratch and validating that it works,

1. Stop all the services by running:

```
$ systemctl stop nginx
$ systemctl stop slipstream
$ systemctl stop ssclj
$ systemctl stop hsqldb
```

2. Inflate the backup tarball as follows:

```
$ tar -zxvf ~/SlipStream-backup.tgz -C /
```

This should inflate

- database to `/opt/slipstream/SlipStreamDB`
- reports to `/var/tmp/slipstream/reports`
- logs to `/opt/slipstream/server/logs` and `/var/log/slipstream/ssclj/`

3. Change the service configuration to reference the new host IP the service is running on by:

```
# sed -i -e '/SERVICECONFIGURATIONPARAMETER/ s/<old-IP>/<new-IP>/g' \
     /opt/slipstream/SlipStreamDB/slipstreamdb.{log,script}
```

4. Update the SlipStream nginx cache location:

```
# sed -i -e 's|proxy_cache_path.*keys_zone=zone_one:10m;|proxy_cache_path /var/
↪local/slipstream/nginx/cache keys_zone=zone_one:10m;|' \
    /etc/nginx/conf.d/slipstream-ssl.conf
```

5. Start all the services in the following order:

```
$ systemctl start hsqldb
$ systemctl start ssclj
$ systemctl start slipstream
$ systemctl start nginx
```

This completes the migration process. Validate the migration by logging to the service and launching a test deployment.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v3.0 (candidate) - 7 March 2016

New features and bug fixes in v3.0

For managers and super users [Bob]:

- Provide better header information in the browser UI when a manager or super users is viewing information from several users.

For SlipStream administrators [Dave]:

- **SlipStream must now be deployed on CentOS 7.** All services have been updated to support systemd only. Caches have been moved from */tmp* and */var/tmp* to avoid startup problems.

For application users, developers, and SlipStream administrators [Alice, Clara, Dave]:

- Improve query performance when retrieving event resources through the API and in the UI.

- Improve graphical feedback when viewing virtual machines to indicate those that are not known to SlipStream.
- OpenNebula connector allows custom template fields to be specified to, for example, attach hardware devices or consoles.
- Fix a bug in the AWS connector that caused the creation of the 'slipstream_managed' security group to fail.

Alice, Bob, Clara, and Dave can be found here.

Migration

Because SlipStream v3 requires the CentOS 7 operating system, an upgrade from the SlipStream v2 series to the SlipStream v3 series requires a complete database migration from the old machine to a new one running CentOS 7. Details for this migration will come with a subsequent release.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.23.2 (stable) - 3 March 2016

New features and bug fixes in v2.23.2

For SlipStream administrators [Dave]:

- Fix a packaging bug that caused the Service Catalog resources not to appear.

Alice, Bob, Clara, and Dave can be found here.

Migration

Database migration is **not** required from v2.23.1 to v2.23.2.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.23.1 (candidate) - 22 February 2016

New features and bug fixes in v2.23.1

For application developers [Clara]:

- Fixed issues with command line client so that the `ss-get --noblock` option works correctly, `ss-abort` no longer requires a message, and the `ss-execute` option `--mutable-run` has been changed to `--scalable`.
- Refactored client clojure API to make actions/functions correspond better to end user needs.
- Fix a bug in which the same resource could be added twice.

For SlipStream administrators [Dave]:

- Fix packaging issue which left out scripts for periodic usage analysis.

For application users, developers, and SlipStream administrators [Alice, Clara, Dave]:

- Improved application state handling to avoid race conditions leading to failures when scaling an application.
- Improve OpenStack connector to reduce time to retrieve the IP address, to order parameters consistently, and to fix a problem where the domain parameter was ignored.
- Extend the OpenStack connector to support the Keystone API v3.
- Stratuslab connector has improved logging of networking errors.
- CloudStack connector now supports multiple zones.
- AWS connector uses only the first SSH key to create a keypair to avoid deployment failures.
- New terminology (application, component, image) is now the default in the user interface.

Alice, Bob, Clara, and Dave can be found here.

Migration

Database migration is **not** required from v2.23 to v2.23.1.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.23 (candidate) - 13 February 2016

New features and bug fixes in v2.23

For application users and developers [Alice, Clara]:

- Provide new Service Catalog (enterprise) implementation along with API documentation for the new ServiceInfo and Attribute resources.

For application developers [Clara]:

- An alpha version of a Clojure API has been created that supports scale up/down features.
- Fix application logging when verbosity level is 0.

For SlipStream administrators [Dave]:

- Optimize data flow by using nginx to route requests to the appropriate SlipStream services.

For application users, developers, and SlipStream administrators [Alice, Clara, Dave]:

- Error handling when starting and stopping runs has been improved.
- CloudStack and Exoscale (enterprise) connectors now support multiple zones.
- OpenStack connector now supports the Keystone API v3 and has been streamlined to avoid unnecessary API calls.
- OpenStack connector has been fixed to accommodate new VM states.
- StratusLab, OpenStack connectors have improved error messages.
- There is now an example application that demonstrates autoscaling.
- A SoftLayer connector (enterprise) that uses native SoftLayer API and that supports vertical scaling is now available.
- Fix problem with vCloud connector (enterprise) caused by missing VM states.
- Fix Firefox display issues for message display and gauges on dashboard.
- Fix bootstrapping failures on Ubuntu 14.04.

Alice, Bob, Clara, and Dave can be found here.

Migration

Database migration is **not** required from v2.22 to v2.23.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.22 (candidate) - 5 February 2016

New features and bug fixes in v2.22

For application users and developers [Alice, Clara]:

- Workaround application logging problem at log level 0

- Improve error reporting from the node executor

For SlipStream administrators [Dave]:

- Roles for users can now be defined by the system administrator
- Remove unnecessary information from service error logs
- Update third-party dependencies for robustness and stability

For application users, developers, and SlipStream administrators [Alice, Clara, Dave]:

- Support GitHub authentication
- Azure connector fully working for linux-based applications
- Fix problem that prevented horizontal scale down from working
- Fix poor or misleading authentication error messages

Alice, Bob, Clara, and Dave can be found here.

Migration

Database migration is required from v2.21 to v2.22. The following steps MUST be followed:

1. Upgrade SlipStream
2. Stop SlipStream

```
$ service slipstream stop
```

3. Stop HSQLDB (or your DB engine)

```
$ service hsqldb stop
```

4. Execute the following SQL script */opt/slipstream/server/migrations/017_add_external_login.sql*:

```
$ java -jar /opt/hsqldb/lib/sqltool.jar --autoCommit --
↪inlineRc=url=jdbc:hsqldb:file:/opt/slipstream/SlipStreamDB/slipstreamdb,user=sa,
↪password= /opt/slipstream/server/migrations/017\_add\_external\_login.sql
```

5. Start HSQLDB (or your DB engine)

```
$ service hsqldb start
```

6. Start SlipStream

```
$ service slipstream start
```

Commits

- Server
- UI
- Client
- Connectors
- Documentation

2.2.5 2016 Stable Releases

v3.14 (stable) - 31 October 2016

Features

Below is the list of the main feature and improvements done in releases 3.9 to 3.14.

- Improvements in the functionality of the placement and ranking service.
- Added complete set of resources for the Service Catalog (Enterprise Edition), allowing policy (and priced) based placement of virtual machines.
- Introduced complete Clojure and ClojureScript API for the SlipStream CIMI resources.
- This set of releases provides much of the groundwork for future improvements. The emphasis has been on preparing new server-side resources for cloud connectors and service configuration; these will improve the management of these resources in the future. There has also been significant work done to streamline the code organization, packaging, and release process. This should speed development of new features.
- Added the delete all versions feature for a module.

The detailed change log is given below. For brevity bug fixes have not been included, see the change logs for the intermediate releases for the full set of changes and fixes.

For application users and developers [Alice, Clara]:

- Provide pricing along with a filtered set of connectors on the run dialog. (Enterprise Edition)
- Improve the bootstrapping process to avoid having the process hang on CentOS 6 systems.
- Provide complete set of service catalog resources (serviceOffer, serviceAttribute, and serviceAttribute-Namespace) to allow policy-based placement using the service catalog information. (Enterprise Edition)
- Provide clojure/clojurescript API for SlipStream CIMI resources. The API provides asynchronous and synchronous implementations of all SCRUD actions. Filtering and subsetting are provided for search operations.
- Use larger modal dialog to avoid truncating long parameter or component names in run dialog.
- Alpha versions of connector and configuration resources. These are available through the API and will be integrated into the web interface in a future release.
- Increase the allowed maximum size of a report.
- Pass SNI information to backend services.
- Add the delete all versions for a module.

For application users [Alice]:

- Provide a better message when a cloud quota has been exceeded. The message now includes the quota, number of running VMs, and number of requested VMs.

For application developers [Clara]:

- Allow application developers to specify a placement policy for application components, for example, limiting the places where a component can run. (Enterprise Edition)
- Improve the error messages reported to users of the SlipStream client API, providing more information about the underlying cause of a problem.
- Selection of specific network for opennebula connector.

For administrators [Dave]:

- Streamline the installation of SlipStream with a packaged version of PhantomJS and with a package for the Elasticsearch repositories.
- Modify service dependencies to ensure cleaner start up of all SlipStream services on boot.
- Improve the collection of virtual machine state information (used in the dashboard) to make it more efficient and reliable. Put in additional logging to make debugging easier.
- Upgrade to the latest production libraries for all server dependencies, improving the robustness of the server (in particular Aleph, Buddy, and ClojureScript).
- Clean up and reorganize the packaging for the pricing service. Logging information will now appear in the standard OS directory.
- Unify build system with boot for clojure code.

The Alice, Bob, Clara, and Dave personae can be found on the SixSq website.

Migration

No migration is required from 3.8 to 3.14.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

---

v3.8 (stable) - 15 July 2016

Features

Version v3.8 is the first stable release of the v3 series. There are major underlying changes to make this release more stable, robust, and performant, including the introduction of Elasticsearch as a database (hsqldb still needed until the transition to Elastisearch is complete), switching from CentOS 6 to CentOS 7, and numerous bug fixes.

In addition, there are a number of new features to make this attactive to both end-users and developers, including better support for scalable applications, improved usage information, an expanded REST API that uses the CIMI standard for new resources, and a streamlined user interface. The Enterprise Edition also contains an alpha-preview of the placement and ranking service that allows policy-based selection of cloud infrastructures when deploying applications and integration with NuvlaBox appliances.

The detailed change log is given below. For brevity bug fixes have not been included, see the change logs for the intermediate releases for the full set of changes and fixes.

For everyone [Alice, Bob, Clara, Dave]:

- Provide a top-level support link for users, if the system administrator has set a support email address.
- In the Enterprise Edition, improve the visualization of the Service Catalog entries and allow more than one entry per cloud connector.

- Provide for status reporting of the NuvlaBox appliances connected to the SlipStream server.

For application users, developers, and SlipStream administrators [Alice, Clara, Dave]:

- Update the general and API documentation to consistently use "scalable" runs for those that can be dynamically scaled while running.
- Improve query performance when retrieving event resources through the API and in the UI.
- Remove the save button on the service catalog when user isn't authorized to make changes.
- Add a "+" to dashboard to make it easier to configure new cloud connectors.
- Make application thumbnails clickable in the App Store.
- Add terminated icon to terminated VMs in the dashboard.
- Improve graphical feedback when viewing virtual machines to indicate those that are not known to SlipStream.
- OpenNebula connector allows custom template fields to be specified to, for example, attach hardware devices or consoles.

For application users and developers [Alice, Clara]:

- Inherited output parameters are visible to the users, allowing an input parameter to be mapped to an inherited output parameter.
- The SlipStream bootstrap process is now able to run on operating systems with only Python 3 installed. The robustness of the bootstrapping process has also been improved.
- Display prices for running components and applications and certain clouds in the run dialog (Enterprise Edition).
- Make the bootstrap mechanism more reliable over low-quality networks (e.g. satellite connections).
- Allow to define relative and absolute paths for module logo.
- Improve the retry mechanism for the SlipStream clients to make them behave more uniformly and to be more robust.
- Added a field in the dashboard run list that indicates how many active VMs are associated with the run.
- Rename service catalog offers (service-offer) and attribute (service-attribute) resources for consistency.
- REST API more strictly validates its inputs on scale up/down requests.
- Add functions to the clojure client API to launch and terminate applications.
- Application component definitions now inherit configuration scripts from their parents, facilitating reuse of existing application components.
- Updated dashboard provides more detailed information about virtual machine states and to which run they belong.
- User profile now provides visual clues as to which cloud connectors are configured and which are not.
- The command line client and API now use nuv.la as the default endpoint for the SlipStream service.
- An early alpha clojure(script) API is now available. It contains functions for scaling runs and for the CRUD actions on CIMI-like resources. Feedback on the API is welcome.
- Restarting an aborted run (through `ss-abort --cancel` now generates an event in the run's event log.
- Expand SlipStream bootstrap mechanism to more operating systems (notably SuSE and OpenSuSE 11-13).
- Improve the logs for machines deployed with SlipStream.

- Exoscale: Add support for Mega and Titan instances.
- OpenStack: Added support for Floating IPs.
- OpenNebula: Added default values for image parameters

For application developers [Clara]:

- Allow the client API to be used for test instances of SlipStream that use a self-signed certificate.
- Update API documentation for cookie authentication. Cookie authentication is now the preferred method; basic authentication is deprecated.
- Add a command to allow the reports from a run to be retrieved.
- Fixed disk size unit in describe instance action in OpenNebula connector.
- DELETE on API resources now returns 200 instead of 204.
- Use readable names for downloaded deployment scripts to make debugging easier.
- Move deployment scripts out of `/tmp` to avoid them disappearing on reboots.
- Ensure that parameter values starting with a dash do not disrupt the application deployment.

For administrators [Dave]:

- Avoid dependency version conflicts by removing hard-coded dependencies for the PRS-lib component.
- Rationalize logging and logging levels
- Improved installation and testing scripts.
- Make the installation script more robust concerning RPM package names.
- Improve the configuration of the nginx configuration to enhance the security of the service.
- Improve logging by providing full URIs of application components.
- Install service catalog by default (Enterprise Edition).
- Allow direct proxying of the two SlipStream services through nginx to provide more efficient and reliable system.
- Remove unnecessary logging to make the server activity easier to understand.
- **SlipStream must now be deployed on CentOS 7.** All services have been updated to support systemd only. Caches have been moved from */tmp* and */var/tmp* to avoid startup problems.

For managers and super users [Bob]:

- Cloud managers can now see an overview of the activity on their cloud from all users.
- Provide better header information in the browser UI when a manager or super users is viewing information from several users.

The Alice, Bob, Clara, and Dave personae can be found on the SixSq website.

Migration

NB! Because SlipStream v3 requires the CentOS 7 operating system, an upgrade from the SlipStream v2 series to the SlipStream v3 series requires a complete database migration from the old machine to a new one running CentOS 7.

In addition, the names for the service catalog resources have changed. Follow the migration instructions for those resources before migrating the database, if you are running the service catalog.

Below are the full migration instructions.

Installation of SlipStream

Install SlipStream on CentOS 7 following Administrators Guide. Please note that for installation of SlipStream Enterprise edition you will have to (re-)use the client certificate to be able to access SlipStream Enterprise YUM repository. The certificates are usually installed as */etc/slipstream/yum-client.**. On the existing SlipStream installation this can be checked by:

```
# grep sslclient /etc/yum.repos.d/slipstream.repo
sslclientcert=/etc/slipstream/yum-client.crt
sslclientkey=/etc/slipstream/yum-client.key
...
```

When installing cloud connectors, it's important to ensure that the list of the connectors to be installed matches the one configured on the previous SlipStream instance as we are going to fully migrate DB containing the complete service configuration of the current SlipStream instance to the new one. The list of the installed connectors can be obtained on the current SlipStream by:

```
# rpm -qa | \
      grep slipstream-connector | \
      grep -v python | \
      cut -d'-' -f3 | \
      tee installed-connectors.txt
cloudstack
ec2
opennebula
openstack
nuvlabox
nativesoftlayer
stratuslab
azure
exoscale
#
```

After installation of SlipStream and connectors on CentOS 7, verify that the service is properly up and running by accessing the main page of the service.

Migration of Service Catalog Resources

Following renaming of resources linked to Service Catalog, a script needs to be executed. Please contact support to obtain this script with information on how to run it.

Migration of DB, reports and logs

On the current CentOS 6 machine running SlipStream take the following steps.

1. Stop the following services:

   ```
   $ service nginx stop
   $ service slipstream stop
   $ service ssclj stop
   ```

2. Restart hsqldb to checkpoint the DB (this will trigger replay of the WAL log):

   ```
   $ service hsqldb restart
   ```

3. Stop hsqldb:

```
$ service hsqldb stop
```

4. Archive SlipStream DB, deployment reports, service logs, nginx configuration:

```
$ tar -zc /opt/slipstream/SlipStreamDB \
     /opt/slipstream/server/logs \
     /var/log/slipstream/ssclj \
     /var/tmp/slipstream/reports \
     /etc/nginx/{ssl/,conf.d/} \
     --dereference \
     -f ~/SlipStream-backup.tgz
```

5. Copy the archive to the new CentOS 7 machine that will be hosting SlipStream.

On the new CentOS 7 machine, after installing SlipStream from scratch and validating that it works,

1. Stop all the services by running:

```
$ systemctl stop nginx
$ systemctl stop slipstream
$ systemctl stop ssclj
$ systemctl stop hsqldb
```

2. Inflate the backup tarball as follows:

```
$ tar -zxvf ~/SlipStream-backup.tgz -C /
```

This should inflate

- database to `/opt/slipstream/SlipStreamDB`
- reports to `/var/tmp/slipstream/reports`
- logs to `/opt/slipstream/server/logs` and `/var/log/slipstream/ssclj/`

3. Change the service configuration to reference the new host IP the service is running on by:

```
# sed -i -e '/SERVICECONFIGURATIONPARAMETER/ s/<old-IP>/<new-IP>/g' \
     /opt/slipstream/SlipStreamDB/slipstreamdb.{log,script}
```

4. Update the SlipStream nginx cache location:

```
# sed -i -e 's|proxy_cache_path.*keys_zone=zone_one:10m;|proxy_cache_path /var/
↪local/slipstream/nginx/cache keys_zone=zone_one:10m;|' \
    /etc/nginx/conf.d/slipstream-ssl.conf
```

5. Start all the services in the following order:

```
$ systemctl start hsqldb
$ systemctl start ssclj
$ systemctl start slipstream
$ systemctl start nginx
```

This completes the migration process. Validate the migration by logging to the service and launching a test deployment.

Further Incremental Migration Steps

Riemann Service

The following migration is required on SlipStream Enterprise instance.

In this release the Riemann service was introduced. It is intended to be used with NuvlaBox product.

If you are using or intending to start using NuvlaBoxes with SlipStream Enterprise, please follow the migration procedure below. After following this procedure you will be able to see the connection status of the NuvlaBoxes on the SlipStream dashboard.

1. Make sure that NuvlaBox connector is installed on the SlipStream instance. If not, install it with:

   ```
   yum install slipstream-connector-nuvlabox-enterprise
   ```

   Restart SlipStream service on the current instance:

   ```
   systemctl restart slipstream
   ```

2. Add and configure NuvlaBox connector (e.g. *nuvlabox-james-chadwick:nuvlabox*) on the SlipStream instance. See NuvlaBox documentation for the details. The name of the connector should match the name under which the added NuvlaBox will be publishing its metrics.

3. Connect NB to SS for publication of availability metrics:

   ```
   /root/nuvlabox-register-mothership \
      -U nuvlabox-<NB-name> \
      -S "ssh-rsa <ssh-key> root@nuvlabox-<NB-name>"
   ```

   Add the following configuration parameters before first *Match* section in */etc/ssh/sshd_config*:

   ```
   ClientAliveInterval 15
   ClientAliveCountMax 2
   ```

   Restart *sshd*:

   ```
   systemctl restart sshd
   ```

4. Populate Service Offer resource with the information on the NuvlaBox. This step has to be manually done each time when a new NuvlaBox needs to be made available on the SlipStream instance via the NuvlaBox connector.

   Add NuvlaBox info into the service offer:

   ```
   curl -u super:<super-password> -k -s \
     -D - https://<ss-ip>/api/service-offer -d @nuvlabox.json \
     -H "Content-type: application/json"
   ```

   with the following content in *nuvlabox.json*:

   ```
   {
     "connector" : {"href" : "nuvlabox-<nb-name>"},

     "state": "nok",

     "acl" : {
       "owner" : { "principal" : "ADMIN",
                   "type" : "ROLE"},
   ```

(continues on next page)

(continued from previous page)

```
    "rules" : [
      { "principal" : "USER",
        "type" : "ROLE",
        "right" : "VIEW"}
    ]
  }
}
```

5. Run the following to install and configure the Riemann service.

   The command below is required to be run if you are upgrading an existing SlipStream instance. You don't need to run the command below if you've just installed SlipStream from scratch:

   ```
   curl -LkfsS https://raw.githubusercontent.com/slipstream/SlipStream/candidate-
   ↪latest/install/ss-install-riemann.sh | bash
   ```

   Edit */etc/sysconfig/riemann* and export the following environment variables:

   ```
   export SLIPSTREAM_ENDPOINT=https://127.0.0.1
   export SLIPSTREAM_SUPER_PASSWORD=change_me_password
   ```

   Restart Riemann service:

   ```
   systemctl restart riemann
   ```

Elasticsearch

Elasticsearch is now required for the SlipStream service. When upgrading, Elasticsearch will need to be installed, configured, and started by hand. Start by adding the Elasticsearch repository:

```
$ yum install slipstream-es-repo-community
```

Use "community" or "enterprise" as appropriate for you installation.

Install Elasticsearch:

```
$ yum install elasticsearch
$ systemctl daemon-reload
$ systemctl enable elasticsearch.service
```

Update the configuration:

```
$ cd /etc/elasticsearch/
$ mv elasticsearch.yml elasticsearch.yml.orig
$ cat > elasticsearch.yml <<EOF
network.host: 127.0.0.1
EOF
```

And finally start the service:

```
$ systemctl start elasticsearch.service
```

You can test that Elasticsearch is running correctly with:

```
$ systemctl status elasticsearch.service
$ curl http://localhost:9200/_cluster/health?pretty=true
```

The first should show that the service is running and the second should provide the health of the Elasticsearch cluster. It should contain one node and be in a "green" state.

For data persistency, SlipStream is moving from hsqldb, a Java-based SQL relational database, to Elasticsearch, a high-performance, document-oriented data store. The migration from one to the other will be incremental, so during the transition, both databases will be used. This is the first release where Elasticsearch is used.

Before starting the migration procedure, please make sure that `slipstream` and `ssclj` are not running. Both databases (hsqldb and Elasticsearch) must be running.

Then you can migrate the resources with the following commands:

```
$ export ES_HOST=localhost
$ export ES_PORT=9300
$ java -cp /opt/slipstream/server/webapps/slipstream.war/WEB-INF/lib/clojure-1.8.0.
↪jar:/opt/slipstream/ssclj/lib/ssclj.jar com.sixsq.slipstream.ssclj.migrate.script
```

Resources are migrated (from hsqldb to elastic search) by batches of 10'000 documents. Example of output of this script:

```
...
Creating ES client
Index resetted
Will create korma database with db-spec
...
Migrating  usage , nb resources = XXX
Migrating usage 0  ->  9999
...
Migrating  usage-record , nb resources = XXX
Migrating usage-record 0  ->  9999
...
Migrating  event , nb resources = XXX
Migrating event 0  ->  9999
...
```

Known Issues

- The process that collects information abouts users' virtual machines can become saturated, resulting in the loss of this information for most users. When this issue appears, the slipstream service can be restarted to return it to a normal state.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.23.2 (stable) - 3 March 2016

Features

As this is a major release, a large number of bugs have been fixed in addition to the listed features. For bug fixes, see the release notes for the intermediate candidate releases. Only the new features are listed below.

For application users [Alice]:

- Major improvements to the text and workflow of the embedded SlipStream tour, making it easier to understand and to follow.
- Major reorganization of the brower interface (and vocabulary), making the dashboard the initial landing page and providing easy access to the other major interface elements (App Store, Workspace, and Service Catalog).

For application users and developers [Alice, Clara]:

- The new Service Catalog implementation allows for flexible schemas and full CRUD actions through the SlipStream API. This allows it to cover a wider range of different cloud services and cloud service providers.
- Improve the application state machine and associated control processes to ensure that there are fewer spurious errors and that scaling is more reliable.
- Enhanced the error reporting from the cloud connectors and the application control processes to make the returned error messages more precise.
- Dashboard has been markedly improved to provide a clearer and more concise view of your cloud activities. For example, only gauges relevant to you are shown and you can filter out terminated applications. Applications can provide direct, clickable links to the deployed service.
- The events on the "run" page of an application are automatically refreshed (and time-ordered) to allow you to easily follow the progress of your application.

For application developers [Clara]:

- Streamlined and refactored the command line interface to make the usage more intuitive.
- Report tarball has been "flattened" to make navigation of the logs easier.
- A script can now be defined for the orchestrator (beta feature) that allows for deployment-wide actions for an application.
- Provides an alpha client API in clojure that provides functions that allow you to control most of an application's lifecycle, particularly the scaling actions.

For SlipStream administrators [Dave]:

- Improved packaging that simplifies installation of SlipStream, ensures that customized configuration files are not inadvertantly overwritten, and allows the services to run with SELinux.
- Optimized data flow through the nginx proxy to the appropriate, backend SlipStream services; refine rate limits so that they do not affect normal usage.
- Administrators can now assign roles to users that can be used within resource URLs.
- Reduce unnecessary logging to make the log files more effective when trying to find problems.
- SlipStream now supports several external authentication mechanisms to be used, GitHub for example.

For application users, developers, and SlipStream administrators [Alice, Clara, Dave]:

- Improve browser support to ensure a consistent rendering across all of the major browsers.

- SlipStream supports scaling both horizontally (adding more machines) and vertically (adding more resources).
- There is an example application that demonstrates autoscaling with SlipStream.
- Daily, weekly, and monthly summaries of your cloud resource usage are available. Daily reminders can also be enabled in your user profile.
- New events have been added that provide a broader view of important actions within the SlipStream server and managed cloud applications. The events indicate when the server was started/stopped, when user profiles are updated, and when the server configuration changes.
- Automatically create an open security group (on clouds that support it) to avoid application failures due to network connectivity.

The list of available cloud connectors has expanded and existing connectors have been improved:

- AWS (EC2)
  - Connector only uses the first configured SSH key during deployment to avoid provisioning failures.
  - Errors messages in general and those related to the VPC change have been improved.
- Azure
  - A complete connector for Azure is available that allows the full control of linux-based systems.
- CloudStack
  - Connector now supports multiple zones.
- Exoscale
  - This specialized cloud connector allows images to be referenced by name, disk sizes to be controlled, and platform-specific instance sizes.
- OpenNebula
  - A connector to use OpenNebula platforms from SlipStream is available.
  - The OpenNebula machines templates can be customized from the SlipStream interface.
- OpenStack
  - Now supports the Keystone API v3.
  - Connector has been streamlines to reduce the time to retrieve the virtual machine's IP address.
  - Error messages have been improved to help resolve connectivity and cloud problems.
- SoftLayer
  - A connector (enterprise) that uses the native SoftLayer API is now available. The connector supports vertical scaling.
- StratusLab
  - Improved logging of networking errors as well as error messages.

The Alice, Bob, Clara, and Dave personae can be found on the SixSq website.

Migration

When upgrading from previous versions two files must be renamed by hand:

- `mv /etc/default/slipstream.rpmnew /etc/default/slipstream`

- `mv /etc/default/ssclj.rpmnew /etc/default/ssclj`

This is not needed on a fresh installations of v2.23.2.

Database migration is required from v2.14 to v2.23.2. The following steps MUST be followed:

1. Upgrade SlipStream
2. Stop SlipStream:

```
$ service slipstream stop
```

3. Stop HSQLDB (or your DB engine):

```
$ service hsqldb stop
```

4. Execute the following SQL script */opt/slipstream/server/migrations/015_compute_timestamp_usage.sql*:

```
$ java -jar /opt/hsqldb/lib/sqltool.jar --autoCommit --
↪inlineRc=url=jdbc:hsqldb:file:/opt/slipstream/SlipStreamDB/sscljdb,user=sa,
↪password= /opt/slipstream/server/migrations/015_compute_timestamp_usage.sql
```

5. Execute the following SQL script */opt/slipstream/server/migrations/016_add_frequency_usage.sql*:

```
$ java -jar /opt/hsqldb/lib/sqltool.jar --autoCommit --
↪inlineRc=url=jdbc:hsqldb:file:/opt/slipstream/SlipStreamDB/sscljdb,user=sa,
↪password= /opt/slipstream/server/migrations/016_add_frequency_usage.sql
```

6. Execute the following SQL script */opt/slipstream/server/migrations/017_add_external_login.sql*:

```
$ java -jar /opt/hsqldb/lib/sqltool.jar --autoCommit --
↪inlineRc=url=jdbc:hsqldb:file:/opt/slipstream/SlipStreamDB/slipstreamdb,user=sa,
↪password= /opt/slipstream/server/migrations/017\_add\_external\_login.sql
```

7. Start HSQLDB (or your DB engine):

```
$ service hsqldb start
```

8. Delete all usage_summaries, and recompute them thanks to summarizer script:

```
$ java -Dconfig.path=db.spec -cp \ "/opt/slipstream/ssclj/resources:/opt/
↪slipstream/ssclj/lib/ext/*:/opt/slipstream/ssclj/lib/ssclj.jar" \
 com.sixsq.slipstream.ssclj.usage.summarizer -f <frequency> -n <nb-in-past>
```

Use ‘daily, ‘weekly’ and ‘monthly’ for ‘-f’ option. Adapt value given to ‘-n’ option for each frequency.

9. Start SlipStream:

```
$ service slipstream start
```

Known Issues

No major known issues.

Commits

- Server

- UI
- Client
- Connectors
- Documentation

2.2.6 2015 Candidate Releases

v2.21 (candidate) - 18 December 2015

New features and bug fixes in v2.21

For application users and developers [Alice, Clara]:

- The Dashboard can now filter out inactive runs, allowing you to focus on your running applications.
- On the Dashboard and in the Run Dialog, only those clouds that you have configured are shown, reducing visual clutter on the page.

For SlipStream administrators [Dave]:

- Roles can now be added to a user profile. Those roles can eventually be used in the ACLs (Access Control Lists) for resources.
- The RPM packaging has been improved for several components, in particular marking configuration files so that they are not overwritten on upgrades.
- Spurious authentication failures after a server restart have been eliminated.

For application users, developers, and SlipStream administrators [Alice, Clara, Dave]:

- OpenNebula cloud infrastructures can now be accessed from SlipStream.
- SoftLayer cloud infrastructures can now be accessed from SlipStream Enterprise Edition deployments.
- The foundations for a new implementation of service catalog with definable attributes have been laid. This will eventually allow advanced searching of cloud services that can be used for automated placement of applications.
- The SlipStream testing pipeline has been extended, providing more thorough testing and a more stable service for you.

Alice, Bob, Clara, and Dave can be found here.

Migration

Database migration is **not** required from v2.20 to v2.21.

Commits

- Server
- UI
- Client
- Connectors

- Documentation

v2.20 (candidate) - 4 December 2015

New features and bug fixes in v2.20

For application users [Alice]:

- Improve text and workflow of the embedded SlipStream tour text, making it easier understand and follow.

For application users and developers [Alice, Clara]:

- The events on the "run page" that shows the details of a cloud application deployment are automatically refreshed, making it easier to follow the timeline of an application.
- Fix a bug which caused virtual machines that were removed from the deployment via the "scale-down" feature to not be terminated correctly.

For application developers [Clara]:

- The organization of the archive (tarball) containing the reports has been flattened, making navigation to the reports easier.
- A script can now be defined for the orchestrator, which allows deployment-wide actions for an application. (Warning: beta feature!).

For SlipStream administrators [Dave]:

- Better consistency when setting the SlipStream theme: the method for configuring the default and non-default themes is now uniform.
- Extend the custom style sheet to allow the background of the active menubar items to be set within a theme.
- Performance metrics related to the SlipStream servers themselves are now pushed to the local Graphite server, where they can be viewed.
- Username validation at registration is more strict to avoid creation of accounts which wouldn't work correctly.
- Correct the CloudStack connector packaging which could cause the symbolic links to CloudStack connector commands to be removed.
- Refine the nginx rate limits so that they do not kick in for normal usage levels.
- Fix a bug where the administrator ("super") would not see the events for all application deployments.

For everyone [Alice, Bob, Clara, Dave]:

- Weekly and monthly summaries of the cloud resource usage are available, in addition to the existing daily summary.
- New events have been added that provide a broader view of important actions within the SlipStream server and managed cloud applications. The events indicate when the server was started/stopped, when user profiles are updated, and when the server configuration changes.
- Make the application deployment workflow more reliable by introducing retries when encountering transient failures.
- Fix a bug where the usage records could be incorrect if the SlipStream server was restarted.
- Fix pagination of entries on the run and module displays. Requesting a new page happens immediately rather than waiting for the next automatic refresh cycle.

Alice, Bob, Clara, and Dave can be found here.

Migration

Database migration is required from v2.19.1 to v2.20. The following steps MUST be followed:

1. Upgrade SlipStream
2. Stop SlipStream

```
$ service slipstream stop
```

3. Stop HSQLDB (or your DB engine)

```
$ service hsqldb stop
```

4. Execute the following SQL script */opt/slipstream/server/migrations/016_add_frequency_usage.sql*:

```
$ java -jar /opt/hsqldb/lib/sqltool.jar --autoCommit --
↪inlineRc=url=jdbc:hsqldb:file:/opt/slipstream/SlipStreamDB/sscljdb,user=sa,
↪password= /opt/slipstream/server/migrations/016_add_frequency_usage.sql
```

5. Start HSQLDB (or your DB engine)

```
$ service hsqldb start
```

6. Delete all usage_summaries, and recompute them thanks to summarizer script:

```
$ java -Dconfig.path=db.spec -cp \ "/opt/slipstream/ssclj/resources:/opt/slipstream/
↪ssclj/lib/ext/*:/opt/slipstream/ssclj/lib/ssclj.jar" \
 com.sixsq.slipstream.ssclj.usage.summarizer -f <frequency> -n <nb-in-past>
```

Use ‘daily, ‘weekly’ and ‘monthly’ for ‘-f’ option. Adapt value given to ‘-n’ option for each frequency.

7. Start SlipStream

```
$ service slipstream start
```

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.19.1 (candidate) - 17 November 2015

New features and bug fixes in v2.19.1

For everyone [Alice, Bob, Clara, Dave], a couple bug fixes:

- Fix instabilities in the authentication system that caused erratic behavior.
- Make the application deployment workflow more reliable by introducing retries when encountering transient failures.

Alice, Bob, Clara, and Dave can be found here.

Migration

Database migration is **not** required from v2.19 to v2.19.1.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.19 (candidate) - 12 November 2015

New features and bug fixes in v2.19

For users [Alice, Clara]:

- The run page has been enhanced to show the time-ordered list of events associated with a run.
- The vocabulary in the interface has been made more clear and precise to make understanding SlipStream easier.

For users [Alice, Clara] and administrators [Dave]:

- There is now a prototype (alpha) Azure connector available, which will be extended to a production connection over the next couple of releases.
- There is a specialized cloud connector for the Exoscale cloud platform that allows images to be referenced by name, disk sizes to be controlled, and platform-specific instance sizes.
- Allow the proper inheritance of image parameters to avoid having to edit/save child images when a parent has been modified.

For administrators [Dave]:

- There is now a configuration option that will allow server metrics (e.g. request responses, request rates, service resource usage) to be pushed to a Graphite server.
- Logging levels have been reduced in many cases to avoid noise in the logs.
- A new authentication system is being used that will allow external authentication mechanisms to be used for a SlipStream server.
- SElinux can now be used for the machine running the SlipStream server, allowing the service to be more tightly secured.

For everyone [Alice, Bob, Clara, Dave], a few bug fixes:

- Modify the introductory tour to follow the new application layout.
- When an attribute error is raised, provide a correct error message rather than a misleading one referring to an illegal state.

- Upgrade internal SSH libraries to allow deployment to work with newer versions of Ubuntu (15.04+).
- Correct a problem that caused new projects to be created but not visible.
- Truncate log error messages in run parameters to avoid masking the real error with an internal server error (500).

Alice, Bob, Clara, and Dave can be found here.

Migration

Database migration is **not** required from v2.18 to v2.19.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.18 (candidate) - 23 october 2015

New features and bug fixes in v2.18

- Make the Dashboard the landing page for users
- Dashboard, Modules, App Store, and Service Catalog are split in the UI and have direct links from top menubar
- Include root disk volumes for StratusLab clouds
- Improve units for displaying cloud resource usage
- Consolidated monthly usage available through API
- Improve EC2 connector to catch errors related to VPC change and to provide more informative error message
- fix: add missing module in SlipStream client package for *pip* (affected *ss-config-dump* command)

Migration

Database migration is required from v2.17 to v2.18. The following steps MUST be followed:

1. Upgrade SlipStream
2. Stop SlipStream

```
$ service slipstream stop
```

3. Stop HSQLDB (or your DB engine)

```
$ service hsqldb stop
```

4. Execute the following SQL script */opt/slipstream/server/migrations/015_compute_timestamp_usage.sql*:

```
$ java -jar /opt/hsqldb/lib/sqltool.jar --autoCommit --
↪inlineRc=url=jdbc:hsqldb:file:/opt/slipstream/SlipStreamDB/sscljdb,user=sa,
↪password= /opt/slipstream/server/migrations/015_compute_timestamp_usage.sql
```

5. Start HSQLDB (or your DB engine)

```
$ service hsqldb start
```

6. Start SlipStream

```
$ service slipstream start
```

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.17 (candidate) - 5 october 2015

New features and bug fixes in v2.17

- Allow use of the http-kit or aleph web application containers (clojure server)
- Allow initialization of resources before starting server (clojure server)
- Clean up main and server namespaces for ssclj server (clojure server)
- After launching a run, the user gets redirected to the dashboard (previously the redirection was to the run page)
- Add back the environment variable SLIPSTREAM_CONNECTOR_INSTANCE
- fix: terminate button is properly updated after closing dialog in the dashboard
- fix: fixed an issue which prevented multi-cloud deployment to work
- fix: add missing index in resources table (clojure server)

Migration

A database migration from v2.16 to v2.17 is not needed.

Commits

- Server
- UI
- Client
- Connectors

- Documentation

v2.16 (candidate) - 18 September 2015

New features and bug fixes in v2.16

- HTML representations of event and usage resources available
- improved configuration for cloud connector configuration
- upgrade to latest libcloud release (0.18.0) for all connectors
- allow easier automated installation from configuration files
- allow finer control over information dumped in `ss-config-dump`
- create open security group to avoid app. failures on clouds that support it
- add prototype user-editable service catalog (enterprise)
- fix: `ss-config-dump` for unaliased connector names
- fix: reintroduce older EC2 VM sizes
- fix: allow multiple versions of Java on SlipStream machines
- fix: missing python dependency in packages for cloud connectors
- fix: incorrect path for dependency in OpenStack and CloudStack connectors
- fix: run parameters not shown on image module

Migration

A database migration from v2.15 to v2.16 is not needed. However, when upgrading from previous versions two files must be renamed by hand:

- `mv /etc/default/slipstream.rpmnew /etc/default/slipstream`
- `mv /etc/default/ssclj.rpmnew /etc/default/ssclj`

This is not needed on a fresh installations of v2.16.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.15 (candidate) - 29 August 2015

New features and bug fixes in v2.15

- documentation for horizontal and vertical scaling of applications (horizontal scaling is supported by all connectors; **vertical scaling is currently only supported by flexiant and okeanos connectors**)
- update terminology in UI: mutable changed to scalable
- dashboard improvements: auto-refresh, service URL link, and terminate button
- improve layout of workflow scripts on image modules
- allow SlipStream configuration to be dumped and restored from files
- change location of log files to permanent `/var/log/slipstream` location
- upgrade jetty (9.3.2), libcloud (0.18.0), and other java/clojure dependencies
- fix: failures on CloudStack connector when service returns empty body in requests
- fix: make CIMI CloudEntryPoint conform to standard
- fix: pagination in image and deployment pages
- fix: pagination in run section of a module

Migration

A database migration from v2.14 to v2.15 is not needed.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.14 (stable) - 13 August 2015

New features and bug fixes in v2.14

- add SlipStream instance to the text of usage email messages
- mark Java 1.7 as conflicting with SlipStream RPM package installation
- improve mechanism for initial bootstrap configuration of server from configuration files
- change URLs for event (and other clojure) resources from camel-case to kebab-case
- change change CIMI root resource api/CloudEntryPoint
- fix: pagination of results in UI
- fix: crash of node executor on empty target script output

Migration

A database migration from v2.13 to v2.14 is not needed.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.13 (candidate) - 30 July 2015

New features and bug fixes in v2.13

- reduced dependency from jdk to jre
- migrated to java 8
- provide more metrics from connectors (cpu, ram, instance type, root disk size)
- multiple bug fixes and improvements in UI
- run page refreshes asynchronously on background
- on run page alert (abort) messages are truncated (full abort message can still be seen in Global section)
- display a loading screen while waiting for request from the server
- added an ability for machine executor (orchestrator and node) to survive reboot of the host they are running on
- more metrics can now be returned by OpenStack and CloudStack connectors
- VMs section of dashboard can now display cpu, ram, instance type and root disk size if provided by the cloud connectors
- improved collection of the usage records

Migration

A database migration from v2.12 to v2.13 is not needed.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.12 (candidate) - 10 July 2015

New features and bug fixes in v2.12

- added documentation on obtaining API Key and Secret on CloudStack
- improved packaging of python code for cloud connectors
- updated and improved example image and deployment modules that are shipped with SlipStream; added documentation on how to publish the modules to running SlipStream instance
- bug fixes and improvements of the machine executor (orchestrator and node)
- initial implementation of vertical scaling of node instances
- new SlipStream dashboard layout with correspondingly adapted tour
- numerous fixes and improvements in UI

Migration

A database migration from v2.11 to v2.12 is not needed.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.11 (candidate) - 22 June 2015

New features and bug fixes in v2.11

- users can now receive daily cloud usage emails (turn on via parameter in user profile)
- changes to the server for better support of application scaling
- the cloud contextualization method can be chosen through the cloud connector configuration
- Java 1.8 is now required by the SlipStream server
- fix: handling of open subsection in URL
- fix: catch EINTR interrupt to prevent script failures on Windows
- fix: invalid URL when clicking on VM gauge in dashboard
- fix: problem with scaling scripts not being called on scaling actions
- fix: various browser issues with embedded SlipStream tour

Migration

A database migration from v2.10 to v2.11 is not needed.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.10 (candidate) - 7 June 2015

New features and bug fixes in v2.10

- interactive tour available through SlipStream interface (beta)
- clicking on dashboard gauges opens the corresponding cloud section
- allow event and usage resources to be filtered
- disallow changes to parameter types through UI to be consistent with server
- improve contextualization mechanisms for Windows
- allow admins to choose contextualization method used for a cloud
- fix: dashboard gauges incorrectly rendered in some cases
- fix: wrong version comment sometimes displayed for module
- fix: module logo is not displayed
- fix: Windows deployments intermittently fail
- fix: “noscript” message was not working when JavaScript

Migration

A database migration from v2.9 to v2.10 is not needed.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.9 (stable) - 18 May 2015

New features and bug fixes in v2.9

- only allow configured clouds to be used in UI
- provide pagination of event and usage resources
- package scripts for preparing usage summaries
- reduce resource requirements for collected metrics
- patch timezone handling bug in UI
- fix storage of service configuration enum parameters
- remove unnecessary dependencies in build artifacts

Migration

A database migration from v2.8 to v2.9 is not needed.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.8 (candidate) - 29 April 2015

New features and bug fixes in v2.8

- allow connectors to indicate when a VM is usable (for usage records)
- improve logging (more concise messages, longer retention times)
- provide quick installation script with documentation of procedure
- provide "event" resource with standard lifecycle events
- expose "usage" summary as a resource
- updated advanced tutorial for current release
- fix bug which prevented deployments from being saved
- fix bug which erased parameters starting with `http://`
- fix deadlock associated with multiple database clients
- fix run ordering by time
- fix truncation of fields hiding information (popovers used everywhere)
- improve rendering of errors to make the cause more visible

Migration

Database migration is required from v2.7 to v2.8. The following steps MUST be followed:

1. Upgrade SlipStream
2. Stop SlipStream

```
$ service slipstream stop
```

3. Stop HSQLDB (or your DB engine)

```
$ service hsqldb stop
```

4. Execute the following SQL script */opt/slipstream/server/migrations/014_enumvalues_size_fix.sql*:

```
$ java -jar /opt/hsqldb/lib/sqltool.jar --autoCommit --
↪inlineRc=url=jdbc:hsqldb:file:/opt/slipstream/SlipStreamDB/slipstreamdb,user=sa,
↪password= /opt/slipstream/server/migrations/014_enumvalues_size_fix.sql
```

5. Start HSQLDB (or your DB engine)

```
$ service hsqldb start
```

6. Start SlipStream

```
$ service slipstream start
```

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.7 (stable) - 15 April 2015

New features and bug fixes from v2.7

- Bug fixes for launching and accessing Windows virtual machines
- Support for v5.5 of vCloud API
- Allow input parameters to be specified for simple image run to avoid having to create a deployment for this
- Add back App Store to the image chooser
- Add custom error pages for SlipStream frontend proxy
- Make forward/backward navigation more natural (avoid URLs with fragment changes in history)
- Improve rendering of tables on mobile devices

Migration

No migration is required from v2.6.1 to v2.7.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.6.1 (stable) - 7 April 2015

This release has been promoted to a stable release.

New features and bug fixes from v2.6

- UI critical bug fix: null pointer exception in the VMs section of dashboard
- UI bug fix: ‘Undefined’ incorrectly prepended to ‘Provisioning’ message

Migration

No migration is required from v2.6 to v2.6.1.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.6 (candidate) - 2 April 2015

New features and bug fixes from v2.5

- Expose event resource
- Allow usage notes to be added to image and deployment modules
- Filter VMs by User (for administrator) and by Run Owner
- Add more node information in VM resources (UI and XML)

- Allow input parameters for simple run
- Allow
- Improvements to VMs resource: additional node information, ability to filter by User/Run Owner/Run UUID
- Ability to run an image with installation scripts even if the image has not been built.
- Ensure that a module “copy” operation copies all fields
- Fix for time zone parsing error
- Ensure build image operation works
- Fix bugs in v2.5 that caused SlipStream to stop responding to requests and that caused ready applications to be moved to “finalizing” incorrectly
- Improve standard example applications: Ubuntu Standalone, CentOS Standalone, Wordpress, and LAMP++
- Improve monitoring of service with collectd
- Ensure time is aligned between SlipStream services by adding ntpd to SlipStream deployments
- Move documentation to dedicated server and remove the embedded documentation from the SlipStream server
- Numerous UI improvements: disactivating buttons when actions are not allowed, display user-friendly state in dashboard, improvements for touch devices, fix wrapping of fields on small devices, improve organization of sections in user profile

Migration

You have to execute the following script (while HSQLDB is running) to do the BD migration:

```
java -jar /opt/hsqldb/lib/sqltool.jar --autoCommit --inlineRc=url=jdbc:hsqldb:hsql://
↪localhost:9001/slipstream,user=sa,password= --sql "UPDATE VmRuntimeParameterMapping␣
↪SET hostnameRuntimeParameterUri = CONCAT(REGEXP_
↪SUBSTRING(vmstateRuntimeParameterUri,'^[^:]+'),':hostname') WHERE␣
↪hostnameRuntimeParameterUri IS NULL;"
```

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.5 (candidate) - 20 March 2015

New features and bug fixes from v2.4.2 (stable)

- Added the Event server
- Improved authorization mechinisme
- Improved logging

- Improved the collector
- Improved stability of the /vms resource when there is a huge amount of VMs
- Improved the Run dialog on the UI:
- The Cloud for all node can be selected at one place
- The two checkboxes in the user profile to define the `keep running` behaviour was converted into a dropdown menu
- The `keep running` behaviour can be redefined
- Tags can be defined when creating a Run.
- The value selected for `Cloud` and `Keep running` dropdown menus correspond to the default of the user profile.
- It's now possible to create a Run even if there is no SSH key in the user profile
- An error is displayed if SSH access is asked but there is no key in the user profile
- Improved the time needed to terminate VMs with `stratuslabiter-terminate-instances`.
- Increased the maximum amount of items returned by /vms and /run to 500
- New packaging for the community edition.
- Fixed a bug where deployment scripts were not executed when running a simple image.
- Bugfixes

Migration

IMPORTANT: v2.5 requires data migration from v2.4.2. The following steps MUST be followed:

1. Upgrade SlipStream
2. Ensure SlipStream is running
3. Execute the following python script *012_edit_save_all_users.py* from the directory */opt/slipstream/server/migrations/*

```
$ cd /opt/slipstream/server/migrations/
$ python 012_edit_save_all_users.py <username> <password>
```

   `<username>` and `<password>` have to be credentials of a SlipStream administrator.

4. Stop SlipStream

```
$ service slipstream stop
```

5. Stop HSQLDB (or your DB engine)

```
$ ss-db-shutdown
```

6. Execute the following SQL script */opt/slipstream/server/migrations/013_convert_to_keep_running.sql*:

```
$ java -jar /opt/hsqldb/lib/sqltool.jar --inlineRc=url=jdbc:hsqldb:file:/opt/
↪slipstream/SlipStreamDB/slipstreamdb,user=sa,password= /opt/slipstream/server/
↪migrations/013_convert_to_keep_running.sql
```

7. Start HSQLDB (or your DB engine)

```
$ service hsqldb start # ignore start error
```

8. Start SlipStream

```
$ service slipstream start
```

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.4.2 (stable) - 28 February 2015

This release has been promoted to a stable release.

For this and previous stable releases see the "Stable Releases" page.

2.2.7 2015 Stable Releases

v2.14 (stable) - 13 August 2015

From 2.13 (candidate) to v2.14 (stable)

- add SlipStream instance to the text of usage email messages
- mark Java 1.7 as conflicting with SlipStream RPM package installation
- improve mechanism for initial bootstrap configuration of server from configuration files
- change URLs for event (and other clojure) resources from camel-case to kebab-case
- change change CIMI root resource api/CloudEntryPoint
- fix: pagination of results in UI
- fix: crash of node executor on empty target script output

From 2.12 (candidate) to v2.13 (candidate)

- reduced dependency from jdk to jre
- migrated to java 8
- provide more metrics from connectors (cpu, ram, instance type, root disk size)
- multiple bug fixes and improvements in UI
- run page refreshes asynchronously on background
- on run page alert (abort) messages are truncated (full abort message can still be seen in Global section)

- display a loading screen while waiting for request from the server
- added an ability for machine executor (orchestrator and node) to survive reboot of the host they are running on
- more metrics can now be returned by OpenStack and CloudStack connectors
- VMs section of dashboard can now display cpu, ram, instance type and root disk size if provided by the cloud connectors
- improved collection of the usage records

From 2.11 (candidate) to v2.12 (candidate)

- added documentation on obtaining API Key and Secret on CloudStack
- improved packaging of python code for cloud connectors
- updated and improved example image and deployment modules that are shipped with SlipStream; added documentation on how to publish the modules to running SlipStream instance
- bug fixes and improvements of the machine executor (orchestrator and node)
- initial implementation of vertical scaling of node instances
- new SlipStream dashboard layout with correspondingly adapted tour
- numerous fixes and improvements in UI

From 2.10 (candidate) to v2.11 (candidate)

- users can now receive daily cloud usage emails (turn on via parameter in user profile)
- changes to the server for better support of application scaling
- the cloud contextualization method can be chosen through the cloud connector configuration
- Java 1.8 is now required by the SlipStream server
- fix: handling of open subsection in URL
- fix: catch EINTR interrupt to prevent script failures on Windows
- fix: invalid URL when clicking on VM gauge in dashboard
- fix: problem with scaling scripts not being called on scaling actions
- fix: various browser issues with embedded SlipStream tour

From v2.9 (stable) to v2.10 (candidate)

- interactive tour available through SlipStream interface (beta)
- clicking on dashboard gauges opens the corresponding cloud section
- allow event and usage resources to be filtered
- disallow changes to parameter types through UI to be consistent with server
- improve contextualization mechanisms for Windows
- allow admins to choose contextualization method used for a cloud
- fix: dashboard gauges incorrectly rendered in some cases

- fix: wrong version comment sometimes displayed for module
- fix: module logo is not displayed
- fix: Windows deployments intermittently fail
- fix: “noscript” message was not working when JavaScript

Migration

No migration is needed from v2.9 to v2.14.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.9 (stable) - 18 May 2015

From v2.8 (candidate) to v2.9 (stable)

- only allow configured clouds to be used in UI
- provide pagination of event and usage resources
- package scripts for preparing usage summaries
- reduce resource requirements for collected metrics
- patch timezone handling bug in UI
- fix storage of service configuration enum parameters
- remove unnecessary dependencies in build artifacts

From v2.7 (stable) to v2.8 (candidate)

- allow connectors to indicate when a VM is usable (for usage records)
- improve logging (more concise messages, longer retention times)
- provide quick installation script with documentation of procedure
- provide “event” resource with standard lifecycle events
- expose “usage” summary as a resource
- updated advanced tutorial for current release
- fix bug which prevented deployments from being saved
- fix bug which erased parameters starting with “http://”

- fix deadlock associated with multiple database clients
- fix run ordering by time
- fix truncation of fields hiding information (popovers used everywhere)
- improve rendering of errors to make the cause more visible

Migration

The migration procedures should be run in the order from the last stable release to the current release.

From v2.8 (candidate) to v2.9 (stable)

No migration required.

From v2.7 (stable) to v2.8 (candidate)

Database migration is required from v2.7 to v2.8. The following steps MUST be followed:

1. Upgrade SlipStream
2. Stop SlipStream

```
$ service slipstream stop
```

3. Stop HSQLDB (or your DB engine)

```
$ service hsqldb stop
```

4. Execute the following SQL script */opt/slipstream/server/migrations/014_enumvalues_size_fix.sql*:

```
$ java -jar /opt/hsqldb/lib/sqltool.jar --autoCommit --
↪inlineRc=url=jdbc:hsqldb:file:/opt/slipstream/SlipStreamDB/slipstreamdb,user=sa,
↪password= /opt/slipstream/server/migrations/014_enumvalues_size_fix.sql
```

5. Start HSQLDB (or your DB engine)

```
$ service hsqldb start
```

6. Start SlipStream

```
$ service slipstream start
```

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.7 (stable) - 15 April 2015

New features and bug fixes from v2.7

- Bug fixes for launching and accessing Windows virtual machines
- Support for v5.5 of vCloud API
- Allow input parameters to be specified for simple image run to avoid having to create a deployment for this
- Add back App Store to the image chooser
- Add custom error pages for SlipStream frontend proxy
- Make forward/backward navigation more natural (avoid URLs with fragment changes in history)
- Improve rendering of tables on mobile devices

Migration

No migration is required from v2.6.1 to v2.7.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.6.1 (stable) - 7 April 2015

New features and bug fixes

From 2.6 (candidate) to 2.6.1 (stable)

- UI critical bug fix: null pointer exception in the VMs section of dashboard
- UI bug fix: ‘Undefined’ incorrectly prepended to ‘Provisioning’ message

From 2.5 (candidate) to 2.6 (candidate)

- Expose event resource
- Allow usage notes to be added to image and deployment modules
- Filter VMs by User (for administrator) and by Run Owner
- Add more node information in VM resources (UI and XML)
- Allow input parameters for simple run
- Allow

- Improvements to VMs resource: additional node information, ability to filter by User/Run Owner/Run UUID
- Ability to run an image with installation scripts even if the image has not been built.
- Ensure that a module "copy" operation copies all fields
- Fix for time zone parsing error
- Ensure build image operation works
- Fix bugs in v2.5 that caused SlipStream to stop responding to requests and that caused ready applications to be moved to "finalizing" incorrectly
- Improve standard example applications: Ubuntu Standalone, CentOS Standalone, Wordpress, and LAMP++
- Improve monitoring of service with collectd
- Ensure time is aligned between SlipStream services by adding ntpd to SlipStream deployments
- Move documentation to dedicated server and remove the embedded documentation from the SlipStream server
- Numerous UI improvements: disactivating buttons when actions are not allowed, display user-friendly state in dashboard, improvements for touch devices, fix wrapping of fields on small devices, improve organization of sections in user profile

From v2.4.2 (stable) to v2.5 (candidate)

- Added the Event server
- Improved authorization mechinisme
- Improved logging
- Improved the collector
- Improved stability of the /vms resource when there is a huge amount of VMs
- Improved the Run dialog on the UI:
- The Cloud for all node can be selected at one place
- The two checkboxes in the user profile to define the `keep running` behaviour was converted into a dropdown menu
- The `keep running` behaviour can be redefined
- Tags can be defined when creating a Run.
- The value selected for `Cloud` and `Keep running` dropdown menus correspond to the default of the user profile.
- It's now possible to create a Run even if there is no SSH key in the user profile
- An error is displayed if SSH access is asked but there is no key in the user profile
- Improved the time needed to terminate VMs with `stratuslabiter-terminate-instances`.
- Increased the maximum amount of items returned by /vms and /run to 500
- New packaging for the community edition.
- Fixed a bug where deployment scripts were not executed when running a simple image.
- Bugfixes

Migration

The migration procedures should be run in the order from the last stable release to the current release.

From v2.6 (candidate) to v2.6.1 (stable)

No migration necessary.

From v2.5 (candidate) to v2.6 (candidate)

You have to execute the following script (while HSQLDB is running) to do the BD migration:

```
java -jar /opt/hsqldb/lib/sqltool.jar --autoCommit --inlineRc=url=jdbc:hsqldb:hsql://
↪localhost:9001/slipstream,user=sa,password= --sql "UPDATE VmRuntimeParameterMapping␣
↪SET hostnameRuntimeParameterUri = CONCAT(REGEXP_
↪SUBSTRING(vmstateRuntimeParameterUri,'^[^:]+'),':hostname') WHERE␣
↪hostnameRuntimeParameterUri IS NULL;"
```

From 2.4.2 (stable) to v2.5 (candidate)

IMPORTANT: v2.5 requires data migration from v2.4.2. The following steps MUST be followed:

1. Upgrade SlipStream
2. Ensure SlipStream is running
3. Execute the following python script *012_edit_save_all_users.py* from the directory */opt/slipstream/server/migrations/*

   ```
   $ cd /opt/slipstream/server/migrations/
   $ python 012_edit_save_all_users.py <username> <password>
   ```

   `<username>` and `<password>` have to be credentials of a SlipStream administrator.
4. Stop SlipStream

   ```
   $ service slipstream stop
   ```

5. Stop HSQLDB (or your DB engine)

   ```
   $ ss-db-shutdown
   ```

6. Execute the following SQL script */opt/slipstream/server/migrations/013_convert_to_keep_running.sql*:

   ```
   $ java -jar /opt/hsqldb/lib/sqltool.jar --inlineRc=url=jdbc:hsqldb:file:/opt/
   ↪slipstream/SlipStreamDB/slipstreamdb,user=sa,password= /opt/slipstream/server/
   ↪migrations/013_convert_to_keep_running.sql
   ```

7. Start HSQLDB (or your DB engine)

   ```
   $ service hsqldb start # ignore start error
   ```

8. Start SlipStream

```
$ service slipstream start
```

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.4.2 - 28 February 2015

New features and bug fixes from v2.4.0

- Change monitoring implementation to avoid corrupted dashboard information
- Improve monitoring implementation to avoid peaks in activity
- Allow deployments to set a tolerance for provisioning failures
- Fix bug that caused service catalog entries to be deleted
- Allow style of UI to be more easily customized
- Validate multiplicity values in deployments
- SlipStream client now backs off and waits when server is loaded
- Add network mapping parameters for OpenStack connector
- Add pagination support for VM listings on dashboard
- Optimize uploading of reports to improve performance
- Numerous minor improvements and bug fixes in UI

Migration

IMPORTANT: v2.4.2 requires data migration from v2.4.0. The following steps MUST be followed:

1. Stop SlipStream
2. Stop HSQLDB (or your DB engine)
3. Execute the following SQL files located in `/opt/slipstream/server/migrations`:

- `011_add_maxprovisioningfailures_in_node.sql`

4. Start HSQLDB (or your DB engine)
5. Start SlipStream**

Command to stop HSQLDB:

```
java -jar /opt/hsqldb/lib/sqltool.jar --inlineRc=url=jdbc:hsqldb:hsql://
↪localhost:9001/slipstream,user=sa,password= --sql 'SHUTDOWN;'
```

Example command to execute the migration script:

```
java -jar /opt/hsqldb/lib/sqltool.jar --autoCommit --inlineRc=url=jdbc:hsqldb:file:/
↪opt/slipstream/SlipStreamDB/slipstreamdb,user=sa,password= /opt/slipstream/server/
↪migrations/011_add_maxprovisioningfailures_in_node.sql
```

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.4.1 - 20 February 2015

This release is deprecated because of problems discovered after deployment. Use the v2.4.2 release.

v2.4.0 - 13 January 2015

New features and bug fixes

- New UI based on Bootstrap
- Added export of users as CSV
- Image Run will attach extra disk if defined in cloud parameters and the action is supported by the cloud connector
- Minor updates and fixes in StratusLab and StratusLabIter connector

Migration

No DB migration (from v2.3.9) is required.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

2.2.8 2014 Stable Releases

v2.3.9 - 19 December 2014

New features and bug fixes

- Bugfix of the service catalog on the welcome page.
- Improvements in documentation around traoubleshooting of the user deployments.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.3.8 - 17 December 2014

Migration procedure

IMPORTANT: v2.3.8 requires data migration from v2.3.7. The following steps MUST be followed:

1. Stop SlipStream
2. Stop HSQLDB (or your DB engine)
3. Execute the following SQL files located in `/opt/slipstream/server/migrations`:
   - `010_varchar_size_fix_3.sql`
4. Start HSQLDB (or your DB engine)
5. Start SlipStream**

Command to stop HSQLDB:

```
java -jar /opt/hsqldb/lib/sqltool.jar --inlineRc=url=jdbc:hsqldb:hsql://
↪localhost:9001/slipstream,user=sa,password= --sql 'SHUTDOWN;'
```

Example command to execute the migration script:

```
java -jar /opt/hsqldb/lib/sqltool.jar --autoCommit --inlineRc=url=jdbc:hsqldb:file:/
↪opt/slipstream/SlipStreamDB/slipstreamdb,user=sa,password= /opt/slipstream/server/
↪migrations/010_varchar_size_fix_3.sql
```

New features and bug fixes

- Performance improvement for Runs with a big amount of VMs.
- StratusLab connector was refactored.
- Support Cloud images without wget preinstalled (fallback to curl).

- Bug fixes.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.3.7 - 7 November 2014

New features and bug fixes

- Refactored cloud connector base classes to simplify connector development and maintenance on both Java and Python parts.
- EC2 connector: migrated to the AWS python-boto 2.32.
- StratusLab connector: RPM name changed - `slipstream-connector-stratuslab-python` obsoletes `stratuslab-slipstream-downloads`.
- Bug fixes.

Migration

No DB migration (from v2.3.6) is required.

Commits

- Server
- UI
- Client
- Connectors
- Documentation

v2.3.6 - 29 October 2014

New features and bug fixes

- Removed all usage of the deprecated SSLv3
- Prefer the usage of TLSv1 for secure communications.
- Bug fixes

Migration

No DB migration (from v2.3.5) is required.

Commits

- Server
- UI
- Client
- Documentation

v2.3.5 - 23 October 2014

New features and bug fixes

- Removed autocreation of the users test and sixsq.
- Improvement of the logging.
- Fixed a bug where the ownership of a module can be changed implicitly when editing the module (#14).
- Fixed a bug in the orchestrator that can generate a error in a mutable run (#15).
- Fixed a bug in the StratusLab connector that prevent to Run an Image with an extra disk (#16).
- Fixed a bug in the vCloud connector that prevent it to work with SlipStream v2.3.4+ (#17).
- Added support for building an image with ss-execute.

Migration

No DB migration (from v2.3.4) is required.

Commits

- Server
- UI
- Client
- Documentation

v2.3.4 - 3 October 2014

Migration procedure

IMPORTANT: v2.3.4 requires data migration from v2.3.0. The following steps MUST be followed:

1. Stop SlipStream
2. Stop HSQLDB (or your DB engine)

3. Execute the following SQL files located in `/opt/slipstream/server/migrations`:
   - `008_runtimeparameter_new_name_column.sql`
   - `009_embedded_authz_in_module.sql`
4. Start HSQLDB (or your DB engine)
5. Start SlipStream**

Command to stop HSQLDB:

```
java -jar /opt/hsqldb/lib/sqltool.jar --inlineRc=url=jdbc:hsqldb:hsql://
↪localhost:9001/slipstream,user=sa,password= --sql 'SHUTDOWN;'
```

Example command to execute the migration script:

```
java -jar /opt/hsqldb/lib/sqltool.jar --autoCommit --inlineRc=url=jdbc:hsqldb:file:/
↪opt/slipstream/SlipStreamDB/slipstreamdb,user=sa,password= /opt/slipstream/server/
↪migrations/008_runtimeparameter_new_name_column.sql
```

New features and bug fixes

- Database performance improvement.
- Added support of mutable Run in ss-execute.
- All server-side connectors are now extracted in individual packages.
- Added per-connector config files.
- Improved XML importation.
- Improved error reporting from SlipStream Clients to the SlipStream Server.
- Increase the maximal size of runtime parameter values to 4096 bytes.
- Fixed a bug which prevent to get the runtimeparameters ‘ids’ and ‘multiplicity’ with ss-get.
- Fixed a bug where a failure in a deployment script might not be detected.
- Fixed a bug where deployment refuse to start if the cloudservice is set to ‘default’.
- Fixed a bug of circular reference in modules.
- Updated the documentation.

Commits

- Server
- UI
- Client
- Documentation

v2.3.0 - 14 August 2014

New features and bug fixes

- Mutable Run.
- Some UI improvements related to the mutable run.
- SlipStream Client is now tolerant to network fault.
- Refactored the SlipStream Client. Connectors needs to be upgraded to work with this version.
- Improved the security of all resources by generating a restricted cookie for each Run.
- When Metering is disabled the data collection is now also disabled.
- Overall performance improvements.

Migration

No DB migration (from v2.2.5) is required.

Commits

- Server
- UI
- Client
- Documentation

v2.2.5 - 18 June 2014

New features and bug fixes

- Some UI improvements related to the new state machine.
- In the UI when a Run page is loaded the delay of 10 seconds before the first update of the overview section was removed.
- Added the ability for privileged users to see the vmstate in the Runs of other users.
- Improved the migration of the garbage collector.
- Improved the logging and the error handling of describeInstance.
- Fixed an HTTP 500 when there is no user-agent in the request.
- Fixed a bug where when you try to build an image, run a deployment or run an image, the latest version is always used even if you were not on the latest version when creating the Run.

Commits

- Server
- UI

- Client
- Documentation

v2.2.4 - 13 June 2014

Migration procedure

IMPORTANT: v2.2.4 requires data migration from v2.2.3. The following steps MUST be followed:

1. Stop SlipStream
2. Stop HSQLDB (or your DB engine)
3. Execute the SQL files located in `/opt/slipstream/server/migrations` (files 006 and 007)
4. Start HSQLDB (or your DB engine)
5. Start SlipStream**

Example command to execute the migration script:

```
java -jar /opt/hsqldb/lib/sqltool.jar --debug --autoCommit --
↪inlineRc=url=jdbc:hsqldb:file:/opt/slipstream/SlipStreamDB/slipstreamdb,user=sa,
↪password= /opt/slipstream/server/migrations/006_run_states_fix.sql
```

New features and bug fixes

- New State Machine.
- New logic for the garbage collector.
- Auto-discovery of connectors.
- Fixed a bug where module parameters disappear of the old version when a new version is saved.
- Improved some RuntimeParameters.
- Fixed a bug where SSH login with keys doesn't work on images with SELinux enabled.
- Improved messages displayed during a Build.
- Added target script termination when abort flag is raised.
- Improved the detection of VMs not killed in a final state.

Commits

- Server
- UI
- Client
- Documentation

v2.2.3 - 2 June 2014

New features and bug fixes

- Improved error handling of CloudStack connector
- Fixed a bug with SSH (paramiko)
- Updated RPM packaging of SlipStream client
- Updated xFilesFactor of graphite. For local update run the following

  for f in $(find /var/lib/carbon/whisper/slipstream/ -name *.wsp); do whisper-resize $f –xFilesFactor=0 –aggregationMethod=max 10s:6h 1m:7d 10m:5y; done

Commits

- Server
- UI
- Client
- Documentation

v2.2.2 - 27 May 2014

New features and bug fixes

- Updated CloudStack connector to use the new TasksRunner when terminating instances
- Force draw on usage panel, since now default section

Commits

- Server
- UI
- Client
- Documentation

v2.2.1 - 26 May 2014

Migration procedure

IMPORTANT: v2.2.1 requires data migration from v2.2.0. The following steps MUST be followed:

1. Stop SlipStream
2. Stop HSQLDB (or your DB engine)
3. Execute the SQL files located in `/opt/slipstream/server/migrations` (file 005)
4. Start HSQLDB (or your DB engine)
5. Start SlipStream**

New features and bug fixes

- Multi-thread bulk VM creation can be limited for clouds that can't cope
- Added support for CloudStack Advanced Zones as a sub-connector
- Fix issues related to API doc and xml processing
- Made c3p0 optional (see jar-persistence/src/main/resources/META-INF/persistence.xml for details)
- Add persistence support for MySQL and Postgres
- Update the OpenStack connector to use the new OpenStack CLI
- Update poms following SlipStreamParent -> SlipStream git repo rename
- Upgrade c3p0 version
- Now using Apache HTTP client connector unstead of default Restlet Client connector
- Streamline log entries for asynchronous activity
- Upgrade Restlet to v2.2.1
- Metering update communicate via temporary file instead of stdin
- Remove StratusLab from default configuration
- Fix strange orm issue with JPA 2.0
- A few more minor bug fixes

Commits

- Server
- UI
- Client
- Documentation

v2.2.0 - 10 May 2014

Migration procedure

IMPORTANT: v2.2.0 requires data migration from v2.1.x. The following steps MUST be followed:

1. Stop SlipStream
2. Stop HSQLDB (or your DB engine)
3. Execute the SQL files located in `/opt/slipstream/server/migrations` (files 001..004)
4. Start HSQLDB (or your DB engine)
5. Start SlipStream**

New features and bug fixes

- Fixed performance issue under heavy load due to HashMap causing infinite loop
- Wrapping parameters of Parameterized into ConcurrentHashMap
- Improved asynchronious behaviour
- Improved metering feature
- Removed dependency on jclouds-slf4j
- Removed hibernate3 maven plugin
- Added SQL migration scripts
- Removed Nexus tasks for repo generation
- Migrate to Hibernate 4.3.5
- Fix checkbox not set correctly in edit mode for user
- Enable c3p0 database connection pooling by default
- Improve ergonomics of run dashboard
- Fixed issue with the metering legend items ending with a parenthesis
- Fix several minor bug

Commits

- Server
- UI
- Client
- Documentation

CHAPTER 3

Tutorials

These hands-on tutorials demonstrate the core features of SlipStream and NuvlaBox.

3.1 SlipStream Tutorial

This hands-on tutorial demonstrates the core features of SlipStream by building a number of example applications through a series of exercises. The tutorial shows how SlipStream:

- Abstracts away differences between clouds, making **cloud portability** possible,
- Speeds application development by **promoting reuse** of parameterized application components,
- Ensures **consistent, fast, and error-free deployment** of applications through automation,
- Coordinates the configuration of an application's component services to ensure **reliable initialization**,
- Improves reliability through geo-redundant services in **multi-cloud deployments**,
- Provides **universal access** through a web-base UI, command line interface, and a REST API, and
- Optimizes resource use though **horizontal and vertical scaling**.

The tutorial is divided into a number of modules that treat specific topics (see the Table of Contents). The first two provide an overview of the cloud ecosystem and SlipStream concepts. These provide a good foundation for understanding the core SlipStream functionality that is presented in Module III. The subsequent modules treat more advanced topics and can be followed in any order.

The exercises provided in the tutorial build on one another starting from a simple, single machine web server and ending with an auto-scalable distributed multi-component application. Solutions to the exercises can be found on Nuvla.

3.1.1 Module I - Overview of Cloud and SlipStream (Basic)

To get an overview of cloud technology and SlipStream, we recommend reading the following sections. They provide a high-level view on the technologies and how different users can benefit from them.

Cloud Technology

Cloud technologies have reached levels of maturity and market penetration such that organizations have to justify why they are *not* using cloud rather than the other way around.

This has occurred because cloud technologies offer real benefits to users and organizations. With cloud infrastructures, people can easily deploy large applications and can dynamically tune the allocated resources to maximize responsiveness and reliability. This flexibility has made the use of cloud technologies economically attractive by reducing capital and/or personnel costs.

Critical Components

There are three critical components of cloud technologies that have lead to its success:

- **Mature Virtualization Technologies** Virtualization techniques have been directly integrated into mainstream operating systems and modern CPUs. This integration allows use of virtualization with only a negligible impact on performance.
- **Simple, Universal APIs** Most clouds use Resource Oriented Architectures and "REST" APIs over the ubiquitous HTTP protocol. Doing so makes the service universally accessible from all programming languages and reuses the well understood HTTP service model.
- **Ubiquitous, Reliable Networking** Robust, high-bandwidth cellular networks, wifi, and wired Internet connections provide universal, reliable, 24/7 access to critical services hosted in remote infrastructures like clouds.

By combining these components, cloud platforms provide efficient, powerful computing resources that consumers can easily and reliably access from anywhere.

Cloud Jargon

The constant barrage of marketing using the term "cloud" can make it difficult to develop a precise understanding of what cloud technologies are. Fortunately, the American standards institute (NIST) provides clear, precise definitions that fit our needs. These definitions are the *de facto* standard for discussing cloud platforms.

Service Models

What resources (or services) are provided by the cloud? In answer to this question, NIST and others define three "service models":

- **Software as a Service (SaaS)** Provides a complete application to a customer hosted on a cloud platform to provide lower latencies, better bandwidth, scaling, or other features. Typically the customer will access the service through a web browser or another client on the customers computer.
- **Platform as a Service (PaaS)** Provides a programming environment and cloud infrastructure featuring high-level capabilities like load-balancing, scaling, etc., relieving the programmer from having to construct those services from scratch in the application. Typically the customer is an application developer who accesses the service through a proprietary, language-specific API.
- **Infrastucture as a Service (IaaS)** Provides access to raw computing resources (virtual machines, storage, etc.) that can be provisioned (and released) rapidly. Customers access these services either through a simple (usually REST) API or through a web interface.

These service models are often presented as a hierarchy as a PaaS is often built over a IaaS, as well as a SaaS over a PaaS.

Despite these clear-cut definitions, real services tend to be more complicated, offering elements of the different service models from the same cloud infrastructure.

Deployment Models

The "deployment model" answers the question: **Who uses the cloud infrastructure?** NIST defines three deployment models:

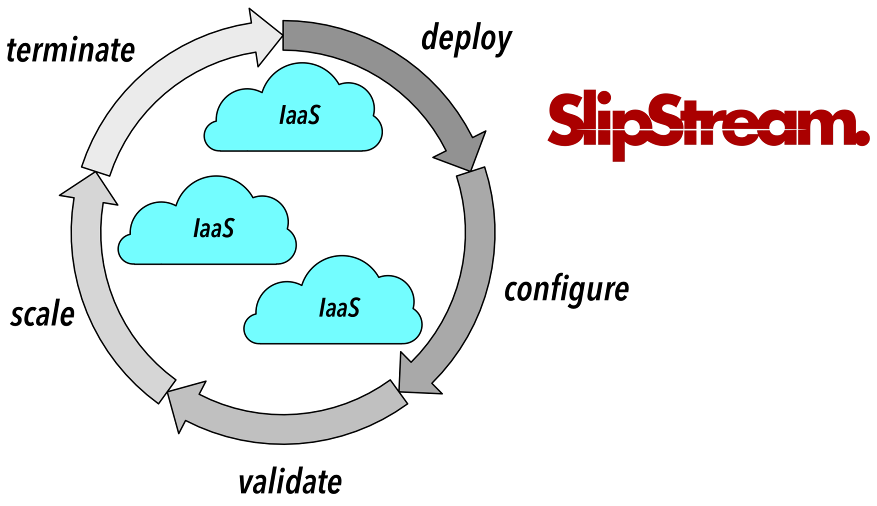
- **Private** These are infrastructures in which the computing resources are co-located with its primary users. The users (or their institute) buy the computing resources directly and run them as a cloud for their own purposes.
- **Community** These are infrastructures run by and for a group of collaborating institutes with similar aims. The users usually buy some fraction of the computing resources and share those resources with others in the community. Allocation of resources between people is usually done via "horse trading."
- **Public** Cloud infrastructures that offer their resources to the general public. The customers pay for use of the computing resources directly (usually via a credit card). The computing resources are housed in data centers controlled by the owner of the cloud, not by the customer or the customer's institute.

NIST actually defines a fourth deployment model, **Hybrid Cloud**, which is really just a mix of the other deployment models. This usually comes up in the context of "cloud bursting", where remote cloud resources (public clouds) are used when a local cloud resource (private cloud) becomes saturated.

What is SlipStream?

SlipStream, developed by SixSq, is a **multi-cloud application management platform**.

Using resources from **Infrastructure as a Service (IaaS)** cloud infrastructures, SlipStream manages cloud applications through the full lifecycle: deployment, configuration, validation, scaling, and termination.



You can use SlipStream from the Nuvla service maintained by SixSq, a **hosted** SlipStream instance, or an **on-premise** installation of SlipStream.

SlipStream Ecosystem

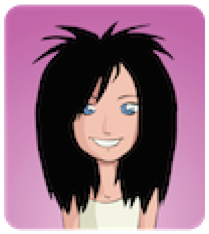
Cloud technologies provide real benefits to users and organizations, but they also have their own challenges. For example, at the "Infrastructure as a Service" level:

- **Incompatible APIs** Make it difficult to move applications from one cloud to another and complicate the simultaneous use of different clouds.
- **Opaque VMs** Keeping track of what virtual machines contain (data and services) and managing their updates are difficult.
- **Component vs. Application** Most applications comprise multiple layers with numerous individual machines. Cloud services oriented towards single VMs make application management more tedious.

SlipStream addresses these challenges by providing its users with an efficient platform for the management of the full lifecycle of cloud applications.

Users and Benefits

A number of different types of people within an organization can benefit from SlipStream. We've created personas to describe those people and how they benefit.



Alice is busy working on different projects. She needs IT applications and resources, but has little patience for IT related issues. She benefits from the SlipStream App Store where she can start the applications she needs with one click.

Bob manages a number of workers taking advantage of cloud resources. He wants an overview of their use of those resources to understand costs and their evolving needs. SlipStream provides the ability to monitor resource utilization.

Clara develops cloud applications for people within her organization. She benefits from SlipStream by creating a rich catalog of services that can be automatically and reliably deployed.

Dave manages the SlipStream installation. He's able to integrate his own cloud infrastructure into SlipStream and control what external cloud resources are available to his users.

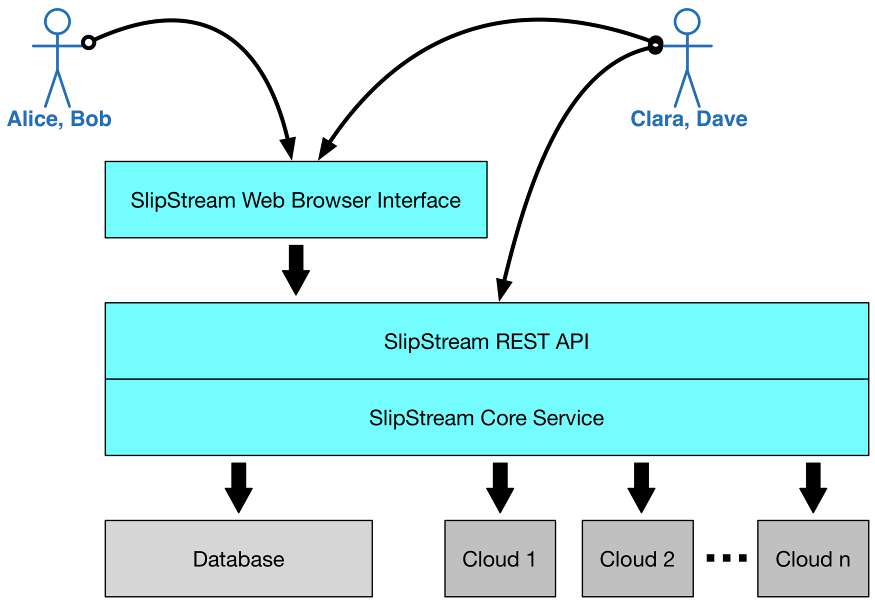
This tutorial focuses primarily on "Clara", showing how applications can be brought into SlipStream and made available to the "Alices" in her organization. Nonetheless, you will see some features (e.g. usage statistics) that will appeal to the Bobs and Daves in your organization.

Interacting with SlipStream

All of the users will interact with SlipStream primarily through the **web-based user interface**. This provides a graphical view of the system allowing applications to be defined, deployed, and managed.

At the core of SlipStream, however, is a **REST API**: a resource-oriented model implemented over the HTTP protocol. The API can be accessed from any programming language, allowing SlipStream to be driven from other systems and integrated into existing application management processes.

There is also a **command line client** that takes advantage of the REST API. It can be used for quick interactions with SlipStream without the need for a web browser. It is used extensively in application definitions to coordinate service configuration and to pass information back to the SlipStream user.

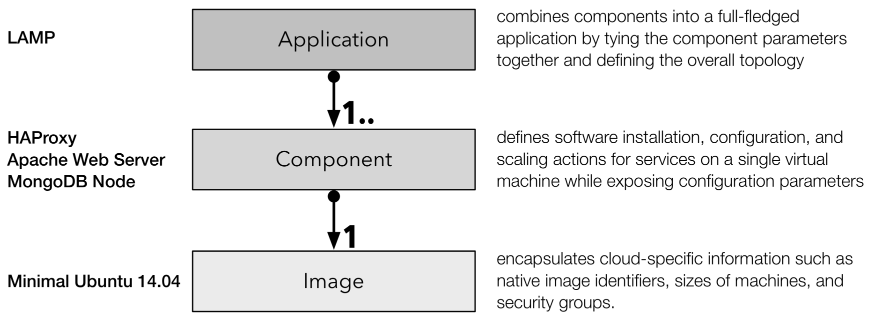


Key Concepts

There are a few key concepts that are important to understand how the various resources are put together to define and to manage cloud applications.

Application Model

SlipStream uses a hierarchical description of applications to allow for portability and easy scaling of applications when they are deployed. Applications are composed of parameterized components which reference generic virtual machine images. The following diagram shows this composition.


LAMP
Application
combines components into a full-fledged application by tying the component parameters together and defining the overall topology
1..
HAProxy
Apache Web Server
MongoDB Node
Component
defines software installation, configuration, and scaling actions for services on a single virtual machine while exposing configuration parameters
1
Minimal Ubuntu 14.04
Image
encapsulates cloud-specific information such as native image identifiers, sizes of machines, and security groups.

By isolating the cloud-specific information in the images, SlipStream enhances the portability of the applications. By defining the application topology, separate from the components, it facilitates scaling the application's functional elements and reusing components across applications.

As a concrete example consider a 3-tier web application: the LAMP (Linux, Apache, MongoDB, and PHP) stack. The components are:

- **HAProxy**: a single load-balancer between the client and web servers,
- **Apache Web Server(s)**: the front-end of the service, and
- **MongoDB Worker(s)**: providing resources for the application's database.

Each of these are built over a minimal Ubuntu 14.04 virtual machine image that exists in the cloud infrastructure being used. The numbers of web servers and/or workers could be scaled to deal with increasing or decreasing application load.

Vocabulary

We have recently updated the vocabulary we use to describe the various resources within SlipStream to make the concepts more intuitive. This new vocabulary hasn't made it through all of the code and documentation, so you may come across some of the old terms. The older terms are shown in parentheses.

Image (base or native image) A virtual machine image that encapsulates cloud-specific information, such as image identifiers, sizes of a machine, and associated security groups. The referenced, native images in each cloud are expected to be effectively identical.

Component (machine image, node) A single virtual machine definition that references an image and may contain scripts for the installation and configuration of additional services. These components can be parameterized and can often be run as standalone applications.

Application (deployment) An application brings together one or more components into a coordinated deployment of cooperating virtual machines. This allows complex (potentially multi-cloud) applications to be defined and managed as a single entity.

Project A "folder" that allows Image, Component, and Application definitions to be organized hierarchically.

Module A generic name for Image, Component, Application, and Project definitions.

Run A deployed (running) application or application component. A "run" encapsulates all of the runtime information of the application and acts as a resource by which the application is managed.

These sections do not contain any exercises and can be read without needing access to a SlipStream server.

Once you understand the technologies and key concepts, follow *Module II* that provides a tour of the SlipStream cloud application management platform.

3.1.2 Module II - SlipStream Web Interface Tour (Basic)

This module lists the prerequisites for working with SlipStream and then guides you through the core features via the embedded SlipStream tour. As part of this tour, you will deploy your first application on a cloud with SlipStream. You will also learn to view and edit your user profile, allowing you to set important parameters and provide credentials for the cloud infrastructures you want to use.

This module presumes familiarity with cloud technologies and the key concepts of SlipStream. Review *Module I* if this is not the case.

Prerequisites

Before starting the tutorial, you must have:

- An account on Nuvla (or another SlipStream installation),
- Accounts on two[1] different cloud infrastructures, and
- A properly configured workstation.

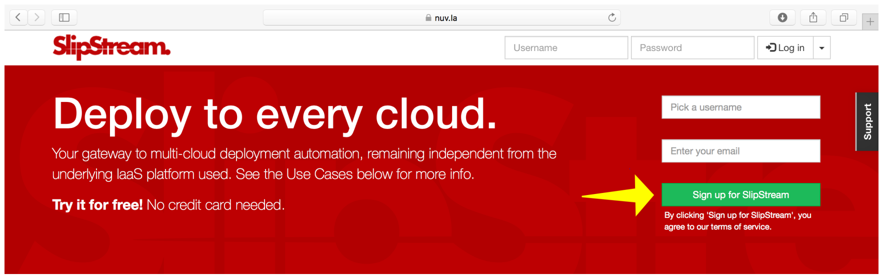
Instructions on how to obtain accounts and how to configure your workstation are below.

Nuvla Account

SlipStream allows people to use multiple cloud infrastructures transparently and easily. Personal accounts allow users to keep their work private and to protect their cloud credentials. This tutorial assumes that you're using Nuvla, a free SlipStream service ("SlipStream SaaS") operated by SixSq.

Registration (New Account)

You can create a new account by registering directly through the service.


nuv.la
SlipStream.
Username
Password
Log in
Deploy to every cloud.
Your gateway to multi-cloud deployment automation, remaining independent from the underlying IaaS platform used. See the Use Cases below for more info.
Try it for free! No credit card needed.
Pick a username
Enter your email
Sign up for SlipStream
By clicking 'Sign up for SlipStream', you agree to our terms of service.
Support

The registration procedure follows the usual pattern for web applications and will probably be familiar to you. Nonetheless, the detailed instructions for creating an account on Nuvla are:

1. Fill in the registration form on the Nuvla login page, providing a username and an email address.
2. You will then receive an email with a link to verify your email address.

[1] You can also use an account on a cloud infrastructure with more than one region. You can also follow the tutorial using only one cloud account, but in this case, you won't be able to complete the exercises demonstrating the multi-cloud features of SlipStream.

3. Visit the URL provided in that email, either by clicking on the link or copying it into your browser.
4. You should then see a page that says that your email address was validated.
5. A second email, containing a temporary password, will then be sent.
6. Visit the Nuvla login page again and log in with your username and temporary password. **For now, close the splash screen offering a tour of SlipStream. We'll get back to that!**
7. View your user profile by clicking on "Profile" under your username at the top, right side of the page.
8. Change your temporary password by clicking on "Edit", updating the password fields in the "Summary" section and then clicking on "Save".

In your profile, you will also need to provide cloud credentials and optionally an SSH public key to make full use of SlipStream. The configuration steps are provided below.

Tip: If you're using your own SlipStream installation, replace the Nuvla endpoint with the endpoint of your server. Note that the administrator of the SlipStream service may *not* allow open registration of users.

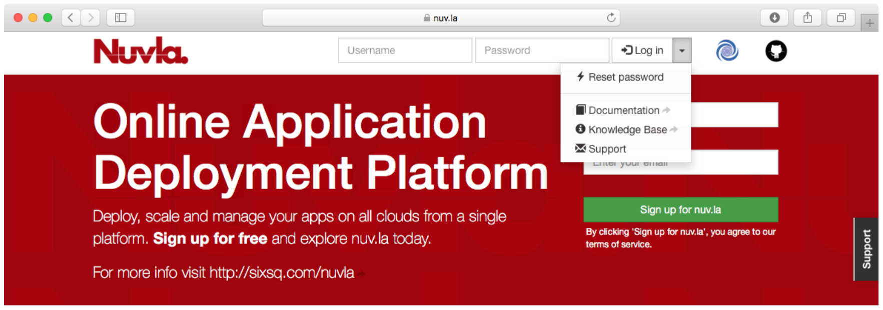
EXERCISES

1. Follow the Nuvla registration procedure to obtain an account.
2. Change the temporary password and logout/login to verify that it works.

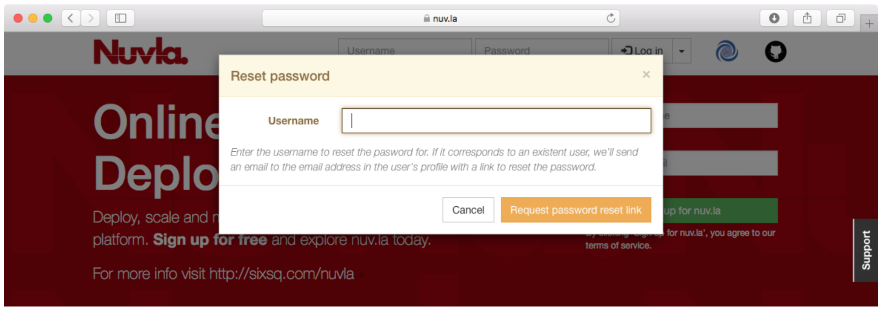
Reset Password (Existing Account)

If an account has been created for you, then you can reset the password to gain access to the account. To do this, you must have the username associated with your account. The administrator of the SlipStream service should have provided you with this information.

First, from Nuvla click on the password reset link. You can find the location in the following screenshot.


nuv.la
Nuvla.
Username
Password
Log in
Reset password
Documentation
Knowledge Base
Support
Online Application Deployment Platform
Deploy, scale and manage your apps on all clouds from a single platform. Sign up for free and explore nuv.la today.
For more info visit http://sixsq.com/nuvla
Sign up for nuv.la
By clicking 'Sign up for nuv.la', you agree to our terms of service.
Support

Next, provide your username and then request the password reset.


Reset password

Username

Enter the username to reset the pasword for. If it corresponds to an existent user, we'll send an email to the email address in the user's profile with a link to reset the password.

nuv.la
Username
Password
Log in
Cancel
Request password reset link
Online
Deplo
Deploy, scale and m
platform. Sign up for free and explore nuv.la today.
For more info visit http://sixsq.com/nuvla
up for nuv.la
for nuv.la', you agree to our terms of service.
Support

You will receive a message sent to the email address associated with the account. Validate the password reset request by visiting the link in the message. You'll then receive a new password for the account. You can change the password to something you'll remember on your user profile page.

EXERCISES

1. Reset the password on your account.
2. Change the temporary password and logout/login to verify that it works.

Cloud Infrastructure Accounts

You will also need to provide the credentials for at least one cloud infrastructure before being able to use SlipStream to deploy cloud applications. To complete the **multi-cloud** examples in this tutorial, you will need access to **two cloud infrastructures** or **one cloud service with two regions**.

SlipStream supports nearly all major cloud service providers and open source cloud solutions. The registration procedure is similar for all cloud services, but you may need to contact your cloud administrator for all of the necessary configuration parameters.

The detailed procedure is provided for Exoscale below.

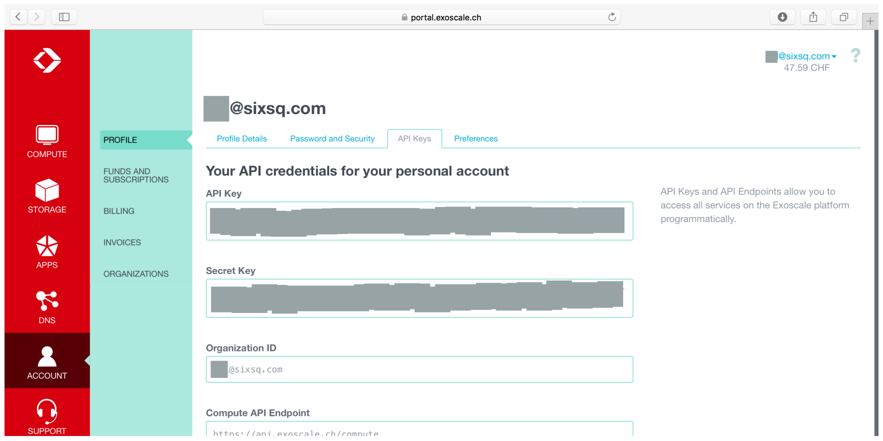
Exoscale

If you need to create an account at Exoscale, you can visit their registration page. If you already have an account go directly to Exoscale login page.

Tip: If you have a promotional card with an initial credit, be sure to use it when you first sign up. To use the promotional card you need to access the registration resource using the following URL https://portal.exoscale.ch/register?coupon=YOURCODEHERE by copy/pasting it to browser's URL bar and replacing YOURCODEHERE with the promotional code found on the card.

Once you've obtained your account with Exoscale, then you'll need to provide your Exoscale credentials to Nuvla. To find the information you need in the Exoscale portal:

1. Click on the "Account" icon on the left after logging into the Exoscale portal.
2. Click to open the "API Keys" tab.
3. You will need the values of the "API Key" and "Secret Key" fields for the SlipStream configuration.



[redacted]@sixsq.com
47.59 CHF

[redacted]@sixsq.com

Your API credentials for your personal account

API Key
[redacted]

Secret Key
[redacted]

Organization ID
[redacted]@sixsq.com

Compute API Endpoint
https://api.exoscale.ch/compute

API Keys and API Endpoints allow you to access all services on the Exoscale platform programmatically.

Add your Exoscale credential to your Nuvla account for the Exoscale regions defined there by repeating steps 3. and 4. for "exoscale-ch-gva" and "exoscale-ch-dk" connector instances.

1. Open your user profile (top-right, under your username).
2. Click on "Edit".
3. Open the section "exoscale-ch-gva" (or "exoscale-ch-dk") by clicking on the section header.
4. Provide the "API Key" value in the "Key" field and the "Secret Key" value in the "Secret" field.
5. Click on "Save".

You will now be able to use two regions of the Exoscale cloud with your account through Nuvla.

Tip: The optional "domain" parameter allows you to delegate control of a DNS zone that you own, to Exoscale. **This parameter for advanced use cases can be left blank.**

EXERCISES

1. Follow the Exoscale registration procedure to obtain an account.
2. Add your Exoscale credentials to your user profile.
3. Set the "Default cloud" parameter to Exoscale in your user profile.

Workstation Configuration

You will need to have the following software/tools installed and configured on your workstation to follow the exercises in this tutorial.

- Modern web browser: any recent version of one of the major browsers will be fine. **You must have Javascript enabled.**
- Advanced REST client: Install the latest version of Chrome and install the "Advanced REST client" extension.
- Secure Shell (SSH) client: This comes by default on most operating systems. With Windows, you'll need to install PuTTY.
- SSH Key: For the SSH connections, you'll need to have an SSH public/private keypair.

See the Appendix of this tutorial for more detailed information for the installation and configuration of these tools.

EXERCISES

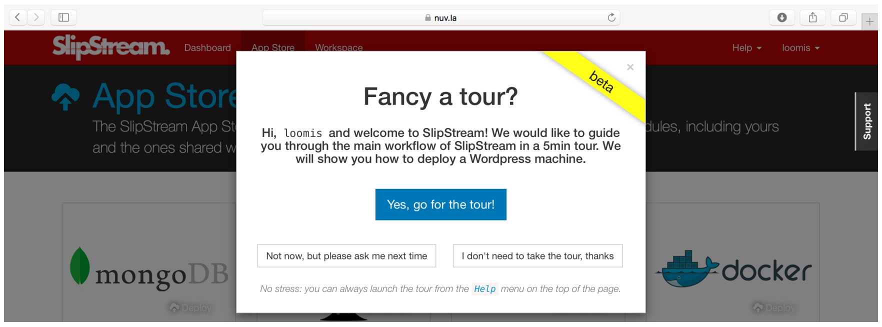
1. Configure your workstation with an SSH client and provide your SSH **public key** under the “General” section of your user profile.
2. Verify that you can start the Advanced REST client on Chrome.

Web Interface Tour

We postponed going through the inline tour of the SlipStream interface to allow us to concentrate on the configuration of our cloud credentials. We will go through that tour now because it:

- Provides a good overview of the major sections of the SlipStream web interface and
- Verifies that our user profile and cloud accounts have been configured correctly.

To activate the tour, choose the “Start guided tour” option below the “Help” menu. You should see a splash screen like the following:



Fancy a tour?

Hi, `loomis` and welcome to SlipStream! We would like to guide you through the main workflow of SlipStream in a 5min tour. We will show you how to deploy a Wordpress machine.

No stress: you can always launch the tour from the `Help` *menu on the top of the page.*

Click on the “Yes, go for the tour!” button to follow the tour. This will lead you through the interface and show you how to deploy a simple application from the App Store.

The tour is aimed at “Alice”, but it is useful also for developers (“Clara”) to see how the App Store can simplify the deployment of cloud applications for end-users.

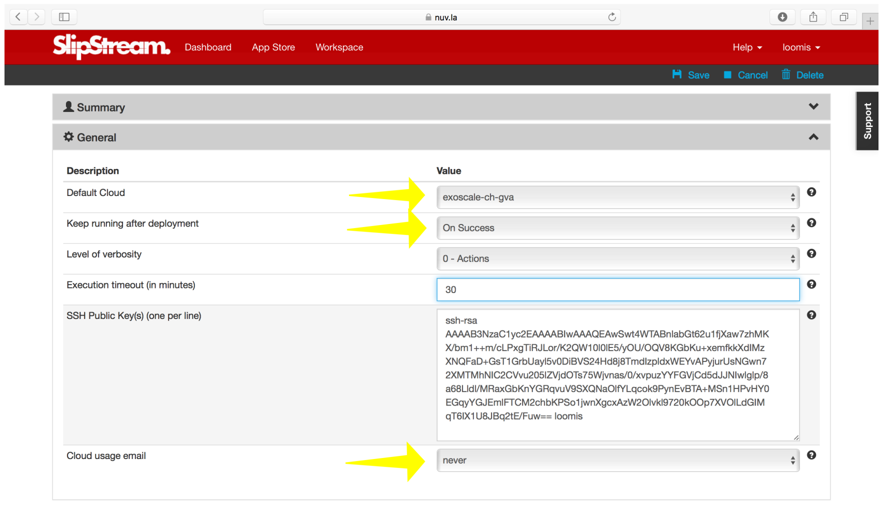
Note: In the tour workflow, you were redirected to the run page after you started an application. In the normal workflow, you are redirected to the dashboard instead.

EXERCISES

1. Follow the tour and verify that the Wordpress deployment works for your account.
2. Correct any problems that you encounter. Be sure to terminate your Wordpress deployment.

User Profile

You already saw and updated your user profile when you were configuring your account earlier. This page contains all of your user parameters, several of which are worth pointing out explicitly.



Description	Value
Default Cloud	exoscale-ch-gva
Keep running after deployment	On Success
Level of verbosity	0 - Actions
Execution timeout (in minutes)	30
SSH Public Key(s) (one per line)	ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAwSwt4WTABnlabGt62u1fjXaw7zhMKX/bm1++m/cLPxgTiRJLor/K2QW10l0lE5/yOU/OQV8KGbKu+xemfkkXdIMzXNQFaD+GsT1GrbUayl5v0DiBVS24Hd8j8TmdlzpldxWEYvAPyjurUsNGwn72XMTMhNIC2CVvu205lZVjdOTs75Wjvnas/0/xvpuzYYFGVjCd5dJJNIwlglp/8a68Lldl/MRaxGbKnYGRqvuV9SXQNaOlfYLqcok9PynEvBTA+MSn1HPvHY0EGqyYGJEmlFTCM2chbKPSo1jwnXgcxAzW2Olvkl9720kOOp7XVOlLdGIMqT6lX1U8JBq2tE/Fuw== loomis
Cloud usage email	never

Important Parameters

Default Cloud Parameter

The "Default cloud" parameter indicates which cloud infrastructure will be used by default for your applications unless you specify another cloud explicitly.

Keep Running Parameter

The "Keep running after deployment" indicates what SlipStream will do by default when it is finished deploying an application onto a cloud infrastructure. The default is "On success". The possible values are:

- **"On success"** Leave the application running until you explicitly terminate it.
- **"On error"** Leave the application running only in the case of an error to allow you to debug the problems.
- **"Never"** Always terminate the application.
- **"Always"** Always leave the application running.

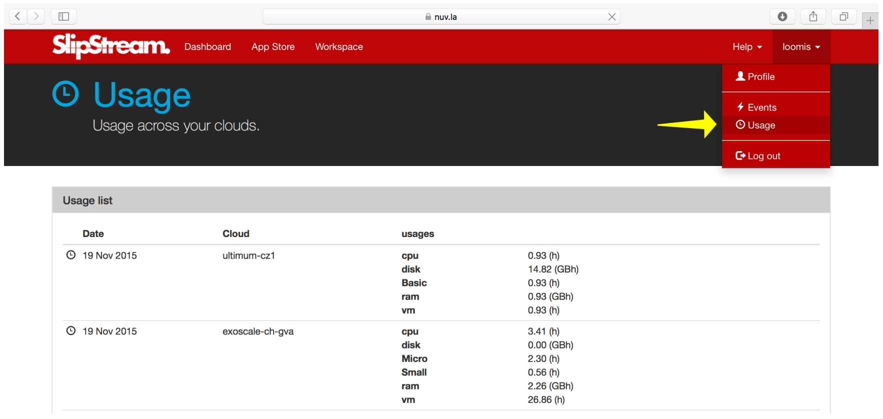
Pay attention to values that leave the applications running. They will continue to use cloud resources (and incur charges) until you explicitly terminate them.

Usage Email Parameter

The third parameter of interest is the "Cloud usage email" option. You can choose either "daily" or "never". If you choose daily, you'll receive a daily reminder of your cloud resource usage. This can be useful reminder to stop applications that you've forgotten about!

Usage

In the menu under your username, you can also find the "Usage" page. This page gives you a summary of your cloud usage (per day) over time. This is the information that will be emailed to you if you activated that option.



Usage

Usage across your clouds.

Usage list

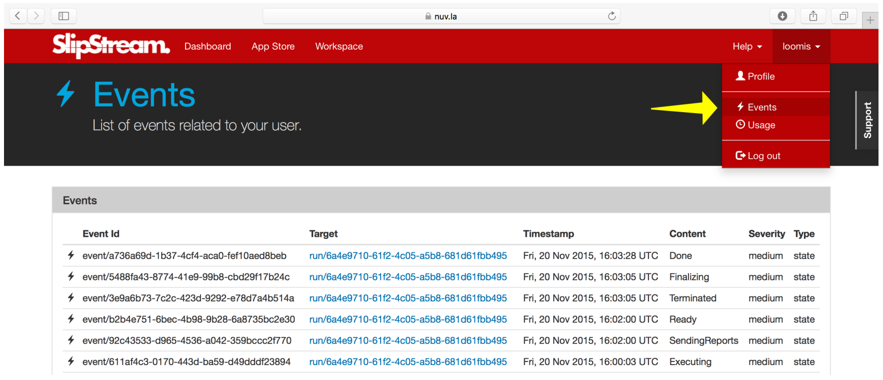
	Date	Cloud	usages	
	19 Nov 2015	ultimum-cz1	cpu	0.93 (h)
			disk	14.82 (GBh)
			Basic	0.93 (h)
			ram	0.93 (GBh)
			vm	0.93 (h)
	19 Nov 2015	exoscale-ch-gva	cpu	3.41 (h)
			disk	0.00 (GBh)
			Micro	2.30 (h)
			Small	0.56 (h)
			ram	2.26 (GBh)
			vm	26.86 (h)

The usage is calculated each morning, so it will initially be empty. It should show some activity tomorrow!

Events

SlipStream records events for important changes in the application lifecycle. These events can be used to understand the timeline of a given application deployment. All of the events related to your account can be seen on the "Events" page, which can be found in the menu under your username.

The page will look like the following screenshot. You should have events in the list related to the deployment of Wordpress from the web interface tour.



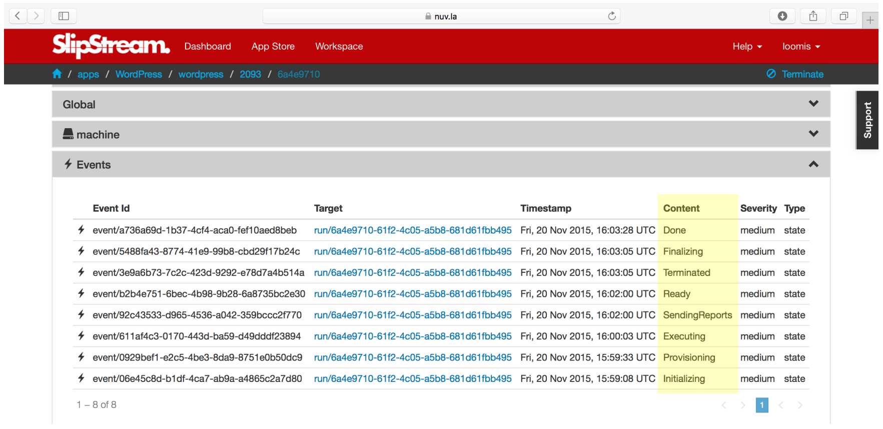
Events

List of events related to your user.

Events

Event Id	Target	Timestamp	Content	Severity	Type
event/a736a69d-1b37-4cf4-aca0-fef10aed8beb	run/6a4e9710-61f2-4c05-a5b8-681d61fbb495	Fri, 20 Nov 2015, 16:03:28 UTC	Done	medium	state
event/5488fa43-8774-41e9-99b8-cbd29f17b24c	run/6a4e9710-61f2-4c05-a5b8-681d61fbb495	Fri, 20 Nov 2015, 16:03:05 UTC	Finalizing	medium	state
event/3e9a6b73-7c2c-423d-9292-e78d7a4b514a	run/6a4e9710-61f2-4c05-a5b8-681d61fbb495	Fri, 20 Nov 2015, 16:03:05 UTC	Terminated	medium	state
event/b2b4e751-6bec-4b98-9b28-6a8735bc2e30	run/6a4e9710-61f2-4c05-a5b8-681d61fbb495	Fri, 20 Nov 2015, 16:02:00 UTC	Ready	medium	state
event/92c43533-d965-4536-a042-359bccc2f770	run/6a4e9710-61f2-4c05-a5b8-681d61fbb495	Fri, 20 Nov 2015, 16:02:00 UTC	SendingReports	medium	state
event/611af4c3-0170-443d-ba59-d49dddf23894	run/6a4e9710-61f2-4c05-a5b8-681d61fbb495	Fri, 20 Nov 2015, 16:00:03 UTC	Executing	medium	state

When trying to see events related to a particular application deployment (run), visit the run page. It contains a section with just the events for that application deployment.



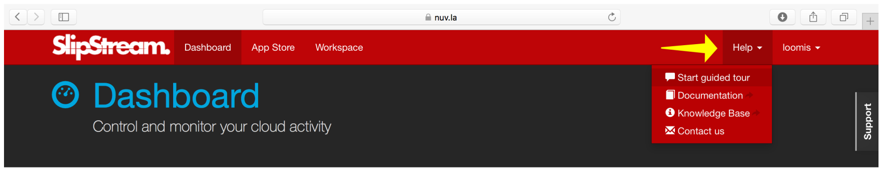
Event Id	Target	Timestamp	Content	Severity	Type
event/a736a69d-1b37-4cf4-aca0-fef10aed8beb	run/6a4e9710-61f2-4c05-a5b8-681d61fbb495	Fri, 20 Nov 2015, 16:03:28 UTC	Done	medium	state
event/5488fa43-8774-41e9-99b8-cbd29f17b24c	run/6a4e9710-61f2-4c05-a5b8-681d61fbb495	Fri, 20 Nov 2015, 16:03:05 UTC	Finalizing	medium	state
event/3e9a6b73-7c2c-423d-9292-e78d7a4b514a	run/6a4e9710-61f2-4c05-a5b8-681d61fbb495	Fri, 20 Nov 2015, 16:03:05 UTC	Terminated	medium	state
event/b2b4e751-6bec-4b98-9b28-6a8735bc2e30	run/6a4e9710-61f2-4c05-a5b8-681d61fbb495	Fri, 20 Nov 2015, 16:02:00 UTC	Ready	medium	state
event/92c43533-d965-4536-a042-359bccc2f770	run/6a4e9710-61f2-4c05-a5b8-681d61fbb495	Fri, 20 Nov 2015, 16:02:00 UTC	SendingReports	medium	state
event/611af4c3-0170-443d-ba59-d49dddf23894	run/6a4e9710-61f2-4c05-a5b8-681d61fbb495	Fri, 20 Nov 2015, 16:00:03 UTC	Executing	medium	state
event/0929bef1-e2c5-4be3-8da9-8751e0b50dc9	run/6a4e9710-61f2-4c05-a5b8-681d61fbb495	Fri, 20 Nov 2015, 15:59:33 UTC	Provisioning	medium	state
event/06e45c8d-b1df-4ca7-ab9a-a4865c2a7d80	run/6a4e9710-61f2-4c05-a5b8-681d61fbb495	Fri, 20 Nov 2015, 15:59:08 UTC	Initializing	medium	state

1 – 8 of 8

This page is also instructive because is shows the complete list of application states. From the events, you can also understand how much time is spent in each state.

Help

You can also get help, by consulting the SlipStream documentation, Knowledge Base, or by contacting SixSq directly.



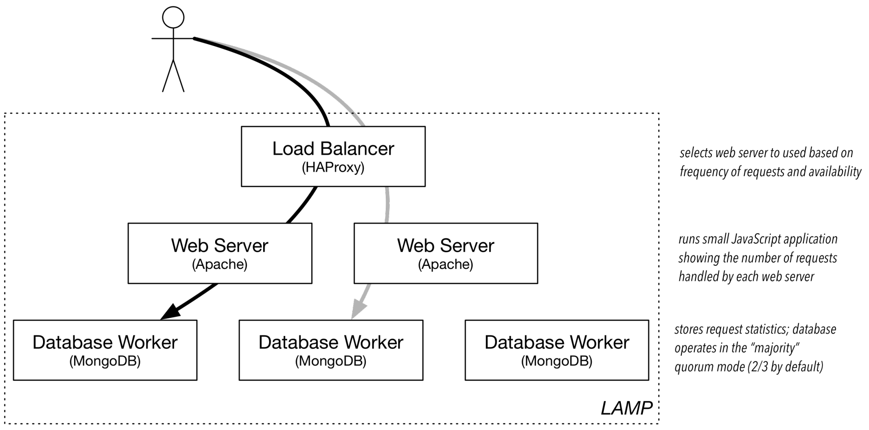
3-Tier Web App (LAMP)

SlipStream can efficiently manage simple applications like Wordpress, but it really shines when trying to deploy large, multi-machine applications.

Deployment of those types of applications, for example 3-tiered web applications, can be complex and error prone. SlipStream automates these deployments, ensuring both consistency and reliability.

Description

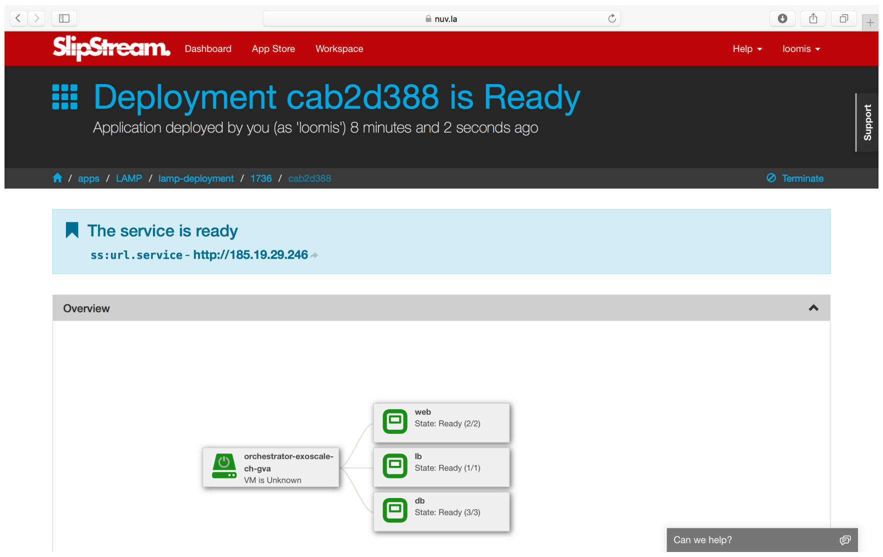
The LAMP++ (Linux, Apache, MongoDB, and PHP) application is an example 3-tier web application that uses a load balancer to distribute requests through multiple web front-ends and a distributed MongoDB database. (The definition can be found in the module.)


Load Balancer
(HAProxy)
Web Server
(Apache)
Web Server
(Apache)
Database Worker
(MongoDB)
Database Worker
(MongoDB)
Database Worker
(MongoDB)
LAMP
selects web server to used based on frequency of requests and availability
runs small JavaScript application showing the number of requests handled by each web server
stores request statistics; database operates in the "majority" quorum mode (2/3 by default)

The previous diagram shows the components of the LAMP application. The web application simply displays the current request statistics.

Operation

The run page for this application shows its deployment topology.



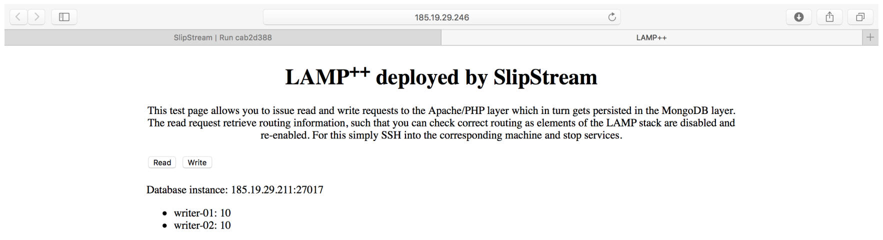
Deployment cab2d388 is Ready

Application deployed by you (as 'loomis') 8 minutes and 2 seconds ago

The service is ready

ss:url.service - http://185.19.29.246

Following the link to the deployed application, one can see which database node is being accessed and the distribution of write requests between the two front-end servers. The requests are roughly distributed uniformly between the front end servers.



LAMP++ deployed by SlipStream

This test page allows you to issue read and write requests to the Apache/PHP layer which in turn gets persisted in the MongoDB layer. The read request retrieve routing information, such that you can check correct routing as elements of the LAMP stack are disabled and re-enabled. For this simply SSH into the corresponding machine and stop services.

Database instance: 185.19.29.211:27017

- writer-01: 10
- writer-02: 10

Robustness

This LAMP application is resiliant to failure. This can be shown for instance by logging into one of the front-end nodes and turning off the apache server. In this case all of the requests will go through a single front end.

The same test can be done with the MongoDB nodes. If one is removed, the system should still function normally. However, since the system is running with a quorum of 2, the service will fail if two of the MongoDB nodes are stopped.

EXERCISES

1. Deploy the LAMP example and make sure that the application shows the request statistics and that the load balancer switches between nodes.
2. Turn off the Apache server on one of the web front ends. The command is `service apache2 stop`. Then click the read/write buttons to verify that only one web front-end is responding.
3. Kill one of the MongoDB nodes and verify that writes to the database will work correctly.
4. Kill a second MongoDB node. In this case, the database should stop responding because it has fallen below its configured quorum of two nodes.

In this module you've acquired enough knowledge about SlipStream and its basic capabilities. After working through this module, you will be ready to proceed to *Module III*, where you will learn how to build, deploy, and manage your own applications with SlipStream on multiple clouds.

3.1.3 Module III - Creating, Deploying, and Managing Cloud Applications (Basic)

This module provides detailed information about the creation, deployment and management of applications. Once you've completed it, you'll be able to build, deploy, and share your own applications on clouds of your choice.

This module builds on the knowledge acquired about Slipstream in the previous module *Module II*. Your user profile must contain credentials for at least one cloud. If necessary, review the *Prerequisites* from the previous module.

Workspace

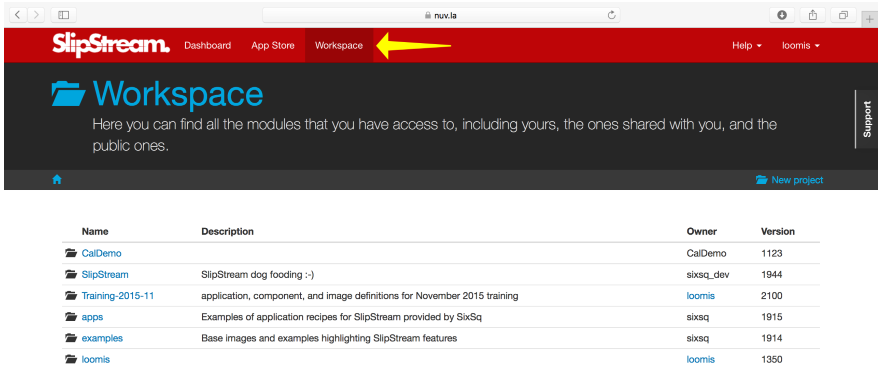
The workspace allows you to create, store, and organize your projects and modules (image, component, and application definitions).

In this section you'll learn how to:

- Create and navigate projects,
- Protect or share your projects and modules, and
- Understand versioning within SlipStream.

Projects

At the top-level of the workspace, you will see your projects as well as any projects that have been shared with you.



Here you can find all the modules that you have access to, including yours, the ones shared with you, and the public ones.

Name	Description	Owner	Version
CalDemo		CalDemo	1123
SlipStream	SlipStream dog fooding :-)	sixsq_dev	1944
Training-2015-11	application, component, and image definitions for November 2015 training	loomis	2100
apps	Examples of application recipes for SlipStream provided by SixSq	sixsq	1915
examples	Base images and examples highlighting SlipStream features	sixsq	1914
loomis		loomis	1350

"Projects" are essentially folders, in which you can keep related images, components, and applications together. As projects can contain other projects, you can create hierarchies to help organize your work.

There are two important projects that are shared with everyone:

- The "examples/images" project contains basic operating system images (like Ubuntu 14.04 or 16.04 and CentOS 6 or 7) that are reused extensively in the definition of application components.
- The "apps" project contains set of application definitions created by SixSq to highlight features of SlipStream and to serve as a basis for your own applications.

When you have time, it is instructive to look through the definitions of those components and applications to understand how to take advantage of SlipStream features.

Important: Because the **root of the workspace is shared by all users**, it is recommended that you create your own top-level project (usually with the same name as your username) to hold your private modules.

Access Control

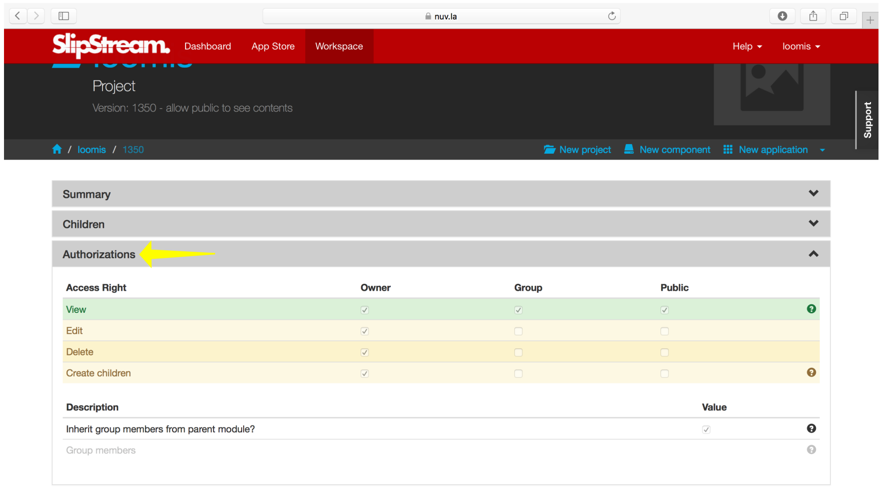
SlipStream has a uniform access control model across all modules. Permissions can be defined separately for three categories of users:

- **User** is the owner of the module
- **Group** is a list of other users
- **Public** is the set of all **authenticated** users

The list of users for a group can be defined explicitly on a module or inherited from the enclosing project.

Important: Modules may optionally inherit the group definition (but not permissions) from its enclosing project (grand-project, etc.). Nothing other than the group definition can be inherited from a project.

The available permissions are different depending on the type of module. The following screenshot shows the available permissions for a typical project. The access control information is always visible in the "Authorizations" section of the module.



Access Right	Owner	Group	Public
View	✓	✓	✓
Edit	✓	☐	☐
Delete	✓	☐	☐
Create children	✓	☐	☐

Description	Value
Inherit group members from parent module?	✓
Group members	

Using the access control mechanisms, you can **share your work with other people on the SlipStream server**. You can also request that the SlipStream administrator publish your module in the App Store to make it more visible to others.

Important: When you share a module with someone else, you allow them to see and optionally execute the application definition. If they run the application, it will use **their cloud accounts** and not yours. You're sharing your knowledge, not your credit card!

Versioning

The full history for all modules is kept by the SlipStream server; each saved modification of a module is associated with a unique version number. The version number and a link to the full history is available in the "Summary" section of the module.

Tip: Every time you save a module, you may provide an optional description of your changes. These comments are available in the module history and help understand the evolution of the module. Providing these comments is best practice!

EXERCISES

1. Create your top-level project.
2. Verify with your neighbors that you can see your own project but not theirs.
3. Change the permissions on the module (either Group or Public) and then verify that others can see your project.
4. Change your project back to a private module when your finished.
5. View the history of your project, containing the versions and comments. Are the version numbers sequential?

Images

An image defines a virtual machine image that encapsulates cloud-specific information. **All** application components must eventually reference an image that can be provisioned in the selected cloud(s).

In this section you'll learn how to:

- Reference cloud-specific images,
- Change the resource allocations,
- Define the security group, and
- Deploy an image through SlipStream.

"Native" Images

To have full application portability between clouds, you must have identical or functionally equivalent virtual machine images available in each of the clouds.

Identical Images

One possibility is to create your own virtual machine images and upload them into every cloud you want to use. In theory, this is the best approach. In practice as with so many ideas, it doesn't work well because:

- Some cloud providers do not allow user-created images,
- Other cloud providers modify uploaded images (e.g. by changing the kernel), and
- Maintaining images for every cloud becomes a significant burden.

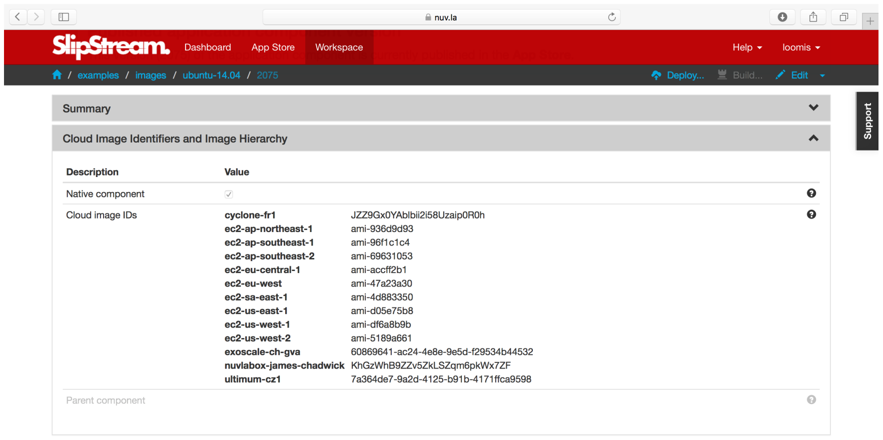
The last point is particularly troublesome because making images (often tens of Gigabytes in size) and uploading them is time-consuming. By doing this, we'd lose the dynamicity that is a fundamental for switching to cloud technologies.

Identical Recipes

SlipStream's approach overcomes these problems by reusing existing, functionally equivalent images supplied by the cloud providers, then maintaining common recipes for customizing them.

A SlipStream image definition references existing ("native") virtual machine images in each cloud, creating a group of functionally equivalent images (e.g. a "minimal Ubuntu 14.04" image). The slight differences between the images in various clouds rarely (if ever) have noticable effects on the cloud application and we free ourselves from maintaining our own images.

If you look at the definition of the image, you'll see the list of native image identifiers in each cloud in the "Cloud Image Identifiers and Image Hierarchy" section.

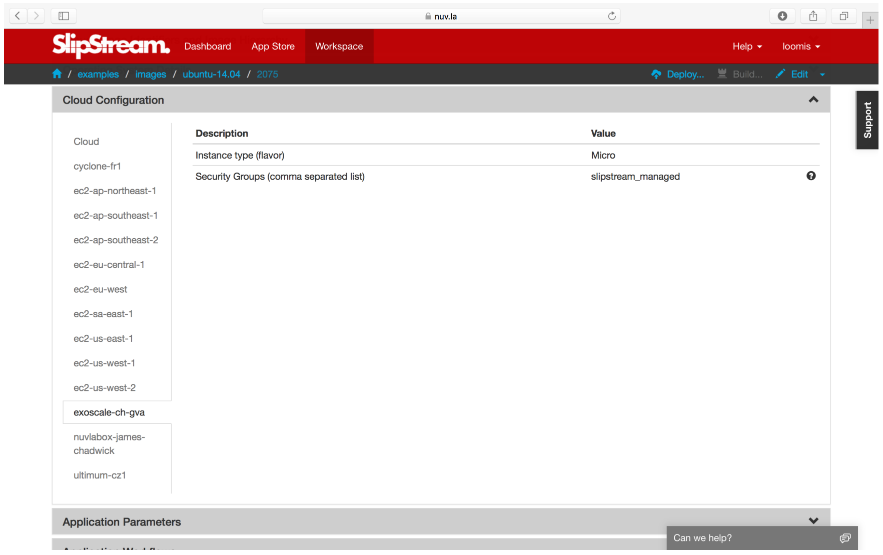


Description	Value	
Native component	✓	
Cloud image IDs	**cyclone-fr1**	JZZ9Gx0YAblbii2i58Uzaip0R0h
	ec2-ap-northeast-1	ami-936d9d93
	ec2-ap-southeast-1	ami-96f1c1c4
	ec2-ap-southeast-2	ami-69631053
	ec2-eu-central-1	ami-accff2b1
	ec2-eu-west	ami-47a23a30
	ec2-sa-east-1	ami-4d883350
	ec2-us-east-1	ami-d05e75b8
	ec2-us-west-1	ami-df6a8b9b
	ec2-us-west-2	ami-5189a661
	exoscale-ch-gva	60869641-ac24-4e8e-9e5d-f29534b44532
	nuvlabox-james-chadwick	KhGzWhB9ZZv5ZkLSZqm6pkWx7ZF
	ultimum-cz1	7a364de7-9a2d-4125-b91b-4171ffca9598
Parent component		

When creating a new image, you'll need to find the appropriate image identifiers through the cloud provider's interface. For Exoscale, the identifiers are available from a .

VM Size

Very often you will want to tailor the CPU, RAM, and other resources allocated to the machine. Specifying the "size" of a virtual machine is another area where the cloud providers differ. Within an image, the size is defined, per-cloud, in the "Cloud Configuration" section (see).



Description	Value
Instance type (flavor)	Micro
Security Groups (comma separated list)	slipstream_managed

Some providers allow you to specify the CPU and RAM resources explicitly; others only allow you to specify a "t-shirt" size. **You will have to consult documentation from the cloud providers to understand the allowed values and their meanings.** The following table shows the differing sizes for Exoscale and Ultimum (just another cloud brought here for a comparision).

Exoscale	Ultimum	CPU	RAM (GB)
Micro	–	1	0.5
Tiny	Basic	1	1
Small	Standard	2	2
Medium	–	2	4
–	Standard Plus	4	4
Large	–	4	8
Extra-large	–	4	16
–	Large	8	8
Huge	–	8	32
–	Large Plus	16	16
–	Extra Large	32	32

You will not be able to change the resource allocation for image definitions for the shared images (or more generally for any image not owned by you). However you can get around this by:

- Making a copy of the image within SlipStream and modifying your copy, or
- Updating the size in the components you create that reference a shared image.

We will see how values can be inherited or changed when we see what can be done with components.

Networking

Currently SlipStream takes a very simple approach to managing network connectivity to virtual machines. On clouds that support it, SlipStream will create a security group (set of cloud-level firewall rules, not VM Operating System firewall rules) called "slipstream_managed" that allows access on any port from anywhere.

When you use the standard shared image definitions, the "slipstream_managed" security group will be used, allowing the services on the machine to be accessed through the network. Note on the previous screenshot that there is a parameter to specify what security group(s) to use.

You can more tightly secure your deployed applications by:

- Running a firewall within your images (and components) and/or
- Specifying a different security group in your image definitions (the referenced security groups should be present on the cloud).

In production, you should take every opportunity to secure your running systems. In the interests of simplicity, this tutorial does not follow best practices in this respect.

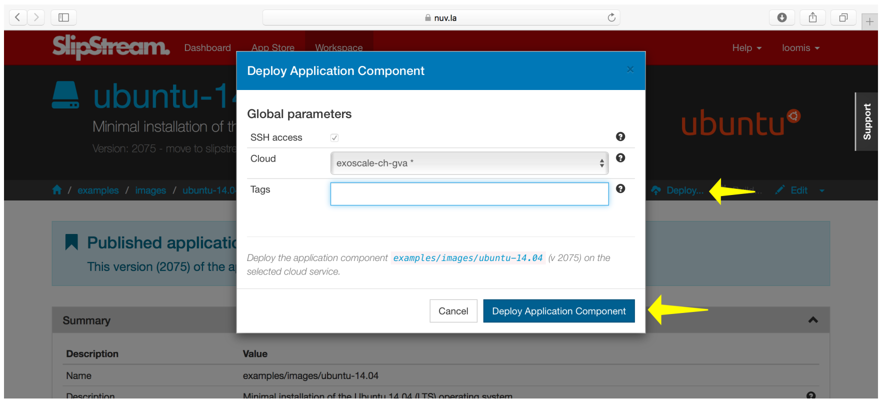
Important: If you do specify your own security group in your images, you need to maintain the same security group on all the clouds you use to ensure that you can switch between clouds easily.

Important: For clouds that do not support security groups (or their equivalent), you must manually adjust the networking parameters for the machines that are deployed.

Deploy a VM

At its simplest, SlipStream can be used as a multi-cloud VM management console. To show how this is done, navigate to the `examples/images` project, which contains a set of minimal images you can use.

Clicking on the `ubuntu-14.04` module and then on "Deploy...", you should see a screenshot like the following.



ubuntu-1

Minimal installation of th

Version: 2075 - move to slipstre

/ examples / images / ubuntu-14.0

Published applicati

This version (2075) of the a

Deploy Application Component

Global parameters

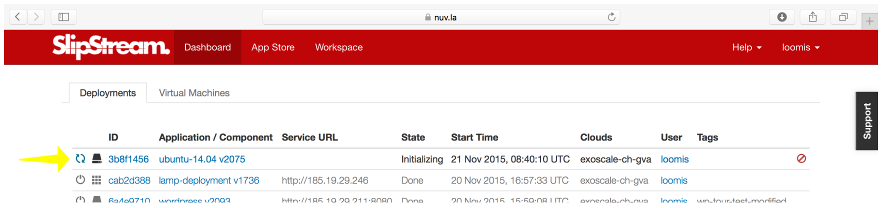
SSH access	✓
Cloud	exoscale-ch-gva *
Tags	

Deploy the application component `examples/images/ubuntu-14.04` *(v 2075) on the selected cloud service.*

Summary

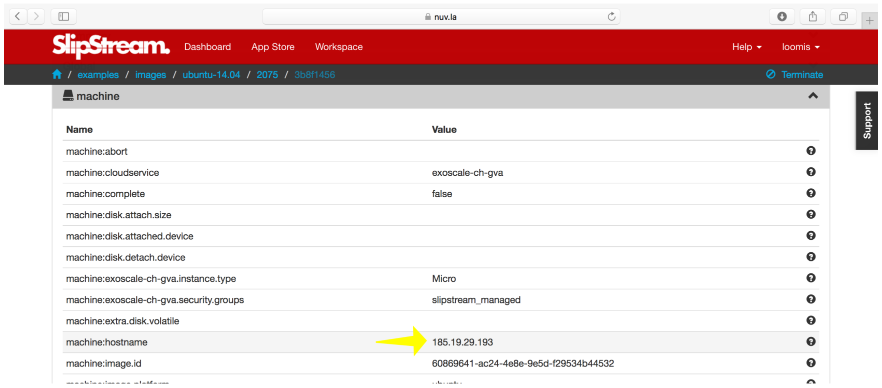
Description	Value
Name	examples/images/ubuntu-14.04
Description	Minimal installation of the Ubuntu 14.04 (LTS) operating system

From the run dialog you can choose the cloud to use and then deploy the image by clicking on the dialog's "Deploy..." button. This will redirect you to the dashboard, where you will see a new entry for the image.



ID	Application / Component	Service URL	State	Start Time	Clouds	User	Tags
3b8f1456	ubuntu-14.04 v2075		Initializing	21 Nov 2015, 08:40:10 UTC	exoscale-ch-gva	loomis	
cab2d388	lamp-deployment v1736	http://185.19.29.246	Done	20 Nov 2015, 16:57:33 UTC	exoscale-ch-gva	loomis	
6a4e9710	wordpress v2093	http://185.19.29.211:8080	Done	20 Nov 2015, 15:59:08 UTC	exoscale-ch-gva	loomis	wp-tour-test-modified

You can either follow the progress of the machine from the dashboard or click on the "ID" to see the more detailed run page. On the run page, you can find the IP address of the machine and an SSH link in the "machine" section.



/ examples / images / ubuntu-14.04 / 2075 / 3b8f1456

machine

Name	Value
machine:abort	
machine:cloudservice	exoscale-ch-gva
machine:complete	false
machine:disk.attach.size	
machine:disk.attached.device	
machine:disk.detach.device	
machine:exoscale-ch-gva.instance.type	Micro
machine:exoscale-ch-gva.security.groups	slipstream_managed
machine:extra.disk.volatile	
machine:hostname	185.19.29.193
machine:image.id	60869641-ac24-4e8e-9e5d-f29534b44532
machine:image.platform	ubuntu

Or you can log in manually from the command line, using the username and IP address on the run page:

```
advanced_tutorial> ssh root@185.19.29.193
[...]
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.13.0-49-generic x86_64)
[...]
root@machine3b8f1456-cb5d-49ba-b7d5-430f97642850:~#
```

Or if your browser is configured for SSH links, you can click on the link for the parameter "machine:url.ssh".

Important: Note that the username may change depending on the underlying native image being used. The usernames "root" and "ubuntu" are common. The username may not be correct in the parameters as they are not always uniform across cloud providers.

Running simple virtual machines in this way is often useful for quick debugging or developing service installation recipes.

EXERCISES

1. **Create a copy of the Ubuntu 14.04 image and change the size for either** Exoscale exoscale-ch-gva or exoscale-ch-dk. Leave the other unchanged.
2. **Deploy the copy on both Exoscale exoscale-ch-gva or exoscale-ch-dk. Note** the deployment times.
3. Verify that you can log into both machines with SSH.
4. Verify that the number of CPUs (`/proc/cpuinfo`) and the RAM (`/proc/meminfo`) are the values expected from the size.
5. Try to deploy and access a web server on the deployed machine. The commands to use are:

   ```
   $ apt-get update
   $ apt-get install -y nginx
   $ service nginx start
   ```

   You should then be able to navigate to "http://*ip_address*/" and see an nginx welcome page. You may have to add `sudo` before these commands, if you had to log in with the "ubuntu" account rather than "root".

Components Part I

A minimal operating system isn't very useful by itself. We want to transform such a simple image into a component running services that we need for our cloud application.

In this section you'll learn how to:

- Automate the installation and configuration of services,
- Parameterize the component to allow deploy-time configuration, and
- Pass information from the application back to the user.

In doing this you'll also understand:

- How to reference images from components,
- The purpose of the various SlipStream "recipes", and
- When those recipes are executed.

In this chapter, we'll create a web server component that can be customized when deployed and that protects a "secret" with basic authentication.

Important: The command line examples throughout the tutorial use Linux bash syntax. If you're using a different shell (csh, PowerShell, etc.), you'll need to adjust the syntax accordingly.

Similarly, the example applications developed in the tutorial are uniformly built over the Ubuntu operating system. This is to reduce distractions and make the tutorial as uniform as possible. SlipStream supports a wide range of different operating systems, including Windows.

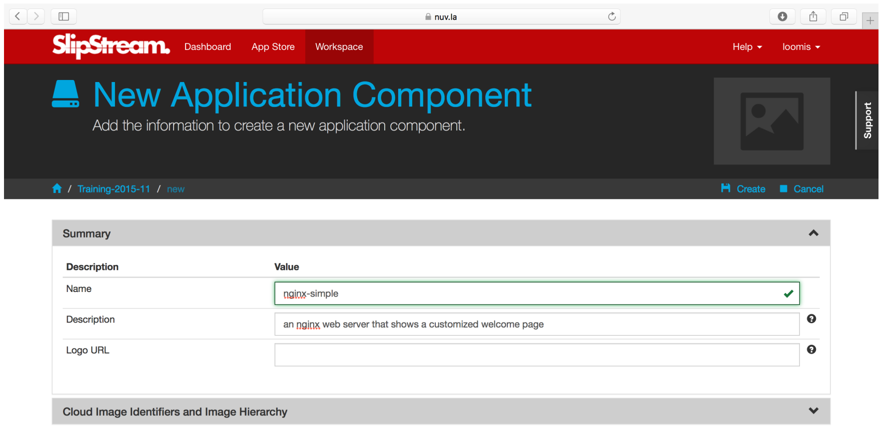
Web Server Component

It wouldn't be practical to have to log into every deployed virtual machine to install software and to configure it. This process needs to be automated to allow people to concentrate on less tedious tasks and to ensure that the process is consistent and error-free.

Let's start by creating a new web server component that will automatically install and configure nginx, a standard web server.

Create Component

Navigate to your home project and then click on the menu item to create a "New component". You should then see a page that looks like the following screenshot.

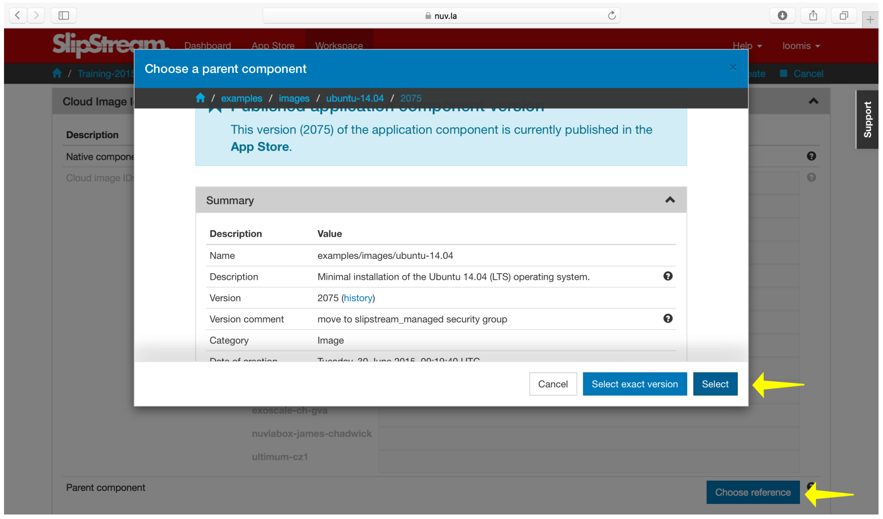

nuv.la
Dashboard
App Store
Workspace
Help
loomis
New Application Component
Add the information to create a new application component.
Support
Training-2015-11 / new
Create
Cancel
Summary
Cloud Image Identifiers and Image Hierarchy

Description	Value
Name	nginx-simple
Description	an nginx web server that shows a customized welcome page
Logo URL	

Provide values for the name and description, but don't save the component yet.

Reference Native Image

Open the "Cloud Image Identifiers and Image Hierarchy" section. At the bottom of this section we want to click on the button that says "Choose reference". This allows us to choose the native image that we want to use for our component. When you click the button, you will see a "chooser" dialog, like in the following screenshot.



This version (2075) of the application component is currently published in the **App Store**.

Description	Value
Name	examples/images/ubuntu-14.04
Description	Minimal installation of the Ubuntu 14.04 (LTS) operating system.
Version	2075 (history)
Version comment	move to slipstream_managed security group
Category	Image
Date of creation	Tuesday, 30 June 2015, 09:19:40 UTC

Navigate to the `examples/images/ubuntu-14.04` and click on the "Select" button. This will add the reference to the image description. **Do not click on "Create" yet.**

Warning: There are two buttons to choose from in the chooser. What's the difference?

- **Select**: Chooses the given image and will use the **latest** version of the image when the component is deployed.
- **Select exact version**: Chooses the given image and will always use this exact version when deploying the component.

With the floating version, the component will always take advantage of improvements to the referenced image, with a slight possiblity of running into breaking changes.

Locking the version avoids this problem, but you may run into the case where the referenced image has been removed by the provider.

Generally, you will almost always want to choose "Select"!

Add Nginx

To get our web server installed and running, we need to:

- Install the nginx software
- Configure the nginx server to start automatically, and
- Add our customized web page

To accomplish these tasks we want to add the appropriate "recipes" or "hooks" to the component definition. Open the "Application Workflows" section of the component. Along the left edge you'll see the set of recipes that you can add. They are essentially run in the order that they are listed.

Important: What type of information do you add to each recipe? Here is the general guide:

- **Pre-install** This is the first recipe to be executed. It will be run before SlipStream installs anything else on the machine (either the SlipStream client or specified packages). This can be used, for instance, to update the package manager configuration on the machine or to upgrade the system.
- **Install packages** This recipe is a list of packages to be installed on the machine. SlipStream will use the appropriate package manager for the operating system, which normally will also install any dependencies. This only supports Debian and RedHat families of operating systems. If you're using something else (e.g. Windows), install packages manually with the "Post-install" recipe.
- **Post-install** This recipe should be used for any **static** configuration of the machine. That is configuration that will never need to be changed during the deployment or operation of the machine.
- **Deployment** Dynamic configuration of the machine should be handled in this recipe. This includes configuration based on the component's parameters and this is where one would use SlipStream CLI/API for orchestrating the deploment between components.
- **Reporting** This will be executed when gathering up the reports from the deployment. In addition to the usual files, you can add additional files to be copied back to the SlipStream server.
- **On VM add** A recipe which is executed when an application containing the component is "scaling up", that is adding new resources. Ignore this recipe for now, you'll learn more about it in the application scaling chapter later.

- **On VM remove** A recipe which is executed when an application containing the component is "scaling down", that is removing existing resources. Ignore this recipe for now, you'll learn more about it in the application scaling chapter later.
- **Pre-Scale** Runs on the node instance before scale IaaS action (any vertical (subject to implementation by connector) and only horizontal down scaling). Ignore this recipe for now, you'll learn more about it in the application scaling chapter later.
- **Post-Scale** Runs on the node instance after scale IaaS action (only after vertical scaling (subject to implementation by connector)). Ignore this recipe for now, you'll learn more about it in the application scaling chapter later.

Using the recipe for installing the nginx server from before, add the following to the "Pre-install" recipe:

```
#!/bin/bash -xe
apt-get update -y
```

which will update the configuration of the package manager.

Then add the package "nginx" to the "Install packages" recipe. Nginx is a high-performance web server.

In the "Post-install" recipe, we want to create our customized welcome page, ensure that the nginx server is started, and that nginx always starts when the machine boots. Add the following:

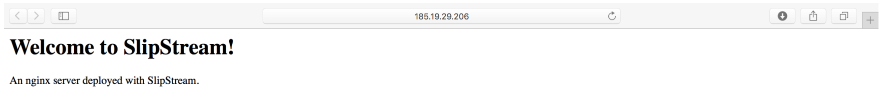
```
#!/bin/bash -xe

# remove default site and create our own
rm -f /etc/nginx/sites-enabled/default
cat > /etc/nginx/sites-enabled/mysite <<EOF
server {
  listen 80 default_server;

  root /var/www/html;
  index index.html;
}
EOF

# customize the welcome page
mkdir -p /var/www/html
cat > /var/www/html/index.html <<EOF
<html>
  <head>
    <title>Welcome to SlipStream!</title>
  </head>

  <body>
    <h1>Welcome to SlipStream!</h1>
    <p>An nginx server deployed with SlipStream.</p>
  </body>
</html>
EOF

# start web server on boot
update-rc.d nginx enable

# ensure web server is running with changes
service nginx restart
```

With these definitions you can now click on the "Create" button to create the component definition.

Warning: All of the recipes must be executable by the underlying operating system. Make sure that you've added the shebang line to all of the recipes `#!/bin/bash -xe` (or similar)!

Note: Using the "-xe" options on the shebang line helps with debugging when there are problems. The "-x" option will print each line in the script to the stdout before executing it. The "-e" option will stop the script on the first error.

You can then click on the "Deploy…" button to deploy the web server and ensure that it works as expected. When visiting the URL for the machine "http://*host_ip*/", you should see something like the following screenshot.



Welcome to SlipStream!

An nginx server deployed with SlipStream.

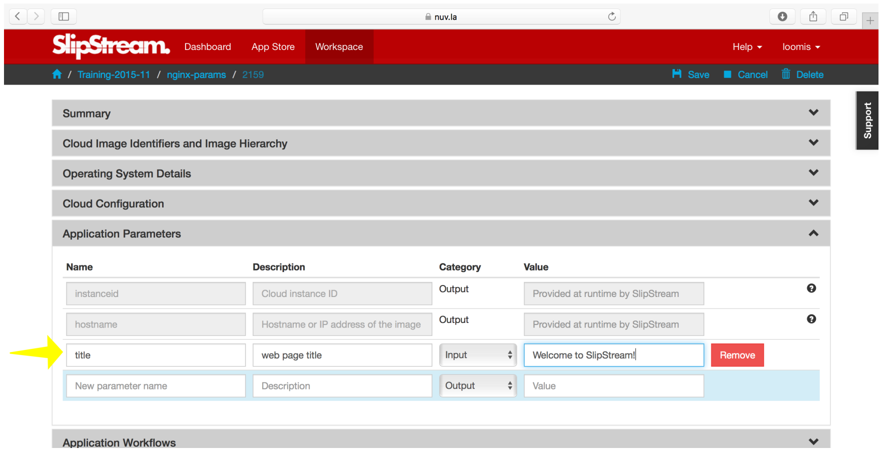
Parameterized Web Server

It wouldn't be very useful if we had to create a new component definition every time we wanted to change some behavior: like the location of a database, password for a server, descriptive text, etc. We want to parameterize the component to promote reuse. In this case we'll keep it simple and parameterize the title of the page.

At the same time, we'd like to provide more feedback (through SlipStream) about the state of the application and make it easy to find the deployed web server. We'll improve this in the next version of the component.

Title Parameter

Let's begin by defining an input parameter that allows the title to be specified. You can copy your previous component (look under the triangle next to the "Edit" button) or just modify the old one directly. Click "Edit" and then go to the "Application Parameters" section and add an **input** parameter called "title".



Name	Description	Category	Value
instanceid	Cloud instance ID	Output	Provided at runtime by SlipStream
hostname	Hostname or IP address of the image	Output	Provided at runtime by SlipStream
title	web page title	Input	Welcome to SlipStream!
New parameter name	Description	Output	Value

If you provide a value here, that will be the default value used when deploying the component. If you don't specify anything, then this will force the user to provide a value.

Now we need to modify the recipes to use the value of this parameter in the configuration. Change the welcome page definition in the "Post-install" recipe to this:

```
<html>
  <head>
    <title>__TITLE__</title>
  </head>

  <body>
    <h1>__TITLE__</h1>
    <p>An nginx server deployed with SlipStream.</p>
  </body>
</html>
```

We will then replace "__TITLE__" with the actual parameter value.

Deployment Configuration

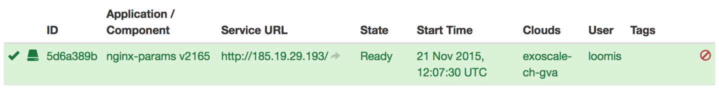
We must still add some deployment-time configuration in the "Deployment" recipe to take into account the parameter's value. In the deployment recipe add the following:

```
#!/bin/bash -xe

# get the value from slipstream
title="`ss-get title`"

# replace the title in the welcome page
sed -i "s/__TITLE__/${title}/g" /var/www/html/index.html

# provide a link to the webserver through slipstream
hostname=`ss-get hostname`
link=http://${hostname}/
ss-set ss:url.service ${link}

# provide status information through web UI
ss-display "Webserver ready on ${link}!"
```

This uses some magic commands that will be described in the next section. There is also some help for these commands below the editor window in the web interface.

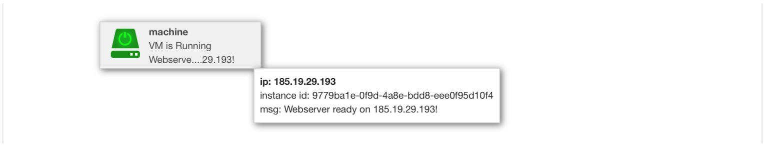
Now you can save the component and deploy it. When deploying it, you should see an input parameter in the run dialog. Change the value so that you can be sure that it was used in the configuration. Verify that it shows up in the welcome page.

In the dashboard, you should see that a service URL has been provided for the web server. This makes accessing the service much easier.



		ID	Application / Component	Service URL	State	Start Time	Clouds	User	Tags	
✓		5d6a389b	nginx-params v2165	http://185.19.29.193/	Ready	21 Nov 2015, 12:07:30 UTC	exoscale-ch-gva	loomis		⊘

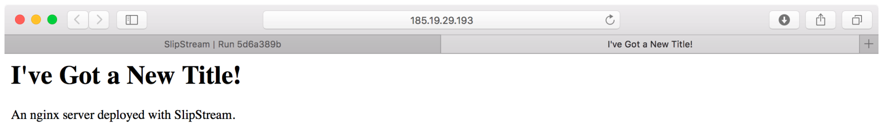
You can also see that an informative message has been displayed on the run page.



machine
VM is Running
Webserve....29.193!

ip: 185.19.29.193
instance id: 9779ba1e-0f9d-4a8e-bdd8-eee0f95d10f4
msg: Webserver ready on 185.19.29.193!

And finally, you should also see that the value of your title parameter has been taken into account.



I've Got a New Title!

An nginx server deployed with SlipStream.

Run Database

When you deploy a component (or later an application), SlipStream creates a mini-database of parameters which can be used to pass information into or out of the running component.

In this run database, there are some global variables that are always defined. One of these is the `ss:url.service` parameter, which is the service URL for the deployed component. The web interface picks up this value and displays it as a link in the dashboard and run page. All of the global variables are prefixed with `ss:`.

Note: In general, any parameter that starts with "url." will be interpreted by the web interface as a link and rendered as such. In addition to the service URL, there are also similar ones generated by default for SSH.

The `ss-display` command is a shortcut to set the `statecustom` parameter on a particular machine. You'll find this in the section for the machine on the run page. Notice that the input parameter we defined for the title, also shows up in the parameters of the machine.

As seen above the `hostname` is automatically defined by SlipStream for each node. This can reliably be used to recover the hostname of the machine running the recipe.

The commands such as `ss-set`, `ss-get`, etc. are installed automatically by SlipStream on the machine and can be used in the deployment recipe.

Warning: The `ss-*` commands are installed at the end of the post-install recipe. They **cannot** be used in the recipes that are executed earlier.

Secured Web Server

Enhance the web server to also serve a protected page that can only be accessed with a username and password. To do this you need to:

- Create a page that we want to protect,
- Modify the nginx configuration to use basic authentication,
- Create the credentials to access the page.

You'll need a utility from Apache to generate a username and password for the protected content. Add the package "apache2-utils" to the "Install packages" recipe.

In the "Post-install" recipe, update the server configuration:

```
cat > /etc/nginx/sites-enabled/mysite <<EOF
server {
  listen 80 default_server;

  root /var/www/html;
  index index.html;

```

(continues on next page)

(continued from previous page)

```
  location /protected {
    auth_basic "Restricted";
    auth_basic_user_file /etc/nginx/htpasswd;
  }
}
EOF
```

Add an empty password file and create a protected page:

```
# create empty password file
touch /etc/nginx/htpasswd

# provide a page with a secret
mkdir -p /var/www/html/protected
cat > /var/www/html/protected/index.html <<EOF
<html>
  <head>
    <title>SECRET</title>
  </head>

  <body>
    <h1>SECRET</h1>
    <p>This is a protected page; username and password required.</p>
  </body>
</html>
EOF
```

Update the deployment script to generate a random password:

```
# create an entry in the password file
username='nginx-user'
password=`ss-random`
htpasswd -bc /etc/nginx/htpasswd ${username} ${password}

# publish this information in slipstream
ss-set username ${username}
ss-set password ${password}
```

Notice that this publishes the username and password as parameters into SlipStream. You must define those parameters in the component definition or the deployment will fail. Add the "username" and "password" **output** parameters in the "Application Parameters" section.

This can now be saved and deployed. When it is available you should be able to see the old welcome page and see the secret page at http://*host_ip*/protected/ if you provide the username and password. The values of those will be published in the parameters on the run page.

Note: As generating a password is fairly common for securing services, the SlipStream client provides the `ss-random` command to facilitate this. Generating a password like this, allows the running instance to be accessible only to its owner, while the component definition can be shared.

EXERCISES

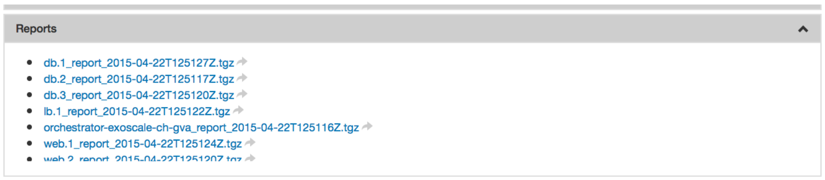
1. Create the simple web server and verify that it works.
2. Parameterize the web server and verify that you can change the title through the input parameter.

3. Secure a part of the web server and verify that this protection works as expected.

Deployment Logs

Although it has not yet been mentioned, perhaps you have noticed that for each deployment there is a “Reports” section on the run page.

At the end of the deployment phase, all of the logs from the SlipStream recipes are collected into a tarball and sent back to the server. From there you can download them to diagnose any problems.



Reports

- db.1_report_2015-04-22T125127Z.tgz
- db.2_report_2015-04-22T125117Z.tgz
- db.3_report_2015-04-22T125120Z.tgz
- lb.1_report_2015-04-22T125122Z.tgz
- orchestrator-exoscale-ch-gva_report_2015-04-22T125116Z.tgz
- web.1_report_2015-04-22T125124Z.tgz
- web.2_report_2015-04-22T125120Z.tgz

In the “Reporting” recipe, you can copy any additional files that you would like bundled with the reports into the location defined by the environmental variable `SLIPSTREAM_REPORT_DIR`.

Components Part II

Now that you have a component that runs a web server, create a second component that will run the tests against the server that you’ve been running by hand.

In this section you’ll exercise the knowledge from building the web server in building the test client. This shows how you can develop and test components individually.

Testing Goals

To test the web server fully, you should check that the conditions in the following table are true.

Web Page	credentials?	HTTP status
unprotected	none	200
protected	none	401
protected	wrong	401
protected	correct	200

Since the point of the exercise is learning about SlipStream and not shell programming, you can use the following (inelegant!) script to make the above tests. The script takes three parameters: host, username, and password:

```
#!/bin/bash

exit_code=0

web_hostname=$1
web_user=$2
web_password=$3
web_url_unprotected="http://${web_hostname}/"
web_url_protected="http://${web_hostname}/protected/"

# Check that web page responds with 200.
```

(continues on next page)

(continued from previous page)

```
rc=`curl -s -o /dev/null -w "%{http_code}" ${web_url_unprotected}`
echo "Return code from unprotected page is " ${rc}
if [ "${rc}" -ne "200" ]; then
  echo "Return code from unprotected page was not 200."
  exit_code=1
fi

# Check that web page responds with 401.
rc=`curl -s -o /dev/null -w "%{http_code}" ${web_url_protected}`
echo "Return code from protected page w/o password is " ${rc}
if [ "${rc}" -ne "401" ]; then
  echo "Return code from protected page was not 401."
  exit_code=1
fi

# Check that web page responds with 401.
rc=`curl -u ${web_user}:WRONG_PWD -s -o /dev/null -w "%{http_code}" ${web_url_
↪protected}`
echo "Return code from protected page w/ wrong password is " ${rc}
if [ "${rc}" -ne "401" ]; then
  echo "Return code from protected page was not 401."
  exit_code=1
fi

# Check that web page responds with 200.
rc=`curl -u ${web_user}:${web_password} -s -o /dev/null -w "%{http_code}" ${web_url_
↪protected}`
echo "Return code from protected page w/ password is " ${rc}
if [ "${rc}" -ne "200" ]; then
  echo "Return code from protected page was not 200."
  exit_code=1
fi

# set the customstate to inform user about the result
ss-set statecustom "Web client exit code: ${exit_code}"

exit ${exit_code}
```

Important: Be careful to escape variable references (or avoid them entirely) when writing the above script to the system.

Important: Also be careful to make the script you create executable with the command `chmod a+x script_name.sh`!

Web Client

Before developing your web client, you should ask yourself the following questions to plan your attack strategy:

- Where should the above checking script be installed?
- Does the script need any additional software installed?
- What input and/or output parameters need to be defined?

- How are the needed values obtained and passed to the checking script?
- What happens if the checking starts before the web server is ready?

EXERCISES

1. Create your own test client component.
2. Run the client pointing to a non-existant server, what happens?
3. Start your web server component.
4. Using interactive debugging, verify that the test client works.
5. Start a new client with the web server running, does the full test work correctly without any manual intervention?
6. Verify the correct behavior by downloading the reports.

Applications

Although some useful applications can be deployed as single machines, most applications nowadays are large, multi-layer beasts with several different functional blocks. The typical 3-tiered web application falls into this category with its load balancer, application front-ends, and database, usually all with redundancy and failover.

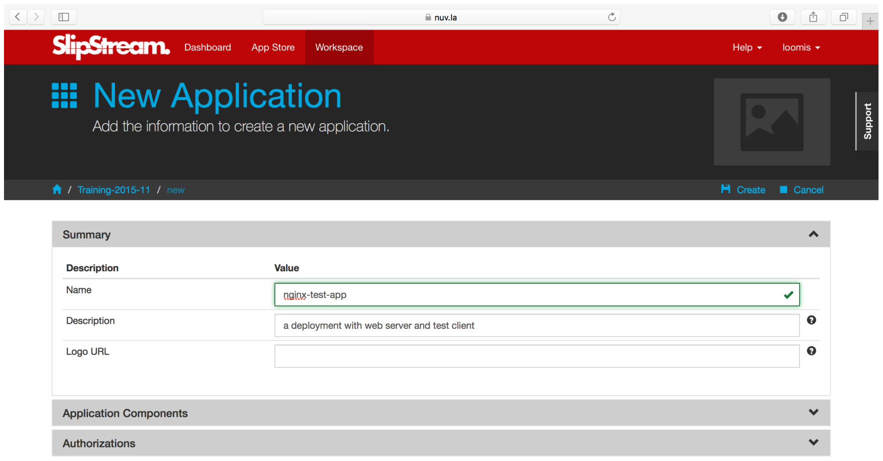
In this section, we will build our first multi-node application. You will learn how to:

- Combine your components into multi-machine deployment
- Connect parameters between application components
- Coordinate the configuration of the components
- Change the number of machines deployed
- Spread a deployment between clouds

Application Definition

An application is a set of nodes (that are tied to component definitions) that will be deployed together. The application will also define the mapping between the input/output parameters of the nodes that are tied together as well as the multiplicity of each node.

Create a new application definition in your home project. The first section of the new application page just asks for generic information.



New Application

Add the information to create a new application.

Training-2015-11 / new

Summary

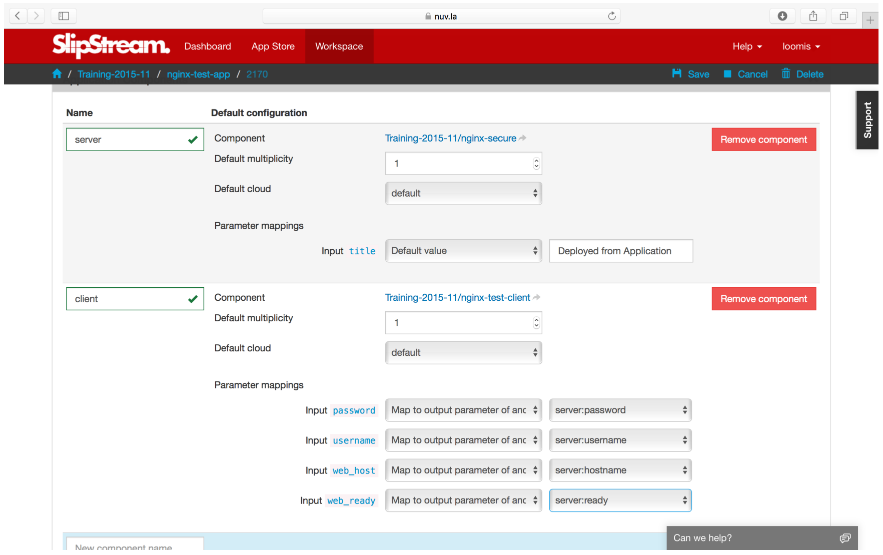
Description	Value
Name	nginx-test-app
Description	a deployment with web server and test client
Logo URL	

Application Components

Authorizations

The second section provides information about the nodes that will comprise the full application deployment. If there are input parameters for a node, then you can set the values here or provide a link to another node's output parameter.

Add two nodes for this deployment, the web server and the test client.



Training-2015-11 / nginx-test-app / 2170

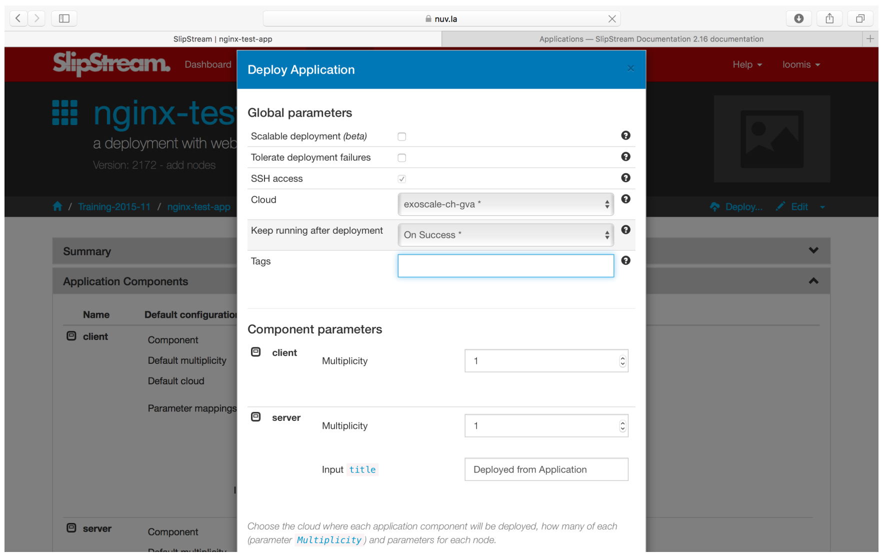
Name	Default configuration		
server	Component	Training-2015-11/nginx-secure	
	Default multiplicity	1	
	Default cloud	default	
	Parameter mappings		
	Input `title`	Default value	Deployed from Application
client	Component	Training-2015-11/nginx-test-client	
	Default multiplicity	1	
	Default cloud	default	
	Parameter mappings		
	Input `password`	Map to output parameter of anc	server:password
	Input `username`	Map to output parameter of anc	server:username
	Input `web_host`	Map to output parameter of anc	server:hostname
	Input `web_ready`	Map to output parameter of anc	server:ready

There are several important things to notice about the application definition. You can:

- Add any number of different components to the application.
- Define the default multiplicity for each node (default is 1).
- Define the cloud to use for the node (although you usually want to use "default").
- Tie output parameters from one node to the input parameters of another.

Deploying the Application

Now that the application has been defined, you can deploy the full application by clicking the "Deploy…" button. Doing so will bring up the usual run dialog.



Deploy Application

Global parameters

Scalable deployment *(beta)*	☐
Tolerate deployment failures	☐
SSH access	✓
Cloud	exoscale-ch-gva *
Keep running after deployment	On Success *
Tags	

Component parameters

client	Multiplicity	1
server	Multiplicity	1
	Input `title`	Deployed from Application

Choose the cloud where each application component will be deployed, how many of each (parameter `Multiplicity`) and parameters for each node.

Verify that all of the input parameter values are OK and then run the application. As usual you will be redirected to the dashboard. You can view the advancement of the application from there or from the run page.

When the application completes the deployment, review the reports to ensure that everything worked correctly.

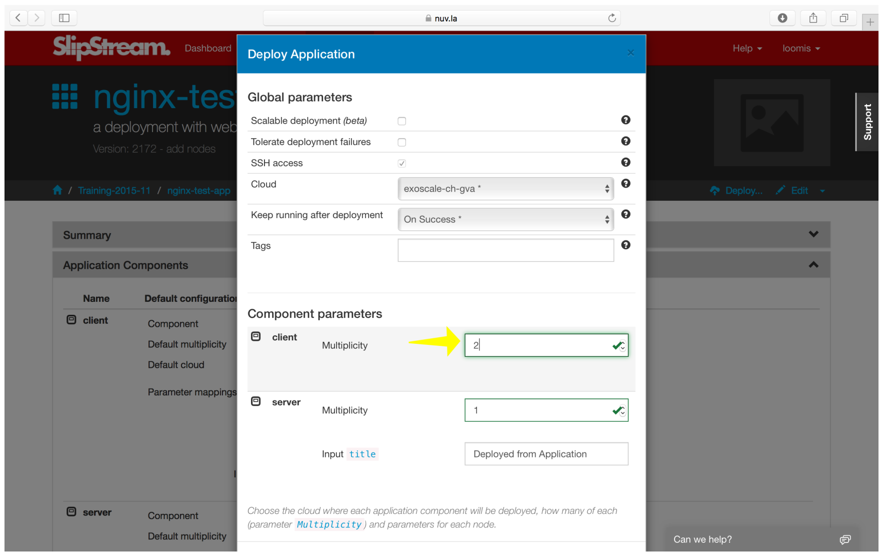
Advanced Deployments

From the run dialog you can make a number of significant changes to the application deployment without having to change the application definition itself.

Changing Multiplicity

You can change the multiplicity of the nodes in a particular deployment. The default is to have one web server and one test client. It doesn't make much sense for this application to have more than one server, but having more than one client could be interesting.

You can do this by changing the client multiplicity in the run dialog.



Deploy Application

Global parameters

Scalable deployment *(beta)*	☐
Tolerate deployment failures	☐
SSH access	✓
Cloud	exoscale-ch-gva *
Keep running after deployment	On Success *
Tags	

Component parameters

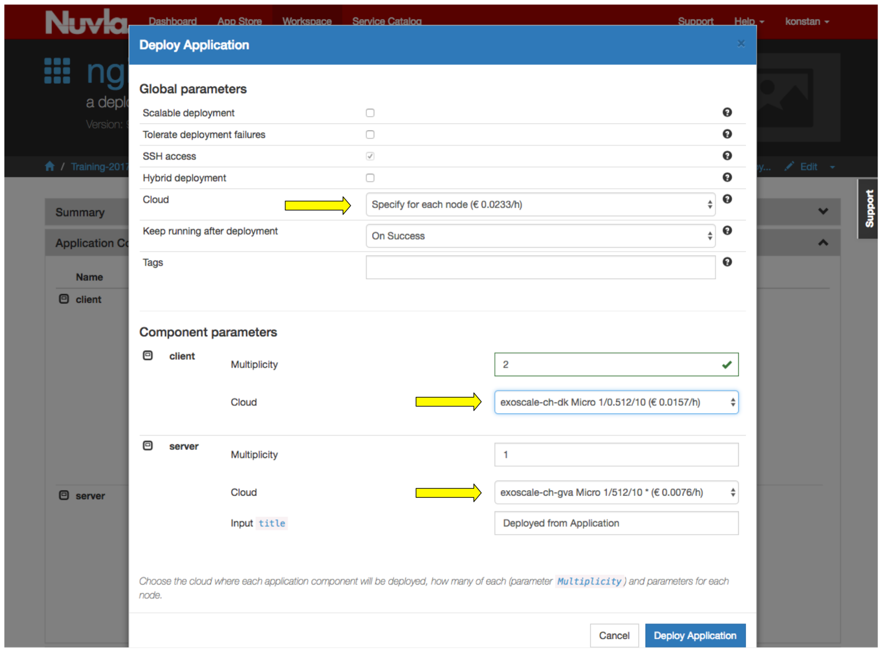
client	Multiplicity	2
server	Multiplicity	1
	Input `title`	Deployed from Application

Choose the cloud where each application component will be deployed, how many of each (parameter `Multiplicity`) and parameters for each node.

This will then deploy three machines in total. You can check that the reports for each of the clients shows successful results.

Multi-Cloud Deployments

You can also deploy the nodes of the application into different clouds or different cloud regions of the same cloud, creating a real multi-cloud deployment.



Deploy Application

Global parameters

Scalable deployment	☐
Tolerate deployment failures	☐
SSH access	✓
Hybrid deployment	☐
Cloud	Specify for each node (€ 0.0233/h)
Keep running after deployment	On Success
Tags	

Component parameters

client	Multiplicity	2
	Cloud	exoscale-ch-dk Micro 1/0.512/10 (€ 0.0157/h)
server	Multiplicity	1
	Cloud	exoscale-ch-gva Micro 1/512/10 * (€ 0.0076/h)
	Input `title`	Deployed from Application

Choose the cloud where each application component will be deployed, how many of each (parameter `Multiplicity`) and parameters for each node.

To do this, choose the option to set the cloud for each node separately. Then do so for each node type. In the above screenshot, I'm running the server on Exoscale in exoscale-ch-gva and exoscale-ch-dk regions.

Orchestrators

You may have noticed that during the deployment of an application, an additional machine is deployed per cloud. This machine is called the orchestrator and is created and deployed by SlipStream to handle the deployment of multi-machine applications.

The orchestrators remain active only when the application can go through a "provisioning" stage. For the types of deployments that we've done up to this point, the orchestrators will be terminated after all of the reports have been sent back to the SlipStream server.

Although very lightweight, the orchestrator does represent a small overhead when deploying applications through SlipStream.

EXERCISES

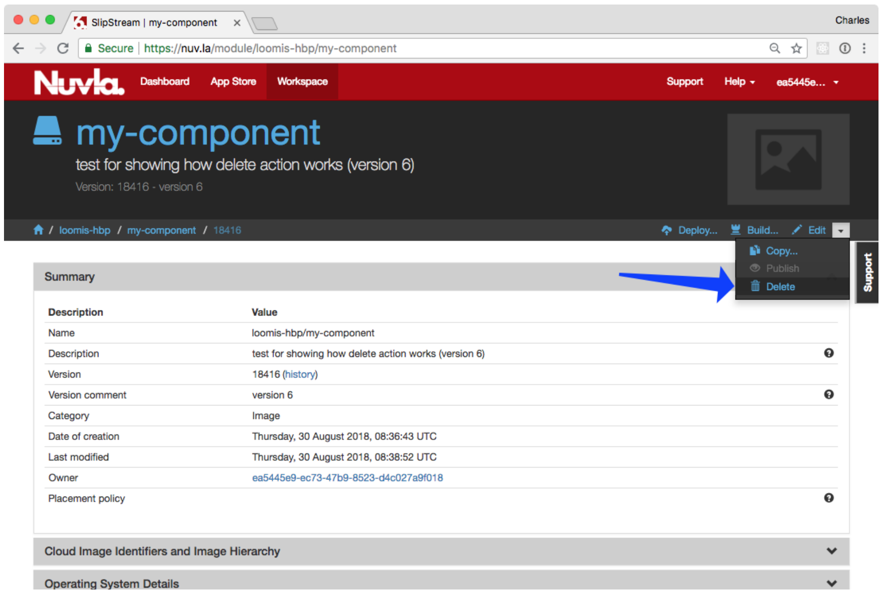
1. Deploy your application and verify from the reports that everything worked correctly.
2. Change the number of clients and then verify from the reports that all of the clients had the correct responses from the server.
3. Deploy the client node and server nodes in different clouds and verify that the deployment works.
4. Can you modify the application so that you can put clients in different clouds? Verify that your solution works.

Deleting Modules

SlipStream saves all versions of all modules by default. This allows you to rollback to a previous version or to view the evolution of the module. There are times, however, where you would like to remove a specific version of a module or all versions.

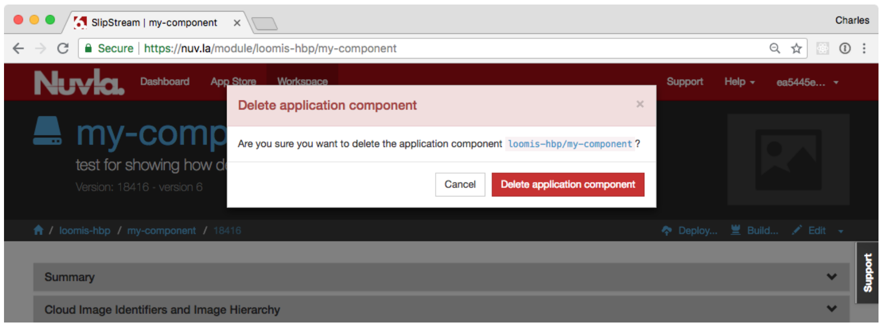
Delete Module Version

When you view the detail page for module, you will see the "Delete" action under the "Edit" dropdown near the top right of the page. This will be greyed out, if you don't have permission to delete the module.


SlipStream | my-component
Charles
Secure | https://nuv.la/module/loomis-hbp/my-component
Nuvla
Dashboard
App Store
Workspace
Support
Help
ea5445e...
my-component
test for showing how delete action works (version 6)
Version: 18416 - version 6
loomis-hbp / my-component / 18416
Deploy...
Build...
Edit
Copy...
Publish
Delete
Support
Summary
Cloud Image Identifiers and Image Hierarchy
Operating System Details

Description	Value
Name	loomis-hbp/my-component
Description	test for showing how delete action works (version 6)
Version	18416 (history)
Version comment	version 6
Category	Image
Date of creation	Thursday, 30 August 2018, 08:36:43 UTC
Last modified	Thursday, 30 August 2018, 08:38:52 UTC
Owner	ea5445e9-ec73-47b9-8523-d4c027a9f018
Placement policy	

When you click on "Delete" action, you will then see a dialog to confirm the action.


SlipStream | my-component
Charles
Secure | https://nuv.la/module/loomis-hbp/my-component
Delete application component
Cancel
Delete application component
Summary
Cloud Image Identifiers and Image Hierarchy
Support

Are you sure you want to delete the application component `loomis-hbp/my-component` ?

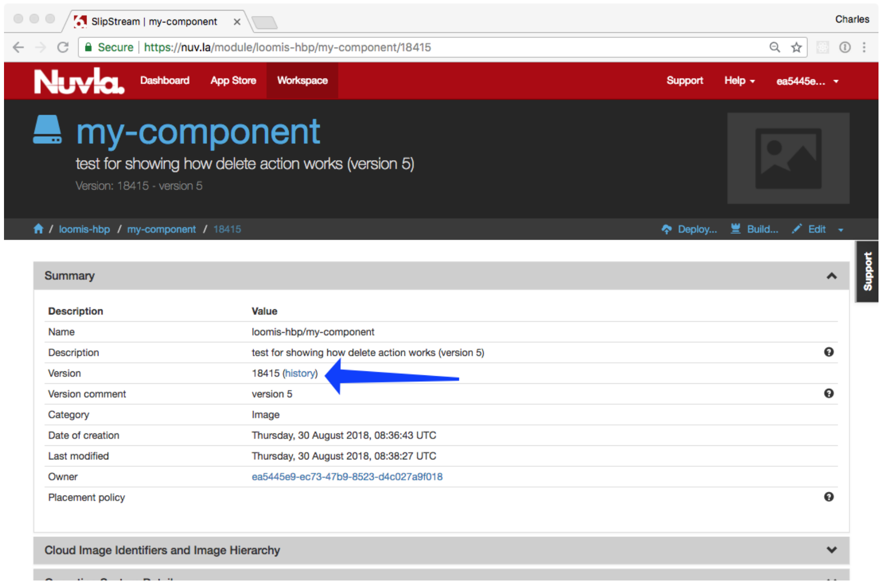
This will delete **only the version of the module that you are viewing.** After you have confirmed the action, you will be redirected to the latest, existing version of the module (or to the parent, if there are no remaining versions).

Note: Project modules are not versioned. The "Delete" action will delete the project entirely, provided that you have permission to do so and that the project contains no children.

Delete Entire Module

There are times when you want to delete all versions of a module. Although the versions can be deleted one at a time, this becomes tedious for a module with a long history.

To delete all versions of a module, click on the “history” link next to the “Version” field in the “Summary” section.


SlipStream | my-component
Charles
Secure | https://nuv.la/module/loomis-hbp/my-component/18415
Dashboard
App Store
Workspace
Support
Help
ea5445e...
loomis-hbp / my-component / 18415
Deploy...
Build...
Edit
Support

my-component

test for showing how delete action works (version 5)

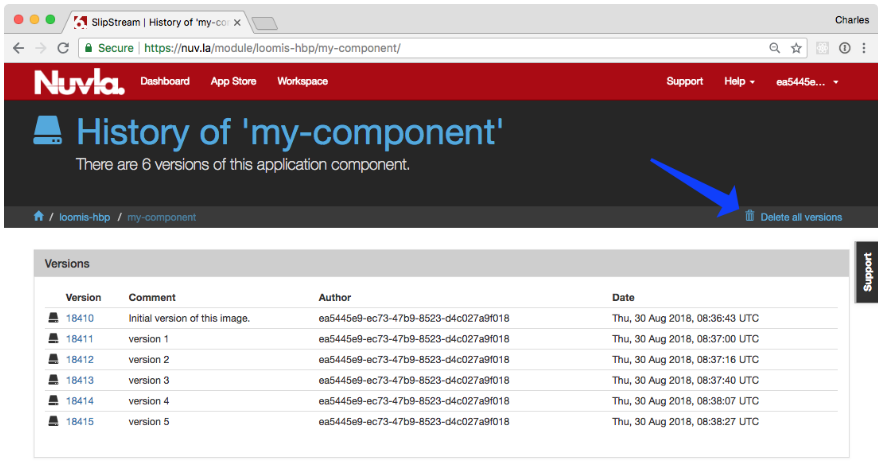
Version: 18415 - version 5

Summary

Description	Value
Name	loomis-hbp/my-component
Description	test for showing how delete action works (version 5)
Version	18415 (history)
Version comment	version 5
Category	Image
Date of creation	Thursday, 30 August 2018, 08:36:43 UTC
Last modified	Thursday, 30 August 2018, 08:38:27 UTC
Owner	ea5445e9-ec73-47b9-8523-d4c027a9f018
Placement policy	

Cloud Image Identifiers and Image Hierarchy

This will bring up the history that contains all visible versions of the module.


SlipStream | History of 'my-co
Charles
Secure | https://nuv.la/module/loomis-hbp/my-component/
Dashboard
App Store
Workspace
Support
Help
ea5445e...
loomis-hbp / my-component
Delete all versions
Support

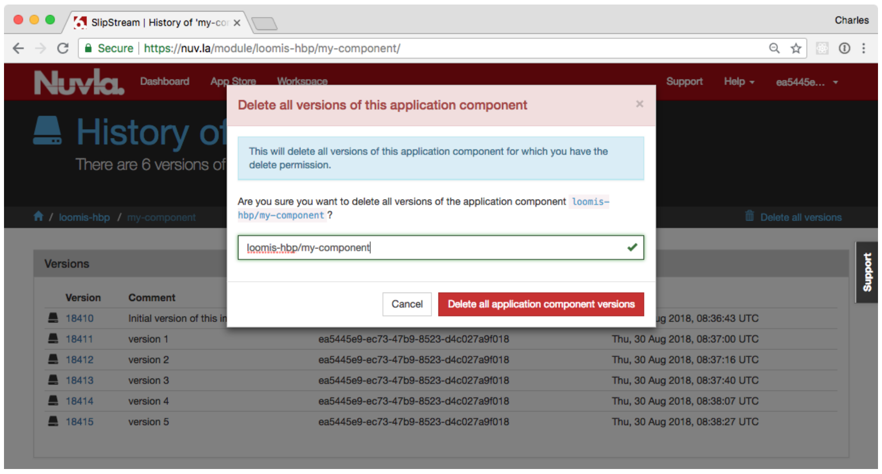
History of 'my-component'

There are 6 versions of this application component.

Versions

Version	Comment	Author	Date
18410	Initial version of this image.	ea5445e9-ec73-47b9-8523-d4c027a9f018	Thu, 30 Aug 2018, 08:36:43 UTC
18411	version 1	ea5445e9-ec73-47b9-8523-d4c027a9f018	Thu, 30 Aug 2018, 08:37:00 UTC
18412	version 2	ea5445e9-ec73-47b9-8523-d4c027a9f018	Thu, 30 Aug 2018, 08:37:16 UTC
18413	version 3	ea5445e9-ec73-47b9-8523-d4c027a9f018	Thu, 30 Aug 2018, 08:37:40 UTC
18414	version 4	ea5445e9-ec73-47b9-8523-d4c027a9f018	Thu, 30 Aug 2018, 08:38:07 UTC
18415	version 5	ea5445e9-ec73-47b9-8523-d4c027a9f018	Thu, 30 Aug 2018, 08:38:27 UTC

This page contains the “Delete all versions” action. Click on that and you will then see a confirmation dialog.


SlipStream | History of 'my-co
Charles
Secure https://nuv.la/module/loomis-hbp/my-component/
Nuvla
Dashboard
App Store
Workspace
Support
Help
ea5445e...
Support

History of

There are 6 versions of

loomis-hbp / my-component

Delete all versions

Delete all versions of this application component

This will delete all versions of this application component for which you have the delete permission.

Are you sure you want to delete all versions of the application component `loomis-hbp/my-component` ?

loomis-hbp/my-component

Cancel | Delete all application component versions

Versions

Version	Comment		
18410	Initial version of this in		ug 2018, 08:36:43 UTC
18411	version 1	ea5445e9-ec73-47b9-8523-d4c027a9f018	Thu, 30 Aug 2018, 08:37:00 UTC
18412	version 2	ea5445e9-ec73-47b9-8523-d4c027a9f018	Thu, 30 Aug 2018, 08:37:16 UTC
18413	version 3	ea5445e9-ec73-47b9-8523-d4c027a9f018	Thu, 30 Aug 2018, 08:37:40 UTC
18414	version 4	ea5445e9-ec73-47b9-8523-d4c027a9f018	Thu, 30 Aug 2018, 08:38:07 UTC
18415	version 5	ea5445e9-ec73-47b9-8523-d4c027a9f018	Thu, 30 Aug 2018, 08:38:27 UTC

You must provide the name of the module (as a protection) to enable the confirmation button. When you click on the confirmation, all versions of the module will be deleted.

EXERCISES

1. Create a component with a number of different versions.
2. Navigate to a specific version, delete it, and verify that it disappears from the module history.
3. Use the "Delete all version" action from the history page and verify that the module disappears.

After finishing this module, building, deploying and managing cloud applications should no longer be a problem for you, especially with such a powerful tool as SlipStream in your DevOps tool belt.

You are now prepared for the remaining advanced modules. Choose those that interest you. See *Module IV* for (auto-)scaling your cloud applications or *Module V* for interacting with SlipStream via its API and optimizing your deployments.

3.1.4 Module IV - Scalable Applications (Advanced)

In this module you will learn how to: deploy an application that can be scaled horizontally and vertically; scale an application through the API; respond to resource changes to update configurations; and automate the scaling process.

The module requires a good knowledge of the core application creation and management features of SlipStream. If necessary, review the material in *Module III*.

Scalable Applications

Very few, if any, applications experience a constant load. Most experience large fluctuations that require adjustments to maintain responsiveness at peak demand and to avoid wasting resources when demand declines.

In this section you will learn how to:

- Deploy an application that can be scaled horizontally and vertically
- Scale the application through the API
- Respond to resource changes to update configurations

Types of Scaling

By default, the resource allocation of a deployment is fixed when the deployment is started. However clicking on the "Scalable deployment" option in run dialog will allow the deployment to be vertically and horizontally scaled through the SlipStream API. The following scaling actions are possible:

- **Horizontal Scaling**
  - add VM (scale up)
  - remove VM (scale down)
- **Vertical Scaling**
  - change VM in size (CPU/RAM or instance type)
  - attach/detach extra disk to/from VM

When a run of a deployment is declared as scalable, the orchestrator VM will not be terminated when the initial deployment finishes. This necessary overhead allows the SlipStream to respond to requests to scale the deployment.

Triggering Scaling Actions

For a running application, the scaling actions can be triggered through the SlipStream REST API or through the command line client. **Scaling actions are not yet supported in the web interface.**

The API calls to trigger the scaling actions are available under the Run section of the SlipStream API documentation.

The SlipStream CLI comes with the following commands that trigger the scaling actions on deployments and VMs:

```
ss-node-add [options] <run> <node-name> [<number>]
ss-node-remove [options] <run> <node-name> <ids> [<ids> ...]
ss-scale-resize [options] [--cpu <num>, --ram <num>]|[--instance-type <type>] <run>
↪<node-name> <ids> [<ids> ...]
ss-scale-disk [options] [--attach <GB> | --detach <device>] <run> <node-name> <ids> [
↪<ids> ...]
```

Where `node-name` stands for application component name.

Scalability Workflow Hooks (Scripts)

Hooks that define application-specific scaling actions are available for running before, after and during the respective scalability actions. These scripts are required so that the application components can be correctly reconfigured when scaling actions happen.

The available hooks are defined in the following table.

Script	Action	When and Where Executed
"On VM Add"	*horizontal scale up*	after addition of new VMs on all the VMs of the deployment except the ones that were just added.
"On VM Remove"	*horizontal scale down*	after the removal of the requested VMs on all the VMs left in the deployment.
"Pre-Scale"	*horizontal scale down*	before VMs removal action, on the VMs targeted for the removal, and therefore, before the "On VM Remove" script.
"Pre-Scale"	*vertical scale up/down*	before any vertical scaling action (VM resizing or attaching/detaching of extra disk) on the VMs that are subject to the scaling action.
"Post-Scale"	*vertical scale up/down*	after any vertical scaling action (VM resizing or attaching/detaching of extra disk) on the VMs that are subject to the scaling action.

Detailed information about how to write those scripts is available from the SixSq GitHub repository.

Horizontal Scaling - Add or Remove VMs

When adding a component instance (VM), you must specify the component type of the machine that you want to add. The server (and then orchestrator) will mutate the deployment, provisioning the new node instance and then notifying all of the machines in the application.

The notification takes place by running the "On VM Add" script (if it exists) on all VMs, except the ones that were just added. On the newly added VMs only the deployment target script is executed.

As an example, look at the . Because Elasticsearch has its own discovery service and was designed for horizontal scaling it doesn't need elaborate hook scripts in the SlipStream configuration.

In the “Deployment” script, we can allow Elasticsearch to discover other members of the cluster by listing its peers in the configuration. The code to do this is the following:

```
# application or application component?
run_type=`ss-get ss:category`

if [ "${run_type}" = "Image" ]; then
  hosts=`ss-get hostname`
else
  # discover all of the workers
  nodename=`ss-get nodename`
  workers=$(echo `ss-get ${nodename}:ids` | tr ',' "\n")

  # create list of their hostnames for discovery
  hosts=""
  for w in $workers; do
    h=`ss-get ${nodename}.${w}:hostname`
    hosts="${hosts} \"${h}\""
  done
fi

hosts=$(echo ${hosts} | tr ' ' ',')

# rewrite the elasticsearch configuration file
# this will allow all hosts to discover each other
cat > /etc/elasticsearch/elasticsearch.yml <<EOF
network.host: 0.0.0.0
discovery.zen.ping.unicast.hosts: [${hosts}]
EOF
```

The interesting part for this tutorial is how the list of Elasticsearch workers is built up.

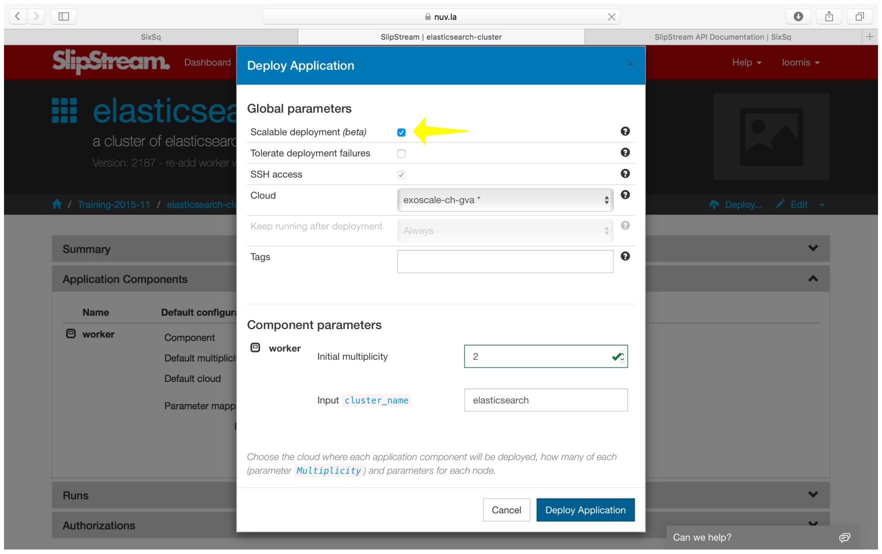
We initially detect whether the deployment was an application (“Deployment”) or an application component (“Image”) because the parameters are slightly different in the two cases. By doing this test, the resulting component can be run as a standalone image or as part of an application.

For an application, where more than one worker is possible, we use the `nodename` and `ids` to iterate over all of the workers and collect their hostnames. After all of the values have been substituted, the variable names for the hosts of the workers look like “worker.1:hostname”, “worker.2:hostname”, etc.

Note that we only need to do this in the deployment script because Elasticsearch keeps track of the cluster state and transmits this information to all of the nodes in the cluster itself. If it didn’t do this, we would need to tell each node of the changes through the “on VM add” script.

After the configuration of the service, we restart Elasticsearch to take into account the configuration changes. Look in the application and component definition for details.

To see how the scaling works, deploy the elasticsearch-cluster application.



Global parameters

Scalable deployment *(beta)*	✓
Tolerate deployment failures	☐
SSH access	✓
Cloud	exoscale-ch-gva *
Keep running after deployment	Always
Tags	

Component parameters

worker	Initial multiplicity	2
	Input `cluster_name`	elasticsearch

Choose the cloud where each application component will be deployed, how many of each (parameter `Multiplicity`) and parameters for each node.

To be able to scale the application later, **it is very important to tick the checkbox indicating that this is a scalable deployment!** By default, this will deploy a cluster with two nodes.

When the deployment is complete, it will provide a URL that gives the health of the cluster. The important thing to look at is the number of nodes in the cluster. It should initially be 2. This is the result:

```
{
  "cluster_name" : "elasticsearch",
  "status" : "green",
  "timed_out" : false,
  "number_of_nodes" : 2,
  "number_of_data_nodes" : 2,
  "active_primary_shards" : 0,
  "active_shards" : 0,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 0,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 100.0
}
```

The deployment worked correctly: the status is green and there are 2 nodes.

Scale Up with CLI

To scale the run via the command line, use the *ss-node-add* command. It takes the run ID, the type of node to scale ("worker" in our case) and the number of nodes to add:

```
$ ss-node-add ced28f99-e08b-4667-86db-73f53c059c58 worker 1
```

This will drive the application through another provisioning phase for the new worker. When the provisioning and configuration is complete, the application will return to the "Ready" state.

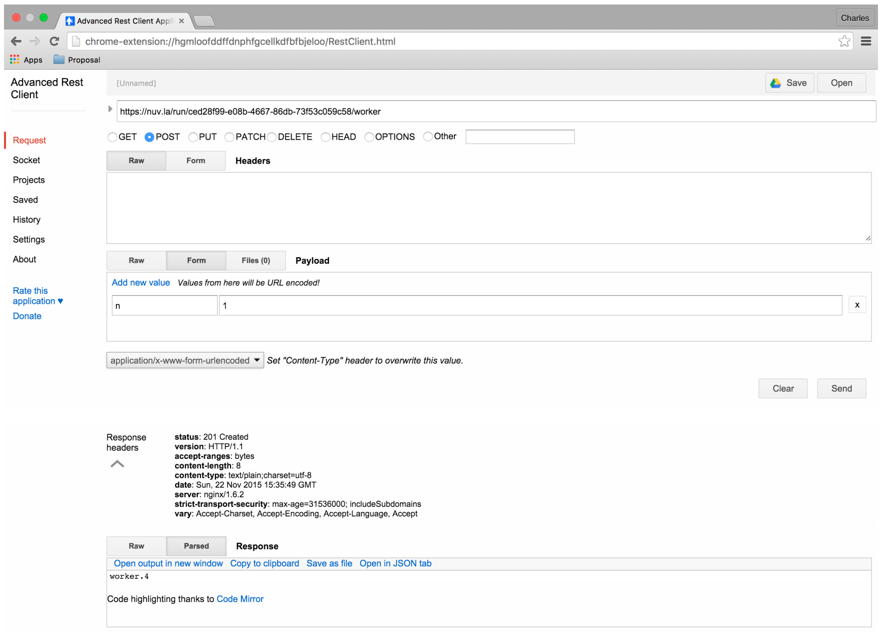
Note: Only one scaling action, on one type of component, can be active. Previous scaling actions must complete before a new one can be started.

Scale Up with REST

To do the same thing from the REST API, send a POST request to the URL:

```
https://nuv.la/run/ced28f99-e08b-4667-86db-73f53c059c58/worker
```

the body of the request should be a form with the parameter "n" and the number of nodes to add.



https://nuv.la/run/ced28f99-e08b-4667-86db-73f53c059c58/worker

GET [✓ POST] PUT PATCH DELETE HEAD OPTIONS Other

Values from here will be URL encoded!

n	1

application/x-www-form-urlencoded *Set "Content-Type" header to overwrite this value.*

Response headers

```
status: 201 Created
version: HTTP/1.1
accept-ranges: bytes
content-length: 8
content-type: text/plain;charset=utf-8
date: Sun, 22 Nov 2015 15:35:49 GMT
server: nginx/1.6.2
strict-transport-security: max-age=31536000; includeSubdomains
vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
```

```
worker.4
```

Note that the response gives the created node(s).

Just to verify that both of the add requests worked, look again at the health output from the service URL:

```
{
  "cluster_name" : "elasticsearch",
  "status" : "green",
  "timed_out" : false,
  "number_of_nodes" : 4,
  "number_of_data_nodes" : 4,
  "active_primary_shards" : 0,
  "active_shards" : 0,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 0,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 100.0
}
```

A heathy green with 4 nodes! Perfect.

Scale Down with CLI

We can also remove nodes in nearly the same way. The only difference is that you must specify exactly which node(s) you want to remove. From the command line, do the following:

```
$ ss-node-remove ced28f99-e08b-4667-86db-73f53c059c58 worker 1
```

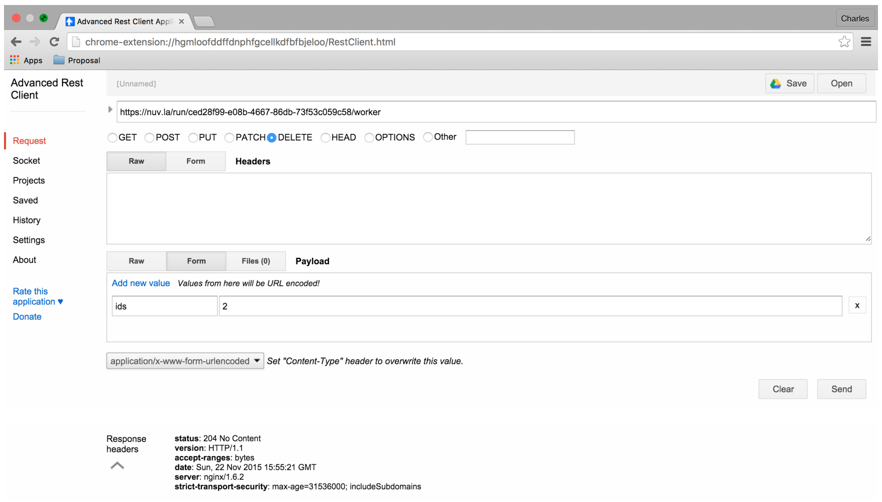
Again, after the (un-)provisioning cycle, the removed node instances will disappear from the deployment.

Scale Down with REST

Doing the same with the REST API, requires sending a DELETE request to the URL:

```
https://nuv.la/run/ced28f99-e08b-4667-86db-73f53c059c58/worker
```

with a form body containing the "ids" parameter. The values of "ids" must be a comma-separated list of machines to remove.



https://nuv.la/run/ced28f99-e08b-4667-86db-73f53c059c58/worker

GET POST PUT PATCH [✓ DELETE] HEAD OPTIONS Other

Values from here will be URL encoded!

ids	2

application/x-www-form-urlencoded Set "Content-Type" header to overwrite this value.

Response headers

status: 204 No Content
version: HTTP/1.1
accept-ranges: bytes
date: Sun, 22 Nov 2015 15:55:21 GMT
server: nginx/1.6.2
strict-transport-security: max-age=31536000; includeSubdomains

After these actions, check the health and make sure it is green with 2 nodes:

```
{
  "cluster_name" : "elasticsearch",
  "status" : "green",
  "timed_out" : false,
  "number_of_nodes" : 2,
  "number_of_data_nodes" : 2,
  "active_primary_shards" : 0,
  "active_shards" : 0,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 0,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 100.0
}
```

Everything looks to have worked correctly!

Scale Down Scripts

Before the removal of the node instances, the “Pre-scale” script gets run on them. This allows to execute any application related pre-removal actions on the targeted node instance.

Similarly, the “On VM Remove” script will be run on each node instance after the given node instance(s) have been removed.

None of these scripts is necessary for the Elasticsearch cluster itself. However, we want to maintain a working service URL for the deployment as a whole. If the node referenced in the service URL disappears, we want to update the service URL to a working node:

```
#!/bin/bash -xe

# application or application component?
run_type=`ss-get ss:category`

if [ "${run_type}" = "Image" ]; then
  # should never be called from a component deployment anyway
  exit 0
else

  # only run if my own node type is being scaled
  nodename=`ss-get nodename`
  if [ "${SLIPSTREAM_SCALING_NODE}" = "${nodename}" ]; then

    # collect all of the remaining workers
    workers=$(echo `ss-get ${nodename}:ids` | tr ',' "\n")

    echo "WORKERS: ${workers}"
    echo "REMOVED: ${SLIPSTREAM_SCALING_VMS}"

    # update URL with first remaining host
    for w in $workers; do
      hostname=`ss-get ${nodename}.${w}:hostname`
      link="http://${hostname}:9200/"
      health="${link}_cluster/health?pretty=true"
      ss-set ss:url.service ${health}
      exit 0
    done
  fi
fi
```

This script just resets the URL using the hostname associated with the first one in the cluster.

Vertical Scaling

Important: Vertical scalability is not available for all clouds. It requires the availability of the feature in the underlying cloud as well as in the SlipStream cloud connector.

Change the VM Size

An application manager may discover that a running application would be more efficient if certain virtual machines were allocated additional CPU, RAM, or disk space. To request the resizing of the VM from the command line, run the following command, providing the desired new size of the VM:

```
$ ss-scale-resize --cpu 8 --ram 16 \
    f9390d34-10b1-4621-bd05-f4d8c7557754 db 1 3
```

The size specification depends on the cloud being used. Only CPU or RAM can be specified.

The same way one can scale down the size of the VM(s) by simply defining the required size of the VM(s).

Note: Virtual machines that are vertically scaled, will go through a reboot cycle to force the new resource values to be taken into account.

Attach and Detach Disks

To add an extra disk, the following command should be used:

```
$ ss-scale-disk --attach 75 \
    f9390d34-10b1-4621-bd05-f4d8c7557754 db 1 3
```

The extra disk is attached as a block device and on Linux systems should appear as block device under the `/dev` folder usually as `/dev/sd*` or `/dev/vd*`. The block device name depends on the virtualization driver used and this should be checked with the cloud provider.

Detaching an extra disk requires either the block device name (e.g., `/dev/vdc`) or its cloud ID (usually in the form of a UUID). Here is the example of detaching of the extra disk by the block device name:

```
$ ss-scale-disk --detach /dev/vbc \
    f9390d34-10b1-4621-bd05-f4d8c7557754 db 1 3
```

In all the above cases the **"Pre-Scale"** and **"Post-Scale"** scripts will be run respectively right before and after the IaaS scaling action is requested from the Cloud by the orchestrator. On most of the clouds the resizing of VMs requires their reboot. The scripts allow the user to do the needful to prepare for the scaling action and later to account for the changes made to the VM.

The examples of the **"Pre-Scale"** and **"Post-Scale"** can be found here.

EXERCISES

1. Deploy an Elasticsearch cluster.
2. Add nodes through the command line or REST API.
3. Remove nodes through the command line or REST API.

Autoscalable Applications

Scalable Applications described how it is possible to define and manually manage the scaling of applications with SlipStream. This section presents the mechanisms and the accompanying infrastructure in SlipStream that allows to auto-scale such applications.

Auto-scaling of User Applications

With SlipStream one can automatically scale up and down the number of instances of an application component of a (possibly) complex application. Currently only one component can be scaled at a time. Work to remove this limitation is ongoing. There are two ways to take advantage of auto-scaling with the existing SlipStream implementation:

- **Black-box autoscaling** The simplest approach is to use the default implementation of the auto-scaling in SlipStream. Currently, it is suitable for the applications where the only one component will scale based on only one metric. If these requirements are met, adding a special autoscaler component to the user's application, providing configuration file with the application component scalability constraints, and deploying the application as a scalable deployment allows user to benefit from the automated scalability provided "out of the box" by SlipStream.
- **Do-it-yourself (DIY) autoscaling** The implementation of the autoscaler in SlipStream allows users to supply their own implementations of the autoscaling logic. In this case, SlipStream takes care of the deployment of the autoscaler platform and running the user supplied auto-scaling logic. What is required from the user is the inclusion of the autoscaler in the user's application as a component and providing a public URL with the autoscaling implementation.

In the above cases, it is required that chosen components of the user's application publish metrics on which the scaling actions will be based and to support the automatic scaling.

SlipStream uses Riemann to implement its auto-scaling decision making feature and consequently requires that the metrics be published as Riemann events. To facilitate metrics collection, Riemann has a wide range of metric publishers; these include a Riemann plugin for Collectd and a Python CLI and library API.

The implementation of the autoscaler in SlipStream is completely open and flexible. It allows users to write their own decision making logic as Riemann streams and to provide it to the autoscaler component as an input parameter. This can be useful in cases when the default autoscaling implementation, that comes with the SlipStream autoscaler, does not fully satisfy the needs of the user's application.

Configuration of Auto-scale Constraints

Below is the example configuration file (in edn format) to be used with the black-box autoscaling approach. The configuration defines scalability constraints for an application component called `webapp`:

```
[
 {
  ;;
  ;; Mandatory parameters.

  ; name of the component in the application
  :comp-name         "webapp"

  ; service tags as sent by Riemann publisher in the event
  :service-tags      ["webapp"]

  ; monitored service metric name (as sent with Riemann event)
  :service-metric    "avg_response_time"

  ; value of the metric after which start adding component instances
  :metric-thold-up   500.0
  ; value of the metric after which start removing component instances
  :metric-thold-down 200.0

  ; max number of component instances allowed
```

(continues on next page)

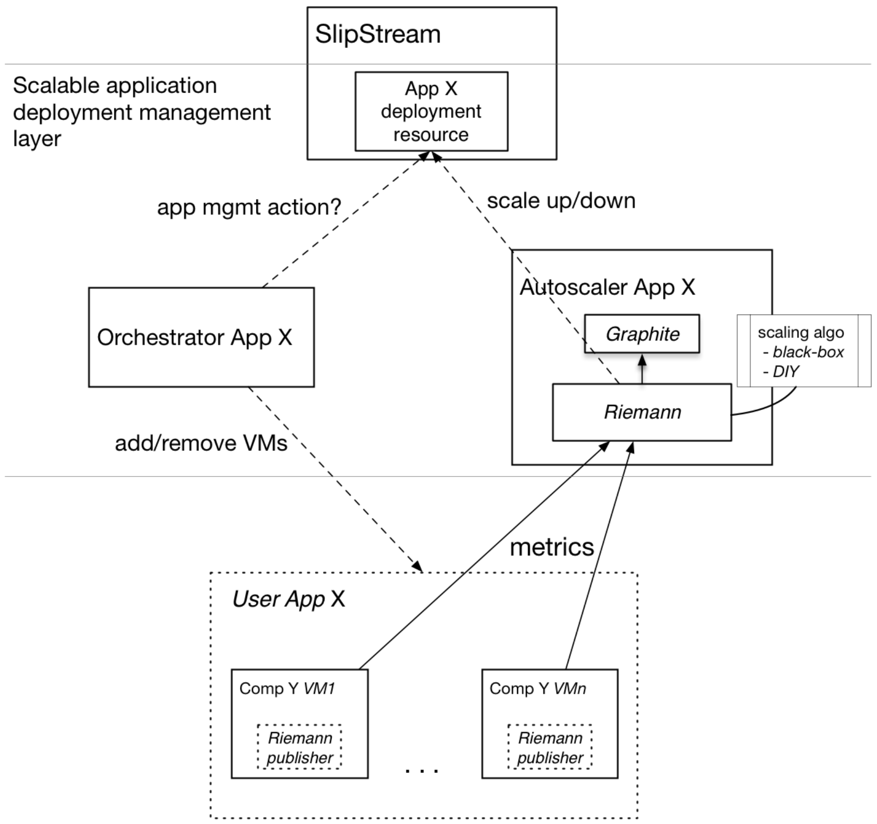


Fig. 1: Auto-scaling applications with SlipStream

(continued from previous page)

```
  :vms-max           4   ; "Price" constraint.

  ;;
  ;; Optional parameters (with defaults).

  ; min number of component instances allowed
  ;:vms-min       1

  ; number of instances by which to scale up
  ;:scale-up-by   1
  ; number of instances by which to scale down
  ;:scale-down-by 1
 }
]
```

It reads the following way:

1. For an application component `webapp` the autoscaler expects to receive Riemann events:
   1. with the service name `avg_response_time`,
   2. one of tags being `webapp`;
2. When the value of the metric in the event is
   1. above `metric-thold-up` (500.0), then the autoscaler should perform a scale up action,
   2. below `metric-thold-down` (200.0), then the autoscaler should perform a scale down action;
3. The autoscaler should not perform scale up action if there are already `vms-max` (4) component instances running,
4. The minimum number of component instances should not go below `vms-min`. That is, the autoscaler should not attempt a scale down action if the number of the currently running component instances is `vms-min` (1),
5. The autoscaler should scale up by `scale-up-by` number of instances and scale down by `scale-down-by`.

Autoscaler Component

The autoscaler component is available on Nuvla at . Its source code is published on GitHub.

The application uses Riemann to process incoming component metrics as events. The main part of the application is the Riemann configuration file, `app/riemann-ss-streams.clj`, the script is written in Clojure programming language.

The default implementation loads the auxiliary library (as the `sixsq.slipstream.riemann.scale` Clojure namespace) that defines custom event processing streams and helper functions. It then uses those functions to read in the scalability constraints for the component(s), to process/update incoming events (or generate new events), to make the scaling decisions, and to request the scaling actions from SlipStream.

The decision-making algorithm uses Riemann's moving time window stream with a window size of 30s to smooth out spikes in the incoming metrics' time series.

The functions defined in the `sixsq.slipstream.riemann.scale` namespace simplify the main configuration script by providing a number of utility functions that hide the details of interacting with SlipStream to request scaling actions, creating new and/or updating old events, and (re-)indexing and/or publishing them to Graphite. The namespace is defined in a the `run-proxy/api` module in the GitHub repository. All the public functions are documented.

Based on the example in `app/riemann-ss-streams.clj` and taking advantage of the utility functions, application developers can write their own scaling logic and then provide it to the autoscaler component as a public URL via an input parameter. For details, see the component on Nuvla.

The deployment script of the autoscaler module deploys Riemann, the Riemann dashboard, and Graphite. After deployment, the Riemann dashboard can be found at the URL `http://<autoscaler IP>:6006` and Graphite, at `http://<autoscaler IP>`.

Riemann acts as a stream processing engine with no or little memory of the events. Graphite is used to store the historical data of the events. Primarily this is used to plot the historical data, but could also be read into the Riemann streams to consider the history when making scalability decisions.

Publishing Component Metrics to Autoscaler

The autoscaler makes the scalability decisions based on the metrics coming from the user's application components and the thresholds provided by the user. Because the autoscaler is a Riemann application, the user must use one of the many Riemann clients to push the metrics (as events) to it. Check Riemann clients for the list of available clients.

Example Auto-scale Application

The following section describes the example auto-scale application that uses write Riemann plugin of collectd and a custom publisher written in Python using Riemann client library API.

The example auto-scale application is a web application that uses the Riemann `collectd` plugin and a custom publisher written in Python and the Riemann Client API. It consists of the following components:

- **webapp:** a stateless web application that takes requests, synchronously performs a moderately intensive computation (calculating Pi up to 100 digits), and returns the result,
- **nginx:** a load balancer based on Nginx that distributes client requests to the set of stateless web servers,
- **client:** a test client based on Locust that simulates a varying number of clients, and
- **autoscaler:** Standard SlipStream autoscaler component that makes scaling decisions.

and it is schematically depicted in the figure below.

Application Configuration and Deployment

The figure below shows the definition of the autoscalable application in SlipStream.

Note that values for the autoscaler input parameters, namely, `riemann_config_url` and `scale_constraints_url` must be provided. The first one defines the URL with a resource under which the following files are expected:

- **riemann-ss-streams.clj:** Riemann streams using SlipStream scale up/down actions,
- **dashboard.json:** Riemann dashboard layout and queries, and
- **dashboard.rb:** Riemann dashboard configuration.

They contain the processing logic for autoscaling actions and the configuration for the Riemann dashboard. This allows users to provide their own implementation of the scaling logic and dashboard configuration. The second input parameter `scale_constraints_url` provides the URL with the application scaling definitions. For this application, they look like:

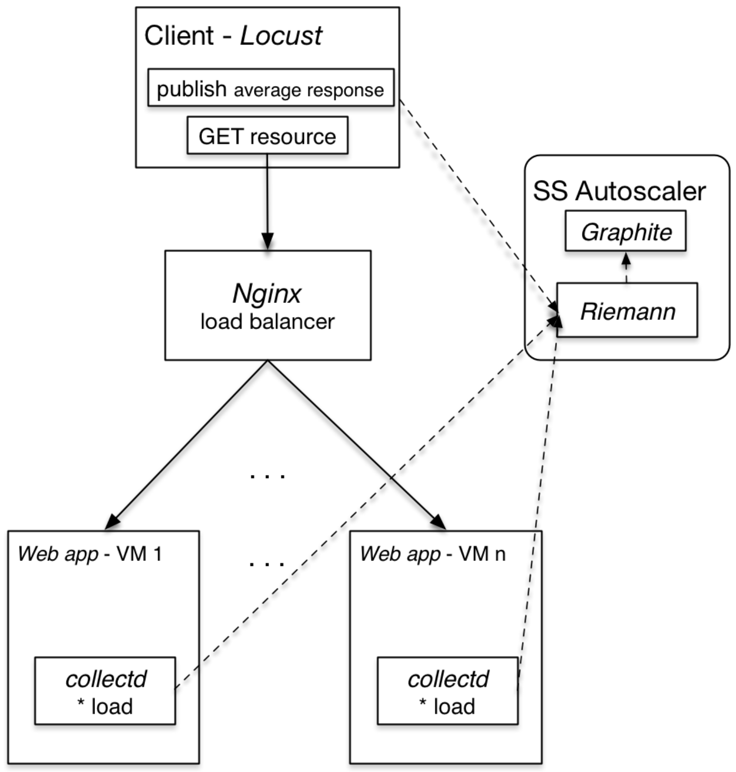
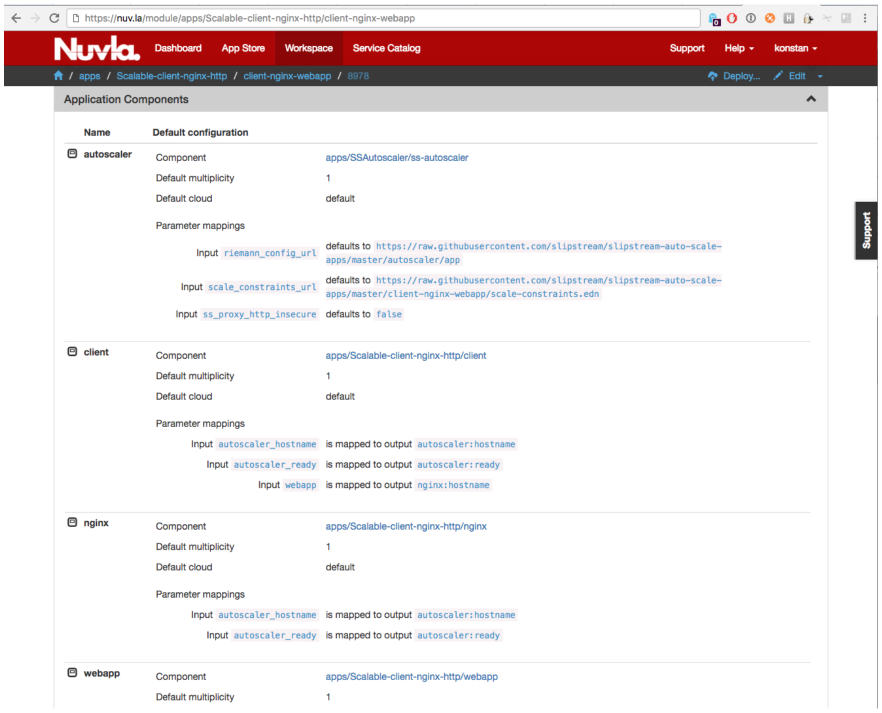


Fig. 2: Components of Example Auto-scale Application


https://nuv.la/module/apps/Scalable-client-nginx-http/client-nginx-webapp
Nuvla.
Dashboard
App Store
Workspace
Service Catalog
Support
Help
konstan
apps / Scalable-client-nginx-http / client-nginx-webapp / 8978
Deploy...
Edit
Support

Application Components

Name	Default configuration	
autoscaler	Component	apps/SSAutoscaler/ss-autoscaler
	Default multiplicity	1
	Default cloud	default
	Parameter mappings	
	Input `riemann_config_url`	defaults to `https://raw.githubusercontent.com/slipstream/slipstream-auto-scale-apps/master/autoscaler/app`
	Input `scale_constraints_url`	defaults to `https://raw.githubusercontent.com/slipstream/slipstream-auto-scale-apps/master/client-nginx-webapp/scale-constraints.edn`
	Input `ss_proxy_http_insecure`	defaults to `false`
client	Component	apps/Scalable-client-nginx-http/client
	Default multiplicity	1
	Default cloud	default
	Parameter mappings	
	Input `autoscaler_hostname`	is mapped to output `autoscaler:hostname`
	Input `autoscaler_ready`	is mapped to output `autoscaler:ready`
	Input `webapp`	is mapped to output `nginx:hostname`
nginx	Component	apps/Scalable-client-nginx-http/nginx
	Default multiplicity	1
	Default cloud	default
	Parameter mappings	
	Input `autoscaler_hostname`	is mapped to output `autoscaler:hostname`
	Input `autoscaler_ready`	is mapped to output `autoscaler:ready`
webapp	Component	apps/Scalable-client-nginx-http/webapp
	Default multiplicity	1

Fig. 3: Example Scalable Application Definition in SlipStream

```
$ cat scale-constraints.edn

[
 {:comp-name          "webapp"
  :service-tags       ["webapp"]
  :service-metric     "avg_response_time"
  :metric-thold-up    7000.0
  :metric-thold-down  4000.0
  :vms-max            4}
]
```

This file is application-specific; for this example, it comes from the user application module on GitHub.

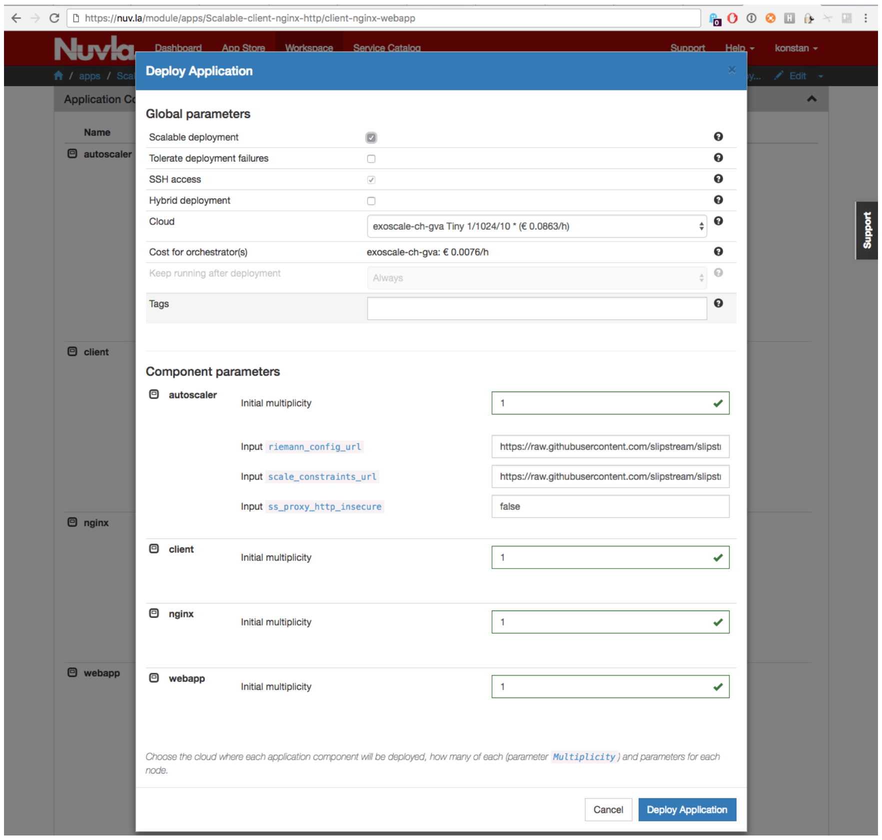
Clicking on Deploy button brings the Application Deployment dialog (see figure below). In this dialog, you must check the box to indicate that this is a scalable application; you can optionally change the multiplicity of the webapp component. Select the cloud and proceed with the deployment.

Usage of the Application After Deployment

After a successful deployment of the application one should first open the HTTP URL published by the client component (see ellow arrow on the figure below).

It provides a page with a description of the application, a deployment diagram, and links to the services running on other components (see figure below).

The first link to follow is the one pointing to the Locust load generator; this simulates a varying number of clients. Although Locust can be used as an automatic load generator, this application assumes manual configuration of the load parameters. Figure below shows the definition of three parallel users that Locust will use to access the web layer



https://nuv.la/module/apps/Scalable-client-nginx-http/client-nginx-webapp

Deploy Application

Global parameters

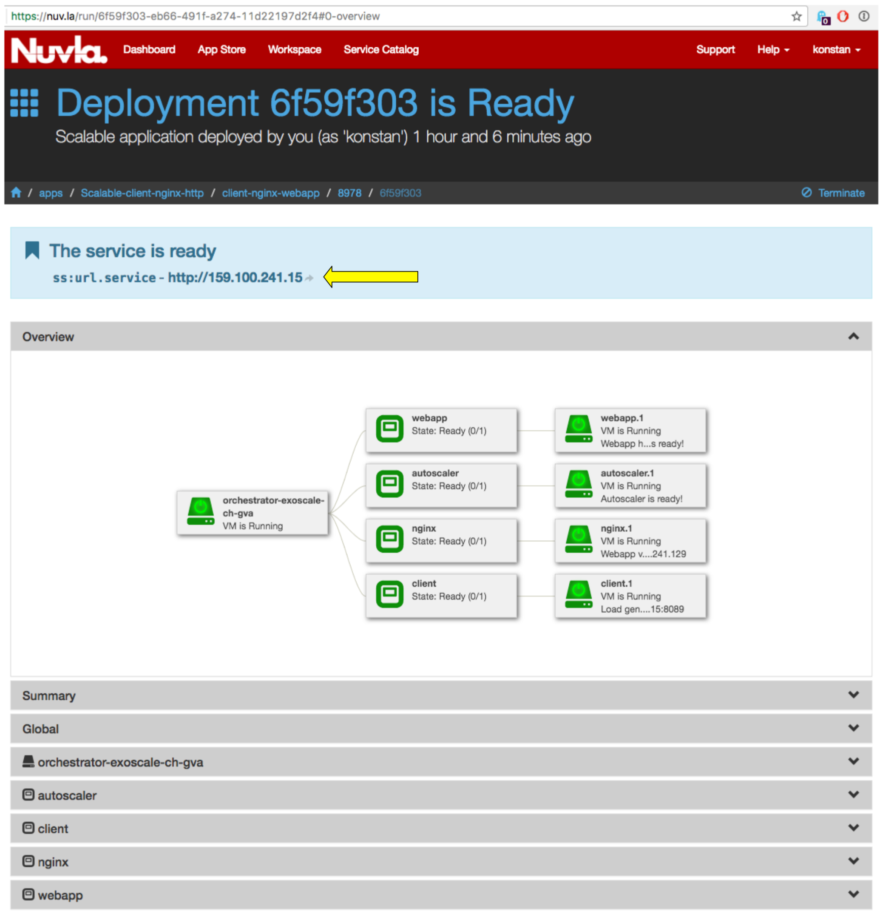
Scalable deployment	✓
Tolerate deployment failures	☐
SSH access	✓
Hybrid deployment	☐
Cloud	exoscale-ch-gva Tiny 1/1024/10 * (€ 0.0863/h)
Cost for orchestrator(s)	exoscale-ch-gva: € 0.0076/h
Keep running after deployment	Always
Tags	

Component parameters

autoscaler	Initial multiplicity	1
	Input `riemann_config_url`	https://raw.githubusercontent.com/slipstream/slipstı
	Input `scale_constraints_url`	https://raw.githubusercontent.com/slipstream/slipstı
	Input `ss_proxy_http_insecure`	false
client	Initial multiplicity	1
nginx	Initial multiplicity	1
webapp	Initial multiplicity	1

Choose the cloud where each application component will be deployed, how many of each (parameter `Multiplicity`) and parameters for each node.

Fig. 4: Deployment of Example Scalable Application



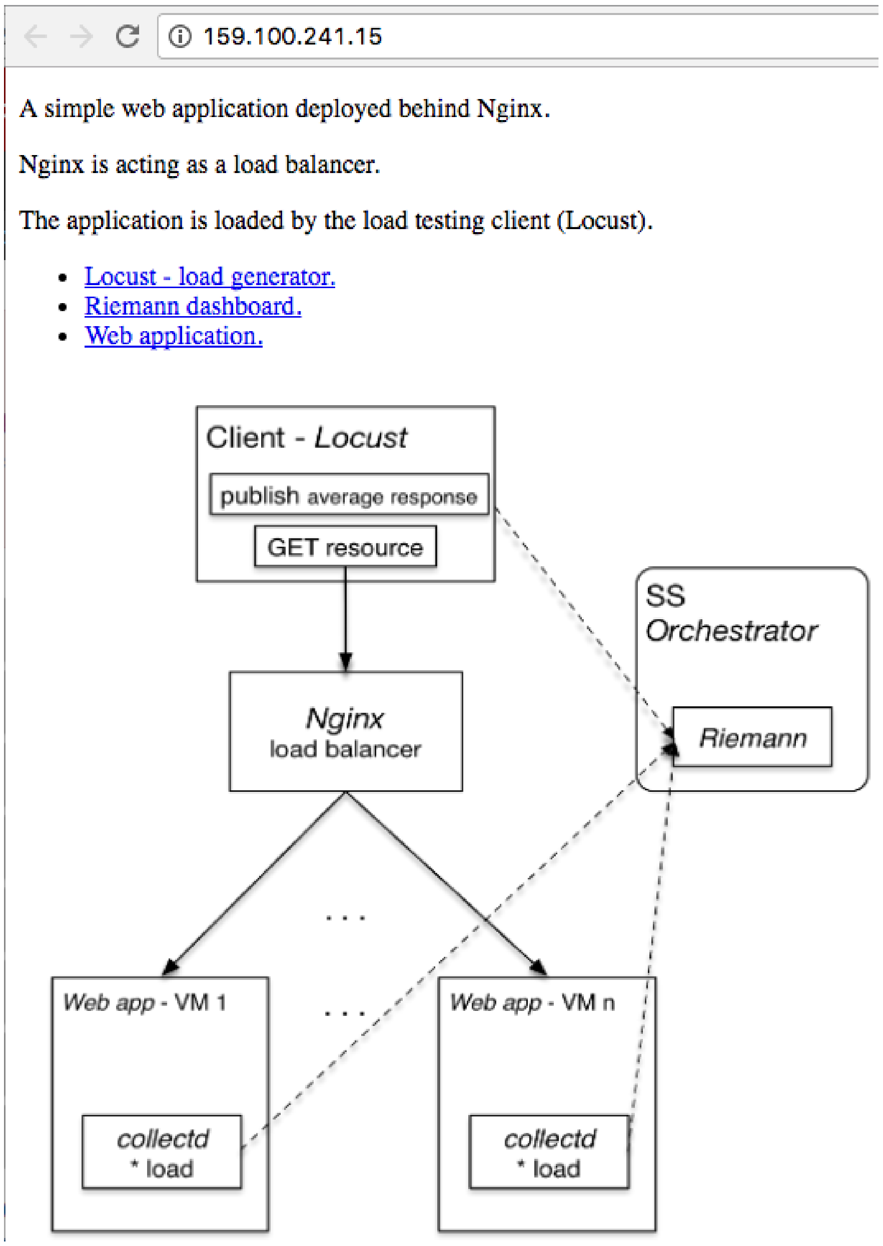
Deployment 6f59f303 is Ready

Scalable application deployed by you (as 'konstan') 1 hour and 6 minutes ago

The service is ready

`ss:url.service` - http://159.100.241.15

Fig. 5: Deployed Autoscalable Application



A simple web application deployed behind Nginx.

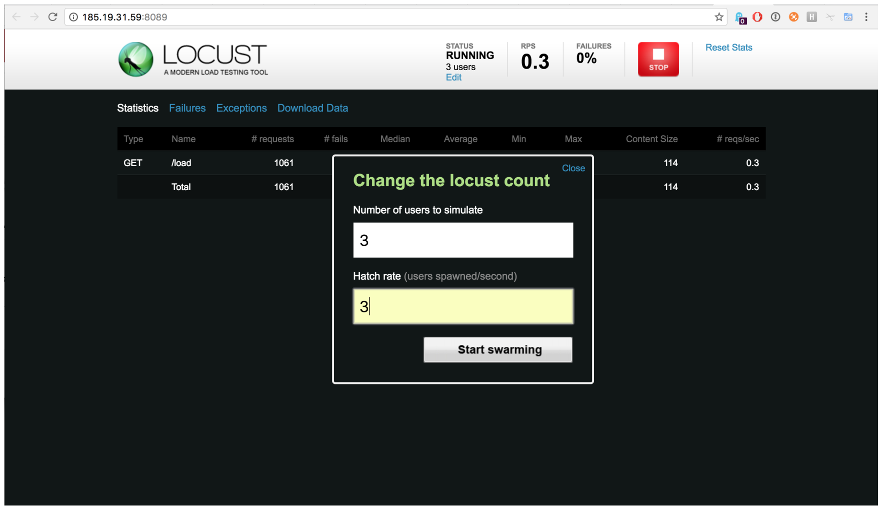
Nginx is acting as a load balancer.

The application is loaded by the load testing client (Locust).

- Locust - load generator.
- Riemann dashboard.
- Web application.

Fig. 6: Application Entry Point

and load it with requests. The endpoint to contact and the resource of the web application were already automatically configured during the component deployment.



STATUS RUNNING 3 users Edit | RPS 0.3 | FAILURES 0%

Type	Name	# requests	# fails	Median	Average	Min	Max	Content Size	# reqs/sec
GET	/load	1061						114	0.3
	Total	1061						114	0.3

Change the locust count

Number of users to simulate: 3

Hatch rate (users spawned/second): 3

Fig. 7: Locust: Defining Work (as 3 Users) to Load the Application

After Locust starts loading the web application, a custom Riemann events publisher collects the average response time metric from Locust and publishes it as an event to the Riemann service running on the autoscaler. The event looks like this:

```
{"service": "avg_response_time",
  "tags": ["webapp"],
  "ttl": 10,
  "host": "httpclient",
  "time": 1479202972,
  "metric_f": 3167.896}
```

Other metrics are also collected and sent from client (Locust) and webapp component instances. For example:

- The *client* publishes number of concurrent clients used by Locust, current requests per second, and the request fail rate;
- The *webapp* instances publish their current 1, 5, and 15 min load. This metric is published through the Riemann collectd plugin. This is deployed and configured automatically on each *webapp* with the `collectd.sh` script available from the repository.

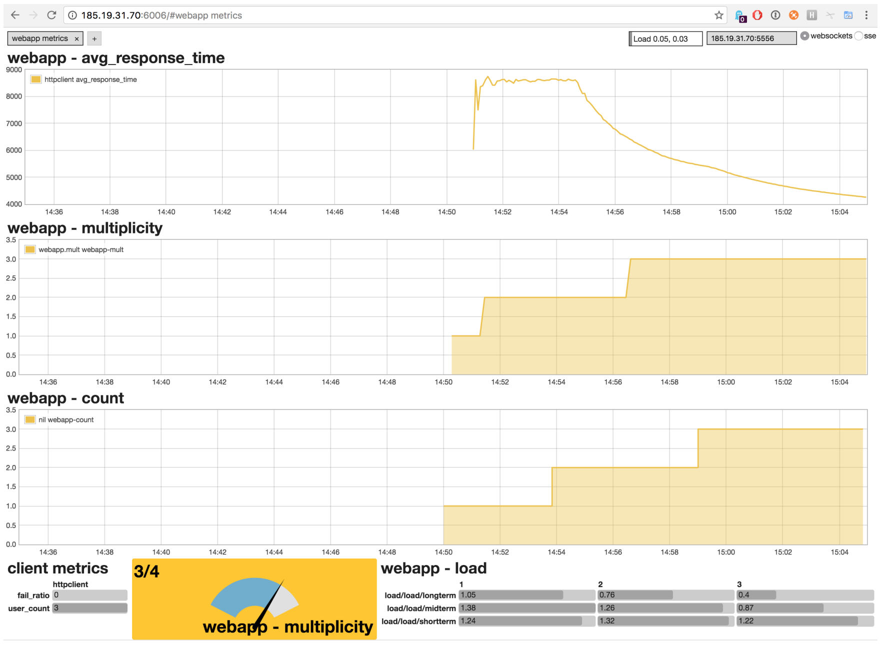
Based on the load metrics reported by the *webapp* instances, a Riemann stream dynamically calculates the current number of the instances and indexes it to allow the Riemann dashboard to query and display it.

The current multiplicity of the *webapp* component is queried by a Riemann stream directly from SlipStream and indexed. As one can see in the figure below, there is a delay between the provision request (multiplicity as reported by SlipStream) and availability of the virtual machine (as reported by the load metrics). This is almost entirely due to the provisioning latency on IaaS level; SlipStream's control flow contributes negligibly to the latency.

Based on the given constraints, the autoscaler attempted to satisfy the scalability constraints provided for the *webapp* component by keeping the average response time metric within the requested bounds.

All the application level metrics (native or generated) are published to Graphite, which is deployed on the autoscaler and runs alongside Riemann. See the figure below, which shows historical evolution of the average response time in Graphite.

From here you may want to learn more about SlipStream automation, application optimization, and debugging in *Module V*.



client metrics

	httpclient
fail_ratio	0
user_count	3

webapp - load

	1	2	3
load/load/longterm	1.05	0.76	0.4
load/load/midterm	1.38	1.26	0.87
load/load/shortterm	1.24	1.32	1.22

Fig. 8: Riemann-dash: High Load on the Web Layer

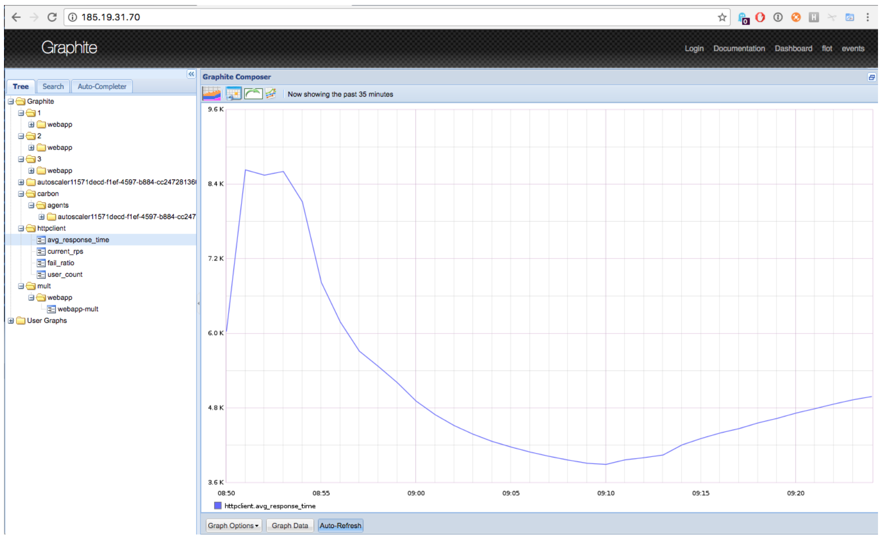


Fig. 9: Average Response Time as a Function of Time in Graphite

3.1.5 Module V - Automation and Optimization (Advanced)

Instead of working with SlipStream via web browser interface, you may want to use the SlipStream Command Line Interface (CLI) or the SlipStream REST API. These are useful for scripting or interacting with SlipStream via another service.

This module provides an overview of **SlipStream API** and demonstrates how to authenticate with SlipStream and manage the full lifecycle of applications. The module also covers the SlipStream **build image** feature, which optimizes application deployments by minimizing start up latencies.

Cloud applications, like all things in the universe, break or go wrong. See the section on **debugging of applications** to understand how to troubleshoot problems efficiently.

Automating SlipStream

Although the SlipStream web interface provides a great interface for humans, there are cases where programmatic access is more appropriate. These cases include the need for bulk operations, programmatic control for continuous integration systems, and in a few cases access to features that don't yet appear in the web interface.

In this chapter, you'll learn about programmatic access to SlipStream through the command line client and through the REST API. The text and exercises will concentrate on the management of components and applications.

The REST API is documented on the http://ssapi.sixsq.com website. It provides the full list of SlipStream resources. Although we strive for complete documentation, there may be some holes in the API docs. If you run into trouble, please contact support and we'll give you a hand.

Setup

Command Line Client

The SlipStream client is installed by default on all machines started through SlipStream. To install it on your local machine, it can be installed with `pip`, the python installer.

You will need to have python (v2.6+, not v3.x) installed on the machine as well as `pip`. See the Python and pip websites for installation of these tools on your machine.

Once python and `pip` are available, just running the following should work:

```
$ pip install slipstream-client
```

for a system-level installation. If you don't have administrator access to your machine, you can also perform a user-level installation:

```
$ pip install --user slipstream-client
```

In this case, be sure to modify your `PYTHONPATH` and `PATH` to include the directories used by the `pip` user-level installation.

cURL

The REST API uses the standard HTTP protocol, so any HTTP library can be used to access the API. A ubiquitous tool for quick HTTP requests is cURL. This is available natively on most operating systems.

If it is not available and you want to use it, see the cURL website.

If you're going to use the cURL command line extensively, you will probably want to define the following alias:

```
$ alias ss-curl="curl --cookie-jar ~/cookies -b ~/cookies -sS"
```

This avoids having to repeat the options that you'll need for every command. The following text, will assume that this alias has been defined.

Advanced REST Client

The Advanced REST Client (ARC) extension to the Chrome web browser allows a more graphical method of sending HTTP requests and visualizing the responses. To be able to reuse authentication cookies with ARC, you will have to install ARC cookie exchange Chrome extention as well.

Consult the referenced web sites for installation of Chrome and the REST client along with its cookie exchange on your machine.

Authentication

Just like with the web interface, you must be authenticated to use most of the SlipStream features through the command line or the REST API. At the moment, SlipStream supports both basic authentication and cookie-based authentication.

Command Line Client

The command line client uses primarily basic authentication. When using the client, you must supply your SlipStream username and password either via environmental variables or via command line options.

The environmental variables to set are:

```
$ export SLIPSTREAM_USERNAME=ssuser
$ export SLIPSTREAM_PASSWORD=sspass
```

substituting the `ssuser` and `sspass` values with your own username and password. You can also set these values explicitly on the command line with the `-u` or `--username` options for the username and with the `-p` or `--password` options for the password.

You can quickly test the authentication by listing your user account information:

```
$ ss-user-get ssuser
```

again changing `ssuser` to your username. This should return an XML representation of your user account.

cURL

It is strongly recommended to use cookie-based authentication when using the REST API. The basic authentication method is deprecated and will be removed at some point in the near future.

To obtain a valid authentication cookie, you must login to SlipStream, much like you'd do with the web interface. To do this, create a "session create" document like the following:

```
{
  "sessionTemplate": {
     "href": "session-template/internal",
     "username" : "your-username",
     "password" : "your-password"
```

(continues on next page)

(continued from previous page)

```
  }
}
```

substituting `your-username` and `your-password` with your actual username and password. Name the file something like `session-create-internal.json`.

You can then POST this document to the session resource collection to create a new session and to recover an authentication cookie.

Doing this with cURL (and the alias defined above!):

```
$ ss-curl https://nuv.la/api/session \
    -D - \
    -o /dev/null \
    -XPOST \
    -H content-type:application/json \
    -d@session-create-internal.json
```

This should return only the headers from the response, which should include a "Set-Cookie" header with a value and a "201 created" response code. The cookie should also end up in the `~/cookies` file.

If you provide the wrong credentials, you will get a "403 forbidden" response.

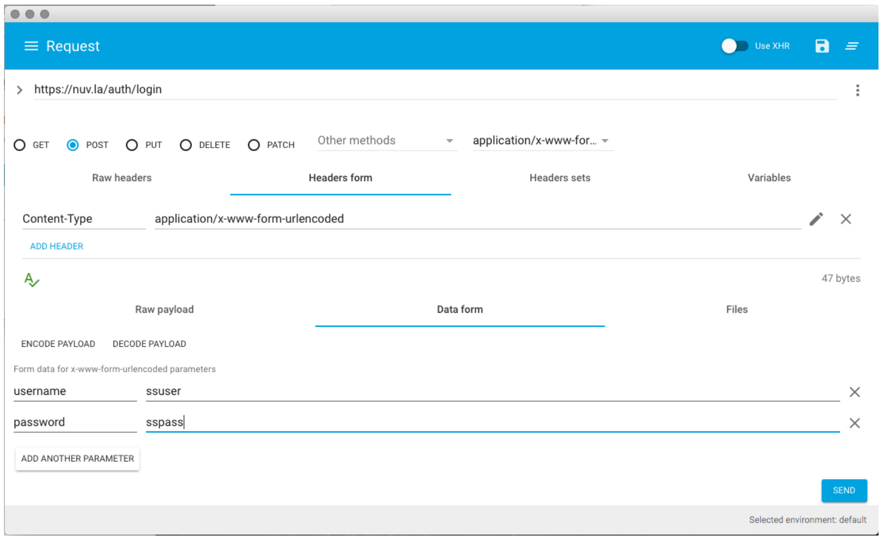
The "internal" login method is always available. You may also be able to log in via another method. You can find the full list of available methods by listing the Session Template resources:

```
$ ss-curl https://nuv.la/api/session-template
```

Note that external methods (e.g. GitHub) may not support login workflows that are adapted to command line interactions.

Advanced REST Client

Doing this with the Advanced REST Client in Chrome, you can fill in the form for the login request, which should look like the following screenshot.



https://nuv.la/auth/login

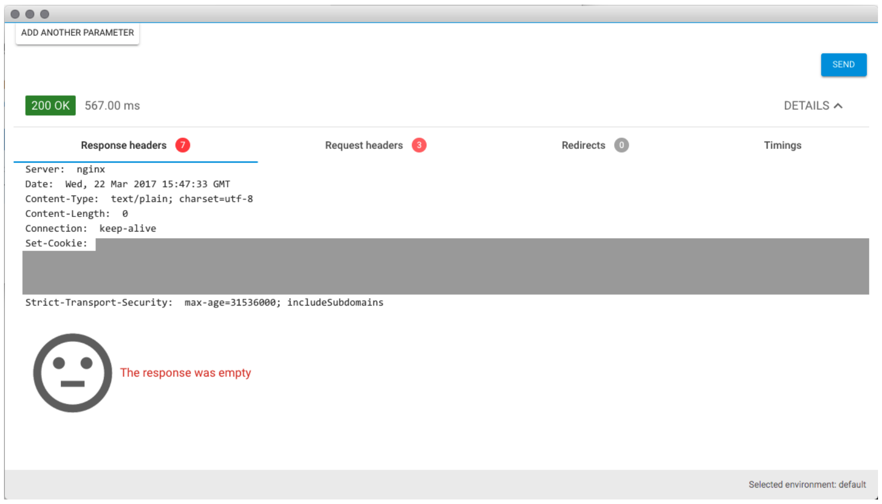
[✓ POST] application/x-www-for...

Content-Type: application/x-www-form-urlencoded

Form data for x-www-form-urlencoded parameters

username	ssuser
password	sspass

and which should return something like the following screenshot.



```
Server:  nginx
Date:  Wed, 22 Mar 2017 15:47:33 GMT
Content-Type:  text/plain; charset=utf-8
Content-Length:  0
Connection:  keep-alive
Set-Cookie: [redacted]
Strict-Transport-Security:  max-age=31536000; includeSubdomains
```

As for cURL, the "Set-Cookie" header should have a value. To automatically reuse the cookie for the next requests in the Advanced REST Client, you need to have the ARC cookie exchange extension installed and enabled in Chrome and also enabled in ARC: in the right corner the `Use XHR` should be turned on.

Note: If you want to logout by destroying your access cookie, then you can either delete the cookie manually or send a HTTP DELETE request to the logout resource http://nuv.la./logout.

Managing an Application

Command Line Client

To deploy an application or a component via the command line client use the `ss-execute` command. To deploy the web server and client application defined earlier:

```
$ ss-execute --parameters="server:title=Great Title" \
             --kill-vms-on-error \
             Training-2015-11/nginx-test-app
```

This will return the URL of the created run.

This is essentially a "shoot and forget" feature intended for deploying test applications. There are no comparable commands for finding the application's status or terminating it. Those actions either need to be done through the web interface or REST API.

cURL

The REST API allows complete control over the application lifecycle.

To start an application with cURL, you must use the following command:

```
$ ss-curl https://nuv.la/run \
  -X POST \
  -d refqname=Training-2015-11/nginx-test-app \
  -d keep-running=always \
  -d parameter--node--server--title='Great Title' \
```

(continues on next page)

(continued from previous page)

```
  -H 'Accept: application/xml' \
  -D - \
  -o response-body.txt
```

This will send a POST request to the "run" resource to start an application. The "Location" header will contain the run identifier if the command completes successfully.

The command shows how parameter values are encoded for the REST API. You can also specify other parameters such as the "keep-running" value. The "refqname" is required as it identifies the application to run.

You can see the full state of the run by performing a GET request on the given run URL:

```
$ ss-curl https://nuv.la/run/815a8f66-fc9d-4444-849c-d12e883982c1
```

All of the responses related to the applications and runs are in XML. Newer resources such as events and usage are in JSON. All of the resources will eventually be migrating toward JSON in the future.

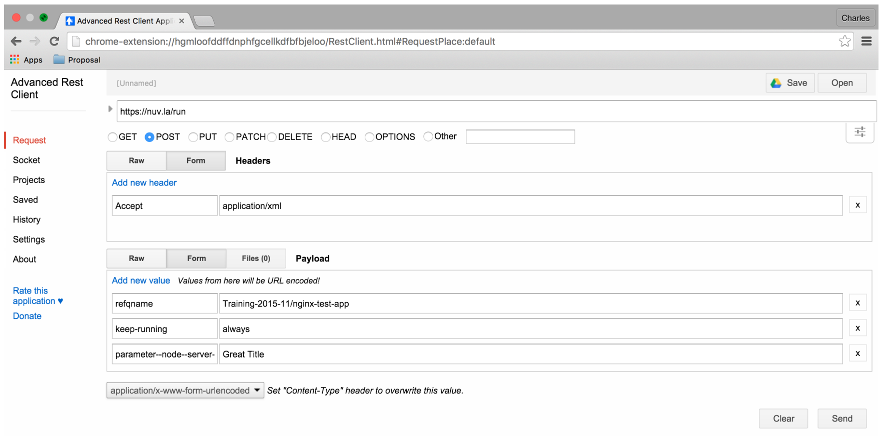
You can then terminate the run by sending a DELETE request to the given run URL:

```
$ ss-curl -X DELETE \
  https://nuv.la/run/815a8f66-fc9d-4444-849c-d12e883982c1
```

This will immediately terminate the application, so be careful when using DELETE requests.

Advanced REST Client

You can perform the same lifecycle with the Advanced REST Client. Start with deploying the application.



chrome-extension://hgmloofddffdnphfgcellkdfbfbjeloo/RestClient.html#RequestPlace:default

https://nuv.la/run

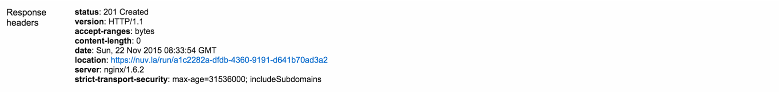
GET [✓ POST] PUT PATCH DELETE HEAD OPTIONS Other

Accept	application/xml

refqname	Training-2015-11/nginx-test-app
keep-running	always
parameter--node--server-	Great Title

application/x-www-form-urlencoded

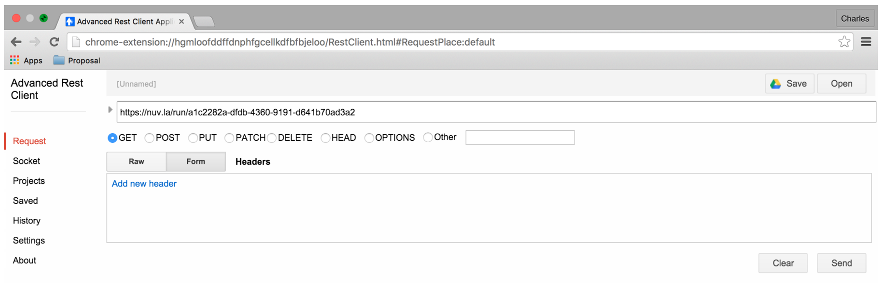
If successful, it will return a 201 response with the run identifier in the "Location" header.



Response headers

status: 201 Created
version: HTTP/1.1
accept-ranges: bytes
content-length: 0
date: Sun, 22 Nov 2015 08:33:54 GMT
location: https://nuv.la/run/a1c2282a-dfdb-4360-9191-d641b70ad3a2
server: nginx/1.6.2
strict-transport-security: max-age=31536000; includeSubdomains

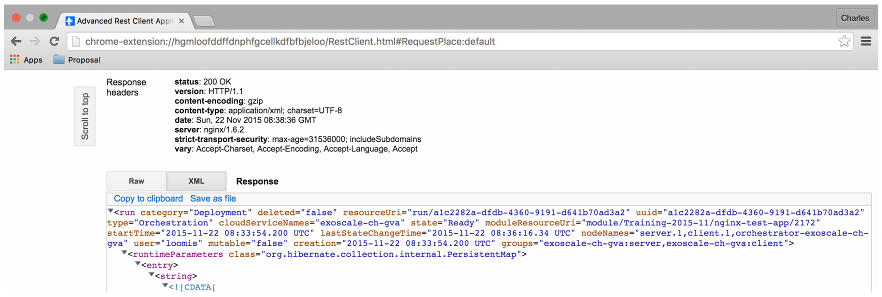
Performing a GET on the returned URL will give you the status.



chrome-extension://hgmloofddffdnphfgcellkdfbfbjeloo/RestClient.html#RequestPlace:default

https://nuv.la/run/a1c2282a-dfdb-4360-9191-d641b70ad3a2

[✓ GET] POST PUT PATCH DELETE HEAD OPTIONS Other

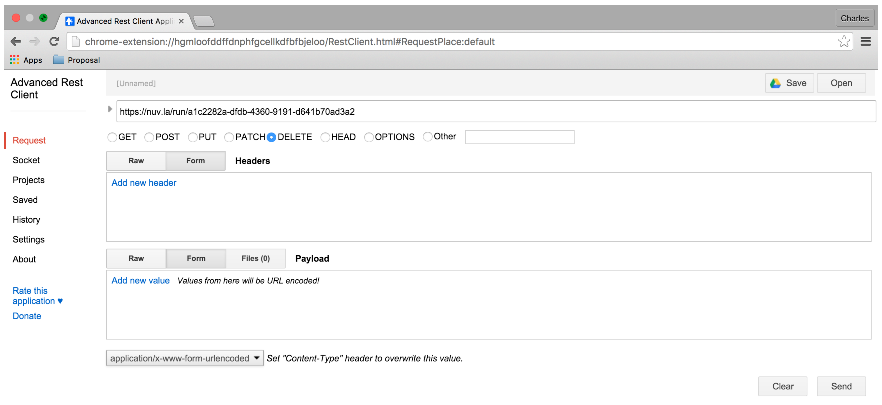


chrome-extension://hgmloofddffdnphfgcellkdfbfbjeloo/RestClient.html#RequestPlace:default

```
Response    status: 200 OK
headers     version: HTTP/1.1
            content-encoding: gzip
            content-type: application/xml; charset=UTF-8
            date: Sun, 22 Nov 2015 08:38:36 GMT
            server: nginx/1.6.2
            strict-transport-security: max-age=31536000; includeSubdomains
            vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
```

```
<run category="Deployment" deleted="false" resourceUri="run/a1c2282a-dfdb-4360-9191-d641b70ad3a2" uuid="a1c2282a-dfdb-4360-9191-d641b70ad3a2"
type="Orchestration" cloudServiceNames="exoscale-ch-gva" state="Ready" moduleResourceUri="module/Training-2015-11/nginx-test-app/2172"
startTime="2015-11-22 08:33:54.200 UTC" lastStateChangeTime="2015-11-22 08:36:16.34 UTC" nodeNames="server.1,client.1,orchestrator-exoscale-ch-
gva" user="loomis" mutable="false" creation="2015-11-22 08:33:54.200 UTC" groups="exoscale-ch-gva:server,exoscale-ch-gva:client">
  <runtimeParameters class="org.hibernate.collection.internal.PersistentMap">
    <entry>
      <string>
        <![CDATA[
```

And finally the application can be terminated with a DELETE request.



chrome-extension://hgmloofddffdnphfgcellkdfbfbjeloo/RestClient.html#RequestPlace:default

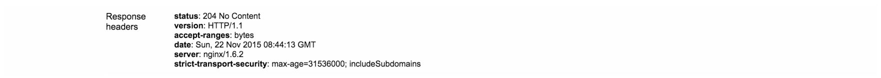
https://nuv.la/run/a1c2282a-dfdb-4360-9191-d641b70ad3a2

GET POST PUT PATCH [✓ DELETE] HEAD OPTIONS Other

Add new value Values from here will be URL encoded!

application/x-www-form-urlencoded Set "Content-Type" header to overwrite this value.

A successful termination will return a "No Content 204" response.



```
Response    status: 204 No Content
headers     version: HTTP/1.1
            accept-ranges: bytes
            date: Sun, 22 Nov 2015 08:44:13 GMT
            server: nginx/1.6.2
            strict-transport-security: max-age=31536000; includeSubdomains
```

EXERCISES

1. Start your web test application with the `ss-execute` command.
2. Perform a full lifecycle of your web test application with the REST API.
3. Perform a full lifecycle of your web component with the REST API.

Faster Deployments

The software installation and much of the configuration for a component are static. Repeating this for every deployment increases the startup latency for the component and for the application as a whole.

"Building" an image, creates a new native image within a cloud infrastructure that already includes the static initialization, reducing the deployment latency.

This should only be considered as an optimization after the deployment is debugged and working because:

- Building an image can take a significant amount of time (20+ minutes)
- Not all clouds support user-generated images
- Some clouds require additional configuration to share the generated images with others
- Securing persistent images and ensuring they work can be difficult.

Nonetheless, the additional effort can significantly speed the deployment for frequently deployed applications, improving the customer's experience and reducing their costs.

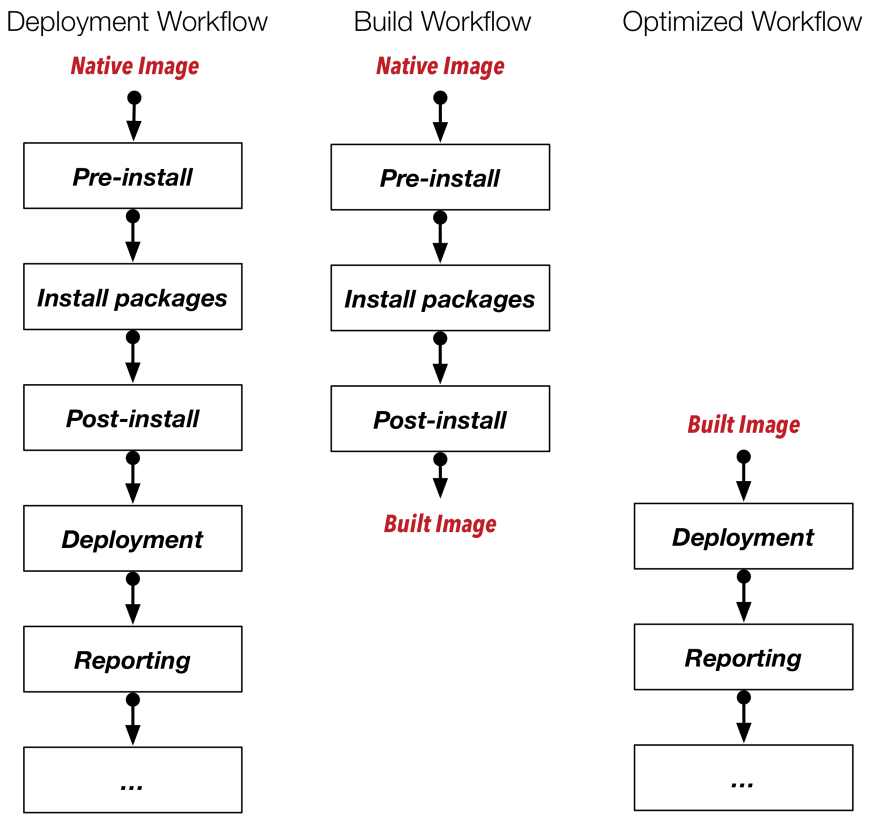
Worth the Effort?

Before deciding whether to build an image, measure the latencies in your actual application deployment. For each deployed application you can see the times at which the application entered each state. The time spent in the "executing" state is the maximum amount of time by which you can reduce the startup times.

Also determine whether the clouds that you target support user-defined images. If you will need to support cloud infrastructures that both support and do not support user-defined images, then you will need to be especially careful to ensure that the component behaves identically on both infrastructures.

Understanding Workflows

Presumably if you're still reading, then you've decided that building images is interesting for your application. You'll need to understand how the standard application workflow differs from the build workflow. The following diagram shows which scripts are run for the standard deployment workflow, for the build workflow, and for the optimized workflow with a built image.



As you can see from the diagram, the “Pre-install”, “Install packages”, and “Post-install” are not run in the optimized workflow because those actions were already completed in the build workflow and saved in the created image. The work done in those phases represents the speed up you can achieve by building an image.

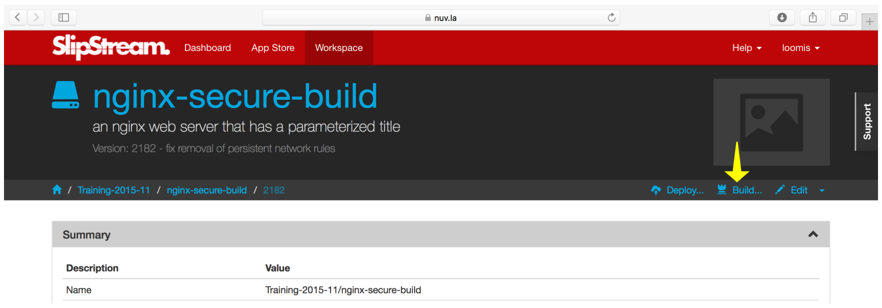
Building an Image

If you’ve followed the advice about what type of actions to put into each recipe, then your component should be nearly ready to be built. The important distinction is to ensure that **static** installation and configuration is done in the “Post-install” recipe and before. All dynamic configuration should be put into the “Deployment” recipe.

However to ensure that the component works identically on all of the clouds, you should consider adding a few things to your component description:

- Operating systems tend to persist networking configuration. Make sure all such files are removed from the system in the “Post-install” recipe. This usually comes down to a command like: `rm -f /etc/udev/rules.d/*net*.rules`.
- Any command histories, temporary files, etc. will be saved in the generated image. If you’ve used or generated sensitive information, be sure to remove it from the virtual machine in the “Post-install” recipe.
- Upgrading a system often involves the installation of a new kernel. To ensure that the new kernel is used in both standard and optimized workflows, you may want to insert a reboot of the machine at the end of the “Post-install” recipe. SlipStream is tolerant of such reboots.
- Similarly for a built image, you may want to also upgrade the system at the beginning of the “Deployment” recipe to pick up any new packages since the image was created.

After these changes, building an image is just a matter of clicking on the “Build…” button for the component you want to create.



nginx-secure-build

an nginx web server that has a parameterized title

Version: 2182 - fix removal of persistent network rules

/ Training-2015-11 / nginx-secure-build / 2182

Summary

Description	Value
Name	Training-2015-11/nginx-secure-build

You can then follow the progress on the dashboard as with any other deployment. When completed, you will see an image identifier has been added to the component in the “Cloud Image Identifiers and Image Hierarchy” section.

When SlipStream encounters a component with such an image identifier it will always use the optimized workflow for that cloud.

Warning: SlipStream will invalidate the built image if any of “Pre-install”, “Install packages” or “Post-install” of the component are modified; you must rebuild the component after any change in any of those steps of the Application Workflow. Also, SlipStream will not delete the built image from the cloud. That must be done manually, if desired.

Debugging

Things go wrong. That is just the nature of the world. SlipStream provides various methods for debugging problems when they arise.

When creating a new application, very often there are bugs in the deployment scripts. Iteratively modifying the scripts through SlipStream and redeploying the machines can cause unnecessary delays. **If you have selected an option that lets failed deployments continue to run**, you can instead:

1. Log into a failed deployment,
2. Setup the environment for the SlipStream client (see below),
3. Reset the abort flag with the `ss-abort --cancel` command, and
4. Update and rerun the deployment script(s)

This allows for a much faster development cycle. The deployment scripts can be found in the directory `/var/lib/slipstream`, with names that correspond to each phase of the deployment. The logs from the initial execution of these scripts are below

- `/var/log/slipstream/client` on Linux
- `%TMP%\slipstream\reports` on Windows.

Once the problems in the deployment scripts have been ironed out, just copy them back into SlipStream.

Modifying the Environment

SlipStream minimizes its footprint to avoid any unintended interference with the deployed applications. Because of this, you must specifically setup the environment to make the SlipStream client commands accessible. Usually you will want to do the following:

```
$ source /opt/slipstream/client/sbin/slipstream.setenv
```

You should then have all of the SlipStream client commands (all prefixed with `ss-`) in your path. All of the commands support the `--help` option to give you information about the command.

Major Client Commands

There are actually just a few commands in the SlipStream client that are used in deployment scripts and in debugging. The following table summarizes them.

`ss-get`	Retrieves a named parameter, waiting if the parameter has not yet been set.
`ss-set`	Sets the value of a named parameter.
`ss-random`	Generates a random string value and optionally sets a named parameter with this value.
`ss-abort`	Sets the deployment abort flag or clears it with the `--cancel` option.
`ss-display`	Sets a string in the run for display purposes.

All of the parameters used in the deployment must have been defined in the components used in the deployment. Trying to set or get an undefined parameter will cause the command to raise an error, which in turn aborts the run.

Important: Although the "parameter database" and the associated commands are quite simple, the fact that `ss-get` will wait for a value to be set allows it to act as a semaphore to coordinate the configuration scripts on different machines in a multi-node deployment.

EXERCISES

1. Log into a machine that has been deployed via SlipStream and setup the environment to access the SlipStream client.
2. Understand the options and behavior of the major commands by looking through the `--help` text.
3. Find the deployment scripts for the machine that you've deployed. Try executing them by hand to see what happens.
4. Set and clear the abort flag for your deployment. How does the behavior of `ss-set` and `ss-get` change when the abort flag is set?

If you've not done so already, you might want to follow *Module IV* to learn about scalable applications.

3.1.6 Module VI - Data Management

S3 (Simple Storage Service) object storage has become a *de facto* standard since its introduction by Amazon Web Services in 2006. IaaS cloud infrastructures and cloud software provide S3-compatible object storage nearly universally. SlipStream takes advantage of these services and its own CIMI resource database to provide a **multi-cloud object storage solution**.

Benefits of the SlipStream multi-cloud object storage include:

- **Global queries/views** of your data across clouds, enabled by the centralized storage of object metadata in SlipStream.
- **Uniform access control** across all clouds, using SlipStream users' identities and roles.
- **Efficient IO** via direct access to the S3 storage service holding an object by any client.
- All data access is via HTTPS, so **no special client software or APIs are required**.
- Easy **replication and caching** strategies enabled by Write-Once, Read-Mostly (WORM) semantics.
- Definition of **rich object metadata** via optional use of the SlipStream Service Catalog.
- **Low complexity** to facilitate use.
- **Scalable** architecture to handle large datasets.

With this solution, you can take advantage of cloud-native storage from multiple providers without the pain of manual bookkeeping and application-level authorization.

Note: This functionality is in **active development**. Feedback on all aspects of the multi-cloud object storage solution is welcome, but real-world feedback on the existing implementation is especially useful.

Data Management Model

The SlipStream multi-cloud object storage provides **Write-Once, Read-Mostly (WORM) semantics**. That is, a particular object is created, populated with data, and then remains immutable for the rest of its lifecycle. This makes strategies for replication and caching easier to implement.

Model Entities

The full data management model contains three entities that interact to provide all the stated benefits:

- **S3 object storage** on the supported cloud infrastructures,

- **ExternalObject** resources within SlipStream that provide links to the real data objects in the S3 object storage, and
- **ServiceOffer** resources that optionally provide rich, user-defined metadata for an object or set of objects.

For efficiency, all IO operations on a data object occur directly between the client and the S3 object store containing the object. In contrast, the management functions (creating, updating properties, and deleting) are handled through SlipStream via the standard CIMI CRUD actions on the data object's ExternalObject and/or ServiceOffer resource.

Operations

Accessing a data object follows the simple, three-step process shown in the following diagram.

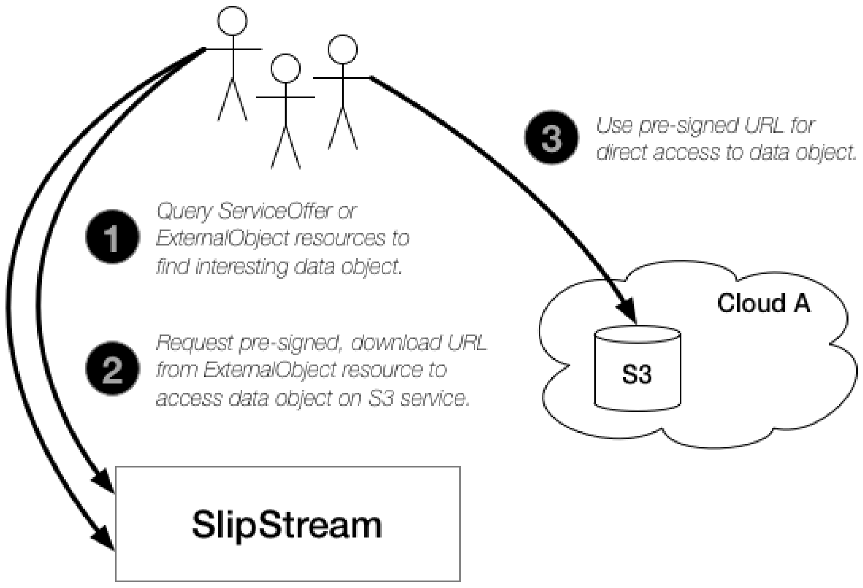

3
Use pre-signed URL for direct access to data object.
1
Query ServiceOffer or ExternalObject resources to find interesting data object.
Cloud A
2
Request pre-signed, download URL from ExternalObject resource to access data object on S3 service.
S3
SlipStream

Fig. 10: Data Access Process

The process for **creating data objects** is similarly easy:

1) Create ExternalObject resource in SlipStream,
2) Request a pre-signed upload URL,
3) Upload the data, and
4) Mark the ExternalObject as "ready".

Here, the explicit state change of the object to "ready" allows the owner of the object to indicate when the object is ready for consumption by others.

Deleting an object is a single step process that synchonizes the deletion of the metadata and the actual object within the S3 storage.

Object Metadata

The metadata for objects is stored in SlipStream, either in ExternalObject or ServiceOffer resources.

For many, simple use cases, the attributes available on the ExternalObject resource will be sufficient. These attributes include a name, description, simple properties, bucket name, and object name.

For those use cases that require richer metadata, ServiceOffer resources can be used together with the ExternalObject resources. The open schema of the ServiceOffer resource allows any general or domain-specific attributes to be associated with the data objects and the standard CIMI filtering provides rich queries.

Authorization

Access to objects is controlled through SlipStream ACLs. Those users/roles with "modify" access to an object can upload data, download data, and delete the object. Those with "view" access can only download data. Any authenticated user can create an object.

Links to Data Objects

Data objects can be very large and often contain binary data. To avoid capacity, performance, and security issues, the data objects are not directly managed within the SlipStream database. Instead, SlipStream manages "links" to these data objects, providing "actions" that facilitate creating, reading, writing, and deleting the objects.

The following sequence diagram provides an overview of these actions and what entities are involved to complete them.

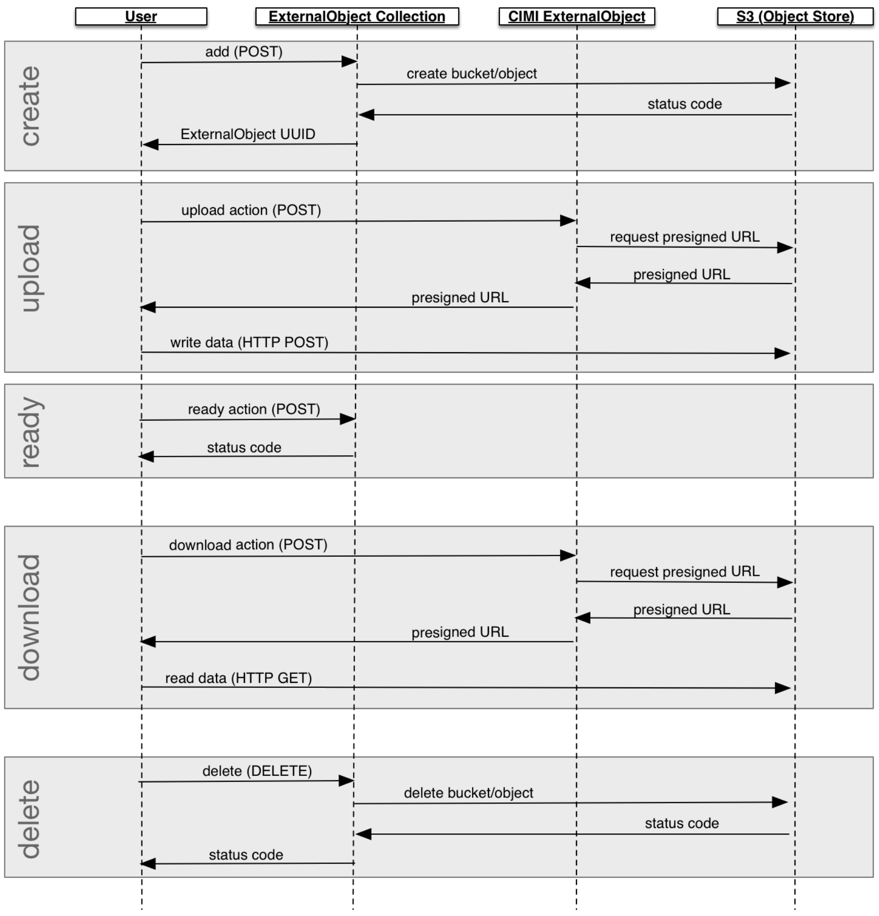


Fig. 11: Sequence Diagram of ExternalObject Actions

Create

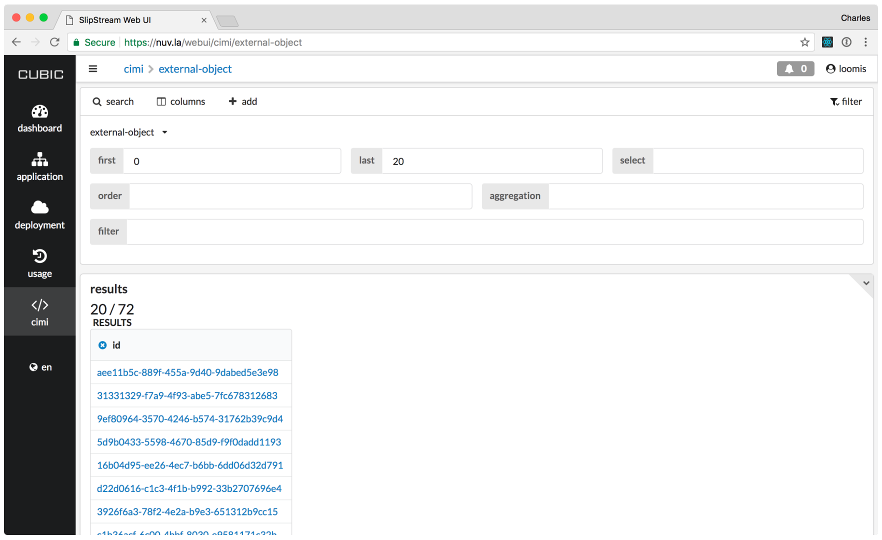
The ExternalObject resource is a templated CIMI resource. This means that the creation request must reference an ExternalObjectTemplate when creating a new ExternalObject resource.

When the server creates the resource, it:

- Checks that the bucket exists, **creating the bucket** if necessary using the referenced credentials,
- **Creates the ExternalObject** resource with the provided information, and
- **Sets the state to "new"** in preparation of having the data uploaded to the object.

If the bucket doesn't exist and cannot be created, an error will be returned. The name of the object within the bucket depends on the template used to create the object.

To create an ExternalObject resource via the web browser interface, navigate to the CIMI ExternalObject collection and click on the search action at the top to see the list of existing objects. You should see a screenshot like the following, showing an "add" action that will allow new reports to be created.



first	0	last	20	select	
order		aggregation			
filter					

results
20 / 72
RESULTS

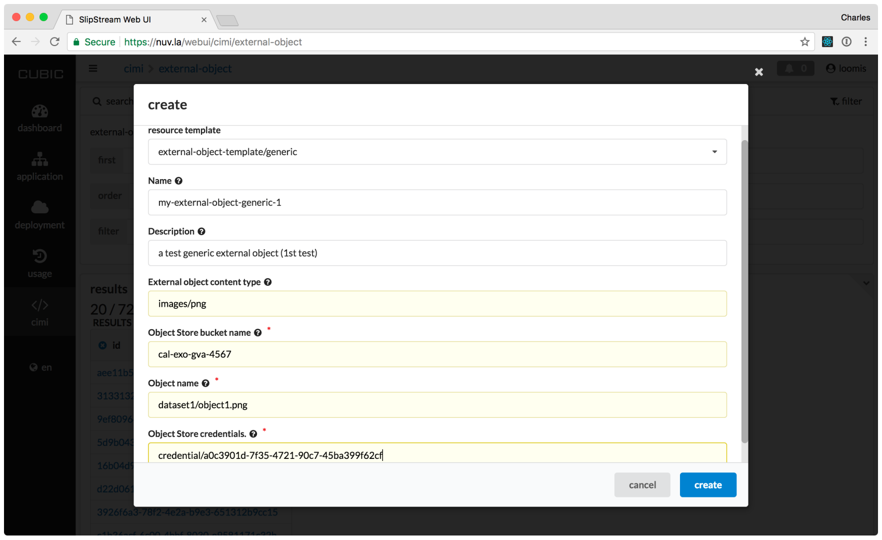
id
aee11b5c-889f-455a-9d40-9dabed5e3e98
31331329-f7a9-4f93-abe5-7fc678312683
9ef80964-3570-4246-b574-31762b39c9d4
5d9b0433-5598-4670-85d9-f9f0dadd1193
16b04d95-ee26-4ec7-b6bb-6dd06d32d791
d22d0616-c1c3-4f1b-b992-33b2707696e4
3926f6a3-78f2-4e2a-b9e3-651312b9cc15

Fig. 12: List of ExternalObject Resources

When you click on the "add" action, you will be presented with a form to create a new ExternalObject. In most cases, you will want to use the "external-object-template/generic" template; the "external-object-template/report" is for creating the reports from deployments.

In this form, you can fill in the following information:

- **resource template**: normally you will want to use the value "external-object-template/generic"
- **Name**: optional, human-readable name for the resource
- **Description**: optional, human-readable description of the resource
- **External object content type**: optional, but strongly recommended MIME type of the object
- **Object Store bucket name**: name of the bucket on the S3 storage service
- **Object name**: name of the object in S3 storage, this can be a hierarchical name using slashes as separators.
- **Object Store credentials**: reference to the cloud credential to use for the object. This implicitly defines the cloud being used.



create

resource template: external-object-template/generic

Name: my-external-object-generic-1

Description: a test generic external object (1st test)

External object content type: images/png

Object Store bucket name *: cal-exo-gva-4567

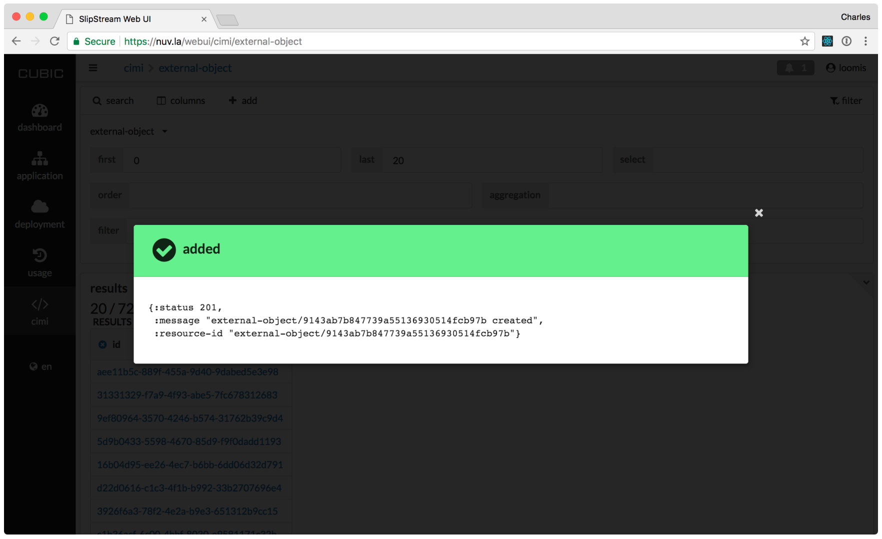
Object name *: dataset1/object1.png

Object Store credentials. *: credential/a0c3901d-7f35-4721-90c7-45ba399f62cf

Fig. 13: Form to Create a New Generic ExternalObject

The last three values are required. The identifier for the resource is a hashed value of the bucket and object names. **Only one ExternalObject with the same bucket and object names can be created.**

When you click on "create", it will then create the ExternalObject resource. You should see a success message like the following.



added

```
{:status 201,
 :message "external-object/9143ab7b847739a55136930514fcb97b created",
 :resource-id "external-object/9143ab7b847739a55136930514fcb97b"}
```

Fig. 14: Successful Creation of Generic ExternalObject

Be sure to note the identifier of the created ExternalObject. In this case, the identifier is:

```
external-object/9143ab7b847739a55136930514fcb97b
```

You can always use the search functionalities to find the resource later, if necessary.

The creation process can also be done directly with the REST API. The process looks like the following.

```
$ alias ss-curl='curl --cookie-jar ~/cookies -b ~/cookies -sS'
$
$ # create a session with the server
$ export USER=...
$ export PASS=...
$ ss-curl -XPOST \
    -d username=${USER} \
    -d password=${PASS} \
    -d href=session-template/internal \
    https://nuv.la/api/session
$
$ # response: JSON with 201 status, session identifier
$
$ # JSON document to create a generic ExternalObject
$ cat > eo-create.json <<EOF
{
  "name": "dataset1/object1",
  "description": "test image for ExternalObject resource",
  "externalObjectTemplate": {
    "href": "external-object-template/generic",
    "objectStoreCred": {
      "href": "credential/a0c3901d-7f35-4721-90c7-45ba399f62cf"
    },
  "bucketName": "data-exo-gva",
  "objectName": "dataset1/object1.png",
  "contentType": "image/png"
  }
}
EOF

$ # create the ExternalObject
$ ss-curl -XPOST \
    -H 'content-type:application/json' \
    -d@eo-create.json \
    https://nuv.la/api/external-object
$
$ # response: JSON document with 201 status, external object identifer
```

This should create the ExternalObject resource, similarly to the way it was done through the web interface.

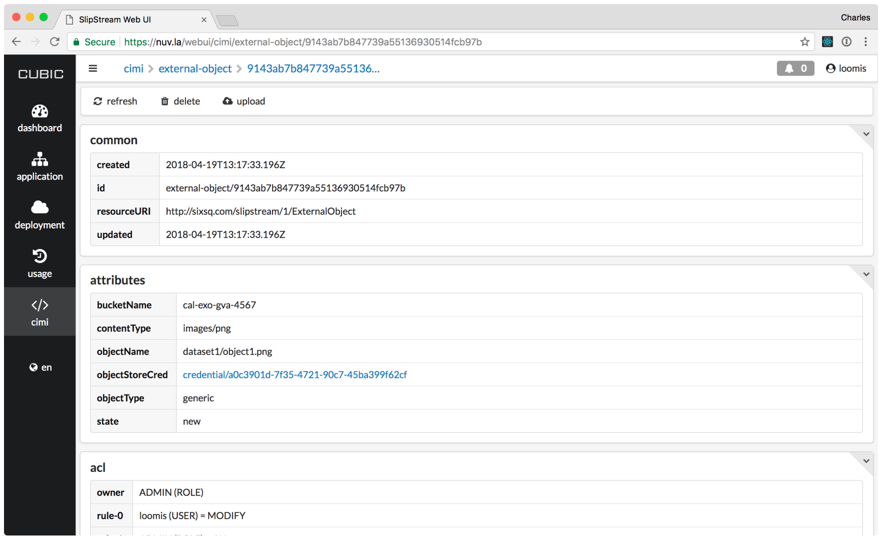
Upload Data

When the ExternalObject resource is in the "new" state, anyone with "modify" access to the resource can request a presigned upload URL for the S3 object via the CIMI action "upload" on the resource. A lifetime can be specified for the returned URL to limit security concerns with a presigned URL.

The returned presigned URL can be used to upload the contents of the object directly to the S3 object store via HTTPS. This is convenient because it **does not require direct authentication or special software** to be installed by the client uploading the data.

Once the presigned upload URL has been provided, the state of the ExternalObject resource will be changed to "uploading". Upload URLs can still be requested (for example, in the case of a data upload error), but the object cannot yet be downloaded.

In our case, we will visit the detail page for the created ExternalObject resource. You can click on the link in the collection or directly navigate to the URL which will have the UUID of the resource appended to `https://nuv.la/webui/cimi/external-object/`.



https://nuv.la/webui/cimi/external-object/9143ab7b847739a55136930514fcb97b

cimi > external-object > 9143ab7b847739a55136...

common

created	2018-04-19T13:17:33.196Z
id	external-object/9143ab7b847739a55136930514fcb97b
resourceURI	http://sixsq.com/slipstream/1/ExternalObject
updated	2018-04-19T13:17:33.196Z

attributes

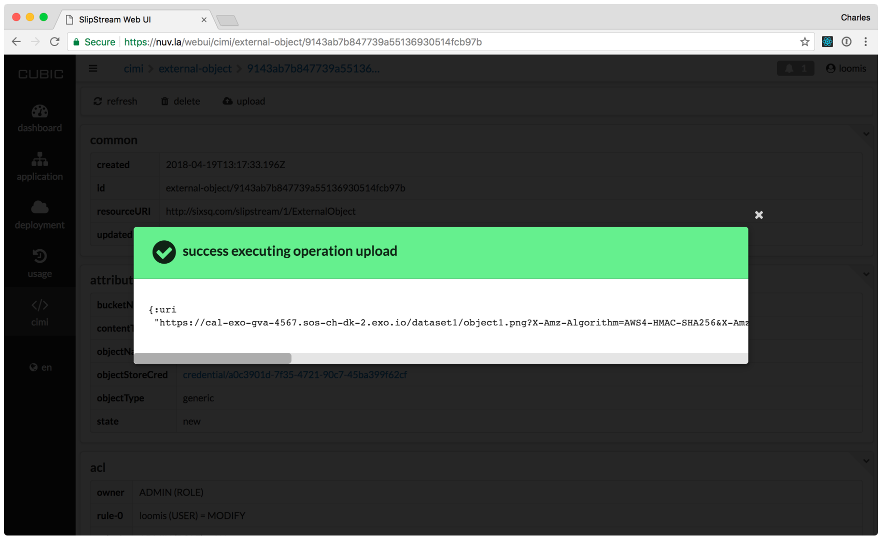
bucketName	cal-exo-gva-4567
contentType	images/png
objectName	dataset1/object1.png
objectStoreCred	credential/a0c3901d-7f35-4721-90c7-45ba399f62cf
objectType	generic
state	new

acl

owner	ADMIN (ROLE)
rule-0	loomis (USER) = MODIFY

Fig. 15: Detail of Created ExternalObject Resource

On the detail page, you can see the actions that are possible. To upload the content of the ExternalObject, click on “upload” and then note the returned upload URL.



success executing operation upload

```
{:uri
 "https://cal-exo-gva-4567.sos-ch-dk-2.exo.io/dataset1/object1.png?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz
```

Fig. 16: Requesting ExternalObject Upload URL

To request an upload URL through `curl`, you can execute the following command. (This assumes that you’ve defined the alias above and still have an active session.)

```
$ # get resource to find operation URLs
$ export LINK=https://nuv.la/api/external-object/0565...
$ ss-curl ${LINK}
$
$ # use upload operation URL to get upload URL
$ export GET_UPLOAD_LINK=${LINK}/upload
$ ss-curl ${GET_UPLOAD_LINK}
$
$ # response: JSON with uri key with the upload URL
```

You should extract the correct URL to use from the "operations" key provided in the object itself.

With the upload URL (either from the web interface or the command line), any HTTP client can upload contents to the ExternalObject. This is a presigned URL with a limited lifetime. To upload the object contents, you can do the following:

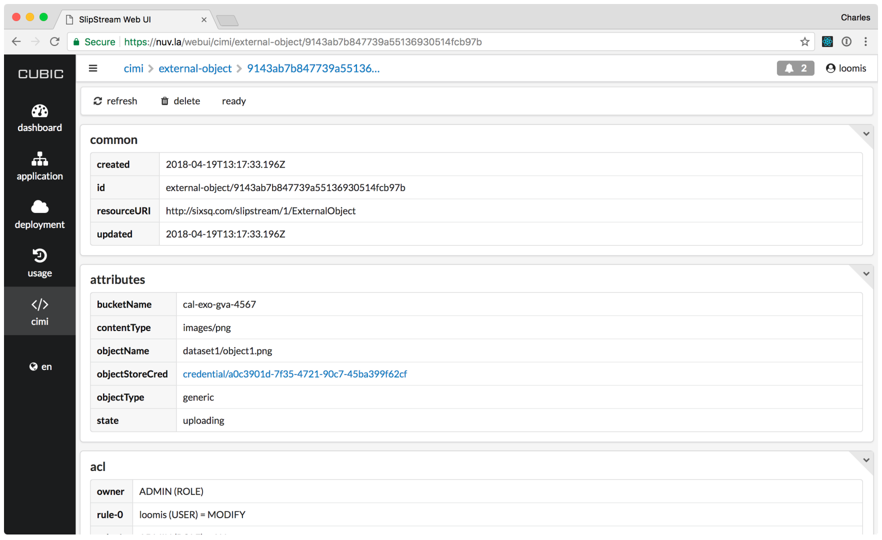
```
$ export LINK="https://cal-exo-gva-4567.sos-ch-dk-2.exo.io/..."
$
$ curl -XPUT \
    -T screenshot.png \
    -H content-type:image/png \
    ${LINK}
$ echo $?
0
```

This example uses `curl`, but any HTTP client could have been used.

Ready

To prevent further changes to the object and to allow others to download the data, you must set the ExternalObject resource's state to "ready". This is done by sending a POST request to the "ready" action URL.

For the web browser interface, you can just click on the "ready" button that appears on the detail page of the ExternalObject resource. You should see a success modal from this action. If you refresh the detail page, you'll see that the status has changed to "ready".


SlipStream Web UI
Charles
Secure https://nuv.la/webui/cimi/external-object/9143ab7b847739a55136930514fcb97b
CUBIC
dashboard
application
deployment
usage
cimi
en
cimi > external-object > 9143ab7b847739a55136...
2
loomis
refresh
delete
ready

common

created	2018-04-19T13:17:33.196Z
id	external-object/9143ab7b847739a55136930514fcb97b
resourceURI	http://sixsq.com/slipstream/1/ExternalObject
updated	2018-04-19T13:17:33.196Z

attributes

bucketName	cal-exo-gva-4567
contentType	images/png
objectName	dataset1/object1.png
objectStoreCred	credential/a0c3901d-7f35-4721-90c7-45ba399f62cf
objectType	generic
state	uploading

acl

owner	ADMIN (ROLE)
rule-0	loomis (USER) = MODIFY

Fig. 17: Setting the Ready State of ExternalObject

When using `curl` or another HTTP client, the request looks like the following.

```
$ # get resource to find operation URLs
$ export LINK=https://nuv.la/api/external-object/0565...
$ ss-curl ${LINK}
$
$ # use ready operation URL to get ready URL
$ export GET_READY_LINK=${LINK}/ready
$ ss-curl ${GET_READY_LINK}
```

(continues on next page)

(continued from previous page)

```
$
$ # response: JSON with uri key with the ready URL
```

Again this assumes that you have an active session. The URL that you use should be taken from the "operations" attribute in the ExternalObject document.

Once the object is in the ready state, upload URLs can no longer be requested; the download action will be available to those with "view" access.

Download Data

When the ExternalObject resource is in the "ready" state, anyone with "view" access to the resource can request a presigned download URL for the S3 object via the CIMI operation "download" on the resource. A lifetime can be specified for the returned presigned URL.

Similarly to the upload URL, the returned URL allows access to the data object directly on S3. It does not require direct authentication or special software by the client reading the data.

For the web browser interface, you can just click on the "download" button that appears on the detail page of the ExternalObject resource.

With the returned download URL, you can verify the contents of the ExternalObject by either visiting this URL with a web browser or using another HTTP client (like `curl` above).

```
$ # get resource to find operation URLs
$ export LINK=https://nuv.la/api/external-object/0565...
$ ss-curl ${LINK}
$
$ # use download operation URL to get download URL
$ export GET_DOWNLOAD_LINK=${LINK}/download
$ ss-curl ${GET_DOWNLOAD_LINK}
$
$ # response: JSON with uri key with the download URL
$ export DOWNLOAD_URL=...
$
$ # note: this doesn't need authentication information
$ #       using normal curl operation
$ curl ${DOWNLOAD_URL}
```

The download URL can be accessed from anywhere without the need for authentication or special software.

Delete

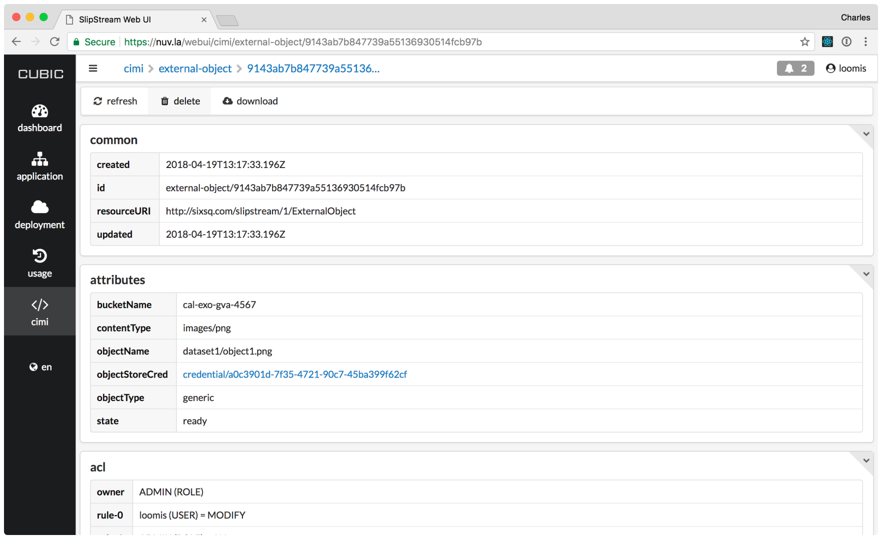
Deleting an ExternalObject uses the standard CIMI delete pattern. This action will also delete the referenced object in S3 storage.

The resource can also be deleted by clicking on the "delete" button on the ExternalObject detail page in the web browser interface. You must confirm this via a dialog before it will actually be deleted.

The `curl` command to do the same thing is:

```
$ # get resource to find operation URLs
$ export LINK=https://nuv.la/api/external-object/0565...
$ ss-curl -XDELETE ${LINK}
$
```

(continues on next page)


SlipStream Web UI
Charles
Secure
https://nuv.la/webui/cimi/external-object/9143ab7b847739a55136930514fcb97b
CUBIC
cimi > external-object > 9143ab7b847739a55136...
2
loomis
dashboard
application
deployment
usage
cimi
en
refresh
delete
download

common

created	2018-04-19T13:17:33.196Z
id	external-object/9143ab7b847739a55136930514fcb97b
resourceURI	http://sixsq.com/slipstream/1/ExternalObject
updated	2018-04-19T13:17:33.196Z

attributes

bucketName	cal-exo-gva-4567
contentType	images/png
objectName	dataset1/object1.png
objectStoreCred	credential/a0c3901d-7f35-4721-90c7-45ba399f62cf
objectType	generic
state	ready

acl

owner	ADMIN (ROLE)
rule-0	loomis (USER) = MODIFY

Fig. 18: Deleting an ExternalObject

(continued from previous page)

```
$ # response: JSON object with 200 status and delete message
$
$ # further 'gets' should fail with 404 status code
$ ss-curl ${LINK}
$
$ # response: JSON object with 404 status
```

As usual you must have an active session to successfully execute the delete action.

ServiceOffer resources that reference that ExternalObject must be kept synchonized manually for any changes to the ExternalObject resources, notably the ACLs and the references to ExternalObject resources.

Metadata

ExternalObject resources support the common CIMI attributes, such as `name`, `description`, and `property`, as well as a `bucketName` and `objectName`. These attributes can be used to provide descriptive metadata for the underlying data objects. The standard CIMI filtering mechanisms can then be used to select ExternalObject resources based on that metadata. These limited attributes are sufficient for many use cases.

Some use cases, however, will require a richer set of metadata. SlipStream already provides a resource that allows for rich metadata to be provided for other managed resources–the ServiceOffer. In the same way that ServiceOffers can describe Virtual Machines offers, they can provide metadata for ExternalObjects as well. In this way, the CIMI filtering capabilities can be used with the richer metadata to provide full-featured search of data sets.

As the schema for the ServiceOffer is completely open, all metadata associated with an ExternalObject, both general and domain-specific, can be tied to the underlying data object. The relationship between these resources is shown below.

Once the ServiceOffer and ExternalObject resources have been registered in the system, users can find data objects of interest by doing the following:

1. Using the standard CIMI search and filtering capabilities to find ServiceOffer resources that describe data objects of interest.
2. Follow the "href" links to ExternalObject resources that provide a proxy for the underlying data object.

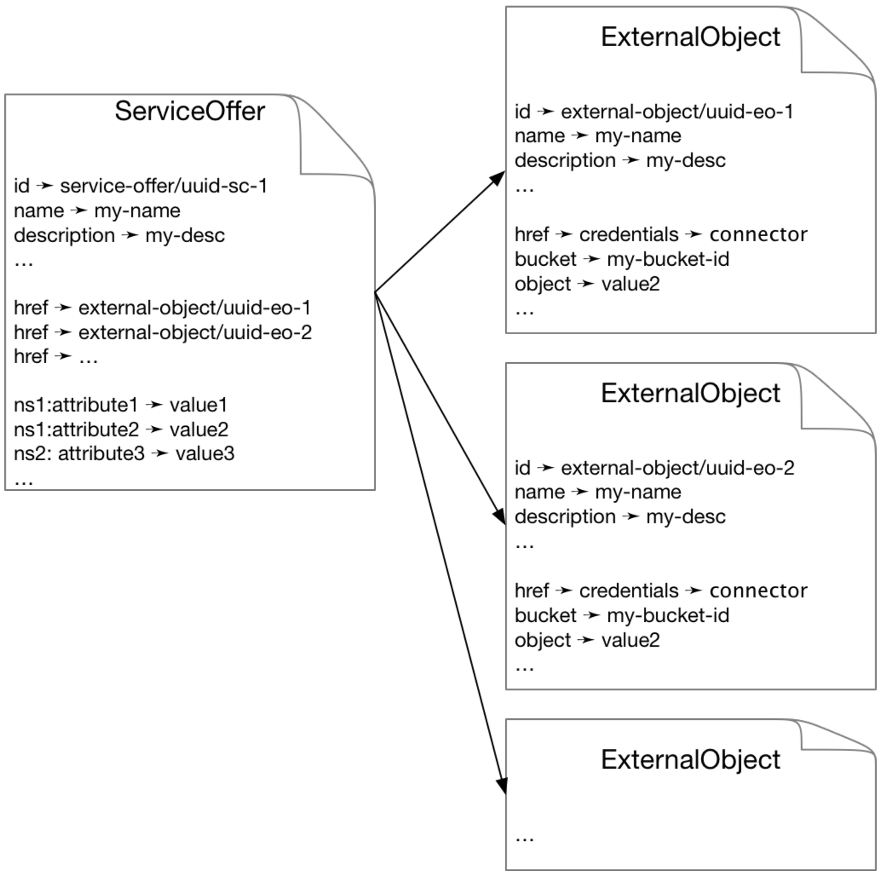


Fig. 19: SlipStream Data Management Resources

3. Request a presigned, read URL from the ExternalObject resources to get direct access to the data via the HTTPS protocol.

This process combines high-level filtering capabilities with efficient, direct access to data.

Gotchas

The data management system involves a couple of CIMI resources and potentially many S3 services. Nonetheless, the model is simple, and managing data objects through SlipStream should be straightforward.

The one critical assumption is that **data objects in SlipStream are managed only by SlipStream**. Direct management of the objects on the S3 storage may create inconsistencies in the SlipStream ExternalObject and ServiceOffer resources.

In particular, data managers should be aware of the following:

1. Do not share buckets that are managed by SlipStream and those that you manage directly. You risk corrupting objects' data or metadata.
2. SlipStream will **not** "autodiscover" any new objects that appear in referenced buckets or update metadata based on changes in S3.
3. Follow the write-once, read-mostly model. Only modify objects through an upload URL provided by SlipStream. Do not modify objects directly via the cloud's object storage API.
4. If you use ServiceOffer resources, ensure that the ACLs for the ServiceOffer and ExternalObject resources are aligned. If they are not, you may leak information to unauthorized people or lose visibility over data objects that you manage.
5. When deleting ExternalObject resources, be sure to propagate changes to ServiceOffer resources that reference them.

In general, these guidelines boil down to: **do not use the S3 API directly for data objects managed by SlipStream** and **keep the ExternalObject and ServiceOffer resources synchronized**.

The synchronization is not difficult in most cases. SlipStream may at some point in the future, provide a higher-level API to simplify this and other aspects of the data management. **Proposals for desirable high-level functions is particularly welcome feedback.**

3.1.7 Roadmap

SixSq produces a new, incremental release of SlipStream approximately every two weeks. Those "candidate" releases are deployed on Nuvla to allow everyone to benefit from bugfixes and new features.

SixSq promotes a release to "stable" about once a quarter. These releases are fully supported by SixSq and recommended for other production deployments.

The release notes for all releases are available on the SlipStream documentation website.

In addition to the continual bugfixing, robustness, and usability improvements, there are several areas where important new functionality is planned:

- Connectors: Continued expansion of the supported cloud infrastructures
- APIs: A gradual shift away from our custom REST API towards the CIMI standard, accompanied by a shift from XML to JSON.
- Storage: Incorporating storage (block and object) as first-class resources within the SlipStream computing model
- Containers: Support for container-based technologies in addition to virtual machine technologies

- Networking: component-level control of firewall rules and dynamic control over network connectivity and quality

Your feedback on what features you'd like to see in SlipStream is important. We continuously adjust our development priorities based on user feedback. Give us your feedback by contacting us at support@sixsq.com.

3.1.8 Appendix

Remote Machine Access

If the deployment of an application works correctly, then you can just use the public interfaces for the service (e.g. a web browser interface) without having to directly access the deployed virtual machines. Unfortunately, especially when developing new application deployments, failures do happen. In these cases **it is extremely useful to be able to log into a remote virtual machine for debugging**.

Access via SSH (secure shell) is nearly universally supported for Unix-like systems. Although Windows can be configured to support SSH, this is rarely done. Instead use of the "Remote Desktop Protocol" is more typical for Windows machines.

Secure Shell (SSH) Access

Providing a public SSH (secure shell) key to SlipStream will allow you to log into the virtual machine instances you create through SlipStream. Although this is optional, it is **strongly recommended that you provide such a key**. It makes debugging cloud and application failures much easier.

To use SSH you must have:

- A valid SSH key pair (public and private key) and
- An SSH client installed on your machine (laptop).

The procedure for meeting these requirements depends on the operating system you're using on your laptop.

Creating SSH Key Pair

Linux and Mac OS X

For Linux and Mac OS X machines (and in general Unix-like environments), all of the software to run an SSH client and to generate an SSH key pair is normally pre-installed.

From the command line on your machine, you can generate a new key pair with the following command:

```
$ ssh-keygen
```

This will then prompt for information. The default location is usually fine. You can use a private key with a password, but this often requires typing the password repeatedly. It is often more convenient to create the **SSH private key without a password**; provide nothing for the password value when prompted to do this.

The public key you will need to give to SlipStream will then usually be in the file `~/.ssh/id_rsa.pub`. (The exact path will be indicated in the prompts from the `ssh-keygen` command.) See below for how to configure your user profile with this information.

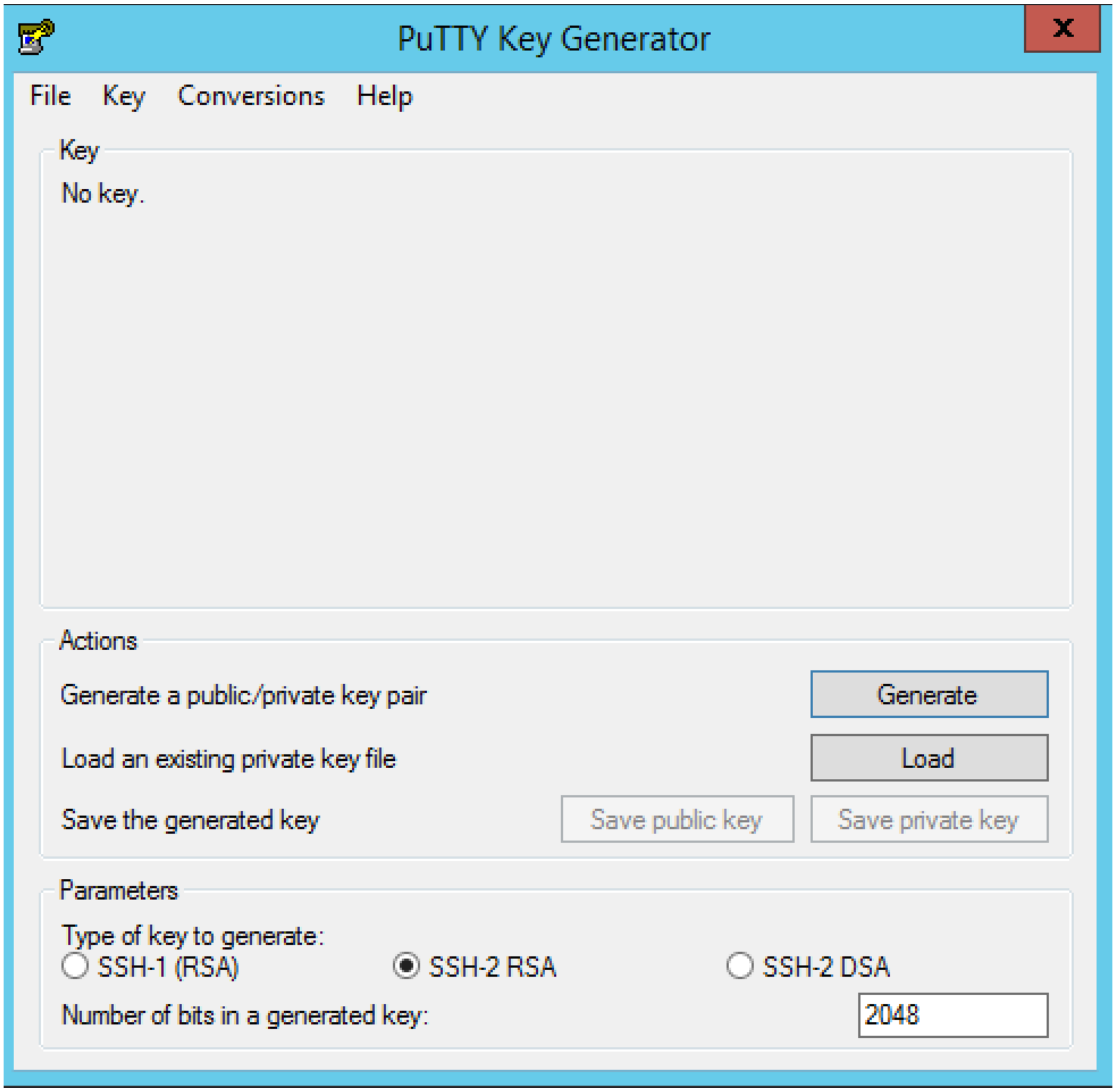
Windows

The standard SSH client to use on Windows is PuTTY. You will need both the `putty.exe` and `puttygen.exe` executables. Download these executables and put them in a convenient location.

Key Pair

The SSH protocol requires a pair of cryptographic keys; one private and the other public. You will need to generate a key pair to use with SlipStream and the underlying cloud infrastructures.

Launch PuTTyGen. You should see a screenshot like the following.



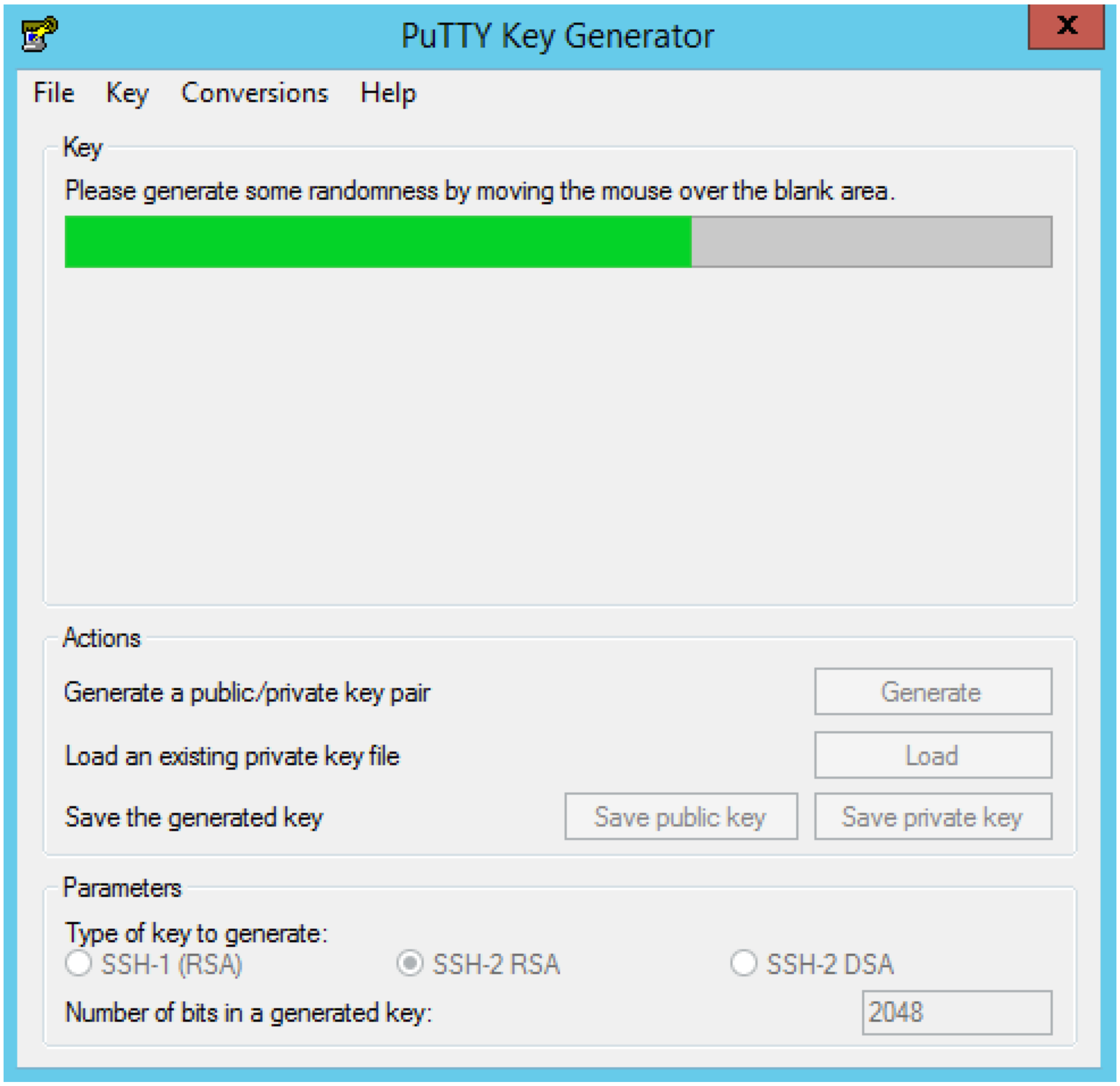
Key
No key.

Actions
Generate a public/private key pair
Load an existing private key file
Save the generated key

Parameters
Type of key to generate:
SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA
Number of bits in a generated key: 2048

Fig. 20: PuTTyGen Start Page

Launch PuTTyGen and then click on the "Generate" button. Move your mouse around in the empty rectangle to generate some randomness for the key generation process.



Please generate some randomness by moving the mouse over the blank area.

Generate a public/private key pair

Load an existing private key file

Save the generated key

Type of key to generate:
SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA

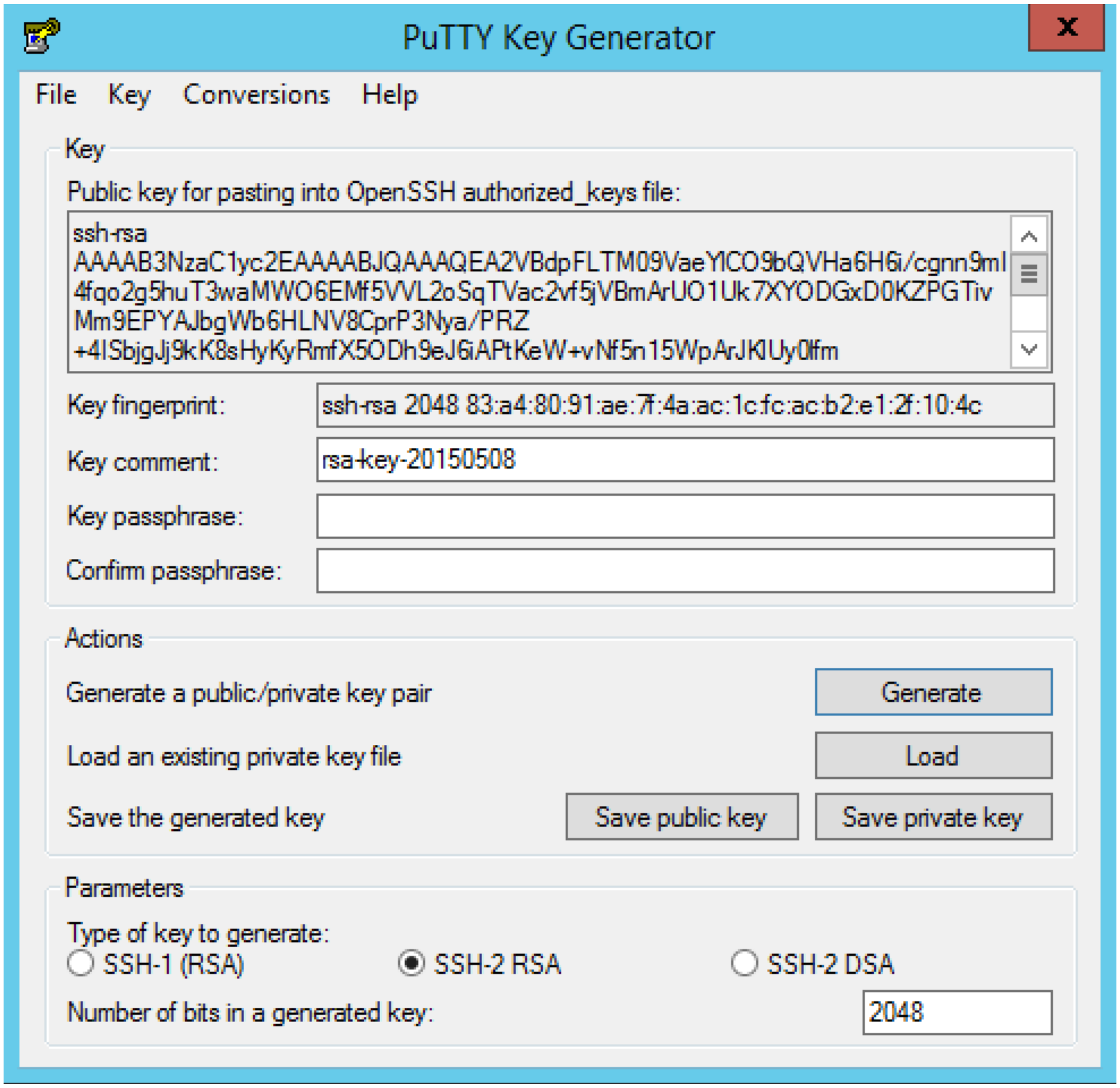
Number of bits in a generated key: 2048

Fig. 21: PuTTyGen Key Generation

When the process is finished. You will see a window like the following.

The value you want to copy and paste into the SlipStream SSH key configuration is given in the box entitled "Public key for pasting into OpenSSH authorized_keys file". Ensure that you copy the **entire value** and that the pasted value is **entirely on one line**.

You must save the public and private keys to use when you want to connect to a virtual machine. Click on the "Save public key" and "Save private key" buttons to do so. You can answer "Yes" when asked whether to save the private key without a password.



Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEA2VBdpFLTM09VaeYlCO9bQVHa6H6i/cgnn9ml
4fqo2g5huT3waMWO6EMf5VVL2oSqTVac2vf5jVBmArUO1Uk7XYODGxD0KZPGTiv
Mm9EPYAJbgWb6HLNV8CprP3Nya/PRZ
+4ISbjgJj9kK8sHyKyRmfX5ODh9eJ6iAPtKeW+vNf5n15WpArJKlUy0lfm
```

Key fingerprint:	ssh-rsa 2048 83:a4:80:91:ae:7f:4a:ac:1c:fc:ac:b2:e1:2f:10:4c
Key comment:	rsa-key-20150508
Key passphrase:	
Confirm passphrase:	

Actions

Generate a public/private key pair

Load an existing private key file

Save the generated key

Parameters

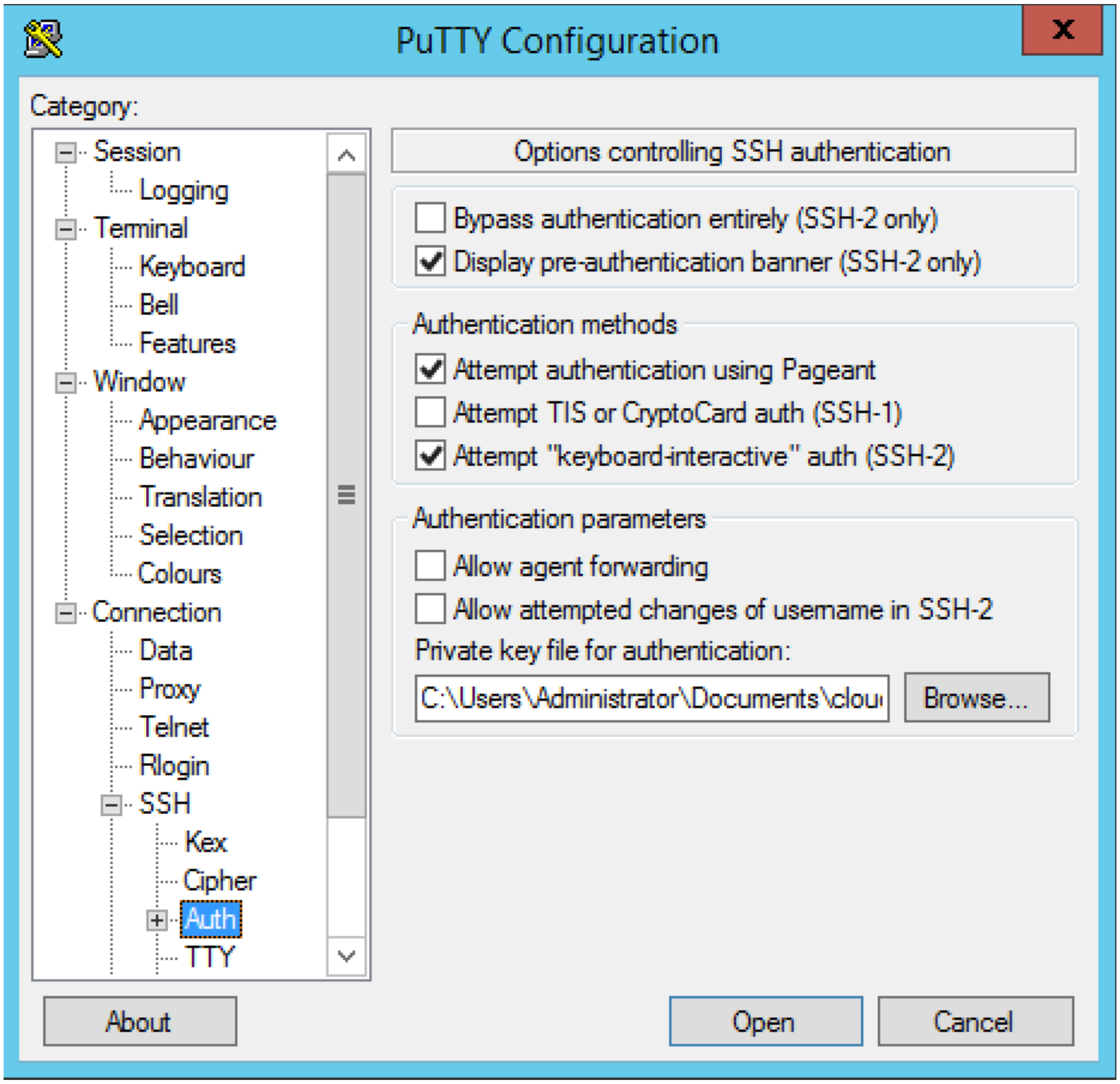
Type of key to generate:
SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA

Number of bits in a generated key: 2048

Fig. 22: PuTTyGen Key

Using the Key Pair

To access a remote machine running the SSH daemon, you will need to use the key pair that you've generated. Start `putty.exe` and then navigate in the tree on the left to "SSH -> Auth". You can then add the private key to use by clicking on the "Browse" button and selecting the private key that you saved earlier.



Options controlling SSH authentication

- [ ] Bypass authentication entirely (SSH-2 only)
- [✓] Display pre-authentication banner (SSH-2 only)

Authentication methods

- [✓] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [✓] Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

- [ ] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication:
C:\Users\Administrator\Documents\clou

Fig. 23: PuTTyGen Key

You may want to download and configure `pagent.exe` with your private key to avoid having to specify the private key for each session. See the PuTTy documentation for how to do this.

Adding SSH Key to User Profile

To allow SlipStream to configure your virtual machines for remote access, you must provide your SSH public key in your user profile:

1. Open your user profile and click on the "Edit" button.
2. Open the "General" section by clicking on the section header.
3. In the "SSH Public Key(s)" field, provide the contents of your SSH public key file (usually `~/.ssh/id_rsa.pub` on Linux, or on Windows the file in which you saved the public key after generating it with PuTTY).
4. Save your profile.

Changes to your SSH key will only affect virtual machines started after the change has been saved; it will not affect virtual machines that are already running.

Warning: Your SSH public key must be on a single line in the OpenSSH format. If you have more than one key, then each must appear on a separate line.

Remote Desktop Protocol

Windows machines rarely support SSH for remote access. Instead Windows uses its own protocol (Remote Desktop Protocol) to provide remote graphical access to a Windows machine. You must have a "Remote Desktop Connection" client installed on your laptop to access Windows machines.

Mac OS X

The "Remote Desktop Connection" application by Microsoft is available in the Mac App Store. Simply search for the application in the App Store and install it.

Linux

Several "Remote Desktop Protocol" clients exist for Linux operating systems. Use the package installer of your operating system to search and to install one of these clients.

Windows

The "Remote Desktop Connection" functionality is integrated into all recent (and not so recent) releases of Windows. The client should already be available on your machine.

VNC (Virtual Network Computing)

The VNC software allows the graphical console of a Linux machine to be used across the network from another machine; this is essentially the equivalent to the "Remote Desktop Protocol" for Windows.

If the VNC software has been installed and configured on the virtual machine, a VNC client can be used to connect to it. A client can be easily downloaded and installed for all operating systems. Use your package manager (or Google) to find and to install a VNC client.

Important: Many machines do not have VNC installed. Before trying to connect with a VNC client, verify that the machine supports VNC and that the necessary ports (5900-5902) are open in the cloud's firewall.

Exercises

1. Ensure you have an SSH client installed on your machine
2. Ensure you have an SSH key pair, generating one if necessary
3. Configure your user profile with your SSH public key

3.2 NuvlaBox Tutorial

This hands-on tutorial demonstrates the core features of NuvlaBox, a **simple and secure private cloud appliance**. It is ideal for **edge computing**, **IoT**, **Smart City**, and **Smart Grid** projects, allowing for **remote management** and application updates. It can also be easily incorporated into **DevOps** processes to speed development and validation of such projects.

3.2.1 Module I - Overview of NuvlaBox (Basic)

The following sections provide an overview of the NuvlaBox appliance. They provide a high-level view of the NuvlaBox features and how different people can benefit from them.

What is a NuvlaBox?

NuvlaBox is a simple and secure private cloud solution that offers easy and affordable access to the benefits of cloud computing. It is a plug & play device with remote management and application update capabilities. It can be easily and securely deployed by a non-technical operator. The NuvlaBox's ability to work both autonomously and as a remote controlled connected device gives great flexibility, supporting a diverse set of deployment scenarios.

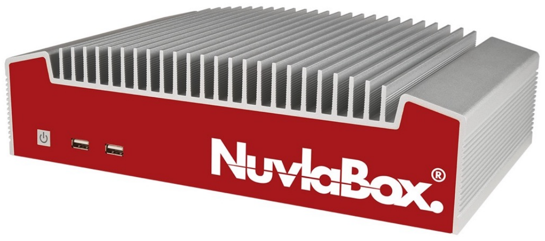


Fig. 24: NuvlaBox Standard v2

Key business benefits

- Built-in **security**. All services are configured for secure operation and all communication is encrypted.
- **Secure web access** to the applications running in your NuvlaBox.
- **Reliable**. All deployments are fully automated to avoid human error. The appliance will restart all applications in case of an unexpected power outage.
- **Plug & Play** device. The NuvlaBox is pre-configured; the user simply has to plug it in to have an operational machine.
- Central control point. Using a **user-friendly web interface** the NuvlaBox can be controlled via a simple dashboard.

- **Remote updates** mean reduced costs. No need to send an engineer or expert on site for software updates.
- Supports **hybrid cloud** deployment scenarios. Workloads can be easily moved between private and public clouds, providing flexibility and potential cost savings.
- **Reliable local connectivity**. Once powered up the NuvlaBox creates its own local WiFi network, which can be used by any device.
- **Flexible WAN connectivity**. The NuvlaBox can be connected to the wide area network using many popular connectivity technologies, including Ethernet, mobile, WiFi, and satellite.

Usage Scenarios

The NuvlaBox platform has been designed to bring the flexibility and ease-of-use of cloud infrastructures to Internet-of-Things (IoT) and Edge Computing platforms. Incorporating NuvlaBox machines into these platforms improves their reliability, performance, and security. A wide range of scenarios can benefit from NuvlaBox systems.

Smart Grid

All industrial control systems acquire equipment monitoring and performance data. They use this data to control operations and to ensure that the industrial plant is working correctly. Unfortunately, these platforms are static, limited by the hardcoded algorithms embedded into the system. They cannot quickly react and adapt to changes within the industrial environment.

Through the SCISSOR project, co-funded by the European Commission, SixSq has demonstrated how cloud technologies including the NuvlaBox can provide a SCADA platform that can react rapidly to changes and evolves as the knowledge of the industrial platform improves. The platform benefits both from the remote management possible from SlipStream and the NuvlaBox, while allowing for autonomous operation in cases where remote locations are temporarily isolated from the rest of the platform.

Smart Cities

Communities can markedly improve the life of their citizens and streamline their operations by taking advantage of the vast amount of data provided by public transport systems, traffic monitoring, environmental sensors, and the like.

NuvlaBox deployments can improve these “Smart City” platforms by:

- Avoiding large data flows over low-bandwidth connections, though local analysis of data.
- Protecting data with privacy concerning by limited the diffusion of such data to a local area.
- Remove silos of information allowing the platforms to increase performance by understanding the correlations between different data sources.

SixSq, through the CityZen initiative , has shown how lighting systems can adapt to changing traffic conditions to increase safety and energy efficiency.

DevOps

Cloud technologies can drastically speed up DevOps processes. Cloud infrastructures allow new resources to be created on demand and to be available within minutes. Deploying test and production services onto those infrastructures can allow quick testing of full systems and the quick, incremental upgrades.

The NuvlaBox can fulfill this role for applications of a limited size or for those applications that require access to real sensors or controllers. Given the small footprint of the NuvlaBox and its ability to run autonomously, development and operations personnel can build confidence in an (updated) application before deploying it to a public cloud.

How does it work?

The customer can decide which applications should be pre-installed in the NuvlaBox. The boxes are then shipped directly to the customer's location. Once on site, simply connect the NuvlaBox(es) to a power supply and a network device. The NuvlaBoxes can then be locally or remotely operated. When connected to the network, new applications can also be downloaded and installed on any box, remotely, at the click of a button.

Hardware Specifications

Supply chain

SixSq choose Logic Supply to design, assemble, and initialize the NuvlaBox hardware. Each NuvlaBox is individualized and each one in the current production series is named after a prize nobel recipient. Each NuvlaBox comes with a prospectus that provides all the unique credentials to connect to it.

NuvlaBox Standard v2

Up to 8 Virtual Machines, sharing 14 GB of RAM and 215 GB disk space.

Description	
Hardware Line	Industrial Fanless Intel Haswell Computer
Processor	Intel Core i5-4570TE Haswell / 2 Cores - 4 Threads / 2.7 GHz
Memory	2 x 8 GB - DDR3 1600 SO-DIMM Memory
Primary Storage	256 GB - Transcend 370 2.5" SSD
WiFi/Bluetooth	Intel 7260 Dual Band 802.11ac/n/g/a/b
Graphics/GPU	Intel 4th Generation HD Graphics
Rear I/O ports	4 x USB 3.0 / 2x USB 2.0 / 2 x Gb LAN
Rear I/O ports	1 x HDMI / 1 x DisplayPort / 1 DVI-I / 2 x Audio jacks / 1 x DC-Jack
Rear I/O ports	3 x RS-232/422/485 COM / 3 x RS-232 COM
Front I/O ports	2 x USB 2.0
Dimensions (WxHxD)	187 x 79 x 290mm
Operating Temperature Range	0°C ~ 50°C
LAN Controller	Intel 82583V PCIe GbE
Expected Life Cycle	5 Years
Regulatory Information	CE standards / FCC / RoHS
Mounting Options	VESA / DIN / Wall
Warranty	2 year limited warranty on parts and services

NuvlaBox Mini v2

Up to 8 Virtual Machines, sharing 6 GB of RAM and 98 GB of disk space.

Description	
Hardware Line	Industrial Fanless Intel Bay NUC Computer
Processor	Intel Celeron N2930 / 4 Cores - 4 Threads / 1.83 GHz
Memory	8 GB - Transcend SO-DIMM DDR3L Low Voltage 1600 Memory
Primary Storage	128 GB - Transcend 370 mSATA SSD
WiFi/Bluetooth	Intel 7260 Dual Band 802.11ac/n/g/a/b
Graphics/GPU	Intel HD Graphics
Rear I/O ports	2 USB 2.0 / 2 Gb LAN / 2 HDMI / 1 Line-out / 1 DC-Jack
Front I/O ports	1 USB 2.0 / 1 USB 3.0 / 1 RS-232/422/485 COM
Dimensions (WxHxD)	142 x 62 x 107 mm
Operating Temperature Range	0°C ~ 50°C
LAN Controller	Intel 82583V PCIe GbE
Expected Life Cycle	4 Years
Regulatory Information	CE standards / FCC / RoHS
Mounting Options	VESA / DIN / Wall
Warranty	2 year limited warranty on parts and services

Note: Contact SixSq if you are interested in installing NuvlaBox Firmware on your own hardware. This is often the case if you need to use precertified hardware or need larger resources than those provided by the standard hardware.

Reliability & Security

The NuvlaBox software has been designed to make the machines as reliable and secure as possible, while keeping the flexibility and remote management features of cloud technologies.

- Security
  - All communications with the NuvlaBox are encrypted
  - Only needed ports are exposed over the local and wide area networks
  - Default credentials are unique for each NuvlaBox
- Privacy
  - The NuvlaBox lets you keep your data within your network, making maintaining privacy and confidentiality easier
- Connectivity

  - The NuvlaBox offers many ways to connect to the local or wide area network
  - Multiple layers try hard to maintain network connectivity once established
  - When the NuvlaBox is connected to the Internet, it tries to connect to a remote SlipStream instance to allow for remote operation
- Resiliency
  - In the case of a power outage, the NuvlaBox will restore existing VMs when power is restored
  - To ensure consistency and to avoid system performance deterioration, all unnecessary system data are erased at each reboot; VM data and user configurations are maintained across reboots
  - The firmware persists only some configuration files, logs, and services databases. The possibility of system corruption after a power cut is extremely small.
  - A watchdog is configured to restart the system in case of kernel crash, ensuring high availability of the system

These sections do not contain any exercises and can be read without needing access to a NuvlaBox.

Once understand the key concepts behind the NuvlaBox, follow *Module II*, which will show you how to use it.

3.2.2 Module II - Using NuvlaBox (Basic)

NuvlaBox a cloud-in-a-box that's child play to use. This module lists the prerequisites for working with NuvlaBox and then guides you from unpacking your NuvlaBox to launching your first application deployment on it via SlipStream.

This module presumes familiarity with cloud technologies, NuvlaBox concepts, and with the key concepts of Slip-Stream. Review *Module I* and *SlipStream Module I* if this is not the case.

Using the NuvlaBox

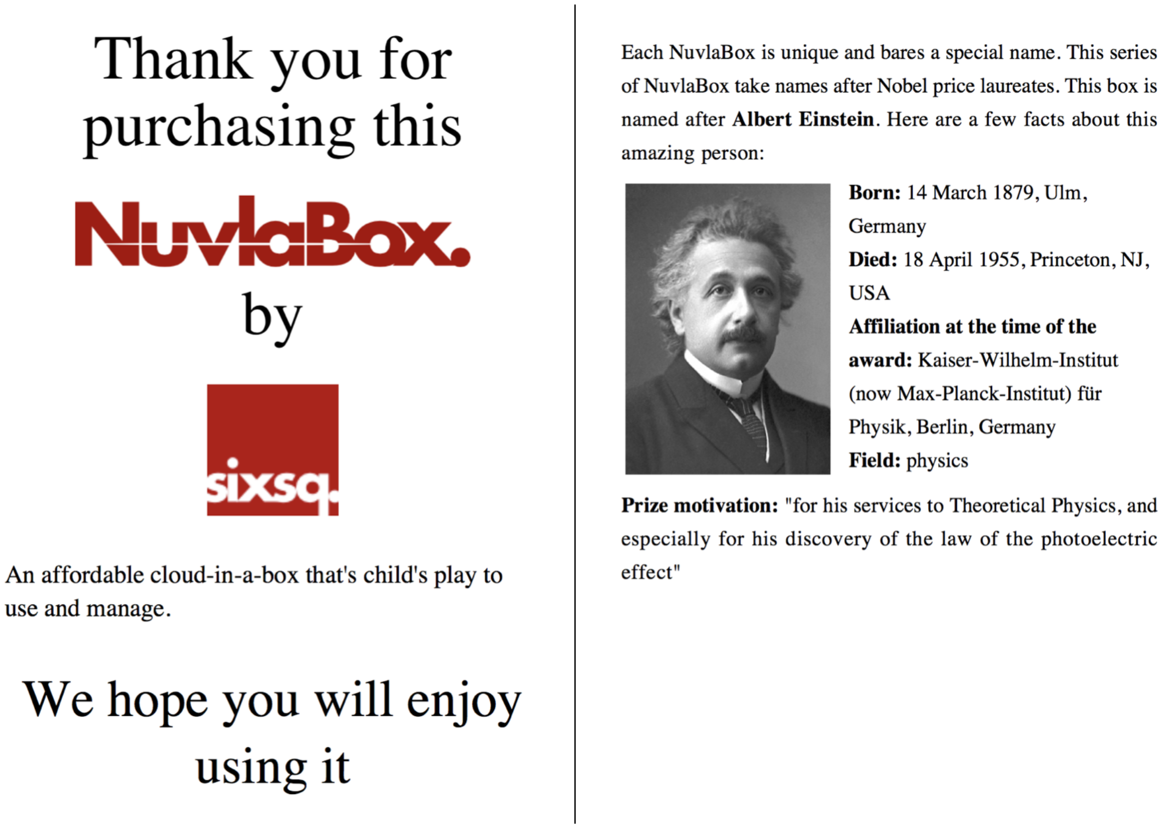
Here are the first things to do with the NuvlaBox:

1. Remove the packaging
2. Attach the WiFi antennas
3. Connect the NuvlaBox's power cord to the mains and to the NuvlaBox
4. Push the power button
5. Keep credentials prospectus in a safe place

Using another device such as laptop, smartphone or tablet (not included) connect to the NuvlaBox WiFi or LAN network. This connection give you access to a local SlipStream and allows you later on to access your running virtual machines.

Note: Please follow *Module II* of the SlipStream tutorial to learn the necessary basis about SlipStream.

You can treat the NuvlaBox as your private *mini* IaaS cloud. As with any IaaS Cloud you want a friendly SaaS or PaaS service to help you with building and deploying applications on IaaS. This is where SlipStream comes into the picture. SlipStream can be used to manage your applications on your NuvlaBox(es) just like you can do for other cloud infrastructures.



Thank you for purchasing this

NuvlaBox.

by

sixsq.

An affordable cloud-in-a-box that's child's play to use and manage.

We hope you will enjoy using it

Each NuvlaBox is unique and bares a special name. This series of NuvlaBox take names after Nobel price laureates. This box is named after **Albert Einstein**. Here are a few facts about this amazing person:

Born: 14 March 1879, Ulm, Germany
Died: 18 April 1955, Princeton, NJ, USA
Affiliation at the time of the award: Kaiser-Wilhelm-Institut (now Max-Planck-Institut) für Physik, Berlin, Germany
Field: physics

Prize motivation: "for his services to Theoretical Physics, and especially for his discovery of the law of the photoelectric effect"

Fig. 25: Example of credentials prospectus for the NuvlaBox named after Albert Einstein

Hint: If no DHCP is available in your network, you can connect to your LAN, access NuvlaBox admin UI and set a static IP for the NuvlaBox WAN interface.

Deploying Applications

The preferred way to manage applications on NuvlaBox is through SlipStream. There is no difference between deploying applications on clouds or the NuvlaBox when using SlipStream. Please follow *SlipStream Module III* of the SlipStream tutorial to learn how to create, deploy and manage your applications on a NuvlaBox.

Using the NuvlaBox from a Remote SlipStream

A NuvlaBox is able to connect to a remote SlipStream via SSH tunneling. When an Internet access is available, the NuvlaBox tries to contact the remote SlipStream and register with it. You can check if your NuvlaBox is connected in the remote SlipStream server by checking the status of the NuvlaBox in the corresponding gauges in the dashboard.

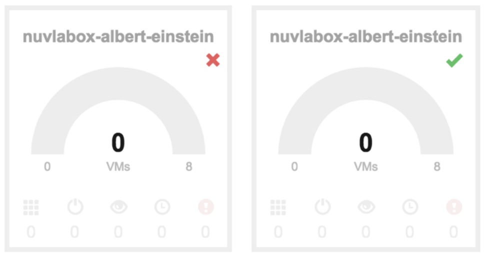


Fig. 26: Gauges of disconnected and connected NuvlaBox machines

By default, NuvlaBoxes are pre-configured to connect to a remote SlipStream named Nuvla, which is a managed service run by SixSq. Connect the NuvlaBox's WAN port to a network with an Internet access and where a DHCP/DNS services are available.

Then connect to Nuvla with following URL: https://nuv.la.

To be able to manage applications on NuvlaBoxes from SlipStream one has to configure credentials of the corresponding Nuvlabox in the SlipStream's user profile. Please follow "Accounts" section of the *SlipStream Prerequisites* to achieve this.

From the remote SlipStream, remote tunnel ports are opened that allow the remote SlipStream to access NuvlaBox endpoints for SlipStream, OpenNebula, and SSH.

Note: Contact SixSq if you are interested in installing your own on premise SlipStream service instead of using Nuvla.

Hint: If you are connected to the NuvlaBox network, you can connect to your VM by using default SSH `port 22` and by using the `displayed IP` in your deployment on SlipStream.

Using the NuvlaBox from the Local SlipStream

Using another device such as laptop, smartphone or tablet (not included) connect to the NuvlaBox WiFi or LAN network.

Then connect to the URL: https://nuvlabox. This URL corresponds to the SlipStream service.

Choose `local` in NuvlaBox welcome page and login into SlipStream with the "nuvlabox" user account.

Hint: If you are connected to the NuvlaBox network, you can connect to your VM by using default SSH `port 22` and by using the `displayed IP` in your deployment on SlipStream **only if you've configured the nuvlabox account with your public SSH key**.

In this module you've acquired enough knowledge about NuvlaBox. After working through this module, you will be ready to proceed to *Module III*, where you will learn about the NuvlaBox architecture.

3.2.3 Module III - NuvlaBox configuration (Advanced)

In this module you will learn a little more about the NuvlaBox software stack, networking configuration, and how to attach a device to the NuvlaBox.

This module presumes familiarity with NuvlaBox basic usage. Review *Module I* and *Module II* if this is not the case.

Manageability

Configuration of the NuvlaBox

An administration UI is available and accessible from the local SlipStream. To access it, login with "super" account in local SlipStream, click on **Configuration** in the menubar and choose **NuvlaBox**.

Available actions in NuvlaBox UI admin:

- System activity (CPU usage, RAM usage, partition usage)
- USB status. This is the listing of plugged USB devices in the NuvlaBox.
- Remote SSH SlipStream tunnel settings
- Network configuration (WAN, LAN, WLAN)
- System actions (restart, poweroff or Factory reset)

All this actions are also available from the NuvlaBox console by using a CLI tool named `nuvlabox-admin`.

Updating a NuvlaBox

Using a USB key with last version, or by using a migration script. An update through the admin UI should be possible check the roadmap.

Installation from Scratch

There is 2 different way to install NuvlaBox:

- Flash directly NuvlaBox firmware through USB key (Hardware specific)

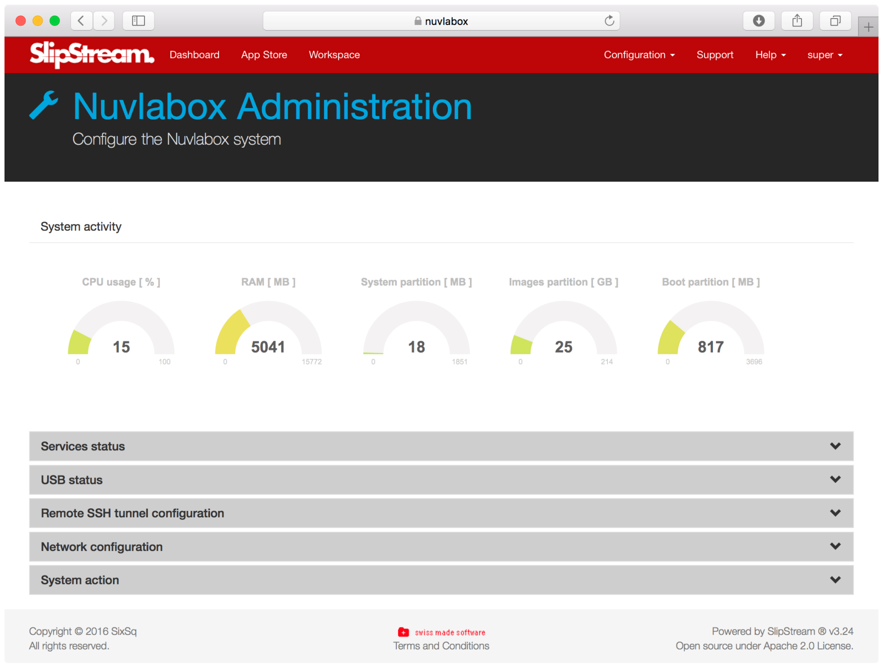


Fig. 27: NuvlaBox Administrator UI

- Install scripts (Hardware generic)

Flash NuvlaBox Firmware

If you have a NuvlaBox USB flash key for your specific NuvlaBox, you need a screen and a keyboard to be able to flash your NuvlaBox.

Hint: There is a contextualization file in the USB key, which contain specific credentials for your box (private key, WiFi password, etc.).

Here is the procedure to boot on a USB key:

- Start the NuvlaBox
- Enter bios boot device menu selection. You have to quickly hit following key until the menu appear:
  - Hit <F11> key for NuvlaBox Standard v2
  - Hit <F7> key for NuvlaBox Mini v2
- Choose USB key in the list to boot from it
- Follow instruction on screen

The flash of the firmware should take about 10 minutes.

NuvlaBox will reboot 3 times after the end of installation:

- 1st reboot - End of firmware installation
- 2nd reboot - Configuration of the system in concordance with your hardware
  - Notice the update of the CLI prompt => nuvlabox

- 3rd reboot - Contextualization of your NuvlaBox with appropriate credentials
  - Notice the update of the CLI prompt => nuvlabox-<nobel-prize-name>

Your NuvlaBox is now ready to be used. Remove the flash USB key.

Install Scripts

Contact SixSq if you are interested in installing NuvlaBox Firmware on specific hardware. This is often the case when you need a more powerfull NuvlaBox which is able to run more than 8 VMs.

Attaching Sensors & Controllers

To attach a sensor, you have to edit your component in SlipStream and add an *Additional custom textual VM template*. Anything placed in this field is transmited unmodified to the KVM hypervisor.

First of all, we need to add a USB controller, on which we attach the USB device. To do so, for USB 2.0 or USB 3.0 devices, use the following template to attach your device to your virtual machine at startup.

You should only update the `vendor id` value and the `product id` with values corresponding to your device.

```
RAW = [ TYPE = "kvm",
        DATA = "
        <devices>
          <controller type='usb' index='1' model='piix3-uhci'/>
          <hostdev mode='subsystem' type='usb' managed='yes'>
            <source>
              <vendor id='0x046d'/>
              <product id='0x0826'/>
            </source>
            <address type='usb' bus='0' port='1'/>
          </hostdev>
        </devices>"]
```

Hint: In NuvlaBox administrator UI, you can easily get `vendor id` and `product id` of attached devices.

Warning: You have to attach specified devices in *Additional custom textual VM template* to your NuvlaBox, otherwise deployed concerned Virtual machine can't boot.

To attach other host devices, please refer to the libvirt documentation.

Deploying Sensor-based Applications

Here on Nuvla you can find different sensor-based applications to be launched from Nuvla on your NuvlaBox.

Other examples

SixSq is participating in SCISSOR, a H2020 project supported by the European Commission. The aim of this project is to design a new generation SCADA security monitoring framework.

The "edge agents" in the above figure run on NuvlaBox machines.

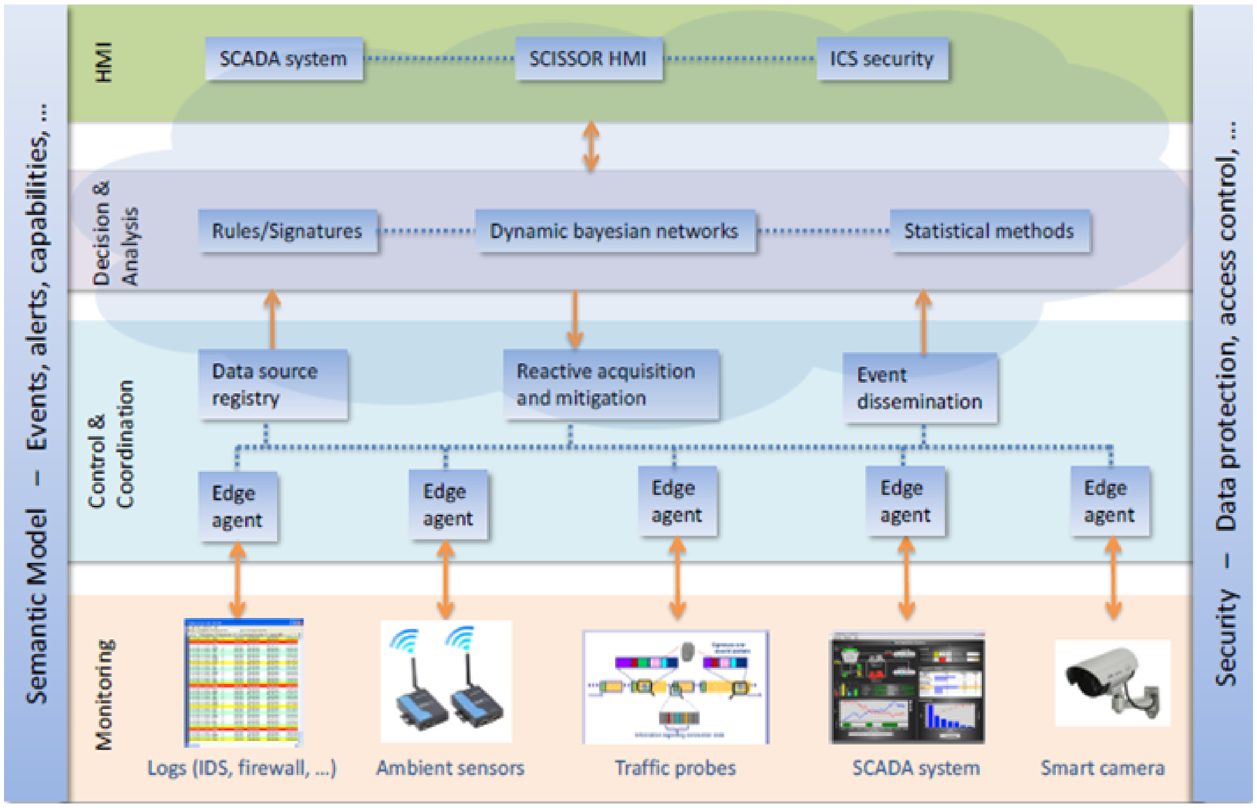


Fig. 28: SCISSOR four-layer SCADA security monitoring framework

Through SixSq's participation in this project, the following types of devices have been demonstrated with the NuvlaBox:

- IP camera
- USB microphone
- Serial communication to capture information from RFID sensors
- Network activity from IDS analysers
- Logging

Software Architecture

Software Stack

The NuvlaBox is built atop a robust Open Source foundation, benefiting from the reliability and diverse features of tested, open software. No other cloud appliance has the level of functionality and integration provided by the NuvlaBox.

The NuvlaBox also takes advantage of numerous system-level services to support the core software stack. Many of them can be configured to support customized NuvlaBox installations.

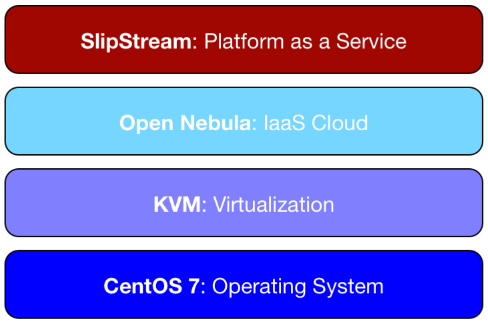


Fig. 29: NuvlaBox Software Stack

Accessible Service Endpoints

Users connected to the WiFi/LAN network of the NuvlaBox can access the following endpoint:

- nuvlabox (172.16.0.1)

If you are not connected to the NuvlaBox on the wide area network, you can connect to those services by using the IP of the NuvlaBox in your network.

Services Names	Listening Ports
SlipStream	443
OpenNebula Sunstone	9870
SSH	22

User Accounts

Service Name	User-name	Password location	Description
SSH	root	In credentials prospectus	NuvlaBox administrator
SlipStream	nuvlabox	In credentials prospectus	NuvlaBox cloud user
SlipStream	super	In credentials prospectus	SlipStream administrator
OpenNebula	ssuser	In credentials prospectus	OpenNebula cloud user configured for NuvlaBox cloud user
OpenNebula	oneadmin	In /var/lib/one/.one/one_auth	OpenNebula cloud administrator

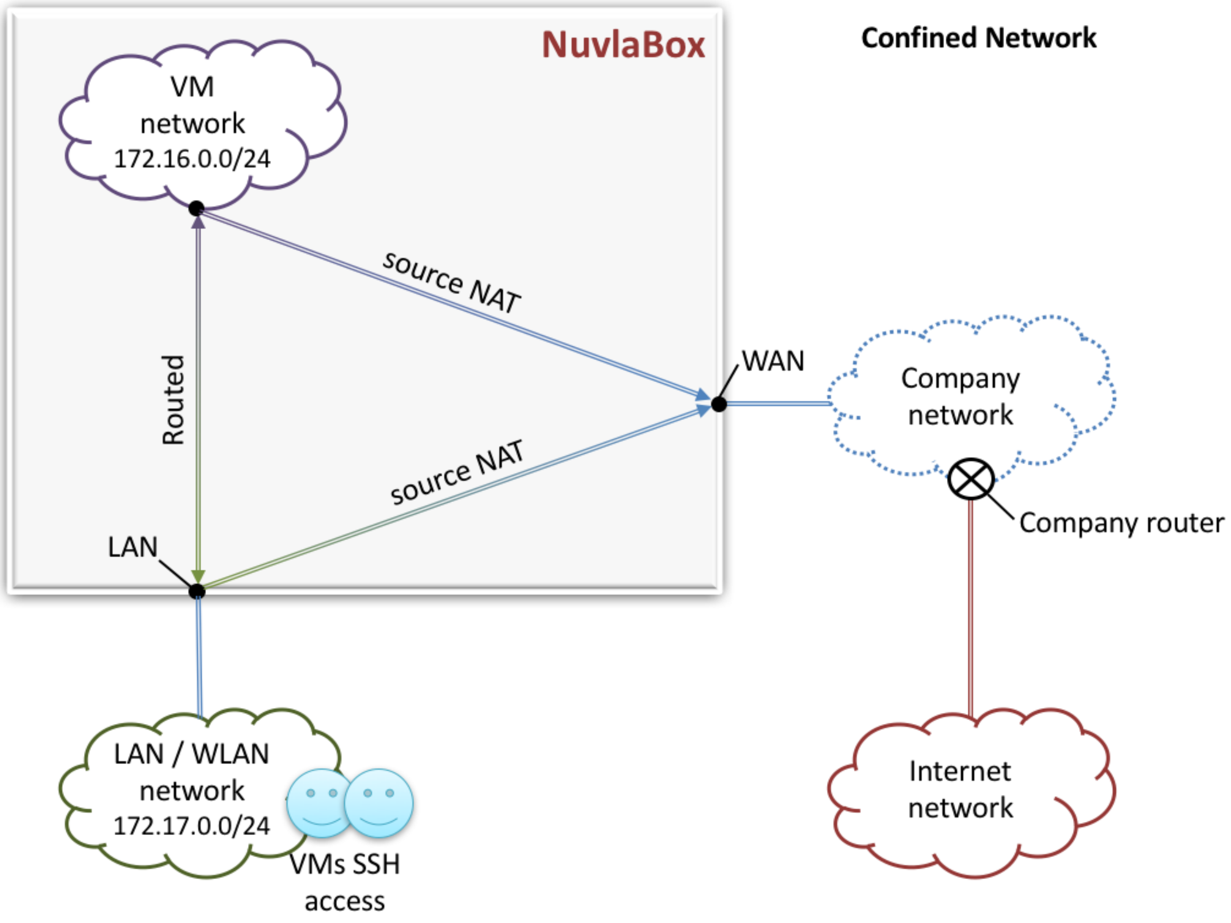
Networking

The NuvlaBox has been designed to be flexible and to support several network scenarios. By default, the NuvlaBox is configured with a specific network scenario which doesn't need any configuration from the operator. The default network scenario used by the NuvlaBox is named *Confined Network*. This as well as other well tested network scenarios are described in detail in the following sections.

Confined Network

In this scenario, all Virtual Machines (VMs) and all users are connected to the network managed by the NuvlaBox. The advantage of this network scenario is that no configuration is needed to access and to use the NuvlaBox. Specifically,

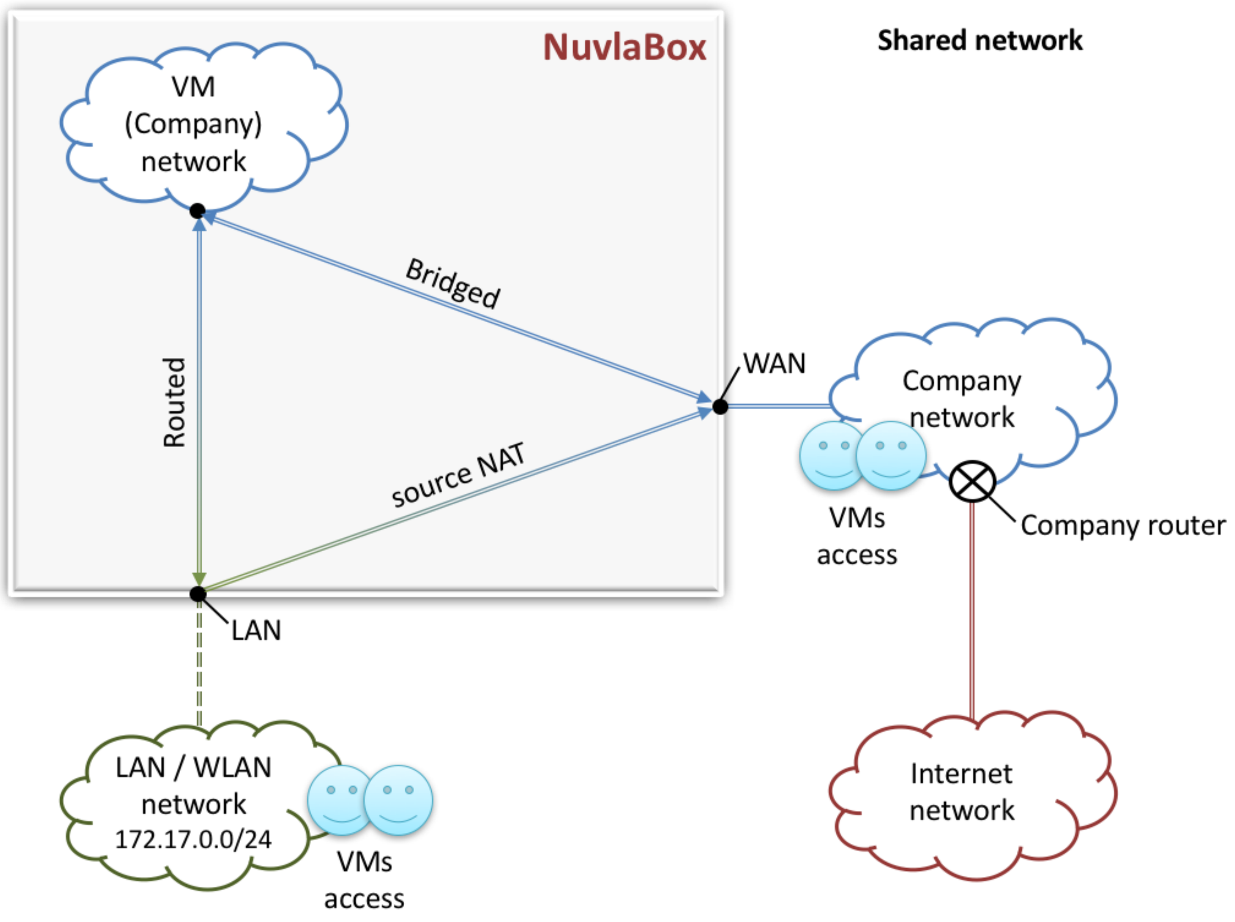
- VMs are created in internal virtual networks inside the NuvlaBox.
- An internal DHCP allocates IP addresses to VMs and to users.
- Connected users in *LAN / WLAN network* and virtual machines in *VM network* have access to the company network and to the Internet through Network Address Translation (NAT).
- VMs are reachable through the *LAN / WLAN network*.



Shared Network

In this scenario, all VMs are reachable from the *Company Network* without being connected directly to the NuvlaBox. The advantage of this network scenario is that services running within VMs on the NuvlaBox can be accessed directly through the *Company Network*. Specifically,

- Virtual machines and users are bridged onto the *Company Network*.
- The company's DHCP server allocates users' IP addresses, but a range of IP addresses on the *Company Network* for VMs is allocatec to and managed by the NuvlaBox itself.

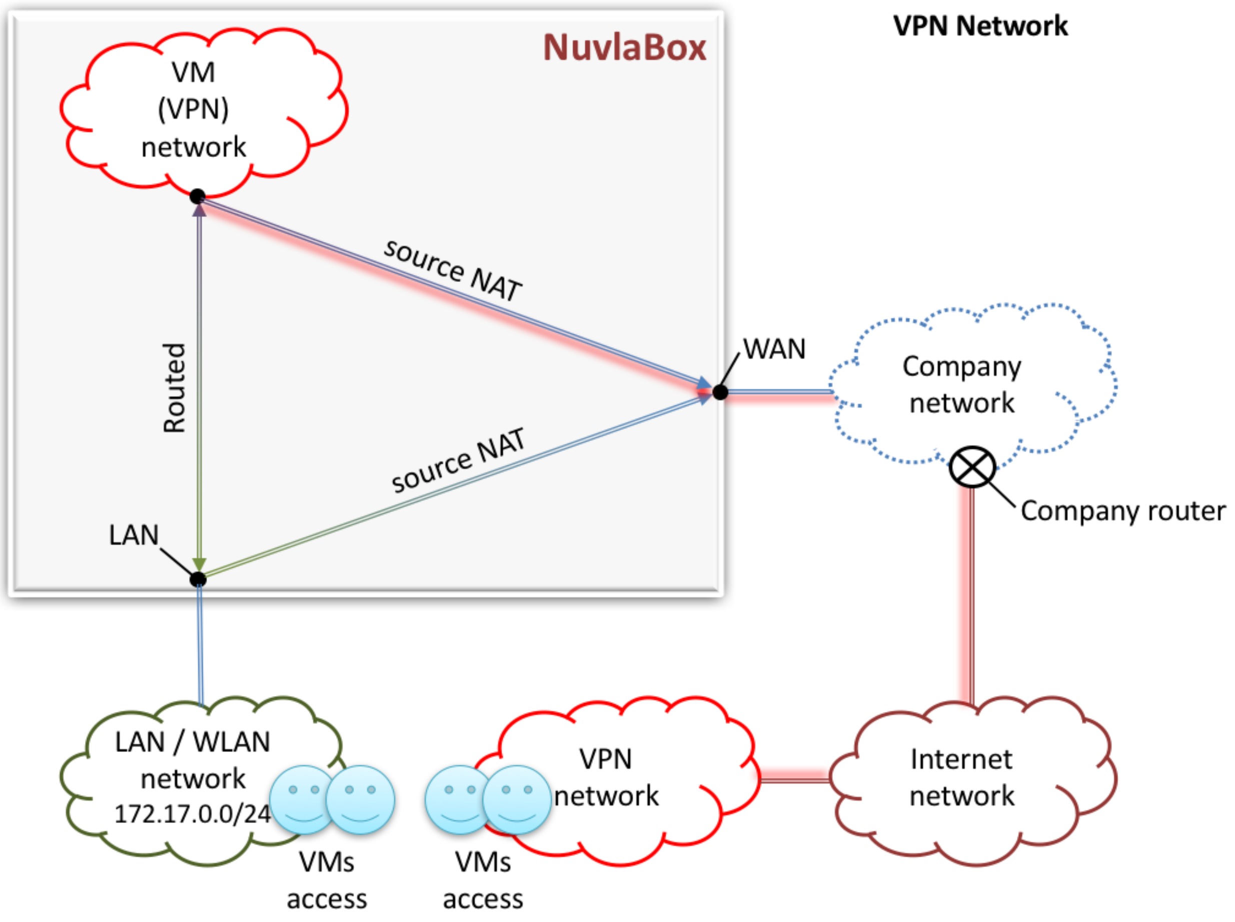


VPN Network

The idea about this network scenario is to have running VMs on a shared VPN network transparently to the VMs. The advantage of this network scenario is to have access to VMs from anywhere from the internet by using a VPN connection.

Specificity:

- Virtual machines are bridged onto the *VPN Network*.
- If a local connection is available with access to the Internet, VMs use this connection to access internet and VPN connection to communicate with the *VPN Network*.



VM level VPN connection

This is not a real network scenario, because it use the *Confined Network* scenario and SlipStream recipe configuration to make the VMs connect to a remote VPN server. The advantage of this network scenario is to have access to VMs from anywhere from the internet without changing the default network scenario of the NuvlaBox.

Note: Contact SixSq if you need a custom network scenario.

After completing this module you should be familiar with the using the NuvlaBox and understanding how it functions. From here, you can view the *Roadmap* of future developments.

3.2.4 Roadmap

SixSq produces a new, incremental release of NuvlaBox approximately every two months.

In addition to the continual bugfixing, robustness, and usability improvements, there are several areas where important new functionalities are planned:

Software improvements:

- Creating a dedicated UI for managing NuvlaBoxes
- Make admin UI visible from the remote SlipStream

- Cleanup of built images from SlipStream
- Container support on the NuvlaBox
- Simplifying the use of sensors and controllers

Your feedback on what features you'd like to see in NuvlaBox is important. We continuously adjust our development priorities based on user feedback. Give us your feedback by contacting us at support@sixsq.com.

3.3 SlipStream and NuvlaBox Training - 2017-04

3.3.1 Introduction (CL)

- *Cloud Technology*
- *What is SlipStream?*
- *What is a NuvlaBox?*

3.3.2 Understanding SlipStream (KS)

Understanding SlipStream by using Nuvla, SixSq's managed SlipStream service.

- *SlipStream Ecosystem*
- *Key concepts*
- *Prerequisites*
  - Nuvla account
  - Cloud accounts: Exoscale (public) and NuvlaBox (private)
  - Workstation configuration
- *SlipStream Web UI Tour*

3.3.3 Understanding the NuvlaBox (KB)

- *Hardware Configuration*
- *Reliability & Security*
- *Using the NuvlaBox*
  - RStudio on NuvlaBox (demo and exercise)
- *NuvlaBox administration UI*
- *Sensors*
  - People recognition (video sensor) from Nuvla (exercise)
- *Software Architecture*
- *Networking*

3.3.4 Building and Running Cloud Applications (KS)

Based on *Module III*. Use Nuvla. Run all deployments on NuvlaBox. For multi-cloud deployment run on NuvlaBox and Exoscale.

- *Workspace* (projects, ACLs, versioning)
- *Images* (native images)
- *Components I* (parameterized and secured web server)
- *Deployment Logs*
- *Components II* (test client for the web server)
- *Applications* (multi-machine deployment, parameters exchange, component multiplicity, multi-cloud deployment (test clients on NuvlaBox and web server on Exoscale))

3.3.5 Demo (LS)

Sensor-based applications on NuvlaBox.

- Car counting camera and application

3.3.6 Automation, Optimization and Debugging (KS)

- *Automating SlipStream*
- *Faster Deployments*
- *Debugging applications*

3.3.7 Scalable Applications (KS)

Theory and demonstration of scalable and autoscalable application in SlipStream. Based on *SlipStream Module IV*.

- *Scalable applications*
- *Auto-scalable applications*
  - Demo of autoscalable application on Exoscale

The tutorials are built of sufficiently autonomous learning modules covering different aspects of the solutions. It is assembled so that it's possible to follow it independently for SlipStream or NuvlaBox while, whenever necessary, bridging the two.

The path through the tutorial is cross referenced, providing the hints and guidelines on the most suitable module/section to follow after completion of a current one.

Important: The command line examples throughout the tutorial use Linux Bash syntax. If you're using a different shell (csh, PowerShell, etc.), you'll need to adjust the syntax accordingly.

CHAPTER 4

Administrators Guide

4.1 Introduction

SlipStream, developed by SixSq, is an affordable, multi-cloud application management platform that supports DevOps, Big Data and Smart City strategies platform.

4.1.1 Main Features

Multi-machine Provisioning System: SlipStream allows users to define and execute deployments, based on high-level recipes that are independent of the cloud on which the deployment will run. SlipStream coordinates the provisioning of virtual machines, synchonizing the deployment and configuration of individual machines as defined in the deployment recipe.

For example, users can deploy entire multi-tier software systems on-demand and in a single action. With SlipStream, this operation takes a few minutes, compared to hours (or days!) for manual installations.

Multi-cloud Provisioning: SlipStream supports multiple cloud providers and multi-cloud deployments, through a plug-in connector design. (See supported connectors below.) Users select the cloud service providers and technologies when deploying virtual machines, from within a single SlipStream service. Furthermore, users can place parts of a deployment on different cloud services or regions, automating redundant and error-resistent deployments.

To illustrate, users can deploy computing clusters, load-balanced applications, and multi-media processing pipelines, for example, across several cloud providers to improve the resilience of the service and to reduce latencies by co-locating data and computing resources, while reducing costs.

Support continuous integration processes with continuous deployment: SlipStream encourages users to parameterize image creation and deployment recipes, such that key parameters (e.g. software version, package location, dependencies, inter-relationships) can be provided at runtime. This means that it is easy to integrate SlipStream with continuous integration servers to provide a full deployment chain.

For instance, users can provision complete server-side systems in a single action in order to test and certify a specific version of the system.

Independence from specific IaaS interfaces and hypervisors: the SlipStream recipes are independent of specific IaaS interfaces, avoiding vendor lock-in and allowing you to focus on configuration and deployment of your application, instead of the specifics of each IaaS.

For example, users can seamlessly apply the same image construction recipes and multi-machine deployments to different cloud providers, yielding the same results every time.

Community sharing platform: the SlipStream data model permits users to share their image and deployment recipes with other users, encouraging collaboration within a community, something dear to agile and DevOps principles.

4.1.2 Supported Clouds

SlipStream communicates to IaaS clouds services via a connector architecture. In general, the connectors for open source IaaS implementations are released under the same license (Apache 2) as the SlipStream core; connectors for proprietary solutions are closed source and available via a commercial license.

The following table shows the production connectors. Alpha and beta connectors for other proprietary clouds may also be available; request information via the SlipStream support.

Connector	Type
Amazon EC2	proprietary
CloudStack	open source
Exoscale	proprietary
OpenNebula	open source
OpenStack	open source
Open Telekom Cloud	proprietary
NuvlaBox	proprietary
VMWare vCloud	proprietary

4.1.3 SlipStream Edition

SlipStream is available in two editions, **Community** and **Enterprise**. Commercial support and proprietary connectors (see above) are only available on the Enterprise edition; otherwise the two editions are identical.

For each edition, we maintain three repositories:

- **Snapshots**: most recent code base but software may be unstable
- **Candidates**: software releases that may be stable but not yet been validated for production
- **Releases**: stable releases verified for production

We create new candidate releases every two weeks and deploy those candidates on our Nuvla service.

4.2 Prerequisities

SlipStream is migrating towards a micro-server architecture and consists of a number of individual services that work together. The front-end services are stateless; all of the persistent state is stored in separate databases (currently Elasticsearch and HSQLDB). Many administrators are familiar with this type of architecture and will be comfortable running the necessary services and databases.

4.2.1 Software Requirements

The software is tested, packaged, and supported for CentOS 7 systems. The packages should install and run correctly on any RHEL 7 compatible system.

The software itself should build and run on any Unix-like system (internally we use MacOS as a development platform), although some changes may be needed for integration with the operating system's service management infrastructure.

4.2.2 Hardware Requirements

The hardware requirements of SlipStream are modest. Any modern, multicore, server-class machine should run the SlipStream services without problems. We recommend a machine with a minimum of 4 CPU cores and 8 GB of RAM. A machine with 20-50 GB of disk space should be sufficient for initial use.

As with any service, the resource requirements for the server will increase with the number of users and with the number of system deployments. Use a larger machine if you expect particularly large scale or intense use of the service. For a production system, you should strongly consider using a separate cluster for the Elasticsearch and Zookeeper services.

4.2.3 Networking Requirements

SlipStream is available through a web-proxy on the standard HTTPS (443) port. The proxy redirects all traffic on the HTTP (80) port to the HTTPS port. Consequently, firewalls on the machine or site must allow access to the HTTPS port from the users' machines; they may also allow access to the HTTP port.

To administer and monitor the machine running the SlipStream service, you may also want to have the SSH port (22) and the Nagios NRPE port (5666) open to your administrators' machines.

For an initial test deployment, the Elasticsearch database is deployed on the same machine as the other servers. In this case, no additional ports need to be opened to the outside world. Production deployments should have the Elasticsearch database deployed on separate machines. In this case, the Elasticsearch ports (9200 and 9300) will need to be opened to the machine(s) running the other SlipStream services.

The configured SlipStream connectors act as clients of the correponding cloud service provider. Consequently, the server must also have *outgoing* access to the underlying cloud service endpoints. The ports used vary depending on the cloud service provider.

For any cloud service provider that does not support a virtual machine contextualization mechanism, the server requires direct SSH access (port 22) to virtual machines within those clouds.

4.2.4 Cloud Requirements

SlipStream is a cloud deployment engine and requires access to at least one supported cloud infrastucture. See the list of supported clouds and conditions on the associated cloud connector to select the cloud(s) you will be making available through your SlipStream instance.

You must have at least one valid account on each cloud you want to support to verify the SlipStream configuration for those clouds.

4.3 Quick Installation

An automated installation script is available to install SlipStream and all of its dependencies on a CentOS 7 machine.

4.3.1 Node Preparation

Deploy the machine that you'll be using for your SlipStream server with the CentOS 7 operating system. (See the prerequisites in the previous section for sizing information. A machine with 4 CPU cores, 8 GB of RAM and 20 GB of disk space is recommended as a minimum.)

The installation script will **not** perform a general upgrade of your CentOS 7 system. You may want to run:

```
$ yum upgrade -y
```

to ensure your system has the latest software and security patches.

4.3.2 Installation Overview

This installation script will run through the following phases:

- **Prepares the node:**
  - Adds YUM repositories (EPEL, Nginx, Elasticstack, SlipStream)
  - Installs global dependencies (unzip, curl, wget, . . .)
  - **Configures the firewall**
    - Allow all outgoing packets
    - Allow loopback connections
    - Allow incoming ICMP requests
    - Allow incoming SSH connections (port 22)
    - Allow incoming HTTP connections (port 80)
    - Allow incoming HTTPS connections (port 443)
    - Deny all other incoming connections
    - Deny forwarding
  - Disables SELinux
- **Installs SlipStream dependencies:**
  - Elasticsearch
  - HSQLDB
  - Graphite
  - Zookeeper
- Installs the SlipStream client
- **Installs the SlipStream servers**
  - Stops running services
  - Installs and configures packages
  - Starts SlipStream services
  - Installs and configures nginx proxy
- **Installs and configures time synchronization service**
  - Required for checking validity of certificates, etc.

- Install SlipStream in /opt/slipstream
- Install HSQLDB in /opt/hsqldb
- Install nginx
- Install some other dependencies with yum and python-pip
- Configure nginx, HSQLDB and SlipStream
- Start nginx, HSQLDB and SlipStream

These instructions assume that you will be using the prebuilt binary packages for SlipStream. If you want to build your own packages from source, refer to the Developer Guide.

4.3.3 Run the Installation Script

Note: If you are installing the **Enterprise Edition** you must install the certificates (`yum-client.crt` and `yum-client.key`) provided by SixSq into `/etc/slipstream/` before starting the SlipStream installation process.

Warning: If you're behind a web proxy, see the *Installation via Web Proxy* section before starting.

This command will take care of the full SlipStream installation process:

```
$ curl -sSfL https://raw.githubusercontent.com/slipstream/SlipStream/master/install/
↪slipstream-install.sh | bash -s {edition} {repokind}
```

Replace the placeholders with the values for your installation. The possible values are provided in the following table.

placeholder	values
`{edition}`	`community` or `enterprise`
`{repokind}`	`snapshot`, `candidate` or `release`

The logs of the installation are available in the file `/tmp/slipstream-install.log` and `/tmp/slipstream-connectors-install.log`. These can be consulted to understand the details of the installation process or to debug problems.

4.3.4 SSL Certificate

The SlipStream installation script will generate a self-signed certificate for the server. If you want to avoid browser warnings about insecure certificate, install a certificate signed by a widely accepted certificate authority.

Place your server certificate (formatted as an X509 certificate and key) in the files `/etc/nginx/ssl/server.crt` and `/etc/nginx/ssl/server.key`. You must restart the server to have SlipStream use the new certificate.

4.3.5 Cloud Connectors

If you have installed Community Edition of SlipStream, the quick install script will have already installed all the open-source connectors.

If you have installed the Enterprise Edition, you have to install all connectors you have a license for.

To install connectors, simply execute:

```
$ curl -sSfL https://raw.githubusercontent.com/slipstream/SlipStream/master/install/
↪ss-install-connectors.sh | bash -s -- -r {repokind} {connector names}
```

placeholder	values
`{connector names}`	space-separated list of connectors
`{repokind}`	`snapshot`, `candidate` or `release`

Once all needed connectors are installed, restart SlipStream:

```
$ systemctl restart slipstream
```

You will then be able to configure the cloud connectors that you have installed.

4.3.6 Testing the Service

You should now be able to contact the SlipStream server with a web browser using HTTPS; the URL should be `https://your_machine/`. You should be redirected to the login page that looks similar to the following screenshot.

If everything looks good, you are ready to configure the server and cloud connectors.

4.3.7 Configuration

To configure your SlipStream server, log into the server as `super` and use the pages found under the "Configuration" menu item. Use the information in the Authentication, Cloud Connectors, and Connector Installation and Configuration sections.

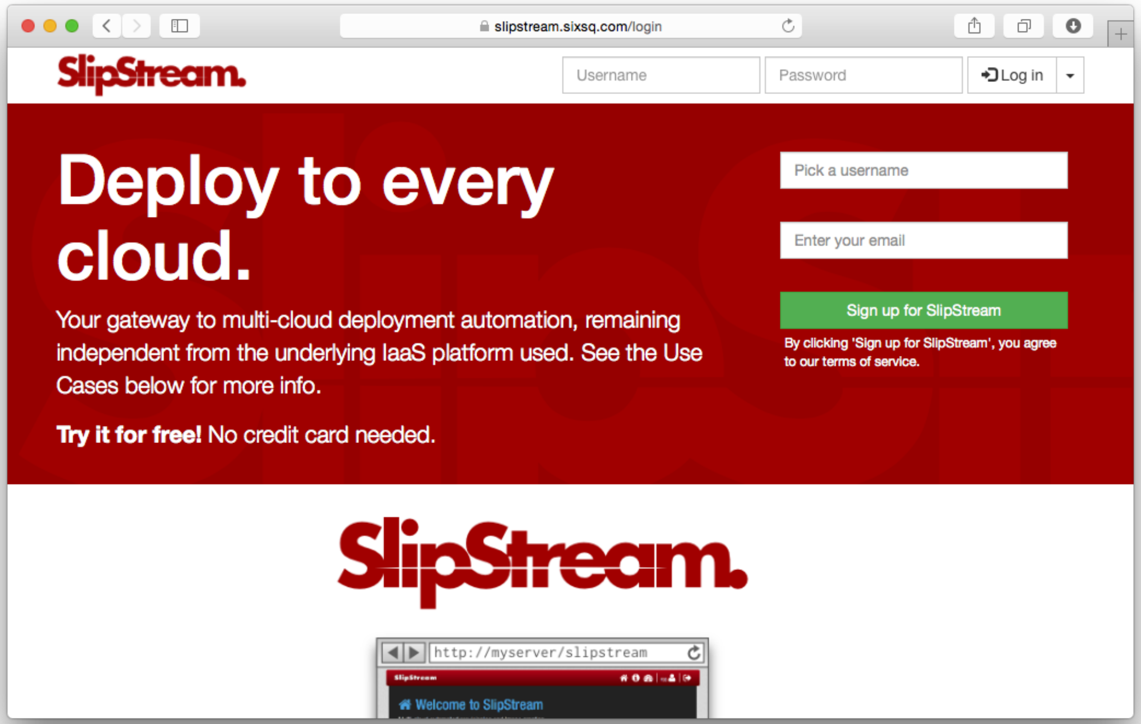
4.3.8 Installation via Web Proxy

If you are installing SlipStream from behind a web proxy, you'll need some additional configuration to ensure that the scripts and packages can be downloaded through the proxy. Set the following environmental variables:

```
http_proxy=http://user:password@ip_addr:3128/
HTTPS_PROXY=https://user:password@ip_addr:3128/
HTTP_PROXY=http://user:password@ip_addr:3128/
```

replacing the `user`, `password`, and `ip_addr` with the appropriate values. This allows the commands `curl` and `rpm` to access everything.

In addition, you need to add the following lines to `/etc/yum.conf`:



Deploy to every cloud.

Your gateway to multi-cloud deployment automation, remaining independent from the underlying IaaS platform used. See the Use Cases below for more info.

Try it for free! No credit card needed.

By clicking 'Sign up for SlipStream', you agree to our terms of service.

Fig. 1: SlipStream Login Page

```
# The proxy server - proxy server:port number
proxy=http://xxx.xxx.xxx.xxx:3128
# The account details for yum connections
proxy_username=xxx
proxy_password=yyy
```

again replacing the values as appropriate.

4.4 Authentication

By default, SlipStream will be configured to authenticate users against its internal database using usernames and passwords for credentials.

SlipStream can also be configured to authenticate users against generated API keys and secrets as well as against external identity providers that use either the OpenID Connect (OIDC) or GitHub OAuth protocols. Doing so requires:

- Registration of the SlipStream server with the external identity provider,
- Creating an appropriate "Session Template" resource within SlipStream, and
- Configuring the server with the parameters of the identity provider.

You may configure SlipStream to use any number of compatible external identity providers.

4.4.1 Internal

When the `ssclj` server starts, it will create a Session Template resource (`session-template/internal`) that allows users to authenticate with SlipStream via usernames and passwords within its own database.

Warning: Although the administrator can delete the `session-template/internal` resource, this is **not** recommended. Deleting this resource will prevent everyone from logging in via their usernames and passwords until the server is restarted.

The values in this template are used by browser and command line clients to guide users during the login process. You may want to customize the "name" or "description" values to better suit your SlipStream deployment.

Assuming that you've configured `ss-curl` (*cURL*) and that you've logged into the server as an administrator using `ss-curl` (*cURL*), you can then download the existing template with the command:

```
ss-curl https://<slipstream_host>/api/session-template/internal
```

substituting your SlipStream host name. You can then modify the JSON document and upload the modified version like so:

```
ss-curl -XPUT -H content-type:application/json \
        -d@internal-modified.json \
        https://<slipstream_host>/api/session-template/internal
```

Your modifications will be persisted in the database and will survive restarts of the `ssclj` service. You should generally not add or remove keys, as this will likely result in a document that does not conform to the defined schema.

4.4.2 API Key and Secret

Users can generate API keys and secrets to allow for timed or revocable access to the SlipStream server, either via the browser or the API.

To allow users to authenticate with an API key and secret, you must create a Session Template that authorizes this. No other configuration is required.

Upload Session Template

Each available authentication method is associated with a "Session Template" resource. You must create one that will use the API keys and secrets generated by users. A JSON representation of the Session Template resource looks like the following:

```
{
   "method": "api-key",
   "instance": "api-key",

   "name" : "Login with API Key and Secret",
   "description" : "Authentication with API Key and Secret",
   "group" : "Login with API Key and Secret",

   "key" : "key",
   "secret" : "secret",

   "acl": {
             "owner": {"principal": "ADMIN",
                       "type":      "ROLE"},
             "rules": [{"principal": "ADMIN",
                        "type":      "ROLE",
                        "right":     "ALL"},
                       {"principal": "ANON",
                        "type":      "ROLE",
                        "right":     "VIEW"},
                       {"principal": "USER",
                        "type":      "ROLE",
                        "right":     "VIEW"}]
          }
}
```

For API key and secret access, the value for the "method" key must be "api-key". You may set "instance" to any identifier that you would like, although "api-key" is a good choice.

The values for the "name", "description", and "group" keys are used by the clients to present useful information to the users.

The ACL must allow the "ANON" role to view the template; if you do not allow this, then unauthenticated users will not be able to view and to use this Session Template for logging into the server.

Assuming that you've configured `ss-curl` (*cURL*) that you've logged into the server as an administrator using `ss-curl` (*cURL*), you can then **create** a new resource from your file like so:

```
ss-curl -XPOST \
        -H content-type:application/json \
        -d@api-key.json \
        https://<slipstream_host>/api/session-template
```

If this responds with a "201 Created" response, then the resource was properly created.

If the resource already exists, you'll get a "409 conflict" response. If you want to modify an existing resource, simply use PUT the entire modified resource to the resource URL:

```
https://<slipstream_host>/api/session-template/<instance>
```

where the last part corresponds to the "instance" of the resource.

To delete, the session template, just use DELETE on the same URL.

4.4.3 GitHub

GitHub uses a variant of the OAuth protocol to allow external services to use GitHub as an identity provider. To configure SlipStream to use GitHub as an identity provider you must:

1. Register an OAuth application within your GitHub organization,
2. Add an appropriate Session Template resource to your SlipStream server, and
3. Add the associated Configuration resource to your SlipStream server.

The following sections provide detailed instructions for each of these steps.

Register OAuth Application

On GitHub, navigate to the "settings" page for your account or organization. On the left of the page, you will see a link to "OAuth Apps" in the "Developer settings" section. Click on this link.

In the upper-right corner of the page, you will find a "Register a new application" button. Click on this button. You will then see a form similar to the following screenshot.

You must provide an appropriate "Application name". This will be presented to the users who try to log in via SlipStream and should be something that users will recognize and associate with your SlipStream instance. You must also provide a "Homepage URL", which will normally point to your SlipStream installation. You may provide a more detailed "Application description" if you want.

The most important field is the "Authorization callback URL". This must contain the right value for the full GitHub authentication workflow to complete correctly. This field prevents other servers from spoofing your SlipStream installation. The value should be:

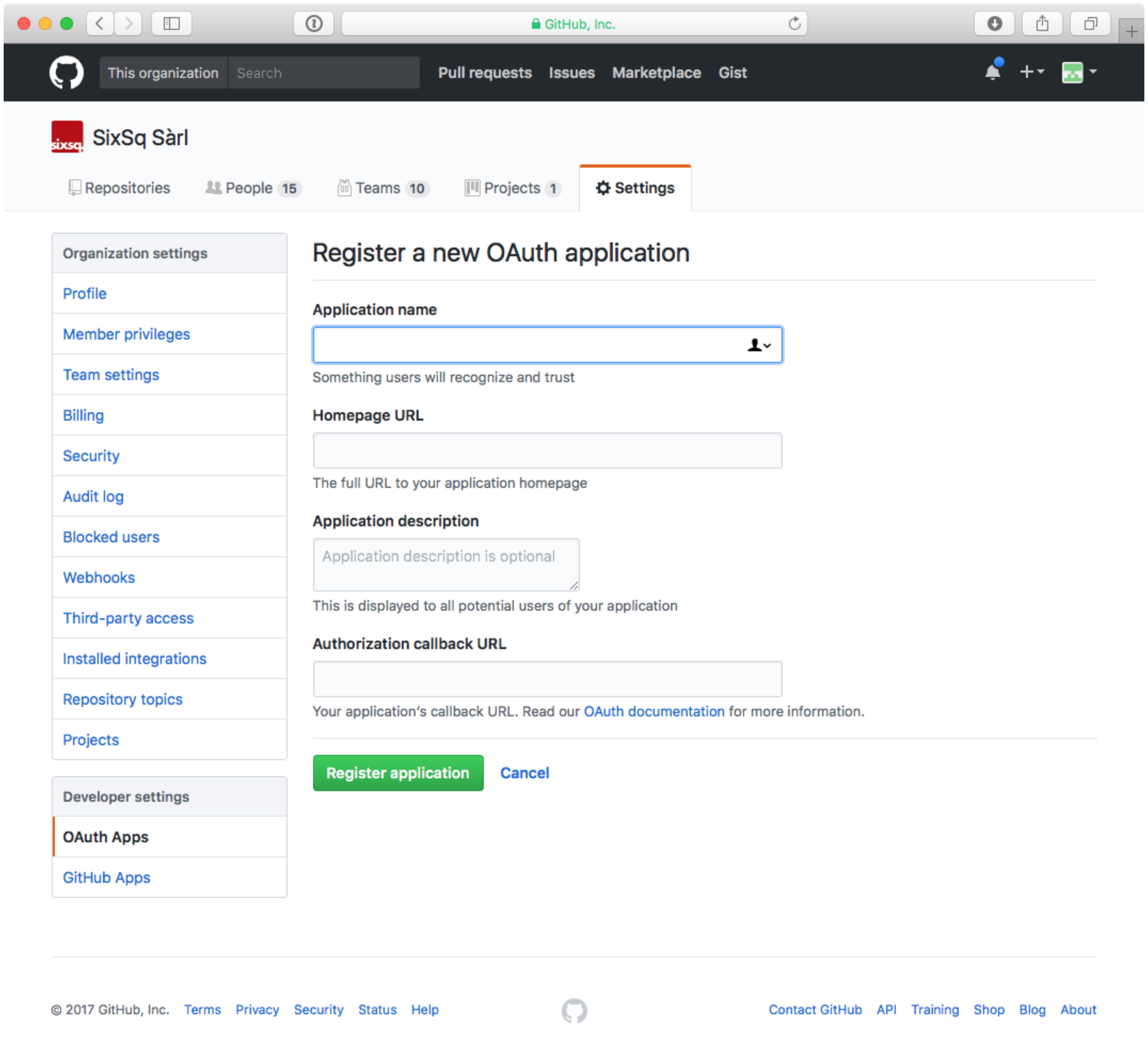
```
https://<slipstream_host>/api/session/
```

where you replace "<slipstream_host>" with the hostname of your SlipStream server.

Once you have provided all of this information, you can click on the "Register application" button. This will then take you to the application page that will show you the "Client ID" and "Client Secret". **These two values are required for the SlipStream server configuration.** From this page, you can also supply a service logo that will be presented to the user during the authentication process.

Upload Session Template

Each available authentication method is associated with a "Session Template" resource. You must create one that will use the GitHub OAuth App that you have defined. A JSON representation of the Session Template resource looks like the following:



Register a new OAuth application

Application name

Something users will recognize and trust

Homepage URL

The full URL to your application homepage

Application description

Application description is optional

This is displayed to all potential users of your application

Authorization callback URL

Your application's callback URL. Read our OAuth documentation for more information.

Fig. 2: GitHub OAuth Application Registration Form

```
{
   "method": "github",
   "instance": "github-test",
   "name": "Sign In with GitHub (Test)",
   "description": "GitHub Authentication Using the Test Application Definition",
   "acl": {
             "owner": {"principal": "ADMIN",
                       "type":      "ROLE"},
             "rules": [{"principal": "ADMIN",
                        "type":      "ROLE",
                        "right":     "ALL"},
                       {"principal": "ANON",
                        "type":      "ROLE",
                        "right":     "VIEW"},
                       {"principal": "USER",
                        "type":      "ROLE",
                        "right":     "VIEW"}]
          }
}
```

For GitHub OAuth Apps, the value for the "method" key must be "github". You may set "instance" to any identifier that you would like; this identifier is used in the server configuration described below.

The values for the "name" and "description" keys are used by the clients to present useful information to the users.

The ACL must allow the "ANON" role to view the template; if you do not allow this, then unauthenticated users will not be able to view and to use this Session Template for logging into the server.

Assuming that you've configured `ss-curl` (*cURL*) that you've logged into the server as an administrator using `ss-curl` (*cURL*), you can then **create** a new resource from your file like so:

```
ss-curl -XPOST \
        -H content-type:application/json \
        -d@github.json \
        https://<slipstream_host>/api/session-template
```

If this responds with a "201 Created" response, then the resource was properly created.

If the resource already exists, you'll get a "409 conflict" response. If you want to modify an existing resource, simply use PUT the entire modified resource to the resource URL:

```
https://<slipstream_host>/api/session-template/<instance>
```

where the last part corresponds to the "instance" of the resource.

To delete, the session template, just use DELETE on the same URL.

Configure SlipStream

You must provide the configuration parameters for the GitHub OAuth application to the `ssclj` server. This is done by adding a Configuration resource to the server.

```
{
    "configurationTemplate": {
        "href": "configuration-template/session-github",
        "instance": "github-test",
        "clientID": "<your client id>",
```

(continues on next page)

(continued from previous page)

```
        "clientSecret": "<your client secret>"
    }
}
```

Note that the value of the `href` attribute must be exactly as above and the value of the `instance` must be the same as in your Session Template resource.

The "Client ID" and "Client Secret" are the values that you obtained from your application registration in GitHub.

Like the other resources, this can be added to the server via a POST request.

```
ss-curl -XPOST \
        -H content-type:application/json \
        -d@configuration-github.json \
        https://<slipstream_host>/api/configuration
```

This will create the resource. Use a PUT or DELETE on the created resource to modify or delete it, respectively.

4.4.4 OpenID Connect (OIDC)

OpenID Connect (OIDC) is an identity layer built over the OAuth 2.0 protocol. Many services support the OIDC protocol (or variants of it) and can potentially be used as identity providers for SlipStream, for example, Google and LinkedIn.

SlipStream has been tested with the Keycloak service, which acts as a federated identity provider and which can be used to access many other services even if they are not directly supported by SlipStream.

The deployment and configuration of a Keycloak server is not described here. Please see the Keycloak website for that information. You take a look at SixSq's Keycloak configuration for the Nuvla service.

Upload Session Template

Each available authentication method is associated with a "Session Template" resource. You must create one that will use the OIDC protocol with Keycloak (or another compatible OIDC identity provider). A JSON representation of the Session Template resource looks like the following:

```
{
   "method": "oidc",
   "instance": "keycloak",
   "name": "Sign In with eduGAIN or Elixir AAI",
   "description": "OIDC Authentication Using Nuvla Keycloak Server for eduGAIN or␣
↪Elixir AAI",
   "acl": {
             "owner": {"principal": "ADMIN",
                       "type":      "ROLE"},
             "rules": [{"principal": "ADMIN",
                        "type":      "ROLE",
                        "right":     "ALL"},
                       {"principal": "ANON",
                        "type":      "ROLE",
                        "right":     "VIEW"},
                       {"principal": "USER",
                        "type":      "ROLE",
                        "right":     "VIEW"}]
```

(continues on next page)

(continued from previous page)

```
          }
}
```

For OIDC-based services, the value for the "method" key must be "oidc". You may set "instance" to any identifier that you would like; this identifier is used in the server configuration described below.

The values for the "name" and "description" keys are used by the clients to present useful information to the users.

The ACL must allow the "ANON" role to view the template; if you do not allow this, then unauthenticated users will not be able to view and to use this Session Template for logging into the server.

Assuming that you've configured `ss-curl` (*cURL*) and that you've logged into the server as an administrator using `ss-curl` (*cURL*), you can then upload your template like so:

```
ss-curl -XPOST \
        -H content-type:application/json \
        -d@keycloak.json \
        https://<slipstream_host>/api/session-template
```

If this responds with a "201 Created" response, then the resource was properly created.

If the resource already exists, you'll get a "409 conflict" response. If you want to modify an existing resource, simply use PUT the entire modified resource to the resource URL:

```
https://<slipstream_host>/api/session-template/<instance>
```

where the last part corresponds to the "instance" of the resource.

To delete, the session template, just use DELETE on the same URL.

Configure SlipStream

You must provide the configuration parameters for the OIDC server to the `ssclj` server by adding a Configuration resource.

```
{
    "configurationTemplate": {
        "href": "configuration-template/session-oidc",
        "instance": "keycloak",
        "clientID": "<your client ID>",
        "baseURL": "<your base URL>",
        "publicKey": "<your RSA public key>",
    }
}
```

Note that the value of the `href` attribute must be exactly as above and the value of the `instance` must be the same as in your Session Template resource.

The "Client ID", "baseURL", and "publicKey" can be obtained from the administrator of the OIDC service which you are using.

Like the other resources, this can be added to the server via a POST request.

```
ss-curl -XPOST \
        -H content-type:application/json \
        -d@configuration-keycloak.json \
        https://<slipstream_host>/api/configuration
```

This will create the resource. Use a PUT or DELETE on the created resource to modify or delete it, respectively.

4.4.5 Link Authentications to a User Account

A user may login to her account using multiple different authentication methods. For example, the user may have both a username/password pair and be able to authenticate via GitHub. Currently, the administrator must configure the server to link multiple authentication methods to a single account.

Warning: Although each authentication method will link to the same SlipStream account, the rights given to the user may be different depending on the authentication method. This is particularly true when using external identity federations that pass attributes to SlipStream.

Via the Browser Interface

In order to know which authentication(s) (aka user identifiers) are currently linked to a given user: visit the page user-identifier link on the CIMI resources page, and add a filter such as:

```
user/href="user/my-user"
```

Use the `Columns` button to include the *identifier* attribute in the displayed table.

An *identifier* is composed of :

- A domain instance name prefix, followed by
- A `:` separator, and
- The actual external login name.

Examples include:

```
github:agithublogin
sixsq:john.smith@unitedid.orghttps://idp.unitedid.org/idp/shibboleth!https://fed-id.
↪nuv.la/samlbridge/module.php/saml/sp/metadata.php/sixsq-saml-bridge!aaa-bbb-ccc
```

The prefix comes from that last part of the `id` field of the associated `session-template` resource.

To link a new authentication to a user via the `Add` button (available after an initial `Search` on user-identifier listing page), replace the values `<MY_USER>`, `<INSTANCE>` and `<EXTERNAL_LOGIN>` in the JSON document that will be submitted, using the pattern below:

```
{
"user" : {"href" : "user/<MY_USER>"},
"identifier" : "<INSTANCE>:<EXTERNAL_LOGIN>"
}
```

Once the document has been created, the user will be able to login to SlipStream with the associated method and identifier.

Note: Attempting to assign the same identifier to more than one Nuvla user will return a 409 (conflict) status and will not create the document.

Via the ss-curl Command

A new user identifier can be added to the server via a POST request.

```
ss-curl -XPOST \
        -H content-type:application/json \
        -d@user-identifier.json \
        https://<slipstream_host>/api/user-identifier
```

where the `user-identifier.json` content follows the same pattern as above:

```
{
"user" : {"href" : "user/<MY_USER>"},
"identifier" : "<INSTANCE>:<EXTERNAL_LOGIN>"
}
```

Like when using the web interface, you can not assign the same identifier value (<INSTANCE>:<EXTERNAL_LOGIN>) to more than one Nuvla user.

4.5 Maintenance

SlipStream, as for every service, needs proper care and feeding! This chapter provides some information about best practices for this service.

4.5.1 Activity Overview

When logged in as an administrator, you can get an overview of all running deployments and virtual machines by visiting the dashboard. Just click on the "dashboard" icon at the top of the page.

4.5.2 Log Files

The Web service log files are located in `/opt/slipstream/server/logs`. For the API Service, they are located in `/var/log/slipstream/ssclj`. (The configuration of API Service logging is driven by `/opt/slipstream/ssclj/resources/log4j.properties`) These are named by date and rotated daily (or when the service is restarted). You should regularly review errors in the log to see if there are configuration or resource problems.

4.5.3 Database Backups

The real value of a SlipStream instance is in the defined images and deployments. These and a full history are kept in the HSQLDB database. This database should be backed up regularly to avoid having any of this valuable information lost.

4.5.4 Maintenance mode

SlipStream can be set in maintenance mode to prevent users to access it and to display an explanatory message instead. Moreover the properly HTTP code (503) will be returned so that search engines can handle that case correctly.

Enabling

To enable the maintenance mode, edit the file `/etc/nginx/conf.d/slipstream-extra/maintenance.map`.

```
# This file define which IPs will receive an error page which explain that the␣
↪application is in maintenance.
# 0 - Maintenance disabled
# 1 - Maintenance enabled
# One IP per line

default  0;

# Localhost
"~^127\.[0-9]+\.[0-9]+\.[0-9]+$" 0;

# Autoconfig IPs
#  "~^169\.254\.[0-9]+\.[0-9]+$"  0;

# Private subnets
#  "~^10\.[0-9]+\.[0-9]+\.[0-9]+$"  0;
#  "~^192\.168\.[0-9]+\.[0-9]+$"    0;
#  "~^172\.(1[6-9]|2[0-9]|3[0-1])\.[0-9]+\.[0-9]+$"  0;
```

On this file, you can define which IPs are affected or not by the maintenance mode. On the original file above, everybody (`default`) will be affected by the maintenance mode if you set it to `1` but SlipStream can still be accessed from localhost (`~^127\.[0-9]+\.[0-9]+\.[0-9]+$`) if you keep it to `0`.

Then execute the following command:

```
$ service nginx reload
```

Disabling

To disable the maintenance mode, you have to set `default` and all custom IPs to `0`. And then reload nginx (see command above).

4.5.5 Customize error pages

All error pages are static files that you can find in `/opt/slipstream/server/webapps/slipstream.war/static-content/error/`

4.5.6 Authentication keys

Certificates for generation of authentication tokens are not password-protected. The new unencrypted certificates are by default generated under `/etc/slipstream/auth` as part of the post-install script of `slipstream-ssclj` RPM. Next time when RPM gets updated the files will not be overwritten.

Regeneration of Authentication keys

The authentication keys can be regenerated at any time. Check `/opt/slipstream/ssclj/bin/generate-auth-keys.sh` script for the details.

Only one service `ssclj.service` requires private key for encrypting the authentication token. All other services require only public key for decryption. Locations of both can be configured in their respective `systemd` configuration files or in the respective `/etc/default/<service>` files.

After the new keys are generated and/or their locations are updated in the configuration files, restart the following services:

```
systemctl restart ssclj slipstream ss-pricing
```

4.6 Cloud Connectors

SlipStream supports a variety of different cloud providers through a set of cloud connectors. Generally the connectors for open source cloud providers are also open source; those for commercial providers are released under a proprietary license.

Doing the following:

```
$ yum search slipstream-connector-
```

will list all of the available cloud connector packages.

4.6.1 Activating a Cloud Connector

To activate a connector for a specific cloud provider, you must install the appropriate cloud connector package and configure the server to make the connector visible to users.

4.6.2 Installing Cloud Connector Packages

All of the SlipStream cloud connectors are packaged separately. You must install the packages for the clouds that you want to support through your SlipStream server.

See the connector-specific sections for detailed installation instructions for each cloud connector. If the documentation for your connector is not present, then get in touch with SixSq Support.

Important: Note that the SlipStream server must be restarted after installing or removing a cloud connector.

4.6.3 Making Connector Visible to Users

In the configuration page, under the *SlipStream Basics* section, you specify which connectors you want the SlipStream server to load and activate.

The value for this parameter is a comma-separated list of `name:connector` pairs. If the name is not supplied, the default will be used. The same connector can be used multiple times with different names (and parameters).

For example, the value:

```
cloudstack,cloud2:cloudstack,cloud9:openstack
```

will load the CloudStack connector twice (with names "cloudstack" and "cloud2") and the CloudStack connector once (with name "cloud9"). The names given to the connecto instances should be in kebab-case.

The cloud and connector names are given in the following table. Note that the connector names are case insensitive.

Cloud	Connector Name	License
Amazon EC2	aws	proprietary
CloudStack	cloudstack	open-source (Apache 2.0)
Exoscale	exoscale	proprietary
OpenNebula	opennebula	open-source (Apache 2.0)
OpenStack	openstack	open-source (Apache 2.0)
Open Telekom Cloud	otc	proprietary
NuvlaBox	nuvlabox	proprietary
VMware vCloud	vcloud	proprietary

4.6.4 Connector Licenses

The connectors are released under different licenses, depending on the cloud solution with which they interface. Generally, connectors talking to open source IaaS cloud solutions are released under an open source license, while for proprietary IaaS solutions, the connectors are released under a proprietary license.

To get details on the terms for proprietary licenses, please get in touch with SixSq Support.

4.7 Connector-Specific Info

4.7.1 CloudStack

Preparation

Server-side client (`cloudstack-*` CLI) requires `apache-libcloud`. It can be installed with `pip`. The version should be the same as defined in `slipstream.sixsq.com:SlipStream:pom.xml` with `libcloud.version` property.

Installation

You can install the CloudStack connector with:

```
$ yum install slipstream-connector-cloudstack
```

You will need to restart the SlipStream server to make this connector visible.

Configuration

To allow users to take advantage of this connector, you must add one or more instances of this connector by either:

1. Using the *UI*.
2. Drop a *configuration file* and restart the service.

With the UI

Instanciate one or more instances of the connector

Once logged-in with a privileged user (e.g. *super*), open the configuration page by clicking on *Configuration* -> *System* at the top of the page. Then open the *SlipStream Basics* section and define a new instance of the connector with the following format:

```
<connector-instance-name>:<connector-name>
```

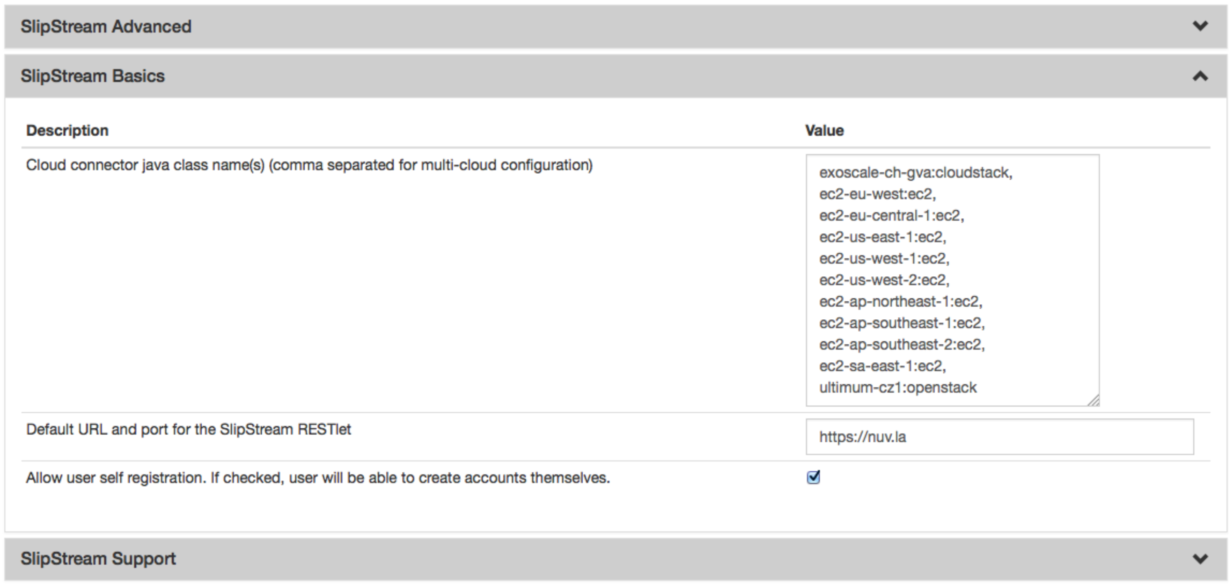
Here is an example:

```
exoscale-ch-gva:cloudstack
```

You can also instantiate the connector several times (in compliance with your license) by comma separating the connector string. Here is an example:

```
my-cs-1:cloudstack, my-cs-2:cloudstack, ...
```

Here is a screenshot of the parameter to define:



Description	Value
Cloud connector java class name(s) (comma separated for multi-cloud configuration)	exoscale-ch-gva:cloudstack, ec2-eu-west:ec2, ec2-eu-central-1:ec2, ec2-us-east-1:ec2, ec2-us-west-1:ec2, ec2-us-west-2:ec2, ec2-ap-northeast-1:ec2, ec2-ap-southeast-1:ec2, ec2-ap-southeast-2:ec2, ec2-sa-east-1:ec2, ultimum-cz1:openstack
Default URL and port for the SlipStream RESTlet	https://nuv.la
Allow user self registration. If checked, user will be able to create accounts themselves.	✓

Fig. 3: SlipStream Configuation - Basics section

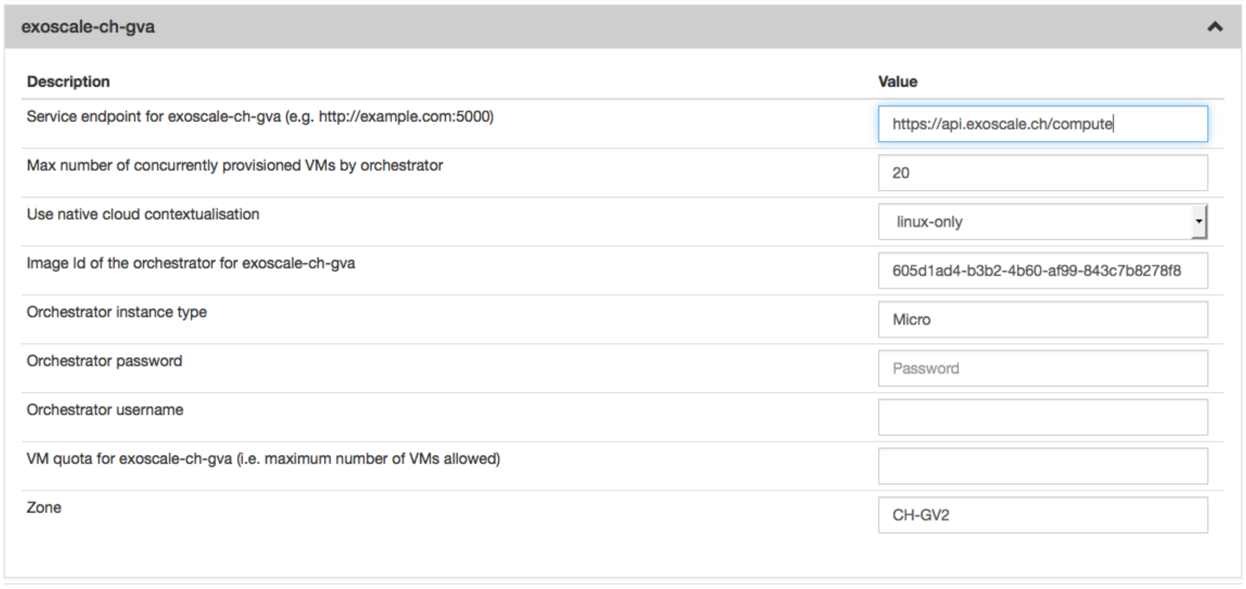
Don't forget to save the configuration!

Now that the connector is loaded, you need to configure it.

Configure the connector instance

With the connector loaded in SlipStream, a new section in the configuration page will appear, allowing you to configure how the connector is to communicate with the IaaS cloud endpoint.

You can find a detailed description of each parameter as well as an explaination of how to find the right value of them in the `Parameters <#parameters>'__ paragraph below.



exoscale-ch-gva

Description	Value
Service endpoint for exoscale-ch-gva (e.g. http://example.com:5000)	https://api.exoscale.ch/compute
Max number of concurrently provisioned VMs by orchestrator	20
Use native cloud contextualisation	linux-only
Image Id of the orchestrator for exoscale-ch-gva	605d1ad4-b3b2-4b60-af99-843c7b8278f8
Orchestrator instance type	Micro
Orchestrator password	Password
Orchestrator username	
VM quota for exoscale-ch-gva (i.e. maximum number of VMs allowed)	
Zone	CH-GV2

Fig. 4: SlipStream Configuation - CloudStack section

With a Configuration File

Please see *Configuration Files* for details about this method of configuration.

Here is an example, which will configure the CloudStack connector to interact with Exoscale:

```
> cat /etc/slipstream/connectors/exoscale-ch-gva.conf
cloud.connector.class = exoscale-ch-gva:cloudstack
exoscale-ch-gva.endpoint = https://api.exoscale.ch/compute
exoscale-ch-gva.zone = CH-GV2
exoscale-ch-gva.quota.vm = 20
exoscale-ch-gva.orchestrator.imageid = 605d1ad4-b3b2-4b60-af99-843c7b8278f8
exoscale-ch-gva.orchestrator.instance.type = Micro
exoscale-ch-gva.orchestrator.ssh.password =
exoscale-ch-gva.orchestrator.ssh.username =
exoscale-ch-gva.native-contextualization = linux-only
exoscale-ch-gva.max.iaas.workers = 20
```

You can find a detailed description of each parameter as well as an explaination of how to find the right value of them in the ``Parameters` <#parameters>`__ paragraph below.

Parameters

Note: All the CloudStack API examples come from 'cloudmonkey <https://cwiki.apache.org/confluence/display/CLOUDSTACK/CloudSta configured with 'Exoscale Open Cloud API <https://community.exoscale.ch/api/compute/>'__ endpoint.

Zone

The availability zone where the virtual machines will be provisionned.

Use the `name` value from the listZones results:

```
> list zones
count = 1
zone:
name = ch-gva-2
id = 1128bd56-b4d9-4ac6-a7b9-c715b187ce11
[...]
```

Image Id of the Orchestrator

The image id of the Orchestrator needs to match a Linux image with `wget` and `python` installed. An Ubuntu 12.04 will do the job perfectly.

Use the `id` value from the listTemplates results:

```
> list templates templatefilter=featured
count = 37
template:
id = 8c7e60ae-3a30-4031-a3e6-29832d85d7cb
name = Linux Ubuntu 12.04 LTS 64-bit
[...]
```

For Exoscale you can browse the available templates and choose the one that suits your need.

Flavor of the Orchestrator

The flavor (instance type) is a name which is linked to a hardware specification defined by the Cloud. The Orchestrator doesn't need a large amount of resources so you can choose a small flavor (like 1 CPU and 512 MB of RAM).

Use the `name` value from the listServiceOfferings results:

```
> list serviceofferings
count = 7
serviceoffering:
name = Micro
id = 71004023-bb72-4a97-b1e9-bc66dfce9470
[...]
```

Quota

The quota is a SlipStream feature which enable the SlipStream administrator to set a default quota for all users of a specified connector. You can also override this value per user in the user profile. If this feature is disabled in the *SlipStream Advanced* section of this page, you can leave this field blank.

Service endpoint

The CloudStack API Endpoint used by SlipStream to communicate with the CloudStack Cloud.

Example: `https://api.exoscale.ch/compute`

Configure Native Images for This Connector Instance

Now you need to update SlipStream native images to add the image id and some parameters for CloudStack.

This can be done via the UI or via configuration file. Documentation about how to do it via configuration file can be found here *Unique Cloud Identifier Configuration Files*.

Please go on a SlipStream base image (e.g. Ubuntu 12.04) and click on the *Edit* button. Add the image id for CloudStack in the section named *Cloud Image Identifiers and Image Hierarchy*.

And then configure the default amount of CPU and RAM on the tab *CloudStack* (or the name you gave your CloudStack connector earlier) of the section *Cloud Configuration*.

User Credentials

Now that the connector is configured and the native images updated, inform your users that they need to configure their credentials for CloudStack in their user profile to take advantage of your new connector.

4.7.2 Amazon EC2

License

This connector is distributed by SixSq under a commercial license and is available via various pricing plans. You can check it out with a **free** trail via our SaaS service. Feel free to contact the SlipStream Support team with any questions about how the SlipStream Amazon EC2 connector can be beneficial to you and your business.

Once you have purchased a new commercial connector, SixSq will provide you with a specific yum configuration.

Installation

Once your yum configuration is in place, you can install the connector executing the following command:

```
$ yum install slipstream-connector-ec2-enterprise
```

With the software installed, you need to restart the SlipStream service in order for it to take the new connector into account:

```
$ systemctl restart slipstream
```

Now we need to configure SlipStream to take advantage of the new connector.

Configuration

To allow users to take advantage of this connector, you must add one or more instances of this connector by either:

1. Using the *UI*.
2. Drop a *configuration file* and restart the service.

With the UI

Instanciate one or more instances of the connector

Once logged-in with a privileged user (e.g. *super*), open the configuration page by clicking on *Configuration* -> *System* at the top of the page. Then open the *SlipStream Basics* section and define a new instance of the connector with the following format:

```
<connector-instance-name>:<connector-name>
```

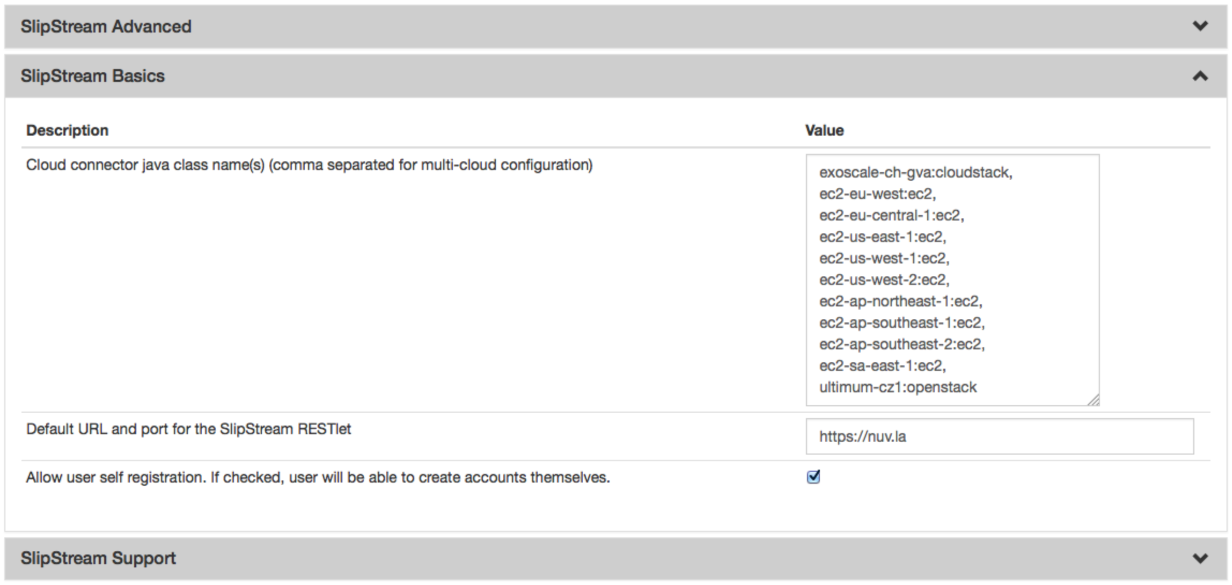
Here is an example:

```
amazon-ec2:ec2
```

You can also instantiate the connector several times (in compliance with your license) by comma separating the connector string. Here is an example:

```
ec2-eu-central-1:ec2, ec2-eu-west-1:ec2, ...
```

Here is a screenshot of the parameter to define:



Description	Value
Cloud connector java class name(s) (comma separated for multi-cloud configuration)	exoscale-ch-gva:cloudstack, ec2-eu-west:ec2, ec2-eu-central-1:ec2, ec2-us-east-1:ec2, ec2-us-west-1:ec2, ec2-us-west-2:ec2, ec2-ap-northeast-1:ec2, ec2-ap-southeast-1:ec2, ec2-ap-southeast-2:ec2, ec2-sa-east-1:ec2, ultimum-cz1:openstack
Default URL and port for the SlipStream RESTlet	https://nuv.la
Allow user self registration. If checked, user will be able to create accounts themselves.	✓

Fig. 5: SlipStream Configuation - Basics section

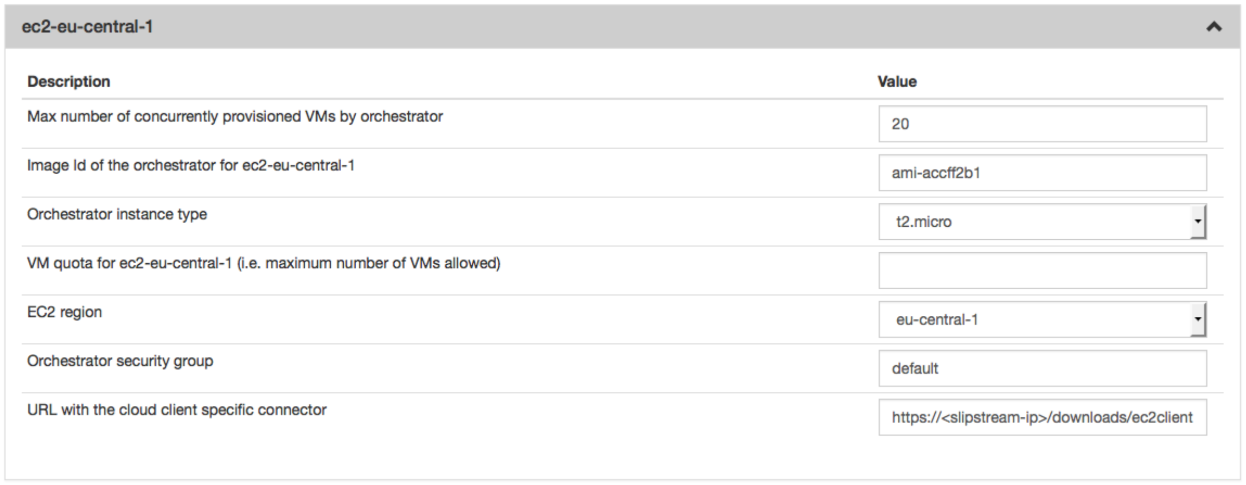
Don't forget to save the configuration!

Now that the connector is loaded, you need to configure it.

Configure the connector instance

With the connector loaded in SlipStream, a new section in the configuration page will appear, allowing you to configure how the connector is to communicate with the IaaS cloud endpoint.

You can find a detailed description of each parameter as well as an explaination of how to find the right value of them in the `` `Parameters `` <#parameters>'__ paragraph below.


ec2-eu-central-1

Description	Value
Max number of concurrently provisioned VMs by orchestrator	20
Image Id of the orchestrator for ec2-eu-central-1	ami-accff2b1
Orchestrator instance type	t2.micro
VM quota for ec2-eu-central-1 (i.e. maximum number of VMs allowed)	
EC2 region	eu-central-1
Orchestrator security group	default
URL with the cloud client specific connector	https://<slipstream-ip>/downloads/ec2client

Fig. 6: SlipStream Configuation - EC2 section

With a Configuration File

Please see *Configuration Files* for details about this method of configuration.

Here is an example, which will configure the EC2 connector to interact with the region eu-central-1:

```
> cat /etc/slipstream/connectors/ec2-eu-central-1.conf
cloud.connector.class = ec2-eu-central-1:ec2
ec2-eu-central-1.security.group = default
ec2-eu-central-1.update.clienturl = https://<slipstream-ip>/downloads/ec2client.tgz
ec2-eu-central-1.orchestrator.instance.type = t2.micro
ec2-eu-central-1.region = eu-central-1
ec2-eu-central-1.quota.vm =
ec2-eu-central-1.max.iaas.workers = 20
ec2-eu-central-1.orchestrator.imageid = ami-accff2b1
```

You can find a detailed description of each parameter as well as an explaination of how to find the right value of them in the `Parameters <#parameters>‘__ paragraph below.

Parameters

EC2 region

The EC2 region defines where your EC2 instances will be deployed. Some parameters may be different between regions (e.g. image id: ami-...). If you want to use multiple regions concurrently, you will need to instantiate this connector multiple times.

Cloud Client Connector

This field corresponds to the URL where the Orchestrator will download the tarball of the connector for the SlipStream Client. In a default installation the URL will be `https://ip_or_hostname/downloads/ec2client.tgz` where `ip_or_hostname` corresponds to the IP or the hostname of your SlipStream Server.

Orchestrator security group

The EC2 security group should allow TCP connexions from the Orchestrator itself to the SlipStream server and to the EC2 API. The default security group named `default` should normally work perfectly.

Image Id of the Orchestrator

The image id of the Orchestrator needs to match a Linux image with `wget` and `python` installed. An Ubuntu 12.04 or 14.04 will do the job perfectly (at the time or writing, for the region `eu-west-1` the image id is `ami-a0dd3dd7`). EC2 image ids start with `ami-`. You can found them in the EC2 web interface.

Quota

The quota is a SlipStream feature which enables the SlipStream administrator to set a default quota for all users of a specified connector. You can also override this value for each user in the user profile. If this feature is disabled in the *SlipStream Advanced* section of this page, you can leave this field blank.

Orchestrator instance type

The instance type is a name which is linked to a hardware specification defined by EC2. You can find the list of all possible values here. The Orchestrator doesn't need a big amount of resources so you can choose a small instance type (like `t2.micro` or `t2.small`).

Configure Native Images for This Connector Instance

Now you need to update SlipStream native images to add the image id and some parameters specific to EC2.

This can be done via the UI or via configuration file. Documentation about how to do it via configuration file can be found here *Unique Cloud Identifier Configuration Files*.

Please go on a SlipStream base image (e.g. Ubuntu 12.04) and click on the *Edit* button. Add the image id for EC2 in the section named *Cloud Image Identifiers and Image Hierarchy*.

And then configure the default amount of CPU and RAM on the tab *ec2* (or the name you gave your EC2 connector earlier) of the section *Cloud Configuration*.

User Credentials

Now that the connector is configured and the native images updated, inform your users that they need to configure their credentials for EC2 in their user profile to take advantage of your new connector.

4.7.3 OpenStack

Installation

You can install the OpenStack connector with:

```
$ yum install slipstream-connector-openstack
```

You will need to restart the SlipStream server to make this connector visible.

Configuration

To allow users to take advantage of this connector, you must add one or more instances of this connector by either:

1. Using the *UI*.
2. Drop a *configuration file* and restart the service.

With the UI

Instanciate one or more instances of the connector

Once logged-in with a privileged user (e.g. *super*), open the configuration page by clicking on *Configuration* -> *System* at the top of the page. Then open the *SlipStream Basics* section and define a new instance of the connector with the following format:

```
<connector-instance-name>:<connector-name>
```

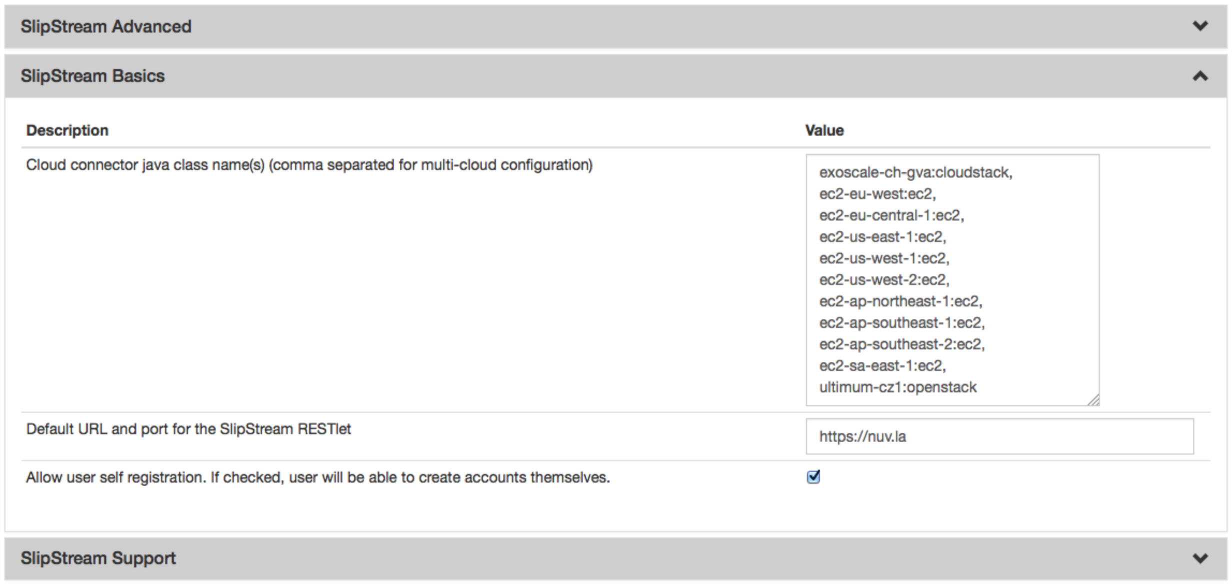
Here is an example:

```
ultimum-cz1:openstack
```

You can also instantiate the connector several times (in compliance with your license) by comma separating the connector string. Here is an example:

```
my-os-1:openstack, my-os-2:openstack, ...
```

Here is a screenshot of the parameter to define:



Description	Value
Cloud connector java class name(s) (comma separated for multi-cloud configuration)	exoscale-ch-gva:cloudstack, ec2-eu-west:ec2, ec2-eu-central-1:ec2, ec2-us-east-1:ec2, ec2-us-west-1:ec2, ec2-us-west-2:ec2, ec2-ap-northeast-1:ec2, ec2-ap-southeast-1:ec2, ec2-ap-southeast-2:ec2, ec2-sa-east-1:ec2, ultimum-cz1:openstack
Default URL and port for the SlipStream RESTlet	https://nuv.la
Allow user self registration. If checked, user will be able to create accounts themselves.	✓

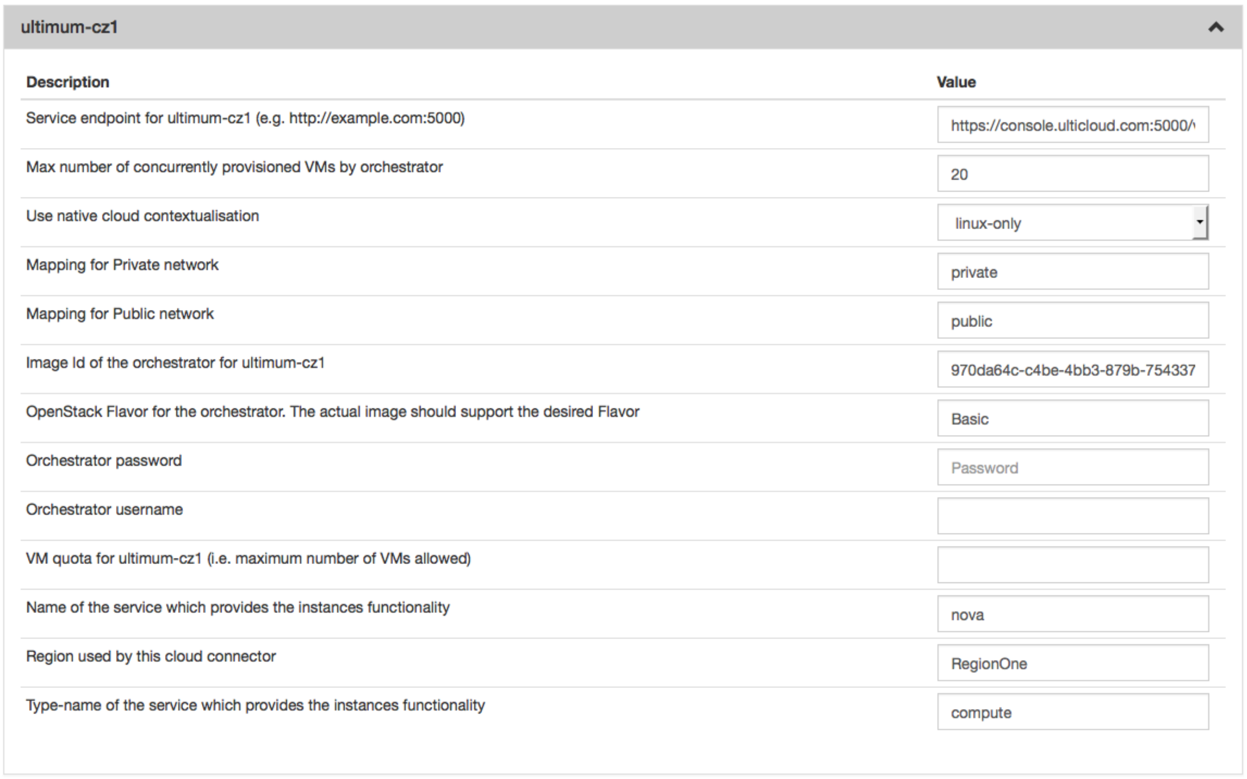
Fig. 7: SlipStream Configuation - Basics section

Don't forget to save the configuration!

Now that the connector is loaded, you need to configure it.

Configure the connector instance

With the connector loaded in SlipStream, a new section in the configuration page will appear, allowing you to configure how the connector is to communicate with the IaaS cloud endpoint.


ultimum-cz1

Description	Value
Service endpoint for ultimum-cz1 (e.g. http://example.com:5000)	https://console.ulticloud.com:5000/
Max number of concurrently provisioned VMs by orchestrator	20
Use native cloud contextualisation	linux-only
Mapping for Private network	private
Mapping for Public network	public
Image Id of the orchestrator for ultimum-cz1	970da64c-c4be-4bb3-879b-754337
OpenStack Flavor for the orchestrator. The actual image should support the desired Flavor	Basic
Orchestrator password	Password
Orchestrator username	
VM quota for ultimum-cz1 (i.e. maximum number of VMs allowed)	
Name of the service which provides the instances functionality	nova
Region used by this cloud connector	RegionOne
Type-name of the service which provides the instances functionality	compute

Fig. 8: SlipStream Configuation - OpenStack section

You can find a detailed description of each parameter as well as an explaination of how to find the right value of them in the `Parameters <#parameters>‘__ paragraph below.

With a Configuration File

Please see *Configuration Files* for details about this method of configuration.

Here is an example, which will configure the OpenStack connector to interact with Ultimum:

```
$ cat /etc/slipstream/connectors/ultimum-cz1.conf
cloud.connector.class = ultimum-cz1:openstack
ultimum-cz1.quota.vm =
ultimum-cz1.max.iaas.workers = 20
ultimum-cz1.service.name = nova
ultimum-cz1.native-contextualization = linux-only
ultimum-cz1.service.region = RegionOne
ultimum-cz1.network.private = private
ultimum-cz1.orchestrator.instance.type = Basic
ultimum-cz1.service.type = compute
ultimum-cz1.orchestrator.ssh.username =
ultimum-cz1.orchestrator.imageid = 970da64c-c4be-4bb3-879b-75433751e71f
ultimum-cz1.network.public = public
```

(continues on next page)

(continued from previous page)

```
ultimum-cz1.endpoint = https://console.ulticloud.com:5000/v2.0/tokens
ultimum-cz1.orchestrator.ssh.password =
```

You can find a detailed description of each parameter as well as an explaination of how to find the right value of them in the `` `Parameters `` <#parameters>'__ paragraph below.

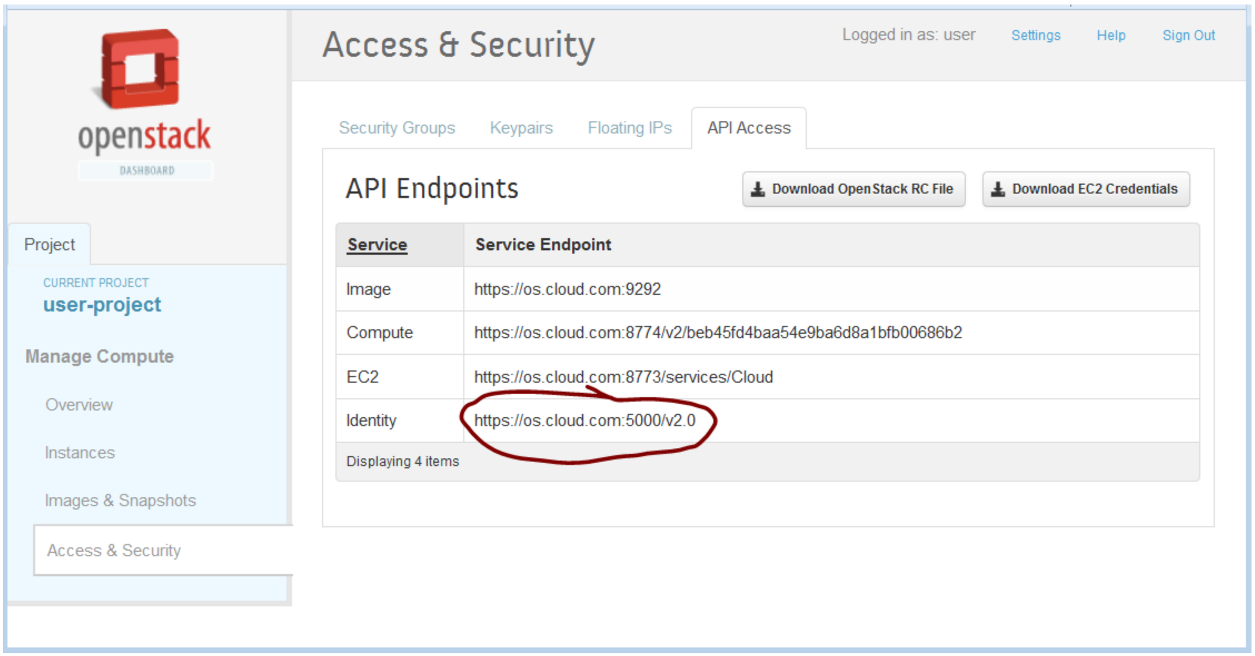
Parameters

Type name of the service which provides the instances functionality

This field should always be `compute`. If it doesn't work with this value, please ask your OpenStack provider/administrator for the correct value.

Service endpoint

The service endpoint is the URL SlipStream will use to communicate with the OpenStack. This url needs to match the OpenStack identity service (Keystone). You can find it in the OpenStack dashboard under *Access & Security* in the tab *API Access*. Most of the time this value will match the following pattern: `https://OpenStack_ip:5000/v2.0`



Service	Service Endpoint
Image	https://os.cloud.com:9292
Compute	https://os.cloud.com:8774/v2/beb45fd4baa54e9ba6d8a1bfb00686b2
EC2	https://os.cloud.com:8773/services/Cloud
Identity	https://os.cloud.com:5000/v2.0

Displaying 4 items

Fig. 9: Openstack web interface - Access & Security - API Access

Quota

The quota is SlipStream feature which enable the SlipStream administrator to set a default quota for all users of a specified connector. You can also override this value per user in the user profile. If this feature is disabled in the *SlipStream Advanced* section of this page, you can leave this field blank.

Image Id of the Orchestrator

The image id of the Orchestrator needs to match a Linux image with `wget` and `python` installed. An Ubuntu 12.04 or 14.04 will do the job perfectly.

To find an image id go on the OpenStack web interface and click on the link named *Images & Snapshots* and then click on the image you want. The *ID* value is what you need to paste in.

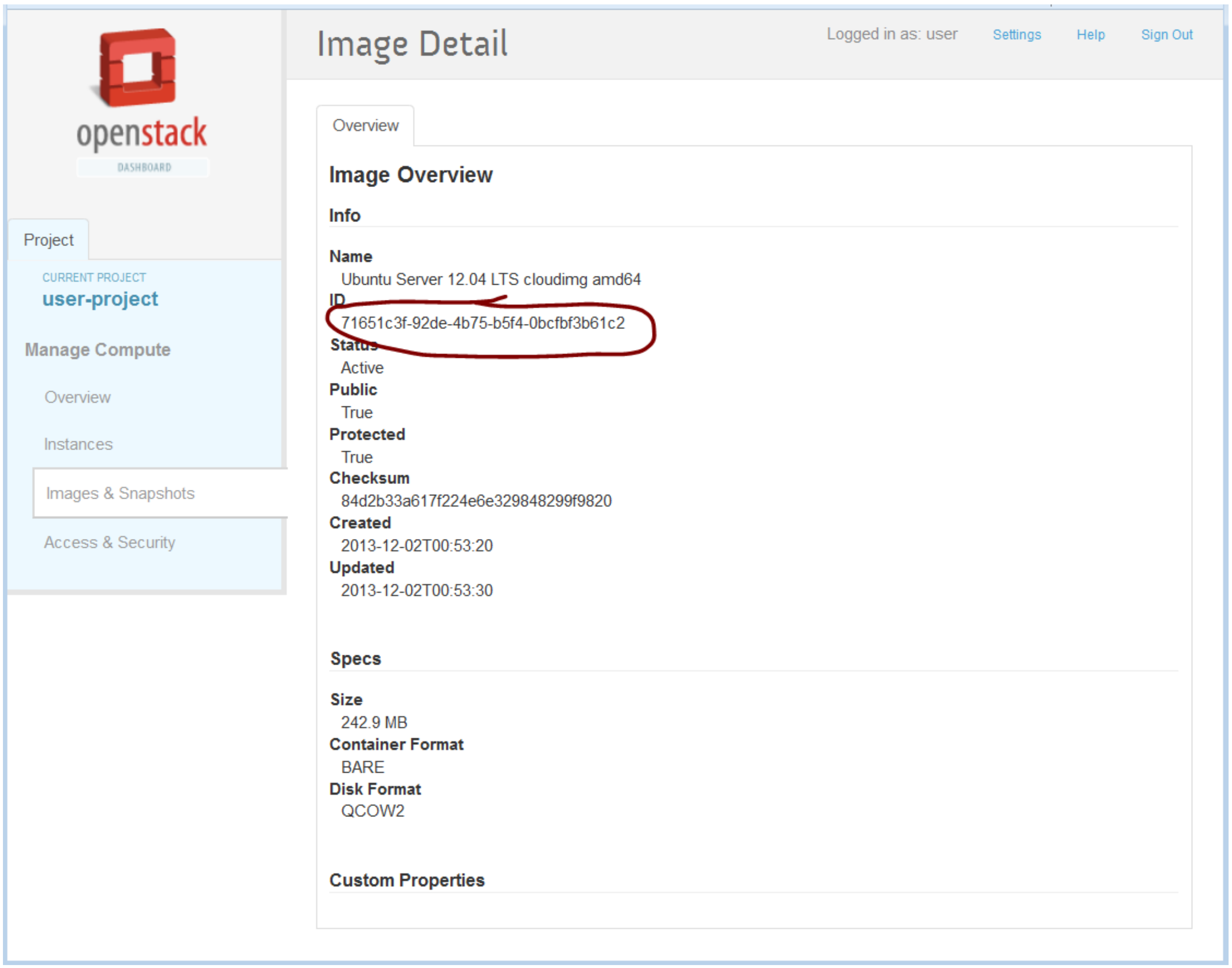


Image Detail

Image Overview

Info

Name
Ubuntu Server 12.04 LTS cloudimg amd64
ID
71651c3f-92de-4b75-b5f4-0bcfbf3b61c2
Status
Active
Public
True
Protected
True
Checksum
84d2b33a617f224e6e329848299f9820
Created
2013-12-02T00:53:20
Updated
2013-12-02T00:53:30

Specs

Size
242.9 MB
Container Format
BARE
Disk Format
QCOW2

Custom Properties

Fig. 10: Openstack web interface - Image details

Region

Check this value in the OpenStack documentation or ask your OpenStack administrator. The default region is `RegionOne` or `regionOne` depending of the OpenStack version.

Name of the service which provides the instances functionality

Most of time the value of this field will be `nova` and sometime `compute`. If it doesn't work with these values, please ask your OpenStack administrator for the correct value.

Flavor of the Orchestrator

The flavor (instance type) is a name which is linked to a hardware specification defined by the Cloud. To find the list of all possible values, please go on the OpenStack web interface and find a link called *Flavor* or *Instance type*. The Orchestrator doesn't need a big amount of resources so you can choose a small flavor (like 1 CPU and 512 MB of RAM).

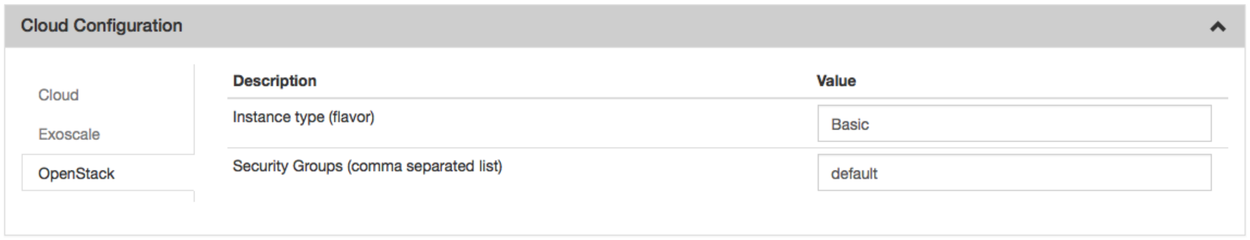
Configure Native Images for This Connector Instance

Now you need to update SlipStream native images to add the image id and some parameters for OpenStack.

This can be done via the UI or via configuration file. Documentation about how to do it via configuration file can be found here *Unique Cloud Identifier Configuration Files*.

Please go on a SlipStream base image (e.g. Ubuntu 14.04) and click on the *Edit* button. Add the image id for OpenStack in the section named *Cloud Image Identifiers and Image Hierarchy*.

And then configure the default *instance type* and the default *security groups* on the tab *OpenStack* (or the name you gave your OpenStack connector earlier) of the section *Cloud Configuration*.



Description	Value
Instance type (flavor)	Basic
Security Groups (comma separated list)	default

Fig. 11: SlipStream Image - edit mode OpenStack

User Credentials

Now that the connector is configured and the native images updated, inform your users that they need to configure their credentials for OpenStack in their user profile to take advantage of your new connector.

CHAPTER 5

Contributors Guide

This guide gives developers the information they need to contribute source code, to build the SlipStream artifacts, and to test SlipStream components.

5.1 Introduction

SlipStream is a multi-cloud application management platform. The core of SlipStream and many of the cloud connectors are open-source and released under the Apache 2 License. Collectively this version is called the **SlipStream Community Edition**.

This guide focuses on the **Community Edition**, providing information on how to:

- Obtain the source code from GitHub,
- Build and package the software, and
- Run SlipStream locally for development.

This guide does not cover the SlipStream API. If you want to use SlipStream programmatically, you can find the complete API documentation on a separate, dedicated site.

The procedures (and API) are identical for the **Enterprise Edition**, except that modules for additional cloud connectors for public cloud services must be included.

The primary programming languages for SlipStream are Clojure, ClojureScript, and Python. Other languages such as Java, JavaScript, bash, etc. are also used.

Maven and Leiningen are used to build the software and various language-specific frameworks are used for testing.

5.2 Dependencies

To build SlipStream, you need to have a variety of languages, tools, and libraries installed on your system.

The target production platform is CentOS 7. However, the software should build without problems on any Unix-like environment (Linux, FreeBSD, MacOS, etc.). Binary packages will only be built on platforms supporting RPM. The core developers primarily use MacOS.

Note: The SlipStream build is **not** supported on Windows and will **not** work.

The following sections describe how to configure your development environment on CentOS 7 and MacOS. If you are using another Unix-like platform, use the CentOS 7 instructions as a guide.

There is also a Docker container that has the entire SlipStream build environment preconfigured. See the *Docker Container for Builds* section for details.

5.2.1 System Packages

CentOS 7

CentOS 7 is the target platform for production deployments of SlipStream. Builds on CentOS 7 are the only ones that are officially supported.

These instructions assume that you are building the software on an **up-to-date, minimal CentOS 7 system**. You should upgrade your system to ensure you have the latest versions of dependencies.

Several of the packages required for the build are not available in the core CentOS 7 distribution. You will need to configure your machine to use the EPEL 7 package repository:

```
$ yum install -y yum-utils epel-release
$ yum-config-manager --enable epel
```

If you're not using a CentOS release, you'll need to find and install the RPM for EPEL configuration on their website. You can find the URL and package name via the information in the "How can I use these extra packages?" section on the EPEL welcome page.

All of the packaged dependencies can be installed directly with `yum`. The following table lists the RPM packages that must be installed and describes how those packages are used within the build.

The command:

```
$ yum install -y \
      git \
      java-1.8.0-openjdk-devel \
      python \
      python-devel \
      python-pip \
      rpm-build \
      createrepo \
      gcc \
      docker \
      which
```

will install all of the listed packages.

Package	Comment
git	Download sources from GitHub
java-1.8.0-openjdk-devel	Compile and run the server
python	Client CLI build and testing
python-devel	Needed for python module dependencies
python-pip	Needed for dependencies installation
rpm-build	Creates binary distribution packages
createrepo	Create local yum repository
gcc	Needed for building python dependencies
docker	Needed to build containers
which	Needed for maven configuration

MacOS

The primary platform used by the SlipStream developers is Mac OS X. Consequently, this is a well-understood and well-supported SlipStream development environment. We're going to assume you're running MacOS with Homebrew.

Your default locale settings may conflict with some of the programs we'll need. If you want to be on the safe side, add these lines to your **.bash_profile** file:

```
export LC_ALL=en_US.UTF-8
export LANG=en_US.UTF-8
```

Most of the necessary dependencies are already installed by default in recent versions of MacOS.

You must install the Java Development Kit on your system. Download and install the JDK 8 package from Oracle's Java download page.

You must also install Docker. You can find the DMG package from the Docker website.

5.2.2 Python Dependencies

CentOS 7

The correct version of Python will have been installed with the system packages described above. Nonetheless, it is recommended that you install and use `pyenv`:

```
$ curl -L https://github.com/pyenv/pyenv-installer/raw/master/bin/pyenv-installer |␣
↪bash
```

You should also add the following to you bash login script:

```
export PATH="/username/.pyenv/bin:$PATH"
eval "$(pyenv init -)"
eval "$(pyenv virtualenv-init -)"
```

This will configure your Python environment and let you choose between different Python versions. **Be sure to change the path to match your username.**

You must also install some Python dependencies via `pip`. See the common configuration below.

MacOS

It is strongly recommended that you install and use pyenv. This provides a more flexible and consistent Python environment on MacOS.

From an account with administrator access, install pyenv:

```
$ brew update
$ brew install pyenv
```

Then from your normal account (if different from the administrator account), adjust your bash login:

```
export PATH="/Users/username/.pyenv:$PATH"
eval "$(pyenv init -)"
eval "$(pyenv virtualenv-init -)"
```

This will allow you to change dynamically the version of Python being used. **Be sure to change the path to match your username.**

If you have errors related to the virtualenv-init command, then run the command:

```
git clone https://github.com/yyuu/pyenv-virtualenv.git ~/.pyenv/plugins/pyenv-
↪virtualenv
```

which will install the missing plugin.

If Python plugin executables are not visible, then you may also need to add the path /Users/username/.local/bin to your PATH variable.

Next install the latest 2.7 release of Python and set this as the default:

```
$ pyenv install 2.7.13
$ pyenv global 2.7.13
```

Note the the download, compilation, and installation of Python will take some time.

Verify that you are using the correct version of Python with:

```
$ pyenv versions
```

and verify with pip -V that pip works and comes from the Python installation that you just performed.

Proceed with the installation of the Python package dependencies described in the next section.

All Platforms

Install the following dependencies that are needed to run the unit tests for the python code:

```
$ pip install tox
```

You can verify that tox is available with which.

5.2.3 Build Tools

Maven

The overall SlipStream build is controlled through Maven.

Download and install the **latest** Maven release from the Apache Maven website. You will need to download the Maven distribution (choose the most recent binary distribution), unpack the distribution and modify the environment to make the `mvn` command visible.

> **Warning:** The Maven version supplied by most operating systems is too old to work with the SlipStream build. You must have at least version 3.3.

Once you have downloaded and unpacked Maven, you can setup the environment with:

```
$ export MAVEN_HOME=<installation directory>/apache-maven-3.3.9
$ export PATH=$PATH:$MAVEN_HOME/bin
```

The `mvn` command should now be visible. The software will build with any maven version later than 3.3.

Leiningen

The clojure SlipStream server (ssclj) and its components are built using Leiningen. Leiningen (`lein`) is triggered via Maven to allow for an integrated build process.

To install `lein`, follow its installation instructions. After installation you must make sure that the `lein` command is in your path by executing:

```
$ lein -v
```

This will download the most recent version of Leiningen and provide you with the version number. The download will only happen on the first invocation of the command.

5.3 Docker Container for Builds

There is a Docker container available from the Docker Cloud/Hub that contains the entire environment for the complete build of SlipStream. You can start this container with the command:

```
$ docker run -it sixsq/slipstream:build bash
```

You can then follow the instructions on checking out the SlipStream sources and then perform the build. The container obviously does not contain any credentials, so you will need to add credentials if you want to check changes into the repositories.

> **Warning:** This is an **alpha feature** and not officially supported. Nonetheless, we would appreciate your feedback on the usefulness of providing a container and on the contents and configuration of the container itself.

5.4 Sources

All of the SlipStream source code is stored in GitHub repositories. The open-source **Community Edition** repositories are under the the SlipStream Organization. You can browse the source code through the GitHub web interface or clone the repositories with a git client.

5.4.1 Prerequisites

The fastest way to checkout all of the source code is through the "bootstrapping" process described below. To use that process, you'll need to have the following installed on your development machine:

- A git client,
- A java development environment (JDK), version 1.8 or later, and
- Maven, version 3.2.0 or later.

Ensure that all of these are installed and working correctly before proceeding.

5.4.2 Bootstrapping

The major SlipStream components are stored in separate repositories on GitHub. To build the entire system, all of the component repositories must be cloned locally. To make this process easier, we've created a "bootstrap" repository that automates the process.

Using the git command line (or your graphical git client), clone the bootstrap repository. The command for this is:

```
$ git clone https://github.com/slipstream/SlipStreamBootstrap
```

This is a public GitHub URL, so you should be able to clone the repository from here either with or without a GitHub account.

5.4.3 Obtaining SlipStream Components

Once you've checked out the SlipStreamBootstrap repository, descend into the root of that repository. From there (assuming that you have a GitHub account), you can then use maven to clone all of the SlipStream repositories.

```
$ cd SlipStreamBootstrap
$ mvn generate-sources
```

All of the SlipStream component repositories will be cloned in the same directory. The checkout by default will be at the HEAD of the master branch. The repository names all start with "SlipStream".

5.4.4 Public GitHub URLs

The above procedure will use the GitHub developer URLs, which require a GitHub account. **If you do not have a GitHub account**, then use the command:

```
$ mvn -P public generate-sources
```

instead. This uses the public URLs for the repositories.

5.4.5 Branches or Tags

By default, the **master** branch will be checked out. If you want a different branch or tag, then use the following command:

```
$ mvn -Dslipstream.version.default=master generate-sources
```

setting the value of "master" to the desired branch or tag. This will consistently use the same branch or tag for all components.

The tags for releases follow the pattern: "v3.55". Look at the release notes to determine tag to use.

Although not recommended, it is sometimes useful to mix and match versions of different components. Look at the `pom.xml` file for the properties to set from the command line if you want to do this.

5.5 Build Everything

Maven is used to control the build of all of the SlipStream components. To perform the full build, just drop into the "SlipStream" repository that you've checked out and perform the usual maven dance:

```
$ cd SlipStream
$ mvn clean install
```

This command will build and test all of the components. If you wish to skip the tests, you can add the option `-DskipTests` to the maven command line.

5.6 Running

The following instructions describe how to run the SlipStream server locally, primarily for development. If you are interested in running SlipStream in production, you should install it from the generated packages following the instructions in the Administrator Guide.

5.6.1 Databases

SlipStream needs a JDBC friendly database. We use HSQLDB. The SlipStream CIMI server uses Elasticsearch.

5.6.2 Database Configuration

Create an HSQLDB definition file in your home area:

```
$ cat > ~/sqltool.rc << EOF
urlid slipstream
url jdbc:hsqldb:hsql://localhost/slipstream
username SA
password
EOF
```

The default username is "SA" and the default password is blank.

5.6.3 Starting the Database

The HSQLDB database (a pure Java database) will have been downloaded to your local maven repository when you built SlipStream. You can run the database with the downloaded jar file:

```
$ java -cp ~/.m2/repository/org/hsqldb/hsqldb/2.3.4/hsqldb-2.3.4.jar \
       org.hsqldb.server.Server \
       --database.0 file:slipstreamdb \
       --dbname.0 slipstream &
```

Note that starting the database in this way should not be done in production. This is intended only for development testing.

5.6.4 Elasticsearch installation

Follow instructions from Elasticsearch documentation: https://www.elastic.co/guide/en/elasticsearch/reference/current/_installation.html

Please use same version as the one found used SlipStream server (see pom.xml in `SlipStream` project).

To start Elasticsearch engine, just type (in the `bin` directory):

```
$ ./elasticsearch
```

5.6.5 SlipStream Services

SlipStream is composed of a number of services. This section describes how to start these services.

5.6.6 Starting the Ancillary SlipStream Service

This service should be started first and includes additional resources, such as `event` and `usage` resources. As opposed to the main service, which is written in Java, this service is written in Clojure. The service uses both HSQLDB and Elasticsearch, so they should be up and running prior to starting the service.

The service should be started from the `cimi-resources` module of `SlipStreamServer` project.

```
$ cd SlipStreamServer/cimi-resources
```

To run the service, export the required environment variables, start Clojure REPL with lein and in the REPL run the commands listed below:

```
$ lein repl
user=> (do
         (require '[sixsq.slipstream.server.ring-container :as rc])
         (def stop (rc/start 'com.sixsq.slipstream.ssclj.app.server/init 8201)))
```

The services will be started on port `8201`. You can set it as needed, taking into account that it will be required later during the startup of the main SlipStream service. The default port is *rc/default-port*.

It is assumed that an instance of Elasticsearch is running on `localhost:9300`. If this is not the case, export the following environment variables defining the coordinates of Elasticsearch:

```
$ export ES_HOST=<es-host>
$ export ES_PORT=<es-port>
```

The service's log file can be found under `logs/ssclj-.log`.

You can add other dependencies to the classpath as needed. This can be done either by editing the list of dependencies in `project.clj` under `defproject -> :profiles` and adding:

```
:dev      {:resource-paths ["test-resources"]
           :dependencies [[com.sixsq.slipstream/slipstream-ring-container ~+version+]
                          [com.sixsq.slipstream/SlipStreamConnector-OpenStack-conf ~
↪+version+]]}})
```

or providing the dependencies to `lein` command as follows:

```
$ lein update-in :profiles merge \
  '{:dev {:resource-paths ["test-resources"] :dependencies [[com.sixsq.slipstream/
↪SlipStreamConnector-OpenStack-conf]]}}' \
  -- repl
```

By adding connectors jar to the classpath of the service (as shown above) we allow the service to create the connector instances.

5.6.7 Starting Pricing and Ranking Service (PRS)

To start PRS service go to `SlipStreamServer/prs` and run:

```
$ lein run
```

The service starts on `localhost:3000` by default. Logs go to stdout/err.

5.6.8 Starting the Main SlipStream Service

To run the main server, drop into the `war` subdirectory in the `SlipStreamServer` project and then use Jetty to run the SlipStream web archive (war file).

```
$ cd SlipStreamServer/war
$ mvn jetty:run-war -Dorg.eclipse.jetty.annotations.maxWait=120
```

If the last command returns an error like `JettyRunWarMojo : Unsupported major.minor version 51.0` make sure you have Java 8 installed. You can find the appropriate download from the Java web site. You may also want to consult this article for setting up the environment.

As you can see, we run SlipStream as a war behind Jetty. Now that the server's running, visit http://localhost:8080/ with your Web browser.

During development, especially when working on the UI with css and JavaScript files, to avoid the war building round trip, you can start the server pointing to source static location as following:

```
$ export ES_HOST=localhost
$ export ES_PORT=9300
$ mvn jetty:run-war \
      -Dstatic.content.location=file:../../SlipStreamUI/clj/src/slipstream/ui/views
```

The server makes use of Elasticsearch as database backend, therefore, you see the need to set the host and port of Elasticsearch.

To add cloud connectors you need to modify `pom.xml`. Below is an example of adding Exoscale connector that depends on CloudStack connector. Please note that both `jar` and `conf` artifacts should be added.

```
<dependency>
  <groupId>com.sixsq.slipstream</groupId>
  <artifactId>SlipStreamConnector-Exoscale-jar</artifactId>
```

(continues on next page)

(continued from previous page)

```
  <version>${project.version}</version>
</dependency>
<dependency>
  <groupId>com.sixsq.slipstream</groupId>
  <artifactId>SlipStreamConnector-Exoscale-conf</artifactId>
  <version>${project.version}</version>
</dependency>
<dependency>
  <groupId>com.sixsq.slipstream</groupId>
  <artifactId>SlipStreamConnector-CloudStack-jar</artifactId>
  <version>${project.version}</version>
</dependency>
<dependency>
  <groupId>com.sixsq.slipstream</groupId>
  <artifactId>SlipStreamConnector-CloudStack-conf</artifactId>
  <version>${project.version}</version>
</dependency>
```

You are now ready to *configure* your new SlipStream server.

Warning: If you intend to configure your system from configuration files, do not start your service just yet and read on.

5.7 Configuration

To do anything useful with the local SlipStream server, you will need to configure it.

You can either use the web UI that you just started, or use *configuration files*.

5.7.1 User(s)

During the initial startup of the server, an administrator account ("super") will be created. The initial password for this account is "supeRsupeR". You should log in as this user, visit the profile page (single user icon at top), and change the password to another value.

Alternatively, you can change the password on first server startup by *dropping a password file in your configuration directory*.

You can also create new user accounts by visiting the "users" page (*Configuration* -> *Users* from top nav bar), or dropping user configuration files in your configuration directory (see *User Configuration Files*).

5.7.2 Connector(s)

Once the server is up and running you need to configure a connector before trying to deploy a module. Out of the box, using the local connector is the easiest way to get started. To do so, navigate to the server configuration page and define a cloud connector instance in the SlipStream Basics section:

```
test-cloud:local
```

You must be logged in with an administrator account to do this. The value of this field has the form "name1:connector1,name2:connector2"; multiple instances of a single connector are permitted. If the name isn't given, it defaults to the connector name.

For configuration of other cloud connectors, check our blog.

Alternatively, you can create new connector instance by *dropping connector configuration files in your configuration directory*.

5.7.3 Load base images and apps modules

The client module includes examples containing base images and tutorial that can be loaded.

```
$ cd ../../SlipStreamClient/client/src/main/python
$ ./ss-module-upload.py \
      --endpoint http://localhost:8080 \
      -u test -p tesTtesT \
      ../resources/doc/*
```

Change the username and password to an existing (preferably non-administrator) account.

You now only need to configure the cloud parameters of a user (e.g. "test"). And add the cloud IDs to the native images (e.g. Ubuntu, CentOS) you just created.

Building on these base images, you can install apps from the Nuvla™ service.

5.8 Configuration Files

At this point you have a naked SlipStream instance running on your system. This is great, but you probably want a few basic configuration items loaded for your instance to be useable and useful.

The following instructions describe how you can drive the entire configuration of SlipStream by simply dropping a few files on the file system and restarting the service.

The configuration files will only be loaded once, on an empty system. It is therefore safe to restart the SlipStream service with modified configuration files. A specific post request can be invoked to force a reload of the files.

Warning: If your SlipStream server is already started, don't forget to restart it for the configuration files to be loaded.

5.8.1 File System Structure

Initially, SlipStream will look for configuration files in the file system to load. Typically, the loader will look for files in the following locations, stopping at the first occurrence:

```
/etc/slipstream/
~/.slipstream/
```

Inside this structure of the expected configuration is as follows:

Configuration file	Meaning
`./slipstream.conf`	Main configuration file
`./connectors/*.conf`	Connector specific configuration files
`./modules/*.xml`	Module definition files
`./cloud-ids/*.conf`	Unique cloud image identifiers
`./users/*.xml`	User definition files
`./passwords/*`	User passwords

Note that these files are not created by default during a standard installation.

5.8.2 Main Configuration File

The main configuration file is normally placed in /etc/slipstream when performing a standard installation. The default values are taken from the system default shipped with SlipStream.

If you want to override these defaults, a good place to start is from the system default file, that you can modify and drop in one of the standard location (see previous section).

Warning: Do not forget to rename the default file to `slipstream.conf`.

5.8.3 Connector Configuration Files

Each connector can be configured by simply dropping a configuration file in the `./connectors/` directory. While the same can be achieved using the main configuration file, we recommend you split your connector configuration in separate files. This method is also more configuration management tool friendly, such as Puppet, Chef or Ansible.

The connector configuration file must include the `cloud.connector.class` and the name given to the connector instance must be used to qualify all other configuration parameter. Here is an example of a connector file to configure the Exoscale cloud:

```
cloud.connector.class = exoscale-ch-gva:cloudstack
exoscale-ch-gva.endpoint = https://api.exoscale.ch/compute
exoscale-ch-gva.zone = CH-GV2
exoscale-ch-gva.quota.vm = 20
exoscale-ch-gva.orchestrator.imageid = 605d1ad4-b3b2-4b60-af99-843c7b8278f8
exoscale-ch-gva.orchestrator.instance.type = Micro
exoscale-ch-gva.native-contextualization = linux-only
```

Warning: Do not forget to have a `cloud.connector.class` key in each file.

You can have as many connector configuration files as you wish.

Warning: If you are using the enterprise edition, makes sure you have the license matching the number of connector instance configured.

5.8.4 Module Configuration Files

To load automatically module definitions (i.e. projects, images and deployments), simply drop module files in the `./modules` configuration directory. These files (now in xml and soon in json) can be retrieved from a live SlipStream instance by simply GETting the module or appending `?media=xml` to the end of a URL, either using your browser, curl or any other http tool.

Loading Apps from Nuvla™

Nuvla™, a SlipStream service managed by SixSq, contains a number of interesting apps you can take advantage of. You can download these following a simple procedure. Once you have the xml files, simply drop them in the `modules/` directory and restart your SlipStream service.

Warning: Modules will only be loaded from file if no modules already exist.

More information about loading apps is available.

5.8.5 Unique Cloud Identifier Configuration Files

As you build a full SlipStream system configuration, you will also need to provide unique cloud identifiers, especially for base images. Typically, these files will be cloud service specific. Here is an example of a unique cloud identifier file for Exoscale:

```
apps/Minecraft/minecraft-server = exoscale-ch-gva:aaabbbccc
```

where the key is the module to set, and the value is composed of a `:` separated tuple, composed of the cloud service name and the unique cloud identifier.

Warning: Ensure the cloud service name part of the value matches the cloud service name defined via the `cloud.connector.class` configuration parameter.

5.8.6 User Configuration Files

The last piece of configuration required to have a fully functional system is one or several users. By default, SlipStream will create a privileged user during the first startup of the service. But you can add more users to the system dropping files in the `./users/` configuration directory.

As for modules (see above), user configuration files can be created GETting existing users.

For security reasons, since user passwords are never transmitted over the wire once the user is registered, a second set of configuration file is required to define the user password. Only a hashed version of the password is kept in the database.

5.8.7 User Password Configuration File

As mentioned above, the password of each user must be defined using separate files, located in the `./passwords/` directory. The file name must match the username, without any extension. The file must only contain the password in clear text.

Here is an example of the content of the password file for the user `test`:

```
$ cat ./passwords/test
Change_Me
```

You can also, using such a password file change the password of the `super` user created automatically during the first execution of the service. However, ensure that the file is in place before the first execution, or that no user exist in the database.

Warning: Ensure each password file has the exact same name as the user it corresponds to, without any extension.

5.9 Contributing

SixSq appreciates bug reports and feedback. Please provide feedback through GitHub issues or via the SixSq Support website.

SixSq also welcomes contributions to the SlipStream code base. However, to ensure that the intellectual properties rights are well-defined, we require that all contributors sign a "Contributor's Agreement" before their code will be accepted. Contact us if you are interested in contributing code.

CHAPTER 6

Legal Information

6.1 Licenses

6.1.1 Software Licenses

The Community Edition of SlipStream is released under the Apache 2.0 license. The source code can be found within the GitHub “slipstream” organization.

The Enterprise Edition of SlipStream is released only under a paid, proprietary license.

6.1.2 Documentation License

The SlipStream documentation is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

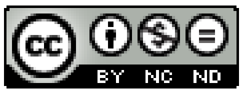


Fig. 1: by-nc-nd

6.2 Terms of Service

This document contains the Terms of Service and Privacy Policies associated with the SlipStream service.

BY CHECKING THE “I AGREE” BUTTON DURING REGISTRATION AND/OR ACCESSING THIS SERVICE, YOU ACKNOWLEDGE THAT YOU HAVE READ, UNDERSTOOD AND AGREE TO BE BOUND BY THESE TERMS OF SERVICE. IF YOU DO NOT AGREE TO ALL OF THESE TERMS OF SERVICE YOU WILL NOT BE GIVEN ACCESS TO USE THE SERVICE.

6.2.1 Description of Service

SixSq Sarl (“SixSq”), hosts this website (the “Site”) on its SlipStream environment (the “Software”). The Software and all information, communications, documentation, scripting, images, graphics and other materials and services found on the Site (collectively, the “Service”) is for the use of registered developers, employees and other authorized users of the Site, but only for the purpose of *software development, integration, deployment and testing*.

6.2.2 Grant of License to Use

Subject to the terms and conditions set forth in these Terms of Service (“TOS”), SixSq grant user (“You”) an individual, personal, non-exclusive, non-transferable, and revocable license (the “License”) to use the Service offered through the Site, including the Software. Upon termination of these TOS for any reason, the License will terminate, and You will cease to have access to the Service. Any rights not expressly granted herein by SixSq are reserved.

You acknowledge and agree that You will not, directly or indirectly: (i) reverse engineer, decompile, disassemble or otherwise attempt to discover the underlying source code or underlying ideas or algorithms of the Software; (ii) modify, translate, or create derivative works based on the Software; (iii) rent, lease, distribute, sell, resell or assign, or otherwise transfer rights to the Software; or (iv) remove any proprietary notices in the Software.

You acknowledge and agree that the Software includes the valuable proprietary and trade secret information and property of SixSq or its licensors. Title, ownership rights and intellectual property rights, including but not limited to, copyright and patent rights in the Software shall remain in SixSq and/or its licensors. You acknowledge the ownership and intellectual property rights of SixSq and will not take any action to jeopardize, limit or interfere in any manner with SixSq’s or its licensors’ ownership of or rights with respect to the Software. The Software is protected by copyright law and other intellectual property laws and by international treaties.

6.2.3 Other Licenses

This Site may contain other license(s) applicable to content on this Site (“Other License(s)”). You agree to abide by the terms of any such Other License(s). In the event of any conflict between certain provisions of the Other License(s) and the TOS, the applicable provisions of the Other License(s) shall prevail.

6.2.4 Prohibited Conduct

You may only use the Site and the Service offered through the Site in accordance with SixSq use policy (“Use Policy”), as set forth in this section. You are responsible for the content of any comments, information, questions, data, computer code or programs, feedback, ideas, descriptions of processes or other information submitted through the Site by You. You may only submit to the Site content that is (i) owned by You, (ii) submitted with the express permission of the owner, or (iii) in the public domain. In addition, all material posted to the Site by You must be relevant to the specified software development project and must be in good taste as determined by common sense and the most restrictive community standards. The Site may not be used to:

- upload, post or otherwise transmit any data or content that is unlawful, harmful, threatening, abusive, harassing, tortious, defamatory, vulgar, obscene, pornographic, libelous, invasive of another’s privacy, hateful, or racially, ethnically or otherwise objectionable;
- harm minors in any way;
- impersonate any person or entity, including, but not limited to, any SixSq official, forum leader, guide or host, or falsely state or otherwise misrepresent your affiliation with a person or entity;
- upload, post or otherwise transmit any content that You do not have a right to transmit under any law or under contractual or fiduciary relationships (such as inside information, proprietary and confidential information learned or disclosed as part of employment relationships or under nondisclosure agreements);

- upload, post or otherwise transmit any content that violates any law or violates the personal, proprietary, or intellectual property rights or other rights of any third party;
- upload, post or otherwise transmit any unsolicited or unauthorized advertising, promotional materials, "junk mail," "spam," "chain letters," "pyramid schemes," or any other form of solicitation;
- upload, post or otherwise transmit any material that contains software viruses or any other computer code, files or programs designed to interrupt, destroy or limit the functionality of any computer software or hardware or telecommunications equipment;
- interfere with or disrupt the Site or servers or networks connected to the Site, or disobey any requirements, procedures, policies or regulations of networks connected to the Site;
- "stalk" or otherwise harass another;
- collect or store personal data about other users; or
- promote or provide instructional information about illegal activities, promote physical harm or injury against any group or individual. This may include, but is not limited to, providing instructions on how to assemble bombs and other weapons.

SixSq does not prescreen content that is posted to the Site; however, SixSq reserves the right to remove any content that fails to meet the foregoing Use Policy and to revoke your account. SixSq further reserves the right to investigate complaints or reported violations of this Use Policy, and to take any appropriate action, including, without limitation, reporting and providing information of any suspected unlawful activity to law enforcement officials, regulators, or other third parties, including disclosing any information necessary or appropriate to such person or entities relating to user profiles, e-mail addresses, usage history, posted materials, IP addresses and traffic information.

To comment on this Use Policy or to report material or conduct that You believe is inappropriate, please contact feedback@sixsq.com.

6.2.5 Compliance with Laws

You agree to comply with all applicable laws and regulations of the Switzerland and foreign authorities, including, without limitation, laws regarding the transmission of technical data, such as encryption, exported from Switzerland or from the country in which You reside. You agree not to export, re-export or import the Software or any direct product thereof in violation of any such restrictions, laws or regulations, or without all necessary authorizations.

6.2.6 Registration Data and Privacy

Registration is required for you to use the Service. You must provide certain current, complete, and accurate information about yourself as prompted to do so by the registration form ("Registration Data"), and maintain and update such registration information as required to keep such information current, complete and accurate. You warrant that the Registration Data is accurate and current, and that You are authorized to provide such Registration Data. You authorize us to verify your Registration Data at any time. If any Registration Data that You provide is untrue, inaccurate, not current or incomplete, SixSq retain the right, in their sole discretion, to suspend or terminate your rights to use the Service.

6.2.7 Passwords, Cloud Credentials and Security

At the time of registration, You must provide the Site with your full first and last name, the name of your organization, if any, your username and your valid e-mail address. You are responsible for maintaining the confidentiality of the password and account, and are fully responsible for all activities that occur under your password or account. Further, you might be required to provide cloud credentials for using Third Party Cloud Providers (see Section 9) the Site uses (e.g. Amazon Web Service, StratusLab) as back-end. You agree to (i) immediately notify SixSq of any unauthorized use of your password or account or cloud credentials or any other breach of security, and (ii) ensure that You exit from your account at the end of each session. SixSq cannot and will not be liable for any loss or damage arising from your failure to comply with this Section 7.

6.2.8 Trademarks

SixSq and the SlipStream logo are trademarks or registered trademarks of SixSq Sarl (the "SixSq Marks") in Europe and other countries. You agree not to display or use the SixSq Marks in any manner without SixSq's prior written permission.

6.2.9 Notice and Procedure for Making Claims of Copyright Infringement

SixSq is committed to respecting the intellectual property rights of others, and we ask You do the same. SixSq may, in their sole discretion, terminate the accounts or access rights of users who violate the intellectual property rights of others. If you believe that your work has been copied in a way that constitutes copyright infringement on our Site, please deliver the following information to SixSq' Copyright Agent:

- An electronic or physical signature of the person authorized to act on behalf of the copyright owner;
- A description of the copyrighted work that You claim has been infringed;
- A description of where the material that You claim is infringing is located on the Site;
- Your address, telephone number and e-mail address so that we can contact you;
- A statement by You that you have a good faith belief that the disputed use is not authorized by the copyright owner, its agent or the law; and
- A statement by You, made under penalty of perjury, that the information in your notice to us is accurate and that you are the copyright owner or authorized to act on the copyright owner's behalf.

The Copyright Agent can be contacted by postal mail at SixSq Sarl Rue du Bois-du-Lan, 1217, Meyrin, Geneva, Switzerland or by email at copyright@sixsq.com.

6.2.10 Third Party Cloud Providers

The Site may use third party cloud service providers (collectively, "Third Party Cloud Provider(s)"). SixSq is not responsible for the availability of such other Third Party Cloud Providers and do not endorse and are not responsible or liable for any content, products or other materials available on such Third Party Cloud Providers. The usage by the Site of Third Party Cloud Providers do not constitute an endorsement by SixSq of such Third Party Cloud Providers or the sponsors of such Third Party Cloud Providers or the content, products, advertising or other materials presented on such Third Party Cloud Providers. Users of the Site must comply with the Terms of Use of the Third Party Cloud Providers they wish to use. By providing the Site with Third Party Cloud Providers credentials, You explicitly state that You will comply with the Terms of Use and that you are authorized to use these Third Party Cloud Credentials by the owner of these credentials. SixSq cannot be held responsible in any way, shape, or form if the Third Party Cloud Provider credentials are misused, stolen or altered following providing these credentials to the Site. You acknowledge and agree that SixSq shall not be responsible or liable, directly or indirectly, for any damage or loss caused or alleged to be caused by or in connection with use of or reliance on any information, goods or services available on or through any such Third Party Cloud Providers. If You decide to provide the Site with Third Party Cloud credentials and use these Third Party Cloud Providers, You do so entirely at your own risk.

6.2.11 Disclaimer of Warranties

THE SERVICE IS PROVIDED ON AN “AS IS” AND “AS AVAILABLE” BASIS FROM SIXSQ. SIXSQ MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH REGARDS TO THE SERVICE. TO THE FULLEST EXTENT PERMISSIBLE BY APPLICABLE LAW, SIXSQ DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND/OR NONINFRINGMENT. YOU ASSUME THE ENTIRE RISK OF SELECTION AND USE OF THE SERVICE AVAILABLE AT THE SITE. WITHOUT LIMITING THE FOREGOING, SIXSQ MAKES NO WARRANTY THAT (i) THE SERVICE WILL MEET YOUR REQUIREMENTS, (ii) THE SERVICE WILL BE UNINTERRUPTED, TIMELY, SECURE, OR ERROR-FREE, (iii) THE RESULTS THAT MAY BE OBTAINED FROM THE USE OF THE SERVICE WILL BE ACCURATE OR RELIABLE, OR (iv) THE CONTENT OR INFORMATION AVAILABLE ON THE SITE IS COMPLETE, ACCURATE OR AVAILABLE.

NO ADVICE OR INFORMATION OBTAINED BY YOU FROM SIXSQ OR THROUGH THE SITE SHALL CREATE ANY WARRANTY NOT EXPRESSLY MADE HEREIN.

6.2.12 Limitation of Liability

SIXSQ AND THEIR RESPECTIVE SHAREHOLDERS, AFFILIATES, EMPLOYEES, AGENTS, SUCCESSORS, OFFICERS AND ASSIGNS SHALL NOT BE LIABLE FOR ANY LOSS, INCLUDING, WITHOUT LIMITATION, LOSS OF BUSINESS, LOSS OF USE OR OF DATA, INTERRUPTION OF BUSINESS, LOST PROFITS OR GOODWILL, OR OTHER INDIRECT, SPECIAL, INCIDENTAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES OF ANY KIND ARISING OUT OF YOUR USE OF THE SERVICE, EVEN IF THEY HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSS AND WHETHER OR NOT THEY HAD ANY KNOWLEDGE, ACTUAL OR CONSTRUCTIVE, THAT SUCH DAMAGES MIGHT BE INCURRED. THIS EXCLUSION INCLUDES ANY LIABILITY THAT MAY ARISE OUT OF THIRD-PARTY CLAIMS AGAINST YOU.

YOUR SOLE AND EXCLUSIVE REMEDY FOR ANY DISPUTE WITH SIXSQ IN CONNECTION WITH THE SERVICE IS TO CEASE YOUR USE OF THE SERVICE.

6.2.13 Exclusions and Limitations

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF CERTAIN WARRANTIES OR THE EXCLUSION OR LIMITATION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES. ACCORDINGLY, SOME OF THE ABOVE LIMITATIONS OF SECTIONS 12 AND 13 MAY NOT APPLY TO YOU.

6.2.14 Indemnity

You shall indemnify, defend, and hold harmless SixSq and their respective shareholders, affiliates, employees, agents, successors, officers, and assigns, from any suits, losses, claims, demands, liabilities, costs and expenses (including attorney and accounting fees) that they may sustain or incur as a result of your use of the Service.

6.2.15 Termination

You agree that SixSq may, at their sole discretion, deny You access to the Site and disable any user name and password associated with You for any reason, including, without limitation, if SixSq believe that You have violated or acted inconsistently with the terms and conditions of the TOS. SixSq reserves the right at any time and from time to time to modify or discontinue, temporarily or permanently, the Service offered through the Site (or any part thereof) with or without notice. You agree that SixSq shall not be liable to You or to any third party for any modification, suspension or discontinuance of the Service.

6.2.16 Notice

Notices to You may be made via e-mail. SixSq may also provide notices to You by displaying notices or links to notices on the Site.

6.2.17 General Provisions

Governing Law, Jurisdiction and Venue.
The Site (excluding any linked Third Party Sites) is controlled by SixSq from the Canton of Geneva, Switzerland. The parties agree that the laws of the Canton of Geneva, Switzerland, without regard to the conflicts of laws principles thereof, shall govern all matters relating to access to, or use of, the Site and the Service available through the Site. The parties agree that any action brought by the parties to enforce or interpret any provision of the TOS shall be brought exclusively in an appropriate court in the Canton of Geneva, Switzerland. The parties hereby consent to such jurisdiction and waive any objection to such venue. SixSq make no representation that the Site is appropriate or available for use in other locations, and accessing the Site from territories where its contents are illegal is prohibited. Those who choose to access the Site from other locations do so upon their own initiative and are responsible for compliance with local laws.

Force Majeure.
Except with regard to the payment of money, no party shall be responsible for any delays caused by acts of God or any other cause beyond its reasonable control, including but not limited to such things as strikes, riots, acts of war or terrorism, restricting legislation, embargo, blockage, work stoppage, major outage of a public communications carrier, etc. Any delay caused by one party hereto which affects any other party's ability to perform according to the terms of the TOS shall extend the non-delaying party's obligation to perform by the same number of days by which the delaying party delayed in performing its obligations.

Severability / Waiver.
Each provision of the TOS is severable; if any provision is declared void, illegal, or unenforceable, such provision shall be modified by the appropriate judicial body to the minimum extent necessary to render it valid, legal, and enforceable while effectuating insofar as possible the basic purposes of such provision. The remaining provisions shall remain in full force and effect. No delay, omission, or failure to exercise any right or remedy provided for in the TOS shall be deemed to be a waiver thereof or an acquiescence in the event giving rise to such remedy (but every such right or remedy may be exercised as the party exercising such right or remedy deems expedient).

Entire Agreement / Survival.
The TOS constitute the entire agreement between SixSq, on the one hand, and You, on the other, with respect to use of the Service and supersedes all prior agreements or previous discussions (written or oral) between the parties. The TOS have been written in the English language and, in the event of any conflict or inconsistency between the English-language version and any translation hereof, the English language version shall prevail. The TOS may be modified by SixSq by giving You the opportunity to electronically review and agree to the new TOS. SixSq also will post the new TOS on the Site in the same location the previous TOS was posted. Any provision in the TOS that may reasonably be interpreted as being intended by the parties to survive the termination or expiration of the TOS shall survive any such termination or expiration.

6.3 Trademarks

"SlipStream", "SixSq", "Nuvla", "NuvlaBox", the SlipStream logo, SixSq logo, Nuvla logo and NuvlaBox logo are trademarks or registered trademarks of SixSq Sarl in Switzerland and other countries.

The SlipStream trademark and logo can be used only with unmodified copies of the SlipStream software. All other trademarks and logos can be used only with explicit written permission from SixSq Sarl.

6.4 Copyright

Copyright (c) 2018, SixSq Sarl.

SixSq Sarl is the copyright holder for both the SlipStream software and documentation. Use of the software and documentation are subject to the defined license terms.

CHAPTER 7

Reference

Reference documentation for the SlipStream service and its associated APIs and libraries.

7.1 Application Programming Interface

SlipStream provides an HTTP-based, RESTful Application Programming Interface (API). The details of this interface can be found in the SlipStream API documentation web site.

7.2 Libraries

Although SlipStream's RESTful API can be consumed directly, it is sometimes easier to use a library that wraps the RESTful interface to allow for more idomatic use from a given programming language.

7.2.1 Clojure(Script)

A complete Clojure API is provided. This library provides full-featured synchronous and asynchronous clients. The asynchronous client is compatible with ClojureScript and can be used in JavaScript runtime environments.

The Clojure API source code is in GitHub and the Clojure API documentation is hosted in GitHub Pages.

7.2.2 Python

A complete Python API is provided. This native library provided a full coverage of the SlipStream features. The Python API source code is hosted in GitHub with the Python API documentation in GitHub Pages.

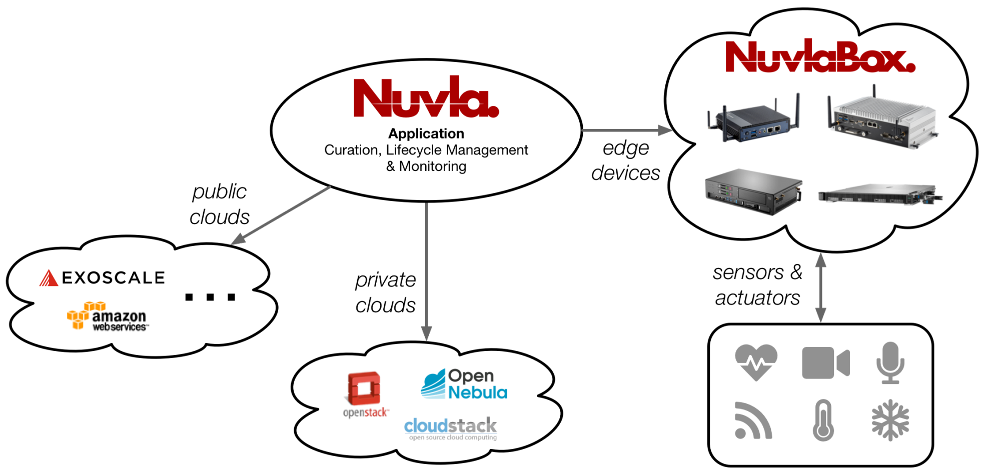
There is also a Libcloud Driver for SlipStream that supports the basic cloud application management features. As for the other libraries, the Libcloud driver source code is in GitHub with the Libcloud Driver Documentation in GitHub Pages.

CHAPTER 8

NuvlaCity

NuvlaCity provides an **edge computing** platform that transparently integrates a wide variety of cloud infrastructures and edge devices.

- **Nuvla**, a SlipStream service operated by SixSq, allows users to define, manage, and monitor their applications from a single point.
- **NuvlaBox** machines (with varying capacities) provide edge computing capabilities near data sources and sensors.



This demonstration covers the core elements and functionality of the NuvlaCity platform. In particular, it shows how you can:

- Setup your account on **Nuvla** and manage applications.
- Initialize a **NuvlaBox** and activate it through Nuvla.
- Manage and monitor applications running with **virtual machines**.
- Manage and monitor **Docker** (container-based) applications.
- Deploy and use a container orchestration engine, specifically **Docker Swarm**.

- Update the state of running applications, with **Anisible** or other similar systems.

To make the demonstration as useful as possible, it uses concrete container and configuration management technologies. Keep in mind however, that **NuvlaCity is a generic platform that allows a wide variety of technologies to be used**.

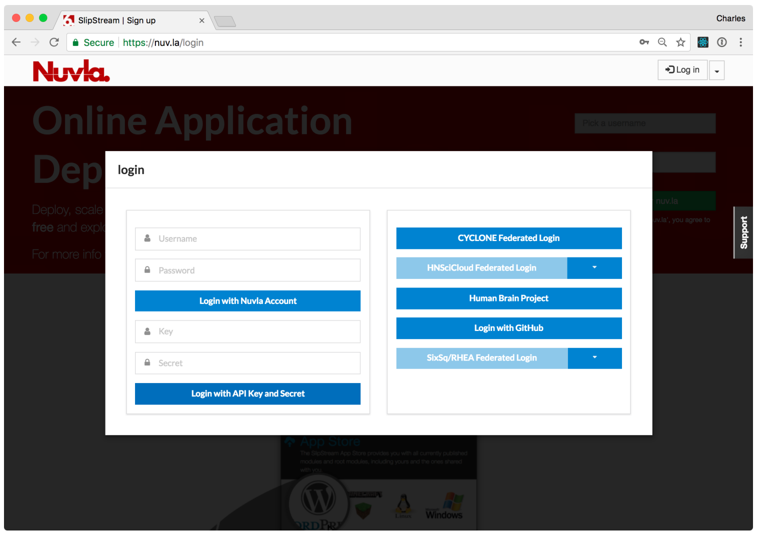
8.1 Nuvla

Nuvla, a SlipStream service operated by SixSq, allows you to curate your cloud applications, manage their lifecycle, and monitor them. It provides a convenient web browser interface that gives you an overview of your applications and resources. It also provides a complete REST API to allow integration with other systems (but which is not covered in this demonstration).

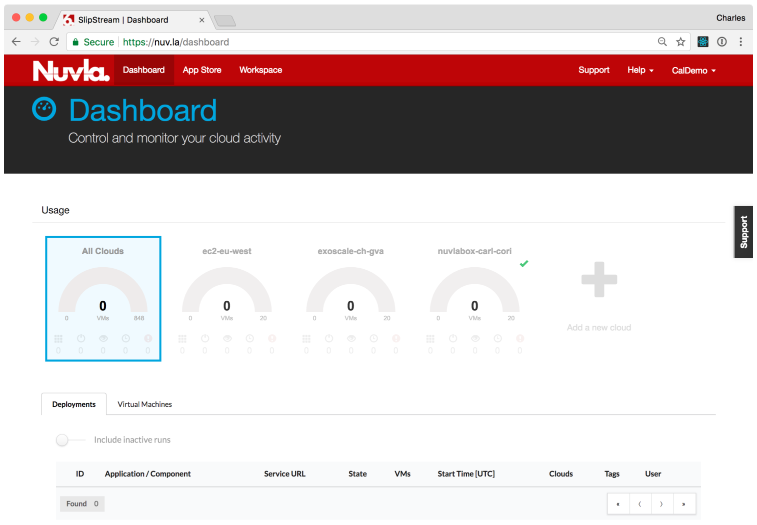
8.1.1 Account

You will have been either asked to register for an account or given an account already created by Nuvla administrators. To log into your account:

- Visit the Nuvla service at https://nuv.la,
- Click on the `Log in` button in the upper-right corner,
- Fill in your username and password,
- Click on the `Login with Nuvla Account`.



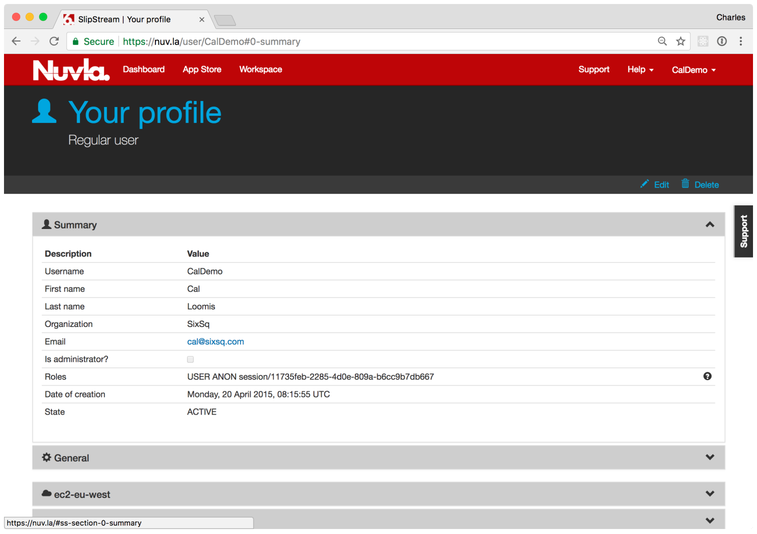
The first time you log in, you will be redirected to the App Store and offered a tour. Click one of the buttons to ignore the tour for now. On subsequent logins, you will be redirected to the Dashboard. Click on the Dashboard menu item at the top to see your Dashboard.



On the Dashboard, you will see NuvlaBox edge devices and clouds that you can access and an empty list of deployments. The user in the screenshot has access to one NuvlaBox device (nuvlabox-carl-cori), one region in the Exoscale cloud, and one region in the AWS cloud.

8.1.2 User Profile

Before proceeding, you need to setup or verify a few values in your profile. Open the menu item in the upper right corner labelled with your username and then click on the "Profile" item. You should then see a page similar to the following screenshot.



Description	Value
Username	CalDemo
First name	Cal
Last name	Loomis
Organization	SixSq
Email	cal@sixsq.com
Is administrator?	☐
Roles	USER ANON session/11735feb-2285-4d0e-809a-b6cc9b7db667
Date of creation	Monday, 20 April 2015, 08:15:55 UTC
State	ACTIVE

Click on the `Edit` action and open the "General" section by clicking on the header. We will change three values:

- **Default Cloud**: Change the value to your NuvlaBox device (or one of your NuvlaBox devices). This will be the default computing resource when deploying applications.
- **Keep running after deployment**: Verify that this is set to "on success". This will keep deployments running when then succeed.
- **SSH Public Key(s)**: Add your OpenSSL-formated, public SSH key. This allows you to log into deployed machines via SSH.

After the changes, click on the `Save` action.

8.1.3 Summary

With the described actions, you should now have a Nuvla account that is ready to be used for the rest of the demonstration. Explore the various aspects of the Nuvla user interface before preparing the NuvlaBox.

8.2 NuvlaBox

The NuvlaBox is a secure and intelligent edge computing solution that integrates seamlessly with Nuvla, which provides management and a unified view of your edge resources. Hardware from Hewlett Packard Enterprise (HPE) and Logic Supply are certified for the NuvlaBox software.

This demonstration shows how to install the NuvlaBox software stack on certified HPE hardware and then to activate the NuvlaBox to make it accessible from Nuvla.

8.2.1 Network Access

A NuvlaBox can be installed without a network connection, but it must have a connection to be activated.

The Internet connection must be provided via a physical ethernet cable connected to the NuvlaBox machine's primary network interface. For certified hardware with multiple ports, SixSq support can tell you which ethernet port to use on your machine for the Internet connection.

DHCP

Your local network must be configured to provide an IPv4 address via DHCP to the NuvlaBox machine. Static IP address assignment is recommended.

Outbound Connectivity

It is strongly recommended that your firewall be configured to permit **unrestricted outbound access** from the assigned IP address. As a strict minimum, the firewall must allow access to port 443 (HTTPS) on nuv.la; other ports may be required depending on which inbound connectivity scenario is chosen.

Inbound Connectivity

In many cases, inbound access to applications running on a NuvlaBox is required. The NuvlaBox can be configured in several ways to provide this type of access.

- By default, the NuvlaBox will provide direct access to applications running on the NuvlaBox to clients that are attached to the NuvlaBox's **local network**, either via a physical cable or via WiFi (if available). The NuvlaBox bridges access to the Internet for client connected to the local network. **This is the network configuration used in this demonstration.**
- Nuvla and the NuvlaBox can be configured to **proxy ports to the nuvlabox.com domain**. This makes services running on those ports globally available to the Internet, although the number of available ports is limited. This solution currently relies on SSH tunneling between Nuvla and the NuvlaBox, but will use the VPN (described below) in the future.

- The NuvlaBox can be integrated with the organization's **corporate network**. This allows full access to services running on the NuvlaBox from clients within the corporate network. This requires specific configuration of the NuvlaBox and the corporate network's DHCP server.
- Finally, the NuvlaBox can be integrated with a **Virtual Private Network (VPN)**. This allows full access to services running on the NuvlaBox from other clients within the same VPN, but credentials for accessing the VPN must be managed.

You are invited to discuss with SixSq engineers which solution best meets the requirements of your NuvlaBox deployment.

8.2.2 Installation

Note: If you order NuvlaBox machines directly through SixSq, the NuvlaBox software stack will already have been installed on the machines. If this case, you can skip ahead to the activation section.

Note: If your NuvlaBox is connected to the Internet during the installation process, then it will automatically perform **both** the installation and activation.

The installation and configuration of the NuvlaBox hardware is accomplished by booting from an installation USB stick.

USB Stick

First, find a USB stick to use for the installation:

- Obtain a USB stick with at least 2 GB of space. Using a USB 3 stick will allow faster installation, but USB 2 will work as well.
- **The stick will be entirely reformatted.** Do not use a stick with data you want to save.
- Attach the USB stick to the laptop (or other machine) that you will use to initialize the stick.

Copy Installation Image

Second, download and copy the NuvlaBox installation image to your USB stick.

- Ask SixSq support for the URL to download the latest NuvlaBox USB image.
- Using this URL, download the image to a laptop where you have attached the USB stick.
- Determine the USB device for your stick.
- Perform a raw copy of the image to your USB stick.

On a Linux or Mac OS machine, the copy can be done with the `dd` command. On a Windows machine, there are a number of tools that can be used to create a bootable USB stick from an image. Use your favorite!

Boot from USB

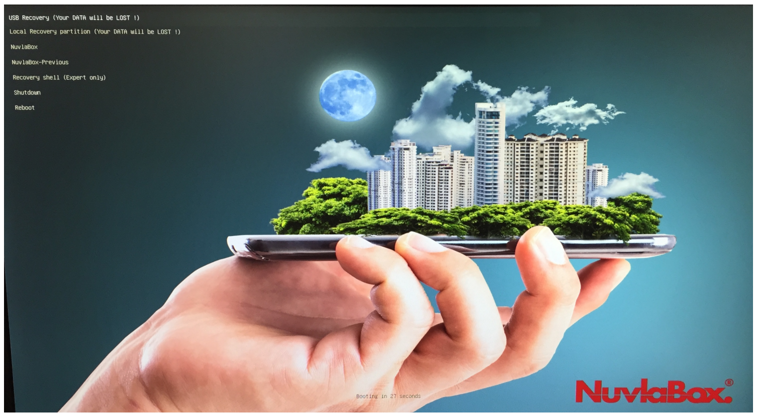
Third, boot the target machine from the USB stick:

- Plug the USB stick into target machine.

- Turn on the machine.
- Enter into the BIOS settings, on HPE machines this is usually accomplished by pressing the `F10` key while the machine boots.
- In the BIOS, navigate to the boot options and force the machine to boot from the USB stick.
- **Do not change the default boot device to the USB stick.**
- Exit the BIOS and allow the machine to start from the USB stick.

Installation Type

Fourth, select the installation type. The option you want is “USB Recovery” for a fresh installation. This is the default.


USB Recovery (Your DATA will be LOST !)
Local Recovery partition (Your DATA will be LOST !)
NuvlaBox
NuvlaBox-Previous
Recovery shell (Expert only)
Shutdown
Reboot
Booting in 21 seconds
NuvlaBox

The machine will then boot several times; when the process is finished, remove the USB key.

8.2.3 Activation

A NuvlaBox must be “activated” before it can be used as a computing resource from Nuvla. The one-time registration activates a “NuvlaBox connector” resource on Nuvla that is identified by the machine’s MAC address.

NuvlaBox Connectors

If you have purchased:

- NuvlaBox hardware directly from SixSq, NuvlaBox connectors with the correct MAC addresses will already have been created and assigned to your Nuvla account(s).
- Hardware from other suppliers, then you must provide SixSq with the MAC addresses of the machine(s) you will activate. SixSq will then create the necessary NuvlaBox connectors.

The activation process will not succeed until the associated NuvlaBox connector is available on Nuvla.

Note: This activation can only be done once; further attempts to activate the same MAC address will fail. This prevents someone else from impersonating your NuvlaBox with another machine.

Note: If a machine is compromised, you can quarantine the NuvlaBox connector on Nuvla. This will deactivate the NuvlaBox and prevent any new workloads, data, etc. being sent to the machine. A quarantined NuvlaBox can only be recovered with help from SixSq support.

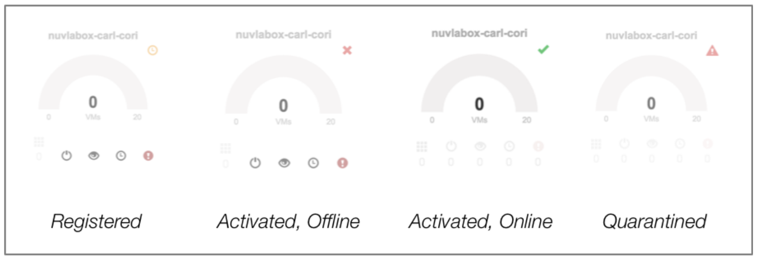
Reboot to Activate

To start the activation process, simply reboot the machine once the machine has network access. You can watch the progress from the machine's console and see the activation status on the Nuvla Dashboard.

Note: As a reminder, if network access was available during the installation process, the machine will initiate the activation process without a reboot.

The Nuvla dashboard indicates the status of a NuvlaBox machine with an icon in the upper-right corner of the tile:

- For a NuvlaBox that has been **registered** with Nuvla (that is, the NuvlaBox connector with the correct MAC address is present), the dashboard will show the machine with a **yellow clock icon**.
- When the activation process is complete, it will initially show the NuvlaBox to be **offline**. This is indicated by a **red cross**.
- After a final reboot, the machine will be **online**, indicated by a **green checkmark**.



Once it is online, applications can be started on the NuvlaBox machine. Like for all resources, the Dashboard will show the state of those applications and provide the means to control them.

8.2.4 Summary

In this section, you have seen how to install the NuvlaBox software stack on a compatible machine and activate the machine so that it can be used through Nuvla. The following sections provide more details on using the NuvlaBox to run your applications.

8.3 Virtual Machine Management

The NuvlaCity ecosystem allows you to define and to deploy cloud applications across public clouds, private clouds, and NuvlaBox edge devices. Operators can deploy new application instances in a few clicks and developers can create new applications with a minimum of effort.

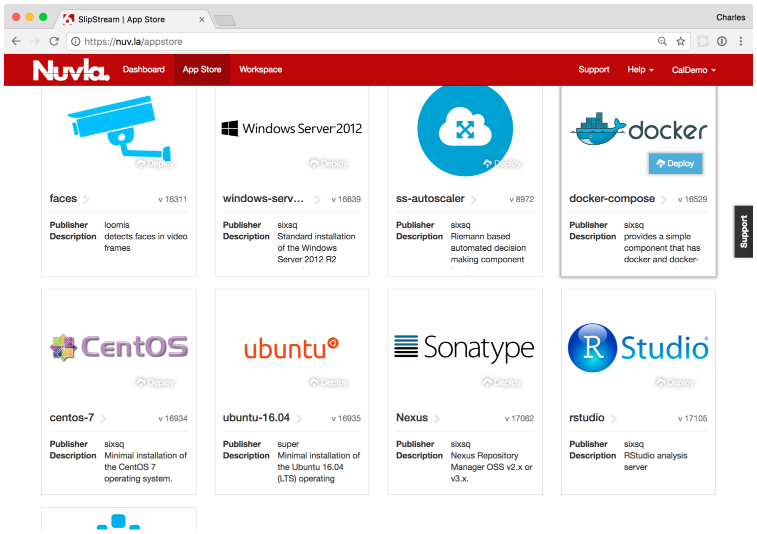
8.3.1 Minimal Operating Systems

SixSq maintains SlipStream components on Nuvla that reference base operating system images that are (relatively) consistent across all supported cloud infrastuctures. The CentOS 7 and Ubuntu 16.04 operating systems are used most heavily and consequently, are embedded in the NuvlaBox machines by default.

When you are logged into Nuvla, you can find these "native components" in the Workspace:

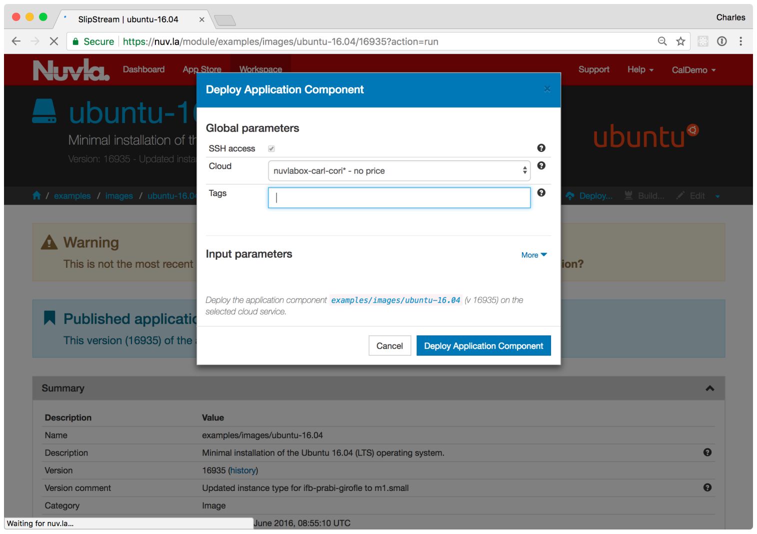
- examples/images/centos-7
- example/images/ubuntu-16.04

and in the Nuvla App Store.



faces v 16311	windows-serv... v 16639	ss-autoscaler v 8972	docker-compose v 16529
Publisher loomis	Publisher sixsq	Publisher sixsq	Publisher sixsq
Description detects faces in video frames	Description Standard installation of the Windows Server 2012 R2	Description Riemann based automated decision making component	Description provides a simple component that has docker and docker-
centos-7 v 16934	ubuntu-16.04 v 16935	Nexus v 17062	rstudio v 17105
Publisher sixsq	Publisher super	Publisher sixsq	Publisher sixsq
Description Minimal installation of the CentOS 7 operating system.	Description Minimal installation of the Ubuntu 16.04 (LTS) operating	Description Nexus Repository Manager OSS v2.x or v3.x.	Description RStudio analysis server

New virtual machines based on these components can be provisioned by clicking on the `Deploy` action on the component page in the Workspace or on the tile's `Deploy` button in the App Store. Both will bring up a deployment dialog.


SlipStream | ubuntu-16.04
Charles
Secure
https://nuv.la/module/examples/images/ubuntu-16.04/16935?action=run
Nuvla
Dashboard
App Store
Workspace
Support
Help
CalDemo
ubuntu-1
Minimal installation of th
ubuntu
Deploy...
Build...
Edit
Warning
This is not the most recent
ion?
Published applicati
This version (16935) of the
Waiting for nuv.la...

Deploy Application Component

Global parameters

SSH access ✓

Cloud: nuvlabox-carl-cori* - no price

Tags:

Input parameters — More

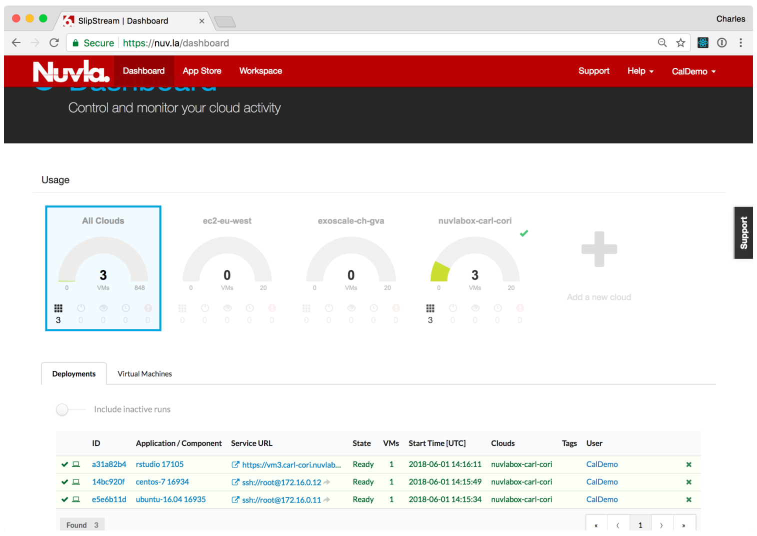
Deploy the application component `examples/images/ubuntu-16.04` *(v 16935) on the selected cloud service.*

Cancel | Deploy Application Component

Summary

Description	Value
Name	examples/images/ubuntu-16.04
Description	Minimal installation of the Ubuntu 16.04 (LTS) operating system.
Version	16935 (history)
Version comment	Updated instance type for ifb-prabi-girofle to m1.small
Category	Image
	June 2016, 08:55:10 UTC

From here, just select the target infrastructure, then click the `Deploy Application Component` button. This will take you to the Dashboard, where you can follow the progress of the deployments.



	ID	Application / Component	Service URL	State	VMs	Start Time [UTC]	Clouds	Tags	User	
✓	a31a82b4	rstudio 17105	https://vm3.carl-cori.nuvlab...	Ready	1	2018-06-01 14:16:11	nuvlabox-carl-cori		CalDemo	✖
✓	14bc920f	centos-7 16934	ssh://root@172.16.0.12	Ready	1	2018-06-01 14:15:49	nuvlabox-carl-cori		CalDemo	✖
✓	e5e6b11d	ubuntu-16.04 16935	ssh://root@172.16.0.11	Ready	1	2018-06-01 14:15:34	nuvlabox-carl-cori		CalDemo	✖

You can do the same for the CentOS 7 image as well. Detailed deployment information can be found by clicking on the deployment "ID" link.

Once these are in the ready state, you can log into these machines with SSH (provided you've configured your account with your public SSH key). Just click on the SSH link or use the IP address (with the 'root' account) from a terminal.

As the administrator of these machines, you can install and configure any services you need for your application.

When you're finished with them, you can free up resources by terminating the machines. Either click the "X" icon for the deployment on the Dashboard or click the `Terminate` action on the deployment page.

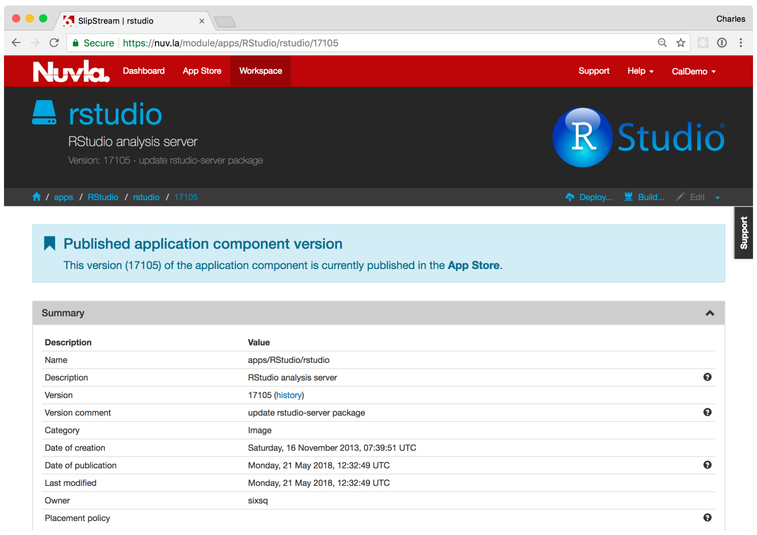
8.3.2 Preconfigured Applications

Although you have complete freedom with the minimal operating system images, you will normally be running a small number of pre-defined services. Application developers can create specialized SlipStream components (single VM services) or applications (multiple VM services) to automate the deployment. This reduces the time to deploy the services, as well as eliminating mistakes from manual steps.

There are many examples of pre-defined components and applications in the App Store. Here we will demonstrate RStudio, a web-browser interface to the R statistical analysis program.

Find the RStudio tile in the App Store and click the `Deploy` button to start it. The procedure for RStudio is the same as for CentOS 7 or Ubuntu 16.04.

While RStudio is being deployed, you may want to review how the RStudio component was created. This information can be found on the RStudio component definition page.



rstudio

RStudio analysis server

Version: 17105 - update rstudio-server package

Published application component version

This version (17105) of the application component is currently published in the **App Store**.

Summary

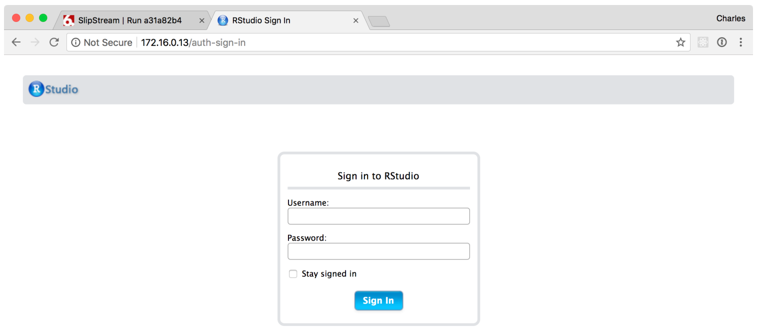
Description	Value
Name	apps/RStudio/rstudio
Description	RStudio analysis server
Version	17105 (history)
Version comment	update rstudio-server package
Category	Image
Date of creation	Saturday, 16 November 2013, 07:39:51 UTC
Date of publication	Monday, 21 May 2018, 12:32:49 UTC
Last modified	Monday, 21 May 2018, 12:32:49 UTC
Owner	sixsq
Placement policy	

The information for the component is divided into sections. Click on any of the section headers to see the details:

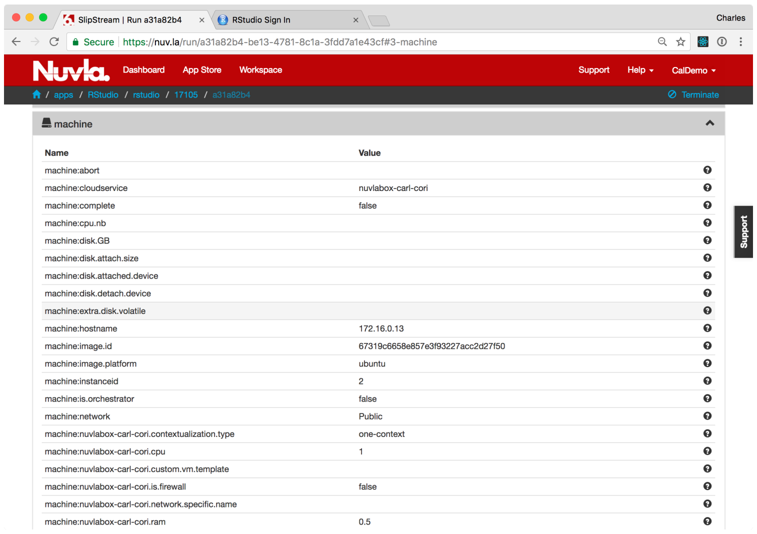
- The **Summary** section provides general information about the component. From here, you can also find the entire history of the component.
- The **Cloud Image Identifiers and Image Hierarchy** section will show you the parent image. For RStudio this is the Ubuntu 16.04 image that we were using previously.
- The **Application Parameters** section shows the input and output parameters for the component. The output parameter of most interest for RStudio is the generated user password for the deployment.
- The **Application Workflows** section shows the actions that are performed at each stage of the application lifecycle. You can look at these scripts to understand how RStudio is configured and installed, as well as how it is integrated with the SlipStream parameter database.

The other sections also contain a few details of the deployment.

Once the deployment is ready, you can click on the provided link to access the RStudio application. You will immediately see that the page requires a username and password.

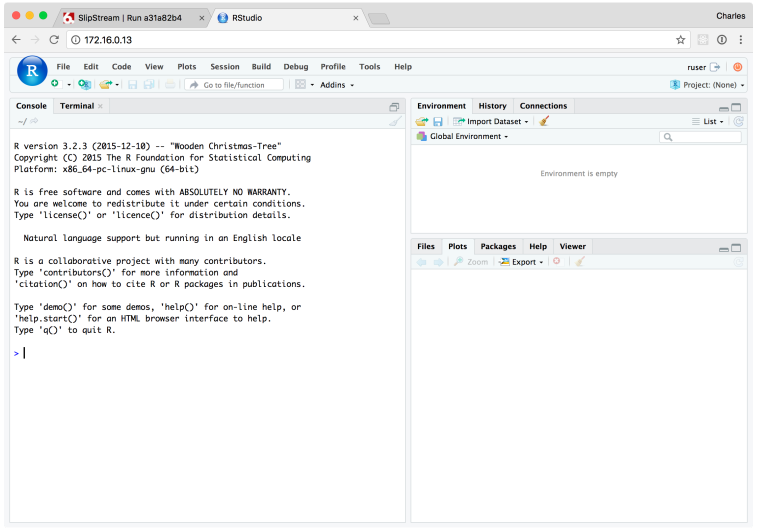


This information is communicated to you via the component's output parameters. These can be found on the deployment page in the "Machine" section.


SlipStream | Run a31a82b4
RStudio Sign In
Charles
Secure https://nuv.la/run/a31a82b4-be13-4781-8c1a-3fdd7a1e43cf#3-machine
Nuvla
Dashboard
App Store
Workspace
Support
Help
CalDemo
apps / RStudio / rstudio / 17105 / a31a82b4
Terminate
machine
Support

Name	Value
machine:abort	
machine:cloudservice	nuvlabox-carl-cori
machine:complete	false
machine:cpu.nb	
machine:disk.GB	
machine:disk.attach.size	
machine:disk.attached.device	
machine:disk.detach.device	
machine:extra.disk.volatile	
machine:hostname	172.16.0.13
machine:image.id	67319c6658e857e3f93227acc2d27f50
machine:image.platform	ubuntu
machine:instanceid	2
machine:is.orchestrator	false
machine:network	Public
machine:nuvlabox-carl-cori.contextualization.type	one-context
machine:nuvlabox-carl-cori.cpu	1
machine:nuvlabox-carl-cori.custom.vm.template	
machine:nuvlabox-carl-cori.is.firewall	false
machine:nuvlabox-carl-cori.network.specific.name	
machine:nuvlabox-carl-cori.ram	0.5

With the provided username and password, you can log into the RStudio console. You should see a dashboard similar to the following screenshot.



```
R version 3.2.3 (2015-12-10) -- "Wooden Christmas-Tree"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

In the "Console" tab on the left you can type `demo(graphics)` to see a demonstration of the RStudio graphics capabilities. Just keep hitting return to advance through the demo.

When you are finished, you can terminate the RStudio application with the `Terminate` action.

8.3.3 Summary

SlipStream allows you to define both simple and complex cloud applications. Once defined, you can manage the lifecycle of those applications easily. We will see an example of a complex application when we deploy Docker Swarm later.

8.4 Container Management

Docker and Docker Hub have popularized containers as a way to conveniently package and distribute applications. In addition, they start up rapidly and consume less resources than hypervisor-based virtual environments.

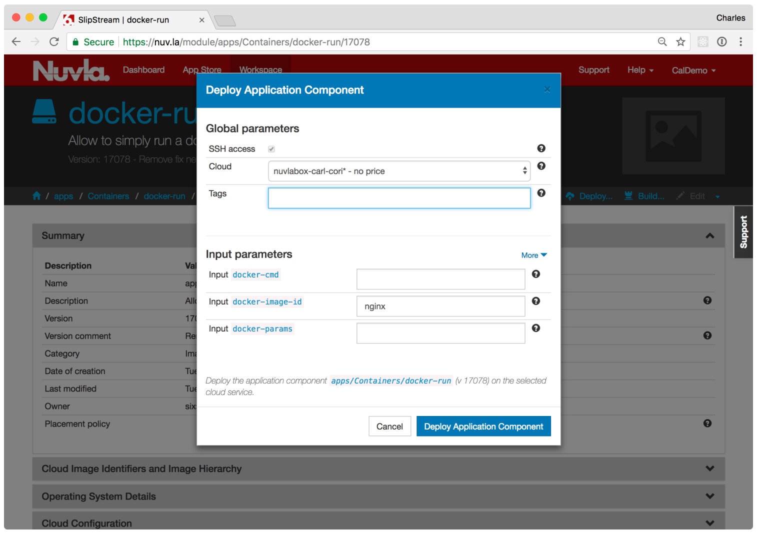
NuvlaCity targets container technologies to allow users to take advantage of their benefits, especially on resource-constrained hardware that is common in edge-computing environments. Although planned, NuvlaCity does not yet treat containers as first-class entities, instead deploying containers within virtual machines. This technique allows containers to be exploited now with NuvlaCity.

8.4.1 Single Container Deployment

Nuvla contains an application component called docker-run that allows you to deploy a named container from the Docker Hub. To deploy this component:

- Navigate to the docker-run module,
- Click on the "Deploy…" action,
- Choose the NuvlaBox to target,
- Select the container to run,
- Provide the command and parameters (if any), and
- Click the "Deploy Application Component" button.

By default, the component will run the nginx web server. The component exposes all ports from the container on the host virtual machine. In the case of nginx, the web server will be visible on port 80. The dialog you see should be similar to the following screenshot.



Deploy Application Component

Global parameters

SSH access	✓
Cloud	nuvlabox-carl-cori* - no price
Tags	

Input parameters

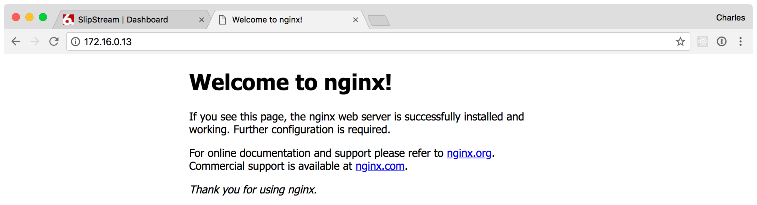
Input `docker-cmd`	
Input `docker-image-id`	nginx
Input `docker-params`	

Deploy the application component `apps/Containers/docker-run` *(v 17078) on the selected cloud service.*

Cancel | Deploy Application Component

You can follow the progress of the deployment either from the Dashboard or from the deployment detail page (by clicking on the deployment ID link).

Once the deployment reaches the "Ready" state, you can browse to port 80 on the deployed machine to verify that the nginx welcome page appears.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

From the deployment dialog you can choose any other image from the Docker Hub by specifying its name and providing any necessary command and/or parameters.

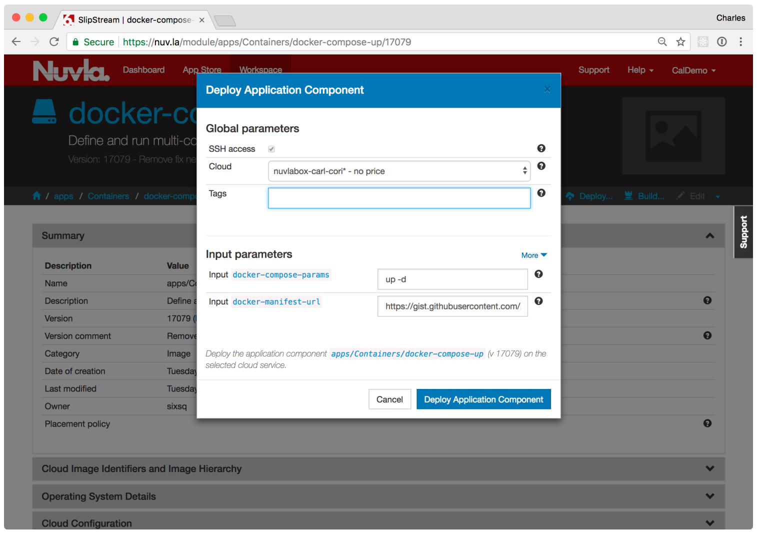
8.4.2 Docker Compose Deployment

Many times an application consists of a number of services working together. For example, a Wordpress deployment needs to have a MySQL database backing it. Within the Docker ecosystem, composite applications like this are defined by Docker Compose files. The `docker-compose` command can be used to start/stop the entire ensemble.

Nuvla contains an application component called docker-compose-up that makes deployments of Docker Compose applications easy. Similar to what was done previously:

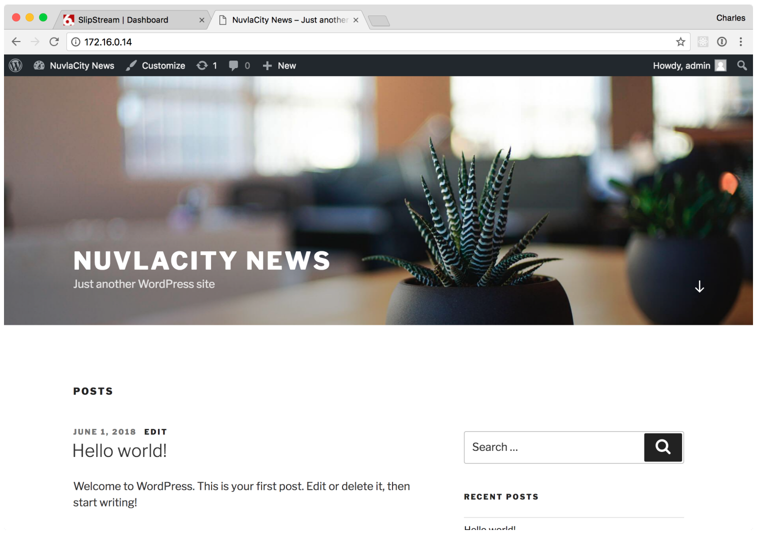
- Navigate to the docker-compose-up module,
- Click on the “Deploy…” action,
- Choose the NuvlaBox to target,
- Provide the URL for the Docker Compose manifest,
- Provide the `docker-compose` parameters, and
- Click the “Deploy Application Component” button.

The dialog you see should be similar to the following screenshot.


Deploy Application Component
Global parameters
SSH access ✓
Cloud: nuvlabox-carl-cori* - no price
Tags
Input parameters
More
Input docker-compose-params: up -d
Input docker-manifest-url: https://gist.githubusercontent.com/
Deploy the application component apps/Containers/docker-compose-up (v 17079) on the selected cloud service.
Cancel
Deploy Application Component

The default URL references a Docker Compose manifest that runs an instance of Wordpress with a backing MySQL database. As before, the component exposes all ports from the container on the host virtual machine. In this case, the Wordpress instance will be visible on port 80.

After the deployment reaches the "Ready" state, navigate to port 80 on this machine with your web browser. You will have to do the initial configuration of the Wordpress application. Afterwards, you can click on the "View Site" link from the Wordpress dashboard to see the list of blogs. The welcome page of Wordpress should look like the following screenshot.


172.16.0.14
NUVLACITY NEWS
Just another WordPress site
POSTS
JUNE 1, 2018 EDIT
Hello world!
Welcome to WordPress. This is your first post. Edit or delete it, then start writing!
Search ...
RECENT POSTS

8.4.3 Summary

This section has shown you how to run containerized applications on the NuvlaCity platform. Tighter integration of containers will be offered in the future, but already containers can be an effective mechanism for deploying applications on the NuvlaCity platform.

Support for private repositories or other customizations can be done easily by copying the existing docker-run and docker-compose-up components and modifying them to suit your needs.

8.5 Container Orchestration

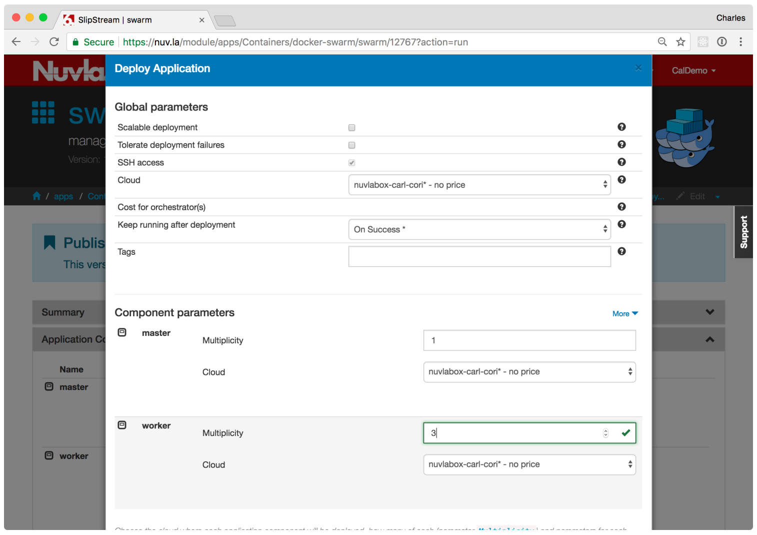
For production, container-based services, you will often want to take advantage of a container orchestration engine to provide automated fail-over, load balancing, replication, and other high-level features. The most popular container orchestration engines are Kubernetes, Mesos, and Docker Swarm.

8.5.1 Docker Swarm

For resource-constrained environments like the NuvlaBox, we prefer to use Docker Swarm as a container orchestration engine. It is simpler to manage, more lightweight, and compatible with the standard Docker API.

Similar to the SlipStream components you have seen so far, Docker Swarm is available in the Nuvla App Store. Unlike the other examples however, this will be the first that involves multiple virtual machines. A Docker Swarm deployment consists of one master node and one or more worker nodes. Despite the additional complexity, a Docker Swarm cluster can be deployed as easily as the other examples.

Find the Docker Swarm tile in the App Store and click the `Deploy` button to bring up the deployment dialog. You will see that the dialog is more complex than before.


SlipStream | swarm
Charles
Secure https://nuv.la/module/apps/Containers/docker-swarm/swarm/12767?action=run
CalDemo
Support
Summary
Name

Deploy Application

Global parameters

Scalable deployment	☐
Tolerate deployment failures	☐
SSH access	✓
Cloud	nuvlabox-carl-cori* - no price
Cost for orchestrator(s)	
Keep running after deployment	On Success *
Tags	

Component parameters

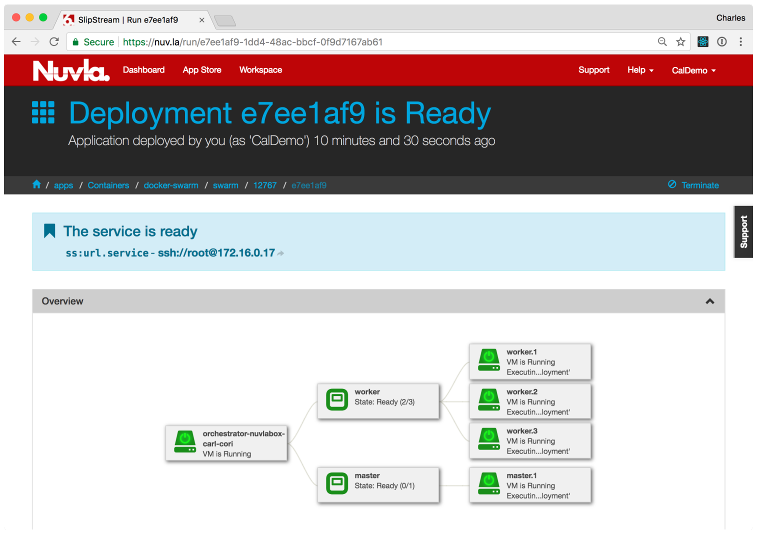
master	Multiplicity	1
	Cloud	nuvlabox-carl-cori* - no price
worker	Multiplicity	3
	Cloud	nuvlabox-carl-cori* - no price

On the dialog, you see a general section, where you can choose the target infrastructure for all nodes. (You can also choose different infrastructures for different nodes, although we will not demonstrate that here.)

Then there is a section for each node of the deployment. Here there are two: “master” and “worker”. By default, you will have one master and one worker. You can increase the number of workers.

Once you have set the target infrastructure and the node multiplicities, click the `Deploy Application` button to start the deployment. As usual this will take you to the Dashboard where you can follow the deployment process.

The deployment detail page will show the topology of the deployment and provide the input/output parameter values for all of the deployed nodes.


Deployment e7ee1af9 is Ready

Application deployed by you (as 'CalDemo') 10 minutes and 30 seconds ago

The service is ready

ss:url.service - ssh://root@172.16.0.17

Note that the SlipStream application has handled the deployment of all of the virtual machines and coordinated the configuration of each node.

8.5.2 Replicated Service

The features provided by Docker Swarm for managing "services" are vast, so the SlipStream component does not try to provide a parameterized interface to the system. To manage services on the deployed Docker Swarm cluster, log into the master node via SSH. You can find the IP address on the deployment detail page.

When you have logged in, you can check that the number of nodes agrees with the number you requested:

```
$ docker node ls

ID                           HOSTNAME                                      STATUS    ␣
↪           AVAILABILITY       MANAGER STATUS      ENGINE VERSION
cax3ifys3ixoogt0025k1o8bc *  master1aab3929a-9167-4e73-8f65-29e33376b759   Ready     ␣
↪           Active             Leader              18.03.1-ce
wwzmkb2q61o4r0ooe4wu3ug6q    worker1aab3929a-9167-4e73-8f65-29e33376b759   Ready     ␣
↪           Active                                 18.03.1-ce
ic67gr0k1v0z5cin0fxvlcsd3    worker2aab3929a-9167-4e73-8f65-29e33376b759   Ready     ␣
↪           Active                                 18.03.1-ce
awd7cdf05i37xq4wmv4kubyb7    worker3aab3929a-9167-4e73-8f65-29e33376b759   Ready     ␣
↪           Active                                 18.03.1-ce
```

We asked for 1 master and 3 workers. They are all present and accounted for.

Now let's use the Swarm to deploy a replicated nginx service. This command will run 3 replicas of nginx and make the service available on port 8080 from any node of the cluster:

```
$ docker service create --name nuvlacity_swarm \
                        --replicas 3 \
                        --publish published=8080,target=80 \
                        nginx

72ecrbhuy8p8o3h6jpmtn7v8c
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
```

(continues on next page)

(continued from previous page)

```
3/3: running   [==================================================>]
verify: Service converged
```

You can then check that the service is running with the correct configuration:

```
$ docker service ls

ID                  NAME                MODE                REPLICAS            IMAGE␣
↪               PORTS
72ecrbhuy8p8        nuvlacity_swarm     replicated          3/3                ␣
↪nginx:latest        *:8080->80/tcp
```

The nginx server should respond from any node within the cluster. You can get the IP address of the nodes through Nuvla or via Docker:

```
$ docker node inspect -f '{{ .Status.Addr  }}' wwzmkb2q61o4r0ooe4wu3ug6q
159.100.241.48
```

Using `curl` to get the page content:

```
swarm-master# curl 159.100.241.48:8080

<!DOCTYPE html>
...
<h1>Welcome to nginx!</h1>
...
</html>
```

You can verify that this works from all nodes of the Docker Swarm cluster.

8.5.3 Summary

With Nuvla, you can deploy a full container orchestration engine easily, allowing you to take advantage of the replication, load balancing, and other features. Tighter integration of SlipStream with Docker Swarm is in the development roadmap.

8.6 Configuration Management

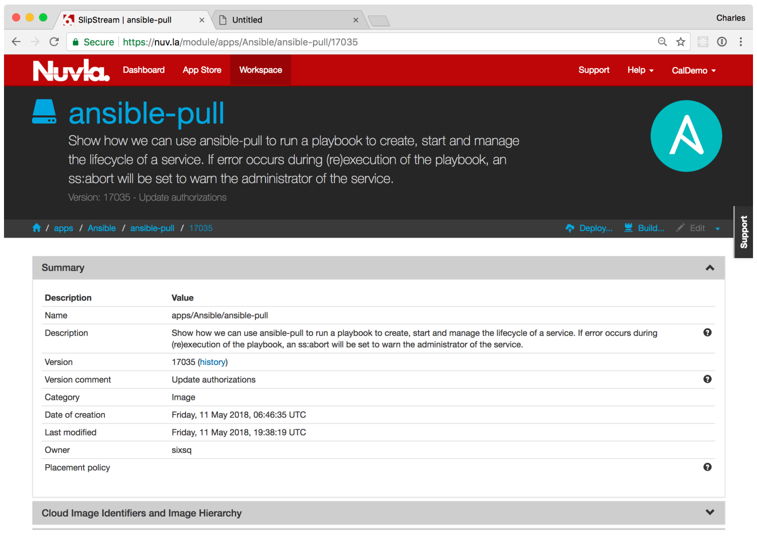
Although cloud-based systems encourage the upgrades via redeployment of services, many scenarios still call for the use of traditional configuration management systems. These systems, for example, may minimize network traffic to the edge during the update cycle or may be deeply embedded in your business' operations processes.

Ansible, Puppet, Chef, Salt, and Quattor are just a few of the available configuration management systems. Given the large number of these systems, NuvlaCity, by design, remains independent of them, allowing you to leverage the knowledge in your existing configuration management platform, whatever it is.

To demonstrate how configuration management systems can be integrated with the NuvlaCity management workflows, we will walk through a simple deployment that uses Ansible.

8.6.1 Ansible Component

A generic Ansible Pull SlipStream component has been developed to show how easily configuration management systems can be integrated with the NuvlaCity platform. The example builds on Ubuntu 16.04, installs git and ansible, and uses a short (40 line) shell script to integrate with NuvlaCity.


SlipStream | ansible-pull
Untitled
Charles
Secure
https://nuv.la/module/apps/Ansible/ansible-pull/17035
Dashboard
App Store
Workspace
Support
Help
CalDemo
apps / Ansible / ansible-pull / 17035
Deploy...
Build...
Edit
Support

ansible-pull

Show how we can use ansible-pull to run a playbook to create, start and manage the lifecycle of a service. If error occurs during (re)execution of the playbook, an ss:abort will be set to warn the administrator of the service.

Version: 17035 - Update authorizations

Summary

Description	Value
Name	apps/Ansible/ansible-pull
Description	Show how we can use ansible-pull to run a playbook to create, start and manage the lifecycle of a service. If error occurs during (re)execution of the playbook, an ss:abort will be set to warn the administrator of the service.
Version	17035 (history)
Version comment	Update authorizations
Category	Image
Date of creation	Friday, 11 May 2018, 06:46:35 UTC
Last modified	Friday, 11 May 2018, 19:38:19 UTC
Owner	sixsq
Placement policy	

Cloud Image Identifiers and Image Hierarchy

8.6.2 Ansible Repository

As the example runs in "pull" mode, a repository of Ansible playbooks must be available. We use a public GitHub repository, ansible-example, for the demonstration.

8.6.3 Scenario Overview

During the demonstration, we will perform the following actions:

- Deployment of the "ansible pull" component,
- Verification that the deployment starts a machine with the lighttpd web server,
- Make an invalid modification to the playbook and verify that Nuvla raises the abort flag,
- Correct the playbook and verify that the abort flag is cleared,
- Switch from lighttpd to nginx for the web server,
- Verify that the nginx server is now available on the machine.

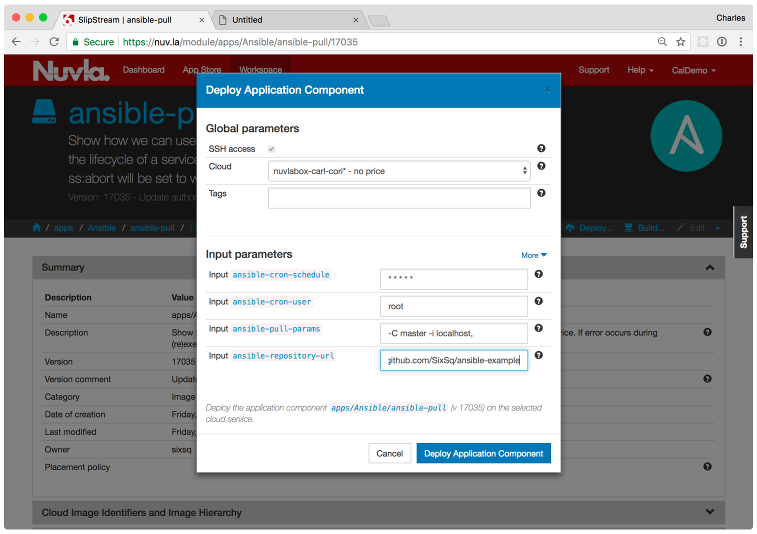
The demonstration provides a complete overview of how configuration management systems can be easily integrated with NuvlaCity.

8.6.4 Deploy Ansible Pull

When logged into your account on Nuvla:

- Navigate to the Ansible Pull component and
- Click "Deploy…".

You should see a deployment dialog similar to the following screenshot.



Deploy Application Component

Global parameters

SSH access ✓

Cloud: nuvlabox-carl-cori* - no price

Tags:

Input parameters

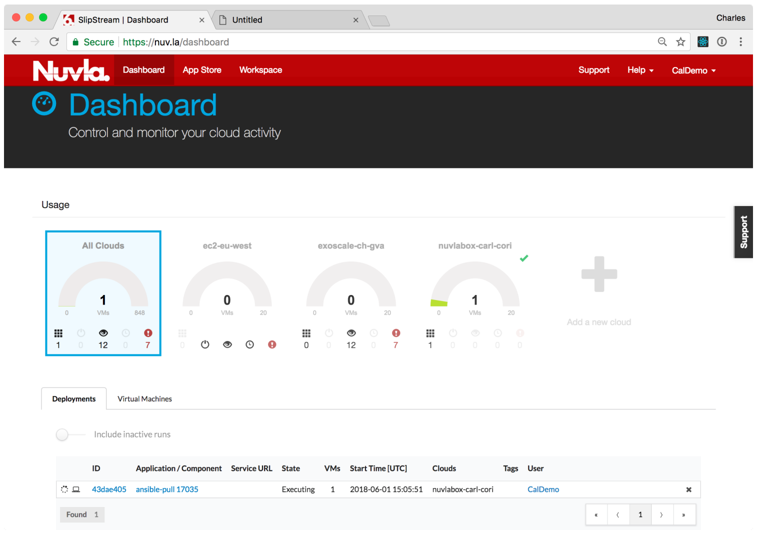
Input `ansible-cron-schedule`	* * * * *
Input `ansible-cron-user`	root
Input `ansible-pull-params`	-C master -i localhost,
Input `ansible-repository-url`	github.com/SixSq/ansible-example

Deploy the application component `apps/Ansible/ansible-pull` (v 17035) on the selected cloud service.

On the deployment dialog:

- Choose the target NuvlaBox,
- Adjust the cron schedule to every minute,
- Provide the URL of a Git repository that contains the demonstration playbook and **that you can update**, and
- Click "Deploy Application Component".

This will take you to the deployment page.

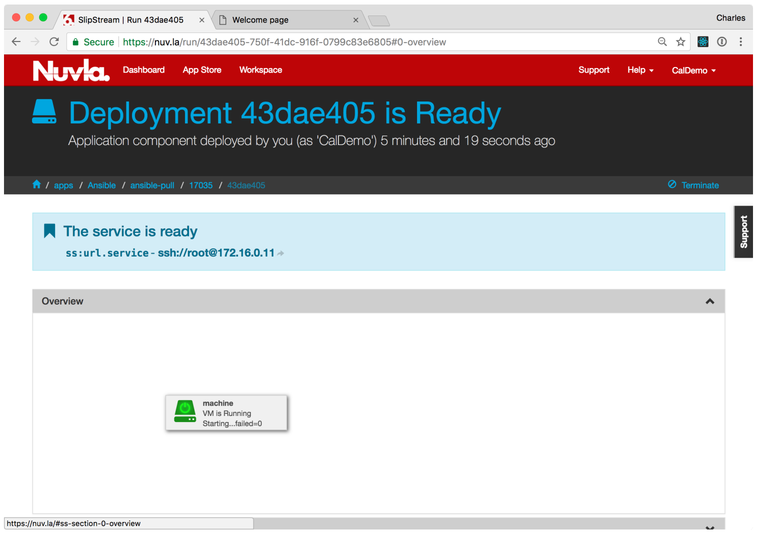


Dashboard

Control and monitor your cloud activity

ID	Application / Component	Service URL	State	VMs	Start Time [UTC]	Clouds	Tags	User
43dae405	ansible-pull 17035		Executing	1	2018-06-01 15:05:51	nuvlabox-carl-cori		CalDemo

Click on the deployment ID to see the details of the deployment. Wait for this component to start on the NuvlaBox that you have chosen.



Deployment 43dae405 is Ready

Application component deployed by you (as 'CalDemo') 5 minutes and 19 seconds ago

apps / Ansible / ansible-pull / 17035 / 43dae405

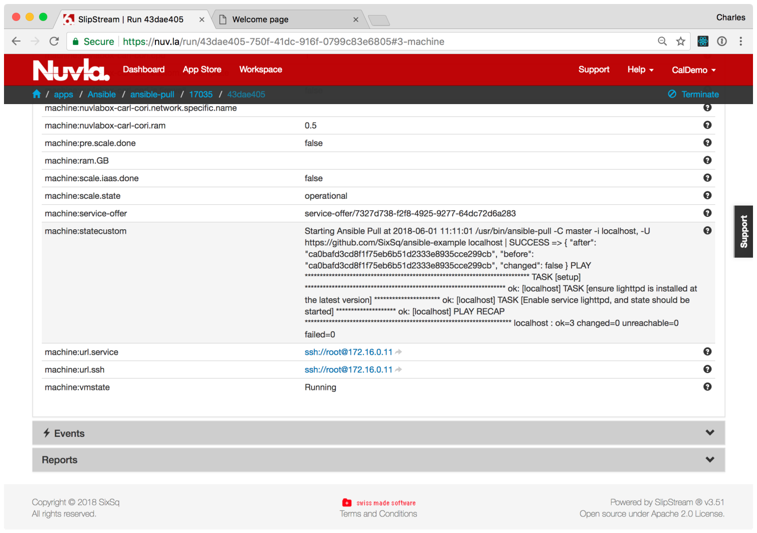
The service is ready

`ss:url.service` - ssh://root@172.16.0.11

8.6.5 Verify Correct Deployment

If the deployment worked correctly, the deployment should have ended in a "Ready" state and a successful Ansible update result should be available from the `machine:statecustom` parameter.

To see the ansible result, open the "Machine" section by clicking on the header. Look for the `machine:statecustom` parameter. The value should be similar to the one in the following screenshot.

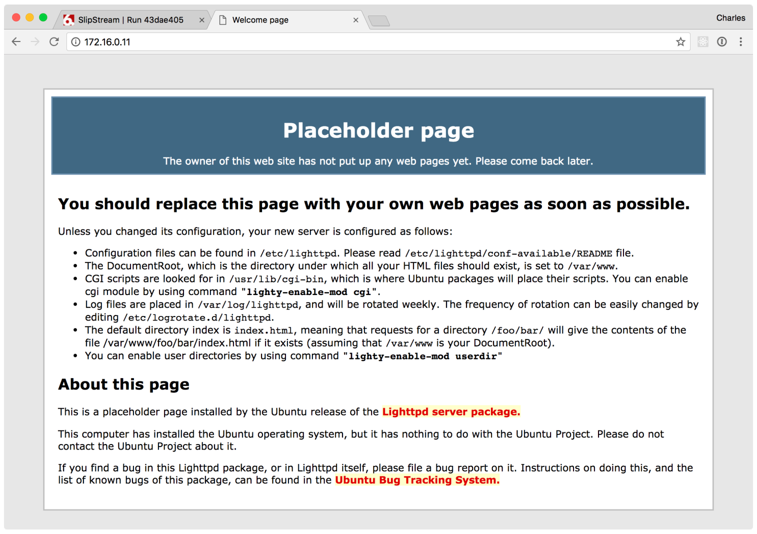


machine:nuvlabox-carl-cori.network.specific.name	
machine:nuvlabox-carl-cori.ram	0.5
machine:pre.scale.done	false
machine:ram.GB	
machine:scale.iaas.done	false
machine:scale.state	operational
machine:service-offer	service-offer/7327d738-f2f8-4925-9277-64dc72d6a283
machine:statecustom	Starting Ansible Pull at 2018-06-01 11:11:01 /usr/bin/ansible-pull -C master -i localhost, -U https://github.com/SixSq/ansible-example localhost | SUCCESS => { "after": "ca0bafd3cd8f1f75eb6b51d2333e8935cce299cb", "before": "ca0bafd3cd8f1f75eb6b51d2333e8935cce299cb", "changed": false } PLAY *** TASK [setup] *** ok: [localhost] TASK [ensure lighttpd is installed at the latest version] ********************* ok: [localhost] TASK [Enable service lighttpd, and state should be started] ******************* ok: [localhost] PLAY RECAP *** localhost : ok=3 changed=0 unreachable=0 failed=0
machine:url.service	ssh://root@172.16.0.11
machine:url.ssh	ssh://root@172.16.0.11
machine:vmstate	Running

SlipStream | Run 43dae405
Welcome page
Charles
Secure
https://nuv.la/run/43dae405-750f-41dc-916f-0799c83e6805#3-machine
Dashboard
App Store
Workspace
Support
Help
CalDemo
apps / Ansible / ansible-pull / 17035 / 43dae405
Terminate
Support
Events
Reports
Copyright © 2018 SixSq
All rights reserved.
swiss made software
Terms and Conditions
Powered by SlipStream ® v3.51
Open source under Apache 2.0 License.

The message should contain "SUCCESS" and the Git hash values related to the change.

The playbook should have started the lighttpd HTTP server on port 80. Browse to the deployed machine, to verify that you see the lighttpd welcome page.



Placeholder page

The owner of this web site has not put up any web pages yet. Please come back later.

You should replace this page with your own web pages as soon as possible.

Unless you changed its configuration, your new server is configured as follows:

- Configuration files can be found in `/etc/lighttpd`. Please read `/etc/lighttpd/conf-available/README` file.
- The DocumentRoot, which is the directory under which all your HTML files should exist, is set to `/var/www`.
- CGI scripts are looked for in `/usr/lib/cgi-bin`, which is where Ubuntu packages will place their scripts. You can enable cgi module by using command "**lighty-enable-mod cgi**".
- Log files are placed in `/var/log/lighttpd`, and will be rotated weekly. The frequency of rotation can be easily changed by editing `/etc/logrotate.d/lighttpd`.
- The default directory index is `index.html`, meaning that requests for a directory `/foo/bar/` will give the contents of the file /var/www/foo/bar/index.html if it exists (assuming that `/var/www` is your DocumentRoot).
- You can enable user directories by using command "**lighty-enable-mod userdir**"

About this page

This is a placeholder page installed by the Ubuntu release of the **Lighttpd server package.**

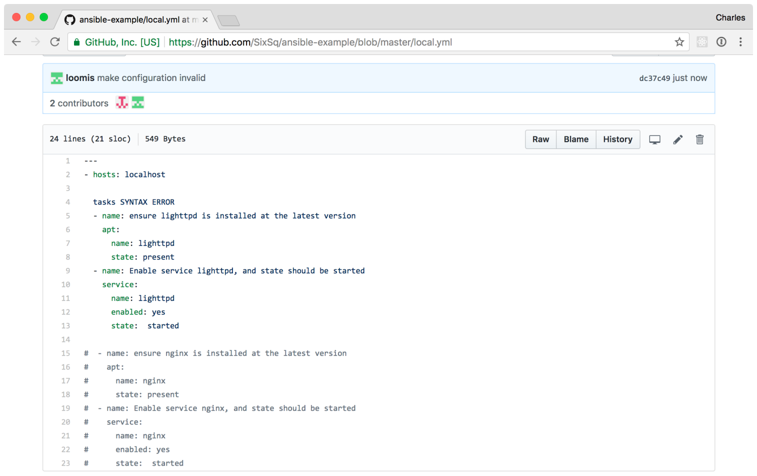
This computer has installed the Ubuntu operating system, but it has nothing to do with the Ubuntu Project. Please do not contact the Ubuntu Project about it.

If you find a bug in this Lighttpd package, or in Lighttpd itself, please file a bug report on it. Instructions on doing this, and the list of known bugs of this package, can be found in the **Ubuntu Bug Tracking System.**

If everything's gone according to plan, you will see the welcome page. Moreover, Ansible is still active, checking for updates every minute.

8.6.6 Invalid Playbook

To show that Ansible is indeed polling for configuration changes and that the component will make errors visible through the Nuvla interface, we will modify the playbook with an **invalid configuration**.



https://github.com/SixSq/ansible-example/blob/master/local.yml

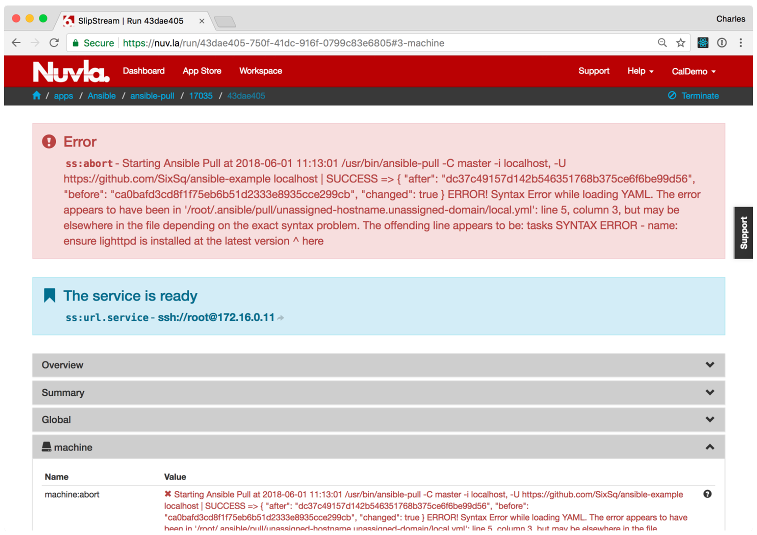
loomis make configuration invalid   dc37c49 just now

2 contributors

24 lines (21 sloc) | 549 Bytes

```
---
- hosts: localhost

  tasks SYNTAX ERROR
  - name: ensure lighttpd is installed at the latest version
    apt:
      name: lighttpd
      state: present
  - name: Enable service lighttpd, and state should be started
    service:
      name: lighttpd
      enabled: yes
      state:  started

#  - name: ensure nginx is installed at the latest version
#    apt:
#      name: nginx
#      state: present
#  - name: Enable service nginx, and state should be started
#    service:
#      name: nginx
#      enabled: yes
#      state:  started
```

When Ansible next tries to update the machine, the component will enter the "abort" state and will display the Ansible error. In this case, it should complain about the syntax of the YAML file.


SlipStream | Run 43dae405
Charles
Secure
https://nuv.la/run/43dae405-750f-41dc-916f-0799c83e6805#3-machine
Nuvla
Dashboard
App Store
Workspace
Support
Help
CalDemo
apps / Ansible / ansible-pull / 17035 / 43dae405
Terminate
Overview
Summary
Global
machine

Error

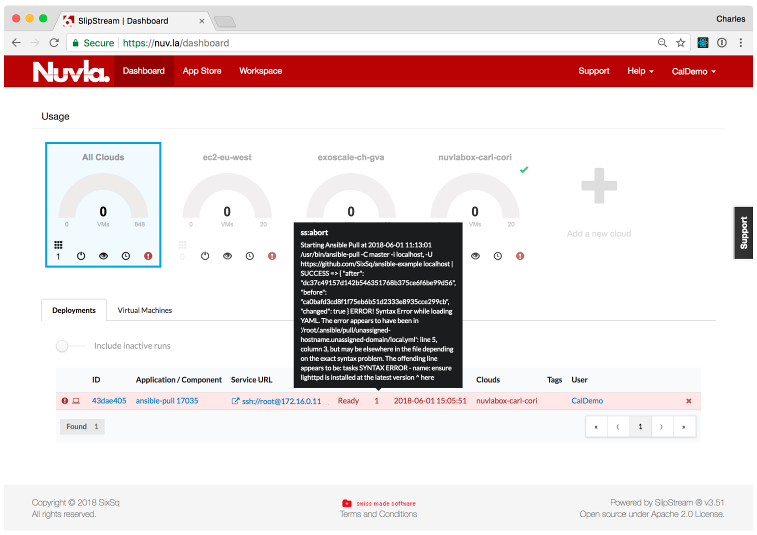
ss:abort - Starting Ansible Pull at 2018-06-01 11:13:01 /usr/bin/ansible-pull -C master -i localhost, -U https://github.com/SixSq/ansible-example localhost | SUCCESS => { "after": "dc37c49157d142b546351768b375ce6f6be99d56", "before": "ca0bafd3cd8f1f75eb6b51d2333e8935cce299cb", "changed": true } ERROR! Syntax Error while loading YAML. The error appears to have been in '/root/.ansible/pull/unassigned-hostname.unassigned-domain/local.yml': line 5, column 3, but may be elsewhere in the file depending on the exact syntax problem. The offending line appears to be: tasks SYNTAX ERROR - name: ensure lighttpd is installed at the latest version ^ here

The service is ready

ss:url.service - ssh://root@172.16.0.11

Name	Value
machine:abort	✖ Starting Ansible Pull at 2018-06-01 11:13:01 /usr/bin/ansible-pull -C master -i localhost, -U https://github.com/SixSq/ansible-example localhost | SUCCESS => { "after": "dc37c49157d142b546351768b375ce6f6be99d56", "before": "ca0bafd3cd8f1f75eb6b51d2333e8935cce299cb", "changed": true } ERROR! Syntax Error while loading YAML. The error appears to have been in '/root/.ansible/pull/unassigned-hostname.unassigned-domain/local.yml': line 5, column 3, but may be elsewhere in the file

This status can also be seen from the dashboard.



ss:abort

Starting Ansible Pull at 2018-06-01 11:13:01 /usr/bin/ansible-pull -C master -i localhost, -U https://github.com/SixSq/ansible-example localhost | SUCCESS => { "after": "dc37c49157d142b546351768b375ce6f6be99d56", "before": "ca0bafd3cd8f1f75eb6b51d2333e8935cce299cb", "changed": true } ERROR! Syntax Error while loading YAML. The error appears to have been in '/root/.ansible/pull/unassigned-hostname.unassigned-domain/local.yml': line 5, column 3, but may be elsewhere in the file depending on the exact syntax problem. The offending line appears to be: tasks SYNTAX ERROR - name: ensure lighttpd is installed at the latest version ^ here

ID	Application / Component	Service URL				Clouds	Tags	User
43dae405	ansible-pull 17035	ssh://root@172.16.0.11	Ready	1	2018-06-01 15:05:51	nuvlabox-carl-cori		CalDemo

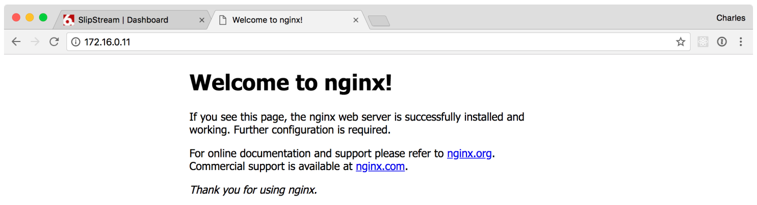
8.6.7 Corrected Playbook

We will now correct the playbook and switch from lighttpd to nginx as the web server. To do this:

- Fix the syntax error that was introduced,
- For lighttpd, set "enabled" to "no",
- For lighttpd, set "state" to "stopped",
- Uncomment the nginx section.

Once this playbook is saved, Ansible should update the machine with the changes and the abort status should be cleared.

Visiting the same web server URL should now show the nginx welcome page instead of the lighttpd page that was seen before.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

As before, the machine will continue to check for configuration changes and apply them as necessary.

8.6.8 Terminating the Deployment

The machine can be terminated by either clicking on the "x" next to the deployment on the Dashboard or on the "Terminate…" action on the deployment detail page.

8.6.9 Summary

The demonstration showed how easy it is to use configuration management systems with NuvlaCity. You only need to create a "shell" SlipStream component that references your preferred operating system and provide a short script to inform Nuvla about the success or failure of updates.