Spinal Stack Documentation

Release J.1.1.0

eNovance

August 28, 2016

Contents

The *Spinal Stack* aims to deliver a flexible, high available and scalable OpenStack infrastructure running all OpenStack core components.

CHAPTER 1

Tables of contents

1.1 Introduction

Spinal Stack is the solution to install OpenStack in production at eNovance.

1.1.1 Project features

1. Bare-metal provisioner: hardware discovery, validation and configuration
2. Configure OpenStack services to be highly available and scalable
3. Flexible environments to adapt to various needs
4. Post-deployment checks and runset of tests to validate the deployed OpenStack
5. Orchestration of upgrades without downtime
6. View logs in real-time through the UI
7. Support for Red Hat
8. Based 1OO% on upstream

1.1.2 OpenStack features

Spinal Stack is released with all OpenStack core projects.

service	component	backends/drivers	notes
load-balancer	HAproxy	•	SSL termination support
clustering	Pacemaker	Corosync	•
cache	Memcached	•	used by Horizon & service group API
dashboard	Horizon	•	SSL support
database	MySQL	Galera	MariaDB packages
compute	Nova	local + RBD + NFS	RBD only on Debian
identity	Keystone	MySQL	•
image	Glance	local + RBD + NFS	•
network	Neutron	OVS/GRE + Cisco N1KV	•
orchestration	Heat	•	•
telemetry	Ceilometer	MongoDB	•
block storage	Cinder	RBD + NetAPP + iSCSI	•
object storage	Swift	•	•
logging	fluentd	•	with kibana3 & elasticsearch
monitoring	sensu	•	with uchiwa dashboard
database as a service	Trove	•	experimental now
message queuing	RabbitMQ	•	HA queues (cluster mode)

1.2 Architecture

Spinal Stack use currently eNovance reference architecture to run OpenStack in production.

1.3 Deployment guide

This section aims to document how to perform Spinal Stack deployments.

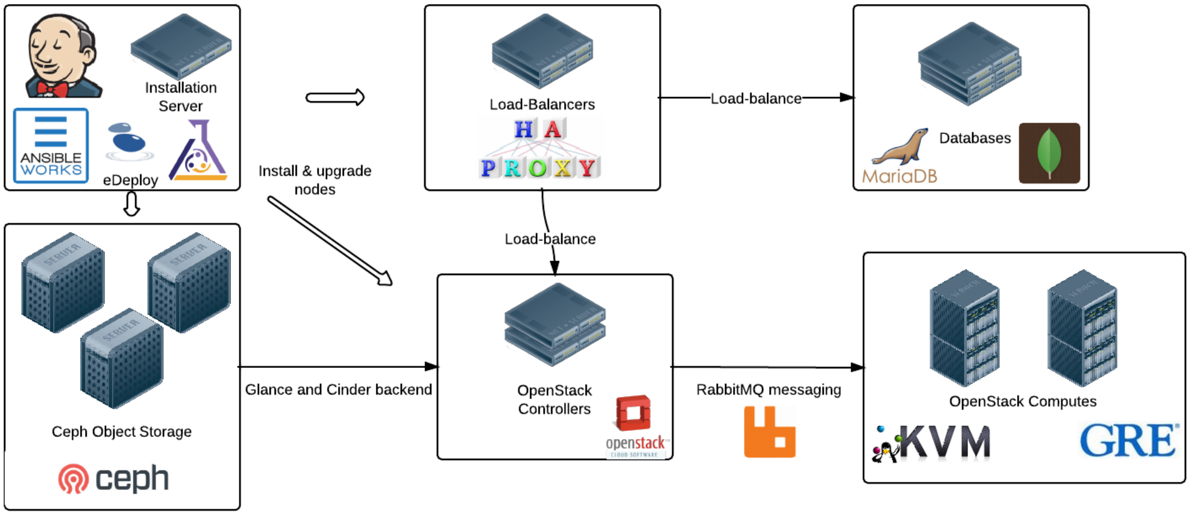

Fig. 1.1: Spinal Stack architecture

1.3.1 Introduction

The deployment of Spinal Stack is done by 3 steps:

- **bootstrap**: manage lifecycle of servers (hardware + Operating System + packages)
- **configuration**: configure all services to make Spinal Stack working
- **sanity**: ensure Spinal Stack is deployed and working as expected by running advanced functionnal testing

Requirements

Before running a deployment, there are some requirements to understand the way Spinal Stack works do be deployed.

step	project(s)	external documentation
bootstrap	eDeploy, PXEmgr	eDeploy, PXEmngr
configuration, upgrade	config-tools, Puppet, Hiera, Ansible	Puppet, Hiera, Ansible
sanity	Tempest	Tempest

The minimal requirements to install Spinal Stack are:

Deployment type	Number of Servers	Operating Systems
PoC, dev	4	RHEL 7.0
Production	> 10	RHEL 7.0

Note: Spinal Stack can be installed on bare-metal server or in a virtualized environment.

1.3.2 Bootstrap

Spinal Stack is able to be bootstrapped on physical servers or virtual machines.

In this guide, we will see the physical servers use case.

The bootstrap steps are performed by the installation server node, which will be in charge of deploying all other nodes.

eDeploy is the tool to provision and update systems (physical or virtual) using trees of files instead of packages or VM images.

The installation server node is built by using install-server eDeploy role.

All other nodes (load-balancers, controllers, compute nodes, etc) are built by using openstack-full eDeploy role.

Before starting the deployment, you need to get the last build of these roles.

Build eDeploy roles

To build eDeploy roles by yourself on your eDeploy server, you can run:

```
$ export ROLES="openstack-full install-server"
$ export VIRTUALIZED=/srv/edeploy/tools/virt.conf
$ /srv/edeploy/tools/jenkins-build.sh <src dir> <build dir> <archive dir> [<make params>]
```

When making continuous integration, this step can be done by Jenkins (upstream) which will distribute eDeploy roles on the installation servers (downstream).

Deploy the install-server role

This role contains all these components:

- eDeploy
- PXEmngr
- PuppetDB: running on port 8080 and a vhost acting as a proxy to terminate the SSL listening on this URL: http://install-server:8081
- Puppet Dashboard: monitor Puppet nodes using this URL: http://install-server:82
- Kibana3: View logs in real-time through UI using this URL: http://install-server:8300
- Jenkins: manage lifecyle of our infrastructure (configuration, sanity and upgrade) using this URL: http://install-server:8282

The first step, is to bootstrap the install-server by using eDeploy itself. There is several ways to do it and this manual explains them in details.

In the case you don't have an eDeploy server to bootstrap the install-server node, you can use the local installation with an USB key method.

You have to follow eDeploy manual to finish the installation of eDeploy service on the install-server node. Once we have eDeploy working, we can continue to prepare the configuration.

Note: There is no need to install Puppet, Ansible or any service. They are already present in install-server role.

Warning: independently of the process used to bootstrap the install-server, a valid pem file should be dropped at /etc/puppet/ssl/puppetdb.pem to secure the connection between the nodes and the install server. The use of cloud-init is recommended to realize this action. Without this file, the deployment of Spinal Stack will fail.

Deploy the openstack-full roles

Once the install-server node is ready, we can deploy all our infrastructure by using eDeploy over the network. To start the deployment, boot the targeted server by using the proper boot device regarding the kind of deployment you choose (PXE versus USB).

1.3.3 Configuration guide

After bootstrapping the servers, we need to generate the configuration.

To allow more flexibility and to easily define the parameters of Spinal Stack, config-tools is in charge of generating and running the configuration remotely. Config Tools are a set of tools to use puppet to configure a set of nodes with complex configuration using a step by step approach. Each step is validated by serverspec tests before going to the next step. If the tests of a step fail, Puppet is called again on all the nodes.

YAML is the format that Spinal Stack is using to generate the configuration.

To configure Spinal Stack correctly, there are two notions to understand:

Infrastructure (common accross multiple deployments) describe how Spinal Stack will be configured. It will describe the different kind of nodes, which services are running on them and some default parameters in the deployment. The infrastructure YAML files can be use by multiple deployment and should never be tied to information specific to a single environment (like IP addresses or passwords).

Environment (deployment-specific) describe the details of our deployment: IP addresses, domain name, passwords, SSH keys, etc. This file should not be shared with any other deployment and has to be stored safely.

puppet-openstack-cloud

puppet-openstack-cloud module is the Spinal Stack composition layer. All parameters and details are documented on this page.

Before deploying Spinal Stack, you should learn about the composition layer capabilities, so you will have the desired environment in your deployment scenario.

Infrastructure YAML files

Currently, there are two examples of infrastructure YAML file which work for 2 scenarios: The infrastructure YAML repository is flexible enough to support multiple scenarios.

In “scenario” directory, you will find differents scenarios that contain a “infra.yaml” file which will fit (or not) with your deployment. If none of these scenarios fits with your deployment, you have to create your own, respecting this YAML architecture:

- *name*: Name of the profile (ex: install-server)
- *arity*: Define a rule for the number of servers expected in the deployment (ex: 1+n means at least one server with this profile)
- *edeploy*: eDeploy role to use for the deployment. Usually, we use install-server role for installation server node and openstack-full role for all other nodes.
- *steps/roles*: contains the different steps of the deployments by defining for each step the Puppet classes expected to be applied on the nodes. It’s optionnal to define the steps for each profile.
- *serverspec_config*: parameters to make Spinal Stack serverspec tests working on the profiles.

- *ordered_profiles*: describe the order to run Puppet for each step. For example, if we want to run puppet on compute nodes before the controller.
- *serverspec*: Spinal Stack Puppet classes / Spinal Stack serverspec tests names assotiation

This directory is common to all scenarios:

data *data* directory contains Hiera files. Spinal Stack is structured in 3 Hiera levels:

- *FQDN* (Full qualified domain name): parameters and classes for some hosts only. (ex: set VRRP priority on a load-balancer node in particular)
- *type*: defines some classes and parameters for each profile that we created in the *infra* YAML file
- *common*: common classes & parameters that will be applied on all the nodes of the deployment.

Note: They are listed from the highest priority to the lowest.

data/fqdn Later, we will describe another YAML file which is the *environment*. In this file, we will see how Hiera will use at this level. FQDN hiera level is useful when we want to set specific parameters for a host (like the storage devices for Swift and Ceph, etc).

data/type This file contains all the parameters that we pass to the different Puppet classes. *config-tools* will generate the different profiles with the corresponding Puppet classes for each profile. If we want to override the paramater of a class in particular, we can do this at the FQDN Hiera level. If there are too many modifications to do at this level, it may be because the deployment does not fit with this scenario and you may have to create a new infrastructure environment.

data/common YAML file which will contain all parameters consummed by *data/type* to configure the Puppet classes. The actual data could be pulled from the *environment file* that we will write later (ex: config.mysql_root_password) or from common itself (ex: galera_ip: “%{hiera(‘vip_public_ip’)}”) by using Hiera syntax.

heat OpenStack Heat (Orchestration) template that could be used to run the infrastructure in OpenStack.

Environment YAML file

The *environment* that we just talked in *infrastructure* is something specific for each deployment and has to be generated each time you deploy Spinal Stack.

There is an example of *environment* file here.

Let’s create this file (in the example it will be named joe.yml) and describe its structure: * *infra*: infrastructure name (ex: 3nodes)

For each host:

- *hostname*: hostname of the host
- *profile*: profile name used in infra.yaml in the *infrastructure* YAML file
- *ip*: IP used to reach the servers from internal network.

Note: The servers have to be reachable from the node where we will execute the configuration process.

- *config*: contains FQDN Hiera data level parameters.

After hosts description:

- *config*: contains confidentials & deployment specific informations, like IP addresses, subnets, passwords, SSH keys, etc.

This file should be stored on a private versionned git repository since it contains confidential informations.

This is an example of a 3 nodes deployment: From the installation server, the configuration is generated by first creating this YAML file:

```
**env-joe.yaml**

module:
  git@github.com:stackforge/puppet-openstack-cloud
ansible:
  git@github.com:enovance/edeploy-roles
serverspec:
  git@github.com:enovance/openstack-serverspec.git
edeploy_repo:
  git@github.com:enovance/edeploy.git
environment:
  joe
infrastructure:
  git@github.com:enovance/openstack-yaml-infra.git
scenario:
  ref-arch
jenkins:
  git@github.com:enovance/jjb-openstack.git
```

Note: If scenario is empty, the "ref-arch" will be configured by default.

Note: Before continuing to the next steps, you have to ensure these following requirements:

- installation server has Internet access
- all nodes have the user defined in the environment YAML file (joe.yml) with sudo permissions (usually configured by cloud-init)
- installation server can SSH all nodes with the user defined in the environment YAML file (joe.yml)

We are now ready to generate the Spinal Stack configuration.

Generate the configuration

From the installation server, run:

```
$ git clone git@github.com:enovance/config-tools.git
$ cd config-tools
$ git checkout J.1.0.0
$ ./provision.sh -i J.1.0.0 git@my-env-git-repo:spinalstack-env/env-joe.yaml version=RH7.0-J.1.0.0
```

Run the configuration

From the installation server, connect to the Jenkins server by using this URL: http://install-server:8282 and run the job *puppet*.

Depending of the infrastructure size, the configuration can take 30 minutes or more.

During the configuration

1. Puppet master is being prepared by **config-tools** *provision.sh* script. It will take care of Puppet configuration, Hiera data dynamic management (according to the steps) and the install-server will be puppetized itself.
2. Then the deployment of OpenStack nodes is starting:
   - Step 1: Memcached, MySQL, MongoDB, RabbitMQ,i logging agent, Ceph monitor and Ceph Key management
   - Step 2: HAproxy, Ceph OSD, Horizon, Swift storage nodes
   - Step 3: Keystone users, tenants, services and endpoints; Create Ceph storage pools
   - Step 4: OpenStack API, schedulers and compute services
   - Step 5: OpenStack Networking services

For each step, Puppet is run on the nodes where the step is needed. Puppet is run until serverspec tests pass with a limit of 5 times. If after 5 times of Puppet run, the serverspec tests still fail, Jenkins job will fail and provide an output to let the deployer know which tests fail. The concept of steps make easy the debugging when something is wrong during the deployment process.

Warning: serverspec tests do not certify OpenStack is up and running. They just validate that Puppet did what we expected from the configuration point of view.

1.3.4 Components Configuration

This section aims to document how to configure each main component of Spinal Stack

ElasticSearch

ElasticSearch is a full-text search engine. In Spinal Stack it stores all the logs provided by the various components Spinal Stack deploys and configure.

Configuration

Configuration	Description	Default
elasticsearch	Enable HAProxy binding Internal network IP	true
elasticsearch_bind_options	HAProxy bind options Internal network IP	HAProxy default bind options
elasticsearch_bind_ip	IP address on which ElasticSearch will be listening	Internal network IP
elasticsearch_port	Port on which ElasticSearch will be listening	9200

puppet-openstack-cloud

- cloud::database::nosql::elasticsearch
- cloud::logging::server (include cloud::database::nosql::elasticsearch)

Non-HA Setup

In a non-HA setup, simply instantiate the ‘cloud::logging::server’ on the concerned node.

HA Setup

In an HA setup, instantiate 'cloud::logging::server' on all the nodes needed, puppet-openstack-cloud will take care of putting the cluster behind HAProxy, and ElasticSearch itself will take care of creating the internal cluster.

Kibana

Kibana is a front-end to ElasticSearch. It enables to visualize all the Spinal Stack components logs.l

Configuration

Configuration	Description	Default
kibana_bind_ip	IP on which Kibana will be listening on	Internal network IP
kibana_servername	Servername in which Kibana will be listening on	VIP public full qualified domain name
kibana_port	Port on which Kibana will be listening on	8300
kibana_manage_ws	Boolean to describe if the apache vhost should be managed	true
kibana_es_server	IP address or hostname of the ElasticSearch server to connect to	VIP internal network IP
kibana_es_index	Index on which logs are stored on the ElasticSearch server	fluentd

Memcached

Memcached is an in-memory key-value store for small chunks of arbitrary data (strings, objects) from results of database calls, API calls, or page rendering. In Spinal Stack, and in Openstack in general it is used by Keystone.

Configuration

Configuration	Description	Default
memcache_servers	An array of memcached servers	An array of the controller node ip with the port ':11211' suffixed

MongoDB

MongoDB is a document oriented database. In Spinal Stack it is used as a backend for ceilometer.

Configuration

Configuration	Description	Default
mongo_nodes	An array of the MongoDB cluster nodes	An array with the controller private ip addresses
mongo_nojournal	Boolean to describe if journaling is disabled	false
mongo_replset_members	An array of MongoDB instance part of the replicaset	An array with the machine hostname
replicatset_enabled	Boolean to described if replicaset is enabled	true
mongodb_version	The version of MongoDB to install	2.4.0

PuppetDB

PuppetDB collects data generated by Puppet. It allows to have a global overview of every node of the Spinal Stack deployment and their resource in a centralized place.

Requirements

In Spinal Stack, PuppetDB is configured the following way :

- The PuppetDB daemon runs on the install-server node with SSL disabled. Hence, it runs on `127.0.0.1:8080`
- Nodes still reach PuppetDB using SSL. SSL is terminated at the webserver level via a vhost listening on port `8081` that also serves as a proxy to `127.0.0.1:8080`

Configuration of PuppetDB happens before step 1 since it needs to be ready for the really first run. Hence the SSL certificate needs to be present a deployment time.

How to push the PuppetDB certificate during the deployment:

- On baremetal or virtualized deployment: push the file in your environment in `etc/puppet/ssl/puppetdb.pem` and it will automatically copied on the install-server. Spinal Stack will take care of the permissions during the bootstrap.
- On Inception deployment, running Heat, you can provide `puppetdb_pem` parameter in your environment, containing the RAW data of your certificate. During the stack creation with Heat, cloud-init will take care to create the file on the install-server instance.

Warning: To start correctly, the PuppetDB certificate has to be in place at `/etc/puppet/ssl/puppetdb.pem` otherwise PuppetDB won't start and the whole deployment will fail. If the file does not exist during the deployment, the `configure.sh` script will fail to avoid useless debug during the deployment.

Note: The `puppetdb.pem` is a *standard* webserver SSL certificate (see security guide's SSL section for more details)

Configuration

Configuration	Description	Default
`puppetdb_server`	The PuppetDB server URL	None
`puppetdb_pem`	The PuppetDB certificate	None

RabbitMQ

RabbitMQ is a message broker software. It enables the various part of Openstack to communicate together.

Configuration

Configuration	Description	Default
rabbit_names	An array of rabbit servers hostnames	An array of the controller hostnames
rabbit_hosts	An array of rabbit servers ip addresses	An array of the controller node ip with the port ':5672' suffixed
rabbit_password	RabbitMQ password	rabbitpassword
rabbit_cluster_node_type	RabbitMQ cluster node type	disc
rabbit_port	Port on which RabbitMQ will listen on	5672
rabbit_ip	Ip address of RabbitMQ interface	IP of the node it is one
rabbit	Boolean to described if RabbitMQ should be load balanced	false
rabbit_bind_options	An array of HAProxy bind options	Global HAProxy binding options array
erlang_cookie	Erlang cookie to use	No default, the parameter is required
rabbit_cluster_count	Number of nodes where queues are mirrored	undef, so all RabbitMQ nodes will be mirrored

Upgrade from I.1.3.0 to J.1.0.0

This is a special note when upgrading from I.1.3.0 to J.1.0.0. Due to the new puppetlabs-rabbitmq implementation, a new parameter is now required in the environment file: erlang_cookie.

If you are already running Spinal Stack I.1.3.0 and you want to upgrade to J.1.0.0, you will have to look at the current content of /var/lib/rabbitmq/.erlang.cookie and copy it to erlang_cookie parameter, so your upgrade will work correctly. If you miss that part, Spinal Stack will take care of re-building the RabbitMQ cluster, but you'll have to expect some downtime.

If you miss the parameter in your env, config-tools scripts (configure.sh) will fail to run. It prevents to install Spinal Stack without cookie so RabbitMQ-server could not work properly.

Redis

Redis is an advanced key-value cache and store. In Spinal Stack it is used as :

- a backend for Ceilometer
- a backend for Sensu API

Configuration

Configuration	Description	Default
redis	Enable HAProxy binding	true
redis_bind_ip	IP address on which Redis will be listening	Internal Network IP
redis_port	Port on which Redis will listening	6379
redis_bind_options	HAProxy bind options	HAProxy default bind options
redis_haproxy_port	Port on which HAProxy will bind to for Redis traffic	6379
sentinel_port	Port on which sentinel will listening	26739
sentinel_down_after	Time in ms an instance should not be reachable for a Sentinel starting to think it is down	1000
sentinel_failover_timeout	The time needed to restart a failover after a previous failover was already tried against the same master by a given Sentinel	1000
sentinel_notification_script	Script called for any sentinel event that is generated in the WARNING level (for instance -sdown, -odown, and so forth)	/bin/redis-notifications.sh
redis_master_name	Hostname of the Redis master	•
redis_master_ip	IP address of the Redis master	redis_master_name
sentinel_haproxy_monitor_ip	IP address of the HAProxy Redis is ran behind	Public Virtual IP
sentinel_haproxy_monitor_port	Port of the HAProxy API Redis is ran behind	10300

Warning: The redis_master_name parameter is mandatory. Without it the deployment of Redis in a HA setup will fail.

puppet-openstack-cloud

- cloud::database::nosql::redis::server
- cloud::database::nosql::redis::sentinel

Non-HA Setup

In a non-HA setup, only the following parameters are necessary to configure: redis, redis_bind_ip, redis_port, redis_bind_options, redis_haproxy_port.

To deploy Redis in a non-HA setup, include the 'database::nosql::redis::server' on the concerned node in your scenario and adjust the aforementioned parameters in the environment file.

HA Setup

In an HA setup, all the aforementioned parameters will have an impact on your setup.

To deploy Redis in HA setup, include both 'database::nosql::redis::server' and 'database::nosql::redis::sentinel' on the concerned nodes in your scenario.

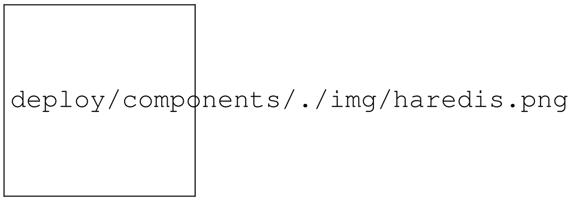

Fig. 1.2: HA Redis architecture

How does it work ? Redis is deployed in an active/passive setup. Redis stream will always land on the same Redis server, the master, through HAProxy. On the node where Redis runs, there is another process called Sentinel that is also running, this process is in charge of electing a master in case the master fails. If the master fails, the sentinel will detect this change and notify HAProxy to route the Redis stream to the new elected Redis master.

RHN

When deploying Spinal Stack on Red Hat, you need to configure your RHN subscription.

Configuration

Configuration	Description	Default
rhn_registration	RHN Credentials	•

```
---
rhn_registration:
  username: joe
  password: secrete
```

Sensu

Sensu is a modern monitoring system made for the cloud. It is the monitoring system installed by default with Spinal Stack.

Configuration

Configuration	Description	Default
sensu_api	Enable HAProxy binding for the Sensu API	true
sensu_api_ip	IP address on which Sensu API will be listening	Internal Network IP
sensu_api_port	Port on which Sensu API will be listening	4568
sensu_api_bind_options	HAProxy bind options for the Sensu API	HAProxy default bind options
sensu_dashboard	Enable HAProxy binding for the Sensu Dashboard	true
sensu_dashboard_ip	IP address on which the Sensu Dashboard will be listening	Internal Network IP
sensu_dashboard_port	Port on which the Sensy Dashboard will be listening	3000
sensu_dashboard_bind_options	HAProxy bind options for the Sensu Dashboard	HAProxy default bind options
sensu_install_repo	Should puppet manage the sensu repo	false
sensu_uchiwa_install_repo	Should puppet manage the uchiwa repo	false
sensu_rabbitmq_user	Sensu Rabbitmq user	sensu
sensu_rabbitmq_password	Sensu Rabbitmq user’s password	rabbitpassword
sensu_rabbitmq_vhost	Sensu Rabbitmq vhost	/sensu
sensu_redis_host	Redis host Sensu will connect to	Public Virtual IP
sensu_redis_port	Redis port Sensu will connect to	6379
sensu_dashboard_user	Sensu Dashboard user name	admin
sensu_dasbhaord_password	Sensu Dashboard password	password
sensu_plugins	Sensu list of plugins	{}
sensu_handlers	Sensu list of handlers	{}
sensu_checks	Sensu list of checks	{}

puppet-openstack-cloud

- cloud::monitoring::server::sensu
- cloud::monitoring::agent::sensu

Import Plugins

If the check you want is not already installed on the machines, you can rely on the ‘sensu_plugins’ parameter to retrieve them in the environment file:

```
sensu_plugins:
  'https://raw.githubusercontent.com/sensu/sensu-community-plugins/master/plugins/system/check-mem.sh
   type: url
```

As we previously noted, the plugins will be dropped in ‘/etc/sensu/plugins’. You can apply the exact same pattern to retrieve handlers not present on the system:

```
sensu_plugins:
  'https://raw.githubusercontent.com/sensu/sensu-community-plugins/master/handlers/notification/pager
    type: url
    install_path: /etc/sensu/handlers/
```

Enable Checks

Checks are normally enabled per profile, but if you want some checks to be enabled for each and every node, the setting can be located in the environment file:

```
sensu_checks:
  'check-mem' :
    command: '/etc/sensu/plugins/check-mem.sh -w 150 -c 100'
    subscribers: base
    standalone: false
```

Enable Handlers

Each handler will require its own configuration, to enable a handler edit the 'sensu_handlers' in the environment file:

```
sensu_handlers:
  'handler_irc':
    command: '/etc/sensu/handlers/irc.rb'
    type: 'pipe'
    config:
      irc_server: 'irc://user:password@irc.freenode.net:6667/#mycompany'
      irc_ssl: false
```

Sudo

When deploying Spinal Stack a user might have the need to specify sudoers rules for users. This is possible by configuring the sudo_configs variable in your environment.

Configuration

Configuration	Description	Default
sudo_configs	An hash of sudoers rules	Refer to the example

By default sudo_configs is set with the following content (for sensu proper operations)

```
---
'sensu':
  'content' : "Defaults:sensu !requiretty\nDefaults:sensu secure_path = /usr/local/sbin:/usr/local/bi
```

An administrator can add more sudoers rules or override the sensu sudoer rule like this

```
---
sudo_configs:
  'user1':
    content: 'Defaults:user1 !requiretty'
  'user2':
    content: 'Defaults:user2 !requiretty'
  'sensu':
    content: 'Default:sensu requiretty'
```

Limits

When deploying Spinal Stack a user might have the need to specify security limits (with pam_limits) rules. This is possible by configuring the limit_rules variable in your environment.

Configuration

Configuration	Description	Default
limit_rules	An hash of limits rules	Refer to the example

By default limit_rules is set to empty. An deployer can add limits rules for MySQL (useful in production) like this:

```
---
limit_rules:
  'mysql_nofile':
    ensure: 'present'
    user: 'mysql'
    limit_type: 'nofile'
    both: '16384'
```

sysctl

When deploying Spinal Stack a user might have the need to sysctl rules. This is possible by configuring the sysctl_rules variable in your environment.

Configuration

Configuration	Description	Default
sysctl_rules	An hash of sysctl rules	Refer to the example

By default sysctl_rules is set to empty.

This is an example of usage:

```
---
sysctl_rules:
  'net.ipv4.ip_forward':
    value: '1'
  'net.ipv6.conf.all.forwarding':
    value: '1'
```

Neutron

Floating IP network

Configuration	Description	Default
floating_network_name	Name of floating IP network	unset
floating_network_subnet	IP Subnet of floating IP network	unset
floating_network_start	First IP of floating IP subnet	unset
floating_network_end	Last IP of floating IP subnet	unset
floating_network_gateway	Gateway of floating IP subnet	unset

If floating_network_name is defined, we require all floating_* parameters defined below. The Sanity process will ensure that the floating IP network is created, in order for Javelin to spawn servers and test connectivity. If none of these parameters are defined, Javelin won't create server resources during Sanity process.

L3 Agent HA

Configuration	Description	Default
neutron_l3_ha_enabled	Enable/disable HA with VRRP	false
neutron_l3_ha_vrrp_auth_type	VRRP auth type	PASS
neutron_ha_vrrp_auth_password	VRRP auth password	•
neutron_l3_allow_automatic_l3agent_failover	Enable/disable auto rescheduling	false
neutron_l3_agent_mode	Working agent for the agent	legacy
enable_distributed_routing	Enable/disable DVR	false
enable_l2_population	Enable/disable L2 population	true

Warning: The upgrade process makes it possible to enable L3 HA if it was previously disabled on your setup, except if you've activated the L2 population mechanism driver. If you really need to activate L3 HA on your setup, we recommend you either doing a fresh install or disabling the L2 population driver.

Warning: In OpenStack Juno, DVR & VRRP can't work together. That means you can't enable neutron_l3_ha_enabled and enable_distributed_routing in the same time. If you try to do it, Puppet will fail to avoid any configuration error.

Warning: In OpenStack Juno, VRRP can't work with l2population mechanism driver. Ensure neutron_mechanism_drivers parameter does not contain l2population in the array and enable_l2_population is at false. Please read this blogpost for more informations about HA compatiblity.

Note: neutron_l3_allow_automatic_l3agent_failover allows to re-schedule virtual routers if a Neutron L3 agent goes down. If neutron_l3_ha_enabled is enabled, this is not useful to enable neutron_l3_allow_automatic_l3agent_failover.

More documentation about L3 configuration on official manual.

1.3.5 Sanity guide

Spinal Stack presents a validation suite that can be run after each deployment to ensure that every feature is working correctly. It consists of a series of tests based on tools developed by the upstream OpenStack QA project. This test suite is referred to as “Sanity tests” and can be launched from Jenkins under the name ‘Sanity’. As of the J.1.1.0 version, this job uses 2 tools:

- Javelin
- Tempest

Javelin

Javelin is a tool that allows the creation, check and destruction of some OpenStack resources. We use it to validate that existing resources are able to survive a version upgrades. Here's how it works:

1. Before running, Javelin will compare the current version of Spinal Stack with the latest version tested. The current version is provided by the tool *edeploy* and the latest version tested is fetched from disk. This will determine if Javelin is running before or after an upgrade process.

2. If Javelin is running before an upgrade (or if it's a first run after the initial deployment), it will simply create a bunch of resources.
3. If Javelin is running after an upgrade (in other words, if the current version of Spinal Stack is superior to the latest version tested), Javelin will check for the existence of the OpenStack resources. If it cannot find them, Javelin considers that the resources didn't survive the upgrade and will report an error.
4. After every run, Javelin will update the latest tested version on disk with the current version of Spinal Stack.

The latest version tested by Javelin can be found on the install server, in the file: */opt/tempest-scripts/javelin-latest-run*

The resources manipulated by Javelin are described in a Yaml file managed by the Sanity suite. As of the J.1.1.0 release, here are the resources:

Tempest

Tempest is the suite of functional tests that is used in the gate to validate every API call against new developments. We use it to validate that the deployment went well. When Sanity is launched, it will take care of:

- Generating a tempest.conf relevant to the Spinal Stack deployment
- Create the users, tenants, roles and images that will be used in the tests
- Launch over 1600 functional tests, testing API and CLI
- Report the tests result in the Jenkins interface

A couple of things to note about Tempest:

- Tempest will create a bunch of resources during its tests. These resources will be cleaned up after Sanity, even in the case of tests failure.
- Sanity enables a feature of Tempest called "tenant isolation", which means that every resource created by a test will be only visible inside its own tenant. This allows parallel execution of the tests while preventing potential race conditions.

More info

For more information on Tempest, you should read:

- the Spinal Stack Tempest configuration file, located at: */usr/share/openstack-tempest-<version>/etc/tempest.conf*, where <version> stands for the OpenStack version in use ('icehouse' or 'juno' for instance).
- the upstream Tempest documentation

1.3.6 Post Deployment

Tips and tricks for post-deployment actions and configuration.

Glance

Import Cloud images

Currently Cloud images are not available online yet. We will provide both *QCOW2* and *RAW* format images. But in the meantime this is how we add a new image:

```
glance image-create --name <name> --disk-format raw --container-format bare --file <path_to_img> --is
```

Mirosoft Windows images

Microsoft Cloud images need special properties to run properly. In order to improve the user experience we use the QLX drivers:

```
glance image-create --name Microsoft_Windows_Server_2012_R2 --disk-format raw --container-format bare
```

Properties can also be edited like this:

```
glance image-update <image-id> --property hw_video_model=qxl --property hw_video_ram=6
```

The Nova flavor must be edited accordingly to reflect the properties:

```
nova flavor-key m1.microsoft.windows set hw_video:ram_max_mb=128
```

Old Cloud images

For CentOS 5 images, the virtio support is pretty poor and tend to make the VM randomly not bootable (2/5 times). You will likely encounter the following message from the Kernel during the boot sequence.:

```
Kernel panic - not syncing: IO-APIC + timer doesn't work!
```

In order to bypass this, we use *IDE* disks and *rtl8139* network card instead of virtio models.:

```
glance image-create --name GNU_Linux_CentOS_5.1 --disk-format raw --container-format bare --file <pat
```

Nova

Delete flavors and re-create tiny flavors

In order to have bootable tiny flavor we need to increase the size of the *m1.tiny* flavor:

```
nova flavor-delete 1
nova flavor-create m1.tiny 1 512 6 1
```

Boot Microsoft Windows instances to specific hypervisors

Due to some licensing constraint, Microsoft Windows virtual machines must be booted on dedicated hypervisors. Basically you only license a few amount of hypervisor.

Warning: you will not be enable to boot the image on other flavors

Note: however the other way around is not true, you will be able to boot other images with this flavor

First you must add a new filter *AggregateInstanceExtraSpecsFilter* to the scheduler filters:

```
scheduler_default_filters=RetryFilter,AvailabilityZoneFilter,RamFilter,ComputeFilter,ImagePropertiesF
```

Then create the aggregate that will contain the Windows instances:

```
nova aggregate-create microsoft-windows
```

Note: you can use *nova host-list* to retrieve hypervisor names

Add host to your aggregate:

```
nova aggregate-add-host <aggregate> <server>
```

Create a new metadata for this aggregate:

```
nova aggregate-set-metadata <microsoft-windows aggregate ID> windowshypervisors=true
```

Warning: be careful if you modify the name of an aggregate all the metadata will be deleted (behavior seen on Icehouse)

Create a new flavor for the Windows instances:

```
nova flavor-create m1.microsoft.windows 6 4096 40 2
```

Assign to this flavor a special property:

```
nova flavor-key m1.microsoft.windows set windowshypervisors=true
```

Neutron

Initiale br-pub: Public Network

Create a public network on br-pub, the default provider network:

```
neutron net-create public --router:external=True
neutron subnet-create public --name ext-subnet --allocation-pool start=$ALLOCATION_PUBLIC_POOL_START,
```

Create a new provider network (on VLAN 100)

Create a new provider on VLAN 100:

```
neutron net-create public --provider:network_type vlan --provider:physical_network public --provider:
```

Create some metering labels for traffic passing through routers external interface

The following will meter all traffic going through routers' external interfaces:

```
neutron meter-label-create public-in
neutron meter-label-rule-create public-in 0.0.0.0/0 --direction ingress

neutron meter-label-create public-out
neutron meter-label-rule-create public-out 0.0.0.0/0 --direction egress
```

Rules can be more specific and include or exclude some IP ranges. See http://docs.openstack.org/admin-guide-cloud/content/metering_operations.html for more details.

Cinder

Create a QoS for a volume type

Originally both QEMU and KVM support rate limitation. This is obviously implemented through libvirt and available as an extra xml flag within the <*disk*> section called iotune.

QoS options are:

- *total_bytes_sec*: the total allowed bandwidth for the guest per second
- *read_bytes_sec*: sequential read limitation
- *write_bytes_sec*: sequential write limitation
- *total_iops_sec*: the total allowed IOPS for the guest per second
- *read_iops_sec*: random read limitation
- *write_iops_sec*: random write limitation

Set QoS options:

```
cinder qos-create high-iops consumer="front-end" read_iops_sec=2000 write_iops_sec=1000
cinder type-create high-iops
cinder qos-associate c38d72f8 9c746ca5
```

1.3.7 Upgrade guide

Spinal Stack aims to provide continuous integration and continuous delivery. Upgrades are fully automated and ensure the minimum of service downtime.

Warning: Spinal Stack upgrades only work from n to n+1. It's not possible to skip a release.

The upgrade is orchestred by 3 steps:

- Generate the new configuration on the install-server. Example with J.1.0.0 (release that we want) on 3 nodes:

```
    $ git clone git@github.com:enovance/config-tools.git
    $ cd config-tools
    $ ./provision.sh -i J.1.1.0 git@my-env-git-repo:spinalstack-env/env-joe.yaml version=RH7.0-J.1.1
```

- Run *upgrade* Jenkins job to perform Spinal Stack upgrade. It will automatically:
  - Prepare OpenStack nodes to be upgraded by running the ansible playbooks with *before_config* tag.
  - Run Puppet on all the nodes and re-validate all the steps.
  - Run some actions on OpenStack nodes by running the ansible playbooks with *after_config* tag.
- Run *sanity* Jenkins job to ensure that we still have Spinal Stack working as expected.

If you want to know how upgrades work under the hood, look *upgrade* page.

1.4 Operations guide

This section aims to document how to manage Spinal Stack deployments in production.

1.4.1 Introduction

This guide help operators to manage a Spinal Stack cloud.

1.4.2 Monitoring

Being able to monitor its infrastructure is crucial, Spinal Stack, with that in mind, provides a robust, scalable and easy way to monitor all the machines in the OpenStack cloud platform.

Monitoring Stack Components

Before tackling how to configure monitoring in Spinal Stack, the following is the list of components used to create the monitoring stack. Every component is open-source and freely available.

Name	Role	Technology	License	Homepage
RabbitMQ	Transport layer	Erlang	Mozilla Public License	http://www.rabbitmq.com/
redis	Data store	C	BSD	http://www.redis.io/
Sensu	Monitoting router	Ruby	MIT	http://www.sensuapp.org/
Uchiwa	Dashboard for sensu	NodeJS	MIT	http://sensuapp.org/docs/latest/dashboards_uchiwa

RabbitMQ

RabbitMQ is a robust and scalable AMQP broker - among other protocols.

In Spinal Stack, it allows Sensu server to publish a set of check to specific exchanges, within the sensu vhost, and receive the results published by the clients in return.

Redis

Redis is an advanced key-value cache and store

In Spinal Stack, it allows Sensu server to store persistent data with the need of a full-blown RDBMS.

Sensu

Sensu is often described as the “monitoring router”. Essentially, Sensu takes the results of “check” scripts run across many systems, and if certain conditions are met; passes their information to one or more “handlers”.

In Spinal Stack, it is the core monitoring components. Some checks are common to every nodes (ie. system checks), other are standalone checks and test nodes role specific feature.

Uchiwa

Uchiwa is the dashboard that let the administrator interact with Sensu.

In Spinal Stack, it is the component that will be the entry point for every monitoring related task, be it down-timing an alert, checking the health of the various nodes, etc...

Workflow

Overview

An image should be put here

Details

1. On the nodes

On each and every node, a sensu agent has to be enabled. A sensu agent has two roles :

- Subscribe the node to a profile of checks
- Return the status of standalone checks

2. On the monitoring server

On the monitoring server, sensu, sensu-api and redis are set up. This is where the profile of checks are described and the returned values of checks stored.

3. Monitoring dashboard

In the simplest case scenario, Uchiwa the Community Sensu Dashboard is install on the same host as the sensu server. However one is free to set it up wherever one needs and change the configuration appropriately.

Configuration

Spinal Stack provides two classes related to monitoring. monitoring/agent.pp and monitoring/server.pp. Please read below for more details.

Puppet

monitoring/agent/sensu.pp The purpose of this puppet class is to be able to fully configure sensu on the agent side.

The class itself takes no parameter it only installs the sensu packages.

As for the rest of Spinal Stack, the whole configuration takes place in the according hiera's yaml configuration file.

monitoring/server/sensu.pp The purpose of this puppet class is to be able to fully configure the sensu server.

In order to be fully operational a sensu server needs to be properly connected to Redis and RabbitMQ, which is taken care of by this class. Also, a sensu server can be more powerfull by adding plugins and handlers, which is also taken care of by this class.

This class takes the following parameters :

- checks (hash): A hash of checks to install
- handlers (hash): A hash of handlers to install
- plugins (hash): A hash of plugins to install
- rabbitmq_user (string): The RabbitMQ user to connect to
- rabbitmq_password (string): The RabbitMQ password
- rabbitmq_vhost (string); The RabbitMQ virtual host to connect to

Components Configuration

Following is the list of the puppet modules used for Spinal Stack logging infrastructure :

Name	Homepage
sensu-puppet	https://github.com/enovance/sensu-puppet
yelp-uchiwa	https://github.com/enovance/yelp-uchiwa
puppet-redis	https://github.com/enovance/puppet-redis
puppetlabs-rabbitmq	https://github.com/enovance/puppetlabs-rabbitmq

Each and every of those modules are highly configurable. Listed below are the most important parameters for a proper integration. If those parameters is not enough for your needs, please refer to the implementation of the actual module.

sensu-puppet

name	default	purpose
sensu::server	false	Include the sensu server
sensu::api	false	Include the sensu api
sensu::rabbitmq_port	5672	RabbitMQ port to be used by sensu
sensu::rabbitmq_host	localhost	Host running RabbitMQ for sensu
sensu::rabbitmq_user	sensu	Username to connect to RabbitMQ with sensu
sensu::rabbitmq_password	‘’	Password to connect to RabbitMQ with sensu
sensu::rabbitmq_vhost	sensu	Virtual host to connect to RabbitMQ with sensu
sensu::rabbitmq_ssl	false	Use SSL transport to connect to RabbitMQ
sensu::rabbitmq_ssl_private_key	undef	Private key to be used by sensy to connect to RabbitMQ
sensu::rabbitmq_cert_chain	undef	Private SSL cert chain to be used by sensy to connect to RabbitMQ
sensu::redis_host	localhost	Hostname of Redis to be used by sensu
sensu::redis_port	6379	Redis port to be used by sensu
sensu::api_bind	0.0.0.0	IP to bind the api service to
sensu::api_host	localhost	Hostname of the sensu api serice
sensu::api_port	4567	Port of the sensu api service
sensu::api_user	undef	User of the sensu api service
sensu::api_password	undef	Password of the sensu api service
sensu::subscriptions	undef	Default subscriptions used by the client

yelp-uchiwa

name	default	purpose
uchiwa::install_repo	true	Should the package manage the repos
uchiwa::user	‘’	Username to access the dashboard. Leave empty for none.
uchiwa::pass	‘’	Password to access the dashboard. Leave empty for none.
uchiwa::host	0.0.0.0	IP address / hostname to bind the monitoring dashboard to
uchiwa::port	3000	Port to bind the monitoring dashboard to
uchiwa::stats	10	Determines the retention, in minutes, of graphics data
uchiwa::refresh	1000	Determines the interval to pull the Sensu API, in milliseconds

puppet-redis

name	default	purpose
redis::port	6379	Accept redis connections on this port.
redis::bind_address	false	Address to bind to.

puppetlabs-rabbitmq This class is managed by cloud::messaging, please refer to it for it’s configuration.

Hiera

agent Since the class cloud::monitoring::agent::sensu simply install the sensu packages no configuration is required hiera wise.

server

This is an example of the configuration of the sensu monitoring server

```
---
sensu::server: true
sensu::api: true
sensu::rabbitmq_password: password
sensu::rabbitmq_host: brk01
sensu::rabbitmq_vhost: /sensu
uchiwa::user: admin
uchiwa::password: password
uchiwa::host: "%{::ipaddress}"
uchiwa::install_repo: false
cloud::monitoring::server::sensu::checks:
  check_ntp:
    command: /path/to/check/check_ntp.sh
  check_https:
    command: /path/to/check/check_http.sh
cloud::monitoring::server::sensu::rabbitmq_user: sensu
cloud::monitoring::server::sensu::rabbitmq_password: password
cloud::monitoring::server::sensu::rabbitmq_vhost: '/sensu'
```

Scalability

1.4.3 Logging

The possibility to see what is happening or happened at any point in time in an OpenStack cloud, is a key feature to ensure proper operation of the latter. Spinal Stack, with that in mind, provides a robust, scalable and easy way to view all kind of logs on all the machines in the OpenStack cloud platform.

Logging Stack Components

Before tackling how to configure logging in Spinal Stack, the following is the list of components used to create the logging stack. Every component is open-source and freely available.

name	role	technology	license	homepage
rsyslog	Logs exporter (optional)	C	GNU GPLv3	http://www.rsyslog.com
fluentd	Data collector	Ruby / C++	Apache license v2.0	http://www.fluentd.org
elasticsearch	Logs storage	Java	Apache license v2.0	http://www.elasticsearch.org
kibana3	Logs browser	JavaScript	Apache License V2.0	http://www.elasticsearch.org/overview/kibana

Rsyslog

Rsyslog provides a way to configure logging per host. It allows to export log and specify facility level per program.

In Spinal Stack, rsyslog is implemented in case it has to be plugged on a customer logging infrastracture. The ideal setup remains to configure a fluentd agent to export specifly formated logs.

fluentd

Fluentd (aka td-agent) is a log collector. It *has* to be installed in the log server and can be installed on every other node. It works by specifying inputs, modifier and outputs.

In Spinal Stack, on the log server, fluentd listens for inputs on either a tcp or upd port and then insert them into ElasticSearch.

elasticsearch

Elasticsearch is a distributed - very scalable - search server. It provides the ability to have schema-free JSON documents.

In Spinal Stack, it is the component that will store all the log sent to the log server, by default it is installed in the install-server. It can be easily scaled, simply by adding a new ElasticSearch instance to the same cluser name.

kibana3

Kibana is the visual interface to the ElasticSearch cluster. It is developed by the ElasticSearch team itself, it hence provides a good integration with ElasticSearch. Kibana is a set a javascript files ran behind a web server.

Workflow

Overview

An image should be put here

Details

1. On the nodes

On each and every node, either rsyslog or fluentd agent should be configured to export logs to the log server. Based on the chosen option (rsyslog/fluentd), log formatting/pruning is applied on the agent nodes.

2. On the log server

On the log servers, fluentd is configured to listen for incoming logs. Based on the chosen options on the agents logging configuration, it could be UDP (rsyslog), TCP (fluentd/forward), HTTP (fluentd/http). If logs are not formatted on the agents, they can be formatted on the log server using fluentd matches. Once logs are well formatter they are then sent to an ElasticSearch cluster to be stored.

3. Logs retrieval

To view, query and analyze the logs, the user must log onto Kibana. There s/he will be provided with all the power of Lucene, a full-featured text search library, to query the logs.

Configuration

Spinal Stack provides two classes related to logging. logging/agent.pp and logging/server.pp. Please read below for more details.

Puppet

logging/agent.pp The purpose of this puppet class is to be able to fully configure either fluentd and/or rsyslog on the agent nodes.

The class itself takes only 4 parameters :

- syslog_enable (boolean) : Should rsyslog be configured in the agent.
- sources (hash) : An hash describing the source of the logs.
- matches (hash) : An hash describing the matches for specific tagged logs.
- plugins (hash) : An hash describing the list of plugins that should be installed on the instance.

As for the rest of Spinal Stack, the whole configuration takes place in the according hiera's yaml configuration file.

logging/server.pp One important thing to understand about this class, is that a logging server is also a logging agent, with ElasticSearch and Kibana installed along side.

The configuration mentionned above applies here, plus the operator should configure ElasticSearch and Kibana from hiera.

Components Configuration

Following is the list of the puppet modules used for Spinal Stack logging infrastructure :

name	homepage
puppet-rsyslog	https://github.com/enovance/puppet-rsyslog
puppet-fluentd	https://github.com/enovance/puppet-fluentd
puppet-elasticsearch	https://github.com/enovance/puppet-elasticsearch
kibana3	https://github.com/enovance/kibana3

Each and every of those modules are highly configurable. Listed below are the most important parameters for a proper integration. If those parameters is not enough for your needs, please refer to the implementation of the actual module.

puppet-rsyslog

name	default	purpose
rsyslog::client::log_remote	true	Should the log be sent to remote server
rsyslog::client::remote_type	tcp	Protocol to use. Possible value 'tcp' or 'udp'
rsyslog::client::remote_forward_format	RSYSLOG_ForwardFormat	Rsyslog format
rsyslog::client::port	514	Remote port to send data to
rsyslog::client::server	log	Server name to send data to

puppet-elasticsearch

name	default	purpose
elasticsearch::init_template	undef	Name of the initd file. Possible value 'elasticsearch.RedHat.erb'

kibana3

name	default	purpose
kibana3::config_es_port	9200	Port the ElasticSearch cluster is listening on
kibana3::config_es_protocol	http	Protocol to contact the Elasticsearch cluster. Possible value ‘http’ or ‘https’
kibana3::config_es_server	‘”_+window.location.hostname”’	IP address or servername of the ElasticSearch cluster to contact
kibana3::config_kibana_index	kibana-int	ElasticSearch index to retrieve logs from
kibana3::manage_ws	true	Should the module manage the apache webserver
kibana3::ws_servername	kibana3	Apache vhost name
kibana3::ws_port	80	Port apache is listening on

Hiera

agent

Scenario 1: The agent export all its log to the log server via rsyslog

```
---
cloud::logging::agent::syslog_enable: true
rsyslog::client::log_remote: true
rsyslog::client::remote_type: update
rsyslog::client::port: 5140
rsyslog::client::server: logserver.example.com
rsyslog::client::remote_forward_format: RSYSLOG_TraditionalForwardFormat
```

Scenario 2: The agent export its log using fluentd, rsyslog isn’t configured. Logs are formatted on the agent side

```
---
cloud::logging::agent::syslog_enable:false
cloud::logging::agent::sources:
  apache:
    configfile: apache
    format: apache2
    type: tail
    tag: log.apache
    config:
      path: /var/log/apache2/access.log
      pos_file: /var/tmp/fluentd.pos
cloud::logging::agent::matches:
  forward:
    configfile: forward
    pattern: "**"
    type: forward
    servers:
    -
      host: logserver.example.com
      port: 24224
```

server

Scenario 1: The log server receives its log from rsyslog and stores them in elasticseach

```
---
elasticsearch::init_template: elasticsearch.RedHat.erb
kibana3::manage_ws: false
kibana3::config_es_server: 127.0.0.1
kibana3::config_kibana_index: fluentd
kibana3::ws_servername: logserver.example.com
cloud::logging::agent::sources:
  syslog:
    configfile: syslog
    type: syslog
    tag: log.syslog
    config:
      port: 5140
      bind: 0.0.0.0
      with_priority:
cloud::logging::agent::matches:
  elasticsearch
    configfile: elasticsearch
    pattern: "**"
    type: elasticsearch
    config:
      logstash_format: true
      index_name: fluend
      type_name: fluentd
      port: 9200
      host: 127.0.0.1
```

Scenario 2: The log server receives its log from a fluentd forward and stores them in elasticsearch

```
---
elasticsearch::init_template: elasticsearch.RedHat.erb
kibana3::manage_ws: false
kibana3::config_es_server: 127.0.0.1
kibana3::config_kibana_index: fluentd
kibana3::ws_servername: logserver.example.com
cloud::logging::agent::sources:
  forward
    configfile: forward
    type: forward
    tag: log.syslog
    config:
      port: 24224
cloud::logging::agent::matches:
  elasticsearch
    configfile: elasticsearch
    pattern: "**"
    type: elasticsearch
    config:
      logstash_format: true
      index_name: fluend
      type_name: fluentd
      port: 9200
      host: 127.0.0.1
```

Scalability

1.4.4 OpenStack

1.4.5 Ceph

1.5 Security guide

Security is a key part in a successful OpenStack cloud deployment. It needs to be ensured at every level of the platform and Spinal Stack supports some security features.

1.5.1 SSL

Currently, Spinal Stack can terminate SSL only on the loadbalancer (HAProxy) (Since version I-1.1.0). It is in the current plan to provide also the possibility to terminate SSL on each node using an SSL Proxy.

Overview

Put an image here, with rows showing which part is https ans which part is not

Configuration

Certificates

Vendor Certificate

Generate the PEM file Before moving forward with the SSL configuration, the operator needs to have a valid pem file.

There are two cases possible in order to generate a proper pem file :

- The operator has a SSL chain file : The correct pem file is the concatenation, in this exact order of : chain file + private key + certificate authority
- The operator has no SSL chain file : The correct pem file is the concatenation, in this exact order of : private key + certificate authority

Self-Signed Certificate There are some limitations with a self-signed certificate. Most of openstack-client commands should use ‘‘‘–insecure’‘‘. Whithin the spinalstack deployment you must trust your certification authority on all the servers (openstack and install-server).

Generate CA and certificates

```
mkdir self-signed-cert && cd self-signed-cert
# change theses values to match your requirements
# country, 2 letters
country="my_country"
location="my_town"
organisation="my_organisation"
domain="my_domain.com"

```

```
# generate the CA
openssl req -days 3650 -out ca.pem -new -x509 -subj /C=$country/L=$location/O=$organisation

# generate a private key for your servers
openssl genrsa -out star_$domain.key 2048

# create csr
openssl req -key star_$domain.key -new -out star_$domain.req -subj /C=$country/L=$location/O=$organis
echo '00' > file.srl

# generate the crt file
openssl x509 -req -days 3650 -in star_$domain.req -CA ca.pem -CAkey privkey.pem -CAserial file.srl -

# generate the pem file for haproxy
cat star_$domain.key star_$domain.crt > star_$domain.pem

# you can rename star_my_domain.com.* star_my_domain_com.* if needed
```

Trust the CA You have to add the ca on install-server and all the others servers, for Redhat systems:

```
cp ca.pem /etc/pki/ca-trust/source/anchors/
update-ca-trust
```

Configure HAProxy

The Spinal Stack loadbalancer class, allows the operator to enable SSL for the public facing API endpoints. It does work the same way for internal's and admin's one.

To enable SSL for a specific service, the loadbalancer.pp parameter named SERVICENAME_bind_options should contain ['ssl', 'crt', '/path/to/pem'].

So for example if one wants to set the nova api, to only accept SSL, I would set the following in the matching hiera file :

```
---
cloud::loadbalancer::nova_bind_options:
  - ssl
  - crt
  - /patch/to/pem
```

Configure Endpoints

Configuring the HAProy is just the first part of an SSL configuration. Keystone should also be aware to use https when talking to a specific interface. Hence, when registring the various services, the 'https' protocol should be specified for the various SSL aware services.

To tell Spinal Stack that nova should be contacted via 'https' on its publicUrl, apply the following configuration in your hiera file :

```
---
cloud::identity::ks_nova_public_proto: https
```

To tell Spinal Stack that nova should be contacted via 'https' on its internalUrl, apply the following configuration in your hiera file :

```
---
cloud::identity::ks_nova_internal_proto: https
```

To tell Spinal Stack that nova should be contacted via 'https' on its adminUrl, apply the following configuration in your hiera file :

```
---
cloud::identity::ks_nova_admin_proto: https
```

Note: Do not enable SSL for Nova Metadata API and Neutron Metadata Agent. This feature is not supported yet.

1.5.2 SElinux

Spinal Stack is able to run SElinux in 'enforced' mode on RHEL7.

Overview

Security-Enhanced Linux (SELinux) is a Linux kernel security module that provides a mechanism for supporting access control security policies, including United States Department of Defense–style mandatory access controls (MAC). Spinal Stack is able to enable SElinux in 'enforced' mode in a flexible way where you can configure your own booleans and modules.

Configuration

Enforce SElinux

To tell Spinal Stack to enable SElinux, apply the following configuration in your environment file :

```
selinux_mode: enforcing
selinux_directory: /usr/share/selinux/custom/
```

Adding your own configuration

In advanced deployments, you will need to tweak SElinux policy, and Spinal Stack is able to configure custom policies. You can use the Hiera per host or per profile to specify the extra booleans and modules like this:

```
---
cloud::selinux_booleans:
  - global_ssp
  - haproxy_connect_any
  - rsync_full_access
  - swift_can_network
  - nis_enabled
  - domain_kernel_load_modules
  - virt_use_execmem
  - httpd_can_network_connect
cloud::selinux_modules:
  - systemctl
  - nova-cert
  - nova-consoleauth
  - nova-scheduler
  - keepalived_haproxy
  - keepalived_init
  - mysql_rsync
  - swiftobject-sps
```

Note that this is a first implementation of SElinux in Spinal Stack. This feature remains experimental for now and is not supported on install-server nodes.

1.5.3 Firewall

Spinal Stack is able to configure IPtables rules on the nodes.

Overview

The firewalling implementation is very flexible and disabled by default. If you enable it without specific paramters, all the rules will be created depending of the services that are actually running. Example, if you have a Nova API node, this node will automatically have Nova API firewall rules (8774 tcp by default).

Configuration

Activate firewalling

To tell Spinal Stack to enable firewalling, apply the following configuration in your environment file :

```
manage_firewall: true
```

Personalize firewalling

In advanced deployments, you will need to tweak IPtables rules, and Spinal Stack is able to configure custom rules. You can use the Hiera per host or per profile to specify the extra rules like this:

```
---
cloud::firewall_rules:
  '300 allow custom application 1':
    port: 999
    proto: udp
    action: accept
  '301 allow custom application 2':
    port: 8081
    proto: tcp
    action: accept
```

Note that these rules will be applied during the deployment. If you need to create rules before the deployment, you can use "cloud::firewall_pre_extras". Or if you need to create rules after the deployment, you can use "cloud::firewall_post_extras".

Also, if you want to purge IPtables rules on all the nodes before applying Spinal Stack rules (and eventually your custom rules), you can set purge_firewall_rules to "true" in your environment file.

Limit OpenStack API requests

Some OpenStack projects does not have ratelimit middleware so you may want to use IPtables to limit the requests on the endpoints.

Example with Keystone:

```
---
cloud::identity::firewall_settings:
  '301 limit keystone API requests':
      rate_limiting: '50/sec'
```

With this parameter, Keystone API will get 50 requests per second maximum.

1.6 Developer guide

This section will help anyone who want to contribute to Spinal Stack. It has been splitted in sections related to the different steps of Spinal Stack deployment & lifecycle.

1.6.1 upgrade

The upgrade process consists of 3 steps:

Provisioning

The first step of an upgrade is to re-provision the install-server. The procedure is already documented on *Upgrade guide* page. During this process, the install-server will update: * Puppet modules * Serverspec tests and configuration * eDeploy roles * Ansible playbooks (see next section for details)

Ansible playbooks

Ansible playbooks manage the upgrade orchestration, so we don't have to worry about doing the process manually.

The playbooks are generated by config-tools during the provisioning step. Each Puppet class is associated with a playbook snippet.

For example, to manage Ceilometer Collector upgrade, we run:

```
- name: restart ceilometer-collector
  service: name={{ ceilometer_collector }} state=restarted
  tags: 7
```

You can find more information on snippets in the openstack-yaml-infra page.

For each profile (ie. controller/compute/load-balancer/network/storage) defined in your infra.yaml file, config-tools will build Ansible playbooks according to the Puppet classes you are running on each node. In other words, Ansible playbooks will be generated for every deployment so you don't ever have to worry about doing an upgrade manually.

As of version J.1.1.0, the association between Puppet classes and Ansible playbooks is still done manually. We are working on a tool inside eDeploy to associate Puppet resources with eDeploy roles. That way, we will know what is actually updated and what we need to execute during the upgrade.

Example: if in the previous eDeploy role we detect an upgrade in Ceilometer collector, we will associate the ceilometer-collector snippet with the node where cloud::telemetry::collector is run.

The goal here is to upgrade Spinal Stack in a continuous way with automatic orchestration thanks to config-tools, eDeploy, Ansible and Puppet.

eDeploy upgrade

eDeploy updates each node by copying the roles from the install-server using rsync. Currently, eDeploy repository contains metadatas to upgrade roles release after release. The most important file is the 'exclude' file. It contains all the files/directories that we want to exclude from the upgrade. Basically, we exclude all the data so we don't loose it during the upgrade. Once the rsync process is done, we execute a script called 'post'. This script is automatically generated during the build of the eDeploy roles. It contains all actions executed by packaging during the build (create users, directories, etc). Since this script is executed after an upgrade, you don't have to worry about new packages in the role, all packaging scripts are automatically executed and your system will be ready to use the service.

Note: If SElinux is enabled, 'setfiles' command will be run to re-apply all SElinux file contexts so our system will be secured again after an upgrade.

Puppet

After a run of the Ansible playbooks, we run step 0 and step 5 of Puppet. This is the usual process, so Puppet will be run in a maximum of 5 times with serverspec testing that everything is configured correctly.

Sanity

After Upgrade job, you should run Sanity job to ensure OpenStack is up and running. During Sanity process, Tempest will check both API & CLI feature are still working. Javelin will check that resources created in a previous release are still alive (servers, networks, volumes, containers, etc).

1.7 High-Availability guide

Spinal Stack aims to be a fault tolerant system, making every pieces it can highly available.

This section will describe how, each component, can be made highly available and our recommendation to set them up.

1.7.1 ElasticSearch

Status: WIP

1.7.2 Galera

Status: WIP

1.7.3 HAProxy

Status: WIP

1.7.4 Logging

Status: WIP

1.7.5 MongoDB

Status: WIP

1.7.6 Monitoring

Status: WIP

1.7.7 Openstack APIs

Status: WIP

1.7.8 RabbitMQ

Status: WIP

1.7.9 Redis

Status: WIP

1.8 Troubleshooting

This section aims to document how to troubleshoot Spinal Stack deployments during the installation phase and beyond.

1.8.1 Hiera

Hiera is a key/value lookup tools, fully integrated with Puppet, that retrieves datas from a preconfigured backend.

In Spinal Stack, hiera holds a key role. The Spinal Stack puppet module should *never* be edited, the whole deployment is data driven, hence hiera driven.

For those reason a good understanding of hiera and how to troubleshoot it is a required skill for sucessful deployments.

Principles

The following is the declaration of our messaging class

```
class cloud::messaging(
  $cluster_node_type = 'disc',
  $rabbit_names      = $::hostname,
  $rabbit_password   = 'rabbitpassword'
){
```

As one can see it has default value for each parameters. With this current state they are three ways to specify a value to the cloud::messaging class. Hiera being the way Spinal Stack works.

1. Let the default

   One can let the default value and simply include the cloud::messaging class to use it

2. Specify specific valude

   One can specify specific values in another manifest, the call would look like the following

```
class {'cloud::messaging' :
  cluster_node_type = 'foo',
  rabbit_names      = 'rabbit.example.com',
  rabbit_password   = 'secret',
}
```

3. Use Hiera

   One can use hiera to specify those values in a configured backend (yaml file, json file, mysql). Spinal Stack relies on a yaml backend at the moment. (More in Configuration file)

   The yaml hiera file would look like the following

```
---
cloud::messaging::cluster_node_type: foo
cloud::messaging::rabbit_names: rabbit.example.com
cloud::messaging::rabbit_password: secret
```

Configuration file

The configuration files is located at /etc/puppte/hiera.yaml and the default provided by Spinal Stack is the following

```
---
:backends:
  - yaml
:yaml:
  :datadir: /etc/puppet/data
:hierarchy:
  - "%{::type}/%{::fqdn}"
  - "%{::type}/common"
  - common
```

Feel free to change it to meet your needs, here the explanation of the default hiera.yaml.

- backend: Specify the backend one wants to use (can be json, mysql, etc...)
- yaml: Specify backend specifics, here for the yaml backend. Simply specify the directory where to find the files
- hierarchy: Indicates, in prefered order, where to find the datas. The actual file name on the filesystem should end with .yaml.

Admiting that the fact type is ‘controller’ and the fact fqdn is ‘controller01.example.com’, this will be the order in which data will be looked for :

- /etc/puppet/data/controller/controller01.example.com.yaml
- /etc/puppet/data/controller/common.yaml
- /etc/puppet/data/common.yaml

Debug

From time to time one might find hiera, not popullating the variable as one would have expect, then it's time to get one's hands dirty.

Log onto the puppet master server, and start looking what hiera actually returns to puppet, using the following commands.

Find the value hiera returns to puppet

```
hiera MYVARIABLE -c /etc/puppet/hiera.yaml
```

The latter will show one what hiera returned to puppet.

Note: On the previous messaging class exmaple. If one wants to find the rabbit_names value, don't use `hiera rabbit_names -c /etc/puppet/hiera.yaml`, but `hiera cloud::messaging::rabbit_names -c /etc/puppet/hiera.yaml`. `cloud::messaging::rabbit_names` is your actual parameter name.

Find which files are opened by hiera to look for informations

```
hiera MYVARIABLE -c /etc/puppet/hiera.yaml --debug
```

The latter will print a trace of all the files hiera has been looking into to retrieve data.

Specify facts in your query

```
hiera MYVARIABLE -c /etc/puppet/hiera.yaml ::type=controller ::fqdn=lb001.example.com
```

The latter tells hiera to look for MYVARIABLE, with the fact type set to controller and the fact fqdn set to lb001.example.com

1.9 Changelog

1.9.1 Icehouse

I-1.3.0

Spinal Stack I-1.3.0

OpenStack version and supported distros

- Red Hat Enterprise Linux 7: OSP5 - Icehouse 2014.1.3
- Debian Wheezy: Icehouse 2014.1.2

Note: this is the last version supporting Icehouse.

New features

- SElinux is now supported and can be “enforced” on Red Hat platform
- Firewalling is now supported and automatically configured if needed
- PuppetDB and Puppet Master (running Passenger) are now provisionned by Puppet manifests and not config-tools anymore
- Linuxbridge ML2 plugin support in Neutron
- Swift backend support in Glance
- Dell EQLX Backend support in Cinder
- Sanity job is now part of Spinal-Stack and has left CI scripts to be included in config-tools, so sanity can be run out of the box
- Environment YAML file now supports host ranges
- Ansible configuration is now automatically generated so upgrade is easier

Bugs fixed since I.1.2.1

Redhat

- Fix openstack-ceilometer-central with Pacemaker/Systemd

Debian

- None

Security fixes

- Redhat
  - CVE-2014-9322: privilege escalation through #SS segment
  - CVE-2014-9295: ntp remote code execution
  - CVE-2014-3566: nss fixes for POODLE
  - CVE-2014-7821: Neutron DoS through invalid DNS configuration
  - Still vulnerable to
    - CVE-2014-8124: Horizon denial of service attack through login page
    - No CVE yet (OSSA 2014-041): Glance v2 API unrestricted path traversal: fixed in Puppet by changing policy.json by default. It may affect some Glance features.
- Debian
  - CVE-2014-9295: ntp remote code execution

Components in this release

- edeploy-roles-I.1.3.0.tar.gz source code of the eDeploy roles built in this release.
- install-server-D7-I.1.3.0.edeploy binary eDeploy role for the installation server (puppet master, logging server, monitoring server, edeploy server, dnsmasq, serverspec).
- openstack-full-{D7;RH7.0}-I.1.3.0.edeploy binaries eDeploy roles for OpenStack and Ceph nodes.

- puppet-openstack-cloud-I.1.3.0.tgz puppet modules used to configure the OpenStack and Ceph nodes.
- serverspec-I.1.3.0.tgz serverspec tests to validate deployments.

All the sources have been tagged with the tag I.1.3.0 in their respective repositories. Upgrade is supported from I.1.2.0 or I.1.2.1 to I.1.3.0 on Debian and Red Hat systems.

Documentation can be found on http://spinalstack.enovance.com/en/latest/, fixes are welcome at https://github.com/enovance/spinalstack-doc

Packages

- Red Hat 7.0
  - Full list of packages for the install-server for Red Hat
  - Full list of packages for the openstack-full for Red Hat
  - Diff between I-1.2.1 and I-1.3.0 for Red Hat
- Debian 7
  - Full list of packages for the install-server for Debian
  - Full list of packages for the openstack-full for Debian
  - Diff between I-1.2.1 and I-1.3.0 for Debian

I-1.2.1

Spinal Stack I-1.2.1

OpenStack version and supported distros

- Red Hat Enterprise Linux 7: OSP5 (Icehouse)
- Debian Wheezy: Icehouse 2014.1.2

No new feature

Bugs fixed since I.1.2.0

- eDeploy bug with detection and debug message
- Puppetfile: fix NFS git commit ID
- Heat: switch deferred_auth_method to 'password'
- Keystone: don't specify API version in {public,admin}_endpoints
- Keystone: update puppet-keystone git ref to fix a bug upstream when compiling the catalog.

eDeploy roles have also been updated.

Components in this release

- edeploy-roles-I.1.2.1.tar.gz source code of the eDeploy roles built in this release
- install-server-D7-I.1.2.1.edeploy binary eDeploy role for the installation server (puppet master,logging server, monitoring server, edeploy server, dnsmasq, serverspec)

- openstack-full-{D7;RH7.0}-I.1.2.1.edeploy binaries eDeploy roles for OpenStack and Ceph nodes
- puppet-openstack-cloud-I.1.2.1.tgz puppet modules used to configure the OpenStack and Ceph nodes
- serverspec-I.1.2.1.tgz serverspec tests to validate deployments

All the sources have been tagged with the tag I.1.2.1 in their respective repositories. Documentation can be found on http://spinalstack.enovance.com/en/latest/, fixes are welcome at https://github.com/enovance/spinalstack-doc

Packages

- Red Hat 7.0
  - Full list of packages for the install-server for Red Hat
  - Full list of packages for the openstack-full for Red Hat
  - Diff between I-1.2.0 and I-1.2.1 for Red Hat
- Debian 7
  - Full list of packages for the install-server for Debian
  - Full list of packages for the openstack-full for Debian
  - Diff between I-1.2.0 and I-1.2.1 for Debian

I-1.2.0

Spinal Stack I-1.2.0

OpenStack version and supported distros

- Red Hat Enterprise Linux 7: OSP5 (Icehouse)
- Debian Wheezy: Icehouse 2014.1.2

Major Features

- RHEL 7/OSP 5 support
- [Experimental] Sensu is the first implementation of monitoring in Spinal Stack. You can access the dashboard on: http://<install-server>:3000/
- Glance now supports NFS image storage backend
- Cinder now supports EMC VNX & iSCSI volume backends
- Nova now supports NFS instance storage backend
- [Experimental] Neutron now supports Cisco plugins for N1KV hardware
- RabbitMQ can now be load-balanced by HAproxy
- Keystone roles for Heat are now created automatically
- Support for keepalived authentication
- MongoDB replicaset is now optional, MongoDB can be used in standalone mode
- MySQL Galera has been tweaked to improve performance at scale
- Nova configuration has been tweaked to use the read-only database feature to improve performance at scale
- When running KVM, we check if VTX is really enabled

- Trove has been disabled by default since it's still experimental
- HAproxy: Allow user to bind multiple public/private IPs
- HAproxy checks have been improved for OpenStack services
- keepalived: allow vrrp traffic on a dedicated interface
- Neutron: allow to specify tunnel type (i.e. VXLAN)
- Horizon: allow to configure specific parameters in Apache2 vhost
- Enable RBD support for Red Hat OSP5
- MOTD can be customized
- Keepalived has now a second VIP binding which stays optional (i.e. for internal network)
- First iteration of NFV support in eDeploy roles for RHEL7 with OSP5
- It's now possible to add specific parameters to Horizon's Apache2 vhost
- Flexible YAML configuration and multiple deployments scenarios supported, see openstack-yaml-infra/scenarios

Bugs fixed since I.1.1.0

- TSO issues: now disabled by default to avoid network performances issues (both east-west & north-south)
- Fix correct Puppet Ceph dependencies which could lead to bootstraping issues
- Fix issues with instance live migration support (nova configuration)
- Fix HAproxy checks for Spice (TCP instead of HTTP)
- Clean Pacemaker resources when an error occurs (usually during bootstrap)

Security

- Shellshock has been fixed in new eDeploy roles
- some OpenStack CVE fixed in Icehouse (details in release launchpad)

Upgrades

- You can now see the packages that changed between releases
- Ansible playbooks to upgrade from Debian Wheezy I.1.1.0 to I.1.2.0 on edeploy-roles/upgrade, this is the recommended upgrade method
- From now on upgrades will also be supported on RHEL7 platform so you will be able to upgrade from I.1.2.0 to I.1.3.0 on both Debian & Red Hat.
- We validated 1741 functional tests with Tempest
- DO NOT use the master branch going forward, use the release tag I.1.2.0

Components in this release

- edeploy-roles-I.1.2.0.tar.gz source code of the eDeploy roles built in this release.
- install-server-D7-I.1.2.0.edeploy binary eDeploy role for the installation server (puppet master, logging server, monitoring server, edeploy server, dnsmasq, serverspec).

- openstack-full-{D7;RH7.0}-I.1.2.0.edeploy binaries eDeploy roles for OpenStack and Ceph nodes.
- puppet-openstack-cloud-I.1.2.0.tgz puppet modules used to configure the OpenStack and Ceph nodes.
- serverspec-I.1.2.0.tgz serverspec tests to validate deployments.

All sources have been tagged with the tag I.1.2.0 in their respective repositories.

Documentation can be found on http://spinalstack.enovance.com/en/latest/, fixes are welcome at https://github.com/enovance/spinalstack-doc

Packages

- Red Hat 7.0
  - Full list of packages for the install-server for Red Hat
  - Full list of packages for the openstack-full for Red Hat
  - Diff between I-1.1.0 and I-1.2.0 for Red Hat
- Debian 7
  - Full list of packages for the install-server for Debian
  - Full list of packages for the openstack-full for Debian
  - Diff between I-1.1.0 and I-1.2.0 for Debian

I-1.1.0

Icehouse I-1.1.0

Features

- SSL termination suppot on load-balancer nodes
- Centralized logging support (Stack: fluentd + elasticsearch + kibana)
- Puppet dashboard is now installed by default in install-server and available at http://install-server:82/
- Heat-engine is no more managed by Pacemaker and can be run on all controller nodes
- Pacemaker support on Red Hat
- OpenStack database as a service (Trove) support
- Glance local backend support
- Add galera-status script to check MySQL cluster status
- Add jenkins on the install-server which manages Puppet, Sanity and Upgradejobs
- Ansible playbooks to upgrade from Debianwheezy I-1.0.0 to I.1.1.0

Release Components

- edeploy-I-1.1.0.tar.gz
- edeploy-roles-I-1.1.0.tar.gz
- install-server-D7-I.1.1.0.edeploy
- openstack-full-D7-I.1.1.0.edeploy

- openstack-full-RH6.5-I.1.1.0.edeploy
- puppet-openstack-cloud-I.1.1.0.tgz
- serversoec-I.1.1.0.tgz

All the sources have been tagged with the tag I.1.1.0 in their respective repositories on Github

Package Diff

Added

Package	Version
crash	6.0.6-1
dnsutils	1:9.8.4.dfsg.P1-6+nmu2+deb7u1
iotop	0.4.4-4
less	444-4
ltrace	0.5.3-2.1
manpages	3.44-1
mtr	0.82-3
ncdu	1.8-1
nmap	6.00-0.3+deb7u1
percona-toolkit	2.1.2-1
python-designateclient	1.0.3-1~bpo70+1
python-trove	2014.1-5~bpo70+1
shared-mime-info	1.0-1+b1
socat	1.7.1.3-1.4
strace	4.5.20-2.3
sysstat	10.0.5-1
telnet	0.17-36
tmux	1.6-2
trove-api	2014.1-5~bpo70+1
trove-common	2014.1-5~bpo70+1
trove-conductor	2014.1-5~bpo70+1
trove-guestagent	2014.1-5~bpo70+1
trove-taskmanager	2014.1-5~bpo70+1
w3m	0.5.3-8

Updated

Package	Old Version	New Version
cinder-api	2014.1.1-2~bpo70+1	2014.1.1-3~bpo70+1
cinder-backup	2014.1.1-2~bpo70+1	2014.1.1-3~bpo70+1
cinder-common	2014.1.1-2~bpo70+1	2014.1.1-3~bpo70+1
cinder-schedule	2014.1.1-2~bpo70+1	2014.1.1-3~bpo70+1
cinder-volume	2014.1.1-2~bpo70+1	2014.1.1-3~bpo70+1
cloud-initramfs-growroot	0.18.debian3~bpo70+1	0.8.debian5
facter	2.0.2-1puppetlabs1	2.1.0-1puppetlabs1
haproxy	1.4.25-1~bpo70+1	1.5~dev26-2~bpo70+1
keystone	2014.1.1-2~bpo70+1	2014.1.1-3~bpo70+1
linux-image-3.2.2-4-amd64	3.2.57-3+deb7u2	3.2.60-1+deb7u1
nova-api	2014.1.1-3~bpo70+1	2014.1.1-7~bpo70+1
nova-cert	2014.1.1-3~bpo70+1	2014.1.1-7~bpo70+1
nova-common	2014.1.1-3~bpo70+1	2014.1.1-7~bpo70+1
nova-compute	2014.1.1-3~bpo70+1	2014.1.1-7~bpo70+1
nova-compute-kvm	2014.1.1-3~bpo70+1	2014.1.1-7~bpo70+1
nova-conductor	2014.1.1-3~bpo70+1	2014.1.1-7~bpo70+1
nova-consoleauth	2014.1.1-3~bpo70+1	2014.1.1-7~bpo70+1
nova-consoleproxy	2014.1.1-3~bpo70+1	2014.1.1-7~bpo70+1
openstack-dashboard	2014.1.1-2~bpo70+1	2014.1.1-3~bpo70+1
python-cinder	2014.1.1-2~bpo70+1	2014.1.1-3~bpo70+1
python-compressor	1.3-1~bpo70+1	1.4-1~bpo70+1
python-keystone	2014.1.1-2~bpo70+1	2014.1.1-3~bpo70+1
python-nova	2014.1.1-3~bpo70+1	2014.1.1-7~bpo70+1
python-django-horizon	2014.1.1-2~bpo70+1	2014.1.1-3~bpo70+1
td-agent	1.1.20-1	2.0.3-0

Removed

Package	Version
dkms	2.2.0.3-1.2
libssl0.9.8	0.9.8o-4squeeze14
linux-hearder-3.2.0-4-amd64	3.2.57-3+deb7u2
linux-hearder-3.2.0-4-common	3.2.57-3+deb7u2
linux-libc-dev:amd64	3.2.57-3+deb7u2
openvswitch-datapath-dkms	2.0.1.1~bpo70+1

Bug Solved TBD

Known Bug TBD

Parameter	Type	Optional	Description	Defa
root_password	String	False	root password to access the vm via ssh	
domain	String	False	Domain name used to configure all the servers and services	
region	String	False	Region used in Spinal Stack	
dns_ips	String	False	Servername ip address	
ntp_servers	Array	True	List of NTP servers to sync with	
vip_public_ip	String	False	Public VIP IP	
vip_admin_ip	String	False	Admin VIP IP	
vip_public_fqdn	String	False	Public FQDN	

Continued on n

Table 1.1 – continued from previous page

Parameter	Type	Optional	Description	Defa
vip_internal_fqdn	String	False	Internal FQDN	
vip_admin_fqdn	String	False	Admin FQDN	
endpoint_proto	String	False	Endpoint protocol type. (ex. http, https)	
public_network	String	False	CIDR notation of the public network	
internal_network	String	False	CIDR notation of the internal network	
admin_network	String	False	CIDR notation of the admin network	
storage_network	String	False	CIDR notation of the storage network	
public_netif	String	False	Public network interface	
internal_netif	String	False	Internal network interface	
admin_netif	String	False	Admin network interface	
storage_netif	String	False	Storage network interface	
external_netif	String	False	External network interface	
public_netif_ip	String	True	Public interface IP address	ipadd
internal_netif_ip	String	True	Internal interface IP address	ipadd
admin_netif_ip	String	True	Admin interface IP address	ipadd
storage_netif_ip	String	True	Storage interface IP address	ipadd
db_allowed_hosts	Array	False	Array of MySQL style Host connections are allowed from	
galera_master_name	String	False	Galera master hostname	
mysql_root_password	String	False	MySQL root password	
mysql_sys_maint_password	String	False	MySQL maintener password	
galera_clustercheck_dbuser	String	False	Galera cluster check user	
galera_clustercheck_dbpassword	String	False	Galera cluster check user password	
rabbit_host	String	False	RabbitMQ host	
rabbit_password	String	False	RabbitMQ password	
secret_key	String	False	** TBD **	
cinder_db_password	String	False	Cinder database password	
ks_cinder_password	String	False	ks_cinder_password	
heat_db_password	String	False	Heat database password	
ks_heat_password	String	False	ks_heat_password	
heat_auth_encryption_key	String	False	Heat authentication encryption key	
glance_db_password	String	False	Glance database password	
ks_glance_password	String	False	ks_glance_password	
ceilometer_secret	String	False	ceilometer secret	
ks_ceilometer_password	String	False	ks_ceilometer_password	
ceph_fsid	String	False	ceph_fsid	
ceph_mon_secret	String	False	Monitor secret	
ceph_osd_devices	Array	False	Array of osd devices	
cinder_rbd_pool	String	False	RBD pool for cinder	
cinder_rbd_user	String	False	RBD user for cinder	
keystone_db_password	String	False	Keystone database user	
ks_admin_email	String	False	Keystone administrator email	
ks_admin_password	String	False	Keystone administrator password	
ks_admin_token	String	False	Keystone administrator token	
ks_nova_password	String	False	ks_nova_password	
nova_db_password	String	False	Nova database password	
nova_ssh_public_key	String	False	Nova ssh public key	
nova_ssh_private_key	String	False	Nova ssh private key	
libvirt_type	String	False	Libvirt type	
neutron_db_password	String	False	Neutron database password	

Continued on n

Table 1.1 – continued from previous page

Parameter	Type	Optional	Description	Defa
ks_neutron_password	String	False	ks_neutron_password	
neutron_metadata_proxy_shared_secret	String	False	Shared secret	
trove_db_password	String	False	Trove database password	
ks_trove_password	String	False	ks_trove_password	
ks_swift_dispersion_password	String	False	Swift dispersion	
ks_swift_password	String	False	Keystone Swift password	
replicas	Integer	False	Swift number of replicas	
statsd_host	String	False	Statsd host IP address	
swift_hash_suffix	String	False	Hash suffix	
syslog_port	Integer	False	Remote syslog port to export to	
syslog_server	String	False	Remote syslog ip address to export to	
haproxy_auth	String	False	HAProxy admin interface credentials	

Environment Parameters

1.9.2 Havana

No changelogs were available during the Havana cycle

1.10 Security Advisories

1.10.1 OSSA-2014-041

Announce: http://lists.openstack.org/pipermail/openstack-announce/2014-December/000317.html

Impacted versions : Every version until Kilo Fixable versions: Starting from I-1.3.0

How Spinal Stack prevents it ?

To prevent OSSA-2014-041 on your deployed platforms, a user needs to set specific policies for Glance API via the glance_api_policies configuration setting in the environment file.

The needed policies are the following :

> 'delete_image_location': 'role:admin' 'get_image_location': 'role:admin' 'set_image_location': 'role:admin'

By default, if the user does not specify any glance_api_policies in his configuration file, those parameters will be added automatically else the user will need to add them manually in addition of the policies she's adding.

Find below the snippet to use

```
glance_api_policies:
  delete_image_location:
    key: delete_image_location
    value: 'role:admin'
  get_image_location:
    key: get_image_location
    value: 'role:admin'
  set_image_location:
    key: set_image_location
    value: 'role:admin'
```

To completely disable Glance API policies, set the following in your environment file

```
glance_api_policies: false
```

1.10.2 CVE-2015-3456 (aka VENOM)

Announce: venom.crowdstrike.com Affects: versions through J-1.4.0

Description: An attacker with root priviliged in a guest instance (either a malicious cloud user or through a remote access to the vm), can exploit a flaw in the Qemu hypervisor and potencially execute code on the compute node resulting in a vm escape.

Am I affected ?

If you have untrusted cloud user or hosting internet facing application then you are at risk.

How to get protected from VENOM ?

In order to get an early fix and avoid a full upgrade of the plateform, here is a simple procedure to apply manually on each compute node:

1/ Activate package manager:: # edeploy activate-pkgmngr

2/ Register to Red Hat CDN. (Please read this documentation)

3/ Update the Qemu package:: # yum update qemu-kvm-rhev

4/ Deactivate package manager:: # edeploy deactivate-pkgmngr

5/ Following the update, the guests (virtual machines) need to be powered off and started up again for the update to take effect. It is also possible to migrate guests away from the affected host, update the host, and then migrate the guests back. Please note that it is not enough to restart the guests because a restarted guest would continue running using the same (old, not updated) QEMU binary:

```
    # nova evacuate
```

Will I get protected after a full upgrade ?

Unless the new version is J-1.4.0, the qemu package is not fixed and the above procedure needs to be re-applied.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search