bootmachine Documentation

Release 0.6.0

Thomas Schreiber

April 20, 2015

Contents

Contents:

CHAPTER 1

bootmachine

Bootmachine is a bootstrapping tool for securely provisioning virtual servers up until the point where customized configuration management begins.

The bootmachine's goal is to allow getting started with, maintenance of, and exploring new stack options through a simple, pluggable and highly customizable interface.

1.1 Configuration Management Tools

Currently supported:

- Salt http://saltstack.org
- Write your own

Next in the queue:

- Chef http://www.opscode.com/chef/
- Puppet http://puppetlabs.com/
- Write your own

1.2 Providers

Currently supported:

- Rackspace Openstack Compute API v2 via python-novaclient
- Non-rackspace Openstack Compute via python-novaclient
- Rackspace Openstack Compute API v1 via openstack.compute (deprecated) http://www.rackspace.com/cloud/
- Write your own

Next in the queue:

- Amazon EC2 via boto https://aws.amazon.com/ec2/
- Virtualbox / Vagrant http://vagrantup.com/
- Write your own

1.3 Distros

Currently supported:

- Arch Linux
- Ubuntu
- Fedora
- Debian
- Write your own

Next in the queue:

- FreeBSD
- Gentoo
- openSUSE
- CentOS
- RHEL
- Write your own

Documentation about the usage and installation of the bootmachine can be found at: http://bootmachine.readthedocs.org

The source code and issue tracker can be found on GitHub: https://github.com/rizumu/bootmachine

Motivation

There are many options when it comes to server providers, configuration management tools, and distros. The bootmachine's goal is to reduce maintenance and overhead when managing a server stack, but through its pluggable api it also simplifies exploring new options.

The bootmachine is written in PEP8 compliant Python and is at its most basic, simply a specialized Fabric libary.

Providers:

- Rackspace Openstack API v2
- Rackspace Openstack API v1 (deprecated)
- Amazon EC2 (forthcoming)
- Write your own..

Configurators:

- Salt http://saltstack.org/
- Chef (forthcoming)
- Puppet (forthcoming)
- Write your own...

Distros:

- Arch Linux
- Fedora 16+17
- Ubuntu 12.04 LTS
- Debian 6
- Gentoo 11.0 (forthcoming)
- OpenSUSE 12 (forthcoming)
- CentOS 6.2 (forthcoming)
- Red Hat Enterprise Linux 6 (forthcoming)
- write your own...

First, the bootmachine boots each new server as defined in its `settings.py` using the distro and provider of your choice. Next it bootstaps setting up the distro and installs the configuration management tool of your choice. Finally it uses the configuration management tool to provision the server to a secure state. The supplied states/recipes handle iptables, ssh, users, and not much more. The idea is to keep things simple, but it is up to you to customize the stack to your preferences.

For example, you could create a new stack with four Arch Linux servers using the Rackspace api. After defining your settings and running *fab bootmachine*. Salt is installed and the server is configured as the states tell it to be. The stack could contain a loadbalancer, cache, application, and database server. It is up to you to define the individual roles, but the bootmachine gets you as close as it can to a secure and ready system before personalization takes over.

CHAPTER 3

Booting

Bootmachine simplifies the initial boot phase of creating new servers, by reading configuration from `settings.SERVERS`. Choose from the built-in cloud providers, or write a custom `settings.PROVIDER_MODULE`.

Note: Write a custom backed if your intention is to support local virtual machines, an not yet included cloud provider, or hardware in a private datacenter. If your module is generic enough to share with others, please consider contributing it back to the bootmachine.

CHAPTER 4

Provisioning with Configuration Management Tools

The bootmachine reads the settings file and checks for servers that are not yet booted. It will then boot each new server, with its defined distro and size, and next bootstrap the configuration management tool of your choice.

Note: The bootmachine supports a stack with multiple distros, but it is assumed that only *one* provider and *one* configuration management tool will be used per stack.

Although you may boot a mixture of distro types, it is advised against because this will most likely create unnecessary complexity down the road. Mainly because the configuration management tools could have conflicting versions per distro. If you know what your doing than go ahead, otherwise be warned.

After your configuration management tool of choice is bootstrapped on the new servers, the last step is provisioning the server to a secure state. For this it following community approved best-practices in the supplied salt-states, chef/puppet recipes, etc.

Bootmachine adheres to the Slicehost provisioning documentation and the Arch Linux wiki:

- Slicehost provisioning docs: http://articles.slicehost.com/ubuntu-10
- Arch Linux wiki : https://wiki.archlinux.org/

CHAPTER 5

What's Next

The function of the bootmachine is to create a new server, or cluster of servers, based on a configuration file and provision them into a secure state. It is the job of a configuration management tool to setup the server for its real task, such as application server, database server, loadbalancer, etc.

Every application is different, have fun.

CHAPTER 6

Installation

Bootmachine runs locally on OSX or GNU/Linux.

First clone the bootmachine to an easily accessible folder on your local machine. The recommended place is `/srv/` but any folder in your home directory would work just the same.

If you are unfamiliar with Python's pip or virtualenv packages, understand them, at least a little, before proceeding:

```
* http://pypi.python.org/pypi/virtualenv
* http://pypi.python.org/pypi/pip
```

If you don't already have a virtualenv that you would like to use, first create a new virtualenv for the bootmachine:

```
$ cd ~/.virtualenvs/
$ virtualenv --no-site-packages --distribute bootmachine
$ source bootmachine/bin/activate
```

Install the bootmachine:

```
$ pip install bootmachine
```

From within the directory you want the configuration files to be copied to, execute the following command:

```
$ bootmachine-admin start
```

Now, customize the example settings file and for your stack. A suggestion is to instead store these private files a private git repository.

Some info on choosing the type of encryption for your ssh key:

```
https://wiki.archlinux.org/index.php/SSH_Keys#Generating_an_SSH_key_pair
http://pthree.org/2011/02/17/elliptic-curve-cryptography-in-openssh/
```

Note: Elliptic curve cryptography is excluded from Fedora presumably due to patent concerns. Check Fedora 18 once released. http://comments.gmane.org/gmane.linux.redhat.fedora.legal/1576

To use Rackspace's openstack api v2 you must also set some environment variables for your interactive shell. The recommended installation is to add the following to your `~/.bashrc`:

```
export OS_USERNAME=""  # your rackspace username
export OS_PASSWORD=""  # your rackspace password
export OS_TENANT_NAME=""  # your rackspace accountid
export OS_AUTH_URL="https://identity.api.rackspacecloud.com/v2.0/"
export OS_REGION_NAME="DFW"
export OS_COMPUTE_API_VERSION="2"
```

For Rackspace api v1 (deprecated):

```
export OPENSTACK_COMPUTE_USERNAME=""  # your rackspace username
export OPENSTACK_COMPUTE_APIKEY=""  # your rackspace apikey
```

The same is true for the Amazon boto api:

```
export AWS_ACCESS_KEY_ID=""  # your amazon access key id
export AWS_SECRET_ACCESS_KEY=""  # your amazon secret access key
```

CHAPTER 7

bootmachine overview and usage

The `bootmachine-admin start` command simply copies two files to the current working directory. A standard Fabric `fabfile.py` and a `settings.py` for which you need to customize. Additionally it copies over the `configuration` folder containing the initial states/recipes to configure the servers.

After customizing your settings, all it takes to convert bare metal servers from aluminium into rhodium is one simple Fabric command:

```
$ fab bootmachine
```

Internally this does two things. First `provider.bootem` checks if there are any non-booted servers listed in the `settings.PROVIDER_BACKENDS`. If `provider.boot` finds a non-booted server, it will boot it in parallel. In the meantime `provider.bootem` queries the provider to ensure that all servers are `ACTIVE` before continuing.

Second, after all servers are found to be active, the `bootstrap` method is called to check if there are any servers which have not yet been bootstrapped. These servers are then bootstrapped in parallel.

Note: These commands can also be run separately:

```
$ fab boot
$ fab each bootstrap_distro
$ fab each bootstrap_configurator
$ fab master configure
```

After provisioning is complete you can manually login, with the user credentials and port as defined in your settings.py, to the machine using the following format:

```
$ ssh -p {port} {username}@{ip}
```

To list the details for your new machines, including ip addresses:

```
$ fab provider
```

Or if you already have `openstack-compute` or `python-novaclient` installed, you could just as easily:

```
$ openstack-compute list
    or
$ nova list
```

All available commands can be seen by typing:

```
$ fab -l  # which is short for fab --list
```

CHAPTER 8

Bootmachine Changelog

8.1 0.6.1 (development)

CHANGES:

- add support for arch20147 with pacman.
- drop yaourt complexities.

8.2 0.6.0 (25.05.2013)

CHANGES:

- add support for debian7, arch20132, fedora18 and ubuntu1304 rackspace images.
- drop support for arch201208 and fedora16 rackspace images
- bump Arch salt version to 0.11.1
- require the salt package and release version in the settings, when building salt on a rolling release such as Arch.
- Arch 2012.08 is now a pure systemd installation
- nova client now requires specifying pub and priv networks during a build. sleep longer during boot stage to prevent extra api calls to rackspace.
- refactor salt states and move them to a new folder.
- add support for changing the remote directory of states and pillars.
- bump Fabric to 1.5.1, later to 1.6 (which helpfully resolved some rebooting issues)
- bump python-novaclient to 2.13.0
- bump Jinja2 to 2.7

8.3 0.5.9 (25.10.2012)

CHANGES:

- resolve issue #8 by using pgrep to check if salt-master/minion daemons instead of relying on Fabric.
- deprecate explicit support for rackspace api v1

8.4 0.5.4, 0.5.5, 0.5.6, 0.5.7, 0.5.8 (13.09.2012)

CHANGES:

- fix setup.py and use a MANIFEST.in to include configuration files

8.5 0.5.3 (13.09.2012)

CHANGES:

- some fixes to the provided salt-states

8.6 0.5.2 (13.08.2012)

NEW FEATURES:

- debian 6 is now supported on rackspace with salt

CHANGES: * the *fab all* task has been renamed to *fab each*, mostly to

  avoid the name clash with the Python's global all() method.

- the *fab provision* task has been renamed to *fab bootstrap* and a new *fab configure* task has been added to better clarify intent. Additionaly a thorough refactoring of core.py has taken place.
- instead of aborting, a y/n continue prompt has been added when a rackspace server found to be not *ACTIVE*.

8.7 0.5.1 (23.07.2012)

NEW FEATURES:

- add a test runner that runs all builds, logs output and scans for failures
- add a requirements.txt, so installing in a new virtualenv is simpler when working on the bootmachine
- friendlier time counter while waiting for servers to boot
- add a warning prompt to require confirmation before deleting servers, with an option to force.
- changed fab all reboot_servers() to fab reboot_server(servername)
- added an internal __set_ssh_vars(valid_object) method. After performing a few sanity tests this method adds reliable ssh variables to the passed in object (env or server).

SALT:

- add a bootmachine-pillar/deploymachine.sls as an example of where post bootmachine pillar data can be stored. Bootmachine boots your servers, deploymachine is the states for your custom stack.
- fix require relationship between users/ssh/iptables states
- fix iptables issues and simplify the salt-state

ARCH LINUX:

- upgrade salt-state for grub to install and use grub2
- fix the recent glibc update that broke the build

- add a salt-state for the rc.conf
- add a way for the saltmaster to open a port for newly booted minions
- use hostname instead of ip in the salt-minion config
- follow netcfg best practice, by removing networking settings from rc.conf

8.8 0.5.0 (23.07.2012) – Initial release.

The bootmachine grew out of the desire to automate the launching, configuration, and scaling of a stack of servers in the cloud.

The provider, configurator, and distro functionality has been written in such a way that each module is pluggable. Therefore customization and extension can be achieved with little effort.

The bootmachine has existing modules for Rackspace, Salt and a handful of distros. Additional modules could easily be written to support EC2, Chef, Puppet, and other distros.

Any contributions to the core or submissions of new modules for the contrib will be much appreciated. I'd like to see this project allow developers to switch between providers with ease, simplify the process of configuring a cloud stack, and encourage experimentation with new distros.

This is not a 1.0 yet, but please give it a try. It has been working well for me and I'm excited about this first public release.

Github page: https://github.com/rizumu/bootmachine

CHAPTER 9

Indices and tables

- *genindex*
- *modindex*
- *search*