Installation of Salt with Nginx Servers

Systems Administration is changing, with the huge scale of internet company deployments and the popularity of cloud computing. Server deployments are often scaling faster than the systems administration teams supporting them. In order to meet the demand those teams are finding themselves changing the ways they have traditionally managed servers.

One of those changes is automation, where once a sysadmin would need to spend time installing packages by hand (via apt or yum) and modifying configuration files. Systems Automation tools have taken that task and pushed it to policy based configurations that install and configure these packages automatically. Leaving Sysadmins free to perform other tasks, like automating more tasks & finding solutions to the fires that plague Sysadmins daily.

There are many tools out in the wild for Systems Automation, the major players in the Open Source arena are [Puppet](http://puppetlabs.com/), [CFEngine](http://cfengine.com/) and [Chef](http://www.opscode.com/). These tools all have their quirks and benefits and unfortunately I have not used them all to the extent to tell you which one is better than the other.

In this article, I am going to highlight a newer player to the Systems Automation scene; [SaltStack](http://saltstack.com/).

## About SaltStack

SaltStack or Salt as it is commonly referred to, is an extremely flexible and easy to learn automation tool. The speed of which one can learn and effectively use SaltStack is what really sets it apart. With the previously mentioned tools a Sysadmin must learn Ruby or a product specific programming language.

With SaltStack the most basic usage is simple without any programming required; however, if you want you can extend Salt's functionality with your own Python code. Salt is written in Python, which makes it a bit easier to justify implementing as Python is becoming a favorite amongst Sysadmins which decreases the learning curve for Salt.

## Setting up the Salt-Master

Salt servers have two types, Master and Minion. The master server is the server that hosts all of the policies and configurations and pushes those to the various minions. The minions, are the infrastructure that you want managed. All of the pushed information is communicated via ZeroMQ; this communication is also encrypted and minions must be authenticated on the master before receiving any commands/configurations.

### Installing on Ubuntu

I will be showing you how to install Salt on Ubuntu; however if you want to install Salt on other distributions you can find instructions and a bootstrap script at [docs.saltstack.com](http://docs.saltstack.com/topics/installation/index.html).

#### Installing Python Software Properties

Saltstack maintains a PPA (Personal Package Archive) that can be added as an apt repository. On my systems before I could add a PPA Repository I had to install the python-software-properties package.

root@saltmaster:~# apt-get --yes -q install python-software-properties

#### Adding the SaltStack PPA Repository

root@saltmaster:~# add-apt-repository ppa:saltstack/salt

You are about to add the following PPA to your system:

Salt, the remote execution and configuration management tool.

More info: https://launchpad.net/~saltstack/+archive/salt

Press [ENTER] to continue or ctrl-c to cancel adding it

Make sure that you press **[ENTER]** otherwise the repository will not be added.

##### Update Apt's Package Indexes

After adding the repository make sure that you update Apt's package index.

root@saltmaster:~# apt-get --yes -q update

#### Install The Salt-Master package

root@saltmaster:~# apt-get --yes -q install salt-master

### Configuring The Salt Master

Now that Salt has been installed, we will configure the master server. Unlike many other tools the configuration of SaltStack is pretty simple. This article is going to show a very simple "get you up and running" configuration. I will make sure to cover more advanced configurations in later articles.

In order to configure the salt master we will need to edit the /etc/salt/master configuration file.

root@saltmaster:~# vi /etc/salt/master

#### Changing the bind interface

Salt is not necessarily push only, the salt minions can also send requests to the salt master. In order to ensure that this happens we will need to tell salt which network interface to listen to.

**Find:**

# The address of the interface to bind to

#interface: 0.0.0.0

**Replace with:**

# The address of the interface to bind to

interface: youripaddress

**Example:**

# The address of the interface to bind to

interface: 192.168.100.102

#### Setting the states file\_roots directory

All of salt's policies or rather salt "states" need to live somewhere. The file\_roots directory is the location on disk for these states. For this article we will place everything into /salt/states/base.

**Find:**

#file\_roots:

#base:

#- /srv/salt

**Replace with:**

file\_roots:

base:

- /salt/states/base

Not all states are the same, sometimes you may want a package to be configured one way in development and another in production. While we won't be covering it yet in this article you can do this by using salt's "environments" configuration.

Each salt master must have a base environment, this is used to house the top.sls file which defines which salt states apply to specific minions. The base environment is also used in general for states that would apply to all systems.

For example, I love the screen command and want it installed on every machine I manage. To do this I add the screen state into the base environment.

To add additional environments simply append them to the file\_roots configuration.

**Adding the development environment:**

file\_roots:

base:

- /salt/states/base

development:

- /salt/states/dev

#### Setting the pillar\_roots

While this article is not going to cover pillars (I will add more articles for salt don't worry) I highly suggest configuring the pillar\_roots directories as well. I have found that pillars are extremely useful for reusing state configuration and reducing the amount of unique state configurations.

**Find:**

#pillar\_roots:

#base:

#- /srv/pillar

**Replace:**

pillar\_roots:

base:

- /salt/pillars/base

Pillars also understand environments, the method to adding additional environments is the same as it was for file\_roots.

#### Restart the salt-master service

That's all of the editing that we need to perform for a basic salt installation. For the settings to take effect we will need to restart the salt-master service.

root@saltmaster:~# service salt-master restart

salt-master stop/waiting

salt-master start/running, process 1036

#### Creating the salt states and pillars directories

Before we move on to the salt minion's installation we should create the file\_roots and pillar\_roots directories that we specified in /etc/salt/master.

root@saltmaster:~# mkdir -p /salt/states/base /salt/pillars/base

## Setting up the Salt-Minion

Now that the salt master is setup and configured we will need to install the salt-minion package on all of the systems we want salt to manage for us. Theoretically once these minions have been connected to the salt master, you could get away with never logging into these systems again.

### Installing on Ubuntu

The below process can be repeated on as many minions as needed.

#### Installing Python Software Properties

root@saltminion:~# apt-get --yes -q install python-software-properties

#### Adding the SaltStack PPA Repository

root@saltminion:~# add-apt-repository ppa:saltstack/salt

You are about to add the following PPA to your system:

Salt, the remote execution and configuration management tool.

More info: https://launchpad.net/~saltstack/+archive/salt

Press [ENTER] to continue or ctrl-c to cancel adding it

Make sure that you press **[ENTER]** otherwise the repository will not be added.

##### Update Apt's Package Indexes

After adding the repository make sure that you update Apt's package index.

root@saltminion:~# apt-get --yes -q update

#### Install The Salt-Minion package

root@saltminion:~# apt-get --yes -q install salt-minion

### Configuring the Salt-Minion

Configuring the salt minion is even easier than the salt master. In simple implementations like the one we are performing today all we need to do is set the salt master IP address.

root@saltminion:~# vi /etc/salt/minion

#### Changing the Salt-Master target IP

**Find:**

#master: salt

**Replace with:**

master: yourmasterip

**Example:**

master: 192.168.100.102

By default the salt-minion package will try to resolve the "salt" hostname. A simple trick is to set the "salt" hostname to resolve to your salt-master's IP in the /etc/hosts file and allow the salt-master to push a corrected /etc/salt/minion configuration file. This trick let's you setup a salt minion server without having to edit the minion configuration file.

#### Restarting the salt-minion service

In order for the configuration changes to take effect, we must restart the salt-minion service.

root@saltminion:~# service salt-minion restart

salt-minion stop/waiting

salt-minion start/running, process 834

### Accepting the Minions key on the Salt-Master

Once the salt-minion service is restarted the minion will start trying to communicate with the master. Before that can happen we must accept the minions key on the master.

#### On the salt master list the salt-key's

We can see what keys are pending acceptance by running the salt-key command.

root@saltmaster:~# salt-key -L

\*\*Accepted Keys:\*\*

\*\*Unaccepted Keys:\*\*

saltminion

\*\*Rejected Keys:\*\*

#### Accept the saltminion's key

To accept the saltminion's key we can do this two ways, via the saltminions specific name or accept all pending keys.

##### **Accept by name:**

root@saltmaster:~# salt-key -a saltminion

The following keys are going to be accepted:

Unaccepted Keys:

saltminion

Proceed? [n/Y] Y

Key for minion saltminion accepted.

##### **Accept all keys:**

root@saltmaster:~# salt-key -A

The following keys are going to be accepted:

Unaccepted Keys:

saltminion

Proceed? [n/Y] Y

Key for minion saltminion accepted.

## Installing and Configuring nginx with SaltStack

While the above information gets you started with Salt, it doesn't explain how to use Salt to install a package. The below steps will outline how to install a package and deploy configuration files using Salt.

### Creating the nginx state

SaltStack has policies just like any other configuration automation tools, however in Salt they are referred to as "states". You can think of these as the desired states of the items being configured.

#### Creating the nginx state directory and file

Each state in salt needs a sub-directory in the respective environment. Because we are going to use this state to install and configure nginx I will name our state nginx and I am placing it within our base environment.

root@saltmaster:~# mkdir /salt/states/base/nginx

Once the directory is created we will need to create the "init.sls" file.

root@saltmaster:~# vi /salt/states/base/nginx/init.sls

#### Specifying the nginx state

Now that we have the Salt State file open, we can start adding the desired state configuration. The Salt State files by default utilize the YAML format. By using YAML these files are very easy to read and easier to write.

##### Managing the nginx package and service

The following configuration will install the nginx package and ensure the nginx service is running. As well as watch the package nginx and nginx.conf file for updates. If these two items are updated the service nginx will be automatically restarted the next time salt is run against the minions.

**Add the following to init.sls:**

nginx:

pkg:

- installed

service:

- running

- watch:

- pkg: nginx

- file: /etc/nginx/nginx.conf

The configuration is dead simple, but just for clarity I will comment each line to explain how this works.

nginx: ## This is the name of the package and service

pkg: ## Tells salt this is a package

- installed ## Tells salt to install this package

service: ## Tells salt this is also a service

- running ## Tells salt to ensure the service is running

- watch: ## Tells salt to watch the following items

- pkg: nginx ## If the package nginx gets updated, restart the service

- file: /etc/nginx/nginx.conf ## If the file nginx.conf gets updated, restart the service

With configuration this simple, a Jr. Sysadmin can install nginx on 100 nodes in less than 5 minutes.

##### Managing the nginx.conf file

Salt can do more than just install a package and make sure a service is running. Salt can also be used to deploy configuration files. Using our nginx example we will also configure salt to deploy our nginx.conf file for us.

The below configuration when added to the init.sls will tell salt to deploy a nginx.conf file to the minion using the /salt/states/base/nginx/nginx.conf file as a template.

**Append the following to the same init.sls:**

/etc/nginx/nginx.conf:

file:

- managed

- source: salt://nginx/nginx.conf

- user: root

- group: root

- mode: 644

Again the configuration is dead simple, but let us break this one down as well.

/etc/nginx/nginx.conf: ## Name of the file

file: ## Tells salt this is a file

- managed ## Tells salt to mange this file

- source: salt://nginx/nginx.conf ## Tells salt where it can find a local copy on the master

- user: root ## Tells salt to ensure the owner of the file is root

- group: root ## Tells salt to ensure the group of the file is root

- mode: 644 ## Tells salt to ensure the permissions of the file is 644

After appending the nginx.conf configuration into the Salt State file you can now save and quit the file.

Make sure before continuing that you place your nginx.conf file into /salt/states/base/nginx/ as if Salt cannot find the file than it will not deploy it. It is also worth noting that if the nginx.confon the minion differs from the nginx.conf on the salt-master than Salt will overwrite the file automatically on its next run. This means that the nginx.conf on the master is now your master copy.

## Creating the top.sls file

The top.sls file is the Salt State configuration file, this file will define what States should be in effect on specific minions. The top.sls file by convention is usually in the base environment.

To add our nginx state to our salt-minion we will perform the following steps.

### Create the top.sls file

root@saltmaster:~# vi /salt/states/base/top.sls

**Append the following:**

base:

'saltminion\*':

- nginx

The configuration, much like the Salt State files is very simple. Let's break down the configuration a bit more though.

base: ## Tells salt what environment the following lines are for

'saltminion\*': ## Tells salt to apply the following to any hosts matching a hostname of saltminion\*

- nginx ## Tells salt to apply the nginx state to these hosts

That's it, we are done configuring salt stack.

### Apply The Salt States

Unlike other configuration management tools, by default SaltStack does not automatically deploy the state configurations. Though this can be done, it is not the default.

To apply our nginx configuration run the following command

root@saltmaster:~# salt '\*' state.highstate

saltminion:

----------

State: - file

Name: /etc/nginx/nginx.conf

Function: managed

Result: True

Comment: File /etc/nginx/nginx.conf is in the correct state

Changes:

----------

State: - pkg

Name: nginx

Function: installed

Result: True

Comment: The following packages were installed/updated: nginx.

Changes: nginx-full: { new : 1.1.19-1ubuntu0.2

old :

}

httpd: { new : 1

old :

}

nginx-common: { new : 1.1.19-1ubuntu0.2

old :

}

nginx: { new : 1.1.19-1ubuntu0.2

old :

}

----------

State: - service

Name: nginx

Function: running

Result: True

Comment: Started Service nginx

Changes: nginx: True

That's it, nginx is installed & configured. While this might have seemed like a lot of work for installing nginx, if you expand your salt configuration to php, varnish, mysql client/server, nfs and plenty of other packages and services. At the end of the day SaltStack can save SysAdmin's valuable time.