SALT FUNDAMENTALS

INTRODUCTIONS

SALT TERMS

MASTER

A central Salt daemon from which commands can be issued to listening minions.

MINION

A server running a Salt minion daemon which can listen to commands from a master and perform the requested tasks. Generally, minions are servers which are to be controlled using Salt.

TARGETING

Specifying which minions should run a command or execute a state by matching against hostnames, or system information, or defined groups, or even combinations thereof.

MODULE

Functions called by the salt command that perform specific tasks.

There are many types of modules

- execution module
- state module

SALT-KEY

Executes simple management of Salt server public keys used for authentication between masters and minions.

GRAIN

A key-value pair which contains a fact about a system, such as its hostname, network addresses. This should always be static information.

STATE

A description of the disired configuration of salt minions, States comprise of single or multiple salt modules.

STATE MODULE

A module which contains a set of functions (things that need to be done).

SLS

Stands for Salt State, and is the file extension used for state files.

YAML

is a human-readable data format.

the primary language to define and configure salt.

WHY USE SALT?

- Automate
- Configure
- Deploy
- Provision

WHAT CAN SALT DO?

- Configure
- Deploy
- Build
- Execute and retrieve
- Monitor

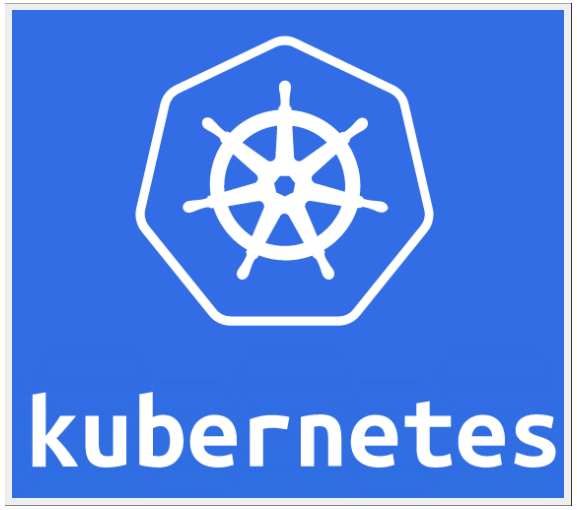

Chef

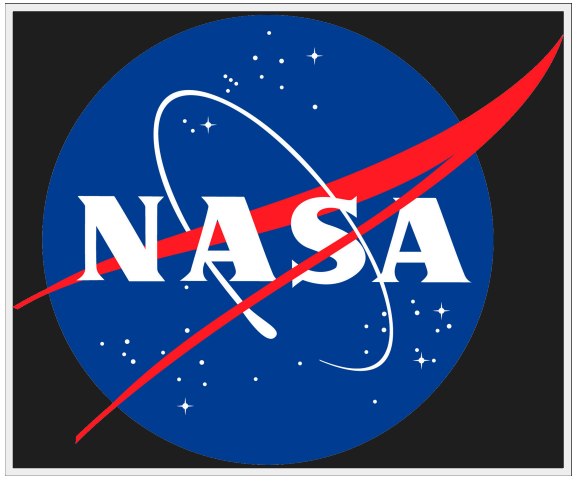

puppet
labs


ANSIBLE


SALTSTACK

WHO USES SALT NOW?



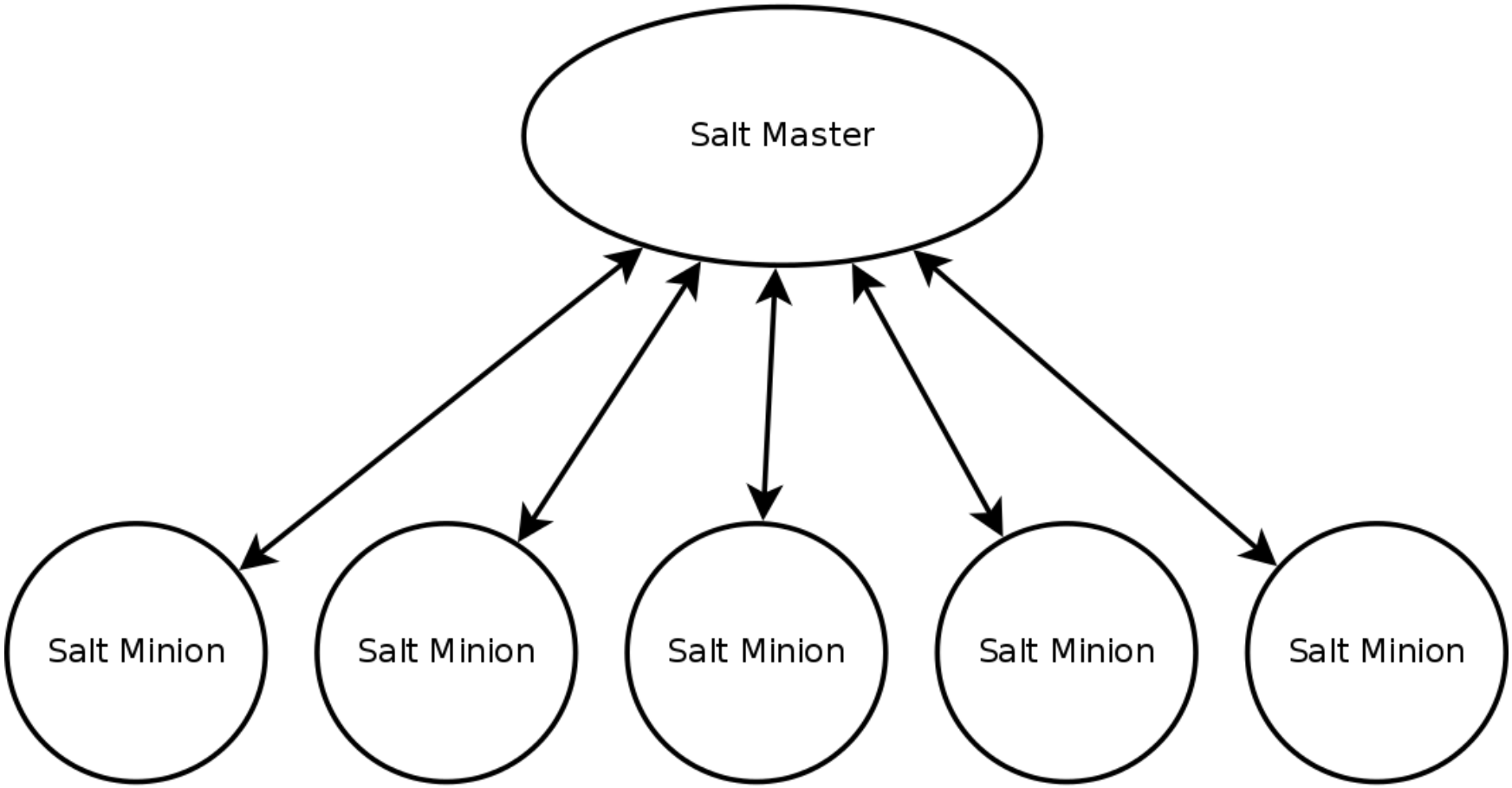


WHAT COULD YOU USE SALT FOR?

- In small groups discuss other ways salt could help improve a customers infrastructure.
- See what examples you can come up with where configuration management like salt would have saved time or resources.

SALT ARCHITECTURE

- Master and Minions
- SSH key management
- Salt Modules


Salt Master
Salt Minion
Salt Minion
Salt Minion
Salt Minion
Salt Minion

SALT MASTER

Controller of your Salt Minions

- /etc/salt/master
- /var/log/salt/master
- /etc/init.d/salt-master
- Communicate via ZeroMQ or SSH
- New RAET system replacing ZeroMQ

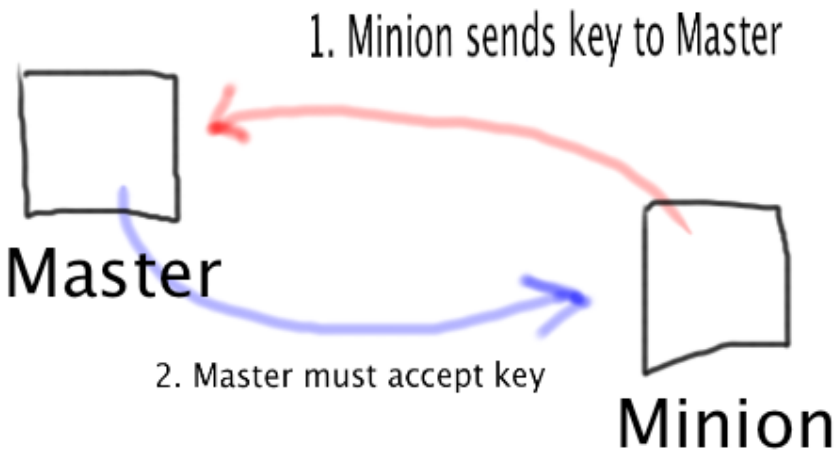
SALT MINION

Allows for a Salt Master to control a server

- /etc/salt/minion
- /var/log/salt/minion
- /etc/init.d/salt-minion

KEY MANAGEMENT

- RSA keys used for encryption and authentication
- Minions generate keys
- salt-key on Master
- /etc/salt/pki


1. Minion sends key to Master
Master
2. Master must accept key
Minion

SALT MODULES

- Execution Modules
- State Modules
- and the rest...

SALT V. CHEF V. ANSIBLE

NUTSHELL

- Chef (2009) has been an industry standard, Puppet lineage, (2) flavors: Open Source, and Enterprise (adds hosted/private server, push capable, better web UI, central authentication, multi-tenancy, RBAC, reporting and support)

- Ansible (2012) is very recent, fully open source, active community driven development. Released as an open source command line tool, and a licensed GUI.

- Salt (2011) is fully open source, supports Windows, and is built on remote execution. Highly modular and extensible system.

PHILOSOPHY

- Chef is about coding in ruby to suit your needs, and leverages a large pre-built base of re-usable code
- Ansible is about simple, extensible, batteries included, best ease of use, no coding
- Salt is about simple, extensible, ultra high speed, enterprise scale, batteries included, coding optional

LANGUAGES

- Chef: Ruby code using a specialize DSL (Domain Specific Language)
- Ansible: Python (but YAML for users)
- Saltstack: Python (but YAML for users)

ADDUSER IN CHEF

```
user "fred" do
  supports :manage_home => true
  comment "Fred Flintstone"
  uid 4000
  home "/home/fred"
  shell "/bin/zsh"
end
group "wheel" do
  action :modify
  members "fred"
  append true
end
group "storage" do
  action :modify
  members "fred"
  append true
end
```

ADDUSER IN SALT

```
fred:
  user.present:
    - fullname: Fred Flintstone
    - shell: /bin/zsh
    - home: /home/fred
    - uid: 4000
    - gid: 4000
    - groups:
      - wheel
      - storage
```

ARCHITECTURE

- Chef: Server + Client agent, or Solo (no network component)
- Ansible: Agentless (uses ssh)
- Salt: Master + Minion (client), or Standalone

MANAGEMENT

- Chef: Code based pull, but added ZeroMQ push feature recently
- Ansible: YAML based push using ssh or ZeroMQ
- Salt: YAML based push/pull

COMMUNICATIONS

- Chef: standard key-based SSL
- Ansible: standard SSH (or paramiko), MQ option
- Salt: ZeroMQ, SSH, or the new RAET

TEMPLATING

- Chef: Ruby ERB
- Ansible: Jinja2
- Salt: Jinja2, others

ADVANTAGES OF SALT

- Agents add (targeting) flexibility
- HA options available
- Highly modular
- Friendly community driven development
- API
- Vendor agnostic infrastructure management

INSTALL AND DEPLOY SALT

- salt-bootstrap
- Download salt-bootstrap
- Run salt-bootstrap
- Salt Keys
- Verify installation
- Lab

SALT-BOOTSTRAP

salt-bootstrap is the easiest method to install SaltStack.

- Multiple OS support
- Highly extendable
- Can install Salt Master / Minion

DOWNLOAD SALT-BOOTSTRAP

```
wget -O install_salt.sh https://bootstrap.saltstack.com
```

RUN SALT-BOOTSTRAP

```
sh install_salt.sh
```

This example installs the minion

Use -h to show the help

EXAMPLE FLAGS

```
sh install_salt.sh -M -A localhost
```

This will install the Master and Minion (and configure it) all on the same machine.

SALT KEYS

Salt Keys manages which machines are allowed / not allowed to commicate with the Salt Master

```
salt-key
```

The -A flag will accept all pending keys

VERIFY INSTALLATION

```
salt-master --versions-report
```

If installed correctly, you will see something like:

```
Salt: 2014.1.0
Python: 2.7.6 (default, Mar 22 2014, 22:59:56)
Jinja2: 2.7.2
M2Crypto: 0.21.1
msgpack-python: 0.3.0
msgpack-pure: Not Installed
pycrypto: 2.6.1
PyYAML: 3.10
PyZMQ: 14.0.1
ZMQ: 4.0.4
```

LAB

- Step One: Download the salt-bootstrap script
- Step Two: Read the help file on salt-bootstrap
- Step Three: Install the Salt Master / Minion on the same box
- Step Four: Verify that the Salt Master is installed
- Step five: Accept the minion's keys

EXAMPLE COMMANDS AND STATE FILES

- State and Execution Modules
- test.ping
- command
- pkg and service
- Lab

EXECUTION MODULES

Execution modules are used when calling modules directly from the command line.

```
salt '*' user.add fred uid gid groups home shell
```

TEST.PING

This is the most basic module you can run. It simply connects to the targeted minion(s) and returns True if that minion responds.

```
salt '*' test.ping
minion1:
    True
minion2:
    True
```

There is no matching state module for test.ping.

TEST.PING DEMO

RUNNING COMMANDS

Running arbitrary commands can come in handy for one time operations. This is done with the cmd.run module.

Execution Example

```
salt webservers cmd.run "service apache2 restart"
```

State Example

```
touches_foo:
  cmd.run:
    - name: touch /tmp/foo
```

DATE COMMAND DEMO

YOUR TURN!

- on your salt master run the test.ping module targeting all minions

YAML EXAMPLES

SaltStack has some good examples documented here:

```
http://docs.saltstack.com/en/latest/topics/yaml/index.html
```

STATE MODULES

State modules are used when calling state.sls or highstate against a minion.

```
salt '*' state.sls myslsfile
```

```
fred:
  user.present:
    - fullname: Fred Jones
    - shell: /bin/zsh
    - home: /home/fred
    - uid: 4000
    - gid: 4000
    - groups:
      - wheel
      - storage
      - games
```

INSTALLING A PACKAGE

Package installations are abstracted by Salt. You don't have to invoke a specific yum or apt module, but a pkg module.

Execution Example

```
salt '*' pkg.install vim
```

State Example

```
install_vim:
  pkg.installed:
    - name: vim
```

Package names can still be different between distributions. (ie. apache2 and httpd)

INSTALLING SYSSTAT DEMO

STARTING A SERVICE

Starting services is done with the service module.

Execution Example

```
salt '*' service.start apache2
```

State Example

```
run_apache2:
  service:
    - running
    - name: apache2
    - enable: True
```

Service names can still be different between distributions. (ie. apache2 and httpd)

RESTART SSH DEMO

YOUR TURN!

- Write a state.sls file that will install the cowsay package
- execute that state file from the command line using the state.sls module

HINT

```
vim_install:
  pkg.installed:
    - name: vim
```

GRAINS

A grain is a piece of data related directly to a Salt Minion. You can view all grains related to a node with the grains.items module.

```
salt minion1 grains.items
```

You can target minions using data in grains. This example will target all your CentOS servers with a test.ping.

```
salt -G 'os:CentOS' test.ping
```

GRAINS DEMO

WRITE YOUR OWN STATE FILE TO...

- install the nginx webserver package
- start the ngnix service
- add the user thatch
- attempt to target using grains