Salt and Ceph

Automating ceph

Owen Synge

Owen.synge@suse.com

Why am I talking about automation?

- Distributed computing without automation:
  - Is monotonous!
    - Setting up nodes is boring.
  - Is Unreliable!
    - Human errors creep in.
  - Does not scale!
    - Upgrading 20 nodes takes all day?
    - Installing 100 disks is tedious.
  - Has no recovery strategy!
    - Redeploying a server from bare metal often cures issues!

Overview

- Comparing Configuration Management Systems.
  - Mostly from salt perspective.
- What can already be done with Salt for Ceph.
  - With work we have already done.
- What we will be doing with salt next.
  - Moving from configuration to management.

- Comparing configuration management systems.

Salt, Puppet, Chef, Ansible

- Configuration management tools are now common.
  - Found DESY HEPIX talk replacing one over 30 years ago.
- CERN does not write their own anymore.
- CMS mostly do the same thing.
  - Manage state transitions on many computers.
  - Take booted bare OS to a production service
    - Non-interactively.

CMS: Usual structure to user

- Made up of a library of reusable modules.
- Have a DSL to call the libraries
  - Express dependency
  - Include other DSL files.
  - Express branching.
- Have meta-data about nodes.
  - Can query this meta-data in the DSL.

Puppet Comparing to Salt.

- Puppet has biggest deployment base.
  - CERN our hosts use Puppet.
- Polls master server for config to apply.
  - Minimized dependency on master service.
  - Salt was first a remote execution service.
    - Similar to mcollective.
      - Puppet added mcollective much later.
  - Salt added state management later.
- Puppet is ruby based while Salt is python based.

Chef comparing to Salt

- Chef has the biggest deployment base in Germany.
  - Quiet mature but I find docs confusing.
  - Newer than puppet.
- Chef relies on polling.
  - Salt allows you to push configuration to client.
- Chef uses json for config
  - Salt uses yaml.
- Chef is ruby based / Salt is python based.

I don't know chef as well as I know puppet and salt

Ansible comparing to Salt

- Ansible uses ssh rather than agents.
  - Pushes commands to clients.
  - Low startup costs.
  - Fast growing community (Red hat now owns Ansible).
- Python based just like salt.
- Newer than puppet and chef
- Great test suite.

I don't know ansible as well as I know puppet and salt

Salt compared to other CMS.

- Youngest major player.
- Steep learning curve.
  - Documentation is improving, but many components
- More ambitious in making an event based site.
  - More moving parts (beacons, mines, pillars, reactors)
- Based on Event bus.
  - Events sent between

Salt : Programming your data center

- Basic usage similar to Puppet / Chef / Ansible
  - Thin DSL in YAML calling modules.
- Advanced usage:
  - Database integration
    - Pillar (as a data source) Mine (For read write)
  - Monitoring events.
    - Beacons (can dynamically be started on minions)
  - Event chaining.
    - Reactors, Orchestration engine.

Salt overview

- Message Queue at its core (zmq).
  - Master/Slave (Minion) model.
- Agent based, Event based.
- Think of it as a framework for distributed computing.
  - Extendable modules (master and minion).
  - Database modules (master and minion).
    - Backend can be simple jaml to full RDBMS (called pillars or mines)
  - Extendable attributes (called grains).
  - Events can be fired by any module.

- What can already be done with Salt for Ceph.

What we in salt-ceph trying to do?

- Make ceph quick and easy to setup.
  - You should not require the skills of a CERN admin.
- Do everything we can in parallel.
  - You should not have to wait (more than 1 min).
- Easy to support.
  - Clear errors, logs, debugging, dependency management.
- Automate Ceph management.
  - Now this is hard! (See later in the talk)
- Multiple user interfaces
  - Version controlled config files and GUI operation.

Salt execution modules

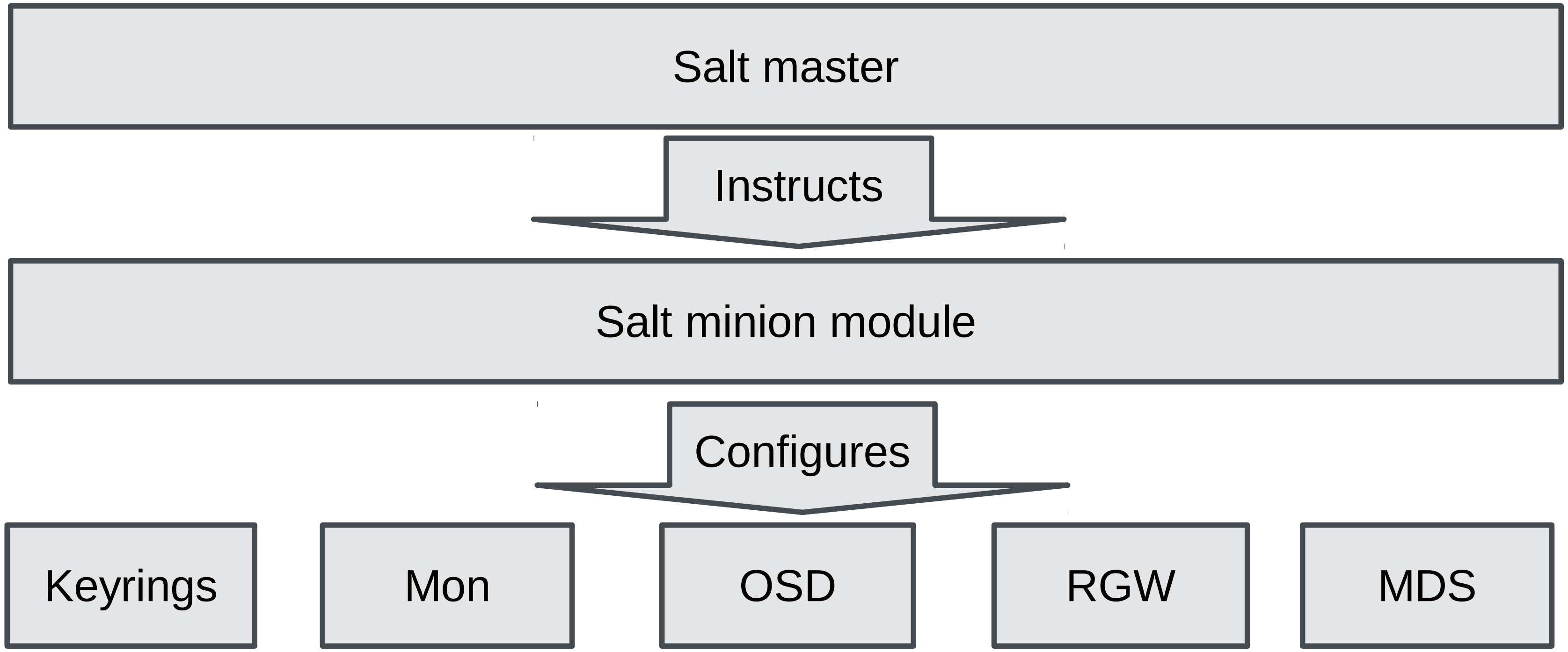
- Provide components of configuration.
  - Present a name to the DSL
  - Contain methods to be called by DSL.
  - No restrictions on return structure
- Example calling from salt DSL the ceph module:

```
prepare_vdb:
  module.run:
    - name: ceph.osd_prepare
    - kwargs: {
        osd_dev: /dev/vdb
        }
    - require:
      - module: keyring_osd_auth_add
      - pkg: ceph_packages_osd
```

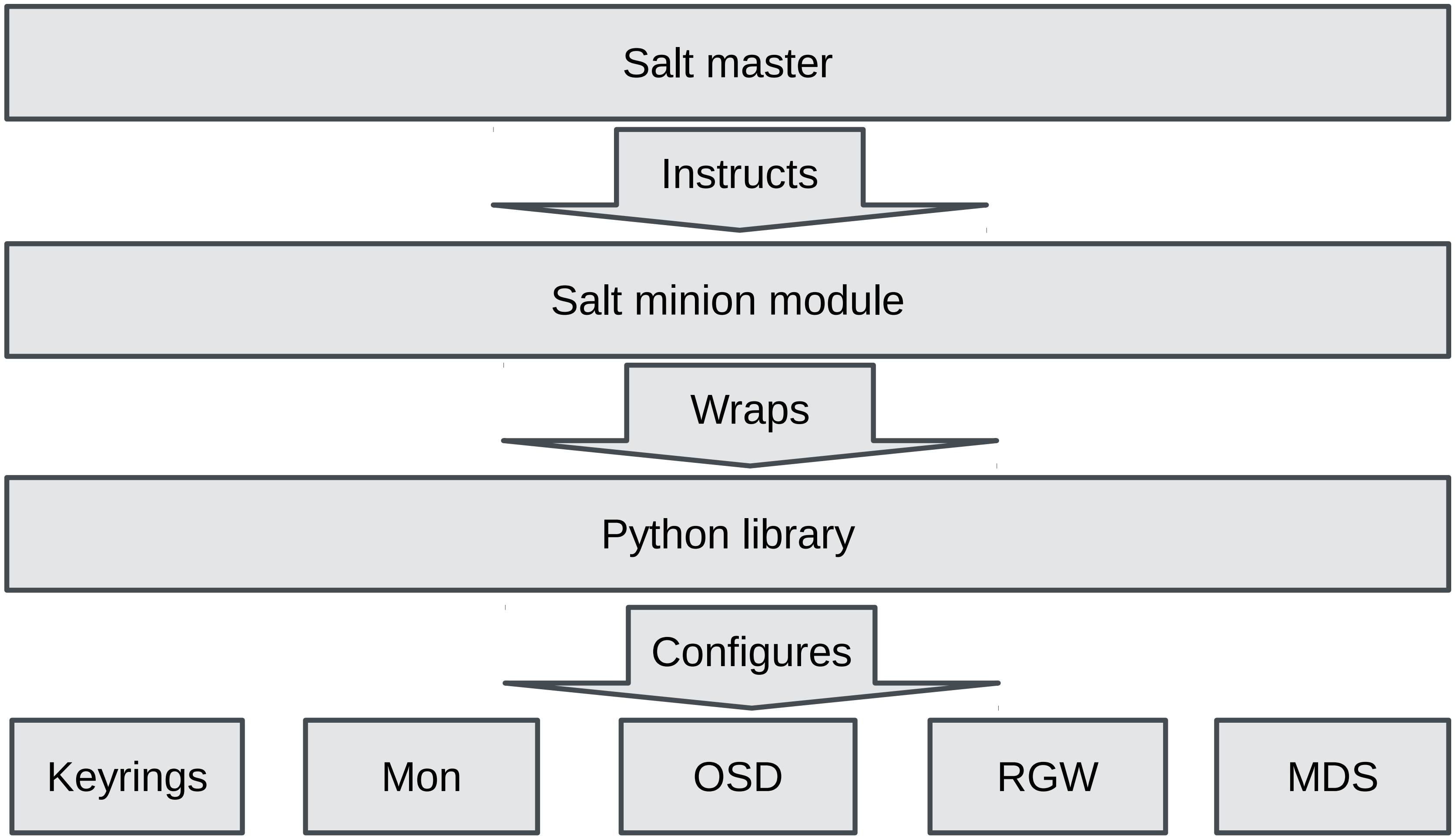
Background : Ceph Components

- Ceph has a nice dependency hierarchy
  - Keyrings (have a hierarchy of dependencies)
  - MON service (depend on keys)
  - OSD service (depend on mon + keys)
  - RGW service (depend on osd + mon + keys)
  - MDS Service (depend on osd + mon + keys)
  - RBD Service (depend on osd + mon + keys)
  - iSCSI Service (depend on rbd + osd + mon + keys)

Basic salt module implementation.



Reusable Ceph module implementation.



•Python-ceph-cfg

- Pure python library to configure ceph
  - No salt dependencies
  - Installs and runs like any other python library.
  - 54 documented public methods
- Manages ceph entities and lifecycles.
  - Keyrings, mon, osd, rgw, mds

“The Analyst” and I might make a YAIM inspired CLI to reuse this code.

Python-ceph-cfg : Code structure

- Mostly “Model View Controller” design pattern.
  - Code reuse very high
- Alternatives usually presented with “Facade pattern”
  - init system (systemd / sysVinit)
  - Bootstrap Keyrings (mon, admin, osd, mds, rgw)
- Started test suite based:
  - py.test, flake8, tox

Python-ceph-cfg : externals

- Runtime dependencies
  - ceph.conf
    - Currently needed to establish if node is mon node
    - Used to default cluster UUID and cluster name if not set.
  - Environment aware:
    - If run inside Salt uses salt to launch processes.
    - If run naively uses python subprocess library.
- Publicly available and open source
  - https://github.com/oms4suse/python-ceph-cfg

Salt-ceph execution module overview.

- Salt-ceph
  - https://github.com/oms4suse/sesceph
  - Execution module for salt (Nearly complete)
  - State module for salt (Basic so far)
  - Simple example to build up and tear down ceph clsuters.

Salt-ceph execution module.

- Execution module with 54 methods
  - Simply wraps python-ceph-cfg
  - This is the low level salt library to configure ceph
  - Most common functionality of ceph-deploy.
  - All methods are idempotent.
  - Approximately 50/50, discovery/configuration.

Salt state modules : going salt native

- Syntactic sugar on top of execution modules
  - Fixed return structure:
    - Require tasks to be report operations requested.
      - So you now which if any operations where performed.
    - Require Success / Failure
      - So DSL can trigger on success or failure
    - Calling context.
  - Require a test parameter
    - No-op, only show steps to test operation.
      - Will report what would have been done as separate steps.

Salt-ceph state module

- Only has one method currently in released branch.
  - Quorum method.
    - Checks if the cluster is in quorum either locally or cluster wide.
  - Have proto typed a lot of processes.
    - See later in talk.

What can we do with what we have?

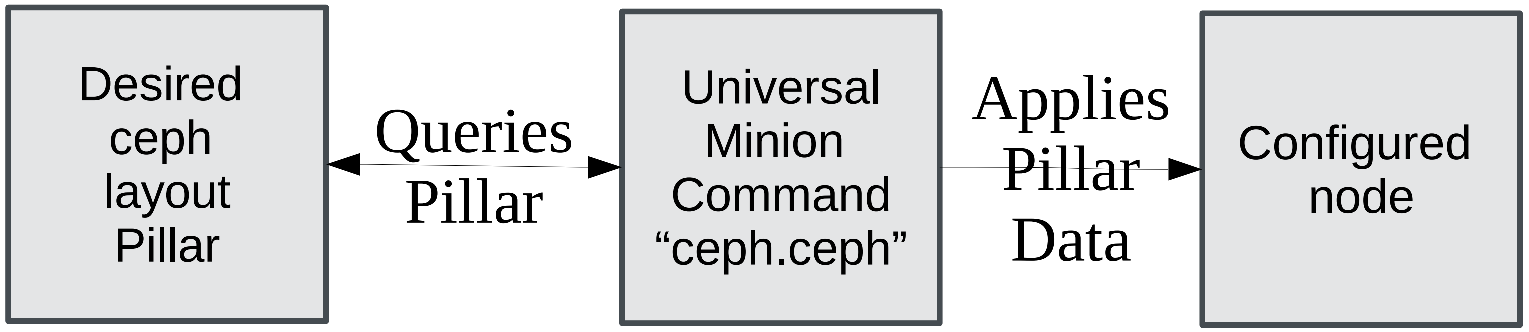
- Install ceph with a single text file.
  - Assign roles to nodes
  - Configure nodes based on roles.
- Here is an demo cluster_buildup.sls file.
  - Sets up keyrings mon, osd, mds, rgw per node.
- Example role based setup
  - roles for mon, osd, mds, rgw.
- Do we do a demo, or do we talk plans?

- What is planned?

Config: Moving to a data driven model.

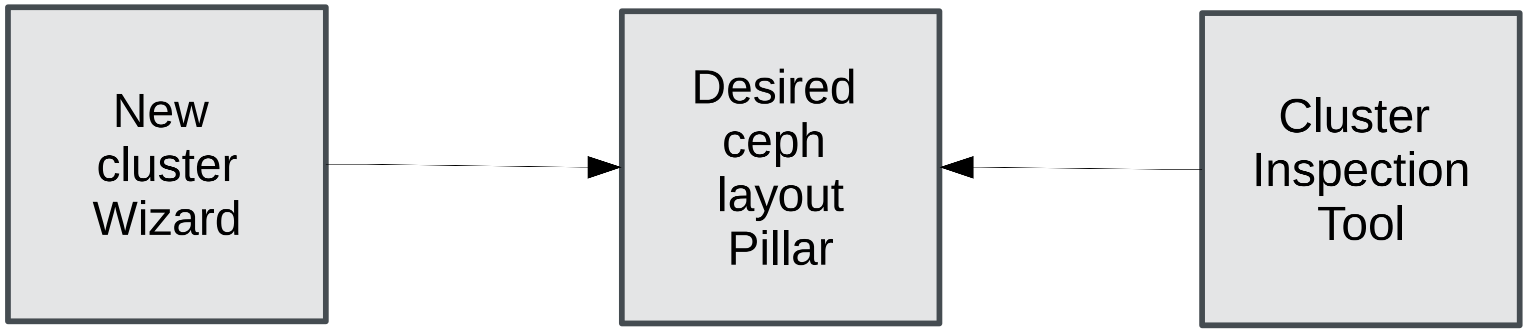
- All data in a central pillar.
  - Just apply “ceph.ceph” to every node.
  - All decision inside a single state module method.

```
cluster_ceph:
    ceph.ceph:
    - clusterID: b7c3ccd2-701c-4857-b347-240853038da4
```


Desired ceph layout Pillar
Queries Pillar
Universal Minion Command “ceph.ceph”
Applies Pillar Data
Configured node

Pillar population tool

- Wizard to setup your ceph cluster
- Import tool to query an existing ceph cluster.



Maintenance: Why Beacons?

- Unsafe removal of OSD
  - Must be restricted by failure domains (To avoid data loss).
- Nice removal of an OSD:
  - OSD weight set to Zero (Fast).
  - Ceph empties data from OSD (Very slow).
  - Take OSD down (Fast).
  - Remove OSD from cluster (Fast).
  - Remove OSD Keys from authorized keys list (Fast).

Oh dear we now have a task that might take hours to complete!

What are salt beacons

- Beacons monitor things
  - Like disk usage
- Beacons fire events based on conditionals
  - Such as OSD is empty.
- Reactors receive events and trigger actions.

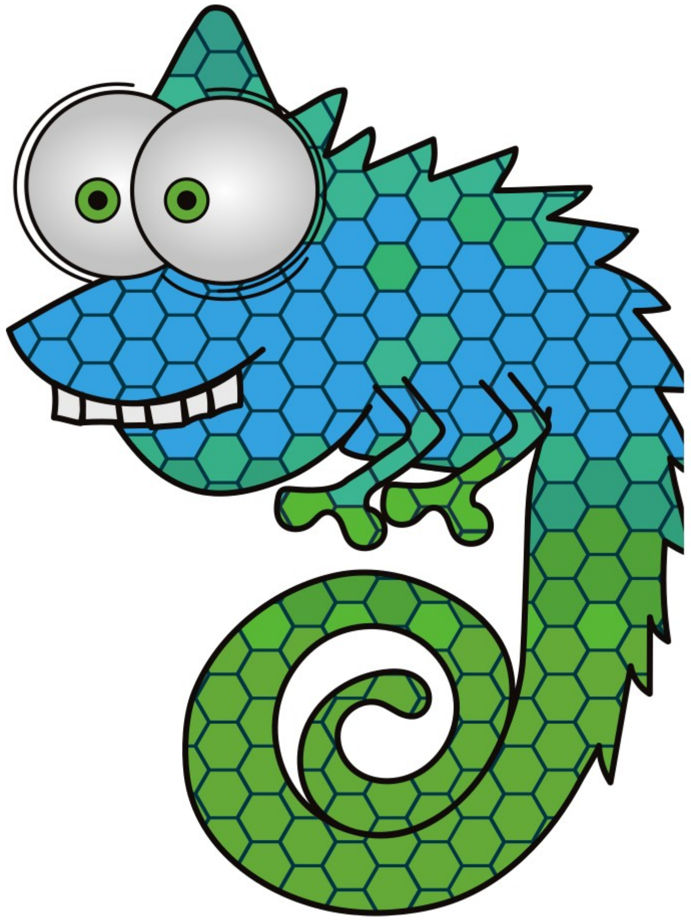
Maintenance : salt based OSD removal.

- We want to automate OSD removal.
  - In 3 stages:
    - Trigger ceph to drain OSD.
      - Set desired state in pillar, and trigger beacon on minion.
    - Notice OSD has drained and Trigger update
      - Beacon checks locally, and then fires events.
    - Notice expected OSD still exists and is drained
      - Can now remove OSD from cluster.

Questions?

Join the conversation,
contribute & have a lot of fun!
www.opensuse.org

Thank you.



Have a Lot of Fun, and Join Us At:

www.opensuse.org

Credits

Template
Richard Brown
rbrown@opensuse.org

Design & Inspiration
openSUSE Design Team
http://opensuse.github.io/branding-guidelines/