Config Management: Ansible

@NoahJS

Noah Spirakus

[…] Ansible has the lowest learning curve of all the CM tools, so if you found this chapter at all challenging, you should use Ansible and not even consider the other tools.

- Matt Jaynes
  [Taste Test]

Infrastructure as Code

We want to move from:

```
$ ssh web1
  $ yum update
  $ yum install httpd
  $ rpm -Uvh http://mirror.webtatic.com/yum/el6/latest.rpm
  $ yum install php55w, php55w-common, php55w-devel, php55w-cli, php55w-gd, php55w-mcrypt, php55w-mbstring,
                        php55w-mysql, php55w-pecl-memcache, php55w-opcache, php55w-process, php55w-xml
  $ curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin
  $ mv /usr/local/bin/composer.phar /usr/local/bin/composer
  $ vim /etc/conf.d/httpd.conf
    # Make sure configs are correct
  $ vim /etc/php
    # Change logging, configs, error reporting, timezone etc…
  $ service httpd restart
  $ chkconfig httpd

$ ssh web2
  $ yum update
  $ yum install httpd
  $ rpm -Uvh http://mirror.webtatic.com/yum/el6/latest.rpm
    •••
```

To:

```
$ bash “Make me a webserver”
```

To: (Even better)

```
$ bash “Make me a webserver like last time”
```

Solution

Configuration Management

*

*Maybe slightly over exaggerated

The Options

SALTSTACK

puppet labs®

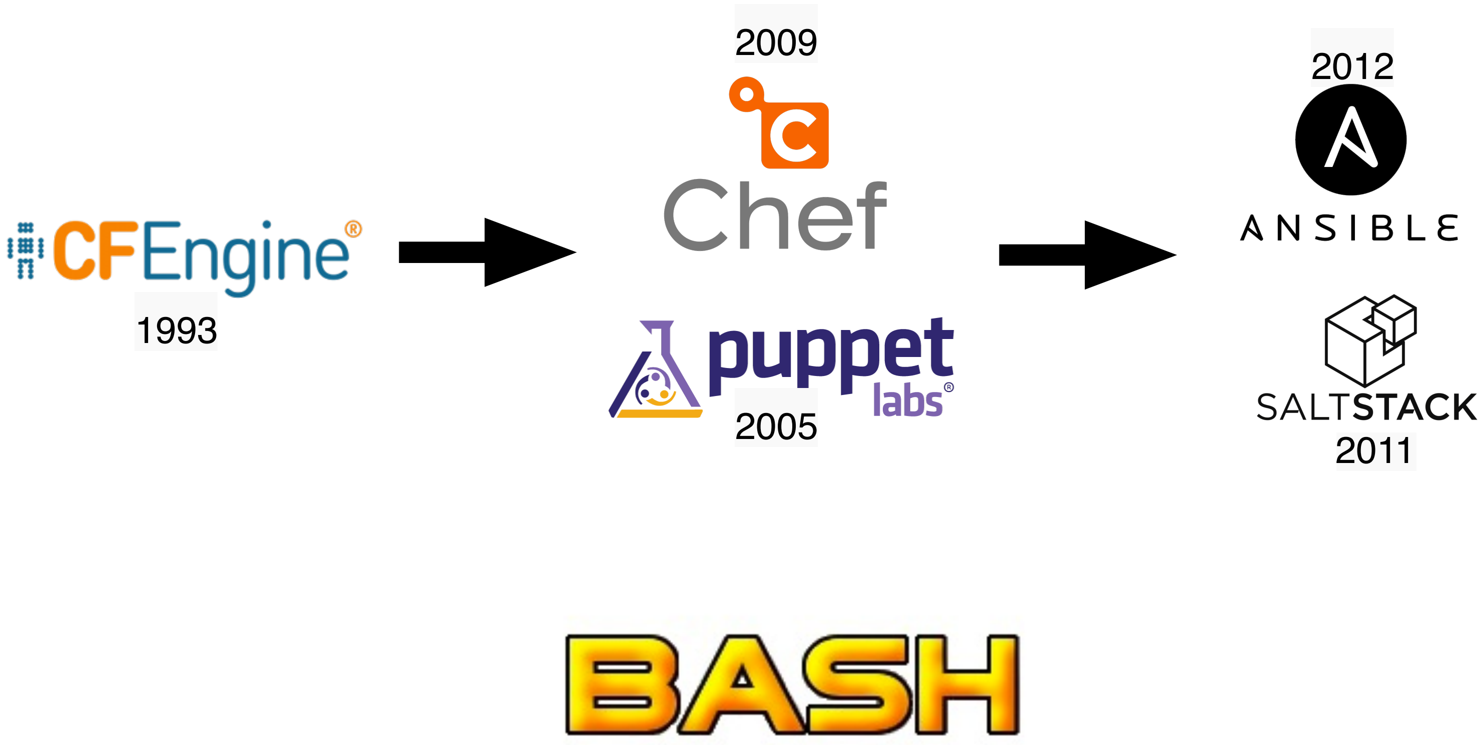
ANSIBLE

CFEngine®

BASH

Chef

Timeline


CFEngine
1993
2009
Chef
puppet labs
2005
2012
ANSIBLE
SALTSTACK
2011
BASH

Comparison

Configs

Chef

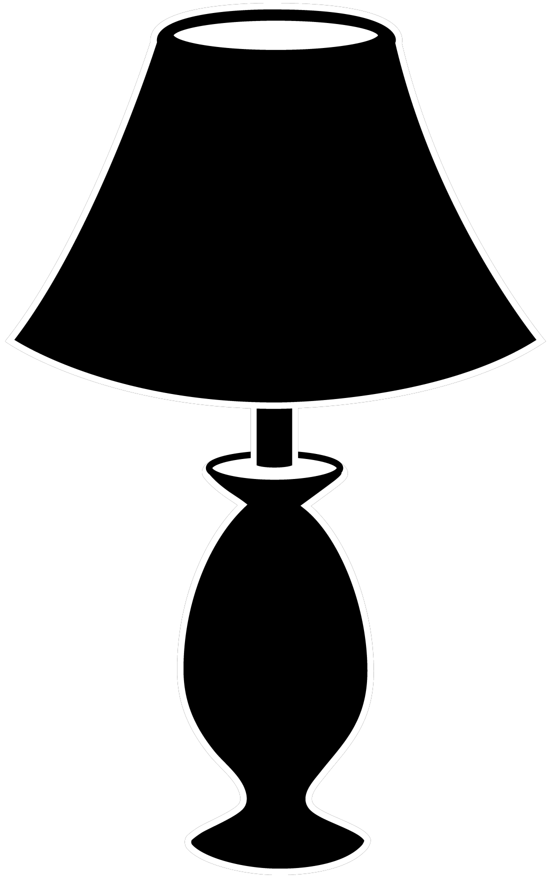
puppet labs®

Configs + Execution

SALTSTACK

ANSIBLE

Lets build a LAMP

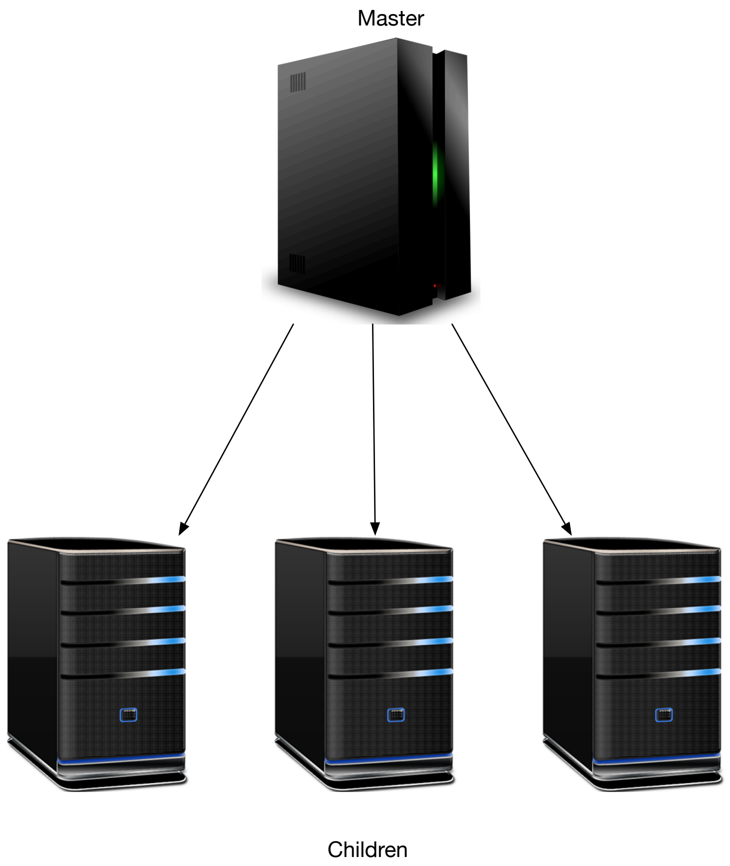


LAMP: Bash

```
$ ssh web1
  $ yum -y update
  $ yum -y install httpd
  $ rpm -Uvh http://mirror.webtatic.com/yum/el6/latest.rpm
  $ yum install php55w, php55w-common, php55w-devel, php55w-cli, php55w-gd, php55w-mcrypt, php55w-mbstring,
                    php55w-mysql, php55w-pecl-memcache, php55w-opcache, php55w-process, php55w-xml
  $ curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin
  $ mv /usr/local/bin/composer.phar /usr/local/bin/composer
  $ vim /etc/conf.d/httpd.conf
    # Make sure configs are correct
  $ vim /etc/php
    # Change logging, configs, error reporting, timezone etc…
  $ service httpd restart
  $ chkconfig httpd
```

LAMP: Puppet

LAMP: Puppet (Layout)



LAMP: Puppet (Terminology)

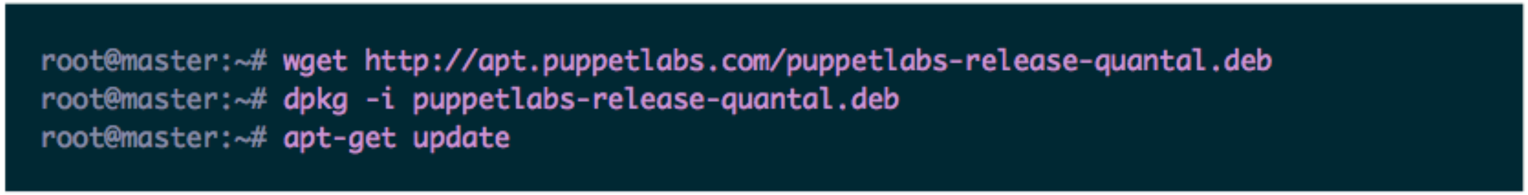
Directives = Resources

Directives Script = Manifest

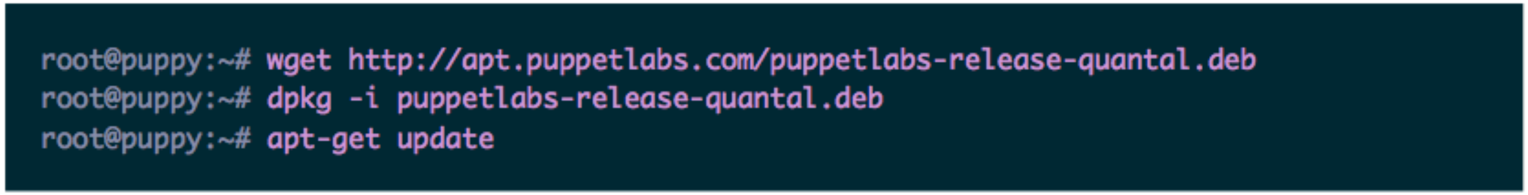
Facter is the utility Puppet uses for detecting node metadata.

LAMP: Puppet

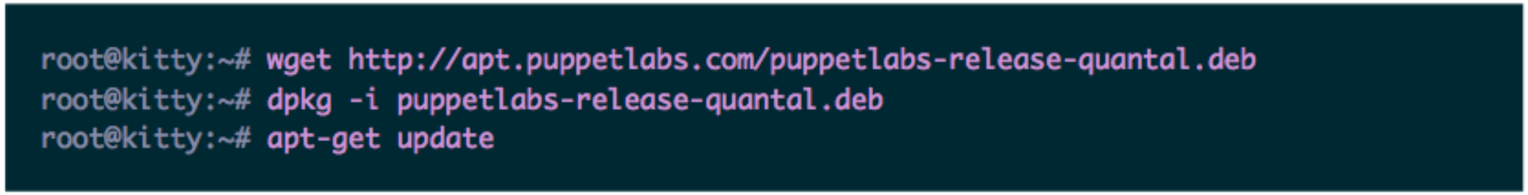
1. Install Puppet on all nodes



```
root@master:~# wget http://apt.puppetlabs.com/puppetlabs-release-quantal.deb
root@master:~# dpkg -i puppetlabs-release-quantal.deb
root@master:~# apt-get update
```



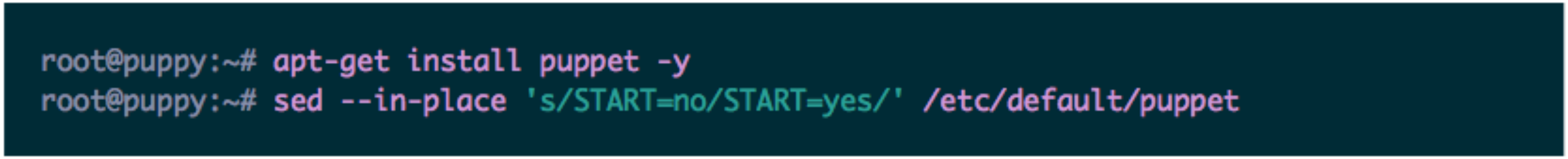
```
root@puppy:~# wget http://apt.puppetlabs.com/puppetlabs-release-quantal.deb
root@puppy:~# dpkg -i puppetlabs-release-quantal.deb
root@puppy:~# apt-get update
```



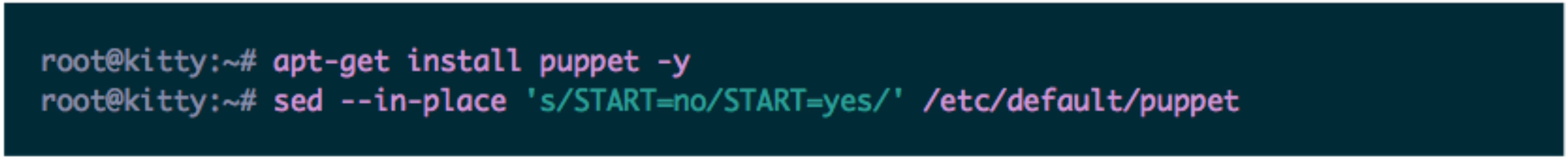
```
root@kitty:~# wget http://apt.puppetlabs.com/puppetlabs-release-quantal.deb
root@kitty:~# dpkg -i puppetlabs-release-quantal.deb
root@kitty:~# apt-get update
```

LAMP: Puppet

2. Setup Puppet clients on the children nodes to automatically start on reboots:



```
root@puppy:~# apt-get install puppet -y
root@puppy:~# sed --in-place 's/START=no/START=yes/' /etc/default/puppet
```

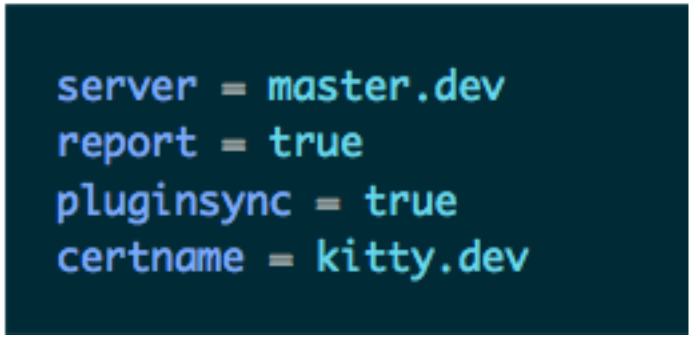


```
root@kitty:~# apt-get install puppet -y
root@kitty:~# sed --in-place 's/START=no/START=yes/' /etc/default/puppet
```

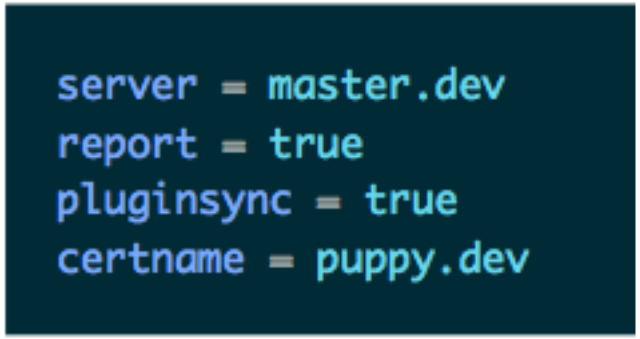
LAMP: Puppet

3. Add settings to the children nodes in the [main] section of

/etc/puppet/puppet.conf



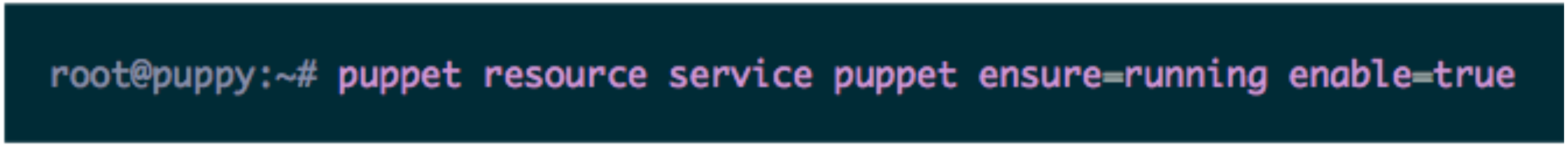
```
server = master.dev
report = true
pluginsync = true
certname = kitty.dev
```



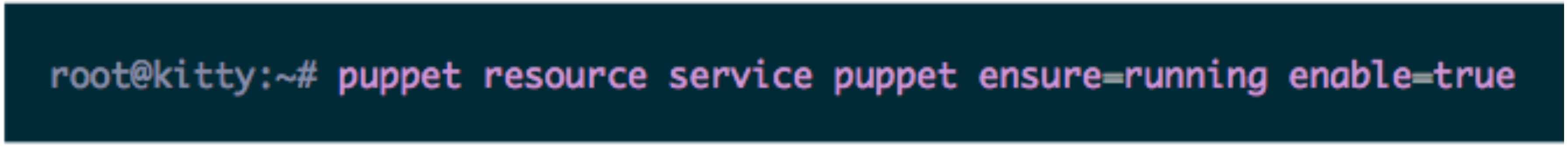
```
server = master.dev
report = true
pluginsync = true
certname = puppy.dev
```

LAMP: Puppet

4. Start & Enable Puppet clients



```
root@puppy:~# puppet resource service puppet ensure=running enable=true
```



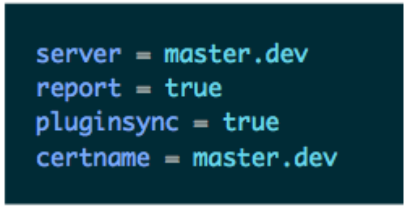
```
root@kitty:~# puppet resource service puppet ensure=running enable=true
```

LAMP: Puppet

5. Setup Master node

- root@master:~# apt-get install puppetmaster -y

- Add to /etc/puppet/puppet.conf



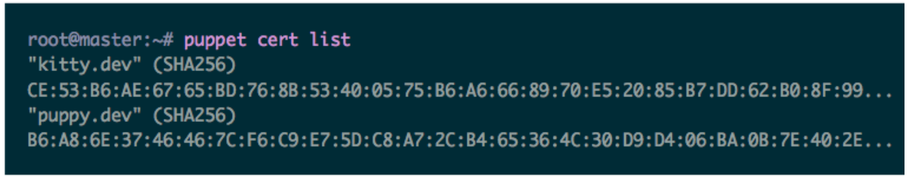
```
server = master.dev
report = true
pluginsync = true
certname = master.dev
```

- root@master:~# puppet resource service puppetmaster ensure=running enable=true

LAMP: Puppet

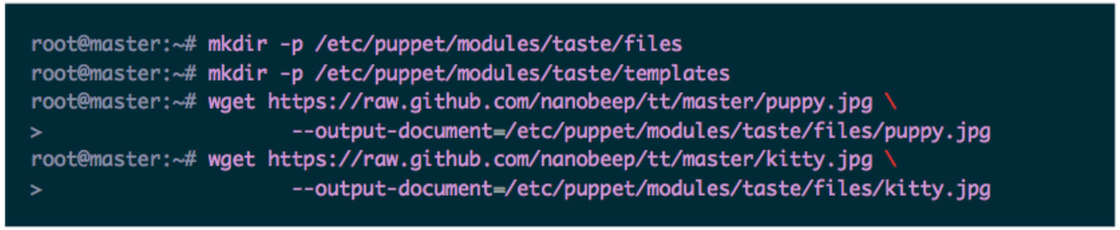
5. Setup Master node (cont)

- Sign Client certificates



```
root@master:~# puppet cert list
"kitty.dev" (SHA256)
CE:53:B6:AE:67:65:BD:76:8B:53:40:05:75:B6:A6:66:89:70:E5:20:85:B7:DD:62:B0:8F:99...
"puppy.dev" (SHA256)
B6:A8:6E:37:46:46:7C:F6:C9:E7:5D:C8:A7:2C:B4:65:36:4C:30:D9:D4:06:BA:0B:7E:40:2E...
```

- root@master:~# puppet cert sign –all
- Add supporting files



```
root@master:~# mkdir -p /etc/puppet/modules/taste/files
root@master:~# mkdir -p /etc/puppet/modules/taste/templates
root@master:~# wget https://raw.github.com/nanobeep/tt/master/puppy.jpg \
>                    --output-document=/etc/puppet/modules/taste/files/puppy.jpg
root@master:~# wget https://raw.github.com/nanobeep/tt/master/kitty.jpg \
>                    --output-document=/etc/puppet/modules/taste/files/kitty.jpg
```

LAMP: Puppet

Puppet Files:

/etc/puppet/modules/taste/templates/index.erb

/etc/puppet/manifests/site.pp

REF:
https://github.com/noahjs/azphp-configmanagement-presentation/example-code/puppet

LAMP: Puppet (Notes)

Directive Language: Custom

Execution Order:

Custom internal ordering for directives

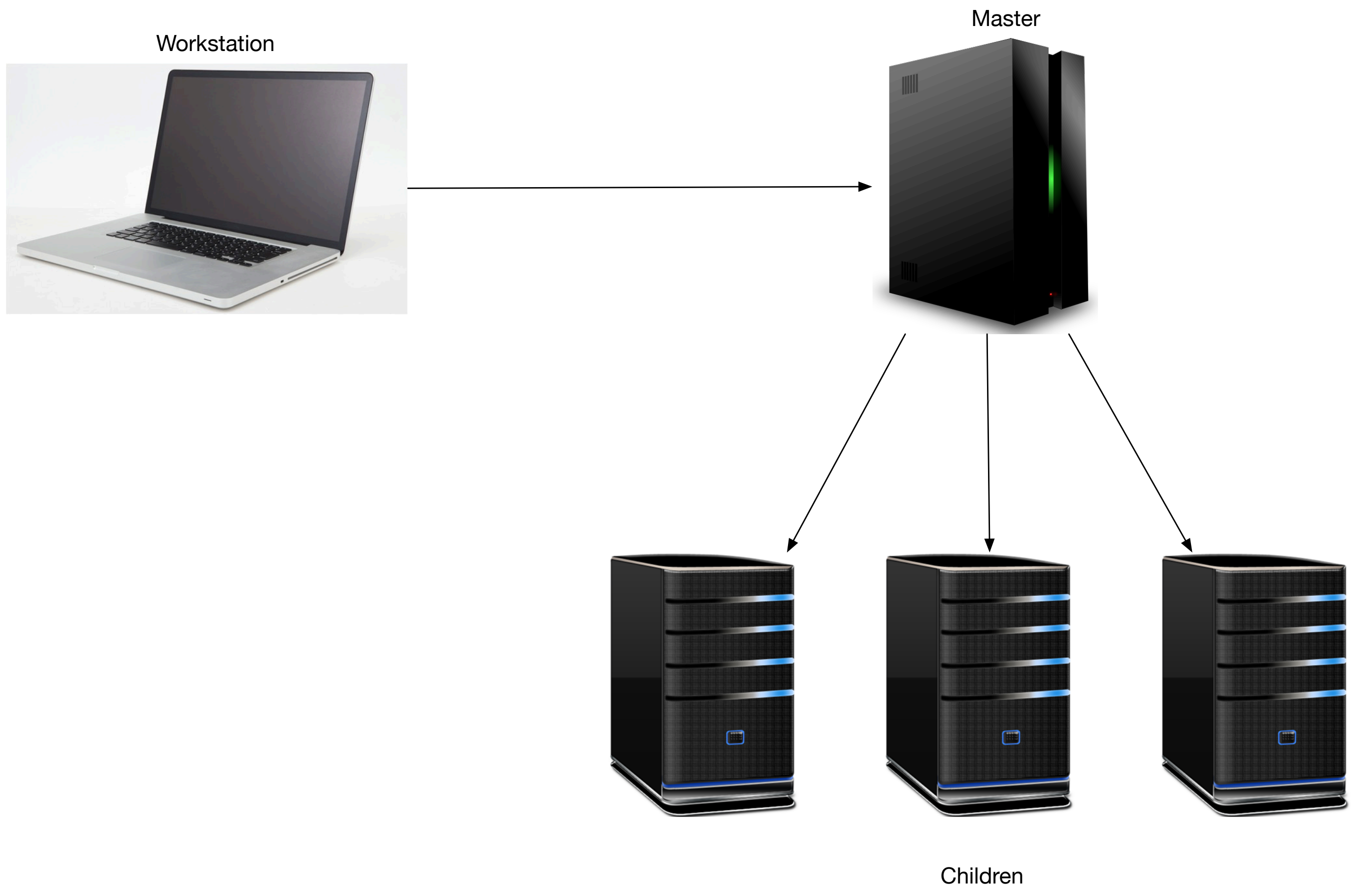
Remote execution: N/A

Recommends: mcollective

LAMP: Chef

Chef

LAMP: Chef (Layout)



LAMP: Chef (Terminology)

Directives = Resources

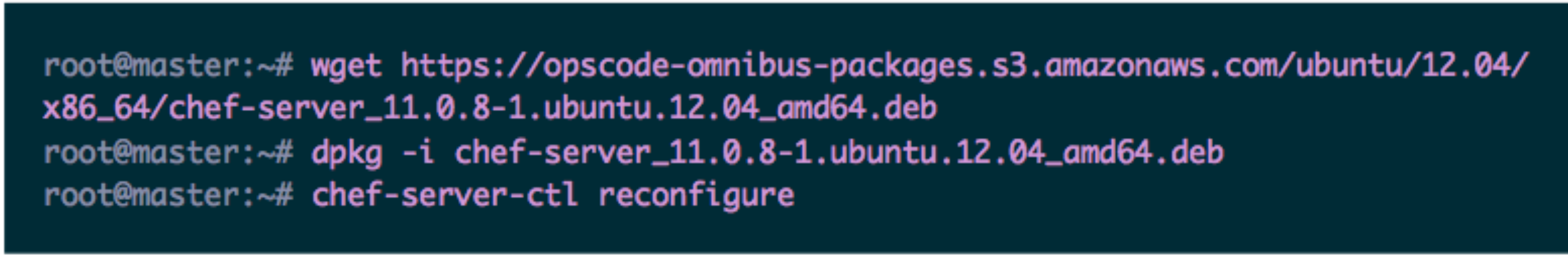
Directives Script = Recipe

Group of recipes and supporting files = Cookbook

Ohai is the utility Chef uses for detecting node metadata.

LAMP: Chef

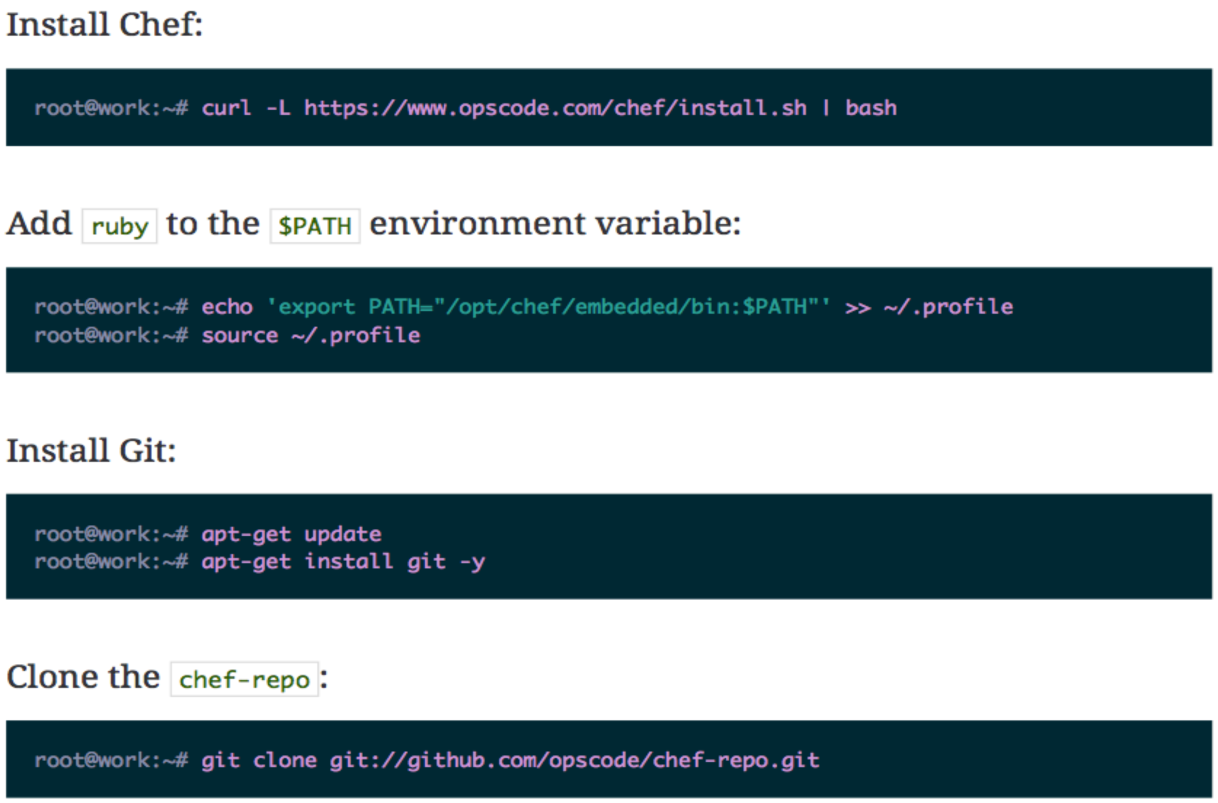
1. Install Chef on Master



```
root@master:~# wget https://opscode-omnibus-packages.s3.amazonaws.com/ubuntu/12.04/
x86_64/chef-server_11.0.8-1.ubuntu.12.04_amd64.deb
root@master:~# dpkg -i chef-server_11.0.8-1.ubuntu.12.04_amd64.deb
root@master:~# chef-server-ctl reconfigure
```

LAMP: Chef

2. Install Chef on Workstation



Install Chef:

```
root@work:~# curl -L https://www.opscode.com/chef/install.sh | bash
```

Add `ruby` to the `$PATH` environment variable:

```
root@work:~# echo 'export PATH="/opt/chef/embedded/bin:$PATH"' >> ~/.profile
root@work:~# source ~/.profile
```

Install Git:

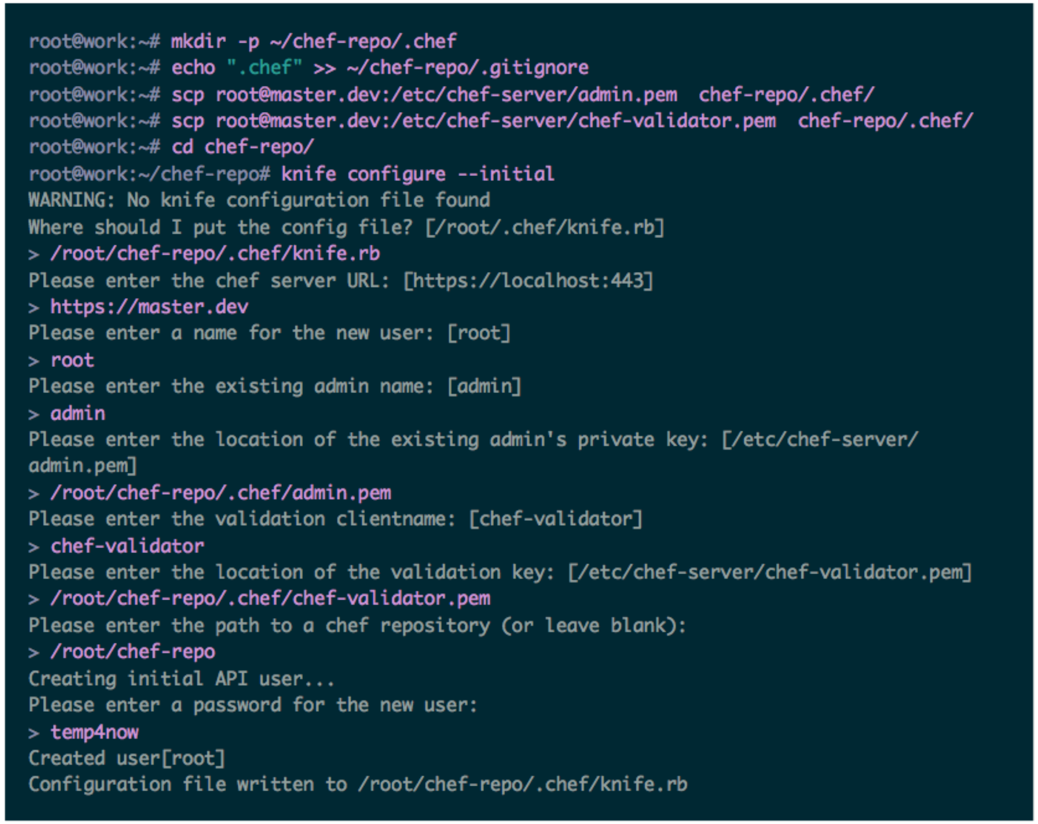
```
root@work:~# apt-get update
root@work:~# apt-get install git -y
```

Clone the `chef-repo`:

```
root@work:~# git clone git://github.com/opscode/chef-repo.git
```

LAMP: Chef

3. Set up the configuration files



```
root@work:~# mkdir -p ~/chef-repo/.chef
root@work:~# echo ".chef" >> ~/chef-repo/.gitignore
root@work:~# scp root@master.dev:/etc/chef-server/admin.pem  chef-repo/.chef/
root@work:~# scp root@master.dev:/etc/chef-server/chef-validator.pem  chef-repo/.chef/
root@work:~# cd chef-repo/
root@work:~/chef-repo# knife configure --initial
WARNING: No knife configuration file found
Where should I put the config file? [/root/.chef/knife.rb]
> /root/chef-repo/.chef/knife.rb
Please enter the chef server URL: [https://localhost:443]
> https://master.dev
Please enter a name for the new user: [root]
> root
Please enter the existing admin name: [admin]
> admin
Please enter the location of the existing admin's private key: [/etc/chef-server/
admin.pem]
> /root/chef-repo/.chef/admin.pem
Please enter the validation clientname: [chef-validator]
> chef-validator
Please enter the location of the validation key: [/etc/chef-server/chef-validator.pem]
> /root/chef-repo/.chef/chef-validator.pem
Please enter the path to a chef repository (or leave blank):
> /root/chef-repo
Creating initial API user...
Please enter a password for the new user:
> temp4now
Created user[root]
Configuration file written to /root/chef-repo/.chef/knife.rb
```

LAMP: Chef

4. Install Chef on Children

```
root@work:~/chef-repo# knife bootstrap puppy.dev -x root
root@work:~/chef-repo# knife bootstrap kitty.dev -x root
```

That was pretty easy no?

LAMP: Chef

5. Create Cookbook

```
root@work:~/chef-repo# knife cookbook create taste
```

Chef Cookbook Files:

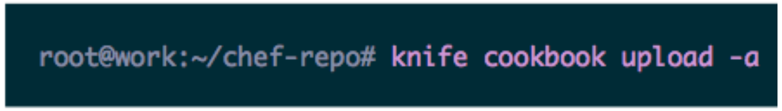
```
/root/chef-repo/cookbooks/taste/templates/default/index.html.erb

/root/chef-repo/cookbooks/taste/recipes/default.rb
```

REF:
https://github.com/noahjs/azphp-configmanagement-presentation/example-code/chef

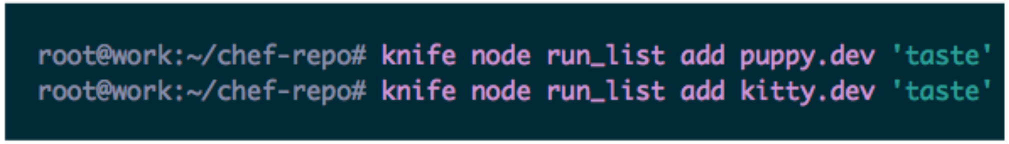
LAMP: Chef

6. Upload the directives and supporting files to the master



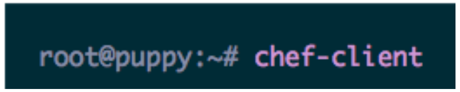
```
root@work:~/chef-repo# knife cookbook upload -a
```

7. Add the directives to the children nodes' "run list"

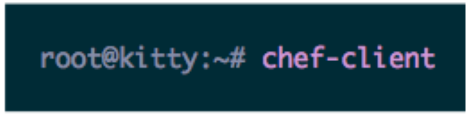


```
root@work:~/chef-repo# knife node run_list add puppy.dev 'taste'
root@work:~/chef-repo# knife node run_list add kitty.dev 'taste'
```

8. Run the directives on the children



```
root@puppy:~# chef-client
```



```
root@kitty:~# chef-client
```

It's not necessary to do this manually, but we do this since we don't want to wait the 30 minutes for it to run automatically.

LAMP: Chef (Notes)

Requires Master w/ 1GB RAM, will fail on 512MB RAM.

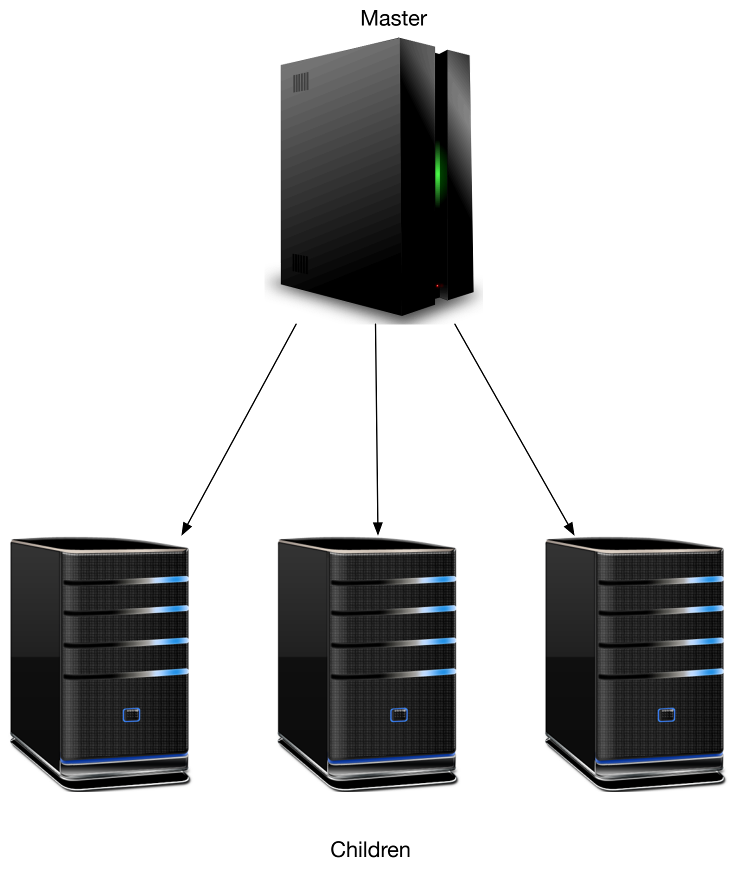
Execution Order: Sequential

Files: Ruby with an extended DSL

LAMP: Saltstack

SALTSTACK

LAMP: Saltstack (Layout)



LAMP: Saltstack (Notes)

Directives = States

Directives Script = SLS Formula (SLS stands for SaLt State)

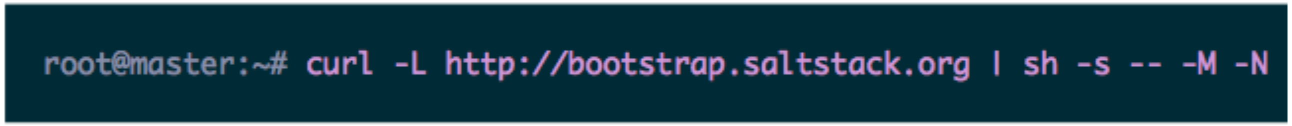
Children Nodes = Minions

Node metadata = Grains

Pillar = interface used to generate arbitrary data for specific minions

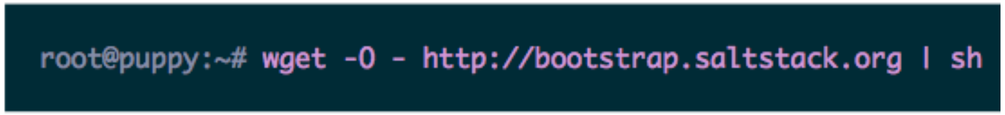
LAMP: Saltstack

1. Install Salt on Master

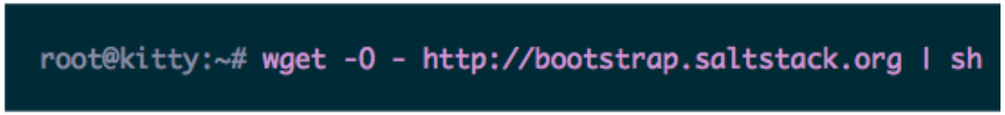


```
root@master:~# curl -L http://bootstrap.saltstack.org | sh -s -- -M -N
```

2. Install Salt on Children



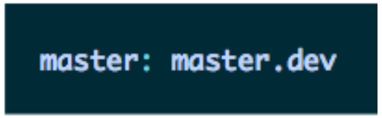
```
root@puppy:~# wget -O - http://bootstrap.saltstack.org | sh
```



```
root@kitty:~# wget -O - http://bootstrap.saltstack.org | sh
```

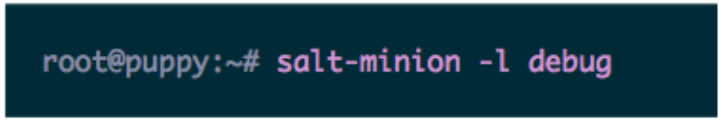
LAMP: Saltstack

3. On child node: Edit /etc/salt/minion

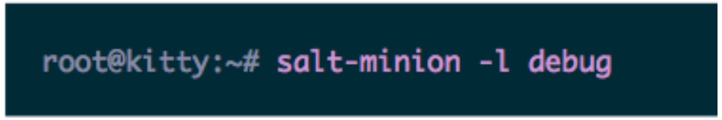


```
master: master.dev
```

3b. Test children



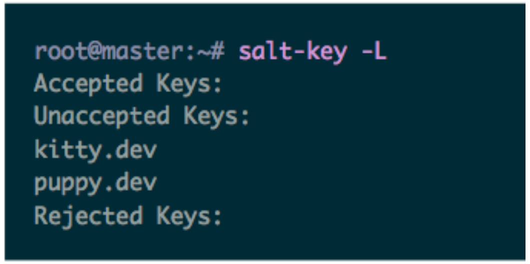
```
root@puppy:~# salt-minion -l debug
```



```
root@kitty:~# salt-minion -l debug
```

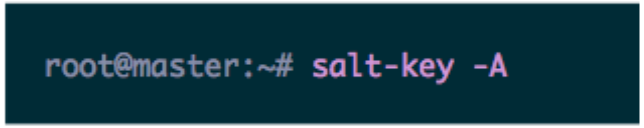
LAMP: Saltstack

4. Check certificate requests



```
root@master:~# salt-key -L
Accepted Keys:
Unaccepted Keys:
kitty.dev
puppy.dev
Rejected Keys:
```

5. Accept requests



```
root@master:~# salt-key -A
```

LAMP: Saltstack

6. Create Directives

```
root@master:~# mkdir /srv/salt
root@master:~# touch /srv/salt/taste.sls
root@master:~# touch /srv/salt/index.html
```

Saltstack Directive Files:

/srv/salt/taste.sls

/srv/salt/index.html

REF:
https://github.com/noahjs/azphp-configmanagement-presentation/example-code/salt

LAMP: Saltstack (Notes)

Execution Order: Sequential

Files: YAML and Jinja2

Runs on ZeroMQ socket between all nodes

Remote execution is standard and robust with Salt

Saltstack: Remote Execution

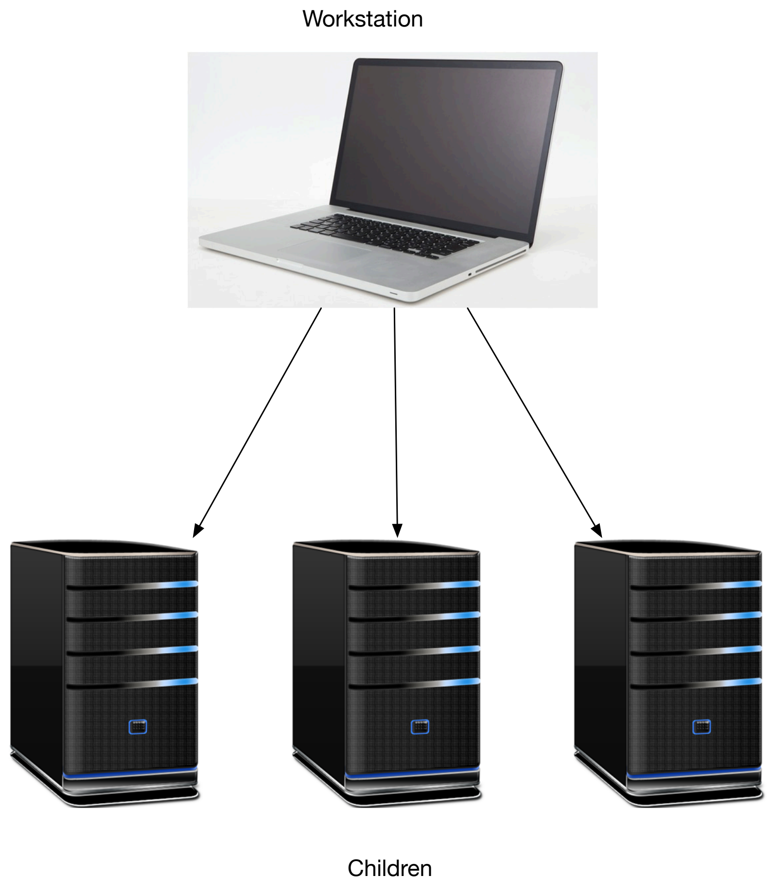
```
root@master:~# salt '*' cmd.run 'date’

puppy.dev:
    Tue Aug  6 08:33:45 UTC 2013
kitty.dev:
    Tue Aug  6 08:33:45 UTC 2013
```

LAMP: Ansible

ANSIBLE

LAMP: Ansible (Layout)



LAMP: Ansible (Notes)

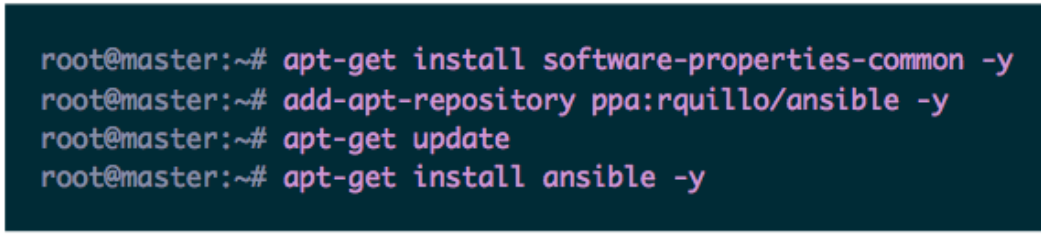
Directives = Tasks

Directives Script = Playbook

Children Nodes = Hosts

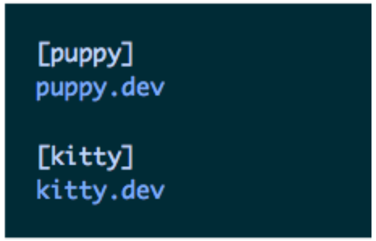
LAMP: Ansible

1. Install Salt on Master



```
root@master:~# apt-get install software-properties-common -y
root@master:~# add-apt-repository ppa:rquillo/ansible -y
root@master:~# apt-get update
root@master:~# apt-get install ansible -y
```

2. Tell Master about children, Edit: /etc/ansible/hosts



```
[puppy]
puppy.dev

[kitty]
kitty.dev
```

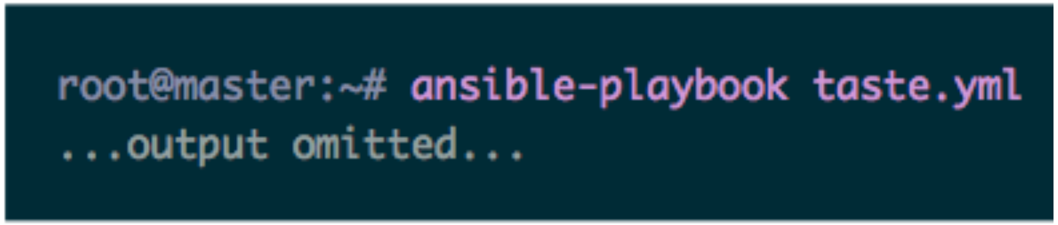
LAMP: Ansible

3. Make sure Master ssh key exists on clients

4. Write Playbook
    - /root/playbook.yml

REF:
https://github.com/noahjs/azphp-configmanagement-presentation/example-code/ansible

5. Run:



```
root@master:~# ansible-playbook taste.yml
...output omitted...
```

LAMP: Ansible (Notes)

Execution Order: Sequential

Files: YAML and Jinja2

Remote execution runs across SSH

Ansible: Remote Execution

```
root@master:~# ansible all –m
 command -a "date”

kitty.dev | success | rc=0 >>
Tue Aug  6 06:23:47 UTC 2013
puppy.dev | success | rc=0 >>
Tue Aug  6 06:23:47 UTC 2013
```

Full Ansible Demonstration

REF:

https://github.com/noahjs/azphp-configmanagement-presentation/full-ansible-example

Requirements:

1. Install Ansible
2. Run “vagrant up” from that directory