Sequentially Ordered Execution in SaltStack

Ryan Lane - Mar 5, 2015

Ryan Lane

DevOps Engineer

Why this talk?

Isn’t Salt sequentially ordered by default?

Requisites

require
watch
order
onchanges
onfail
...

Still though, why?

But don’t requisites ensure good behavior?

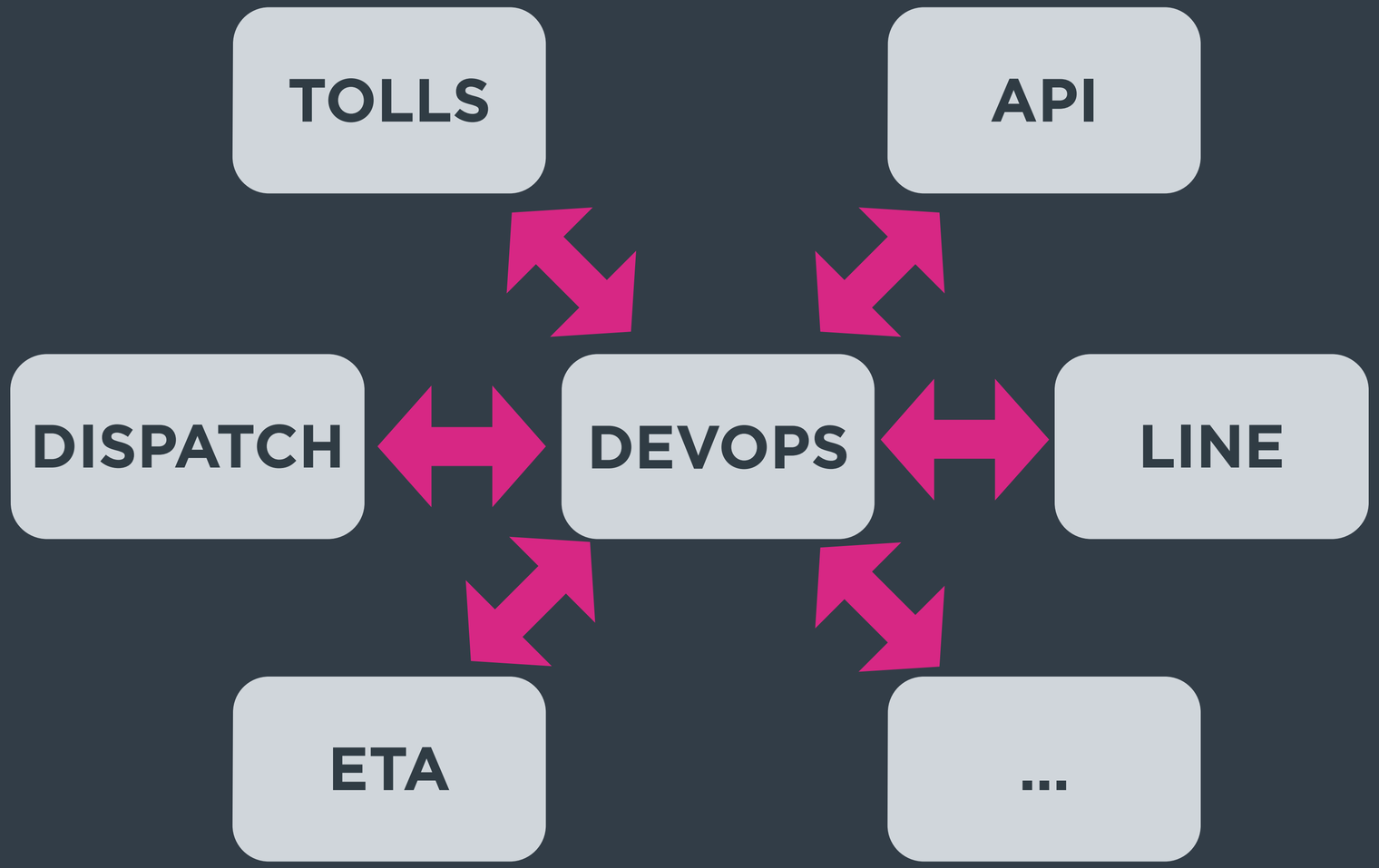
Why

Sequentially ordered states are easier to understand.

Engineering culture

If you build it, you run it.

DevOps as consultants



Architecture Constraints Culture

service
+
base (shared)

Why

Sequentially ordered states are more repeatable.

Deployment

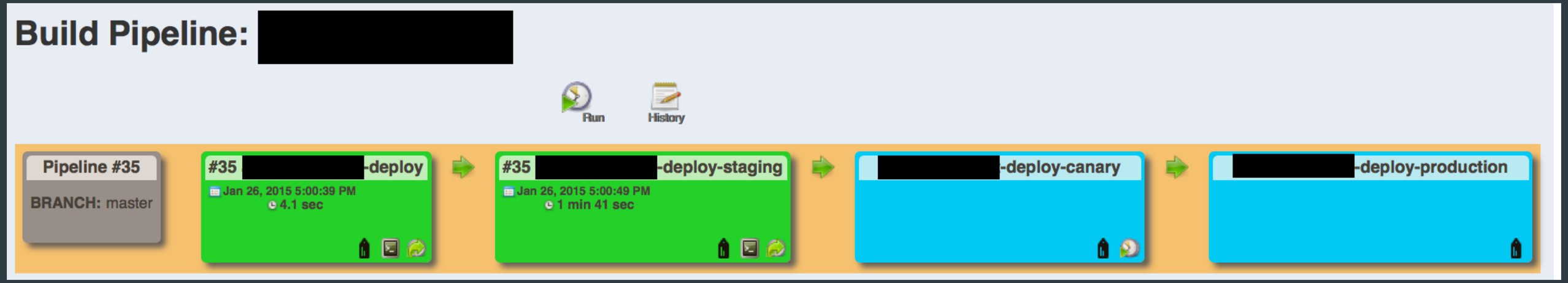
- Make feature branch
- Get review
- Ensure tests pass
- Merge PR
- Start deployment via jenkins pipeline

Deployment

- Generate artifact
- Push artifact to S3
- Release to environment
- Run orchestration

- Fetch artifacts
- Switch ‘next’ links to current SHAs
- Run pre-deploy hook
- Run Salt
- Switch ‘current’ links to current SHAs
- Run post-deploy hook
- Report status

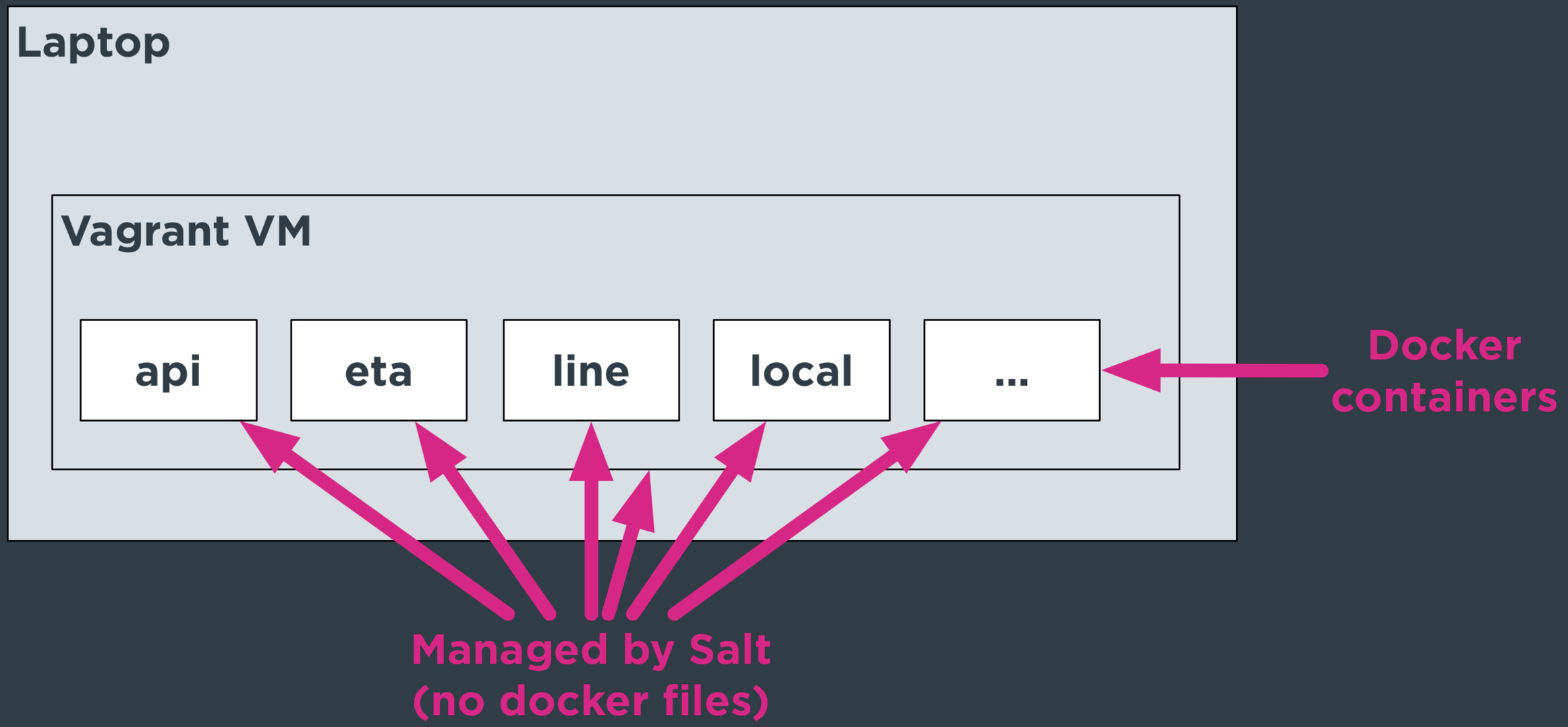
Deployment


Build Pipeline: [redacted]
Run
History
Pipeline #35
BRANCH: master
#35 [redacted]-deploy
Jan 26, 2015 5:00:39 PM
4.1 sec
#35 [redacted]-deploy-staging
Jan 26, 2015 5:00:49 PM
1 min 41 sec
[redacted]-deploy-canary
[redacted]-deploy-production

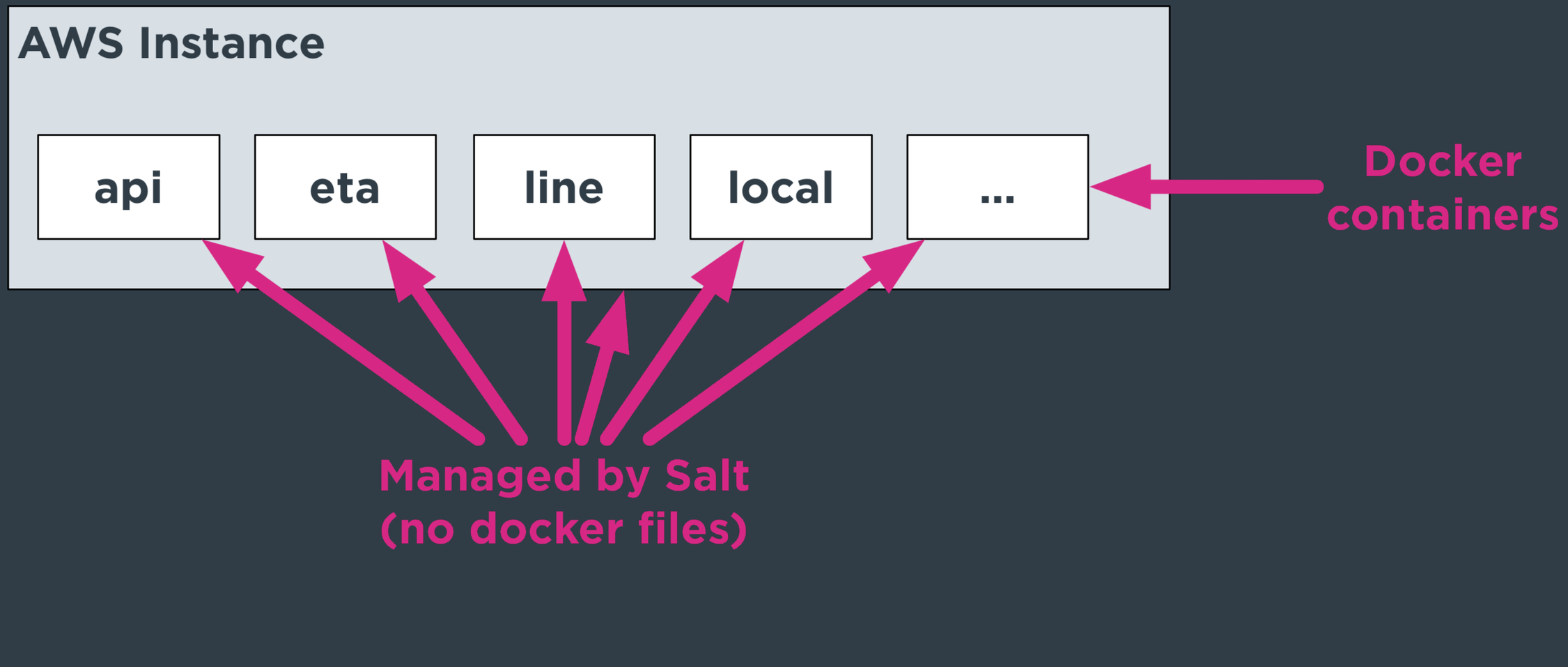
Why

Requisites are unnecessary when building docker images.

Development



Test/CI


AWS Instance
api
eta
line
local
...
Docker containers
Managed by Salt
(no docker files)

Hear more about Docker and AWS

Masterless SaltStack at Scale
Tomorrow, 10:45
Riviera

How

base, then service
Small number of modules
Consistency
Failhard

```
Ensure apache2 is installed:
  pkg.installed:
    - name: apache2

Ensure apache2 is running:
  service.running:
    - name: apache2

Ensure apache2 is configured:
  file.managed:
    - name: /etc/apache2/apache2.conf
    - source: salt://apache2/apache2.conf
   - listen_in:
      - service: apache2
```

listen/listen_in

```
Ensure mongodb is running:
  service.running:
    - name: mongodb

Ensure mongodb is configured:
  file.managed:
    - name: ...
    - source: ...
    - watch_in:
      - service: mongodb

Ensure myuser mongodb user exists:
  mongodb_user.present:
    - name: myuser
    - database: mydatabase
```

watch/watch_in

watch/watch_in

```
Ensure myuser mongodb user exists:
  mongodb_user.present:
    - name: myuser
    - database: mydatabase
    - host: 127.0.0.1
    - port: 27017

Ensure the mydatabase mongodb database is populated:
  cmd.wait:
    - name: /srv/mycode/populatedb.py
    - watch:
      - mongodb_user: myuser
```

Include order

top.sls:

```
base:
  '*':
    - base
    - service
```

service.sls:

```
include:
  - apache
  - redis
```

apache.sls:

```
include:
  - vhost
```

Include order

```
apache.sls:

Ensure apache is installed:
  pkg.installed:
    - name: apache2

include:
  - vhost

Ensure apache is started:
  service.running:
    - name: apache2
```

Include order

base -> vhost -> apache
-> redis -> service

Include order

apache.sls:

```
Ensure apache is installed:
  pkg.installed:
    - name: apache2

{% include 'vhost.sls' %}

Ensure apache is started:
  service.running:
    - name: apache2
```

Jinja + YAML ordering

```
Ensure myelb exists:
  boto_elb.present:
    - name: myelb
...

{% set elb = salt['boto_elb.get_elb_config']('myelb',
profile='myprofile') %}

Ensure myrecord.example.com cname points at ELB:
  boto_route53.present:
    - name: myrecord.example.com.
    - zone: example.com.
...
```

Grain patterns

```
Ensure hostname is set in /etc/hosts:
  host.present:
    - ip:
      - 127.0.0.1
    - names:
      - {{ fqdn }}
      - {{ hostname }}

Ensure hostname is set:
  cmd.run:
    - name: hostname {{ hostname }}
    - unless: hostname | grep {{ hostname }}
    - reload_grains: True
```

Grain patterns

```
state.highstate:

Ensure deployer runs postdeploy:
  module.wait:
    - name: grains.setval
    - key: postdeploy_needed
    - val: True

state.sls:
...
Mark postdeploy finished:
  grains.present:
    - name: postdeploy_needed
    - value: False
```

Pillar patterns

```
Ensure etcd is aware of the node:
  etcd.present:
    - name: servers/{{ grains.service_node }}/address
    - value: {{ grains.ip_interfaces.eth0.0 }}
    - reload_pillar: True

Ensure etcd supervisor config exists:
  file.managed:
    - name: /etc/supervisor/conf.d/etcd.conf
    - source: salt://etcd/config/supervisor.conf
    - template: jinja
    - listen_in:
      - cmd: Reload supervisor
```

Embracing out-of-order states

```
Ensure motd is set:
  file.managed:
    - name: /etc/motd
    - source: salt://base/motd
    - template: jinja
    - order: last
```

Config

Salt config

```
failhard: True
state_output: mixed
state_verbose: False
log_level: info
file_client: local
file_roots:
  base:
    - /srv/service/next/salt/config/states
    - /srv/base/next/salt/config/states
pillar_roots:
  base:
    - /srv/service/next/salt/config/pillar
    - /srv/base/next/salt/config/pillar
module_dirs:
  - /srv/service/next/salt/config/modules
  - /srv/base/next/salt/config/modules
ext_pillar:
  <redacted>
```

Pillar Top

```
base:
  '*':
    - base
    - {{ grains.service_name }}
  'service_group:{{ grains.service_group }}':
    - match: grain
    - {{ grains.service_group }}
    - ignore_missing: true
```

State Top

```
base:
  '*':
    - base
    - {{ grains.service_name }}
```

Sequential ordering issues

Formulas are mostly unusable.
Some modules require requisites.
Failhard support is spotty.

Features wanted

Queue requisite.
Better include support.

Thank You.

@squiddlane

rlane@lyft.com