Salt Remote execution & config

START HERE

`salt '*' test.ping`
Which minions answer?

`salt '<target>' cmd.run "<cmd>"`
Run a shell command

`salt-key -L`
List minion keys

`salt-key -a <id>`
Accept a minion

`salt '<target>' state.apply test=True`
Dry-run the highstate

`salt '<target>' state.apply`
Apply the highstate

`salt '<target>' grains.items`
Facts about a minion

`salt-call --local <function>`
Run on this machine only

CONCEPTS

> Always quote the target: an unquoted * is expanded by your shell. Run state.apply with test=True first to see what would change.

salt (master) you type commands
↓ target + function
Minions matched by ID, grain, pillar
↓ execute
Modules / states cmd.run, pkg.install, state.apply

RUNNING COMMANDS

`salt '<target>' <module.function> [args]`
Run a function on minions
Synopsis of every remote call; the target comes first.

`salt '*' test.ping`
Check minions respond
Not ICMP: it confirms the minion process answers the master.

`salt '<target>' cmd.run "<cmd>"`
Run a shell command

`salt '<target>' sys.doc <module>`
Documentation of a module

`salt '<target>' saltutil.sync_all`
Sync custom modules and grains

TARGETING

`salt 'web*' <fn>`
Glob on minion ID (default)

`salt -L 'a,b,c' <fn>`
Explicit list of IDs

`salt -E 'web[0-9]+' <fn>`
Regex on minion ID

`salt -G 'os:Ubuntu' <fn>`
Match a grain value

`salt -I 'role:db' <fn>`
Match a pillar value

`salt -N <nodegroup> <fn>`
Use a group from the master config

TARGETING (cont.)

`salt -C 'G@os:Ubuntu and web* not L@web9' <fn>`
Combine matchers
Letter prefixes: G grain, I pillar, L list, E regex, S subnet, N nodegroup.

`salt -b 10% '<target>' <fn>`
Run in batches
Takes a count or a percentage; good for rolling restarts.

STATES

`salt '<target>' state.apply`
Apply the highstate (top.sls)

`salt '<target>' state.apply test=True`
Dry run, change nothing

`salt '<target>' state.apply <sls>`
Apply one state file or directory
Use dots for paths: webserver.nginx maps to webserver/nginx.sls.

`salt '<target>' state.show_highstate`
Show the compiled highstate

`salt '<target>' state.show_sls <sls>`
Show a rendered state file

`salt --state-output=changes '<target>' state.apply`
Print only states that changed

PACKAGES & SERVICES

`salt '<target>' pkg.install <pkg>`
Install a package

`salt '<target>' pkg.remove <pkg>`
Remove a package

`salt '<target>' pkg.list_pkgs`
Installed packages

`salt '<target>' service.status <svc>`
Is the service running?

`salt '<target>' service.restart <svc>`
Restart a service

`salt '<target>' service.enable <svc>`
Start at boot

GRAINS & PILLAR

`salt '<target>' grains.items`
All grains

`salt '<target>' grains.get <key>`
One grain
Nested keys use a colon: ip4_interfaces:eth0.

`salt '<target>' grains.setval <key> <val>`
Set a custom grain
Stored in the minion's grains file.

`salt '<target>' pillar.items`
All pillar data

`salt '<target>' pillar.get <key>`
One pillar value

`salt '<target>' saltutil.refresh_pillar`
Reload pillar after editing

KEYS

`salt-key -L`
List accepted, pending, rejected

`salt-key -a <id>`
Accept a pending key

`salt-key -d <id>`
Delete a key

`salt-key -f <id>`
Show a key fingerprint
Compare with salt-call --local key.finger on the minion before accepting.

SALT-CALL & SALT-RUN

`salt-call --local <fn>`
Run a function without a master
Reads the local minion config; handy for testing states.

`salt-call state.apply test=True`
Dry run on this minion

`salt-call --local state.apply <sls> -l debug`
Debug a state locally

`salt-call --out=json <fn>`
Output as JSON

`salt-run manage.up`
Minions that are up

SALT-CALL & SALT-RUN (cont.)

`salt-run jobs.active`
Jobs running now

`salt-run jobs.list_jobs`
Recent jobs

`salt-run jobs.lookup_jid <jid>`
Result of a past job

`salt --async '<target>' <fn>`
Return a job ID, do not wait

OUTPUT & OPTIONS

`salt --out=json '<target>' <fn>`
JSON output

`salt -t <seconds> '<target>' <fn>`
Wait longer for replies
Default is 5 seconds; slow commands may show 'Minion did not return'.

`salt --versions-report`
Salt and dependency versions

RECIPES

Safe rollout

```
salt 'web*' state.apply nginx test=True
salt -b 25% 'web*' state.apply nginx
salt 'web*' service.status nginx
```

Onboard a minion

```
# on the minion: id in /etc/salt/minion_id
systemctl restart salt-minion
# on the master
salt-key -L
salt-key -a web1
salt 'web1' test.ping
```

Minimal state + top file

```
# /srv/salt/top.sls
base:
  'web*':
    - nginx
# /srv/salt/nginx.sls
nginx:
  pkg.installed: []
  service.running:
    - enable: True
```

Target by role and find stragglers

```
salt -C 'G@os:Ubuntu and I@role:web' pkg.version nginx
salt-run manage.down
salt-run jobs.list_jobs
```