Salt Remote execution & config

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

`salt '*' test.ping`
Check minions respond

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

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

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 -C 'G@os:Ubuntu and web* not L@web9' <fn>`
Combine matchers

`salt -b 10% '<target>' <fn>`
Run in batches

STATES

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

STATES (cont.)

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

`salt '<target>' state.apply <sls>`
Apply one state file or directory

PACKAGES & SERVICES

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

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

GRAINS & PILLAR

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

`salt '<target>' grains.get <key>`
One grain

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

`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-CALL & SALT-RUN

`salt-call --local <fn>`
Run a function without a master

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

OUTPUT & OPTIONS

`salt -t <seconds> '<target>' <fn>`
Wait longer for replies