mrxpalmeiras

Search this site

Navigation

Ansible
AWS
Docker
Crystal
Elastic
FiSH
Git
Golang
JavaScript
Linux
Puppet
Python
Ruby
SaltStack
More...
Scratchpad
Ninja Tools

SaltStack >

Salt Cheat Sheet

Contents

SERVICES

Start / Stop / Restart service on Minion
```
salt 'target' service.start "service name"
 (start/stop/restart)
```

Restart Minion on Win target
```
salt 'target' cmd.run 'start powershell "Restart-Service -
Name salt-minion"'
```

Restart Minion on Linux target
```
salt 'target' cmd.run 'service salt-minion restart"'
```

Execute a script remotely

FILE (

Check if file
```
salt '*' f:
```

Check if a fi
```
salt "*" f
```

check if file
```
salt target
```

Find a file
```
salt '*' f:
```

```
salt target cmd.exec_code python 'import sys; print sys.version'
2.7.8 GCC 4.9.1

salt target cmd.exec_code sh 'echo $PATH'
/usr/local/bin:/usr/local/sbin
```

Check service on minion
```
salt target service.status httpd
```

Check if service is available
```
salt target service.available httpd
```

get all services
```
salt target service.get_all
```

reload a service config (avoids restart)
```
salt target service.reload httpd
```

start | stop | restart a service
```
salt target service.start httpd
```

TARGETING & OUTPUT

by OS grain
```
salt -G os:Windows cmd.run "net stop Firewall"
```

by other grains
salt –G 'server_type:app and env:prod' state.highstate

target EC2 instances only
salt -G uuid:ec2* test.ping

compound match
salt –C 'server_type:web and clo*' state.sls nginx

list based match
```
salt –L 'hostname1,hostname2,hostname3' state.sls ntp
```

Nodegroup match
salt –N ny_db_servers cmd.run 'ps –ef | grep mysql'

add to Nodegroup filter
vim /etc/salt/master.d/nodegroups.conf

```
nodegroups:
  prod:  'L@nycweb1 or myhost or web*'
  uat: '*-uat'
```

regex OR
```
salt -E "(nyweb|db5)" test.ping
```

by pillar value
```
salt -I 'role:webserver' test.ping
```

run Highstate directly from the minion, show only changes
```
salt-call state.highstate test=true --state-output=changes
```

KEY MANAGEMENT

```
result
- /etc/host
- /etc/host
- /etc/host
```

copy small f
```
salt-cp 'ta
```

copy dir fror
```
salt 'targe
```

copy large f
```
salt 'targe
```

copy file f
https://do

add Minic
```
vi /etc/
```

add this li
```
minionfs_
file_rec
```

restart Ma

get the fil
```
salt 'ta
```

files are s
```
/var/cacl
```

copy file fro

```
on Maste

to copy

salt \*

all file
```

add host en
```
salt targe
```

Replace cor
```
salt '*' f
```

Create folde
```
salt '*'
C:/temp/c
```

Delete folde
```
salt '*' f
```

Create new
```
salt '*' f
```

manage file

```
/etc/fst
  file.
    - cc
    - mc
    - af
```

replace con

Add Minions to Master
```
salt-key -L (show pending to be accepted)
salt-key -A (accept all pending)
salt-key -a target (accept by hostname)
```

Remove inactive minions from Salt
```
salt-run manage.down removekeys=True
```

Remove minions by name
```
salt-key -D targetName
```

SERVER DIAGNOSTICS

Test Connection
```
salt 'target' test.ping
```

Diagnostics
```
salt target status.all_status // gets all info
status.cpu_info
status.cpustats
status.uptime
status.diskusage  // or disk.usage
status.loadavg
status.meminfo
status.netdev  // network device
status.netstats //network stats
status.procs
status.version //system version
status.vmstats //virtual mem stats
status.w  //who is logged in
```

Show Minions by State (Up/Down)
```
 salt-run manage.up
 salt-run manage.down
 salt-run manage.status  (show all by status)
```

Compliance and Audit

to get a compliance result, run a State check with test=True
salt * state.highstate test=True

This will return any differences from existing configuration to whats in the Top file

Show Salt Master version
salt --versions-report

Show Salt Minion version
```
salt-call --versions-report
```

Start Minion in Debug mode
```
salt-master --log-level=debug
```

Restart everything on Master:
```
pkill salt-minion  //Kill minion
pkill salt-syndic  // Kill Syndic
salt-run cache.clear_all  //Clear all cache
salt '*' saltutil.sync_grains  //Sync grains
```

```
update_m
  file.r
    - na
    - pa
    - re
    - ap
```

create symli
```
symlink:
  file.sy
    - nam
    - tar
```

create direc
```
/home/qb
  file.c
    - us
    - gr
    - di
    - fi
    - re
```

replace file
```
sshd_confi
  augeas.c
    - cont
    - char
      - se
      - se
      - se
```

copy directo
```
app_ta_n
    file
```

File Manag

try several f
```
monit_co
    file
```

```
salt-master -d
```
//Start master daemon

```
salt-minion -d
```
//Start minion daemon

```
salt-syndic -d
```
//Start syndic daemon

Agent Env Info

show all information about a minion (lots of data)
```
salt minion status.all_status
```

show memory
```
salt minion status.meminfo
```

show disk usage
```
salt minion status.diskusage
```

show who is logged in
```
salt minion status.w
```

GRAINS

Show Grain data
```
salt '*' grains.ls
salt '*' grains.items
```

get specific Grain
```
salt cent7 grains.get selinux
cent7:
    ----------
    enabled:
        True
    enforced:
        Enforcing
```

can also do the same with
```
salt cent7 grains.item selinux
```

set a Grain data on a node
```
salt cent7 grains.set 'apps:Myapp:port' 2500

salt cent7 grains.item apps
cent7:
    ----------
    apps:
        ----------
        Myapp:
            ----------
            port:
                2500
```

All grain data is stored on the minion in **/etc/salt/grains file**
if adding more data manually, refresh Grains on the Master to pick up changes
```
salt target saltutil.refresh_modules
```

Use grain in a state file
```
apache:
  pkg.installed:
    {% if grains['os'] == 'RedHat' %}
    - name: httpd
```

show JSON output

USER

Set user's p
```
salt '*' s
'$6$EYk3o5
```

Generate a
```
salt 'targ
$6$nTul6WP
```

Additional w

1. pytho
2. open

Add User
```
salt target
```

Remove Us
```
salt '*' u
```

Show all use
```
salt target
```

Info on all u
```
salt target
```

Info on spec
salt target u

Add User to
```
salt target
// or
salt target
```

Remove Us
```
salt targe
```

Show users
```
salt target
```

Change Use
```
salt '*' us
```

get info on a
```
salt target
```

get info o a
```
salt targe
```

Delete grou
```
salt targe
```

STATE

Highstate
```
salt '*' st
```

Deploy spec
```
salt '*' st
```

Run multiple
```
salt target
salt target grains.item ipv4 --out=json
{
    "target": {
        "ipv4": [
            "10.0.2.15",
            "127.0.0.1",
            "192.168.56.102"
        ]
    }
}
```

Use grain as a variable

```
{% set nodename = grains['nodename'] %}

base:
  '*':
    - common
    - packages
    - users
    - servers.{{ nodename }}
```

MINE

show mine data

```
salt \* mine.get \* x509.get_pem_entries
```

PACKAGES AND INSTALLATION

Verbose output (timeout 300 sec)

```
salt 'target' state.hightstate -t 300 -v
```

Show package version

```
salt 'target' pkg.version apache
```

install package on minions

salt 'target' pkg.install apache

Uninstall pkg

```
salt 'target' pkg.remove 'npp'
salt 'target' pkg.purge 'npp'
```

Show Installed Packages or Software

```
salt 'target' pkg.list_pkgs
```

show all packages that need updates

```
salt target pkg.list_upgrades
```

Requisites

https://docs.

unless

```
vim:
  pkg.ir
    - ur
      -
      -
```

onlyif

```
set_RTC:
  cmd.ru
    - na
    - or
```

require

```
bar:
  pkg.in:
    - re
      - :
```

onchang

```
extract_
  archiv
    - na
    - so
    - ar
    - on
      - 
```

watch

```
ntpd:
  servic
    - wa
      - 
```

prereq

prereq allow
state contai
statement is

```
graceful
  cmd.ru
    - na
    - pr
      - 
```

use

The use req
need to hav

```
/etc/foo
  file.m
    - sc
    - te
```

upgrade all packages
`salt target pkg.upgrade`

Windows (Chocolatey)
install chocolatey
`salt wintarget chocolatey.bootstrap`

install pkg using choco
`salt mrxwin7 chocolatey.install 7zip`

JOBS AND PROCESS CONTROL

Show all Salt jobs run history
`salt-run jobs.list_jobs`

Show active Salt jobs
`salt-run jobs.active    // returns a Job ID`

Show currently running processes on a minion
`salt '*' saltutil.running`

Kill active job
```
salt 'target' saltutil.kill_job $JOB_ID
salt '*' saltutil.term_job <job id>
```

Clear Job cache
`salt '*' saltutil.clear_cache`

REACTOR

examples of reactor matching

/etc/salt/master.d/reactor.conf
```
reactor:
  - 'sayhello':
    - /srv/reactor/test.sls
```

/srv/reactor/test.sls
```
{% if data['id'].startswith('web') %}
sayhello:
  local.state.apply:
    - tgt: {{ data['id'] }}
    - arg:
      - say-hello

  local.cmd.run:
    - tgt: minion1
    - arg:
      - "echo 'hello' > /tmp hello"

{% endif %}
```

DEBUG

Run highstate in debug
`salt-call -l debug state.highstate`

```
    - mk
    - us
    - gr
    - mc

/etc/bar
  file.m
    - sc
    - us
      -
```

require_i

```
vim:
  pkg.in
    - re
      -
```

import YAMl

```
{% impor

show_cor
  - test
    - te
```

SSH M

edit the Ros

`vi /etc/sa`

# Sample sa
mrxcloud1:
  host: 104.
  user: fred
  passwd: f
  sudo: True

Run comma
salt-ssh -i m

To run SSH

1. copy
   shou
   ```
   /hom
   /hom
   /hom
   /hom
   /hom
   /hom
   /hom
   ```
2. confi

Run specific state in debug
```
salt-call -l debug state.sls elasticsearch
```

show highstate process (debug YAML syntax errors)
```
salt-call state.show_highstate
```

show specific State details
```
salt 'target' state.show_sls apache
```

show only Changed and Failed during run
modify /etc/salt/master and /etc/salt/minion, restart Master after change
```
state_verbose: True
state_output: mixed
```

start minion in debug, see connection errors
```
salt-minion -l debug
```

https://docs.saltstack.com/en/latest/topics/troubleshooting/minion.html

if Master not seeing Minion key requests, add IPTables rules to Master,

```
root@master# iptables -I INPUT -s 172.31.23.0/24 -p tcp -m
multiport --dports 4505,4506 -j ACCEPT
root@master# iptables -I INPUT -s 172.31.25.0/24 -p tcp -m
multiport --dports 4505,4506 -j ACCEPT

# reject everything else,
root@master# iptables -A INPUT -p tcp -m multiport --
dports 4505,4506 -j REJECT
```

Log Jinja variables to Minion
```
{% do salt.log.error('testing jinja logging') -%}
```

show Options passed to a State (ie, test=true)
```
{% do salt.log.error(opts['test']) -%}
```

Output variables from State file,

```
show_var:
    - test.show_notification:
        - text:  This is my var {{ var }}
```

Exit w failure message

```
fail_run:
  test.fail_without_changes:
    - name: your message here
```

3. edit /

4. chan
```
pidf
log_
pki_
cach
```

5. Add
```
mach
  ho
  us
  su
  pr
```

6. run c
```
salt
```

7. apply
```
salt
```

Salt Roster

PILLA

Sample Pilla

Salt top file

/srv/salt/t
```
base:
  '*':
    - co
    - us
```

Pillar top file

/srv/pillar/
```
base:
  '*':
    - us
```

Users Pillar

/srv/pillar/
```
users:
  spider
    uid:
    full
    shel
    ssh-
      - 

  black.
    uid:
    full
    shel
    ssh-
      - 
    superg
      uid:
      full
      shel
      ssh-
        -
        -
```

Salt Users s

/srv/salt/u

```
{% for u

{{ user

  group.
    - gi

  user.p
    - fu
    - ui
    - gi
    - sh
    - ho

{% endfo
```

Refresh pilla
`salt \* sa`

Look at pilla
`salt \* pi`

get a Pillar
`{{ salt['p`

`get Pillar`
`{% for rt`

`get nestec`
`salt nycwe`

PORT

Master - Ag

Get IP of a
`salt targe`

Ping from M
`salt targe`

get all active
`salt targe`

get ARP tab
`salt targe`

test port cor
`salt targe`

get hardwar

```
salt target
```

get intet add

```
salt target
```

get all interf

```
salt target
```

JINJA

For Loop

```
{% for usr
{{ usr }}:
  group:
    - pres
  user:
    - pres
    - gid_
    - requ
      - gr
{% endfor
```

```
If Conditi
```

```
{% if var
    Var is
{% elif va
    var is
{% else %}
    var is
{% endif %
```

get shell cor

```
 {% set pr
```

disable tab s

```
  {% for se
  {{ server
  {% endfor

  results :
     server
     server

  to disabl

  #jinja2:

  to top of
```

Run a state

```
  {% if not

  deploy_ke
      file


  extract_
```

```
    arch

 {% endif
 Test for
 {% if no
```

render para
```
{{ var_name
```

set a param
```
{% set fru
```

Iterate dictio
```
{% for name
   {{ name
   {{ app[
{% endfor
```

sort a list
```
{% for vm
    {{ vm
{% endfor
```

or by attribu
```
 {% for vm
```

get total # o
```
{{ myList|
```

If statement
```
{% if var
```

convert vari
```
{{ somevar
```

set a defaul
```
{{ somevar
```

match by re
```
{% if grai
```

will match a

Jinja Tric

difference
```
{{ [1, 2,
') }}
>> 1
```

avg, min, m
```
{{ [1, 2, 3
```

generate ra
```
{{ 'random
```

date format
```
{{ 1457456
{{ 1457456
```

```
2017-03-08
08.03.2017

string to
{{ '5' | t

run Salt
{{ salt.c
{{ salt.g
```

ETC

generate ra

```
python -c
crypt.mks
```

Subpages (1): Saltstack minicheat

Comments

You do not have permission to add comments.

my stuff https://github.com/perfecto25/