SaltStack Cheat Sheet

A quick reference guide to SaltStack, covering essential commands, configurations, and concepts for efficient task automation and system management.

Core Concepts & Architecture

Key Components

Salt Master	Central control node that issues commands and manages configurations.
Salt Minion	Agent installed on managed nodes that executes commands received from the Salt Master.
Salt States	Declarative configurations written in YAML that define the desired state of a system.
Salt Modules	Python modules that provide functions for managing system resources and services.
Salt Grains	System properties discovered by the Minion and reported to the Master (e.g., OS, architecture).
Salt Pillars	Secure data store for sensitive information (e.g., passwords, API keys) that can be used in States and Modules.

Communication Flow

1. Master authenticates Minions using cryptographic keys.
2. Master sends commands and States to Minions.
3. Minions execute commands and apply States.
4. Minions return execution results to the Master.

Configuration Files

Master Configuration:	`/etc/salt/master`
Minion Configuration:	`/etc/salt/minion`
Pillar Data:	`/srv/pillar`
State Files:	`/srv/salt`

Common Salt Commands

Basic Commands

`salt '*' test.ping`	Check the connectivity to all minions.
`salt 'minion_id' test.ping`	Check the connectivity to a specific minion.
`salt '*' state.apply`	Apply all states to all minions.
`salt 'minion_id' state.apply`	Apply all states to a specific minion.
`salt '*' cmd.run 'command'`	Execute a shell command on all minions.
`salt 'minion_id' cmd.run 'command'`	Execute a shell command on a specific minion.

Targeting Minions

`salt -E '.*'`	Target all minions using regular expression.
`salt -G 'os:Ubuntu' test.ping`	Target minions with the 'os' grain equal to 'Ubuntu'.
`salt -I 'role:webserver' test.ping`	Target minions with the 'role' grain equal to 'webserver'.
`salt -L 'minion1,minion2' test.ping`	Target a list of minions.
`salt -N 'web* and db*' test.ping`	Target minions using compound matching.

State Management

`salt '*' state.highstate`	Apply all states defined in the top file.
`salt 'minion_id' state.sls 'statename'`	Apply a specific state to a minion.
`salt '*' state.show_highstate`	Show the compiled highstate for all minions.
`salt '*' state.show_sls 'statename'`	Show the compiled state for a specific SLS file.
`salt '*' state.test`	Dry run, show changes without applying them.

Salt States and SLS Files

Basic SLS Structure

SLS (Salt State) files are written in YAML and define the desired state of a system.

```
# /srv/salt/apache.sls
apache:
  pkg.installed:
    - name: httpd
  service.running:
    - name: httpd
    - enable: True
    - require:
      - pkg: apache
```

Common State Modules

`pkg.installed`	Install a package.
`pkg.removed`	Remove a package.
`service.running`	Ensure a service is running.
`service.stopped`	Ensure a service is stopped.
`file.managed`	Manage a file's content.
`file.absent`	Ensure a file is absent.
`user.present`	Create a user.
`user.absent`	Remove a user.

State Requisites

`require`	Ensures a state is executed before the current state.
`require_in`	Ensures the current state is executed before another state.
`watch`	Executes a state when another state changes.
`watch_in`	Another state executes when the current state changes.
`use`	Includes another state as if its declarations were part of the current state.

Pillars and Grains

Pillar Data

Pillars are used to define sensitive data that should be available only to certain minions.

```
# /srv/pillar/top.sls
base:
  '*':
    - secrets

# /srv/pillar/secrets.sls
password: 'mysecretpassword'
```

Accessing Pillar Data

In States:	`{{ pillar['password'] }}`
In Jinja Templates:	`{{ salt['pillar.get']('password') }}`
From the Command Line:	`salt '*' pillar.get password`

Grains

Grains are system properties that are automatically discovered by the Minion and made available to the Master.

Examples:
`os`, `os_family`, `kernel`, `architecture`, `ip_address`

Accessing Grains

In States:	`{{ grains['os'] }}`
In Jinja Templates:	`{{ salt['grains.get']('os') }}`
From the Command Line:	`salt '*' grains.get os`