Chef Cheatsheet

A comprehensive cheat sheet for Chef, covering essential concepts, commands, resources, and best practices for infrastructure automation and configuration management.

Core Concepts & Architecture

Key Components

Chef Server	Central repository for cookbooks, roles, environments, and node metadata. The heart of the Chef infrastructure.
Chef Workstation	Local machine used to develop, test, and upload cookbooks to the Chef Server. Your development environment.
Chef Client	Agent installed on each node (server) that pulls configuration information from the Chef Server and applies it to the node.
Chef Infra Client	Command-line interface (CLI) to run the Chef Infra Client (formerly just Chef Client) on a node. `chef-client` command.
Chef Automate	Provides visibility and control over your Chef infrastructure. Includes features like compliance scanning and reporting.
Ohai	A tool within the Chef client that gathers system configuration data (e.g., OS, network, CPU) and makes it available as node attributes.

Chef Workflow

1. **Develop Cookbooks:** Create and modify cookbooks on your Chef Workstation.
2. **Test Cookbooks:** Use testing tools like Test Kitchen and ChefSpec to ensure cookbooks function correctly.
3. **Upload Cookbooks:** Upload tested cookbooks to the Chef Server.
4. **Configure Nodes:** Define node attributes and run lists, either directly or via roles and environments.
5. **Run Chef Client:** Execute `chef-client` on each node to apply the configurations defined in the cookbooks.
6. **Converge:** The process of bringing a node into the desired state defined by the cookbooks.

Chef Objects

Cookbooks	The fundamental unit of configuration in Chef. Contain recipes, attributes, and other resources that define how to configure a system.
Recipes	Contain instructions (resources) that specify how to configure a specific aspect of a node. Written in Ruby.
Resources	Represent a desired state for a system component (e.g., a file, a package, a service). Chef provides a wide range of built-in resources.
Attributes	Variables that define aspects of a node's configuration. Used to customize cookbooks for different nodes or environments.
Roles	A way to group and apply cookbooks, recipes, and attributes to nodes based on their function (e.g., web server, database server).
Environments	Define different stages of your infrastructure (e.g., development, staging, production). Allow you to apply different configurations to nodes in different environments.

Knife Command Reference

Node Management

`knife node list`	Lists all nodes registered with the Chef Server.
`knife node show <node_name>`	Displays the attributes and run list for a specific node.
`knife node edit <node_name>`	Opens the node data in your default editor for modification.
`knife node delete <node_name>`	Deletes a node from the Chef Server.
`knife node run_list add <node_name> 'role[<role_name>]'`	Adds a role to a node's run list.
`knife node run_list remove <node_name> 'recipe[<recipe_name>]'`	Removes a recipe from a node's run list.

Cookbook Management

`knife cookbook create <cookbook_name>`	Generates a basic cookbook structure.
`knife cookbook upload <cookbook_name>`	Uploads a cookbook to the Chef Server.
`knife cookbook download <cookbook_name>`	Downloads a cookbook from the Chef Server.
`knife cookbook list`	Lists all cookbooks available on the Chef Server.
`knife cookbook show <cookbook_name> <version>`	Shows details for a specific cookbook version.
`knife cookbook delete <cookbook_name> <version>`	Deletes a cookbook from the Chef Server.

Role and Environment Management

`knife role create <role_name>`	Creates a new role.
`knife role from file <role_name>.json`	Creates a role from a JSON file.
`knife role show <role_name>`	Displays details for a role.
`knife environment create <environment_name>`	Creates a new environment.
`knife environment from file <environment_name>.json`	Creates an environment from a JSON file.
`knife environment show <environment_name>`	Displays details for an environment.

Common Chef Resources

File Management

```
file '/path/to/file' do
  content 'This is the content of the
file'
  owner 'user'
  group 'group'
  mode '0644'
  action :create
end
```

Creates or modifies a file with specified content, ownership, and permissions.

```
cookbook_file '/path/to/destination' do
  source 'source_file_in_cookbook'
  owner 'user'
  group 'group'
  mode '0644'
  action :create
end
```

Copies a file from the cookbook's `files` directory to a specified destination on the node.

```
template '/path/to/destination' do
  source 'template_file.erb'
  owner 'user'
  group 'group'
  mode '0644'
  variables({:variable1 => 'value1',
:variable2 => 'value2'})
  action :create
end
```

Generates a file from a template (ERB) file in the cookbook's `templates` directory.

```
directory '/path/to/directory' do
  owner 'user'
  group 'group'
  mode '0755'
  action :create
end
```

Creates a directory with specified ownership and permissions.

Package and Service Management

```
package 'package_name' do
  action :install
  version '1.2.3'
end
```

Installs, upgrades, or removes a package on the node.

```
service 'service_name' do
  action [:enable, :start]
  supports :status => true, :restart =>
true
end
```

Manages a system service, including starting, stopping, enabling, and disabling.

Execution and User Management

```
execute 'command_name' do
  command 'some_command'
  user 'user'
  cwd '/path/to/directory'
end
```

Executes a command on the node.

```
user 'user_name' do
  action :create
  home '/home/user_name'
  shell '/bin/bash'
  password 'hashed_password'
end
```

Creates, modifies, or deletes a user account on the node.

```
group 'group_name' do
  action :create
  members ['user1', 'user2']
end
```

Creates, modifies, or deletes a group on the node.

Chef Best Practices

Cookbook Structure

A well-structured cookbook is essential for maintainability and reusability. Common directories include:

- `recipes/`: Contains the main configuration logic.
- `attributes/`: Defines node attributes.
- `templates/`: Stores ERB template files.
- `files/`: Contains static files to be copied to nodes.
- `resources/`: Custom resources (advanced).
- `providers/`: Custom providers for custom resources (advanced).
- `libraries/`: Ruby helper functions.
- `test/`: Unit and integration tests.

Idempotency

Ensure that your recipes are idempotent, meaning that running them multiple times has the same effect as running them once. Use `not_if` and `only_if` guards to prevent unnecessary actions.

```
file '/path/to/file' do
  content 'This is the content of the file'
  not_if { ::File.exist?('/path/to/file') }
end
```

This ensures the file is only created if it doesn't already exist.

Testing

Use testing tools like ChefSpec and Test Kitchen to thoroughly test your cookbooks before deploying them to production. ChefSpec provides unit testing capabilities, while Test Kitchen allows you to test your cookbooks in a virtualized environment.

ChefSpec Example:

```
require 'chefspec'

describe 'my_cookbook::default' do
  let(:chef_run) { ChefSpec::SoloRunner.converge(described_recipe) }

  it 'creates a file' do
    expect(chef_run).to create_file('/path/to/file')
  end
end
```

Attribute Management

Use attributes to externalize configuration values and make your cookbooks more flexible. Define default attributes in `attributes/default.rb` and override them in roles, environments, or node-specific attributes.

Attribute precedence:

1. `default` attributes (lowest precedence)
2. `force_default` attributes
3. `normal` attributes
4. `override` attributes
5. `force_override` attributes (highest precedence)