I have reviewed the document for chef. The propose of the document is for an admin to pick it up and know how to use chef and deploy changes to the dsw hosts.

So have to presume the user knows nothing.

• Can you construct the document in the following way.

o High level overview

<https://docs.chef.io/chef_overview.html>

o The components involved. (cookbook, nodes, recipes etc)

Cookbooks:

A cookbook is the fundamental unit of configuration and policy distribution. A cookbook defines a scenario and contains everything that is required to support that scenario:

* Recipes that specify the resources to use and the order in which they are to be applied
* Attribute values
* File distributions
* Templates
* Extensions to Chef, such as custom resources and libraries

Recipes:

A recipe is the most fundamental configuration element within the organization. A recipe:

* Is authored using Ruby, which is a programming language designed to read and behave in a predictable manner
* Is mostly a collection of resources, defined using patterns (resource names, attribute-value pairs, and actions); helper code is added around this using Ruby, when needed
* Must define everything that is required to configure part of a system
* Must be stored in a cookbook
* May be included in a recipe
* May use the results of a search query and read the contents of a data bag (including an encrypted data bag)
* May have a dependency on one (or more) recipes
* May tag a node to facilitate the creation of arbitrary groupings
* Must be added to a run-list before it can be used by the chef-client
* Is always executed in the same order as listed in a run-list

Node:

A node is any machine—physical, virtual, cloud, network device, etc.—that is under management by Chef.

A chef-client is installed on every node that is under management by Chef. The chef-client performs all of the configuration tasks that are specified by the run-list and will pull down any required configuration data from the Chef server as it is needed during the chef-client run.

o Getting user setup (installing, configuring the applications and where to install them?)

o Server dependencies before chef can be run on node (I can provide you this info) (/app /opt size etc)

All machines in a Chef server deployment have the following hardware requirements. Disk space for standalone and backend servers should scale up with the number of nodes that the servers are managing. A good rule to follow is to allocate 2 MB per node. The disk values listed below should be a good default value that you will want to modify later if/when your node count grows. Fast, redundant storage (SSD/RAID-based solution either on-prem or in a cloud environment) is preferred.

For all deployments:

64-bit architecture

For a standalone deployment:

4 total cores (physical or virtual)

8 GB of RAM or more

5 GB of free disk space in /opt

5 GB of free disk space in /var

For complete prerequisites list , see below link:

<https://docs.chef.io/install_server_pre.html>

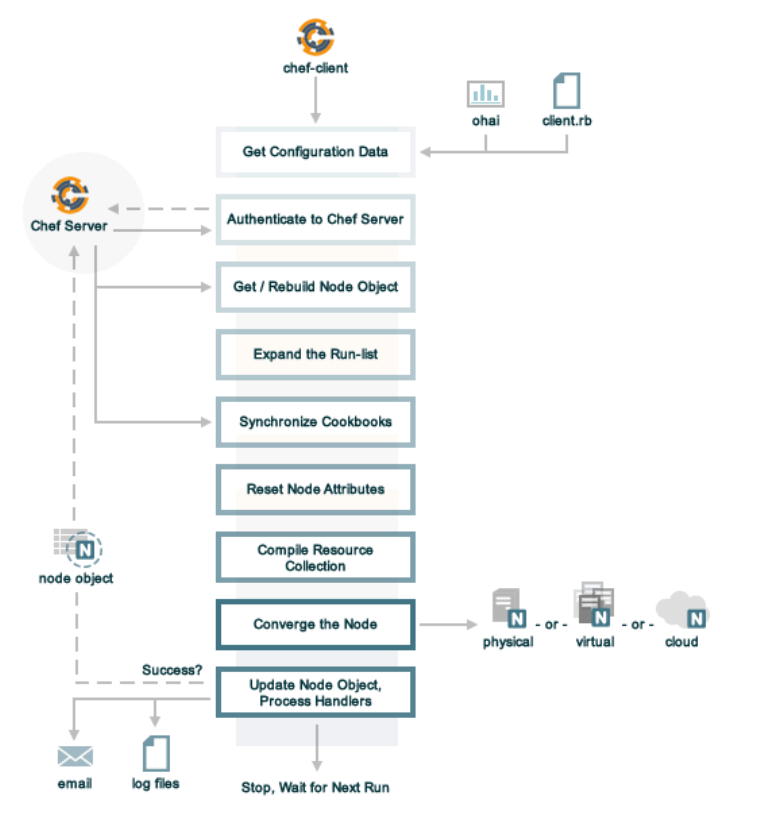
o Zones, environments, zones (currently)

o How to write recipes (best practices, etc)

<https://blog.chef.io/2017/09/28/chef-101-the-road-to-best-practices/>

o How to deploy to chef service

o Flow diagram of chef client execution process



o Troubleshooting / lessons learned

• What are cookbooks, recipes, nodes? (examples)

• What is Jenkins?

Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and deploying software.

Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.

• What is a hook in git? Why do I need it, how do I create it

Git hooks are scripts that run automatically every time a particular event occurs in a Git repository. They let you customize Git’s internal behavior and trigger customizable actions at key points in the development life cycle.

Common use cases for Git hooks include encouraging a commit policy, altering the project environment depending on the state of the repository, and implementing continuous integration workflows. But, since scripts are infinitely customizable, you can use Git hooks to automate or optimize virtually any aspect of your development workflow.

<https://www.atlassian.com/git/tutorials/git-hooks>

• How do you install chef client? Where is installation files.

Bootstrap/Add nodes to chef server:

A bootstrap is a process that installs the chef-client on a target system so that it can run as a chef-client and communicate with a Chef server.

Use the below command to bootstrap a node:

knife bootstrap <ipaddress/hostname> --ssh-user <user-name> --sudo -P <password> --node-name <node-name>

• Chef commands.

Chef generator commands

Usage: chef generate GENERATOR [options]

Available generators:

app Generate an application repo

cookbook Generate a single cookbook

recipe Generate a new recipe

attribute Generate an attributes file

template Generate a file template

file Generate a cookbook file

helpers Generate a cookbook helper file in libraries

lwrp Generate a custom resource

resource Generate a custom resource

repo Generate a Chef code repository

policyfile Generate a Policyfile for use with the install/push commands

generator Copy ChefDK's generator cookbook so you can customize it

build-cookbook Generate a build cookbook for use with Delivery

-h, --help Show this message

-v, --version Show chef version

Chef execution commands:

chef-client Run chef client mode

chef-client -z -o recipe[cookbook] Run chef in local mode

chef-solo -c solo.rb -j node.json Run chef in chef solo mode

For more chef and knife commands:

<https://docs.chef.io/ctl_chef.html>

<https://docs.chef.io/knife.html>

• Difference between test and prod

• How do I test and where do I test?

The workstation is the location from which users interact with Chef. On the workstation users author and test cookbooks using tools such as Test Kitchen, Foodcritic, Rubocop, ChefSpec

Test Kitchen:

Kitchen provides a test harness to execute infrastructure code on one or more platforms in isolation.

A driver plugin architecture is used to run code on various cloud providers and virtualization technologies such as Vagrant, Amazon EC2, and Docker.

<https://docs.chef.io/kitchen.html>

Foodcritic:

Foodcritic is a static linting tool that analyzes all of the Ruby code that is authored in a cookbook against a number of rules, and then returns a list of violations.

<https://docs.chef.io/foodcritic.html>

Cookstyle:

Cookstyle is a linting tool based on the RuboCop Ruby linting tool. Cookstyle helps you author better cookbook code in the following ways:

* Enforcing style conventions and best practices
* Evaluating the code in a cookbook against metrics like “line length” and “function size”
* Helping every member of a team to author similary structured code
* Estabilishing uniformity of source code
* Setting expectations for fellow (and future) project contributors

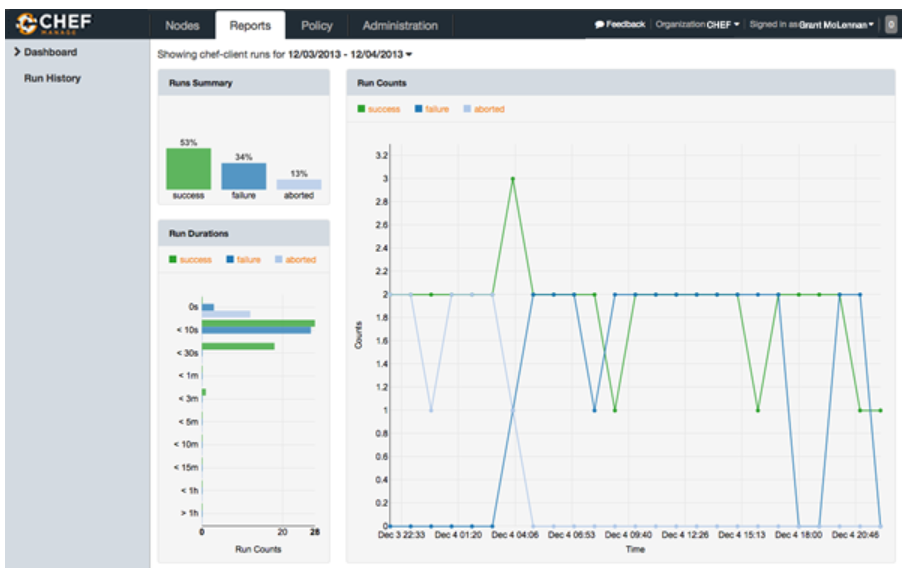
<https://docs.chef.io/cookstyle.html>

• Chef client is always running on the nodes, can you highlight and example is more.

• Is there any Reporting?

The Chef management console provides ways for users to configure the timeframe around which a report is built, and then to review the reports that are available for that timeframe.

Use Reporting to keep track of what happens during the execution of chef-client runs across all of the machines that are under management by Chef.



• What happens if a deployment fails, how can I tell?

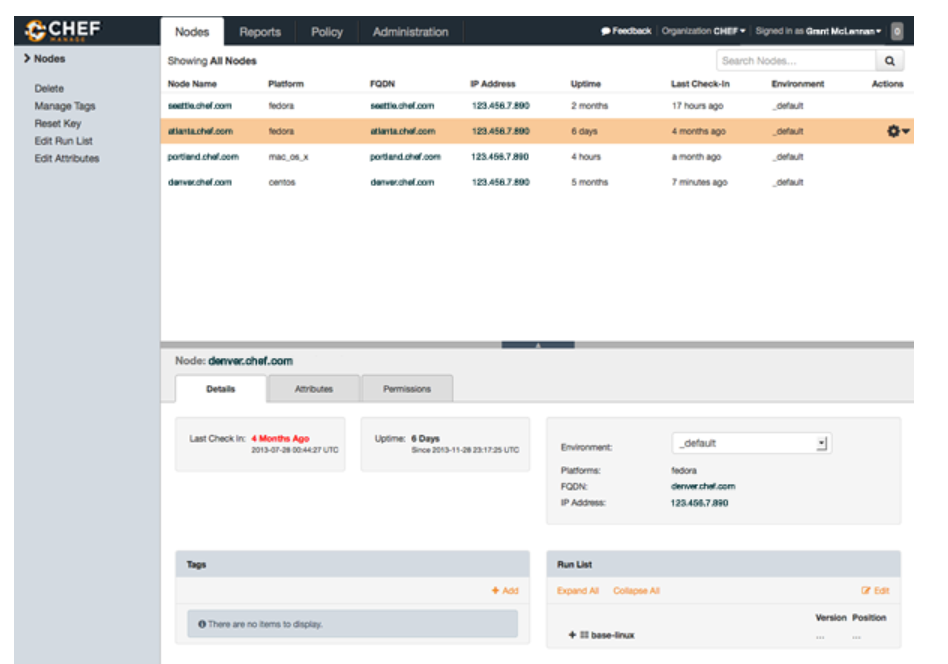
When a deployment fails during cookbook execution, chef client will report a failure status , which can viewed in the reports of failure servers in chef manage console.

• How do I manage nodes, how do I know what nodes are configed with chef

The Chef management console enables the management of nodes, data bags, roles, environments, and cookbooks by using a web user interface.

For more information, please refer to the below link:

https://docs.chef.io/manage.html



• How do I add a new recipe?

Chef generate commands can be used to generate a new recipe.

Command:

chef generate recipe <cookbook-name> <recipe-name>

eg: chef generate recipe optum-dsi-mapr anaconda

This will generate anaconda.rb in optum-dsi-mapr cookbook recipes folder.

• How do I change a recipe

• How do I add / remove nodes

Bootstrap/Add nodes to chef server:

A bootstrap is a process that installs the chef-client on a target system so that it can run as a chef-client and communicate with a Chef server.

Use the below command to bootstrap a node:

knife bootstrap <ipaddress/hostname> --ssh-user <user-name> --sudo -P <password> --node-name <node-name>

Remove node from chef server:

Use the below command to remove a node from chef server:

knife node delete <node-name>

knife client delete <node-name>

• Section: adding new server as chef client

Bootstrap/Add nodes to chef server:

A bootstrap is a process that installs the chef-client on a target system so that it can run as a chef-client and communicate with a Chef server.

Use the below command to bootstrap a node:

knife bootstrap <ipaddress/hostname> --ssh-user <user-name> --sudo -P <password> --node-name <node-name>

“Navigate to desired cookbook (optum\_mapr or optum\_dsi\_asc)” – how do I naviagate? Where is it?