## CHEF WORKFLOW and ACRONOYMS

**Overview of CHEF**

Chef is a powerful automation platform that transforms infrastructure into code. Whether you’re operating in the cloud, on-premises, or in a hybrid environment, Chef automates how infrastructure is configured, deployed, and managed across your network, no matter its size.



This diagram shows how you develop, test, and deploy your Chef code.

* **The workstation** is the location from which all of Chef is managed, including installing the Chef DK, authoring cookbooks, and using tools like Kitchen, chef-zero (a command-line tool that runs locally as if it were connected to a real Chef server), command-line tools like Knife (for interacting with the Chef server) and chef (for interacting with your local chef-repo), and resources like core Chef resources (for building recipes) and InSpec (for building security and compliance checks into your workflow).[In our case **chef work**]
* **The Nodes** are the machines—physical, virtual, cloud, and so on—that are under management by Chef. The chef-client is installed on each node and is what performs the automation on that machine. [in our case **chefnode9, cehfnode10 and others]**
* **The Chef server** acts as a hub for configuration data. The Chef server stores cookbooks, the policies that are applied to nodes, and metadata that describes each registered node that is being managed by the chef-client. Nodes use the chef-client to ask the Chef server for configuration details, such as recipes, templates, and file distributions.

## Chef Components/others

**Starter kit:** You can download this from chef server (chef-server>administration>organization>starter-kit) .

This kit had .pem files and knife file which is required by chef workstation to communicate with chef server.

**Knife.rb:** knife.rb is an configuration file which hold values like **chef server url , validation\_key and validation\_client\_name** this all together along with ssl certificate is required to communicate with chef server

**Client.rb:** This file is created on node at the bootstrap time. A client.rb file is used to specify the configuration details for the chef-client.

**Chef includes two important command-line tools:**

**Knife :** knife command-line tool is used to interact with nodes or work with objects on the Chef server. For e.g (knife node list, knife ssl fetch etc)

**Chef:** command-line tool is used to work with items in a chef-repo, which is the primary location in which cookbooks are authored, tested, and maintained, and from which policy is uploaded to the Chef server. For e.g (chef-apply, chef-client)

**Knife commands**

* **Knife ssl fetch:**

Use the knife ssl fetch subcommand to copy SSL certificates from an HTTPS server to the trusted\_certs\_dir directory that is used by knife and the chef-client to store trusted SSL certificates

**Knife ssl fetch (**command**)**

* **Uploading Cookbook:**

This command is used to upload the cookbook over chef-server.

**Knife cookbook upload <cookbook name>**

* **Deleting node:**

This command used to deletes node from chef server.

**Knife nodes delete <node-name>**

* **Deleting client :**

This command is used to deletes client over the node

.**Knife client delete <node-name>**

**\*\***Note: sometimes we need to clear the chef folder content over the node also.

* **Bootstrapping a node:**

Use the knife bootstrap subcommand to run a bootstrap operation that installs the chef-client on the target system. The bootstrap operation must specify the IP address or FQDN of the target system.

**Knife bootstrap <address> -x <ssh-user> -P <sshpass> -N <node-name to display on chef server> -r <runlist> -E <Environment>**

\*\*Note: Bootstrapping can also be done without passing a run list, it will only install a chef-client on respective node and register it over chef server. Later we can add run list to it and issue **knife ssh** command to run that run list over respective node.

* **Adding Run list :**

This will add runlist to the existing run list over the respective node

**Knife node run\_list add <node name> <’recipe[cookbook::recipe]’ >**

* **Setting Run list:**

This will add runlist and will remove all the previous run lists over the previous node.

**Knife node run\_list set <node name> <’recipe[cookbook::recipe]’ >:**

* **Removing Run list :**

This will remove run list over the respective node

**Knife node run\_list remove <node name> <’recipe[cookbook::recipe]’ >**

* **Running run list :**

If node is already bootstrapped, and if we need to run a recipe over it than we don’t run bootstrap command again instead we issue knife ssh command to do the task.

**Knife ssh name:<node name> -x <sshuser> -P <pass> “sudo chef-client”**

**T**his command will run the run list set from the above command

**\*\*NOTE:**

**There are many ways to execute cookbook on node, we will be discussing over the two most used methods.**

**Method 1 (All from Workstation)**

1. Create a cookbook **<knife cookbook create ..>**
2. Upload it **<knife cookbook upload ..>**
3. Bootstrap a node **<knife bootstrap .. –x .. –P.. –N ..>**
4. Set a run list over it **<knife node run\_list set .. ..>**
5. Run run list over node. **<knife ssh name:.. –x.. –P.. “sudo chef-client” >**

**This all activities we run from chef work station and no need to log into node**

**Method 2 (running chef-client from node)**

1. Same from **point 1 to point 4** of method 1
2. Just Instead of knife ssh command we log into nod e and run **chef client** command**.**

**This method is not ideal for big infrastructures.**