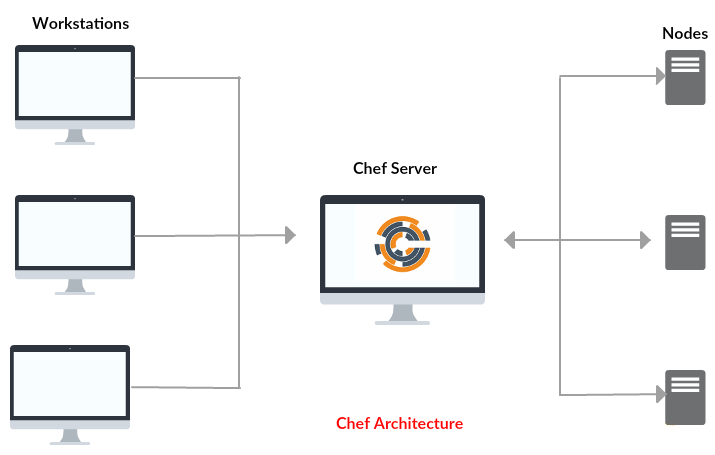


[Chef](https://chef.io) is a powerful [configuration management](http://www.itzgeek.com/tag/configuration-management) utility that turns infrastructure into code. With the Chef users can easily manage, configure and deploy the resources across the network from the centralized location irrespective of the environment (cloud, on-premises, or hybrid).

Starting with the release of Chef server 11, the front end of Chef server is written in Erlang and client uses Ruby to handle configuration changes.

## Components of Chef:



Chef is consist of a Chef server, one or more workstations, and a node where the chef-client is installed. Components name is based on the roles played by each machine in the Chef ecosystem.

**Chef Server**: This is the central hub server that stores the cookbooks and recipes uploaded from workstations, which is then accessed by chef-client for configuration deployment.

**Chef Workstations**: This where recipes, cookbooks, and other chef configuration details are created or edited. All these are then pushed to the Chef server from the workstation, where they will be available to deploy to chef-client nodes.

**Chef Client**: This the target node where the configurations are deployed in which the chef-client is installed. A node can be any machine (physical, virtual, cloud, network device, etc..)

**Setup a Chef 12 on CentOS 7:**

**System Requirements / Environment:**

Make sure your system meets the following resource requirements. Here I will use CentOS 7 as the base operating system for Chef installation.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **HOST NAME** | **IP ADDRESS** | **OS** | **CPU** | **MEMORY** | **DISK** | **PURPOSE** |
| chefserver.itzgeek.local | 192.168.12.11 | CentOS 7 | 4 | 4 GB | 40 GB | Chef Server |
| chefdk.itzgeek.local | 192.168.12.12 | CentOS 7 | 1 | 512 MB | NA | Chef Workstation (Chef Development Kit) |
| chefclient.itzgeek.local | 192.168.12.20 | CentOS 7 | 1 | 512 MB | NA | Chef Client |

## Prerequisites:

Host [should have fully qualified domain name](http://www.itzgeek.com/how-tos/mini-howtos/change-hostname-without-reboot-on-centos-7-rhel-7.html).

Should have [DNS](http://www.itzgeek.com/how-tos/linux/centos-how-tos/configure-dns-bind-server-on-centos-7-rhel-7.html) or Host entry in place.

vi /etc/hosts

192.168.12.11 chefserver.itzgeek.local chefserver

192.168.12.12 chefdk.itzgeek.local chefdk

192.168.12.20 chefclient.itzgeek.local chefclient

Install wget package.

yum -y install wget

## Chef Server:

The Chef server acts as a central hub for configuration data.  The server stores cookbooks, the policies that are applied to nodes and metadata related to a registered node. Registered node uses the chef-client to ask the Chef Server for configuration details, such as recipes, templates, and file distribution.

### Install and Configure the Chef Server:

[Download](https://downloads.chef.io/) the latest version of Chef server core (12.10 at the time of writing).

wget https://packages.chef.io/stable/el/7/chef-server-core-12.10.0-1.el7.x86\_64.rpm

Once the download is complete, install the chef server core using the following command.

rpm -ivh chef-server-core-\*.rpm

If your chef server system does not meet the recommended hardware requirement, this step may fail.

Once the installation is complete, you must reconfigure the chef server components to make up the server to work together. **Reconfiguring may take a little bit longer**.

chef-server-ctl reconfigure

Check the status of Chef Server components by using the following command.

chef-server-ctl status

**Output:**

run: bookshelf: (pid 6084) 387s; run: log: (pid 6114) 385s

run: nginx: (pid 5973) 417s; run: log: (pid 6276) 359s

run: oc\_bifrost: (pid 5816) 477s; run: log: (pid 5831) 476s

run: oc\_id: (pid 5961) 420s; run: log: (pid 5966) 419s

run: opscode-erchef: (pid 6186) 379s; run: log: (pid 6176) 381s

run: opscode-expander: (pid 6039) 388s; run: log: (pid 6071) 388s

run: opscode-solr4: (pid 5992) 399s; run: log: (pid 5999) 398s

run: postgresql: (pid 5805) 478s; run: log: (pid 5809) 477s

run: rabbitmq: (pid 5767) 480s; run: log: (pid 5760) 481s

run: redis\_lb: (pid 5377) 595s; run: log: (pid 6272) 359s

### Create an Admin user and Organization:

We need to create an admin user. This user will have access to make changes to the infrastructure components in the organization we will be creating. Below command will generate the RSA private key automatically and should be saved to a safe location.

User details are below.

**User Name**: admin

**First Name**: admin

**Last Name**: admin

**Email**: admin@itzgeek.local

**Password**: password

**File Name**: admin.pem

**Path**: /etc/chef

chef-server-ctl user-create admin admin admin admin@itzgeek.local password -f /etc/chef/admin.pem

**Original Command:**

chef-server-ctl user-create USER\_NAME FIRST\_NAME LAST\_NAME EMAIL 'PASSWORD' -f PATH\_FILE\_NAME

It is the time for us to create an organization to hold the chef configurations.

**Short Name**: itzgeek (Note: Name must begin with lowercase letter or digit, may contain lowercase letter, numbers, hyphens, and underscores, and must be between 1 and 255 characters)

**Full Organization Name**: ITzGeek Inc (Note: Must begin with non-white space character and must be between 1 and 1023 characters)

**Association User**: admin (Note: This option will associate the previously created user (***admin***) with the ***admins*** security group on the chef server)

**Filename**: itzgeek-validator.pem (Note: command will generate the RSA private key automatically and should be saved to a safe location)

**Path**: /etc/chef

chef-server-ctl org-create itzgeek "ITzGeek, Inc" --association\_user admin -f /etc/chef/itzgeek-validator.pem

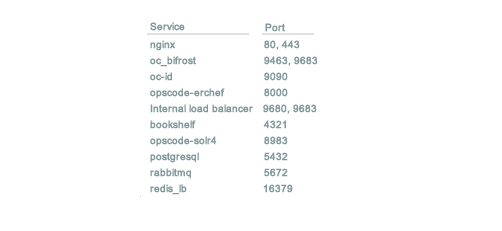
**Original Command:**

chef-server-ctl org-create short\_name 'full\_organization\_name' --association\_user user\_name --filename ORGANIZATION-validator.pem

As of now, you will have two **.pem** keys in **/etc/chef** directory. In our case, they will be called **admin.pem** and **itzgeek-validator.pem**. Soon we will place these two files in Chef workstation machine.

### Firewall:

The Chef server requires the following ports to be open through the firewall. But enabling only **80 and 443** would also do for us.



Setup a Chef 12 on CentOS 7 – Port Numbers

Run the following command to allow **80 and 443** through the firewall.

firewall-cmd --permanent --zone public --add-service http

firewall-cmd --permanent --zone public --add-service https

firewall-cmd --reload

## Chef Workstations:

A workstation is a computer that is configured to the author, test and maintain cookbooks. These cookbooks are then uploaded to Chef server. It is also used to bootstrapping a node that installs the chef-client on nodes.

### Setting up a Workstation:

[Download](https://downloads.chef.io/) the latest version of Chef Development Kit (0.19.6 at the time of writing).

wget https://packages.chef.io/stable/el/7/chefdk-0.19.6-1.el7.x86\_64.rpm

Install ChefDK.

rpm -ivh chefdk-\*.rpm

Verify the components of Chef Development Kit.

chef verify

**Output:**

Running verification for component 'berkshelf'

Running verification for component 'test-kitchen'

Running verification for component 'tk-policyfile-provisioner'

Running verification for component 'chef-client'

Running verification for component 'chef-dk'

Running verification for component 'chef-provisioning'

Running verification for component 'chefspec'

Running verification for component 'generated-cookbooks-pass-chefspec'

Running verification for component 'rubocop'

Running verification for component 'fauxhai'

Running verification for component 'knife-spork'

Running verification for component 'kitchen-vagrant'

Running verification for component 'package installation'

Running verification for component 'openssl'

Running verification for component 'inspec'

Running verification for component 'delivery-cli'

Running verification for component 'git'

Running verification for component 'opscode-pushy-client'

Running verification for component 'chef-sugar'

.................................................................

---------------------------------------------

Verification of component 'kitchen-vagrant' succeeded.

Verification of component 'openssl' succeeded.

Verification of component 'delivery-cli' succeeded.

Verification of component 'test-kitchen' succeeded.

Verification of component 'rubocop' succeeded.

Verification of component 'opscode-pushy-client' succeeded.

Verification of component 'berkshelf' succeeded.

Verification of component 'fauxhai' succeeded.

Verification of component 'inspec' succeeded.

Verification of component 'tk-policyfile-provisioner' succeeded.

Verification of component 'chefspec' succeeded.

Verification of component 'knife-spork' succeeded.

Verification of component 'git' succeeded.

Verification of component 'chef-dk' succeeded.

Verification of component 'chef-sugar' succeeded.

Verification of component 'chef-client' succeeded.

Verification of component 'generated-cookbooks-pass-chefspec' succeeded.

Verification of component 'package installation' succeeded.

Verification of component 'chef-provisioning' succeeded.

Some of the users may want to set Ruby version default to Ruby version installed with Chef. Check the current Ruby location.

which ruby

This command will yield you a result if your machine has Ruby installed. Run the below command to load CheDK variables to user profile file.

echo 'eval "$(chef shell-init bash)"' >> ~/.bash\_profile

Load the user profile.

. ~/.bash\_profile

Now, check the Ruby. You should get the similar output.

# which ruby

/opt/chefdk/embedded/bin/ruby

### Install git:

Before generating chef-repo, you must install an open source version control tool called git on the machine.

yum -y install git

One the installation is complete. Generate Chef-Repo using “**chef generate repo**” command.

cd ~

chef generate repo chef-repo

This command places the basic chef repo structure into a directory called “**chef-repo**” in your home directory.

ls -al ~/chef-repo/

**Output:**

total 32

drwxr-xr-x. 8 root root 4096 Nov 12 18:30 .

dr-xr-x---. 5 root root 4096 Nov 12 18:29 ..

-rw-r--r--. 1 root root 1133 Nov 12 18:29 chefignore

-rw-r--r--. 1 root root  255 Nov 12 18:29 .chef-repo.txt

drwxr-xr-x. 3 root root   36 Nov 12 18:29 cookbooks

drwxr-xr-x. 3 root root   36 Nov 12 18:29 data\_bags

drwxr-xr-x. 2 root root   41 Nov 12 18:29 environments

drwxr-xr-x. 7 root root 4096 Nov 12 18:29 .git

-rw-r--r--. 1 root root  106 Nov 12 18:29 .gitignore

-rw-r--r--. 1 root root   70 Nov 12 18:29 LICENSE

-rw-r--r--. 1 root root 1499 Nov 12 18:29 README.md

drwxr-xr-x. 2 root root   41 Nov 12 18:29 roles

### Add version control:

Setup a user with the email address to begin the git configuration. Replace the “**green**” colored values according to your environment.

git config --global user.name "**admin**"

git config --global user.email "**admin@itzgeek.local**"

Go to the **chef-repo** directory and initialize it.

cd ~/chef-repo/

git init

Now, let’s create a hidden directory called “**.chef**” under the **chef-repo** directory. This hidden directory will hold the RSA keys that we created on the Chef server.

mkdir -p ~/chef-repo/.chef

Since this hidden directory stores the RSA keys, it should not be exposed to the public. To do that we will add this directory to “**.gitignore**” to prevent uploading the contents to GitHub.

echo '.chef' >> ~/chef-repo/.gitignore

Add and commit all existing files.

cd ~/chef-repo/

git add .

git commit -m "initial commit"

Check the status of the directory.

git status

**Output:**

nothing to commit, working directory clean

### Copy the RSA Keys to the Workstation:

The RSA keys (**.pem**) generated when setting up the Chef Server will now need to be placed on the workstation. Place it under “**~/chef-repo/.chef**” directory.

scp -pr root@chefserver:/etc/chef/admin.pem ~/chef-repo/.chef/

scp -pr root@chefserver:/etc/chef/itzgeek-validator.pem ~/chef-repo/.chef/

### Create knife.rb File:

Knife is a command line interface for between a local chef-repo and the Chef server. To make the knife to work with your chef environment, we need to configure it by creating knife.rb in the “**~/chef-repo/.chef/**” directory.

Now, create and edit the knife.rb file using your favorite editor.

vi ~/chef-repo/.chef/knife.rb

In this file, paste the following information:

current\_dir = File.dirname(\_\_FILE\_\_)

log\_level                :info

log\_location             STDOUT

node\_name                "**admin**"

client\_key               "#{current\_dir}/**admin.pem**"

validation\_client\_name   "**itzgeek-validator**"

validation\_key           "#{current\_dir}/**itzgeek-validator.pem**"

chef\_server\_url          "https://**chefserver.itzgeek.local**/organizations/**itzgeek**"

syntax\_check\_cache\_path  "#{ENV['HOME']}/.chef/syntaxcache"

cookbook\_path            ["#{current\_dir}/../cookbooks"]

Adjust the following items to suit for your infrastructure.

**node\_name**: This the username with permission to authenticate to the Chef server. Username should match with the user that we created on the Chef server.

**client\_key**: The location of the file that contains user key that we copied over from the Chef server.

**validation\_client\_name**: This should be your organization’s **short name** followed by **-validator**.

**validation\_key**: The location of the file that contains validation key that we copied over from the Chef server. This key is used when a chef-client is registered with the Chef server.

**chef\_server\_url**: The URL of the Chef server. It should begin with **https://**, followed by **IP address** or **FQDN** of Chef server, organization name at the end just after **/organizations/**.

**{current\_dir}** represents **~/chef-repo/.chef/** directory, assuming that **knife.rb** file is in **~/chef-repo/.chef/**. So you don’t have to write the fully qualified path.

### Testing Knife:

Now, test the configuration by running knife client list command. Make sure you are in **~/chef-repo/** directory.

cd ~/chef-repo/

knife client list

You may get an error like below on your first attempt:

ERROR: SSL Validation failure connecting to host: chefserver.itzgeek.local - SSL\_connect returned=1 errno=0 state=error: certificate verify failed

ERROR: Could not establish a secure connection to the server.

Use `knife ssl check` to troubleshoot your SSL configuration.

If your Chef Server uses a self-signed certificate, you can use

`knife ssl fetch` to make knife trust the server's certificates.

Original Exception: OpenSSL::SSL::SSLError: SSL Error connecting to https://chefserver.itzgeek.local/organizations/itzgeek/clients - SSL\_connect returned=1 errno=0 state=error: certificate verify failed

To resolve this issue, we need to fetch the Chef server’s SSL certificate on our workstation beforehand running the above command.

knife ssl fetch

This command will add the Chef server’s certificate file to trusted certificate directory.

WARNING: Certificates from chefserver.itzgeek.local will be fetched and placed in your trusted\_cert

directory (/root/chef-repo/.chef/trusted\_certs).

Knife has no means to verify these are the correct certificates. You should

verify the authenticity of these certificates after downloading.

Adding certificate for chefserver.itzgeek.local in /root/chef-repo/.chef/trusted\_certs/chefserver\_itzgeek\_local.crt

Once the SSL certificate has been fetched, run the previous command to test the knife configuration.

knife client list

**Output:**

itzgeek-validator

The output confirms the verification has been completed successfully.

## Bootstrapping a New Node with Knife:

Bootstrapping a node is a process of installing chef-client on a target machine so that it can run as a chef-client node and communicate with the chef server.

From the workstation, you can bootstrap the node either by using the node’s root user, or a user with elevated privileges.

knife bootstrap chefclient.itzgeek.local -x root -P pass --sudo

**Important options**:

**-x**: The ssh username

**-P**: The ssh password

**-p**: The ssh port

**-N**: Set your chef-client node name. Leaving this out will usually make hostname being used as the chef-client node name.

**–sudo**: If the user name on the node will need to use sudo to perform administrative actions, then use this flag. Note: It will prompt you for sudo the sudo password.

Since I didn’t use -N in the command, the hostname will become chef node name.

**Output:**

Doing old-style registration with the validation key at /root/chef-repo/.chef/itzgeek-validator.pem...

Delete your validation key in order to use your user credentials instead

Connecting to chefclient.itzgeek.local

chefclient.itzgeek.local -----> Installing Chef Omnibus (-v 12)

chefclient.itzgeek.local downloading https://omnitruck-direct.chef.io/chef/install.sh

chefclient.itzgeek.local   to file /tmp/install.sh.2626/install.sh

chefclient.itzgeek.local trying curl...

chefclient.itzgeek.local el 7 x86\_64

chefclient.itzgeek.local Getting information for chef stable 12 for el...

.     .     . .     .     .chefclient.itzgeek.local [2016-11-12T19:24:36-05:00] WARN: Node chefclient.itzgeek.local has an empty run list.

chefclient.itzgeek.local Converging 0 resources

chefclient.itzgeek.local

chefclient.itzgeek.local Running handlers:

chefclient.itzgeek.local Running handlers complete

chefclient.itzgeek.local Chef Client finished, 0/0 resources updated in 05 seconds

Once the bootstrapping is complete, list down the nodes using the following command.

knife node list

**Output:**

chefclient.itzgeek.local

Get the client node details.

knife client show chefclient.itzgeek.local

**Output:**

admin:     false

chef\_type: client

name:      chefclient.itzgeek.local

validator: false

That’s All for now. We will soon meet again with another post on creating chef cookbooks.