Environment requirements for CHEF

CHEF client runs on nodes pulling (periodic polling) the information from the SERVER.

Centralized storage of infrastructure configurations.

KNIFE CLI is used to upload config changes to SERVER.

CHEF WORKSTATION

KNIFE is used to communicate with nodes using SSH.

Work station is a computer with local chef repo, configured with KNIFE CLI.

Chef is an automation tool that provides a way to define infrastructure as code. Infrastructure as code (IAC) simply means that managing infrastructure by writing code (Automating infrastructure) rather than using manual processes. It can also be termed as programmable infrastructure. Chef uses a pure-Ruby, domain-specific language (DSL) for writing system configurations.

Below are the types of automation done by Chef, irrespective of the size of infrastructure:

* Infrastructure configuration
* Application deployment
* Configurations are managed across your network

Following are the steps to install Chef:

1. Install Chef DK (Development Kit) on Chef Workstation
2. Setup a Chef Server
3. Create a Recipe or a Cookbook / download a Cookbook from Chef Supermarket in Workstation
4. Upload a Cookbook on the Chef Server
5. Connect A Node To The Chef Server
6. Deploy the Cookbook from the Server to the Node

As mentioned in CHEF environment we need these below four components:

1. CHEF Repository :

Here we store the every configuration component of the CHEF server. We can integrate this with the central CHEF server using the KNIFE CLI.

1. KNIFE :

As mentioned above we use KNIFE CLI to communicate with the central CHEF server from work station. Like uploading recipes, cookbooks to the server.

1. CHEF server :

CHEF server configures the nodes with the details that are applicable to the nodes like recipes, cookbooks, templates. Each node will have the client installed on them, so that it will pull the configurations on periodical basis.

1. CHEF Nodes :

For the communication between the nodes and the CHEF server we need to have a agent which is called CLIENT. Using the CHEF Client nodes can pull down the cookbooks that are applicable to them and configure themselves.

As infrastructure as code deserves tests too we use the test kitchen which provides the test harness to execute the infrastructure as code. Kitchen is used by all the chef-managed community cookbooks (available supermarket.chef.io) and is the integration testing tool of choice for cookbooks.

For this, .kitchen.yml file contains the information that kitchen CI needs to actually run the cookbook. So we need to mention the vagrant name, provisioner name, Platforms and suites (run lists).

Some of the commands we use to test the infrastructure code are:

$kitchen create

$kitchen converge

$kitchen setup

$kitchen verify

$kitchen destroy

There are two different ways to set up a Chef Development machine. You can either install Chef locally or launch a server in the cloud.

General requirements: Listed are commercially supported platforms and versions by CHEF ,

| **Platform** | **Architecture** | **Version** |
| --- | --- | --- |
| AIX | ppc64 | 6.1 (TL8 or higher, required), 7.1 (TL0 SP3 or higher, recommended), 7.2 |
| Amazon Linux |  | current version |
| CentOS | i386 (5.x only), x86\_64 | 5, 6, 7 |
| FreeBSD | amd64 | 10, 11 |
| macOS | x86\_64 | 10.10, 10.11, 10.12 |
| Oracle Enterprise Linux | i386 (5.x only), x86\_64 | 5.x, 6.x, 7.x |
| Red Hat Enterprise Linux | i386 (5.x only), x86\_64, s390x, ppc64le (7.x only), ppc64(7.x only) | 5.x, 6.x, 7.x |
| Solaris | sparc, x86 | 10 1/13("10U11"), 11.2, 11.3 |
| SUSE Enterprise Linux Server | x86\_64, s390x, ppc64le, ppc64 | 11 SP4, 12 SP1 |
| Ubuntu (LTS releases) | x86, x86\_64, ppc64le(14.04 only) | 14.04, 16.04 |
| Microsoft Windows | x86, x86\_64 | 2008r2, 2012, 2012r2, 2016, 7, 8, 8.1, 10 |

Listed are Community supported platforms and versions by CHEF ,

| **Platform** | **Architecture** | **Version** |
| --- | --- | --- |
| Arch Linux |  | Current version |
| Debian | i686, x86\_64 | 7,8,9 |
| Fedora |  | current non-EOL releases |
| Gentoo |  | Current version |
| OmniOS |  | stable and LTS releases |
| openSUSE |  | 42 |
| Scientific Linux | i386 (5.x only), x86-64 | |  | | --- | | 5.x, 6.x, 7.x | |  | |
| Ubuntu | x86, x86\_64 | non-LTS releases |

To find more on supported platforms and version for CHEF DK, CHEF server, CHEF automate and CHEF automate job runners, CHEF push job clients, CHEF push job servers, CHEF backend, CHEF compliance servers and CHEF manage can access through this link : <https://docs.chef.io/platforms.html>

The Chef server has the following prerequisites:

There are three configuration scenarios for the Chef server:

* [Standalone](https://docs.chef.io/install_server.html#standalone) (everything on a single machine)
* [High availability](https://docs.chef.io/install_server.html#high-availability) (machines configured for front-end and back-end, allowing for failover on the back-end and load-balancing on the front-end, as required)
* [Tiered](https://docs.chef.io/install_server.html#tiered-single-backend) (machines configured for front-end and back-end, with a single back-end and load-balancing on the front-end, as required)

[Standalone](https://docs.chef.io/install_server.html#standalone):

* An x86\_64 compatible system architecture; Red Hat Enterprise Linux and CentOS may require updates prior to installation
* A resolvable hostname that is specified using a FQDN or an IP address
* A connection to Network Time Protocol (NTP) to prevent clock drift
* If host-based firewalls (iptables, ufw, etc.) are being used, ensure that ports 80 and 443 are open..
* A local mail transfer agent that allows the Chef server to send email notifications
* Using cron and the /etc/cron.d directory for periodic maintenance tasks
* Disabling the Apache Qpid daemon on CentOS and Red Hat systems..
* Optional. A local user account under which services will run, a local user account for PostgreSQL, and a group account under which services will run. See [UIDs and GIDs](https://docs.chef.io/install_server_pre.html#uids-and-gids)for more information.

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

[High availability](https://docs.chef.io/install_server.html#high-availability):

Before installing the Chef server software, perform the following steps:

* Backend servers [should have a dedicated connection](http://www.drbd.org/users-guide/s-prepare-network.html). This is required for replication between the two servers.
* Backend servers will share a virtual IP address that must also be accessible from each frontend server. This virtual IP address is created and managed by the Chef server, but will also need to be added to the DNS so that all servers in the high availability configuration may access it.
* Persistent data on backend servers of the Chef server is primarily composed of cookbook files and directories. Separate disks should be dedicated entirely to storing this data prior to installing the Chef server.
* Load-balancing should be used with frontend machines, along with a DNS entry for the virtual IP address used for load balancing. This virtual IP address is added to the chef-server.rb file as the api\_fqdn.
* All required ports must be open.. All connections to and from the Chef server are accomplished via TCP. Refer to the operating system’s manual or your systems administrators for instructions on how to configure to ports, if necessary.
* The hostname for the Chef server must be an FQDN, including the domain suffix, and must be resolvable by the other backend and frontend servers..

For a high availability deployment:

General requirements

* Three backend servers; as many frontend servers as required
* 1 x GigE NIC interface (if on premises)

Frontend requirements

* 4 cores (physical or virtual)
* 4GB RAM
* 20 GB of free disk space (SSD if on premises, Premium Storage in Microsoft Azure, EBS-Optimized GP2 in AWS)

Backend requirements

* 2 cores (physical or virtual)
* 8GB RAM
* 50 GB/backend server (SSD if on premises, Premium Storage in Microsoft Azure, EBS-Optimized GP2 in AWS)

[Tiered](https://docs.chef.io/install_server.html#tiered-single-backend):

Before installing the Chef server software, perform the following steps:

* The backend server must be accessible from each frontend server. A virtual IP address is created and managed by the Chef server, but will also need to be added to the DNS so that all machines in the tiered configuration may access it.
* Persistent data on the backend Chef server is primarily composed of cookbook files and directories. Separate disks should be dedicated entirely to storing this data prior to installing the Chef server.
* Load-balancing should be used with frontend servers, along with a DNS entry for the virtual IP address used for load balancing. This virtual IP address is added to the chef-server.rb file as the api\_fqdn.
* All required ports must be open..All connections to and from the Chef server are accomplished via TCP. Refer to the operating system’s manual or your systems administrators for instructions on how to configure to ports, if necessary.
* The hostname for the Chef server must be an FQDN, including the domain suffix, and must be resolvable by the backend and frontend servers..
* Chef-server-ctl reconfigure will not bind the backend\_vip to the backend server. The easiest thing to do is just define backend\_vip as the already configured main IP address of the backend system. If you need to use an additional address, it will need to be configured and bound on the system before chef-server-ctl reconfigure is run.

For a tiered deployment, your backend server should support the following hardware requirements:

* 64-bit architecture
* 8 total cores (physical or virtual)
* 16GB RAM
* Fast, redundant storage (SSD/RAID-based solution)
  + 50 GB/backend server (SSD if on premises, Premium Storage in Microsoft Azure, EBS-Optimized GP2 in AWS)
* 1 GigE NIC interface
* A back-end server; all other systems will be front-end servers.

Before installing the Chef server, ensure that each machine has the following installed and configured properly:

* Hostnames — Ensure that all systems have properly configured hostnames. The hostname for the Chef server must be a FQDN, including the domain suffix, and must be resolvable. See [Hostnames, FQDNs](https://docs.chef.io/install_server_pre.html#hostnames) for more information
* FQDNs — Ensure that all systems have a resolvable FQDN
* NTP — Ensure that every server is connected to NTP; the Chef server is sensitive to clock drift
* Mail Relay — The Chef server uses email to send notifications for various events; a local mail transfer agent should be installed and available to the Chef server
* cron — Periodic maintenance tasks are performed using cron
* git — git must be installed so that various internal services can confirm revisions
* libfreetype and libpng — These libraries are required
* Apache Qpid — This daemon must be disabled on CentOS and Red Hat systems
* Required users — If the environment in which the Chef server will run has restrictions on the creation of local user and group accounts, ensure that the correct users and groups exist before reconfiguring
* Firewalls and ports — If host-based firewalls (iptables, ufw, etc.) are being used, ensure that ports 80 and 443 are open. These ports are used by the nginxservice
* Hostname — The hostname for the Chef server must be a FQDN, including the domain suffix, and must be resolvable. See [Hostnames, FQDNs](https://docs.chef.io/install_server_pre.html#hostnames) for more information

SETUP VM’S:

Through the Azure Marketplace and the [Azure portal](https://portal.azure.com/), virtual machines can be bootstrapped and ready to run Chef Automate, Chef Compliance and Chef client.

Through the Azure portal, you can provision a virtual machine with chef-client running as a background service. Once provisioned, these virtual machines are ready to be managed by a Chef server.

Start the virtual machine creation workflow, and workflow, and then bootstrap virtual machines with Chef using the following steps:

1. Sign in to the [Azure portal](https://portal.azure.com/) and authenticate using your Microsoft Azure account credentials.
2. Choose Virtual Machines in the left pane of the portal.
3. Click the Add option at the top of the blade.
4. Select either Windows Server or Ubuntu Server in the Recommended category.

The Chef extension on the Azure portal may be used on the following platforms:

* Windows Server 2008 R2 SP1, 2012, 2012 R2, 2016
* Ubuntu 12.04 LTS, 14.04 LTS, 16.04 LTS, 16.10
* CentOS 6.5+
* RHEL 6+
* Debian 7, 8

1. In Step 3 on the portal UI, open the Extensions blade and click Add extension.
2. Depending on the OS you selected earlier, select either Windows Chef Extension or Linux Chef Extension and then Create.
3. Using the chef-repo/.chef/knife.rb file you downloaded during your Chef server setup, enter values for the Chef server URL and the validation client name. You can also use this file to help you find the location of your validation key.
4. Browse on your local machine and find your validation key (chef-repo/.chef/<orgname>-validator.pem).
5. Upload it through the portal in the Validation Key field.
6. For **Client Configuration File**, browse to the chef-repo/.chef/knife.rb file and upload it through your web browser.

After the process is complete, the virtual machine will be registered with the Chef server and it will have been provisioned with the configuration (applications, services, etc.) from the specified run-list. The Chef server can now be used to perform all ongoing management of the virtual machine node.

CERTIFICATES:

SSL: Initial configuration of the Chef server is done automatically using a self-signed certificate to create the certificate and private key files.

PRIVATE CERTIFICATE AUTHORITY: If an organization is using an internal certificate authority, then the root certificate will not appear in any cacerts.pem file that ships by default with operating systems and web browsers. Because of this, no currently deployed system will be able to verify certificates that are issued in this manner. To allow other systems to trust certificates from an internal certificate authority, this root certificate will need to be configured so that other systems can follow the chain of authority back to the root certificate.

REGENERATE CERTIFICATES: SSL certificates should be regenerated periodically. This is an important part of protecting the Chef server from vulnerabilities and helps to prevent the information stored on the Chef server from being compromised.

Some More..

* CHEF CLIENT can be used to manage the machines that run on :

Windows (version: 2008R2, 2012, 2012R2; Architecture: X86\_64)

* Min RAM size of chef client to run should be 512 MB
* Each node and work station must have the access to the CHEF server via HTTPS.
* Ruby versions should be 1.9.1 or 1.9.2 with SSL bindings is required
* Two KNIFE plugins dedicated to Microsoft platforms are available,
  1. KNIFE AZURE used to manage virtual instances in Microsoft AZURE
  2. KNIFE WINDOWS used to interact and manage physical nodes that are running such as desktops or servers.
* A Microsoft installer packages are available to install the chef client on the Microsoft windows machines.
* The knife azure sub-command must use a management certificate for secure communication with Microsoft Azure. The management certificate is required for secure communication with the Microsoft Azure platform via the REST APIs.
* Microsoft Windows sub-command requires WinRM to be installed, and then configured correctly, including ensuring the correct ports are open.

This is high-level architecture of how we can automate the entire infrastructure using the CHEF as infrastructure as code.

Useful links

<https://downloads.chef.io/chef-server/>.

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

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

<https://docs.chef.io/server_components.html#capacity-planning>

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

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