# IaC - Infrastructure as Code

## Introduction

Infrastructure as Code (IaC) is a paradigm in which infrastructure provisioning and management are done using code and automation, treating infrastructure elements such as servers, networks, and configurations as programmable entities. Instead of manually setting up and configuring servers and resources, IaC involves writing code to define and deploy infrastructure.

## Concepts of Infrastructure as Code:

1. **Declarative Configuration:** IaC uses declarative code to define the desired state of your infrastructure. You describe what you want the infrastructure to look like rather than specifying the steps to get there.
2. **Version Control:** IaC code, just like software code, can be stored in version control systems (e.g., Git). This allows you to track changes, collaborate with others, and revert to previous configurations if needed.
3. **Automation:** IaC automates the provisioning, configuration, and management of infrastructure. This reduces human error, ensures consistency, and saves time compared to manual setups.
4. **Idempotency:** IaC tools are designed to be idempotent, meaning you can run the same code multiple times without causing unintended side effects. The system will only make necessary changes to match the desired state.
5. **Reproducibility:** With IaC, you can easily recreate identical infrastructure environments in different locations, such as development, testing, and production.

## Benefits of Infrastructure as Code:

1. **Consistency:** IaC ensures that your infrastructure is consistent across environments. This reduces the "works on my machine" problem and promotes reliable software deployments.
2. **Speed and Efficiency:** Automating infrastructure provisioning and configuration significantly speeds up the process. New environments can be created and updated quickly, reducing deployment times.
3. **Scalability:** IaC makes it easier to scale your infrastructure horizontally or vertically by defining resources and configurations as code. You can quickly spin up new servers or instances as needed.
4. **Reduced Risk:** Manual configurations are prone to errors and inconsistencies. IaC reduces human error and minimizes the risk of misconfigurations that can lead to security vulnerabilities or downtime.
5. **Documentation:** IaC code serves as living documentation. It's clear, explicit, and describes how your infrastructure is set up, making it easier for new team members to understand and contribute.
6. **Collaboration:** IaC promotes collaboration between development, operations, and other teams. Infrastructure changes are code changes, which can be reviewed, tested, and approved like any other code.
7. **Versioning and Rollbacks:** With version control, you can roll back to a previous state of infrastructure if an issue arises, enabling quick recovery and reducing downtime.
8. **Compliance and Auditing:** IaC helps maintain compliance by providing a clear audit trail of changes made to infrastructure configurations over time.
9. **Cost Optimization:** IaC allows you to define and manage resources more precisely, reducing the likelihood of over-provisioning and saving costs.
10. **Easier Disaster Recovery:** IaC simplifies disaster recovery by allowing you to recreate entire environments quickly using code.

Overall, IaC streamlines the process of managing infrastructure, improves collaboration, reduces errors, enhances scalability, and increases the speed at which software is deployed. It's a fundamental practice in modern DevOps and software development workflows.Top of Form

# Chef:

* Established in 2008, Chef Software came into existence. Our initial offering, known as Chef (currently referred to as Chef Infra), comprises a collection of utilities designed to streamline the process of configuring your server infrastructure, whether it's based in the cloud or on-premises.
* Chef possesses the capability to seamlessly integrate with cloud-centric platforms like Microsoft Azure, Google Cloud, and Amazon Web Services (AWS), enabling the automatic provisioning and configuration of computing resources hosted in these cloud environments.
* To illustrate, consider a scenario where a sizable retail enterprise intends to set up and arrange 50 servers for an impending sales event. They could leverage Chef Infra to automate the deployment of this infrastructure.
* In October 2020, Progress Software acquired Chef Software, which now functions as a distinct business division within Progress Software. Chef Software has consistently maintained its prominence in the realms of DevOps and DevSecOps, retaining its leadership position.

## Components of Chef:

1. **Chef Server:**

This is the central hub that stores configuration data, known as "cookbooks," and manages the communication between nodes (servers) and the Chef client. It allows you to store and manage your infrastructure's configuration and policies.

1. **Chef Client:**

The Chef client runs on each node that you want to manage with Chef. It communicates with the Chef server to retrieve configuration information and apply it to the node. The client ensures that the node's state matches the desired configuration described in the cookbooks.

1. **Cookbooks:**

Cookbooks are the fundamental units of configuration in Chef. They consist of recipes, attributes, templates, and other resources. A cookbook defines how a specific piece of software or configuration should be installed, configured, and managed on a node.

User upload a set of Chef cookbooks to the Chef Infra server. You can think of Chef cookbooks as a set of configuration files, called recipes, that will instantiate, configure, and maintain your infrastructure nodes in an automated fashion. (A node is a physical or virtual machine.)

The Chef Infra server in turn loads those cookbooks to the correct nodes. Chef Infra can do this on a massive scale thus eliminating the tedious task of manually configuring your infrastructure nodes individually.

1. **Recipes:**

Recipes are collections of resources and instructions that define the desired state of a particular component on a node. They are written in a domain-specific language (DSL) and describe the sequence of actions needed to configure a specific part of the system.

1. **Nodes:**

Nodes are the servers or systems that are managed using Chef. Each node has a corresponding client that communicates with the Chef server to receive instructions and configuration information.

## Workflow of using Chef:

The typical workflow of using Chef involves the following steps:

1. **Write Cookbooks:**

Create or customize cookbooks to define the desired configuration of your infrastructure. This includes specifying how software should be installed, configurations applied, and services managed.

1. **Upload Cookbooks:**

Upload your cookbooks to the Chef server, making them available to the Chef clients.

1. **Node Registration:** Register nodes (servers) with the Chef server by installing the Chef client on them.
2. **Apply Configurations:** The Chef client on each node regularly checks with the Chef server for configuration updates. It then applies the required changes to ensure that the node's actual state matches the desired state described in the cookbooks.
3. **Continuous Management:** As your infrastructure evolves, you can update and modify your cookbooks, and the changes will be propagated to the nodes, ensuring consistency and desired configuration.

## Installation Chef Workstation:

### Supported Platforms

|  |  |
| --- | --- |
| **Platform** | **Version** |
| Amazon Linux | 2 |
| Apple macOS | 10.15, 11, 12 |
| Windows | 10, 11, Server 2012, Server 2012 R2, Server 2016, Server 2019, Server 2022 |
| Red Hat Enterprise Linux / CentOS | 7.x, 8.x, 9.x |
| Ubuntu | 18.04, 20.04, 22.04 |
| Debian | 10, 11 |

### System Requirements

* RAM: 4GB
* Disk: 8GB
* Additional memory and storage space may be necessary to take advantage of Chef Workstation tools such as Test Kitchen which creates and manages virtualized test environments.

Additional Chef Workstation App Requirements:

* On Linux you must have a graphical window manager running with support for system tray icons. For some distributions you may also need to install additional libraries. After you install the Chef Workstation package from the terminal, the post-install message will tell you which, if any, additional libraries are required to run the Chef Workstation App.

## Installation

### Linux (Debian/Ubuntu)

* Visit the [Chef Downloads](https://www.chef.io/downloads) page and download the appropriate package for your distribution:

*$ wget https://packages.chef.io/files/stable/chef-workstation/21.10.640/ubuntu/20.04/chef-workstation\_21.10.640-1\_amd64.deb*

* Use your distribution’s package manager to install Chef Workstation:

$ sudo dpkg -i chef-workstation\_21.10.640-1\_amd64.deb

A screenshot of a computer

Description automatically generated

* Verify

$ chef -v

A screen shot of a computer

Description automatically generated

* To Uninstall:

$ sudo dpkg -P chef-workstation

A diagram of a chef server

Description automatically generated

A computer with words on it

Description automatically generated

A table with text and images

Description automatically generated with medium confidence

Source: <https://www.9tut.com/chef-tutorial>

**Workstation**

* This is where the admin sits.
* Here, the code is made for setting up and handling the infrastructure.
* The code is called Recipe.
* A cookbook is a collection of recipes, along with any necessary supporting files such as templates, attribute files, and libraries.
* Using Knife, a CLI tool, you can put the cookbook on the server.

**Server**

* The server acts as an intermediary connecting the workstation and the nodes.
* Cookbooks are stored on the server.
* It supplies the necessary tools to manage node configuration.
* The server can be hosted either nearby or at a distant location.

**Node**

* Nodes are the machines that need configuration adjustments.
* Ohai retrieves the current condition of the node where it resides.
* The Chef client handles communication with the server.
* Distinct nodes might have diverse configuration needs.

Step1: System admin creates Recipes. Recipes are created at workstations and will be send to the server

Step2. Knife tool will be used to send the above recipes to the server.

$ knife upload simpli-db

Flavours of Chef

1. Chef Solo – No separate server. Cookbooks are located at the local site itself. Will be used when you have a single node.

Local Machine (Everything here)

1. Hosted Chef – Chef server is located at Cloud. No need to setup a server yourself.

Work Station 🡪 Server 🡨🡪 Node1

1. Chef Client/Server – Traditional Chef architecture where a remote server is hosted that communicates between workstation and nodes

Work Station 🡪 Server 🡨🡪 Node1/2/3/…

1. Private Chef (aks Enterprise Chef) – Enterprise version of Chef where the workstation+node is hosted within the enterprise infrastructure. Can add extra layers of the security

Work Station 🡪 Server 🡨🡪 Node1/2/3/…

Setup a NGINX in the local machine:

# Setup Chef in the VM

## Install in Ubuntu

$ wget https://packages.chef.io/files/stable/chef-workstation/21.10.640/ubuntu/20.04/chef-workstation\_21.10.640-1\_amd64.deb

$ sudo dpkg -i chef-workstation\_21.10.640-1\_amd64.deb

A black screen with white text

Description automatically generated

## Check the version:

$ chef -v

A black rectangle with white text

Description automatically generated

# Setup Nginx in a local machine using Chef

## Create a folder – chef-repo/cookbooks

$ mkdir chef-repo

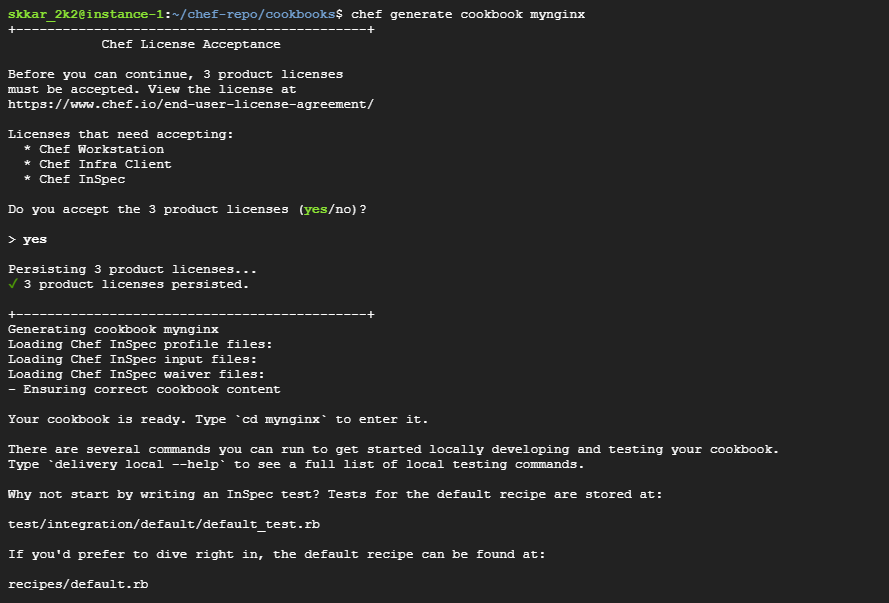
$ cd chef-repo

$ mkdir cookbooks

$ cd cookbooks

## Create cookbook with name – mynginx

$ chef generate cookbook mynginx



## Edit default.rb and add contents for ***nginx***

$ sudo nano mynginx/recipes/default.rb

A screenshot of a computer

Description automatically generated

## Generate cookbook file – index.html *(should be inside mynginx folder)*

$ cd mynginx/

$ chef generate file index.html

A screen shot of a computer

Description automatically generated

## Add contents to the index.html

$ sudo nano files/index.html

A screen shot of a computer code

Description automatically generated

## Generate cookbook template – index.html *(should be inside mynginx folder)*

$ chef generate template nginx.conf

A screenshot of a computer

Description automatically generated

## Add template for nginx to ./templates/nginx.conf.erb

A computer screen shot of a black screen

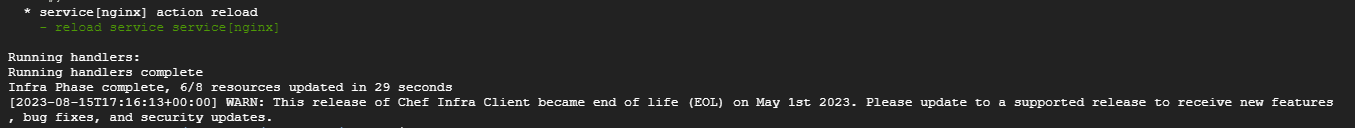
Description automatically generated

## Run chef command

$ sudo chef-client -z --runlist "mynginx"

A screenshot of a computer program

Description automatically generated



# Access the public URL:

A screenshot of a computer

Description automatically generated