CHEF CONFIGURATION MANAGEMENT TOOL

BY:

ALI MEHRANI

ALI RANJBARI

AMIRREZA GHAHREMANI

TOPICS

- Introduction
- What is Chef?
- Configuration management in Chef
- Why Chef?
- Architecture and components of Chef

INTRODUCTION

- Permanent demands for deploying software services in organizations.
- Deployment is a tedious and sensitive task, especially for large organizations.
- Chef enables infrastructure management.

We’ll start with an example.

AN EXAMPLE

Consider a system administrator maintaining multiple systems in a network.

- System failures can occur frequently.
- A single system admin can manage individual system failures.
- Multiple simultaneous issues are hard to handle.

AN EXAMPLE

- Risk of severe network disruptions.
- Can be resolved by synchronizing environments and automating processes.
- Chef makes this easy.

WHAT IS CHEF?

- An open-source **configuration management tool**
- Uses **Ruby** to develop essential building blocks like recipes and cookbooks.
- Converts infrastructure to code as an automation tool.

WHAT IS CHEF?

- writing code instead of using the manual process.
- Manage and config multiple systems.
- Continuous test and deployment.

WHAT IS CHEF?

Chef supports various cloud computing platforms including:

- AWS
- Google Cloud Platform
- OpenStack
- SoftLayer (IBM cloud)

CONFIGURATION MANAGEMENT IN CHEF

Consider yourself as the company's system administrator

- Have to update the software on numerous systems.
- Better not to do that manually!
- Getting back to the previous state if system is crashed during update?

Savior configuration management helps here.

CONFIGURATION MANAGEMENT IN CHEF

Configuration Management:

- keeps all the info about the system's hardware, software, and infrastructure.
- Helps managing complex infrastructures.
- Automates time-consuming tasks.

CONFIGURATION MANAGEMENT IN CHEF

- Enables automated deployments to restore services of a server
- Automatically handles changes in the infrastructure
- A big help in unwanted disasters!

WHY CHEF?

A DevOps tool that helps in:

- Automating the process of infrastructure provisioning.
- Speeding up the deployment process and software delivery.
- Simplifying the **configuration task.**

WHY CHEF?

- Managing the system's server.
- Configure, deploy and manage the entire application infrastructure.

Next, we will mention some of its popular features.

WHY CHEF?

- Chef can easily manage a large number of servers with fewer employees.
- Allows continuity in the deployment process from building to the end.
- Can be managed using different Operating Systems like Linux, and Windows.
- Can be integrated with several major cloud service providers.

WHY CHEF?

- A single chef-server may be used as a center for all policy deployment.
- Chef is a reliable and stable tool for large deployments.
- Helps in managing the risk at all stages of deployment.

ARCHITECTURE AND COMPONENTS OF CHEF

The architecture of Chef can be divided into three components:

1. **Workstation**
2. **Chef Server**
3. **Nodes**

WORKSTATION

- Simply, the admin's system.
- Makes possible interactions with **Chef-nodes** and **Chef-server.**
- Cookbooks are created, tested, and deployed here.
- Other Chef users cookbooks can also be downloaded using the workstation.

WORKSTATION COMPONENTS

- Development Kit
- Chef-Repo
- Knife
- Test Kitchen

WORKSTATION COMPONENTS

- **Development Kit:**
  - Contains numerous packages
  - Contained packages are required for using Chef.

WORKSTATION COMPONENTS

- **Chef-Repo**
  - A directory of the workstation where all the cookbooks are maintained.
  - Can be created using this command: **“chef to generate repo repo-name”.**

WORKSTATION COMPONENTS

- **Knife**
  - Provides a communication interface.
  - Communication between a local chef-repo and chef server
  - Helps managing Nodes, Cookbooks, and Recipes.

WORKSTATION COMPONENTS

- **Test Kitchen**
  - Provides a development environment to the workstation
  - Enables it to create and test workbooks before they are distributed.

CHEF SERVER

- Chef Server is the center of the workstation and the nodes.
- All the cookbooks, recipes, and metadata are stored here.
- The workstation sends the cookbook to the server using a **knife**.

CHEF SERVER

- The nodes communicate with the server using the Chef client.
- Any changes made to the infrastructure code, must be passed to the Chef Server.
- Chef Server applies the changes to the nodes.

NODES

- Nodes refer to the machines that are managed or configured by the Chef-Server.
- Might be virtual servers, network devices, or storage devices.
- Chef client executes the steps needed to bring the node into the required state

NODES

- Steps are defined by a cookbook.
- Chef client makes it possible for the nodes to stay up to.
- Chef client runs individually on each node to configure them.

A CODE SAMPLE

- A simple code sample in chef for the following concern:
  - Installing Nginx on the server.
  - Ensuring Nginx is enabled to start at boot.
  - Template file for defining Nginx server configuration.

A CODE SAMPLE

webserver.rb

```ruby
# installs and configures nginx on the server

package 'nginx' do
  action :install
end

service 'nginx' do
  action [:enable, :start] #ensures that nginx is enabled to start at boot
end

template '/etc/nginx/sites-available/default' do
  source 'default.erb'
  owner 'root'
  group 'root'
  mode '0644'
  notifies :reload, 'service[nginx]'
end
```

A CODE SAMPLE

default.erb

```
# template file that defines the Nginx server configuration.

server {
    listen 80 default_server;
    listen [::]:80 default_server;

    server_name _;

    root /var/www/html;
    index index.html index.htm;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

THANK YOU

FOR YOUR ATTENTION!