Chef Introduction

Infrastructure Automation

- *Infrastructure automation* is the process of scripting environments — from installing an operating system,

  to installing and configuring servers on instances,

  to configuring how the instances and

  software communicate with one another,

  and much more.

By scripting environments, you can apply the same configuration to a single node or to thousands.

Infrastructure Automation

- Infrastructure automation also goes by other names:
  - configuration management,
  - IT management,
  - provisioning, scripted infrastructures,
  - system configuration management, and
  - many other overlapping terms.
- The point is the same: you are describing your infrastructure and its configuration as a script or set of scripts so that environments can be replicated in a much less error-prone manner.
- Infrastructure automation brings agility to both development and operations because any authorized team member can modify the scripts while applying good *development* practices — such as automated testing and versioning — to your infrastructure.

What is 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.

Chef Components

Chef Workstation

- 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),
    - Knife (Command line for interacting with the Chef server)
    - chef (for interacting with your local chef-repo),
    - core Chef resources (for building recipes) and
    - InSpec (for building security and compliance checks into your workflow).

Workstations

- A workstation is a computer that is configured to run various Chef command-line tools that synchronize with a chef-repo, author cookbooks, interact with the Chef server, and interact with nodes.
- The workstation is the location from which most users do most of their work, including:
  - Developing cookbooks and recipes (and authoring them using Ruby syntax and patterns)
  - Keeping the chef-repo synchronized with version source control
  - Using command-line tools
  - Configuring organizational policy, including defining roles and environments and ensuring that critical data is stored in data bags
  - Interacting with nodes, as (or when) required, such as performing a bootstrap operation

components of workstations

Chef development kit

- The Chef development kit is a package that contains everything that is needed to start using Chef:
  - chef-client
  - chef
  - Ohai
  - chef-zero
  - Testing tools like Kitchen, ChefSpec, Cookstyle, and Foodcritic
  - Chef provisioning
  - Everything else needed to author cookbooks and upload them to the Chef server

command-line tools

- Use the chef command-line tool 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
- Use the knife command-line tool to interact with nodes or work with objects on the Chef server

Chef-repo

- The chef-repo is the repository structure in which cookbooks are authored, tested, and maintained:
- Cookbooks contain recipes, attributes, custom resources, libraries, definitions, files, templates, tests, and metadata
- The chef-repo should be synchronized with a version control system (such as git), and then managed as if it were source code
- The directory structure within the chef-repo varies. Some organizations prefer to keep all of their cookbooks in a single chef-repo, while other organizations prefer to use a chef-repo for every cookbook.

Kitchen

- Use Kitchen to automatically test cookbook data across any combination of platforms and test suites:
  - Defined in a .kitchen.yml file
  - Uses a driver plugin architecture
  - Supports cookbook testing across many cloud providers and virtualization technologies
  - Supports all common testing frameworks that are used by the Ruby community
  - Uses a comprehensive set of base images provided by Bento

ChefSpec

- Use ChefSpec to simulate the convergence of resources on a node:
  - Runs the chef-client on a local machine
  - Is an extension of RSpec, a behavior-driven development (BDD) framework for Ruby
  - Is the fastest way to test resources and recipes

Chef Node

Chef Node

- 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.

Physical Node

- A physical node is typically a server or a virtual machine, but it can be any active device attached to a network that is capable of sending, receiving, and forwarding information over a communications channel.
- In other words, a physical node is any active device attached to a network that can run a chef-client and also allow that chef-client to communicate with a Chef server.

Containers

- Containers are an approach to virtualization that allows a single operating system to host many working configurations,
- where each working configuration—a container—is assigned a single responsibility that is isolated from all other responsibilities.
- Containers are popular as a way to manage distributed and scalable applications and services.

Cloud Node

- A cloud-based node is hosted in an external cloud-based service, such as
  - Amazon Web Services (AWS),
  - OpenStack, Rackspace,
  - Google Compute Engine, or
  - Microsoft Azure.
- Plugins are available for knife that provide support for external cloud-based services.
- knife can use these plugins to create instances on cloud-based services.
- Once created, the chef-client can be used to deploy, configure, and maintain those instances.

Networking Device Node

- A network node is any networking device—a switch, a router—that is being managed by a chef-client,
- such as networking devices by Juniper Networks, Arista, Cisco, and F5.
- Use Chef to automate common network configurations, such as physical and logical Ethernet link properties and VLANs, on these devices.

Components of Chef Node

Chef Client

- A chef-client is an agent that runs locally on every node that is under management by Chef.
- When a chef-client is run, it will perform all of the steps that are required to bring the node into the expected state, including:
  - Registering and authenticating the node with the Chef server
  - Building the node object
  - Synchronizing cookbooks
  - Compiling the resource collection by loading each of the required cookbooks, including recipes, attributes, and all other dependencies
  - Taking the appropriate and required actions to configure the node
  - Looking for exceptions and notifications, handling each as required
- RSA public key-pairs are used to authenticate the chef-client with the Chef server every time a chef-client needs access to data that is stored on the Chef server.

Ohai

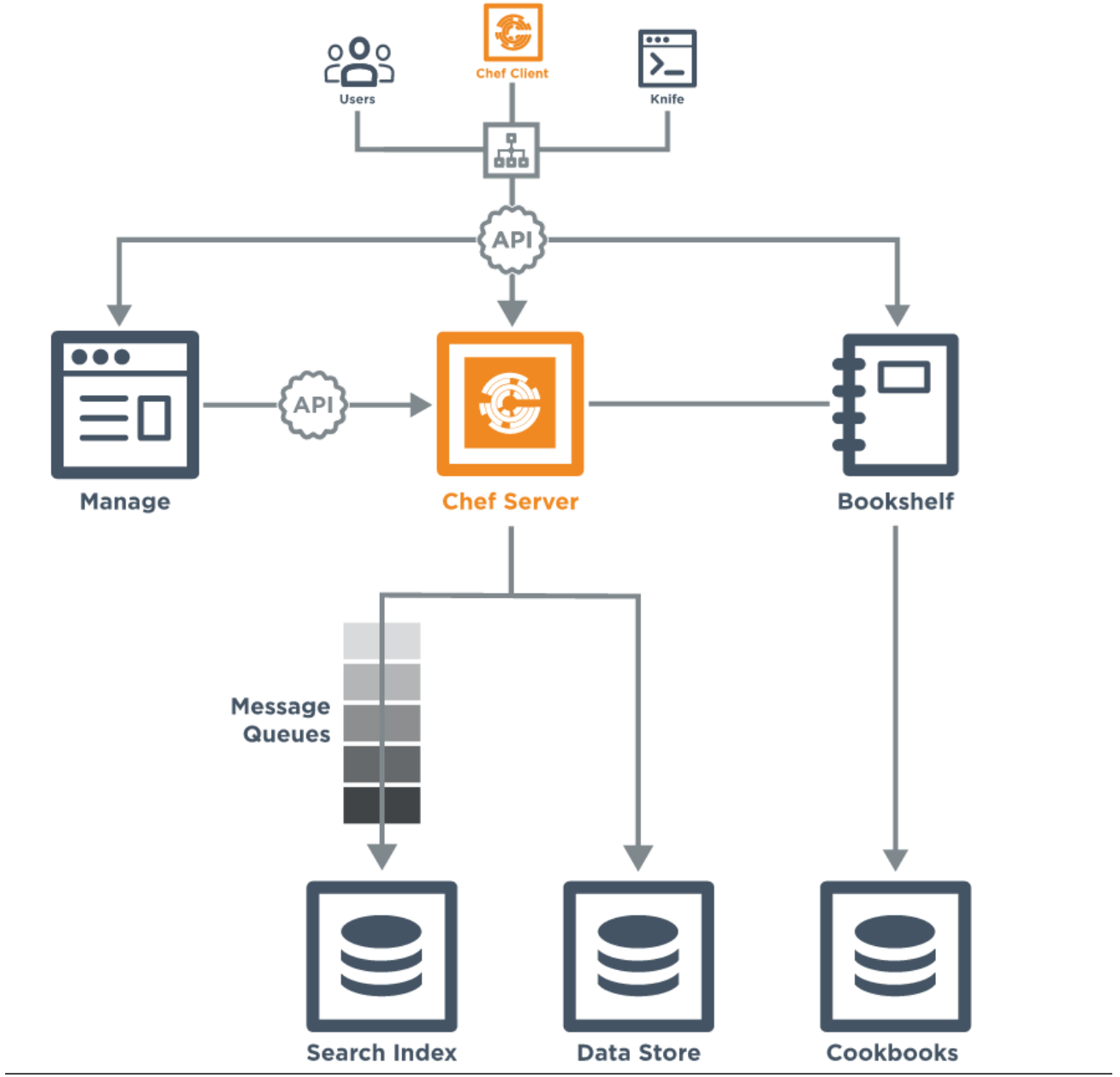
- Ohai is a tool that is used to detect attributes on a node, and then provide these attributes to the chef-client at the start of every chef-client run.
- Ohai is required by the chef-client and must be present on a node.
- The types of attributes Ohai collects include (but are not limited to):
  - Platform details
  - Network usage
  - Memory usage
  - CPU data
  - Kernel data
  - Host names
  - Fully qualified domain names
  - Other configuration details

Chef Server

Chef Server

- Use the Chef server as your foundation to create and manage flexible, dynamic infrastructure whether you manage 50 or 500,000 nodes, across multiple datacenters, public and private clouds, and in heterogeneous environments.
- 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.
- The chef-client then does as much of the configuration work as possible on the nodes themselves (and not on the Chef server).
- This scalable approach distributes the configuration effort throughout the organization.

Chef Server Architecture



Chef Manage

- chef-server-webui is a Ruby on Rails 3.0 application that hosts the web interface for the Chef server.
- The Chef management console uses the Chef server API for all communication to the Chef server.

Chef Server

- chef is a complete written on Erlang.
- It exposes the API t communicate with client and Web UI
- Which Manages Bookshelf, Data Store, Search Indexes.

Bookshelf

- Bookshelf is used to store cookbook content—files, templates, and so on—that have been uploaded to the Chef server as part of a cookbook version.
- Cookbook content is stored by content checksum.
- If two different cookbooks or different versions of the same cookbook include the same file or template, Bookshelf will store that file only once. T
- he cookbook content managed by Bookshelf is stored in flat files and is separated from the Chef server and search index repositories.
- All cookbooks are stored in a dedicated repository.

Message Queues

- Messages are sent to the search index using the following components:
- RabbitMQ is used as the message queue for the Chef server. All items that will be added to the search index repository are first added to a queue.
- chef-expander is used to pull messages from the RabbitMQ queue, process them into the required format, and then post them to chef-solr for indexing.
- chef-solr wraps Apache Solr and exposes its REST API for indexing and search.
- All messages are added to a dedicated search index repository.

PostgreSQL

- PostgreSQL is the data storage repository for the Chef server.
- This represents the independently configured set of servers that are running PostgreSQL and are configured to act as the data store for the Chef server

Cookbooks

- A cookbook is the fundamental unit of configuration and policy distribution.
- A cookbook defines a scenario and contains everything that is required to support that scenario:
  - Recipes that specify the resources to use and the order in which they are to be applied
  - Attribute values
  - File distributions
  - Templates
  - Extensions to Chef, such as custom resources and libraries

Cook Book

- The chef-client uses Ruby as its reference language for creating cookbooks and defining recipes, with an extended DSL for specific resources.
- The chef-client provides a reasonable set of resources, enough to support many of the most common infrastructure automation scenarios;
- however, this DSL can also be extended when additional resources and capabilities are required.

Recipie

- A recipe is the most fundamental configuration element within the organization.
- Is authored using Ruby, which is a programming language designed to read and behave in a predictable manner
- Is mostly a collection of resources, defined using patterns having
  - resource names,
  - attribute-value pairs, and
  - actions);
- helper code is added around this using Ruby, when needed

Recipie

- Must define everything that is required to configure part of a system
- Must be stored in a cookbook
- May be included in a recipe
- May use the results of a search query and read the contents of a data bag
- May have a dependency on one (or more) recipes
- May tag a node to facilitate the creation of arbitrary groupings
- Must be added to a run-list before it can be used by the chef-client
- Is always executed in the same order as listed in a run-list

Recipie

- The chef-client will run a recipe only when asked.
- When the chef-client runs the same recipe more than once, the results will be the same system state each time.
- When a recipe is run against a system, but nothing has changed on either the system or in the recipe, the chef-client won’t change anything.

Attribute

- An attribute can be defined in a cookbook (or a recipe) and then used to override the default settings on a node.
- When a cookbook is loaded during a chef-client run, these attributes are compared to the attributes that are already present on the node.
- Attributes that are defined in attribute files are first loaded according to cookbook order.
- For each cookbook, attributes in the default.rb file are loaded first
- Then additional attribute files (if present) are loaded in lexical sort order
- When the cookbook attributes take precedence over the default attributes, the chef-client will apply those new settings and values during the chef-client run on the node.

Resource

- Piece of the system and its desired state
- A resource instructs the chef-client to complete various tasks like installing packages, running Ruby code, or accessing directories and file systems.
- The chef-client includes built-in resources that cover many common scenarios.

Resources - Package

Package that should be installed

```
package "mysql-server" do
  action :install
end
```

Resources - Service

Service that should be running and restarted on reboot

```
service "iptables" do
  action [ :start, :enable ]
end
```

Resources - Service

File that should be generated

```
file "/etc/motd" do
  content "Property of Chef Software"
end
```

Resources - Cron

Cron job that should be configured

```
cron "restart webserver" do
  hour '2'
  minute '0'
  command 'service httpd restart'
end
```

Resources - User

User that should be managed

```
user "nginx" do
  comment "Nginx user <nginx@example.com>"
  uid 500
  gid 500
  supports :manage_home => true
end
```

Resources – Registry Key

Registry key that should be created

```
registry_key
"HKEY_LOCAL_MACHINE\\SOFTWARE\\Mic
rosoft\\Windows\\CurrentVersion\\P
olicies\\System" do
  values [{
    :name => "EnableLUA",
    :type => :dword,
    :data => 0
    }]
  action :create
end
```

Resources

- Piece of the system and its desired state

- http://docs.chef.io/chef/resources.html

chef-apply

- chef-apply is an executable program that allows you to work with resources
- Is included as part of the ChefDK
- A great way to explore resources
- NOT how you'll eventually use Chef in production

Install telnet

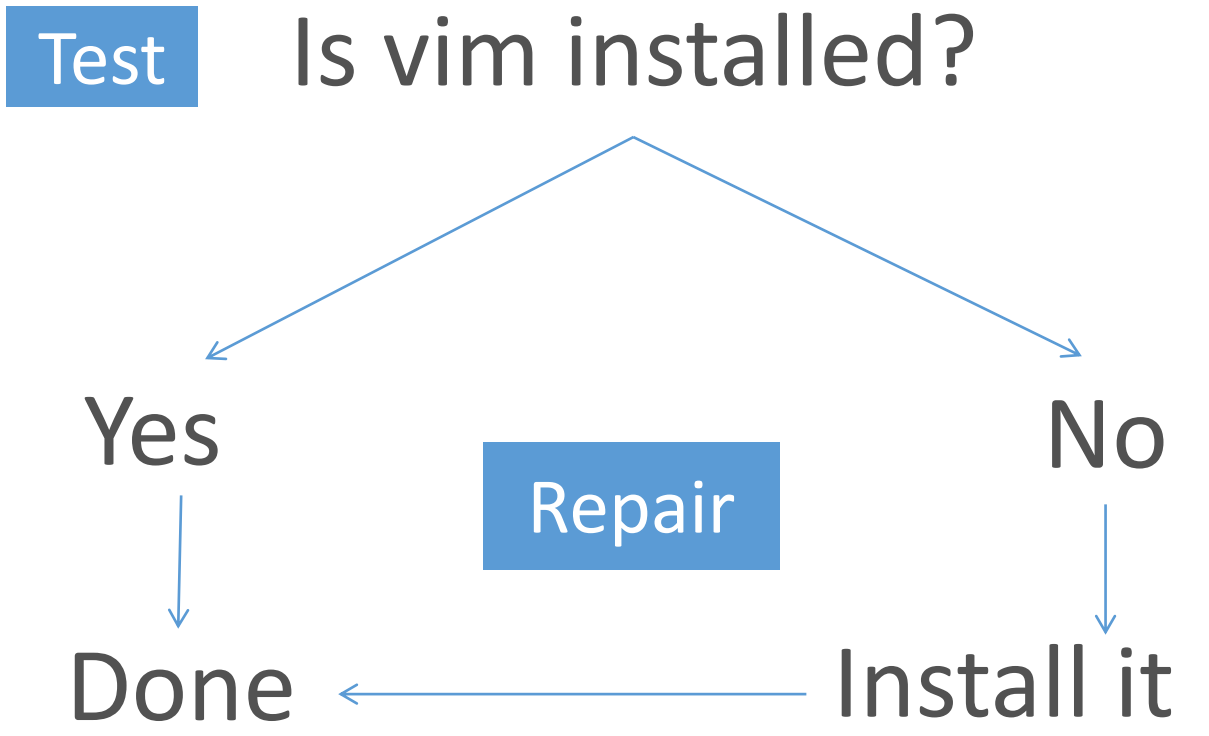
```
$ sudo chef-apply -e "package 'vim'"
```

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * yum_package[vim] action install
    - install version xxxxxx of package vim
```

Test and Repair

Resources follow a **test** and **repair** model

```
package "vim"
```



Templates

- A cookbook template is an Embedded Ruby (ERB) template that is used to dynamically generate static text files
- Templates may contain Ruby expressions and statements, and are a great way to manage configuration files.

```
<%= "my name is #{$ruby}" %>
```

- Use the **template** resource to add cookbook templates to recipes;
- place the corresponding Embedded Ruby (ERB) template file in a cookbook's /template directory
- To use a template, two things must happen:
  - A template resource must be added to a recipe
  - An Embedded Ruby (ERB) template must be added to a cookbook

Cookbook File

- A file distribution is a specific type of resource that tells a cookbook how to distribute files, including by node, by platform, or by file version.
- Use the cookbook_file resource to manage files that are added to nodes based on files that are located in the directory in a cookbook.

```
cookbook_file '/var/www/customers/public_html/index.php'
do
    source 'index.php'
    owner 'web_admin'
    group 'web_admin'
    mode '0755'
    action :create
end
```

Files

- Use the file resource to manage files directly on a node.

```
file '/var/www/customers/public_html/index.php' do
    content '<html>This is a placeholder for the home
page.</html>'
    mode '0755'
    owner 'web_admin'
    group 'web_admin'
end
```

Remote file

- A **remote_file** resource block manages files by using files that exist remotely.

```
remote_file '/var/www/customers/public_html/index.php' do
    source 'http://somesite.com/index.php'
    owner 'web_admin'
    group 'web_admin'
    mode '0755'
    action :create
end
```

Definition

- A definition behaves like a compile-time macro that is reusable across recipes.
- A definition is typically created by wrapping arbitrary code around resources that are declared as if they were in a recipe.
- A definition is then used in one (or more) actual recipes as if the definition were a resource.

Definition

- A definition has four components:
  - A resource name
  - Zero or more arguments that define parameters their default values; if a default value is not specified, it is assumed to be nil
  - A hash that can be used within a definition's body to provide access to parameters and their values
  - The body of the definition

```
define :host_porter, :port => 4000, :hostname => nil do
    params[:hostname] ||= params[:name]
directory '/etc/#{params[:hostname]}' do
        recursive true
    end
file '/etc/#{params[:hostname]}/#{params[:port]}' do
        content 'some content'
    end
end
```

Definition

which is then used in a recipe like this:

```
host_porter node['hostname'] do
  port 4000
end
host_porter 'www1' do
    port 4001
end
```

library

- A library allows arbitrary Ruby code to be included in a cookbook, either as a way of extending the classes that are built-in to the chef-client
- A library file is a Ruby file that is located within a cookbook's /libraries directory
- Because a library is built using Ruby, anything that can be done with Ruby can be done in a library file.

Custom Resource

- A custom resource:
  - Is a simple extension of Chef that adds your own resources
  - Is implemented and shipped as part of a cookbook
  - Follows easy, repeatable syntax patterns
  - Effectively leverages resources that are built into Chef and/or custom Ruby code
  - Is reusable in the same way as resources that are built into Chef
- A custom resource is defined as a Ruby file and is located in a cookbook's /resources directory
  - Declares the properties of the custom resource
  - Loads current properties, if the resource already exists
  - Defines each action the custom resource may take

Custom Resource

```
property :name, RubyType, default: 'value'
load_current_value do
    # some Ruby
end
action :name do
  # a mix of built-in Chef resources and Ruby
end
action :name do
  # a mix of built-in Chef resources and Ruby
end
```

Custom Resource

```
property :homepage, String, default: '<h1>Hello world!</h1>'
load_current_value do
    if ::File.exist?('/var/www/html/index.html')
        homepage IO.read('/var/www/html/index.html')
    end
end
action :create do
    package 'httpd'
service 'httpd' do
        action [:enable, :start]
    end
file '/var/www/html/index.html' do
        content homepage
    end
end
action :delete do
    package 'httpd' do
        action :delete
    end
end
```

Custom Resource

```
exampleco_site 'httpd' do
    homepage '<h1>Welcome to the Example Co. website!</h1>'
    action :create
end

exampleco_site 'httpd' do
    action :delete
end
```

Cookbook Directories and Metadata

- The cookbooks/ directory is used to store the cookbooks that are used by the chef-client when configuring the various systems in the organization.
- This directory contains the cookbooks that are used to configure systems in the infrastructure.
- Each cookbook can be configured to contain cookbook-specific copyright, email, and license data.

Cookbook Directories and Metadata

- Every cookbook requires a small amount of metadata.
- A file named metadata.rb is located at the top of every cookbook directory structure.
- The contents of the metadata.rb file provides hints to the Chef server to help ensure that cookbooks are deployed to each node correctly.
- A metadata.rb file is:
  - Located at the top level of a cookbook's directory structure
  - Compiled whenever a cookbook is uploaded to the Chef server or when the knife cookbook metadatasubcommand is run, and then stored as JSON data
  - Created automatically by knife whenever the knife cookbook create subcommand is run
  - Edited using a text editor, and then re-uploaded to the Chef server as part of a cookbook upload

Chef components interaction