Fundamental of Chef

Day 1 & 2

Rajesh Kumar

RajeshKumarIN

RajeshKumarIN

RajeshKumarIN

DevOps@RajeshKumar.XYZ

Agenda of the day

Melting Ice off Chef

Formal

- Overview of Chef
- Workstation Setup
- Test Node Setup
- Dissecting your first Chef run
- Introducing the Node object
- Writing your first cookbook

Questions?

No Question of Small Question!!!

What Is Configuration Management?

With respect to IT, *configuration management* covers the set of engineering practices for managing the following entities involved in delivering software applications to consumers:

- Hardware
- Software
- Infrastructure
- People
- Process

Trying to coordinate the work of multiple system administrators and developers involving hundreds, or even thousands, of servers and applications to support a large customer base is complex and typically requires the support of a tool.

Tools

Examples of modern IT configuration management tools are …

CFEngine,

Puppet,

Ansible,

SaltStack, and of course,

Chef

Why You Need a Configuration Management Tool to Automate IT

- **Consistency:** If your infrastructure is being configured manually, how do you know your servers are being set up in a consistent manner? Further, how do you know these changes are being performed in a way that meets your compliance and security requirements?
- **Efficient change management:** Whenever infrastructure is built manually without the aid of a configuration management tool, people tend to fear change. Over time, servers that are maintained by hand tend to become fragile environments that are hard to understand and modify.
- **Simplicity in rebuild:** When servers are built manually, it's typically not easy to rebuild them from scratch. What would happen if you suddenly lost your servers in a catastrophic event? How quickly could you restore service if disaster struck?
- **Visibility:** Configuration management tools include auditing and reporting capabilities. Monitoring the work performed by one system administrator doesn't require a sophisticated tool. But trying to understand what is going on with a team of, say, 10 system administrators and 10 software developers deploying software changes many times per day? You need a configuration tool.

What is Chef?

Chef is a Ruby framework for automating, reusing, and documenting server configuration.

Chef is like a unit test for your servers

Ruby

Lets keep for end of the session

More

- Chef is a configuration management tool written in ruby and Erlang.
- Used to streamline the task of configuring and maintaining a company’ servers
- It can integrate with cloud based platforms such as Rackspace, Amazon EC2, Google Cloud Platform, OpenStacke, Softlayer and Microsoft Azure to automatically provision and configure the new machine
- It contains solutions for both small and large scale systems

Chef does not

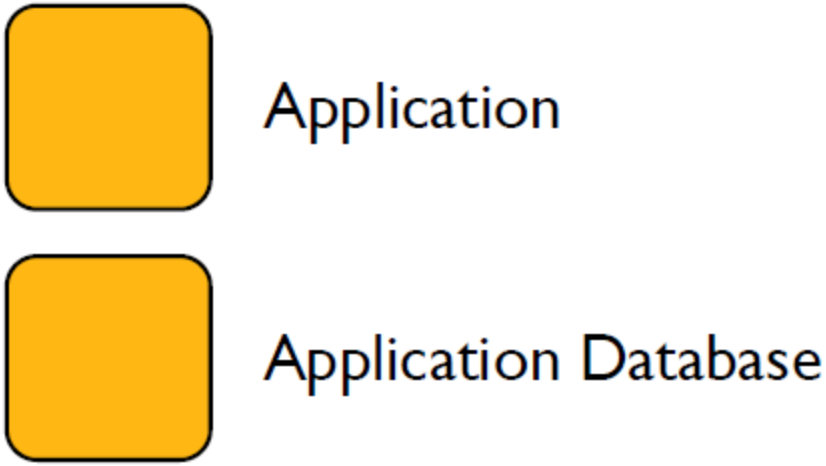
- Chef does not
  - Monitor the runtime behaviour of any of the software it configures
  - Chef can not
    - Tell you whether or not a service is running
    - Undoing changes

- If configuration file does not need to updated and it should not be updated
  - You can run a script several times, but it wont change anything after the first run
  - If none of your inputs to chef change, running it over and over should not try to run all the same commands over and over

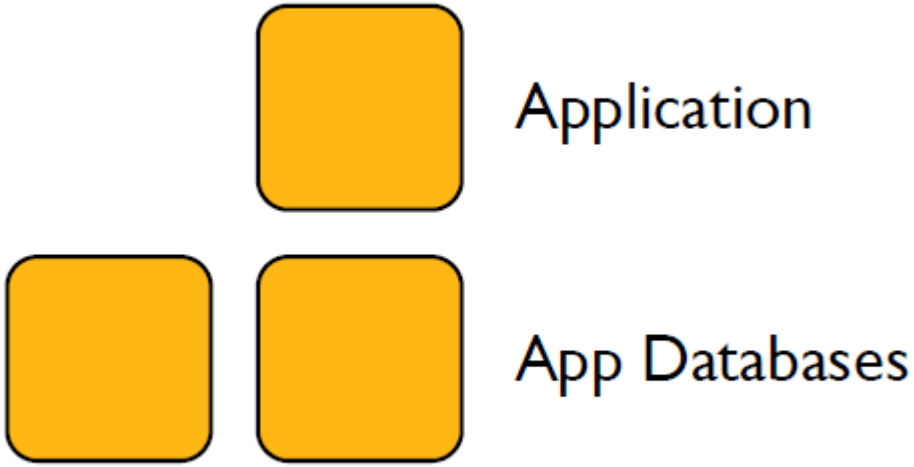
A tale of growth...

Application

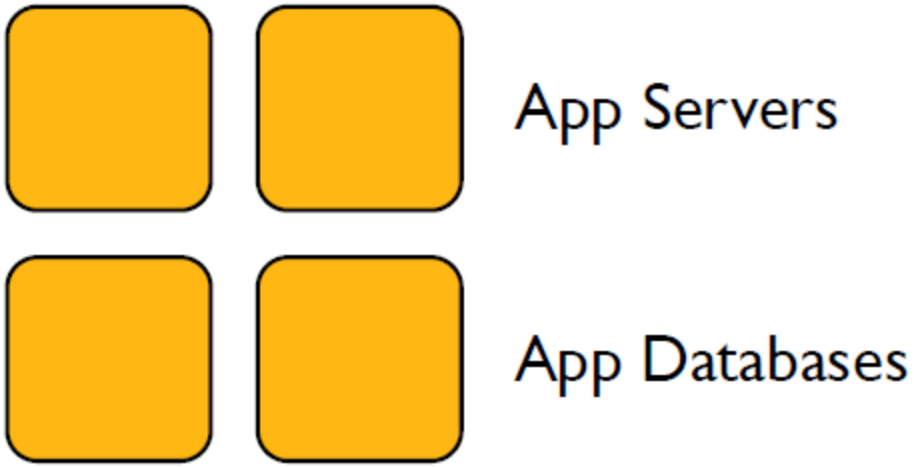
Add a database



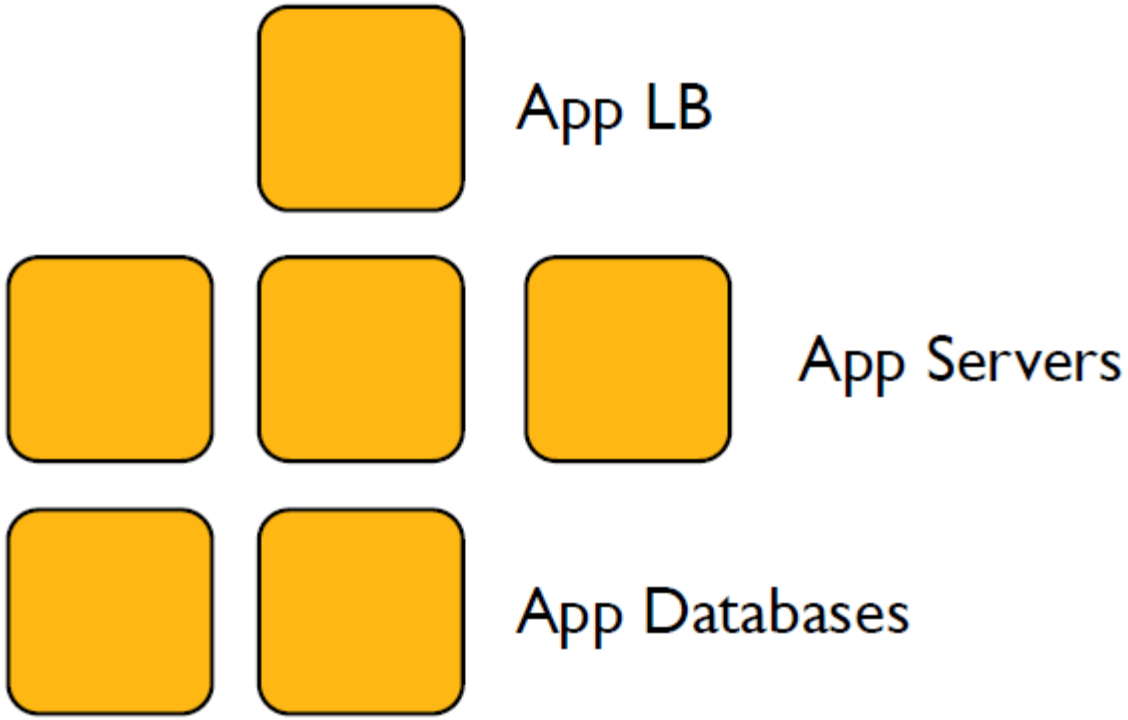
Make database redundant



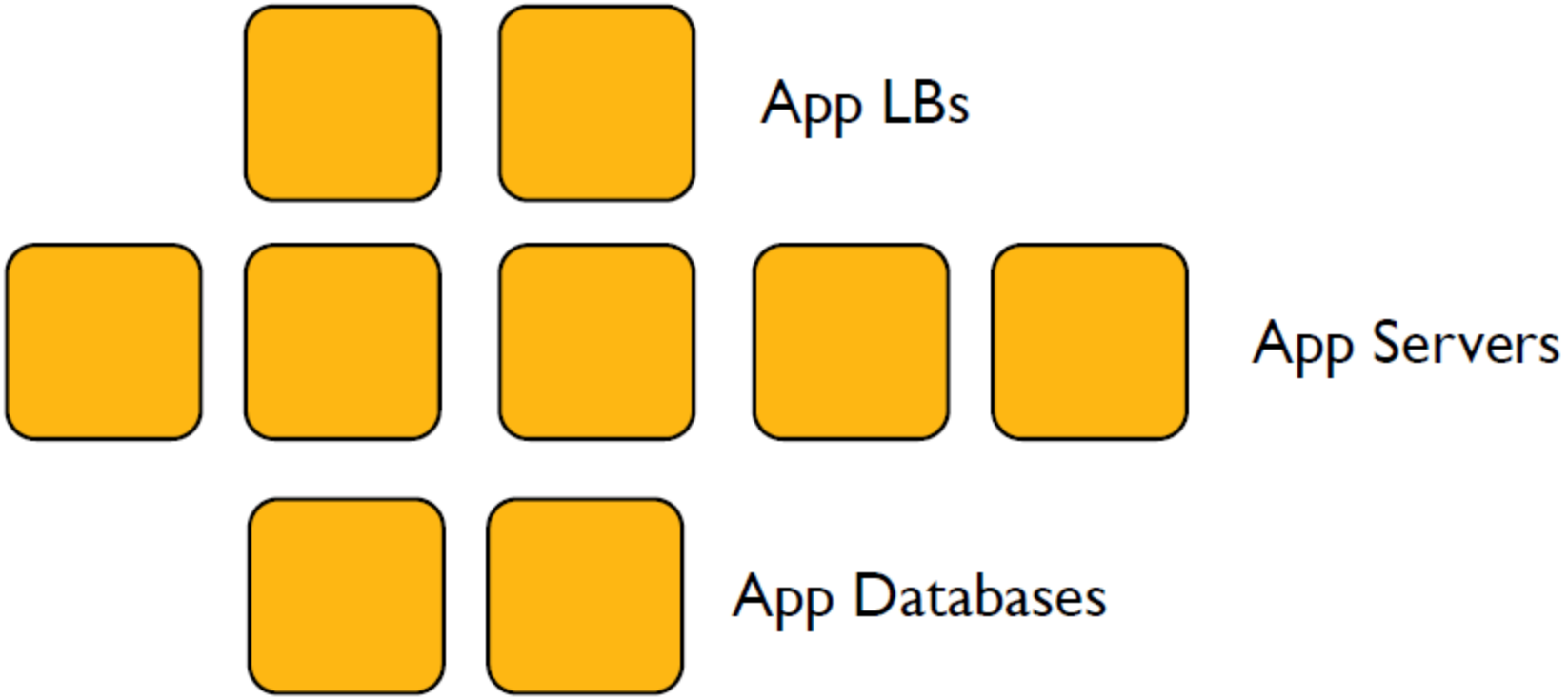
Application server redundancy



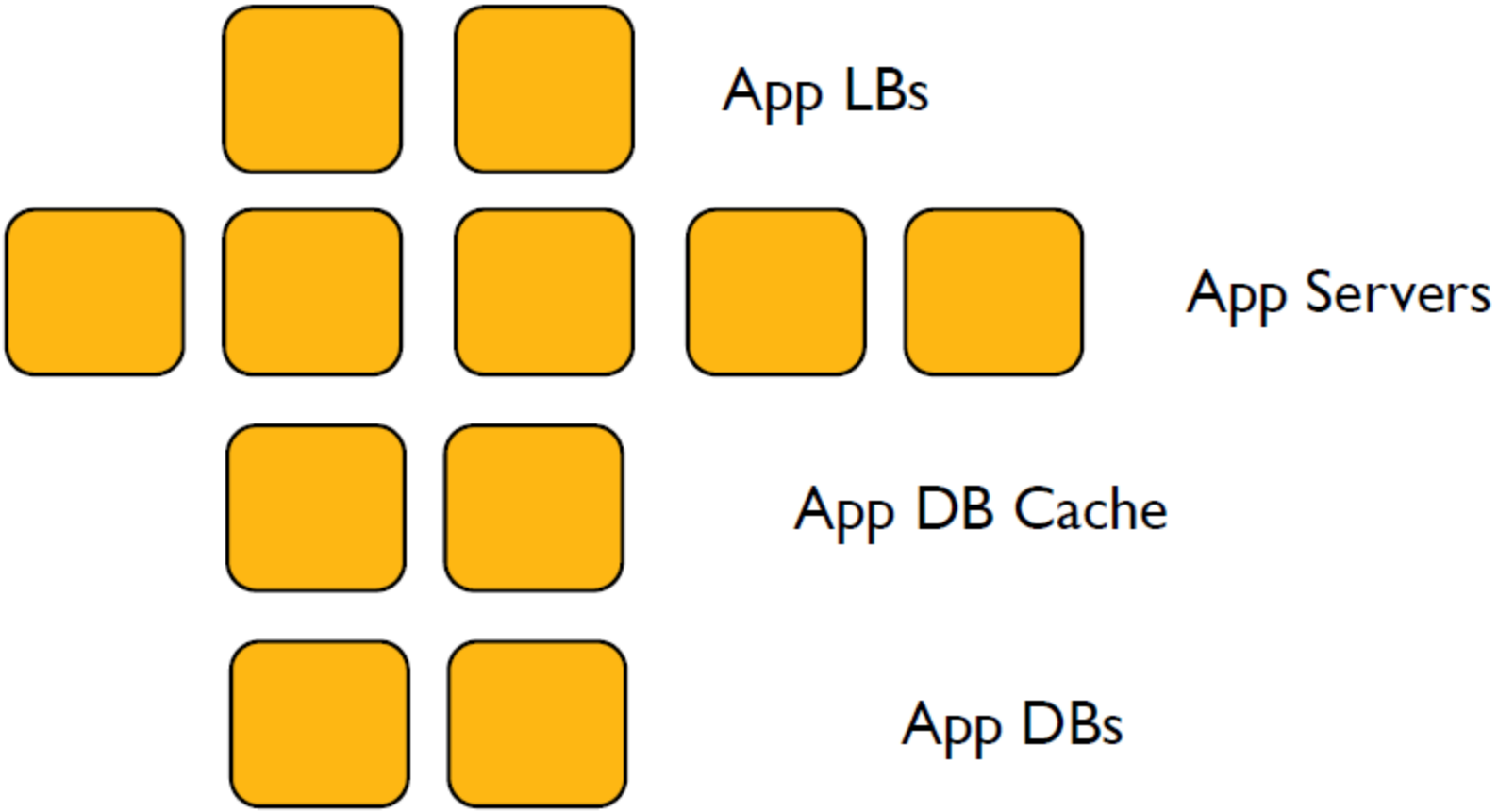
Add a load balancer



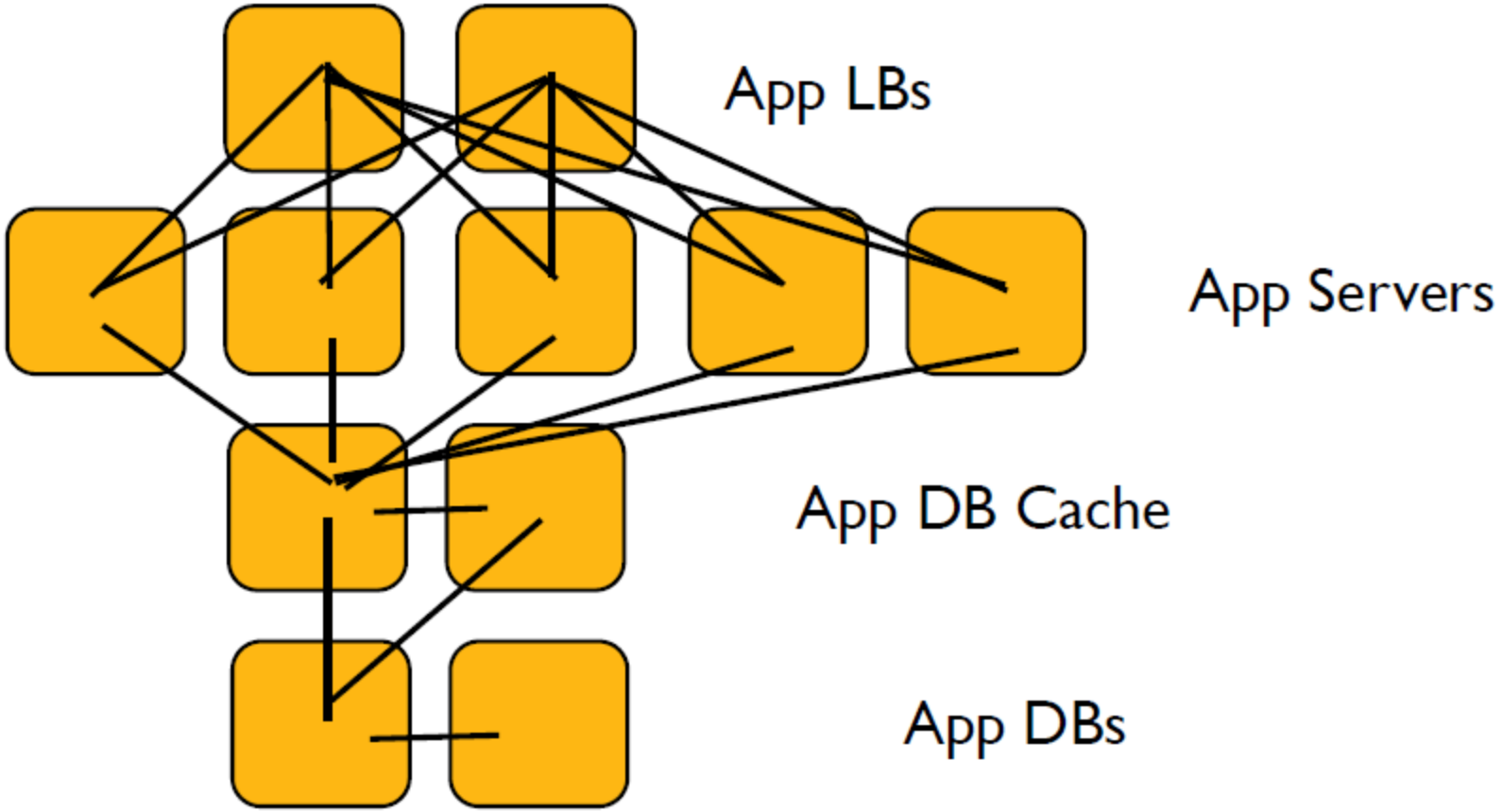
Webscale!



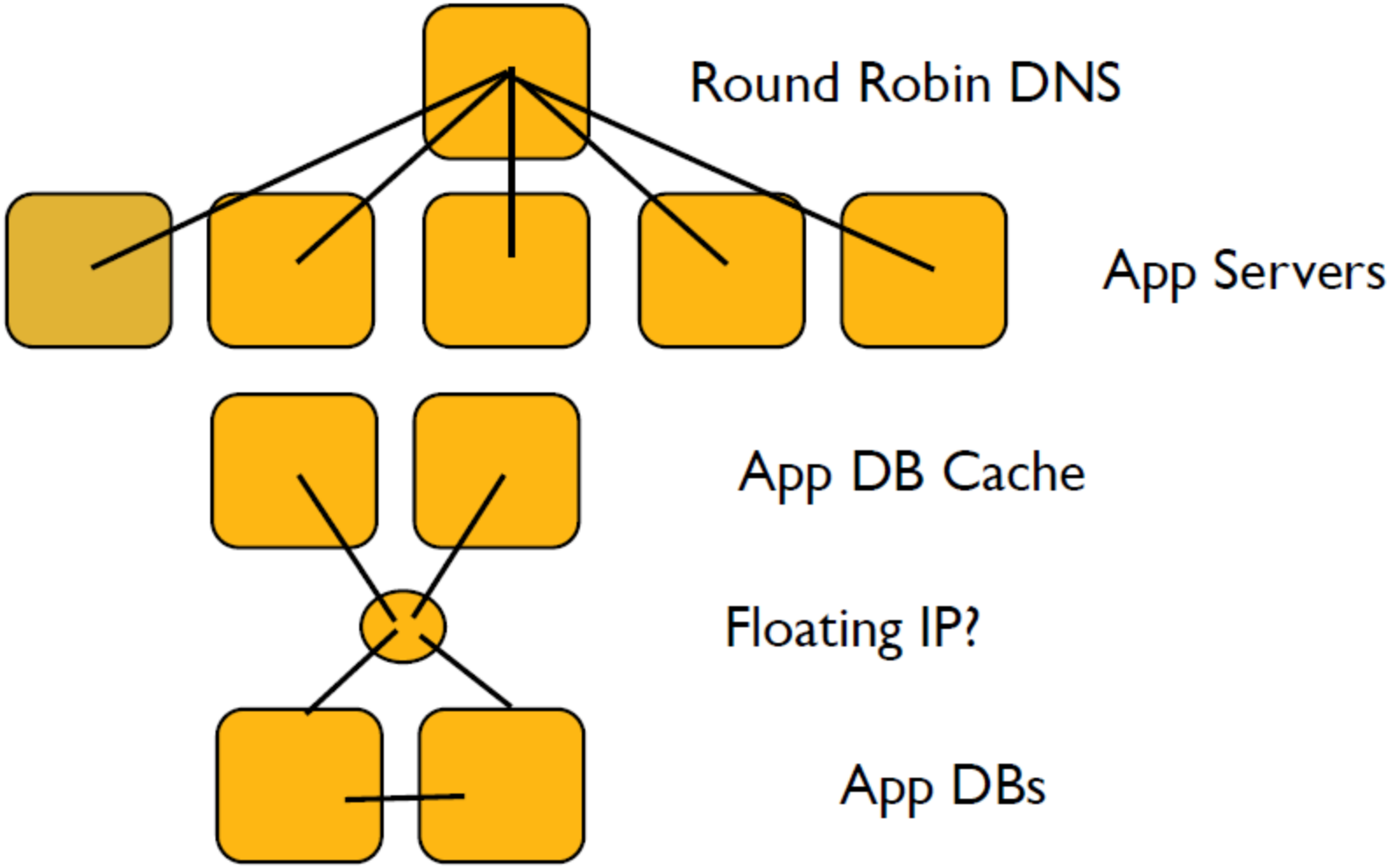
Now we need a caching layer



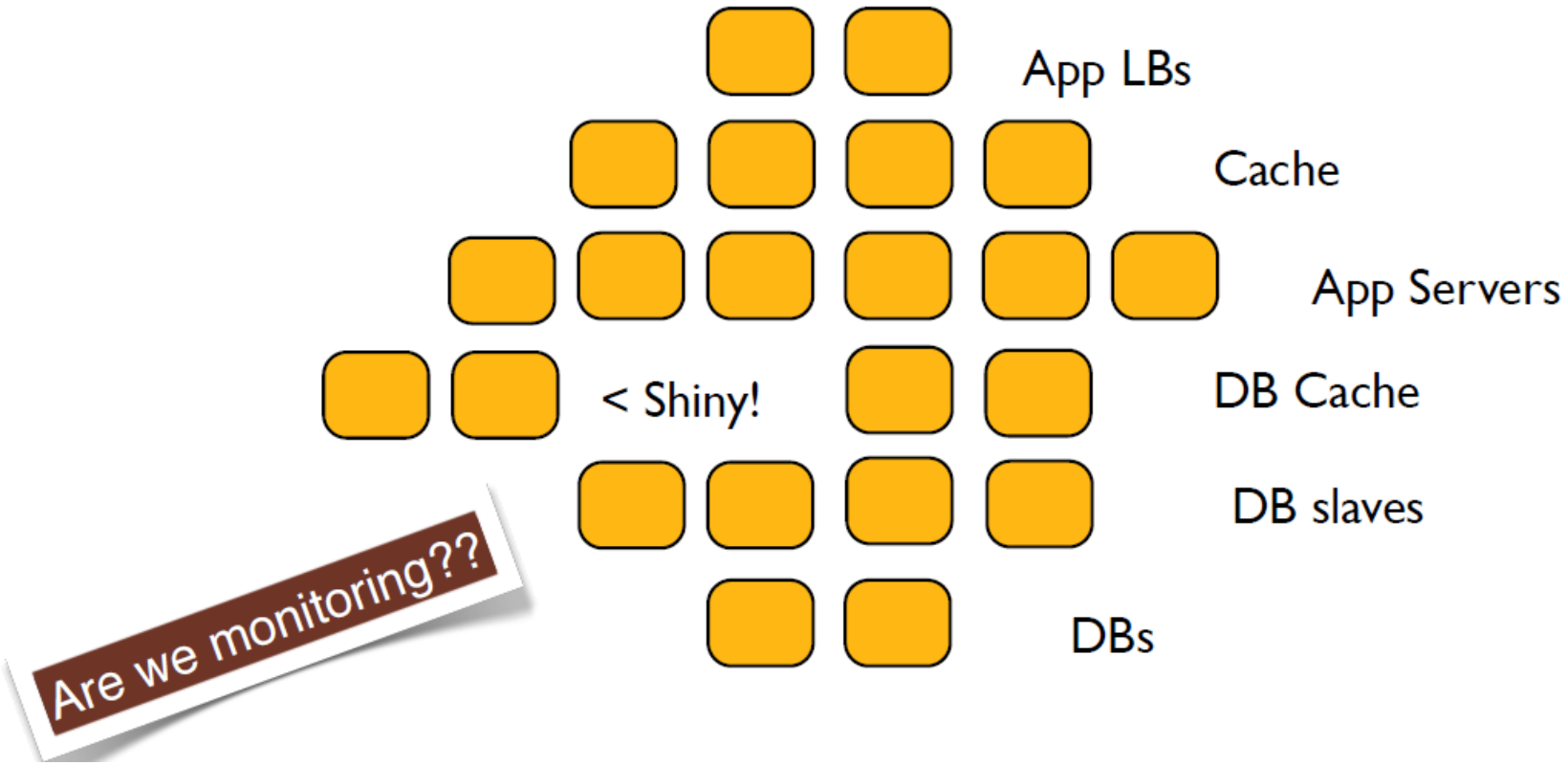
Infrastructure has a Topology



Your Infrastructure is a Snowflake



Complexity Increases Quickly



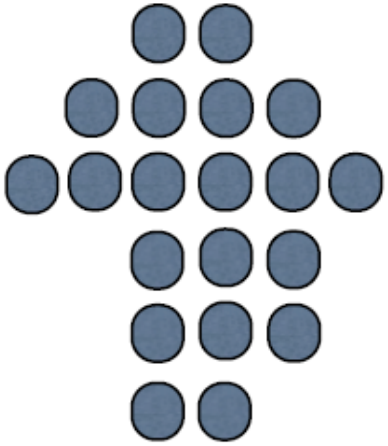
Chef Solves this problem

Chef Managing Complexity

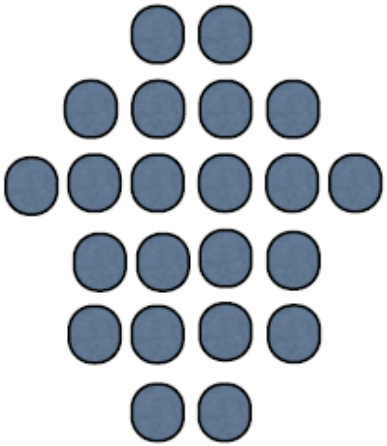
- Organizations
- Environments
- Roles
- Nodes
- Recipes
- Cookbooks
- Search

Organizations

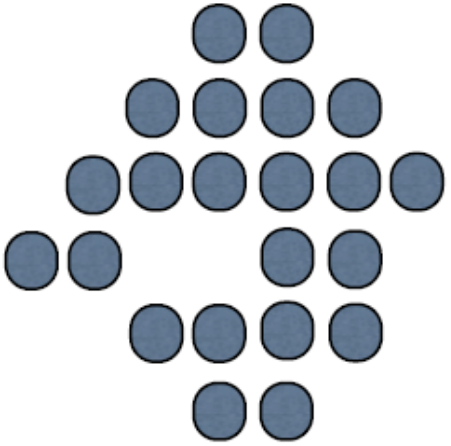
My Infrastructure



Your Infrastructure



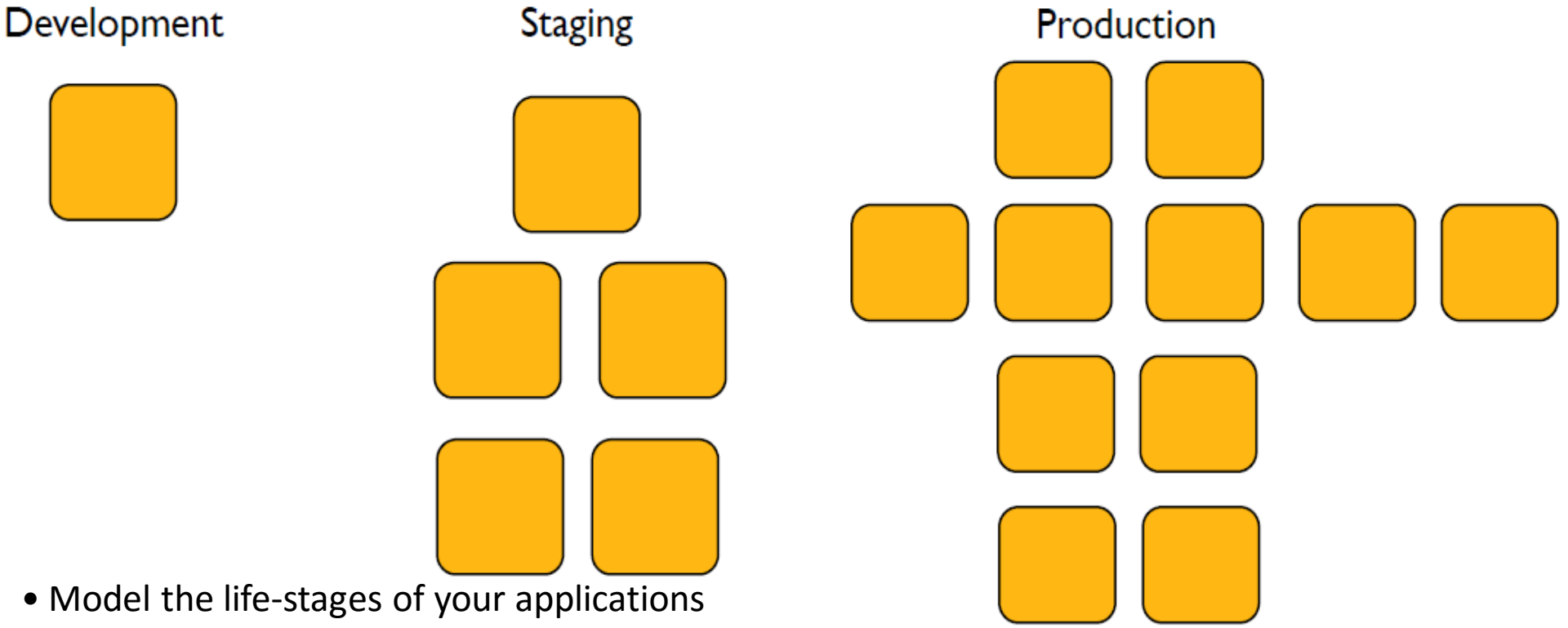
Their Infrastructure



Completely independent tenants of Enterprise Chef
Share nothing with other organizations
May represent different

- Companies
- Business Units
- Departments

Environments



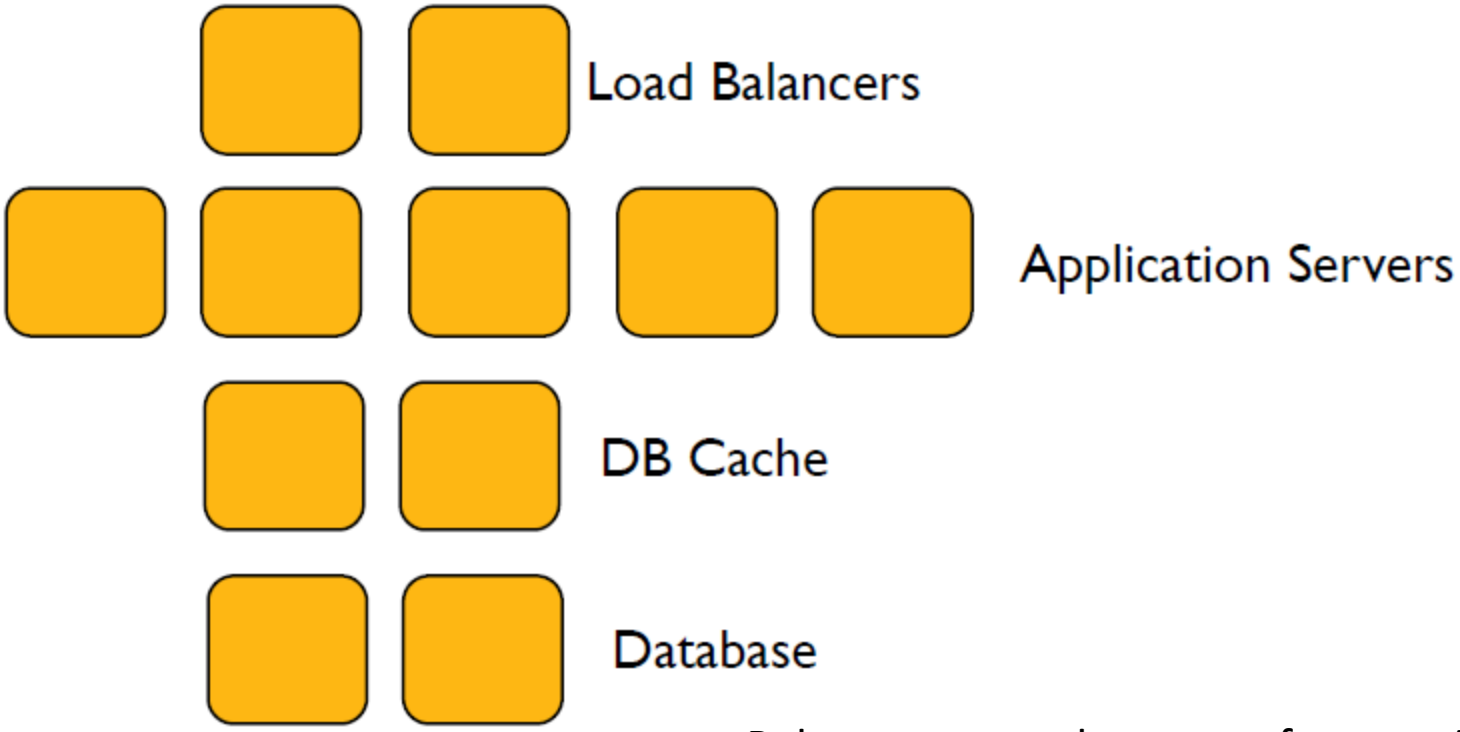
- Model the life-stages of your applications
- Every Organization starts with a single environment
- Environments to reflect your patterns and workflow
    - Development
    - Test
    - Staging
    - Production
    - etc.

Environments Define Policy

Environments may include data attributes necessary for configuring your infrastructure

- The URL of your payment service's API
- The location of your package repository
- The version of the Chef configuration files that should be used

Roles



Roles represent the types of servers in your infrastructure

- Load Balancer
- Application Server
- Database Cache
- Database
- Monitoring

Roles Define Policy

Roles may include a list of Chef configuration files that should be applied. We call this list a Run List

Roles may include data attributes necessary for configuring your infrastructure

- The port that the application server listens on
- A list of applications that should be deployed

Why Chef

- Idempotent: Safe to re run the script
- Thick Clients, Thin Server
- A Level of platform independencies
- Rich Collection of Recipes
- Readable
- Accessible
- Repeatable
- Speed
- Overcome of fears of system admin work
- Development env in sync with production

Chef terminology

Not now

After completing this lesson…

- Have a basic understanding of what happens when chef runs.
- Be able to write Chef code that defines a basic policy
- Be able to apply that policy to server

Get a Virtual Machine

http://learn.chef.io/learn-the-basics/rhel/get-set-up/

Install Chef Development Kit (ChefDK)

https://downloads.chef.io/chef-dk/

How to install ChefDk in RHEL, Ubantu, Mac and Windows:

http://www.scmgalaxy.com/scm/configuration-management-tools/how-to-install-chefdk-in-rhel-ubantu-mac-and-windows.html

Configure a resources

Create the MOTD file

mkdir ~/chef-repo
cd ~/chef-repo

Hello.rb

```
file 'motd' do
        content 'hello world'
end
```

Save to hello.rb
&
chef-apply hello.rb

Now what?

- Run the command a second time
- Update the MOTD file's contents
- Ensure the MOTD file's contents are not changed by anyone else

Delete the MOTD file

goodbye.rb

```
file 'motd' do
        action :delete
end
```

file ‘/use/path/motd' do
*   action :create*
*   content 'hello world'*
end

chef-apply (executable)

Use chef-apply to run a single recipe from the command line.

More - https://docs.chef.io/ctl_chef_apply.html

Excercise

- What is a *resource*?
- What is a *recipe*?
- What happens when you don't specify a resource's action?
- Modify the hello.rb recipe you wrote in this lesson to manage the MOTD file under the/tmp/messages directory, and not in the current directory.

Answer

When you don't specify a resource's action, Chef applies the default action. For example, this resource:

```
directory '/tmp/messages'
        file '/tmp/messages/motd' do
        content 'hello world'
        end
```

Chef Terminology (1)

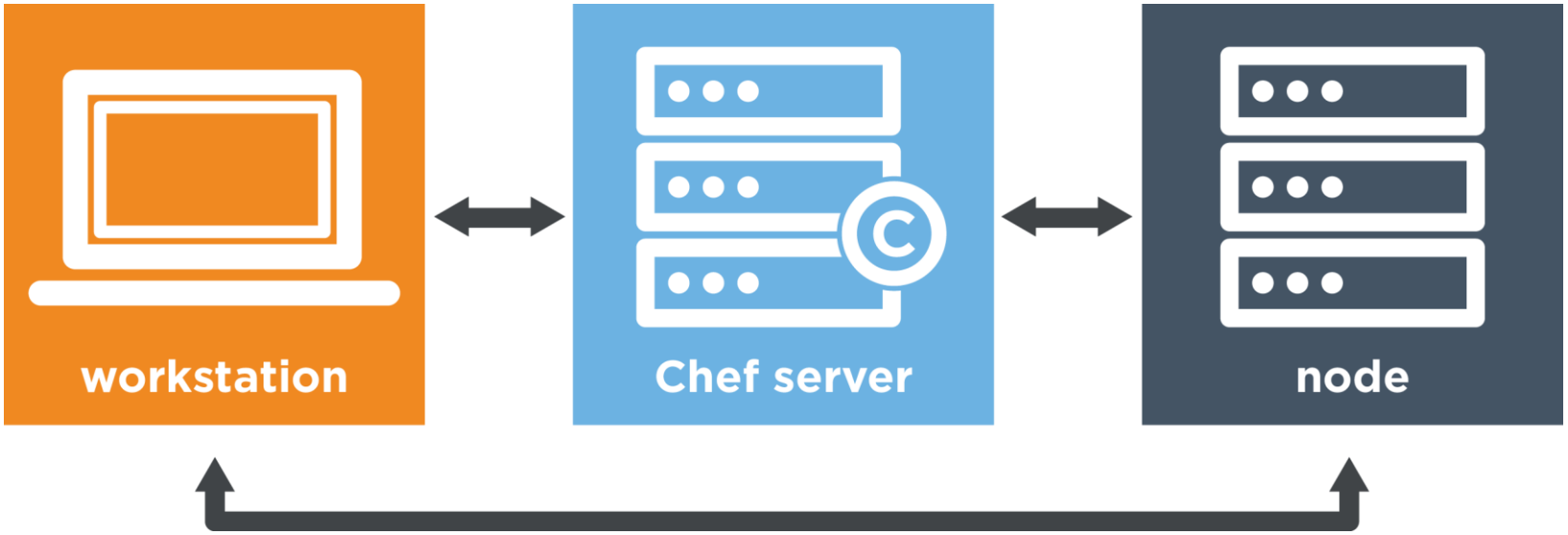
- Chef client is installed on nodes (machines) which are registered with the Chef server
- Developers write code on workstations and use tools such as knife to interact with server
- Chef models node configuration as a set of DSL resources (e.g. package, service, directory) which are mapped to internal providers (actual code to execute)
  - Can define custom resources

Chef Terminology (2)

- A recipe declares a set of resources with desired configuration
- A cookbook contains a set of semantically-related code and is the fundamental unit of distribution for Chef code
- A data bag holds JSON information in one or more data bag items accessible from Chef code
- Chef environments model deployed environments
- Each node has a run list containing recipes

Chef Server

A database-backed web server that stores searachable information about your production servers. REST-based.


workstation
Chef server
node

Chef Client

The command line programs that configures servers

Cookbooks

- Recipes are stored in Cookbooks
- Cookbooks contain recipes,templates, files, custom resources, etc
- Code re-use and modularity

Chef solo

- A Standalone version of the chef client that does not rely on the server for configuration

Chef Recipe

- A Single file of Ruby code that contains commands to run on a node. It describe a series of resources that should be in particualar state:
  - Package that should be installed
  - Services that should be running or
  - files that should be written

- A recipe is a collection of resources that describes a particular configuration or policy. A recipe describes everything that is required to configure part of a system. Recipes do things such as:
  - install and configure software components.
  - manage files.
  - deploy applications.
  - execute other recipes.

Chef Resources

- A Node’s Resources includes files, directories, users, and services (Unix processing).
- A resource represents a piece of infrastructure and its desired state, such as a package that should be installed, a service that should be running, or a file that should be generated.
- Every resource in Chef has a default action, and it's often the most common affirmative one – for example, *create* a file, *install* a package, and *start* a service.

Resources

A Resource represents a piece of the system and its desired state

- A package that should be installed
- A service that should be running
- A file that should be generated
- A cron job that should be configured
- A user that should be managed
- Resources are the fundamental building blocks of Chef configuration
- Resources are gathered into Recipes
- Recipes ensure the system is in the desired state

Resources can be of many different types

- **package**: Used to manage packages on a node
- **service**: Used to manage services on a node
- **user**: Manage users on the node
- **group**: Manage groups
- **template**: Manage files with embedded ruby templates
- **cookbook_file**: Transfer files from the files subdirectory in the cookbook to a location on the node
- **file**: Manage contents of a file on node
- **directory**: Manage directories on node
- **execute**: Execute a command on the node
- **cron**: Edit an existing cron file on the node

Items of Manipulation (Resources)

- Nodes
- Networking
- Files
- Directories
- Symlinks
- Mounts
- Routes
- Users
- Groups
- Packages
- Services
- Filesystems

Roles

- Reusable configuration of multiple nodes

Run list

- A List of Recipes and roles that define what will be executed on a node. Chef figures out the intersection of these and configures a node accordingly

Attributes

- Variable that are passed through Chef and used in recipes and templates eg. The version number of Nginx to install.

Template

- A file with placeholders for attributes. This will be use to create configuration files

Notification

- When a resources is changed, it can trigger an update is another resource.

Chef folders

- **folder**
  - **recipes**
    - default.rb
  - **templates**
  - **attributes**
  - providers
  - resources
  - metadata.rb
  - files

Chef Install

- sudo apt-get install filters
- sudo apt-get install chef (to install chef client and solo)

Configure a package and service

webserver.rb

```
package 'httpd'

service 'httpd' do
       action [:start, :enable]
end

file '/var/www/html/index.html' do
  content '<html>
  <body>
    <h1>hello world</h1>
  </body>
</html>'
end

service 'iptables' do
  action :stop
end
~
```

sudo chef-apply webserver.rb

order

- Chef works in the order you specify

Excercise

Are these two recipes the same?

```
package 'httpd‘
service 'httpd' do
  action [:start, :enable]
End

service 'httpd' do
  action [:start, :enable]
end
package 'httpd'
```

Answer

No, they are not. Remember that Chef applies resources in the order they appear. So the first recipe ensures that thehttpd package is installed and then configures the service. The second recipe configures the service and then ensures the package is installed. The second recipe may not work as you'd expect because
the service resource will fail if the package is not yet installed.

Exercise

Are these two recipes the same?

```
package 'httpd'

service 'httpd' do
  action [:enable, :start]
end
```

```
package 'httpd'

service 'httpd' do
  action [:start, :enable]
end
```

Answer

No, they are not. Although both recipes ensure that the httpd package is installed before configuring its service, the first recipe enables the service when the system boots and then starts it. The second recipe starts the service and then enables it to start on reboot.

Excercise

Are these two recipes the same?

```
file '/etc/motd' do
  owner 'root'
  group 'root'
  mode '0755'
  action :delete
end

file '/etc/motd' do
  action :create
  mode '0755'
  group 'root'
  owner 'root'
end
```

Answer

Yes, they are! Order matters with a lot of things in Chef, but you can order resource attributes any way you want.

Excercise

Write a service resource that stops and then disables the apache2 service from starting when the system boots.

Answer

```
service 'httpd' do
  action [:stop, :disable]
end
```

Manage your recipe

Create a cookbook

```
> chef generate cookbook learn_chef_httpd

tree
.
└── learn_chef_httpd
    ├── Berksfile
    ├── chefignore
    ├── metadata.rb
    ├── README.md
    └── recipes
        └── default.rb

2 directories, 5 files
```

Create a template

- chef generate template learn_chef_httpd index.html

tree

```
.
└── learn_chef_httpd
    ├── Berksfile
    ├── chefignore
    ├── metadata.rb
    ├── README.md
    ├── recipes
    │   └── default.rb
    └── templates
        └── default
            └── index.html.erb

4 directories, 6 files
```

The .erb extension simply means that the file can have placeholders.

Update template file

```
<html>
        <body>
                <h1>hello world</h1>
        </body>
</html>
```

Update the recipe to reference the HTML template

Write out the recipe, default.rb, like this.

```
package 'httpd'

service 'httpd' do
  action [:start, :enable]
end

template '/var/www/html/index.html' do
  source 'index.html.erb'
end

service 'iptables' do
  action :stop
end
```

Run the cookbook

sudo chef-client --local-mode --runlist 'recipe[learn_chef_httpd]‘

Note:

When you run `chef-client`, it looks for a ./cookbooks directory for cookbooks that it can use in the run-list you supply. You can modify the paths that it searches in the ./.chef/knife.rb or ~/.chef/knife.rb

Reference - https://docs.chef.io/config_rb_client.html

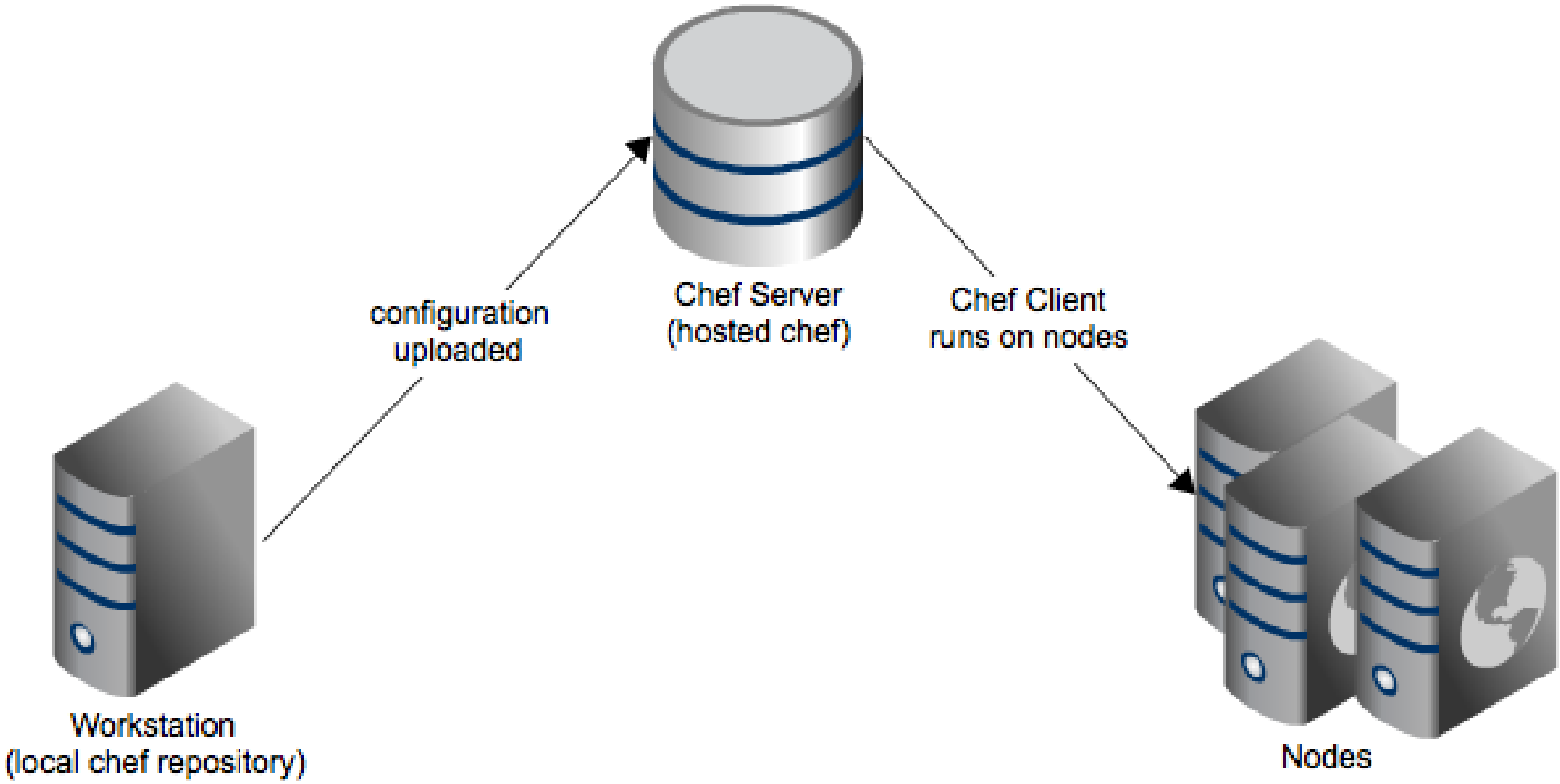
local_mode

Use to run the chef-client in local mode. This allows all commands that work against the Chef server to also work against the local chef-repo.

- Curl localhost

chef-apply to run a single recipe from the command line.

chef-client is what you use to run a cookbook.


configuration
uploaded
Chef Server
(hosted chef)
Chef Client
runs on nodes
Workstation
(local chef repository)
Nodes

Excercise

How does a cookbook differ from a recipe?

Answer

A recipe is a collection of resources, and typically configures a software package or some piece of infrastructure.

A cookbook groups together recipes and other information in a way that is more manageable than having just recipes alone.

Excercise

How does chef-apply differ from chef-client?

Answer

chef-apply applies a single recipe; chef-client applies a cookbook.

For learning purposes, we had you start off with chef-apply because it helps you understand the basics quickly. In practice, chef-apply is useful when you want to quickly test something out. But for production purposes, you typically run chef-client to apply one or more cookbooks.

Excercise

- What's the *run-list*?

Answer

The run-list lets you specify which recipes to run, and the order in which to run them.

The run-list is important for when you have multiple cookbooks, and the order in which they run matters.

Lab

Install **Nginx**
Start **Nginx**
Stop Nginx
Modify the file **Nginx**
Start Nginx

Index.html - /usr/share/nginx/www/index.html (RHEL)

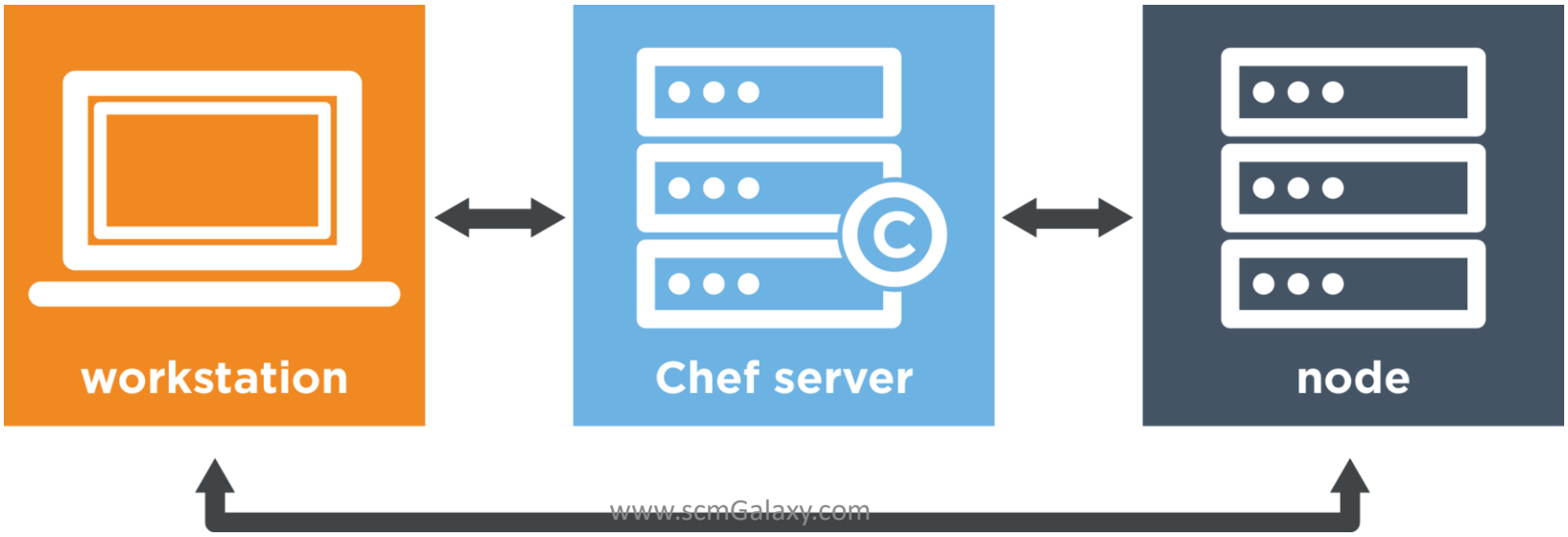
Nginx (pronounced "engine-x") is an open source reverse proxy server for HTTP, HTTPS, SMTP, POP3, and IMAP protocols, as well as a load balancer, HTTP cache, and a web server (origin server). The **nginx**project started with a strong focus on high concurrency, high performance and low memory usage.

Git & Github - Done

Manage Node

Typically, Chef is comprised of three elements –

- your workstation,
- a Chef server,
- and nodes.



- Your **workstation** is the computer from which you author your cookbooks and administer your network. It's typically the machine you use everyday. Although you'll be configuring a Red Hat Enterprise Linux server, your workstation can be any OS you choose – be it Linux, Mac OS, or Windows.
- **Chef server** acts as a central repository for your **cookbooks** as well as for information about every **node** it manages. For example, the Chef server knows a node's fully qualified domain name (FQDN) and its platform.
- A **node** is any computer that is managed by a Chef server. Every node has the Chef client installed on it. The Chef client talks to the Chef server. A node can be any physical or virtual machine in your network.

After completing this session, you'll:

- Be able to write Chef code to define a policy from your workstation.
- be able to apply that policy to a node.
- understand how to access cookbooks written by the Chef community.

Setup Workstation

- Install Chefdk In your workstation
- https://downloads.chef.io/chef-dk/

ChefDK

- ChefDK contains:
- An early version of a brand new command-line tool, chef, that aims to streamline Chef workflow, starting with new generators.
- The well-known cookbook dependency manager Berkshelf 3.0.
- The Test Kitchen integration testing framework.
- ChefSpec, which makes unit testing cookbooks a breeze.
- Foodcritic, a linting tool for doing **static code analysis** on cookbooks.
- All of the Chef tools you're already familiar with: Chef Client, Knife, Ohai and Chef Zero.

Install ChefDK

- Windows
  - exe
- Ubantu
  - sudo dpkg -i askubuntu_2.0.deb
- RHEL
  - rpm –i file

Download - https://www.chef.io/chef/choose-your-version/

Install - https://docs.chef.io/install_dk.html#get-package-run-installer

chef verify

Setup Chef Server

Setup your own Chef Server

Or

Sign up for hosted Chef

https://manage.chef.io/signup/

Install starterkit

Upload Your cookbook

> knife cookbook upload learn_chef_httpd

Get a Apache Cookbook

- knife cookbook site download learn_chef_httpd
- tar -zxvf learn_chef_httpd-0.1.0.tar.gz -C cookbooks

Exercise

- What are the two ways to set up a Chef server?

Answer

- Install an instance on your own infrastructure.
- Use hosted Chef.

Excercise

What's the role of the Starter Kit?

The Starter Kit provides certificates and other files that enable you to securely communicate with the Chef server.

Excercise

Where can you get reusable cookbooks that are written and maintained by the Chef community?

Answer

Chef Supermarket, https://supermarket.chef.io.

Excercise

What's the command that enables you to interact with the Chef server?

Answer

knife

Bootstrap your node

you ran chef-apply and chef-client to configure the node directly.

Now you'll use knife to configure your node remotely, from your workstation.

Get a Linux machine to bootstrap

http://learn.chef.io/manage-a-node/rhel/bootstrap-your-node/

Bootstrap Your nodes

- knife bootstrap {{address}} --ssh-user {{user}} --ssh-password '{{password}}' --sudo --use-sudo-password --node-name node1 --run-list 'recipe[learn_chef_httpd]‘

- {{address}} – Node Host Address
- {{user}} – With Node login id
- {{password}} – With Login ID password
- --node-name – Name of the node

Confirm the result

- knife node list
- knife node show <nodename>

knife bootstrap

- The knife bootstrap command established an SSH connection to the node, installed chef-client, downloaded the Learn Chef Apache cookbook on the node, and ran it. In one command, Chef carried out most of the steps you previously dealt with manually.
- A powerful part of the knife bootstrap process is that you did not need to connect to or interact with the server directly. This enables you to further automate the process of provisioning and configuring your infrastructure. But if you'd like, you can connect to the server now to verify that everything is set up as you'd expect.

Excercise

- What is a node?
- What information do you need to in order to bootstrap?
- What happens during the bootstrap process?

Answer

- During the bootstrap process, the node downloads and installs chef-client, registers itself with the Chef server, and does an initial checkin. During this checkin, the node applies any cookbooks that are part of its run-list.

Update your node Configuration

Use the template resource with placeholders:

On the local workstation copy of your learn_chef_httpd cookbook, change index.html.erb to look like this.

```
<html>
  <body>
    <h1>hello from <%= node['fqdn'] %></h1>
  </body>
</html>
```

Upload your cookbook to the Chef server

```
> knife cookbook upload learn_chef_httpd
```

Run the cookbook on your node

> knife ssh {{address}} 'sudo chef-client' --manual-list --ssh-user {{user}} --ssh-password '{{password}}'

knife ssh

- You ran knife ssh to update your node. knife ssh invokes the command you specify over an SSH connection on a node – in our case sudo chef-client. You didn't have to specify the run-list because you already set that up when you bootstrapped the node

Excercise

- What is the command you use to upload a cookbook to the Chef server?
- How do you apply an updated cookbook to your node?
- Update your Apache cookbook to display your node's host name, platform, total installed memory, and number of CPUs in addition to its FQDN on the home page.

Update your Apache cookbook to display your node's host name, platform, total installed memory, and number of CPUs in addition to its FQDN on the home page.

<%= node[‘hostname'] %>
<%= node['fqdn'] %>
<%= node['fqdn'] %>
<%= node['fqdn'] %></

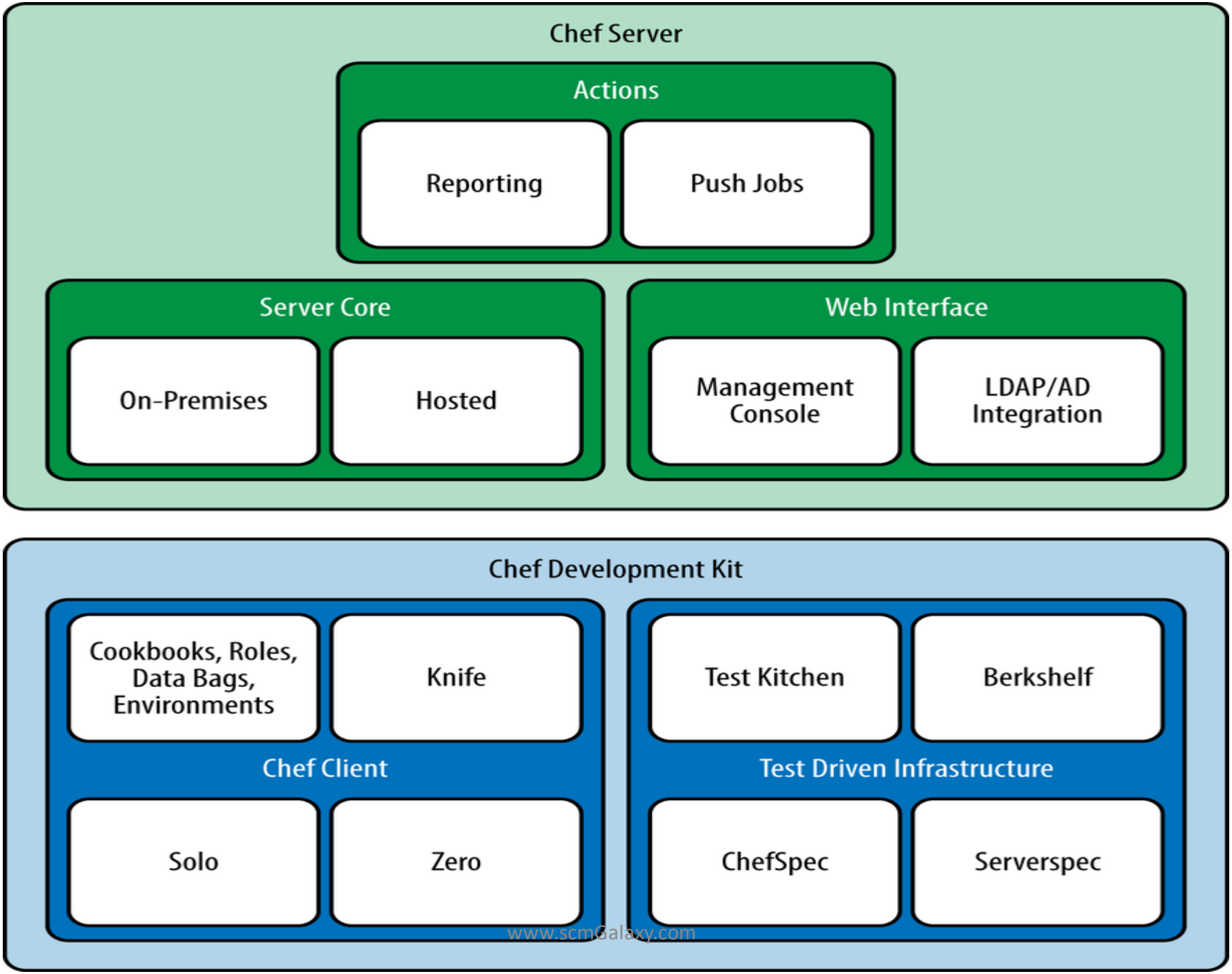
```
<html>
 <body>
  <h1>hello from <%= node['fqdn'] %></h1>
 </body>
</html>
```

Answer

3.

```
<html>
  <body>
    <h1>hello from <%= node['fqdn'] %></h1>

    <pre>
      <%= node['hostname'] %>
      <%= node['platform'] %> - <%= node['platform_version'] %>
      <%= node['memory']['total'] %> RAM
      <%= node['cpu']['total'] %> CPUs
    </pre>
  </body>
</html>
```



Reference

- https://docs.chef.io/
- https://supermarket.chef.io/

Questions?