Infraestrutura como Código

Chef

Michel Vasconcelos
michel.vasconcelos@gmail.com

Relembrando...

- Necessidade:
  - Tornar VM ou container aptos a executarem minhas aplicações

- Provisionamento vs Configuração

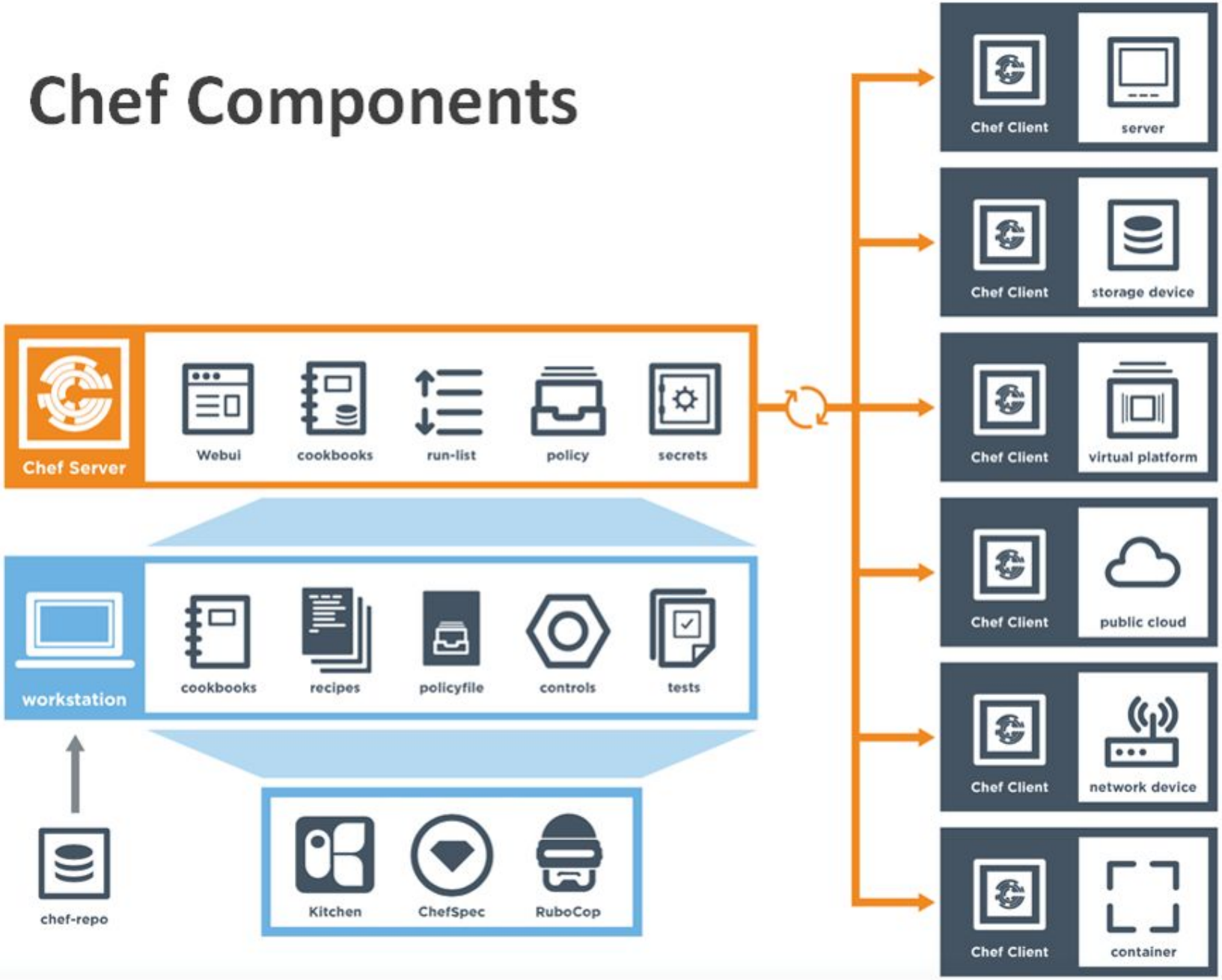
Chef

- Plataforma de automação de infraestrutura

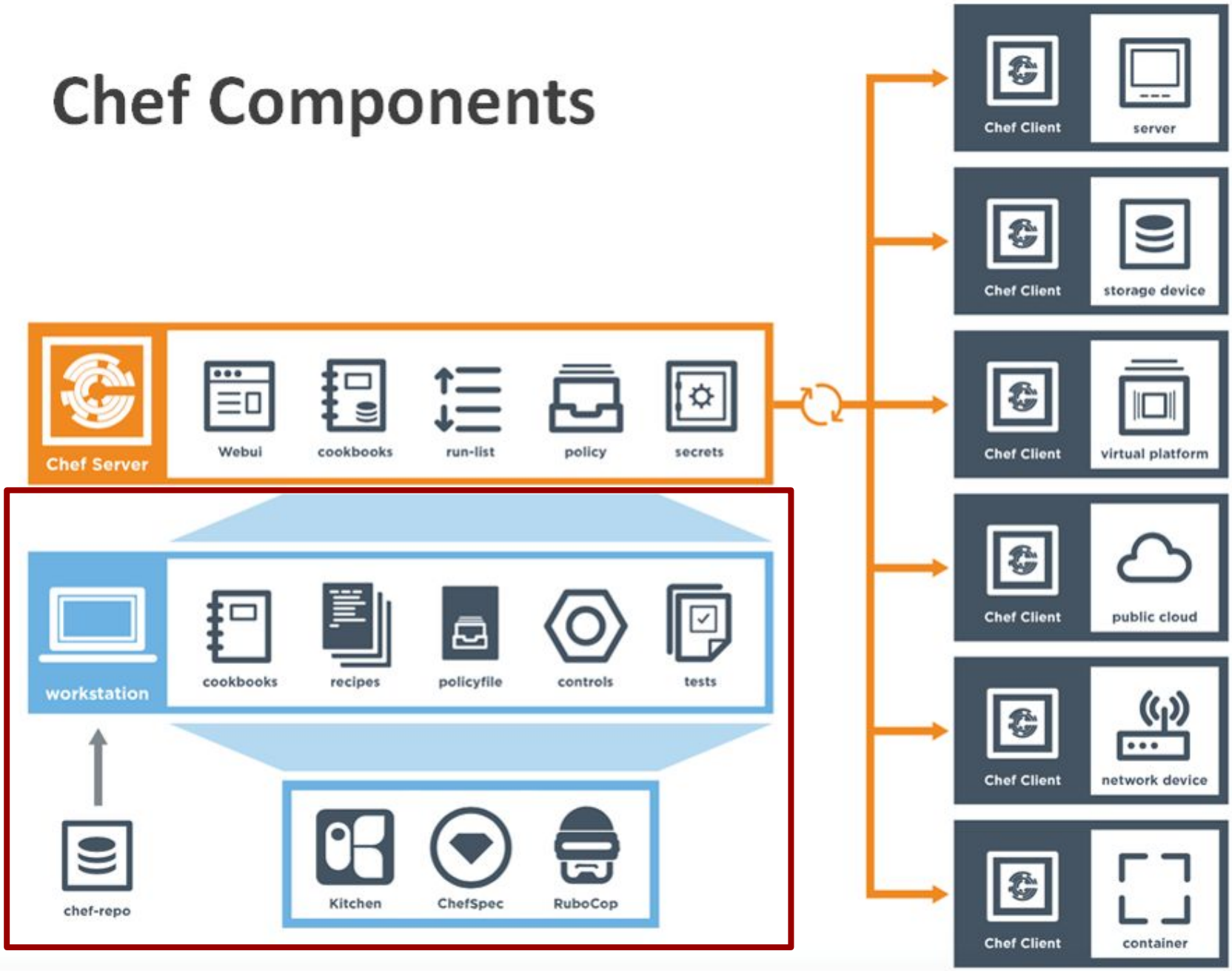
- Programática
  - Definições são escritas em Ruby

- Modelo Pull + Agentes
  - Há um servidor

Chef Components



Chef Components



Workstation

- Máquina que contém o kit de desenvolvimento e permite a interação com o servidor chef
  - Desenvolvimento de receitas (recipes) e livros de receita (cookbooks)
  - Sincronização de informações com o servidor chef
  - Repositório de código
  - (Eventualmente) Interação com os nós

- Todas as definições de infraestrutura podem (e devem) ser testadas por meio de ferramentas existentes na própria suite

Receitas (*Recipes*)

- Arquivos de definição de infraestrutura
  - Coleção de recursos, atributos e ações (até mesmo código)
- Modelo programático
  - Script Ruby
- Armazenados em livros de receita (*cookbooks*)

Livro de receitas (Cookbooks)

- Principal meio para distribuição de configurações e políticas

- Composto por:
  - atributos
  - bibliotecas
  - metadados
  - receitas
  - etc

- Armazenados em um repositório chef (chef-repo)

Lista de Execução (Run-List)

- Define a lista de etapas a serem executadas para que o nó chegue ao seu estado adequado
  - Configuração específica por nó

- Mantido por meio do knife e submetido ao servidor Chef

Repositório Chef (Chef-repo)

- Repositório de guarda de receitas, livros de receita, etc
  - repositório central vs repositório por cookbooks

- Pode ser armazenado em uma pasta local ou de rede

- Deve ser sincronizado com um sistema de controle de versão

Chef e Knife

- Interação com o servidor e repositório por linha de comando

- Chef
  - Relação com o repositório local
  - Criação de receitas, cookbooks, etc

- Knife
  - Interação com o servidor chef e nós

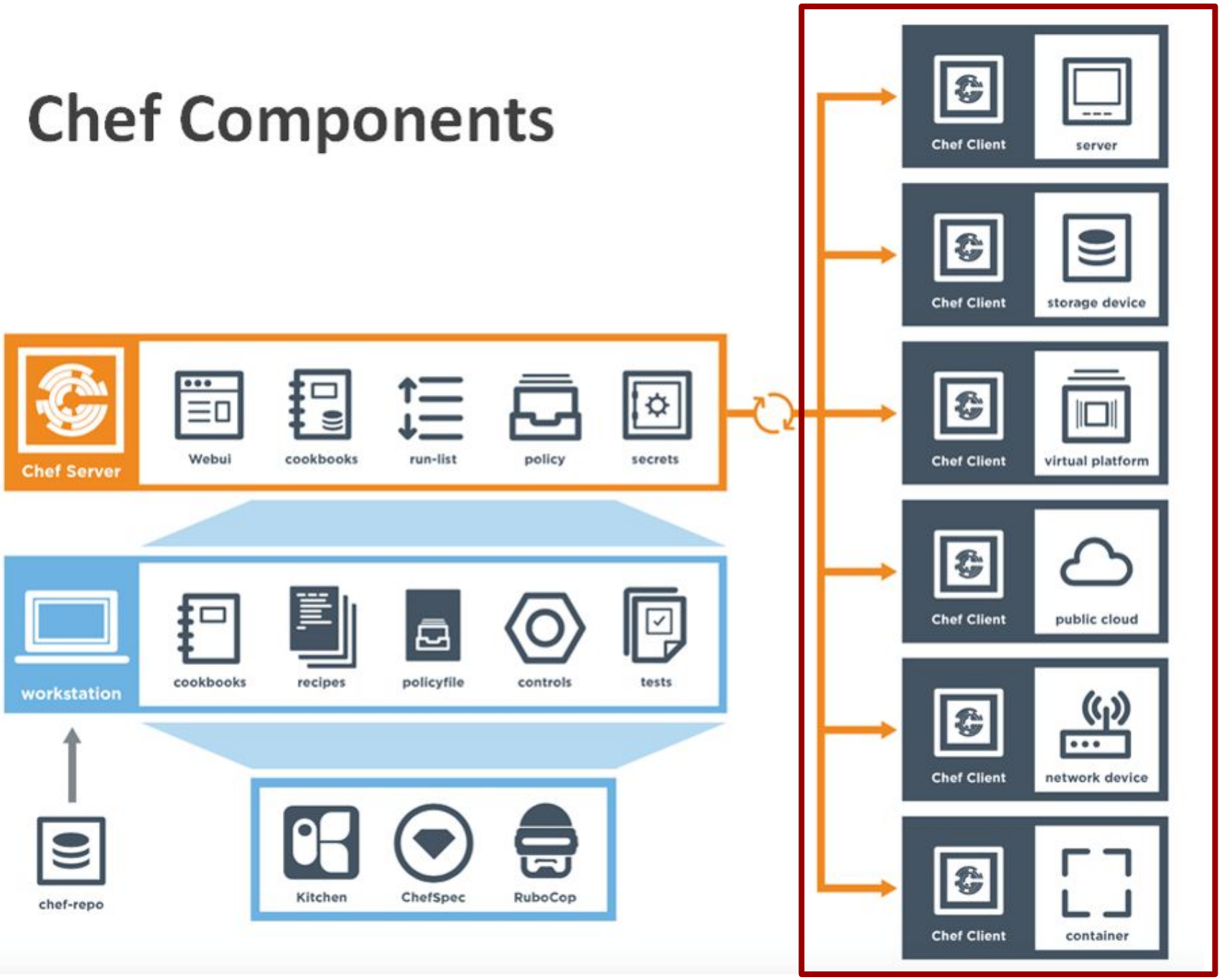
Outras ferramentas

RuboCop

Kitchen

ChefSpec

Chef Components



node

Nó

- Qualquer máquina gerenciada pelo Chef

server

cloud

container

virtual machine

network device

Chef-client

- Agente que é executado em cada máquina gerenciada pelo Chef

- Responsável por garantir registro e que o nós esteja no estado adequado

- Ações
  - Registro e autenticação
  - Sincronização de receitas
  - Avaliação do estado
  - Aplicação das mudanças no nó

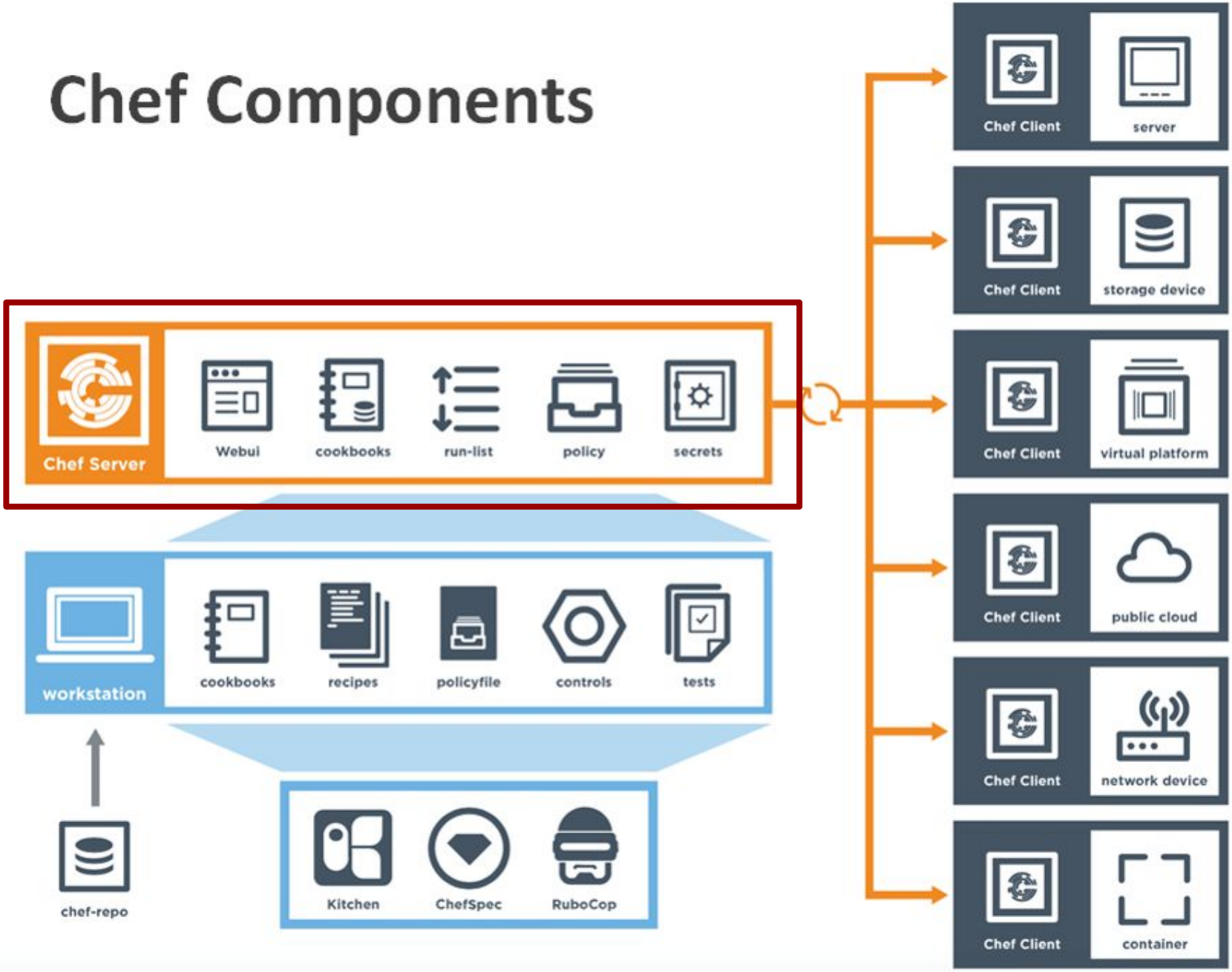
Ohai

- Ferramenta auxiliar que permite a coleta de informações, dados de configuração e estado do nó

- Possui relação direta com o chef-client

- Informações coletadas
  - OS
  - Rede
  - Memória
  - Disco
  - etc

Chef Components



Servidor Chef

- Local central de armazenamento de informações como: receitas, livros de receita, políticas de aplicação de mudanças, recursos, etc

- Além da interação com as *workstations*, o servidor chef também pode ser acessado via console de gerenciamento
  - data bags
  - políticas
  - recursos
  - etc

Servidor Chef

- O Chef também consegue operar em um modo simplificado para casos em que não há necessidade de servidor
  - Local Mode
  - Chef Solo
  - Chef Zero

Políticas

- Relacionam papéis e atribuições entre nós e a arquitetura de infraestrutura definida
  - Papéis
    - Padrão de operação de um nó (servidor web, servidor de aplicação)
  - Ambiente
    - Forma de mapear os perímetros organizacionais e as configurações no servidor chef (produção, dev, teste)
  - Versão
  - Lista de execução

Cookbooks, *Recipes e ChefDK*

Receitas (recipes)

- Principal elemento de configuração da ferramenta
  - Escrita em *ruby* (`*.rb`)
  - Define:
    - Estado desejado (recursos)
    - Informações necessárias
    - Como fazer (ações)
  - Armazenado em um cookbook
  - Pode relacionar-se com outras receitas (dependência ou referência)

- Sintaxe Ruby

Recurso

- Declaração de estado e quaisquer ações necessárias para se chegar a esse estado
  - Na prática: código Ruby contendo um tipo, um nome, propriedades e ações

```
type 'name' do
   attribute 'value'
   action :type_of_action
end
```

Recurso

- Declaração de estado e quaisquer ações necessárias para se chegar a esse estado
  - Na prática: código Ruby contendo um tipo, um nome, propriedades e ações

```
type 'name' do
   attribute 'value'
   action :type_of_action
end
```

```
package 'tar' do
  version '1.16.1'
  action :install
end
```

Importante

Em poucas palavras, uma receita é um conjunto de recursos definidos em um arquivo Ruby

Recursos mais utilizados

- `apt_package`
- `apt_update`
- `bash`
- `directory`
- `file`
- `service`
- `template`

APT_PACKAGE

- Gerencie pacotes em distribuições debian e ubuntu

```
apt_package 'name' do
  default_release
  notifies
  options
  overwrite_config_files
  package_name
  source
  subscribes
  timeout
  version
  action
end
```

APT_PACKAGE

- Gerencie pacotes em distribuições debian e ubuntu

```
apt_package 'name' do
  default_release
  notifies
  options
  overwrite_config_files
  package_name
  source
  subscribes
  timeout
  version
  action
end
```

```
apt_package 'jwhois' do
  action :install
  source '/path/to/jwhois.deb'
end
```

```
package 'apache2' do
  options '--no-install-recommends'
end
```

APT_UPDATE

- Mantém o repositório de pacotes atualizado

```
apt_update 'name' do
  frequency
  action
end
```

APT_UPDATE

- Mantém o repositório de pacotes atualizado

```
apt_update 'name' do
  frequency
  action
end
```

```
apt_update 'all platforms' do
  frequency 86400
  action :periodic
end
```

BASH

- Executa um script de linha de comando

```
bash 'name' do
  code
  creates
  cwd
  environment
  flags
  group
  notifies
  path
  returns
  subscribes
  timeout
  user
  umask
  action
end
```

BASH

- Executa um script de linha de comando

```
bash 'name' do
  code
  creates
  cwd
  environment
  flags
  group
  notifies
  path
  returns
  subscribes
  timeout
  user
  umask
  action
end
```

```
bash 'install_something' do
  user 'root'
  cwd '/tmp'
  code <<-EOH
  wget http://www.example.com/tarball.tar.gz
  tar -zxf tarball.tar.gz
  cd tarball
  ./configure
  make
  make install
  EOH
end
```

DIRECTORY

- Declara um diretório e suas permissões

```
directory 'name' do
  group
  inherits
  mode
  notifies
  owner
  path
  recursive
  rights
  subscribes
  action
end
```

DIRECTORY

- Declara um diretório e suas permissões

```
directory 'name' do
  group
  inherits
  mode
  notifies
  owner
  path
  recursive
  rights
  subscribes
  action
end
```

```
directory '/etc/apache2' do
  owner 'root'
  group 'root'
  mode '0755'
  action :create
end
```

FILE

- Gerencie arquivos diretamente em um nó

```
file 'name' do
  atomic_update
  backup
  checksum
  content
  force_unlink
  group
  inherits
  manage_symlink_source
  mode
  notifies
  owner
  path
  rights
  sensitive
  subscribes
  verify
  action
end
```

FILE

- Gerencie arquivos diretamente em um nó

```
file 'name' do
  atomic_update
  backup
  checksum
  content
  force_unlink
  group
  inherits
  manage_symlink_source
  mode
  notifies
  owner
  path
  rights
  sensitive
  subscribes
  verify
  action
end
```

```
file '/tmp/something' do
  owner 'root'
  group 'root'
  mode '0755'
  action :create
end
```

```
file '/tmp/something' do
  action :delete
end
```

SERVICE

- Gerencie um serviço diretamente em um nó

```
service 'name' do
  init_command
  notifies
  options
  pattern
  priority
  reload_command
  restart_command
  service_name
  start_command
  status_command
  stop_command
  subscribes
  supports
  timeout
  action
end
```

SERVICE

- Gerencie um serviço diretamente em um nó

```
service 'name' do
  init_command
  notifies
  options
  pattern
  priority
  reload_command
  restart_command
  service_name
  start_command
  status_command
  stop_command
  subscribes
  supports
  timeout
  action
end
```

```
service "tomcat" do
  action :start
end
```

```
service 'apache' do
  action [ :enable, :start ]
  retries 3
end
```

Outros recursos

- Há uma extensa lista de recursos e a descrição de uso no link abaixo:

https://docs.chef.io/resource.html

Livro de Receitas (cookbook)

- Unidade principal de configuração e aplicação de políticas

- Podemos encontrar em um cookbook:
  - atributos
  - receitas
  - arquivos
  - templates
  - Bibliotecas
  - Metadados
  - etc

Livro de Receitas (cookbook)

- Na prática, um livro de receitas é uma árvore de diretórios onde cada tipo de componente pode ser localizado em sua respectiva pasta
  - atributos na pasta attributes
  - receitas na pasta recipes
  - etc

- Importante
  - metadata.rb
    - Configurações gerais do cookbook (versão, mantenedores, etc)
  - Berksfile
    - Arquivo de definição de dependência entre receitas
  - kitchen.yml
    - Definição dos testes integrados da infraestrutura

ChefDK

- Pacote que contém tudo para que uma máquina atue como workstation ou nó
  - chef-client
    - Agente que executa nos nós registrados
  - chef
    - Executável que controla o fluxo de trabalho do administrador
  - knife
    - Executável que permite a interação com o sevidor Chef ou nós
    - Configurado por meio do arquivo `knife.rb`
  - Berkshelf
    - Gerenciador de dependência entre receitas

Fluxo de Trabalho

Fluxo de Trabalho

- O administrador da infraestrutura cria um repositório local, sincroniza com algum controle de versão e obtém as definições atuais da infra
  - `chef create cookbook <nome>`
  - `git add, commit, clone, etc`

Fluxo de Trabalho

- O administrador também pode obter receitas já existentes da comunidade (git) ou repositório oficial (supermarket.chef.io)
    - `knife cookbook site [download | install]`

Fluxo de Trabalho

- O administrador atualiza suas definições de infraestrutura e as submete ao servidor
  - `knife upload [cookbook]`

Fluxo de Trabalho

- Caso deseje operar no modo solo, você pode utilizar diretamente o agente com o parâmetro --local-mode
  - `chef-client --local-mode [cookbook]`

Fluxo de Trabalho

- O Servidor Chef atualiza as informações em seu repositório
  - As mudanças são associadas aos diversos nós clientes conforme registro prévio

Fluxo de Trabalho

- Os clientes buscam por possíveis atualizações
  - Um agente instalado busca por novas atualizações
  - Ele monta uma representação do objeto em memória coletando os dados do *ohai*

Fluxo de Trabalho

- Os clientes aplicam as atualizações necessárias
  - O agente compara a representação existente e as informações do servidor chef aplicando aquelas que são necessárias

Fluxo de Trabalho

- Os clientes atualizam as informações no servidor
  - Os clientes reportam as modificações realizadas para que o servidor contenha a informação mais atual do cliente

Fluxo de Trabalho

- Você também pode disparar a execução do chef-client diretamente por meio de ssh ou comando knife
  - **`knife ssh SEARCH_QUERY SSH_COMMAND`**

Exercício 01

Crie um cookbook para servidor web e o aplique localmente

Há um guia no repositório git da disciplina:

https://github.com/michelav/espec-iac/blob/master/chef/ex01.md

Configurando o Servidor Chef e Nós

Nós

- O processo de configuração de um nó para gerenciamento é chamado de *bootstraping*

- Modos de instalação
  - URL
    - `curl -L https://omnitruck.chef.io/install.sh | sudo bash`
  - Knife bootstrap

Nós

- Knife Bootstrap
  - `knife bootstrap [IP] [Opções]`

- Na verdade, esse comando é um atalho para as etapas manuais de acesso via ssh e instalação via URL
  - O Terraform possui um recurso que faz a instalação diretamente

- Importante
  - Você deve configurar o `knife.rb` antes

Servidor Chef

- Modos de Operação
  - Hosted (AWS, Opscode, etc)
  - On-Premise Standalone
  - On-Premise Alta Disponibilidade

https://manage.chef.io/

Exercício 02

Crie uma na Chef e utilize um servidor hospedado para configurar um nó

Há um guia no repositório git da disciplina:

https://github.com/michelav/espec-iac/blob/master/chef/ex02.md

Exercício 03

Modifique a página padrão do servidor web e atualize o cookbook.

Configure o nó para que as atualizações sejam feitas automaticamente (chef-client executando periodicamente)

Utilize as orientações existentes no site do Chef e do Terraform para tal.