Automação do Ambiente

Prof. Matheus Sousa Faria

Automação do Ambiente

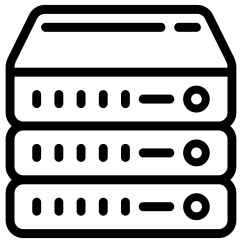
- IaC (Infrastructure as Code)
- Descrever em código:
  - Dependências
  - Configurações
  - Ordem de instalação
  - ...
- A equipe vai ter o mesmo ambiente
  - Configuração
- Documentar o seu ambiente
- Incorporar rapidamente mudanças
  - Configuração acompanha o repositório
- Acabar com tutoriais e “scripts”
  - Só o fulano sabe mexer no script xpto

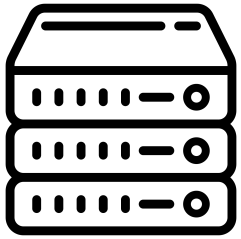
Sem automação

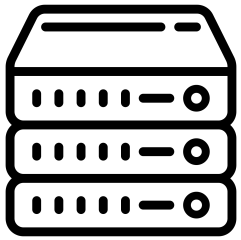
Infra

Manual passo a passo

1. …..
2. …..
3. …..
4. ……
5. ……
6. ……
7. …..
8. …..
9. …..
10. …
11. ..
12. ……







Sem automação

Infra

Manual passo a passo

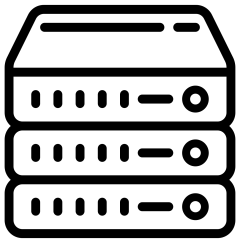
1. …..
2. …..
3. …..
4. ……
5. ……
6. ……
7. …..
8. …..
9. …..
10. …
11. ..
12. ……

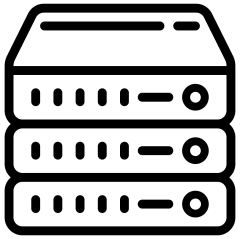
Errou o passo

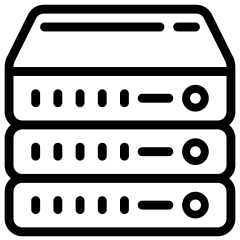
Esqueceu o passo

Confundiu as máquinas

Erro humano







Sem automação

Infra

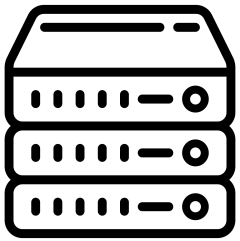
Manual passo a passo

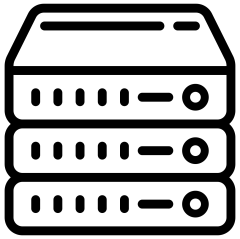
1. …..
2. …..
3. …..
4. ……
5. ……
6. ……
7. …..
8. …..
9. …..
10. …
11. ..
12. ……

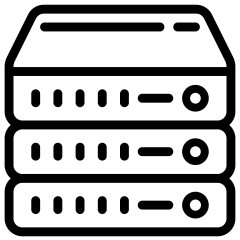
Formata tudo

Começa do zero

+3h de trabalho







Sem automação

Programador

Manual passo a passo

1. .....
2. .....
3. .....
4. .....
5. .....
6. .....
7. .....
8. .....
9. .....
10. ...
11. ..
12. .....

Manual passo a passo

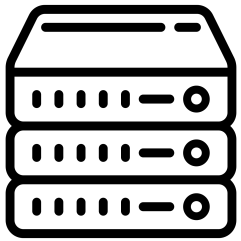
1. .....
2. .....
3. .....
4. .....

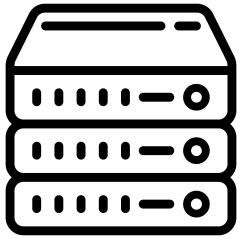
script.sh

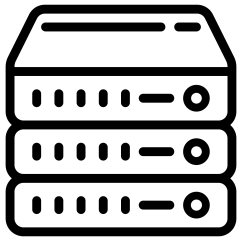
script.sh

script.sh

script.sh







Sem automação (semi-automação)

Infra

Facilitou o trabalho

Diminuiu a possibilidade de erros

Manual passo a passo

1. .....
2. .....
3. .....
4. ......

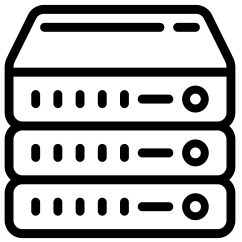
script.sh

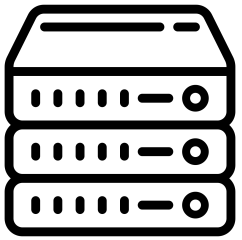
script.sh

script.sh

script.sh

1h de trabalho





Sem automação (semi-automação)

Infra

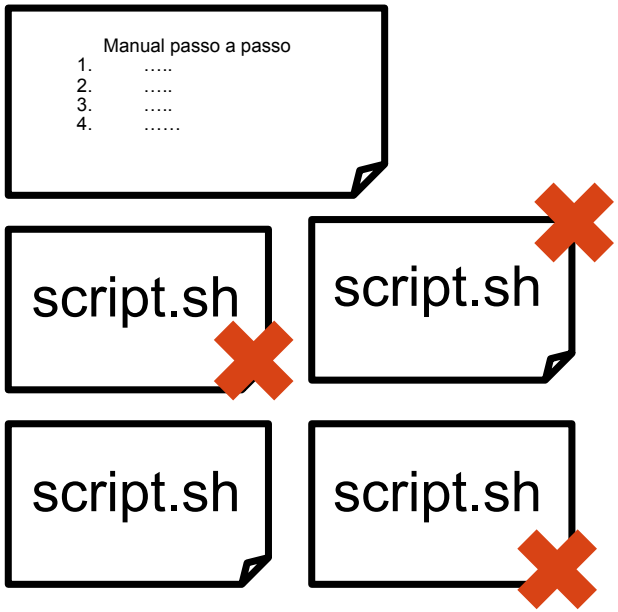
Programador
+
Infra

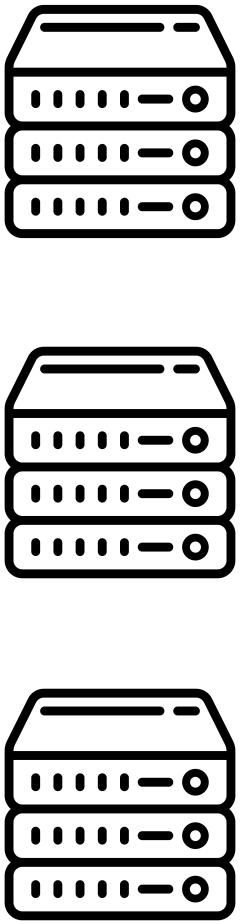
Sistema atualizou

Dependencia X atualizou

Mudamos a configuração Y

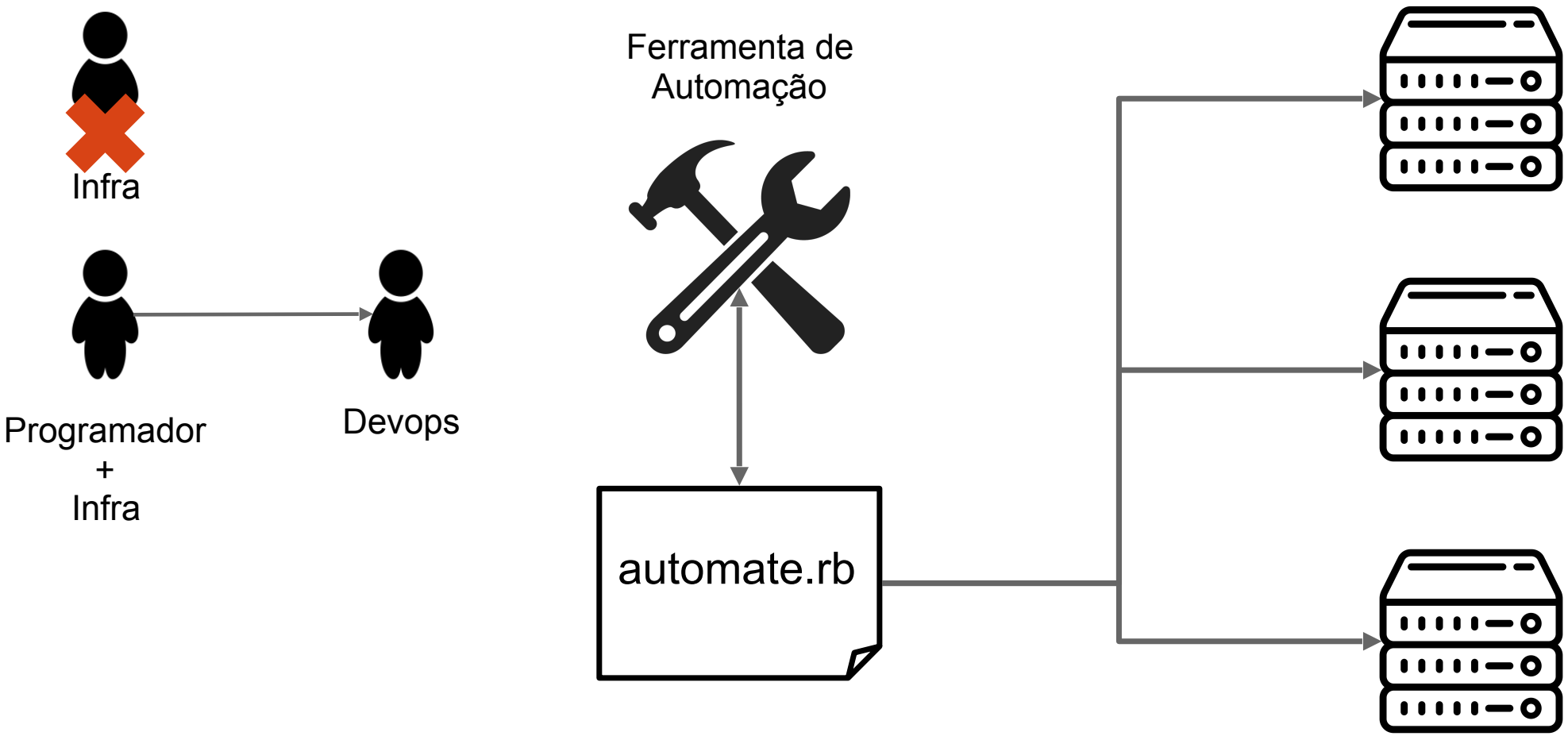
Dependência de múltiplas equipes



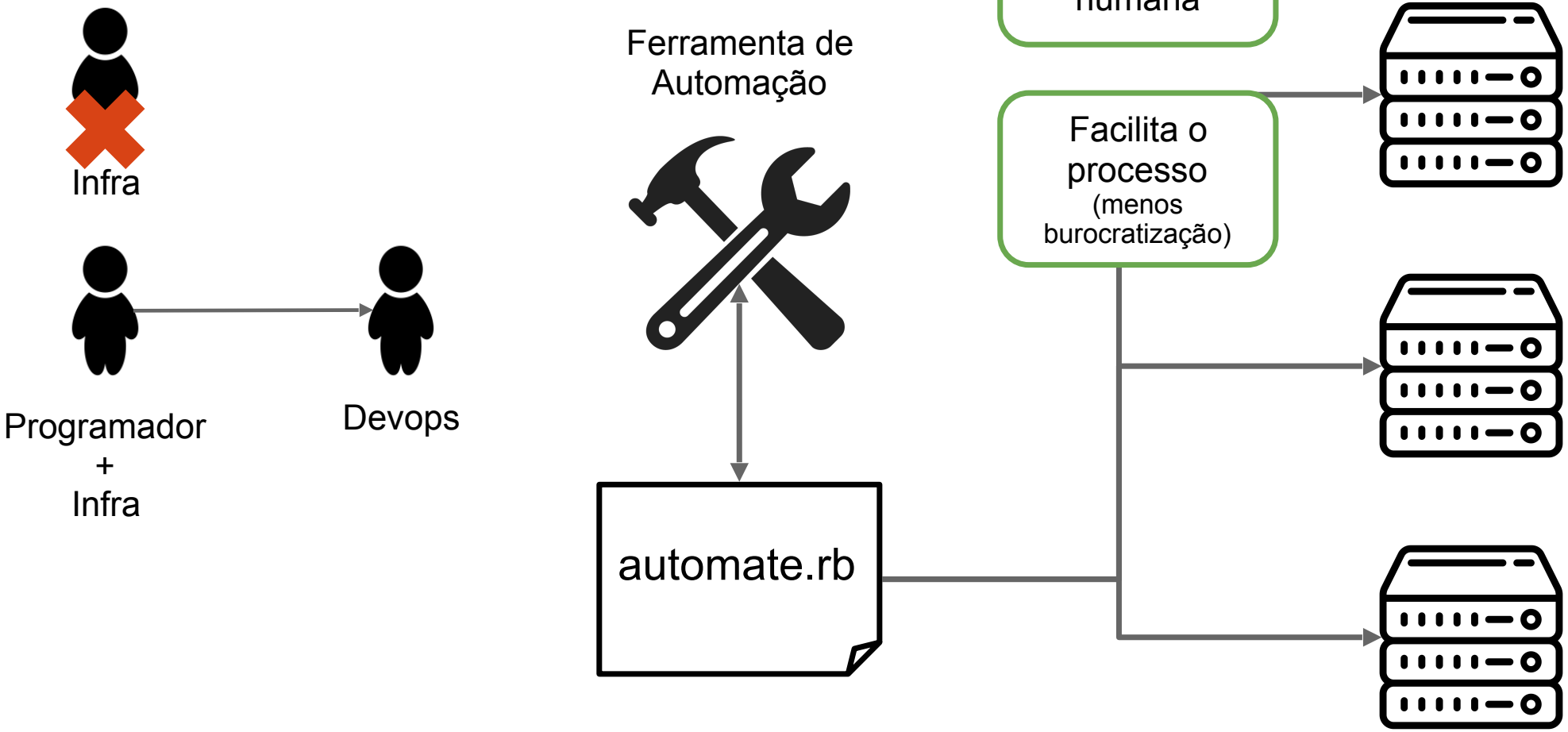


Precisamos de uma ferramenta que vá evitar esta dependência

Com Automação



Com Automação



Benefícios

1. Menos suscetível a erro
    a. Humano
    b. Erros de programação são testáveis
2. Múltiplos ambientes:
    a. Desenvolvimento
    b. Produção
    c. Teste
3. Modularização do ambiente
    a. Front
    b. Back
    c. Banco
4. Deploy Automático
    a. Rápido
    b. Facilitado

Benefícios

1. Menos suscetível a erro
   a. Humano
   b. Erros de programação são testáveis
2. Múltiplos ambientes:
   a. Desenvolvimento
   b. Produção
   c. Teste
3. Modularização do ambiente
   a. Front
   b. Back
   c. Banco
4. Deploy Automático
   a. Rápido
   b. Facilitado

- Erros de digitação
- Pular passos
- Executar passos fora do roteiro
  - Stackoverflow
  - Google

- Melhora a reprodução e verificação de bugs
- Testes automatizados para verificar

Benefícios

1. Menos suscetível a erro
   a. Humano
   b. Erros de programação são testáveis
2. Múltiplos ambientes:
   a. Desenvolvimento
   b. Produção
   c. Teste
3. Modularização do ambiente
   a. Front
   b. Back
   c. Banco
4. Deploy Automático
   a. Rápido
   b. Facilitado

- Separação e especialização dos ambientes
- Dados sensíveis
  - Senhas
  - Acesso aos bancos
- Facilita a entrada de integrantes na equipe
- Representa melhor o ciclo de vida da aplicação

Benefícios

1. Menos suscetível a erro
   a. Humano
   b. Erros de programação são testáveis
2. Múltiplos ambientes:
   a. Desenvolvimento
   b. Produção
   c. Teste
3. Modularização do ambiente
   a. Front
   b. Back
   c. Banco
4. Deploy Automático
   a. Rápido
   b. Facilitado

- Saber quais partes do ambiente necessitam:
  - Configuração
  - Dependências
- Duas partes dependem das mesmas coisas
- Programador tem o ambiente que precisa
- Facilita testes de integração

Benefícios

1. Menos suscetível a erro
    a. Humano
    b. Erros de programação são testáveis
2. Múltiplos ambientes:
    a. Desenvolvimento
    b. Produção
    c. Teste
3. Modularização do ambiente
    a. Front
    b. Back
    c. Banco
4. Deploy Automático
    a. Rápido
    b. Facilitado

- Automação do deploy em CI
- Deploys
    - Nightly
    - Semanais
- Não consome o tempo da equipe
- Mantém o sistema atualizado
- Tem certeza que o deploy está funcionando
- Continuous Deploy

Ferramentas de Automação

CHEF

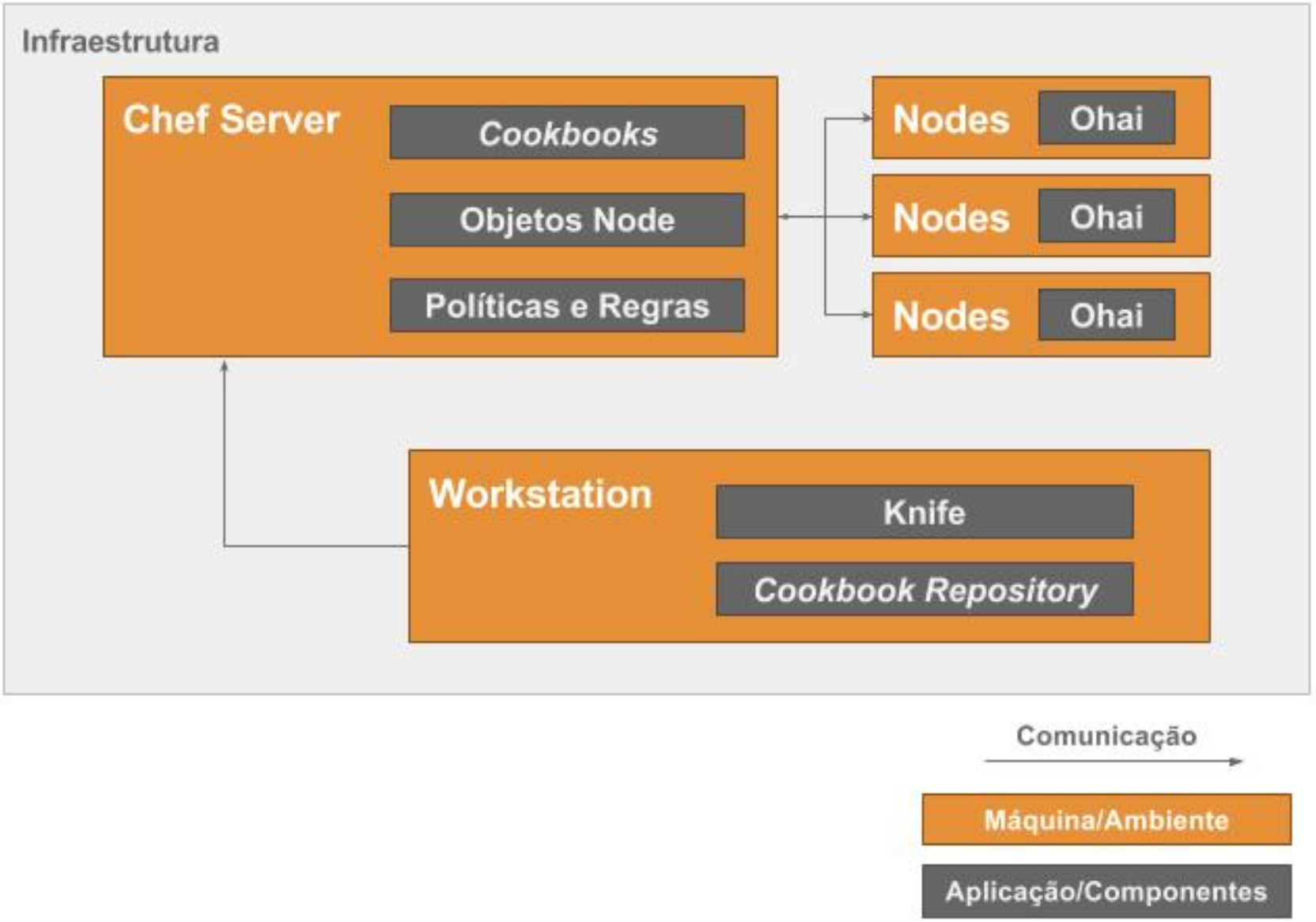
- “Linguagem de Configuração”
- Engine de automação remota
  - Controle a partir do nó principal (host)
- Automatiza
  - Configuração
  - Instalação
  - Comandos customizados
  - Templates
- Fácil de ler e entender
- Não requer código
  - Na medida do possível
- Não requer configuração dos targets, só do host (agentless)

- Componentes Principais
  - Workstation (criação dos scipts)
  - Chef Server (Armazena as configurações e as distribui para os Nós)
  - Nodes (Máquina que roda o código Chef e é configurada por ele.

- Chef *resources*
  - Package
  - File
  - Service

- Cookbook
- Recipes (Podem chamar outros *recipes*)


CHEF


Infraestrutura
Chef Server
Cookbooks
Objetos Node
Políticas e Regras
Nodes
Ohai
Nodes
Ohai
Nodes
Ohai
Workstation
Knife
Cookbook Repository
Comunicação
Máquina/Ambiente
Aplicação/Componentes

CHEF

- Comandos
  - chef –help
  - chef generate –help
  - chef generate cookbook cookbooks/apache
  - chef generate recipe cookbooks/apache server
  - sudo chef-client --local-mode file.rb
  - sudo chef-client --local-mode --runlist "cookbook[recipe::name]"
  - sudo chef-client -z -r “cookbook[recipe::name]"


ANSIBLE

- Mais popular
- “Linguagem de Configuração”
- Engine de automação remota
  - Controle a partir do nó principal (host)
- Automatiza
  - Configuração
  - Instalação
  - Comandos customizados
  - Templates
- Fácil de ler e entender
- Não requer código
  - Na medida do possível
- Não requer configuração dos targets, só do host (agentless)

Inventory

- Quem são os meus targets
  - Quais máquinas que eu posso controlar remotamente
  - Acesso SSH permitido
- /etc/ansible/hosts

```
[web]
192.168.1.10

[dbservers]
db01.intranet.mydomain.net
db02.intranet.mydomain.net
0.25.1.56
10.25.1.57
```

Comandos / Modules

- Comandos isolados para targets específicos
- Não é uma boa prática
  - Tudo deve ser feito no playbook
- Executar o comando ping

```
ansible all -m ping
```

```
ansible all -m ping -s -u vagrant
```

Playbook

- Descreve um roteiro a ser seguido
- Orquestração
- YAML
- Possui:
  - Tasks
  - Vars
  - Handles
- Descreve também:
  - Permissões de Execução
  - O uso de módulos
  - Uso de roles
  - Targets escolhidos

Playbook - Hosts

```
---
- hosts: local
  tasks:
   - name: Install Nginx
     apt: pkg=nginx state=installed
update_cache=true
```

- Hosts: máquinas em que o playbook vai rodar
  - all seria o valor para rodar em todas

Playbook - Tasks

```
---
- hosts: local
  tasks:
   - name: Install Nginx
     apt: pkg=nginx state=installed
update_cache=true
```

- Lista de tarefas com:
  - name - Descrição legível da tarefa
  - module - Módulo que a tarefa vai utilizar

Playbook - Executando como root

```
---
- hosts: local
  tasks:
   - name: Install Nginx
     apt: pkg=nginx state=installed
update_cache=true
     become: true
```

- O become ativa a execução como o root por padrão

Playbook - Executando como outro usuário

```
---
- hosts: local
  tasks:
   - name: Install Nginx
     apt: pkg=nginx state=installed
update_cache=true
     become: true
     become_user: matheusfaria
     become_method: sudo
```

Playbook - Handlers

```
---
- hosts: local
  tasks:
   - name: Install Nginx
     apt: pkg=nginx state=installed update_cache=true
     notify:
        - Handler de Restart

  handlers:
   - name: Handler de Restart
     service: name=nginx state=started
```

São tasks chamadas ao fim de outras tasks

Playbook - Comando para rodar

```
ansible-playbook playbook.yml
```

Assim como comandos, usuário e outras opções podem ser escolhidos

```
ansible-playbook -s -u vagrant nginx.yml
```

Roles

- Aglomerado de tasks, arquivos e configurações específicas de uma parte
- Organizam a automação
- Facilitam a criação de playbooks muito grandes

```
├── roles
|   └── nginx
|       ├── files
|       ├── handlers
|       |   └── main.yml
|       ├── meta
|       ├── tasks
|       |   └── main.yml
|       ├── templates
|       └── vars
```

Roles - Pastas

- **Files:** arquivos que vão ser copiados
- **Handlers:** guarda os handlers
- **Meta:** lista de dependencia de outros roles
- **Tasks:** guardam as tasks
- **Templates:** arquivos que vão ser modificados e copiados
- **Vars:** variáveis a serem utilizadas neste role

```
├── roles
│   └── nginx
│       ├── files
│       ├── handlers
│       │   └── main.yml
│       ├── meta
│       ├── tasks
│       │   └── main.yml
│       ├── templates
│       └── vars
```

Roles - Uso em playbooks

- Playbook “esqueleto” para um ou mais roles

```
---
- hosts: droplets
  roles:
    - role: nginx
```

Integração com o vagrant

- Atua como provisionador
- Prepara o ambiente com Ansible no vagrant provision
- Adicione no Vagrantfile:

```
  config.vm.provision "ansible" do |
ansible|
    ansible.playbook = "playbook.yml"
  end
```

Perguntas?