

Infraestrutura como Código

Michel Vasconcelos
michel.vasconcelos@gmail.com

Agenda

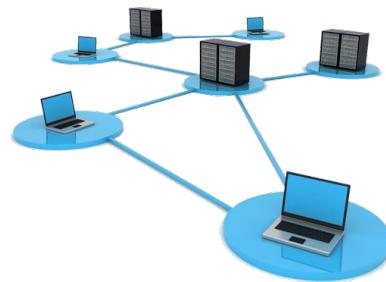
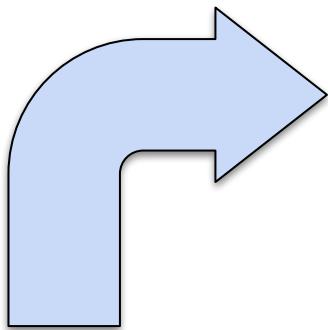
- Introdução
 - Motivação
 - Princípios e Conceitos
- Atividades
 - Provisionamento
 - Configuração
- Servidores
 - Ciclo de vida
 - Padrões e Práticas

Atualmente

- Processos ágeis
 - Entrega e implantação contínuas
 - Janela mínima de indisponibilidade
- Aplicações para Internet
 - Escalabilidade
 - Confiabilidade
 - etc
- Frameworks, Arquiteturas e Tecnologias
 - Cloud
 - Virtualização
 - Microserviços

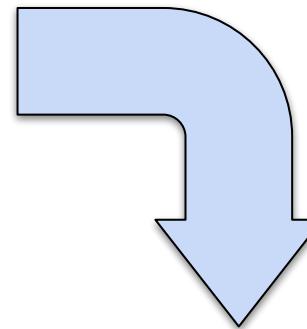
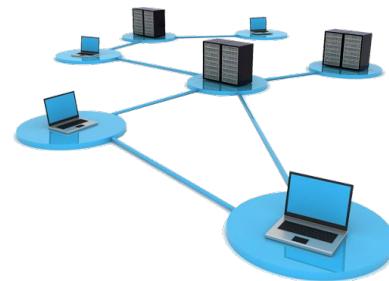
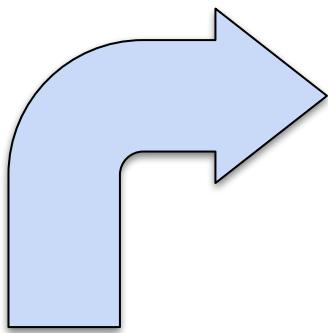
```
:attachEvent("onreadystatechange",H),e.attachC  
oolean Number String Function Array Date RegExp  
,_=();function F(e){var t=_[e]=();return b.ea  
t[i]==!=!1&&e.stopOnFalse){r=!1;break}n=!1,u&  
r=u.length:r&&(s=t,c(r))}return this},remove  
ction(){return u=[],this},disable:function()  
re:function(){return p.fireWith(this,argument  
ending"},r={state:function(){return n},always:  
romise)?e.promise().done(n.resolve).fail(n.re  
id(function(){n=s}),t[i^e][2].disable,t[2][2]  
.e0,n=h.call(arguments),r=n.length,i=1==r|!e&  
(r),l=Array(r);r>t;t++)n[t]&&b.isFunction(n[t  
><table><a href='/a'>a</a><input type='TagName("input")[0],r.style.cssText="top:1px  
est(r.getAttribute("style")),hrefNormalized:
```

DEV



```
attachEvent("onreadystatechange",H),e.attachC
oolean Number String Function Array Date RegE
,_={};function F(e){var t=_[e]=();return b.ea
t[i]==!=!1&&e.stopOnFalse){r=!1;break}n!=1,u&
r=u.length:r&&(s=t,c(r))}return this},remove
ction(){return u=[],this},disable:function()
re:function(){return p.fireWith(this,argument
"ending"},r={state:function(){return n},always:
romise)?e.promise().done(n.resolve).fail(n.re
id(function(){n=s}),t[i^e][2].disable,t[2][2].
e0,n=h.call(arguments),r=n.length,i=1!=r|j&&
(r),l=Array(r);r>t;t++)n[t]&&b.isFunction(n[t
]><table><a href='/a'>a</a><input type
'TagName("input")[0],r.style.cssText="top:1px
est(r.getAttribute("style")),hrefNormalized:
```

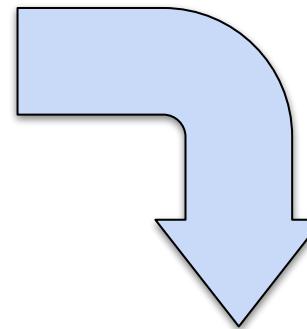
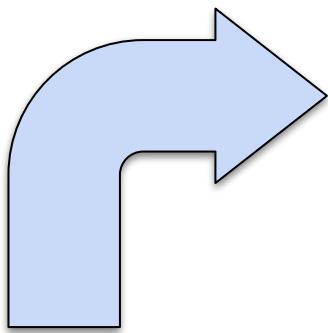
SERVS.



```
attachEvent("onreadystatechange",H),e.attachC  
oolean Number String Function Array Date RegE  
_={};function F(e){var t=_[e]=();return b.ea  
t[i]==!=!1&&e.stopOnFalse){r=!1;break}n!=1,u&  
?o=u.length:r&&(s=t,c(r))}return this},remove  
ction(){return u=[],this},disable:function()  
re:function(){return p.fireWith(this,argument  
ending"},r={state:function(){return n},always:  
romise)?e.promise().done(n.resolve).fail(n.re  
id(function(){n=s}),t[i^e][2].disable,t[2][2].  
=0,n=h.call(arguments),r=n.length,i=1!=r|j&  
&(r),l=Array(r);r>t;t++)n[t]&&b.isFunction(n[t  
><table><a href='/a'>a</a><input type  
'TagName("input")[0],r.style.cssText="top:1px  
est(r.getAttribute("style")),hrefNormalized:
```

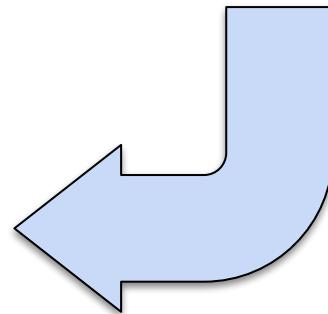
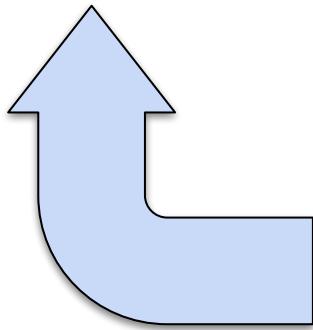
CONF.





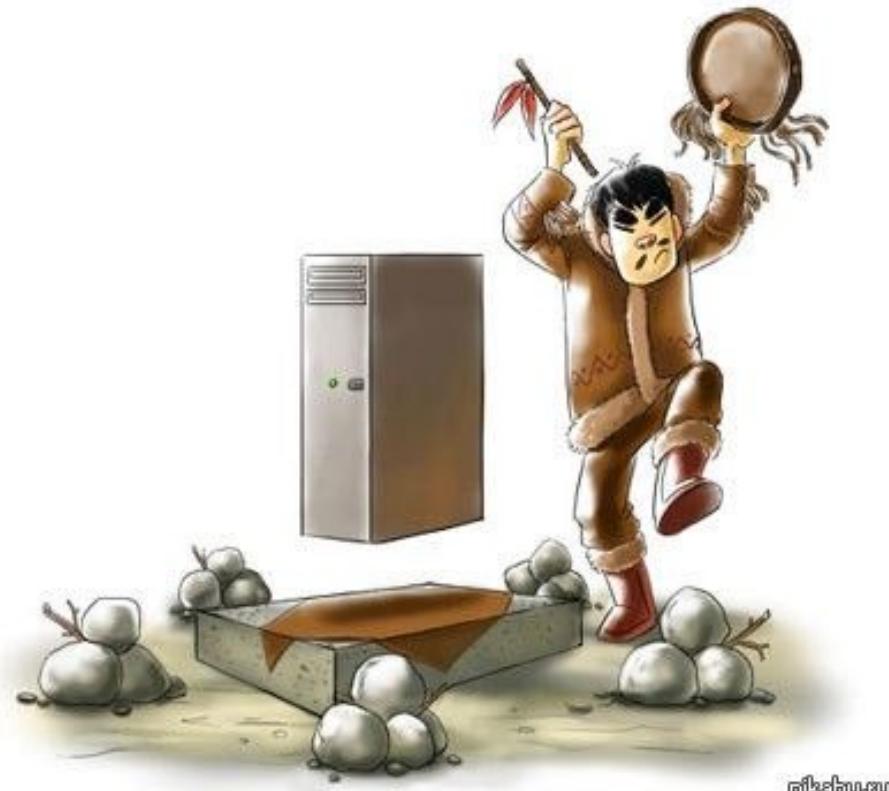
```
attachEvent("onreadystatechange",H),e.attachC  
oolean Number String Function Array Date RegE  
_={};function F(e){var t=_[e]=();return b.ea  
t[i]==!=!1&&e.stopOnFalse){r=!1;break}n!=1,u&  
?o=u.length:r&&(s=t,c(r))}return this},remove  
ction(){return u=[]},this},disable:function()  
re:function(){return p.fireWith(this,argument  
ending"},r={state:function(){return n},always:  
romise)?e.promise().done(n.resolve).fail(n.re  
id(function(){n=s},t[1]^e[2].disable,t[2][2].  
=0,n=h.call(arguments),r=n.length,i=1==r|!e&  
(r),l=Array(r);r>t;t++)n[t]&&b.isFunction(n[t  
><table></table><a href='/a'>a</a><input type  
'TagName('input')[0],r.style.cssText="top:1px  
est(r.getAttribute('style')),hrefNormalized:
```

IMPL.

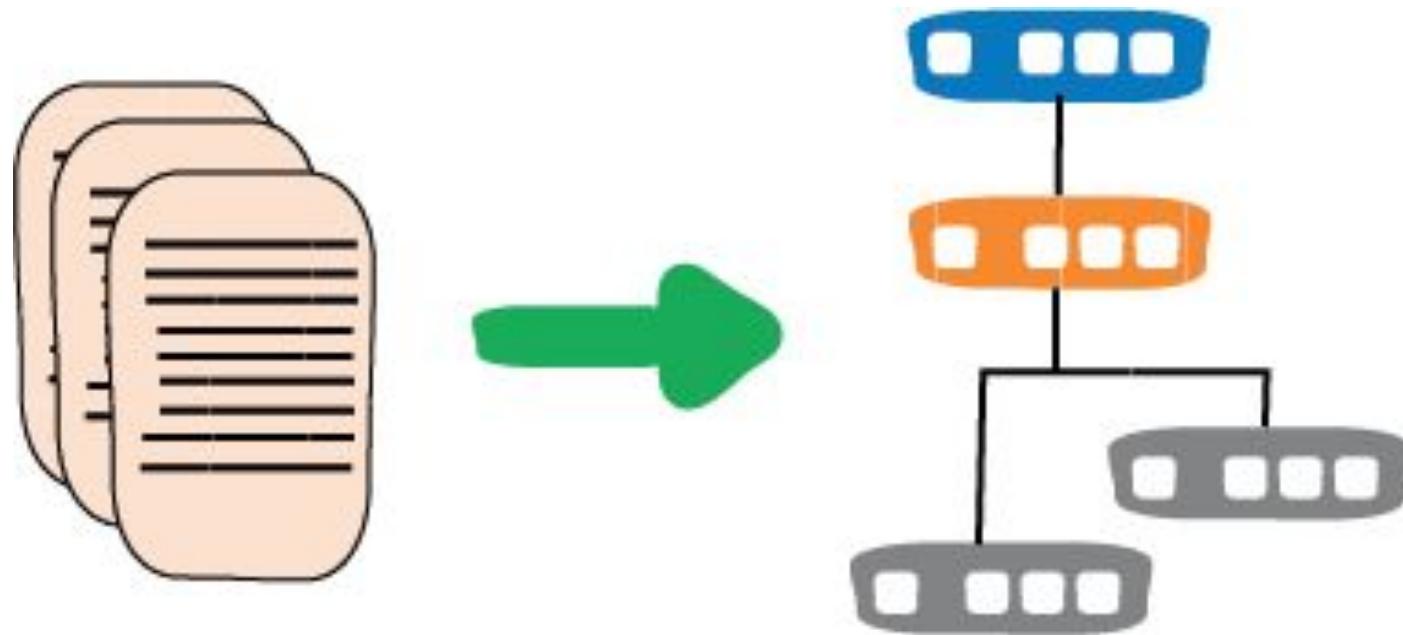


Resumindo...

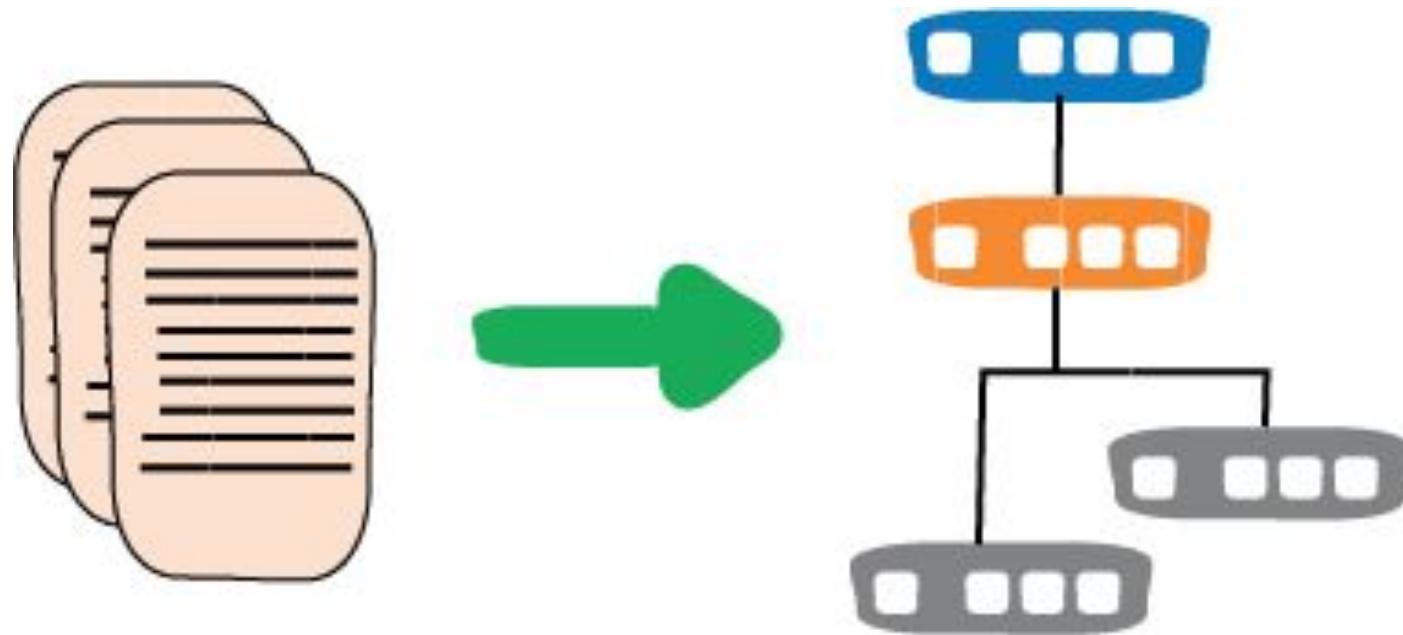
- Preparação manual do ambiente
- Implantação desconexa do desenvolvimento
 - Devs x Ops
- Ambiente não reproduzível
 - “Alguém sabe o que está instalado naquele servidor?”



Como resolver esse problema?



Infraestrutura como código



E o que é infraestrutura como código?!?!

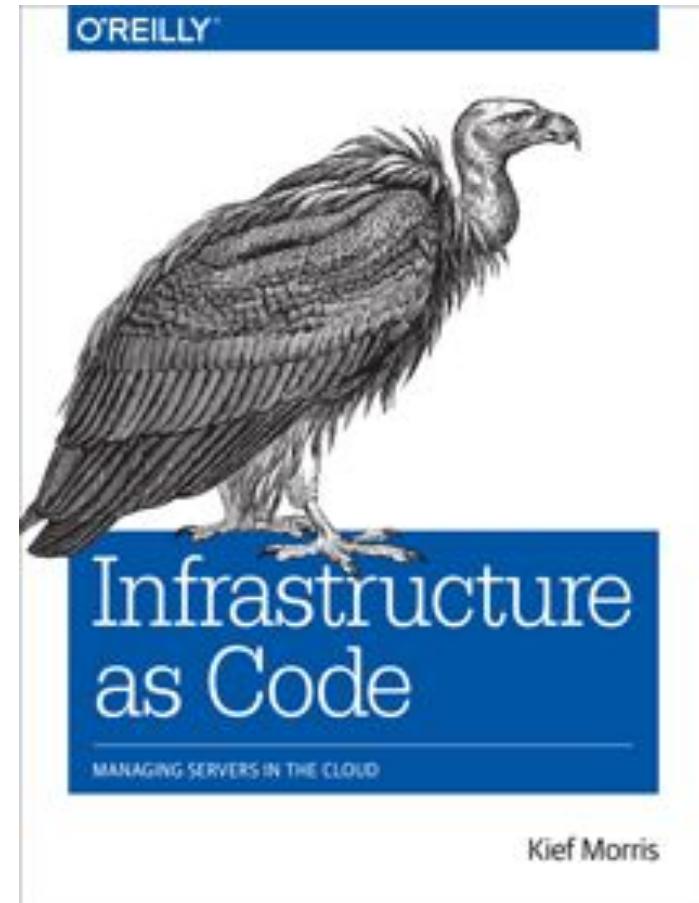
IaC é uma abordagem que permite definir elementos computacionais (servidores, rede, storage, etc) por meio de código e, portanto podem ser tratados como um artefato de software.

Fowler

IaC é uma abordagem que permite construir e gerenciar uma infraestrutura de forma dinâmica.

A infraestrutura, suas ferramentas e serviços são tratados como software, podendo ser controlados, testados e processados de maneira similar a uma aplicação.

Morris



Benefícios e Características

- Aplicação e infraestrutura disponíveis em uma mesma linha de base
- Infraestrutura testável
- Automação
 - Porém, IaC não se resuma só a isso
- Infra “facilmente” reproduzível

Princípios

- Reproduzível
- Consistência
- Descartabilidade
- Continuidade de serviços
- Testabilidade
- Documentável
- Versionável

Reproduzível



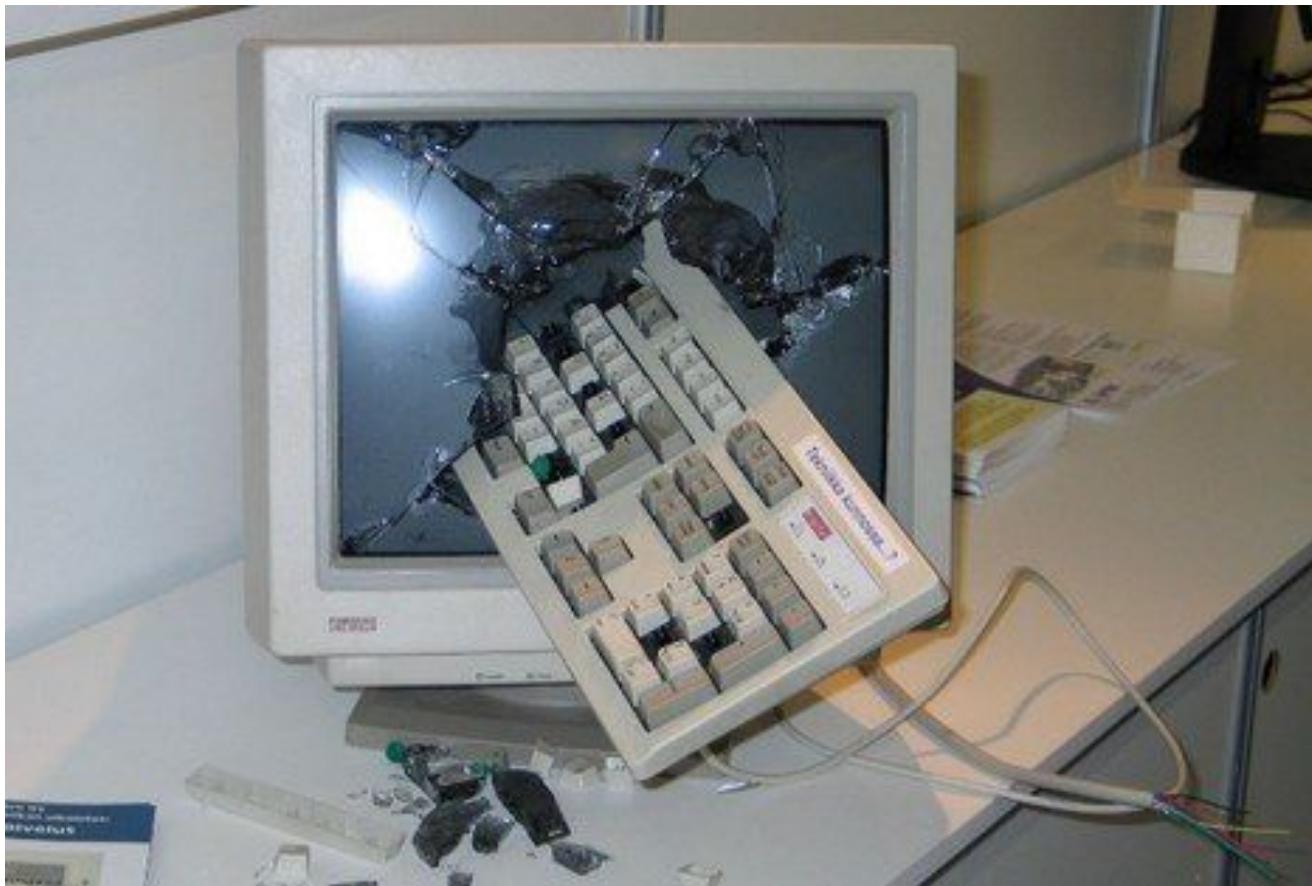
Deve ser possível reconstruir (com pouquíssimo esforço) qualquer elemento de sua infraestrutura

Consistência



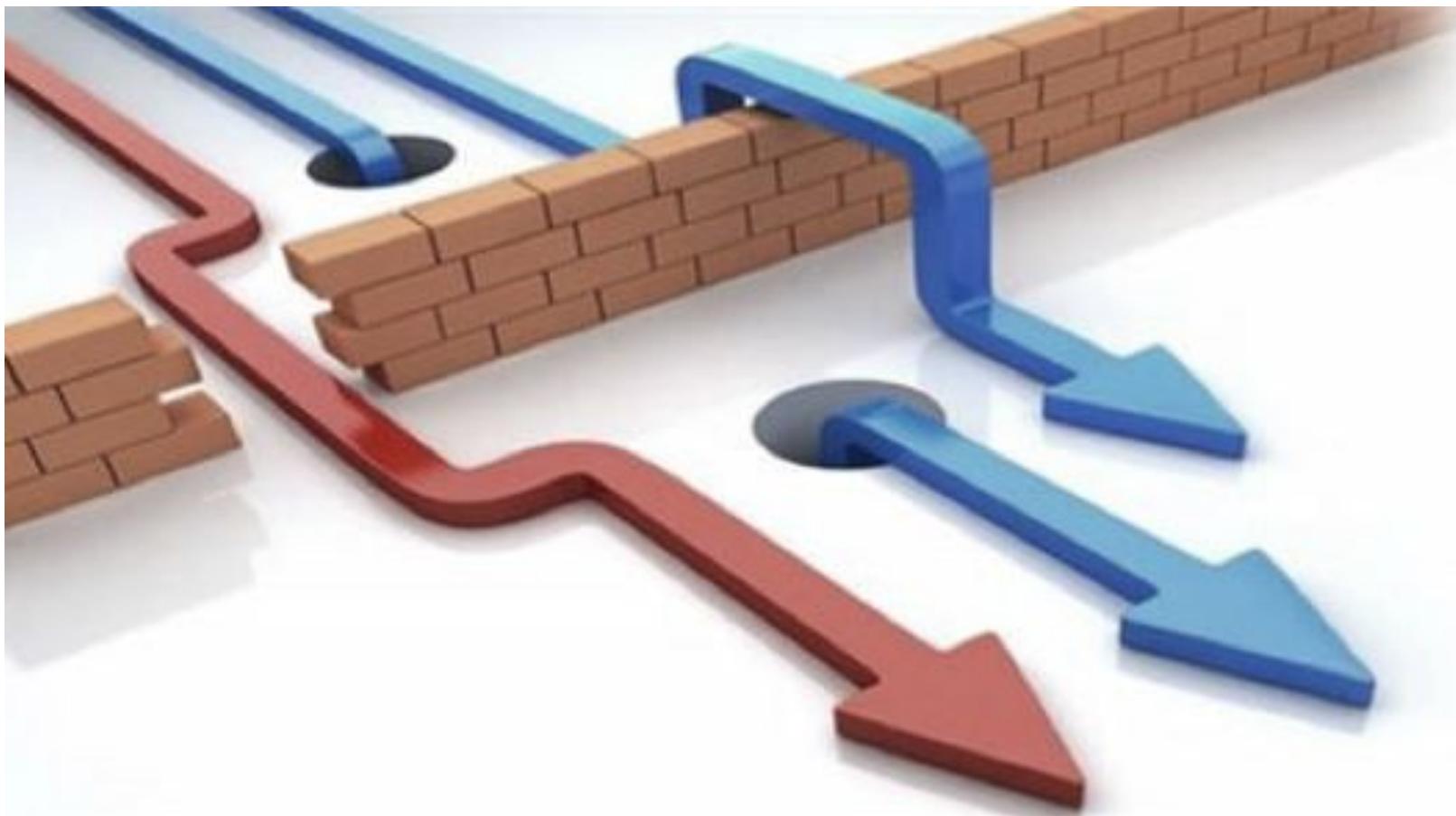
Se dois elementos de sua infra fornecem o mesmo serviço, eles devem ser (praticamente) idênticos!

Descartabilidade



Qualquer elemento de sua infra pode e será destruído em algum momento sem que ninguém perceba!

Continuidade dos serviços



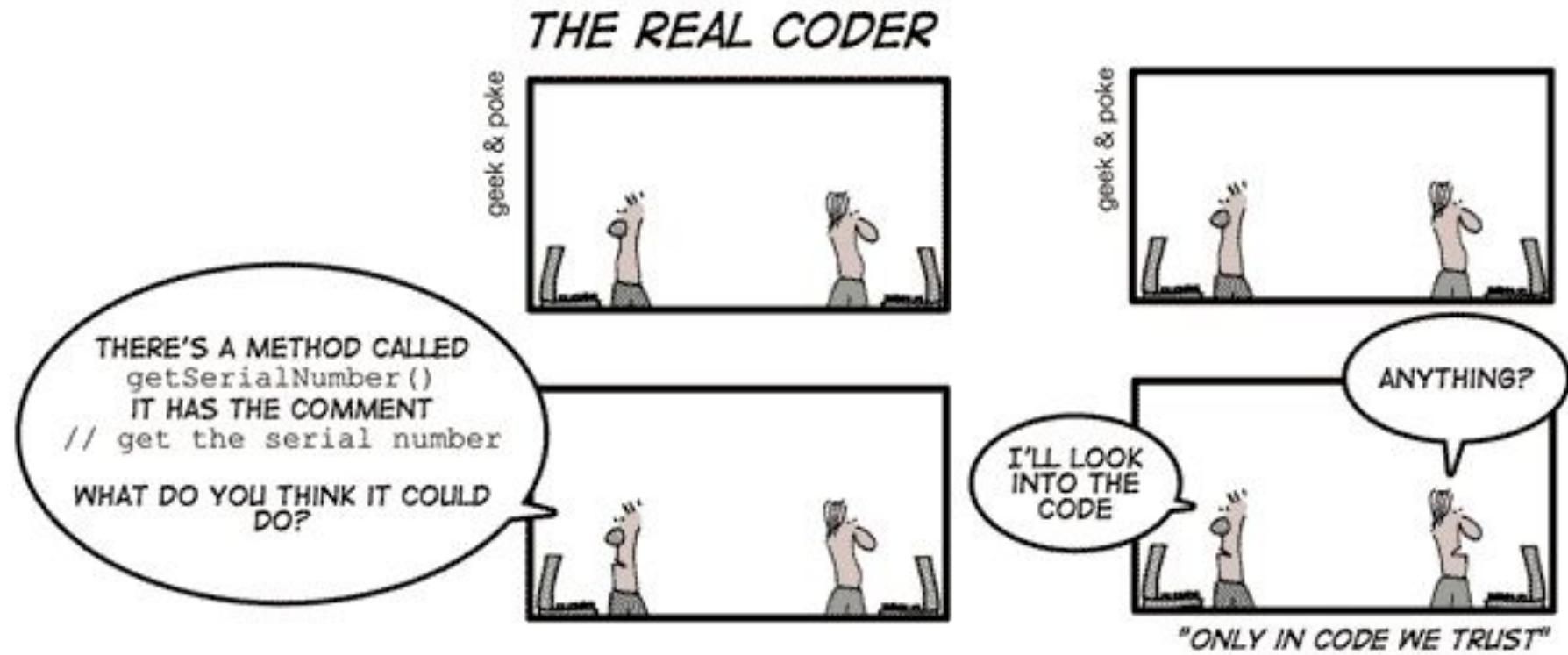
Dado um serviço em sua infra, ele deve estar continuamente disponível!

Testabilidade



Sua infra deve ser testável em conjunto com sua aplicação.

Documentável



O “código” será sua documentação.

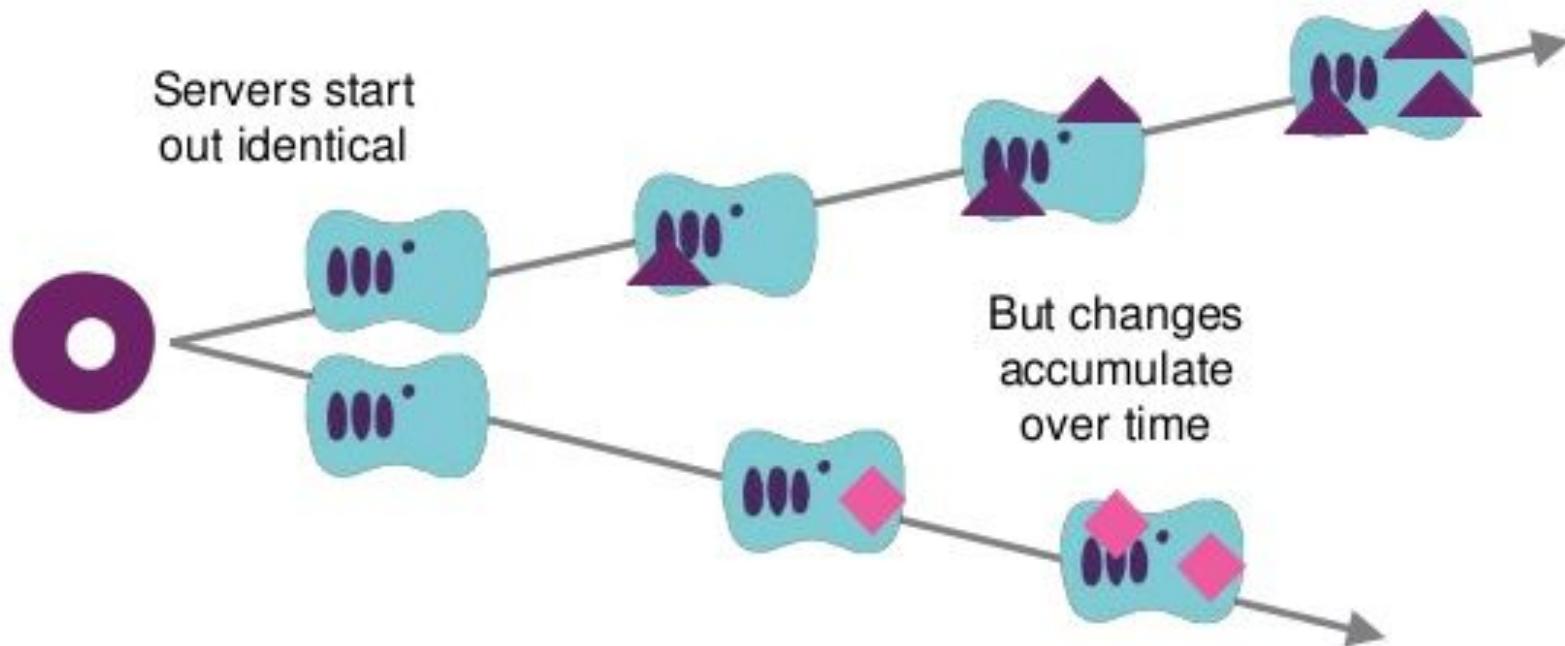
Versionável



Rastreabilidade, *rollback*, transparéncia...

Mas cuidado...

Configuration Drift



Servidores floco de neve

Servidores em execução
por muito tempo, recebendo
diversas atualizações.



Servidores floco de neve

Servidores em execução
por muito tempo, recebendo
diversas atualizações.

**Alguém saberia que
serviços estão instalados
nesse servidor?!?!**

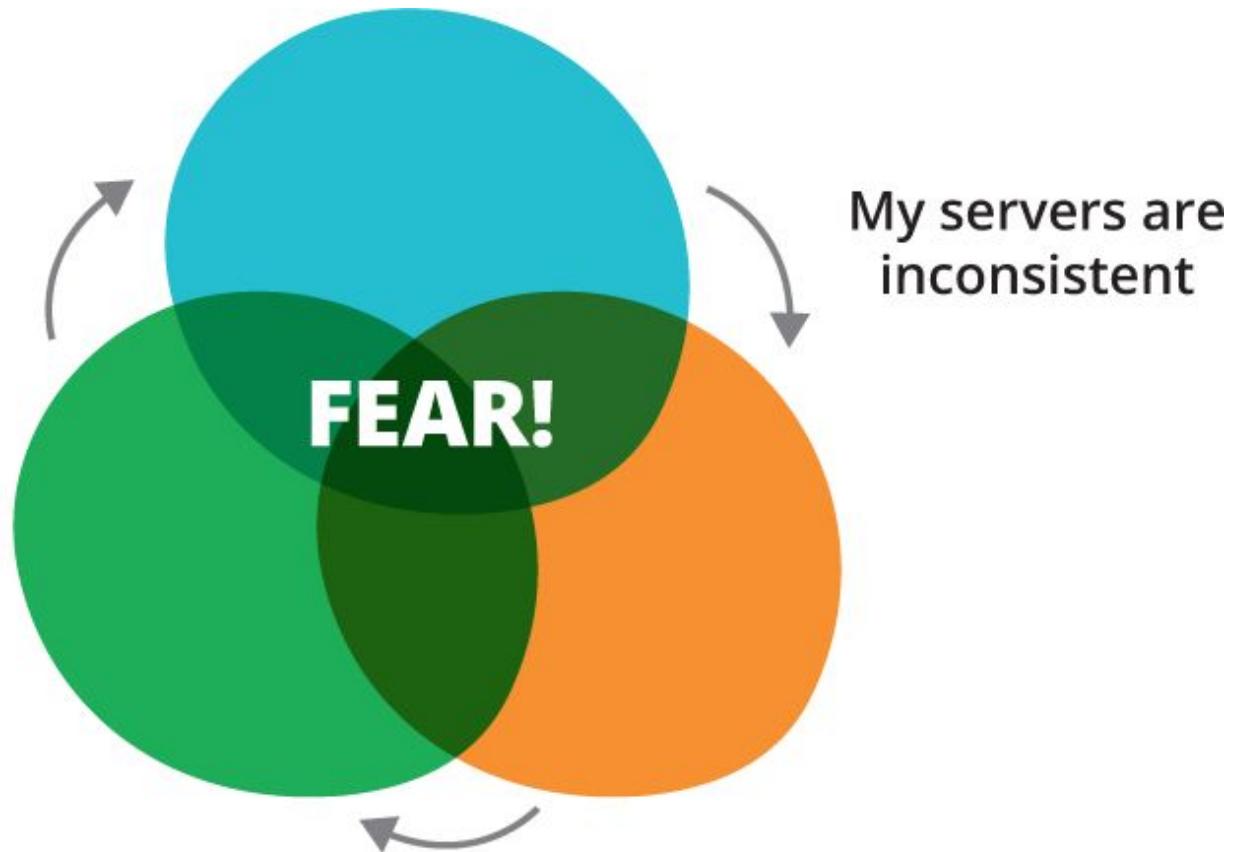


Infraestrutura “Jenga”



Medo da Automação

I make changes
outside my
automation tool



I'm afraid that running
my automation tool will
break something

My servers are
inconsistent

Erosão \ Desgaste

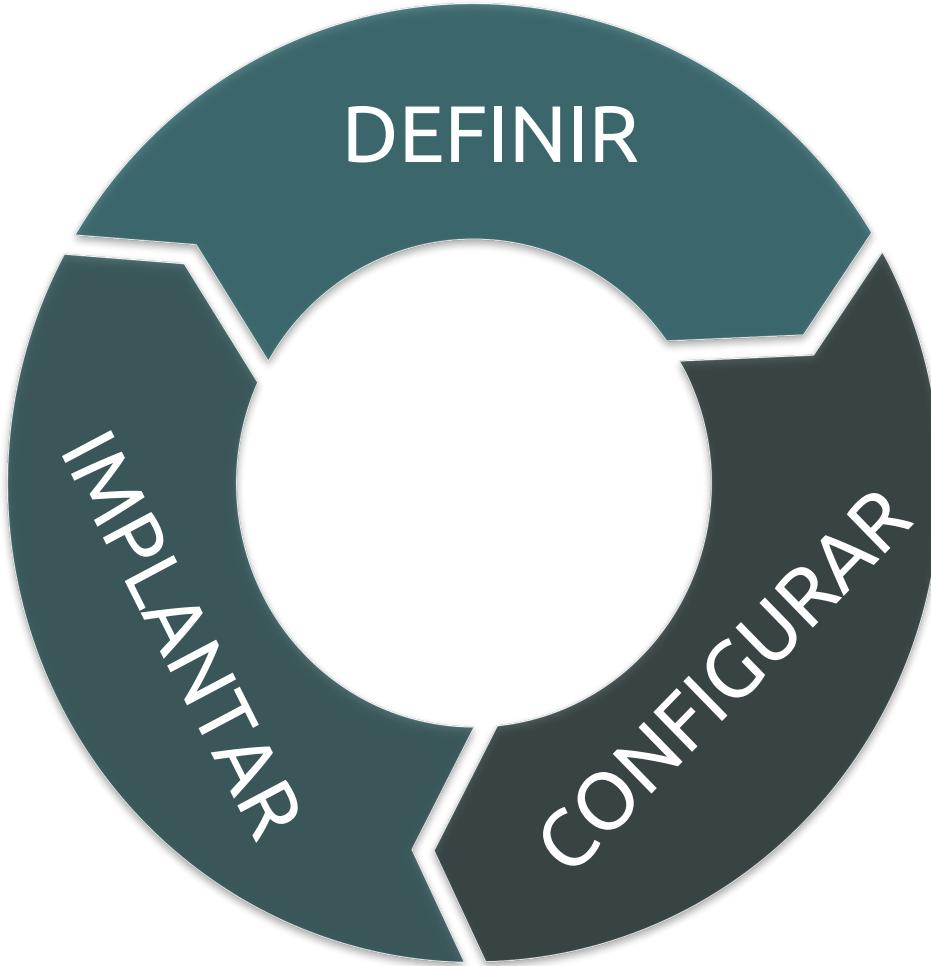


Atenção aos hábitos com...

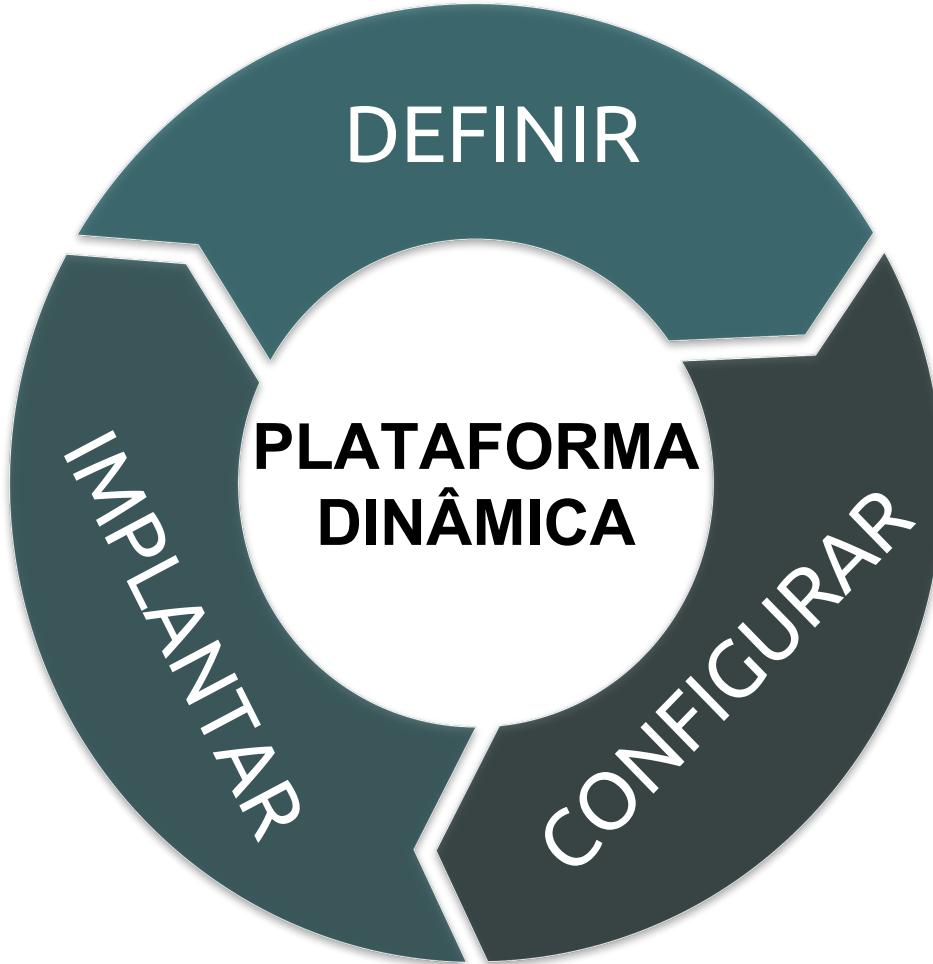
Más práticas

- Rotinas manuais
 - No Terminal Service é mais fácil!
- Execução de scripts manuais
 - Mas é só um *ssh*!
- Pular a Automação
 - E isso funciona?!

CICLO



CICLO



Quem são?

vmware®

rackspace®



DigitalOcean

 terremark®



Microsoft Azure



 Joyent®

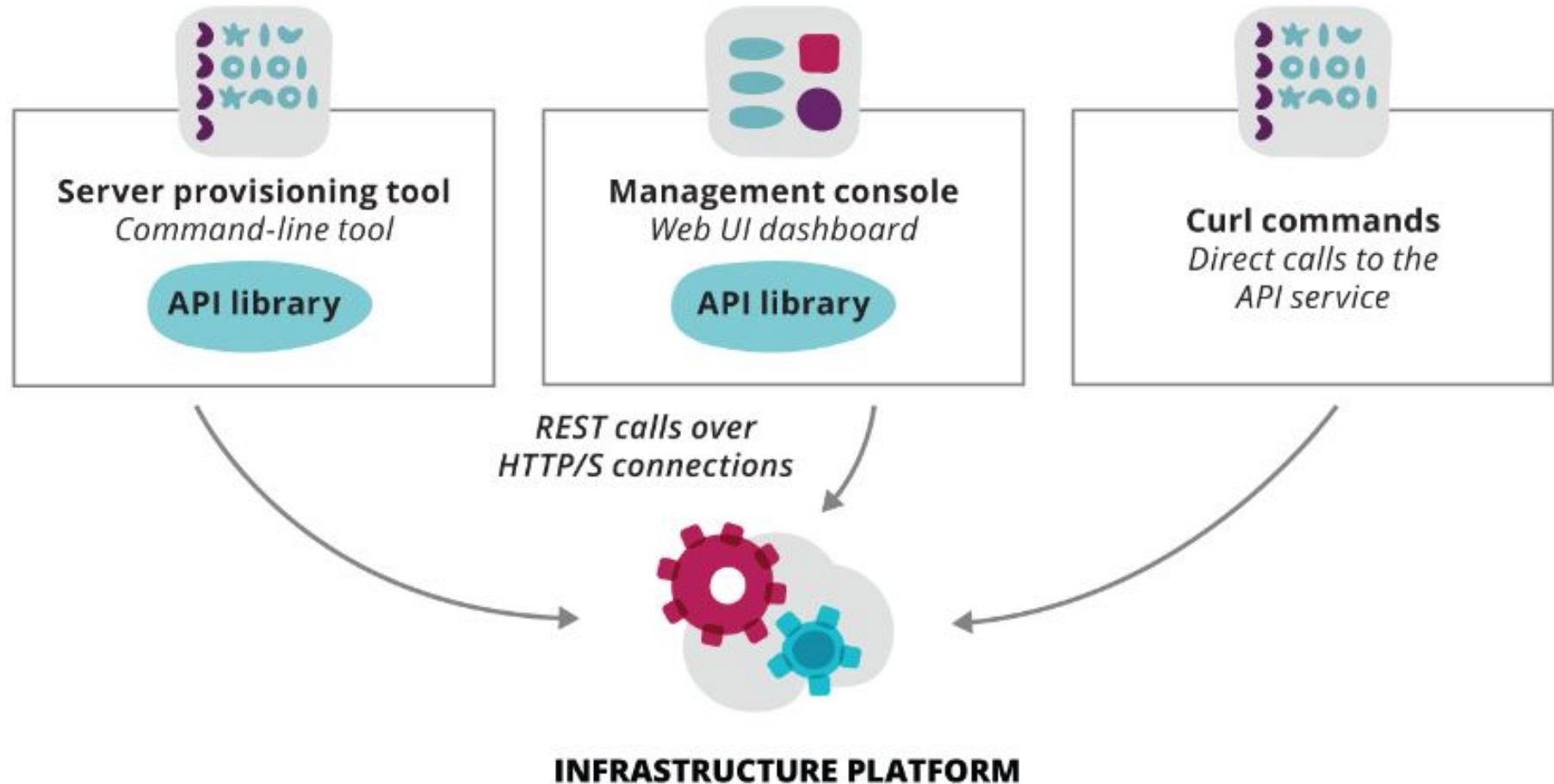


Google Cloud Platform



amazon
web services

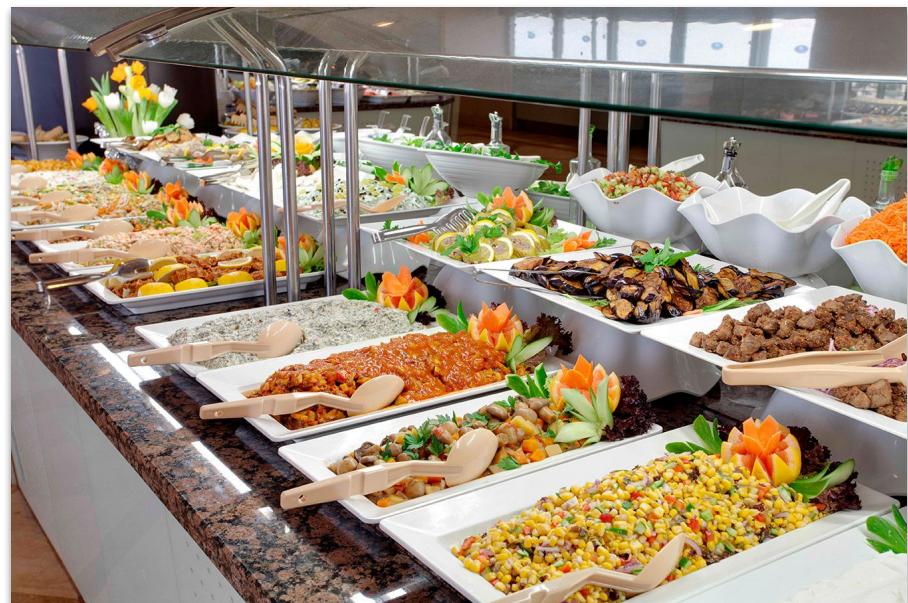
Programável





Self Service

On demand



Cuidado com a manivela (*Hand-cranked Infra*)

*Precisão
de
cálculo
é o que
interessa*



Esta voltinha dá o balanço certo

As modas passam e a Facitinha continua.
Continua há mais de um século, resolvendo
qualquer tipo de problema que aparecer pela frente,
da contabilidade, do departamento de vendas
ou da produção.

A Facitinha calcula com absoluta precisão, é
mais econômica e conta com assistência técnica
garantida.

É a solução ideal nos escritórios, nas obras, em
qualquer lugar, para todo o tipo de problema,
inclusive o seu.

Escolha a Facitinha em novas cores: bege e
amarela.



FACITINHA

COISA BOA NÃO MUDA

FACIT

PRECISÃO DE CÁLCULO - PERFEIÇÃO DE ESCRITA

MATRIZ - São Paulo - Rua 13 de Maio, 012 - tel. 284-0133

FILIAIS - Brasília, Belo Horizonte, Curitiba, Porto Alegre,

Rio de Janeiro, Salvador, Aracaju e Santa

RCY ENDEDORES EM TODO O BRASIL



Atividades

- Provisionamento
- Configuração

Provisionamento

Diz respeito a **definição** e **criação** de toda infraestrutura necessária para execução de uma aplicação



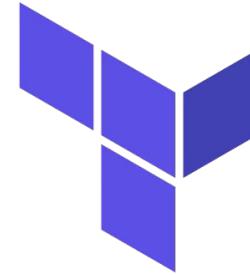
Provisionamento

- Alocar ou desalocar infraestrutura
 - Criar ou destruir servidores
 - Espaço em disco
 - Instalar SO
 - Criar interface de rede
 - Adicionar ou remover servidor em um pool
 - etc
- Provisionamento acontece em diversos momentos do ciclo de vida da minha Infra
 - Ex.: mudar a responsabilidade de um servidor

Provisionamento

- Templates
 - VMs, AMIs, etc
- Configuração é Provisionamento?
 - Veremos isso depois!
- As ferramentas de provisionamento podem manter estado
 - Ex.: Reorganizar a quantidade de servidores em um pool

Ferramentas



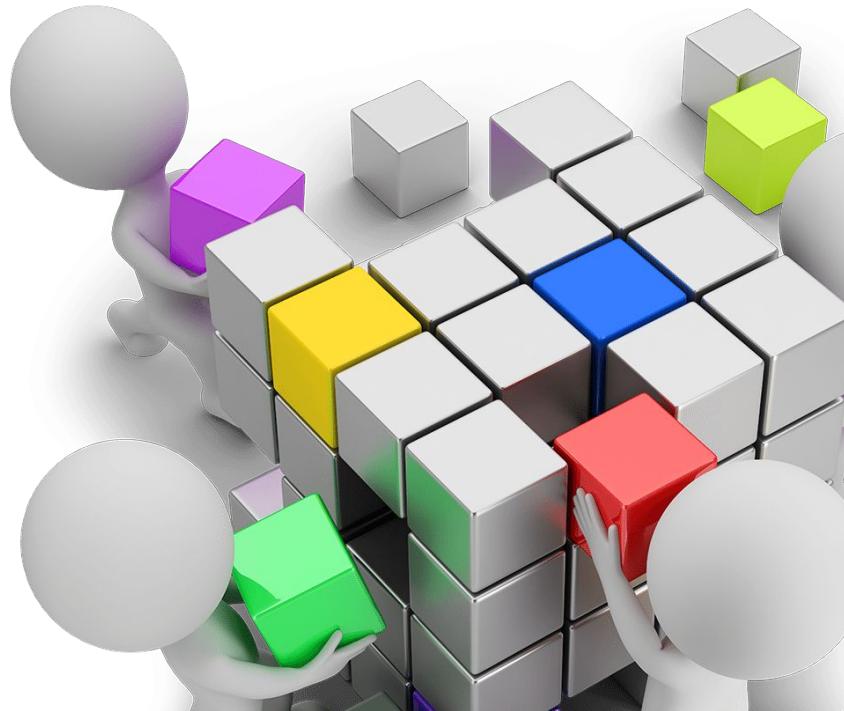
HashiCorp
Terraform



CloudFormation

Configuração

Ajustar minha infraestrutura para a necessidade da minha aplicação \ negócio



Configuração

- Controlar como o sistema ou aplicação funcionam
 - arquivos de configuração
 - tabelas de rota
 - algoritmo de balanceamento
 - Configuração de serviços
 - DHCP
 - Serviço de nomes
 - Firewall
 - etc
- Idempotência
 - **Mesma operação + mesmas entradas = mesmos resultados**

Configuração

- Geralmente organizado em:
 - Interface linha de comando
 - API REST
 - DSL para descrição de configurações
 - Modelo declarativo vs imperativo
 - Mecanismo para atualização
 - Pull vs Push

Modelo Declarativo

- Foca no “que” e não no “como”
- Define estados
- Tende a ser mais simples, porém menos poderoso

```
directory '/var/www/repo' do
  mode '0755'
  owner 'www'
  group 'www'
end

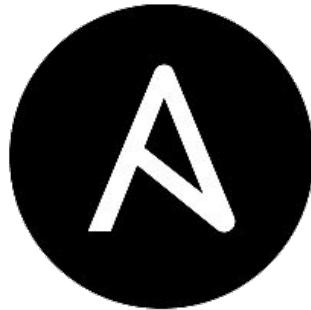
file '/var/www/repo/index.html' do
  source '/mnt/repository/index.html'
  mode '0644'
  owner 'www'
  group 'www'
end
```

Modelo Imperativo ou Procedural

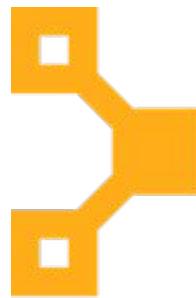
- Foca no “como” e não no “que”
- Define procedimentos
- Mais poderoso, porém complexo

```
if [ ! -d /var/www/repo ] ; then
    mkdir -p -m 0755 /var/www/repo
fi
cp /mnt/repository/index.html /var/www/repo
chown -R www:www /var/www/repo
chmod 0644 /var/www/repo/index.html
```

Ferramentas



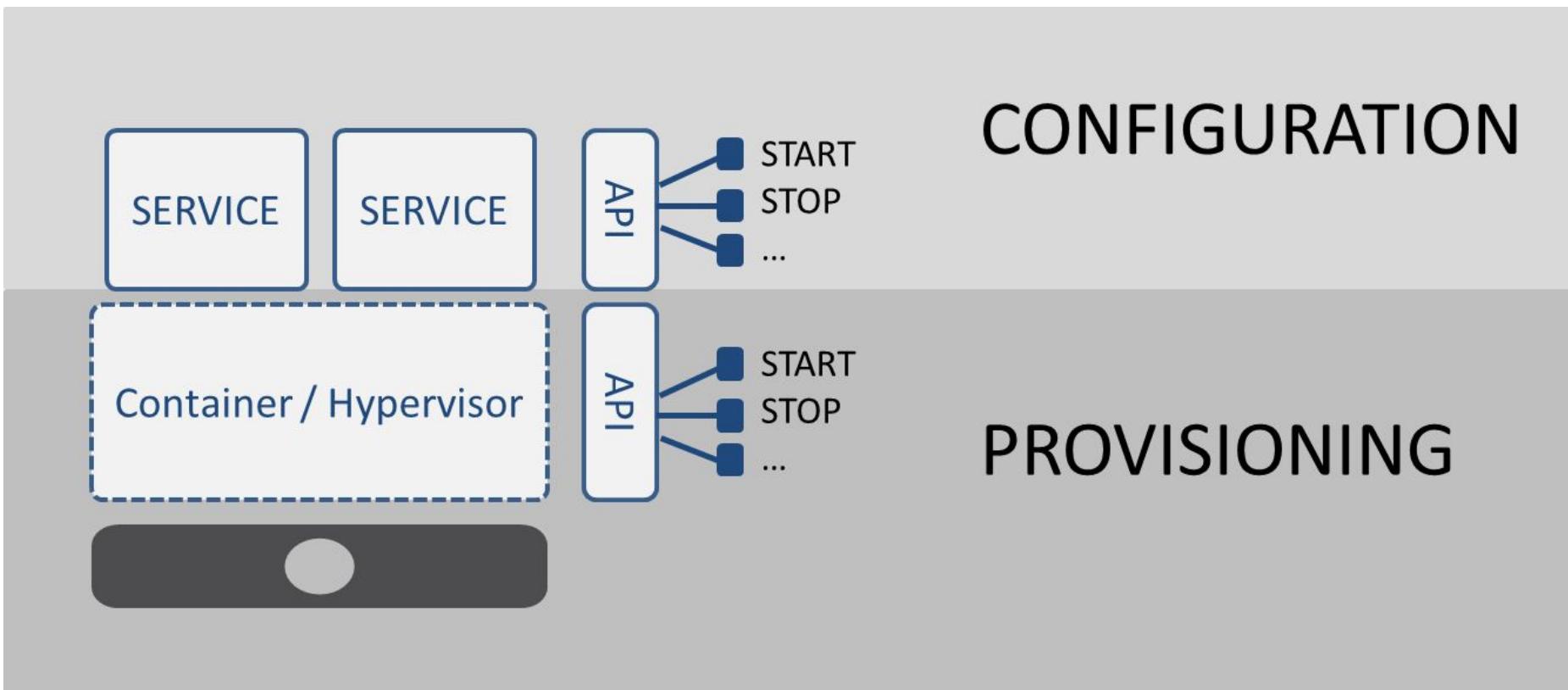
ANSIBLE



puppet

Importante!!!

Configuração não é igual a Provisionamento



Exercício

PROVISIONAMENTO vs CONFIGURAÇÃO



PROVISIONAMENTO vs CONFIGURAÇÃO



PROVISIONAMENTO vs CONFIGURAÇÃO

Criar um Servidor



PROVISIONAMENTO vs CONFIGURAÇÃO

Criar um Servidor

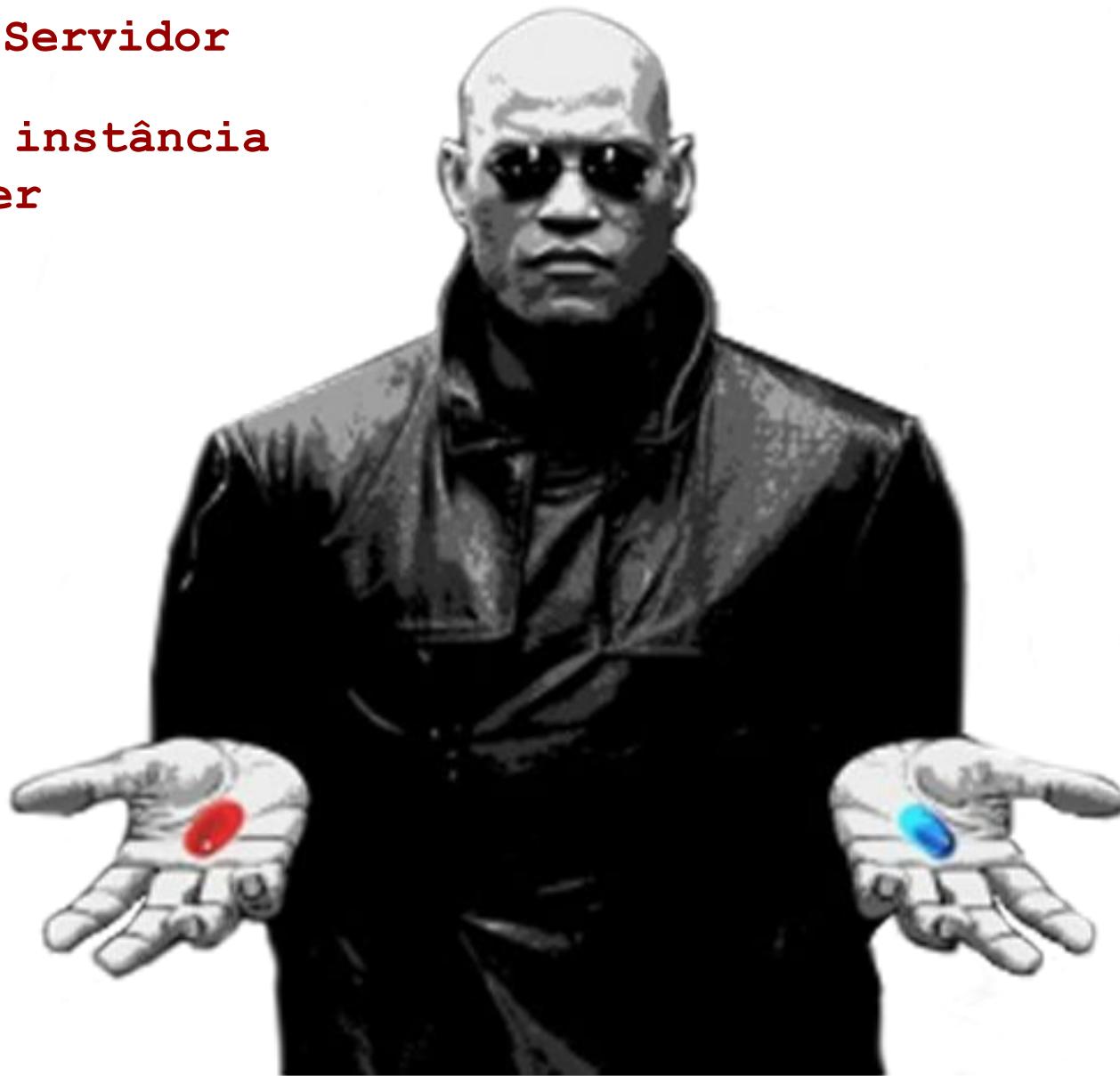
Destruir instância
do cluster



PROVISIONAMENTO vs CONFIGURAÇÃO

Criar um Servidor

Destruir instância
do cluster

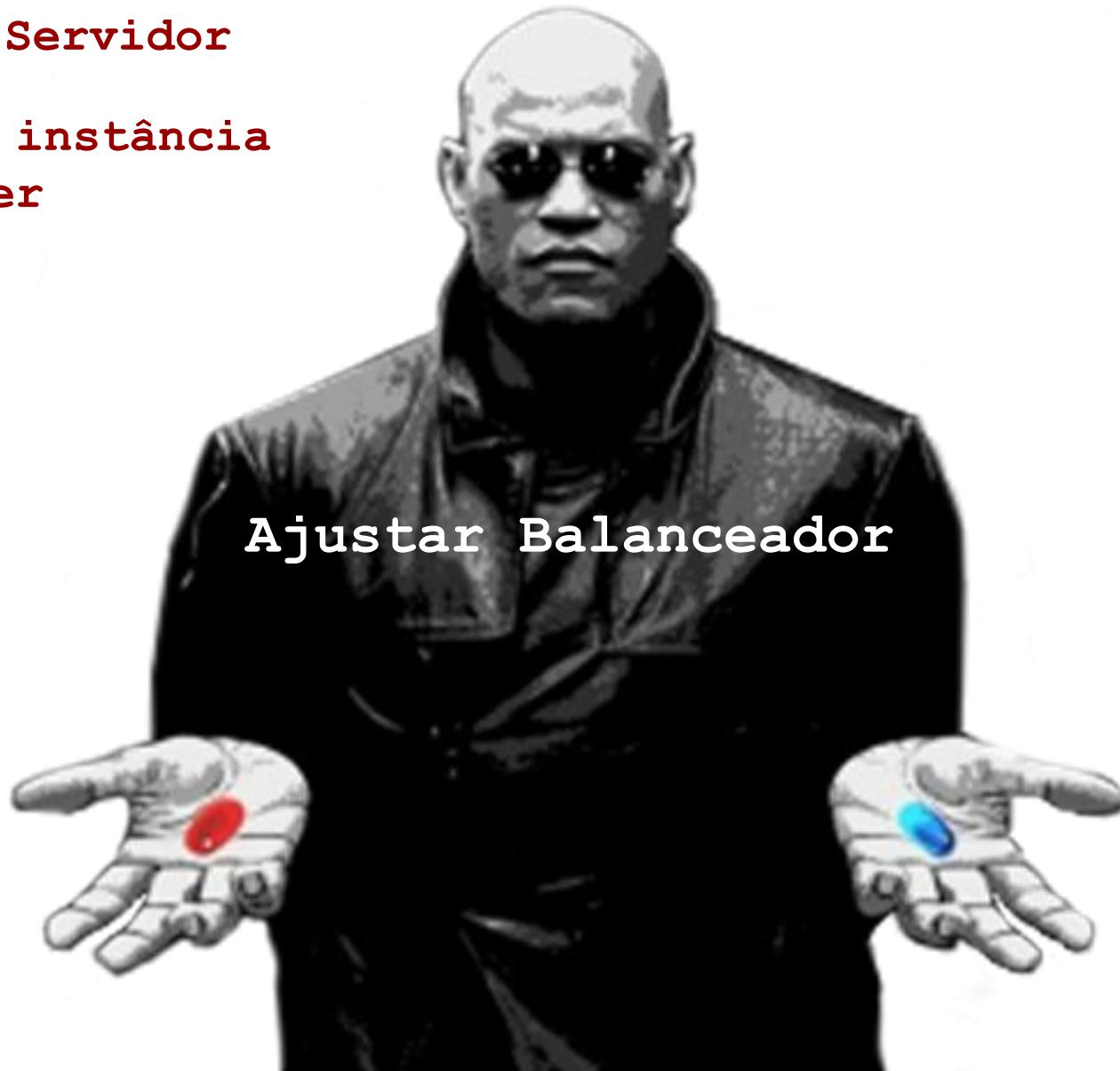


PROVISIONAMENTO vs CONFIGURAÇÃO

Criar um Servidor

Destruir instância
do cluster

Ajustar Balanceador

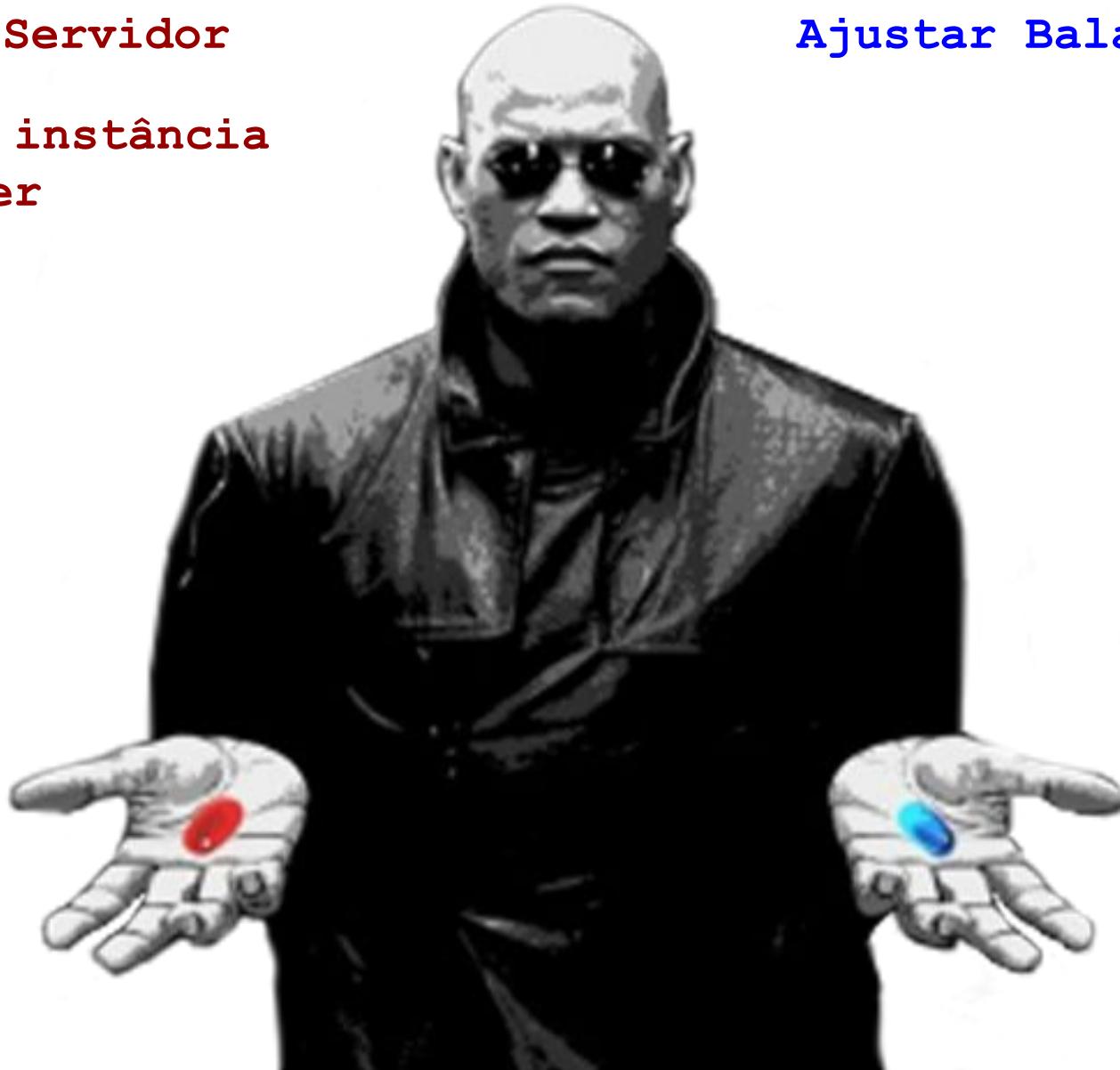


PROVISIONAMENTO vs CONFIGURAÇÃO

Criar um Servidor

Destruir instância
do cluster

Ajustar Balanceador



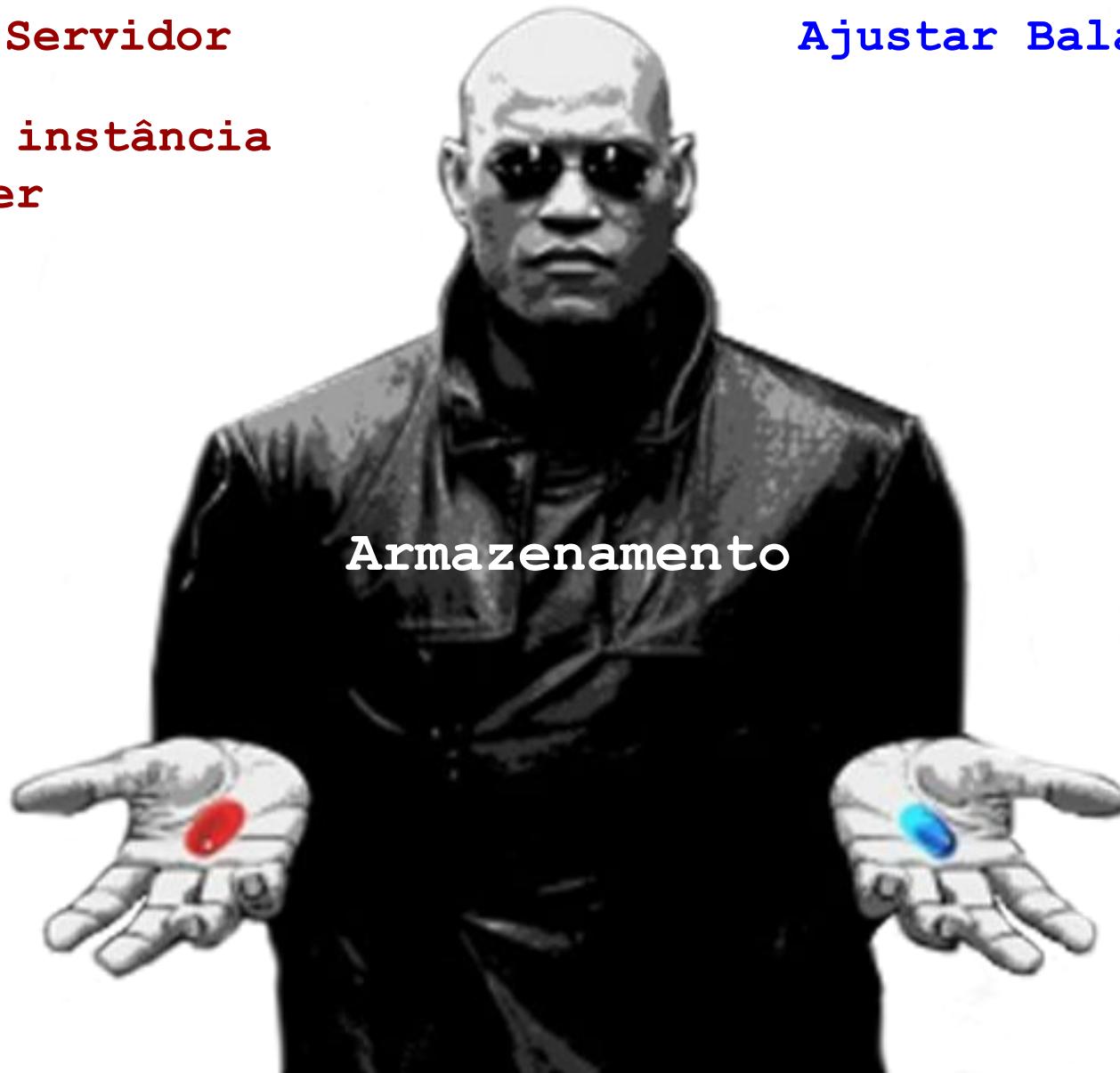
PROVISIONAMENTO vs CONFIGURAÇÃO

Criar um Servidor

Destruir instância
do cluster

Ajustar Balanceador

Armazenamento



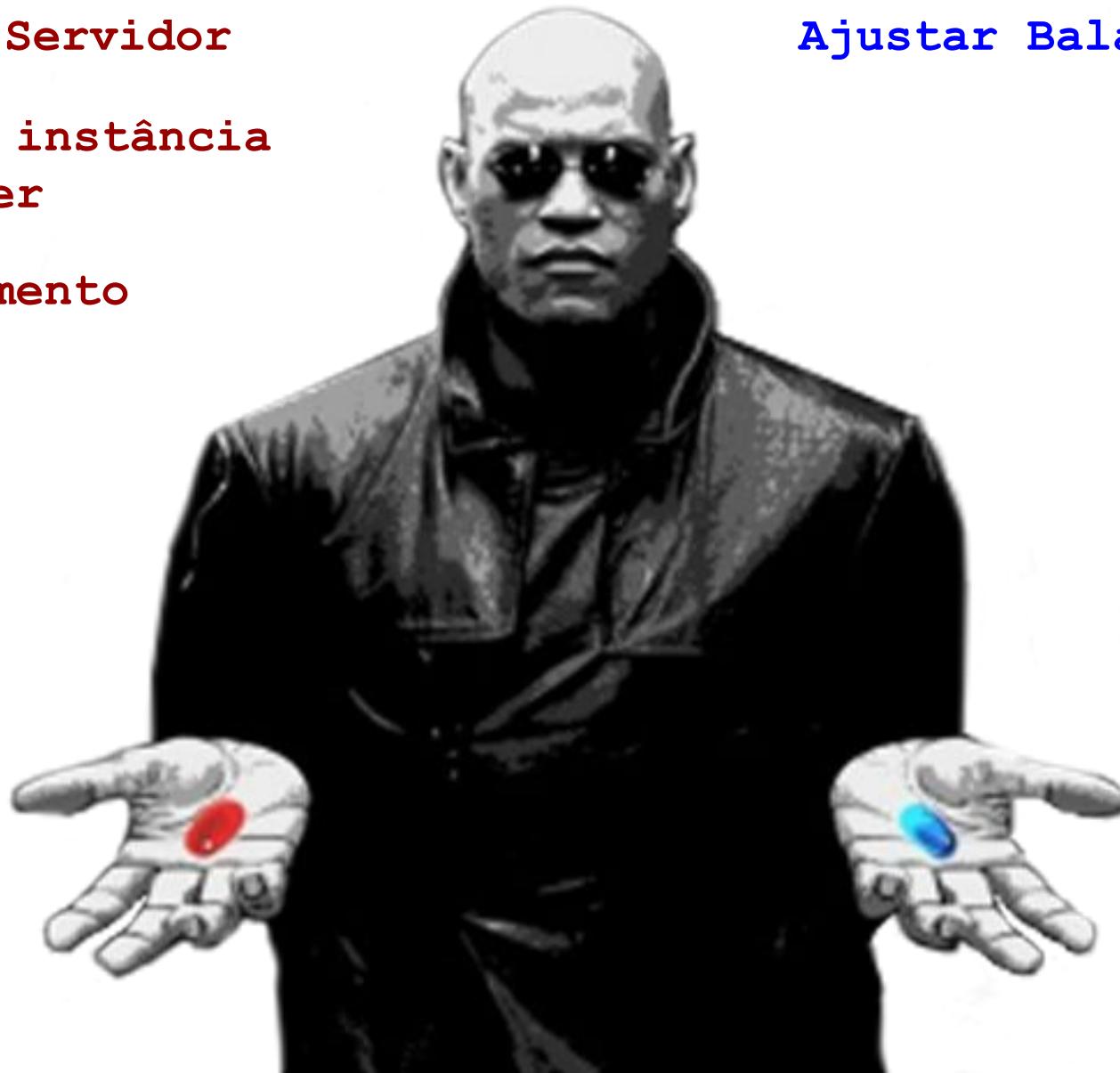
PROVISIONAMENTO vs CONFIGURAÇÃO

Criar um Servidor

Destruir instância
do cluster

Armazenamento

Ajustar Balanceador



PROVISIONAMENTO vs CONFIGURAÇÃO

Criar um Servidor

Destruir instância
do cluster

Armazenamento

Ajustar Balanceador

Ponto de Montagem



PROVISIONAMENTO vs CONFIGURAÇÃO

Criar um Servidor

Destruir instância
do cluster

Armazenamento

Ajustar Balanceador

Ponto de Montagem



PROVISIONAMENTO vs CONFIGURAÇÃO

Criar um Servidor

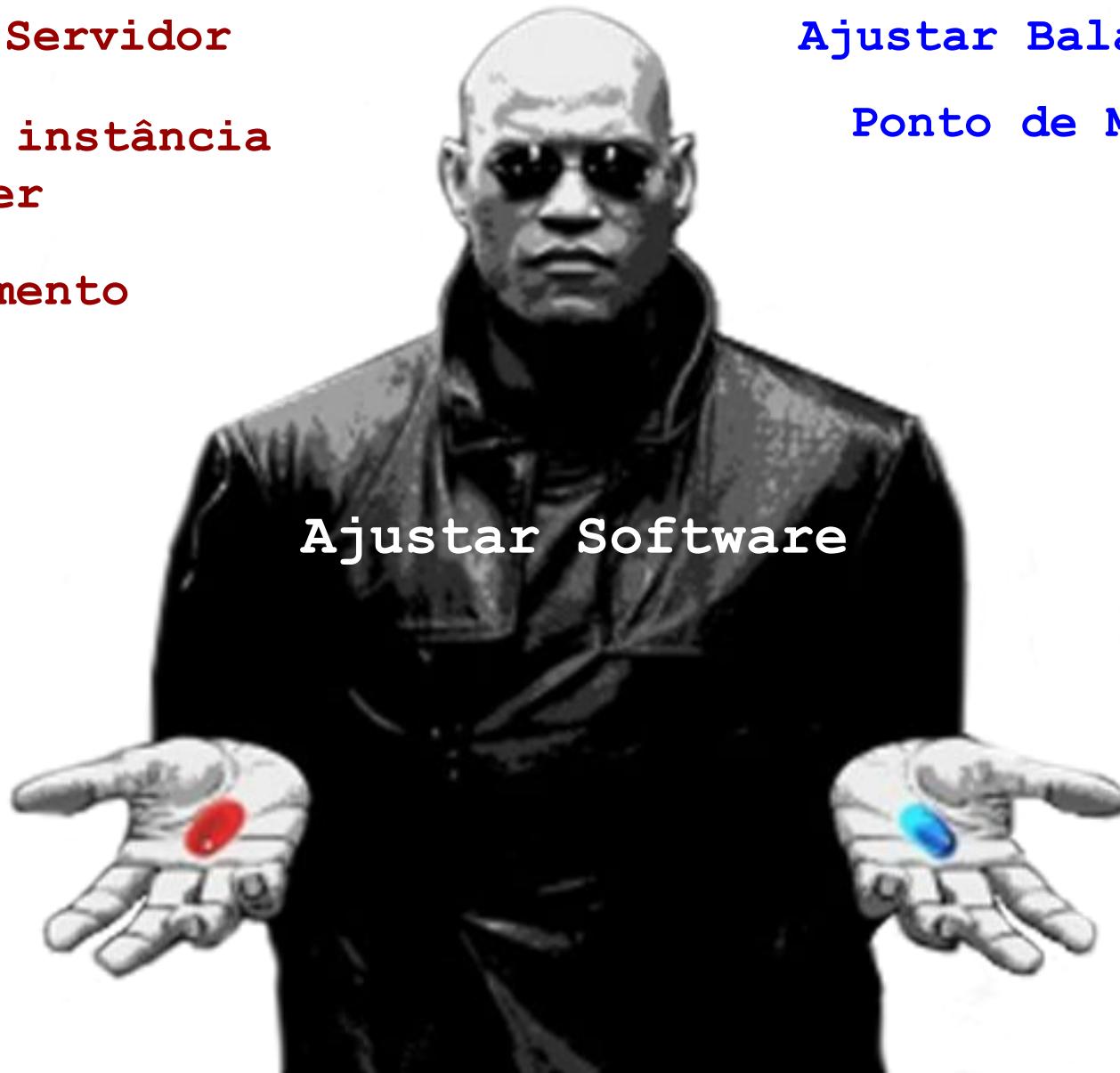
Destruir instância
do cluster

Armazenamento

Ajustar Balanceador

Ponto de Montagem

Ajustar Software



PROVISIONAMENTO vs CONFIGURAÇÃO

Criar um Servidor

Destruir instância
do cluster

Armazenamento

Ajustar Balanceador

Ponto de Montagem

Ajustar Software



Orquestração

- Otimização de processos
- Ajustes em larga escala e várias aplicações
- Monitoramento e ação
- A maioria das ferramentas de CM possuem aplicações separadas para orquestração

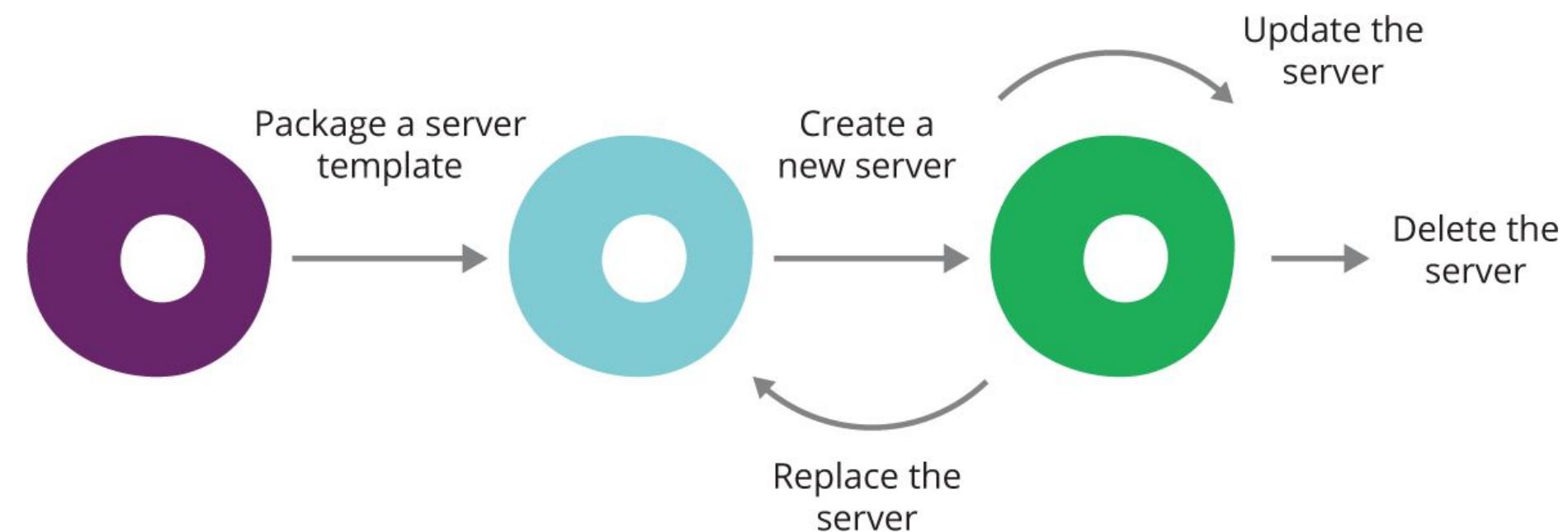
Servidores

- Gerenciamento
 - Ciclo de vida
- Padrões
- Práticas (Boas e más)

“Em uma infraestrutura, há muito mais que servidores. Contudo, eles geralmente nos dão muito trabalho e sugam nossa energia!”

Kief Morris

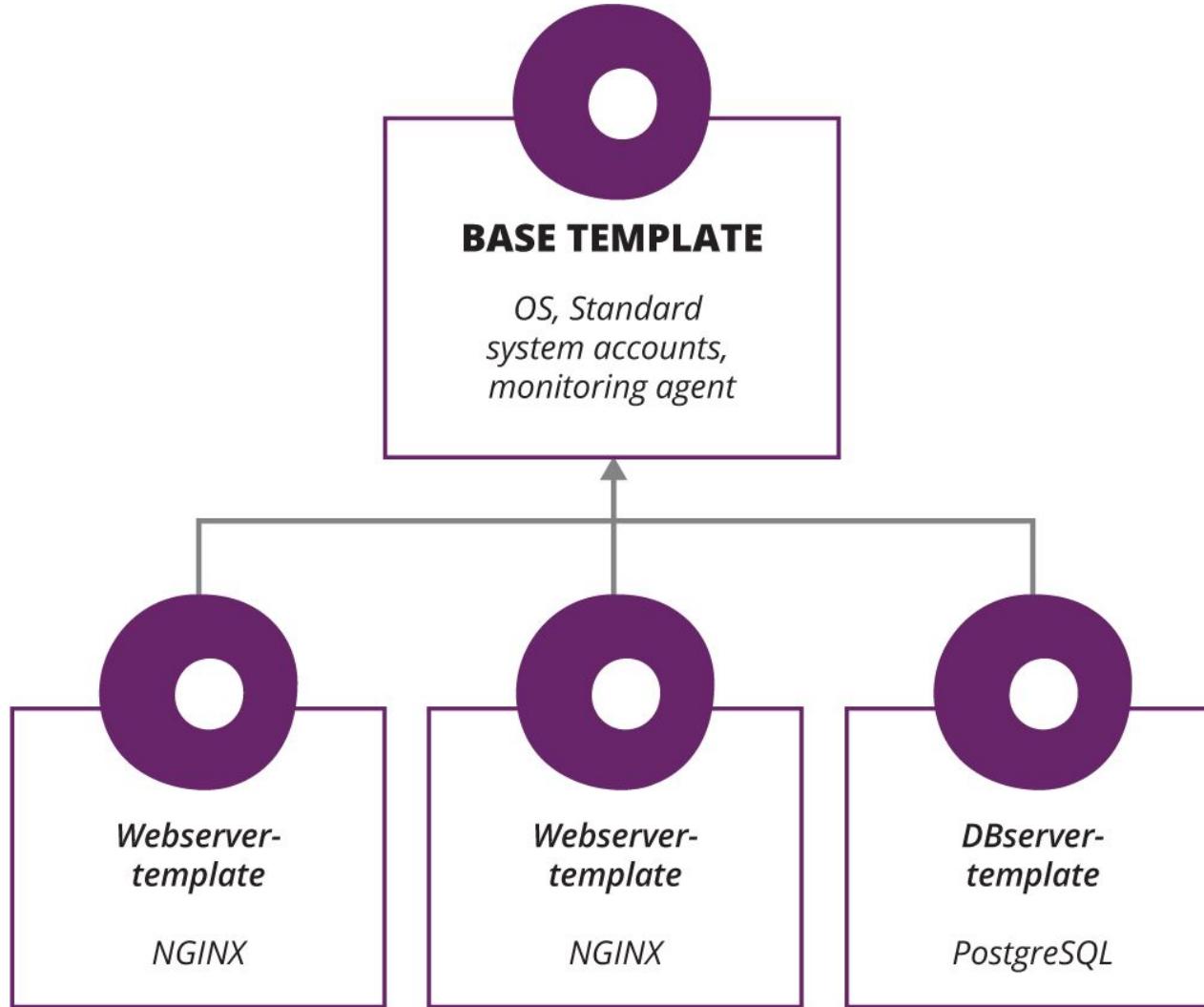
Ciclo de vida



Outras ações...

- Recuperação de desastres
- Redimensionamento de *pools*
- Reconfiguração de hardware

Servidores possuem papéis



O que temos em cada servidor?

- Software
 - Aplicações, bibliotecas, código...
 - Devem ser os mesmos para cada instância de um tipo de servidor
- Configurações
 - Arquivos de controle (SO, aplicação, etc)
 - Diferentes para cada instância
 - Importante manter a consistência
- Dados
 - Conteúdo vivo

Criação e Atualização de Servidores

Como criamos servidores?

Como criamos servidores?

- Clonagem
- Usando snapshots
- Instalando do “zero”
- etc

Criando servidores “na mão”...

- Características
 - Simples...
 - Não escalável
 - Dificilmente repetível e reproduzível

“Se na mão você criar, problema você terá! O problema leva a raiva... A raiva leva a dor de cabeça... Resolver isso, depois pior será!”



Clone os servidores...

- Prático
- Problemas se propagam pelas cópias
- Configurações copiadas
- Leva ao desvio de configuração

Estratégias melhores...

- Defina templates e os copie
 - Evita a cópia dos dados de configuração
 - Cuidado com templates “inchados”
- Use scripts ou softwares para criar, definir e configurar o que o servidor terá
 - O processo pode ser repetível e te dará escala

Como atualizamos servidores?

Como atualizamos servidores?

- O processo deve ser:
 - Natural
 - Compreender todos os servidores envolvidos
 - Não acompanhado
 - Livre de inconsistências
 - Escalável
 - Esforço deve tender a ser o mesmo independente da quantidade de servidores

Como atualizamos servidores?

- Na munheca (Tradicional)
- Sincronização contínua
 - Pull vs Push
- Servidores imutáveis

Modelo Tradicional

- os servidores são atualizados conforme a necessidade das aplicações
 - SSH
 - Acesso físico
- Histórico de mudanças é importante
- Alternativa padrão às ferramentas

Sincronia Contínua

- Mudanças em configuração implicam em mudanças nos servidores
- Uma ferramenta é responsável por identificar a mudança e propagá-la
- Não necessita de suporte humano
- Por si só, não elimina problema como *desvio de configuração*

Padrão Push

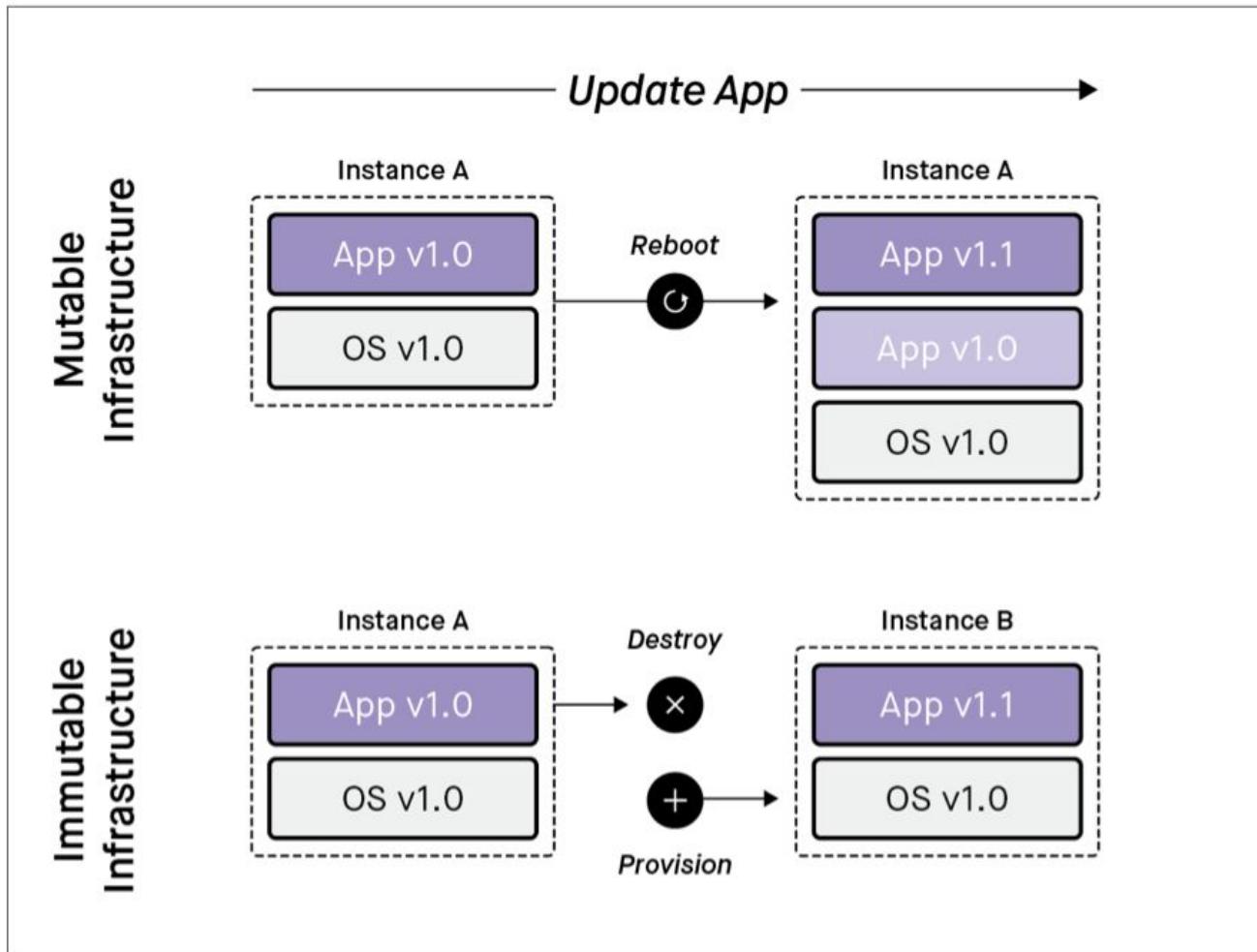
- Também chamada Sem Master ou Sem Agente
 - Responsável pela atualização informa a mudança necessária para os nós
 - Nós realizam as atualizações conforme necessidade
- Pode ou não ter elemento centralizado

Padrão Pull

- Também chamada Master ou Agente
 - Há elemento centralizado que coordena nós
 - Nós checam status das atualizações
 - Nós realizam as atualizações conforme necessidade
- requer *daemon* proprietário instalado em todos os nós

Infraestrutura Imutável

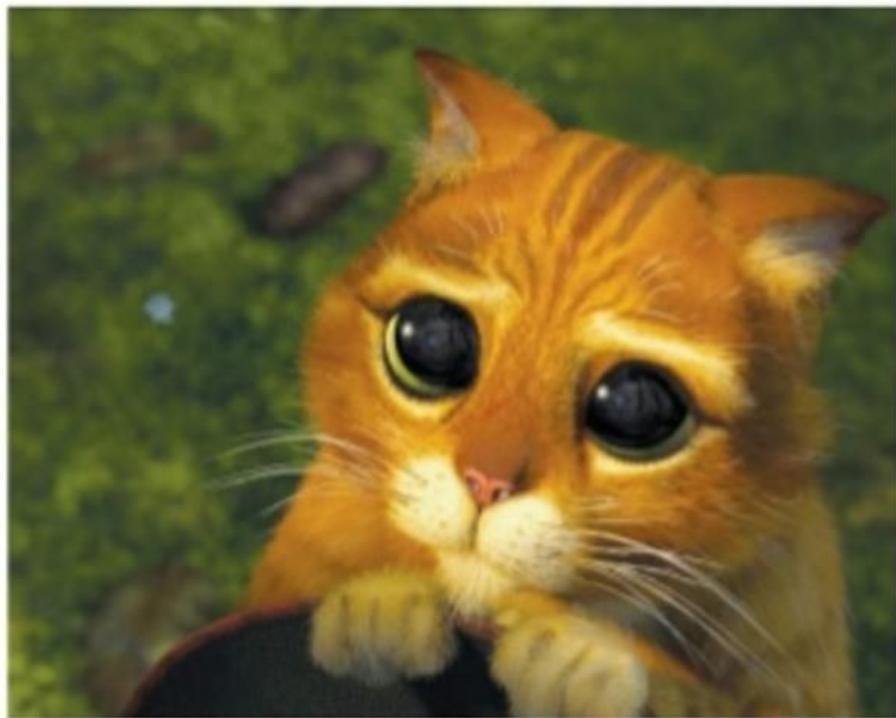
Servidores nunca são atualizados, mas sim destruídos e criados novamente



Infraestrutura Imutável

- Stateless
- Tende a eliminar o desvio de configuração
- Testes automatizados mais efetivos
- Pode ser mais complicado de garantir funcionamento

Bichinhos ou Gado

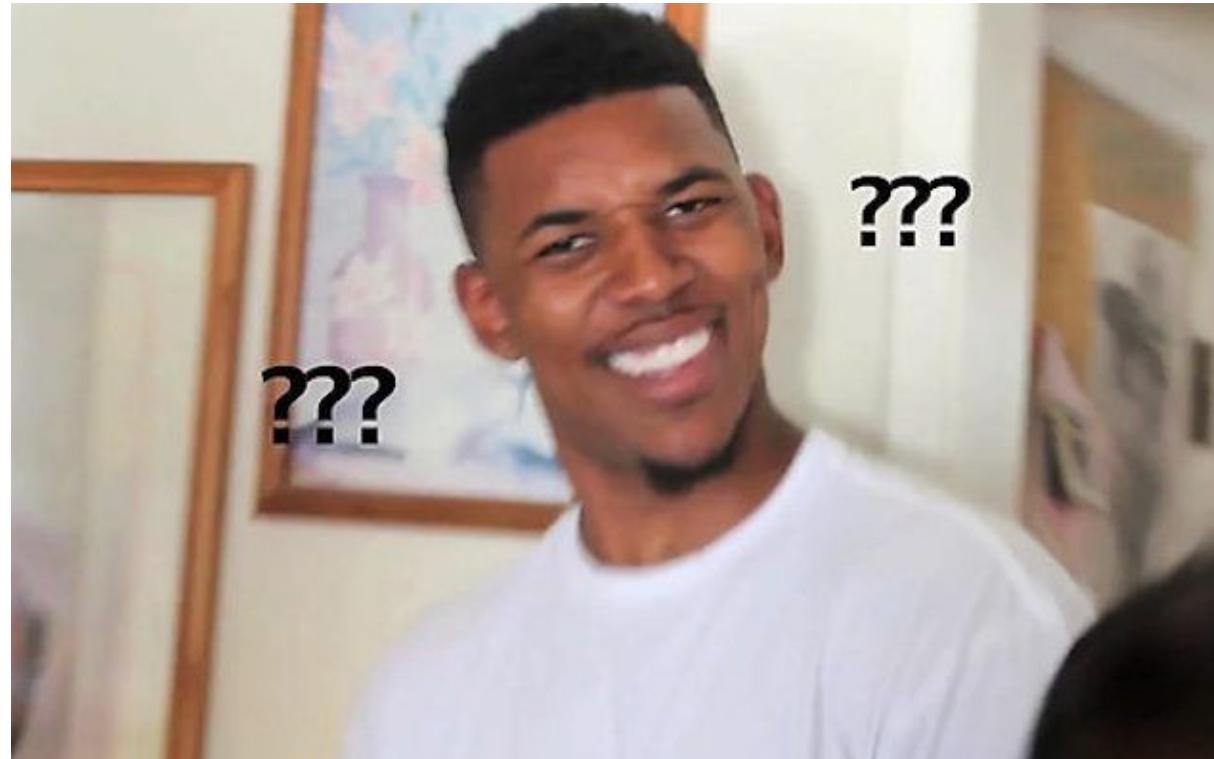


Servidores Fênix

Destrua e crie os servidores... mesmo que não haja
mudanças!

Servidores Fênix

Destrua e crie os servidores... mesmo que não haja mudanças!



Servidores Fênix

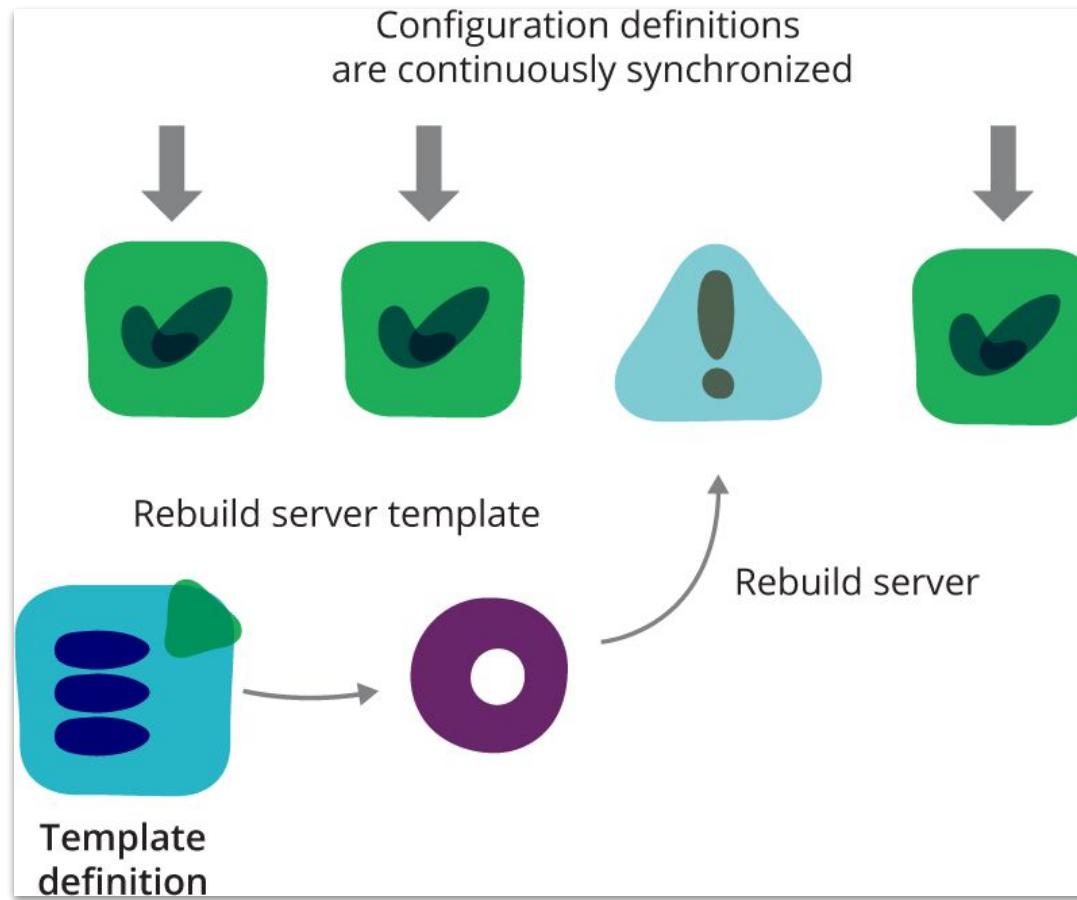
- combate o desvio de configuração e a erosão
- Processo
 - Configure um tempo de vida máximo para seus servidores
 - Destrua os servidores mais antigos que esse tempo de vida
 - Reconstrua os servidores confirme a configuração de seu pool
- Cuidado com a indisponibilidade do serviço!

Templates Enxutos

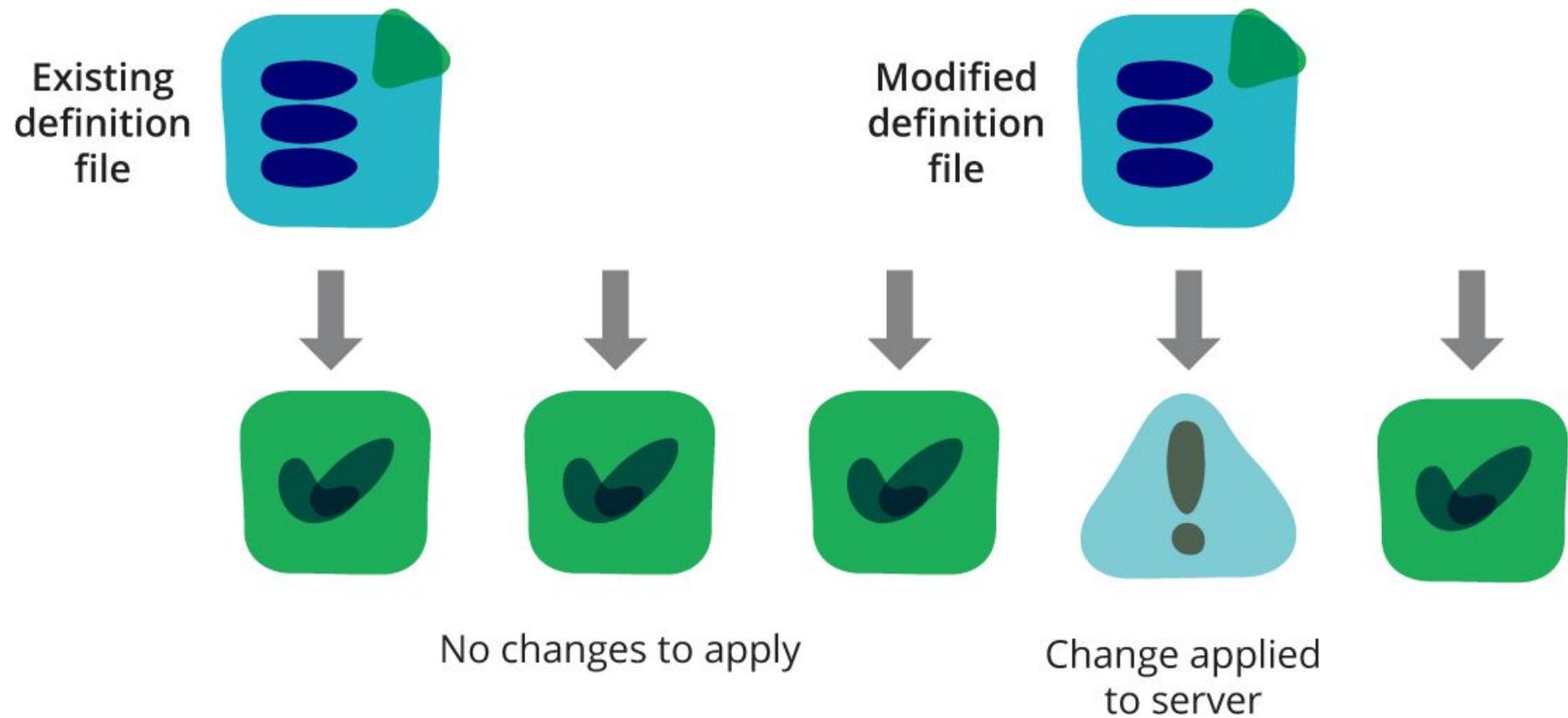
- Templates pequenos implicam em menos coisas para gerenciar
- 1 template = 1 papel

Mudou o template, mude os servidores

- Evita o desvio de configuração



Ciclo de sincronização contínuo



Ciclo de sincronização contínuo



Existing
definition
file

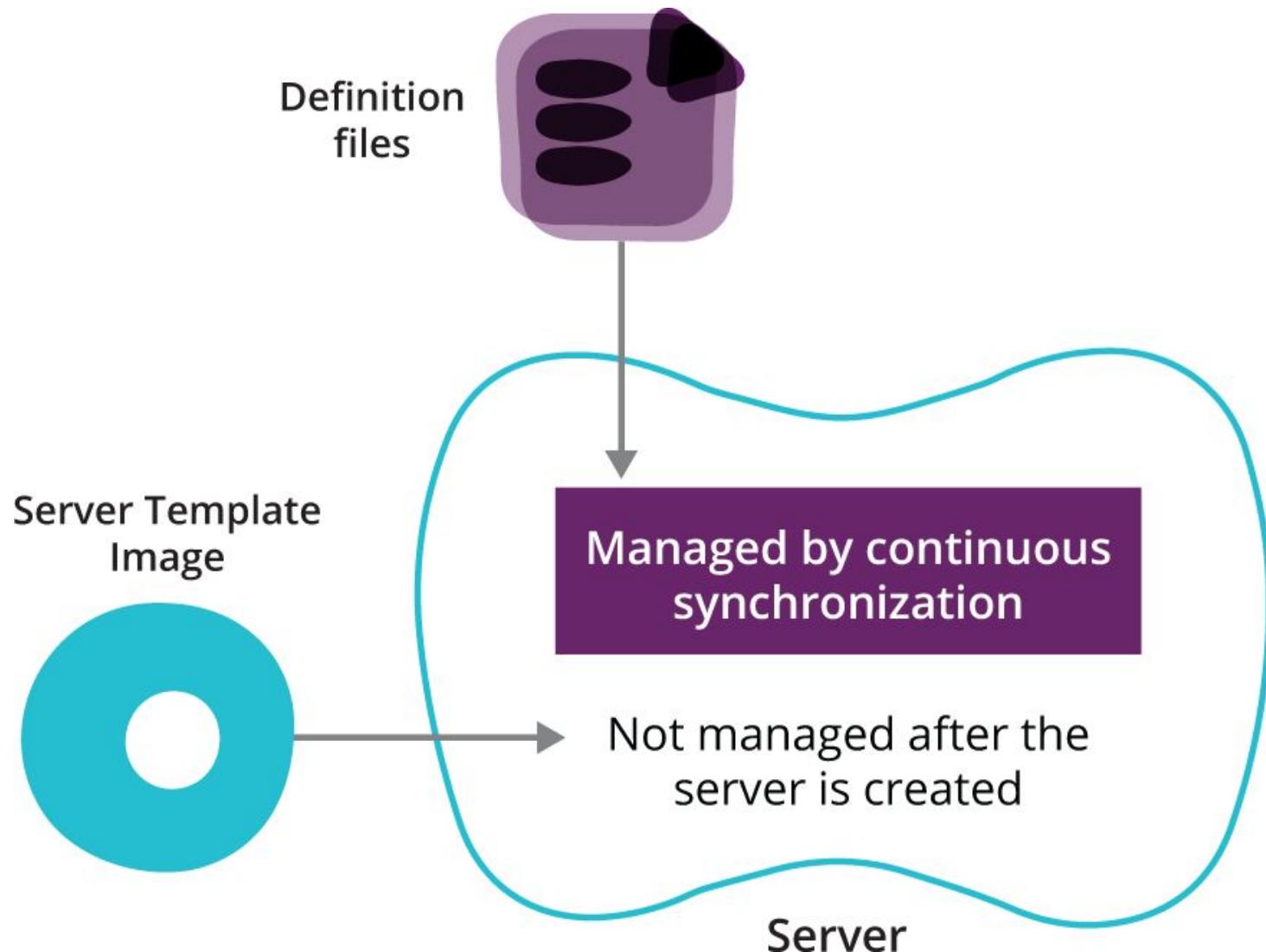


Manual change
made on server

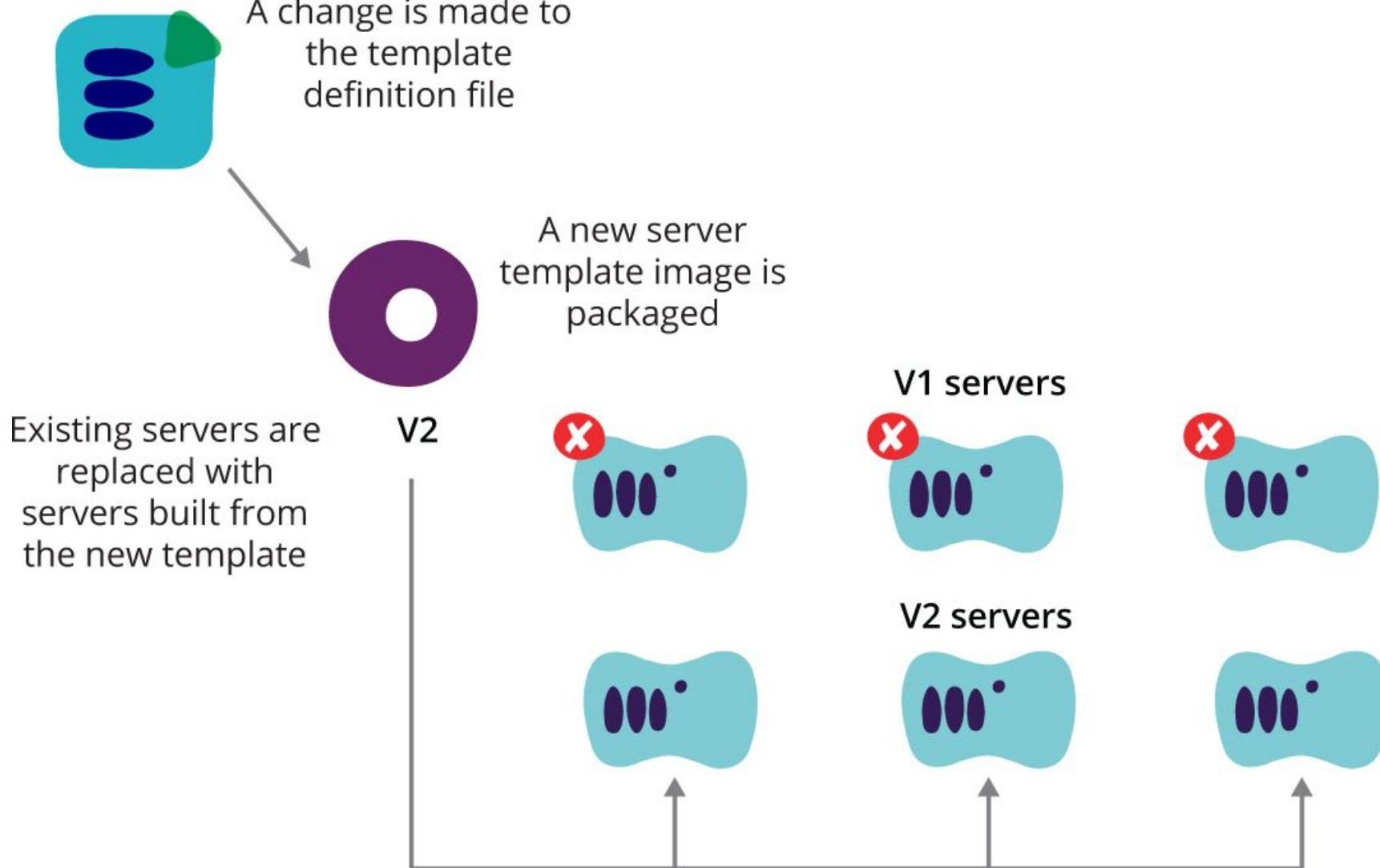
Server brought
back to
baseline



Mas cuidado com áreas não cobertas



Estabeleça um fluxo de imutabilidade



E tem mais...

- Deixe as configurações no mínimo
- Organize, empacote (e reuse) suas definições
- **TESTE!!!**

Organizando e Definindo Elementos da Infraestrutura

Stack

- Coleção de elementos de uma infraestrutura organizados como uma unidade
 - Pode ser um único servidor ou diversos elementos
- Facilita
 - controle
 - gestão
 - aplicação de políticas
 - etc

Ambiente / Perímetro

- Coleção de diversas *stacks* que representam o mesmo conjunto de servidores
- Uso
 - Testes
 - Separação
 - DEV, TESTE, HML, PROD

DICAS

- Mantenha a consistência entre ambientes reusando suas definições
- Promova suas definições entre ambientes

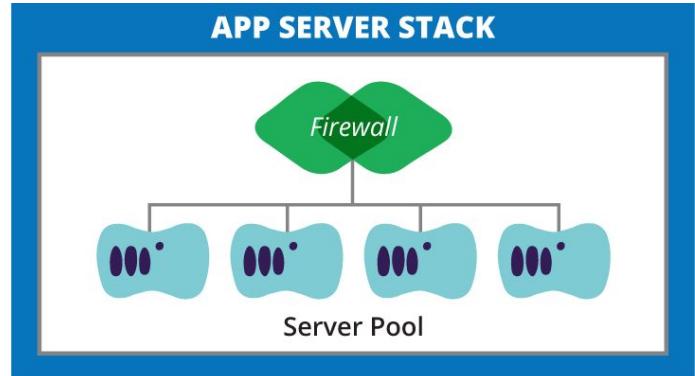
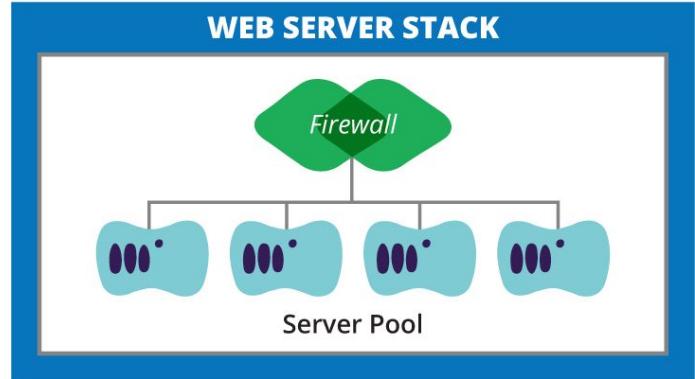
```
variable "environment" {}

resource "aws_instance" "web_server_1" {
  ami = "ami-47a23a30"
  instance_type = "t2.micro"
  tags {
    Name = "Web Server 1"
    Environment = "${var.environment}"
  }
}

resource "aws_instance" "web_server_2" {
  ami = "ami-47a23a30"
  instance_type = "t2.micro"
  tags {
    Name = "Web Server 2"
    Environment = "${var.environment}"
  }
}
```

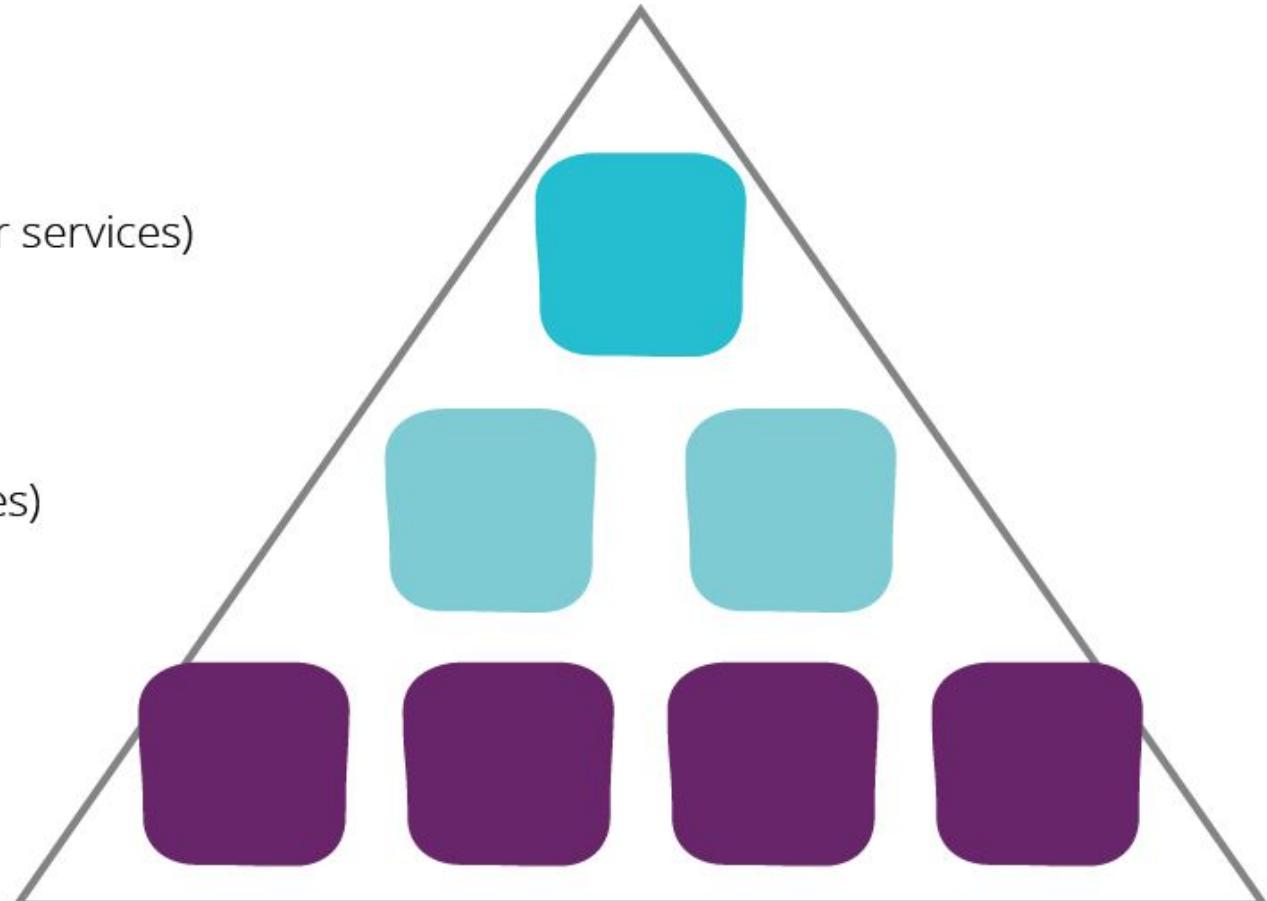
DICAS

- Cuidado com as stacks monolíticas
 - Segregue sua infra por camadas ou grupos que façam sentido estarem juntos
- Cuidado com o compartilhamento de elementos entre diversas aplicações
 - Ex.: diversas aplicações utilizando o mesmo cluster de proxies reversos



Testando a Infraestrutura

Pirâmide de testes



High-level tests

(e.g. deploy and test multi-tier services)

Medium-level tests

(e.g. build and test server roles)

Low-level tests

(e.g. validate definition files)

Pontos Importantes

- Cuidado com a inversão da pirâmide
- Foque nos testes de baixo nível
- Testes satisfeitos levam a promoção da sua Infra

O que testar?

- Sintaxe e semântica das definições
 - A maioria das ferramentas já suporta isso
- Status de serviços
 - Cheque PIDs
 - HeartBeats
- Instalação de softwares
- Tolerância a falhas e capacidade

Exercício

Como você montaria sua infraestrutura para as práticas de infraestrutura como código?

Organize um fluxo de trabalho em que suas mudanças seriam testadas e promovidas entre ambientes!

Ferramentas

- Virtualização e Containers
 - Vagrant
 - Packer
 - Docker
- Provisionamento
 - Cloud Formation
 - TerraForm

Vagrant

- Construção e gestão de máquinas virtuais
 - Baixando templates
 - Iniciando e Parando VMs
 - etc
- Facilita o fluxo de trabalho para desenvolvedores
 - Ex.: vários ambientes de desenvolvimento, um em cada VM instanciada
- Compartilhamento de VMs

Vagrantfile

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|

  config.vm.box = "debian-7.2.0-amd64"

  # Create a private network, which allows host-only access to the machine
  # using a specific IP.
  config.vm.network :private_network, ip: "10.10.10.2"

  # Share an additional folder to the guest VM. The first argument is
  # the path on the host to the actual folder. The second argument is
  # the path on the guest to mount the folder. And the optional third
  # argument is a set of non-required options.
  config.vm.synced_folder ".", "/vagrant", nfs: true

  # Provider-specific configuration so you can fine-tune various
  # backing providers for Vagrant. These expose provider-specific options.
  # Example for VirtualBox:
  #

  config.vm.provider :virtualbox do |vb|

    # Use VBoxManage to customize the VM. For example to change memory:
    vb.customize ["modifyvm", :id, "--memory", "2048"]
    vb.customize ["modifyvm", :id, "--vram", "10"]
    vb.customize ["setextradata", :id, "VBoxInternal/Devices/ahci/0/LUN#0/Config/IgnoreFlu
end

config.vm.provision :chef_client do |chef|
  chef.chef_server_url = "https://api.opscode.com/organizations/ORGNAME"
  chef.validation_key_path = "ORGNAME-validator.pem"
end
```

Provisioning

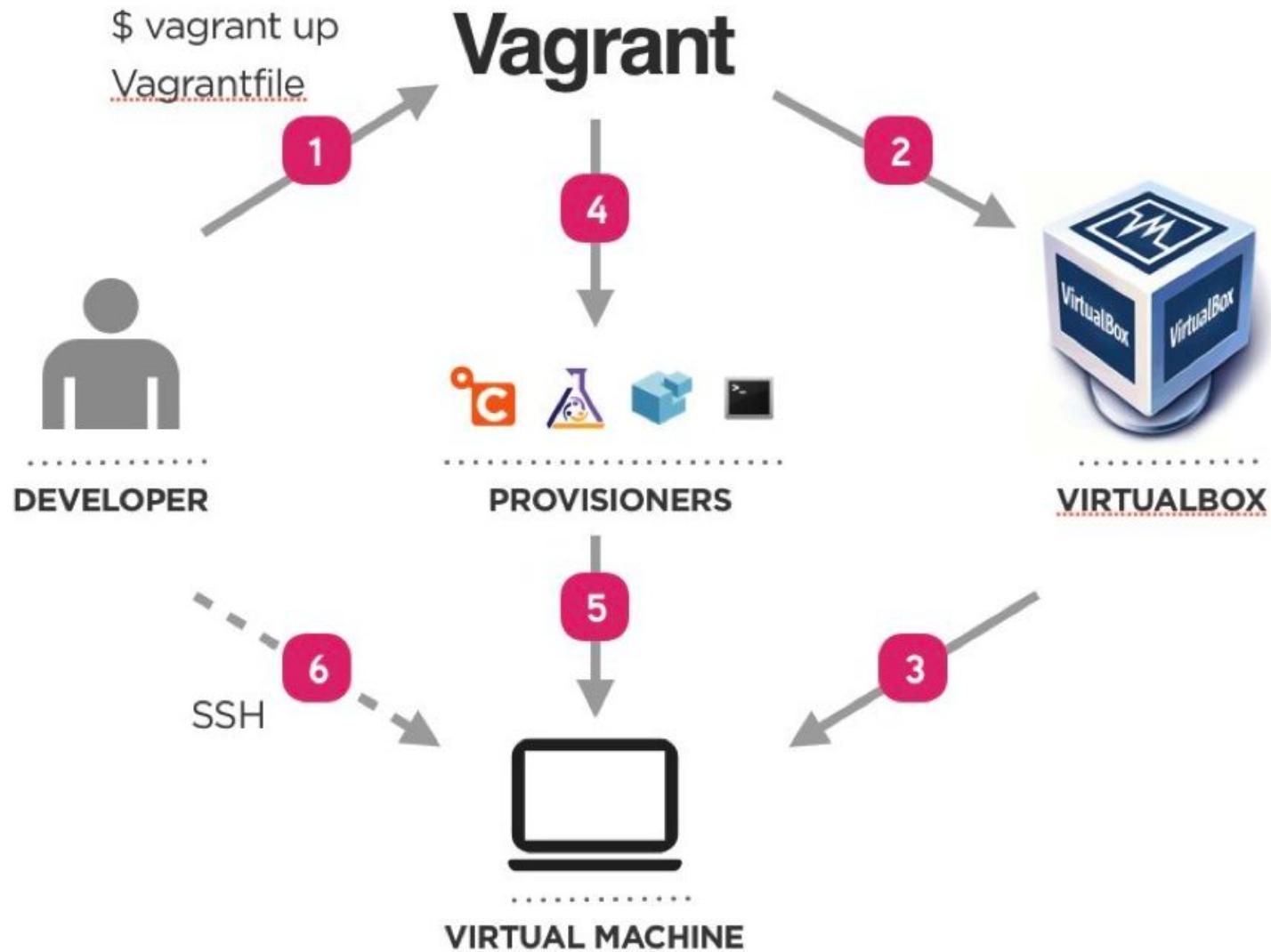
- Método que permite configurar o Box para sua necessidade
- Métodos
 - Arquivo
 - Shell script
 - Ansible, Chef, Salt, etc

```
Vagrant.configure("2") do |config|
  config.vm.provision "shell", inline: "echo foo"

  config.vm.define "web" do |web|
    web.vm.provision "shell", inline: "echo bar"
  end

  config.vm.provision "shell", inline: "echo baz"
end
```

Fluxo



Box

- Template para as VMs que serão instanciadas
- Há um “marketplace” com boxes disponíveis
- Packer
 - Criar imagens de máquinas virtuais para diversas plataformas
 - AMIs
 - VMKD
 - OVF
 - ISO
 - etc

Uso

- Compartilhamento de ambiente virtualizado em uma equipe
- Uma máquina física, vários ambientes de desenvolvimento
- Configuração de várias VMs simultaneamente

Cloud Formation

Características

- Serviço de provisionamento de infraestrutura da AWS
 - Permite focar na aplicação e não em infra
 - Templates, Stacks e ChangeSets
- Configuração de várias VMs simultaneamente
- Integração com mecanismos de CI \ CD da AWS
- Possui uma ferramenta de design

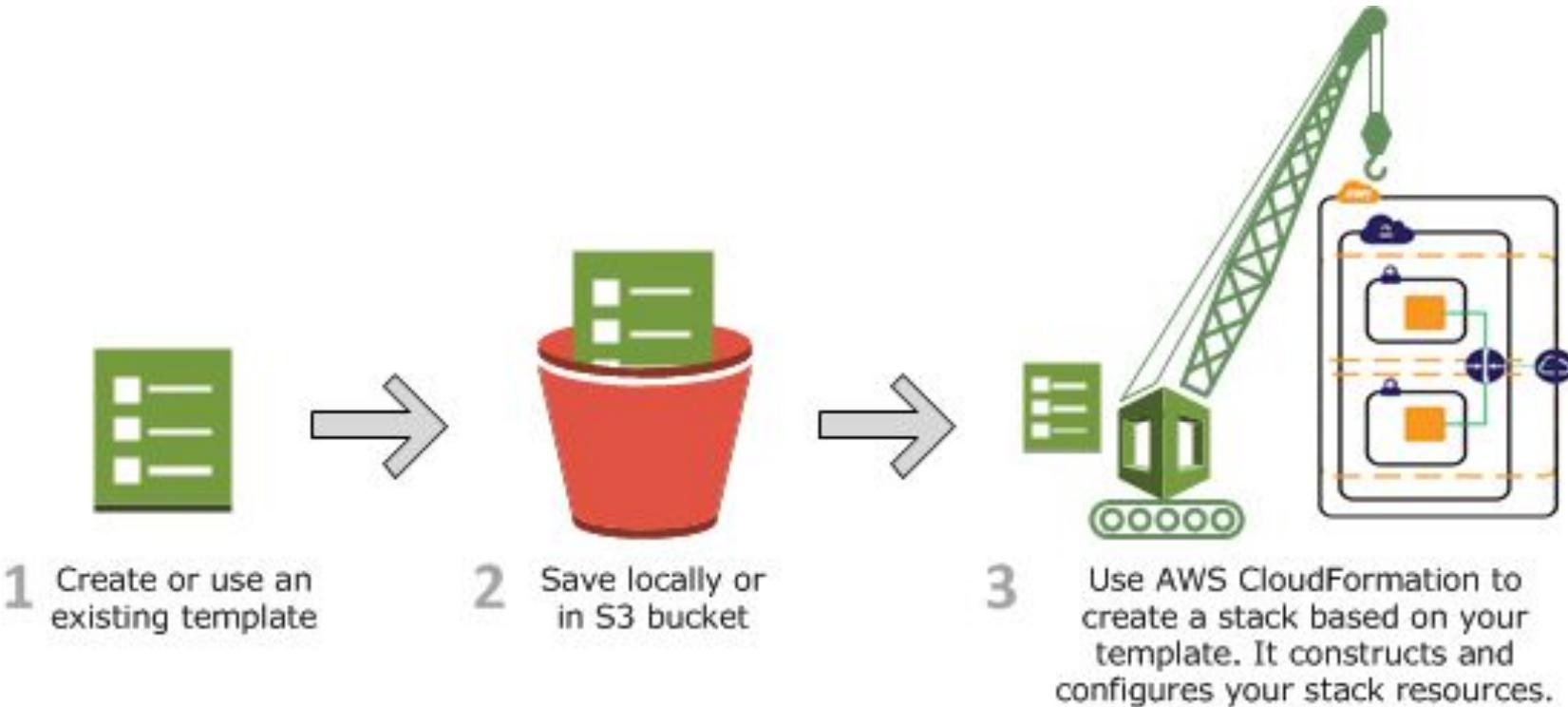
Templates

- Arquivos de descrição da infraestrutura na AWS
 - formatos JSON e YAML
- Principais seções
 - AWSTemplateFormatVersion
 - Metadata
 - Parameters
 - Mappings
 - Conditions
 - Transform
 - **Resources**
 - Outputs

Stacks

- Coleção de recursos tratados como uma só unidade
 - Agrupador
 - formatos JSON e YAML
- Principais seções
 - AWSTemplateFormatVersion
 - Metadata
 - Parameters
 - Mappings
 - Conditions
 - Transform
 - **Resources**
 - Outputs

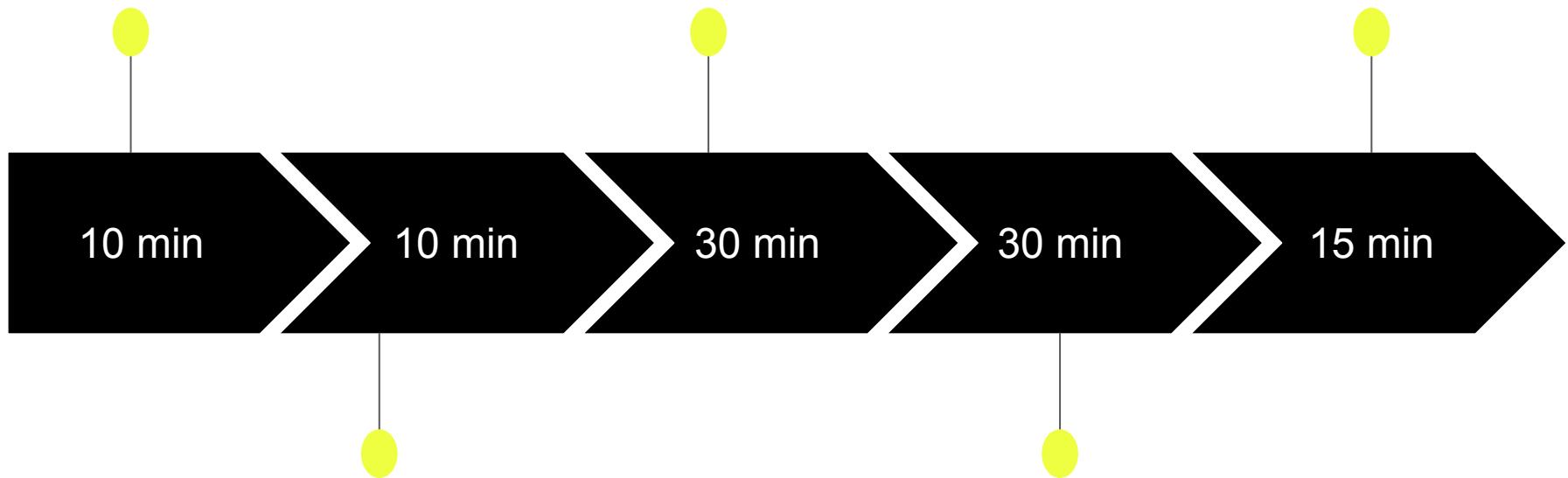
Funcionamento



Digite seu texto aqui
Digite seu texto aqui
Digite seu texto aqui

Digite seu texto aqui
Digite seu texto aqui
Digite seu texto aqui

Digite seu texto aqui
Digite seu texto aqui
Digite seu texto aqui



Digite seu texto aqui
Digite seu texto aqui
Digite seu texto aqui

Digite seu texto aqui
Digite seu texto aqui
Digite seu texto aqui

Lição de casa

Digite seu texto aqui Digite seu texto aqui

1. Digite seu texto aqui
2. Digite seu texto aqui Digite seu texto aqui
Digite seu texto aqui Digite seu texto aqui
Digite seu texto aqui

Digite seu texto aqui Digite seu texto aqui Digite seu texto aqui Digite seu texto aqui Digite seu texto aqui

Gerenciando servidores

