# VAGRANT COOKBOOK

Guia prático para o Vagrant e seus principais Provisioners

Erika Heidi

# Vagrant CookBook (PT-BR)

## Um Guia Prático

Erika Heidi

Esse livro está à venda em http://leanpub.com/vagrantcookbook-ptbr

Essa versão foi publicada em 2014-07-07

# Tweet Sobre Esse Livro!

Por favor ajude Erika Heidi a divulgar esse livro no Twitter!

A hashtag sugerida para esse livro é #vagrantcookbook.

Descubra o que as outras pessoas estão falando sobre esse livro clicando nesse link para buscar a hashtag no Twitter:

https://twitter.com/search?q=#vagrantcookbook

# Conteúdo

# Prefácio

Eu lembro claramente a primeira vez que ouvi falar sobre o Vagrant. Início de 2013, minha segunda visita aos *meetups* do grupo de usuários AmsterdamPHP. A palestra era da amiga Michelle Sanver (a.k.a. Geekie), sobre Open Source. Ela mostrou como era fácil se envolver e contribuir com projetos Open Source, apenas clonando um repositório e rodando o misterioso comando `vagrant up`.

Comecei a usar o Vagrant no dia seguinte.

Alguns meses depois, e não antes de enfrentar uma resistência considerável dos meus colegas de trabalho e do líder da equipe (eles estavam usando servidores remotos para testar as aplicações - upload via FTP para cada modificação), eu consegui introduzir o Vagrant na empresa em que estava trabalhando em Amsterdam. Apesar de ter "sofrido" um pouco para criar o provisionamento usando Puppet, considerando que o projeto era complexo e eu ainda não tinha quase nenhuma experiência no assunto, essa foi uma excelente oportunidade para aprender mais sobre o Vagrant e seus provisioners.

Depois de sair da empresa e voltar a trabalhar com meus projetos pessoais, comecei a submeter palestras para conferências PHP, e naturalmente Vagrant estava no topo da minha lista de assuntos interessantes para falar sobre. Na mesma época, conheci o LeanPub e a excelente plataforma que eles construíram para auto-publicação (self-publishing) - preciso dizer que eu realmente amo todo tipo de serviço que promove trabalhos independentes. Também não é nenhuma novidade o fato de que escrever é uma paixão para mim desde que eu era criança. Juntando os pontos, a idéia era óbvia: eu tinha que escrever um livro sobre minhas experiências com o Vagrant.

Esse livro é baseado em muitos experimentos e pesquisas, escrito por uma usuária entusiástica e realmente curiosa. Tentei juntar nesse livro tudo o que você precisa para ter uma experiência proveitosa com o Vagrant, de uma forma totalmente prática.

Aprecio todo tipo de feedback construtivo que possa fazer esse livro melhor - sugestões, correções, qualquer coisa. Sinta-se à vontade para me contactar no Twitter (@erikaheidi), IRC (erikaheidi, na Freenode) ou via e-mail (erika@erikaheidi.com)

# Introdução

Quantas vezes você já ouviu a frase "*funciona na minha máquina*" ? Aposto que você também já usou essa "desculpa" - sim, acontece. É praticamente impossível lembrar de tudo que você já instalou e de todas as configurações que você já mexeu em sua máquina de trabalho, então geralmente leva algum tempo para descobrir o que deu errado na hora em que o projeto foi compartilhado com outro colega de trabalho, ou pior, quando o projeto foi para produção.

Também é bastante difícil lidar com múltiplos projetos quando eles requerem ambientes de desenvolvimento diferentes - imagine que você está trabalhando em um projeto que usa PHP 5.4, mas em um dado momento surge a necessidade de trabalhar em um projeto legado que usa PHP 5.3.

Se você ainda não está familiarizado com o Vagrant[1], esse é um bom momento para conhecê-lo melhor. Vagrant proporciona um ambiente de desenvolvimento reproduzível e portátil usando máquinas virtuais, tudo configurado em alguns arquivos dentro do repositório do seu projeto - clone o repositório, execute `vagrant up` e seu projeto estará rodando em um ambiente especialmente criado para ele, com todas as dependências necessárias, em alguns minutos. Você nunca será refém da desculpa "funciona na minha máquina" novamente - o ambiente é exatamente o mesmo para todos os desenvolvedores envolvidos no projeto, independente do sistema operacional rodando por trás do Vagrant.

## Vagrant para projetos privados

Com Vagrant, você pode ter um fluxo de trabalho muito mais fácil quando trabalhando em um time; garantir o mesmo ambiente de desenvolvimento e testes para todos os colaboradores irá evitar muitos problemas e tornar o processo de desenvolvimento mais consistente e confiável.

## Vagrant para projetos Open Source

Vagrant possibilita que mais desenvolvedores possam contribuir com o seu projeto Open Source - ter o projeto rodando localmente na sua máquina é tão fácil quanto clonar um repositório e rodar `vagrant up`. Não apenas setando o ambiente de desenvolvimento correto, mas também automatizando processos como instalar um banco de dados, clonar repositórios, adicionar dados no banco e até rodar testes.

[1] http://docs.vagrantup.com/v2/

### Vagrant para devops / administradores de sistema

Se você trabalha com administração de sistemas, Vagrant é a ferramenta ideal para os seus testes. Ele suporta as ferramentas de automação mais populares do mercado - Puppet, Ansible, Chef, dentre outras. Faça experimentos com diversos setups, construa uma infraestrutura com múltiplos servidores e garanta que tudo está funcionando como você esperava, localmente, antes de lidar com servidores "reais".

## O que esperar desse livro

*Vagrant Cookbook* foi idealizado para ser um livro bastante prático, cobrindo Vagrant desde os requerimentos e instalação até conceitos mais avançados, como o uso de múltiplas máquinas virtuais e criação de servidores "reais" em serviços como Digital Ocean e Amazon AWS. Você terá a oportunidade de conhecer, testar e comparar os *provisioners* mais usados da atualidade - Puppet, Chef e Ansible.

Esse livro também irá cobrir algumas dicas importantes para criar projetos Vagrant otimizados; no final você terá um capítulo com *receitas* para as tarefas mais comuns de um *provisionador*, como instalar pacotes, usar templates, executar comandos, dentre outras.

*Vagrant Cookbook* é destinado a usuários iniciantes e intermediários, também servindo como um guia rápido para os *provisionadores* Ansible, Puppet e Chef.

## O que esperamos de você

Esse livro assume que você é um desenvolvedor com alguma experiência em linha de comando, e você sabe como instalar e configurar um servidor Linux - você precisa entender o problema antes de ser capaz de automatizar a solução, certo? As ferramentas que iremos ver nesse livro requerem um conhecimento básico de programação, já que elas trabalham conceitos como variáveis, condicionais e laços.

*Vagrant Cookbook* é um livro sobre Vagrant, para pessoas que se sentem confortáveis o bastante com programação e também com tarefas administrativas, como configurar um servidor Web no Linux.

Os exemplos mostrados nesse livro terão como alvo a criação de servidores PHP, apenas como uma maneira de mostrar exemplos práticos e realistas; você não precisa ser um desenvolvedor PHP (e você não precisa *gostar* de PHP) para aproveitar bem o conteúdo desse livro.

# Primeiros Passos

Este capítulo cobre o básico - terminologia, instalação e uso geral do Vagrant - incluindo como inicializar a sua primeira máquina virtual.

## Como o Vagrant funciona

Vagrant controla o processo de criar uma máquina virtual baseada em suas definições, usando ferramentas de automação como Ansible e Puppet para *provisionar* a customização da máquina virtual - instalar pacotes, obter informações, executar tarefas etc.

Rodando um simples `vagrant up`, uma máquina virtual será preparada de acordo com o que foi definido no projeto, e em alguns minutos o seu projeto estará rodando (por exemplo, uma aplicação Web), acessível através da sua rede local. Você pode logar na máquina usando `ssh` e fazer o que quiser, ela será exatamente igual a uma máquina "de verdade".

Também é possível usar o Vagrant para fazer deploy de servidores em serviços como AWS e Digital Ocean. Nós falaremos sobre isso no capítulo "Tópicos Avançados".

## Terminologia

Antes de avançarmos, é importante conhecer os termos mais comuns usados com o Vagrant.

### Box

Uma box (caixa) é basicamente um pacote que representa um sistema operacional instalado (e alguns pacotes básicos), para um *provedor* específico. O Vagrant irá replicar essa imagem para a sua máquina virtual. Ao definir um projeto, você escolhe qual *box* será a base do seu ambiente de desenvolvimento. Na primeira vez que você rodar o Vagrant com uma *box* nova, ele irá fazer o download da box e importá-la para o seu sistema.

### Host e Guest

A máquina / sistema operacional **Host** é onde o Vagrant está instalado. A máquina **Guest**, como você pode deduzir, é a máquina virtual inicializada pela Host.

## Provedores

De uma maneira geral, o *provedor* (provider) é o software de virtualização responsável por criar as máquinas virtuais que o Vagrant administra. Porém, Vagrant não trabalha apenas com máquinas virtuais; alguns provedores específicos utilizam outros métodos para criação do seu ambiente, como no caso do Docker, que utiliza *containers*. Para simplificar as definições contidas no livro, vamos usar o termo "máquina virtual" ou "VM" (virtual machine, termo original do inglês) para nos referirmos ao ambiente de desenvolvimento gerado pelo *provedor*, independentemente do tipo de provedor referenciado.

VirtualBox é o provedor padrão do Vagrant, mas você também pode usar VMWare, KVM, dentre outros. Para usar outros provedores, normalmente é preciso instalar um plugin.

## Plugins

Um plugin pode adicionar funcionalidades extras ao Vagrant, como suporte a outros provedores, por exemplo.

## Provisionadores

Um *provisionador* (provisioner) irá automatizar o processo de instalação e configuração do seu ambiente, instalando pacotes e executando tarefas em geral. Não é obrigatório usar um provisionador, mas também não faz muito sentido usar o Vagrant sem um - você teria que logar na máquina e instalar o seu ambiente manualmente, da mesma maneira que fazia antigamente (ou você poderia simplesmente usar o VirtualBox sozinho). O Vagrant suporta vários provisionadores diferentes, do mais básico Shell até ferramentas complexas de automação como Chef. Iremos falar mais detalhadamente sobre provisionadores em breve.

## Vagrantfile

O Vagrantfile é o arquivo de configuração do Vagrant, que conterá todas as definições da sua máquina virtual. Normalmente ele fica localizado na raiz do seu projeto. Teremos um capítulo dedicado ao Vagrantfile e suas opções de configuração.

## Diretórios Sincronizados

Os diretórios sincronizados (synced folders) são usados para compartilhar conteúdo entre a máquina **Host** e a máquina **Guest**. Dessa maneira, podemos continuar usando nossa IDE favorita, na máquina Host, para editar os arquivos da aplicação, enquanto usamos a máquina Guest apenas para rodar e testar a aplicação. As modificações são refletidas em tempo real porque o conteúdo do diretório é compartilhado entre as máquinas. A sincronização em tempo real tem um custo em performance, iremos falar sobre isso em um capítulo posterior.

# Requerimentos

Para provisionar sua máquina virtual, o Vagrant precisa de um software de virtualização, como VirtualBox ou VMWare. O VirtualBox é o padrão, por ser gratuito e open source. Iremos trabalhar com o VirtualBox nesse livro. Você precisará ter ambos (Vagrant e VirtualBox) instalados na sua máquina.

Tanto o Vagrant quanto o VirtualBox estão disponíveis para os principais sistemas operacionais (Linux, OSX e Windows). Algumas funcionalidades, porém, podem não estar presentes por padrão em alguns sistemas e podem requerer a instalação de pacotes adicionais - como por exemplo o NFS (network file system) que é usado com os diretórios sincronizados - para usar essa funcionalidade no Linux, você precisa instalar o **nfsd**, que já vem instalado por padrão no OSX (e não é suportado pelo Windows).

# Instalação

Para iniciar, a primeira coisa que você deve fazer é instalar o Vagrant e o VirtualBox. O ideal é obter os pacotes diretamente de suas páginas oficiais, já que gerenciadores de pacotes como o **apt** no Ubuntu provavelmente terão versões antigas desses softwares, o que pode levar a problemas de incompatibilidade. Verifique a página de downloads do Vagrant[2] e a página de downloads do VirtualBox[3], seguindo as instruções de instalação para o seu sistema operacional, para ambos os softwares.

## Obtendo a versão certa do VirtualBox

É recomendado que você verifique a documentação do Vagrant[4] para checar qual versão do VirtualBox é atualmente compatível. Normalmente você não terá problemas baixando as versões mais recentes de ambos, mas em alguns casos, quando o VirtualBox lança uma nova versão, incompatibilidades podem acontecer.

# Atualizando o Vagrant

O Vagrant sempre procura manter compatibilidade com suas versões anteriores, de maneira que você não terá problemas ao atualizar para versões mais recentes. É recomendado que você mantenha a sua versão do Vagrant sempre atualizada, pois os updates costumam trazer importantes correções e novos recursos.

[2] http://www.vagrantup.com/downloads.html
[3] https://www.virtualbox.org/wiki/Downloads
[4] http://docs.vagrantup.com/v2/virtualbox/index.html

## Verificando se a sua versão está atualizada (1.6+)

A partir da versão 1.6, você pode verificar facilmente se a sua versão instalada é a mais recente, com o novo comando **version**. Basta executar:

```
$ vagrant version
```

Se a sua versão estiver atualizada, você verá um output similar a este:

```
Installed Version: 1.6.1
Latest Version: 1.6.1

You're running an up-to-date version of Vagrant!
```

Se a sua versão instalada não for a mais recente, a saída do comando irá informá-lo qual é a versão mais atual para download:

```
Installed Version: 1.6.1
Latest Version: 1.6.3

To upgrade to the latest version, visit the downloads page and
download and install the latest version of Vagrant from the URL
below:

  http://www.vagrantup.com/downloads.html

If you're curious what changed in the latest release, view the
CHANGELOG below:

  https://github.com/mitchellh/vagrant/blob/v1.6.3/CHANGELOG.md
```

# Vagrant - Comandos

Essa é uma referência rápida para os principais comandos do Vagrant:

| comando | descrição | uso comum |
|---|---|---|
| **up** | Inicializa a máquina virtual e roda o(s) provisionador(es) | Quando a máquina não está rodando ainda |
| **reload** | Reinicia a máquina virtual | Quando você faz modificações no Vagrantfile |
| **provision** | Roda apenas o(s) provisionador(es) | Quando você faz modificações no provisionamento |
| **init** | Cria um novo Vagrantfile baseado em uma box | Quando você quer gerar um Vagrantfile |
| **halt** | Desliga a máquina virtual | Quando você quer desligar a VM |
| **destroy** | Destrói a máquina virtual | Quando você quer começar do zero |
| **suspend** | Suspende a execução da máquina virtual | Quando você quer salvar o estado da VM |
| **resume** | Retoma a execução da máquina virtual | Quando você quer retomar a execução de uma máquina virtual que foi suspensa anteriormente |
| **ssh** | Faz login via SSH (não pede senha ou login) | Quando você quer fazer modificações manuais ou debugar |

## Status e Controle Global (Vagrant 1.6+)

A partir da versão 1.6, Vagrant permite que você verifique quais ambientes foram criados e estão ativos no momento, e você também pode controlar esses ambientes diretamente a partir de qualquer diretório na sua máquina. Antes da versão 1.6, era bastante difícil saber quantas e quais VMs estavam ativas em um dado momento, e para controlar um ambiente você teria de acessar o diretório onde o ambiente foi criado pela primeira vez (onde o Vagrantfile daquele ambiente está localizado).

Para listar todos os ambientes Vagrant que estão ativos, use:

```
$ vagrant global-status
```

Se você tiver um ou mais ambientes Vagrant rodando (isso inclui máquinas que foram suspensas ou desligadas), você verá um output similar a este:

```
id       name    provider   state    directory
-----------------------------------------------------------
037588f  default virtualbox running  /projects/project1
f47c729  default virtualbox poweroff /projects/project2
```

Como você pode ver, o primeiro ambiente está com o status "*running*" (rodando), enquanto o segundo ambiente está com o status "*poweroff*" (desligado).

Agora, por exemplo, se você quiser suspender um dos ambientes, você só precisa adicionar o **id** daquele ambiente no final do comando:

```
$ vagrant suspend f47c729
```

A mesma abordagem é válida para os outros comandos do Vagrant - você só precisa adicionar o **id** do ambiente, obtido através do comando `global-status`.

Nota: uma VM que foi apenas **suspensa** será listada com o estado "*running*". Apenas ambientes que foram realmente desligados (com `vagrant halt`) serão listados como "poweroff".

O status e controle global funcionará apenas com ambientes que foram **criados** com versões a partir da **1.6**. Se você tem um ambiente já criado (suspenso ou desligado) com versões anteriores, você precisará destruí-lo com **vagrant destroy** e criá-lo novamente, para que ele seja listado pelo `global-status`.

# Seu primeiro Vagrant Up

Vamos inicializar nossa primeira máquina virtual usando o Vagrant. A primeira coisa que iremos precisar é de um **Vagrantfile**. Você pode criar esse arquivo manualmente, mas você também pode fazer com que o Vagrant gere um Vagrantfile automaticamente para você, baseado na *box* que você deseja usar. Para iniciarmos, vamos usar o Vagrantfile gerado automaticamente. Não se preocupe muito com isso agora, pois no próximo capítulo iremos estudar o Vagrantfile e suas opções de configuração mais comuns.

Primeiramente, crie uma pasta para começarmos nossos testes. Através da linha de comando, acesse essa pasta e execute:

### Vagrant 1.5+

```
$ vagrant init hashicorp/precise64
```

Esse comando irá apenas criar um Vagrantfile padrão, setando uma única opção:

```
config.vm.box = "hashicorp/precise64"
```

A opção `config.vm.box` definirá qual box seu ambiente virtual irá usar. Com as versões mais recentes do Vagrant, a partir da 1.5, essa opção normalmente contém um identificador para uma box hospedada na **Vagrant Cloud**[5]. Vagrant Cloud é um novo recurso lançado com a versão 1.5 - um local centralizado para descobrir e compartilhar boxes para o Vagrant, provendo também versionamento das boxes.

### Vagrant < 1.5

Para versões anteriores do Vagrant (menores que 1.5), sem suporte à Vagrant Cloud, normalmente precisamos também prover a URL da box que queremos usar - apenas dessa maneira o Vagrant saberá de onde fazer o download da box. Para inicializar um Vagrantfile básico usando a mesma box do exemplo anterior (Ubuntu 12.04 64 bits) você deve usar:

```
$ vagrant init precise64 http://files.vagrantup.com/precise64.box
```

O comando acima criará um Vagrantfile com as seguintes opções:

```
config.vm.box = "precise64"
config.vm.box_url = "http://files.vagrantup.com/precise64.box"
```

## Rodando sua nova VM

Agora que temos um Vagrantfile, é hora de iniciarmos nossa máquina virtual. Na linha de comando, no mesmo diretório, execute:

```
$ vagrant up
```

O processo de baixar e importar a *box* é feito automaticamente quando você inicializa a máquina virtual. Na primeira vez que você roda o Vagrant com uma *box* nova, ela será baixada e importada para o seu sistema - o processo pode levar vários minutos, dependendo da sua internet. Depois de baixar e importar a *box*, a máquina será inicializada e você verá um output como este:

[5] https://vagrantcloud.com

```
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'hashicorp/precise64' could not be found. Attempting to find and\
 install...
    default: Box Provider: virtualbox
    default: Box Version: >= 0
==> default: Loading metadata for box 'hashicorp/precise64'
    default: URL: https://vagrantcloud.com/hashicorp/precise64
==> default: Adding box 'hashicorp/precise64' (v1.1.0) for provider: virtualbox
    default: Downloading: https://vagrantcloud.com/hashicorp/precise64/version/2/\
provider/virtualbox.box
==> default: Successfully added box 'hashicorp/precise64' (v1.1.0) for 'virtualbo\
x'!
==> default: Importing base box 'hashicorp/precise64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'hashicorp/precise64' is up to date...
==> default: Setting the name of the VM: vagrant-lab_default_1394635286089_95645
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 => 2222 (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
    default: The guest additions on this VM do not match the installed version of
    default: VirtualBox! In most cases this is fine, but in rare cases it can
    default: prevent things such as shared folders from working properly. If you \
see
    default: shared folder errors, please make sure the guest additions within the
    default: virtual machine match the version of VirtualBox you have installed on
    default: your host and reload your VM.
    default:
    default: Guest Additions Version: 4.2.0
    default: VirtualBox Version: 4.3
==> default: Mounting shared folders...
    default: /vagrant => /media/export/Projects/vagrant-lab
```

Ta-Da! Sua primeira máquina virtual Vagrant está rodando. Agora, faça login usando:

```
$ vagrant ssh
```

Você perceberá que se trata de uma máquina Ubuntu como qualquer outra, conectada à internet (desde que a sua máquina Host esteja conectada também), tudo funcional. Se você executar um **`ls /vagrant`**, irá perceber que, por padrão, o diretório atual (onde o Vagrantfile está presente, na máquina **Host**) é compartilhado entre as máquinas Host e Guest, então qualquer arquivo colocado dentro dessa pasta estará acessível dentro da máquina virtual, na pasta **`/vagrant`**.

No próximo capítulo, veremos como customizar melhor o Vagrantfile e como iniciar com o uso dos provisionadores, automatizando tarefas que serão executadas logo depois que a máquina é inicializada pelo Vagrant.