O QUE É VAGRANT?

O Vagrant é uma ferramenta de código aberto que nos ajuda a automatizar a criação e o gerenciamento de Máquinas Virtuais. Resumindo, podemos especificar a configuração de uma máquina virtual em um arquivo de configuração simples, e o Vagrant cria a mesma máquina virtual usando apenas um comando simples. Ele fornece interfaces de linha de comando para automatizar essas tarefas.

Máquina virtual é uma máquina que não existe fisicamente, mas pode ser usada como um computador físico. Qualquer tarefa que pode ser realizada em uma máquina física também pode ser executada em uma máquina virtual. Mas a máquina virtual é construída sobre um sistema físico e várias máquinas virtuais podem ser criadas em um único computador físico. Todas as máquinas virtuais compartilham o mesmo hardware, mas cada uma delas pode ter um sistema operacional separado. O sistema físico que hospeda todas as máquinas virtuais é chamado de **computador host** . O meio que separa o hardware do computador host e os ambientes virtuais é algo chamado **hipervisor** ou **Hyper-V**.

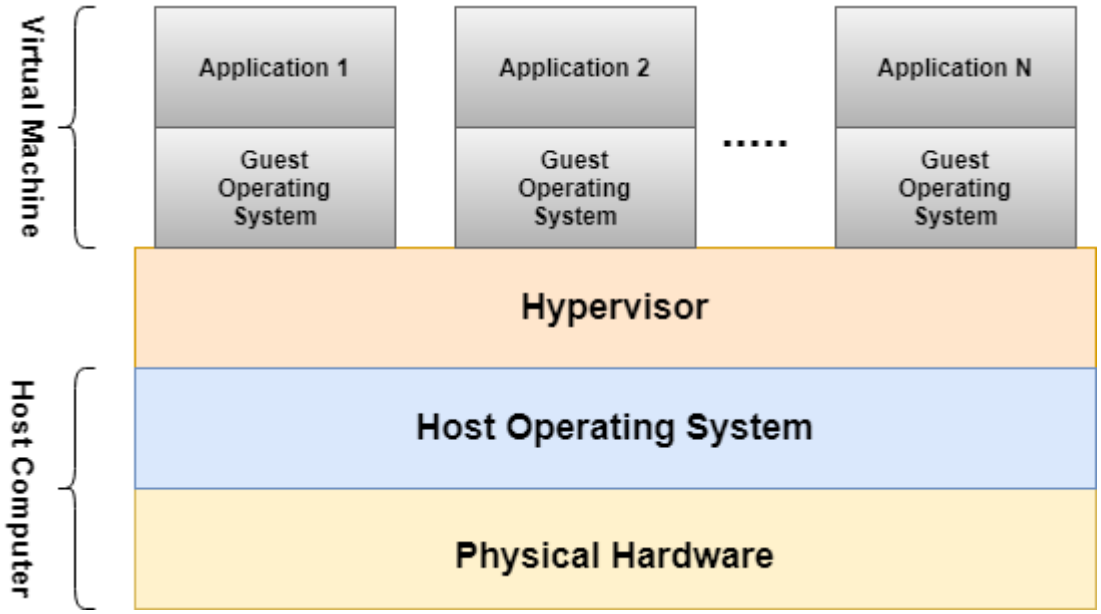


Fig: Estrutura da Máquina Virtual

Cada Máquina Virtual deve ter sua própria configuração como sistema operacional, CPUs, RAM, Hard Disk Memory, networking, etc. E a criação de tais VMs, configurando manualmente todas as propriedades é realmente uma tarefa agitada. Nesse cenário, o Vagrant entra em cena.

Por que Vagrant?
Um aplicativo consiste em vários componentes que precisam ser configurados

corretamente para executar o aplicativo. Por exemplo, um aplicativo da web moderno pode ter componentes como Java, JavaScript, Python, etc. como linguagem, MySQL, Oracle, MongoDB, etc. como bancos de dados, outros componentes como servidor da web, balanceador de carga, gateway de API, fila de mensagens, etc. com base nos requisitos.

Antes do Vagrant, todos esses componentes precisam ser configurados manualmente. Durante o processo de configuração, muitos problemas são enfrentados-

- Em cada máquina, a configuração deve ser feita separadamente, o que leva muito tempo.
- A configuração manual pode estar incorreta, o que precisa ser depurado e corrigido todas as vezes.
- O ambiente de desenvolvimento, teste e produção deve ser idêntico. Mas devido a esta instalação manual e configuração dos componentes, pode haver uma ligeira diferença que nos traz muita dor, pois, em tal cenário, o aplicativo pode rodar em um ambiente de desenvolvimento, mas enfrenta problemas no de produção.

O Vagrant é a solução moderna para todos esses problemas. Em vez de configurar todos os componentes manualmente, o Vagrant nos fornece o recurso de usar um arquivo de configuração (chamado Vagrantfile), onde todos os componentes de software necessários e suas informações de configuração são especificados. Portanto, enquanto este mesmo arquivo de configuração é executado em várias máquinas, o Vagrant cria a Máquina Virtual no topo de cada máquina física, instala e configura todos os componentes mencionados automaticamente e fornece uma Máquina Virtual pronta para começar a trabalhar. E uma vez que isso fornece uma máquina virtual, não há necessidade de se preocupar se um componente de software seria executado no sistema operacional Windows ou Linux e qual deveria ser sua configuração e também do desenvolvedor, o controle de qualidade funciona em uma máquina separada, mas ambas as máquinas deveriam ter uma configuração completamente idêntica.

Terminologias do Vagrant: Antes de entrar nos detalhes da instalação e como usar o Vagrant, vamos primeiro discutir as terminologias básicas relacionadas ao Vagrant.

1. Vagrant Box: A unidade básica de configuração do Vagrant é a Vagrant Box. Assim como o Docker Image, o Vagrant Box é uma imagem independente do sistema operacional . Mais especificamente, é uma Máquina Virtual

empacotada. Em vez de instalar um sistema operacional e todos os componentes de software dentro de uma VM manualmente,

- A caixa vagrant é uma imagem base pronta de um ambiente de máquina virtual.
- Por exemplo, se houver necessidade de obter alguma VM com todas as configurações padrão de desenvolvimento de aplicativos Spring Boot, pode-se obter o mesmo Vagrant Box. Tudo o que é necessário fazer é baixar o Vagrant Box e executá-lo. O Vagrant cria a VM e o trabalho de desenvolvimento pode ser iniciado imediatamente.
- Muito do Vagrant Box está presente no **catálogo de caixas do Vagrant Cloud** , que podemos usar como imagem base para nossa própria máquina virtual.

2. Arquivo Vagrant: O Vagrant mantém um arquivo de configuração, chamado **Vagrantfile** , onde todas as configurações de uma VM são mencionadas. E o Vagrant cria a Máquina Virtual com a mesma configuração mencionada no arquivo. Mesmo que haja necessidade de instalar algum software na VM, pode-se especificar o mesmo no Vagrantfile, e o Vagrant baixa e instala o mesmo para nós.

Vejamos as etapas para provisionar uma VM usando o Vagrant.

Instalação

Passo 1: Baixe o Vagrant com base no seu sistema operacional e instale-o no sistema.
Etapa 2: Verifique a instalação do vagrant usando o comando ***vagrant -v*** no prompt de comando. Ele irá mostrar a versão do nosso vagrant instalada conforme abaixo.

Etapa 3: Baixe o Virtual Box com base em seu sistema operacional e instale-o no sistema.

Configuração do projeto Vagrant

Passo 1: Crie uma pasta onde queremos salvar todos os arquivos relacionados ao Vagrant.
Passo 2: Crie um arquivo chamado **Vagrantfile** para mencionar a configuração da VM. Como esta é a primeira vez que o Vagrant está sendo usado após a instalação, é aconselhável deixar que o Vagrant crie o arquivo para nós com configuração mínima. E a modificação pode ser feita mais tarde.
Etapa 3: Abra o PowerShell (para Windows) ou Terminal (para Linux) e vá para o local da pasta que foi criada na etapa 1.
Etapa 4: Execute o comando ***vagrant init ubuntu/focal64*** e deixe a execução ser concluída.
Este comando inicializará o diretório com o Vagrant Box especificado (ubuntu/focal64). Encontraremos o Vagrantfile criado neste local. Se aberto, você verá que alguma configuração de amostra já foi mencionada com exemplos adequados e todos eles estão comentados. Qualquer pessoa pode conferir para construir uma Máquina Virtual com uma configuração mais específica. Uma informação importante mencionada no arquivo pode ser vista.

```
config.vm.box = "ubuntu/focal64"
```

Esta é a Vagrant Box, que foi mencionada durante a inicialização.

Inicialize a máquina virtual usando o Vagrant:

```
vagrant up
```

Este comando pegará o Vagrantfile e provisionará uma VM com todas as configurações mencionadas. Por enquanto, ele inicializa uma máquina virtual com uma versão Ubuntu-16.04 do sistema operacional.

SSH na máquina virtual

Etapa 1: Agora, a VM foi criada. Então, para entrar nisso, execute o comando ***vagrant ssh***. Depois de executar este comando, você agora está dentro da VM recém-criada. Agora, tudo pode ser feito dentro da VM por meio deste terminal. A tela a seguir ficará visível-

```
PS E:\vagrant-project> vagrant ssh
Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.4.0-209-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


This system is built by the Bento project by Chef Software
More information can be found at https://github.com/chef/bento
vagrant@vagrant:~$
```

Etapa 2: Para fazer logout da Máquina Virtual, execute o comando ***logout*** . Novamente, o terminal voltará para a máquina host.

```
vagrant@vagrant:~$ logout
Connection to 127.0.0.1 closed.
PS E:\vagrant-project>
```

Limpar o Vagrant:

Etapa 1: Para desligar a VM usando o Vagrant, execute o comando ***vagrant halt*** . Este comando desligará a VM e, novamente, para ligá-la,
execute ***o*** comando ***vagrant up***.

```
PS E:\vagrant-project> vagrant halt
==> default: Attempting graceful shutdown of VM...
PS E:\vagrant-project>
```

Etapa 2: Para desligar uma máquina virtual, mantendo seu estado atual, execute o ***vagrant suspend.*** Nesse caso, quando a VM for reiniciada usando o ***vagrant up*** , o sistema iniciará da mesma posição em que parou. Todos os trabalhos não salvos podem ser restaurados.
Mas, neste caso, a VM não liberará os recursos da máquina host, mesmo que leve um pouco mais de espaço em disco para armazenar o estado atual de sua própria RAM dentro da máquina host.

```
PS E:\vagrant-project> vagrant suspend
==> default: Saving VM state and suspending execution...
PS E:\vagrant-project>
```

Etapa 3: para excluir a VM com todos os seus recursos, execute ***vagrant destroy*** . Este comando desligará a VM e a excluirá do sistema host.

```
PS E:\vagrant-project> vagrant destroy
    default: Are you sure you want to destroy the 'default' VM? [y/N] y
==> default: Forcing shutdown of VM...
==> default: Destroying VM and associated drives...
PS E:\vagrant-project>
```

Configurando o projeto de amostra na VM usando o Vagrant:

Vamos instalar um servidor web dentro da VM e acessar o mesmo de seu computador host. Siga os passos abaixo:

Passo 1: Faça ***vagrant ssh,*** para ir para a VM criada.
Etapa 2: dentro da VM, instale um servidor web, digamos Nginx, manualmente. Execute os comandos abaixo-

```
sudo apt update
sudo apt install nginx
```

Etapa 3: verificar se o serviço Nginx está em execução ou não usando o comando

```
systemctl status nginx
```

Se o serviço não estiver em execução, inicie o serviço usando o comando-

```
sudo service nginx start
```

Nota- A instalação acima também pode ser feita com o Vagrantfile, mas por enquanto, vamos usar a funcionalidade **vagrant ssh** .

Etapa 4: vá para **/var/www/html** e crie um arquivo HTML de amostra, diga **index1.html** conforme abaixo:

```
<!DOCTYPE html>
<html>
  <body>
    <h2 style = "color: green"> Welcome to Bootcamp DevOps </h2>
    <body>
</html>
```

Etapa 5: agora o Nginx está sendo executado em sua porta padrão 80 dentro da VM. Mas para acessar o servidor Nginx da VM da máquina host, é necessário mapear a porta 80 da VM com alguma porta no computador host. E esse mapeamento é chamado de **Port Forwarding**. O encaminhamento de porta pode ser feito manualmente usando o Virtual Box Manager. Mas como o Vagrant está sendo usado aqui, vamos fazer isso usando o Vagrantfile.

Etapa 6: Abra o Vagrantfile e adicione a linha

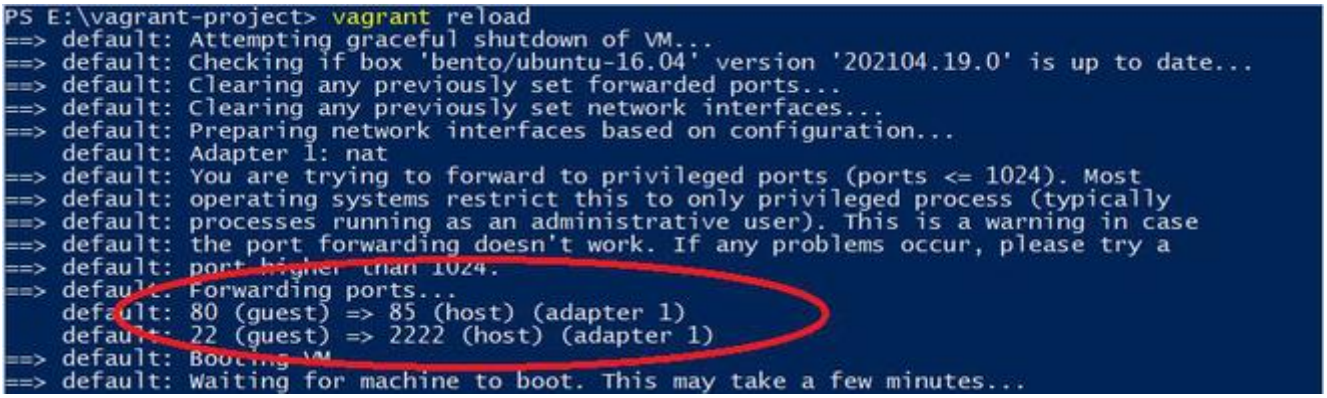
```
config.vm.network "forwarded_port", guest: 80, host: 85
```

depois da linha

```
config.vm.box = "ubuntu/focal64"
```

e salve o arquivo.

Aqui, basicamente, a porta 80 da VM é mapeada com a porta 85 do sistema host.
Etapa 7: Saia da VM no PowerShell ou terminal e execute o comando ***vagrant reload*** para recarregar a nova configuração que adicionamos no Vagrantfile. Durante o recarregamento, pode-se ver a porta encaminhada conforme abaixo. No instantâneo abaixo, a primeira porta encaminhada é a que foi adicionada recentemente e a segunda é o padrão para conectividade SSH.



Etapa 8: verifique se o Nginx está em execução ou não. Agora, o servidor Nginx da VM pode ser acessado usando a porta 85 do sistema host. O arquivo HTML criado acima pode ser recuperado, usando o URL- **localhost:85/index1.html**

Vantagens do Vagrant:

- O Vagrant é gratuito e de código aberto.
- Ele oferece uma estrutura de projeto extremamente eficiente de um ambiente de desenvolvimento / teste.
- O Vagrant tem uma grande variedade de comunidades e plug-ins.

Desvantagens do Vagrant:

- As atualizações de sintaxe têm grandes repercussões na compatibilidade do plugin
- O apoio da comunidade é bom, mas é um processo demorado.
- As principais atualizações introduziram alguns bugs sérios com consequências nos projetos dev.