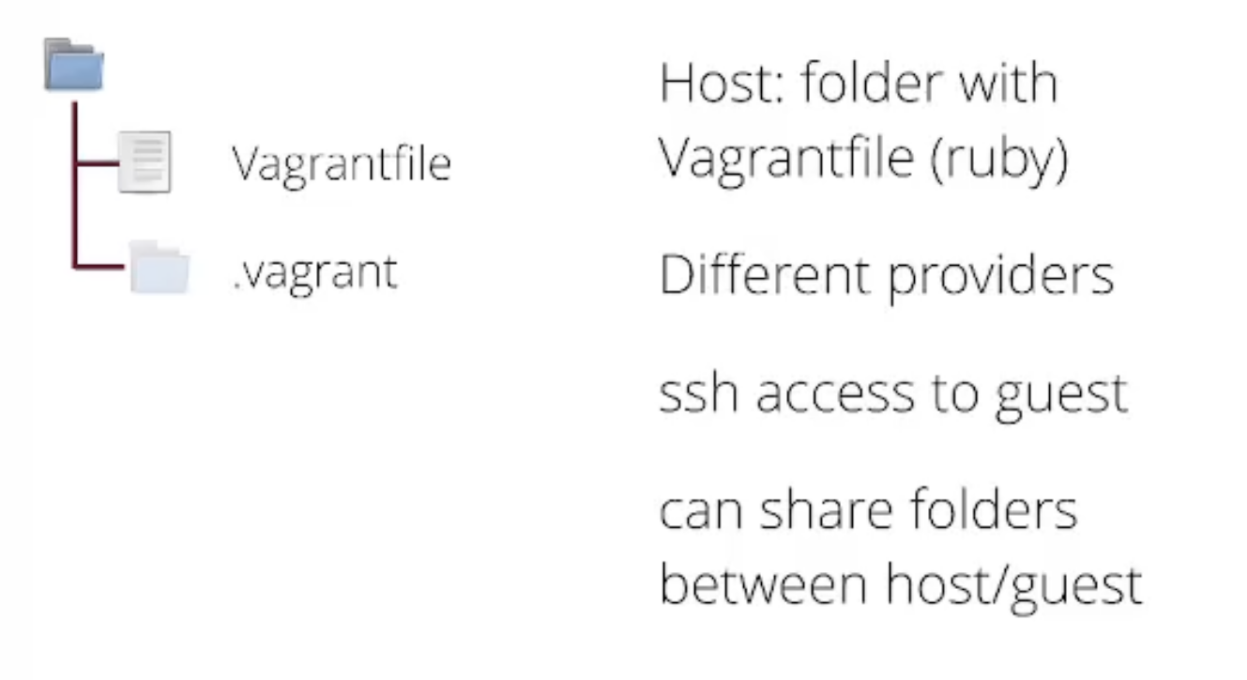
Vagrant (1)

- this is an example of **Virtualisation Software**
  - i.e. software that allows your personal operating system to act as though you were running a different operating system entirely (virtual machine)
    - running an operating system within an operating system

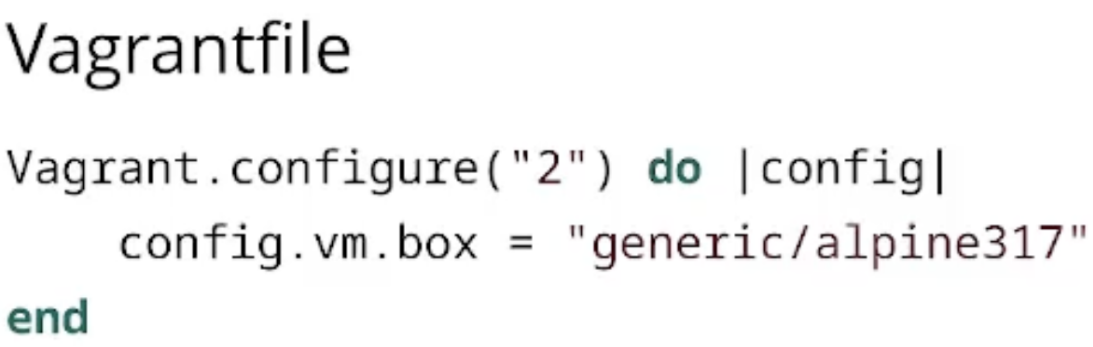
what are some uses/benefits of this:

- emulating a different stack
  - to play a game built on a different OS
- when developing software
  - if you know your software runs in a specific environment, virtual machines aid in **portability** if you are able to simple hand others that specific environment so they can run your software
- **muct cheaper** to have virtualisation model to be able to create numerous differen operating systems

Setting up Vagrant



- you need a directory that contains a **Vagrantfile**
  - written in a language called: ruby



to launch vagrant

```
vagrant up
```

- this will launch a VM with the specification in the local Vagrantfile

Vagrant Commands:

```
vagrant up        // starts the machine

vagrant ssh       // logs you in

vagrant halt      // stops the machine

vagrant reload    // stops+starts the machine for config update

vagrant destroy   // deletes the machine
```

- all these commands require a Vagrantfile in the **current directory** else they will not work

Logging into the machine

- after launching the machine it is time to log in using `vagrant ssh`
- youll now find yourself in a shell with a different prompt to the usual one, if you use `whoami` command it will tell you which user you are i.e. vagrant user

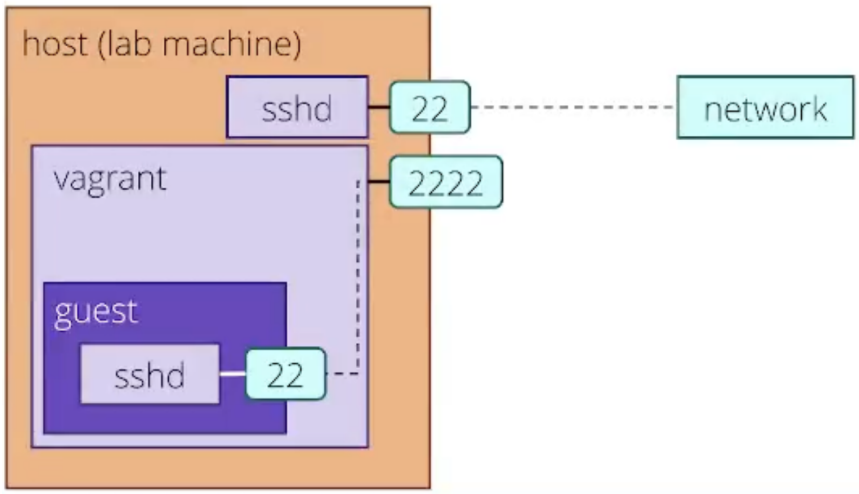
```
$ vagrant ssh
alpine317:~$
alpine317:~$ whoami
vagrant
alpine317:~$ exit
logout
Connection to 127.0.0.1 closed.
$
```

```
$ vagrant up
...
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host)
...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    default: Vagrant insecure key detected.
Vagrant will automatically replace this with a
newly generated keypair for better security.
    default: Inserting generated public key within
guest...
```

- this output when you launch a new VM says that `port 22 (guest) => port 2222 (host)`
  - host is the lab machine or whereever you are running vagrant

ssh and port forwarding

- vagrant is starting up the VM we have specified, one part of this specification is that the machine is going to run the SSH daemon (SSHD)
- by **default** SSHD runs on **port 22** and vagrant is letting our guerst LINUX machine *believe* that is what port it is running on
- it has mapped the Virtual Machine's port 22 to our real machine's port 2222
    - this istn open to the network and would'nt interfere with the host machines own SSHD instance

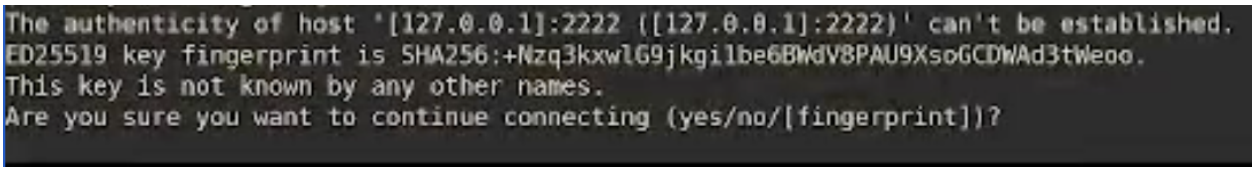


you could also ssh into the new virtual machine you have just launched with vagrant up by:

```
ssh -p 2222 vagrant@127.0.0.1
```

- this is saying you want to connect to port 2222 (`-p 2222`)
- you know the username is `vagrant`
- and the host you're connecting to is `127.0.0.1` which just means your own machine

Doing this gives you the challenge youd expect to see for connections to new machines:



- the SSH client is asking us if we trust this new SSHS intance, which is using a key we have never seen before

if you said *yes* to connecting you would be faced with another problem: **you dont have the key to access this machine** even thought it is a virtual machine you just started

This is because the vagrant machine has been set up to use key-based ssh login

- running vagrant up creates a **new** key pair and stored the private key needed to login somewhere in its config detaisl
- so it is not using one of your exisitng private keys, it has created one of its own
- you could find where the key is stored by vagrant and tell ssh to use it for login by using the `-i` flag for ssh: `ssh -i`

Vagrant saves us from doing this however by using `vagrant ssh` which just autoimatically finds all of the config details for us which allow us to ssh into the VM it is running

Storage of VMs

Where does the content of virtual machines go?

For normal use, i.e. on your own machine

- it would go into a directory:

LINUX: `~/.vagrant.d`

Windows: `C:\Users\NAME\.vagrant.d`

Storage on lab machines

VMs are stored in `/tmp` folder so may disappear if the physical lab machine you are using is switched off or rebooted

- they are also not visible from other lab machines

Alpine Linux

This is the linux distribution you have installed by using vagrant

- minimal distrubution with **minimally** installed by default
- strong on security
- small: it can fit in 8mb of memory meaing it is lightweight and portable

Saving the state of the VM and loading it into new VM

```
// halt the VM if it is running:

vagrant halt


// package up the vm:

vagrant package --output ~/vagrant_boxes/[BOX NAME].box



// to then create a new VM and import a specific box:

vagrant box add <box-name> <path-to-box-file>

// e.g. in my file system:

vagrant box add [BOX_NAME] ~/vagrant_boxes/[BOX_NAME].box
```

Initialize a New Vagrant Environment: After adding the box,
you can initialize a new Vagrant environment using this box.
Create a new Vagrantfile in your current directory (or use an existing one) and configure it to use the box you just added. Open the Vagrantfile in a text editor and set the config.vm.box to the name you gave your box:

```
config.vm.box = "my_project_box"
```

then load up the VM again:

```
vagrant up
```

- this boots up a new instance of the VM using the box we selected