5. Vagrant and Linux Servers

1. Intro to Vagrant
2. Create VM automatically
3. Vagrant commands
4. Vagrant Networking
5. Provisioning
6. RAM, CPU & Disk
7. Multi VM Vagrantfile
8. Documentation

Vagrant for VM's

- No OS Installations
- VM Setup through Images(vagrant boxes)
- Images/Boxes available in vagrant cloud
- Manage VM's with a file(Vagrantfile)
- VM changes automatic through Vagrant file
- Vagrant commands to manage VM's
- Provisioning VM/Executing commands and scripts
- Etc.

Commands

vagrant global-status : to show the status of all the vagrant machines

`ls -a` will show all the hidden files and directories

Messing with Vagrantfile configuration

```
# Create a private network, which allows host-only access to the machine
# using a specific IP.
 config.vm.network "private_network", ip: "192.168.33.10"

# Create a public network, which generally matched to bridged network.
# Bridged networks make the machine appear as another physical device on
# your network.
 config.vm.network "public_network"
```

```
  config.vm.provider "virtualbox" do |vb|
#   # Display the VirtualBox GUI when booting the machine
#   vb.gui = true
#
#   # Customize the amount of memory on the VM:
    vb.memory = "1600"
    vb.cpus = 2
  end
```

to apply changes since it is already running we will

do **vagrant reload**

```
hello world@DESKTOP-1BL5LTS MINGW64 /c/vagrant-vms/ubuntu18
$ vagrant reload
==> default: Attempting graceful shutdown of VM...
==> default: Checking if box 'ubuntu/bionic64' version '20220518.0.0' is up to date...
==> default: Clearing any previously set forwarded ports...
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
    default: Adapter 2: hostonly
    default: Adapter 3: bridged
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> 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: 5.2.42
    default: VirtualBox Version: 6.1
==> default: Configuring and enabling network interfaces...
==> default: Mounting shared folders...
    default: /vagrant => C:/vagrant-vms/ubuntu18
==> default: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> default: flag to force provisioning. Provisioners marked to run always will still run.
```

to see the RAM size

`free -m`

```
vagrant@ubuntu-bionic:~$ free -m
              total        used        free      shared  buff/cache   available
Mem:           1550          87        1266           0         197        1323
Swap:             0           0           0
```

`ifconfig`

```
vagrant@ubuntu-bionic:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::7b:b6ff:fed3:df31  prefixlen 64  scopeid 0x20<link>
        ether 02:7b:b6:d3:df:31  txqueuelen 1000  (Ethernet)
        RX packets 693  bytes 85992 (85.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 495  bytes 84806 (84.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.33.10  netmask 255.255.255.0  broadcast 192.168.33.255
        inet6 fe80::a00:27ff:fe95:ad30  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:95:ad:30  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10  bytes 796 (796.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s9: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.72  netmask 255.255.255.0  broadcast 192.168.100.255
        inet6 fe80::a00:27ff:fe30:37bd  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:30:37:bd  txqueuelen 1000  (Ethernet)
        RX packets 52  bytes 3674 (3.6 KB)
        RX errors 0  dropped 47  overruns 0  frame 0
        TX packets 8  bytes 1202 (1.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 712 (712.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 712 (712.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Vagrant Sync Directores

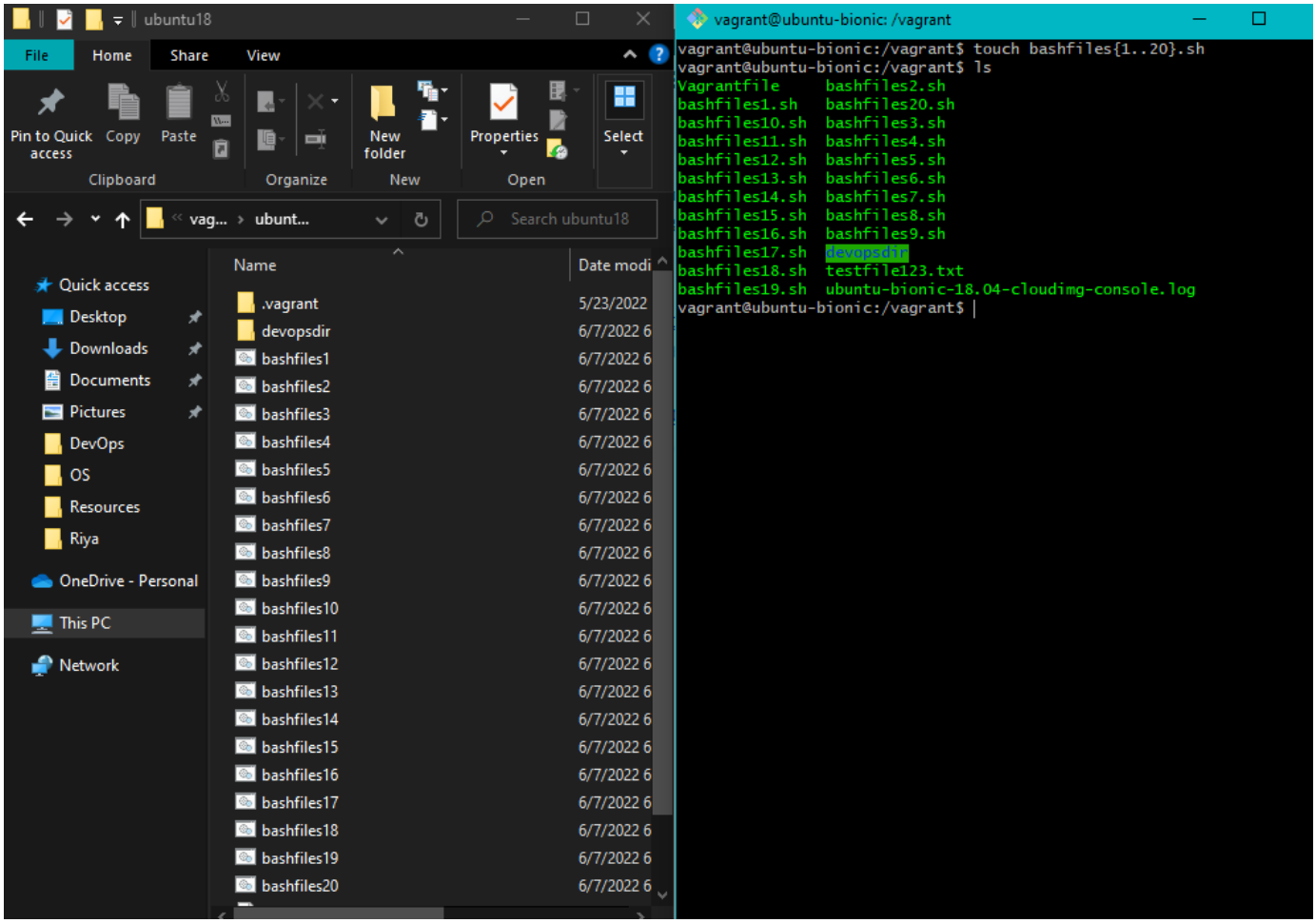
the vagrant directory in the machine is synced with the /c/vagrant-vms/ubuntu18 directory (where the vagrant file is)

```
hello world@DESKTOP-1BL5LTS MINGW64 /c/vagrant-vms/ubuntu18
$ ls
Vagrantfile  testfile123.txt
devopsdir/   ubuntu-bionic-18.04-cloudimg-console.log
```

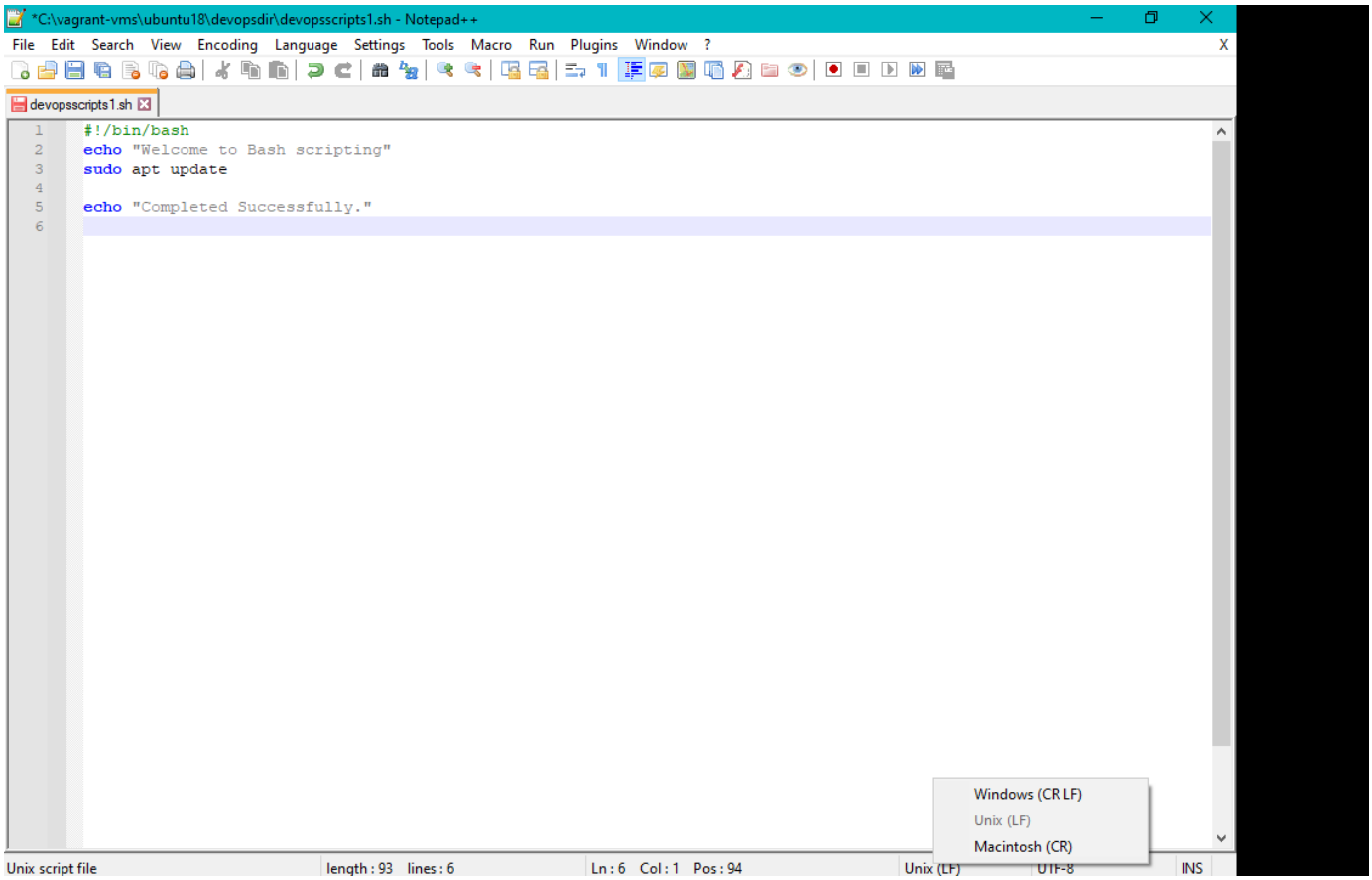
```
vagrant@ubuntu-bionic:~$ cd /vagrant/
vagrant@ubuntu-bionic:/vagrant$ ls
Vagrantfile  testfile123.txt
devopsdir    ubuntu-bionic-18.04-cloudimg-console.log
vagrant@ubuntu-bionic:/vagrant$
```

Sync directory

Even if the machine is damanged the files will not be harmed.

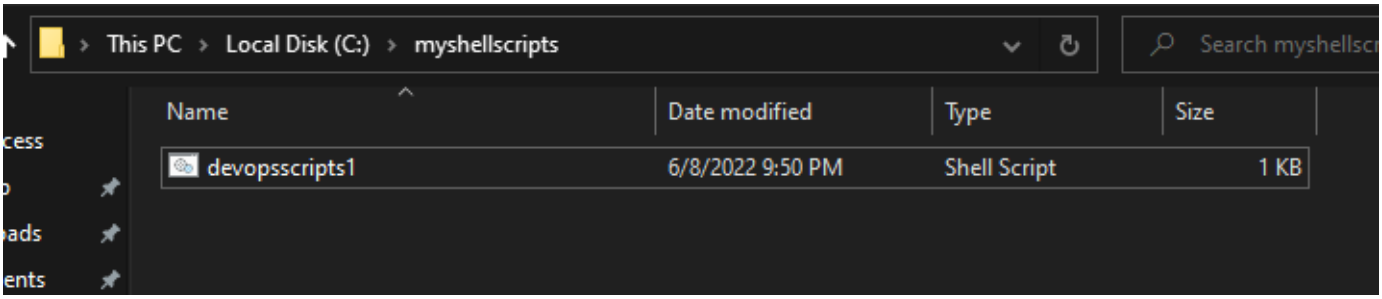


Simple bash script to print welcome and update.



now we will set our **own sync directories.**

here we are trying to sync our myshellscripts



on host directory to /opt/scripts directory on machine.

while doing this you need to put double backslash for windows

```
Vagrantfile
34  # using a specific IP.
35   config.vm.network "private_network", ip: "192.168.33.10"
36
37  # Create a public network, which generally matched to bridged network.
38  # Bridged networks make the machine appear as another physical device on
39  # your network.
40   config.vm.network "public_network"
41
42  # Share an additional folder to the guest VM. The first argument is
43  # the path on the host to the actual folder. The second argument is
44  # the path on the guest to mount the folder. And the optional third
45  # argument is a set of non-required options.
46   config.vm.synced_folder "C:\\myshellscripts", "/opt/scripts"
47
48  # Provider-specific configuration so you can fine-tune various
49  # backing providers for Vagrant. These expose provider-specific options.
50  # Example for VirtualBox:
```

line no.46 we made changes then we logout of gitbash and reload the vagrant machine by running the command `vagrant reload`

```
vagrant@ubuntu-bionic:~$ cd /opt/scripts/
vagrant@ubuntu-bionic:/opt/scripts$ ls
devopsscripts1.sh
vagrant@ubuntu-bionic:/opt/scripts$
```

as we can see the directory have been synced.

Provisioning

Provisioning in vagrant means executing commands or scritps when your VM comes up

In other technologies these things are also called as bootstrapping. Which means when OS is booting we execute commands or some scripts.

So if you are bringing a VM to configure something. you bring up the VM by making changes in the vagrant file and then you log in then you execute your commands

But

Instead of that you can just all those commands in the provisioning section of vagrantfile and then vagrant will execute that for you.

We will do this in Centos for now

First we will open the vagrantfile

we will give it an IP

```
37  # Create a public network, which generally matched to bridged network.
38  # Bridged networks make the machine appear as another physical device on
39  # your network.
40   config.vm.network "public_network"
41
```

then

```
# Enable provisioning with a shell script. Additional provisioners such as
# Ansible, Chef, Docker, Puppet and Salt are also available. Please see the
# documentation for more information about their specific syntax and use.
# config.vm.provision "shell", inline: <<-SHELL
#   apt-get update
#   apt-get install -y apache2
# SHELL
end
```

This is the section for provisioning

```
# Enable provisioning with a shell script. Additional provisioners such as
# Ansible, Chef, Docker, Puppet and Salt are also available. Please see the
# documentation for more information about their specific syntax and use.
 config.vm.provision "shell", inline: <<-SHELL
   yum install httpd wget unzip -y
   mkdir /opt/devopsdir
   free -m
   uptime
 SHELL
end
```

here we put commands which we want the machine to run while booting. Its simple

if the machine is already running you need to run the command `vagrant reload --provision` we will see this with ubuntu vm

```
MINGW64:/c/vagrant-vms/centos7
    default: Warning: Connection reset. Retrying...
    default: Warning: Remote connection disconnect. Retrying...
    default: Warning: Connection aborted. Retrying...
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
==> default: Configuring and enabling network interfaces...
==> default: Mounting shared folders...
    default: /vagrant => C:/vagrant-vms/centos7
==> default: Running provisioner: shell...
    default: Running: inline script
    default: Loaded plugins: fastestmirror
    default: Determining fastest mirrors
    default:  * base: centos.excellmedia.net
    default:  * epel: mirrors.tuna.tsinghua.edu.cn
    default:  * extras: centos.excellmedia.net
    default:  * updates: centos.excellmedia.net
```

we can see here the provision command is running

```
MINGW64:/c/vagrant-vms/centos7
B  00:01
    default: Running transaction check
    default: Running transaction test
    default: Transaction test succeeded
    default: Running transaction
    default:   Installing : apr-1.4.8-7.el7.x86_64
         1/6
    default:   Installing : apr-util-1.5.2-6.el7.x86_64
         2/6
    default:   Installing : httpd-tools-2.4.6-97.el7.centos.5.x86_64
         3/6
    default:   Installing : mailcap-2.1.41-2.el7.noarch
         4/6
    default:   Installing : httpd-2.4.6-97.el7.centos.5.x86_64
         5/6
    default:   Installing : unzip-6.0-24.el7_9.x86_64
         6/6
    default:   Verifying  : httpd-tools-2.4.6-97.el7.centos.5.x86_64
         1/6
    default:   Verifying  : mailcap-2.1.41-2.el7.noarch
         2/6
    default:   Verifying  : apr-1.4.8-7.el7.x86_64
         3/6
    default:   Verifying  : apr-util-1.5.2-6.el7.x86_64
         4/6
    default:   Verifying  : unzip-6.0-24.el7_9.x86_64
         5/6
    default:   Verifying  : httpd-2.4.6-97.el7.centos.5.x86_64
         6/6
    default:
    default: Installed:
    default:   httpd.x86_64 0:2.4.6-97.el7.centos.5        unzip.x86_64 0:6.0-24
.el7_9
    default:
    default: Dependency Installed:
    default:   apr.x86_64 0:1.4.8-7.el7                      apr-util.x86_64 0:1
.5.2-6.el7
    default:   httpd-tools.x86_64 0:2.4.6-97.el7.centos.5    mailcap.noarch 0:2.
1.41-2.el7
    default:
    default: Complete!
    default:              total        used        free      shared  buff/cache
   available
    default: Mem:           486          99          87           4         300
         370
    default: Swap:         1023           0        1023
    default:  16:28:49 up 0 min,  0 users,  load average: 0.40, 0.14, 0.05

hello world@DESKTOP-1BL5LTS MINGW64 /c/vagrant-vms/centos7
$
```

and it has done every command we had given in the provision section succesfully

OK this is when the VM doesnt exist so what if the VM exists and it is running

```
MINGW64:/c/vagrant-vms/ubuntu18

hello world@DESKTOP-1BL5LTS MINGW64 /c/vagrant-vms/ubuntu18
$ vagrant global-status
id       name    provider   state   directory
--------------------------------------------------------------------------
681b859  default virtualbox running C:/vagrant-vms/ubuntu18
10cea5f  default virtualbox running C:/vagrant-vms/centos7

The above shows information about all known Vagrant environments
on this machine. This data is cached and may not be completely
up-to-date (use "vagrant global-status --prune" to prune invalid
entries). To interact with any of the machines, you can go to that
directory and run Vagrant, or you can use the ID directly with
Vagrant commands from any directory. For example:
"vagrant destroy 1a2b3c4d"

hello world@DESKTOP-1BL5LTS MINGW64 /c/vagrant-vms/ubuntu18
$ |
```

like this one

```
63
64  # Enable provisioning with a shell script. Additional provisioners such as
65  # Ansible, Chef, Docker, Puppet and Salt are also available. Please see the
66  # documentation for more information about their specific syntax and use.
67  config.vm.provision "shell", inline: <<-SHELL
68    apt-get update
69    apt-get install -y apache2
70  SHELL
71 end
```

here in ubuntu we will be running these commands

```
hello world@DESKTOP-1BL5LTS MINGW64 /c/vagrant-vms/ubuntu18
$ vagrant reload --provision
```

ok so here why are we doing --provision but not just reload, becuase

provisioning is for bootstapping and it might cause some issues if it is done time and again in every start so it is only necessary when the machine is loaded for the first time or when there are changes in the provision section.

```
==> default: Running provisioner: shell...
    default: Running: inline script
    default: Hit:1 http://archive.ubuntu.com/ubuntu bionic InRelease
    default: Hit:2 http://security.ubuntu.com/ubuntu bionic-security InRelease
    default: Hit:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease
    default: Hit:4 http://archive.ubuntu.com/ubuntu bionic-backports InRelease
    default: Reading package lists...
    default: Reading package lists...
    default: Building dependency tree...
    default: Reading state information...
    default: apache2 is already the newest version (2.4.29-1ubuntu4.23).
    default: 0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.
```

so here provisioning has been successfully done in a running VM also.

```
vagrant@ubuntu-bionic:~$ systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since Wed 2022-06-08 16:34:42 UTC; 1min 46s ago
  Process: 979 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 1144 (apache2)
    Tasks: 55 (limit: 1829)
   CGroup: /system.slice/apache2.service
           ├─1144 /usr/sbin/apache2 -k start
           ├─1145 /usr/sbin/apache2 -k start
           └─1146 /usr/sbin/apache2 -k start

Warning: Journal has been rotated since unit was started. Log output is incomplete or unavailable.
vagrant@ubuntu-bionic:~$
```

we can see it is running

apache2 is a web service it will be serving default webpages

Website Setup

Server Management

Website
First we will be setting up website on centos7 with httpd service and put some HTML templates

practical
{See chapter 38 - all practical}

Basically what we did was install httpd in centos vm then we downloaded a html site from tooplate.com then we hosted it on our ipaddress.

Therefore basically we hosted a **webiste.**

Wordpress
Then we will be hosting wordpress on ubuntu 18 (Also called LAMP stack) we will be running Apache2, mysql & PHP wordpress template

wordpress setup on ubuntu we did this as shown on the website of ubuntu

we setup a wordpress website connect it though the database.

1. Overview
2. Install Dependencies
3. Install WordPress
4. Configure Apache for WordPress
5. Configure database
6. Configure WordPress to connect to the database
7. Configure WordPress
8. Write your first post
9. That's all!

Automation

We will automate this setup by using vagrant provisioning. we will write 2 vagrant file one for website one for webpage

we did all the above steps manually but now we are gonna do that vagrant provisioning.

when we do this it will be called as Infrastructure as a code [IAAC]

```
# Enable provisioning with a shell script. Additional provisioners such as
# Ansible, Chef, Docker, Puppet and Salt are also available. Please see the
# documentation for more information about their specific syntax and use.
 config.vm.provision "shell", inline: <<-SHELL
   yum install httpd wget unzip -y
   systemctl start httpd
   systemctl enable httpd
   cd /tmp/
   wget https://www.tooplate.com/zip-templates/2113_earth.zip
   unzip -o 2113_earth.zip
   cp -r 2113_earth/* /var/www/html/
   systemctl restart httpd
  SHELL
end
```

we did this on a new machine in the IAAC folder called website

now we are going to automate the wordpress on ubuntu 20

we do it by creating a new folder inside IAAC called wordpress
then run

`vagrant init geerlingguy/ubuntu2004`

then we edit the vagrant file .
we give it a static ip or turn on publlic network.

then we provision it as the documents suggests.

```
config.vm.provision "shell", inline: <<-SHELL
 sudo apt update
 sudo apt install apache2 \
                 ghostscript \
                 libapache2-mod-php \
                 mysql-server \
                 php \
                 php-bcmath \
                 php-curl \
                 php-imagick \
                 php-intl \
                 php-json \
                 php-mbstring \
                 php-mysql \
                 php-xml \
                 php-zip -y
 sudo mkdir -p /srv/www
 sudo chown www-data: /srv/www
 curl https://wordpress.org/latest.tar.gz | sudo -u www-data tar zx -C /srv/www
 cp /vagrant/wordpress.conf /etc/apache2/sites-available/wordpress.conf

 sudo a2ensite wordpress
 sudo a2enmod rewrite
 sudo a2dissite 000-default
 sudo service apache2 reload

 mysql -u root -e 'CREATE DATABASE wordpress;'
 mysql -u root -e 'CREATE USER wordpress@localhost IDENTIFIED BY "admin123";'
 mysql -u root -e 'GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER ON wordpress.* TO wordpress@localhost;'
 mysql -u root -e 'FLUSH PRIVILEGES;'

 sudo -u www-data cp /srv/www/wordpress/wp-config-sample.php /srv/www/wordpress/wp-config.php

 sudo -u www-data sed -i 's/database_name_here/wordpress/' /srv/www/wordpress/wp-config.php
 sudo -u www-data sed -i 's/username_here/wordpress/' /srv/www/wordpress/wp-config.php
 sudo -u www-data sed -i 's/password_here/admin123/' /srv/www/wordpress/wp-config.php
```

i have put this commands in the provision section of the file.

AFTER this we do

`vagrant up`

so this concludes our IAAC. This will help to avoid human errors, save alot of time and version control your infrastucture.

Setting up multiple VMs from one vagrantfile

Docs:: https://www.vagrantup.com/docs

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

  config.vm.define "web" do |web|
    web.vm.box = "apache"
  end

  config.vm.define "db" do |db|
    db.vm.box = "mysql"
  end
end
```

we need to write our own vagrant file the above is just an example

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


  config.vm.define "web01" do |web01|
    web01.vm.box = "ubuntu/bionic64"
    web01.vm.network "private_network", ip: "192.168.40.11"
    web01.vm.provider "virtualbox" do |vb|
      vb.memory = "1600"
      vb.cpus = 2
    end
    web01.vm.provision "shell", inline: <<-SHELL
      apt update
      apt install apache2 wget unzip -y
      systemctl start apache2
      systemctl enable apache2
      cd /tmp/
      wget https://www.tooplate.com/zip-templates/2113_earth.zip
      unzip -o 2113_earth.zip
      cp -r 2113_earth/* /var/www/html/
      systemctl restart apache2
    SHELL
  end

  config.vm.define "db01" do |db01|
    db01.vm.box = "geerlingguy/centos7"
    db01.vm.network "private_network", ip: "192.168.33.12"
    db01.vm.provider "virtualbox" do |vb|
      vb.memory = "1600"
      vb.cpus = 2
  end
  db01.vm.provision "shell", inline: <<-SHELL
  yum install mariadb-server -y
  systemctl start mariadb
  systemctl enable mariadb

  mysql -u root -e 'CREATE DATABASE wordpress;'
  mysql -u root -e 'CREATE USER wordpress@localhost IDENTIFIED BY "admin123";'
  mysql -u root -e 'GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER ON wordpress.* TO wordpress@localhost;'
  mysql -u root -e 'FLUSH PRIVILEGES;'
  SHELL
  end
end
```

```
  db01: Total                                         6.9 MB/s |  21 MB  00:03
  db01: Running transaction check
  db01: Running transaction test
  db01: Transaction test succeeded
  db01: Running transaction
  db01:   Installing : perl-Data-Dumper-2.145-3.el7.x86_64                    1/10
  db01:   Installing : 1:perl-Compress-Raw-Zlib-2.061-4.el7.x86_64            2/10
  db01:   Installing : 1:mariadb-5.5.68-1.el7.x86_64                          3/10
  db01:   Installing : perl-Net-Daemon-0.48-5.el7.noarch                      4/10
  db01:   Installing : perl-Compress-Raw-Bzip2-2.061-3.el7.x86_64             5/10
  db01:   Installing : perl-IO-Compress-2.061-2.el7.noarch                    6/10
  db01:   Installing : perl-PlRPC-0.2020-14.el7.noarch                        7/10
  db01:   Installing : perl-DBI-1.627-4.el7.x86_64                            8/10
  db01:   Installing : perl-DBD-MySQL-4.023-6.el7.x86_64                      9/10
  db01:   Installing : 1:mariadb-server-5.5.68-1.el7.x86_64                  10/10
  db01:   Verifying  : perl-Compress-Raw-Bzip2-2.061-3.el7.x86_64             1/10
  db01:   Verifying  : perl-Net-Daemon-0.48-5.el7.noarch                      2/10
  db01:   Verifying  : perl-Data-Dumper-2.145-3.el7.x86_64                    3/10
  db01:   Verifying  : 1:mariadb-server-5.5.68-1.el7.x86_64                   4/10
  db01:   Verifying  : perl-DBD-MySQL-4.023-6.el7.x86_64                      5/10
  db01:   Verifying  : 1:mariadb-5.5.68-1.el7.x86_64                          6/10
  db01:   Verifying  : 1:perl-Compress-Raw-Zlib-2.061-4.el7.x86_64            7/10
  db01:   Verifying  : perl-DBI-1.627-4.el7.x86_64                            8/10
  db01:   Verifying  : perl-IO-Compress-2.061-2.el7.noarch                    9/10
  db01:   Verifying  : perl-PlRPC-0.2020-14.el7.noarch                       10/10
  db01:
  db01: Installed:
  db01:   mariadb-server.x86_64 1:5.5.68-1.el7
  db01:
  db01: Dependency Installed:
  db01:   mariadb.x86_64 1:5.5.68-1.el7
  db01:   perl-Compress-Raw-Bzip2.x86_64 0:2.061-3.el7
  db01:   perl-Compress-Raw-Zlib.x86_64 1:2.061-4.el7
  db01:   perl-DBD-MySQL.x86_64 0:4.023-6.el7
  db01:   perl-DBI.x86_64 0:1.627-4.el7
  db01:   perl-Data-Dumper.x86_64 0:2.145-3.el7
  db01:   perl-IO-Compress.noarch 0:2.061-2.el7
  db01:   perl-Net-Daemon.noarch 0:0.48-5.el7
  db01:   perl-PlRPC.noarch 0:0.2020-14.el7
  db01:
  db01: Complete!
  db01: Created symlink from /etc/systemd/system/multi-user.target.wants/mariadb.service to /usr/lib/systemd/system/mariadb
service.

ello world@DESKTOP-1BL5LTS MINGW64 /c/vagrant-vms/Multi-VM
```

like this we have created 2 vms with one vagrantfile.

Infrastructure as code (IaC) is the process of managing and provisioning infrastructure (networks, virtual machines, load balancers, and connection topology) through CODE/Config Files.

E:g

Vagrant for local
Terraform for Cloud
Ansible for Servers

Cloudformation for AWS

etc

Provisioning is the process of configuring and deploying an information technology (IT) system resource either locally or in the cloud. In enterprise computing, the term is often associated with virtual machines (VMs) and cloud resource instances.