They can also be used to act on files.

This will run the command 'cat' on file1 and file2

```
for Variable in file1 file2
do
    cat "$Variable"
done
```

or the output from a command

This will cat the output from ls.

```
for Output in $(ls)
do
    cat "$Output"
done
```

While loop

The bash while loop is a control flow statement that allows code or commands to be executed repeatedly based on a given condition. For example, run echo command 5 times or read text file line by line or evaluate the options passed on the command line for a script.

syntax

The syntax is as follows:

```
while [ condition ]
do
   command1
   command2
   command3
done
```

command1 to command3 will be executed repeatedly till condition is true. The argument for a while loop can be any Boolean expression. Infinite loops occur when the conditional never evaluates to false. For example, following while loop will print welcome 5 times on screen:

```
#!/bin/bash
a=1
while [ $a -le 5 ]
do
  echo "Number $a"
  x=$(( $x + 1 ))
done
```

Printing list of host IP addresses from hosts file.

Create a file named hosts

```
$ vi hosts
192.168.1.10
192.168.1.11
192.168.1.12
192.168.1.13
```

```
#!/bin/bash
```

- `for i in `cat hosts`;do`
- `  echo "Printing list of hosts."`
- `  echo $i`
- `done`

Explanation

Line 1. For command substitution we are using backticks `` . It's different from single quote ‘’

`cat hosts` will return the content of hosts file line by line, which will be stored in variable “i”.

6. Real time use cases

Bash script to install Apache, MYSQL and PHP for Ubuntu OS.

Colouring your script.
Check below mentioned link for information on colouring your echo output.

http://misc.flogisoft.com/bash/tip_colors_and_formatting

The **ANSI/VT100** terminals and terminal emulators are not just able to display black and white text ; they can display **colours** and formatted texts thanks to **escape sequences**. Those sequences are composed of the **Escape character** (often represented by "^[" or "<Esc>") followed by some other characters: "<Esc>[*FormatCode*m".

In Bash, the <Esc> character can be obtained with the following syntaxes:

- \e
- \033
- \x1B

```
#!/bin/bash
#COLORS
# Reset
Color_Off='\033[0m'       # Text Reset

# Regular Colors
Red='\033[0;31m'          # Red
Green='\033[0;32m'        # Green
Yellow='\033[0;33m'       # Yellow
Purple='\033[0;35m'       # Purple
Cyan='\033[0;36m'         # Cyan

# Update packages and Upgrade system
echo -e "$Cyan \n Updating System.. $Color_Off"
sudo apt-get update -y && sudo apt-get upgrade -y

## Install AMP
echo -e "$Cyan \n Installing Apache2 $Color_Off"
```

```
sudo apt-get install apache2 apache2-doc apache2-mpm-prefork apache2-utils
libexpat1 ssl-cert -y


echo -e "$Cyan \n Installing PHP & Requirements $Color_Off"

sudo apt-get install libapache2-mod-php5 php5 php5-common php5-curl php5-dev
php5-gd php5-idn php-pear php5-imagick php5-mcrypt php5-mysql php5-ps php5-
pspell php5-recode php5-xsl -y


echo -e "$Cyan \n Installing MySQL $Color_Off"

sudo apt-get install mysql-server mysql-client libmysqlclient15.dev -y


echo -e "$Cyan \n Installing phpMyAdmin $Color_Off"

sudo apt-get install phpmyadmin -y


echo -e "$Cyan \n Verifying installs$Color_Off"

sudo apt-get install apache2 libapache2-mod-php5 php5 mysql-server php-pear
php5-mysql mysql-client mysql-server php5-mysql php5-gd -y


## TWEAKS and Settings

# Permissions

echo -e "$Cyan \n Permissions for /var/www $Color_Off"

sudo chown -R www-data:www-data /var/www

echo -e "$Green \n Permissions have been set $Color_Off"


# Enabling Mod Rewrite, required for WordPress permalinks and .htaccess files

echo -e "$Cyan \n Enabling Modules $Color_Off"

sudo a2enmod rewrite

sudo php5enmod mcrypt


# Restart Apache

echo -e "$Cyan \n Restarting Apache $Color_Off"

sudo service apache2 restart
```

Backup scripts

Being working with systems you may need to take backup of files, directories, log files etc. Below mention scenario shows you how you can automate the backup procedures.

In this example, we will create a file and mention the name of log files that needs to be backup up with tar command. Before taking backup, our script will also tell us if the log file exists or not. It will skip the backup procedure if the file does not exists. After all there is no point running backup command if the log file does not exist.

Create directory for storing log files

```
$ mkdir -p /tmp/scripts/logs
```

Put some files in the logs directory.

```
$ cd /tmp/scripts/logs

$ touch ansible.log  apache.log  mysql.log  nagios.log
```

You can choose to put some content in the log files, touch will just create empty files.

```
$ cd /tmp/scripts
```

Create a file where you place the name of the files that you want to backup.

```
$ vi backup_files.txt

apache.log

mysql.log

nagios.log

ansible.log

chef.log
```

There is one extra filename chef.log which is not present in our logs directory /tmp/scripts/logs.

We will see how we will handle it in our script

```
$ vi backup.sh

#!/bin/bash


LOG_DIR='/tmp/scripts/logs'

BACKUP_DIR='/tmp/scripts/logs_backup'


mkdir -p $BACKUP_DIR

for i in `cat backup_files.txt`; do

  if [ -f $LOG_DIR/$i ];

  then
```

```
    echo "Copying $i to logs_backup directory."
    cp $LOG_DIR/$i $BACKUP_DIR
  else
    echo "$i log file does exist, skipping."
  fi
done
echo
echo
echo "Zipping log files"
tar -czvf logs_backup.tgz logs_backup
echo
echo
echo "Backup completed successfully."
```

Mysql Database Backup Script

mysqldump command is used to take the db dump for mysql. In the script, we're taking the dbdump and sending it to a target directory in a zipped format. We are also removing 8 days old dbbackup file by using find command. This process is called a purging or purging old backup/log files.

```
#!/bin/sh
TIME_NOW="$(date +'%d_%m_%Y_%H_%M_%S')"
BACKUPFILE="db_backup_$TIME_NOW".gz
BACKUP_DIR="/opt/db_backup_dir"
PATHOFBACKUPFILE="$BACKUP_DIR/$BACKUPFILE"
LOG_FILE="$BACKUP_DIR/"backup_log_"$(date +'%Y_%m')".txt
echo "mysqldump started at $(date +'%d-%m-%Y %H:%M:%S')" >> "$LOG_FILE"
mysqldump --user=dbuser--password=dbpass --default-character-set=utf8 mydatabase
| gzip > "$PATHOFBACKUPFILE"
echo "mysqldump finished at $(date +'%d-%m-%Y %H:%M:%S')" >> "$LOG_FILE"
chown myuser "$PATHOFBACKUPFILE"
chown myuser "$LOG_FILE"
echo "file permission changed" >> "$LOG_FILE"
find "$BACKUP_DIR" -name db_backup_* -mtime +8 -exec rm {} \;
echo "old files deleted" >> "$LOG_FILE"
```

```
echo "operation finished at $(date +'%d-%m-%Y %H:%M:%S')" >> "$LOG_FILE"
echo "******************" >> "$LOG_FILE"
exit 0
```

Running command on remote servers/nodes

Sometimes we need to run a command or set of commands on multiple nodes/server.

We use ssh to login to these nodes and run that command individually and manually on all the nodes/servers.

But it's a very time consuming and mundane work if you have to do it many times.

We will write a bash script to do that.

For this exercise, we will choose to run "yum install httpd" on three nodes.

Assumptions:

1. Three centos VMs
2. VMs have internet connection to download and install software.
3. All VMs have same username to connect.

We will create a file named "hosts-dev" and add ip address of all three nodes in that.

```
$ vi hosts-dev
192.168.2.5
192.168.2.6
192.168.2.7
```

We will write a script which will read the ip address from the hosts-dev file, do ssh to all of them one by one and run yum install httpd command over ssh.

```
$ vi install.sh
#!/bin/bash
for hosts in `cat hosts-dev`
do
  ssh vagrant@$hosts sudo yum install httpd -y
done
```

Let's break it down

Line 2: for loop will run over the content on hosts-dev file one by one, which are the ip addresses of the VMs. Notice we have used backticks `` and not single quotes '' to read the hosts-dev file (`cat hosts-dev`).

Line 4: we are running `sudo yum install httpd -y` command over ssh. $hosts variable will hold the ip address and will establish ssh connection with vagrant user(ssh vagrant@$hosts).
This loop will run until we exhaust all the entries in the hosts-dev file, if you have lets say 50 nodes you can add ip address of all the 50 nodes in this file.

Every time it logs into the vm's/sever/nodes it will ask you a password, it will be painful if you have lot of servers that you manage to enter password manually. In the next section, we will deal with this issue by doing ssh key exchange.

The above script will install httpd package on all the three nodes but will also ask you password everytime it does a ssh login. To avoid this we can do key based login which is discussed in next chapter.

7. How To Set Up SSH Keys

About SSH Keys

SSH keys provide a more secure way of logging into a virtual private server with SSH than using a password alone. While a password can eventually be cracked with a brute force attack, SSH keys are nearly impossible to decipher by brute force alone. Generating a key pair provides you with two long string of characters: a public and a private key. You can place the public key on any server, and then unlock it by connecting to it with a client that already has the private key. When the two match up, the system unlocks without the need for a password. You can increase security even more by protecting the private key with a passphrase.

Step One—Create the RSA Key Pair

The first step is to create the key pair on the client machine (there is a good chance that this will just be your computer):

```
$ ssh-keygen -t rsa
```

Step Two—Store the Keys and Passphrase

Once you have entered the Gen Key command, you will get a few more questions:

Enter file in which to save the key (/home/demo/.ssh/id_rsa):

You can press enter here, saving the file to the user home (in this case, my example user is called demo).

Enter passphrase (empty for no passphrase):

It's up to you whether you want to use a passphrase. Entering a passphrase does have its benefits: the security of a key, no matter how encrypted, still depends on the fact that it is not visible to anyone else. Should a passphrase-protected private key fall into an unauthorized user's possession, they will be unable to log in to its associated accounts until they figure out the passphrase, buying the hacked user some extra time. The only downside, of course, to having a passphrase, is then having to type it in each time you use the Key Pair.

The entire key generation process looks like this:

```
$ ssh-keygen -t rsa

Generating public/private rsa key pair.

Enter file in which to save the key (/home/demo/.ssh/id_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/demo/.ssh/id_rsa.
```

```
Your public key has been saved in /home/demo/.ssh/id_rsa.pub.

The key fingerprint is:

4a:dd:0a:c6:35:4e:3f:ed:27:38:8c:74:44:4d:93:67 demo@a

The key's randomart image is:
 '
+--[ RSA 2048]----+

|          .oo.   |

|         .  o.E  |

|        + .  o   |

|     . = = .     |

|      = S = .    |

|     o + = +     |

|      . o + o .  |

|           . o   |

|                 |

+-----------------+
```

The public key is now located in /home/demo/.ssh/id_rsa.pub The private key (identification) is now located in /home/demo/.ssh/id_rsa

Step Three—Copy the Public Key

Once the key pair is generated, it's time to place the public key on the virtual server that we want to use.

You can copy the public key into the new machine's authorized keys file with the ssh-copy-id command. Make sure to replace the example username and IP address below.

```
$ ssh-copy-id user@192.168.2.5
```

Alternatively, you can paste in the keys using SSH:

```
$ cat ~/.ssh/id_rsa.pub | ssh user@192.168.2.5 "mkdir -p ~/.ssh && cat >>
~/.ssh/authorized_keys"
```

No matter which command you chose, you should see something like:

```
The authenticity of host '192.168.2.5 (192.168.2.5)' can't be established.RSA
key fingerprint is b1:2d:33:67:ce:35:4d:5f:f3:a8:cd:c0:c4:48:86:12.

Are you sure you want to continue connecting (yes/no)? Yes

Warning: Permanently added '12.34.56.78' (RSA) to the list of known hosts.

user@12.34.56.78's password:

Now try logging into the machine, with "ssh 'user@12.34.56.78'", and check in:

  ~/.ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

Now you can go ahead and log into user@192.168.2.5 and you will not be prompted for a password. However, if you set a passphrase, you will be asked to enter the passphrase at that time (and whenever else you log in in the future).

Exercise:

- Create 4 centos vm's. One among these four vm one will be the automation box from where we will run our scripts.

- Do SSH key exchange from automation box to rest of the three vm's.

- Write a script which will install Apache & mysql server, start and enable both services and check status of apache, mysql & iptables. Execute this script from automation box for all three vm and validate.

8. Few Sample scripts

A script to automate below mentioned task on Centos

1. Install mlocate search tool, update mlocate database

2. Install, start & enable httpd service.

3. Find files with permission 0777 and delete it.

4. Check hard disk free space and alerts if its running low on disk space.

```
#!/bin/bash
# Sample script to automate tasks:
# -Update local file database:
echo -e "\e[4;32mInstalling mlocate\e[0m"
sudo yum install mlocate -y
echo ""
echo -e "\e[4;32mUPDATING LOCAL FILE DATABASE\e[0m"
sudo updatedb
if [ $? == 0 ]; then
        echo "The local file database was updated correctly."
else
        echo "The local file database was not updated correctly."
fi
echo ""

# Installing and staring HTTPD service
echo -e "\e[4;32mInstalling HTTPD package\e[0m"
sudo yum install httpd -y
echo ""
echo -e "\e[4;32mStarting and enabling HTTPD package\e[0m"
sudo /etc/init.d/httpd start && chkconfig httpd on
echo ""
# For CentOS 7
#sudo systemctl start httpd && sudo systemctl enable httpd
```

```
# -Find and / or delete files with 777 permissions.

echo -e "\e[4;32mLOOKING FOR FILES WITH 777 PERMISSIONS\e[0m"

# Enable either option (comment out the other line), but not both.

# Option 1: Delete files without prompting for confirmation. Assumes GNU version
of find.

#find -type f -perm 0777 -delete

# Option 2: Ask for confirmation before deleting files. More portable across
systems.

find -type f -perm 0777 -exec rm -f {} +;

echo ""

# -Alert when file system usage surpasses a defined limit

echo -e "\e[4;32mCHECKING FILE SYSTEM USAGE\e[0m"

THRESHOLD=10

while read line; do

        # This variable stores the file system path as a string

        FILESYSTEM=$(echo $line | awk '{print $1}')

        # This variable stores the use percentage (XX%)

        PERCENTAGE=$(echo $line | awk '{print $5}')

        # Use percentage without the % sign.

        USAGE=${PERCENTAGE%?}

        if [ $USAGE -gt $THRESHOLD ]; then

                echo "The remaining available space in $FILESYSTEM is critically
low. Used: $PERCENTAGE"

        fi

done < <(df -h --total | grep -vi filesystem)
```

Automatic maintenance script for nginx service.

This Script starts nginx service if its dead.

Place it under cronjob and schedule to run every 2 minutes which acts as a monitoring and maintenance script.

Assumptions: Nginx is already installed on the system.

cat nginstart.sh

```
#!/bin/bash


if [ -f /var/run/nginx.pid ]
then
  echo "Nginx is running."
else
  echo "Starting nginx service."
  service nginx start
fi
```

```
# crontab -e
* * * * * /opt/scripts/nginstart.sh
```

Example Script for automating Jenkins setup.

```
#!/bin/bash


# Author: Pavan Kumar Ranjit
## LOGIC TO CHECK THE TYPE OF DISTRIBUTION (REDHAT OR DEBIAN)
yum --help >> /tmp/log1
if [ $? -eq 0 ]
## "$?" STORES THE EXIT CODE OF THE MOST RECENT COMMAND
then
  echo "RPM Based OS Detected"
  echo "Installing Java-JDK,Jenkins,Maven"
  sleep 3
  sudo yum install java-1.8.0-openjdk -y
  sudo yum install java-1.8.0-openjdk-devel -y
  sudo yum install wget -y
  sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-
ci.org/redhat/jenkins.repo
  sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key
  sudo yum install Jenkins -y
  sudo yum install maven -y
  sudo yum install git -y
```

```
  echo "Configuring services.... Please Wait"
  sleep 5
  sudo service iptables stop
  sudo service Jenkins start
else
 '
  echo "Debian Based OS Detected"
  sleep 3
  echo "Installing Java-JDK,Jenkins,Maven"
  sudo apt-get update
  sudo apt-get install openjdk-8-jdk -y
  sudo apt-get install openjdk-8-jre -y
  sudo apt-get install maven -y
  sudo apt-get install wget -y
  wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key
add -
  sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
  sudo apt-get update -y
  sudo apt-get install jenkins -y
  sudo apt-get install git -y
  echo "Configuring services.... Please Wait"
  sleep 5
  sudo systemctl stop ufw
  sudo systemctl start jenkins
fi
```

Summary:

- Scripting is required for doing system tasks automatically without manual intervention.
- Bash scripting is used by Linux System Admin for ages to automation Linux tasks.
- Variables, Condition's, loops etc are important aspects of scripting language which helps us automate complex tasks.
- Try all the sample and real-time use case scripts to get hold of system automation.

Conclusion:

There are so many advanced options in Bash Scripting like functions, list constructs, Regular Expressions etc, which you may be interested in and feel tempted to use them.

You can check Advanced Bash Scripting guide http://tldp.org/LDP/abs/html/ for advanced options.

Learning all those advanced options are great and make you a Scripting Guru if practised.

There are also so many limitations to Bash script, like its only for Linux systems and there is so much of syntax in it.

Python should be your next choice of scripting language which is easier to read and write and is also versatile, it can be used to automate tasks on windows platform and cloud services also.

Python has very less syntax for example check below "**if condition"** of bash vs python.

Bash way

```
#!/bin/bash
a=5

if [ $a -lt 10 ]
then
 echo "Variable is smaller than 10"
 exit 1
fi
```

Python way

```
#!/usr/bin/python
a=10
if a < 10:
  print "variable is less than 10"
```

Both have their upsides and downside, you should make your choices wisely. But making choices comes with experience and practice. There is no perfect way of doing anything.

Practice makes improvement.

III.Virtualization

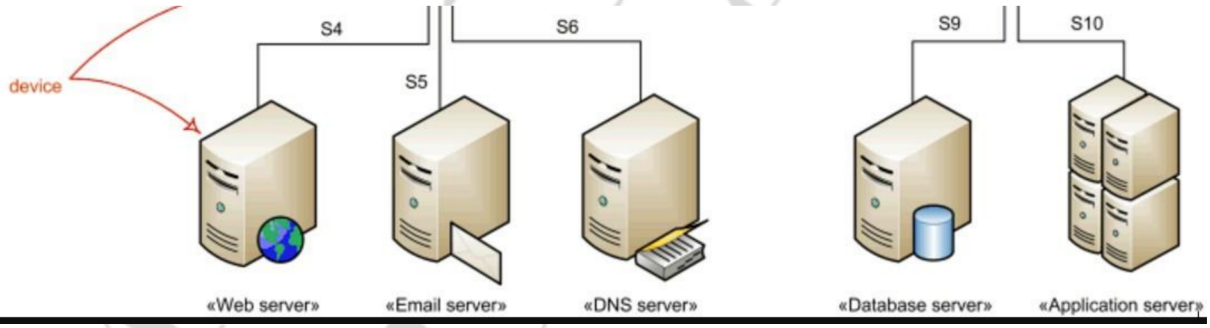
Virtualization is not a new thing in IT industry. Virtualization gained huge momentum because of Vmware as it has solved the problem of running multiple OS and apps on one physical computer. But that is just the tip of the iceberg, Vmware and other virtualization vendors has given great features like Clustering of Virtualized Physical machines, pooling compute resources, storage pooling, High availability, Live migration of virtual machine and so much more.

In this chapter we will understand virtualization, its benefits and how to use it for our DevOps day to day operations.

1. Life without virtualization

In software industry, we create and deliver softwares or as we say nowadays Applications. Application runs businesses or vice versa. If application don't run or underperform, business will suffer.

Applications runs on servers. We deploy one application per server because we want our applications to be isolated. For example, if we need web app, db app and few backend apps. We may end up having multiple physical system each running a single instance of that app.



So, every time we need a new app to run we buy servers, install OS and setup our app on that.
And most of the time nobody knew the performance requirements of the new application!
This meant IT had to make guesses when choosing the model and size of
servers to buy.
As a result, IT did the only reasonable thing - it bought big fast servers with lots of resiliency.
After all, the last thing anyone wanted - including the business - was under-powered servers.
Most part of the time these physical server's computer resource will be underutilized as low as 5-10% of their potential capacity. A tragic waste of company capital and resources.

2. Enter the Vmware

Amid all of this, VMware, Inc. gave the world the virtual machine (VM). And almost overnight the world changed into a much better place! Finally, we had a technology that would let us run multiple business applications on a single server safely and securely.

3.The Virtual Machine

A virtual computer system is known as a “virtual machine” (VM): a tightly isolated software container with an operating system and application inside. Each self-contained VM is completely independent. Putting multiple VMs on a single computer enables several operating systems and applications to run on just one physical server, or “host”.

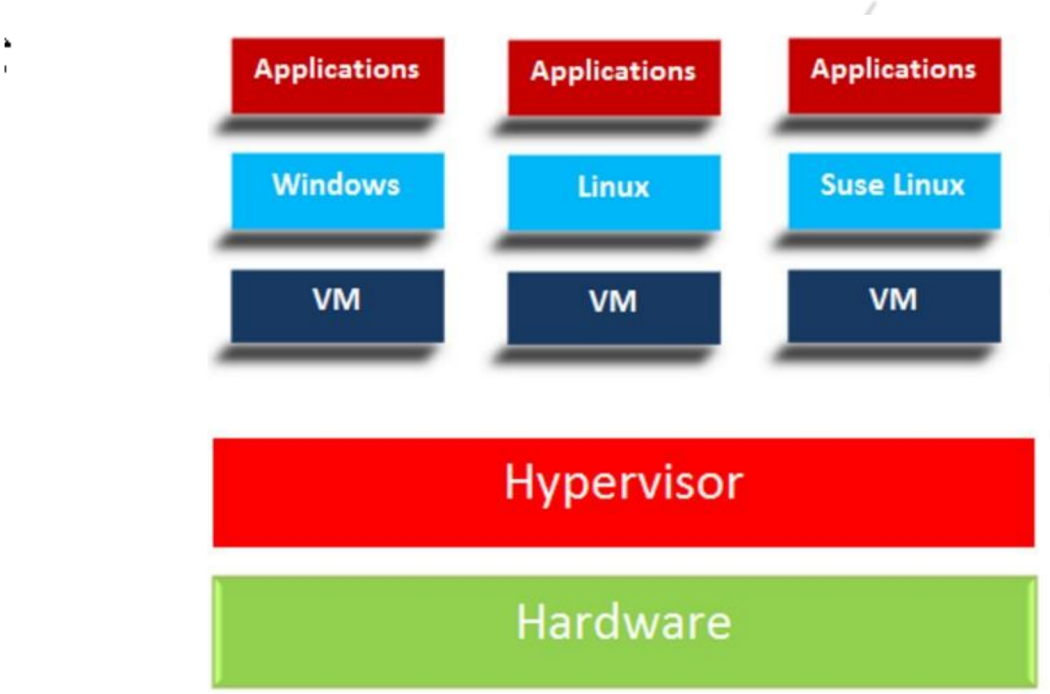
4. Key Properties of Virtual Machines

VMs have the following characteristics, which offer several benefits.

- Partitioning
- Run multiple operating systems on one physical machine
- Divide system resources between virtual machines
- Isolation
- Provide fault and security isolation at the hardware level
- Preserve performance with advanced resource controls
- Encapsulation
- Save the entire state of a virtual machine to files
- Move and copy virtual machines as easily as moving and copying files
- Hardware Independence
- Provision or migrate any virtual machine to any physical server

5. Hypervisors

As generally we install OS on the physical server and then install our apps on that. By setting up Hypervisor on the physical server we can create multiple VMs each with their own OS.



There are two types of hypervisors:

Type 1

Type 1 hypervisors run directly on the system hardware. They are often referred to as a "native" or "bare metal" hypervisors.

They get installed on a physical computer like an OS. So instead of installing OS we install Hypervisor to achieve virtualization.

Example:

- Microsoft Hyper-V
- VMware ESX/ESXi.
- Xen Hypervisors

These hypervisors are used to run production grade virtual machines that gives great performance. Type 1 hypervisors can be grouped together(clustered) and manged centrally by a software like Vmware Vcenter for Esxi.

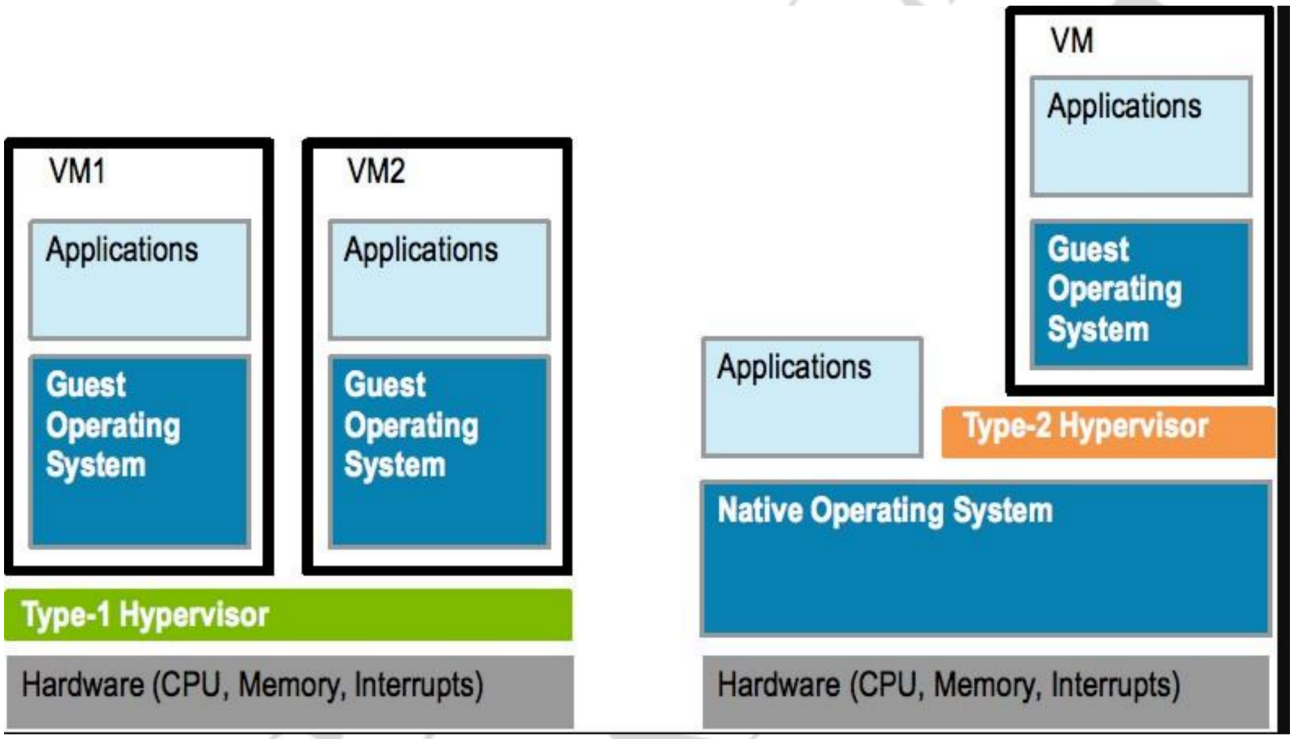
Type 2.

Type 2 hypervisors run on a host operating system.

In this kind, we install hypervisor on the OS like Linux or windows.

Example

- Vmware server/workstation/player
- Oracle virtualbox



6. Your choice of hypervisor.

Type 1 hypervisors are great and gives you amazing features like Clustering of Virtualized Physical machines, pooling compute resources, storage pooling, High availability, Live migration of virtual machine and so much more.

It's used generally to virtualize entire datacentres or pool of servers. This is managed by very expensive softwares like Vmware Vcenter. As the focus of these tutorials is on DevOps tools we will not go into type 1 hypervisors. Our focus is going to be on type 2 hypervisors like oracle virtualbox which we can install on our Laptops/desktops and get going with VMs.

We will need to run multiple VMs on our system to test/run our tools and scripts. We also need to setup web application softwares on multiple OS and we will do it by using Oracle virtualnox Hypervisor.