Lab 1: Install type 2 Hypervisor on windows host OS and configuring Linux as guest OS on it (Type 2 virtualization).

⇨ Introduction

A hypervisor, also known as a virtual machine monitor or VMM, is software that creates and runs virtual machines (VMs). A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing. The physical hardware, when used as a hypervisor, is called the host, while the many VMs that use its resources are guests. The hypervisor treats resources—like CPU, memory, and storage as a pool that can be easily reallocated between existing guests or to new virtual machines. All hypervisors need some operating system-level components—such as a memory manager, process scheduler, input/output (I/O) stack, device drivers, security manager, a network stack, and more to run VMs. Hypervisors make it possible to use more of a system's available resources and provide greater IT mobility since the guest VMs are independent of the host hardware. This means they can be easily moved between different servers.

There are 2 different types of hypervisors that can be used for virtualization: type 1 and type 2 hypervisors:

1. **Type 1 Hypervisor**

   A type 1 hypervisor, also referred to as a native or bare metal hypervisor, runs directly on the host's hardware to manage guest operating systems. It takes the place of a host operating system and VM resources are scheduled directly to the hardware by the hypervisor.

2. **Type 2 Hypervisor**

   A Type 2 hypervisor, also known as a hosted hypervisor, is a software layer that allows virtualization to take place on a host operating system (OS). It enables the execution of several guest operating systems on a single physical machine.

   A Type 2 hypervisor is put on top of a normal operating system, as opposed to a Type 1 hypervisor (bare-metal hypervisor), which operates directly on the host hardware. The host operating system could be a general-purpose operating system such as Windows or Linux. The Type 2 hypervisor then uses the host OS's resources to generate and operate virtual machines (VMs).

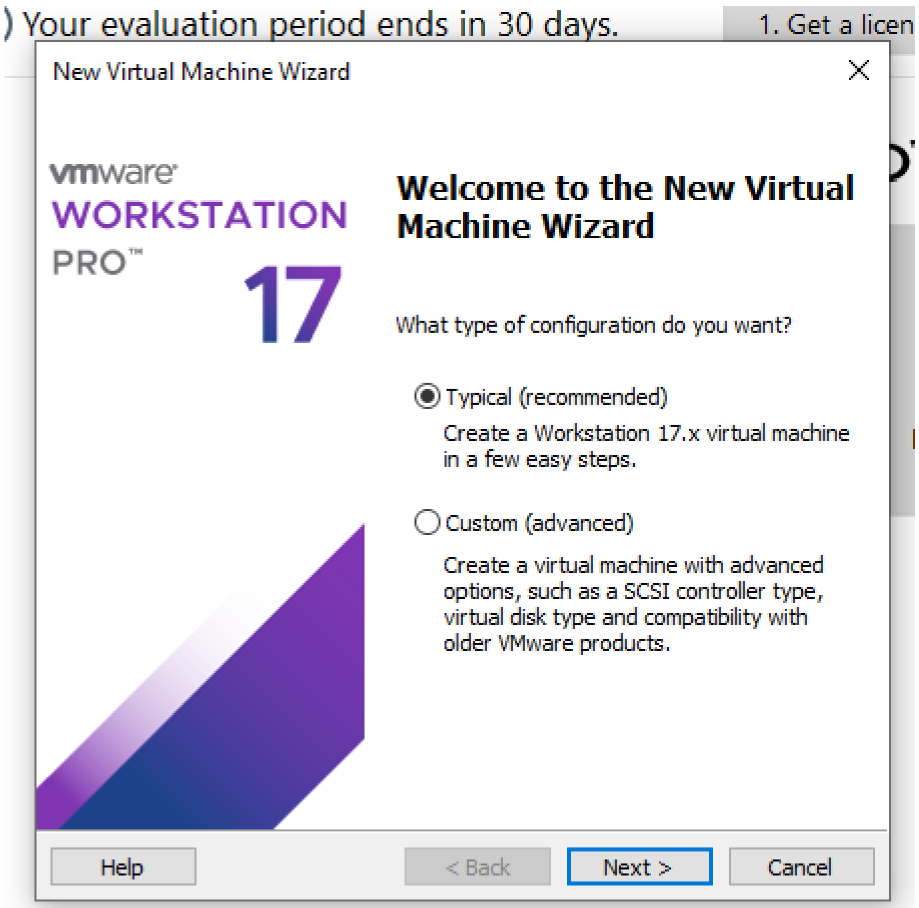
   A Type 2 hypervisor's virtualization layer intercepts and translates the guest OS instructions so that they can interface with the underlying hardware and resources. It creates a separate environment for each guest OS, allowing them to coexist on the same physical computer. The hypervisor oversees the allocation of CPU, memory, storage, and other resources across the VMs, and each guest OS normally runs as a process within the host OS.

Type 2 hypervisors are commonly used in desktop and workstation environments because they allow users to run different operating systems on a single machine at the same time. Oracle VirtualBox, VMware Workstation, and Parallels Desktop are examples of Type 2 hypervisors.

⇨ Procedure

1. Download the trial version of VMware Workstation Pro based on your host operating system.
2. Install it VM workstation pro, then select install windows hypervisor platform (WHP), then check enhance keyboard driver after installation, and restart.
3. Download at least one ISO file of an operating system other than your host OS.
4. Create VM on Hypervisor and load OS.
5. Run the VM.

⇨ Screenshots


) Your evaluation period ends in 30 days.

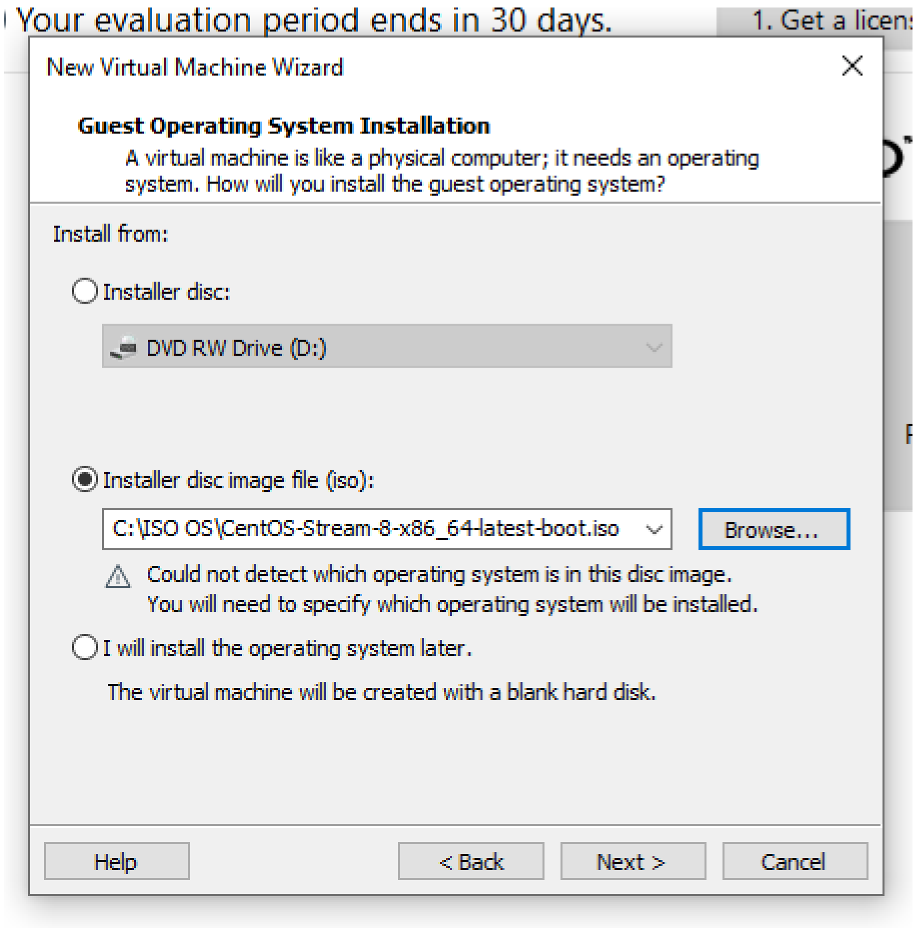
New Virtual Machine Wizard

Welcome to the New Virtual Machine Wizard

What type of configuration do you want?

[✓ Typical (recommended)]
Create a Workstation 17.x virtual machine in a few easy steps.

Custom (advanced)
Create a virtual machine with advanced options, such as a SCSI controller type, virtual disk type and compatibility with older VMware products.



New Virtual Machine Wizard

Guest Operating System Installation

A virtual machine is like a physical computer; it needs an operating system. How will you install the guest operating system?

Install from:

( ) Installer disc:

DVD RW Drive (D:)

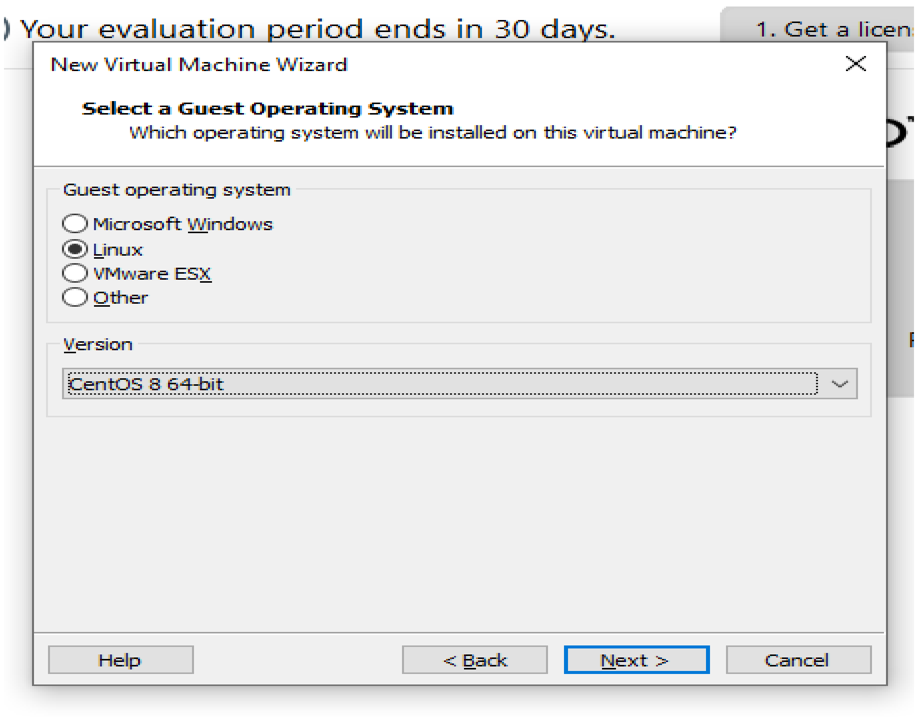
(●) Installer disc image file (iso):

C:\ISO OS\CentOS-Stream-8-x86_64-latest-boot.iso

⚠ Could not detect which operating system is in this disc image.
You will need to specify which operating system will be installed.

( ) I will install the operating system later.

The virtual machine will be created with a blank hard disk.



New Virtual Machine Wizard

Select a Guest Operating System

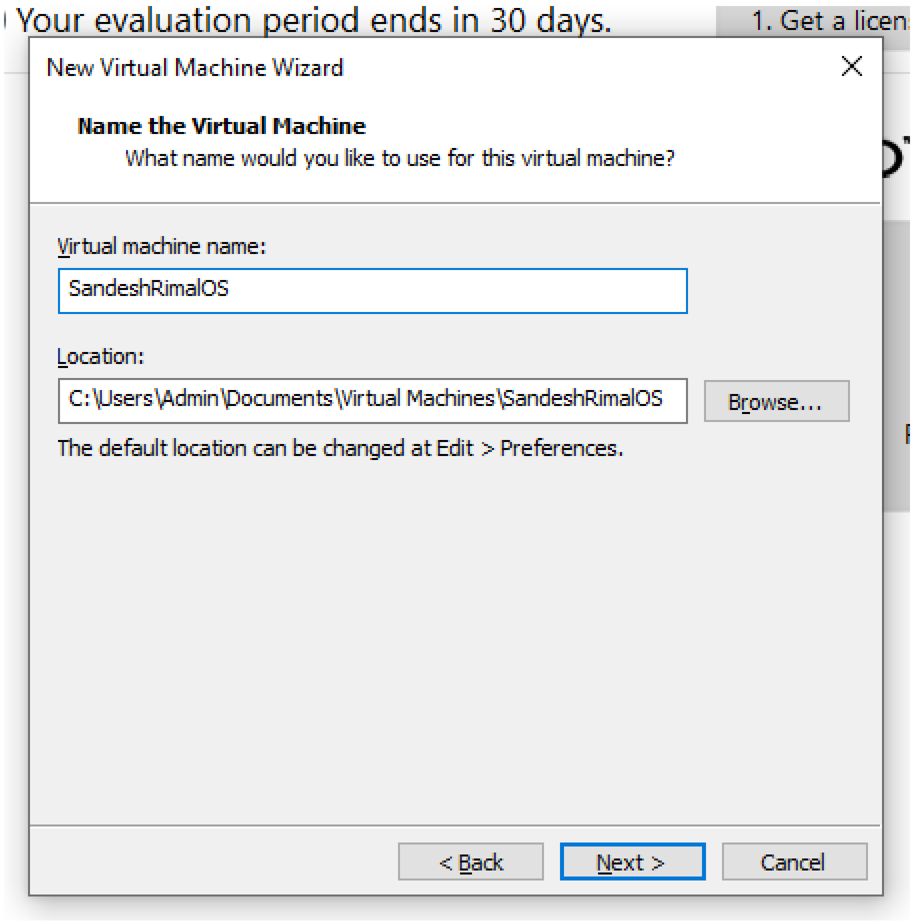
Which operating system will be installed on this virtual machine?

Guest operating system

( ) Microsoft Windows
(●) Linux
( ) VMware ESX
( ) Other

Version

CentOS 8 64-bit


Your evaluation period ends in 30 days.

1. Get a licen

New Virtual Machine Wizard

Name the Virtual Machine

What name would you like to use for this virtual machine?

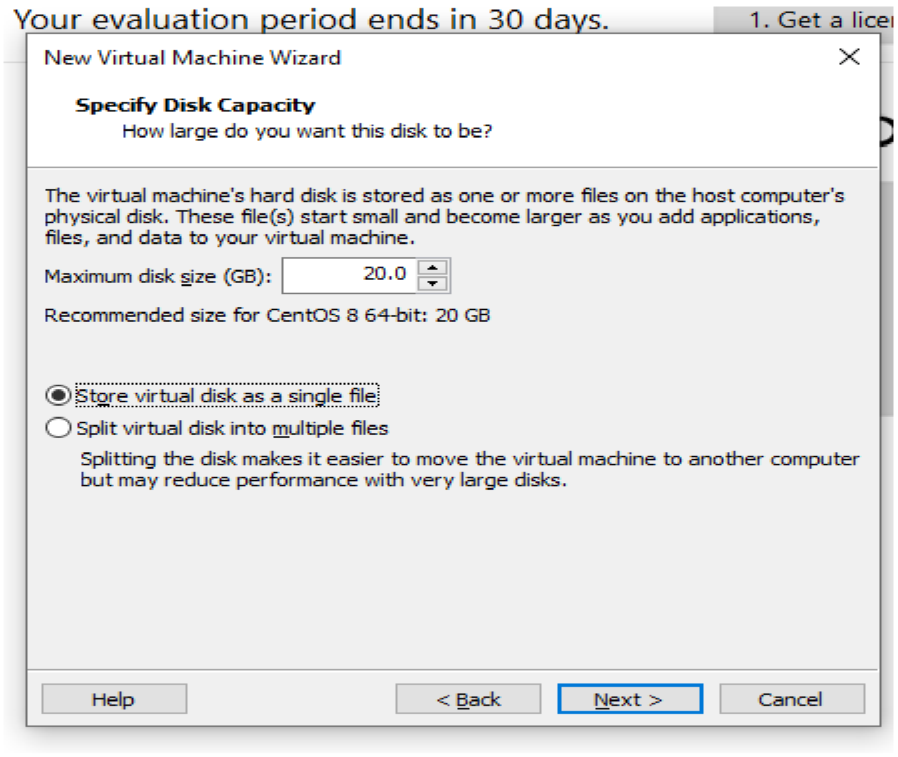
Virtual machine name:

SandeshRimalOS

Location:

C:\Users\Admin\Documents\Virtual Machines\SandeshRimalOS

The default location can be changed at Edit > Preferences.


Your evaluation period ends in 30 days.

1. Get a licer

New Virtual Machine Wizard

Specify Disk Capacity

How large do you want this disk to be?

The virtual machine's hard disk is stored as one or more files on the host computer's physical disk. These file(s) start small and become larger as you add applications, files, and data to your virtual machine.

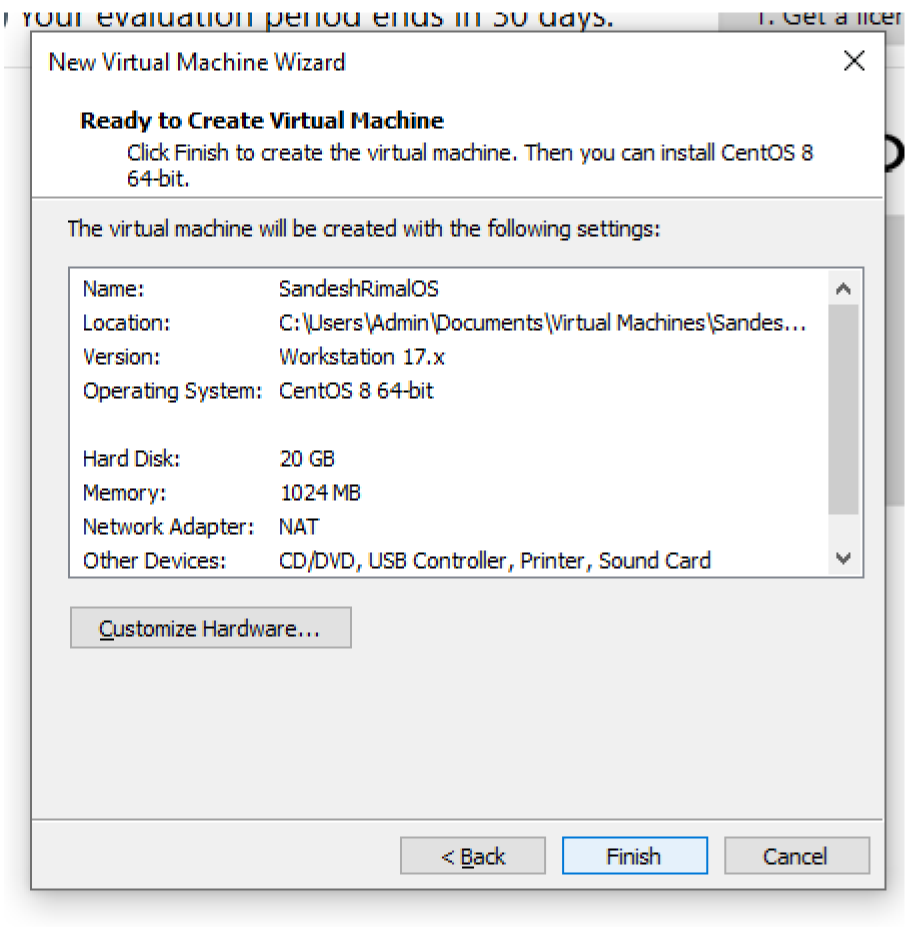
Maximum disk size (GB): 20.0

Recommended size for CentOS 8 64-bit: 20 GB

[✓ Store virtual disk as a single file]

Split virtual disk into multiple files

Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.


Your evaluation period ends in 30 days.

New Virtual Machine Wizard

Ready to Create Virtual Machine

Click Finish to create the virtual machine. Then you can install CentOS 8 64-bit.

The virtual machine will be created with the following settings:

Name:	SandeshRimalOS
Location:	C:\Users\Admin\Documents\Virtual Machines\Sandes...
Version:	Workstation 17.x
Operating System:	CentOS 8 64-bit
Hard Disk:	20 GB
Memory:	1024 MB
Network Adapter:	NAT
Other Devices:	CD/DVD, USB Controller, Printer, Sound Card

⇨ **Discussion**

We successfully installed CentOS and Puppy Linux during our virtualization exploration utilizing a type 2 hypervisor. The procedure was simple to follow and provided a secure environment for testing many operating systems on a single host machine. This experience demonstrated the efficiency and variety of type 2 hypervisors, making them excellent for testing with different OS configurations.

⇨ **Conclusion**

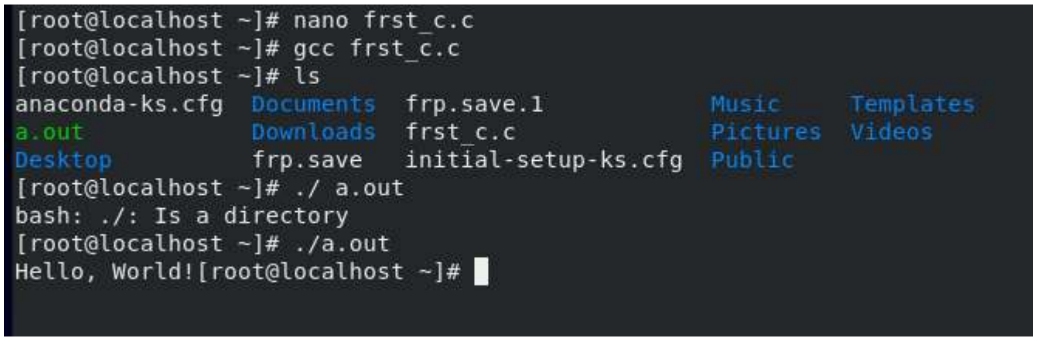
Finally, exploring into virtualization via a type 2 hypervisor was a rewarding experience. Installing CentOS and Puppy Linux demonstrated the ease and convenience of handling several operating systems on a single host machine. The deployment of a type 2 hypervisor proven its capacity to provide a secure and efficient testing and experimentation platform. Overall, this technology provides great opportunity for individuals and companies looking to experiment with different operating systems without the need for specialized hardware.

Lab 2: Run a Simple C Program in Virtual Machine using Type2 Virtualization.

⇨ After successfully installing the CentOS, we ran a simple C program saying “Hello World” in the terminal.

We followed some specific steps while running the C program, they are:

1. Check if the GCC exists on the machine by running gcc –v
2. If not, then it will automatically show the options to install the gcc, install it from there.
3. Then create one file, using the command nano file_name.c
4. After creating the file, write the code on the file.
5. Save the file.
6. Compile the file using gcc file_name.c
7. Then run the file, using ./a.out



```
[root@localhost ~]# nano frst_c.c
[root@localhost ~]# gcc frst_c.c
[root@localhost ~]# ls
anaconda-ks.cfg  Documents  frp.save.1           Music     Templates
a.out            Downloads  frst_c.c             Pictures  Videos
Desktop          frp.save   initial-setup-ks.cfg Public
[root@localhost ~]# ./ a.out
bash: ./: Is a directory
[root@localhost ~]# ./a.out
Hello, World![root@localhost ~]# 
```

⇨ Discussion

When I first tried to run gcc –v, I ran into an error saying the the machine does’nt recognize the gcc. Then it asked me to install the gcc. I followed the command shown over there and installed the gcc. After installing the gcc, I now have the accessibility to run and execute the C program in the machine. Then I created the new file, write the code saying “Hello World”. Saved, compiled and executed the file in order to get the desired result.

⇨ Conclusion

In conclusion, we successfully installed the GCC compiler on CentOS and ran a C program on it, demonstrating the system's effectiveness as a development environment. For programmers and developers, the GCC's accessibility and flawless functionality on CentOS make it a dependable option, guaranteeing a painless coding experience and effective software development process. With these features, CentOS establishes itself as a reliable environment for creating and executing C programs.

Lab 3: Python Program to Demonstrate the Working of Map Reduce.

⇨ **Introduction**

A MapReduce is a data processing tool which is used to process the data parallelly in a distributed form. It was developed in 2004, on the basis of paper titled as "MapReduce: Simplified Data Processing on Large Clusters," published by Google. The MapReduce is a paradigm which has two phases, the mapper phase, and the reducer phase. In the Mapper, the input is given in the form of a key-value pair. The output of the Mapper is fed to the reducer as input. The reducer runs only after the Mapper is over. The reducer too takes input in key-value format, and the output of reducer is the final output.

Steps in MapReduce:

- The map takes data in the form of pairs and returns a list of <key, value> pairs. The keys will not be unique in this case.
- Using the output of Map, sort and shuffle are applied by the Hadoop architecture. This sort and shuffle acts on these list of <key, value> pairs and sends out unique keys and a list of values associated with this unique key <key, list(values)>.
- An output of sort and shuffle sent to the reducer phase. The reducer performs a defined function on a list of values for unique keys, and Final output <key, value> will be stored/displayed.

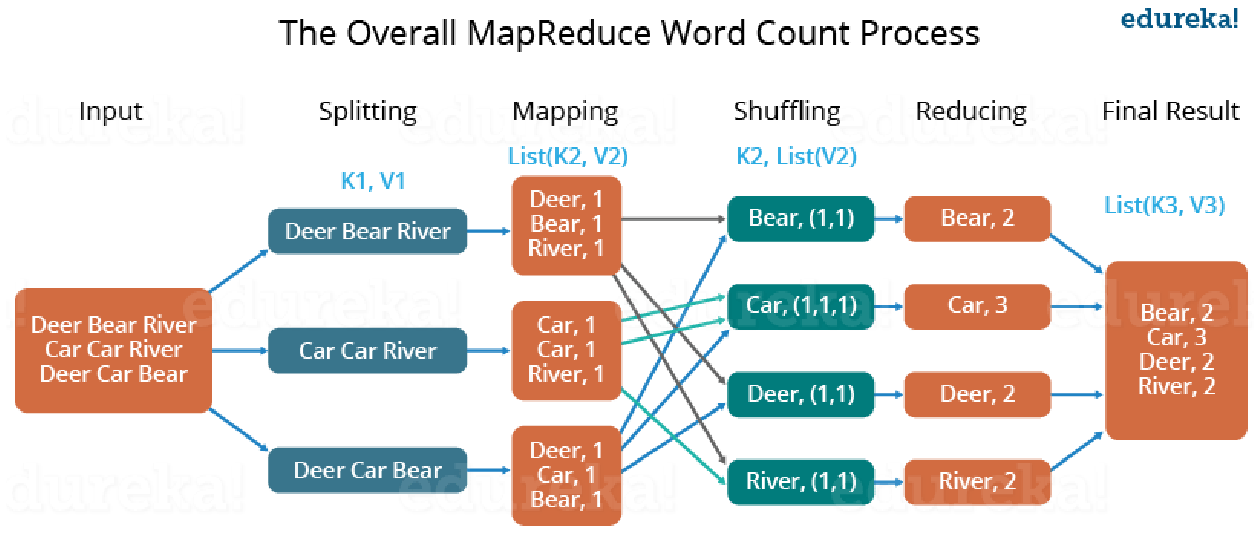
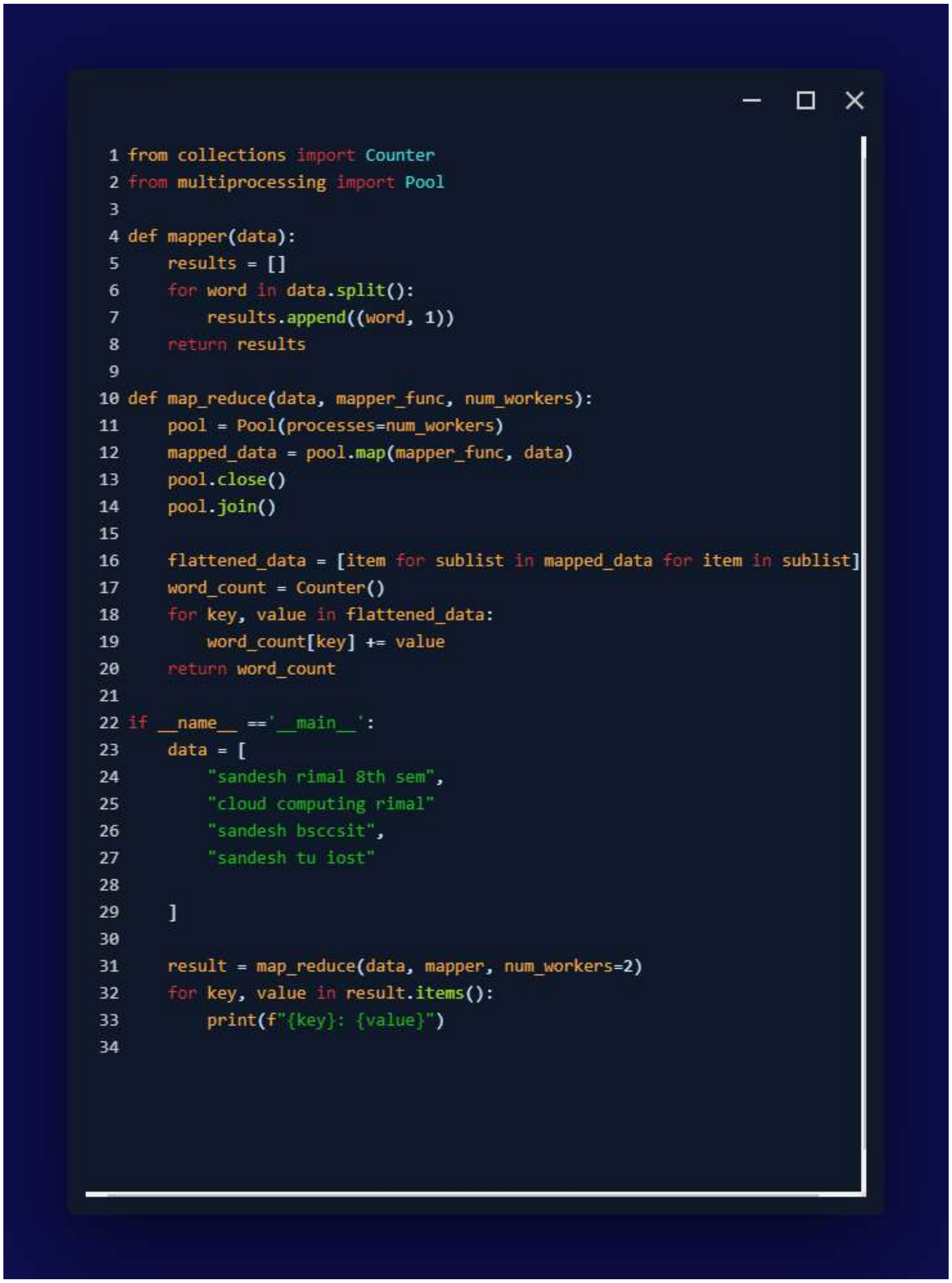


Figure 1 Map Reduce

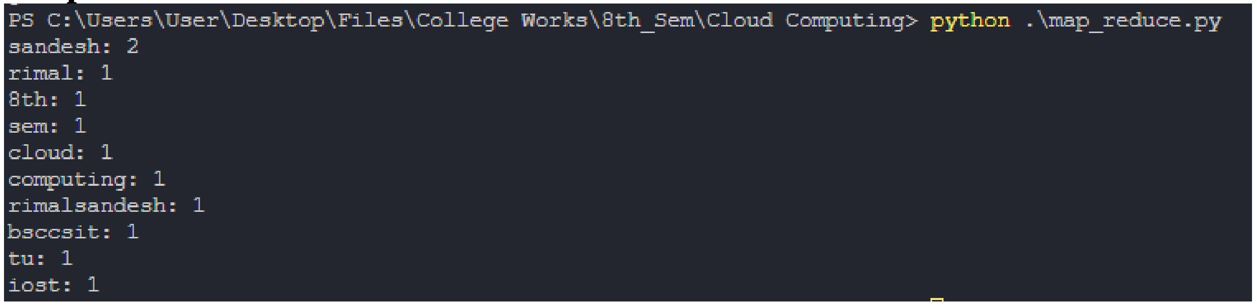
Source: https://www.analyticsvidhya.com/blog/2022/07/learn-everything-about-mapreduce-architecture-and-its-components/

⇨ **Code**



```python
from collections import Counter
from multiprocessing import Pool

def mapper(data):
    results = []
    for word in data.split():
        results.append((word, 1))
    return results

def map_reduce(data, mapper_func, num_workers):
    pool = Pool(processes=num_workers)
    mapped_data = pool.map(mapper_func, data)
    pool.close()
    pool.join()

    flattened_data = [item for sublist in mapped_data for item in sublist]
    word_count = Counter()
    for key, value in flattened_data:
        word_count[key] += value
    return word_count

if __name__ =='__main__':
    data = [
        "sandesh rimal 8th sem",
        "cloud computing rimal"
        "sandesh bsccsit",
        "sandesh tu iost"

    ]

    result = map_reduce(data, mapper, num_workers=2)
    for key, value in result.items():
        print(f"{key}: {value}")

```

⇨ Output



```
PS C:\Users\User\Desktop\Files\College Works\8th_Sem\Cloud Computing> python .\map_reduce.py
sandesh: 2
rimal: 1
8th: 1
sem: 1
cloud: 1
computing: 1
rimalsandesh: 1
bsccsit: 1
tu: 1
iost: 1
```

⇨ Discussion

While implementing the program, I ran into different errors. Sometimes module not found, sometimes indentation error, but eventually finished the program to implement the MapReduce concept in python. MapReduce is the concept of simplifying the large data into smaller parts. The MapReduce works in key value pair, where the key will be the words that is to be reduced and the value will be the number of times the word is repeated in some instance of program. Here in this code, we can see the example of MapReduce. We've given inputs of different words; numbers of words are repeated there. And what the program does is, it analyzes the input and matches the similar words and count the occurrence of the words and add the count to them making the data a lot smaller in size.

⇨ Conclusion

In conclusion, we implemented the MapReduce program using the python programming language. We obtained a stronger knowledge of the MapReduce concept and how it is used to process massive datasets as a result of this activity. Our knowledge of Python programming and the MapReduce architecture has surely increased as a result of this practical experience, which has also given us important insights into the analysis and manipulation of real-world data.

Lab 4: Multi processing program in python

⇨ Introduction

Multiprocessing refers to the utilization of two or more central processing units (CPUs) in a single computer system, allowing for simultaneous processing of programs. It can also refer to the ability of a system to support more than one processor or allocate tasks between them. The key objective of using a multiprocessor is to boost the system's execution speed, with other objectives being fault tolerance and application matching. Multiprocessing is a significant benefit to programmers as it allows them to create more efficient software that can perform tasks and functions, and save time as the software will be able to run multiple tasks without slowing down at all. Multiprocessing is a package that supports spawning processes using an API similar to the threading module.

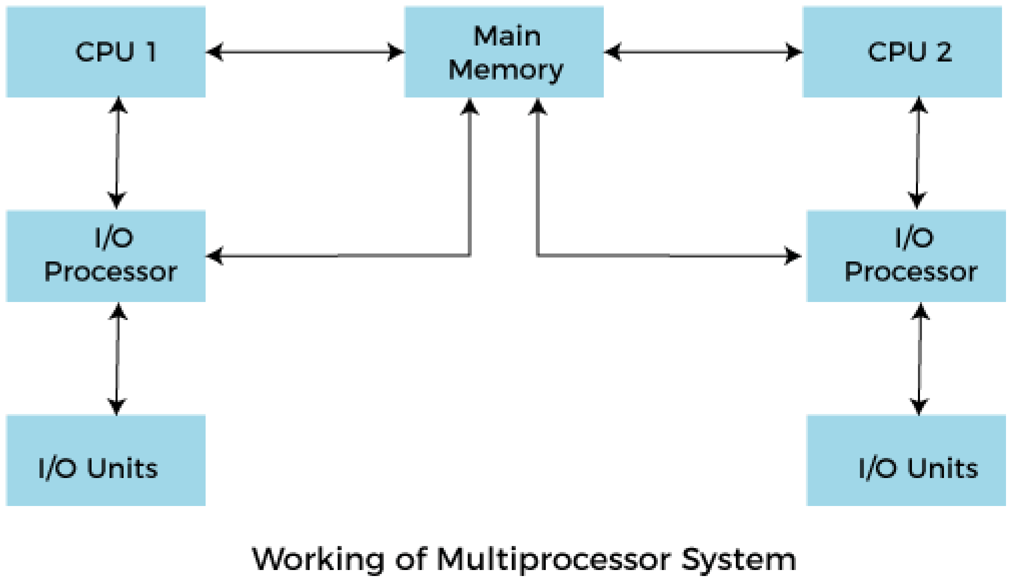
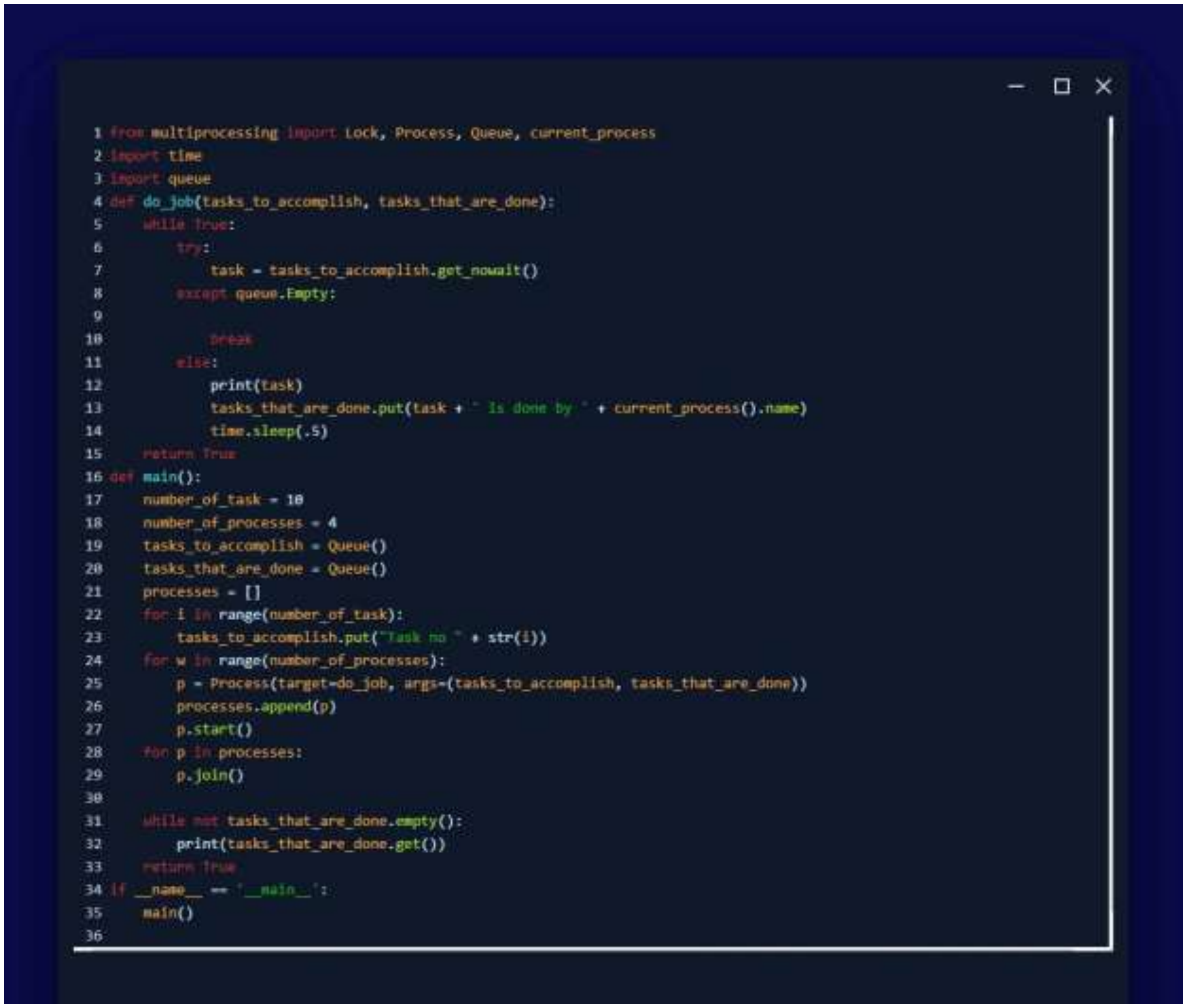


Figure 2Multi-Processing

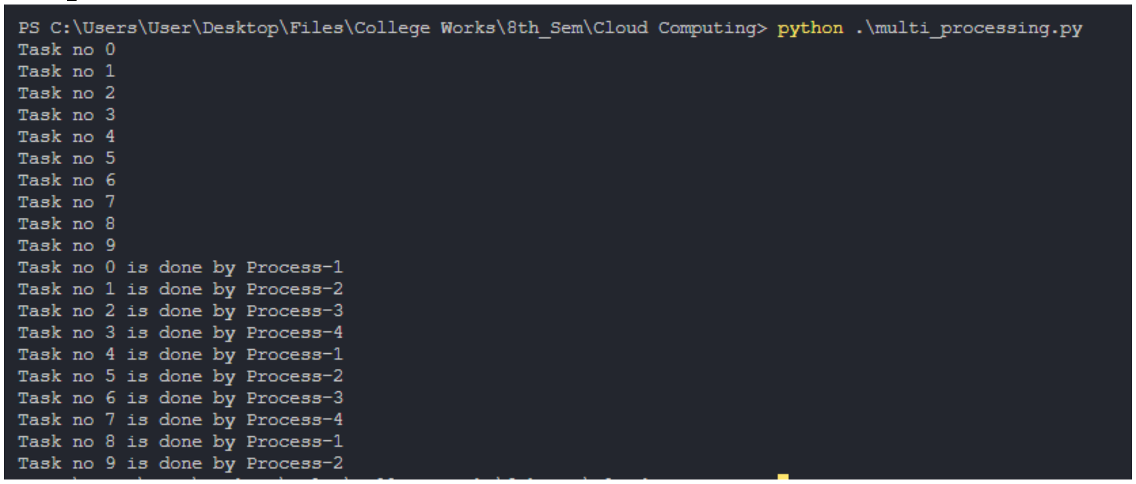
Source: https://www.javatpoint.com/multiprocessing-operating-system

⇨ Code



```python
from multiprocessing import Lock, Process, Queue, current_process
import time
import queue
def do_job(tasks_to_accomplish, tasks_that_are_done):
    while True:
        try:
            task = tasks_to_accomplish.get_nowait()
        except queue.Empty:

            break
        else:
            print(task)
            tasks_that_are_done.put(task + " is done by " + current_process().name)
            time.sleep(.5)
    return True
def main():
    number_of_task = 10
    number_of_processes = 4
    tasks_to_accomplish = Queue()
    tasks_that_are_done = Queue()
    processes = []
    for i in range(number_of_task):
        tasks_to_accomplish.put("Task no " + str(i))
    for w in range(number_of_processes):
        p = Process(target=do_job, args=(tasks_to_accomplish, tasks_that_are_done))
        processes.append(p)
        p.start()
    for p in processes:
        p.join()

    while not tasks_that_are_done.empty():
        print(tasks_that_are_done.get())
    return True
if __name__ == '__main__':
    main()

```

⇨ Output



```
PS C:\Users\User\Desktop\Files\College Works\8th_Sem\Cloud Computing> python .\multi_processing.py
Task no 0
Task no 1
Task no 2
Task no 3
Task no 4
Task no 5
Task no 6
Task no 7
Task no 8
Task no 9
Task no 0 is done by Process-1
Task no 1 is done by Process-2
Task no 2 is done by Process-3
Task no 3 is done by Process-4
Task no 4 is done by Process-1
Task no 5 is done by Process-2
Task no 6 is done by Process-3
Task no 7 is done by Process-4
Task no 8 is done by Process-1
Task no 9 is done by Process-2
```

⇨ Discussion

We have showcases a multi-processing implementation using the multiprocessing module for concurrent task execution. By utilizing multiple processes, the program efficiently processes the specified number of tasks concurrently. This example highlights the advantages of multiprocessing in improving performance and task handling, making it a valuable technique for optimizing resource-intensive operations and enhancing overall program efficiency.

⇨ Conclusion

Finally, the Python code that uses the "multiprocessing" module to demonstrate the advantages of multiprocessing for concurrent task execution. The application considerably increases efficiency and performance, especially when handling resource-intensive activities, by splitting the effort among numerous processes. This example provides a strong demonstration of how multiprocessing can improve program execution, making it a useful method for streamlining different computational processes and increasing system utilization as a whole. The knowledge gained from this exercise prepares the way for utilizing multiprocessing to its fullest in practical applications, where parallel processing is essential for satisfying contemporary computing demands.

Lab 5: Configuring ESXI for Bare Metal Virtualization and Install Linux on Top of ESXI.

⇨ Introduction

VMware ESXi is the bare-metal hypervisor in the VMware vSphere virtualization platform. As a bare-metal hypervisor for creating and running virtual machines (VMs), VMware ESXi runs on top and accesses the hardware directly without the need to install an operating system. This direct access to hardware allows it to perform better, run faster and be more scalable than other types of hypervisors. This makes VMware ESXi ideal for use in a large-scale virtual desktop infrastructure (VDI), in conjunction with the other components in the VMware vSphere platform. Since VMware ESXi doesn't run on an operating system, it has direct access to underlying resources like the CPU, RAM, storage, and networking. For this reason, VMware ESXi performs much better than any Type 2 hypervisor. Consequently, the virtual machines and the applications on those VMs run much faster than they could on a Type 2 hypervisor.

VMWare ESXi is one of the components comprising VMware VSphere, a suite of software solutions for virtualization. Other components that usually work alongside ESXi include vCenter Server, vCenter Server database, VMware Directory Service, and others.

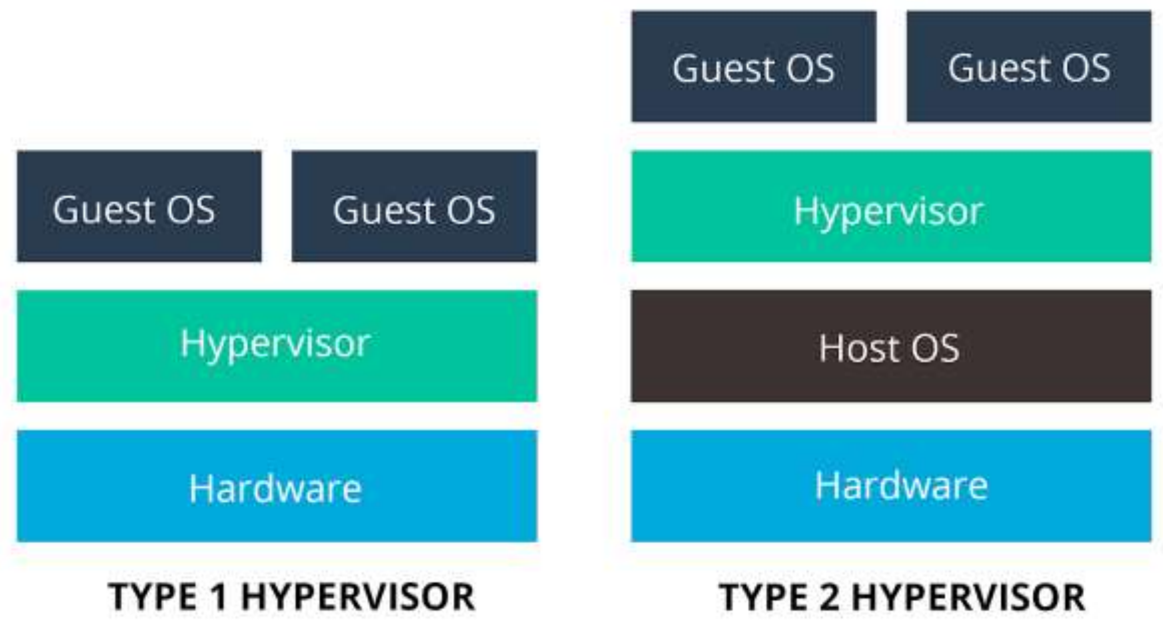


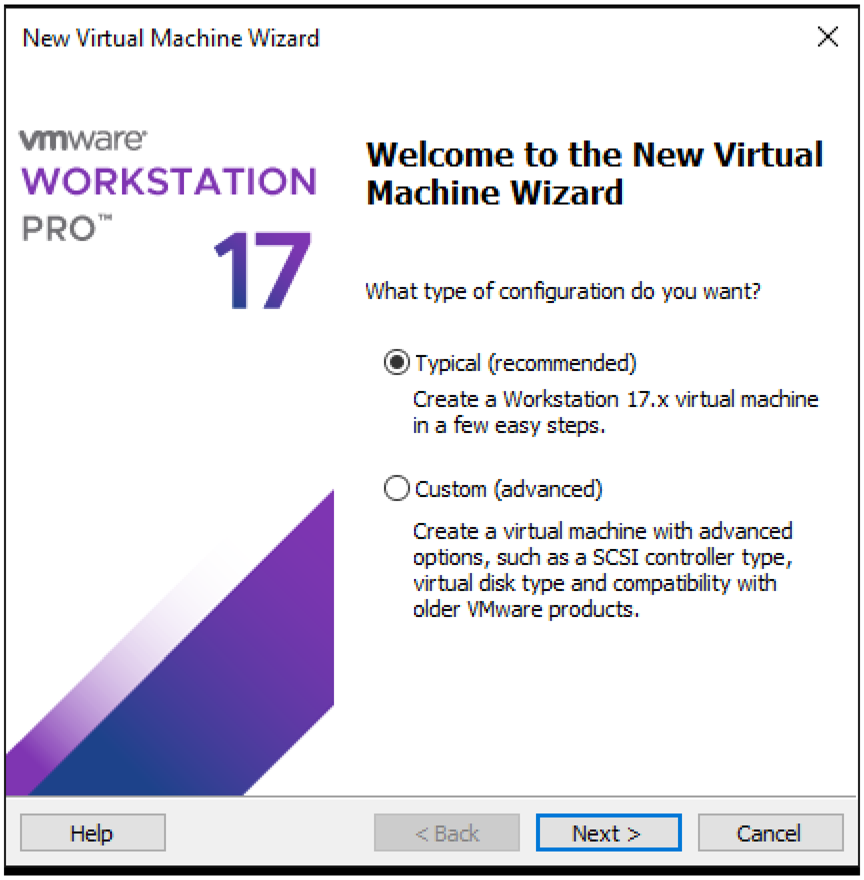
Figure 3 Type 1 vs Type 2 Hypervisor

Source https://medium.com/teamresellerclub/type-1-and-type-2-hypervisors-what-makes-them-different-6a1755d6ae2c

⇨ Procedure

- Download ESXi installer from the official website of vmware.
- Create a new virtual machine in vmware pro.
- Select the downloaded ESXi iso file.
- Allocate the required resource.
- After resource allocation and completing setting install the ESXi.
- Acess ESXi host through browser.
- Adding a new hard disk drive in VMware.
- Installing new Virtual Machine on ESXI host machine.

⇨ Screenshots

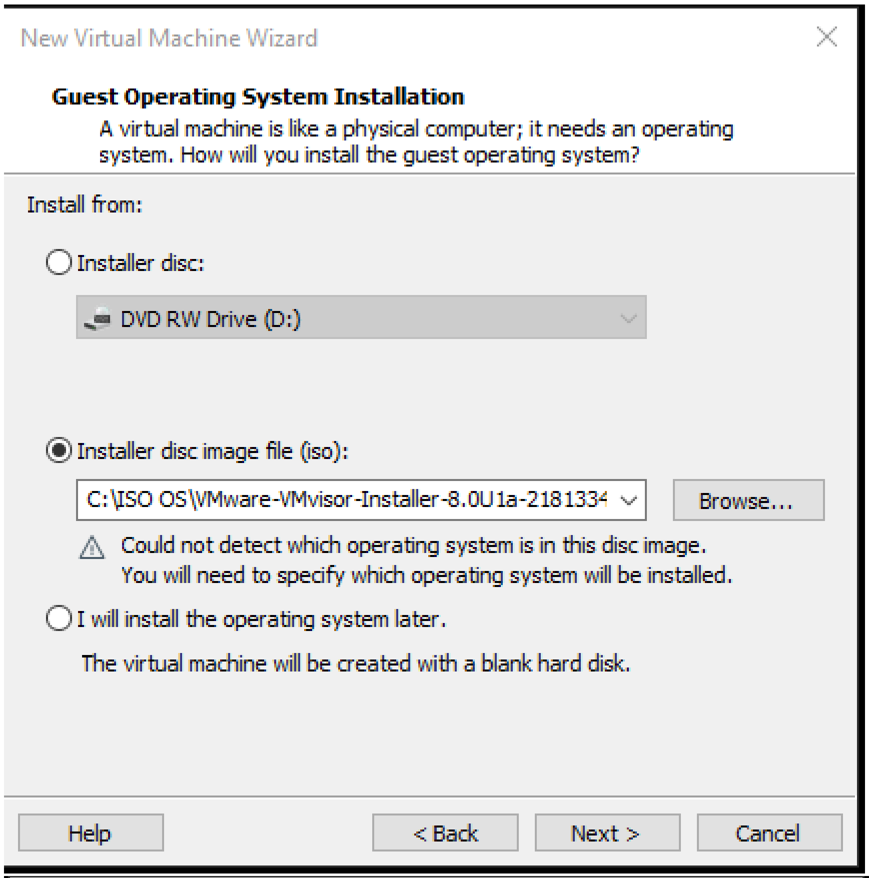

New Virtual Machine Wizard

Welcome to the New Virtual Machine Wizard

What type of configuration do you want?

[✓ Typical (recommended)] Create a Workstation 17.x virtual machine in a few easy steps.

Custom (advanced) Create a virtual machine with advanced options, such as a SCSI controller type, virtual disk type and compatibility with older VMware products.


New Virtual Machine Wizard

Guest Operating System Installation

A virtual machine is like a physical computer; it needs an operating system. How will you install the guest operating system?

Install from:

○ Installer disc:

DVD RW Drive (D:)

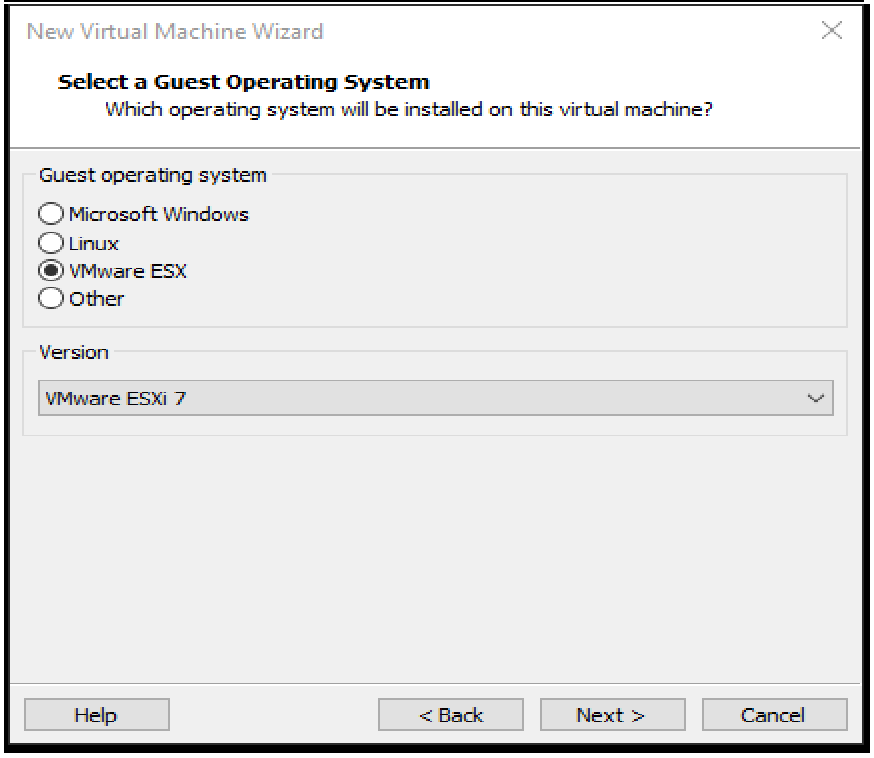
● Installer disc image file (iso):

C:\ISO OS\VMware-VMvisor-Installer-8.0U1a-2181334

⚠ Could not detect which operating system is in this disc image. You will need to specify which operating system will be installed.

○ I will install the operating system later.

The virtual machine will be created with a blank hard disk.


New Virtual Machine Wizard

Select a Guest Operating System

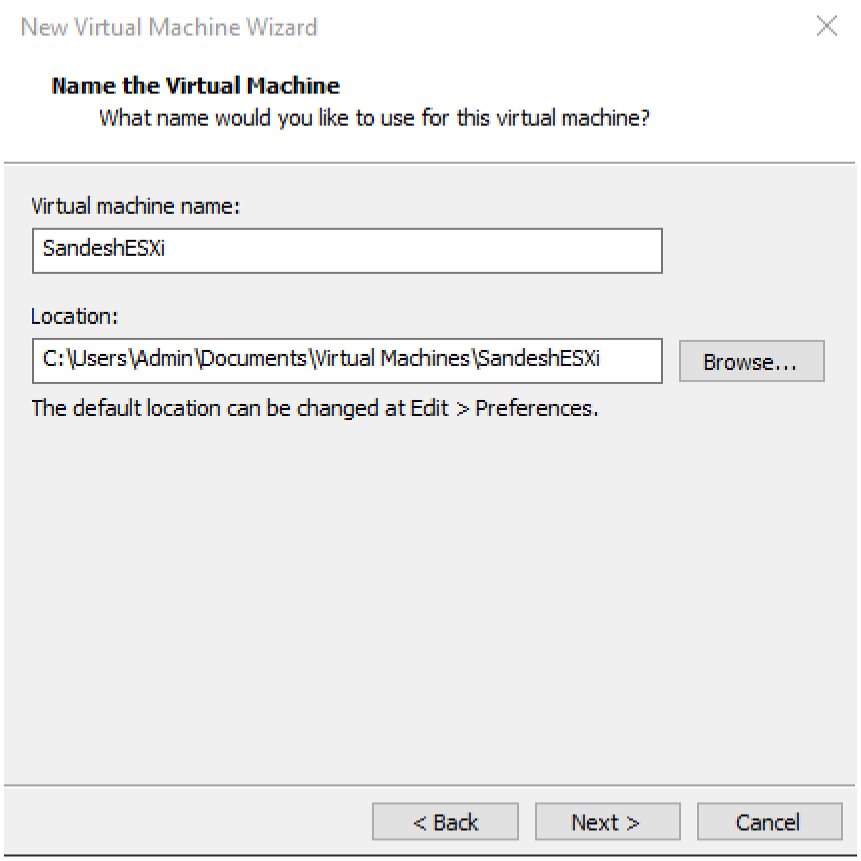
Which operating system will be installed on this virtual machine?

Guest operating system

○ Microsoft Windows
○ Linux
● VMware ESX
○ Other

Version

VMware ESXi 7



New Virtual Machine Wizard

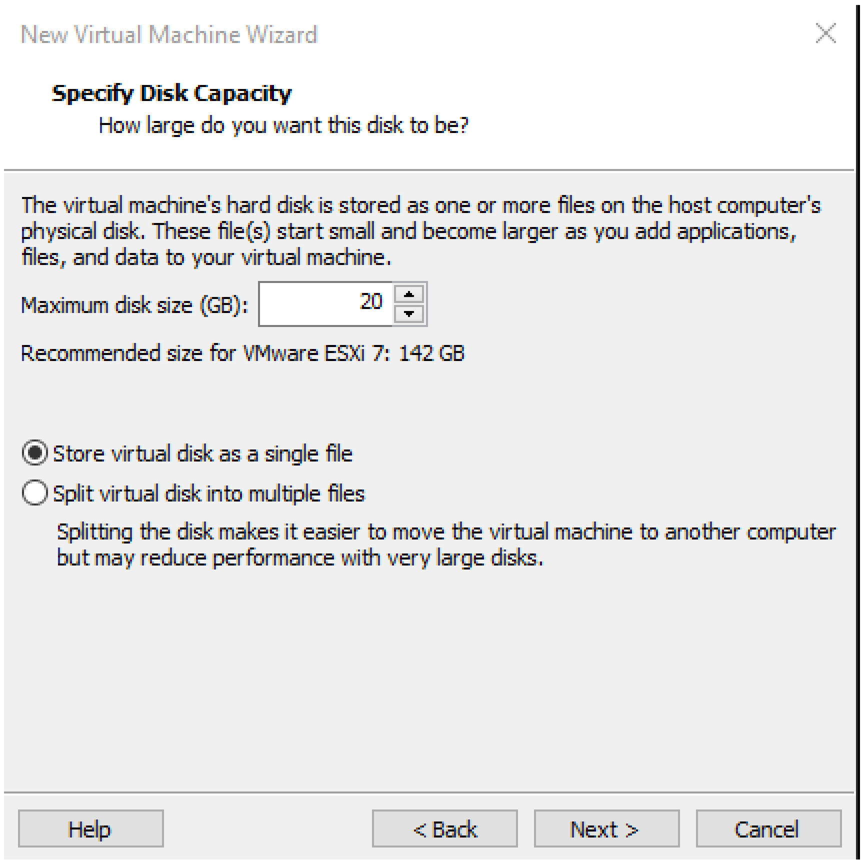
Name the Virtual Machine

What name would you like to use for this virtual machine?

Virtual machine name: SandeshESXi

Location: C:\Users\Admin\Documents\Virtual Machines\SandeshESXi

The default location can be changed at Edit > Preferences.



New Virtual Machine Wizard

Specify Disk Capacity

How large do you want this disk to be?

The virtual machine's hard disk is stored as one or more files on the host computer's physical disk. These file(s) start small and become larger as you add applications, files, and data to your virtual machine.

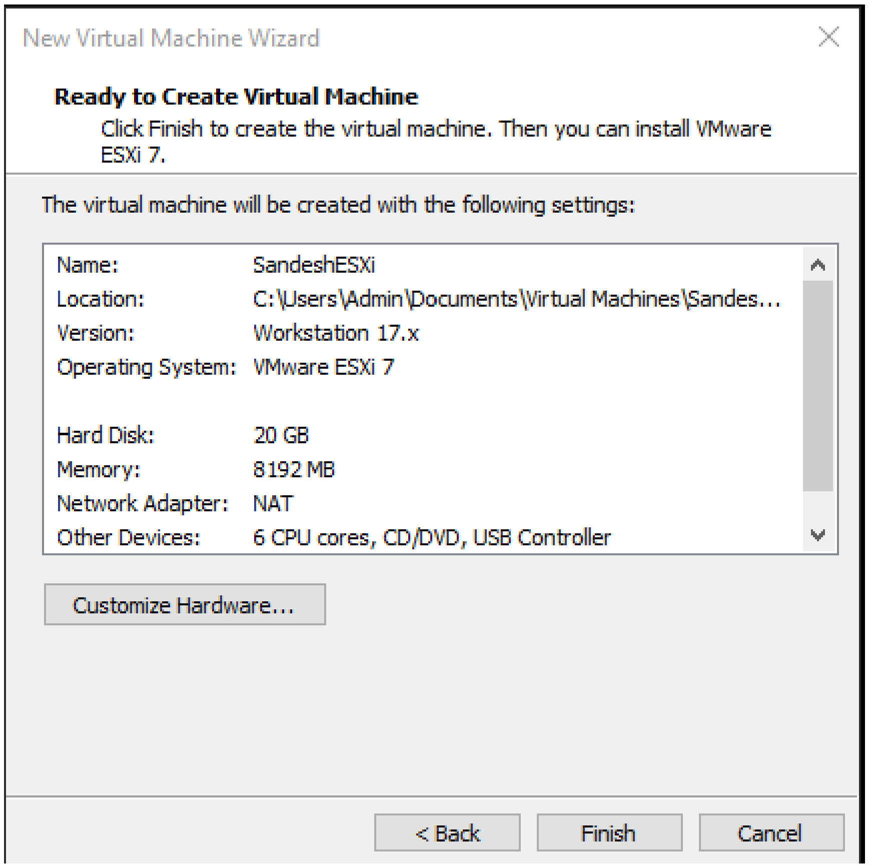
Maximum disk size (GB): 20

Recommended size for VMware ESXi 7: 142 GB

● Store virtual disk as a single file

○ Split virtual disk into multiple files

Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.

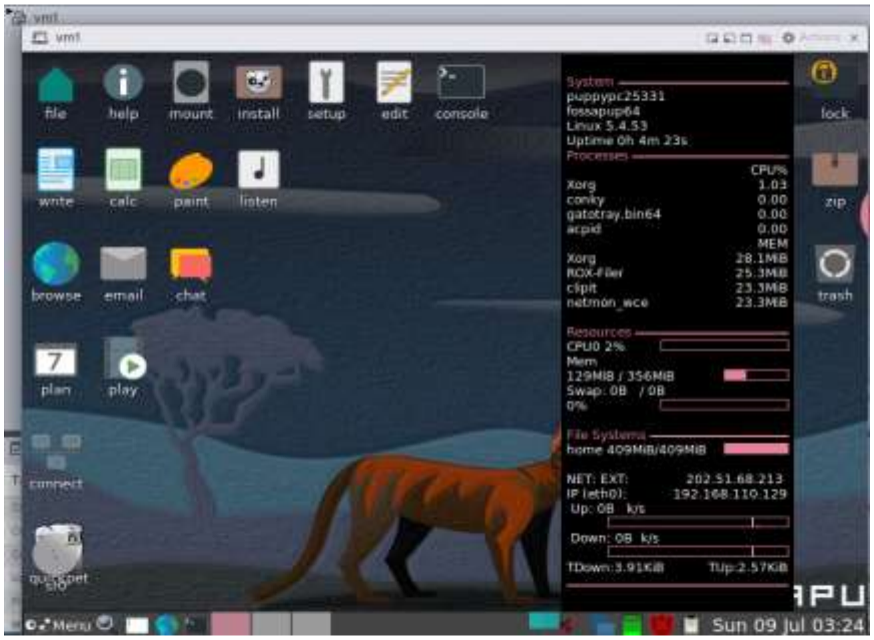


Ready to Create Virtual Machine

Click Finish to create the virtual machine. Then you can install VMware ESXi 7.

The virtual machine will be created with the following settings:

Name:	SandeshESXi
Location:	C:\Users\Admin\Documents\Virtual Machines\Sandes...
Version:	Workstation 17.x
Operating System:	VMware ESXi 7
Hard Disk:	20 GB
Memory:	8192 MB
Network Adapter:	NAT
Other Devices:	6 CPU cores, CD/DVD, USB Controller



```
System
puppypc25331
fossapup64
Linux 5.4.53
Uptime 0h 4m 23s
Processes
                         CPU%
Xorg                     1.03
conky                    0.00
gatotray.bin64           0.00
acpid                    0.00
                          MEM
Xorg                   28.1MB
ROX-Filer              25.3MB
clipit                 23.3MB
netmon_wce             23.3MB

Resources
CPU0 2%
Mem
129MiB / 356MiB
Swap: 0B   / 0B
0%

File Systems
home 409MiB/409MiB

NET: EXT:         202.51.68.213
IP (eth0):      192.168.110.129
 Up: 0B   k/s

 Down: 0B  k/s

TDown:3.91KiB      TUp:2.57KiB
```

⇨ Discussion

A strong and flexible virtualization system can be created by setting up ESXi for Bare Metal Virtualization and running Linux on top of it. Because ESXi is a bare-metal hypervisor, it works with the hardware directly for the best performance and resource use. It builds a strong virtualization layer for hosting numerous guest operating systems by effectively abstracting hardware resources.

Setting up crucial elements including networking, storage, and virtual machine management is a part of the ESXi configuration process. To meet their unique needs, administrators can adjust performance settings, distribute resources, and set up security measures. Organizations may tailor their virtual infrastructure for different workloads because to this flexibility.

⇨ Conclusion

Finally, setting up Linux as a guest OS and configuring ESXi for Bare Metal Virtualization unlocks a powerful and effective virtualization platform. Organizations can create scalable, high-performance virtual infrastructures that can handle the demands of modern computing thanks to the efficiency of bare-metal ESXi and the dependability and versatility of Linux. Businesses can fully utilize Linux and virtualization technologies through this integration to promote innovation, maximize resource utilization, and achieve operational excellence.