

## 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 (VMM), is a software program that allows us to run multiple virtual machines (VMs) on a single physical computer. Each virtual machine has its own operating system and applications, just like a physical computer would. The hypervisor acts as a middleman, allocating the physical resources of the host machine (CPU, memory, storage) to the guest VMs as needed. There are two main types of hypervisors categorized based on their interaction with the physical machine's hardware:

#### 1. Type 1 Hypervisor (Bare Metal Hypervisor)

These hypervisor are installed directly on the physical server's hardware, bypassing the need for a host operating system. It acts like a lightweight OS, managing resources and allocating them to VMs. It offers better performance and security due to direct hardware access and isolation from the host OS vulnerabilities.

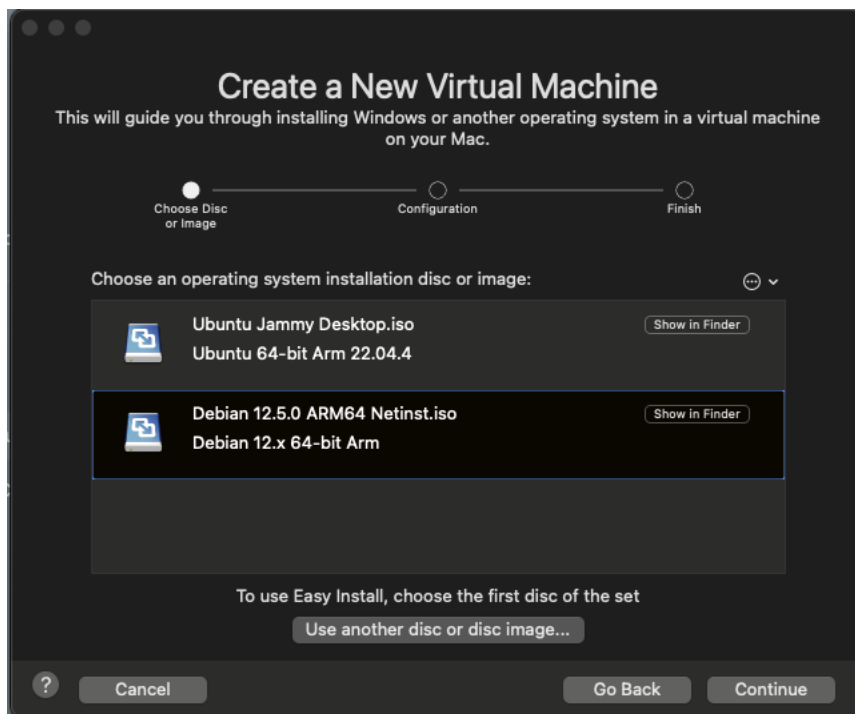
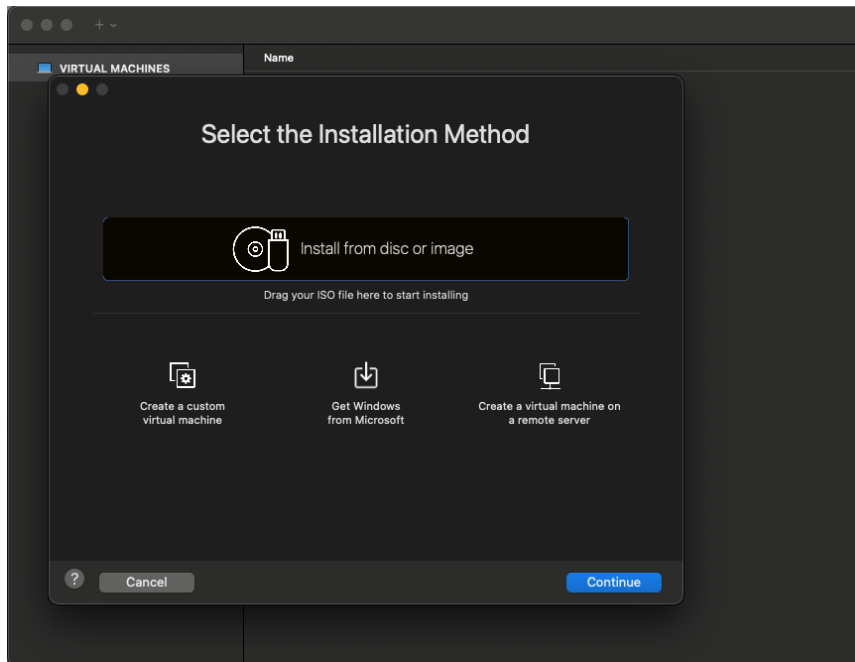
#### 2. Type 2 Hypervisor (Hosted Hypervisor)

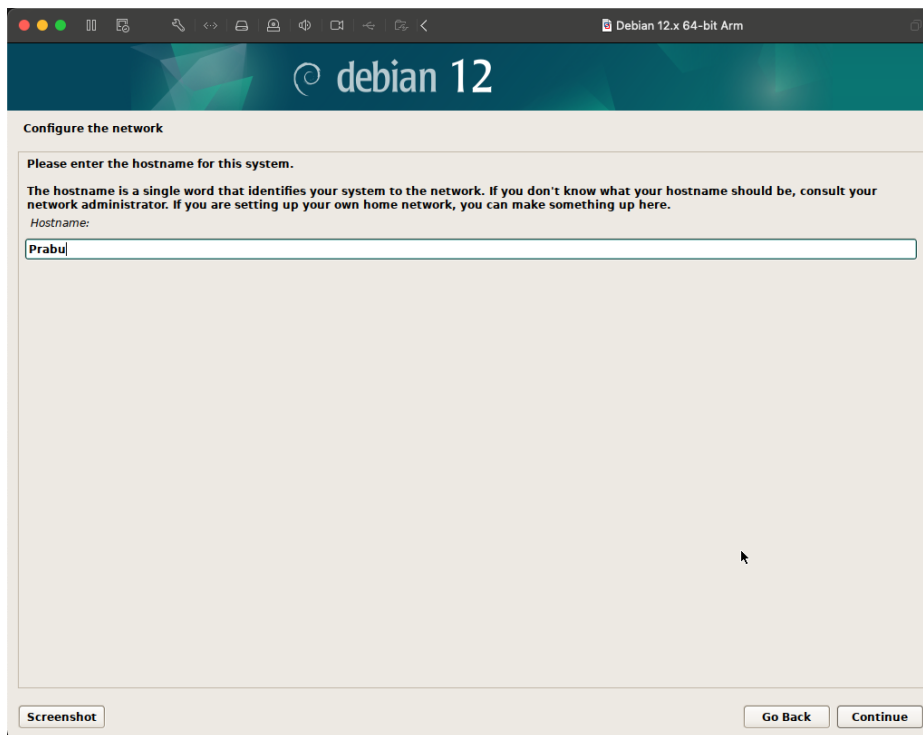
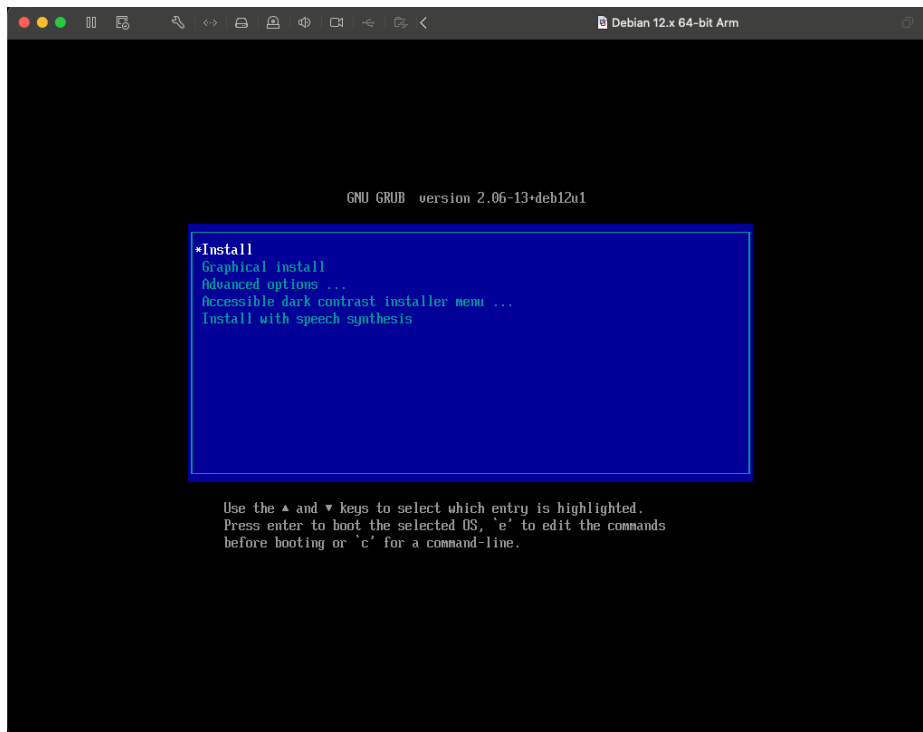
These hypervisor runs on top of an existing operating system on the physical machine like any other software. It is less resource-intensive compared to Type 1, making it suitable for development or desktop virtualization. Also the setup is easier but might slightly lower performance due to the additional layer of the host OS.

### Procedure:

1. Download vm workstation pro trial- as per your host operating system
2. Install it
3. Download at least two ISO file of an operating system other than your host OS.
4. Create VM on Hypervisor and load OS
5. Run the VM

## Screenshots:





## Discussion:

We successfully installed Debian and Ubuntu 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 Debian and Ubuntu 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.

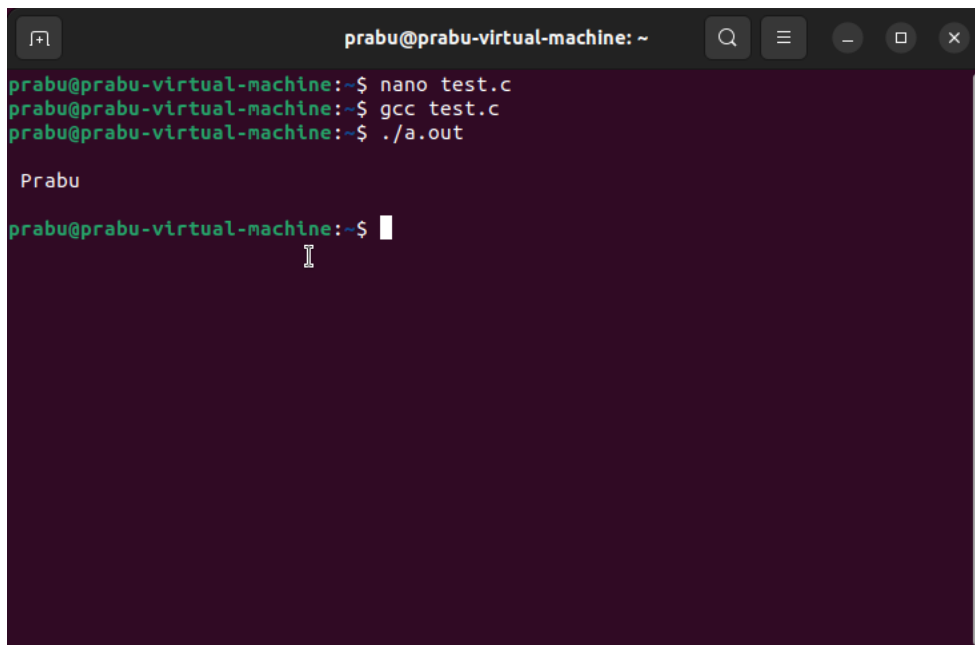
## Lab 2: Run a simple C program in virtual Machine using Type 2 virtualization.

### Introduction:

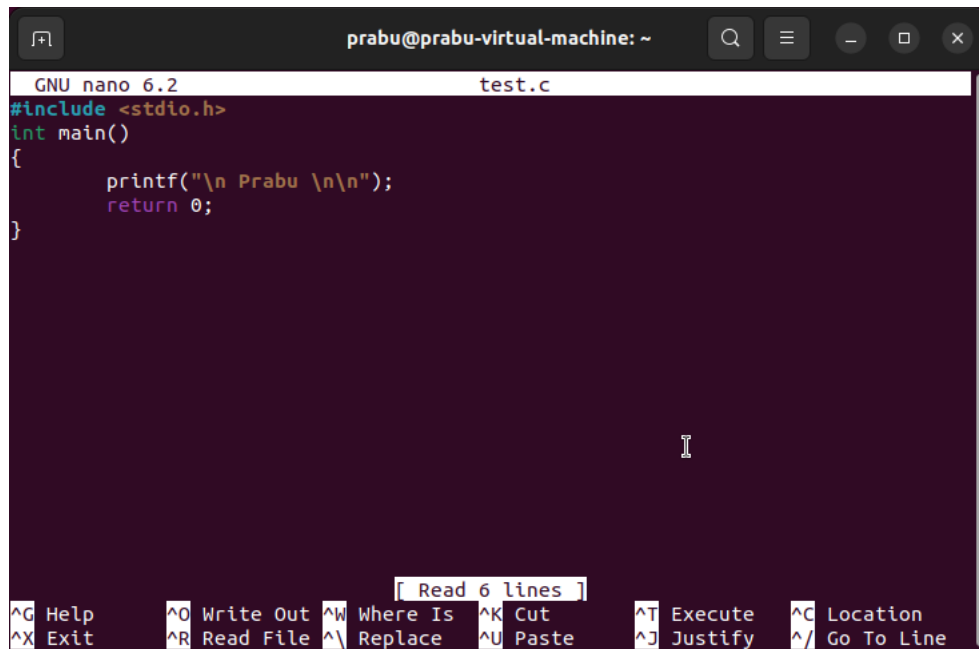
In this lab work, we focus on compiling and running a C program in a VM. To perform this task, first we need to install the GCC compiler on the VM which is a critical tool for compiling C programming language. Following steps were followed while running the C program:

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 test.c`
4. After creating the file, write the code on the file.
5. Save the file.
6. Compile the file using `gcc test.c`
7. Then run the file, using `./a.out`

### Screenshot:



```
prabu@prabu-virtual-machine: ~  
prabu@prabu-virtual-machine:~$ nano test.c  
prabu@prabu-virtual-machine:~$ gcc test.c  
prabu@prabu-virtual-machine:~$ ./a.out  
  
Prabu  
prabu@prabu-virtual-machine:~$
```

A screenshot of a terminal window titled 'prabu@prabu-virtual-machine: ~'. The window shows the GNU nano 6.2 editor editing a file named 'test.c'. The code in the editor is: 

```
#include <stdio.h>
int main()
{
    printf("\n Prabu \n\n");
    return 0;
}
```

 The bottom of the window displays a status bar with various keyboard shortcuts: ^G Help, ^O Write Out, ^W Where Is, ^K Cut, ^T Execute, ^C Location, ^X Exit, ^R Read File, ^\ Replace, ^U Paste, ^J Justify, and ^\_ Go To Line. A small box above the status bar indicates '[ Read 6 lines ]'.

```
prabu@prabu-virtual-machine: ~
GNU nano 6.2 test.c
#include <stdio.h>
int main()
{
    printf("\n Prabu \n\n");
    return 0;
}

[ Read 6 lines ]
^G Help  ^O Write Out  ^W Where Is  ^K Cut  ^T Execute  ^C Location
^X Exit  ^R Read File  ^\ Replace  ^U Paste  ^J Justify  ^_ Go To Line
```

#### Discussion:

First the gcc compiler was installed using the code *sudo apt install gcc*. After the successful installation of the gcc, the above shown code were run to compile and run a simple C program on the machine.

#### Conclusion:

In conclusion, GCC compiler was successfully installed on the VM and a C program was executed on it, demonstrating the system's effectiveness as a development environment. For programmers and developers, the GCC's accessibility and flawless functionality on the VM make it a dependable option, guaranteeing a painless coding experience and effective software development process.

## LAB 3: Configuring ESXi for bare metal (TYPE 1) virtualization and install linux on top of it.

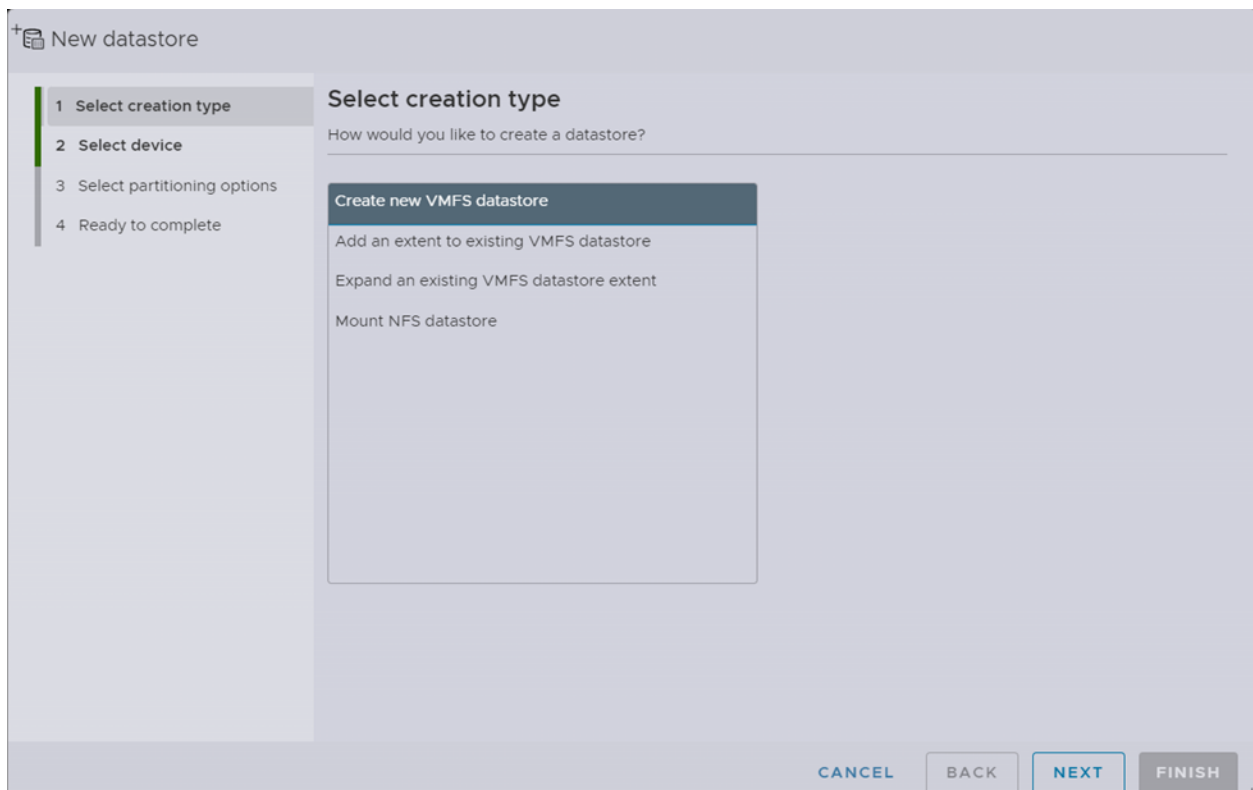
### Introduction:

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.

### 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.
- Access ESXi host through browser.
- Adding a new hard disk drive in VMware.
- Installing new Virtual Machine on ESXi host machine.

### Screenshots:



New datastore - Prabu

1 Select creation type

2 Select device

3 Select partitioning options

4 Ready to complete

Select device

Select a device on which to create a new VMFS partition

Name

Prabu

The following devices are unclaimed and can be used to create a new VMFS datastore

Name	Type	Capacity	Free space
Local VMware, Disk (mpx.vmhba0:C0:T1:L0)	Disk (SSD)	20 GB	20 GB
Local VMware, Disk (mpx.vmhba0:C0:T2:L0)	Disk (SSD)	20 GB	20 GB

2 items

CANCEL

BACK

NEXT

FINISH

New virtual machine - Prabu-vm (ESXi 8.0 virtual machine)

1 Select creation type

2 Select a name and guest OS

3 Select storage

4 Customize settings

5 Ready to complete

Select a name and guest OS

Specify a unique name and OS

Name

Prabu-vm

Virtual machine names can contain up to 80 characters and they must be unique within each ESXi instance.

Identifying the guest operating system here allows the wizard to provide the appropriate defaults for the operating system installation.

Compatibility

ESXi 8.0 virtual machine

Guest OS family

Select Guest OS Family

Guest OS version

Select Guest OS Version

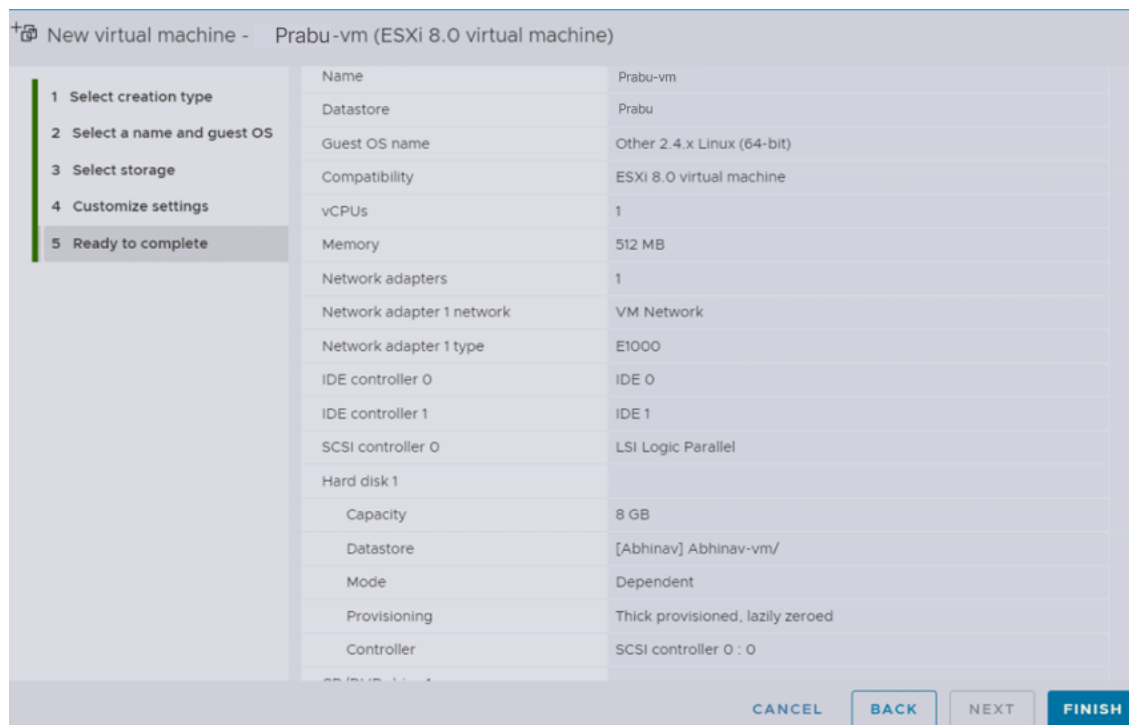
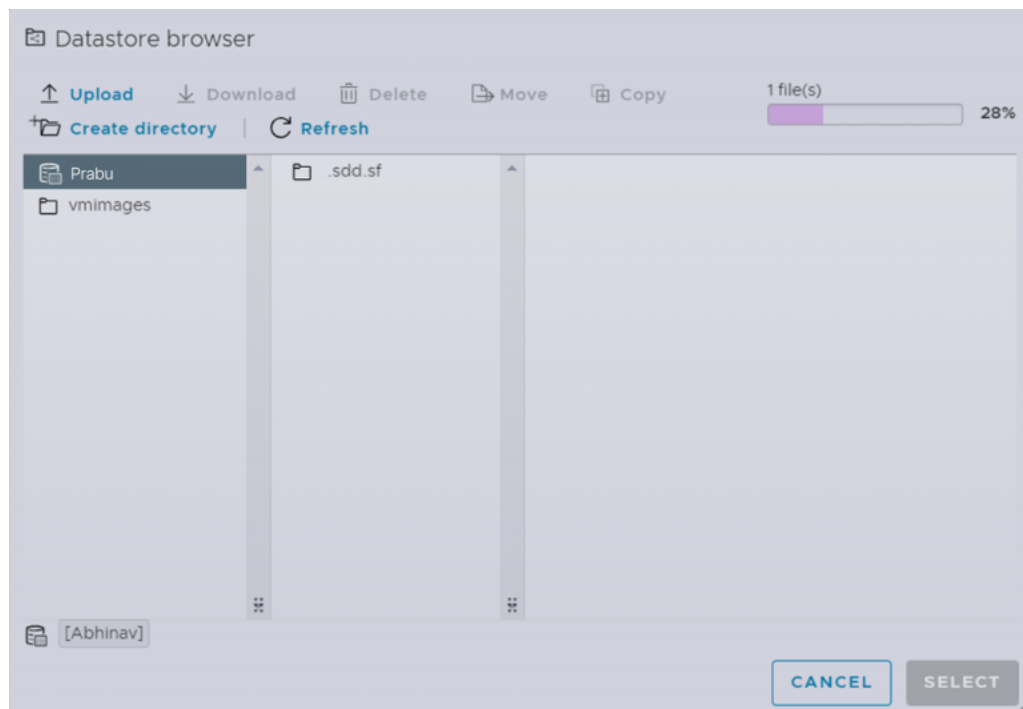
CANCEL

BACK

NEXT

FINISH







#### 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.

## 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.

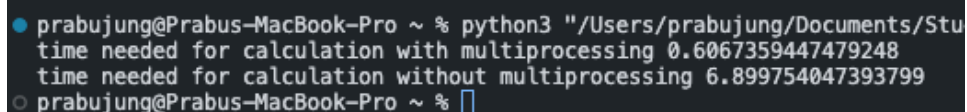
### Code:

```
import multiprocessing as mp
import time
import math
Results_a = []
Results_b = []
def make_calculation_one(numbers):
    for number in numbers:
        Results_a.append(math.sqrt(number **3))

def make_calculation_two(numbers):
    for number in numbers:
        Results_b.append(math.sqrt(number **4))

if __name__ == "__main__":
    number_list= list(range(10000000))
    p1 = mp.Process(target = make_calculation_one, args = (number_list,))
    p2 = mp.Process(target = make_calculation_two, args = (number_list,))
    start = time.time()
    p1.start()
    p2.start()
    end = time.time()
    print("time needed for calculation with multiprocessing", end-start)
    start = time.time()
    make_calculation_one(number_list)
    make_calculation_two(number_list)
    end = time.time()
    print("time needed for calculation without multiprocessing", end-start)
```

### Output:



```
prabujung@Prabus-MacBook-Pro ~ % python3 "/Users/prabujung/Documents/Stu
time needed for calculation with multiprocessing 0.6067359447479248
time needed for calculation without multiprocessing 6.899754047393799
prabujung@Prabus-MacBook-Pro ~ %
```

## Discussion:

We have showcased 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: 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. 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.

### Code:

```
from collections import Counter
from multiprocessing import Pool
def mapper(data):
    # Perform mapping operation on each input data element
    # and return key-value pairs
    results = []
    # Example: Counting the frequency of each word
    for word in data.split(): #splits the input data
        results.append((word, 1)) # append the inputs
    return results

def map_reduce(data, mapper_func, num_workers):
    # Split the input data and distribute it across multiple workers
    pool = Pool(processes=num_workers)
    mapped_data = pool.map(mapper_func, data)
    pool.close() #to close the pool
    pool.join() # wait for completion of all process for final result

    # Flatten the mapped data, join array
    flattened_data = [item for sublist in mapped_data for item in sublist]
    # Reduce the data using Counter
    word_count = Counter()
    for key, value in flattened_data:
        word_count[key] += value
    return word_count

# Example usage
if __name__ == '__main__':
    # Input data from two tiles
    data = [
        "hello world hello",
        "world python world",
        "hello hello hello",
        "world python world",
```

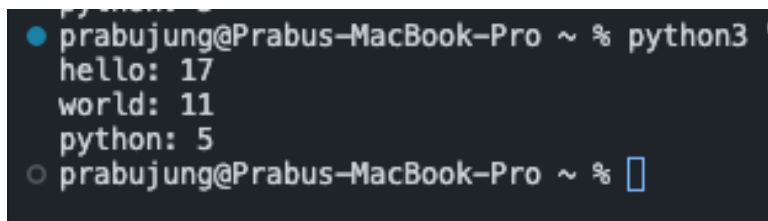
```

    "hello hello hello",
    "world python world",
    "hello hello hello",
    "world python world",
    "hello hello hello",
    "world python world",
    "hello hello hello"
]

# Perform MapReduce operation
result = map_reduce(data, mapper, num_workers=2)
# print(result)
# Print the final result
for key, value in result.items():
    print(f"{key}: {value}")

```

Output:



```

prabujung@Prabus-MacBook-Pro ~ % python3 '
hello: 17
world: 11
python: 5
prabujung@Prabus-MacBook-Pro ~ % 

```

Dicussion:

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 the program 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 6: AWS Academy lab

### Introduction:

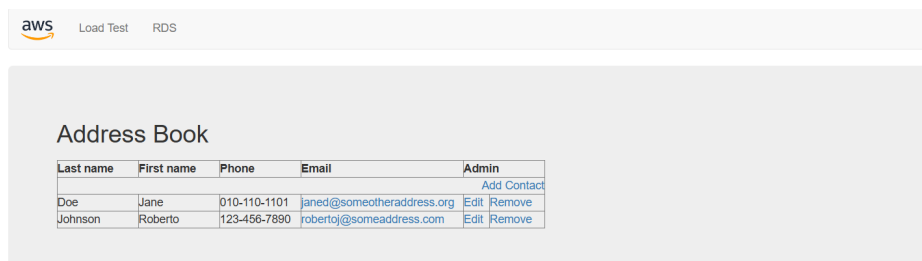
Amazon Web Services (AWS) is a comprehensive and widely adopted cloud computing platform provided by Amazon. Launched in 2006, AWS offers a vast array of services, including computing power, storage options, and networking capabilities, as well as tools for machine learning, analytics, and the Internet of Things (IoT). These services are delivered over the internet, enabling businesses and individuals to build and scale applications quickly and cost-effectively.

### - Building DB Server and Interacting with it Using an App

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks, which allows you to focus on your applications and business. Amazon RDS provides you with six familiar database engines to choose from: Amazon Aurora, Oracle, Microsoft SQL Server, PostgreSQL, MySQL and MariaDB.

### Discussion:

For creating a DB Server, we followed the instructions as suggested in the vocareum workbench. Firstly, a security Group for the RDS DB Instance was created with the name of DB Security Group. Then a DB Subnet Group of name *DB-Subnet-Group* was created. Following this, a Multi-AZ Amazon RDS deployment of a MySQL database instance was configured and launched. Finally, we interacted with the database through a browser. The obtained result after the interaction can be seen in the picture below.



### Conclusion:

Thus, we successfully created and managed a database server using Amazon Web Services (AWS). By leveraging AWS, we were able to set up a robust and reliable database server with ease. The Vocareum workbench provided a seamless interface for interacting with our AWS resources, facilitating efficient database management and operations.

## - Introduction to Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

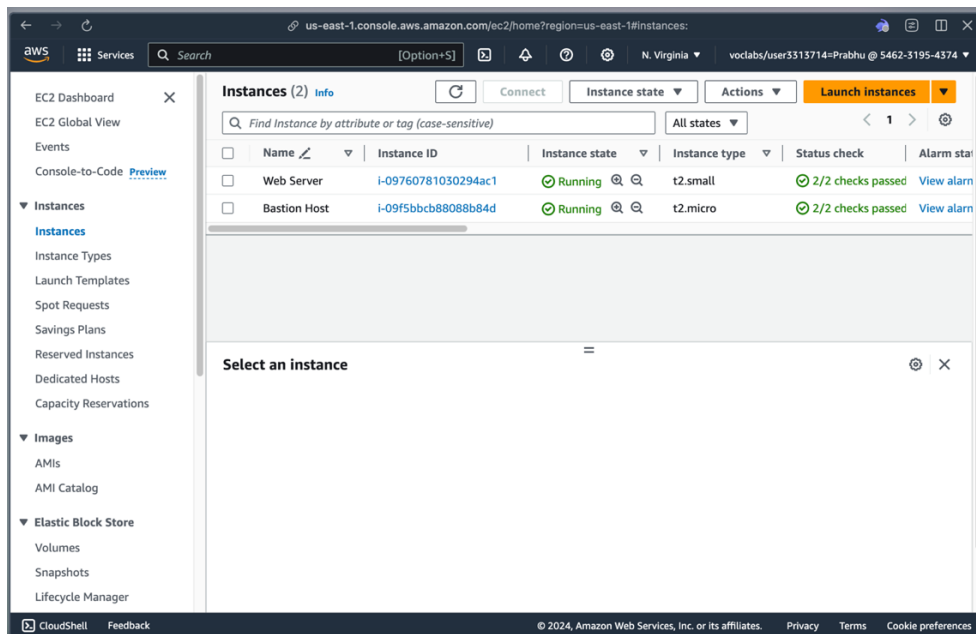
Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change.

Discussion:

We thoroughly followed the instruction provided in the workbench for completing this labwork. Firstly, we launched Amazon EC2 Instance by manual configuration. We set the OS image to Amazon Linux 2023 AMI and instance type to t2.micro. We set the following script in the provided User box section which will run as the root guest OS user when the instance starts.

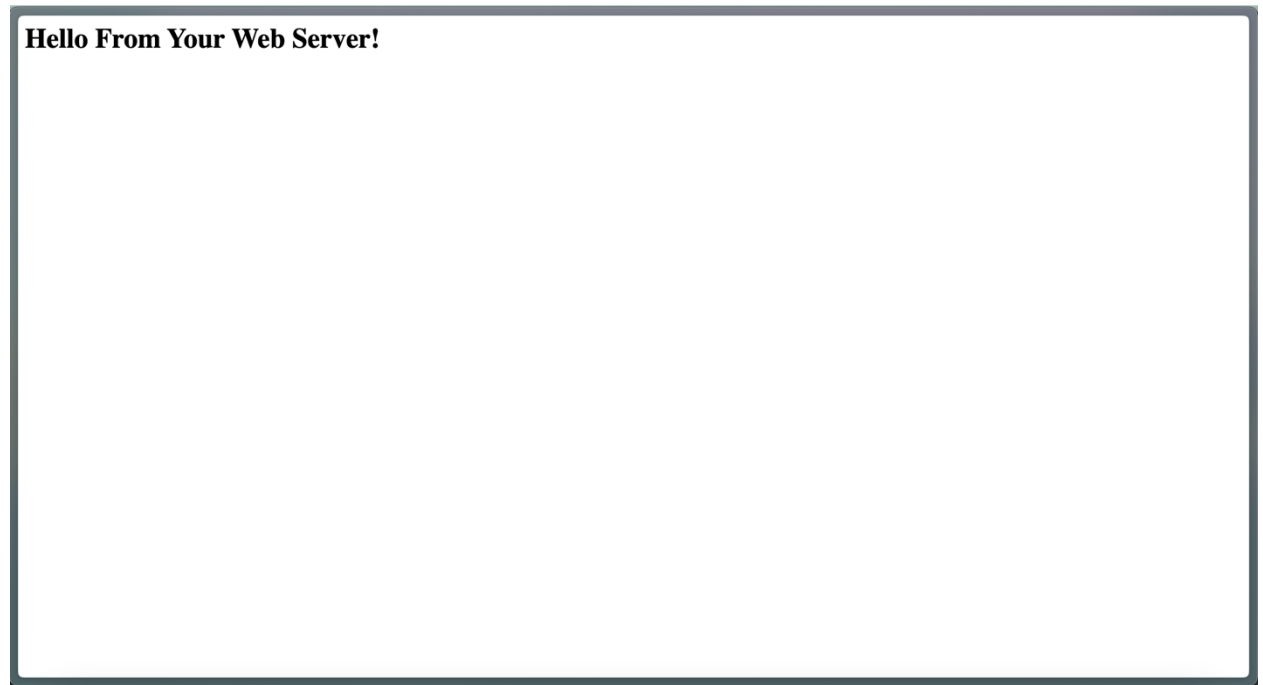
```
#!/bin/bash
dnf install -y httpd
systemctl enable httpd
systemctl start httpd
echo '<html><h1>Hello From Your Web Server!</h1></html>' > /var/www/html/index.html
```

The script will install Apache web server, configure the web server to automatically start on boot, run the web server and create a simple web page. Finally, the instance is launched and following screen is observed.





We accessed the content from the webserver through the Public Ipv4 address and the following screen is observed.



#### Conclusion:

Thus, we delved into the core functionalities of Amazon Web Services (AWS) by deploying and managing virtual servers in the cloud using Amazon EC2 (Elastic Compute Cloud). Through hands-on experience, we grasped the elasticity and scalability of EC2, swiftly launching instances to handle dynamic workloads. The diverse range of instance types provided insight into tailoring resources to specific use cases, while AWS's global availability ensured low-latency access and high availability. Security and compliance were emphasized, with AWS offering robust features for safeguarding instances and adhering to regulations. Cost-effectiveness was highlighted through the pay-as-you-go model, allowing us to optimize expenses by paying for resources as needed.