

LAB 1: Install Virtual Machine and configuring linux on it (Type 2 virtualization).

Introduction:

Type 2 hypervisors are also known as hosted hypervisors, because they are installed on existing OSs, and rely on them for virtualization and resource management. Type 2 hypervisors essentially act like VM management applications running on the existing OS, so you don't need to install separate consoles on another machine remotely. You can perform most tasks using the free built-in features, making them very cost-effective. However, they do not have direct access to the host hardware and resources, all hypervisor and VM activities must pass through the host OS, which inevitably results in latency and wasted resources.

Type 2 Hypervisor examples:

1. VMware Workstation/Fusion/Player
2. Oracle VM VirtualBox
3. VMware Server
4. QEMU
5. Microsoft Virtual PC

VMware Workstation Pro:

VMware Workstation Pro is a hosted (Type 2) hypervisor that runs on x64 versions of Windows and Linux operating systems, it enables users to set up virtual machines (VMs) on a single physical machine and use them simultaneously along with the host machine. Each virtual machine can execute its own operating system, including versions of Microsoft Windows, Linux, BSD, and MS-DOS. VMware Workstation is developed and sold by VMware, Inc. It is a free-of-charge version for non-commercial use. An operating systems license is needed to use proprietary ones such as Windows. Ready-made Linux VMs set up for different purposes are available from several sources.

VMware Workstation Pro is the industry standard for running multiple operating systems as virtual machines (VMs) on a single Linux or Windows PC. IT professionals, developers and businesses who build, test or demo software for any device, platform or cloud rely on Workstation Pro. One of the key features of VMware Workstation Pro is encapsulation. This means that complete environments are contained in a set of files, which can be copied, moved, and accessed quickly and easily. Since an entire disk partition is saved as a file, virtual disks are easy to back up, move, and copy.



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.

[Help](#)

< Back

Next >

Cancel

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:

No drives available

☒ **Installer disc image file (iso):**

C:\Users\user\Desktop\ISO OS\CentOS-Stream-8-x86_1 \ ▾

[Browse...](#)

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.

[Help](#)

< Back

Next >

Cancel

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

Help

< Back

Next >

Cancel

New Virtual Machine Wizard



Name the Virtual Machine

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

Virtual machine name:

CentOS 8 64-bit (2)

Location:

C:\Users\user\Documents\Virtual Machines\CentOS 8 64-bit (2)

Browse...

The default location can be changed at Edit > Preferences.

< Back

Next >

Cancel

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):

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.

Help

< Back

Next >

Cancel

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:

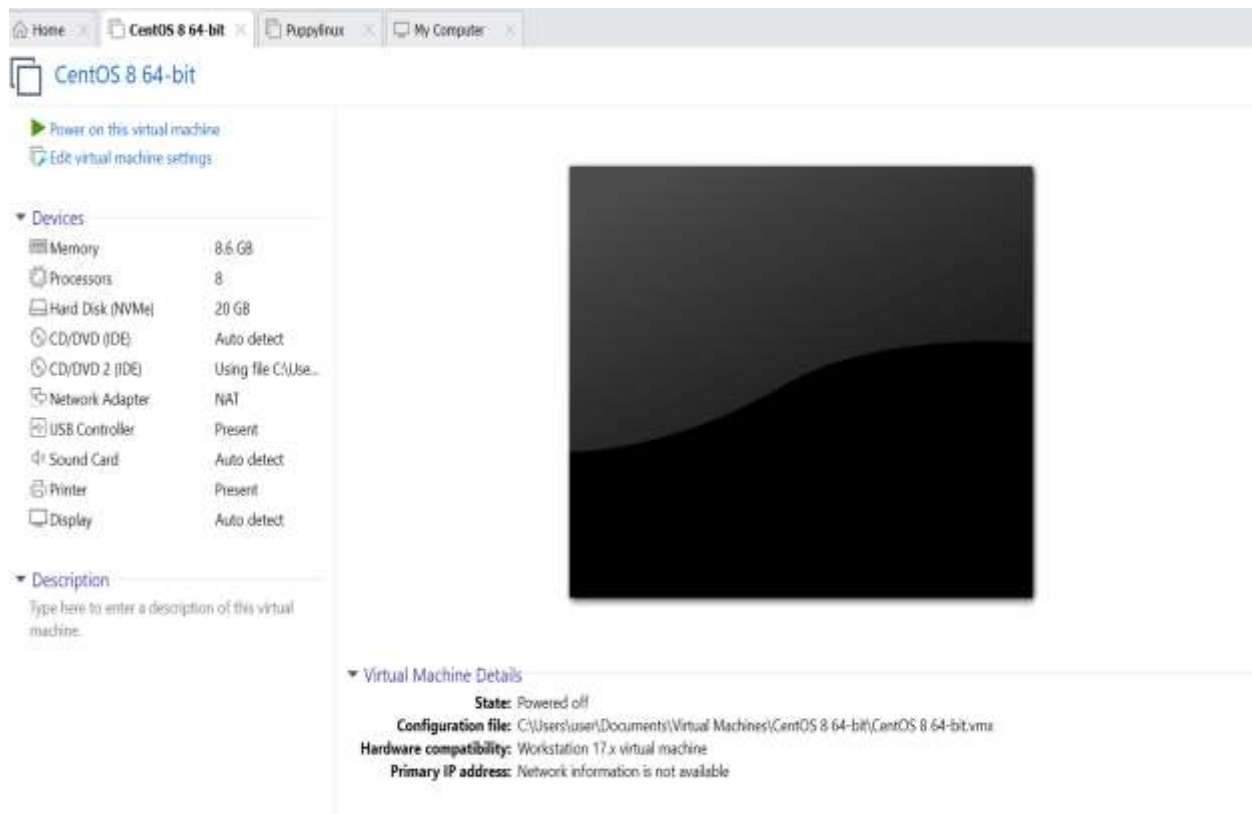
| | |
|-------------------|---|
| Name: | CentOS 8 64-bit (2) |
| Location: | C:\Users\user\Documents\Virtual Machines\CentOS 8 64-b... |
| Version: | Workstation 17.x |
| Operating System: | CentOS 8 64-bit |
| Hard Disk: | 20 GB |
| Memory: | 8192 MB |
| Network Adapter: | NAT |
| Other Devices: | 8 CPU cores, 2 CD/DVDs, USB Controller, Printer, Sound C... |

[Customize Hardware...](#)

< Back

Finish

Cancel



Steps:

1. Download vm workstation pro trial- as per your host operating system
2. Install it vm workstation pro
check install windows hypervisor platform (WHP)
check enhance keyboard driver
after install restart
3. Download at least one operating system iso file other than your host OS
Eg Centos, puppy linux...
4. Create VM on Hypervisor and load OS
5. Run the VM

Discussion:

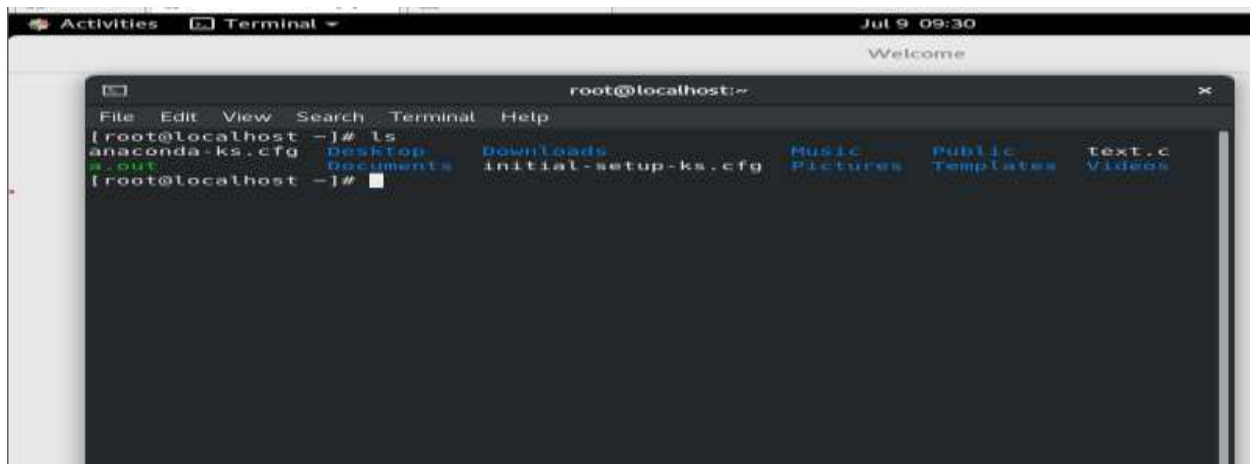
While Installing VMware Workstation Pro and configuring CentOS certain challenges occurred. To ensure that system meets the minimum requirements for VMware Workstation Pro and CentOS. Check if CPU supports virtualization and if the version of VMware Workstation Pro is compatible with operating system. If encounter errors during the installation of VMware Workstation Pro, make sure administrative privileges and that no other virtualization software is running on system. Temporarily disabling antivirus software can also help avoid conflicts. Try downloading the installation package again in case it was corrupted during the initial download. After installing VMware Workstation Pro, it may need to activate it using a license key. When setting up CentOS in VMware Workstation Pro, ensure that the virtual machine's network adapter is properly configured. If the network connection in the virtual machine is not working, check if the host system's network connection is stable and troubleshoot accordingly. If problems occurred with CentOS running in VMware Workstation Pro, installed the VMware Tools package inside the virtual machine. This package provides drivers and utilities to optimize the virtual machine's performance and enable features like copy-paste and drag-and-drop functionality between the host and guest systems. If CentOS is not functioning as expected, review the virtual machine's settings in VMware Workstation Pro. Verify that the allocated resources such as CPU, memory, and disk space are appropriate for your needs. Adjust settings and reinstalling or reconfiguring the CentOS virtual machine. VMware's official documentation, user forums, and other community resources can often provide valuable insights and solutions.

Conclusion:

In conclusion, we have installed VMware Workstation Pro and configure centos properly. Created a new virtual machine, Open VMware Workstation Pro and click on "Create a New Virtual Machine." Select the option to install the operating system later. Configure virtual machine settings, Specify the desired hardware settings for your virtual machine, such as the number of processors, amount of RAM, and network adapter configuration. Start the virtual machine. Power on the virtual machine and proceed with the CentOS installation. In this way, we have successfully installed VMware Workstation Pro and configured CentOS within a virtual machine, allowing you to run and experiment with CentOS in a virtualized environment.

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

Compile and Run C Program in Centos Stream



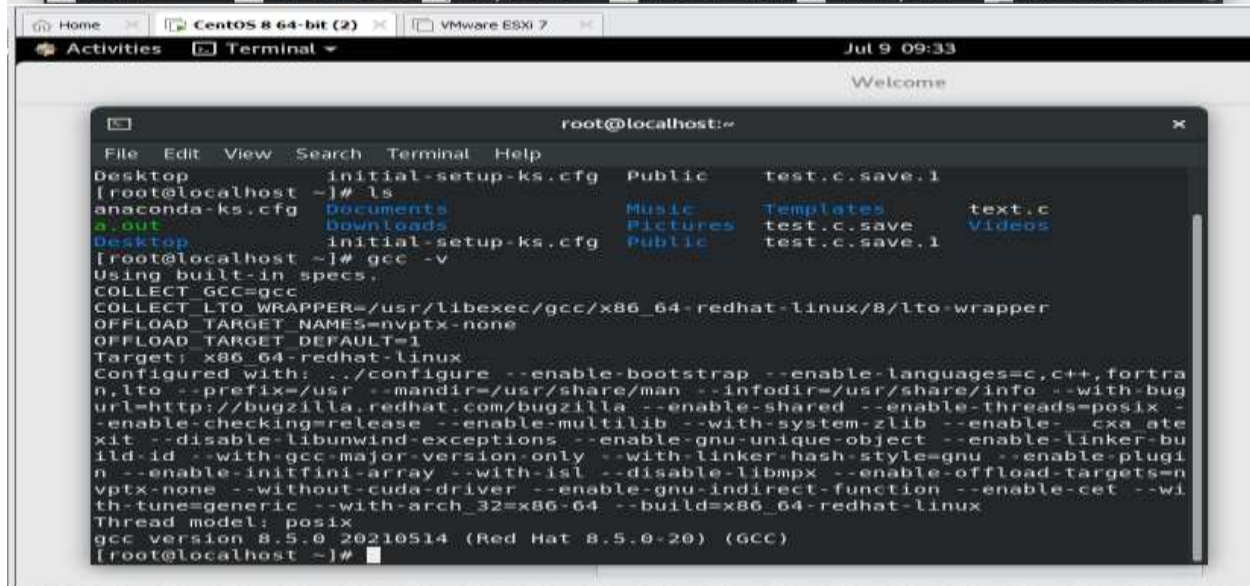
A terminal window titled 'root@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command 'ls' has been executed, showing a directory listing of files and folders: anaconda-ks.cfg, Desktop, Downloads, initial-setup-ks.cfg, Music, Public, Templates, text.c, and Videos. The prompt is '[root@localhost ~]# '.

```
root@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# ls  
anaconda-ks.cfg Desktop Downloads initial-setup-ks.cfg Music Public Templates text.c Videos  
a.out Documents  
[root@localhost ~]#
```



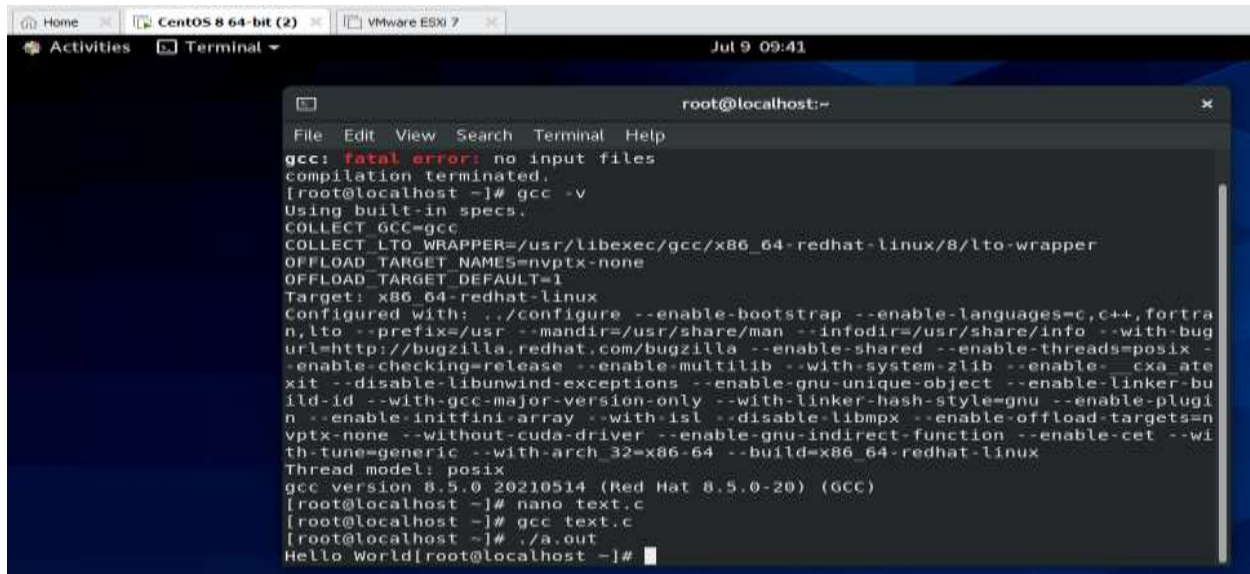
A terminal window titled 'root@localhost:~' showing the GNU nano 2.9.8 editor editing 'test.c'. The code is a simple C program that prints 'Hello World'. The prompt is '[root@localhost ~]# '.

```
root@localhost:~  
File Edit View Search Terminal Help  
GNU nano 2.9.8 test.c Modified  
#include <stdio.h>  
// main function -  
// where the execution of program begins  
int main()  
{  
    // prints hello world  
    printf("Hello World");  
    return 0;  
}
```



A terminal window titled 'root@localhost:~' showing the compilation and execution of the C program. The command 'gcc -v' has been executed, showing the compiler version and options. The prompt is '[root@localhost ~]# '.

```
root@localhost:~  
File Edit View Search Terminal Help  
Desktop initial-setup-ks.cfg Public test.c.save.1  
[root@localhost ~]# ls  
anaconda-ks.cfg Documents Music Templates text.c  
a.out Downloads Pictures test.c.save Videos  
Desktop initial-setup-ks.cfg Public test.c.save.1  
[root@localhost ~]# gcc -v  
Using built-in specs.  
COLLECT_GCC=gcc  
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/8/lto-wrapper  
OFFLOAD_TARGET_NAMES=nvptx-none  
OFFLOAD_TARGET_DEFAULT=1  
Target: x86_64-redhat-linux  
Configured with: ../configure --enable-bootstrap --enable-languages=c,c++,fortran,lto --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --with-bug-url=http://bugzilla.redhat.com/bugzilla --enable-shared --enable-threads=posix --enable-checking=release --enable-multilib --with-system-zlib --enable-cxa-ate-xt --disable-libunwind-exceptions --enable-gnu-unique-object --enable-linker-bu-ild-id --with-gcc-major-version-only --with-linker-hash-style=gnu --enable-plugi-n --enable-initfini-array --with-lsl --disable-libmpx --enable-offload-targets=nvptx-none --without-cuda-driver --enable-gnu-indirect-function --enable-cet --wi-th-tune=generic --with-arch_32=x86-64 --build=x86_64-redhat-linux  
Thread model: posix  
gcc version 8.5.0 20210514 (Red Hat 8.5.0-20) (GCC)  
[root@localhost ~]#
```

A screenshot of a terminal window within a VMware ESXi 7 virtual machine. The terminal shows the output of the 'gcc -v' command, displaying the GCC version 8.5.0 and its configuration options. The user then creates a file 'text.c' using 'nano', writes a 'Hello World' program, and compiles it with 'gcc text.c'. The final output is 'Hello World'.

Steps:

1. Install GCC on CentOS 8
2. Verify whether GCC is installed successfully or not in your Linux System

`gcc -v` or `gcc --version`

3. write a simple basic C program to print Your Name on screen Open any command-line editor like nano, or any other TextEditor and write or copy-paste the following code.

4. save your file with a .c extension eg. firstProgram.c

5. compile a file

`gcc firstProgram.c`

6. To run file

`./a.out`

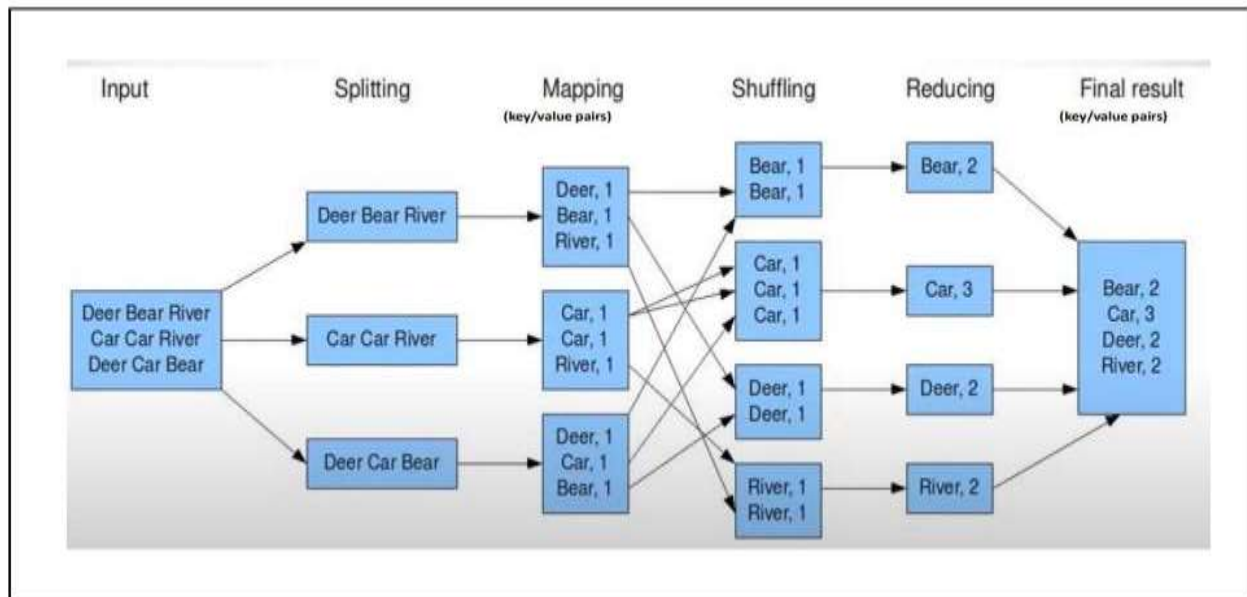
Discussion:

After the successful installation of virtual machine and configuring centos. Open a terminal within the virtual machine. Install the necessary software packages for C programming. This typically includes a C compiler like GCC (GNU Compiler Collection). Open a terminal in CentOS Stream to execute commands. Write and compile the C program Use a text editor within the virtual machine to write your C program. Save the file. In the terminal, navigate to the directory where the C program is saved. Compile the program using the C compiler. Run the C program. After successful compilation, run the program by executing the generated executable.

Conclusion:

In conclusion, running a C program in a virtual machine using Type 2 virtualization offers several advantages. By running the C program within a virtual machine, it creates a self-contained environment separate from the host operating system. This isolation ensures that any changes or issues with the program do not affect the host system. Overall, using a Type 2 virtualization approach to run a C program in a virtual machine provides flexibility, isolation, and control over the development and execution environment. It is a reliable method for developing, testing, and running C programs in a platform-independent and reproducible manner.

Lab 3 TITLE: PYTHON PROGRAM TO DEMONSTRATE THE WORKING OF MAP REDUCE.



```
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
    '''
    process1=[('hello', 1), ('world', 1), ('hello', 1)]
    procees2=[('world', 1), ('python', 1), ('world', 1)]
    mapped_data -> [process1,proces2] =
    [[('hello', 1), ('world', 1), ('hello', 1)],[(('world', 1), ('python', 1),
    ('world', 1)]]
    flatten data ->[(('hello', 1), ('world', 1), ('hello', 1), ('world', 1),
    ('python', 1), ('world', 1)]
    '''
    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 = [
        "Deer Beer Gear",
        "Fear Deer Near",
        "Hello python",
        "Hello World"
    ]

    # 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:



```

Run: main
C:\Users\user\PycharmProjects\pythonProject3\venv\Scripts\python.exe C:\Users\user\PycharmProjects\pythonProject3\main.py
Deer: 2
Beer: 1
Gear: 1
Fear: 1
Near: 1
Hello: 2
python: 1
World: 1
Process finished with exit code 0

```

MapReduce:

MapReduce is a programming model commonly used for processing large amounts of data in a distributed or parallel computing environment. It divides a computation into two main steps: the map step, which processes the data in parallel, and the reduce step, which aggregates the results. MapReduce is a programming model and software framework designed to process and analyze large volumes of data in a distributed computing environment. It was popularized by Google and has become a fundamental concept in big data processing.

Apache Hadoop is a popular open-source framework that implements the MapReduce model. It provides a complete ecosystem for distributed storage and processing of big data, and it has revolutionized the field of data analytics by enabling the processing of massive datasets across commodity hardware.

Steps:

1. The first line imports the Counter class from the collections module and the Pool class from the multiprocessing module. The Counter class is a convenient tool for counting occurrences of elements in a collection, and the Pool class provides a way to create a pool of worker processes.
2. The mapper function takes a piece of data as input and processes it. In this example, the function splits the input string into words and appends each word with a count of 1 to the results list. This step represents the map phase of MapReduce.
3. The map_reduce function performs the MapReduce operation. It takes three arguments: the input data, the mapper function to be applied to each data element, and the number of worker processes to be used.
4. Inside the map_reduce function, a Pool object is created with the specified number of worker processes. The map method of the Pool object is then called, which applies the mapper_func to each element in the data list in parallel, using the available worker processes. The result is a list of mapped data.
5. After the mapping step is complete, the pool is closed and the join method is called to wait for all the worker processes to finish.
6. The mapped data is flattened into a single list using a list comprehension. Each element in the flattened list is a tuple representing a word and its count.
7. The flattened data is then processed using the Counter class to count the occurrences of each word.
8. Finally, the function returns the word_count object, which is a Counter containing the word frequencies.
9. In the example usage section, a list called data is defined with two input strings. The map_reduce function is called with data, the mapper function, and the number of worker processes set to 2.
10. The result is printed by iterating over the key-value pairs in the word_count object and displaying them.

Discussion:

In this program, the mapper function takes a text string and splits it into words. It then counts the occurrence of each word and returns the intermediate key-value pairs as a list of tuples. The reducer function receives a word and a list of counts, and it calculates the total count for that word. The map reduce function is the entry point of our program. The intermediate results are flattened into a single list. Then, the reducer function is applied to each group to calculate the final word count. Finally, we define an input text list with some sample input data and call the map reduce function. The result is printed, showing the word count for each word in the input. The program uses the multiprocessing module to achieve parallelism. It splits the input data across multiple processes

for mapping, and then combines the results in the reduce step. This approach works well for multi-core systems, but it's worth mentioning that there are other distributed frameworks, such as Apache Hadoop or Apache Spark, that provide more robust implementations of the MapReduce paradigm for large-scale distributed computing.

Conclusion:

In conclusion, the Python program to demonstrate the working of MapReduce provides a practical implementation of the MapReduce paradigm, which is a widely used technique for processing large-scale data sets in a distributed computing environment. The program showcases the fundamental steps of MapReduce: mapping, shuffling, and reducing. The mapping step involves dividing the input data into smaller chunks and applying a map function to each chunk, transforming it into a set of key-value pairs. These key-value pairs are then shuffled, grouping together all the values associated with the same key. Finally, the reducing step aggregates the values for each key, producing the final output. As we increase the data, we have to increase the number of processor to get the output data at real time.

LAB 4 Title : Multi processing program in 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))
    # creating processes
    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()
    # completing process
    for p in processes:
        p.join()

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

Output:

```
PS C:\Users\user> & C:/Users/user/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/user/Desktop/from multiprocessing import Lock, Process.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 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-3
Task no 6 is done by Process-4
Task no 7 is done by Process-1
PS C:\Users\user>
```

Discussion:

The code begins by importing the necessary modules. Lock is used for synchronization, Process is used to create processes, Queue is used for inter-process communication, current process is a utility function to get the current process, time is used for the sleep function, and queue is imported for exception handling related to empty queues. The do_job function represents the work to be done by each worker process. It continuously tries to retrieve tasks from the task to accomplish queue using get_nowait(), which raises an exception if the queue is empty. Once a task is obtained, it processes the task, prints it, and puts the result into the Tasks that are done queue along with the name of the current process. The time.sleep(5) introduces a small delay to simulate processing time. The main function sets up the necessary resources and orchestrates the execution. It creates two queues, tasks_to_accomplish and tasks_that_are_done, to hold the tasks and their results, respectively. It then creates a number of worker processes defined by number of processes. Each process is assigned the do_job function as the target, along with the necessary queues as arguments. The processes are started and added to the processes list. After starting all the processes, the code waits for each process to complete by calling p.join() for each process in the processes list. Once all the processes have finished, the results are printed by retrieving items from the tasks_that_are_done queue and printing them. This is essential for multiprocessing to work correctly.

Conclusion:

In conclusion, multiprocessing is a valuable technique in Python for achieving parallelism and improving the performance of programs that can benefit from utilizing multiple processors or cores. It allows for the execution of multiple tasks concurrently, thereby reducing the overall execution time. Depending on the number of task, the code will take some time to show you the output. The output of the following code will vary from time to time. In summary, multiprocessing in Python is a powerful tool for achieving parallelism and improving performance in CPU-bound tasks. However, it requires careful design and consideration of factors such as scalability, communication, synchronization, and exception handling to harness its full potential effectively.

LAB 5 TITLE: CONFIGURING ESXI FOR BARE METAL VIRTUALIZATION AND INSTALL LINUX ON TOP OF ESXI.

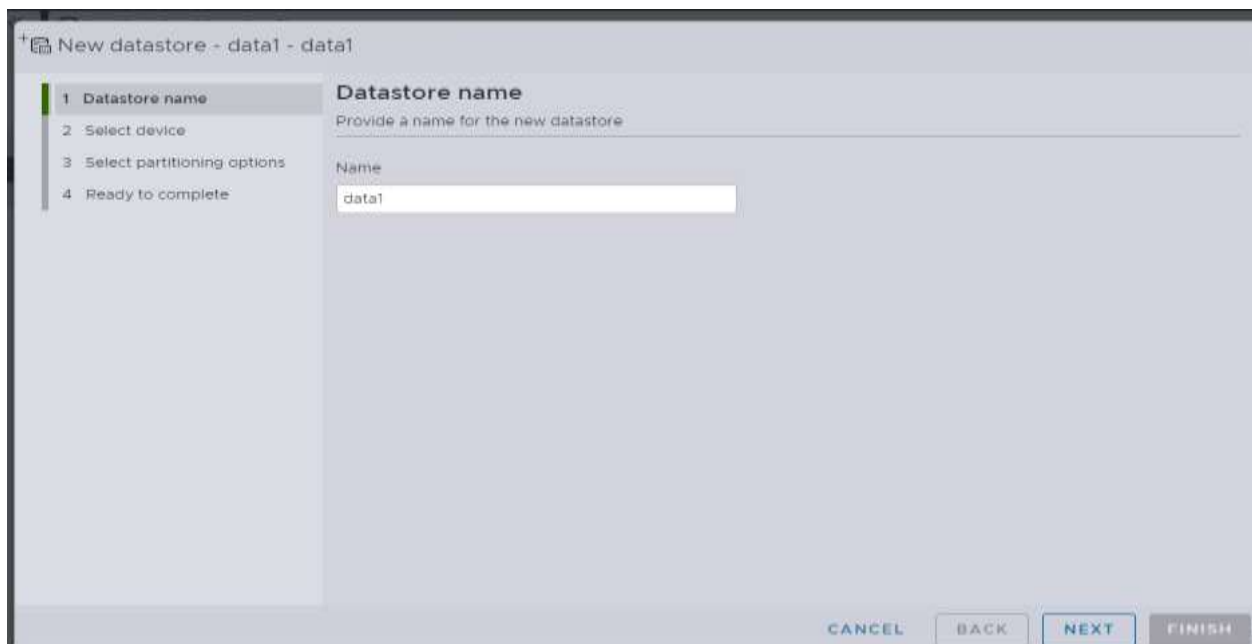
Introduction:

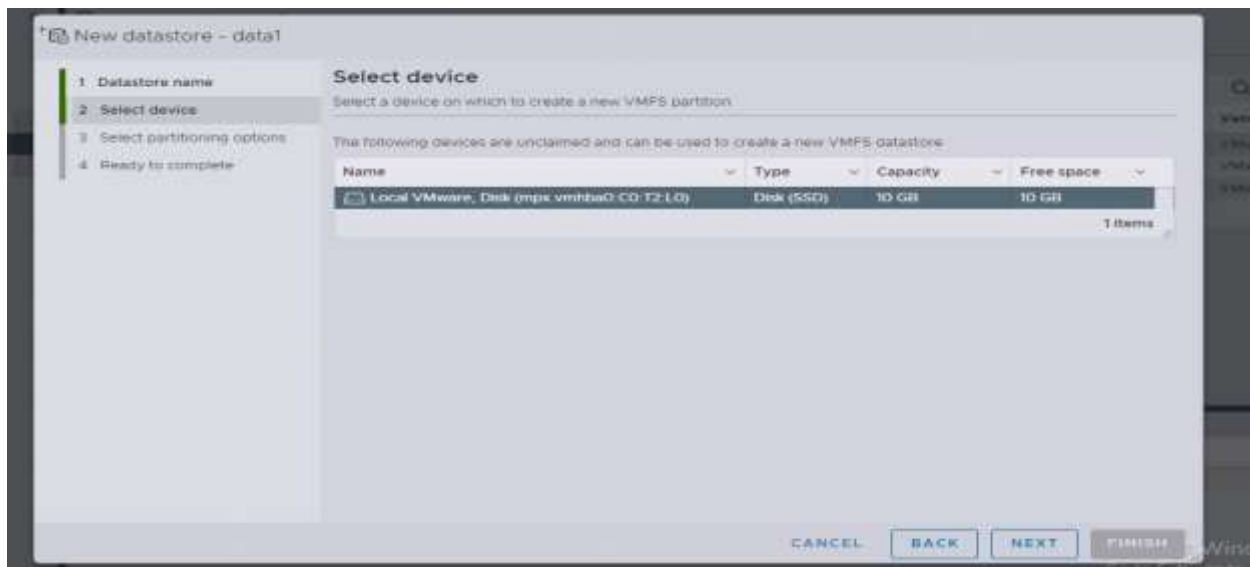
ESXi stands for "ESX integrated hypervisor," is a bare-metal hypervisor developed by VMware. It is part of the VMware vSphere suite, which provides a comprehensive virtualization and cloud computing platform for businesses. ESXi is designed to run directly on physical servers, allowing multiple virtual machines (VMs) to run on a single physical machine.

Some key features of ESXi are:

- 1) Hypervisor
- 2) Virtual Machine Management
- 3) High Availability
- 4) Resource Management
- 5) Security
- 6) vCenter Server Integration

Overall, ESXi is widely used in enterprise environments to consolidate server resources, improve hardware utilization, and simplify IT management through virtualization.





Navigation: Host, Manage, Monitor, Virtual Machines, **Storage**, Networking

localhost.localdomain - Storage

Datastores: adapters, Disks, Persistent Memory

[New datastore](#)
[Increase capacity](#)
[Register a VM](#)
[Datastore browser](#)
[Refresh](#)
[Actions](#)

| Name | Drive Type | Capacity | Provisioned | Free | Type | This datastore | Access |
|-------|------------|----------|-------------|---------|-------|----------------|--------|
| data1 | SSD | 9.75 GB | 1.41 GB | 8.34 GB | VMFS6 | Supported | Single |

1 item

data1

Type: VMFS6

Location: /vmfs/volumes/66aa2105-266a7046-2773-000c29643e95

UUID: 66aa2105-266a7046-2773-000c29643e95

Hosts: 1

Virtual Machines: 0

Storage: 9.75 GB
Used: 1.41 GB
Free: 8.34 GB
Provisioned: 14%

Recent tasks

| Task | Target | Initiator | Queued | Started | Result | Completed |
|------------------------|-----------------------|-----------|---------------------|---------------------|------------------------|---------------------|
| Reconfig vmmx | localhost.localdomain | root | 2019-03-22 08:34:15 | 2019-03-22 08:34:16 | Completed successfully | 2019-03-22 08:34:16 |
| Cluster vmmx Datastore | localhost.localdomain | root | 2019-03-22 08:34:15 | 2019-03-22 08:34:16 | Completed successfully | 2019-03-22 08:34:16 |
| Add Bart Power On | localhost.localdomain | root | 2019-03-22 08:37:26 | 2019-03-22 08:37:26 | Completed successfully | 2019-03-22 08:37:26 |

Activate Windows
Go to Settings to activate Windows.

+ New virtual machine

- Select creation type**
- Select a name and guest OS
- Select storage
- Customize settings
- Ready to complete

Select creation type

How would you like to create a Virtual Machine?

Create a new virtual machine

Deploy a virtual machine from an OVF or OVA file

Register an existing virtual machine

This option guides you through creating a new virtual machine. You will be able to customize processors, memory, network connections, and storage. You will need to install a guest operating system after creation.

CANCEL BACK **NEXT** FINISH

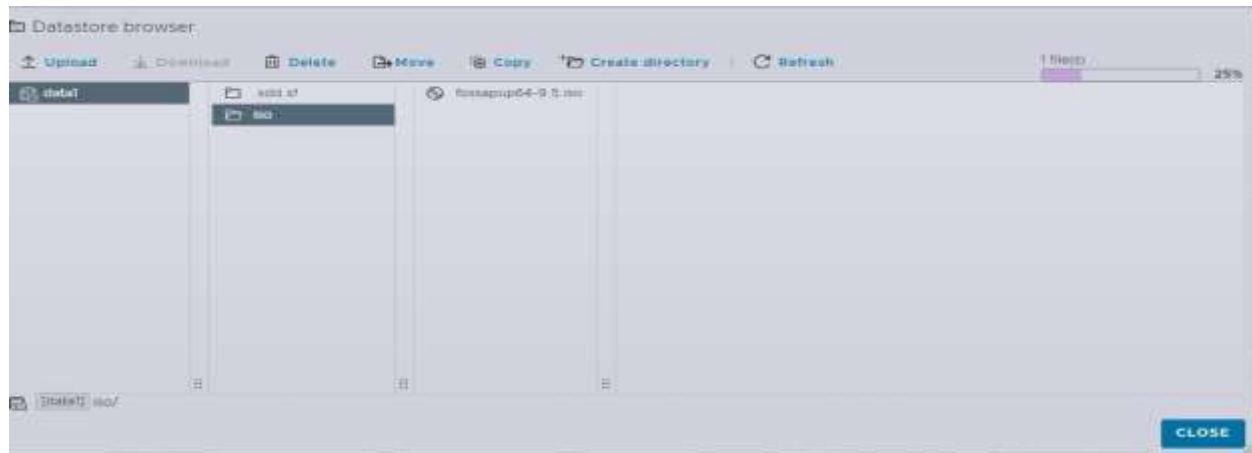
Datastore browser

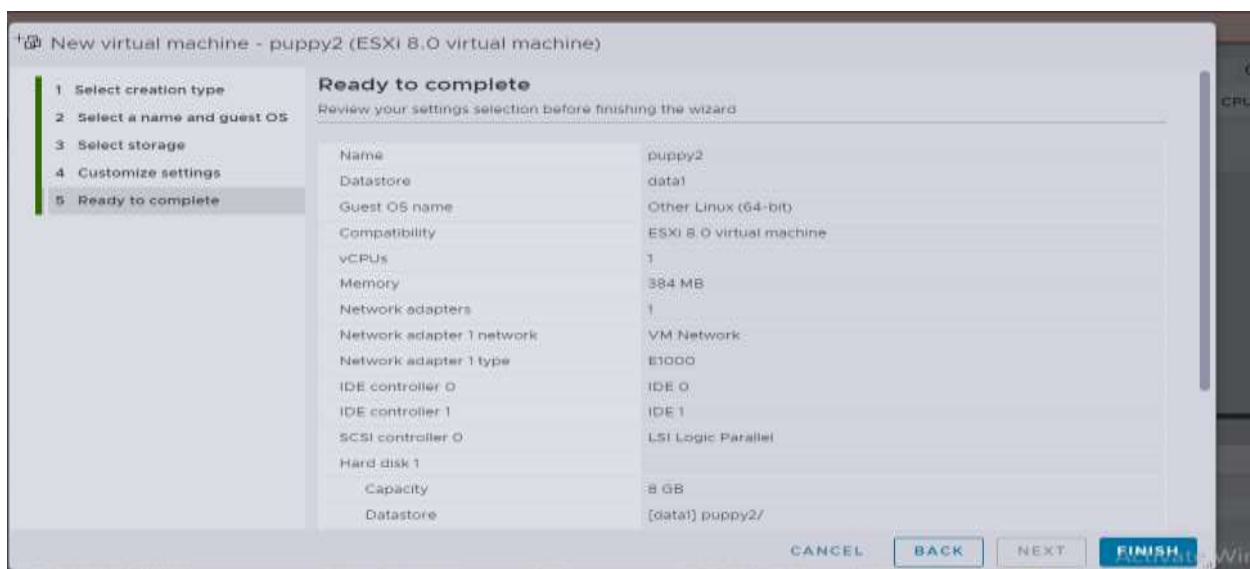
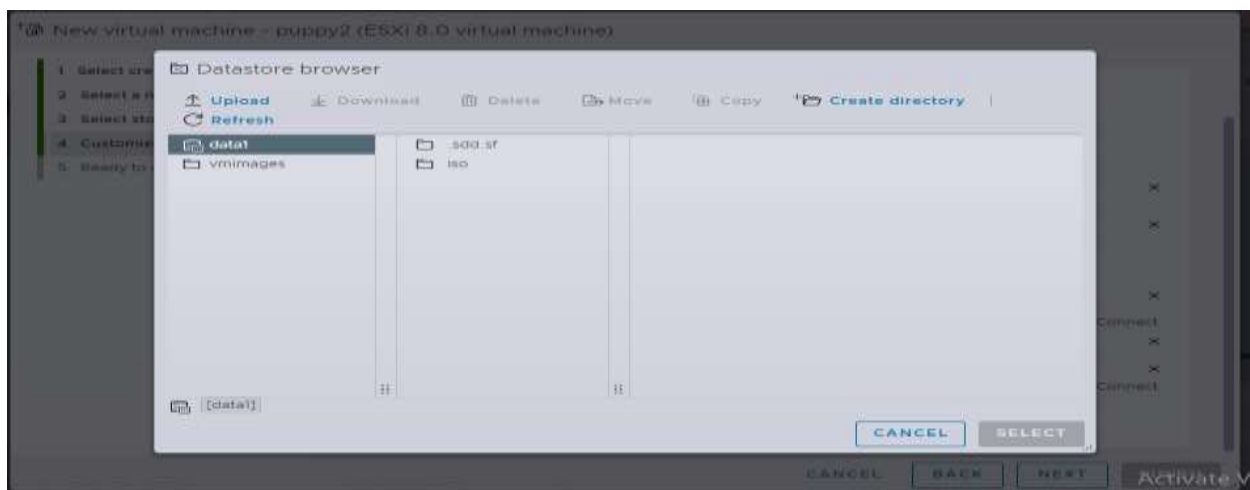
[Upload](#)
[Download](#)
[Delete](#)
[Move](#)
[Copy](#)
[Create directory](#)
[Refresh](#)

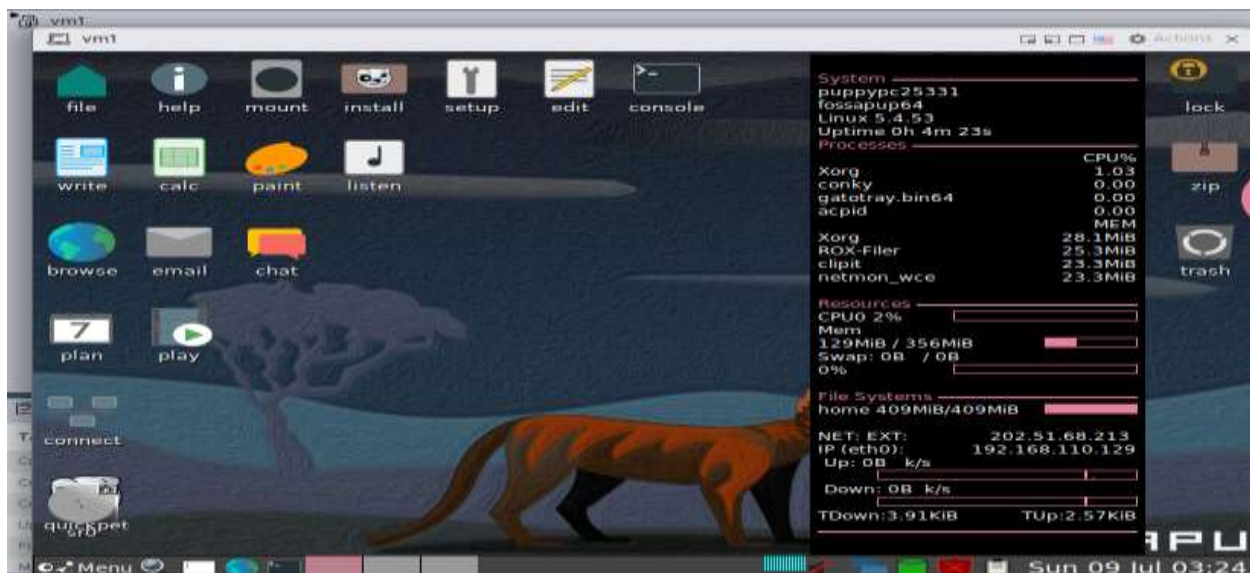
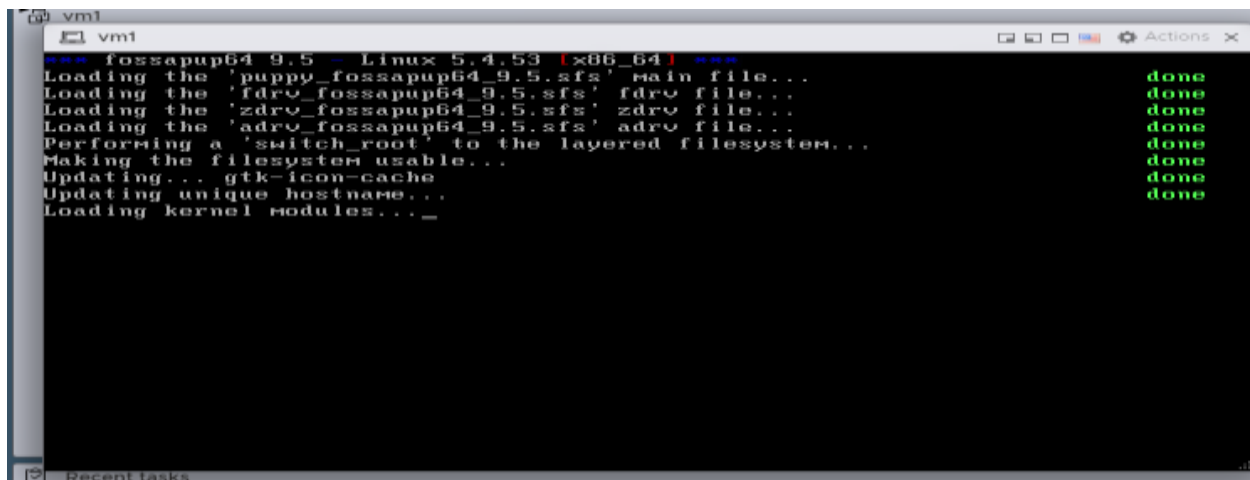
| data1 | data1 |
|-------|--------|
| | add of |
| | no |

[data1] 100%

CLOSE







Steps to install and configure ESXi

- a. Download ESXi installer from the official website of vmware.
- b. Create a new virtual machine in vmware pro.
- c. Select the downloaded ESXi iso file.
- d. Allocate the required resource as following.
- e. After resource allocation and completing setting install the ESXi.
2. Access ESXi host through browser
3. Adding a new hard disk drive in VMware.
4. Installing new Virtual Machine on ESXi host machine if error occurs do following steps:

Discussion:

While configuring ESXi for bare metal virtualization, firstly, download ESXi by obtaining the ESXi installation ISO file from the VMware website or the vendor's official portal. Install ESXi by inserting the bootable media into the server and boot from it. Follow the on-screen instructions to install ESXi on the bare metal server. You may need to configure network settings, storage options, and assign a root password during the installation process. Configure Networking after the installation, configure the networking settings in ESXi. This includes assigning IP addresses, configuring VLANs, setting up network adapters, and enabling services like DHCP, DNS, and routing if necessary. Set Up Storage by Configuring storage options in ESXi. This involves creating datastores, setting up storage adapters, and configuring RAID if applicable. You can use local storage, SAN (Storage Area Network), or NAS (Network-Attached Storage) depending on your requirements. Configure Security by Implementing security measures such as setting up firewall rules, enabling secure shell (SSH) access, and configuring user accounts with appropriate privileges. Regularly update ESXi with the latest patches and security updates. Set Up High Availability if needed, configure high availability features such as VMware vSphere High Availability (vSphere HA) to provide automated failover for virtual machines (VMs) in case of host failures. Installing Linux on top of ESXi, Once you have ESXi configured, you can install Linux as a virtual machine on top of it.

Prepare the Linux ISO, Download the Linux distribution ISO file from the official website. Create a New Virtual Machine by opening the VMware vSphere Client or vSphere Web Client and connect to your ESXi server. Create a new virtual machine by specifying the desired configuration such as CPU, memory, disk size, and network settings. Attach the Linux ISO file as the installation media. Install Linux by powering on the virtual machine and follow the installation process of the Linux distribution. This typically involves selecting the installation language, disk partitioning, setting up network configuration, and creating a user account. By following these steps, you can configure ESXi for bare metal virtualization and install Linux as a virtual machine on top of ESXi.

Conclusion:

Configuring ESXi for bare-metal virtualization and installing Linux on top of ESXi is a powerful and flexible approach to building virtualized environments. This setup allows you to leverage the benefits of both ESXi as the hypervisor and Linux as the guest operating system. ESXi is a robust bare-metal hypervisor developed by VMware. By installing ESXi directly on the hardware of a server, you create a dedicated virtualization layer that abstracts the underlying hardware and enables the creation and management of virtual machines (VMs). ESXi offers excellent performance, security, and management capabilities, making it a popular choice for virtualization. In this way, we can successfully configure ESXi for bare-metal virtualization and install Linux on top of ESXi. This setup provides a flexible platform for running multiple Linux VMs with efficient resource utilization, centralized management, and the ability to scale your virtualized infrastructure as needed.