**1. Create a gcc application program to swap two numbers using pointers in c.**

#include <stdio.h>

void swap(int \*x,int \*y)

{

int t;

t = \*x;

\*x = \*y;

\*y = t;

}

int main()

{

int num1,num2;

printf("Enter value of num1: ");

scanf("%d",&num1);

printf("Enter value of num2: ");

scanf("%d",&num2);

printf("Before Swapping: num1 is: %d, num2 is: %d\n",num1,num2);

swap(&num1,&num2);

printf("After Swapping: num1 is: %d, num2 is: %d\n",num1,num2);

return 0;

}

**1. Create a gcc application program to take input of name, rollno and marks obtained by a**

**Student in 4 subjects and display the name, rollno with percentage score secured**

#include<stdio.h>

int main()

{

char str[20]; int arr[10];

int rollno,i,sum=0; double percentage;

printf("Enter the name of the student :\n");

gets(str);

printf("Enter the roll number of the student :\n");

scanf("%d",&rollno);

printf("Enter the marks of the student : \n");

for(i=0;i<4;i++)

{

printf("subject %d marks : ",i+1);

scanf("%d",&arr[i]);

}

for(i=0;i<=4;i++)

{

sum=sum+arr[i] ;

}

percentage=(sum\*100)/400;

puts(str);

printf("Roll number : %d ",rollno);

printf("\nThe Percentage of marks obtained = %f ",percentage);

}

2. Create a web application(HTML) in cloud for Employee details, which includes EmpID,

Name, Designation, Salary, DOJ. etc.

1. Create a new html file (“index.html”) which comprises java script and css.
2. Host your files in GitHub:

* Create a new GitHub repository.
* Push both index.html and styles.css (if you created it) to the repository.

1. Access your application:

* Open your GitHub repository.
* Navigate to "Settings."
* Scroll down to the "GitHub Pages" section.
* Set the "Source" branch to main (or your preferred branch).

After a few moments, GitHub Pages will provide a link to your hosted web application. You can access it and share the link with others.

Please note that this is a basic example, and in a real-world scenario, you might want to consider additional features, server-side logic, and a more sophisticated data storage solution.

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Employee Details</title>

<style>

body {

font-family: 'Arial', sans-serif;

margin: 20px;

}

h1 {

color: #333;

}

form {

margin-bottom: 20px;

}

label {

display: block;

margin-bottom: 5px;

font-weight: bold;

}

input {

width: 100%;

padding: 8px;

margin-bottom: 10px;

box-sizing: border-box;

}

button {

padding: 10px;

background-color: #4CAF50;

color: white;

border: none;

cursor: pointer;

}

button:hover {

background-color: #45a049;

}

#employeeList {

margin-top: 20px;

}

/\* Additional styling for employee details \*/

div.employeeDetails {

border: 1px solid #ddd;

padding: 10px;

margin-bottom: 10px;

border-radius: 5px;

}

</style>

</head>

<body>

<h1>Employee Details</h1>

<form id="employeeForm">

<label for="empID">Employee ID:</label>

<input type="text" id="empID" name="empID" required><br>

<label for="name">Name:</label>

<input type="text" id="name" name="name" required><br>

<label for="designation">Designation:</label>

<input type="text" id="designation" name="designation" required><br>

<label for="salary">Salary:</label>

<input type="text" id="salary" name="salary" required><br>

<label for="doj">Date of Joining:</label>

<input type="date" id="doj" name="doj" required><br>

<button type="button" onclick="addEmployee()">Add Employee</button>

</form>

<div id="employeeList">

<!-- Employee details will be displayed here -->

</div>

<script>

function addEmployee() {

// Fetching values from the form

var empID = document.getElementById('empID').value;

var name = document.getElementById('name').value;

var designation = document.getElementById('designation').value;

var salary = document.getElementById('salary').value;

var doj = document.getElementById('doj').value;

// Creating a new div to display employee details

var employeeDiv = document.createElement('div');

// Adding a class for styling

employeeDiv.className = 'employeeDetails';

employeeDiv.innerHTML = '<strong>Employee ID:</strong> ' + empID + '<br>' +

'<strong>Name:</strong> ' + name + '<br>' +

'<strong>Designation:</strong> ' + designation + '<br>' +

'<strong>Salary:</strong> ' + salary + '<br>' +

'<strong>DOJ:</strong> ' + doj + '<br><br>';

// Appending the new div to the employee list

document.getElementById('employeeList').appendChild(employeeDiv);

// Clearing the form

document.getElementById('employeeForm').reset();

}

</script>

</body>

</html>

**2. Create a web application in cloud to embed a map along with hot spot, frames and links.**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Interactive Map</title>

<style>

/\* styles.css \*/

body {

margin: 0;

padding: 0;

}

#map-container {

position: relative;

}

#mapFrame {

width: 100%;

height: 100vh;

border: none;

}

.hotspot {

position: absolute;

width: 30px;

height: 30px;

background-color: red;

border-radius: 50%;

display: block;

cursor: pointer;

}

.hotspot:hover {

background-color: darkred;

}

</style>

</head>

<body>

<div id="map-container">

<iframe id="mapFrame" src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d387051.4305748679!2d-74.25908989999999!3d40.6976701!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x89c25903ce155677%3A0xb1ac25a77a1c3e20!2sNew%20York%2C%20NY%2C%20USA!5e0!3m2!1sen!2suk!4v1605211574394!5m2!1sen!2suk" allowfullscreen></iframe>

<a href="link1.html" class="hotspot" style="top: 30%; left: 40%;"></a>

<a href="link2.html" class="hotspot" style="top: 60%; left: 70%;"></a>

</div>

</body>

</html>

**3. Configure a Virtual Machine with 1 Virtual CPU with 1 GB RAM and 10GB Storage using VM**

**ware and then create CLONE of it and then re-launch the VM clone.**

Step 1: Create a Virtual Machine

1. Open VMware: Open VMware and navigate to the "Create a New Virtual Machine" option.
2. New Virtual Machine Wizard:
   * Choose "Typical" configuration for the virtual machine.
3. Guest Operating System:
   * Select "Linux" as the guest operating system and "Ubuntu" as the version.
4. Name and Location:
   * Specify a name for your virtual machine and choose a location to store it.
5. CPU Configuration:
   * Choose 1 virtual CPU.
6. Memory Configuration:
   * Allocate 1 GB RAM.
7. Network Configuration:
   * Choose the appropriate network settings based on your requirements.
8. Disk Configuration:
   * Create a new virtual disk with a size of 10GB.
9. Ready to Create Virtual Machine:
   * Review your settings and click "Finish" to create the virtual machine.

Step 2: Install Ubuntu on the Virtual Machine

1. Power On the Virtual Machine:
   * Power on the virtual machine and mount an Ubuntu ISO to install the operating system.
2. Install Ubuntu:
   * Follow the on-screen instructions to install Ubuntu on the virtual machine.

Step 3: Create a Clone

1. Power Off the Virtual Machine:
   * Ensure that the virtual machine is powered off.
2. Clone Virtual Machine:
   * Right-click on the virtual machine in the VMware interface.
   * Select "Clone" or "Clone to Template."
3. Clone Configuration:
   * Choose a name for the clone.
   * Specify the location for the clone.
   * Choose whether to create a linked clone or a full clone. For this case, you might want to create a full clone.
4. Customize the Clone:
   * If needed, you can customize the configuration of the clone (e.g., change network settings).
5. Finish:
   * Review your settings and click "Finish" to create the clone.

Step 4: Re-launch the VM Clone in Ubuntu

1. Power On the Clone:
   * Once the cloning process is complete, right-click on the cloned virtual machine.
   * Select "Power" and then "Power On."
2. Boot the Clone:
   * The cloned virtual machine is now ready to be powered on. Click on the "Power On" button to start the cloned VM.

**4. Setup a Single Node Hadoop cluster and show all the process through WEB UI.**

**Prerequisites**

**Supported Platforms**

* GNU/Linux is supported as a development and production platform. Hadoop has been demonstrated on GNU/Linux clusters with 2000 nodes.

**Required Software**

Required software for Linux include:

1. Java™ must be installed. Recommended Java versions are described at [HadoopJavaVersions](https://cwiki.apache.org/confluence/display/HADOOP/Hadoop+Java+Versions).
2. ssh must be installed and sshd must be running to use the Hadoop scripts that manage remote Hadoop daemons if the optional start and stop scripts are to be used. Additionally, it is recommmended that pdsh also be installed for better ssh resource management.

**Installing Software**

If your cluster doesn’t have the requisite software you will need to install it.

For example on Ubuntu Linux:

$ sudo apt-get install ssh

$ sudo apt-get install pdsh

**Download**

To get a Hadoop distribution, download a recent stable release from one of the [Apache Download Mirrors](http://www.apache.org/dyn/closer.cgi/hadoop/common/).

**Prepare to Start the Hadoop Cluster**

Unpack the downloaded Hadoop distribution. In the distribution, edit the file etc/hadoop/hadoop-env.sh to define some parameters as follows:

# set to the root of your Java installation

export JAVA\_HOME=/usr/java/latest

0

$ bin/hadoop

This will display the usage documentation for the hadoop script.

Now you are ready to start your Hadoop cluster in one of the three supported modes:

* [Local (Standalone) Mode](https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#Standalone_Operation)
* [Pseudo-Distributed Mode](https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#Pseudo-Distributed_Operation)
* [Fully-Distributed Mode](https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#Fully-Distributed_Operation)

**Standalone Operation**

By default, Hadoop is configured to run in a non-distributed mode, as a single Java process. This is useful for debugging.

The following example copies the unpacked conf directory to use as input and then finds and displays every match of the given regular expression. Output is written to the given output directory.

$ mkdir input

$ cp etc/hadoop/\*.xml input

$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar grep input output 'dfs[a-z.]+'

$ cat output/\*

**Pseudo-Distributed Operation**

Hadoop can also be run on a single-node in a pseudo-distributed mode where each Hadoop daemon runs in a separate Java process.

**Configuration**

Use the following:

etc/hadoop/core-site.xml:

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>

etc/hadoop/hdfs-site.xml:

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

</configuration>

**Setup passphraseless ssh**

Now check that you can ssh to the localhost without a passphrase:

$ ssh localhost

If you cannot ssh to localhost without a passphrase, execute the following commands:

$ ssh-keygen -t rsa -P '' -f ~/.ssh/id\_rsa

$ cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

$ chmod 0600 ~/.ssh/authorized\_keys

**Execution**

The following instructions are to run a MapReduce job locally. If you want to execute a job on YARN, see [YARN on Single Node](https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#YARN_on_a_Single_Node).

1. Format the filesystem:
2. $ bin/hdfs namenode -format
3. Start NameNode daemon and DataNode daemon:
4. $ sbin/start-dfs.sh

The hadoop daemon log output is written to the $HADOOP\_LOG\_DIR directory (defaults to $HADOOP\_HOME/logs).

1. Browse the web interface for the NameNode; by default it is available at:
   * NameNode - http://localhost:9870/
2. Make the HDFS directories required to execute MapReduce jobs:
3. $ bin/hdfs dfs -mkdir -p /user/<username>
4. Copy the input files into the distributed filesystem:
5. $ bin/hdfs dfs -mkdir input
6. $ bin/hdfs dfs -put etc/hadoop/\*.xml input
7. Run some of the examples provided:
8. $ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar grep input output 'dfs[a-z.]+'
9. Examine the output files: Copy the output files from the distributed filesystem to the local filesystem and examine them:
10. $ bin/hdfs dfs -get output output
11. $ cat output/\*

or

View the output files on the distributed filesystem:

$ bin/hdfs dfs -cat output/\*

1. When you’re done, stop the daemons with:
2. $ sbin/stop-dfs.sh

**YARN on a Single Node**

You can run a MapReduce job on YARN in a pseudo-distributed mode by setting a few parameters and running ResourceManager daemon and NodeManager daemon in addition.

The following instructions assume that 1. ~ 4. steps of [the above instructions](https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#Execution) are already executed.

1. Configure parameters as follows:

etc/hadoop/mapred-site.xml:

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

<property>

<name>mapreduce.application.classpath</name>

<value>$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/\*:$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/lib/\*</value>

</property>

</configuration>

etc/hadoop/yarn-site.xml:

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.env-whitelist</name>

<value>JAVA\_HOME,HADOOP\_COMMON\_HOME,HADOOP\_HDFS\_HOME,HADOOP\_CONF\_DIR,CLASSPATH\_PREPEND\_DISTCACHE,HADOOP\_YARN\_HOME,HADOOP\_HOME,PATH,LANG,TZ,HADOOP\_MAPRED\_HOME</value>

</property>

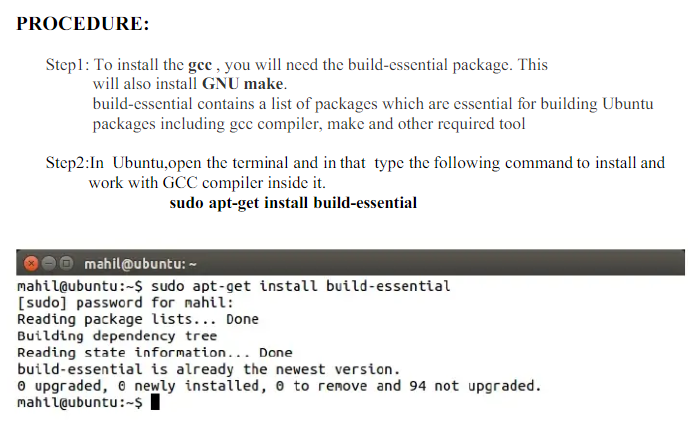
</configuration>

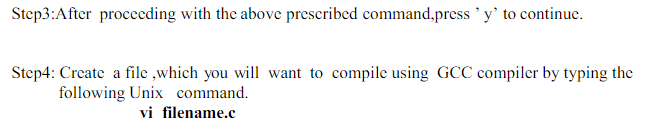
1. Start ResourceManager daemon and NodeManager daemon:
2. $ sbin/start-yarn.sh
3. Browse the web interface for the ResourceManager; by default it is available at:
   * ResourceManager - http://localhost:8088/
4. Run a MapReduce job.
5. When you’re done, stop the daemons with:
6. $ sbin/stop-yarn.sh

**Fully-Distributed Operation**

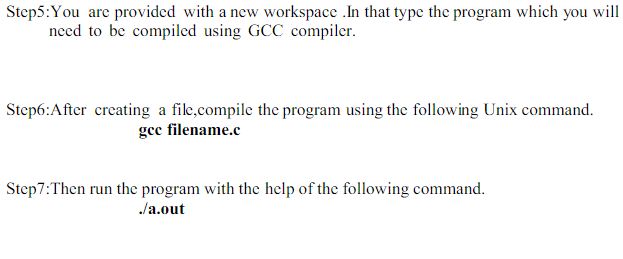
For information on setting up fully-distributed, non-trivial clusters see [Cluster Setup](https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html).

1. **Install a C compiler in the virtual machine created using virtual box and execute simple program.**

****

****

****

****

**6. Implement the procedure to create and mount a One-Node Hadoop Cluster using FUSE.**

**Step 1: Install Hadoop and FUSE-DFS**

Ensure that you have Hadoop installed on your machine. You can download Hadoop from the official Apache Hadoop website: <https://hadoop.apache.org/>.

Additionally, download and install FUSE-DFS, which is a Hadoop-compatible FUSE module for HDFS. You can find the FUSE-DFS source code on the Apache Hadoop website.

**Step 2: Configure Hadoop**

Update your Hadoop configuration files, typically located in the etc/hadoop/ directory. Key configurations to check or set in the core-site.xml and hdfs-site.xml files include:

**core-site.xml**

<**property>**

**<name>fs.defaultFS</name>**

**<value>hdfs://localhost:9000</value>**

**</property>**

**hdfs-site.xml**

**<property>**

**<name>dfs.replication</name>**

**<value>1</value>**

**</property>**

**Step 3: Build and Install FUSE-DFS**

Build the FUSE-DFS source code. Follow the instructions provided in the FUSE-DFS documentation for building and installing. This may involve running a build script or using a package manager.

**Step 4: Start Hadoop Services**

Start the Hadoop services for your one-node cluster. Use the following commands:

**sbin/start-dfs.sh**

**sbin/start-yarn.sh**

This will start the Hadoop Distributed File System (HDFS) and the YARN ResourceManager and NodeManager.

**Step 5: Mount HDFS using FUSE-DFS**

Use the FUSE-DFS tool to mount HDFS on your local machine. The command typically looks like this:

**fuse\_dfs <mount\_point> localhost 9000**

Replace <mount\_point> with the desired local mount point for HDFS. Ensure that Hadoop's Namenode is running on localhost at port 9000.

**Step 6: Interact with HDFS**

Once the HDFS is successfully mounted, you can interact with it like a regular file system. For example:

* List files in HDFS:

**ls <mount\_point>**

* Copy a file to HDFS:

**cp localfile.txt <mount\_point>/hdfsfile.txt**

* Read a file from HDFS:

**cat <mount\_point>/hdfsfile.txt**

**Step 7: Unmount HDFS**

When you're done, unmount the HDFS using the following command:

**fusermount -u <mount\_point>**

**Step 8: Verify the Setup**

Verify that you can read from and write to HDFS through the mounted file system. Check the Hadoop logs (typically in the logs/ directory of your Hadoop installation) and console outputs for any errors or warnings.

Keep in mind that these steps provide a general guideline, and the exact commands and paths may vary based on your Hadoop and FUSE-DFS versions. Always refer to the documentation of the tools and versions you are using for the most accurate instructions.

**6. Use version control systems command to clone, commit, push, fetch, pull, checkout, reset, and delete repositories.**

**Clone a Repository:**

**git clone <repository\_url>**

This command is used to clone a remote repository onto your local machine.

**Commit Changes:**

**git add <file\_name>**

**git commit -m "Commit message"**

These commands stage changes (using git add) and commit them to the local repository.

**Push Changes to Remote Repository**:

**git push origin <branch\_name>**

This command pushes committed changes from your local repository to the remote repository.

**Fetch Changes from Remote Repository**:

**git fetch origin**

This command fetches changes from the remote repository but does not merge them into your working directory.

**Pull Changes from Remote Repository**:

**git pull origin <branch\_name>**

This command fetches changes from the remote repository and merges them into your working directory.

**Checkout a Branch:**

**git checkout <branch\_name>**

This command is used to switch to a different branch in your local repository.

**Reset Changes**:

**git reset <file\_name>**

This command unstages changes, effectively "un-adding" the specified file.

**Delete** a Repository:

# Locally

**rm -rf <repository\_directory>**

# Remotely (if you have the necessary permissions)

**git remote rm origin**

The local command removes the local repository directory, and the remote command removes the remote association.

**8. Implement the procedure to interact with Hadoop API for Accessing HDFS from local file**

**system. (Note: use available Hadoop installed Clone).**

**Step 1: Set Up Hadoop Environment**

Make sure you have Hadoop installed on your system, and the necessary environment variables are set. Also, ensure that your Hadoop cluster is running.

**Step 2: Create a Java Project**

Create a new Java project in your preferred integrated development environment (IDE) or use a simple text editor. In your project, include the Hadoop libraries in the classpath.

**Step 3: Write Java Code**

Create a Java class with the following code. Replace the HDFS and local file paths accordingly.

**Step 4: Replace File Paths**

Replace the placeholder file paths (**/path/in/hdfs/file.txt** and **/path/on/local/file.txt**) with the actual paths you want to use.

**Step 5: Compile and Run**

Compile the Java program and run it. Ensure that the Hadoop libraries are in your classpath.

**Step 6: Verify Results**

Check HDFS and the local file system to verify that the file interactions were successful.

**PROGRAM:**

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

public class HDFSInteraction {

public static void main(String[] args) {

// Step 1: Set Hadoop configuration

Configuration conf = new Configuration();

// Step 2: Set the HDFS URI

conf.set("fs.defaultFS", "hdfs://localhost:9000");

try {

// Step 3: Create a FileSystem object

FileSystem fs = FileSystem.get(conf);

// Step 4: Specify the HDFS file path

Path hdfsFilePath = new Path("/path/in/hdfs/file.txt");

// Step 5: Specify the local file path

Path localFilePath = new Path("/path/on/local/file.txt");

// Step 6: Upload a local file to HDFS

uploadToLocalFileSystem(fs, localFilePath, hdfsFilePath);

// Step 7: Download a file from HDFS to local

downloadFromLocalFileSystem(fs, hdfsFilePath, localFilePath);

// Step 8: Close the FileSystem object

fs.close();

} catch (IOException e) {

e.printStackTrace();

}

}

private static void uploadToLocalFileSystem(FileSystem fs, Path localFilePath, Path hdfsFilePath) throws IOException {

try (InputStream in = fs.open(localFilePath);

OutputStream out = fs.create(hdfsFilePath)) {

byte[] buffer = new byte[4096];

int bytesRead;

while ((bytesRead = in.read(buffer)) > 0) {

out.write(buffer, 0, bytesRead);

}

}

}

private static void downloadFromLocalFileSystem(FileSystem fs, Path hdfsFilePath, Path localFilePath) throws IOException {

try (InputStream in = fs.open(hdfsFilePath);

OutputStream out = fs.create(localFilePath)) {

byte[] buffer = new byte[4096];

int bytesRead;

while ((bytesRead = in.read(buffer)) > 0) {

out.write(buffer, 0, bytesRead);

}

}

}

}

**9.Install VMware with different flavours of Ubuntu on top of Windows 8.**

**Lab Manual: Installing VMware and Ubuntu on Windows 8**

**Table of Contents**

1. [Introduction](https://chat.openai.com/c/af34bcfb-0f9b-4f5d-8ece-79c94d6065e3#introduction)
2. [Prerequisites](https://chat.openai.com/c/af34bcfb-0f9b-4f5d-8ece-79c94d6065e3#prerequisites)
3. [Install VMware Workstation](https://chat.openai.com/c/af34bcfb-0f9b-4f5d-8ece-79c94d6065e3#install-vmware-workstation)
4. [Download Ubuntu ISO](https://chat.openai.com/c/af34bcfb-0f9b-4f5d-8ece-79c94d6065e3#download-ubuntu-iso)
5. [Create a Virtual Machine for Ubuntu Desktop](https://chat.openai.com/c/af34bcfb-0f9b-4f5d-8ece-79c94d6065e3#create-a-virtual-machine-for-ubuntu-desktop)
6. [Install Ubuntu Desktop](https://chat.openai.com/c/af34bcfb-0f9b-4f5d-8ece-79c94d6065e3#install-ubuntu-desktop)
7. [Create a Virtual Machine for Ubuntu Server](https://chat.openai.com/c/af34bcfb-0f9b-4f5d-8ece-79c94d6065e3#create-a-virtual-machine-for-ubuntu-server)
8. [Install Ubuntu Server](https://chat.openai.com/c/af34bcfb-0f9b-4f5d-8ece-79c94d6065e3#install-ubuntu-server)
9. [Install VMware Tools](https://chat.openai.com/c/af34bcfb-0f9b-4f5d-8ece-79c94d6065e3#install-vmware-tools)
10. [Conclusion](https://chat.openai.com/c/af34bcfb-0f9b-4f5d-8ece-79c94d6065e3#conclusion)

**Introduction**

This lab manual provides step-by-step instructions for installing VMware Workstation and setting up different flavors of Ubuntu on a Windows 8 system.

**Prerequisites**

Before starting this lab, ensure that you have the following:

* A Windows 8 system with administrative privileges.
* Sufficient disk space for VMware and virtual machines.
* Internet access for downloading software.

**Install VMware Workstation**

1. Download the latest version of VMware Workstation for Windows from the official website: [VMware Workstation](https://www.vmware.com/products/workstation-pro.html).
2. Run the installer and follow the on-screen instructions to install VMware Workstation.
3. Launch VMware Workstation after installation.

**Download Ubuntu ISO**

1. Visit the official Ubuntu website: [Download Ubuntu](https://ubuntu.com/download).
2. Choose the desired flavor of Ubuntu (Desktop or Server) and download the corresponding ISO file.

**Create a Virtual Machine for Ubuntu Desktop**

1. Open VMware Workstation.
2. Click on "File" -> "New Virtual Machine."
3. Choose "Typical" and click "Next."
4. Select "Installer disc image file (iso)" and browse to the location where you saved the Ubuntu Desktop ISO.
5. Click "Next" and follow the prompts to finish creating the virtual machine.

**Install Ubuntu Desktop**

1. Start the virtual machine.
2. Follow the on-screen instructions to install Ubuntu Desktop.
3. Select language, timezone, keyboard layout, and create a user account.

**Create a Virtual Machine for Ubuntu Server**

1. Open VMware Workstation.
2. Click on "File" -> "New Virtual Machine."
3. Choose "Custom (advanced)" and click "Next."
4. Select "Installer disc image file (iso)" and browse to the location where you saved the Ubuntu Server ISO.
5. Click "Next" and follow the prompts to customize the virtual machine settings.

**Install Ubuntu Server**

1. Start the virtual machine.
2. Follow the on-screen instructions to install Ubuntu Server.
3. Select language, timezone, keyboard layout, and create a user account.

**Install VMware Tools**

1. While the virtual machine is running, go to "Player" -> "Manage" -> "Install VMware Tools."
2. Follow the prompts to install VMware Tools within the Ubuntu virtual machine.

**9. Install Virtualbox Work workstation with different flavours of linux on top of Windows 8**

**Prerequisites:**

1. Windows 8 installed and running.
2. Sufficient free disk space for virtual machines.
3. Internet connectivity to download VirtualBox and Linux ISOs.

**Lab Steps:**

Step 1: Download and Install VirtualBox

1. Go to <https://www.virtualbox.org/>.
2. Navigate to the "Downloads" section.
3. Download the Windows Host version of VirtualBox.
4. Run the installer and follow the on-screen instructions to install VirtualBox.

Step 2: Download Linux ISO Images

1. **Ubuntu:**
   * Go to <https://ubuntu.com/>.
   * Download the Ubuntu Desktop ISO.
2. **CentOS:**
   * Go to <https://www.centos.org/>.
   * Download the CentOS Minimal ISO.

Step 3: Create an Ubuntu Virtual Machine

1. Open VirtualBox.
2. Click on "New" to create a new virtual machine.
3. Name it (e.g., UbuntuVM), set Type to Linux, and Version to Ubuntu.
4. Assign at least 1 GB of RAM.
5. Create a virtual hard disk.
6. Select "Create a virtual hard disk now" and choose VDI.
7. Choose dynamically allocated and set the size (e.g., 20 GB).
8. Click "Create."

Step 4: Install Ubuntu on the Virtual Machine

1. Right-click on the Ubuntu VM and select "Settings."
2. Under "System," move the "Hard Disk" to the top in the "Boot Order."
3. Under "Storage," select the empty disk under "Controller: IDE" and choose the Ubuntu ISO in the "Optical Drive."
4. Click "OK" and start the VM.
5. Follow the Ubuntu installation steps.

Step 5: Create a CentOS Virtual Machine

1. Repeat steps 3-7 from the Ubuntu VM creation.
2. Under "Storage," select the empty disk under "Controller: IDE" and choose the CentOS ISO in the "Optical Drive."
3. Click "OK" and start the VM.
4. Follow the CentOS installation steps.

Step 6: Complete Linux Installations

Follow the on-screen instructions to complete the installations for both Ubuntu and CentOS.

Step 7: Test Virtual Machines

1. Start the Ubuntu VM and log in.
2. Start the CentOS VM and log in.

**11. Setup a Private Cloud by performing the procedure using a Single node OPENSTACK**

**implementation.**

**Installation of Openstack**

1. add new user named stack – This stack user is the adminstrator of the openstack services.

To add new user – run the command as root user.

**adduser stack**

2. run the command

**apt-get install sudo -y || install -y sudo**

3. Be careful in running the command – please be careful with the syntax. If any error in thsi

following command, the system will crash beacause of permission errors.

**echo “stack ALL=(ALL) NOPASSWD:ALL” >> /etc/sudoers**

4. Logout the system and login as stack user

5. Run the command (this installs git repo package)

Run in Root

**export http\_proxy=http://172.16.0.3:8080**

**sudo apt-get install git**

6. Run the command (This clones updatesd version of dev-stack (which is binary auto-installer

**package of Openstack)**

**stack@JBL01:/$ export http\_proxy=http://172.16.0.3:8080**

**stack@JBL01:/$ export https\_proxy=http://172.16.0.3:8080**

**stack@JBL01:/$ git config --global http.proxy $http\_proxy**

**stack@JBL01:/$ git config --global https.proxy $http\_proxy**

**git clone http://git.openstack.org/openstack-dev/devstack**

**ls** (this shows a folder named devstack)

**cd devstack** (enter into the folder)

7. create a file called local.conf. To do this run the command,

**nano local.conf**

stack@JBL01:/devstack$ sudo nano local.conf

8. In the file, make the following entry (Contact Your Network Adminstrator for doubts in these values)

**[[local|localrc]]**

**FLOATING\_RANGE=192.168.1.224/27**

**FIXED\_RANGE=10.11.11.0/24**

**FIXED\_NETWORK\_SIZE=256**

**FLAT\_INTERFACE=eth0**

**ADMIN\_PASSWORD=root**

**DATABASE\_PASSWORD=root**

**RABBIT\_PASSWORD=root**

**SERVICE\_PASSWORD=root**

**SERVICE\_TOCKEN=root**

9. Save this file

**stack@JBL01:/devstack$ sudo gedit stackrc**

Save this file

Change File Permission:

**stack@JBL01:~$ chown stack \* -R**

10. Run the command (This installs Opentack)

**./stack.sh**

11. If any error occurs, then run the command for uninistallation

**./unstack.sh**

1. update the packages

**apt-get update**

2. Then reinstall the package

**./stack.sh**

12. Open the browser, http://IP address of your machine, you will get the openstack portal.

13. If you restart the machine, then to again start open stack

open terminal,

**su stack**

**cd devstack**

**run ./rejoin.sh**

14. Again you can access openstack services in the browser, http://IP address of your machine,

**12. Simulate a cloud scenario using CloudSim and run any one of the scheduling algorithm that is not present in CloudSim.**