

A FILE OF CLOUD COMPUTING LAB

At

BABA BANDA SINGH BAHADUR ENGINEERING COLLEGE

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE

AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(Computer Science & Engineering)



SUBMITTED BY:

PRINCE KUMAR (2001308)

SUBMITTED TO:

PROF. MANDEEP KAUR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**BABA BANDA SINGH BAHADUR ENGINEERING COLLEGE,
FATEHGARH SAHIB**

Table of contents

Sr. no.	content	Page no.
1.	Install VirtualBox/VMware Workstation on different OS.	3-4
2.	Install different operating systems in Virtual Box.	4-6
3.	Simulate a cloud scenario using simulator.	7-19
4.	Implement scheduling algorithms.	20-48
5.	To Study Cloud Security management.	49-57
6.	To study and implementation of identity management.	58-62
7.	Case Study - Amazon Web Services/Microsoft Azure/Google cloud services.	63-68

PRACTICAL NO. 1

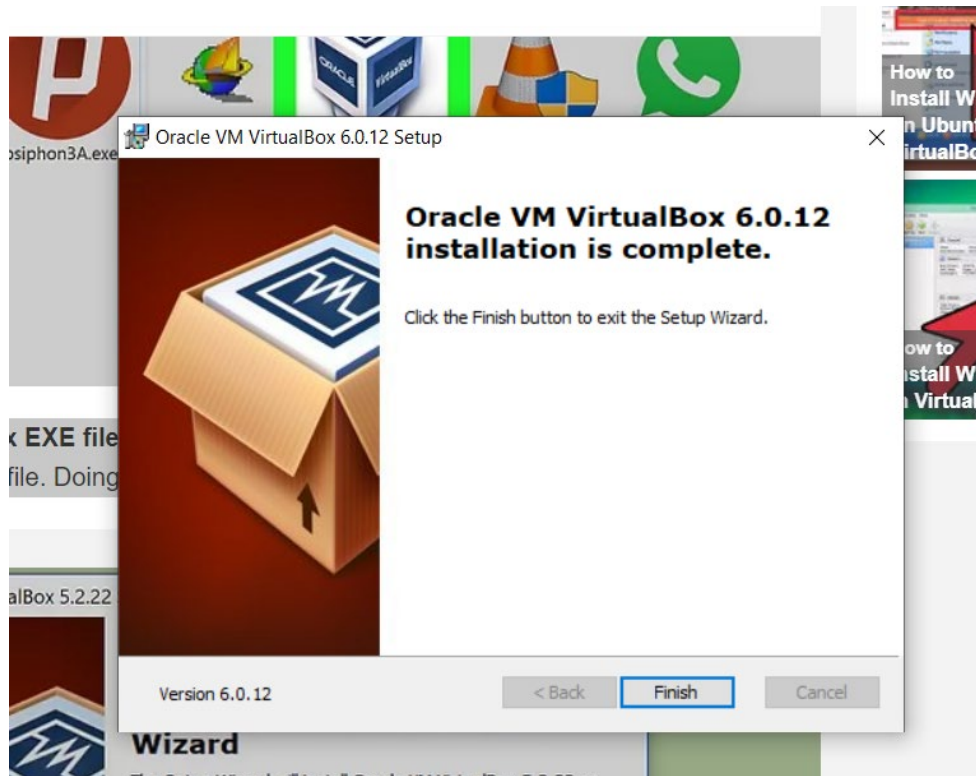
Aim: Install VirtualBox/VMware Workstation on different OS.

Step 1:

Download VirtualBox installer for Windows The installer can be found on its download page here <https://www.virtualbox.org/wiki/Downloads>

Step 2:

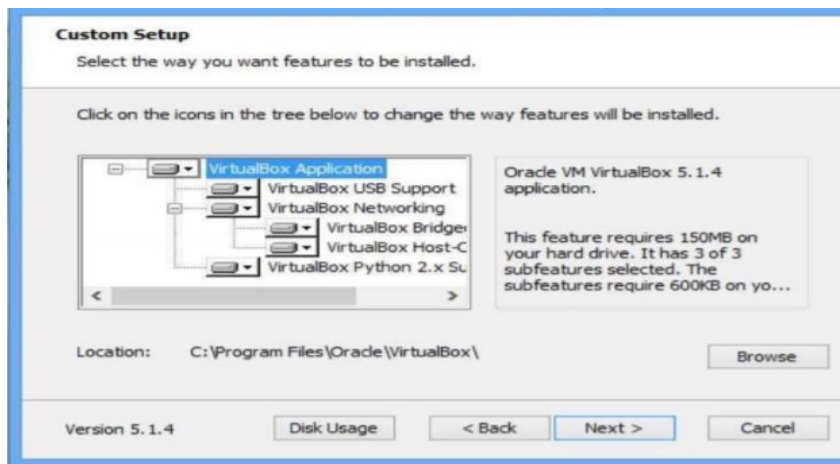
Open the VirtualBox EXE file. Go to the location to which the EXE file downloaded and double-click the file. Doing so will open the VirtualBox installation window.



Step 3:

Navigate through the installation prompts. Do the following:

- Click **Next** on the first three pages.
- Click **Yes** when prompted.
- Click **Install**
- Click **Yes** when prompted.



Step 4:

Click Finish when prompted. It's in the lower-right side of the window. Doing so will close the installation window and open VirtualBox. Now that you've installed and opened VirtualBox, you can [create a virtual machine](#) in order to run any operating system on your PC



PRACTICAL NO. 2

Aim: Install different operating systems in Virtual Box.

Installing kali linux in virtual box

Requirements:

- At least **20 GB of disk space**
- At least **1 GB of RAM** (preferably 2) for i386 and amd64 architectures
- VirtualBox (or alternative virtualization software)

Step 1:

Download Kali Linux ISO Image

On the official Kali Linux website downloads section, you can find Kali Linux .iso images. These images are uploaded every few months, providing the latest official releases.

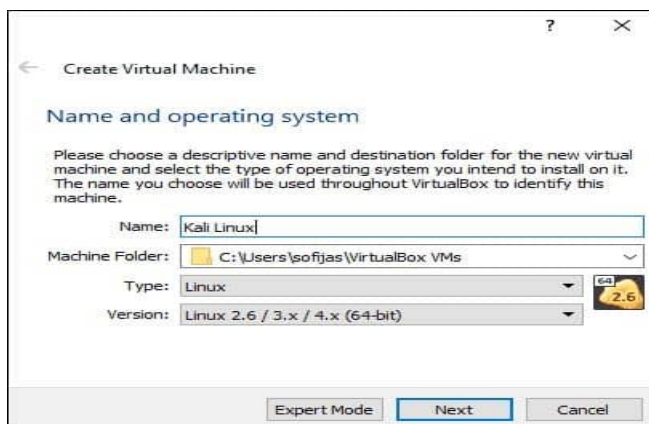
Navigate to the Kali Linux Downloads page and find the packages available for download. Depending on the system you have, download the 64-Bit or 32-Bit version.

Step 2:

Create Kali Linux VirtualBox Container

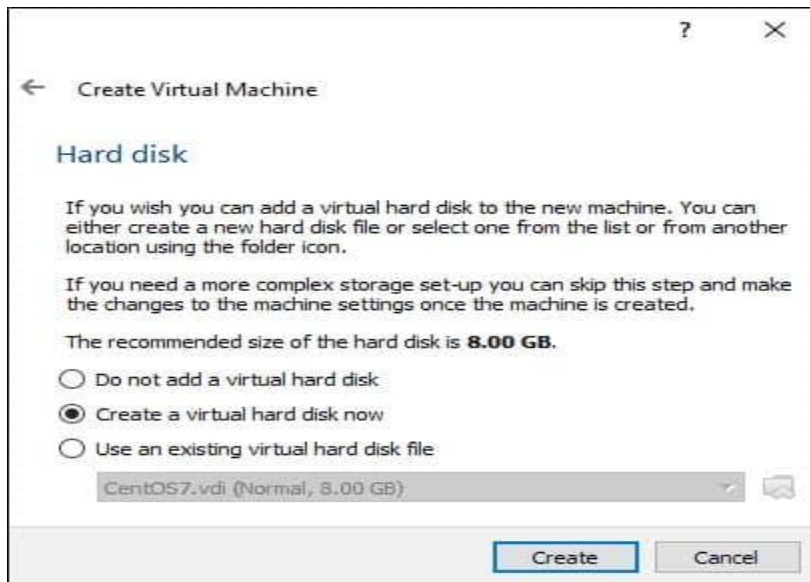
After downloading the .iso image, create a new virtual machine and import Kali as its OS.

1. Launch VirtualBox Manager and click the **New** icon.
2. **Name and operating system.** A pop-up window for creating a new VM appears. Specify a **name** and a **destination folder**. The *Type* and *Version* change automatically, based on the name you provide. **Make sure the information matches the package you downloaded** and click **Next**.



3. **Memory size.** Choose how much **memory** to allocate to the virtual machine and click **Next**. The default setting for Linux is **1024 MB**. However, this varies depending on your individual needs.

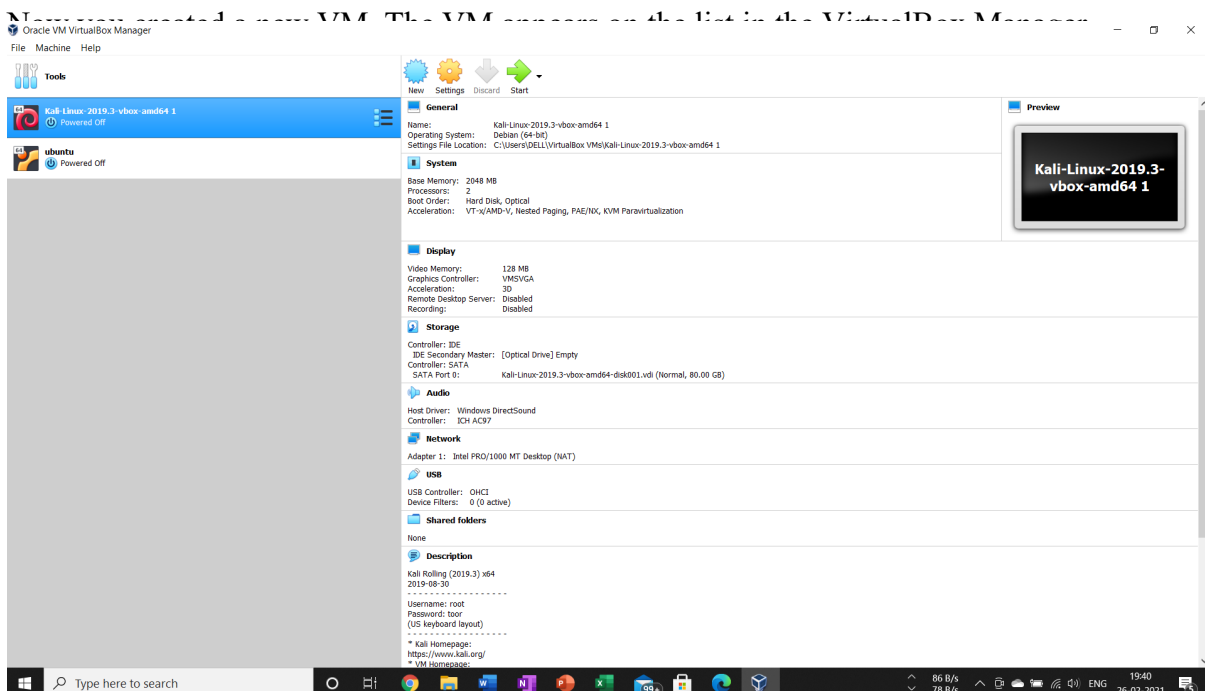
4. **Hard disk.** The default option is to **create a virtual hard disk** for the new VM. Click **Create** to continue. Alternatively, you can use an existing virtual hard disk file or decide not to add one at all.



5. **Hard disk file type.** Stick to the default file type for the new virtual hard disk, **VDI (VirtualBox Disk Image)**. Click **Next** to continue.

6. **Storage on a physical hard disk.** Decide between **Dynamically allocated** and **Fixed size**. The first choice allows the new hard disk to grow and fill up space dedicated to it. The second, fixed size, uses the maximum capacity from the start. Click **Next**.

7. **File location and size.** Specify the name and where you want to store the virtual hard disk. Choose the amount of file data the VM is allowed to store on the hard disk. We advise giving it at least **8 GB**. Click **Create** to finish.



PRACTICAL NO. 3

Aim: Simulate a cloud scenario using simulator.

STEP BY STEP INSTALLATION OF CLOUD SIM INTO ECLIPSE

Before you start to setup CloudSim, following resources must be Installed/downloaded on the local system

Java Development Kit(JDK): As the CloudSim simulation toolkit is a class library written in the Java programming language, therefore, the latest version of Java(JDK) should be installed on your machine, which can be downloaded from <https://www.oracle.com/java/technologies/javase-jdk16-downloads.html>

Eclipse IDE for Java developers: As per your current installed operating system (Linux/Windows). Before you download to make sure to check if 32-bit or 64-bit version is applicable to your Computer machine. <https://www.eclipse.org/downloads/packages/release/kepler/sr1/eclipse-ide-java-developers>

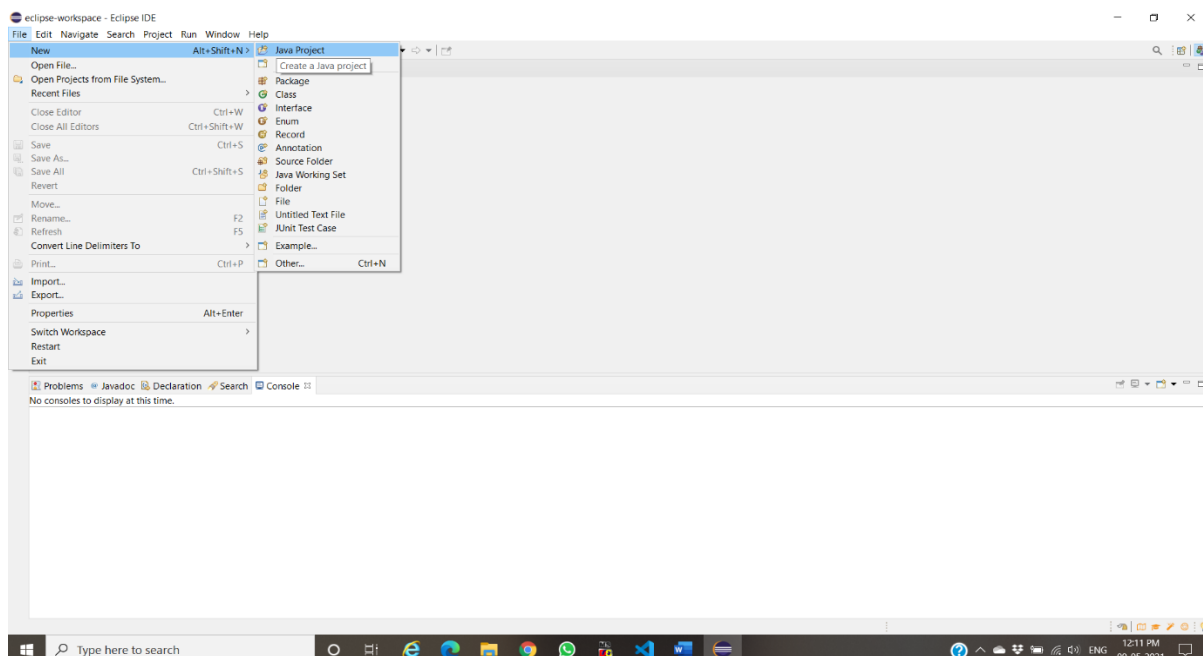
Download CloudSim source code: To date, various versions of CloudSim are released the latest version is 5.0, which is based on a container-based engine. Whereas to **keep the setup simple for beginners we will be setting up the most used version i.e. 3.0.3**, which can be directly downloaded by clicking on link <https://github.com/Cloudslab/cloudsim/releases>

One external requirement of CloudSim i.e. common jar package of math-related functions is to be downloaded from the [Apache website](#) or you may directly download by clicking [here](#).

Unzip Eclipse, CloudSim and Common Math libraries to some common folder.

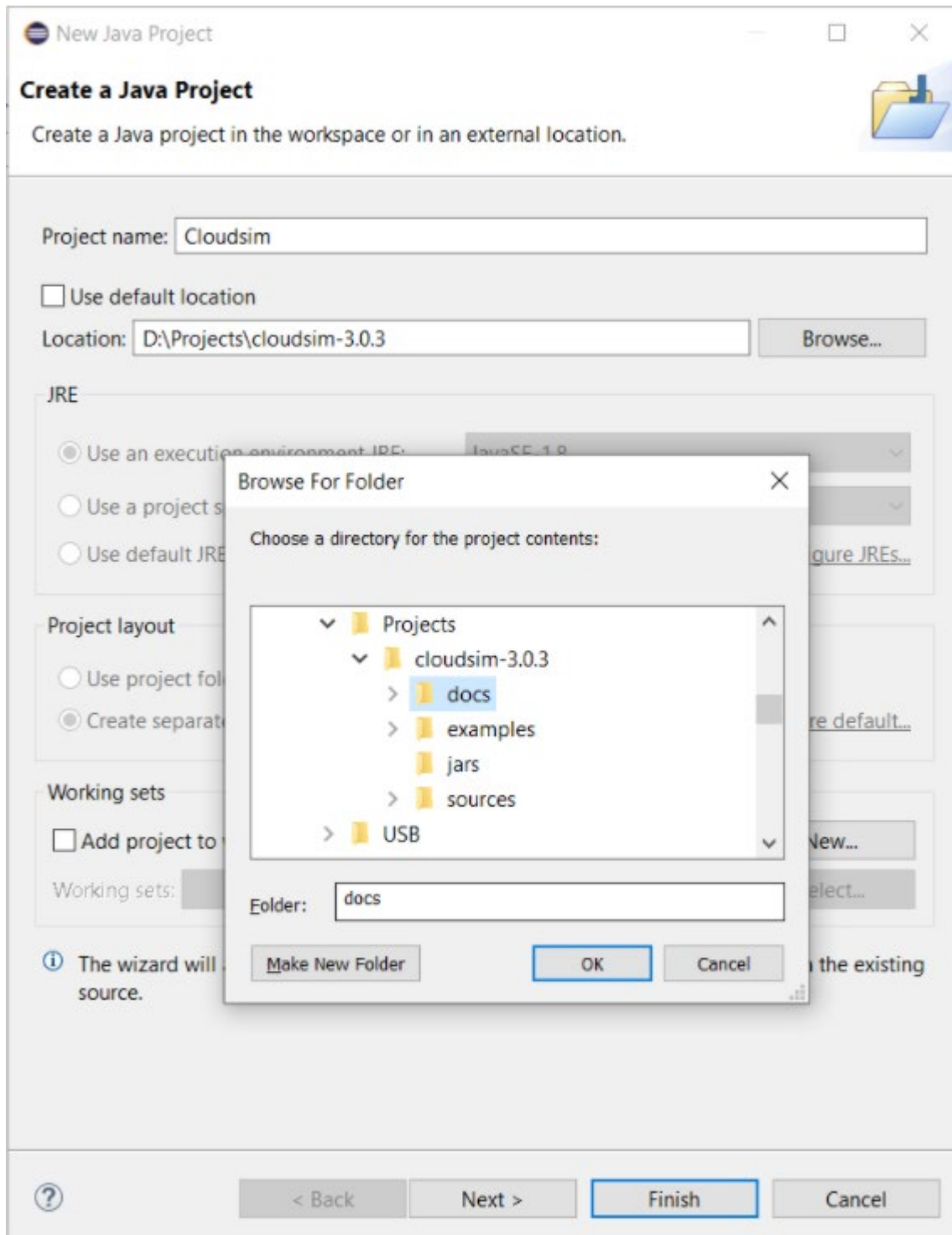
Step 1: First of all, navigate to the folder where you have unzipped the eclipse folder and open Eclipse.exe

Step 2: Now within Eclipse window navigate the menu: *File -> New ->Java Project*, to open the new project wizard

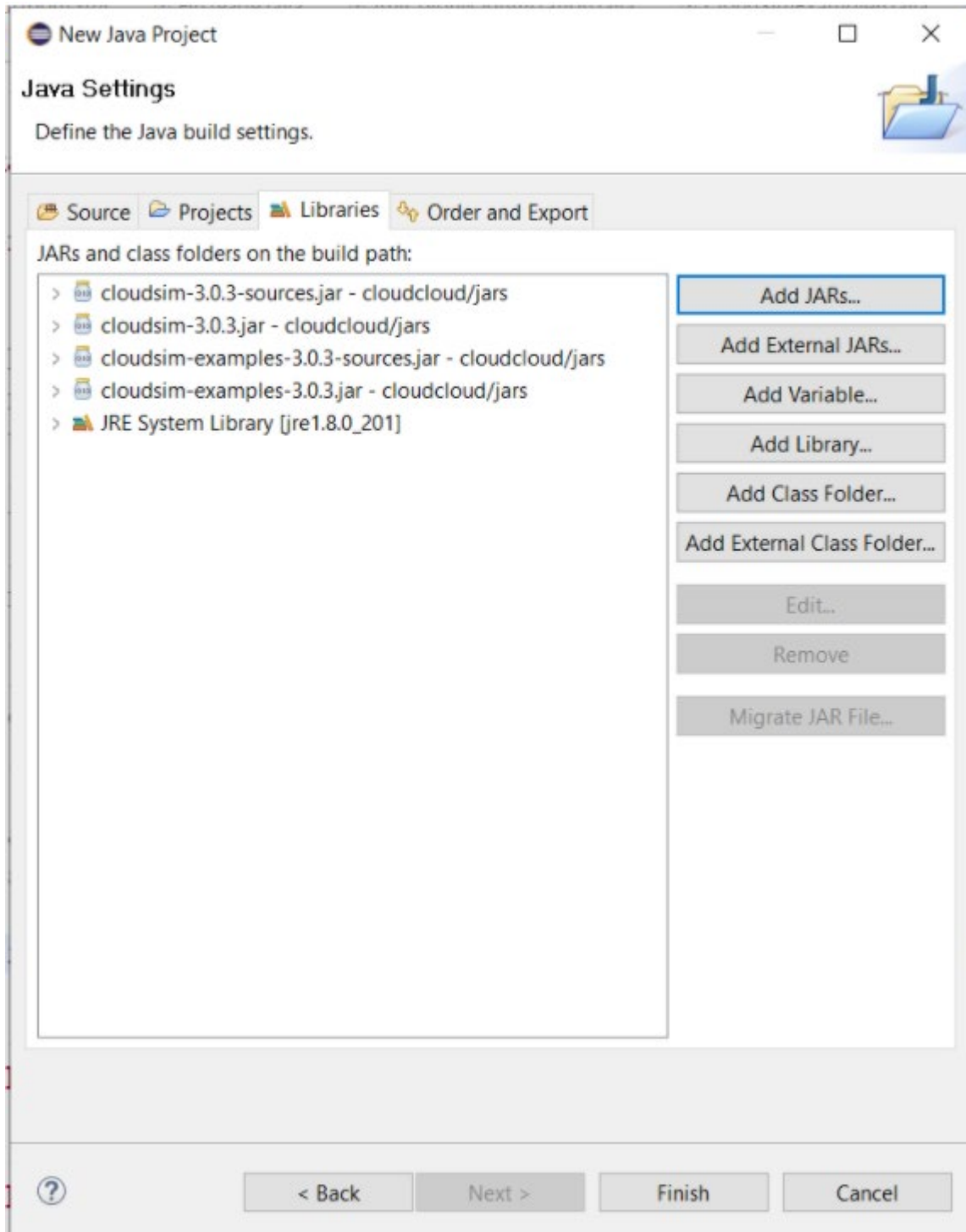


Step 3: Now a detailed new project window will open, here you will provide the project name and the path of CloudSim project source code, which will be done as follows:

- Project Name: CloudSim.
- Unselect the '**Use default location**' option and then click on '**Browse**' to open the path where you have unzipped the CloudSim project and finally click Next to set project settings
- Once done finally, click 'Next' to go to the next step i.e. setting up of project settings



- Now open '**Libraries**' tab and if you do not find commons-math3-3.x.jar (*here 'x' means the minor version release of the library which could be 2 or greater*) in the list then simply click on '**Add External Jar**' (commons-math3-3.x.jar will be included in the project from this step)

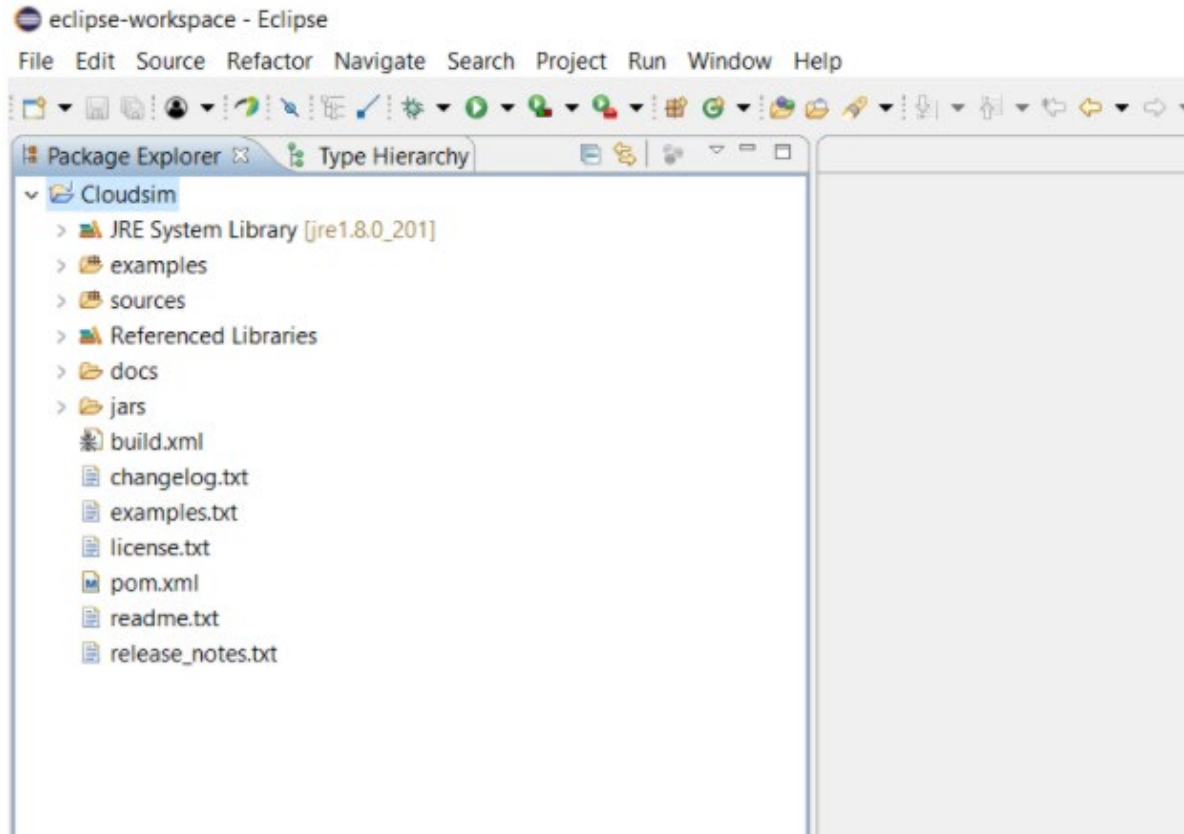


- Once you have clicked on '**Add External JAR's**' Open the path where you have unzipped the commons-math binaries and select '**Commons-math3-3.x.jar**' and click on open.

- Ensure external jar that you opened in the previous step is displayed in the list and then click on '**Finish**' (your system may take 2-3 minutes to configure the project)

Step 4: Once the project is configured you can open the **Project Explorer** and start exploring the CloudSim project. Also, for the first-time eclipse automatically start building the workspace for newly configured CloudSim project, which may take some time depending on the configuration of the computer system.

Following is the final screen which you will see after CloudSim is configured.



SIMULATING A CLOUD SCENARIO SHOWING HOW TO CREATE A DATACENTER WITH ONE HOST AND RUN ONE CLOUDLET ON IT.

CloudSim Components

1. **Data Centre:** Represents Complete hardware. It has set of hosts (physical machine)
2. **Datacenter Broker:** This class represents a broker acting on behalf of a user. It modifies two mechanisms: a mechanism for submitting VM provisioning requests to data centers and another one for submitting tasks to VMs. The CloudSim users have to extend this class for conducting experiments with their own policies.

3. **Hosts:** It is a Physical Machine. It has variables to represent memory, processors, id, scheduling scheme etc.
4. **Virtual Machine (Vm):** It models a virtual machine. One host can initiate multiple virtual machines and allocate cores based on predefined processor sharing policies (Space Shared, Time Shared).
5. **Cloudlets:** A cloudlet class is also known as a task. It Models the Cloud-based application services which are commonly deployed in the data centers. Every application has a pre assigned instruction length.
6. **Cloudlet Scheduler:** Determines how the available CPU resources of virtual machine are divided among Cloudlets.

Source Code

```
package org.cloudbus.cloudsim.examples;

import java.text.DecimalFormat;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.LinkedList;

import java.util.List;

import org.cloudbus.cloudsim.Cloudlet;

import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;

import org.cloudbus.cloudsim.Datacenter;

import org.cloudbus.cloudsim.DatacenterBroker;

import org.cloudbus.cloudsim.DatacenterCharacteristics;

import org.cloudbus.cloudsim.Host;

import org.cloudbus.cloudsim.Log;

import org.cloudbus.cloudsim.Pe;

import org.cloudbus.cloudsim.Storage;

import org.cloudbus.cloudsim.UtilizationModel;
```

```

import org.cloudbus.cloudsim.UtilizationModelFull;

import org.cloudbus.cloudsim.Vm;

import org.cloudbus.cloudsim.VmAllocationPolicySimple;

import org.cloudbus.cloudsim.VmSchedulerTimeShared;

import org.cloudbus.cloudsim.core.CloudSim;

import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;

import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;

import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

/**
 * A simple example showing how to create a datacenter with one host and run one
 * cloudlet on it.
 */

public class CloudSimExample1 {

    /** The cloudlet list. */

    private static List<Cloudlet> cloudletList;

    /** The vm list. */

    private static List<Vm> vmList;

    /**
     * Creates main() to run this example.
     *
     * @param args the args
     */

    @SuppressWarnings("unused")

    public static void main(String[] args) {

        Log.println("Starting CloudSimExample1...");

```

```

try {

    // First step: Initialize the CloudSim package. It should be called
    // before creating any entities.

    int num_user = 1; // number of cloud users

    Calendar calendar = Calendar.getInstance();

    boolean trace_flag = false; // mean trace events

    // Initialize the CloudSim library

    CloudSim.init(num_user, calendar, trace_flag);

    // Second step: Create Datacenters

    // Datacenters are the resource providers in CloudSim. We need at
    // list one of them to run a CloudSim simulation

    Datacenter datacenter0 = createDatacenter("Datacenter_0");

    // Third step: Create Broker

    DatacenterBroker broker = createBroker();

    int brokerId = broker.getId();

    // Fourth step: Create one virtual machine

    vmlist = new ArrayList<Vm>();

    // VM description

    int vmid = 0;

    int mips = 1000;

    long size = 10000; // image size (MB)

    int ram = 512; // vm memory (MB)

    long bw = 1000;

    int pesNumber = 1; // number of cpus

    String vmm = "Xen"; // VMM name

```

```

// create VM

Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm,
new CloudletSchedulerTimeShared());

// add the VM to the vmList

vmList.add(vm);

// submit vm list to the broker

broker.submitVmList(vmList);

// Fifth step: Create one Cloudlet

cloudletList = new ArrayList<Cloudlet>();

// Cloudlet properties

int id = 0;

long length = 400000;

long fileSize = 300;

long outputSize = 300;

UtilizationModel utilizationModel = new UtilizationModelFull();

Cloudlet cloudlet = new Cloudlet(id, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);

cloudlet.setUserId(brokerId);

cloudlet.setVmId(vmid);

// add the cloudlet to the list

cloudletList.add(cloudlet);

// submit cloudlet list to the broker

broker.submitCloudletList(cloudletList);

// Sixth step: Starts the simulation

CloudSim.startSimulation();

```

```

        CloudSim.stopSimulation();

        //Final step: Print results when simulation is over

        List<Cloudlet> newList = broker.getCloudletReceivedList();

        printCloudletList(newList);

        Log.println("CloudSimExample1 finished!");

    } catch (Exception e) {

        e.printStackTrace();

        Log.println("Unwanted errors happen");

    }

}

/**
 * Creates the datacenter.
 *
 * @param name the name
 *
 * @return the datacenter
 */
private static Datacenter createDatacenter(String name) {

    // Here are the steps needed to create a PowerDatacenter:

    // 1. We need to create a list to store

    // our machine

    List<Host> hostList = new ArrayList<Host>();

    // 2. A Machine contains one or more PEs or CPUs/Cores.

    // In this example, it will have only one core.

    List<Pe> peList = new ArrayList<Pe>();

    int mips = 1000;

```

```

// 3. Create PEs and add these into a list.

peList.add(new Pe(0, new PeProvisionerSimple(mips)));

// need to store Pe id and MIPS Rating

// 4. Create Host with its id and list of PEs and add them to the list

// of machines

int hostId = 0;

int ram = 2048; // host memory (MB)

long storage = 1000000; // host storage

int bw = 10000;

hostList.add(

    new Host(

        hostId,

        new RamProvisionerSimple(ram),

        new BwProvisionerSimple(bw),

        storage,

        peList,

        new VmSchedulerTimeShared(peList)

    )

); // This is our machine

// 5. Create a DatacenterCharacteristics object that stores the

// properties of a data center: architecture, OS, list of

// Machines, allocation policy: time- or space-shared, time zone

// and its price (G$/Pe time unit).

String arch = "x86"; // system architecture

String os = "Linux"; // operating system

```



```

String vmm = "Xen";

double time_zone = 10.0; // time zone this resource located

double cost = 3.0; // the cost of using processing in this resource

double costPerMem = 0.05; // the cost of using memory in this resource

double costPerStorage = 0.001; // the cost of using storage in this
                                // resource

double costPerBw = 0.0; // the cost of using bw in this resource

LinkedList<Storage> storageList = new LinkedList<Storage>(); // we are not
adding SAN

// devices by now

DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
    arch, os, vmm, hostList, time_zone, cost, costPerMem,
    costPerStorage, costPerBw);

// 6. Finally, we need to create a PowerDatacenter object.

Datacenter datacenter = null;

try {
    datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
} catch (Exception e) {
    e.printStackTrace();
}

return datacenter;
}

// We strongly encourage users to develop their own broker policies, to

// submit vms and cloudlets according

```

```

// to the specific rules of the simulated scenario

/**
 * Creates the broker.
 *
 * @return the datacenter broker
 */

private static DatacenterBroker createBroker() {

    DatacenterBroker broker = null;

    try {

        broker = new DatacenterBroker("Broker");

    } catch (Exception e) {

        e.printStackTrace();

        return null;

    }

    return broker;

}

* Prints the Cloudlet objects.

* @param list list of Cloudlets

private static void printCloudletList(List<Cloudlet> list) {

    int size = list.size();

    Cloudlet cloudlet;

    String indent = "  ";

    Log.println();

    Log.println("===== OUTPUT =====");

    Log.println("Cloudlet ID" + indent + "STATUS" + indent

```

```

        + "Data center ID" + indent + "VM ID" + indent + "Time" + indent
        + "Start Time" + indent + "Finish Time");

DecimalFormat dft = new DecimalFormat("###.##");

for (int i = 0; i < size; i++) {

    cloudlet = list.get(i);

    Log.print(indent + cloudlet.getCloudletId() + indent + indent);

    if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS) {

        Log.print("SUCCESS");

        Log.println(indent + indent + cloudlet.getResourceId()

            + indent + indent + indent + cloudlet.getVmId()

            + indent + indent

            + dft.format(cloudlet.getActualCPUTime()) + indent

            + indent + dft.format(cloudlet.getExecStartTime())

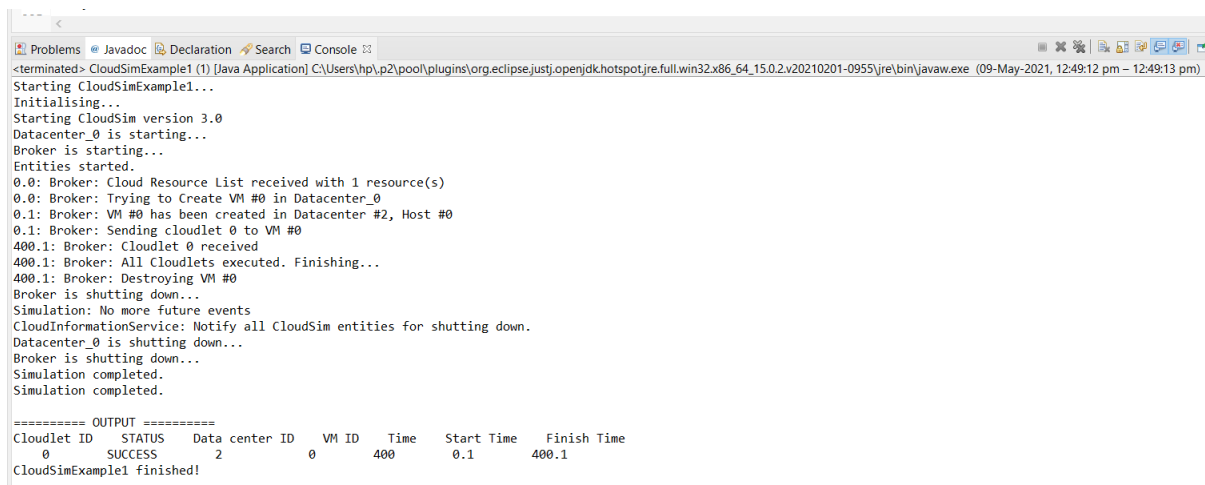
            + indent + indent

            + dft.format(cloudlet.getFinishTime()));

    }    } }

```

Output:



```

<terminated> CloudSimExample1 (1) [Java Application] C:\Users\hp\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\javaw.exe (09-May-2021, 12:49:12 pm - 12:49:13 pm)
Starting CloudSimExample1...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
400.1: Broker: Cloudlet 0 received
400.1: Broker: All Cloudlets executed. Finishing...
400.1: Broker: Destroying VM #0
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

===== OUTPUT =====
Cloudlet ID   STATUS   Data center ID   VM ID   Time   Start Time   Finish Time
0            SUCCESS   2              0       400    0.1          400.1
CloudSimExample1 finished!

```

PRACTICAL NO. 4

Aim: Implement scheduling algorithms.

Implementation of Shortest Job First algorithm using

cloudSim DatacenterBroker.java package

```
org.cloudbus.cloudsim;
```

```
import java.util.ArrayList; import
```

```
java.util.HashMap; import
```

```
java.util.LinkedList; import
```

```
java.util.List; import
```

```
java.util.Map;
```

```
import org.cloudbus.cloudsim.core.CloudSim; import
```

```
org.cloudbus.cloudsim.core.CloudSimTags; import
```

```
org.cloudbus.cloudsim.core.SimEntity; import
```

```
org.cloudbus.cloudsim.core.SimEvent; import
```

```
org.cloudbus.cloudsim.lists.CloudletList; import
```

```
org.cloudbus.cloudsim.lists.VmList;
```

```
/**
```

```
* DatacentreBroker represents a broker acting on behalf of a user. It hides VM management, as
```

```
vm
```

```
* creation, submission of cloudlets to this VMs and destruction of VMs.
```

```
* @author Rodrigo N. Calheiros
```

```
* @author Anton Beloglazov
```

```
* @since CloudSim Toolkit 1.0
```

```
public class DatacenterBroker extends SimEntity {
```

```
/** The vm list. */ protected List<? Extends
```

```

Vm> vmList; /** The vms created list. */

protected List<? extends Vm> vmsCreatedList;

/** The cloudlet list. */ protected List<? extends

Cloudlet> cloudletList;

/** The cloudlet submitted list. */ protected List<? extends

Cloudlet> cloudletSubmittedList; /**

The cloudlet received list. */

protected List<? extends Cloudlet> cloudletReceivedList;

/** The cloudlets submitted. */ protected int

cloudletsSubmitted; /** The vms requested. */

protected int vmsRequested; /** The vms

acks. */ protected int vmsAcks; /** The vms

destroyed. */ protected int vmsDestroyed; /** The

datacenter ids list. */ protected List<Integer>

datacenterIdsList; /** The datacenter requested ids list.

*/ protected List<Integer> datacenterRequestedIdsList;

/** The vms to datacenters map. */ protected

Map<Integer, Integer> vmsToDatacentersMap;

/** The datacenter characteristics list. */ protected Map<Integer,

DatacenterCharacteristics> datacenterCharacteristicsList;

* Created a new DatacenterBroker object.

* @param name name to be associated with this entity (as required by Sim_entity class

from

* simjava package)

* @throws Exception the exception

```

⌞

```

* @pre name != null

* @post $none */

public DatacenterBroker(String name) throws Exception {

    super(name);

    setVmList(new ArrayList<Vm>());

    setVmsCreatedList(new ArrayList<Vm>());

    setCloudletList(new ArrayList<Cloudlet>());

    setCloudletSubmittedList(new

    ArrayList<Cloudlet>());

    setCloudletReceivedList(new ArrayList<Cloudlet>());

    cloudletsSubmitted = 0; setVmsRequested(0);

    setVmsAcks(0); setVmsDestroyed(0);

    setDatacenterIdsList(new LinkedList<Integer>());

    setDatacenterRequestedIdsList(new ArrayList<Integer>());

    setVmsToDatacentersMap(new HashMap<Integer, Integer>());

    setDatacenterCharacteristicsList(new HashMap<Integer,

    DatacenterCharacteristics>()); }

/**

* This method is used to send to the broker the list with virtual machines that must be * created.

* @param list the list

* @pre list !=null

* @post $none

*/ public void submitVmList(List<? extends

Vm> list) { getVmList().addAll(list);

}

<<

```

* This method is used to send to the broker the list of cloudlets.

* @param list the list

* @pre list !=null

* @post \$none

```
public void submitCloudletList(List<? extends Cloudlet> list) {  
    getCloudletList().addAll(list); }
```

* Specifies that a given cloudlet must run in a specific virtual machine.

* @param cloudletId ID of the cloudlet being bount to a vm

* @param vmId the vm id

* @pre cloudletId > 0

* @pre id > 0

* @post \$none

```
public void bindCloudletToVm(int cloudletId, int vmId) {  
    CloudletList.getById(getCloudletList(), cloudletId).setVmId(vmId);  
}  
/**
```

* Processes events available for this Broker.

* @param ev a SimEvent object

* @pre ev != null

* @post \$none

@Override

```
public void processEvent(SimEvent ev) { switch  
(ev.getTag()) {  
    // Resource characteristics request  
    case  
    <>
```

CloudSimTags.RESOURCE_CHARACTERISTICS_REQUEST:

```
processResourceCharacteristicsRequest(ev);
```

```
break;
```

```
// Resource characteristics answer case
```

CloudSimTags.RESOURCE_CHARACTERISTICS:

```
processResourceCharacteristics(ev);
```

```
break;
```

```
// VM Creation answer case
```

CloudSimTags.VM_CREATE_ACK:

```
processVmCreate(ev);
```

```
break;
```

```
// A finished cloudlet returned case
```

CloudSimTags.CLOUDLET_RETURN:

```
processCloudletReturn(ev);
```

```
break;
```

```
// if the simulation finishes case
```

CloudSimTags.END_OF_SIMULATION:

```
shutdownEntity();
```

```
break;
```

```
// other unknown tags are processed by this method
```

```
default: processOtherEvent(ev);
```

```
break;
```

```
} }
```

* Process the return of a request for the characteristics of a PowerDatacenter.

* @param ev a SimEvent object

↵


```

* @pre ev != $null

* @post $none

protected void processResourceCharacteristics(SimEvent ev) {

    DatacenterCharacteristics characteristics = (DatacenterCharacteristics) ev.getData();

    getDatacenterCharacteristicsList().put(characteristics.getId(), characteristics);

    if (getDatacenterCharacteristicsList().size() ==

    getDatacenterIdsList().size()) {

        setDatacenterRequestedIdsList(new ArrayList<Integer>());

        createVmsInDatacenter(getDatacenterIdsList().get(0));

    } }

* Process a request for the characteristics of a PowerDatacenter.

* @param ev a SimEvent object

* @pre ev != $null

* @post $none

*/ protected void

processResourceCharacteristicsRequest(SimEvent ev) {

    setDatacenterIdsList(CloudSim.getCloudResourceList()); setDatacenterCharacteristicsList(new

    HashMap<Integer,

    DatacenterCharacteristics>());

    Log.println(CloudSim.clock() + ": " + getName() + ": Cloud Resource

    List received with " + getDatacenterIdsList().size() + "

    resource(s)"); for (Integer datacenterId :

    getDatacenterIdsList()) {

        sendNow(datacenterId,

        CloudSimTags.RESOURCE_CHARACTERISTICS, getId()); } }

↵

```

```

* Process the ack received due to a request for VM creation.

* @param ev a SimEvent object

* @pre ev != null

* @post $none

*/ protected void

processVmCreate(SimEvent ev) { int[] data =

(int[]) ev.getData(); int datacenterId = data[0];

int vmId = data[1];

int result = data[2];

if (result == CloudSimTags.TRUE) { getVmsToDatacentersMap().put(vmId, datacenterId);

getVmsCreatedList().add(VmList.getById(getVmList(), vmId));

Log.println(CloudSim.clock() + ": " + getName() + ": VM #" + vmId

+ ", Host #"

+ " has been created in Datacenter #" + datacenterId

vmId).getHost().getId());

+ VmList.getById(getVmsCreatedList(),

} else {

Log.println(CloudSim.clock() + ": " + getName() + ": Creation of VM #" + vmId

+ " failed in Datacenter #" + datacenterId); }

incrementVmsAcks();

// all the requested VMs have been created if

(getVmsCreatedList().size() == getVmList().size() - getVmsDestroyed())

{ submitCloudlets(); }

else { // all the acks received, but some VMs were not created if (getVmsRequested() ==

getVmsAcks())

}

}

```

```

{

// find id of the next datacenter that has not been tried

for (int nextDatacenterId : getDatacenterIdsList()) {

if

(!getDatacenterRequestedIdsList().contains(nextDatacenterId)) {

createVmsInDatacenter(nextDatacenterId);

return; } }

// all datacenters already queried if (getVmsCreatedList().size() > 0) { // if some vm
were created

submitCloudlets();

} else { // no vms created. abort

Log.println(CloudSim.clock() + ": " + getName() + ": none of the required VMs could
be created. Aborting");

finishExecution(); } } } }

* Process a cloudlet return event.

* @param ev a SimEvent object

* @pre ev != $null

* @post $none

protected void processCloudletReturn(SimEvent ev) { Cloudlet

cloudlet = (Cloudlet) ev.getData();

getCloudletReceivedList().add(cloudlet);

Log.println(CloudSim.clock() + ": " + getName() +

": Cloudlet " + cloudlet.getCloudletId()

+ " received"); cloudletsSubmitted--;

}

```

```

if (getCloudletList().size() == 0 && cloudletsSubmitted == 0) { // all cloudlets executed

Log.println(CloudSim.clock() + ": " + getName() + ": All
Cloudlets executed. Finishing...");

clearDatacenters();

finishExecution(); } else { // some cloudlets haven't finished
yet if

(getCloudletList().size() > 0 && cloudletsSubmitted == 0) {

// all the cloudlets sent finished. It means that some bount

// cloudlet is waiting its VM be created

clearDatacenters();

createVmsInDatacenter(0);

} } }

* Overrides this method when making a new and different type of Broker. This method is called
* by {@link #body()} for incoming unknown tags.

@param ev a SimEvent object

* @pre ev != null

* @post $none

protected void processOtherEvent(SimEvent ev) { if (ev == null) {

Log.println(getName() + ".processOtherEvent(): " + "Error - an event is null.");

return;

}

Log.println(getName() + ".processOtherEvent(): "

+ "Error - event unknown by this DatacenterBroker.");

}

/*Create the virtual machines in a datacenter.

<0

```

* @param datacenterId Id of the chosen PowerDatacenter

* @pre \$none

* @post \$none

```
protected void createVmsInDatacenter(int datacenterId) {
```

```
// send as much vms as possible for this datacenter before trying the next one
```

```
int requestedVms = 0;
```

```
String datacenterName = CloudSim.getEntityName(datacenterId);
```

```
for (Vm vm : getVmList()) {
```

```
if (!getVmsToDatacentersMap().containsKey(vm.getId())) {
```

```
Log.println(CloudSim.clock() + ": " + getName() + ":
```

```
Trying to Create VM #" + vm.getId()
```

```
+ " in " + datacenterName); sendNow(datacenterId,
```

```
CloudSimTags.VM_CREATE_ACK, vm);
```

```
requestedVms++;
```

```
}}}
```

```
getDatacenterRequestedIdsList().add(datacenterId);
```

```
setVmsRequested(requestedVms); setVmsAcks(0);
```

```
}
```

```
/**
```

* Submit cloudlets to the created VMs.

* @pre \$none

* @post \$none

```
protected void submitCloudlets() {
```

```
int vmIndex = 0;
```

```
List <Cloudlet> sortList= new ArrayList<Cloudlet>(); ArrayList<Cloudlet> tempList = new
```

```
    </pre>
```

```

ArrayList<Cloudlet>()); for(Cloudlet cloudlet: getCloudletList())

{

tempList.add(cloudlet);

}

int totalCloudlets= tempList.size(); for(int

i=0;i<totalCloudlets;i++)

{

Cloudlet smallestCloudlet= tempList.get(0); for(Cloudlet checkCloudlet:

tempList)

{ if(smallestCloudlet.getCloudletLength()>checkCloudlet.getCloudletLength())

{

smallestCloudlet= checkCloudlet;

} }

sortList.add(smallestCloudlet);

tempList.remove(smallestCloudlet);

}

int count=1;

for(Cloudlet printCloudlet: sortList)

{

Log.println(count+".Cloudler

Id:"+printCloudlet.getCloudletId()+" ,Cloudlet

Length:"+printCloudlet.getCloudletLength()); count++;}

for (Cloudlet cloudlet : sortList) {

Vm vm;

// if user didn't bind this cloudlet and it has not been executed yet

}

```

```

if (cloudlet.getVmId() == -1) {

    vm = getVmsCreatedList().get(vmIndex);

    } else { // submit to the specific vm

    vm = VmList.getById(getVmsCreatedList(), cloudlet.getVmId());

    if (vm == null) { // vm was not created

        Log.println(CloudSim.clock() + ": " + getName()

+ ": Postponing execution of cloudlet "

+ cloudlet.getCloudletId() + ": bount

VM not available");

        continue;

    } }

    Log.println(CloudSim.clock() + ": " + getName() + ": Sending cloudlet "

+ cloudlet.getCloudletId() + " to VM #" + vm.getId());

    cloudlet.setVmId(vm.getId());

    sendNow(getVmsToDatacentersMap().get(vm.getId()),

CloudSimTags.CLOUDLET_SUBMIT, cloudlet);

    cloudletsSubmitted++;

    vmIndex = (vmIndex + 1) % getVmsCreatedList().size();

    getCloudletSubmittedList().add(cloudlet);

    }

    // remove submitted cloudlets from waiting list for (Cloudlet cloudlet :

getCloudletSubmittedList()) {

    getCloudletList().remove(cloudlet);

    } }

    * Destroy the virtual machines running in datacenters.

```

```

* @pre $none

* @post $none

protected void clearDatacenters() { for
(Vm vm : getVmsCreatedList()) {

    Log.println(CloudSim.clock() + ": " + getName() + ":
Destroying VM #" + vm.getId());

    sendNow(getVmsToDatacentersMap().get(vm.getId()),
CloudSimTags.VM_DESTROY, vm);

}

getVmsCreatedList().clear();

}

* Send an internal event communicating the end of the simulation.

* @pre $none

* @post $none

protected void finishExecution() { sendNow(getId(),
CloudSimTags.END_OF_SIMULATION);

}

* (non-Javadoc)

* @see cloudsim.core.SimEntity#shutdownEntity()

@Override public void
shutdownEntity() {

    Log.println(getName() + " is shutting down...");

}

* (non-Javadoc)

* @see cloudsim.core.SimEntity#startEntity()

```



```

@Override

public void startEntity() {

    Log.println(getName() + " is starting...");

    schedule(getId(), 0,

CloudSimTags.RESOURCE_CHARACTERISTICS_REQUEST);

}

/** * Gets the

vm list.

* @param <T> the generic type

* @return the vm list

@SuppressWarnings("unchecked") public <T extends Vm>

List<T> getVmList() {

return (List<T>) vmList;

}

* Sets the vm list.

* @param <T> the generic type

* @param vmList the new vm list

protected <T extends Vm> void setVmList(List<T> vmList) { this.vmList = vmList;

}

* Gets the cloudlet list.

* @param <T> the generic type

* @return the cloudlet list

@SuppressWarnings("unchecked") public <T

extends Cloudlet> List<T> getCloudletList() {

return (List<T>) cloudletList; }

}

```

* Sets the cloudlet list.

* @param <T> the generic type

* @param cloudletList the new cloudlet list

```
protected <T extends Cloudlet> void setCloudletList(List<T> cloudletList) {  
    this.cloudletList = cloudletList; }
```

* Gets the cloudlet submitted list.

* @param <T> the generic type

* @return the cloudlet submitted list

```
@SuppressWarnings("unchecked") public <T extends  
Cloudlet> List<T> getCloudletSubmittedList() { return  
(List<T>) cloudletSubmittedList;  
}
```

* Sets the cloudlet submitted list.

* @param <T> the generic type

* @param cloudletSubmittedList the new cloudlet submitted list

*/ protected <T extends Cloudlet>

void

```
setCloudletSubmittedList(List<T> cloudletSubmittedList) {  
    this.cloudletSubmittedList = cloudletSubmittedList;  
}
```

* Gets the cloudlet received list.

@param <T> the generic type

* @return the cloudlet received list

```
@SuppressWarnings("unchecked") public <T extends  
Cloudlet> List<T> getCloudletReceivedList() { return
```

```

(List<T>) cloudletReceivedList;

}

* Sets the cloudlet received list. *

* @param <T> the generic type

* @param cloudletReceivedList the new cloudlet received list

*/ protected <T extends Cloudlet>

void setCloudletReceivedList(List<T>

cloudletReceivedList) {

    this.cloudletReceivedList = cloudletReceivedList;

}

* Gets the vm list.

* @param <T> the generic type

* @return the vm list

@SuppressWarnings("unchecked") public <T extends Vm> List<T> getVmsCreatedList()

{ return (List<T>) vmsCreatedList;

}

* Sets the vm list.

* @param <T> the generic type

* @param vmsCreatedList the vms created list

*/ protected <T extends Vm> void setVmsCreatedList(List<T>

vmsCreatedList) { this.vmsCreatedList = vmsCreatedList;

}

* Gets the vms requested.

* @return the vms requested

protected int getVmsRequested() { return vmsRequested;

}

```

```
}
```

* Sets the vms requested.

* @param vmsRequested the new vms requested

```
protected void setVmsRequested(int vmsRequested) { this.vmsRequested  
= vmsRequested;
```

```
}
```

* Gets the vms acks.

* @return the vms acks

```
protected int getVmsAcks() { return vmsAcks;  
}
```

* Sets the vms acks.

* @param vmsAcks the new vms acks

```
protected void setVmsAcks(int vmsAcks) { this.vmsAcks = vmsAcks;  
}
```

* Increment vms acks.

```
*/ protected void incrementVmsAcks()  
{ vmsAcks++;  
}
```

* Gets the vms destroyed.

@return the vms destroyed

```
protected int getVmsDestroyed() { return vmsDestroyed;  
}
```

* Sets the vms destroyed.

* @param vmsDestroyed the new vms destroyed

```
*/ protected void setVmsDestroyed(int
```

```
vv
```

```

vmsDestroyed) { this.vmsDestroyed = vmsDestroyed;

}

* Gets the datacenter ids list.

* @return the datacenter ids list

protected List<Integer> getDatacenterIdsList() { return datacenterIdsList;

}

* Sets the datacenter ids list.

* @param datacenterIdsList the new datacenter ids list

/* protected void setDatacenterIdsList(List<Integer>
datacenterIdsList) { this.datacenterIdsList = datacenterIdsList;

}

* Gets the vms to datacenters map.

* @return the vms to datacenters map

/* protected Map<Integer, Integer> getVmsToDatacentersMap() {

return vmsToDatacentersMap;

}

/** * Sets the vms to datacenters map.

* @param vmsToDatacentersMap the vms to datacenters map

/* protected void setVmsToDatacentersMap(Map<Integer, Integer>
vmsToDatacentersMap) { this.vmsToDatacentersMap =
vmsToDatacentersMap;

}

* Gets the datacenter characteristics list.

* @return the datacenter characteristics list

protected Map<Integer, DatacenterCharacteristics>

```

```

getDatacenterCharacteristicsList() {

return datacenterCharacteristicsList;

}

* Sets the datacenter characteristics list.

* @param datacenterCharacteristicsList the datacenter characteristics list

protected void setDatacenterCharacteristicsList(

Map<Integer, DatacenterCharacteristics> datacenterCharacteristicsList) {

this.datacenterCharacteristicsList = datacenterCharacteristicsList;

}

* Gets the datacenter requested ids list.

* @return the datacenter requested ids list

protected List<Integer> getDatacenterRequestedIdsList() {

return datacenterRequestedIdsList;

}

* Sets the datacenter requested ids list.

* @param datacenterRequestedIdsList the new datacenter requested ids list

*/ protected void

setDatacenterRequestedIdsList(List<Integer> datacenterRequestedIdsList)

{ this.datacenterRequestedIdsList

= datacenterRequestedIdsList;

}}

```

Simulation.java package

```

examples.org.cloudbus.cloudsim.examples; import

java.text.DecimalFormat; import

java.util.ArrayList; import java.util.Calendar; import

```

```

java.util.LinkedList; import java.util.List;

import java.util.Random;

import org.cloudbus.cloudsim.Cloudlet; import

org.cloudbus.cloudsim.CloudletSchedulerSpaceShared;

import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;

import org.cloudbus.cloudsim.Datacenter; import

org.cloudbus.cloudsim.DatacenterBroker; import

org.cloudbus.cloudsim.DatacenterCharacteristics; import

org.cloudbus.cloudsim.Host; import

org.cloudbus.cloudsim.Log; import org.cloudbus.cloudsim.Pe;

import org.cloudbus.cloudsim.Storage; import

org.cloudbus.cloudsim.UtilizationModel; import

org.cloudbus.cloudsim.UtilizationModelFull; import

org.cloudbus.cloudsim.Vm;

import org.cloudbus.cloudsim.VmAllocationPolicySimple; import

org.cloudbus.cloudsim.VmSchedulerTimeShared; import

org.cloudbus.cloudsim.core.CloudSim; import

org.cloudbus.cloudsim.provisioners.BwProvisionerSimple; import

org.cloudbus.cloudsim.provisioners.PeProvisionerSimple; import

org.cloudbus.cloudsim.provisioners.RamProvisionerSimple; /**

* An example showing how to create * scalable simulations.

public class Simulation {

    /** The cloudlet list. */ private static

    List<Cloudlet> cloudletList;

    /** The vm list. */

    }

```

```

private static List<Vm> vmlist;

private static List<Vm> createVM(int userId, int vms) {

//Creates a container to store VMs. This list is passed to the broker later

LinkedList<Vm> list = new LinkedList<Vm>();

//VM Parameters

long size = 10000; //image size

(MB) int ram = 512; //vm memory (MB) int mips =

1000; long bw =

1000;

int pesNumber = 1; //number of cpus

String vmm = "Xen"; //VMM name

//create VMs

Vm[] vm = new Vm[vms];

for(int i=0;i<vms;i++){

vm[i] = new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new

CloudletSchedulerSpaceShared());

//for creating a VM with a space shared scheduling policy for cloudlets:

//vm[i] = Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new

CloudletSchedulerSpaceShared());

list.add(vm[i]);

}

return list;

} private static List<Cloudlet> createCloudlet(int userId, int

cloudlets){

// Creates a container to store Cloudlets

```



```

LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();

//cloudlet parameters long length =
1000; long fileSize = 300; long
outputSize = 300;

int pesNumber = 1;

UtilizationModel utilizationModel = new UtilizationModelFull();

Cloudlet[] cloudlet = new Cloudlet[cloudlets];

for(int i=0;i<cloudlets;i++){

Random r= new Random();

cloudlet[i] = new Cloudlet(i, length +r.nextInt(2000), pesNumber,
fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

// setting the owner of these Cloudlets

cloudlet[i].setUserId(userId); list.add(cloudlet[i]);

}

return list;

}

////////////////////// STATIC METHODS ////////////////////////

/**
 * Creates main() to run this example
 */

public static void main(String[] args) {

Log.println("Starting CloudSimExample6...");

try {

// First step: Initialize the CloudSim package. It should be called

// before creating any entities. int num_user = 3;

}

```

```

// number of grid users Calendar calendar =
Calendar.getInstance(); boolean trace_flag = false; // mean trace events

// Initialize the CloudSim library

CloudSim.init(num_user, calendar, trace_flag);

// Second step: Create Datacenters

//Datacenters are the resource providers in CloudSim. We need at list one of them to run a
CloudSim simulation

Datacenter datacenter0 = createDatacenter("Datacenter_0");

Datacenter datacenter1 = createDatacenter("Datacenter_1");

//Third step: Create Broker

DatacenterBroker broker = createBroker();

int brokerId = broker.getId();

//Fourth step: Create VMs and Cloudlets and send them to broker vmList =

createVM(brokerId,10); //creating 20 vms

cloudletList = createCloudlet(brokerId,40); // creating 40 cloudlets

broker.submitVmList(vmList);

broker.submitCloudletList(cloudletList);

// Fifth step: Starts the simulation

CloudSim.startSimulation();

// Final step: Print results when simulation is over

List<Cloudlet> newList = broker.getCloudletReceivedList();

CloudSim.stopSimulation();

printCloudletList(newList);

//Print the debt of each user to each datacenter datacenter0.printDebts();

datacenter1.printDebts();

```

```

Log.println("CloudSimExample6 finished!");

}

catch (Exception e)

{

e.printStackTrace();

Log.println("The simulation has been terminated due to an unexpected error");

} }

private static Datacenter createDatacenter(String name){

// Here are the steps needed to create a PowerDatacenter:

// 1. We need to create a list to store one or more

// Machines

List<Host> hostList = new ArrayList<Host>();

// 2. A Machine contains one or more PEs or CPUs/Cores. Therefore, should

// create a list to store these PEs before creating

// a Machine.

List<Pe> peList1 = new ArrayList<Pe>();

int mips = 1000;

// 3. Create PEs and add these into the list. //for a quad-core machine, a list of

4 PEs is required:

peList1.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS

Rating

peList1.add(new Pe(1, new PeProvisionerSimple(mips)));

peList1.add(new Pe(2, new PeProvisionerSimple(mips))); peList1.add(new Pe(3, new

PeProvisionerSimple(mips)));

//Another list, for a dual-core machine List<Pe> peList2

```

```

= new ArrayList<Pe>()); peList2.add(new Pe(0, new
PeProvisionerSimple(mips))); peList2.add(new
Pe(1, new PeProvisionerSimple(mips)));

//4. Create Hosts with its id and list of PEs and add them to the list of machines

int hostId=0;

int ram = 2048; //host memory (MB)

long storage = 1000000; //host storage int

bw = 10000;

hostList.add(
new Host(
hostId,
new RamProvisionerSimple(ram),
new BwProvisionerSimple(bw),
storage,
peList1,
new VmSchedulerTimeShared(peList1)
)
); // This is our first machine

hostId++;

hostList.add(
new Host(
hostId,
new
RamProvisionerSimple(ram), new
BwProvisionerSimple(bw), storage, peList2,
++

```

```

new VmSchedulerTimeShared(peList2)

)

); // Second machine

//To create a host with a space-shared allocation policy for PEs to VMs:

//hostList.add(

// new Host(

// hostId,

//

new CpuProvisionerSimple(peList1),

// new RamProvisionerSimple(ram),

// new BwProvisionerSimple(bw),

// storage,

// new VmSchedulerSpaceShared(peList1)

// )

// );

//To create a host with a oportunistic space-shared allocation policy for PEs to VMs:

//hostList.add(

// new Host(

// hostId,

// new CpuProvisionerSimple(peList1),

// new RamProvisionerSimple(ram),

// new BwProvisionerSimple(bw),

// storage,

```

```

// new

VmSchedulerOpportunisticSpaceShared(peList1)

// )

);

// 5. Create a DatacenterCharacteristics object that stores
the

// properties of a data center: architecture, OS, list of

// Machines, allocation policy: time- or space-shared, time zone // and its
price (G$/Pe time unit).

String arch = "x86"; // system architecture

String os = "Linux"; // operating system

String vmm = "Xen";

double time_zone = 10.0; // time zone this resource located

double cost = 3.0; // the cost of using processing in this resource

double costPerMem = 0.05; // the cost of using memory in this resource

double costPerStorage = 0.1; // the cost of using storage in this resource

double costPerBw = 0.1; // the cost of using bw in this
resource

LinkedList<Storage> storageList = new LinkedList<Storage>(); //we are not adding SAN
devices by now

DatacenterCharacteristics characteristics = new DatacenterCharacteristics( arch,
os, vmm, hostList, time_zone, cost, costPerMem, costPerStorage, costPerBw);

// 6. Finally, we need to create a PowerDatacenter object. Datacenter datacenter = null;

try {

datacenter = new Datacenter(name, characteristics, new

```

```

VmAllocationPolicySimple(hostList), storageList, 0);

    } catch (Exception e) {

        e.printStackTrace();

    }

    return datacenter;

}

//We strongly encourage users to develop their own broker policies, to submit vms and cloudlets
according

//to the specific rules of the simulated scenario private static

DatacenterBroker createBroker(){ DatacenterBroker broker =

null; try { broker = new DatacenterBroker("Broker");

    } catch (Exception e) {

        e.printStackTrace();

        return null;

    }

    return broker;

}

/**

* Prints the Cloudlet objects

* @param list list of Cloudlets

*/

@SuppressWarnings("deprecation")

private static void printCloudletList(List<Cloudlet> list) { int size =

list.size();

    }

```

```

Cloudlet cloudlet;

String indent = " ";

Log.println();

Log.println("===== OUTPUT =====");

Log.println("Cloudlet ID" + indent + "STATUS" + indent +
"Data center ID" + indent + "VM ID" + indent + indent +
"Time" + indent + "Start Time" + indent + "Finish Time" +indent+"user id"+indent);

DecimalFormat dft = new DecimalFormat("###.##");

for (int i = 0; i < size; i++) { cloudlet = list.get(i);

Log.print(indent + cloudlet.getCloudletId() + indent + indent);

if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS){

Log.print("SUCCESS");

Log.println( indent + indent + cloudlet.getResourceId()
+ indent + indent + indent + cloudlet.getVmId() + indent + indent +
indent + dft.format(cloudlet.getActualCPUTime()) + indent + indent
+
dft.format(cloudlet.getExecStartTime())+ indent + indent + indent +
dft.format(cloudlet.getFinishTime())+indent +cloudlet.getUserId());

}}}}

```


PRACTICAL NO. 5

Aim: To study cloud security management.

Objectives: From this experiment, the student will be able,

- To understand the security features of Cloud.
- To learn the technique of application security management and its complexity
- To understand the importance of cloud security management from application point of view

Outcomes: The learner will be able to

- Student can study and implement single-sign-on.
- To use current techniques, skills, and tools necessary for computing practice.
- To match the industry requirements in the domains of Database management, Programming and Networking with the required management skills.

Outcomes: The learner will be able to

- Student can study and implement single-sign-on.
- To use current techniques, skills, and tools necessary for computing practice.
- To match the industry requirements in the domains of Database management, Programming and Networking with the required management skills.

Theory:

Cloud security management is the practice of ensuring the security of data, applications, and infrastructure in cloud computing environments. Cloud computing has become an increasingly popular choice for businesses and individuals alike due to its scalability, flexibility, and cost-effectiveness. However, it also presents a unique set of security challenges, including data breaches, unauthorized access, and cyber attacks. To effectively manage cloud security, it is important to understand the shared responsibility model between cloud service providers (CSPs) and cloud customers. CSPs are responsible for securing the underlying infrastructure and physical security of the data centre, while customers are responsible for securing the applications, data, and user access to the cloud environment.

Physical security

Physical security is an important aspect of cloud security management, as it involves the protection of the physical infrastructure and equipment that are used to host and store data in the cloud. Physical security measures are designed to prevent unauthorized access, theft, damage, or destruction of hardware and data.

Personal security

Personal security is also an important aspect of cloud security management, as it involves protecting the personal data and privacy of individuals who use cloud services. Personal security measures are designed to prevent unauthorized access, use, or disclosure of personal information stored in the cloud.

Some common personal security measures used in cloud security management include:

Application security

Application security is a critical component of cloud security management as it helps to protect against malicious attacks on applications running in the cloud. Cloud service providers often provide tools and services to help customers secure their applications in the cloud. Some of the

Procedure:

Security using MFA(Multi Factor Authentication) device code:

- 1) goto aws.amazon.com
- 2) click on "My Account"
- 3) select "AWS management console" and click on it
- 4) Give Email id in the required field

if you are registering first time then select "I am a new user" radio button

- 5) click on "sign in using our secure server" button
- 6) follow the instruction and complete the formalities

(Note: do not provide any credit card details or bank details)

sign out from

- 7) Again go to "My Account"

select "AWS management console" and click on it

Sign in again by entering the user name and valid password (check "I am returning user and my password is" radio button)

Now you are logged in as a Root User

All AWS project can be viewed by you, but you cant make any changes in it or you cant create new thing as you are not paying any charges to amazon (for reason refer step:6)

To create the user in a root user, follow the steps mentioned below:

- 1) click on "Identity and Access Management" in security and identity project
- 2) click in "Users" from dashboard

It will take you to "Create New Users"

click on create new user button

enter the "User Name"

(select "Generate and access key for each user" checkbox, it will create a user with a specific key)

click on "Create" button at right bottom

3) once the user is created click on it

4) go to security credentials tab

5) click on "Create Access Key", it will create an access key for user.

6) click on "Manage MFA device" it will give you one QR code displayed on the screen

you need to scan that QR code on your mobile phone using barcode scanner (install it in mobile phone) you also need to install "Google Authenticator" in your mobile phone to generate the MFA code

7) Google authenticator will keep on generating a new MFA code after every 60 seconds

that code you will have to enter while logging as a user.

Hence, the security is maintained by MFA device code...

one can not use your AWS account even if it may have your user name and password, because MFA code is on your MFA device (mobile phone in this case) and it is getting changed after every 60 seconds.

Permissions in user account:

After creating the user by following above mentioned steps; you can give certain permissions to specific user

1) click on created user

2) goto "Permissions" tab

3) click on "Attach Policy" button

4) select the needed policy from given list and click on apply.

Result:

Step 1 :goto aws.amazon.com

SECURITY

[AWS Cloud Security](#)

[What is Cloud Security?](#)

[Benefits of AWS Security](#)

[Security Guidance](#)

[Security Bulletins](#)

[Security Resources](#)

[Partner Solutions](#)

RELATED LINKS

[AWS Cloud Compliance](#)

[AWS Architecture](#)

[AWS Security Blog](#)

[Penetration Testing](#)

[Vulnerability Reporting](#)

AWS Cloud Security

Cloud Security at AWS is job zero. All AWS customers benefit from a data center and network architecture built to satisfy the requirements of the most security-sensitive organizations. AWS and its partners offer hundreds of tools and features to help you meet your security objectives around visibility, auditability, controllability and agility. This means that you can have the security you need, but without the capital outlay, and at a much lower operational overhead than in an on-premises environment.



- [Configure Rate-Based Blacklisting with AWS WAF and AWS Lambda](#)
- [DNS Resolution Between On-Premises Networks and AWS Using AWS Directory Service and Microsoft Active Directory](#)
- [DNS Resolution Between On-Premises Networks and AWS Using AWS Directory Service and Amazon Route 53](#)

Start Your Cloud Adoption

Contact AWS Sales



Step 2 : Click on "My Account". Select "AWS management console" and click on it. Give Email id in the required field

Sign In or Create an AWS Account

What is your email (phone for mobile accounts)?

E-mail or mobile number:

☒ I am a new user.

☐ I am a returning user and my password is:

[Sign in using our secure server](#)

[Forgot your password?](#)

New AWS Accounts Include:

12 months of access to the AWS Free Tier

Amazon EC2: 750 hrs/month of Windows and Linux t2.micro instance usage
Amazon S3: 5GBs of Storage
Amazon RDS: 750 hrs/month of Micro DB Instance usage
Amazon DynamoDB: 25 GB of storage, up to 200 million requests/month

AWS Basic Support Features

Customer Service: 24x7x365
Support Forums
Documentation, White Papers, and Best Practice Guides

Visit aws.amazon.com/free for full offer terms.

Learn more about [AWS Identity and Access Management](#) and [AWS Multi-Factor Authentication](#), features that provide additional security for your AWS Account. View full [AWS Free Usage Tier](#) offer terms.

Contact Information

☐ Company Account ☒ Personal Account

* Required Fields

Full Name*

Country*

* If you select India, your country selection cannot be changed after creating the account

Address*

City*

State / Province or Region*

Postal Code*

Phone Number*

Security Check 

[Refresh Image](#)

Step 3: Addition of security features

SECURITY

- [AWS Cloud Security](#)
- [What is Cloud Security?](#)
- [Benefits of AWS Security](#)

AWS Cloud Security

Cloud Security at AWS is job zero. All AWS customers benefit from a data center and network architecture built to satisfy the requirements of the most security-sensitive organizations. AWS and its partners offer

MY ACCOUNT

- [AWS Management Console](#)
- [Account Settings](#)
- [Billing & Cost Management](#)
- [Security Credentials](#)
- [Contact AWS Sales](#)

Step 4: Sign in to an AWS account

Sign In or Create an AWS Account

What is your email (phone for mobile accounts)?

E-mail or mobile number:

pawar.priyanka.12ce1048@gmail.com

☐ I am a new user.

☒ I am a returning user and my password is:

[Sign in using our secure server](#)

[Forgot your password?](#)

New AWS Accounts Include:

12 months of access to the AWS Free Tier

Amazon EC2: 750 hrs/month of Windows and Linux t2.micro instance usage
Amazon S3: 5GBs of Storage
Amazon RDS: 750 hrs/month of Micro DB Instance usage
Amazon DynamoDB: 25 GB of storage, up to 200 million requests/month

AWS Basic Support Features

Customer Service: 24x7x365
Support Forums
Documentation, White Papers, and Best Practice Guides

Visit aws.amazon.com/free for full offer terms.

Learn more about [AWS Identity and Access Management](#) and [AWS Multi-Factor Authentication](#), features that provide additional security for your AWS Account. View full [AWS Free Usage Tier](#) offer terms.

Service Catalog
Create and Use Standardized Products

Trusted Advisor
Optimize Performance and Security

Security & Identity

Identity & Access Management
Manage User Access and Encryption Keys

Directory Service
Host and Manage Active Directory

Inspector PREVIEW
Analyze Application Security

WAF

AWS Services Edit

Priyanka Vijay Pawar Global Support

Dashboard Search IAM

Users

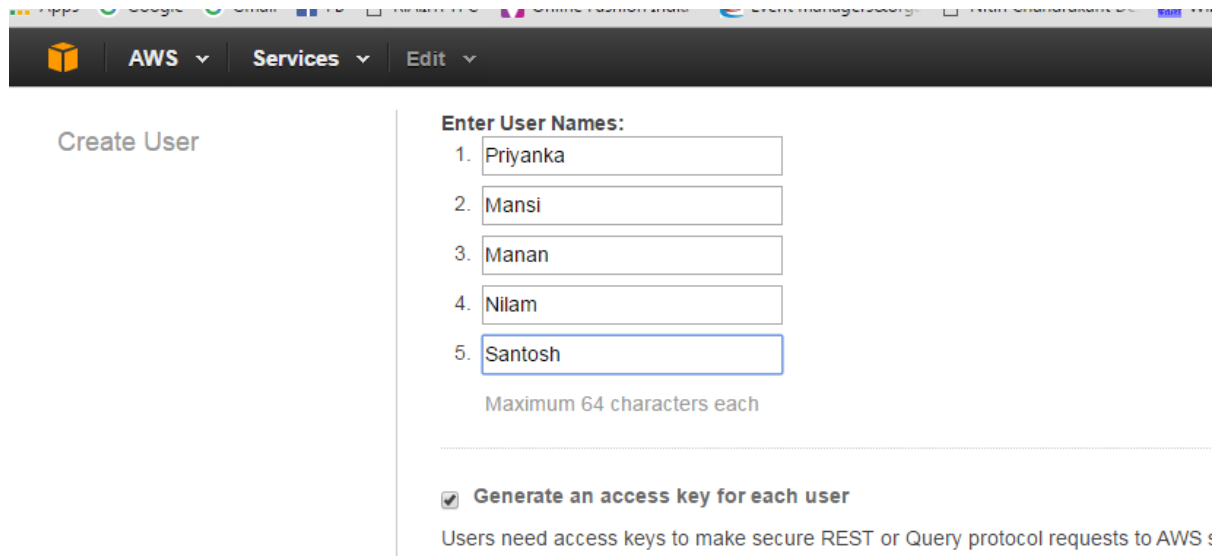
Details Groups Roles Policies Identity Providers

Create New Users User Actions

Filter

<input type="checkbox"/>	User Name	Groups	Password	Password Last Used	Access Keys	Creation Time
No records found.						

Step 5 : Creation of users



Create User

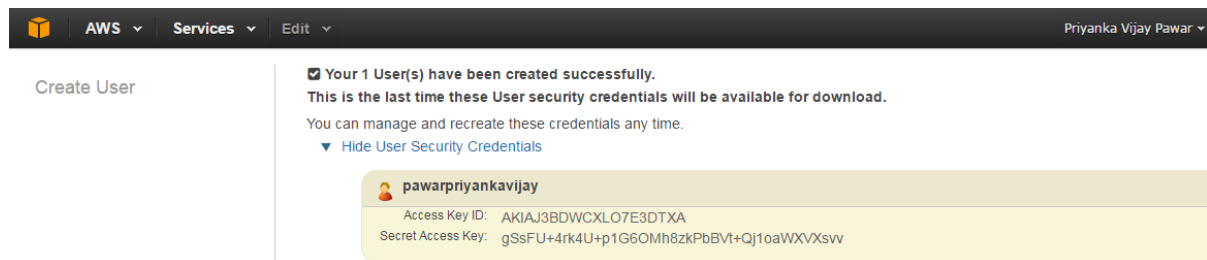
Enter User Names:

1.
2.
3.
4.
5.

Maximum 64 characters each

☒ **Generate an access key for each user**

Users need access keys to make secure REST or Query protocol requests to AWS :

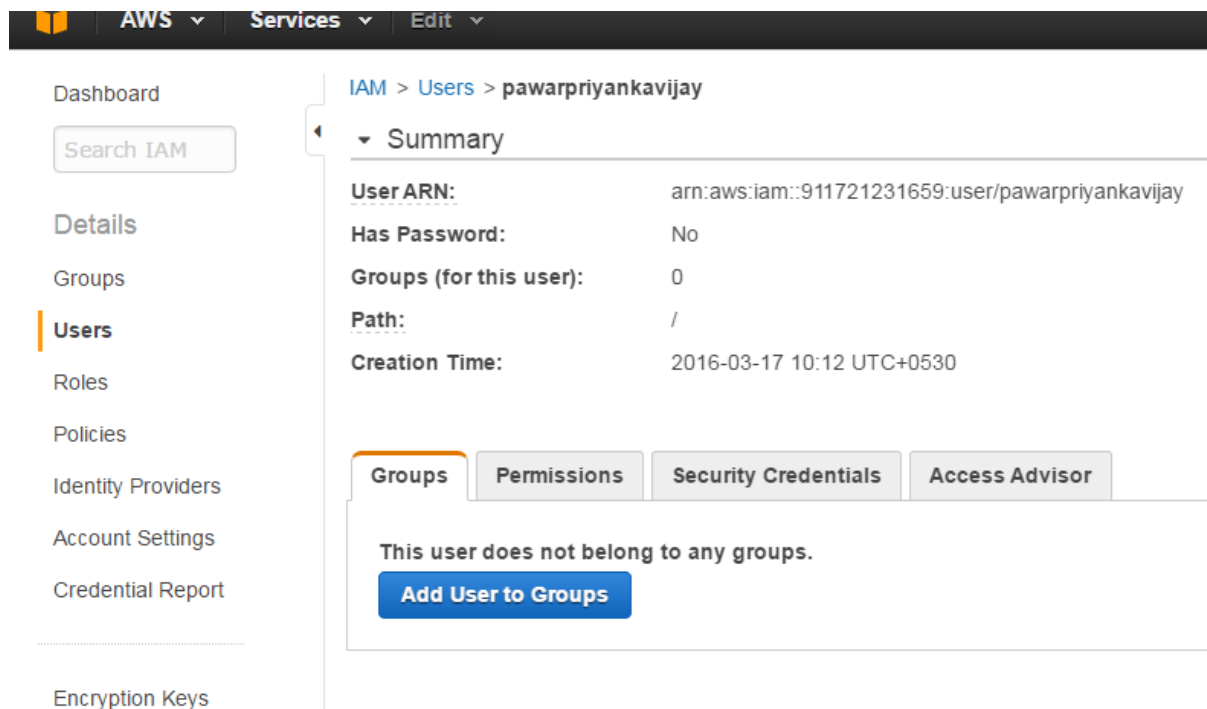


Create User

☒ **Your 1 User(s) have been created successfully.**
This is the last time these User security credentials will be available for download.
You can manage and recreate these credentials any time.
[Hide User Security Credentials](#)

pawarpriyankavijay
Access Key ID: AKIAJ3BDWCXLO7E3DTXA
Secret Access Key: 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

Step 6: Adding users to group



Dashboard

Details

Groups

Users

Roles

Policies

Identity Providers

Account Settings

Credential Report

Encryption Keys

IAM > Users > pawarpriyankavijay

Summary

User ARN: arn:aws:iam::911721231659:user/pawarpriyankavijay

Has Password: No

Groups (for this user): 0

Path: /

Creation Time: 2016-03-17 10:12 UTC+0530

Groups **Permissions** **Security Credentials** **Access Advisor**

This user does not belong to any groups.

Add User to Groups

Step 7: Creating Access key

GroupsPermissionsSecurity CredentialsAccess Advisor

Access Keys

Use access keys to make secure REST or Query protocol requests to any AWS service API. For your protection, you should never share your secret keys with anyone. In addition, industry best practice recommends frequent key rotation. [Learn more about Access Keys](#)

Create Access Key

Access Key ID	Created	Last Used	Last Used Service	Last Used Region	Status	Actions
AKIAJ3BDWCXLO7E3DTXA	2016-03-17 10:12 UTC+0530	N/A	N/A	N/A	Active	Make Inactive Delete

Sign-In Credentials

User Namepawarpriyankavijay

Manage Password

PasswordNo

Create Access Key

✓ Your access key has been created successfully.

This is the last time these User security credentials will be available for download.

You can manage and recreate these credentials any time.

▼ [Hide User Security Credentials](#)

pawarpriyankavijay

Access Key ID: AKIAJSIBMBDSFX3YMLRQ

Secret Access Key: 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

Close

Download Credentials

Manage MFA Device

Select the type of MFA device to activate:

☒ A virtual MFA device

☐ A hardware MFA device

For more information about supported MFA devices, see [AWS Multi-Factor Authentication](#).

Cancel

Next Step

Manage MFA Device

To activate a virtual MFA device, you must first install an AWS MFA-compatible application on the user's smartphone, PC, or other device. You can find a list of AWS MFA-compatible applications [here](#). After the application is installed, click Next Step to configure the virtual MFA.

☒ Don't show me this dialog box again.

Cancel

Previous

Next Step

Step 8 : Setting permissions to users

Groups

Users

Roles

Policies

Identity Providers

Account Settings

Credential Report

Encryption Keys

Path: /

Creation Time: 2016-03-17 10:12 UTC+0530

Groups

Permissions

Security Credentials

Access Advisor

Managed Policies

There are no managed policies attached to this user.





Attach Policy

Attach Policy

Select one or more policies to attach. Each user can have up to 10 policies attached.

Filter: Policy Type

Showing 193 results

	Policy Name	Attached Entities	Creation Time	Edited Time
<input checked="" type="checkbox"/>	 AdministratorAccess	0	2015-02-07 00:09 UTC+0530	2015-02-07 00:09 UTC+0530
<input checked="" type="checkbox"/>	 AmazonAPIGatewayAdministr...	0	2015-07-09 23:04 UTC+0530	2015-07-09 23:04 UTC+0530
<input type="checkbox"/>	 AmazonAPIGatewayInvokeFul...	0	2015-07-09 23:06 UTC+0530	2015-07-09 23:06 UTC+0530
<input type="checkbox"/>	 AmazonAPIGatewayPushToCL...	0	2015-11-12 05:11 UTC+0530	2015-11-12 05:11 UTC+0530

Groups

Users

Roles

Policies

Identity Providers

Account Settings

Credential Report

Encryption Keys

Groups

Permissions



Security Credentials


Access Advisor

Managed Policies

The following managed policies are attached to this user. You can attach up to 10 managed policies.


Attach Policy

Policy Name	Actions
 AdministratorAccess	Show Policy Detach Policy Simulate Policy
 AmazonAPIGatewayAdministrator	Show Policy Detach Policy Simulate Policy

 IAM Policy Simulator

Mode : Existing Policies ▾

Priyanka Vijay Pawar ▾



Policies

Back

Editing policy: AdministratorAccess

AWS Managed Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:*",
      "Resource": "*"
    }
  ]
}
```

Policy Simulator

Select service ▾

Select actions ▾

Select All

Deselect All

Reset Contexts

Clear Results

Run Simulation

▸ Global Settings ⓘ

Action Settings and Results [0 actions selected. 0 actions not simulated. 0 actions allowed. 0 actions denied.]

Service	Action	Resource Type	Simulation Resource	Permission
---------	--------	---------------	---------------------	------------

PRACTICAL NO. 6

Aim: To study and implementation of identity management.

Identity and access management, also known as IAM, is a framework of business procedures, laws, and technological advancements that makes managing digital or electronic identities easier. IAM frameworks allow information technology (IT) managers to manage user access to sensitive data within their organizations.

Single sign-on systems, two-factor authentication, multifactor authentication, and privileged access management are examples of the systems used for IAM technology. Furthermore, these technologies provide the capability of safely storing identity and profile data and data governance features to ensure that only information that is required and pertinent is shared.

IAM systems may be set up on-premises, made available by a third-party vendor under a subscription-based cloud model, or set up in a hybrid model.

IAM configuration fundamentally consists of the following elements:

- Understanding the distinction between identity management and authentication can help you better comprehend how people are identified in a system.
- How roles are defined in a system, and how people are given those roles.
- Updating, deleting, and adding people's roles in a system.
- Granting different levels of access to specific people or groups of people; and
- Both safeguarding the system's sensitive data and maintaining system security.

The Importance of IAM

There is more organizational and governmental pressure on business executives and IT departments to protect access to corporate resources. As a result, they can no longer manually assign and track user privileges and are prone to error processes. These tasks are automated by IAM services, making it possible to audit and control granular access to all corporate assets on-site and in the cloud.

IAM, which has a long list of features that keeps growing and includes biometrics, behavior analytics, and AI, is well suited to the challenges of the new IAM security environment. For instance, IAM's strict control over resource access in highly dispersed and dynamic environments is in line with the industry's shift from firewalls to zero-trust models and with the IoT's security requirements. For more details on the future of IoT security, check out this video.

IAM is a technology that businesses of all sizes can use, despite the misconception among IT experts that it is only appropriate for larger firms with higher budgets.

Identity and Access Management Best Practices



1) Clearly Define IAM Vision

The critical fundamental for successful Identity and Access Management (IAM) implementation is understanding it as a combination of technology solutions and business processes to manage identities and access corporate data and applications.

- Start to tie in business processes with your IAM program from the concept stage itself.
- Build your current and future IT capabilities, such as cloud-based implementations based on the current IT and network infrastructure.
- Engineer the roles between users and applications regarding privileges, rules, policies, and constraints.
- Map access privileges to business roles, identify excessive privileges, accounts, and redundant/dead groups.
- Make sure to fulfil all auditing requirements to be in line with compliance regulations, privacy, and data governance policies. This will help the teams make informed decisions.
- Take the enterprise-wide approach in implementing authorization procedures, security, and management, integration across domains part of your IAM architecture.

2) Develop A Strong Foundation

This requires a comprehensive evaluation of IAM product capabilities and its sync with organizational IT. This should be followed by an effective risk assessment of all organizational applications and platforms.

The assessment should ideally cover:

- Comparison between standard and in-house, and their versions
- Identification of OS, third-party apps currently in use and mapping with the functionalities offered by the IAM program
- Customizations made to fulfil new requirements
- Technological capabilities and limitations

Don't forget to involve IAM Subject Matter Experts (SMEs) in standardizing and enforcement of the IAM policy.

3) Stage-wise Implementation

Based on the first two practices, the IAM program should be implemented. A stage-wise procedure is recommended to avoid complexities in the IAM implementation process.

4) Stakeholder Awareness

Unlike usual training sessions, the IAM program-related stakeholder awareness program should cover detailed training on the underlying technology, product abilities, and scalability factors.

Each IAM solution implementation awareness program should have an approach tailored to the requirements of different user communities.

More than anyone, IT teams require detailed know-how of the IAM program and its core activities. Even the Operations team should be aware of the capabilities across different stages of the IAM lifecycle.

The training process should be a continuous activity and should happen in tandem with the changing processes or emerging capabilities.

5) Consider Identity as Primary Security Perimeter

Organizations should shift from the traditional focus on network security to considering identity as the primary security perimeter. With the explosion of cloud and remote working culture, network perimeter is becoming increasingly porous, and perimeter defense can't be effective. Centralize security controls around user and service identities.

6) Enforce Multi-Factor Authentication

Enable Multi-Factor Authentication (MFA) for all your users, including administrators and C-suite executives. It checks multiple aspects of a user's identity before allowing access to an application or database, instead of regular sign-in aspects. MFA is an integral part of identity and access management.

7) Establish Single Sign-On : Organizations must establish Single Sign-On (SSO) for their devices, apps, and services so users can use the same set of credentials to access the resources they need, wherever and whenever. You can achieve SSO by using the same identity solution for all your apps and resources, whether on-premises or in the cloud.

8) Implement Zero-Trust Policy

The zero-trust model assumes every access request as a threat until verified. Access requests from both inside and outside of the network are thoroughly authenticated, authorized, and scrutinized for anomalies before granting permission.

9) Enforce a Strong Password Policy

Implement an organization-wide password policy to ensure users set strong passwords for access. Make sure that employees update their passwords regularly and avoid using sequential and repetitive characters.

10) Secure Privileged Accounts

Securing privileged accounts is imperative to protect critical business assets. Limiting the number of users having privileged access to the organization's critical assets reduces the chance of unauthorized access to a sensitive resource. You must isolate the privileged accounts from the risk of being exposed to cybercriminals.

11) Conduct Regular Access Audits

Organizations must regularly conduct access audits to review all the granted accesses and check if they are still required. As users often request additional access or want to revoke their access, these audits help you manage such requests accordingly.

12) Implement Passwordless Login

Passwordless login is the process of authenticating users without the need for a password. It prevents scenarios where cybercriminals leverage weak and repetitive passwords to gain access to the network. Passwordless login can be implemented through various approaches, including email-based login, SMS-based login, and biometrics-based login.

These 12 best practices help in the smooth and seamless implementation of an IAM program.

A cost-effective IAM program can also be achieved through:

- In-depth requirement analysis as a combination of information gathering and perfect scope definition
- Effective design backed by a perfectly planned architecture and solution design
- Robust development through perfect process setup and effective integration
- Streamlined production roll-out with seamless migration from User Acceptance Testing to live release
- Effective support and maintenance through proper training, post-production, and enhancements

Most IAM programs fail due to ineffective management in either single or all stages of implementation. This is where the above listed IAM best practices help in the smooth implementation of an IAM program.

Looking for Identity and Access Management Implementation Support in US?

Although the identity and access management sector is constantly evolving, some core IAM best practices can help your business develop its IAM strategy. Expand your IAM architecture and fortify your security posture using these best practices.

Security experts may restrict access to endpoints and minimize the attack surface on corporate networks with the appropriate Identity and Access Management systems. As a result, they can simplify life for legitimate users while monitoring user privileges, preventing unwanted access, and reducing the danger of damaging data breaches. This is where Verities comes in place.

Choosing Verities for IAM Solutions implementation is a good option for the following reasons:

- IAM Subject Matter Expertise support
- Strategic IAM roadmap and design
- Minimized risk scope in modifying IAM architecture designs
- Quicker product evaluation
- Expected ROI and enhanced user experience
- Tailored solutions for smooth roll-out
- Effective application on-boarding
- Easy and effective migration
- Seamless deployment of environments

Verities, the Stevie Award Winner, can assist your business with the immediate implementation of identity and access management through our extensive infrastructure access platform.

PRACTICAL NO. 7

Aim: Case Study - Amazon Web Services/Microsoft Azure/Google cloud services.

What is AWS?

Amazon Web Services (AWS) is a subsidiary of Amazon providing on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you-go basis. Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud platform, offering over 175 fully-featured services from data centres globally. Millions of customers — including the fastest-growing startups, largest enterprises, and leading government agencies — are using AWS to lower costs, become more agile, and innovate faster.

History of AWS:

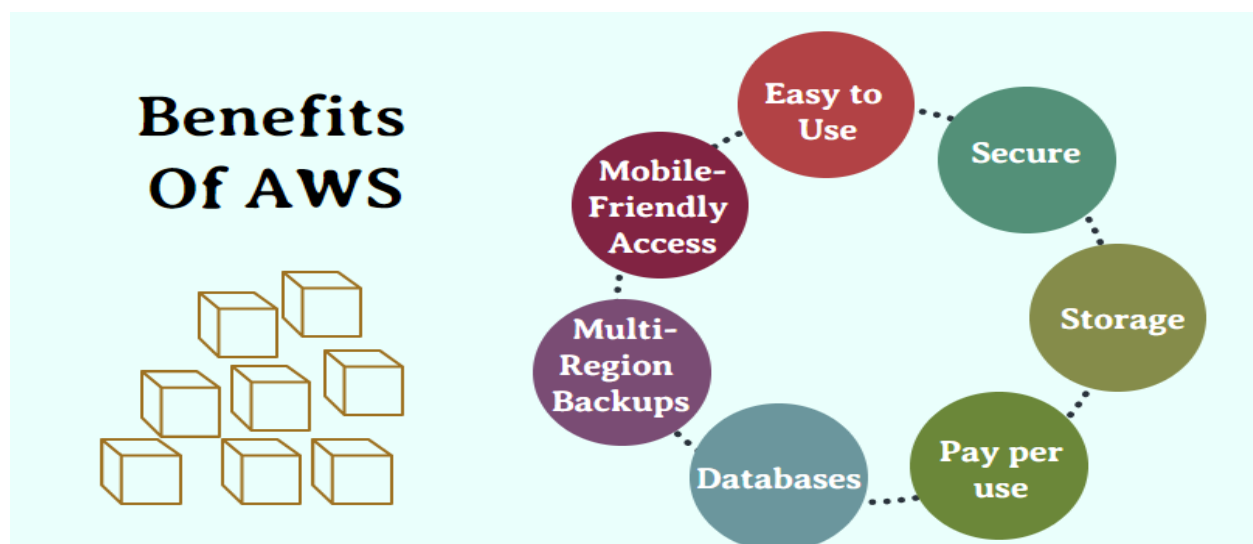
Amazon launched its first cloud computing service, Simple Storage Service (S3) in March of 2006. But the idea for the public cloud began germinating at the company several years earlier.

A popular myth says that Amazon began selling public cloud computing services because it had “excess capacity” from running its eCommerce website. Executives have repeatedly contradicted that story, saying that Amazon Web Services was designed from the ground up as a service for outside customers. However, the company's experiences with eCommerce did help lay the groundwork for AWS.

In the early 2000s, Amazon.com internal development team had a problem. They were adding a lot of software engineers, but despite the growing headcount, the pace of development was staying about the same. The issue was that each developer was setting up new and unique compute, storage and database resources for each project. The IT group realized that if they could standardize those resources and simplify the process of deploying new IT infrastructure, they might be able to speed things up.

In 2003, former Amazon employee Benjamin Black and his boss Chris Pinkham wrote a paper for Amazon founder and CEO Jeff Bezos. It described “a vision for Amazon infrastructure that was completely standardized, completely automated, and relied extensively on web services for things like storage.” In a blog post, Black explained, “Near the end of it, we mentioned the possibility of selling virtual servers as a service.”

Benefits of Amazon Web Services:



- **Most functionality** -AWS has significantly more services, and more features within those services, than any other cloud provider—from infrastructure technologies like compute, storage, and databases—to emerging technologies, such as machine learning and artificial intelligence, data lakes and analytics, and Internet of Things. This makes it faster, easier, and more cost-effective to move your existing applications to the cloud and build nearly anything you can imagine. AWS also has the deepest functionality within those services. For example, AWS offers the widest variety of databases that are purpose-built for different types of applications so you can choose the right tool for the job to get the best cost and performance.
- **The largest community of customers and partners** -AWS has the largest and most dynamic community, with millions of active customers and tens of thousands of partners globally. Customers across virtually every industry and of every size, including startups, enterprises, and public sector organizations, are running every imaginable use case on AWS. The AWS Partner Network (APN) includes thousands of systems integrators who specialize in AWS services and tens of thousands of independent software vendors (ISVs) who adapt their technology to work on AWS.
- **Most secure**-AWS is architected to be the most flexible and secure cloud computing environment available today. Our core infrastructure is built to satisfy the security requirements for the military, global banks, and other high-sensitivity organizations. This is backed by a deep set of cloud security tools, with 230 security, compliance, and governance services and features. AWS supports 90 security standards and compliance certifications, and all 117 AWS services that store customer data offer the ability to encrypt that data.
- **Fastest pace of innovation** -With AWS, you can leverage the latest technologies to experiment and innovate more quickly. We are continually accelerating our pace of innovation to invent entirely new technologies you can use to transform your business. For example, in 2014, AWS pioneered the serverless computing space with the launch of AWS Lambda, which lets developers run their code without provisioning or managing servers. And AWS built Amazon Sage Maker, a fully managed machine learning service that empowers everyday developers and scientists to use machine learning—without any previous experience.
Most proven operational expertise — AWS has unmatched experience, maturity, reliability, security, and performance that you can depend upon for your most important applications. For over 13 years, AWS has been delivering cloud services to millions of customers around the world running a wide variety of use cases. AWS has the most operational experience, at greater scale, of any cloud provider.

Agility- The cloud gives you easy access to a broad range of technologies so that you can innovate faster and build nearly anything that you can imagine. You can quickly spin up resources as you need them—from infrastructure services, such as compute, storage, and databases, to the Internet of Things, machine learning, data lakes and analytics, and much more. You can deploy technology services in a matter of minutes, and get from idea to implementation several orders of magnitude faster than before. This gives you the freedom to experiment, test new ideas to differentiate customer experiences and transform your business.

Elasticity -With cloud computing, you don't have to over-provision resources upfront to handle peak levels of business activity in the future. Instead, you provision the number of resources that you actually need. You can scale these resources up or down to instantly to grow and shrink capacity as your business needs change.

Cost Savings-The cloud allows you to trade capital expenses (such as data centres and physical servers) for variable expenses and only pay for IT as you consume it. Plus, the variable expenses are much lower than what you would pay to do it yourself because of the economies of scale.

Deploy Globally in minutes-With the cloud, you can expand to new geographic regions and deploy globally in minutes. For example, AWS has infrastructure all over the world, so you can deploy your application in multiple physical locations with just a few clicks. Putting applications in closer proximity to end users reduces latency and improves their experience.

Types of cloud computing

The three main types of cloud computing include Infrastructure as a Service, Platform as a Service, and Software as a Service. Each type of cloud computing provides different levels of control, flexibility, and management so that you can select the right set of services for your needs.

1.IaaS (Infrastructure as a Service)- IaaS contains the basic building blocks for cloud IT. It typically provides access to networking features, computers (virtual or on dedicated hardware), and data storage space. IaaS gives you the highest level of flexibility and management control over your IT resources. It is most similar to the existing IT resources with which many IT departments and developers are familiar.

2.Platform as a Service -PaaS removes the need for you to manage underlying infrastructure (usually hardware and operating systems), and allows you to focus on the deployment and management of your applications. This helps you be more efficient as you don't need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.

3.Software as a Service-SaaS provides you with a complete product that is run and managed by the service provider. In most cases, people referring to SaaS are referring to end-user applications (such as web-based email). With a SaaS offering, you don't have to think about how the service is maintained or how the underlying infrastructure is managed. You only need to think about how you will use that particular software.

Who is using cloud computing?

Organizations of every type, size, and industry are using the cloud for a wide variety of use cases, such as data backup, disaster recovery, email, virtual desktops, software development and testing, big data analytics, and customer-facing web applications. For example, healthcare companies are using the cloud to develop more personalized treatments for patients. Financial services companies are using the cloud to power real-time fraud detection and prevention. And video game makers are using the cloud to deliver online games to millions of players around the world.

Best Public Cloud Providers

Amazon Web Services

Amazon Web Services (AWS) is the undisputed market leader in cloud computing, from overall market share to most expansive cloud offering. It has vast resources, allowing it to design and execute new solutions at a dizzying pace, sometimes — or often — faster than customers can understand or incorporate them.

Microsoft Azure

Most market analysts put Microsoft squarely in the number two spot behind Amazon. Its cloud portfolio is exhaustive: in addition to its Azure IaaS and PaaS offerings, Microsoft also has several SaaS offerings, including its Office 365 products, the online versions of its Dynamics line of

IBM Cloud

Although it hasn't always been considered one of the “big three” cloud computing vendors, IBM's cloud business has been coming on strong. Particularly advantageous was IBM's acquisition of Red Hat, which allows it to leverage Red Hat's OpenShift cloud platform, a PaaS solution geared for containers. The company is also strong in hybrid — befitting its customer base of large enterprise clients, which in some cases are working on a classic lift and shift migration to the cloud, for which IBM is well suited.

When to Use AWS

AWS offers something for everyone — whether you are a developer working on a hobby project or a Fortune 500 company looking to become more agile. It is the generalist of the public cloud computing market with a huge array of services available. It is often used in hybrid IT.

When Not to Use AWS

If AWS has a weakness, it is its lack of offerings for hybrid cloud deployments. Analysts say that most enterprises will be pursuing a hybrid cloud, multi-cloud strategy, and Amazon's competitors Microsoft Azure and IBM have an advantage in this area. Because many large organizations already use Microsoft and IBM products in their data centres, they naturally gravitate to these other providers for the public cloud portion of their hybrid clouds.

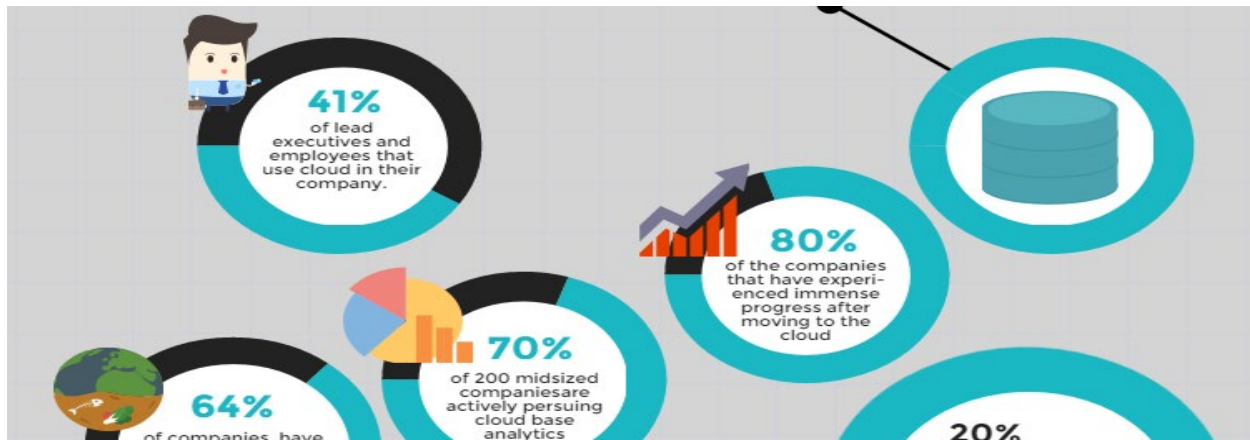
AWS Use Cases

Millions of customers — including the fastest-growing startups, largest enterprises, and leading government agencies — are using AWS to lower costs, become more agile, and innovate faster.

In every field, the AWS service is used. Below are some areas and some top companies use AWS.

- Aerospace (NASA, Maxar, ESA etc.)
- Gaming (MPL, FanFight, Gammaton etc.)
- Education (Coursera, BYJU's etc.)
- Telecommunication (Pinterest, Vodafone, Aircel etc.)
- Entertainment (Netflix, Hotstar etc.)
- Media (BBC, The Hindu, Punjab Kesri etc.)
- Software (Share chat, Slack etc.)

Cloud Computing Facts



By now, it's obvious that the cloud is widely used and that people spend a great deal of money creating, managing, and upgrading their cloud computing systems, but you may be wondering *how* widely used it is and *how* much money the cloud draws. Here are some fascinating facts to wrap your mind around:

Nearly one-half of US Government agencies use the cloud.

Annually, these agencies spend *\$2 billion on creating*, supporting, and *maintaining cloud services*. Some experts say that the government is the largest user of the cloud in the world. Within the different branches of the government that use the cloud, commercial clouds, private clouds, and shared clouds are all used. Private clouds, specifically, are employed by the government in an attempt to maintain security and control over the cloud.

Banking produces the most activity within the cloud.

This is due in large part to the introduction of widespread mobile banking services in 2013. Additionally, the rise of crowdfunding services and money management services like PayPal have grown, driving the act of paying virtually for goods through the roof. With the rising popularity of virtual currencies like Bitcoin, this trend of *cloud computing for banking* is only projected to continue.

The cloud computing market is projected to reach \$106 billion by 2016.

This represents a *30% growth rate from 2013*. This is especially stark when you take into account that the entire enterprise IT industry is only expected to experience a 5% growth rate between 2013 and 2018. Additionally, experts project that by 2018, 59% of all of the cloud's workload will be the result of Software-as-a-Service. This represents a 41% increase since 2013.

60% of U.S. IT decision-makers trust the security of the cloud.

When it comes to storing sensitive data in the cloud, the grand majority of IT experts aren't concerned. Additionally, *80% of enterprises* globally use the cloud to store their data. What's more, 82% of companies reported *saving money* when they adopted the cloud, and 14% report downsizing their IT department as a result of the presence of the cloud.

Within the next five years, the public cloud will experience a 44% growth.

This is in contrast to the *9% growth rate* that is projected for "on-premise" and hard-drive-based forms of computing.

Why is AWS so Successful?

Amazon Web Services (AWS), the cloud platform offered by Amazon.com Inc (AMZN), has become a giant component of the e-commerce giant's business portfolio. In the first quarter of 2020, AWS brought in a record \$10 billion of revenue, accounting for 13.5% of Amazon's total revenue. Having grown steadily in the 30-per cent range the past few quarters, AWS is a frontrunner to other cloud computing platforms such as competitor Microsoft Azure.

AWS is the most robust cloud offering available to date! I believe that ... unfortunately ... people tend to look at user-friendliness, lower or lowest cost, and even a limited amount of CSP products (easier to make up one's mind when there are only a few things to contend with) ... to decide on what's best. This was the case years ago with the microprocessor revolution: Intel Vs AMD Vs Cyrus. The incumbent was the most expensive ... and the competition came with not only more attractive price tags ... but in some cases more speed for the price and other desirables attributes. But at the end of the day ... there would always be a fantastic "must-have" application that would run "buggy at best" ... you know ... the app that YOU bought ... just to find out later after researching online ... or worse yet ... you took the time to read the box off the shelf from Microcenter or Best Buy that states "only runs on Intel processors" ... and one at a certain level of production (i7 for example). So when in doubt ... or ... I care to not dare the scalability Gods and endure their wrath ... just go with the one that may cost a bit more ... and may call for a little more patience to grasp and understand ... that will give you lots of options for your company and customers ... has been around the longest and still holds the lion's share of the market, etc, etc. Plus ... AWS offers a ton of managed services and resources that allow you to focus on your business ... and less on the technology!