Ex. No: 4 Date:	Simulation of VM Scheduling Using CloudSim

AIM

To simulate VM Scheduling using CloudSim.

DESCRIPTION

CloudSim

- A Framework for modeling and simulation of Cloud Computing Infrastructures and services
- Originally built at the Cloud Computing Distributed Systems (CLOUDS) Laboratory, The University of Melbourne, Australia
- It is completely written in JAVA

Main Features of CloudSim

- Modeling and simulation
- Data center network topologies and message-passing applications
- Dynamic insertion of simulation elements
- Stop and resume of simulation
- Policies for allocation of hosts and virtual machines

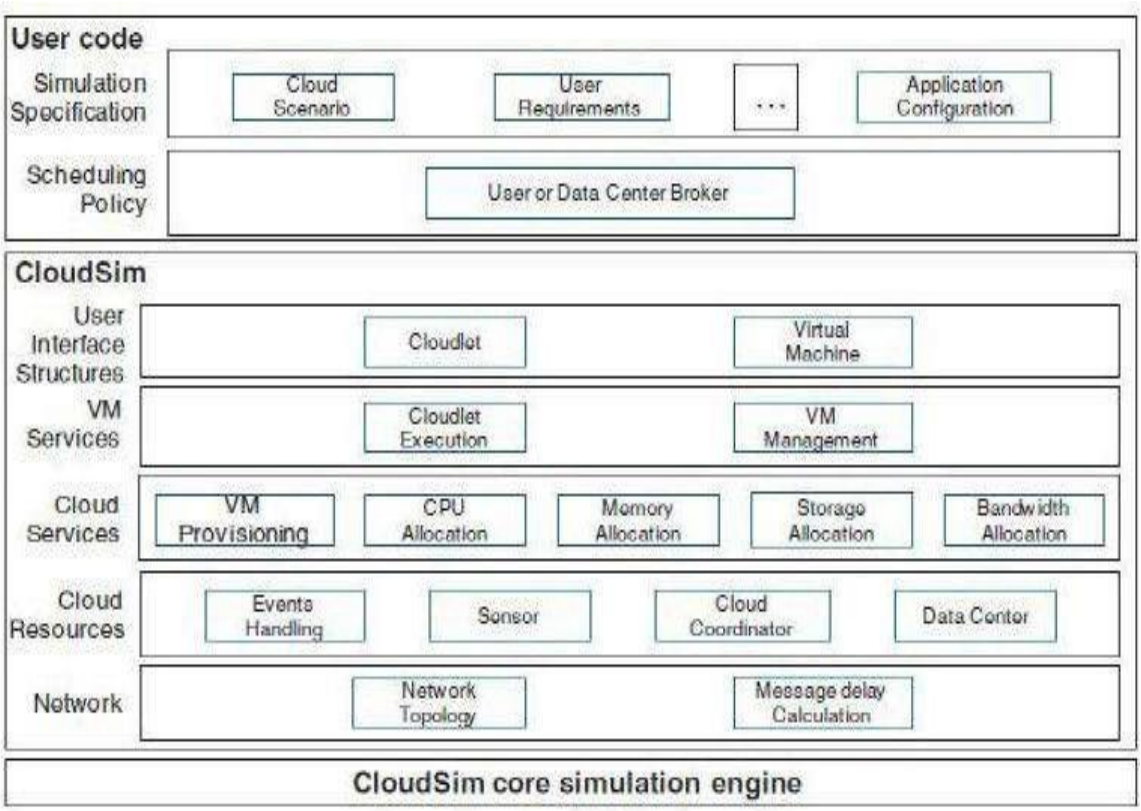
CloudSim – Essentials

- JDK 1.6 or above http://tinyurl.com/JNU-JAVA
- Eclipse 4.2 or above http://tinyurl.com/JNU-Eclipse
- Alternatively NetBeans https://netbeans.org/downloads
- Up & Running with cloudsim guide: https://goo.gl/TPL7Zh

CloudSim – Directory structure

- cloudsim/ -- top level CloudSim directory
- docs/ -- CloudSim API Documentation
- examples/ -- CloudSim examples
- jars/ -- CloudSim jar archives
- sources/ -- CloudSim source code

CloudSim - Layered Architecture



- **Core Functionalities** are queuing and processing of events, creation of Cloud system entities (services, host, data center, broker, VMs), communication between components, and management of the simulation clock.
- **CloudSim simulation layer** provides support for modeling and simulation of virtualized Cloud-based data center environments including dedicated management interfaces for VMs, memory, storage, and bandwidth.
- A Cloud host can be concurrently allocated to a set of VMs that execute applications based on SaaS provider's defined QoS levels.
- Top-most layer in the CloudSim stack is the User Code that exposes **basic entities for hosts** (number of machines, their specification, and so on), **applications** (number of tasks and their requirements), **VMs**, **number of users and their application types, and broker scheduling policies**.

Datacenter Entity

- The infrastructure-level services (IaaS) related to the clouds can be simulated by **extending the data center entity** of CloudSim.
- Data center entity **manages** several **host entities.**
- Hosts are assigned to one or more VMs based on a VM allocation policy that should be defined by the Cloud service provider.
- VM policy stands for the operations control policies related to VM life cycle such as provisioning of a host to a VM, VM creation, VM destruction, and VM migration.

Host Entity

- Represents a physical computing server in a Cloud.
- It assigns a pre-configured processing capability (expressed in millions of instructions per second—MIPS), memory, storage, and a provisioning policy for allocating processing cores to VMs.
- The Host component implements interfaces that support modeling and simulation of both single-core and multi-core nodes.

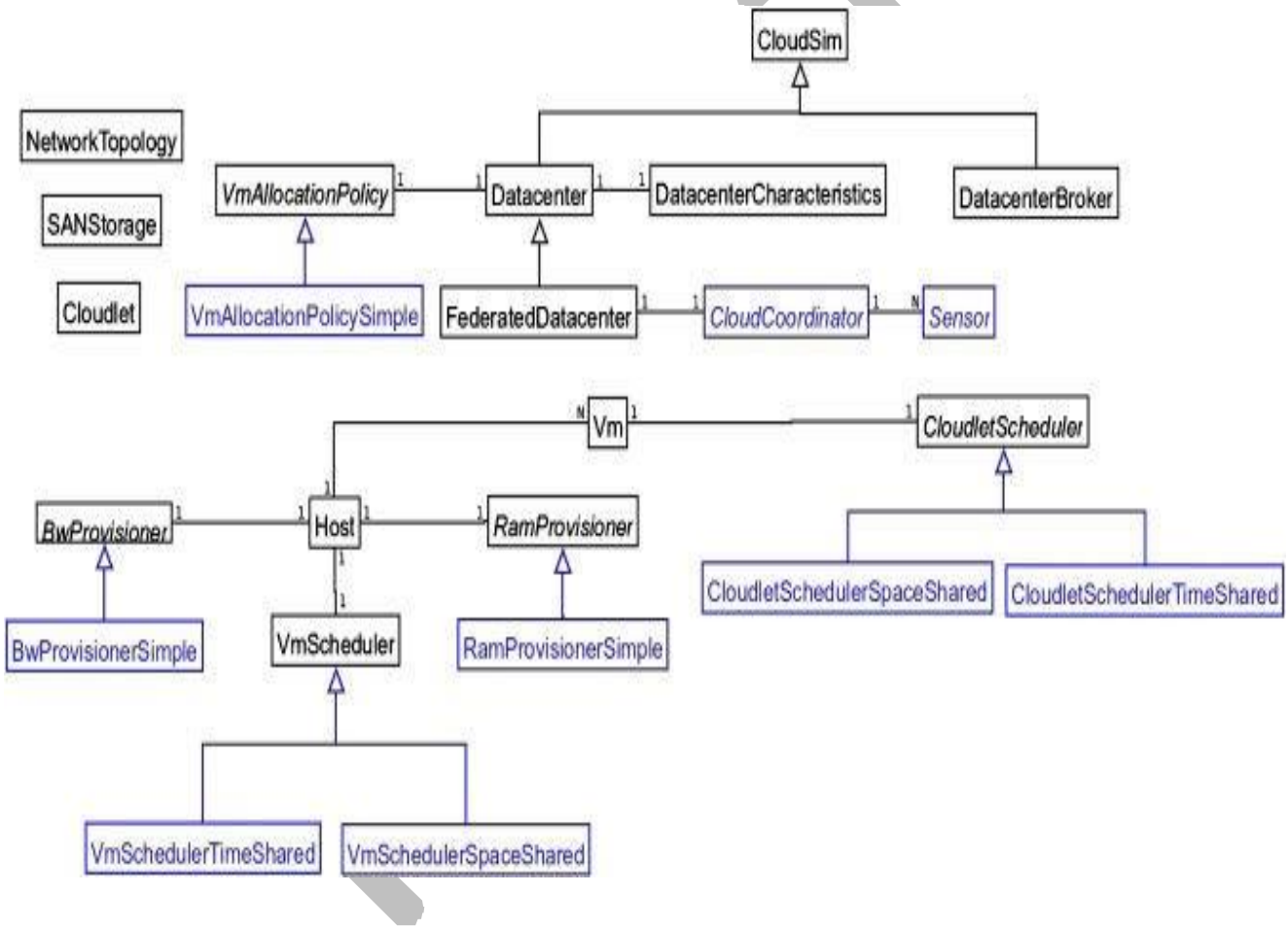
VM Allocation

- Process of creating VM instances on hosts that match the critical characteristics (storage, memory), configurations (software environment), and requirements (availability zone) of the SaaS provider.
- Once an application service (Cloudlet) is defined and modeled, it is assigned to one or more pre-instantiated VMs through a service-specific allocation policy.
- VmAllocationPolicy - VM Allocation controller component to allocate application-specific VMs to hosts in a data center. new policies can be created by the user by extending VmAllocationPolicy Entity.
- Default VmAllocationPolicy is FCFS.
- Hardware requirements, such as the number of processing cores, memory, and storage, form the basis for such provisioning.
- Other policies, including the ones likely to be expressed by Cloud providers, can also be easily simulated and modeled in CloudSim.
- Policies used by public Cloud providers (Amazon EC2, Microsoft Azure) are not publicly available

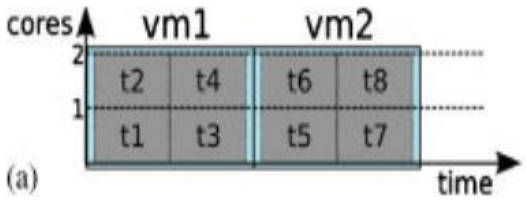
VM scheduler

- Allocation of processing cores to VMs is done by VM Scheduler using host allocation policy.
- Hardware characteristics, such as number of CPU cores, CPU share, and amount of memory (physical and secondary), that are allocated to a given VM instance
- Types:
- **Space Shared:** Assign specific CPU cores to specific VMs
- **Time Shared:** Dynamically distribute the capacity of a core among VMs

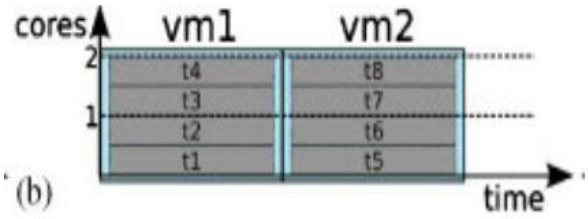
CloudSim – Component Model Classes



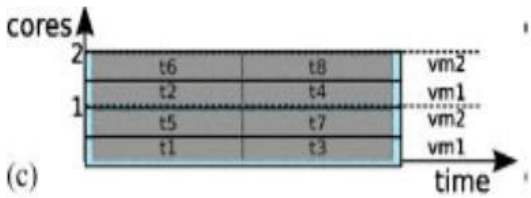
Modelling VM Allocation



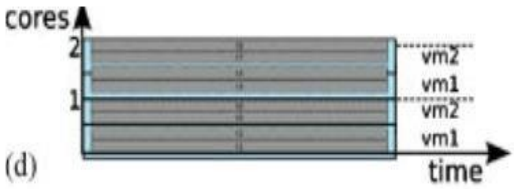
Space-shared provisioning for VMs and tasks



Space-shared provisioning for VMs and time-shared provisioning for tasks

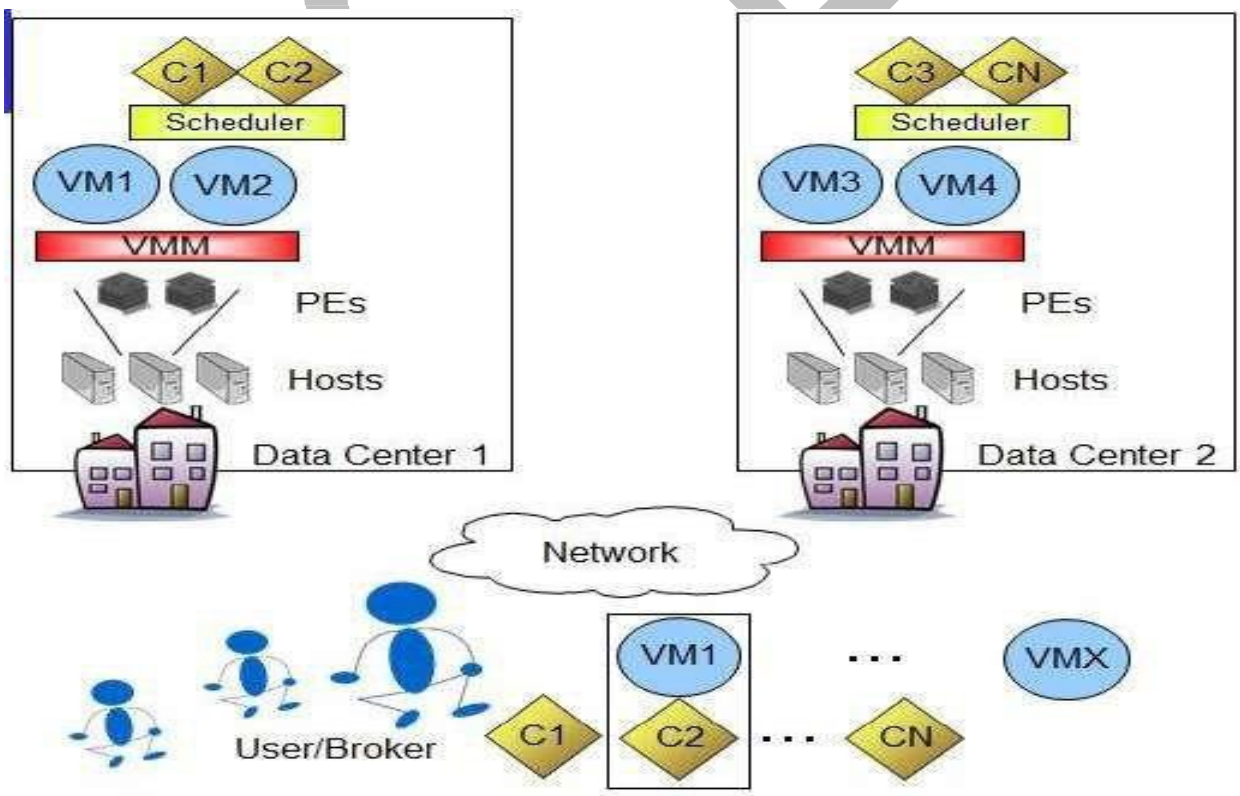


Time-shared provisioning for VMs, space-shared provisioning for tasks



Time-shared provisioning for VMs and tasks

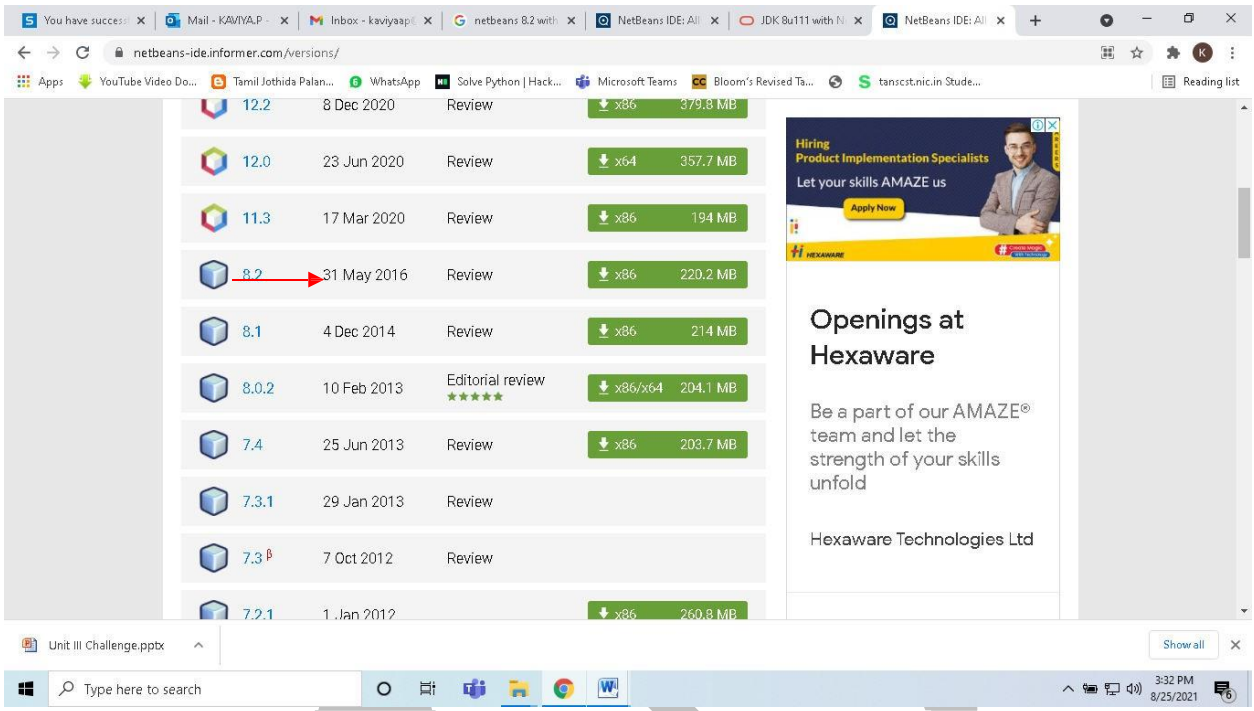
CloudSim Elements / Components



PROCEDURAL STEPS

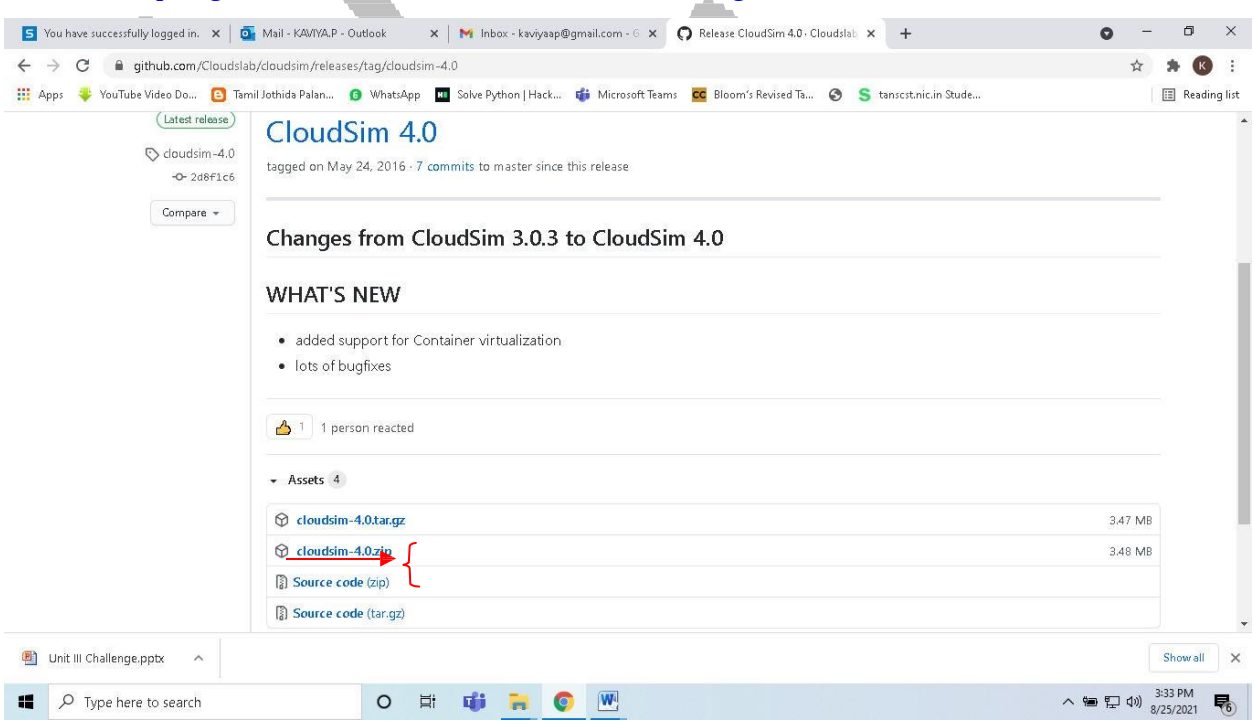
1. Download NetBeans and intall it.

Link: https://netbeans-ide.informer.com/versions/

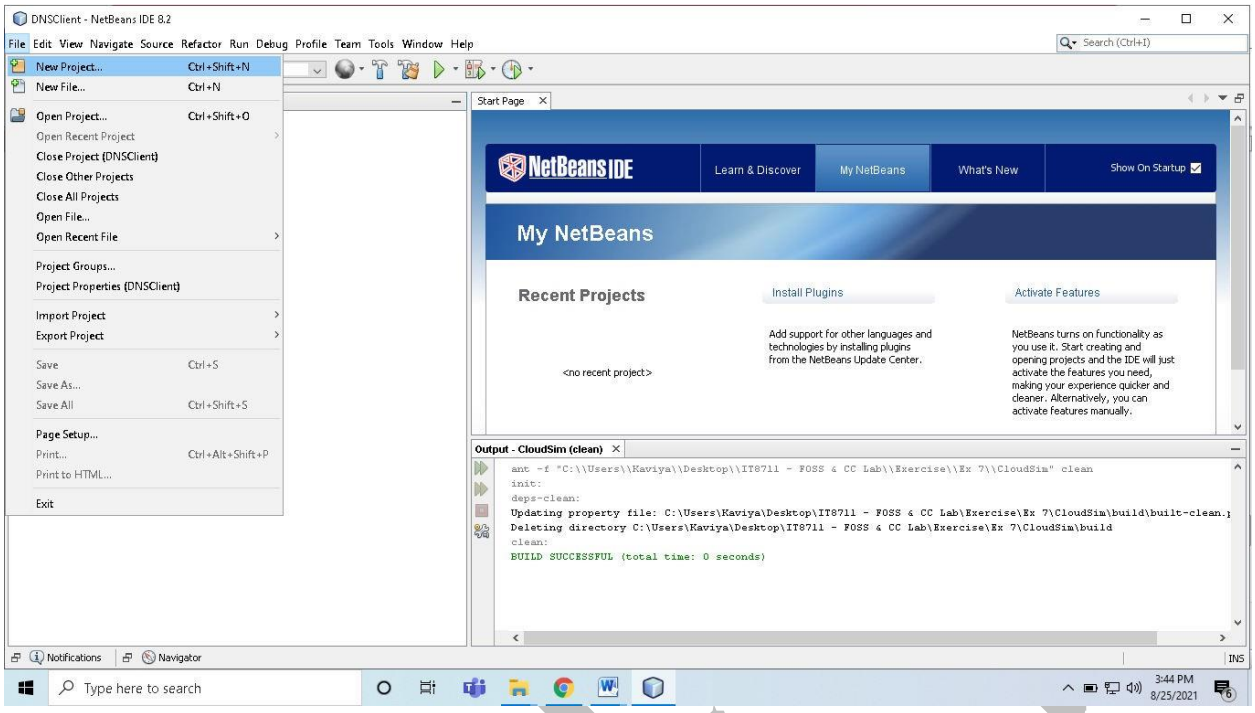


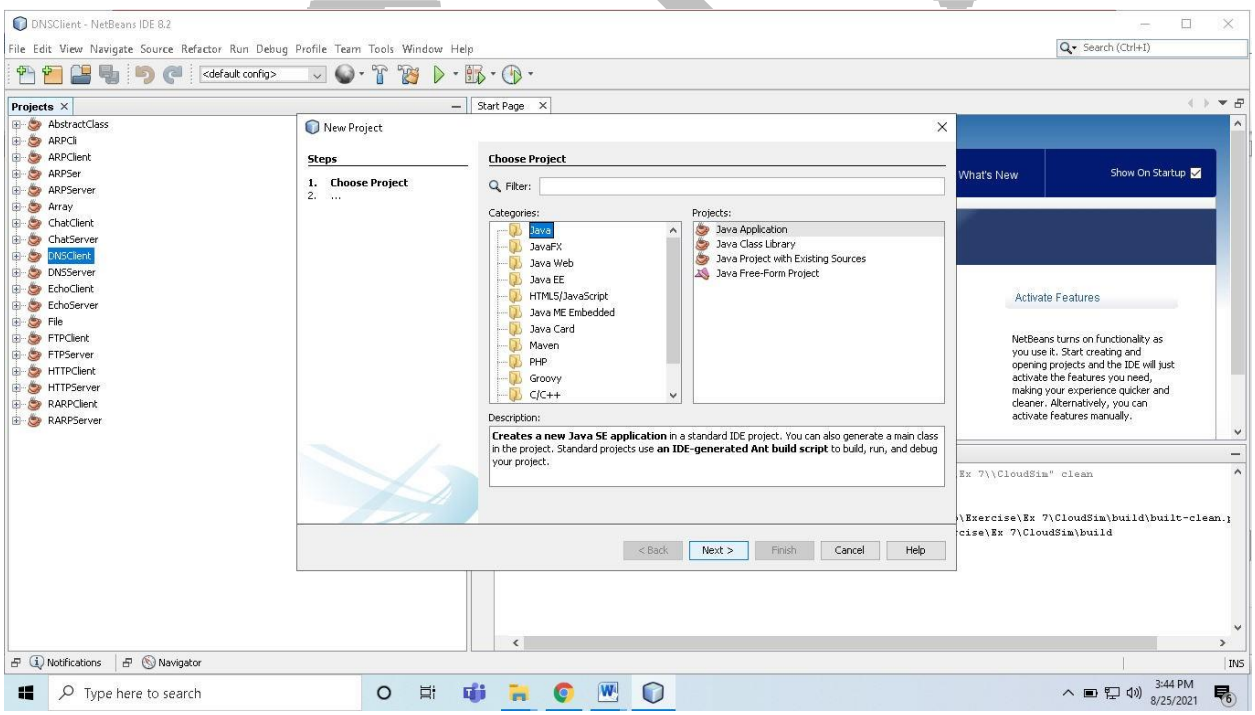
2. Download CloudSim 4.0 (jar and source code), extract and place it in respective folders.

Link: https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-4.0

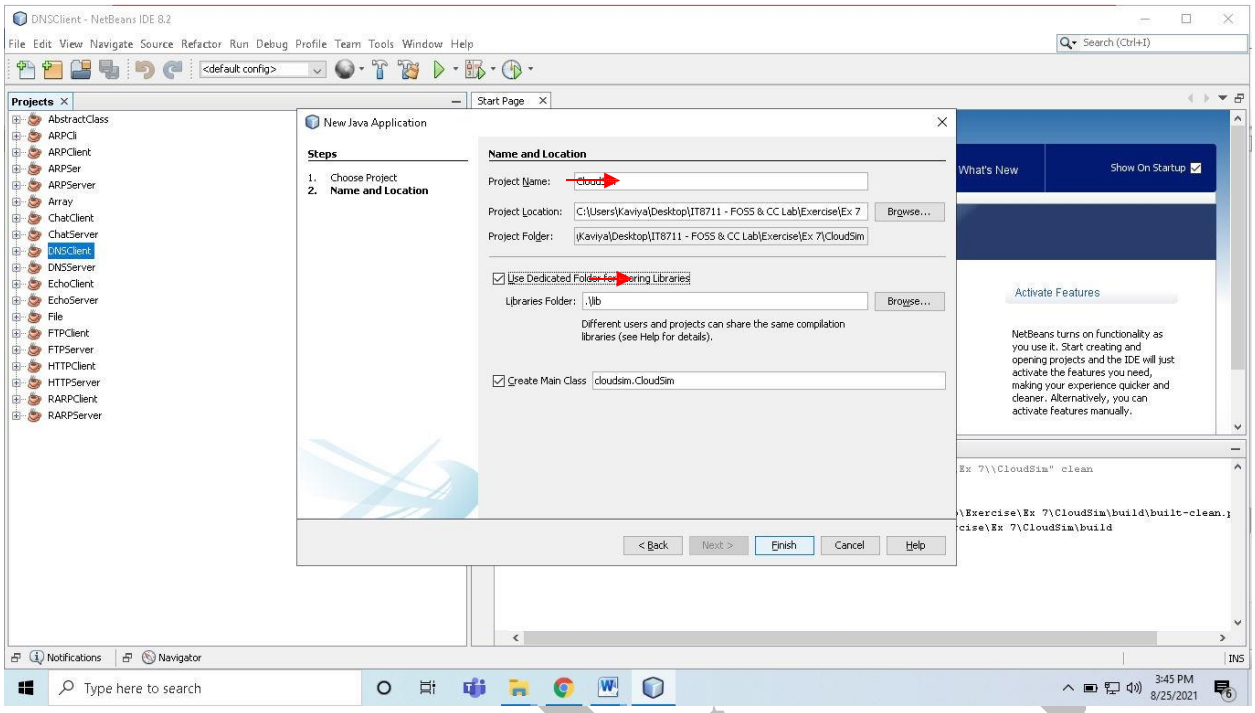


3. Open NetBeans & Create a project “CloudSim”

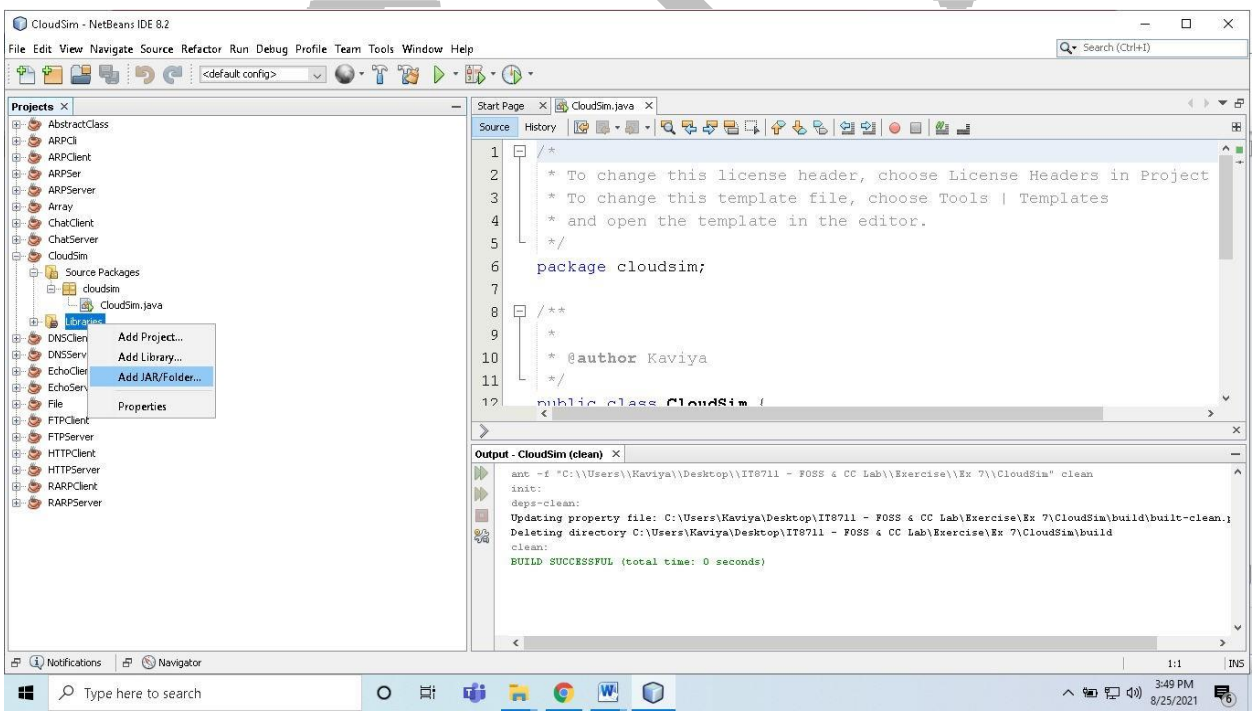


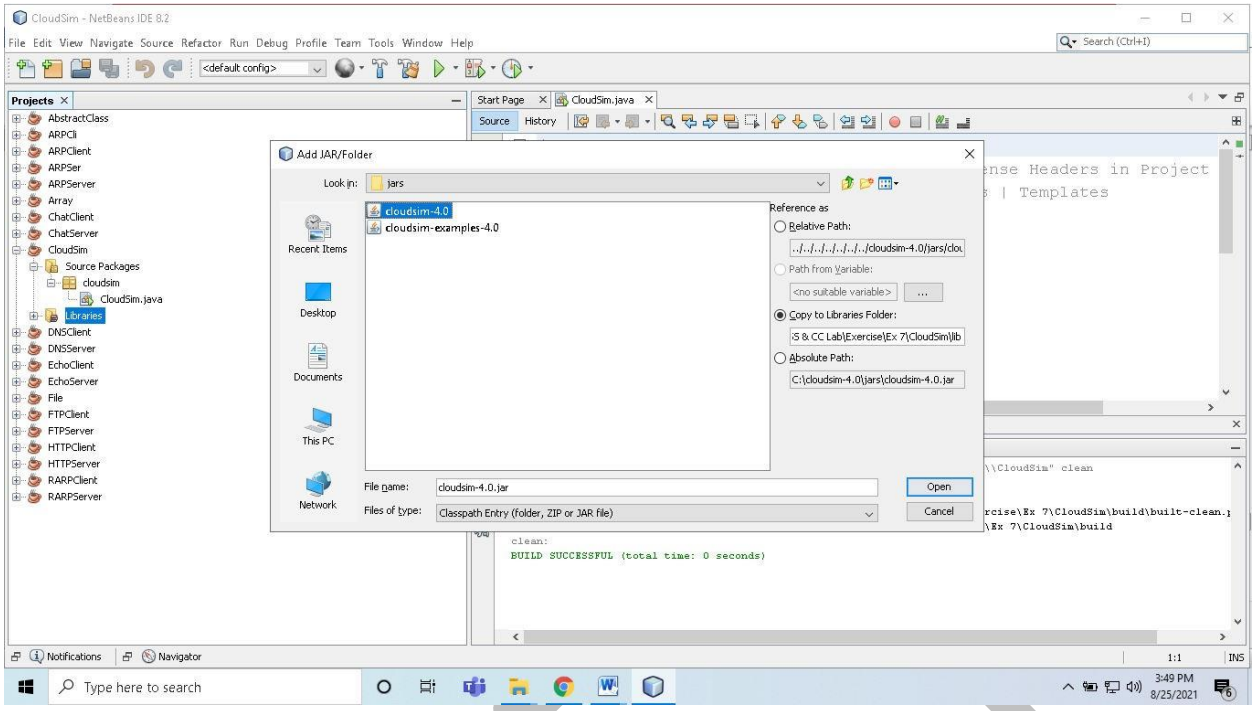


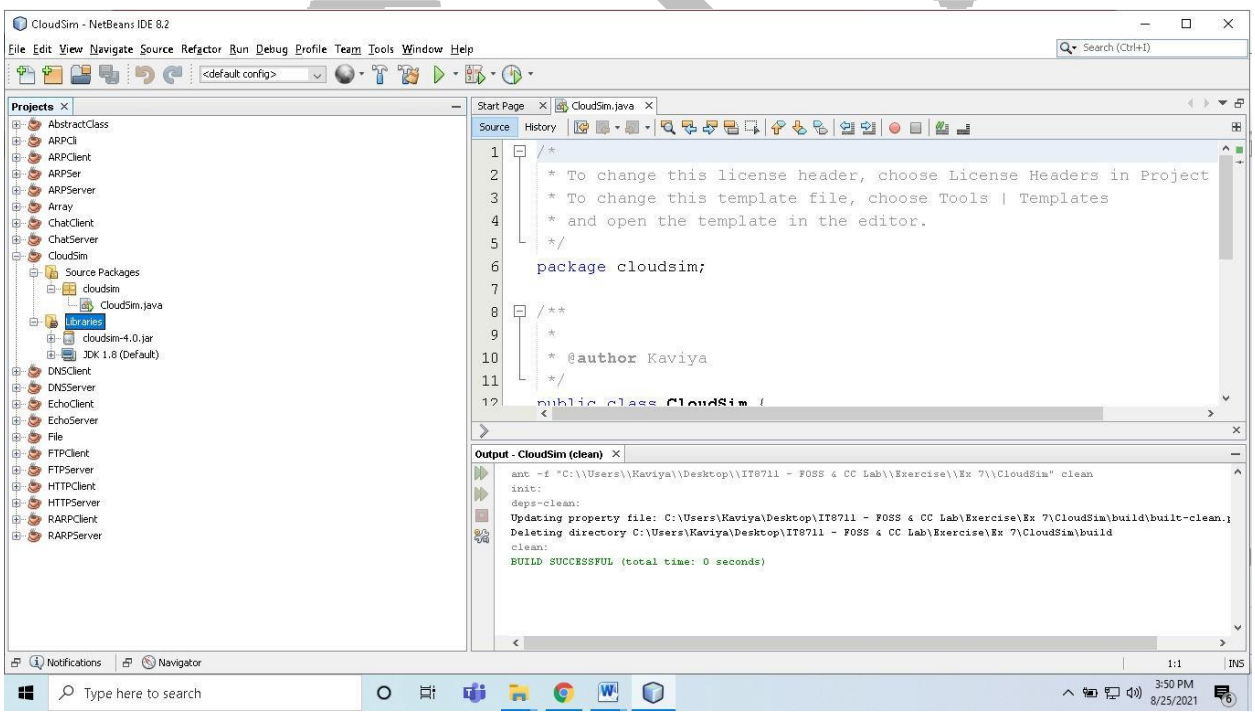
Check “Use dedicated folder for Storing Libraries” → Click “Finish”



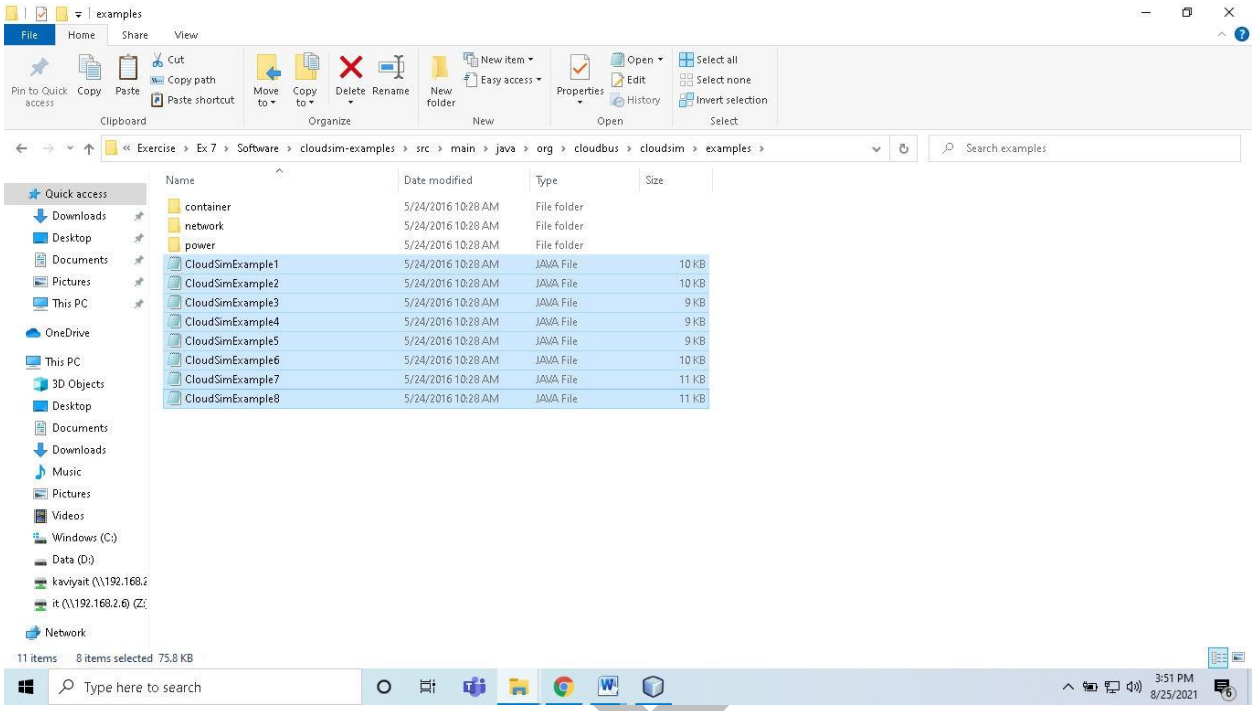
4. Add jar file “cloudsim-4.0”

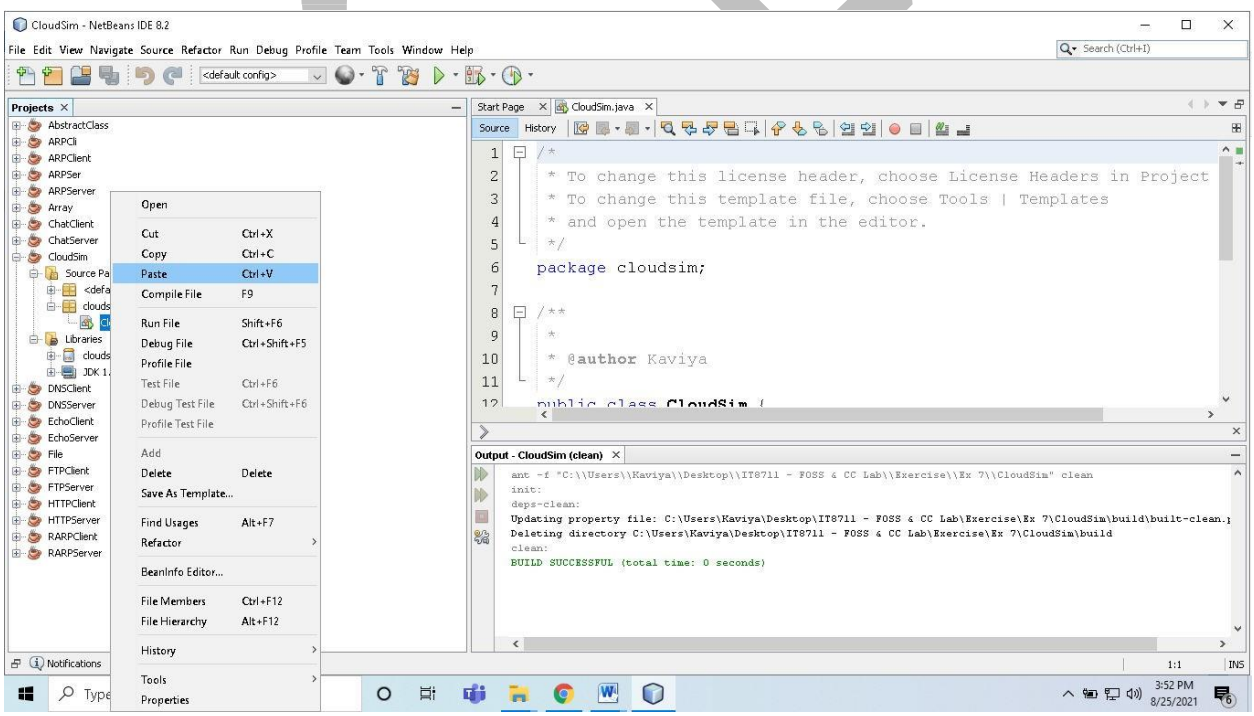


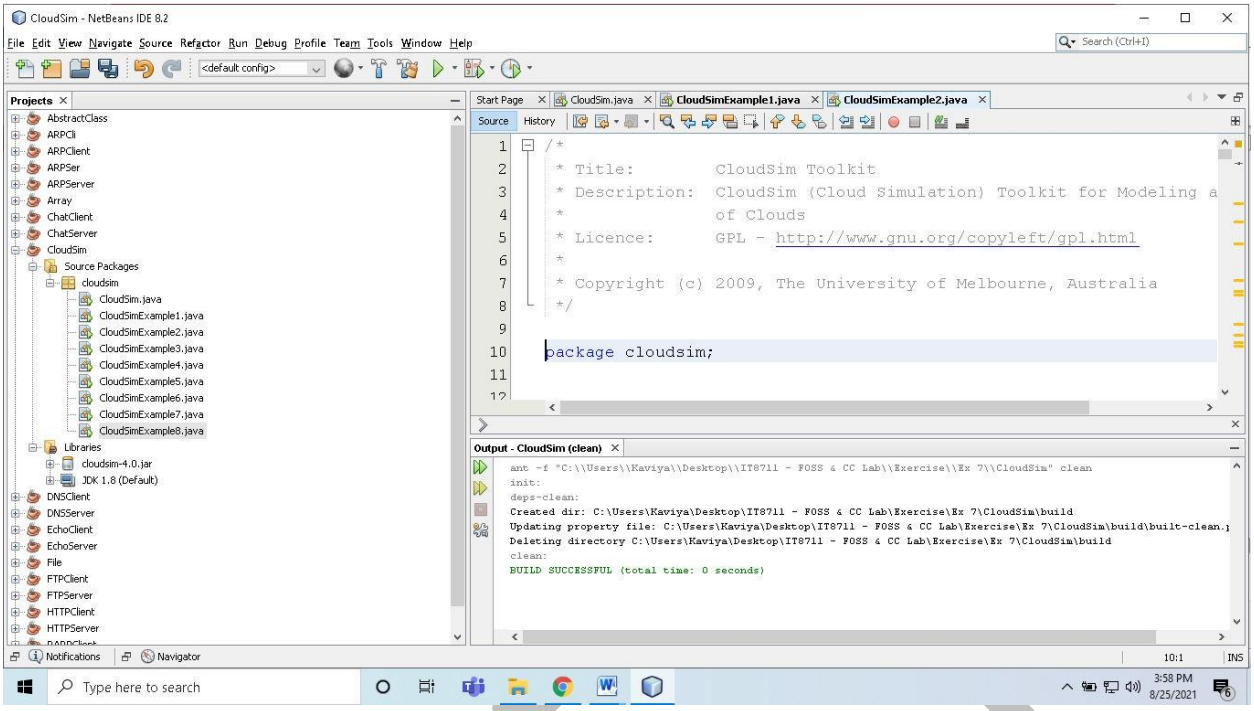

CloudSim - NetBeans IDE 8.2
Add JAR/Folder
Look in: jars
cloudsim-4.0
cloudsim-examples-4.0
Recent Items
Desktop
Documents
This PC
Network
File name: cloudsim-4.0.jar
Files of type: Classpath Entry (folder, ZIP or JAR file)
Open
Cancel
Reference as
Relative Path:
Path from Variable:
Copy to Libraries Folder:
Absolute Path:
C:\cloudsim-4.0\jars\cloudsim-4.0.jar
BUILD SUCCESSFUL (total time: 0 seconds)

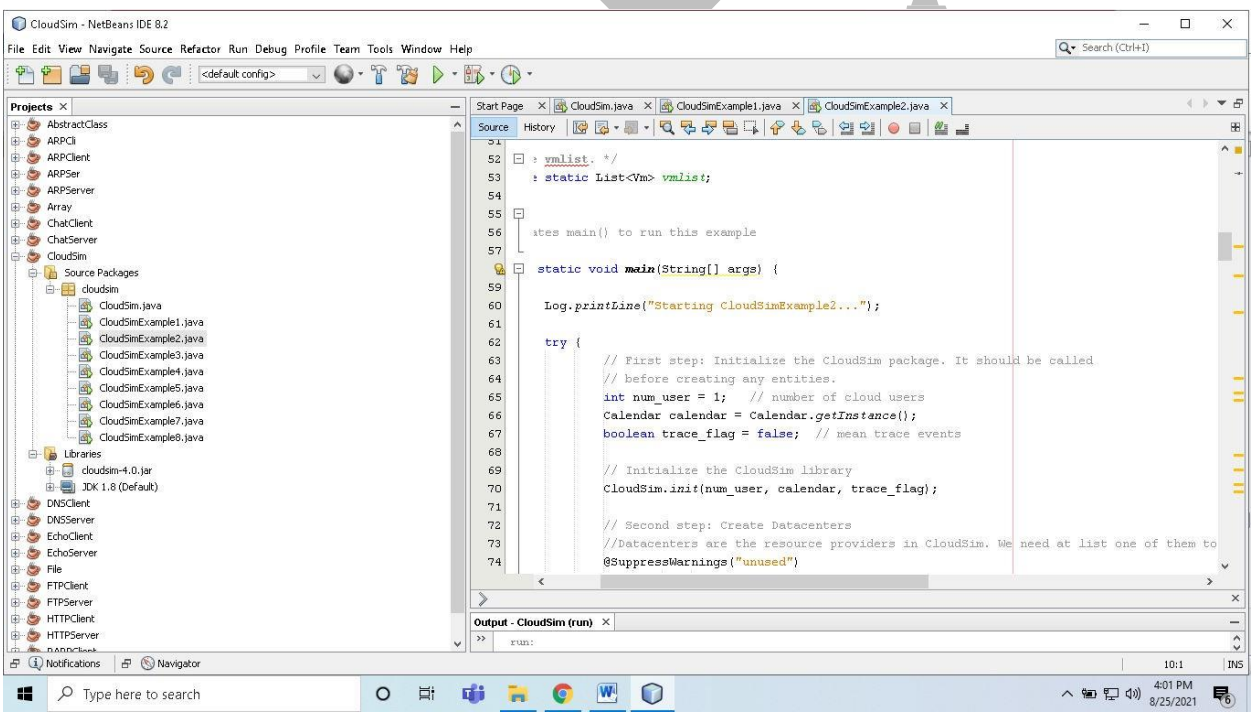

CloudSim - NetBeans IDE 8.2
package cloudsim;
* @author Kaviya
Output - CloudSim (clean)
Updating property file: C:\Users\Kaviya\Desktop\IT8711 - FOSS & CC Lab\Exercise\Ex 7\CloudSim\build\built-clean.
Deleting directory C:\Users\Kaviya\Desktop\IT8711 - FOSS & CC Lab\Exercise\Ex 7\CloudSim\build
BUILD SUCCESSFUL (total time: 0 seconds)

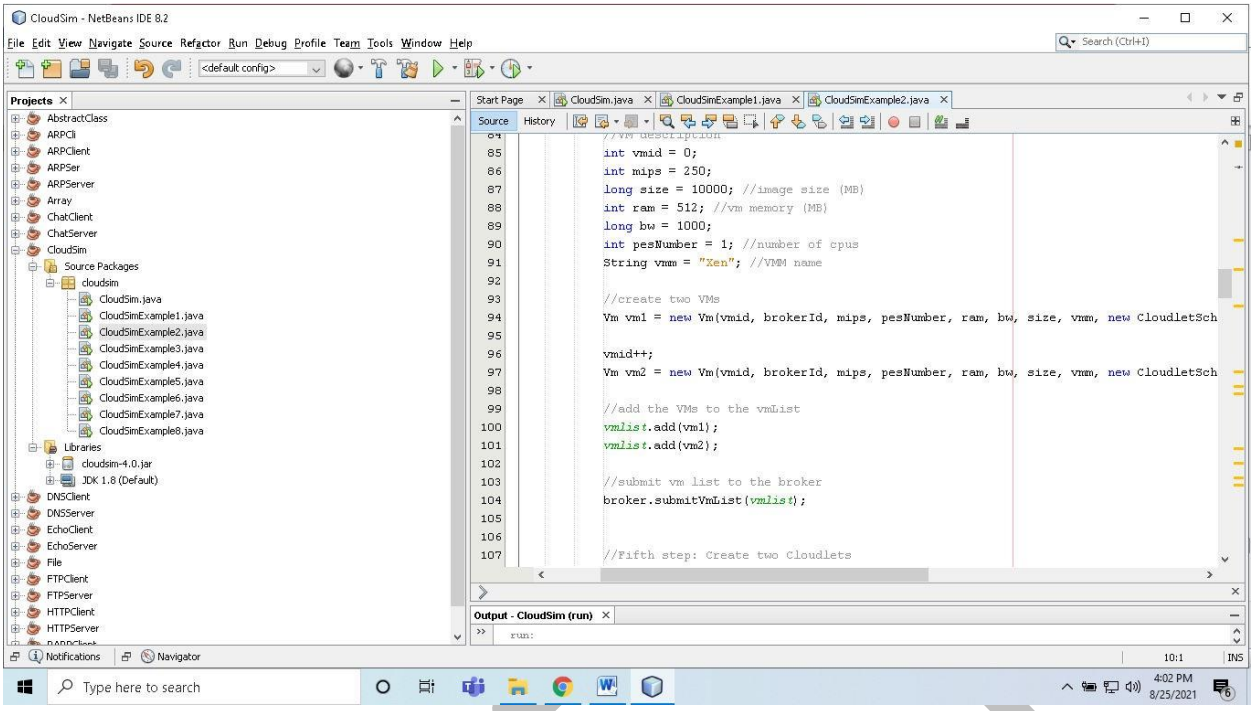
5. Add “cloudsim examples” in CloudSim Project

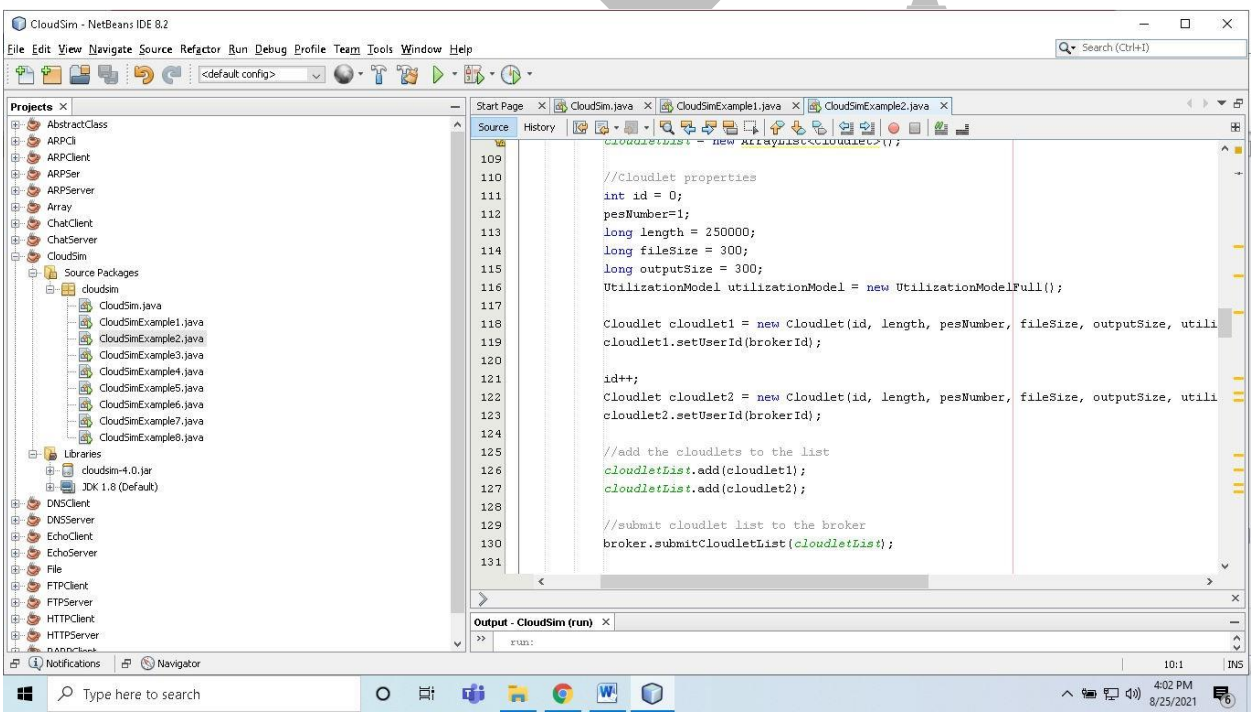


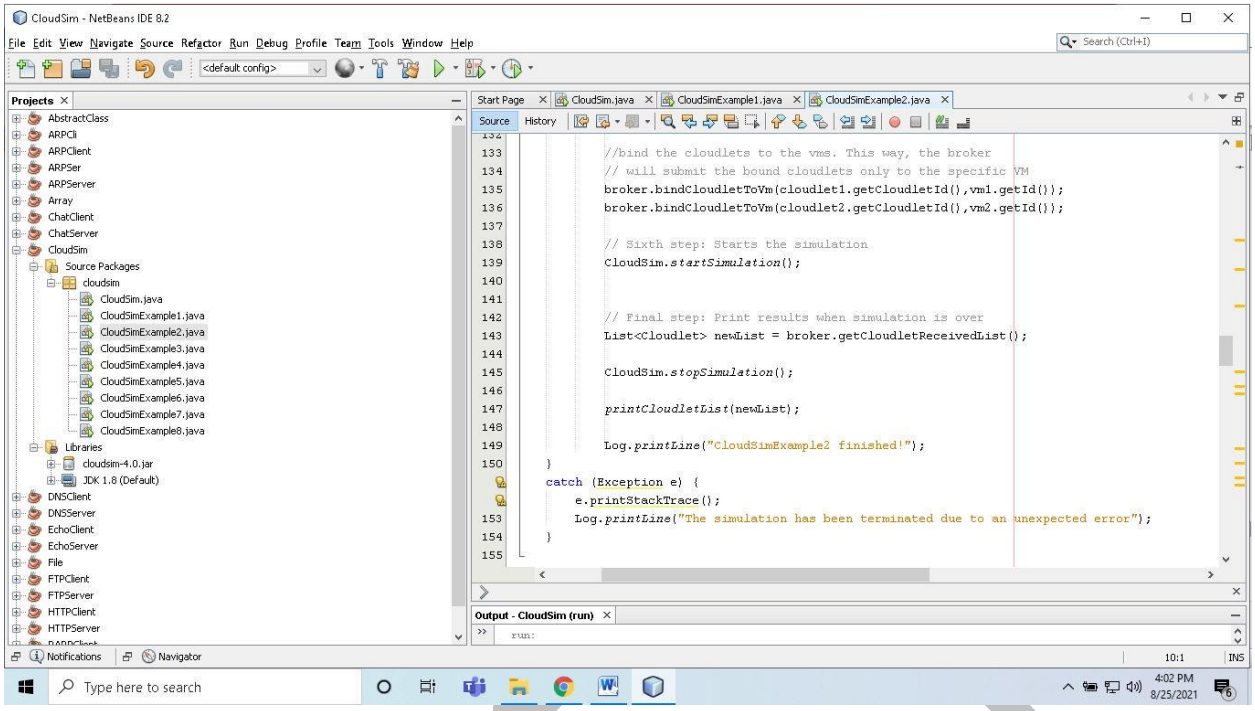



CloudSim - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Start Page
CloudSim.java
CloudSimExample1.java
CloudSimExample2.java
Source
History
1 /*
2 * Title: CloudSim Toolkit
3 * Description: CloudSim (Cloud Simulation) Toolkit for Modeling a
4 * of Clouds
5 * Licence: GPL - http://www.gnu.org/copyleft/gpl.html
6 *
7 * Copyright (c) 2009, The University of Melbourne, Australia
8 */
9
10 package cloudsim;
11
Output - CloudSim (clean)
ant -f "C:\\Users\\Kaviya\\Desktop\\IT8711 - FOSS & CC Lab\\Exercise\\Ex 7\\CloudSim" clean
init:
deps-clean:
Created dir: C:\Users\Kaviya\Desktop\IT8711 - FOSS & CC Lab\Exercise\Ex 7\CloudSim\build
Updating property file: C:\Users\Kaviya\Desktop\IT8711 - FOSS & CC Lab\Exercise\Ex 7\CloudSim\build\built-clean.
Deleting directory C:\Users\Kaviya\Desktop\IT8711 - FOSS & CC Lab\Exercise\Ex 7\CloudSim\build
clean:
BUILD SUCCESSFUL (total time: 0 seconds)
Notifications
Navigator
10:1
INS
Type here to search
3:58 PM
8/25/2021

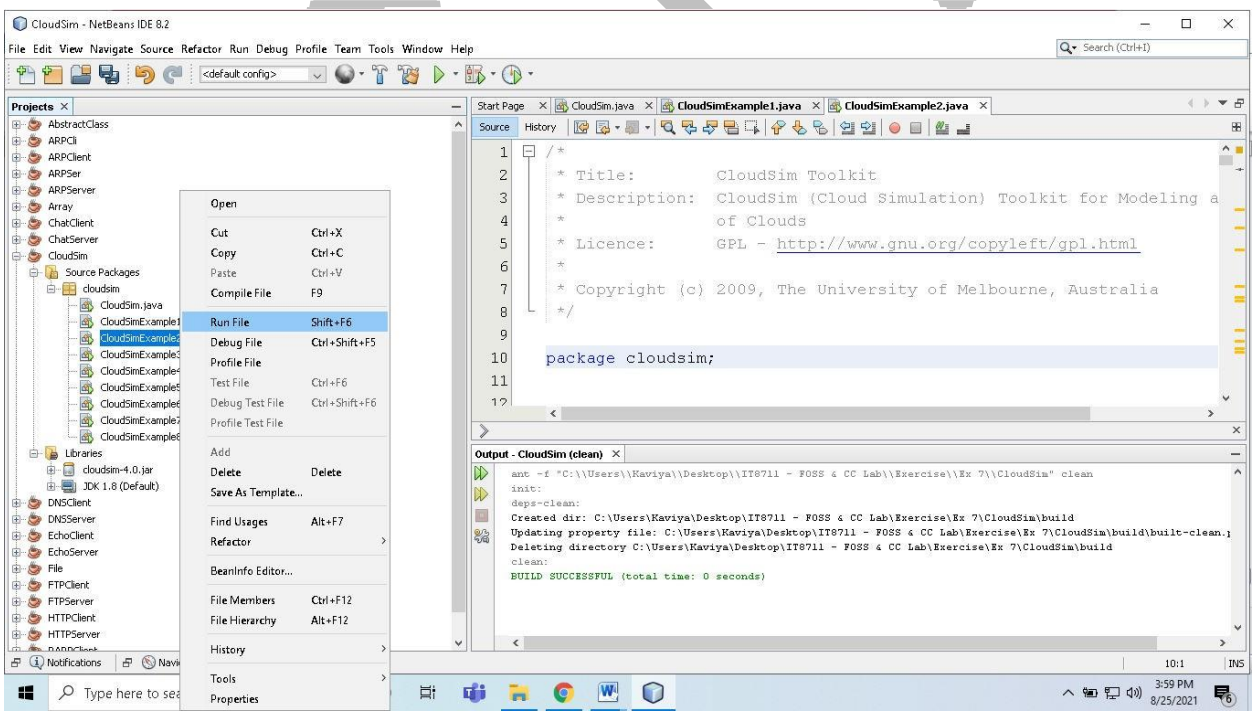

CloudSim - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Start Page
CloudSim.java
CloudSimExample1.java
CloudSimExample2.java
Source
History
52 * vmlist. */
53 static List<Vm> vmlist;
54
55
56 ates main() to run this example
57
58 static void main(String[] args) {
59
60 Log.printLine("Starting CloudSimExample2...");
61
62 try {
63 // First step: Initialize the CloudSim package. It should be called
64 // before creating any entities.
65 int num_user = 1; // number of cloud users
66 Calendar calendar = Calendar.getInstance();
67 boolean trace_flag = false; // mean trace events
68
69 // Initialize the CloudSim library
70 CloudSim.init(num_user, calendar, trace_flag);
71
72 // Second step: Create Datacenters
73 //Datacenters are the resource providers in CloudSim. We need at list one of them to
74 @SuppressWarnings("unused")
Output - CloudSim (run)
run:
Notifications
Navigator
10:1
INS
Type here to search
4:01 PM
8/25/2021


CloudSim - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Start Page
CloudSim.java
CloudSimExample1.java
CloudSimExample2.java
Source
History
int vmid = 0;
int mips = 250;
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 two VMs
Vm vm1 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new CloudletSch
vmid++;
Vm vm2 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new CloudletSch
//add the VMs to the vmList
vmlist.add(vm1);
vmlist.add(vm2);
//submit vm list to the broker
broker.submitVmList(vmlist);
//Fifth step: Create two Cloudlets
Output - CloudSim (run)
run:
Notifications
Navigator
10:1
INS
Type here to search
4:02 PM
8/25/2021

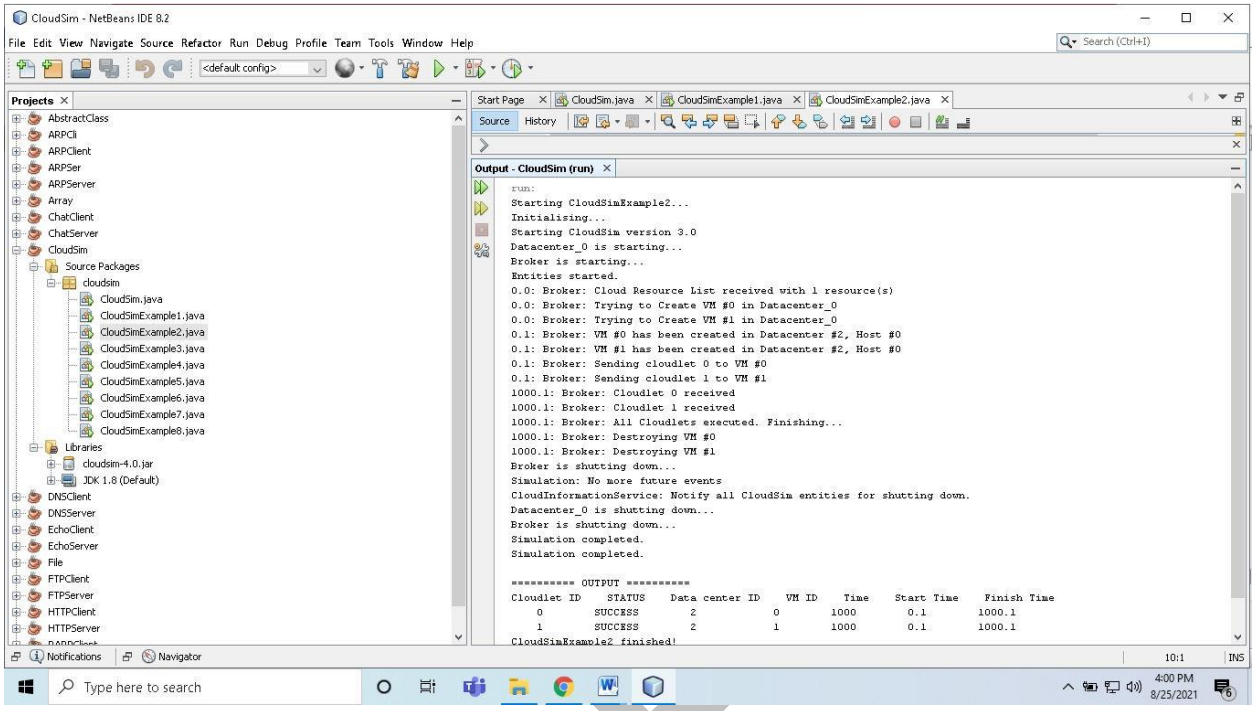

CloudSim - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Start Page
CloudSim.java
CloudSimExample1.java
CloudSimExample2.java
Source
History
//Cloudlet properties
int id = 0;
pesNumber=1;
long length = 250000;
long fileSize = 300;
long outputSize = 300;
UtilizationModel utilizationModel = new UtilizationModelFull();
Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize, outputSize, utili
cloudlet1.setUserId(brokerId);
id++;
Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize, outputSize, utili
cloudlet2.setUserId(brokerId);
//add the cloudlets to the list
cloudletList.add(cloudlet1);
cloudletList.add(cloudlet2);
//submit cloudlet list to the broker
broker.submitCloudletList(cloudletList);
Output - CloudSim (run)
run:
Notifications
Navigator
10:1
INS
Type here to search
4:02 PM
8/25/2021



6. Run the CloudSim examples



7. Output for CloudSimExample2.java



VIVA QUESTIONS

1. What is Clouds?

2. CloudSim is written in .

3. _ class represents any task that is run on a VM.

4. _ is responsible for functioning of VMs, including VM creation, management, destruction, and submission of cloudlets to the VM.

5. _ is the default VM allocation policy.