



Practical -11

Installation of cloud sim.

What is cloudSim ?

- ❑ **CloudSim** is an open-source framework, which is used to simulate cloud computing infrastructure and services. It is developed by the CLOUDS Lab organization and is written entirely in Java. It is used for modelling and simulating a cloud computing environment as a means for evaluating a hypothesis prior to software development in order to reproduce tests and results.

Why use CloudSim?

- *Open source* and *free of cost*, so it favours researchers/developers working in the field.
- Easy to download and set-up.
- It is more *generalized* and *extensible* to support modelling and experimentation.
- Does not require any high-specs computer to work on.
- Provides *pre-defined allocation policies* and *utilization models* for managing resources, and allows implementation of user-defined algorithms as well.
- The documentation provides *pre-coded examples* for new developers to get familiar with the basic classes and functions.
- Tackle bottlenecks before deployment to reduce risk, lower costs, increase performance, and raise revenue.

Features of CloudSim:

- CloudSim provides support for simulation and modelling of:
- Large scale virtualized Datacenters, servers and hosts.
- Customizable policies for provisioning host to virtual machines.
- Energy-aware computational resources.
- Application containers and federated clouds (joining and management of multiple public clouds).
- Datacenter network topologies and message-passing applications.
- Dynamic insertion of simulation entities with stop and resume of simulation.
- User-defined allocation and provisioning policies.

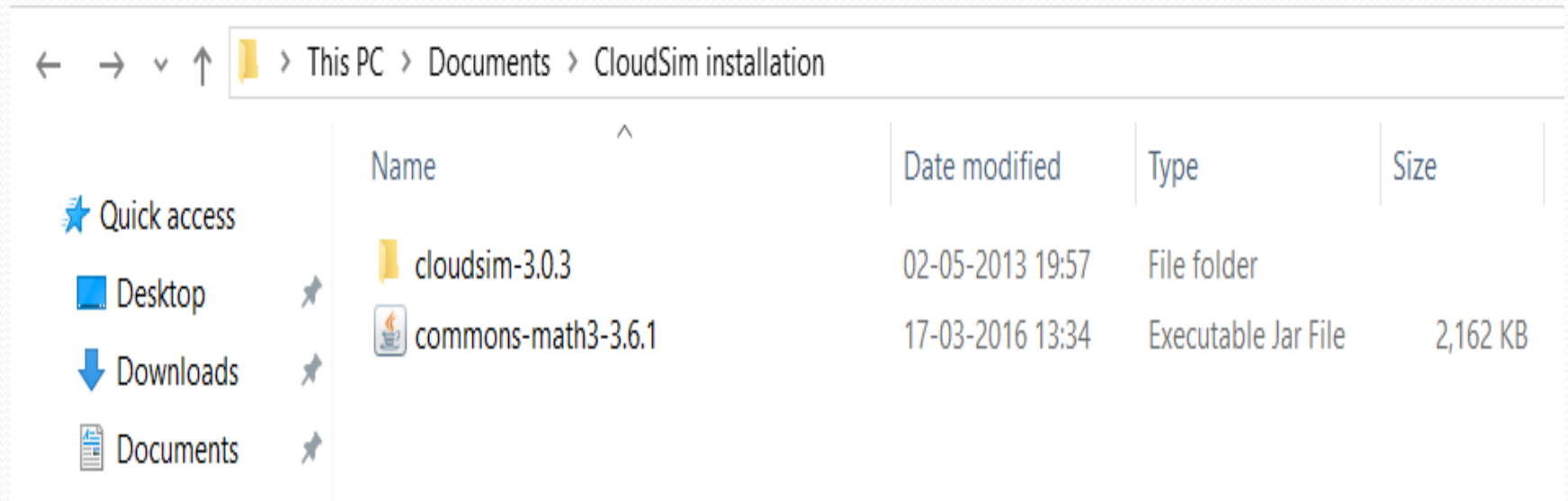
Installation of CloudSim.

- Download JDK AND Eclipse
- **Installation:**
- **Prerequisites:**
- Knowledge of Core Java language features such as OOP and Collections.
<https://www.geeksforgeeks.org/collections-in-java-2/>
- Basics of Cloud Computing.
<https://www.geeksforgeeks.org/cloud-computing/>
- CloudSim is available for download here.
<https://github.com/Cloudslab/cloudsim/releases>.
- For this tutorial, we have downloaded zip file of CloudSim 3.0.3.

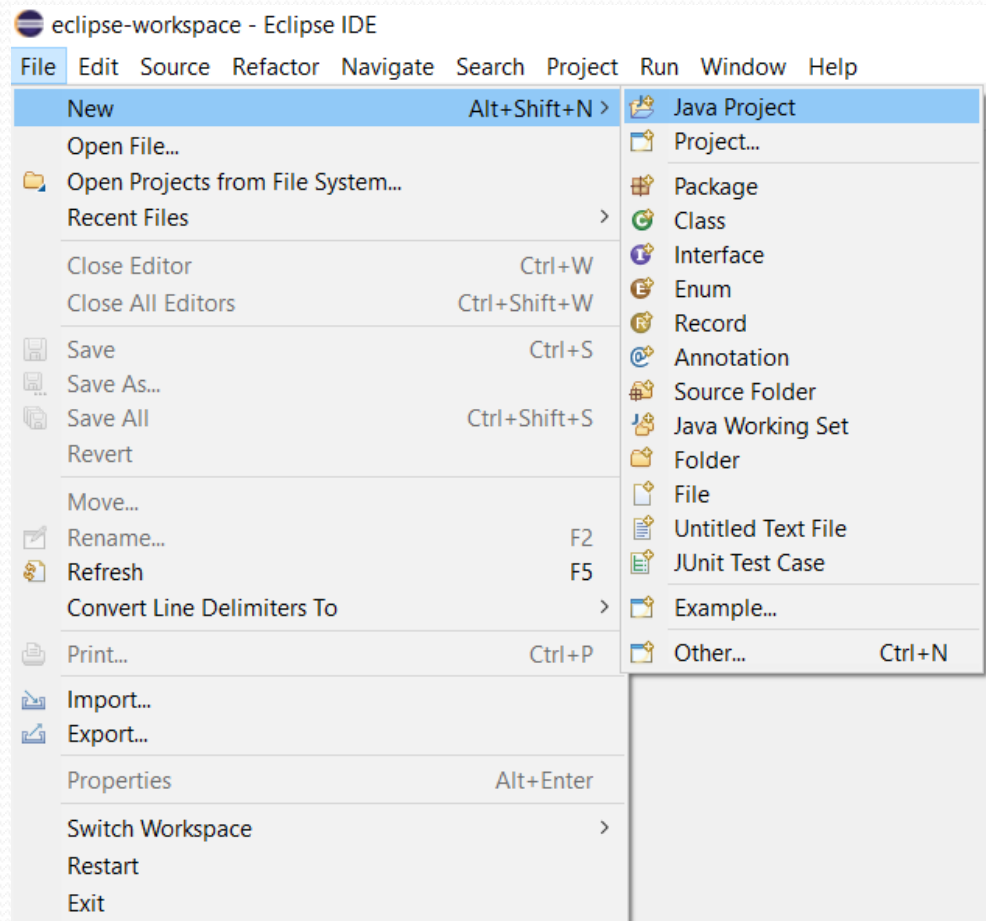
Installation of clousim.

- Note: CloudSim also uses some utilities of Apache's commons-math3 library. Download its Binaries zip file from [here](#).
- https://commons.apache.org/proper/commons-math/download_math.cgi

Step 1: From the zip folder extracts *cloudsim-3.0.3* into a folder. Also, extract the *commons-math3-3.6.1 jar* into the same folder.



Step 2: Open Eclipse IDE and go to *File -> New -> Java Project*.



Step 3: Enter any name for your project and then uncheck the *Use default location* box just under it and click on *Browse*.

New Java Project

Create a Java Project

Enter a location for the project.

Project name: CloudSimTutorial

☐ Use default location

Location: [Browse...](#)

JRE

☒ Use an execution environment JRE: JavaSE-11

☐ Use a project specific JRE: jdk-11.0.9

☐ Use default JRE 'jdk-11.0.9' and workspace compiler preferences [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

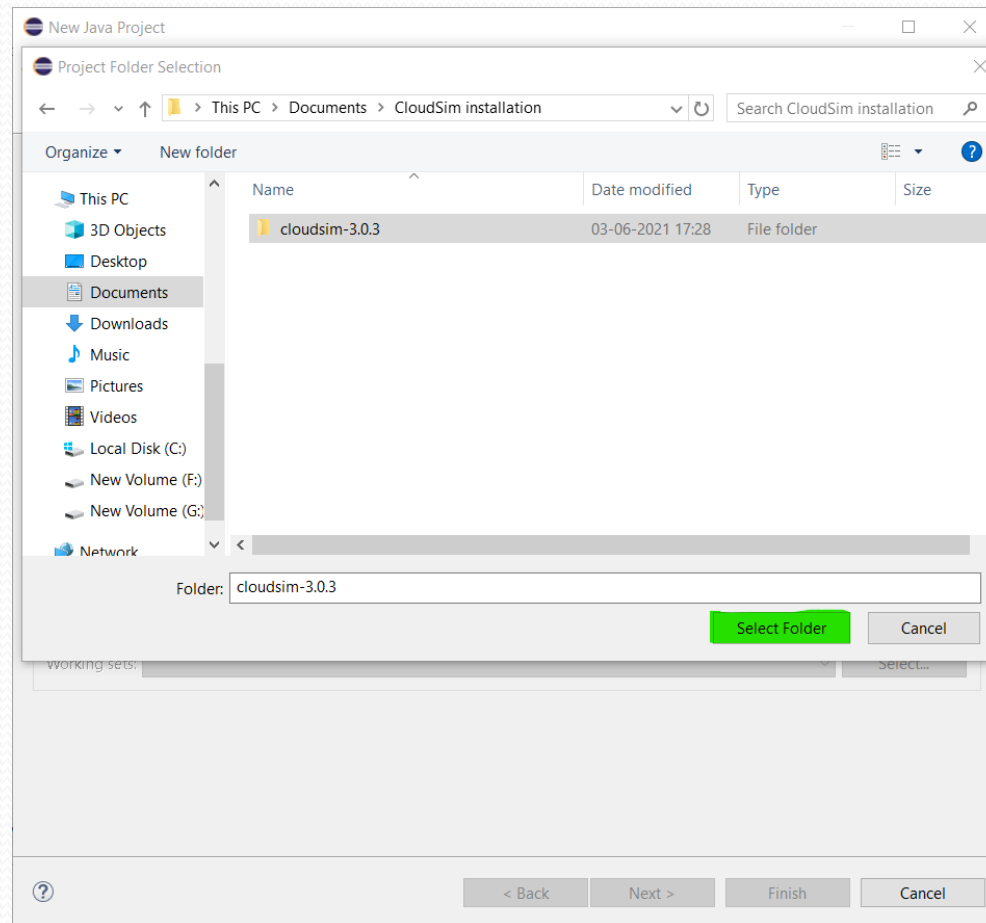
Working sets

☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

Browse to the folder where you extracted your files and select the *cloudsim-3.0.3* folder. Don't click on *Finish* yet, because we need to add a jar file to our project.



- **Step 4** Click *Next* and go to *Libraries* - > *Add External JARs*. Now browse to the same folder where you extracted your *commons-math3* jar file and *Open* it.






Java Settings

Define the Java build settings.



Source Projects Libraries Order and Export Module Dependencies

JARs and class folders on the build path:

- >  cloudsim-3.0.3.jar - CloudSimTutorial/jars
- >  cloudsim-3.0.3-sources.jar - CloudSimTutorial/jars
- >  cloudsim-examples-3.0.3.jar - CloudSimTutorial/jars
- >  cloudsim-examples-3.0.3-sources.jar - CloudSimTutorial/jars
- >  JRE System Library [jdk-11.0.9]

Add JARs...

Add External JARs...

Add Variable...

Add Library...

Add Class Folder...

Add External Class Folder...

Edit...

Remove

Migrate JAR File...

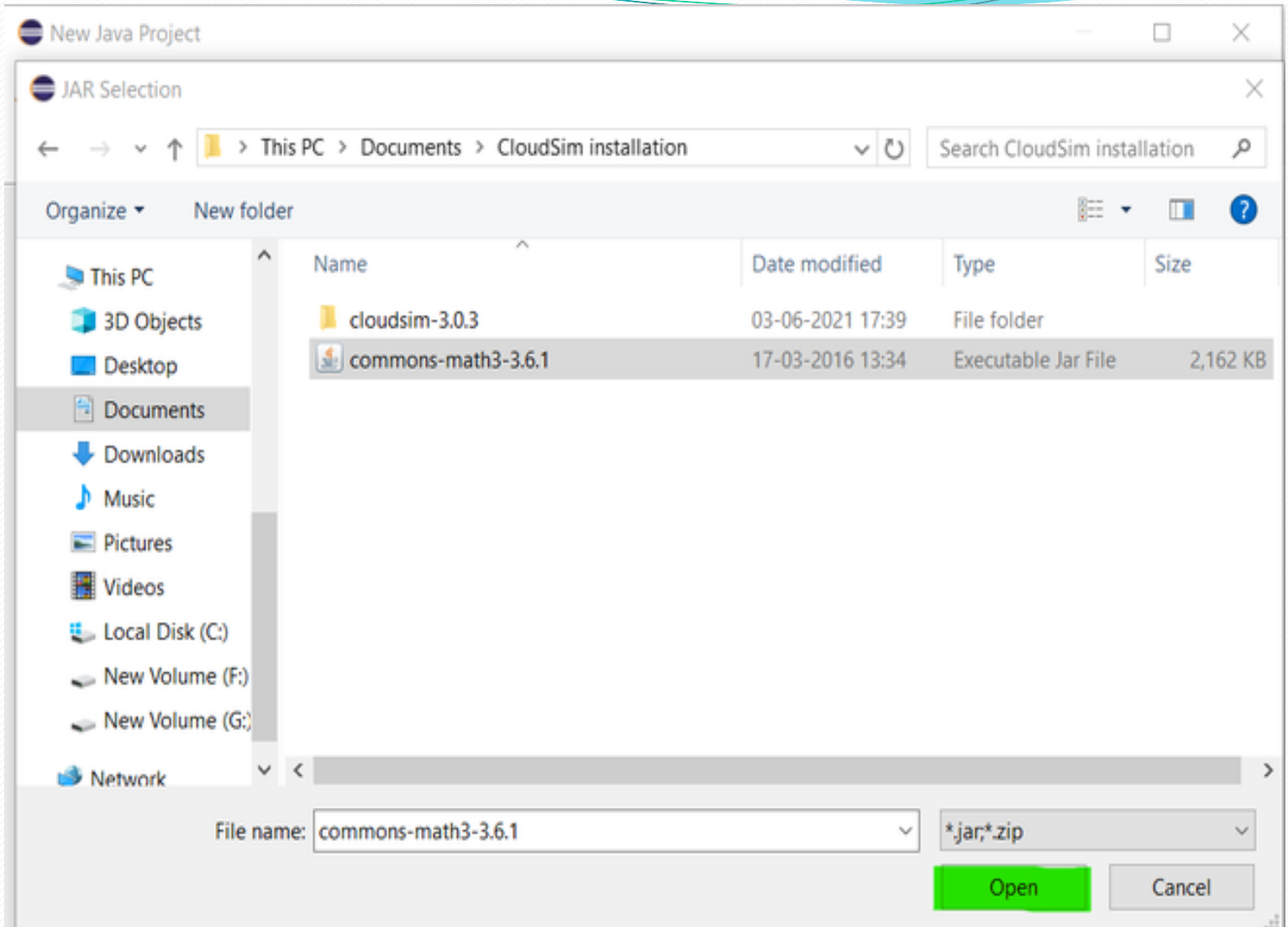



< Back

Next >

Finish

Cancel

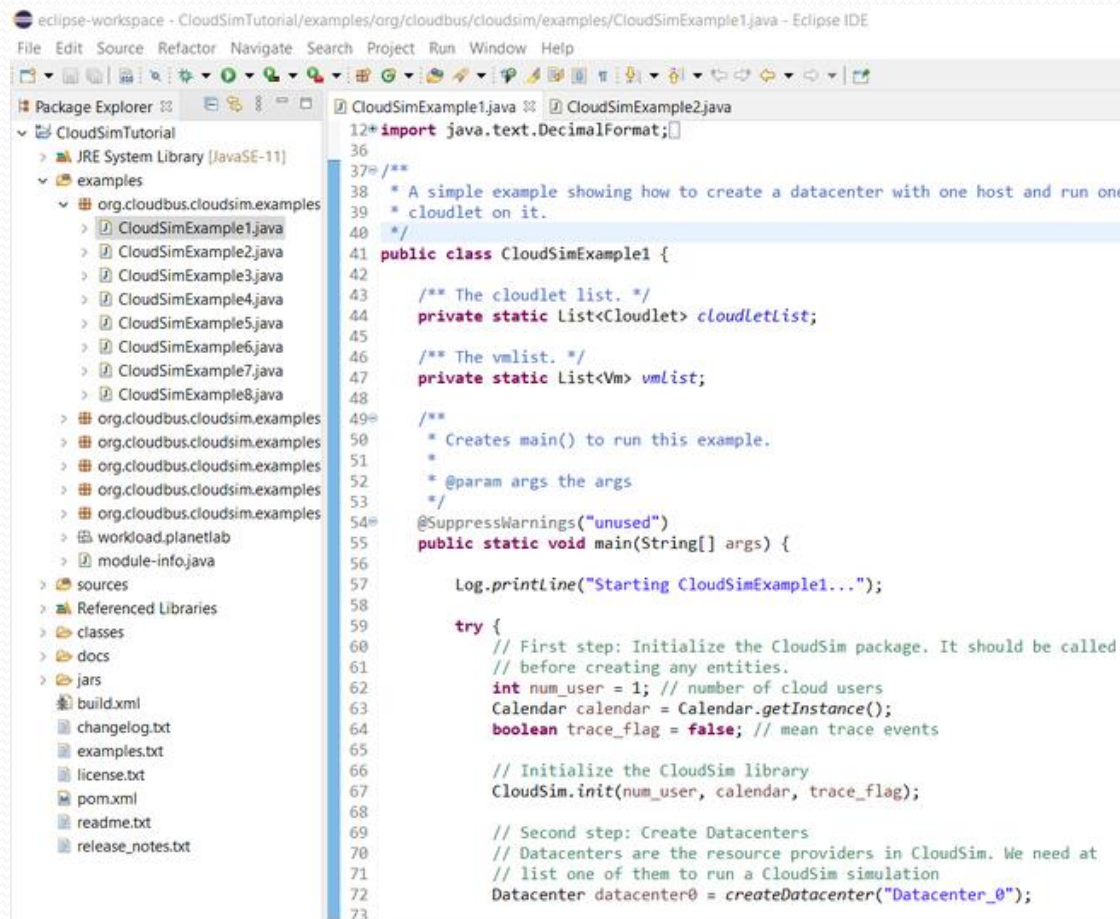




Step 5 - Finally click on *Finish* and wait for the project to build. After the project has been built, from the *Project Explorer* you can click on your project and from the dropdown go-to *examples* -> *org.cloudbus.cloudsim.examples* where you can find pre-written sample codes and try to run them.

Step 5 Finally click on *Finish* and wait for the project to build. After the project has been built, from the *Project Explorer* you can click on your project and

from the dropdown go-to *examples* -> *org.cloudbus.cloudsim.examples* where you can find pre-written sample codes and try to run them.



The screenshot shows the Eclipse IDE interface. On the left, the 'Package Explorer' displays the project structure. The 'CloudSimTutorial' project is expanded, showing a package 'org.cloudbus.cloudsim.examples' which contains several Java files, including 'CloudSimExample1.java'. The main editor on the right shows the code for 'CloudSimExample1.java'. The code includes imports, a class definition, and a main method that initializes the CloudSim environment and creates a datacenter.

```
12*import java.text.DecimalFormat;
36
37/**
38 * A simple example showing how to create a datacenter with one host and run one
39 * cloudlet on it.
40 */
41public class CloudSimExample1 {
42
43    /** The cloudlet list. */
44    private static List<Cloudlet> cloudletList;
45
46    /** The vmlist. */
47    private static List<Vm> vmlist;
48
49    /**
50     * Creates main() to run this example.
51     *
52     * @param args the args
53     */
54    @SuppressWarnings("unused")
55    public static void main(String[] args) {
56
57        Log.println("Starting CloudSimExample1...");
58
59        try {
60            // First step: Initialize the CloudSim package. It should be called
61            // before creating any entities.
62            int num_user = 1; // number of cloud users
63            Calendar calendar = Calendar.getInstance();
64            boolean trace_flag = false; // mean trace events
65
66            // Initialize the CloudSim library
67            CloudSim.init(num_user, calendar, trace_flag);
68
69            // Second step: Create Datacenters
70            // Datacenters are the resource providers in CloudSim. We need at
71            // list one of them to run a CloudSim simulation
72            Datacenter datacenter0 = createDatacenter("Datacenter_0");
73
```

Download the examples.

- **References:**
- <http://www.cloudbus.org/cloudsim/>

Some of the most common classes used during simulation are:

- Some of the most common classes used during simulation are:
- **Datacenter:** used for modelling the foundational hardware equipment of any cloud environment, that is the Datacenter. This class provides methods to specify the functional requirements of the Datacenter as well as methods to set the allocation policies of the VMs etc.
- **Host:** this class executes actions related to management of virtual machines. It also defines policies for provisioning memory and bandwidth to the virtual machines, as well as allocating CPU cores to the virtual machines.
- **VM:** this class represents a virtual machine by providing data members defining a VM's bandwidth, RAM, mips (million instructions per second), size while also providing setter and getter methods for these parameters.
- **Cloudlet:** a cloudlet class represents any task that is run on a VM, like a processing task, or a memory access task, or a file updating task etc. It stores parameters defining the characteristics of a task such as its length, size, mi (million instructions) and provides methods similarly to VM class while also providing methods that define a task's execution time, status, cost and history.
- **DatacenterBroker:** is an entity acting on behalf of the user/customer. It is responsible for functioning of VMs, including VM creation, management, destruction and submission of cloudlets to the VM.
- **CloudSim:** this is the class responsible for initializing and starting the simulation environment after all the necessary cloud entities have been defined and later stopping after all the entities have been destroyed.

Features of CloudSim:

- CloudSim provides support for simulation and modelling of:
- Large scale virtualized Datacenters, servers and hosts.
- Customizable policies for provisioning host to virtual machines.
- Energy-aware computational resources.
- Application containers and federated clouds (joining and management of multiple public clouds).
- Datacenter network topologies and message-passing applications.
- Dynamic insertion of simulation entities with stop and resume of simulation.
- User-defined allocation and provisioning policies.



Thanks