# CloudSim Simulation for VM Allocation and Cloudlet Execution

**Step-by-Step Implementation Process**

1. **Set Up the Environment:**
    - Install **Java JDK**.
    - Install **Eclipse IDE**.
    - Download and set up the **CloudSim 3.0.3** library.
    - Configure CloudSim .jar files in Eclipse's build path.
2. **Create Java Project and Class:**
    - Create a new Java project in Eclipse (e.g., CloudSimProject).
    - Create a class named CloudSimExample1.
3. **Import Required Packages:**
    - Import essential CloudSim classes like Cloudlet, Vm, Datacenter, etc.
4. **Initialize CloudSim:**
    - Use CloudSim.init() with appropriate parameters like number of users, time zone, and trace flag.
5. **Create Datacenter:**
    - Define one datacenter with:
        - One host
        - 1 PE (Processing Element)
        - RAM, storage, and bandwidth provisioning
    - Use VmSchedulerTimeShared for VM scheduling.
6. **Create Broker:**
    - Instantiate a DatacenterBroker to handle VM and cloudlet submissions.
7. **Create VMs:**
    - Create 4 VMs with different configurations:
        - Varying MIPS, RAM, and size.
    - Add VMs to the VM list and submit to broker.
8. **Create Cloudlets:**
    - Create 8 cloudlets with different lengths and I/O sizes.

- Use UtilizationModelFull for full resource utilization.
- Add cloudlets to cloudlet list and submit to broker.

9. **Bind Cloudlets to VMs:**
    - Bind each cloudlet explicitly to a VM using bindCloudletToVm().
10. **Start Simulation:**
    - Start the simulation using CloudSim.startSimulation().
    - Stop the simulation using CloudSim.stopSimulation().
11. **Display Results:**
    - Retrieve executed cloudlets using broker.getCloudletReceivedList().
    - Print cloudlet execution results (status, VM ID, execution time, etc.).
12. **Output Observations:**
    - All VMs and cloudlets executed successfully.
    - Cloudlets were executed on the VMs they were bound to.
    - The result shows execution time, start and finish time for each cloudlet.

# Code:

```
package cloudsimproject;
import java.text.DecimalFormat;
import java.util.*;
import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.*;

public class CloudSimExample1 {

   private static List<Cloudlet> cloudletList;
   private static List<Vm> vmlist;

   public static void main(String[] args) {
      Log.printLine("Starting CloudSimExample1...");

      try {
         int num_user = 1;
         Calendar calendar = Calendar.getInstance();
         boolean trace_flag = false;
```

```
        CloudSim.init(num_user, calendar, trace_flag);

        Datacenter datacenter0 = createDatacenter("Datacenter_0");
        DatacenterBroker broker = createBroker();
        int brokerId = broker.getId();

        vmlist = new ArrayList<Vm>();
        int vmid = 0;
        int mips = 1000;
        long size = 10000;
        int ram = 512;
        long bw = 1000;
        int pesNumber = 1;
        String vmm = "Xen";

        Vm vm1 = new Vm(vmid++, brokerId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
        Vm vm2 = new Vm(vmid++, brokerId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
        Vm vm3 = new Vm(vmid++, brokerId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
        Vm vm4 = new Vm(vmid++, brokerId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());

        vmlist.add(vm1);
        vmlist.add(vm2);
        vmlist.add(vm3);
        vmlist.add(vm4);
        broker.submitVmList(vmlist);

        cloudletList = new ArrayList<Cloudlet>();
        int id = 0;
        long length = 400000;
        long fileSize = 300;
        long outputSize = 300;
        UtilizationModel utilizationModel = new UtilizationModelFull();

        Cloudlet cloudlet1 = new Cloudlet(id++, length, pesNumber, fileSize, outputSize,
utilizationModel, utilizationModel, utilizationModel);
        cloudlet1.setUserId(brokerId);
        Cloudlet cloudlet2 = new Cloudlet(id++, length * 2, pesNumber, fileSize * 2, outputSize
/ 3, utilizationModel, utilizationModel, utilizationModel);
        cloudlet2.setUserId(brokerId);
        Cloudlet cloudlet3 = new Cloudlet(id++, length / 2, pesNumber, fileSize * 3, outputSize
* 3, utilizationModel, utilizationModel, utilizationModel);
```

```
            cloudlet3.setUserId(brokerId);
            Cloudlet cloudlet4 = new Cloudlet(id++, length / 3, pesNumber, fileSize / 3, outputSize /
2, utilizationModel, utilizationModel, utilizationModel);
            cloudlet4.setUserId(brokerId);
            Cloudlet cloudlet5 = new Cloudlet(id++, length * 3, pesNumber, fileSize / 2, outputSize /
4, utilizationModel, utilizationModel, utilizationModel);
            cloudlet5.setUserId(brokerId);
            Cloudlet cloudlet6 = new Cloudlet(id++, length / 4, pesNumber, fileSize * 4, outputSize
* 4, utilizationModel, utilizationModel, utilizationModel);
            cloudlet6.setUserId(brokerId);
            Cloudlet cloudlet7 = new Cloudlet(id++, length * 4, pesNumber, fileSize, outputSize * 2,
utilizationModel, utilizationModel, utilizationModel);
            cloudlet7.setUserId(brokerId);
            Cloudlet cloudlet8 = new Cloudlet(id++, length, pesNumber, fileSize / 4, outputSize / 3,
utilizationModel, utilizationModel, utilizationModel);
            cloudlet8.setUserId(brokerId);

            cloudletList.add(cloudlet1);
            cloudletList.add(cloudlet2);
            cloudletList.add(cloudlet3);
            cloudletList.add(cloudlet4);
            cloudletList.add(cloudlet5);
            cloudletList.add(cloudlet6);
            cloudletList.add(cloudlet7);
            cloudletList.add(cloudlet8);

            broker.submitCloudletList(cloudletList);

            broker.bindCloudletToVm(cloudlet1.getCloudletId(), vm1.getId());
            broker.bindCloudletToVm(cloudlet2.getCloudletId(), vm2.getId());
            broker.bindCloudletToVm(cloudlet3.getCloudletId(), vm3.getId());
            broker.bindCloudletToVm(cloudlet4.getCloudletId(), vm4.getId());
            broker.bindCloudletToVm(cloudlet5.getCloudletId(), vm1.getId());
            broker.bindCloudletToVm(cloudlet6.getCloudletId(), vm2.getId());
            broker.bindCloudletToVm(cloudlet7.getCloudletId(), vm3.getId());
            broker.bindCloudletToVm(cloudlet8.getCloudletId(), vm4.getId());

            CloudSim.startSimulation();
            CloudSim.stopSimulation();

            List<Cloudlet> newList = broker.getCloudletReceivedList();
            printCloudletList(newList);
            Log.printLine("CloudSimExample1 finished!");

        } catch (Exception e) {
            e.printStackTrace();
```

```
            Log.printLine("Unwanted errors happened");
        }
    }

    private static Datacenter createDatacenter(String name) {
        List<Host> hostList = new ArrayList<Host>();
        List<Pe> peList = new ArrayList<Pe>();

        int mipsPerPe = 1000;

        // Add 4 PEs of 1000 MIPS each
        for (int i = 0; i < 4; i++) {
            peList.add(new Pe(i, new PeProvisionerSimple(mipsPerPe)));
        }

        int hostId = 0;
        int ram = 8192; // RAM in MB
        long storage = 1000000; // Storage in MB
        int bw = 10000;

        hostList.add(new Host(hostId, new RamProvisionerSimple(ram), new
BwProvisionerSimple(bw), storage,
                peList, new VmSchedulerTimeShared(peList)));

        String arch = "x86";
        String os = "Linux";
        String vmm = "Xen";
        double time_zone = 10.0;
        double cost = 3.0;
        double costPerMem = 0.05;
        double costPerStorage = 0.001;
        double costPerBw = 0.0;

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

        try {
            return new Datacenter(name, characteristics, new VmAllocationPolicySimple(hostList),
new LinkedList<>(), 0);
        } catch (Exception e) {
            e.printStackTrace();
        }

        return null;
    }
```

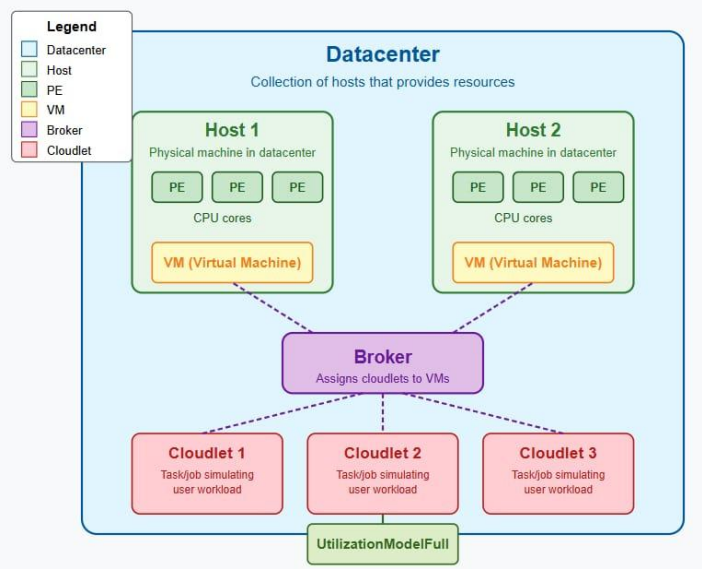
```
    private static DatacenterBroker createBroker() {
        try {
            return new DatacenterBroker("Broker");
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    private static void printCloudletList(List<Cloudlet> list) {
        DecimalFormat dft = new DecimalFormat("###.##");

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

        // Header with fixed width columns
        Log.printLine(String.format("%-12s %-10s %-15s %-8s %-10s %-12s %-12s",
                "Cloudlet ID", "STATUS", "Data center ID", "VM ID", "Time", "Start Time", "Finish
Time"));

        for (Cloudlet cloudlet : list) {
            if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS) {
                Log.printLine(String.format("%-12d %-10s %-15d %-8d %-10s %-12s %-12s",
                        cloudlet.getCloudletId(),
                        "SUCCESS",
                        cloudlet.getResourceId(),
                        cloudlet.getVmId(),
                        dft.format(cloudlet.getActualCPUTime()),
                        dft.format(cloudlet.getExecStartTime()),
                        dft.format(cloudlet.getFinishTime())
                ));
            }
        }
    }
}
```




```
========== OUTPUT ==========
Cloudlet ID  STATUS     Data center ID  VM ID    Time       Start Time   Finish Time
5            SUCCESS    2               1        200        0.1          200.1
3            SUCCESS    2               3        266.66     0.1          266.76
2            SUCCESS    2               2        400        0.1          400.1
7            SUCCESS    2               3        533.33     0.1          533.43
0            SUCCESS    2               0        800        0.1          800.1
1            SUCCESS    2               1        900        0.1          900.1
4            SUCCESS    2               0        1600       0.1          1600.1
6            SUCCESS    2               2        1800       0.1          1800.1
CloudSimExample1 finished!
```