

GURU NANAK DEV ENGINEERING COLLEGE, LUDHIANA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



CLOUD COMPUTING LABORATORY

LPECS-108

Submitted to: Dr. Vivek Thapar

Submitted by: Guneet Kohli
D4 CSE A(1)
URN: 1805172
CRN:1815017

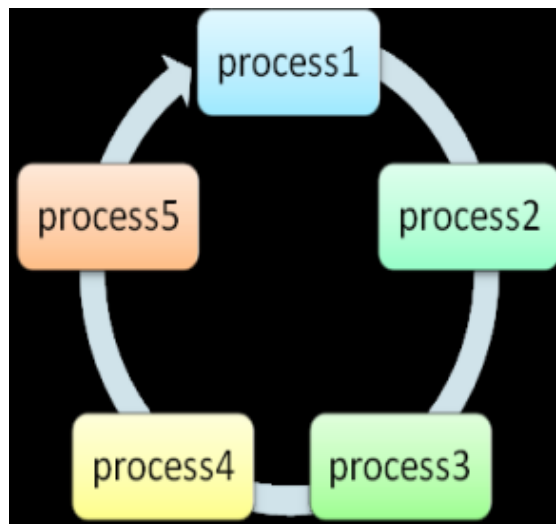
INDEX

S. No.	Topic	Page Number	Date	Remarks
I.1	Use Cloud Analyst Simulation tool and set up simulation with one datacenter and one userbase	2-4	20.09.2021	
I.2	Use Cloud Analyst Simulation tool and set up simulation with multiple userbases in various regions of the world	5-7	27.09.2021	
I.3	Use Cloud Analyst Simulation tool and use closest data center service broker policy and throttled load balancing algorithm	8-10	4.10.2021	
I.4	Use Cloud Analyst Simulation tool and configure the simulation tool to analyze the performance of social networking app	12-13	11.10.2021	
II.1	Use CloudSim Toolkit and create two datacenters with one host each and two cloudlets on them.	14-15	18.10.2021	
II.2	Use CloudSim Toolkit and create datacenter with one host each and two cloudlets on them.	16-17	1.11.2021	
II.3	Use CloudSim Toolkit and create two datacenters with one host each and run cloudlets of two users on them.	18-19	8.11.2021	
II.4	Use CloudSim Toolkit and create a datacenter with one host and a network topology and run one cloudlet on it.	19-21	16.11.2021	
II.5	Use CloudSim Toolkit and create two datacenters with one host each and run cloudlets of two users with network topology on them.	22-23	22.11.2021	
III.1	Examine the architecture and services offered by Google Cloud Platform	24-32	22.11.2021	

Practical I: Use Cloud Analyst Simulation tool and do the following:

1. Set up simulation with one datacenter and one userbase

- a. **Round-Robin Load Balancer:** It is the simplest algorithm that uses the concept of time quantum or slices. Here the time is divided into multiple slices and each node is given a particular time quantum or time interval and in this quantum the node will perform its operations. The resources of the service provider are provided to the client on the basis of this time quantum.



- b. **Closest Data Center Policy:** The datacenter which is having least proximity from the user is selected. Proximity in term of least network latency. If more than one Datacenters having same proximity then it will select datacenter randomly to balance the load.

Simulation Duration:

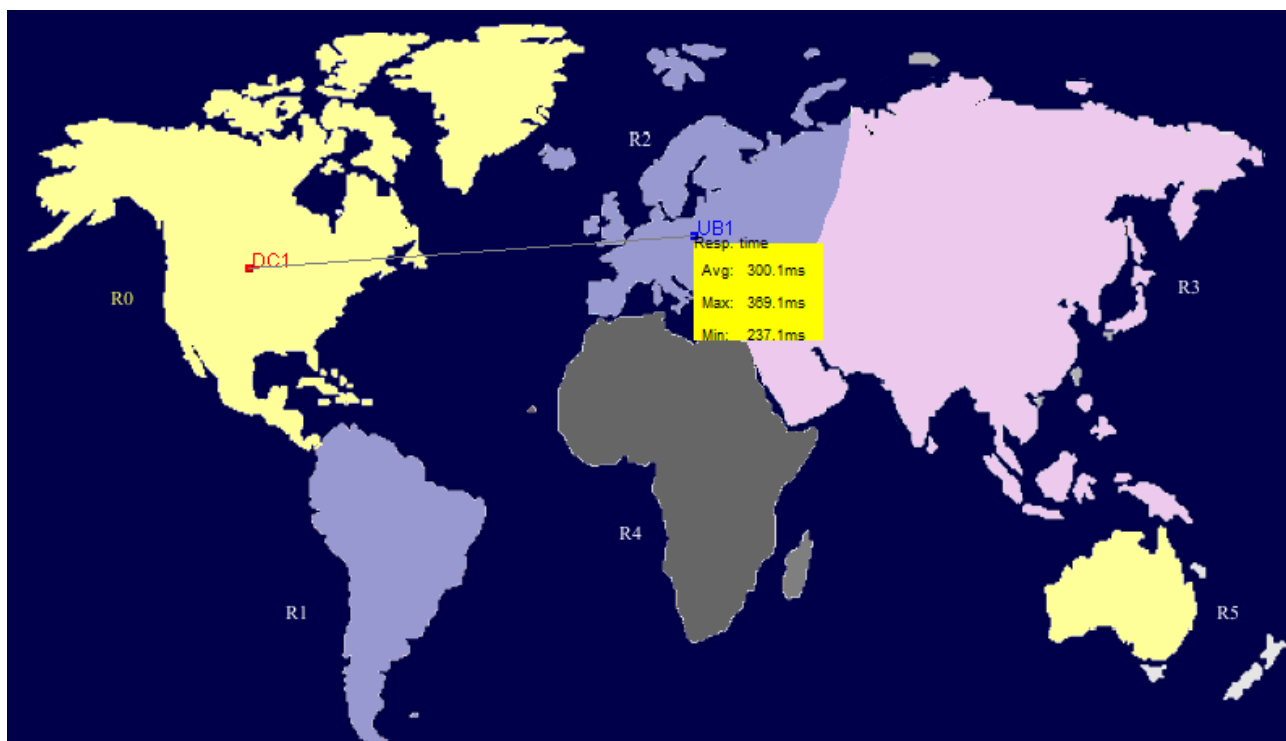
User bases:

Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
UB1	2	60	100	3	9	1000	100

Application Deployment Configuration:

Service Broker Policy:

Data Center	# VMs	Image Size	Memory	BW
DC1	5	10000	512	1000



	Average (ms)	Minimum (ms)	Maximum (ms)	Export Results
Overall Response Time:	300.06	237.06	369.12	
Data Center Processing Time:	0.34	0.02	0.61	
Response Time By Region				
Userbase	Avg (ms)	Min (ms)	Max (ms)	
UB1	300.056	237.059	369.115	

Cost

Total Virtual Machine Cost : \$0.51

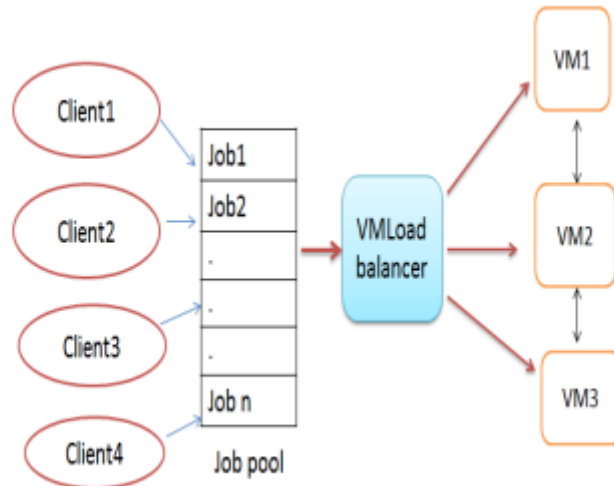
Total Data Transfer Cost : \$0.06

Grand Total : \$0.57

Data Center	VM Cost	Data Transfer Cost	Total
DC1	0.507	0.064	0.571

2. Set up simulation with multiple userbases in various regions of the world

- a. **Equally Spread Current Execution Load:** In Equally Spread Current Execution Algorithm, load balancer makes effort to preserve equal load to all the virtual machines connected with the data center. Load balancer maintains an index table of Virtual machines as well as number of requests currently assigned to the Virtual Machine (VM). If the request comes from the data center to allocate the new VM, it scans the index table for least loaded VM. In case there are more than one VM is found then first identified VM is selected for handling the request of the client/node, the load balancer also returns the VM id to the data center controller



- b. **Optimize Response Time:** First it identifies the closest datacenter but when Closest Datacenter's performance (considers response time) starts degrading it estimates current response time for each datacenter then searches for the datacenter which having least estimated response time. But there may be 50:50 chance for the selection of closest and fastest datacenter. (Again, here random selection)

Main Configuration

Data Center Configuration

Advanced

Simulation Duration:

User bases:

Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
NA	0	60	100	2	6	106537	10653
SA	1	60	100	1	6	105437	10543
EU	2	60	100	19	23	154788	15478
ASIA	3	60	100	17	21	251854	25185
AFRICA	4	60	100	20	0	62284	6228

Add New

Remove

Application Deployment Configuration:

Service Broker Policy:

Data Center	# VMs	Image Size	Memory	BW
DC1	5	10000	512	1000
DC3	5	10000	512	1000
DC2	5	10000	512	1000
DC4	5	10000	512	1000

Add New

Remove

Main Configuration
Data Center Configuration
Advanced

User grouping factor in User Bases:
(Equivalent to number of simultaneous users from a single user base)

10

Request grouping factor in Data Centers:
(Equivalent to number of simultaneous requests a single applicaiton server instance can support.)

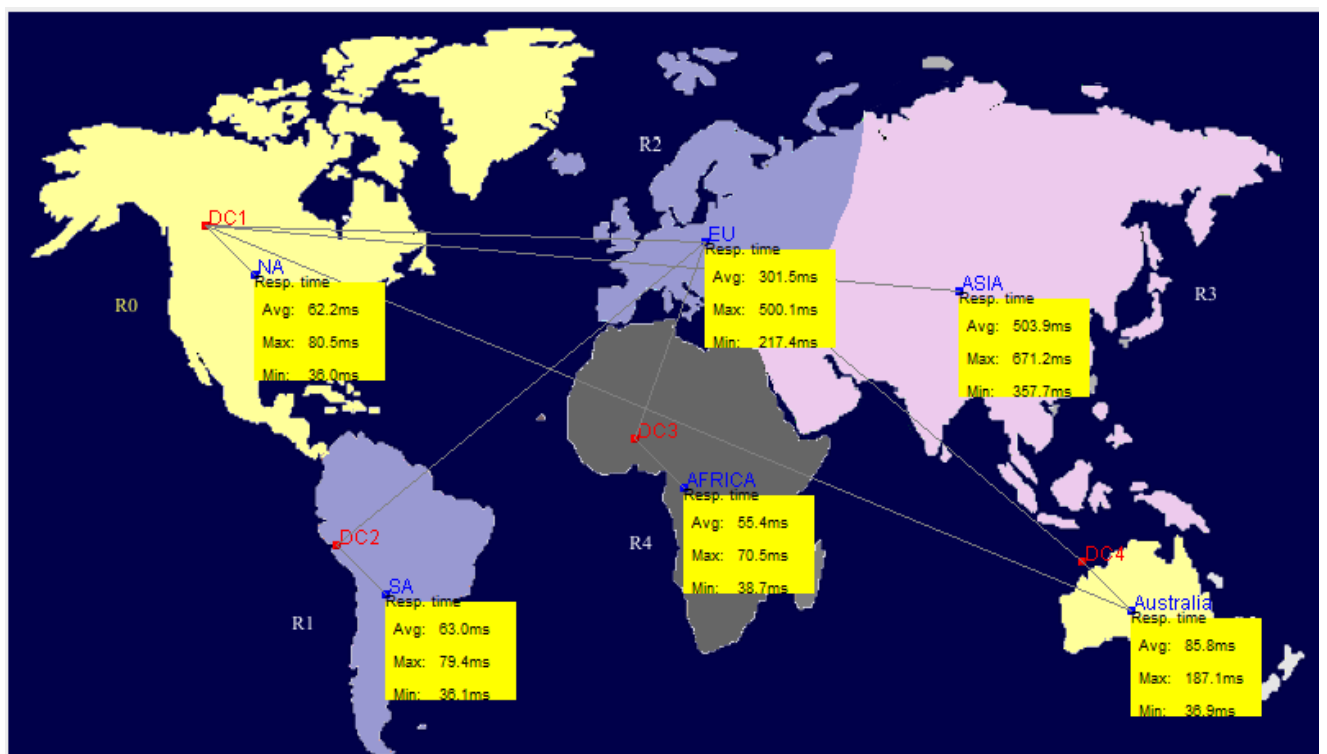
10

Executable instruction length per request:
(bytes)

100

Load balancing policy across VM's in a single Data Center:

Equally Spread Current Execution Lo...



Userbase	Avg (ms)	Min (ms)	Max (ms)
AFRICA	55.354	38.683	70.491
ASIA	503.895	357.744	671.246
Australia	85.828	36.928	187.102
EU	301.514	217.36	500.102
NA	62.162	35.957	80.543
SA	62.97	36.143	79.402

Cost

Total Virtual Machine Cost : \$2.01

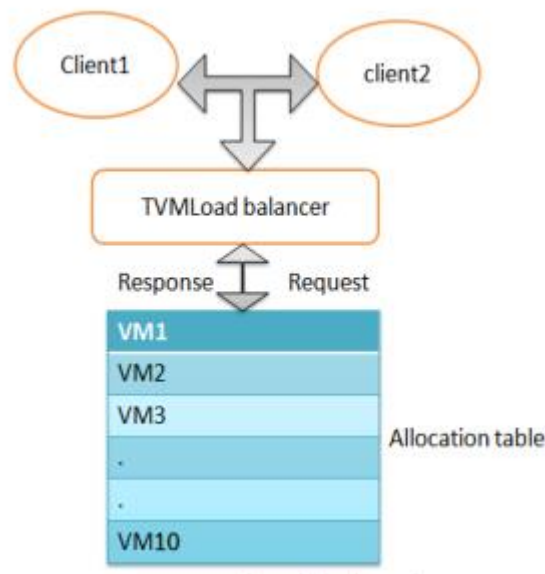
Total Data Transfer Cost : \$54.84

Grand Total : \$56.84

Data Center	VM Cost	Data Transfer Cost	Total
DC2	0.502	6.843	7.345
DC1	0.502	29.56	30.062
DC4	0.502	11.625	12.126
DC3	0.502	6.808	7.31

3. Use closest data center service broker policy and throttled load balancing algorithm

- Closest Data Center Policy:** The datacenter which is having least proximity from the user is selected. Proximity in term of least network latency. If more than one Datacenters having same proximity then it will select datacenter randomly to balance the load.
- Throttled Load Balancing Algorithm:** In this algorithm the load balancer maintains an index table of virtual machines as well as their states (Available or Busy). The client/server first makes a request to data center to find a suitable virtual machine (VM) to perform the recommended job. The data center queries the load balancer for allocation of the VM. The load balancer scans the index table from top until the first available VM is found or the index table is scanned fully. If the VM is found, the load data center.



Main Configuration
Data Center Configuration
Advanced

Simulation Duration: 60.0 min

User bases:

Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
NA	0	60	100	2	6	106537	10653
SA	1	60	100	1	6	105437	10543
EU	2	60	100	19	23	154788	15478
ASIA	3	60	100	17	21	251854	25185
AFRICA	4	60	100	20	0	62284	6228

Add New
Remove

Application Deployment Configuration:
Service Broker Policy: Closest Data Center

Data Center	# VMs	Image Size	Memory	BW
DC1	5	10000	512	1000
DC3	5	10000	512	1000
DC2	5	10000	512	1000
DC4	5	10000	512	1000

Add New
Remove

Main Configuration
Data Center Configuration
Advanced

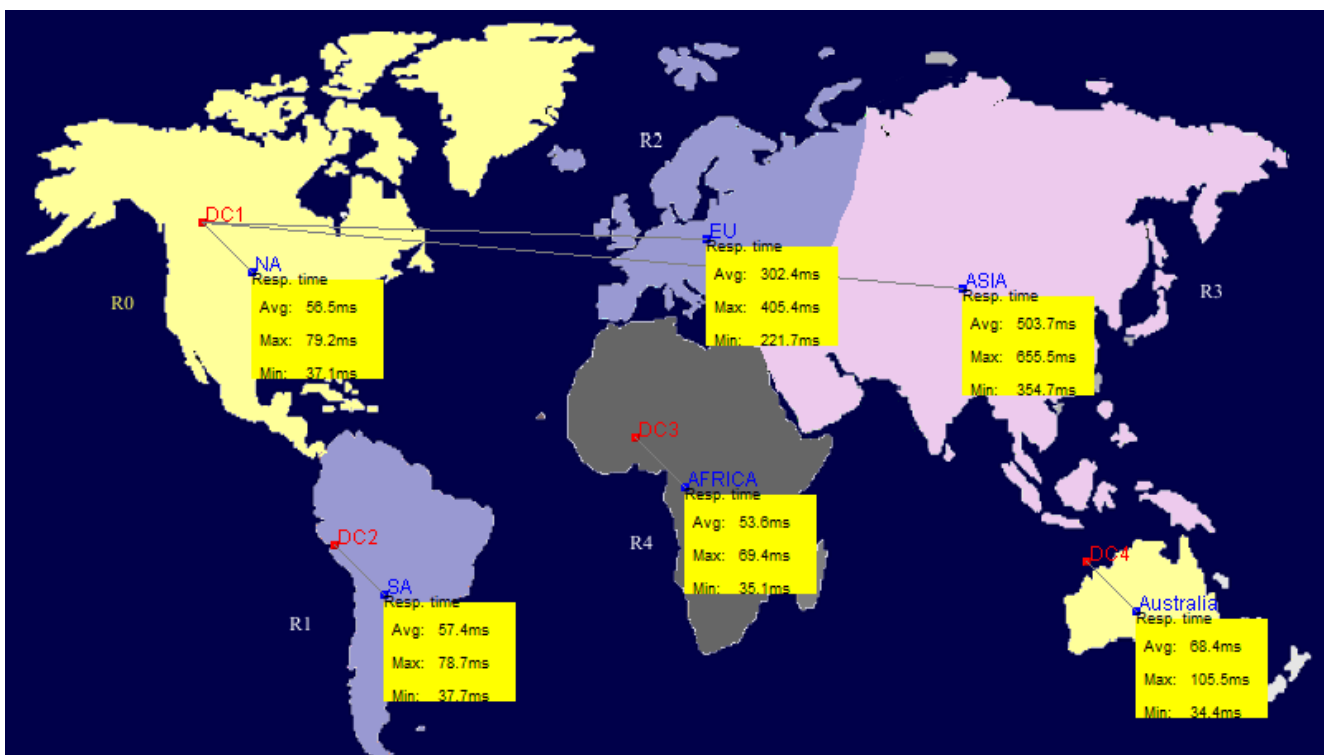
User grouping factor in User Bases:
(Equivalent to number of simultaneous users from a single user base)

Request grouping factor in Data Centers:
(Equivalent to number of simultaneous requests a single applicaiton server instance can support.)

Executable instruction length per request:
(bytes)

Load balancing policy
across VM's in a single Data Center:

Throttled



Response Time by Region

Userbase	Avg (ms)	Min (ms)	Max (ms)
AFRICA	53.61	35.06	69.43
ASIA	503.74	354.66	655.47
Australia	68.36	34.43	105.46
EU	302.41	221.68	405.37
NA	56.52	37.05	79.23
SA	57.37	37.73	78.69

Cost

Total Virtual Machine Cost (\$):	2.01
Total Data Transfer Cost (\$):	54.84
Grand Total: (\$)	56.84

Data Center	VM Cost \$	Data Transfer Cost \$	Total \$
DC2	0.50	6.84	7.34
DC1	0.50	32.29	32.79
DC4	0.50	11.62	12.13
DC3	0.50	4.08	4.58

4. Configure the simulation tool to analyze the performance of social networking app

As Cloud Analyst tool, tests the behavior and performance of the large-scale internet application on the cloud, we would select social networking site like Facebook and gather the data pertaining to Facebook from various sources. According to socialbakers.com, the social networking giant Facebook has 1.09 billion Daily Active Users (DAU) and 1.65 billion Monthly Active Users (MAU). 90% of these users are mobile users i.e., they use mobile devices like smart-phones and gadgets to access the App of this social networking giant. Mobile technology and the growth of smart-phones has fueled the growth of Facebook. On an average, a user spends 20 minutes per Facebook visit. The United States of America leads the list of countries in number of Facebook users. Facebook is an important asset for social marketers. The table 1 below lists the number of Facebook users in six continents of the world up to 15th of November, 2015

Main Configuration

Data Center Configuration

Advanced

Simulation Duration:

60.0

min

User bases:

Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
NA	0	60	100	2	6	106537	10653
SA	1	60	100	1	6	105437	10543
EU	2	60	100	19	23	154788	15478
ASIA	3	60	100	17	21	251854	25185
AFRICA	4	60	100	20	0	62284	6228

Add New

Remove

Application Deployment Configuration:

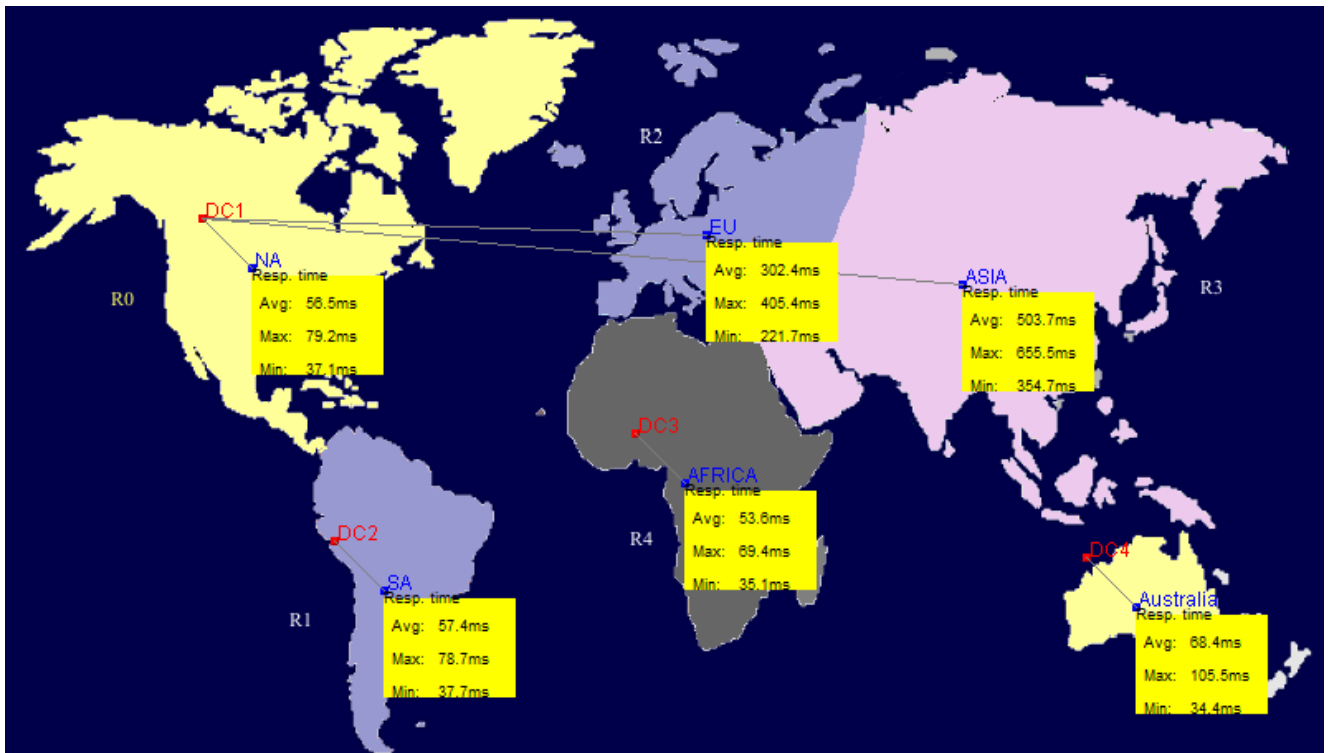
Service Broker Policy:

Closest Data Center

Data Center	# VMs	Image Size	Memory	BW
DC1	5	10000	512	1000
DC3	5	10000	512	1000
DC2	5	10000	512	1000
DC4	5	10000	512	1000

Add New

Remove



Response Time by Region

Userbase	Avg (ms)	Min (ms)	Max (ms)
AFRICA	53.61	35.06	69.43
ASIA	503.74	354.66	655.47
Australia	68.36	34.43	105.46
EU	302.41	221.68	405.37
NA	56.52	37.05	79.23
SA	57.37	37.73	78.69

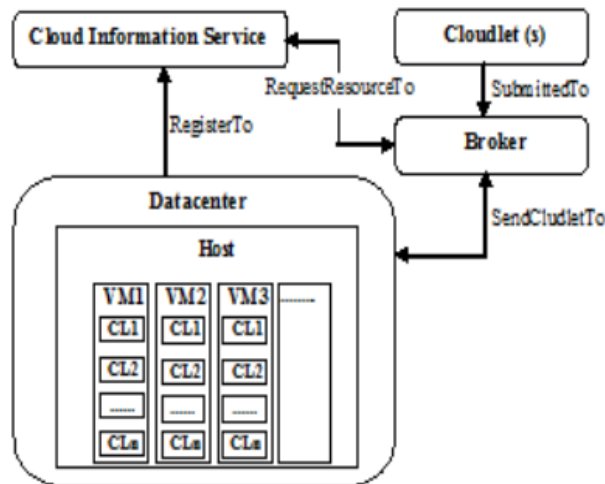
Cost

Total Virtual Machine Cost (\$):	2.01
Total Data Transfer Cost (\$):	54.84
Grand Total: (\$)	56.84

Data Center	VM Cost \$	Data Transfer Cost \$	Total \$
DC2	0.50	6.84	7.34
DC1	0.50	32.29	32.79
DC4	0.50	11.62	12.13
DC3	0.50	4.08	4.58

Practical II: Use CloudSim Toolkit and do the following:

CloudSim Simulation Tool is the most popular simulator used by researchers and developers nowadays for the cloud-related issues in the research field. This manual will ease your learning by providing simple steps to follow up with installing and understanding this simulation tool. The process of simulation is as follows:



The main scenario of CloudSim simulation is revealed in this figure. This scenario may be described the detail of each component of CloudSim as follows: that Data centers (DC) provides resource such as more than one host. Host is physical machine which allocates more than one VMs. Virtual machine (VM) are logical equipment on that the cloudlet will be executed. Broker has DC characteristics that allow it to submit virtual machines to the exact host. Cloud Information Service (CIS) is responsible for the listing of resources, indexing, as well as discovering the efficiency of data centers.

There are two CloudSim scheduling policies such as Cloudlet scheduler policies and VM Scheduler Policies:

Cloudlet scheduler policy: This scheduler policy is an abstract class. It represents the scheduling policy accomplished by a VM. Therefore, classes inheriting this must perform Cloudlets. Furthermore, the management for cloudlet interface is also performed in this class

- **CloudletScheduleTimeShared:** This scheduling policy performed by a VM. Tasks carry out Time-shared in VM.
- **CloudletSchedulerSpaceShared:** This implements a scheduling strategy performed by a virtual machine. It will be considered single task per VM and remaining tasks will be in a waiting list. We believe that file relocate from tasks waiting happens earlier than task execution. Therefore, tasks have to hang around for CPU, data relocates occurs as soon as tasks are submitted.
- **CloudletSchedulerDynamicWorkload:** This implements a scheduling strategy performed by a virtual machine by considering that there is one task that is functioning as an online service.

VM scheduler Policy: VM Scheduler is an abstract class that represents the policy used by a VMM to share processing power among VMs running in a host.

- **VMSchedulerTimeShared:** Time-Shared VM Scheduler is a VMM share strategy that allocates more than one Pe to a Virtual Machine, and permits sharing of PEs by numerous Virtual Machines. This class also implements 10% efficiency decreased due to Virtual Machine relocation. This scheduler doesn't support over-subscription.

- **VMSchedulerSpaceShared**: Space-Shared VM Scheduler is a VMM share strategy that allocates more than one PE to a Virtual Machine, and doesn't permit sharing of PEs. If VM has no free PEs, allotment decline. Free PEs are not allotted to VMs.

- **VM Scheduler Time-Shared Over Subscription**: Time-shared VM Scheduler allows over subscription, so it is known as VMSchedulerTimeShared Oversubscription. Additionally, the scheduler still enables the share of VMs which need extra CPU ability that is obtainable. Oversubscription results in efficiency decreased. Each virtual PE cannot be allocated more CPU capacity than MIPS of a single PE.

1. To create a datacenter with one host and run one cloudlet on it.

Steps:

1. Initialize the CloudSim package. It should be called before creating any entities.

2. Create Datacenters.

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

```
Datacenter datacenter0 = createDatacenter("Datacenter_0");
```

3. Create Broker:

```
DatacenterBroker broker = createBroker();
```

4. Create one virtual machine:

```
vmList = new ArrayList<Vm>();
```

5. Create one Cloudlet:

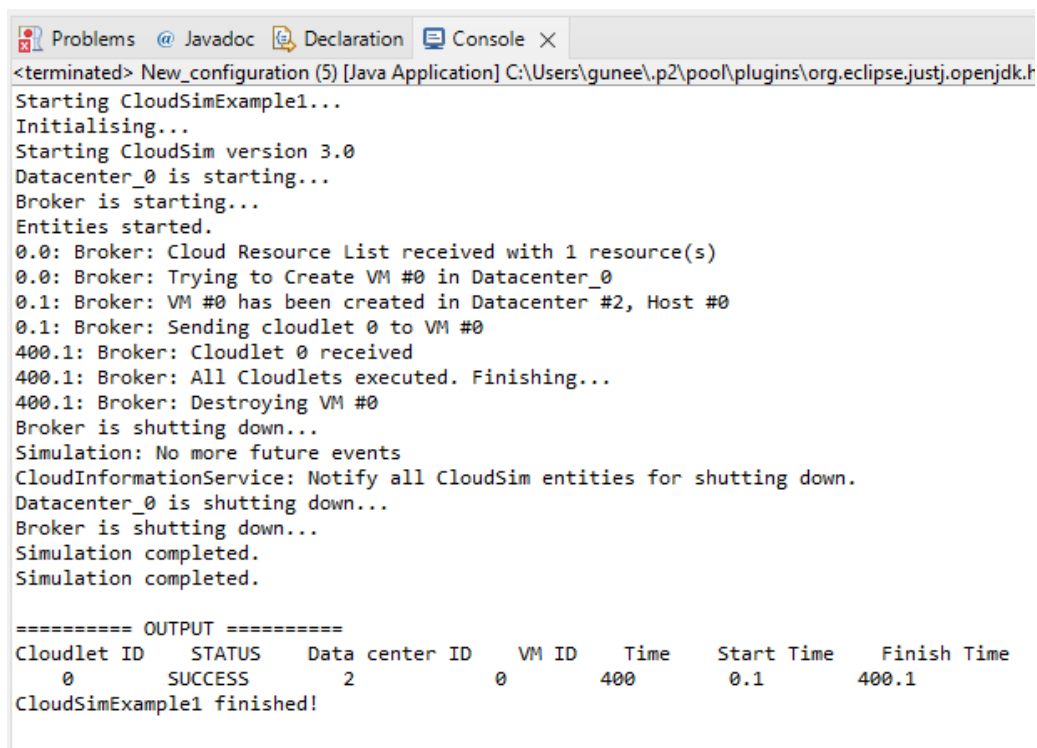
```
cloudletList = new ArrayList<Cloudlet>();
```

6. Starts the simulation

```
CloudSim.startSimulation();
```

7. Print results when simulation is over

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



```
<terminated> New_configuration (5) [Java Application] C:\Users\gunee\p2\pool\plugins\org.eclipse.justj.openjdk.f
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!
```

2. To create two datacenters with one host each and two cloudlets on them.

Steps:

1. Initialize the CloudSim package. It should be called before creating any entities.

2. Create Datacenters.

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

```
Datacenter datacenter0 = createDatacenter("Datacenter_0");
```

```
Datacenter datacenter1 = createDatacenter("Datacenter_1");
```

3. Create Broker:

```
DatacenterBroker broker = createBroker();
```

4. Create one virtual machine:

```
vmList = new ArrayList<Vm>();
```

a. Create 2 VMs

```
Vm vm1 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new  
CloudletSchedulerTimeShared());
```

```
vmid++;
```

```
Vm vm2 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new  
CloudletSchedulerTimeShared());
```

b. Adding vms to the vmList

```
vmList.add(vm1);
```

```
vmList.add(vm2);
```

c. Submit vmList to Broker

```
broker.submitVmList(vmList);
```

5. Create two Cloudlets:

```
cloudletList = new ArrayList<Cloudlet>();
```

a. Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);

```
cloudlet1.setUserId(brokerId);
```

```
id++;
```

```
Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,  
utilizationModel, utilizationModel, utilizationModel);
```

```
cloudlet2.setUserId(brokerId);
```

b. Add the cloudlets to the list

```
cloudletList.add(cloudlet1);
```

```
cloudletList.add(cloudlet2);
```

c. Submit cloudlet list to the broker

```
broker.submitCloudletList(cloudletList);
```

d. Bind the cloudlets to the VMs. This way, broker will submit the bound cloudlets only to the specific VM

```
broker.bindCloudletToVm(cloudlet1.getCloudletId(), vm1.getId());
```

```
broker.bindCloudletToVm(cloudlet2.getCloudletId(), vm2.getId());
```

6. Starts the simulation

```
CloudSim.startSimulation();
```

7. Print results when simulation is over

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

```
printCloudletList(newList);
```

```

Starting CloudSimExample4...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Datacenter_1 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 2 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
[VmScheduler.vmCreate] Allocation of VM #1 to Host #0 failed by MIPS
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Creation of VM #1 failed in Datacenter #2
0.1: Broker: Trying to Create VM #1 in Datacenter_1
0.2: Broker: VM #1 has been created in Datacenter #3, Host #0
0.2: Broker: Sending cloudlet 0 to VM #0
0.2: Broker: Sending cloudlet 1 to VM #1
160.2: Broker: Cloudlet 0 received
160.2: Broker: Cloudlet 1 received
160.2: Broker: All Cloudlets executed. Finishing...
160.2: Broker: Destroying VM #0
160.2: Broker: Destroying VM #1
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Datacenter_1 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       160         0.2        160.2
    1          SUCCESS        3           1       160         0.2        160.2
CloudSimExample4 finished!

```


3. To create two datacenters with one host each and run cloudlets of two users on them.

Steps:

1. Initialize the CloudSim package. It should be called before creating any entities.

2. Create Datacenters.

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

```
Datacenter datacenter0 = createDatacenter("Datacenter_0");
```

```
Datacenter datacenter1 = createDatacenter("Datacenter_1");
```

3. Create Broker:

```
DatacenterBroker broker = createBroker();
```

4. Create one virtual machine:

```
vmList = new ArrayList<Vm>();
```

a. Create 2 VMs

```
Vm vm1 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new  
CloudletSchedulerTimeShared());
```

```
vmid++;
```

```
Vm vm2 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new  
CloudletSchedulerTimeShared());
```

b. Adding vms to the vmList

```
vmList.add(vm1);
```

```
vmList.add(vm2);
```

c. Submit vmList to Broker

```
broker.submitVmList(vmList);
```

5. Create Cloudlets

```
cloudletList = new ArrayList<Cloudlet>();
```

```
Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,  
utilizationModel, utilizationModel, utilizationModel);
```

```
cloudlet1.setUserId(brokerId);
```

```
id++;
```

```
Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,  
utilizationModel, utilizationModel, utilizationModel);
```

```
cloudlet2.setUserId(brokerId);
```

```
//bind the cloudlets to the vms. This way, the broker
```

```
// will submit the bound cloudlets only to the specific VM
```

```
broker.bindCloudletToVm(cloudlet1.getCloudletId(),vm1.getId());
```

```
broker.bindCloudletToVm(cloudlet2.getCloudletId(),vm2.getId());
```

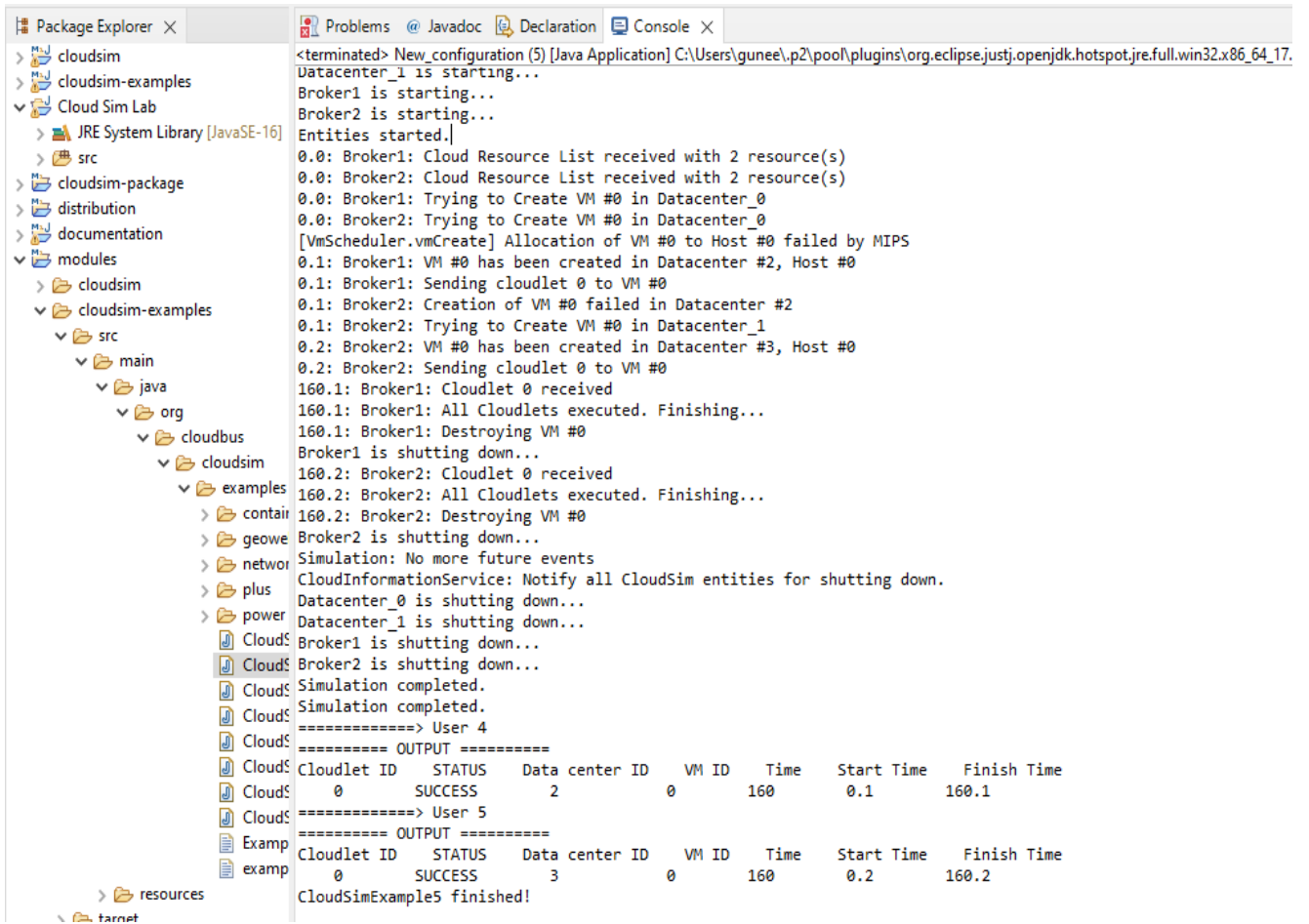
6. Starts the simulation

```
CloudSim.startSimulation();
```

7. Print results when simulation is over

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

```
printCloudletList(newList);
```



4. Use CloudSim Toolkit and create a datacenter with one host and a network topology and run one cloudlet on it.

1. Initialize the CloudSim package. It should be called before creating any entities.

2. Create Datacenters.

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

```
Datacenter datacenter0 = createDatacenter("Datacenter_0");
```

3. Create Broker:

```
DatacenterBroker broker = createBroker();
```

4. Create one virtual machine:

```
Vm vm1 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new  
CloudletSchedulerTimeShared());  
vmList = new ArrayList<Vm>();  
Adding vms to the vmList  
vmList.add(vm1);  
Submit vmList to Broker  
broker.submitVmList(vmList);
```

5. Create Cloudlets

```
cloudletList = new ArrayList<Cloudlet>();  
Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,  
utilizationModel, utilizationModel, utilizationModel);  
cloudlet1.setUserId(brokerId);  
//add the cloudlet to the list  
cloudletList.add(cloudlet1);  
//submit cloudlet list to the broker  
broker.submitCloudletList(cloudletList);
```

6. Configure Network by loading the topology file.

```
NetworkTopology.buildNetworkTopology("topology.brite");  
//maps CloudSim entities to BRITE entities  
//PowerDatacenter will correspond to BRITE node 0  
int briteNode=0;  
NetworkTopology.mapNode(datacenter0.getId(),briteNode);  
//Broker will correspond to BRITE node 3  
briteNode=3;  
NetworkTopology.mapNode(broker.getId(),briteNode);
```

7. Starts the simulation

```
CloudSim.startSimulation();
```

8. Print results when simulation is over

```
List<Cloudlet> newList = broker.getCloudletReceivedList();  
CloudSim.stopSimulation();  
printCloudletList(newList);
```

```
Problems @ Javadoc Declaration Console × Coverage Debug
<terminated> NetworkExample1 [Java Application] C:\Users\gunee\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.0.v20211012-1059\jre\bin\javaw.exe (Nov 20, 2021, 10
Starting NetworkExample1...
Initialising...
Topology file: topology.brite
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)
7.800000190734863: Broker: Trying to Create VM #0 in Datacenter_0
15.700000381469726: Broker: VM #0 has been created in Datacenter #2, Host #0
15.700000381469726: Broker: Sending cloudlet 0 to VM #0
183.50000057220458: Broker: Cloudlet 0 received
183.50000057220458: Broker: All Cloudlets executed. Finishing...
183.50000057220458: 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       160    19.6         179.6
NetworkExample1 finished!
```

5. Use CloudSim Toolkit and create two datacenters with one host each and run cloudlets of two users with network topology on them.

1. Initialize the CloudSim package. It should be called before creating any entities.
2. Create Datacenters.

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

```
Datacenter datacenter0 = createDatacenter("Datacenter_0");
Datacenter datacenter1 = createDatacenter("Datacenter_1");
```

3. Create Brokers:

```
DatacenterBroker broker = createBroker();
DatacenterBroker broker2 = createBroker(2);
```

4. Create one virtual machine for each broker/user:

```
vmList1 = new ArrayList<Vm>();
vmList2 = new ArrayList<Vm>();
//create two VMs: the first one belongs to user1
Vm vm1 = new Vm(vmid, brokerId1, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
Vm vm2 = new Vm(vmid, brokerId2, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
//add the VMs to the vmLists
vmList1.add(vm1);
vmList2.add(vm2);
//submit vm list to the broker
broker1.submitVmList(vmList1);
broker2.submitVmList(vmList2);
```

5. Create Two Cloudlets

```
cloudletList1 = new ArrayList<Cloudlet>();
cloudletList2 = new ArrayList<Cloudlet>();
Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
utilizationModel, utilizationModel, utilizationModel);
cloudlet1.setUserId(brokerId);
Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize, outputSize,
utilizationModel, utilizationModel, utilizationModel);
Cloudlet2.setUserId(brokerId);
//submit cloudlet list to the brokers
broker1.submitCloudletList(cloudletList1);
broker2.submitCloudletList(cloudletList2);
```

6. Configure Network by loading the topology file.

```
NetworkTopology.buildNetworkTopology("topology.brite");
//maps CloudSim entities to BRITE entities
//Datacenter0 will correspond to BRITE node 0
int briteNode=0;
NetworkTopology.mapNode(datacenter0.getId(),briteNode);
//Datacenter1 will correspond to BRITE node 2
briteNode=2;
NetworkTopology.mapNode(datacenter1.getId(),briteNode);
//Broker1 will correspond to BRITE node 3
briteNode=3;
NetworkTopology.mapNode(broker1.getId(),briteNode);

//Broker2 will correspond to BRITE node 4
briteNode=4;
NetworkTopology.mapNode(broker2.getId(),briteNode);
```

7. Starts the simulation

```
CloudSim.startSimulation();
```

8. Print results when simulation is over

```

List<Cloudlet> newList1 = broker1.getCloudletReceivedList();
List<Cloudlet> newList2 = broker2.getCloudletReceivedList();

0.0: Broker1: Trying to create VM #0 in Datacenter_0
0.0: Broker2: Trying to Create VM #0 in Datacenter_0
[VmScheduler.vmCreate] Allocation of VM #0 to Host #0 failed by MIPS
0.1: Broker1: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker1: Sending cloudlet 0 to VM #0
0.1: Broker2: Creation of VM #0 failed in Datacenter #2
0.1: Broker2: Trying to Create VM #0 in Datacenter_1
0.2: Broker2: VM #0 has been created in Datacenter #3, Host #0
0.2: Broker2: Sending cloudlet 0 to VM #0
160.1: Broker1: Cloudlet 0 received
160.1: Broker1: All Cloudlets executed. Finishing...
160.1: Broker1: Destroying VM #0
Broker1 is shutting down...
160.2: Broker2: Cloudlet 0 received
160.2: Broker2: All Cloudlets executed. Finishing...
160.2: Broker2: Destroying VM #0
Broker2 is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Datacenter_1 is shutting down...
Broker1 is shutting down...
Broker2 is shutting down...
Simulation completed.
Simulation completed.
=====> User 4
===== OUTPUT =====
Cloudlet ID   STATUS   Data center ID   VM ID   Time   Start Time   Finish Time
    0         SUCCESS       2           0     160        0.1       160.1
=====> User 5
===== OUTPUT =====
Cloudlet ID   STATUS   Data center ID   VM ID   Time   Start Time   Finish Time
    0         SUCCESS       3           0     160        0.2       160.2
NetworkExample3 finished!

```

Practical III: Examine the architecture and services offered by Google Cloud Platform

Google Cloud Architecture Framework

The Google Cloud Architecture Framework provides recommendations and describes best practices to help architects, developers, administrators, and other cloud practitioners design and operate a cloud topology that's secure, efficient, resilient, high-performing, and cost-effective.

The Architecture Framework is organized into the following categories:

System design

Define the architecture, components, modules, interfaces, and data needed to satisfy cloud system requirements, and learn about Google Cloud products and features that support system design.

Operational excellence

Efficiently deploy, operate, monitor, and manage your cloud workloads.

Security, privacy, and compliance

Maximize the security of your data and workloads in the cloud, design for privacy, and align with regulatory requirements and standards.

Reliability

Design and operate resilient and highly available workloads in the cloud.

Cost optimization

Maximize the business value of your investment in Google Cloud.

Performance optimization

Design and tune your cloud resources for optimal performance.

Google Cloud Services

Compute

App Engine: App Engine enables you to build and host applications on the same systems that power Google applications. App Engine offers fast development and deployment; simple administration, with no need to worry about hardware, patches or backups; and effortless scalability.

Compute Engine: Compute Engine offers scalable and flexible virtual machine computing capabilities in the cloud, with options to utilize certain CPUs, GPUs, or Cloud TPUs. You can use Compute Engine to solve large-scale processing and analytic problems on Google's computing, storage, and networking infrastructure.

Google Cloud VMware Engine (GCVE) is a managed VMware-as-a-Service that is specifically designed for running VMware workloads on Google Cloud Platform. GCVE enables customers to run VMware virtual machines natively in a dedicated, private, software-defined data center.

Databases

Cloud Bigtable: Cloud Bigtable is a fast, fully-managed, highly-scalable NoSQL database service. It is designed for the collection and retention of data from 1TB to hundreds of PB.

Datastore: Datastore is a fully-managed, schemaless, non-relational datastore. It provides a rich set of query capabilities, supports atomic transactions, and automatically scales up and down in response to load. It can scale to support an application with 1,000 users or 10 million users with no code changes.

Firestore: Firestore is a NoSQL document database for storing, syncing, and querying data for mobile and web apps. Its client libraries provide live synchronization and offline support, while its security features and integrations with Firebase and Google Cloud Platform accelerate building serverless apps.

Memorystore: Memorystore, which includes Memorystore for Redis and Memorystore for Memcached, provides a fully-managed in-memory data store service that allows customers to deploy distributed caches that provide sub-millisecond data access.

Cloud Spanner: Cloud Spanner is a fully-managed, mission-critical relational database service. It is designed to provide a scalable online transaction processing (OLTP) database with high availability and strong consistency at global scale.

Cloud SQL: Cloud SQL is a web service that allows you to create, configure, and use relational databases that live in Google's cloud. It is a fully-managed service that maintains, manages, and administers your databases, allowing you to focus on your applications and services.

Networking

Cloud CDN: Cloud CDN uses Google's globally distributed edge points of presence to cache HTTP(S) load balanced content close to your users.

Cloud DNS: Cloud DNS is a high performance, resilient, global, fully-managed DNS service that provides a RESTful API to publish and manage DNS records for your applications and services.

Cloud IDS (Cloud Intrusion Detection System): Cloud IDS is a managed service that aids in detecting certain malware, spyware, command-and-control attacks, and other network-based threats.

Cloud Interconnect: Cloud Interconnect offers enterprise-grade connections to Google Cloud Platform using Google Services for Dedicated Interconnect, Partner Interconnect and Cloud VPN. This solution allows you to directly connect your on-premises network to your Virtual Private Cloud.

Cloud Load Balancing: Cloud Load Balancing provides scaling, high availability, and traffic management for your internet-facing and private applications.

Cloud NAT (Network Address Translation): Cloud NAT enables instances in a private network to communicate with the internet.

Cloud Router: Cloud Router enables dynamic Border Gateway Protocol (BGP) route updates between your VPC network and your non-Google network.

Cloud VPN: Cloud VPN allows you to connect to your Virtual Private Cloud (VPC) network from your existing network, such as your on-premises network, another VPC network, or another cloud provider's network, through an IPsec connection using (i) Classic VPN, which supports dynamic (BGP) routing or static routing (route-based or policy-based), or (ii) HA (high-availability) VPN, which supports dynamic routing with a simplified redundancy setup, separate failure domains for the gateway interfaces, and a higher service level objective.

Google Cloud Armor: Google Cloud Armor offers a policy framework and rules language for customizing access to internet-facing applications and deploying defenses against denial of service attacks as well as targeted application attacks. Components of Google Cloud Armor include: L3/L4 volumetric DDoS Protection, preconfigured web-application firewall (WAF) rules, and custom rules language.

Google Cloud Armor Managed Protection Plus is a managed application protection service subscription that bundles Google Cloud Armor WAF and DDoS Protection with additional services and capabilities including DDoS response support, DDoS bill protection, and Google Cloud Armor Adaptive Protection, which is Google's machine-learning based solution to protect internet-facing endpoints from network and application-based attacks.

Network Connectivity Center: Network Connectivity Center is a hub-and-spoke model for network connectivity management in Google Cloud that facilitates connecting a customer's resources to its cloud network.

Network Intelligence Center: Network Intelligence Center is Google Cloud's comprehensive network monitoring, verification, and optimization platform across the Google Cloud, multi-cloud, and on-prem environments.

Network Service Tiers: Network Service Tiers enable you to select different quality networks (tiers) for outbound traffic to the internet: the Standard Tier primarily utilizes third party transit providers while the Premium Tier leverages Google's private backbone and peering surface for egress.

Service Directory is a managed service that offers customers a single place to publish, discover and connect their services in a consistent way, regardless of their environment. Service Directory supports services in Google Cloud, multi-cloud, and on-prem environments and can scale up to thousands of services and endpoints for a single project.

Traffic Director: Traffic Director is Google Cloud Platform's traffic management service for open service meshes.

Virtual Private Cloud: Virtual Private Cloud provides a private network topology with IP allocation, routing, and network firewall policies to create a secure environment for your deployments.

Management Tools

Cloud Console App is a native mobile app that enables customers to manage key Google Cloud services. It provides monitoring, alerting, and the ability to take actions on resources.

Cloud Deployment Manager is a hosted configuration tool which allows developers and administrators to provision and manage their infrastructure on Google Cloud Platform. It uses a declarative model which allows users to define or change the resources necessary to run their applications and will then provision and manage those resources.

Cloud Shell is a tool that provides command-line access to cloud resources directly from your browser. You can use Cloud Shell to run experiments, execute Cloud SDK commands, manage projects and resources, and do lightweight software development via the built-in web editor.

Recommenders automatically analyze your usage patterns to provide recommendations and insights across services to help you use Google Cloud Platform in a more secure, cost-effective, and efficient manner.

Service Infrastructure: Service Infrastructure is a foundational platform for creating, managing, securing, and consuming APIs and services. It includes:

- Service Management API, which lets service producers manage their APIs and services;
- Service Consumer Management API, which lets service producers manage their relationships with their service consumers; and
- Service Control API, which lets managed services integrate with Service Infrastructure for admission control and telemetry reporting functionality.
- Service Usage API, which lets service consumers manage their usage of APIs and services.

Healthcare and Life Sciences

Cloud Healthcare: Cloud Healthcare is a fully-managed service to send, receive, store, query, transform, and analyze healthcare and life sciences data and enable advanced insights and operational workflows using highly scalable and compliance-focused infrastructure.

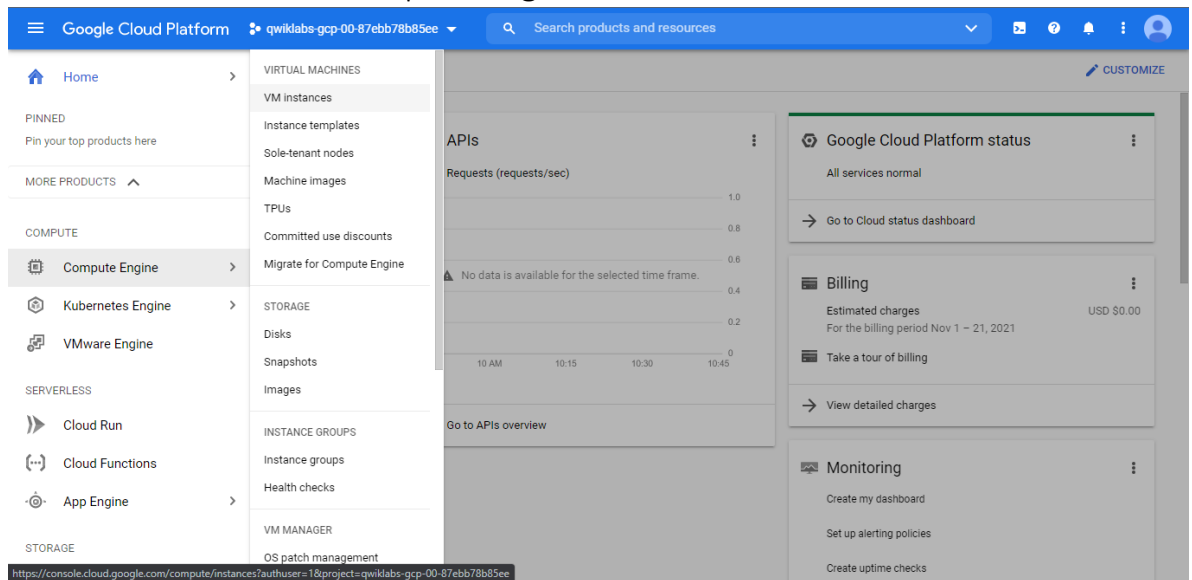
Media and Gaming

Game Servers: Game Servers is a managed service that enables game developers to deploy and manage their dedicated game servers across multiple Agones clusters around the world through a single interface.

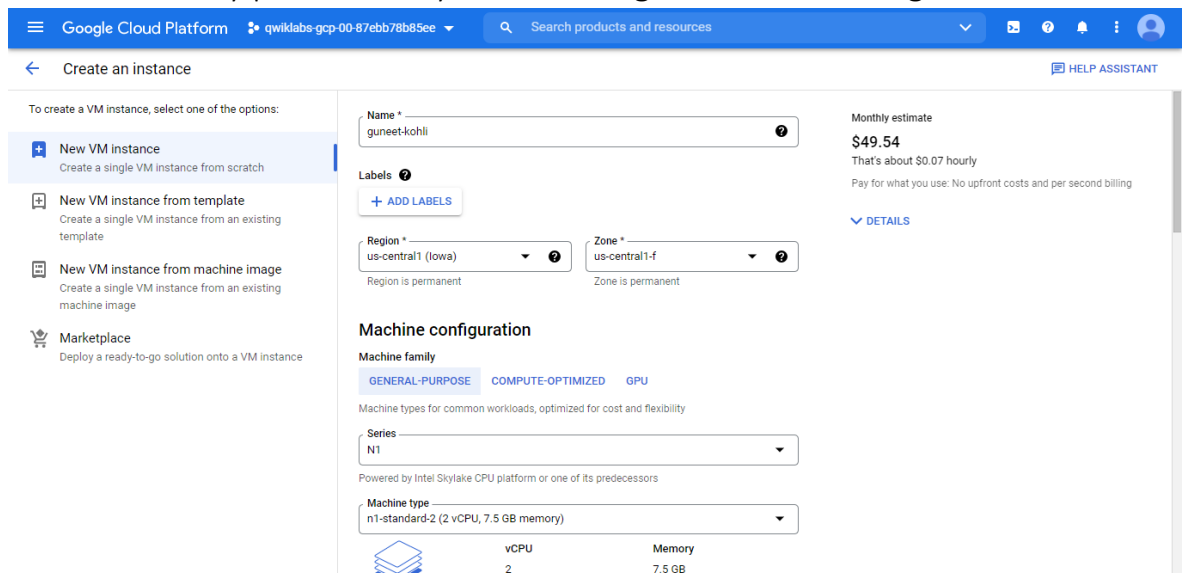
Transcoder API: Transcoder API can batch convert media files into optimized formats to enable streaming across web, mobile, and living room devices. It provides fast, easy to use, large-scale processing of advanced codecs while utilizing Google's storage, networking, and delivery infrastructure.

CREATING A VIRTUAL MACHINE ON GOOGLE CLOUD PLATFORM:

- Creating a new instance from the Cloud Console:
 - NAVIGATION MENU > Compute Engine > VM Instances



- To create a new instance, click CREATE INSTANCE
- There are many parameters you can configure when creating a new instance.



- Click Create: It took about a minute for the machine to be created. After that, the new virtual machine is listed on the VM Instances page.
- To use SSH to connect to the virtual machine, in the row for your machine, click SSH.

Do you want to initiate an SSH connection to VM instance 'guneet-kohli'?

Connect

Cancel

- Installing an NGINX web server:

- To get root access, used “sudo su -” command in SSH shell

```
Last login: Mon Nov 22 05:26:57 2021 from 173.194.93.95
student-04-c6869a1ab1a0@guneet-kohli:~$ sudo su -
root@guneet-kohli:~#
```

- As the root user, updated the operating system

```
student-04-c6869a1ab1a0@guneet-kohli:~$ sudo su -
root@guneet-kohli:~# apt-get update
Get:1 http://security.debian.org/debian-security buster/updates InRelease [65.4 kB]
Hit:2 http://deb.debian.org/debian buster InRelease
Get:3 http://deb.debian.org/debian buster-updates InRelease [51.9 kB]
Get:4 http://deb.debian.org/debian buster-backports InRelease [46.7 kB]
Get:5 http://packages.cloud.google.com/apt cloud-sdk-buster InRelease [6774 B]
Get:6 http://security.debian.org/debian-security buster/updates/main Sources [203 kB]
Get:7 http://packages.cloud.google.com/apt google-cloud-packages-archive-keyring-buster InRelease [5553 B]
Get:8 http://security.debian.org/debian-security buster/updates/main amd64 Packages [309 kB]
Get:9 http://security.debian.org/debian-security buster/updates/main Translation-en [163 kB]
Get:10 http://packages.cloud.google.com/apt google-cloud-engine-buster-stable InRelease [5526 B]
Get:11 http://deb.debian.org/debian buster-backports/main Sources.diff/Index [27.8 kB]
Get:12 http://deb.debian.org/debian buster-backports/main amd64 Packages.diff/Index [27.8 kB]
Get:13 http://deb.debian.org/debian buster-backports/main Sources 2021-11-13-2001.35.pdiff [3433 B]
Get:14 http://deb.debian.org/debian buster-backports/main Sources 2021-11-14-0202.19.pdiff [66 B]
Get:15 http://deb.debian.org/debian buster-backports/main Sources 2021-11-14-0202.19.pdiff [66 B]
Get:16 http://deb.debian.org/debian buster-backports/main amd64 Packages 2021-11-14-0202.19.pdiff [4802 B]
Get:17 http://deb.debian.org/debian buster-backports/main amd64 Packages 2021-11-14-0202.19.pdiff [4802 B]
Get:18 http://packages.cloud.google.com/apt cloud-sdk-buster/main amd64 Packages [197 kB]
Get:19 http://packages.cloud.google.com/apt google-cloud-packages-archive-keyring-buster/main amd64 Packages [390 B]
Get:20 http://packages.cloud.google.com/apt google-cloud-engine-buster-stable/main amd64 Packages [2023 B]
Fetched 1121 kB in 1s (1129 kB/s)
Reading package lists... Done
root@guneet-kohli:~#
```

- To CONFIRM, NGINX IS RUNNING, “ps auwx | grep nginx” COMMAND IS USED

```
root@guneet-kohli:~# ps auwx | grep nginx
root      2046  0.0  0.0  65660  1700 ?        Ss   05:32   0:00 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
www-data  2047  0.0  0.1  80724  11228 ?        S    05:32   0:00 nginx: worker process
www-data  2048  0.0  0.1  80724  11228 ?        S    05:32   0:00 nginx: worker process
root      2132  0.0  0.0   4836   884 pts/0    S+   05:36   0:00 grep nginx
```

- TO SEE THE WEB PAGE, return to the Cloud Console and click the External IP link in the row for your machine,

Recommendations	In use by	Internal IP	External IP	Connect
		10.128.0.2 (nic0)	34.67.24.117 ↗	SSH ▾ ⋮

- Enter <http://34.67.24.117/> in your browser, then the default web page gets displayed

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

- Configuring the load balancing service:

- Create a static external IP address for your load balancer:

```
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$ gcloud compute addresses create network-lb-ip-1 \
> --region us-central1
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-1b3afd0d0fad/regions/us-central1/addresses/network-lb-ip-1].
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$
```

- Add a legacy HTTP health check resource:

```
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$ gcloud compute http-health-checks create basic-check
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-1b3afd0d0fad/global/httpHealthChecks/basic-check].
NAME: basic-check
HOST:
PORT: 80
REQUEST_PATH: /
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$
```

- Add a target pool in the same region as your instances. Run the following to create the target pool and use the health check, which is required for the service to function

```
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$ gcloud compute target-pools create www-pool \
> --region us-central1 --http-health-check basic-check
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-1b3afd0d0fad/regions/us-central1/targetPools/
NAME: www-pool
REGION: us-central1
SESSION_AFFINITY: NONE
BACKUP:
HEALTH_CHECKS: basic-check
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$
```

- Add a forwarding rule

```
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$ gcloud compute forwarding-rules create www-rule \
> --region us-central1 \
> --ports 80 \
> --address network-lb-ip-1 \
> --target-pool www-pool
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-1b3afd0d0fad/regions/us-central1/forwardingRules/www-rule].
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$
```

- Creating an HTTP load balancer

- Creating a LOAD BALANCER TEMPLATE

```
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$ gcloud compute instance-templates create lb-backend-template \
> --region=us-central1 \
> --network=default \
> --subnet=default \
> --tags=allow-health-check \
> --image-family=debian-9 \
> --image-project=debian-cloud \
> --metadata-startup-script='#!/bin/bash
> apt-get update
> apt-get install apache2 -y
> a2ensite default-ssl
> a2enmod ssl
> vm_hostname=$(curl -H "Metadata-Flavor:Google" \
> http://169.254.169.254/computeMetadata/v1/instance/name)"
> echo "Page served from: $vm_hostname" | \
> tee /var/www/html/index.html
> systemctl restart apache2'
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-1b3afd0d0fad/global/instanceTemplates/lb-backend-template].
NAME: lb-backend-template
MACHINE_TYPE: n1-standard-1
PREEMPTIBLE:
CREATION_TIMESTAMP: 2021-11-21T22:44:18.808-08:00
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$
```

- Creating a managed instance group based on the template

```
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$ gcloud compute instance-groups managed create lb-backend-group \
> --template=lb-backend-template --size=2 --zone=us-central1-a
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-1b3afd0d0fad/zones/us-central1-a/instanceGroupManagers/lb-backend-group].
NAME: lb-backend-group
LOCATION: us-central1-a
SCOPE: zone
BASE_INSTANCE_NAME: lb-backend-group
SIZE: 0
TARGET_SIZE: 2
INSTANCE_TEMPLATE: lb-backend-template
AUTOSCALED: no
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$
```

- Creating a fw-allow-health-check firewall rule.

```
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$ gcloud compute firewall-rules create fw-allow-health-check \
> --network=default \
> --action=allow \
> --direction=ingress \
> --source-ranges=130.211.0.0/22,35.191.0.0/16 \
> --target-tags=allow-health-check \
> --rules=tcp:80
Creating firewall...working..Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-1b3afd0d0fad/global/firewalls/fw-allow-health-check].
Creating firewall...done.
NAME: fw-allow-health-check
NETWORK: default
DIRECTION: INGRESS
PRIORITY: 1000
ALLOW: tcp:80
DENY:
DISABLED: False
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$
```

- Create a healthchecker for the load balancer

```
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$ gcloud compute health-checks create http http-basic-check \
> --port 80
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-1b3afd0d0fad/global/healthChecks/http-basic-check].
NAME: http-basic-check
PROTOCOL: HTTP
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$
```

- Create a backend service

```
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$ gcloud compute backend-services create web-backend-service \
> --protocol=HTTP \
> --port-name=http \
> --health-checks=http-basic-check \
> --global
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-1b3afd0d0fad/global/backendServices/web-backend-service].
NAME: web-backend-service
BACKENDS:
PROTOCOL: HTTP
```

- Add your instance group as the backend to the backend service

```
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$ gcloud compute backend-services create web-backend-service \
> --protocol=HTTP \
> --port-name=http \
> --health-checks=http-basic-check \
> --global
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-1b3afd0d0fad/global/backendServices/web-backend-service].
NAME: web-backend-service
BACKENDS:
PROTOCOL: HTTP
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$ gcloud compute backend-services add-backend web-backend-service \
> --instance-group=lb-backend-group \
> --instance-group-zone=us-central1-a \
> --global
Updated [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-1b3afd0d0fad/global/backendServices/web-backend-service].
```

- Create a URL map to route the incoming requests to the default backend service

```
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$ gcloud compute url-maps create web-map-http \
> --default-service web-backend-service
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-1b3afd0d0fad/global/urlMaps/web-map-http].
NAME: web-map-http
DEFAULT_SERVICE: backendServices/web-backend-service
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$
```

- Create a target HTTP proxy to route requests to your URL map

```
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$ gcloud compute target-http-proxies create http-lb-proxy \
> --url-map web-map-http
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-1b3afd0d0fad/global/targetHttpProxies/http-lb-proxy].
NAME: http-lb-proxy
URL_MAP: web-map-http
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$
```

- Create a global forwarding rule to route incoming requests to the proxy

```
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$ gcloud compute forwarding-rules create http-content-rule \
> --address=lb-ipv4-1 \
> --global \
> --target-http-proxy=http-lb-proxy \
> --ports=80
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-01-1b3afd0d0fad/global/forwardingRules/http-content-rule].
student_04_c6869a1ab1a0@cloudshell:~ (qwiklabs-gcp-01-1b3afd0d0fad)$
```