DAYANANDA SAGAR COLLEGE OF ENGINEERING

SHAVIGE MALLESHWARA HILLS, KS LAYOUT, BANGALORE-560078

Department of Computer Science and Engineering

2024-2025

Sixth Semester

Cloud Computing

Laboratory Manual

Sub Code: 22CS62

DSCE
DEPARTMENT OF COMPUTER SCIENCE

DAYANANDA SAGAR COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Vision and Mission of the Department

Vision

To provide a vibrant learning environment in computer science and engineering with focus on industry needs and research, for the students to be successful global

Mission

* To adopt a contemporary teaching learning process with emphasis on hands on and collaborative learning

* To facilitate skill development through additional training and encourage student forums for enhanced learning.

* To collaborate with industry partners and professional societies and make the students industry ready.

DSCE
DEPARTMENT OF COMPUTER SCIENCE

DAYANANDA SAGAR COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Code of Conduct in the Lab

Do's

Students shall

- Come prepared for the program to be developed in the Laboratory.
- Report any broken plugs or exposed electrical wires to your faculty/laboratory technician immediately.
- Turn off the machine once you have finished using it.
- Maintain silence while working in the lab.
- Keep the Computer lab premises clean and tidy.
- Place backpacks under the table or computer counters.
- Treat fellow users of the laboratory, and all equipment within the laboratory, with the appropriate level of care and respect.

Don'ts

Students shall not

- Talk on cell phones in the lab.
- Eat or drink in the laboratory.
- Touch, connect or disconnect any plug or cable without the faculty/laboratory technician's permission.
- Install or download any software or modify or delete any system files on any

lab computers.

- Read or modify other users' files.
- Meddle with other users files. Leave their personal belongings unattended. We are not responsible for any theft.

Course Objectives:

1. Optimize business decisions and create competitive advantage with Cloud and Big data analytics
2. Implement best practices for Hadoop development.
3. Data management and its analysis for storing and sorting is viewed.
4. Impart the architectural concepts of Hadoop and introducing map reduce paradigm.

Course Outcomes: At the end of the course, student will be able to:

CO1	Configure and run an application with a single datacenter in cloud analyst.
CO2	Analyze working of multiple virtual machines and datacenters on cloud in Cloud Analyst.
CO3	Summarize different load balancing algorithms used in Cloud Analyst.
CO4	Analyze the efficiency of various distributed file systems in cloud Environment
CO5	Understand Docker Architecture and its role in Cloud Computing

CLOUD COMPUTING LAB

Laboratory Manual

Sub Code: 22CS62

Experiment No.		Contents of the Experiment	Hours	COs
1.	a	A study on Cloud Analyst tools withs its Steps tp install and Run Cloud Analyst	1	CO1
	b	Write a procedure to launch virtual machine using openstack	1	
2.	a	In given cloud simulator to host a simple web application with the below given configurations i. six user bases and one datacenter with 50 virtual machines each with 1024 Mb of memory and processor speed as 100 MIPS. ii. four user bases and one datacenter with 25 virtual machines each with 1024 Mb of memory and processor speed as 100 MIPS. Analyze the average, minimum and maximum response time and datacenter processing time and write the findings.	1	CO2
	b	A case study on the workflow of Amazon Web Services.	1	
3.	a	In given cloud simulator to host a simple web application on cloud with the configurations given below i. two datacenters with 50 VM's each with 1024 Mb memory and processor speed as 100 MIPS. Set the number of user bases to 6. Conduct experimentation by changing different service broker policies to analyze the average, minimum and maximum response time and total datacenter processing time. Also, analyze the total cost required to run the application.	1	CO3
	b	A Case Study: The GrepTheWeb Application	1	
4	a	In given cloud simulator to host a simple web application on cloud with the configurations given below i. two datacenters with 50 VM's each with 1024 Mb memory and processor speed as 100 MIPS. Set the number of user bases to 6. Conduct experimentation by changing different load balancing policies and conclude which load balancing policy is better by clearly stating the reasons.	1	CO3

	b	A Case study on Cloud Computing Networking: Challenges and Opportunities for Innovations.	1	
5	a	Find a procedure to transfer the files from one virtual machine to another virtual machine.	1	C04
	b	A case study on Performance Evaluation of Distributed File Systems in Cloud Environments	1	
6	a	Docker containers Installing Docker Engine on EC2 instance.	1	C05
	b	Basic Docker commands. Understanding docker images and building images using Dockerfile. Exposing ports for web microservices, creating and using docker networks. Data persistence using docker volumes.	1	

Experiment 1:

Introduction to Cloud Analyst

“CloudAnalyst” is a simulator used for simulating large scaled applications along with a novel approach for such research oriented studies. there are several toolkits that can be used to model a simulated environment to study the behaviour of a large scaled application on the Internet. But having an easy to use tool with a level of visualisation capability is even better than just a toolkit. Such a tool separates the simulation experiment set up exercise from a programming exercise and enables a modeler to concentrate on the simulation parameters rather than the technicalities of programming. A graphical output of the simulation results enables the results to be analyzed more easily and more efficiently and it may also help in quickly highlighting any problems with the performance and accuracy of the simulation logic.

Features of the Simulator
2.2.1 Ease of use
Ease of setting up and executing a simulation experiment is the main point of having a simulation tool. The simulator needs to provide an easy to use graphical user interface which is intuitive yet comprehensive.
2.2.2 Ability to define a simulation with a high degree of configurability and flexibility. Perhaps the most important feature is the level of configurability the tool can provide. A simulation, especially of the nature of modelling something as complex as an Internet Application depends on many parameters and most of the time the values for those parameters need to be assumed. Therefore it is important to be able to enter and change those parameters quickly and easily and repeat simulations.
2.2.3 Graphical output
A picture is said to be worth a thousand words. Graphical output in the form of tables and charts is highly desirable to summarise the potentially large amount of statistics that is collected during the simulation. Such effective presentation helps in identifying the important patterns of the output parameters and helps in comparisons between related parameters.
2.2.4 Repeatability
Repeatability of experiments is a very important requirement of a simulator. The same experiment with the same parameters should produce similar results each time the simulation is executed. Otherwise the simulation becomes just a random

sequence of events rather than a controlled experiment. It is also helpful to be able to save an experiment (the set of input parameters) as a file and also be able to save the results of an experiment as a file.

2.2.5 Ease of extension

As already mentioned simulating something like the Internet is a complex task and it is 9 unlikely a 100% realistic simulation framework and a set of input parameters can be achieved in a few attempts. Therefore the simulator is expected to evolve continuously rather than a program that is written once and for all and then used continuously. Therefore the simulator architecture should support extensions with minimal effort with suitable frameworks.

2.3 Simulation Output / What is being Measured

Following are the statistical measures produced as output of the simulation in the initial version of the simulator.

• Response time of the simulated application
 o Overall average, minimum and maximum response time of all user requests simulated
 o The response time broken down by user groups, located within geographical regions
 o The response time further broken down by the time showing the pattern of change over the duration of a day

• The usage patterns of the application
 o How many users use the application at what time from different regions of the world, and the overall effect of that usage on the data centers hosting the application

• The time taken by data centers to service a user request
 o The overall request processing time for the entire simulation
 o The average, minimum and maximum request processing time by each data center
 o The response time variation pattern during the day as the load changes

2.4 Technologies Used

• Java – The simulator is developed 100% on Java platform, using Java SE 1.6.
• Java Swing – The GUI component is built using Swing components.
• CloudSim – CloudSim features for modelling data centers is used in CloudAnalyst.
• SimJava – Sim Java is the underlying simulation framework of CloudSim and some features of SimJava are used directly in CloudAnalyst.

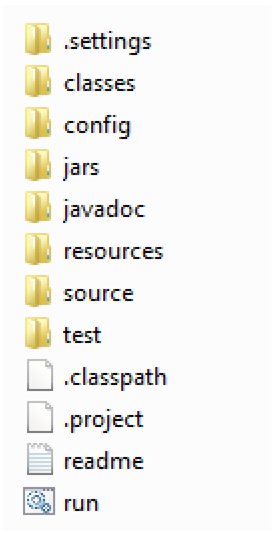
CloudAnalyst Design

The CloudAnalyst is built on top of CloudSim tool kit, by extending CloudSim functionality with the introduction of concepts that model Internet and Internet Application behaviours.

Steps tp install and Run Cloud Analyst

1. Download installer package from http://www.cloudbus.org/cloudsim/CloudAnalyst.zi

2. Extract Files from the compressed package file which will give following folder structure

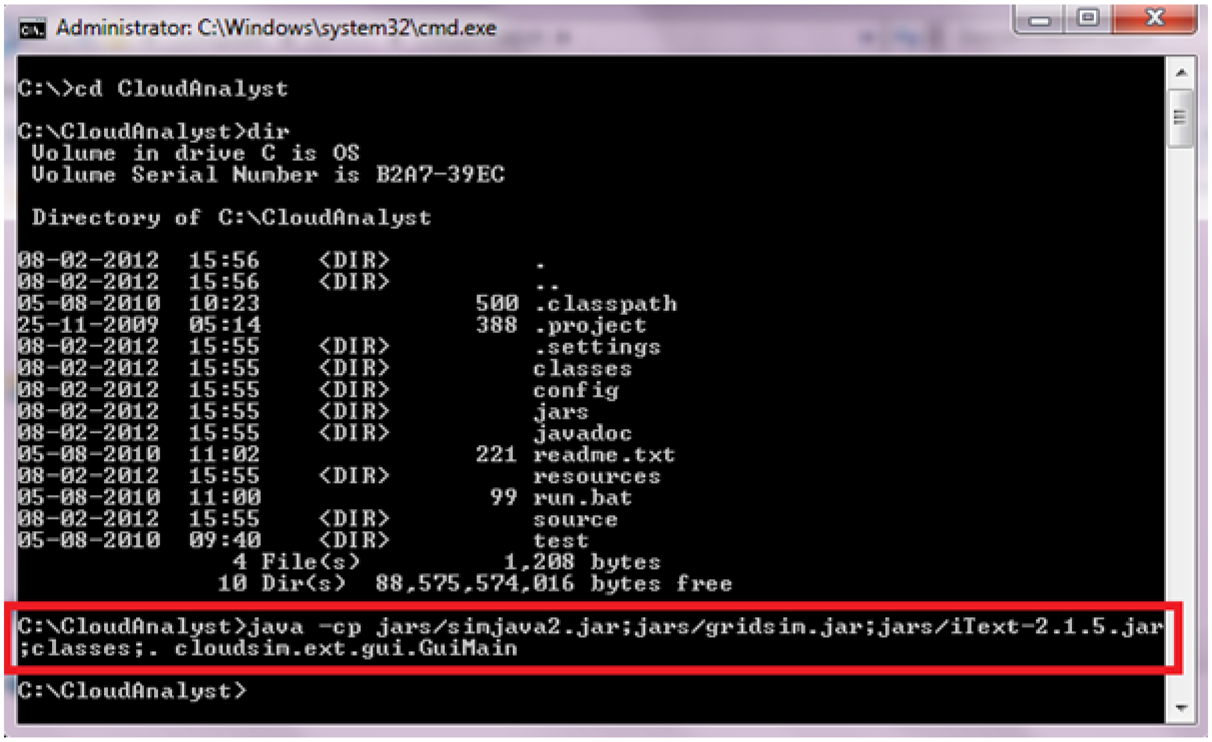


.settings
classes
config
jars
javadoc
resources
source
test
.classpath
.project
readme
run

3. To start the simulator, you can got Command line then use the following command on the command prompt as shown in the screenshot below

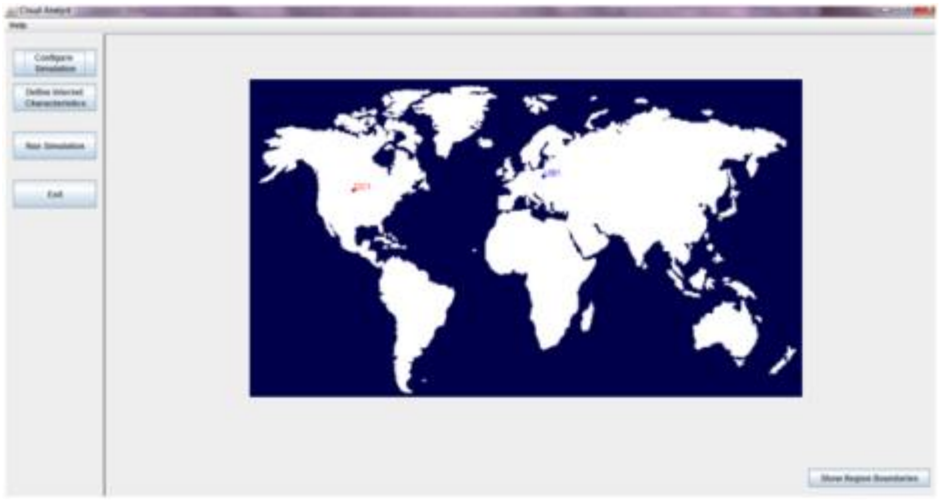
C:\CloudAnalyst>java – cp jars/simjava2.jar;jars/gridsim.jar;jars/iText-2.1.5.jar;classes;. Cloudsim.ext.gui.GuiMain

n

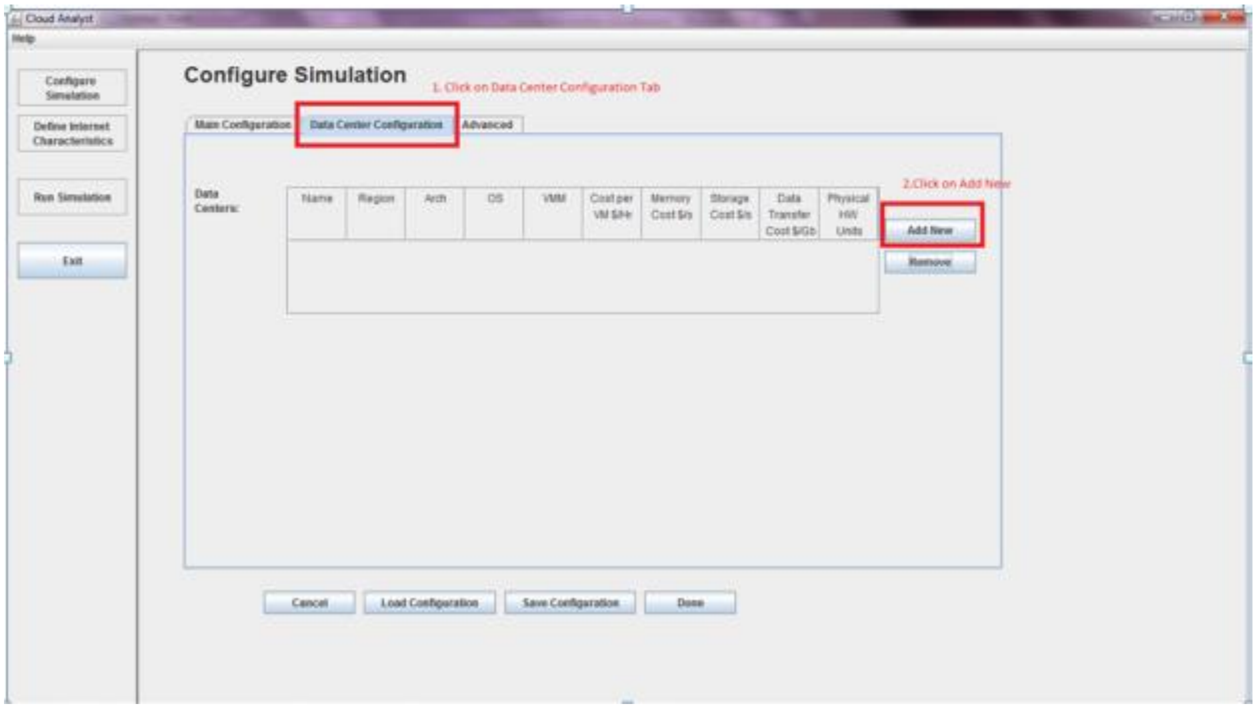

```
C:\>cd CloudAnalyst

C:\CloudAnalyst>dir
 Volume in drive C is OS
 Volume Serial Number is B2A7-39EC

 Directory of C:\CloudAnalyst

08-02-2012  15:56    <DIR>          .
08-02-2012  15:56    <DIR>          ..
05-08-2010  10:23               500 .classpath
25-11-2009  05:14               388 .project
08-02-2012  15:55    <DIR>          .settings
08-02-2012  15:55    <DIR>          classes
08-02-2012  15:55    <DIR>          config
08-02-2012  15:55    <DIR>          jars
08-02-2012  15:55    <DIR>          javadoc
05-08-2010  11:02               221 readme.txt
08-02-2012  15:55    <DIR>          resources
05-08-2010  11:00                99 run.bat
08-02-2012  15:55    <DIR>          source
05-08-2010  09:40    <DIR>          test
               4 File(s)          1,208 bytes
              10 Dir(s)  88,575,574,016 bytes free

C:\CloudAnalyst>java -cp jars/simjava2.jar;jars/gridsim.jar;jars/iText-2.1.5.jar
;classes;. cloudsim.ext.gui.GuiMain

C:\CloudAnalyst>
```

Alternatively you can also click on **run.bat** file and click **Done**!!

4. The welcome screen of the simulator looks like the following image.



5. To configure the parameters for experimentation click on “**Show Region Boundaries”**. Here you can datacenters and other physical hardware details of the datacenter as shown in the following screenshots.



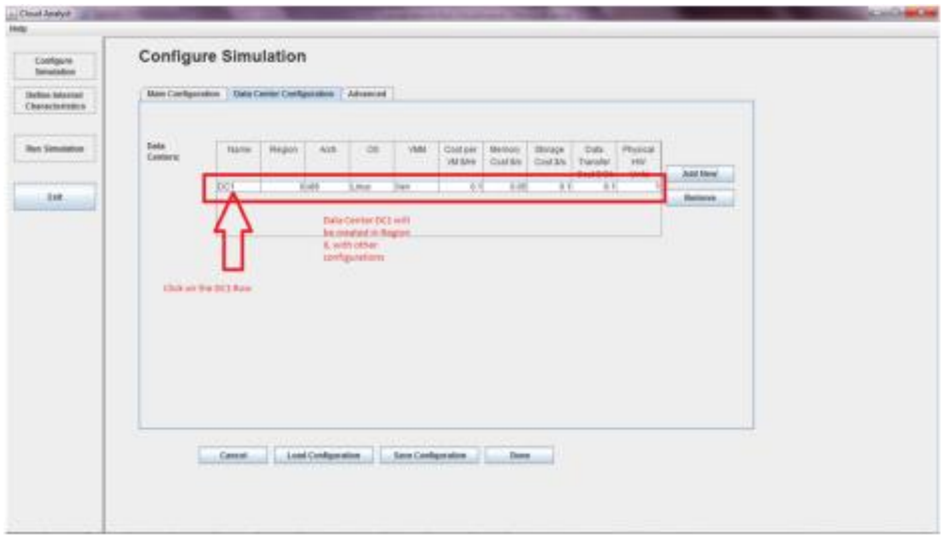
Configure Simulation

1. Click on Data Center Configuration Tab

2.Click on Add New

Data Centers:

Name	Region	Arch	OS	VMM	Cost per VM $/Hr	Memory Cost $/s	Storage Cost $/s	Data Transfer Cost $/Gb	Physical HW Units



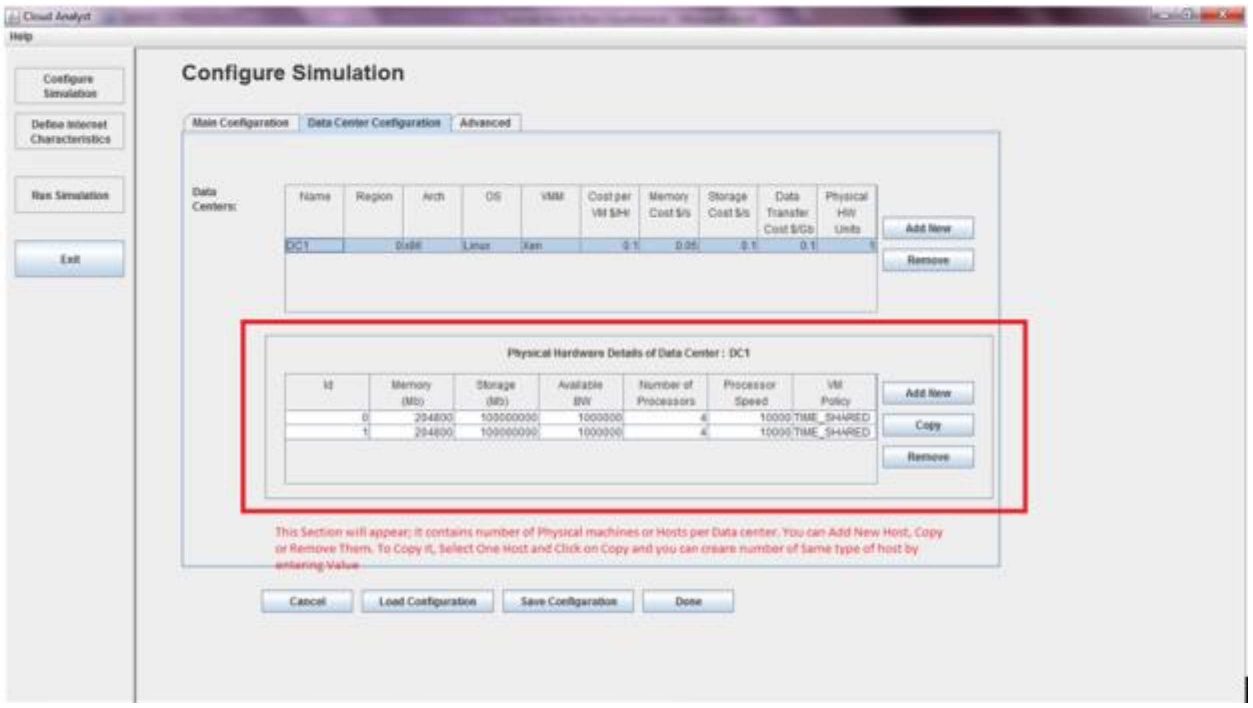
Configure Simulation

Data Centers:

Name	Region	Arch	OS	VMM	Cost per VM $/Hr	Memory Cost $/s	Storage Cost $/s	Data Transfer Cost $/Gb	Physical HW Units
DC1		x86	Linux	Xen	0.1	0.05	0.1	0.1	1

Data Center DC1 will be created in Region 0, with other configurations

Click on the DC1 Row



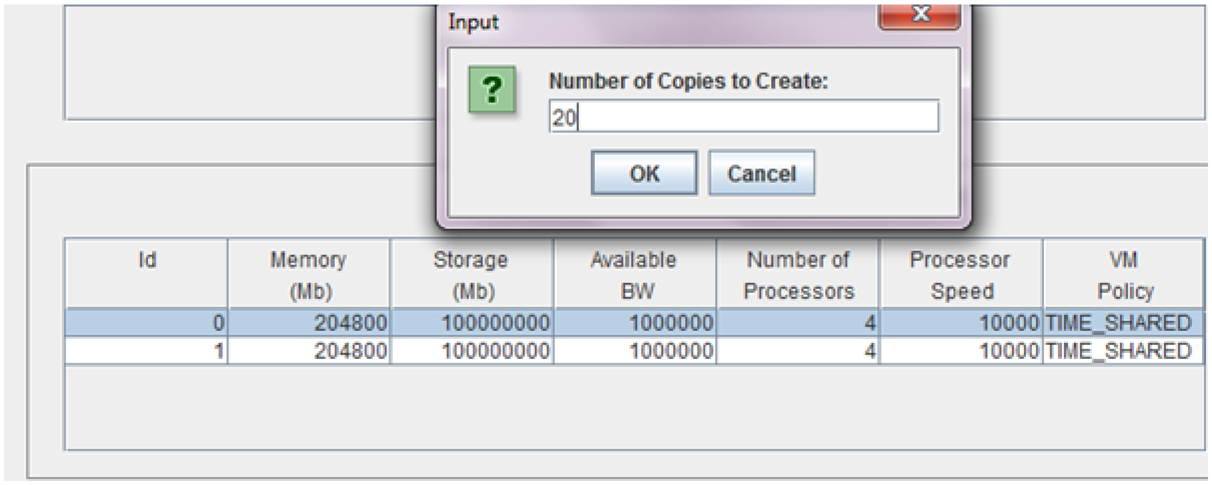
Data Centers:

Name	Region	Arch	OS	VMM	Cost per VM $/Hr	Memory Cost $/s	Storage Cost $/s	Data Transfer Cost $/Gb	Physical HW Units
DC1	0	x86	Linux	Xen	0.1	0.05	0.1	0.1	1

Physical Hardware Details of Data Center : DC1

Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	204800	100000000	1000000	4	10000	TIME_SHARED
1	204800	100000000	1000000	4	10000	TIME_SHARED

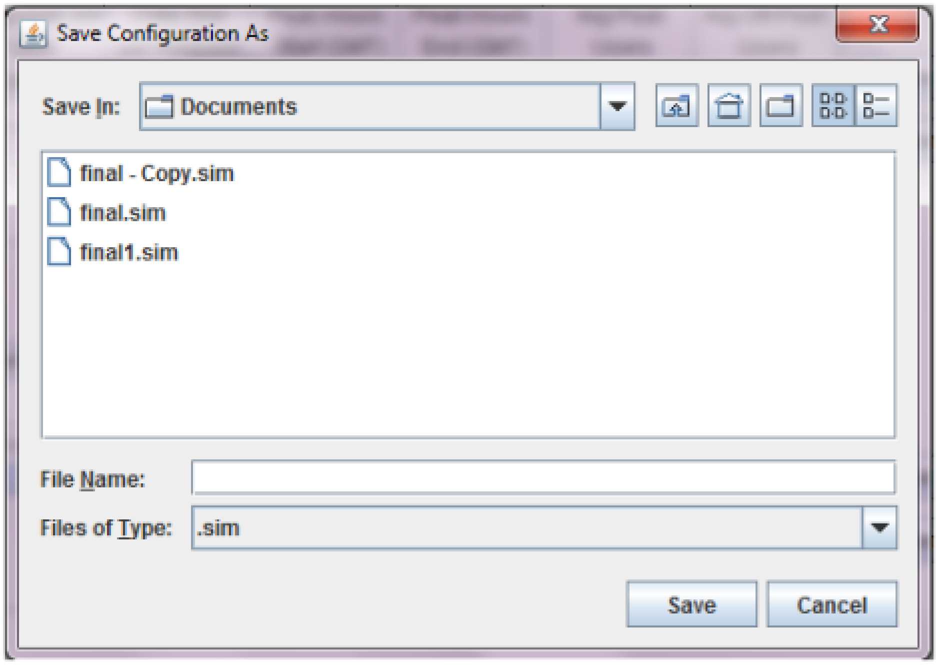
This Section will appear; it contains number of Physical machines or Hosts per Data center. You can Add New Host, Copy or Remove Them. To Copy it, Select One Host and Click on Copy and you can creare number of Same type of host by entering Value



Number of Copies to Create: 20

Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	204800	100000000	1000000	4	10000	TIME_SHARED
1	204800	100000000	1000000	4	10000	TIME_SHARED

6. You can Save this Configuration as well in case you want to use it later. It is stored as .sim file. XML data is generated and saved as Sim file.



Save In: Documents

final - Copy.sim
final.sim
final1.sim

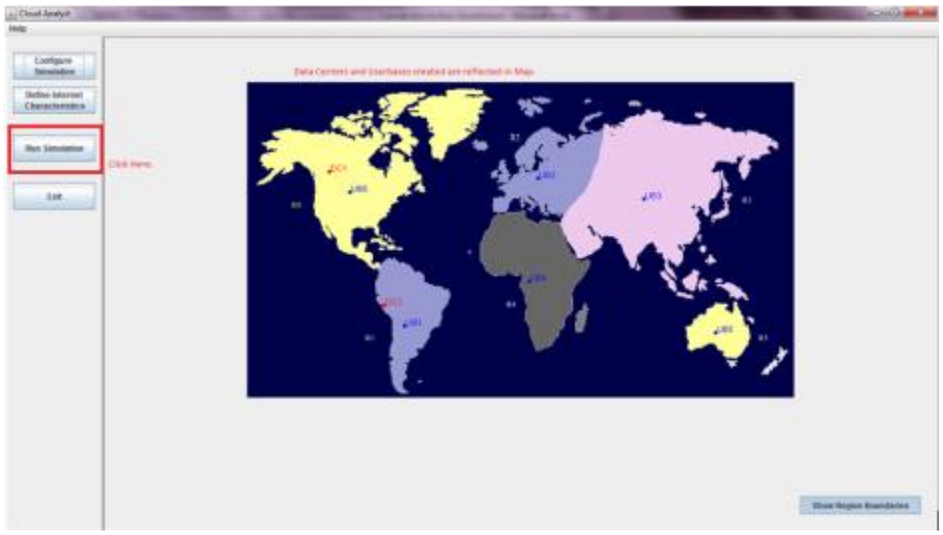
File Name:

Files of Type: .sim

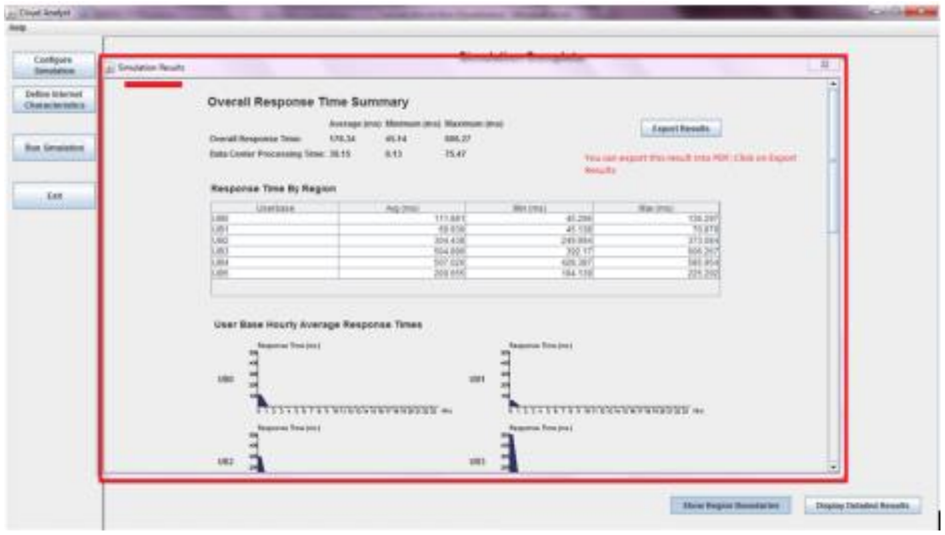
Saved configurations can be loaded anytime easily into Cloud Analyst.

7. Once your are done with Configuration; click on Done.

8. To run the simulation click on Run Simulation



9. Simulation Results Window will Open


Overall Response Time Summary

	Average (ms)	Minimum (ms)	Maximum (ms)
Overall Response Time:	170.34	45.14	686.27
Data Center Processing Time:	38.15	0.13	75.47

You can export this result into PDF. Click on Export Results

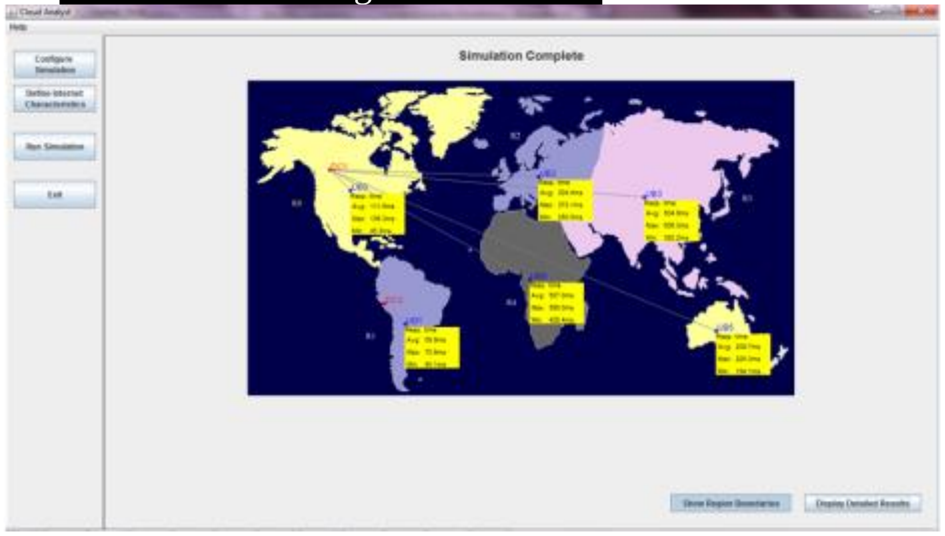
Response Time By Region

Userbase	Avg (ms)	Min (ms)	Max (ms)
UB0	111.681	45.206	138.297
UB1	49.938	45.138	70.878
UB2	309.438	249.884	373.084
UB3	504.898	392.17	606.267
UB4	507.028	426.387	585.854
UB5	209.655	184.138	225.292

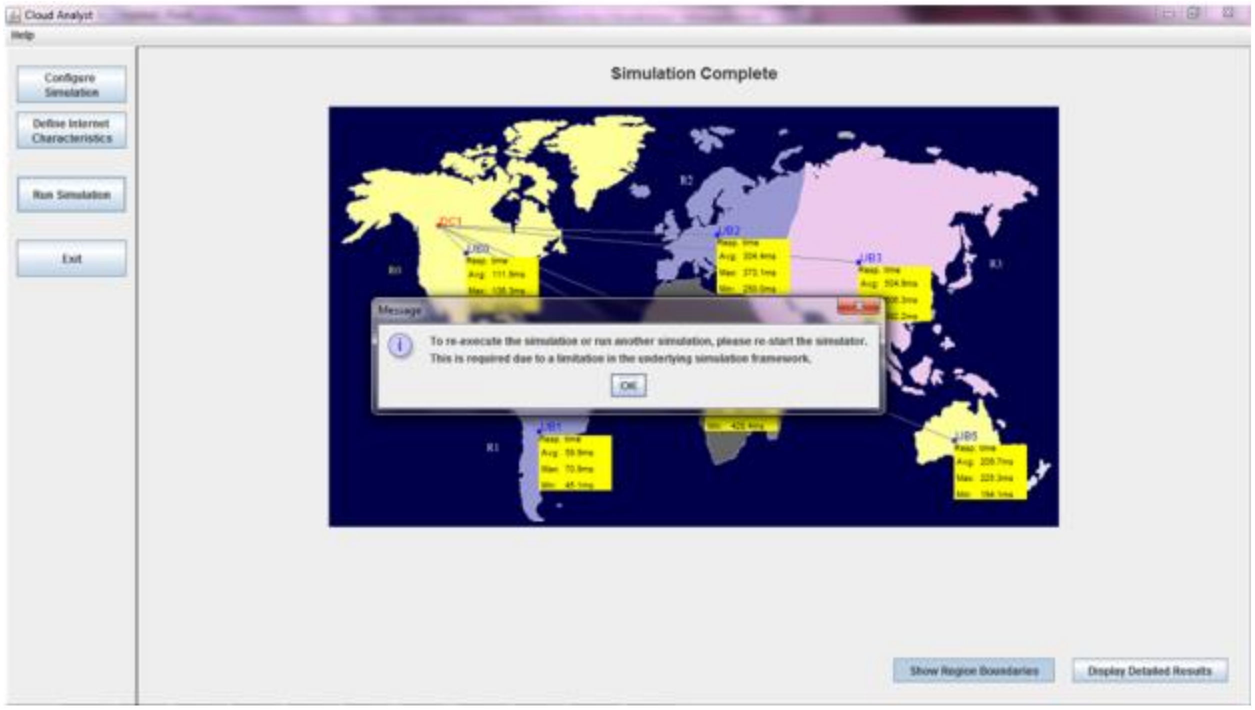
User Base Hourly Average Response Times

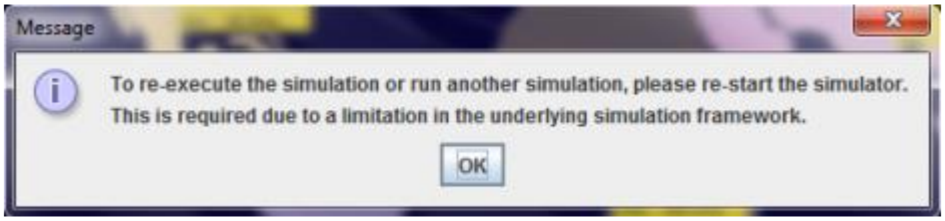
Then click on “Close it.”

10. Main Window will give all statistics



11. If you will try to run Simulation again then it will give Error





To re-execute the simulation or run another simulation, please re-start the simulator.
This is required due to a limitation in the underlying simulation framework.

12. Click on Exit & restart the simulator.

Experiment 2:

In given cloud simulator to host a simple web application with the below given configurations

i. six user bases and one datacenter with 50 virtual machines each with 1024 Mb of memory and processor speed as 100 MIPS.

ii. four user bases and one datacenter with 25 virtual machines each with 1024 Mb of memory and processor speed as 100 MIPS.

Analyze the average, minimum and maximum response time and datacenter processing time and write the findings.

Procedure for (i):

Step1: Start the Cloud Analyst simulator by clicking on **run.bat** file.

Step 2:Once the simulator starts navigate to **Configure Simulation** tab.

Step 3:In Main Configuration tab, add six user bases by clicking add new button in the **User Bases** table.

Step4: Set the memory to 1024 Mb in the **Application Deployment Configuration** table.

Step 5: Set the number of VM's to 50 in the **Application Deployment Configuration.**

Step 5: Navigate to **Data Center Configuration** tab and click on the Datacenter name ie DC1.

Step 6: The **physical hardware details of Data Center** table is displayed where you have to set the processor speed to 100 MIPS.

Step 7: Click on **Save Configuration** to save the simulation by naming it.

Step 8: Click **Done.**

Step 9: Click on **Run Simulation** to run the experiment.

Step 10: Wait for the results to be loaded.

Step 11: Note down the average, maximum and minimum response time for the user bases & datacenter processing times.

Step 12: Draw the corresponding graphs.

Procedure for (ii):

Step1: Start the Cloud Analyst simulator by clicking on **run.bat** file.

Step 2:Once the simulator starts navigate to **Configure Simulation** tab.

Step 3:In Main Configuration tab, add four user bases by clicking add new button in the **User Bases** table.

Step4: Set the memory to 1024 Mb in the **Application Deployment Configuration** table.

Step 5: Set the number of VM's to 25 in the **Application Deployment Configuration.**

Step 5: Navigate to **Data Center Configuration** tab and click on the Datacenter name ie DC1.

Step 6: The **physical hardware details of Data Center** table is displayed where you have to set the processor speed to 100 MIPS.

Step 7: Click on **Save Configuration** to save the simulation by naming it.

Step 8: Click **Done.**

Step 9: Click on **Run Simulation** to run the experiment.

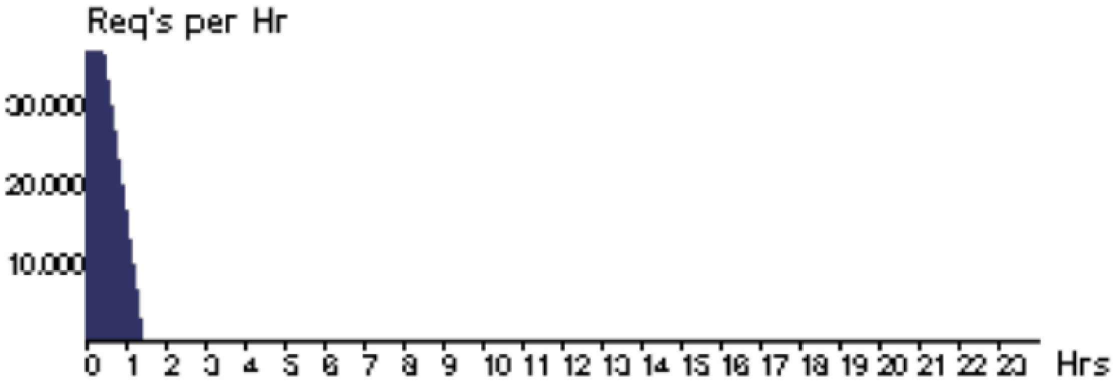
Step 10: Wait for the results to be loaded.

Step 11: Note down the average, maximum and minimum response time for the user bases & datacenter processing times.

Step 12: Observe & note the corresponding graphs.

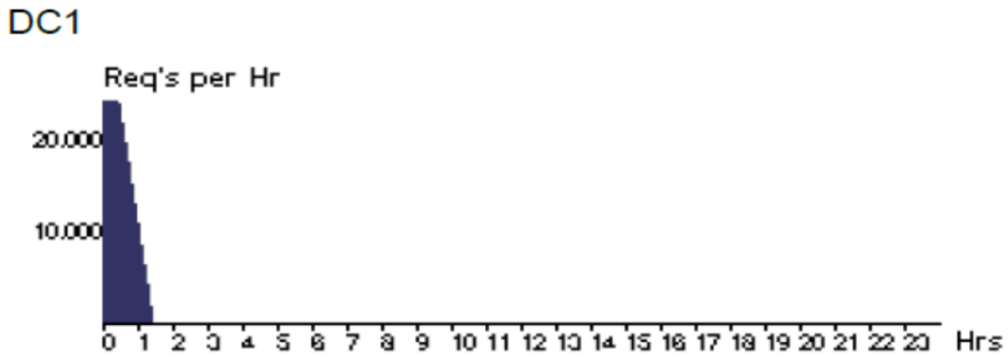
Output of (i):

	Avg (ms)	**Min (ms)**	**Max (ms)**
Overall response time:	319.17	234.64	450.60
Data Center processing time:	19.52	0.07	124.06



Output of (ii)

	Avg (ms)	**Min (ms)**	**Max (ms)**
Overall response time:	300.59	232.81	375.33
Data Center processing time:	0.55	0.04	0.86



Analysis of the experimentation:

The response times & datacenter processing time in (i)   is more than (ii) as we have more number of user bases in (i).

Outcome:

We learnt that the user base, which is collection of users, plays a vital role in generating requests. As the number of user bases increase the time taken to process the user request also increases. The datacenter which is used to process the user requests generated by the user base, needs more processing time. Hence the response time of the datacenter also increases.

Experiment 3:

In given cloud simulator to host a simple web application on cloud with the configurations given below

i. two datacenters with 50 VM's each with 1024 Mb memory and processor speed as 100 MIPS. Set the number of user bases to 6.

Conduct experimentation by changing different service broker policies to analyze the average, minimum and maximum response time and total datacenter processing time. Also, analyze the total cost required to run the application.

Procedure for (i):

Step1: Start the Cloud Analyst simulator by clicking on **run.bat** file.

Step 2:Once the simulator starts navigate to **Configure Simulation** tab.

Step 3:In **Main Configuration** tab, add six user bases by clicking add new button in the **User Bases** table.

Step 4: Navigate to **Data Center Configuration** tab and click on the **Add New** to add second datacenter.

Step5: Navigate back to **Main Configuration** tab, in the **Application Deployment Configuration** table click on **Add New** to configure the datacenter.

Step 6: Configure the datacenters DC1 & DC2 with 50 VM's each and Memory of 1024 Mb each.

Step 7: Configure the **Service Broker Policy** to **Closest Datacenter** for first run. This is the default policy.

Step 8: Click on **Save Configuration** to save the simulation by naming it.

Step 9: Click **Done.**

Step 11: Click on **Run Simulation** to run the experiment.

Step 12: Wait for the results to be loaded.

Step 13: Note down the average, maximum and minimum response time for the user bases & datacenter processing times.

Step 14: Note down the cost computed to run the simulation.

Note:

- For second run all the above mentioned steps remain the same except Step 8, where you have to change the service broker policy to **Optimise Response time**
- For third run all the above mentioned steps remain the same except Step 8, where you have to change the service broker policy to **Reconfigure Dynamically with Load.**

Output

(i)Service Policy: Closest Datacenter

	Avg (ms)	**Min (ms)**	**Max (ms)**
Overall response time:	362.88	226.27	489.53
Data Center processing time:	63.33	0.13	125.56

Cost

Total Virtual Machine Cost: $10.04

Total Data Transfer Cost : $0.38

Grand Total : $10.42

(ii)Service Policy: Optimise Response Time

	Avg (ms)	**Min (ms)**	**Max (ms)**

Overall response time:	301.21	226.26	380.77
Data Center processing time:	1.46	0.13	1.82

Cost

Total Virtual Machine Cost: $10.04

Total Data Transfer Cost : $0.38

Grand Total : $10.42

(iii)Service Policy: Reconfigure Dynamically with Load

	Avg (ms)	**Min (ms)**	**Max (ms)**
Overall response time:	303.83	226.41	7503.02
Data Center processing time:	4.17	0.13	7178.01

Cost

Total Virtual Machine Cost: $15.40

Total Data Transfer Cost : $0.38

Grand Total : $15.79

Analysis:

The data center’s response time & processing time for each of the service policy is different. We find that the response time and processing time is efficient when we use optimize response time service policy.

Outcome of experimentation:

In this experiment we noted the response time and processing time of two datacenters by varying the service policies. It was noted that when we use optimize response time service policy, the response time and processing time of the datacenter is better compared to the other two service policies.

Experiment 4:

In given cloud simulator to host a simple web application on cloud with the configurations given below

i. two datacenters with 50 VM's each with 1024 Mb memory and processor speed as 100 MIPS. Set the number of user bases to 6.

Conduct experimentation by changing different load balancing policies and different Service Policies ,conclude which load balancing policy is better by clearly stating the reasons.

Procedure for (i):

Step1: Start the Cloud Analyst simulator by clicking on **run.bat** file.

Step 2:Once the simulator starts navigate to **Configure Simulation** tab.

Step 3:In **Main Configuration** tab, add six user bases by clicking add new button in the **User Bases** table.

Step 4: Navigate to **Data Center Configuration** tab and click on the **Add New** to add second datacenter.

Step5: Navigate back to **Main Configuration** tab, in the **Application Deployment Configuration** table click on **Add New** to configure the datacenter.

Step 6: Configure the datacenters DC1 & DC2 with 50 VM's each and Memory of 1024 Mb each.

Step 7: Navigate to **Advanced** tab, to change the load balancing policy for each run.(**please refer Note below**)

Step 8: Click on **Save Configuration** to save the simulation by naming it.

Step 9: Click **Done.**

Step 11: Click on **Run Simulation** to run the experiment.

Step 12: Wait for the results to be loaded.

Step 13: Note down the average, maximum and minimum response time for the user bases & datacenter processing times.

Step 14: Note down the cost computed to run the simulation.

Note:

- For first run all the above mentioned steps remain the same except Step 8, where you have to change the load balancing policy to **Round Robin**
- For second run all the above mentioned steps remain the same except Step 8, where you have to change the load balancing policy to **Equally Spread Current Execution Load**

Output

(i) Load balancing policy: Round Robin

	Avg (ms)	**Min (ms)**	**Max (ms)**
Overall response time:	299.95	36.33	720.08
Data Center processing time:	7.84	0.07	127.79

(ii) Load balancing policy: Equally Spread Current Execution Load

	Avg (ms)	**Min (ms)**	**Max (ms)**
Overall response time:	301.10	226.26	366.52

Data Center processing time:	1.45	0.13	1.82

Outcome of experimentation

The analysis of this experiment shows that the response time and data center processing time using equally shared current execution load balancing algorithm is better than Round Robin load balancing algorithm.

Experiment 5:

Aim:

To Find a procedure to transfer the files from one virtual machine to another virtual machine.

Steps:

1. You can copy few (or more) lines with copy & paste mechanism. For this you need to share clipboard between host OS and guest OS, installing Guest Addition on both the virtual machines (probably setting bidirectional and restarting them). You copy from guest OS in the clipboard that is shared with the host OS. Then you paste from the host OS to the second guest OS.
2. You can enable drag and drop too with the same method (Click on the machine, settings, general, advanced, drag and drop: set to bidirectional)
3. You can have common Shared Folders on both virtual machines and use one of the directory shared as buffer to copy. Installing Guest Additions you have the possibility to set Shared Folders too. As you put a file in a shared folder from host OS or from guest OS, is immediately visible to the other. (Keep in mind that can arise some problems for date/time of the files when there are different clock

settings on the different virtual machines). If you use the same folder shared on more machines you can exchange files directly copying them in this folder.

4. You can use usual method to copy files between 2 different computer with client-server application. (e.g. scp with sshd active for linux, winscp... you can get some info about SSH servers e.g. here) You need an active server (sshd) on the receiving machine and a client on the sending machine. Of course you need to have the authorization setted (via password or, better, via an automatic authentication method). Note: many Linux/Ubuntu distribution install sshd by default: you can see if it is running with pgrep sshd from a shell. You can install with sudo apt-get install openssh-server.

5. You can mount part of the file system of a virtual machine via NFS or SSHFS on the other, or you can share file and directory with Samba. You may find interesting the article Sharing files between guest and host without VirtualBox shared folders with detailed step by step instructions.

You should remember that you are dialling with a little network of machines with different operative systems, and in particular:

• Each virtual machine has its own operative system running on and acts as a physical machine.

• Each virtual machine is an instance of a program owned by an user in the hosting operative system and should undergo the restrictions of the user in the hosting OS. E.g Let we say that Hastur and Meow are users of the hosting machine, but they did not allow each other to see their directories (no read/write/execute authorization). When each of them run a virtual machine, for the hosting OS those virtual machine are two normal programs owned by Hastur and Meow and cannot see the private directory of the other user. This is a restriction due to the hosting OS. It's easy to overcame it: it's enough to give authorization to read/write/execute to a directory or to chose a different directory in which both users can read/write/execute.

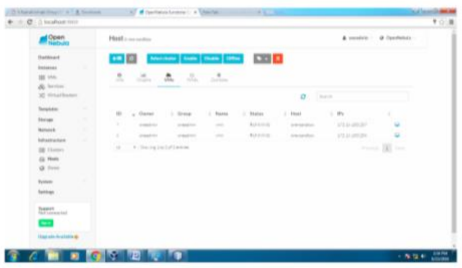
• Windows likes mouse and Linux fingers. :-) I mean I suggest you to enable Drag & drop to be cosy with the Windows machines and the Shared folders or to be cosy with Linux. When you will need to be fast with Linux you will feel the need of ssh-keygen and to Generate once SSH Keys to copy files on/from a remote machine without writing password anymore. In this way it functions bash auto-completion remotely too!

PROCEDURE:

Steps:

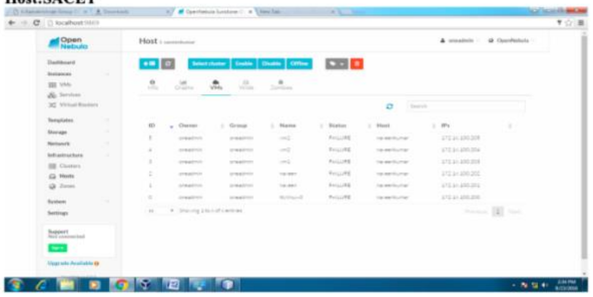
1. Open Browser, type localhost:9869
2. Login using username: oneadmin, password: opennebula
3. Then follow the steps to migrate VMs a. Click on infrastructure b. Select clusters and enter the cluster name c. Then select host tab, and select all host d. Then select Vnets tab, and select all vnet e. Then select datastores tab, and select all datastores f. And then choose host under infrastructure tab g. Click on + symbol to add new host, name the host then click on create.
4. on instances, select VMs to migrate then follow the stpes a. Click on 8th icon ,the drop down list display b. Select migrate on that ,the popup window display c. On that select the target host to migrate then click on migrate.

Before migration Host:SACET Host:one-sandbox After Migration: Host:one-sandbox Host:SACET APPLICATIONS: Easily migrate your virtual machine from one pc to another.



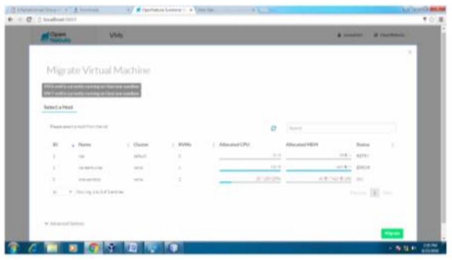
Before migration

Host:SACET



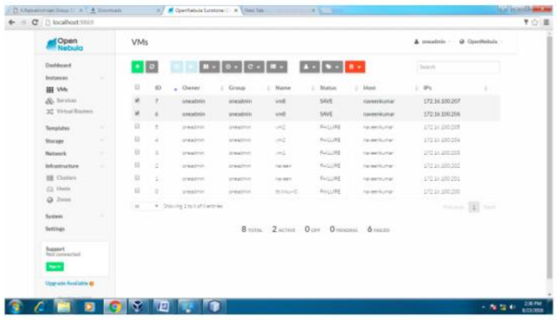
Host

ID	Owner	Group	Name	Status	Host	IPs
5	oneadmin	oneadmin	vm2	FAILURE	naveenkumar	[illegible]
4	oneadmin	oneadmin	vm2	FAILURE	naveenkumar	[illegible]
3	oneadmin	oneadmin	vm1	FAILURE	naveenkumar	[illegible]
2	oneadmin	oneadmin	naveen	FAILURE	naveenkumar	[illegible]
1	oneadmin	oneadmin	naveen	FAILURE	naveenkumar	[illegible]
0	oneadmin	oneadmin	[illegible]	FAILURE	naveenkumar	[illegible]



Migrate Virtual Machine

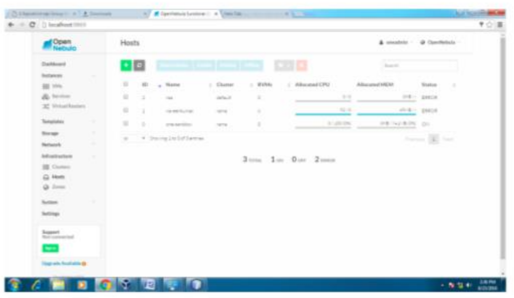
After Migration:



VMs

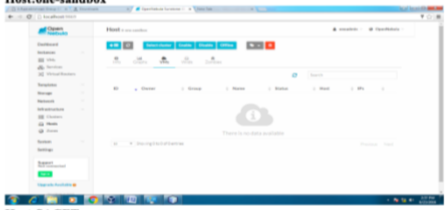
ID	Owner	Group	Name	Status	Host	IPs
7	oneadmin	oneadmin	vm8	SAVE	naveenkumar	172.16.100.207
6	oneadmin	oneadmin	vm8	SAVE	naveenkumar	172.16.100.206
5	oneadmin	oneadmin	vm2	FAILURE	naveenkumar	[illegible]
4	oneadmin	oneadmin	vm2	FAILURE	naveenkumar	[illegible]
3	oneadmin	oneadmin	vm1	FAILURE	naveenkumar	[illegible]
2	oneadmin	oneadmin	naveen	FAILURE	naveenkumar	[illegible]
1	oneadmin	oneadmin	naveen	FAILURE	naveenkumar	[illegible]
0	oneadmin	oneadmin	[illegible]	FAILURE	naveenkumar	[illegible]

8 TOTAL 2 ACTIVE 0 OFF 0 PENDING 6 FAILED

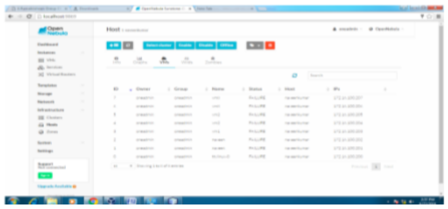


Hosts

Host:one-sandbox



Host:SACET



Result: Thus the file transfer between VM was successfully completed

Experiment 6

Docker containers

a. Installing Docker Engine on EC2 instance.
b. Basic Docker commands. Understanding docker images and building images using Dockerfile. Exposing ports for web microservices, creating and using docker networks. Data persistence using docker volumes.

1. Installing Dock
2. er Engine on AWS EC2 instance
    a. Screenshot of running docker hello-world
3. Docker images and Dockerfiles
    a. Screenshot of C Program successfully run inside container.
4. Exposing ports,docker networks
    a. Sample.html showing the web page on the browser. Screenshot should show public dns of the EC2 host of docker.
    b. Screenshot of docker container running nginx (terminal of EC2 host)
    c. Screenshot of python application successfully writing and reading from the MongoDB database
    d. Screenshot showing mongodb being run within network(docker command has to be clearly highlighted)
    e. Screenshot showing python file being run within network and successfully writing and reading from MongoDB(docker command has to be clearly highlighted)
5. Docker compose
    a. Screenshot of python-mongodb application running as a docker compose application(logs of the application)
    b. Screenshot of 3 python application writes and reads from MongoDB after scaling the python application.

6. Docker volumes (This task is optional, No bonus marks will be given for submitting this screenshot)
    a. Screenshot clearly showing command run to mount the host volume and manually running the python application inside the container

Few key points to note:

1. All required files **except** Dockerfile(s) have been given in the drive folder(task wise) where this manual is uploaded.
2. It is very important to go through all reference material attached in this manual , as this will help you understand and debug the lab tasks.
3. Most of this lab focuses on 2 aspects building the right Dockerfile and running the right command.
4. Apart from the attached resources, you can always refer to the official docker documentation. The docker documentation is well maintained and should help you through all your tasks.https://docs.docker.com/
5. Additional resources have been given at the end of the manual.
6. When you run docker commands in the foreground, you cannot access the command prompt in which case press **[Ctrl+C]** and continue with the next step or run docker in the background mode by using –d option as explained in the demo video. https://www.tecmint.com/run-docker-container-in-background-detached-mode/

Understanding containers and Docker

Please watch the below youtube tutorial **mandatorily** before starting the lab, this will help kickstart your understanding of Docker and the commands.

Docker Basic Commands | Docker Commands with Examples | Docker Commands Tutorial | Intellipaat

Containers are the de-facto standard for creating,maintaining and deploying microservices. A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

Container engines are the technology/software that are used to run the containers. There are many container engines available, but the most popular and easy to use CE is *Docker*.

The following links will help you get started on why we need containers, how they are different from VMs and why Docker:

What is a Container? | App Containerization - **Must Read!**

What is Docker?

Task 1: Installing Docker Engine on EC2

Sub-tasks are:

1. Create an EC2 instance with the following specs:

AMI	Ubuntu Server 20.04 LTS (HVM), SSD Volume Type
Instance Type	t2.medium
Instance Details	Default values
Storage	Default values

2. Get keypair(pem) file and SSH into the instance.
3. Install docker engine on the instance , instructions are in the given link:
    a. Install Docker Engine on Ubuntu
4. Use the following link to make docker a “sudo-less” command:
    a. Post-installation steps for Linux
5. Verify that Docker Engine is installed correctly by running : docker run hello-world

Task 2: Docker images and docker files

In AWS EC2 instances we saw *AMI(Amazon Machine Images)*, which was basically the operating system and in case of snapshots, the current *state* of the VM. Docker images are similar, but are not the same. Docker images are “ready to go” meaning everything that needs to be installed(requirements) has already been taken care of when building the image and they don't need to be installed/taken care of again.

Docker files are used to create Docker images. Dockerfile are a series of steps that specify which base image to use, which files/folders to copy into the container, run which commands while starting up the container and which will be the main process of the container. We need to create dockerfiles, then build it into an actual runnable docker image.

Sub tasks:

1. Docker hub is a central public repository containing Docker images and documentation on how to use these base images. Explore the different images at : https://hub.docker.com/
2. The images in docker hub, need to be *pulled* into our EC2 instance in order to use them(Pulling is the equivalent of downloading the images onto you docker host).
    a. docker pull

Pull the following images onto your docker instance:

- Ubuntu 18.04
- Nginx
- Python
- MongoDB

Stick to the latest images. To find the images that exist on your instance you can run the docker images command. This shows the image name, when you pulled the image, the tags(versions) of the images.

3. Now that we have the images we need to actually run the containers,
    a. We can run a container using the docker run command.
    b. To list the *currently running* container we use the docker ps command

c. To stop a running container safely we use docker stop

d. Usually after a container is stopped it goes into a “exited state”, this can block some I/O operations needed by future containers and you will not be able to use the name of the exited container. What are the possible states for a docker container?

To safely and cleanly remove a container, use the docker rm command.

4. *Containers are light weight VMs*, this means we should be able to somehow interact with a running container, like a terminal.
    a. Start an interactive bash(shell) session with an ubuntu:18.04 as a base image. Explore around the container using the shell. **What is different and what is same as an ubuntu VM?**
        i. Docker 'run' command to start an interactive BaSH session
    b. Get a shell into an already running container
5. Let us now create our own Docker image using Ubuntu:18.04 as a base image. The image you will create will run a simple C program, after installing the GCC compiler. The Dockerfile must:
    a. Specify the base image as **Ubuntu:18.04**
    b. Copy the program.c file from your instance to the docker image.
    c. Update the “apt” repository and install the GCC compiler
    d. Compiler the C program.
    e. Run the ./a.out *command*

    Hint: What should the name of the dockerfile you are creating?

    Use the following C program as a base, only modify your SRN:

```
#include<stdio.h>
int main()
{
   printf("Running this inside a container !\n");
```

```
    printf("My SRN is <YOUR SRN HERE>\n");
}
```

6. After you have your Dockerfile, build the image using docker build and run the container.

References to help you get started:

- How do you write a Dockerfile?
- Docker Build: A Beginner's Guide to Building Docker Images
- How to Create a Docker Image From a Container

Task 3: Exposing ports,docker networks

Containers are also used to run web applications , they can be web servers such as Apache or Nginx, or web applications using REST APIs or any other web application. Similar to *Security Groups* for EC2 instances, we need to *expose* the right ports to access the container's web apps.

1. Download the sample HTML file from the Lab 5 drive folder , and modify your SRN.
2. Create a Dockerfile and then a docker image having an **nginx:latest** base image, and copying the html file into the default folder in the container. Build the docker image and **tag/name it your SRN**.
3. Run the docker container using the previously created docker image. Expose the HTTP port and check connectivity.
    a. Publishing/Exporting ports in a docker container
    b. Docker Container Tutorial #8 Exposing container ports

Hint: You are running a container on an EC2 instance. After exposing the port on the container , you might be able to access the nginx web server from within the EC2 instance, but not from the internet(your system). What needs to be configured for the AWS EC2 instance running your container?

In lab 3 , we saw VPC networks and how we can create a virtual network connectivity between virtual machines(EC2 instances). Similar networks need to be created for containers. We will explore connectivity without docker networks and see how docker networks make it much easier to connect within containers. To demonstrate this we will create a simple application using a python client and a MongoDB NoSQL server.

Note: You don't need to know how to use mongodb, code to use mongodb has been provided to you.

1. Run the mongodb container in a detached mode, exposing the default port(27017) of mongodb.
    a. Use the **mongo:latest** image.
    b. Name the container as **mongodb** while running the container. Naming a container
    c. Docker detached mode
2. Download sample.py from the drive folder. Modify the SRN wherever mentioned.
3. The MongoDB container is running, but we need to find out the *IP address* of the mongodb container. Find this IP using the docker inspect command. Modify this IP address in the sample.py file.
    a. Get a containers IP address
4. Create a Dockerfile, which:
    a. Uses **python** base image
    b. Updates the apt-repository
    c. Installs **pymongo** using pip (pymongo 3.11.2).
    d. Copies the sample.py from the instance to the container.
    e. Runs the python *command* , to run the python file.
5. Build the docker image using the above Dockerfile and run the container.
6. You should see that the data was correctly inserted and fetched from the database container.

The above tasks showed the connectivity between 2 containers *without* a network. This can cause problems as every time a container is created it *could possibly* have a different IP address. This is the issue *docker networks* tries to solve.

Docker Networking Options

Docker Networking | Docker bridge network deep dive | Container bridge drive

1. Create a docker bridge network , called **my-bridge-network.**
2. Stop the mongodb container you had created before and delete(rm) it. Run a mongodb container again, but now with the following parameters;
    a. network : **my-bridge-network**
    b. name: **mongodb**
    c. Exposed ports: **27017**
    d. Image: **mongo:latest**
3. Go back to sample.py , comment line 3 and uncomment line 4. We are now going to use the name of the database containers as the host name, leaving the ip address resolution to docker.
4. Build the python app docker image again as you did previously and run the container using the built image. You should see that insertion and retrieval have been done successfully.

Task 4: Docker compose

Till now, we have been using docker commands to start and stop containers,create docker networks etc. But this becomes challenging in a real environment when a single application may have 100s of containers(microservices) that need to talk to each other within a single network. While docker networks solve this problem partially, it still doesn't solve the issue of having to run multi-container applications and to automatically bootstrap these containers into the same application, this is where *Docker Compose* comes into the picture.

Docker compose helps bootstrap complex multi container applications, helps maintain the dependencies between them , create a network among these containers and even help scale the entire application entirely or particular containers within the application.

Overview of Docker Compose

"What, Why, How" Docker Compose?

We will use the same python-mongodb application to use docker compose and even scale containers within the application automatically. Docker compose uses YAML Files to specify the different *pieces(containers)* of the application, along with the required configurations. YAML files are EXTREMELY popular when it comes to bootstrapping complex applications not only in Docker!

Docker Compose in 12 Minutes

1. For the purposes of this lab make sure all the following files are in the same directory
    a. docker-compose.yml
    b. Dockerfile
    c. sample.py
2. Install docker compose using the following link:
    a. Install compose on linux systems
3. Go to the docker-compose.yml file and try to understand the syntax and what each line does.
4. Within the same directory, run: **docker-compose up**.
    a. Docker Compose command-line reference

What you see is that Docker compose has built your python application, started the mongodb server , created links internally between the containers (network) and started both the containers together as a *unified application*.

The python container exits, since its done with its utility of writing and reading to the database.

5. Now we will scale the python application, so that we have 3 containers of the python application, but keep only one container of the mongodb.

   a. docker-compose scale - Use this link as a reference and scale **only the python application. Use the correct "service" to scale**

Task 5: Docker volumes This task is for you to practice, No bonus marks will be given for submitting this screenshot

Docker volumes are an important and crucial concept. Volumes are the preferred mechanism for persisting data generated by and used by Docker containers. Volumes are either created dynamically by Docker or you can mount the host directory. In this task,we will focus on mounting a host volume.

Docker bind mount | Sharing data between host and Container

1. Create a new directory on the EC2 host , with the name as your SRN.
2. Add the following folders inside it:
   a. sample.py (A simple python application which reads from a text file called details.txt and prints them line by line)
   b. A text file called details.txt containing your name,srn,section,semester.
3. We will be using **Ubuntu:18.04** as a base image.
4. For the ubuntu container, mount the host volume you have created and create a bash shell into the container.
   a. Start a container with a bind mount
5. Inside the container run the python program to display contents of the list.