# CLOUD COMPUTING TUTORIAL

A REPORT SUBMITTED TO

## THE NATIONAL INSTITUTE OF ENGINEERING

(An Autonomous College)


In partial fulfilment for the award of degree of

**Bachelor of Engineering**
**In**
**Computer Science & Engineering**

Submitted By

**Abhilash Shreedhar Hegde (4NI19CS003)**
**Amruta Narayana Hegde (4NI19CS020)**
**Amrutha Satish (4NI19CS021)**
**Arpitha Prakash (4NI19CS026)**
**Deepitha M C (4NI19CS039)**

Under The Guidance Of

**M J Yogesh**
**Assistant Professor**
**Department of CS&E**

**Suhas S**
**Assistant Professor**
**Department of CS&E**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**THE NATIONAL INSTITUTE OF ENGINEERING**
**(An Autonomous College)**
**Mysore – 570 008**
**2022 - 2023**

# THE NATIONAL INSTITUTE OF ENGINEERING

## (An Autonomous Institution, affiliated to VTU)

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## Mysore – 570 008


# CERTIFICATE

This is to certify that the project work entitled "**Cloud Computing Tutorial"**, a work carried out by **Abhilash Shreedhar Hegde (4NI19CS020), Amruta Narayana Hegde (4NI19CS020), Amrutha Satish (4NI19CS021), Arpitha Prakash (4NI19CS026), Deepitha M C (4NI19CS039)** in partial fulfilment for the course tutorial work (Cloud Computing – CS7C01), Seventh semester, Computer Science & Engineering, The National Institute of Engineering (Autonomous Institution under Visvesvaraya Technological University, Belagavi) during the academic year 2022 – 2023. It is certified that all corrections and suggestions indicated for the Internal Assessment have been incorporated in the report.

| | |
|---|---|
| **Signature of the Guide** | **Signature of the Guide** |
| **(M J Yogesh)** | **(Suhas S)** |

# ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who helped us in completing the tutorial work successfully.

We express our profound thanks to **Dr. Rohini Nagapadma, Principal, NIE, Mysore** for all the support and encouragement.

We are grateful to **Dr. Yuvaraju B N, Professor and Head, Department of Computer Science and Engineering, NIE, Mysore** for support and encouragement facilitating the progress of this work.

We sincerely extend our thanks to our guides **M J Yogesh,** Assistant Professor, Department of Computer Science and Engineering, NIE, Mysore and **Suhas S,** Assistant Professor, Department of Computer Science and Engineering, NIE, Mysore for their valuable guidance and support for the tutorial work.

**-Abhilash Shreedhar Hegde**
**-Amruta Narayana Hegde**
**-Amrutha Satish**
**-Arpitha Prakash**
**-Deepitha M C**

***Q2. Compare CPU and GPU chips in terms of their strengths and weaknesses. Discuss the trade – offs between power efficiency, programmability and performance. Also compare various MPP architectures in processor selection, performance target, efficiency and packaging constraints.***

# 1. CPU and GPUs – A comparison:

## • The definition –

**CPU**: A central processing unit (CPU) also called the main processor or just processor, is the electronic circuitry that executes instructions comprising a computer program. It performs basic arithmetic, logic, controlling, and input/output operations specified by the instructions in the program. It is thus the component of a computer system that controls the interpretation and execution of instructions.

**GPU**: A graphics processing unit (GPU) is a specialized electronic circuit designed to manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device. It is thus a specialized processor that renders (or creates) images, animations, and graphics and then displays them on the computer's screen.

## • Benefits and Limitations : CPU –

**Benefits:**

- It can be programmed to execute a number of tasks
- Low power consumption
- It can quickly move data between various memory locations.
- Fast Calculation of Mathematical Data
- Because of their serial processing capabilities, the CPU can multitask across multiple activities in your computer. A strong CPU can provide more speed for typical computer use than a GPU.
- In specific situations, the CPU will outperform the GPU. For example, the CPU is significantly faster when handling several different types of system operations.
- CPUs are more readily available, more widely manufactured, and cost-effective for consumer and enterprise use.

**Limitations:**

- Physical overheating
- Only based on machine language

- The overall cost is huge.
- large size of PCB(printed circuit board) is required for assembling all components
- The processor has a limitation on the size of the data
- CPUs cannot handle parallel processing like a GPU, so large tasks that require thousands or millions of identical operations will choke a CPU's capacity to process data.
- Not every system or software is compatible with every processor. For example, applications written for x86 Intel Processors will not run on ARM processors.

## • Benefits and Limitations : GPU –

**Benefits:**

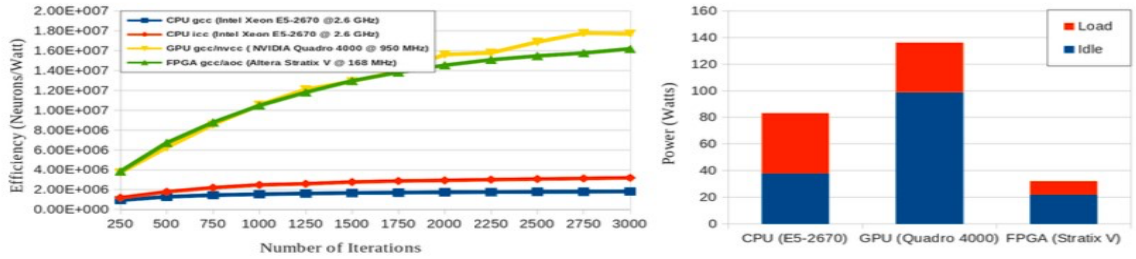
- **High Data Throughput**: a GPU consists of hundreds of cores performing the same operation on multiple data items in parallel. Because of that, a GPU can push vast volumes of processed data through a workload, speeding up specific tasks beyond what a CPU can handle.
- **Massive Parallel Computing**: Whereas CPUs excel in more complex computations, GPUs excel in extensive calculations with numerous similar operations, such as computing matrices or modelling complex systems.

**Limitations:**

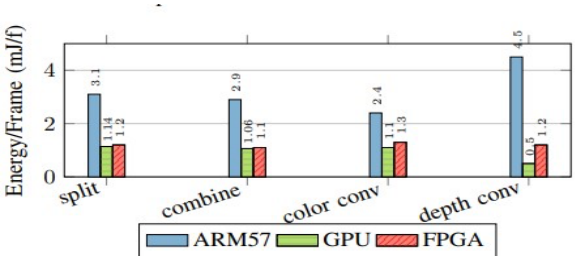
- **Cost**: While the price of GPUs has fallen somewhat over the years, they are still significantly more expensive than CPUs.
- **Power and Complexity**: While a GPU can handle large amounts of parallel computing and data throughput, they struggle when the processing requirements become more chaotic. Branching logic paths, sequential operations, and other approaches to computing impede the effectiveness of a GPU.

## • Power Efficiency: CPU vs GPU –

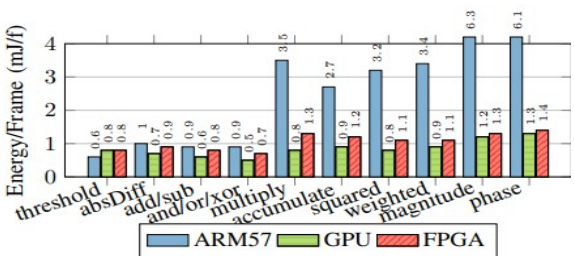
Power efficiency is the amount of computing performed for the amount of extra power consumed by the system, and power consumption is the total power consumed under load.




(fig 1 – Power efficiency and consumption comparison)




(fig 2 – Efficiency for I/O operation)

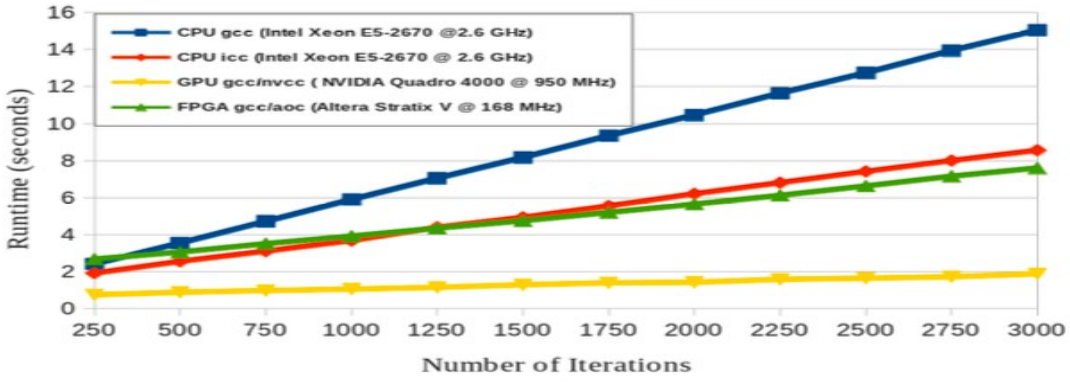



(fig 3 – Efficiency Arithmetic operations)

## • Performance Analysis: CPU vs GPU –

**Graphics Processing Units (GPUs)** in scientific computing have led the computing system to achieve tera-scale computing power in laptops and peta scale computing power to the clusters by combining multi-core **Central Processing Units (CPUs)** and many core GPUs which can be called heterogeneous computer architecture.

For performance comparison, the parameters considered are **latency** and **throughput**. So based on the execution of a GPU and CPU for a given task, the two parameters are measured by increasing the size of the workload. When the task size is increased, GPU is found to be approximately **51%** faster than the multithreaded CPU when GPU achieves 100% occupancy. The throughput of the GPU is found to be **2.1** times higher than that of the CPU for large task sizes.




(fig 1 - PCNN performance comparison for CPU and GPU)

- **Programmability: CPU vs GPU -**

GPUs and CPUs are used for different use cases. GPUs are used in Scientific computations, analytics, and engineering. Mainly GPUs are utilized in Games.

A GPU is essential for the fast, graphics-intensive rendering of the gaming world. Rendering special effects and sophisticated 3D graphics in real time requires some serious computing power. The tasks of modern games become too heavy for CPU graphics solutions. Another major field where GPUs are used is in Machine Learning. GPU architecture is not a substitute for CPU architecture. Instead, they are effective infrastructure accelerators. While the remaining code continues to operate on the CPU, GPU-accelerated computing offloads the application's computationally heavy portions to the GPU.

## 2. **Massively Parallel Processing: A Study:**

- **Massively Parallel Processing: The Definition and Types -**

Massively parallel is the term for using a large number of computer processors to simultaneously perform a set of coordinated computations in parallel. It is a storage structure designed to handle the coordinated processing of program operations by multiple processors. This coordinated processing can work on different parts of a program, with each processor using its own operating system and memory.

There are two types of MPP database architecture: grid computing and computer clustering.

- **Efficiency: Grid and Cluster –**

**Grid:**

Grid computing attempts to achieve high computational performance by non-traditional means. It is about leveraging your available resources and idle processor cycles to solve a problem while at the same time maximizing efficiency and reducing your total cost of ownership. Grid computing allows one to share computer resources across networks. This can both increase the computational power available to programs and reduce the number of machines needed by the organization. In grid computing, every node has its resource director that carries on analogous to a free element

**Cluster:**

Cluster computing resolves the need for content critically and process services in a quicker approach It is cost-effective and provides enhanced performance than mainframe computers. Cluster computers have high and enhanced availability where-in if one node fails, the other nodes will be active and will function as a proxy for the failed node. Cluster computing also provides enhanced scalability and expandability. It can be upgraded to superior specifications or extended through the addition of additional nodes. In cluster computing, the resources are overseen by the centralized resource director.

## • Performance Target: Grid and Cluster –

**Grid:**

Data grid computing allows several users to simultaneously access, change, or transfer distributed data. For instance, a data grid can be used as a large data store where each website stores its own data on the grid.

**Cluster:**

Cluster computing offers solutions to solve complicated problems by providing faster computational speed, and enhanced data integrity. It provides mechanisms that enable critical resources to be automatically available on backup systems. Those resources could include data, application programs, devices, or environmental attributes.

## • Packaging Constraints: Grid and Cluster -

**Grid:**

In grid computing, the nodes are packaged in a slack manner. The odes of a slack cluster are normally connected through standard LANs or WANs.

**Cluster:**

Cluster nodes can be packaged in a compact or a slack fashion. In a compact cluster, the nodes are closely packaged in one or more racks sitting in a room, and the nodes are not attached to peripherals. A compact cluster can utilize a high-bandwidth, low-latency communication network that is often proprietary In a slack cluster, the nodes are attached to their usual peripherals and they may be located in remote regions.

***Q3. Use of Mosix for cluster computing. Check with open literature on current features that have been claimed by designers and developers in supporting Linux clusters, GPU clusters, Multi clusters and even virtualized clouds. Discuss the advantages and shortcomings from the users' perspective.***

## Introduction

MOSIX is a Linux distribution that is designed specifically for cluster computing. It is based on the CentOS operating system and includes a range of tools and features that are optimized for use in a cluster environment. MOSIX (multi computer OS for Linux/Unix) is a cluster operating system which provides users and applications with the impression of running on a single computer with multiple processors. MOSIX is a proprietary distributed operating system. It was developed by Hebrew in the year 1977 and was redesigned in 1999 to run Linux cluster built in x86 platforms. MOSIX supports both interactive processes and batch jobs. It incorporates dynamic resource discovery and automatic workload distribution. In MOSIX, each process has a unique home node (UHN) where the process was created. This node is normally a logic node for application users.

## Two Main Building Blocks Of MOSIX

### Configuration

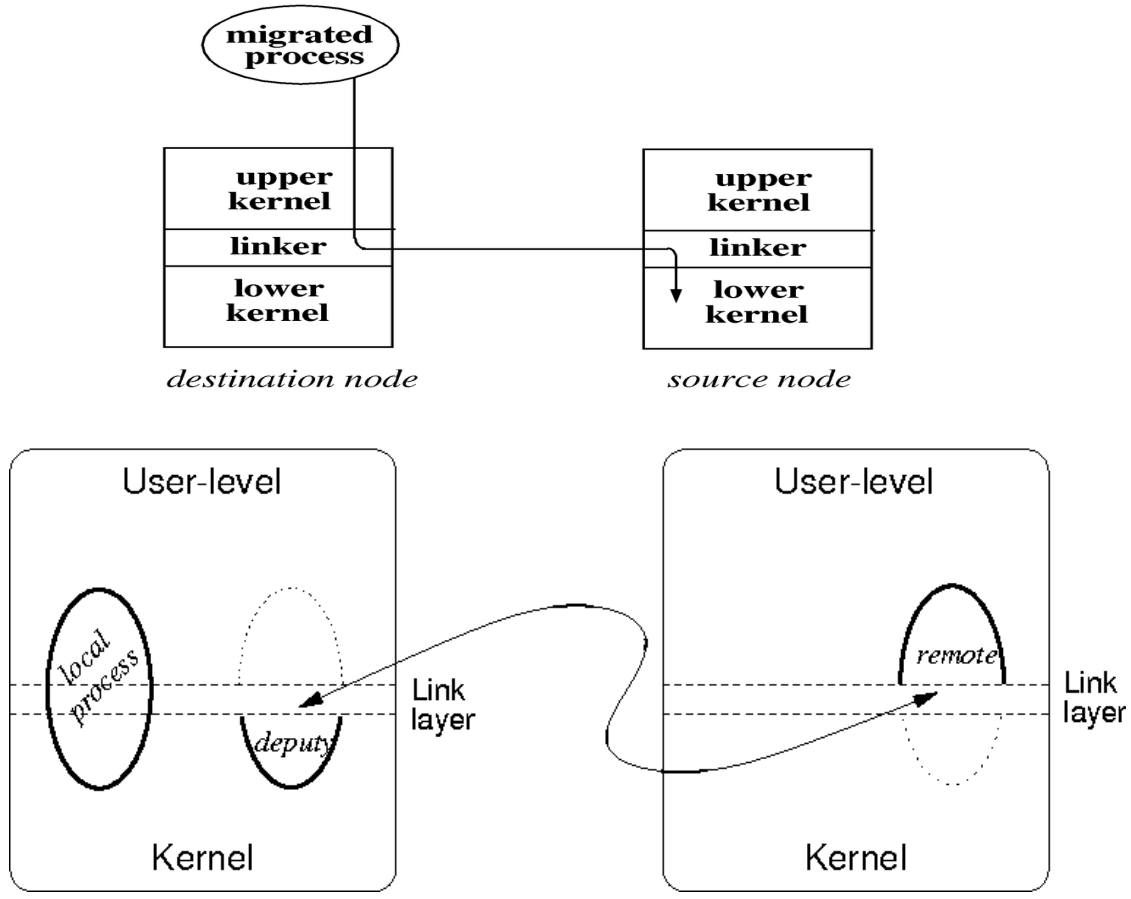
A MOSIX cluster is a set of connected servers and workstations (nodes), that are administrated by a single owner and run the same version of MOSIX. In a MOSIX cluster, each node maintains information about availability and the state of resources in the other nodes. Whereas MOSIX multi-cluster private cloud is a collection of MOSIX clusters that run the same version of MOSIX and are configured to work together and in a MOSIX cloud, each node maintains information about the availability and state of resources of the other nodes in all the connected clusters. In MOSIX clouds there is usually a high degree of trust, i.e., a guarantee that applications are not viewed or tampered with when running in remote clusters.

### Process

MOSIX processes are usually user applications that are suitable and can benefit from migration. MOSIX processes are created by the "*mosrun*" command. MOSIX processes are started from standard Linux executables, but run in an environment that allows each process to migrate from one node to

another. The node in which a process was created is called its home-node. Child processes of MOSIX processes remain under the MOSIX discipline (with the exception of the native utility, that allows programs, mainly shells, already running under *mosrun*, to spawn children in native Linux mode).

**MOSIX Architecture**




(Process Migration)

1. **Some potential advantages of using Mosix for cluster computing include:**

**Ease of use** - Mosix is designed to be easy to install and use, even for users who are not familiar with cluster computing. It includes a range of tools and utilities that can help users set up and manage their cluster.

**Scalability** - Mosix is designed to be scalable, that it can support a wide range of cluster sizes and configurations. This can be useful for users who need to expand their cluster as their needs grow.

**Performance** - Mosix is designed to be fast and efficient, with a lightweight kernel and other optimizations that can help to improve the performance of the cluster.

**Security** - Mosix includes a range of security features, such as firewalls and antivirus software, to help protect the cluster from threats.

2. **Some potential shortcomings of using Mosix for cluster computing include:**

**Limited support** - Mosix is a relatively new Linux distribution, and as such it may not have the same level of support and resources as more established distributions.

**Compatibility issues** - Mosix is based on CentOS, but it includes a number of modifications and customizations that may not be compatible with all software and applications. Users may need to test and verify that their desired software will work with Mosix before using it in their cluster.

Overall, Mosix can be a useful option for users who are looking for a Linux distribution that is optimized for use in a cluster environment. However, it is important for users to carefully consider the potential advantages and drawbacks of using Mosix before implementing it in their cluster.

Many Linux clusters and GPU clusters are designed to support HPC workloads, which require high levels of processing power and fast data transfer rates. These clusters can be used to run simulations, perform data analysis, and run other resource-intensive tasks that would be difficult or impossible to perform on a single machine.

**Scalability** - Linux clusters and GPU clusters are designed to be scalable, meaning that they can be easily expanded as the needs of the user grow. This can be especially useful for users who need to process large amounts of data or run complex simulations.

**Load balancing** - Linux clusters and GPU clusters are designed to distribute workloads across multiple nodes in the cluster, which can help to balance the load and improve performance. This can be especially useful for users who need to run multiple tasks concurrently.

**Fault tolerance** - Many Linux clusters and GPU clusters are designed to be fault tolerant, meaning that they can continue to operate even if one or more nodes in the cluster fail. This can be especially important for users who rely on the cluster for critical tasks.

3. **Some common features of multi clusters and virtualized clouds include:**

**Flexibility** - Multi clusters and virtualized clouds are designed to be flexible, meaning that they can be easily reconfigured to meet the changing needs of the user. This can be especially useful for users who need to run a variety of different types of workloads.

**Resource allocation** - Multi clusters and virtualized clouds are designed to allow users to allocate resources as needed, which can help to optimize the performance of the cluster. This can be especially useful for users who need to run multiple tasks concurrently.

**Cost efficiency** - Multi clusters and virtualized clouds can be more cost efficient than traditional on-

premises solutions, as they allow users to pay for only the resources that they need. This can be especially useful for users who have fluctuating resource requirements.

4. **Some potential shortcomings of Linux clusters, GPU clusters, multi clusters, and virtualized clouds from the user's perspective include:**

**Complexity** - Setting up and maintaining a Linux cluster, GPU cluster, multi cluster, or virtualized cloud can be complex, especially for users who are not familiar with these technologies. This can require a significant investment of time and resources.

**Dependency on network connectivity** - Linux clusters, GPU clusters, multi clusters, and virtualized clouds rely on network connectivity to function properly. If the network goes down or experiences other issues, it can disrupt the operation of the cluster.

**Security concerns** - Linux clusters, GPU clusters, multi clusters, and virtualized clouds can be vulnerable to security threats, such as cyber-attacks. Users may need to implement additional security measures to protect their data and systems.

Overall, the features and capabilities of Linux clusters, GPU clusters, multi clusters, and virtualized clouds can provide a range of advantages to users, depending on their specific needs and requirements. However, it is important for users to carefully consider the potential shortcomings of these technologies before implementing them.

***Q4. Install XEN on an Ubuntu Machine in two methods from the binary code or from the source code. Compile installation guides for the two methods used. Describe the dependencies of utilities and packages along with troubleshooting tips.***

Xen is a type-1 hypervisor, also known as a baremetal hypervisor, that allows multiple operating systems to run concurrently on the same physical machine. It runs directly on top of the hardware and provides virtualization services to the guest operating systems, which are known as domains (DomU) in Xen terminology. The default kernel provided in Ubuntu can be used as the management domain (Dom0), which has access to the physical resources of the machine and can create, manage, and terminate other domains. Xen is open-source software that was developed by the University of Cambridge and is now managed by the Linux Foundation.

**Installing XEN using Binary Code:**

To install XEN from the binary code on an Ubuntu machine, follow these steps:

- Download the XEN binary package from Ubuntu from the XEN website or using the following the command: **Wget** **https://downloads.xenproject.org/release/xen/4.14.0/xen-4.14.0.tar.gz**
- Extract the downloaded tar archive using the following command: **tar xvf xen-4.14.0.tar.gz**
- Change to the directory containing the extracted files: **cd xen-4.14.0**
- Install the required dependencies by running the following command: **sudo apt-get update**

  **sudo apt-get install build-essential python-dev libncursed5-dev uuid-dev python-twisted libyajl-dev libc6-dev-i386 python-pkg-resources**

- Configure XEN by running the following command: **./configure**
- Build XEN by running the following command: **make**
- Install XEN by running the following command: **sudo make install**

**To install XEN from the source code on an Ubuntu machine, follow these steps:**

- Download the XEN source code from the XEN website or using the following command:

**git clone https://github.com/xen-project/xen.git**

- Change to the directory containing the downloaded source code: **cd xen**
- Install the required dependencies by running the following command: **sudo apt-get update**

  **sudo apt-get install build-essential python-dev libncurses5-dev uuid-dev python-twisted libyajl-dev libc6-dev-i386 python-pkg-resources**

- Configure XEN by running the following command: **./configure**
- Build XEN by running the following command: **make**
- Install XEN by running the following command: **sudo make install**

**Dependencies:**

- **build-essential** – A package containing essential tools for building Debian packages.
- **python-dev** – A package containing header files, a static library, and development tools for building Python modules.
- **libncursed5-dev** – A package containing header files and static libraries for building applications that use the ncurses library.
- **uuid-dev** – A package containing header files and static libraries for building applications that use the libuuid library.
- **python-twisted** – A package containing a networking engine for Python.
- **libyajl-dev** – A package containing header files and static libraries for building applications.

**Troubleshooting Tips:**

- If you encounter any errors during the configuration or build process, try installing the missing dependencies or consult the XEN documentation for guidance.
- If you encounter any issues during the installation process, try running the command with the '**-f**' flag to force the installation.
- If you encounter any issues while running XEN, try checking the XEN log files located in '**/var/log/xen**' for errors.

***Q5. Install any five VM's of your choice and network them together. You should be able to access internet on any of the VM's. Document the procedure with appropriate screenshots.***

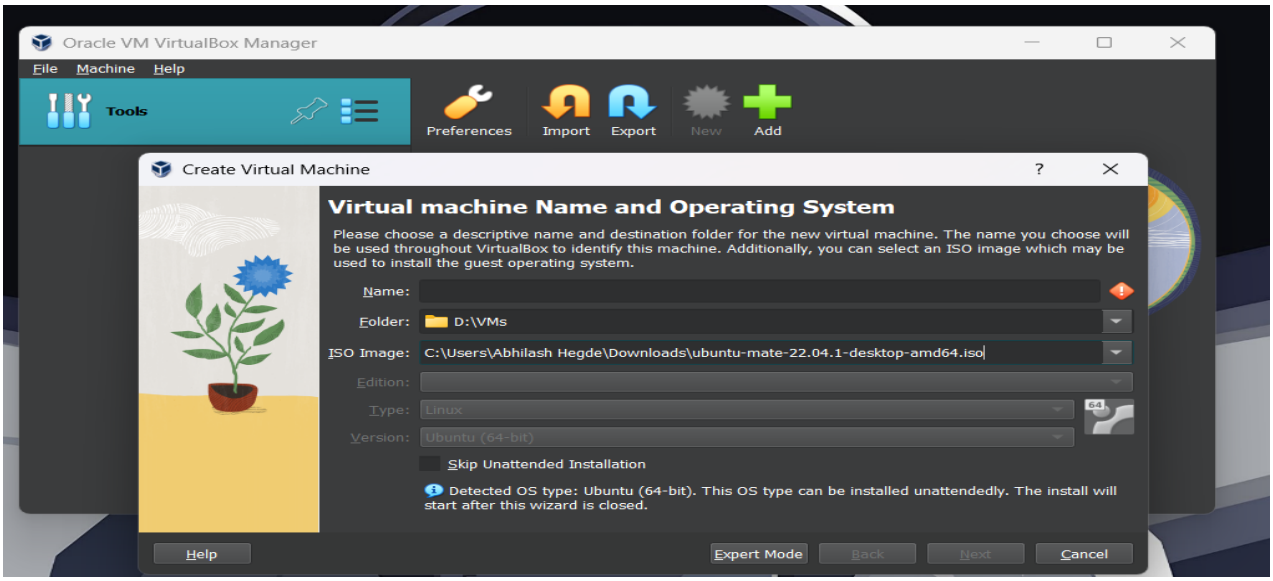
**Step 1:** Download VirtualBox

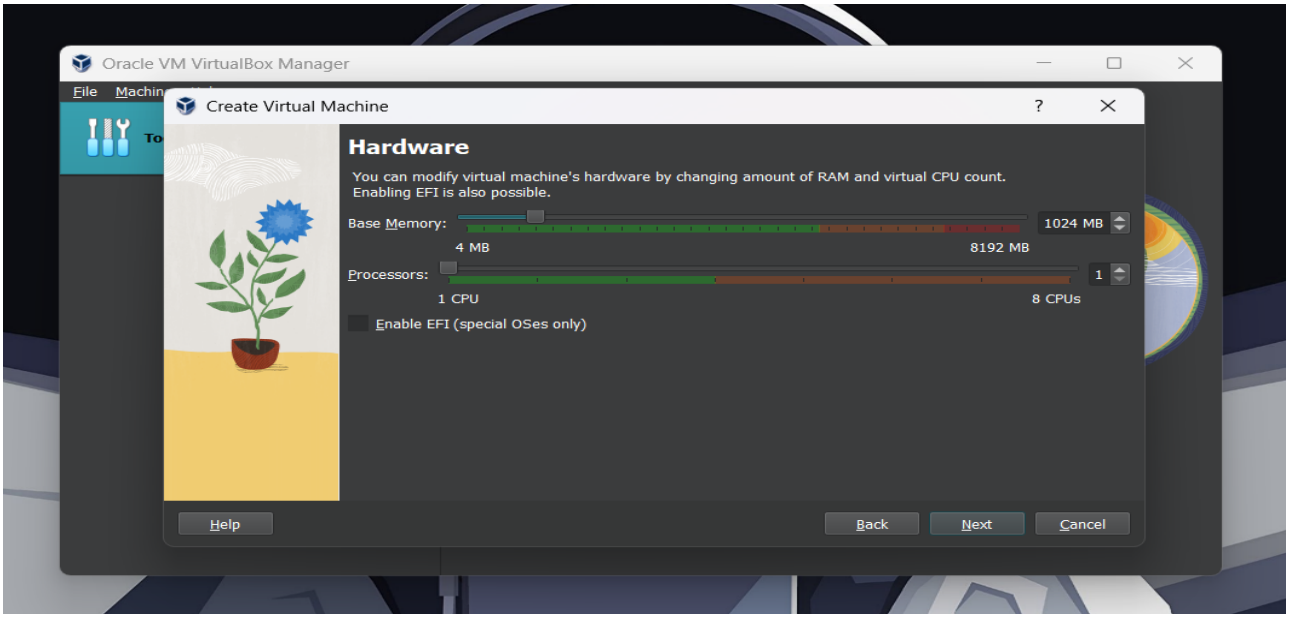
**Step 2:** Click on new and create the virtual machine.

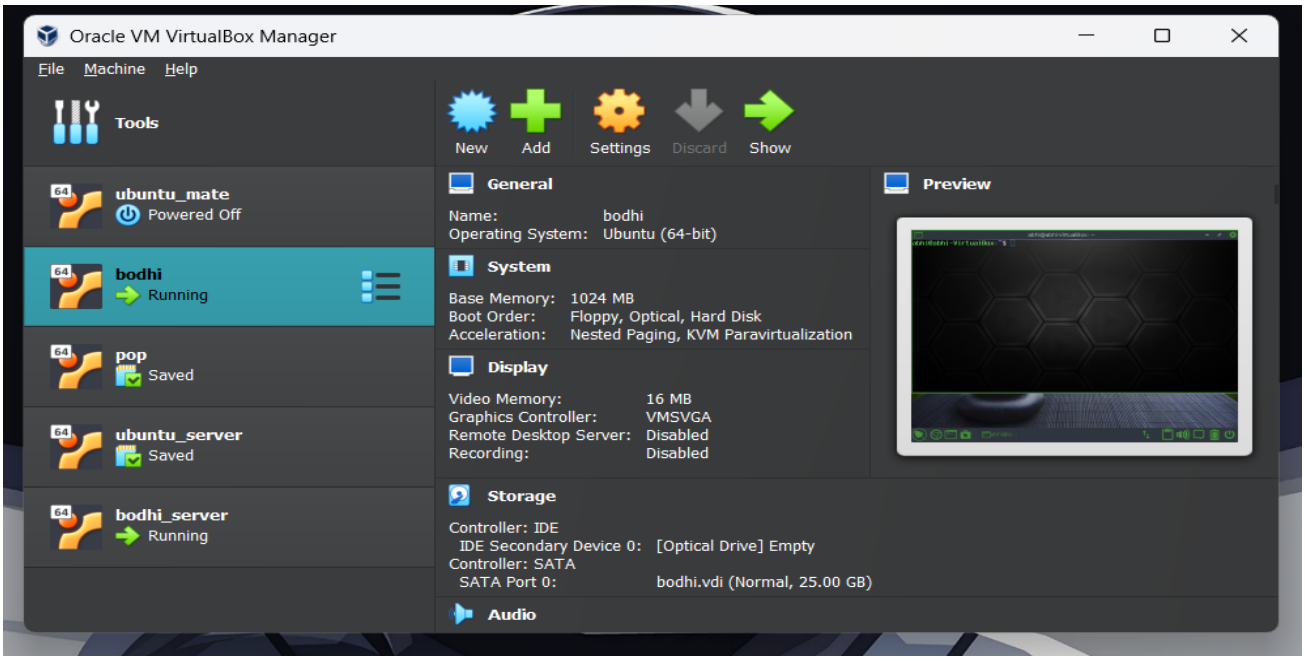
**Step 3:** Type name of the machine

**Step 4:** Select ISO file and configure RAM and storage requirements

**Step 5:** Follow same procedure to install 5 virtual machines

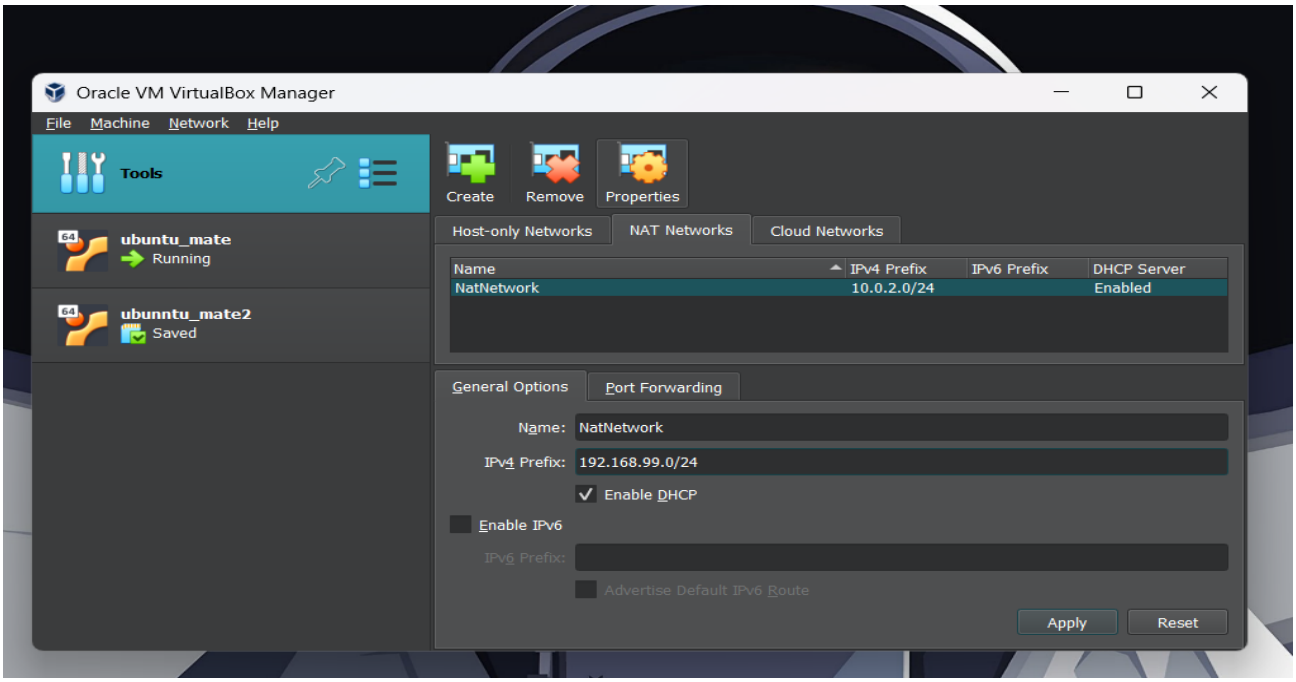




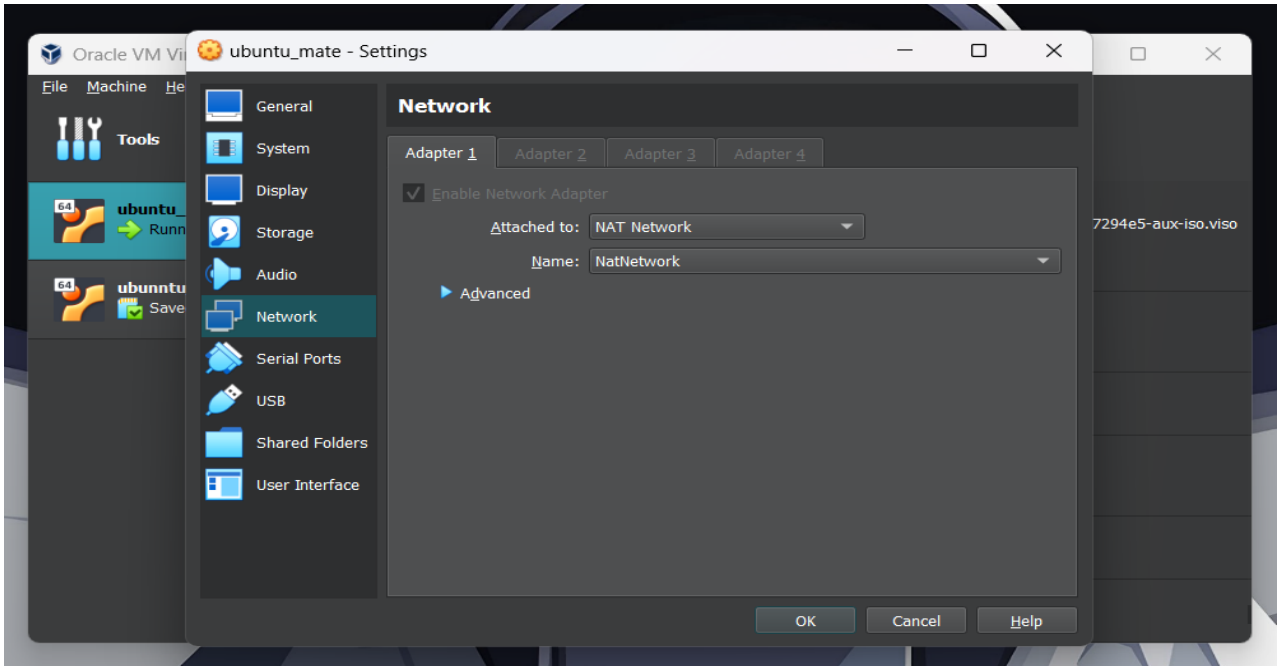

## Step 6 – Add Nat Network

- Tap on tools option
- Select network manager
- Click on Nat Networks
- Then add new network as NatNetwork with IP = 192.168.99.0/24
- Click on check box to enable DHCP



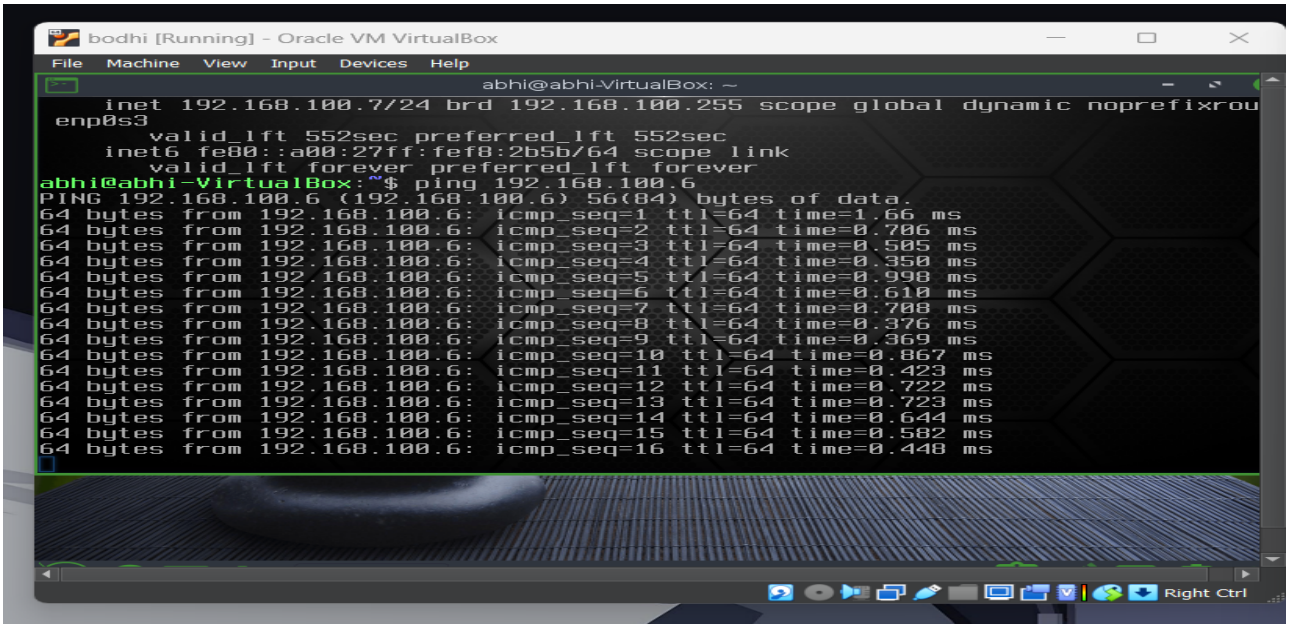
**Step -7 Configure each virtual machine**

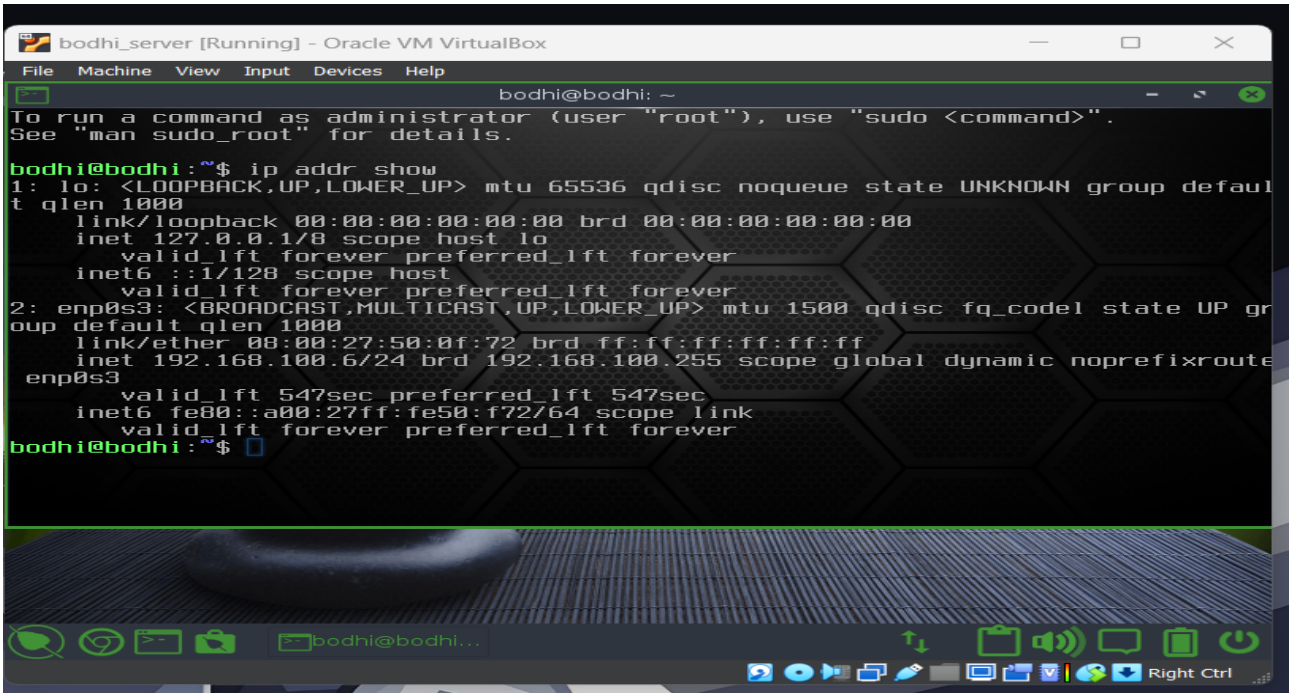
- Select each VM in virtual box go to settings->network.
- Chose adapter and attach NAT network & name.
- Click on apply




**Step - 8**

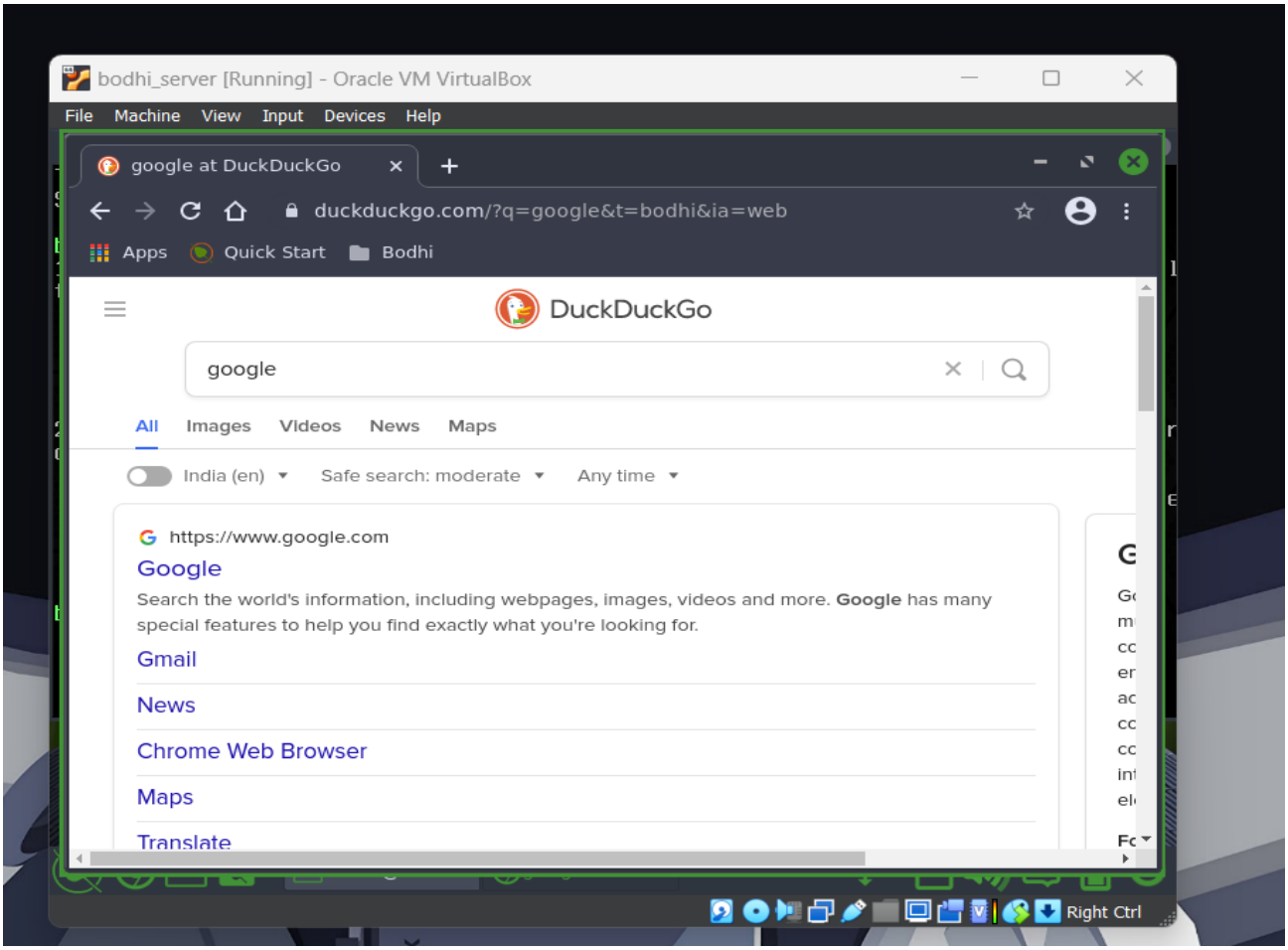
Launch each VM and verify the DHCP-assigned IP address in the console. Use the ping command to ping the other system; if the connection is successful, we will be able to do so.






**Step - 9**

Checking internet connection.



***Q6. Write any application (web or HPC) of your choice which supports Amazon Web Services (AWS – PaaS). Document the code with supporting screenshots.***

The application we created is a movie guesser. This involves guessing movie names based on the brief plot of the movie. It involves guessing letter as well. The application consists of the following files :

**main.dart:**

```dart
import 'package:flutter/material.dart';
import 'package:animated_splash_screen/animated_splash_screen.dart';
import 'package:guessthemovie/screens/home.dart';

void main() {
  runApp(
    MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        fontFamily: 'Questrial',
      ),
      routes: {
        'home': (context) => Home(),
      },
      home: AnimatedSplashScreen(
        splash: Text(
          'MOVIE GUESSER',
          style: TextStyle(fontSize: 40, fontFamily: 'Questrial'),
        ),
        animationDuration: Duration(milliseconds: 600),
        splashTransition: SplashTransition.fadeTransition,
        nextScreen: Home(),
      ),
    ),
  );
}
```

**app.js:**

```
const express = require("express");
const app = express();
const path = require("path");

app.use(express.json());
app.use(express.static(path.join(__dirname, "/web/build/web/")));

app.get("*", (_, res) => {
  res.sendFile(path.resolve(__dirname, "/web/build/web/index.html"));
});

const port = process.env.PORT || 3000;

app.listen(port, () => {
  console.log(`listening at ${port}`);
});
```

**home.dart:**

```
import 'package:flutter/material.dart';

import 'movie_page.dart';

class Home extends StatefulWidget {
  @override
  _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> {
  List<String> types = [
    "Top 250 movies",
    "Top 250 English movies",
    "Top 250 Indian movies",
  ];
  String url = '';
  @override
  Widget build(BuildContext context) {
```

```
return Scaffold(
  appBar: AppBar(
    title: Text(
      'Movie Guesser',
      style: TextStyle(
        fontFamily: 'Questrial',
        color: Colors.black,
      ),
    ),
    backgroundColor: Colors.yellow,
  ),
   body: Stack(
    children: [
      Container(
        width: MediaQuery.of(context).size.width,
        child: Image(
          fit: BoxFit.fill,
          image: AssetImage('asset/images/moviesPoster.jpg'),
        ),
      ),
      Center(
        child: Padding(
          padding: const EdgeInsets.only(top: 250),
          child: Container(
            width: 400,
            child: ListView.builder(
              itemCount: 3,
              itemBuilder: (context, item) {
                return Card(
                  elevation: 5,
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(20),
                  ),
                  clipBehavior: Clip.antiAlias,
                  child: Container(
                    height: 50,
                    width: 180,
                    child: InkWell(
                      child: Center(
                        child: Text(
                          types[item],
                          style: TextStyle(fontFamily: 'Questrial'),
                        ),
                      ),
                      onTap: () {
                        switch (item) {
                          case 0:
                            Navigator.pushReplacement(
                              context,
                              MaterialPageRoute(
                                builder: (context) {
                                  return MoviePage();
                                },
                              ),
                            );
```

```
                                    break;
                                  case 1:
                                    Navigator.pushReplacement(
                                      context,
                                      MaterialPageRoute(
                                        builder: (context) {
                                          return MoviePage();
                                        },
                                      ),
                                    );
                                    break;
                                  case 2:
                                    Navigator.pushReplacement(
                                      context,
                                      MaterialPageRoute(
                                        builder: (context) {
                                          return MoviePage();
                                        },
                                      ),
                                    );
                                    break;
                                  default:
                                    break;
                                }
                              },
                            ),
                          ),
                        );
                      },
                    ),
                  ),
                ),
              ),
            ],
          ),
        );
    }
}
```

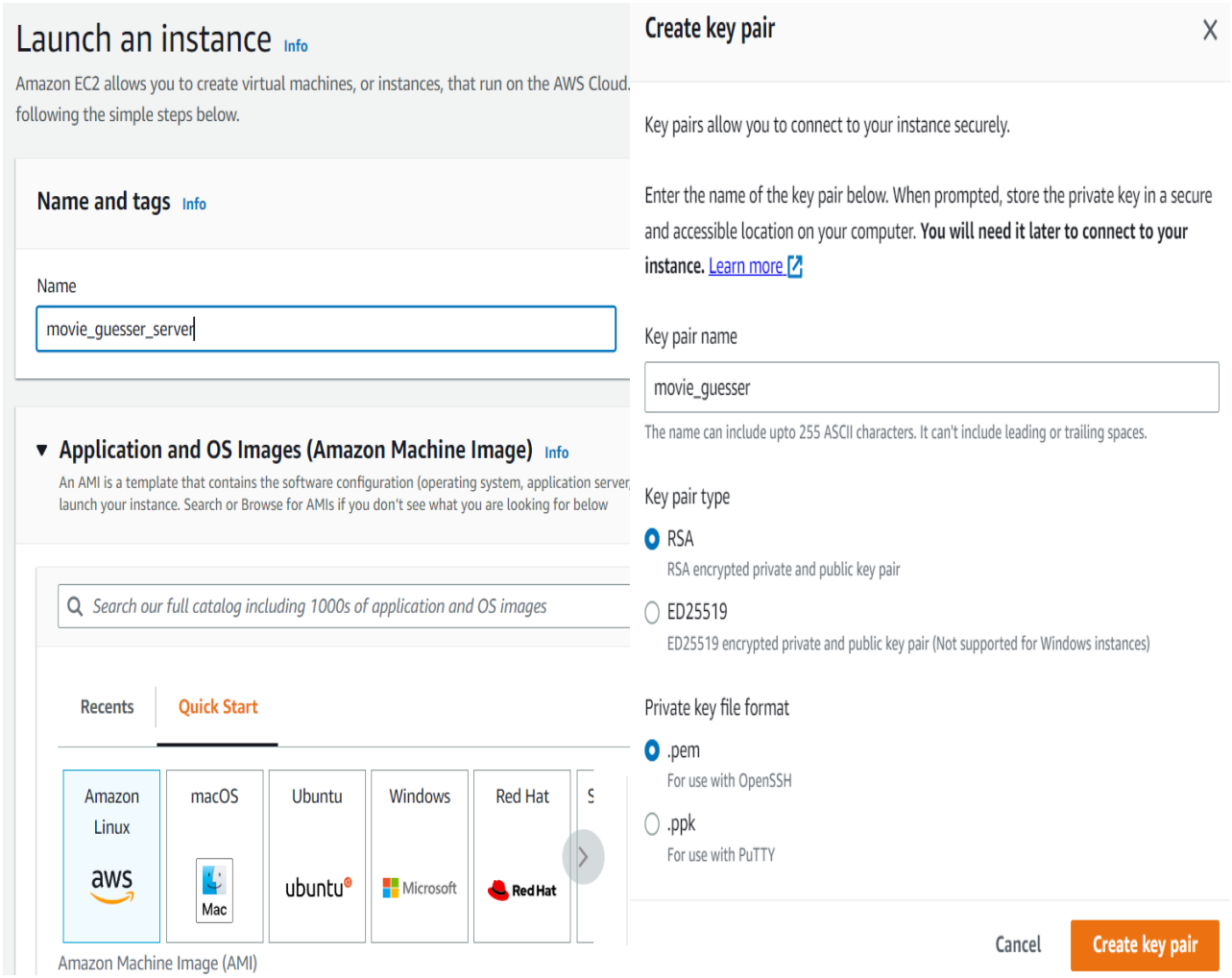
## movie_page.dart:

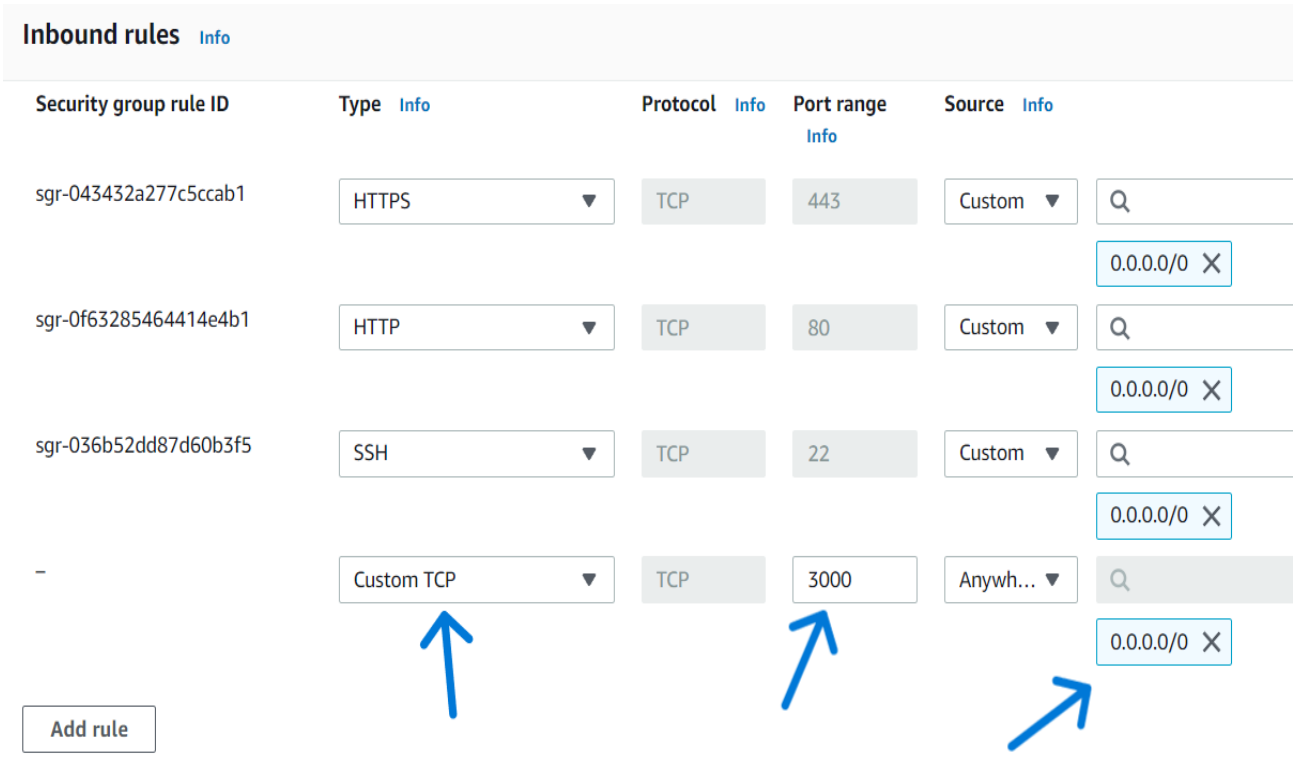
```dart
import 'dart:math';
import 'package:flutter/material.dart';
import 'dart:convert' as convert;
import 'package:http/http.dart' as http;
import '../component/title_page.dart';

class MoviePage extends StatefulWidget {
  @override
  _MoviePageState createState() => _MoviePageState();
}

class _MoviePageState extends State<MoviePage> {
  late String movieTitle;
  late String moviePlot;
  int randomIndex = 0;
  Future fetchTitle() async {
    var url = Uri.parse('https://api.npoint.io/c8bb8492ccb141c867d4');
    var response = await http.get(url);
    var title = convert.jsonDecode(response.body) as Map<dynamic, dynamic>;
    randomIndex = rnd();
    movieTitle = title["Movies"][randomIndex]["Title"].toString().toLowerCase();
    moviePlot = title["Movies"][randomIndex]["Plot"].toString();
  }
```

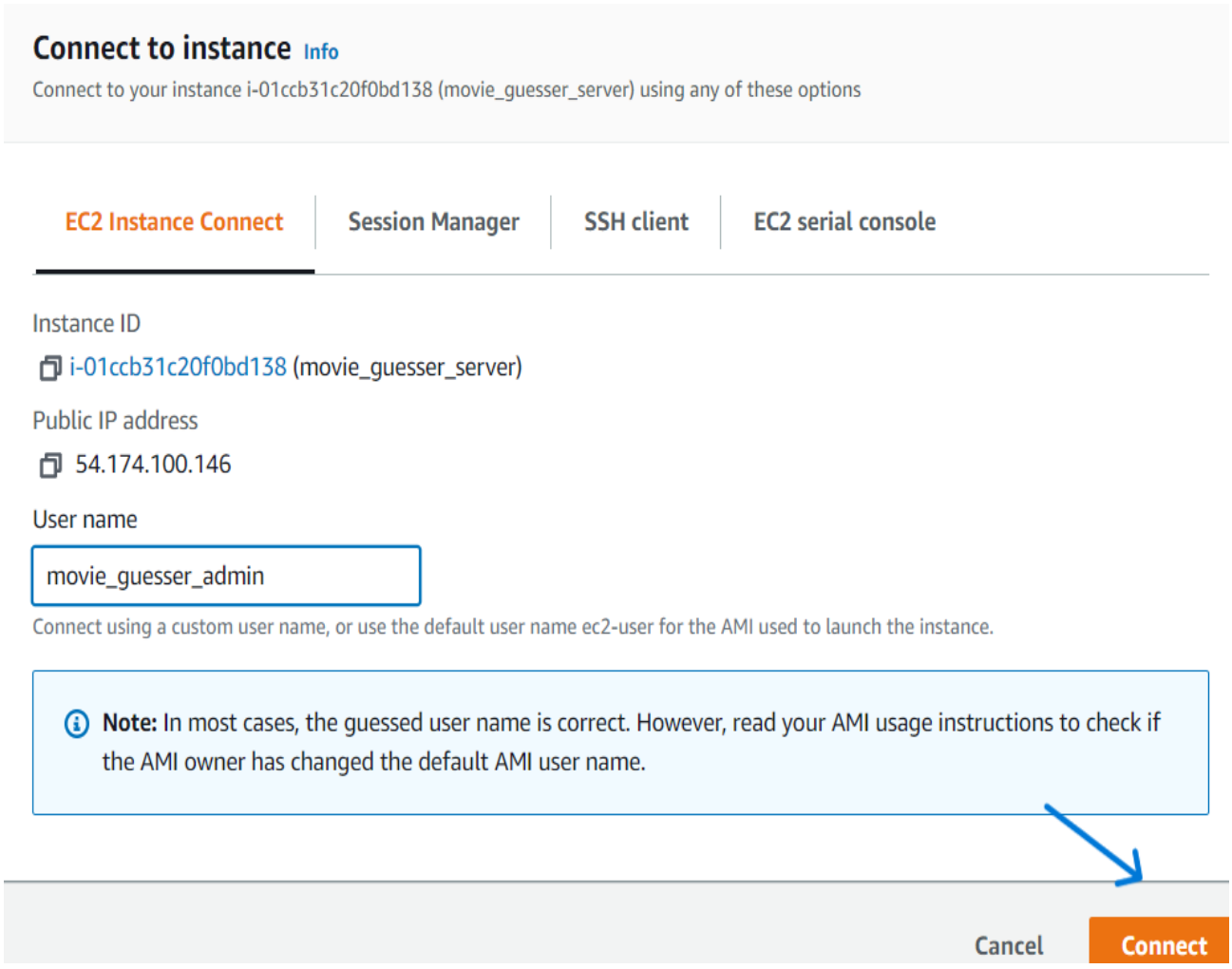
```
  int rnd() {
    var r = Random();
    return 0 + r.nextInt(250 - 0);
  }

  @override
  Widget build(BuildContext context) {
    return FutureBuilder(
      future: fetchTitle(),
      builder: (context, snap) {
        if (snap.connectionState == ConnectionState.done) {
          return TitlePage(
            moviePlot: moviePlot,
            movieTitle: movieTitle,
          );
        } else
          return Scaffold(
            body: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              crossAxisAlignment: CrossAxisAlignment.center,
              children: [
                Center(
                  child: CircularProgressIndicator(
                    color: Colors.yellow[600],
                  ),
                ),
                SizedBox(
                  height: 30.0,
                ),
                Center(
                  child: randomIndex % 2 == 0
                      ? Text(
                          'This question is tricky...',
                          style: TextStyle(
                            fontSize: 15.0,
                            color: Colors.black,
                          ),
                        )
                      : Text(
                          'Finding you an easy question...',
                          style: TextStyle(
                            fontSize: 15.0,
                            color: Colors.black,
                          ),
                        ),
                ),
              ],
            ),
          );
      },
    );
  }
}
```

## I. Steps to deploy create and launch aws ec2 instance –

- Login to the AWS account.
- Navigate to EC2 in services and click on launch instance.
- Provide a name for instance (for example movie_guesser_server).
- Chooser required OS and configs.
- Create a key pair name and download the corresponding .pem file.
- Go to security settings and select security group.
- Create custom TCP and then set port number ( we have used port 3000)
- Launch the instance




Inbound rules Info
Security group rule ID
Type Info
Protocol Info
Port range Info
Source Info
sgr-043432a277c5ccab1
HTTPS
TCP
443
Custom
0.0.0.0/0
sgr-0f63285464414e4b1
HTTP
TCP
80
Custom
0.0.0.0/0
sgr-036b52dd87d60b3f5
SSH
TCP
22
Custom
0.0.0.0/0
–
Custom TCP
TCP
3000
Anywh...
0.0.0.0/0
Add rule


Connect to instance Info
Connect to your instance i-01ccb31c20f0bd138 (movie_guesser_server) using any of these options
EC2 Instance Connect
Session Manager
SSH client
EC2 serial console
Instance ID
i-01ccb31c20f0bd138 (movie_guesser_server)
Public IP address
54.174.100.146
User name
movie_guesser_admin
Connect using a custom user name, or use the default user name ec2-user for the AMI used to launch the instance.
Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.
Cancel
Connect

## II. Steps to install node js on AWS instance –

- Install node version manage

  curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh

  | bash

- Activate nvm by typing the following at the command line.

  . ~/.nvm/nvm.sh

- Install node js

  rm -rf /etc/yum.repos.d/nodesource-el*

  curl -sL https://rpm.nodesource.com/setup_16.x

  | sudo -E bash -

  sudo yum install nodejs --enablerepo=nodesource

```
[ec2-user@ip-172-31-93-164 ~]$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 13226  100 13226    0     0   504k      0 --:--:-- --:--:-- --:--:--  516k
=> Downloading nvm as script to '/home/ec2-user/.nvm'

=> Appending nvm source string to /home/ec2-user/.bashrc
=> Appending bash_completion source string to /home/ec2-user/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This loads nvm bash_completion
[ec2-user@ip-172-31-93-164 ~]$
```

i-01ccb31c20f0bd138 (movie_guesser_server)

PublicIPs: 54.174.100.146 PrivateIPs: 172.31.93.164

## III. Steps to pull and run node js server repository from github –

- git clone <repo url>

  In our case, git clone https://github.com/cloud-comp-tutorial/movieguesser.git

- cd movieguesser - npm install node_module

- To run the server - node app.js