Name:

Division:

Roll No.:

Practical Number: 1

Title: Implement depth-first search algorithm and Breadth First Search algorithm, Use an undirected graph, and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.

DFS:

Code:-

```
graph = {
  '5' : ['3','7'],
  '3' : ['2', '4'],
  '7' : ['8'],
  '2' : [],
  '4' : ['8'],
  '8' : []
}

visited = set()

def dfs(visited, graph, node):
    if node not in visited:
        print (node)
        visited.add(node)
        for neighbour in graph[node]:
            dfs(visited, graph, neighbour)

print("Following is the Depth-First Search:")
dfs(visited, graph, '5')
```

Output:-

```
Following is the Depth-First Search:
5
3
2
4
8
7
```

BFS:

Code:-

```
graph = {
  'A' : ['B','C','D'],
  'B' : ['E'],
  'C' : ['D','F'],
  'D' : [],
  'E' : [],
  'F' : []
}
visited = []
queue = []

def bfs(visited, graph, node):
  visited.append(node)
  queue.append(node)
  while queue:
    s = queue.pop(0)
    print (s, end = " ")

    for neighbour in graph[s]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)

print("Following is the Path using Breadth-First Search:")
bfs(visited, graph, 'A')
```

Output:-

```
Following is the Path using Breadth-First Search:
A B C D E F
```

Practical Number 2:

Title: Implement A star Algorithm for any game search problem.

Code:-

```
from collections import deque

class Graph:
    def __init__(self, adjacency_list):
        self.adjacency_list = adjacency_list

    def get_neighbors(self, v):
        return self.adjacency_list[v]

    def h(self, n):
        H = {
            'A': 1,
            'B': 1,
            'C': 1,
            'D': 1
        }

        return H[n]

    def a_star_algorithm(self, start_node, stop_node):
        open_list = set([start_node])
        closed_list = set([])
        g = {}
        g[start_node] = 0
        parents = {}
        parents[start_node] = start_node

        while len(open_list) > 0:
            n = None

            for v in open_list:
                if n == None or g[v] + self.h(v) < g[n] + self.h(n):
                    n = v;

            if n == None:
                print('Path does not exist!')
                return None

            if n == stop_node:
                reconst_path = []

                while parents[n] != n:
                    reconst_path.append(n)
                    n = parents[n]

                reconst_path.append(start_node)

                reconst_path.reverse()

                print('Path found: {}'.format(reconst_path))
                return reconst_path

            for (m, weight) in self.get_neighbors(n):
                if m not in open_list and m not in closed_list:
                    open_list.add(m)
                    parents[m] = n
                    g[m] = g[n] + weight

                else:
                    if g[m] > g[n] + weight:
                        g[m] = g[n] + weight
                        parents[m] = n

                        if m in closed_list:
                            closed_list.remove(m)
                            open_list.add(m)

            open_list.remove(n)
            closed_list.add(n)

        print('Path does not exist!')
        return None
adjacency_list = {
    'A': [('B', 1), ('C', 3), ('D', 7)],
    'B': [('D', 5)],
    'C': [('D', 12)]
}
graph1 = Graph(adjacency_list)
graph1.a_star_algorithm('A', 'D')
```

Output:-

```
Path found: ['A', 'B', 'D']

['A', 'B', 'D']
```

Practical Number: 3

Title: Implement Greedy search algorithm for Selection Sort

Code:-

```
x = []
n = int(input("Enter how many elements you want for sorting"))
for i in range(n):
    print("Enter list elements")
    a = int(input(""))
    x.append(a)
print("unsorted Elements list is: ", x)

for i in range(0,len(x)-1):
    for j in range(i+1,len(x)):
        if x[i]>x[j]:
            c=x[i]
            x[i]=x[j]
            x[j]=c
print("Sorted Elements List is: ",x)
```

Output:-

```
Enter how many elements you want for sorting 5
Enter list elements
5
Enter list elements
3
Enter list elements
1
Enter list elements
4
Enter list elements
2
unsorted Elements list is:  [5, 3, 1, 4, 2]
Sorted Elements List is:  [1, 2, 3, 4, 5]
```

Practical Number: 4

Title: Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for an n-queens problem.

Code:-

```
n = int(input("Enter the value for n:\n"))
board = [[0 for i in range(n)] for i in range(n)]


def check_column(board, row, column):
    for i in range(row, -1, -1):
        if board[i][column] == 1:
            return False
    return True


def check_diagonal(board, row, column):
    for i, j, in zip(range(row, -1, -1), range(column, -1, -1)):
        if board[i][j] == 1:
            return False
    for i, j, in zip(range(row, -1, -1), range(column, n)):
        if board[i][j] == 1:
            return False
    return True


def nqn(board, row):
    if row == n:
        return True
    for i in range(n):
        if (check_column(board, row, i) == True and check_diagonal(board, row, i) == True):
            board[row][i] = 1
            if nqn(board, row + 1):
                return True
            board[row][i] = 0
    return False


nqn(board, 0)
for row in board:
    print(row)
```

Output:-

```
Enter the value for n
4
[0, 1, 0, 0]
[0, 0, 0, 1]
[1, 0, 0, 0]
[0, 0, 1, 0]
```

Practical Number: 5

Title: Develop an elementary chatbot for any suitable customer interaction application.

Code:-

```
def greet(bot_name, birth_year):
    print("Hello! My name is {0}.".format(bot_name))
    print("I was created in {0}.".format(birth_year))


def remind_name():
    print('Please, remind me your name.')
    name = input()
    print("What a great name you have, {0}!".format(name))


def guess_age():
    print('Let me guess your age.')
    print('Enter remainders of dividing your age by 3, 5 and 7.')

    rem3 = int(input())
    rem5 = int(input())
    rem7 = int(input())
    age = (rem3 * 70 + rem5 * 21 + rem7 * 15) % 105

    print("Your age is {0}; that's a good time to start programming!".format(age))


def count():
    print('Now I will prove to you that I can count to any number you want.')
    num = int(input())

    counter = 0
    while counter <= num:
        print("{0} !".format(counter))
        counter += 1


def test():
    print("Let's test your programming knowledge.")
    print("Why do we use methods?")
    print("1. To repeat a statement multiple times.")
    print("2. To decompose a program into several small subroutines.")
    print("3. To determine the execution time of a program.")
    print("4. To interrupt the execution of a program.")

    answer = 2
    guess = int(input())
    while guess != answer:
        print("Please, try again.")
        guess = int(input())

    print('Completed, have a nice day!')
    print('.....................................')
    print('.....................................')
    print('.....................................')


def end():
    print('Congratulations, have a nice day!')
    print('.....................................')
    print('.....................................')
    print('.....................................')
    input()


greet('Chatbot', '2023')
remind_name()
guess_age()
count()
test()
end()
```

Output:-

```
Hello! My name is Chatbot.
I was created in 2023.
Please, remind me your name.
 ABCD
What a great name you have,  ABCD!
Let me guess your age.
Enter remainders of dividing your age by 3, 5 and 7.
2
0
6
Your age is 20; that's a good time to start programming!
Now I will prove to you that I can count to any number you want.
5
0 !
1 !
2 !
3 !
4 !
5 !
Let's test your programming knowledge.
Why do we use methods?
1. To repeat a statement multiple times.
2. To decompose a program into several small subroutines.
3. To determine the execution time of a program.
4. To interrupt the execution of a program.
2
Completed, have a nice day!
.................................
.................................
.................................
Congratulations, have a nice day!
.................................
.................................
.................................
```

Practical Number: 6

Title: Implement any one Expert System (Help desks management)

Code:-

```
problem_dict = {
    "Printer not working": "Check that it's turned on and connected to the network",
    "Can't log in": "Make sure you're using the correct username and password",
    "Software not installing": "Check that your computer meets the system requirements",
    "Internet connection not working": "Restart your modem or router",
    "Email not sending": "Check that you're using the correct email server settings"
}

def handle_request(user_input):
    if user_input.lower() == "exit":
        return "Goodbye!"
    elif user_input in problem_dict:
        return problem_dict[user_input]
    else:
        return "I'm sorry, I don't know how to help with that problem."

while True:
    user_input = input("What's the problem? Type 'exit' to quit. ")
    response = handle_request(user_input)
    print(response)
```

Output:-

```
What's the problem? Type 'exit' to quit. Software not installing
Check that your computer meets the system requirements
What's the problem? Type 'exit' to quit.  Printer not working
I'm sorry, I don't know how to help with that problem.
What's the problem? Type 'exit' to quit. exit
Goodbye!
```

Practical No: 7

Practical Title: Case study on Amazon EC2 and learn about Amazon EC2 web services.

Objectives:

- To learn Amazon EC2 web services
- To study on Amazon EC2 and learn about Amazon EC2 web services.

Hardware Requirements :

- Pentium IV with latest configuration

Software Requirements :

- Ubuntu 20.04

Theory:

An EC2 instance is nothing but a virtual server in Amazon Web services terminology. It stands for Elastic Compute Cloud. It is a web service where an AWS subscriber can request and provision a compute server in AWS cloud.

An on-demand EC2 instance is an offering from AWS where the subscriber/user can rent the virtual server per hour and use it to deploy his/her own applications.

The instance will be charged per hour with different rates based on the type of the instance chosen. AWS provides multiple instance types for the respective business needs of the user.

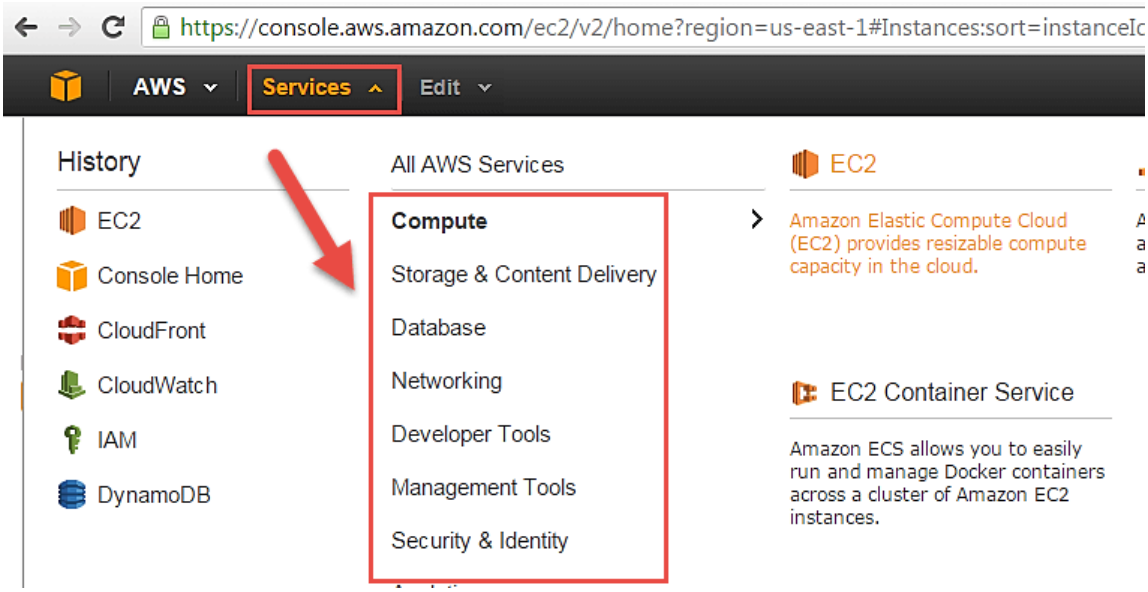
Thus, you can rent an instance based on your own CPU and memory requirements and use itas long as you want. You can terminate the instance when it's no more used and save on costs. This is the most striking advantage of an on-demand instance- you can drastically save on your CAPEX.

Let us see in detail how to launch an on-demand EC2 instance in AWS

Cloud. Login and access to AWS services

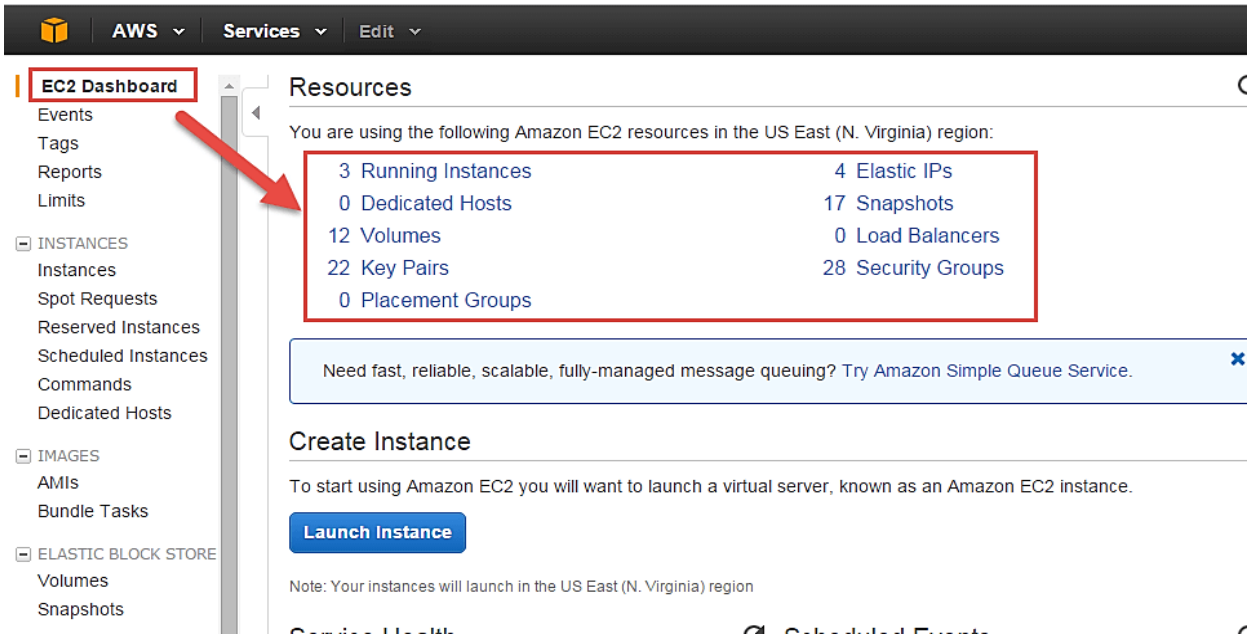
Step 1) In this step,

- Login to your AWS account and go to the AWS Services tab at the top left corner.
- Here, you will see all of the AWS Services categorized as per their area viz. Compute, Storage, Database, etc. For creating an EC2 instance, we have to choose Computeà EC2 as in the next step.



- Open all the services and click on EC2 under Compute services. This will launch the dashboard of EC2.

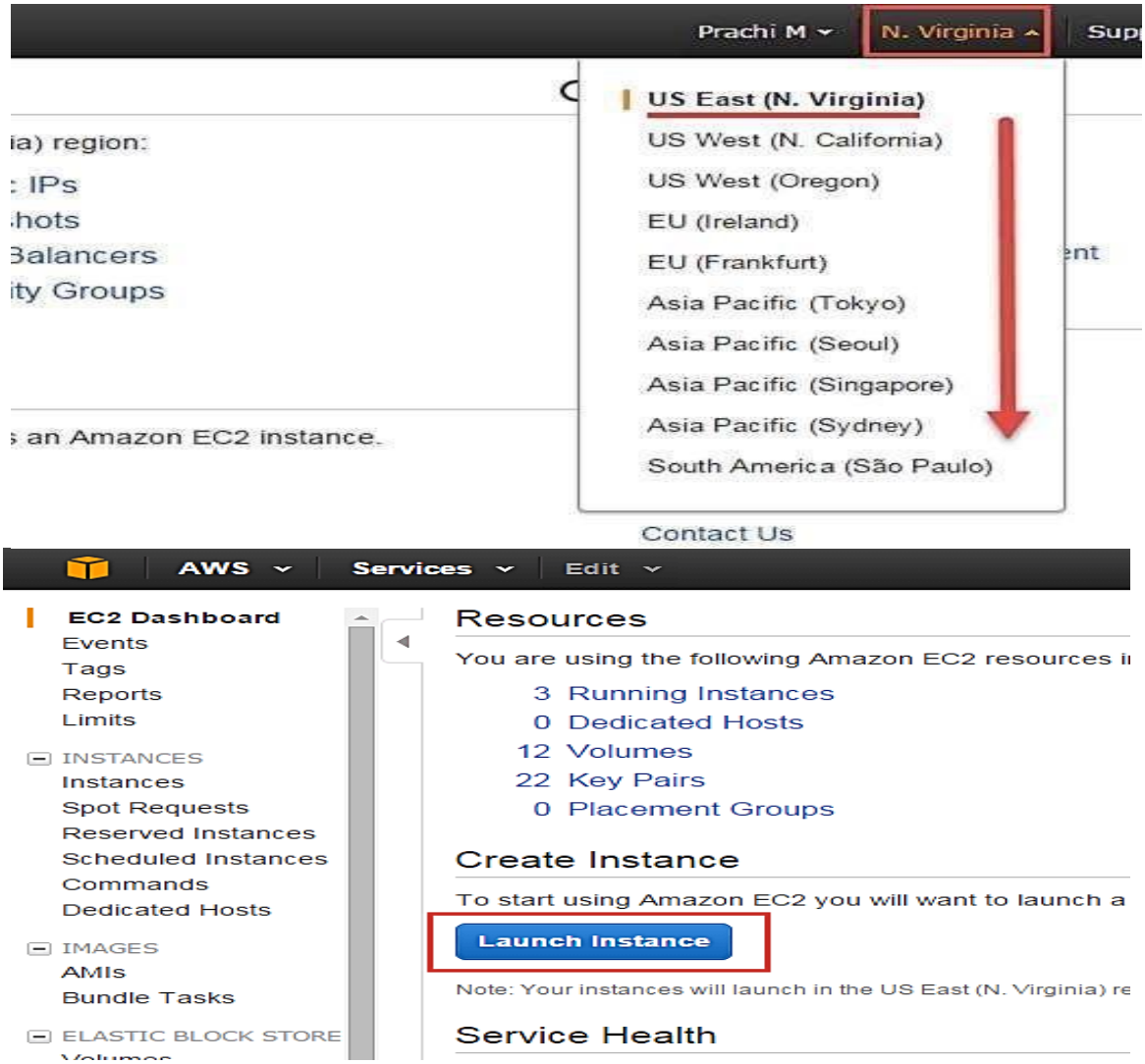
Here is the EC2 dashboard. Here you will get all the information in gist about the AWS EC2 resources running.



Step 2) On the top right corner of the EC2 dashboard, choose the AWS Region in which you want to provision the EC2 server.
Here we are selecting N. Virginia. AWS provides 10 Regions all over the globe

Step 3) In this step

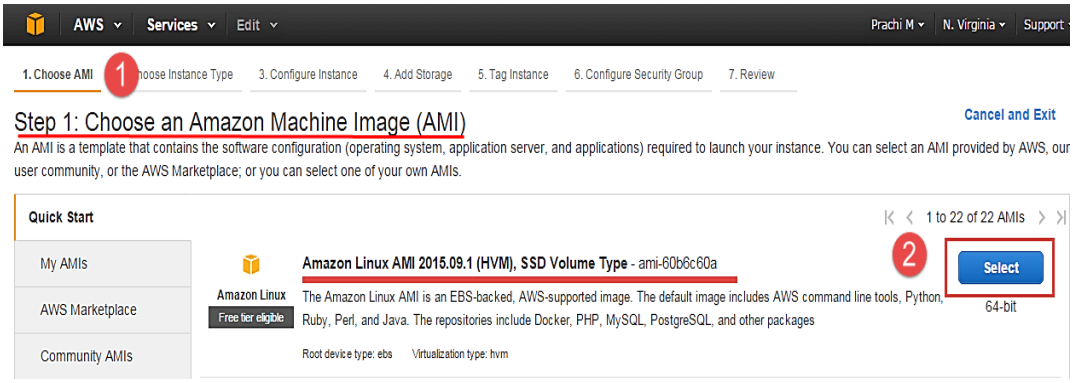
- Once your desired Region is selected, come back to the EC2 Dashboard.
- Click on 'Launch Instance' button in the section of Create Instance (as shown below).



- Instance creation wizard page will open as soon as you click 'Launch Instance'. Choose AMI

Step 1) In this step we will do,

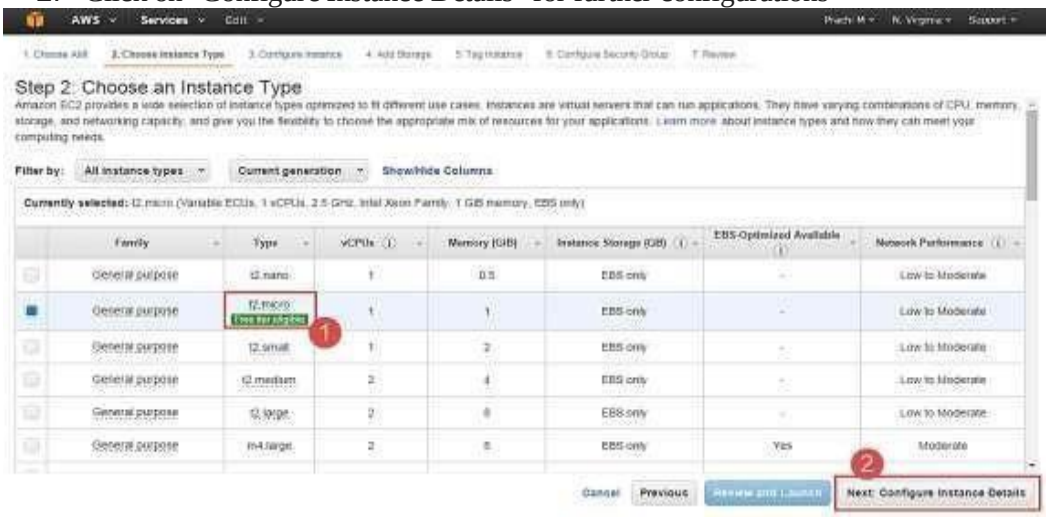
1. You will be asked to choose an AMI of your choice. (An AMI is an Amazon Machine Image. It is a template basically of an Operating System platform which you can use as a base to create your instance). Once you launch an EC2 instance from your preferred AMI, the instance will automatically be booted with the desired OS. (We will see more about AMIs in the coming part of the tutorial).
2. Here we are choosing the default Amazon Linux (64 bit) AMI.



Choose EC2 Instance Types

Step 1) In the next step, you have to choose the type of instance you require based on your business needs.

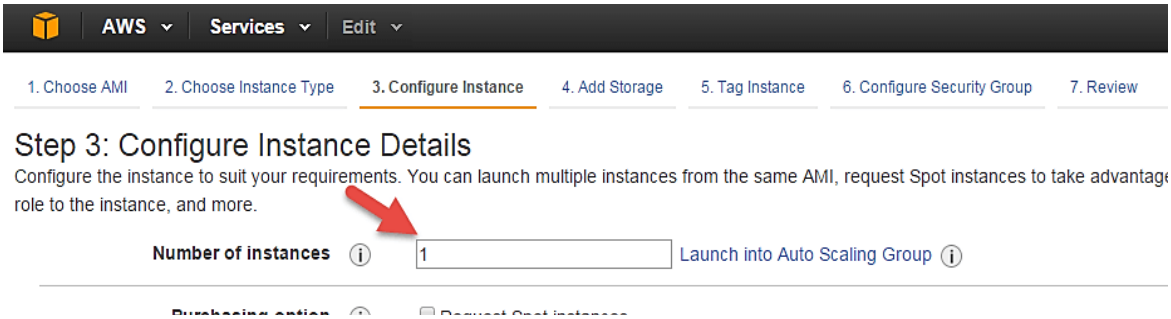
1. We will choose t2.micro instance type, which is a 1vCPU and 1GB memory server offered by AWS.
2. Click on "Configure Instance Details" for further configurations



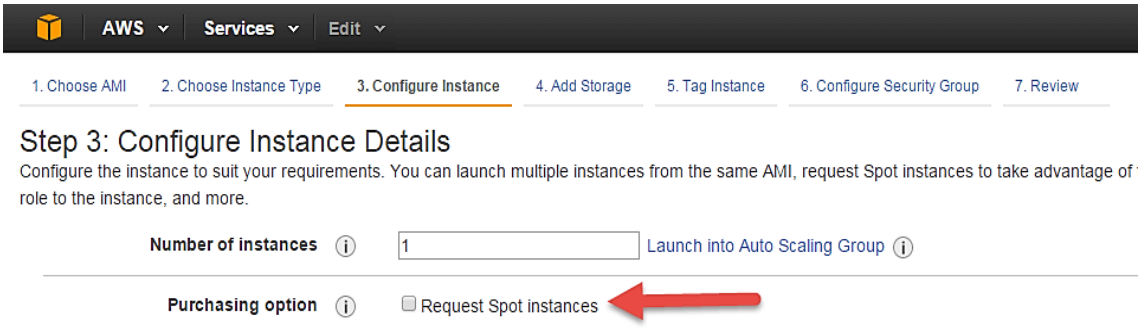
- In the next step of the wizard, enter details like no. of instances you want to launch at a time.
- Here we are launching one

instance. Configure Instance

Step 1) No. of instances- you can provision up to 20 instances at a time. Here we are launching one instance.



Step 2) Under Purchasing Options, keep the option of 'Request Spot Instances' unchecked as ofnow. (This is done when we wish to launch Spot instances instead of on-demand ones. We will come back to Spot instances in the later part of the tutorial).



Step 3) Next, we have to configure some basic networking details for our EC2 server.

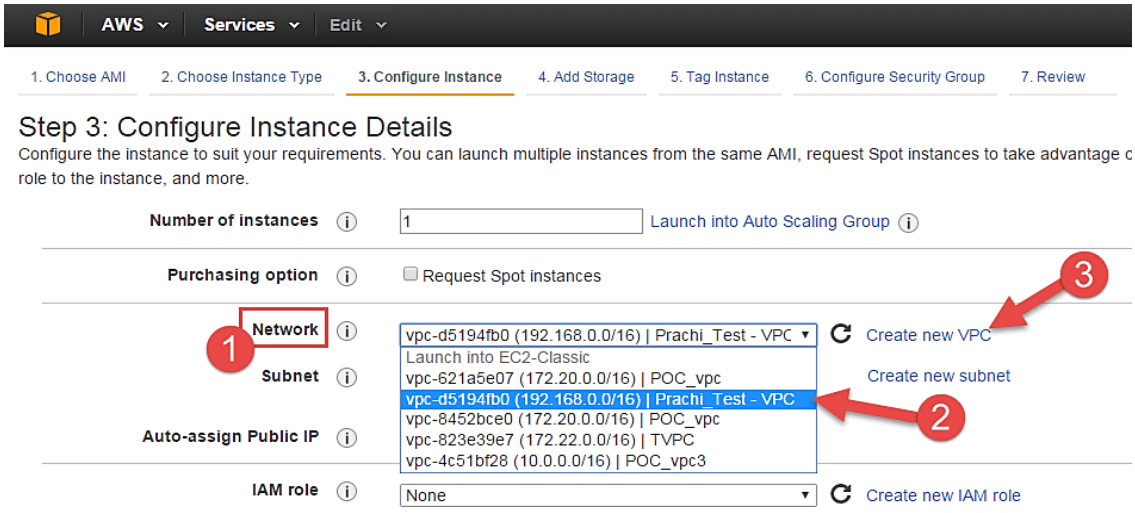
- You have to decide here, in which VPC (Virtual Private Cloud) you want to launch your instance and under which subnets inside your VPC. It is better to determine and plan this prior to launching the instance. Your AWS architecture set-up should include IP ranges for your subnets etc. pre-planned for better management. (We will see how to create a new VPC in Networking section of the tutorial.

- Subnetting should also be pre-planned. E.g.: If it's a web server you should place it in the public subnet and if it's a DB server, you should place it in a private subnet all inside your VPC.

Below,

1. Network section will give a list of VPCs available in our platform.
2. Select an already existing VPC
3. You can also create a new VPC

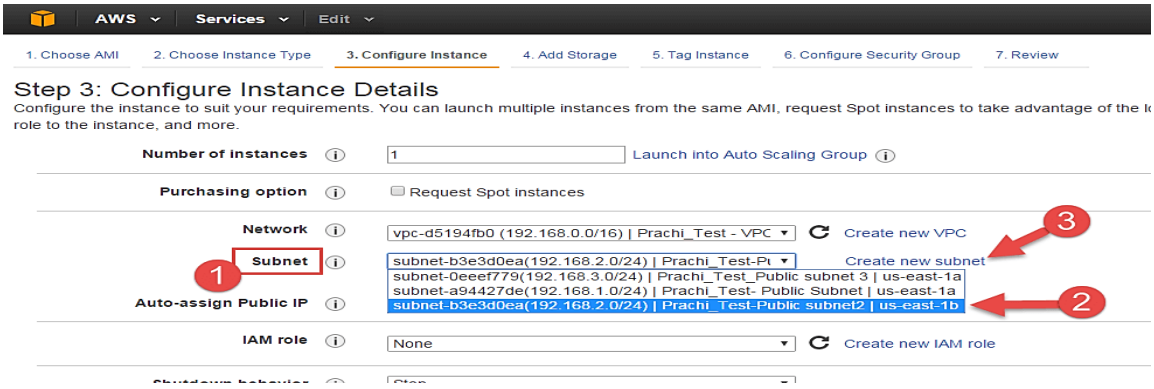
Here I have selected an already existing VPC where I want to launch my instance.



Step 4) In this step,

- A VPC consists of subnets, which are IP ranges that are separated for restricting access.
- Below,

1. Under Subnets, you can choose the subnet where you want to place your instance.
2. I have chosen an already existing public subnet.
3. You can also create a new subnet in this step.

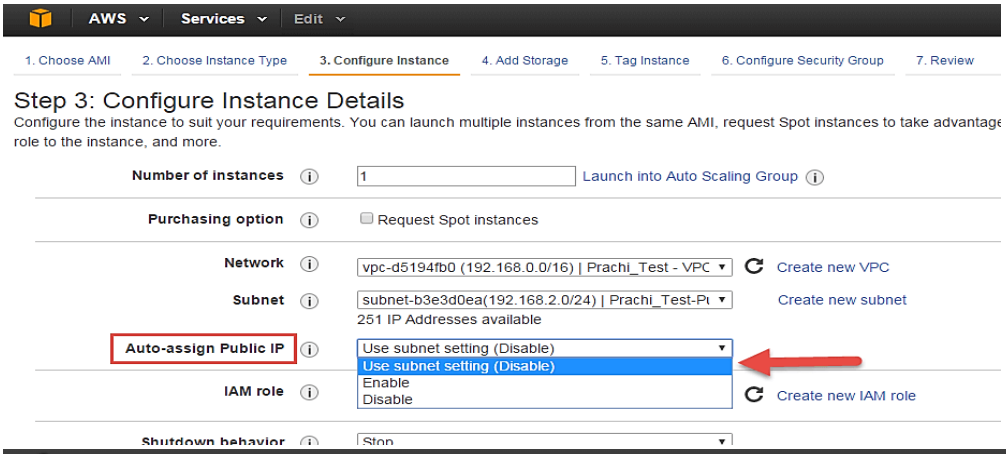


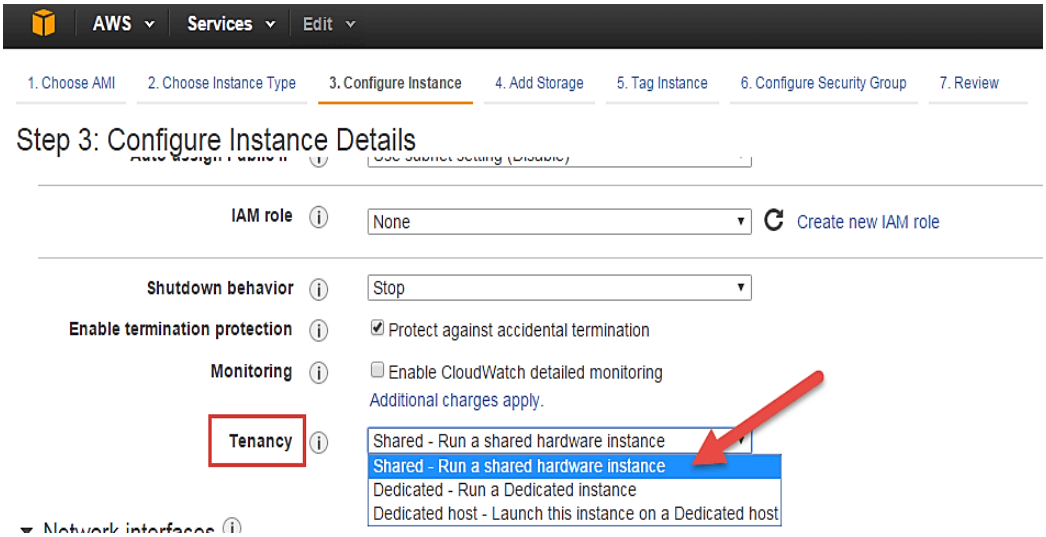
- Once your instance is launched in a public subnet, AWS will assign a dynamic public IP to it from their pool of IPs.

Step 5) In this step,

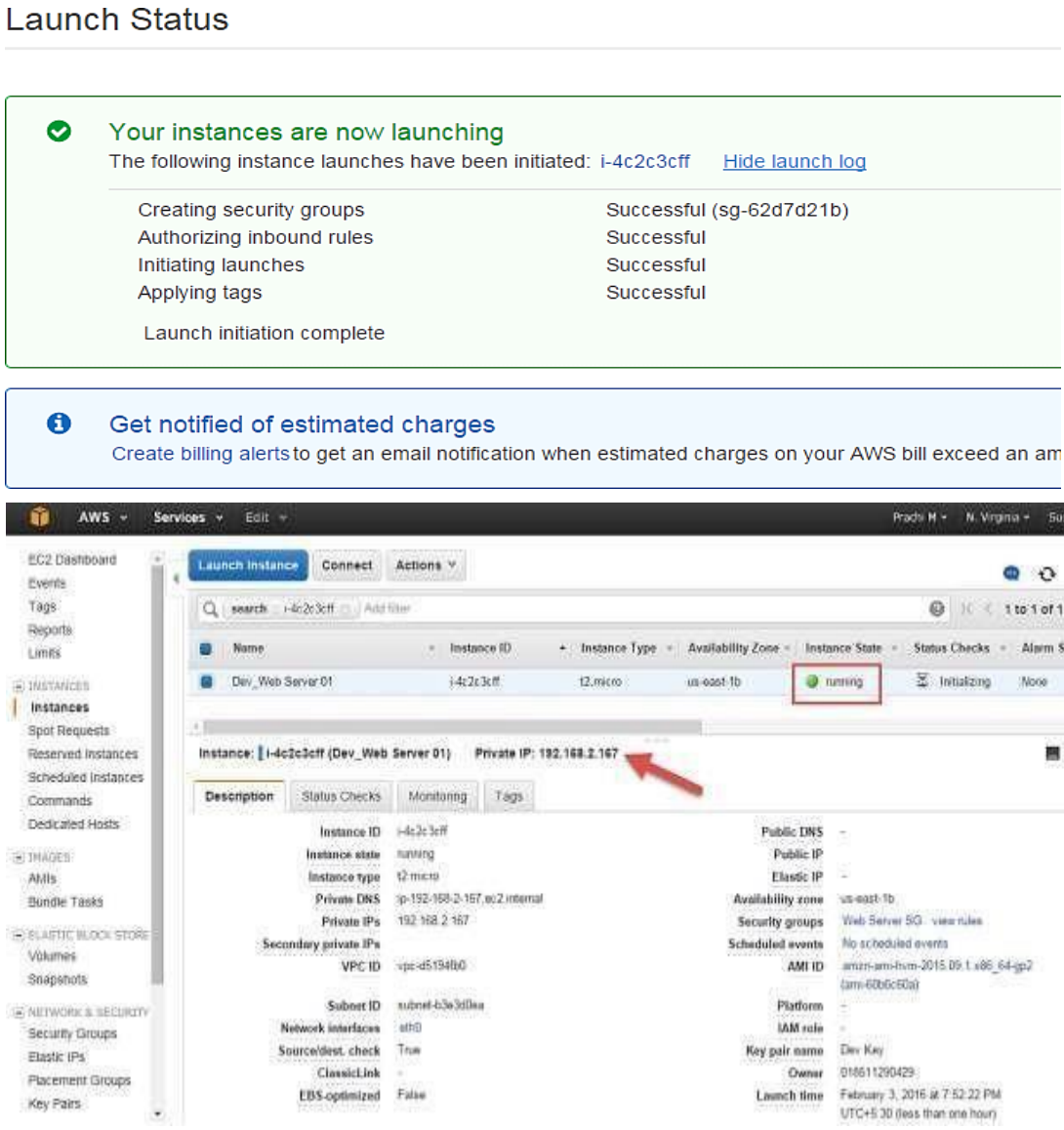
- You can choose if you want AWS to assign it an IP automatically, or you want to do it manually later. You can enable/ disable 'Auto assign Public IP' feature here likewise.

- Here we are going to assign this instance a static IP called as EIP (Elastic IP) later. So we keep this feature disabled as of now.


AWS
Services
Edit
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance
6. Configure Security Group
7. Review
Step 3: Configure Instance Details
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage
role to the instance, and more.
Number of instances
1
Launch into Auto Scaling Group
Purchasing option
Request Spot instances
Network
vpc-d5194fb0 (192.168.0.0/16) | Prachi_Test - VPC
Create new VPC
Subnet
subnet-b3e3d0ea(192.168.2.0/24) | Prachi_Test-Pt
Create new subnet
251 IP Addresses available
Auto-assign Public IP
Use subnet setting (Disable)
Use subnet setting (Disable)
Enable
Disable
IAM role
Create new IAM role
Shutdown behavior
Stop


AWS
Services
Edit
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance
6. Configure Security Group
7. Review
Step 3: Configure Instance Details
IAM role
None
Create new IAM role
Shutdown behavior
Stop
Enable termination protection
Protect against accidental termination
Monitoring
Enable CloudWatch detailed monitoring
Additional charges apply.
Tenancy
Shared - Run a shared hardware instance
Shared - Run a shared hardware instance
Dedicated - Run a Dedicated instance
Dedicated host - Launch this instance on a Dedicated host
Network interfaces


Step 3: Configure Instance Details
Number of instances
Purchasing option
Network
Subnet
Auto-assign Public IP
IAM role
Shutdown behavior
Enable termination protection
Monitoring
Tenancy
Cancel
Previous
Review and Launch
Next: Add Storage



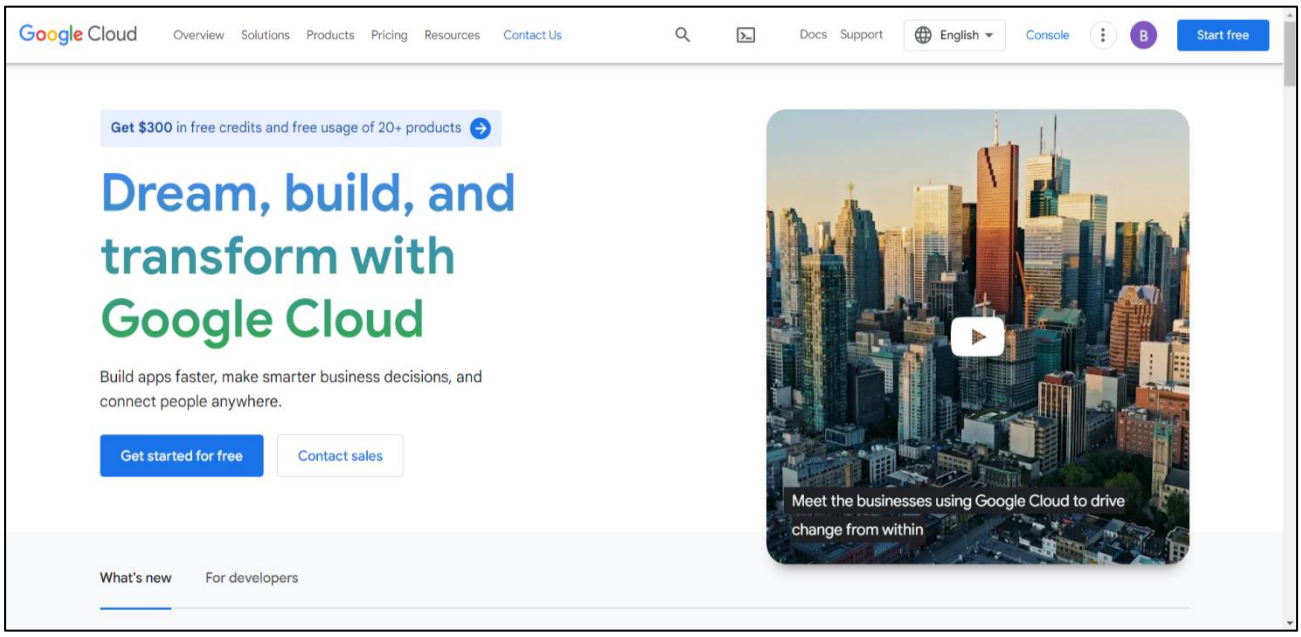
Conclusion:

Thus, we saw in detail how to create an on-demand EC2 instance in this tutorial. Because it is an on-demand server, you can keep it running when in use and 'Stop' it when it's unused to save on your costs

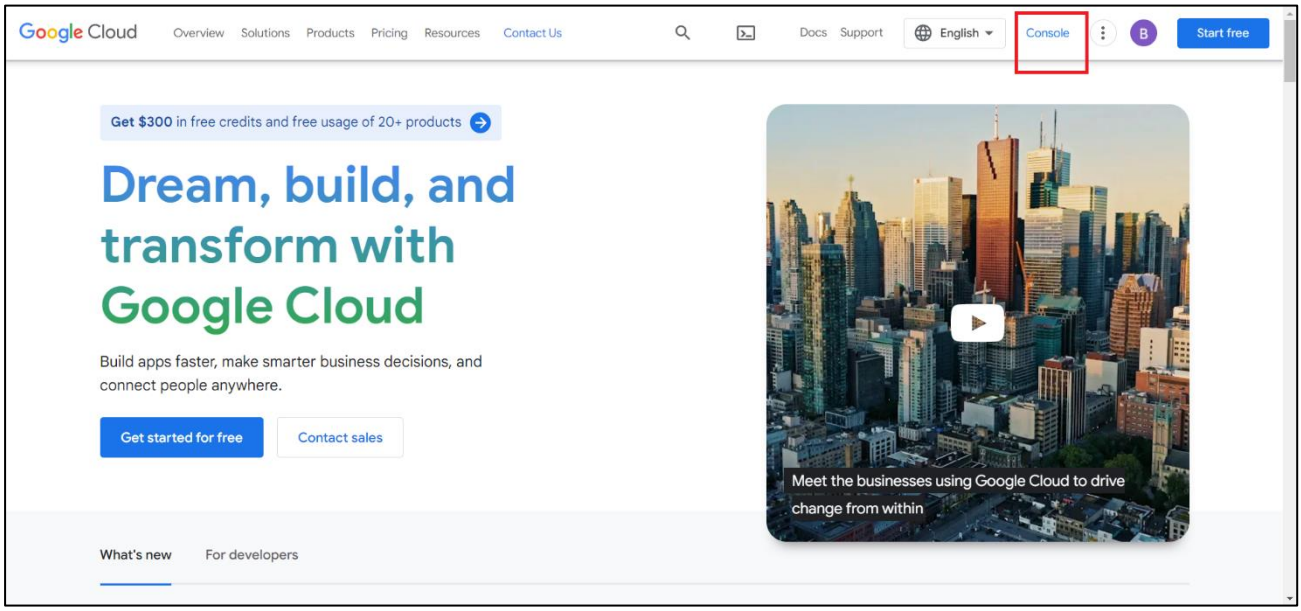
Practical Number: 8

Title: Installation and Configure Google App Engine.

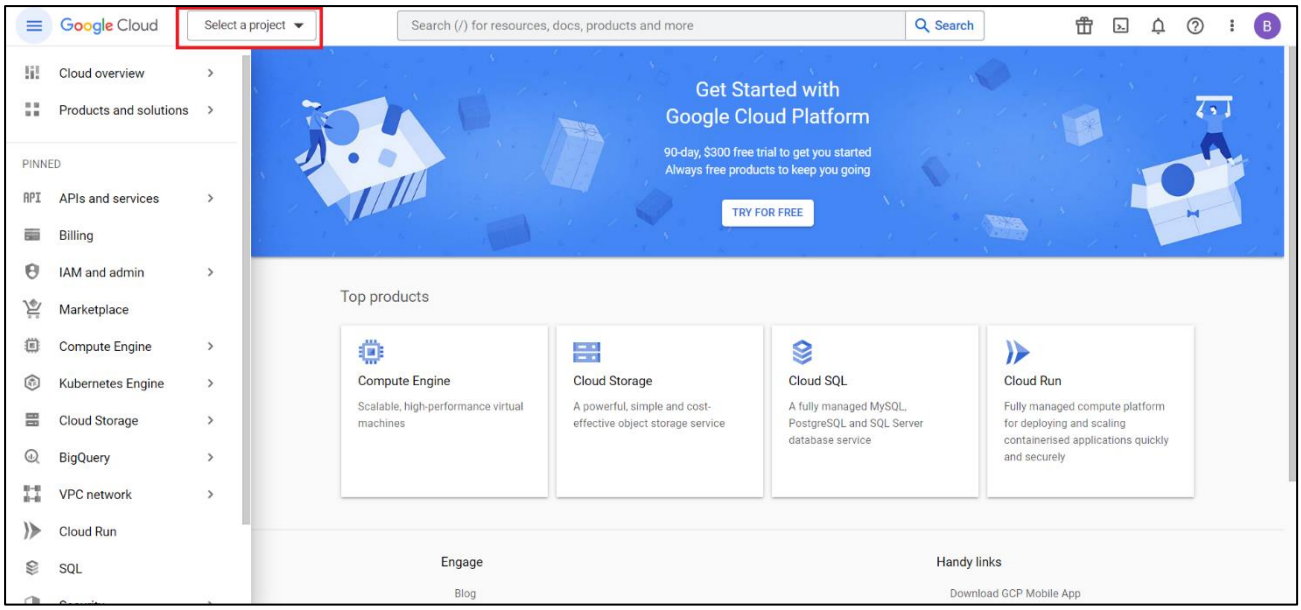
- **Go to the Google Cloud Platform(GCP) Website and Sign in to your Google account: https://cloud.google.com**



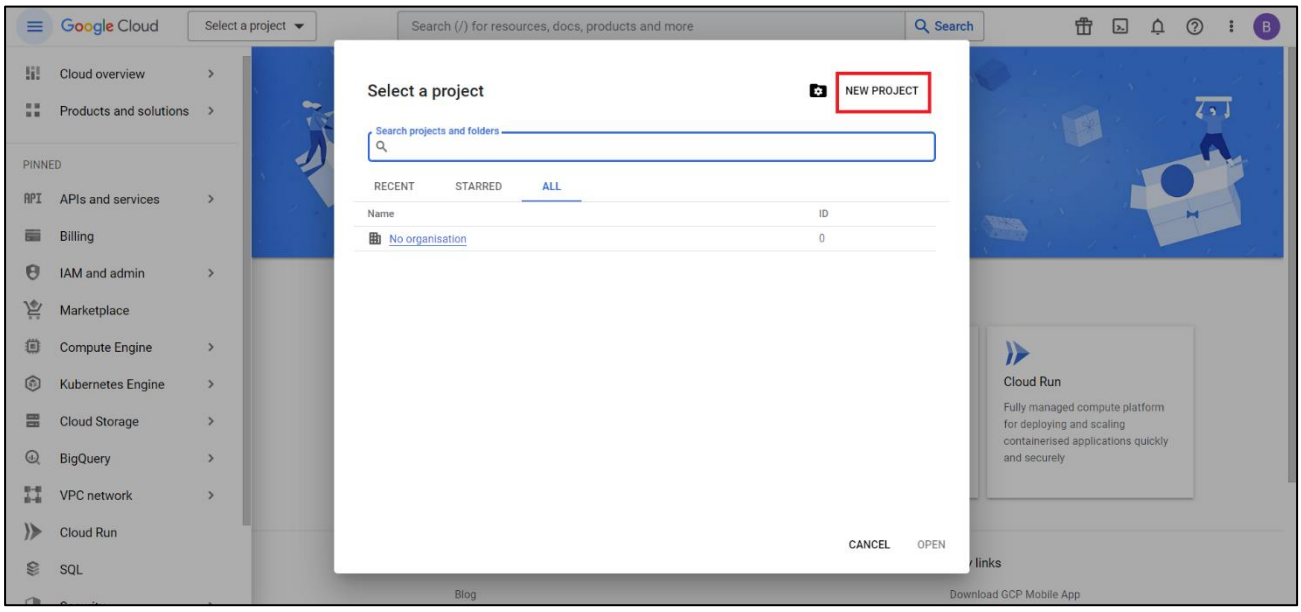
- **Click on "Console":**



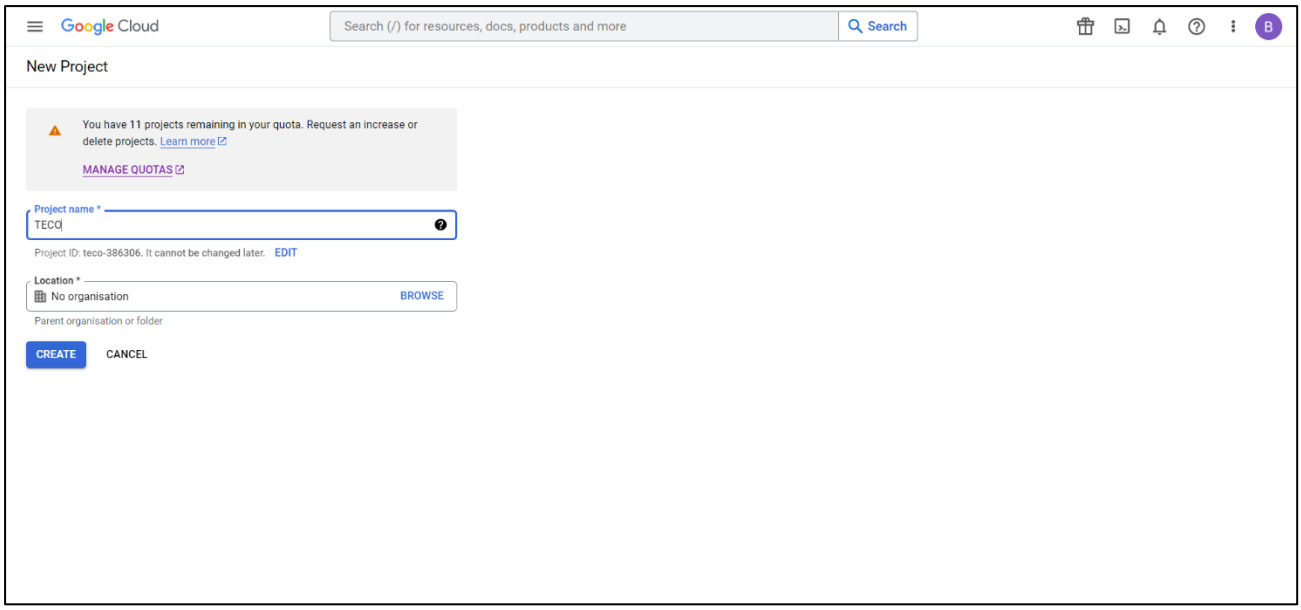
➢ **Click on "Select a Project":**



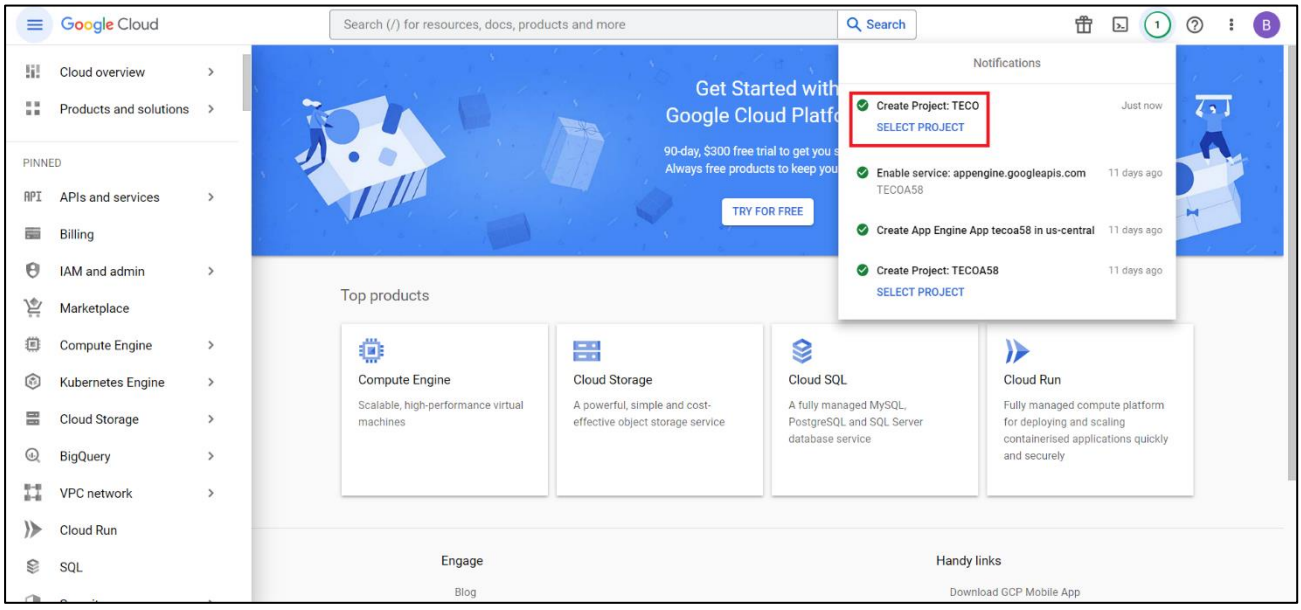
➢ **Click on "NEW PROJECT":**



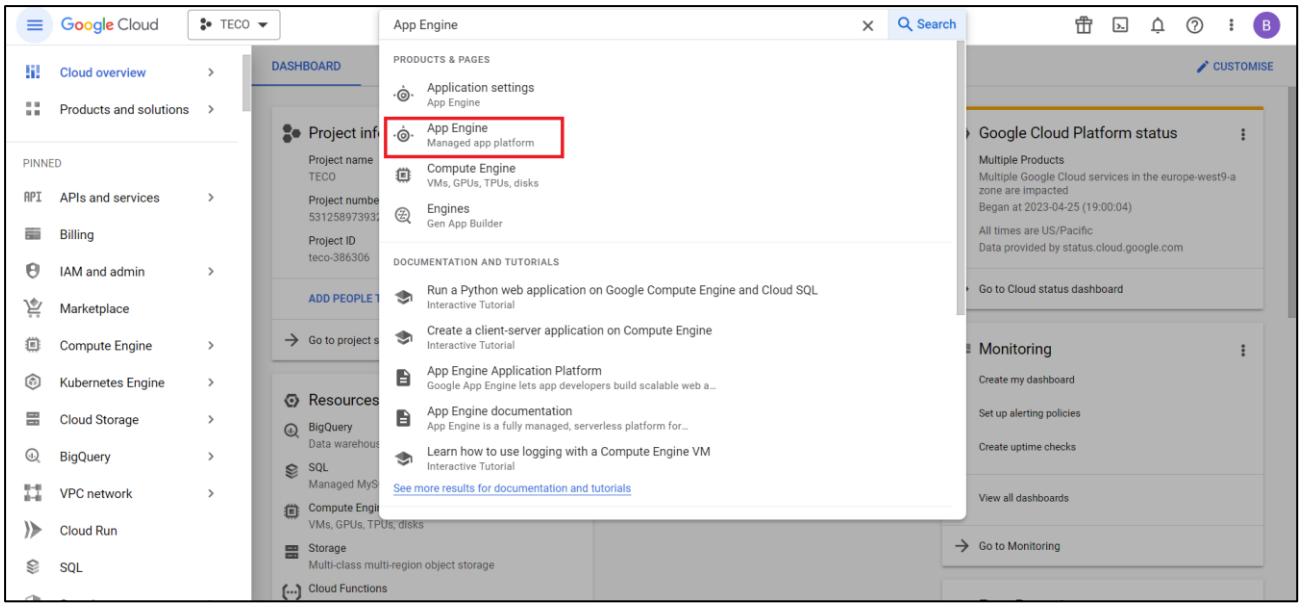
➢ **Enter Project Name :**



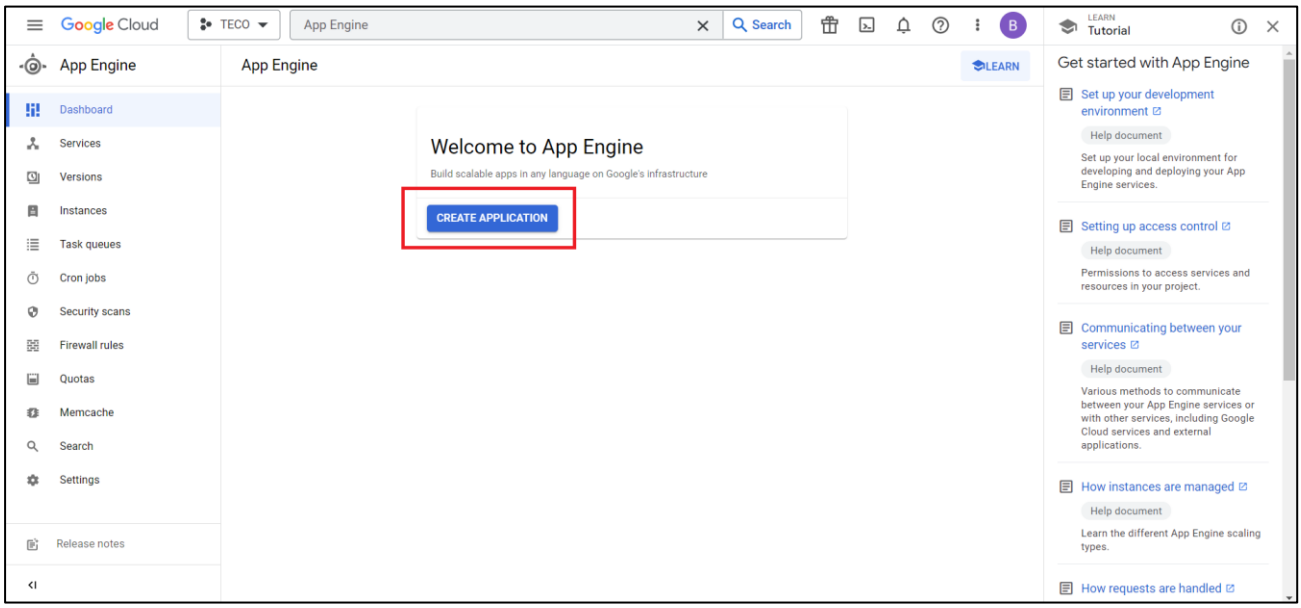
➢ **Click on "Select Project" from the notifications:**



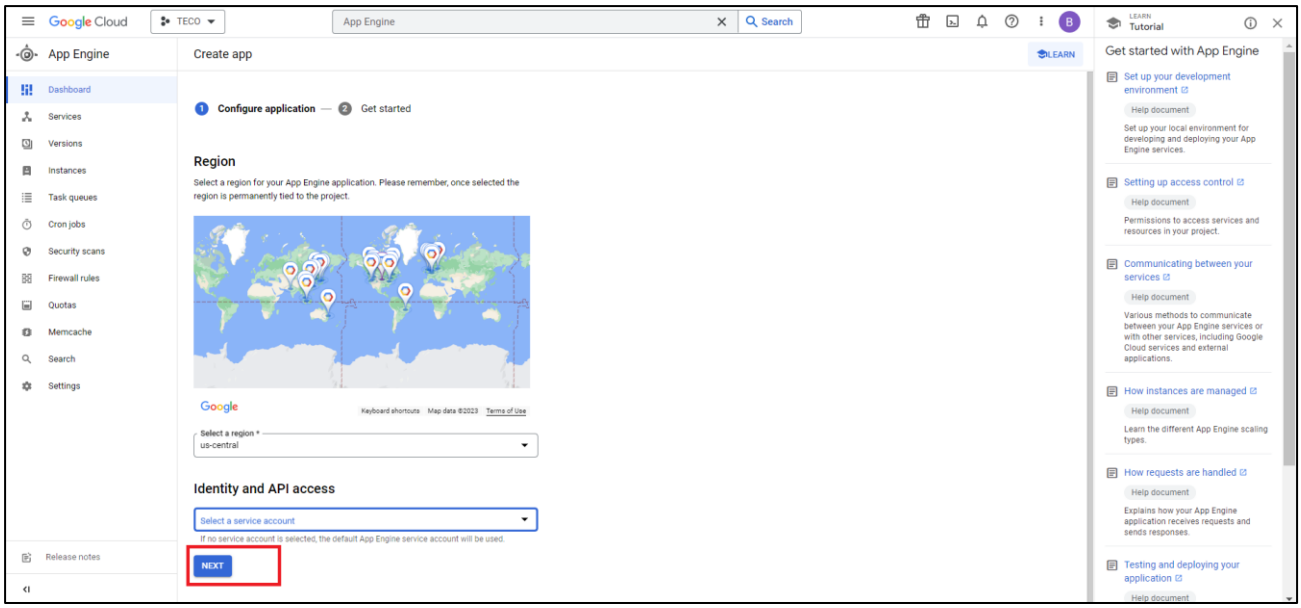
➢ **Search “App Engine” and select it from the list:**



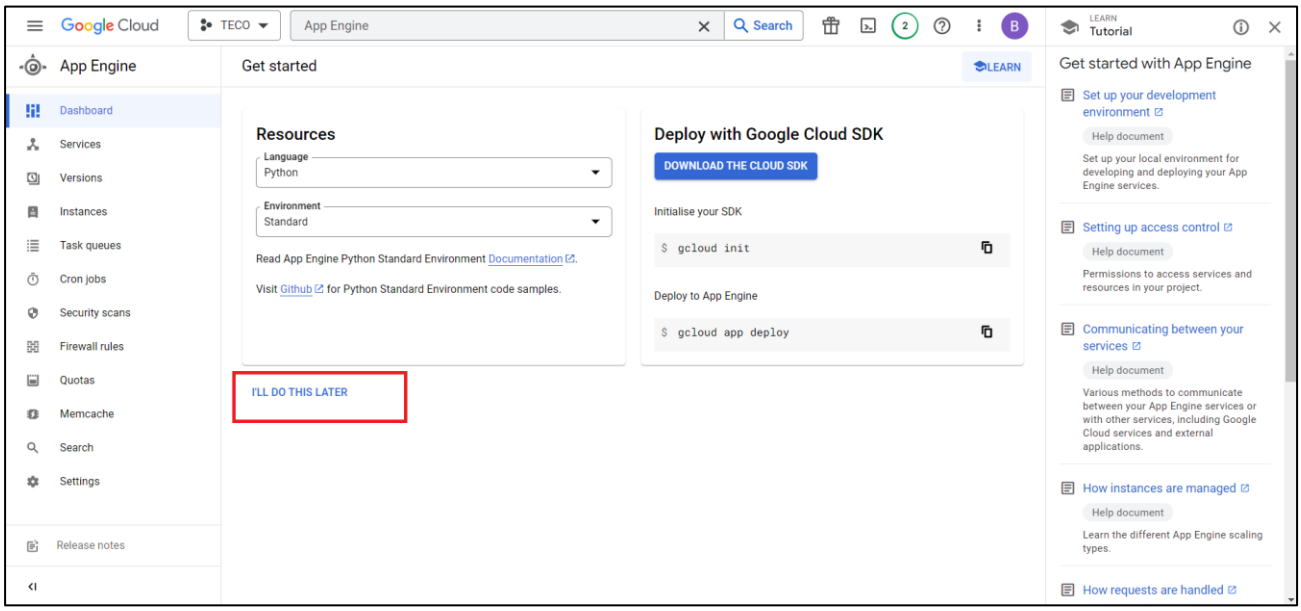
➢ **Click on “CREATE APPLICATION”:**



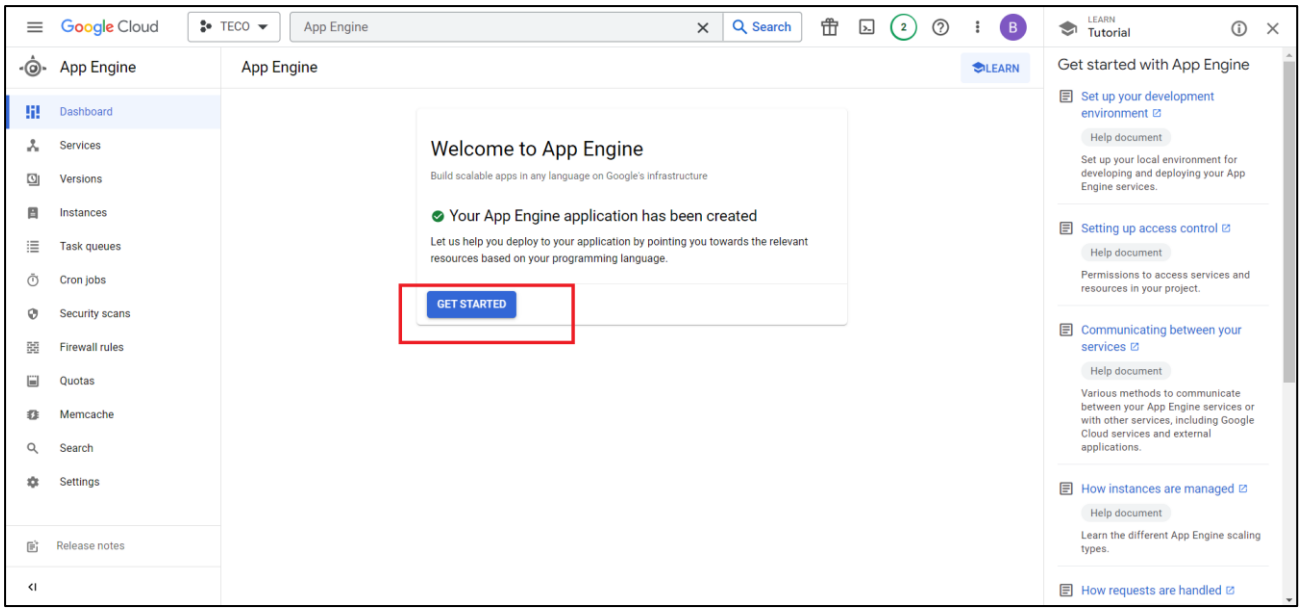
➢ **Click on "Next":**



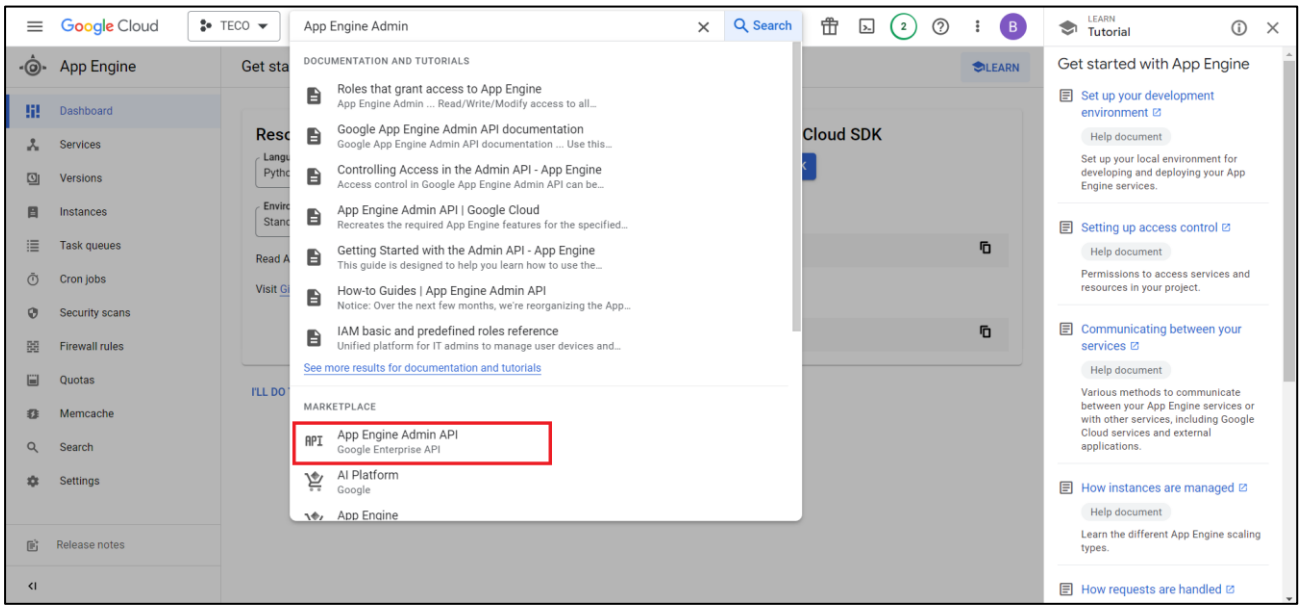
➢ **Click on " I'LL DO THIS LATER":**



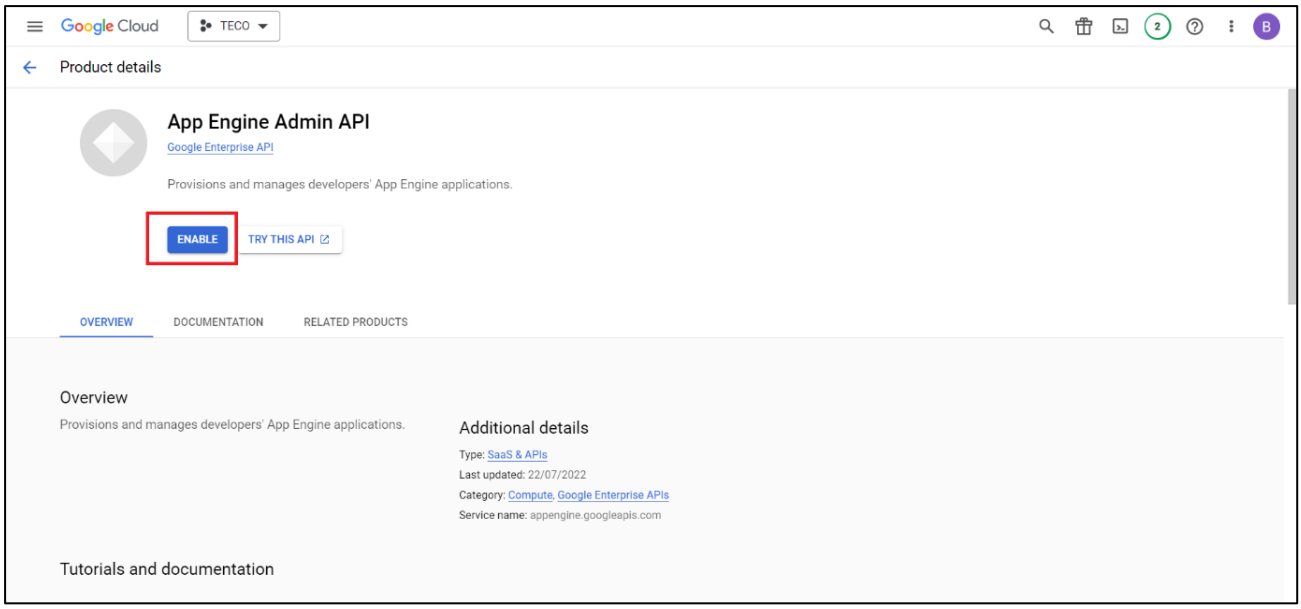
➢ **Click on " GET STARTED":**



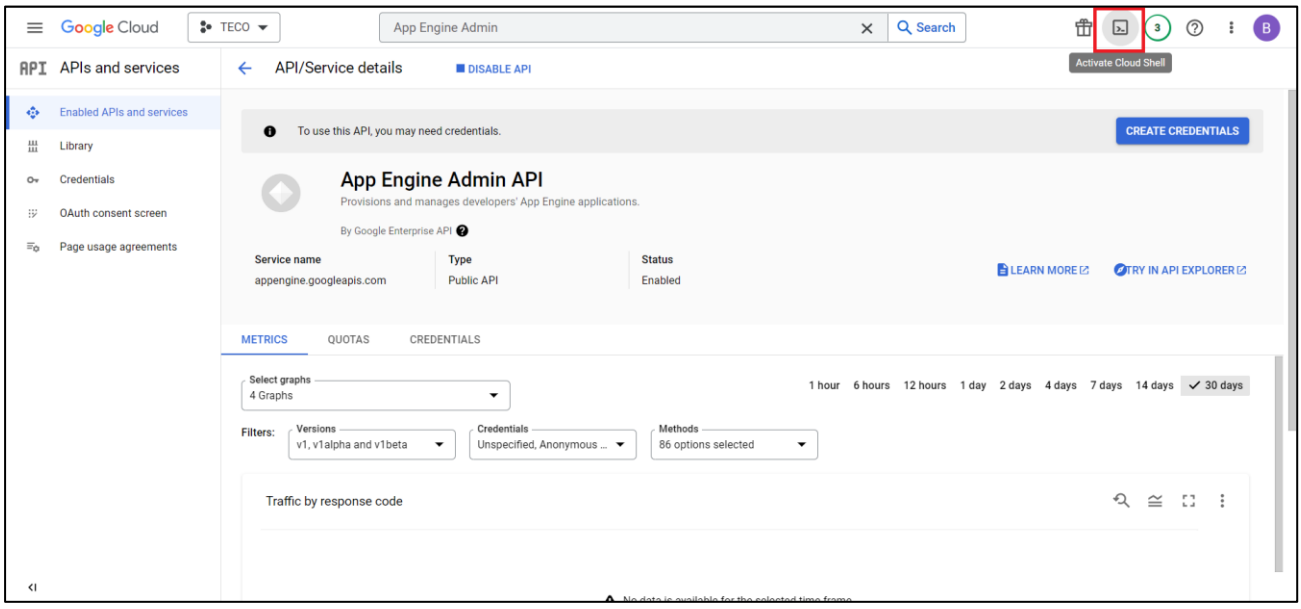
➢ **Search "App Engine Admin API" and select it from the list:**



➢ **Click on “Enable”:**

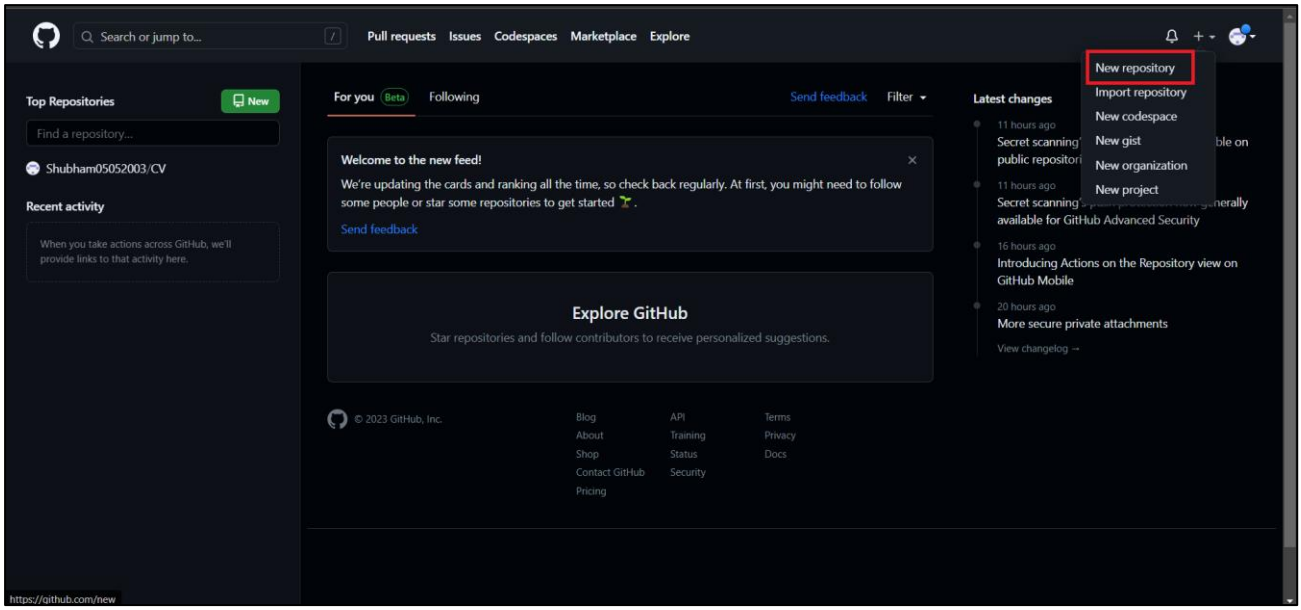


➢ **Click on “Cloud Shell Icon” on the top right corner of the window:**

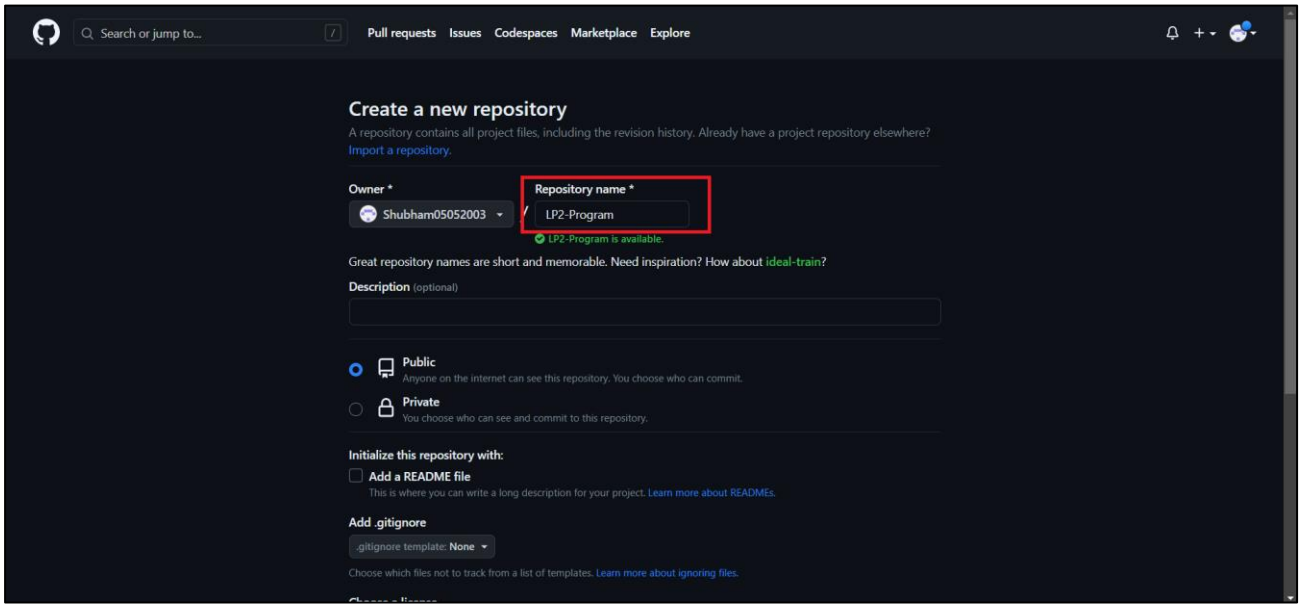


- **Log in to GitHub (https://github.com) :**

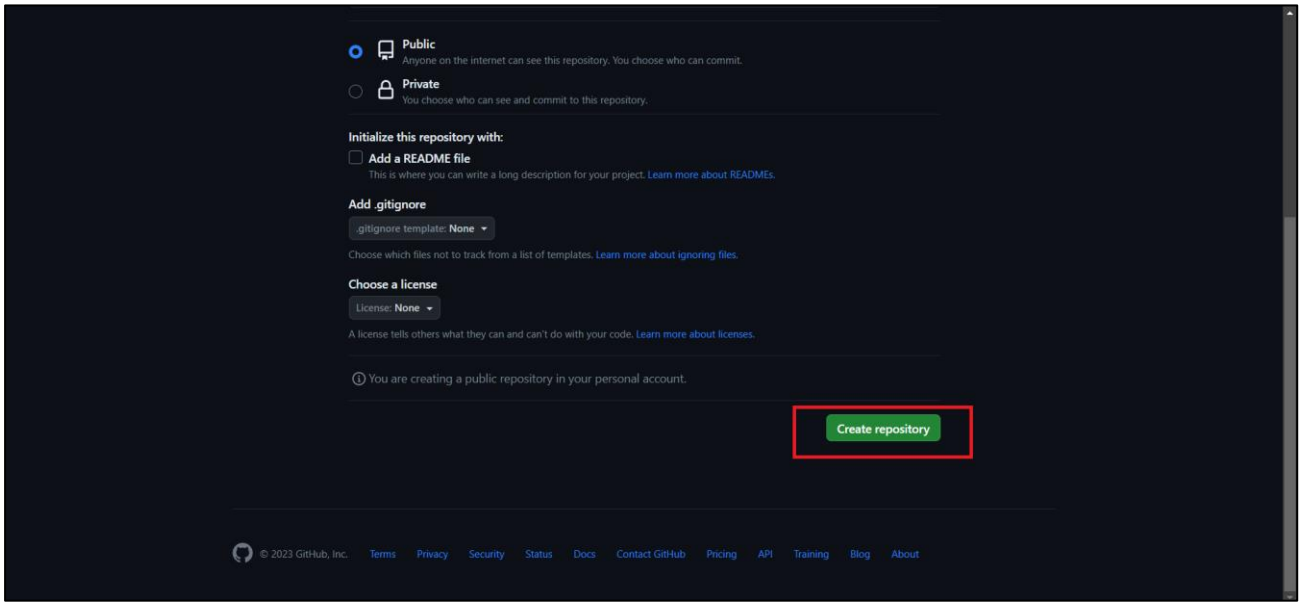
- **Click on '+' icon at top right corner and select "New Repository":**



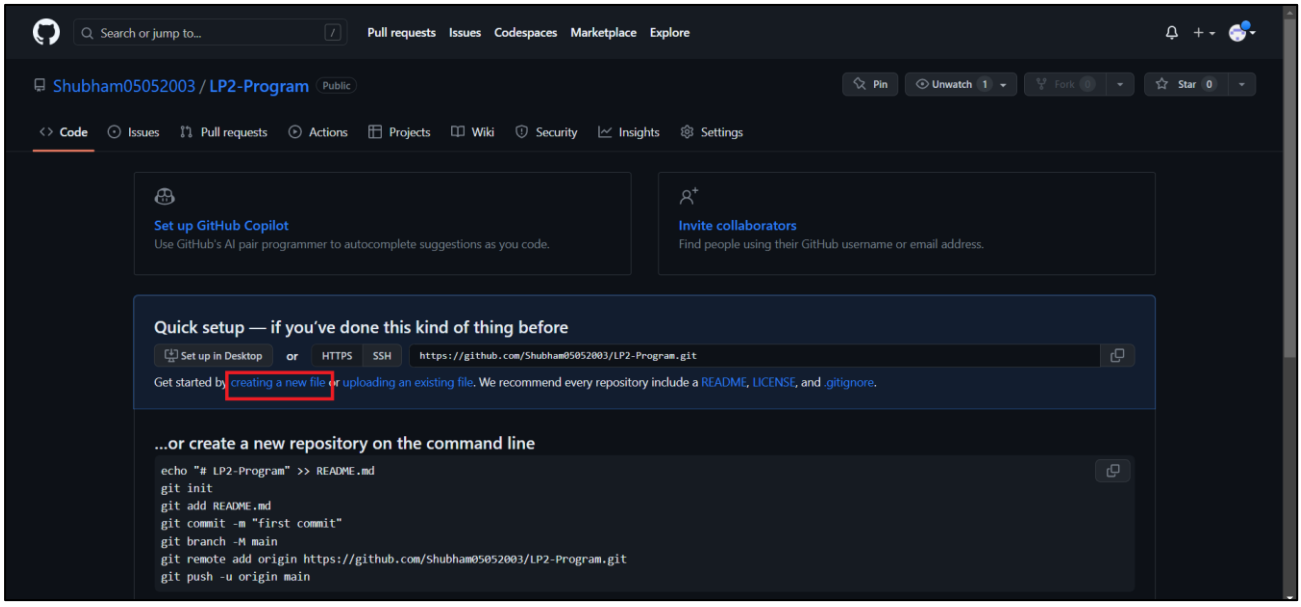
- **Give a name to the repository:**



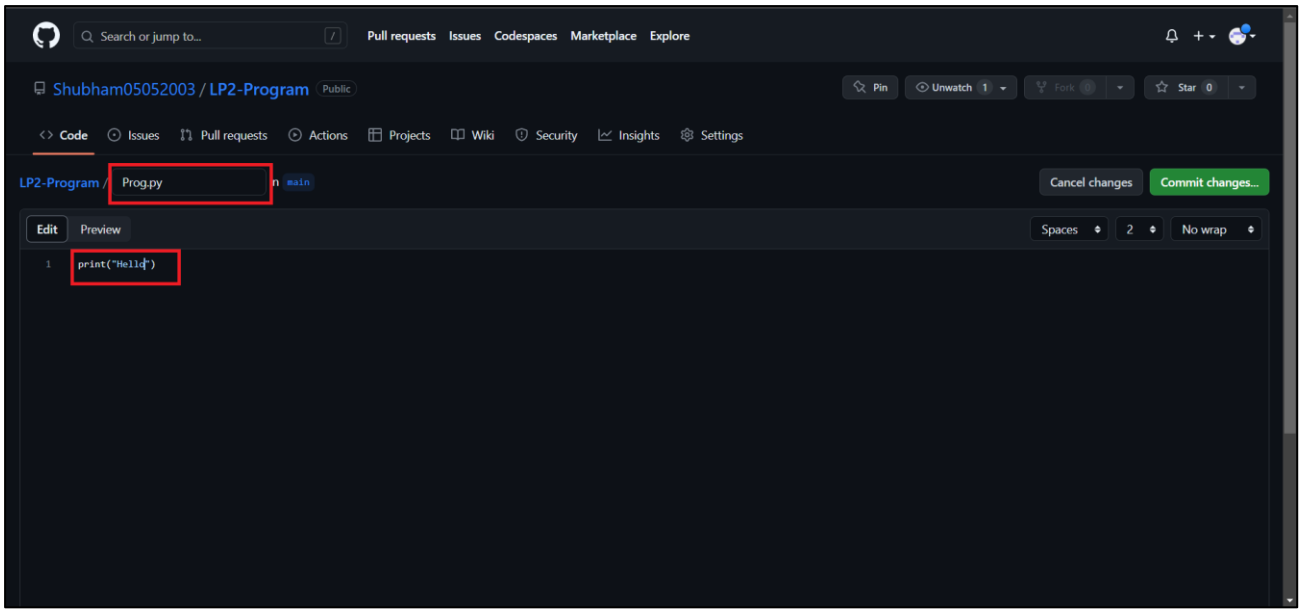
➢ **Click on “Create Repository”:**



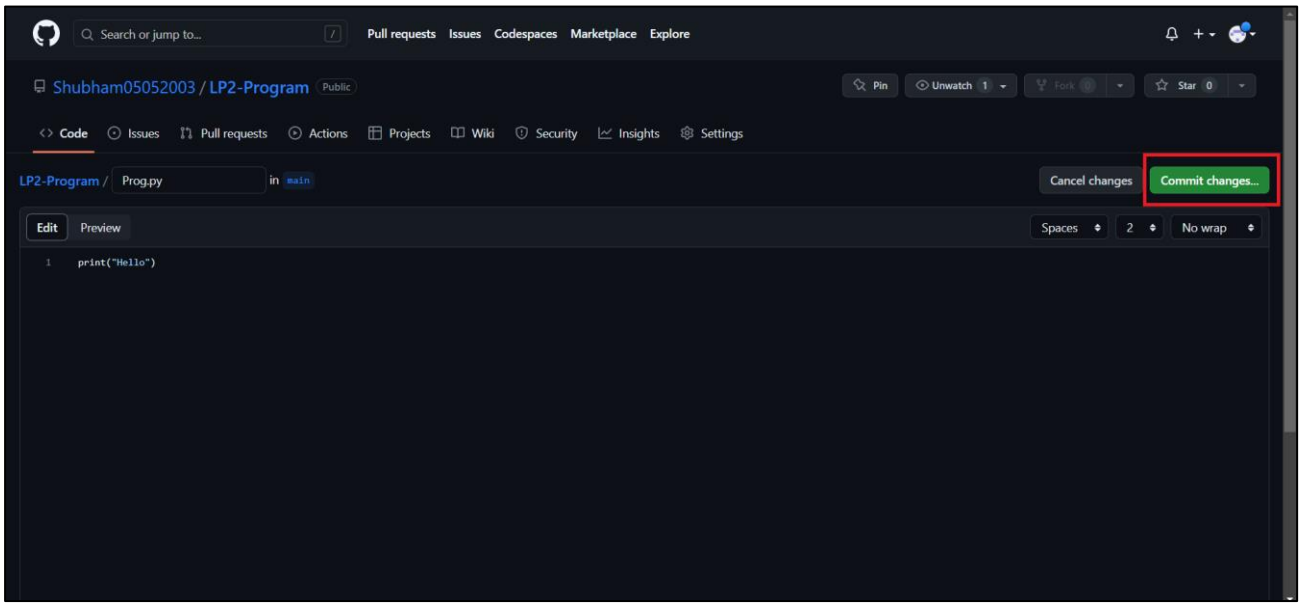
➢ **Click on “creating a new file”:**



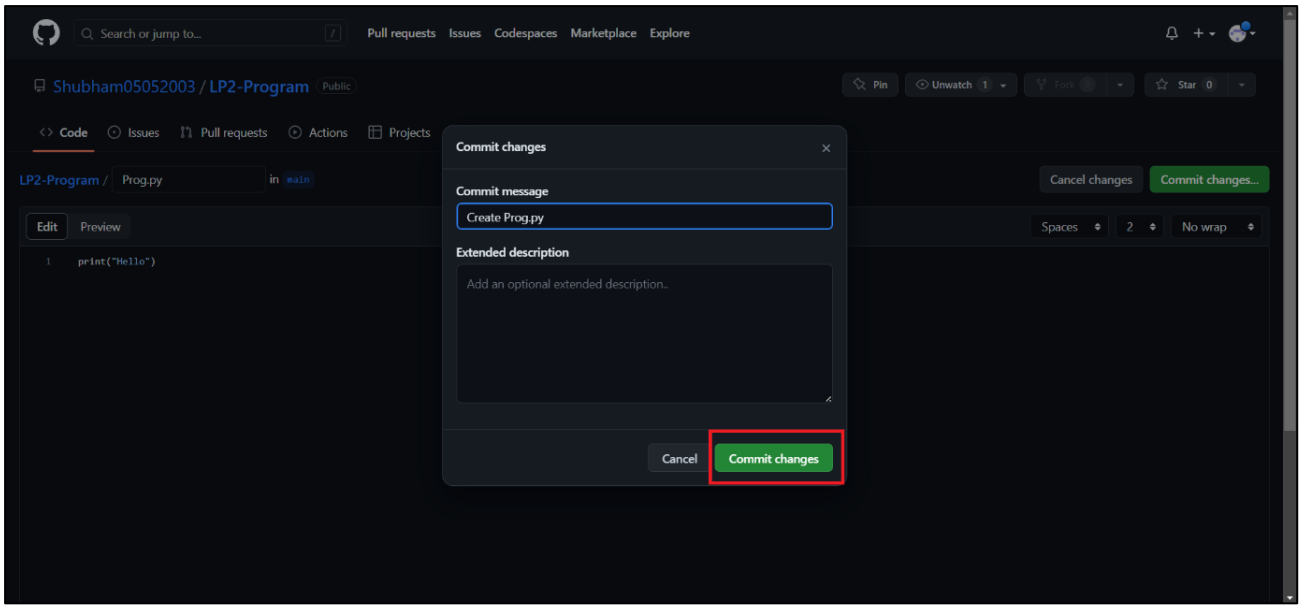
➢ **Give a file name i.e “Prog.py” and write Python code in it i.e print(“Hello”):**



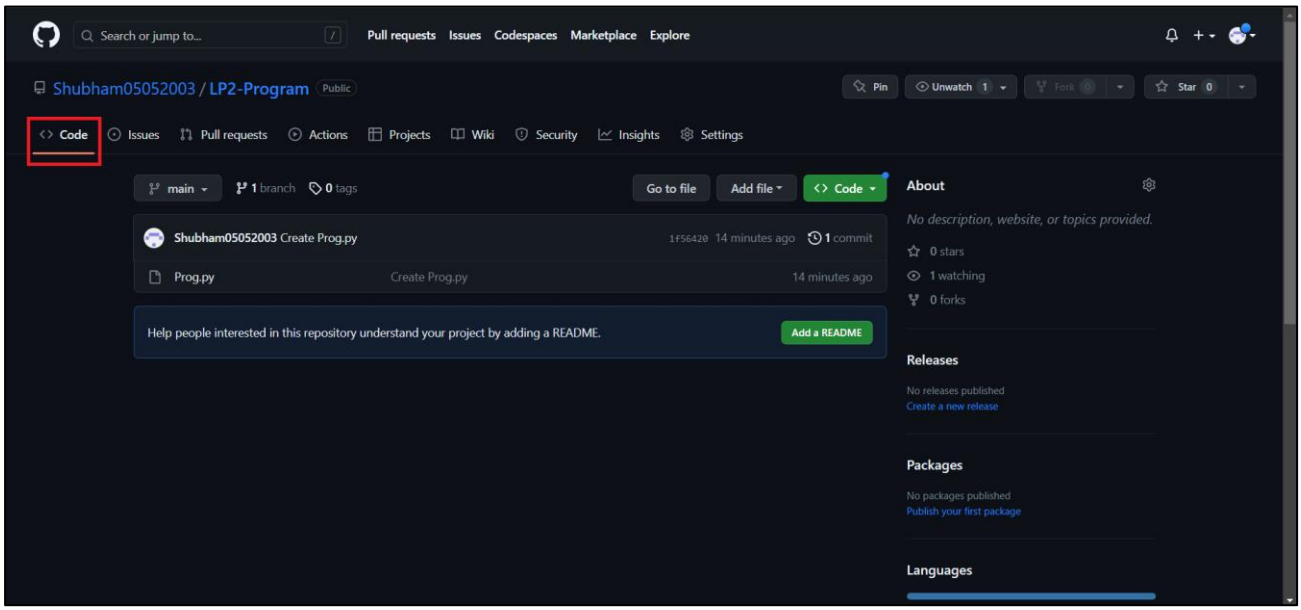
➢ **Click on “Commit changes”:**



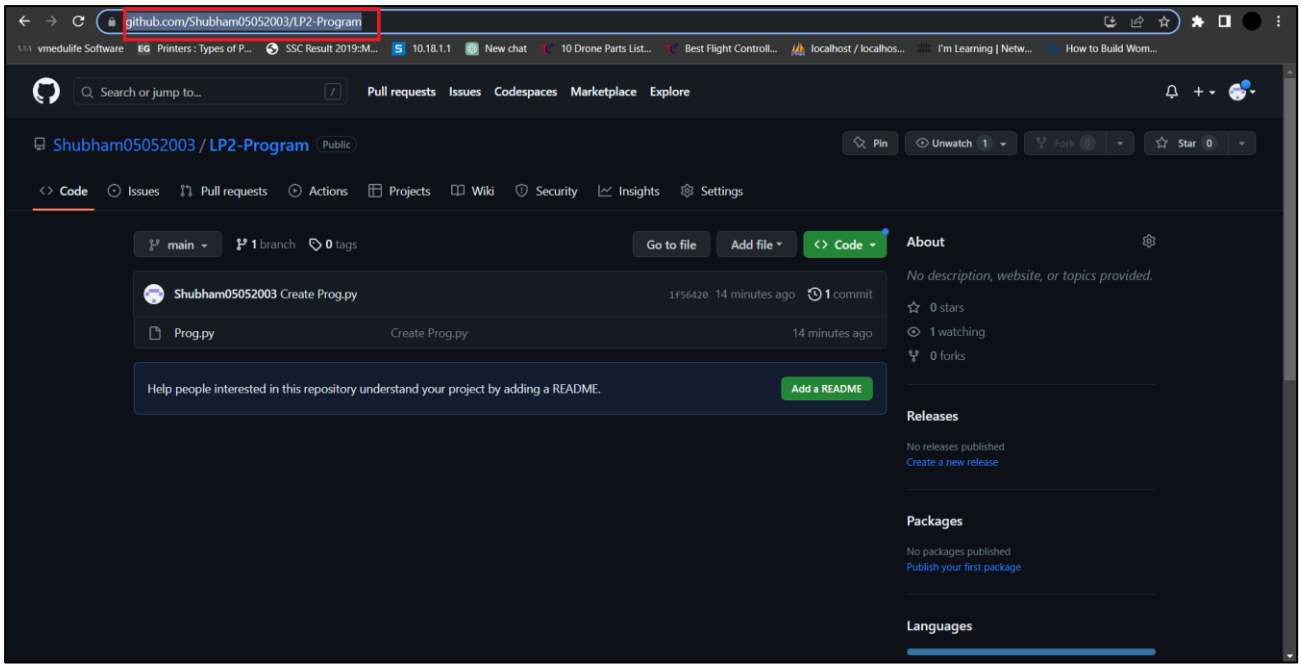
➢ **Again, click on ”Commit changes”:**



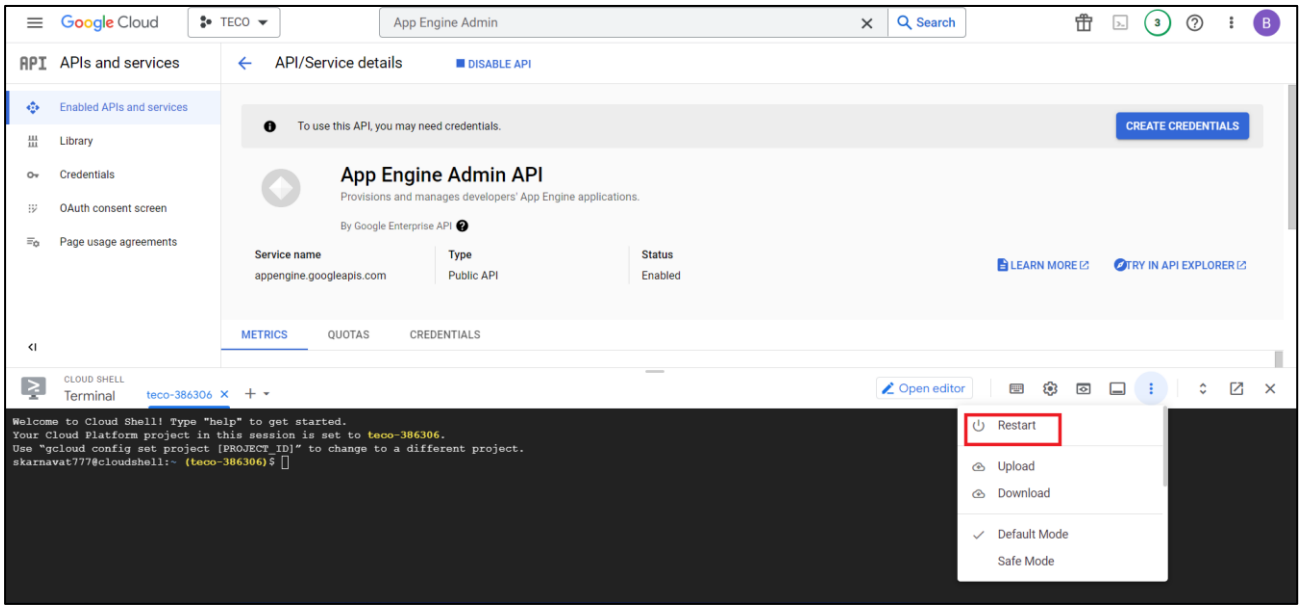
➢ **Go to the code section at the top left corner:**



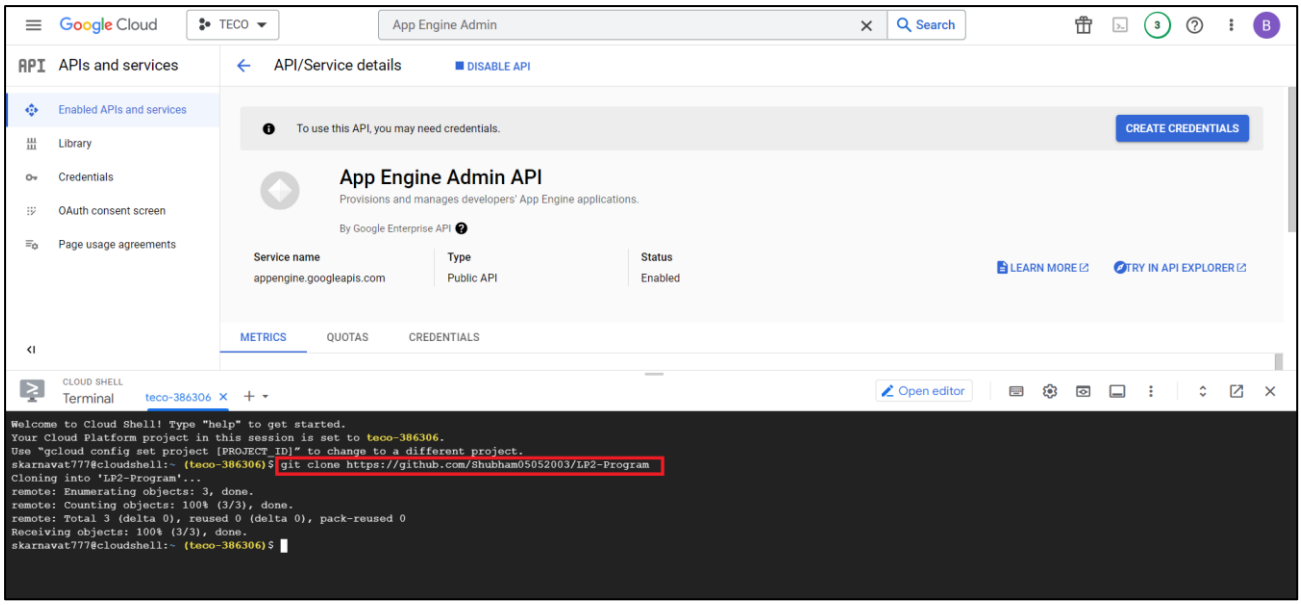
➢ **Copy the URL from the address bar:**



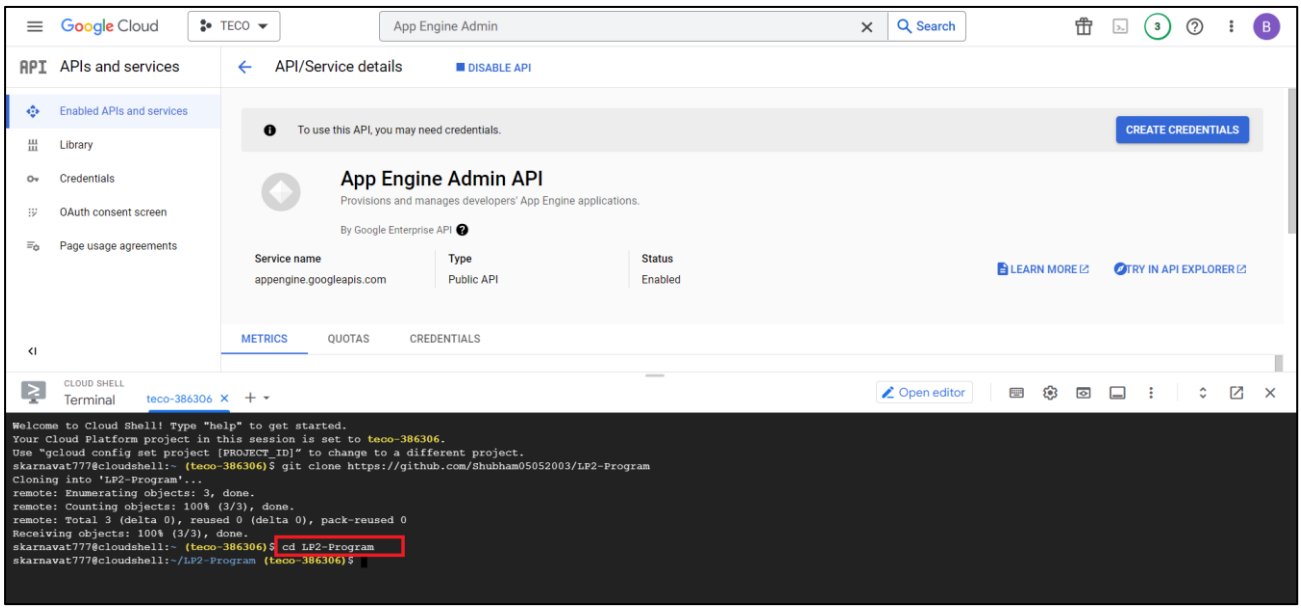
➢ **Again, go to the Google Cloud Platform and if any problem occurs in a shell then "Restart" it:**



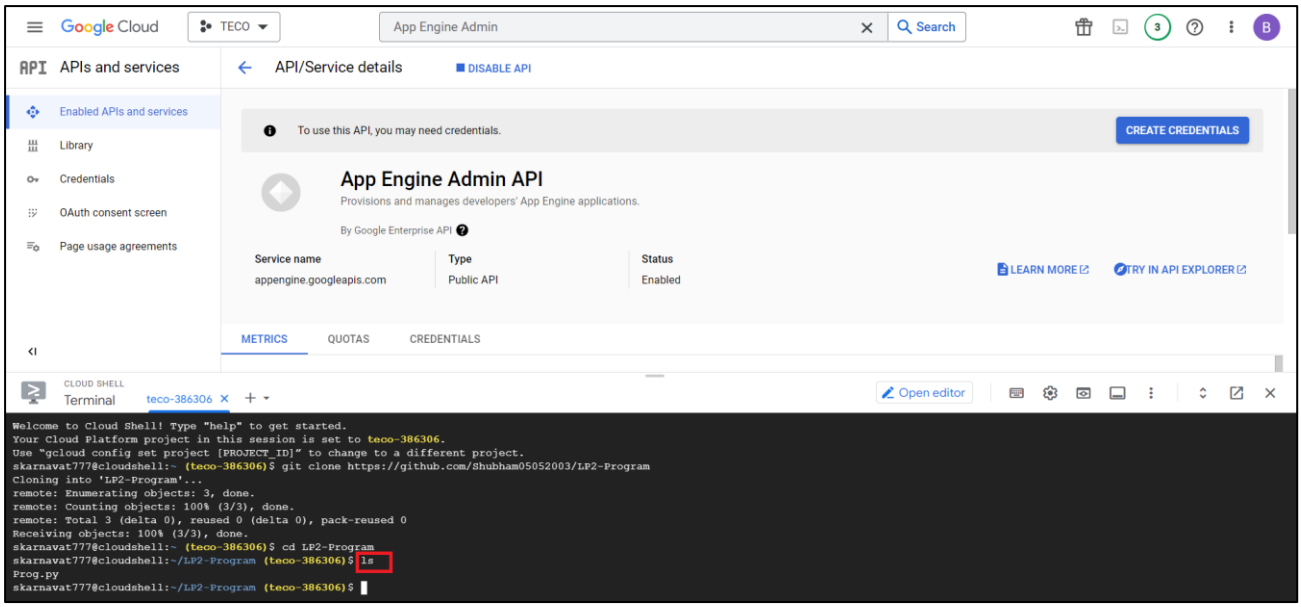
➢ **Type the “git clone” command and paste the URL:**



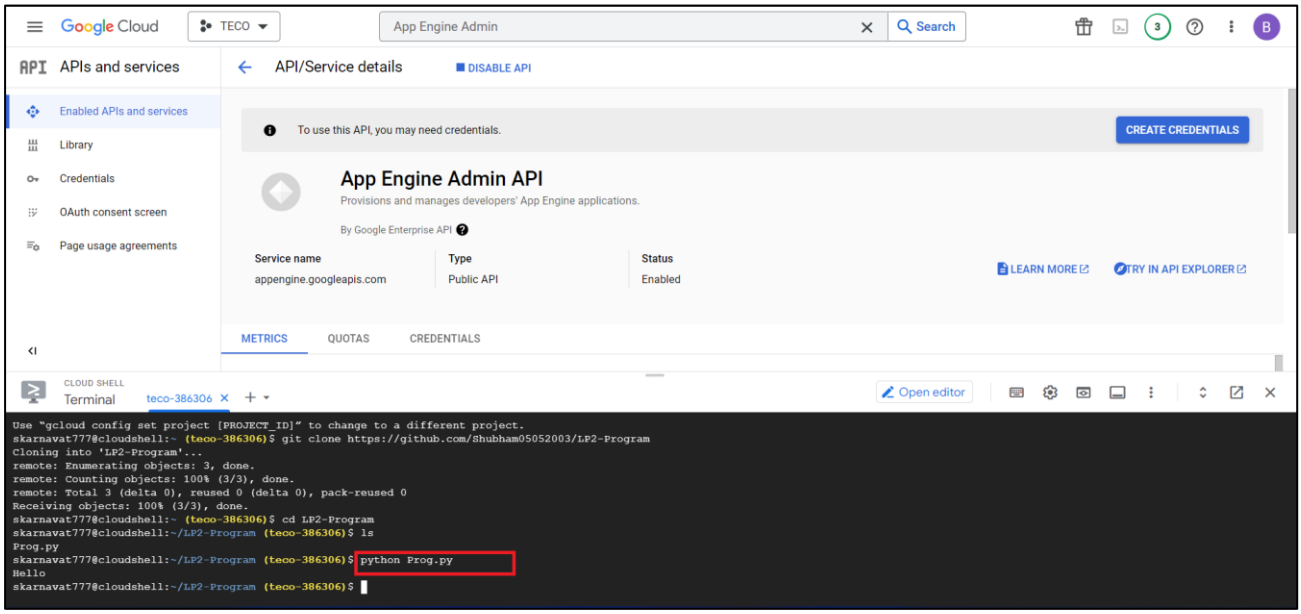
➢ **Change the directory with the help of the “cd” command and enter the repository name i.e “LP2-Program”:**



➢ **Enter the command “ls” to get the file name:**



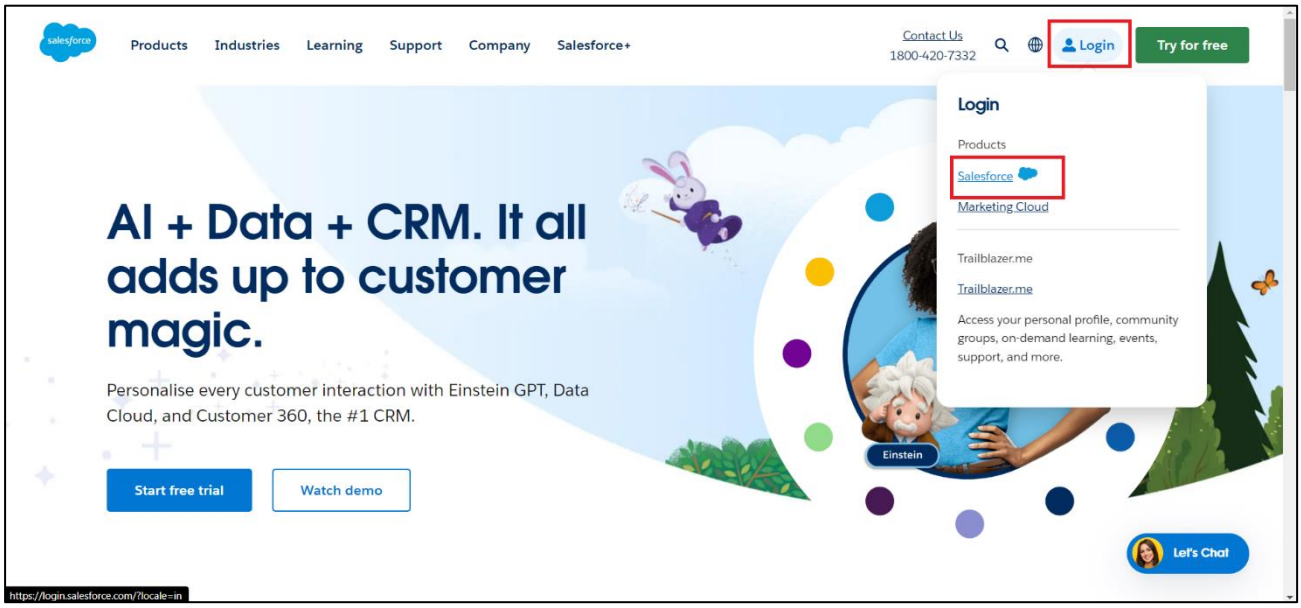
➢ **Enter the command “python” and the “file name” to get output i.e python Prog.py:**



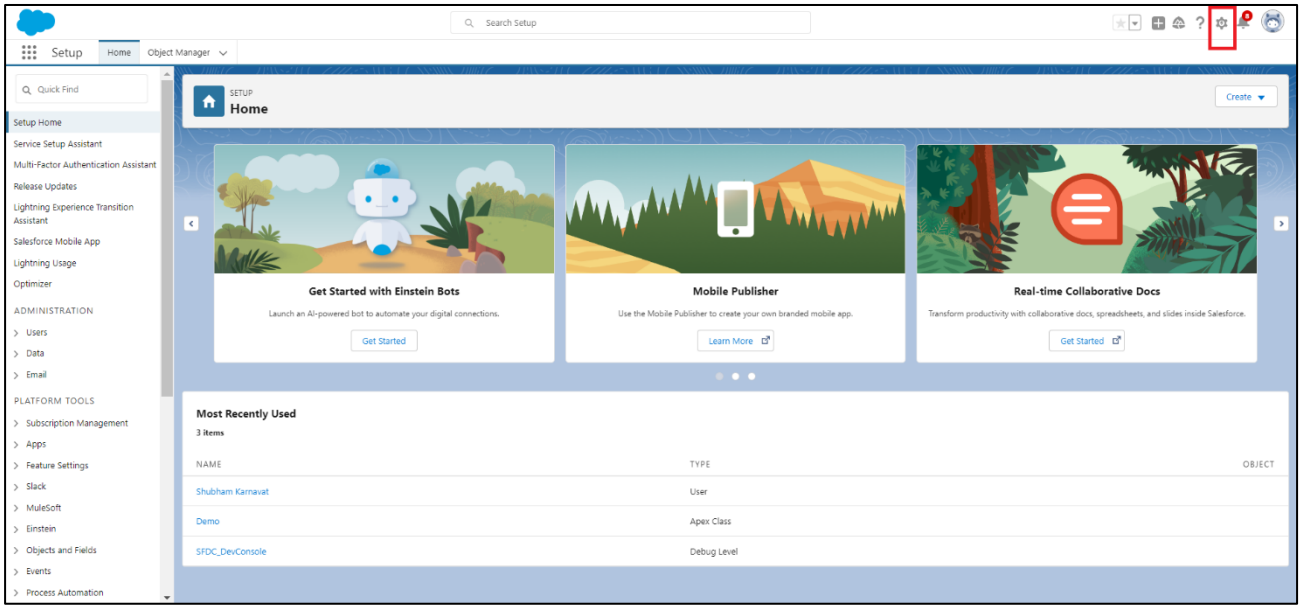
Practical Number :9

Title: Creating an application in SalesForce.com using Apex programming language

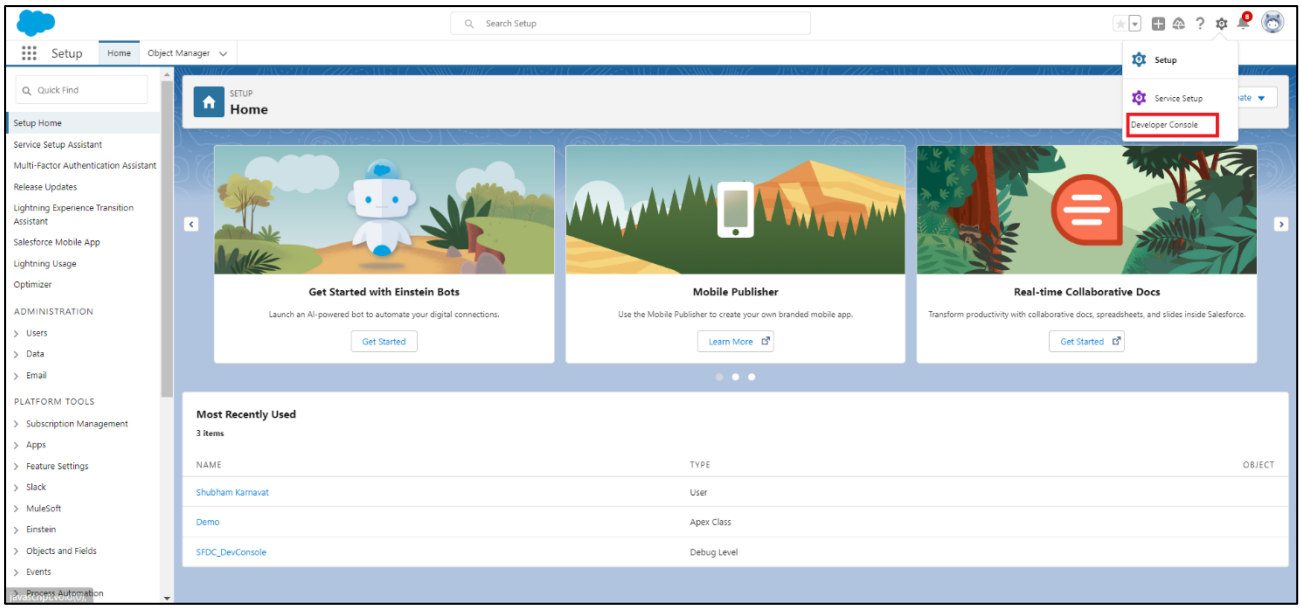
- **Go to SalesForces.com (https://www.salesforce.com/in/) & Login using the credentials:**



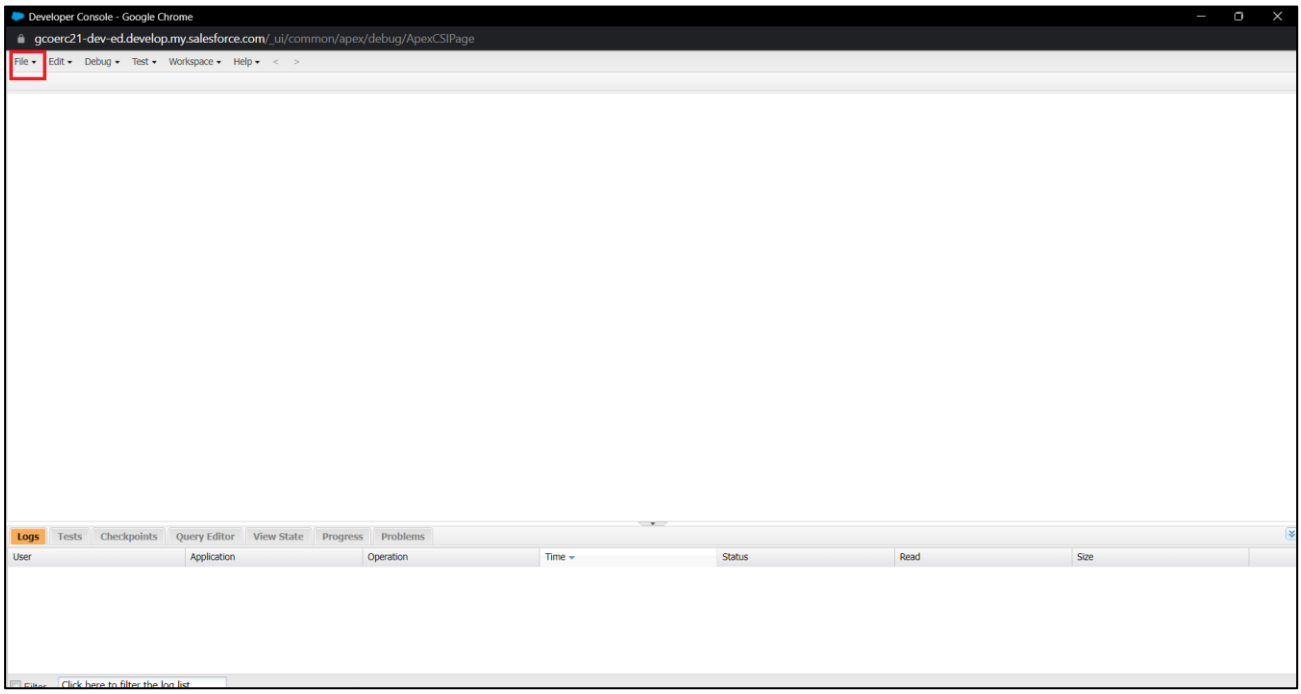
- **Click on "Setting Icon" at the top right corner:**



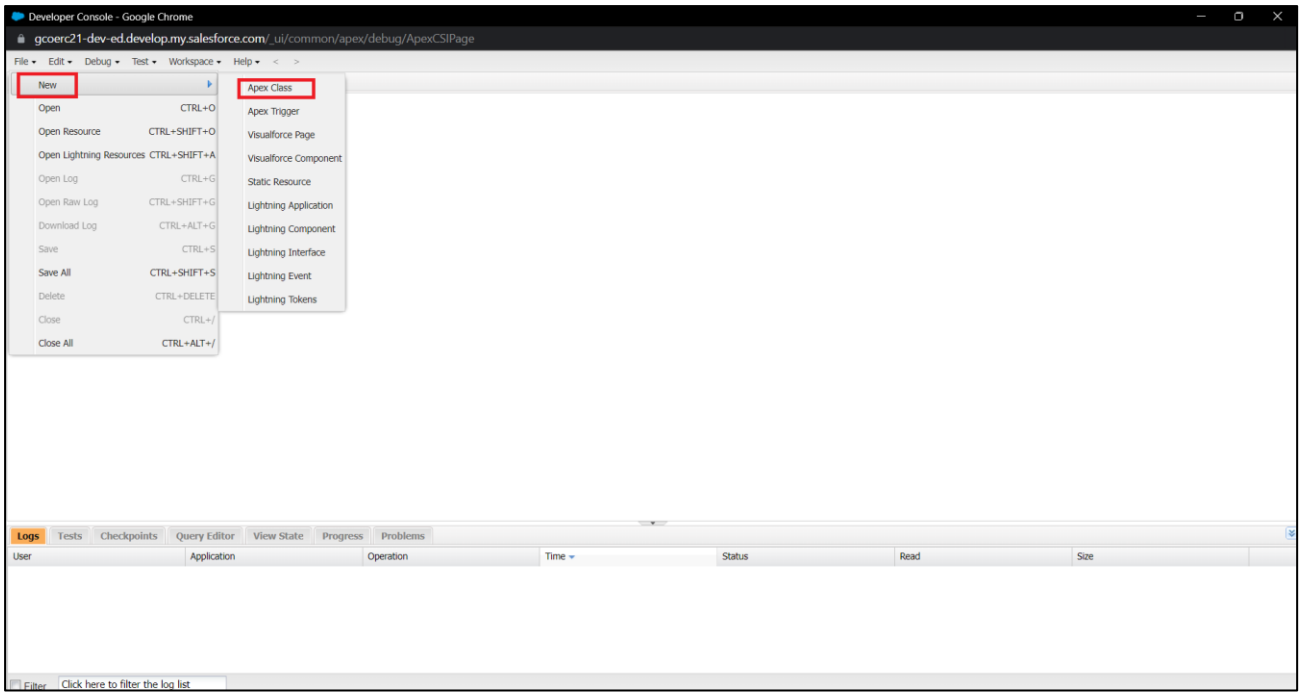
➢ **Select “Developer Console” from the list:**



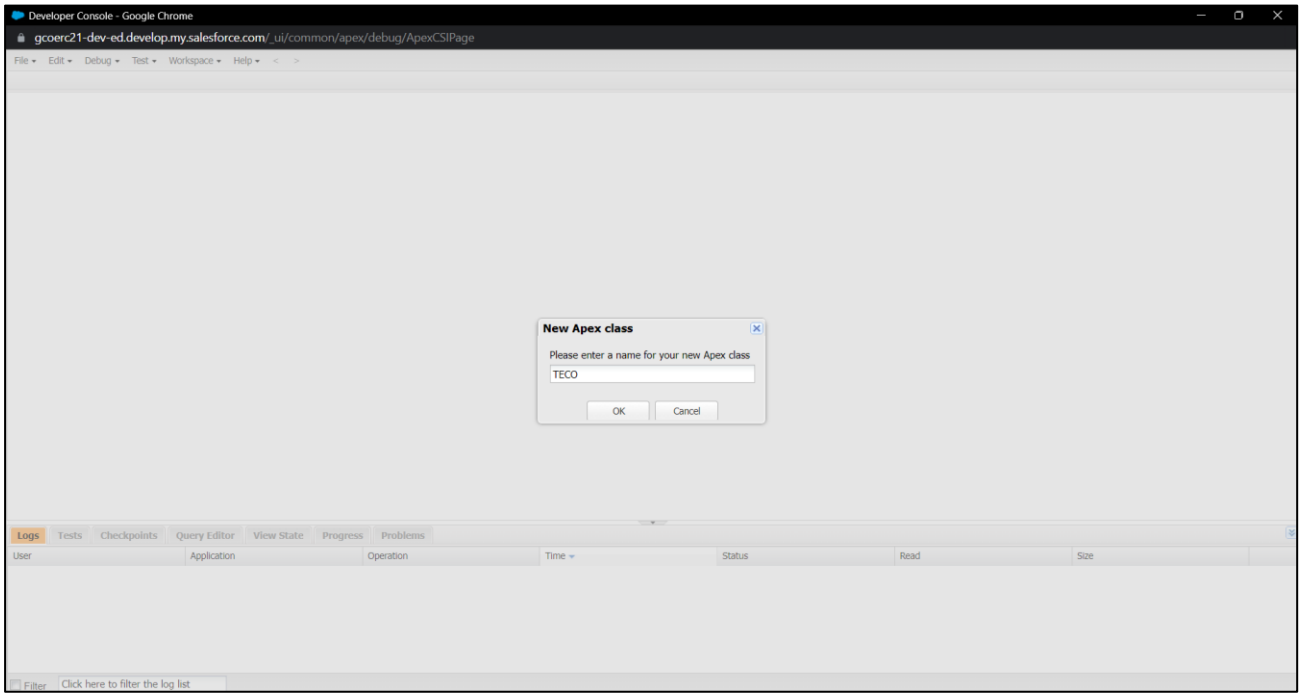
➢ **Go to “File”:**



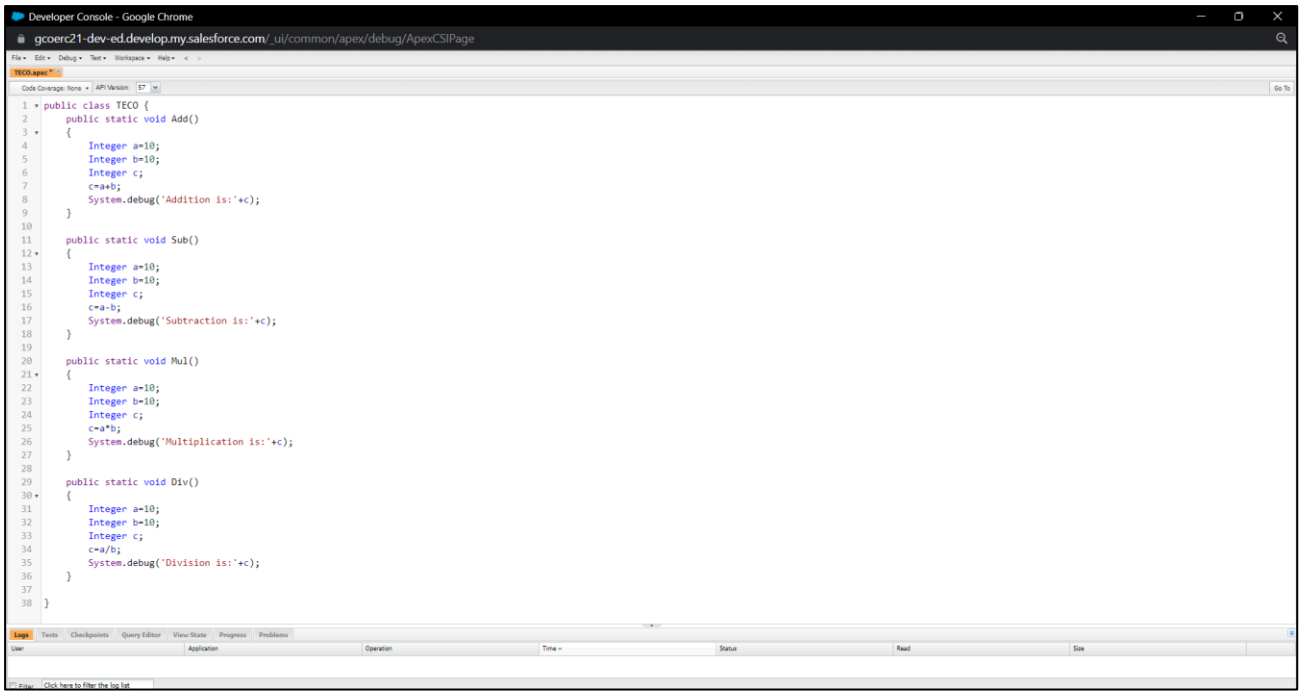
➢ **Select New => Apex Class:**



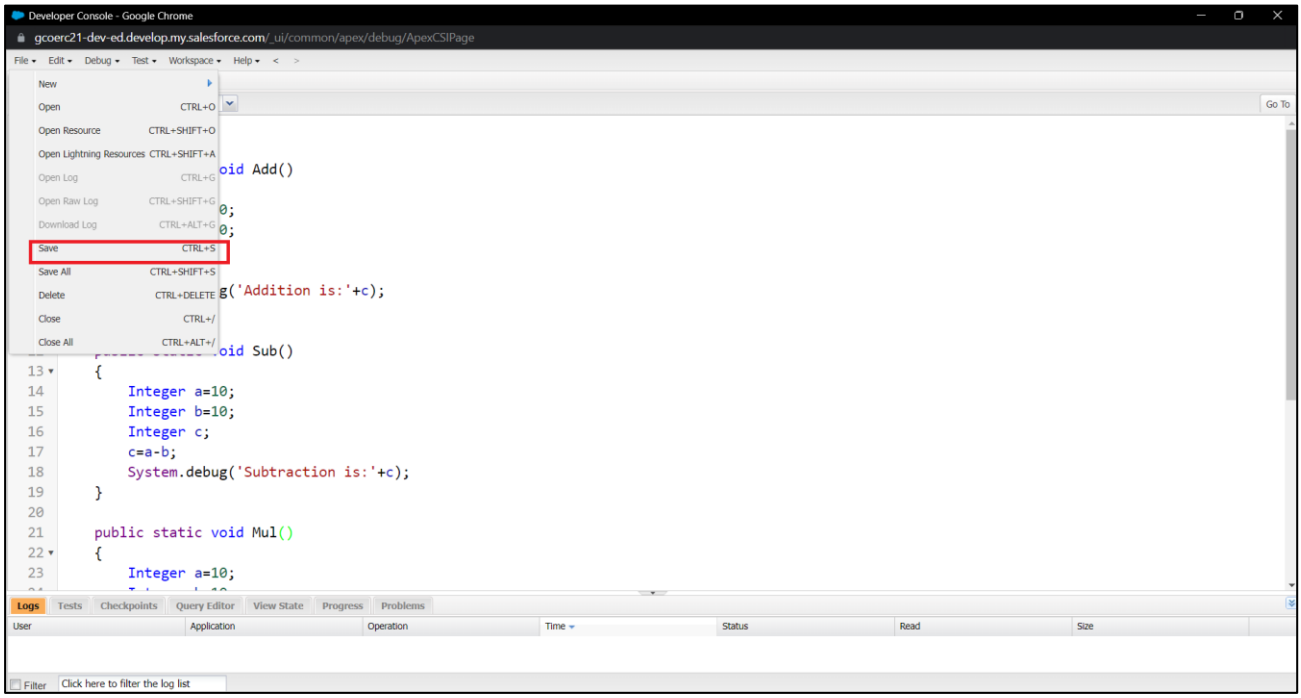
➢ **Enter the name of the Apex class:**



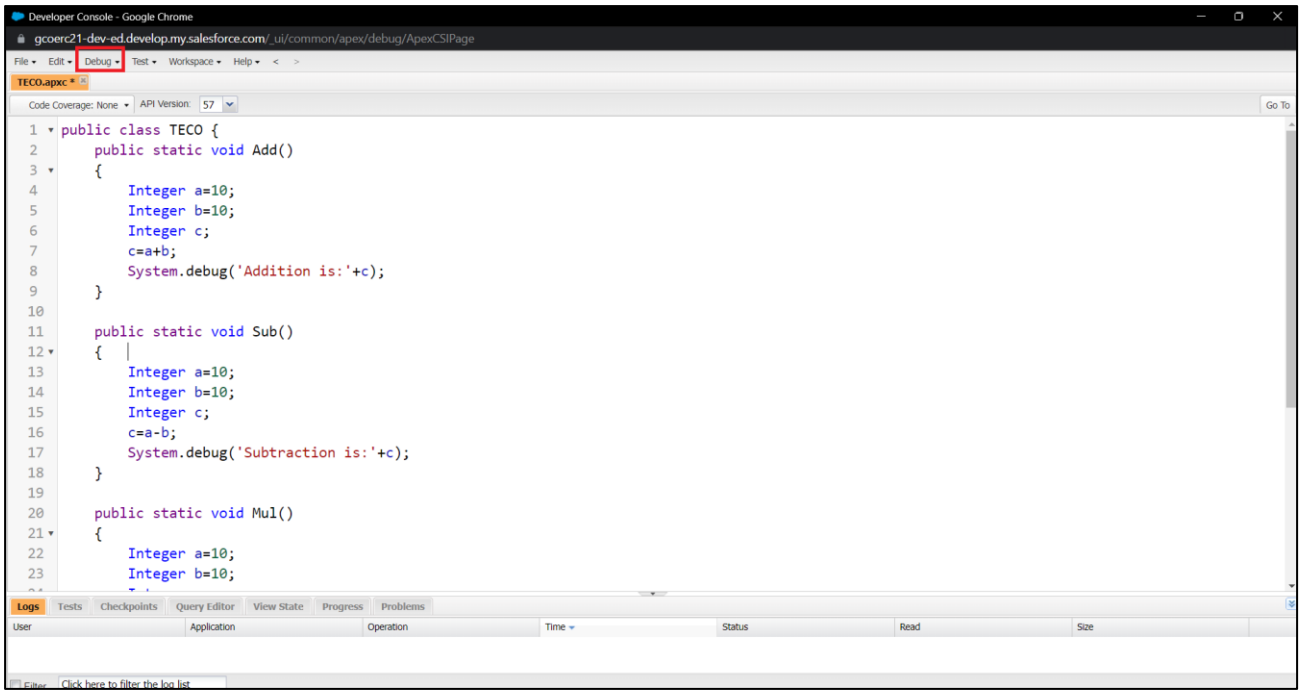
- **Write code in the console:**



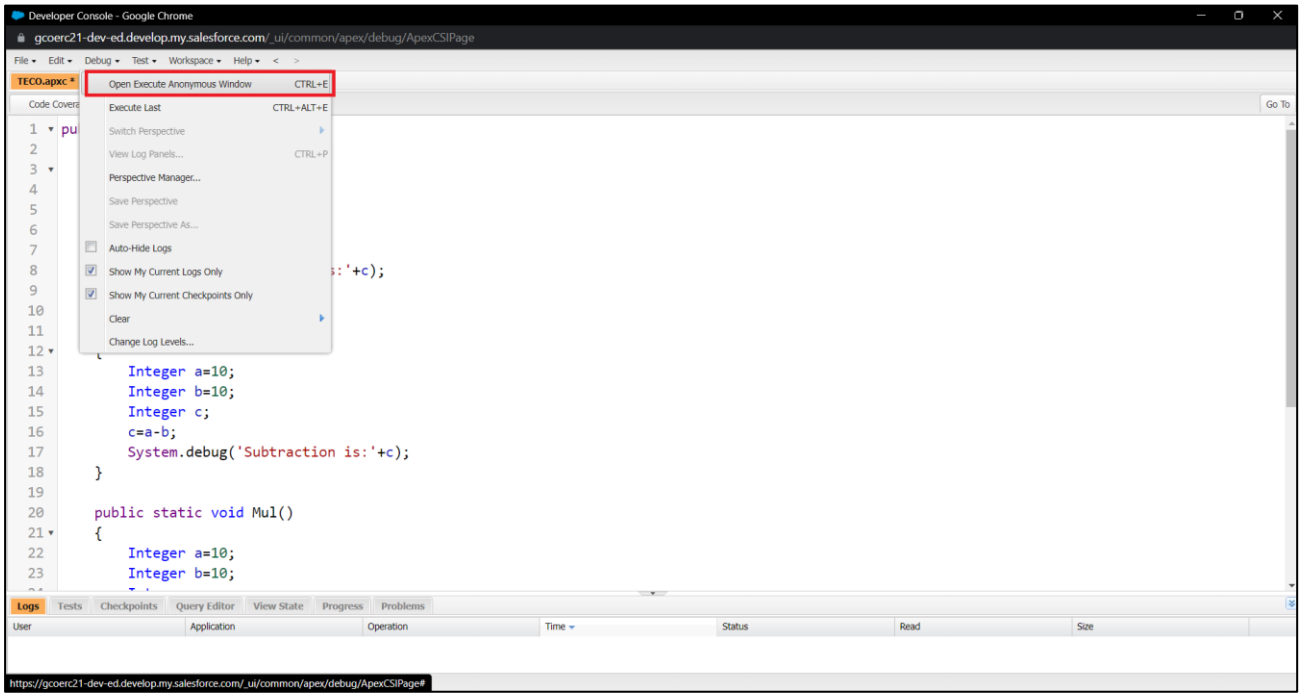
- **Save the code before executing:**



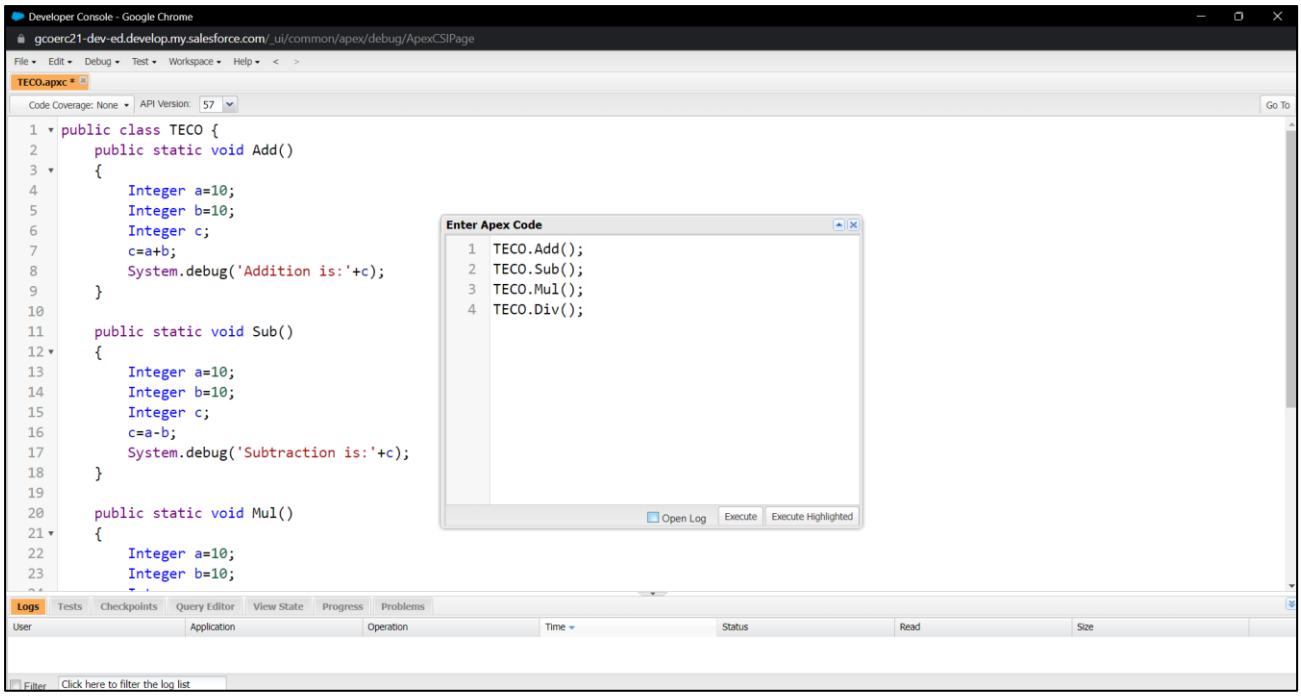
➢ **Click on “Debug”:**



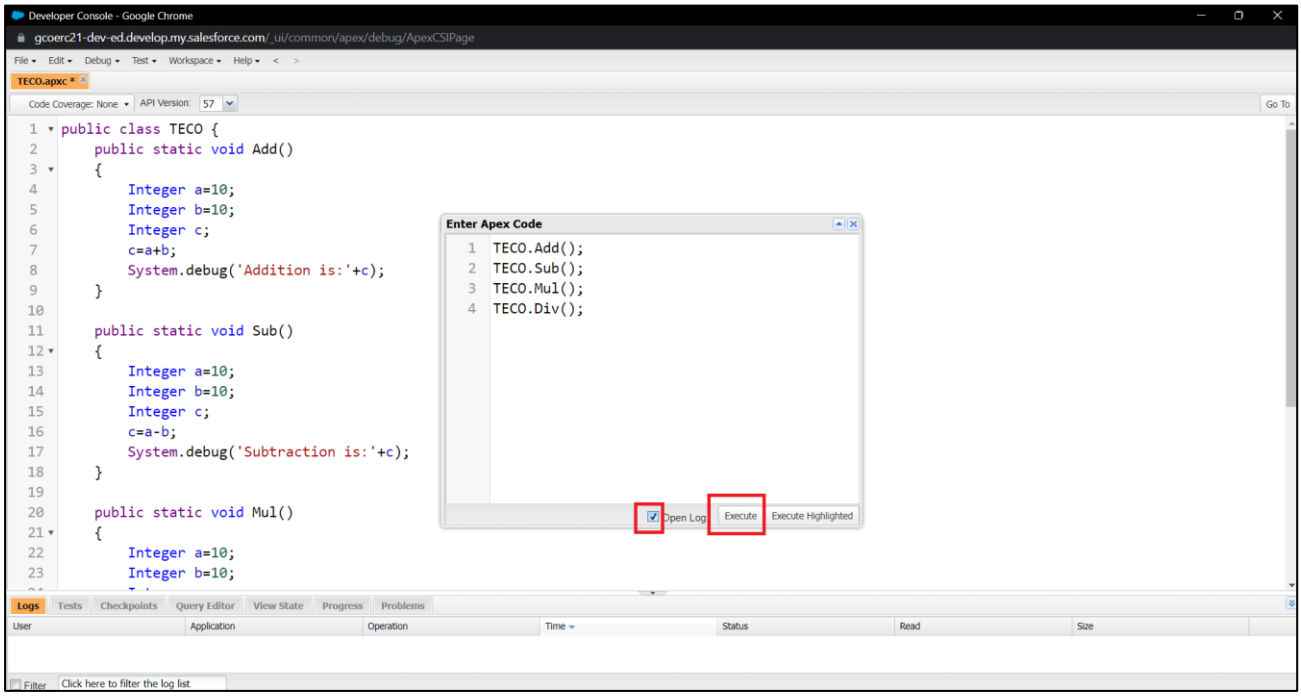
➢ **Select “Open Execute Anonymous Window” from the list:**



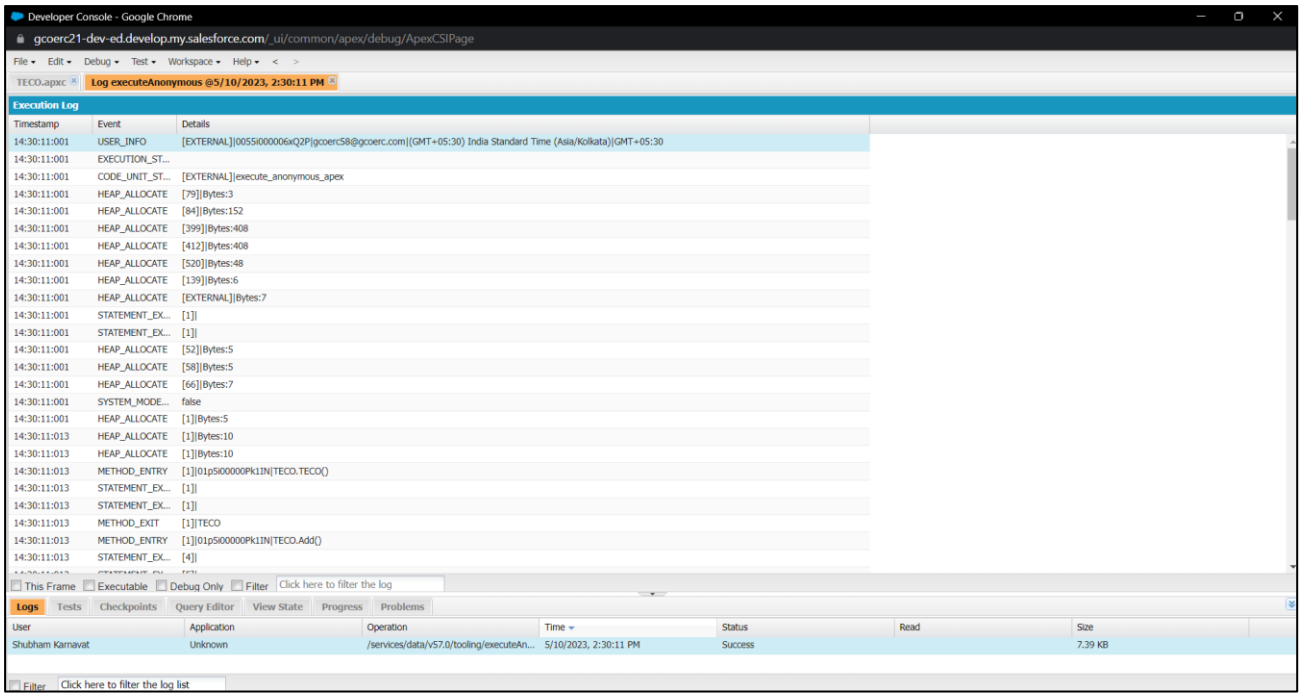
➢ **Call the function with the help of the class name:**



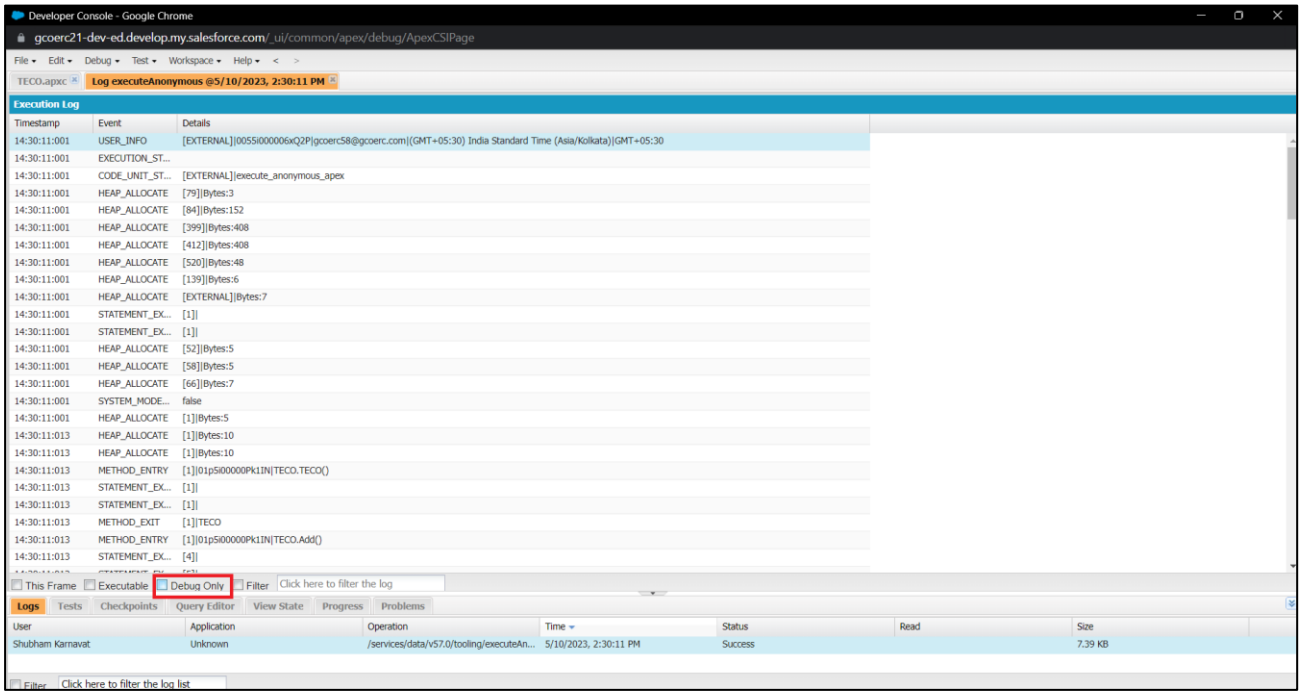
➢ **Tick on "Open log" and click on "Execute":**



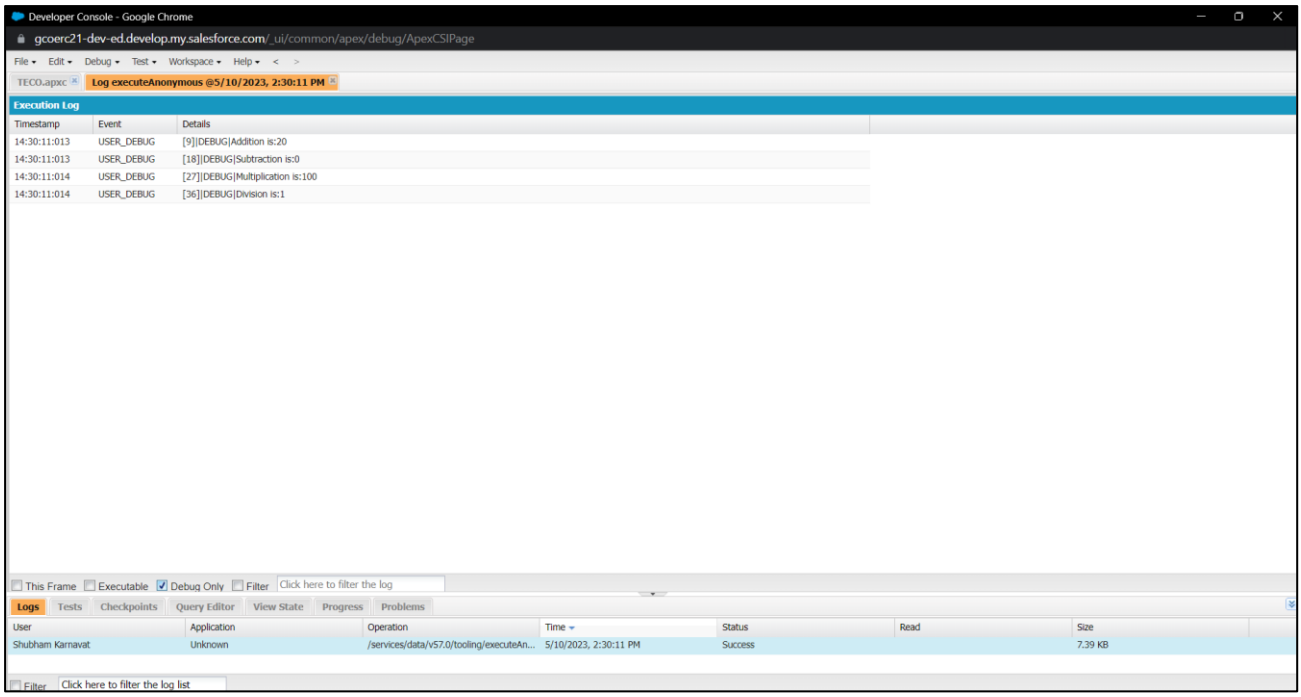
- **Log Window will appear:**



- **Tick on "Debug Only":**



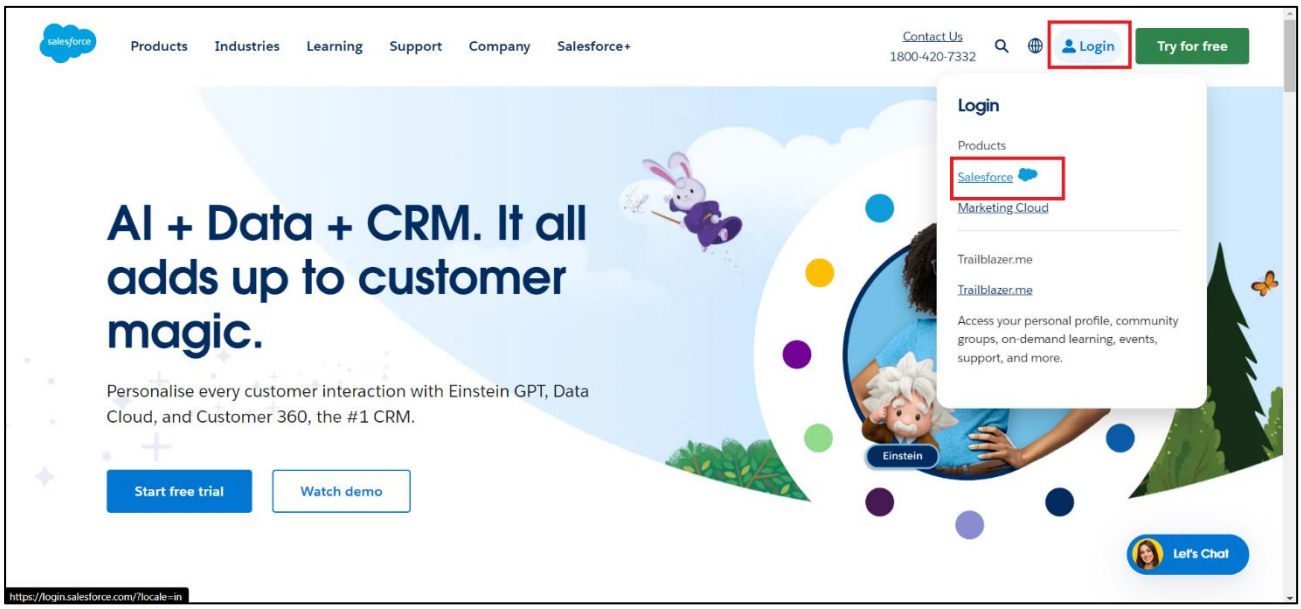
➢ Final Output:



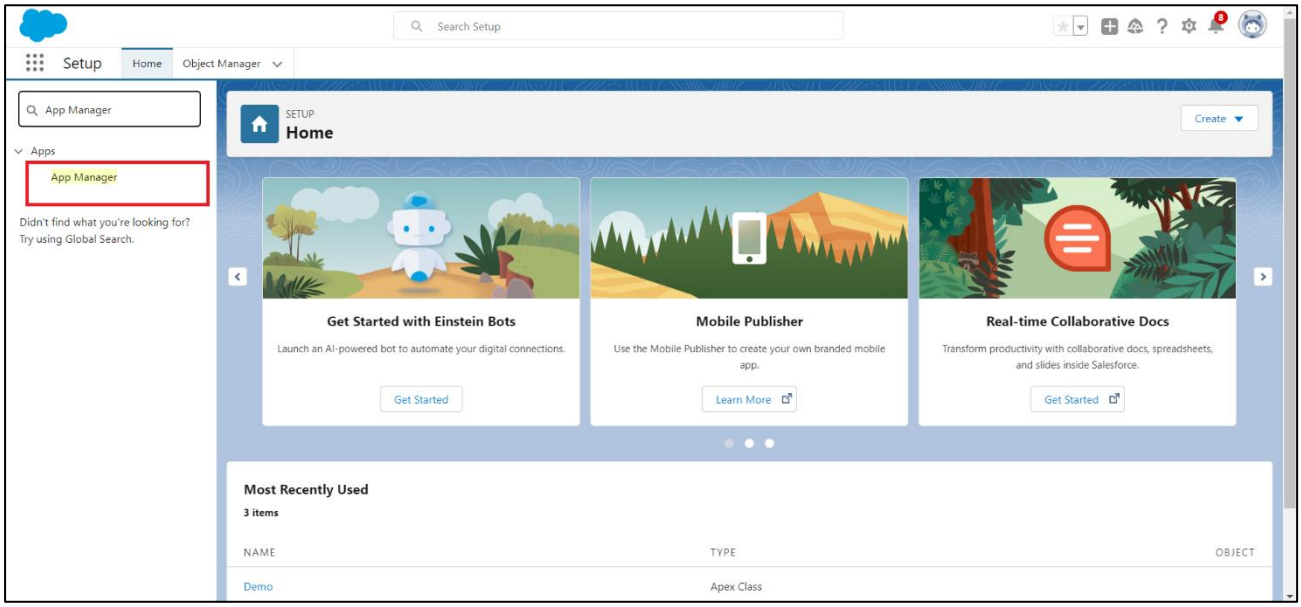
Practical Number :10

Title: Design and develop a custom application using the SalesForce cloud.

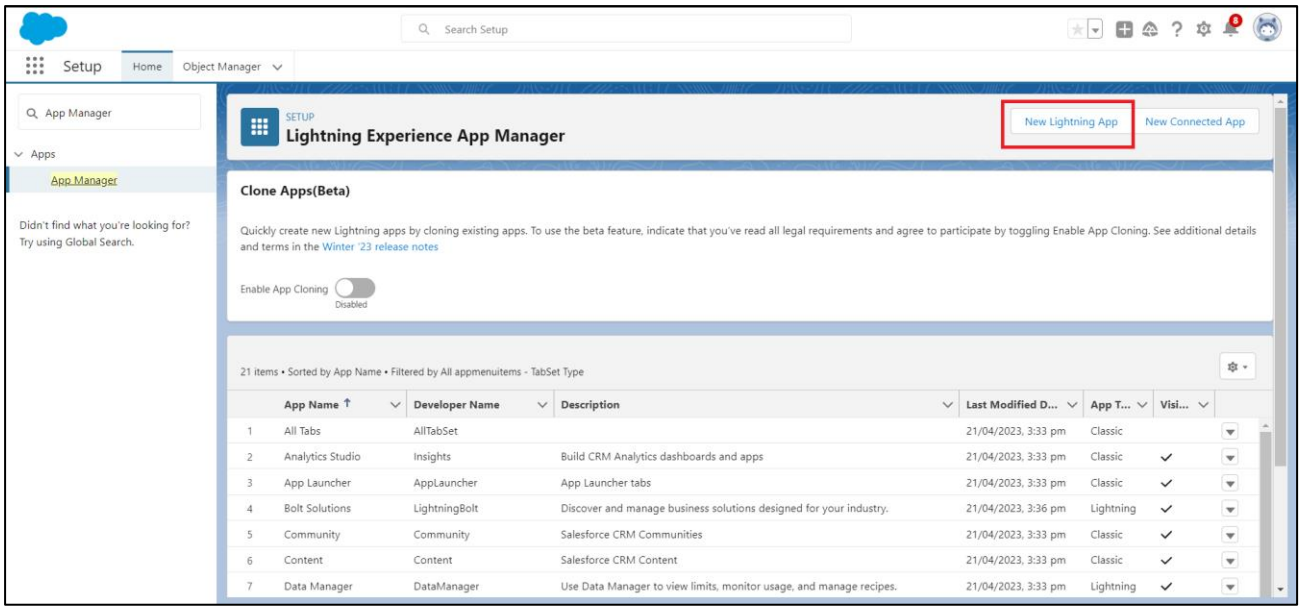
- **Go to SalesForces.com (https://www.salesforce.com/in/) & Login using the credentials:**



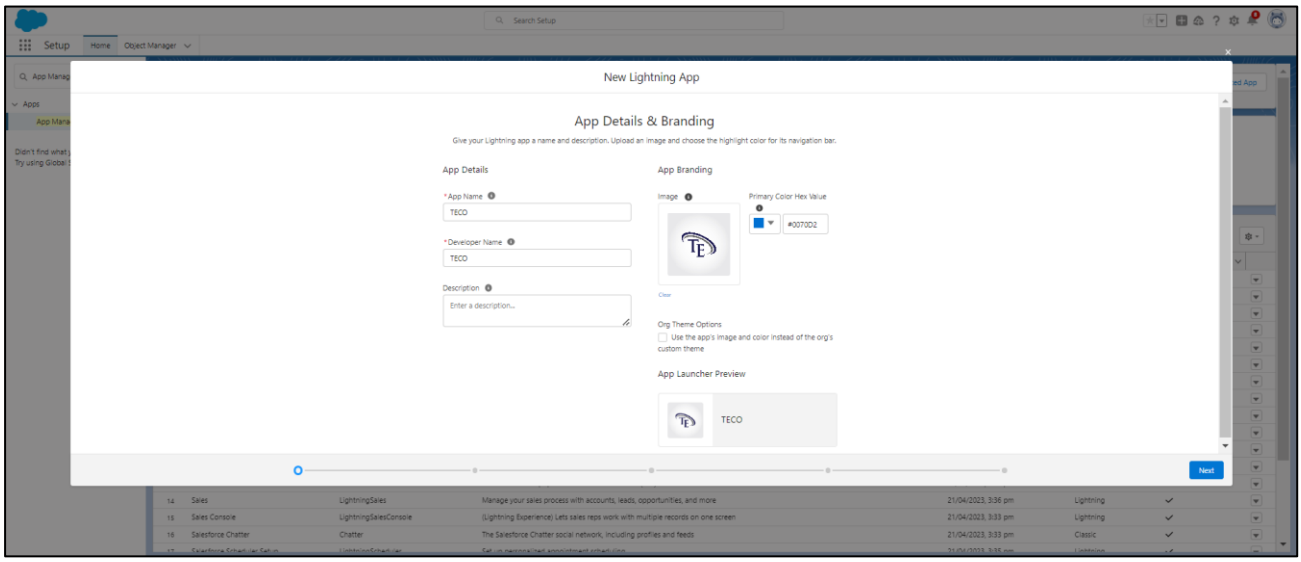
- **Search "App Manager" in the search box and select it from the list:**



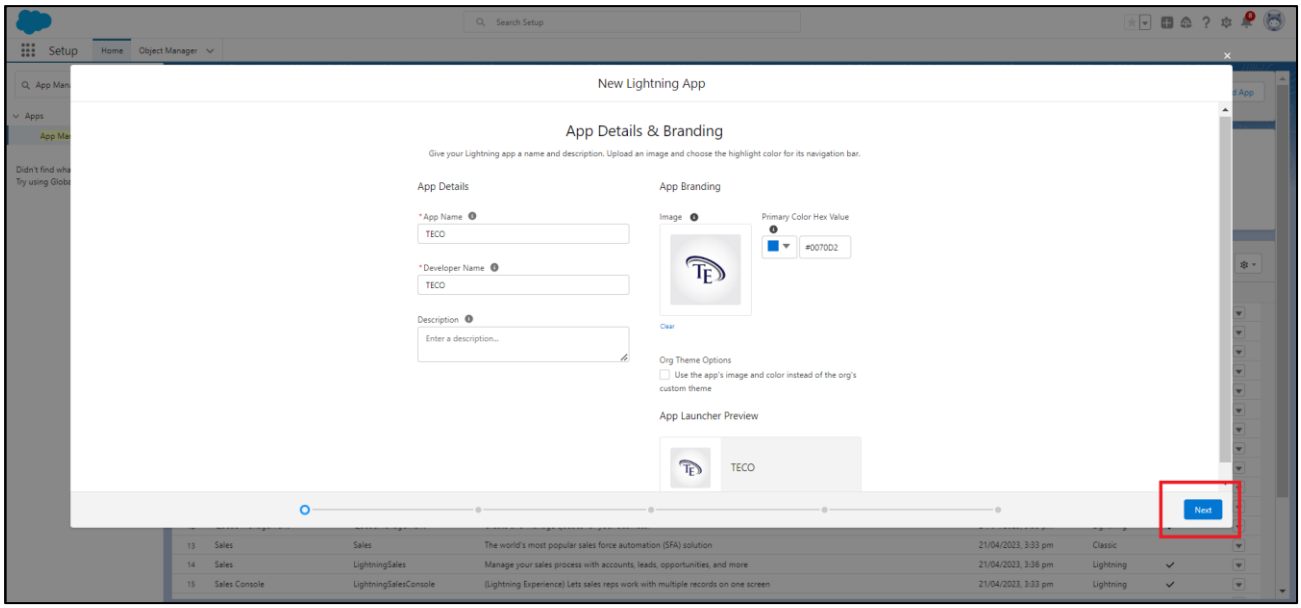
➢ **Click on “New Lightning App”:**



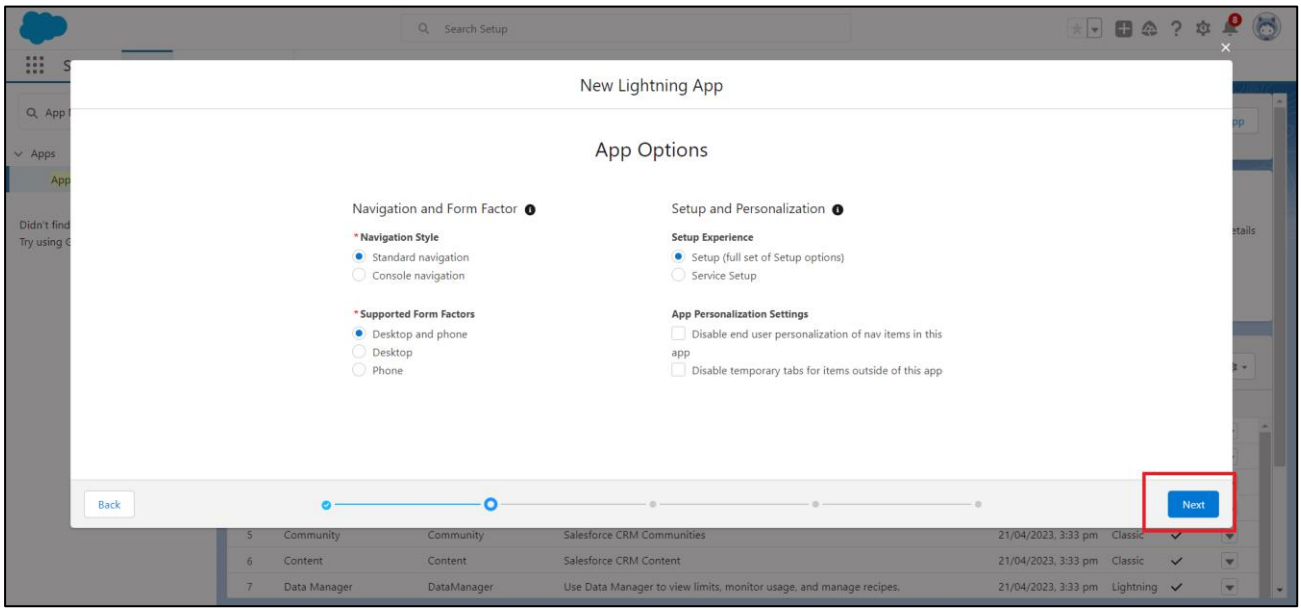
➢ **Enter the App name and Developer name:**



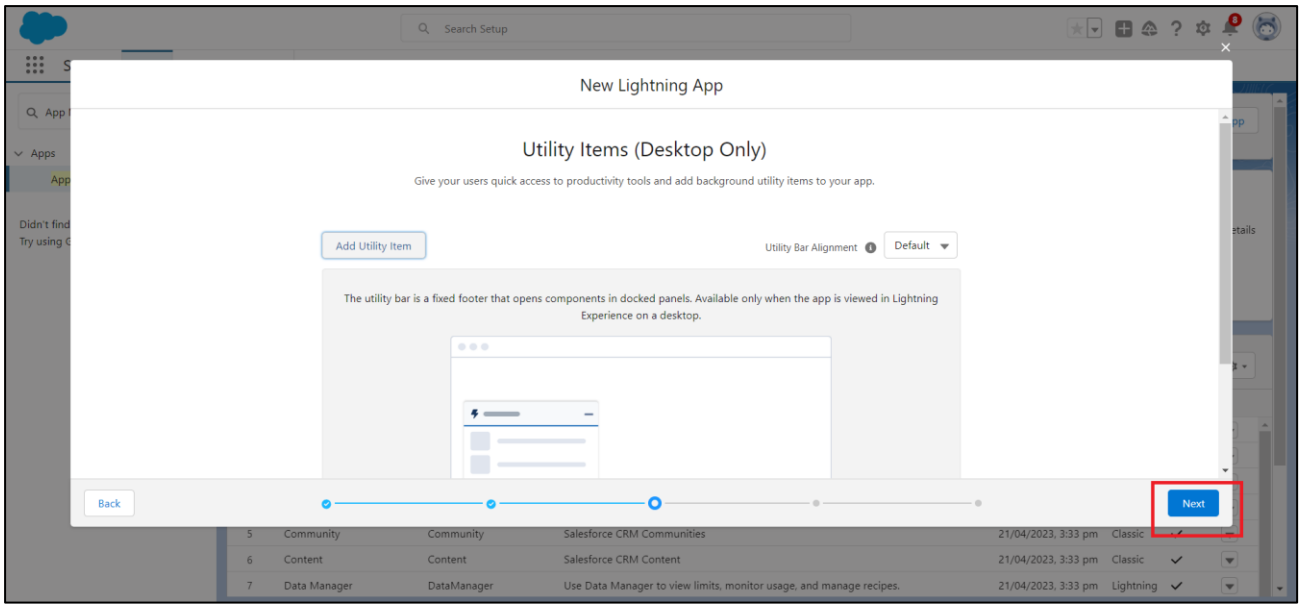
➢ **Click on “Next”:**



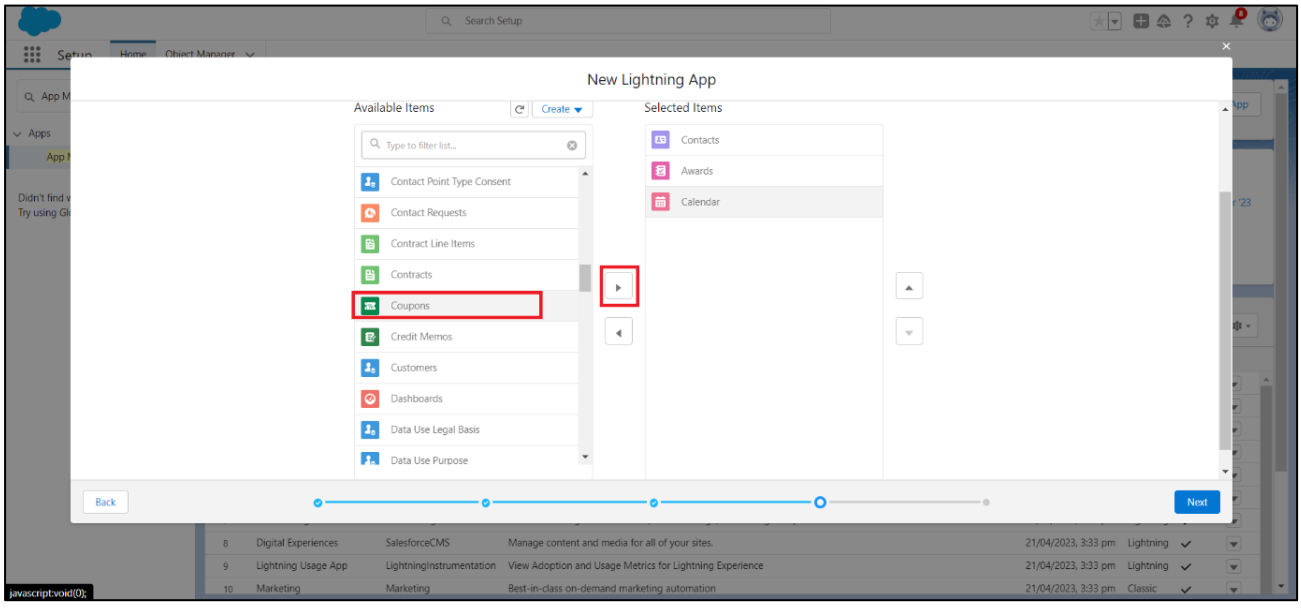
➢ **Click on “Next”:**



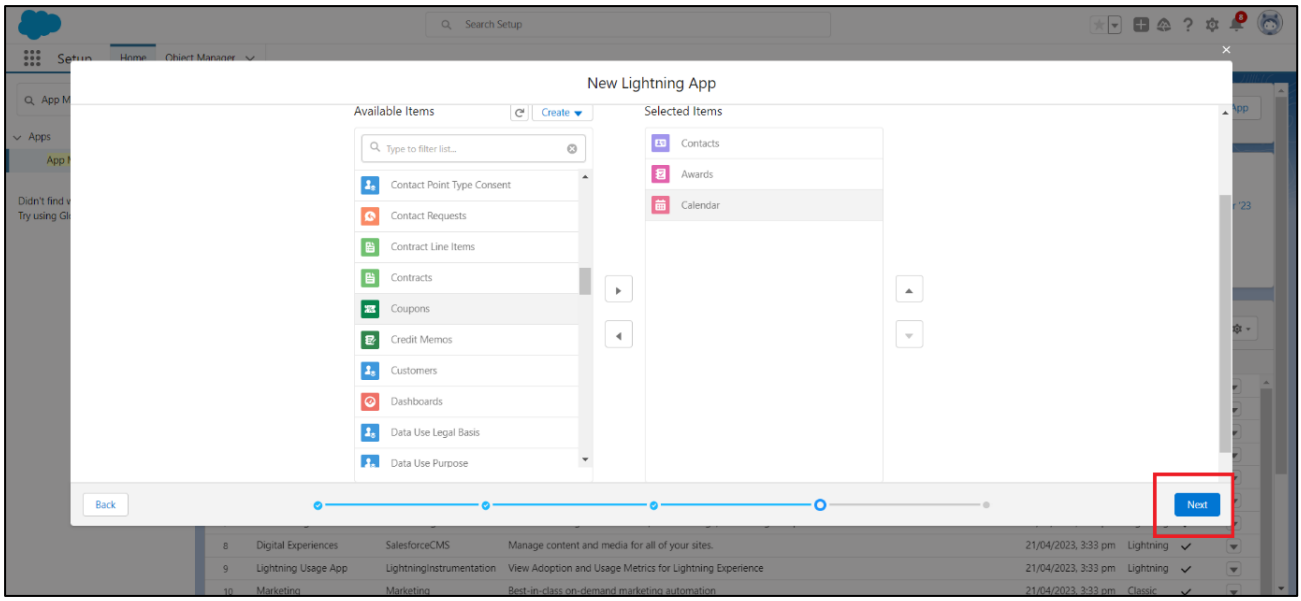
➢ **Click on “Next”:**



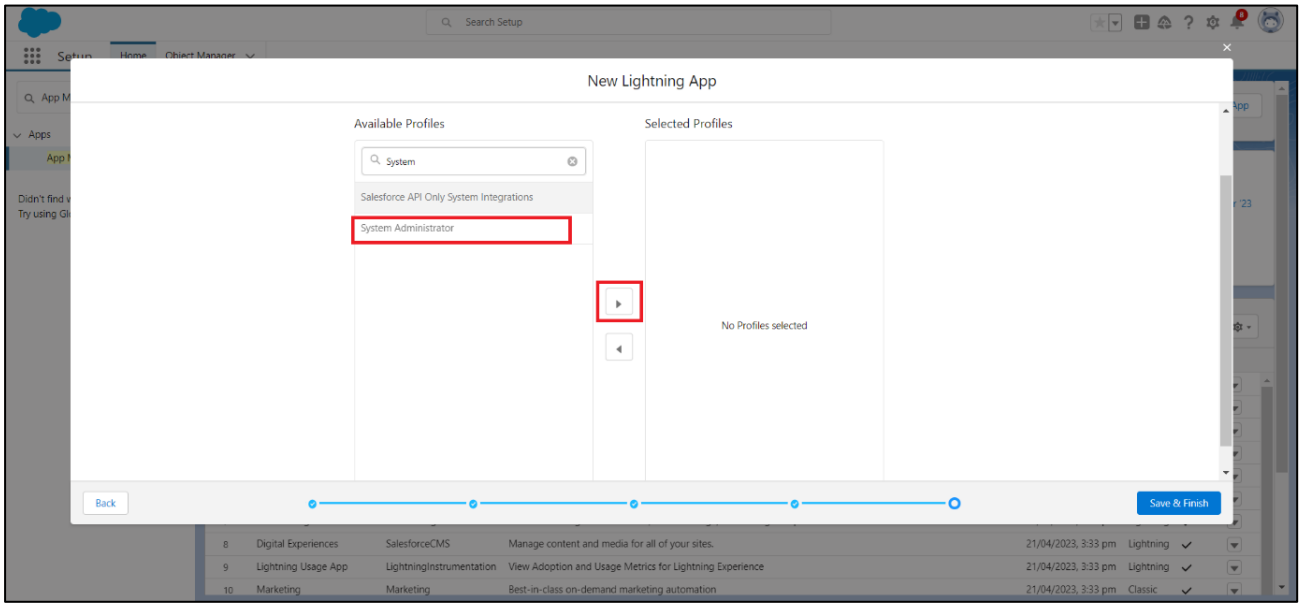
➢ **Select items from the given item list:**



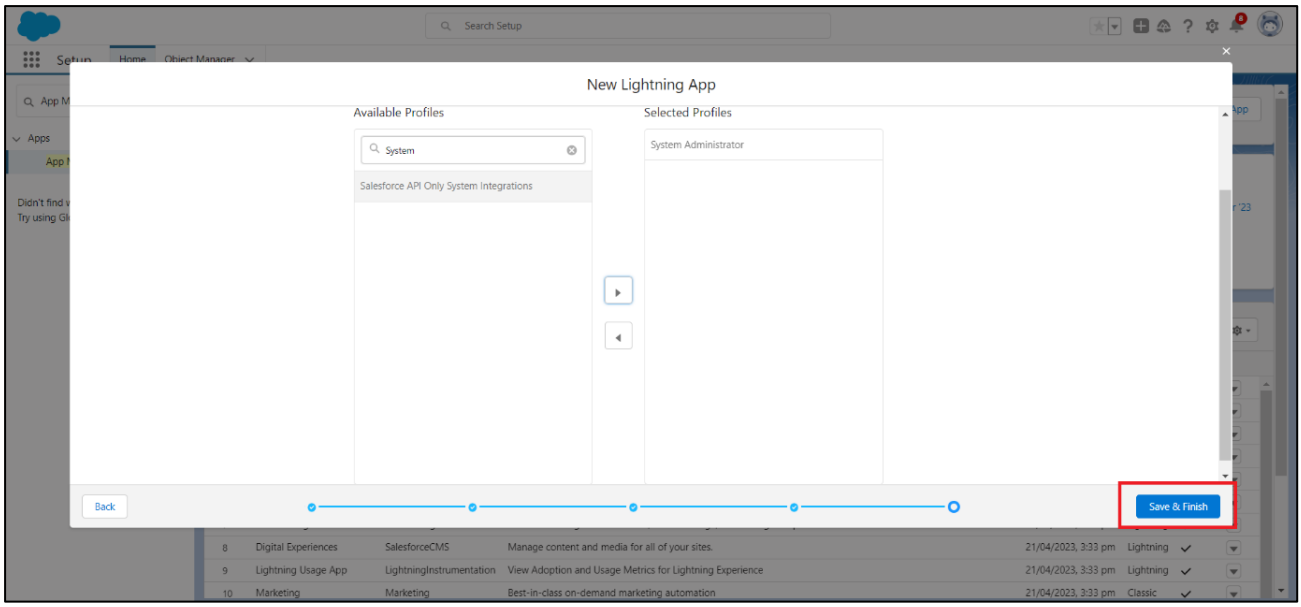
➢ **Click on “Next”:**



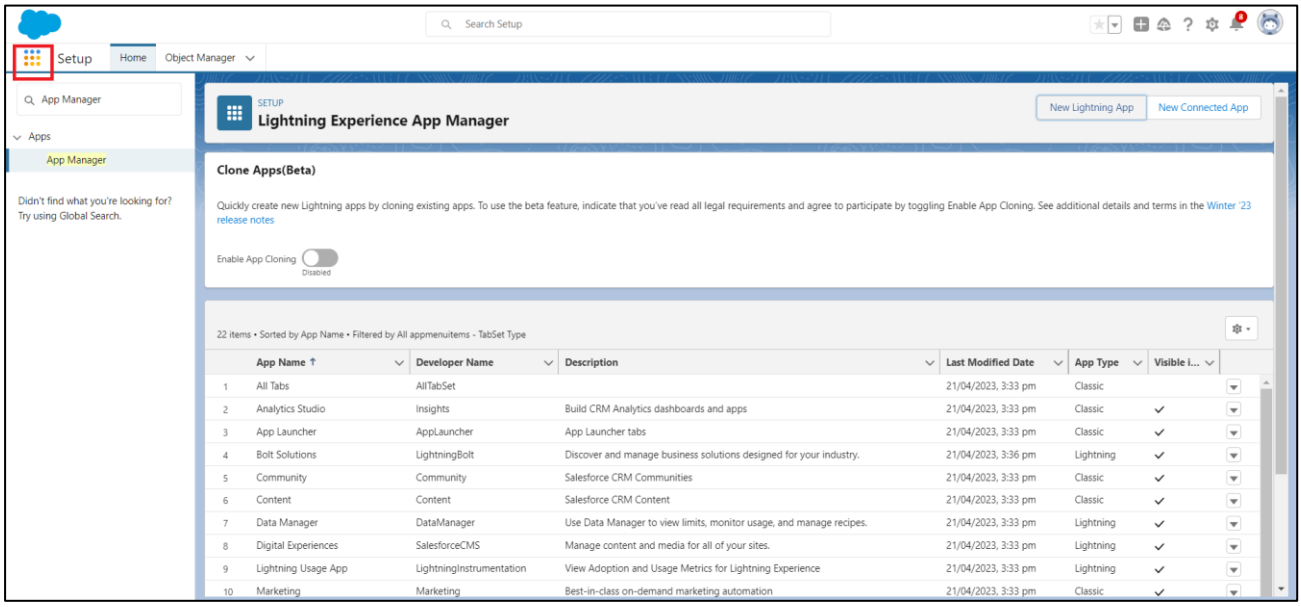
➢ **Search “System Administrator” and select it from the list:**



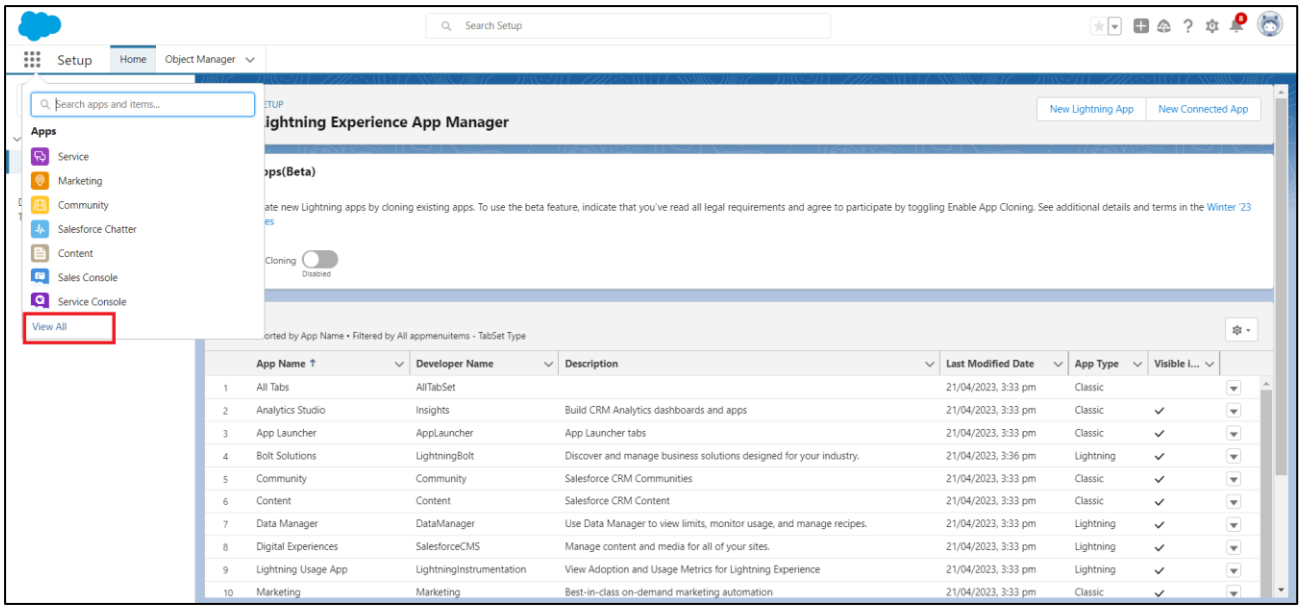
➢ **Click on “Save & Finish”:**



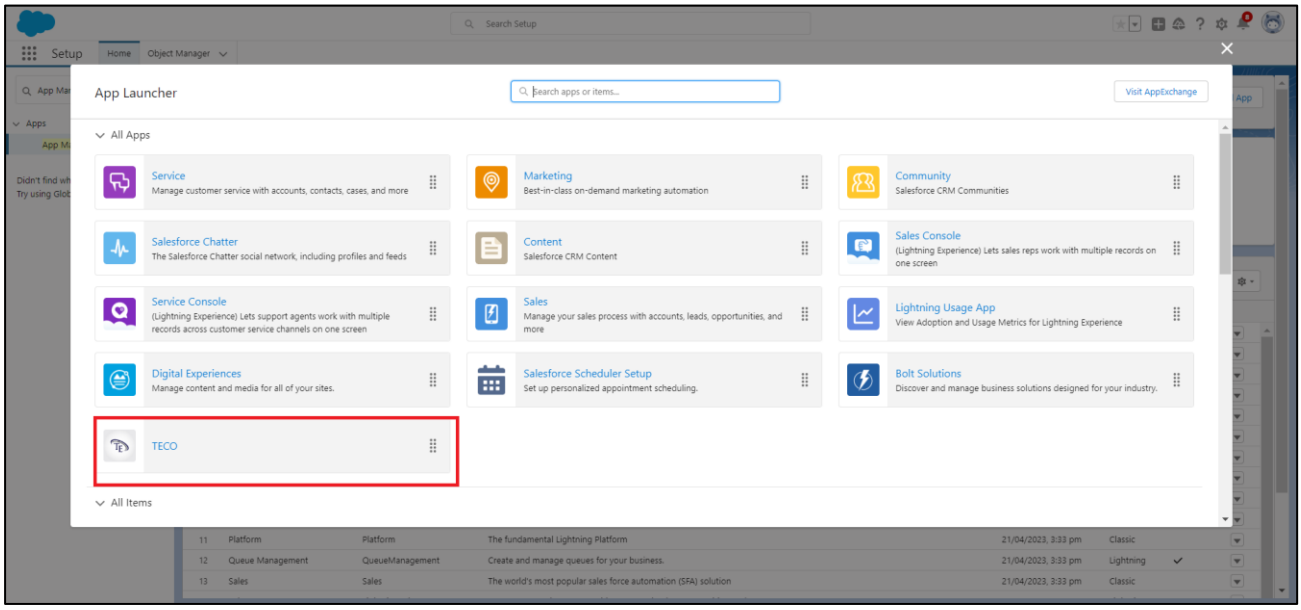
➢ **Click on “App launcher Icon” at the top left side of the window:**



➢ **Click on "View All":**

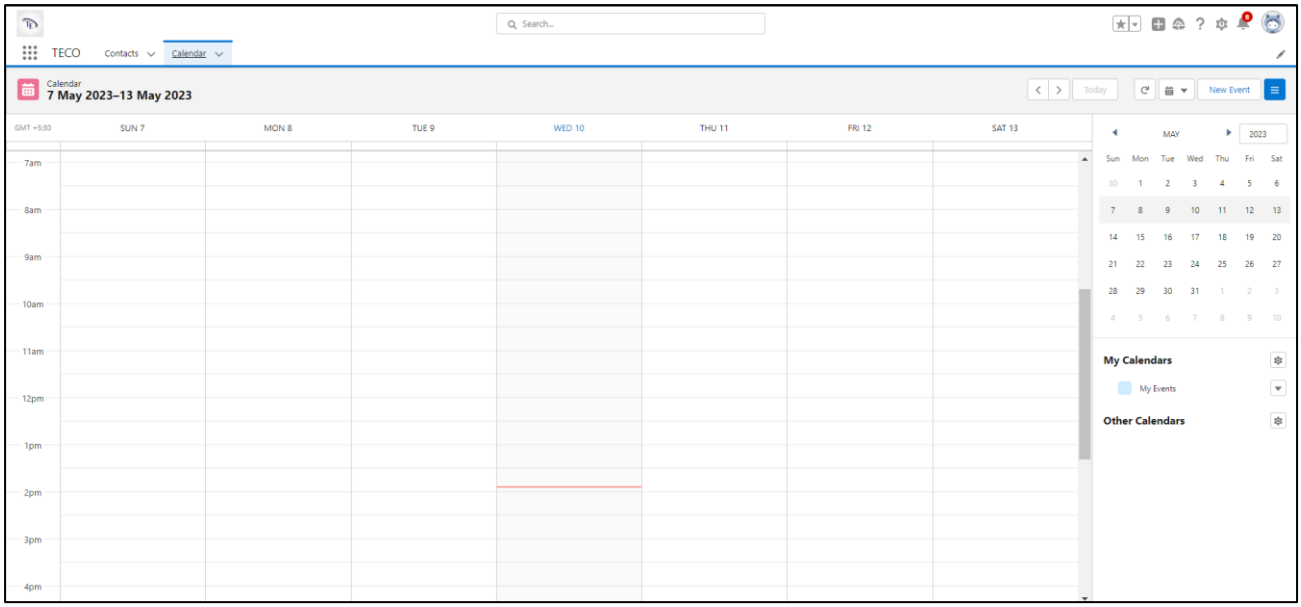


➢ **Select the app from the list of apps :**

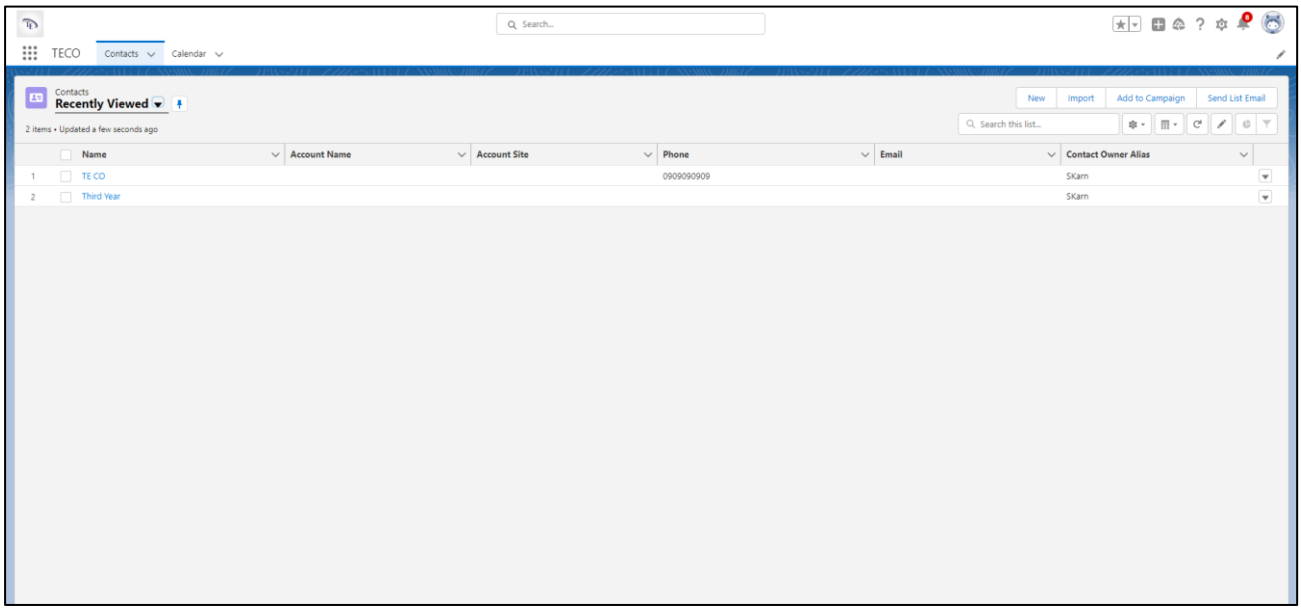


Explore the features of the app:

- **Calendar:**



- **Contacts:**



Practical No : 11

Practical Title: Setup your own cloud for Software as a Service (SaaS) over the existing LAN in your laboratory. In this assignment you have to write your own code for cloud controller using open-source technologies to implement with HDFS. Implement the basic operations may be like to divide the file in segments/blocks and upload/ download file on/from cloud in encrypted form.

Objectives:

- To set your own cloud for SaaS over existing LAN
- To implement the basic operations may be like to divide the file in segments/blocks

Hardware Requirements :

- Pentium IV with latest configuration

Software Requirements :

- Ubuntu 20.04, VMwareESXi cloud

Theory:

Here we are installing VMwareESXi cloud

- Host/NodeESXi installation:-
- **ESXiHardwareRequirements:-**

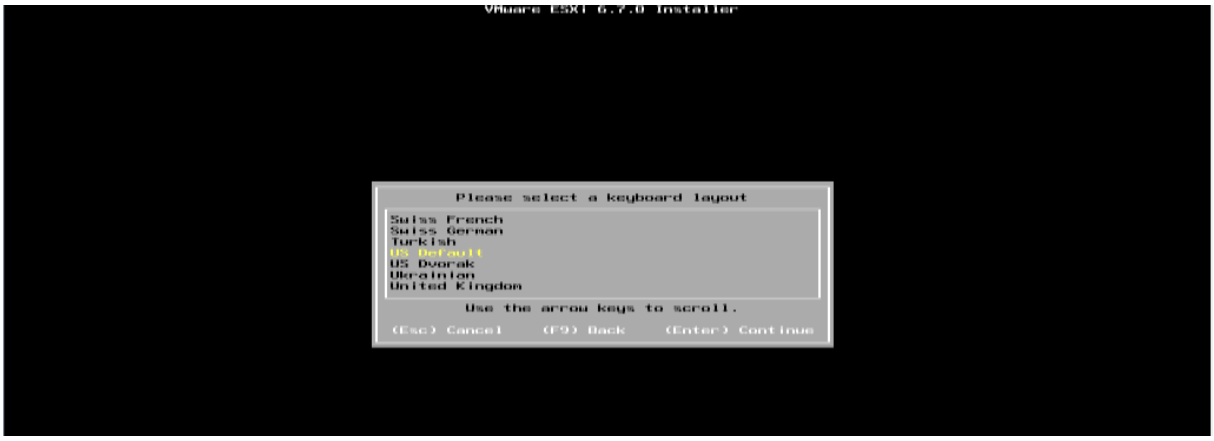
- ESXi6.7requiresahostmachinewithatleasttwoCPUcores.
- ESXi6.7supports64-bitx86processors
- ESXi6.7requirestheNX/XDbit to be enabled for the CPU in the BIOS.
- ESXi6.7requiresaminimumof4GBofphysicalRAM.Itisrecommended to provide atleast 8 GB of RAM to run virtual machines in typical productionenvironments.
- Tosupport64-bitvirtualmachines,support for hardware virtualization (IntelVT-xor AMDRVI) mustbeenabledonx64CPUs.
- One or more Gigabit or faster Ethernet controllers. For a list of supportednetwork adapter models.
- SCSI disk oralocal,non-network,RAIDLUN with unpartitioned space for the virtualmachines.

ForSerialATA(SATA), a disk connected through supported SAS controller or supported on board SATA controllers. SATA disks are considered remote not local. These disks are not used as a scratch partition by default be cause they are seen as remote.

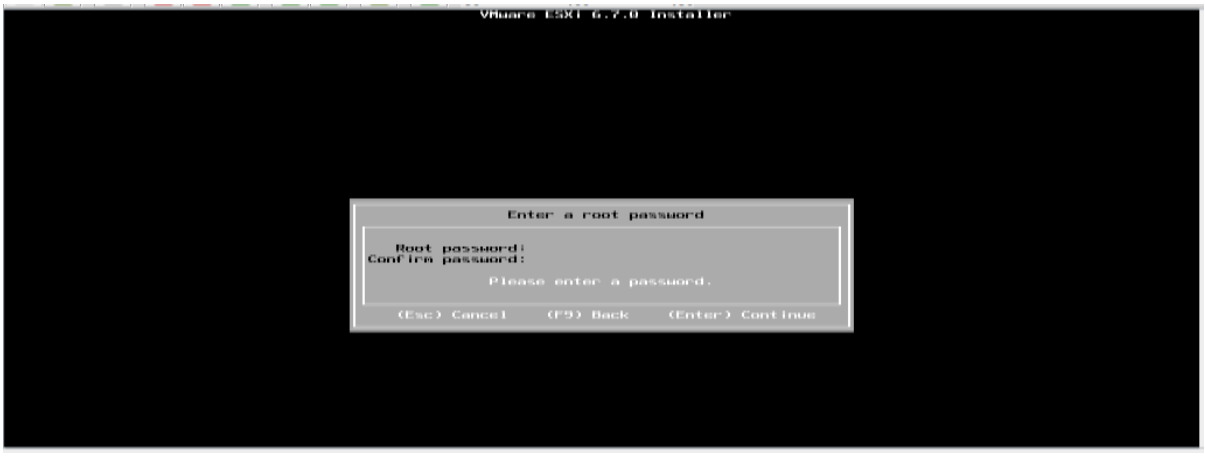
ESXiInstaller:

Accept Agreement:

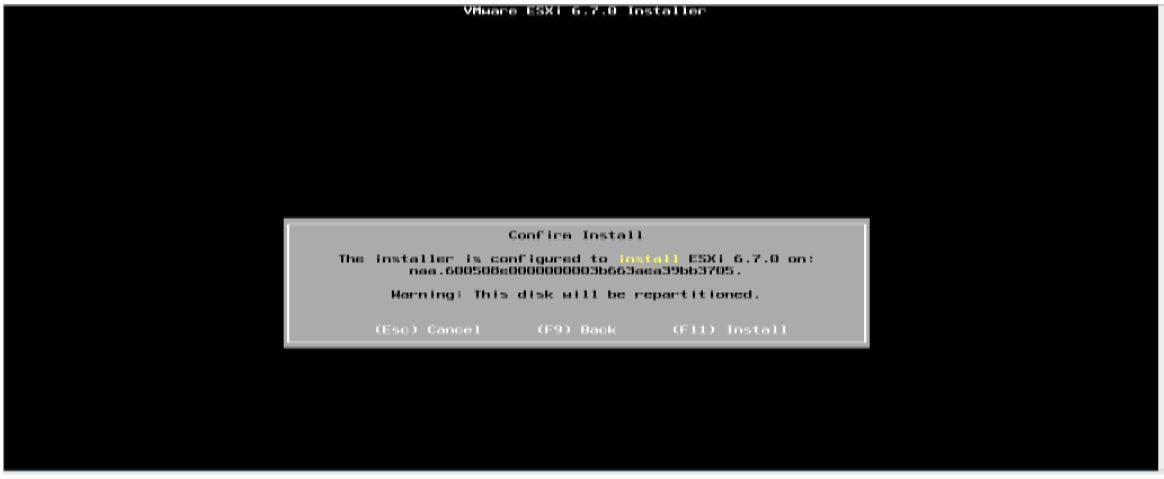
Select storage :



Select Keyboard Layout :

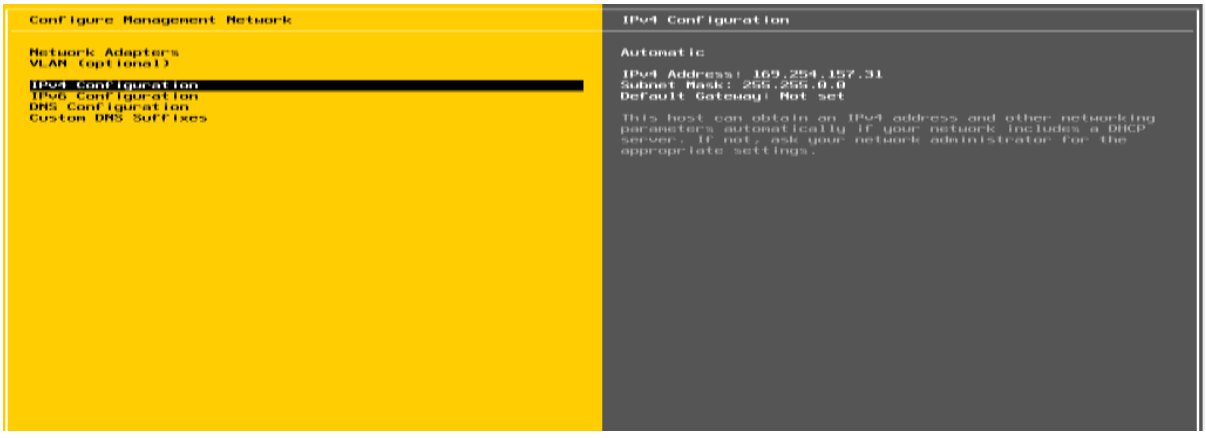


Set NodeESXi Root Password :



Installation complete (Reboot)CLII interface to configuration

CLI Interface to Configuration:

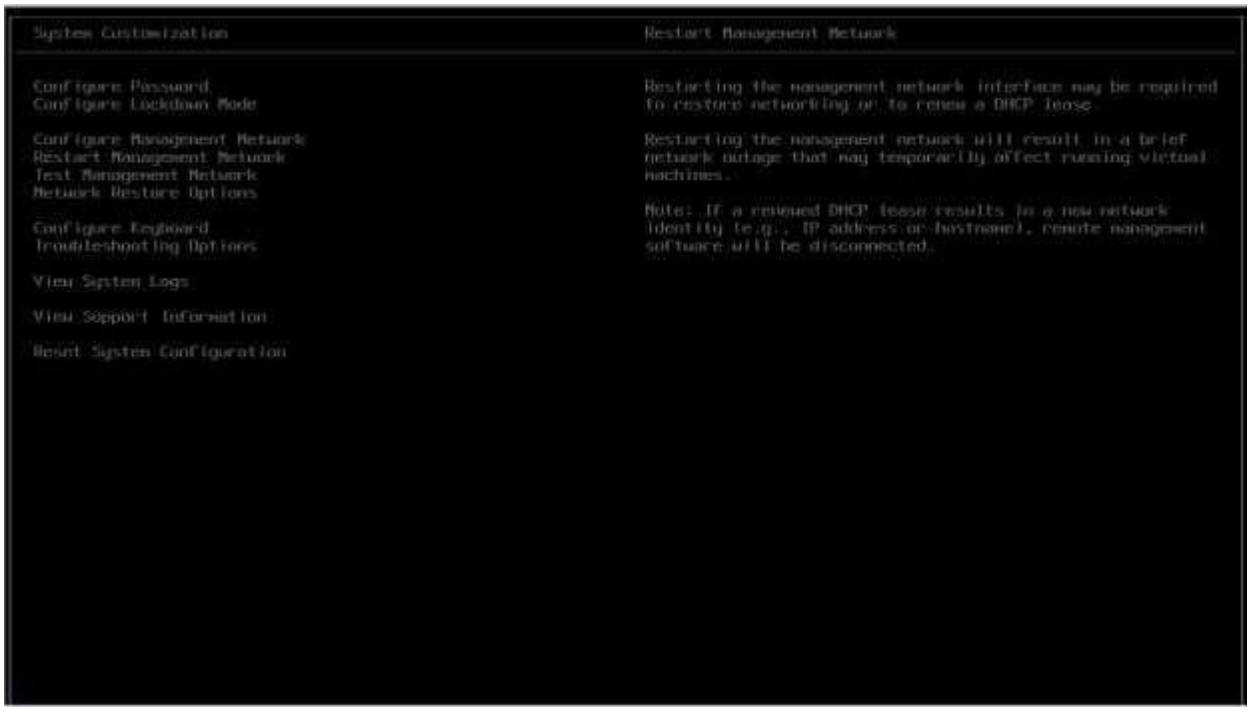


Configure Management Network

Set IPV4

Set DNSeriver :

Restart Management Network

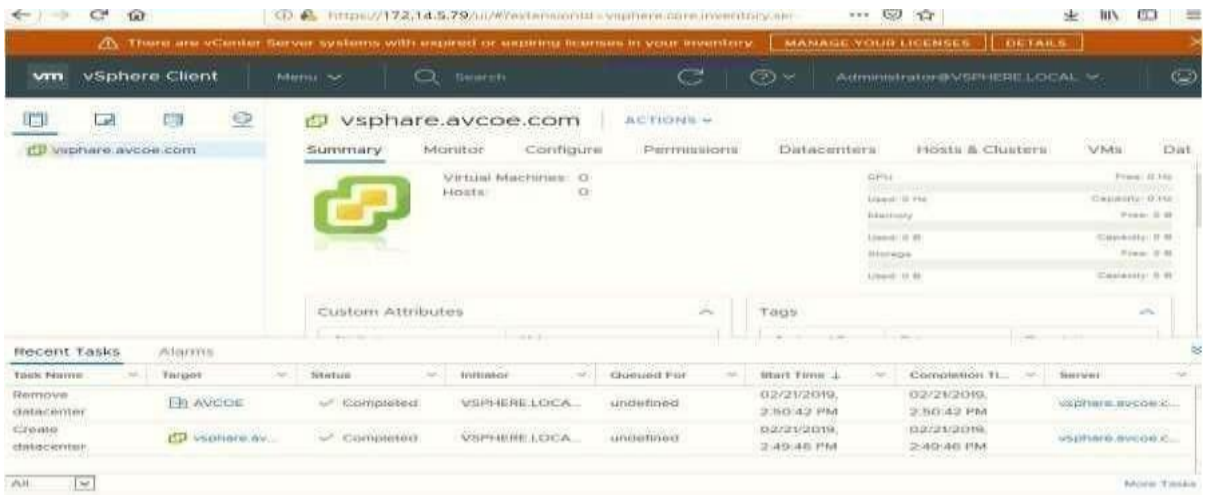


GUIAccess :

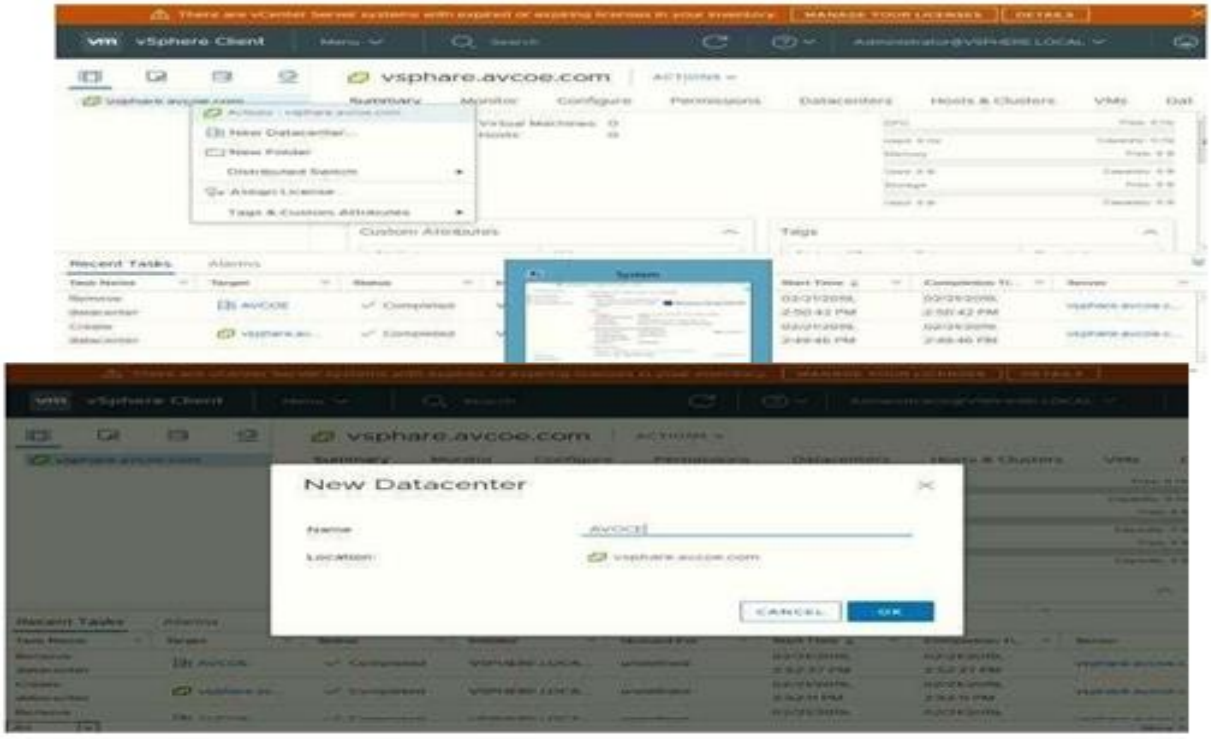
ClusterSetup

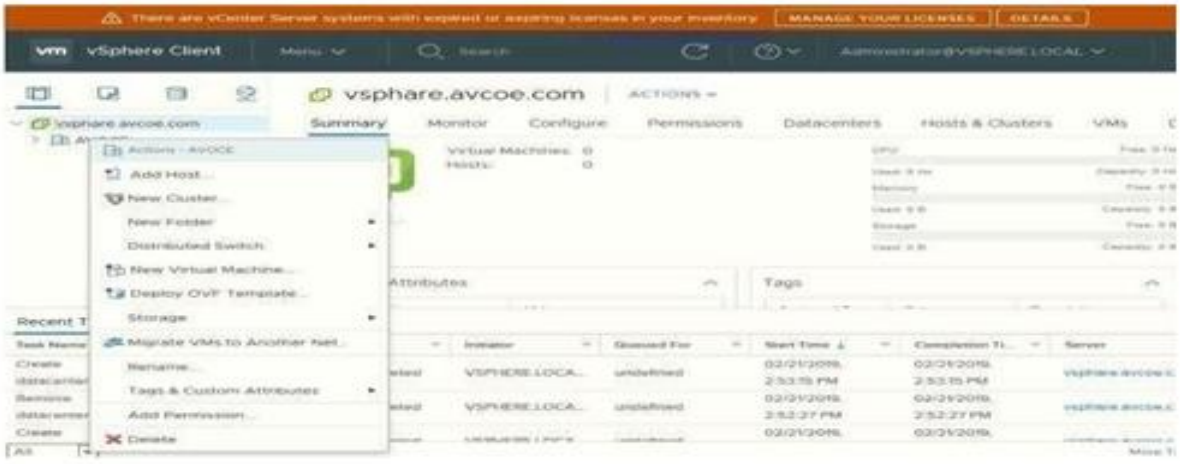
- **CreatingDatacenter**
- **CreatingCluster**
- **Adding Hosts incluster**
- **Resourcesafteraddingcluster.**
- **DRS**
- **Failover**

VCenter Access:



Create Datacenter:

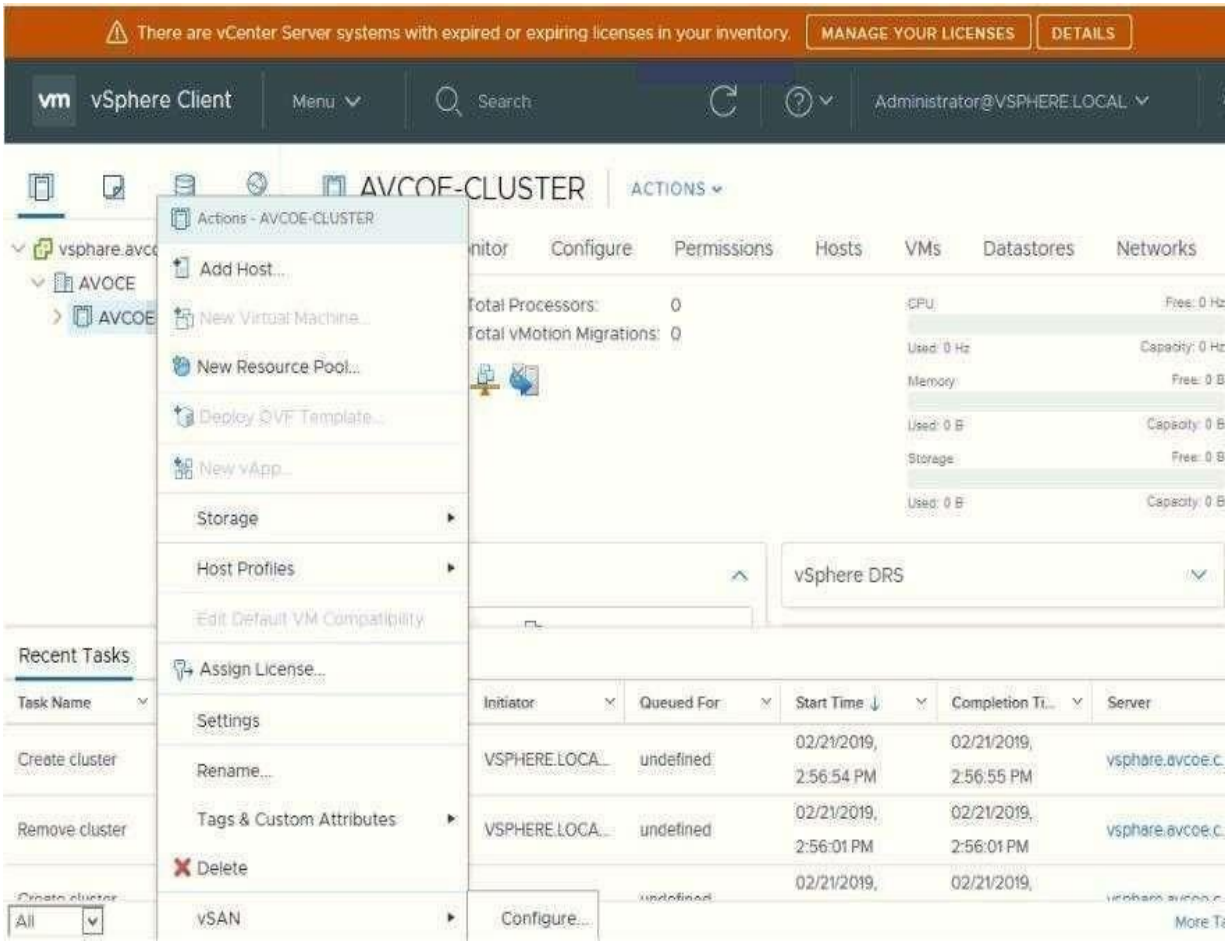




Create cluster : Assign

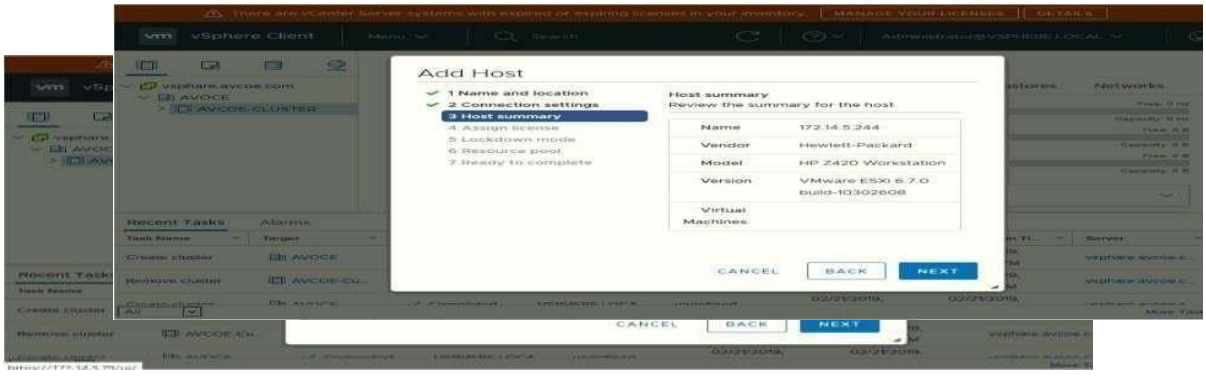
cluster name :

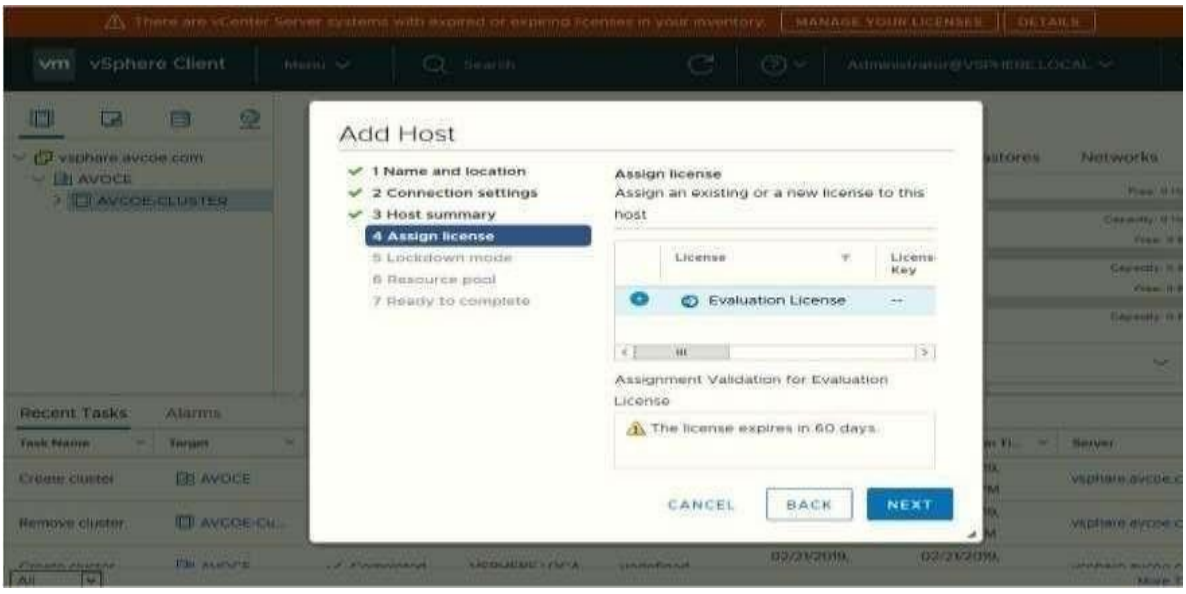
Add host .:



Add host IP :

Enter host credential :

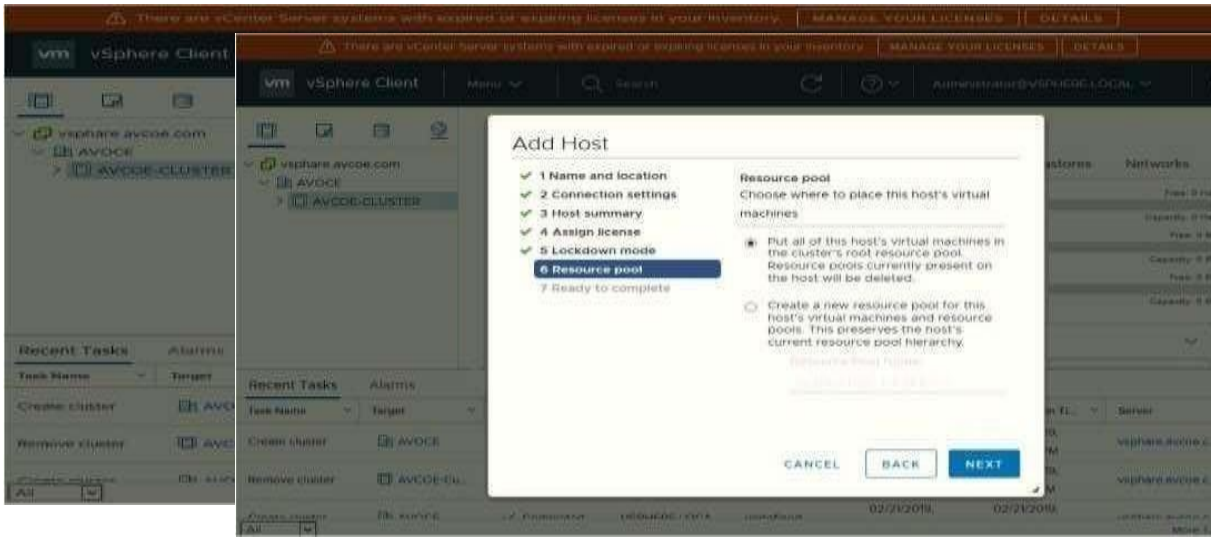




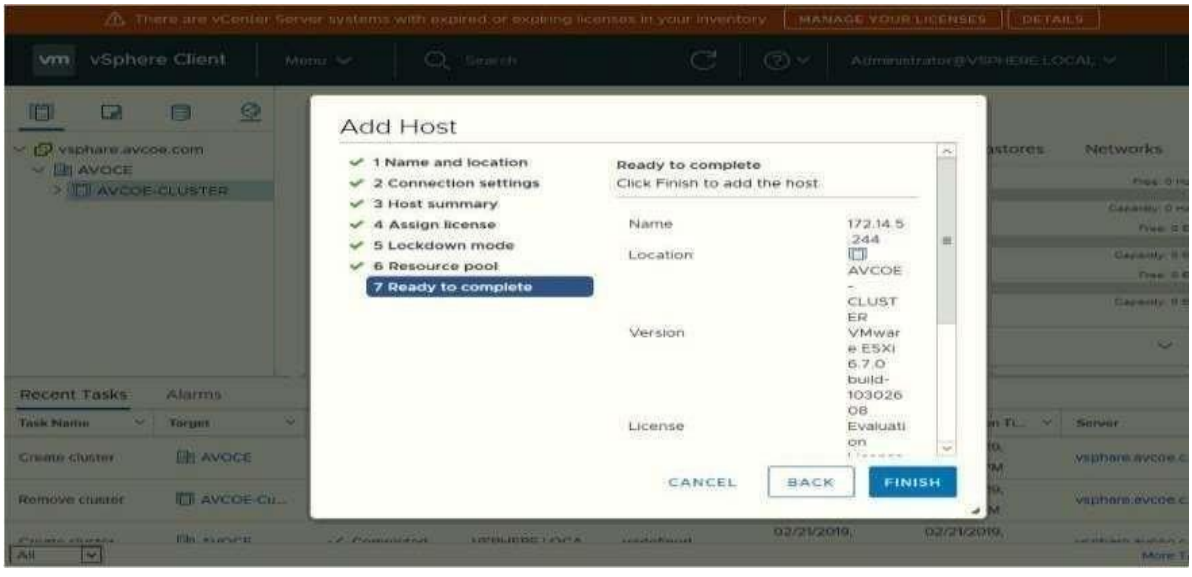
Hot summary :

Lock Down mode:

Add Host In Pool:

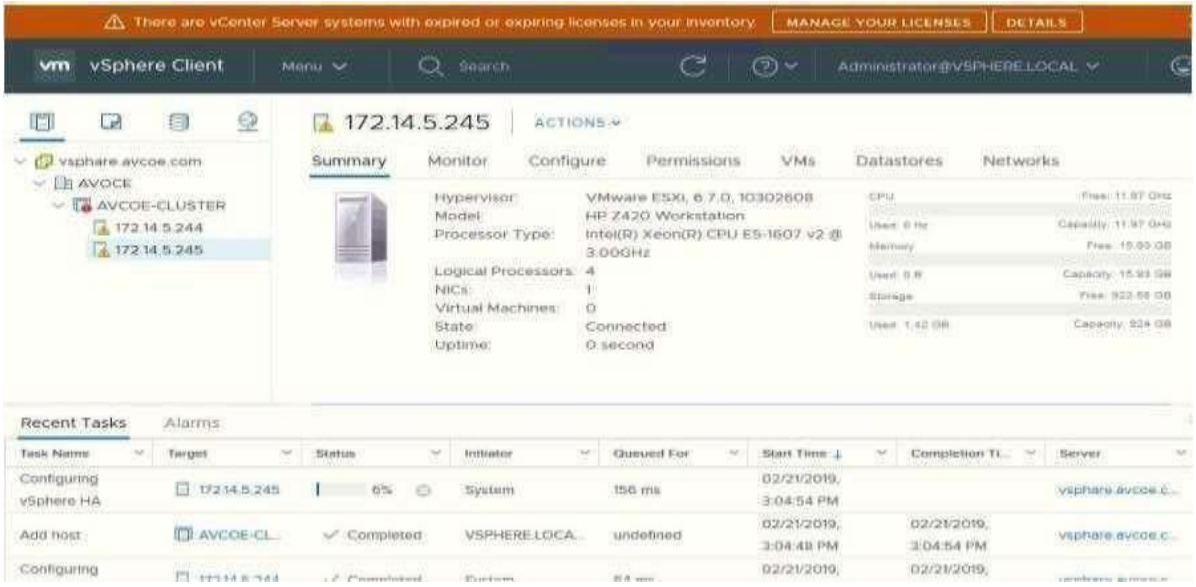


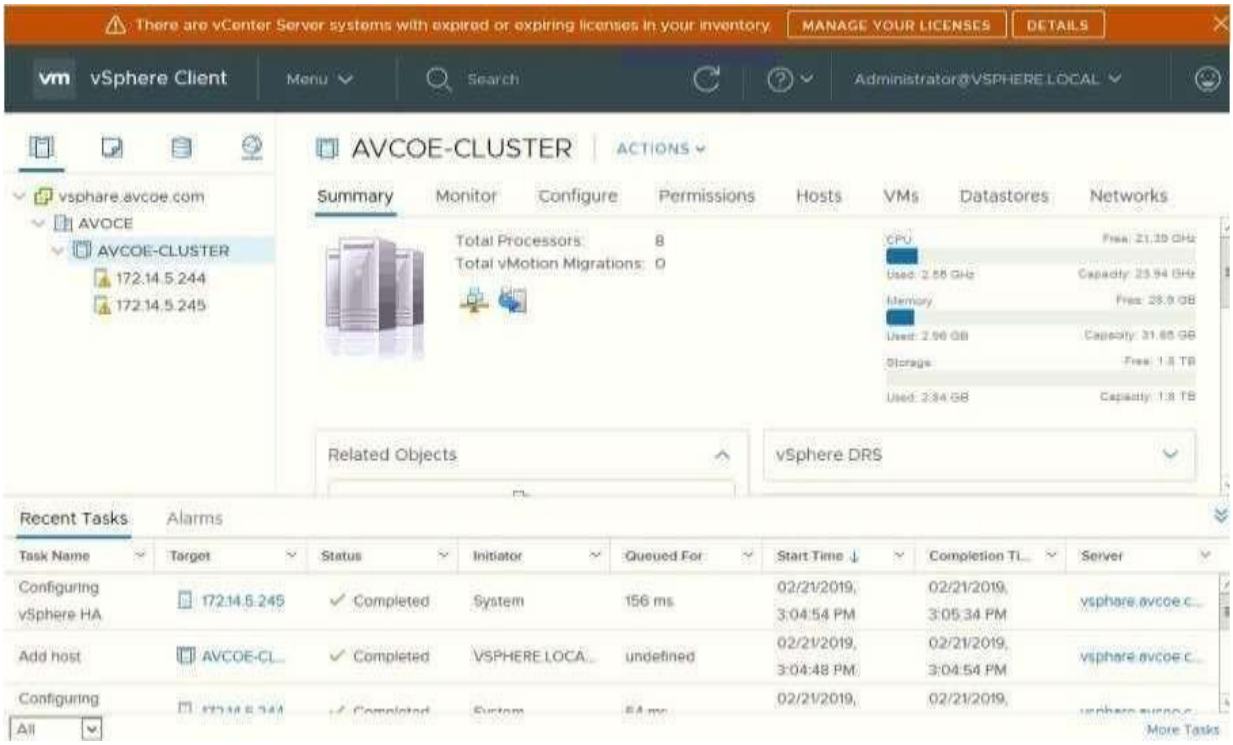
Finish:



Host View and View Config:

Cluster View and Configuration:





Conclusion: Like this we have configure VSphere Private Cloud