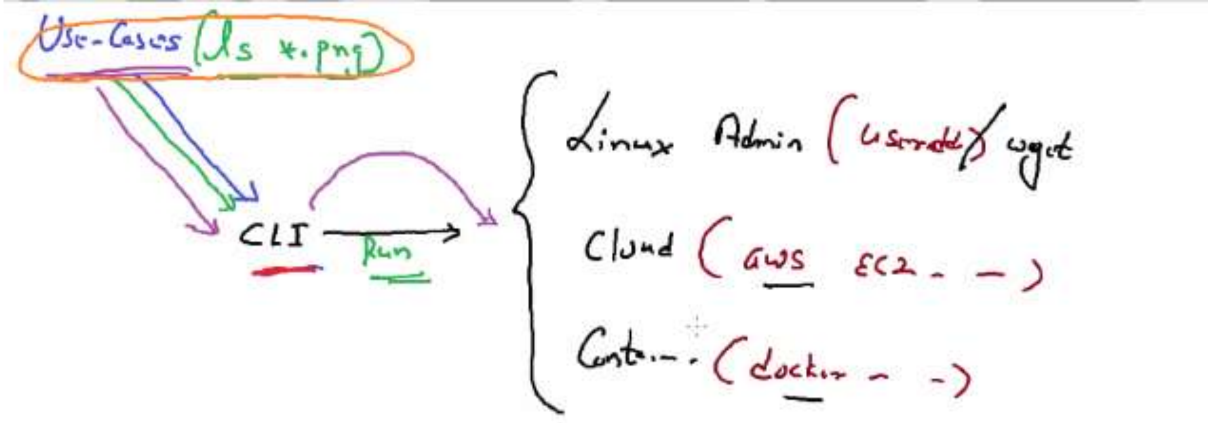
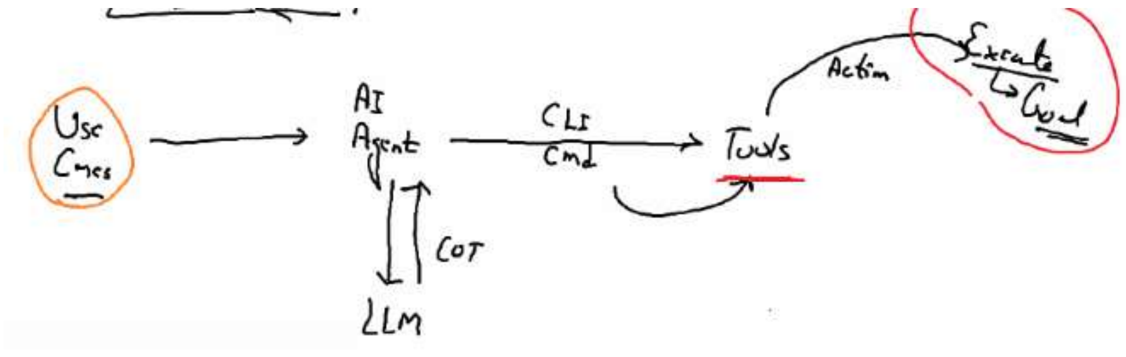
Generative AIOps – LLMO AutoCMD project

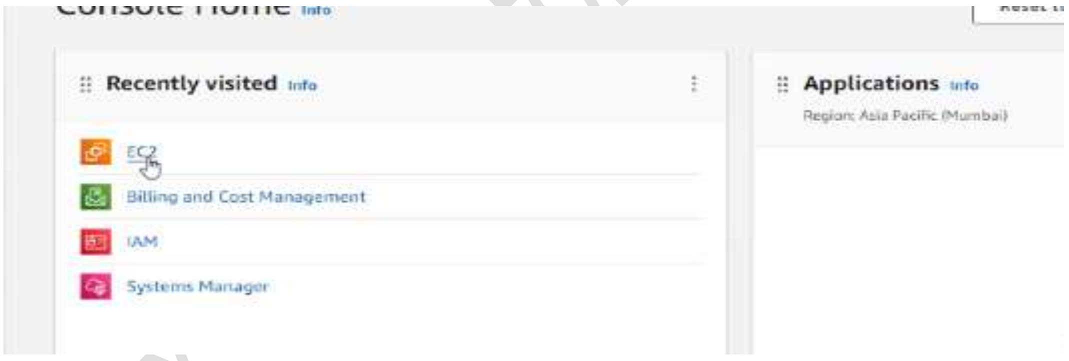
- Today we created a great LLM model for the CLI
- The topic of **GenAI Ops** (Generative AI Operations) and its integration into various tasks, especially those related to command-line interfaces (CLI) and platform management in Linux, cloud environments, and container technologies
- We are now discussing how traditional methods of using CLI for tasks like creating users, managing containers, or working with cloud services (AWS, Docker, etc.) can be replaced or enhanced using **Generative AI tools**.
- The concept of using generative AI in operations to automate or simplify tasks that traditionally require specific commands or CLI usage.
- Traditionally, platforms like Linux, AWS, and Docker require users to execute commands via CLI (e.g., user add in Linux, aws ec2 in AWS, etc.). These commands must be manually learned and executed by the user.



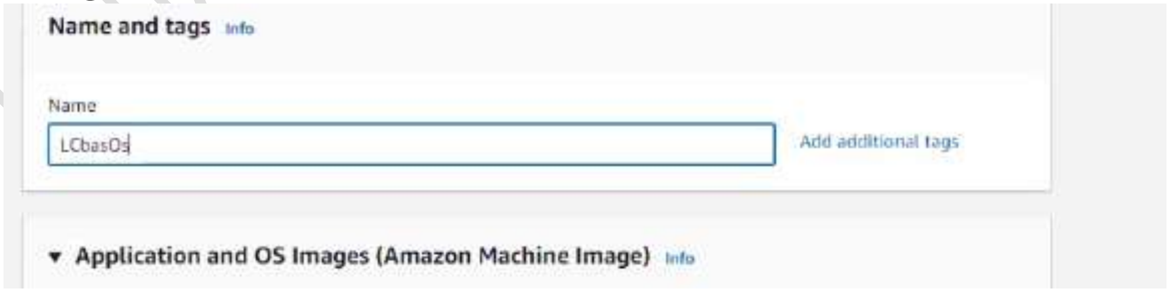
- By using **AI Agents** or LLMs (Large Language Models), users can offload tasks to AI by simply stating their problem or use case, and the AI will handle the command execution for them. For example, the AI could automatically generate the necessary CLI commands based on the user's needs.
- The AI agent communicates with the LLM model and discusses the problem, determining the correct commands or actions required. This eliminates the need for the user to have deep knowledge of individual commands.
- **Real-World Application**: we imply that in future use cases, individuals may no longer need to memorize commands or CLI syntax, as AI tools can execute these tasks based on input from the user.



- we're learning about how to use **GenAI Ops** with **command-line tools** in a **Linux environment**, specifically using the **Bash shell**. The Bash shell is typically found in Linux operating systems, and for the demonstration, Now we use **AWS Cloud** to launch a **Linux instance**.
- Let's start a Practical
- Uses **EC2** (Elastic Compute Cloud) in AWS to create a virtual machine (or instance).
- First we launch the AWS Ec2 instance



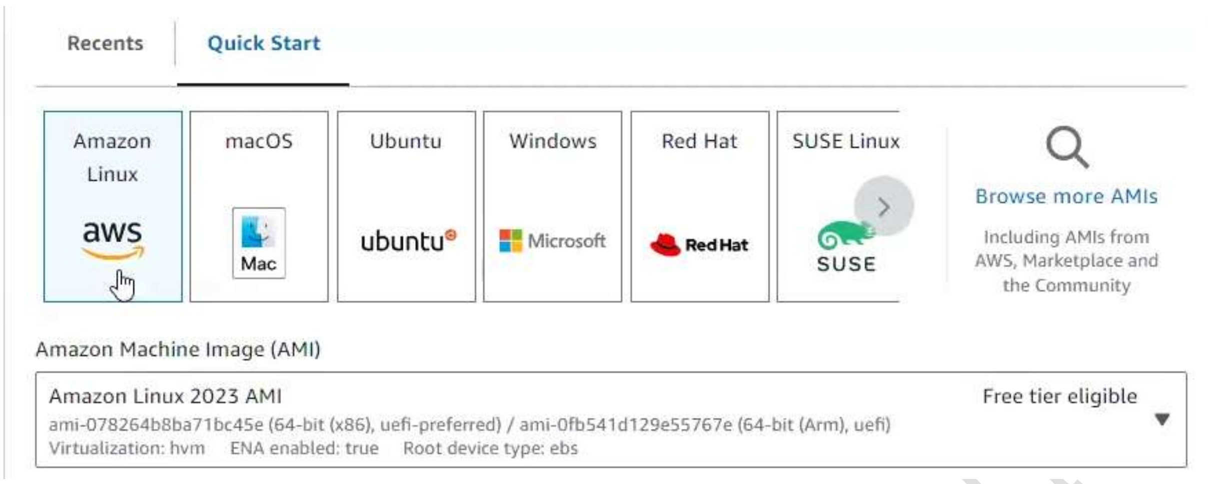
- First give the name of the instance


Name and tags Info

Name

LCbasOs

- After that select **Amazon Linux** as the operating system, which is a common Linux distribution for cloud environments.


Recents | Quick Start

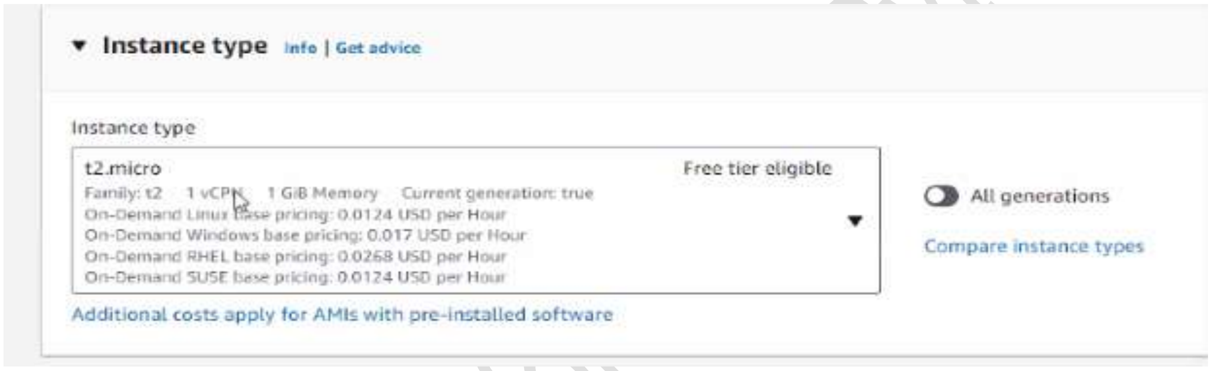
Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Linux

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI — Free tier eligible
ami-078264b8ba71bc45e (64-bit (x86), uefi-preferred) / ami-0fb541d129e55767e (64-bit (Arm), uefi)
Virtualization: hvm   ENA enabled: true   Root device type: ebs

- Now choose a **T2 micro instance**, a small, free-tier instance, and mention enabling a firewall to manage network traffic.


Instance type Info | Get advice

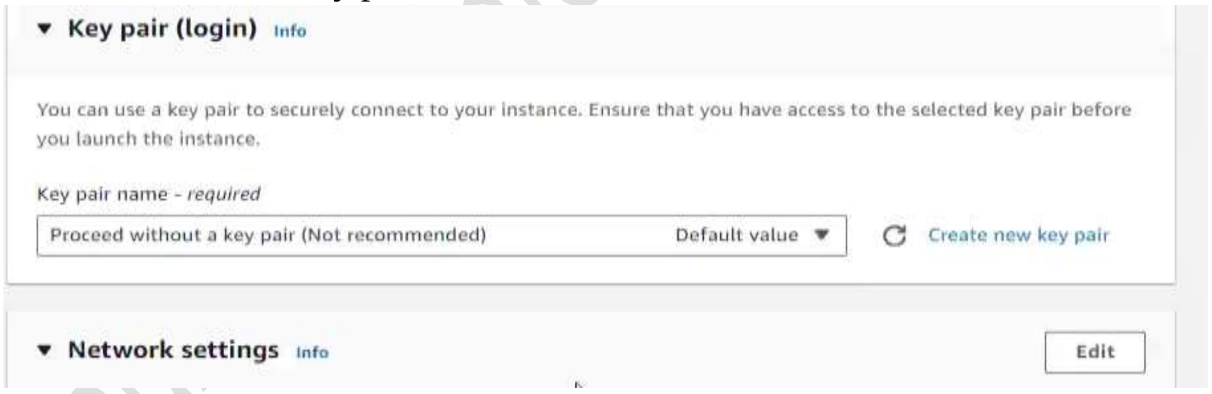
Instance type

t2.micro — Free tier eligible
Family: t2   1 vCPU   1 GiB Memory   Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour
On-Demand Windows base pricing: 0.017 USD per Hour
On-Demand RHEL base pricing: 0.0268 USD per Hour
On-Demand SUSE base pricing: 0.0124 USD per Hour

All generations
Compare instance types

Additional costs apply for AMIs with pre-installed software

- After that select no key pair


Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

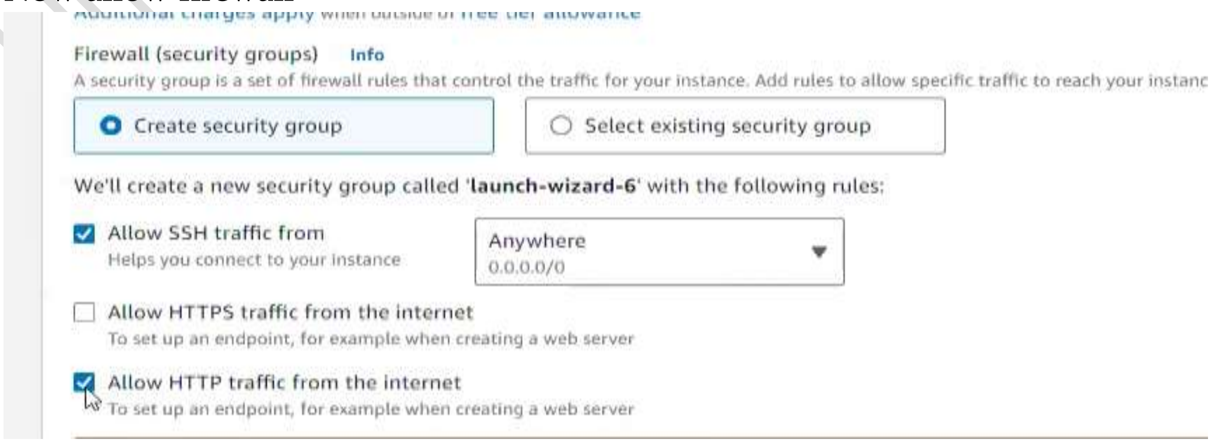
Key pair name - *required*

Proceed without a key pair (Not recommended) — Default value

Create new key pair

Network settings Info   Edit

- Now allow firewall


Additional charges apply when outside of free tier allowance

Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instanc

[✓ Create security group] Select existing security group

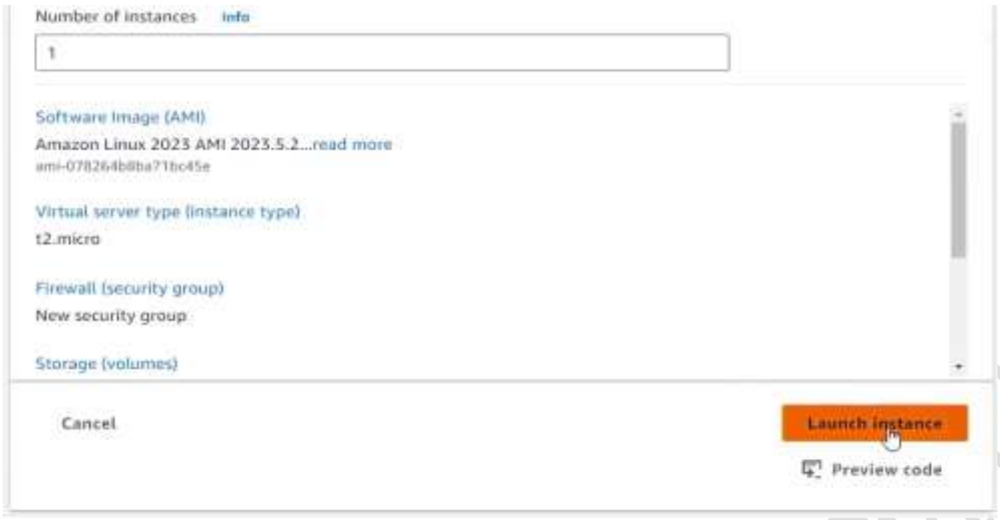
We'll create a new security group called '**launch-wizard-6**' with the following rules:

[✓] Allow SSH traffic from — Anywhere 0.0.0.0/0
Helps you connect to your instance

[ ] Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

[✓] Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

- After that click on the launch firewall



Number of instances Info

1

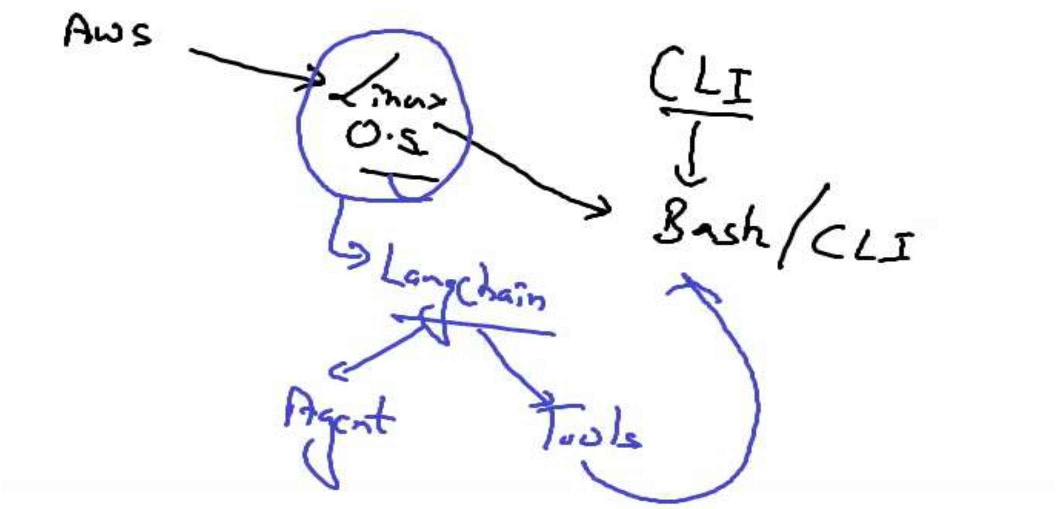
Software Image (AMI)
Amazon Linux 2023 AMI 2023.5.2...read more
ami-078264b8ba71bc45e

Virtual server type (instance type)
t2.micro

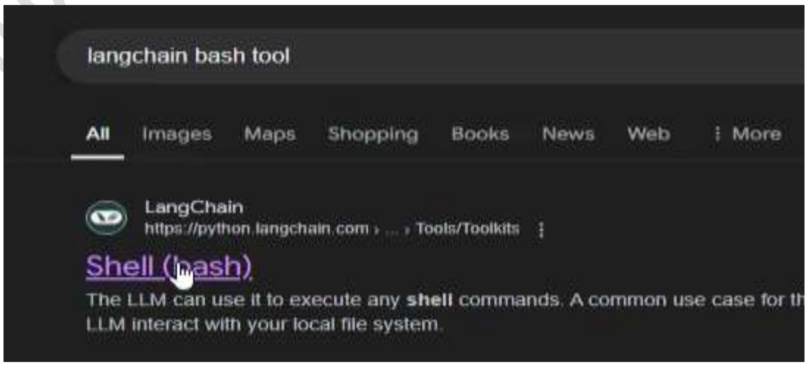
Firewall (security group)
New security group

Storage (volumes)

- After launching the instance, we need to set up **Langchain**, a tool that provides various agents and tools for interacting with the environment.



- In this case, the main tool used is **Bash**, which will allow running commands in the Linux terminal.

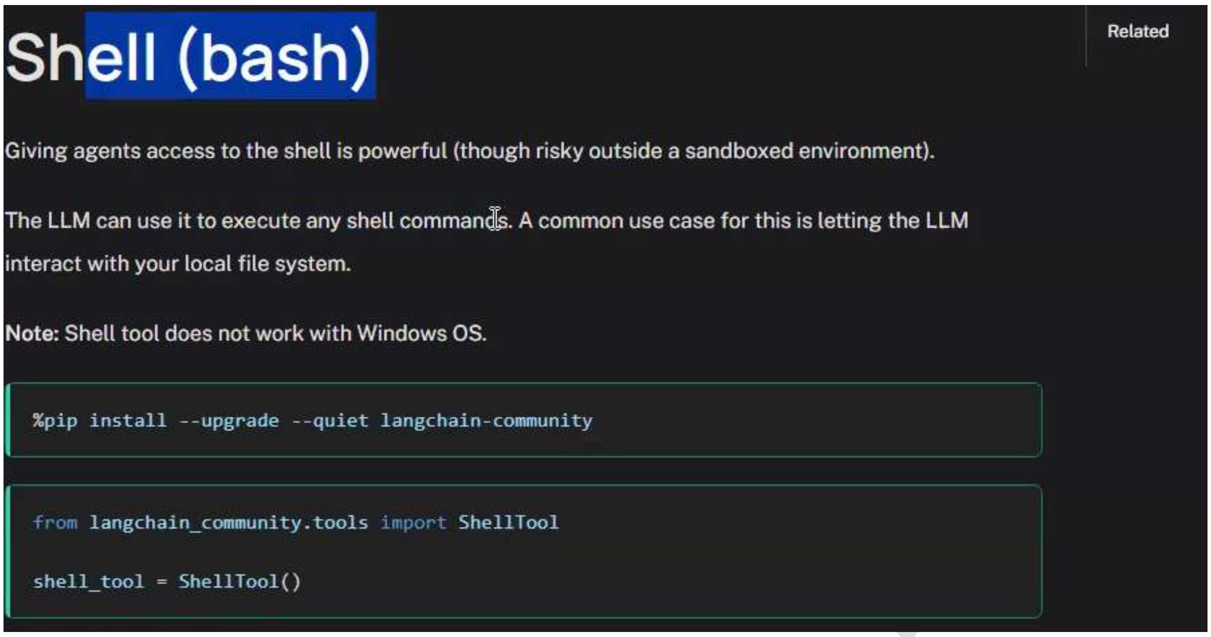


langchain bash tool

LangChain
https://python.langchain.com › ... › Tools/Toolkits

Shell (bash)

The LLM can use it to execute any **shell** commands. A common use case for th
LLM interact with your local file system.



Shell (bash)

Giving agents access to the shell is powerful (though risky outside a sandboxed environment).

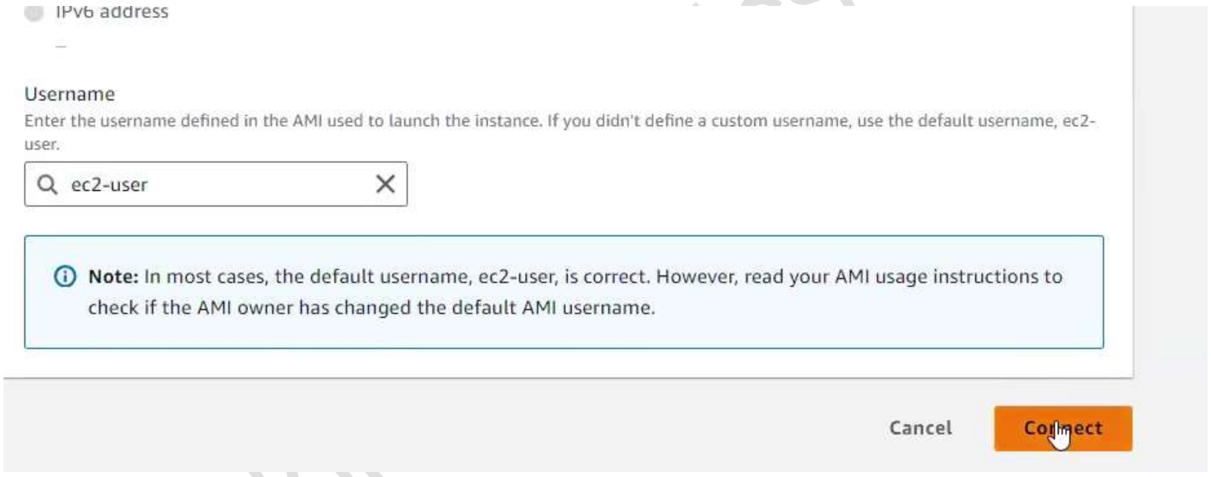
The LLM can use it to execute any shell commands. A common use case for this is letting the LLM interact with your local file system.

Note: Shell tool does not work with Windows OS.

```
%pip install --upgrade --quiet langchain-community
```

```
from langchain_community.tools import ShellTool

shell_tool = ShellTool()
```

- Now we connect with the Ec2 instance CLI

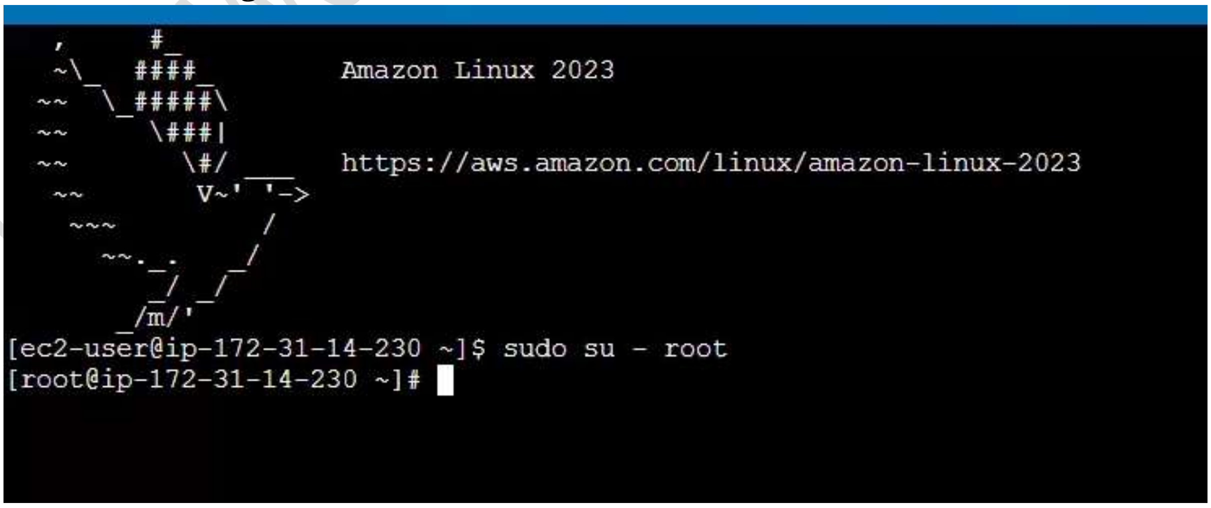


Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.

ec2-user

Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

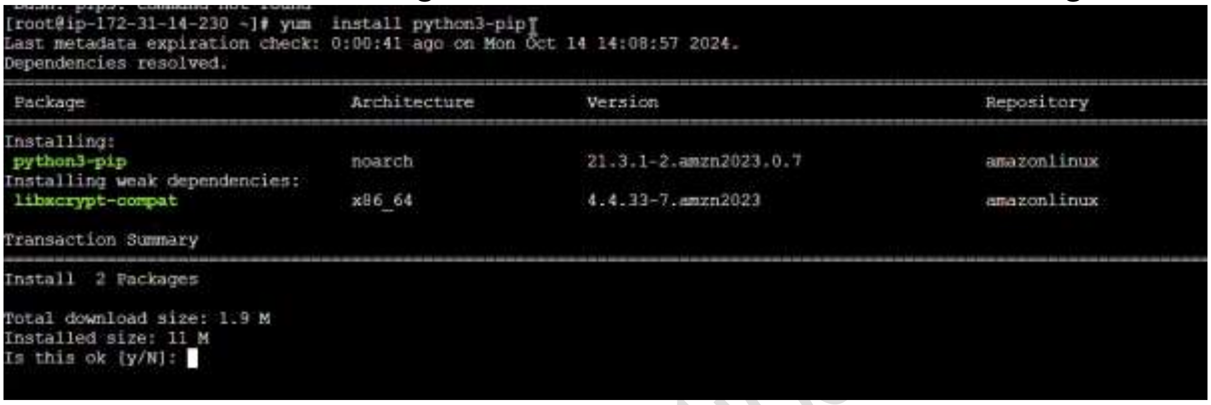
- After that we log in with the root user



```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-14-230 ~]$ sudo su - root
[root@ip-172-31-14-230 ~]# 
```

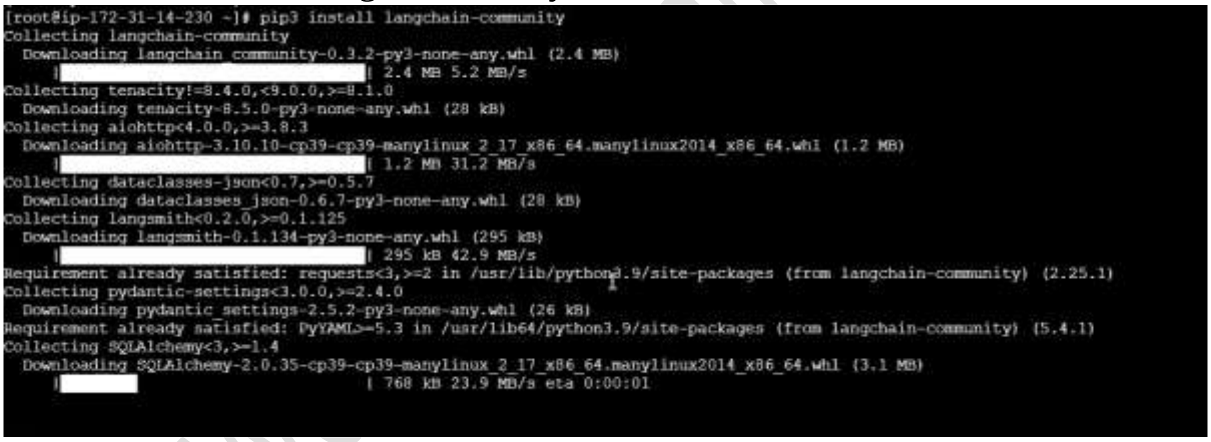
- We are going to install a tool called Linkchain, which is a Python library. This library provides various tools, and in our case, we are particularly interested in a tool for working with Bash commands.

- On this Linux instance, we need the pip command to install Python libraries. Since pip isn't installed by default, we install it using the command yum install python3-pip. This command installs pip3, which will help us manage Python libraries.
- Once pip is installed, we use it to install the Linkchain library (pip3 install linkchain-community). This library is open-source and contains various modules and tools, including the Bash command tool that we'll be using.



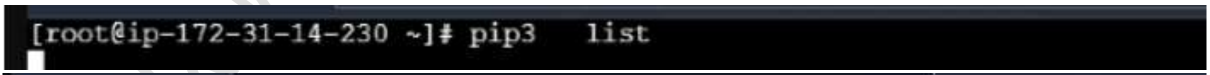
```
[root@ip-172-31-14-230 ~]# yum  install python3-pip
Last metadata expiration check: 0:00:41 ago on Mon Oct 14 14:08:57 2024.
Dependencies resolved.
================================================================================================
 Package                  Architecture        Version                          Repository
================================================================================================
Installing:
 python3-pip              noarch              21.3.1-2.amzn2023.0.7            amazonlinux
Installing weak dependencies:
 libxcrypt-compat         x86_64              4.4.33-7.amzn2023                amazonlinux

Transaction Summary
================================================================================================
Install  2 Packages

Total download size: 1.9 M
Installed size: 11 M
Is this ok [y/N]:
```

- After that install the langchain library

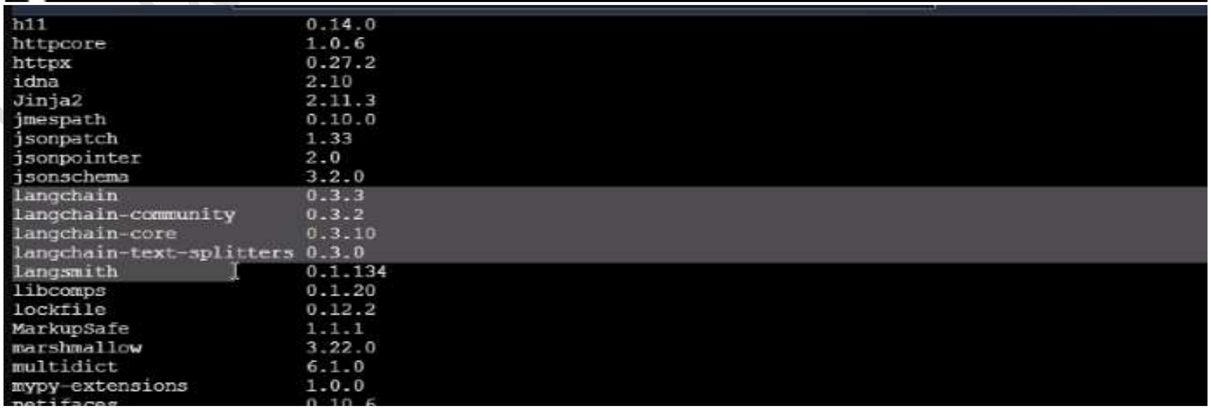


```
[root@ip-172-31-14-230 ~]# pip3 install langchain-community
Collecting langchain-community
  Downloading langchain_community-0.3.2-py3-none-any.whl (2.4 MB)
     |████████████████████████████████| 2.4 MB 5.2 MB/s
Collecting tenacity!=8.4.0,<9.0.0,>=8.1.0
  Downloading tenacity-8.5.0-py3-none-any.whl (28 kB)
Collecting aiohttp<4.0.0,>=3.8.3
  Downloading aiohttp-3.10.10-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2 MB)
     |████████████████████████████████| 1.2 MB 31.2 MB/s
Collecting dataclasses-json<0.7,>=0.5.7
  Downloading dataclasses_json-0.6.7-py3-none-any.whl (28 kB)
Collecting langsmith<0.2.0,>=0.1.125
  Downloading langsmith-0.1.134-py3-none-any.whl (295 kB)
     |████████████████████████████████| 295 kB 42.9 MB/s
Requirement already satisfied: requests<3,>=2 in /usr/lib/python3.9/site-packages (from langchain-community) (2.25.1)
Collecting pydantic-settings<3.0.0,>=2.4.0
  Downloading pydantic_settings-2.5.2-py3-none-any.whl (26 kB)
Requirement already satisfied: PyYAML>=5.3 in /usr/lib64/python3.9/site-packages (from langchain-community) (5.4.1)
Collecting SQLAlchemy<3,>=1.4
  Downloading SQLAlchemy-2.0.35-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.1 MB)
     |████████                        | 768 kB 23.9 MB/s eta 0:00:01
```

- To check use the below command

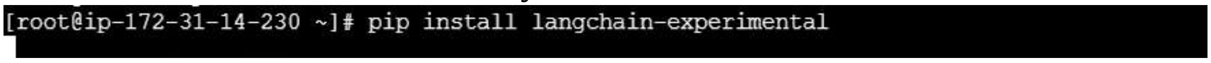


```
[root@ip-172-31-14-230 ~]# pip3   list
```



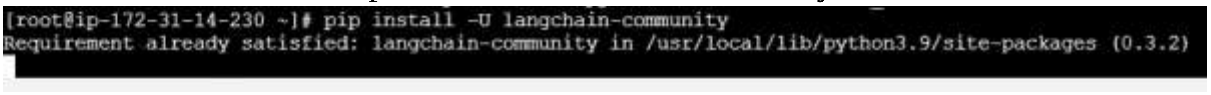
```
h11                        0.14.0
httpcore                   1.0.6
httpx                      0.27.2
idna                       2.10
Jinja2                     2.11.3
jmespath                   0.10.0
jsonpatch                  1.33
jsonpointer                2.0
jsonschema                 3.2.0
langchain                  0.3.3
langchain-community        0.3.2
langchain-core             0.3.10
langchain-text-splitters   0.3.0
langsmith                  0.1.134
libcomps                   0.1.20
lockfile                   0.12.2
MarkupSafe                 1.1.1
marshmallow                3.22.0
multidict                  6.1.0
mypy-extensions            1.0.0
```

- After that we need one more library to install



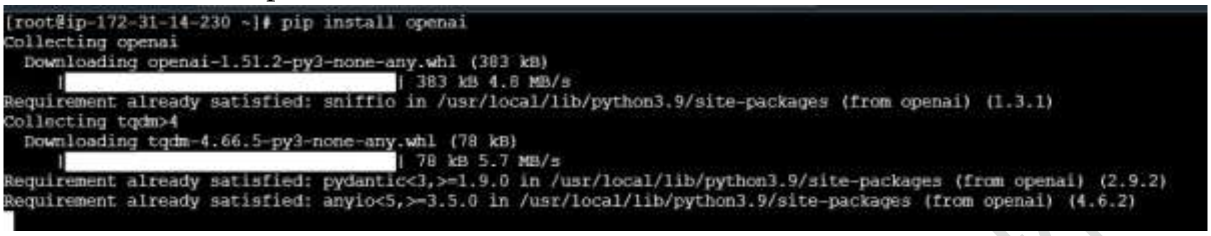
```
[root@ip-172-31-14-230 ~]# pip install langchain-experimental
```

- Sometimes we need to update the linkchain-community



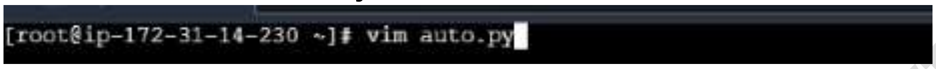
```
[root@ip-172-31-14-230 ~]# pip install -U langchain-community
Requirement already satisfied: langchain-community in /usr/local/lib/python3.9/site-packages (0.3.2)
```

- Now we install OpenAI



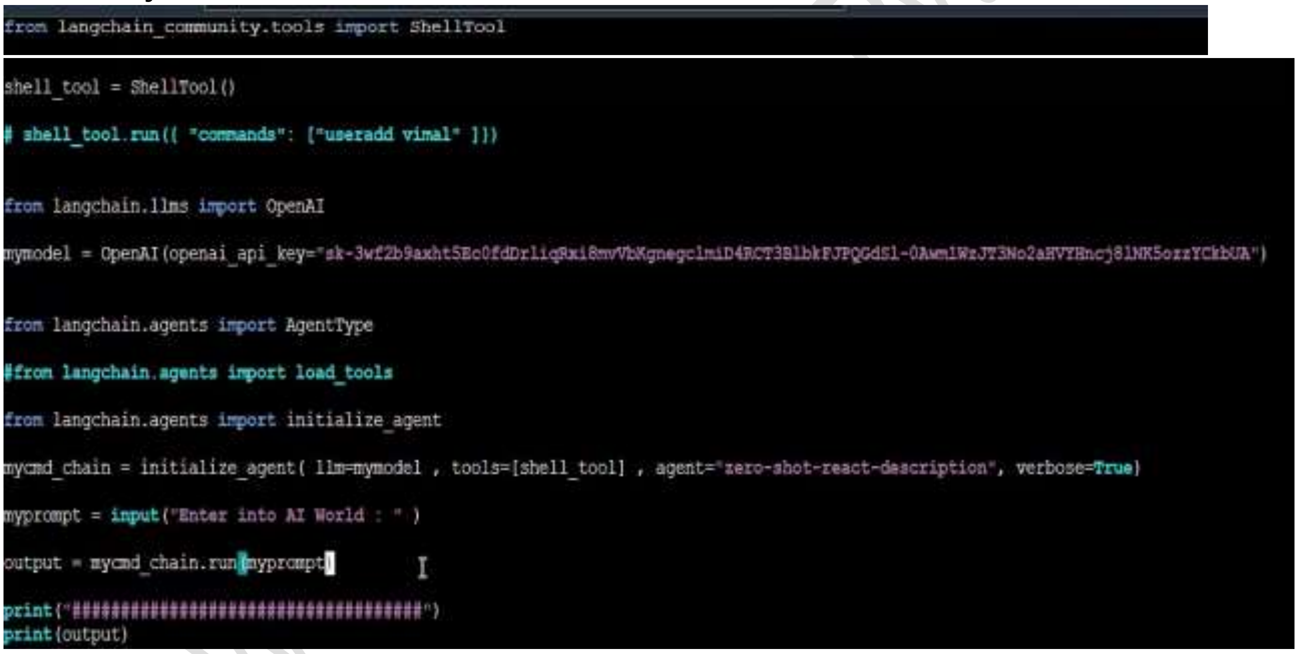
```
[root@ip-172-31-14-230 ~]# pip install openai
Collecting openai
  Downloading openai-1.51.2-py3-none-any.whl (383 kB)
     |████████████████████████████████| 383 kB 4.8 MB/s
Requirement already satisfied: sniffio in /usr/local/lib/python3.9/site-packages (from openai) (1.3.1)
Collecting tqdm>4
  Downloading tqdm-4.66.5-py3-none-any.whl (78 kB)
     |████████████████████████████████| 78 kB 5.7 MB/s
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.9/site-packages (from openai) (2.9.2)
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.9/site-packages (from openai) (4.6.2)
```

- After that we create a Python file



```
[root@ip-172-31-14-230 ~]# vim auto.py
```

- Write a Python code for the LLM model



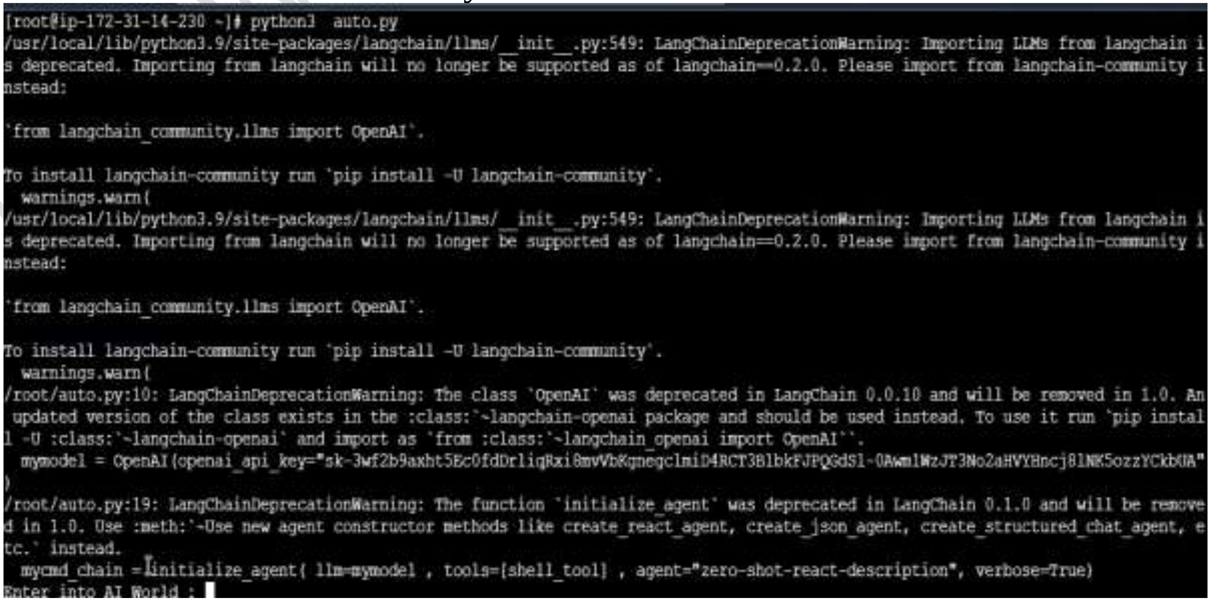
```
from langchain_community.tools import ShellTool

shell_tool = ShellTool()

# shell_tool.run({ "commands": ["useradd vimal" ]})

from langchain.llms import OpenAI

mymodel = OpenAI(openai_api_key="sk-3wf2b9axht5Ec0fdDrliqRxi8mvVbKgnegclmiD4RCT3BlbkFJPQGdSl-0Awm1WzJT3No2aHVYHncj8lNK5ozzYCkbUA")

from langchain.agents import AgentType

#from langchain.agents import load_tools

from langchain.agents import initialize_agent

mycmd_chain = initialize_agent( llm=mymodel , tools=[shell_tool] , agent="zero-shot-react-description", verbose=True)

myprompt = input("Enter into AI World : " )

output = mycmd_chain.run(myprompt)

print("###################################")
print(output)
```

----

from langchain_community.tools import ShellTool
shell_tool = shell_tool()
#shell_tool.run({"commands": ["useradd vimal"]})
from langchain.llms import OpenAI
mymodel = OpenAI(openai_api_key="please enter your API key ")
from langchain.agents import AgentType
from langchain.agents import initialize_agent
mycmd_chain = initialize_agent(llm=mymodel , tools=[shell_tool] ,
agent="zero-shot-react-description" , verbose= True)
myprompt = input("Enter your prompt")
output = mycmd_chain.run(myprompt)
print(ouput)

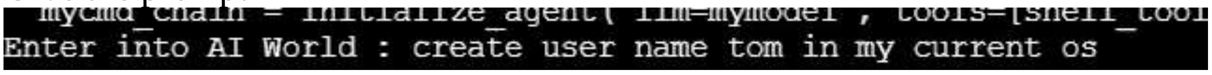
- In the above code use the LangChain library to integrate natural language processing with shell command execution. It allows the user to enter natural language prompts, which will then be interpreted by an AI language model (OpenAI's GPT) to run shell commands
- Importing Required Modules
  from langchain_community.tools import ShellTool
  from langchain.llms import OpenAI
  from langchain.agents import AgentType
  from langchain.agents import initialize_agent
- **ShellTool**: This is a tool provided by LangChain that allows the execution of shell commands through Python. It can be used to run system commands in an operating system shell (e.g., Linux terminal).
- **OpenAI**: This is the API wrapper for OpenAI's language models (like GPT-3, GPT-4). It allows interaction with OpenAI's models using a provided API key.
- **AgentType** and **initialize_agent**: These are utilities from LangChain that help to create agents. Agents are entities that use large language models to take actions, such as running commands, based on input.
- **Initializing the Shell Tool**
  shell_tool = ShellTool()
- **ShellTool()**: This initializes the ShellTool object, which will be responsible for running shell commands.
- Commented Command to Run Shell Tool
  # shell_tool.run({"commands": ["useradd vimal"]})
- This line is commented out, but if it were active, it would use the shell_tool to run a Linux command (useradd vimal) to add a user named "vimal" to the system.
- Initializing the OpenAI Language Model
  mymodel = OpenAI(openai_api_key="please enter your API key ")
- **OpenAI()**: This initializes the OpenAI language model by passing an API key. The language model will interpret user prompts in natural language and generate responses. The API key is required to authenticate and interact with OpenAI's models.
- **Note**: You need to replace "please enter your API key" with your actual OpenAI API key.
- Initializing the Agent
  mycmd_chain = initialize_agent(llm=mymodel, tools=[shell_tool], agent="zero-shot-react-description", verbose=True)

- **initialize_agent()**: This function initializes an agent that can take actions based on input using both the language model (mymodel) and tools (in this case, shell_tool).
    - ➢ **llm=mymodel**: This specifies that the agent will use the OpenAI language model to process input.
    - ➢ **tools=[shell_tool]**: This adds the ShellTool to the agent's list of tools, allowing it to run shell commands.
    - ➢ **agent="zero-shot-react-description"**: This specifies the agent type. The "zero-shot-react-description" agent uses the language model to make decisions with no prior training (zero-shot), based on the prompt and the description of available tools.
    - ➢ **verbose=True**: This flag is set to True to enable detailed logging of the agent's decisions and actions, useful for debugging.
- Accepting User Input for the Prompt
  myprompt = input("Enter your prompt")
- This line allows the user to provide a natural language prompt (e.g., "Create a new user named Vimal"). This prompt will be sent to the language model, which will interpret it and decide how to proceed.
- Running the Command Chain
  output = mycmd_chain.run(myprompt)
  print(output)
- **mycmd_chain.run(myprompt)**: This executes the agent chain. The agent uses the OpenAI model to interpret the user's prompt and decides which tool to use. In this case, it might decide to execute a shell command via shell_tool if the prompt involves such a command (e.g., creating or managing users).
- **print(output)**: Finally, the result (output) of the command is printed.
- After that we need to run the Python code



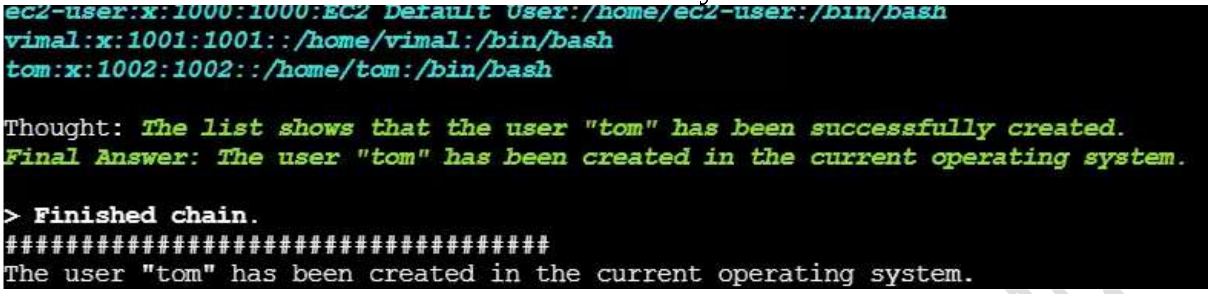
```
[root@ip-172-31-14-230 ~]# python3  auto.py
/usr/local/lib/python3.9/site-packages/langchain/llms/__init__.py:549: LangChainDeprecationWarning: Importing LLMs from langchain i
s deprecated. Importing from langchain will no longer be supported as of langchain==0.2.0. Please import from langchain-community i
nstead:

`from langchain_community.llms import OpenAI`.

To install langchain-community run `pip install -U langchain-community`.
  warnings.warn(
/usr/local/lib/python3.9/site-packages/langchain/llms/__init__.py:549: LangChainDeprecationWarning: Importing LLMs from langchain i
s deprecated. Importing from langchain will no longer be supported as of langchain==0.2.0. Please import from langchain-community i
nstead:

`from langchain_community.llms import OpenAI`.

To install langchain-community run `pip install -U langchain-community`.
  warnings.warn(
/root/auto.py:10: LangChainDeprecationWarning: The class `OpenAI` was deprecated in LangChain 0.0.10 and will be removed in 1.0. An
 updated version of the class exists in the :class:`~langchain-openai package and should be used instead. To use it run `pip instal
l -U :class:`~langchain-openai` and import as `from :class:`~langchain_openai import OpenAI``.
  mymodel = OpenAI(openai_api_key="sk-3wf2b9axht5Ec0fdDrliqRxi8mvVbKgnegclmiD4RCT3BlbkFJPQGdSl-0Awm1WzJT3No2aHVYHncj8lNK5ozzYCkbUA"
)
/root/auto.py:19: LangChainDeprecationWarning: The function `initialize_agent` was deprecated in LangChain 0.1.0 and will be remove
d in 1.0. Use :meth:`~Use new agent constructor methods like create_react_agent, create_json_agent, create_structured_chat_agent, e
tc.` instead.
  mycmd_chain =initialize_agent( llm=mymodel , tools=[shell_tool] , agent="zero-shot-react-description", verbose=True)
Enter into AI World : 
```

- Give the prompt



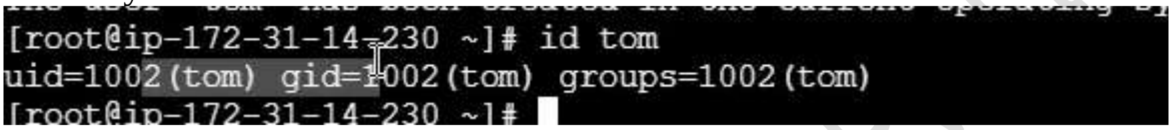
```
 mycmd_chain = initialize_agent( llm=mymodel , tools=[shell_tool
Enter into AI World : create user name tom in my current os
```

- As we can see our LLM model run successfully



```
ec2-user:x:1000:1000:EC2 Default User:/home/ec2-user:/bin/bash
vimal:x:1001:1001::/home/vimal:/bin/bash
tom:x:1002:1002::/home/tom:/bin/bash

Thought: The list shows that the user "tom" has been successfully created.
Final Answer: The user "tom" has been created in the current operating system.

> Finished chain.
####################################
The user "tom" has been created in the current operating system.
```

- And they create a user for me

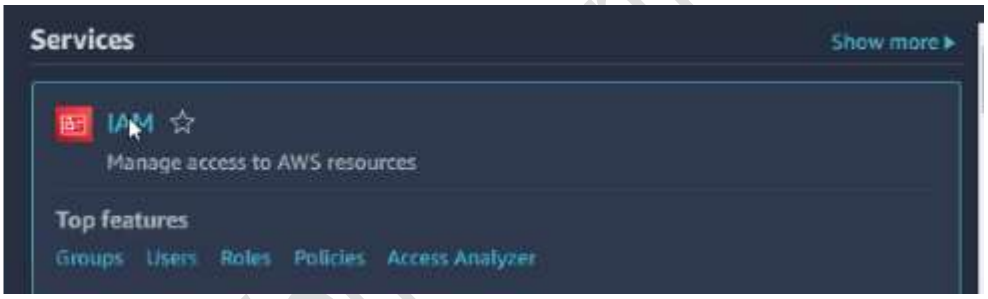


```
The user "tom" has been created in the current operating sy
[root@ip-172-31-14-230 ~]# id tom
uid=1002(tom) gid=1002(tom) groups=1002(tom)
[root@ip-172-31-14-230 ~]#
```

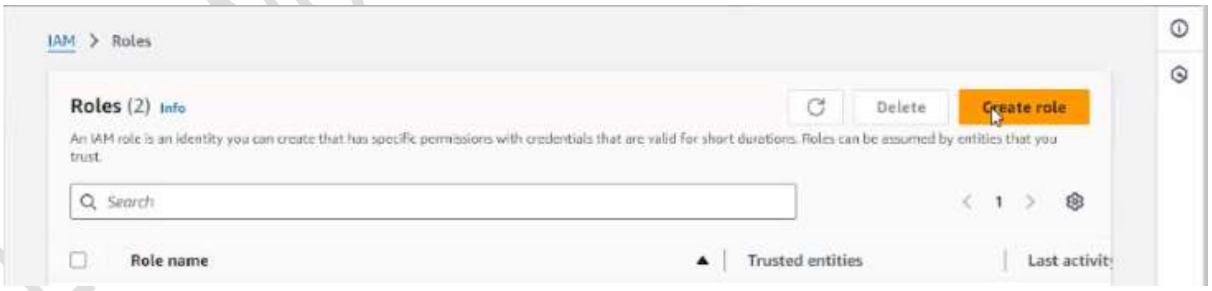
- Note:- a prompt is very important if your prompt is not right then it not work

----

- Now run one interesting demo we launch one Ec2 instance from this tool
- First we create an IAM role for the EC2 service

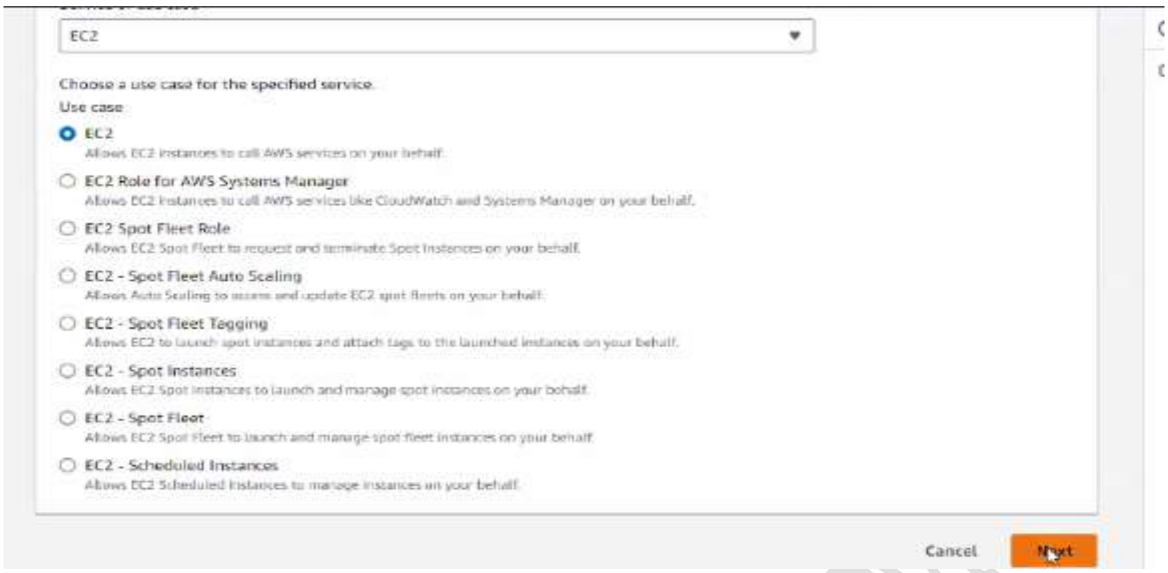


- After that click on the Create role



An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

- After that search for the EC2


EC2

Choose a use case for the specified service.

Use case

[✓ EC2]
Allows EC2 instances to call AWS services on your behalf.

EC2 Role for AWS Systems Manager
Allows EC2 Instances to call AWS services like CloudWatch and Systems Manager on your behalf.

EC2 Spot Fleet Role
Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.

EC2 - Spot Fleet Auto Scaling
Allows Auto Scaling to access and update EC2 spot fleets on your behalf.

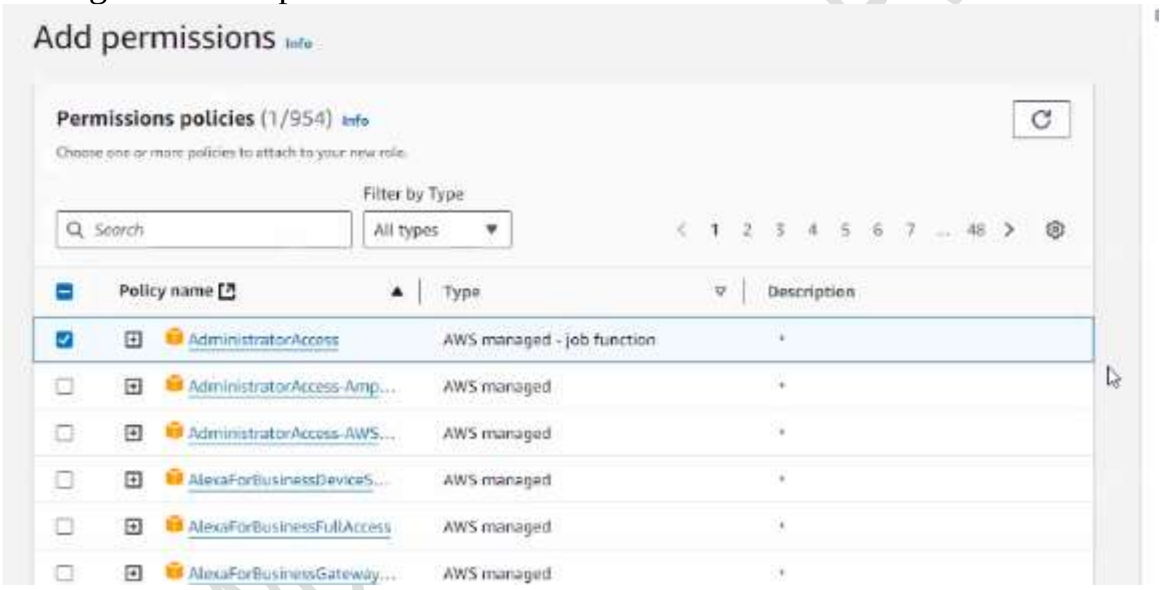
EC2 - Spot Fleet Tagging
Allows EC2 to launch spot instances and attach tags to the launched instances on your behalf.

EC2 - Spot Instances
Allows EC2 Spot Instances to launch and manage spot instances on your behalf.

EC2 - Spot Fleet
Allows EC2 Spot Fleet to launch and manage spot fleet instances on your behalf.

EC2 - Scheduled Instances
Allows EC2 Scheduled Instances to manage instances on your behalf.

- Then give the full power


Add permissions Info

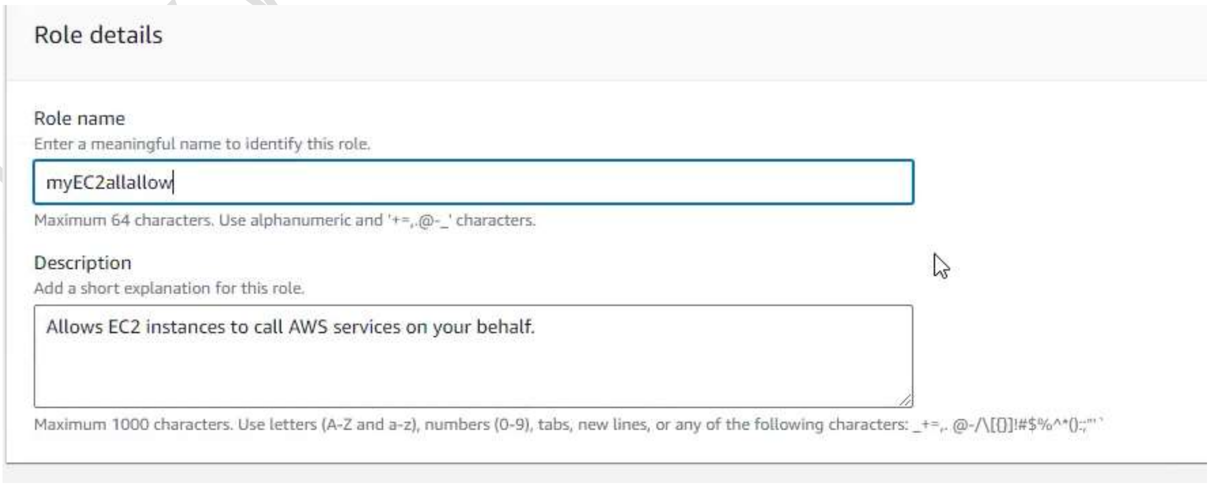
Permissions policies (1/954) Info

Choose one or more policies to attach to your new role.

Filter by Type: All types

	Policy name	Type	Description
✓	AdministratorAccess	AWS managed - job function	-
	AdministratorAccess-Amp...	AWS managed	-
	AdministratorAccess-AWS...	AWS managed	-
	AlexaForBusinessDeviceS...	AWS managed	-
	AlexaForBusinessFullAccess	AWS managed	-
	AlexaForBusinessGateway...	AWS managed	-

- Give role name


Role details

Role name
Enter a meaningful name to identify this role.

myEC2allallow

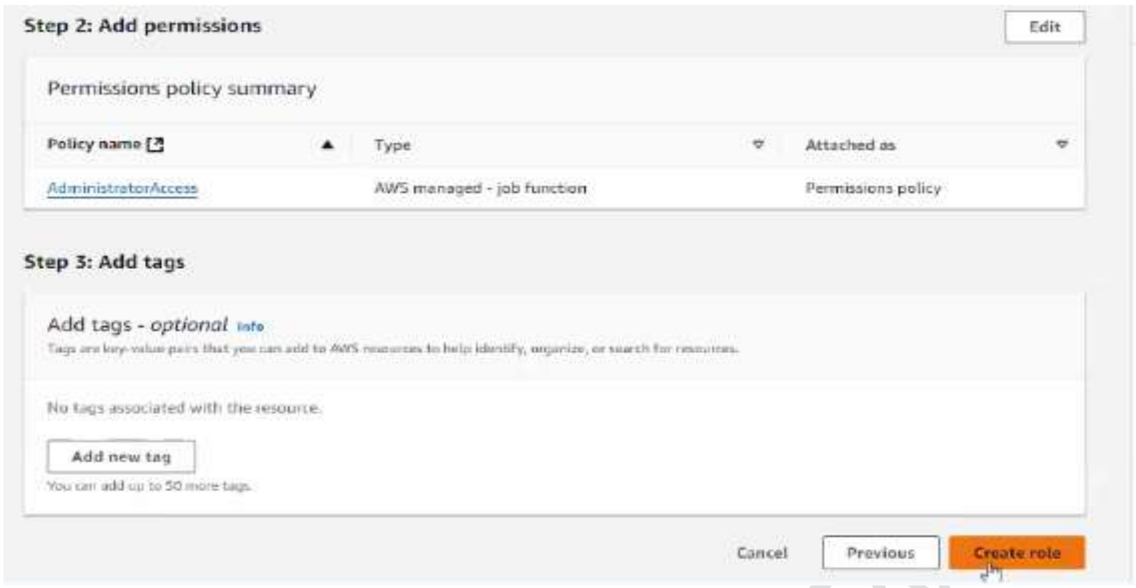
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

Description
Add a short explanation for this role.

Allows EC2 instances to call AWS services on your behalf.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,.@-/\[{}]!#$%^*();"' `

- Click on the Create role


Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
AdministratorAccess	AWS managed - job function	Permissions policy

Step 3: Add tags

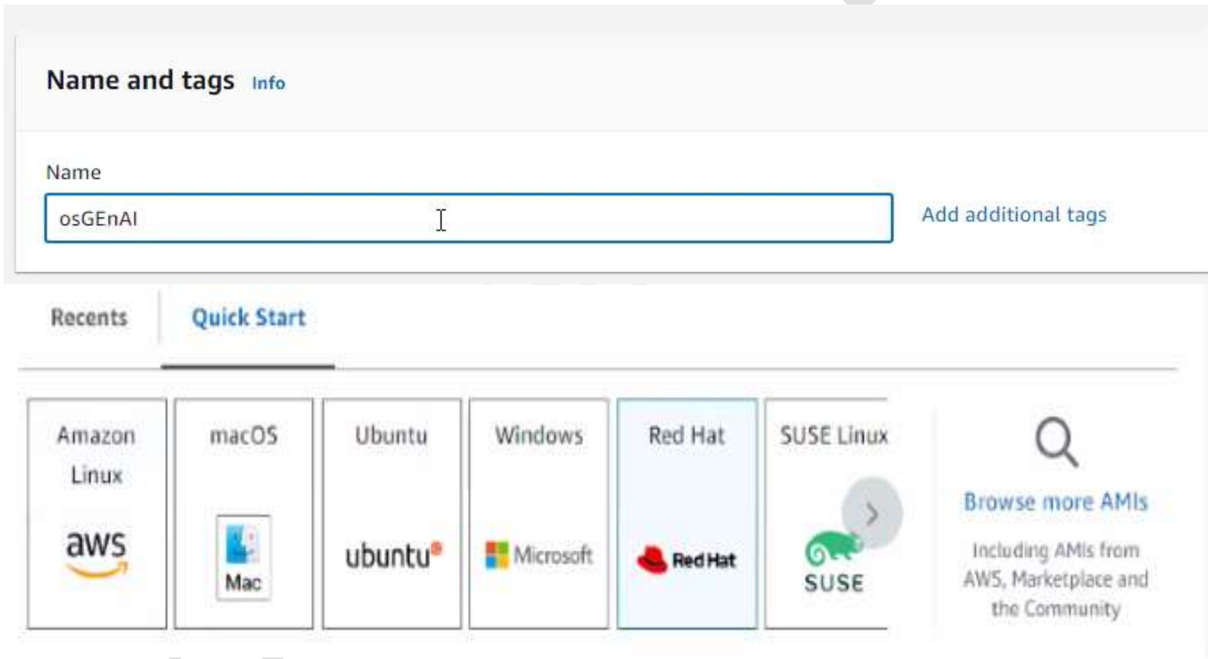
Add tags - *optional*

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

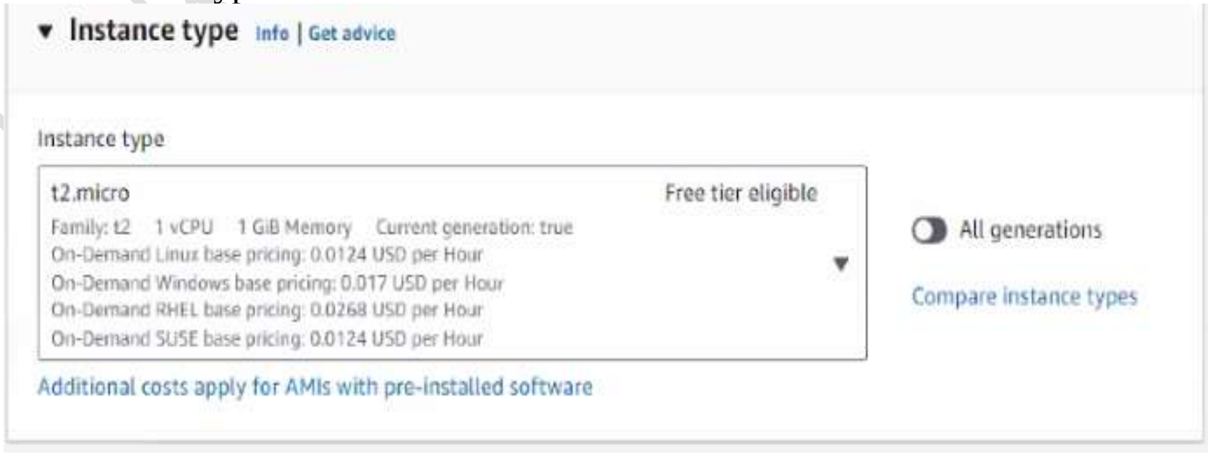
You can add up to 50 more tags.

- Now we need to launch one Ec2 instance with RedHat Linux


Name and tags

Name: osGEnAI

- Use instance type t2.micro

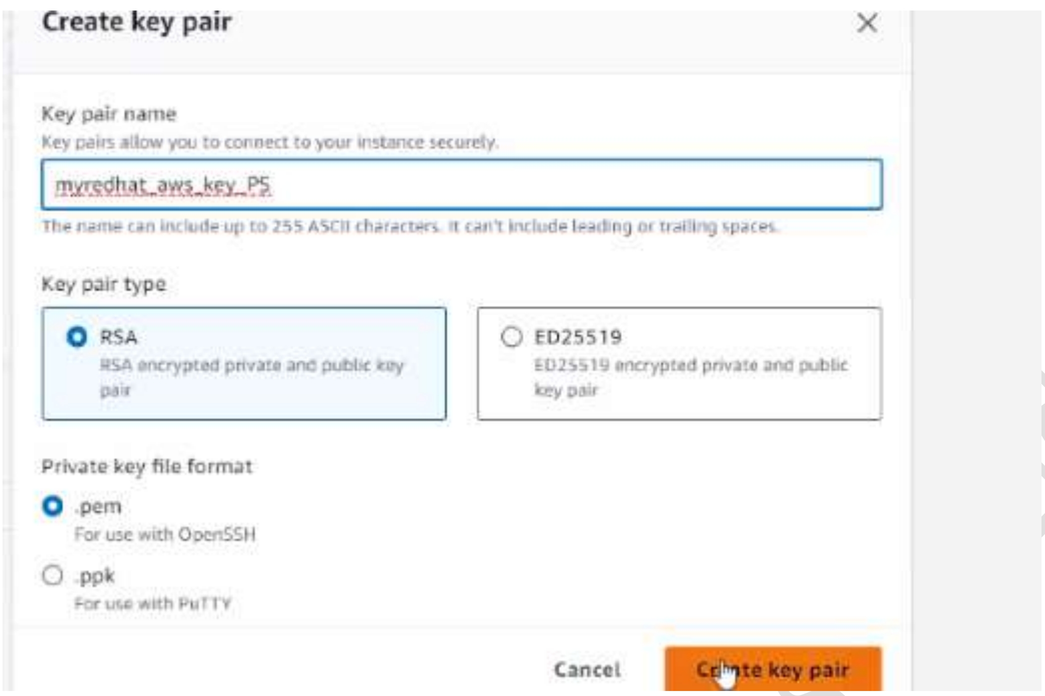

Instance type

Instance type

```
t2.micro                                                   Free tier eligible
Family: t2    1 vCPU    1 GiB Memory    Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour
On-Demand Windows base pricing: 0.017 USD per Hour
On-Demand RHEL base pricing: 0.0268 USD per Hour
On-Demand SUSE base pricing: 0.0124 USD per Hour
```

Additional costs apply for AMIs with pre-installed software

- After that create a key pair



Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

myredhat_aws_key_PS

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

[✓ RSA] RSA encrypted private and public key pair

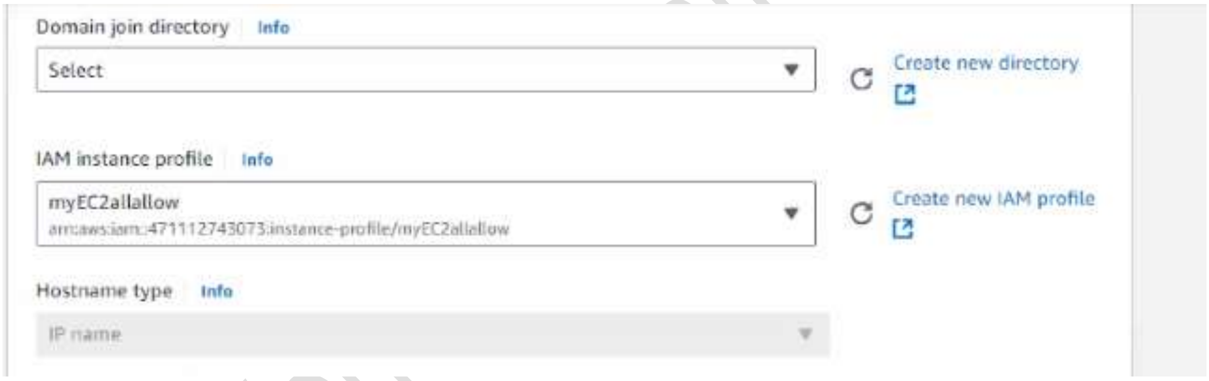
ED25519 ED25519 encrypted private and public key pair

Private key file format

[✓ .pem] For use with OpenSSH

.ppk For use with PuTTY

- After that click on the advance
- Then select the ec2 role



Domain join directory Info

Select

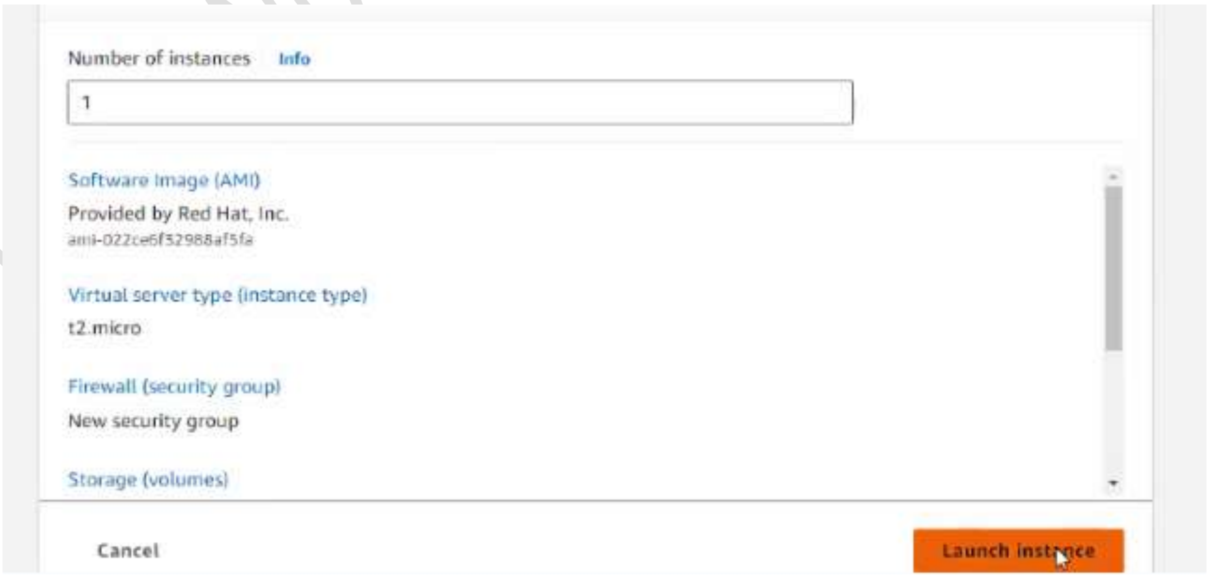
IAM instance profile Info

myEC2allallow
arn:aws:iam::471112743073:instance-profile/myEC2allallow

Hostname type Info

IP name

- Then allow the firewall and click on the launch instance



Number of instances Info

1

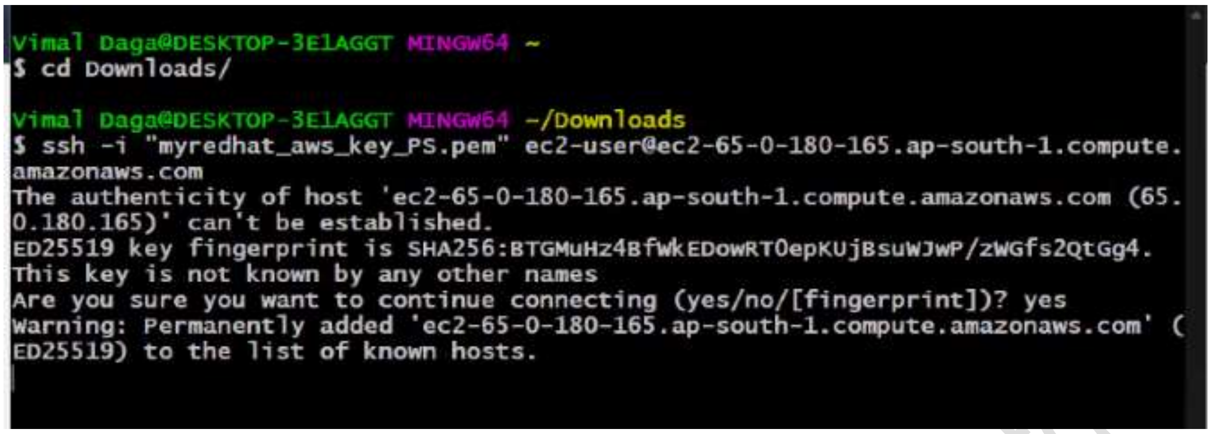
Software Image (AMI)
Provided by Red Hat, Inc.
ami-022ce6f32988af5fa

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

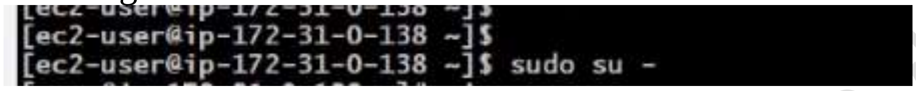
Storage (volumes)

- After that connect with git bash with ssh command



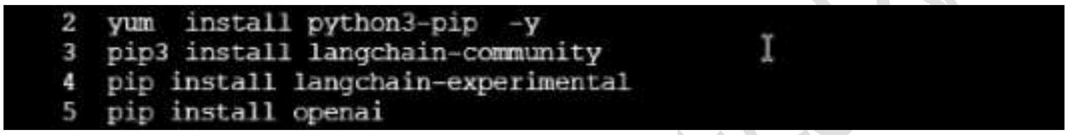
```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~
$ cd Downloads/

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Downloads
$ ssh -i "myredhat_aws_key_PS.pem" ec2-user@ec2-65-0-180-165.ap-south-1.compute.
amazonaws.com
The authenticity of host 'ec2-65-0-180-165.ap-south-1.compute.amazonaws.com (65.
0.180.165)' can't be established.
ED25519 key fingerprint is SHA256:BTGMuHz4BfWkEDowRT0epKUjBsuWJwP/zWGfs2QtGg4.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-65-0-180-165.ap-south-1.compute.amazonaws.com' (
ED25519) to the list of known hosts.
```

- Then login with the root user



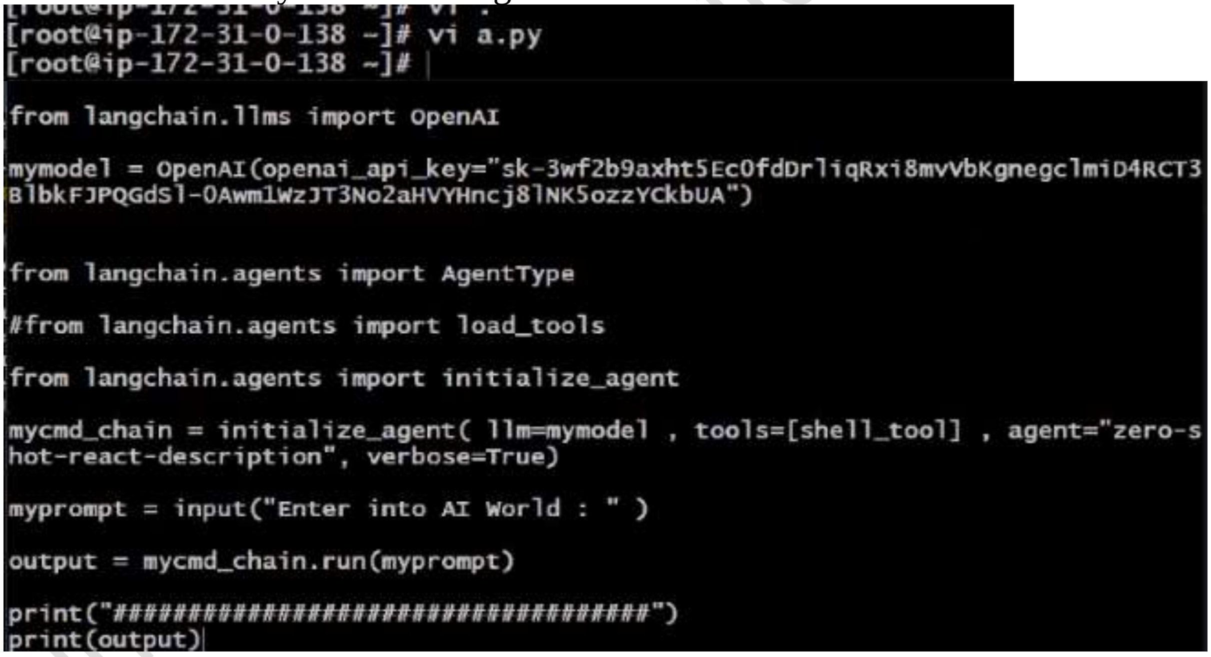
```
[ec2-user@ip-172-31-0-138 ~]$
[ec2-user@ip-172-31-0-138 ~]$ sudo su -
```

- After that install the below libraries



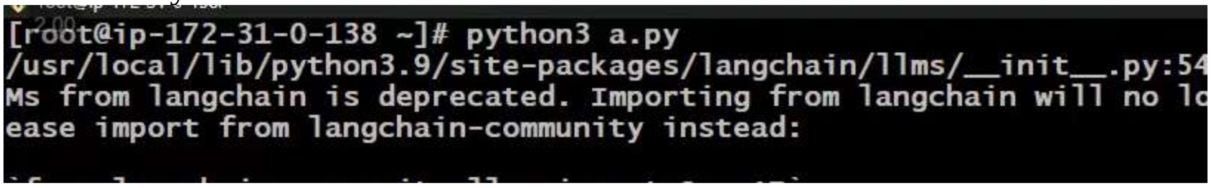
```
yum  install python3-pip  -y
pip3 install langchain-community
pip install langchain-experimental
pip install openai
```

- After that create Python file and give LLM code



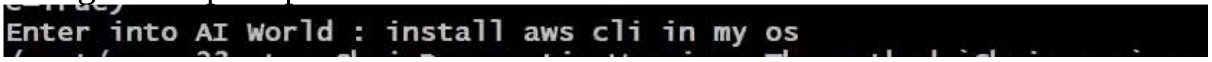
```
[root@ip-172-31-0-138 ~]# vi a.py
[root@ip-172-31-0-138 ~]#

from langchain.llms import OpenAI

mymodel = OpenAI(openai_api_key="sk-3wf2b9axht5Ec0fdDrliqRxi8mvVbKgnegclmiD4RCT3
BlbkFJPQGdSl-0Awm1WzJT3No2aHVYHncj8lNK5ozzYCkbUA")


from langchain.agents import AgentType

#from langchain.agents import load_tools

from langchain.agents import initialize_agent

mycmd_chain = initialize_agent( llm=mymodel , tools=[shell_tool] , agent="zero-s
hot-react-description", verbose=True)

myprompt = input("Enter into AI World : " )

output = mycmd_chain.run(myprompt)

print("####################################")
print(output)
```

- After that run the below command
  #curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
  unzip awscliv2.zip
  sudo ./aws/install
- Then run Python code



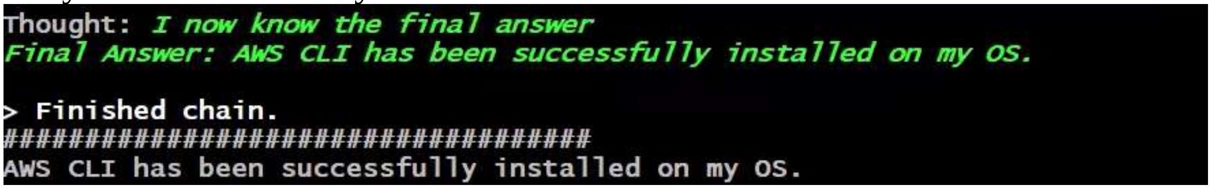
```
[root@ip-172-31-0-138 ~]# python3 a.py
/usr/local/lib/python3.9/site-packages/langchain/llms/__init__.py:54
Ms from langchain is deprecated. Importing from langchain will no lo
ease import from langchain-community instead:
```

- And give the prompt



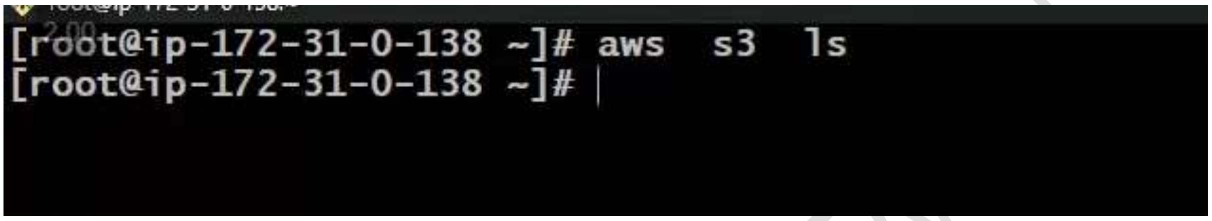
```
Enter into AI World : install aws cli in my os
```

- They install successfully



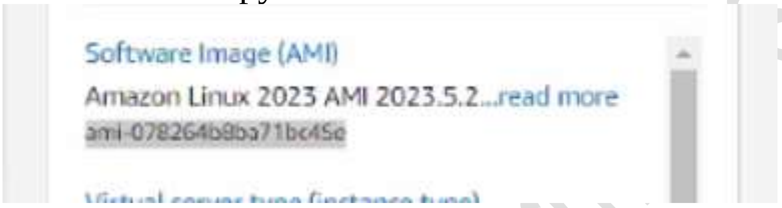
```
Thought: I now know the final answer
Final Answer: AWS CLI has been successfully installed on my OS.

> Finished chain.
########################################
AWS CLI has been successfully installed on my OS.
```

- To check whether aws cli install or not we run the below command



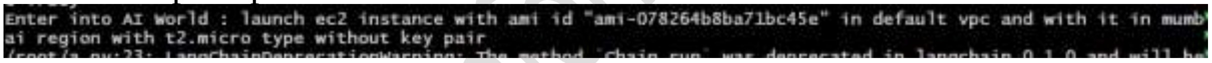
```
[root@ip-172-31-0-138 ~]# aws  s3  ls
[root@ip-172-31-0-138 ~]#
```

- Now ready to launch the Ec2 instance but we need to give a better prompt
- For that first we copy the latest AMI id



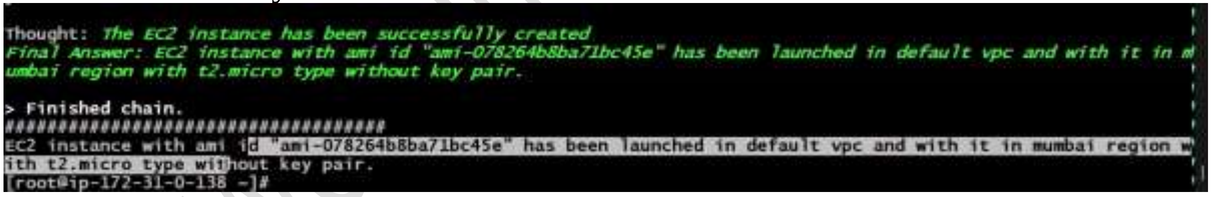
Software Image (AMI)
Amazon Linux 2023 AMI 2023.5.2...read more
ami-078264b8ba71bc45e

- Use below prompt

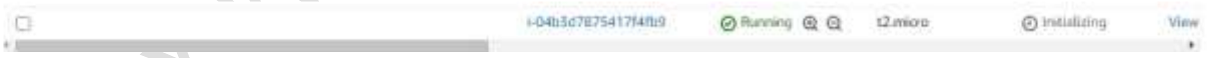


```
Enter into AI World : launch ec2 instance with ami id "ami-078264b8ba71bc45e" in default vpc and with it in mumb
ai region with t2.micro type without key pair
```

- As we can see they launch one new instance



```
Thought: The EC2 instance has been successfully created
Final Answer: EC2 instance with ami id "ami-078264b8ba71bc45e" has been launched in default vpc and with it in m
umbai region with t2.micro type without key pair.

> Finished chain.
########################################
EC2 instance with ami id "ami-078264b8ba71bc45e" has been launched in default vpc and with it in mumbai region w
ith t2.micro type without key pair.
[root@ip-172-31-0-138 ~]#
```



i-04b3d7875417f4fb9   Running   t2.micro   Initializing   View