**Deployment Documentation**

1. **GitHub Repo**

Initial step - clone repository

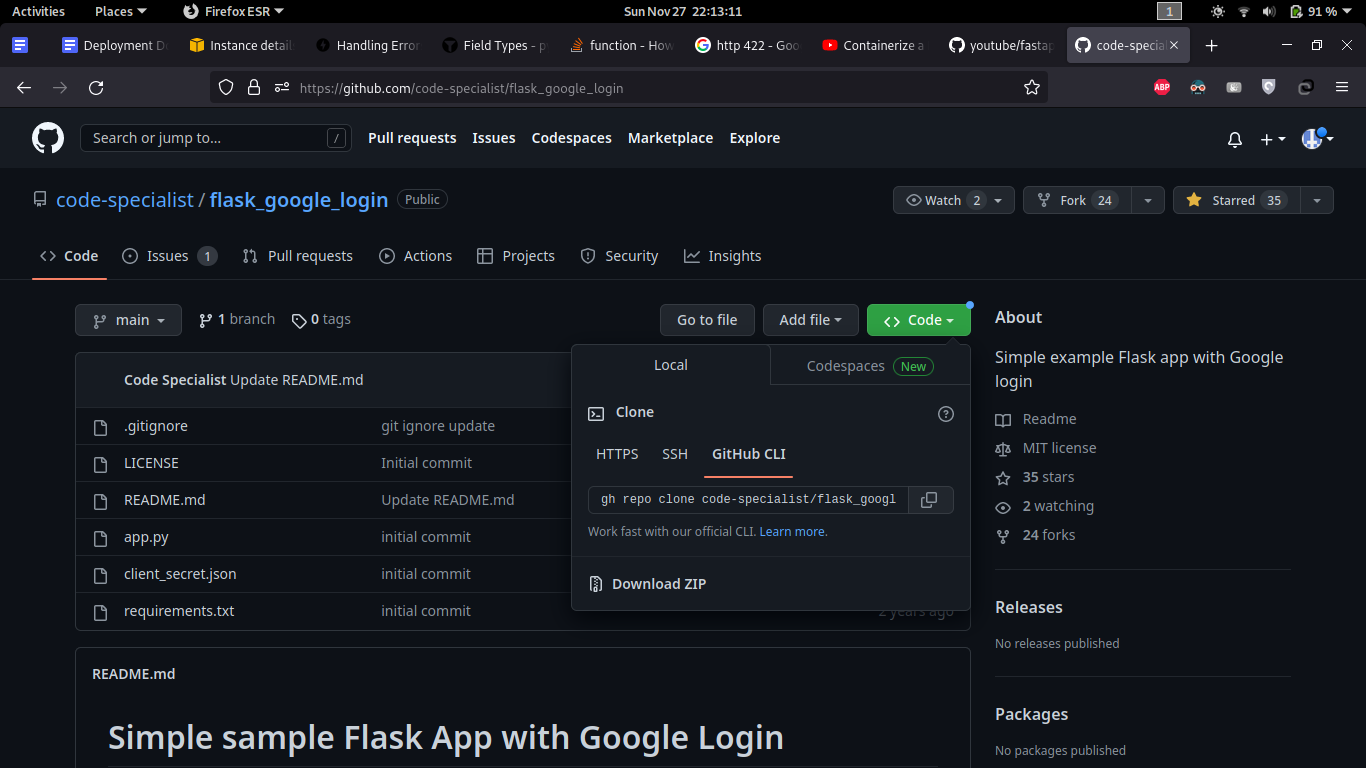
**>> sudo apt update**

**>> sudo apt install git**

**>> sudo apt install git-lfs**  # to clone large files

**>> git clone <repo link>**

**# sample link**



To add ssh access in a new server to clone private repo

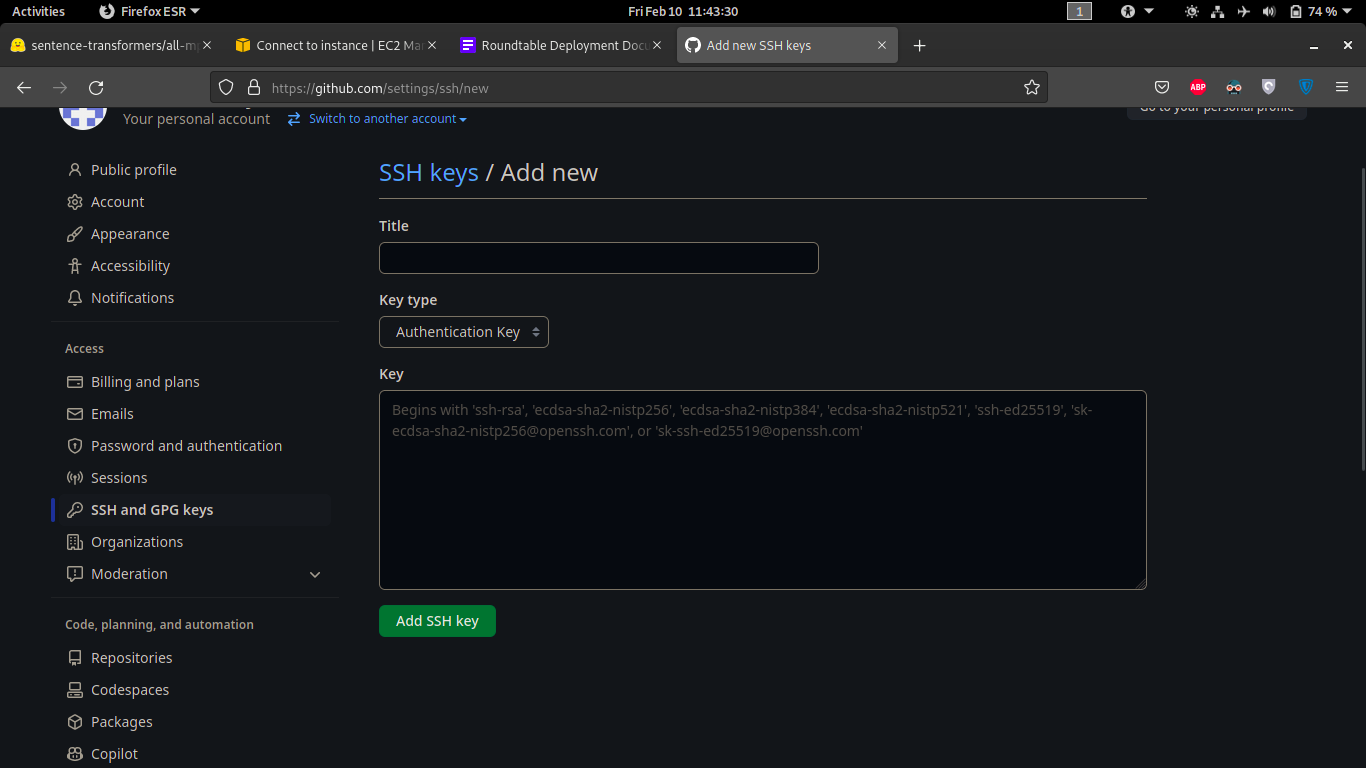
**>> ssh-keygen -t ed25519 -C "example@gmail.com"**

**>> eval "$(ssh-agent -s)"**

**>> ssh-add ~/.ssh/id\_ed25519**

**>> cat ~/.ssh/id\_ed25519.pub**

Copy this public key and paste it in **SSH and GPG keys** section in your Github profile



1. **Install Dependencies**

Run to install all dependent packages

**>> sudo apt update**

**>> sudo apt install python3-pip**

**>> sudo apt install uvicorn**

**>> pip3 install -r requirements.txt**

For NVIDIA GPU enabled servers install this additionally

NVIDIA driver

**>> sudo apt install nvidia-driver-525 nvidia-dkms-525**

**>> sudo apt install nvidia-kernel-common-525**

NVIDIA TOOLKIT for docker

**>> curl https://get.docker.com | sh \**

**&& sudo systemctl --now enable docker**

**>> sudo apt-get install -y nvidia-container-toolkit**

**>> sudo nvidia-ctk runtime configure --runtime=docker**

**>> sudo systemctl restart docker**

1. **Execution**
   1. **Setting up environment variable**

Create .env file in the same directory specify the following variables for each module

**Register/Login Module**

**<AUTH\_KEY>** - secret phrase to encode and decode tokens

**Hash Module**

**<AUTH\_KEY>** - secret phrase to encode and decode tokens. Should be same as phrase given in Register module

**Summary Module**

**<AUTH\_KEY>** - secret phrase to encode and decode tokens. Should be the same phrase given in Register module

**Topic Module**

**<AUTH\_KEY>** - secret phrase to encode and decode tokens. Should be the same phrase given in Register module

**Keyword Module**

**<AUTH\_KEY>** - secret phrase to encode and decode tokens. Should be the same phrase given in Register module

* 1. **Downloading required files**
     1. **From huggingface**
        1. **Summary module**

Go to roundtable -> roundtable\_dockerize -> sumamry\_api -> summarization -> files

Then open terminal and clone the repo from huggingface - including the large files

**>>** git clone https://huggingface.co/facebook/bart-large-cnn

* + - 1. **All other module**

Go to roundtable -> roundtable\_dockerize -> xxxxx\_api -> topic/hashtag/keyword-> files

Then open terminal and clone the repo from huggingface - including the large files

>> git clone https://huggingface.co/sentence-transformers/all-mpnet-base-v2

* + 1. **From Google Drive (Preferred Method)**
       1. **Summary module**

Go to roundtable -> roundtable\_dockerize -> sumamry\_api -> summarization -> files

Download the files and paste in the folder

Link - xxxx

* + - 1. **All other modules**

Go to roundtable -> roundtable\_dockerize -> xxxxx\_api -> topic/hashtag/keyword-> files

Download the files and paste in the folder

Link - xxxxx

* 1. **Config file**

Each module has its own **config\_xxx.ini** file to store parameter values. Changing the values will reflect in module execution.

* 1. **Server**

Run **reg\_login.py**  file to start the authentication module, Once the server is up and running you can use <http://127.0.0.1:8000/><endpoint> to access registration/login endpoint. Similarly run **xxxxx\_fastapi**.py to start the prediction module and use [http://127.0.0.1:8001/](http://127.0.0.1:8000/)<endpoint> to access the endpoint

**>> uvicorn <reg\_login module filename\_without\_py\_extension>:app - -port 8000 - -reload**

**>> uvicorn <prediction filename\_without\_py\_extension>:app - -port 8001 - -reload**

**Note** : port should be the same as mentioned orelse the client script will fail to communicate

* 1. **Client Script**

You can also access the endpoints using the client script. Run **client\_xxx** prefix file to automate the endpoint access. You can do prediction by feeding the .txt file to the client script.

**>> Python3 <filename\_with\_extension>**

* + 1. **Summary Client Script**

Place text file named **sample\_transcript.txt** in **roundtable\_dockerize** folder - file should have the transcript data.

* + 1. **Hashtag Client Script**

Place text file named **sample\_summary.txt** in **roundtable\_dockerize** folder - file should have the summary data.

* + 1. **Keyword Client Script**

Place text file named **sample\_transcript.txt** in **roundtable\_dockerize** folder - file should have the transcript data.

Place text file named **sample\_summary.txt** in **roundtable\_dockerize** folder - file should have the summary data.

* + 1. **Topic Client Script**

Place text file named **sample\_transcript.txt** in **roundtable\_dockerize** folder - file should have the transcript data.

1. **Dockerize**
   1. **Installation**

Dockerizing the application can help to run it in different environments irrespective of operating system, versions of packages installed. Install the docker using the following commands in linux operating system

**>> sudo apt update -qq**

**>> sudo apt install apt-transport-https ca-certificates curl software-properties-common -qq**

**>> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -**

**>> sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"**

**>> sudo apt update -qq**

**>> sudo apt install docker-ce**

* 1. **Dockerfile**

File specifies how the image has to be created and run on execution.

**FROM** - base image to run our module

**COPY** - working directory source path in local system

**RUN** - run requirements.txt file to install all dependent packages

**EXPOSE** - port for app to run

**WORDIR** - working directory path inside docker image

**CMD** - command to run the application after starting the container

* 1. **Building docker image**

Build the docker images as specified in Dockerfile

**>> sudo docker build -t <image\_name>:<version\_tag> <source\_directory>**

**for egs: sudo docker build -t hash:1.0 .**

We have to create two containers for each module - register and prediction.

* 1. **Running docker container**

Run both containers in different ports.

**>> sudo docker run -p <port>:<port> -d <img\_name>:<version\_tag>**

for egs:

**sudo docker run -p 8000:8000 -d hashtag:1.0**

**sudo docker run -p 8001:8001 -d registry:1.0**

gpu enabled container:

**>> sudo docker run --rm -p 8001:8001 --gpus all -d summary:1.0**

1. **Deploying in AWS**

Choose the apt configuration available in EC2 instances and transfer the file from local machine via SCP

You can either transfer model files or clone directly from huggingface site and build the docker image in EC2 instance. We need to open a custom TCP port 8000 and 8001 in security settings to allow access through that channel. After enabling, type the following command in case you face any issue in accessing the ports.

**>> sudo ufw allow 8000/tcp**

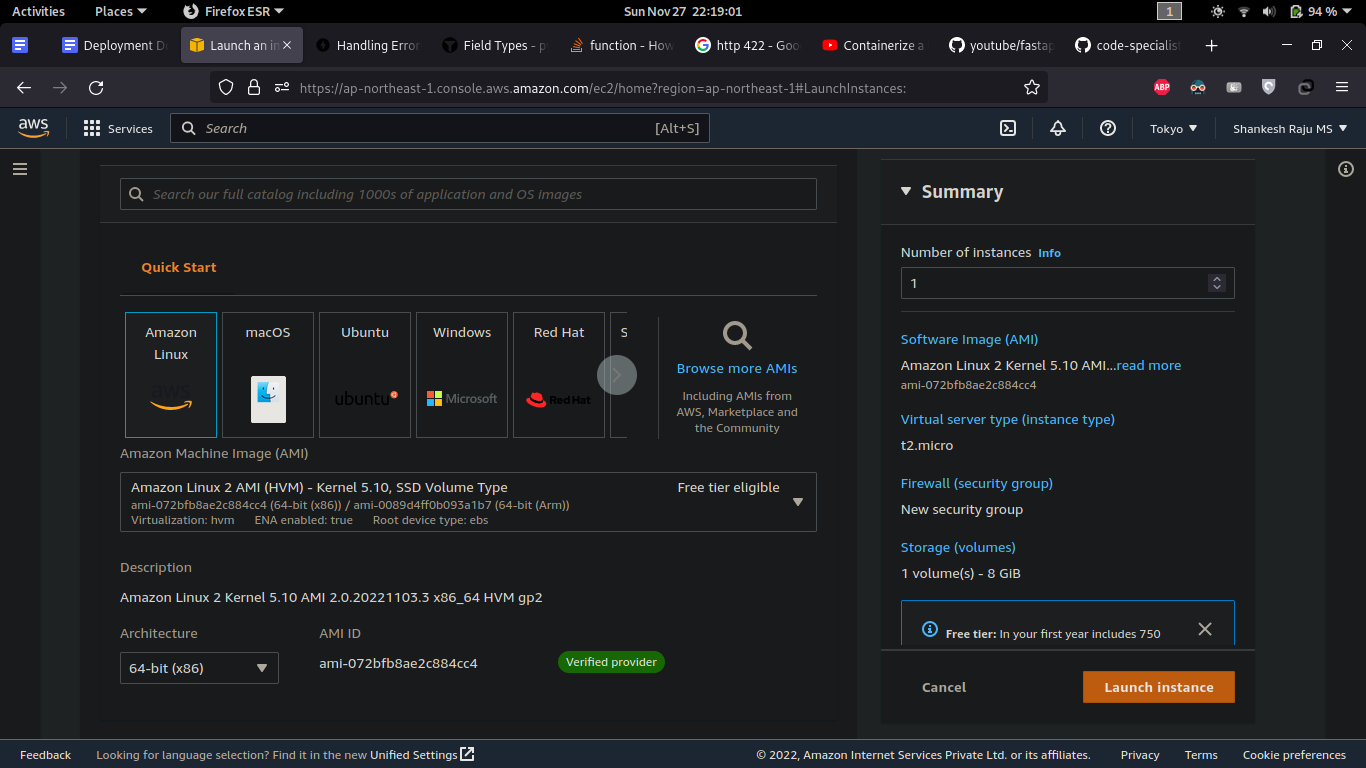
**>> sudo ufw allow 8001/tcp**

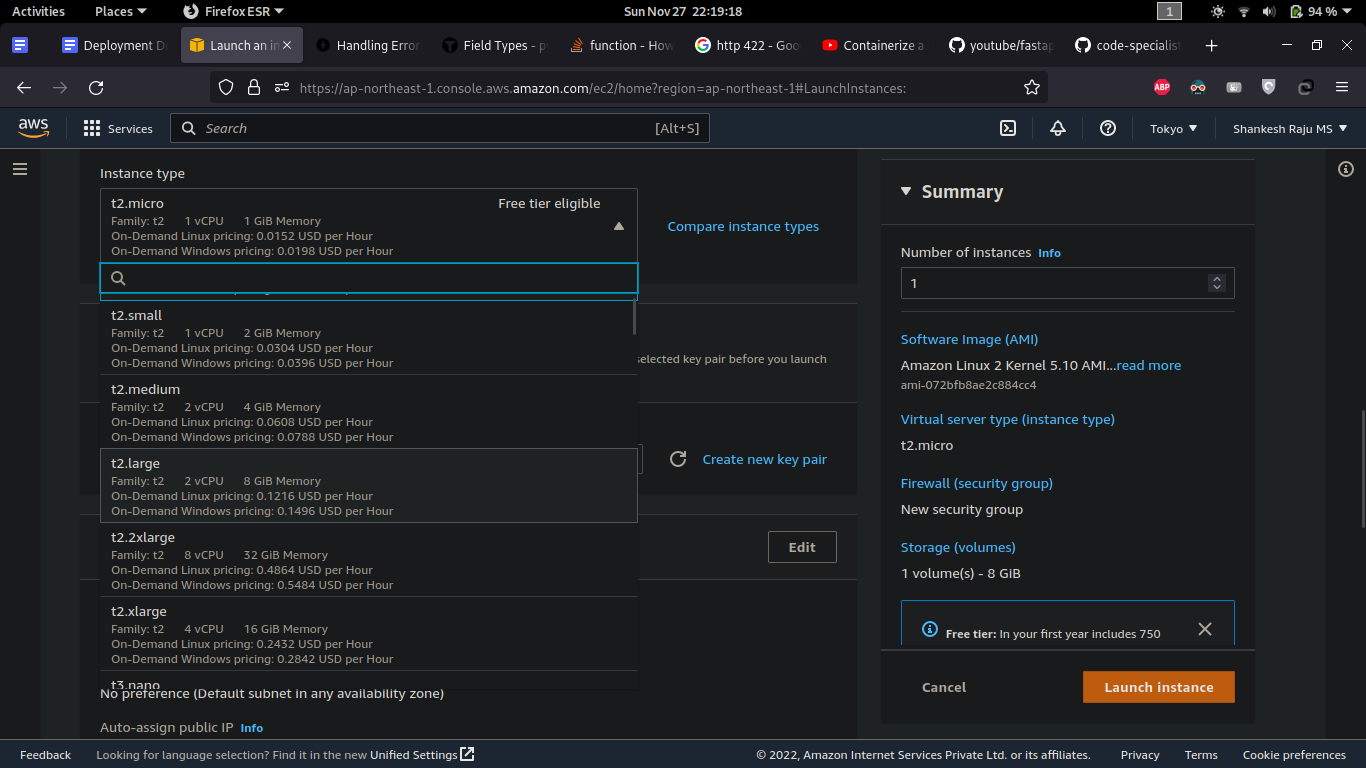
To transfer files from local system to AWS instance

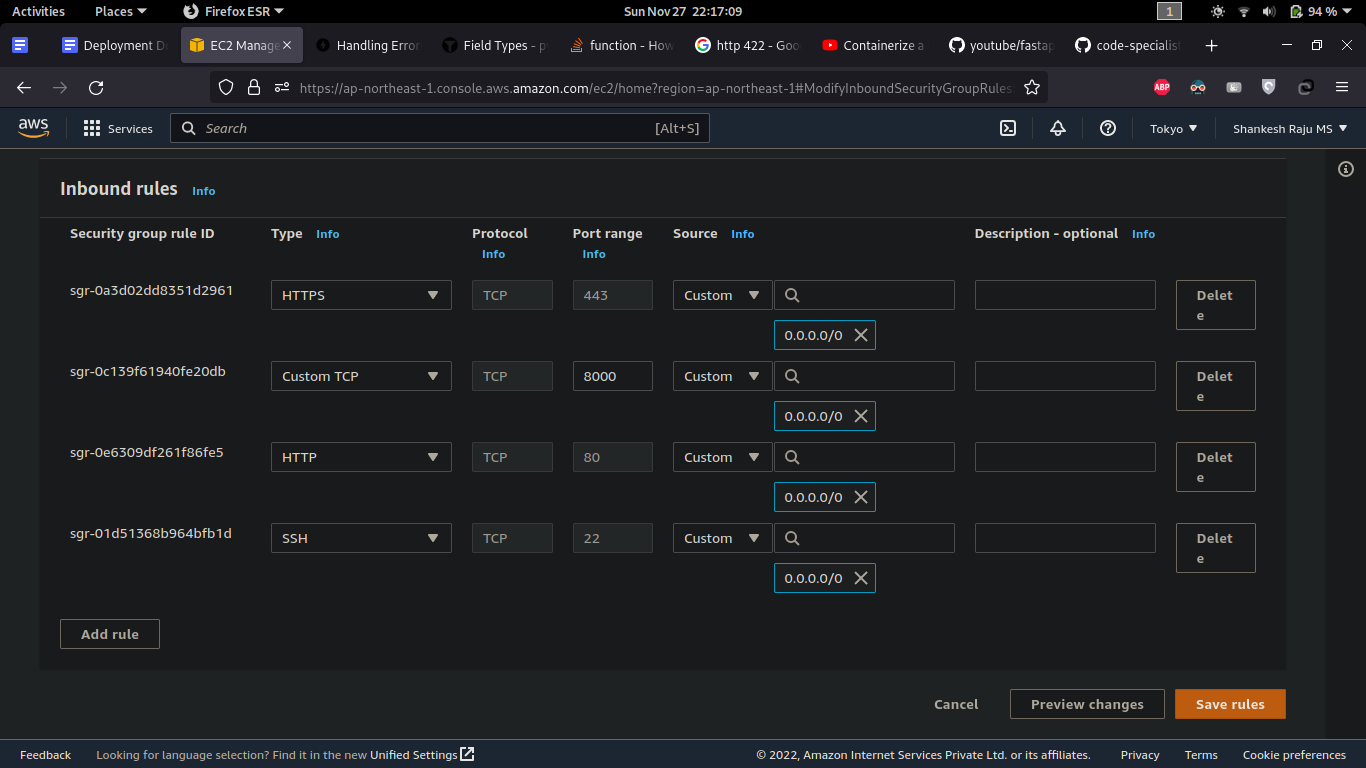
**scp -i <private key.pem> <localpath\_tocopy> username@ec2address:<destination\_path>**

Egs:

**scp -i <token.pem> <source file location>** [**ubuntu@ec2-xx-xx-xx-xx.ap-south-1.compute.amazonaws.com**](mailto:ubuntu@ec2-xx-xx-xx-xx.ap-south-1.compute.amazonaws.com)**:<destination path>**

****

****

****

Once the port is enabled, run the docker and access the endpoints as mentioned below

**http://<EC2\_instance\_public\_ip>:8001/<endpoints> - prediction module**

**http://<EC2\_instance\_public\_ip>:8000/<endpoints> - register\_login module**

**END**