1. Create a Conda environment:

If you don't have anaconda download from here [Link](https://www.anaconda.com/download/success)

| conda create -p <env\_name> python=3.10 -y |
| --- |

1. Activate your conda environment

| Conda activate <env\_path> |
| --- |

If activating on bash terminal use this command:

| Source activate ./<env\_name>  Conda activate <env\_path> |
| --- |

1. Create a requirement.txt file and install it.

| pip install -r requirements.txt |
| --- |

1. Create a .env file for keeping your environment variable.
2. Use setup.py for installing your local package.

| <either mention -e . inside your requirements.txt Or run python setup.py install > |
| --- |

1. Checkout here with full video of end to end project setup [Link](https://youtu.be/iCCdD4ohNi0)

**AWS Deployment:**

1. Push your entire code to github
2. Login to your AWS account [Link](https://aws.amazon.com/console/)
3. Launch your EC2 Instance
4. Configure your EC2 Instance
5. Command for configuring EC2 Instance.
6. sudo apt-get update and sudo apt update are used to update the package index on a Debian-based system like Ubuntu, but they are slightly different in terms of the tools they use and their functionality:

| sudo apt-get update |
| --- |

This command uses apt-get, the traditional package management tool.

| sudo apt update -y |
| --- |

This command uses apt, a newer, more user-friendly command-line interface for the APT package management system.

Install required tools

| sudo apt install git curl unzip tar make sudo vim wget -y |
| --- |

Clone git repository

| git clone <.git url> |
| --- |

Create a .env file there

| touch .env |
| --- |

Open file in VI editor

| vi .env |
| --- |

| Press insert and Mention env variable then press esc for saving and write :wq for exit.  cat .env #for checking the value |
| --- |

For installing python and pip here is a command:

| sudo apt install python3-pip |
| --- |

Then install the requirements.txt

The --break-system-packages flag in pip allows to override the externally-managed-environment error and install Python packages system-wide.

| pip3 install -r requirements.txt  pip3 install -r requirements.txt --break-system-packages |
| --- |

The --break-system-packages flag in pip allows to override the externally-managed-environment error and install Python packages system-wide.

pip install package\_name --break-system-packages

Then run your application

| python3 app.py |
| --- |

Configure your inbound rule:

1. Go inside the security
2. Click on security group
3. Configure your inbound rule with certain values

| Port 5000 0.0.0.0/0 for anywhere traffic TCP/IP protocol |
| --- |

Save it and now run it again.