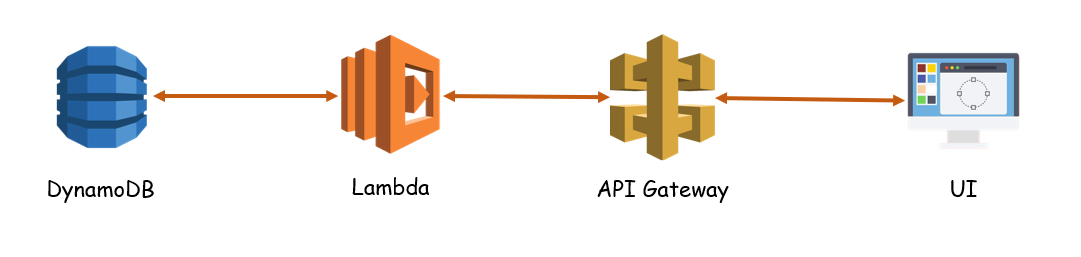
**Build a Plasma Donor App with AWS Serverless Computing**

During the COVID 19 crisis, the requirement of plasma became high and the donor count being low. Saving the donor information and helping the need by notifying the current donors would be a helping hand. In regard to the problem faced, an application is to be built which would take the donor details store it and inform them upon a request.

Serverless computing is the current trend of software application development. Micro services are a popular new approach for building maintainable, scalable, cloud-based applications. AWS is the perfect platform for hosting micro-services. In this project we will be building a plasma donor app with AWS services like lambda functions, API gateway and DynamoDB.

**Architecture:**



**Learning Outcomes:**

By the end of this project:

* You’ll be able to work DynamoDB, lambda functions and API Gateway.
* Build a flask application which will take the user inputs, update the DynamoDB and notify the user upon request.

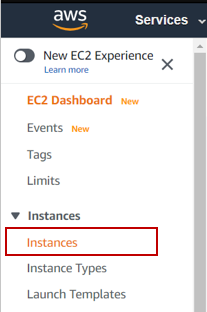
**Project Work Flow:**

* User interacts with the application.
* Registers by giving the details as a donor.
* Database will have all the details and if a user posts a request then the concerned blood group donors will get notified about it.

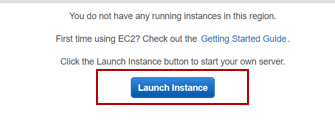
**Task 1: Deploy the application using EC2**

**Activity 1:** Create an EC2 instance

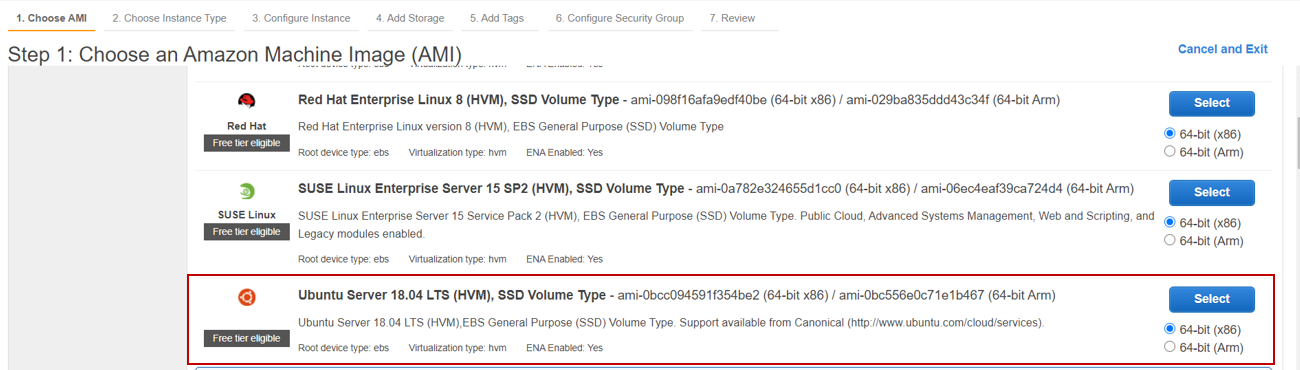
**Step 1:** Under services in AWS Console, search for “EC2” and open it > click on “Instances” (you can find it in the left pane).



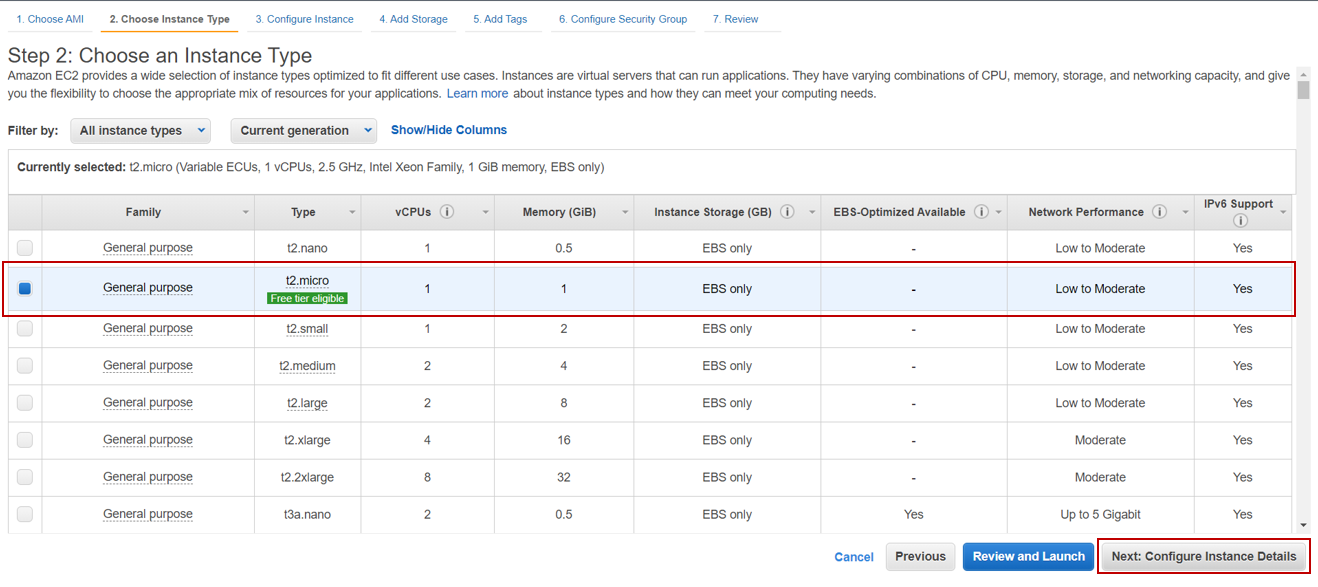
**Step 2:** Click on “Launch Instance”



**Step 3:** Select the ubuntu server 18.04 > click on select by letting the 64-bit radio button selected.

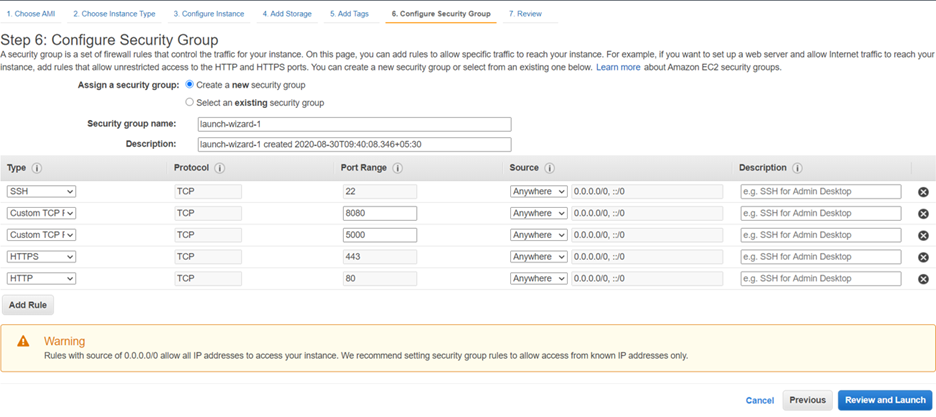


**Step 4:** By default the t2.micro will be selected, check it and click on “Next”

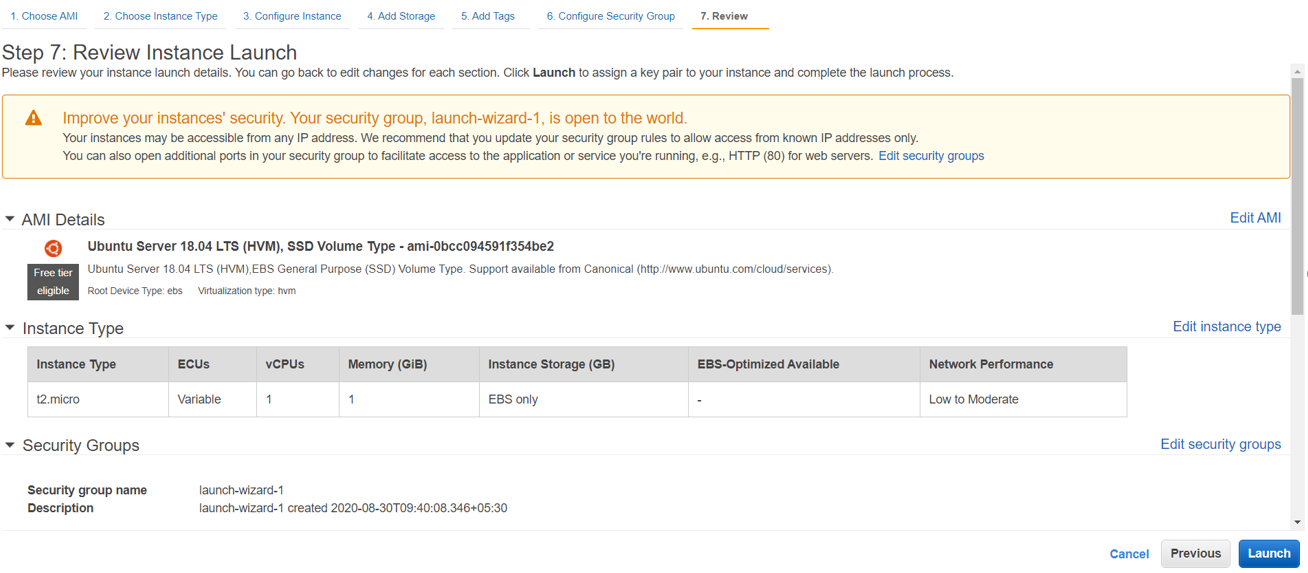


**Step 5:** Click Next > Next > Next > Next. You’ll land up in 6th step (Configure Security Group)

**Step 6:** While setting the required ports, make sure you follow according to the given image below > click on Review and Launch

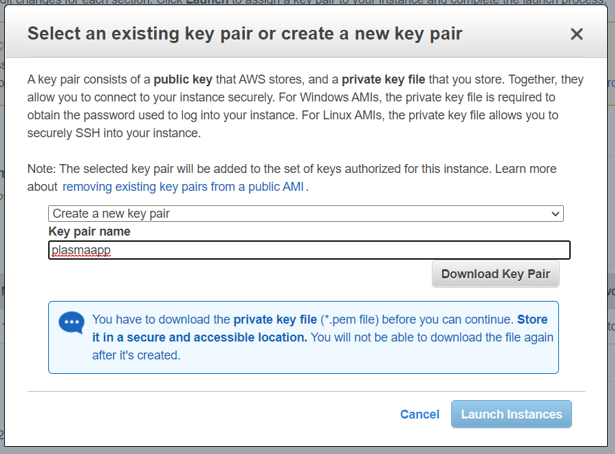


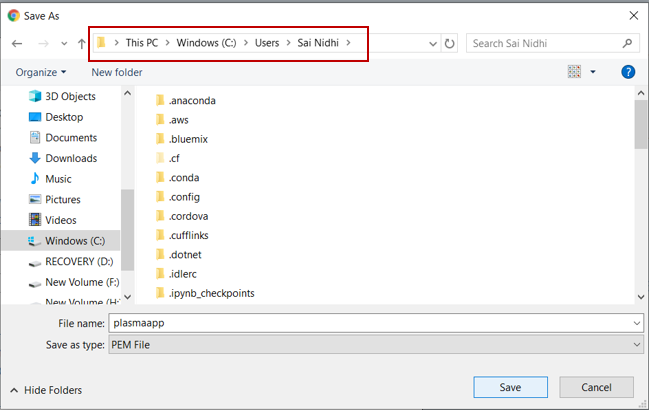
**Step 7:** Click on Launch



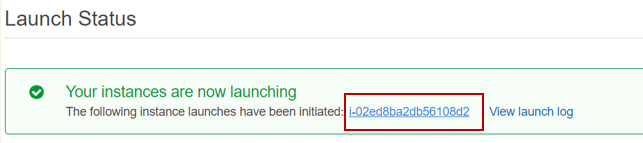
**Step 8:** It’ll ask you select a key pair. This key pair will be used to launch the virtual machine (ubuntu instance created). Choose “Create a new key pair” > give the key pair name (plasmaapp) > click on “Download Key Pair”.

**Note:** The downloaded .pem file (key pair), should be in the C drive (C:/users/username). If it is present in some other location, change the location of the file. Do not rename the file.

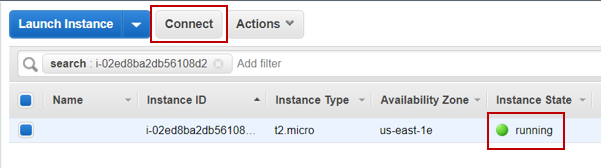




**Step 9:** After the file is downloaded click on Launch Instances. When the instance is launching, click on the link given.

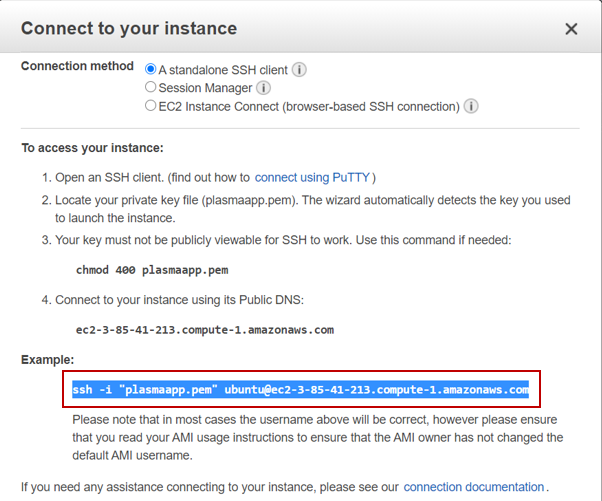


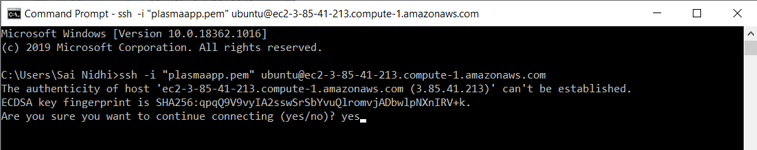
**Step 10:** When the instance status is running > click on Connect



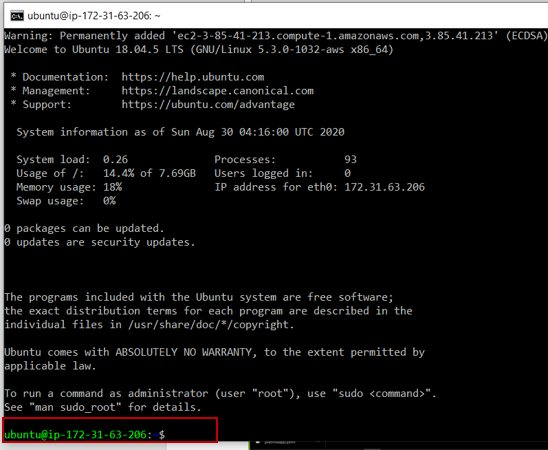
**Step 11:** Copy the link which is present under example and paste it in command prompt > when asked type “yes”

**Note:** If SSH is not present in the system, you need to install putty and then run the instance.





**Step 12:** If you get the screen as the below image, your virtual machine is launched.



**Activity 2:** Deploy the app

**Step 1:** To deploy the flask application execute the below commands:

sudo apt-get update

sudo apt-get install apache2

sudo apt-get install libapache2-mod-wsgi

sudo apt-get install python3-pip

sudo pip3 install flask

sudo pip3 install requests

sudo pip3 install boto3

sudo apt-get install -y unzip

**Step 2:** After all the required installations are done, clone the application.

**Note:** Upload your application to GitHub and clone it in your virtual machine

**Command:** git clone link



**Step 3:** Give the command ls to know the file name

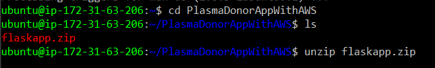
**Step 4:** Navigate into that folder

**Command:** cd filename

**Step 5:** Unzip the zipped folder present

**Command:** unzip filename.zip



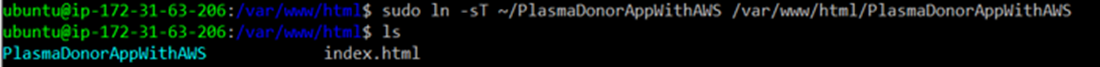


After unzipping the folder

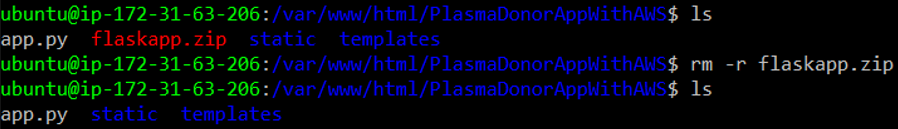


**Step 6:** Link the folder to the site-root defined in apache’s configuration (/var/www/html) by default.

**Command:** sudo ln -sT ~/foldername /var/www/html/foldername



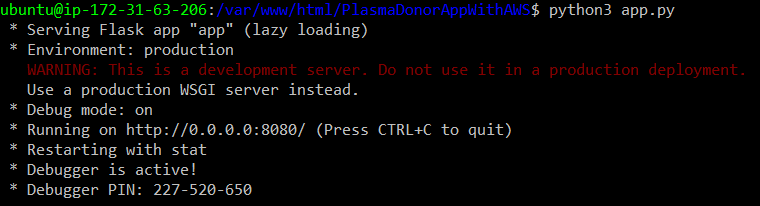
**Step 7:** Open the folder from /var/www/html/foldername > delete the existing zip file



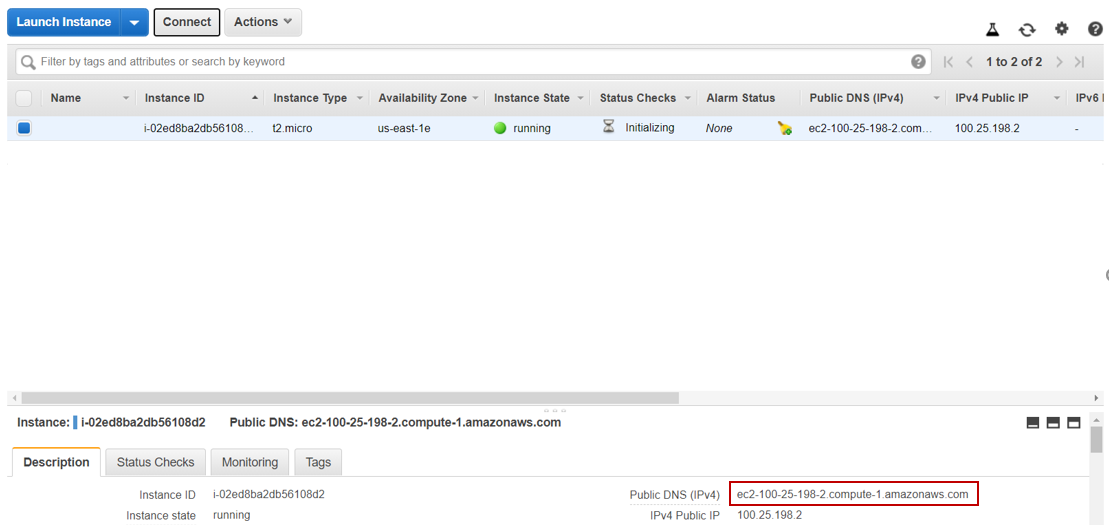
**Command to remove a folder:** rm -r filename.zip

**Step 8:** Run the python file (application)

**Command:** python3 app.py



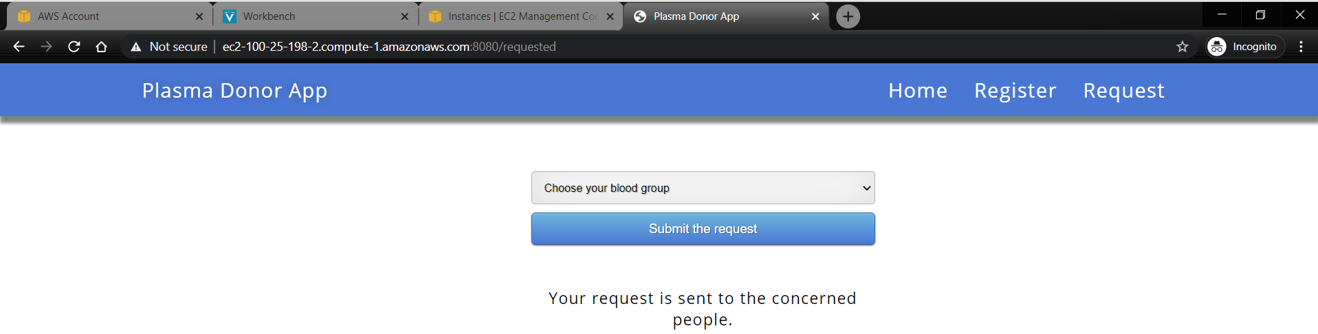
**Step 9:** Copy the public DNS. You can find it in the instances.



**Step 10:** Open the browser > paste the DNS copied > append the URL with :8080 (port number specified in the flask application) > application will be launched

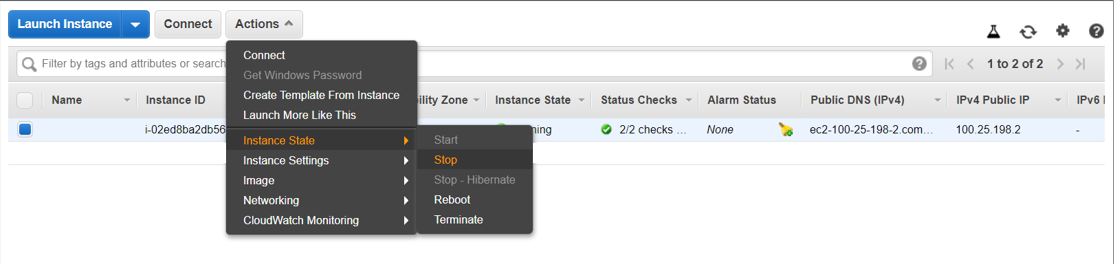


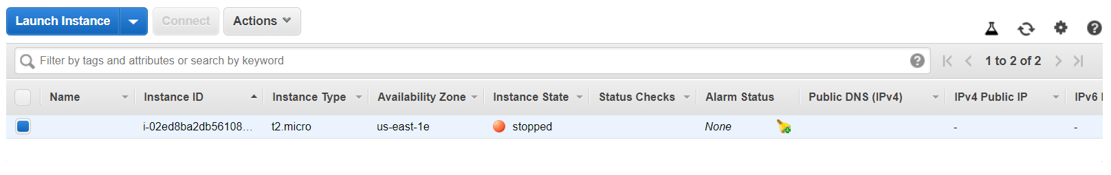




**Important things to be noted while using AWS Cloud:**

* AWS Cloud is pay-as-you-use service.
* To avoid extra billing or loss of credits, make sure you stop the instances that are running when not in use.
* To stop the flask application that is running, give the command ctrl+c.
* To stop the instance of EC2, go to instances > choose your instance > under Actions dropdown > Instance State > Stop





* To start it follow the previous step and click on start.
* Logout from the console when not in use. Do not leave the console logged in if not in use.
* Keep checking the billing page frequently to measure the extra billing in your account.