**1. Copy the updated microservices to the respective docker folders**

Go to visual studio code => Terminal => New terminal~~.~~ Run these commands. \y suppresses the overwrite prompt.

Remember to uncomment this line

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'mysql+mysqlconnector://admin:IloveESMandPaul!<3@esd.cemjatk2jkn2.ap-southeast-1.rds.amazonaws.com/esd\_doctor'

and comment the above uri in ALL the python files inside dockerfiles

copy /y C:\wamp64\www\ESD-ClinicAppointmentServices\app\appointment\appointment.py C:\wamp64\www\ESD-ClinicAppointmentServices\docker\dockerfiles\appointment

copy /y C:\wamp64\www\ESD-ClinicAppointmentServices\app\patient\patient.py C:\wamp64\www\ESD-ClinicAppointmentServices\docker\dockerfiles\patient

copy /y C:\wamp64\www\ESD-ClinicAppointmentServices\app\doctor\doctor.py C:\wamp64\www\ESD-ClinicAppointmentServices\docker\dockerfiles\doctor

copy /y C:\wamp64\www\ESD-ClinicAppointmentServices\app\consultation\consultation.py C:\wamp64\www\ESD-ClinicAppointmentServices\docker\dockerfiles\consultation

copy /y C:\wamp64\www\ESD-ClinicAppointmentServices\app\notification\notification.py C:\wamp64\www\ESD-ClinicAppointmentServices\docker\dockerfiles\notification

copy /y C:\wamp64\www\ESD-ClinicAppointmentServices\app\payment\payment.py C:\wamp64\www\ESD-ClinicAppointmentServices\docker\dockerfiles\payment

**2. Install awscli**

pip install awscli

**3. Allow root access to awscli**

aws configure

Put in the below values:

AWS Access Key ID [\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*gqin]: AKIAI5CGYYPCOA4DMRDA

AWS Secret Access Key [\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ord1]: 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

Default region name [ap-southeast-1]: ap-southeast-1

Default output format [json]: json

**4. Login to aws ECR**

aws ecr get-login-password --region ap-southeast-1 | docker login --username AWS --password-stdin 603184320246.dkr.ecr.ap-southeast-1.amazonaws.com/esd

Go to AWS => ECR => Repositories => Create repository to create 3 repositories called g6t8/patient, g6t8/doctor, g6t8/appointment

**5. Build docker images and push to AWS ECS**

**Need to delete any existing images (NOT repositories) manually first in AWS before pushing**

Open docker.exe. Run these commands in visual studio code

cd C:\wamp64\www\ESD-ClinicAppointmentServices\docker\dockerfiles\patient

docker build -t g6t8/patient .

cd C:\wamp64\www\ESD-ClinicAppointmentServices\docker\dockerfiles\doctor

docker build -t g6t8/doctor .

cd C:\wamp64\www\ESD-ClinicAppointmentServices\docker\dockerfiles\appointment

docker build -t g6t8/appointment .

cd C:\wamp64\www\ESD-ClinicAppointmentServices\docker\dockerfiles\consultation

docker build -t g6t8/consultation .

cd C:\wamp64\www\ESD-ClinicAppointmentServices\docker\dockerfiles\payment

docker build -t g6t8/payment .

cd C:\wamp64\www\ESD-ClinicAppointmentServices\docker\dockerfiles\notification

docker build -t g6t8/notification .

docker tag g6t8/patient:latest 603184320246.dkr.ecr.ap-southeast-1.amazonaws.com/g6t8/patient:latest

docker push 603184320246.dkr.ecr.ap-southeast-1.amazonaws.com/g6t8/patient:latest

docker tag g6t8/doctor:latest 603184320246.dkr.ecr.ap-southeast-1.amazonaws.com/g6t8/doctor:latest

docker push 603184320246.dkr.ecr.ap-southeast-1.amazonaws.com/g6t8/doctor:latest

docker tag g6t8/appointment:latest 603184320246.dkr.ecr.ap-southeast-1.amazonaws.com/g6t8/appointment:latest

docker push 603184320246.dkr.ecr.ap-southeast-1.amazonaws.com/g6t8/appointment:latest

docker tag g6t8/notification:latest 603184320246.dkr.ecr.ap-southeast-1.amazonaws.com/g6t8/notification:latest

docker push 603184320246.dkr.ecr.ap-southeast-1.amazonaws.com/g6t8/notification:latest

docker tag g6t8/payment:latest 603184320246.dkr.ecr.ap-southeast-1.amazonaws.com/g6t8/payment:latest

docker push 603184320246.dkr.ecr.ap-southeast-1.amazonaws.com/g6t8/payment:latest

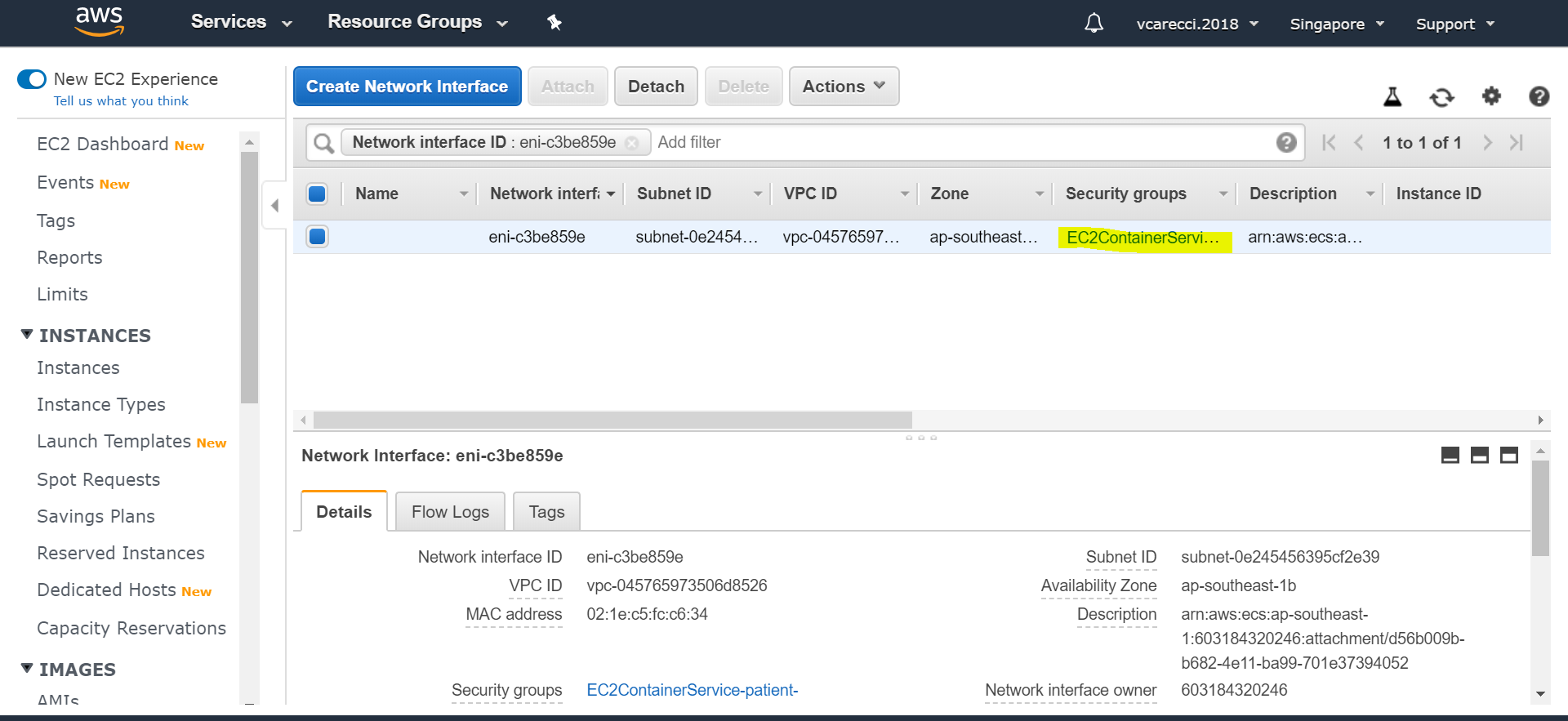
**6. Set up AWS ECR**

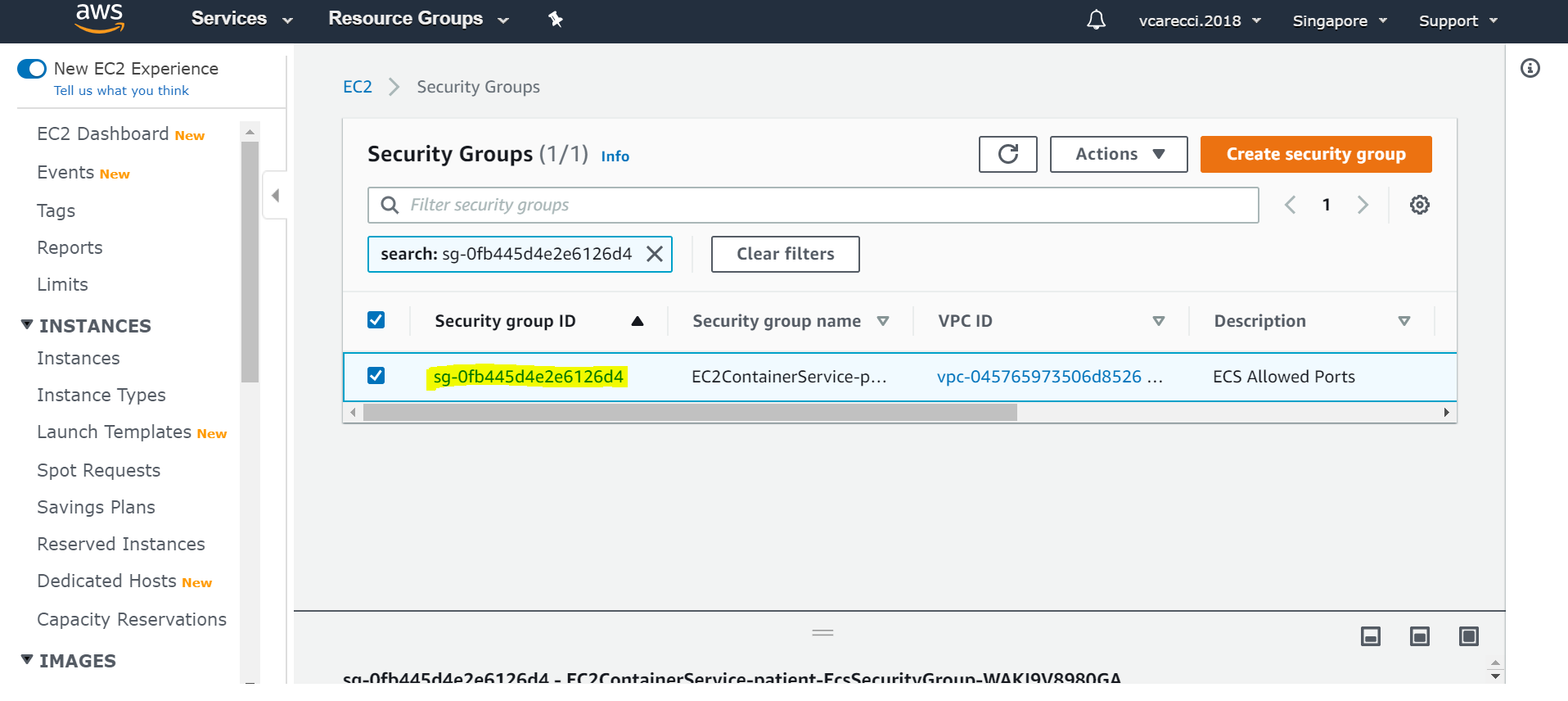
Follow this guide:

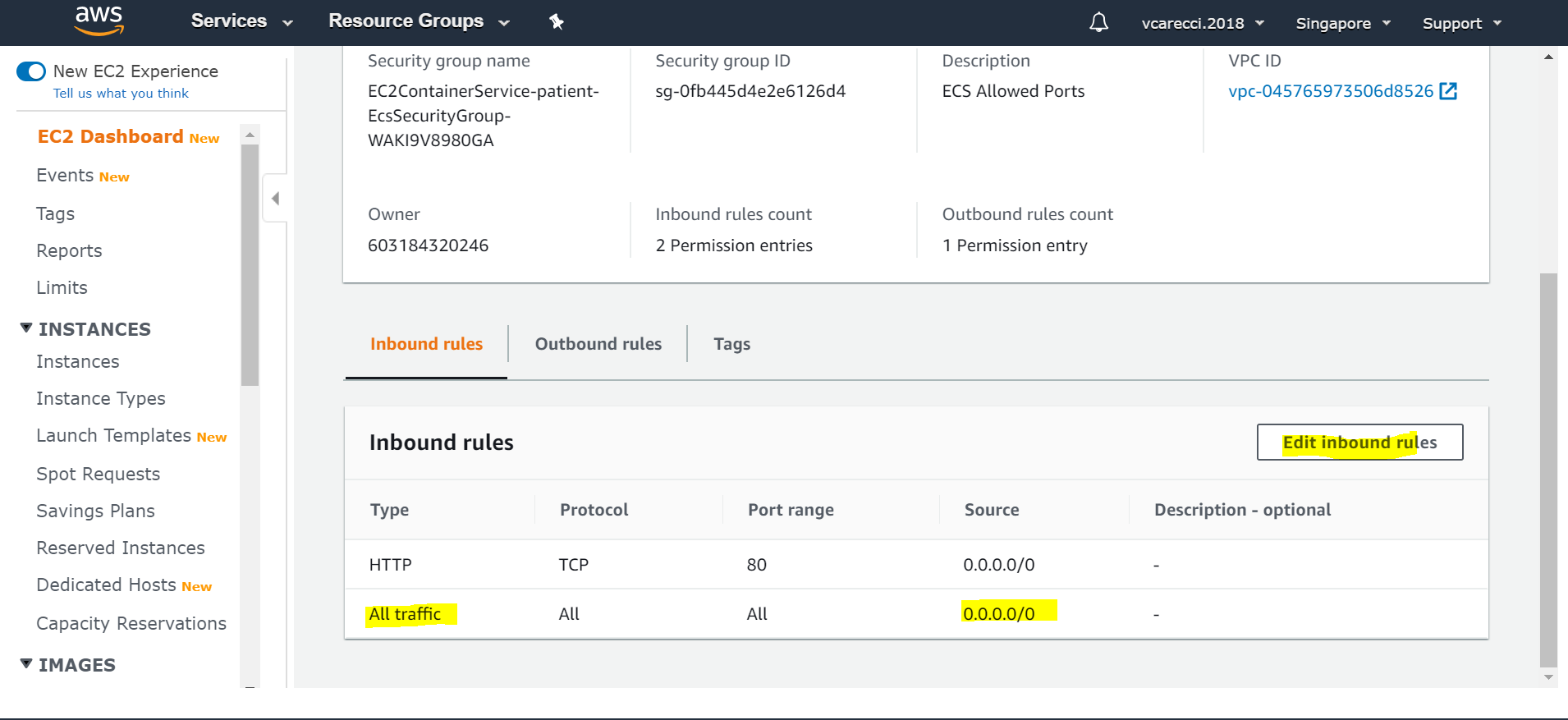
<https://linuxacademy.com/blog/linux-academy/deploying-a-containerized-flask-application-with-aws-ecs-and-docker/>

VERY IMPT note: Do not enable load balancing: It will not work!! Can put 80 for port

Afterwards, need to add in security groups to allow us to access the website







**7. Connect phpMyAdmin to AWS RDS**

ESD Database identifier: esd

Username: admin

Password: IloveESMandPaul!<3

Follow this guide to connect RDS to phpmyadmin

https://scottontechnology.com/connect-to-amazon-rds-mysql-with-phpmyadmin/

C:\wamp64\apps\phpmyadmin4.8.3\config.inc.php

Need to edit the security group of the ec2 container to allow all incoming traffics and 0.0.0.0

**The following steps will set up the Kong API Gateway**

**8. Run an instance of the image of all our microservices:**

docker run --rm -d --name=patient1 --network=kong-net -e dbURL=mysql+mysqlconnector://is213@host.docker.internal:3306/esd\_patient g6t8/patient

docker run --rm -d --name=doctor1 --network=kong-net -e dbURL=mysql+mysqlconnector://is213@host.docker.internal:3306/esd\_doctor g6t8/doctor

docker run --rm -d --name=appointment1 --network=kong-net -e dbURL=mysql+mysqlconnector://is213@host.docker.internal:3306/esd\_appointment g6t8/appointment

docker run --rm -d --name=consultation1 --network=kong-net -e dbURL=mysql+mysqlconnector://is213@host.docker.internal:3306/esd\_consultation g6t8/consultation

Create another container of each microservice for load balancing

docker run --rm -d --name=patient2 --network=kong-net -e dbURL=mysql+mysqlconnector://is213@host.docker.internal:3306/esd\_patient g6t8/patient

docker run --rm -d --name=doctor2 --network=kong-net -e dbURL=mysql+mysqlconnector://is213@host.docker.internal:3306/esd\_doctor g6t8/doctor

docker run --rm -d --name=appointment2 --network=kong-net -e dbURL=mysql+mysqlconnector://is213@host.docker.internal:3306/esd\_appointment g6t8/appointment

docker run --rm -d --name=consultation2 --network=kong-net -e dbURL=mysql+mysqlconnector://is213@host.docker.internal:3306/esd\_consultation g6t8/consultation