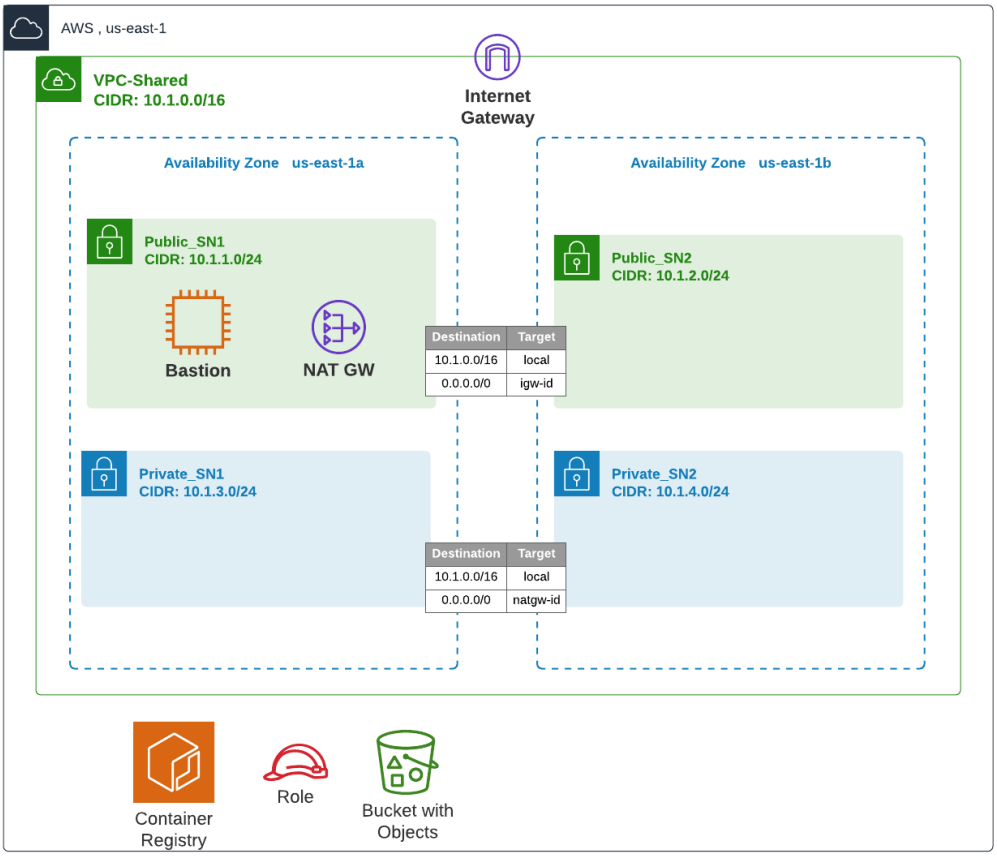
SYST35144: Cloud Systems

Lab3: Deploying Web Application with Docker Containers

Architecture Diagram



Pre-requisites

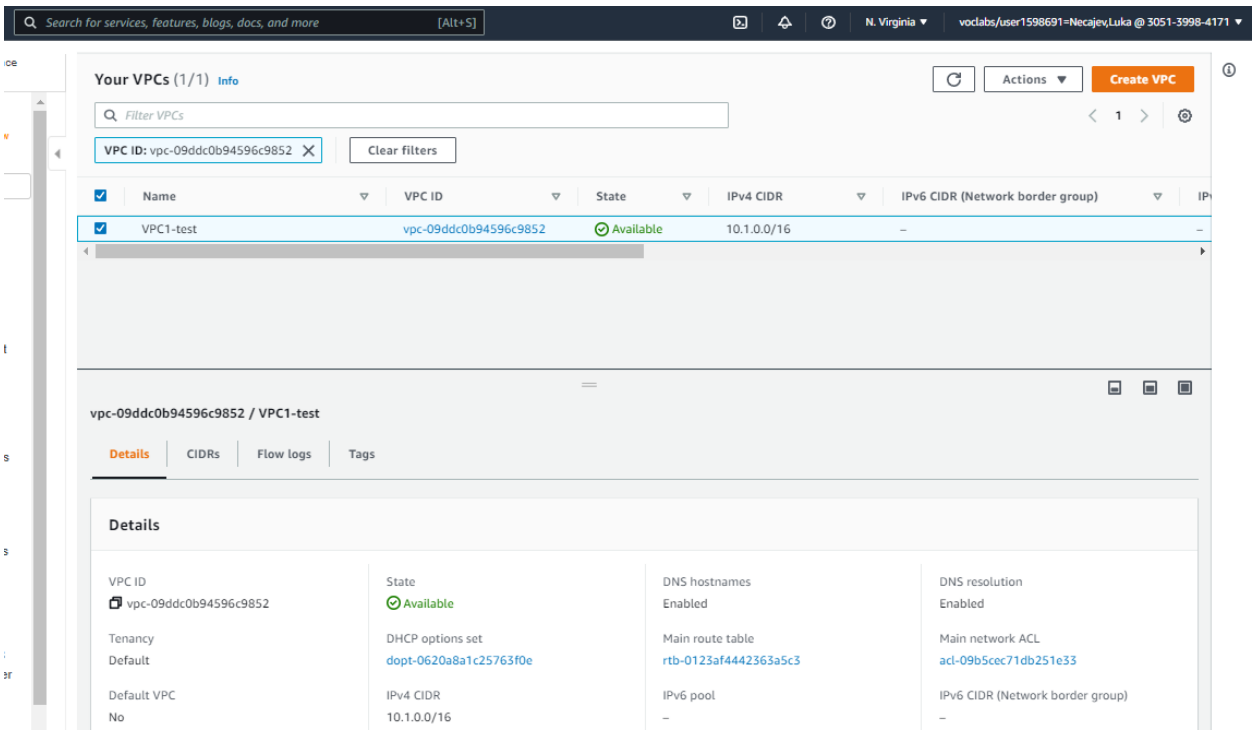
- AWS CLI

- AWS Credentials
- Terraform
- Docker
- IDE of your choice (Cloud9 or Visual Source code)
- Git

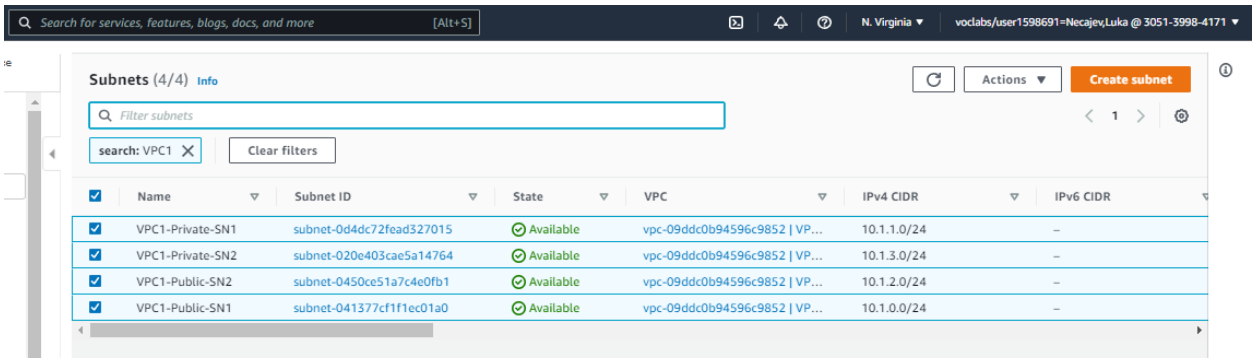
Part 1 – Deployment of basic Networking Infrastructure with Terraform

The deployed infrastructure should include:

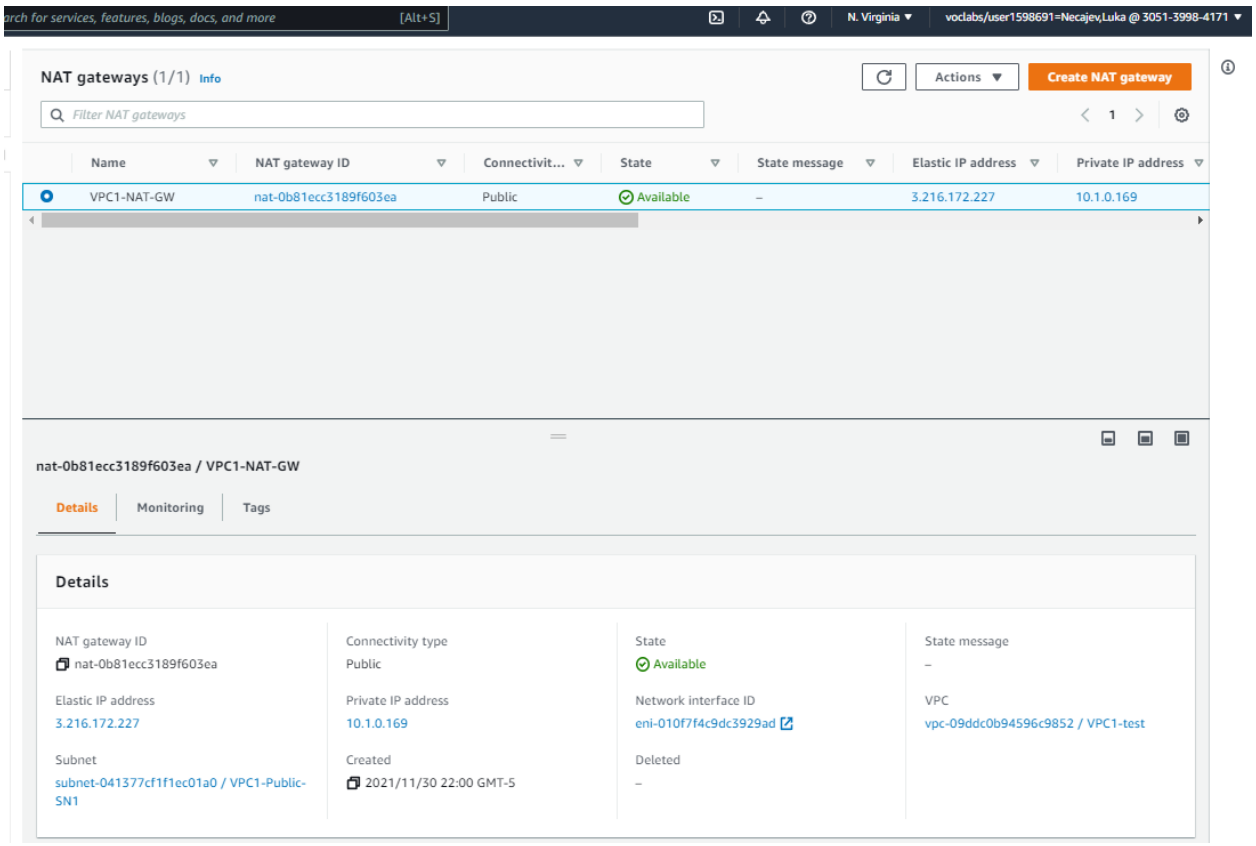
- Custom VPC with CIDR 10.1.0.0/16



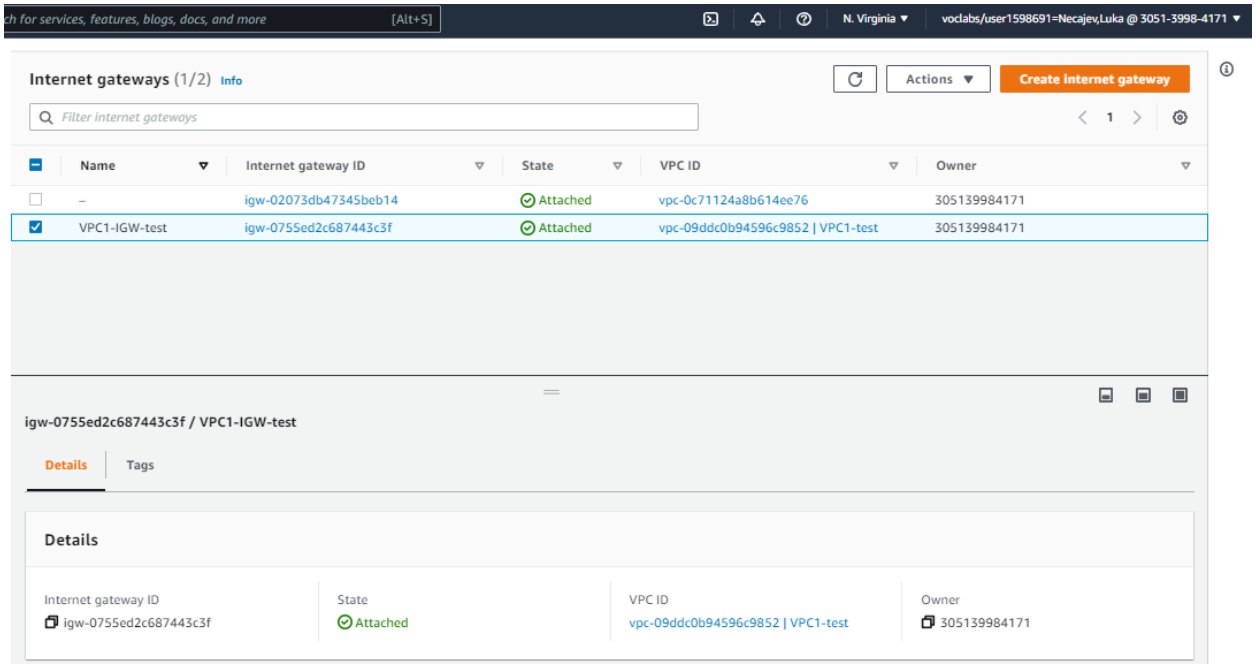
- 2 private and 2 public subnets with 256 IP addresses each



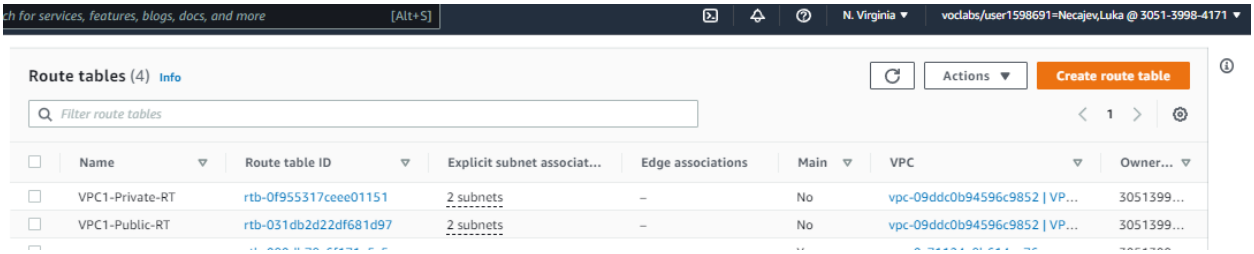
- NAT GW,


arch for services, features, blogs, docs, and more [Alt+S]
N. Virginia
voclabs/user1598691=Necajev,Luka @ 3051-3998-4171
NAT gateways (1/1) Info
Actions
Create NAT gateway
Filter NAT gateways
Name
NAT gateway ID
Connectivit...
State
State message
Elastic IP address
Private IP address
VPC1-NAT-GW
nat-0b81ecc3189f603ea
Public
Available
–
3.216.172.227
10.1.0.169
nat-0b81ecc3189f603ea / VPC1-NAT-GW
Details
Monitoring
Tags
Details
NAT gateway ID
nat-0b81ecc3189f603ea
Connectivity type
Public
State
Available
State message
–
Elastic IP address
3.216.172.227
Private IP address
10.1.0.169
Network interface ID
eni-010f7f4c9dc3929ad
VPC
vpc-09ddc0b94596c9852 / VPC1-test
Subnet
subnet-041377cf1f1ec01a0 / VPC1-Public-SN1
Created
2021/11/30 22:00 GMT-5
Deleted
–

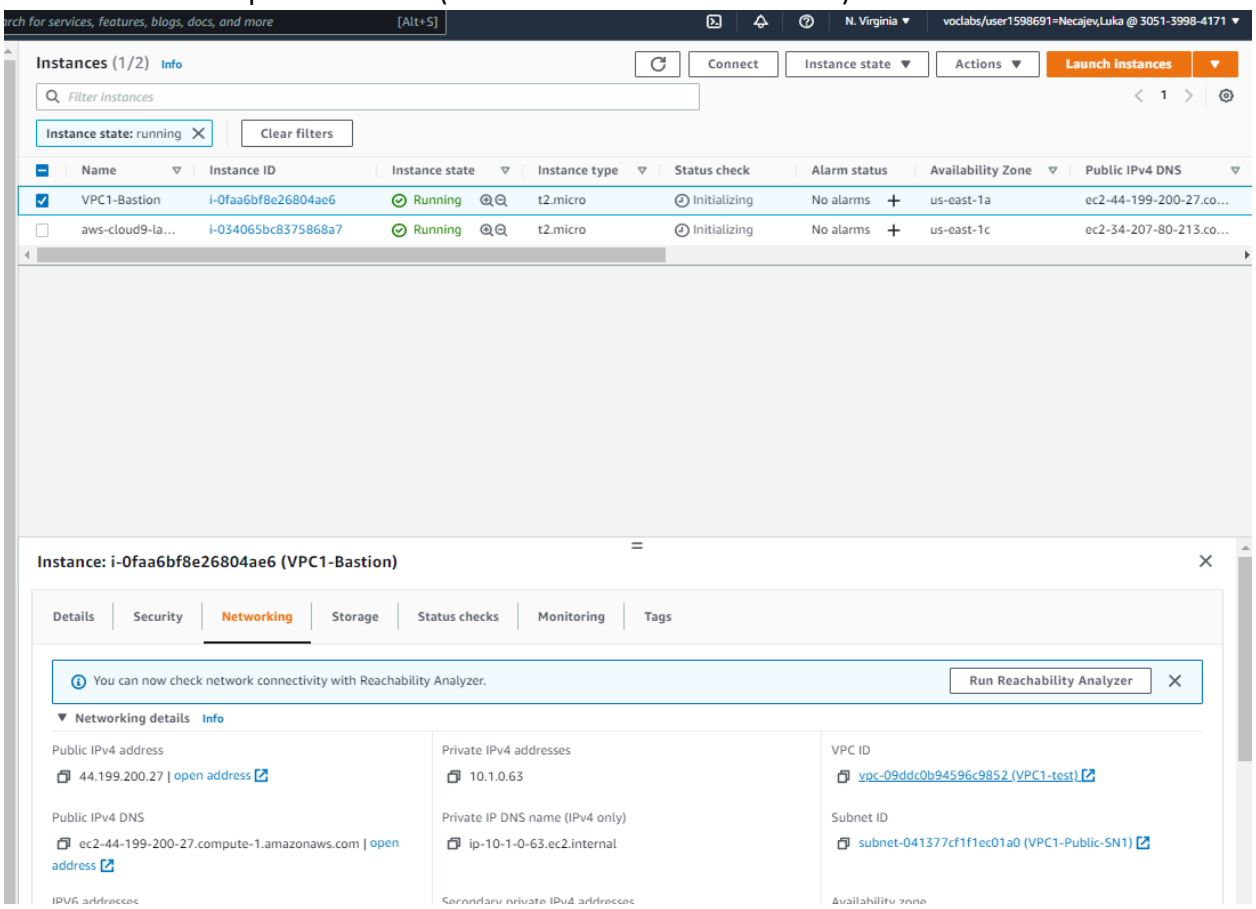
- IGW



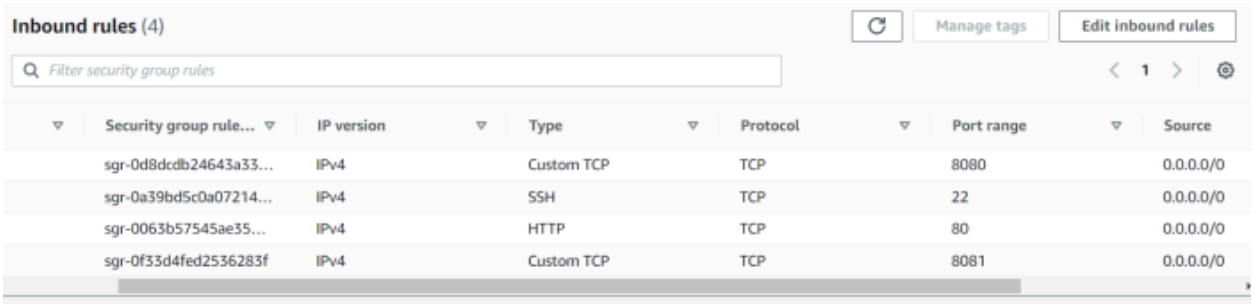
- Route tables (one route table for private subnets, one for public subnets)



- One Ec2 in the public subnet (Second one is Cloud9 Instance)



- Security Group associated with EC2. SG has only required ports open.



I am opening port 80 for testing, port 8080 for my dogs instance and 8081 for my cats instance. I also had to open SSH port 22 to all ip's as I do not know what IP my aws session has.

Code for IAM ROLE & ECR:

```
data "aws_iam_role" "ecr" {
  name = "LabRole"
}

# Amazon ECR repository to store docker images
module "ecr" {
  source                = "cloudposse/ecr/aws"
  namespace             = "lab3"
  stage                 = "test"
  name                  = "dogscats"
  principals_full_access = [data.aws_iam_role.ecr.arn]

}
```

EXPLINATION!

Explanation of IAM role assignment to EC2 instance - why do we need it and what it allowed EC2 to do.

IAM role assignment allows admins to manage access to AWS accounts and permissions given to the accounts within a group. Our role "LabRole" Allows our instance to have the permissions of a IAM group named "LabRole". In our case we created LabRole to give us full access in our ecr instance as shown in the above screenshot. This will give us full access to the ecr we create in the next couple line of code.

Creating the TERRAFORM

```
voclabs:~/environment $ alias tf=terraform
voclabs:~/environment $ cd terraforming/
voclabs:~/environment/terraforming $ tf init
Initializing modules...
Downloading terraform-aws-modules/ec2-instance/aws 3.3.0 for bastion-VPC1...
- bastion-VPC1 in .terraform/modules/bastion-VPC1
Downloading terraform-aws-modules/security-group/aws 4.7.0 for bastion_sg_VPC1...
- bastion_sg_VPC1 in .terraform/modules/bastion_sg_VPC1
Downloading cloudposse/ecr/aws 0.32.3 for ecr...
- ecr in .terraform/modules/ecr
Downloading cloudposse/label/null 0.25.0 for ecr.this...
- ecr.this in .terraform/modules/ecr.this
- networking_VPC1 in 01-VPC

Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...
- Finding hashicorp/aws versions matching ">= 3.0.0, >= 3.1.0, ~> 3.27, >= 3.51.0"...
- Installing hashicorp/aws v3.67.0...
- Installed hashicorp/aws v3.67.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
voclabs:~/environment/terraforming $
```

Applying the Terra form

```
voclabs:~/environment/terraforming $
voclabs:~/environment/terraforming $ tf apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create
```

```
  # module.networking_VPC1.aws_vpc.vpc-tf will be created
  + resource "aws_vpc" "vpc-tf" {
      + arn                              = (known after apply)
      + assign_generated_ipv6_cidr_block = false
      + cidr_block                       = "10.1.0.0/16"
      + default_network_acl_id           = (known after apply)
      + default_route_table_id           = (known after apply)
      + default_security_group_id        = (known after apply)
      + dhcp_options_id                  = (known after apply)
      + enable_classiclink               = (known after apply)
      + enable_classiclink_dns_support   = (known after apply)
      + enable_dns_hostnames             = true
      + enable_dns_support               = true
      + id                               = (known after apply)
      + instance_tenancy                 = "default"
      + ipv6_association_id              = (known after apply)
      + ipv6_cidr_block                  = (known after apply)
      + main_route_table_id              = (known after apply)
      + owner_id                         = (known after apply)
      + tags                             = {
          + "Environment" = "Test"
          + "Name"        = "VPC1-test"
          + "Owner"       = "Luka"
          + "Project"     = "Session9"
        }
      + tags_all                         = {
          + "Environment" = "Test"
          + "Name"        = "VPC1-test"
          + "Owner"       = "Luka"
          + "Project"     = "Session9"
        }
    }

Plan: 21 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

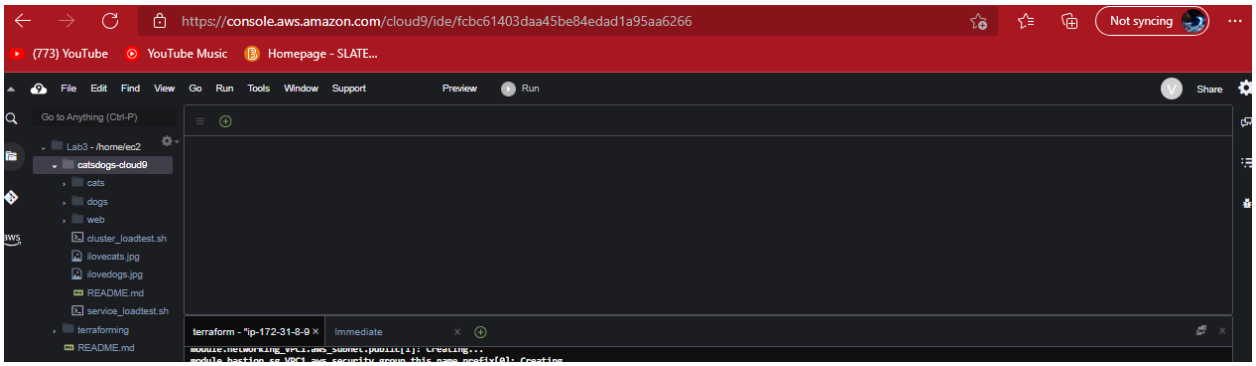
Skipped to the end...

```
module.bastion-VPC1.aws_instance.this[0]: Creating...
module.bastion_sg_VPC1.aws_security_group_rule.ingress_with_cidr_blocks[0]: Creating...
module.bastion_sg_VPC1.aws_security_group_rule.egress_rules[0]: Creating...
module.networking_VPC1.aws_route_table_association.association-pub[1]: Creation complete after 1s [id=rtbassoc-082f3f70395bc7612]
module.networking_VPC1.aws_route_table_association.association-pub[0]: Creation complete after 1s [id=rtbassoc-07234c6f7824682cd]
module.bastion_sg_VPC1.aws_security_group_rule.egress_rules[0]: Creation complete after 1s [id=sgrule-2045685988]
module.bastion_sg_VPC1.aws_security_group_rule.ingress_with_cidr_blocks[0]: Creation complete after 1s [id=sgrule-3341536024]
module.networking_VPC1.aws_nat_gateway.nat[0]: Still creating... [10s elapsed]
module.bastion-VPC1.aws_instance.this[0]: Still creating... [10s elapsed]
module.networking_VPC1.aws_nat_gateway.nat[0]: Still creating... [20s elapsed]
module.bastion-VPC1.aws_instance.this[0]: Still creating... [20s elapsed]
module.networking_VPC1.aws_nat_gateway.nat[0]: Still creating... [30s elapsed]
module.bastion-VPC1.aws_instance.this[0]: Still creating... [30s elapsed]
module.networking_VPC1.aws_nat_gateway.nat[0]: Still creating... [40s elapsed]
module.bastion-VPC1.aws_instance.this[0]: Still creating... [40s elapsed]
module.networking_VPC1.aws_nat_gateway.nat[0]: Still creating... [50s elapsed]
module.bastion-VPC1.aws_instance.this[0]: Still creating... [50s elapsed]
module.networking_VPC1.aws_nat_gateway.nat[0]: Still creating... [1m0s elapsed]
module.bastion-VPC1.aws_instance.this[0]: Still creating... [1m0s elapsed]
module.bastion-VPC1.aws_instance.this[0]: Creation complete after 1m3s [id=i-0cbaa8277f9954ac5]
module.networking_VPC1.aws_nat_gateway.nat[0]: Still creating... [1m10s elapsed]
module.networking_VPC1.aws_nat_gateway.nat[0]: Still creating... [1m20s elapsed]
module.networking_VPC1.aws_nat_gateway.nat[0]: Creation complete after 1m24s [id=nat-0a75ce3a82314a37e]
module.networking_VPC1.aws_route_table.rt-private[0]: Creating...
module.networking_VPC1.aws_route_table.rt-private[0]: Creation complete after 1s [id=rtb-0f716e21a988c668d]
module.networking_VPC1.aws_route_table_association.association-pr[0]: Creating...
module.networking_VPC1.aws_route_table_association.association-pr[1]: Creating...
module.networking_VPC1.aws_route_table_association.association-pr[0]: Creation complete after 1s [id=rtbassoc-0d2fe54bd5bb5804f]
module.networking_VPC1.aws_route_table_association.association-pr[1]: Creation complete after 1s [id=rtbassoc-092dff0b5ed18e813]

Apply complete! Resources: 21 added, 0 changed, 0 destroyed.
voclabs:~/environment/terraforming $
```

Part 2 – Build and deploy cat's and dog's container images

1. Clone the cats and dogs repository into your local environment



2. **Build cats docker image**

Building the docekr Image

```
voclabs:~/environment/catsdogs-cloud9/cats $ docker build -t cats .
Sending build context to Docker daemon  4.608kB
Step 1/6 : FROM nginx
 ---> ea335eea17ab
Step 2/6 : EXPOSE 80
 ---> Using cache
 ---> a11fa492a14d
Step 3/6 : RUN apt-get update -y &&   apt-get upgrade -y &&   apt-get install -y curl &&   cd /tmp &&   apt-get install awscli -y &&   rm -rf /tmp/* &&   rm -rf /var/lib/apt/lists/*
 ---> Using cache
 ---> 9698f8816743
Step 4/6 : COPY ./default.conf /etc/nginx/conf.d/default.conf
 ---> Using cache
 ---> 395084918606
Step 5/6 : COPY ./index.html /usr/share/nginx/html/index.html
 ---> 9a75784611ce
Step 6/6 : CMD nginx -g "daemon off;"
 ---> Running in 1124c15ca873
Removing intermediate container 1124c15ca873
 ---> 3a54aabcbb4a
Successfully built 3a54aabcbb4a
Successfully tagged cats:latest
voclabs:~/environment/catsdogs-cloud9/cats $
```

3. **Build dogs docker image**

Building the docekr Image

```
voclabs:~/environment/catsdogs-cloud9/dogs $ docker build -t dogs .
Sending build context to Docker daemon  4.608kB
Step 1/6 : FROM nginx
latest: Pulling from library/nginx
eff15d958d66: Pull complete
1e5351450a59: Pull complete
2df63e6ce2be: Pull complete
9171c7ae368c: Pull complete
020f975acd28: Pull complete
266f639b35ad: Pull complete
Digest: sha256:097c3a0913d7e3a5b01b6c685a60c03632fc7a2b50bc8e35bcaa3691d788226e
Status: Downloaded newer image for nginx:latest
 ---> ea335eea17ab
Step 2/6 : EXPOSE 80
 ---> Running in 66e4e0194865
Removing intermediate container 66e4e0194865
```

```
update-alternatives: using /usr/share/doc/awscli/scripts/py
Setting up awscli (1.19.1-1) ...
Removing intermediate container 3825632df741
 ---> 9698f8816743
Step 4/6 : COPY ./default.conf /etc/nginx/conf.d/default.
 ---> df2935ba2951
Step 5/6 : COPY ./index.html /usr/share/nginx/html/index.
 ---> fc5fab34cc7e
Step 6/6 : CMD nginx -g "daemon off;"
 ---> Running in e010b032dcca
Removing intermediate container e010b032dcca
 ---> 74f635c2e4d8
Successfully built 74f635c2e4d8
Successfully tagged dogs:latest
voclabs:~/environment/catsdogs-cloud9/dogs $
```

Checking if the image was built

```
voclabs:~/environment/catsdogs-cloud9/dogs $ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
dogs                latest              74f635c2e4d8        21 seconds ago      439MB
nginx               latest              ea335eea17ab        13 days ago         141MB
lambci/lambda       python3.8           094248252696        10 months ago       524MB
lambci/lambda       nodejs12.x          22a4ada8399c        10 months ago       390MB
lambci/lambda       nodejs10.x          db93be728e7b        10 months ago       385MB
lambci/lambda       python3.7           22b4b6fd9260        10 months ago       946MB
lambci/lambda       python3.6           177c85a10179        10 months ago       894MB
lambci/lambda       python2.7           d96a01fe4c80        10 months ago       763MB
lambci/lambda       nodejs8.10          5754fee26e6e        10 months ago       813MB
voclabs:~/environment/catsdogs-cloud9/dogs $
```

Exporting the Repositoryname

```
voclabs:~/environment/catsdogs-cloud9/dogs $ export ECR=305139984171.dkr.ecr.us-east-1.amazonaws.com
voclabs:~/environment/catsdogs-cloud9/dogs $ export REPO_NAME=lab3-test-dogscats
```

4. Log into AWS ECR

Aws ecr get-login-password --region us-east-1 | docker login -u AWS 305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats –password-stdin

```
voclabs:~/environment/catsdogs-cloud9/dogs $ aws ecr get-login-password --region us-east-1 | docker login -u AWS 305139984171.dkr.ecr.us-east-1.amazonaws.com --password-stdin
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
voclabs:~/environment/catsdogs-cloud9/dogs $ docker tag dogs "${ECR}/lab3-test-dogscats:v1.0"
voclabs:~/environment/catsdogs-cloud9/dogs $ docker images
REPOSITORY                                                  TAG         IMAGE ID       CREATED         SIZE
305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats   v1.0        74f635c2e4d8   6 minutes ago   439MB
dogs                                                        latest      74f635c2e4d8   6 minutes ago   439MB
nginx                                                       latest      ea335eea17ab   13 days ago     141MB
lambci/lambda                                               python3.8   094248252696   10 months ago   524MB
lambci/lambda                                               nodejs12.x  22a4ada8399c   10 months ago   390MB
lambci/lambda                                               nodejs10.x  db93be728e7b   10 months ago   385MB
lambci/lambda                                               python3.7   22b4b6fd9260   10 months ago   946MB
lambci/lambda                                               python3.6   177c85a10179   10 months ago   894MB
lambci/lambda                                               python2.7   d96a01fe4c80   10 months ago   763MB
lambci/lambda                                               nodejs8.10  5754fee26e6e   10 months ago   813MB
voclabs:~/environment/catsdogs-cloud9/dogs $ docker push "${ECR}/lab3-test-dogscats:v1.0"
The push refers to repository [305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats]
a0575a233db4: Pushed
2b8d881a3c3d: Pushed
a6906e0adf1c: Pushing [=====>                                             ]  33.09MB/297.3MB
8525cde30b22: Pushed
1e8ad06c81b6: Pushed
49eeddd2150f: Pushed
ff4c72779430: Pushed
37380c5830fe: Pushing [==========================>                        ]  35.34MB/61.09MB
e1bbcf243d0e: Pushing [=========>                                         ]  15.71MB/80.39MB
```

Tag the docker images for cat and dog

```
voclabs:~/environment/catsdogs-cloud9/dogs $ docker tag dogs "${ECR}/lab3-test-dogscats:v1.0"
voclabs:~/environment/catsdogs-cloud9/cats $ docker tag cats "${ECR}/lab3-test-dogscats:v2.0"
```

Check that they are updated

```
voclabs:~/environment/catsdogs-cloud9/cats $ docker images
REPOSITORY                                                  TAG       IMAGE ID       CREATED          SIZE
305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats   v1.0      12c36786d754   12 minutes ago   439MB
305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats   v2.0      12c36786d754   12 minutes ago   439MB
```

5. Push cats docker image to cats ECR repository

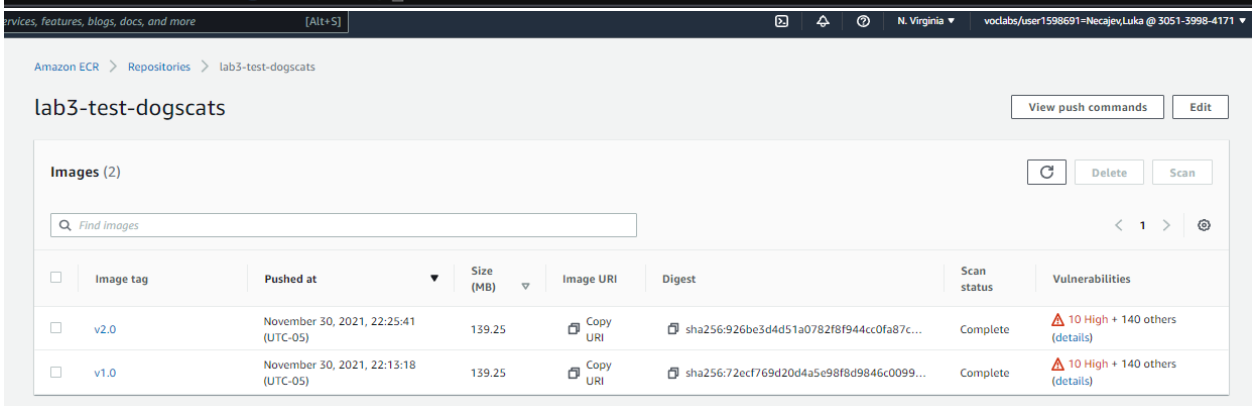
```
voclabs:~/environment/catsdogs-cloud9/cats $ docker tag cats "${ECR}/lab3-test-dogscats:v2.0"
voclabs:~/environment/catsdogs-cloud9/cats $ docker push "${ECR}/lab3-test-dogscats:v2.0"
The push refers to repository [305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats]
f2980f55eb73: Pushed
34662fbc7a81: Pushed
a6906e0adf1c: Pushing [================>                                  ]  98.35MB/297.3MB
8525cde30b22: Layer already exists
1e8ad06c81b6: Layer already exists
49eeddd2150f: Layer already exists
ff4c72779430: Layer already exists
37380c5830fe: Layer already exists
e1bbcf243d0e: Layer already exists
```

6. Push dogs docker image to dogs ECR repository

```
voclabs:~/environment/catsdogs-cloud9/dogs $ docker push "${ECR}/lab3-test-dogscats:v1.0"
The push refers to repository [305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats]
a0575a233db4: Pushed
2b8d881a3c3d: Pushed
a6906e0adf1c: Pushed
8525cde30b22: Pushed
1e8ad06c81b6: Pushed
49eeddd2150f: Pushed
ff4c72779430: Pushed
37380c5830fe: Pushed
e1bbcf243d0e: Pushed
v1.0: digest: sha256:72ecf769d20d4a5e98f8d9846c00995e3b93839db3acc5ad4e9041bd53884957 size: 2196
voclabs:~/environment/catsdogs-cloud9/dogs $
```

Checking that both my images are in the erc

```
voclabs:~/environment/catsdogs-cloud9/cats $ aws ecr describe-images --repository-name lab3-test-dogscats
{
    "imageDetails": [
        {
            "artifactMediaType": "application/vnd.docker.container.image.v1+json",
            "imageSizeInBytes": 139254455,
            "imageDigest": "sha256:926be3d4d51a0782f8f944cc0fa87c8a33c8efeed7694d0e4ec6b5bcce090f46",
            "imageManifestMediaType": "application/vnd.docker.distribution.manifest.v2+json",
            "imageTags": [
                "v2.0"
            ],
            "registryId": "305139984171",
            "imageScanStatus": {
                "status": "COMPLETE",
                "description": "The scan was completed successfully."
            },
            "imageScanFindingsSummary": {
                "imageScanCompletedAt": 1638329150.0,
                "vulnerabilitySourceUpdatedAt": 1637770297.0,
                "findingSeverityCounts": {
                    "HIGH": 10,
                    "MEDIUM": 35,
                    "INFORMATIONAL": 80,
                    "LOW": 19,
                    "UNDEFINED": 6
                }
            },
            "repositoryName": "lab3-test-dogscats",
            "imagePushedAt": 1638329141.0
        },
        {
            "artifactMediaType": "application/vnd.docker.container.image.v1+json",
            "imageSizeInBytes": 139254467,
            "imageDigest": "sha256:72ecf769d20d4a5e98f8d9846c00995e3b93839db3acc5ad4e9041bd53884957",
            "imageManifestMediaType": "application/vnd.docker.distribution.manifest.v2+json",
            "imageTags": [
                "v1.0"
            ],
            "registryId": "305139984171",
            "imageScanStatus": {
                "status": "COMPLETE",
                "description": "The scan was completed successfully."
            },
            "imageScanFindingsSummary": {
                "imageScanCompletedAt": 1638328406.0,
                "vulnerabilitySourceUpdatedAt": 1637770297.0,
                "findingSeverityCounts": {
                    "HIGH": 10,
                    "MEDIUM": 35,
                    "INFORMATIONAL": 80,
                    "LOW": 19,
                    "UNDEFINED": 6
                }
            },
            "repositoryName": "lab3-test-dogscats",
            "imagePushedAt": 1638328398.0
        }
    ]
}
voclabs:~/environment/catsdogs-cloud9/cats $
```



Explanation!
Why do we need to push the images to ECR?

We need to push the images to the ECR because then we do not have to manually operate and scale the infrastructure needed to power the container. Amazon ECR uses the S3 we created at the beginning making our data highly available and accessible to our instances. We do not need to create anything, amazon will host it for us allowing us to deploy new containers easily and quickly!

7. Log into EC2 instance

```
ddd_v1_w_MVbI_882554@runweb43402:~$ ssh -i ~/.ssh/labsuser.pem ec2-user@44.193.73.70
The authenticity of host '44.193.73.70 (44.193.73.70)' can't be established.
ECDSA key fingerprint is SHA256:NUXvvPEuyaBIT3FpIRIp3a1cIhN3DQNJroVC37IM4HY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '44.193.73.70' (ECDSA) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-1-0-63 ~]$ ^C
[ec2-user@ip-10-1-0-63 ~]$
```

8. Log into AWS ECR

aws ecr get-login-password --region us-east-1 | docker login -u AWS 305139984171.dkr.ecr.us-east-1.amazonaws.com --password-stdin

9. Pull cats and dogs images

Pulling the DOG image

```
[ec2-user@ip-10-1-0-63 ~]$ docker pull 305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v1.0
v1.0: Pulling from lab3-test-dogscats
eff15d958d66: Extracting [==================>                                ]  12.12MB/31.37MB
1e5351450a59: Download complete
2df63e6ce2be: Download complete
9171c7ae368c: Download complete
020f975acd28: Download complete
266f639b35ad: Download complete
f756ed6f4076: Downloading [========================>                          ]  42.18MB/82.52MB
a5c56ed4cb92: Download complete
962670e6ec21: Download complete
```

Pulling CAT image

```
[ec2-user@ip-10-1-0-63 ~]$ docker pull 305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v2.0
v2.0: Pulling from lab3-test-dogscats
Digest: sha256:926be3d4d51a0782f8f944cc0fa87c8a33c8efeed7694d0e4ec6b5bcce090f46
Status: Image is up to date for 305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v2.0
305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v2.0
[ec2-user@ip-10-1-0-63 ~]$
```

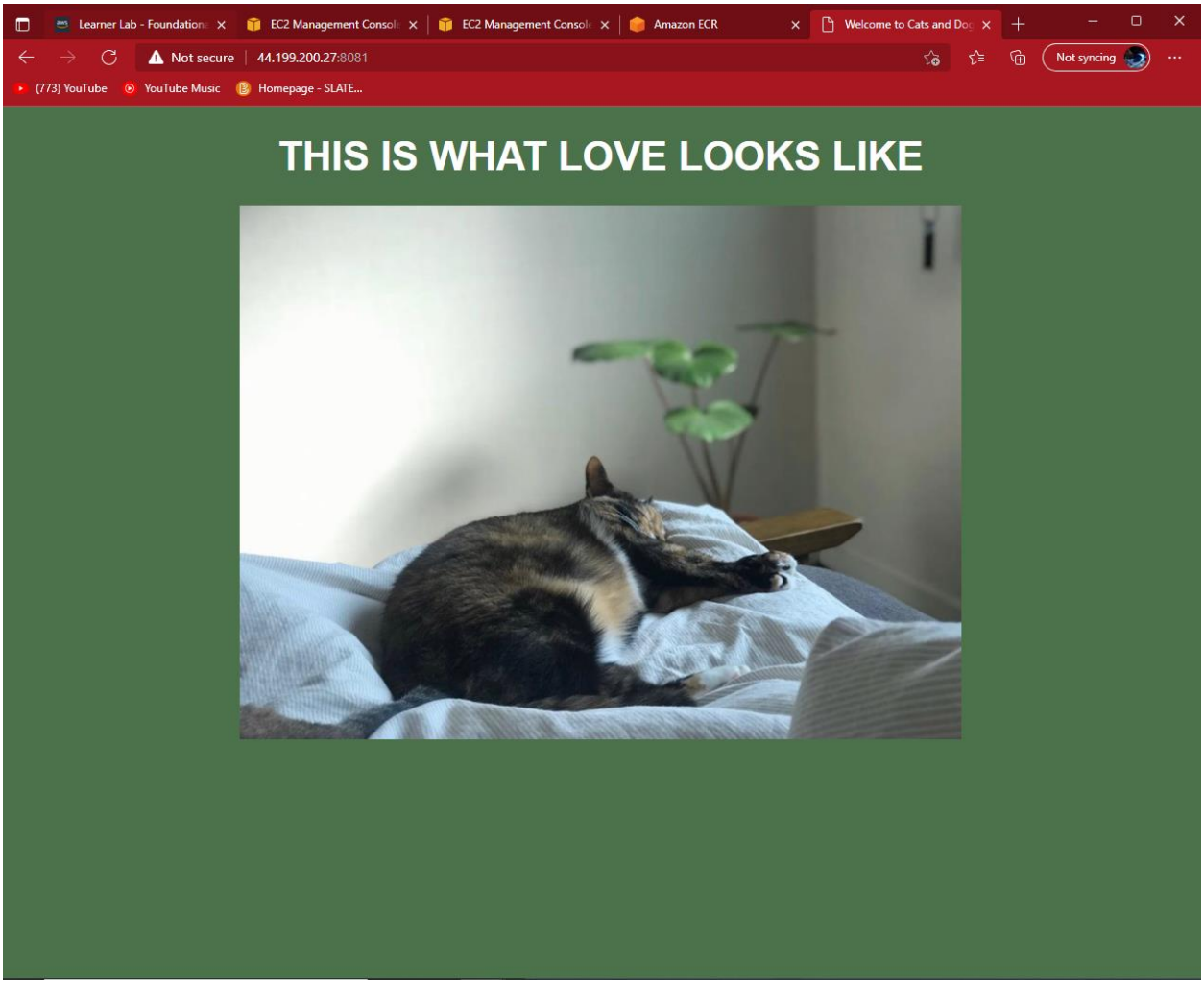
10. Run cat's image. Map container port 80 to host port 8081

Pulling and running cats image on port 8081

```
[ec2-user@ip-10-1-0-63 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-10-1-0-63 ~]$ docker run -it -p 80:8080 -d 305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v2.0
a96acfc075e1a270cce647180df9f7779dfde08daa438d4a34fd653877326b3a
[ec2-user@ip-10-1-0-63 ~]$
```

```
[ec2-user@ip-10-1-0-63 ~]$ docker run -it -p 8081:80 -d 305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v2.0
605cf02f99a384bcbb1a1367ea162286a805dfe7b51d1d8496ab2d2bf0151adb
[ec2-user@ip-10-1-0-63 ~]$ docker ps
CONTAINER ID   IMAGE                                                          COMMAND                  CREATED
STATUS          PORTS                                   NAMES
605cf02f99a3   305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v2.0   "/docker-entrypoint.…"   28 seconds ago
Up 26 seconds   0.0.0.0:8081->80/tcp, :::8081->80/tcp   romantic_goldstine
[ec2-user@ip-10-1-0-63 ~]$
```

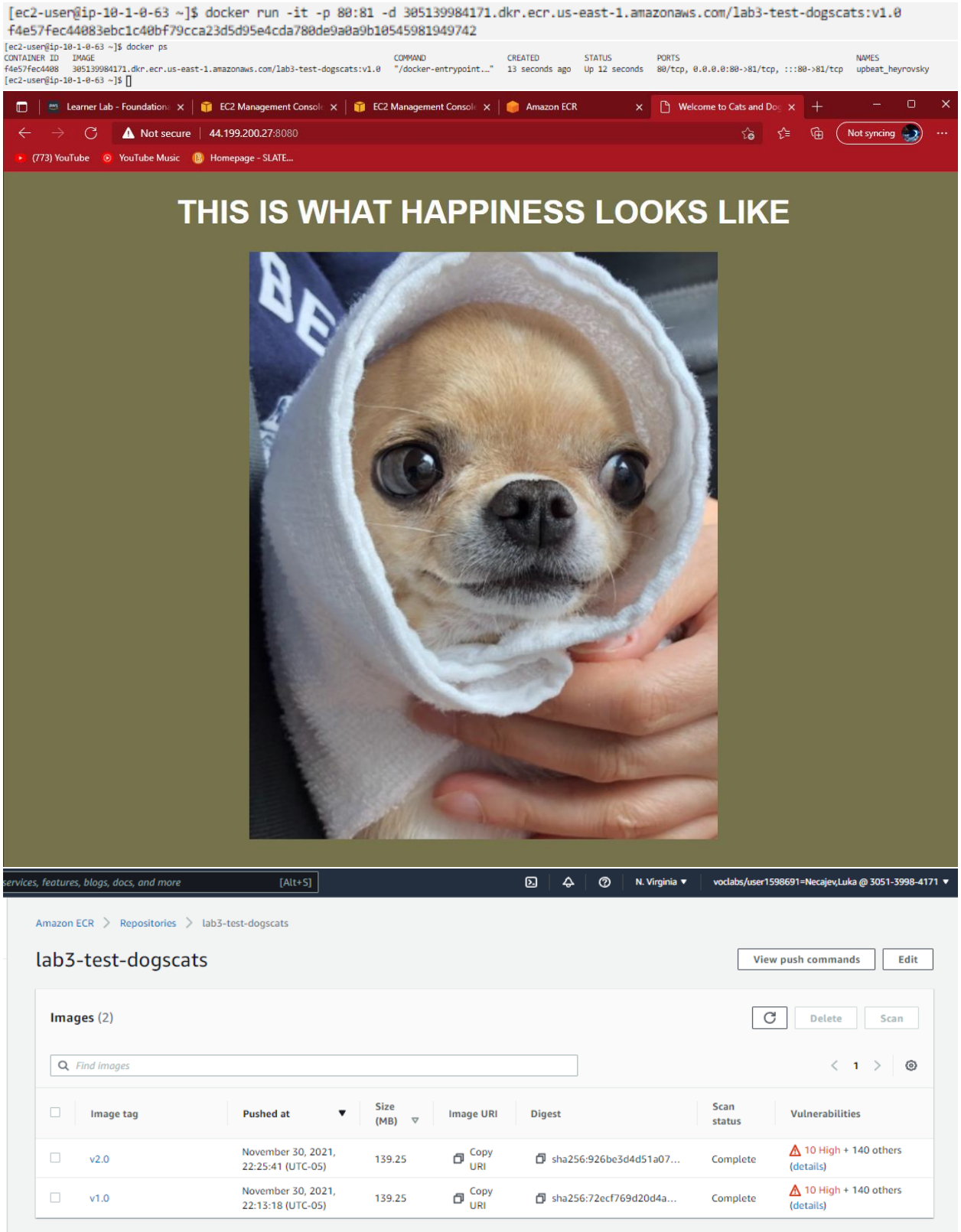
docker run -it -p 8081:80 -d 305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v2.0



11. Run dog's image. Map container port 80 to host port 8080

SYST35144: Cloud Systems

docker run -it -p 80:8080 -d 305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v1.0

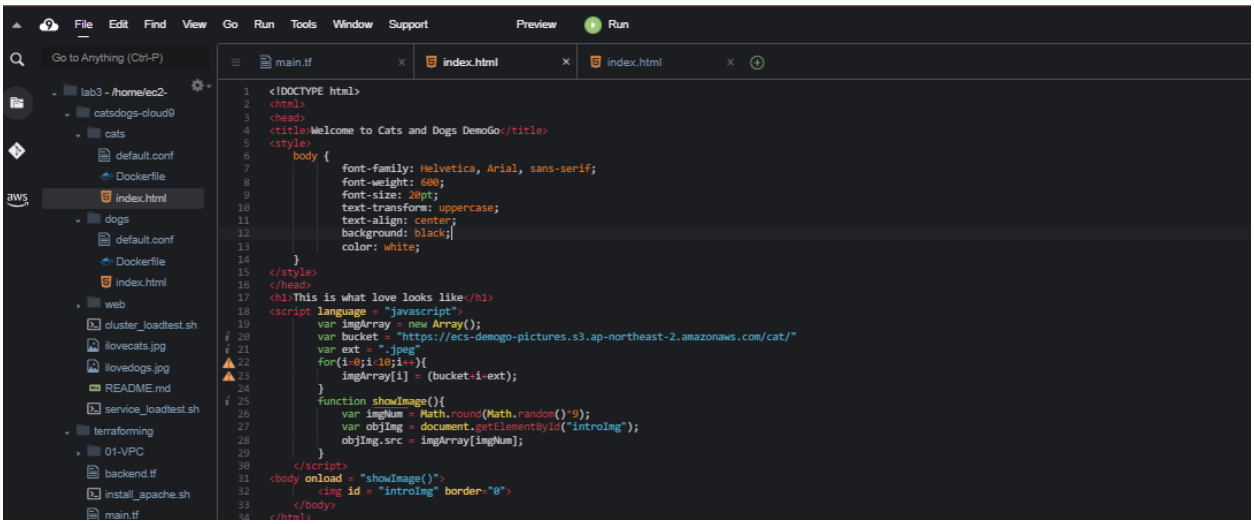


Fall 2021

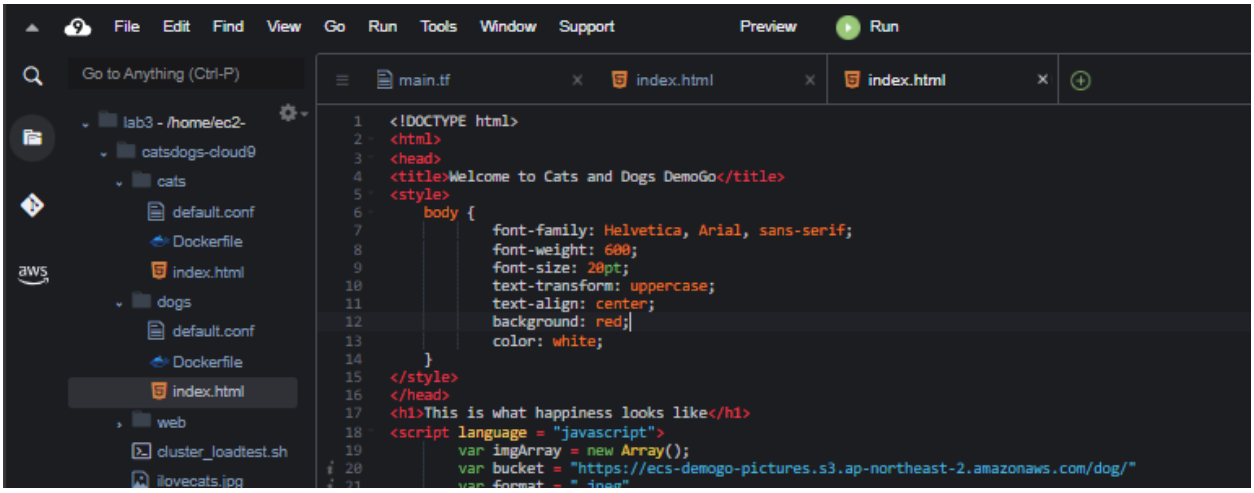
Part 3 – Update HTML background and deploy an updated version of the application

1. Change the cats and dogs HTML to use a different background. Color

Changing Cats HTML to a black background



Changing Dogs HTML to a red background



2. Repeat steps 2 to 11 from the Part 2

Rebuilding Cats docker image

```
voclabs:~/environment/catsdogs-cloud9/cats $ docker build -t cats .
Sending build context to Docker daemon  4.608kB
Step 1/6 : FROM nginx
 ---> ea335eea17ab
Step 2/6 : EXPOSE 80
 ---> Using cache
 ---> a11fa492a14d
Step 3/6 : RUN apt-get update -y &&   apt-get upgrade -y &&   apt-get install -y curl &&   cd /tmp &&   apt-get install awscli -y &&   rm -rf /tmp/* &&   rm -rf /var/lib/apt/lists/*
 ---> Using cache
 ---> 9698f8816743
Step 4/6 : COPY ./default.conf /etc/nginx/conf.d/default.conf
 ---> Using cache
 ---> 395084918606
Step 5/6 : COPY ./index.html /usr/share/nginx/html/index.html
 ---> 9a75784611ce
Step 6/6 : CMD nginx -g "daemon off;"
 ---> Running in 1124c15ca873
Removing intermediate container 1124c15ca873
 ---> 3a54aabcbb4a
Successfully built 3a54aabcbb4a
Successfully tagged cats:latest
voclabs:~/environment/catsdogs-cloud9/cats $
```

Rebuilding Dogs Docker image

```
voclabs:~/environment/catsdogs-cloud9/dogs $ docker build -t dogs .
Sending build context to Docker daemon  4.608kB
Step 1/6 : FROM nginx
 ---> ea335eea17ab
Step 2/6 : EXPOSE 80
 ---> Using cache
 ---> a11fa492a14d
Step 3/6 : RUN apt-get update -y &&   apt-get upgrade -y &&   apt-get install -y curl &&   cd /tmp &&   apt-get install awscli -y &&   rm -rf /tmp/* &&   rm -rf /var/lib/apt/lists/*
 ---> Using cache
 ---> 9698f8816743
Step 4/6 : COPY ./default.conf /etc/nginx/conf.d/default.conf
 ---> Using cache
 ---> df2935ba2951
Step 5/6 : COPY ./index.html /usr/share/nginx/html/index.html
 ---> 9ad87e9a7d93
Step 6/6 : CMD nginx -g "daemon off;"
 ---> Running in 3a02a3e387af
Removing intermediate container 3a02a3e387af
 ---> 5e0a0b7e9003
Successfully built 5e0a0b7e9003
Successfully tagged dogs:latest
```

Tagging the cats and dogs docker containers.

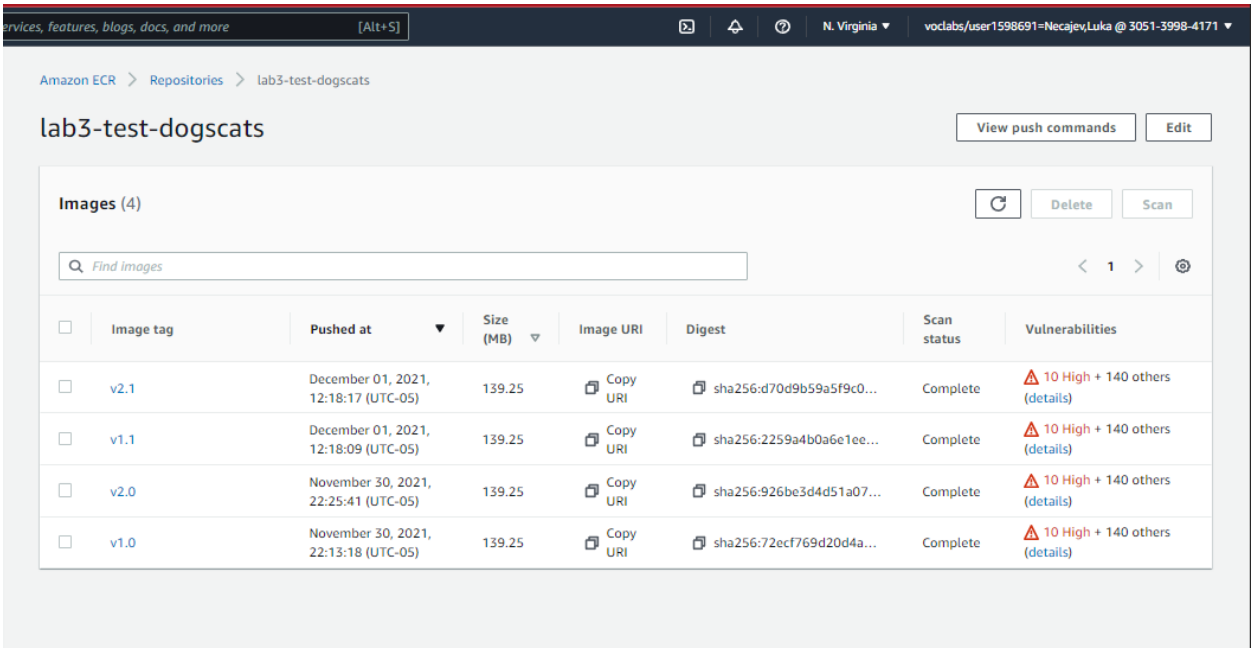
```
Push an image or a repository to a registry
voclabs:~/environment/catsdogs-cloud9/dogs $ docker tag dogs "305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v1.1"
voclabs:~/environment/catsdogs-cloud9/dogs $ docker tag cats "305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v2.1"
```

Pushing Cats and dogs to the ECR

```
voclabs:~/environment/catsdogs-cloud9/dogs $ aws ecr get-login-password --region us-east-1 | docker login -u AWS 305139984171.dkr.ecr.us-east-1.amazonaws.com --password-stdin
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
voclabs:~/environment/catsdogs-cloud9/dogs $ docker push "305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v1.1"
The push refers to repository [305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats]
18a5f4e3bdf1: Pushed
2b8d881a3c3d: Layer already exists
a6906e0adf1c: Layer already exists
8525cde30b22: Layer already exists
1e8ad06c81b6: Layer already exists
49eeddd2150f: Layer already exists
ff4c72779430: Layer already exists
37380c5830fe: Layer already exists
e1bbcf243d0e: Layer already exists
v1.1: digest: sha256:2259a4b0a6e1ee4c93568233df8d41b830d223b5023da5fedc97b437e1e9ea09 size: 2196
voclabs:~/environment/catsdogs-cloud9/dogs $ docker push "305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v2.1"
The push refers to repository [305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats]
a24721c1e463: Pushed
34662fbc7a81: Layer already exists
a6906e0adf1c: Layer already exists
8525cde30b22: Layer already exists
1e8ad06c81b6: Layer already exists
49eeddd2150f: Layer already exists
ff4c72779430: Layer already exists
37380c5830fe: Layer already exists
e1bbcf243d0e: Layer already exists
v2.1: digest: sha256:d70d9b59a5f9c08c12a38e76362d81332811b1dc93d53264909dc46b3933fc10 size: 2196
voclabs:~/environment/catsdogs-cloud9/dogs $
```

Confirming that its pushed to the ECR



Running container with the new image

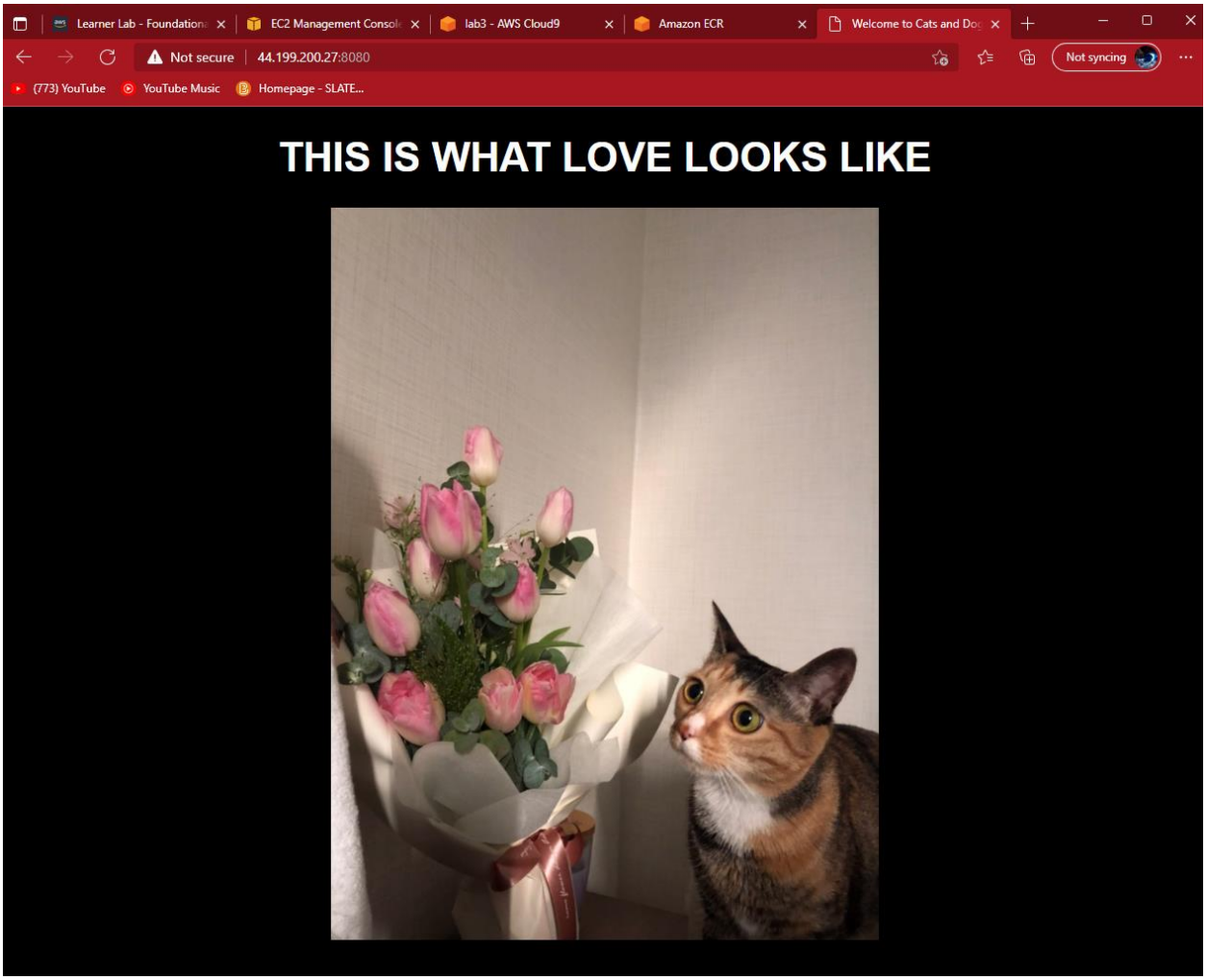
Stopping old instances

```
[ec2-user@ip-10-1-0-63 ~]$ docker stop 3cfb9c981343
^C
[ec2-user@ip-10-1-0-63 ~]$ docker stop 605cf02f99a3
^C
```

Starting new ones

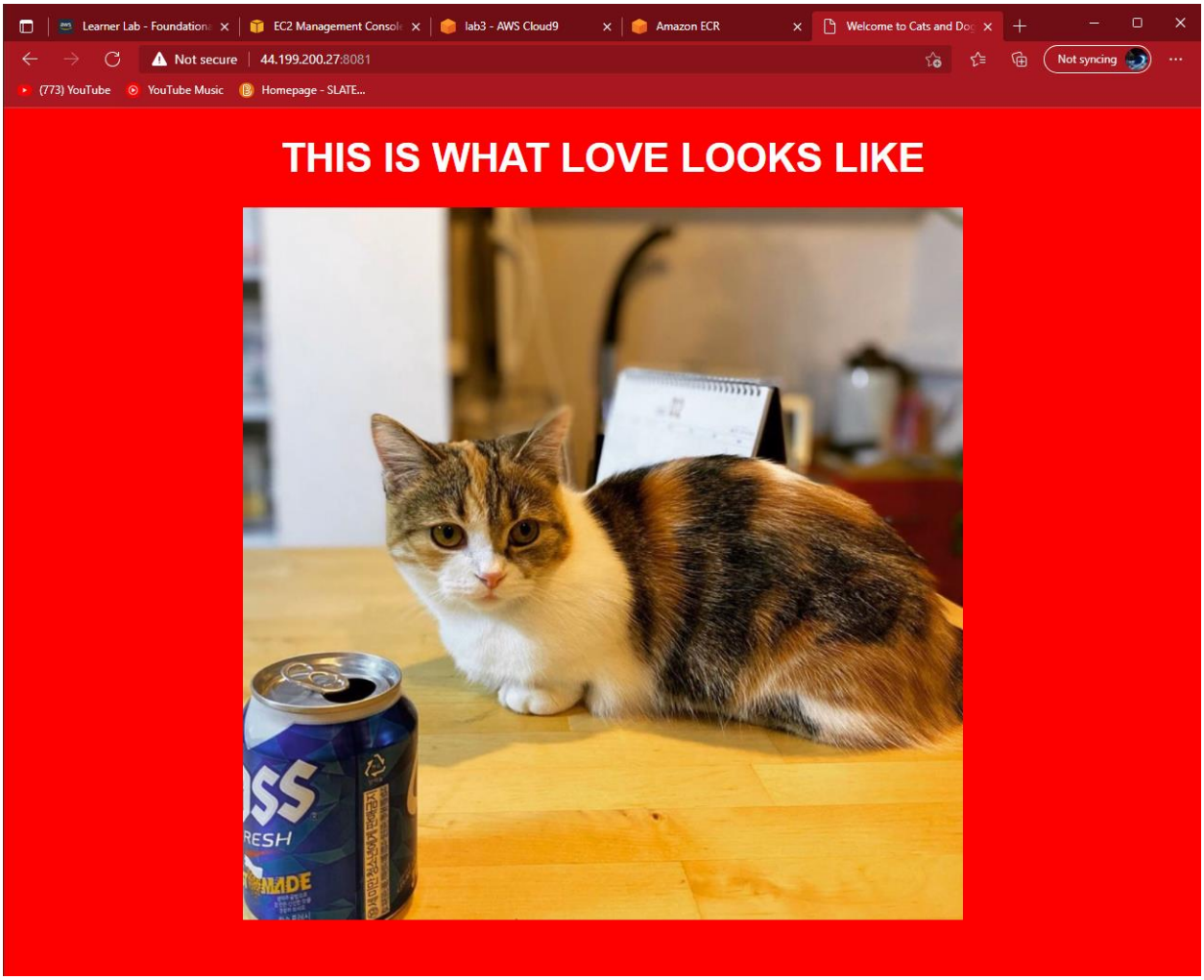
```
[ec2-user@ip-10-1-0-63 ~]$ docker run -it -p 8081:80 -d 305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v1.1
9a2a90d06206f4782f7b9875cad4c725efad1f8a9a9126e5d1bf7a30364dcc36
[ec2-user@ip-10-1-0-63 ~]$ docker run -it -p 8080:80 -d 305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v2.1
f060559fc94fa91a1207521a2d6364c64bd53931d0d189992bd7b0b035b9ccd3
[ec2-user@ip-10-1-0-63 ~]$ docker ps
CONTAINER ID   IMAGE                                                                  COMMAND                  CREATED
STATUS          PORTS                                   NAMES
f060559fc94f   305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v2.1   "/docker-entrypoint.…"   6 seconds ago
Up 5 seconds    0.0.0.0:8080->80/tcp, :::8080->80/tcp   vibrant_villani
9a2a90d06206   305139984171.dkr.ecr.us-east-1.amazonaws.com/lab3-test-dogscats:v1.1   "/docker-entrypoint.…"   16 seconds ago
Up 14 seconds   0.0.0.0:8081->80/tcp, :::8081->80/tcp   cool_brown
[ec2-user@ip-10-1-0-63 ~]$
```

SYST35144: Cloud Systems



Fall 2021

SYST35144: Cloud Systems



Part 4 (BONUS!) – Make the applWication show different pictures from your S3 bucket (e.g., trees and flowers in place of cats and dogs)

1. Create an S3 bucket with the appropriate permissions

Added the photos into S3 Bucket

Changed S3 bucket Policy to allow all public Access

Edited the HTML Page to look for my bucket of

2. Upload pictures of trees and flowers to the S3 bucket
3. Update the application to use your bucket and your pictures
4. Repeat steps 2 to 11 from Part 2

Grading

Task	Artifacts to Submit	Points
Part 1	• Deployable Terraform code as a zip file • Screenshot of AWS Console with VPC, Subnets, Route Tables, Security Group and EC2 details • Explanation of the SG • Explanation of IAM role assignment to EC2 instance - why do we need it and what it allowed EC2 to do.	40
Part 2	• Screenshots of all docker commands used • Browser screenshots with the running application • Explain: Why did we need to push the images to ECR?	40
Part 3	• Screenshots of all docker commands used • Browser screenshots with the running application	20
Part 4 BONUS	• Code of the updated application • Screenshots of all docker commands used • Browser screenshots with the running application	20
Total		120