Managed Cloud Services

When you don’t want to run it yourself
Managed Docker Repository

Elastic Container Service Repository (ECS Repository)
ECS Repository

Store our Docker Images in the Cloud

• What if we want to store our built docker image somewhere other than our laptop?

• What if we don’t want our image to be “public” on hub.docker.com?

• AWS has a managed Docker Image Repository: ECS Repository
ECS Repository

- Get into your AWS account
- Search for “ECS”
ECS Repository

- Get into your AWS account
- Search for “ECS”
ECS Repository

- Create a private repository
ECS Repository

• Create a private repository

• Now we can push docker images from our laptop to this repository

• From there, we can pull them down to an EC2 instance, or to Elastic Container Service to run
ECS Repository

• Create a private repository

• Now we can push docker images from our laptop to this repository

• From there, we can pull them down to an EC2 instance, or to Elastic Container Service to run

• View the push commands
ECS Repository

- There’s a really great “AWS Toolkit” extension for VS Code that Amazon supports.
ECS Repository

• There’s a really great “AWS Toolkit” extension for VS Code that Amazon supports

• Clicking on the “AWS” in the window footer will bring up the AWS commands

• Easily access your credentials file
ECS Repository

- In order to push images to ECR, you need to have current AWS IAM credentials
- Copy them from the AWS Academy site and update your credentials file
ECS Repository

• Build your image
ECS Repository

• Build your image
• Login to ECR
ECS Repository

- Build your image
- Login to ECR
- Tag your local image with the ECR host name that matches your repository
  - This is what tells the docker push command where to send your image

```bash
```
ECS Repository

- Build your image
- Login to ECR
- Tag your local image with the ECR host name that matches your repository
- Push your image up to ECR

```bash
docker push 561707296892.dkr.ecr.us-east-1.amazonaws.com/csc346-chat-app:latest
```
ECS Repository

How do we get our image back out to EC2?

- We still need permissions on our EC2 instance to pull an image back down
- We could copy IAM credentials to our EC2 host just like we do for our laptop
- However within AWS you can leverage IAM Roles
- A role defines a set of permissions that an actor can take on resources
  - We can attach an Role Profile to our instance
ECS Repository

How do we get our image back out to EC2?
How do we get our image back out to EC2?
ECS Repository

• With an IAM role attached we can now do our docker login on the EC2 instance
ECS Repository

- Oh noes! 😱 We have a bad image platform
- Image was built on an arm64 Mac. EC2 is amd64 based Intel.

```
[ec2-user@ip-172-31-84-94 ~]$ sudo docker run -d --rm -p80:80 56107396892.dkr.ecr.us-east-1.amazonaws.com/csc346-chat-app:latest
WARNING: The requested image's platform (linux/arm64/v8) does not match the detected host platform (linux/amd64) and no specific platform was requested
396947640ee76144276c4d2b2f30915edbf60b82d6c078c2ca7a3bb70690827
[ec2-user@ip-172-31-84-94 ~]$ 
```
ECS Repository

- You can build an image for a different architecture by specifying the `--platform` option.
ECS Repository

- Build, tag, push the updated image
- Now we can run the image on our EC2 instance directly from the ECR repository
More Automation

• Combine with CloudFormation to automatically login and start the image at boot time

```yaml
AssociatePublicIpAddress: !If [AssignPublicIPCondition, true, !Ref "AWS::NoValue"]
Tags:
  - Key: "Name"
    Value: !Ref "HostName"
UserData:
  Fn::Base64: !Sub /
  #!/bin/bash -e
  #
  # Basic Updates
  sudo yum update -y
  sudo yum install -y git vim docker
  sudo systemctl enable docker
  sudo systemctl start docker
  sudo aws ecr get-login-password --region us-east-1 | sudo docker login --username AWS --password-stdin
  sudo docker run --rm 561707296892.dkr.ecr.us-east-1.amazonaws.com/csc346-chat-app:latest

# #### Instance Security Group
#
# Security group for the EC2 instance, that allows you to SSH into the instance
InstanceSecurityGroup:
  Type: "AWS::EC2::SecurityGroup"
  Properties:
    GroupDescription: "Allow ssh to client host"
```