Managed Cloud Services

When you don't want to run it yourself

Managed Docker Repository

Elastic Container Service Repository (ECS Repository)

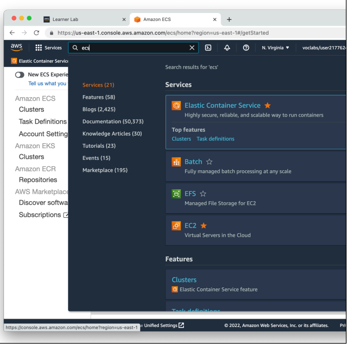
ECS Repository

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

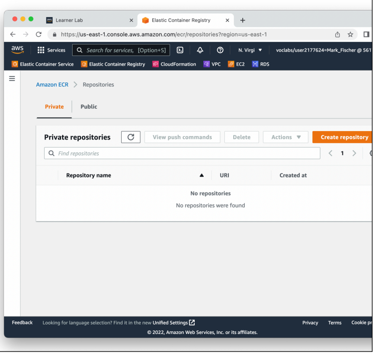
ECS Repository

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



ECS Repository

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

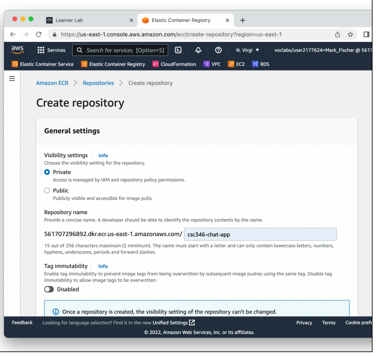


No repositories

No repositories were found

ECS Repository

- Create a private repository


Learner Lab
Elastic Container Registry
https://us-east-1.console.aws.amazon.com/ecr/create-repository?region=us-east-1
Services
Search for services, [Option+S]
N. Virgi
Elastic Container Service
Elastic Container Registry
CloudFormation
VPC
EC2
RDS
Amazon ECR
Repositories
Create repository
Feedback
Privacy
Terms

Create repository

General settings

Visibility settings Info

Choose the visibility setting for the repository.

[✓ Private] Access is managed by IAM and repository policy permissions.

Public — Publicly visible and accessible for image pulls.

Repository name

Provide a concise name. A developer should be able to identify the repository contents by the name.

561707296892.dkr.ecr.us-east-1.amazonaws.com/ csc346-chat-app

15 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

Tag immutability Info

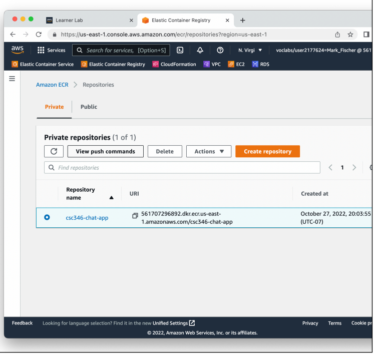
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

Disabled

Once a repository is created, the visibility setting of the repository can't be changed.

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


Learner Lab
Elastic Container Registry
https://us-east-1.console.aws.amazon.com/ecr/repositories?region=us-east-1
Services
Search for services, [Option+S]
N. Virgi
voclabs/user2177624=Mark_Fischer @ 561
Elastic Container Service
Elastic Container Registry
CloudFormation
VPC
EC2
RDS
Amazon ECR
Repositories
Private
Public

Private repositories (1 of 1)

View push commands
Delete
Actions
Create repository
Find repositories

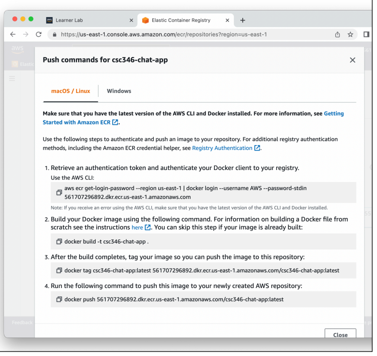
Repository name	URI	Created at
csc346-chat-app	561707296892.dkr.ecr.us-east-1.amazonaws.com/csc346-chat-app	October 27, 2022, 20:03:55 (UTC-07)

Feedback
Looking for language selection? Find it in the new Unified Settings
Privacy
Terms
Cookie pr

© 2022, Amazon Web Services, Inc. or its affiliates.

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


Learner Lab
Elastic Container Registry
https://us-east-1.console.aws.amazon.com/ecr/repositories?region=us-east-1
macOS / Linux
Windows
Close

Push commands for csc346-chat-app

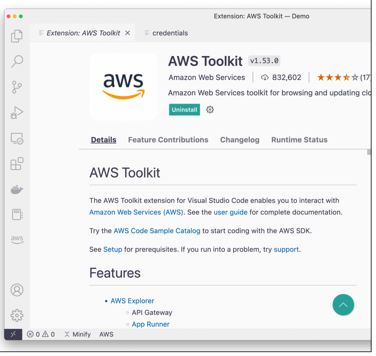
Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see Getting Started with Amazon ECR.

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see Registry Authentication.

1. Retrieve an authentication token and authenticate your Docker client to your registry.
   Use the AWS CLI:
   ```
   aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin
   561707296892.dkr.ecr.us-east-1.amazonaws.com
   ```
   Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.
2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions here. You can skip this step if your image is already built:
   ```
   docker build -t csc346-chat-app .
   ```
3. After the build completes, tag your image so you can push the image to this repository:
   ```
   docker tag csc346-chat-app:latest 561707296892.dkr.ecr.us-east-1.amazonaws.com/csc346-chat-app:latest
   ```
4. Run the following command to push this image to your newly created AWS repository:
   ```
   docker push 561707296892.dkr.ecr.us-east-1.amazonaws.com/csc346-chat-app:latest
   ```

ECS Repository

- There's a really great "AWS Toolkit" extension for VS Code that Amazon supports



Amazon Web Services toolkit for browsing and updating clo

AWS Toolkit

The AWS Toolkit extension for Visual Studio Code enables you to interact with Amazon Web Services (AWS). See the user guide for complete documentation.

Try the AWS Code Sample Catalog to start coding with the AWS SDK.

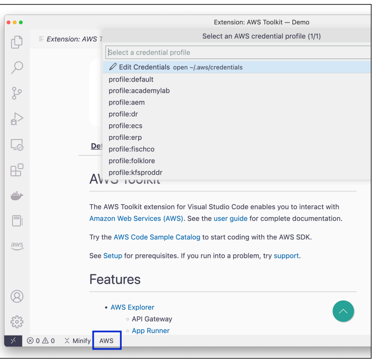
See Setup for prerequisites. If you run into a problem, try support.

Features

- AWS Explorer
  - API Gateway
  - App Runner

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


Select an AWS credential profile (1/1)

```
Edit Credentials open ~/.aws/credentials
profile:default
profile:academylab
profile:aem
profile:dr
profile:ecs
profile:erp
profile:fischco
profile:folklore
profile:kfsproddr
```

The AWS Toolkit extension for Visual Studio Code enables you to interact with Amazon Web Services (AWS). See the user guide for complete documentation.

Try the AWS Code Sample Catalog to start coding with the AWS SDK.

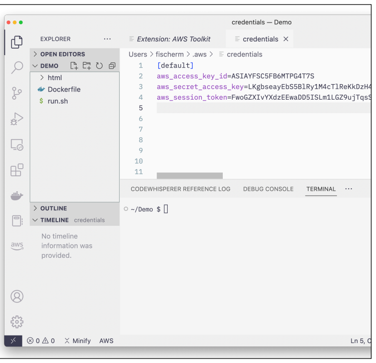
See Setup for prerequisites. If you run into a problem, try support.

Features

- AWS Explorer
  - API Gateway
  - App Runner

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

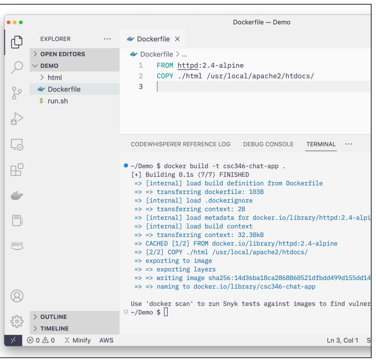

Users > fischerm > .aws > credentials

```
[default]
aws_access_key_id=ASIAYFSC5FB6MTPG4T7S
aws_secret_access_key=LKgbseayEbS5BlRy1M4cTlReKkDzH4
aws_session_token=FwoGZXIvYXdzEEwaDD5ISLm1LGZ9ujTqsS
```

```
~/Demo $ 
```

ECS Repository

- Build your image


Dockerfile

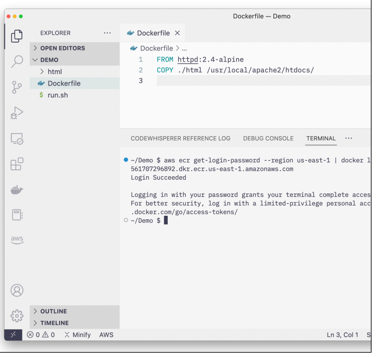
```
FROM httpd:2.4-alpine
COPY ./html /usr/local/apache2/htdocs/
```

```
~/Demo $ docker build -t csc346-chat-app .
[+] Building 0.1s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 103B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/httpd:2.4-alpi
 => [internal] load build context
 => => transferring context: 32.38kB
 => CACHED [1/2] FROM docker.io/library/httpd:2.4-alpine
 => [2/2] COPY ./html /usr/local/apache2/htdocs/
 => exporting to image
 => => exporting layers
 => => writing image sha256:14d36ba18ca2868860521dfbdd499d155dd14
 => => naming to docker.io/library/csc346-chat-app

Use 'docker scan' to run Snyk tests against images to find vulner
~/Demo $ 
```

ECS Repository

- Build your image
- Login to ECR
-



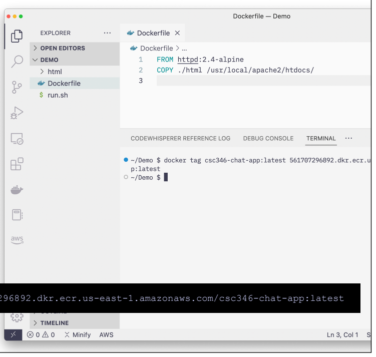
```
FROM httpd:2.4-alpine
COPY ./html /usr/local/apache2/htdocs/
```

```
~/Demo $ aws ecr get-login-password --region us-east-1 | docker l
561707296892.dkr.ecr.us-east-1.amazonaws.com
Login Succeeded

Logging in with your password grants your terminal complete acces
For better security, log in with a limited-privilege personal acc
.docker.com/go/access-tokens/
~/Demo $ 
```

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



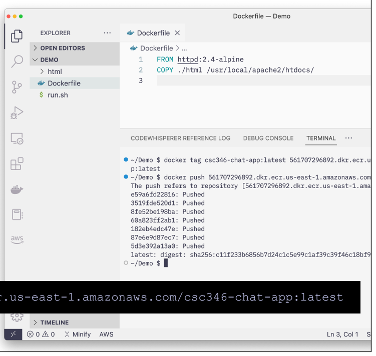
```
FROM httpd:2.4-alpine
COPY ./html /usr/local/apache2/htdocs/
```

```
~/Demo $ docker tag csc346-chat-app:latest 561707296892.dkr.ecr.u
p:latest
~/Demo $ 
```

```
docker tag csc346-chat-app:latest 561707296892.dkr.ecr.us-east-1.amazonaws.com/csc346-chat-app:latest
```

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



```
FROM httpd:2.4-alpine
COPY ./html /usr/local/apache2/htdocs/
```

```
~/Demo $ docker tag csc346-chat-app:latest 561707296892.dkr.ecr.u
p:latest
~/Demo $ docker push 561707296892.dkr.ecr.us-east-1.amazonaws.com
The push refers to repository [561707296892.dkr.ecr.us-east-1.ama
e59a6fd22816: Pushed
3519fde520d1: Pushed
8fe52be198ba: Pushed
60a823ff2ab1: Pushed
182eb4edc47e: Pushed
87e6e9d87ec7: Pushed
5d3e392a13a0: Pushed
latest: digest: sha256:c11f233b6856b7d24c1c5e99c1af39c39f46c18bf9
~/Demo $ 
```

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

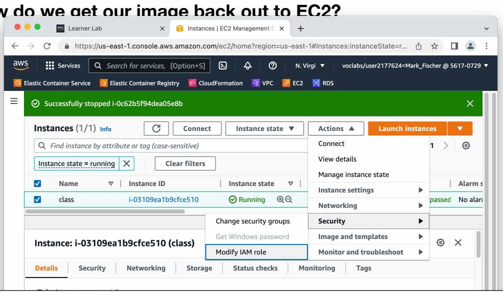
ECS Repository

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?


Learner Lab | Instances | EC2 Management C

https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:instanceState=r...

Successfully stopped i-0c62b5f94dea05e8b

Instances (1/1) Info

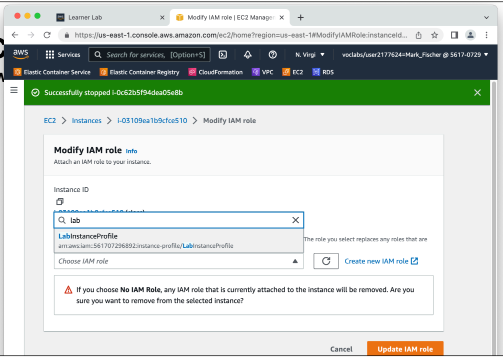
Instance state = running

Name	Instance ID	Instance state		Alarm s
class	i-03109ea1b9cfce510	Running	passed	No aları

Instance: i-03109ea1b9cfce510 (class)

EC

How


Learner Lab | Modify IAM role | EC2 Managen

https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyIAMRole:instanceId...

Successfully stopped i-0c62b5f94dea05e8b

EC2 > Instances > i-03109ea1b9cfce510 > Modify IAM role

Modify IAM role Info

Attach an IAM role to your instance.

Instance ID

lab

LabInstanceProfile
arn:aws:iam::561707296892:instance-profile/LabInstanceProfile

The role you select replaces any roles that are

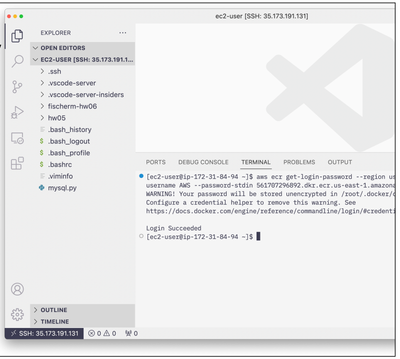
Choose IAM role

If you choose **No IAM Role**, any IAM role that is currently attached to the instance will be removed. Are you sure you want to remove from the selected instance?

Services
Search for services, [Option+S]
N. Virgi
voclabs/user2177624=Mark_Fischer @ 5617-0729
Elastic Container Service
Elastic Container Registry
CloudFormation
VPC
EC2
RDS
Create new IAM role
Cancel
Update IAM role

ECS Repository

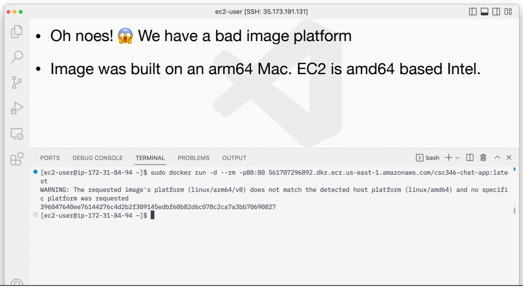
- With an IAM role attached we can now do our docker login on the EC2 instance



```
[ec2-user@ip-172-31-84-94 ~]$ aws ecr get-login-password --region us
username AWS --password-stdin 561707296892.dkr.ecr.us-east-1.amazona
WARNING! Your password will be stored unencrypted in /root/.docker/c
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credenti

Login Succeeded
[ec2-user@ip-172-31-84-94 ~]$ 
```

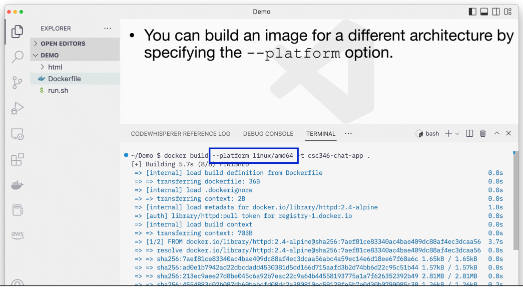
ECS Repository



- 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 561707296892.dkr.ecr.us-east-1.amazonaws.com/csc346-chat-app:late
st
WARNING: The requested image's platform (linux/arm64/v8) does not match the detected host platform (linux/amd64) and no specifi
c platform was requested
396047640ee76144276c4d2b2f309145edbf60b82d6c078c2ca7a3bb70690827
[ec2-user@ip-172-31-84-94 ~]$ 
```

ECS Repository

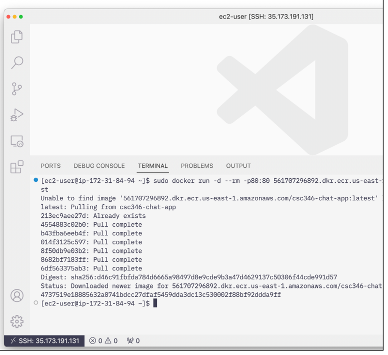


- You can build an image for a different architecture by specifying the `--platform` option.

```
~/Demo $ docker build --platform linux/amd64 -t csc346-chat-app .
[+] Building 5.7s (8/8) FINISHED
 => [internal] load build definition from Dockerfile                                  0.0s
 => => transferring dockerfile: 36B                                                   0.0s
 => [internal] load .dockerignore                                                     0.0s
 => => transferring context: 2B                                                       0.0s
 => [internal] load metadata for docker.io/library/httpd:2.4-alpine                   1.8s
 => [auth] library/httpd:pull token for registry-1.docker.io                          0.0s
 => [internal] load build context                                                     0.0s
 => => transferring context: 703B                                                     0.0s
 => [1/2] FROM docker.io/library/httpd:2.4-alpine@sha256:7aef81ce83340ac4bae409dc88af4ec3dcaa56  3.7s
 => => resolve docker.io/library/httpd:2.4-alpine@sha256:7aef81ce83340ac4bae409dc88af4ec3dcaa56  0.0s
 => => sha256:7aef81ce83340ac4bae409dc88af4ec3dcaa56abc4a59ec14e6d18ee67f68a6c 1.65kB / 1.65kB   0.0s
 => => sha256:ad0e1b7942ad22dbcdadd4530381d5dd166d715aafd3b2d74bb6d22c95c51b44 1.57kB / 1.57kB   0.0s
 => => sha256:213ec9aee27d8be045c6a92b7eac22c9a64b44558193775a1a7f626352392b49 2.81MB / 2.81MB   0.8s
 => => sha256:4554883c02b087db69babcfd00dc2a380810ec59129fe5b7e0d30b0799085c38 1.26kB / 1.26kB   0.2s
```

ECS Repository

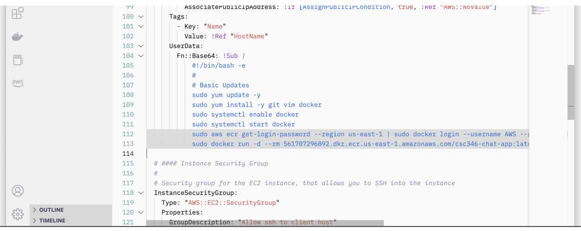
- Build, tag, push the updated image
- Now we can run the image on our EC2 instance directly from the ECR repository



```
[ec2-user@ip-172-31-84-94 ~]$ sudo docker run -d --rm -p80:80 561707296892.dkr.ecr.us-east-
st
Unable to find image '561707296892.dkr.ecr.us-east-1.amazonaws.com/csc346-chat-app:latest'
latest: Pulling from csc346-chat-app
213ec9aee27d: Already exists
4554883c02b0: Pull complete
b43fba6eeb4f: Pull complete
014f3125c597: Pull complete
8f50db9e03b2: Pull complete
8682bf7183ff: Pull complete
6df563375ab3: Pull complete
Digest: sha256:d46c91fbfda784d6665a98497d8e9cde9b3a47d4629137c50306f44cde991d57
Status: Downloaded newer image for 561707296892.dkr.ecr.us-east-1.amazonaws.com/csc346-chat
4737519e18885632a0741bdcc27dfaf5459dda3dc13c530002f88bf92ddda9ff
[ec2-user@ip-172-31-84-94 ~]$
```

More Automation

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



```
        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 --p
            sudo docker run -d --rm 561707296892.dkr.ecr.us-east-1.amazonaws.com/csc346-chat-app:late

  # #### 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"
```