

Managed Cloud Services

When you don't want to run it yourself

Managed Docker Repository

Elastic Container Service Repository (ECS Repository)

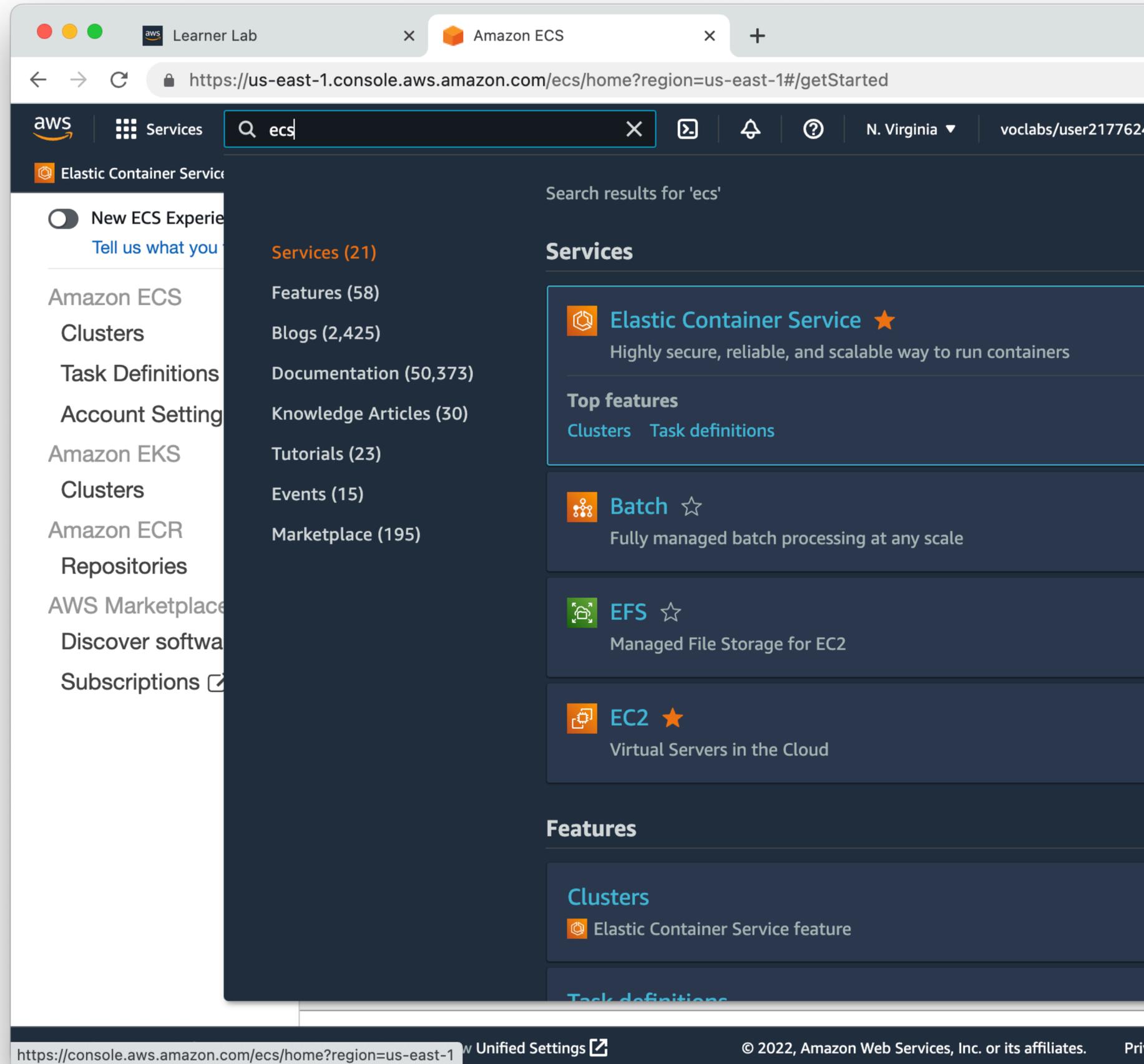
ECS Repository

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

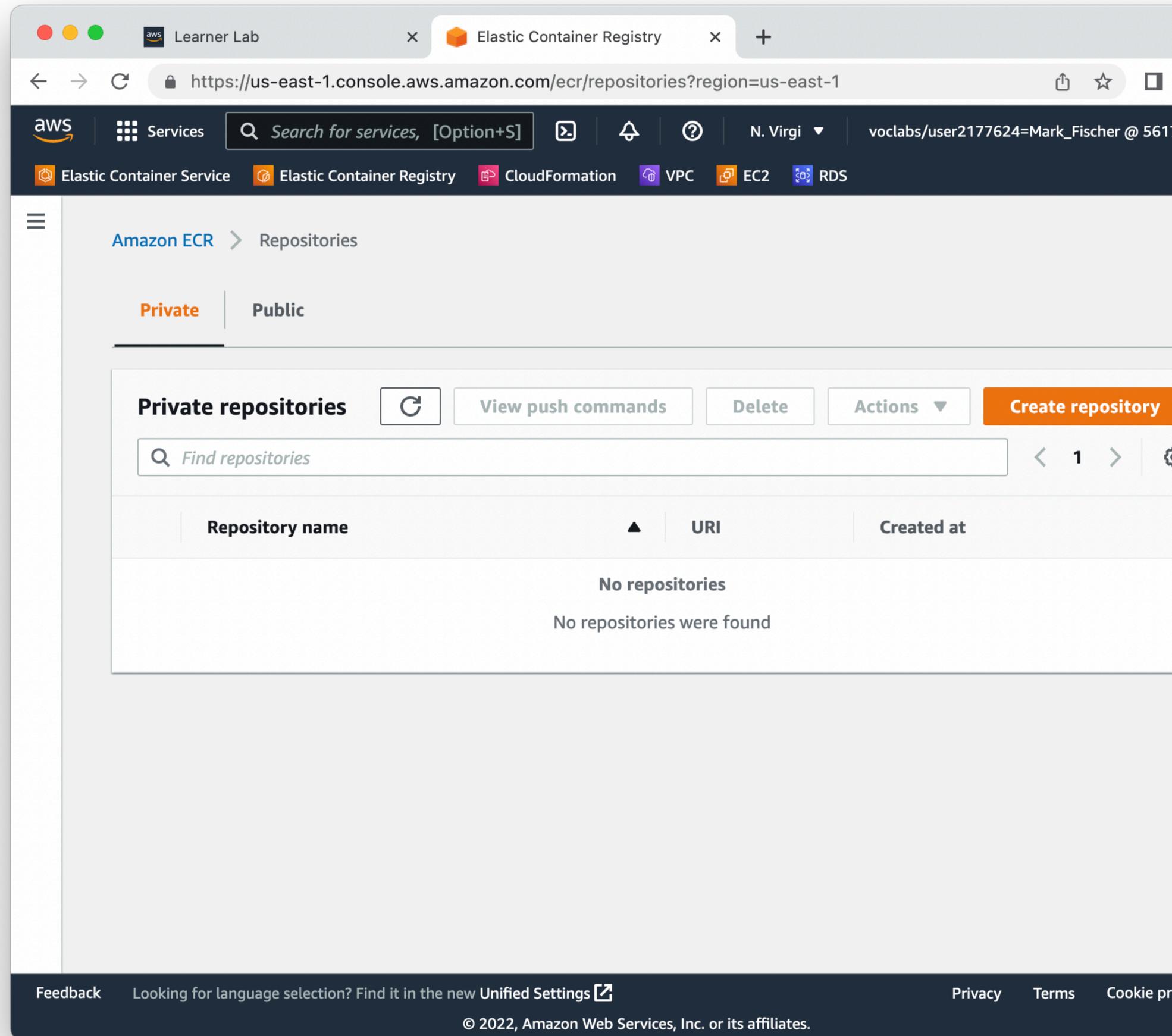
ECS Repository

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



ECS Repository

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



ECS Repository

- Create a private repository

The screenshot shows the AWS Elastic Container Registry (ECR) console in the 'us-east-1' region. The page is titled 'Create repository' and is part of the 'Amazon ECR > Repositories > Create repository' path. The 'General settings' section is visible, with the following details:

- Visibility settings:** The 'Private' option is selected, indicating that access is managed by IAM and repository policy permissions. The 'Public' option is unselected, which would make the repository publicly visible and accessible for image pulls.
- Repository name:** The name '561707296892.dkr.ecr.us-east-1.amazonaws.com/csc346-chat-app' is entered. A note below the field states: '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:** The 'Disabled' option is selected, meaning image tags can be overwritten by subsequent pushes. The 'Info' link indicates that enabling this feature would prevent overwrites.

A light blue information box at the bottom of the form states: 'Once a repository is created, the visibility setting of the repository can't be changed.'

The footer of the console includes 'Feedback', 'Looking for language selection? Find it in the new Unified Settings', 'Privacy', 'Terms', 'Cookie preferences', and '© 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

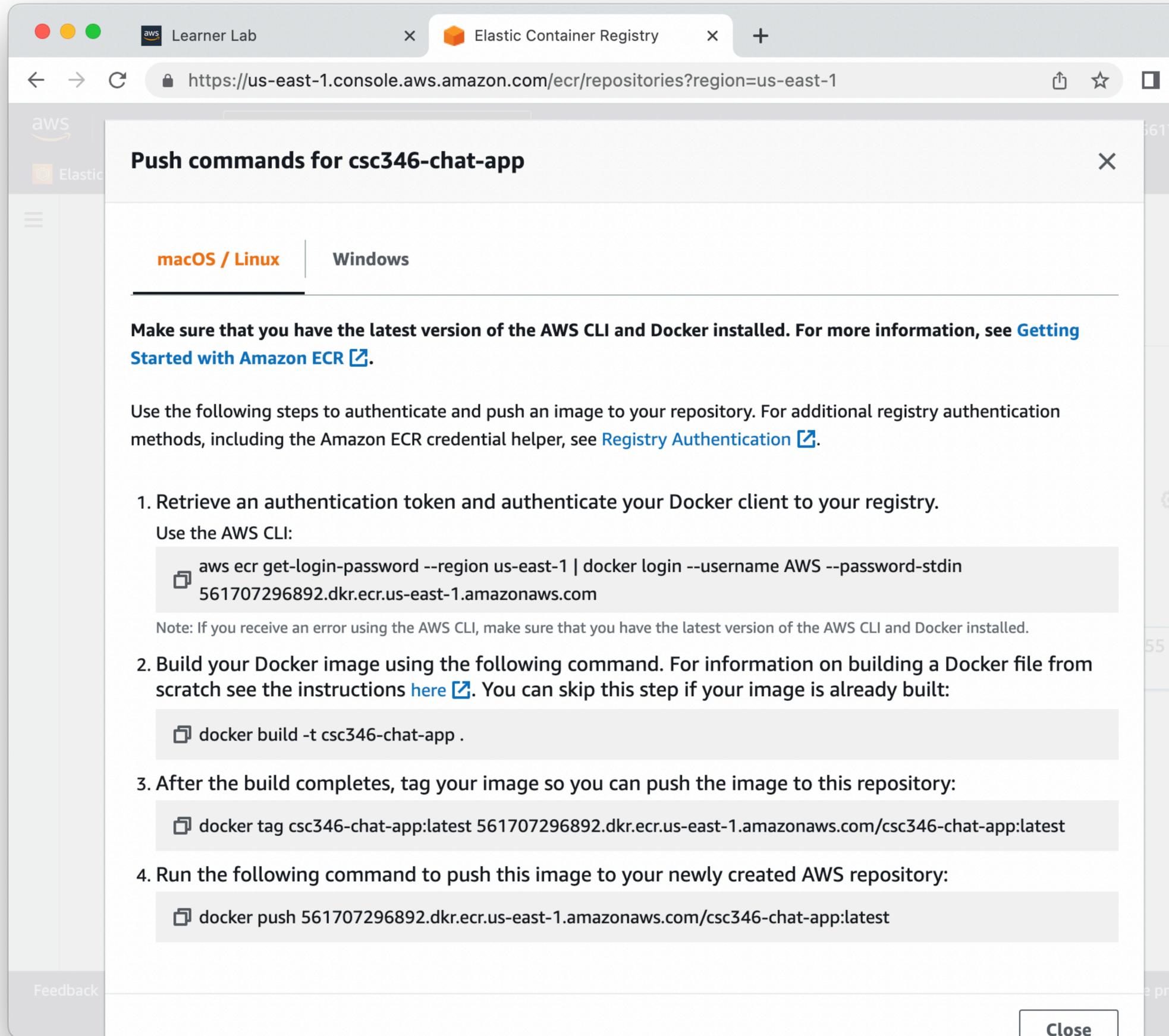
The screenshot shows the AWS Elastic Container Registry (ECR) console. The browser address bar indicates the URL: `https://us-east-1.console.aws.amazon.com/ecr/repositories?region=us-east-1`. The console header includes the AWS logo, a search bar, and navigation links for various services like Elastic Container Service, Elastic Container Registry, CloudFormation, VPC, EC2, and RDS. The main content area is titled "Amazon ECR > Repositories" and has tabs for "Private" and "Public". Under the "Private" tab, there is a section for "Private repositories (1 of 1)" with buttons for "View push commands", "Delete", "Actions", and "Create repository". Below this is a search bar labeled "Find repositories" and a table listing the repository.

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)

At the bottom of the console, there are links for "Feedback", "Looking for language selection? Find it in the new Unified Settings", "Privacy", "Terms", and "Cookie policy". The footer text reads: "© 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



The screenshot shows the AWS Elastic Container Registry console interface. The browser tabs include 'Learner Lab' and 'Elastic Container Registry'. The URL is `https://us-east-1.console.aws.amazon.com/ecr/repositories?region=us-east-1`. The main content area is titled 'Push commands for csc346-chat-app' and has two tabs: 'macOS / Linux' (selected) and 'Windows'. Below the tabs, there is a note: 'Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see [Getting Started with Amazon ECR](#).' This is followed by another note: '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](#).' The steps are:

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
```

At the bottom of the console, there is a 'Feedback' button on the left and a 'Close' button on the right.

ECS Repository

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

The screenshot shows the AWS Toolkit extension page in Visual Studio Code. The page title is "Extension: AWS Toolkit — Demo". The extension name is "AWS Toolkit" with version "v1.53.0". It is published by "Amazon Web Services" and has 832,602 downloads and a 4.5-star rating (17 reviews). The description states: "Amazon Web Services toolkit for browsing and updating cloud resources". There are "Uninstall" and "Settings" buttons. The page has tabs for "Details", "Feature Contributions", "Changelog", and "Runtime Status". The "Details" tab is active, showing the extension's description and features. The features listed are "AWS Explorer", "API Gateway", and "App Runner". The VS Code interface includes a sidebar with icons for Explorer, Search, Source Control, Run and Debug, Remote Explorer, Extensions, AWS Toolkit, and Settings. The status bar at the bottom shows "Minify" and "AWS".

Extension: AWS Toolkit — Demo

Extension: AWS Toolkit × credentials

AWS Toolkit v1.53.0

Amazon Web Services | 832,602 | ★★★★★ (17)

Amazon Web Services toolkit for browsing and updating cloud resources

Uninstall ⚙️

Details Feature Contributions Changelog Runtime Status

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.

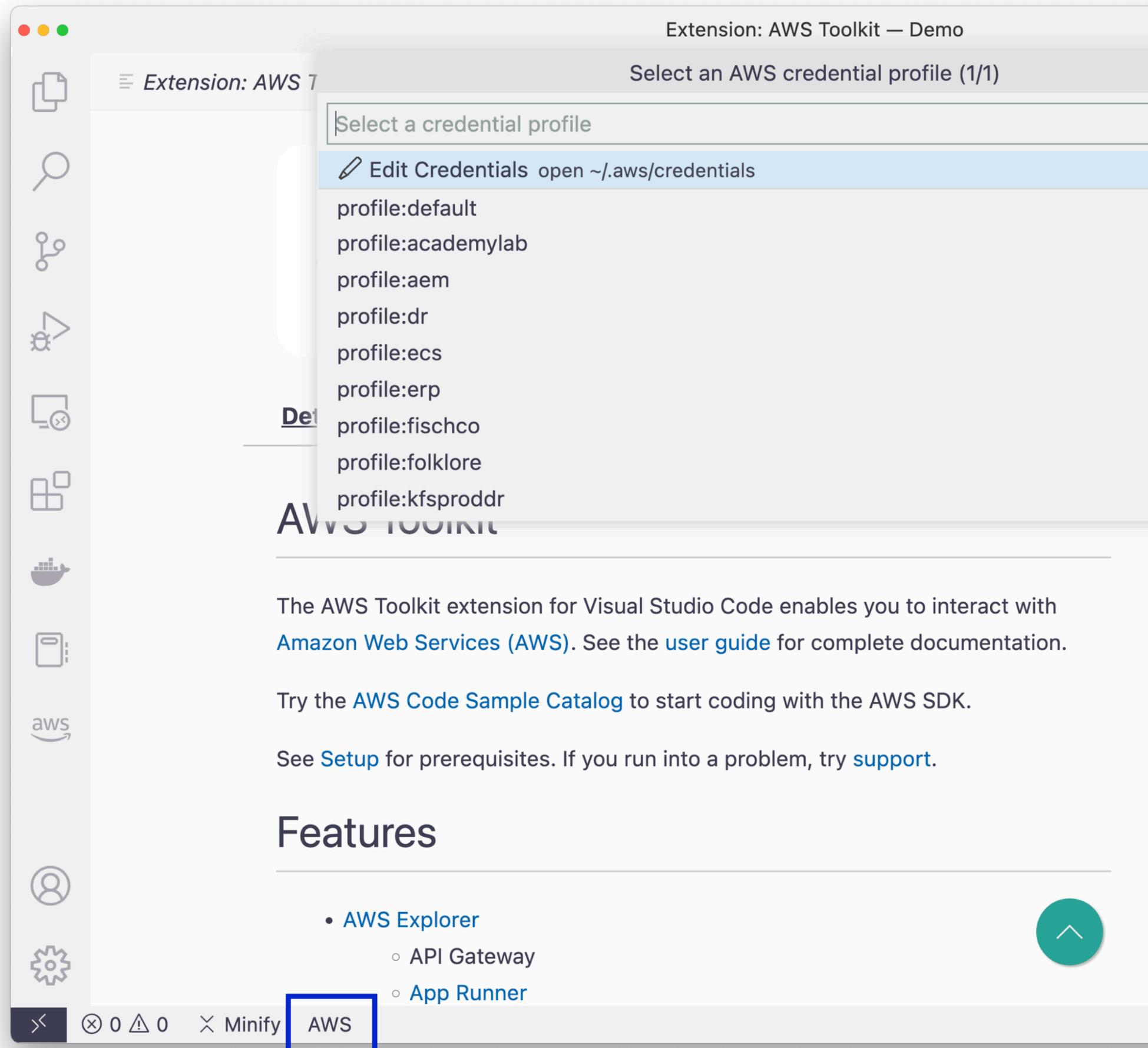
See [Setup](#) for prerequisites. If you run into a problem, try [support](#).

Features

- [AWS Explorer](#)
 - [API Gateway](#)
 - [App Runner](#)

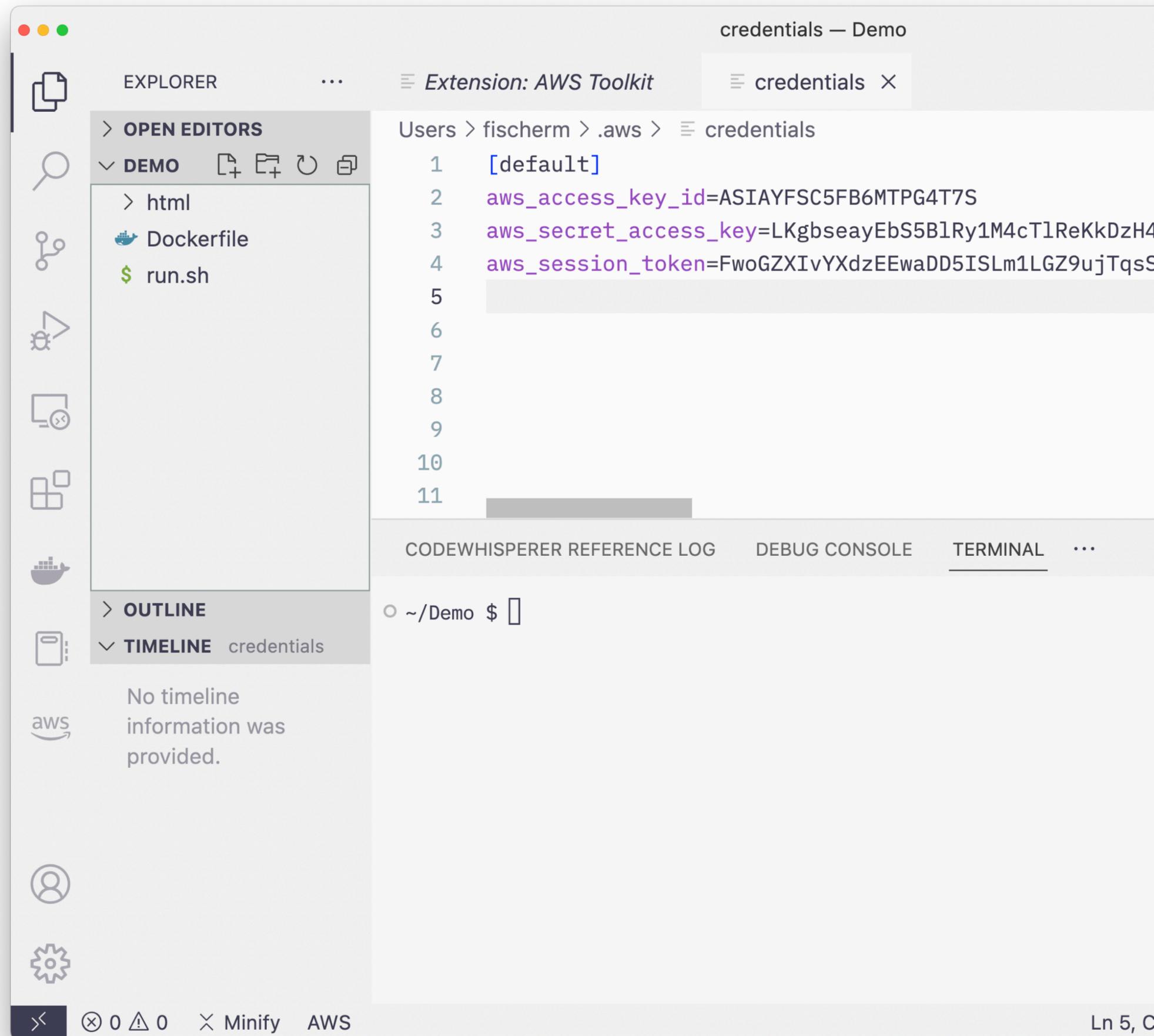
ECS Repository

- 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



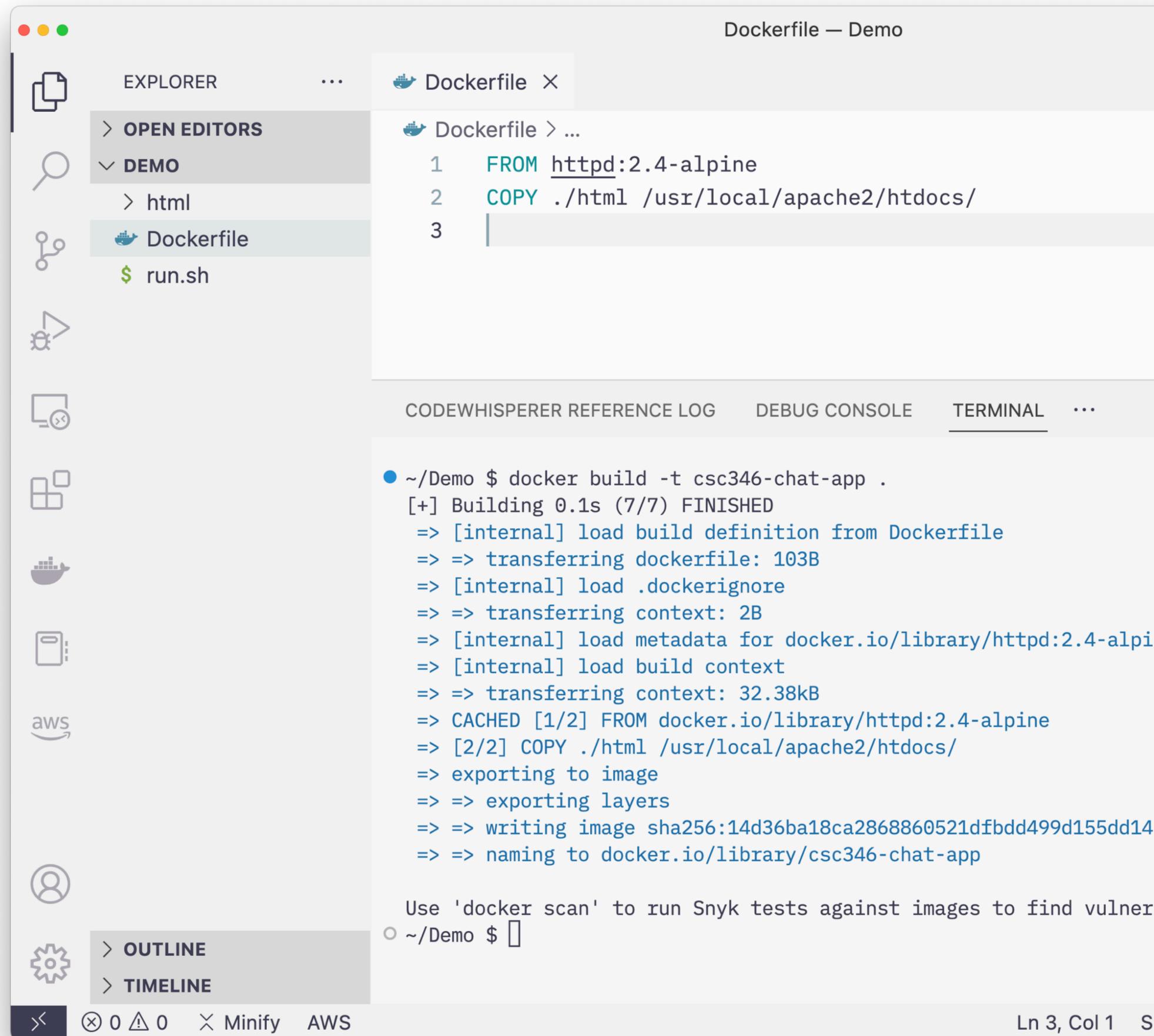
ECS Repository

- 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



The screenshot shows a Visual Studio Code editor window titled "Dockerfile — Demo". The Explorer sidebar on the left shows a project structure with folders "html" and "run.sh", and a file "Dockerfile". The Dockerfile content is as follows:

```
1 FROM httpd:2.4-alpine
2 COPY ./html /usr/local/apache2/htdocs/
3
```

The Terminal panel at the bottom shows the output of a Docker build command:

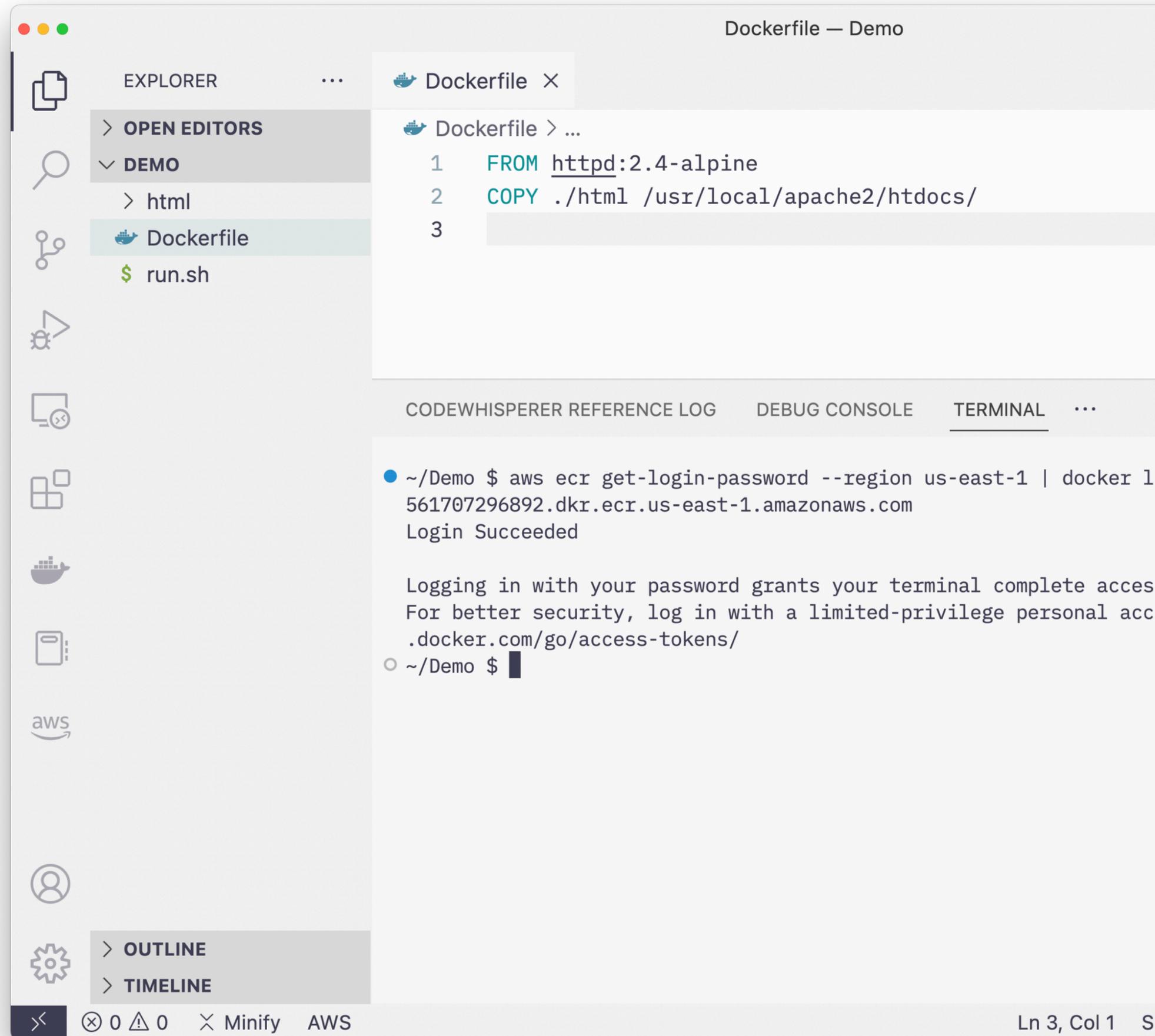
```
~/Demo $ docker build -t csc346-chat-app .
[+] Building 0.1s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 103B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/httpd:2.4-alpine
=> [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 $
```

The status bar at the bottom indicates "Ln 3, Col 1" and "AWS".

ECS Repository

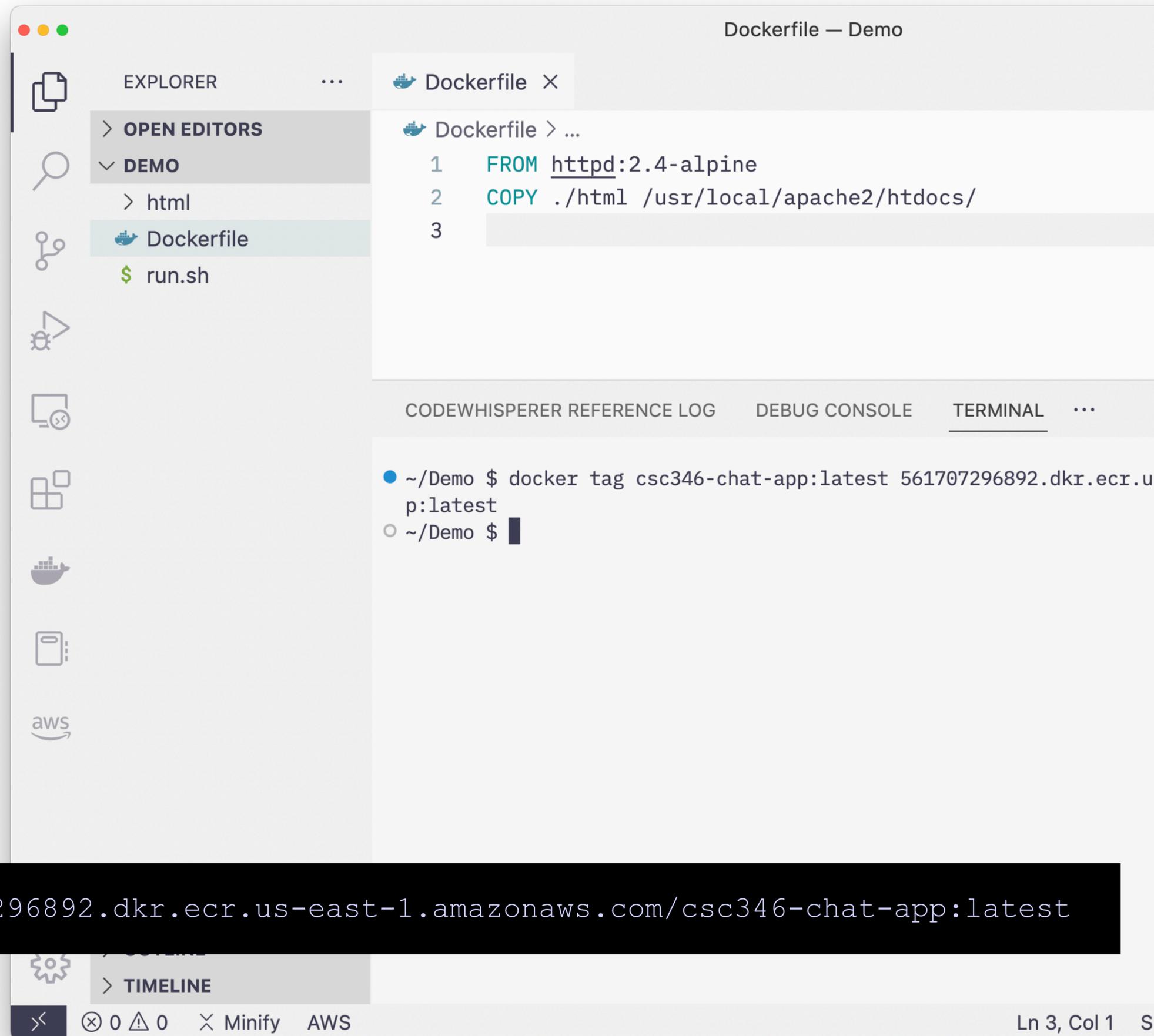
- Build your image
- Login to ECR
-



ECS Repository

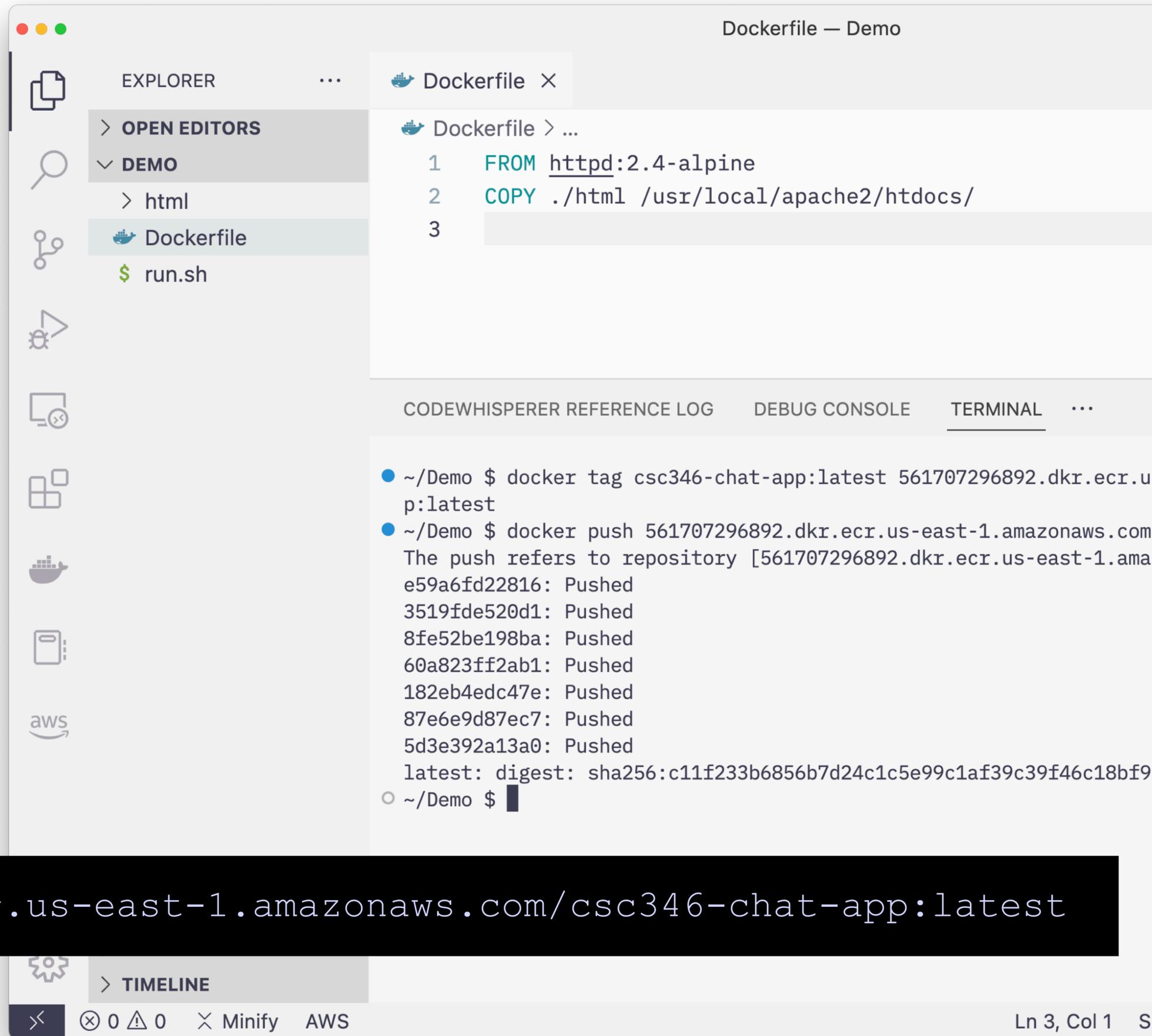
- Build your image
- Login to ECR
- Tag your local image with the ECR host name that matches your repository
 - This is what tells the `docker push` command where to send your image

```
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



The screenshot shows a Visual Studio Code editor window titled "Dockerfile — Demo". The Explorer sidebar on the left shows a project named "DEMO" with a subfolder "html" and a file "Dockerfile". The Dockerfile content is as follows:

```
1 FROM httpd:2.4-alpine
2 COPY ./html /usr/local/apache2/htdocs/
3
```

The Terminal panel at the bottom shows the following commands and output:

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

The status bar at the bottom indicates "Ln 3, Col 1" and "AWS".

```
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?

The screenshot shows the AWS Management Console interface. At the top, there is a navigation bar with the AWS logo, a search bar, and user information. Below this is a service menu with icons for Elastic Container Service, Elastic Container Registry, CloudFormation, VPC, EC2, and RDS. A green notification banner at the top left states "Successfully stopped i-0c62b5f94dea05e8b". The main content area displays the "Instances (1/1)" page. A table lists one instance named "class" with ID "i-03109ea1b9cfce510" in a "Running" state. The "Actions" dropdown menu is open, and the "Modify IAM role" option is highlighted. Below the table, the "Instance: i-03109ea1b9cfce510 (class)" details are visible, with tabs for Details, Security, Networking, Storage, Status checks, Monitoring, and Tags.

aws Learner Lab Instances | EC2 Management C x +

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

aws Services Search for services, [Option+S] N. Virgi voclabs/user2177624=Mark_Fischer @ 5617-0729

Elastic Container Service Elastic Container Registry CloudFormation VPC EC2 RDS

☑ Successfully stopped i-0c62b5f94dea05e8b

Instances (1/1) Info Refresh Connect Instance state Actions Launch instances

Find instance by attribute or tag (case-sensitive)

Instance state = running Clear filters

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state
<input checked="" type="checkbox"/>	class	i-03109ea1b9cfce510	Running

Instance: i-03109ea1b9cfce510 (class)

- Change security groups
- Get Windows password
- Modify IAM role

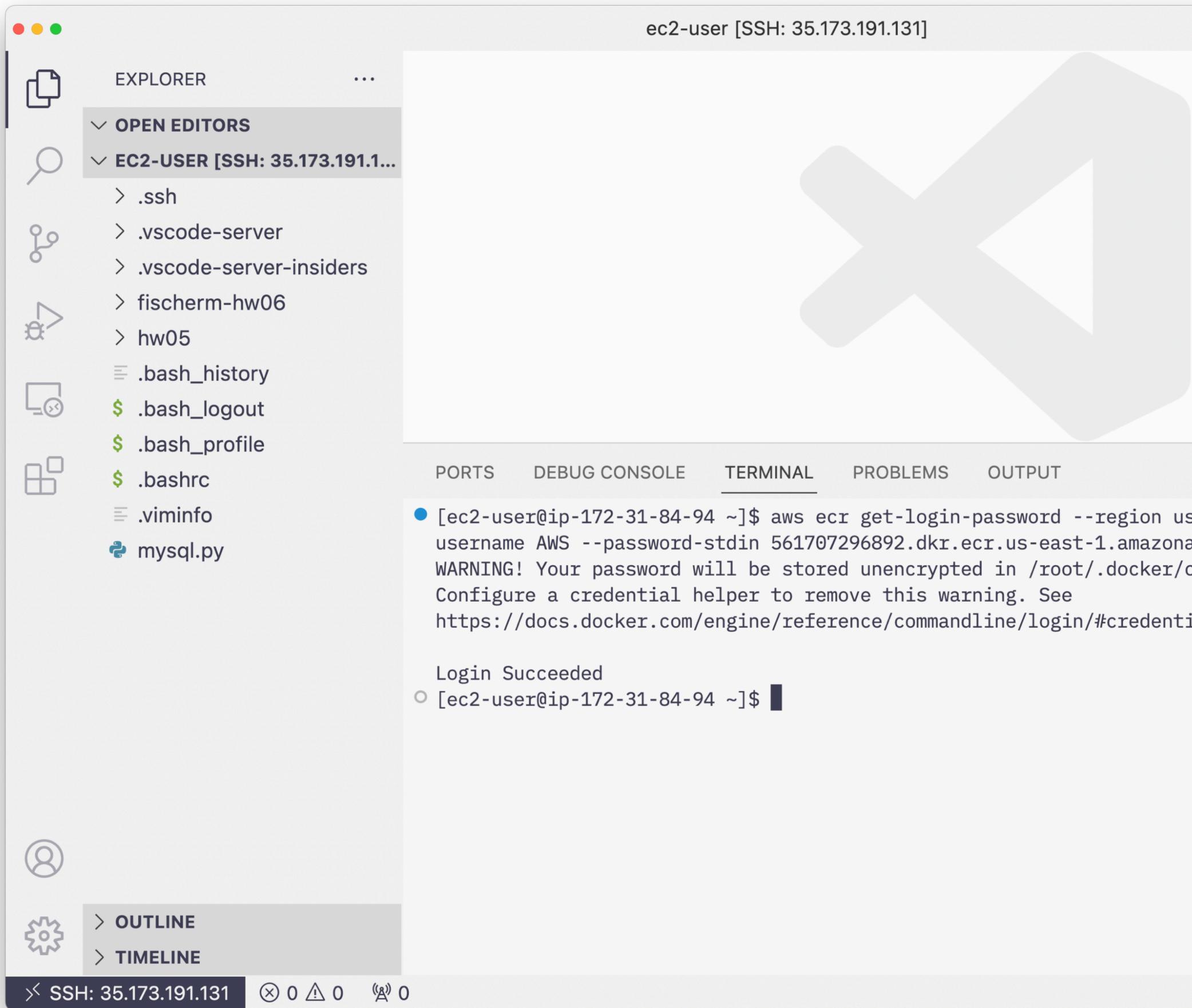
Details Security Networking Storage Status checks Monitoring Tags

EC
How

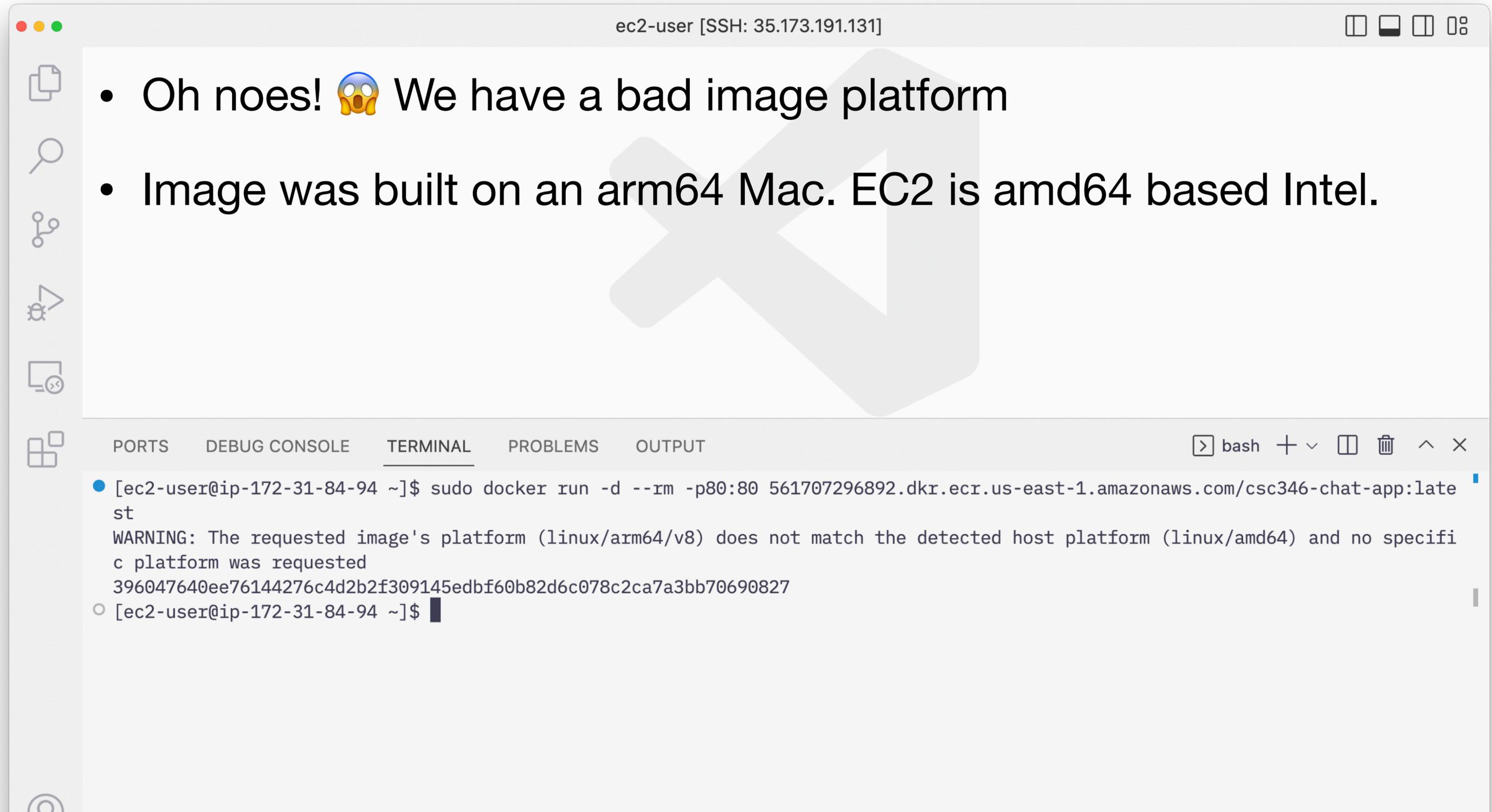
The screenshot shows the AWS Management Console interface. At the top, there are browser tabs for 'Learner Lab' and 'Modify IAM role | EC2 Manager'. The address bar shows the URL: `https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyIAMRole:instanceId...`. The navigation bar includes the AWS logo, a search bar with the text 'Search for services, [Option+S]', and user information for 'N. Virgi' and 'voclabs/user2177624=Mark_Fischer @ 5617-0729'. Below the navigation bar, a green banner displays a success message: 'Successfully stopped i-0c62b5f94dea05e8b'. The main content area shows the breadcrumb path: 'EC2 > Instances > i-03109ea1b9cfce510 > Modify IAM role'. The 'Modify IAM role' section has a sub-header 'Attach an IAM role to your instance.' and a form for selecting an IAM role. The 'Instance ID' field is populated with 'i-03109ea1b9cfce510'. A search box contains the text 'lab', and a dropdown menu shows the selected role 'LabInstanceProfile' with its ARN: `arn:aws:iam::561707296892:instance-profile/LabInstanceProfile`. Below the dropdown is a 'Choose IAM role' button. To the right of the dropdown is a 'Create new IAM role' link. A warning message states: '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?'. At the bottom right, there are 'Cancel' and 'Update IAM role' buttons.

ECS Repository

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



ECS Repository



The screenshot shows a terminal window titled "ec2-user [SSH: 35.173.191.131]". The terminal content is as follows:

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

The terminal output shows the command: `sudo docker run -d --rm -p80:80 561707296892.dkr.ecr.us-east-1.amazonaws.com/csc346-chat-app:latest`. The output is: `WARNING: The requested image's platform (linux/arm64/v8) does not match the detected host platform (linux/amd64) and no specific platform was requested` followed by a long alphanumeric string. The prompt returns to `[ec2-user@ip-172-31-84-94 ~]$`.

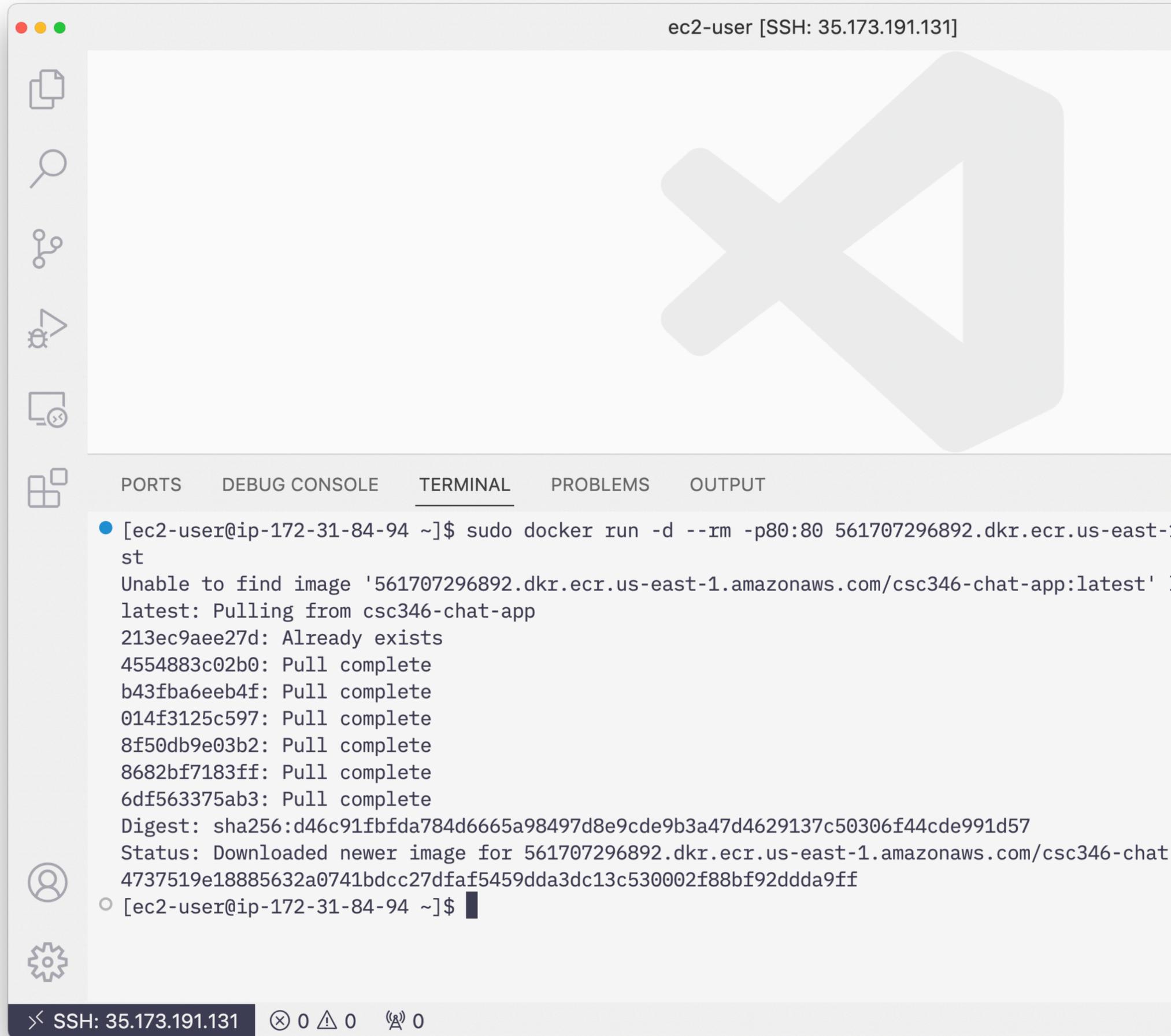
ECS Repository

- 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  
=> => sha256:4554883c02b087db69habcfd00dc2a380810ec59129fe5b7e0d30b0799085c38 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

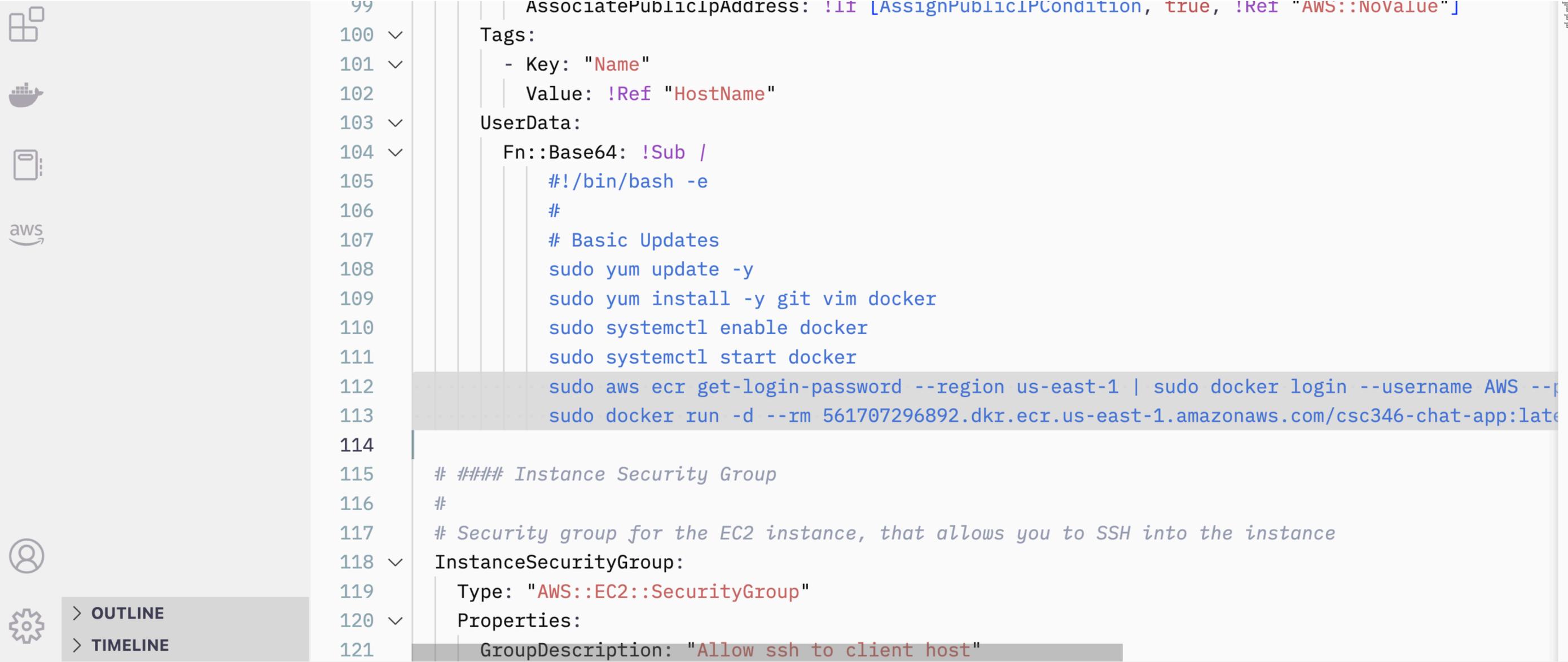


```
ec2-user [SSH: 35.173.191.131]

[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:latest
Unable to find image '561707296892.dkr.ecr.us-east-1.amazonaws.com/csc346-chat-app:latest' locally
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-app:latest
[ec2-user@ip-172-31-84-94 ~]$
```

More Automation

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



```
99     AssociatePublicIpAddress: !If [AssignPublicIPCondition, true, !Ref "AWS::NoValue"]
100   Tags:
101     - Key: "Name"
102       Value: !Ref "HostName"
103   UserData:
104     Fn::Base64: !Sub |
105       #!/bin/bash -e
106       #
107       # Basic Updates
108       sudo yum update -y
109       sudo yum install -y git vim docker
110       sudo systemctl enable docker
111       sudo systemctl start docker
112       sudo aws ecr get-login-password --region us-east-1 | sudo docker login --username AWS --password-stdin
113       sudo docker run -d --rm 561707296892.dkr.ecr.us-east-1.amazonaws.com/csc346-chat-app:latest
114
115     # ##### Instance Security Group
116     #
117     # Security group for the EC2 instance, that allows you to SSH into the instance
118   InstanceSecurityGroup:
119     Type: "AWS::EC2::SecurityGroup"
120   Properties:
121     GroupDescription: "Allow ssh to client host"
```

