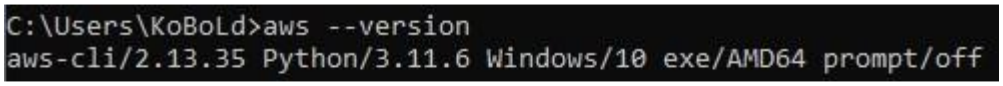
COSC 360 – Server Platform-as-a-Service
Lab 7 – Fall 2023

In this lab, we'll use the ***Amazon CLI*** (Command Line Interface) to log into AWS via our command prompt, and then use ***Docker*** to create and push a Docker image to the ***Amazon Elastic Container Registry*** (***ECR***).

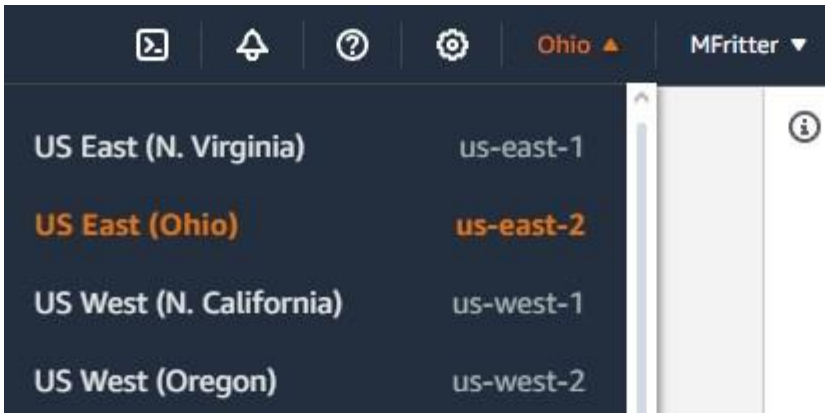
Initial CLI Setup

First, you'll need to download and install the ***Amazon CLI*** interface. You can get the Windows installer from the link below (versions for Mac and Linux are also available on the AWS website):
https://awscli.amazonaws.com/AWSCLIV2.msi

Run through the installation steps. Once complete, open a Windows ***command prompt*** (CMD) and verify that the CLI has been installed:

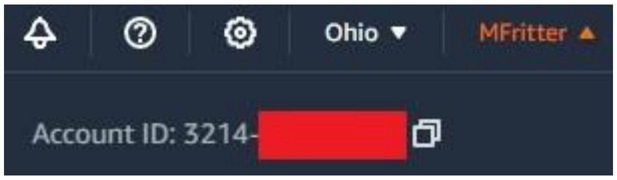


```
C:\Users\KoBoLd>aws --version
aws-cli/2.13.35 Python/3.11.6 Windows/10 exe/AMD64 prompt/off
```


Ohio
MFritter
US East (N. Virginia) us-east-1
US East (Ohio) us-east-2
US West (N. California) us-west-1
US West (Oregon) us-west-2

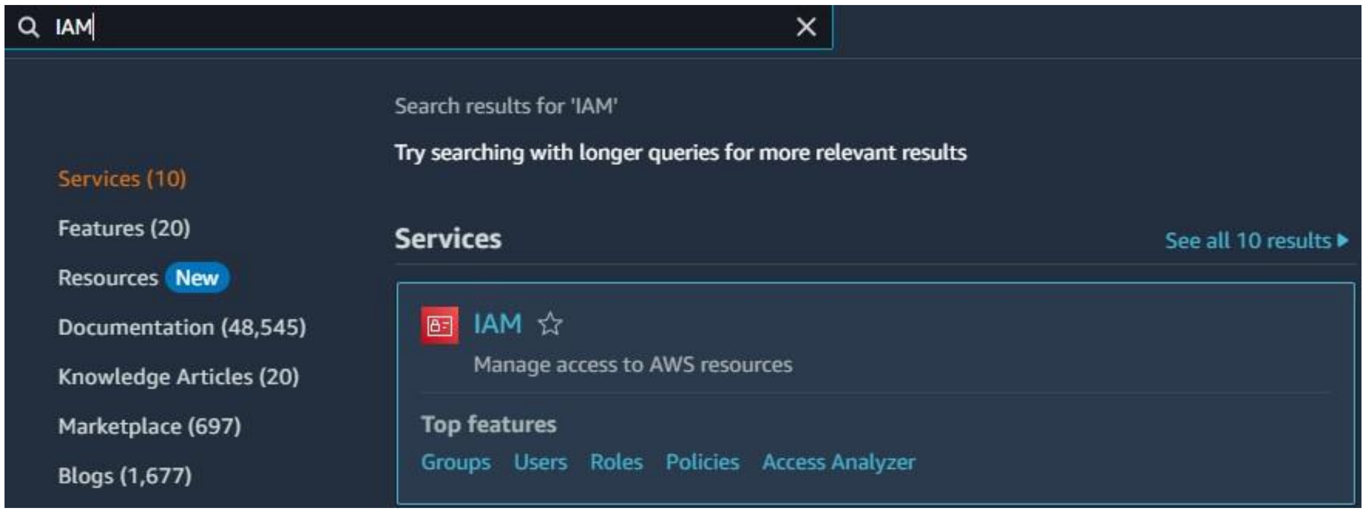
Next, we will need to do some configuration to ***authenticate*** ourselves and link our ***CLI*** to our actual Amazon AWS account. To do this, log into the AWS console via a web browser. First, check the ***region*** in the upper right-hand corner. In my case, my region is "***us-east-2***". Note your region down somewhere:

You will also need to find your ***12-digit Account ID***. This can be found by clicking your username in the top-right corner. It has a copy link, go ahead and copy this value and note it down as well, as well will need it repeatedly:



Account ID: 3214-[redacted]

Next, search for ***IAM*** and open up the ***Identity and Access Management Dashboard***:

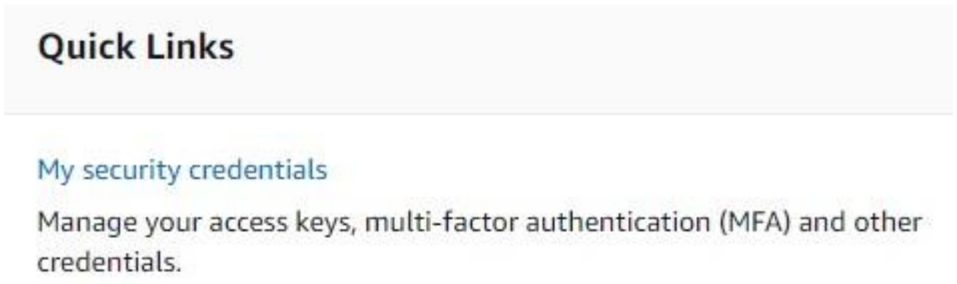


Search results for 'IAM'

Try searching with longer queries for more relevant results

Manage access to AWS resources

On the main dashboard under the ***Quick Links*** section, look for a link to your ***Security Credentials*** page:

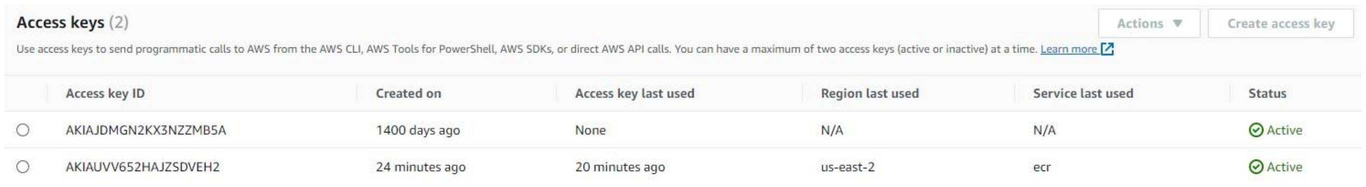


Quick Links

My security credentials

Manage your access keys, multi-factor authentication (MFA) and other credentials.

Click the link, and scroll down to the ***Access Keys*** table. You may have a maximum of ***two*** access keys. If you already have two, you may need to delete one and create a new access key, otherwise just create a new key with the ***Create Access Key*** button:

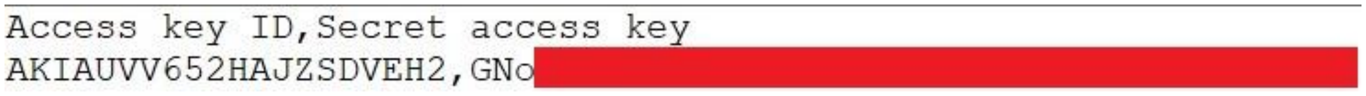


Access keys (2)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

	Access key ID	Created on	Access key last used	Region last used	Service last used	Status
○	AKIAJDMGN2KX3NZZMB5A	1400 days ago	None	N/A	N/A	✓ Active
○	AKIAUVV652HAJZSDVEH2	24 minutes ago	20 minutes ago	us-east-2	ecr	✓ Active

Creating a key will give you ***two important values – a key ID***, and the ***secret access key itself***. Download the CSV file it provides at the end of the key creation process:



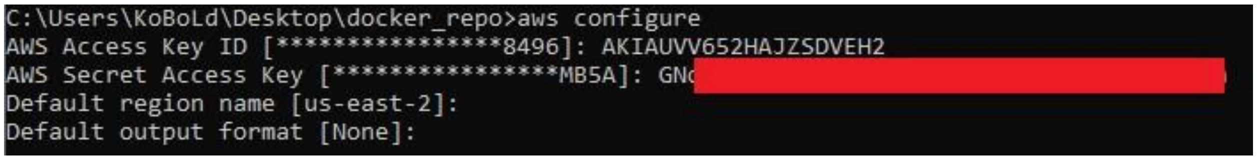
```
Access key ID,Secret access key
AKIAUVV652HAJZSDVEH2,GNo[redacted]
```

You ***cannot*** retrieve the secret access key after this step, so it's important you keep this file somewhere safe, or you will have to delete the key and generate a new one.

Finally, go back to the command prompt, and enter the following command:

aws configure

This will prompt you to enter your ***access key ID***, followed by the ***secret access key*** itself, and a region (this should match the region you found for your account previously). Leave the output format as ***none*** (the default):



```
C:\Users\KoBoLd\Desktop\docker_repo>aws configure
AWS Access Key ID [****************8496]: AKIAUVV652HAJZSDVEH2
AWS Secret Access Key [****************MB5A]: GN[redacted]
Default region name [us-east-2]:
Default output format [None]:
```

Congratulations, you've linked your ***AWS CLI*** to your AWS account, and can now use the CLI to perform AWS tasks.

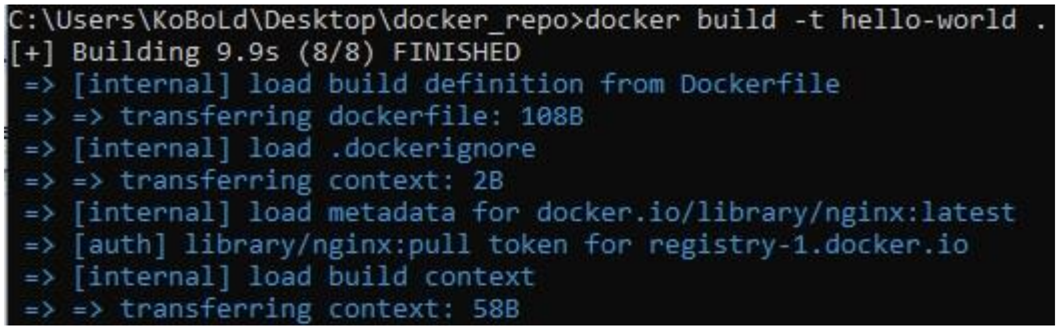
Using the Amazon Elastic Container Registry

Now we'll try using the ***ECR*** to store a Docker image. First, create a ***new folder*** somewhere easy to find, and create two files in it: a ***Dockerfile***, and an ***index.html*** file. Your Dockerfile will be very simple, just an ***nginx*** web server with a custom index page:

```
FROM nginx:latest
COPY ./index.html /usr/share/nginx/html/index.html
```

Inside the ***index.html*** file, go ahead and write a simple Hello World message in it with some header tags (i.e. <h1>Hello World</h1>).

Navigate to the folder in the command prompt, and use ***Docker*** to perform a local build of the Docker image:



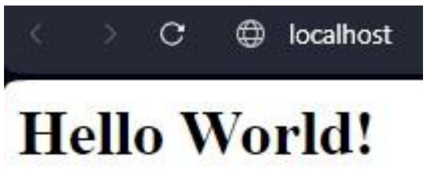
```
C:\Users\KoBoLd\Desktop\docker_repo>docker build -t hello-world .
[+] Building 9.9s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 108B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/nginx:latest
 => [auth] library/nginx:pull token for registry-1.docker.io
 => [internal] load build context
 => => transferring context: 58B
```

It should go through the normal Docker build process. If you get a Docker daemon error, remember to make sure that Docker is running before trying to run the build command.

Try running your newly-built image locally using the ***Docker*** run command:

docker run -t -i -p 80:80 hello-world

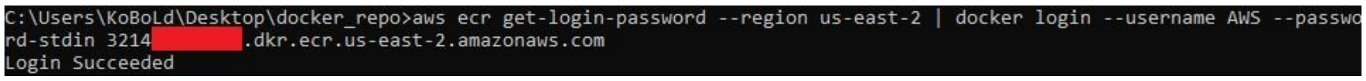
Doing so should allow you to visit the ***localhost*** in your browser to see your default ***index.html*** page:



Go ahead and shut down the container once you've verified it is working. Now we'll need to go ahead and ***log in via Docker*** using ***AWS***. This is what allows ***Docker*** to push to our ***repositories*** in the ECR. Run the following command after modifying it:

aws ecr get-login-password --region REGION | docker login --username AWS --password-stdin ID_NUMBER.dkr.ecr.REGION.amazonaws.com

Where ***REGION*** is the ***region*** that was listed for your account (i.e. "us-east-2") and ***ID_NUMBER*** is the ***twelve digit*** ID number for your account you copied earlier:



```
C:\Users\KoBoLd\Desktop\docker_repo>aws ecr get-login-password --region us-east-2 | docker login --username AWS --passwo
rd-stdin 3214[redacted].dkr.ecr.us-east-2.amazonaws.com
Login Succeeded
```

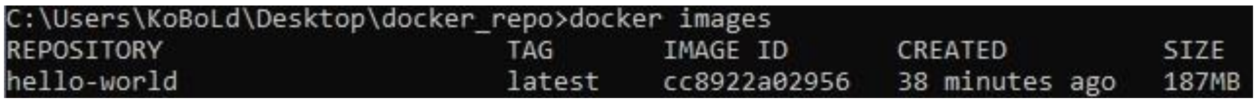
This should return a ***Login Succeeded*** message to let you know you've successfully logged in with your credentials that you provided via the aws configure command you ran earlier.

At this point, we can go ahead and ***create our repository***:

aws ecr create-repository --repository-name hello-world --image-scanning-configuration scanOnPush=true --region REGION

Once again replacing ***REGION*** with the region your account is in. Running this command will generate an empty repository in the ***ECR***.

You can list the images available using ***docker images***. This shows the available ***locally built images*** from Docker. In this case I called my image ***hello-world***, and it appears in the list as such:



```
C:\Users\KoBoLd\Desktop\docker_repo>docker images
REPOSITORY                         TAG       IMAGE ID       CREATED          SIZE
hello-world                        latest    cc8922a02956   38 minutes ago   187MB
```

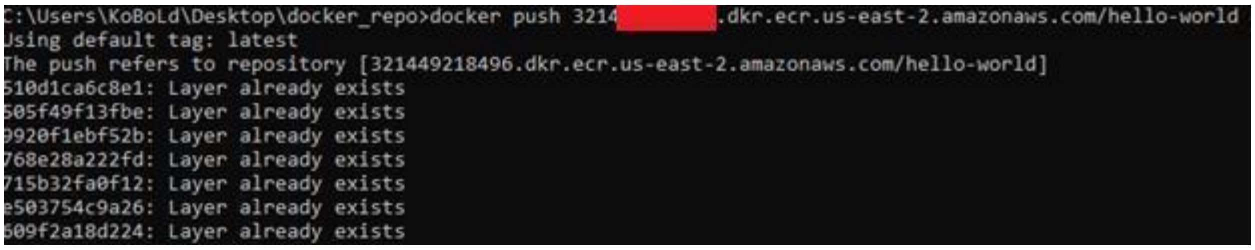
The last step will be to ***tag*** and then ***push*** this ***local*** Docker image to the ECR. First, we need to apply a tag to the image to mark that we would like to push it to the ECR repository:

docker tag hello-world:latest ID_NUMBER.dkr.ecr.REGION.amazonaws.com/hello-world

This links the local hello-world image to our remote repository. As previously, you'll need to include your ***twelve digit ID number*** and the account ***region*** in the command. Finally, we can ***push*** our local image to the ECR, which will actually perform the upload and make our image available from Amazon's ECR servers:

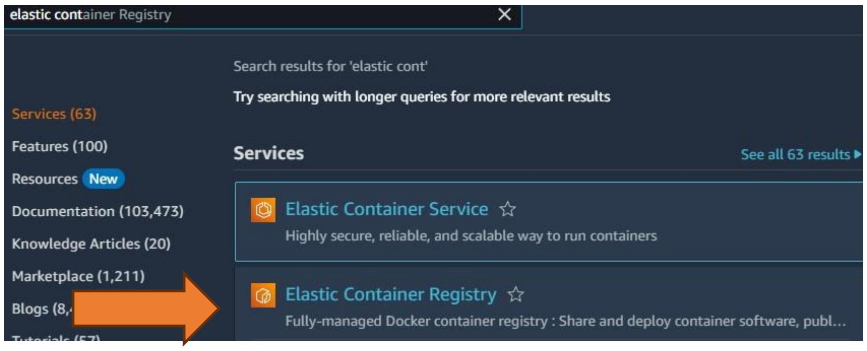
docker push ID_NUMBER.dkr.ecr.REGION.amazonaws.com/hello-world

Once run, you should see the push output for each layer of the image go through, just like when we pushed to ***Docker Hub*** previously:

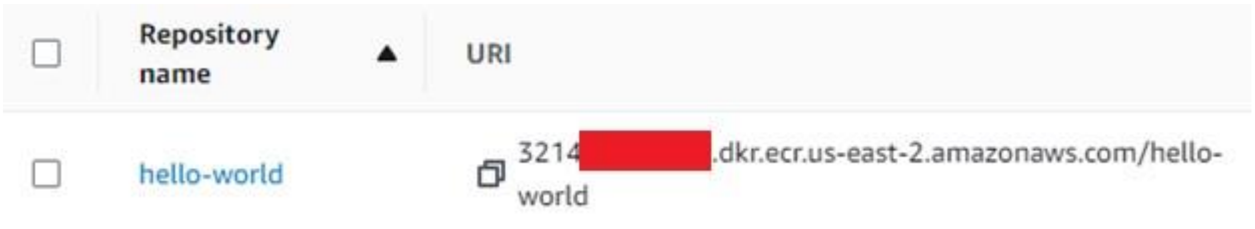


```
C:\Users\KoBoLd\Desktop\docker_repo>docker push 3214[redacted].dkr.ecr.us-east-2.amazonaws.com/hello-world
Using default tag: latest
The push refers to repository [321449218496.dkr.ecr.us-east-2.amazonaws.com/hello-world]
510d1ca6c8e1: Layer already exists
505f49f13fbe: Layer already exists
9920f1ebf52b: Layer already exists
768e28a222fd: Layer already exists
715b32fa0f12: Layer already exists
e503754c9a26: Layer already exists
509f2a18d224: Layer already exists
```

Now we go back to the Amazon web console in our browser, search ***Elastic Container Registry***, and enter the ***ECR dashboard***:

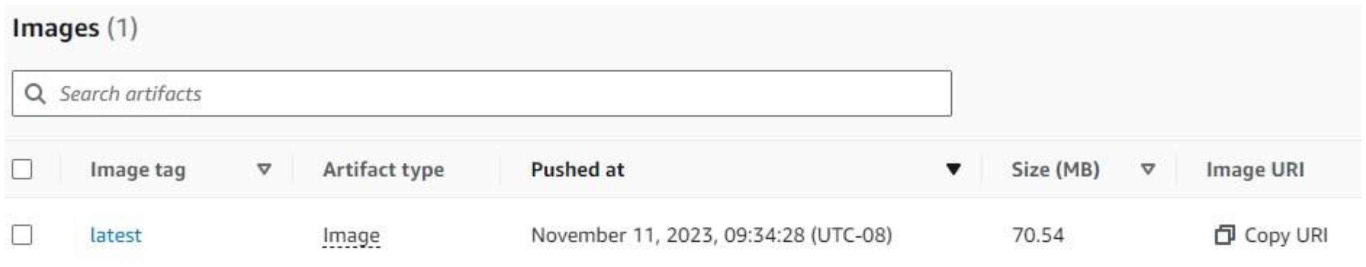

elastic container Registry
Search results for 'elastic cont'
Try searching with longer queries for more relevant results
Services (63)
Features (100)
Resources New
Documentation (103,473)
Knowledge Articles (20)
Marketplace (1,211)
Blogs (8,
Services
See all 63 results
Elastic Container Service
Highly secure, reliable, and scalable way to run containers
Elastic Container Registry
Fully-managed Docker container registry : Share and deploy container software, publ...

We should now see our newly created and pushed repository listed here, similar to the image below. ***Take a screenshot of this page***:



	Repository name ▲	URI
☐	hello-world	3214[redacted].dkr.ecr.us-east-2.amazonaws.com/hello-world

Click on the repository name, and ***also take a screenshot of this page***, showing the last time the repository was pushed to and any vulnerabilities the image may have:



Images (1)

Search artifacts

	Image tag	Artifact type	Pushed at	Size (MB)	Image URI
☐	latest	Image	November 11, 2023, 09:34:28 (UTC-08)	70.54	Copy URI

Lab Submission

When you are finished, upload ***the two screenshots*** you took of the ***ECR repositories*** during the lab on ***Moodle***. Please upload these as ***plain images***, and not as a ***pdf*** or ***Word*** document. They should clearly show that you have successfully created ***and*** pushed to the ECR repository.