**# Hands-on ECR-01 : Pushing Docker Images to ECR Manually**

Purpose of the this hands-on training is to give basic understanding of how to use AWS Elastic Container Registry (ECR) and how to manage docker images using ECR.

**## Learning Outcomes**

At the end of the this hands-on training, students will be able to;

- prepare a Docker Machine with terraform.

- create and configure AWS ECR from the AWS Management Console.

- create and configure AWS ECR from the AWS CLI.

- demonstrate how to build a docker image with Dockerfile.

- use Docker commands effectively to tag, push, and pull images to/from ECR.

- delete images and repositories on ECR from the AWS Management Console.

- delete images and repositories on ECR from the AWS CLI.

**## Outline**

- Part 1 - Launching a Docker Machine Instance Configured for ECR Management

- Part 2 - Creating Repositories on AWS ECR

- Part 3 - Work with AWS ECR

- Part 4 - Managing Docker Images using AWS ECR as Repository

**## Part 1 - Launching a Docker Machine Instance Configured for ECR Management**

- Launch a Compose enabled Docker machine on Amazon Linux 2 AMI with security group allowing HTTP and SSH connections using the terraform file.

- Explain the resources in the terraform file.

**## Part 2 - Creating Repositories on AWS ECR**

- Go to the `Amazon Elastic Container Registry` service and explain what it is.

- Introduce menus on the left side, `Amazon ECS`, `Amazon EKS`, `Amazon ECR`.

- Click on `Repositories` on ECR section, and explain the UI.

- Click on `Create Repository` and explain the default `registry` for user account. (`aws\_account\_id`.dkr.ecr.`region`.amazonaws.com)

- Explain repository name convention. (*\*Repository names can support namespaces, which you can use to group similar repositories. For example if there are several teams using the same registry, `Team A` could use the `team-a` namespace while `Team B` uses the `team-b` namespace. Each team could have their own image called `web-app`, but because they are each prefaced with the team namespace, the two images can be used simultaneously without interference. Team A's image would be called `team-a/web-app`, while Team B's image would be called `team-b/web-app`.\**)

- Enter a repository name ex. `student1-repo/todo-app` (**\*\****\*In this hands-on, we will be working with a simple `todo list manager` that is running in `Node.js`. If you're not familiar with Node.js, don't worry! No real JavaScript experience is needed!\****\*\***)

- Explain `tag immutability` and leave it as default. ##degisitirilemez etiket ayni etiket kullnilmaz

- Explain `scan on push` and leave it as default.#paylasilan image hassas veri olmadurumu kontrol eder enable edersen uzun surer

- Create the repository and explain the complete URI.(*\*Combination of registry and repo name\**)

```text

xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/student1-repo/todo-app

```

**### Creating a lifecycle policy:**

- A lifecycle policy allows you to create a set of rules that expire unused repository images. The following procedure shows you how to create a lifecycle policy. You should expect that after creating a lifecycle policy, the affected images are expired within 24 hours.

- Create a lifecycle policy (AWS Management Console):

- From the navigation bar, choose the Region that contains the repository for which to create a lifecycle policy.

- In the navigation pane, choose Repositories.

- On the Repositories page, on the Private tab, select a repository to view the repository image list.

- On the repository image list view, in the left navigation pane, choose Lifecycle Policy.

\* Note:

If you don't see the Lifecycle Policy option in the navigation pane, ensure that you are in the repository image list view.

- On the repository lifecycle policy page, choose Create rule.

- Enter the following details for your lifecycle policy rule.

- For Rule priority, type a number for the rule priority.

- For Rule description, type a description for the lifecycle policy rule.

- For Image status, choose Tagged, Untagged, or Any.

\* If you specified Tagged for Image status, then for Tag prefixes, you can optionally specify a list of image tags on which to take action with your lifecycle policy. If you specified Untagged, this field must be empty.

- For Match criteria, choose values for Since image pushed or Image count more than (if applicable).

- Choose Save.

**## Part 3 - Configuring AWS CLI to Work with AWS ECR**

- connect to your instance with SSH.

```bash

ssh -i .ssh/xxxxx.pem ec2-user@ec2-3-133-106-98.us-east-2.compute.amazonaws.com

```

- Check your AWS CLI version `aws --version` command.(*\*AWS CLI is already installed on Amazon Linux 2 AMIs, but Amazon ECR functionality is available in the AWS CLI starting with version 1.9.15\**).

- Authenticate the Docker CLI to your default `registry`

```bash

sudo su

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

```

**## Part 4 - Managing Docker Images using AWS ECR as Repository**

- Go your instance from terminal.    ##docker version bu komutla docker kurulumu kontrol edildi

- Download the sample project `to-do-app-nodejs.tar` file from the GitHub Repo on your instance. In this hands-on, we will be working with a simple `todo list manager` that is running in `Node.js`.

```bash

wget https://github.com/awsdevopsteam/jenkins-first-project/raw/master/to-do-app-nodejs.tar   ##awsdevoppsteam reposunda bir image var onu ##indiiriyoruz bu sekilde bir tar dosyasi indirlidi hemen sol  tarafa geldi nodejs seklinde yazilan kod uygulasi var onu uzerinden image olusturucagiz

```

- Extract the `to-do-app-nodejs.tar` file

```bash

tar -xvf to-do-app-nodejs.tar   ##indiridigimiz uygulamayi tar dosyasindan cikariyouz bu dosya icinde bu uygulamanin kodlari var

ls                         ##bu dosya icinde ne oldugunu listeler .

```

- You will see "to-do-app-nodejs.tar" and "to-do-app-nodejs".

- Enter the "/to-do-app-nodejs/" directory. And create "Dockerfile" via vi editor.

```bash

cd to-do-app-nodejs/   ##indiridigimiz uygulama dosyasinin icine girdik burda bir docjker file olusuturcaz

vi Dockerfile       ##vi komutu ile yeni bir dosya olusuturucaz dockerfle isimli  nodejs icinde dockerfile normalde yoktu.dockerfile ##uzeriden bir image olusuturucagiz (code Dockerfile terminalede yazarak ta dosya olusuturulur ama kaydet )

press "i" to edit.

```

- Paste a following content into Dockerfile

```dockerfile

FROM node:12-alpine          ##alpine image olusuturuyor

WORKDIR /app            ##WORKDIR bir klasor olusuturyor orda calismasi icin  /app altinda bu klasor olusacak

COPY . .                     ##burda olusacak herseyi wdorkdir e koplayacak

RUN yarn install --production        ##yarn paketini kullanacak uygulama icin gereken altyapiyi indirecek ve kuracak.yarn bir oaket ###managerdir nodejs icin gereken ortam eklentiler kutuphanlere bu komutla indirilip kurulacaktir image daha sonra kullanarak konteyner olusuturucaz,dockerfile i kullnarak docker image kuruypruz

CMD ["node", "/app/src/index.js"]         ##calistiracagi belirlenir

```

- Press "ESC" and ":wq " to save.

- Build the Docker container image using the `docker build` command.

```bash

docker build -t todo-app:latest .    ##imagei docker ile olusuturuken repo adi ile ypiliyor  burda todoapp diye kendimiz bir isim verdik

```

- Show the Docker image is created successfully.

```bash

docker image ls   #3docker image olusutumu diye lsite ile kontrol edilir

```

- Run `todo app` from the local image.   ##todo app diye bir image olusuturmus simdi bunu run ile calisitiyoruz

```bash

docker run --name todo -d -p 80:3000 todo-app   ##todoapp imageini run ile calisitiriyoruz --nsme todo conatiner adi yazildi -d detach modunda calisiitirilir -p port 80 uzerinden yayin yapmasini isitiyoruz 3000 poru ise image in dir.Makinenin public ip si kullanarak todoapp uygumalasi yayin yapmisimi kontrol edilir aws de public ip alip yeni sekmede yapisitirilir todo app geldi todo uygulamasi calisiti

```

- List running container.

```bash

docker ps   #calisan contairleri gosteriyor

# or

docker container ls  ##calisan cantainerlei listeler

```

- Check if the To-Do App is running by entering `http://<ec2-host-name>` in a browser.

- Stop and remove container

```bash

docker stop todo   ##todo adindaki container durdurulur docker stop container durdurmak icin kullnailir

docker rm todo         ##rm remove ile docker rm silmek icin kullanilir todo contanier siler  burda sildik simdi aws ecr de yaptigimiz contaner registry servisinde olusuturudugumuz repo uzerinde image olusuturup kendi icine atma yontemini kullnicaz olsuturudgumuz repo icine girdik view push commands icine girdik burdan komutlari sirasiyla calisitirarak yeni image olusuturup kendi icine almak icin terminalde bu komutlaru sirasiyla calisitiriyoruz. view icindeki 2. komutta :docker build -t student1-repo/todo-app ." yani bunda bu isimli br yeni image olusuturyor sonra da bunu "docker tag student1-repo/todo-app:latest 393498386407.dkr.ecr.us-east-1.amazonaws.com/student1-repo/todo-app:latest"     bu komutla yeni bir tag yani isimle i=o imagei degisitiriyor. bununla birlik te "docker images " komy=utu ile 3 tane image oldugu artk gorulur .tag deyince yeni image olusturdu bir onceki student1-repo/todo-app i kullnarak yeni image olusuturdu aslinda bu olusan 3 image da ayni image id ye bakarsak ayni id leri var .

```

- Tag the image to push it to ECR repository.

```bash

docker tag todo-app:latest xxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/student1-repo/todo-app:latest  ##burdaki komutlari yukaridaki aciklamada olgdugu gibi aws ecr icinde ki repo icinde  view kismindan aldik

```

- Push the image to your ECR repository and show the pushed image on AWS ECR Console.

```bash

docker push xxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/student1-repo/todo-app:latest   #aws ecr repo view kisiminden yeni yukardaki olsturdugumuz imageleri bu repo icine gondermek icin pushlama yapilir buradki komutu aws ecr repo view kimindan alip terminalde yapisitirp pushlamis olacagiz ama pushlama yapabilmesi icin "no basic credentials" diyor credential girmedigimi icin hata aldik view kismindaki 1,komutu burda calisitiridk "aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 393498386407.dkr.ecr.us-east-1.amazonaws.com" bu komutla otomatik olarak sifreyi alir docker login yaparak username ve sifreyi giriyor "dha onceden docker hub a girerkende docker login sifresini kullniyorduk burdada ayni sekilde docker login var aws passwordu biz bilmiu-ruz aws otoamatik aliyor aws ec2 ya bir role atama yapmisitik terraformu kullanarak ec2 makine ayaga kaldirirken yetki vermisitk ec2 secrty kiminda modify iam role de bir yetkini baglandigini bu maiineye gorduk "ecr-ec2-profile-teckproed"  sonra viec kisnidaki ilk komut calistirilir  ve yetki verildi ve simdi push yapilir 4.sekme "docker push 393498386407.dkr.ecr.us-east-1.amazonaws.com/student1-repo/todo-app:latest" bu komutla repoya gider image .repodan image makineye cekmek icin  docker pull komutu ile "docker pull 393498386407.dkr.ecr.us-east-1.amazonaws.com/student1-repo/todo-app:latest" bu komut ile onceki olusuturup push ile attigimiz image i maineye geri cekmek icin kullanilir.butun imageleri silmek iicn "docker rmi $(docker images -qa) -f " komutu kullanilir

```

- Create an container from image located ECR

```bash

docker pull xxxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/student1-repo/todo-app:latest

docker run --name todo -dp 80:3000 xxxxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/student1-repo/todo-app:latest

```

- Stop and remove container

```bash

docker stop todo

docker rm todo

```

- make change into the script.`<p className="text-center">No items yet! Add one above!</p>` with following new text.

```bash

vi src/static/js/app.js

```

- Create an Docker image

```bash

docker build -t todo-app:latest .  ##tekrar image olusuturuluyor bundan once butun imageleri silmisik makineden onceki repo icinde ki image ile yeni image o nedenle repodan pull ile ordaki image i makineye cekiyoruz

```

- Tag the image to push it to ECR repository.

```bash

docker tag todo-app:latest xxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/student1-repo/todo-app:latest

```

- Push the image to your ECR repository and show the pushed image on AWS ECR Console.

```bash

docker push xxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/student1-repo/todo-app:latest

```

- Show the repository . Immutable disable.

- Create an container from image located ECR

```bash

docker run --name todo-v1 -dp 80:3000 xxxxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/student1-repo/todo-app:latest

```

- Show the changes in the browser

- Stop and remove container

```bash

docker stop todo-v1

docker rm todo-v1

```

- Create a ECR repository named `student2-repo/todo-app` from AWS CLI and show it is created on AWS ECR Console.

```bash

aws ecr create-repository \

    --repository-name student2-repo/todo-app \

    --image-scanning-configuration scanOnPush=false \

    --image-tag-mutability IMMUTABLE \

    --region us-east-1

```

- Tag and push the image to your `student2-repo/todo-app` repository and show the pushed image on AWS ECR Console.

```bash

docker tag todo-app:latest xxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/student2-repo/todo-app:latest

docker push xxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/student2-repo/todo-app:latest

```

- Delete the all local images of `todo-app`.

```bash

docker image ls

docker image rm --force <image\_id>

```

- Show that there is no image locally

```bash

docker image ls

```

- Pull the image from your `student2-repo/todo-app` repository  to the local.

```bash

docker pull xxxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/student2-repo/todo-app:latest

docker run -dp 80:3000 xxxxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/student2-repo/todo-app:latest

```

- Or directly run the `todo-app` from the ECR repo.

```bash

docker run -dp 80:3000 xxxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/student2-repo/todo-app:latest

```

- Check if the To-Do App is running by entering `http://<ec2-host-name>` in a browser.

- Delete Docker image on your `student2-repo/todo-app` repository from AWS CLI. Since we didn't configure aws CLI you may get error. If you can't do the process with root, then exit from root.

```bash

aws ecr batch-delete-image \

      --repository-name student2-repo/todo-app \

      --image-ids imageTag=latest

```

- Delete the ECR repository  `student2-repo/todo-app` from AWS CLI.

```bash

aws ecr delete-repository \

      --repository-name student2-repo/todo-app \

      --force

```

- Delete the image and repository of `student1-repo/todo-app` from AWS ECR Console.