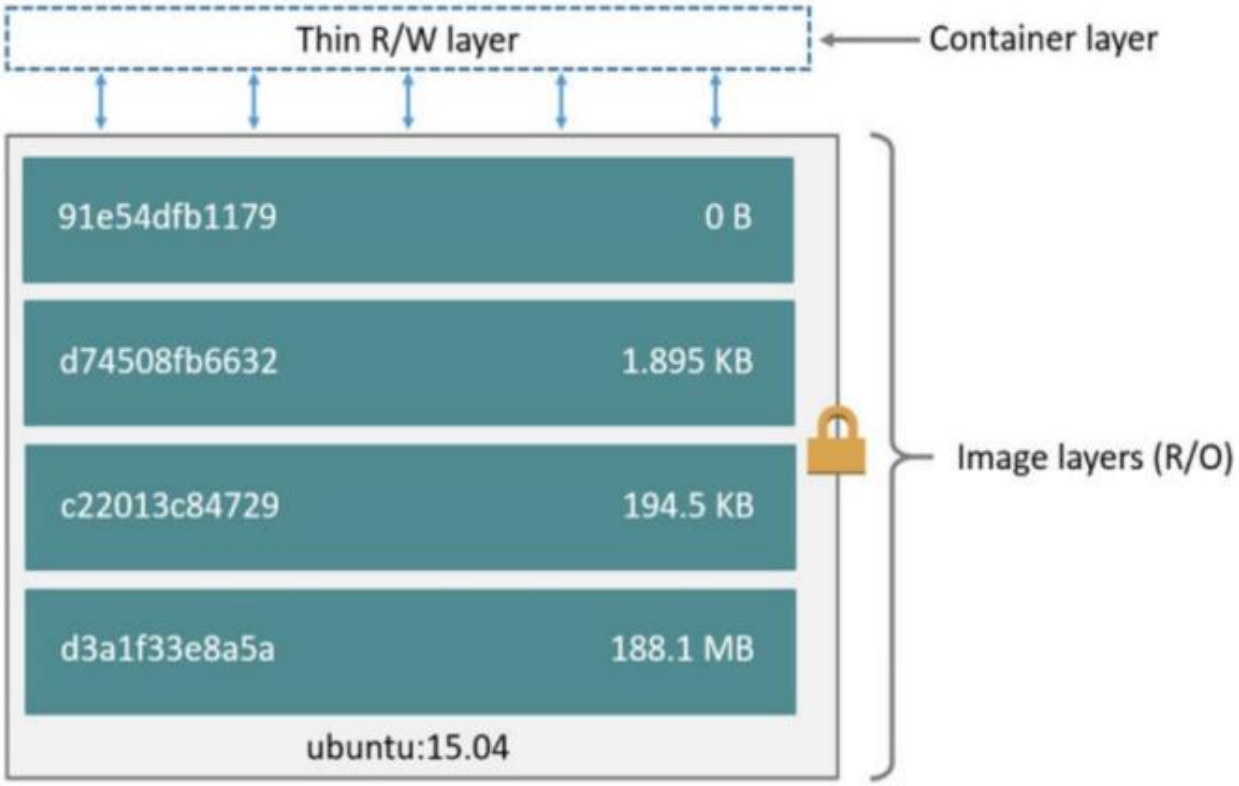
Kubernetes (The Defector Production Environment) Security Guide

Static Scanning of the Docker Base Images to Detect Vulnerabilities Before Deploying the Image Application to Production Environment

In a secure pipeline, Docker vulnerability scanning should be a mandatory step of your CI/ CD process, and any image should be scanned and approved before entering “Running” state in the production clusters. (Resume Key Point)

Docker images are composed of several immutable layers, basically a diff over a base image that adds files and other changes. Each one is associated with a unique hash id:



Any new Docker image that you create will most likely be based on an existing image (FROM statement in the Dockerfile).

That is why you can leverage this layered design to avoid having to re-scan the entire image every time you make a new one, with a small change.

If a parent image is vulnerable, any other images built on top of it will be vulnerable too.

Example of a Dockerfile

```
ARG JENKINS_VER=2.277.1

FROM jenkins/jenkins:${JENKINS_VER}

ARG JENKINS_VER
ARG RELEASE=1

USER root

COPY files/jenkins_wrapper.sh /usr/local/bin/jenkins_wrapper.sh
COPY files/jenkins.yaml  /usr/local/bin/jenkins.yaml

ENV CASC_JENKINS_CONFIG=/usr/local/bin/jenkins.yaml

# create version files to ensure Jenkins does not prompt for setup
# allow slave to master control - https://wiki.jenkins.io/display/JENKINS/Slave+To+Mast
# create file for plugin versioning
RUN echo -n ${JENKINS_VER} > /usr/share/jenkins/ref/jenkins.install.UpgradeWizard.state
    echo -n ${JENKINS_VER} > /usr/share/jenkins/ref/jenkins.install.InstallUtil.lastExe
    mkdir -p /usr/share/jenkins/ref/secrets/ && echo false > /usr/share/jenkins/ref/sec
    echo ${JENKINS_VER}-${RELEASE} > /usr/share/jenkins/ref/jenkins-docker-image-versio
```

Base Image

We must scan the base image which is jenkins/jenkins

If the base image is vulnerable, then the rest of the immutable layers of the image are also vulnerable

The entire file will be present in this repository: https://github.com/target/jenkins-docker-master.git

Vulnerability scanning for Docker local images

There are two ways to achieve this.

1. Through Docker hub repository account
2. Through Amazon ECR

Let start with the https://hub.docker.com/ option

Vulnerability scanning for Docker local images

Estimated reading time: 14 minutes

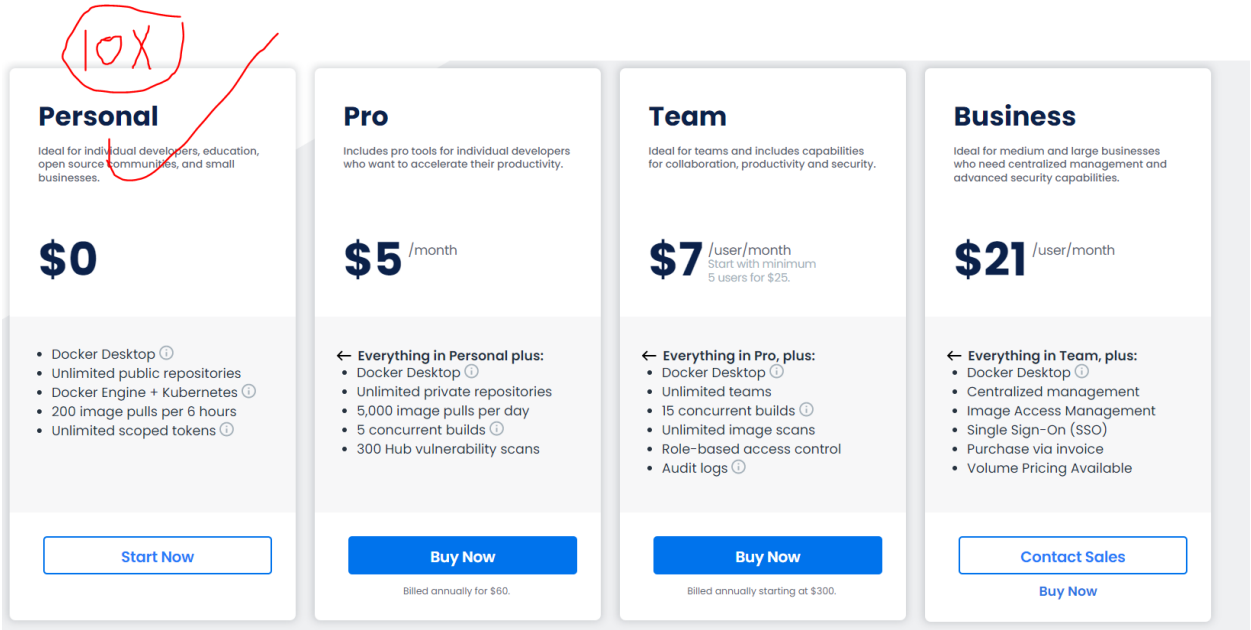
Scan your images for free

Did you know that you can now get 10 free scans per month? Sign in to Docker to start scanning your images for vulnerabilities.

Sign in

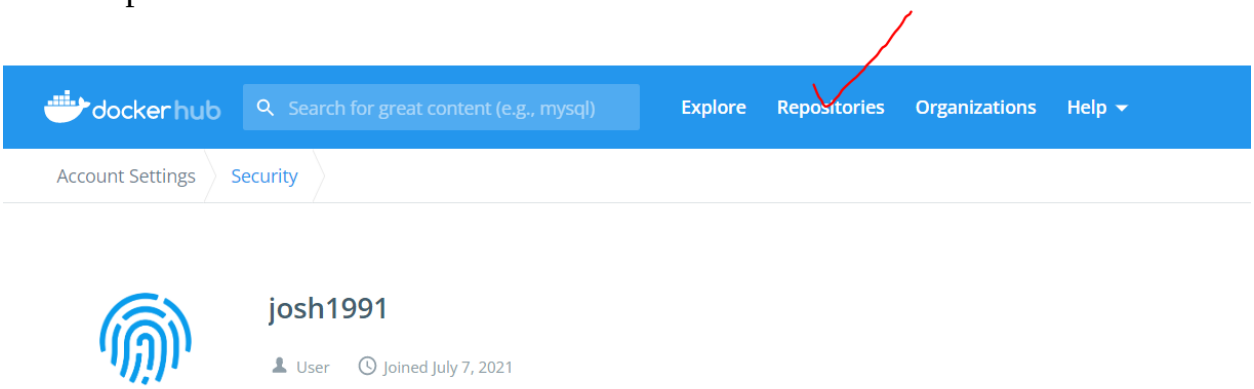
You are entitled to 10 free scans per month (Personal Account). After that you must upgrade.

In the company, this limitation will not exist because the company will have the business account with Docker.

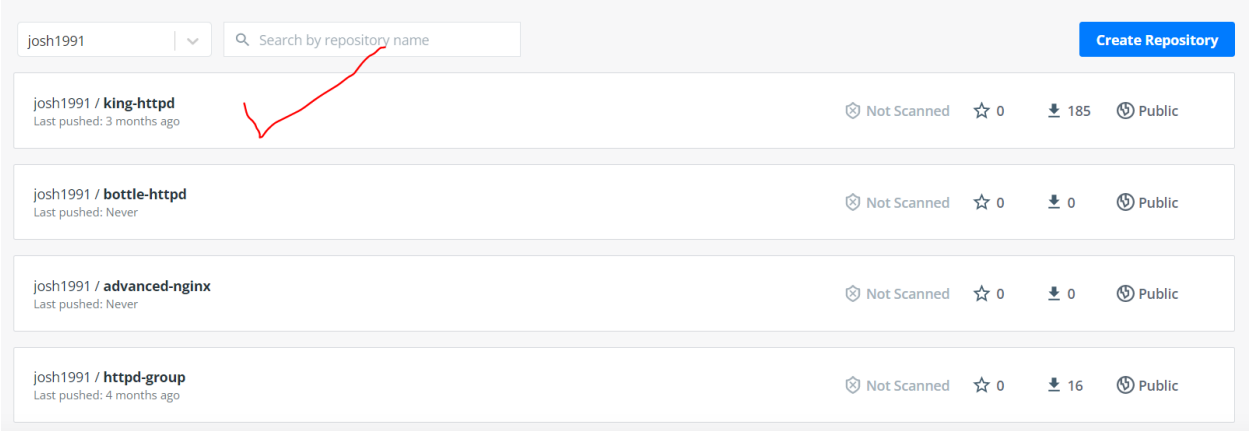


Login to your docker hub account

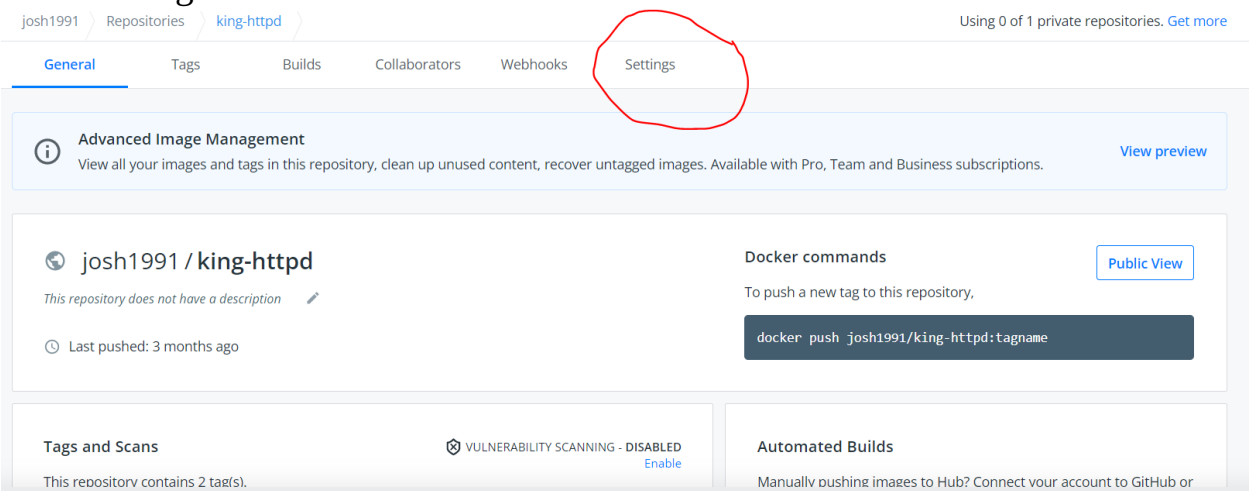
- ⇨ https://hub.docker.com/repositories
- ⇨ Go to repositories



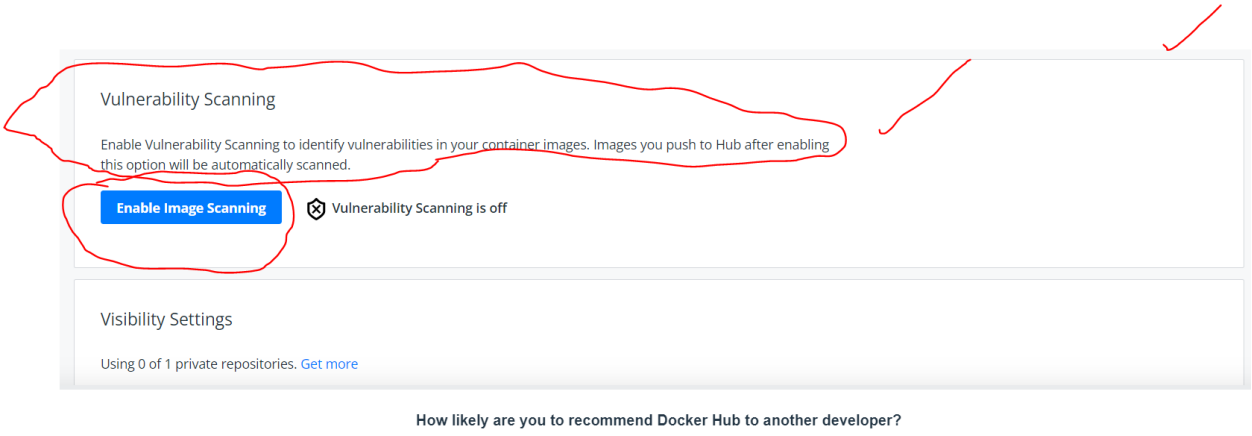
- ⇨ Lick on any of the images you have



⇨ Go to settings

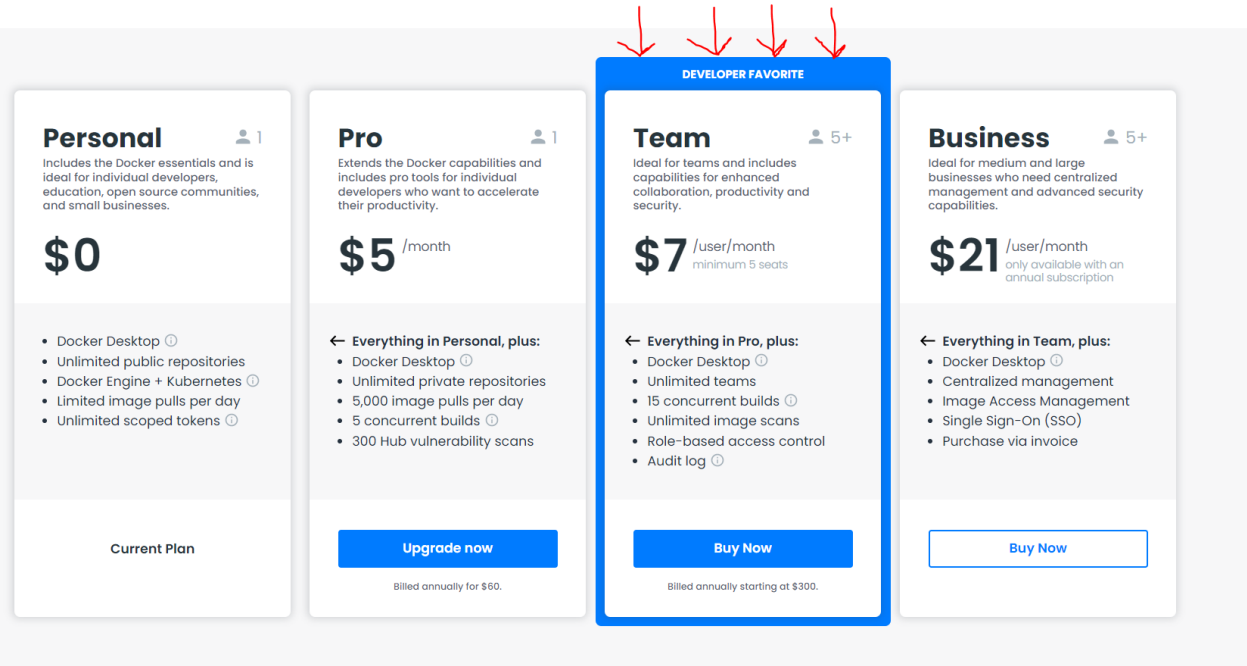


➔



Enable scanning and all the images pushed to this account will automatically get scanned for vulnerability issues.

You must upgrade to Pro, Team or Business account to use this feature.



Option 2

Static scanning - You can the layers of the image before it is deployed to the production like and live production (prior to Deployment).

Dynamic scanning - You can scan containers running in Docker engine (Run Time)

- ⇨ Amazon ECR Option

Steps

- ⇨ Using terraform, automate the provisioning of a t2-micro instance on AWS console.
- ⇨ Create a terraform script ending with .sh to install the latest version of docker, and git.
- ⇨ In the script, add the following information.
- ⇨ # Docker CE Install
  sudo amazon-linux-extras install docker

```
[ec2-user@ip-10-0-25-228 ~]$ docker -v
Docker version 20.10.13, build a224086
[ec2-user@ip-10-0-25-228 ~]$
```

sudo service docker start
sudo usermod -a -G docker ec2-user

- # Make docker auto-start

  sudo chkconfig docker on

- # Because you always need it....

  sudo yum install -y git

- # Reboot to verify it all loads fine on its own.

  sudo reboot

- docker-compose install
- Copy the appropriate docker-compose binary from GitHub:

  sudo curl -L https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m) -o

  /usr/local/bin/docker-compose

- NOTE: to get the latest version (thanks @spodnet):

  sudo curl -L https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose

- Fix permissions after download:

  sudo chmod +x /usr/local/bin/docker-compose

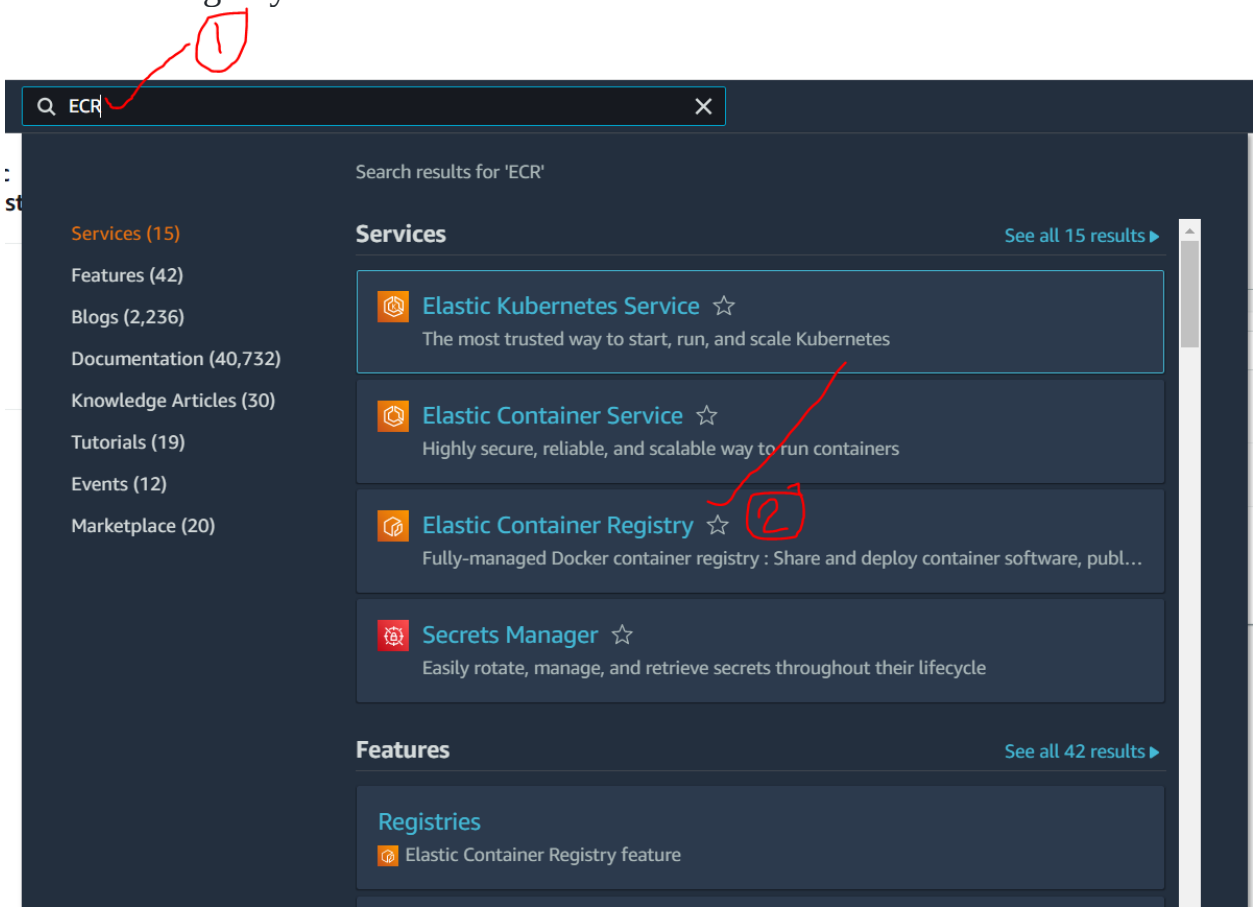
- Verify success:

docker-compose version

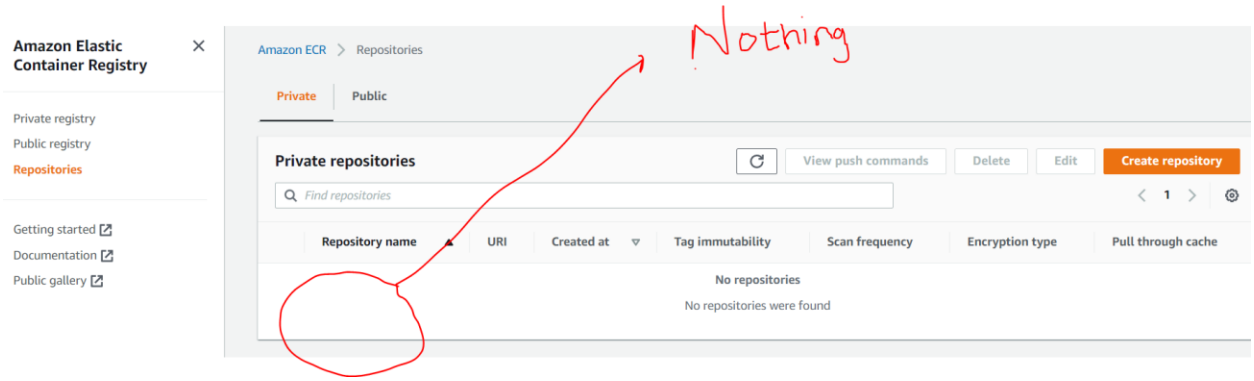
We are ready to create our first ECR

Let confirm we do not have any existing ECR in our Amazon console

- ⇨ Go to Amazon Service search bar and type ECR and click on Elastic Container Registry



Confirm you do not have ECR currently running



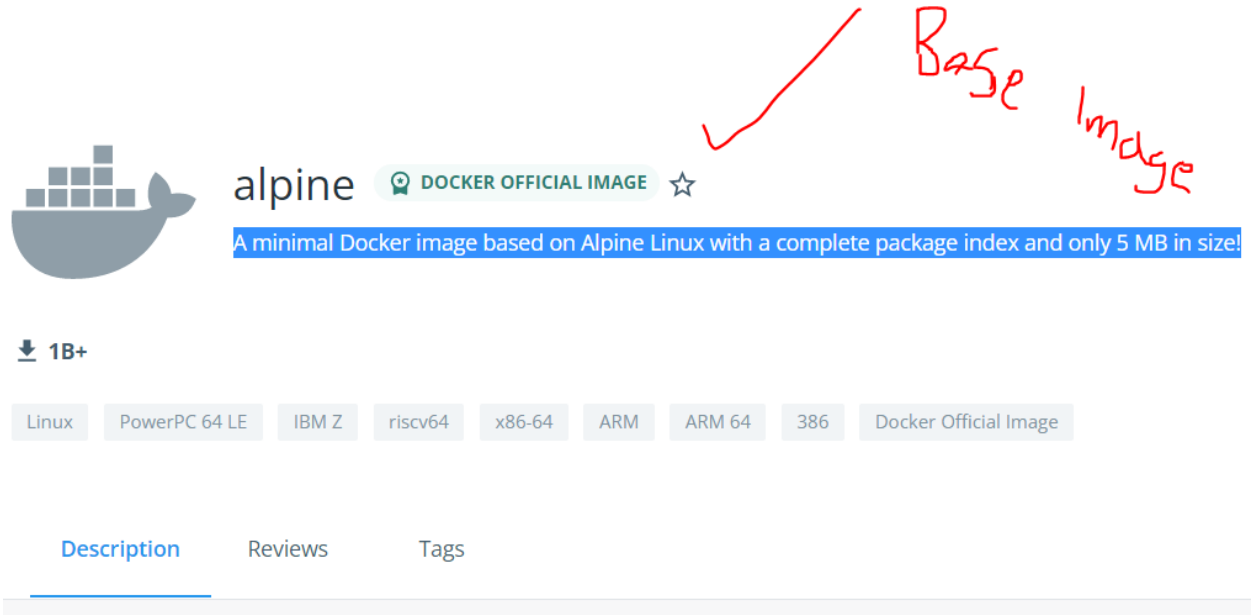
My goal is for us to create an ECR from the Command line

Important

We are going to deal with different versions of Alpine Image

Alpine image is a lightweight image that is used as a base image in many image applications used in the production environment in many companies.

The number of downloads evidence the importance of this image.



We are going to RUN and SCAN different versions of the Alpine images for Vulnerability issues.

Versions we are going to RUN and SCAN

Source: https://hub.docker.com/_/alpine

Supported tags and respective `Dockerfile` links

- `20220328`, `edge`
- `3.15.4`, `3.15`, `3`, `latest`
- `3.14.6`, `3.14`
- `3.13.10`, `3.13`
- `3.12.12`, `3.12`

Go to your LOCAL COMMAND LINE and run the following commands

```
[ec2-user@ip-10-0-25-228 ~]$ rm -f $(docker ps -a -q)
```

Remove all the containers

```
[ec2-user@ip-10-0-25-228 ~]$ docker rmi $(docker images -q)
```

Remove all the images

```
[ec2-user@ip-10-0-25-228 ~]$ docker rmi -f $(docker images -q)
```

Use to remove all the images that are tagged and referenced in different repositories

Confirm no images, and containers exist

```
[ec2-user@ip-10-0-25-228 ~]$ docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
[ec2-user@ip-10-0-25-228 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-10-0-25-228 ~]$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-10-0-25-228 ~]$
```

This is my local command line (Visual Studio) interface that I am using.

```
TERMINAL
[ec2-user@ip-10-0-25-228 ~]$
[ec2-user@ip-10-0-25-228 ~]$
[ec2-user@ip-10-0-25-228 ~]$
[ec2-user@ip-10-0-25-228 ~]$
```

The default user is the ec2-user. This is because I did SSH to the instance running on AWS.

Refer to my notes on how to SSH to an instance running on AWS Console.

Let us pull the first image

Supported tags and respective `Dockerfile` links

- `20220328`, `edge`
- 1 2 3
- `3.15.4`, `3.15`, `3`, `latest`
- `3.14.6`, `3.14`
- `3.13.10`, `3.13`
- `3.12.12`, `3.12`

Run the following command

docker pull alpine:latest

```
[ec2-user@ip-10-0-25-228 ~]$ docker pull alpine:latest
latest: Pulling from library/alpine
df9b9388f04a: Pull complete
Digest: sha256:4edbd2beb5f78b1014028f4fbb99f3237d9561100b6881aabbf5acce2c4f9
454
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
[ec2-user@ip-10-0-25-228 ~]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
alpine       latest    0ac33e5f5afa   4 weeks ago   5.57MB
[ec2-user@ip-10-0-25-228 ~]$
```

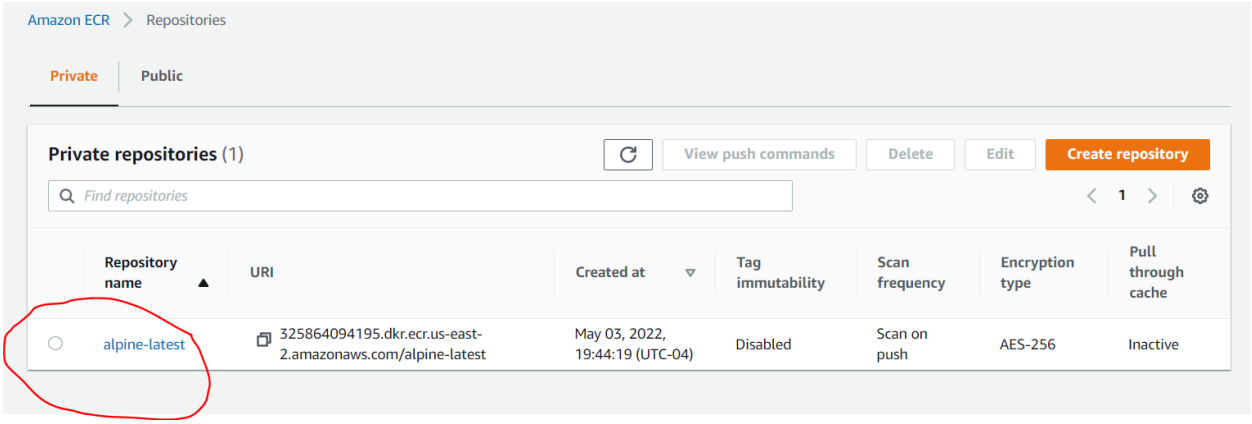
Create a repository from Local CLI (Command Line Interface)

Run the following command

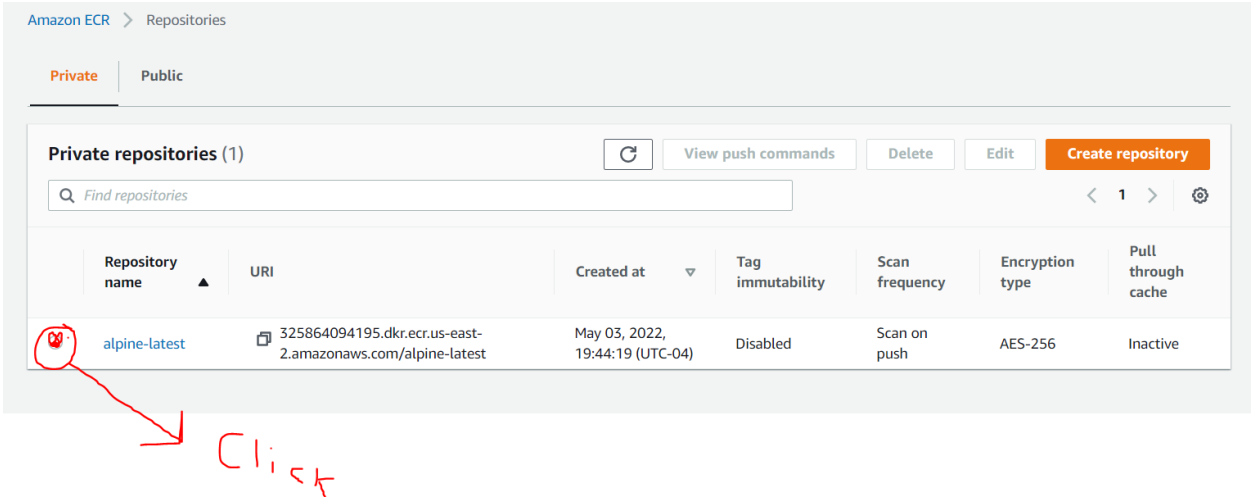
```
aws ecr create-repository \
  --repository-name alpine-latest \
  --image-scanning-configuration scanOnPush=true \
  --region us-east-2
```

```
[ec2-user@ip-10-0-25-228 ~]$ aws ecr create-repository \
>     --repository-name alpine-latest \
>     --image-scanning-configuration scanOnPush=true \
>     --region us-east-2
{
    "repository": {
        "repositoryUri": "325864094195.dkr.ecr.us-east-2.amazonaws.com/alpin
e-latest",
        "imageScanningConfiguration": {
            "scanOnPush": true
        },
        "encryptionConfiguration": {
            "encryptionType": "AES256"
        },
        "registryId": "325864094195",
        "imageTagMutability": "MUTABLE",
        "repositoryArn": "arn:aws:ecr:us-east-2:325864094195:repository/alpi
ne-latest",
        "repositoryName": "alpine-latest",
```

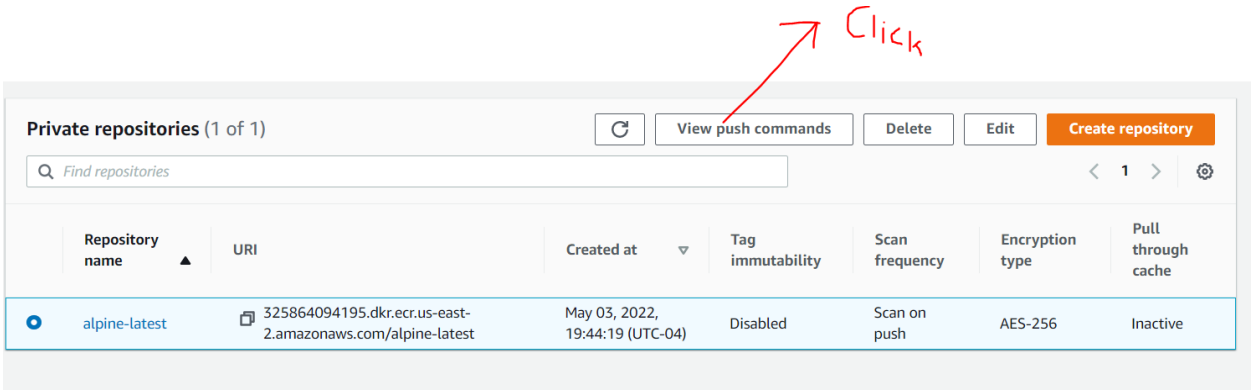
Go to Amazon Console and Confirm the ECR has been Created



ECR successfully created!!!



Click on View Push Commands



I am using Linux (Amazon Linux)

Linux

Push commands for alpine-latest

macOS / Linux | Windows

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-2 | docker login --username AWS --password-stdin 325864094195.dkr.ecr.us-east-2.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 alpine-latest .
   ```

3. After the build completes, tag your image so you can push the image to this repository:

   ```
   docker tag alpine-latest:latest 325864094195.dkr.ecr.us-east-2.amazonaws.com/alpine-latest:latest
   ```

4. Run the following command to push this image to your newly created AWS repository:

   ```
   docker push 325864094195.dkr.ecr.us-east-2.amazonaws.com/alpine-latest:latest
   ```

View push command make it easier to run, build, tag, and push repository

For this case, we just running, tagging, and Pushing (No building).

Allow us to successfully login into the ECR we just created.

```
[ec2-user@ip-10-0-25-228 ~]$ aws ecr get-login-password --region us-east-2 |
 docker login --username AWS --password-stdin 325864094195.dkr.ecr.us-east-2
.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/
config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-stor
e

Login Succeeded
```

Let us run

docker images again

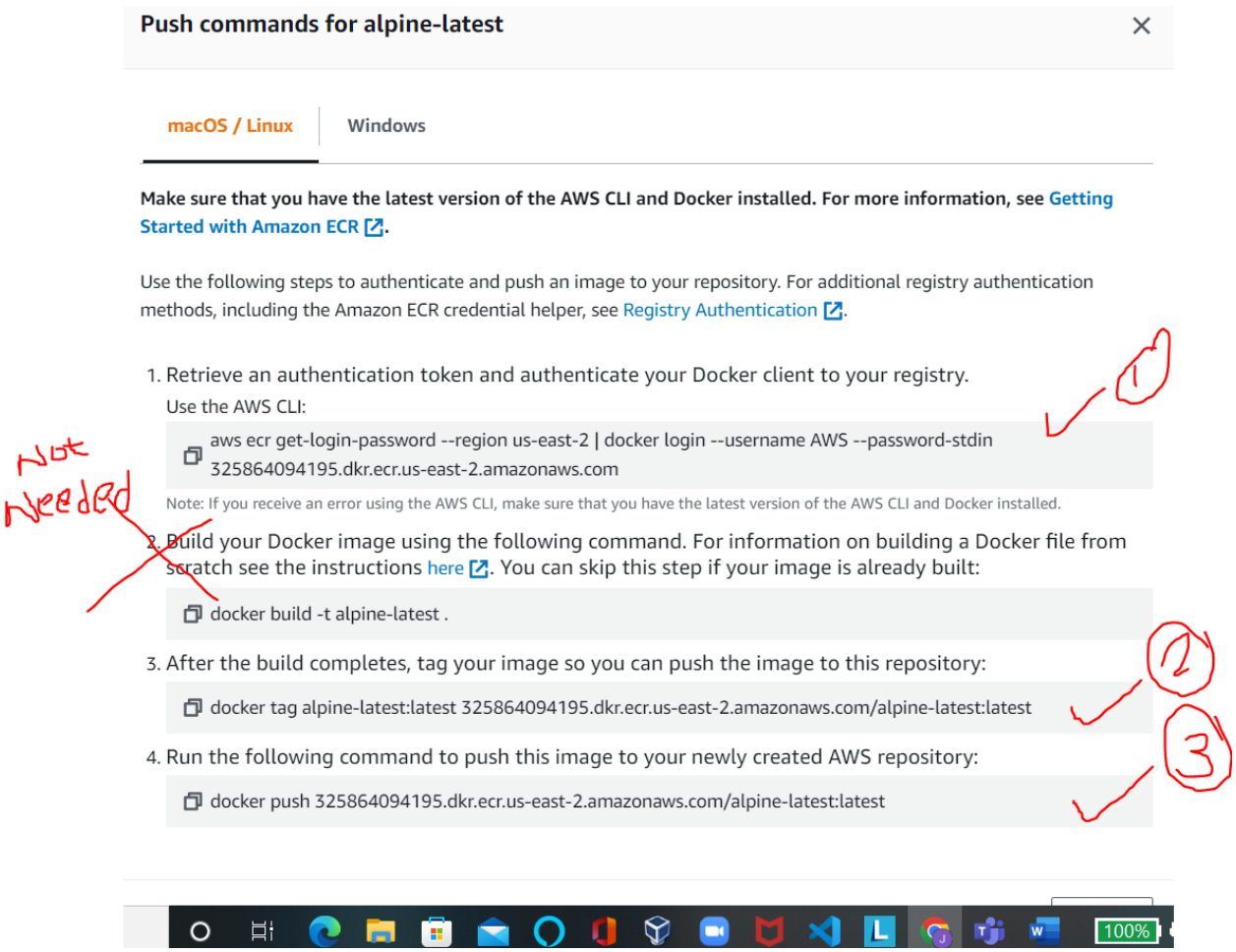
Attn!!!!

```
[ec2-user@ip-10-0-25-228 ~]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
alpine       latest    0ac33e5f5afa   4 weeks ago   5.57MB
[ec2-user@ip-10-0-25-228 ~]$
```

Pay attention to the TAG name.

In our case, the TAG name is: latest

Let us evaluate the view push command for the TAG



In the command number 2, circled in red, the actual name of the TAG is alpine-latest:latest

We need to adjust this name because we do not have an image with this name

Adjusted View Push Command

```
docker tag alpine:latest 325864094195.dkr.ecr.us-east-2.amazonaws.com/alpine:latest
```

Which now resembles the image running in our local CLI

Attn!!!!

```
[ec2-user@ip-10-0-25-228 ~]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
alpine       latest    0ac33e5f5afa   4 weeks ago   5.57MB
[ec2-user@ip-10-0-25-228 ~]$
```

Let tag this image

Success

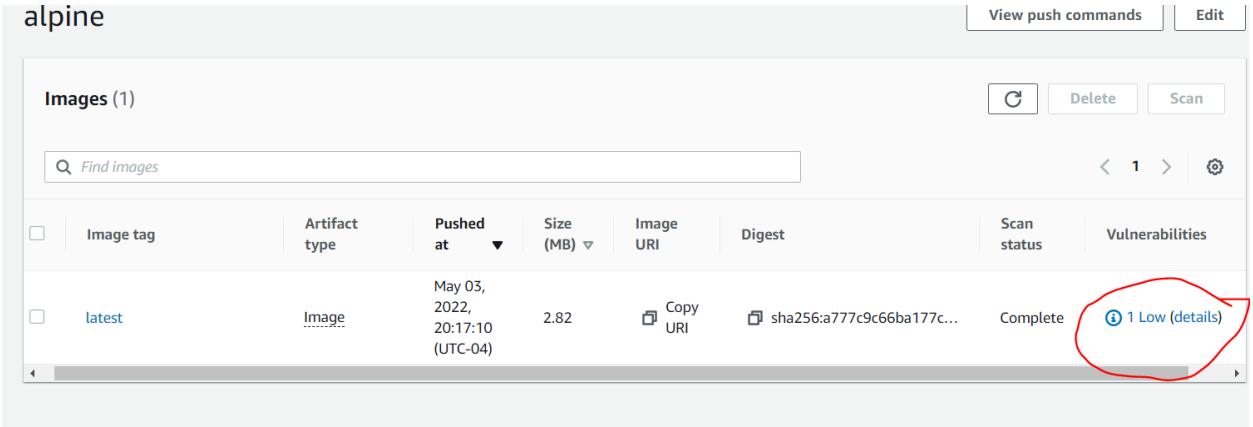
```
[ec2-user@ip-10-0-25-228 ~]$ docker tag alpine:latest 325864094195.dkr.ecr.u
s-east-2.amazonaws.com/alpine:latest
[ec2-user@ip-10-0-25-228 ~]$ docker images
REPOSITORY                                          TAG       IMAGE ID
   CREATED       SIZE
325864094195.dkr.ecr.us-east-2.amazonaws.com/alpine   latest    0ac33e5f5afa
   4 weeks ago   5.57MB
alpine                                              latest    0ac33e5f5afa
   4 weeks ago   5.57MB
```

Let push the image to ECR in the Amazon Console

Run the third command circled in red

```
[ec2-user@ip-10-0-25-228 ~]$ docker push 325864094195.dkr.ecr.us-east-2.amaz
onaws.com/alpine:latest
The push refers to repository [325864094195.dkr.ecr.us-east-2.amazonaws.com/
alpine]
4fc242d58285: Pushed
latest: digest: sha256:a777c9c66ba177ccfea23f2a216ff6721e78a662cd17019488c41
7135299cd89 size: 528
```

Go to the Amazon Console and confirm the Image has been scanned



Vulnerability exists but it is very low.

This image is secure to use in production environment

Useful Resources

https://docs.aws.amazon.com/AmazonECR/latest/userguide/getting-started-cli.html

https://gist.github.com/npearce/6f3c7826c7499587f00957fee62f8ee9