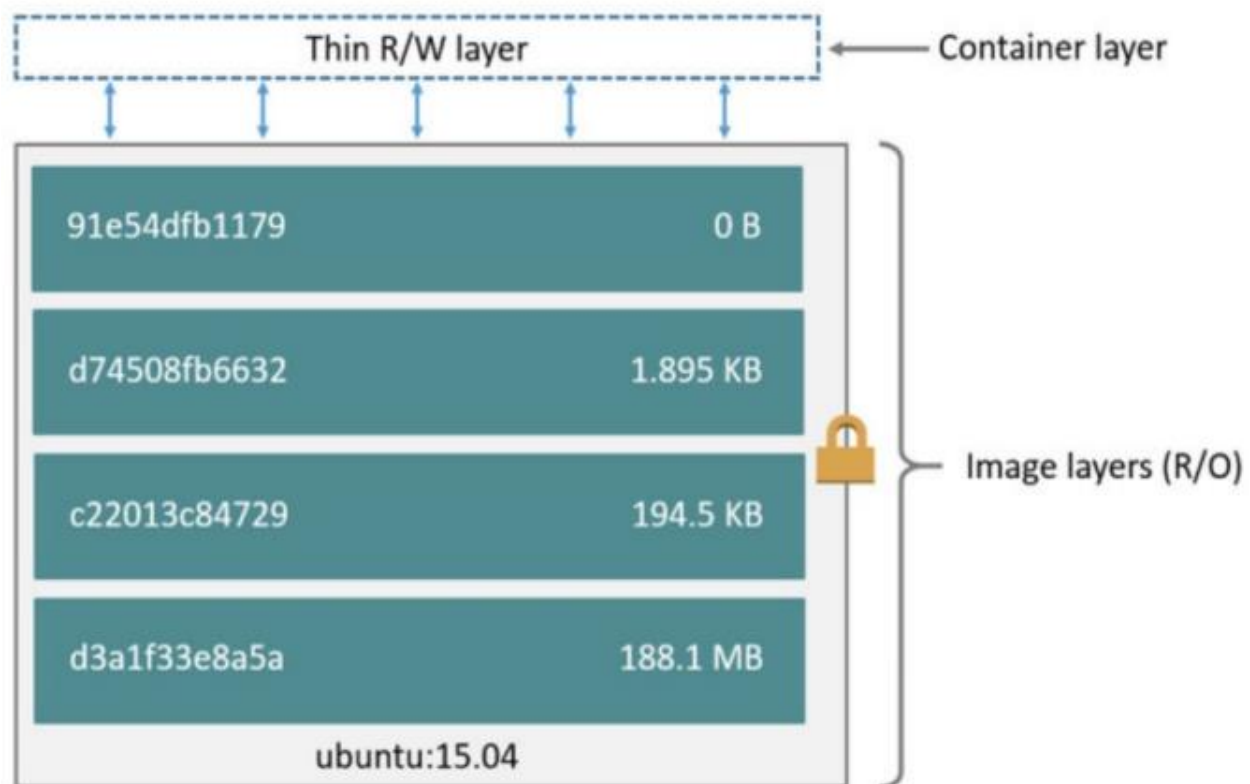


Kubernetes (The Defector Production Environment) Security Guide

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} ✓ Base Image

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+Master
# 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.lastExecVersion
    mkdir -p /usr/share/jenkins/ref/secrets/ && echo false > /usr/share/jenkins/ref/secrets/initialAdminPassword
    echo ${JENKINS_VER}-${RELEASE} > /usr/share/jenkins/ref/jenkins.docker.image.version
```

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.

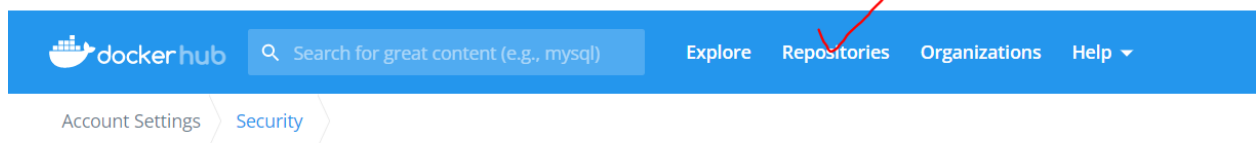
The image shows a comparison of four Docker pricing plans: Personal, Pro, Team, and Business. A handwritten red circle with '10X' is drawn around the Personal plan's price (\$0), and a red checkmark is drawn next to it. The Business plan's 'Contact Sales' button is also highlighted with a red checkmark.

Personal	Pro	Team	Business
Ideal for individual developers, education, open source communities, and small businesses.	Includes pro tools for individual developers who want to accelerate their productivity.	Ideal for teams and includes capabilities for collaboration, productivity and security.	Ideal for medium and large businesses who need centralized management and advanced security capabilities.
\$0	\$5 /month	\$7 /user/month Start with minimum 5 users for \$25.	\$21 /user/month
<ul style="list-style-type: none">• Docker Desktop• Unlimited public repositories• Docker Engine + Kubernetes• 200 image pulls per 6 hours• Unlimited scoped tokens	<p>← Everything in Personal plus:</p> <ul style="list-style-type: none">• Docker Desktop• Unlimited private repositories• 5,000 image pulls per day• 5 concurrent builds• 300 Hub vulnerability scans	<p>← Everything in Pro, plus:</p> <ul style="list-style-type: none">• Docker Desktop• Unlimited teams• 15 concurrent builds• Unlimited image scans• Role-based access control• Audit logs	<p>← Everything in Team, plus:</p> <ul style="list-style-type: none">• Docker Desktop• Centralized management• Image Access Management• Single Sign-On (SSO)• Purchase via invoice• Volume Pricing Available
Start Now	Buy Now <small>Billed annually for \$60.</small>	Buy Now <small>Billed annually starting at \$300.</small>	Contact Sales Buy Now

Login to your docker hub account

⇒ <https://hub.docker.com/repositories>

⇒ Go to repositories



⇒ Click on any of the images you have

josh1991	Search by repository name	Create Repository
josh1991 / king-httpd Last pushed: 3 months ago	Not Scanned	0 stars, 185 downloads, Public
josh1991 / bottle-httpd Last pushed: Never	Not Scanned	0 stars, 0 downloads, Public
josh1991 / advanced-nginx Last pushed: Never	Not Scanned	0 stars, 0 downloads, Public
josh1991 / httpd-group Last pushed: 4 months ago	Not Scanned	16 downloads, Public

⇒ Go to settings

josh1991 > Repositories > king-httpd > Using 0 of 1 private repositories. [Get more](#)

General Tags Builds Collaborators Webhooks **Settings**

Advanced Image Management
View all your images and tags in this repository, clean up unused content, recover untagged images. Available with Pro, Team and Business subscriptions. [View preview](#)

josh1991 / king-httpd
This repository does not have a description
Last pushed: 3 months ago

Docker commands [Public View](#)
To push a new tag to this repository,
`docker push josh1991/king-httpd:tagname`

Tags and Scans **VULNERABILITY SCANNING - DISABLED**
This repository contains 2 tag(s). [Enable](#)

Automated Builds
Manually pushing images to Hub? Connect your account to GitHub or



Vulnerability Scanning
Enable Vulnerability Scanning to identify vulnerabilities in your container images. Images you push to Hub after enabling this option will be automatically scanned.

[Enable Image Scanning](#) ☒ Vulnerability Scanning is off

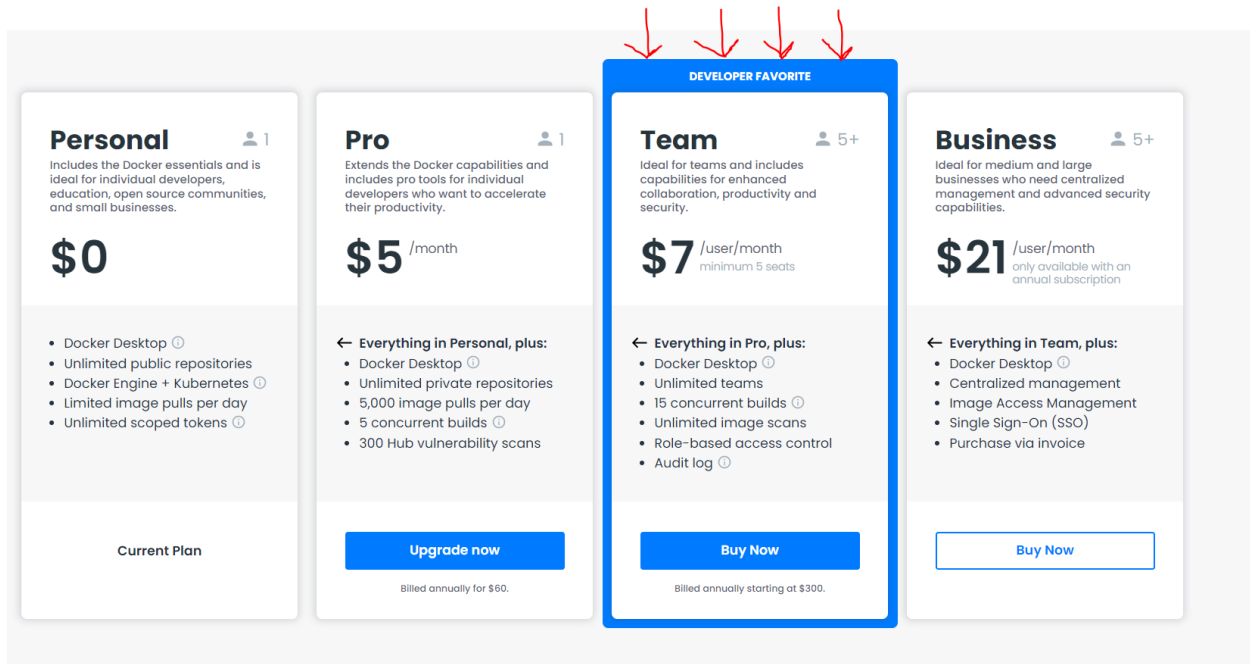
Visibility Settings
Using 0 of 1 private repositories. [Get more](#)

How likely are you to recommend Docker Hub to another developer?



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).

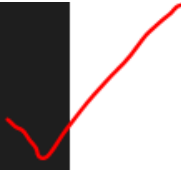
Dynamic scanning - You can scan containers running in Docker engine (Run Time)

⇒ 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
```

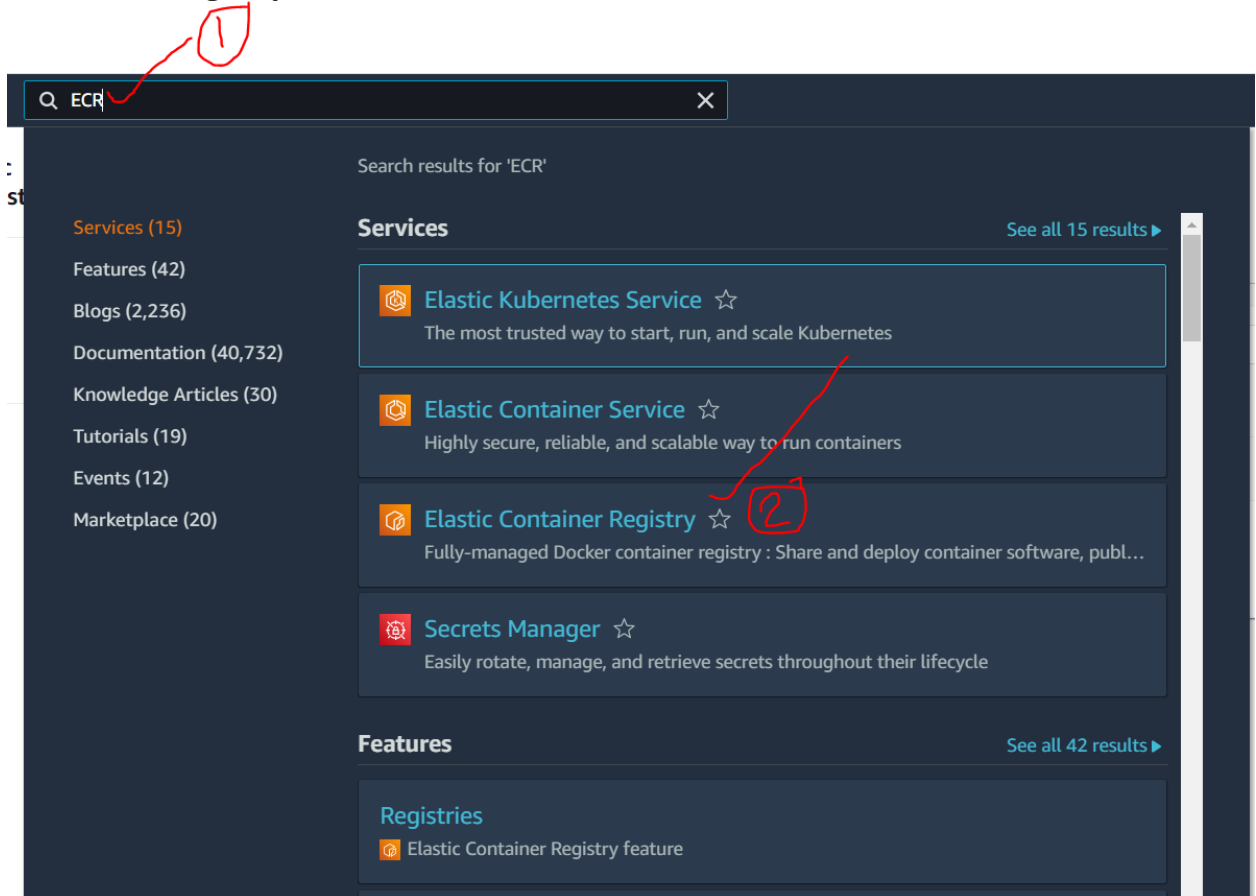
⇒ Verify success:

docker-compose version

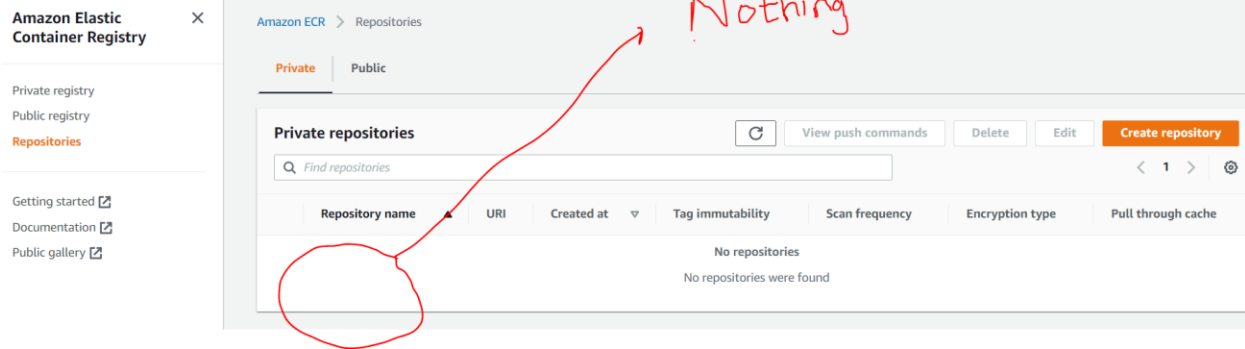
We are ready to create our first ECR

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



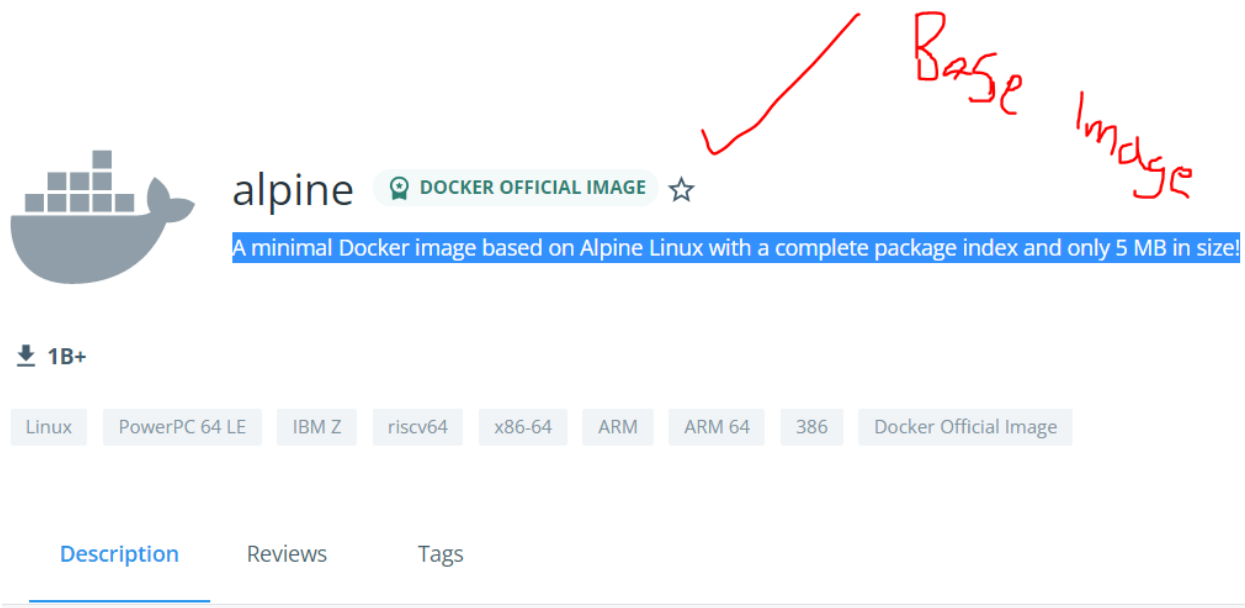
My goal is for us to create an ECR from the Command line

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
- ① 3.15.4 , ② 3.15 , ③ 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 3.15.4 , ② 3.15 , ③ 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: sha256:4edbd2beb5f78b1014028f4fbb99f3237d9561100b6881aabbf5acce2c4f9454
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/alpine-latest",
    "imageScanningConfiguration": {
      "scanOnPush": true
    },
    "encryptionConfiguration": {
      "encryptionType": "AES256"
    },
    "registryId": "325864094195",
    "imageTagMutability": "MUTABLE",
    "repositoryArn": "arn:aws:ecr:us-east-2:325864094195:repository/alpine-latest",
    "repositoryName": "alpine-latest",
```

Go to Amazon Console and Confirm the ECR has been Created

The screenshot shows the Amazon ECR console with the 'Private' tab selected. Under 'Private repositories (1)', there is a table with one repository listed: 'alpine-latest'. A red circle highlights the repository name. The table columns are: Repository name, URI, Created at, Tag immutability, Scan frequency, Encryption type, and Pull through cache.

Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	Pull through cache
alpine-latest	325864094195.dkr.ecr.us-east-2.amazonaws.com/alpine-latest	May 03, 2022, 19:44:19 (UTC-04)	Disabled	Scan on push	AES-256	Inactive

ECR successfully created!!!

This screenshot is identical to the previous one, but with a red circle around the 'alpine-latest' repository name and a red arrow pointing to it from the word 'Click' written in red.

Click

Click on View Push Commands

This screenshot shows the same interface, but with a red arrow pointing to the 'View push commands' button. The word 'Click' is written in red next to the arrow.

Click

I am using Linux (Amazon 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

ATTENTION!!!

```
[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

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
```

Not needed

1

2

3



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.us-east-2.amazonaws.com/alpine:latest
[ec2-user@ip-10-0-25-228 ~]$ docker images
```

REPOSITORY	SIZE	TAG	IMAGE ID
325864094195.dkr.ecr.us-east-2.amazonaws.com/alpine	4 weeks ago 5.57MB	latest	0ac33e5f5afa
alpine	4 weeks ago 5.57MB	latest	0ac33e5f5afa

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.amazonaws.com/alpine:latest
The push refers to repository [325864094195.dkr.ecr.us-east-2.amazonaws.com/alpine]
4fc242d58285: Pushed
latest: digest: sha256:a777c9c66ba177ccfea23f2a216ff6721e78a662cd17019488c417135299cd89 size: 528
```

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

alpine

View push commands Edit

Images (1)

Find images

	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
<input type="checkbox"/>	latest	Image	May 03, 2022, 20:17:10 (UTC-04)	2.82	Copy URI	sha256:a777c9c66ba177c...	Complete	1 Low (details)

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>