

## Deploying a Java Application - AWS Elastic Container Service, Elastic Container Registry, Jenkins Pipeline, Containerization, DockerHub

### Major Objectives:

- Use ECS to launch a cluster and AWS Fargate to manage a container that will host a Jenkins Main Server
- This Jenkins container will come from an AWS ECR image that we create and push up
- Configure that Jenkins Server to build on an Agent which will be an Ubuntu EC2 Instance
- This agent will pull a Java application (jar file) from a Github Repository and Containerize it using Docker
- Finally the Agent will create an image of Java app and push it to DockerHub

### Part 1 - Creating the EC2 Build Agent

1. **Create an EC2** using Ubuntu Ami, I used us-east-2
  - a. Under subnet setting, selected default subnet 2a
2. **Install Java** in the EC2
  - a. `sudo apt update && sudo apt upgrade -y`
  - b. `sudo apt install default-jre`
3. **Install Docker** on the EC2 by sshing into it and running the following
  - a. `curl -fsSL https://get.docker.com/ -o get-docker.sh`
  - b. `sudo sh get-docker.sh`

You can also install docker by Downloading and running a different sh script

- c. Can navigate to the URL to check contents of install script

Also can use following commands:

<https://docs.docker.com/engine/install/ubuntu/>

`sudo apt-get update`

`sudo apt install apt-transport-https ca-certificates curl`

`software-properties-common -y`

`curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`

`sudo add-apt-repository "deb [arch=amd64]`

`https://download.docker.com/linux/ubuntu focal stable"`

`sudo apt update`

`sudo apt-cache policy docker-ce`

`sudo apt install docker-ce -y`

`sudo systemctl status docker`

`sudo usermod -aG docker ubuntu`

Eventually you will need to ssh into this instance and run the following command (docker does not allow non root users to use docker so this fixes that)

`usermod -aG docker $USER`

## **Part 2 - Generate a Jenkins Image to push up to Amazon ECR (2 options)**

d. Inside of Ubuntu EC2

e. `sudo apt install aws cli`

f. <https://linuxide.com/how-to-install-aws-cli-on-ubuntu-20-04/>

g. `sudo apt install unzip`

h. Now need to configure aws cli in the ubuntu ec2

i. To do so, need to make new access key:

[https://console.aws.amazon.com/iam/home#/security\\_credentials](https://console.aws.amazon.com/iam/home#/security_credentials)

### **Generate the Jenkins image**

Can also run this command on local machine

i. `sudo docker pull jenkins/jenkins`

did this on my host computer, otherwise would have to update and install aws cli in ubuntu ec2

## **4. Create a Private ECR Repository**

a. Then run AWS push commands (1,3,4) to push jenkins image to the repo from wherever the image was made (your local machine or ec2)

i. Step 3, needs to say `docker tag <jenkins/jenkins>`, That is the name of the correct jenkins image

`docker tag jenkins/jenkins:latest`

`913777954597.dkr.ecr.us-east-2.amazonaws.com/dep7-java-docker:latest`

This uses the URI of the repository

## Push commands for dep7-java-docker



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-
```

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 dep7-java-docker .
```

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

```
docker tag dep7-java-docker:latest 913777954597.dkr.ecr.us-east-2.amazonaws.com/dep7-java-
```

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

```
docker push 913777954597.dkr.ecr.us-east-2.amazonaws.com/dep7-java-docker:latest
```

## Part 3 - Configure an ECS Cluster and your Jenkins Container

5. **Create a New Cluster** and simply name it for now
6. **Create a new Task Definition**.
  - a. Name it whatever you want
  - b. Give memory and cpu
  - c. Add Container:
    - i. Provide any name
    - ii. Copy and Paste ECR URI from Part 2 above
    - iii. And press make
7. **Go back to your Cluster to Configure your Fargate Resource**
  - a. Go to Tasks → Run new task
  - b. Select Launch type Fargate
  - c. Select your Cluster
  - d. Select default vpc and select the same subnet you made the Ubuntu EC2 in in Part 1

- e. Then go to security groups → this needs to have port 8080 open
- 8. Now you can click on the task name and go to **public url to Access your Jenkins Main**
  - a. Copy the Tasks Public Ip and go to <publicip:8080>
  - b. Then go to logs
    - i. And find the line that says var/jenkins, and find the initial admin password
- 9. **Setup Jenkins** with following plugins
  - a. Recommended plugins
  - b. Docker pipeline plugin
  - c. Amazon ec2 plugin
- 10. Next We Set Up our **Github Repo with 3 main parts**
  - a. **A DockerFile**, this will build an image of the Java app from .jar file found in the github repo

3 lines (3 sloc) | 85 Bytes

```
1 FROM openjdk:16
2 COPY ./demo-0.0.1-SNAPSHOT.jar app.jar
3 CMD ["java", "-jar", "app.jar"]
```

**b. A JenkinsFile (see below)**

- i. **Security Point:** here is a case where one should pay attention to private credentials being present in a code file. There are a number of ways to hide your credentials while authenticating.
- ii. Your Jenkinsfile credentials could just be **credentials <dockerUserName>**
- iii. You could set up docker username and docker access token as a formal credential in the Main Jenkins UI

```

Users > franklinordonez > Desktop > Jenkinsfile
1 pipeline {
2   agent { label "dep7agent" }
3
4   environment{
5     DOCKERHUB_CREDENTIALS_USR = "fordonez20"
6     DOCKERHUB_CREDENTIALS_PSW = "0dec76c9-96ff-4a8a-8f81-6a526fb2e41c" //access token
7   }
8   stages {
9     stage ('Build') {
10      steps {
11        sh 'docker build -t dep7jenkinstodocker .'
12      }
13    }
14    stage ('Login') {
15      steps {
16        sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin'
17      }
18    }
19    stage ('Push') {
20      steps {
21        sh 'docker tag dep7jenkinstodocker:latest'
22        sh 'docker push fordonez20:dep7jenkinstodocker:latest'
23      }
24    }
25  }
26 }
27
28
29

```

c. Include a previously used Java Demo .jar file in the Git Repo

## Part 3 - Setup the Jenkins Pipeline and Build Agent Configuration

### 11. Next Set Up Pipeline and Agent In Jenkins

- New item in jenkins - named dep7build - selected Multibranch pipeline
- Add your git repo URL
- Use your credentials with Username and Access Token from Github
- Configuring the Build Agent Now (dep7agent) Using your Already Made Ubuntu EC2

#### Adding an Agent to your Jenkins Configuration

- In Jenkins UI navigate to Manage nodes, add node (agent name in jenkinsfile)
- Host → private ip (of EC2)
- Launch agent via ssh, use private ip in Host section
- Remote root directory **/home/ubuntu/jenkins**
- Credentials - ssh with private key - user
- Ubuntu is the username → Username must be username on instance (in this case it was ubuntu)
- Then paste private key that you used to ssh into the ubuntu ec2

## Part 4 - Finally Run your Pipeline Build on Jenkins

- a Jenkins Main configured on AWS ECS commands an EC2 in the same subnet of the Cluster to build a docker image
- The pipeline script is configured with DockerHub credentials and pushes an image up to DockerHub
- One could then run a container on an instance by pulling this image from DockerHub

**Jenkins** Search ? 1 Franklin Z. Ordonez log out

Dashboard > Dep7Build > main

Up

Status

Changes

Build Now

View Configuration

Full Stage View

GitHub

Pipeline Syntax

Build History trend

Filter builds...

#3 Oct 9, 2021, 9:56 PM

#2 Oct 9, 2021, 9:51 PM

#1 Oct 9, 2021, 9:01 PM

Atom feed for all Atom feed for failures

### Branch main

Full project name: Dep7Build/main

Recent Changes

### Stage View

Average stage times:  
(Average full run time: ~32s)

	Declarative: Checkout SCM	Build	Login	Push
#3 Oct 09 17:56 No Changes	5s	17s	911ms	5s
#2 Oct 09 17:51 5 commits	2s	900ms failed	113ms failed	111ms failed
#1 Oct 09 17:01 No Changes	19s	2s failed	561ms failed	110ms failed

### Permalinks


- Last build (#3), 1 min 4 sec ago
- Last stable build (#3), 1 min 4 sec ago
- Last successful build (#3), 1 min 4 sec ago
- Last failed build (#2), 5 min 59 sec ago
- Last unsuccessful build (#2), 5 min 59 sec ago
- Last completed build (#3), 1 min 4 sec ago



## Console Output


```
Started by user Franklin Z. Ordonez
21:56:08 Connecting to https://api.github.com using Fordonez20/*****
Obtained Jenkinsfile from 03e91c45e32c67d9410b7ba15c514b87f1d1f482
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on dep7agent in /home/ubuntu/jenkins/workspace/Dep7Build_main
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
The recommended git tool is: NONE
using credential laec2d02-3e18-44c3-8040-f47b694b374f
Fetching changes from the remote Git repository
Fetching without tags
Checking out Revision 03e91c45e32c67d9410b7ba15c514b87f1d1f482 (main)
> git rev-parse --resolve-git-dir /home/ubuntu/jenkins/workspace/Dep7Build_main/.git # timeout=10
> git config remote.origin.url https://github.com/Fordonez20/JenkinsToDockerDep7.git # timeout=10
Fetching upstream changes from https://github.com/Fordonez20/JenkinsToDockerDep7.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
using GIT_ASKPASS to set credentials
> git fetch --no-tags --force --progress -- https://github.com/Fordonez20/JenkinsToDockerDep7.git +refs/heads/main:refs/remotes/origin/main # timeout=10
> git config core.sparsecheckout # timeout=10
> git checkout -f 03e91c45e32c67d9410b7ba15c514b87f1d1f482 # timeout=10
Commit message: "Update Dockerfile"
> git rev-list --no-walk 03e91c45e32c67d9410b7ba15c514b87f1d1f482 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] sh
+ docker build -t fordonez20/dep7jenkinstodocker .
Sending build context to Docker daemon 35.76MB

Step 1/3 : FROM openjdk:16
16: Pulling from library/openjdk
6540cb32758b: Pulling fs layer
3acb1bb05d77: Pulling fs layer
```

 **dockerhub**


[Explore](#) [Repositories](#) [Organizations](#) [Help](#)


[Upgrade](#)


 **fordonez20**


[Create Repository](#)

**fordonez20 / dep7jenkinstodocker**  
Updated a few seconds ago

 Not Scanned

 0

 4

 Public