Objective: To automate the process of building a Docker image and pushing the image to Dockerhub using Jenkins and AWS ECS

1. Create an AWS EC2 with an Ubuntu ami. Then ssh into the EC2 and install docker on it using: sudo apt-get install \ apt-transport-https \ ca-certificates \ curl \ gnupg-agent \ software-properties-common curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - apt-cache madison docker-ce sudo apt install docker.io sudo apt install docker-compose sudo usermod -a -G docker ubuntu then exit out of the EC2 and ssh back into it
2. I built a Dockerfile that would create the image of the java application:

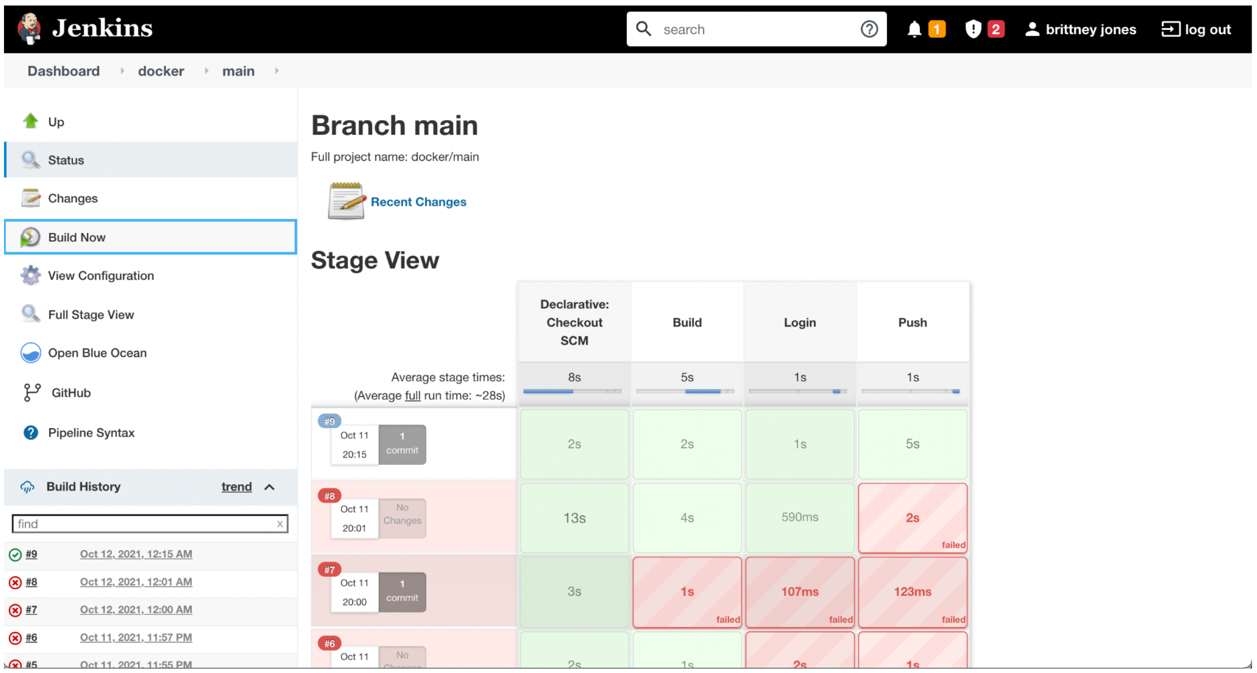
   ```
   FROM openjdk:11

   COPY demo-0.0.1-SNAPSHOT.jar .

   CMD ["java", "-jar","app.jar"]
   ```

   And added both the Dockerfile and application file (.jar) to my repository on Github
3. We also need to create a Dockerfile of the Jenkins image in order to use the Jenkins server on AWS ECS:

   ```
   FROM jenkins/jenkins:2.303.1-jdk11
    USER root
    RUN apt-get update && apt-get install -y apt-transport-https \
          ca-certificates curl gnupg2 \
          software-properties-common
    RUN curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add -
    RUN apt-key fingerprint 0EBFCD88
    RUN add-apt-repository \
          "deb [arch=amd64] https://download.docker.com/linux/debian \
          $(lsb_release -cs) stable"
    RUN apt-get update && apt-get install -y docker-ce-cli
    USER jenkins
    RUN jenkins-plugin-cli --plugins "blueocean:1.25.0 docker-workflow:1.26"
   ```

4. When you login to your AWS Account and Navigate to the Elastic Container Registry and create a repository which will contain the image we made of Jenkins
5. Using the push commands given in the repo we created, we will authenticate and push the image we have locally to AWS. Since I have an M1 Macbook I had to use the following command when building the image: docker buildx build --platform linux/amd64 -t jenkins-image .
6. Then we go to ECS and create a networking only cluster that will utilize Fargate, which is a serverless compute engine, meaning they manage the servers created for you.

7. After creating a cluster, you then need to create a Task Definition where you will link to the ECR repo where your image is stored using the URI and add port mapping so we are able to access our application.
8. Once the task is running navigate to the logs which is where you will find the initial temporary password to login to Jenkins
9. Once logged into Jenkins install recommended plugins, Docker pipeline plugin and AWS EC2 plugin
10. Create a new Pipeline and connect it to you Github Repo
11. Add the Jenkins file to the Github Repo that you will be using to create the image of the Java application, login and push to Docker Hub
12. Generate an access token from Docker Hub and add it to the global credentials in Jenkins along with your Docker Hub username
13. Create an agent Node on Jenkins on which to run your build
14. Run your build on Jenkins and then check Docker Hub to see if the image was successfully pushed
15. Stop the Task from running in your ECS Cluster and terminate the EC2



Branch main

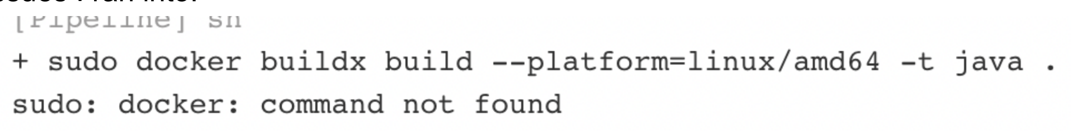
Full project name: docker/main

Stage View

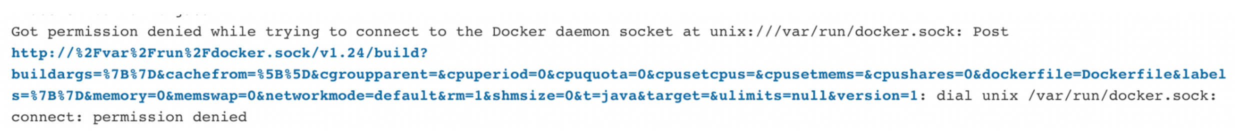
	Declarative: Checkout SCM	Build	Login	Push
Average stage times: (Average full run time: ~28s)	8s	5s	1s	1s
#9 Oct 11 20:15 1 commit	2s	2s	1s	5s
#8 Oct 11 20:01 No Changes	13s	4s	590ms	2s failed
#7 Oct 11 20:00 1 commit	3s	1s failed	107ms failed	123ms failed
#6 Oct 11 No Changes	2s	1s	2s	1s

#9	Oct 12, 2021, 12:15 AM
#8	Oct 12, 2021, 12:01 AM
#7	Oct 12, 2021, 12:00 AM
#6	Oct 11, 2021, 11:57 PM
#5	Oct 11, 2021, 11:55 PM

Issues I ran into:



```
[Pipeline] sh
+ sudo docker buildx build --platform=linux/amd64 -t java .
sudo: docker: command not found
```



```
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post
http://%2Fvar%2Frun%2Fdocker.sock/v1.24/build?
buildargs=%7B%7D&cachefrom=%5B%5D&cgroupparent=&cpuperiod=0&cpuquota=0&cpusetcpus=&cpusetmems=&cpushares=0&dockerfile=Dockerfile&label
s=%7B%7D&memory=0&memswap=0&networkmode=default&rm=1&shmsize=0&t=java&target=&ulimits=null&version=1: dial unix /var/run/docker.sock:
connect: permission denied
```

After installing Docker on my EC2 I needed to exit and ssh back in for changes to take effect.