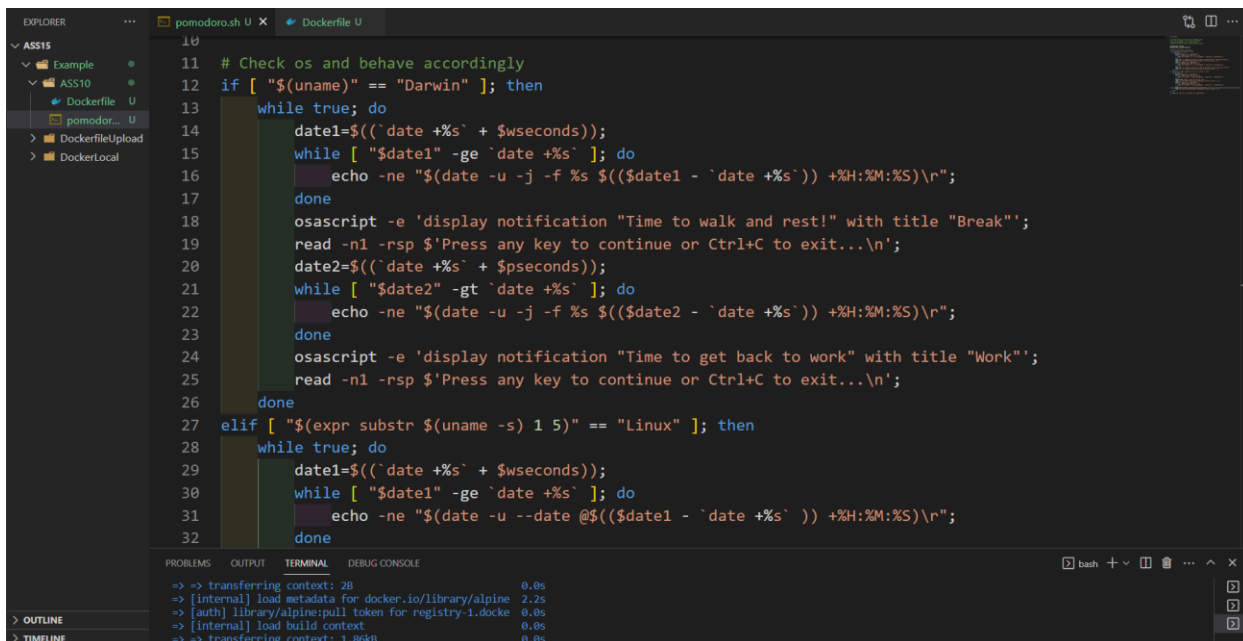


## Assignment – 10

Create a container for the below bash script(pomodoro.sh).

1. Create a Dockerfile
2. Create a Docker Image
3. Push the image to the docker hub
4. Pull it and test it.

Find the script from here <https://github.com/DARK-art108/Bash-Scripts>

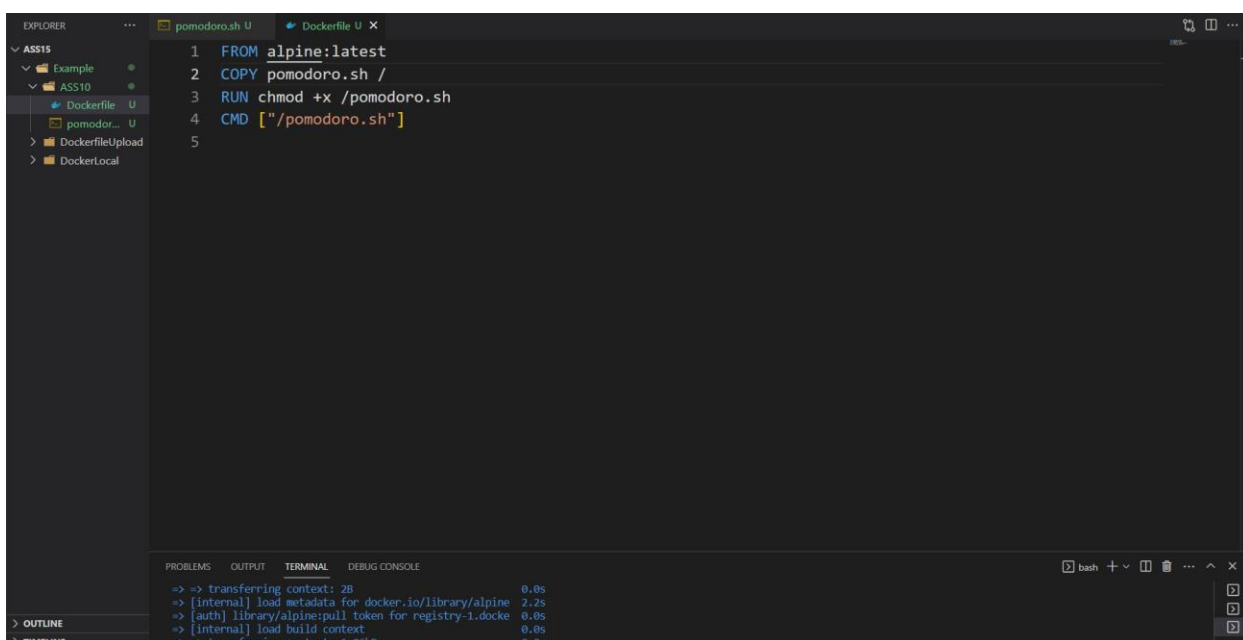


The screenshot shows the Visual Studio Code editor with a file named `pomodoro.sh` open. The script is a bash script that checks the operating system and behaves accordingly. It uses `while` loops and `osascript` to display notifications. The terminal window at the bottom shows the output of the script, which is a series of timestamps and notifications.

```
10
11 # Check os and behave accordingly
12 if [ "$(uname)" == "Darwin" ]; then
13     while true; do
14         date1=$((date +%s` + $wseconds));
15         while [ "$date1" -ge `date +%s` ]; do
16             echo -ne "$(date -u -j -f %s $((date1 - `date +%s`)) +%H:%M:%S)\n";
17         done
18         osascript -e 'display notification "Time to walk and rest!" with title "Break"';
19         read -n1 -rsp $'Press any key to continue or Ctrl+C to exit...\n';
20         date2=$((date +%s` + $pseconds));
21         while [ "$date2" -gt `date +%s` ]; do
22             echo -ne "$(date -u -j -f %s $((date2 - `date +%s`)) +%H:%M:%S)\n";
23         done
24         osascript -e 'display notification "Time to get back to work" with title "Work"';
25         read -n1 -rsp $'Press any key to continue or Ctrl+C to exit...\n';
26     done
27 elif [ "$(expr substr $(uname -s) 1 5)" == "Linux" ]; then
28     while true; do
29         date1=$((date +%s` + $wseconds));
30         while [ "$date1" -ge `date +%s` ]; do
31             echo -ne "$(date -u --date @$((date1 - `date +%s`)) +%H:%M:%S)\n";
32         done
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/alpine 2.2s
=> [auth] library/alpine:pull token for registry-1.docker.io 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 1.86kB 0.0s
```



The screenshot shows the Visual Studio Code editor with a file named `Dockerfile` open. The Dockerfile contains instructions to build a Docker image from the `alpine:latest` base image, copy the `pomodoro.sh` script into the image, and set the command to run the script. The terminal window at the bottom shows the output of the Docker build process, which is a series of timestamps and progress bars.

```
1 FROM alpine:latest
2 COPY pomodoro.sh /
3 RUN chmod +x /pomodoro.sh
4 CMD ["/pomodoro.sh"]
5
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/alpine 2.2s
=> [auth] library/alpine:pull token for registry-1.docker.io 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 1.86kB 0.0s
```

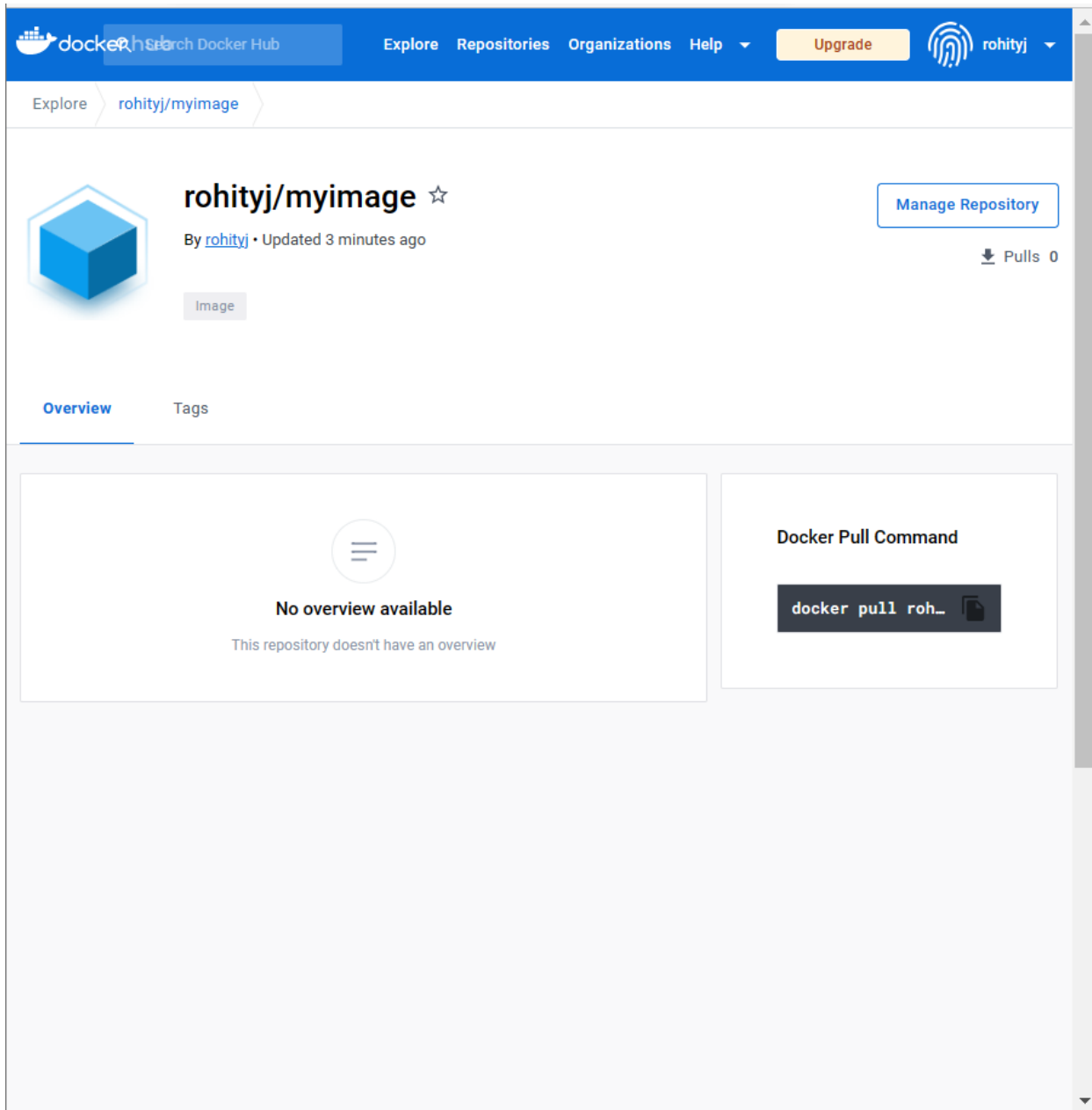
The image shows a VS Code editor window with a file explorer on the left and a terminal on the right. The file explorer shows a project structure with folders 'ASS15', 'Example', 'ASS10', and 'DockerfileUpload'. The 'DockerfileUpload' folder is expanded, showing 'pomodoro.sh' and 'Dockerfile'. The 'Dockerfile' is open in the editor, showing the following content:

```
1 FROM alpine:latest
2 COPY pomodoro.sh /
```

The terminal shows the output of the Docker build process. It starts with the command `docker build -t myimage .` and shows the progress of building the image. The output includes the following steps and durations:

- [+] Building 3.5s (9/9) FINISHED
- [internal] load build definition from Dockerfile 0.0s
- [internal] load .dockerignore 0.0s
- [internal] load metadata for docker.io/library/alpine 2.2s
- [auth] library/alpine:pull token for registry-1.docker 0.0s
- [internal] load build context 0.0s
- [internal] transfering context: 1.86kB 0.0s
- [internal] transfering context: 2B 0.0s
- [internal] load metadata for docker.io/library/alpine:latest@sha 0.0s
- [2/3] COPY pomodoro.sh / 0.0s
- [3/3] RUN chmod +x /pomodoro.sh 1.0s
- exporting to image 0.0s
- exporting layers 0.0s
- writing image sha256:487b710836caf5de356617639423f 0.0s
- namings to docker.io/library/myimage 0.0s

The terminal also shows the command `docker tag myimage rohityj/myimage` and the command `docker push rohityj/myimage`. The output of the push command shows the image being pushed to the repository [docker.io/rohityj/myimage] with the digest sha256:487b710836caf5de356617639423f. The image is pushed to the repository [docker.io/rohityj/myimage] with the digest sha256:487b710836caf5de356617639423f. The image is pushed to the repository [docker.io/rohityj/myimage] with the digest sha256:487b710836caf5de356617639423f.



```
PROBLEMS  OUTPUT  TERMINAL  ...  bash  +  -  [ ]  [X]  ...  ^  X

rohiti@rohit MINGW64 /d/DevOps/Assignment/ass15/Example/ASS10 (ma
in)
$ docker pull rohityj/myimage
Using default tag: latest
latest: Pulling from rohityj/myimage
Digest: sha256:5c4f556909dd6d1243ef0dd604639e07f4401d10a6a2c439a
0cc038044d38161
Status: Image is up to date for rohityj/myimage:latest
docker.io/rohityj/myimage:latest
```

