

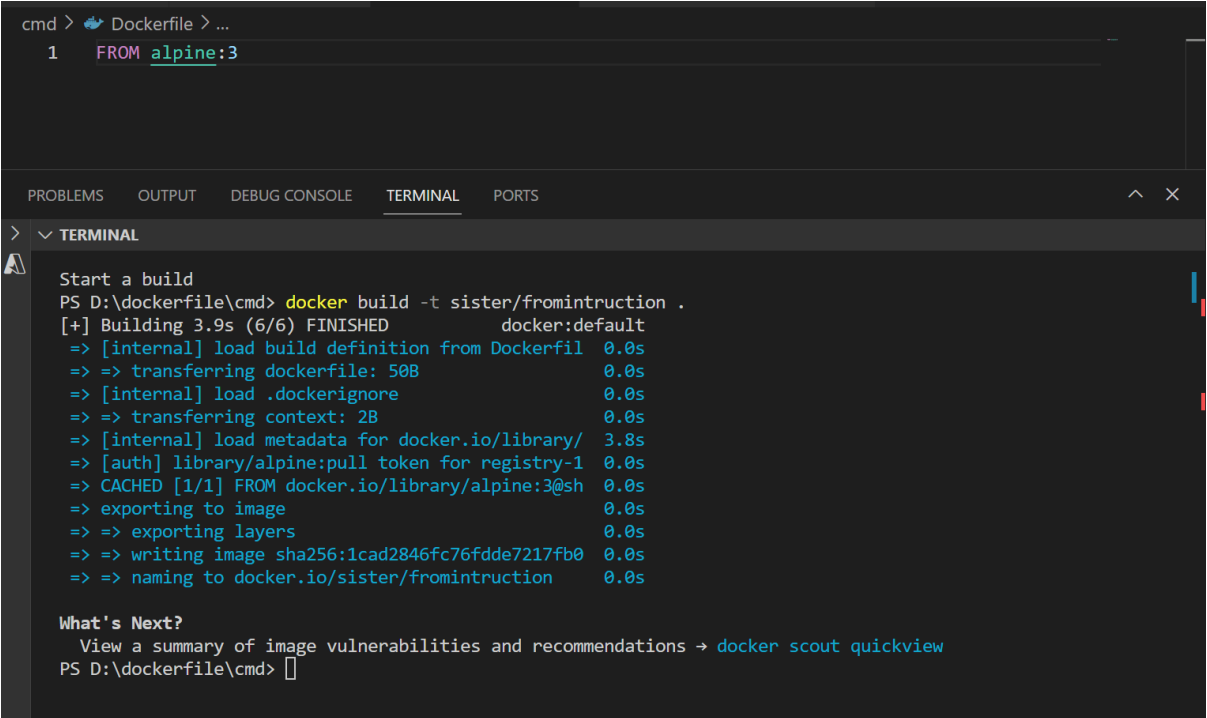
Nama : Ahmad Maulana Rismadin

NIM : 11201004

Github : https://github.com/rootAmr/Dockerfile_sistemterdistribusi

DOCKER FILE

1. From Instruction

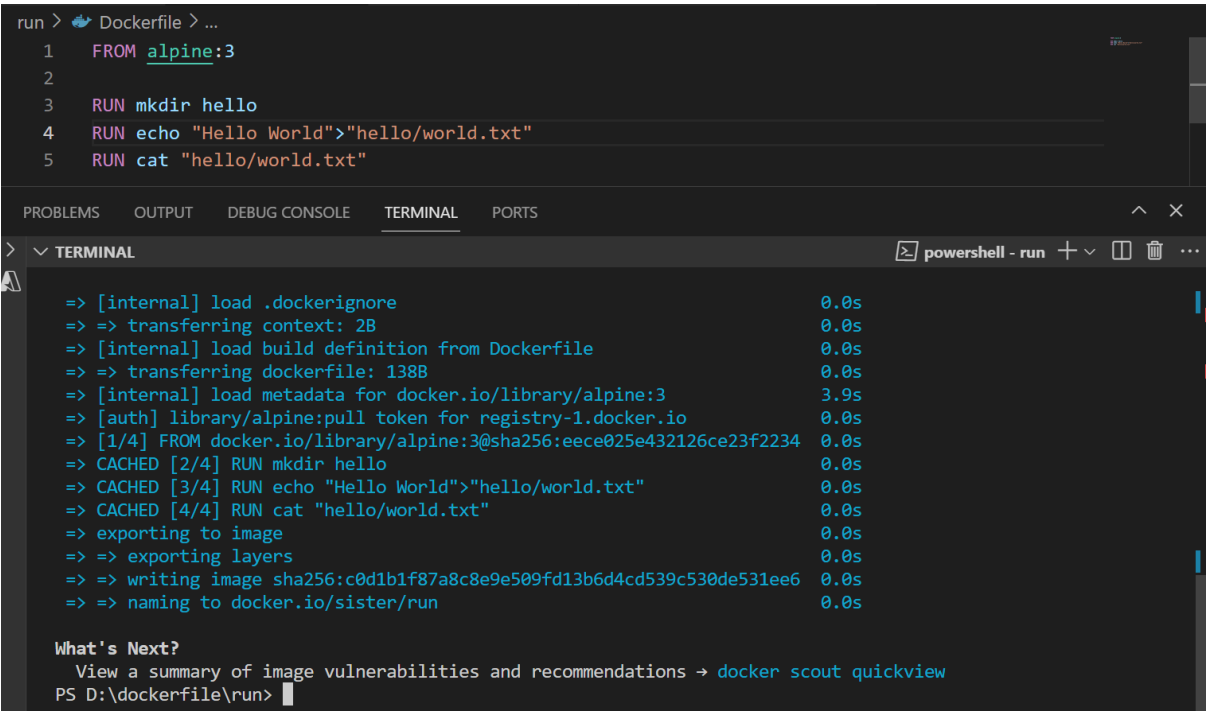


```
cmd > Dockerfile > ...
1 FROM alpine:3

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
> v TERMINAL
Start a build
PS D:\dockerfile\cmd> docker build -t sister/fromintruction .
[+] Building 3.9s (6/6) FINISHED docker:default
=> [internal] load build definition from Dockerfil 0.0s
=> => transferring dockerfile: 50B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/ 3.8s
=> [auth] library/alpine:pull token for registry-1 0.0s
=> CACHED [1/1] FROM docker.io/library/alpine:3@sh 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:1cad2846fc76fdde7217fb0 0.0s
=> => naming to docker.io/sister/fromintruction 0.0s

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS D:\dockerfile\cmd> 
```

2. Run Instruction

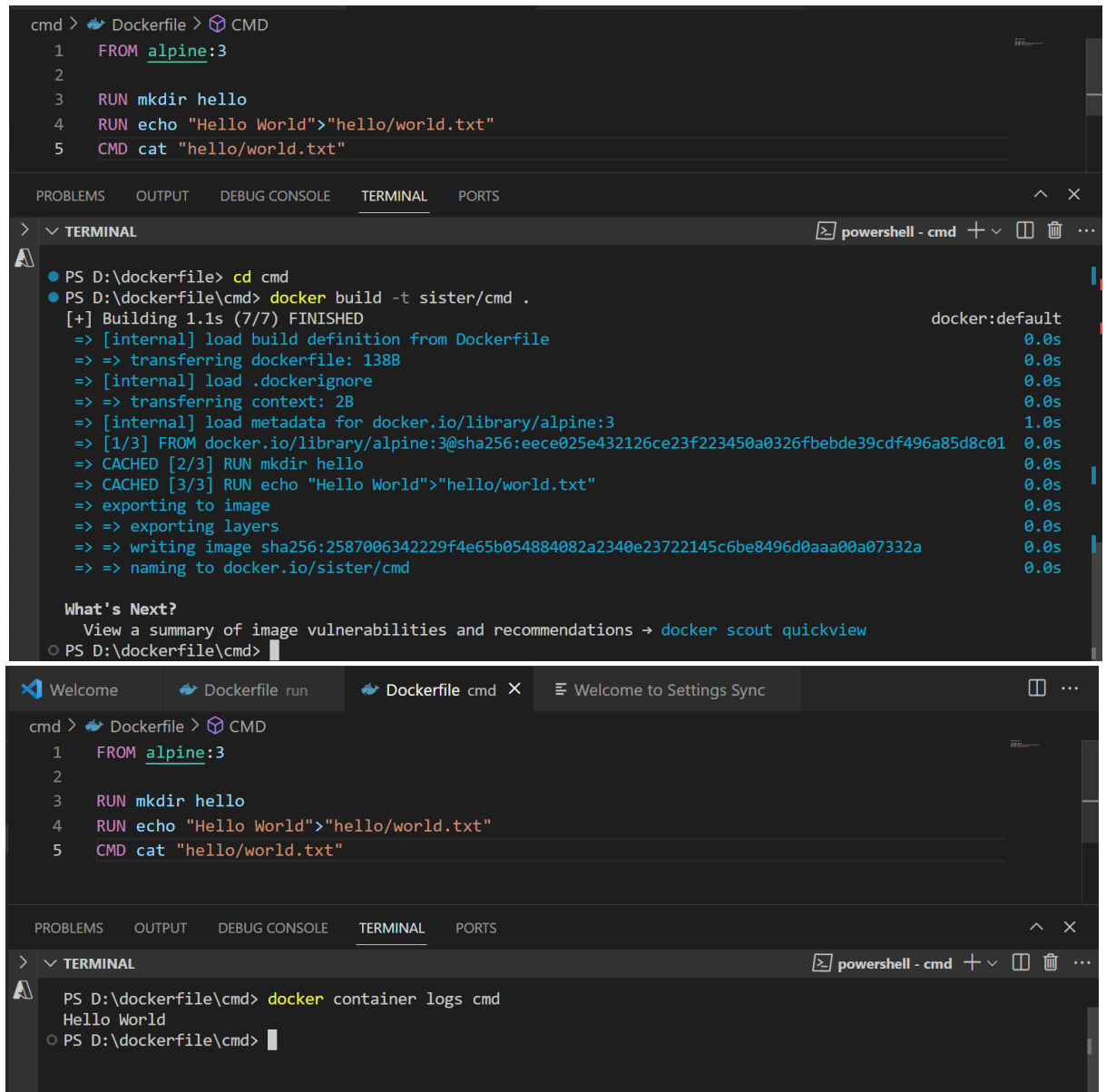


```
run > Dockerfile > ...
1 FROM alpine:3
2
3 RUN mkdir hello
4 RUN echo "Hello World">"hello/world.txt"
5 RUN cat "hello/world.txt"

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
> v TERMINAL
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 138B 0.0s
=> [internal] load metadata for docker.io/library/alpine:3 3.9s
=> [auth] library/alpine:pull token for registry-1.docker.io 0.0s
=> [1/4] FROM docker.io/library/alpine:3@sha256:eece025e432126ce23f2234 0.0s
=> CACHED [2/4] RUN mkdir hello 0.0s
=> CACHED [3/4] RUN echo "Hello World">"hello/world.txt" 0.0s
=> CACHED [4/4] RUN cat "hello/world.txt" 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:c0d1b1f87a8c8e9e509fd13b6d4cd539c530de531ee6 0.0s
=> => naming to docker.io/sister/run 0.0s

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS D:\dockerfile\run> 
```

3. Command Instruction



The image consists of two screenshots of a Visual Studio Code terminal window. The top screenshot shows the Docker build process for a container named 'cmd'. The bottom screenshot shows the command to view the logs of the 'cmd' container.

Top Screenshot: Docker Build Process

```
cmd > Dockerfile > CMD
1 FROM alpine:3
2
3 RUN mkdir hello
4 RUN echo "Hello World">"hello/world.txt"
5 CMD cat "hello/world.txt"
```

TERMINAL

```
PS D:\dockerfile> cd cmd
PS D:\dockerfile\cmd> docker build -t sister/cmd .
[+] Building 1.1s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 138B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/alpine:3
=> [1/3] FROM docker.io/library/alpine:3@sha256:eece025e432126ce23f223450a0326fbebde39cdf496a85d8c01
=> CACHED [2/3] RUN mkdir hello
=> CACHED [3/3] RUN echo "Hello World">"hello/world.txt"
=> exporting to image
=> => exporting layers
=> => writing image sha256:2587006342229f4e65b054884082a2340e23722145c6be8496d0aaa00a07332a
=> => naming to docker.io/sister/cmd
```

What's Next?
View a summary of image vulnerabilities and recommendations → [docker scout quickview](#)

PS D:\dockerfile\cmd>

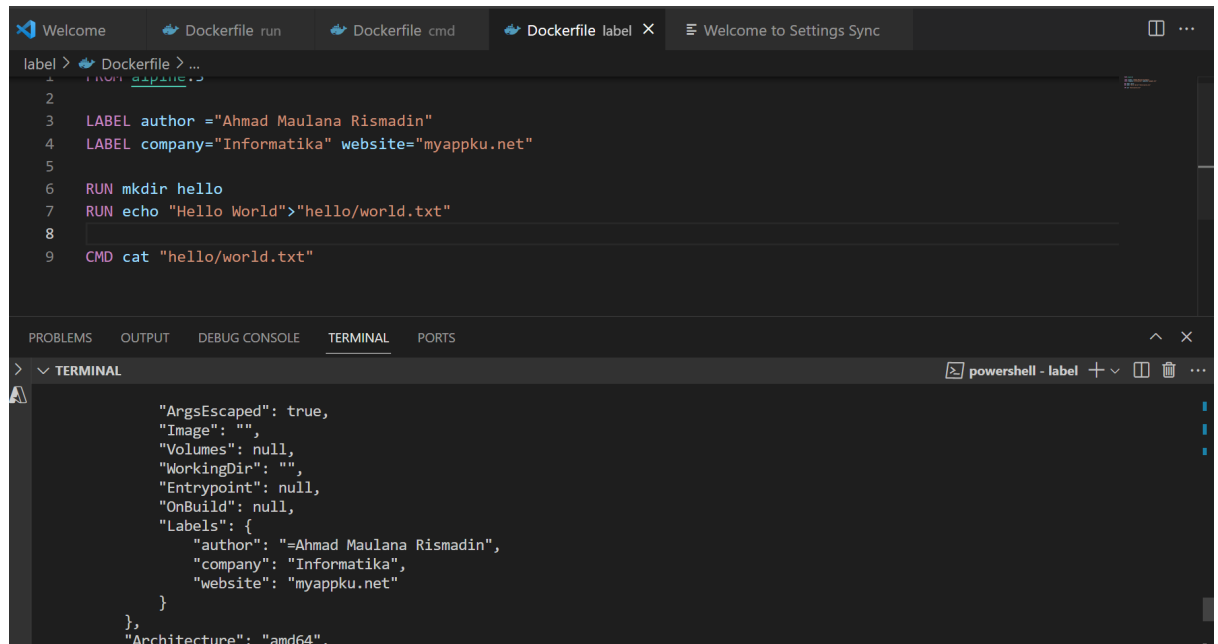
Bottom Screenshot: Container Logs

```
cmd > Dockerfile > CMD
1 FROM alpine:3
2
3 RUN mkdir hello
4 RUN echo "Hello World">"hello/world.txt"
5 CMD cat "hello/world.txt"
```

TERMINAL

```
PS D:\dockerfile\cmd> docker container logs cmd
Hello World
PS D:\dockerfile\cmd>
```

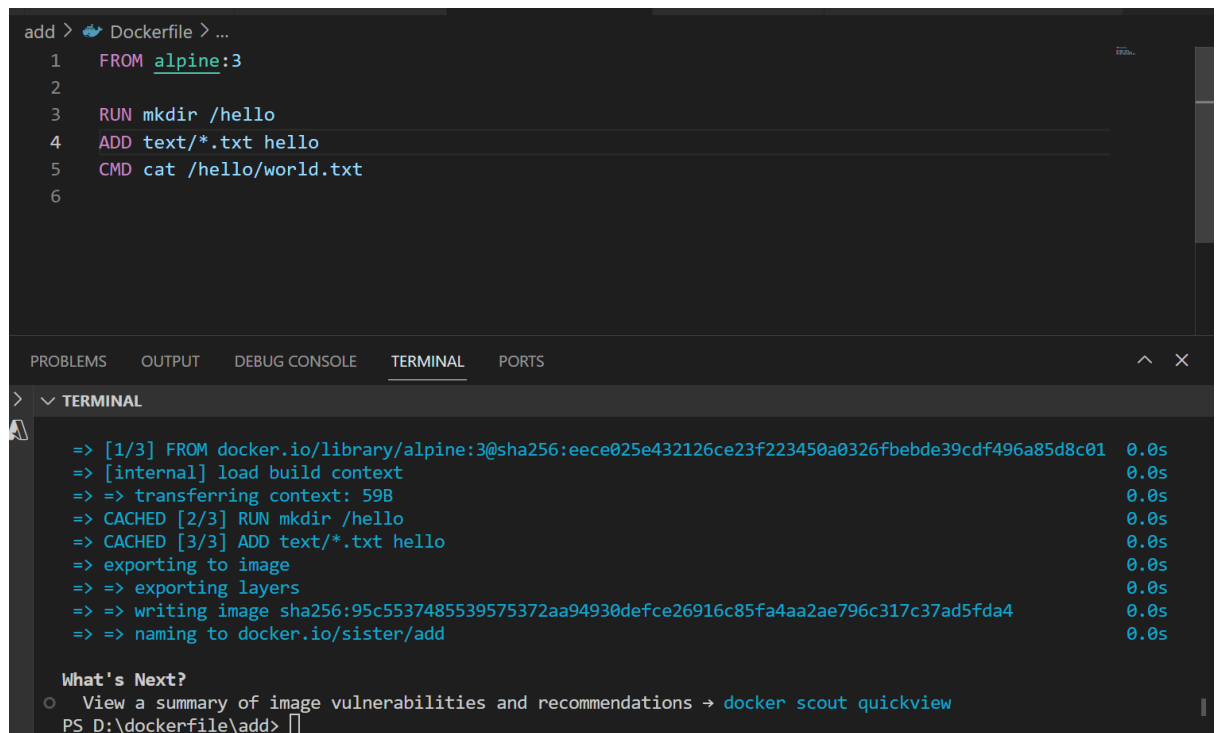
4. Label Instruction



```
label > Dockerfile > ...
1 FROM alpine:3
2
3 LABEL author="Ahmad Maulana Rismadin"
4 LABEL company="Informatika" website="myappku.net"
5
6 RUN mkdir hello
7 RUN echo "Hello World">"hello/world.txt"
8
9 CMD cat "hello/world.txt"
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
> v TERMINAL powershell - label + - X
{
  "ArgsEscaped": true,
  "Image": "",
  "Volumes": null,
  "WorkingDir": "",
  "Entrypoint": null,
  "OnBuild": null,
  "Labels": {
    "author": "Ahmad Maulana Rismadin",
    "company": "Informatika",
    "website": "myappku.net"
  },
  "Architecture": "amd64",
```

5. Add Instruction

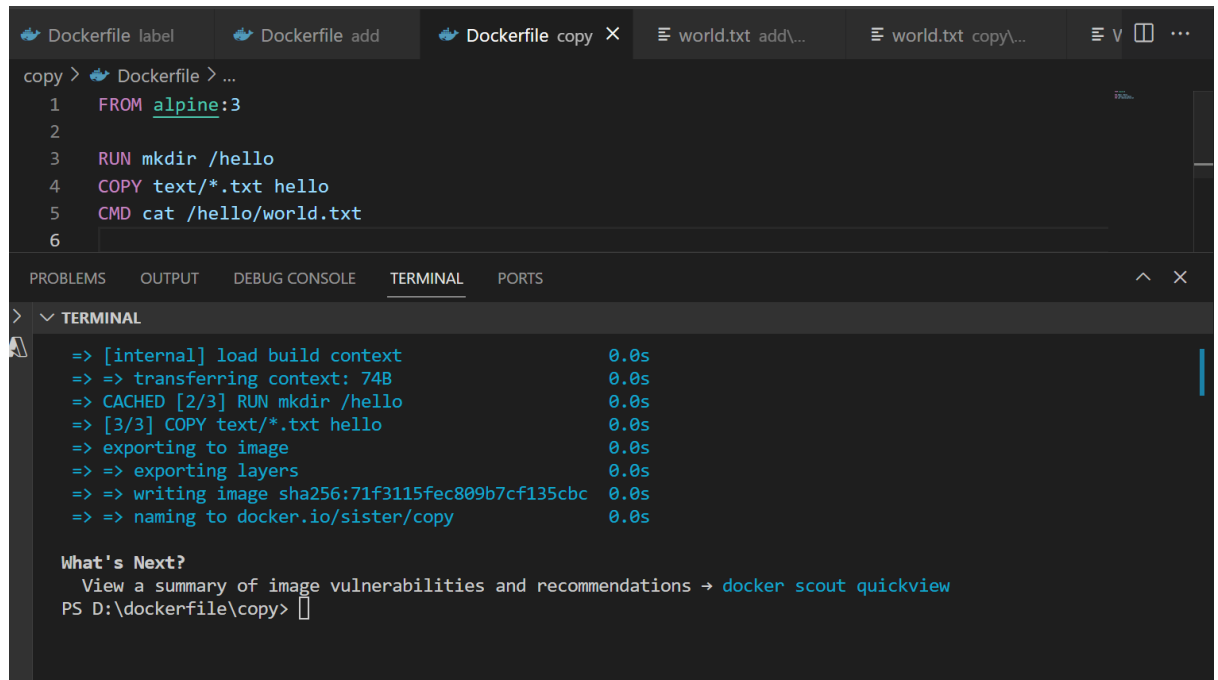


```
add > Dockerfile > ...
1 FROM alpine:3
2
3 RUN mkdir /hello
4 ADD text/*.txt hello
5 CMD cat /hello/world.txt
6
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
> v TERMINAL
=> [1/3] FROM docker.io/library/alpine:3@sha256:eece025e432126ce23f223450a0326fbebde39cdf496a85d8c01 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 59B 0.0s
=> CACHED [2/3] RUN mkdir /hello 0.0s
=> CACHED [3/3] ADD text/*.txt hello 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:95c5537485539575372aa94930defce26916c85fa4aa2ae796c317c37ad5fda4 0.0s
=> => naming to docker.io/sister/add 0.0s

What's Next?
o View a summary of image vulnerabilities and recommendations -> docker scout quickview
PS D:\dockerfile\add>
```

6. Copy Instruction



The screenshot shows the VS Code interface with a Dockerfile open. The Dockerfile contains the following instructions:

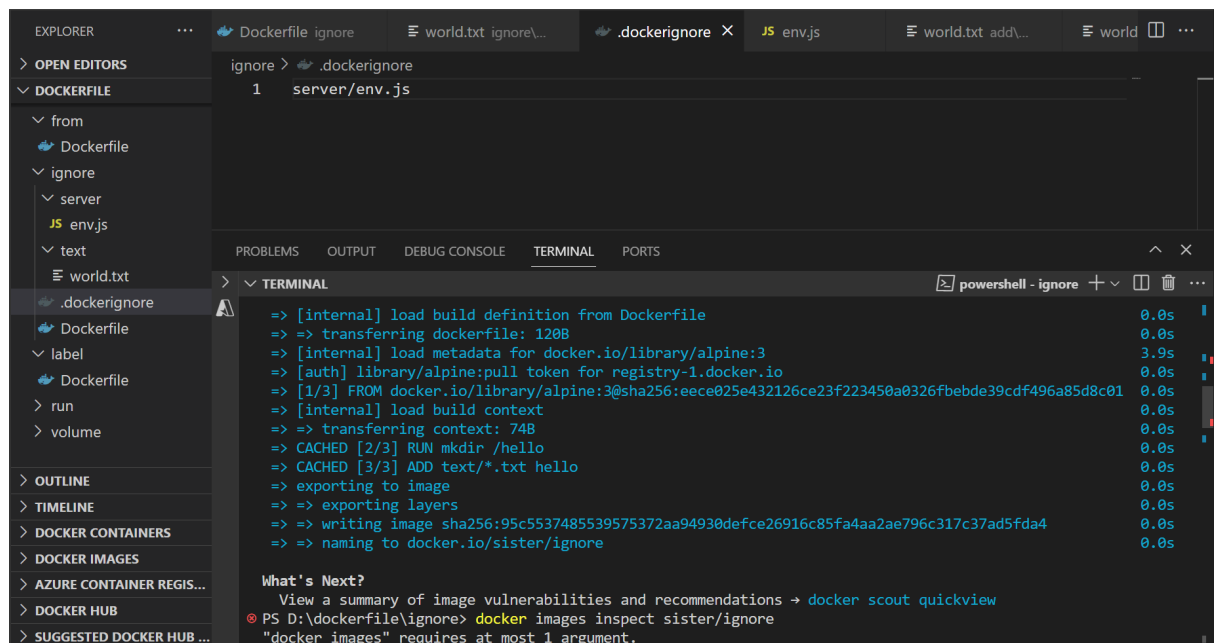
```
1 FROM alpine:3
2
3 RUN mkdir /hello
4 COPY text/*.txt hello
5 CMD cat /hello/world.txt
6
```

The terminal output shows the build process:

```
=> [internal] load build context 0.0s
=> => transferring context: 74B 0.0s
=> CACHED [2/3] RUN mkdir /hello 0.0s
=> [3/3] COPY text/*.txt hello 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:71f3115fec809b7cf135cbc 0.0s
=> => naming to docker.io/sister/copy 0.0s
```

Below the terminal output, there is a "What's Next?" section with a link to [docker scout quickview](#) and a PowerShell prompt: `PS D:\dockerfile\copy>`

7. .dockerignore File



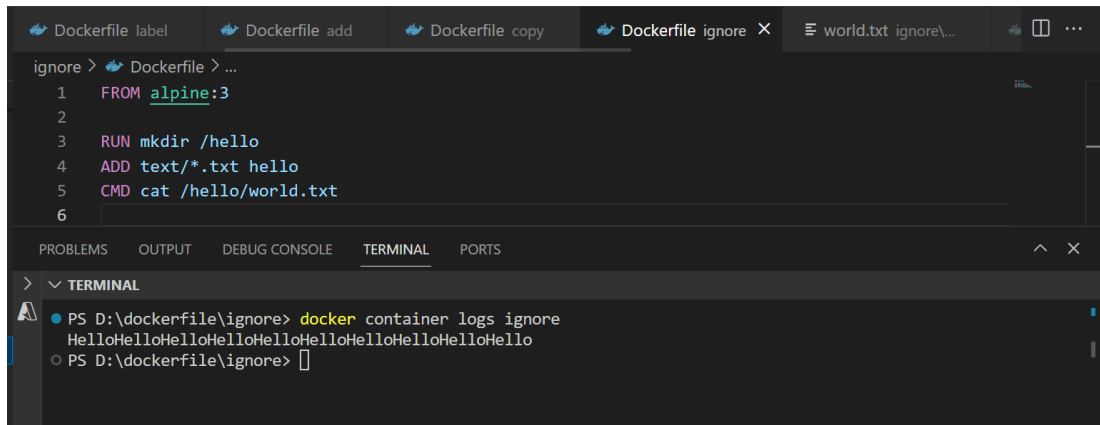
The screenshot shows the VS Code interface with a .dockerignore file open. The .dockerignore file contains the following content:

```
1 server/env.js
```

The terminal output shows the build process:

```
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 120B 0.0s
=> [internal] load metadata for docker.io/library/alpine:3 3.9s
=> [auth] library/alpine:pull token for registry-1.docker.io 0.0s
=> [1/3] FROM docker.io/library/alpine:3@sha256:eece025e432126ce23f223450a0326fbebde39cdf496a85d8c01 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 74B 0.0s
=> CACHED [2/3] RUN mkdir /hello 0.0s
=> CACHED [3/3] ADD text/*.txt hello 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:95c5537485539575372aa94930defce26916c85fa4aa2ae796c317c37ad5fda4 0.0s
=> => naming to docker.io/sister/ignore 0.0s
```

Below the terminal output, there is a "What's Next?" section with a link to [docker scout quickview](#) and a PowerShell prompt: `PS D:\dockerfile\ignore> docker images inspect sister/ignore`. A warning message is also displayed: `"docker images" requires at most 1 argument.`

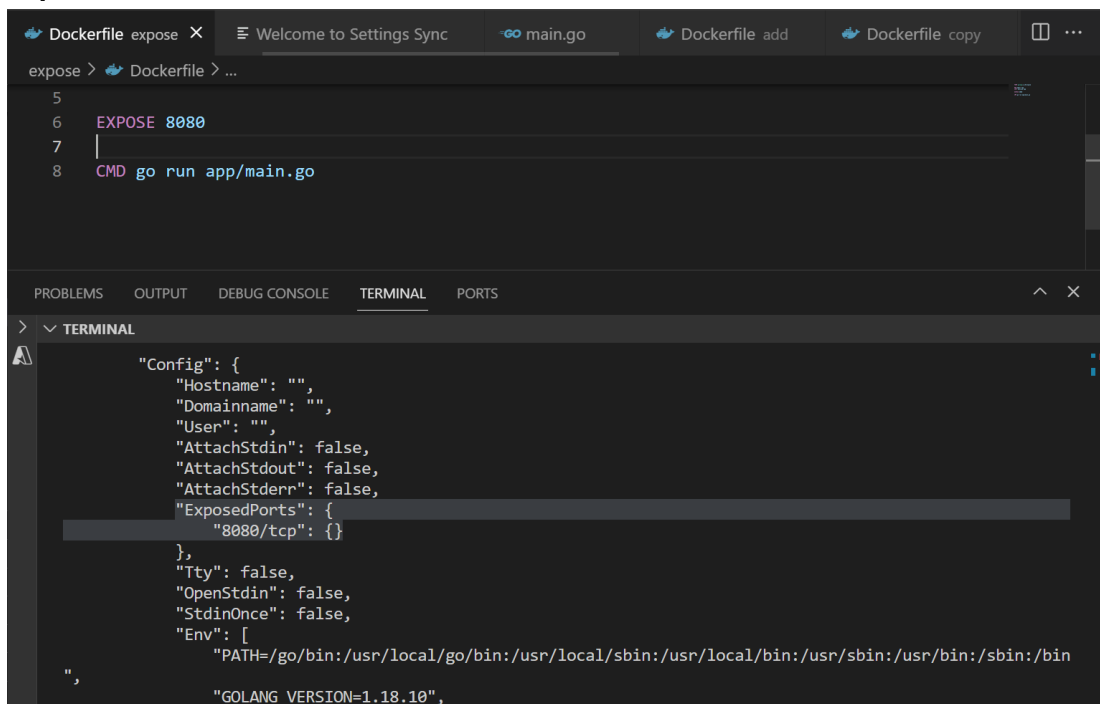


The screenshot shows a VS Code editor with a Dockerfile open. The Dockerfile contains the following instructions:

```
1 FROM alpine:3
2
3 RUN mkdir /hello
4 ADD text/*.txt hello
5 CMD cat /hello/world.txt
6
```

The terminal window at the bottom shows the command `docker container logs ignore` being executed, resulting in the output: `HelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHello`.

8. Expose Instruction

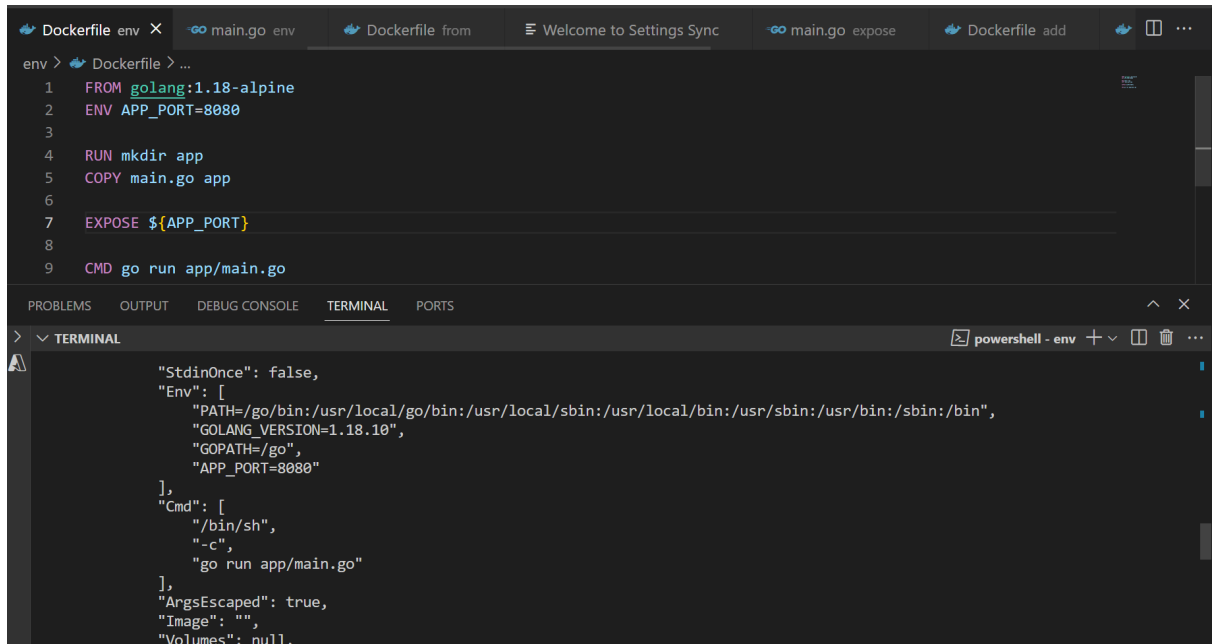


The screenshot shows a VS Code editor with a Dockerfile open. The Dockerfile contains the following instructions:

```
5
6 EXPOSE 8080
7
8 CMD go run app/main.go
```

The terminal window at the bottom shows the command `docker container logs ignore` being executed, resulting in the output: `Config: { "Hostname": "", "Domainname": "", "User": "", "AttachStdin": false, "AttachStdout": false, "AttachStderr": false, "ExposedPorts": { "8080/tcp": {} }, "Tty": false, "OpenStdin": false, "StdinOnce": false, "Env": ["PATH=/go/bin:/usr/local/go/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin", "GOLANG_VERSION=1.18.10",] }`

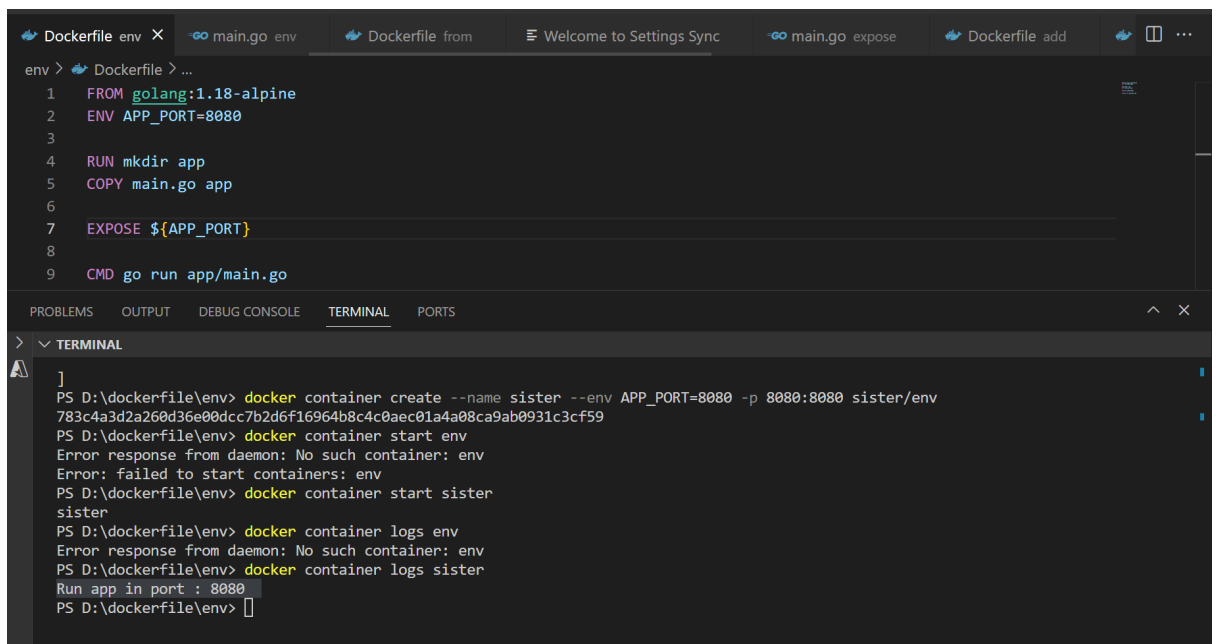
9. Environment Variable Instruction



The screenshot shows the VS Code editor with a Dockerfile open. The Dockerfile contains instructions to build a Go application container. Below the editor, the 'TERMINAL' tab displays the JSON representation of the container configuration generated by Docker.

```
env > Dockerfile > ...
1 FROM golang:1.18-alpine
2 ENV APP_PORT=8080
3
4 RUN mkdir app
5 COPY main.go app
6
7 EXPOSE ${APP_PORT}
8
9 CMD go run app/main.go
```

```
{
  "StdinOnce": false,
  "Env": [
    "PATH=/go/bin:/usr/local/go/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
    "GOLANG_VERSION=1.18.10",
    "GOPATH=/go",
    "APP_PORT=8080"
  ],
  "Cmd": [
    "/bin/sh",
    "-c",
    "go run app/main.go"
  ],
  "ArgsEscaped": true,
  "Image": "",
  "Volumes": null,
```

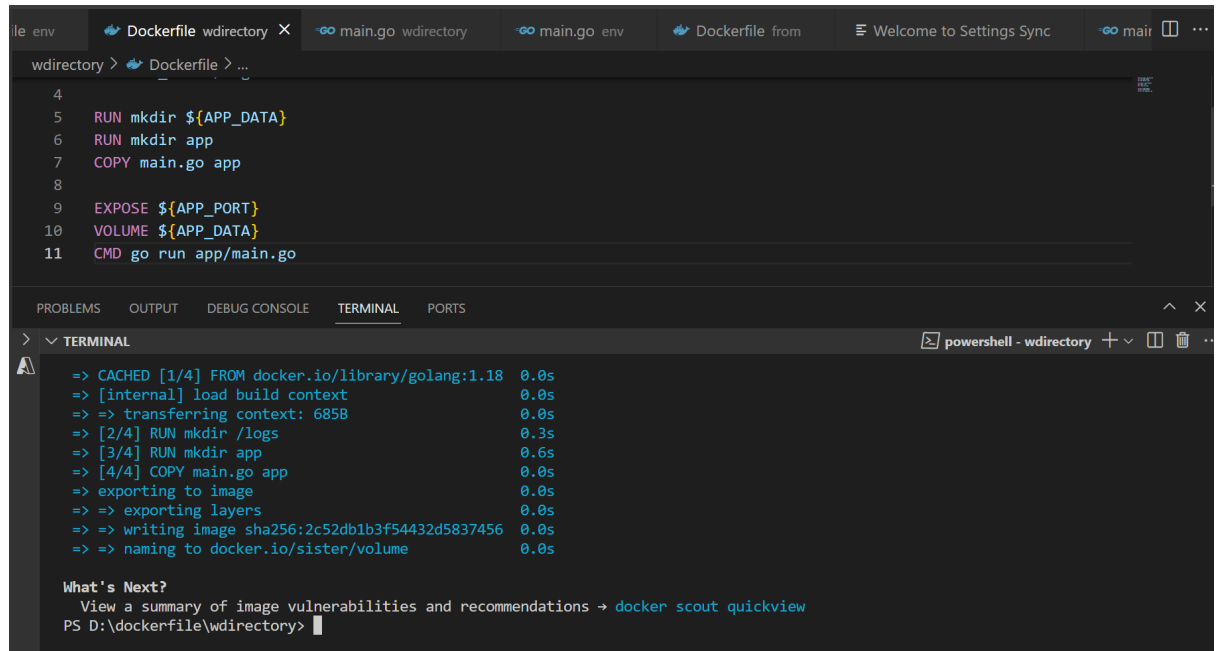


The screenshot shows the VS Code editor with the same Dockerfile. The 'TERMINAL' tab now displays the output of several Docker commands executed in a PowerShell terminal.

```
env > Dockerfile > ...
1 FROM golang:1.18-alpine
2 ENV APP_PORT=8080
3
4 RUN mkdir app
5 COPY main.go app
6
7 EXPOSE ${APP_PORT}
8
9 CMD go run app/main.go
```

```
PS D:\dockerfile\env> docker container create --name sister --env APP_PORT=8080 -p 8080:8080 sister/env
783c4a3d2a260d36e00dcc7b2d6f16964b8c4c0aec01a4a08ca9ab0931c3cf59
PS D:\dockerfile\env> docker container start env
Error response from daemon: No such container: env
Error: failed to start containers: env
PS D:\dockerfile\env> docker container start sister
sister
PS D:\dockerfile\env> docker container logs env
Error response from daemon: No such container: env
PS D:\dockerfile\env> docker container logs sister
Run app in port : 8080
PS D:\dockerfile\env>
```

10. Volume Instruction



The screenshot shows the VS Code interface with a Dockerfile editor and a terminal window. The Dockerfile contains instructions for building a Go application. The terminal displays the build progress, showing that the image is being built and named.

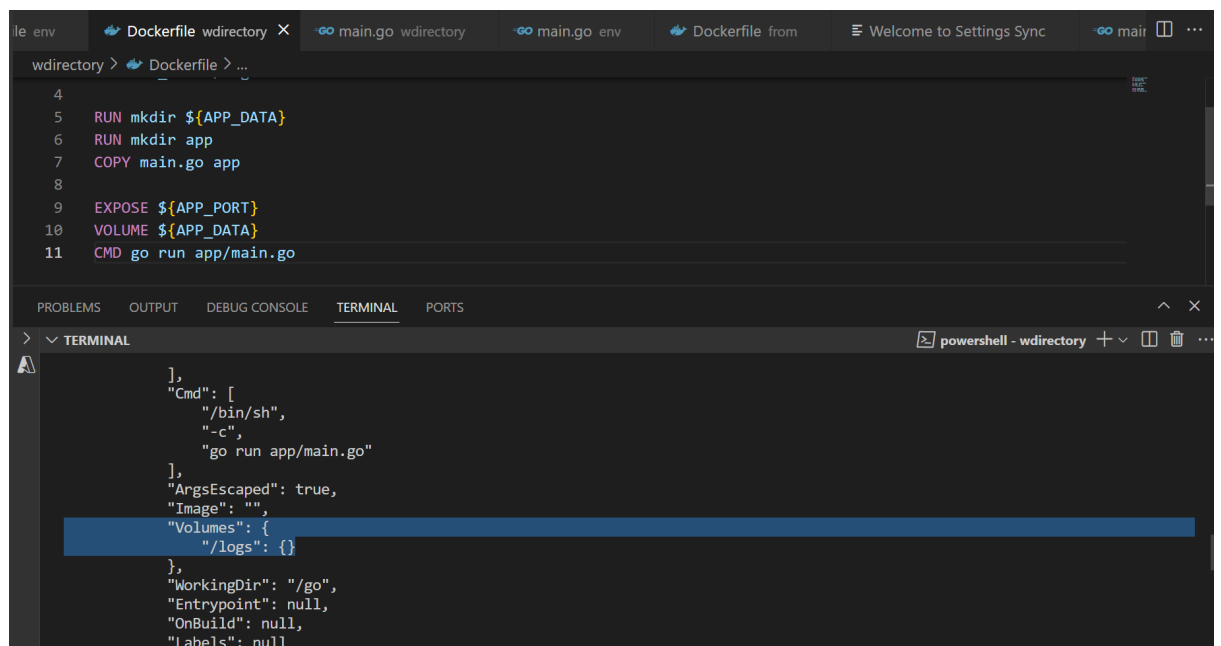
```
le env Dockerfile wdirectory X main.go wdirectory main.go env Dockerfile from Welcome to Settings Sync mail ...

wdirectory > Dockerfile > ...
4
5 RUN mkdir ${APP_DATA}
6 RUN mkdir app
7 COPY main.go app
8
9 EXPOSE ${APP_PORT}
10 VOLUME ${APP_DATA}
11 CMD go run app/main.go

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

> TERMINAL powershell - wdirectory + - -
=> CACHED [1/4] FROM docker.io/library/golang:1.18 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 685B 0.0s
=> [2/4] RUN mkdir /logs 0.3s
=> [3/4] RUN mkdir app 0.6s
=> [4/4] COPY main.go app 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:2c52db1b3f54432d5837456 0.0s
=> => naming to docker.io/sister/volume 0.0s

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS D:\dockerfile\wdirectory>
```



The screenshot shows the VS Code interface with a Dockerfile editor and a terminal window. The Dockerfile contains instructions for building a Go application. The terminal displays the build progress, showing that the image is being built and named. The 'Volumes' section of the Dockerfile is highlighted in the terminal output.

```
le env Dockerfile wdirectory X main.go wdirectory main.go env Dockerfile from Welcome to Settings Sync mail ...

wdirectory > Dockerfile > ...
4
5 RUN mkdir ${APP_DATA}
6 RUN mkdir app
7 COPY main.go app
8
9 EXPOSE ${APP_PORT}
10 VOLUME ${APP_DATA}
11 CMD go run app/main.go

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

> TERMINAL powershell - wdirectory + - -
],
"Cmd": [
  "/bin/sh",
  "-c",
  "go run app/main.go"
],
"ArgsEscaped": true,
"Image": "",
"Volumes": {
  "/logs": {}
},
"WorkingDir": "/go",
"Entrypoint": null,
"OnBuild": null,
"Labels": null
```

```
le env Dockerfile wdirectory X main.go wdirectory main.go env Dockerfile from Welcome to Settings Sync mair ...

wdirectory > Dockerfile > ...
7 COPY main.go app
8
9 EXPOSE ${APP_PORT}
10 VOLUME ${APP_DATA}
11 CMD go run app/main.go

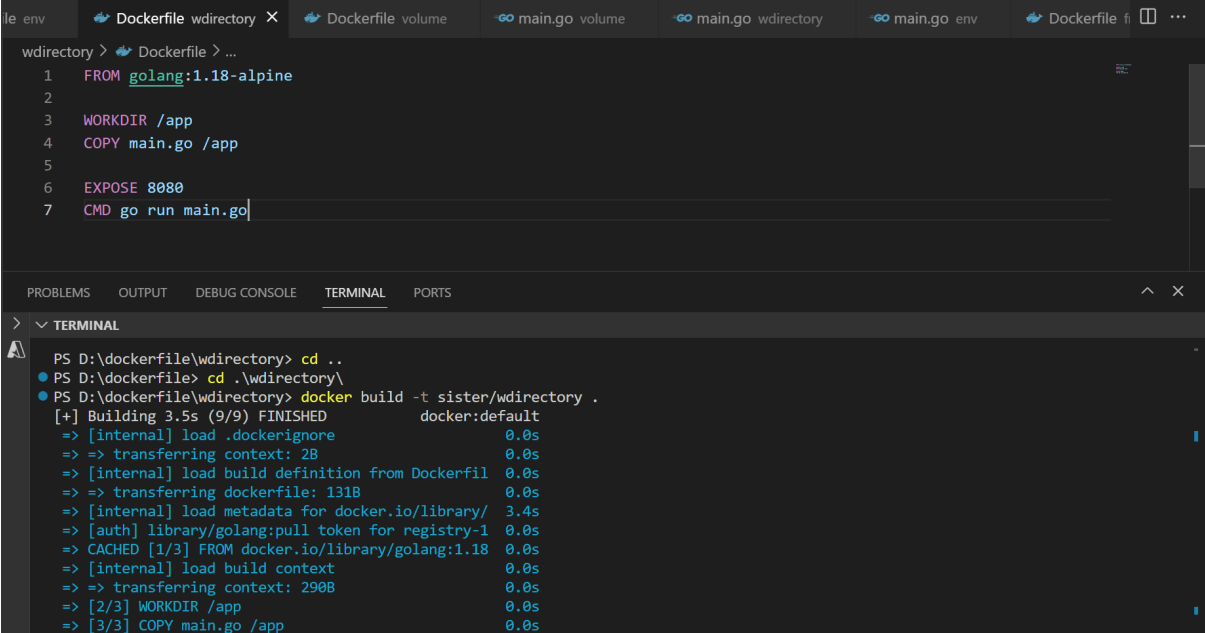
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
> v TERMINAL powershell - wdirectory + -
"UpperDir": "/var/lib/docker/overlay2/56b5d7f408be7e10671f7ce3f9c8318280821d7865d6a8fc235c0e93501c5c71/diff",
"WorkDir": "/var/lib/docker/overlay2/56b5d7f408be7e10671f7ce3f9c8318280821d7865d6a8fc235c0e93501c5c71/work"
},
"Name": "overlay2"
},
"Mounts": [
{
"Type": "volume",
"Name": "00619ab434848120dff60e343095f8fa8c3906ac4325c024ede32f83e4d3f082",
"Source": "/var/lib/docker/volumes/00619ab434848120dff60e343095f8fa8c3906ac4325c024ede32f83e4d3f082/_data",
"Destination": "/logs",
"Driver": "local",
"Mode": "",
"RW": true,
"Propagation": ""
}
```

```
le env Dockerfile wdirectory X main.go wdirectory main.go env Dockerfile from Welcome to Settings Sync mair ...

wdirectory > Dockerfile > ...
7 COPY main.go app
8
9 EXPOSE ${APP_PORT}
10 VOLUME ${APP_DATA}
11 CMD go run app/main.go

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
> v TERMINAL powershell - wdirectory + -
PS D:\dockerfile\wdirectory> docker volume ls
DRIVER VOLUME NAME
local 04ed2f1134375291eb280580b79c709488dbe58c788d71f59dc6e688ad8dce3
local 81ca10d37458b536a1212921ef01c90dcde8d94560c3d4c52f36dc73e1b4ae38
local 464d3a70bddeec309de655f3e1d836475bc7c2790422c0a2a0d7550b6209d8ad
local 00619ab434848120dff60e343095f8fa8c3906ac4325c024ede32f83e4d3f082
local 5610c3d011c9e4712965875c1bc2c7366fb8b9402a0b1af34404406191d1b6b9
local dbd0c56c4d24b2b945ab65a395d000fdb3a3d4ea2016a04adb7eac2d7e56b316
PS D:\dockerfile\wdirectory>
```


11. Working Directory Instruction

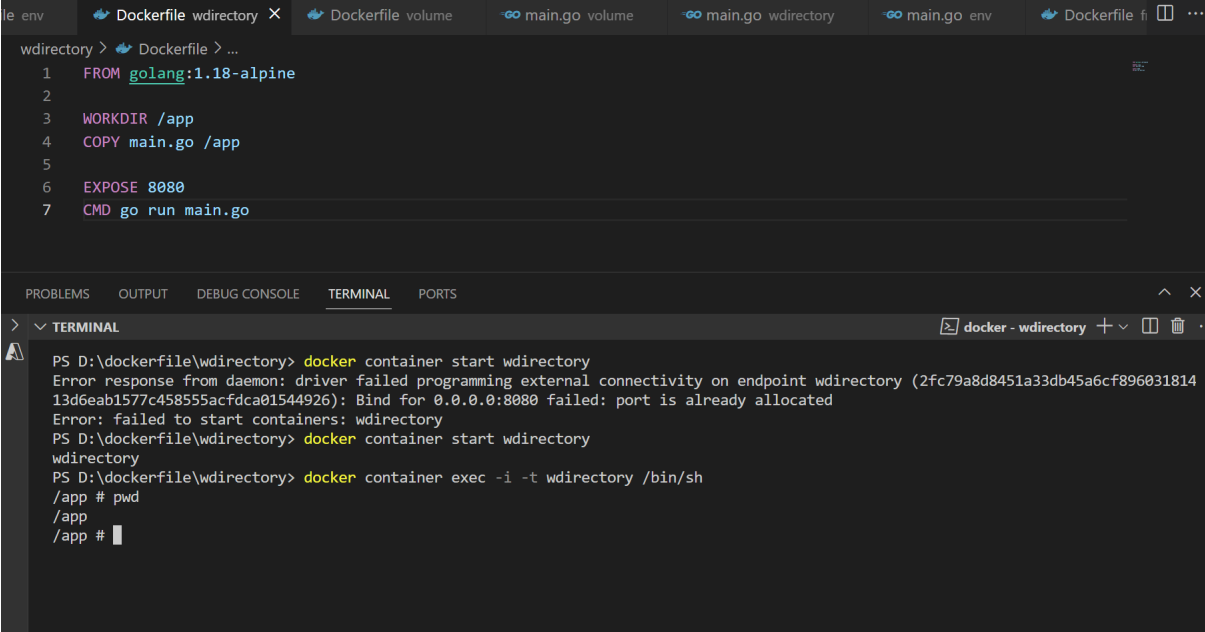


The screenshot shows the VS Code interface with a Dockerfile editor and a terminal. The Dockerfile contains the following instructions:

```
1 FROM golang:1.18-alpine
2
3 WORKDIR /app
4 COPY main.go /app
5
6 EXPOSE 8080
7 CMD go run main.go
```

The terminal shows the output of the `docker build` command:

```
PS D:\dockerfile\wdirectory> cd ..
PS D:\dockerfile> cd .\wdirectory\
PS D:\dockerfile\wdirectory> docker build -t sister/wdirectory .
[+] Building 3.5s (9/9) FINISHED      docker:default
=> [internal] load .dockerignore      0.0s
=> => transferring context: 2B        0.0s
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 131B   0.0s
=> [internal] load metadata for docker.io/library/ 3.4s
=> [auth] library/golang:pull token for registry-1 0.0s
=> CACHED [1/3] FROM docker.io/library/golang:1.18 0.0s
=> [internal] load build context      0.0s
=> => transferring context: 290B      0.0s
=> [2/3] WORKDIR /app                0.0s
=> [3/3] COPY main.go /app           0.0s
```

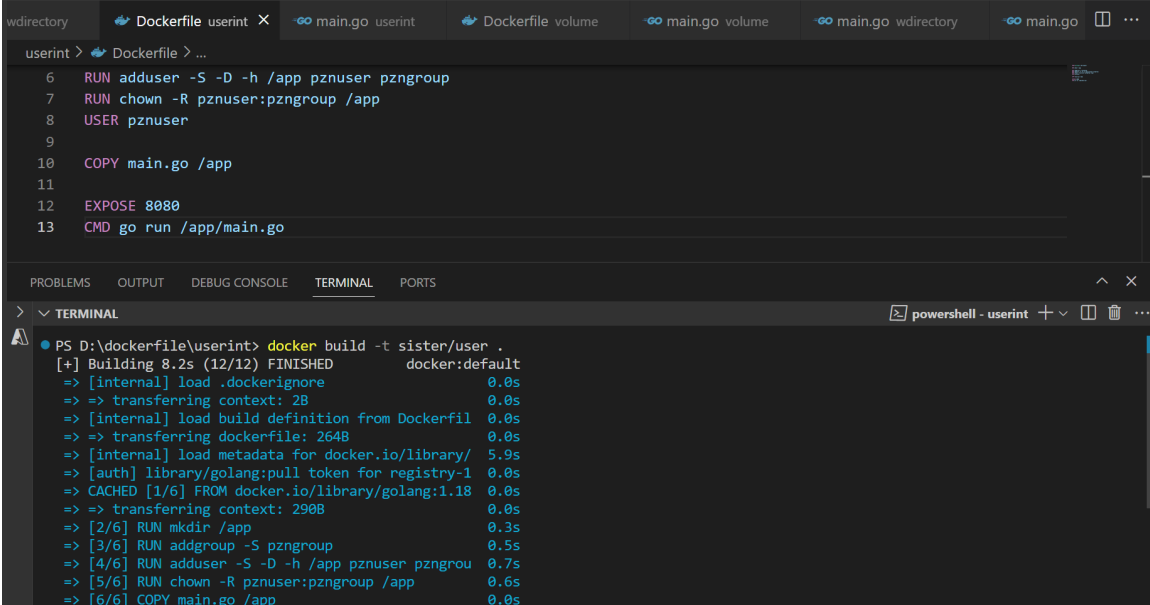


The screenshot shows the VS Code interface with the same Dockerfile editor and a terminal. The Dockerfile content is identical to the previous screenshot.

The terminal shows the output of the `docker container start` and `docker container exec` commands:

```
PS D:\dockerfile\wdirectory> docker container start wdirectory
Error response from daemon: driver failed programming external connectivity on endpoint wdirectory (2fc79a8d8451a33db45a6cf89603181413d6eab1577c458555acfdca01544926): Bind for 0.0.0.0:8080 failed: port is already allocated
Error: failed to start containers: wdirectory
PS D:\dockerfile\wdirectory> docker container start wdirectory
wdirectory
PS D:\dockerfile\wdirectory> docker container exec -i -t wdirectory /bin/sh
/app # pwd
/app
/app #
```

12. User Instruction

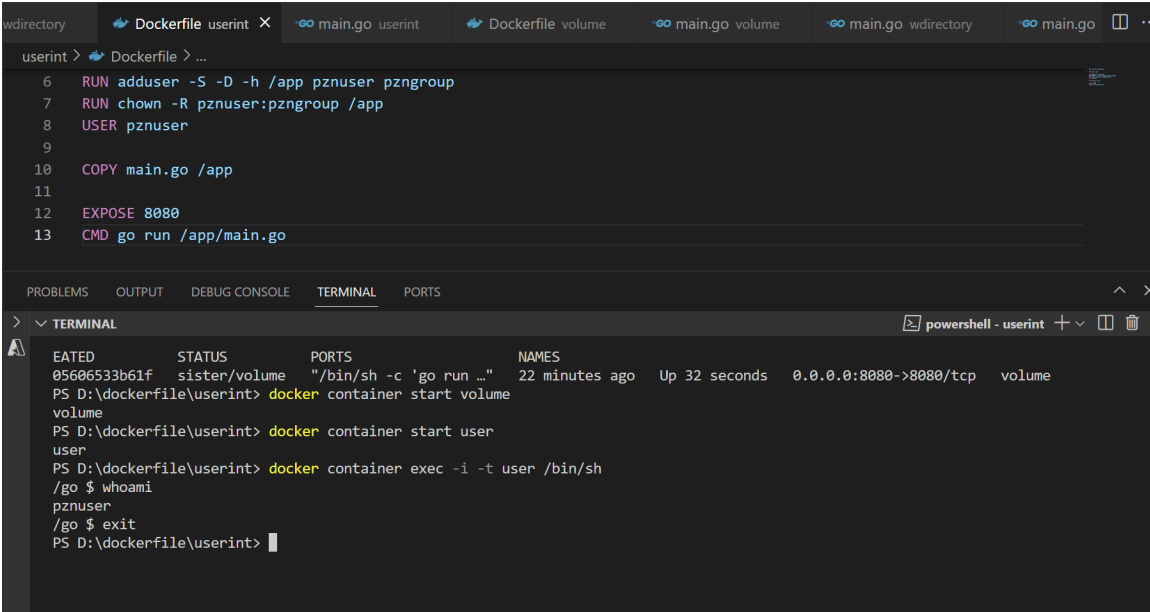


The screenshot shows the VS Code interface with a Dockerfile open in the editor. The Dockerfile contains the following instructions:

```
6 RUN adduser -S -D -h /app pznuser pzngroup
7 RUN chown -R pznuser:pzngroup /app
8 USER pznuser
9
10 COPY main.go /app
11
12 EXPOSE 8080
13 CMD go run /app/main.go
```

Below the editor, the TERMINAL tab is active, showing the output of the command `docker build -t sister/user .` in a PowerShell session:

```
PS D:\dockerfile\userint> docker build -t sister/user .
[+] Building 8.2s (12/12) FINISHED docker:default
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load build definition from Dockerfil 0.0s
=> => transferring dockerfile: 264B 0.0s
=> [internal] load metadata for docker.io/library/ 5.9s
=> [auth] library/golang:pull token for registry-1 0.0s
=> CACHED [1/6] FROM docker.io/library/golang:1.18 0.0s
=> => transferring context: 290B 0.0s
=> [2/6] RUN mkdir /app 0.3s
=> [3/6] RUN addgroup -S pzngroup 0.5s
=> [4/6] RUN adduser -S -D -h /app pznuser pznrou 0.7s
=> [5/6] RUN chown -R pznuser:pzngroup /app 0.6s
=> [6/6] COPY main.go /app 0.0s
```



The screenshot shows the VS Code interface with the same Dockerfile open. The TERMINAL tab is active, showing the output of the command `docker container start volume` in a PowerShell session:

```
PS D:\dockerfile\userint> docker container start volume
volume
PS D:\dockerfile\userint> docker container start user
user
PS D:\dockerfile\userint> docker container exec -i -t user /bin/sh
/go $ whoami
pznuser
/go $ exit
PS D:\dockerfile\userint>
```

Below the terminal output, a table shows the status of the containers:

EATED	STATUS	PORTS	NAMES
05606533b61f	sister/volume	"/bin/sh -c 'go run ...'"	22 minutes ago Up 32 seconds 0.0.0.0:8080->8080/tcp volume