Lab Exercise 5- Understanding CMD, RUN, and ENTRYPOINT in Dockerfile

Objective:

To learn the differences between CMD, RUN, and ENTRYPOINT instructions in Dockerfiles by creating and running Docker containers with different configurations.

Prerequisites:

- Docker installed on your machine
- Basic understanding of Docker and Dockerfile

Part 1: Overview of CMD, RUN, and ENTRYPOINT

- **RUN:** Executes commands at build time to install software, download dependencies, or configure the environment. The result is saved in the image.
- **CMD:** Specifies the default command to be executed when a container starts. It can be overridden when running a container.
- **ENTRYPOINT:** Defines the main executable for the container, which can't be easily overridden. However, additional arguments can be passed when the container starts.

Part 2: Exploring RUN Command

1. **Create a Dockerfile with RUN:**

Create a directory called dockerfile-run-cmd-entrypoint and navigate to it:

```
mkdir dockerfile-run-cmd-entrypoint && cd dockerfile-run-cmd-entrypoint
```

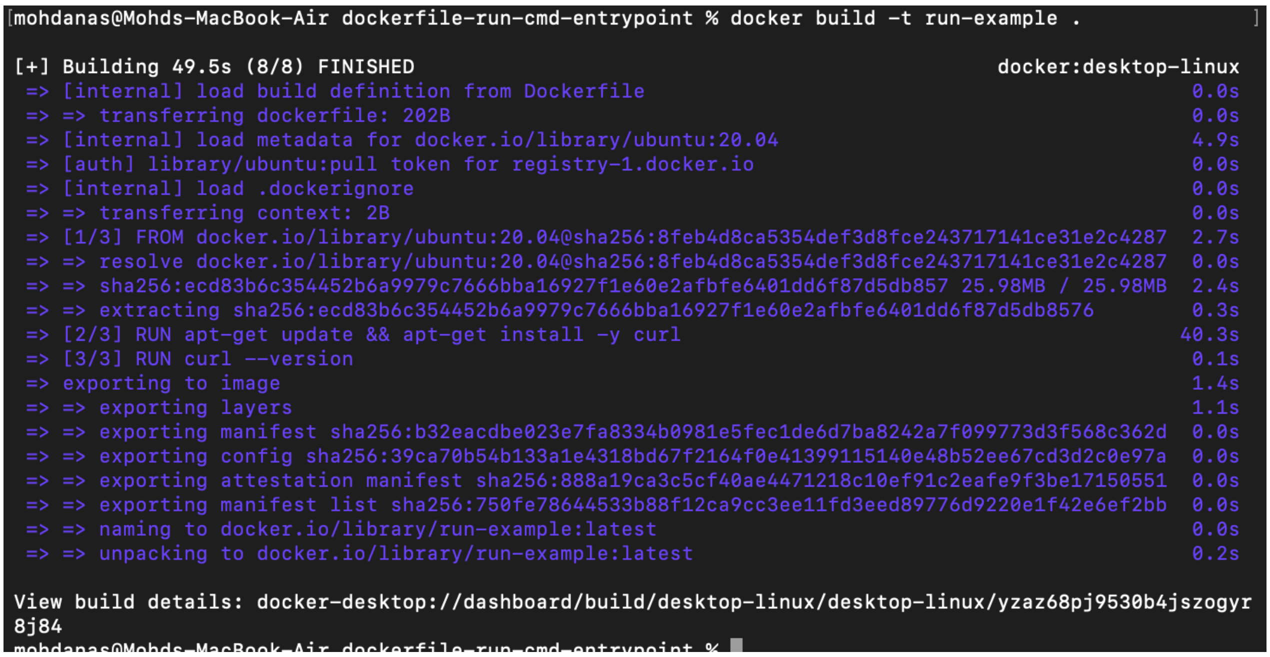
Create a simple Dockerfile that uses the RUN instruction:

```
# Use an official Ubuntu base image

FROM ubuntu:20.04



# Update the package repository and install curl

RUN apt-get update && apt-get install -y curl



# Print the version of curl

RUN curl --version
```

2. **Build the Docker Image:**

Build the image using the Dockerfile:

```
docker build -t run-example .
```



```
mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker build -t run-example .

[+] Building 49.5s (8/8) FINISHED                                           docker:desktop-linux
 => [internal] load build definition from Dockerfile                                        0.0s
 => => transferring dockerfile: 202B                                                        0.0s
 => [internal] load metadata for docker.io/library/ubuntu:20.04                             4.9s
 => [auth] library/ubuntu:pull token for registry-1.docker.io                               0.0s
 => [internal] load .dockerignore                                                           0.0s
 => => transferring context: 2B                                                             0.0s
 => [1/3] FROM docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c4287  2.7s
 => => resolve docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c4287  0.0s
 => => sha256:ecd83b6c354452b6a9979c7666bba16927f1e60e2afbfe6401dd6f87d5db857 25.98MB / 25.98MB  2.4s
 => => extracting sha256:ecd83b6c354452b6a9979c7666bba16927f1e60e2afbfe6401dd6f87d5db8576        0.3s
 => [2/3] RUN apt-get update && apt-get install -y curl                                    40.3s
 => [3/3] RUN curl --version                                                                0.1s
 => exporting to image                                                                      1.4s
 => => exporting layers                                                                     1.1s
 => => exporting manifest sha256:b32eacdbe023e7fa8334b0981e5fec1de6d7ba8242a7f099773d3f568c362d  0.0s
 => => exporting config sha256:39ca70b54b133a1e4318bd67f2164f0e41399115140e48b52ee67cd3d2c0e97a  0.0s
 => => exporting attestation manifest sha256:888a19ca3c5cf40ae4471218c10ef91c2eafe9f3be17150551  0.0s
 => => exporting manifest list sha256:750fe78644533b88f12ca9cc3ee11fd3eed89776d9220e1f42e6ef2bb  0.0s
 => => naming to docker.io/library/run-example:latest                                       0.0s
 => => unpacking to docker.io/library/run-example:latest                                    0.2s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/yzaz68pj9530b4jszogyr
8j84
mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint %
```

3. **Explanation:**

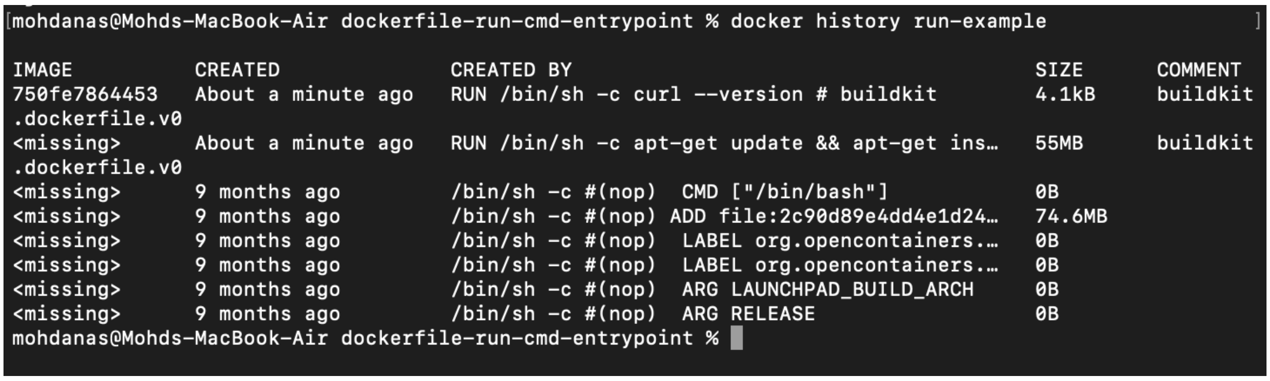
The RUN commands in this Dockerfile are executed during the image build process. The first RUN installs curl, and the second RUN command checks and prints the curl version. After the image is built, the commands executed by RUN are already baked into the image.

4. **Verify with Docker History:**

You can check the layers created by RUN using:

```
docker history run-example
```

Each RUN command creates a new layer in the image.



```
mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker history run-example

IMAGE          CREATED            CREATED BY                                      SIZE      COMMENT
750fe7864453   About a minute ago   RUN /bin/sh -c curl --version # buildkit     4.1kB     buildkit
.dockerfile.v0
<missing>      About a minute ago   RUN /bin/sh -c apt-get update && apt-get ins…   55MB      buildkit
.dockerfile.v0
<missing>      9 months ago       /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>      9 months ago       /bin/sh -c #(nop) ADD file:2c90d89e4dd4e1d24…   74.6MB
<missing>      9 months ago       /bin/sh -c #(nop)  LABEL org.opencontainers.…   0B
<missing>      9 months ago       /bin/sh -c #(nop)  LABEL org.opencontainers.…   0B
<missing>      9 months ago       /bin/sh -c #(nop)  ARG LAUNCHPAD_BUILD_ARCH     0B
<missing>      9 months ago       /bin/sh -c #(nop)  ARG RELEASE                  0B
mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint %
```

Part 3: Exploring CMD Command

1. **Create a Dockerfile with CMD:**

Modify the Dockerfile to include the CMD instruction:

```
# Use an official Ubuntu base image
FROM ubuntu:20.04

# Install curl
RUN apt-get update && apt-get install -y curl

# Set default command to display the curl version
CMD ["curl", "--version"]
```

2. **Build the Docker Image:**

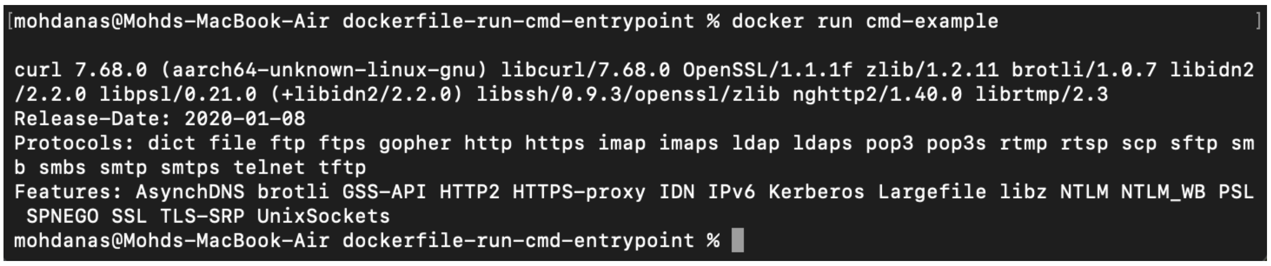
Build the Docker image again:

```
docker build -t cmd-example .
```

3. **Run the Container:**

Run the container and see the output:

```
docker run cmd-example
```



```
mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker run cmd-example

curl 7.68.0 (aarch64-unknown-linux-gnu) libcurl/7.68.0 OpenSSL/1.1.1f zlib/1.2.11 brotli/1.0.7 libidn2
/2.2.0 libpsl/0.21.0 (+libidn2/2.2.0) libssh/0.9.3/openssl/zlib nghttp2/1.40.0 librtmp/2.3
Release-Date: 2020-01-08
Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s rtmp rtsp scp sftp sm
b smbs smtp smtps telnet tftp
Features: AsynchDNS brotli GSS-API HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM NTLM_WB PSL
 SPNEGO SSL TLS-SRP UnixSockets
mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint %
```

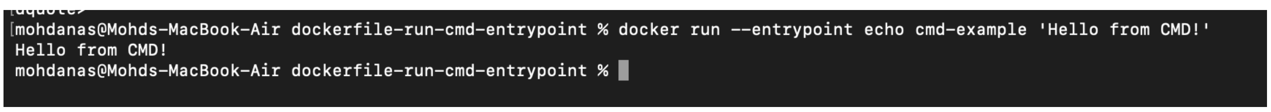
The output will display the curl version as the default command defined by CMD is executed when the container starts.

4. **Override CMD:**

You can override the CMD by specifying a different command when you run the container:

```
docker run cmd-example echo "Hello from CMD!"
```

This will print Hello from CMD!, showing that the CMD can be overridden at runtime.



```
[mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker run --entrypoint echo cmd-example 'Hello from CMD!'
Hello from CMD!
mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint %
```

Part 4: Exploring ENTRYPOINT Command

1. **Create a Dockerfile with ENTRYPOINT:**

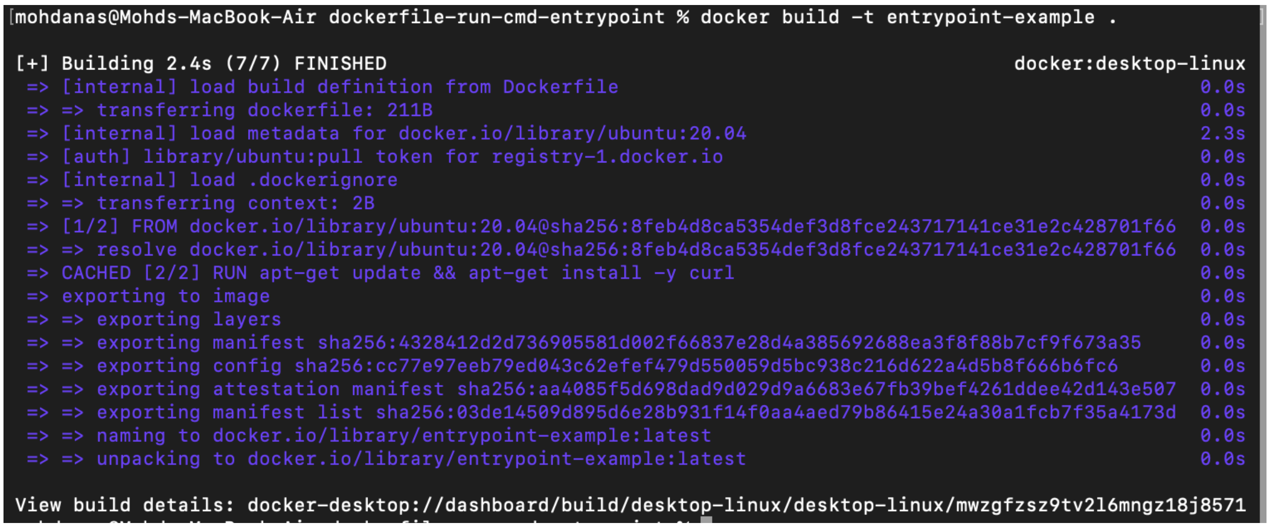
Modify the Dockerfile to use ENTRYPOINT instead of CMD:

```
# Use an official Ubuntu base image
FROM ubuntu:20.04

# Install curl
RUN apt-get update && apt-get install -y curl

# Set entrypoint to curl command
ENTRYPOINT ["curl"]
```

2. **Build the Docker Image:**

Build the image with the ENTRYPOINT instruction:

```
docker build -t entrypoint-example .
```

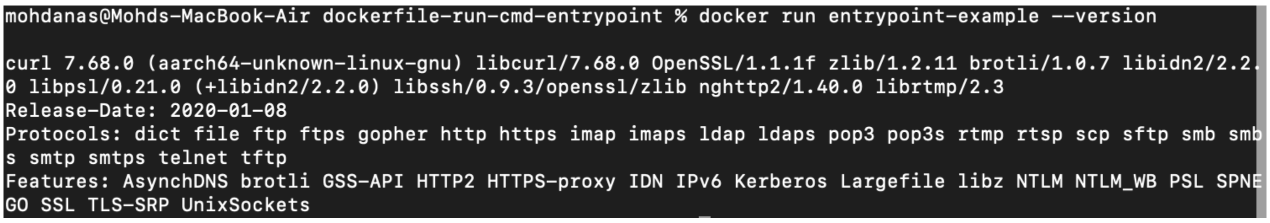


```
mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker build -t entrypoint-example .

[+] Building 2.4s (7/7) FINISHED                                                      docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                  0.0s
 => => transferring dockerfile: 211B                                                                  0.0s
 => [internal] load metadata for docker.io/library/ubuntu:20.04                                       2.3s
 => [auth] library/ubuntu:pull token for registry-1.docker.io                                         0.0s
 => [internal] load .dockerignore                                                                     0.0s
 => => transferring context: 2B                                                                       0.0s
 => [1/2] FROM docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c428701f66  0.0s
 => => resolve docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c428701f66  0.0s
 => CACHED [2/2] RUN apt-get update && apt-get install -y curl                                        0.0s
 => exporting to image                                                                                0.0s
 => => exporting layers                                                                               0.0s
 => => exporting manifest sha256:4328412d2d736905581d002f66837e28d4a385692688ea3f8f88b7cf9f673a35     0.0s
 => => exporting config sha256:cc77e97eeb79ed043c62efef479d550059d5bc938c216d622a4d5b8f666b6fc6       0.0s
 => => exporting attestation manifest sha256:aa4085f5d698dad9d029d9a6683e67fb39bef4261ddee42d143e507  0.0s
 => => exporting manifest list sha256:03de14509d895d6e28b931f14f0aa4aed79b86415e24a30a1fcb7f35a4173d  0.0s
 => => naming to docker.io/library/entrypoint-example:latest                                          0.0s
 => => unpacking to docker.io/library/entrypoint-example:latest                                       0.0s

 View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/mwzgfzsz9tv2l6mngz18j8571
```

3. **Run the Container:**

When you run the container, since ENTRYPOINT is set to curl, you need to provide arguments to the curl command:

```
docker run entrypoint-example --version
```



```
mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker run entrypoint-example --version

curl 7.68.0 (aarch64-unknown-linux-gnu) libcurl/7.68.0 OpenSSL/1.1.1f zlib/1.2.11 brotli/1.0.7 libidn2/2.2.
0 libpsl/0.21.0 (+libidn2/2.2.0) libssh/0.9.3/openssl/zlib nghttp2/1.40.0 librtmp/2.3
Release-Date: 2020-01-08
Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s rtmp rtsp scp sftp smb smb
s smtp smtps telnet tftp
Features: AsynchDNS brotli GSS-API HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM NTLM_WB PSL SPNE
GO SSL TLS-SRP UnixSockets
```

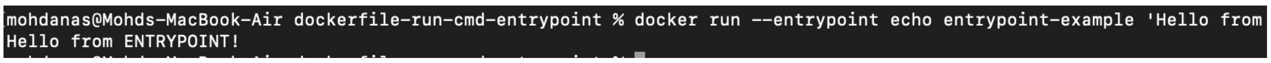
This will print the curl version because ENTRYPOINT defines the main executable (in this case, curl) and --version is passed as an argument to curl.

4. **Override ENTRYPOINT:**

Unlike CMD, the ENTRYPOINT is not easily overridden. If you try to override it using:

```
docker run entrypoint-example echo ‘Hello from ENTRYPOINT!’
```

It will result in an error because curl will interpret echo as an argument.

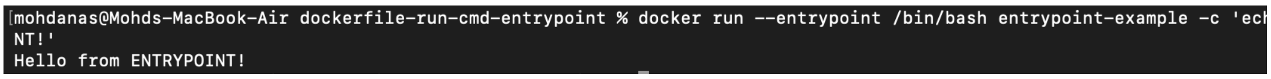


```
mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker run --entrypoint echo entrypoint-example 'Hello from
Hello from ENTRYPOINT!
```

However, you can use the --entrypoint option to change the entrypoint:

```
docker run --entrypoint /bin/bash entrypoint-example -c "echo Hello from ENTRYPOINT!"
```

This runs the container with /bin/bash as the entrypoint, overriding the default ENTRYPOINT.



```
[mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker run --entrypoint /bin/bash entrypoint-example -c 'ech
NT!'
Hello from ENTRYPOINT!
```

Part 5: Combining CMD and ENTRYPOINT

1. **Create a Dockerfile with Both CMD and ENTRYPOINT:**

Modify the Dockerfile to use both CMD and ENTRYPOINT:

```
# Use an official Ubuntu base image
FROM ubuntu:20.04

# Install curl
RUN apt-get update && apt-get install -y curl

# Set entrypoint to curl
ENTRYPOINT ["curl"]

# Set default arguments to --version
```

```
CMD ["--version"]
```

2. **Build the Image:**

Build the new image:

```
docker build -t combined-example .
```

3. **Run the Container:**

When you run the container without specifying any arguments, it will use the CMD as arguments to ENTRYPOINT:

```
docker run combined-example
```

The output will show the curl version, as ENTRYPOINT is curl and CMD provides --version as the argument.

4. **Override CMD Arguments:**

You can override the CMD arguments by specifying your own arguments:

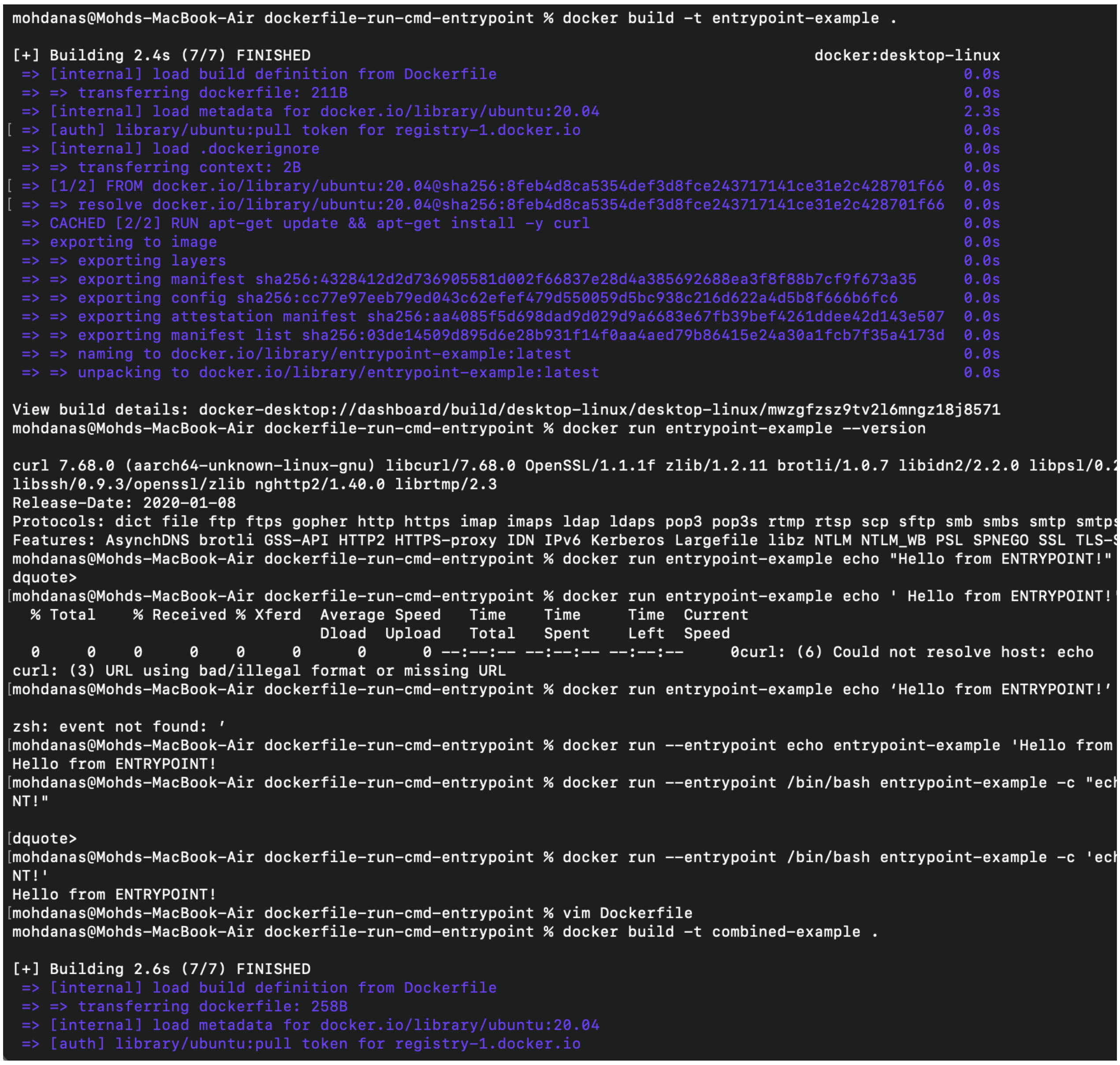
```
docker run combined-example https://www.google.com
```

This command will run curl https://www.google.com inside the container.

Summary of Differences:

- **RUN:** Executes commands during the image build process and creates layers. It is used to install packages and configure the environment.
- **CMD:** Specifies the default command to run when the container starts. It can be overridden by passing a different command when running the container.

- **ENTRYPOINT:** Specifies the main command for the container. It is harder to override but allows passing arguments from the command line. When combined with CMD, CMD provides the default arguments for ENTRYPOINT.



```
mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker build -t entrypoint-example .

[+] Building 2.4s (7/7) FINISHED                                                 docker:desktop-linux
 => [internal] load build definition from Dockerfile                                             0.0s
 => => transferring dockerfile: 211B                                                             0.0s
 => [internal] load metadata for docker.io/library/ubuntu:20.04                                  2.3s
[ => [auth] library/ubuntu:pull token for registry-1.docker.io                                   0.0s
 => [internal] load .dockerignore                                                                0.0s
 => => transferring context: 2B                                                                  0.0s
[ => [1/2] FROM docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c428701f66  0.0s
[ => => resolve docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c428701f66  0.0s
 => CACHED [2/2] RUN apt-get update && apt-get install -y curl                                   0.0s
 => exporting to image                                                                           0.0s
 => => exporting layers                                                                          0.0s
 => => exporting manifest sha256:4328412d2d736905581d002f66837e28d4a385692688ea3f8f88b7cf9f673a35     0.0s
 => => exporting config sha256:cc77e97eeb79ed043c62efef479d550059d5bc938c216d622a4d5b8f666b6fc6       0.0s
 => => exporting attestation manifest sha256:aa4085f5d698dad9d029d9a6683e67fb39bef4261ddee42d143e507  0.0s
 => => exporting manifest list sha256:03de14509d895d6e28b931f14f0aa4aed79b86415e24a30a1fcb7f35a4173d  0.0s
 => => naming to docker.io/library/entrypoint-example:latest                                     0.0s
 => => unpacking to docker.io/library/entrypoint-example:latest                                  0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/mwzgfzsz9tv2l6mngz18j8571
mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker run entrypoint-example --version

curl 7.68.0 (aarch64-unknown-linux-gnu) libcurl/7.68.0 OpenSSL/1.1.1f zlib/1.2.11 brotli/1.0.7 libidn2/2.2.0 libpsl/0.2
libssh/0.9.3/openssl/zlib nghttp2/1.40.0 librtmp/2.3
Release-Date: 2020-01-08
Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s rtmp rtsp scp sftp smb smbs smtp smtps
Features: AsynchDNS brotli GSS-API HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM NTLM_WB PSL SPNEGO SSL TLS-S
mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker run entrypoint-example echo "Hello from ENTRYPOINT!"
dquote>
[mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker run entrypoint-example echo ' Hello from ENTRYPOINT!'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0curl: (6) Could not resolve host: echo
curl: (3) URL using bad/illegal format or missing URL
[mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker run entrypoint-example echo ‘Hello from ENTRYPOINT!’

zsh: event not found: ’
[mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker run --entrypoint echo entrypoint-example 'Hello from
Hello from ENTRYPOINT!
[mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker run --entrypoint /bin/bash entrypoint-example -c "ech
NT!"

[dquote>
[mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker run --entrypoint /bin/bash entrypoint-example -c 'ech
NT!'
Hello from ENTRYPOINT!
[mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % vim Dockerfile
mohdanas@Mohds-MacBook-Air dockerfile-run-cmd-entrypoint % docker build -t combined-example .

[+] Building 2.6s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 258B
 => [internal] load metadata for docker.io/library/ubuntu:20.04
 => [auth] library/ubuntu:pull token for registry-1.docker.io
```

Conclusion:

This lab exercise demonstrates the fundamental differences between RUN, CMD, and ENTRYPOINT in Docker. Each command serves a different purpose, from image build-time configuration (RUN) to defining the container’s behavior at runtime (CMD and

ENTRYPOINT). Understanding these differences is crucial for building effective and flexible Docker images.