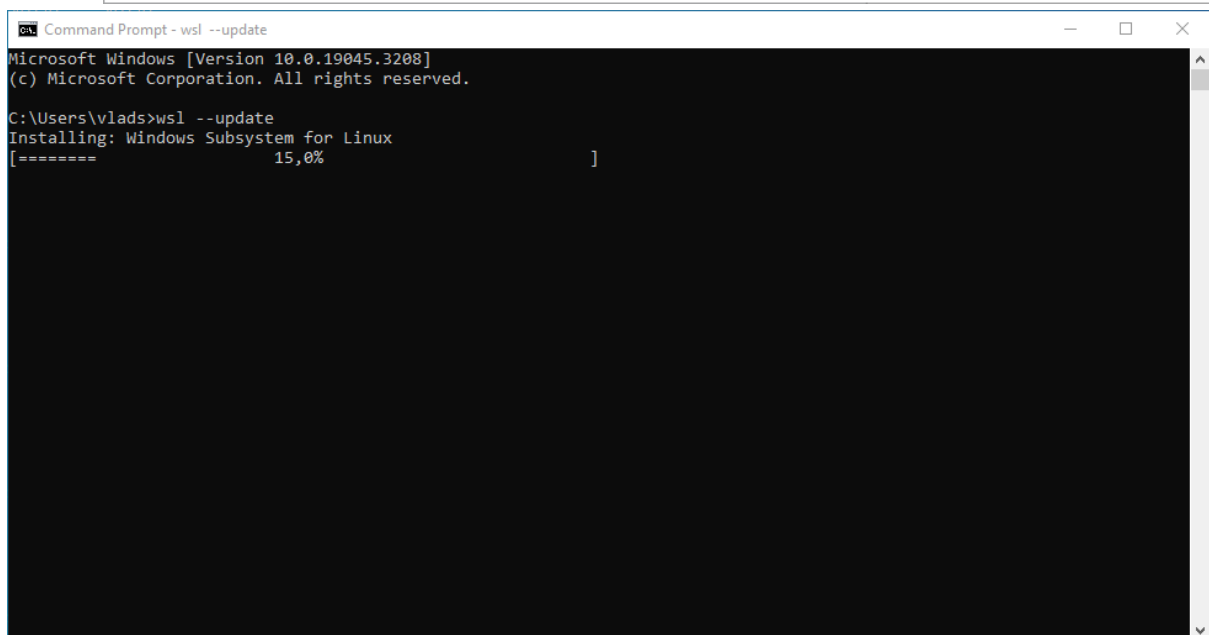
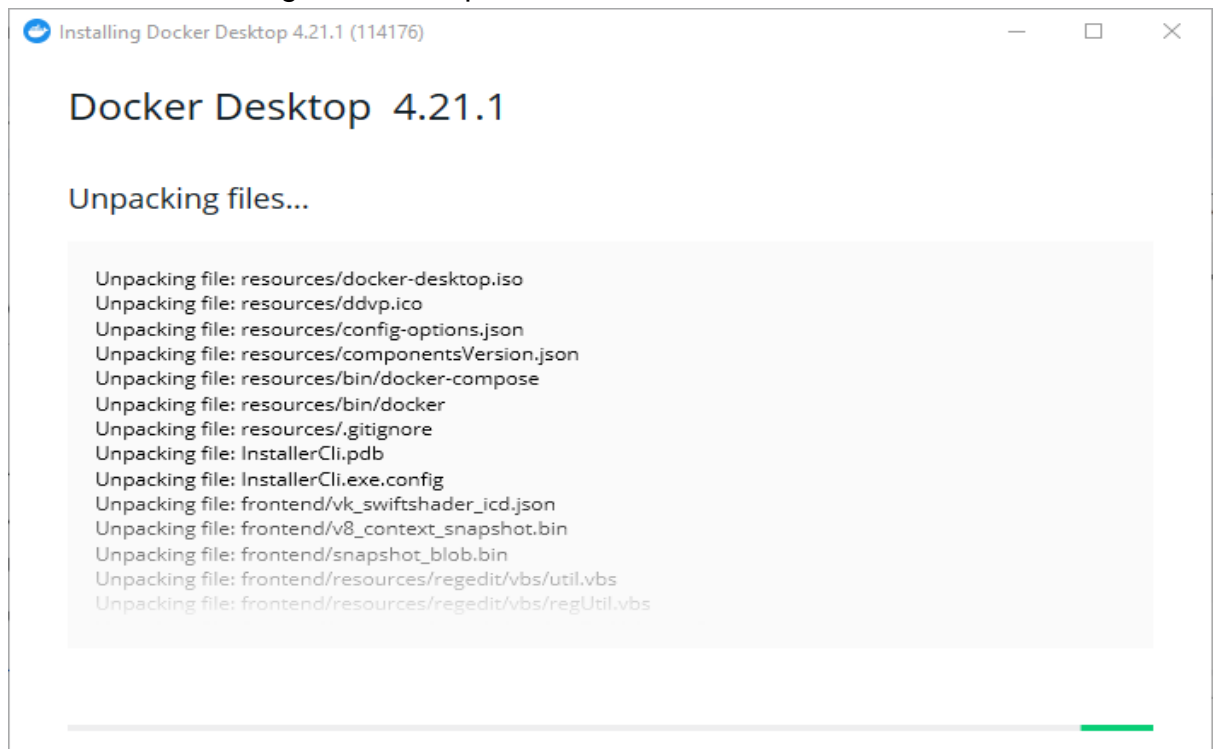


I will introduce the key concepts of Docker Compose whilst building a simple Python web application. I installed the Docker Desktop which includes both Docker Engine and Docker Compose. From July 2023 Compose V1 stopped receiving updates so I needed to migrate to Compose V2.



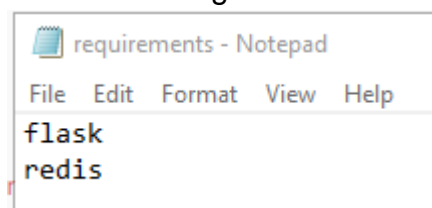
- I created a directory for the project:

```
C:\Users\vlads>mkdir composetest  
C:\Users\vlads>cd composetest
```

- I created a file called `app.py` in my project directory and put the following code in:

```
C: > Users > vlads > composetest > app.py
1  import time
2
3  import redis
4  from flask import Flask
5
6  app = Flask(__name__)
7  cache = redis.Redis(host='redis', port=6379)
8
9  def get_hit_count():
10     retries = 5
11     while True:
12         try:
13             return cache.incr('hits')
14         except redis.exceptions.ConnectionError as exc:
15             if retries == 0:
16                 raise exc
17             retries -= 1
18             time.sleep(0.5)
19
20 @app.route('/')
21 def hello():
22     count = get_hit_count()
23     return 'Hello World! I have been seen {} times.\n'.format(count)
```

- In this example, `redis` is the hostname of the redis container on the application's network. I used the default port for Redis, `6379`.
- I created another file called `requirements.txt` in my project directory and put the following code in:



```
requirements - Notepad
File Edit Format View Help
flask
redis
```

- The Dockerfile is used to build a Docker image. The image contains all the dependencies the Python application requires, including Python itself.
- In my project directory, I created a file named `Dockerfile` and inserted the following code in:

```

C: > Users > vlads > composetest > Dockerfile
1  # syntax=docker/dockerfile:1
2  FROM python:3.7-alpine
3  WORKDIR /code
4  ENV FLASK_APP=app.py
5  ENV FLASK_RUN_HOST=0.0.0.0
6  RUN apk add --no-cache gcc musl-dev linux-headers
7  COPY requirements.txt requirements.txt
8  RUN pip install -r requirements.txt
9  EXPOSE 5000
10 COPY . .
11 CMD ["flask", "run"]

```

This tells Docker to:

- Build an image starting with the Python 3.7 image.
- Set the working directory to `/code`.
- Set environment variables used by the `flask` command.
- Install `gcc` and other dependencies
- Copy `requirements.txt` and install the Python dependencies.
- Add metadata to the image to describe that the container is listening on port 5000
- Copy the current directory `.` in the project to the `workdir` `.` in the image.
- Set the default command for the container to `flask run`.

! `Dockerfile` has no file extension like `.txt`

- I created a file called `docker-compose.yml` in my project directory and inserted the following:

```

C: > Users > vlads > composetest > docker-compose.yml
1  services:
2    web:
3      build: .
4      ports:
5        - "8000:5000"
6    redis:
7      image: "redis:alpine"

```

This Compose file defines two services: `web` and `redis`. The `web` service uses an image that's built from the `Dockerfile` in the current directory. It then binds the container and the host machine to the exposed port, `8000`. This example service uses the default port for the Flask web server, `5000`. The `redis` service uses a public `Redis` image pulled from the Docker Hub registry.

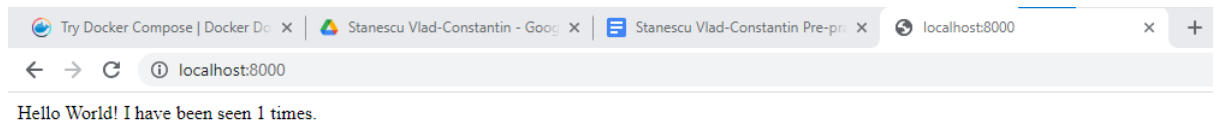
- From my project directory, I started up my application by running `docker compose up`.

```
C:\Users\vlad>composetest\docker compose up
[+] Building 0s / 7.1s
# redis 6 layers [cached] 88/88 Pulled
# 31e327af5f3 Pull complete      1.4s
# 029a81f9585 Pull complete      1.5s
# 7aa60a97d81 Pull complete      2.0s
# 2bfed931134 Pull complete      2.5s
# 952ba9c1ebd Pull complete      2.5s
# d909cbefed9 Pull complete      2.6s
[+] Building 27.2s (13/13) FINISHED
[web internal] load_dockerignore
=> transferring context: 2B
[web internal] load build definition from Dockerfile
=> transferring dockerfile: 309B
[web] resolve image config for docker.io/docker/dockerfile1
[web] docker-image://docker.io/docker/dockerfile1:gsha256:39ab8dbfa753a5faceb3772a462a3b30f4da77824a0b14
=> resolving dockerfile1: gsha256:39ab8dbfa753a5faceb3772a462a3b30f4da77824a0b14
=> sha256:39ab8dbfa753a5faceb3772a462a3b30f4da77824a0b14 9.40MB / 9.40MB
=> sha256:966a40f9a3667ac27fa35f76bc7b167d20af2ad242bd8b9b33a04621 482B / 482B
=> sha256:d8d11720762d5f4dc8a161c94de5eb4805947aada4a8034a59843bay 2.90KB / 2.90KB
=> sha256:a57ff70a6597ea012ba0b201716535e3b08f75a000c5d6ac9a10150be1bd 11.59MB / 11.59MB
=> extracting sha256:a57ff70a6597ea012ba0b201716535e3b08f75a000c5d6ac9a10150be1bd 8.22s
[web internal] load metadata for docker.io/library/python:3.7-alpine
[web internal] load build context
=> transferring context: 1.1kB
[web 1/6] FROM docker.io/library/python:3.7-alpine#gsha256:94cd807604a8da09cfe6597f11808adbcb2f9b4dc8ebc8134e4b733a15f
=> resolving docker.io/library/python:3.7-alpine#gsha256:94cd807604a8da09cfe6597f11808adbcb2f9b4dc8ebc8134e4b733a15f
=> sha256:94cd807604a8da09cfe6597f11808adbcb2f9b4dc8ebc8134e4b733a15f 1.65KB / 1.65KB
=> sha256:a8ccae2632109e3c473cf47ecccda84433595wcb892ba05b473336da 1.37KB / 1.37KB
=> sha256:c3f230870272178ec2d244dd1480ade09fc5f9774d4wd 0.87KB / 0.87KB
=> sha256:cfc2276e4459e359094a257df47647034e3b2a01f0d8c0b0a3485c39075ce 622.30Ks / 622.30Ks
=> sha256:84518510b3db0cdca452f07c9acc95d93b3a0d445f33cdf8b0db1ac5b 10.94MB / 10.94MB
=> sha256:41f2053e407719d2c10a06570b1671908cc47a7454ade3b20a0278b 24MB / 24MB
=> extracting sha256:cfc2276e4459e359094a257df47647034e3b2a01f0d8c0b0a3485c39075ce 0.44s
=> extracting sha256:84518510b3db0cdca452f07c9acc95d93b3a0d445f33cdf8b0db1ac5b 1.08s
=> extracting sha256:f03299a8149f4680b234a48083d7c80f15e4ab5d153cd3742b9c19f8 2.85MB / 2.85MB
=> extracting sha256:84518510b3db0cdca452f07c9acc95d93b3a0d445f33cdf8b0db1ac5b 1.08s
=> extracting sha256:c1f2053e407719d2c10a06570b1671908cc47a7454ade3b20a0278b 0.92s
=> extracting sha256:f03299a8149f4680b234a48083d7c80f15e4ab5d153cd3742b9c19f8 0.15s
[web 2/6] WORKDIR /code
[web 3/6] RUN apk add --no-cache gcc musl-dev linux-headers
[web 4/6] COPY requirements.txt requirements.txt
[web 5/6] RUN pip install -r requirements.txt
[web 6/6] COPY . .
[web] exporting to image
=> exporting layers
=> writing image sha256:c8f35ace5ebd542be27255f95584105b3c13493a345f053e54a6d6b7599f
=> naming to docker.io/library/composetest-web
[+] Network composetest_default Created
[+] Container composetest-web-1 Created
Attaching to composetest-red1-1, composetest-web-1
composetest-red1-1 | I[C 28 Jul 2023 11:11:11.599 # 000000000000 Redis is starting 000000000000
composetest-red1-1 | I[C 28 Jul 2023 11:11:11.599 Redis version=7.0.12, bits=64, commit=00000000, modified=0, pid=1, just started
composetest-red1-1 | I[C 28 Jul 2023 11:11:11.599 WARNING: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
composetest-red1-1 | I[M 28 Jul 2023 11:11:11.599 * monotonic clock: POSIX clock_gettime
composetest-red1-1 | I[M 28 Jul 2023 11:11:11.605 * Running mode standalone, port=6379.
composetest-red1-1 | I[M 28 Jul 2023 11:11:11.605 * Server initialized
composetest-red1-1 | I[M 28 Jul 2023 11:11:11.605 * WARNING Memory overcommit must be enabled! Without it, a background save or replication may fail under low memory condition. Being disabled, it can also cause failures without low memory condition, see https://github.com/jemalloc/jemalloc/issues/1328. To fix this issue add '*vm.overcommit_memory = 1' to /etc/sysctl.conf' and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
composetest-web-1 | I[W 28 Jul 2023 11:11:11.607 * Ready to accept connections
composetest-web-1 | * Serving flask app 'app.py'
```

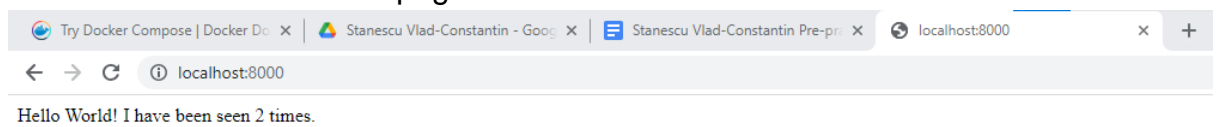
Compose pulls a Redis image, builds an image for my code, and starts the services I defined. In this case, the code is statically copied into the image at build time.

I entered `http://localhost:8000/` in a browser to see the application running and I see a message in my browser saying:

```
Hello World! I have been seen 1 times.
```



- After I refreshed the page the number has incremented:



- I typed `docker image ls` to list local images.

```
C:\Users\vlads>docker image ls
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
composetest-web     latest     c8f35e3e5e8d  7 minutes ago  214MB
redis               alpine     c1dc010e6f24  2 weeks ago   30.2MB
```

To accomplish this task, I followed this guide:

<https://docs.docker.com/compose/gettingstarted/>