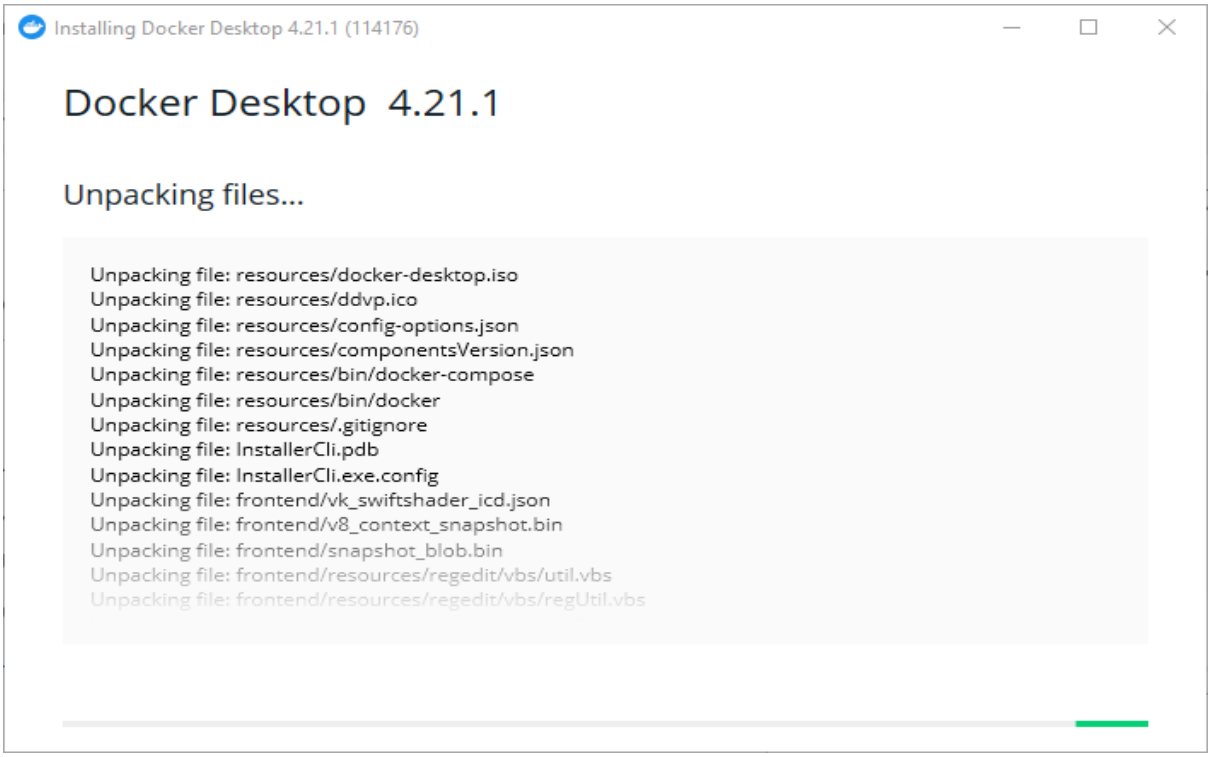
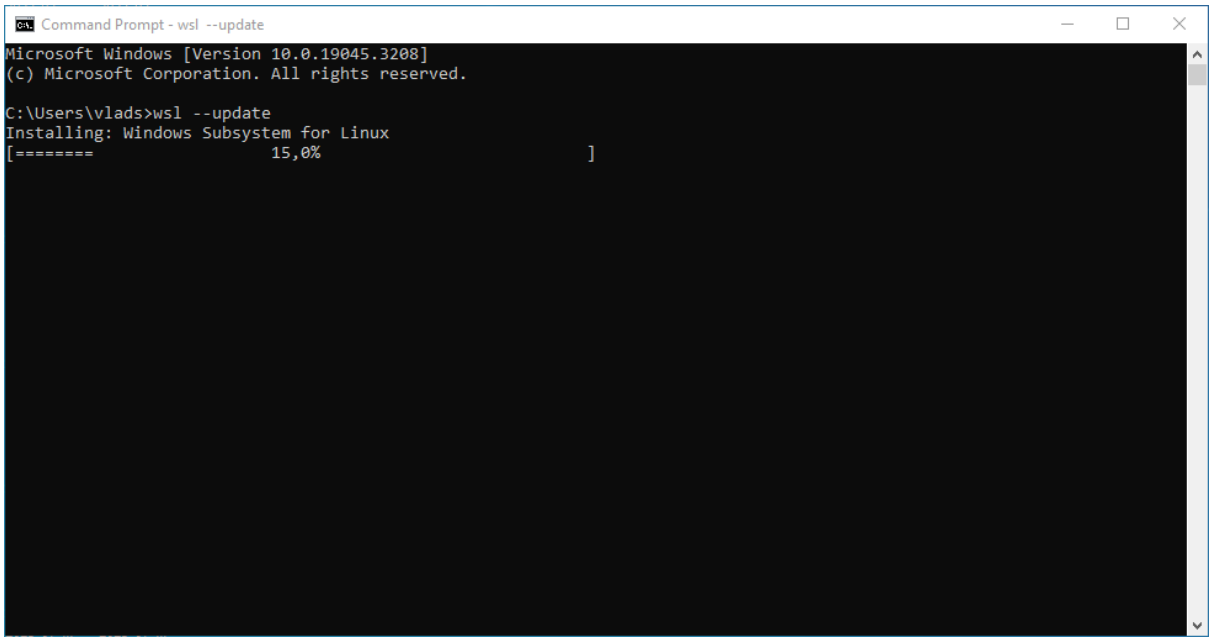
I will introduce the key concepts of Docker Compose whilst building a simple Python web application. I installed the Docker Desktop which includes both Docker Engine and Docker Compose. From July 2023 Compose V1 stopped receiving updates so I needed to migrate to Compose V2.





- 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

```python
import time

import redis
from flask import Flask

app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)

def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    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

```
# syntax=docker/dockerfile:1
FROM python:3.7-alpine
WORKDIR /code
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
EXPOSE 5000
COPY . .
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

```
services:
  web:
    build: .
    ports:
      - "8000:5000"
  redis:
    image: "redis:alpine"
```

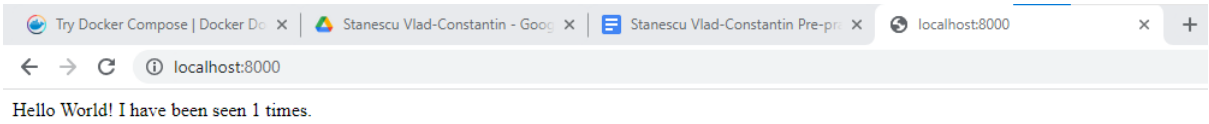
This Compose file defines two services: web and redis. The web service uses an image that's built from the Dockerfile in the current directory. It then binds the container and the host machine to the exposed port, 8000. This example service uses the default port for the Flask web server, 5000. The redis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

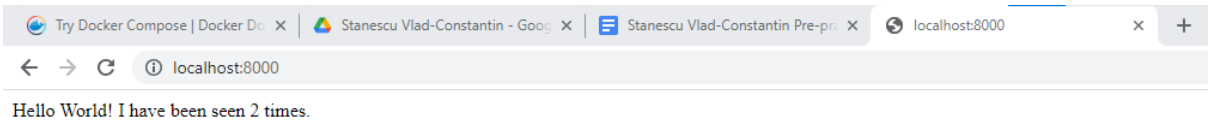
Compose pulls a Redis image, builds an image for my code, and starts the services I defined. In this case, the code is statically copied into the image at build time.

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/