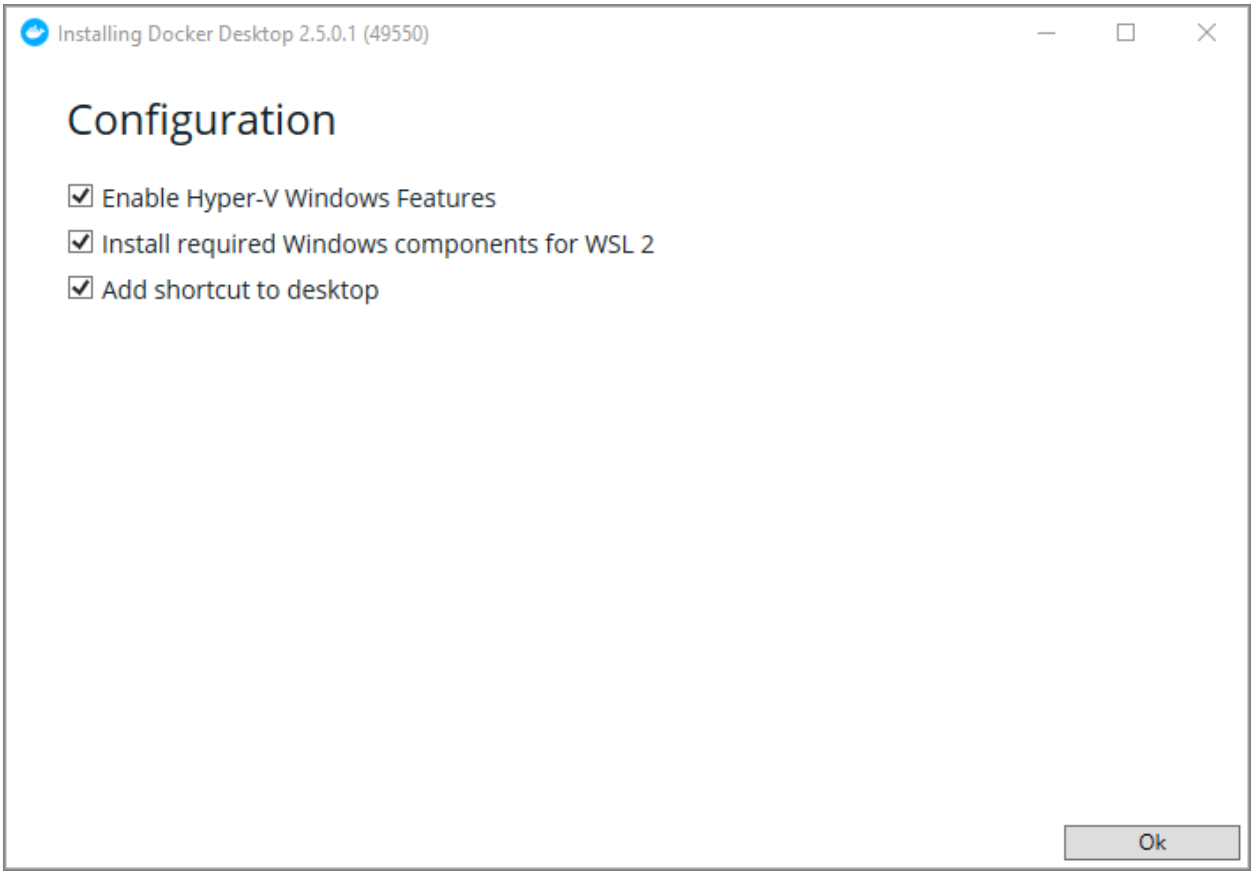
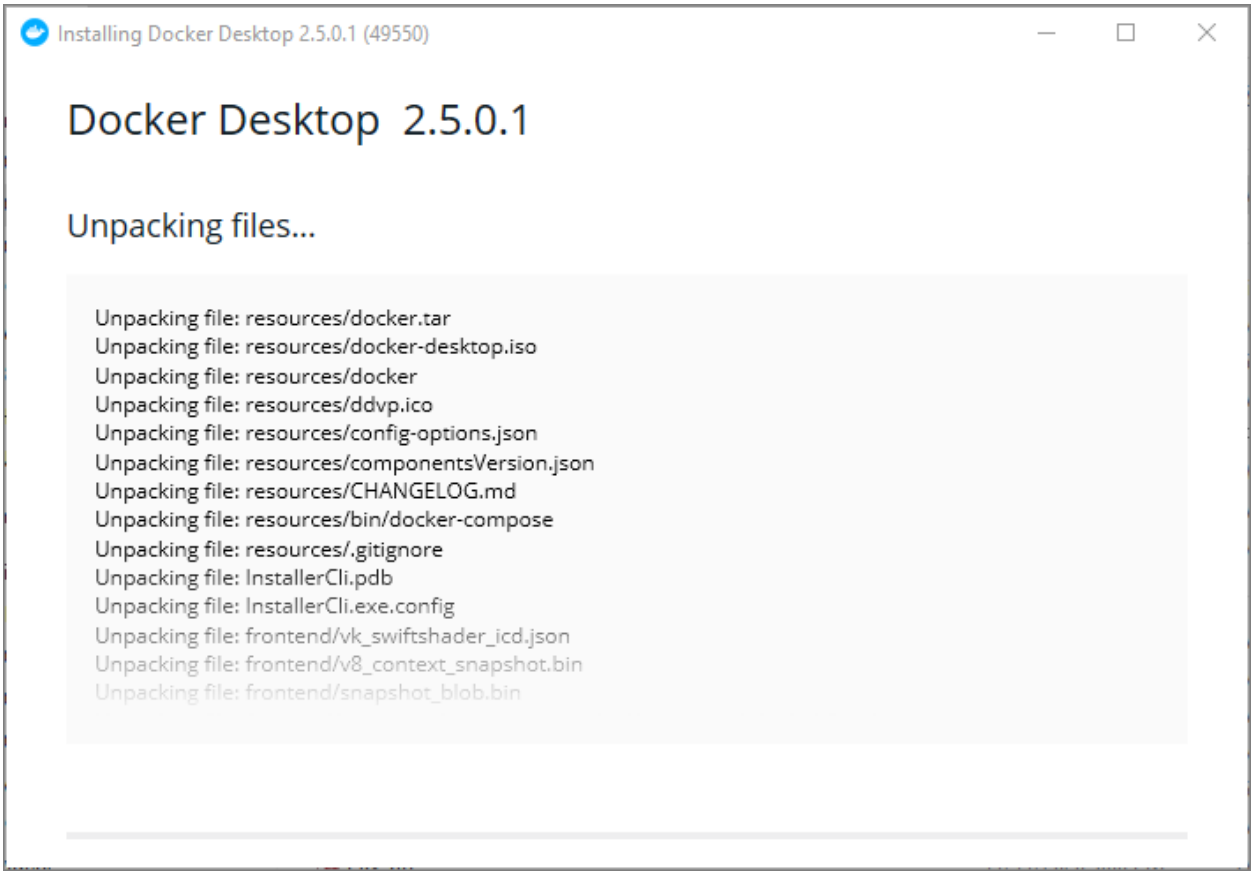
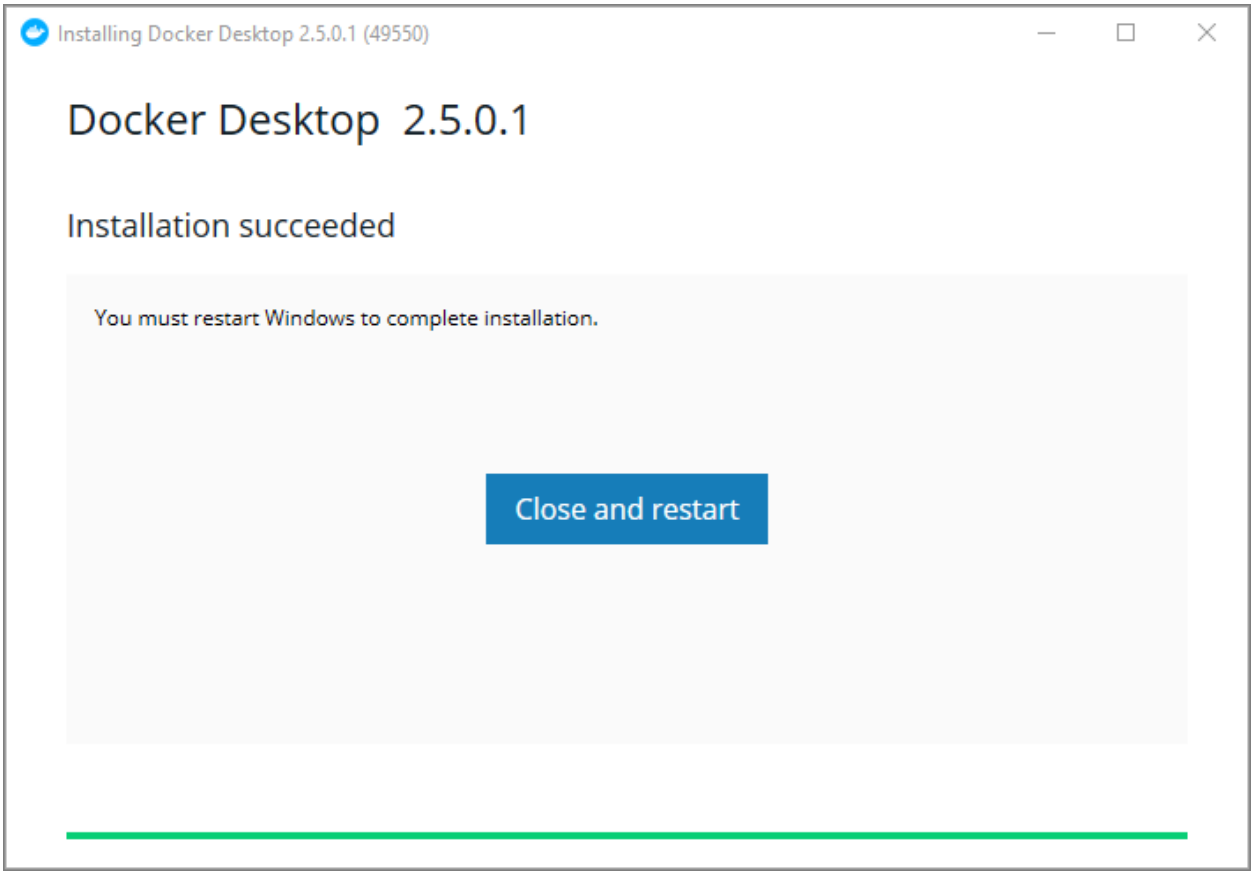
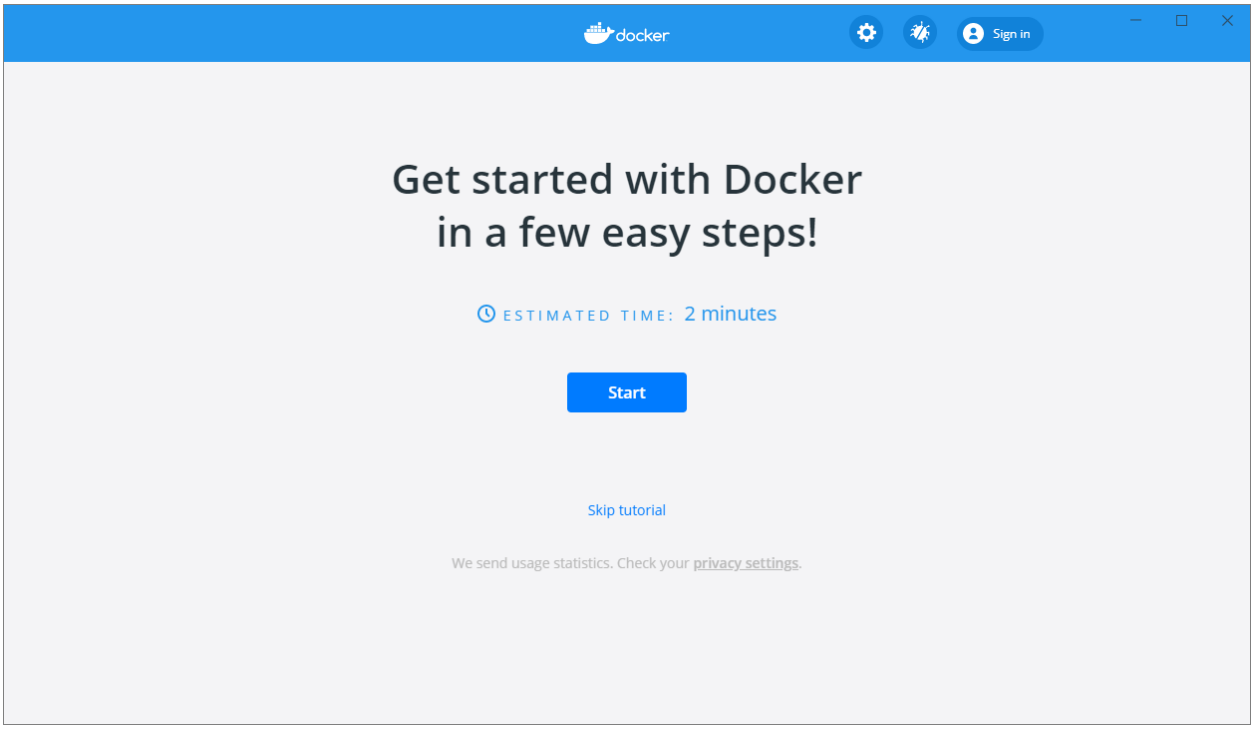
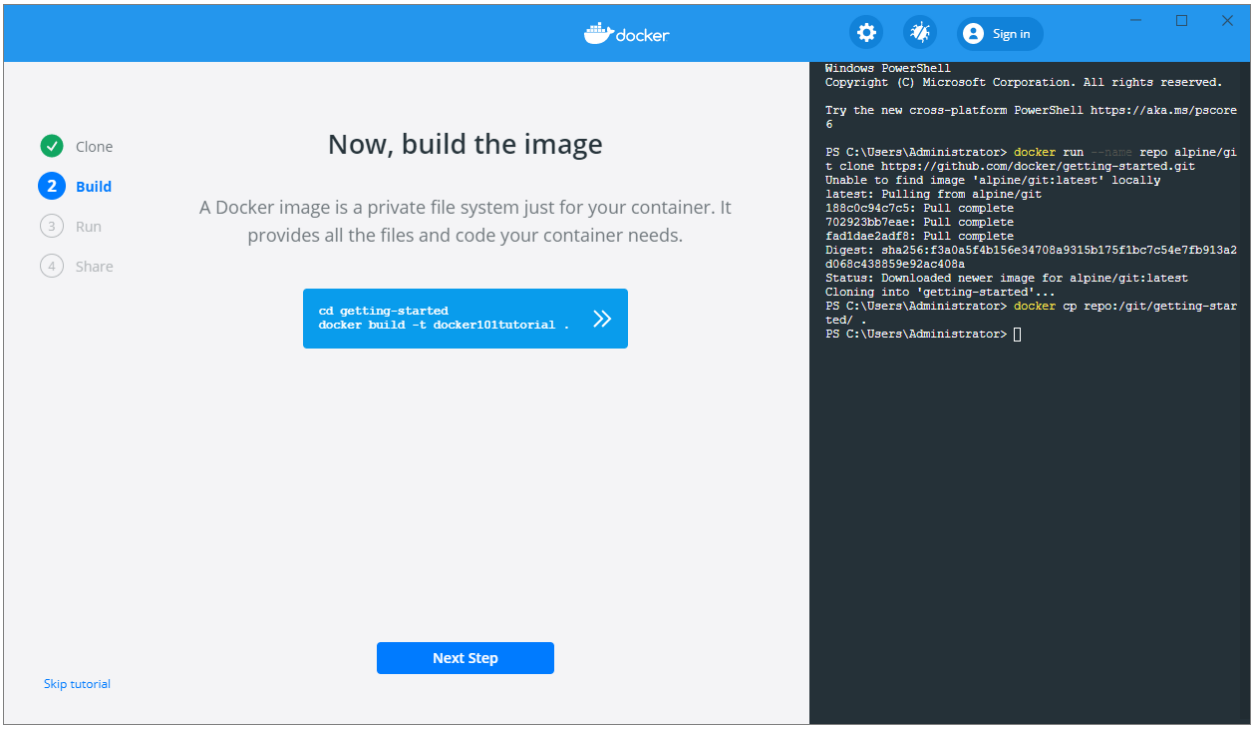

Installing Docker Desktop 2.5.0.1 (49550)
Configuration
Enable Hyper-V Windows Features
Install required Windows components for WSL 2
Add shortcut to desktop
Ok


Installing Docker Desktop 2.5.0.1 (49550)
Docker Desktop 2.5.0.1
Unpacking files...
Unpacking file: resources/docker.tar
Unpacking file: resources/docker-desktop.iso
Unpacking file: resources/docker
Unpacking file: resources/ddvp.ico
Unpacking file: resources/config-options.json
Unpacking file: resources/componentsVersion.json
Unpacking file: resources/CHANGELOG.md
Unpacking file: resources/bin/docker-compose
Unpacking file: resources/.gitignore
Unpacking file: InstallerCli.pdb
Unpacking file: InstallerCli.exe.config
Unpacking file: frontend/vk_swiftshader_icd.json
Unpacking file: frontend/v8_context_snapshot.bin
Unpacking file: frontend/snapshot_blob.bin


Installing Docker Desktop 2.5.0.1 (49550)
Docker Desktop 2.5.0.1
Installation succeeded
You must restart Windows to complete installation.
Close and restart


docker
Sign in
Get started with Docker
in a few easy steps!
ESTIMATED TIME: 2 minutes
Start
Skip tutorial
We send usage statistics. Check your privacy settings.



Creating webapp.py application

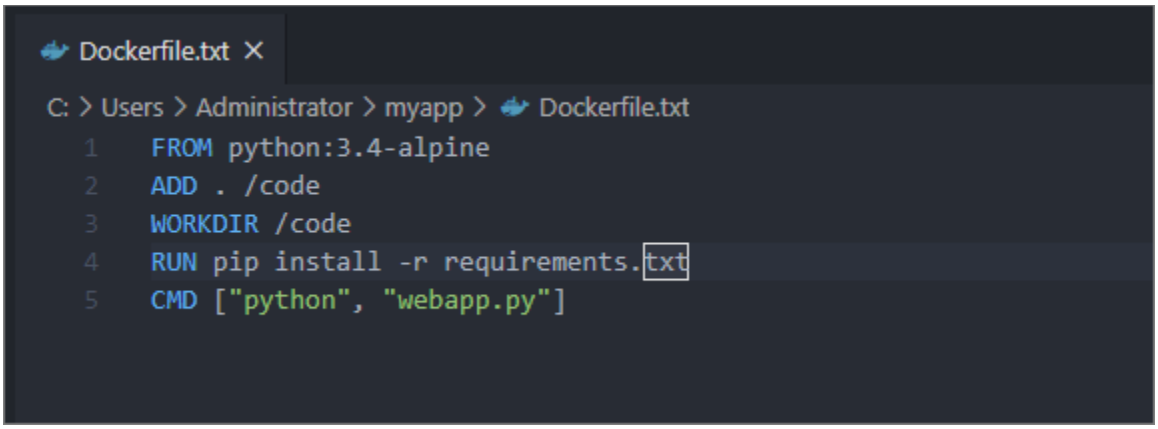
docker-compose.yml | webapp.py

C: > Users > Administrator > myapp > webapp.py > get_hit_count

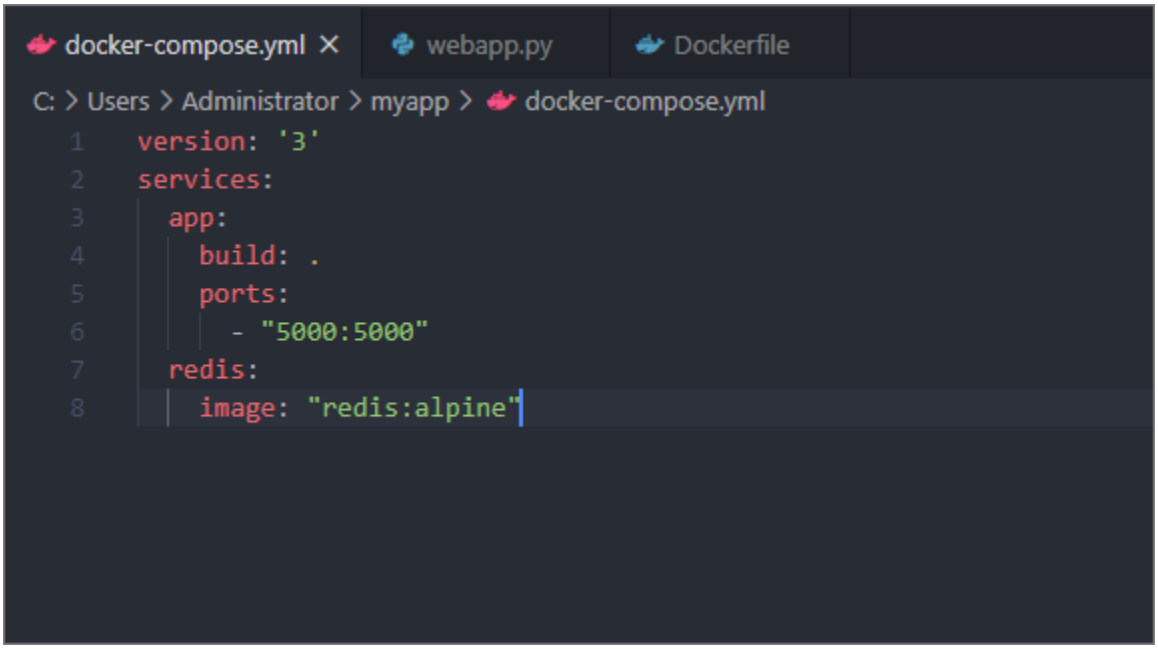
```python
import time
import redis
from flask import Flask
app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)
def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if 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.'.format(count))

if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True)
```

Creating requirements.txt file

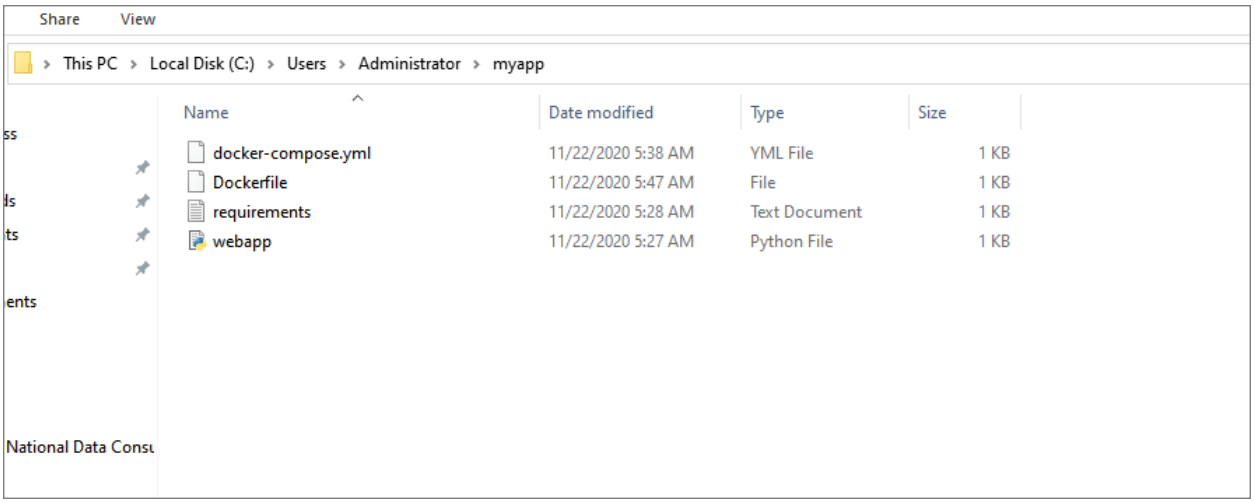
Creating Dockerfile. Note on saving .txt extension was removed



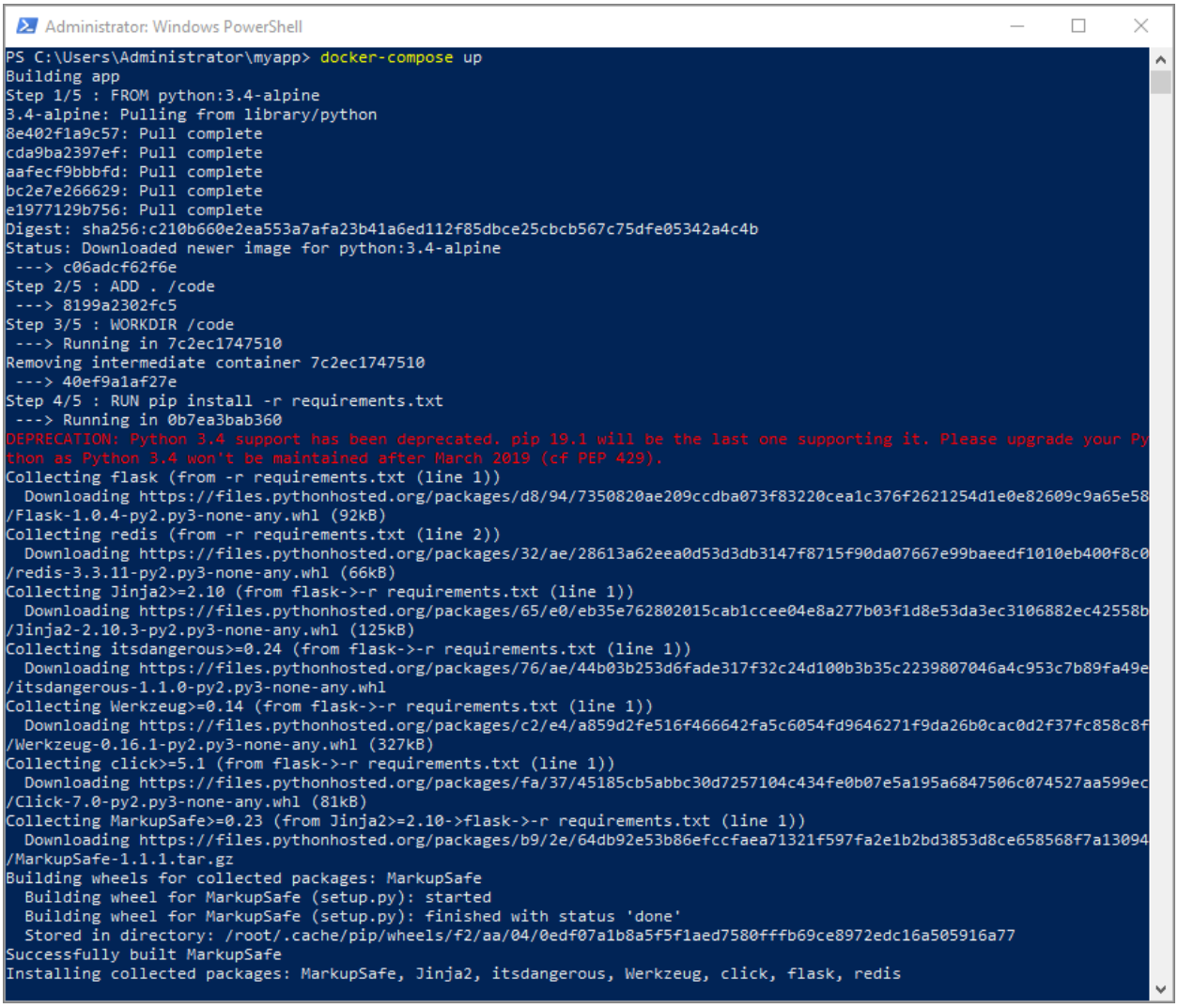
Create YAML file called docker-compose



Folder containing all 4 files required to run docker-compose up command



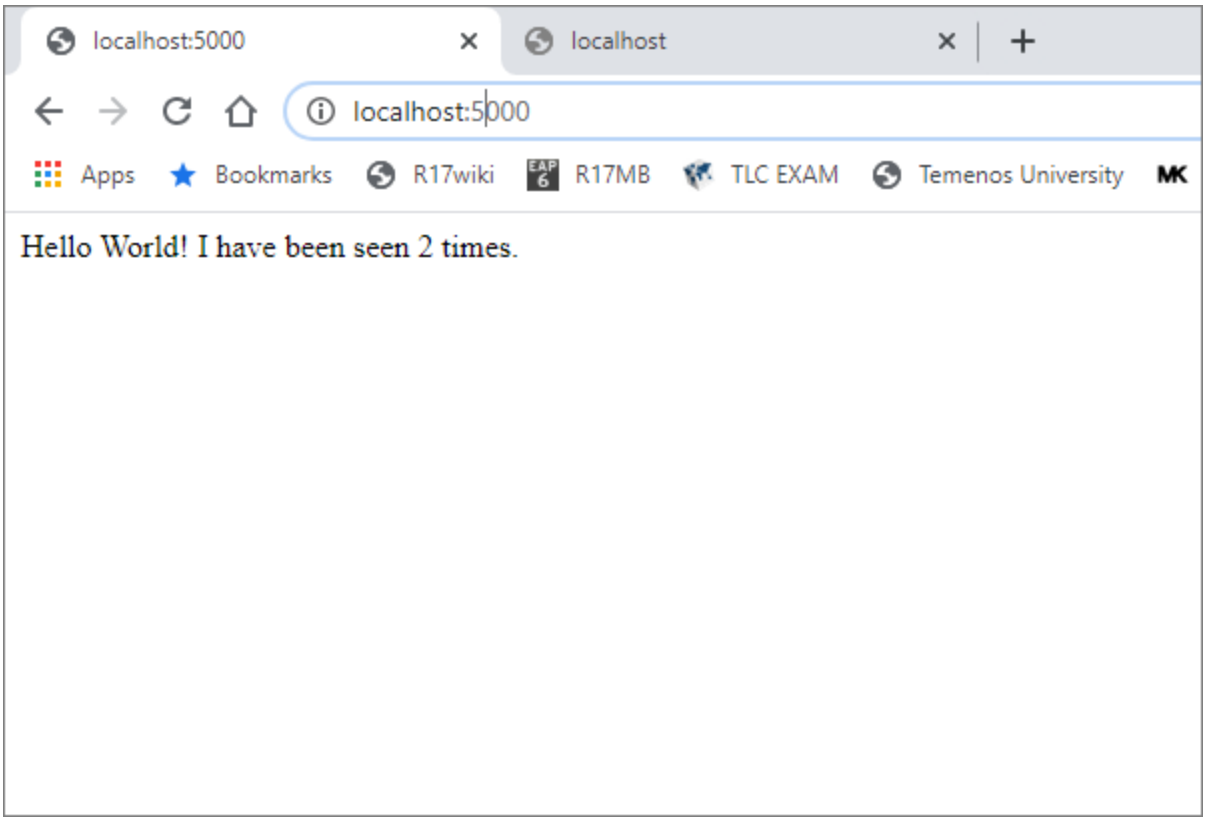
Running **docker-compose up** command



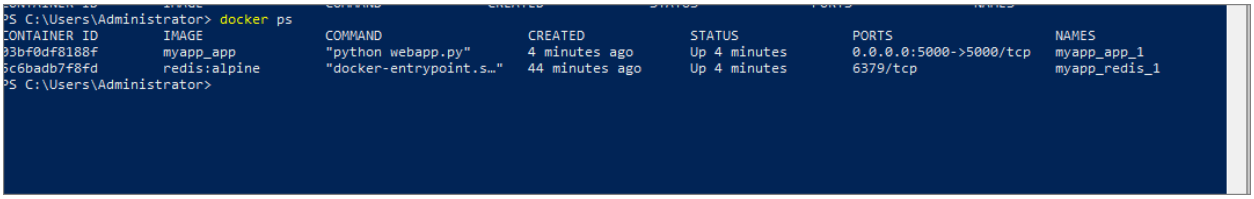
Our application is running now

```
redis_1  | 1:M 22 Nov 2020 01:29:08.163 # Server initialized
redis_1  | 1:M 22 Nov 2020 01:29:08.163 # WARNING overcommit_memory is set to 0! Background save may fail under low memo
ry condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 's
ysctl vm.overcommit_memory=1' for this to take effect.
redis_1  | 1:M 22 Nov 2020 01:29:08.163 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel.
This will create latency and memory usage issues with Redis. To fix this issue run the command 'echo madvise > /sys/kern
el/mm/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after a reb
oot. Redis must be restarted after THP is disabled (set to 'madvise' or 'never').
redis_1  | 1:M 22 Nov 2020 01:29:08.164 * Loading RDB produced by version 6.0.9
redis_1  | 1:M 22 Nov 2020 01:29:08.164 * RDB age 45 seconds
redis_1  | 1:M 22 Nov 2020 01:29:08.164 * RDB memory usage when created 0.77 Mb
redis_1  | 1:M 22 Nov 2020 01:29:08.164 * DB loaded from disk: 0.000 seconds
redis_1  | 1:M 22 Nov 2020 01:29:08.164 * Ready to accept connections
app_1    |  * Serving Flask app "webapp" (lazy loading)
app_1    |  * Environment: production
app_1    |    WARNING: This is a development server. Do not use it in a production deployment.
app_1    |    Use a production WSGI server instead.
app_1    |  * Debug mode: on
app_1    |  * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
app_1    |  * Restarting with stat
app_1    |  * Debugger is active!
app_1    |  * Debugger PIN: 110-616-467
app_1    | 172.18.0.1 - - [22/Nov/2020 01:29:18] "GET / HTTP/1.1" 200 -
app_1    | 172.18.0.1 - - [22/Nov/2020 01:29:18] "GET /favicon.ico HTTP/1.1" 404 -
app_1    | 172.18.0.1 - - [22/Nov/2020 01:29:29] "GET / HTTP/1.1" 200 -
```

On browser we can check. On refreshing again we see that the counter has changed



docker ps shows containers that are running



Q) What are the methods to create a docker swarm?

A docker swarm is a cluster where multiple containers are running on different nodes connected and communicated over a network. These can be physical machines or virtual machines

To create a docker swarm, we need at least one machine called manager node and other as worker node depending on how many nodes are needed. An IP address must be assigned to a network interface available to the host operating system. All nodes in the swarm need to connect to the manager at the IP address which should be fixed

1. Create Docker machines by running command:
   "**docker-machine create --driver hyperv manager1"** on one machine &
   "docker-machine create --driver hyperv worker1" on another machine

2. Get manager1 ip by running below command

   "**docker-machine ip manager1**"

   After that run below command

   "**docker-machine ssh manager1**" and initialize docker swarm:

   "**docker swarm init --advertise-addr <MANAGER-IP>**"

   Once we run this command, docker swarm will be initialized and a link to join docker swarm is displayed. We will use this link on worker nodes so they can join. The link will look something like this:

**docker swarm join **

**--token SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c **

192.168.99.100:2377

3. On worker node run “**docker-machine ssh worker1”** and then enter the link obtained from step 2. This was worker1 node will become part of docker swarm

4. We will deploy a service and thereby creating replicas