Docker Compose

It is a tool which is used to create and start Docker application by using a single command. We can use it to file to configure our application's services.

It is a great tool for development, testing, and staging environments.

It provides the following commands for managing the whole lifecycle of our application.

- Start, stop and rebuild services
- View the status of running services
- Stream the log output of running services
- Run a one-off command on a service

To implement compose, it consists the following steps.

1. Put Application environment variables inside the Dockerfile to access publicly.
2. Provide services name in the docker-compose.yml file so they can be run together in an isolated environment.
3. run docker-compose up and Compose will start and run your entire app.

A typical **docker-compose.yml** file has the following format and arguments.

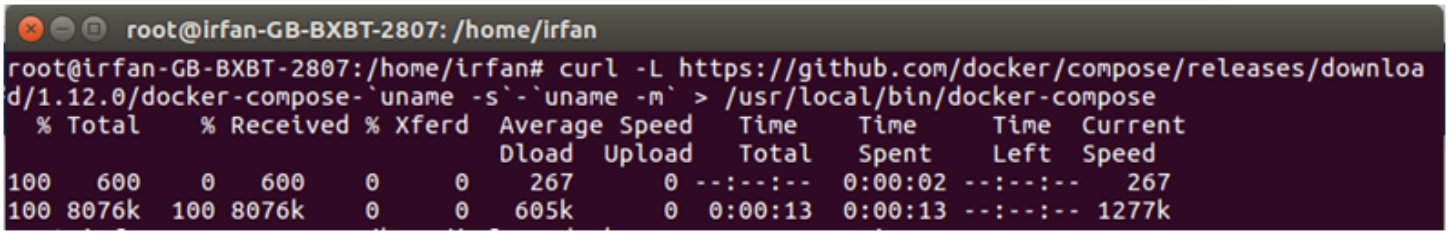
// docker-compose.yml

```
version: '3'
services:
web:
build: .
ports:
- "5000:5000"
volumes:
- .:/code
- logvolume01:/var/log
links:
- redis
redis:
image: redis
volumes:
logvolume01: {}
```

Installing Docker Compose

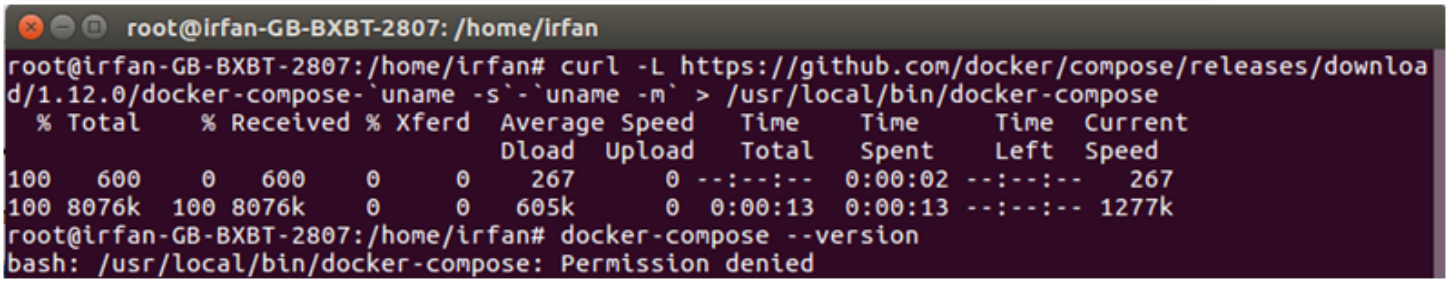
Following are the instructions to install Docker Compose in Linux Ubuntu.

```
curl -L https://github.com/docker/compose/releases/download/1.12.0/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose
```



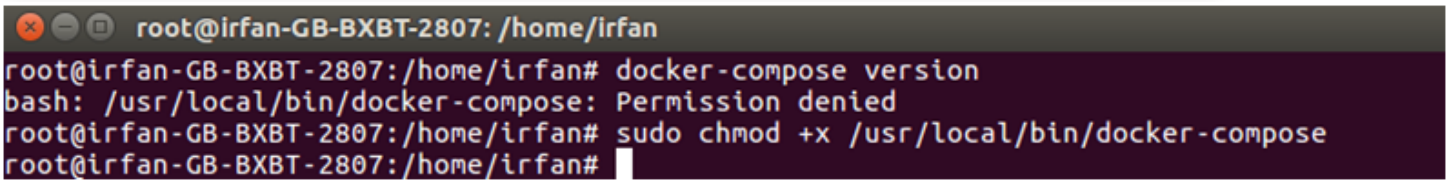
Docker-compose version

```
$ docker-compose --version
```



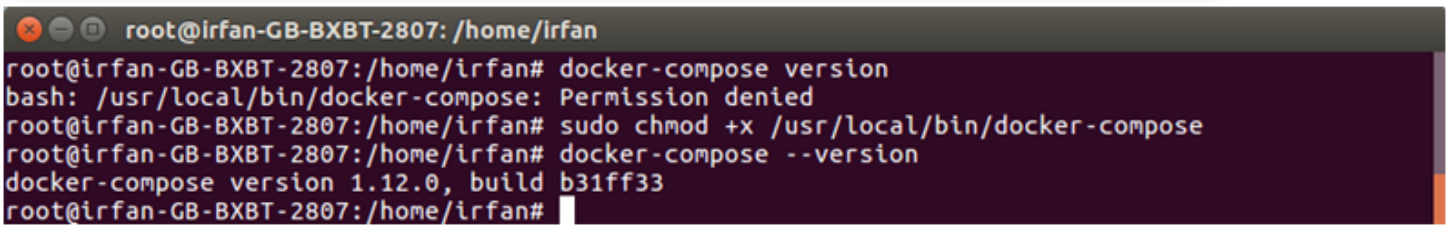
It says, permission denied. So, make file executable.

```
$ sudo chmod +x /usr/local/bin/docker-compose
```



Now, check version again.

```
$ docker-compose ?version
```



Running Application using Docker Compose

Example

Follow the following example

1) Create a Directory

```
$ mkdir docker-compose-example
$ cd docker-composer-example
```

2) Create a file **app.py.**

// app.py

```
from flask import Flask
from redis import Redis
app = Flask(__name__)
redis = Redis(host='redis', port=6379)
@app.route('/')
def hello():
count = redis.incr('hits')
return 'Hello World! I have been seen {} times.\n'.format(count)
if __name__ == "__main__":
app.run(host="0.0.0.0", debug=True)
```

3) Create a file **requirements.txt.**

// requirements.txt

```
flask
redis
```

4) Create a Dockerfile.

// Dockerfile

```
FROM python:3.4-alpine
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
```

5) Create a Compose File.

// docker-compose.yml

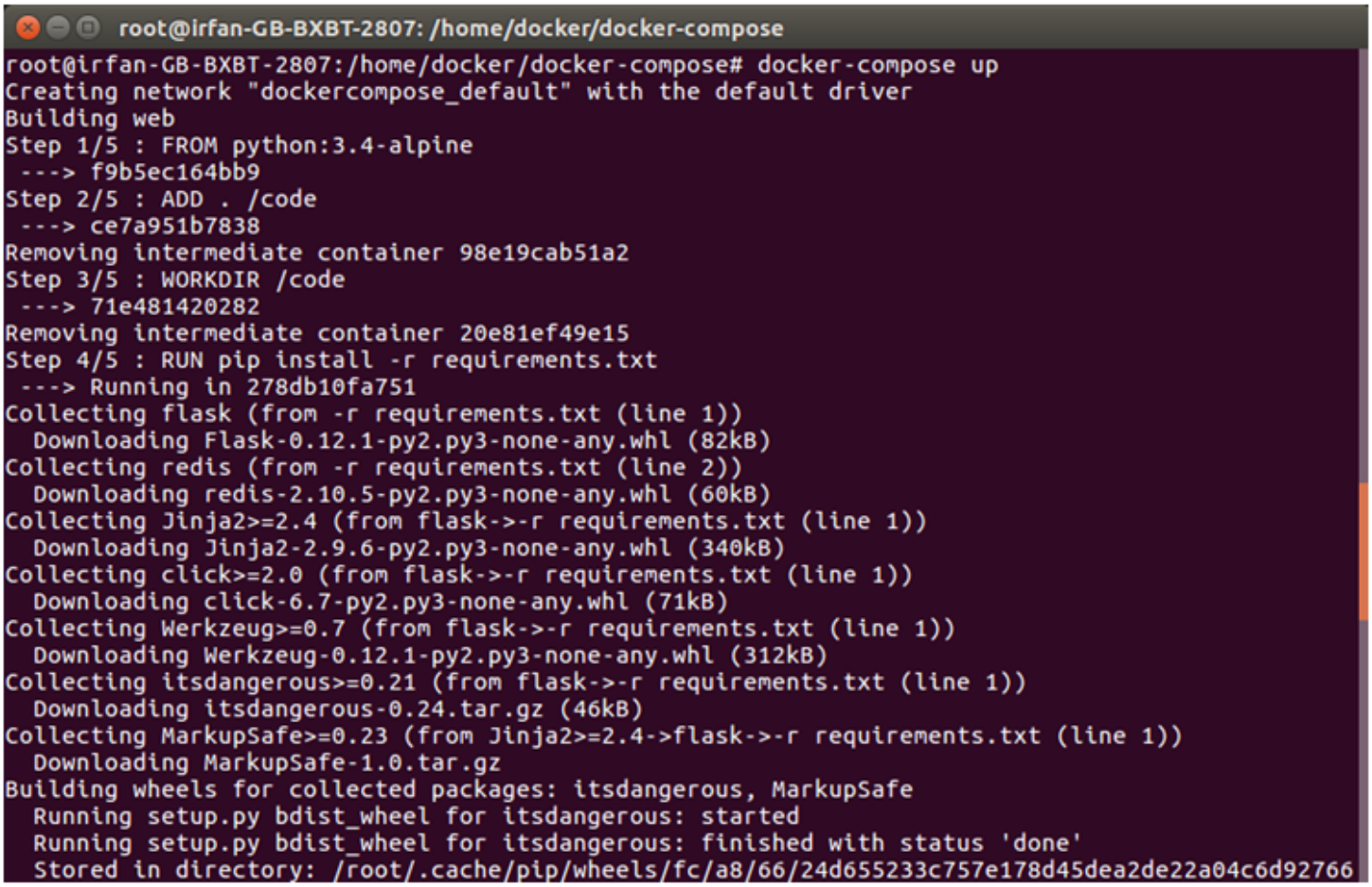
```
version: '2'
services:
web:
build: .
```

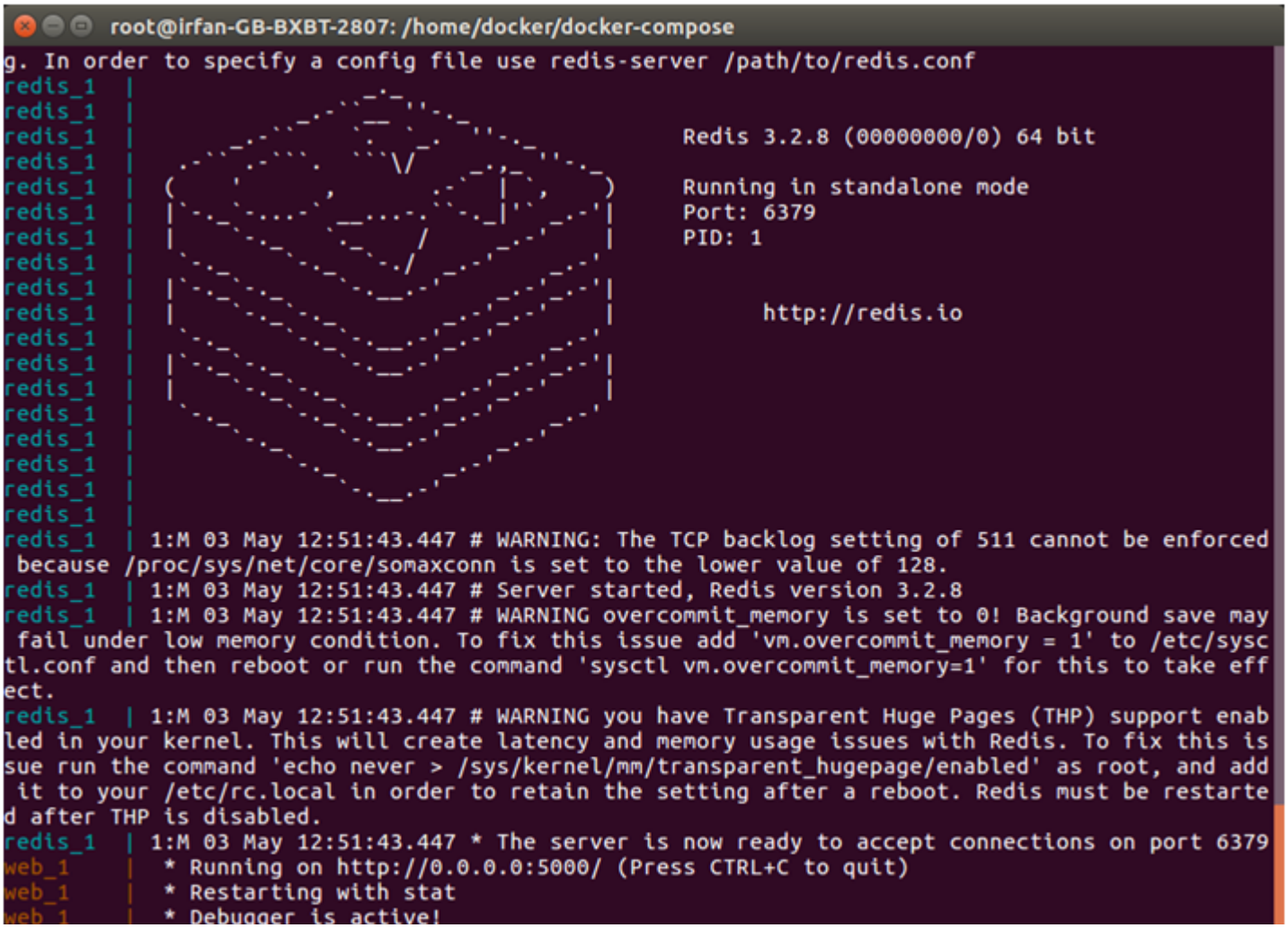
```
ports:
- "5000:5000"
volumes:
- .:/code
redis:
image: "redis:alpine"
```

6) Build and Run Docker App with Compose

```
$ docker-compose up
```

After running the above command, it shows the following output.


root@irfan-GB-BXBT-2807: /home/docker/docker-compose
root@irfan-GB-BXBT-2807:/home/docker/docker-compose# docker-compose up
Creating network "dockercompose_default" with the default driver
Building web
Step 1/5 : FROM python:3.4-alpine
 ---> f9b5ec164bb9
Step 2/5 : ADD . /code
 ---> ce7a951b7838
Removing intermediate container 98e19cab51a2
Step 3/5 : WORKDIR /code
 ---> 71e481420282
Removing intermediate container 20e81ef49e15
Step 4/5 : RUN pip install -r requirements.txt
 ---> Running in 278db10fa751
Collecting flask (from -r requirements.txt (line 1))
  Downloading Flask-0.12.1-py2.py3-none-any.whl (82kB)
Collecting redis (from -r requirements.txt (line 2))
  Downloading redis-2.10.5-py2.py3-none-any.whl (60kB)
Collecting Jinja2>=2.4 (from flask->-r requirements.txt (line 1))
  Downloading Jinja2-2.9.6-py2.py3-none-any.whl (340kB)
Collecting click>=2.0 (from flask->-r requirements.txt (line 1))
  Downloading click-6.7-py2.py3-none-any.whl (71kB)
Collecting Werkzeug>=0.7 (from flask->-r requirements.txt (line 1))
  Downloading Werkzeug-0.12.1-py2.py3-none-any.whl (312kB)
Collecting itsdangerous>=0.21 (from flask->-r requirements.txt (line 1))
  Downloading itsdangerous-0.24.tar.gz (46kB)
Collecting MarkupSafe>=0.23 (from Jinja2>=2.4->flask->-r requirements.txt (line 1))
  Downloading MarkupSafe-1.0.tar.gz
Building wheels for collected packages: itsdangerous, MarkupSafe
  Running setup.py bdist_wheel for itsdangerous: started
  Running setup.py bdist_wheel for itsdangerous: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/fc/a8/66/24d655233c757e178d45dea2de22a04c6d92766


root@irfan-GB-BXBT-2807: /home/docker/docker-compose
g. In order to specify a config file use redis-server /path/to/redis.conf
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
redis_1 |
Redis 3.2.8 (00000000/0) 64 bit
Running in standalone mode
Port: 6379
PID: 1
http://redis.io
redis_1 | 1:M 03 May 12:51:43.447 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.
redis_1 | 1:M 03 May 12:51:43.447 # Server started, Redis version 3.2.8
redis_1 | 1:M 03 May 12:51:43.447 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. 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.
redis_1 | 1:M 03 May 12:51:43.447 # 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 never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after a reboot. Redis must be restarted after THP is disabled.
redis_1 | 1:M 03 May 12:51:43.447 * The server is now ready to accept connections on port 6379
web_1 | * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
web_1 | * Restarting with stat
web_1 | * Debugger is active!

Now, we can see the output by following the running http url.

Output:

Each time, when we refresh the page. It shows counter incremented by 1.