


# Docker Compose

## docker run

Filter (Ctrl+Shift+F)

```
docker run -d --name=redis redis
docker run -d --name=db postgres
docker run -d --name=vote -p 5000:80 voting-app
docker run -d --name=result -p 5001:80 result-app
docker run -d --name=worker worker
```



500 Internal Server Error

Cats vs Dogs -- Result

192.168.56.101:5000

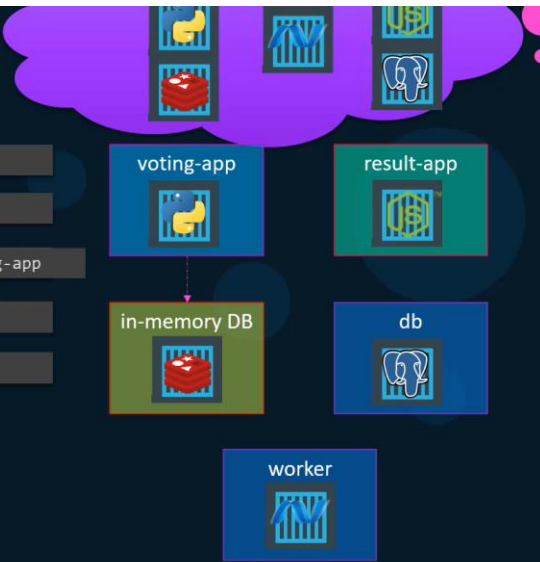
### Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

## docker run --links

Filter (Ctrl+Shift+F)

```
docker run -d --name=redis redis
docker run -d --name=db postgres
docker run -d --name=vote -p 5000:80 --link redis:redis voting-app
docker run -d --name=result -p 5001:80 result-app
docker run -d --name=worker worker
```



```
def get_redis():
    if not hasattr(g, 'redis'):
        g.redis = Redis(host="redis", db=0, socket_timeout=5)
    return g.redis
```

# docker run --links

Filter (Ctrl+Shift+F)

```
docker run -d --name=redis redis
```

```
docker run -d --name=db postgres
```

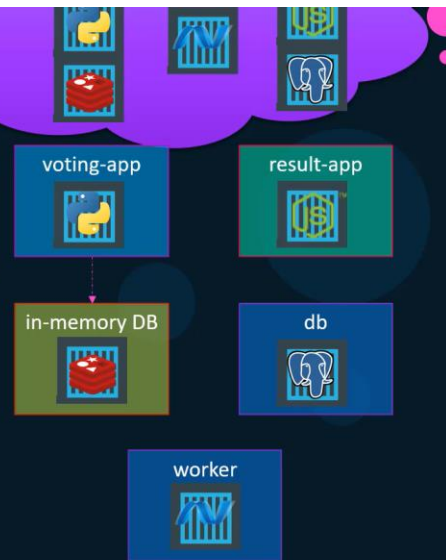
```
docker run -d --name=vote -p 5000:80 --link redis:redis voting-app
```

```
docker run -d --name=result -p 5001:80 result-app
```

```
docker run -d --name=worker worker
```

```
def get_redis():  
    if not hasattr(g, 'redis'):  
        g.redis = Redis(host="redis", db=0, socket_timeout=5)  
    return g.redis
```

```
/app # cat /etc/hosts  
127.0.0.1      localhost  
::1           localhost ip6-localhost ip6-loopback  
fe00::0       ip6-localnet  
ff00::0       ip6-mcastprefix  
ff02::1       ip6-allnodes  
ff02::2       ip6-allrouters  
172.17.0.2     redis 89cd8eb563da
```



# docker run --links

Filter (Ctrl+Shift+F)

```
docker run -d --name=redis redis
```

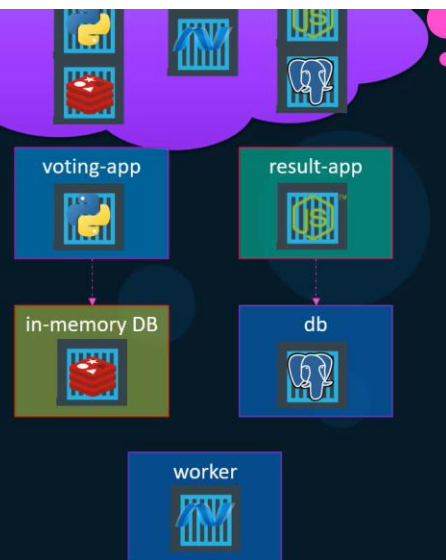
```
docker run -d --name=db postgres
```

```
docker run -d --name=vote -p 5000:80 --link redis:redis voting-app
```

```
docker run -d --name=result -p 5001:80 --link db:db result-app
```

```
docker run -d --name=worker worker
```

```
pg.connect('postgres://postgres@db/postgres', function(err, client, done) {  
    if (err) {  
        console.error("Waiting for db");  
    }  
    callback(err, client);  
});
```



# docker run --links

Filter (Ctrl+Shift+F)

```
docker run -d --name=redis redis
```

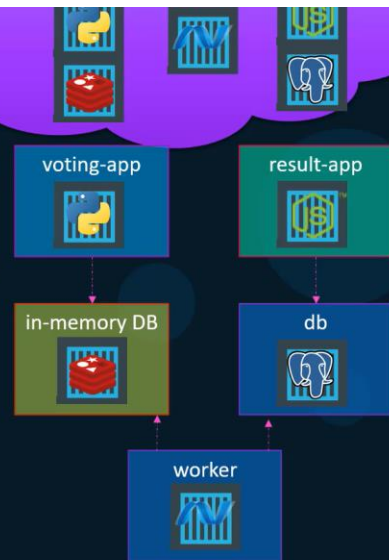
```
docker run -d --name=db postgres
```

```
docker run -d --name=vote -p 5000:80 --link redis:redis voting-app
```

```
docker run -d --name=result -p 5001:80 --link db:db result-app
```

```
docker run -d --name=worker --link db:db --link redis:redis worker
```

```
try {  
  Jedis redis = connectToRedis("redis");  
  Connection dbConn = connectToDB("db");  
  
  System.err.println("Watching vote queue");  
}
```



# Docker compose

Filter (Ctrl+Shift+F)

```
docker run -d --name=redis redis
```

```
docker run -d --name=db postgres:9.4
```

```
docker run -d --name=vote -p 5000:80 --link redis:redis voting-app
```

```
docker run -d --name=result -p 5001:80 --link db:db result-app
```

```
docker run -d --name=worker --link db:db --link redis:redis worker
```

docker-compose.yml

```
redis:  
  image: redis  
db:  
  image: postgres:9.4  
vote:  
  image: voting-app  
  ports:  
    - 5000:80  
  links:  
    - redis  
result:  
  image: result-app  
  ports:  
    - 5001:80  
  links:  
    - db  
worker:  
  image: worker  
  links:  
    - redis  
    - db
```

# Docker compose

Filter (Ctrl+Shift+F)

```
docker run -d --name=redis redis
```

```
docker run -d --name=db postgres:9.4
```

```
docker run -d --name=vote -p 5000:80 --link redis:redis voting-app
```

```
docker run -d --name=result -p 5001:80 --link db:db result-app
```

```
docker run -d --name=worker --link db:db --link redis:redis worker
```

db:db = db

```
docker-compose up
```

docker-compose.yml

```
redis:
  image: redis
db:
  image: postgres:9.4
vote:
  image: voting-app
  ports:
    - 5000:80
  links:
    - redis
result:
  image: result-app
  ports:
    - 5001:80
  links:
    - db
worker:
  image: worker
  links:
    - redis
    - db
```

To run `docker-compose`, follow these steps:

1. Create a `docker-compose.yml` file in the root directory of your project. This file defines the services that make up your app and how they interact.
2. Open a terminal and navigate to the directory that contains the `docker-compose.yml` file.
3. Run the command `docker-compose up` to start the containers defined in the `docker-compose.yml` file. This command will start the containers in the foreground and output their logs to the console.
4. If you want to run the containers in the background, run the command `docker-compose up -d` instead. This will start the containers in detached mode, meaning that they will run in the background and you can continue to use the terminal.
5. To stop the containers, run the command `docker-compose down`. This command will stop and remove the containers, as well as any networks and volumes that were created by `docker-compose up`.

Note that `docker-compose` requires Docker to be installed and running on your machine. Also, make sure that your `docker-compose.yml` file is properly formatted and defines all the required services and dependencies.

## Docker compose - build

Filter (Ctrl+Shift+F)

```
docker-compose.yml
redis:
  image: redis
db:
  image: postgres:9.4
vote:
  image: voting-app
  ports:
    - 5000:80
  links:
    - redis
result:
  image: result
  ports:
    - 5001:80
  links:
    - db
worker:
  image: worker
  links:
    - db
    - redis
```

```
docker-compose.yml
redis:
  image: redis
db:
  image: postgres:9.4
vote:
  build: ./vote
  ports:
    - 5000:80
  links:
    - redis
result:
  build: ./result
  ports:
    - 5001:80
  links:
    - db
worker:
  build: ./worker
  links:
    - db
    - redis
```

## Docker compose - build

Filter (Ctrl+Shift+F)

```
docker-compose.yml
redis:
  image: redis
db:
  image: postgres:9.4
vote:
  image: voting-app
  ports:
    - 5000:80
  links:
    - redis
result:
  image: result
  ports:
    - 5001:80
  links:
    - db
worker:
  image: worker
  links:
    - db
    - redis
```

```
docker-compose.yml
redis:
  image: redis
db:
  image: postgres:9.4
vote:
  build: ./vote
  ports:
    - 5000:80
  links:
    - redis
result:
  build: ./result
  ports:
    - 5001:80
  links:
    - db
worker:
  build: ./worker
  links:
    - db
    - redis
```

dockersamples / example-voting-app

Code Issues 3 Pull requests 4

Branch: master example-voting-app / vote /

bfirsh Put gunicorn command in list

..

- static/stylesheets Re
- templates Re
- Dockerfile Pu
- app.py Re
- requirements.txt Re