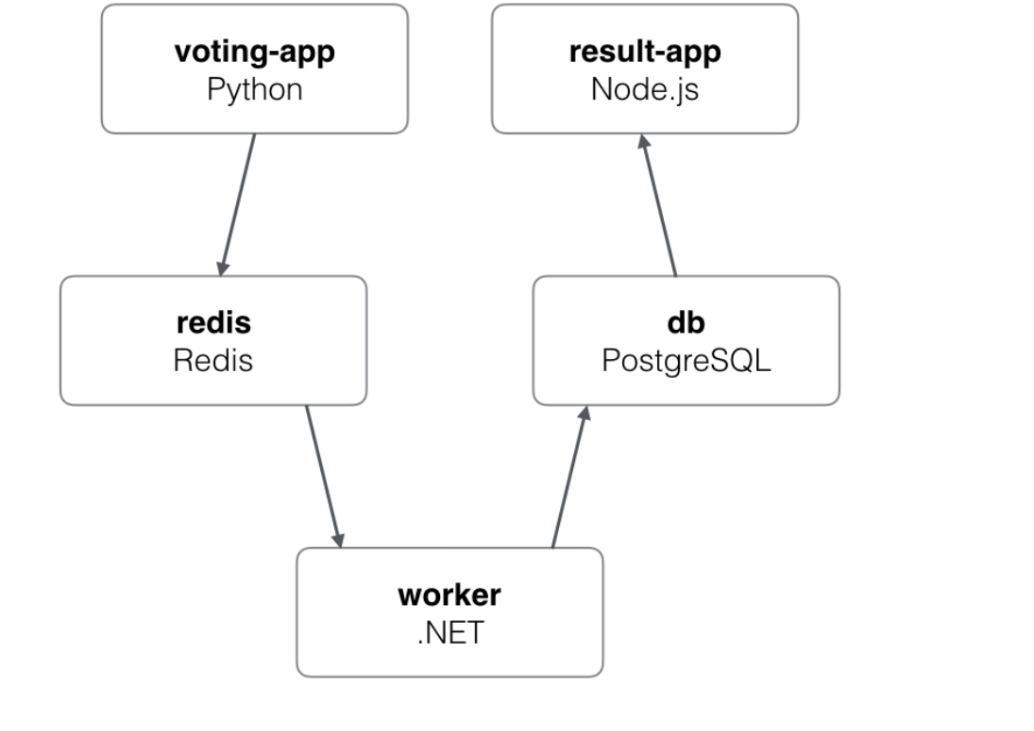
**Creating and Analyzing Voting Results using Docker Compose**

**Ref:** [dockersamples/example-voting-app: Example Docker Compose app (github.com)](https://github.com/dockersamples/example-voting-app)



* A front-end web app in [Python](https://github.com/dockersamples/example-voting-app/blob/master/vote) or [ASP.NET Core](https://github.com/dockersamples/example-voting-app/blob/master/vote/dotnet) which lets you vote between two options
* A [Redis](https://hub.docker.com/_/redis/) or [NATS](https://hub.docker.com/_/nats/) queue which collects new votes
* A [.NET Core](https://github.com/dockersamples/example-voting-app/blob/master/worker/src/Worker), [Java](https://github.com/dockersamples/example-voting-app/blob/master/worker/src/main) or [.NET Core 2.1](https://github.com/dockersamples/example-voting-app/blob/master/worker/dotnet) worker which consumes votes and stores them in…
* A [Postgres](https://hub.docker.com/_/postgres/) or [TiDB](https://hub.docker.com/r/dockersamples/tidb/tags/) database backed by a Docker volume
* A [Node.js](https://github.com/dockersamples/example-voting-app/blob/master/result) or [ASP.NET Core SignalR](https://github.com/dockersamples/example-voting-app/blob/master/result/dotnet) webapp which shows the results of the voting in real time

**Update details of all the services in a sing docker-compose file and run.**

vi docker-compose.yaml

version: "3"

services:

# A Redis key-value store to serve as message queue

redis:

image: redis:alpine

ports:

- "6379"

networks:

- frontend

# A PostgreSQL database for persistent storage

db:

image: postgres:9.4

volumes:

- db-data:/var/lib/postgresql/data

networks:

- backend

# Web UI for voting

vote:

image: dockersamples/examplevotingapp\_vote:before

ports:

- 5000:80

networks:

- frontend

depends\_on:

- redis

# Web UI to count voting results

result:

image: dockersamples/examplevotingapp\_result:before

ports:

- 5001:80

networks:

- backend

depends\_on:

- db

# Worker service to read from message queue

worker:

image: dockersamples/examplevotingapp\_worker

networks:

- frontend

- backend

networks:

frontend:

backend:

volumes:

db-data:

**2. Execute the docker-compose file .It would pull the images , create the containers and apply the settings**

[root@ip-172-31-10-59 votingapp]# docker-compose up -d

Creating network "votingapp\_frontend" with the default driver

Creating network "votingapp\_backend" with the default driver

Creating volume "votingapp\_db-data" with default driver

Pulling redis (redis:alpine)...

alpine: Pulling from library/redis

540db60ca938: Pull complete

29712d301e8c: Pull complete

8173c12df40f: Pull complete

8cc52074f78e: Pull complete

Creating votingapp\_redis\_1 ... done

Creating votingapp\_worker\_1 ... done

Creating votingapp\_db\_1 ... done

Creating votingapp\_result\_1 ... done

Creating votingapp\_vote\_1 ... done

**3.Check the created containers and observe the port details**

[root@ip-172-31-10-59 votingapp]# docker ps

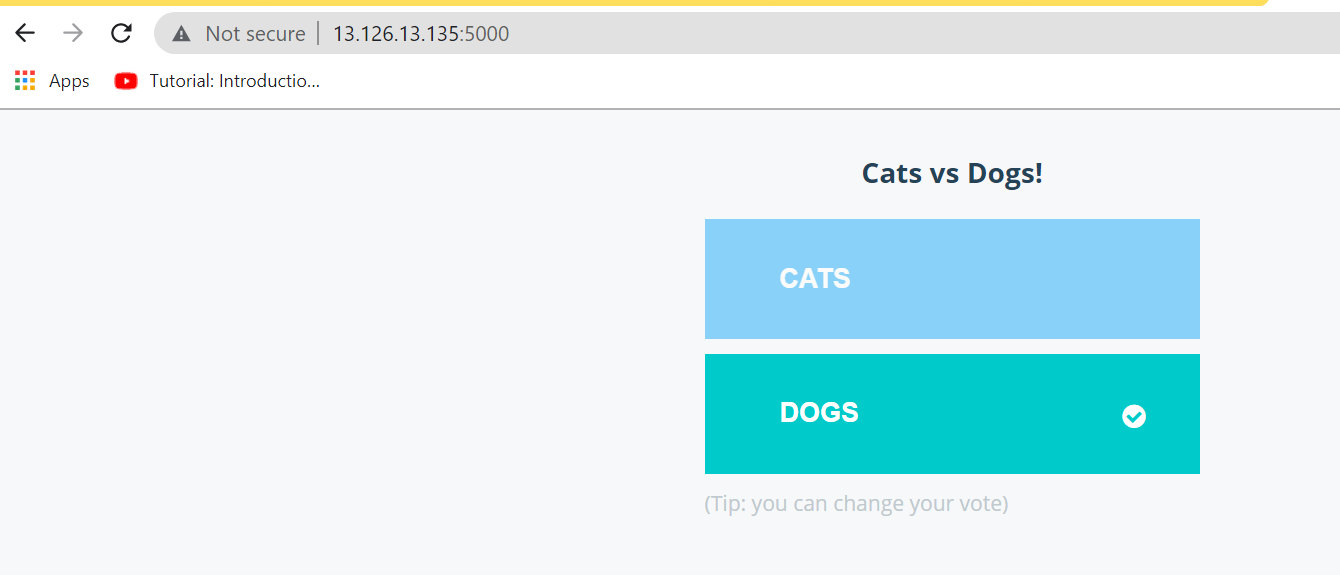
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

9bbcd1e90c96 dockersamples/examplevotingapp\_vote:before "gunicorn app:app -b…" 3 minutes ago Up 3 minutes 0.0.0.0:5000->80/tcp votingapp\_vote\_1

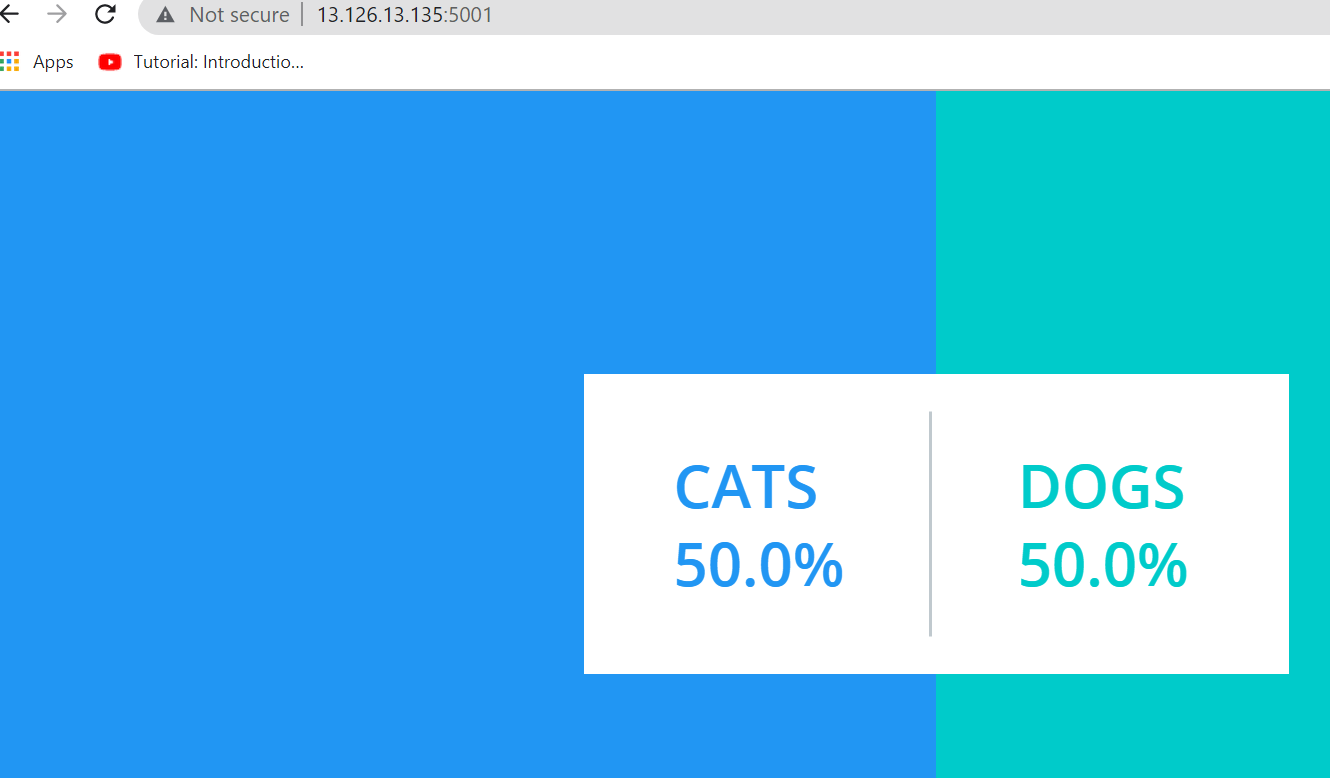
0bf601255ce5 dockersamples/examplevotingapp\_result:before "node server.js" 3 minutes ago Up 3 minutes 0.0.0.0:5001->80/tcp votingapp\_result\_1

c07f927b3434 redis:alpine "docker-entrypoint.s…" 3 minutes ago Up 3 minutes 0.0.0.0:49153->6379/tcp votingapp\_redis\_1

**Great! The app is deployed so we can cast votes by accessing the service that’s listening on port 5000.**



**Check the results:**



**Clean the environment:**

docker-compose down