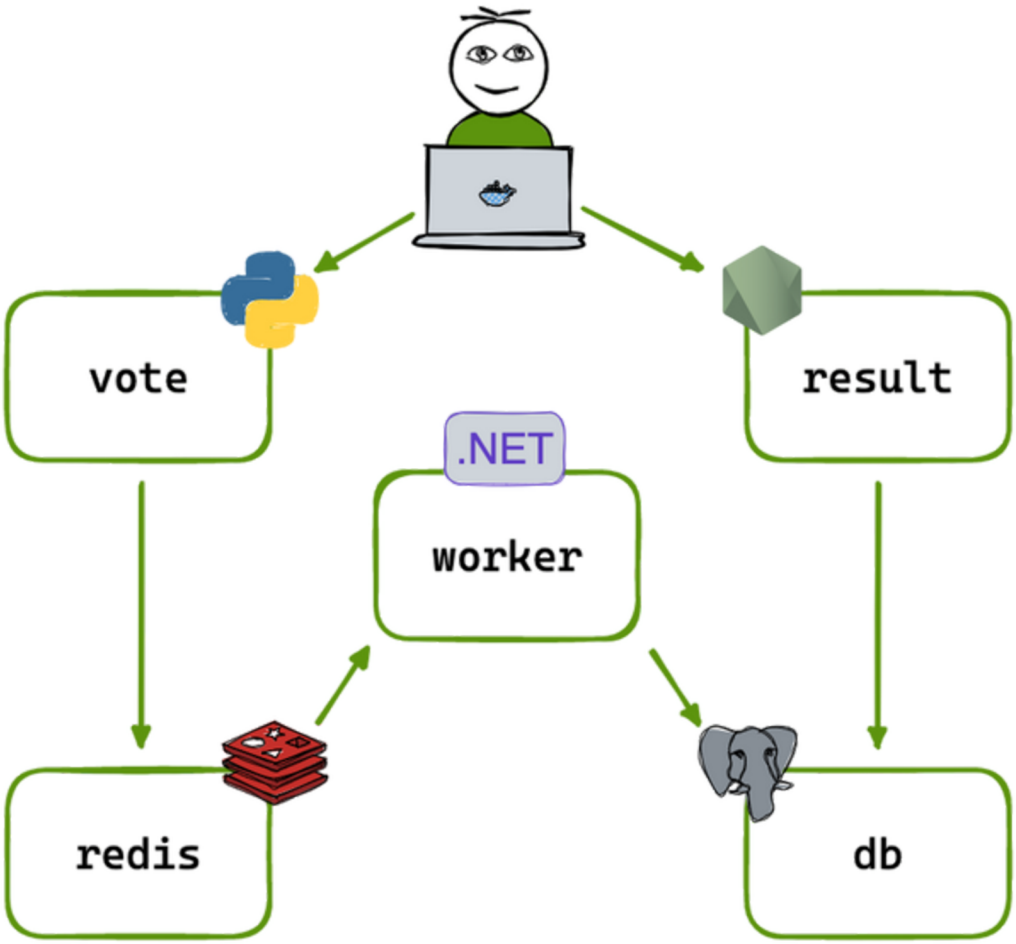
5-Tier Voting-Application Deployment Using Docker🐳

The Example Voting App is a microservices-based web application used to demonstrate Docker + containers



🏗️ Architecture (5 Tiers)

- 🐍 **Vote App (Python/Flask)** → Users vote (Cats 🐱 or Dogs 🐶).
- 💾 **Redis (In-memory DB)** → Stores votes temporarily.
- ⚙️ **Worker (.NET Service)** → Moves votes from Redis → Postgres.
- 🐘 **Postgres (SQL DB)** → Stores votes permanently in a volume.
- 🌐 **Result App (Node.js)** → Shows live voting results in browser.

Clone repo if not done:

- git clone https://github.com/dockersamples/example-voting-app.git
- cd example-voting-app

```
controlplane:~$ git clone https://github.com/dockersamples/example-voting-app.git
Cloning into 'example-voting-app'...
remote: Enumerating objects: 1179, done.
remote: Total 1179 (delta 0), reused 0 (delta 0), pack-reused 1179 (from 1)
Receiving objects: 100% (1179/1179), 1.21 MiB | 1.98 MiB/s, done.
Resolving deltas: 100% (449/449), done.
controlplane:~$ ls
example-voting-app  filesystem
controlplane:~$ cd example-voting-app/
controlplane:~/example-voting-app$ ls
LICENSE      README.md                 docker-compose.images.yml  docker-stack.yml  k8s-specifications  seed-data  worker
MAINTAINERS  architecture.excalidraw.png  docker-compose.yml         healthchecks      result              vote
controlplane:~/example-voting-app$
```

1) Create one Docker network (service discovery) 🌐

- docker network create votingapp-net

```
controlplane:~/example-voting-app$ docker network create votingapp-net
b24c5c438870d1842c88fad447381b4c2a903152bd0e517b84f49583f24b0fa6
controlplane:~/example-voting-app$
```

2) Create Postgres volume (persistence) 💾

- docker volume create db-data

```
controlplane:~$ docker volume create db-data
db-data
```

Where data lives: host path managed by Docker (e.g. /var/lib/docker/volumes/db-data/_data).

3) Start Redis (in-memory queue) 🐇

- **docker run -d --name redis --network votingapp-net redis:alpine**

```
controlplane:~$ docker run -d --name redis --network votingapp-net redis:alpine
Unable to find image 'redis:alpine' locally
alpine: Pulling from library/redis
9824c27679d3: Pull complete
9880d81ff87a: Pull complete
168694ef5d62: Pull complete
f8eab6d4856e: Pull complete
1f79dac8d2d4: Pull complete
4f4fb700ef54: Pull complete
61cfb50eeff3: Pull complete
Digest: sha256:987c376c727652f99625c7d205a1cba3cb2c53b92b0b62aade2bd48ee1593232
Status: Downloaded newer image for redis:alpine
a45804b883d9c31b2fabddd21cf2519d4e2cd4a122cf7194354dfb129cb4197f
controlplane:~$
```

4) Start Postgres (persistent DB) 🐘

**docker run -d --name db --network votingapp-net **
**-e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres **
**-v db-data:/var/lib/postgresql/data -p 5432:5432 **
postgres:15-alpine

```
controlplane:~$ docker run -d --name db --network votingapp-net \
  -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres \
  -v db-data:/var/lib/postgresql/data -p 5432:5432 \
  postgres:15-alpine
Unable to find image 'postgres:15-alpine' locally
15-alpine: Pulling from library/postgres
9824c27679d3: Already exists
45222bf54768: Pull complete
cc455ce33b7a: Pull complete
6e59367d2e76: Pull complete
a9186699d5f8: Pull complete
90793d70c9dc: Pull complete
d97eb1910934: Pull complete
8a4d728ab743: Pull complete
cb75953ddf8d: Pull complete
4c5335ab75e4: Pull complete
02c858cee38f: Pull complete
Digest: sha256:dfcf0459185089e88a43197975780f5a3078acd5ece84824a14c9d6fbbab02d0
Status: Downloaded newer image for postgres:15-alpine
0ff06087c8373b2c18f705f0f4160c9d9b399696c61354d41ca8ae7dba0974ff
controlplane:~$
```

5) Build service images (vote, result, worker) 🧱

Vote (Python/Flask):

- **docker build -t vote-img .**

```
controlplane:~/example-voting-app$ ls
LICENSE      README.md                  docker-compose.images.yml  docker-stack.yml  k8s-specific
MAINTAINERS  architecture.excalidraw.png  docker-compose.yml         healthchecks      result
controlplane:~/example-voting-app$ cd vote/
controlplane:~/example-voting-app/vote$ ls
Dockerfile  app.py  requirements.txt  static  templates
controlplane:~/example-voting-app/vote$ docker build -t vote-img .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon   12.8kB
Step 1/13 : FROM python:3.11-slim AS base
3.11-slim: Pulling from library/python
ce1261c6d567: Pull complete
11b89692b208: Pull complete
764e05fe66b6: Pull complete
a4aefcec16c5: Pull complete
```

Result (Node.js):

- **docker build -t result-img .**

```
controlplane:~/example-voting-app$ ls
LICENSE      README.md                  docker-compose.images.yml  docker-stack.yml  k8s-specifications  seed-data  worker
MAINTAINERS  architecture.excalidraw.png  docker-compose.yml         healthchecks      result              vote
controlplane:~/example-voting-app$ cd result/
controlplane:~/example-voting-app/result$ ls
Dockerfile  docker-compose.test.yml  package-lock.json  package.json  server.js  tests  views
controlplane:~/example-voting-app/result$ docker build -t result-img .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  291.3kB
Step 1/11 : FROM node:18-slim
18-slim: Pulling from library/node
61320b01ae5e: Pull complete
b98d3ae1ab80: Pull complete
b1831021e35a: Pull complete
c768ab8cba73: Pull complete
8c994cf49dd1: Pull complete
Digest: sha256:f9ab18e354e6855ae56ef2b290dd225c1e51a564f87584b9bd21dd651838830e
Status: Downloaded newer image for node:18-slim
 ---> 101e0128c8ea
Step 2/11 : RUN apt-get update &&     apt-get install -y --no-install-recommends curl tini &&     rm -rf /var/lib/apt/lists/*
 ---> Running in 215ecbd3014d
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8791 kB]
```

Worker (.NET) :

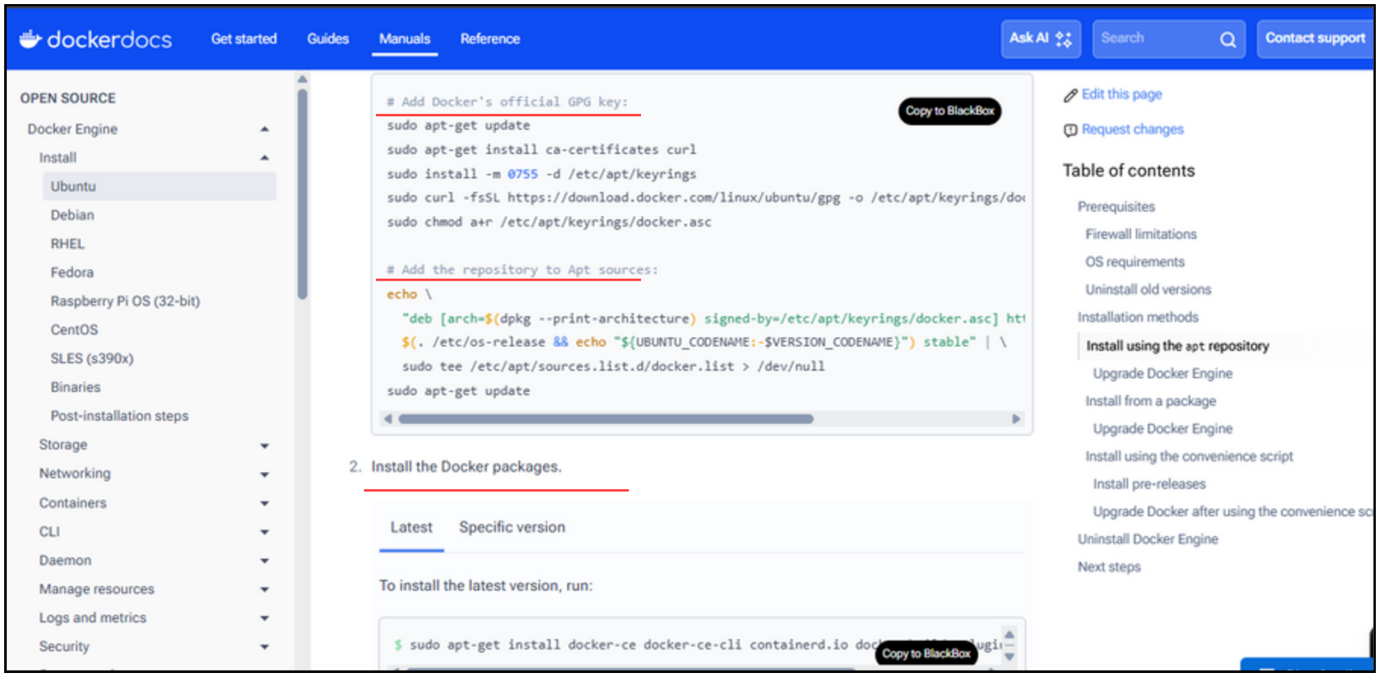
- **docker build -t worker-img .**

```
controlplane:~/example-voting-app$ ls
LICENSE      README.md                docker-compose.images.yml  docker-stack.yml  k8s-specifications  seed-data  worker
MAINTAINERS  architecture.excalidraw.png  docker-compose.yml         healthchecks      result              vote
controlplane:~/example-voting-app$ cd worker/
controlplane:~/example-voting-app/worker$ ls
Dockerfile  Program.cs  Worker.csproj
controlplane:~/example-voting-app/worker$ docker build -t worker-img .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  10.24kB
Step 1/14 : FROM --platform=${BUILDPLATFORM} mcr.microsoft.com/dotnet/sdk:7.0 AS build
failed to parse platform : "" is an invalid OS component of "": OSAndVersion specifier component must match "^([A-Za-z0-9_-]+)(?:\\(([A-Za-z0-9_.-]*)\\))?$": invalid
argument
controlplane:~/example-voting-app/worker$
```

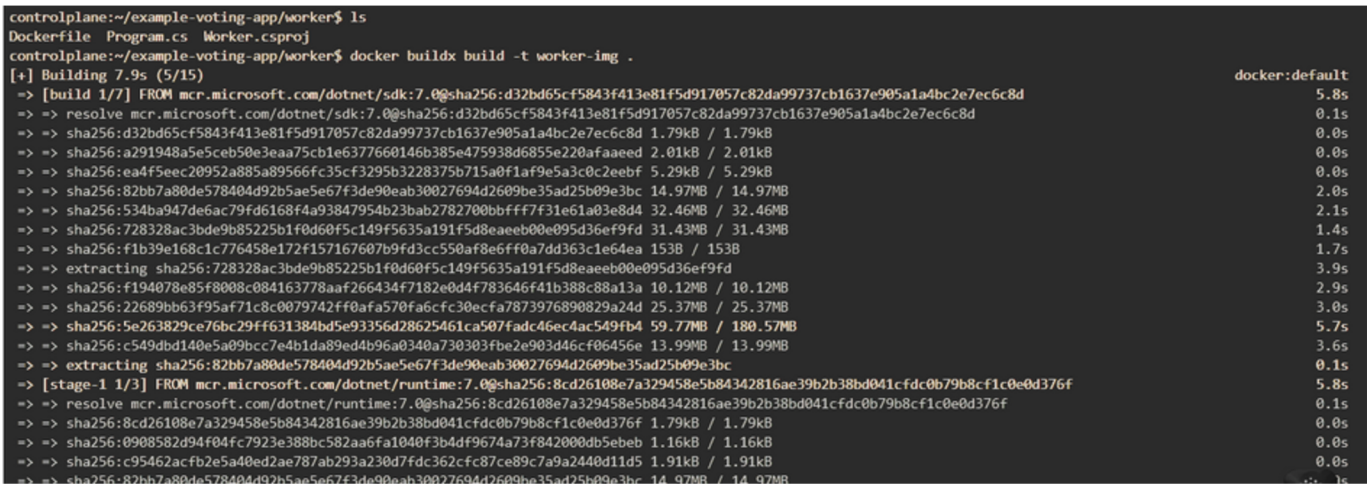
Gives this error --platform=${BUILDPLATFORM}

So, Use the buildx for the avoid this error



Download Buildx from the docker offiicial documentations

- **docker buildx build -t worker-img .**



images package runtime + code so containers run the same everywhere.

6) Run the app containers (explicit envs so they can connect) 🔌

Vote (frontend) — port 8080 on host:

```
docker run -d --name vote --network votingapp-net -p 8080:80 \
  -e REDIS_HOST=redis -e REDIS_PORT=6379 \
  -e POSTGRES_HOST=db -e POSTGRES_PORT=5432 \
  vote-img
```

```
controlplane:~$ docker run -d --name vote --network votingapp-net -p 8080:80 \
  -e REDIS_HOST=redis -e REDIS_PORT=6379 \
  -e POSTGRES_HOST=db -e POSTGRES_PORT=5432 \
  vote-img
987bdf92baa314a48014b04629ded2f446ddb55b5f1e77f1dcfcb5913e49f933
```

Result (frontend) — port 8081 on host

```
docker run -d --name result --network votingapp-net -p 8081:80 \
 -e POSTGRES_HOST=db -e POSTGRES_PORT=5432 \
 -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres \
 result-img
```

```
controlplane:~$ docker run -d --name result --network votingapp-net -p 8081:80 \
  -e POSTGRES_HOST=db -e POSTGRES_PORT=5432 \
  -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres \
  result-img
df53c4272419542a6746ab4fe7d17d1c33fa32a2fe264d3c31b8ffc3e21cb3dc
```

Worker (background processor) — no host port needed:

```
docker run -d --name worker --network myapp-net \
 -e REDIS_HOST=redis -e REDIS_PORT=6379 \
 -e POSTGRES_HOST=db -e POSTGRES_PORT=5432 \
 -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres \
 worker-img
```

```
controlplane:~$ docker run -d --name worker --network votingapp-net \
  -e REDIS_HOST=redis -e REDIS_PORT=6379 \
  -e POSTGRES_HOST=db -e POSTGRES_PORT=5432 \
  -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres \
  worker-img
2f2ffe47709df754d69a8b03367498dc895b35bef294324be22ad8e06cab9625
controlplane:~$
```

7) Quick status checks ✅

List running containers:

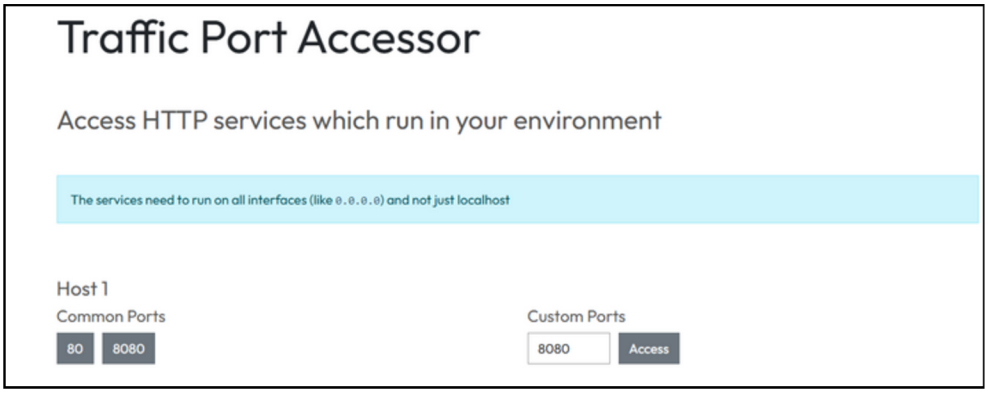
- docker ps

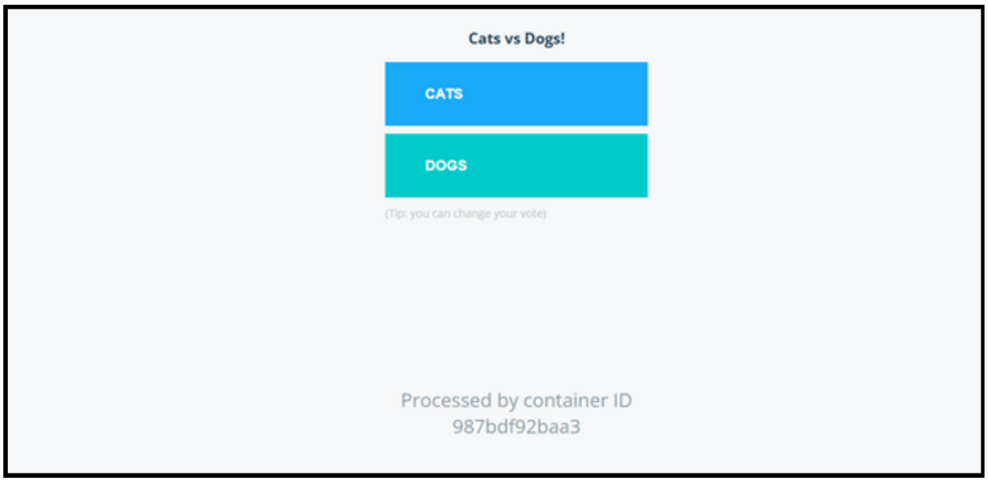
```
controlplane:~$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED              STATUS              PORTS                                       NAMES
2f2ffe47709d   worker-img           "dotnet Worker.dll"      About a minute ago   Up About a minute                                               worker
df53c4272419   result-img           "/usr/bin/tini -- no…"   About a minute ago   Up About a minute   0.0.0.0:8081->80/tcp, [::]:8081->80/tcp     result
987bdf92baa3   vote-img             "gunicorn app:app -b…"   2 minutes ago        Up 2 minutes        0.0.0.0:8080->80/tcp, [::]:8080->80/tcp     vote
0ff06087c837   postgres:15-alpine   "docker-entrypoint.s…"   16 minutes ago       Up 3 seconds        0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp db
a45804b883d9   redis:alpine         "docker-entrypoint.s…"   17 minutes ago       Up 3 seconds        6379/tcp                                    redis
controlplane:~$
```

8) Verify

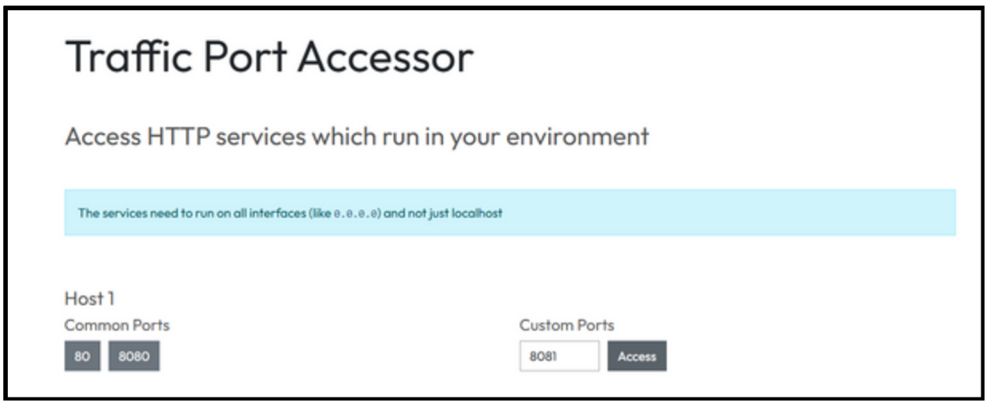
Now you can access:

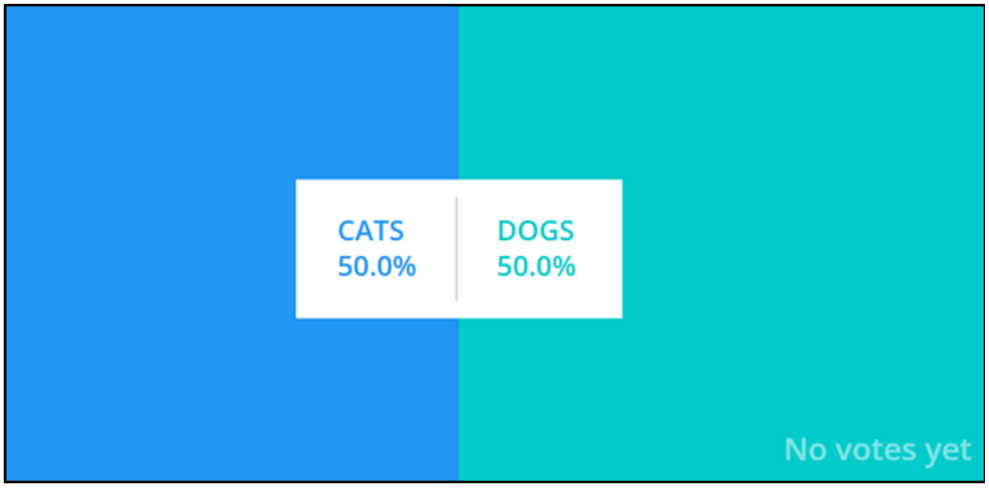
- **Voting app: http://localhost:8080**



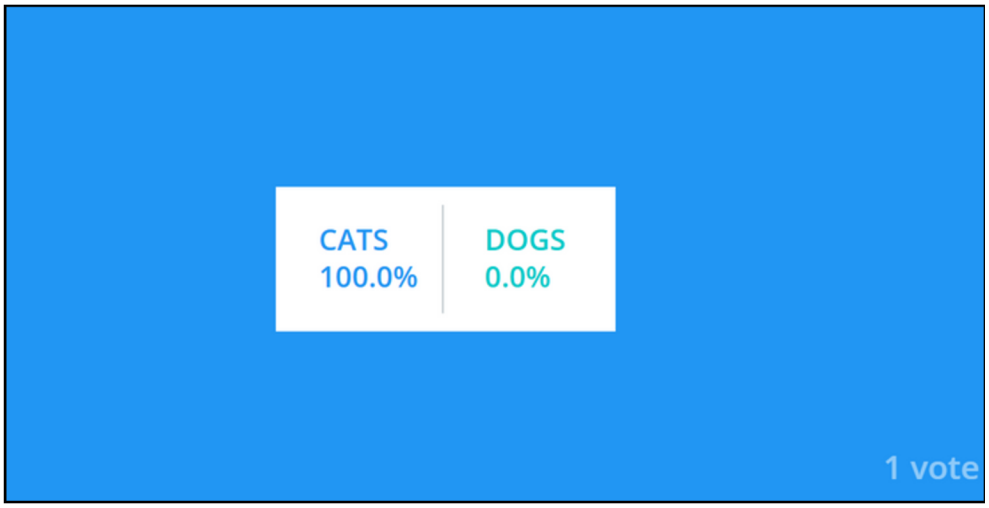


- **Results app: http://localhost:8081**





- **When I vote Cat the result is change :**



✓ **Built a 5-tier microservices application**
✓ **Deployed manually using Docker CLI (no Compose/K8s)**
✓ **Configured networking + volumes**
✓ **Validated persistence with Postgres DB**
✓ **Demonstrated real-time processing of data**