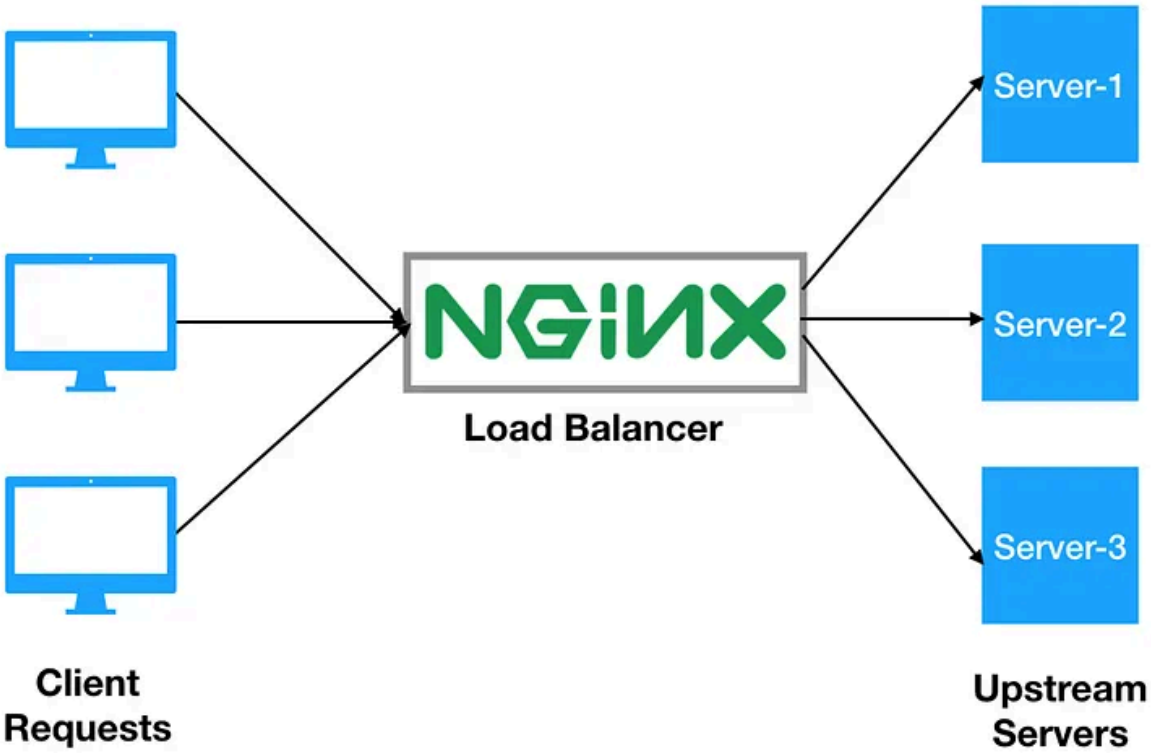
Microservices: Scaling and Load Balancing using docker compose



Scaling and Load Balancing using docker compose

In this post, we will look at an example of how to run multiple instances of a service and perform load balancing with Docker Compose.

For this post, we will work with the following example Docker Compose file:

```
version: '3.5'
```

```
services:
  petstore:
    image: petstore:latest
    hostname: petstore01
    ports:
      - "5000:5000"
    environment:
      - CONFIG_FILE=/config/app1/petstore.json
      - ENVIRONMENT=staging
   - USER: admin
      - PASSWORD: password11
    volumes:
      - ./conf:/conf

networks:
  default:
    driver: bridge
    name: petstore
```

In this file, I have defined petstore service that configures petstore application. I have defined the port mappings for the Docker container.
For port mappings, Docker Compose uses the HOST:CONTAINER format. In our example, we expose the port 5000 from the container to the same port on the host machine.

To start the service run following command.

```
docker-compose up -d
```

We can access the server with the http://localhost:5000 URL on host machine.

Scaling up services

Each service defined in Docker compose configuration can be scaled using following command.
docker-compose scale <service name> = <no of instances>

For example :

```
docker-compose scale petstore=2
```

The problem with our current configuration is that we are trying to run two instances of the petstore service and map them all to the same port on our host machine. The host machine can only bind an unallocated port to the container, we will get the Bind for 0.0.0.0:5000 failed: port is already allocated error for additional petstore service.

The solution to fix this issue is to change line 8 in Docker Compose file to — “5000”. With this configuration, the scaled services will bind on different ports.

Load Balancing

We will define a load balancer to the system configuration. The load balancer will access the petstore service without exposing the container ports and distribute the traffic across the containers.

In this example, we will use NGINX as the load balancer. To add the load balancer to our Docker Compose system configuration, we create the following nginx.conf file in the same directory where the docker-compose.yml file exist.

```
user  nginx;

events {
    worker_connections   1000;
}
http {
        server {
              listen 4000;
              location / {
                proxy_pass http://petstore:5000;
              }
        }
}
```

This will configure nginx to forward the request from port 4000 to http://petstore:5000. Docker embedded DNS server will resolve name to IP address. nginx will use a round robin implementation to distribute the traffic across the services.

The nginx service will handle the requests and forward them to a petstore service, we don’t need to map the port 5000 from the petstore services to a host machine port. Now we can remove the port mapping configuration from our Docker Compose file and only expose the port 5000 to link the services.

For nginx configuration file we just created, we have to mount it as a volume in the nginx service and add port mappings to the host container for that server. In this example, we configured NGINX to listen on the port 4000, which is why we have to add port mappings for this port.

All set! This is what our final docker-compose.yml file looks like.

```
version: '3.5'
services:
  petstore:
    image: petstore:latest
    hostname: petstore01
    expose:
      - "5000"
    environment:
      - CONFIG_FILE=/config/app1/petstore.json
      - ENVIRONMENT=staging
   - USER: admin
      - PASSWORD: password11
    volumes:
      - ./conf:/conf

nginx:
    image: nginx:latest
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on:
      - petstore
    ports:
      - "4000:4000"

networks:
  default:
    driver: bridge
    name: petstore
```

Now start multiple instances of the petstore service by setting the scale parameter of the Docker Compose command to the number of services we want to start.

For example:

```
docker-compose up --scale petstore=3
```

The above command will start three instances of petstore application, which can be accessed at http://localhost:4000. The requests to this URL will get load balanced and distributed to one of the five petstore docker containers.

Load Balancer | Nginx | Scaling | Docker Compose | Docker

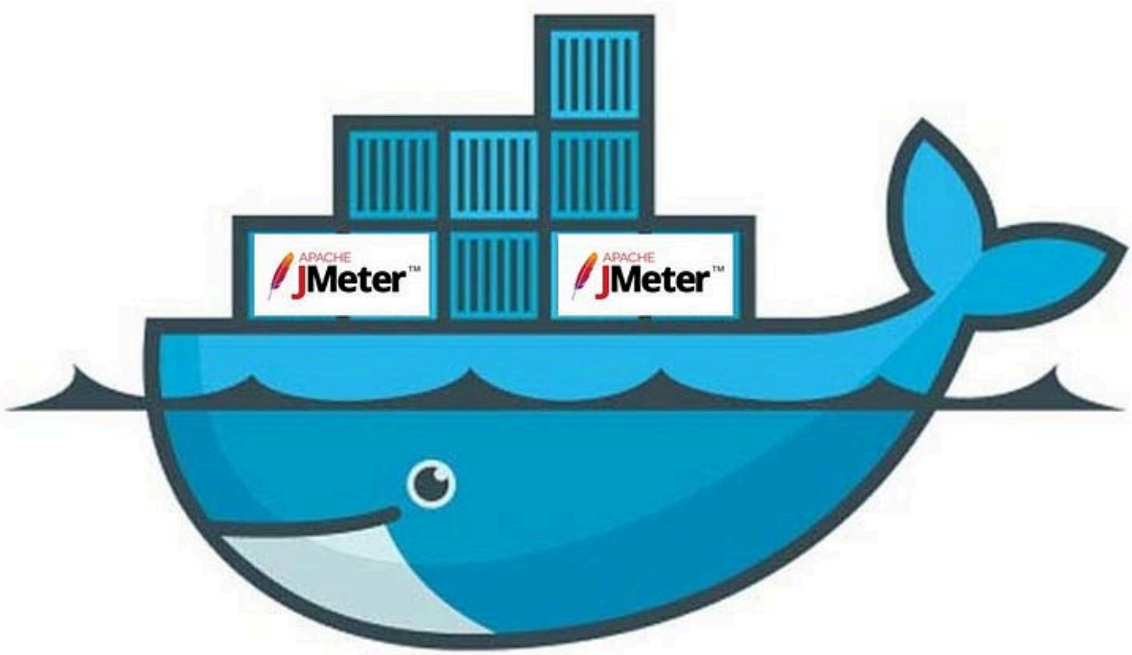
Follow

Written by Vinod Rane

21 Followers

Hi Everyone, and welcome to my medium profile! I'm Vinod Rane, software professional with a passion for building rock-solid test automation frameworks.

More from Vinod Rane

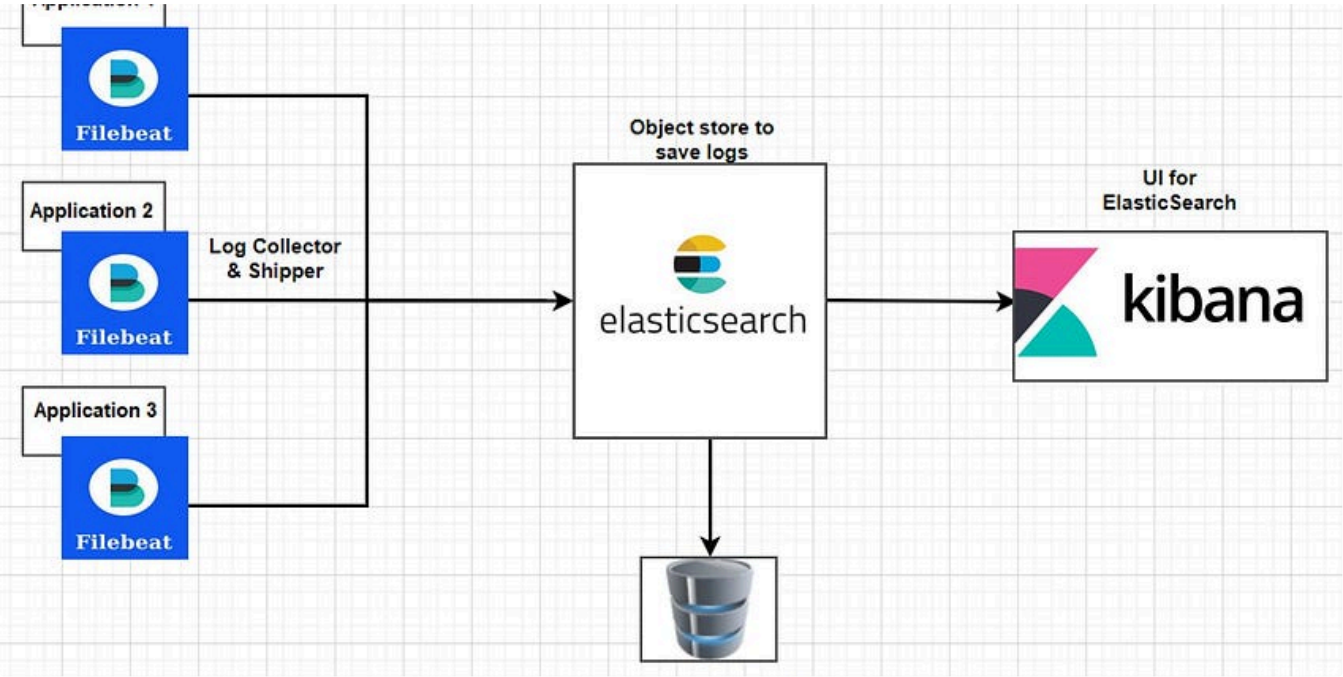


Vinod Rane

How to Use the Docker Container with JMeter

To get started with dockerized JMeter, make sure you meet the prerequisites.

Jul 19, 2020 54 2



Vinod Rane

Deploy an EFK stack to Kubernetes

Introduction to EFK stack.

Dec 28, 2021 4

Vinod Rane

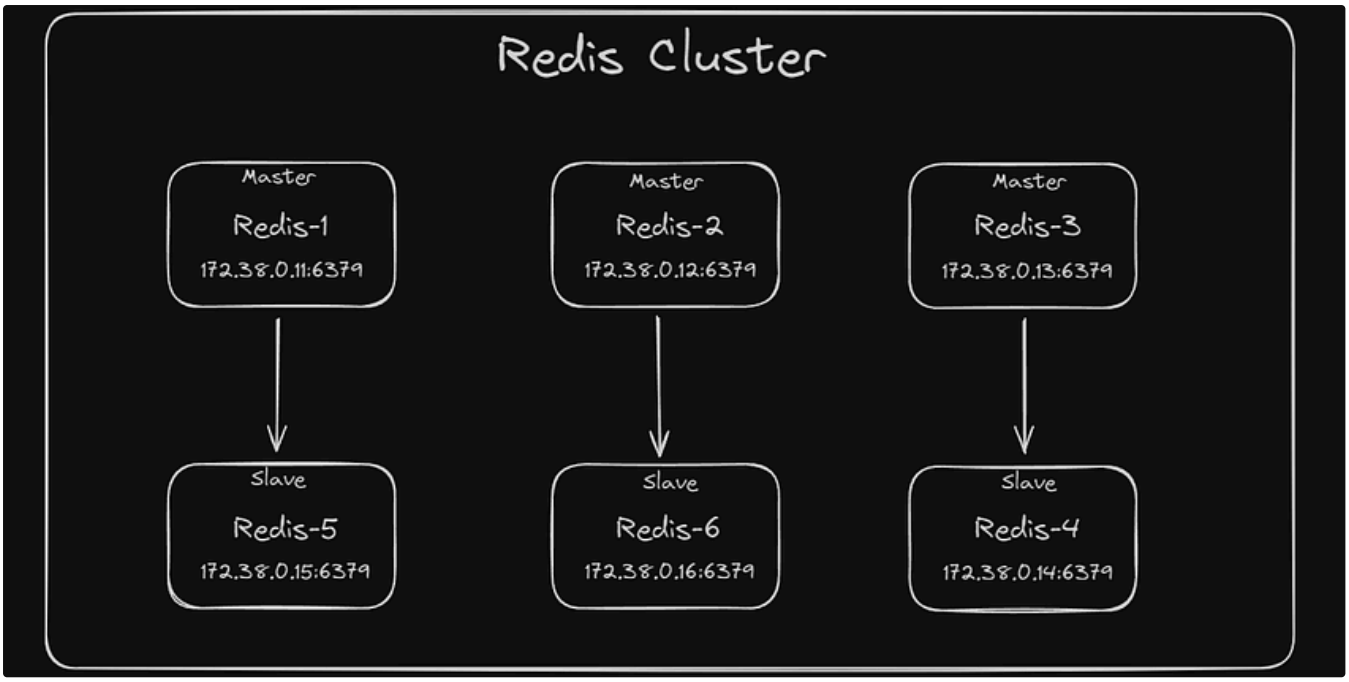
How to choose a test automation testing tool?

Each framework in the market is neither entirely good nor bad, everything depends on your particular project, solution, and software...

Oct 12, 2020 13

See all from Vinod Rane

Recommended from Medium

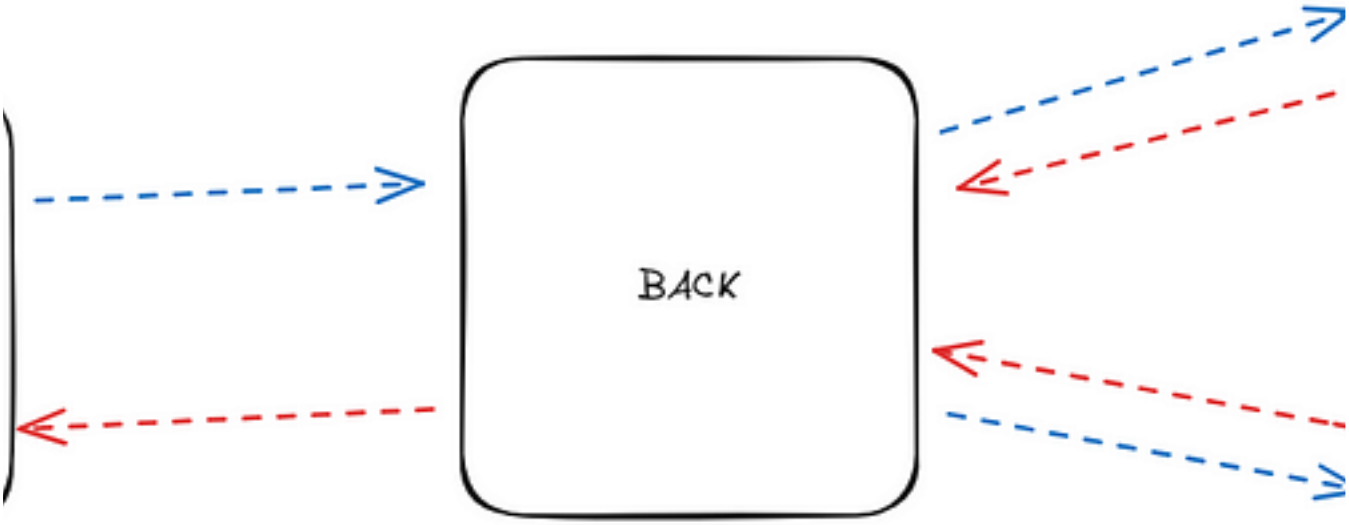


Lim Yee Jie

Basic Docker Compose and Build a Redis Cluster with Docker Compose

Docker Compose

Jan 23 30 1



Marius NIEMET

Containerize your multi-services app with docker compose

This article was posted first on my personal blog mariusniemet.me

Mar 28

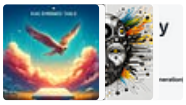
Lists

Coding & Development

11 stories · 637 saves

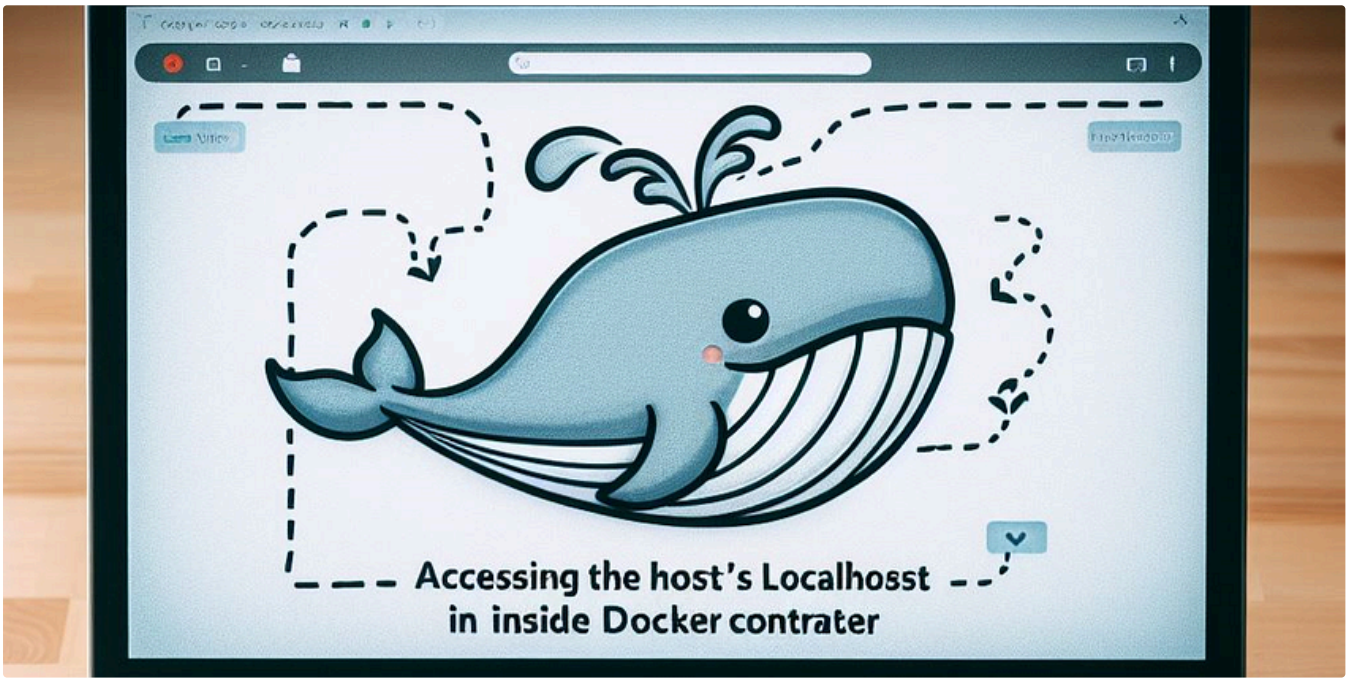
Staff Picks

654 stories · 1015 saves



Natural Language Processing

1488 stories · 999 saves

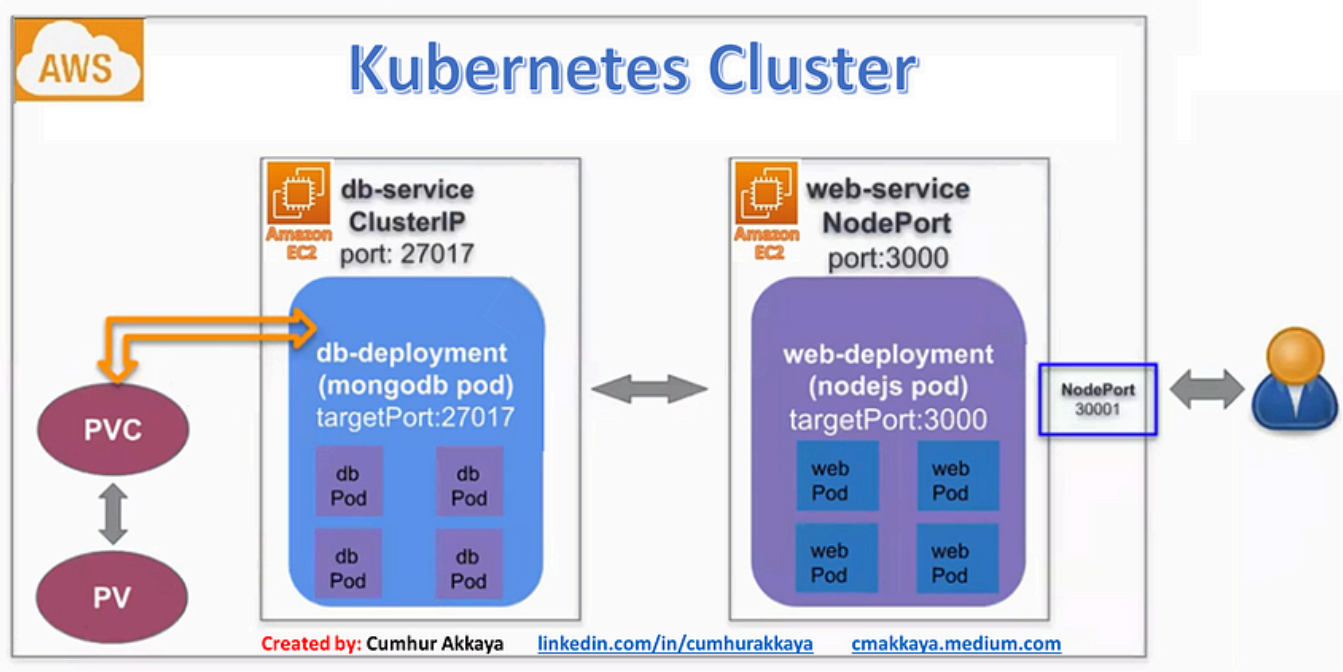


gladevise

Accessing the host's localhost from inside a Docker container

In this article, I will show how to access a server on the host's localhost from inside a Docker container by specifying ` —add-host...

Dec 25, 2023 103

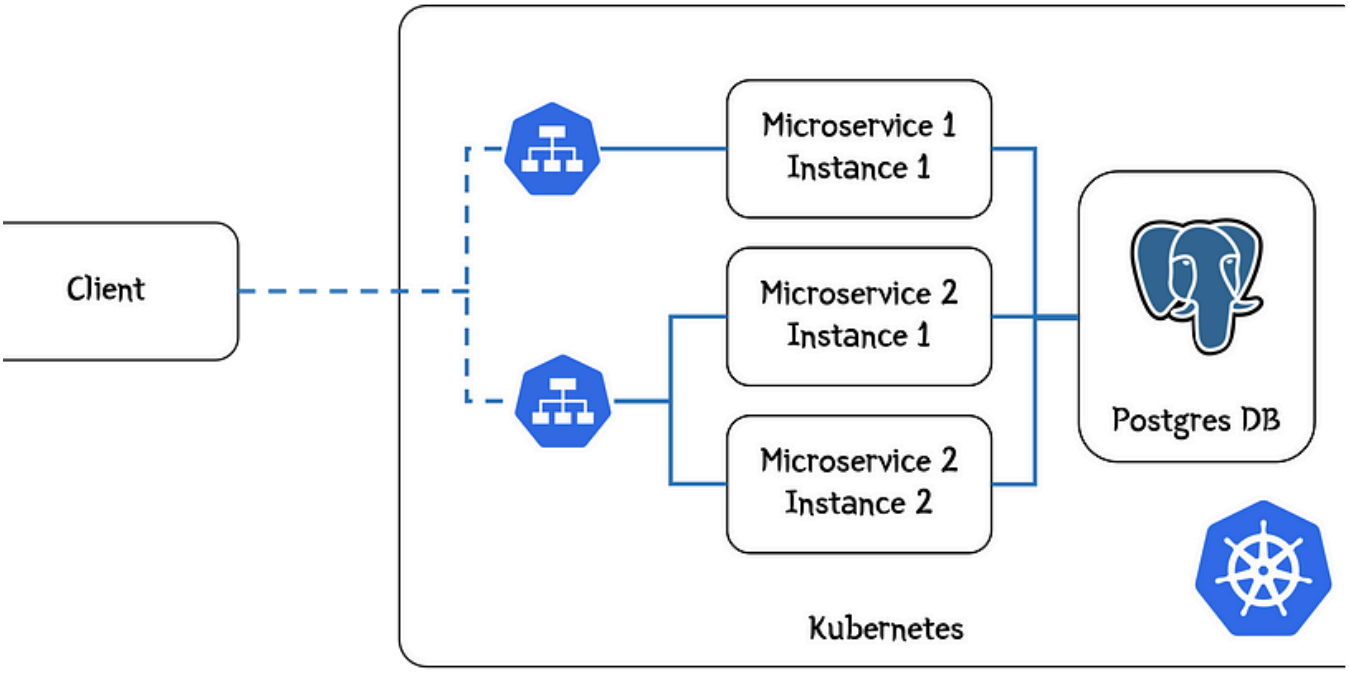


Cumhur Akkaya

Diving into Kubernetes-3: Running a MongoDB and Nodejs App together in the Kubernetes cluster Using...

In this article, we will learn about Kubernetes Volume and Networking, which are important topics of Kubernetes, and we will make a...

Dec 18, 2023 68

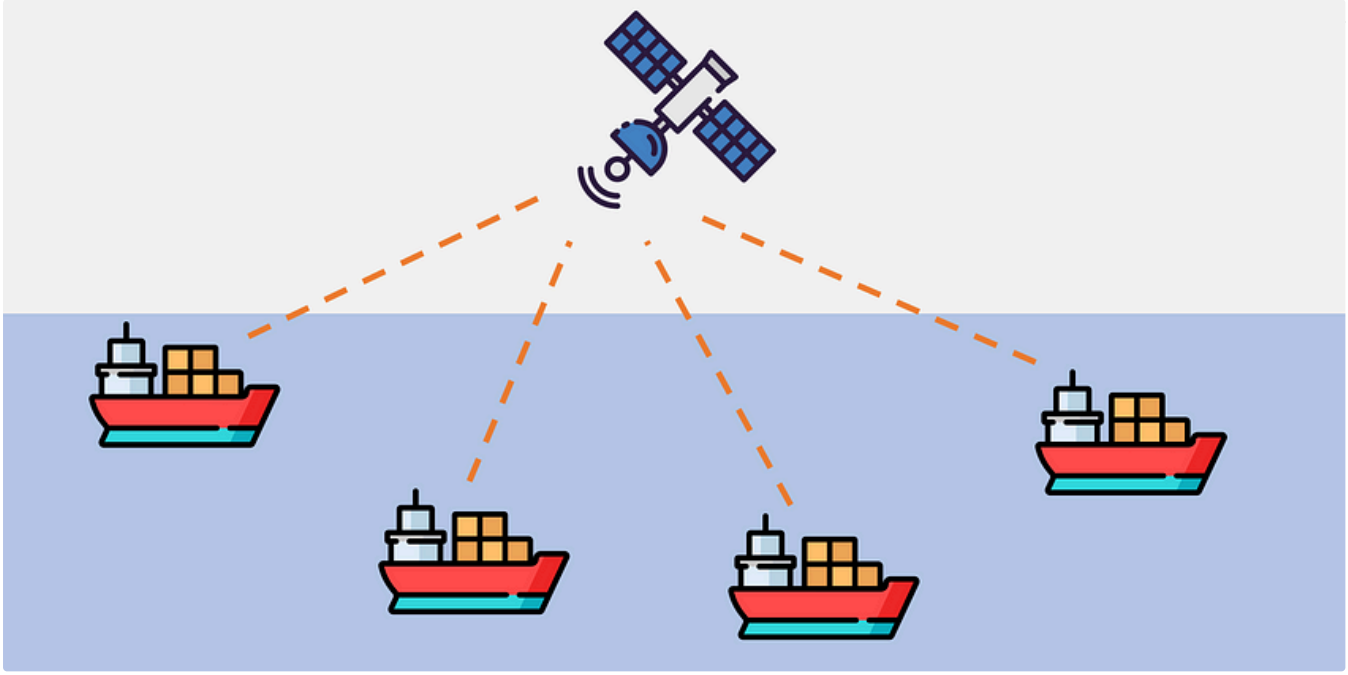


Martin Hodges

Adding a Postgres High-Availability database to your Kubernetes cluster

In this article I describe the process for installing a Postgres database in a High Availability (HA) configuration into your Kubernetes...

Jan 26 157 3



Ahmet Tuncer

Redis Cluster using Docker

We will talk about creating a redis cluster using docker, especially on Windows.

Dec 11, 2023 9 1

See more recommendations