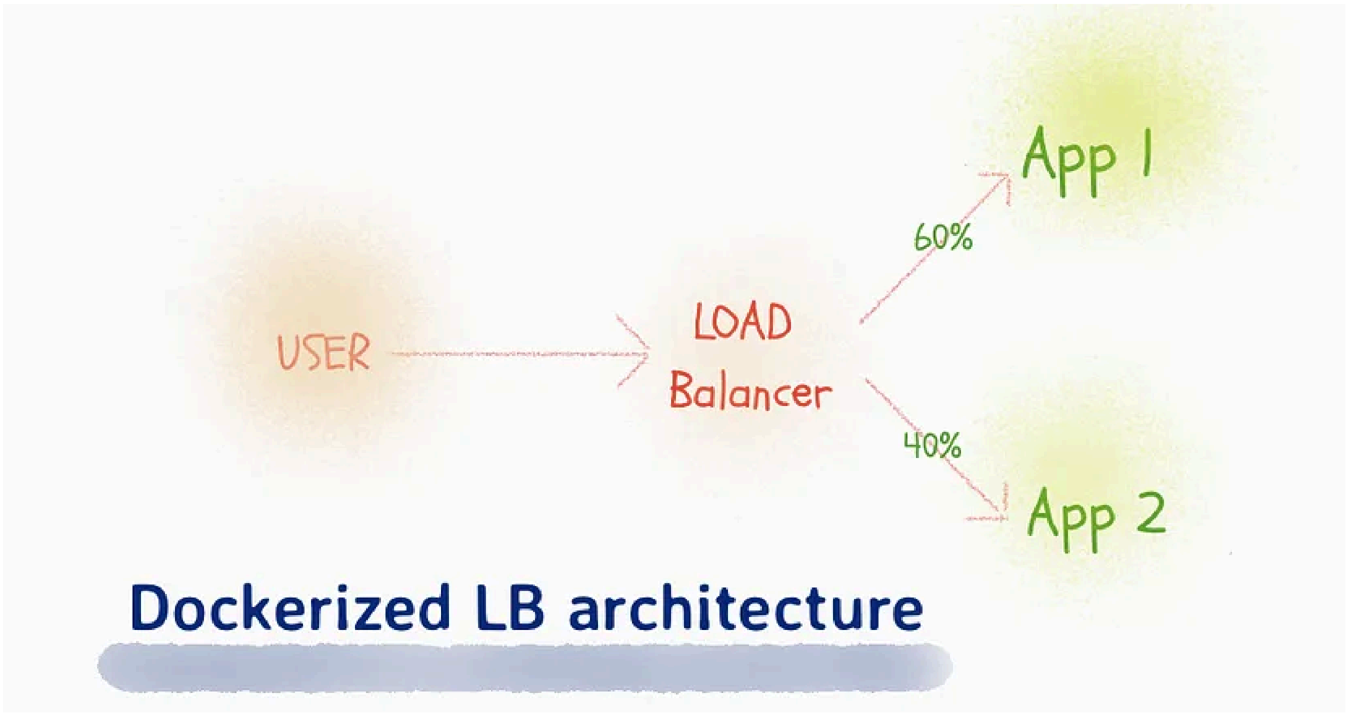
Open in app ↗ Sign up Sign in

Medium Search

Abdelilah OUASSINI · Follow

3 min read · Mar 25, 2020

Listen Share



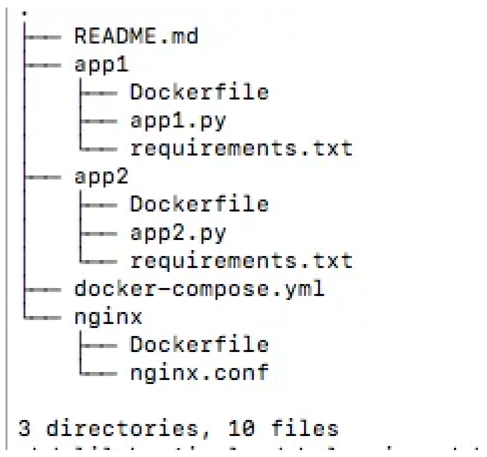
Most of today's business applications use load balancing to distribute traffic among different resources and avoid overload of a single resource.

One of the obvious advantages of load balancing architecture is to increase the availability and reliability of applications, so if a certain number of clients request some number of resources to backends, Load balancer stays between them and route the traffic to the backend that fills most the routing criteria (less busy, most healthy, located in a given region .. etc).

There are a lot of routing criteria, but we will focus on this article on fixed round-robin criteria — meaning each backend receives a fixed amount of traffic — which I think rarely documented :).

To simplify we will create two backends "applications" based on flask Python files. We will use NGINX as a load balancer to distribute 60% of traffic to application1 and 40% of traffic to application2.

Let's start the coding, hereafter the complete architecture of our project:



```
.
├── README.md
├── app1
│   ├── Dockerfile
│   ├── app1.py
│   └── requirements.txt
├── app2
│   ├── Dockerfile
│   ├── app2.py
│   └── requirements.txt
├── docker-compose.yml
└── nginx
    ├── Dockerfile
    └── nginx.conf

3 directories, 10 files
```

Load balancing Project tree

app1/app1.py

```
from flask import request, Flask
import json

app1 = Flask(__name__)
@app1.route('/')

def hello_world():
return 'Salam alikom, this is App1 :) '

if __name__ == '__main__':
app1.run(debug=True, host='0.0.0.0')
```

app2/app2.py

```
from flask import request, Flask
import json

app1 = Flask(__name__)
@app1.route('/')

def hello_world():
return 'Salam alikom, this is App2 :) '

if __name__ == '__main__':
app1.run(debug=True, host='0.0.0.0')
```

Then we have to dockerize both applications by adding the requirements.txt file. It will contain only the flask library since we are using the python3 image.

app1/requirements.txt

You may create the same for app2.

```
Flask==1.1.1
```

app1/Dockerfile

You may create the same for app2.

```
FROM python:3
COPY ./requirements.txt /requirements.txt
WORKDIR /
RUN pip install -r requirements.txt
COPY . /
ENTRYPOINT [ "python3" ]
CMD [ "app1.py" ]
```

We will use NGINX as a load balancer, the routing criteria will be guaranteed by the round-robin **weight** parameter :

nginx/nginx.conf

```
upstream loadbalancer {
server 172.17.0.1:5001 weight=6;
server 172.17.0.1:5002 weight=4;
}
server {
location / {
proxy_pass http://loadbalancer;
}}
```

We will then dockerize our load balancer by creating a Dockerfile using the Nginx image. It will copy the above conf file on the related path inside the container when starting it.

nginx/Dockerfile

```
FROM nginx
RUN rm /etc/nginx/conf.d/default.conf
COPY nginx.conf /etc/nginx/conf.d/default.conf
```

Now we will create the docker-compose file which will spin-up our complete architecture so we can access it from the browser.

docker-compose.yml

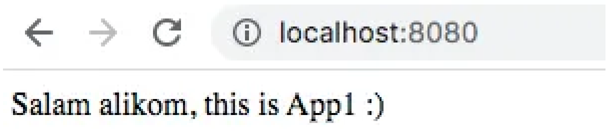
```
version: '3'
services:
app1:
build: ./app1
ports:
- "5001:5000"
app2:
build: ./app2
ports:
- "5002:5000"
nginx:
build: ./nginx
ports:
- "8080:80"
depends_on:
- app1
- app2
```

Some important points regarding docker-compose.yml:

- It will build images for app1, app2, Nginx based on our Dockerfiles and then spin up containers from those images.
- The opened port inside app1 and app2 containers are 5000 (default port used by flask), these ports will be mapped to 5001 and 5002.
  The load balancer will route traffic to the appropriate application based on that port.
- The load balancer (Nginx) will expose his internal 80 port to 8080, so we can access the application from http://localhost:8080

Finally, we will spin up our architecture by running below command and enjoy the result in the browser :)

```
docker-compose up
```



```
Salam alikom, this is App1 :)
```

You may also find the complete code on the below GitLab repo:

Abdelilah OUASSINI / load-balancing
GitLab.com
gitlab.com

As always, I hope you have learned something new. Salam :)

Load Balancing · Nginx · Docker Load Balancing · Flask Load Balancing · Python Flask

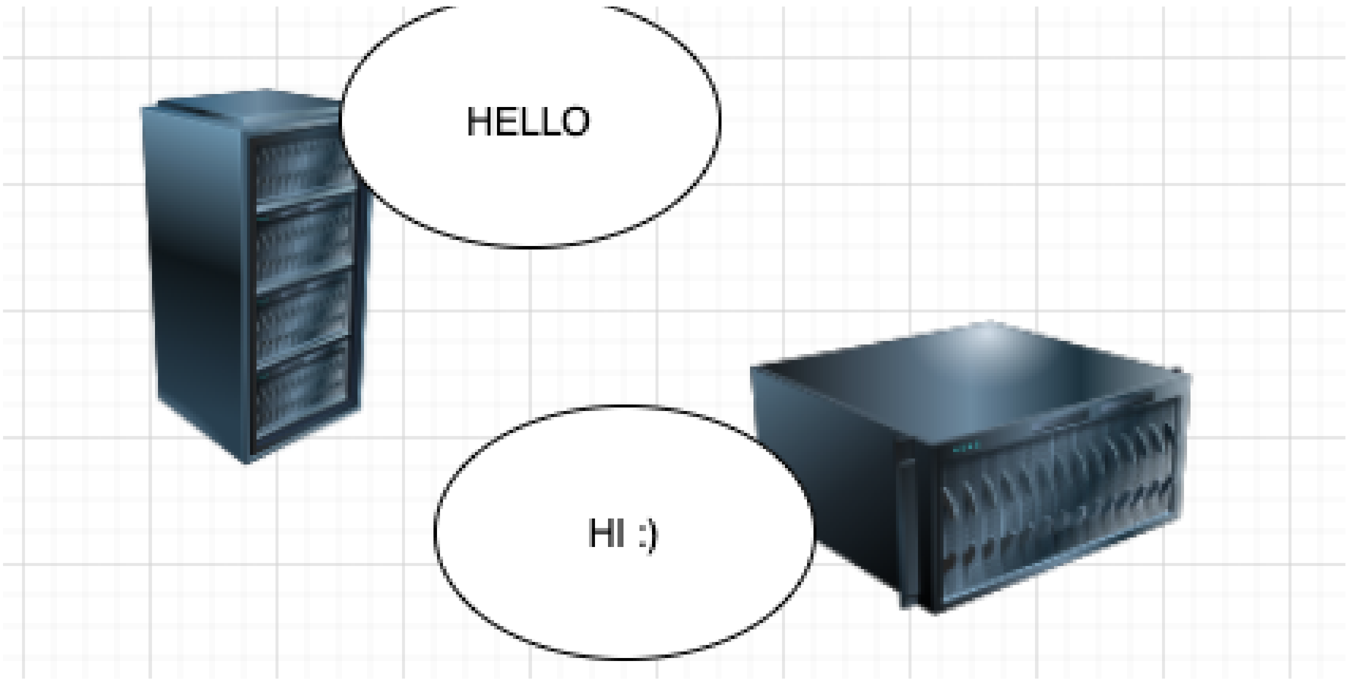
Follow

Written by Abdelilah OUASSINI

62 Followers

DevOps/Data scientist — Rabat

More from Abdelilah OUASSINI

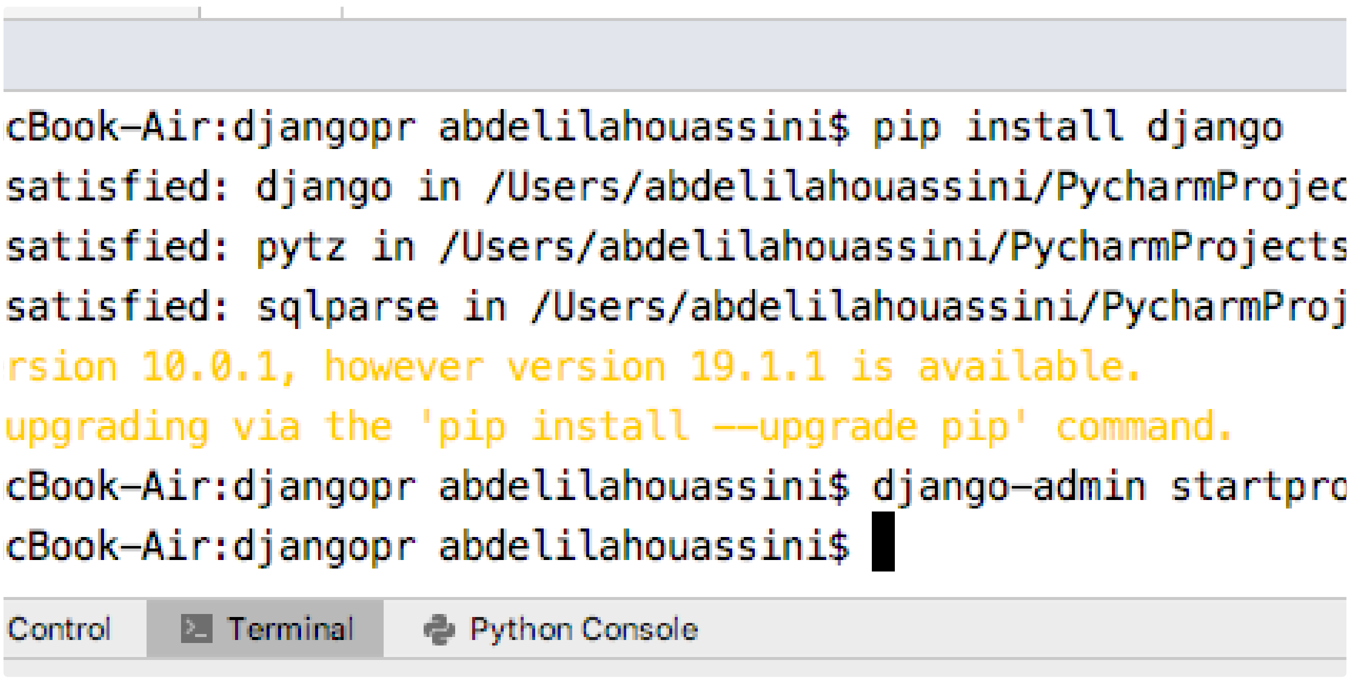


Abdelilah OUASSINI in Towards Data Science

Ansible—Ports Monitoring.

Today I will show you how Ansible can simplify hard technical challenges that we may face during monitoring inter-servers network...

Jun 17, 2020 5 1



```
cBook-Air:djangopr abdelilahouassini$ pip install django
satisfied: django in /Users/abdelilahouassini/PycharmProjec
satisfied: pytz in /Users/abdelilahouassini/PycharmProjects
satisfied: sqlparse in /Users/abdelilahouassini/PycharmProj
rsion 10.0.1, however version 19.1.1 is available.
upgrading via the 'pip install --upgrade pip' command.
cBook-Air:djangopr abdelilahouassini$ django-admin startpro
cBook-Air:djangopr abdelilahouassini$
```

Abdelilah OUASSINI in Towards Data Science

How to contain your first Django in docker and access it from AWS

Today a lot of junior software engineering Jobs require a tech stack including Django, python, docker, and AWS. If you are a beginner and...

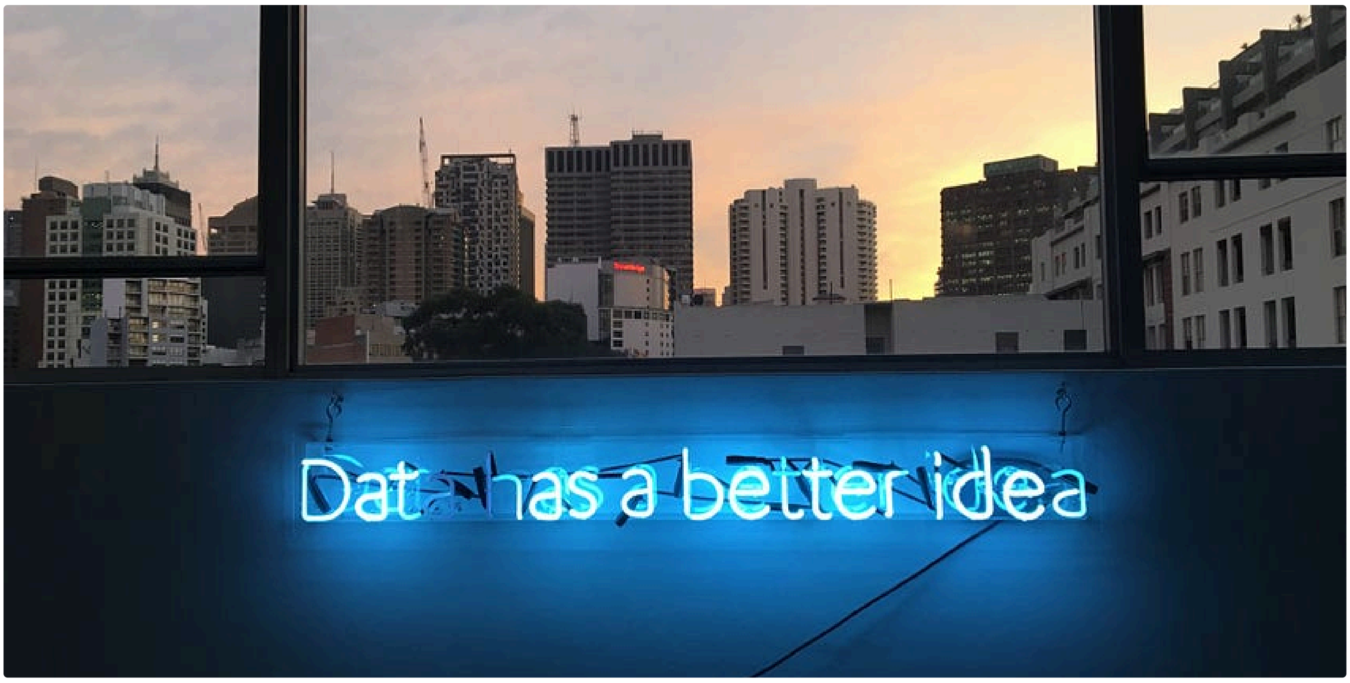
Jul 8, 2019 186 4

Abdelilah OUASSINI in Towards Data Science

How to put your machine learning in a Docker

You are a data scientist, you understand the business issue, you collect data, perform some feature engineering, and came up with an...

Jul 19, 2019 89 1



Abdelilah OUASSINI in Towards Data Science

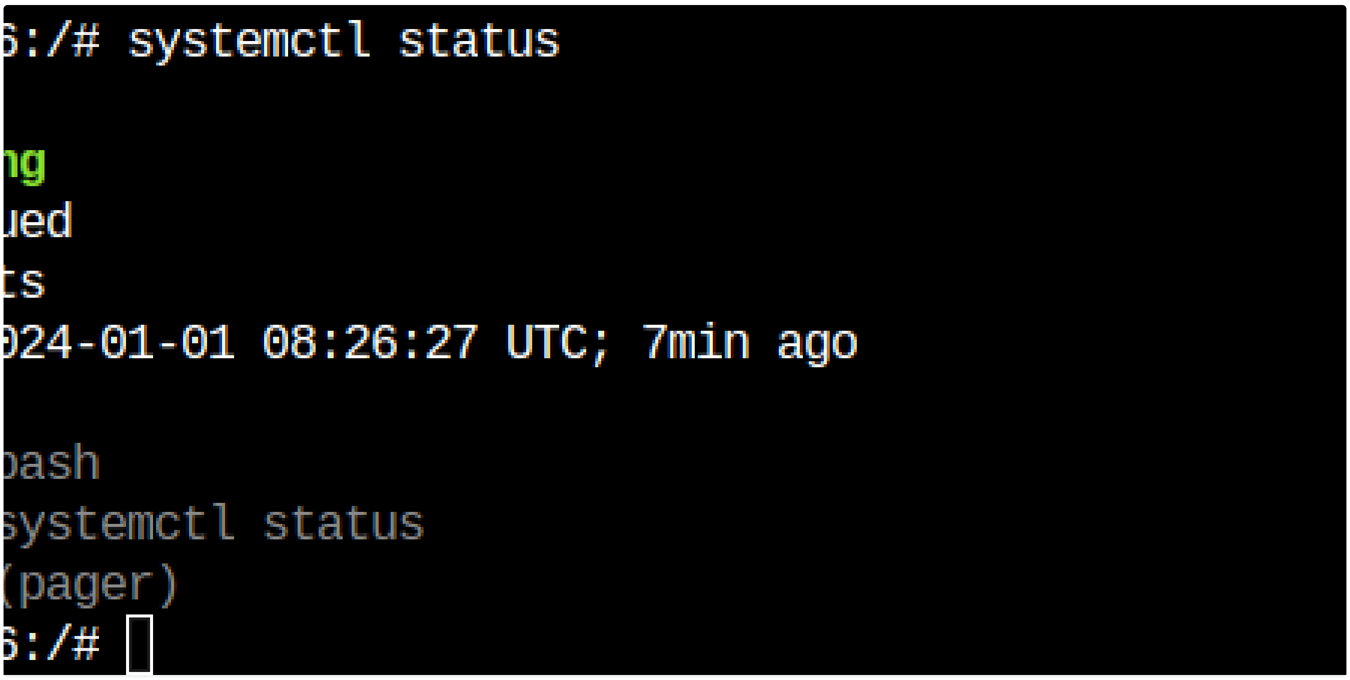
Reducing Data Inconsistencies with POI Normalization

Data normalization is an elegant technique that reduces data inconsistency. Especially when we are dealing with a huge dataset. I will...

Jul 13, 2019 20

See all from Abdelilah OUASSINI

Recommended from Medium



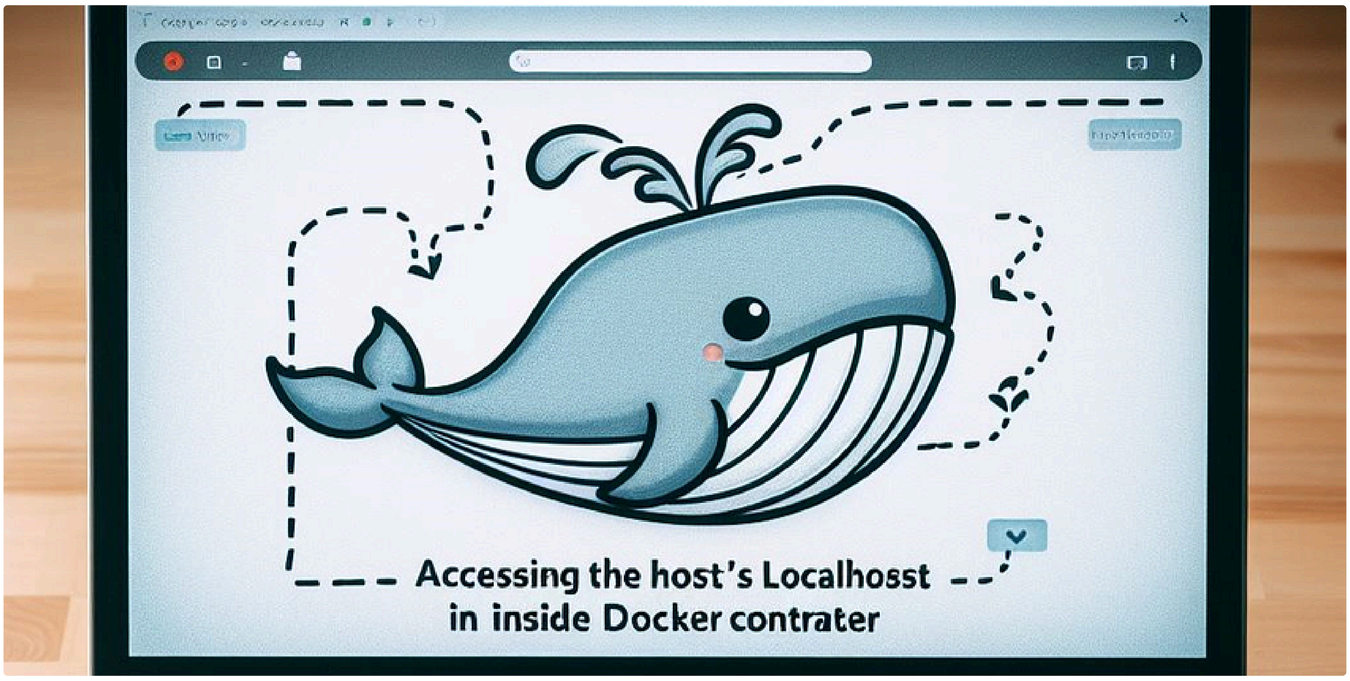
```
:/# systemctl status

ng
ued
ts
024-01-01 08:26:27 UTC; 7min ago

bash
systemctl status
(pager)
:/# 
```

Prateek Srivastava in DevOps.dev

Running Systemctl Inside Docker Container: A Comprehensive Guide

Introduction:

Jan 1 6



gladevise

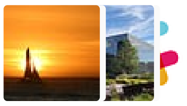
Accessing the host's localhost from inside a Docker container

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…

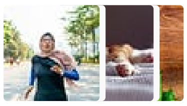
Dec 25, 2023 103

Lists

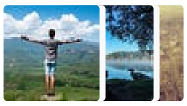
Staff Picks
654 stories · 1015 saves



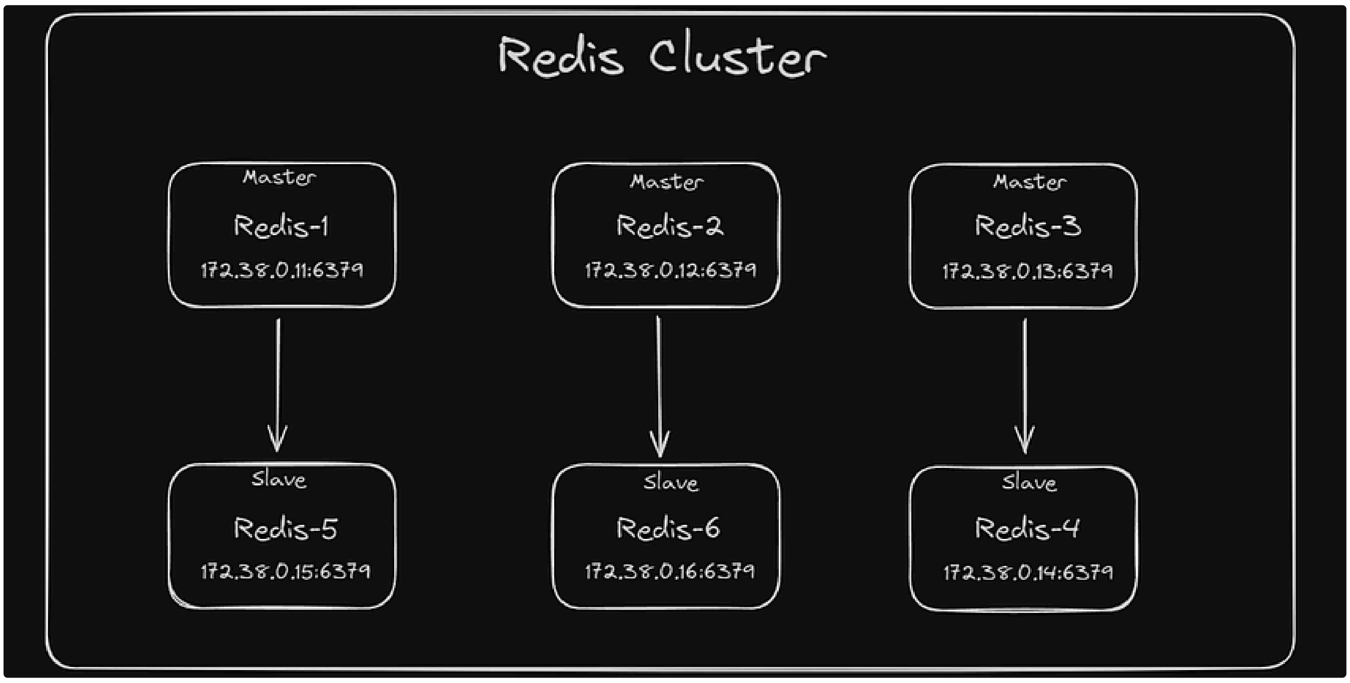
Stories to Help You Level-Up at Work
19 stories · 635 saves



Self-Improvement 101
20 stories · 1950 saves



Productivity 101
20 stories · 1772 saves

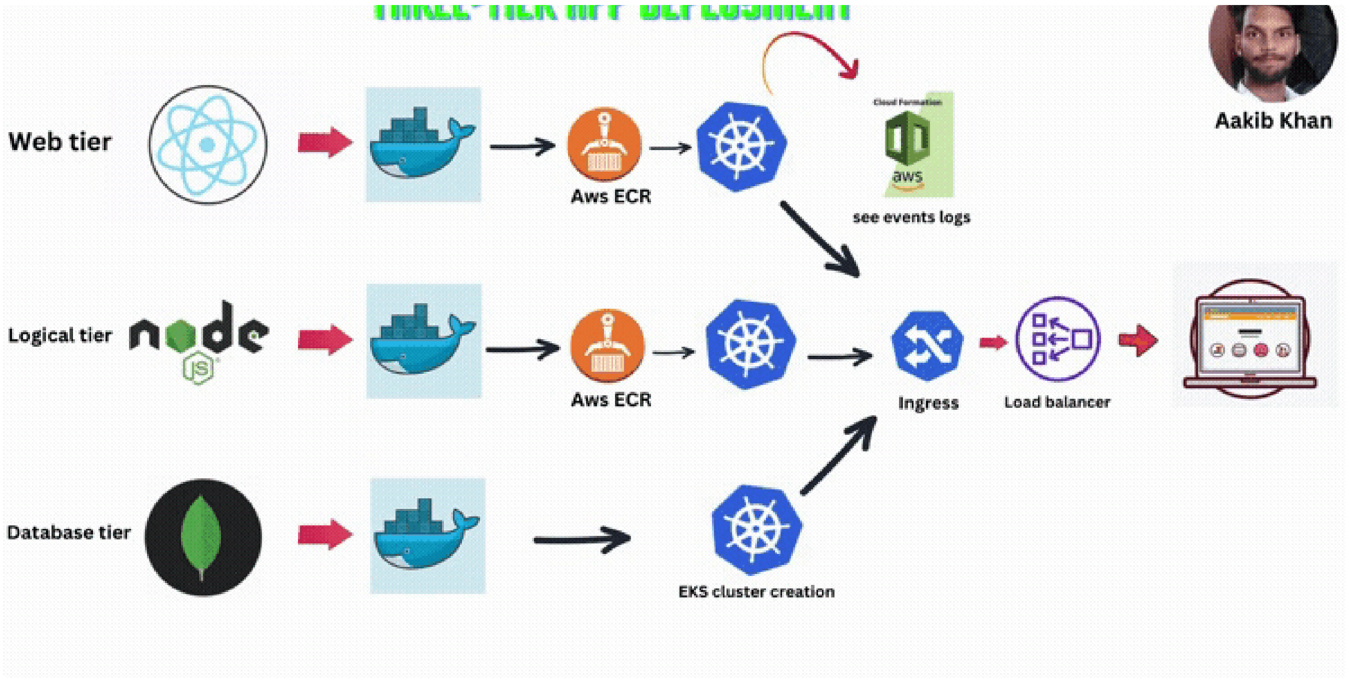


Lim Yee Jie

Basic Docker Compose and Build a Redis Cluster with Docker Compose

Docker Compose

Jan 23 30 1



Aakib

Project 8 → Three tier application deployment on Kubernetes

what do you mean by three tier ?

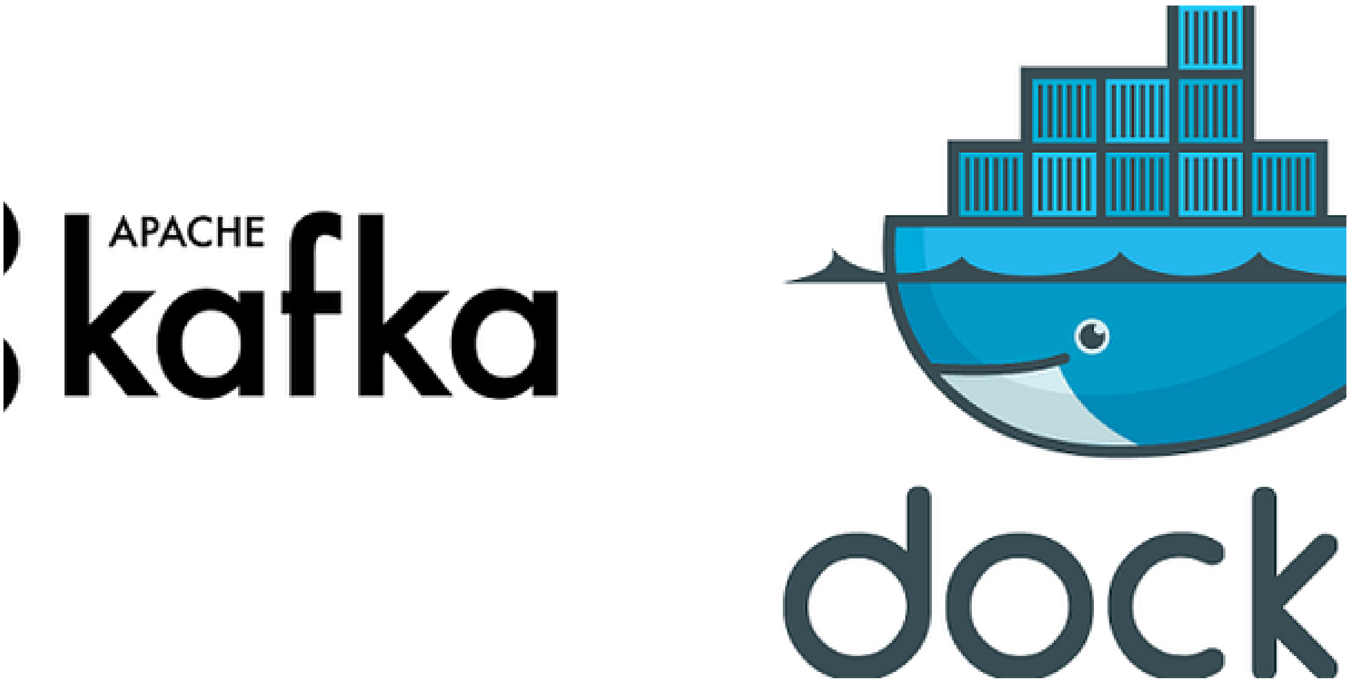
Jan 26 355 3

Mustafa GÜÇ

Building and Deploying a Python Microservice on Kubernetes with GitHub Actions: A Step-by-Step...

This article will guide you through the process of building a simple Python FastAPI microservice and deploying it on a Kubernetes cluster...

Jan 2 192 1



Amber Kakkar

Getting Started with Apache Kafka on Docker: A Step-by-Step Guide

Apache Kafka, a distributed streaming platform, is a powerful tool for building real-time data pipelines and streaming applications. Docker...

Dec 27, 2023 19 1

See more recommendations