# An introductory guide to NGINX using docker

Sri Datta Pavan Ch · Follow

5 min read · 3 days ago

Listen Share More

Nginx ( Pronounced “En-jin-X”) is an open source web server which can be used as an entry point to your application. It can be defined as a web server that can also be used for multiple use cases in your deployment environment.

Nginx is designed to handle a large number of simultaneous connections efficiently while serving static content quickly and acting as a load balancer for HTTP, HTTPS, SMTP, POP3, and IMAP traffic.

A few of the use cases where nginx ca be used are:
. Web server for static html content
. Reverse proxy server
. Load balancing

Let’s explore each of these use cases.

**Static web page loading :**

Let’s say you have created a web page. In local, it can be accessed directly using your browser. But we need it to be deployed and accessible via a domain. Using nginx, you can map the file with a url and when the url is hit, the file is served and the user can access the static page.

Here is a static html page I am using:

```
<!DOCTYPE html>
<html lang="en">
```

```
<body>
 <h1>Welcome to Nginx!</h1>
 <p>This is a simple static HTML page served by Nginx.</p>
</body>
</html>
```

Let's create our nginx.conf file

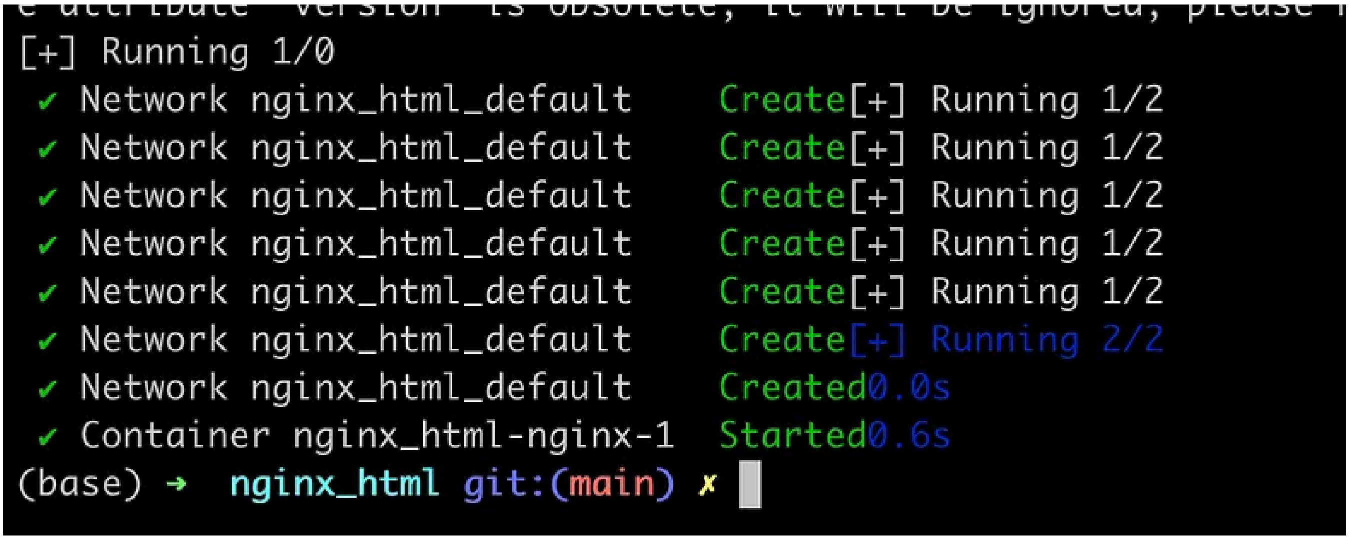
```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    server {
        listen 80;
        server_name localhost;

        location / {
            root /usr/share/nginx/html;
            index index.html;
        }
    }
}
```

In the nginx conf, we are mentioning that the nginx server listens at the port 80 in the localhost (Here we are deploying nginx in docker. So the localhost means the docker container. We port forward and connect from outside. This will be shown in the next sections).
We are mapping the location "/" to the index.html file in "/usr/share/nginx/html".

Let's create our docker file to host this and access.

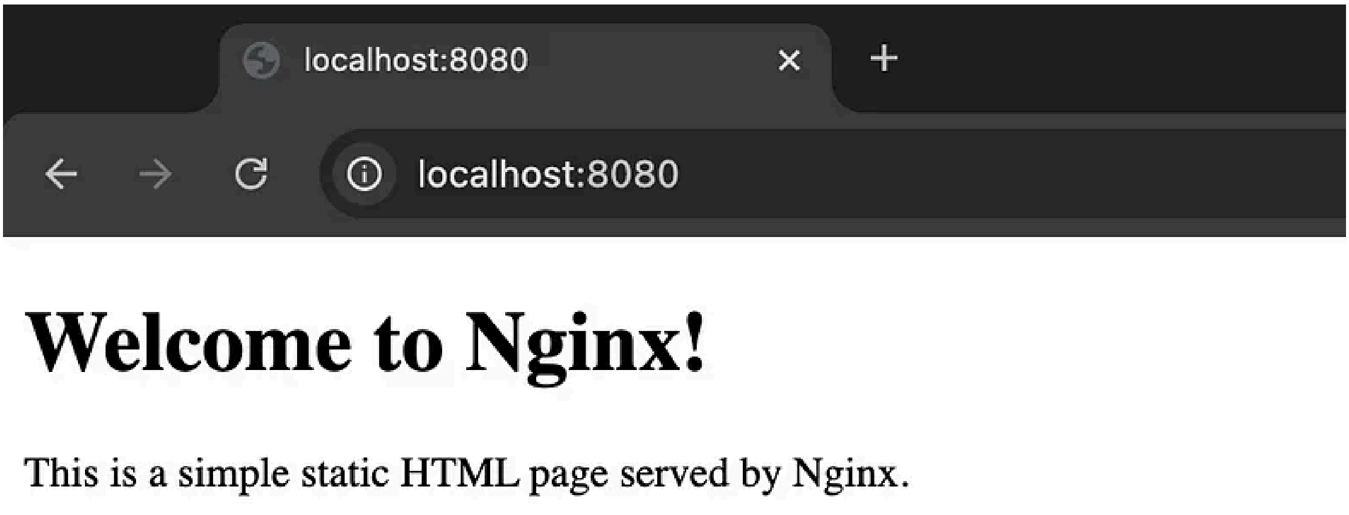
```
version: '3'
services:
  nginx:
    image: nginx
    ports:
      - "8080:80"
    volumes:
      - ./html:/usr/share/nginx/html
      - ./nginx.conf:/etc/nginx/nginx.conf
```

Here we are mounting html folder onto "/usr/share/nginx/html" folder in the nginx container. Also, nginx.conf is mounted to "/etc/nginx/nginx.conf"



```
e attribute `version` is obsolete, it will be ignored, please r
[+] Running 1/0
 ✔ Network nginx_html_default    Create[+] Running 1/2
 ✔ Network nginx_html_default    Create[+] Running 1/2
 ✔ Network nginx_html_default    Create[+] Running 1/2
 ✔ Network nginx_html_default    Create[+] Running 1/2
 ✔ Network nginx_html_default    Create[+] Running 1/2
 ✔ Network nginx_html_default    Create[+] Running 2/2
 ✔ Network nginx_html_default    Created0.0s
 ✔ Container nginx_html-nginx-1  Started0.6s
(base) ➜  nginx_html git:(main) ✗ 
```

Now we access the port in browser



# Welcome to Nginx!

This is a simple static HTML page served by Nginx.

### Reverse proxy :

Reverse proxy, as the name suggests acts reverse to the forward proxy.

A forward proxy sits between a client and the internet, intercepting requests from the client and forwarding them to the internet.

A reverse proxy sits between a client and an origin server, intercepting requests from the client and forwarding them to the origin server.

For this example, I have taken three simple servers, backend, backend1 and backend2. Based on the url let's redirect to the respective server and display the content.

The nginx.conf for this use case:

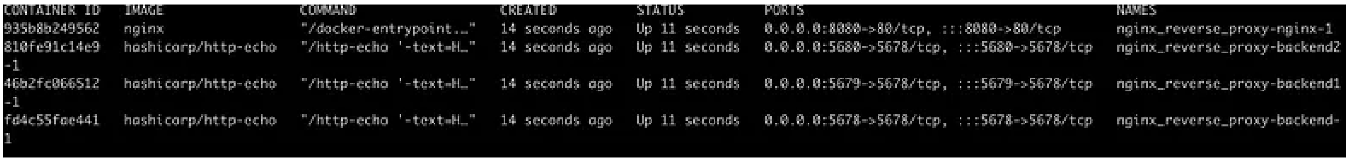
```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    server {
        listen 80;
        server_name localhost;

        location / {
            proxy_pass http://nginx_reverse_proxy-backend-1:5678;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

        location /app1 {
            proxy_pass http://nginx_reverse_proxy-backend1-1:5678;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

        location /app2 {
            proxy_pass http://nginx_reverse_proxy-backend2-1:5678;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
            }
        }
    }
```

Here, we are mapping “/” to backend, “/app1” to backend1 and “/app2” to backend2.

Our docker-compose file to host the three servers is:

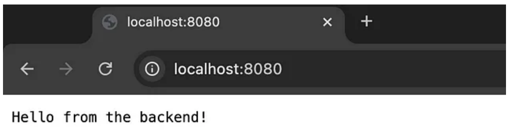
```
version: '3'
services:
  nginx:
    image: nginx
    ports:
      - "8080:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
  backend:
    image: hashicorp/http-echo
    command: -text="Hello from the backend!"
    ports:
      - "5678:5678"
  backend1:
    image: hashicorp/http-echo
    command: -text="Hello from Backend 1!"
    ports:
      - "5679:5678"
  backend2:
    image: hashicorp/http-echo
    command: -text="Hello from Backend 2!"
    ports:
      - "5680:5678"
```

Here we have 4 servers in total including the nginx server. Let’s bring our servers up and try the urls.

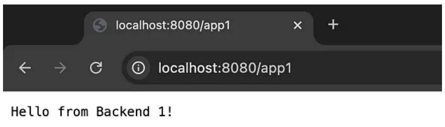


```
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS          PORTS                                       NAMES
935b8b249562   nginx                 "/docker-entrypoint.…"   14 seconds ago   Up 11 seconds   0.0.0.0:8080->80/tcp, :::8080->80/tcp       nginx_reverse_proxy-nginx-1
810fe91c14e9   hashicorp/http-echo   "/http-echo '-text=H…"   14 seconds ago   Up 11 seconds   0.0.0.0:5680->5678/tcp, :::5680->5678/tcp   nginx_reverse_proxy-backend2
-1
46b2fc066512   hashicorp/http-echo   "/http-echo '-text=H…"   14 seconds ago   Up 11 seconds   0.0.0.0:5679->5678/tcp, :::5679->5678/tcp   nginx_reverse_proxy-backend1
-1
fd4c55fae441   hashicorp/http-echo   "/http-echo '-text=H…"   14 seconds ago   Up 11 seconds   0.0.0.0:5678->5678/tcp, :::5678->5678/tcp   nginx_reverse_proxy-backend-
1
```

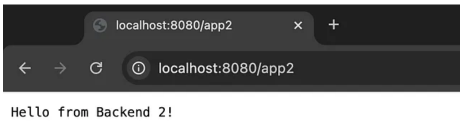
Let’s access the urls in browser



Hello from the backend!



Hello from Backend 1!



Hello from Backend 2!

With the three urls, we can access the three servers.

**Load balancer :**

A load balancer is a device or service that distributes network traffic across a set of servers to improve application performance and availability.

Nginx can perform as a load balancer effectively.

We are creating three servers and hitting the same url will result in content coming from any of the three servers.

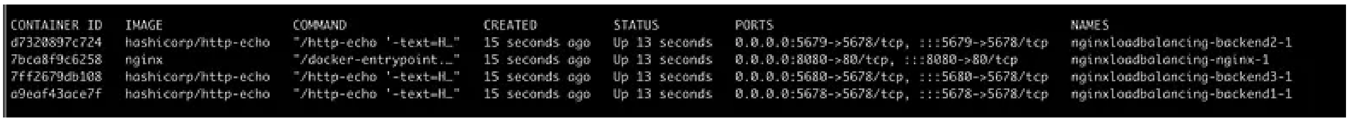
The nginx.conf file is:

```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    upstream backend_cluster {
        server nginxloadbalancing-backend1-1:5678;
        server nginxloadbalancing-backend2-1:5678;
        server nginxloadbalancing-backend3-1:5678;
    }

    server {
        listen 80;
        server_name localhost;

        location / {
            proxy_pass http://backend_cluster;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
        }
    }
```

The docker-compose file for this is:

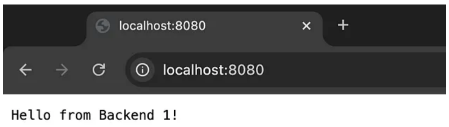
```
version: '3'
services:
  nginx:
    image: nginx
    ports:
      - "8080:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf

  backend1:
    image: hashicorp/http-echo
    command: -text="Hello from Backend 1!"
    ports:
      - "5678:5678"

  backend2:
    image: hashicorp/http-echo
    command: -text="Hello from Backend 2!"
    ports:
      - "5679:5678"

  backend3:
    image: hashicorp/http-echo
    command: -text="Hello from Backend 3!"
    ports:
      - "5680:5678"
```

Let’s bring our servers up:

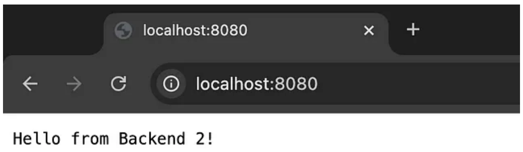


```
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS          PORTS                                         NAMES
d7320897c724   hashicorp/http-echo   "/http-echo '-text=H…"   15 seconds ago   Up 13 seconds   0.0.0.0:5679->5678/tcp, :::5679->5678/tcp   nginxloadbalancing-backend2-1
7bca8f9c6258   nginx                 "/docker-entrypoint.…"   15 seconds ago   Up 13 seconds   0.0.0.0:8080->80/tcp, :::8080->80/tcp       nginxloadbalancing-nginx-1
7ff2679db108   hashicorp/http-echo   "/http-echo '-text=H…"   15 seconds ago   Up 13 seconds   0.0.0.0:5680->5678/tcp, :::5680->5678/tcp   nginxloadbalancing-backend3-1
a9eaf43ace7f   hashicorp/http-echo   "/http-echo '-text=H…"   15 seconds ago   Up 13 seconds   0.0.0.0:5678->5678/tcp, :::5678->5678/tcp   nginxloadbalancing-backend1-1
```

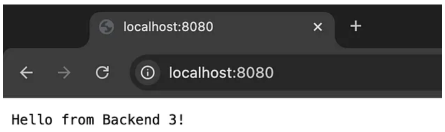
Let’s hit localhost in the browser.



```
Hello from Backend 1!
```



```
Hello from Backend 2!
```



```
Hello from Backend 3!
```

For the same url, we are able to access the three servers. This is because, nginx acted as a load balancer and based on the traffic redirected the user.

Nginx is not limited to these use cases alone.

Further we can explore and play with caching, SSL termination, security control, monitoring and scaling and as a mail proxy.

Nginx Docker Reverse Proxy Load Balancer

Follow

## Written by Sri Datta Pavan Ch

0 Followers

## Recommended from Medium

Hariharan

## Enable cors origin — Nginx

1. Open NGINX Server Configuration

Sep 20 50

Open in app ↗

Medium Search

Harendra

## Top 10 Most Used Open Source SaaS Products

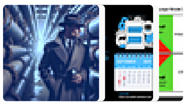
Start using top open-source products for your daily tasks and save money

✦ Sep 24 799 5

## Lists

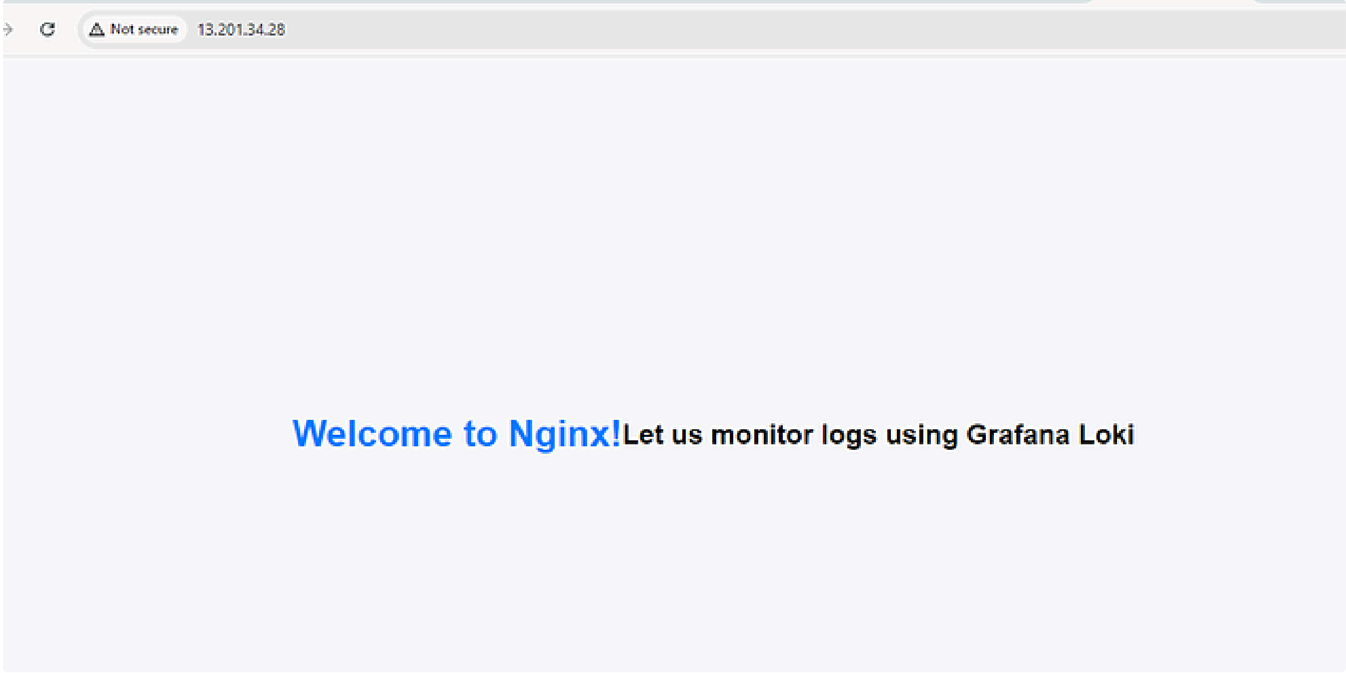
### Coding & Development

11 stories · 826 saves



### Natural Language Processing

1733 stories · 1314 saves



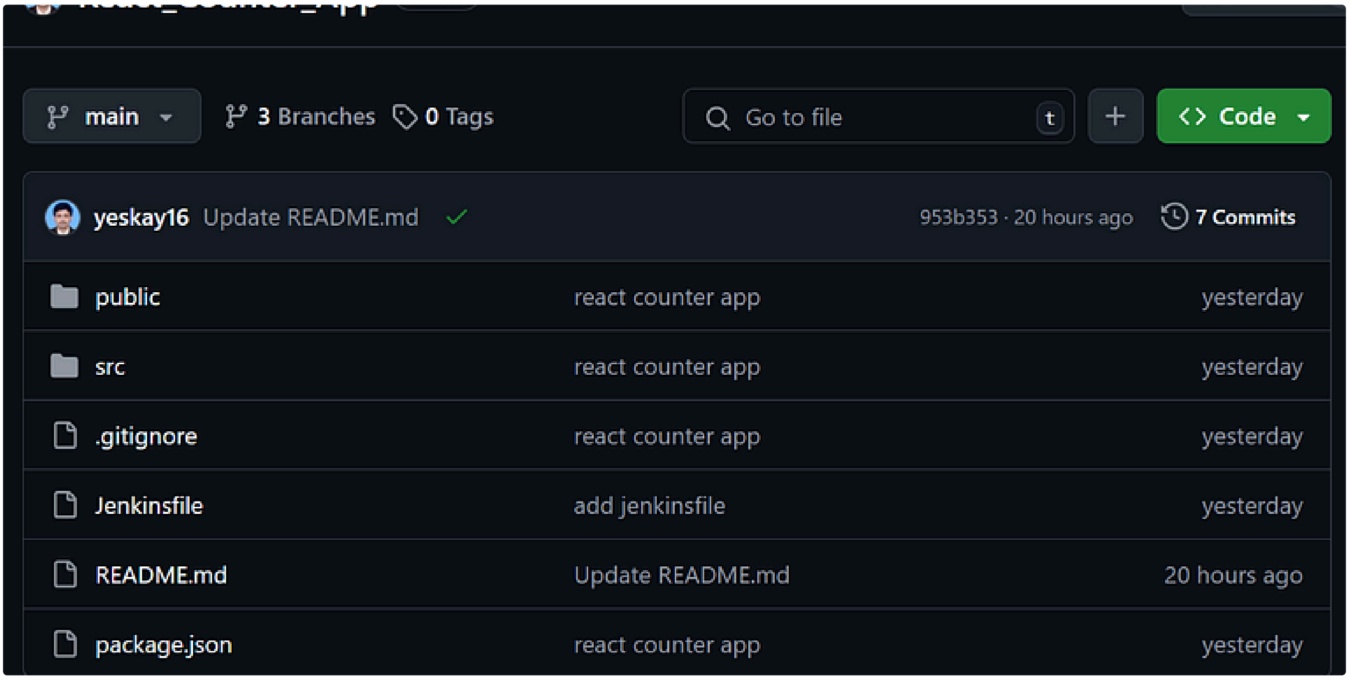
Welcome to Nginx!Let us monitor logs using Grafana Loki

Naman Sharma in DevOps.dev

## Harnessing the Power of Grafana Loki: Pro Tips for Nginx Log Management

What is Grafana , Grafana-Loki and Promtail.

4d ago 1



yeskay16 Update README.md ✓ 953b353 · 20 hours ago 7 Commits

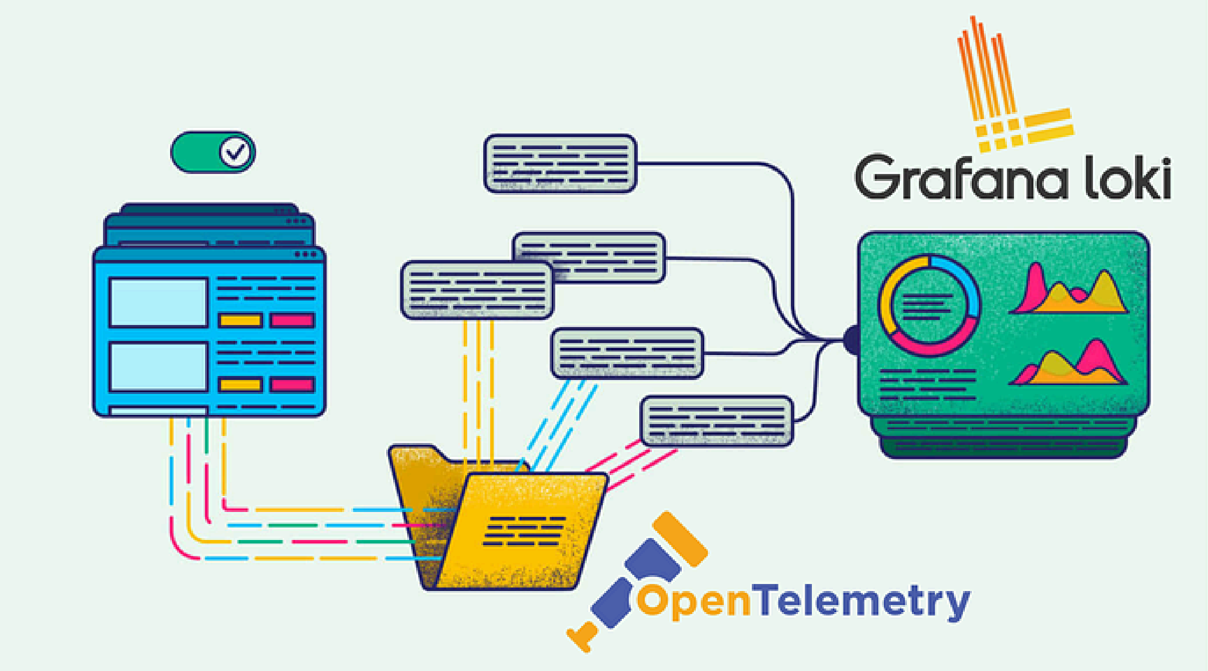
| | | |
|---|---|---|
| public | react counter app | yesterday |
| src | react counter app | yesterday |
| .gitignore | react counter app | yesterday |
| Jenkinsfile | add jenkinsfile | yesterday |
| README.md | Update README.md | 20 hours ago |
| package.json | react counter app | yesterday |

Karthik Seenuvasan

## Trigger Jenkins—Multibranch Pipleline using Jenkinsfile.

https://youtu.be/vQmS0cEjTyw

Sep 25 55



İlayda Dastan

## Log Analysis and Visualization with Grafana Loki and OpenTelemetry

Hey there! What challenges have you faced in your log analysis processes? Personally, I've learned a lot about tackling these issues with...

Jun 24 26

Sachinthana Buddhika

## CI/CD Pipeline with GitHub, Jenkins and Docker

A CI/CD pipeline automates code deployment using GitHub, Jenkins, and Docker.

✦ Jun 16 13

See more recommendations