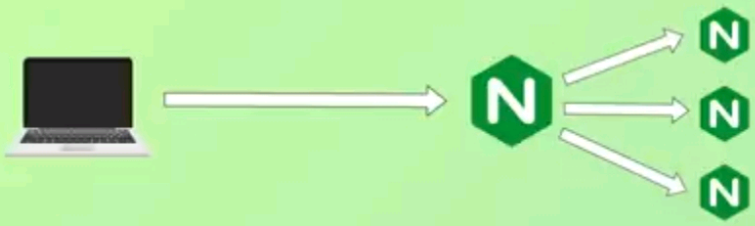
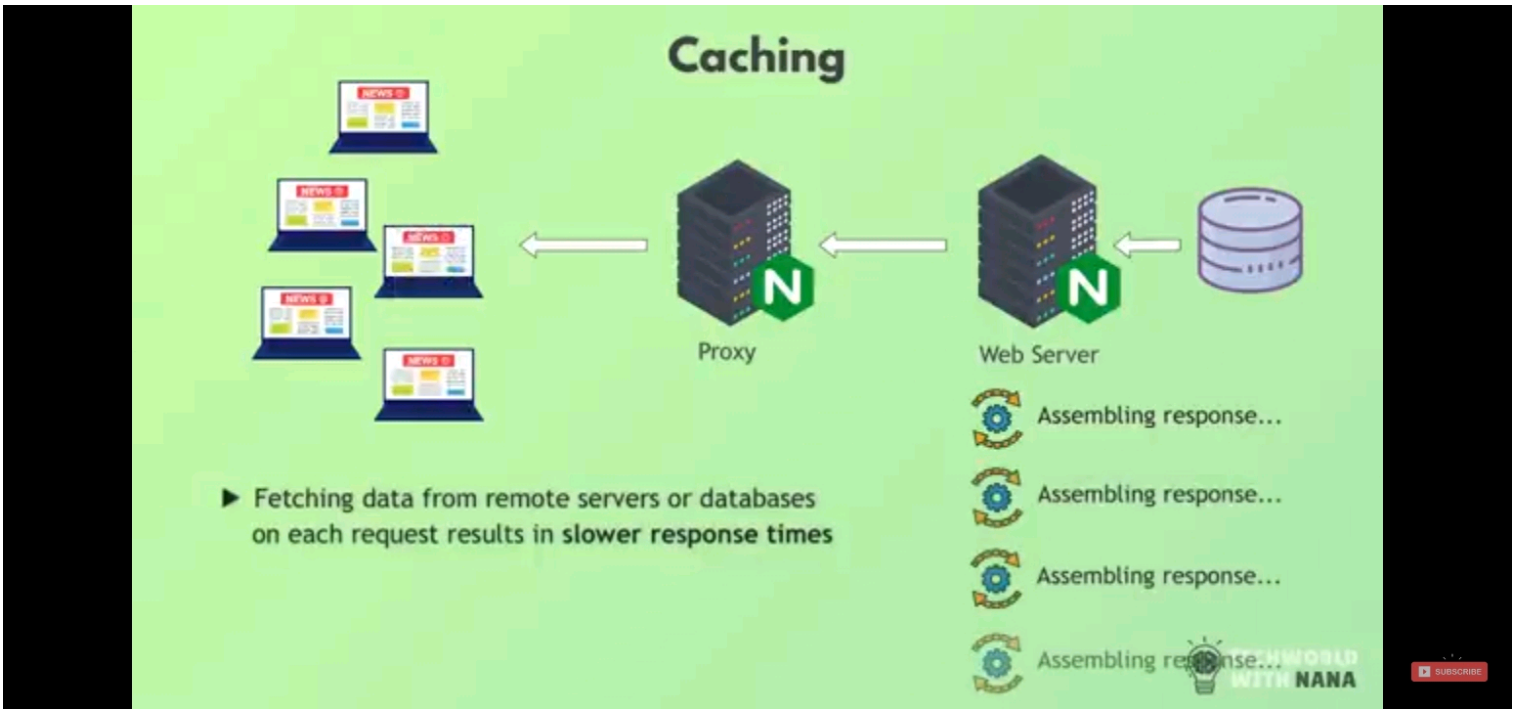
NGINX as Load Balancer



Some **Load Balancing Methods** include

- **Least Connections**
  Routes traffic to the server with the fewest active connections
- **Round Robin**
  Distributes client requests in a **sequential, cyclical manner** to each server in the group


Caching
Proxy
Web Server
Assembling response...
Assembling response...
Assembling response...
Assembling response...
Fetching data from remote servers or databases on each request results in slower response times
TECHWORLD WITH NANA
SUBSCRIBE

Caching

Instead we want to

- Do the heavy lifting **once**
- And store the response

One entrypoint

Exposing all servers to the public internet

- **Increased Attack Surface**: Each publicly exposed server represents an entry point that attackers can target
- You need to secure each and every server

TECHWORLD WITH NANA

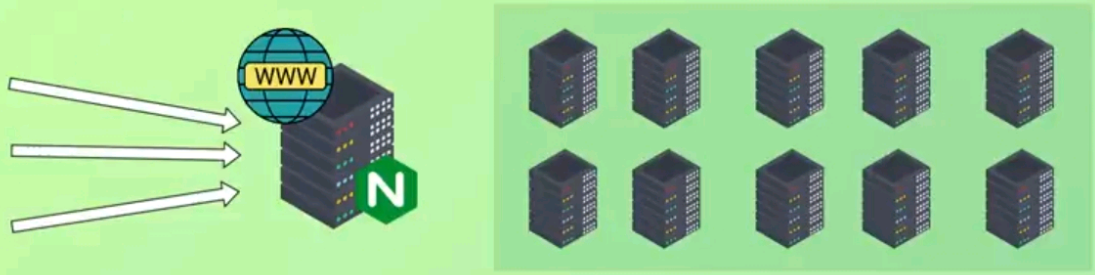
SUBSCRIBE

One entrypoint

Only one server that is publicly available

- ✓ Consolidated Security
- ✓ Minimized Exposure
- ✓ Centralized Access Control
- ✓ Centralized Logging & Monitoring

One entrypoint



Using NGINX Proxy significantly reduces attack surface

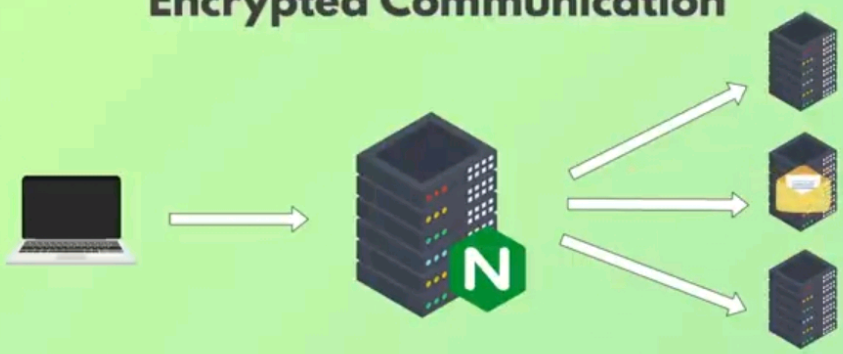
- ✓ Consolidated Security
- ✓ Minimized Exposure
- ✓ Centralized Access Control
- ✓ Centralized Logging & Monitoring

Encrypted Communication

SSL/Termination Offloading

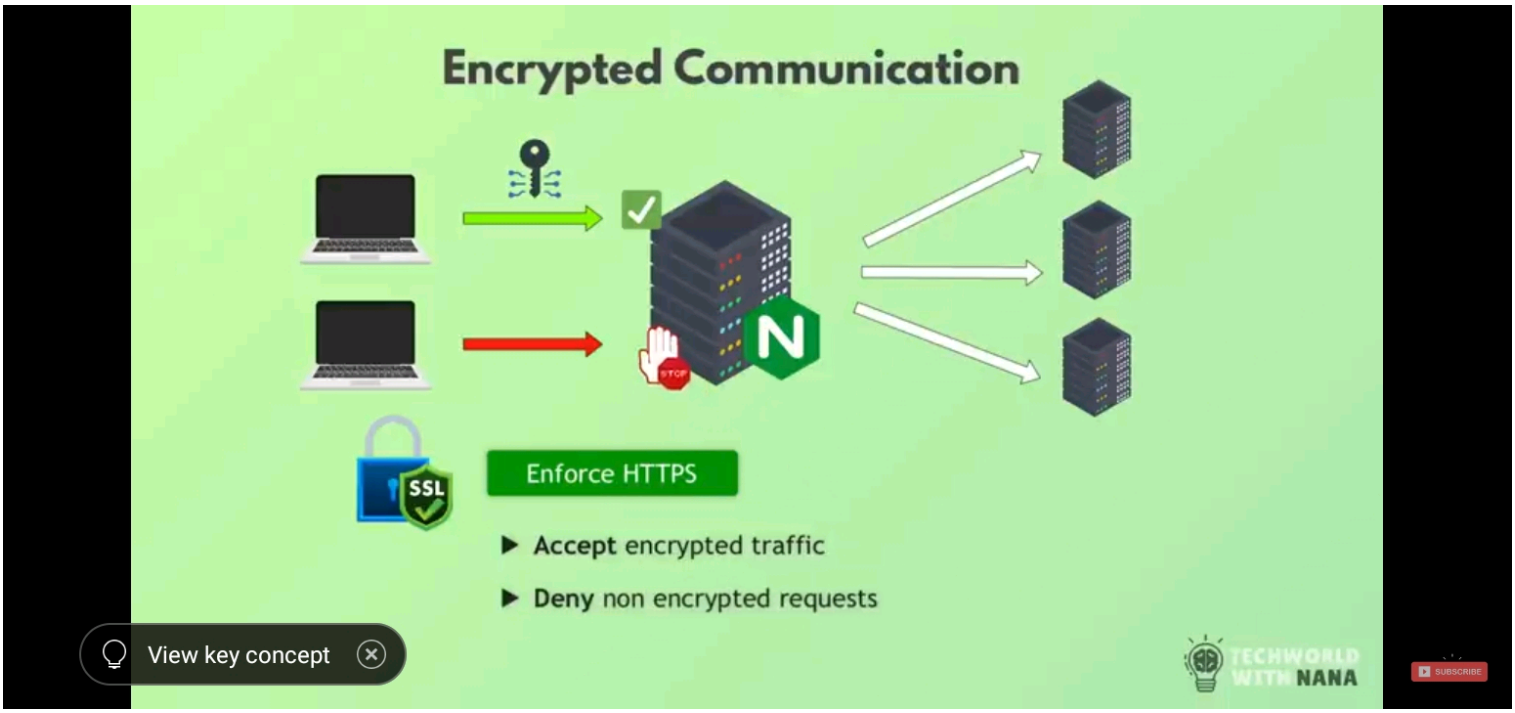
- NGINX can handle SSL/TLS encryption and decryption
- If attacker intercepts message, they **can't read it**

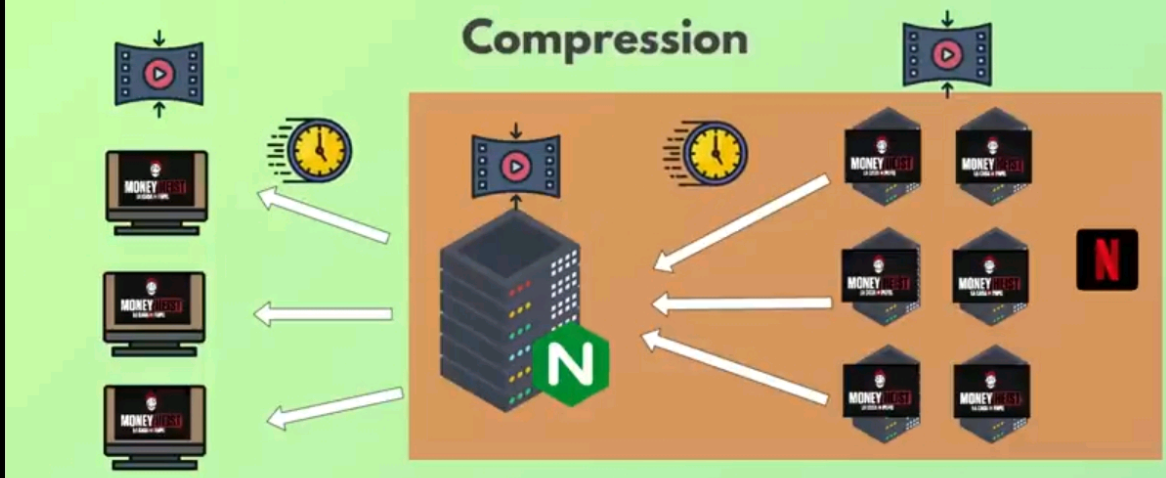
Encrypted Communication

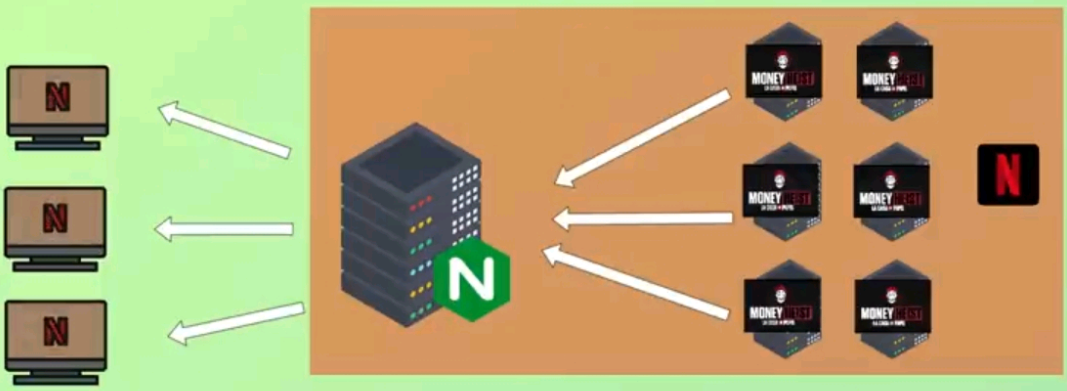


Better: NGINX forwards the encrypted message

Web server decrypts the message itself


Encrypted Communication
SSL
Enforce HTTPS
Accept encrypted traffic
Deny non encrypted requests
View key concept
TECHWORLD WITH NANA
SUBSCRIBE


Compression
NGINX Proxy can compress the response
To reduce bandwidth usage and improve load times
View key concept
TECHWORLD WITH NANA
SUBSCRIBE



Segmentation - Sending response in **chunks**

- Breaks the file into smaller chunks (video streaming)

TECHWORLD WITH NANA

SUBSCRIBE

NGINX Configuration

nginx.config

- The main config file is typically named "**nginx.conf**" and is usually located in the "**/etc/nginx/**" folder
- Using a custom syntax comprising: Directives Blocks
- This sets up the **server's behavior**

NGINX Configuration

```
server {
    listen 80;
    server_name example.com www.example.com;

    location / {
        root /var/www/example.com;
        index index.html index.htm;
    }
}
```

Serving files itself

```
server {
    listen 80;
    server_name api.example.com;

    location / {
        proxy_pass http://backend_server_address;
        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;
    }
}
```

Forward traffic to other web servers

NGINX Configuration

```
server {
    listen 80;
    server_name example.com www.example.com;

    location / {
        root /var/www/example.com;
        index index.html index.htm;
    }
}
```

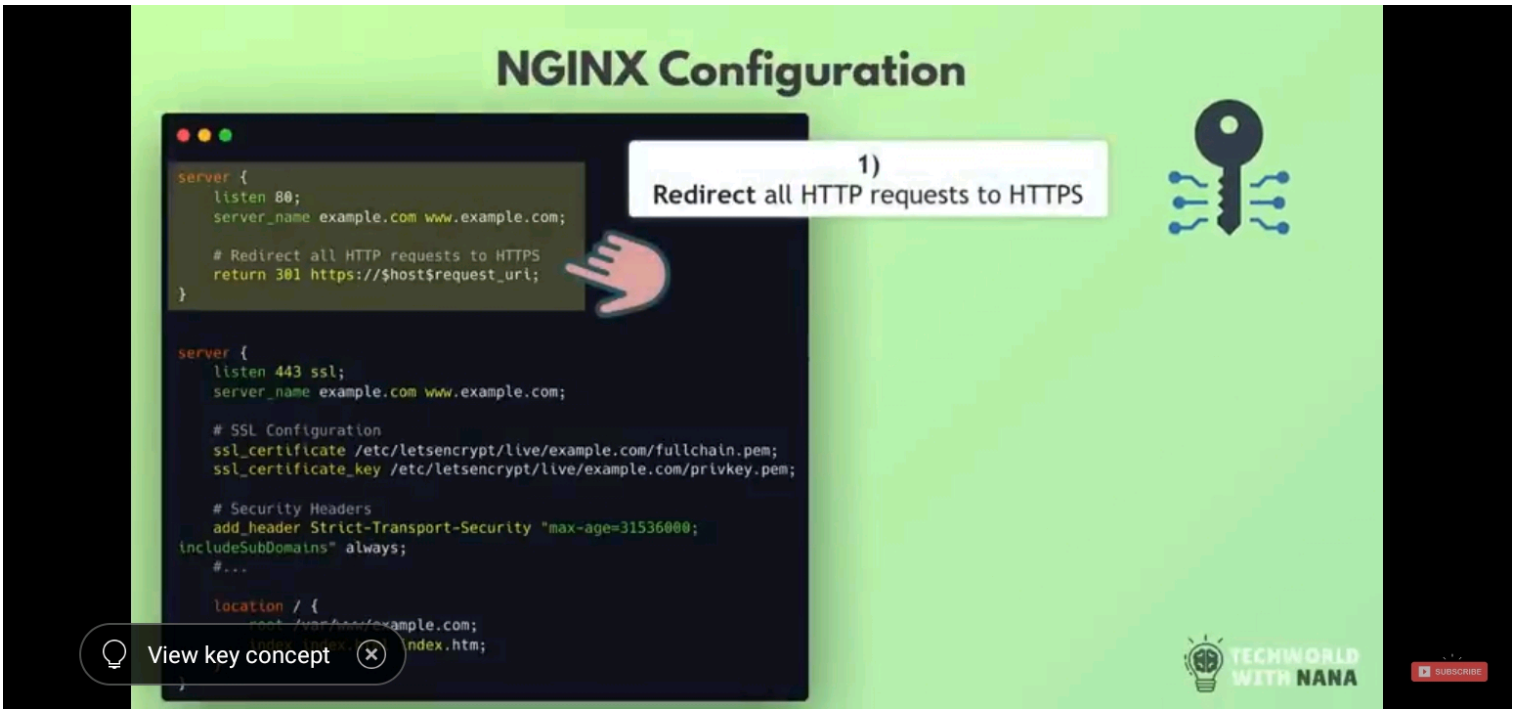
Serving files itself

- Example of web server serving static files

- "**location**" directive defines how the server should process specific types of requests
- Specify the location that contains your files

TECHWORLD WITH NANA

SUBSCRIBE


NGINX Configuration
server {
listen 80;
server_name example.com www.example.com;
Redirect all HTTP requests to HTTPS
return 301 https://$host$request_uri;
}
server {
listen 443 ssl;
server_name example.com www.example.com;
SSL Configuration
ssl_certificate /etc/letsencrypt/live/example.com/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;
# Security Headers
add_header Strict-Transport-Security "max-age=31536000;
includeSubDomains" always;
#...
location / {
1)
Redirect all HTTP requests to HTTPS
View key concept
TECHWORLD
WITH NANA
SUBSCRIBE

NGINX Configuration

```
server {
    listen 80;
    server_name example.com www.example.com;

    # Redirect all HTTP requests to HTTPS
    return 301 https://$host$request_uri;
}
```

```
server {
    listen 443 ssl;
    server_name example.com www.example.com;

    # SSL Configuration
    ssl_certificate /etc/letsencrypt/live/example.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;

    # Security Headers
    add_header Strict-Transport-Security "max-age=31536000;
includeSubDomains" always;
    #...

    location / {
        root /var/www/example.com;
        index index.html index.htm;
    }
}
```

2)
Serve content over **HTTPS** with SSL/TLS configured

TECHWORLD WITH NANA

SUBSCRIBE

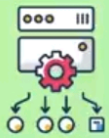
NGINX Configuration

```
http {
    upstream myapp1 {
        server srv1.example.com;
        server srv2.example.com;
        server srv3.example.com;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://myapp1;
        }
    }
}
```

Configure **Load Balancing**



- In the example, there are 3 instances of the same app running on *srv1-srv3*
- All requests are proxied to the server group *myapp1*

```
upstream myapp1 {
    least_conn;
    server srv1.example.com;
    server srv2.example.com;
    server srv3.example.com;
}
```

View key concept

TECHWORLD WITH NANA

SUBSCRIBE

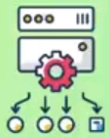
NGINX Configuration

```
http {
    upstream myapp1 {
        server srv1.example.com;
        server srv2.example.com;
        server srv3.example.com;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://myapp1;
        }
    }
}
```

Configure **Load Balancing**



- In the example, there are 3 instances of the same app running on *srv1-srv3*
- All requests are proxied to the server group *myapp1*
- Defaults to **round-robin** algorithm

NGINX Configuration

```
http {
    # ...
    proxy_cache_path /data/nginx/cache keys_zone=mycache:10m;
}
```

Enable the **Caching of Responses**

View key concept

TECHWORLD WITH NANA

SUBSCRIBE

NGINX Configuration

Copy

```
http {
    # ...
    proxy_cache_path /data/nginx/cache keys_zone=mycache:10m;
}
```

Then include the `proxy_cache` directive in the context (protocol type, virtual server, or location) for which you want to cache server responses, specifying the zone name defined by the `keys_zone` parameter to the `proxy_cache_path` directive (in this case, `mycache`):

Copy

```
http {
    # ...
    proxy_cache_path /data/nginx/cache keys_zone=mycache:10m;
    server {
        proxy_cache mycache;
        location / {
            proxy_pass http://localhost:8000;
        }
    }
```

Enable the **Caching of Responses**

View key concept

TECHWORLD WITH NANA

SUBSCRIBE

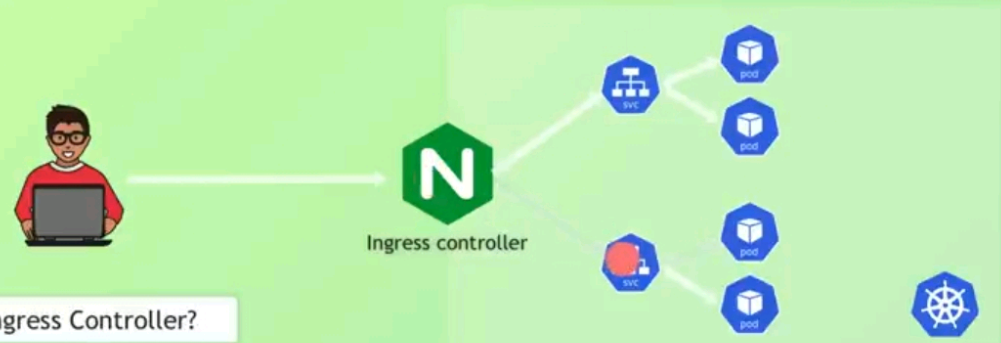
NGINX Configuration

Configuration is **straightforward**

nginx.config

Granular configuration possible

NGINX as a K8s Ingress Controller



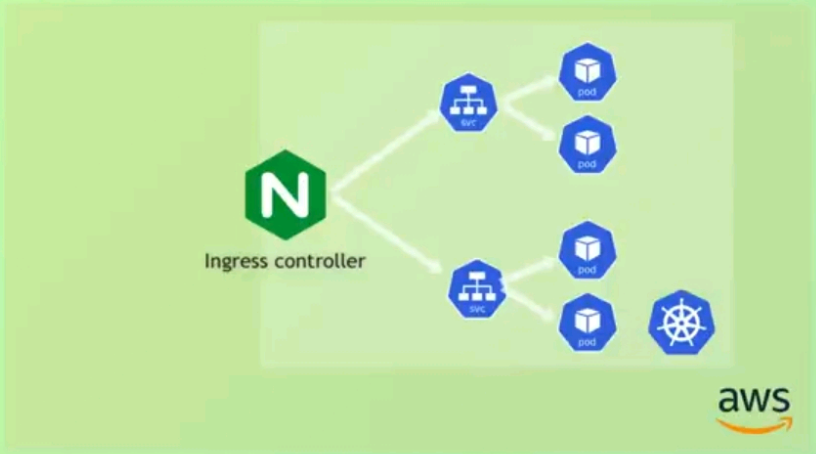
What is Ingress Controller?

- A specialized load balancer for **managing ingress (incoming) traffic** in Kubernetes
- It handles the **routing to the appropriate services** based on rules defined in an Ingress resource

TECHWORLD WITH NANA

SUBSCRIBE

Cloud Platform Load Balancer

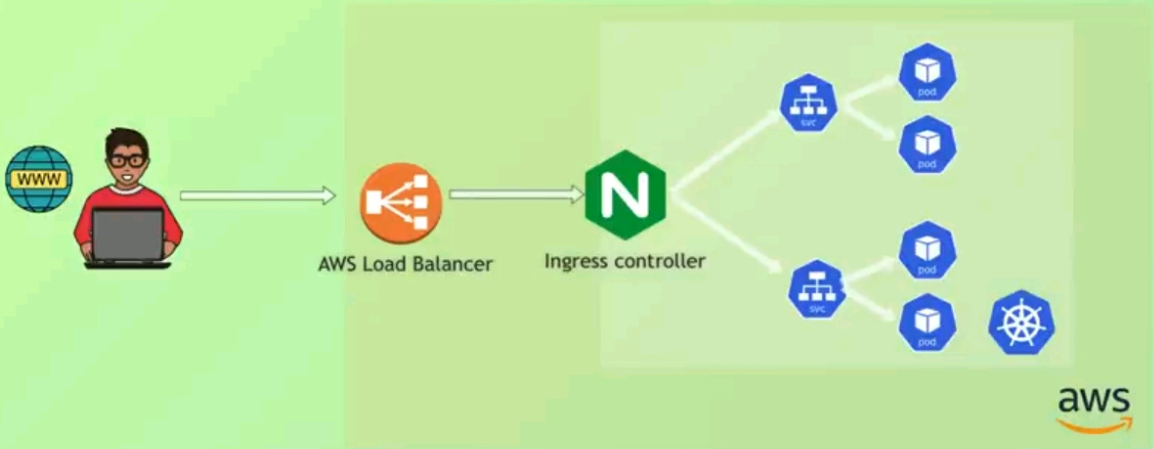


- NGINX Ingress controller acts as load balancer **inside** the K8s cluster
- **NOT publicly accessible**

TECHWORLD WITH NANA

SUBSCRIBE

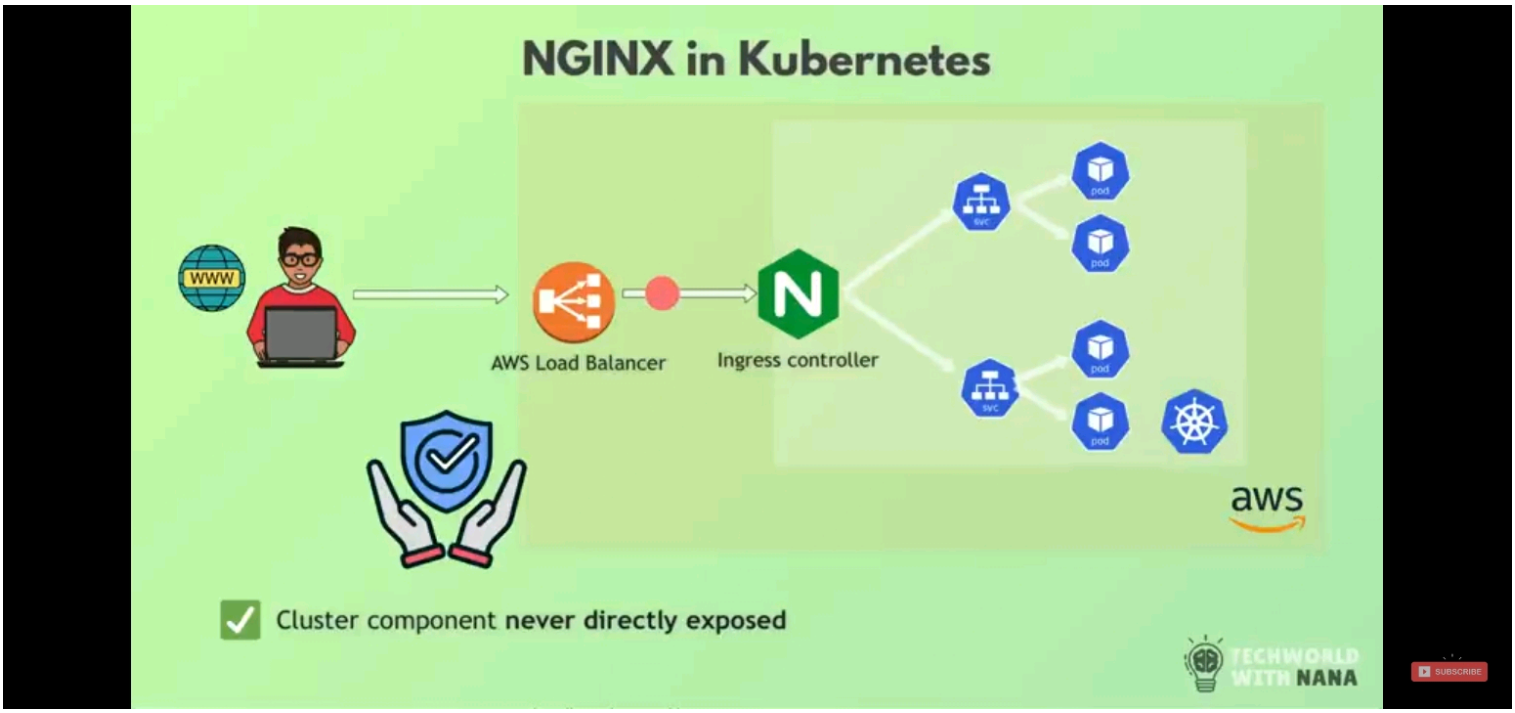
NGINX in Kubernetes

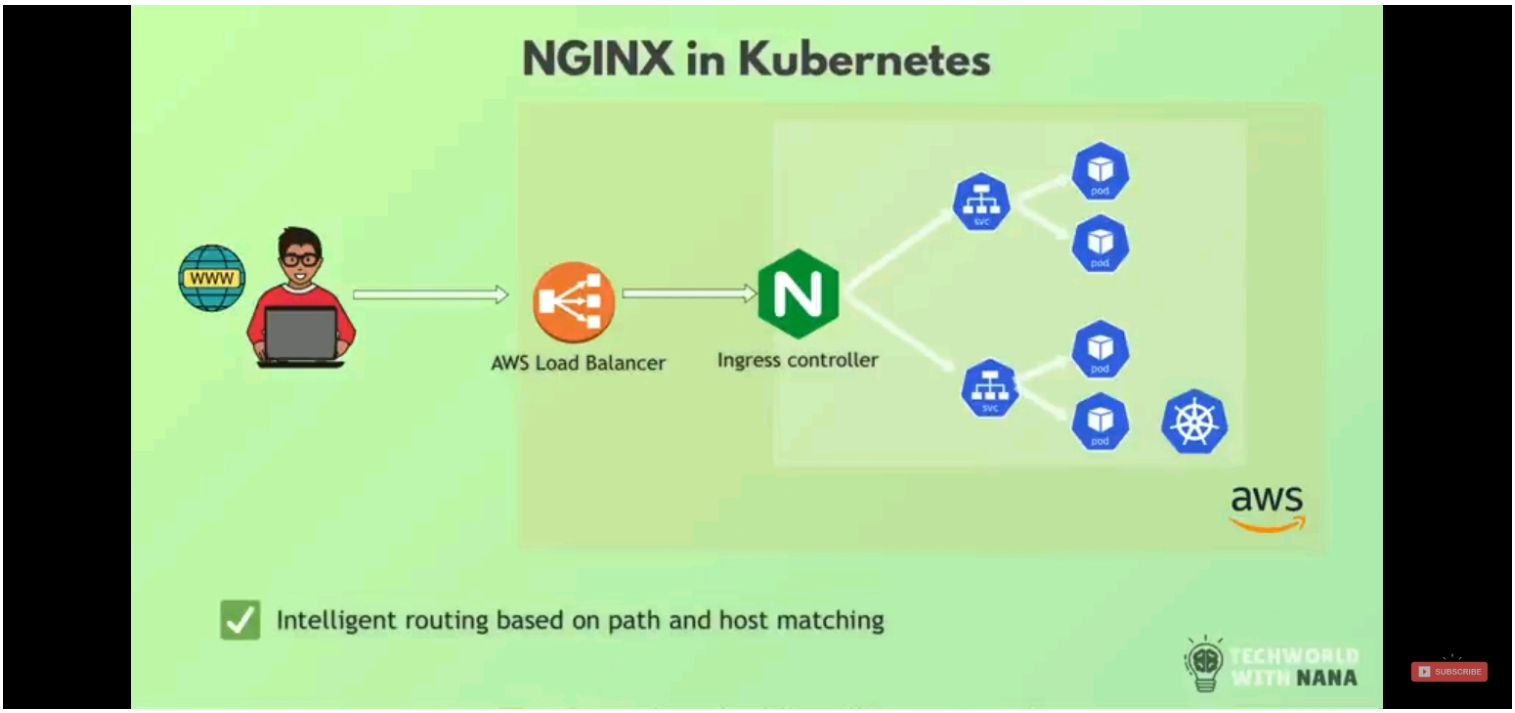


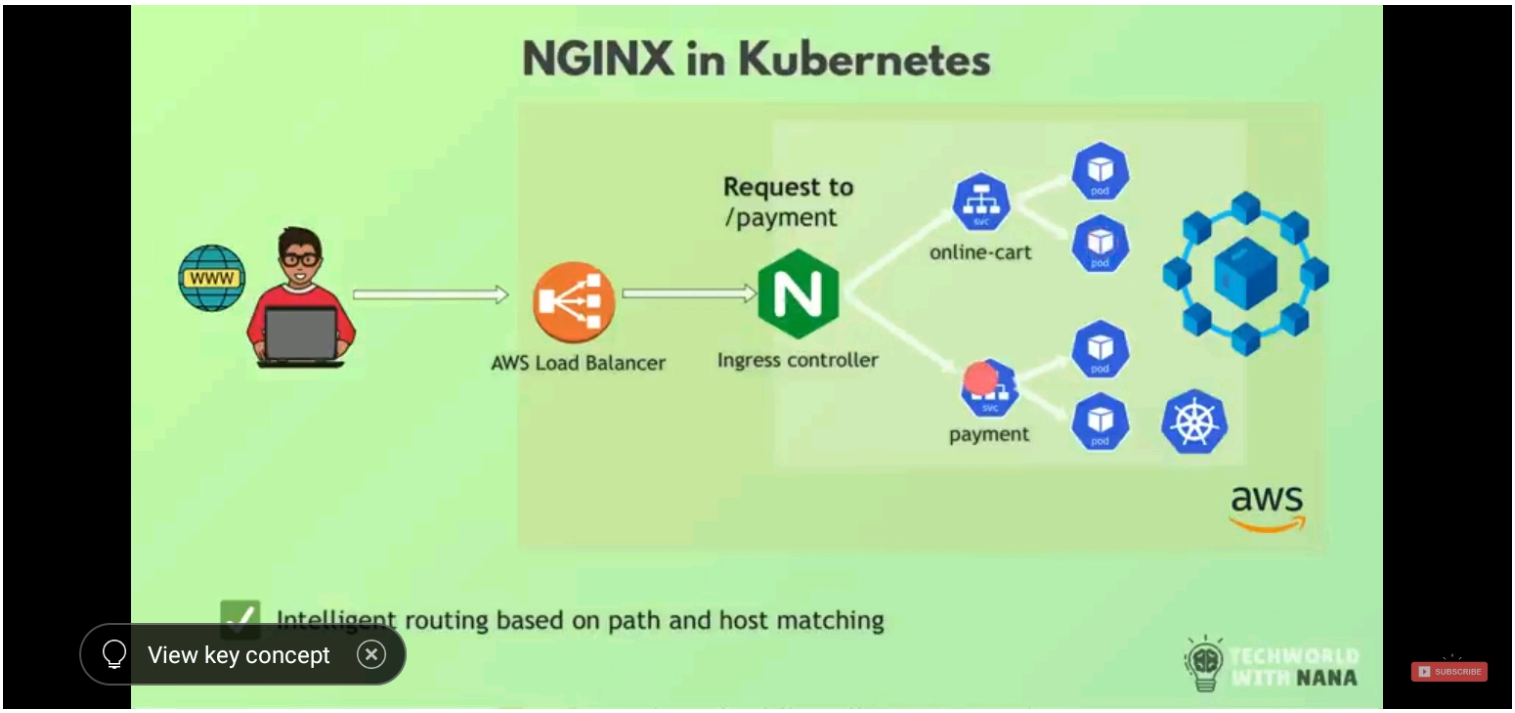
- Cloud load balancer handles the incoming traffic from the internet
- Forwards reqest to Ingress controller

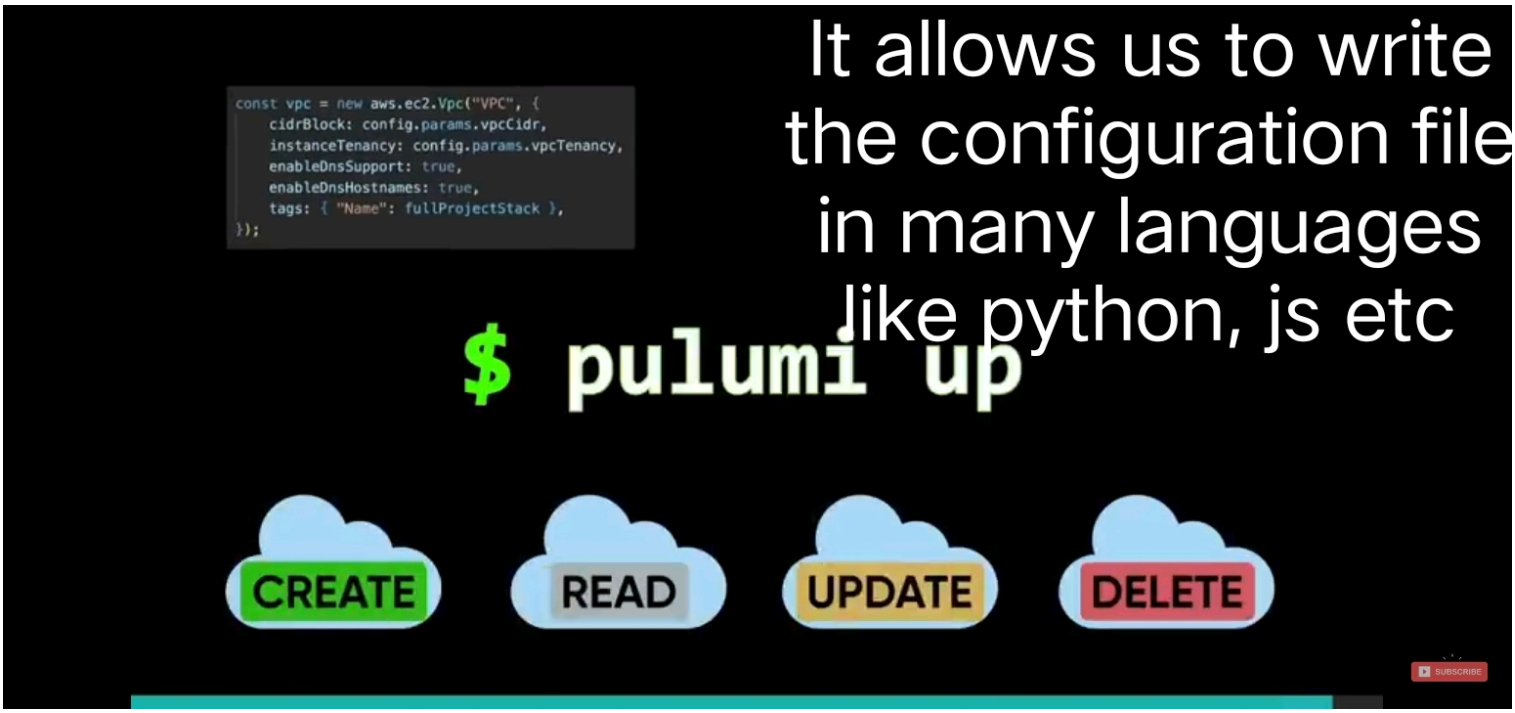
TECHWORLD WITH NANA

SUBSCRIBE


NGINX in Kubernetes
WWW
AWS Load Balancer
Ingress controller
pod
pod
svc
pod
pod
svc
aws
Cluster component never directly exposed
TECHWORLD WITH NANA
SUBSCRIBE


NGINX in Kubernetes
WWW
AWS Load Balancer
Ingress controller
pod
pod
pod
pod
svc
svc
aws
Intelligent routing based on path and host matching
TECHWORLD WITH NANA
SUBSCRIBE


NGINX in Kubernetes
WWW
AWS Load Balancer
Request to
/payment
Ingress controller
online-cart
payment
pod
svc
aws
Intelligent routing based on path and host matching
View key concept
TECHWORLD WITH NANA
SUBSCRIBE


const vpc = new aws.ec2.Vpc("VPC", {
cidrBlock: config.params.vpcCidr,
instanceTenancy: config.params.vpcTenancy,
enableDnsSupport: true,
enableDnsHostnames: true,
tags: { "Name": fullProjectStack },
});
It allows us to write the configuration file in many languages like python, js etc
$ pulumi up
CREATE
READ
UPDATE
DELETE
SUBSCRIBE

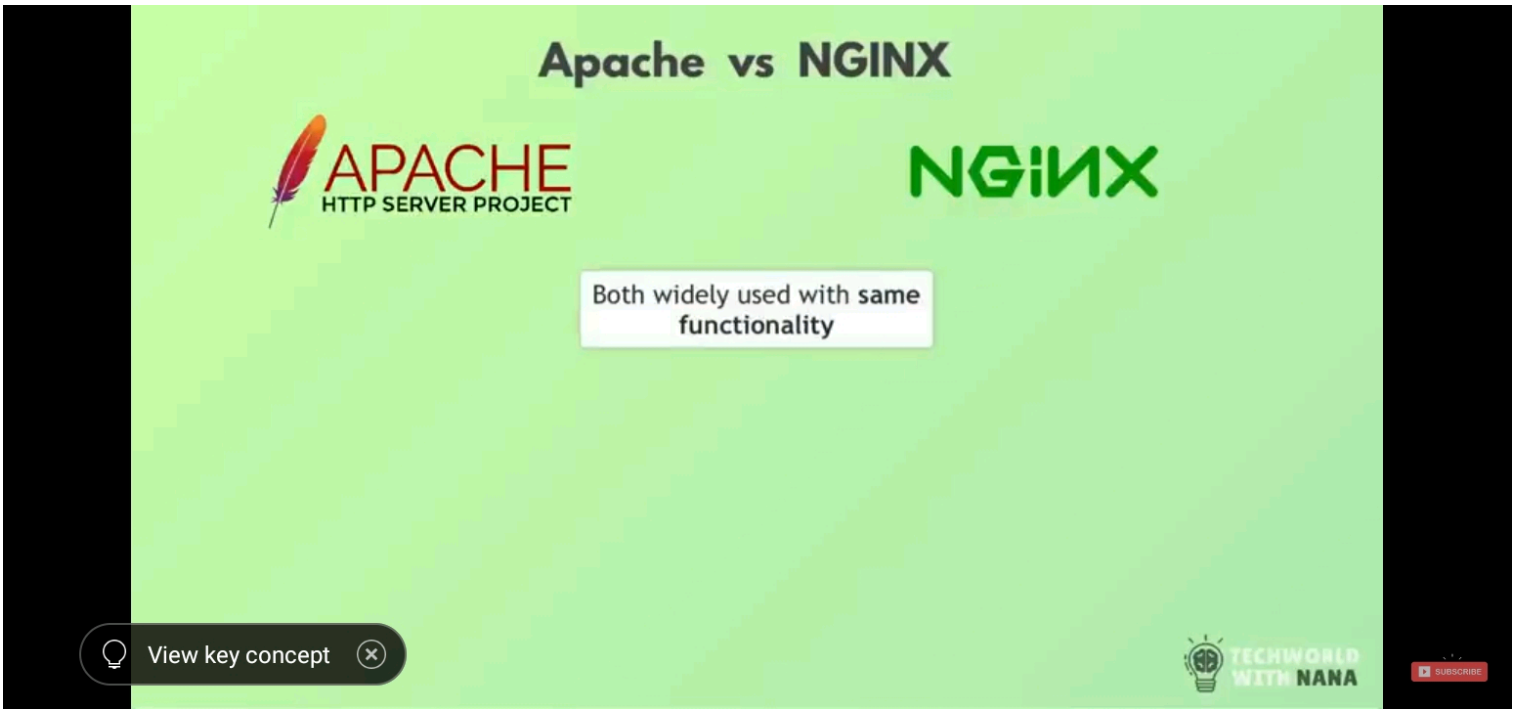
Apache vs NGINX

APACHE HTTP SERVER PROJECT

- Apache was released in 1995

NGINX

- Nginx was released in 2004

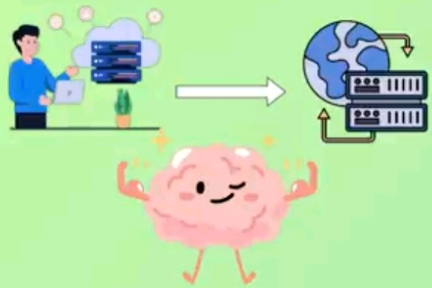

Apache vs NGINX
APACHE
HTTP SERVER PROJECT
NGINX
Both widely used with same functionality
View key concept
TECHWORLD WITH NANA
SUBSCRIBE

Apache vs NGINX

APACHE
HTTP SERVER PROJECT

NGINX

- Apache extended the functionality as a proxy



Apache vs NGINX

APACHE HTTP SERVER PROJECT

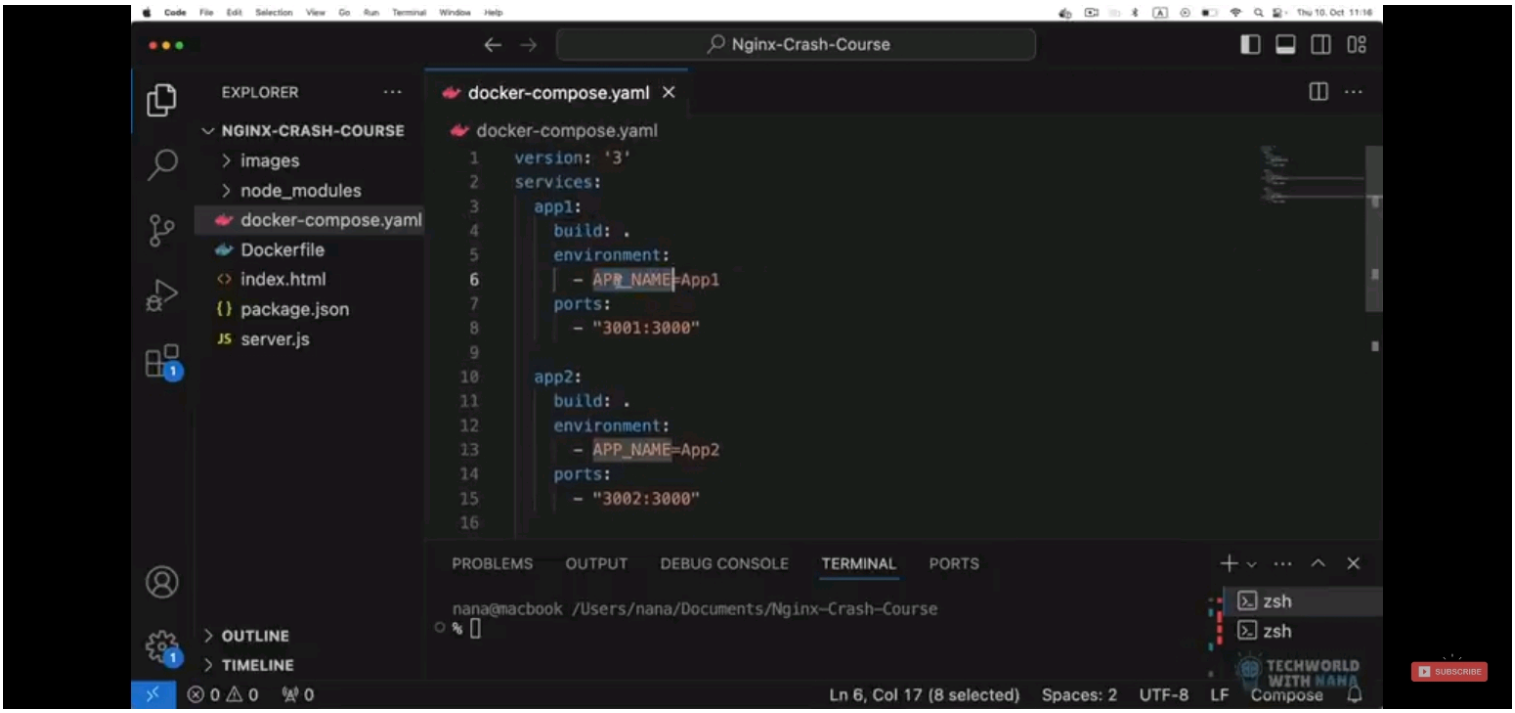
- Highly customizable and extensible
- Good choice for dynamic content handling and legacy support

NGINX

- Faster and more lightweight
- Best suited for high-performance environments and serving static content
- Simple configuration

TECHWORLD WITH NANA

SUBSCRIBE


Nginx-Crash-Course
EXPLORER
NGINX-CRASH-COURSE
images
node_modules
docker-compose.yaml
Dockerfile
index.html
package.json
server.js
docker-compose.yaml
version: '3'
services:
app1:
build: .
environment:
- APP_NAME=App1
ports:
- "3001:3000"
app2:
build: .
environment:
- APP_NAME=App2
ports:
- "3002:3000"
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
nana@macbook /Users/nana/Documents/Nginx-Crash-Course
zsh
zsh
OUTLINE
TIMELINE
Ln 6, Col 17 (8 selected)
Spaces: 2
UTF-8
LF
Compose
TECHWORLD WITH NANA
SUBSCRIBE

worker_processes

- Controls **how many parallel processes** Nginx spawns to handle client requests

What is a "**Worker Process**"?

- Instead of using a new process for every incoming connection, Nginx uses worker processes that **handle many connections** using a single-threaded event loop

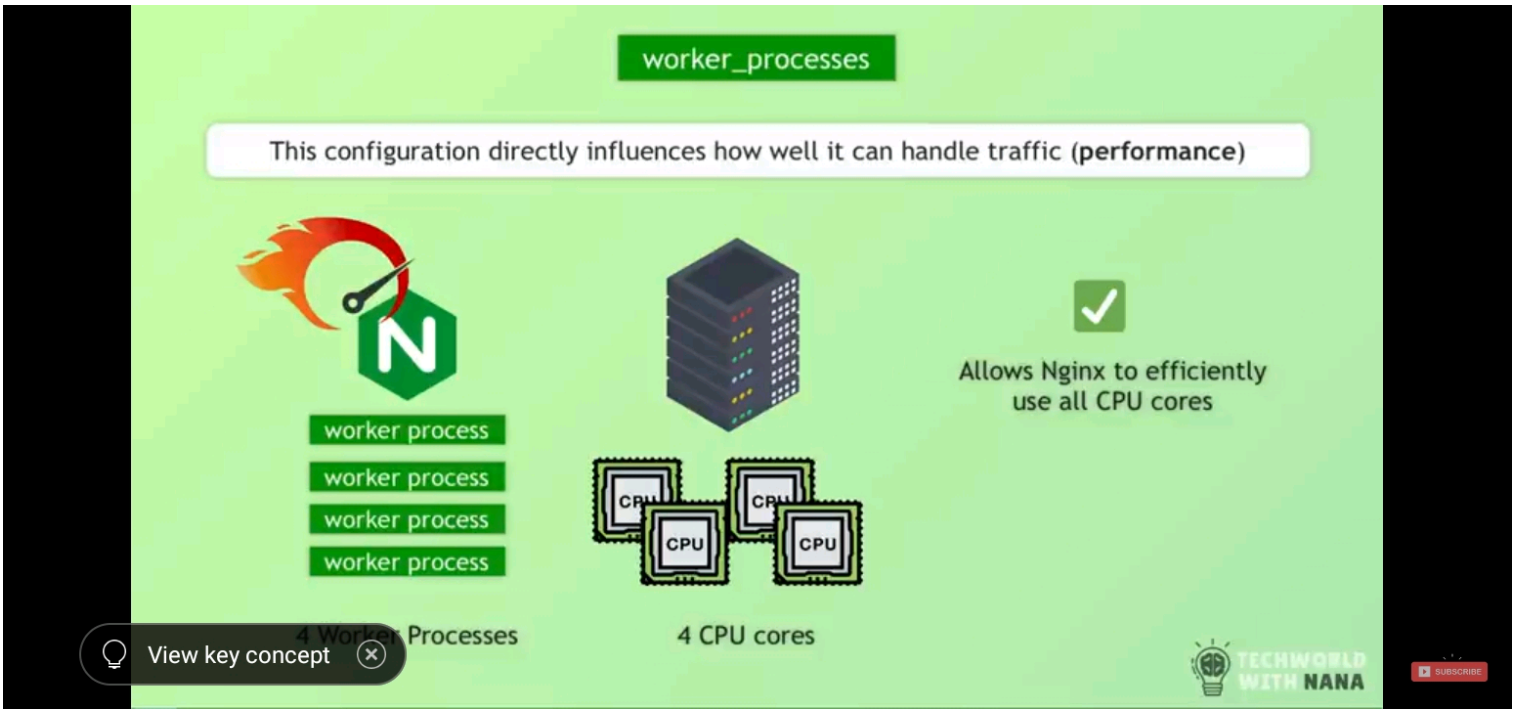
worker_processes

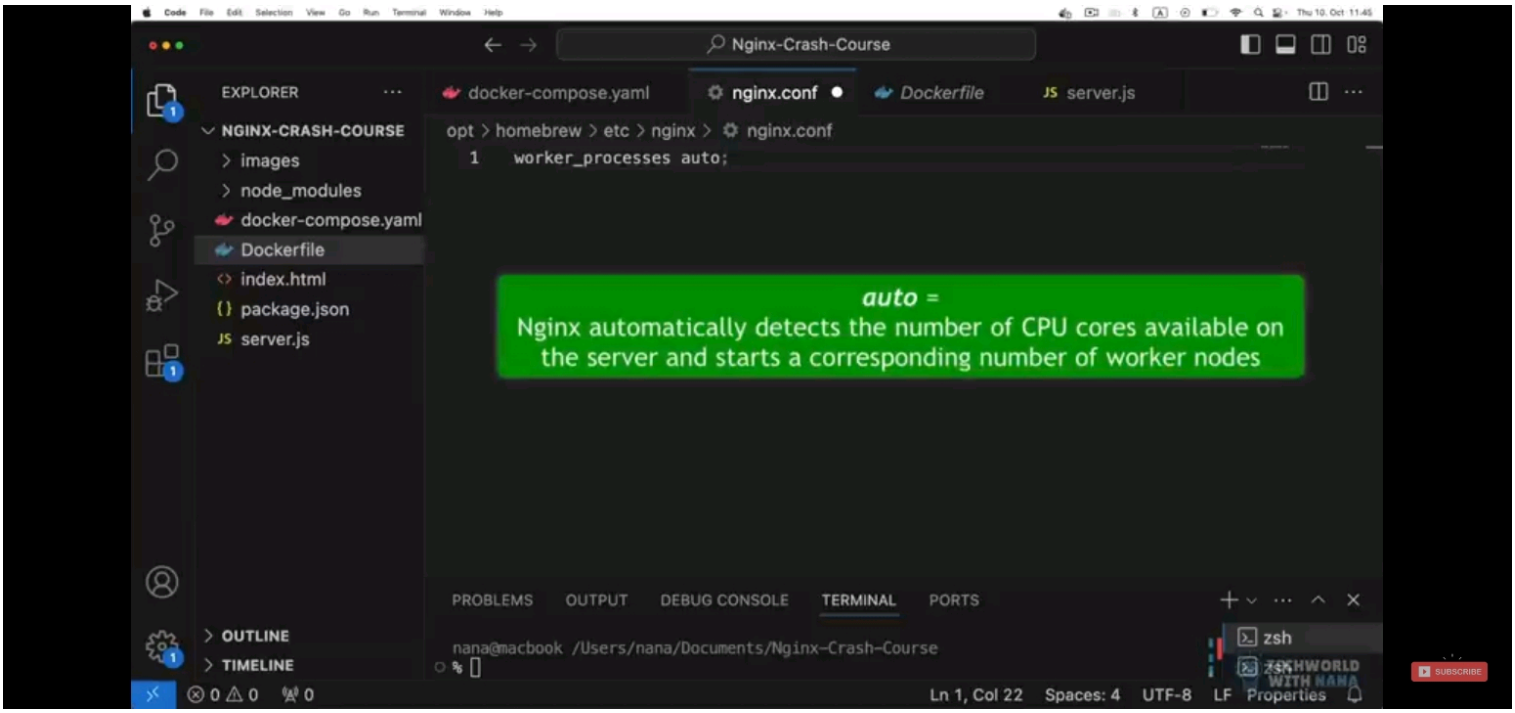
What does the **number** represent?

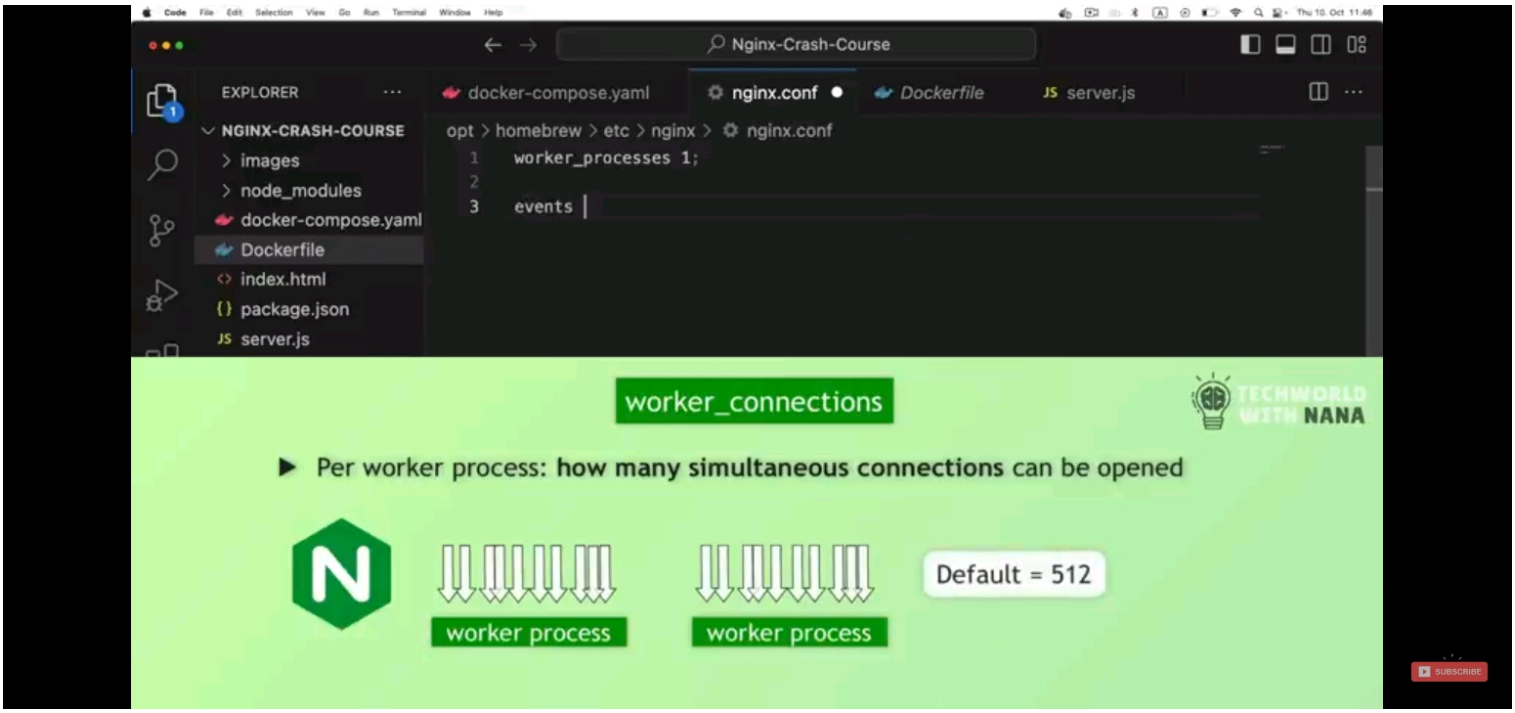
- The value is the number of worker processes Nginx should create
- Each worker process runs independently and can handle its own set of connections

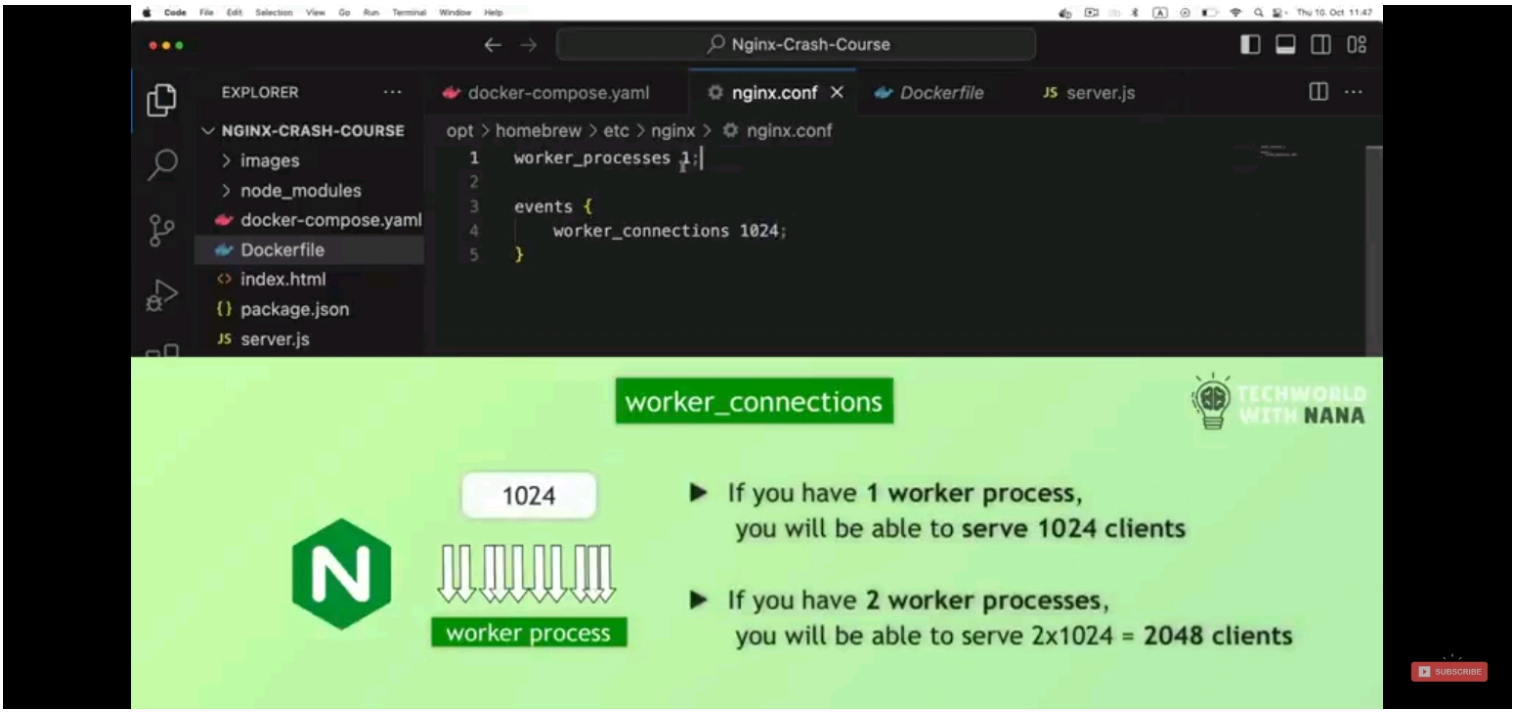
This configuration directly influences how well it can handle traffic (**performance**)

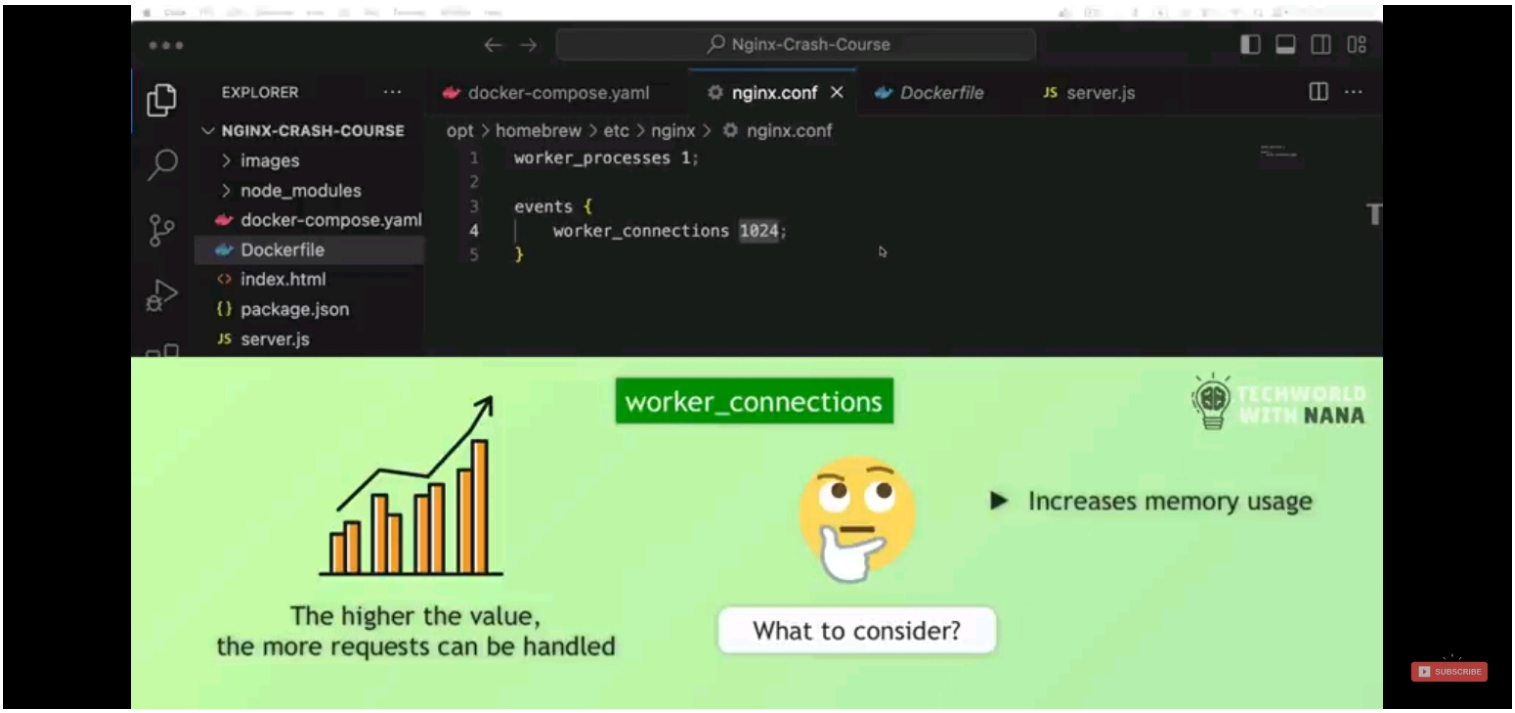
- Should be tuned according to the server's hardware (CPU cores) and expected traffic load

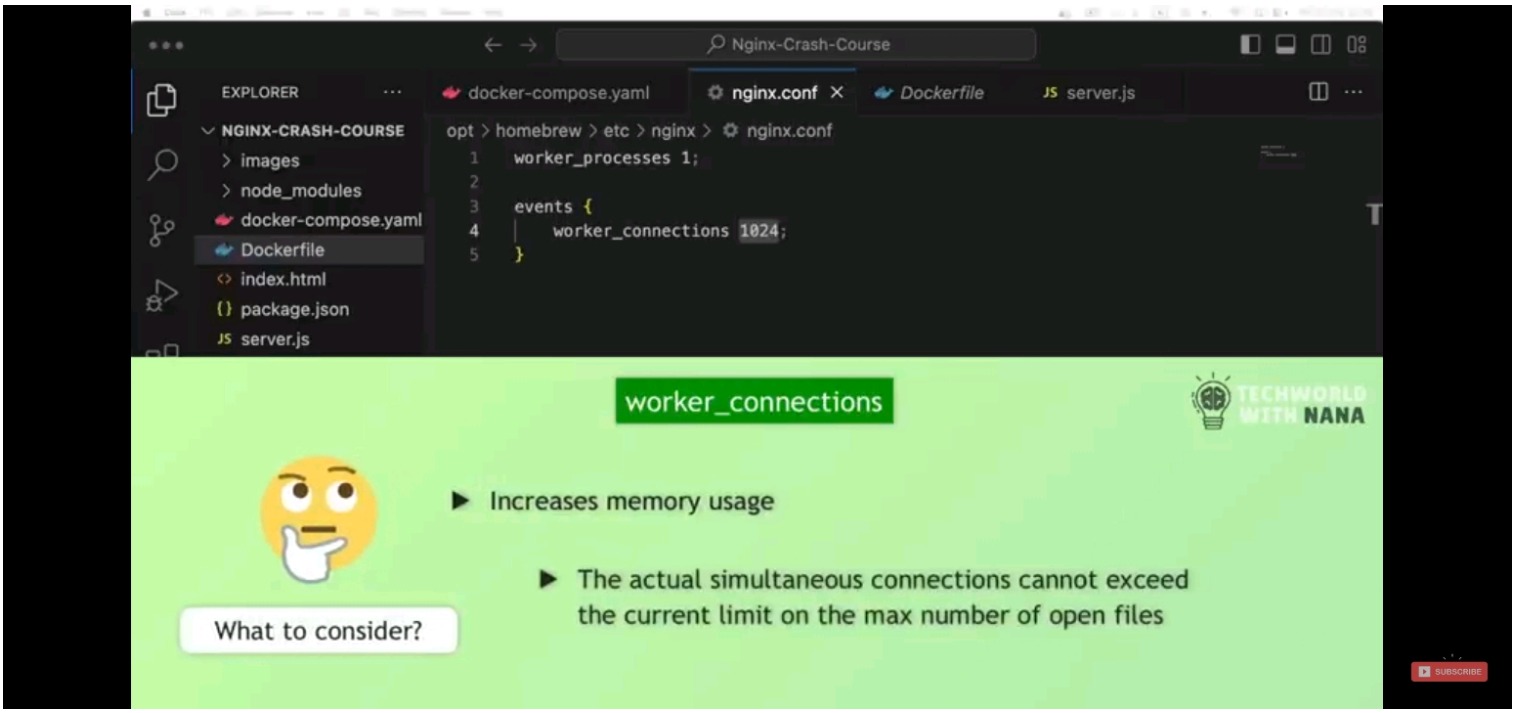

worker_processes
This configuration directly influences how well it can handle traffic (performance)
worker process
worker process
worker process
worker process
4 Worker Processes
CPU
CPU
CPU
CPU
4 CPU cores
Allows Nginx to efficiently use all CPU cores
View key concept
TECHWORLD WITH NANA
SUBSCRIBE

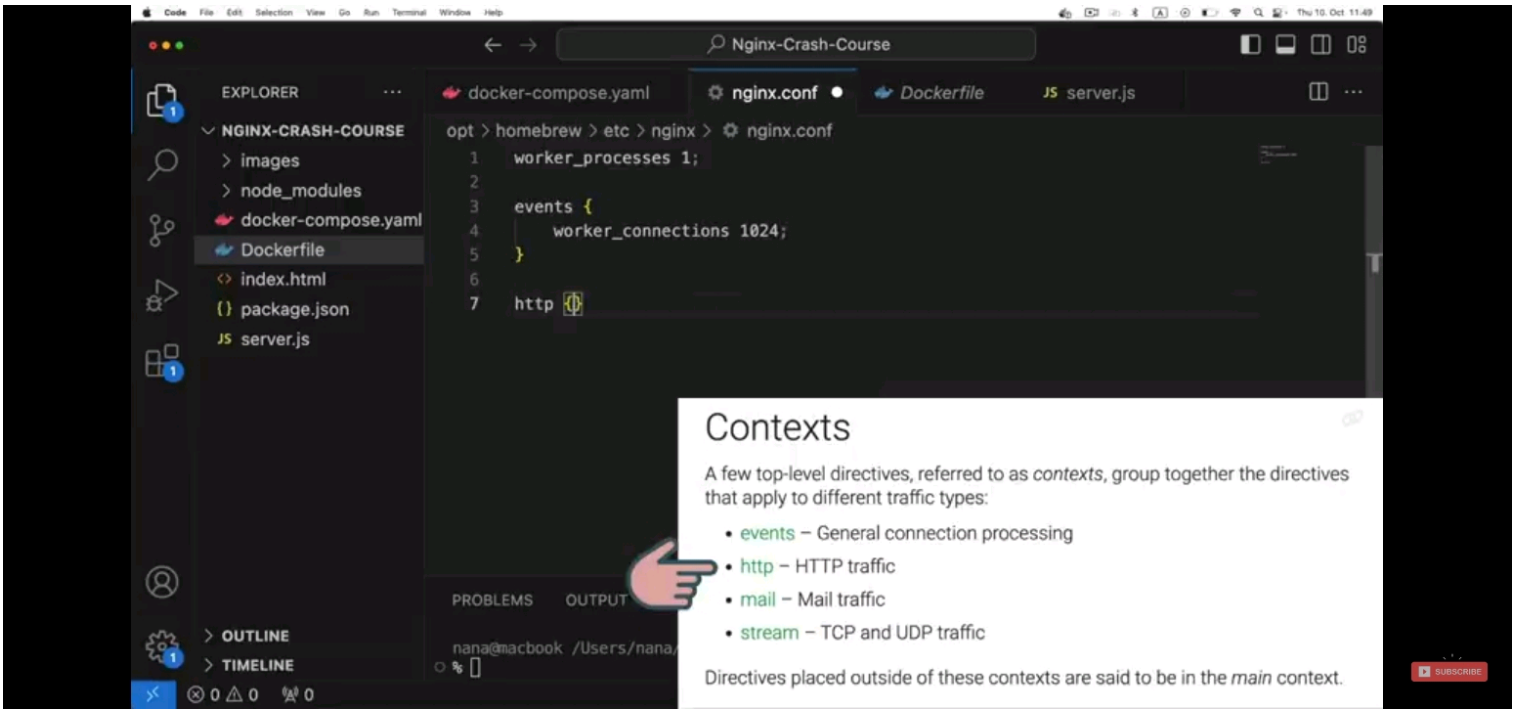

Nginx-Crash-Course
EXPLORER
NGINX-CRASH-COURSE
images
node_modules
docker-compose.yaml
Dockerfile
index.html
package.json
server.js
docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > nginx > nginx.conf
1 worker_processes auto;
auto =
Nginx automatically detects the number of CPU cores available on the server and starts a corresponding number of worker nodes
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
nana@macbook /Users/nana/Documents/Nginx-Crash-Course
% zsh
zsh
OUTLINE
TIMELINE
Ln 1, Col 22
Spaces: 4
UTF-8
LF
Properties
TECHWORLD WITH NANA
SUBSCRIBE

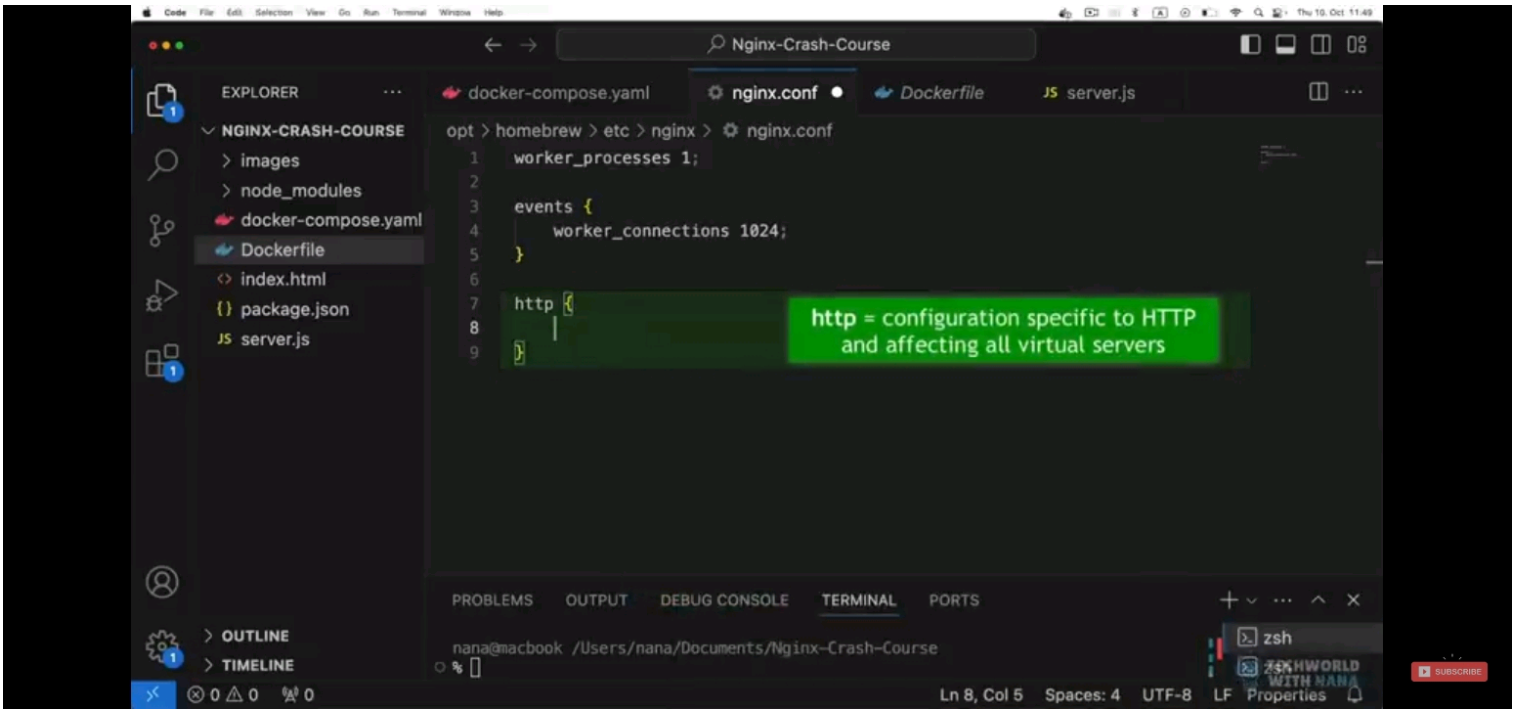

Nginx-Crash-Course
EXPLORER
NGINX-CRASH-COURSE
images
node_modules
docker-compose.yaml
Dockerfile
index.html
package.json
server.js
docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > nginx > nginx.conf
1 worker_processes 1;
2
3 events
worker_connections
TECHWORLD WITH NANA
Per worker process: how many simultaneous connections can be opened
worker process
worker process
Default = 512
SUBSCRIBE

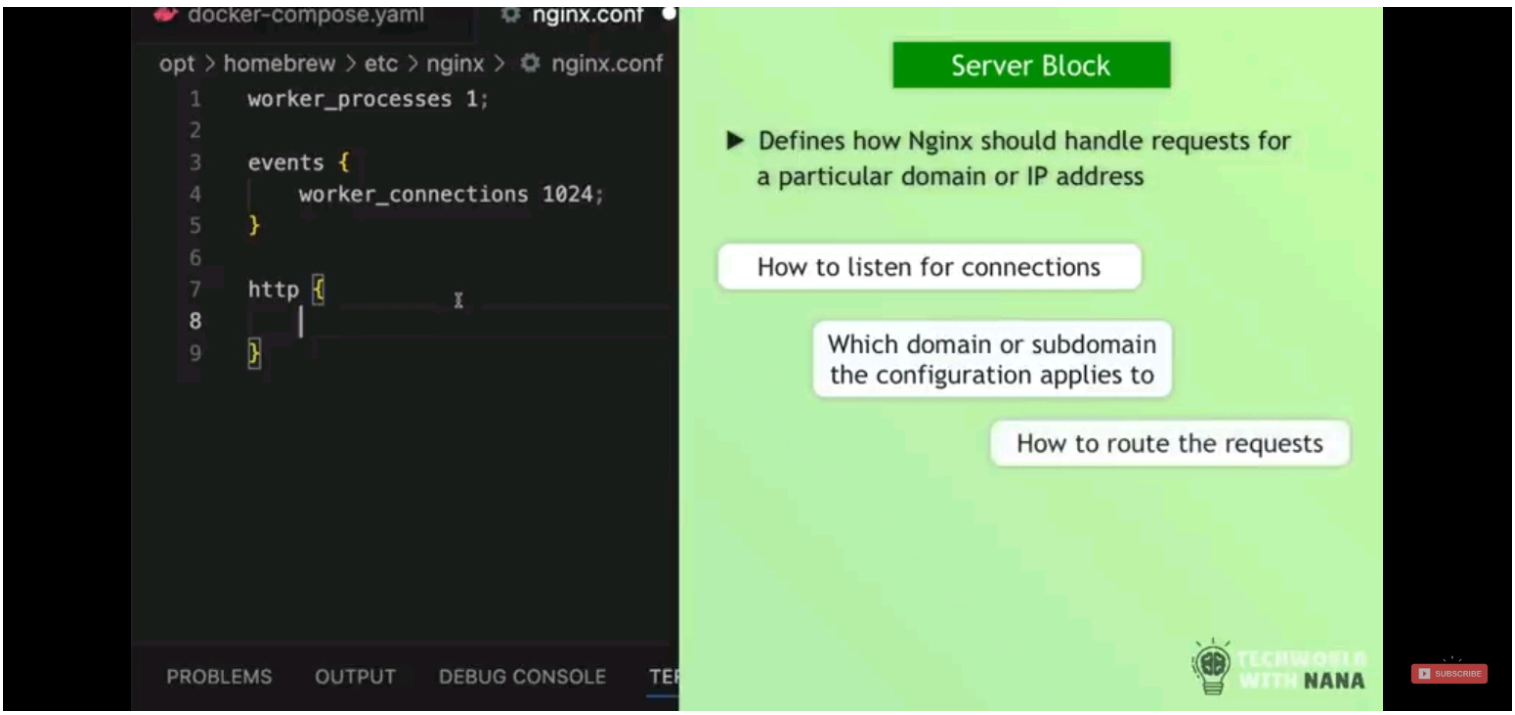

Nginx-Crash-Course
EXPLORER
NGINX-CRASH-COURSE
images
node_modules
docker-compose.yaml
Dockerfile
index.html
package.json
server.js
docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > nginx > nginx.conf
1 worker_processes 1;
2
3 events {
4 worker_connections 1024;
5 }
worker_connections
TECHWORLD WITH NANA
1024
worker process
If you have 1 worker process, you will be able to serve 1024 clients
If you have 2 worker processes, you will be able to serve 2x1024 = 2048 clients
SUBSCRIBE

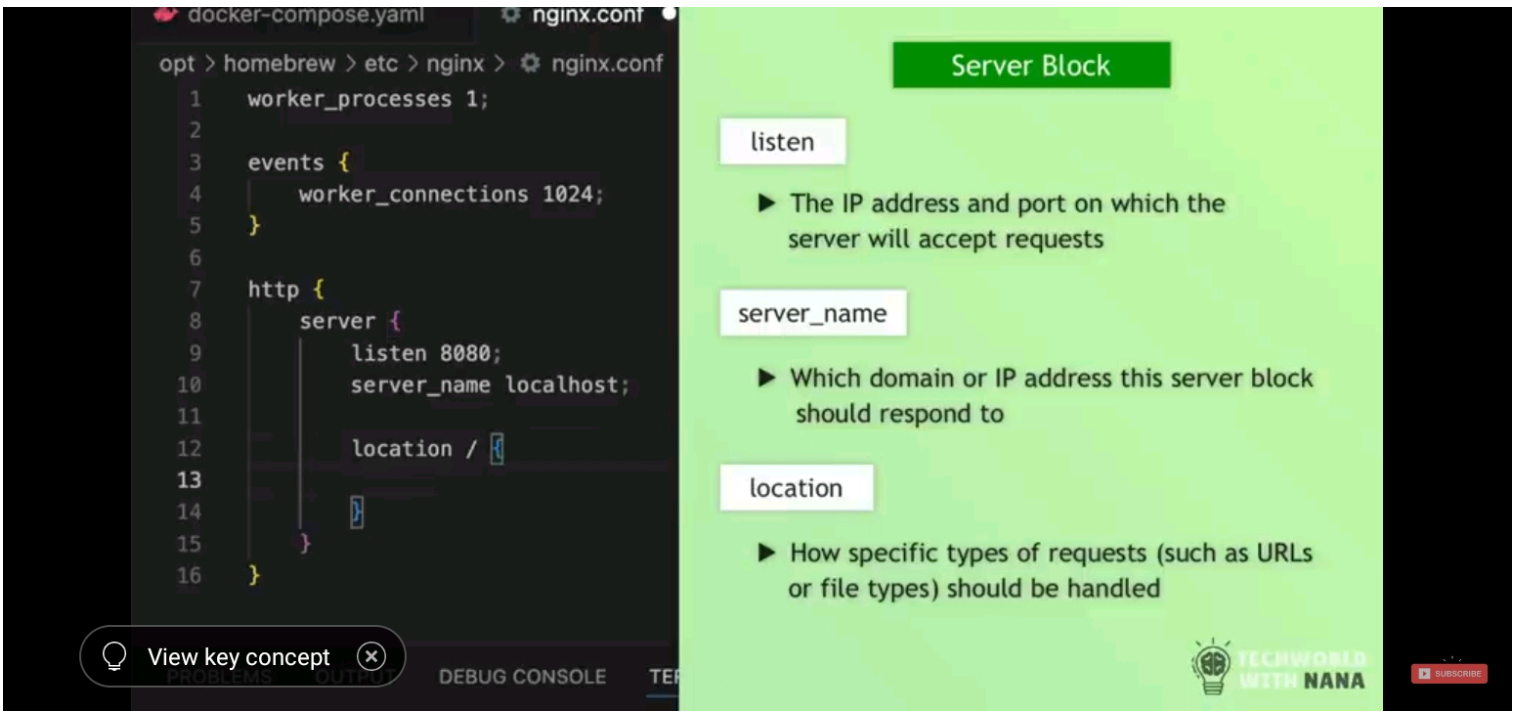

Nginx-Crash-Course
EXPLORER
NGINX-CRASH-COURSE
images
node_modules
docker-compose.yaml
Dockerfile
index.html
package.json
server.js
docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > nginx > nginx.conf
1 worker_processes 1;
2
3 events {
4 worker_connections 1024;
5 }
worker_connections
TECHWORLD WITH NANA
Increases memory usage
The higher the value,
the more requests can be handled
What to consider?
SUBSCRIBE

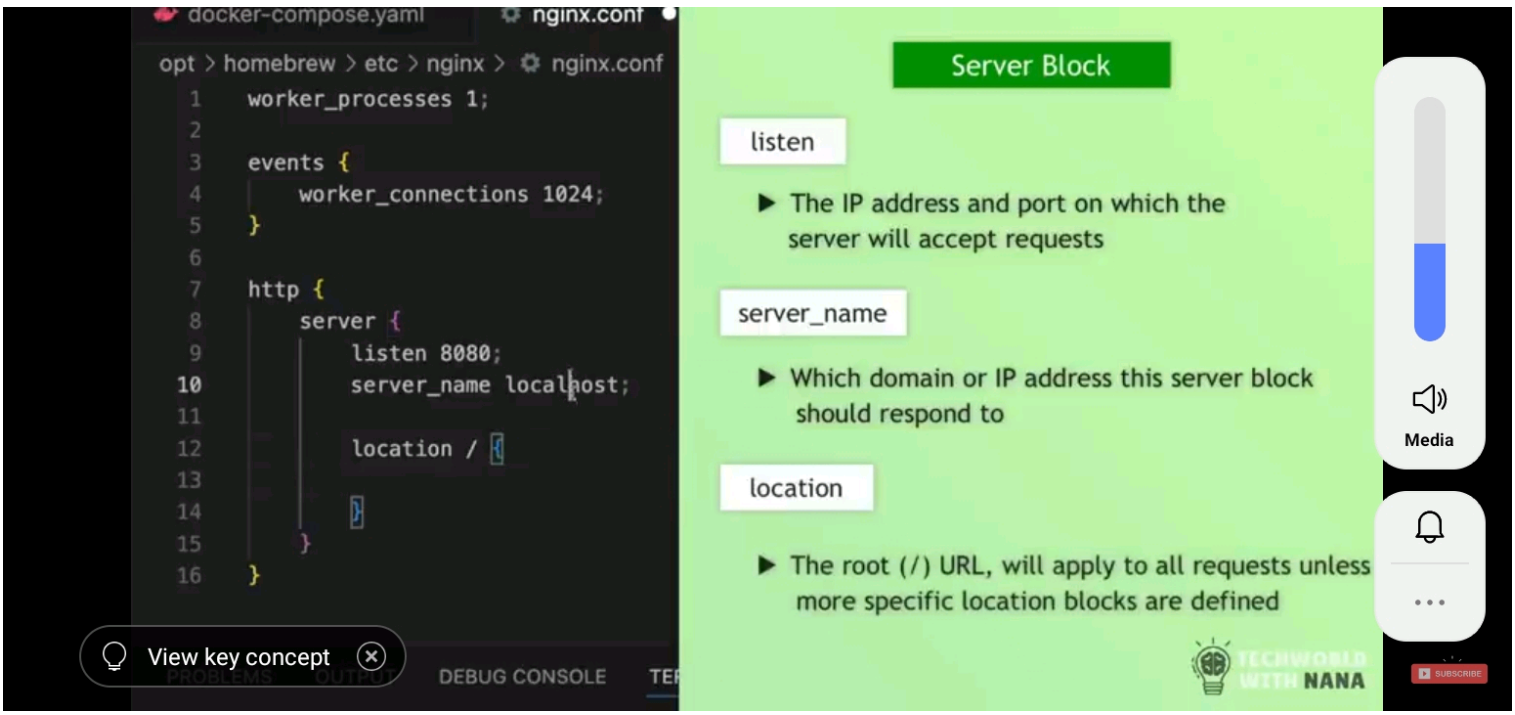

Nginx-Crash-Course
EXPLORER
NGINX-CRASH-COURSE
images
node_modules
docker-compose.yaml
Dockerfile
index.html
package.json
server.js
docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > nginx > nginx.conf
1 worker_processes 1;
2
3 events {
4 worker_connections 1024;
5 }
worker_connections
TECHWORLD WITH NANA
Increases memory usage
The actual simultaneous connections cannot exceed the current limit on the max number of open files
What to consider?
SUBSCRIBE


Nginx-Crash-Course
EXPLORER
NGINX-CRASH-COURSE
images
node_modules
docker-compose.yaml
Dockerfile
index.html
package.json
server.js
docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > nginx > nginx.conf
1 worker_processes 1;
2
3 events {
4 worker_connections 1024;
5 }
6
7 http
PROBLEMS
OUTPUT
OUTLINE
TIMELINE
Contexts
A few top-level directives, referred to as contexts, group together the directives that apply to different traffic types:
events – General connection processing
http – HTTP traffic
mail – Mail traffic
stream – TCP and UDP traffic
Directives placed outside of these contexts are said to be in the main context.


Nginx-Crash-Course
EXPLORER
NGINX-CRASH-COURSE
images
node_modules
docker-compose.yaml
Dockerfile
index.html
package.json
server.js
docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > nginx > nginx.conf
worker_processes 1;
events {
worker_connections 1024;
}
http {
}
http = configuration specific to HTTP and affecting all virtual servers
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
zsh
nana@macbook /Users/nana/Documents/Nginx-Crash-Course
OUTLINE
TIMELINE
Ln 8, Col 5
Spaces: 4
UTF-8
LF
Properties


docker-compose.yaml
nginx.conf
opt > homebrew > etc > nginx > nginx.conf
1 worker_processes 1;
2
3 events {
4 worker_connections 1024;
5 }
6
7 http {
8
9 }
PROBLEMS
OUTPUT
DEBUG CONSOLE
Server Block
Defines how Nginx should handle requests for a particular domain or IP address
How to listen for connections
Which domain or subdomain the configuration applies to
How to route the requests
TECHWORLD WITH NANA
SUBSCRIBE


docker-compose.yaml
nginx.conf
opt > homebrew > etc > nginx > nginx.conf
1 worker_processes 1;
2
3 events {
4 worker_connections 1024;
5 }
6
7 http {
8 server {
9 listen 8080;
10 server_name localhost;
11
12 location / {
13
14 }
15 }
16 }
View key concept
DEBUG CONSOLE
Server Block
listen
The IP address and port on which the server will accept requests
server_name
Which domain or IP address this server block should respond to
location
How specific types of requests (such as URLs or file types) should be handled
TECHWORLD WITH NANA
SUBSCRIBE


docker-compose.yaml
nginx.conf
opt > homebrew > etc > nginx > nginx.conf
1 worker_processes 1;
2
3 events {
4 worker_connections 1024;
5 }
6
7 http {
8 server {
9 listen 8080;
10 server_name localhost;
11
12 location / {
13
14 }
15 }
16 }
View key concept
DEBUG CONSOLE
Server Block
listen
▶ The IP address and port on which the server will accept requests
server_name
▶ Which domain or IP address this server block should respond to
location
▶ The root (/) URL, will apply to all requests unless more specific location blocks are defined
TECHWORLD WITH NANA
Media

docker-compose.yaml nginx.conf

opt > homebrew > etc > nginx > nginx.conf

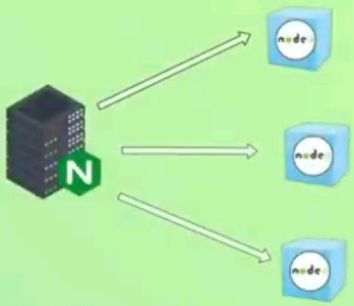
```
1   worker_processes 1;
2
3   events {
4       worker_connections 1024;
5   }
6
7   http {
8       server {
9           listen 8080;
10          server_name localhost;
11
12          location / {
13              proxy_pass
14          }
15      }
16  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TE

Location Block

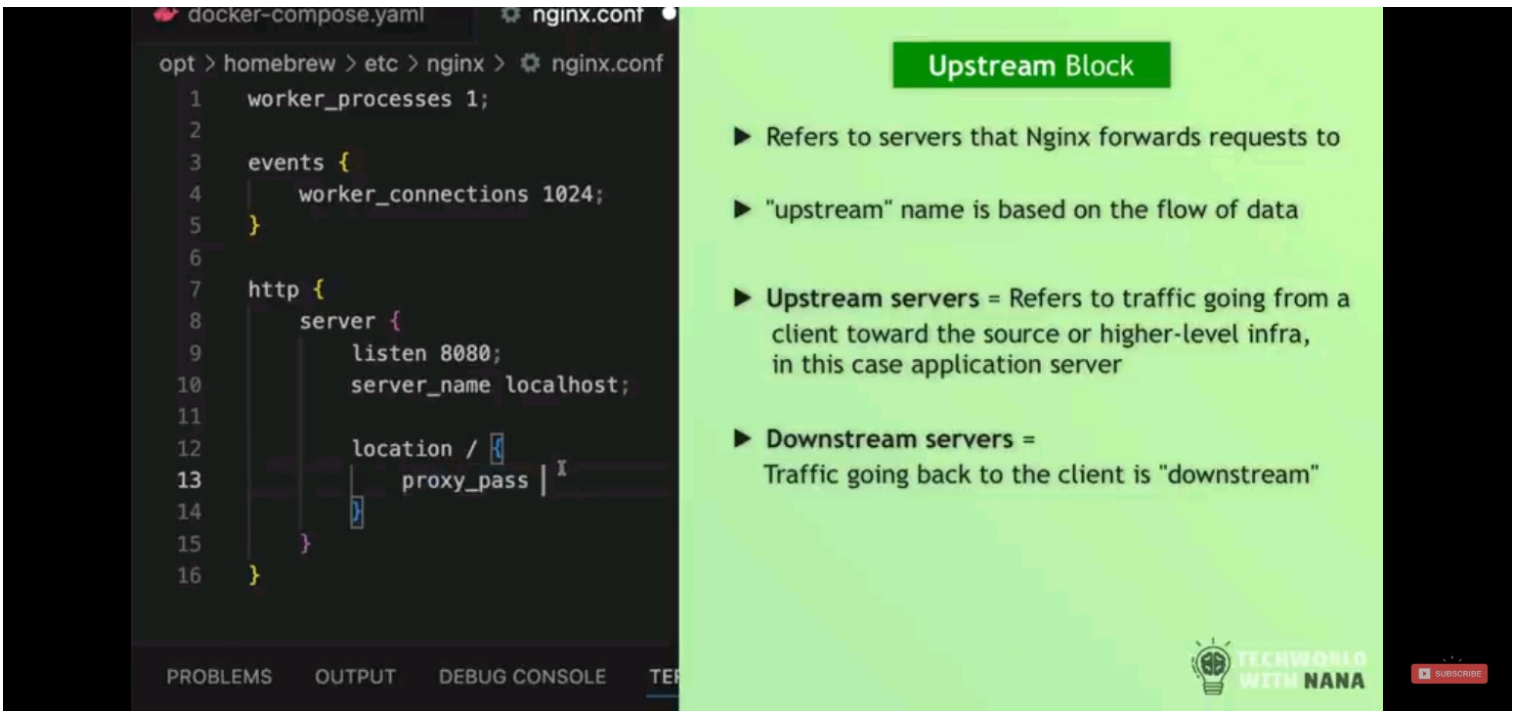
proxy_pass

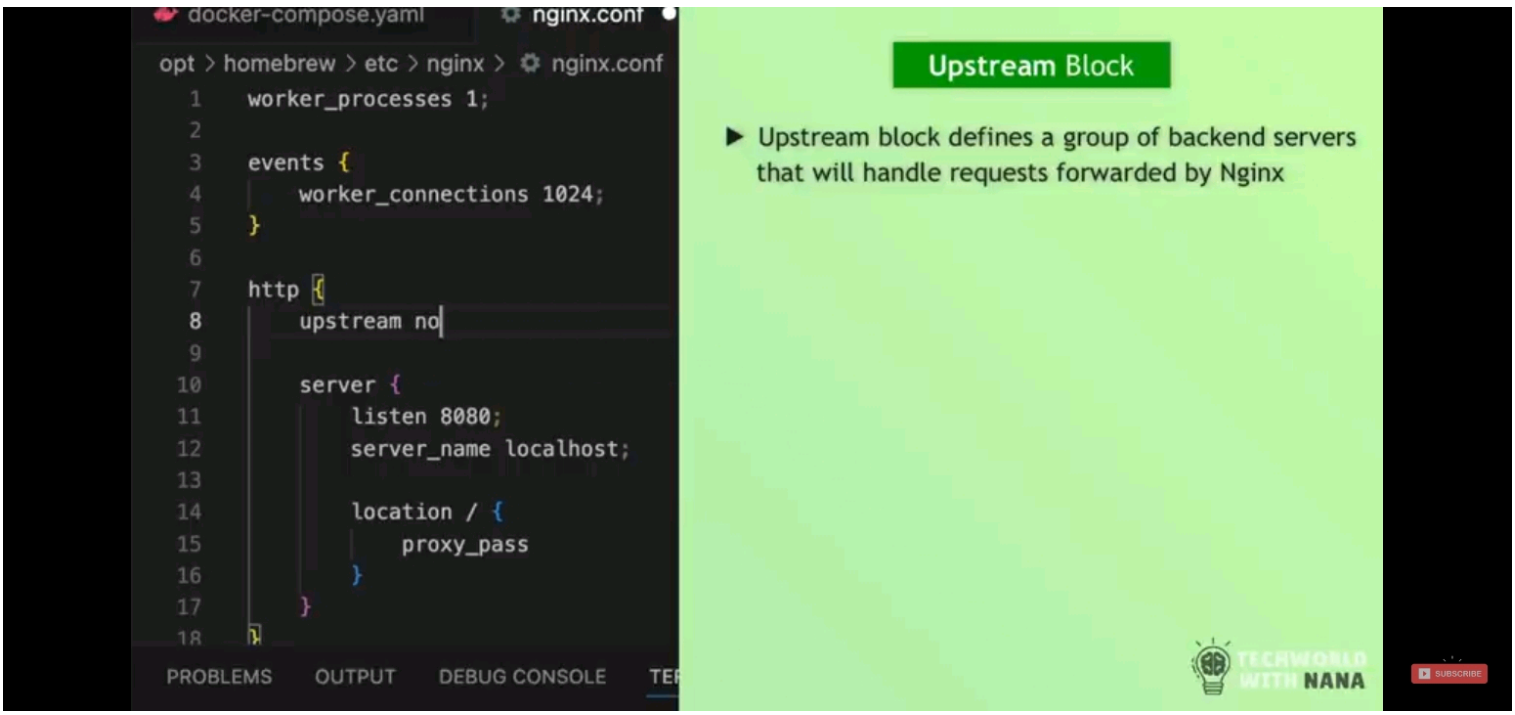
- Tells Nginx to "pass" the request to another server, making it act as a reverse proxy

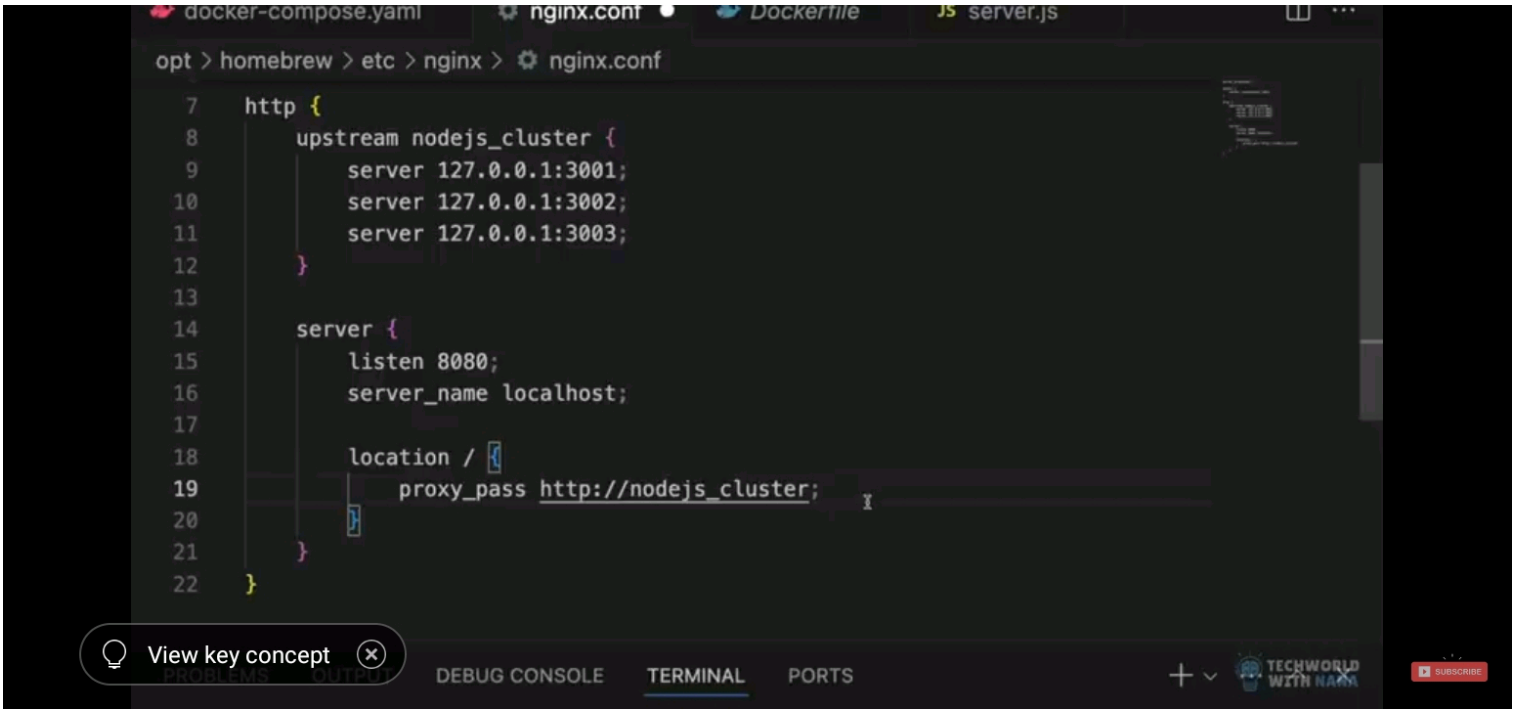


TECHWORLD WITH NANA

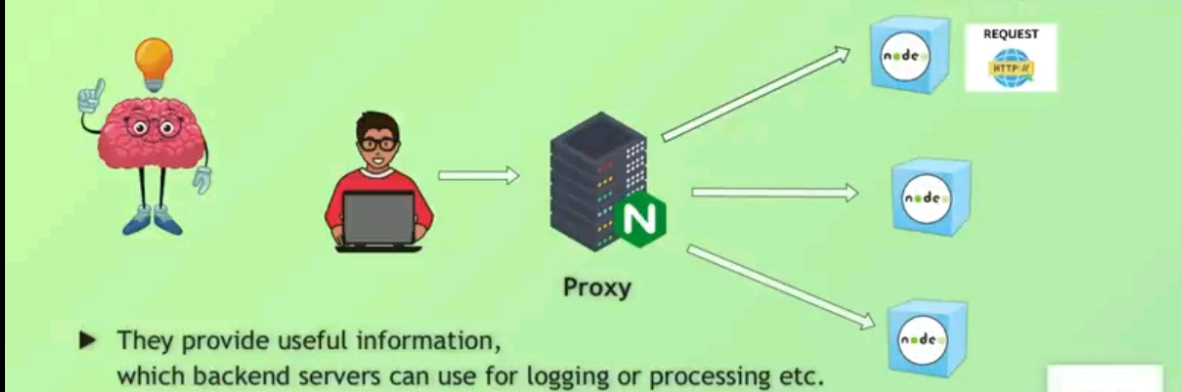
SUBSCRIBE


docker-compose.yaml
nginx.conf
opt > homebrew > etc > nginx > nginx.conf
1 worker_processes 1;
2
3 events {
4 worker_connections 1024;
5 }
6
7 http {
8 server {
9 listen 8080;
10 server_name localhost;
11
12 location / {
13 proxy_pass
14 }
15 }
16 }
PROBLEMS
OUTPUT
DEBUG CONSOLE
Upstream Block
Refers to servers that Nginx forwards requests to
"upstream" name is based on the flow of data
Upstream servers = Refers to traffic going from a client toward the source or higher-level infra, in this case application server
Downstream servers = Traffic going back to the client is "downstream"
TECHWORLD WITH NANA
SUBSCRIBE


docker-compose.yaml
nginx.conf
opt > homebrew > etc > nginx > nginx.conf
1 worker_processes 1;
2
3 events {
4 worker_connections 1024;
5 }
6
7 http {
8 upstream no
9
10 server {
11 listen 8080;
12 server_name localhost;
13
14 location / {
15 proxy_pass
16 }
17 }
18 }
PROBLEMS
OUTPUT
DEBUG CONSOLE
Upstream Block
▶ Upstream block defines a group of backend servers that will handle requests forwarded by Nginx
TECHWORLD WITH NANA
SUBSCRIBE

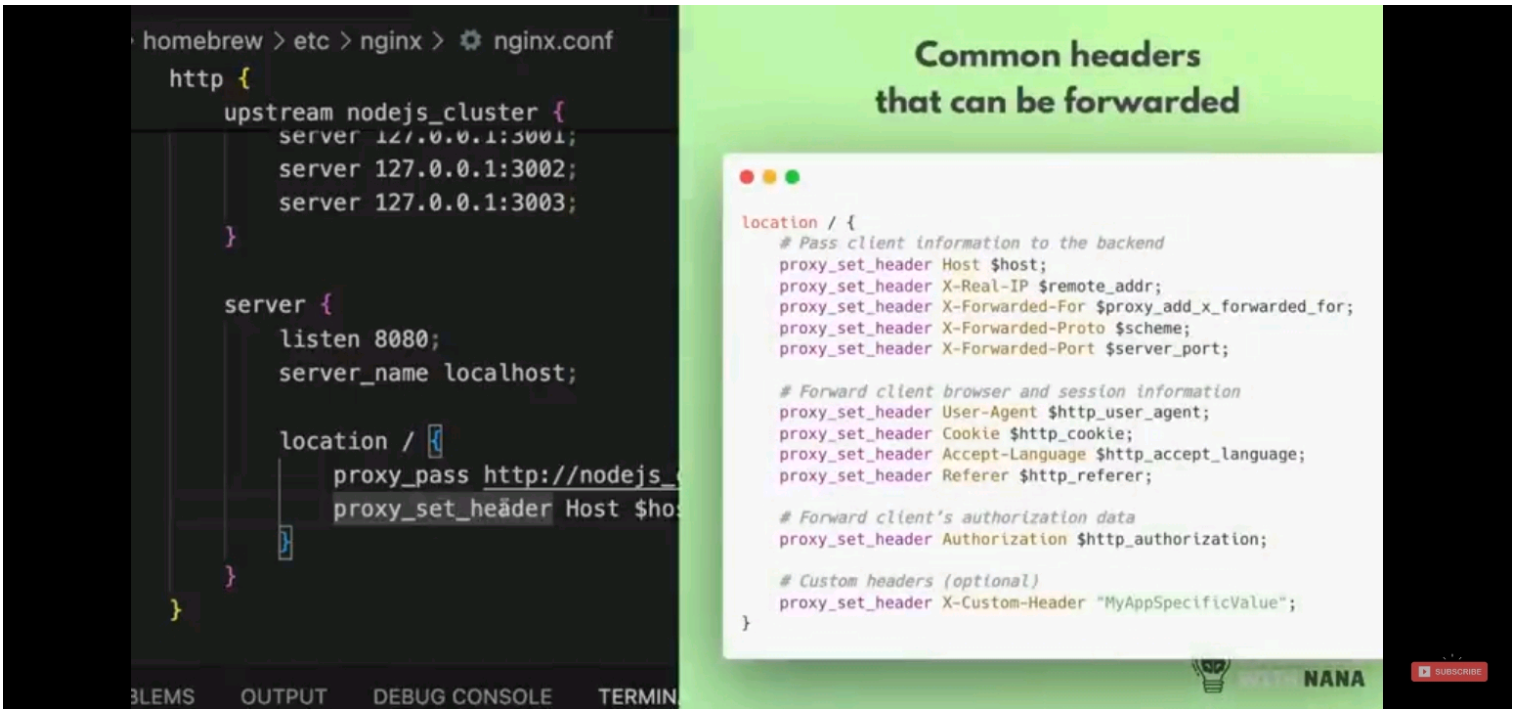

docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > nginx > nginx.conf
http {
upstream nodejs_cluster {
server 127.0.0.1:3001;
server 127.0.0.1:3002;
server 127.0.0.1:3003;
}
server {
listen 8080;
server_name localhost;
location / {
proxy_pass http://nodejs_cluster;
}
}
}
View key concept
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
TECHWORLD WITH NANA
SUBSCRIBE

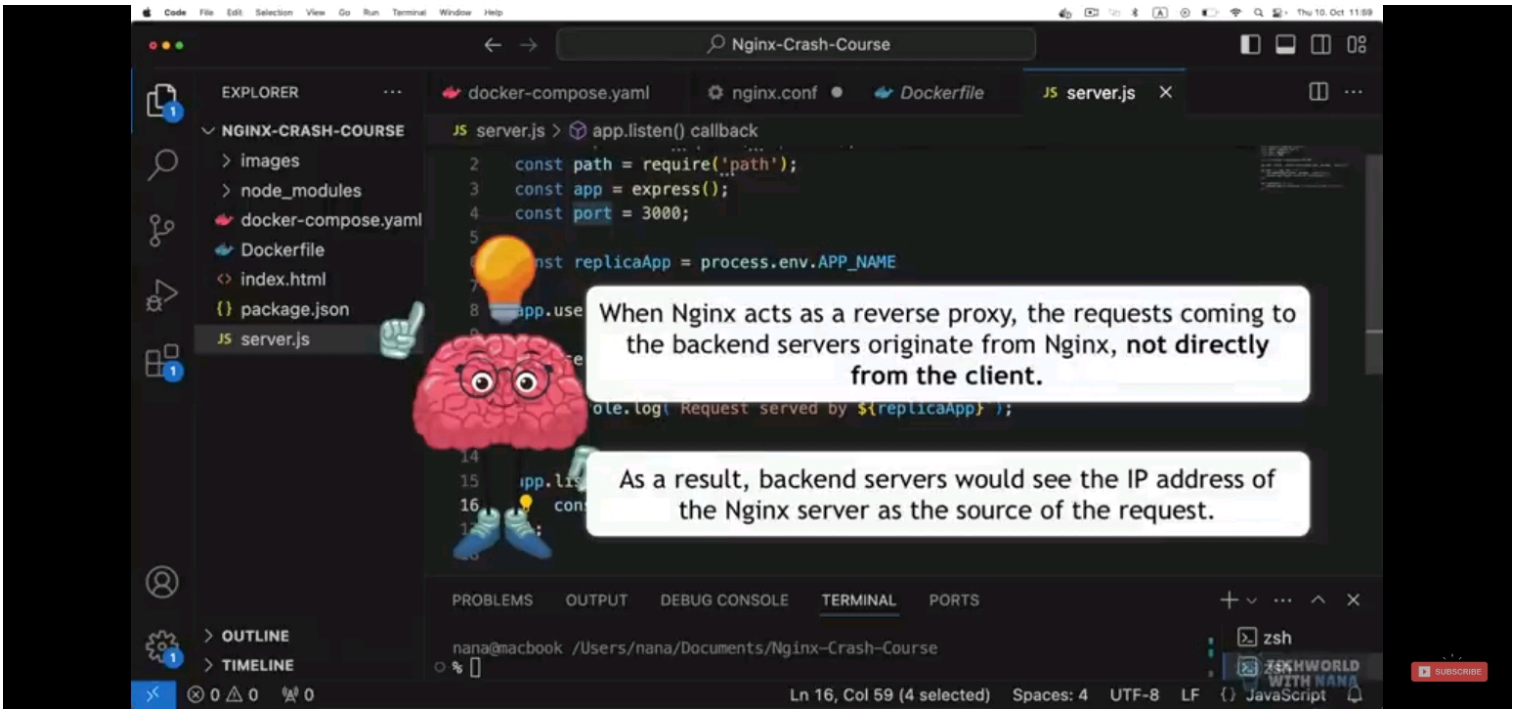
We want to forward info from the original client requests to the backend servers

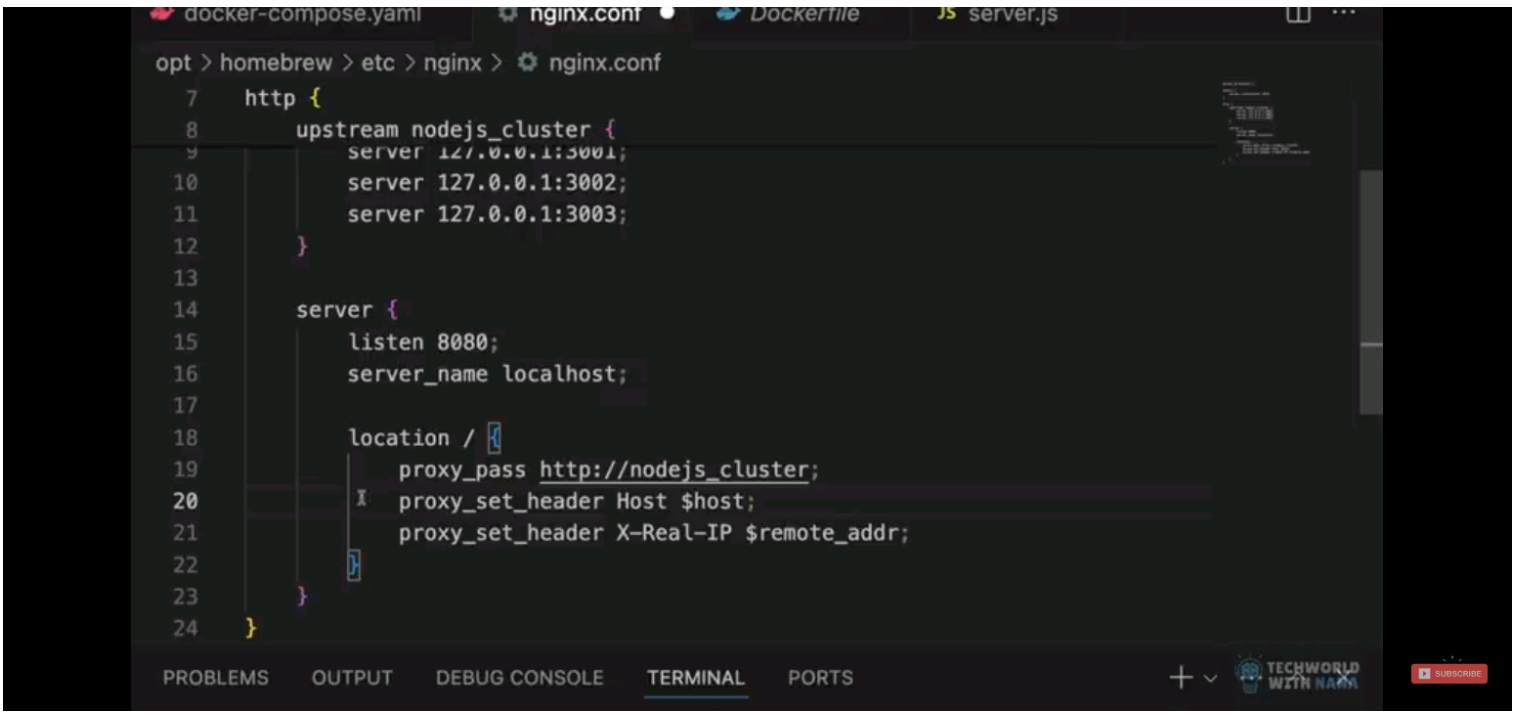


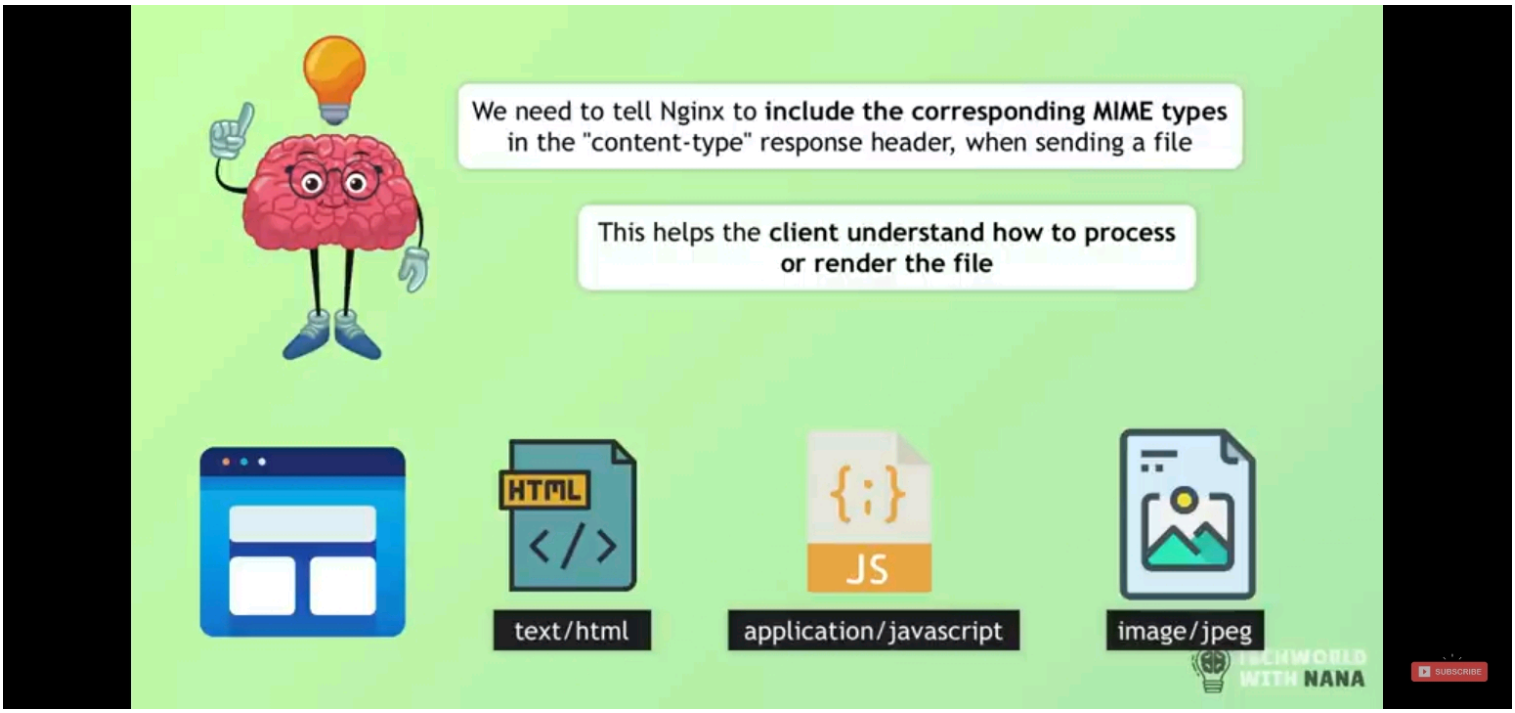
- They provide useful information, which backend servers can use for logging or processing etc.

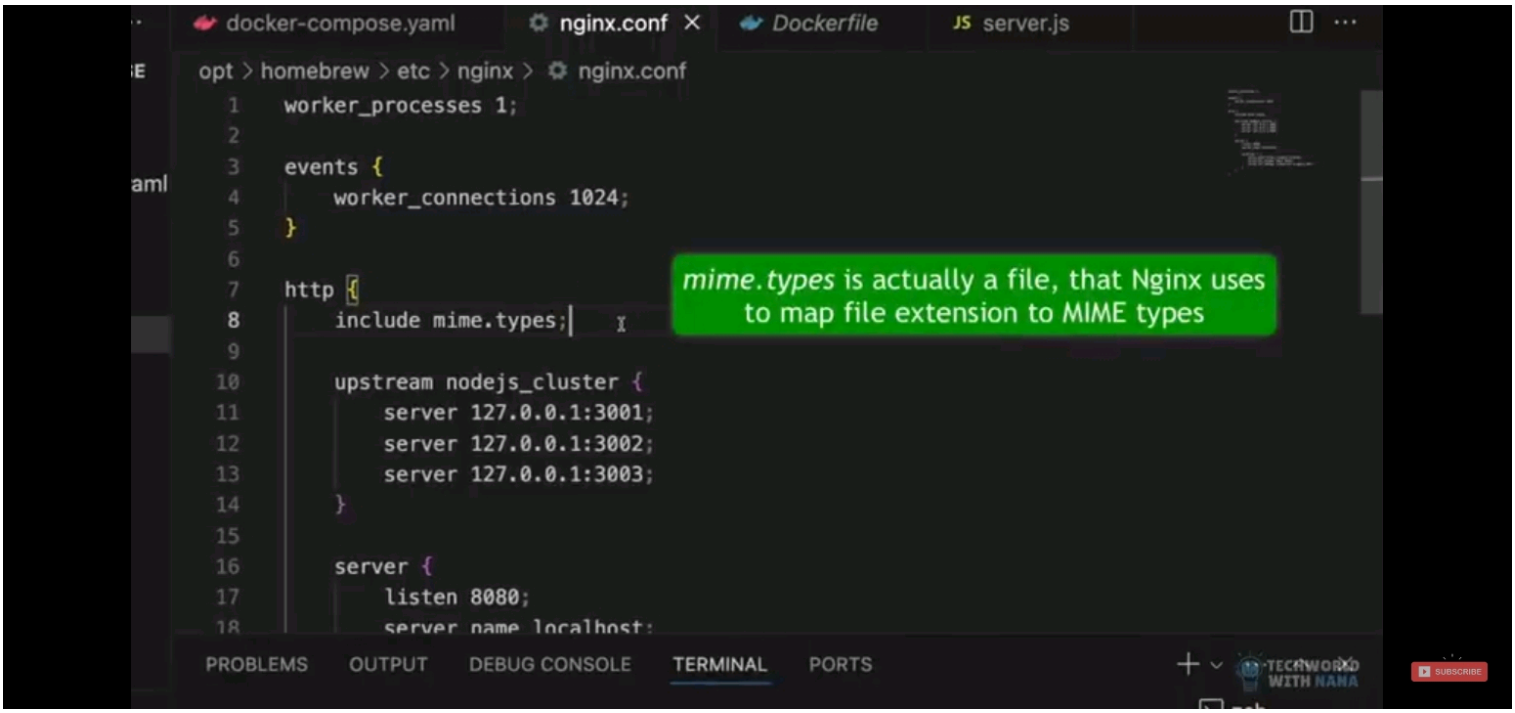
Original **IP address** | Original **Host** | Referrer Information | Custom Headers | ...


homebrew > etc > nginx > nginx.conf
http {
upstream nodejs_cluster {
server 127.0.0.1:3002;
server 127.0.0.1:3003;
}
server {
listen 8080;
server_name localhost;
location / {
proxy_pass http://nodejs_
proxy_set_header Host $ho
}
}
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMIN
Common headers
that can be forwarded
location / {
# Pass client information to the backend
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;
proxy_set_header X-Forwarded-Port $server_port;
Forward client browser and session information
proxy_set_header User-Agent $http_user_agent;
proxy_set_header Cookie $http_cookie;
proxy_set_header Accept-Language $http_accept_language;
proxy_set_header Referer $http_referer;
Forward client's authorization data
proxy_set_header Authorization $http_authorization;
Custom headers (optional)
proxy_set_header X-Custom-Header "MyAppSpecificValue";
}
NANA
SUBSCRIBE

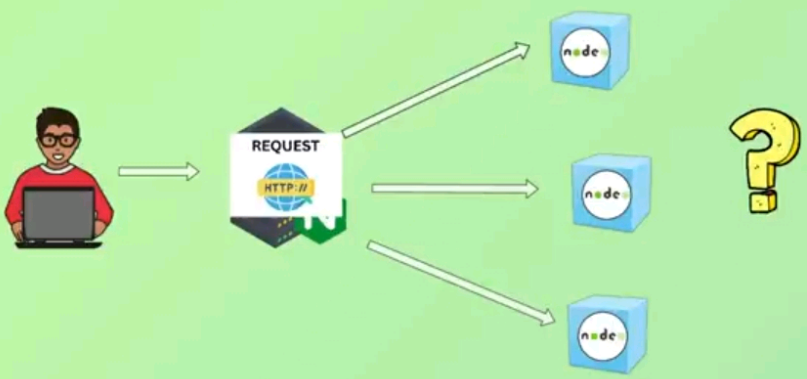

Nginx-Crash-Course
EXPLORER
NGINX-CRASH-COURSE
images
node_modules
docker-compose.yaml
Dockerfile
index.html
package.json
server.js
docker-compose.yaml
nginx.conf
Dockerfile
server.js
server.js > app.listen() callback
const path = require('path');
const app = express();
const port = 3000;
const replicaApp = process.env.APP_NAME
When Nginx acts as a reverse proxy, the requests coming to the backend servers originate from Nginx, not directly from the client.
As a result, backend servers would see the IP address of the Nginx server as the source of the request.
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
nana@macbook /Users/nana/Documents/Nginx-Crash-Course
zsh
OUTLINE
TIMELINE
Ln 16, Col 59 (4 selected)
Spaces: 4
UTF-8
LF
JavaScript
TECHWORLD WITH NANA
SUBSCRIBE

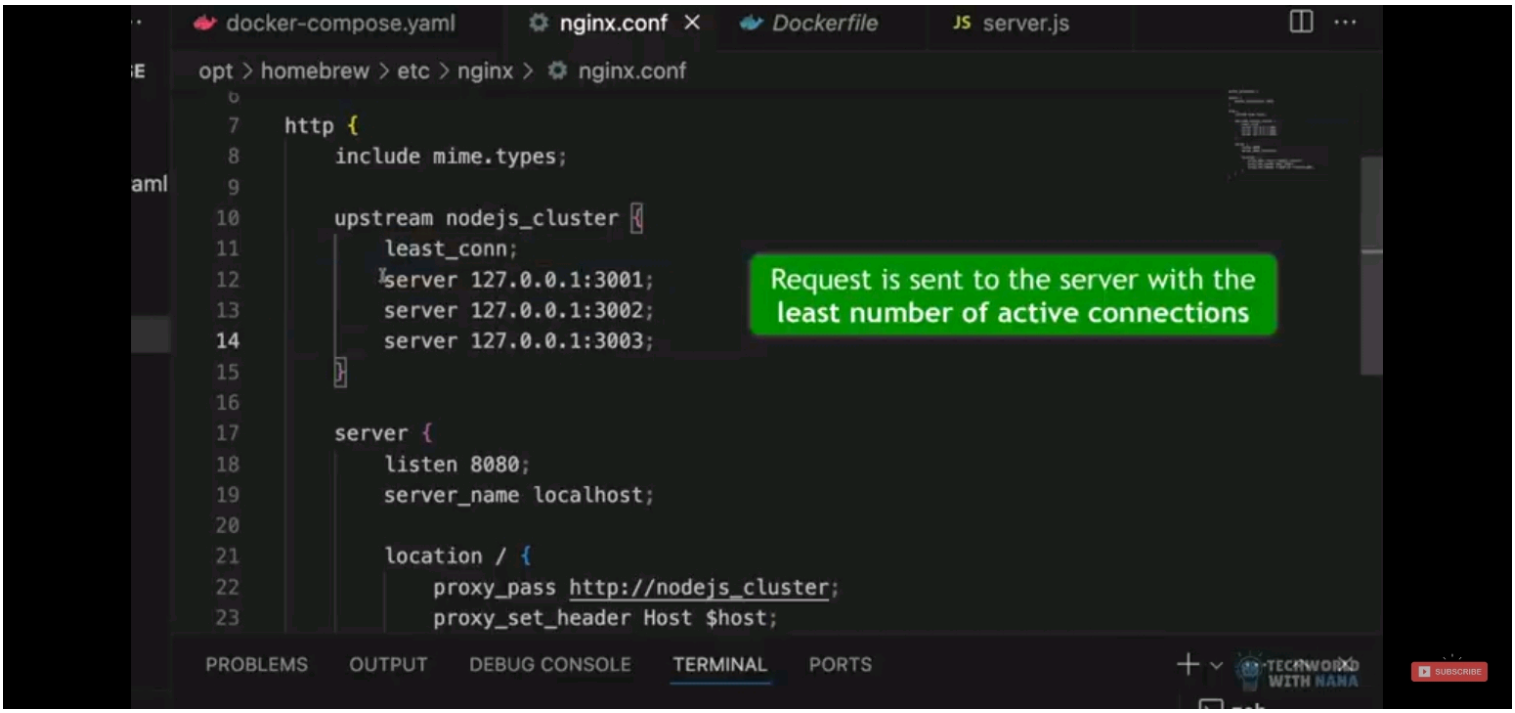

docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > nginx > nginx.conf
http {
upstream nodejs_cluster {
server 127.0.0.1:3001;
server 127.0.0.1:3002;
server 127.0.0.1:3003;
}
server {
listen 8080;
server_name localhost;
location / {
proxy_pass http://nodejs_cluster;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
}
}
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
TECHWORLD WITH NANA
SUBSCRIBE

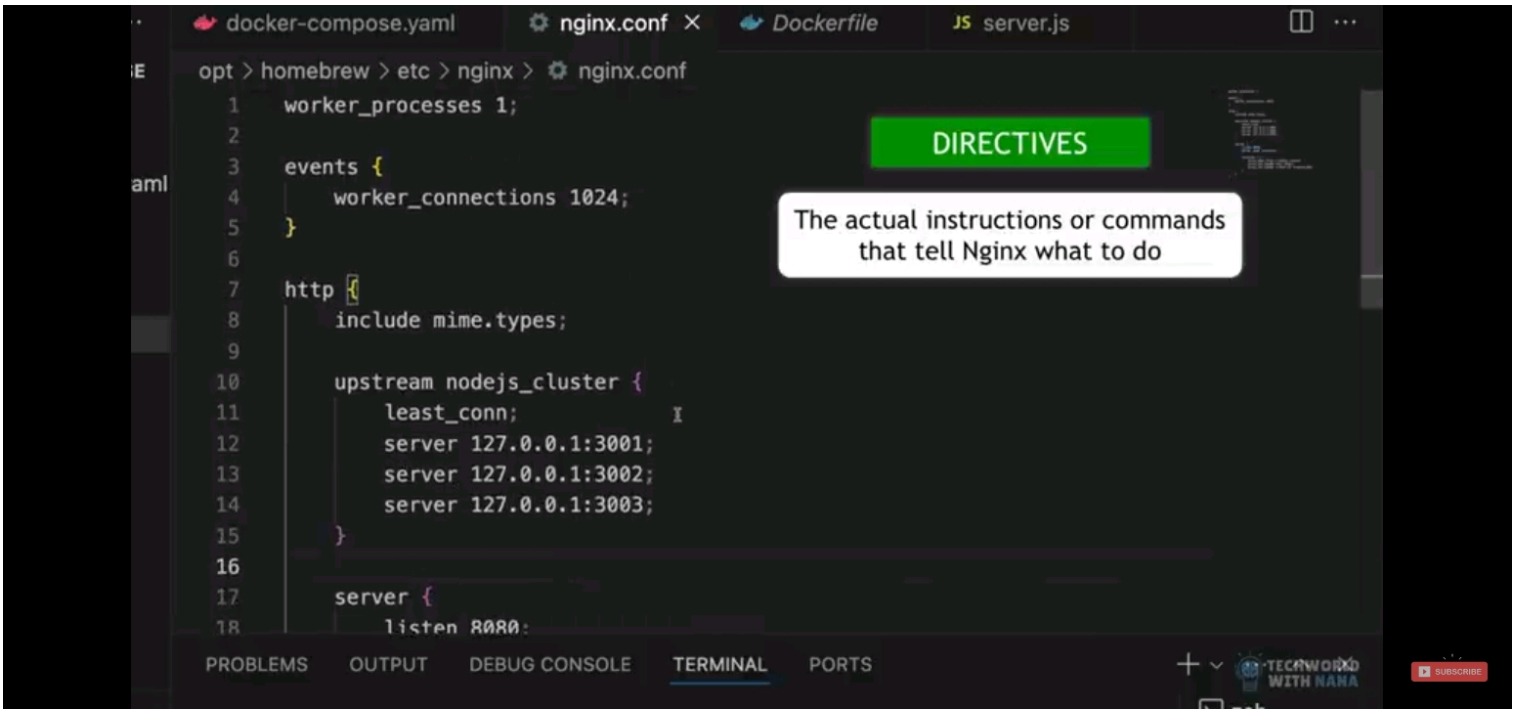

We need to tell Nginx to include the corresponding MIME types in the "content-type" response header, when sending a file
This helps the client understand how to process or render the file
HTML
text/html
JS
application/javascript
image/jpeg
TECHWORLD WITH NANA
SUBSCRIBE

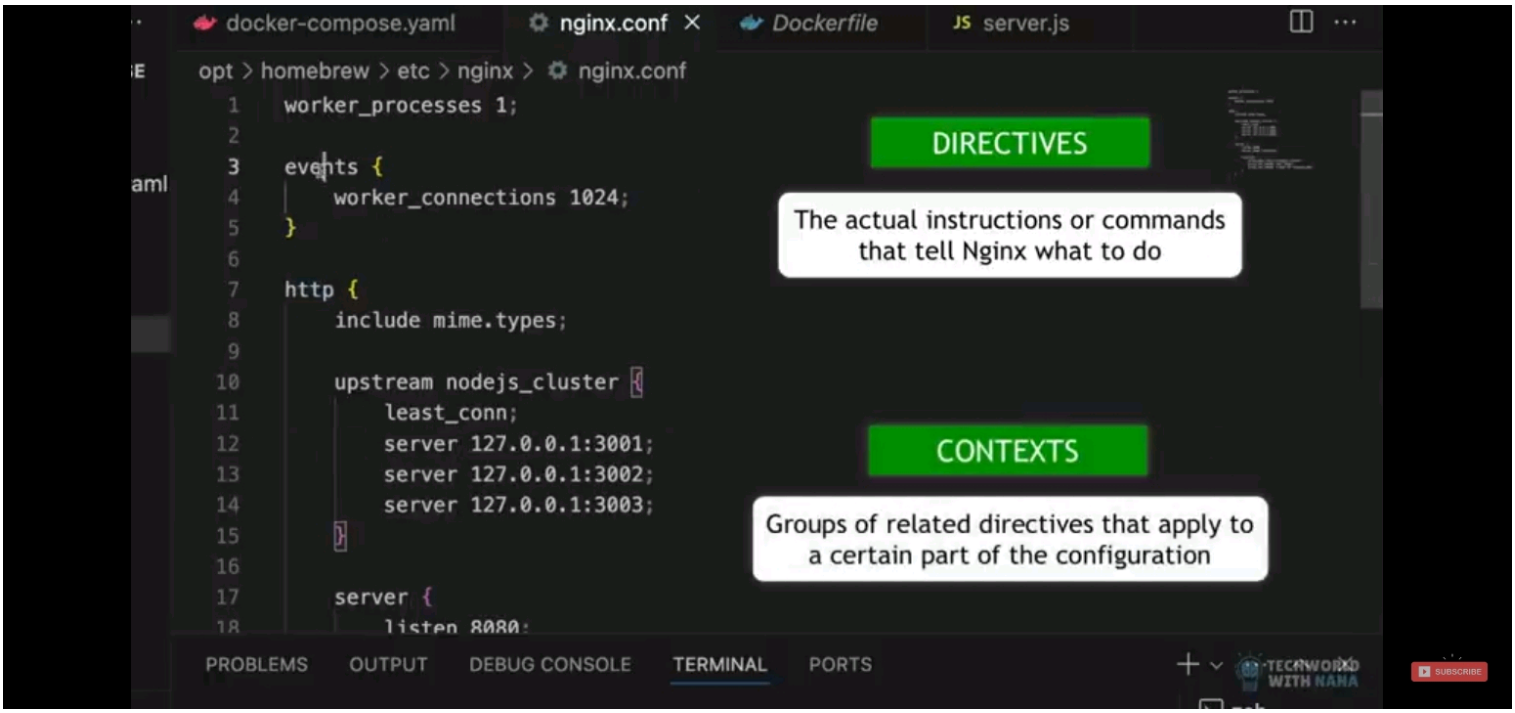

docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > nginx > nginx.conf
worker_processes 1;
events {
worker_connections 1024;
}
http {
include mime.types;
upstream nodejs_cluster {
server 127.0.0.1:3001;
server 127.0.0.1:3002;
server 127.0.0.1:3003;
}
server {
listen 8080;
mime.types is actually a file, that Nginx uses to map file extension to MIME types
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
TECHWORLD WITH NANA
SUBSCRIBE

How does Nginx decide to which server to forward the request to?



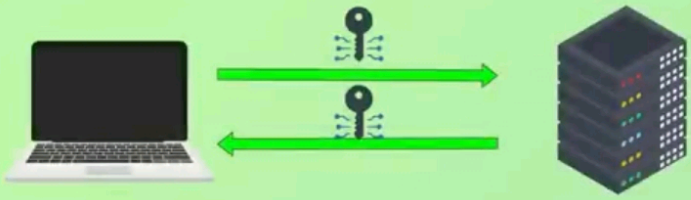

docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > nginx > nginx.conf
http {
include mime.types;
upstream nodejs_cluster {
least_conn;
server 127.0.0.1:3001;
server 127.0.0.1:3002;
server 127.0.0.1:3003;
}
server {
listen 8080;
server_name localhost;
location / {
proxy_pass http://nodejs_cluster;
proxy_set_header Host $host;
Request is sent to the server with the least number of active connections
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
TECHWORLD WITH NANA
SUBSCRIBE


docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > nginx > nginx.conf
worker_processes 1;
events {
worker_connections 1024;
}
http {
include mime.types;
upstream nodejs_cluster {
least_conn;
server 127.0.0.1:3001;
server 127.0.0.1:3002;
server 127.0.0.1:3003;
}
server {
listen 8080;
DIRECTIVES
The actual instructions or commands that tell Nginx what to do
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
TECHWORLD WITH NANA
SUBSCRIBE


docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > nginx > nginx.conf
worker_processes 1;
events {
worker_connections 1024;
}
http {
include mime.types;
upstream nodejs_cluster {
least_conn;
server 127.0.0.1:3001;
server 127.0.0.1:3002;
server 127.0.0.1:3003;
}
server {
listen 8080;
DIRECTIVES
The actual instructions or commands that tell Nginx what to do
CONTEXTS
Groups of related directives that apply to a certain part of the configuration
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
TECHWORLD WITH NANA
SUBSCRIBE

Configure HTTPS - Encrypted Communication

- HTTPS uses SSL to encrypt the data transmitted over the web
- All communication between the client and server is **encrypted**



View key concept

TECHWORLD WITH NANA

SUBSCRIBE

Configure HTTPS - Encrypted Communication

- HTTPS uses SSL to encrypt the data transmitted over the web
- All communication between the client and server is **encrypted**
- Even if someone intercepts the data, they **cannot read it**

1) Obtain an SSL/TLS Certificate

- SSL certificates enable encryption by using **public-key cryptography**
- When a user connects to a website via HTTPs, the **web server provides its SSL certificate, which contains a public key**

The client (**browser**) uses this key to **establish a secure**, encrypted session with the server

1) Obtain an SSL/TLS Certificate

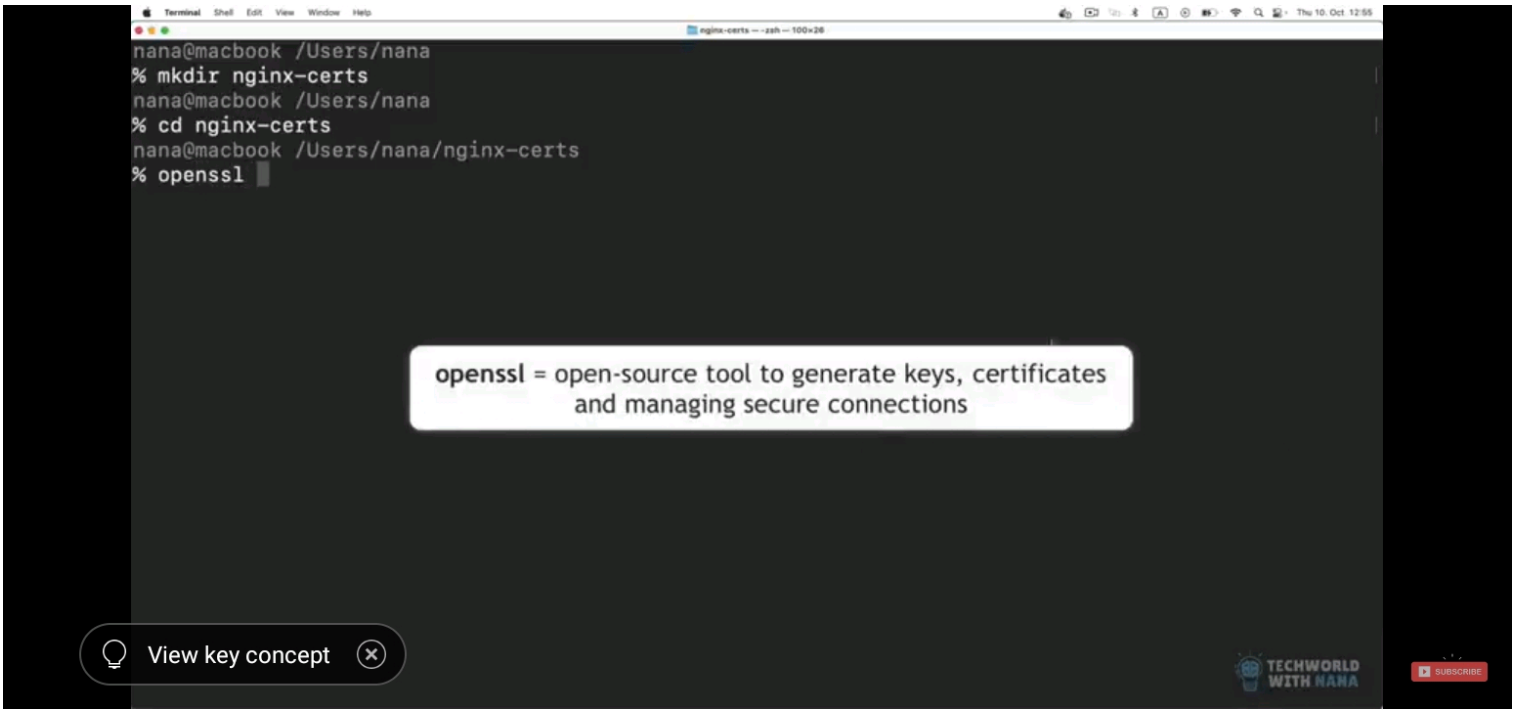
Generate a **self-signed certificate**

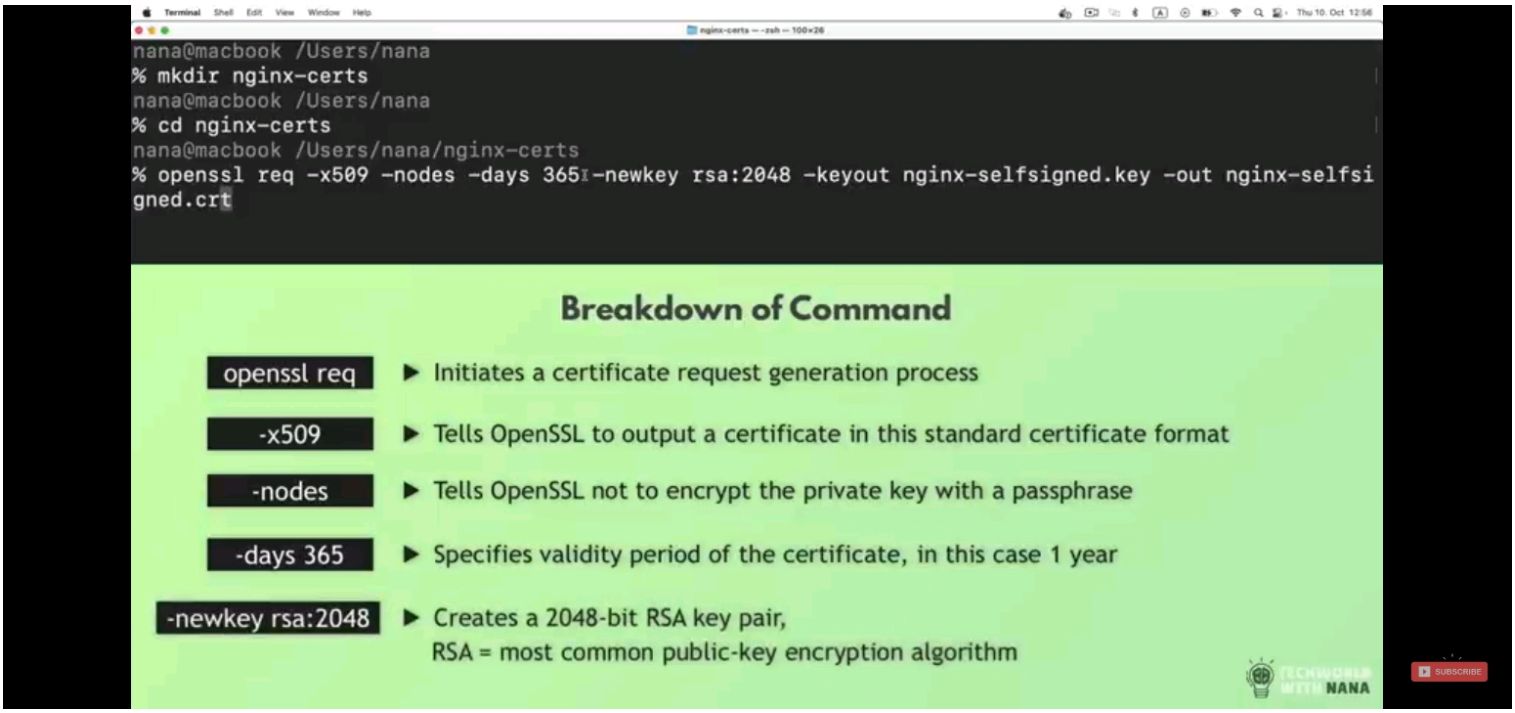
- Generated and signed by the server itself

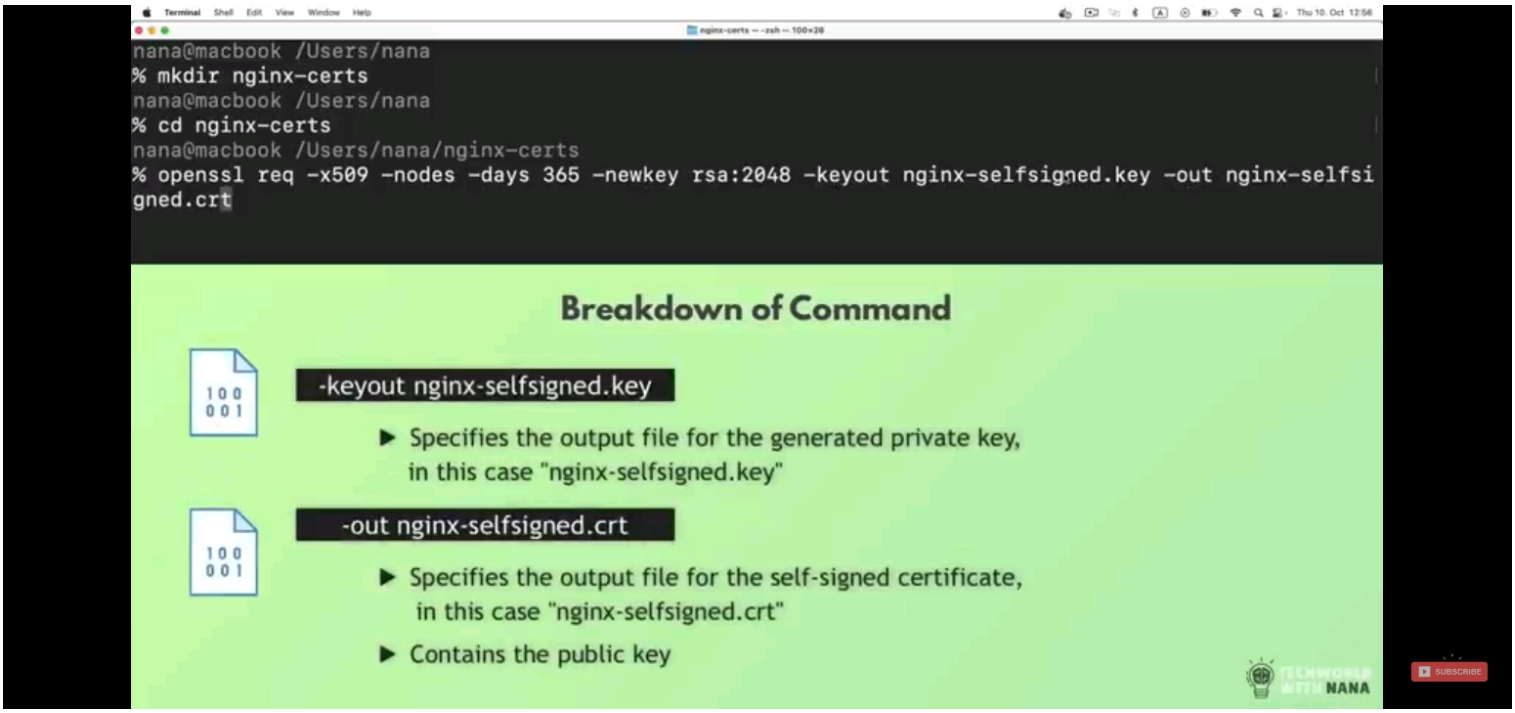
CA-Signed Certificate

- Issued and authenticated by a trusted certificate authority
- CA **verifies the identity** of the organization requesting the certificate

✅ Trusted by browsers (no warnings)


Terminal Shell Edit View Window Help
Thu 10. Oct 12:55
nginx-certs — -zsh — 100×26
nana@macbook /Users/nana
% mkdir nginx-certs
nana@macbook /Users/nana
% cd nginx-certs
nana@macbook /Users/nana/nginx-certs
% openssl
openssl = open-source tool to generate keys, certificates and managing secure connections
View key concept
TECHWORLD WITH NANA
SUBSCRIBE

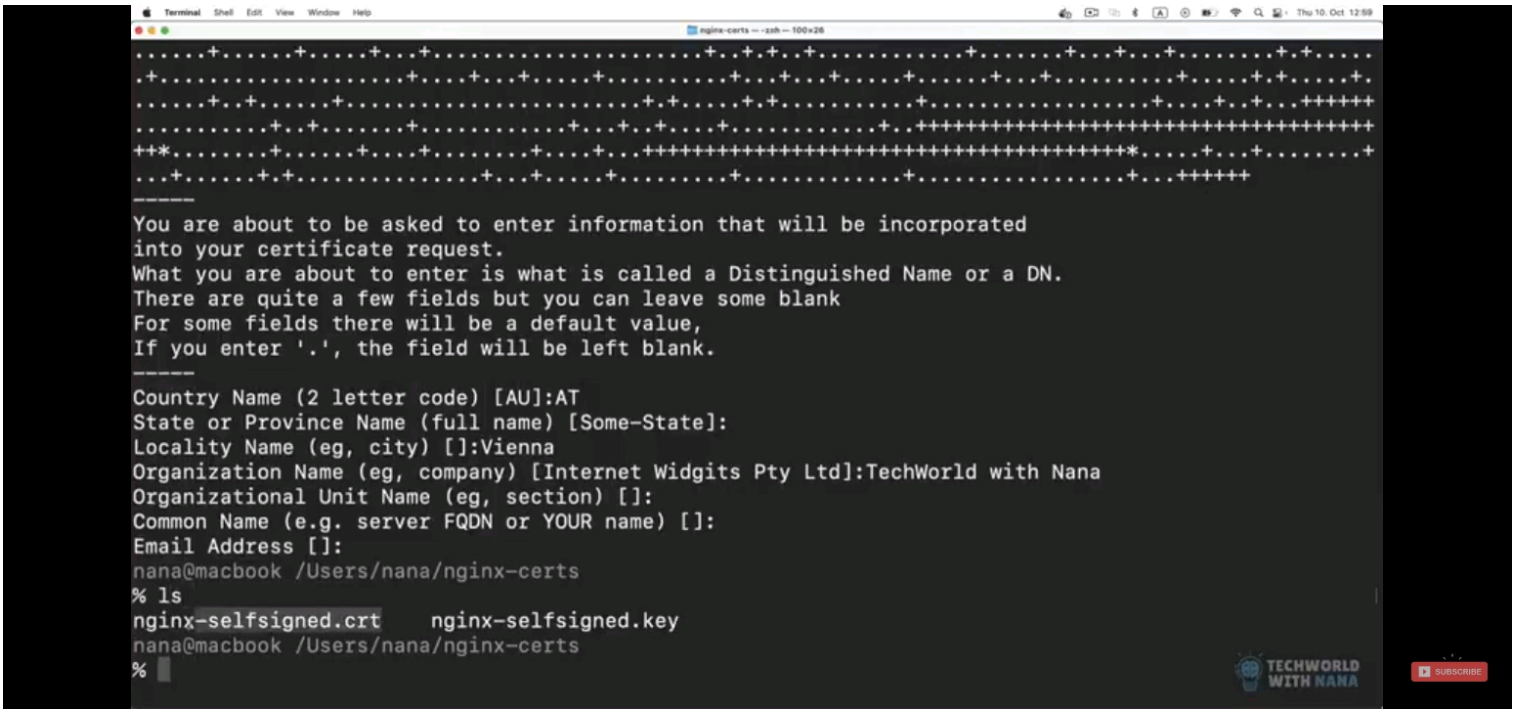

Terminal Shell Edit View Window Help
Thu 10. Oct 12:56
nginx-certs — -zsh — 100×26
nana@macbook /Users/nana
% mkdir nginx-certs
nana@macbook /Users/nana
% cd nginx-certs
nana@macbook /Users/nana/nginx-certs
% openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout nginx-selfsigned.key -out nginx-selfsigned.crt
Breakdown of Command
openssl req
Initiates a certificate request generation process
-x509
Tells OpenSSL to output a certificate in this standard certificate format
-nodes
Tells OpenSSL not to encrypt the private key with a passphrase
-days 365
Specifies validity period of the certificate, in this case 1 year
-newkey rsa:2048
Creates a 2048-bit RSA key pair,
RSA = most common public-key encryption algorithm
TECHWORLD WITH NANA
SUBSCRIBE

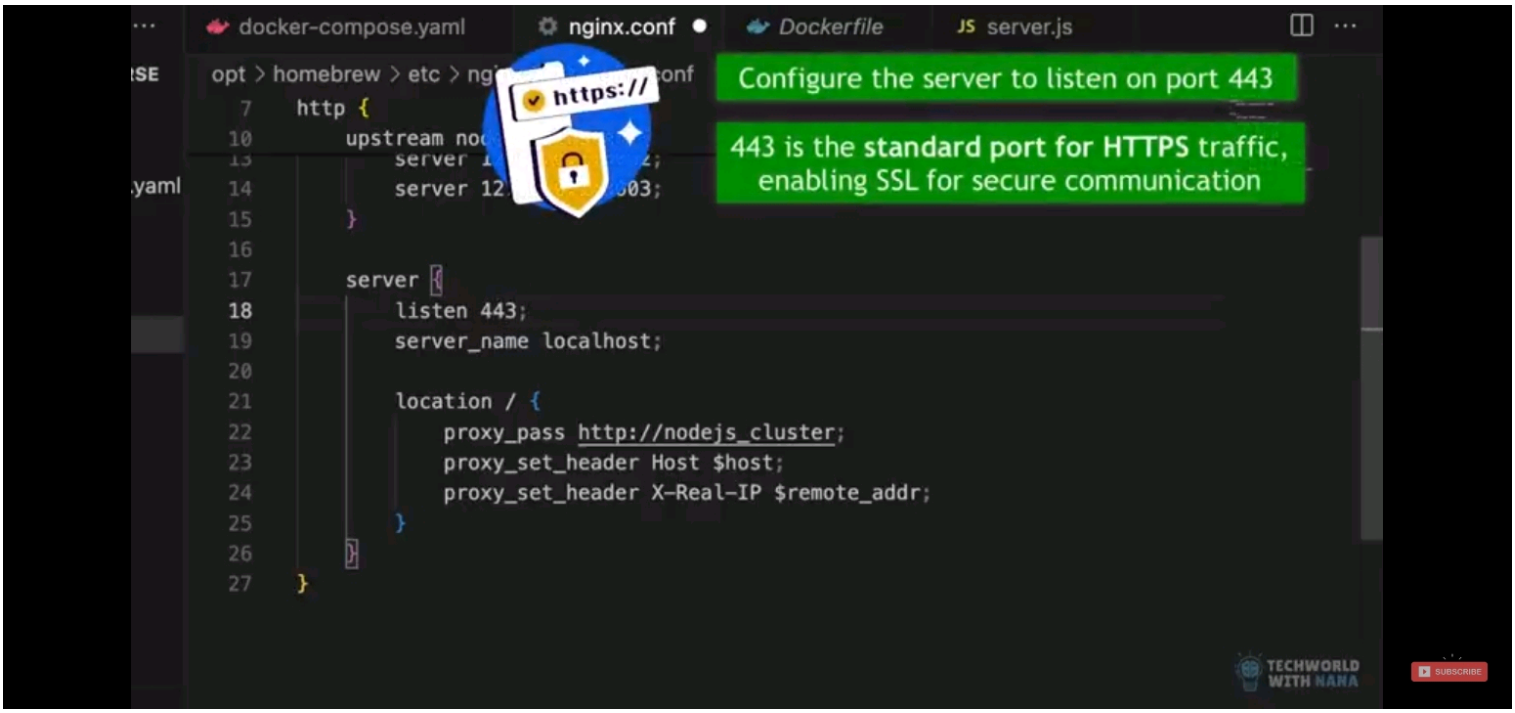

Terminal Shell Edit View Window Help
Thu 10. Oct 12:56
nginx-certs — -zsh — 100×26
nana@macbook /Users/nana
% mkdir nginx-certs
nana@macbook /Users/nana
% cd nginx-certs
nana@macbook /Users/nana/nginx-certs
% openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout nginx-selfsigned.key -out nginx-selfsigned.crt
Breakdown of Command
-keyout nginx-selfsigned.key
Specifies the output file for the generated private key, in this case "nginx-selfsigned.key"
-out nginx-selfsigned.crt
Specifies the output file for the self-signed certificate, in this case "nginx-selfsigned.crt"
Contains the public key
TECHWORLD WITH NANA
SUBSCRIBE

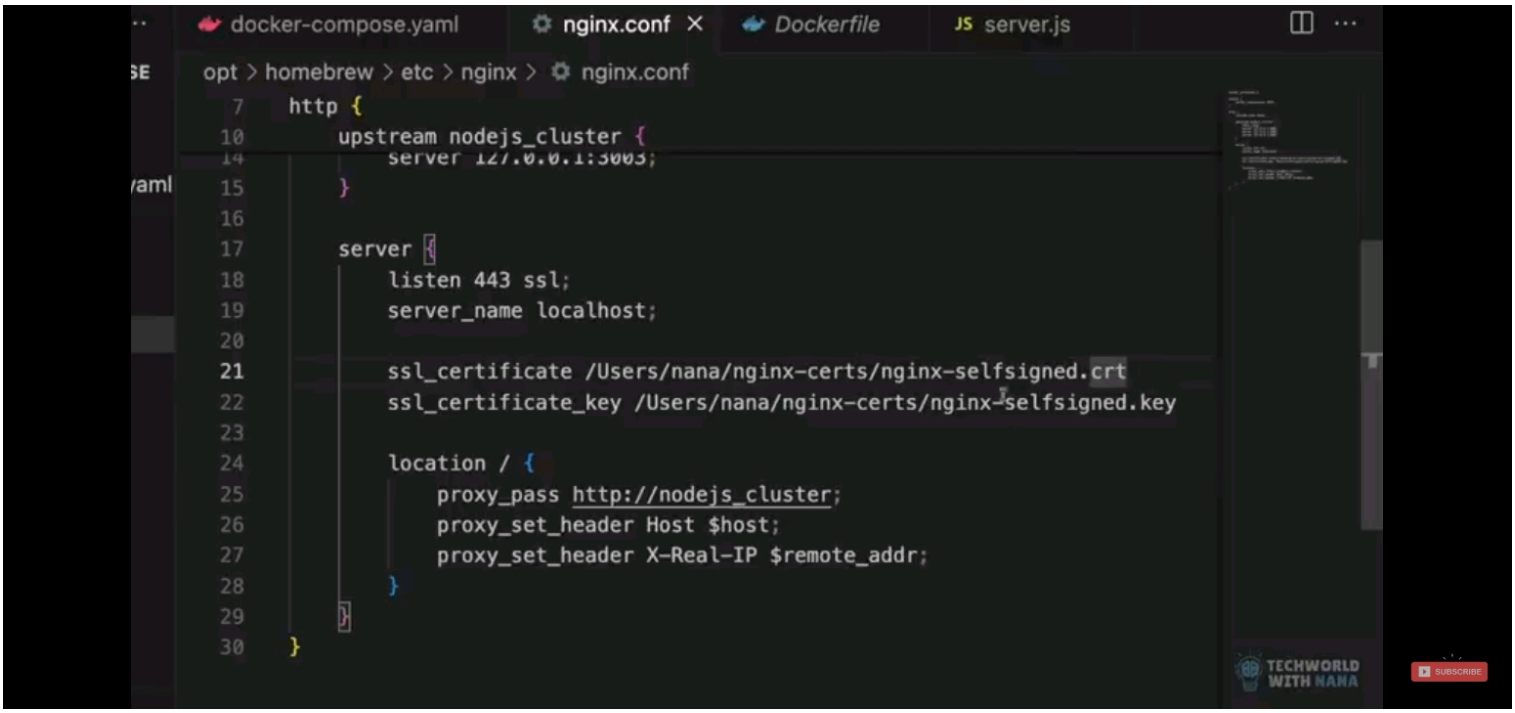
```
nana@macbook /Users/nana
% mkdir nginx-certs
nana@macbook /Users/nana
% cd nginx-certs
nana@macbook /Users/nana/nginx-certs
% openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout nginx-selfsigned.key -out nginx-selfsigned.crt
```

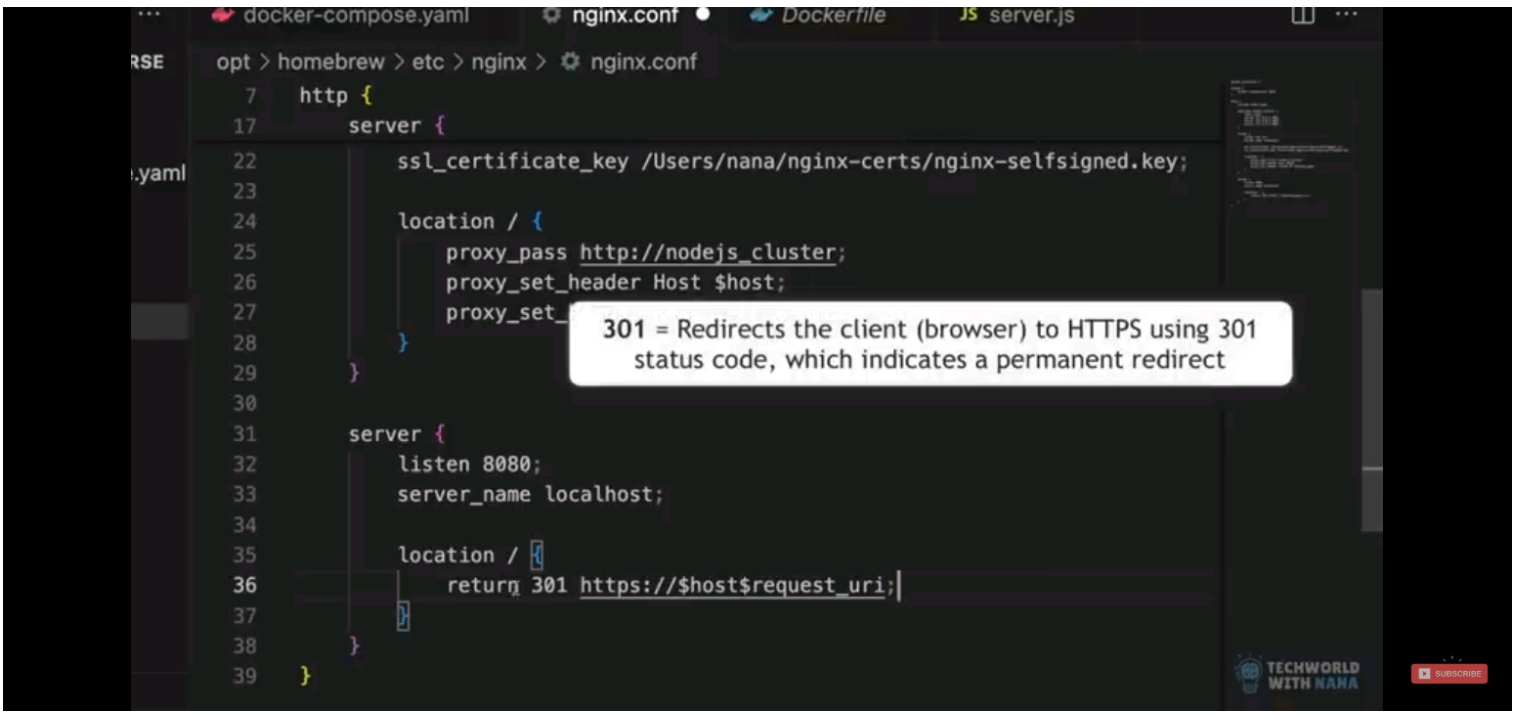
- Private key must be kept secret. It's never shared publicly
- Used to decrypt data

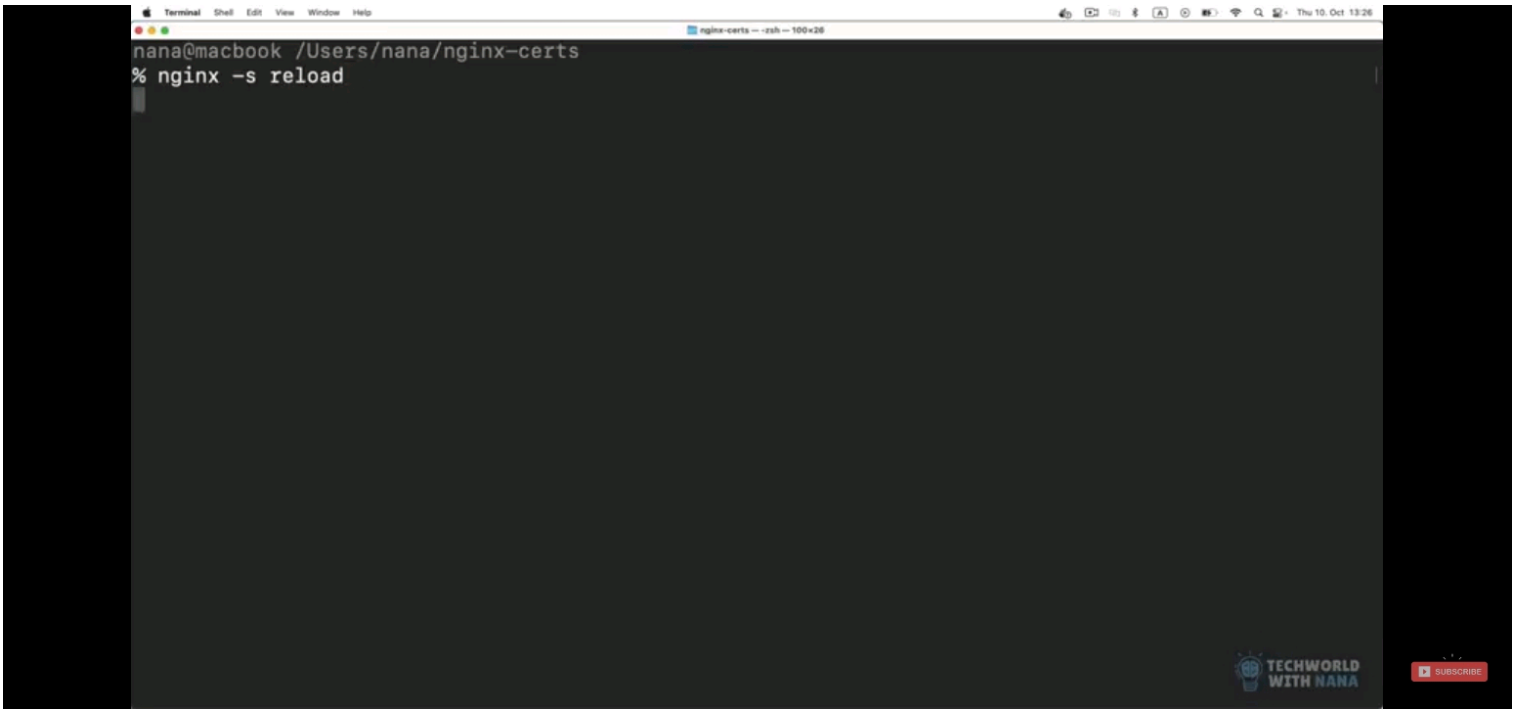
- Public key is shared with clients to encrypt data

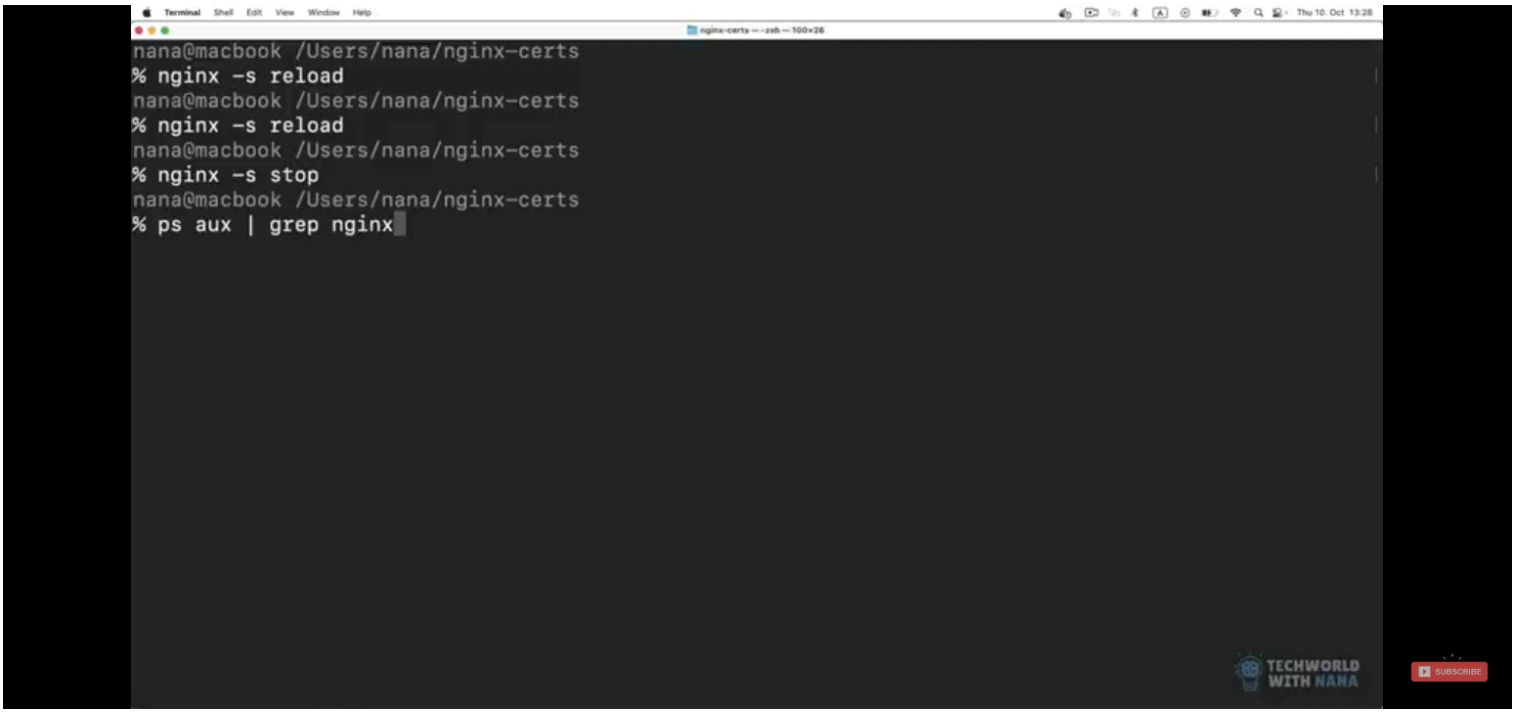

Terminal Shell Edit View Window Help
nginx-certs — -zsh — 100×26
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
Country Name (2 letter code) [AU]:AT
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:Vienna
Organization Name (eg, company) [Internet Widgits Pty Ltd]:TechWorld with Nana
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
nana@macbook /Users/nana/nginx-certs
% ls
nginx-selfsigned.crt nginx-selfsigned.key
nana@macbook /Users/nana/nginx-certs
%
TECHWORLD WITH NANA
SUBSCRIBE


docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > ng
conf
https://
Configure the server to listen on port 443
443 is the standard port for HTTPS traffic, enabling SSL for secure communication
http {
upstream no
server 1
server 12
003;
}
server {
listen 443;
server_name localhost;
location / {
proxy_pass http://nodejs_cluster;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
}
}
}
TECHWORLD WITH NANA
SUBSCRIBE


docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > nginx > nginx.conf
http {
upstream nodejs_cluster {
server 127.0.0.1:3003;
}
server {
listen 443 ssl;
server_name localhost;
ssl_certificate /Users/nana/nginx-certs/nginx-selfsigned.crt
ssl_certificate_key /Users/nana/nginx-certs/nginx-selfsigned.key
location / {
proxy_pass http://nodejs_cluster;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
}
}
}
TECHWORLD WITH NANA
SUBSCRIBE


docker-compose.yaml
nginx.conf
Dockerfile
server.js
opt > homebrew > etc > nginx > nginx.conf
http {
server {
ssl_certificate_key /Users/nana/nginx-certs/nginx-selfsigned.key;
location / {
proxy_pass http://nodejs_cluster;
proxy_set_header Host $host;
}
}
server {
listen 8080;
server_name localhost;
location / {
return 301 https://$host$request_uri;
}
}
}
301 = Redirects the client (browser) to HTTPS using 301 status code, which indicates a permanent redirect
TECHWORLD WITH NANA
SUBSCRIBE


Terminal Shell Edit View Window Help
Thu 10. Oct 13:26
nginx-certs — -zsh — 100×26
nana@macbook /Users/nana/nginx-certs
% nginx -s reload
TECHWORLD WITH NANA
SUBSCRIBE


Terminal Shell Edit View Window Help
Thu 10. Oct 13:28
nginx-certs — -zsh — 100×26
nana@macbook /Users/nana/nginx-certs
% nginx -s reload
nana@macbook /Users/nana/nginx-certs
% nginx -s reload
nana@macbook /Users/nana/nginx-certs
% nginx -s stop
nana@macbook /Users/nana/nginx-certs
% ps aux | grep nginx
TECHWORLD WITH NANA
SUBSCRIBE