DevOps

Write it, Deploy it, Manage it

Pascal Sun

UWA NLP-TLP Group

The University of Western Australia

December 6, 2024

Table of Contents

Introduction

Scope

- Frontend: Next.js deployment with automated CI/CD
- Backend: Platform deployment using Docker Compose
- ML Models: Containerized Python package deployment

What we need?

- A domain for Platform and Frontend
- Vercel[1] for React/Nextjs
- A Ubuntu 22.04 server for Platform
  - Nectar provides free instance [2]
  - AWS also has free layer
- Nginx and Https for server

[1] https://vercel.com/
[2] https://dashboard.rc.nectar.org.au/

Preparation

What is Domain?

- Purchase domain from registrar
  - AWS
  - NameCheap
  - cloudflare
- DNS Record
  - CNAME
  - A Record

What is Vercel?

Frontend hosting includes two steps: build and host generated static files.

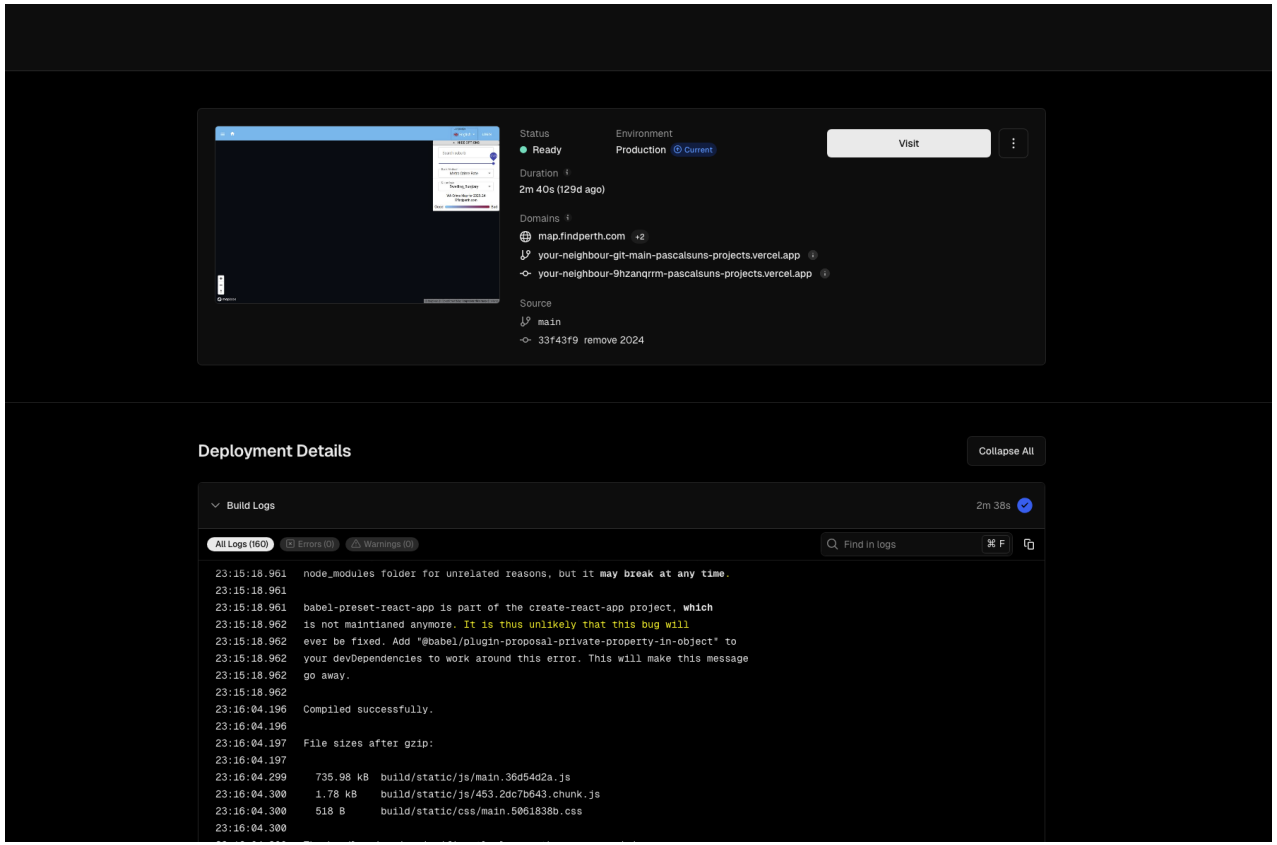

Figure: Vercel is free for Hobby usage[3]

[3] https://vercel.com/pascalsuns-projects/your-neighbour/DtP8RAmpC92KBw2qVDY27SU2P4KV

What is Nginx?

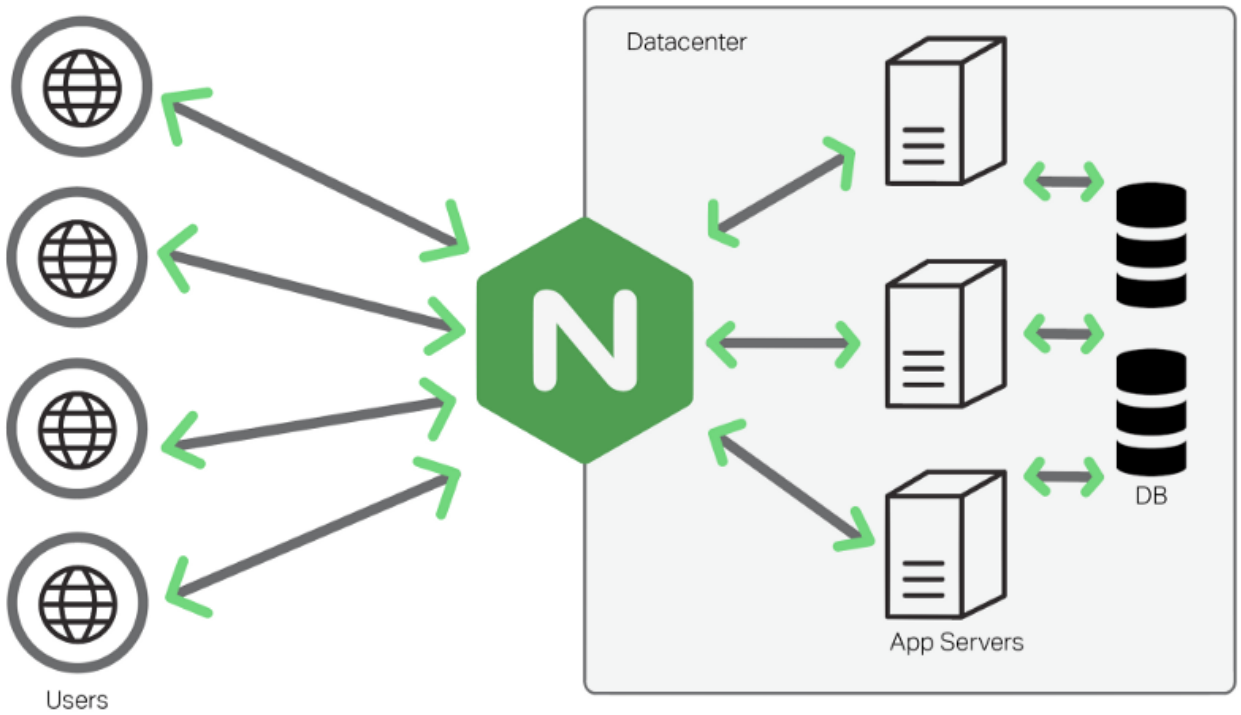

Figure: Nginx Usage

What is https and certbot?

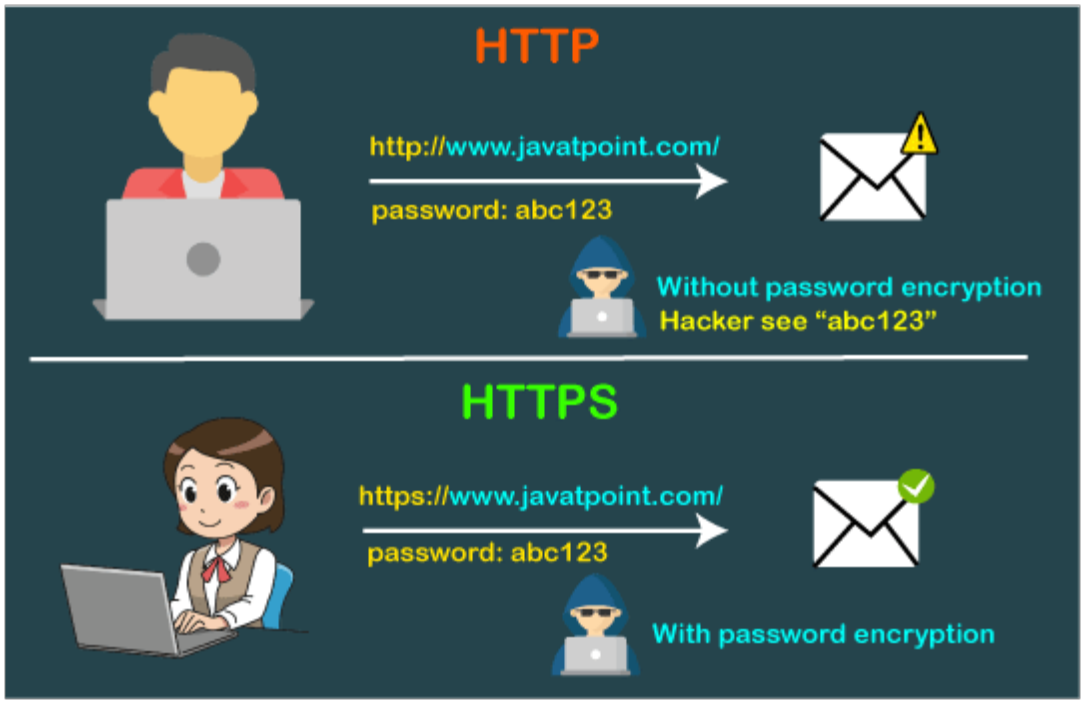

Figure: HTTPS vs HTTP

Frontend

How Deploy Works for Next.js

Build Phase

- `next build` generates optimized production build:

```
$ next build
   Next.js 14.0.1
 - Creating production build...
 - Generating static pages (8/8)
 - Collecting page data...
  Compiled successfully

Route (pages)  Size  First Load
   /           4 kB     84.1 kB
   /about      2 kB     82.1 kB
   /blog       3 kB     83.1 kB
```

How Deploy Works for Next.js

Host Phase

Output:

- Creates `.next/` directory containing:
  - `static/`: CSS, JavaScript, images
  - `server/`: Server-side code
  - `cache/`: Build cache

Steps to deploy frontend

- Connect your GitHub Repo to Vercel
- Select the Nextjs as framework
- Click Deploy
- Add CNAME record for your Domain

Platform

Steps

- An Ubuntu22.04 sever or any sever you prefer
- Install docker on the server[4]
- Install Nginx on the server
- Setup Nginx
- Put the code in the sever and docker compose up
- Change DNS A record

[4] refer back to docker install on Linux in Docker session

Nginx Installation

Installation script:

```
# Update package list
sudo apt update

# Install Nginx
sudo apt install nginx

# Start Nginx service
sudo systemctl start nginx

# Enable on boot
sudo systemctl enable nginx

# Check status
sudo systemctl status nginx

```

Nginx Configuration

Basic Nginx configuration for reverse proxy:

```
# /etc/nginx/sites-available/yourapp.conf
server {
    listen 80;
    server_name yourapp.com www.yourapp.com;
    # Frontend proxy
    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
    # proxy: for django, hasura, etc
    location /api {
        proxy_pass http://localhost:8000;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}

```

CI/CD via GitHub Actions

```yaml
name: Deployment
on:
  push:
    branches: [prod, develop]
jobs:
  platform:
    name: Deploy Platform
    runs-on: ubuntu-latest
    steps:
      - name: Package codebase
        uses: actions/checkout@v1
      - name: Configure and zip
        run: |
          tar -cvf platform.tar .
      - name: Upload to server
        uses: appleboy/scp-action@master
        with:
          host: ${{ secrets.SERVER_HOST }}
          username: ${{ secrets.SERVER_USER }}
          password: ${{ secrets.SERVER_PASSWORD }}
          port: 22
          source: "platform.tar"
          target: "/home/user/app"
      - name: Deploy
        uses: appleboy/ssh-action@master
        with:
          host: ${{ secrets.SERVER_HOST }}
          username: ${{ secrets.SERVER_USER }}
          password: ${{ secrets.SERVER_PASSWORD }}
          port: 22
          script: |
            cd /home/user/app
            tar -xvf platform.tar
            rm platform.tar
            docker compose up -d

```

GitHub Secrets Setup

- Required secrets:
  - SERVER_HOST: Production server IP
  - SERVER_USER: SSH username
  - SERVER_PASSWORD: SSH password
- Add secrets:
  - GitHub repository settings
  - Secrets and variables
  - Actions → New repository secret

Python Package

How to deploy Python Package?

- A lot of ways: container or serverless
- However, we get it containerized and running locally

Demo

Q&A

Questions?

Feel free to ask anything