KERJA PRAKTIK – IF184801

IMPLEMENTASI DAN KONFIGURASI INFRASTRUKTUR SISTEM CERDAS 3DVT

Departemen Teknik Komputer Insitut Teknologi Sepuluh Nopember

Keputih, Kec. Sukolilo, Kota Surabaya, Jawa Timur 60111

Periode: 01 September 2022 – 31 Oktober 2022

Oleh:

Ega Prabu Pamungkas 05111940000014

Pembimbing Departemen

Dr. Ahmad Saikhu, S.Si., MT.

Pembimbing Lapangan

Arta Kusuma Hernanda S. T., M. T.

DEPARTEMEN TEKNIK INFORMATIKA

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya 2023

KERJA PRAKTIK – IF184801

IMPLEMENTASI DAN KONFIGURASI INFRASTRUKTUR SISTEM CERDAS 3DVT

Departemen Teknik Komputer Insitut Teknologi Sepuluh Nopember

Keputih, Kec. Sukolilo, Kota Surabaya, Jawa Timur 60111

Periode: 01 September 2022 – 31 Oktober 2022

Oleh:

Ega Prabu Pamungkas 05111940000014

Pembimbing Departemen

Dr. Ahmad Saikhu, S.Si., M.T.

Pembimbing Lapangan

Arta Kusuma Hernanda S. T., M. T.

DEPARTEMEN TEKNIK INFORMATIKA

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya 2023

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

LEMBAR PENGESAHAN
KERJA PRAKTIK

Implementasi dan Konfigurasi Infrastruktur Sistem Cerdas 3DVT

Oleh:

Ega Prabu Pamungkas 05111940000014

Disetujui oleh Pembimbing Kerja Praktik:

1. Dr. Ahmad Saikhu, S.Si., M.T.
   NIP 197107182006041001

(Pembimbing Departemen)

2. Arta Kusuma Hernanda, S.T., M.T.

(Pembimbing Lapangan)

[Halaman ini sengaja dikosongkan]

Perancangan dan Implementasi Sistem Cerdas Aplikasi 3DVT

Nama Mahasiswa	: Ega Prabu Pamungkas
NRP	: 05111940000014
Departemen	: Teknik Informatika FTEIC-ITS
Pembimbing Departemen	: Dr. Ahmad Saikhu, S.Si., M.T.
Pembimbing Lapangan	: Arta Kusuma Hernanda S.T., M.T.

ABSTRAK

Sistem cerdas 3DVT merupakan sebuah sistem cerdas untuk melakukan segmentasi gumpalan darah (thrombus) pada kasus Deep Vein Thrombosis (DVT) melalui citra 2D USG. Selain dapat melakukan segmentasi secara otomatis area thrombus dan pembuluh darah, 3DVT juga dilengkapi dengan fitur rekonstruksi 3D pembuluh darah serta thrombus dari hasil pengolahan citra 2D USG. Sistem 3DVT tersedia di 3 platform yaitu Windows, Website, dan Android Mobile. Sistem ini memerlukan infrastruktur yang dapat mendukung kerjanya machine learning dari 3DVT. Infrastruktur dibuat dengan menggunakan bantuan Docker. Server 3DVT dibangun dengan VPS dari Digital Ocean (Droplets) dan menggunakan penyimpanannya berupa Digital Ocean Spaces. Sistem ini dibangun menggunakan webserver Nginx dan untuk membantu proses pembangunan (*development*) dibangun CI/CD menggunakan GitHub Actions. Dengan teknologi tersebut, sistem 3DVT dapat dibangun dengan baik saat proses maupun pasca proses pembangunan sistem.

***Kata Kunci**: Website, Nginx, Docker, DigitalOcean, Spaces, GitHub Actions*

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur saya sampaikan kepada Allah SWT karena berkat rahmat-Nya kami dapat melaksanakan salah satu kewajiban saya sebagai mahasiswa Departemen Informatika, yakni Kerja Praktik (KP).

Saya menyadari masih ada kekurangan baik dalam pengerjaan kerja praktik maupun penyusunan buku laporan ini. Namun, saya berharap dengan disusunnya buku laporan ini dapat menambah pengetahuan dan wawasan pembaca serta menjadi sumber referensi. Saya juga terbuka dengan kritik dan saran yang dapat membantu menyempurnakan buku laporan kerja praktik ini.

Melalui buku laporan ini, saya juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu, baik langsung maupun tidak langsung, dalam pengerjaan kerja praktik hingga penyusunan buku laporan ini. Orang-orang tersebut antara lain adalah:

1. Kedua orang tua penulis.
2. Bapak Ary Mazharuddin S.Kom., M.Comp.Sc. selaku koordinator kerja praktik.
3. Bapak Dr. Ahmad Saikhu, S.Si., M.T. selaku dosen pembimbing departemen.
4. Mas Arta Kusuma Hernanda, S.T., M.T. selaku pembina lapangan kerja praktik.

Surabaya, 05 Agustus 2023

Ega Prabu Pamungkas

NRP 05111940000014

[Halaman ini sengaja dikosongkan]

BAB I PENDAHULUAN

1.1. Latar Belakang

Sistem cerdas 3DVT merupakan sebuah sistem cerdas untuk melakukan segmentasi gumpalan darah (thrombus) pada kasus Deep Vein Thrombosis (DVT) melalui citra 2D USG. Selain dapat melakukan segmentasi secara otomatis area thrombus dan pembuluh darah, 3DVT juga dilengkapi dengan fitur rekonstruksi 3D pembuluh darah serta thrombus dari hasil pengolahan citra 2D USG. Sistem ini untuk mengembangkan sistem ultrasound tiga dimensi untuk melakukan penentuan volume gumpalan darah pada pembuluh darah sebelum aspirasi (tindakan penyedotan) ataupun setelah aspirasi. Dengan demikian akan bisa ditentukan berapa volume gumpalan darah yang akan diaspirasi dan nantinya dicocokkan dengan berapa volume gumpalan darah yang sudah diaspirasi.

Sistem 3DVT tersedia di 3 platform yaitu Windows, Website, dan Android Mobile. Sistem ini terdapat Machine Learning yang digunakan untuk merokonstruksi 3D pembuluh darah serta thrombus dari hasil pengolahan citra 2D USG. Oleh karena itu,, diperlukan infrastruktur sistem yang dapat mendukung kerjanya machine learning dari 3DVT. Saya diberikan kesempatan untuk merancang dan meimplementasikan infrastruktur sistem 3DVT dan sekaligus

membantu otomasi dari deployment backend dan frontend 3DVT.

1.2. Tujuan

Tujuan kerja praktik ini adalah menyelesaikan kewajiban kerja praktik sebesar 2 SKS. Selain itu, membantu mengkonfigurasi dan men-deploy sistem 3DVT serta mealakukan otomasi terhadap backend dan frontend 3DVT agar perkembangan sistemnya berjalan dengan lancar.

1.3. Manfaat

Manfaat kerja praktik ini adalah memperlancar jalannya sistem 3DVT saat masa pengembangan maupun pasca pengembangan serta sebagai wadah untuk mengembangkan skill pengembang sistem.

1.4. Rumusan Masalah

Rumusan masalah pada kerja praktik ini adalah sebagai berikut.

1. Bagaimana implementasi infrastruktur server yang dibutuhkan sistem 3DVT?
2. Bagaimana otomasi pengembangan backend dan frontend sistem 3DVT?

1.5. Lokasi dan Waktu Kerja Praktik

Pengerjaan kerja praktik ini dikerjakan secara *remote* dengan meeting tatap muka tim dengan pengembang lainnya untuk koordinasi satu kali dan koordinasi lainnya melalui *online.*

Adapun pengerjaan kerja praktik ini dilakukan selama 2 bulan dimana dimulai pada tanggal 1 September sampai 31 Oktober 2022.

1.6. Metodologi Kerja Praktik

1.6.1. Perumusan Masalah

Dalam rangka mengetahui kebutuhan sistem 3DVT, saya mengikuti rapat tim pengembang sistem yang terdiri dari 5 orang termasuk ketua tim. Pada meeting ini dijelaskan apa itu 3DVT dan bagaimana 3DVT bekerja. Setelah itu, ketua tim pengembang menjelaskan fitur-fitur yang diterapkan dan terakhir bagaimana kebutuhan infrastruktur dari sistem 3DVT.

1.6.2. Studi Literatur

Setelah mendapat gambaran bagaimana sistem berjalan dan kebutuhan apa saja yang dibutuhkan, ketua tim juga memberitahu tinjauan apa saja yang akan diimplementasikan. Tinjauan yang dipakai meliputi Digital Ocean sebagai cloud

hosting, Nginx sebagai webserver, serta PostgreSQL sebagai DBMS.

1.6.3. Analisis dan Perancangan

Berdasarkan tinjauan yang akan dipakai, untuk merancang sistem yang dapat menunjang lebih lanjut maka akan digunakan Docker untuk pengembangan sistem berbasis container dan Github Actions untuk CI/CD pada proses pengembangan.

1.6.4. Implementasi Sistem

Realisasi sistem akan dilakukan dengan melakukan deployment server menggunakan Digital Ocean sebagai cloud hosting serta mengimplementasikan CI/CD menggunakan Github Actions.

1.6.5. Pengujian dan Evaluasi

Setelah sistem telah terkonfigurasi dan terdeploy, akan dilihat peforma sistem baik dan CI/CD atau otomasi backend dan frontend sistem berjalan dengan baik.

1.6.6. Kesimpulan dan Saran

Setelah sistem selesai diuji dan dievaluasi akan dibuat kesimpulan dan saran berdasarkan performa infrastruktur sistem.

1.7. Sistematika Laporan

Buku laporan ini terdiri dari enam bab dengan rincian sebagai berikut.

1.7.1. Bab I Pendahuluan

Pada bab ini dijelaskan latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

1.7.2. Bab II Profil Perusahaan

Pada bab ini dijelaskan gambaran umum Departemen Teknik Komputer Institut Teknologi Sepuluh Nopember mulai dari profil dan lokasi.

1.7.3. Bab III Tinjauan Pustaka

Bab ini berisi landasan teori dari teknologi yang akan digunakan dalam pengerjaan kerja praktik.

1.7.4. Bab IV Implementasi dan Konfigurasi Sistem

Bab ini merupakan implementasi sistem dimana diatur sedemikian rupa sehingga dapat memenuhi kebutuhan sistem 3DVT.

1.7.5. Bab VI Pengujian dan Evaluasi

Bab ini memaparkan hasil pengujian dan evaluasi sistem berdasarkan konfigurasi dan implementasi yang telah dilakukan.

1.7.6. Bab VI Kesimpulan dan Saran

Bab ini menjelaskan kesimpulan yang didapat dari hasil pengujian dan evaluasi yang telah dilakukan serta saran untuk kedepannya.

[Halaman ini sengaja dikosongkan]

BAB II PROFIL PERUSAHAAN

2.1. Profil Departemen Teknik Komputer ITS

Departemen Teknik Komputer ITS merupakan salah satu departemen di Institut Teknologi Sepuluh Nopember. Teknik Komputer ITS mengajarkan disiplin ilmu yang memadukan software dari bidang ilmu komputer dengan hardware dari bidang komputer, elektronika, dan telekomunikasi untuk diaplikasikan pada sistem komputasi modern (Cloud Computing, Wireless Sensor Network, IoT/ Internet of Things, Wearable Device, dan Embedded System) maupun pada peralatan yang dikontrol oleh komputer, dan jaringan dari perangkat cerdas, serta Robotika. Teknik Komputer juga mempelajari pengolahan data skala besar (big data), keamanan data dan jaringan (network security), dan komputasi multimedia.

2.2. Lokasi

Fakultas Teknologi Elektro dan Informatika Cerdas, Gedung B dan C, Keputih, Sukolilo, Surabaya, Jawa Timur 60117

[Halaman ini sengaja dikosongkan]

BAB 3

TINJAUAN PUSTAKA

3.1. Digital Ocean

Digital Ocean (DO) adalah penyedia infrastruktur cloud terkemuka yang menawarkan pengembang platform yang mudah digunakan, fleksibel, dan dapat diskalakan untuk menerapkan, mengelola, dan menskalakan aplikasi. Didirikan pada tahun 2011 oleh Ben Uretsky, Moisey Uretsky, Mitch Wainer, Jeff Carr, dan Alec Hartman, DigitalOcean berfokus pada penyederhanaan kompleksitas infrastruktur web dan menawarkan pengalaman pengguna yang intuitif. Rangkaian produk intinya meliputi server virtual (Droplets), Kubernetes terkelola (DigitalOcean Kubernetes), penyimpanan objek (Spaces), dan database terkelola (Database Terkelola DigitalOcean), di antara layanan lainnya [2].

Droplets digunakan untuk menerapkan dan menskalakan aplikasi web, API, dan layanan mikro, memanfaatkan jaringan pusat data globalnya untuk memastikan latensi rendah dan ketersediaan tinggi. Droplets dapat diskalakan mulai dari CPU, RAM, hingga penyimpanannya sehingga dapat memudahkan pengembang dalam memilih infrastruktur yang tepat untuk projek mereka.

Spaces dapat digunakan pengembang untuk menyimpan dan melayani aset statis, seperti gambar, video, dan dokumen saat menggunakan layanan Database terkelola. Dalam menggunakan

Spaces pengembang dapat menghubungkan aplikasi mereka menggunakan API Digital Ocean.

Database terkelola Digital Ocean menyediakan berbagai macam DBMS, salah satunya PostgreSQL. Databse terkelola DO dapat diskalakan CPU, RAM, maupun penyimpanannya.

3.2. Docker

Docker adalah layanan yang menyediakan kemampuan untuk mengemas dan menjalankan sebuah aplikasi dalam sebuah lingkungan terisolasi yang disebut dengan container. Dengan adanya isolasi dan keamanan yang memadai memungkinkan untuk menjalankan banyak container di waktu yang bersamaan pada host tertentu [1]. Docker dapat dijalankan pada beberapa platform cloud, salah satunya Droplets Digital Ocean. Docker menggunakan image yang tersedia pada docker hub atau menggunakan image yang telah dibangun sendiri. Konfigurasi dari container itu sendiri dapat dibuat dengan membuat sebuah file bernama Dockerfile.

Salah satu fitur Docker adalah Docker-Compose. Docker-Compose adalah alat untuk mendefinisikan dan menjalankan satu atau beberapa container yang saling terkait dengan sebuah command. Pada implementasinya, dapat menggunakan dengan membuat sebuah file berekstensi yaml/yml yang di dalamnya terdapat konfigurasi terhadap service aplikasi yang akan dijalankan. Sederhananya seperti menyatukan semua

Dockerfile dari setiap service aplikasi ke daam sebuah file yaml/yml (docker-compose file). Sehingga, dapat menyederhanakan dan mempercepat kerja untuk meng-create maupun memulai container-container.

3.3. Github Actions

Github memiliki salah satu fitur untuk melakukan otomasi pengembangan, yaitu GitHub Actions. GitHub Actions adalah platform *continuous integration* and *continuous delivery* (CI/CD) yang memungkinkan mengotomatiskan build, test, dan deployment pipeline. Pengembang dapat membuat alur kerja (*workflows*) yang membangun dan menguji setiap permintaan tarikan ke repositori pengembang, atau menerapkan permintaan tarikan gabungan ke produksi.

Pengembang dapat mengonfigurasi alur kerja GitHub Actions untuk dipicu (*trigger*) ketika suatu *event* terjadi di repositori, seperti *pull request* atau *push* pada repositori. Alur dapat berisi satu atau beberapa pekerjaan (*job*) yang dapat dijalankan secara berurutan ataupun paralel. Setiap pekerjaan akan berjalan di dalam mesin virtualnya sendiri (*runner*), atau di dalam *container*, dan memiliki satu atau beberapa langkah yang menjalankan skrip yang telah ditetapkan atau menjalankan tindakan (*actions*), yang merupakan ekstensi yang dapat digunakan kembali yang dapat menyederhanakan alur kerja. Alur

kerja ditulis pada file berekstensi yaml/yml dan harus diletakkan pada *.github/workflows* [4].

3.4. Web Server (Nginx)

Web server berfungsi sebagai penerima request dari browser yang kemudian memberikan tanggap dengan mengirimkan halaman situs web dalam bentuk dokumen HTML Salah satu webserver yang cukup populer adalah Nginx.. Nginx memberikan performa yang andal dan mempunyai beberapa fitur canggih lain yang mudah dikonfigurasi. Nginx memiliki banyak kelebihan dalam hal fitur, di antaranya *URL rewriting, virtual host, file serving, reverse proxying, access control,* dan masih banyak lagi.

Salah saatu keuntungan Nginx adalah penggunaan memori yang kecil dengan konkurensi yang tinggi. Jadi, Nginx tidak membuat proses baru ketika ada permintaan (*web request*), tapi ditangani di dalam satu thread (*asynchronous* dan pendekatan *event-driven*).

Jika menggunakan Nginx, satu proses utama (*master process*) dapat mengontrol berbagai proses lainnya (*worker process*). Jadi proses utama tetap menjalankan tugasnya sambil mengontrol proses lain yang ada di bawahnya. Karena Nginx menggunakan *asynchronous,* setiap *web request* dapat dieksekusi oleh proses lain tanpa mengganggu *web request* lainnya. [5]

3.5. PostgreSQL

PostgreSQL adalah sistem database objek-relasional sumber terbuka yang kuat yang menggunakan dan memperluas bahasa SQL yang dikombinasikan dengan banyak fitur yang menyimpan dan menskalakan beban kerja data yang paling rumit dengan aman. PostgreSQL memiliki banyak fitur yang ditujukan untuk membantu pengembang membangun aplikasi, administrator untuk melindungi integritas data dan membangun lingkungan yang toleran terhadap kesalahan, dan membantu

mengelola data tidak peduli seberapa besar atau kecil kumpulan datanya. Selain gratis dan open source, PostgreSQL sangat bisa dikembangkan. Misalnya, pengembang dapat menentukan tipe data sendiri, membangun fungsi kustom, bahkan menulis kode dari berbagai bahasa pemrograman tanpa mengkompilasi ulang database yang telah dibuat.

[Halaman ini sengaja dikosongkan]

BAB IV IMPLEMENTASI DAN KONFIGURASI SISTEM

Sistem 3DVT dibangun pada platform website dan aplikasi android. Backend sistem dibangun menggunakan kerangka kerja Django. Frontend website dibangun dengan React JS sedangkan aplikasi android menggunakan native Kotlin. Infrastruktur pada sistem akan menyesuaikan dengan kerangka kerja yang digunakan agar dapat mendukung kerja dari teknologinya dan sistem dapat berjalan tanpa gangguan maupun konflik pada backend maupun frontend.

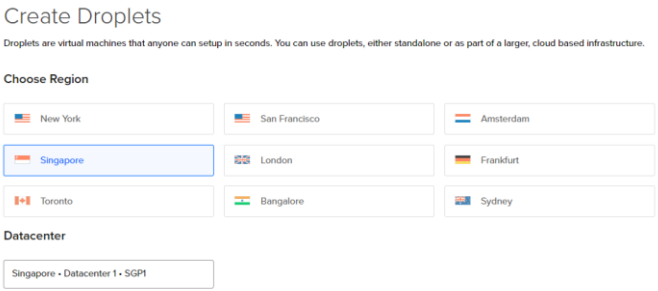
4.1. Konfigurasi Server (Droplets)

Spesifikasi yang dibutuhkan sistem 3DVT dapat dilihat pada tabel berikut.

OS	Storage	Memory	Bandwith
Ubuntu 22 64 bit	25GB SSD	1 GB	1 TB

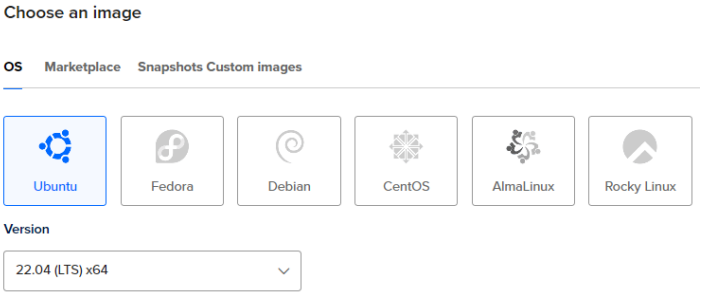
Tabel 4.1 *Spesifikasi Server 3DVT*

Droplets dapat dibuat dengan menggunakan command line ataupun panel pada website Digital Ocean. Untuk kasus ini akan menggunakan paanel pada Digital Ocean. Pertama, login pada website Digital Ocean, klik create dan pilih Droplets. Pada bagian region dipilih Singpura karena region yang terdekat dengan Indonesia agar server dapat berjalan dengan baik.

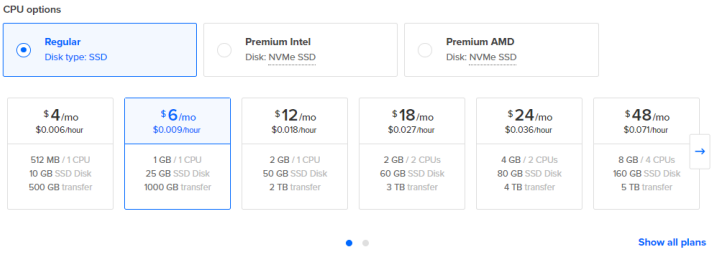


Gambar 4.1 *Buat Droplet Digital Ocean [1]*

Sesuai dengan spesifikasi yang dibutuhkan dipilih OS Ubuntu 22.04(LTS) x64 dan pada size dipilih Basic dan yang sesuai dengan spesifikasi yang dibutuhkan, yaitu 1GB Memory, 25 GB SSD, dan 1TB Bandwith.

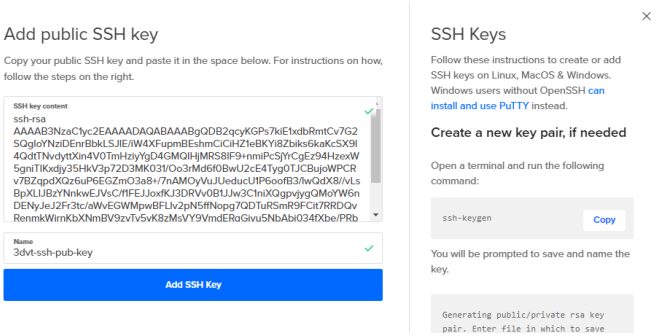


Gambar 4.2 *Buat Droplet Digital Ocean [2]*



Gambar 4.3 *Buat Droplet Digital Ocean [3]*

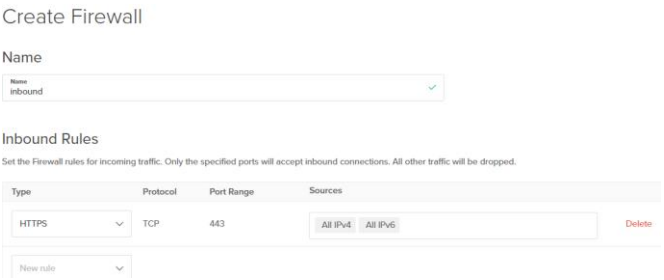
Pada bagian metode Autentikasi dipilih menggunakan SSH dengan menambahkan key baru. SSH Key dapat dibuat dengan menggunakan command "*ssh-keygen*" pada terminal. Setelah mendapatkan private dan public key, dapat memasukkan publik key beserta namanya pada panel Digital Ocean.



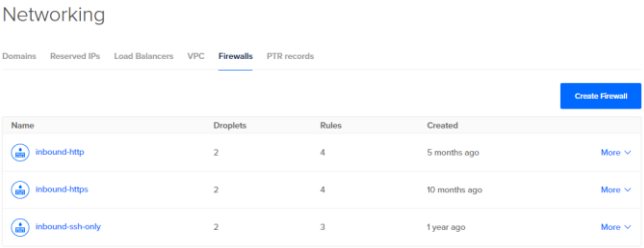
Gambar 4.4 *Menambahkan SSH Key*

Selanjutnya masukkan nama droplet dan pastikan berada di project 3DVT. Selanjutnya perlu mengatur firewall dari Droplets DO. Untuk mengatur firewall dapat menggunakan panel pada DO dengan mengarahkan ke bagian Networking ->

Firewalls -> Create Firewall. Pada bagian ini dapat mengatur inbound rule dengan memilih tipe dan setelah itu atur agar diapply untuk ke droplet 3DVT. Untuk firewall ini dapat dibuat terpisah untuk inbound port 80 (HTTP), 443 (HTTPS), dan 22 (SSH).



***Gambar 4.5** Menambahkan Firewall*



***Gambar 4.6** Tampilan List Firewall*

Server sudah dapat dipakai dan siap untuk dideploy backend dan frontend sistem 3DVT.

4.2. Konfigurasi API Spaces

Pada backend sistem 3DVT (menggunakan Django) diperlukan penyimpanan pada server dengan menggunakan

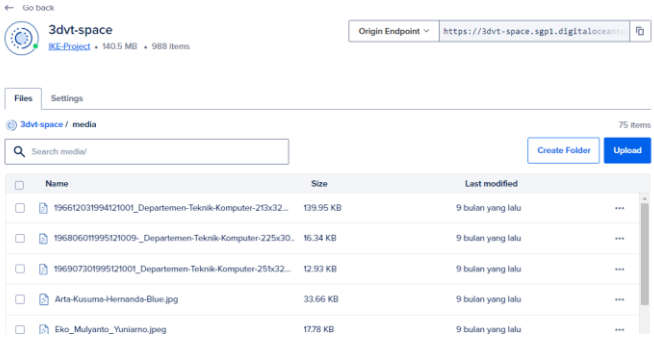
Digital Ocean Spaces. Sebelum dapat menggunakan Spaces, diperlukan akses API DO untuk Spaces. Pada panel DO dapat menggunakan fitur API lalu memilih Spaces. Jika dikllik "Generate New Key", akan diperlukan memasukkan nama dan setelahnya akan disediakan Akses Key dan Secret Key yang akan muncul hanya sekali. Kedua key tersebut dimasukkan ke dalam file *settings.py*. Selanjutnya, perlu dibuat Spaces untuk penyimpanan media dan file. Dengan menggunakan panel DO dapat dibuat Spaces dengan klik Create -> Spaces lalu mengatur region di Singapura dan namanya *3dvt-spaces*.

```python
# Digital Ocean SPACE Configuration
if USE_SPACES == "TRUE":
    AWS_ACCESS_KEY_ID = "[ACCESS KEY]"
    AWS_SECRET_ACCESS_KEY = "[SECRET ACCESS KEY]"
    AWS_STORAGE_BUCKET_NAME = "3dvt-space"
    AWS_DEFAULT_ACL = "public-read"
    AWS_S3_ENDPOINT_URL = "https://sgp1.digitaloceanspaces.com"
    AWS_S3_OBJECT_PARAMETERS = {
        "CacheControl": "max-age=2592000",
    }

    # Static file Settings
    AWS_LOCATION = "static"
    PUBLIC_MEDIA_LOCATION = "media"

    STATICFILES_STORAGE = "backend.storage_backends.StaticStorage"
    STATIC_ROOT = "static/"
    STATIC_URL = f"https://{AWS_S3_ENDPOINT_URL}/{AWS_LOCATION}/"

    # Media file Settings
    DEFAULT_FILE_STORAGE = "backend.storage_backends.PublicMediaStorage"
    MEDIA_ROOT = "media/"
    STATIC_URL = f"https://{AWS_S3_ENDPOINT_URL}/{PUBLIC_MEDIA_LOCATION}/"

else:
    # Static files (CSS, JavaScript, Images)
    # https://docs.djangoproject.com/en/1.9/howto/static-files/
    STATIC_ROOT = os.path.join(BASE_DIR, "api_static")
    STATIC_URL = "/api_static/"

    # Actual directory user files go to
    MEDIA_ROOT = os.path.join(BASE_DIR, "mediafiles")
    # URL used to access the media
    MEDIA_URL = "/media/"
```

Source Code 4.1 *Konfigurasi API Spaces di settings.py*

Source Code di atas diatur untuk staging development dan production dimana jika tidak menggunakan Spaces maka media akan disimpan di file lokal. Ada beberapa variabel yang diperlukan untuk dapat menggunakan API Spaces, seperti Akses Key, Secret Key, dan nama spaces yang digunakan. Setelah mendefiniskannya, perlu diatur untuk static URL setiap media yang diperlukan mengarah ke Endpoint URL agar penyimpanan

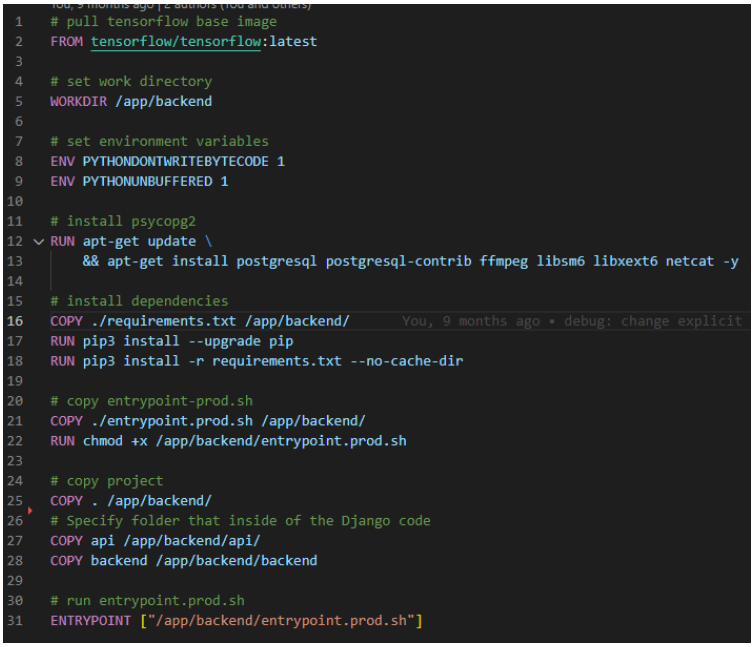
dilakukan pada DO Spaces. Jika berhasil, maka pada DO dapat terlihat medianya di DO Spaces.



***Gambar 4.7** Tampilan Digital Ocean Spaces*

4.3. Konfigurasi Docker

Docker digunakan untuk mengatur konfigurasi dalam membangun dan mendeploy aplikasi sistem 3DVT. Konfigurasi dibuat pada file bernama Dockerfile. Pada bagian backend digunakan image Tensorflow yang tersedia pada *hub.docker.com*. Setelah itu, menginstall kebutuhan untuk PostgreSQL serta kebutuhan lainnya yang digunakan Machine Learning 3DVT dan Backend 3DVT.



```
# pull tensorflow base image
FROM tensorflow/tensorflow:latest

# set work directory
WORKDIR /app/backend

# set environment variables
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

# install psycopg2
RUN apt-get update \
    && apt-get install postgresql postgresql-contrib ffmpeg libsm6 libxext6 netcat -y

# install dependencies
COPY ./requirements.txt /app/backend/      You, 9 months ago • debug: change explicit
RUN pip3 install --upgrade pip
RUN pip3 install -r requirements.txt --no-cache-dir

# copy entrypoint-prod.sh
COPY ./entrypoint.prod.sh /app/backend/
RUN chmod +x /app/backend/entrypoint.prod.sh

# copy project
COPY . /app/backend/
# Specify folder that inside of the Django code
COPY api /app/backend/api/
COPY backend /app/backend/backend

# run entrypoint.prod.sh
ENTRYPOINT ["/app/backend/entrypoint.prod.sh"]
```

Source Code 4.2 *Konfigurasi Docker untuk Django Backend*

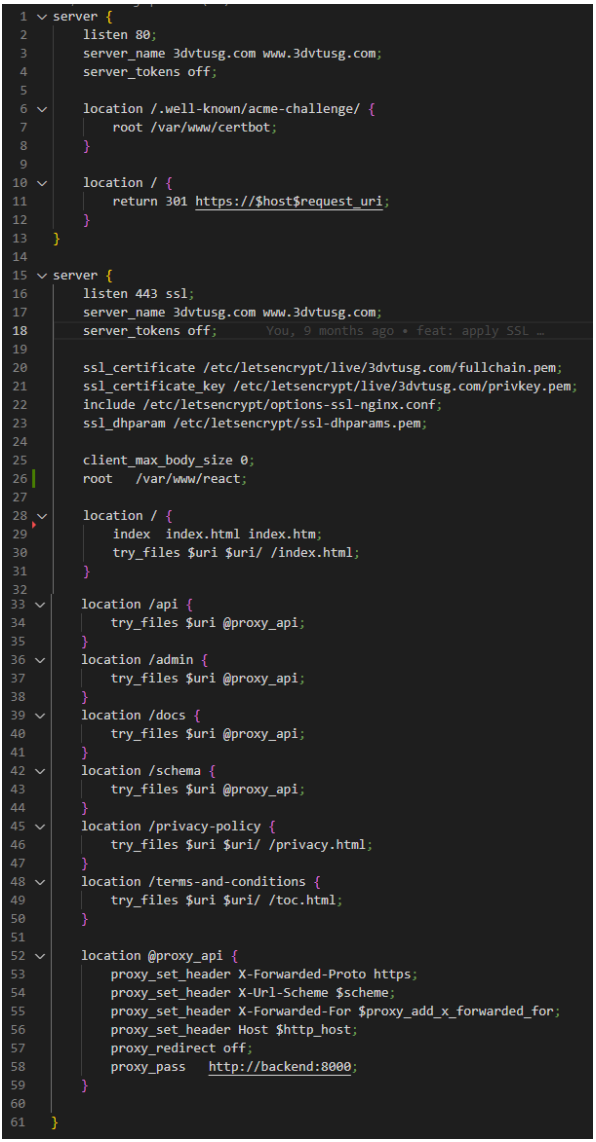
Pada bagian Frontend pembangunan container dibagi menjadi 2 bagian. Bagian pertama untuk membangun aplikasi React menggunakan image Node di *hub.docker.com*. Bagian kedua adalah membangun webserver Nginx menggunakan image Nginx di *hub.docker.com*. Dan terakhir perlu menggantikan file *default.conf* pada container awal dengan konfigurasi Nginx yang sesuai dengan sistem 3DVT dimana konfigurasi akan dibahas pada bagian selanjutnya.

```
# Stage 1
# Build react app using Node image as the builder
FROM node:lts-alpine AS builder
WORKDIR /react
ENV PATH /react/node_modules/.bin:$PATH
COPY package.json ./
COPY package-lock.json ./

RUN npm install -g npm@latest
RUN npm install --omit=dev
COPY ./build ./build
COPY ./src ./src
COPY ./public ./public
RUN npm dedupe
RUN npm prune
RUN npm run build

# Stage 2
# Nginx image for serving the assets
FROM nginx:stable-alpine

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

COPY --from=builder /react/build /var/www/react

# Copy Policy Privacy and TOC pages
COPY additional_pages/privacy.html /var/www/react
COPY additional_pages/toc.html /var/www/react
```

Source Code 4.3 *Konfigurasi Docker untuk Nginx dan React Frontend*

4.4. Konfigurasi Nginx

Perlu dibuat konfigurasi Nginx dan diberi nama default.conf dimana isinya adalah keperluan routing untuk backend dan frontend berkomunikasi, serta routing untuk koneksi HTTPS.



```
server {
    listen 80;
    server_name 3dvtusg.com www.3dvtusg.com;
    server_tokens off;

    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl;
    server_name 3dvtusg.com www.3dvtusg.com;
    server_tokens off;       You, 9 months ago • feat: apply SSL …

    ssl_certificate /etc/letsencrypt/live/3dvtusg.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/3dvtusg.com/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    client_max_body_size 0;
    root   /var/www/react;

    location / {
        index  index.html index.htm;
        try_files $uri $uri/ /index.html;
    }

    location /api {
        try_files $uri @proxy_api;
    }
    location /admin {
        try_files $uri @proxy_api;
    }
    location /docs {
        try_files $uri @proxy_api;
    }
    location /schema {
        try_files $uri @proxy_api;
    }
    location /privacy-policy {
        try_files $uri $uri/ /privacy.html;
    }
    location /terms-and-conditions {
        try_files $uri $uri/ /toc.html;
    }

    location @proxy_api {
        proxy_set_header X-Forwarded-Proto https;
        proxy_set_header X-Url-Scheme $scheme;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_redirect off;
        proxy_pass   http://backend:8000;
    }

}
```

Source Code 4.4 *Konfigurasi Nginx*

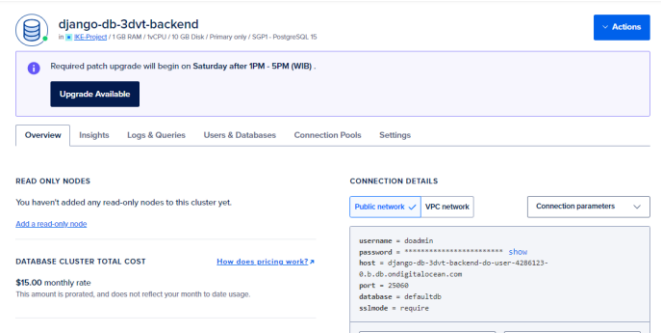
4.5. Konfigurasi Database Terkelola PostgreSQL

Digital Ocean memiliki Database Terkelola yang dapat digunakan sebagai tempat database sistem 3DVT. Spesifikasi database yang diperlukan adalah sebagai berikut.

DBMS	Memory	Storage
PostgreSQL	1 GB	10 GB SSD

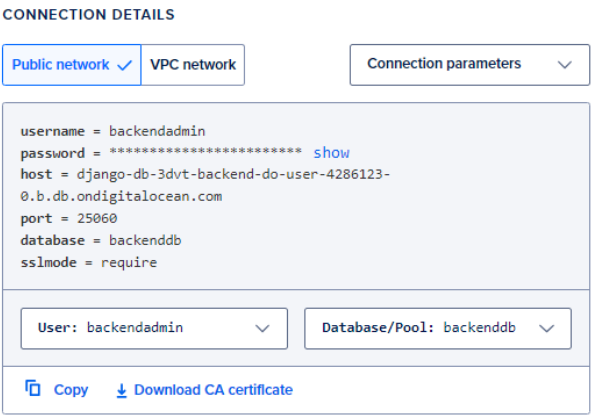
Tabel 4.2 *Spesifikasi Database*

Untuk membuat Database Terkelola dapat klik Create -> Databases. Pilih region terdekat, yaitu Singapura, database engine PostgreSQL, dan kebutuhan spesifikasi yang diperlukan. Terkahir beri nama dan klik "Create Database Cluster". Ini adalah database terkelola dimana dapat dibuat lebih dari satu database.



Gambar 4.8 *Digital Ocean Database Terkelola PostgreSQL*

Untuk membuat database 3DVT, navigasikan ke menu "*Users and Database*" dan buat username baru serta database baru. Setelah itu pada menu "*Overview*" dapat dilihat detail untuk melakukan koneksi pada database.



Gambar 4.9 *Detail Koneksi Database*

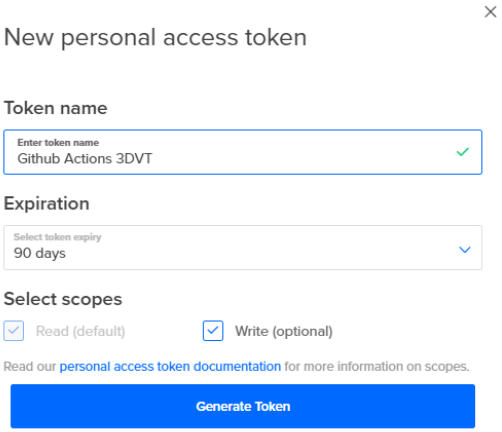
Dengan detail tersebut, masukkan ke dalam *settings.py* pada backend 3DVT. Detail tersebut akan dimasukkan ke dalam variabel-variabel. *SQL_ENGINE* akan berisi "*django.db.backends.postgresql_psycopg2*" yang merupakan engine PostgreSQL untuk Django, *SQL_DATABASE* akan berisi nama database, *SQL_USER* berisi username, *SQL_PASSWORD* berisi password, *SQL_HOST* berisi host, *SQL_PORT* berisi port. Semua variabel tersebut akan dimasukkan pada konfigurasi CI/CD yang akan dibahas pada konfigurasi Github Actions.

```
DATABASES = {
    "default": {
        "ENGINE": os.environ.get("SQL_ENGINE", "django.db.backends.sqlite3"),
        "NAME": os.environ.get("SQL_DATABASE", BASE_DIR / "db.sqlite3"),
        "USER": os.environ.get("SQL_USER", "user"),
        "PASSWORD": os.environ.get("SQL_PASSWORD", "password"),
        "HOST": os.environ.get("SQL_HOST", "localhost"),
        "PORT": os.environ.get("SQL_PORT", "5432"),
    }
}
```

Source Code 4.5 *Konfigurasi Database di settings.py*

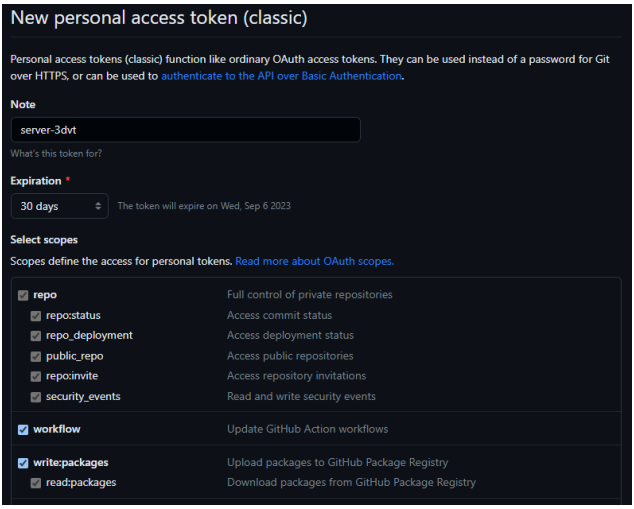
4.6. Konfigurasi Github Actions

Sebelum mengatur pada CI/CD untuk Backend dan Frontend, perlu dilakukan konfigurasi pada repositori untuk memberikan akses CI/CD pada Droplets Digital Ocean. DO memiliki fitur Personal Access untuk melakukan *read* maupun *write* yang dilakukan pada Droplets. Token Personal Access dapat dibuat melalui panel DO dengan klik Genererate New Token lalu memberi nama dan memilih mode *read* dan *write*. Setelah itu akan muncul token hanya sekali dan pastikan untuk menyimpannya. Token ini digunakan Github untuk mengakses Droplet 3DVT.



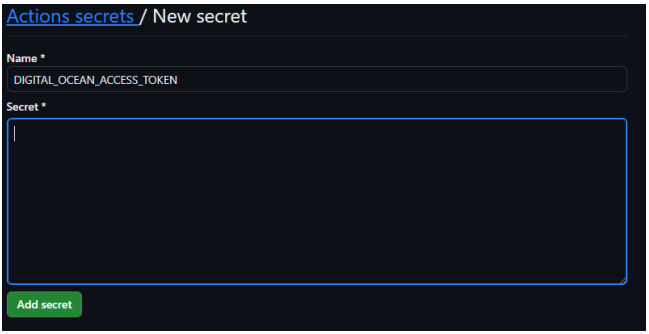
Gambar 4.10 *Buat Personal Acces Token Digital Ocean*

Github perlu akses pada repositori yang akan digunakan sehingga diperlukan token Personal Access pemilik repositori. Untuk mendapatkan token tersebut bisa menuju ke akun Settings -> Developer Settings -> Personal Access Tokens -> Tokens (classic) -> Generate New Token. Selanjutnya beri nama dan centang bagian repo (agar dapat mengakses repositori), workflow (agar dapat menggunakan Github Actions), dan *write:packages* (agar dapat mengupload image ke Gihub Package Registary). Abaikan yang lainnya dan klik "*Generate token*". Setelah itu, akan muncul tokennya yang muncul hanya sekali jadi perlu disimpan terlebih dahulu.



Gambar 4.11 *Buat Personal Access Token GitHub*

Setelah mendapat kedua token tersebut, menavigasikan ke repositori backend 3DVT dan pilih menu Settings -> Secret and Variables -> Actions. Pada bagian ini akan dimasukkan variabel yang penting agar dapat mendeploy aplikasi 3DVT ke server Droplets DO. Variabel penting disimpan disini sebagai keamanan agar yang dapat mengakses hanya pengembang saja. Untuk membuat variabel *secret* dapat klik "*New repository secret"* lalu memasukkan nama variabel dan nilainya.



Gambar 4.12 *Buat Variabel Secret*

Pada repositori backend 3DVT variabel *secret* yang perlu dimasukkan adalah sebagai berikut.

a. DIGITAL_OCEAN_ACCESS_TOKEN, berisi token token Personal Access DO.
b. DIGITAL_OCEAN_IP_ADDRESS, berisi IP Address dari Droplets 3DVT.
c. DJANGO_ALLOWED_HOSTS, berisi nama domain yang akan digunakan
d. NAMESPACE, username pemilik akun repositori.
e. PERSONAL_ACCESS_TOKEN, berisi token Personal Access Github pemilik repositori.
f. PRIVATE_KEY, berisi *private ssh key* untuk Droplet 3DVT.
g. SECRET_KEY, berisi *key* yang di-generate secara random untuk keperluan API Django.
h. SQL_DATABASE, berisi nama database.
i. SQL_HOST, berisi nama host.

j. SQL_PASSWORD, berisi password.
k. SQL_PORT, berisi port.
l. SQL_USER, berisi username.

Sedangkan pada repositori frontend 3DVT, variabel *secret* yang perlu dimasukkan adalah sebagai berikut.

a. DIGITAL_OCEAN_ACCESS_TOKEN, berisi token token Personal Access DO.
b. DIGITAL_OCEAN_IP_ADDRESS, berisi IP Address dari Droplets 3DVT.
c. NAMESPACE, username pemilik akun repositori.
d. PERSONAL_ACCESS_TOKEN, berisi token Personal Access Github pemilik repositori.
e. PRIVATE_KEY, berisi *private ssh key* untuk Droplet 3DVT.

4.7. CI/CD Backend

Dalam CI/CD dipakai salah satu fitur Docker, yaitu docker-compose. Docker-compose digunakan untuk mempermudah pembuatan aplikasi. Dibuat file *docker-compose.ci.yml* yang akan digunakan sebagai awal pembuatan image. Variabel lainnya yang dibutuhkan untuk proses pembuatan image Docker akan dispesifikkan di file *.env*.

```yaml
services:      You, 10 months ago • add docker settings and github
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile.prod
    image: "${BACKEND_IMAGE}"
    command: gunicorn backend.wsgi:application --bind 0.0.0.0:8000
    expose:
      - 8000
    env_file:
      - .env

```

Source Code 4.6 *File docker-compose.ci.yml untuk Backend*

Dibuat juga file *docker-compose.backend.yml* yang akan digunakan sebagai *deployment* ke server 3DVT dimana perbedaannya adalah menggunakan image yang telah dibangun di awal.

```yaml
services:
  backend:
    image: "${BACKEND_IMAGE}"
    command: gunicorn backend.wsgi:application --bind 0.0.0.0:8000
    expose:
      - 8000
    env_file:
      - .env      You, 10 months ago • BREAKING CHANGE: …
```

Source Code 4.7 *File docker-compose.frontend.yml*

Setelah dibuat file docker-compose dan dilakukan konfigurasi terhadap Github Actions, repositori dapat melakukan CI/CD. Setelah itu, membuat file *.yaml* untuk konfigurasi alur kerja (*workflow*) dari CI/CD dan disimpan di *.github/workflows/*. *Job* dari alur kerja dibagi menjadi 2 bagian, *build* dan *deploy.* Di bagian *on* dispesifikkan *branch* yang dimana jika terjadi push pada respositori dan *branch* tersebut akan melakukan CI/CD.

Pada bagian *build*, akan dibangun image docker menggunakan file *docker-compose.ci.yml* dan variabel yang telah dimasukkan. Pada bagian *deploy*, akan dilakukan deployment ke server 3DVT menggunakan image yang telah dibuat dengan menggunakan *docker-compose.backend.yml*.

```yaml
name: Continuous Integration and Delivery

on:
  push:
    branches:
      - develop
  pull_request:
    branches:
      - develop

env:
  BACKEND_IMAGE: ghcr.io/$(echo $GITHUB_REPOSITORY | tr '[:upper:]' '[:lower:]')/app_backend

jobs:
  build:
    name: Build Docker Images
    runs-on: ubuntu-latest
    steps:
      - name: Checkout develop
        uses: actions/checkout@v3
      - name: Add environment variables to Github
        run: |
          echo "DEBUG=True" >> .env
          echo "TARGET=staging" >> .env
          echo "SQL_ENGINE=django.db.backends.postgresql" >> .env
          echo "DATABASE=postgres" >> .env
          echo "DJANGO_ALLOWED_HOSTS='${{ secrets.DJANGO_ALLOWED_HOSTS }}'" >> .env
          echo "CORS_ALLOWED_ORIGINS='https://3dvtusg.com:8000 https://localhost:8000 https://127.0.0.1:8000'" 
          echo "CSRF_TRUSTED_ORIGINS='https://3dvtusg.com:8000 https://localhost:8000 https://127.0.0.1:8000'" 
          echo "SECRET_KEY=${{ secrets.SECRET_KEY }}" >> .env
          echo "SQL_DATABASE=${{ secrets.SQL_DATABASE }}" >> .env
          echo "SQL_USER=${{ secrets.SQL_USER }}" >> .env
```

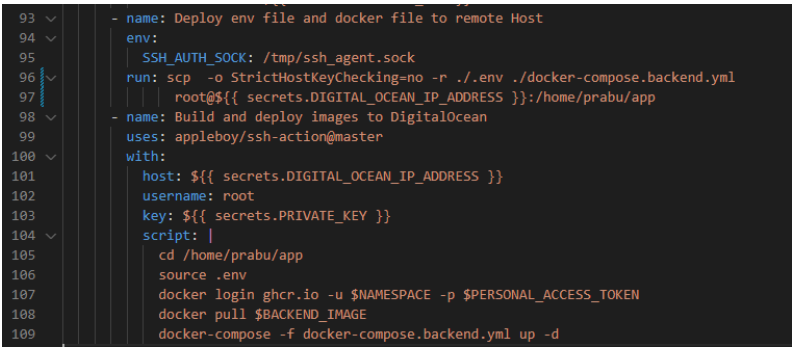
Source Code 4.8 *.github/workflows/backend.yaml [1]*

```
          echo "SQL_PASSWORD=${{ secrets.SQL_PASSWORD }}" >> .env
          echo "SQL_HOST=${{ secrets.SQL_HOST }}" >> .env
          echo "SQL_PORT=${{ secrets.SQL_PORT }}" >> .env
          echo "PERSONAL_ACCESS_TOKEN=${{ secrets.PERSONAL_ACCESS_TOKEN }}" >> .env
          echo "DIGITAL_OCEAN_ACCESS_TOKEN=${{ secrets.DIGITAL_OCEAN_ACCESS_TOKEN }}" >> .env
      - name: Set environment variables
        run: |
          echo "BACKEND_IMAGE=$(echo ${{env.BACKEND_IMAGE}} )" >> $GITHUB_ENV
      - name: Log in to GitHub Packages
        run: echo ${PERSONAL_ACCESS_TOKEN} | docker login ghcr.io -u ${{ secrets.NAMESPACE }} --password-stdin
        env:
          PERSONAL_ACCESS_TOKEN: ${{ secrets.PERSONAL_ACCESS_TOKEN }}
      - name: Change entrypoint permission
        run: |
          chmod +x backend/entrypoint.prod.sh
          ls -la backend
      - name: Pull images
        run: |
          docker pull ${{ env.BACKEND_IMAGE }} || true
      - name: Build images
        run: |
          docker-compose -f docker-compose.ci.yml build
      - name: Push images
        run: |
          docker push ${{ env.BACKEND_IMAGE }}
  deploy:
    name: Deploy to DigitalOcean
    runs-on: ubuntu-latest
    needs: build
    if: github.ref == 'refs/heads/develop'
    steps:
      - name: Checkout develop
```

Source Code 4.9 *.github/workflows/backend.yaml [2]*

```
        uses: actions/checkout@v3
      - name: Add environment variables to .env
        run: |
          echo "DEBUG=True" >> .env
          echo "TARGET=staging" >> .env
          echo "SQL_ENGINE=django.db.backends.postgresql_psycopg2" >> .env
          echo "DATABASE=postgres" >> .env
          echo "DJANGO_ALLOWED_HOSTS='${{ secrets.DJANGO_ALLOWED_HOSTS }}'" >> .env
          echo "CORS_ALLOWED_ORIGINS='https://3dvtusg.com:8000 https://localhost:8000 https://127.0.0.1:8000'"
          echo "CSRF_TRUSTED_ORIGINS='https://3dvtusg.com:8000 https://localhost:8000 https://127.0.0.1:8000'"
          echo "SECRET_KEY=${{ secrets.SECRET_KEY }}" >> .env
          echo "SQL_DATABASE=${{ secrets.SQL_DATABASE }}" >> .env
          echo "SQL_USER=${{ secrets.SQL_USER }}" >> .env
          echo "SQL_PASSWORD=${{ secrets.SQL_PASSWORD }}" >> .env
          echo "SQL_HOST=${{ secrets.SQL_HOST }}" >> .env
          echo "SQL_PORT=${{ secrets.SQL_PORT }}" >> .env
          echo "BACKEND_IMAGE=${{ env.BACKEND_IMAGE }}" >> .env
          echo "NAMESPACE=${{ secrets.NAMESPACE }}" >> .env
          echo "PERSONAL_ACCESS_TOKEN=${{ secrets.PERSONAL_ACCESS_TOKEN }}" >> .env
          echo "DIGITAL_OCEAN_ACCESS_TOKEN=${{ secrets.DIGITAL_OCEAN_ACCESS_TOKEN }}" >> .env
      - name: Add the private SSH key to the ssh-agent
        env:
          SSH_AUTH_SOCK: /tmp/ssh_agent.sock
        run: |
          mkdir -p ~/.ssh
          ssh-agent -a $SSH_AUTH_SOCK > /dev/null
          ssh-keyscan github.com >> ~/.ssh/known_hosts
          ssh-add - <<< "${{ secrets.PRIVATE_KEY }}"
```

Source Code 4.10 *.github/workflows/backend.yaml [3]*



```
- name: Deploy env file and docker file to remote Host
  env:
    SSH_AUTH_SOCK: /tmp/ssh_agent.sock
  run: scp  -o StrictHostKeyChecking=no -r ./.env ./docker-compose.backend.yml
       root@${{ secrets.DIGITAL_OCEAN_IP_ADDRESS }}:/home/prabu/app
- name: Build and deploy images to DigitalOcean
  uses: appleboy/ssh-action@master
  with:
    host: ${{ secrets.DIGITAL_OCEAN_IP_ADDRESS }}
    username: root
    key: ${{ secrets.PRIVATE_KEY }}
    script: |
      cd /home/prabu/app
      source .env
      docker login ghcr.io -u $NAMESPACE -p $PERSONAL_ACCESS_TOKEN
      docker pull $BACKEND_IMAGE
      docker-compose -f docker-compose.backend.yml up -d
```

Source Code 4.11 *.github/workflows/backend.yaml [4]*

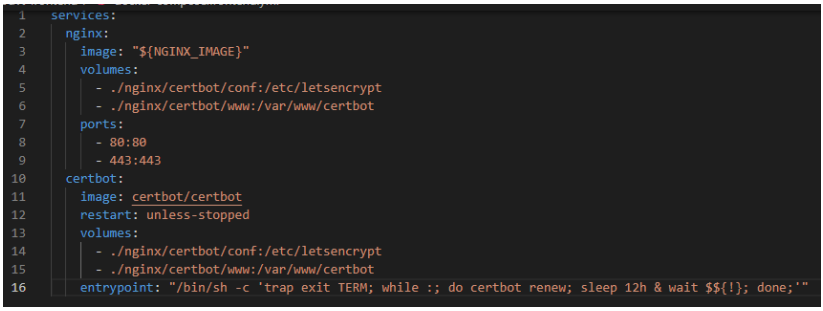
4.8.CI/CD Frontend

Sama seperti pada backend, dibuat file *docker-compose.ci.yml* yang akan digunakan sebagai awal pembuatan image. Terdapat 2 *service* yang dipakai, yang pertama adalah webserver Nginx dan kedua adalah certbot yang digunakan untuk otomasi pembaruan sertifikat Let's Encrypt untuk mengaktifkan HTTPS pada website 3DVT. Pada webserver digunakan port 80 untuk HTTP dan 443 untuk HTTPS.

```
services:      You, 10 months ago • dockerized react-app
  nginx:
    restart: unless-stopped
    build:
      context: .
      dockerfile: Dockerfile
      cache_from:
        - "${NGINX_IMAGE}"
    image: "${NGINX_IMAGE}"
    volumes:
      - ./nginx/certbot/conf:/etc/letsencrypt
      - ./nginx/certbot/www:/var/www/certbot
    ports:
      - 80:80
      - 443:443
  certbot:
    image: certbot/certbot
    restart: unless-stopped
    volumes:
      - ./nginx/certbot/conf:/etc/letsencrypt
      - ./nginx/certbot/www:/var/www/certbot
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew; sleep 12h & wait $${!}; done;'"
```

Source Code 4.12 *docker-compose.ci.yml untuk Frontend*

Dibuat juga file *docker-compose.frontend.yml* yang akan digunakan untuk deploy ke server 3DVT dimana akan menggunakan image yang telah dibangun di awal.



```
services:
  nginx:
    image: "${NGINX_IMAGE}"
    volumes:
      - ./nginx/certbot/conf:/etc/letsencrypt
      - ./nginx/certbot/www:/var/www/certbot
    ports:
      - 80:80
      - 443:443
  certbot:
    image: certbot/certbot
    restart: unless-stopped
    volumes:
      - ./nginx/certbot/conf:/etc/letsencrypt
      - ./nginx/certbot/www:/var/www/certbot
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew; sleep 12h & wait $${!}; done;'"
```

Source Code 4.13 *docker-compose.frontend.yml*

Sama dengan backend, alur kerja pada frontend akan dibagi menjadi 2 bagian, yaitu *build* dan *deploy*. Di bagian *on* dispesifikkan *branch* yang dimana jika terjadi push pada respositori dan *branch* tersebut akan melakukan CI/CD. Pada

bagian *build*, akan dibangun image docker menggunakan file *docker-compose.ci.yml* dan variabel yang telah dimasukkan. Pada bagian *deploy*, akan dilakukan deployment ke server 3DVT menggunakan image yang telah dibuat dengan menggunakan *docker-compose.frontend.yml*.

```yaml
name: Continuous Integration and Delivery

on:
  push:
    branches: [ "develop" ]
  pull_request:
    branches: [ "develop" ]

env:
  NGINX_IMAGE: ghcr.io/$(echo $GITHUB_REPOSITORY | tr '[:upper:]' '[:lower:]')/app_nginx

jobs:

  build:
    name: Build Docker Images
    runs-on: ubuntu-latest
    steps:
      - name: Checkout develop
        uses: actions/checkout@v3
      - name: Set environment variables
        run: |
          echo "NGINX_IMAGE=$(echo ${{env.NGINX_IMAGE}} )" >> $GITHUB_ENV
      - name: Log in to GitHub Packages
        run: echo ${PERSONAL_ACCESS_TOKEN} | docker login ghcr.io -u ${{ secrets.NAMESPACE }} --password-stdin
        env:
          PERSONAL_ACCESS_TOKEN: ${{ secrets.PERSONAL_ACCESS_TOKEN }}
      - name: Pull images
        run: |
          docker pull ${{ env.NGINX_IMAGE }} || true
      - name: Build images
        run: |
          docker-compose -f docker-compose.ci.yml build
```

Source Code 4.14 *.github/workflows/frontend.yaml [1]*

```
33        - name: Push images
34          run: |
35            docker push ${{ env.NGINX_IMAGE }}
36
37    deploy:
38      name: Deploy to DigitalOcean
39      runs-on: ubuntu-latest
40      needs: build
41      if: github.ref == 'refs/heads/develop'
42      steps:      You, 10 months ago • fix: workflow github actions …
43        - name: Checkout develop
44          uses: actions/checkout@v3
45        - name: Add environment variables to .env
46          run: |
47            echo "NGINX_IMAGE=${{ env.NGINX_IMAGE }}" >> .env
48            echo "NAMESPACE=${{ secrets.NAMESPACE }}" >> .env
49            echo "PERSONAL_ACCESS_TOKEN=${{ secrets.PERSONAL_ACCESS_TOKEN }}" >> .env
50        - name: Add the private SSH key to the ssh-agent
51          env:
52            SSH_AUTH_SOCK: /tmp/ssh_agent.sock
53          run: |
54            mkdir -p ~/.ssh
55            ssh-agent -a $SSH_AUTH_SOCK > /dev/null
56            ssh-keyscan github.com >> ~/.ssh/known_hosts
57            ssh-add - <<< "${{ secrets.PRIVATE_KEY }}"
```

Source Code 4.15 *.github/workflows/frontend.yaml [2]*

```
58        - name: Deploy env file and docker file to remote Host
59          env:
60            SSH_AUTH_SOCK: /tmp/ssh_agent.sock
61          run: |
62            scp  -o StrictHostKeyChecking=no -r ./.env ./docker-compose.frontend.yml init-letsencrypt.sh
63            root@${{ secrets.DIGITAL_OCEAN_IP_ADDRESS }}:/home/prabu/app
64        - name: Build and deploy images to DigitalOcean
65          uses: appleboy/ssh-action@master
66          with:
67            host: ${{ secrets.DIGITAL_OCEAN_IP_ADDRESS }}
68            username: root
69            key: ${{ secrets.PRIVATE_KEY }}
70            script: |
71              cd /home/prabu/app
72              source .env
73              chmod +x ./init-letsencrypt.sh
74              docker login ghcr.io -u $NAMESPACE -p $PERSONAL_ACCESS_TOKEN
75              docker pull $NGINX_IMAGE
76              docker-compose -f docker-compose.frontend.yml up -d      You, 5 months ago • fix: typo on y
```

Source Code 4.16 *.github/workflows/frontend.yaml [3]*

[Halaman ini sengaja dikosongkan]

BAB V PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tahap uji coba terhadap Website Sistem Cerdas 3DVT yang dapat diakses pada https://3dvtusg.com/. Pengujian dilakukan untuk memastikan pengimplementasian infrastruktur dapat menjalankan fungsionalitas dari backend dan otomasi pada deployment dapat berjalan.

5.1. Tujuan Pengujian

Pengujian dilakukan terhadap Website Sistem Cerdas 3DVT guna menguji fungsionalitas dari backend dan frontend 3DVT serta otomasi pada deployment dapat berjalan.

5.2. Kriteria Pengujian

Penilaian atas pencapaian tujuan pengujian didapatkan dengan memerhatikan beberapa hasil berikut.

a. Infrastruktur dapat melayani tampilan website.
b. Infrastruktur dapat melayani API website.
c. Infrastruktur dapat melayani query data dari aplikasi ke database.
d. Infrastruktur dapat melayani request aplikasi ke Digital Ocean Spaces.
e. CI/CD dapat berjalan pada bagian backend maupun frontend 3DVT.

5.3. Skenario Pengujian

Skenario pengujian dilakukan untuk menguji infrastruktur berdasarkan kriteria pengujian. Pengujian dilakukan bersama pembimbing lapangan, Mas Arta Kusuma Hernanda, untuk mengecek backend dan frontend aplikasi.

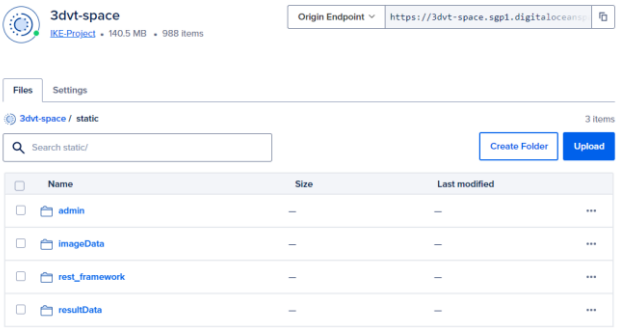
a. Tampilan website 3DVT dapat diakses melalui https://3dvtusg.com/ dan telah menggunakan HTTPS.

b. API 3DVT dapat bekerja dengan menguji setiap fungsionalitas pada sistem 3DVT, dapat dilakukan melalui https://3dvtusg.com/api/ .
c. Data yang di-query dari aplikasi dapat direspon oleh database 3DVT.

```
backenddb=> \dt
                    List of relations
 Schema |                Name                | Type  |    Owner
--------+------------------------------------+-------+--------------
 public | api_filedata                       | table | backendadmin
 public | api_imagedata                      | table | backendadmin
 public | api_landingpage                    | table | backendadmin
 public | api_publication                    | table | backendadmin
 public | api_reconstruction                 | table | backendadmin
 public | api_reconstruction_files           | table | backendadmin
 public | api_researcher                     | table | backendadmin
 public | api_segmentation                   | table | backendadmin
 public | api_segmentation_images            | table | backendadmin
 public | api_suggestions                    | table | backendadmin
 public | api_users                          | table | backendadmin
 public | api_users_groups                   | table | backendadmin
 public | api_users_user_permissions         | table | backendadmin
 public | auth_group                         | table | backendadmin
 public | auth_group_permissions             | table | backendadmin
 public | auth_permission                    | table | backendadmin
 public | authtoken_token                    | table | backendadmin
 public | django_admin_log                   | table | backendadmin
 public | django_content_type                | table | backendadmin
 public | django_migrations                  | table | backendadmin
 public | django_session                     | table | backendadmin
 public | token_blacklist_blacklistedtoken   | table | backendadmin
 public | token_blacklist_outstandingtoken   | table | backendadmin
(23 rows)
```

Gambar 5.1 *List Tabel pada Database 3DVT*

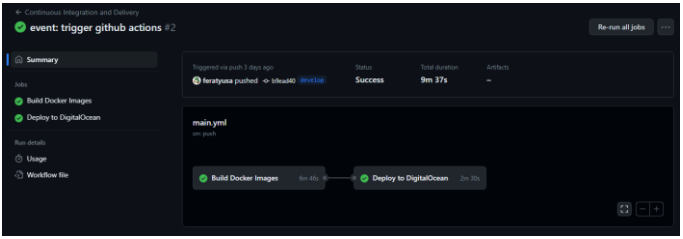
d. API Spaces dapat digunakan oleh sistem 3DVT.



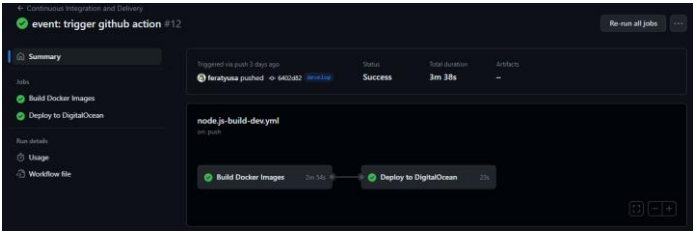
Gambar 5.2 *Digital Ocean Spaces 3DVT*

e. CI/CD berjalan saat melakukan push ke repositori backend maupun frontend 3DVT.
Repositori Backend:
- *https://github.com/feratyusa/3dvt-backend/*

Repositori Frontend:
- *https://github.com/feratyusa/3dvt-frontend/*



Gambar 5.3 *Actions CI/CD pada repositori Backend*



Gambar 5.4 *Actions CI/CD pada repositori Frontend*

5.4. Evaluasi Pengujian

Hasil pengujian dilakukan berdasarkan skenario pengujian yang telah dilakukan bersama Pembimbing Lapangan. Hasil pengujian adalah sebagai berikut.

No.	Kriteria Pengujian	Hasil Pengujian
1.	Infrastruktur dapat melayani tampilan website.	Terpenuhi
2.	Infrastruktur dapat melayani API website.	Terpenuhi
3.	Infrastruktur dapat melayani query data dari aplikasi ke database.	Terpenuhi
4.	Infrastruktur dapat melayani request aplikasi ke Digital Ocean Spaces.	Terpenuhi
5.	CI/CD dapat berjalan pada bagian backend maupun frontend 3DVT.	Terpenuhi

Tabel 5.1 *Hasil Evaluasi*

[Halaman ini sengaja dikosongkan]

BAB VI KESIMPULAN DAN SARAN

6.1. Kesimpulan

Kesimpulan yang didapat setelah melakukan implementasi dan konfigurasi Sistem Cerdas 3DVT pada kegiatan kerja praktek ini adalah sebagai berikut :

- Infrastruktur sistem berjalan dengan baik dan sesuai dengan kebutuhan.
- CI/CD pada Backend dan Frontend dapat digunakan, serta dapat membantu jikalau pembangunan sistem cerdas 3DVT dilanjutkan.
- Docker dan GitHub Actions cocok digunakan dalam pengembangan infrastruktur sistem cerdas 3DVT karena kemudahan melakukan konfigurasi pada server.

6.2. Saran

Saran untuk infrastruktur sistem cerdas 3DVT adalah sebagai berikut.

- Dapat menggunakan Kubernetes dalam melakukan *containerized* infrastruktur sistem
- Dapat mengimplementasikan *Load Balancing* dalam infrastruktur agar lebih memperata traffic pada sistem serta mengurangi resiko sistem *down*.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] Setiawan, Rony. 2021. Apa Itu Docker? Apa Kegunaan dan Kelebihannya? https://www.dicoding.com/blog/apa-itu-docker/ (diakses tanggal 1 Agustus 2023)

[2] Garcia, Carlos. 2023. Apa itu DigitalOcean, dan Mengapa Anda Harus Mempertimbangkan untuk Menggunakannya? https://appmaster.io/id/blog/apa-itu-samudra-digital (diakses tanggal 1 Agustus 2023).

[3] Yulian, Nano. 2020. Berkenalan Dengan Docker Compose (Docker) Contoh kasus Dockerize PHP , Apache, dan PostgreSQL. https://medium.com/@nanoyulian/berkenalan-dengan-docker-compose-docker-63208f45ca4c/ (diakses tanggal 1 Agustus 2023)

[4] GitHub. 2023. https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions (diakses tanggal 1 Agustus 2023)

[5] K, Yasin. 2019. Apa Itu Nginx dan Cara Kerjanya**.** https://www.niagahoster.co.id/blog/nginx-adalah/ (diakses tanggal 1 Agustus 2023)

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Nama	: Ega Prabu Pamungkas
Tempat, Tanggal Lahir	: Mataram, 31 Januari 2001
Jenis Kelamin	: Laki-laki
Telepon	: +62 812 1659 7878
Email	: prabuega56@gmail.com

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC ITS
Angkatan	: 2019
Semester	: 8 (delapan)