

I

# DOCKER



# Pengertian Docker

---

Docker adalah layanan manajemen kontainer. Kata kunci Docker adalah mengembangkan, mengirimkan, dan menjalankan di mana saja.

Ide keseluruhan Docker adalah agar develop dapat mengembangkan aplikasi dengan mudah, mengirimkannya ke dalam container yang kemudian dapat dideploy di mana saja



# Komponen Utama Docker

## Docker Registry

---

Docker Registry merupakan tempat penyimpanan (publik atau privat) dimana kita bisa mengupload dan mendownload images. Registry public docker disebut dengan Docker Hub yang didalamnya terdapat banyak image, baik image yang dibuat sendiri ataupun image yang lain.



# Komponen Utama Docker

## Docker Images

---

Docker images merupakan sebuah template yang bersifat read-only. Image ini digunakan untuk menjalankan container.

Docker menyediakan cara yang sederhana untuk membangun image baru atau merubah image yang sudah ada. Dibangun oleh Docker user. Disimpan di Docker Hub atau registri lokal.



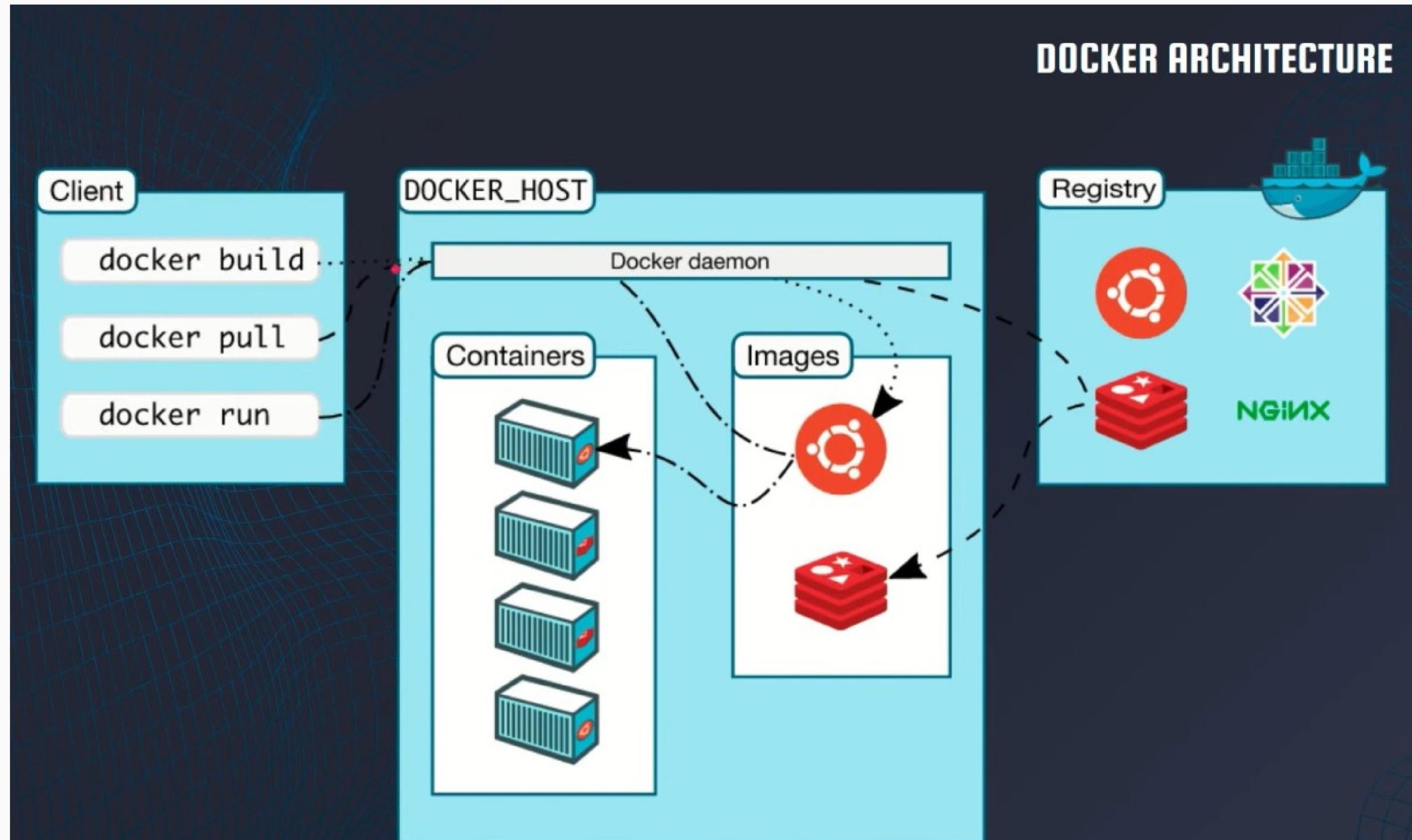
# Komponen Utama Docker

## Docker Container

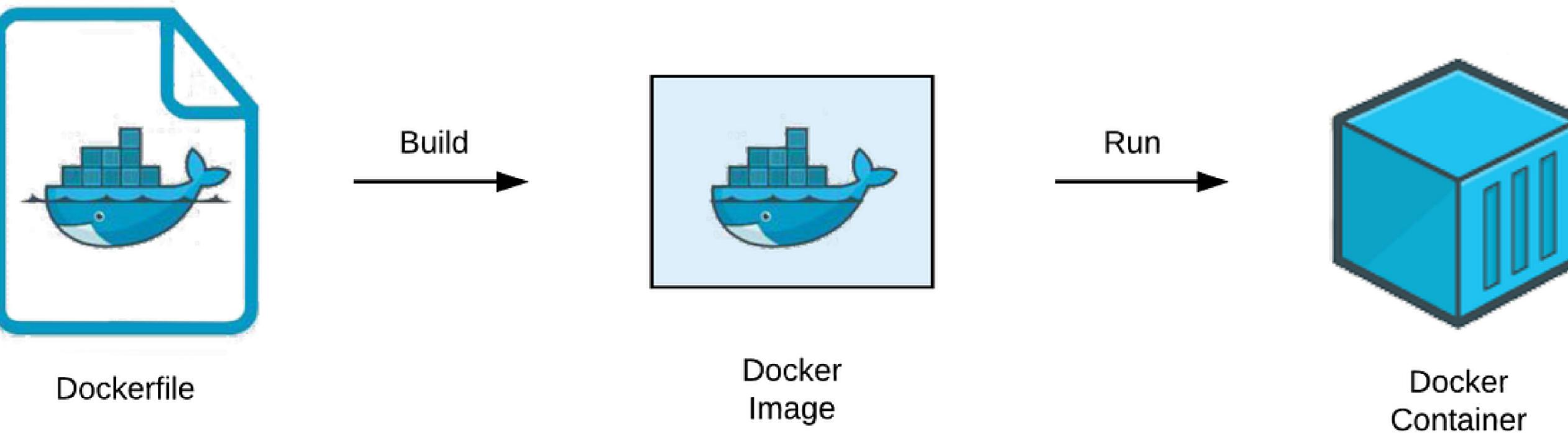
---

Docker container merupakan sebuah image yang bersifat read-write yang berjalan di atas image. Docker menggunakan union-file sistem sebagai back-end file sistem containernya, dimana setiap perubahan yang disimpan pada container akan menyebabkan terbentuknya layer baru di atas base image. Jadi container merupakan layer dimana kita bisa melakukan instalasi aplikasi di dalamnya.



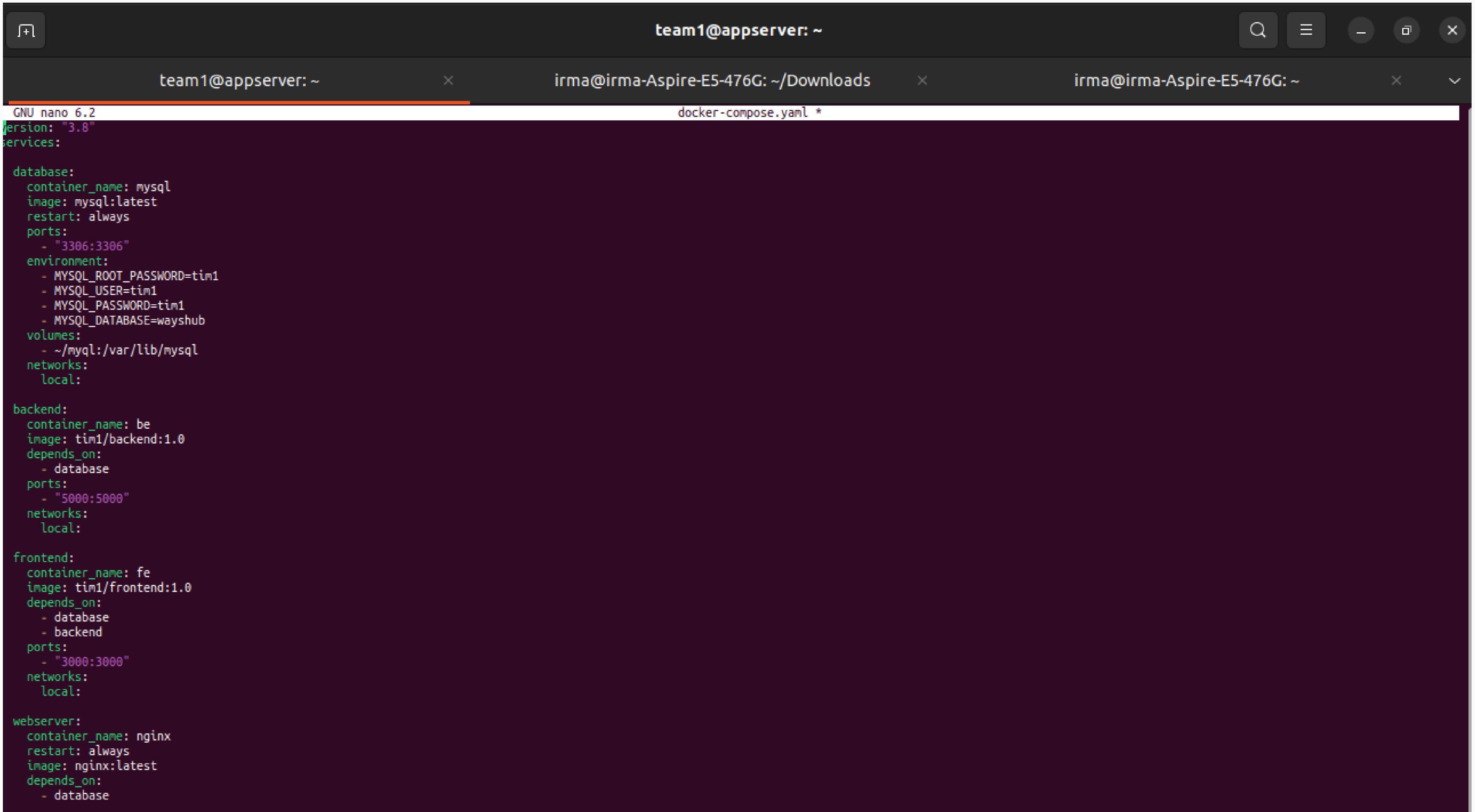


7



8

# Dokumentasi



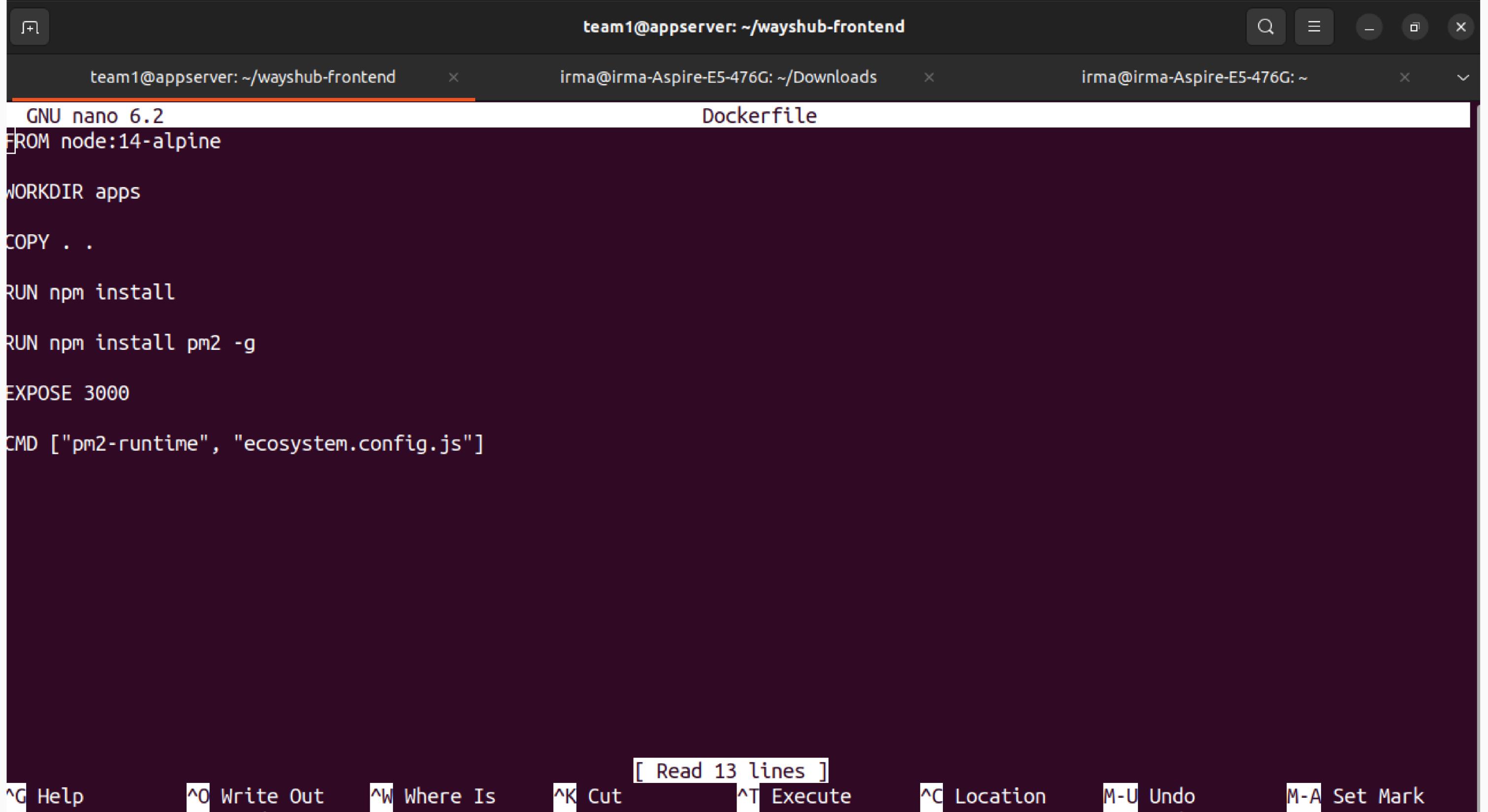
The screenshot shows a terminal window with three tabs. The active tab, located at the bottom left, displays a Docker Compose configuration file named `docker-compose.yaml`. The file defines four services: `database`, `backend`, `frontend`, and `webserver`. The `database` service uses a MySQL container with port 3306 mapped to 3306. It sets environment variables for MySQL root password, user, and database, and maps its data volume to the host's `/myql` directory. The `backend` service uses a custom image `tim1/backend:1.0` and depends on the `database` service, with port 5000 mapped to 5000. The `frontend` service uses a custom image `tim1/frontend:1.0` and depends on both `database` and `backend`, with port 3000 mapped to 3000. The `webserver` service uses an Nginx container with port 80 mapped to 80, always restarts, and depends on the `database` service.

```
team1@appserver: ~
team1@appserver: ~
irma@irma-Aspire-E5-476G: ~/Downloads
irma@irma-Aspire-E5-476G: ~

GNU nano 6.2
version: "3.8"
services:
  database:
    container_name: mysql
    image: mysql:latest
    restart: always
    ports:
      - "3306:3306"
    environment:
      - MYSQL_ROOT_PASSWORD=tim1
      - MYSQL_USER=tim1
      - MYSQL_PASSWORD=tim1
      - MYSQL_DATABASE=wayshub
    volumes:
      - ~/myql:/var/lib/mysql
    networks:
      local:
  backend:
    container_name: be
    image: tim1/backend:1.0
    depends_on:
      - database
    ports:
      - "5000:5000"
    networks:
      local:
  frontend:
    container_name: fe
    image: tim1/frontend:1.0
    depends_on:
      - database
      - backend
    ports:
      - "3000:3000"
    networks:
      local:
  webserver:
    container_name: nginx
    restart: always
    image: nginx:latest
    depends_on:
      - database
```

9

# Dokumentasi Docker File



The screenshot shows a terminal window with three tabs. The active tab is titled "Dockerfile" and contains the following Dockerfile code:

```
GNU nano 6.2
FROM node:14-alpine

WORKDIR apps

COPY . .

RUN npm install

RUN npm install pm2 -g

EXPOSE 3000

CMD ["pm2-runtime", "ecosystem.config.js"]
```

The terminal interface includes a status bar at the bottom with various keyboard shortcuts and a message "[ Read 13 lines ]". On the left side of the terminal, there is a vertical column of three colored circles: white, black, and gold.

10

# Dokumentasi Docker File



The screenshot shows a terminal window with three tabs. The active tab is titled "Dockerfile" and contains the following Dockerfile code:

```
team1@appserver: ~/wayshub-backend
team1@appserver: ~/wayshub-backend
irma@irma-Aspire-E5-476G: ~/Downloads
irma@irma-Aspire-E5-476G: ~

GNU nano 6.2
FROM node:14

WORKDIR apps

COPY . .

RUN npm install

RUN npm install sequelize-cli -g

RUN npm install pm2 -g

RUN npx sequelize db:migrate

EXPOSE 5000

CMD ["pm2-runtime", "ecosystem.config.js"]
```

The terminal interface includes a status bar at the bottom with various keyboard shortcuts and a message "[ Read 17 lines ]". On the far left, there is a vertical column of three colored circles: white, black, and gold.

11

# Dokumentasi reverse\_proxy

```
GNU nano 6.2
server {
    listen 443 ssl;
    server_name jenkins.irma.studentdumbways.my.id;

    ssl_certificate /etc/nginx/certs/fullchain.pem;
    ssl_certificate_key /etc/nginx/certs/privkey.pem;

    location / {
        proxy_pass http://jenkins:8080/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-Port $server_port;
    }
}
```

The terminal window shows the configuration of an Nginx reverse proxy. The configuration file, `reverse_proxy.conf`, defines a single server block. This block listens on port 443 using SSL certificates from `/etc/nginx/certs/fullchain.pem` and `/etc/nginx/certs/privkey.pem`. It proxies requests to `http://jenkins:8080/`, setting various headers to forward the original client information.

At the bottom of the terminal, a status bar indicates "Read 17 lines". The keyboard navigation keys are listed at the bottom:

- ^G Help
- ^O Write Out
- ^W Where Is
- ^K Cut
- ^T Execute
- ^C Location
- M-U Undo
- M-A Set Mark
- M-[ To Bracket
- ^X Exit
- ^R Read File
- ^V Replace
- ^U Paste
- ^J Justify
- ^/ Go To Line
- M-E Redo
- M-6 Copy
- ^Q Where Was

12

# Dokumentasi docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nurliza/backend	latest	13b20bdbf80e	2 days ago	1GB
nurliza/frontend	latest	b4d0cf5c6d68	3 days ago	340MB
tim1/frontend	1.0	fefe0c2c898a	5 days ago	340MB
tim1/backend	1.0	8cf61deb4723	5 days ago	1GB
mysql	latest	73246731c4b0	2 weeks ago	619MB
nginx	latest	d453dd892d93	2 months ago	187MB



13

# Dokumentasi built aplikasi frontend

```
1@appserver:~/wayshub-frontend$ ls
Dockerfile  README.md  debug.log  ecosystem.config.js  package-lock.json  package.json  public  src  yarn.lock
1@appserver:~/wayshub-frontend$ docker build -t tim1/frontend:1.0 .
Building 231.7s (10/10) FINISHED
[internal] load .dockerignore
=> transferring context: 2B
[internal] load build definition from Dockerfile
=> transferring dockerfile: 181B
[internal] load metadata for docker.io/library/node:14-alpine
[1/5] FROM docker.io/library/node:14-alpine@sha256:434215b487a329c9e867202ff89e704d3a75e554822e07f3e0c0f9e606121b33
=> resolve docker.io/library/node:14-alpine@sha256:434215b487a329c9e867202ff89e704d3a75e554822e07f3e0c0f9e606121b33
=> sha256:0dac3dc27b1ad570e6c3a7f7cd29e88e7130ff0cad31b2ec5a0f222fbe971bdb 6.44kB / 6.44kB
=> sha256:434215b487a329c9e867202ff89e704d3a75e554822e07f3e0c0f9e606121b33 1.43kB / 1.43kB
=> sha256:4e84c956cd276af9ed14a8b2939a734364c2b0042485e90e1b97175e73dfd548 1.16kB / 1.16kB
=> sha256:f56be85fc22e46face30e2c3de3f7fe7c15f8fd7c4e5add29d7f64b87abdaa09 3.37MB / 3.37MB
=> sha256:8f665685b215c7daf9164545f1bbdd74d800af77d0d267db31fe0345c0c8fb8b 37.17MB / 37.17MB
=> sha256:e5fca6c395a62ec277102af9e5283f6edb43b3e4f20f798e3ce7e425be226ba6 2.37MB / 2.37MB
=> extracting sha256:f56be85fc22e46face30e2c3de3f7fe7c15f8fd7c4e5add29d7f64b87abdaa09
=> sha256:561cb69653d56a9725be56e02128e4e96fb434a8b4b4decf2bdeb479a225feaf 448B / 448B
=> extracting sha256:8f665685b215c7daf9164545f1bbdd74d800af77d0d267db31fe0345c0c8fb8b
=> extracting sha256:e5fca6c395a62ec277102af9e5283f6edb43b3e4f20f798e3ce7e425be226ba6
=> extracting sha256:561cb69653d56a9725be56e02128e4e96fb434a8b4b4decf2bdeb479a225feaf
[internal] load build context
=> transferring context: 5.51MB
[2/5] WORKDIR apps
[3/5] COPY . .
[4/5] RUN npm install
131.49
```

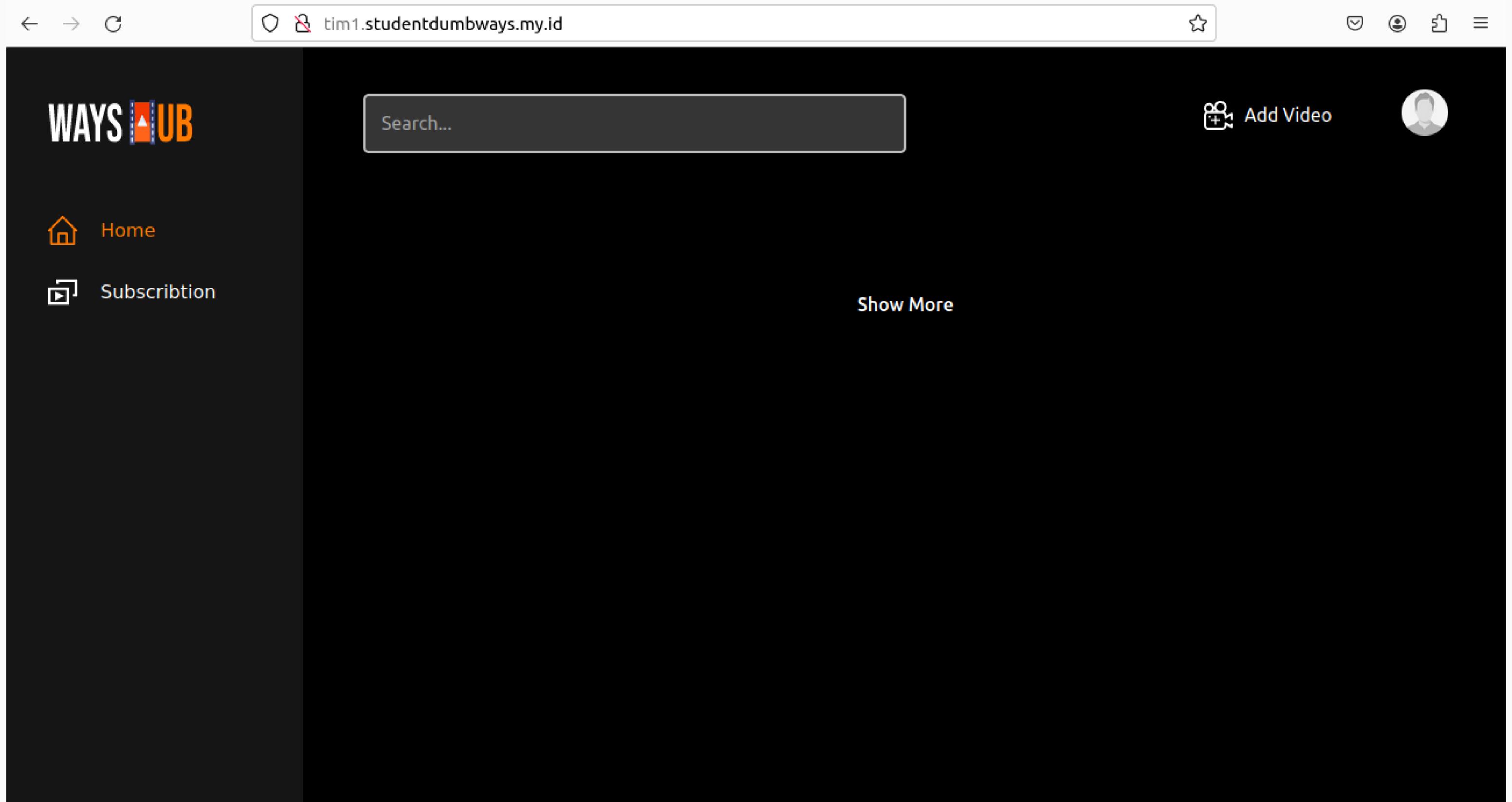
14

# Dokumentasi built aplikasi backend

```
3306/tcp, 33060/tcp
team1@appserver:~/wayshub-backend$ docker build -t tim1/backend:1.0 .
[+] Building 47.8s (12/12) FINISHED                                            docker:default
=》 [internal] load build definition from Dockerfile                         0.1s
=》 => transferring dockerfile: 238B                                         0.0s
=》 [internal] load .dockerignore                                         0.1s
=》 => transferring context: 2B                                           0.0s
=》 [internal] load metadata for docker.io/library/node:14                  1.0s
=》 [1/7] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa 0.0s
=》 [internal] load build context                                         0.2s
=》 => transferring context: 3.63kB                                       0.1s
=》 CACHED [2/7] WORKDIR apps                                         0.0s
=》 CACHED [3/7] COPY . .                                              0.0s
=》 CACHED [4/7] RUN npm install                                         0.0s
=》 CACHED [5/7] RUN npm install sequelize-cli -g                      0.0s
=》 CACHED [6/7] RUN npm install pm2 -g                                 0.0s
=》 [7/7] RUN npx sequelize db:migrate                                7.1s
=》 exporting to image                                                 38.8s
=》 => exporting layers                                               38.7s
=》 => writing image sha256:f2affff90665271882dddcfa79d9ab8b9575f2a1b5df2cc9e1f11387eea367f8c 0.0s
=》 => naming to docker.io/tim1/backend:1.0                           0.0s
team1@appserver:~/wayshub-backend$ cd
team1@appserver:~$ cd wayshub-frontend/
team1@appserver:~/wayshub-frontend$ ls
Dockerfile README.md debug.log ecosystem.config.js package-lock.json package.json public src yarn.lock
team1@appserver:~/wayshub-frontend$ docker build -t tim1/frontend:1.0 .
[+] Building 177.3s (9/9) FINISHED                                            docker:default
=》 [internal] load build definition from Dockerfile                         0.5s
=》 => transferring dockerfile: 101B                                         0.0s
```

15

# Dokumentasi aplikasi berjalan





# Jenkins



2

# Pengertian Jenkins

---

1. Jenkins adalah alat otomatisasi open source yang digunakan untuk membangun dan menguji proyek perangkat lunak
  2. Alat ini memudahkan developer untuk mengintegrasikan perubahan pada suatu proyek
  3. Jenkins mencapai integrasi berkelanjutan dengan bantuan plugin
  4. Jenkins cocok untuk membangun pipeline CI/CD karena fleksibilitasnya, keterbukaannya, kemampuan pluginnya, dan sifatnya yang mudah digunakan.
- 

3

# Building CI/CD with Jenkins

---

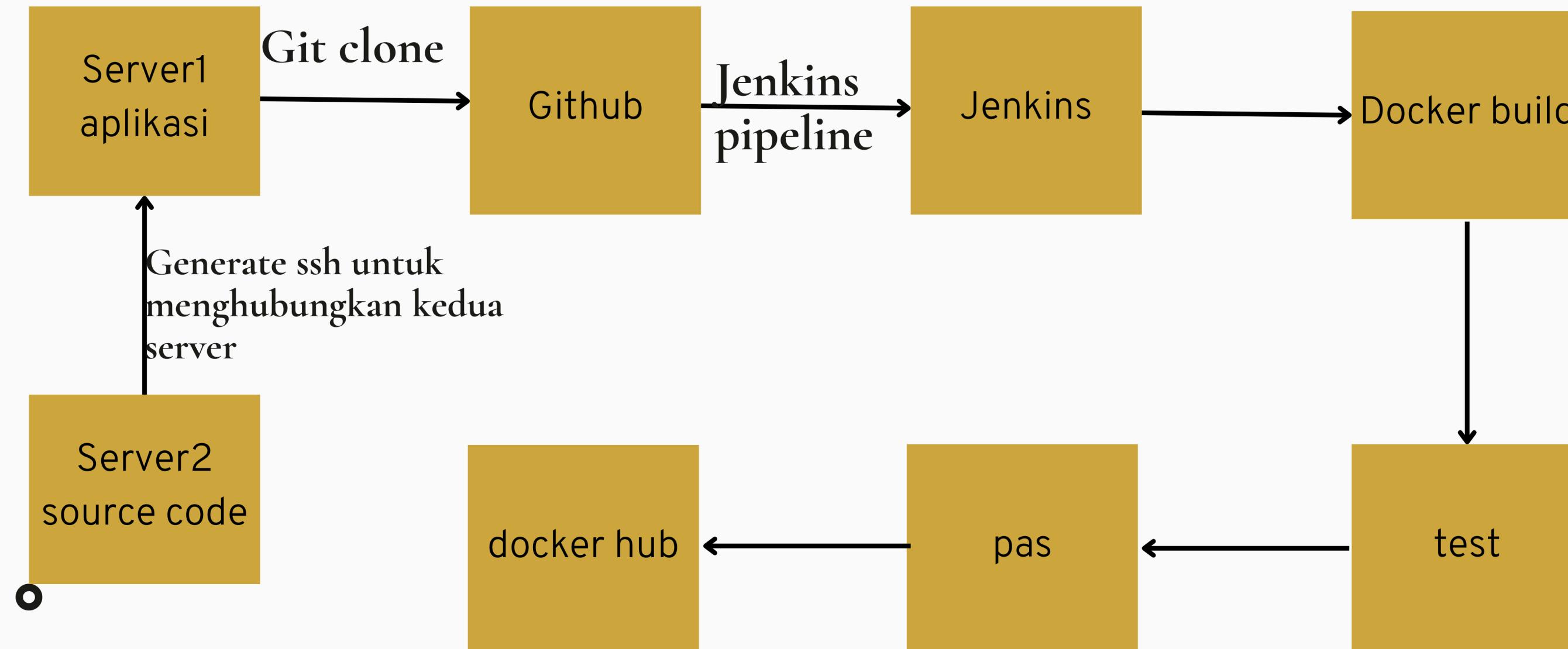
Membangun aplikasi dengan menggunakan jenkins

Dalam penggeraan projek ini menggunakan dua server, server pertama digunakan untuk menjalankan aplikasi dan server ke dua digunakan untuk menjalankan jenkins



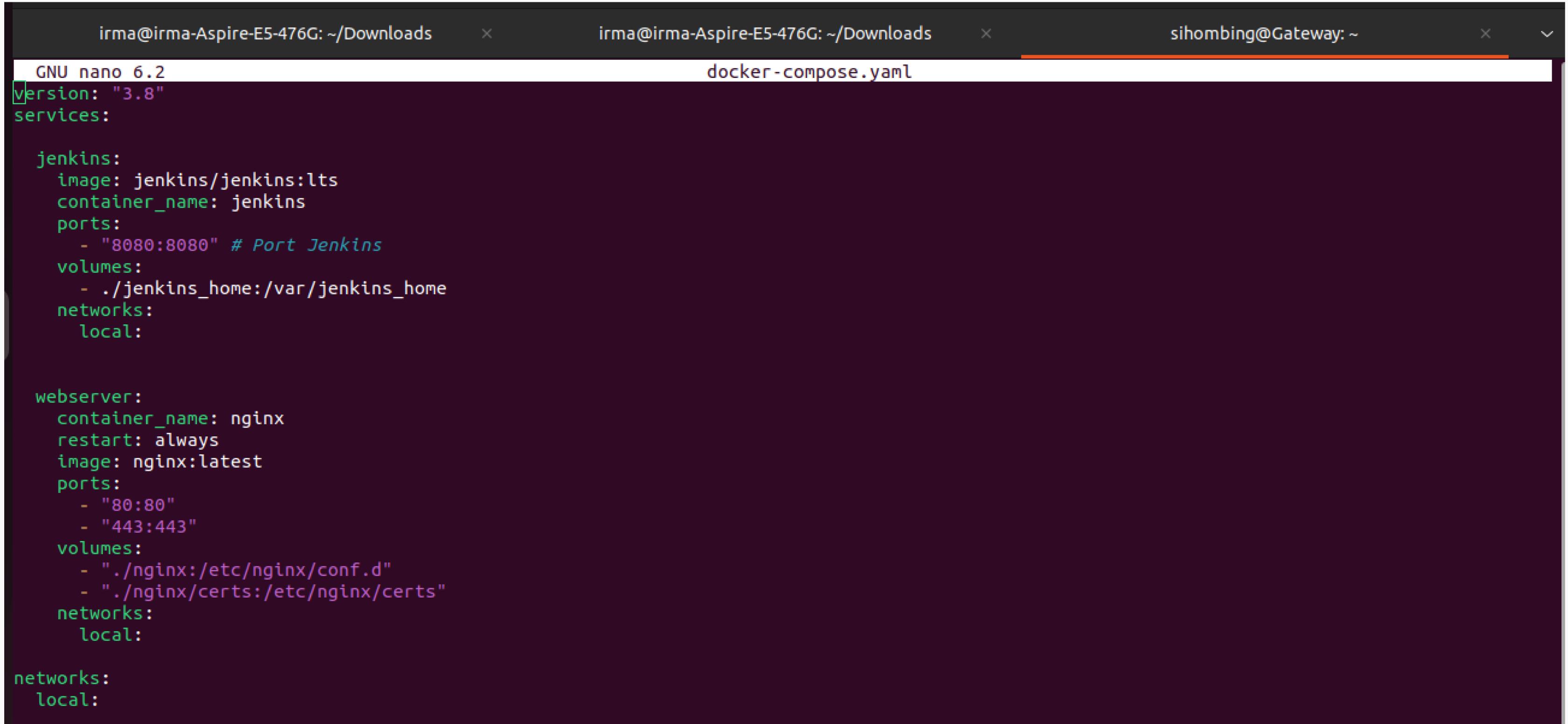
4

# Building CI/CD with Jenkins



## 5

# Dokumentasi

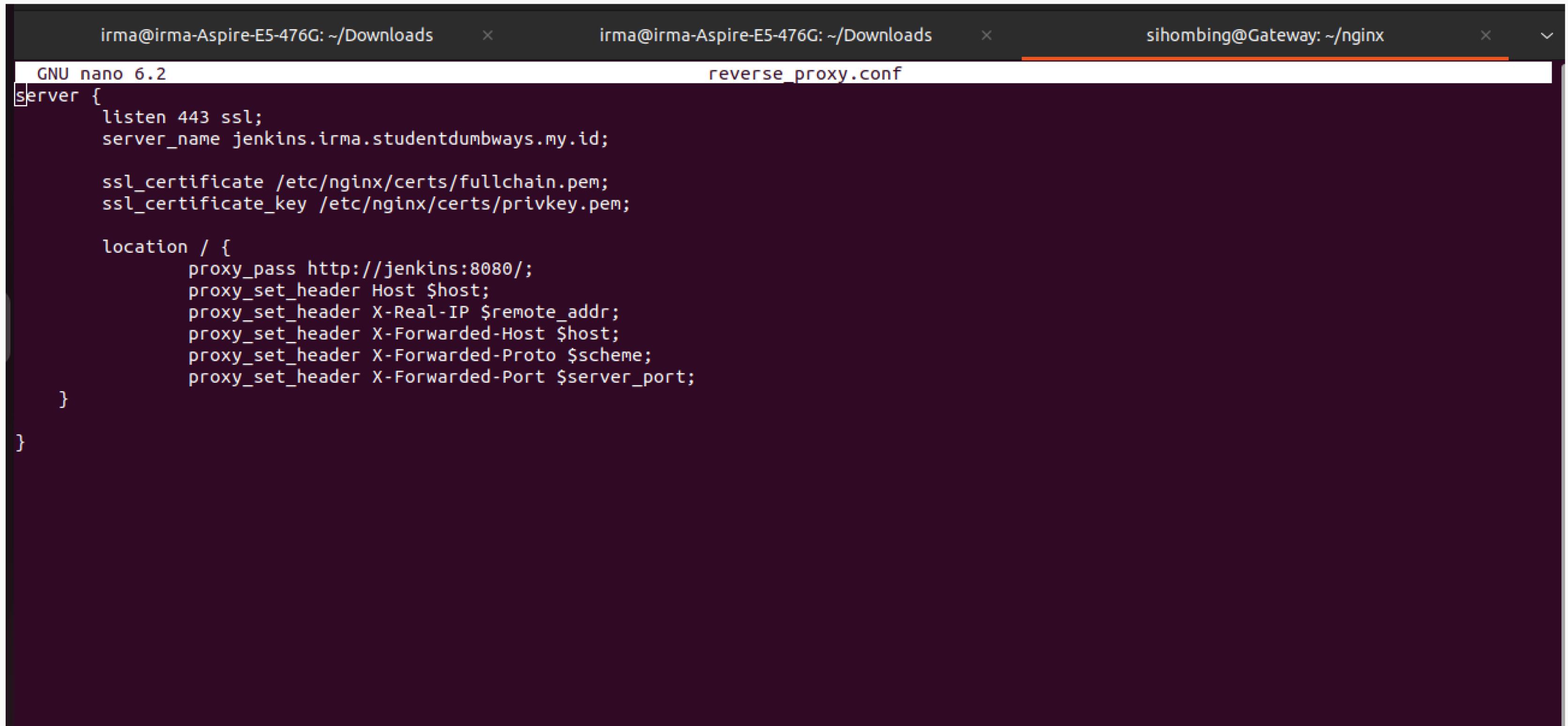


The screenshot shows a terminal window with three tabs. The active tab is titled 'docker-compose.yaml' and contains a Docker Compose configuration file. The file defines two services: 'jenkins' and 'webserver'. The 'jenkins' service uses the 'jenkins/jenkins:lts' image, runs on port 8080, and maps its home directory to '/var/jenkins\_home'. The 'webserver' service uses the 'nginx:latest' image, runs on ports 80 and 443, and maps its configuration and certificate directories to '/etc/nginx/conf.d' and '/etc/nginx/certs' respectively. Both services are connected to a 'local' network.

```
GNU nano 6.2
version: "3.8"
services:
  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    ports:
      - "8080:8080" # Port Jenkins
    volumes:
      - ./jenkins_home:/var/jenkins_home
    networks:
      local:
  webserver:
    container_name: nginx
    restart: always
    image: nginx:latest
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx:/etc/nginx/conf.d
      - ./nginx/certs:/etc/nginx/certs"
    networks:
      local:
networks:
  local:
```

6

# Dokumentasi reverse\_proxy jenkins



The screenshot shows a terminal window with three tabs. The active tab is titled "reverse proxy.conf" and contains the following Nginx configuration file:

```
GNU nano 6.2
server {
    listen 443 ssl;
    server_name jenkins.irma.studentdumbways.my.id;

    ssl_certificate /etc/nginx/certs/fullchain.pem;
    ssl_certificate_key /etc/nginx/certs/privkey.pem;

    location / {
        proxy_pass http://jenkins:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-Port $server_port;
    }
}
```

7

# Dokumentasi pipeline frontend

Pipeline script

Script ?

```
1 pipeline {
2     agent any
3     environment{
4         credential = 'cicd'
5         server = 'team1@103.150.93.34'
6         directory = '/home/team1/wayshub-frontend'
7         branch = 'main'
8         service = 'frontend'
9         image = 'nurliza/frontend'
10    }
11    stages {
12        stage('Pull code dari repository'){
13            steps {
14                sshagent([credential]) {
15                    sh '''ssh -o StrictHostKeyChecking=no ${server} << EOF
16                    docker compose down
17                    cd ${directory}
18                    git pull origin ${branch}
19                    exit+'''
```

# Dokumentasi pipeline frontend

## Pipeline script

### Script

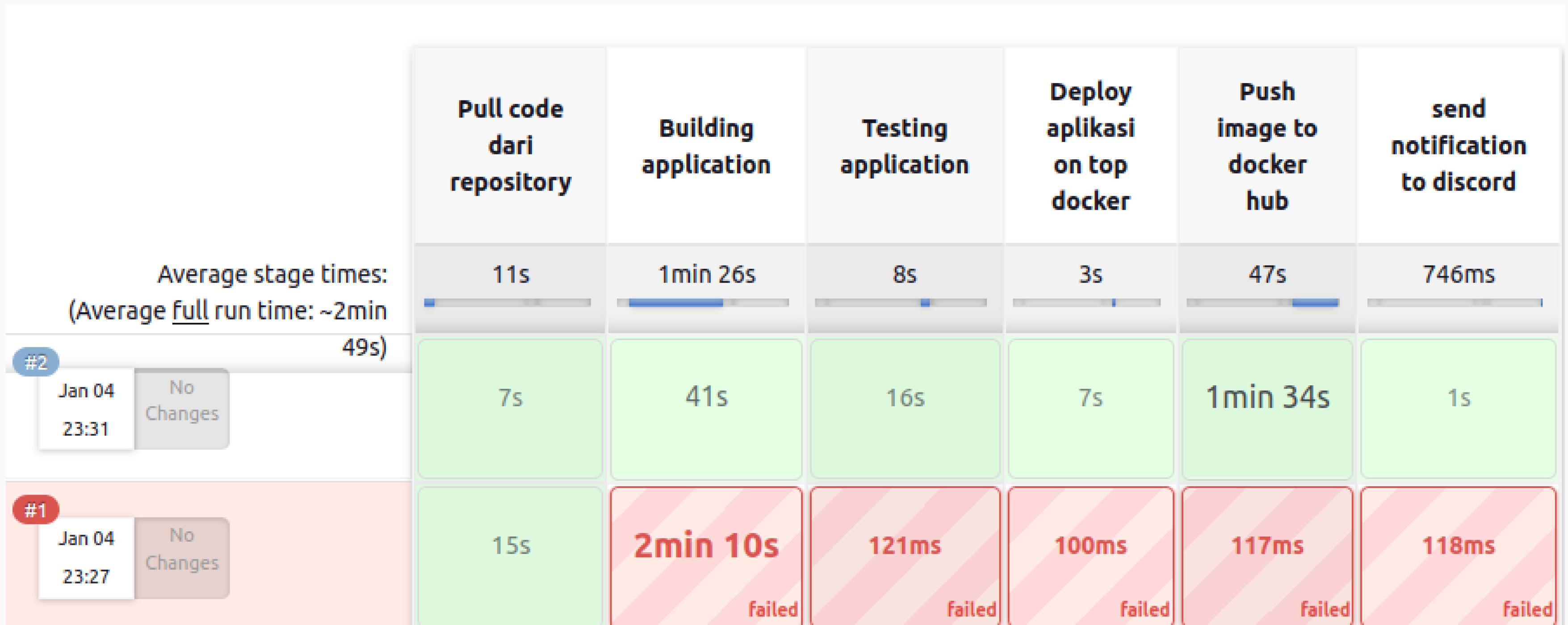
```
1 ▾ pipeline {
2   agent any
3   environment{
4     credential = 'cicd'
5     server = 'team1@103.150.93.34'
6     directory = '/home/team1/wayshub-backend'
7     branch = 'main'
8     service = 'backend'
9     image = 'nurliza/backend'
10    }
11   stages {
12     stage('Pull code dari repository'){
13       steps {
14         sshagent([credential]) {
15           sh '''ssh -o StrictHostKeyChecking=no ${server} << EOF
16             docker compose down
17             cd ${directory}
18             git pull origin ${branch}
19             exit
20           EOF
21         }
22       }
23     }
24   }
25 }
```

9

# Dokumentasi building, testing, dan deploy frontend

	<b>Pull code dari repository</b>	<b>Building application</b>	<b>Testing application</b>	<b>Deploy aplikasi on top docker</b>	<b>Push Image to docker hub</b>	<b>send notification to discord</b>
	Average stage times: (Average <u>full run time</u> : ~2min)	13s	47s	10s	7s	46s
#5	10s)					
#5	Jan 04 23:17	No Changes	14s	4s	15s	9s
#4	Jan 04 23:13	No Changes	19s	9s	14s	7s
#3	Jan 04 10:50	No Changes	8s	9s	14s	21s
					3min 29s	1s

# Dokumentasi building, testing, dan deploy backend





# GitLab



2

# Pengertian GitLab

---

GitLab adalah repositori Git berbasis web yang menyediakan repositori terbuka dan pribadi secara gratis. GitLab adalah platform DevOps lengkap yang memungkinkan para profesional melakukan semua tugas dalam sebuah proyek –mulai dari perencanaan proyek dan manajemen kode sumber hingga pemantauan dan keamanan. Selain itu, GitLab memungkinkan tim untuk berkolaborasi dan membangun perangkat lunak yang lebih baik.



3

# Building CI/CD GitLab Pipeline

---

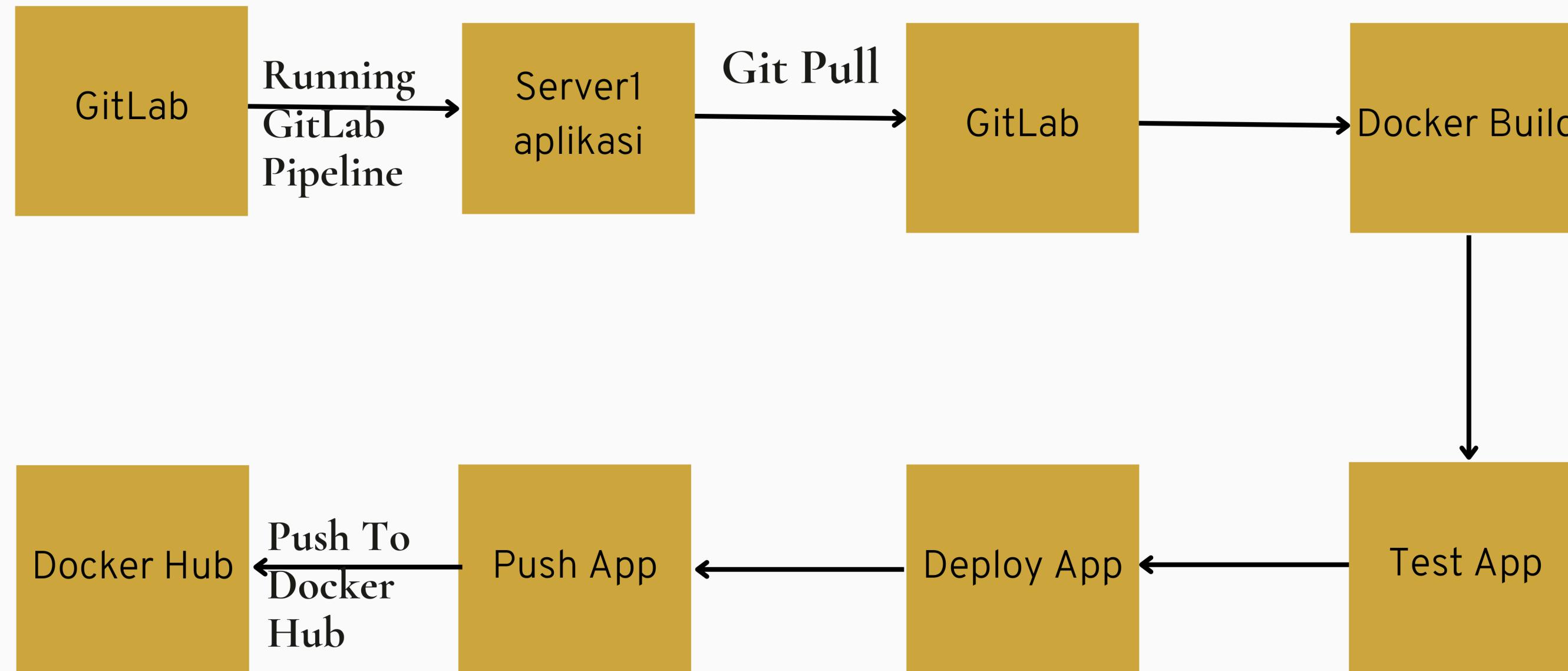
Membangun aplikasi menggunakan GitLab

Pada bagian ini aplikasi akan berjalan menggunakan GitLab pipeline yang akan berjalan pada docker di server.



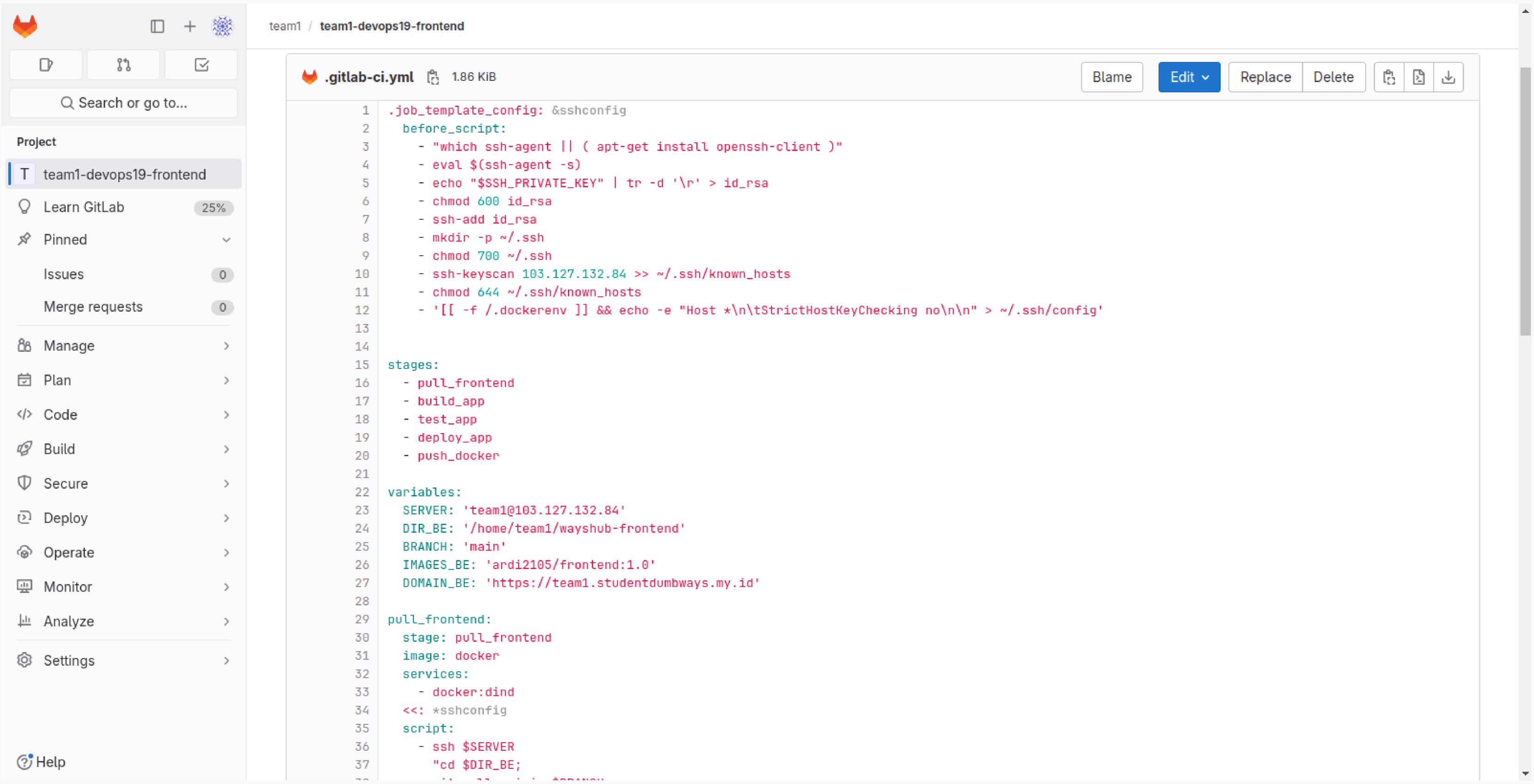
4

# Building CI/CD GitLab Pipeline



5

# Dokumentasi GitLab Script Frontend

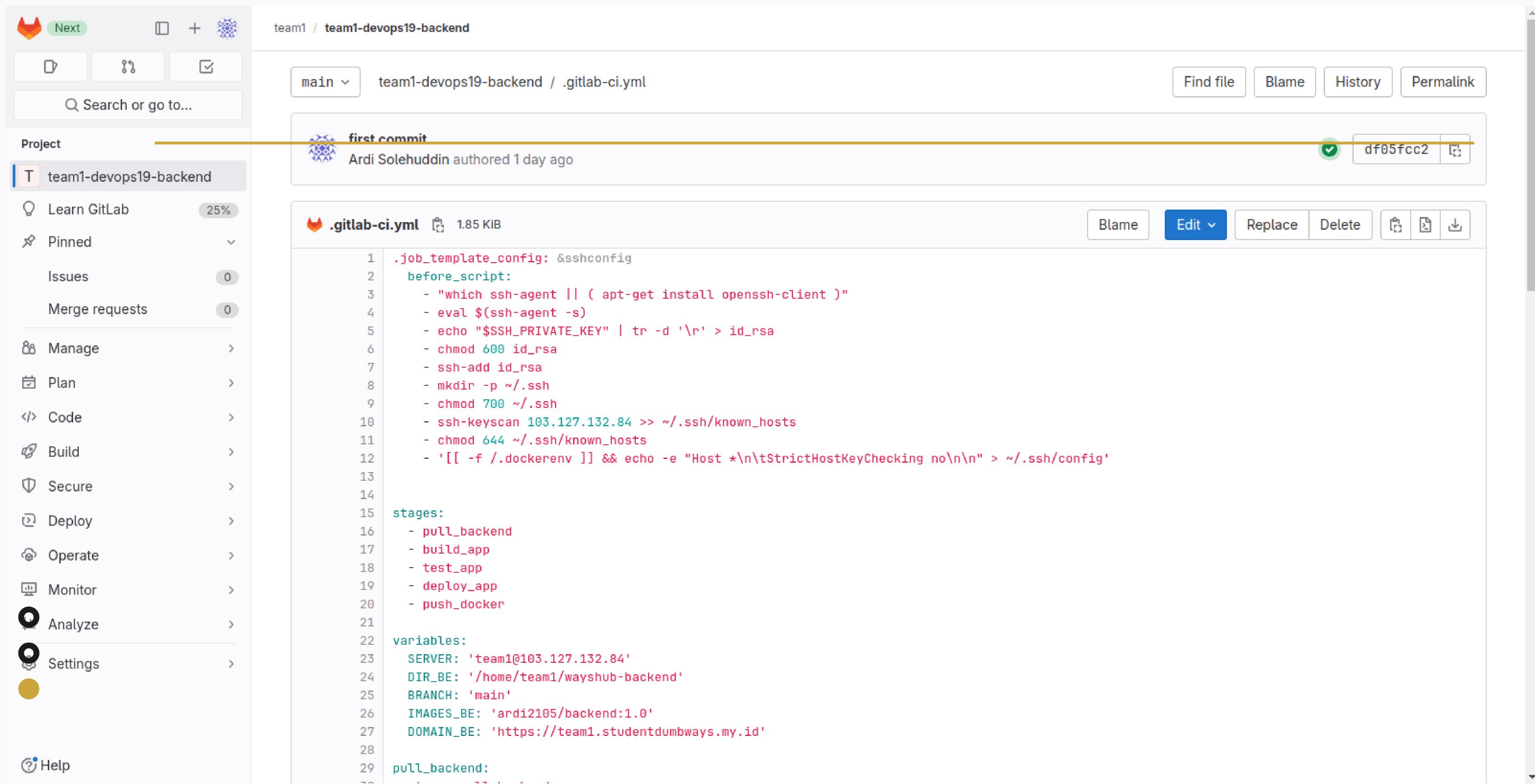


The screenshot shows a GitLab project interface for 'team1 / team1-devops19-frontend'. The left sidebar contains project navigation links such as Learn GitLab (25%), Pinned, Issues (0), Merge requests (0), Manage, Plan, Code, Build, Secure, Deploy, Operate, Monitor, Analyze, Settings, and Help. The main area displays the contents of the '.gitlab-ci.yml' file.

```
1 .job_template_config: &sshconfig
2   before_script:
3     - "which ssh-agent || ( apt-get install openssh-client )"
4     - eval $(ssh-agent -s)
5     - echo "$SSH_PRIVATE_KEY" | tr -d '\r' > id_rsa
6     - chmod 600 id_rsa
7     - ssh-add id_rsa
8     - mkdir -p ~/.ssh
9     - chmod 700 ~/.ssh
10    - ssh-keyscan 103.127.132.84 >> ~/.ssh/known_hosts
11    - chmod 644 ~/.ssh/known_hosts
12    - "[[ -f /.dockerenv ]] && echo -e \"Host *\n\tStrictHostKeyChecking no\n\" > ~/.ssh/config"
13
14
15 stages:
16   - pull_frontend
17   - build_app
18   - test_app
19   - deploy_app
20   - push_docker
21
22 variables:
23   SERVER: 'team1@103.127.132.84'
24   DIR_BE: '/home/team1/wayshub-frontend'
25   BRANCH: 'main'
26   IMAGES_BE: 'ardi2105/frontend:1.0'
27   DOMAIN_BE: 'https://team1.studentdumbways.my.id'
28
29 pull_frontend:
30   stage: pull_frontend
31   image: docker
32   services:
33     - docker:dind
34   <<: *sshconfig
35   script:
36     - ssh $SERVER
37       "cd $DIR_BE;
38         .." ..
```

## 6

# Dokumentasi GitLab Script Backend



The screenshot shows a GitLab project interface for 'team1 / team1-devops19-backend'. The left sidebar contains navigation links like Learn GitLab, Pinned, Issues, Merge requests, Manage, Plan, Code, Build, Secure, Deploy, Operate, Monitor, Analyze, Settings, and Help. The main area displays a commit titled 'first commit' by Ardi Solehuddin from 1 day ago. The commit hash is df05fcc2. Below the commit, the '.gitlab-ci.yml' file is shown with its content:

```
1 .job_template_config: &sshconfig
2   before_script:
3     - "which ssh-agent || ( apt-get install openssh-client )"
4     - eval $(ssh-agent -s)
5     - echo "$SSH_PRIVATE_KEY" | tr -d '\r' > id_rsa
6     - chmod 600 id_rsa
7     - ssh-add id_rsa
8     - mkdir -p ~/.ssh
9     - chmod 700 ~/.ssh
10    - ssh-keyscan 103.127.132.84 >> ~/.ssh/known_hosts
11    - chmod 644 ~/.ssh/known_hosts
12    - '[[ -f /.dockerenv ]] && echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config'
13
14
15 stages:
16   - pull_backend
17   - build_app
18   - test_app
19   - deploy_app
20   - push_docker
21
22 variables:
23   SERVER: 'team1@103.127.132.84'
24   DIR_BE: '/home/team1/wayshub-backend'
25   BRANCH: 'main'
26   IMAGES_BE: 'ardi2105/backend:1.0'
27   DOMAIN_BE: 'https://team1.studentdumbways.my.id'
28
29 pull_backend:
30   . . .
```

# Dokumentasi Pulling, Building, Testing, Deploy dan Push Image Frontend

The screenshot shows the GitLab interface for a project named "team1-devops19-frontend". The pipeline titled "Update .gitlab-ci.yml file" has completed successfully. The pipeline consists of five stages: "pull\_frontend", "build\_app", "test\_app", "deploy\_app", and "push\_docker", connected sequentially. Each stage contains a green checkmark icon indicating success. The "Pipeline" tab is selected, showing 5 Jobs, 5.94 duration, and 5 minutes 56 seconds queued time.

team1 / team1-devops19-frontend / Pipelines / #1129890185

### Update .gitlab-ci.yml file

Passed Ardi Solehuddin created pipeline for commit 121276f2 finished 24 minutes ago

For main

latest 5 Jobs 5.94 5 minutes 56 seconds, queued for 0 seconds

Pipeline Needs Jobs 5 Tests 0

```
graph LR; pull_frontend[pull_frontend] --> build_app[build_app]; build_app --> test_app[test_app]; test_app --> deploy_app[deploy_app]; deploy_app --> push_docker[push_docker]
```

**Project**

- team1-devops19-frontend
- Learn GitLab 25%
- Pinned
- Issues 0
- Merge requests 0
- Manage
- Plan
- Code
- Build
- Pipelines**
- Jobs
- Pipeline editor
- Pipeline schedules
- Artifacts
- Secure
- Deploy
- Operate
- Help

# Dokumentasi Pulling, Building, Testing, Deploy dan Push Image Backend

The screenshot shows the GitLab interface for a project named "team1-devops19-backend". The pipeline titled "first commit" has completed successfully. The pipeline consists of five stages: "pull\_backend", "build\_app", "test\_app", "deploy\_app", and "push\_docker". Each stage is represented by a box containing a green checkmark icon and a circular progress bar.

```
graph LR; pull_backend[pull_backend] --> build_app[build_app]; build_app --> test_app[test_app]; test_app --> deploy_app[deploy_app]; deploy_app --> push_docker[push_docker]
```

**Project:** team1-devops19-backend

**Pipeline:** first commit

**Status:** Passed (Ardi Solehuddin created pipeline for commit df05fcc2, finished 1 day ago)

**For main:** latest (5 Jobs, 6.37 minutes, 6 minutes 21 seconds, queued for 0 seconds)

**Jobs:** Pipeline, Needs, Jobs 5, Tests 0

**Stages:**

- pull\_backend (Passed)
- build\_app (Passed)
- test\_app (Passed)
- deploy\_app (Passed)
- push\_docker (Passed)

**Navigation:**

- Project: team1-devops19-backend
- Learn GitLab (25%)
- Pinned
- Issues (0)
- Merge requests (0)
- Manage
- Plan
- Code
- Build
- Pipelines (selected)
- Jobs
- Pipeline editor
- Pipeline schedules
- Artifacts
- Secure
- Deploy
- Operate
- Help

# Dokumentasi Docker Compose

```
team1@appserver:~$ docker compose ps -a
          NAME      IMAGE        COMMAND
back-apps  ardi2105/backend:1.0  "docker-entrypoint.sh npm start"
front-apps  ardi2105/frontend:1.0 "docker-entrypoint.sh npm start"
mysql      mysql:latest        "docker-entrypoint.sh mysqld"
nginx      nginx:latest        "/docker-entrypoint.sh nginx -g 'daemon off;'" 
                                SERVICE    CREATED     STATUS      PORTS
                                backend    26 minutes ago Up 26 minutes  0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
                                frontend   26 minutes ago Up 26 minutes  0.0.0.0:3000->3000/tcp, :::3000->3000/tcp
                                database   26 minutes ago Up 26 minutes  0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp
                                webserver  26 minutes ago Up 26 minutes  0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp
team1@appserver:~$
```



# Terima Kasih

---

