Tugas Hasil Uji Coba

Sistem Paralel dan Terdistribusi A

Docker Compose

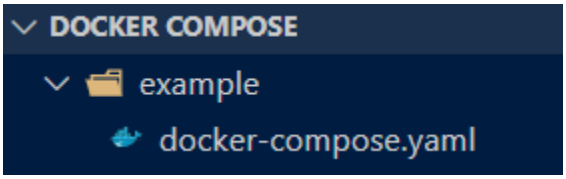
Disusun Oleh :

Zakaria Fattawari 11231092

10 November 2025

DOCKER COMPOSE

Apa itu Docker Compose?

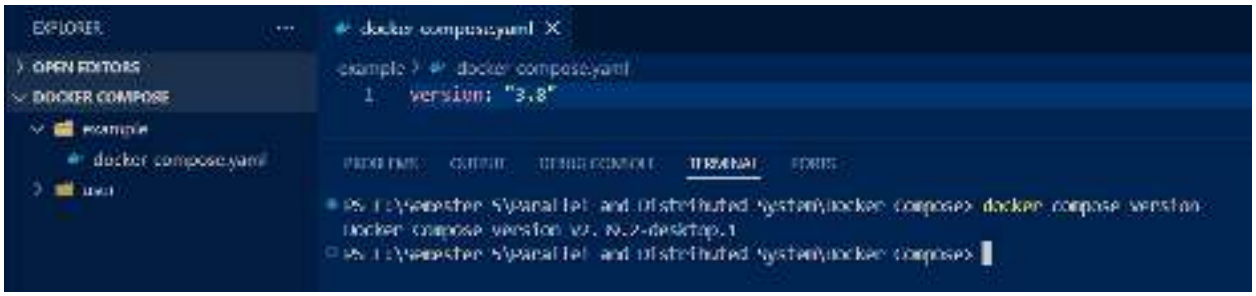


Docker Compose: Version

CONFIGURATION FILE

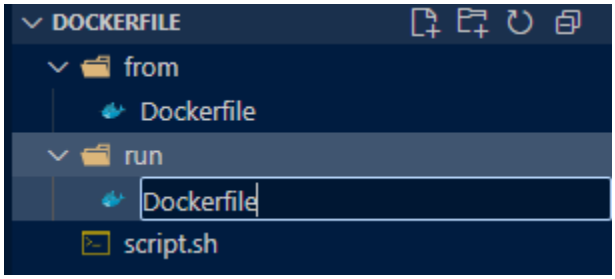
Docker Compose: Versi Konfigurasi

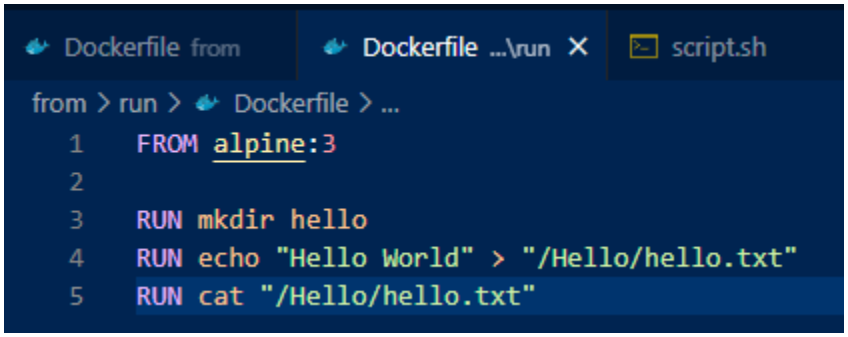
Docker Compose: Docker Compose File



YAML

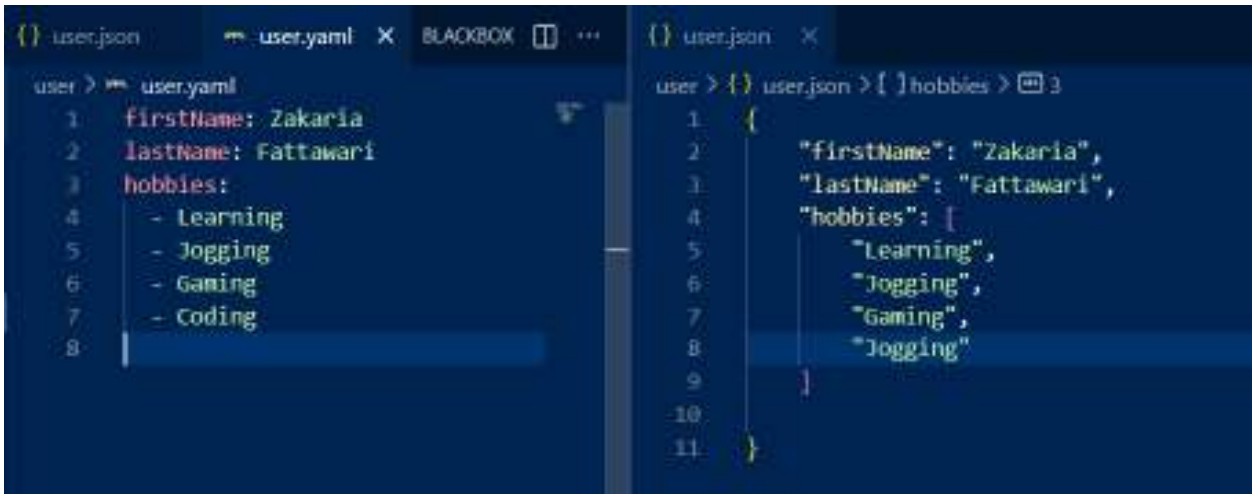
Pengenalan YAML





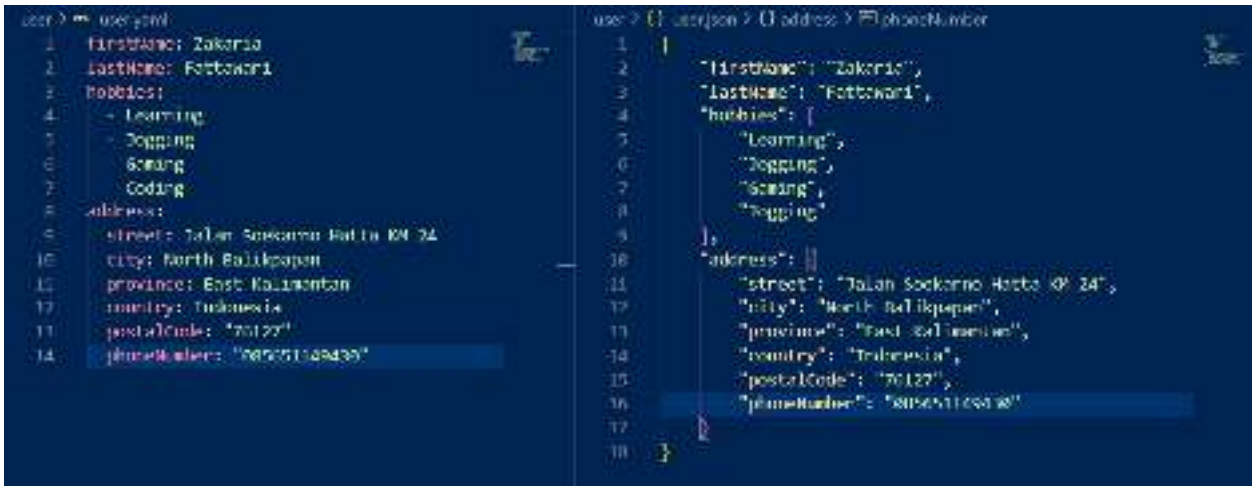
Docker Compose: YAML Attribute

Docker Compose: YAML Array

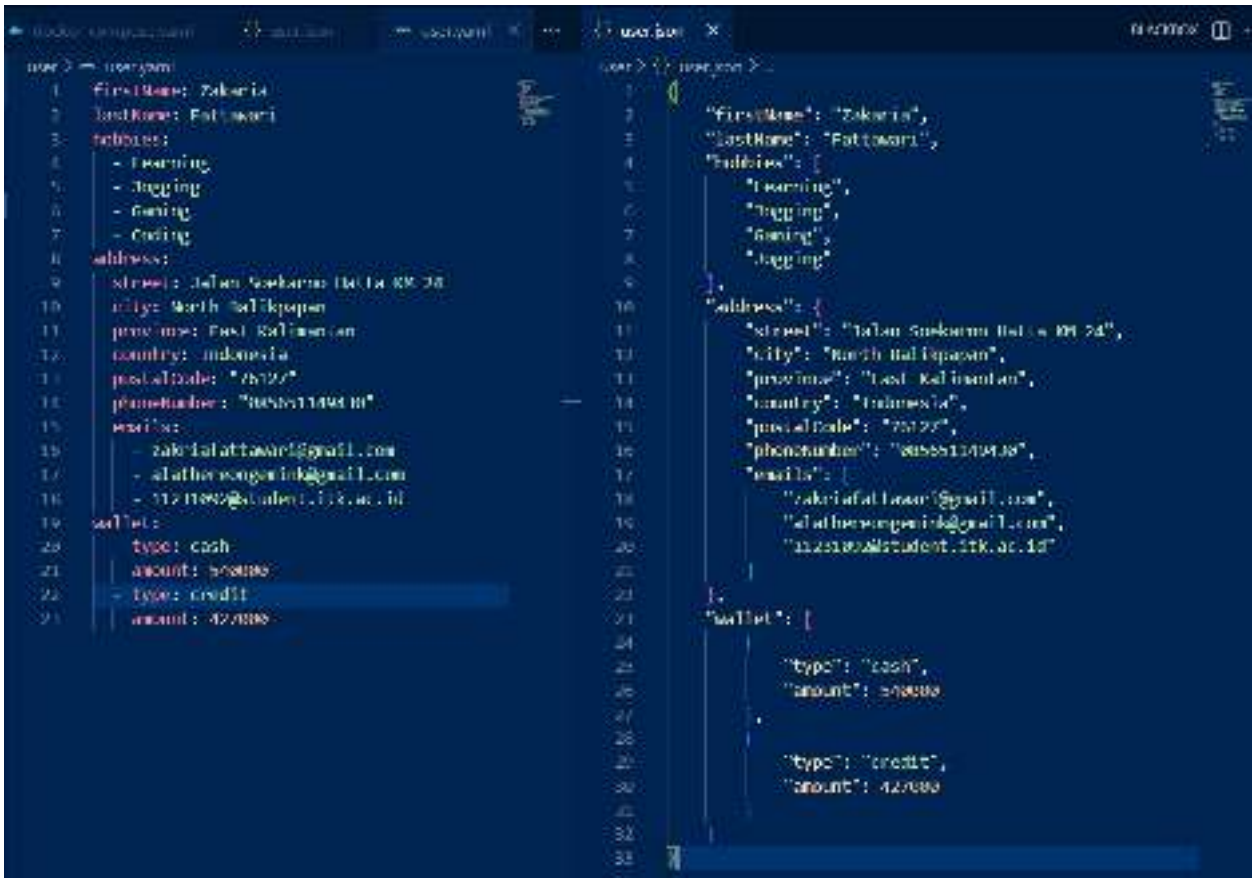


=

Docker Compose: YAML Nested Object

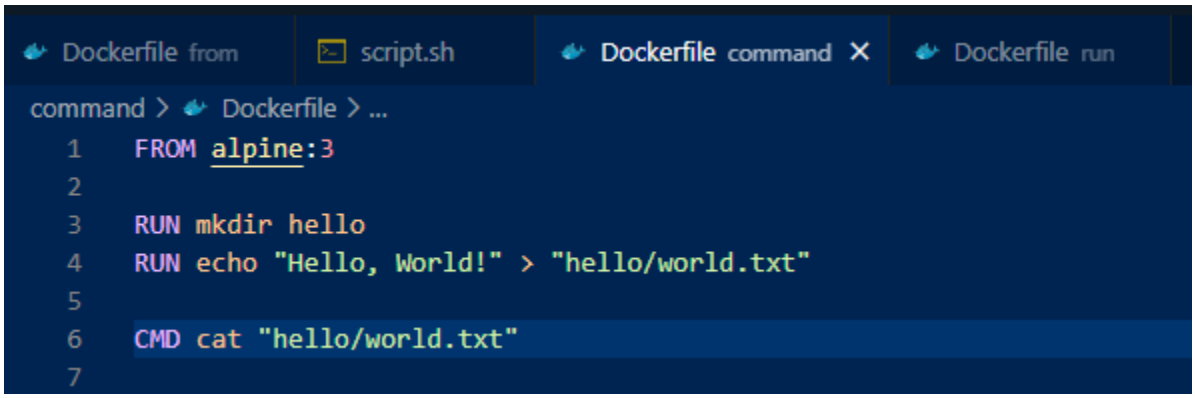


Docker Compose: YAML Array Nested Object

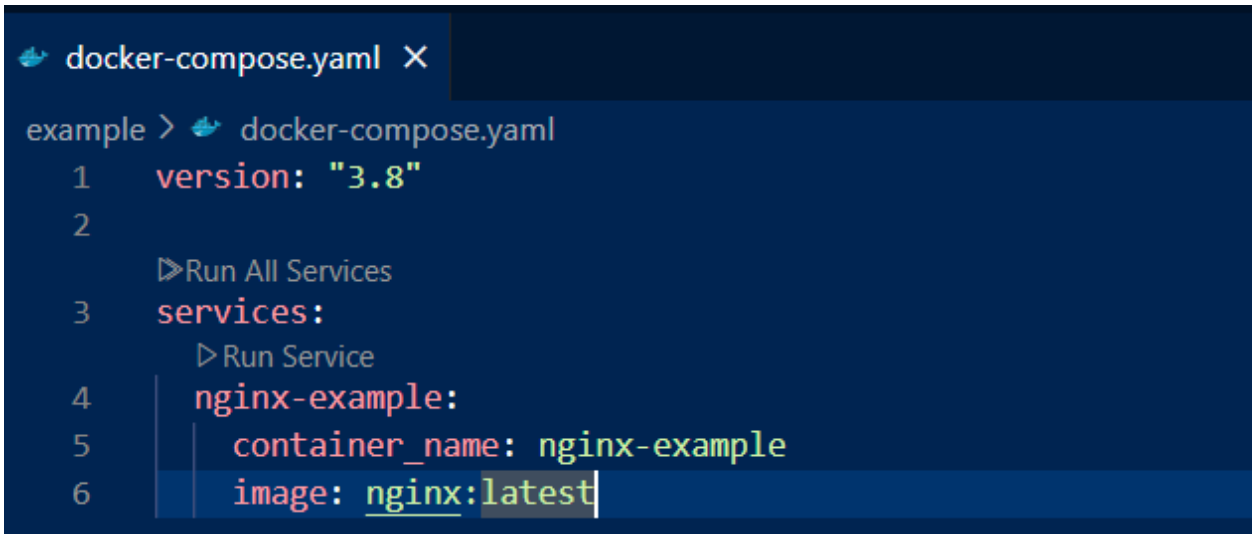


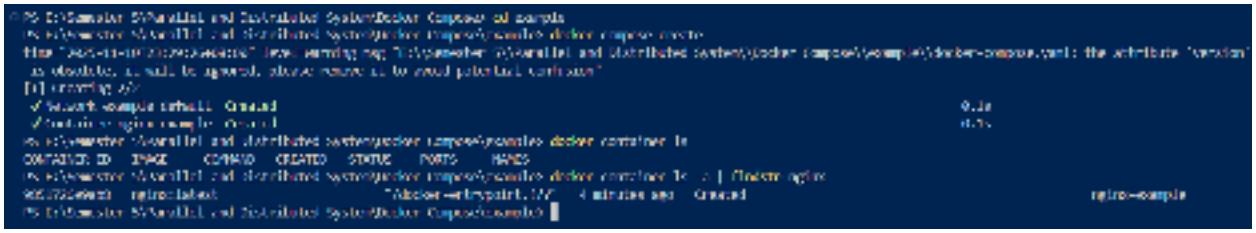
MEMBUAT CONTAINER

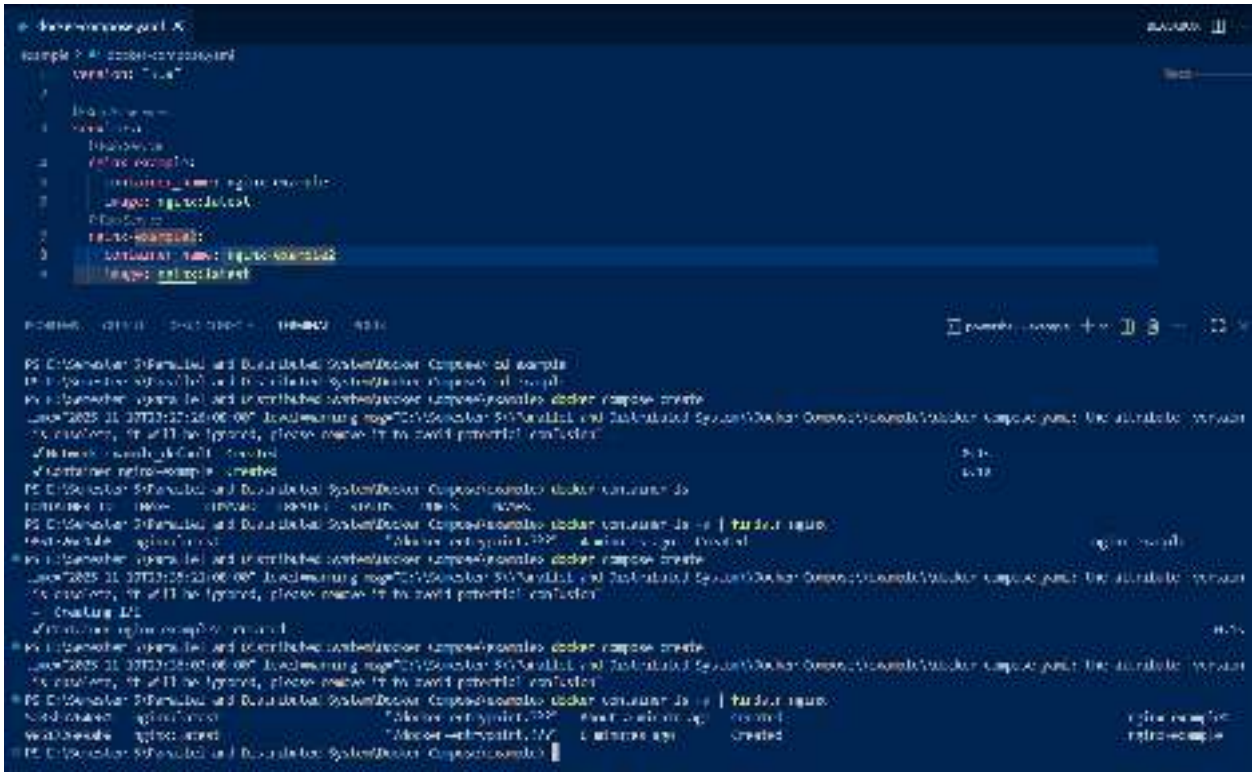
Membuat Konfigurasi Kontainer



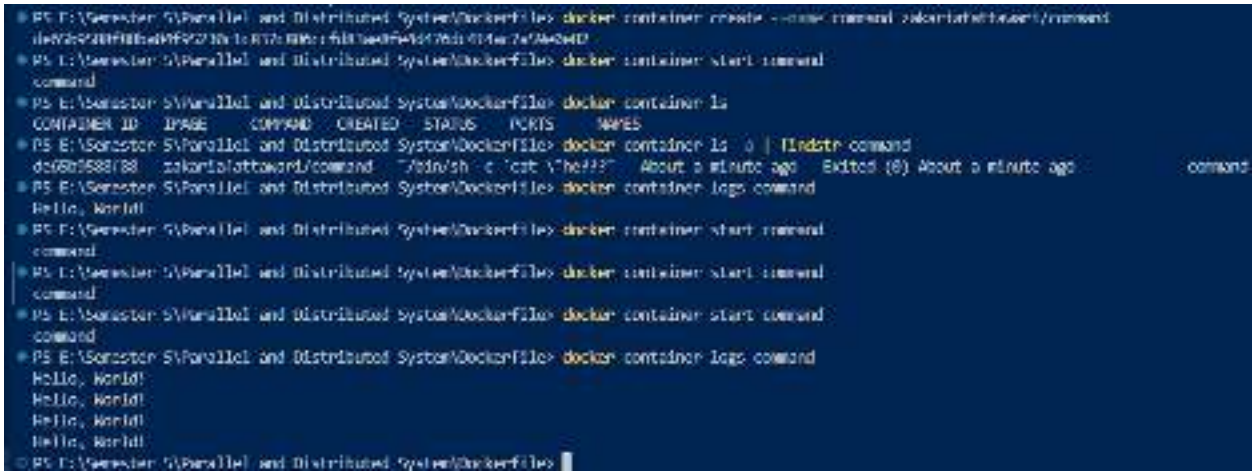
Docker Compose: Membuat Container







Docker Container: COMMAND



MENJALANKAN CONTAINER

Pemgenalan Menjalankan Container

```
PS E:\Semester 5\Parallel and Distributed System\Docker Compose> cd example
PS E:\Semester 5\Parallel and Distributed System\Docker Compose\example> docker compose start
time="2025-11-10T25:57:45+08:00" level=warning msg="E:\\Semester 5\\Parallel and Distributed System\\Docker Compose\\example\\docker-compose.yaml: the at
tribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 2/2
 ✓ Container nginx-example2  Started                                                                                                              0.4s
 ✓ Container nginx-example   Started                                                                                                              0.4s
PS E:\Semester 5\Parallel and Distributed System\Docker Compose\example> docker container ls
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS     NAMES
5a83ba266e60   nginx:latest   "/docker-entrypoint.…"   2 minutes ago   Up 5 seconds   80/tcp    nginx-example2
985172ee9ab8   nginx:latest   "/docker-entrypoint.…"   8 minutes ago   Up 5 seconds   80/tcp    nginx-example
PS E:\Semester 5\Parallel and Distributed System\Docker Compose\example>
```

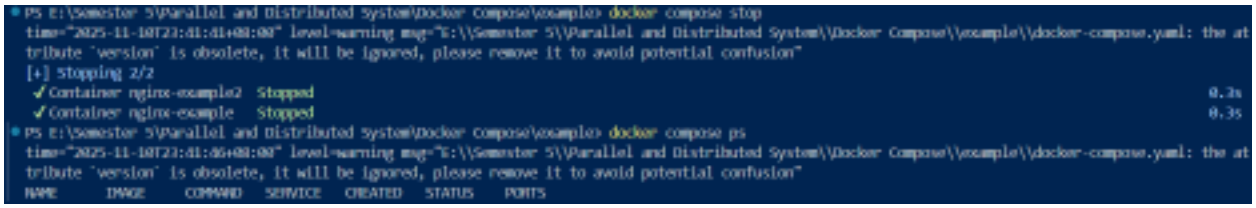
Dockerfile from | script.sh | Dockerfile command | Dockerfile label | Dockerfile run

label > Dockerfile > ...

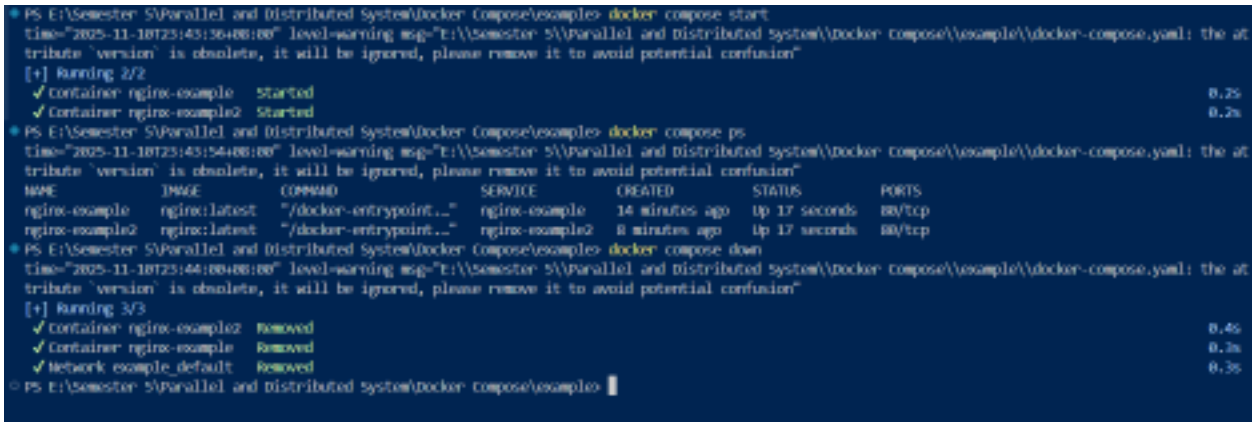
```
1  FROM alpine:3
2
3  LABEL author="Zakaria Fattawari"
4  LABEL company="A206" website="https://www.instagram.com/alathereonn_"
5
6  RUN mkdir hello
7  RUN echo "Hello, World!" > "hello/world.txt"
8  CMD cat "hello/world.txt"
9
10
```

Docker Compose: Melihat Container

Docker Compose: Menghentikan Container

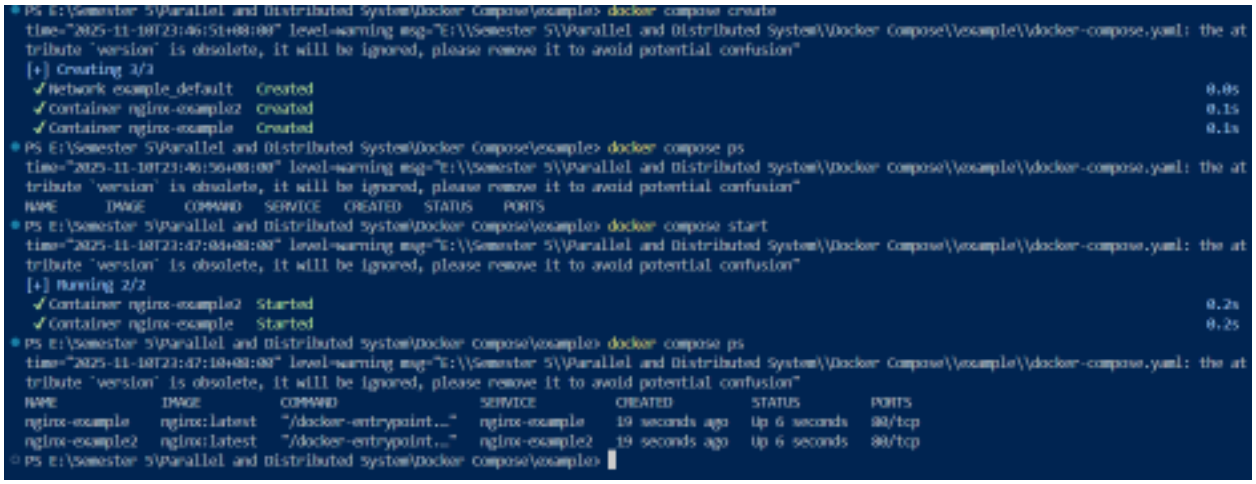


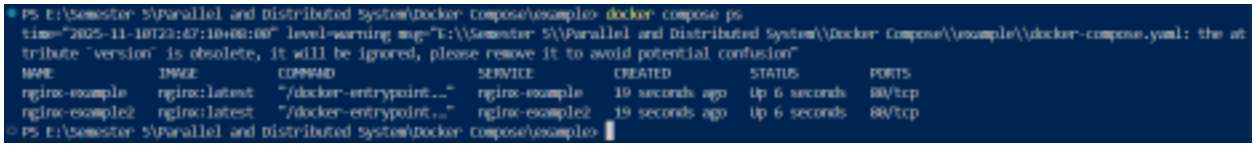
Docker Compose: Menghapus Container



PROJECT NAME

Pengenalan Project Name



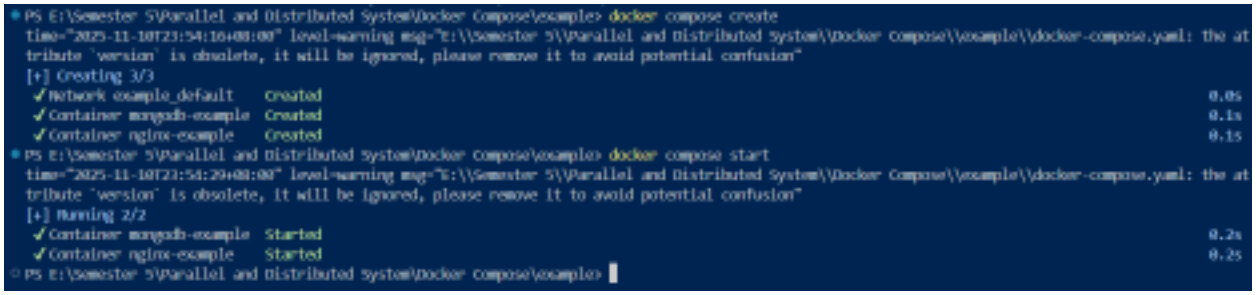


SERVICES

Pengenalan SERVICES

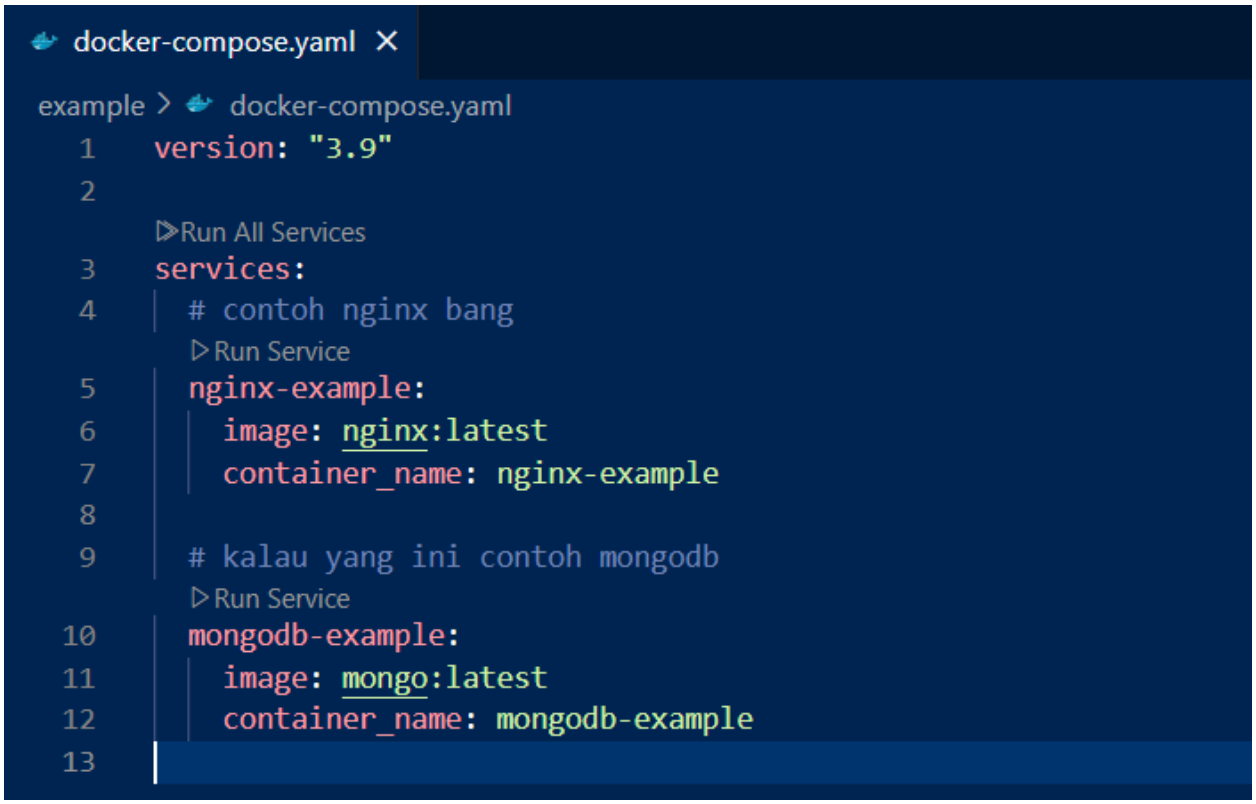
```yaml
version: "3.9"

services:
  nginx-example:
    image: nginx:latest
    container_name: nginx-example
  mongodb-example:
    image: mongo:latest
    container_name: mongodb-example
```

Docker Compose: SERVICES



KOMENTAR

Pengenalan Komentar



PORT

Pengenalan PORT

docker-compose.yaml

example > docker-compose.yaml

```yaml
version: "3.9"

services:
  # contoh port dengan long syntax
  nginx-port1:
    image: nginx:latest
    container_name: nginx-port1
    ports:
      - protocol: tcp
        published: 8080
        target: 80
  # contoh port dengan short syntax
  nginx-port2:
    image: nginx:latest
    container_name: nginx-port2
    ports:
      - 8081:80

```

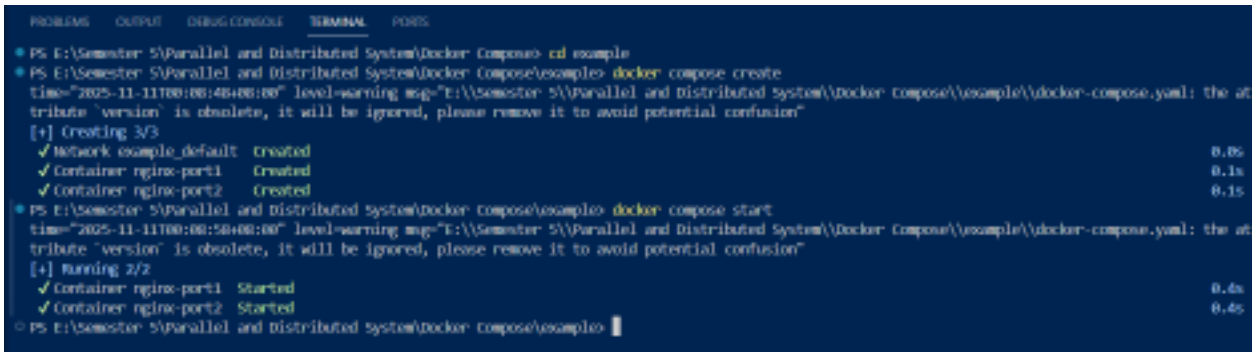
Docker Compose: Short Syntax

```
12  # contoh port dengan short syntax
    ▷ Run Service
13  nginx-port2:
14    image: nginx:latest
15    container_name: nginx-port2
16    ports:
17      - 8081:80
18
```

Docker Compose: Long Syntax

```
4   # contoh port dengan long syntax
    ▷ Run Service
5   nginx-port1:
6     image: nginx:latest
7     container_name: nginx-port1
8     ports:
9       - protocol: tcp
10        published: 8080
11        target: 80
```

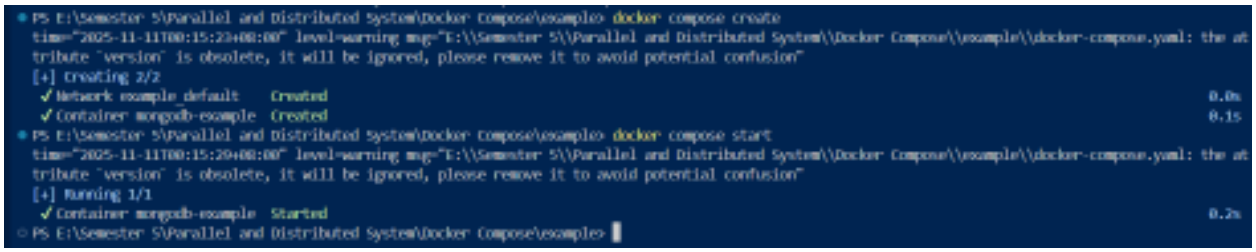
Docker Compose: PORT (Menjalankan Container)

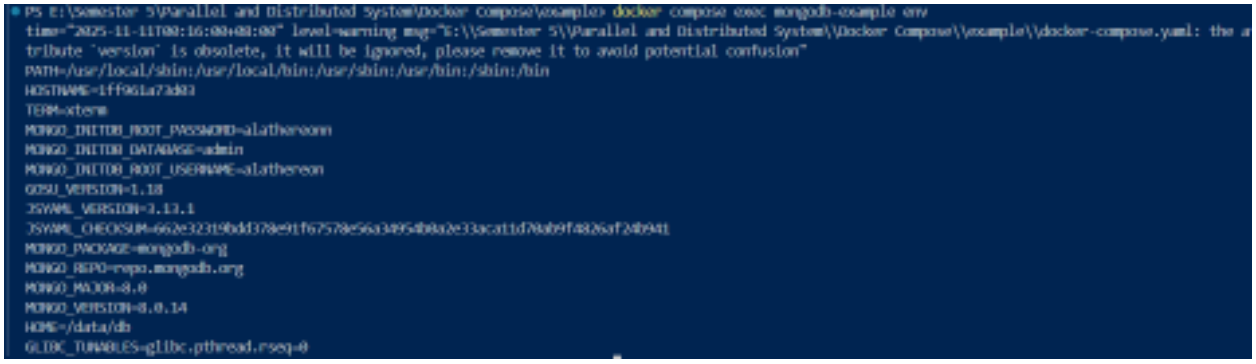


ENVIRONMENT VARIABLE

Pengenalan ENVIRONMENT VARIABLE

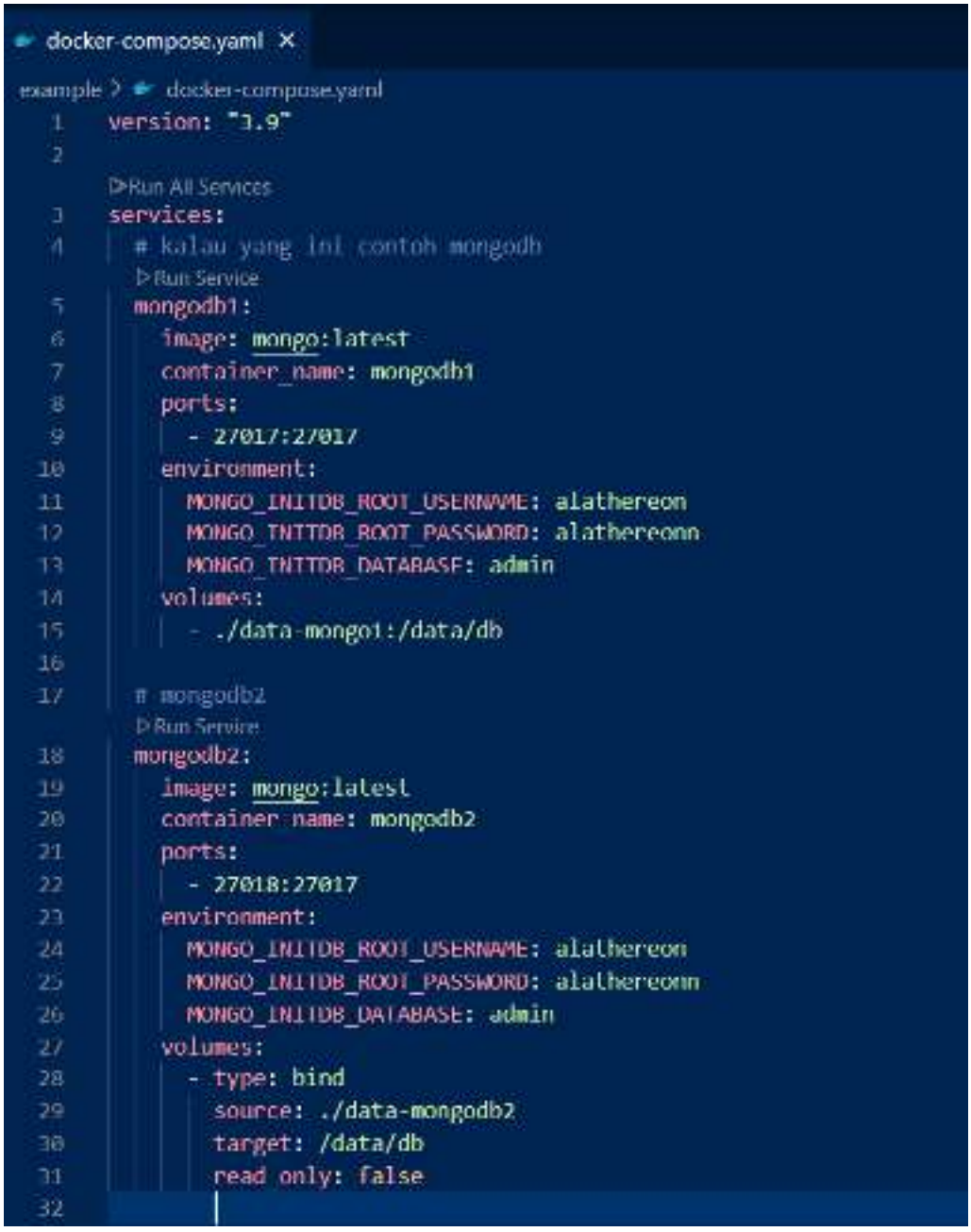
Docker Compose: ENVIRONMENT VARIABLE





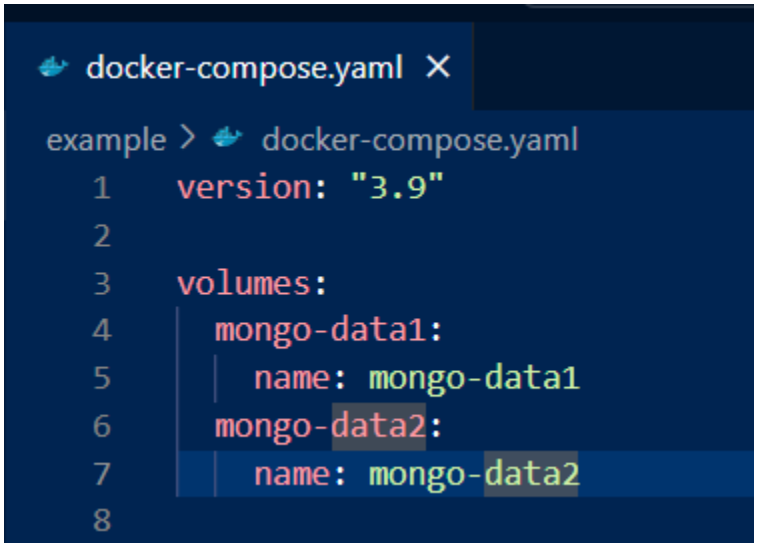
BIND MOUNT

Pengenalan BIND MOUNT



VOLUME

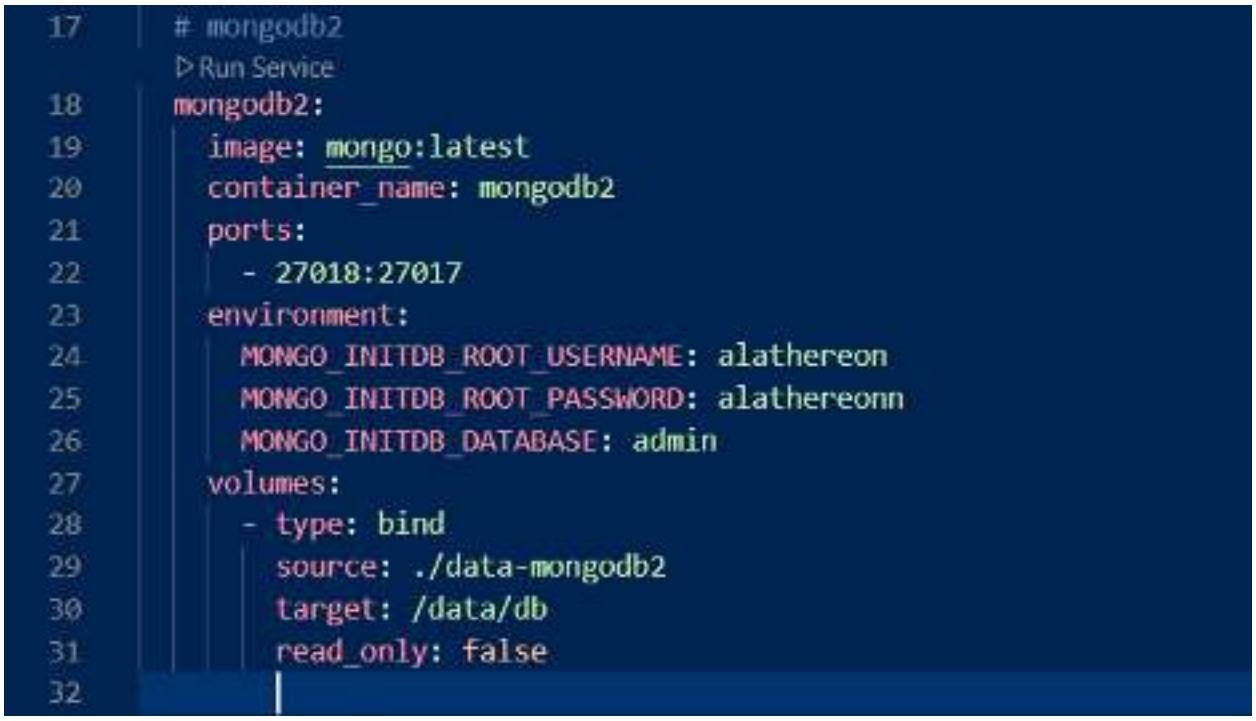
Pengenalan VOLUME



Docker Compose: SHORT SYNTAX VOLUME

```yaml
# kalau yang ini contoh mongodb
▷ Run Service
mongodb1:
  image: mongo:latest
  container_name: mongodb1
  ports:
    - 27017:27017
  environment:
    MONGO_INITDB_ROOT_USERNAME: alathereon
    MONGO_INITDB_ROOT_PASSWORD: alathereonn
    MONGO_INITDB_DATABASE: admin
  volumes:
    - ./data-mongo1:/data/db
```

Docker Compose: LONG SYNTAX VOLUME

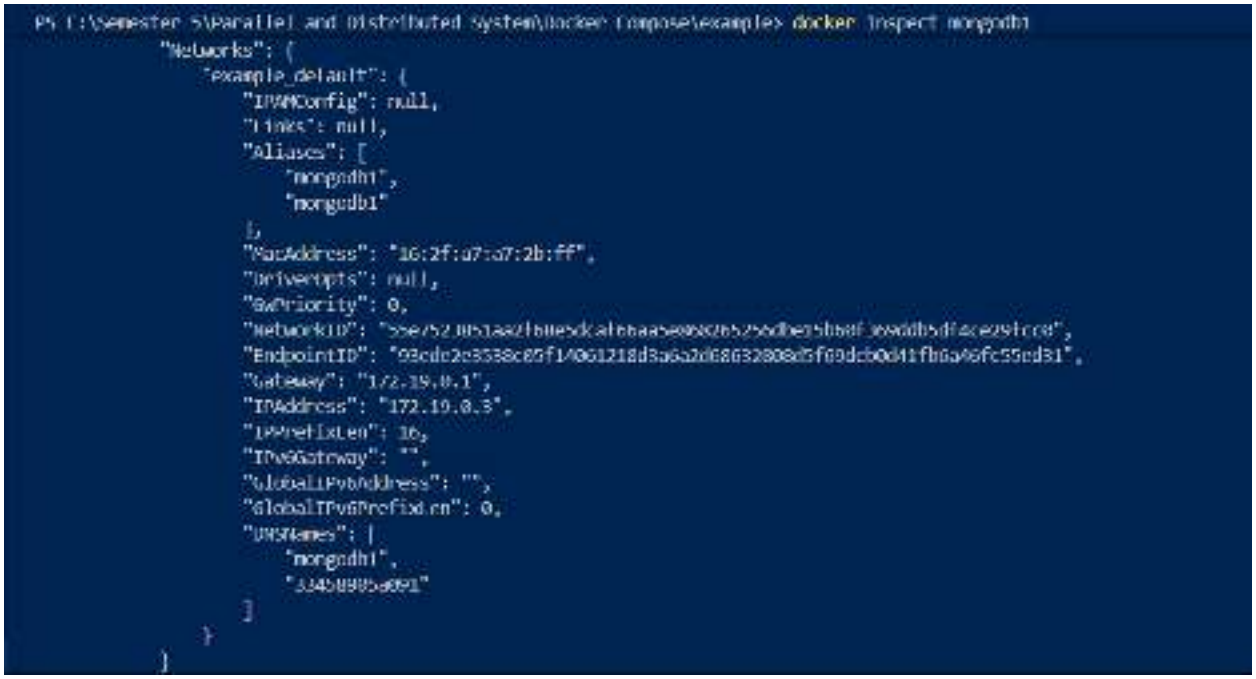


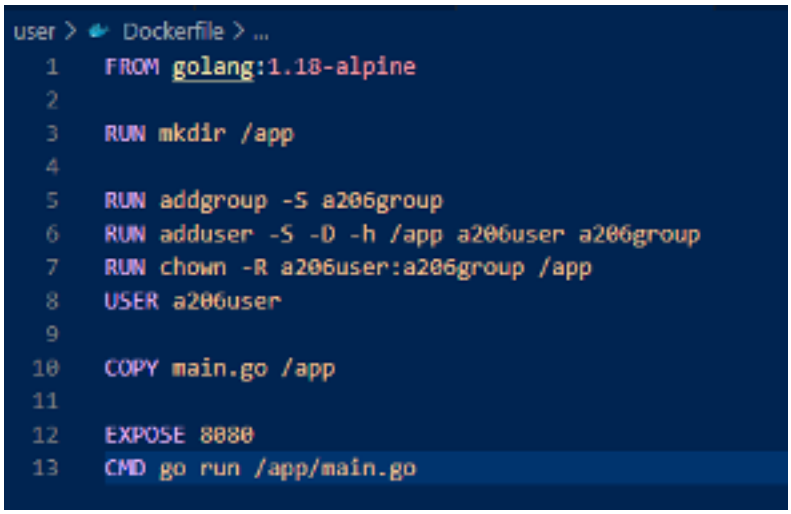
```
17  # mongodb2
    ▷ Run Service
18  mongodb2:
19    image: mongo:latest
20    container_name: mongodb2
21    ports:
22      - 27018:27017
23    environment:
24      MONGO_INITDB_ROOT_USERNAME: alathereon
25      MONGO_INITDB_ROOT_PASSWORD: alathereonn
26      MONGO_INITDB_DATABASE: admin
27    volumes:
28      - type: bind
29        source: ./data-mongodb2
30        target: /data/db
31        read_only: false
32
```

Docker Compose: Menghapus VOLUME

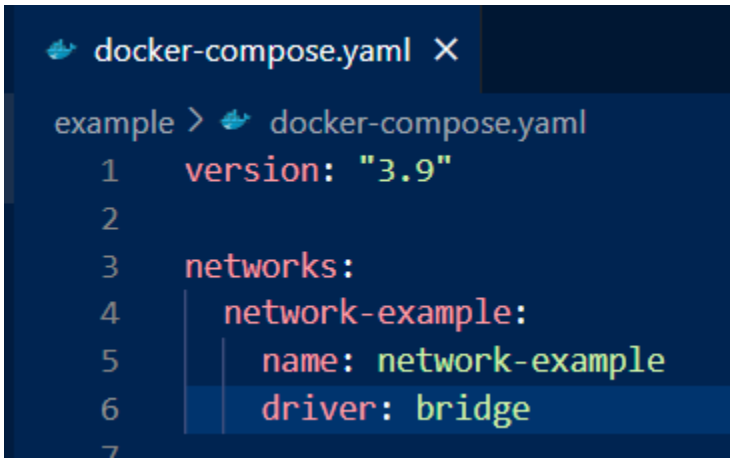
NETWORK

Pengenalan NETWORK

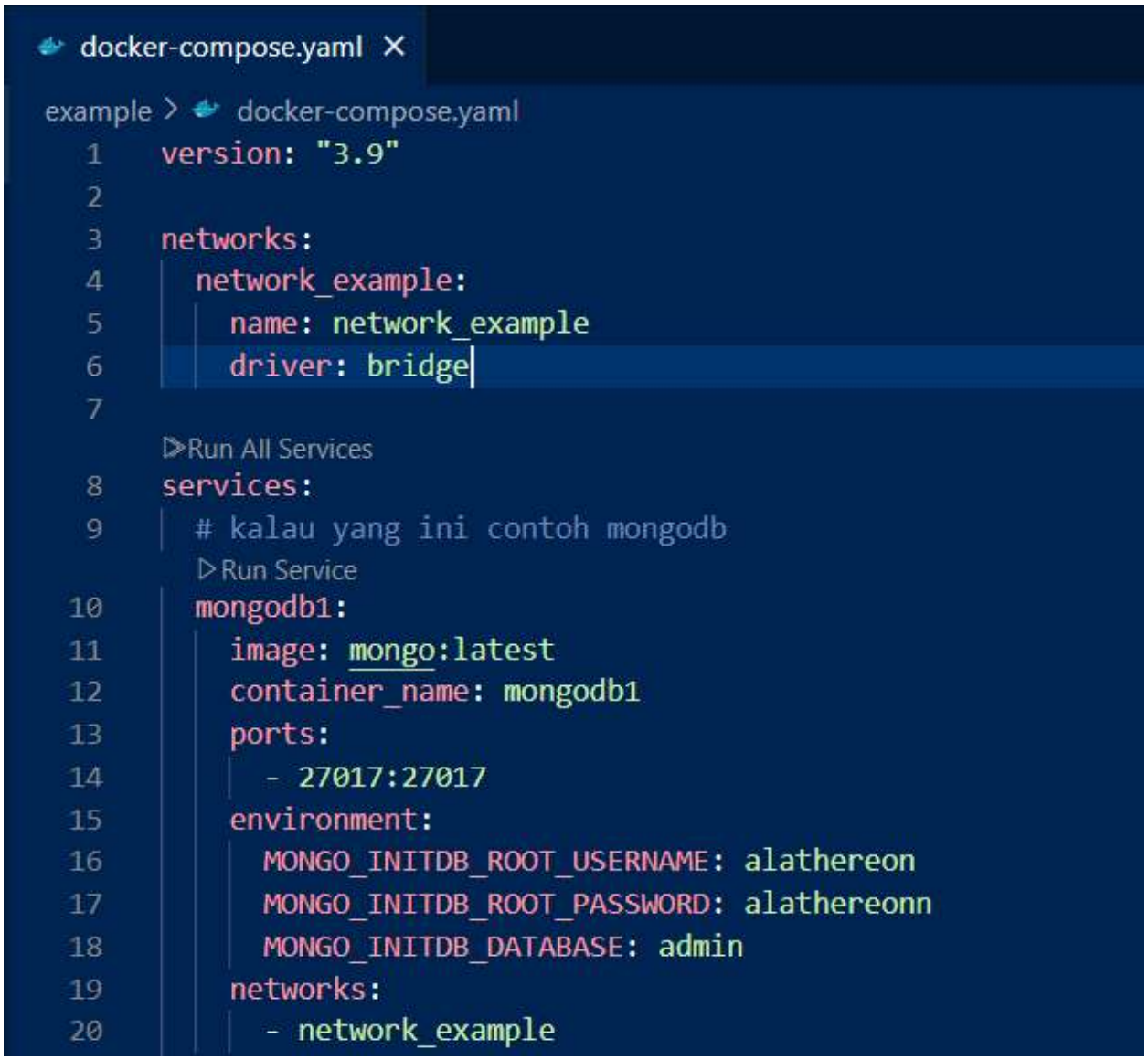




Docker Compose: Membuat NETWORK



Docker Compose: Menggunakan NETWORK



```
PS E:\Semester 5\Parallel and Distributed System\Docker compose\example> docker inspect mongodb1
            "Networks": {
                "network_example": {
                    "IPAMConfig": null,
                    "Links": null,
                    "Aliases": [
                        "mongodb1",
                        "mongodb1"
                    ],
                    "MacAddress": "ea:fc:a0:8d:f9:3a",
                    "DriverOpts": null,
                    "GwPriority": 0,
                    "NetworkID": "[illegible]",
                    "EndpointID": "[illegible]",
                    "Gateway": "172.19.0.1",
                    "IPAddress": "172.19.0.2",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "DNSNames": [
                        "mongodb1",
                        "[illegible]"
                    ]
                }
```

DEPENDS ON

Pengenalan DEPENDS ON

docker-compose.yaml

example > docker-compose.yaml

```yaml
version: "3.9"

networks:
  network_example:
    name: network_example
    driver: bridge

Run All Services
services:
  Run Service
  mongodb-express-example:
    image: mongo-express-latest
    container_name: mongodb-express-example
    depends_on:
      - mongodb-example
    ports:
      - 8081:8081
    envirnoment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: alathereon
      ME_CONFIG_MONGODB_ADMINPASSWORD: alathereonn
      ME_CONFIG_MONGODB_SERVER: mongodb-example
    networks:
    - network_example
```

RESTART

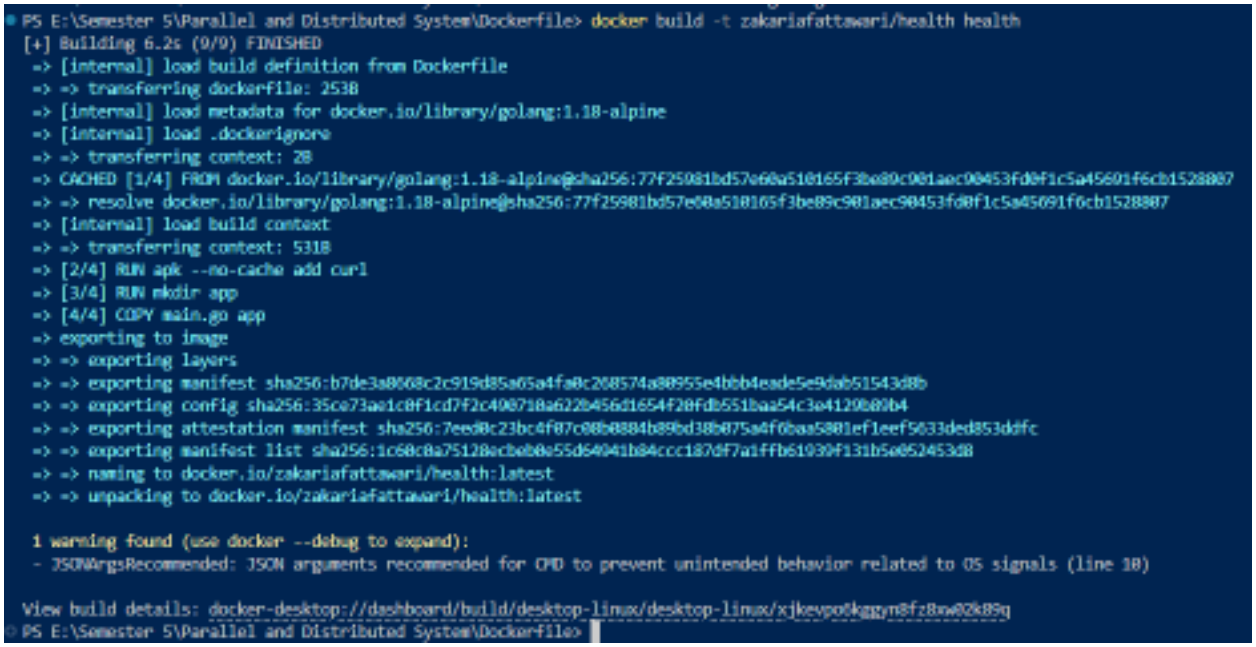
Pengenalan RESTART

docker-compose.yaml

example > docker-compose.yaml

```yaml
version: "3.9"

networks:
  network_example:
    name: network_example
    driver: bridge

▷Run All Services
services:
  ▷Run Service
  mongodb-express-example:
    image: mongo-express-latest
    container_name: mongodb-express-example
    restart: always
    depends_on:
      - mongodb-example
    ports:
      - 8081:8081
    envirnoment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: alathereon
      ME_CONFIG_MONGODB_ADMINPASSWORD: alathereonn
      ME_CONFIG_MONGODB_SERVER: mongodb-example
    networks:
    - network_example
```

Docker Compose: Monitor Docker Events

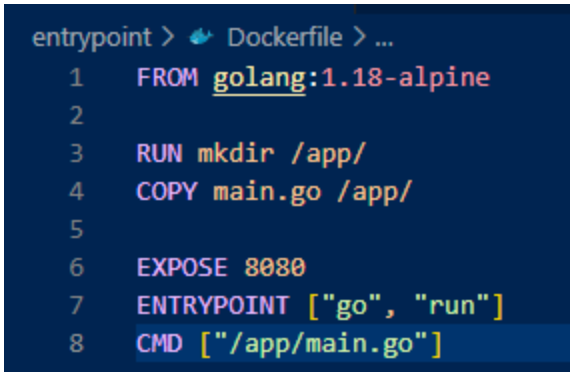


RESOURCE LIMIT

Pengenalan RESOURCE LIMIT

resource limit > docker-compose.yaml

```yaml
version: "3.9"

nginx-example:
  image: nginx:latest
  container_name: nginx-example
  ports:
    - "8080:80"
  deploy:
    resources:
      reservations:
        cpus: "0.25"
        memory: 50M
      limits:
        cpus: "0.5"
        memory: 100M
```

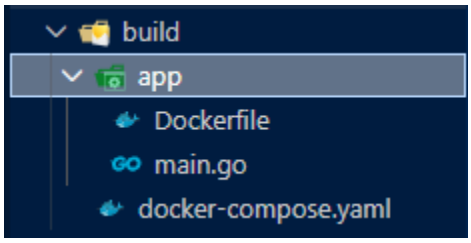

entrypoint > Dockerfile > ...
1 FROM golang:1.18-alpine
2
3 RUN mkdir /app/
4 COPY main.go /app/
5
6 EXPOSE 8080
7 ENTRYPOINT ["go", "run"]
8 CMD ["/app/main.go"]

Docker Compose: CONTAINER STATS

```
CONTAINER ID   NAME            CPU %     MEM USAGE / LIMIT   MEM %     NET I/O       BLOCK I/O       PIDS
d244126b3dd6   nginx-example   0.00%     17.4MiB / 100MiB    17.40%    1.7kB / 126B  13MB / 12.3kB   17
```

DOCKERFILE

Membuat Dockerfile



build > app > main.go

```go
package main

import (
    "fmt"
    "net/http"
    "os"
)

func main() {
    port := os.Getenv("APP_PORT")
    fmt.Println("Run app in port : " + port)
    http.HandleFunc("/", HelloServer)
    http.ListenAndServe(":"+port, nil)
}

func HelloServer(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello, World!")
}
```

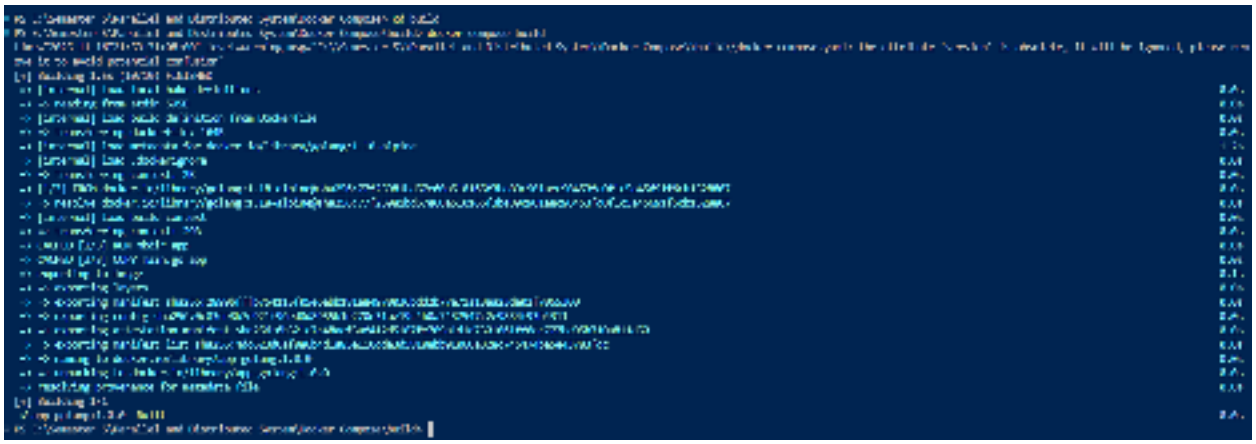
build > app > Dockerfile > FROM

```dockerfile
FROM golang:1.18-alpine

ENV APP_PORT=8080

RUN mkdir app
COPY main.go app

EXPOSE ${APP_PORT}

CMD go run app/main.go
```

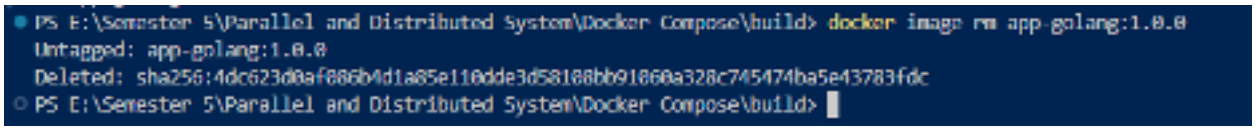
build > docker-compose.yaml

```yaml
version: "3.9"

Run All Services
services:
  Run Service
  app:
    container_name: app
    build:
      context: "./app"
      dockerfile: Dockerfile
    image: "app-golang:1.0.0"
    environment:
      - "APP_PORT=8080"
    ports:
      - "8080:8080"
```

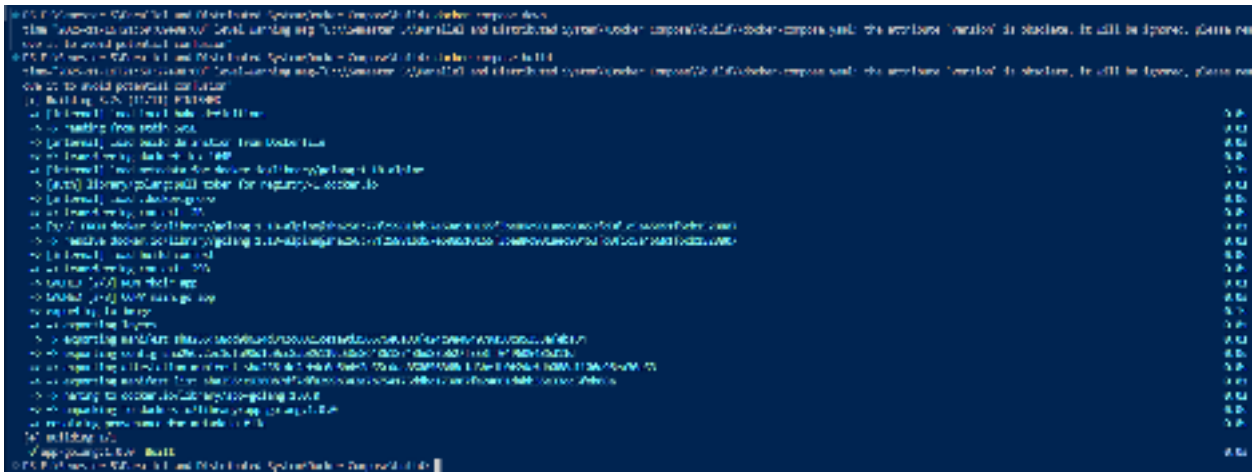
Docker Compose: BUILD DOCKEFILE



Docker Compose: Menghapus Image

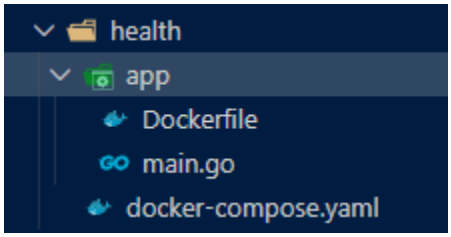


Docker Compose: BUILD Ulang



HEALTHCHECK

Pengenalan HEALTHCHECK



health > docker-compose.yaml

```yaml
version: "3.9"

Run All Services
services:
  Run Service
  app:
    container_name: app
    build:
      context: "./app"
      dockerfile: Dockerfile
    image: "app-golang:1.0.0"
    environment:
      - "APP_PORT=8080"
    ports:
      - "8080:8080"
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8080/health"]
      interval: 5s
      timeout: 5s
      retries: 3
      start_period: 3s
```

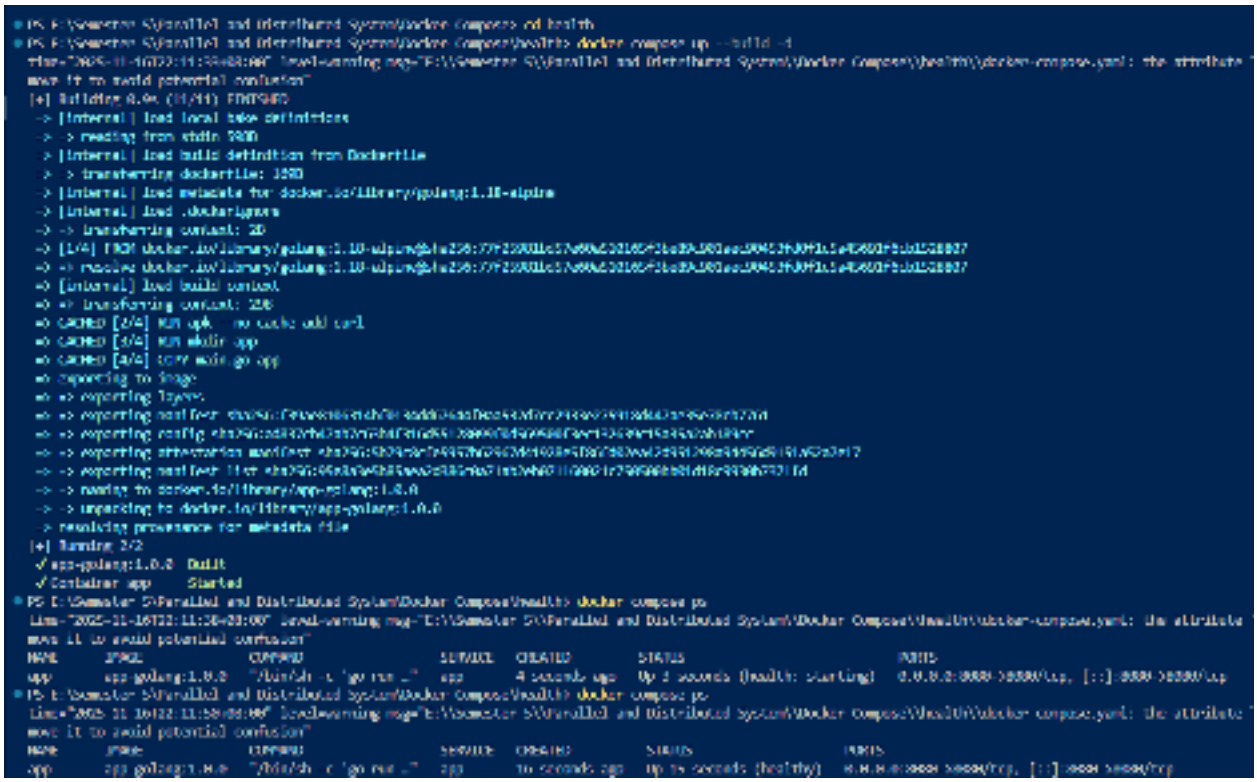
health > app > Dockerfile > ...

```dockerfile
FROM golang:1.18-alpine

RUN apk --no-cache add curl
RUN mkdir app

COPY main.go app

EXPOSE 8080

CMD go run app/main.go
```

health > app > main.go

```go
package main

import (
    "fmt"
    "net/http"
)

var counter = 0

func main() {
    http.HandleFunc("/", HelloServer)
    http.HandleFunc("/health", HealthCheck)

    http.ListenAndServe(":8080", nil)
}

func HealthCheck(w http.ResponseWriter, r *http.Request) {
    counter = counter + 1
    if counter > 5 {
        w.WriteHeader(500)
        fmt.Fprintf(w, "KO")
    } else {
        fmt.Fprintf(w, "OK")
    }
}

func HelloServer(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello, World!")
}
```

Docker Compose: HEALTHCHECK



Docker Compose: Disable HEALTHCHECK

health > docker-compose.yaml

```yaml
version: "3.9"

Run All Services
services:
  Run Service
  app:
    container_name: app
    build:
      context: "./app"
      dockerfile: Dockerfile
    image: "app-golang:1.0.0"
    environment:
      - "APP_PORT=8080"
    ports:
      - "8080:8080"
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8080/health"]
      interval: 5s
      timeout: 5s
      retries: 3
      start_period: 3s
      disable: true
```

EXTEND SERVICE

Pengenalan EXTEND SERVICE

service > docker-compose.yaml

```yaml
version: "3.9"

services:
  app:
    container_name: app
    build:
      context: "./app"
      dockerfile: Dockerfile
    image: "app-golang:1.0.0"
    environment:
      - "APP_PORT=8080"
      - "MODE=local"
    ports:
      - "8080:8080"
```

service > dev.yaml

```yaml
version: "3.9"

services:
  app:
    environment:
      -"MODE=dev"
```

service > prod.yaml

```yaml
version: "3.9"

services:
  app:
    environment:
      -"MODE=prod"
```

service > app > Dockerfile > ...

```
FROM golang:1.18-alpine

ENV APP_PORT=8080
ENV MODE=local

RUN mkdir app
COPY main.go app

EXPOSE ${APP_PORT}

CMD go run app/main.go
```

service > app > main.go

```
package main

import (
    "fmt"
    "net/http"
    "os"
)

func main() {
    port := os.Getenv("APP_PORT")
    fmt.Println("Run app in port : " + port)
    http.HandleFunc("/", HelloServer)
    http.ListenAndServe(":"+port, nil)
}

func HelloServer(w http.ResponseWriter, r *http.Request) {
    mode := os.Getenv("MODE")
    response := "Hello " + mode
    fmt.Fprintf(w, response)
}
```

Docker Compose: EXTEND SERVICE