Nama : Ervalsa Dwi Nanda

NIM : 11201028

Mata Kuliah : Sistem Terdistribusi

Praktikum Docker Compose

Link Github: https://github.com/ervalsa/sister-3

Penjelasan Docker Compose

Tool yang digunakan untuk mendefinisikan dan menjalankan multiple Docker Container sekaligus. Dapat menggunakan file YAML untuk melakukan konfigurasi Docker Container. Sebuah perintah dapat membuat semua Docker Container dan menjalankannya sekaligus dari file konfigurasi itu dan tidak perlu mengetikan perintah Docker Create secara manual ketika ingin membuat Docker Container

Syntax Praktikum

Service

Container yang dibuat akan disimpan di dalam konfigurasi bernama services, kita dapat menambahkan satu atau lebih services dalam konfigurasi file.

Buatlah file docker-compose.yaml seperti gambar di bawah ini

Lalu jalankan, docker compose create, docker compose start, dan docker compose ps

```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Praktikum/praktikum-docker-compose/services (main)
$ docker compose create
[+] Running 2/2
 - Container mongodb-example  Created                                                                                    0.2s
 - Container nginx-example    Created                                                                                    0.2s

erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Praktikum/praktikum-docker-compose/services (main)
$ docker compose start
[+] Running 2/2
 - Container nginx-example    Started                                                                                    0.6s
 - Container mongodb-example  Started                                                                                    0.7s

erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Praktikum/praktikum-docker-compose/services (main)
$ docker compose ps
NAME                COMMAND                  SERVICE             STATUS              PORTS
mongodb-example     "docker-entrypoint.s…"   mongodb-example     running             27017/tcp
nginx-example       "/docker-entrypoint.…"   nginx-example       running             80/tcp
```

Jika tidak terpakai, maka berhentikan dengan perintah docker compose down

```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Praktikum/praktikum-docker-compose/services (main)
$ docker compose down
[+] Running 3/3
 - Container nginx-example    Removed                                                                                    0.7s
 - Container mongodb-example  Removed                                                                                    0.5s
 - Network services_default   Removed                                                                                    0.3s
```

Komentar

Keunggulan menggunakan Yaml dibandingkan JSON adalah kita dapat menambahkan komentar. Komentar secara otomatis akan dihiraukan oleh Docker Compose.

Contohnya seperti ini

```
docker-compose.yaml U ×   script.sh U
services > docker-compose.yaml > {} services
docker-compose.yml - The Compose specification establishes a standard for
1   version: "3.9"
2
3   services:
4     # Untuk membuat container nginx-example
5     nginx-example:
6       image: nginx:latest
7       container_name: nginx-example
8
9     # Untuk membuat container mongo-example
10    mongodb-example:
11      image: mongo:latest
12      container_name: mongodb-example
```

Jika kita menjalankan command docker compose, maka tidak aka nada masalah. Ini hanya seperti menambahkan dokumentasi pada kode.

Port

Saat membuat container, kita bisa mengekspos port di container menggunakan Port Forwarding. Attribute ports berisi array object port.

Short Syntax

Berisi string port HOST:CONTAINER. Contoh 8080:80, artinya kita akan menggunakan port 8080 di host untuk di forward ke port 80 di Container.

Long Syntax

Bisa digunakan dalam bentuk object.

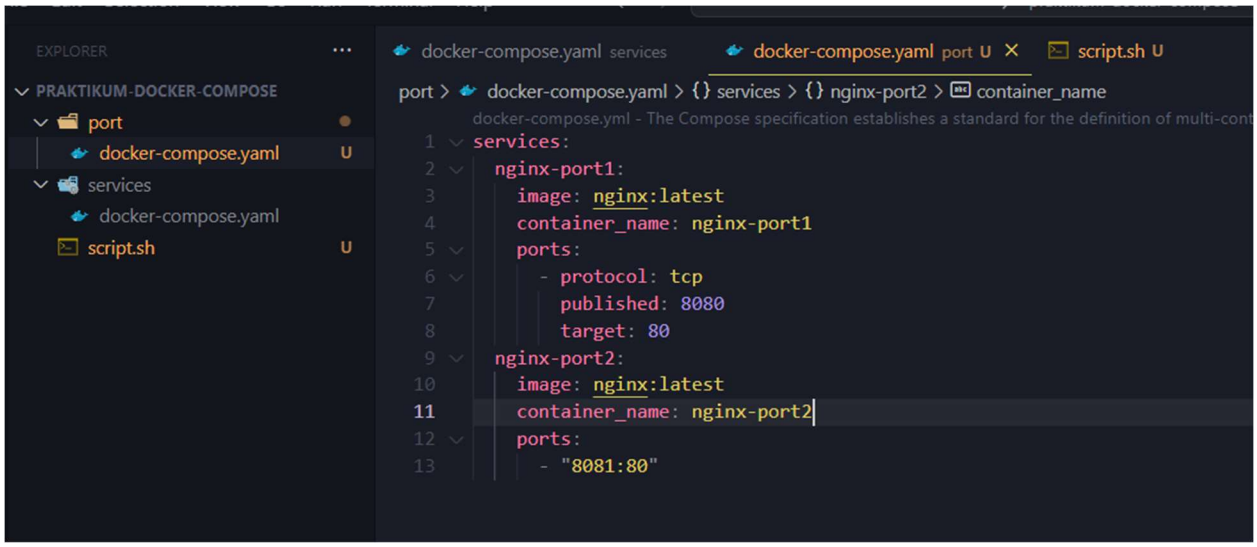
target: Port

published: Port

protocol: Protocol

mode: host

Buatlah file docker-compose.yaml di dalam folder port seperti gambar di bawah.



Lakukanlah beberapa perintah di bawah ini.

```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Praktikum/praktikum-docker-compose/ports (main)
$ docker compose create
[+] Running 3/3
 - Network ports_default  Created                                                                                    0.1s
 - Container nginx-port2  Created                                                                                    0.1s
 - Container nginx-port1  Created                                                                                    0.1s

erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Praktikum/praktikum-docker-compose/ports (main)
$ docker compose start
[+] Running 2/2
 - Container nginx-port2  Started                                                                                    0.6s
 - Container nginx-port1  Started                                                                                    0.7s

erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Praktikum/praktikum-docker-compose/ports (main)
$ docker compose ps
NAME               COMMAND                  SERVICE            STATUS             PORTS
nginx-port1        "/docker-entrypoint.…"   nginx-port1        running            0.0.0.0:8080->80/tcp
nginx-port2        "/docker-entrypoint.…"   nginx-port2        running            0.0.0.0:8081->80/tcp
```

Kita dapat melihat bahwa ports sesuai dengan yang kita definisikan. Container nginx-port1 berjalan pada port 8080:80 dan Container nginx-port2 berjalan di port 8081:80.

Cek localhost:8080

```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Praktikum/prak
$ curl localhost:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

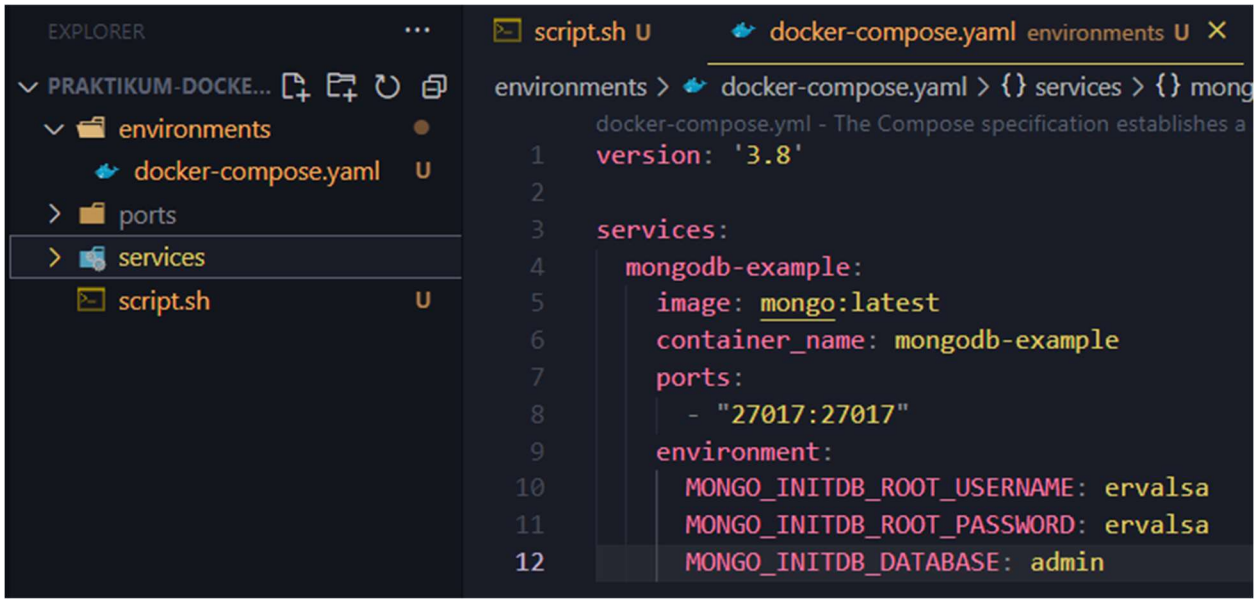
Cek localhost:8081

```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Praktikum/praktikum-docker-compose/ports (main)
$ curl localhost:8081
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Environment Variable

Kita perlu menambahkan environment untuk digunakan di dalam Conainer. Saat mengkonfigurasi file Docker Compose, kitab isa tambahkan environment variable menggunakan attribute environment.

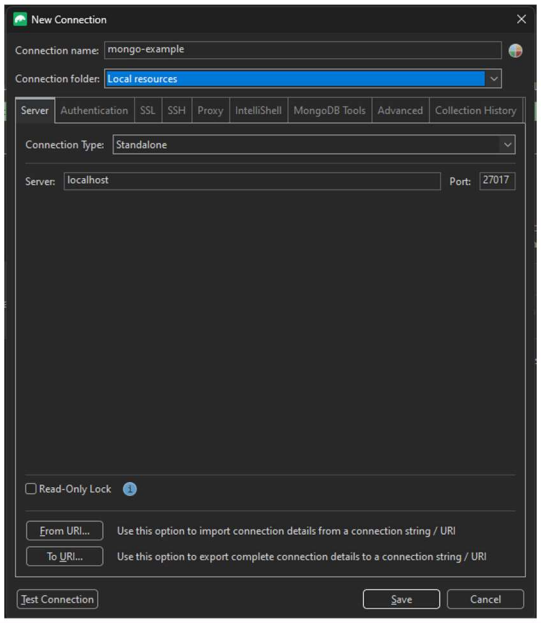
Buatlah file docker-compose.yaml di dalam folder environments

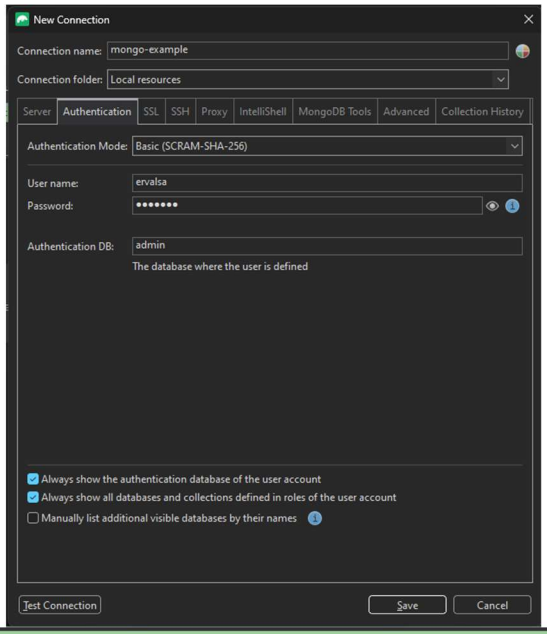


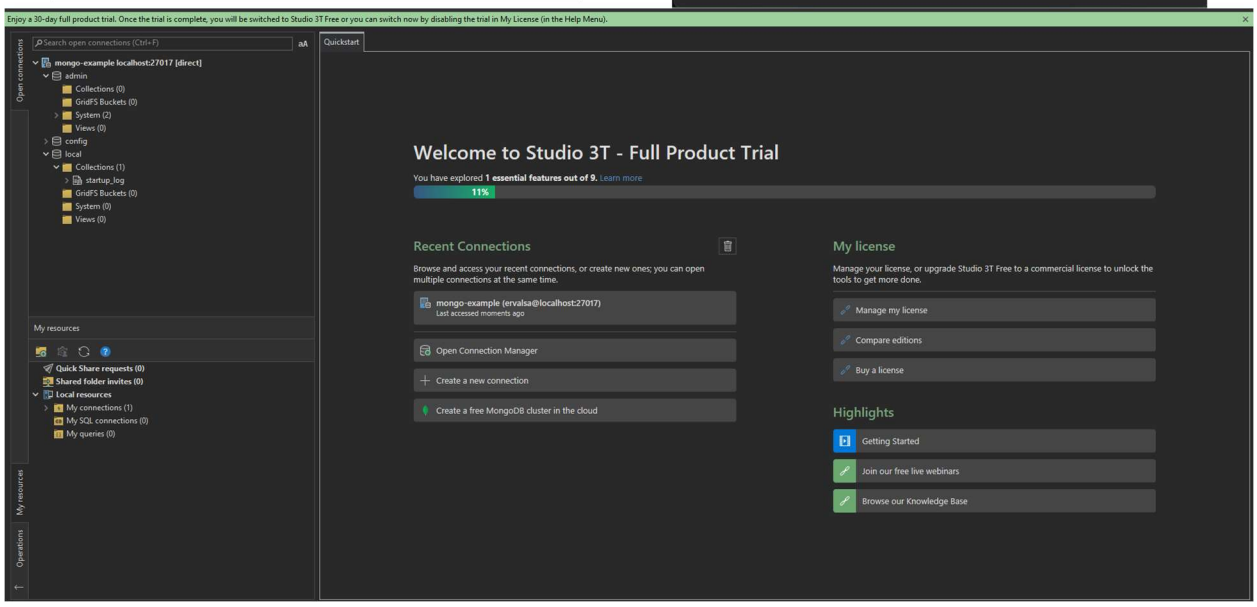
Lalu create Docker Compose lalu jalankan.

```
$ docker compose create
[+] Running 2/2
 - Network environments_default  Created                                    0.0s
 - Container mongodb-example     Created                                    0.1s

erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Prak
tikum/praktikum-docker-compose/environments (main)
$ docker compose start
[+] Running 1/1
 - Container mongodb-example  Started                                       0.9s
```

Tes koneksi mongodb menggunakan mongodb client studio 3T







Jika sudah bisa terbaca seperti gambar di atas, maka koneksi berhasil.

Bind Mount

Bind mount dapat dilakkan di konfirgurasi file Docker Compose. Bisa menggunakan attribute volumes di services dan dapat ditambahkan satu atau lebih bind mount jika diperlukan.

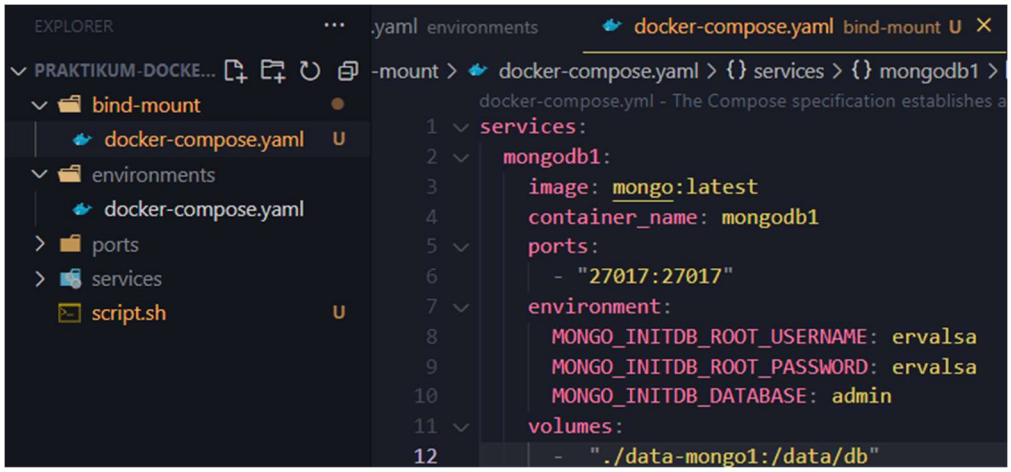
Short Syntax

SOURCE:TARGET:MODE

SOURCE = lokasi di host, bisa menggunakan relative path yang diawali dengan . (titik) atau absolute path

TARGET = lokasi di container

MODE = mode bind mount, ro (readonly), rw (read write) – default




```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem
[+] Running 2/2
 - Network bind-mount_default  Created                    0.1s
 - Container mongodb1          Cr...                      0.1s

erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem
 Terdistribusi/A. Praktikum/praktikum-docker-compose/bind-mount (main)
$ docker compose start
[+] Running 1/1
 - Container mongodb1  Started                            0.9s
```

Data di folder data-mongo1

Long Syntax

Dapat membuat nested object di volumes dengan attribute.

Type = tipe mounth, volume atau bind.

Source = sumber path di host atau nama volume

Target = target path di container

Read_only = flag readonly atau tidak, default nya false

```yaml
mongodb2:
  image: mongo:latest
  container_name: mongodb2
  ports:
    - "27017:27017"
  environment:
    MONGO_INITDB_ROOT_USERNAME: ervalsa
    MONGO_INITDB_ROOT_PASSWORD: ervalsa
    MONGO_INITDB_DATABASE: admin
  volumes:
    - type: bind
      source: "./data-mongo2"
      target: "/data/db"
      read_only: false
```

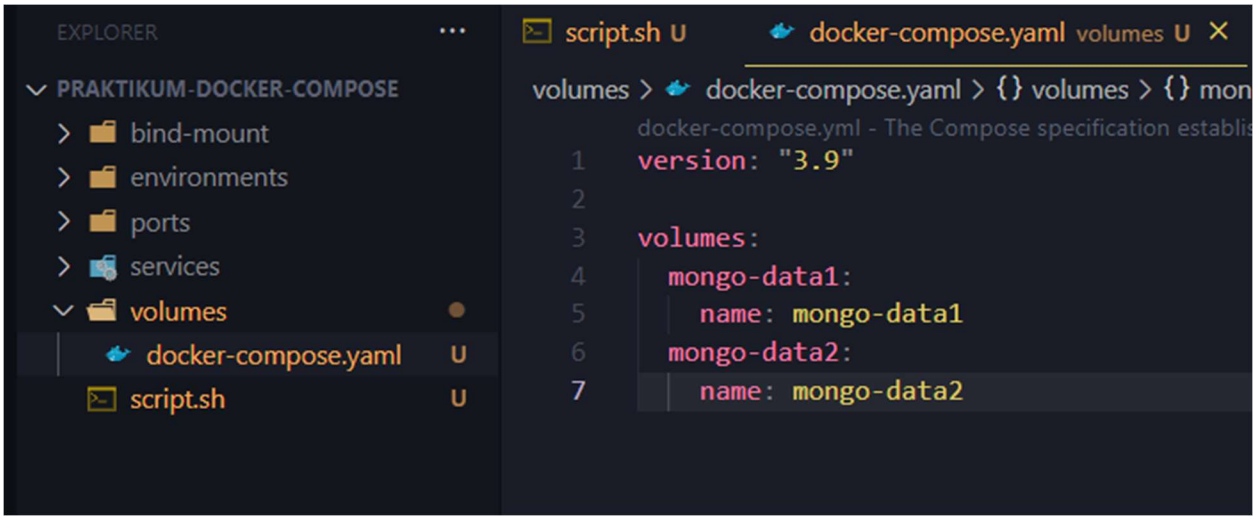
```
$ docker compose create
[+] Running 3/3
 - Network bind-mount_default  Created                                    0.1s
 - Container mongodb2          Created                                    0.2s
 - Container mongodb1          Created                                    0.2s

erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Prak
tikum/praktikum-docker-compose/bind-mount (main)
$ docker compose start
[+] Running 1/2
 - Container mongodb2  Starting                                           0.8s
 - Container mongodb1  Started                                            0.8s
Error response from daemon: driver failed programming external connectivity on endpoint mongod
b2 (7e6b75b23a619606c5aeee8d828e6b84c7089ff8a2d057de56781db2334a763c): Bind for 0.0.0.0:27017
failed: port is already allocated

erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Prak
tikum/praktikum-docker-compose/bind-mount (main)
$ docker compose create
[+] Running 2/2
 - Container mongodb2  Recreated                                          0.3s
 - Container mongodb1  Running                                            0.0s

erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Prak
tikum/praktikum-docker-compose/bind-mount (main)
$ docker compose start
[+] Running 1/1
 - Container mongodb2  Started                                            0.6s
```

Note: jangan menggunakan port yang sama untuk container service yang sama.

Volume

Docker Compose juga tidak hanya bisa membuat container, tapi bisa juga digunakan untuk membuat volume. Kita dapat menggunakan attribute volumes pada konfigurasi file.

Buatlah file docker-compose.yaml di dalam folder volumes



Lengkapnya seperti ini

```yaml
services:
  mongodb1:
    image: mongo:latest
    container_name: mongodb1
    ports:
      - "27017:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: ervalsa
      MONGO_INITDB_ROOT_PASSWORD: ervalsa
      MONGO_INITDB_DATABASE: admin
    volumes:
      - "mongo-data1:/data/db"
  mongodb2:
    image: mongo:latest
    container_name: mongodb2
    ports:
      - "27018:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: ervalsa
      MONGO_INITDB_ROOT_PASSWORD: ervalsa
      MONGO_INITDB_DATABASE: admin
    volumes:
      - type: volume
        source: mongo-data2
        target: "/data/db"
        read_only: false

volumes:
  mongo-data1:
    name: mongo-data1
  mongo-data2:
    name: mongo-data2
```

Lalu buatlah Docker Compose

```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semeste
r 7/Sistem Terdistribusi/A. Praktikum/praktikum-docker-compose
/volumes (main)
$ docker compose create
[+] Creating 3/0
 ✓ Network volumes_default  Created                    0.0s
 ✓ Volume "mongo-data1"     Cr...                      0.0s
 ✓ Volume "mongo-data2"     Cr...                      0.0s
[+] Creating 3/5godb1       Crea...                    0.0s
 ✓ Network volumes_default  Created                    0.0s
 ✓ Volume "mongo-data1"     Cr...                      0.0s
 ✓ Volume "mongo-data2"     Cr...                      0.0s
[+] Creating 5/5godb1       Crea...                    0.1s
 ✓ Network volumes_default  Created                    0.0s
 ✓ Volume "mongo-data1"     Cr...                      0.0s
 ✓ Volume "mongo-data2"     Cr...                      0.0s
 ✓ Container mongodb1       Crea...                    0.2s
 ✓ Container mongodb2       Crea...                    0.2s
```

Dan start

```
$ docker compose start
[+] Running 2/2
 ✓ Container mongodb2  Started                    0.9s
 ✓ Container mongodb1  Started                    0.9s
```

Maka, hasilnya di folder volumes tidak akan membuat folder mongo-data1 ataupun mongo-data2, namun akan tersimpan sebagai volumes.

Ceklah volumes dengan perintah docker volume ls

```
local     mongo-data1
local     mongo-data2
```

Untuk menghapus volume, dapat menggunakan perintah docker volume rm nama-volume.

```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semeste
r 7/Sistem Terdistribusi/A. Praktikum/praktikum-docker-compose
/volumes (main)
$ docker volume rm mongo-data1
mongo-data1

erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semeste
r 7/Sistem Terdistribusi/A. Praktikum/praktikum-docker-compose
/volumes (main)
$ docker volume rm mongo-data2
mongo-data2
```

Namun, jika masih digunakan akan terjadi error seperti ini.

```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semeste
r 7/Sistem Terdistribusi/A. Praktikum/praktikum-docker-compose
/volumes (main)
$ docker volume rm mongo-data1
Error response from daemon: remove mongo-data1: volume is in u
se - [7564c1591013ce3505c94275a28eba875b6a9db35fc55bea4a7f940a
8b0bf2d1]
```

Network

Selain dapat membuat Container dan Volume, kita dapat membuat Network secara otomatis. Secara default semua container akan dihubungkan dalam sebuah network bernama nama-project_default. Jadi tidak perlu membuat network secara manual.

```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semeste
r 7/Sistem Terdistribusi/A. Praktikum/praktikum-docker-compose
/example (main)
$ docker compose create
[+] Creating 3/3
 ✓ Network example_default   Created                0.0s
 ✓ Container nginx-example2  Created                0.1s
 ✓ Container nginx-example   Created                0.1s
```

Terlihat bahwa network dibuat secara otomatis. Bisa melakukan inspect pada docker compose jika ingin melihat detailnya.

```
"Networks": {
    "example_default": {
        "IPAMConfig": null,
        "Links": null,
        "Aliases": [
            "nginx-example",
            "nginx-example",
            "d603ffeea70a"
        ],
```

Membuat network manual

Perlu 2 hal dibawah ini.

Name: nama network

Driver: driver network seperti bridge, host atau none

```
docker-compose.yml - The Compose specific
1  version: "3.9"
2
3  networks:
4    network-example:
5      name: network-example
6      driver: bridge
```

```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semeste
r 7/Sistem Terdistribusi/A. Praktikum/praktikum-docker-compose
/networks (main)
$ docker compose create
[+] Creating 2/2
 ✓ Network network-example    Created                    0.1s
 ✓ Container mongodb-example  Created                    0.1s
```

Depends On

Terkadang kita membutuhkan Container lain sebelum membuat container baru. Kita bisa menjalankan semua container secara bersamaan di Docker Compose tanpa ada urutan pasti. Disinilah peran Depends On agar kita dapat menyebutkan satu atau lebih container lainnya pada konfigurasi file

```yaml
version: '3.8'

services:
  mongodb-example:
    image: mongo:latest
    container_name: mongodb-example
    ports:
      - "27017:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: ervalsa
      MONGO_INITDB_ROOT_PASSWORD: ervalsa
      MONGO_INITDB_DATABASE: admin
    networks:
      - network_example

  mongodb-express-example:
    image: mongo-express:latest
    container_name: mongodb-express-example
    ports:
      - "8081:8081"
    environment:
      ME_CONFIG_MONGODB_ADMINUSER: ervalsa
      ME_CONFIG_MONGODB_ADMINPASSWORD: ervalsa
      ME_CONFIG_MONGODB_SERVER: mongodb-example
    networks:
      - network_example
    depends_on:
      - mongodb-example

networks:
  network_example:
    name: network_example
    driver: bridge
```

container mongodb-express-example akan dibuat setelah mongodb-example dibuat, karena container mongodb-express-example depends_on mongodb-example.

Berikut hasilnya.

```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Prak
tikum/praktikum-docker-compose/depends-on (main)
$ docker compose create
[+] Creating 3/3
 ✓ Network network_example           Created                                        0.1s
 ✓ Container mongodb-example         Created                                        0.1s
 ✓ Container mongodb-express-example Created                                        0.1s

erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Prak
tikum/praktikum-docker-compose/depends-on (main)
$ docker compose start
[+] Running 2/2
 ✓ Container mongodb-example         Started                                        0.5s
 ✓ Container mongodb-express-example Started                                        0.4s

erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Prak
tikum/praktikum-docker-compose/depends-on (main)
$ docker compose ps
NAME                     IMAGE                  COMMAND                  SERVICE
      CREATED         STATUS         PORTS
mongodb-example          mongo:latest           "docker-entrypoint.s…"   mongodb-example
      9 seconds ago   Up 4 seconds   0.0.0.0:27017->27017/tcp
mongodb-express-example   mongo-express:latest   "/sbin/tini -- /dock…"   mongodb-express-exam
ple   9 seconds ago   Up 3 seconds   0.0.0.0:8081->8081/tcp
```

Restart

Saat Container mati, maka Docker tidak akan menjalankan Containernya lagi, kita harus secara manual untuk menyalakannya. Kita bisa memaksa sebuah Container untuk selalu melakukan restart jika missal terjadi masalah pada Containernya. Ada beberapa attribute restart dengan beberapa value

No = tidak pernah restart

Always = selalu restart jika container mati

On-failure = restart jika container mengalami error dengan indikasi error ketika exit

Unless-stopped = selalu restart container, kecuali ketika dihentikan manual

```yaml

services:
  mongodb-example:
    image: mongo:latest
    container_name: mongodb-example
    ports:
      - "27017:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: ervalsa
      MONGO_INITDB_ROOT_PASSWORD: ervalsa
      MONGO_INITDB_DATABASE: admin
    networks:
      - network_example

  mongodb-express-example:
    image: mongo-express:latest
    container_name: mongodb-express-example
    restart: always
    ports:
      - "8081:8081"
    environment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: ervalsa
      ME_CONFIG_MONGODB_ADMINPASSWORD: ervalsa
      ME_CONFIG_MONGODB_SERVER: mongodb-example
    networks:
      - network_example
    depends_on:
      - mongodb-example

networks:
  network_example:
    name: network_example
    driver: bridge
```

Jika ingin melihat, apakah container tersebut pernah di restart apa tidak. Maka bisa menggunakan command docker container logs nama-container, jika ingin melihat semua events, bisa menggunakan docker events

Resource Limit

Kita dapat mengatur Resource limit untuk CPU dan Memory tiap Container yang akan kita buat. Dapat menggunakan attribute deploy, lalu didalamnya menggunakan attribute resources. Di dalam attribute resources dapat menentukan limit dan reservations. Reservations adalah resource yang dijamin bisa digunakan oleh container. Limi adalah maksimal untuk resource yang diberikan ke container namun bisa rebutan dengan container lain.

```yaml
services:
  nginx-port1:
    image: nginx:latest
    container_name: nginx-port1
    ports:
      - protocol: tcp
        published: 8080
        target: 80
    deploy:
      resources:
        reservations:
          cpus: "0.25"
          memory: 50M
        limits:
          cpus: "0.5"
          memory: 100M
```

Jika ingin melihat status dari container maka bisa menggunakan perintah docker container stats

```
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O       BLOCK I/O
 PIDS
7951844008f0   nginx-port1   0.00%     9.949MiB / 100MiB   9.95%     1.18kB / 0B   0B / 0B
 13
```

Dockerfile

Buatlah file docker-compose.yaml seperti gambar dibawah.

```
EXPLORER
PRAKTIKUM-DOCKER-COMPOSE
  bind-mount
  build
    app
      Dockerfile U
      main.go U
      docker-compose.yaml U
  depends-on
  environments
  example
  networks
  ports
  resource-limit
  restarts
  services
  volumes
  script.sh
```

```yaml
docker-compose.yaml build U ×   Dockerfile
build > docker-compose.yaml > {} services > {} a
docker-compose.yml - The Compose specificatio
version: "3.8"

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

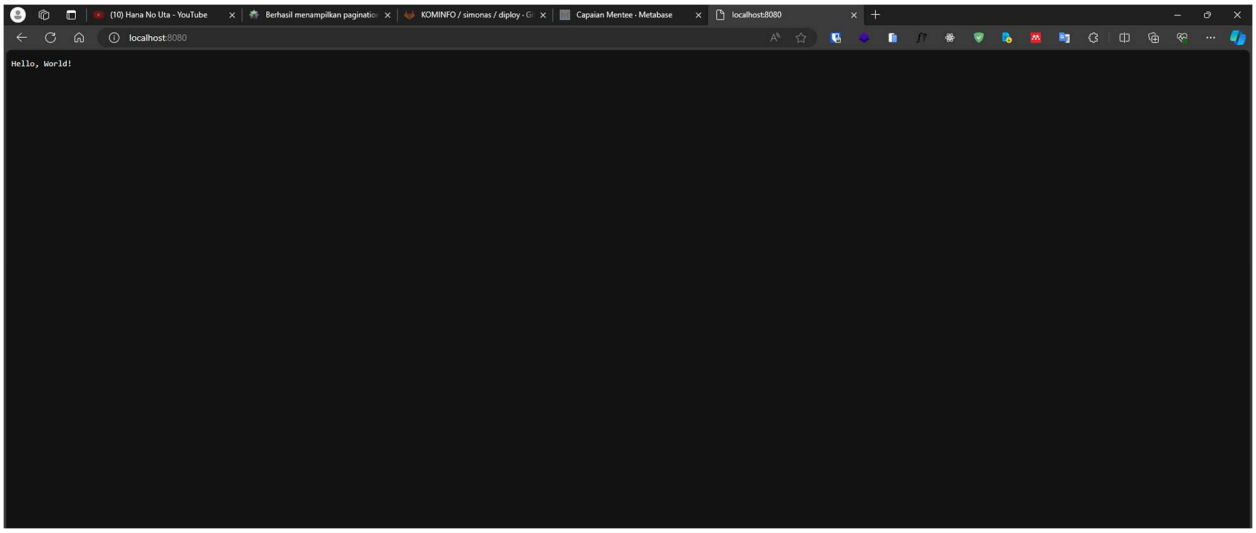
Lalu kita build docker compose-nya.

```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Prak
tikum/praktikum-docker-compose/build (main)
$ docker compose build
[+] Building 2.9s (8/8) FINISHED                                          docker:default
 => [app internal] load build definition from Dockerfile                            0.1s
 => => transferring dockerfile: 164B                                                0.0s
 => [app internal] load .dockerignore                                               0.1s
 => => transferring context: 2B                                                     0.0s
 => [app internal] load metadata for docker.io/library/golang:1.18-alpine           1.8s
 => [app 1/3] FROM docker.io/library/golang:1.18-alpine@sha256:77f25981bd57e60a510165f3  0.2s
 => => resolve docker.io/library/golang:1.18-alpine@sha256:77f25981bd57e60a510165f3be89  0.1s
 => => sha256:77f25981bd57e60a510165f3be89c901aec90453fd0f1c5a45691f6cb 1.65kB / 1.65kB  0.0s
 => => sha256:ab5685692564e027aa84e2980855775b2e48f8fc82c1590c0e1e8cbc2 1.16kB / 1.16kB  0.0s
 => => sha256:a77f45e5f987fb7def8755903ad89fe37a38105dcf475be26550d7d86 5.01kB / 5.01kB  0.0s
 => [app internal] load build context                                               0.1s
 => => transferring context: 360B                                                   0.0s
 => [app 2/3] RUN mkdir app                                                         0.4s
 => [app 3/3] COPY main.go app                                                      0.1s
 => [app] exporting to image                                                        0.1s
 => => exporting layers                                                             0.1s
 => => writing image sha256:87fbbec3d35962b8f7fcd907a925d38604fc91cf475e67b80ebea52eb84  0.0s
 => => naming to docker.io/library/app-golang:1.0.0                                 0.0s
```

```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Prak
tikum/praktikum-docker-compose/build (main)
$ docker image ls
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
app-golang                  1.0.0     87fbbec3d359   23 seconds ago   330MB
```

```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Prak
tikum/praktikum-docker-compose/build (main)
$ docker compose create
[+] Creating 2/2
 ✓ Network build_default  Created                                          0.0s
 ✓ Container app          Created                                          0.1s

erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Prak
tikum/praktikum-docker-compose/build (main)
$ docker compose start
[+] Running 1/1
 ✓ Container app  Started                                                  0.3s
```

Kita buat docker compose dan start. Maka akan terlihat seperti gambar di atas.



Health Check

Health check attribute

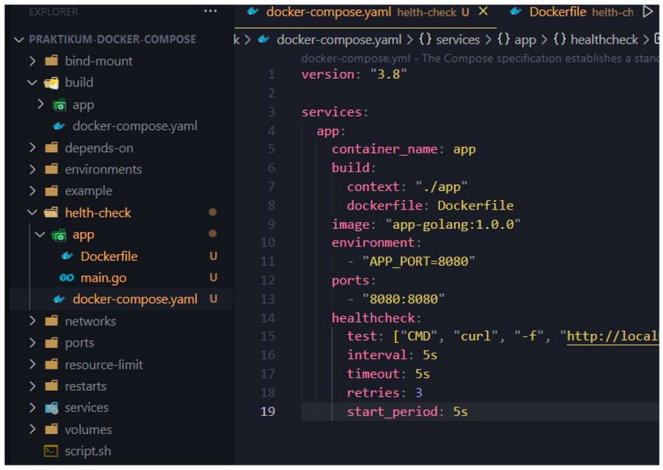
Test = cara melakukan test health check

Interval = interval melakukan health check

Timeout = timeout melakukan health check

Retries = total retry ketika ggal

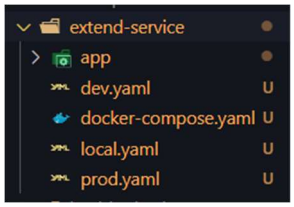
Start_period = waktu mulai melakukan helth check




```
erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Prak
tikum/praktikum-docker-compose/health-check (main)
$ docker compose build
[+] Building 1.3s (9/9) FINISHED                                          docker:default
 => [app internal] load build definition from Dockerfile                            0.1s
 => => transferring dockerfile: 169B                                                0.0s
 => [app internal] load .dockerignore                                               0.1s
 => => transferring context: 2B                                                     0.0s
 => [app internal] load metadata for docker.io/library/golang:1.18-alpine           1.0s
 => [app 1/4] FROM docker.io/library/golang:1.18-alpine@sha256:77f25981bd57e60a510165f3  0.0s
 => [app internal] load build context                                               0.0s
 => => transferring context: 26B                                                    0.0s
 => CACHED [app 2/4] RUN apk --no-cache add curl                                    0.0s
 => CACHED [app 3/4] RUN mkdir app                                                  0.0s
 => CACHED [app 4/4] COPY main.go app                                               0.0s
 => [app] exporting to image                                                        0.0s
 => => exporting layers                                                             0.0s
 => => writing image sha256:a1fe7f9026b15ac8ef64a67f7895691193f9739545ffa0e9e9a08eaceae  0.0s
 => => naming to docker.io/library/app-golang:1.0.0                                 0.0s

erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Prak
tikum/praktikum-docker-compose/health-check (main)
$ docker compose create
[+] Creating 2/2
 ✓ Network health-check_default  Created                                           0.0s
 ✓ Container app                 Created                                           0.1s

erval@Ervalsa MINGW64 /e/A. File Kuliah/A. Perkuliahan/Semester 7/Sistem Terdistribusi/A. Prak
tikum/praktikum-docker-compose/health-check (main)
$ docker compose start
[+] Running 1/1
 ✓ Container app  Started                                                          0.4s
```

Extends Service

Kita bisa membagi file konfigurasi berdasarkan env kita



Nanti akan diatur melalui MODE ini. Hal ini memudahkan bila kita ingin membagi environment berbeda dengan settingan dan kebutuhan yang berbeda.