Module 6

Containers Networking

Advanced Infrastructures for Data Science

Pedro Neves, pedroneves@dei.uc.pt, 2024/2025

Master in Data Science and Engineering (MDSE) Course

Outline

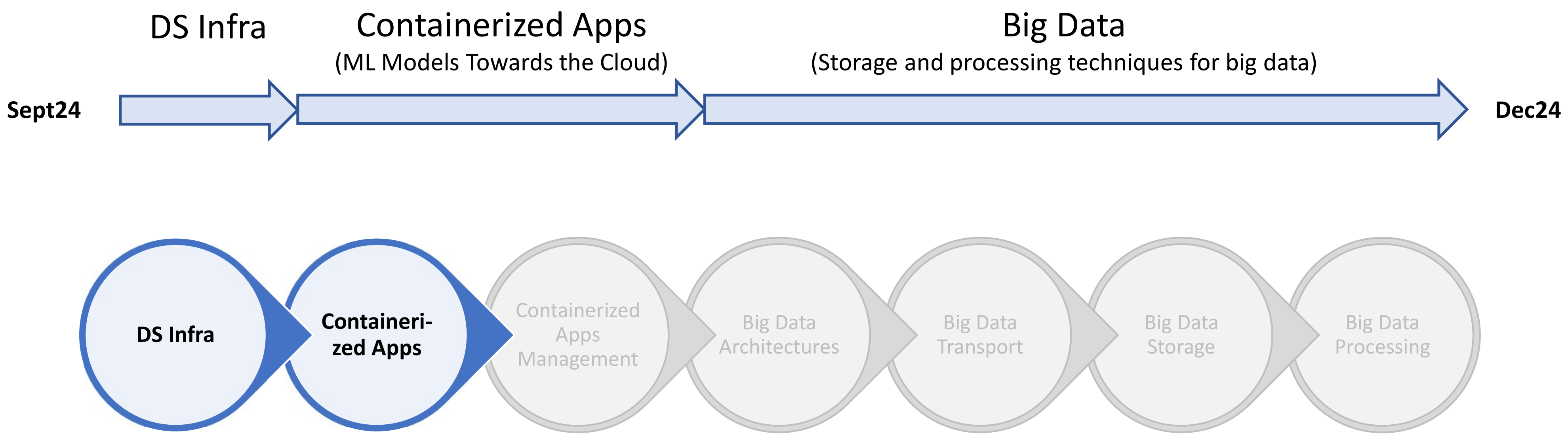
Outline

- Objectives
- Docker Networking Introduction
- Manipulating Docker Networking
- Summary & Bibliography

Objectives
What you will learn

1. Understand how Docker **containers manage** the **communication** with external entities

Unit Program

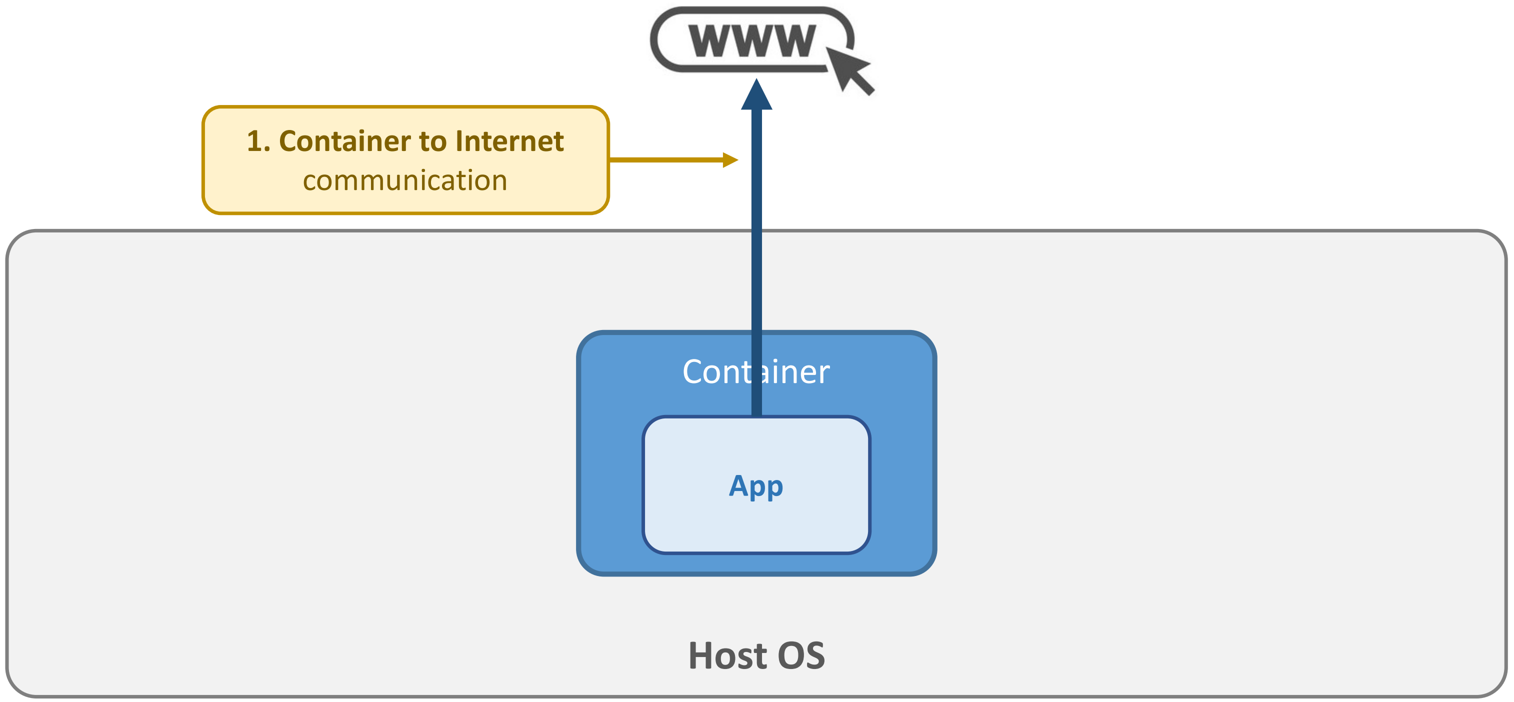


Outline

- Objectives
- Docker Networking Introduction
- Manipulating Docker Networking
- Summary & Bibliography

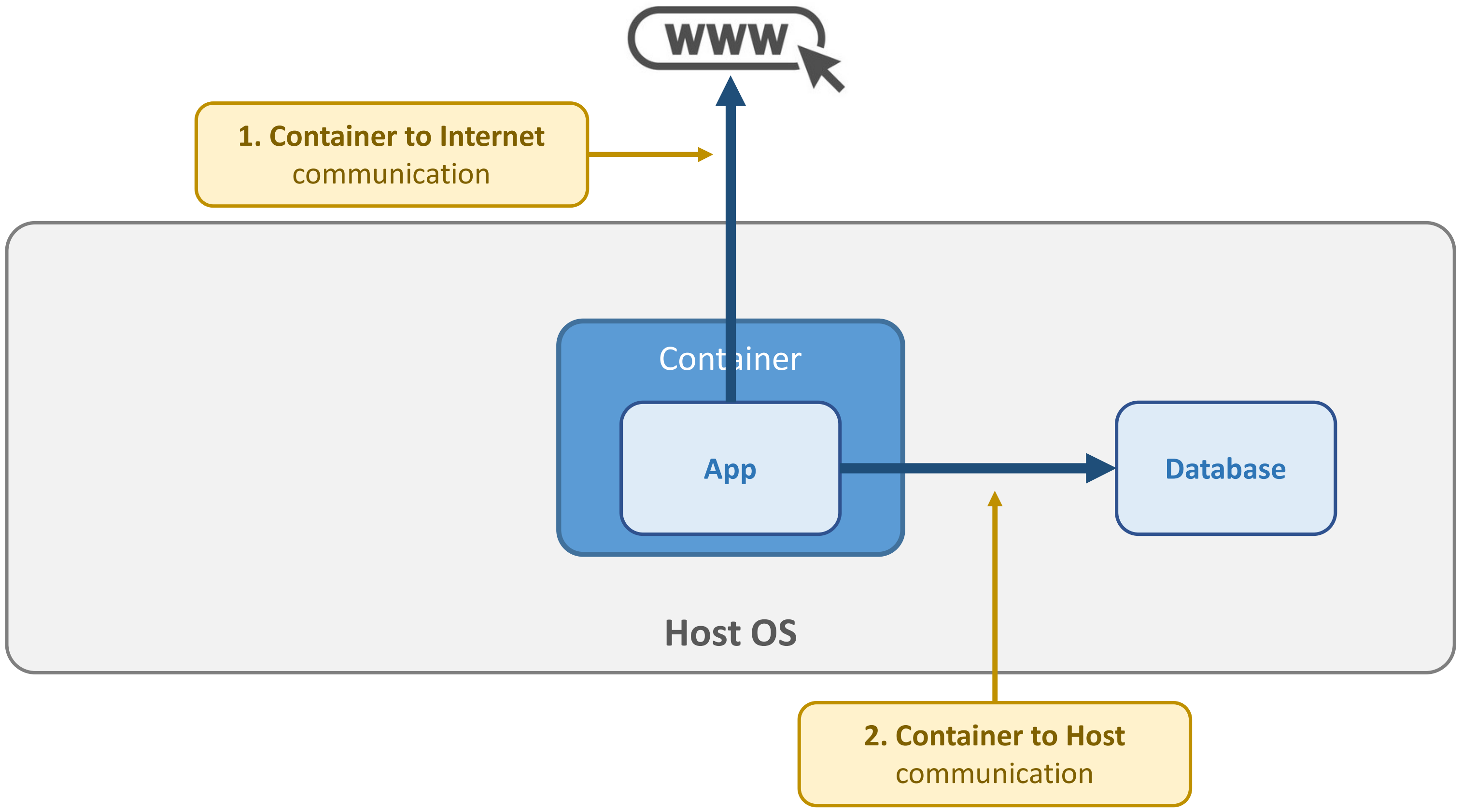
Docker Networking

Container to Internet Communication



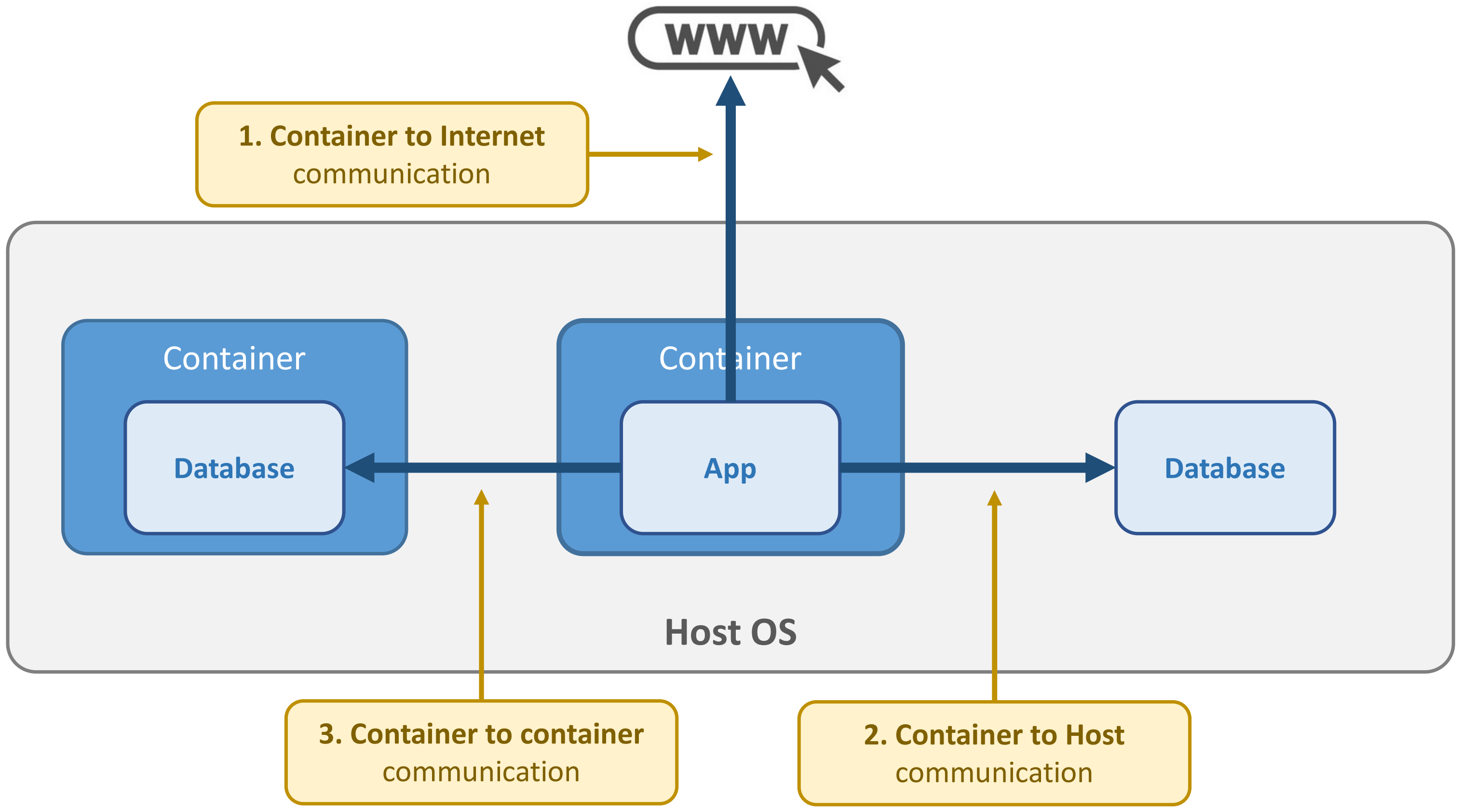
Docker Networking

Container to Host OS Communication


www
1. Container to Internet communication
Container
App
Database
Host OS
2. Container to Host communication

Docker Networking

Container to Container Communication


WWW
1. Container to Internet communication
Container
Database
Container
App
Database
Host OS
3. Container to container communication
2. Container to Host communication

Outline

- Objectives
- Docker Networking Introduction
- Manipulating Docker Networking
- Summary & Bibliography

Step 1

Analysis of the dummy **web app** for demonstrating Docker Networking

Favorite Movies Web App Example

App Objective

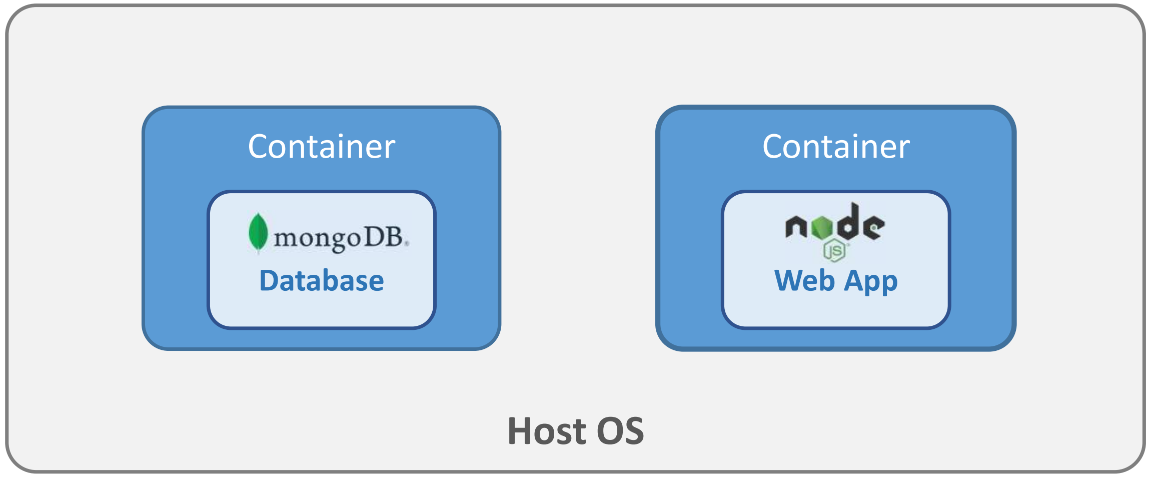
Objective → Application to fetch and store your favorite movies

Favorite Movies Web App Example

Architecture & Technologies

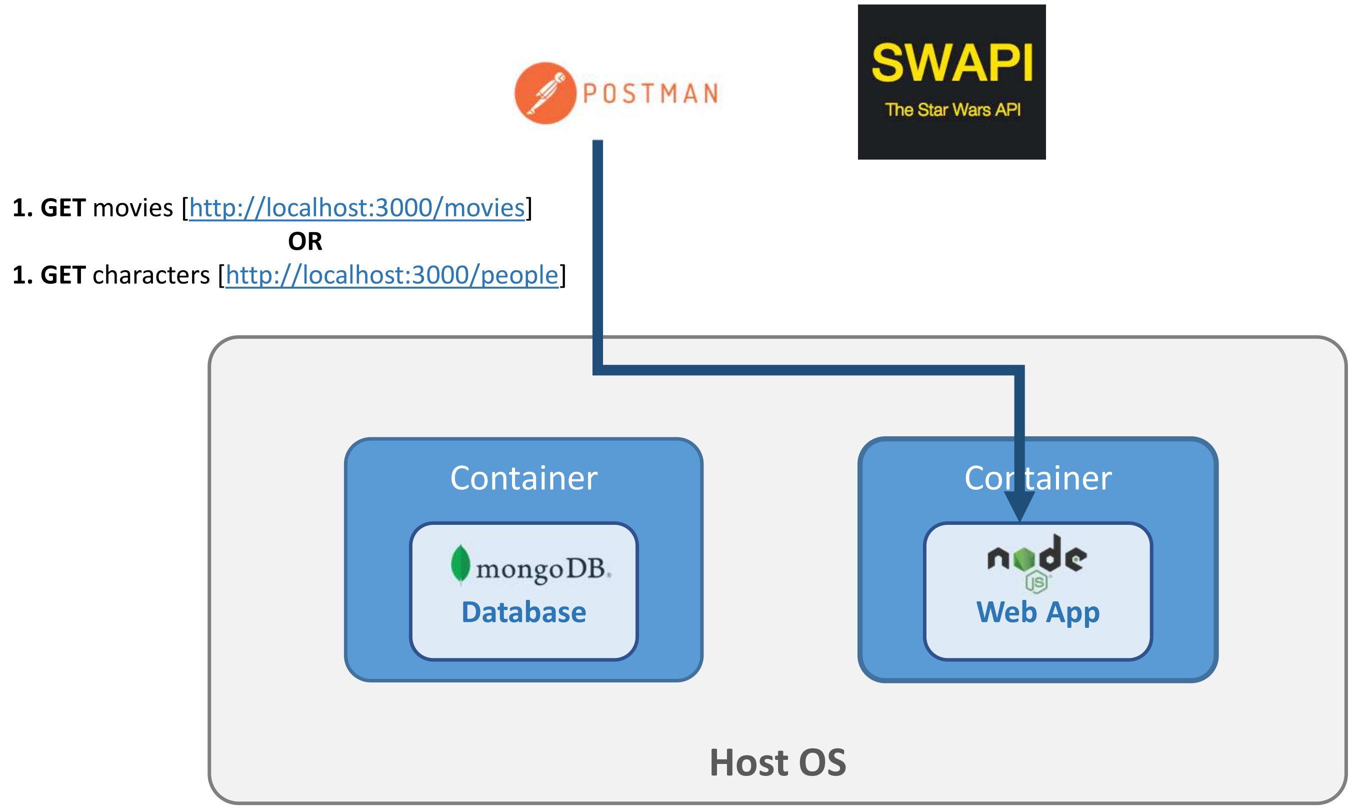
POSTMAN

SWAPI
The Star Wars API



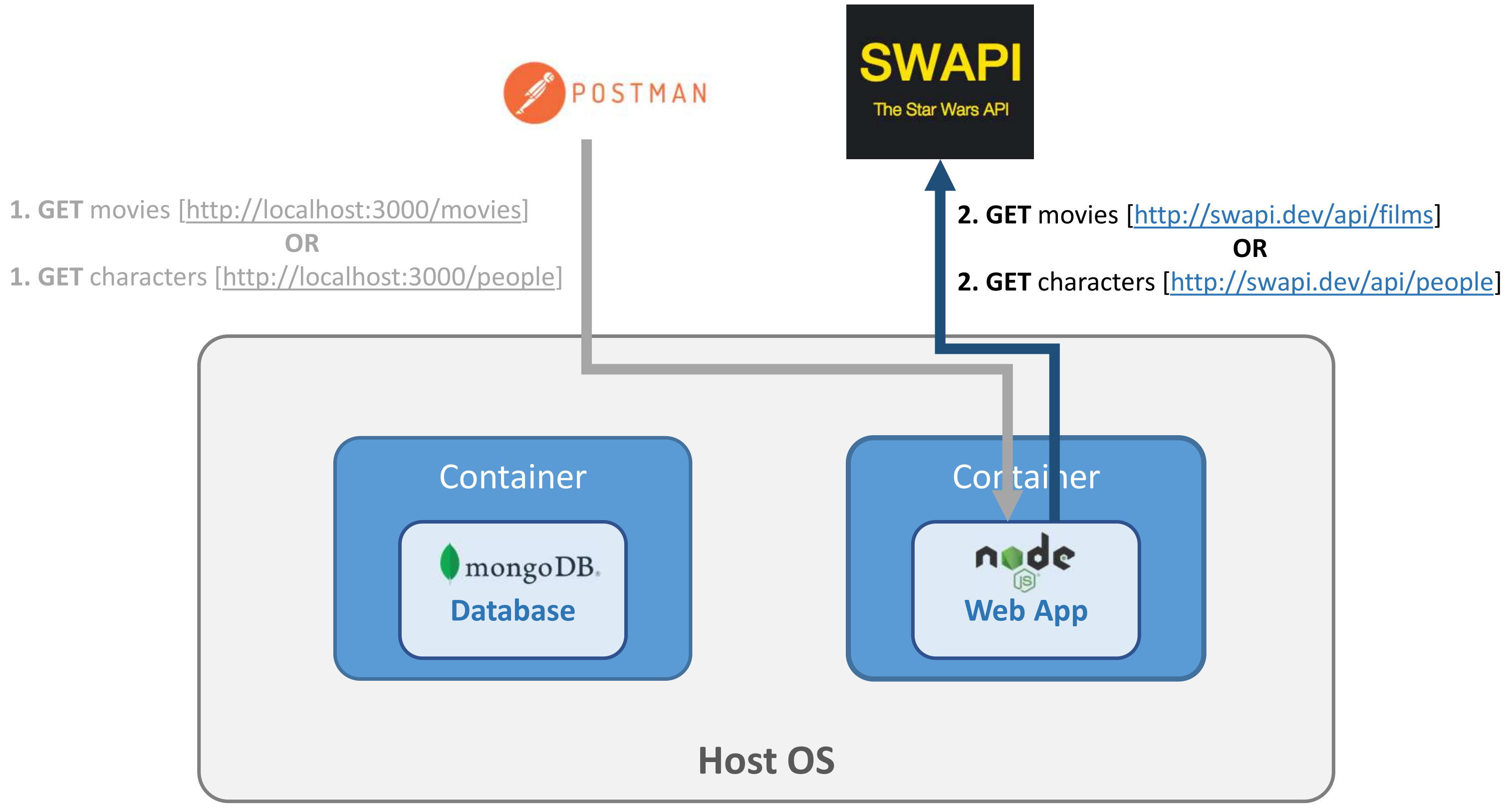
Favorite Movies Web App Example

Workflow – HTTP GET Movies and Characters Stored in Web App


POSTMAN
SWAPI
The Star Wars API
1. GET movies [http://localhost:3000/movies]
OR
1. GET characters [http://localhost:3000/people]
Container
mongoDB.
Database
Container
node JS
Web App
Host OS

Favorite Movies Web App Example

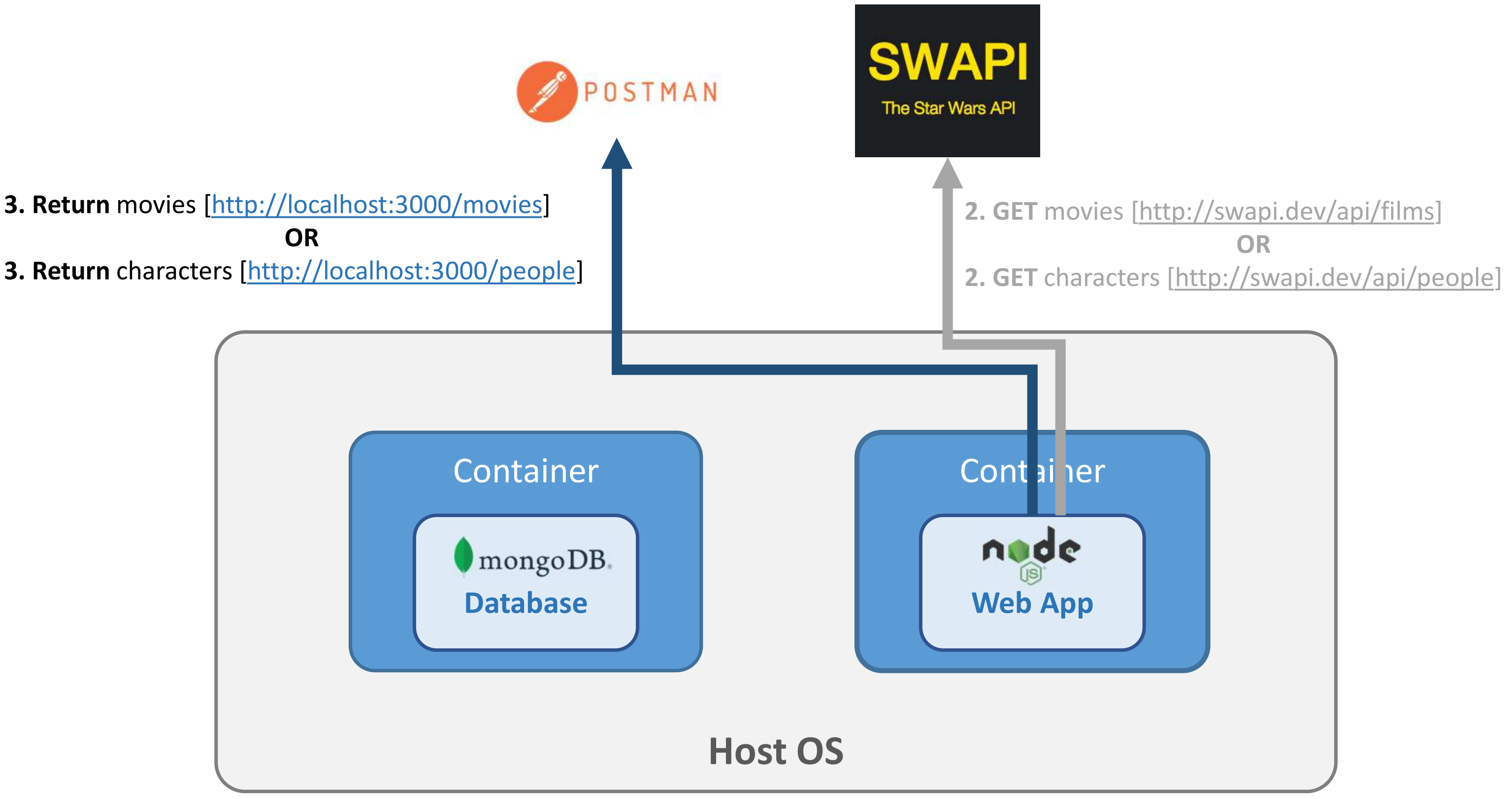
Workflow – HTTP GET Movies and Characters Stored in SWAPI



POSTMAN
SWAPI
The Star Wars API
1. GET movies [http://localhost:3000/movies]
OR
1. GET characters [http://localhost:3000/people]
2. GET movies [http://swapi.dev/api/films]
OR
2. GET characters [http://swapi.dev/api/people]
Container
mongoDB.
Database
Container
node JS
Web App
Host OS

Favorite Movies Web App Example

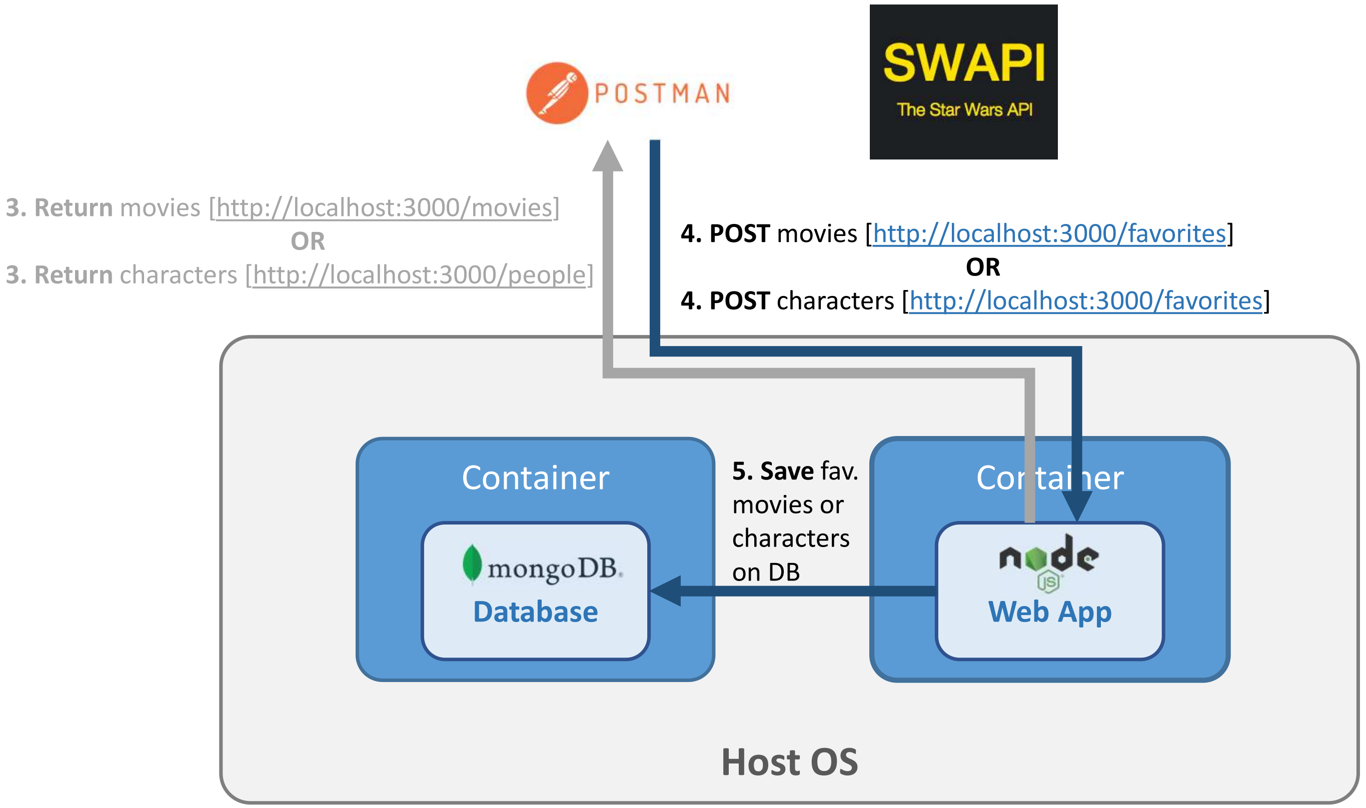
Workflow – Movies and Characters are Delivered to POSTMAN



POSTMAN
SWAPI
The Star Wars API
3. Return movies [http://localhost:3000/movies]
OR
3. Return characters [http://localhost:3000/people]
2. GET movies [http://swapi.dev/api/films]
OR
2. GET characters [http://swapi.dev/api/people]
Container
mongoDB.
Database
Container
node JS
Web App
Host OS

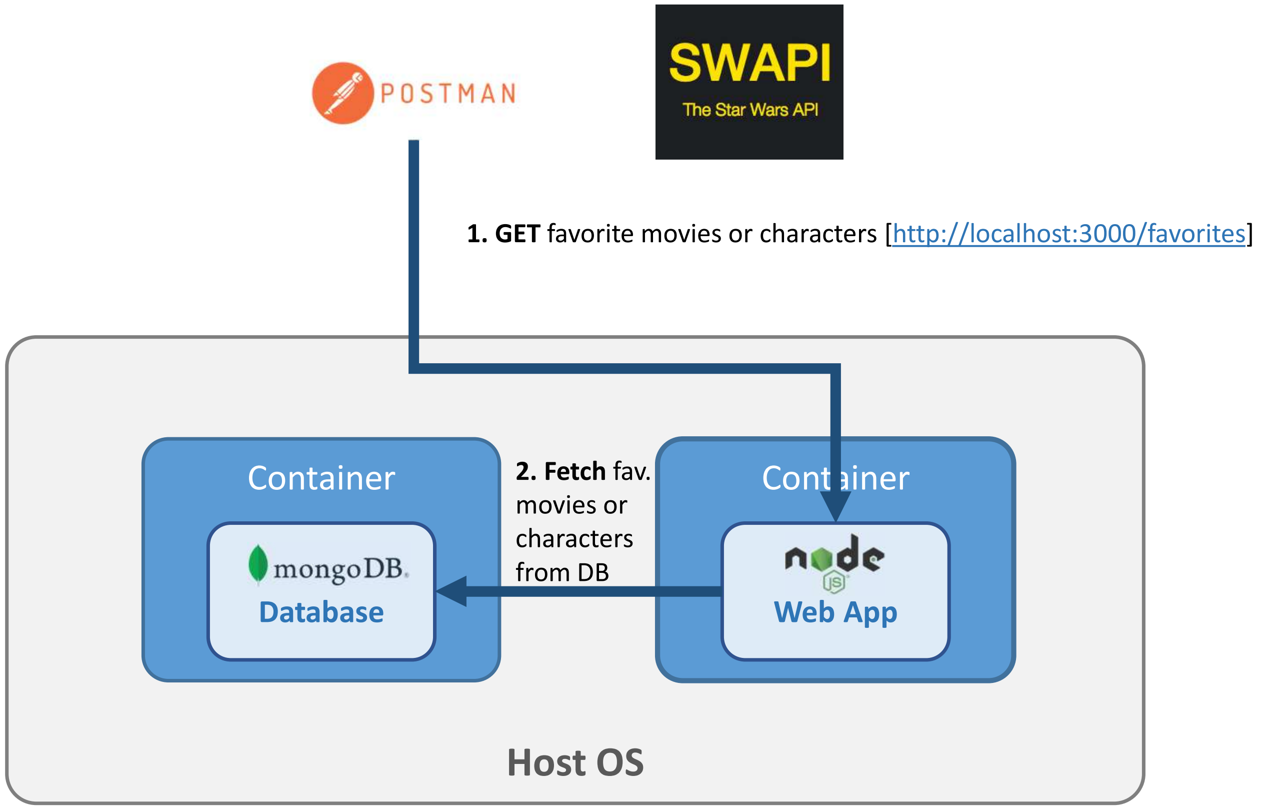
Favorite Movies Web App Example

Workflow – HTTP POST Favorite Movies and Characters in DB


POSTMAN
SWAPI
The Star Wars API
3. Return movies [http://localhost:3000/movies]
OR
3. Return characters [http://localhost:3000/people]
4. POST movies [http://localhost:3000/favorites]
OR
4. POST characters [http://localhost:3000/favorites]
Container
mongoDB.
Database
5. Save fav. movies or characters on DB
Container
node JS
Web App
Host OS

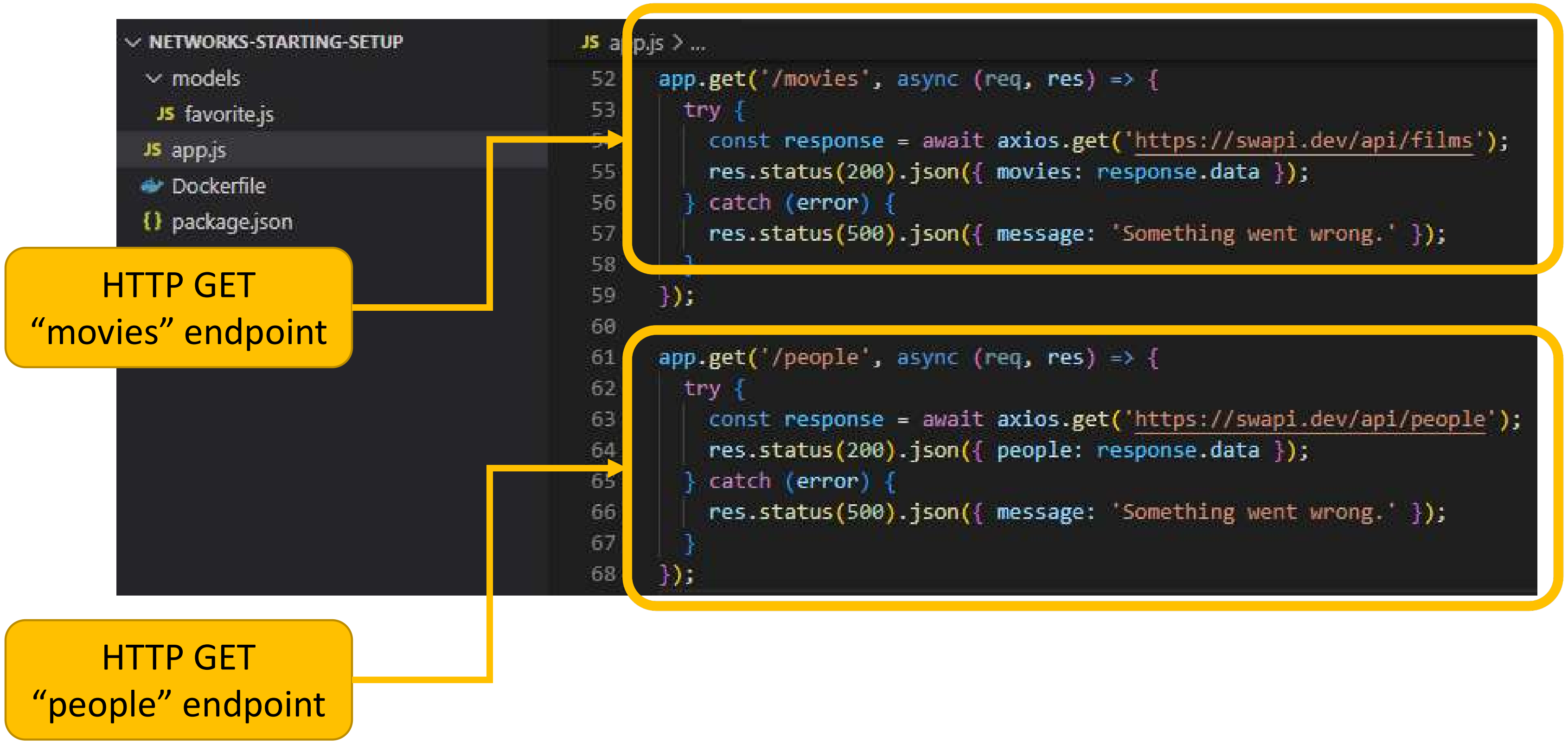
Favorite Movies Web App Example

Workflow – HTTP GET Favorite Movies and Characters in DB


POSTMAN
SWAPI
The Star Wars API
1. GET favorite movies or characters [http://localhost:3000/favorites]
Container
mongoDB.
Database
2. Fetch fav. movies or characters from DB
Container
node JS
Web App
Host OS

Favorite Movies Web App Example

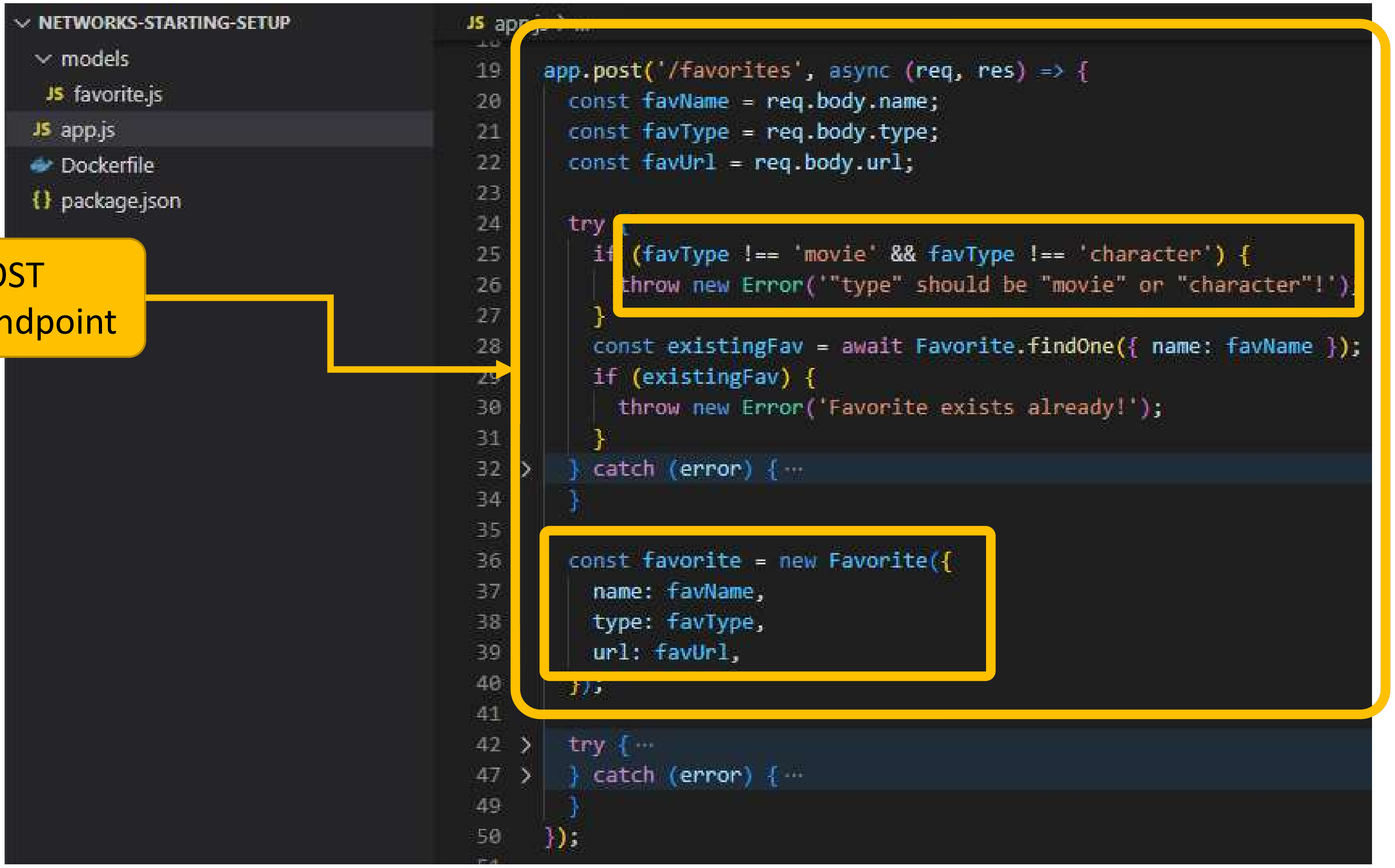
HTTP GET Endpoints – Movies and People



```
app.get('/movies', async (req, res) => {
  try {
    const response = await axios.get('https://swapi.dev/api/films');
    res.status(200).json({ movies: response.data });
  } catch (error) {
    res.status(500).json({ message: 'Something went wrong.' });
  }
});

app.get('/people', async (req, res) => {
  try {
    const response = await axios.get('https://swapi.dev/api/people');
    res.status(200).json({ people: response.data });
  } catch (error) {
    res.status(500).json({ message: 'Something went wrong.' });
  }
});
```

Favorite Movies Web App Example

HTTP POST Endpoint – Favorites

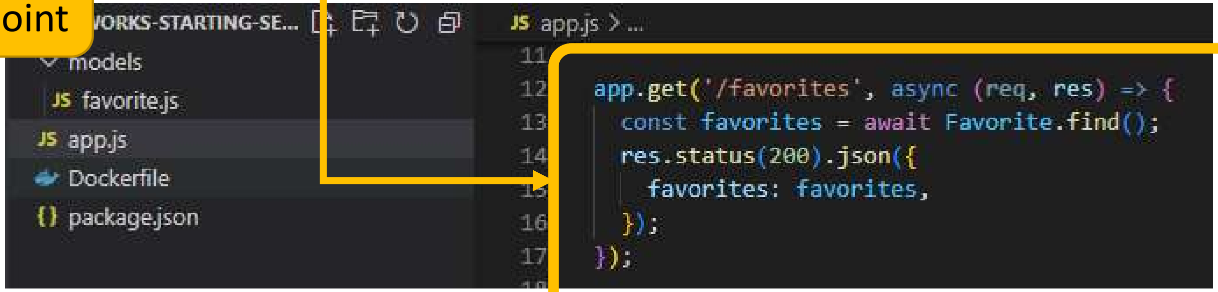


```
app.post('/favorites', async (req, res) => {
  const favName = req.body.name;
  const favType = req.body.type;
  const favUrl = req.body.url;

  try {
    if (favType !== 'movie' && favType !== 'character') {
      throw new Error('"type" should be "movie" or "character"!');
    }
    const existingFav = await Favorite.findOne({ name: favName });
    if (existingFav) {
      throw new Error('Favorite exists already!');
    }
  } catch (error) {⋯
  }

  const favorite = new Favorite({
    name: favName,
    type: favType,
    url: favUrl,
  });

  try {⋯
  } catch (error) {⋯
  }
});
```

Favorite Movies Web App Example

HTTP GET Endpoint – Favorites

HTTP GET
“favorites” endpoint



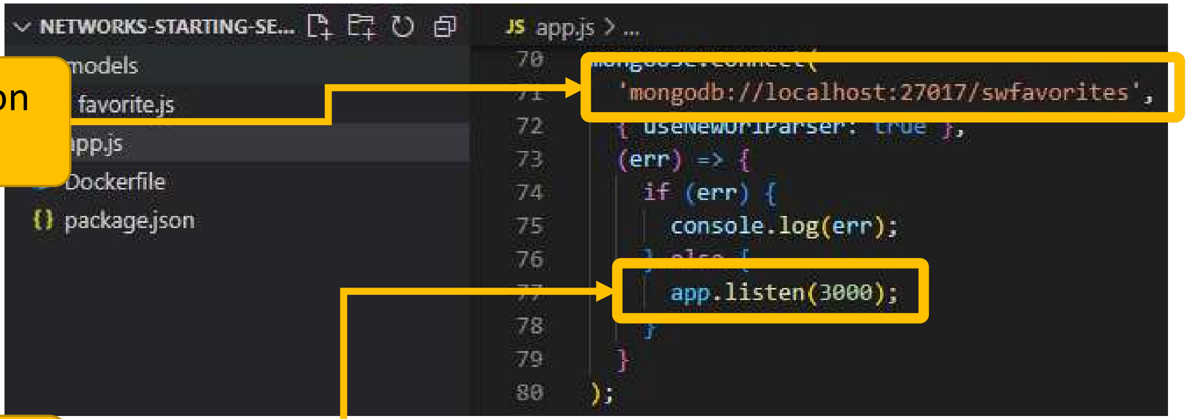
```
app.get('/favorites', async (req, res) => {
  const favorites = await Favorite.find();
  res.status(200).json({
    favorites: favorites,
  });
});
```

Favorite Movies Web App Example

HTTP GET Endpoint – Favorites

MongoDB connection address

Web app listens on port 3000



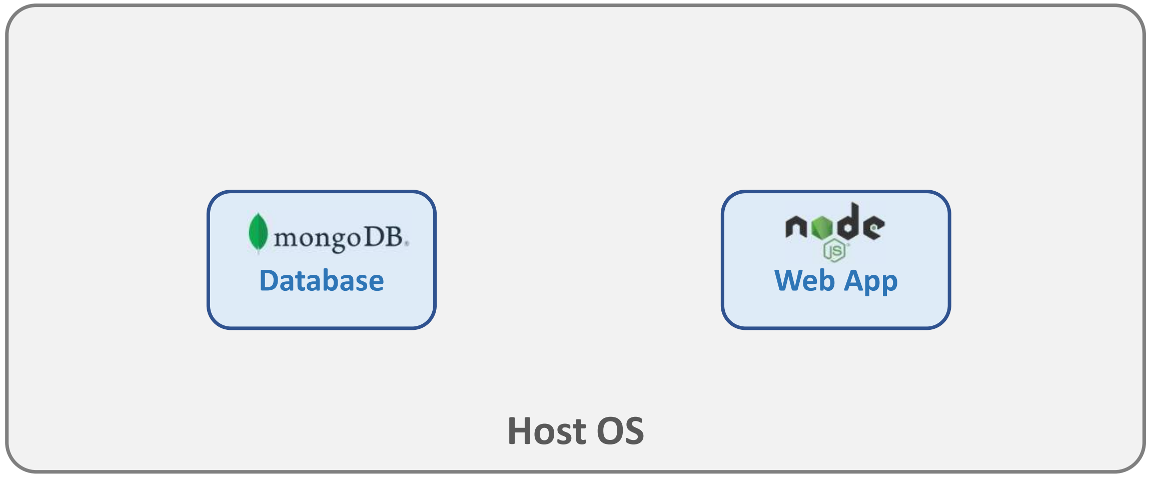
```
mongoose.connect(
  'mongodb://localhost:27017/swfavorites',
  { useNewUrlParser: true },
  (err) => {
    if (err) {
      console.log(err);
    } else {
      app.listen(3000);
    }
  }
);
```

Step 2

Running the App **without** containers

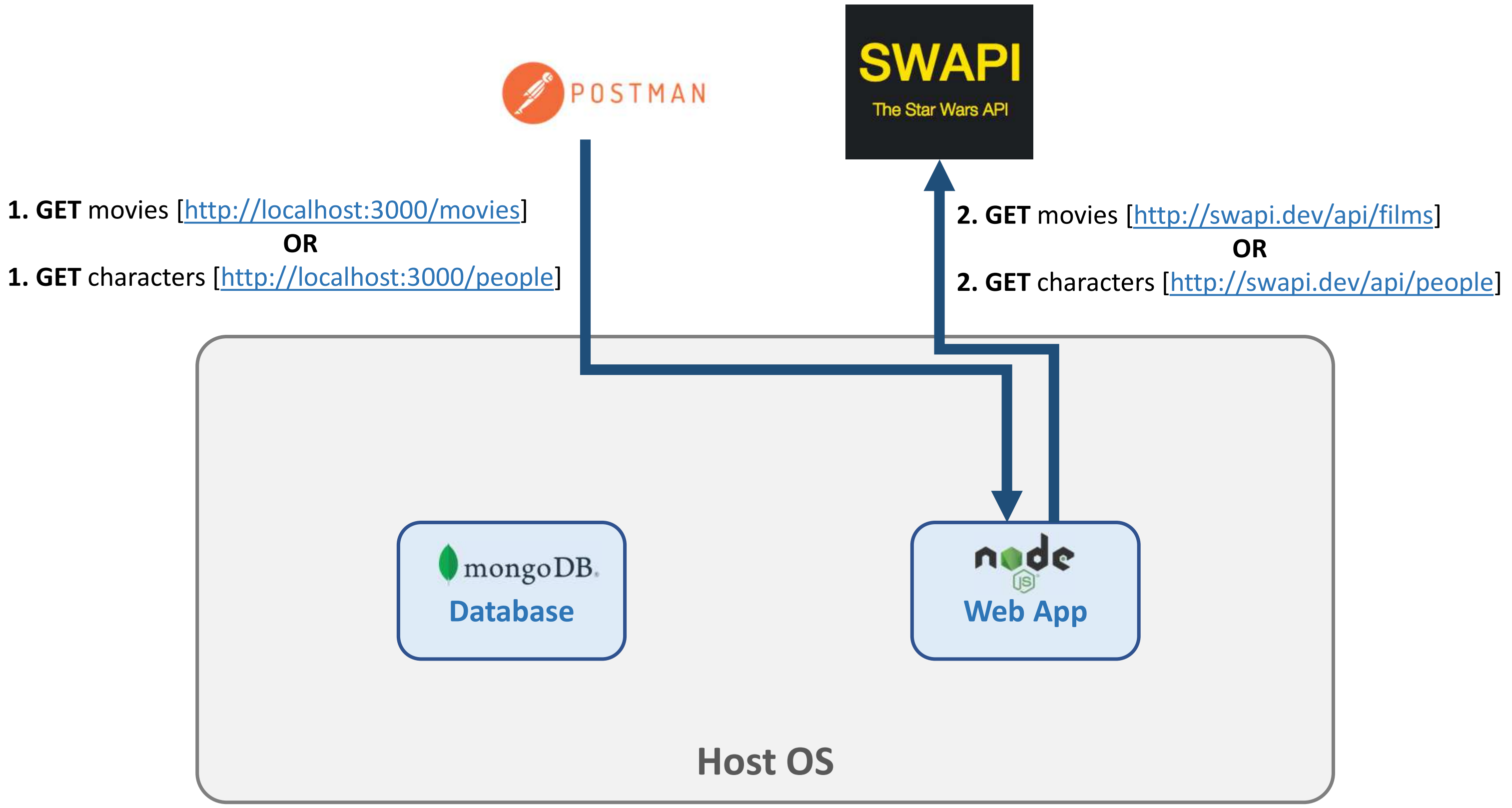
Favorite Movies Web App Example

Running the App Without Containers



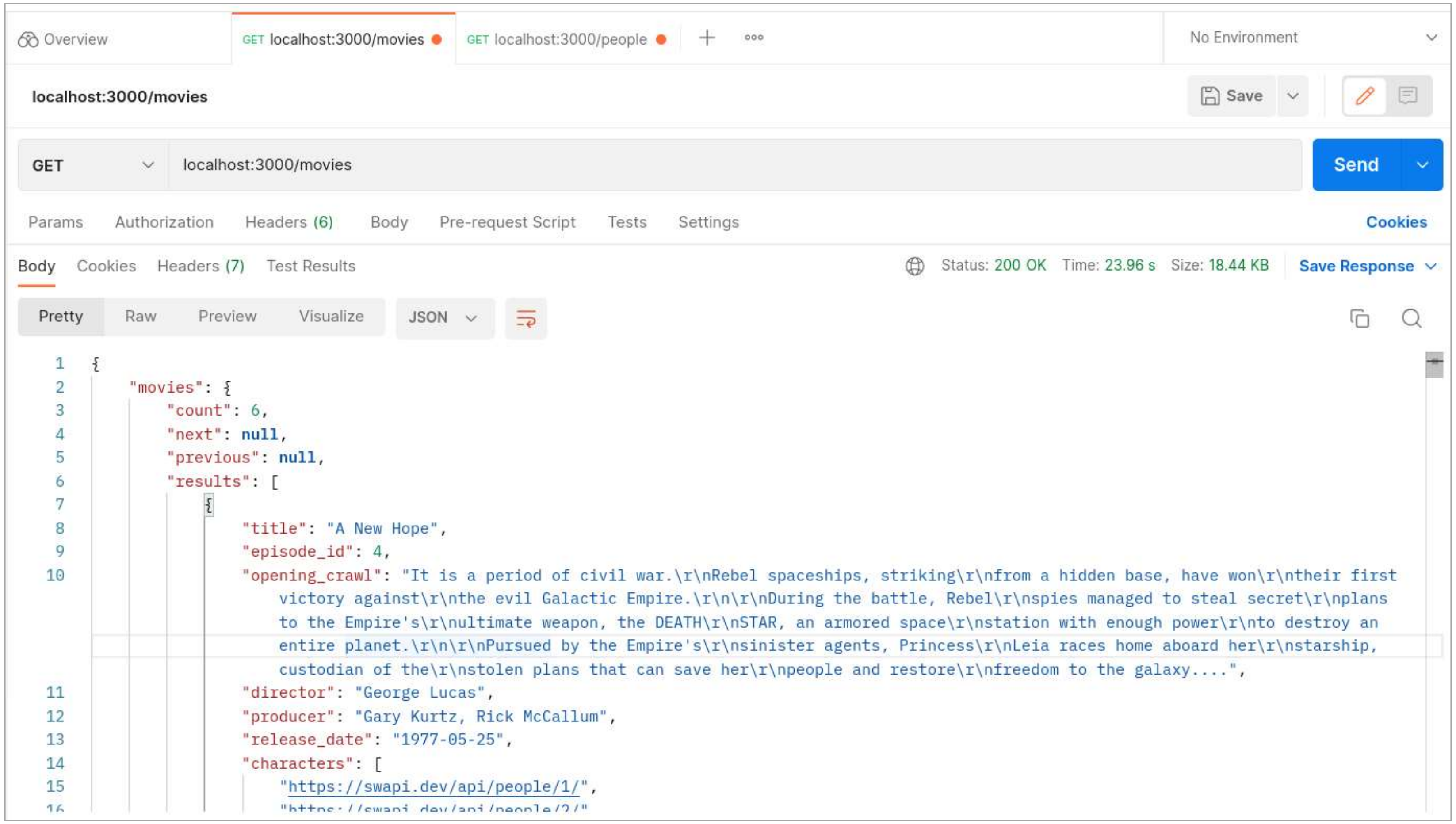
Favorite Movies Web App Example

Running the App Without Containers – HTTP GET Movies/Characters


POSTMAN
SWAPI
The Star Wars API
1. GET movies [http://localhost:3000/movies]
OR
1. GET characters [http://localhost:3000/people]
2. GET movies [http://swapi.dev/api/films]
OR
2. GET characters [http://swapi.dev/api/people]
mongoDB
Database
node JS
Web App
Host OS

Favorite Movies Web App Example

Running the App Without Containers – HTTP GET Movies



localhost:3000/movies

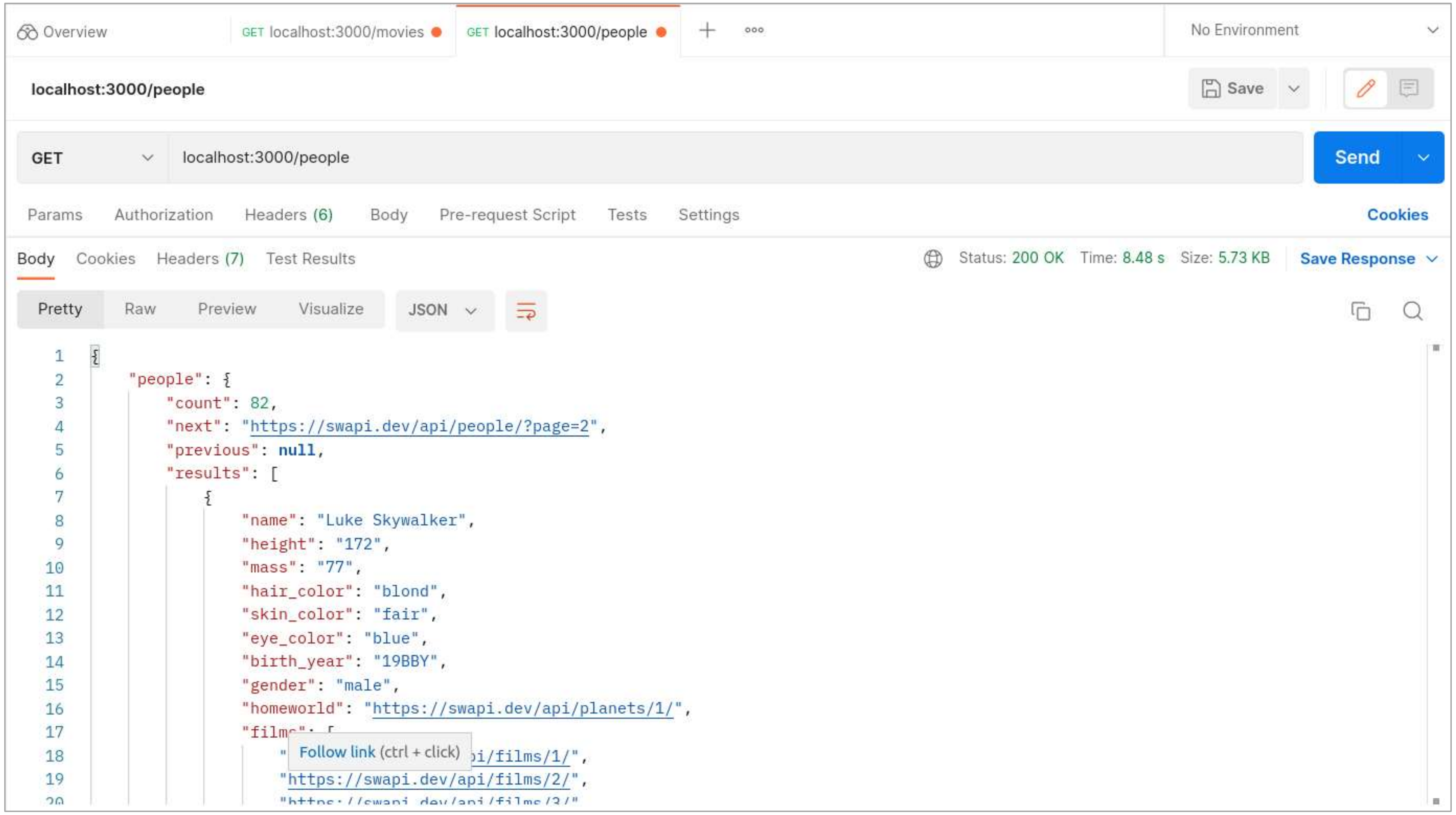
GET localhost:3000/movies

Status: 200 OK Time: 23.96 s Size: 18.44 KB

```
{
    "movies": {
        "count": 6,
        "next": null,
        "previous": null,
        "results": [
            {
                "title": "A New Hope",
                "episode_id": 4,
                "opening_crawl": "It is a period of civil war.\r\nRebel spaceships, striking\r\nfrom a hidden base, have won\r\ntheir first
                    victory against\r\nthe evil Galactic Empire.\r\n\r\nDuring the battle, Rebel\r\nspies managed to steal secret\r\nplans
                    to the Empire's\r\nultimate weapon, the DEATH\r\nSTAR, an armored space\r\nstation with enough power\r\nto destroy an
                    entire planet.\r\n\r\nPursued by the Empire's\r\nsinister agents, Princess\r\nLeia races home aboard her\r\nstarship,
                    custodian of the\r\nstolen plans that can save her\r\npeople and restore\r\nfreedom to the galaxy....",
                "director": "George Lucas",
                "producer": "Gary Kurtz, Rick McCallum",
                "release_date": "1977-05-25",
                "characters": [
                    "https://swapi.dev/api/people/1/",
                    "https://swapi.dev/api/people/2/"
```

Favorite Movies Web App Example

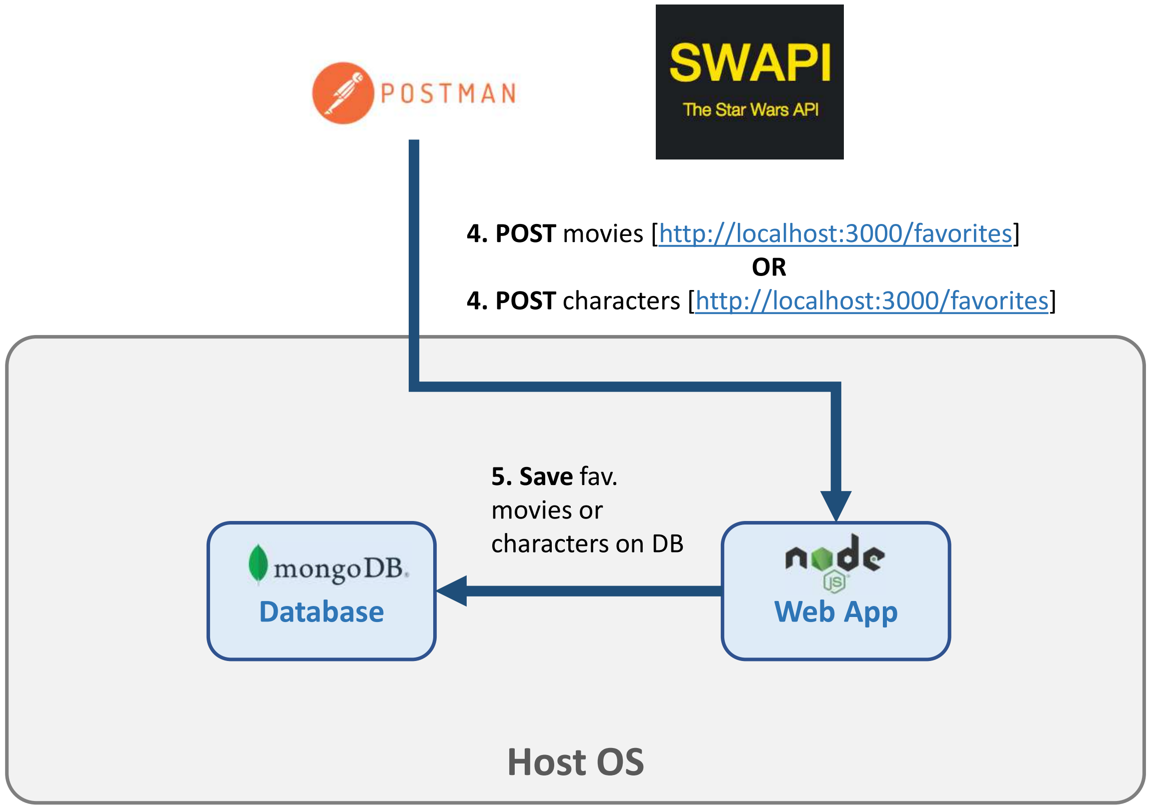
Running the App Without Containers – HTTP GET People



```
{
    "people": {
        "count": 82,
        "next": "https://swapi.dev/api/people/?page=2",
        "previous": null,
        "results": [
            {
                "name": "Luke Skywalker",
                "height": "172",
                "mass": "77",
                "hair_color": "blond",
                "skin_color": "fair",
                "eye_color": "blue",
                "birth_year": "19BBY",
                "gender": "male",
                "homeworld": "https://swapi.dev/api/planets/1/",
                "films": [
                    "                         i/films/1/",
                    "https://swapi.dev/api/films/2/",
```

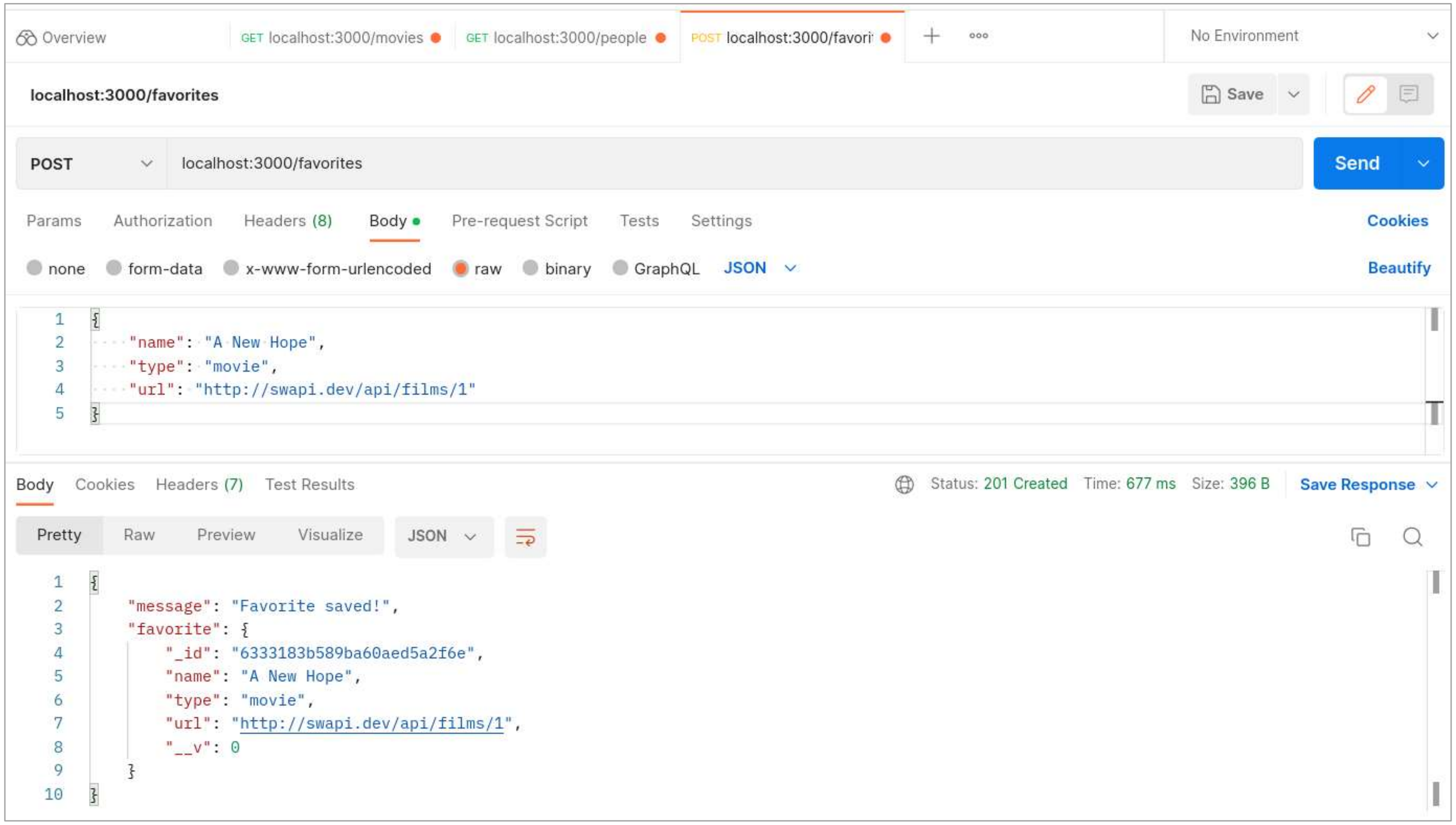
Favorite Movies Web App Example

Running the App Without Containers – HTTP POST Favorite Movies


POSTMAN
SWAPI
The Star Wars API
4. POST movies [http://localhost:3000/favorites]
OR
4. POST characters [http://localhost:3000/favorites]
5. Save fav. movies or characters on DB
mongoDB
Database
node js
Web App
Host OS

Favorite Movies Web App Example

Running the App Without Containers – HTTP POST Favorite Movies



localhost:3000/favorites

POST localhost:3000/favorites

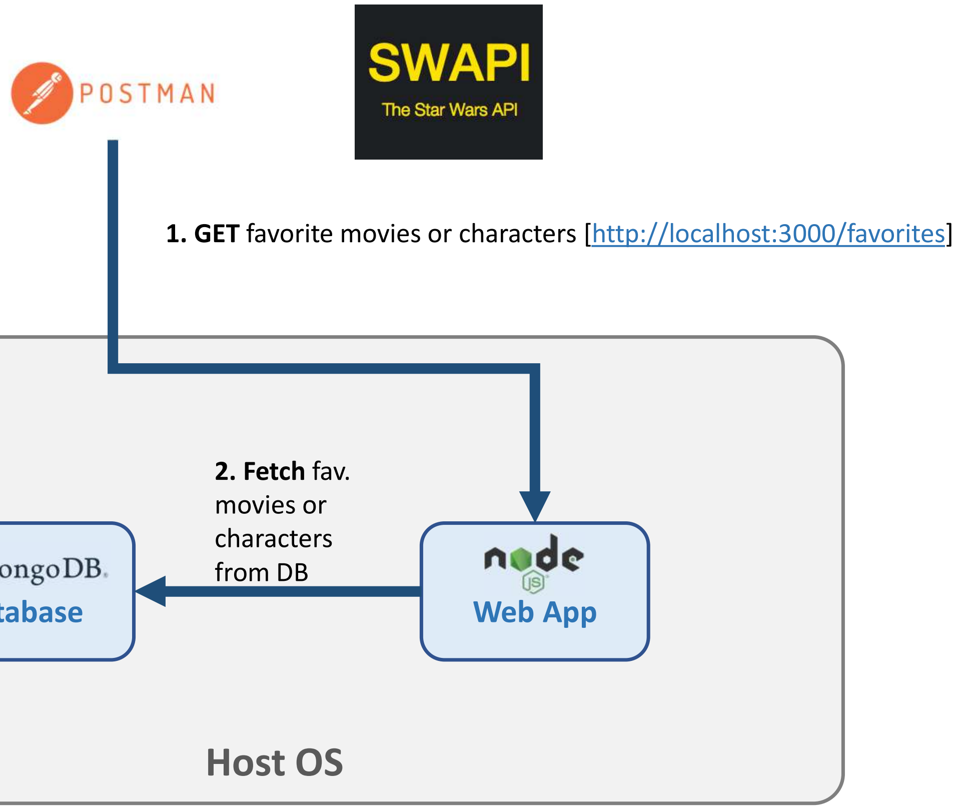
```
{
    "name": "A New Hope",
    "type": "movie",
    "url": "http://swapi.dev/api/films/1"
}
```

Status: 201 Created Time: 677 ms Size: 396 B

```
{
    "message": "Favorite saved!",
    "favorite": {
        "_id": "6333183b589ba60aed5a2f6e",
        "name": "A New Hope",
        "type": "movie",
        "url": "http://swapi.dev/api/films/1",
        "__v": 0
    }
}
```

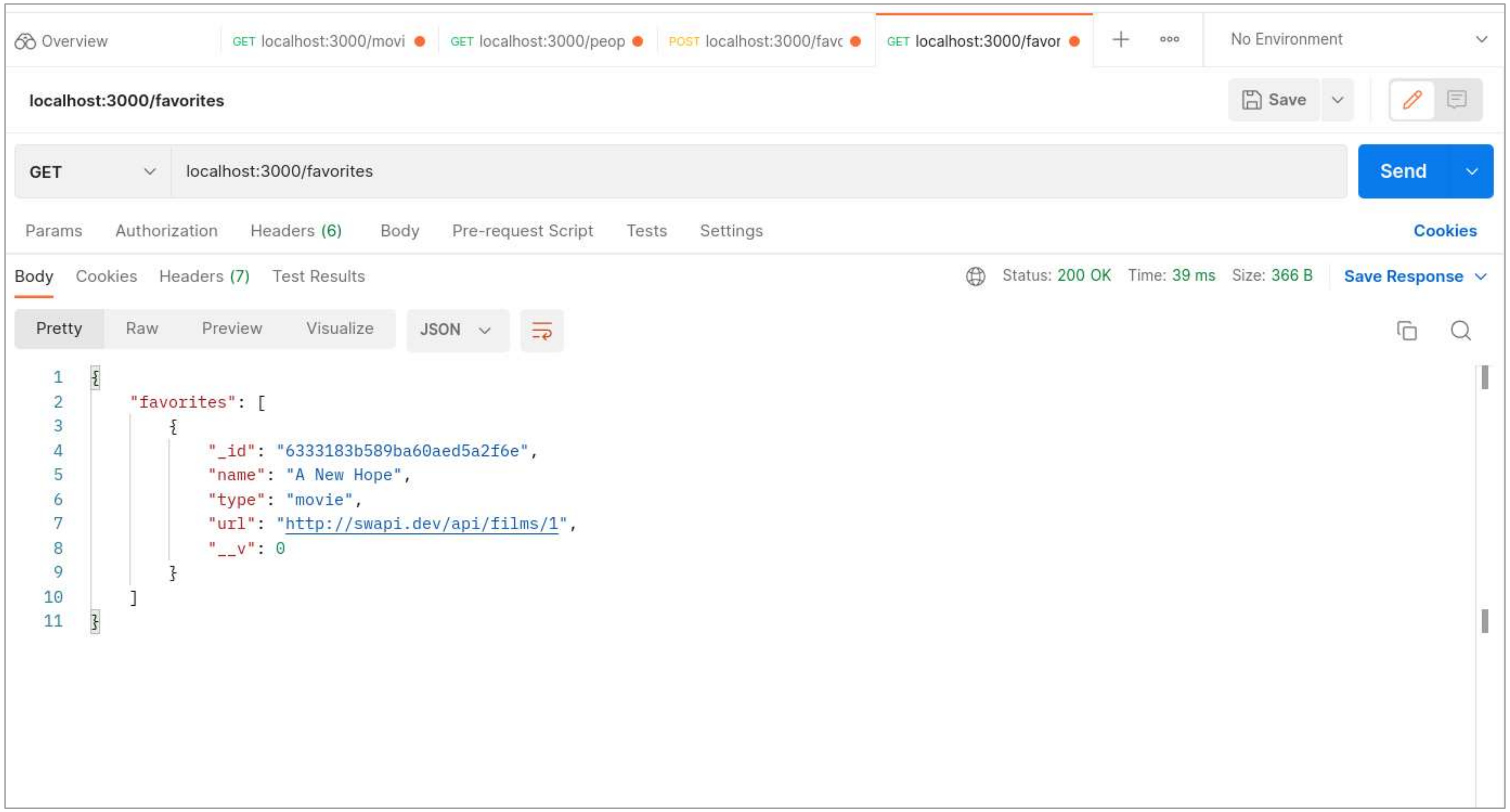
Favorite Movies Web App Example

Running the App Without Containers – HTTP GET Favorite Movies and Characters from DB


POSTMAN
SWAPI
The Star Wars API
1. GET favorite movies or characters [http://localhost:3000/favorites]
2. Fetch fav. movies or characters from DB
mongoDB.
Database
node JS
Web App
Host OS

Favorite Movies Web App Example

Running the App Without Containers – HTTP GET Favorite Movies and Characters from DB



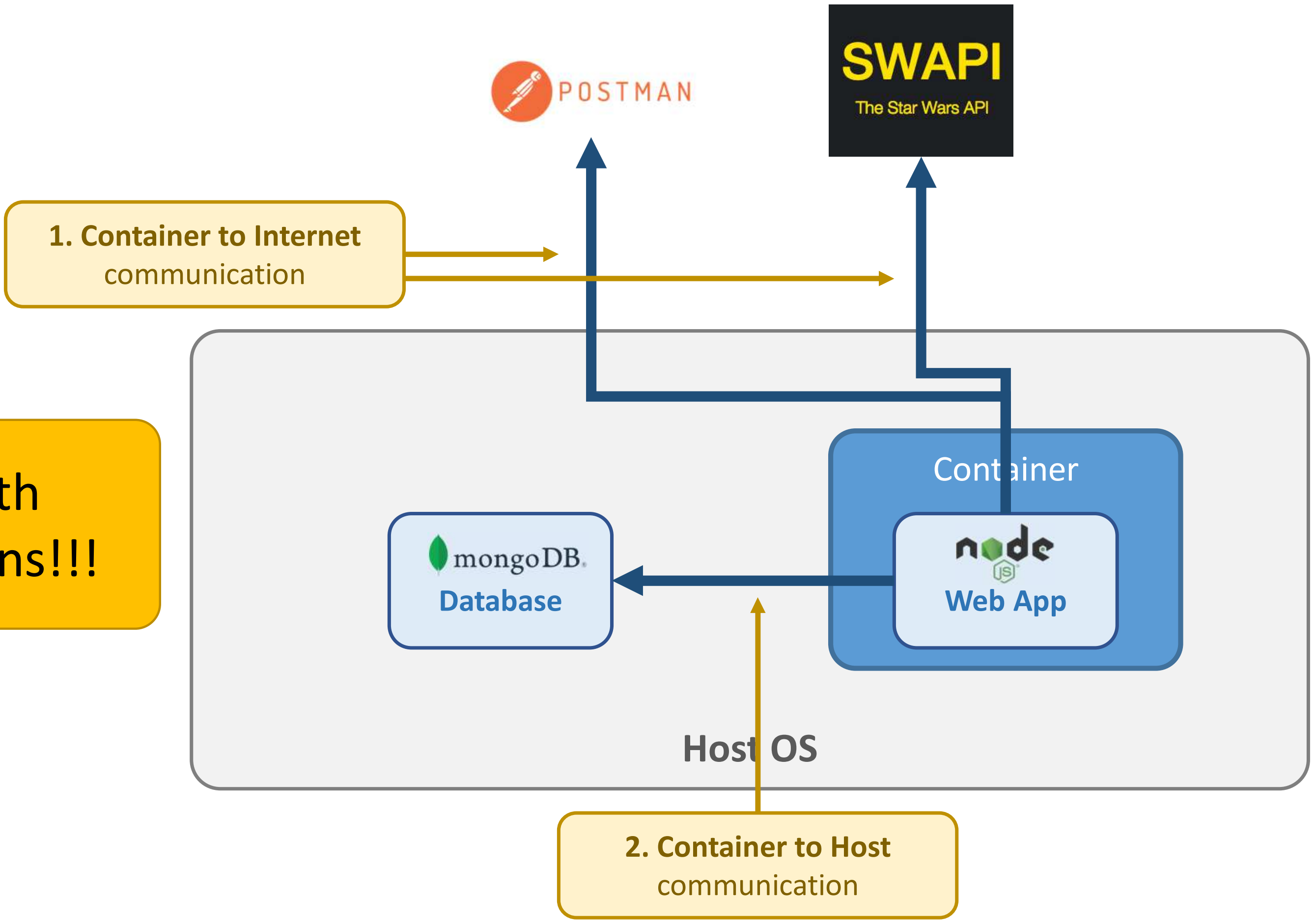
```
{
    "favorites": [
        {
            "_id": "6333183b589ba60aed5a2f6e",
            "name": "A New Hope",
            "type": "movie",
            "url": "http://swapi.dev/api/films/1",
            "__v": 0
        }
    ]
}
```

Step 3

Running the App **with** containers and non-containers

Favorite Movies Web App Example

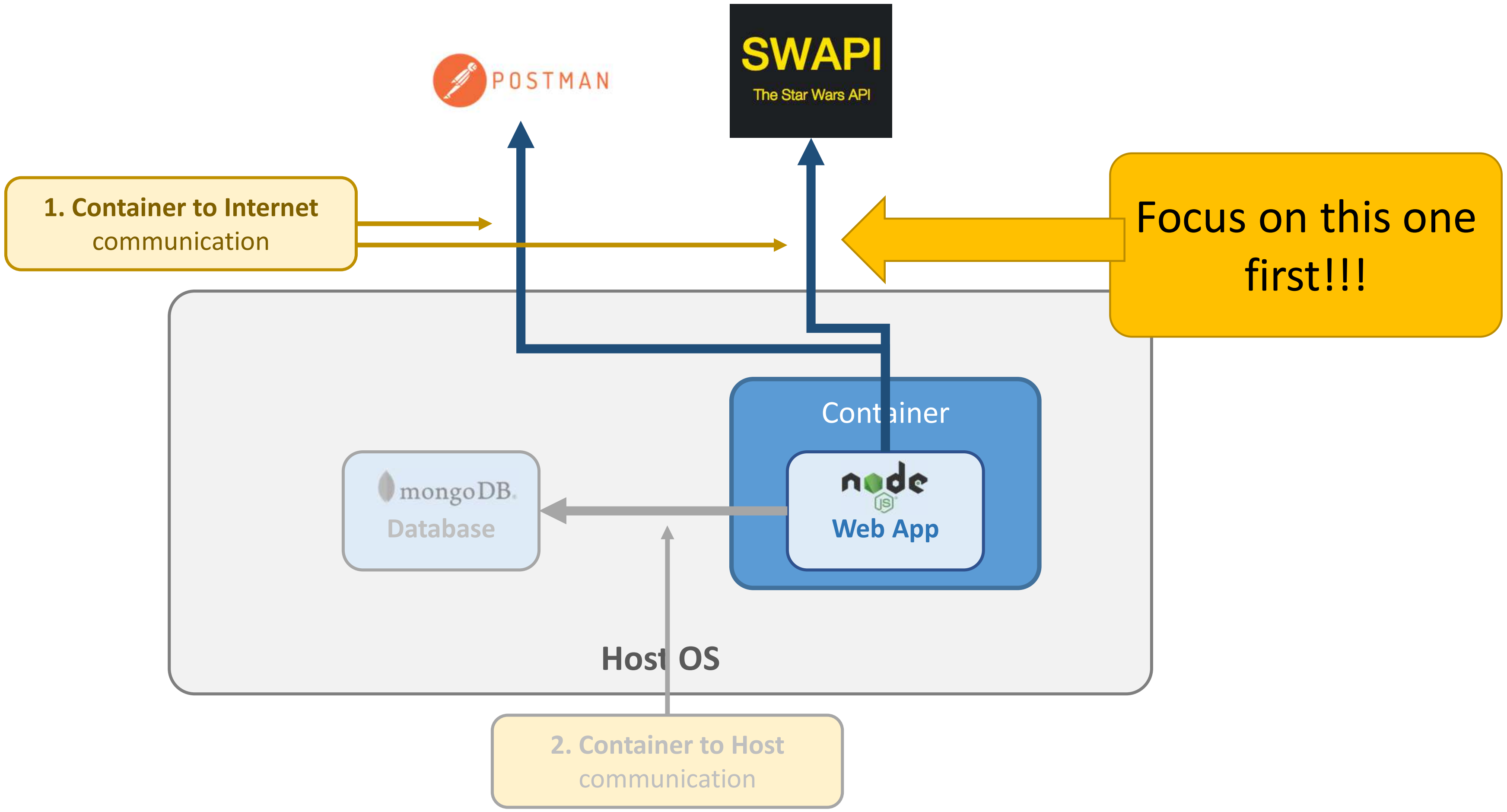
Running the App With Containers and non-Containers



Test both connections!!!

Favorite Movies Web App Example

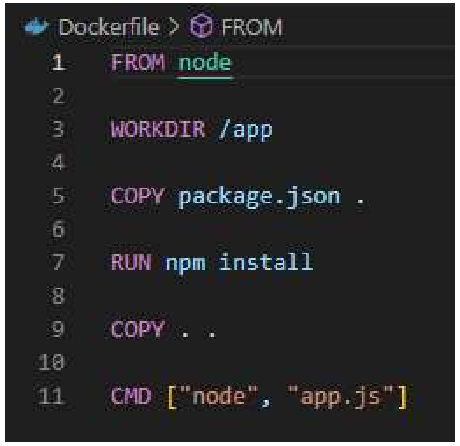
Running the App With Containers and non-Containers



Favorite Movies Web App Example

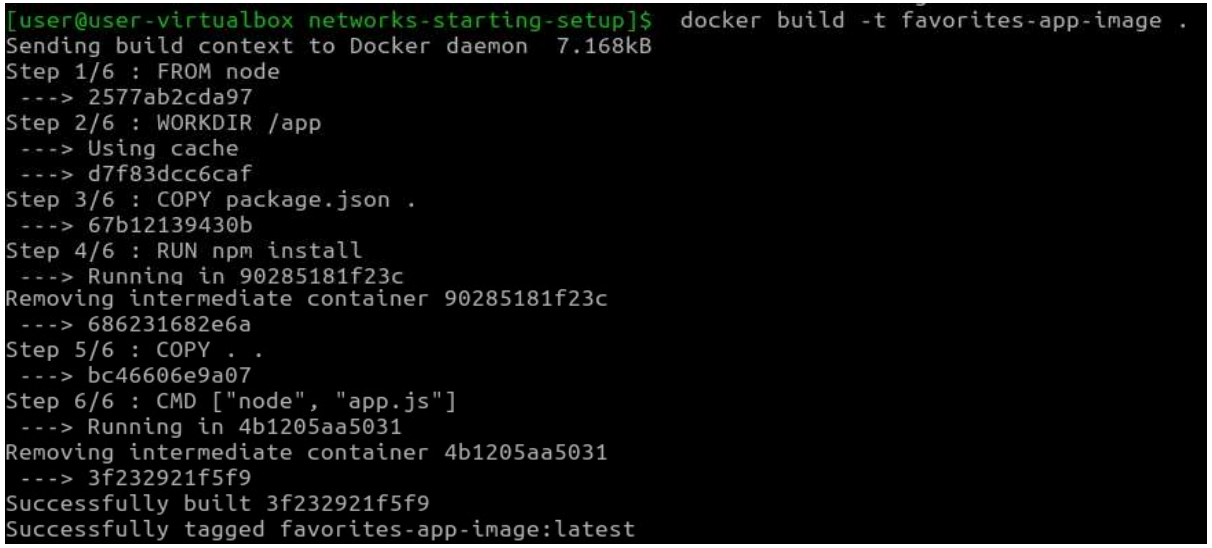
Container to Internet Communication – Dockerize and Build the Image

1. Create Dockerfile



```
FROM node

WORKDIR /app

COPY package.json .

RUN npm install

COPY . .

CMD ["node", "app.js"]
```

2. Build Image

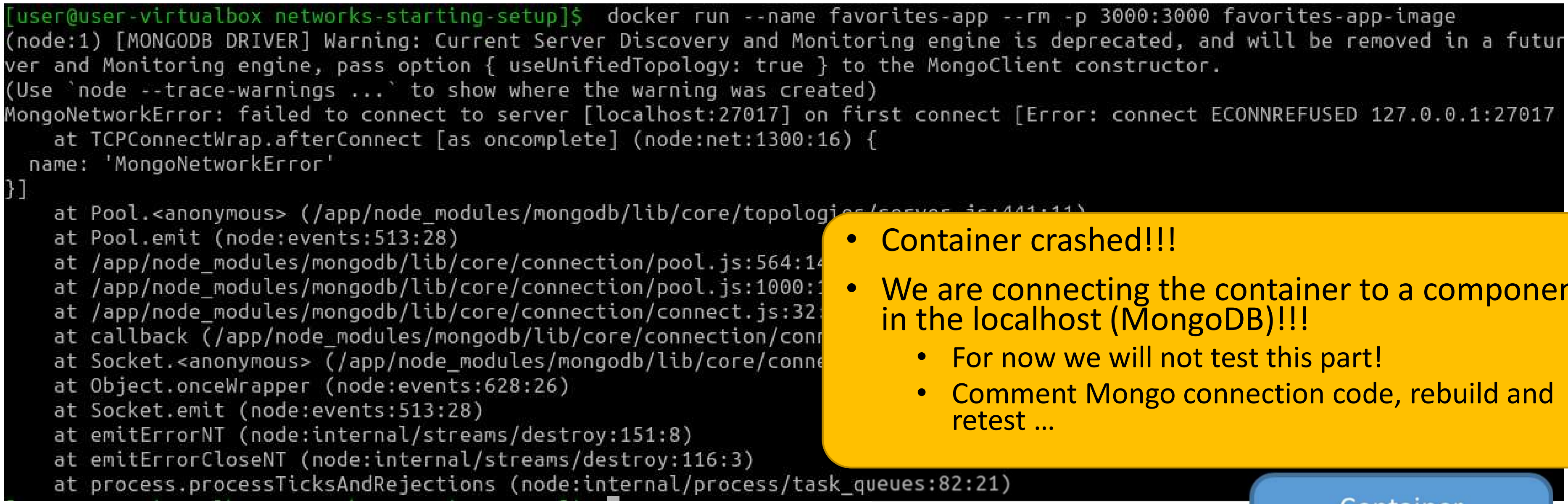


```
[user@user-virtualbox networks-starting-setup]$  docker build -t favorites-app-image .
Sending build context to Docker daemon  7.168kB
Step 1/6 : FROM node
 ---> 2577ab2cda97
Step 2/6 : WORKDIR /app
 ---> Using cache
 ---> d7f83dcc6caf
Step 3/6 : COPY package.json .
 ---> 67b12139430b
Step 4/6 : RUN npm install
 ---> Running in 90285181f23c
Removing intermediate container 90285181f23c
 ---> 686231682e6a
Step 5/6 : COPY . .
 ---> bc46606e9a07
Step 6/6 : CMD ["node", "app.js"]
 ---> Running in 4b1205aa5031
Removing intermediate container 4b1205aa5031
 ---> 3f232921f5f9
Successfully built 3f232921f5f9
Successfully tagged favorites-app-image:latest
```

Favorite Movies Web App Example

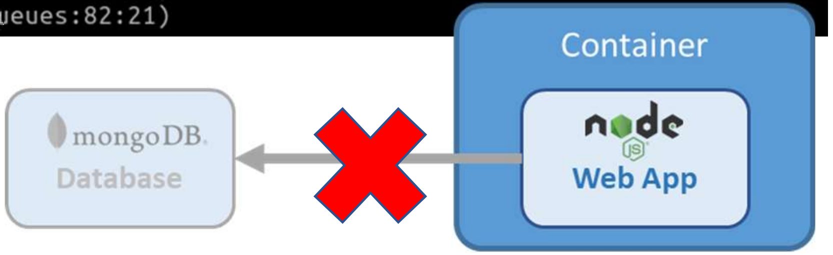
Container to Internet Communication – Run the Container

3. Run Web App Container



```
[user@user-virtualbox networks-starting-setup]$  docker run --name favorites-app --rm -p 3000:3000 favorites-app-image
(node:1) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a futur
ver and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
(Use `node --trace-warnings ...` to show where the warning was created)
MongoNetworkError: failed to connect to server [localhost:27017] on first connect [Error: connect ECONNREFUSED 127.0.0.1:27017
    at TCPConnectWrap.afterConnect [as oncomplete] (node:net:1300:16) {
  name: 'MongoNetworkError'
}]
    at Pool.<anonymous> (/app/node_modules/mongodb/lib/core/topologi
    at Pool.emit (node:events:513:28)
    at /app/node_modules/mongodb/lib/core/connection/pool.js:564:14
    at /app/node_modules/mongodb/lib/core/connection/pool.js:1000:1
    at /app/node_modules/mongodb/lib/core/connection/connect.js:32
    at callback (/app/node_modules/mongodb/lib/core/connection/conn
    at Socket.<anonymous> (/app/node_modules/mongodb/lib/core/conne
    at Object.onceWrapper (node:events:628:26)
    at Socket.emit (node:events:513:28)
    at emitErrorNT (node:internal/streams/destroy:151:8)
    at emitErrorCloseNT (node:internal/streams/destroy:116:3)
    at process.processTicksAndRejections (node:internal/process/task_queues:82:21)
```

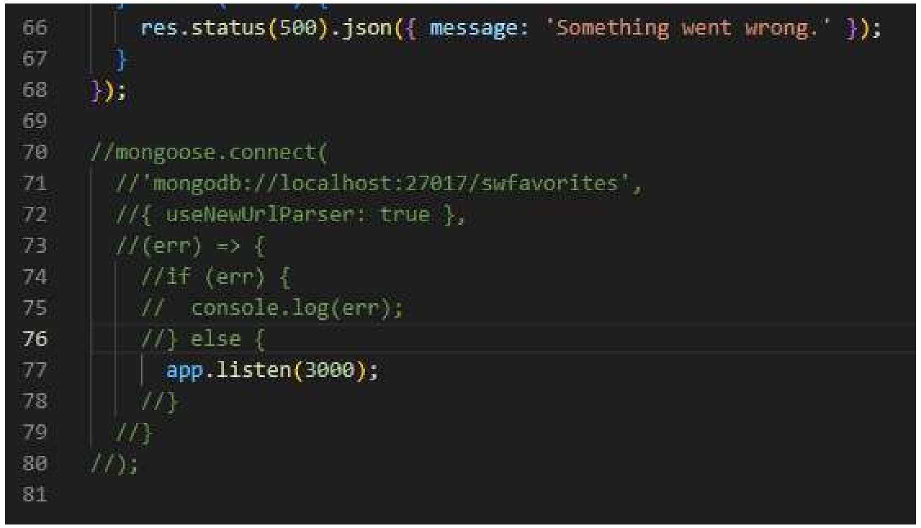
- Container crashed!!!
- We are connecting the container to a component in the localhost (MongoDB)!!!
  - For now we will not test this part!
  - Comment Mongo connection code, rebuild and retest …



Favorite Movies Web App Example

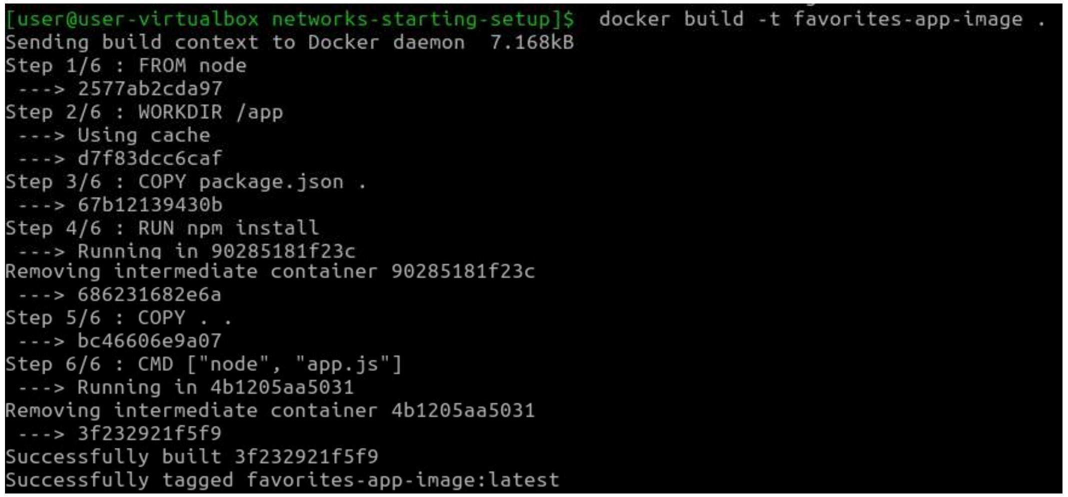
Container to Internet Communication – Comment Mongo Connection, Build Image and Run Container

1. Comment Mongo Connection Code



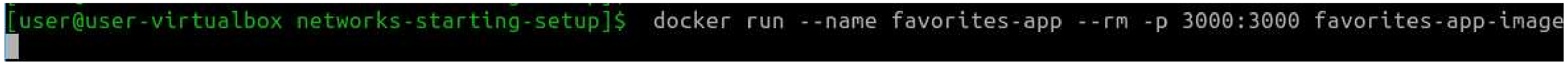
```
    res.status(500).json({ message: 'Something went wrong.' });
  }
});

//mongoose.connect(
  //'mongodb://localhost:27017/swfavorites',
  //{ useNewUrlParser: true },
  //(err) => {
    //if (err) {
    //  console.log(err);
    //} else {
      app.listen(3000);
    //}
  //}
//);

```

2. Build Image



```
[user@user-virtualbox networks-starting-setup]$  docker build -t favorites-app-image .
Sending build context to Docker daemon  7.168kB
Step 1/6 : FROM node
 ---> 2577ab2cda97
Step 2/6 : WORKDIR /app
 ---> Using cache
 ---> d7f83dcc6caf
Step 3/6 : COPY package.json .
 ---> 67b12139430b
Step 4/6 : RUN npm install
 ---> Running in 90285181f23c
Removing intermediate container 90285181f23c
 ---> 686231682e6a
Step 5/6 : COPY . .
 ---> bc46606e9a07
Step 6/6 : CMD ["node", "app.js"]
 ---> Running in 4b1205aa5031
Removing intermediate container 4b1205aa5031
 ---> 3f232921f5f9
Successfully built 3f232921f5f9
Successfully tagged favorites-app-image:latest
```

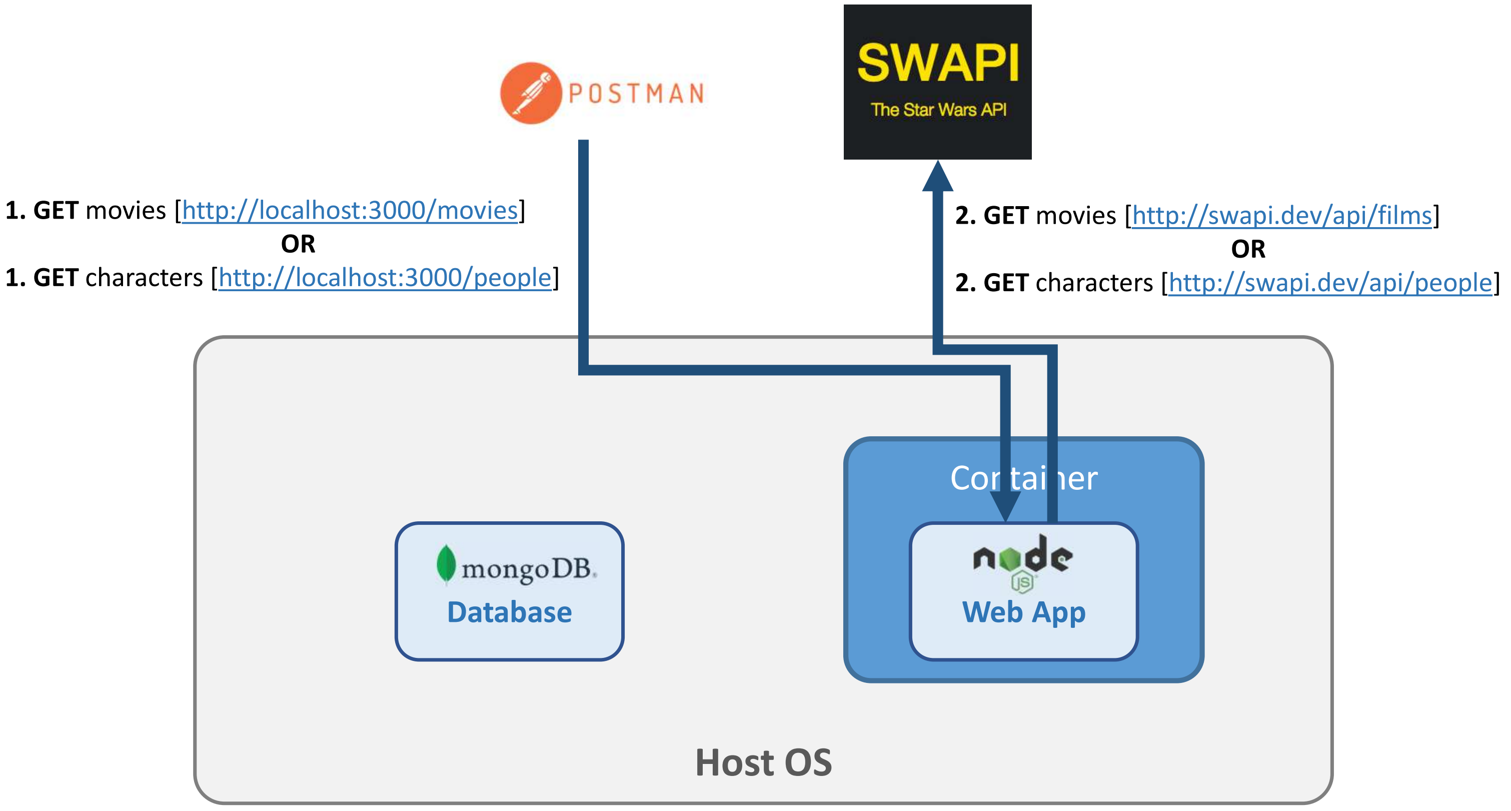
3. Run Web App Container



```
[user@user-virtualbox networks-starting-setup]$  docker run --name favorites-app --rm -p 3000:3000 favorites-app-image
```

Favorite Movies Web App Example

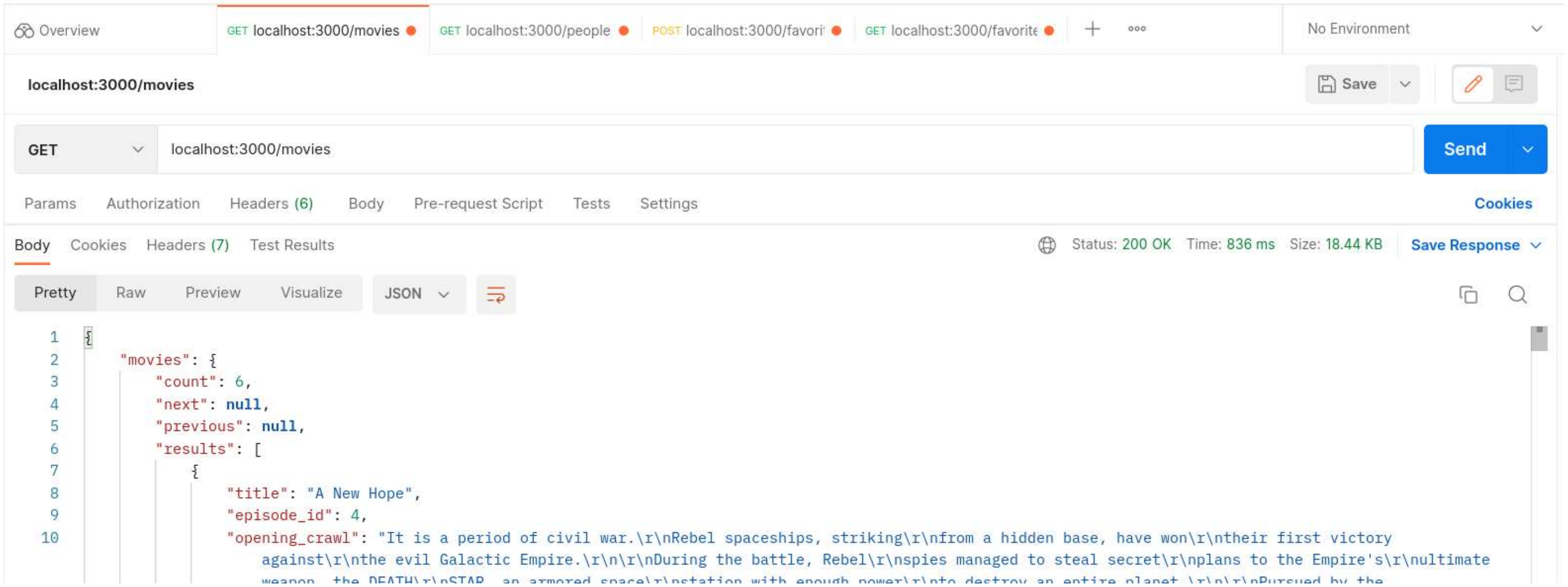
Container to Internet Communication – Test Connectivity


POSTMAN
SWAPI
The Star Wars API
1. GET movies [http://localhost:3000/movies]
OR
1. GET characters [http://localhost:3000/people]
2. GET movies [http://swapi.dev/api/films]
OR
2. GET characters [http://swapi.dev/api/people]
Container
mongoDB
Database
node JS
Web App
Host OS

Favorite Movies Web App Example

Container to Internet Communication – Test Connectivity

4. Test the GET movies endpoint



localhost:3000/movies

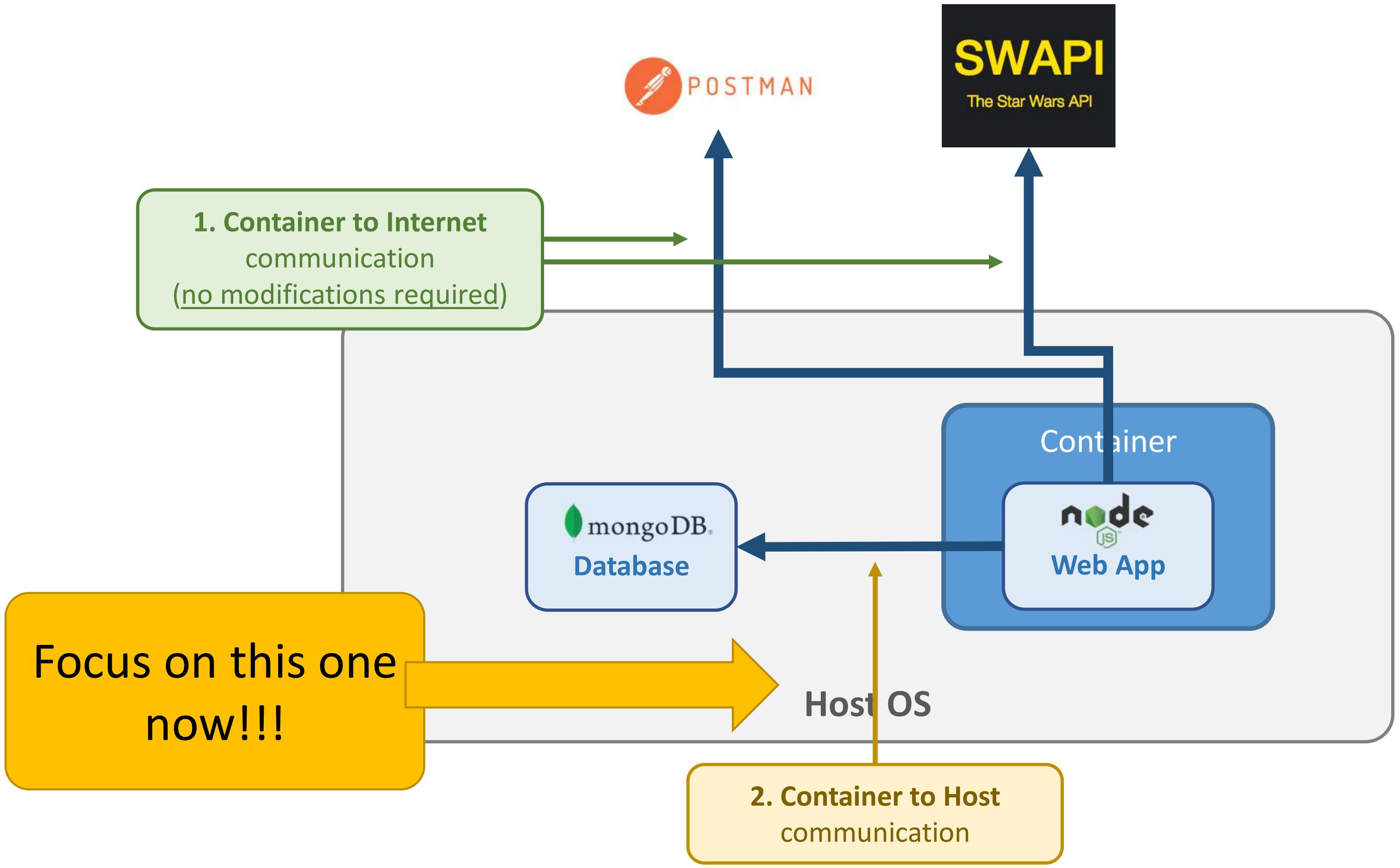
GET localhost:3000/movies

Status: 200 OK Time: 836 ms Size: 18.44 KB

```
{
    "movies": {
        "count": 6,
        "next": null,
        "previous": null,
        "results": [
            {
                "title": "A New Hope",
                "episode_id": 4,
                "opening_crawl": "It is a period of civil war.\r\nRebel spaceships, striking\r\nfrom a hidden base, have won\r\ntheir first victory
                    against\r\nthe evil Galactic Empire.\r\n\r\nDuring the battle, Rebel\r\nspies managed to steal secret\r\nplans to the Empire's\r\nultimate
                    weapon, the DEATH\r\nSTAR, an armored space\r\nstation with enough power\r\nto destroy an entire planet.\r\n\r\nPursued by the
```

Favorite Movies Web App Example

Running the App With Containers and non-Containers


POSTMAN
SWAPI
The Star Wars API
1. Container to Internet communication (no modifications required)
Container
mongoDB
Database
Web App
Focus on this one now!!!
Host OS
2. Container to Host communication

Container to Host Communication

Replace localhost domain

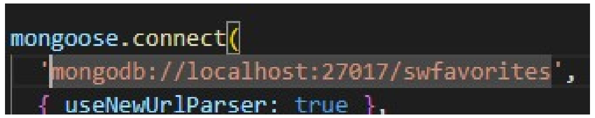
- Replace “localhost” domain in source code by the host IP address
  - Use *ifconfig* command to check the host IP address

```
‘…localhost:…’→ ‘…<host ip address>:…’
```

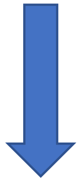
Favorite Movies Web App Example

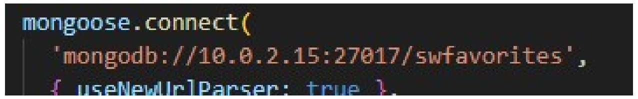
Container to Host Communication – Replace Domain and Build the Image

1. Replace localhost domain



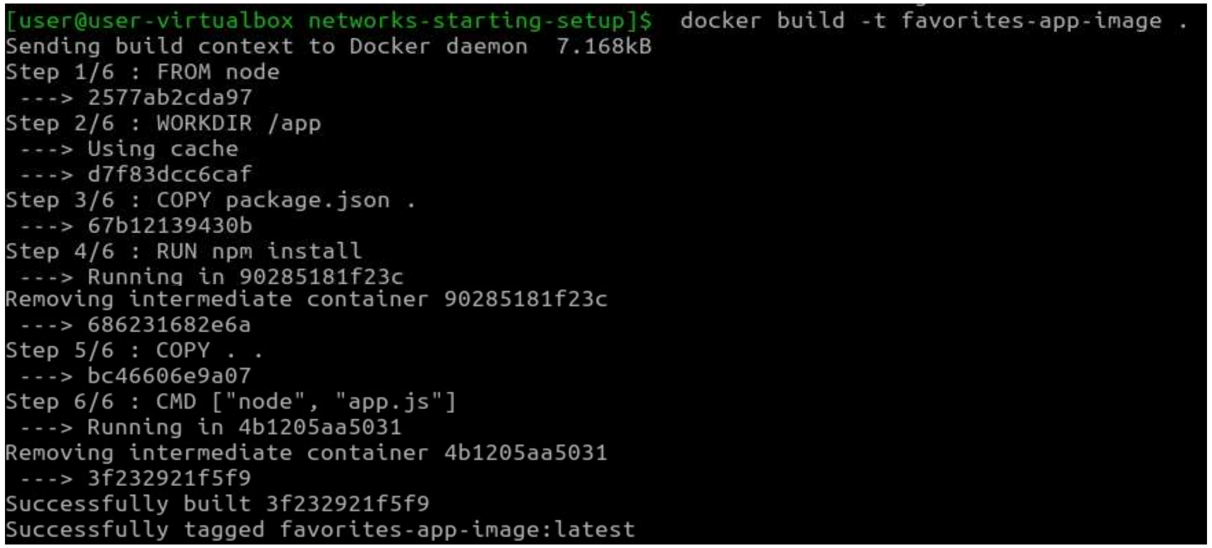
```
mongoose.connect(
  'mongodb://localhost:27017/swfavorites',
  { useNewUrlParser: true },
```





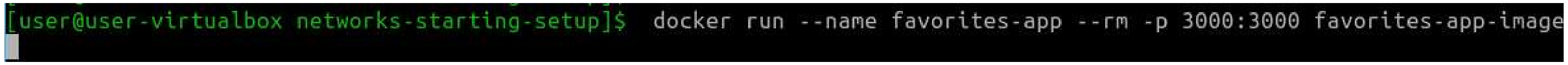
```
mongoose.connect(
  'mongodb://10.0.2.15:27017/swfavorites',
  { useNewUrlParser: true },
```

2. Build Image



```
[user@user-virtualbox networks-starting-setup]$  docker build -t favorites-app-image .
Sending build context to Docker daemon  7.168kB
Step 1/6 : FROM node
 ---> 2577ab2cda97
Step 2/6 : WORKDIR /app
 ---> Using cache
 ---> d7f83dcc6caf
Step 3/6 : COPY package.json .
 ---> 67b12139430b
Step 4/6 : RUN npm install
 ---> Running in 90285181f23c
Removing intermediate container 90285181f23c
 ---> 686231682e6a
Step 5/6 : COPY . .
 ---> bc46606e9a07
Step 6/6 : CMD ["node", "app.js"]
 ---> Running in 4b1205aa5031
Removing intermediate container 4b1205aa5031
 ---> 3f232921f5f9
Successfully built 3f232921f5f9
Successfully tagged favorites-app-image:latest
```

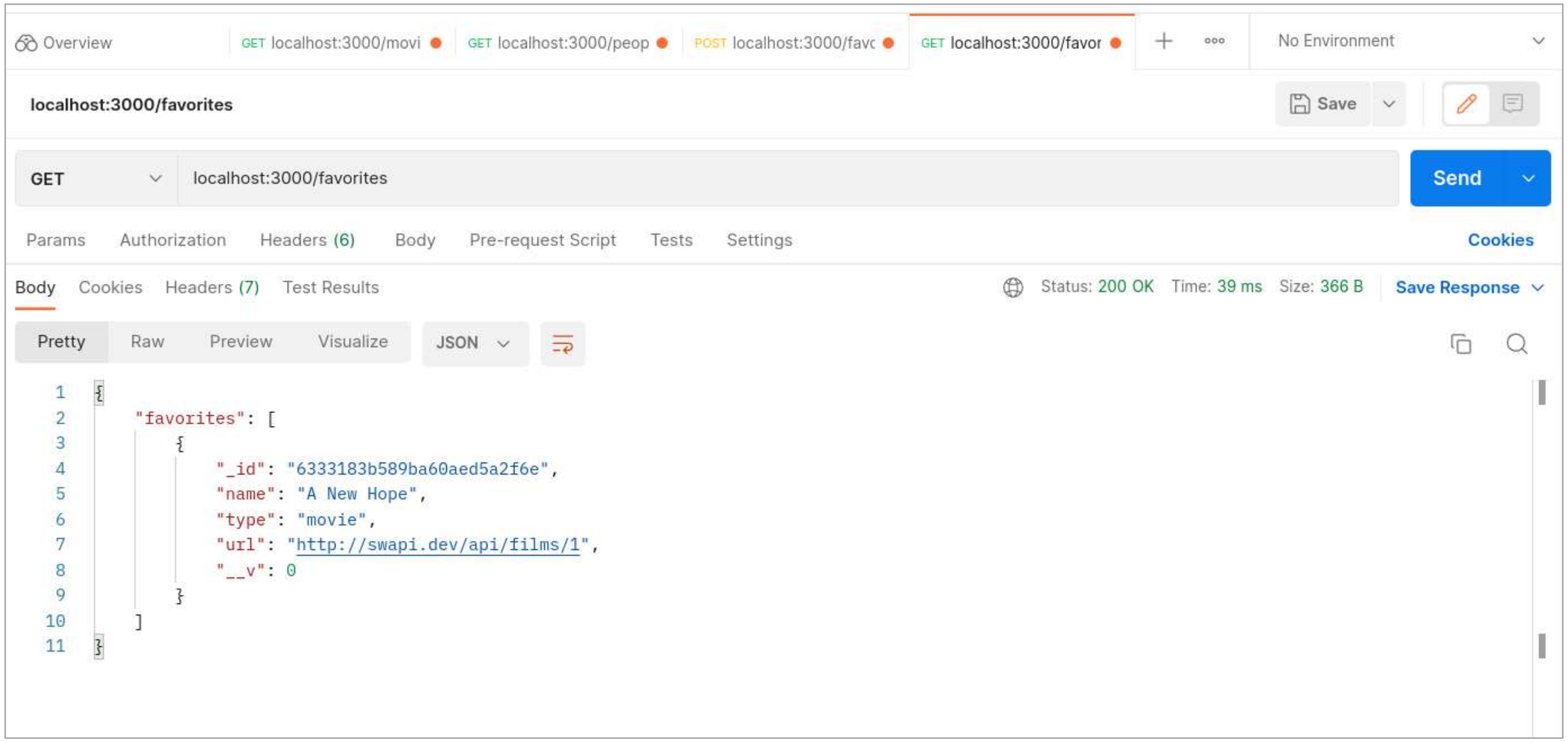
3. Run Web App Container



```
[user@user-virtualbox networks-starting-setup]$  docker run --name favorites-app --rm -p 3000:3000 favorites-app-image
```

Favorite Movies Web App Example
Container to Host Communication – Test Connectivity

4. Test the GET favorite movies endpoint



localhost:3000/favorites

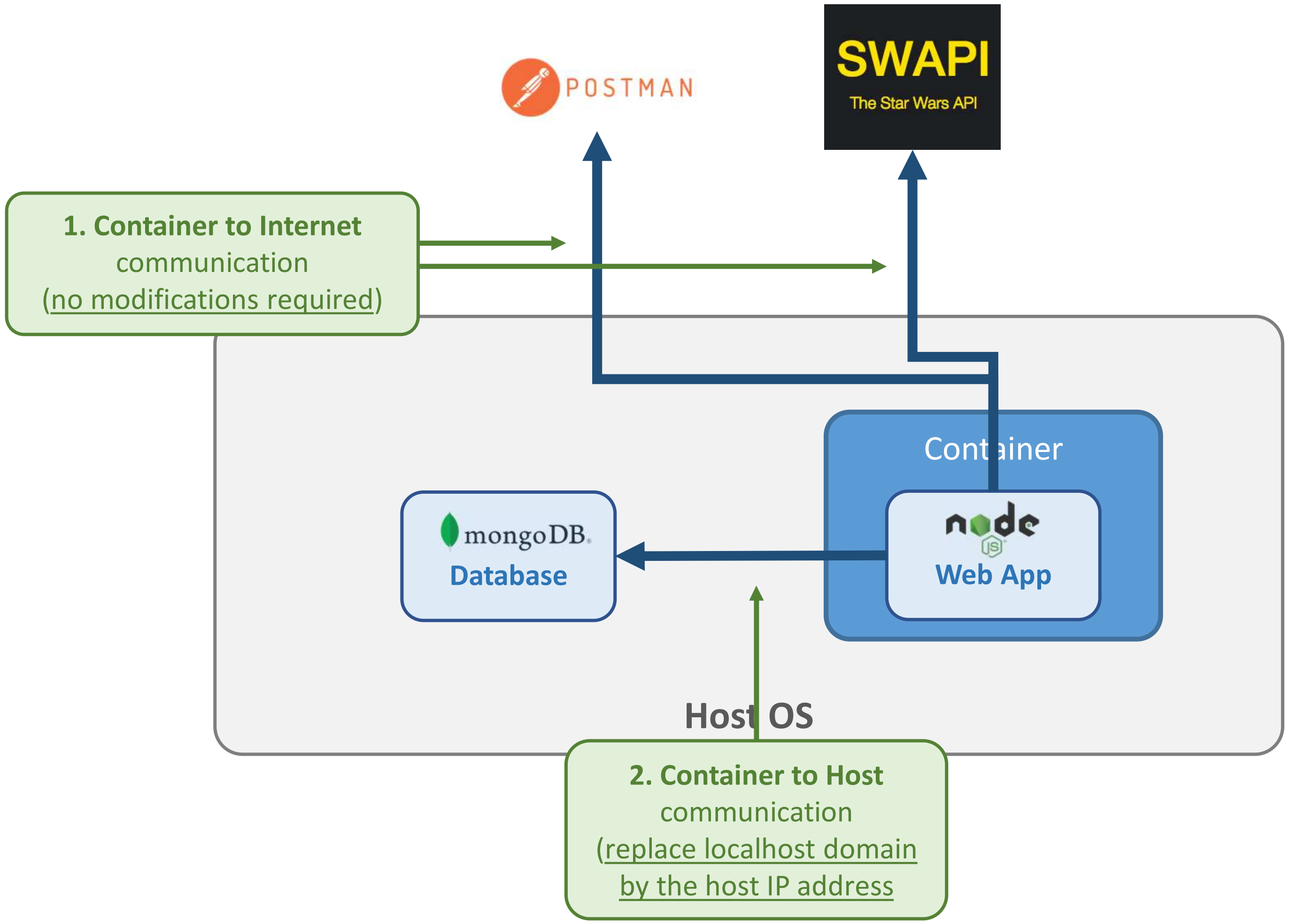
GET localhost:3000/favorites

Status: 200 OK Time: 39 ms Size: 366 B

```
{
    "favorites": [
        {
            "_id": "6333183b589ba60aed5a2f6e",
            "name": "A New Hope",
            "type": "movie",
            "url": "http://swapi.dev/api/films/1",
            "__v": 0
        }
    ]
}
```

Favorite Movies Web App Example

Running the App With Containers and non-Containers



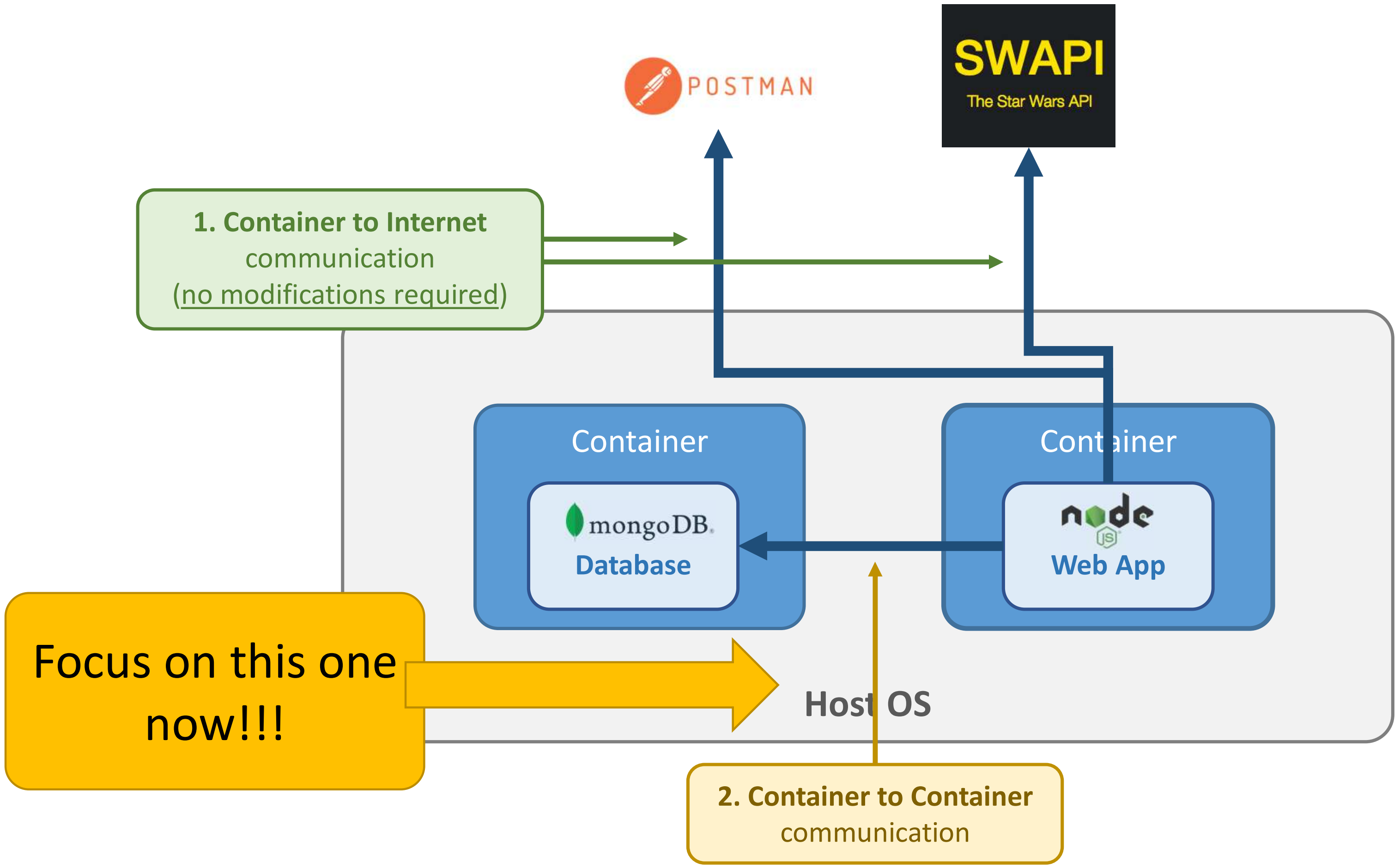
SWAPI
The Star Wars API
POSTMAN
1. Container to Internet communication (no modifications required)
Container
mongoDB
Database
node JS
Web App
Host OS
2. Container to Host communication (replace localhost domain by the host IP address

Step 4

Running the App **fully** containerized

Favorite Movies Web App Example

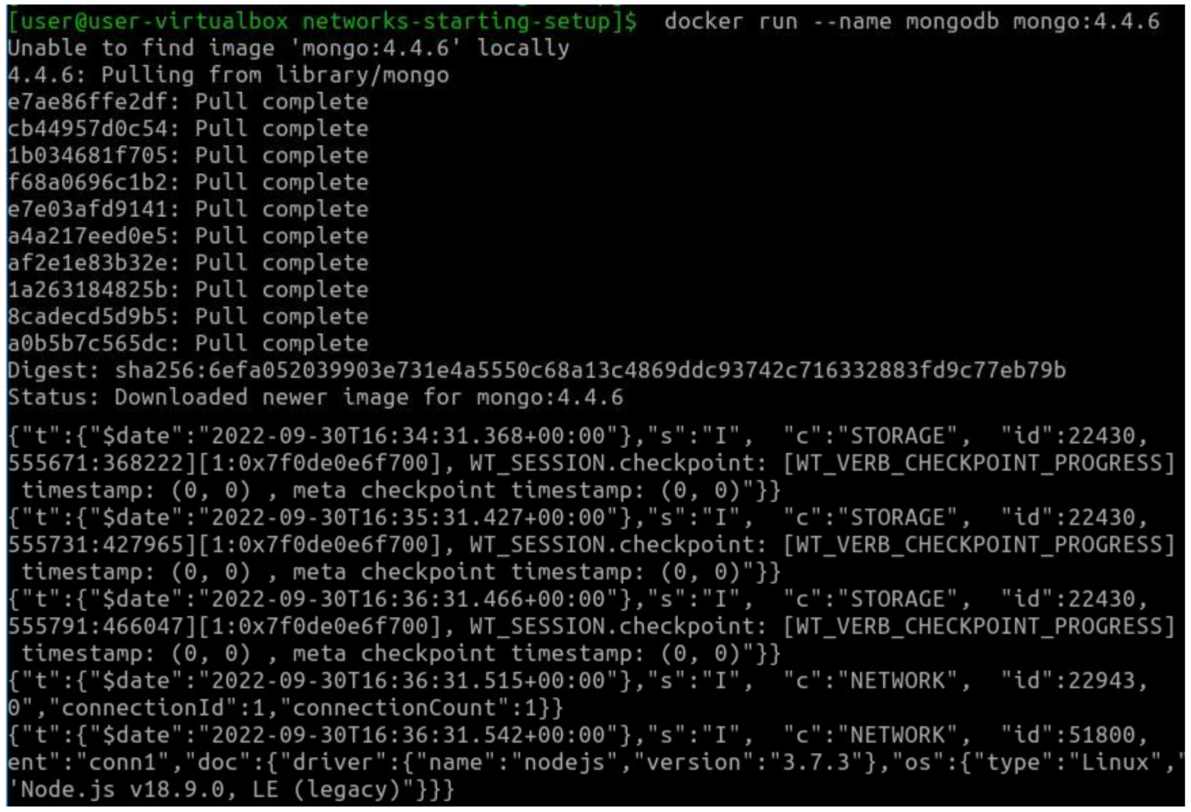
Running the App Fully Containerized



Favorite Movies Web App Example

Container to Container Communication – Run MongoDB Container

1. Run MongoDB Container (pre-built image)



```
[user@user-virtualbox networks-starting-setup]$  docker run --name mongodb mongo:4.4.6
Unable to find image 'mongo:4.4.6' locally
4.4.6: Pulling from library/mongo
e7ae86ffe2df: Pull complete
cb44957d0c54: Pull complete
1b034681f705: Pull complete
f68a0696c1b2: Pull complete
e7e03afd9141: Pull complete
a4a217eed0e5: Pull complete
af2e1e83b32e: Pull complete
1a263184825b: Pull complete
8cadecd5d9b5: Pull complete
a0b5b7c565dc: Pull complete
Digest: sha256:6efa052039903e731e4a5550c68a13c4869ddc93742c716332883fd9c77eb79b
Status: Downloaded newer image for mongo:4.4.6
{"t":{"$date":"2022-09-30T16:34:31.368+00:00"},"s":"I",  "c":"STORAGE",  "id":22430,
555671:368222][1:0x7f0de0e6f700], WT_SESSION.checkpoint: [WT_VERB_CHECKPOINT_PROGRESS]
 timestamp: (0, 0) , meta checkpoint timestamp: (0, 0)"}}
{"t":{"$date":"2022-09-30T16:35:31.427+00:00"},"s":"I",  "c":"STORAGE",  "id":22430,
555731:427965][1:0x7f0de0e6f700], WT_SESSION.checkpoint: [WT_VERB_CHECKPOINT_PROGRESS]
 timestamp: (0, 0) , meta checkpoint timestamp: (0, 0)"}}
{"t":{"$date":"2022-09-30T16:36:31.466+00:00"},"s":"I",  "c":"STORAGE",  "id":22430,
555791:466047][1:0x7f0de0e6f700], WT_SESSION.checkpoint: [WT_VERB_CHECKPOINT_PROGRESS]
 timestamp: (0, 0) , meta checkpoint timestamp: (0, 0)"}}
{"t":{"$date":"2022-09-30T16:36:31.515+00:00"},"s":"I",  "c":"NETWORK",  "id":22943,
0","connectionId":1,"connectionCount":1}}
{"t":{"$date":"2022-09-30T16:36:31.542+00:00"},"s":"I",  "c":"NETWORK",  "id":51800,
ent":"conn1","doc":{"driver":{"name":"nodejs","version":"3.7.3"},"os":{"type":"Linux","
'Node.js v18.9.0, LE (legacy)"}}}
```

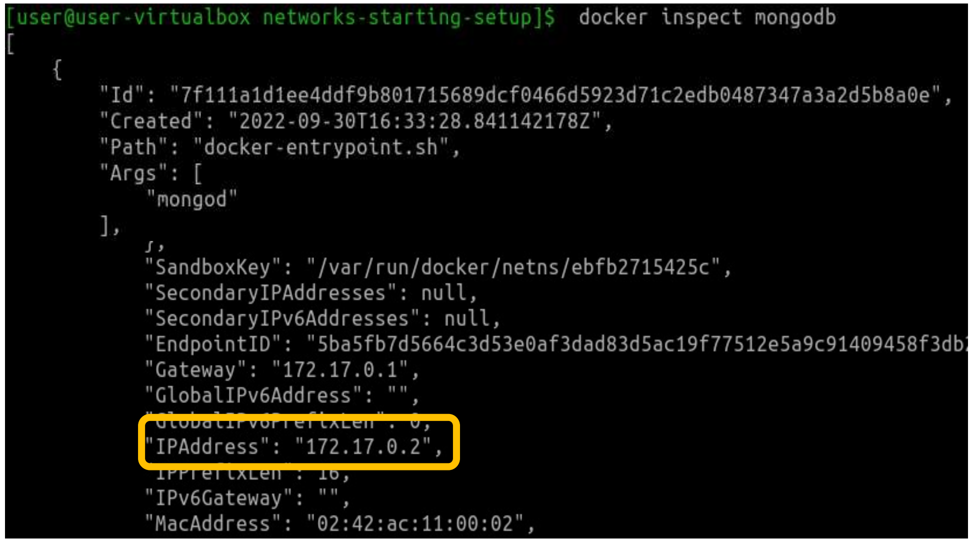
Container to Container Communication

Replace localhost domain

- Replace “localhost” domain in source code by “mongoDB container IP address”

```
‘…localhost:…’→ ‘…container-IP-address:…’
```

- Use `docker inspect` command to discover container IP address
  - **Not easy process!!!**

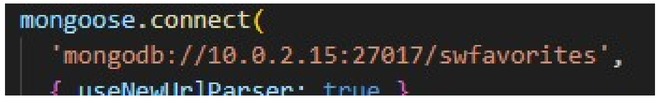


```
[user@user-virtualbox networks-starting-setup]$  docker inspect mongodb
[
    {
        "Id": "7f111a1d1ee4ddf9b801715689dcf0466d5923d71c2edb0487347a3a2d5b8a0e",
        "Created": "2022-09-30T16:33:28.841142178Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "mongod"
        ],
            },
            "SandboxKey": "/var/run/docker/netns/ebfb2715425c",
            "SecondaryIPAddresses": null,
            "SecondaryIPv6Addresses": null,
            "EndpointID": "5ba5fb7d5664c3d53e0af3dad83d5ac19f77512e5a9c91409458f3db
            "Gateway": "172.17.0.1",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "IPAddress": "172.17.0.2",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
            "MacAddress": "02:42:ac:11:00:02",
```

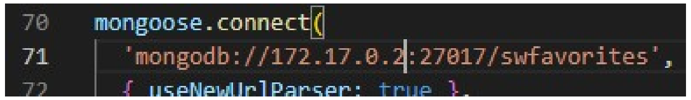
Favorite Movies Web App Example

Container to Container Communication – Replace IP address and Run Container

2. Replace localhost domain

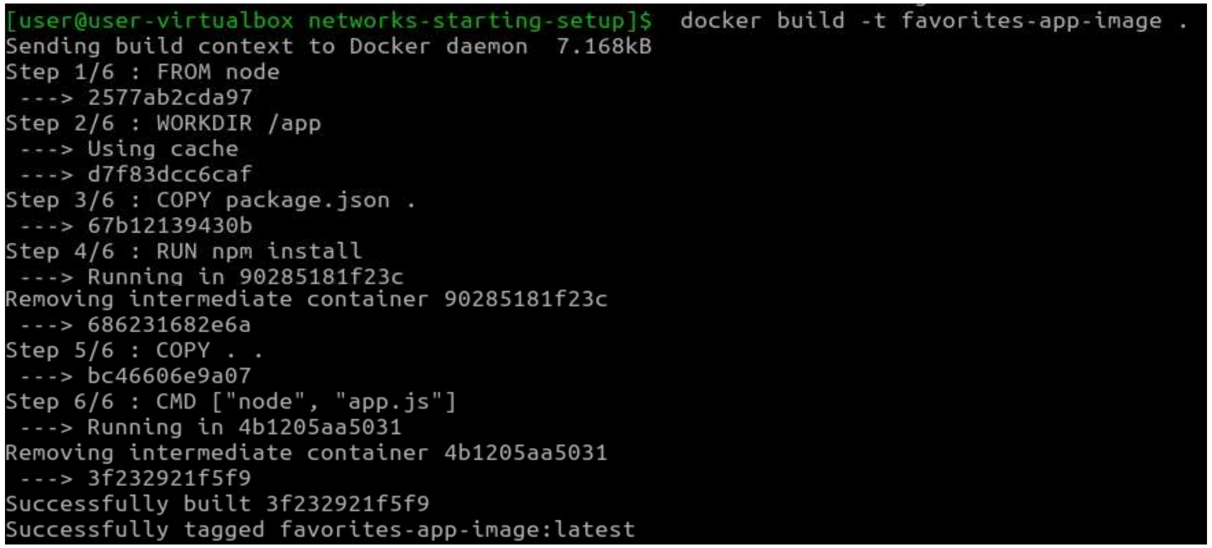


```
mongoose.connect(
  'mongodb://10.0.2.15:27017/swfavorites',
  { useNewUrlParser: true }
```



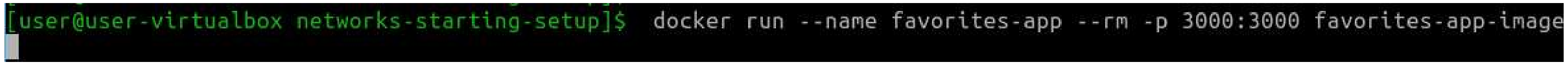
```
mongoose.connect(
  'mongodb://172.17.0.2:27017/swfavorites',
  { useNewUrlParser: true },
```

3. Build Image



```
[user@user-virtualbox networks-starting-setup]$  docker build -t favorites-app-image .
Sending build context to Docker daemon  7.168kB
Step 1/6 : FROM node
 ---> 2577ab2cda97
Step 2/6 : WORKDIR /app
 ---> Using cache
 ---> d7f83dcc6caf
Step 3/6 : COPY package.json .
 ---> 67b12139430b
Step 4/6 : RUN npm install
 ---> Running in 90285181f23c
Removing intermediate container 90285181f23c
 ---> 686231682e6a
Step 5/6 : COPY . .
 ---> bc46606e9a07
Step 6/6 : CMD ["node", "app.js"]
 ---> Running in 4b1205aa5031
Removing intermediate container 4b1205aa5031
 ---> 3f232921f5f9
Successfully built 3f232921f5f9
Successfully tagged favorites-app-image:latest
```

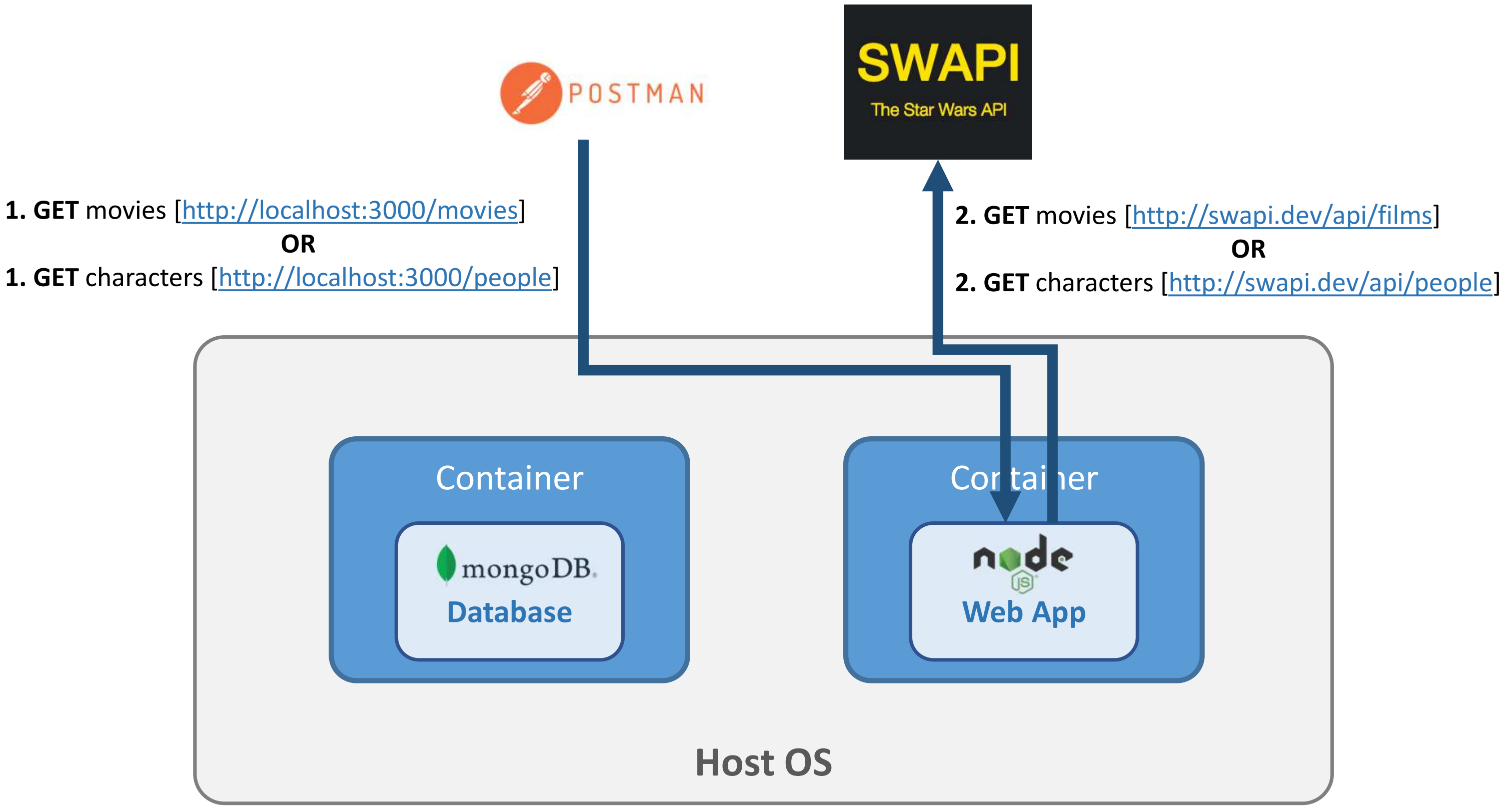
4. Run Web App Container



```
[user@user-virtualbox networks-starting-setup]$  docker run --name favorites-app --rm -p 3000:3000 favorites-app-image
```

Favorite Movies Web App Example

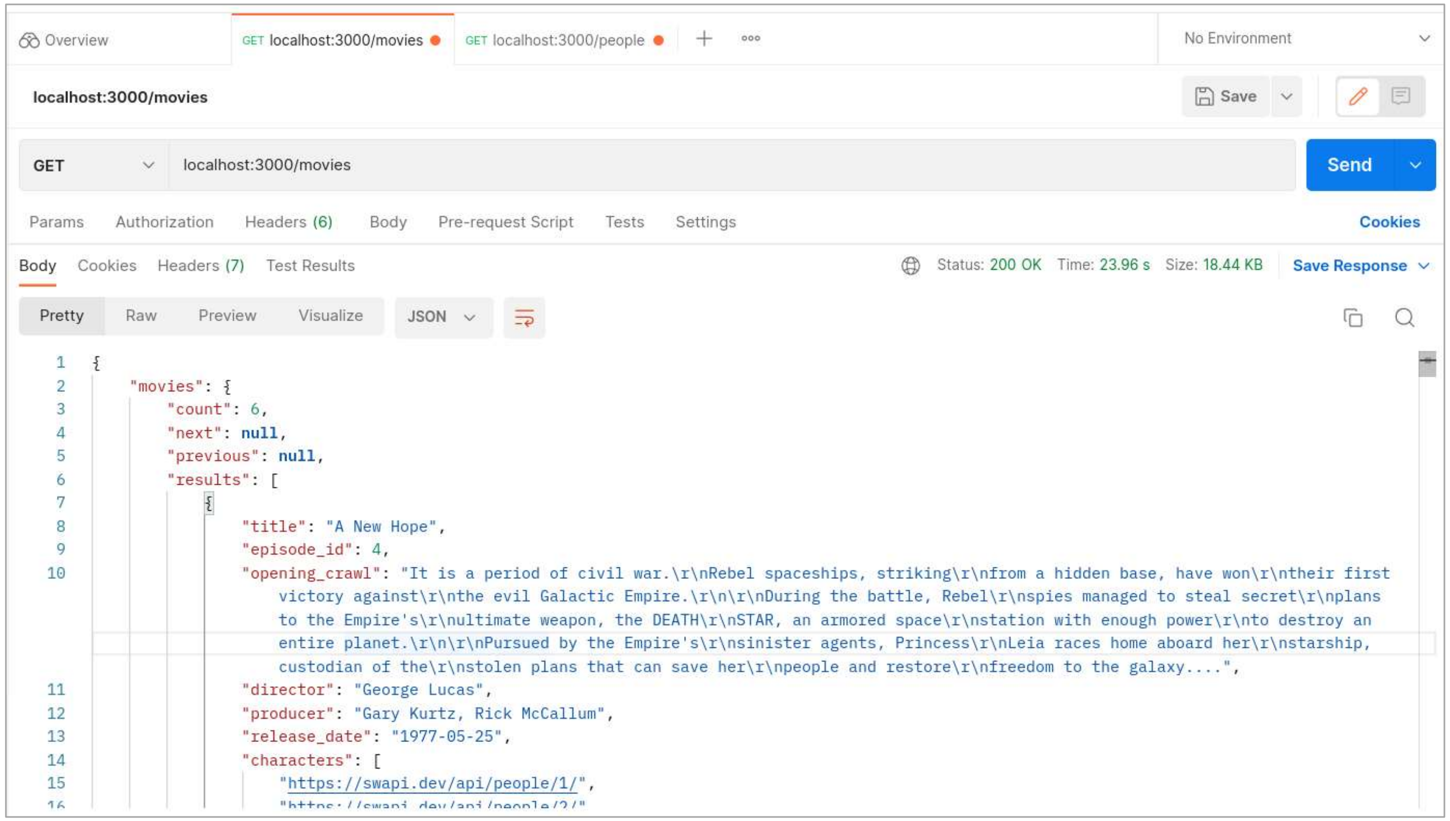
Running the App Fully Containerized – HTTP GET Movies/Characters


POSTMAN
SWAPI
The Star Wars API
1. GET movies [http://localhost:3000/movies]
OR
1. GET characters [http://localhost:3000/people]
2. GET movies [http://swapi.dev/api/films]
OR
2. GET characters [http://swapi.dev/api/people]
Container
mongoDB
Database
Container
node
Web App
Host OS

Favorite Movies Web App Example

Running the App Fully Containerized – HTTP GET Movies

5. Test the GET movies endpoint

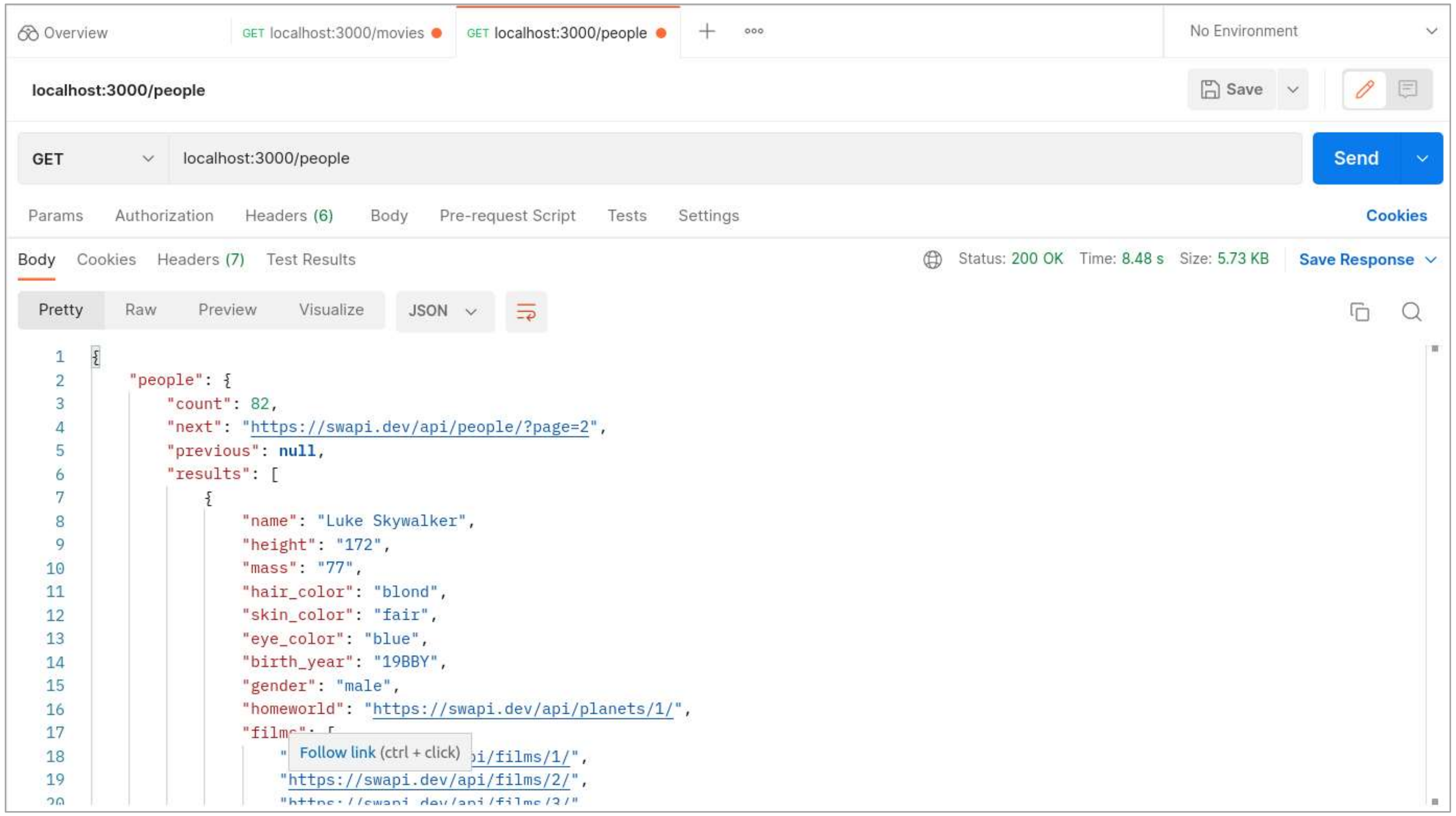


```
{
    "movies": {
        "count": 6,
        "next": null,
        "previous": null,
        "results": [
            {
                "title": "A New Hope",
                "episode_id": 4,
                "opening_crawl": "It is a period of civil war.\r\nRebel spaceships, striking\r\nfrom a hidden base, have won\r\ntheir first
                    victory against\r\nthe evil Galactic Empire.\r\n\r\nDuring the battle, Rebel\r\nspies managed to steal secret\r\nplans
                    to the Empire's\r\nultimate weapon, the DEATH\r\nSTAR, an armored space\r\nstation with enough power\r\nto destroy an
                    entire planet.\r\n\r\nPursued by the Empire's\r\nsinister agents, Princess\r\nLeia races home aboard her\r\nstarship,
                    custodian of the\r\nstolen plans that can save her\r\npeople and restore\r\nfreedom to the galaxy....",
                "director": "George Lucas",
                "producer": "Gary Kurtz, Rick McCallum",
                "release_date": "1977-05-25",
                "characters": [
                    "https://swapi.dev/api/people/1/",
                    "https://swapi.dev/api/people/2/"
```

Favorite Movies Web App Example

Running the App Fully Containerized – HTTP GET People

6. Test the GET characters endpoint



localhost:3000/people

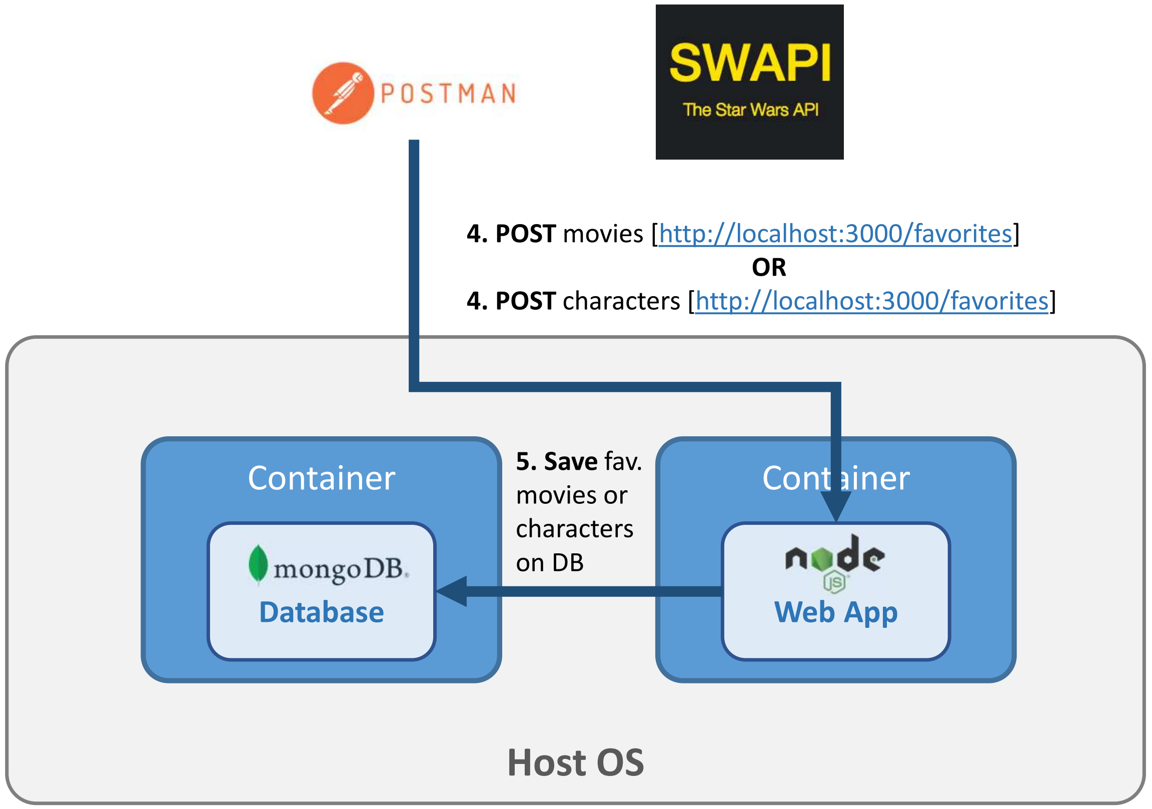
GET localhost:3000/people

Status: 200 OK Time: 8.48 s Size: 5.73 KB

```
{
    "people": {
        "count": 82,
        "next": "https://swapi.dev/api/people/?page=2",
        "previous": null,
        "results": [
            {
                "name": "Luke Skywalker",
                "height": "172",
                "mass": "77",
                "hair_color": "blond",
                "skin_color": "fair",
                "eye_color": "blue",
                "birth_year": "19BBY",
                "gender": "male",
                "homeworld": "https://swapi.dev/api/planets/1/",
                "films": [
                    "                pi/films/1/",
                    "https://swapi.dev/api/films/2/",
```

Favorite Movies Web App Example

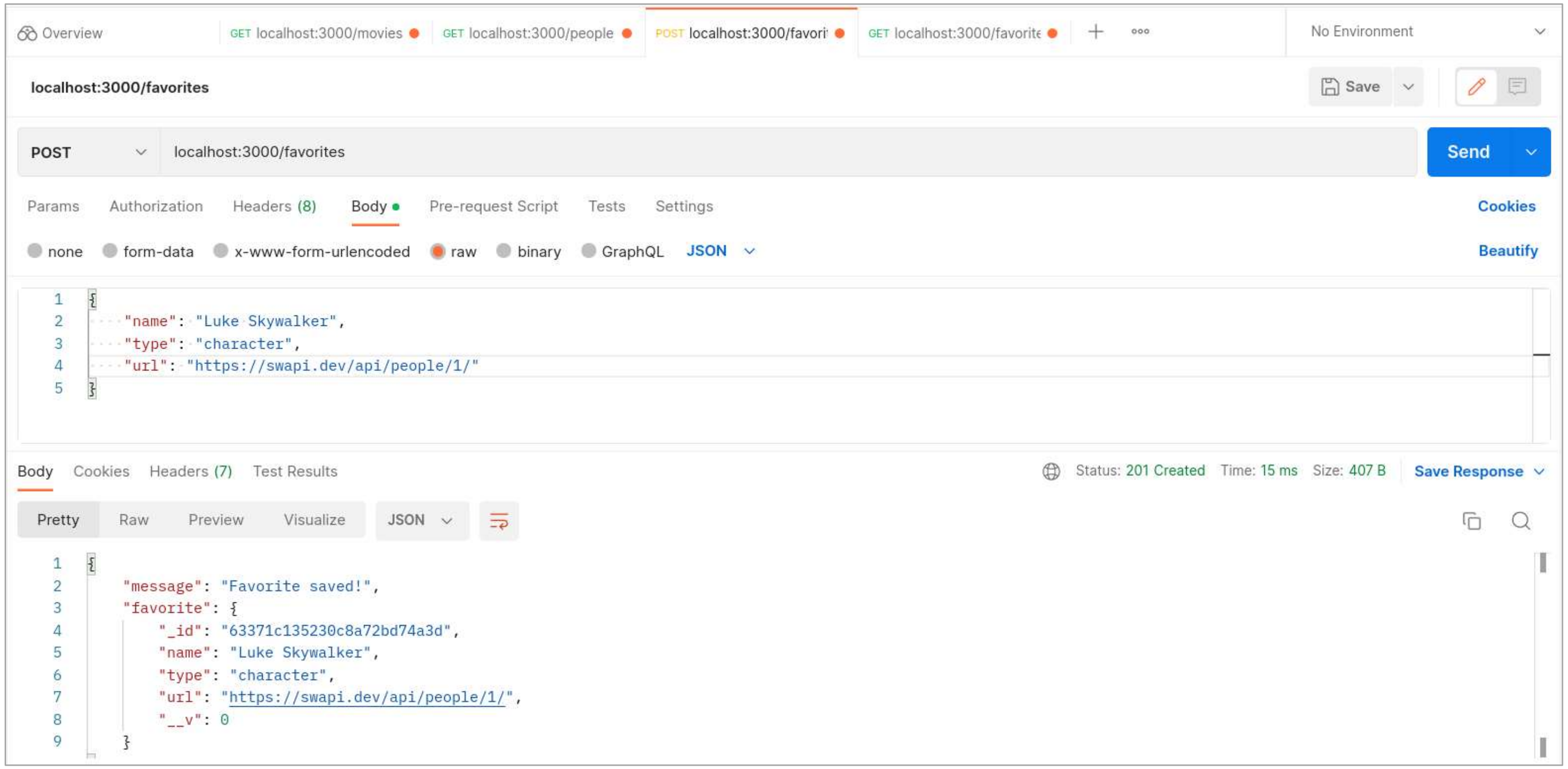
Running the App Fully Containerized – HTTP POST Favorite Movies


POSTMAN
SWAPI
The Star Wars API
4. POST movies [http://localhost:3000/favorites]
OR
4. POST characters [http://localhost:3000/favorites]
Container
mongoDB.
Database
5. Save fav. movies or characters on DB
Container
node JS
Web App
Host OS

Favorite Movies Web App Example

Running the App Fully Containerized – HTTP POST Favorite Movies

7. Test the POST favorites movies/characters endpoint



localhost:3000/favorites

POST localhost:3000/favorites

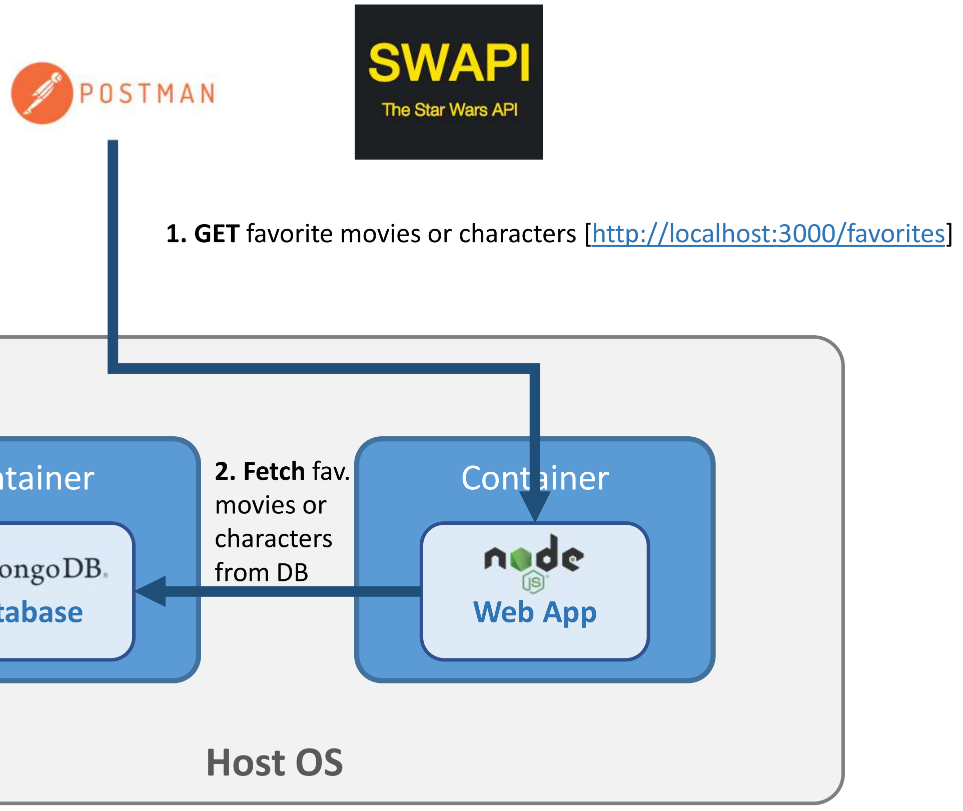
```
{
    "name": "Luke Skywalker",
    "type": "character",
    "url": "https://swapi.dev/api/people/1/"
}
```

Status: 201 Created Time: 15 ms Size: 407 B

```
{
    "message": "Favorite saved!",
    "favorite": {
        "_id": "63371c135230c8a72bd74a3d",
        "name": "Luke Skywalker",
        "type": "character",
        "url": "https://swapi.dev/api/people/1/",
        "__v": 0
    }
```

Favorite Movies Web App Example

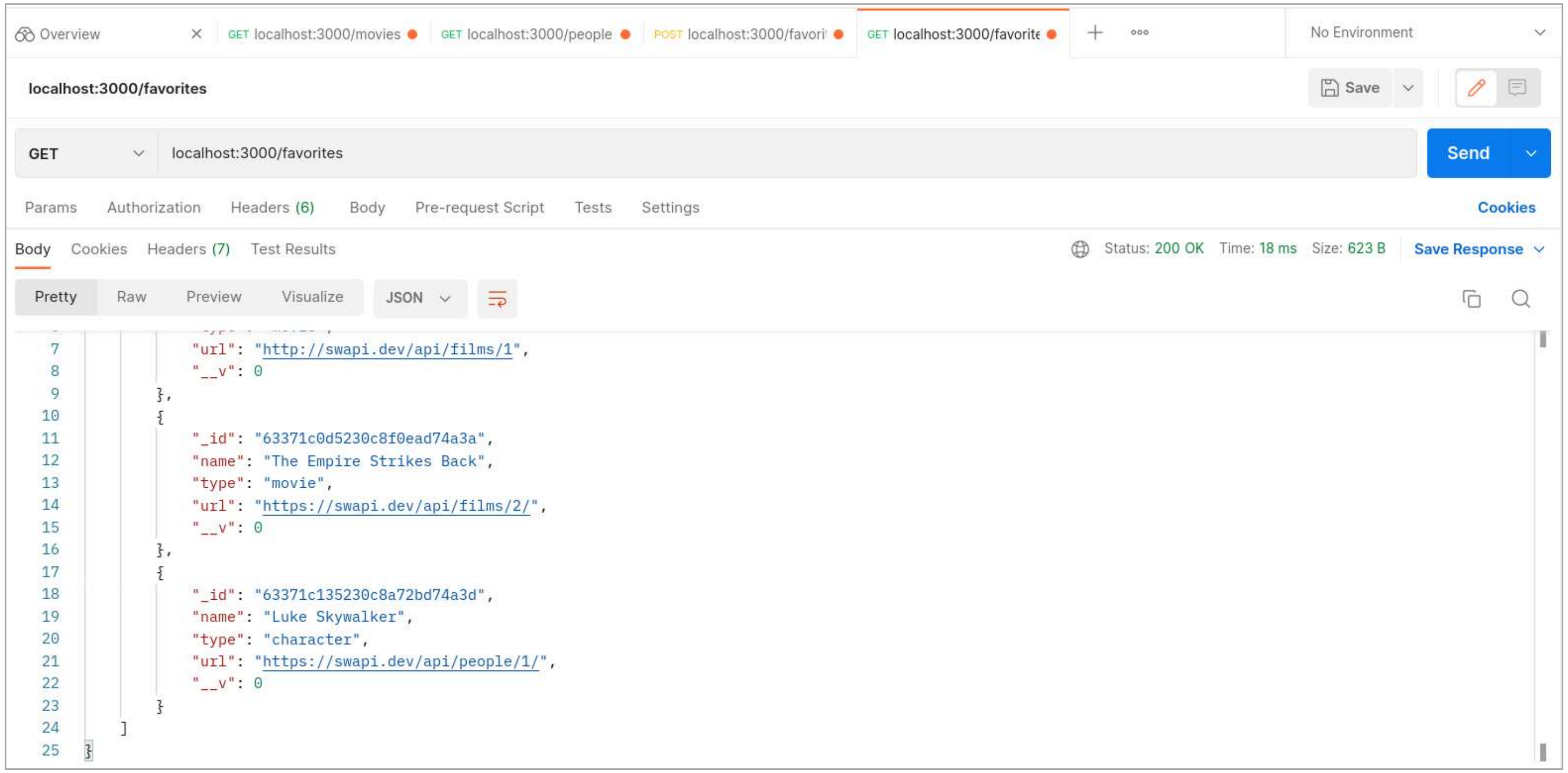
Running the App Fully Containerized – HTTP GET Favorite Movies and Characters from DB


POSTMAN
SWAPI
The Star Wars API
1. GET favorite movies or characters [http://localhost:3000/favorites]
Container
mongoDB.
Database
2. Fetch fav. movies or characters from DB
Container
node JS
Web App
Host OS

Favorite Movies Web App Example

Running the App Fully Containerized – HTTP GET Favorite Movies and Characters from DB

8. Test the GET favorites endpoint



localhost:3000/favorites

GET localhost:3000/favorites

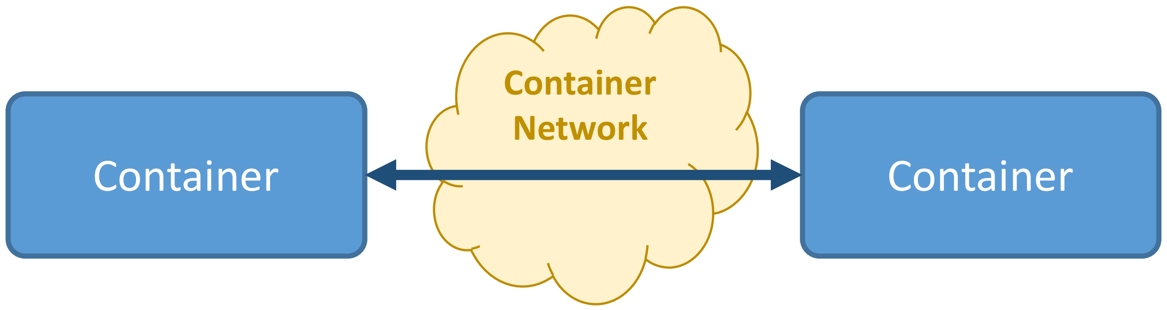
Status: 200 OK Time: 18 ms Size: 623 B

```
            "url": "http://swapi.dev/api/films/1",
            "__v": 0
        },
        {
            "_id": "63371c0d5230c8f0ead74a3a",
            "name": "The Empire Strikes Back",
            "type": "movie",
            "url": "https://swapi.dev/api/films/2/",
            "__v": 0
        },
        {
            "_id": "63371c135230c8a72bd74a3d",
            "name": "Luke Skywalker",
            "type": "character",
            "url": "https://swapi.dev/api/people/1/",
            "__v": 0
        }
    ]
}
```

Step 5

Running the App **fully** containerized in a **more simple** way

Container to Container Communication
Create Container Network



- Create a **container network**
  - Creates a network in which all containers are able to communication with each other

- How?

Manage Networks

Objective & Syntax

- Objective
  - Create a docker containers network

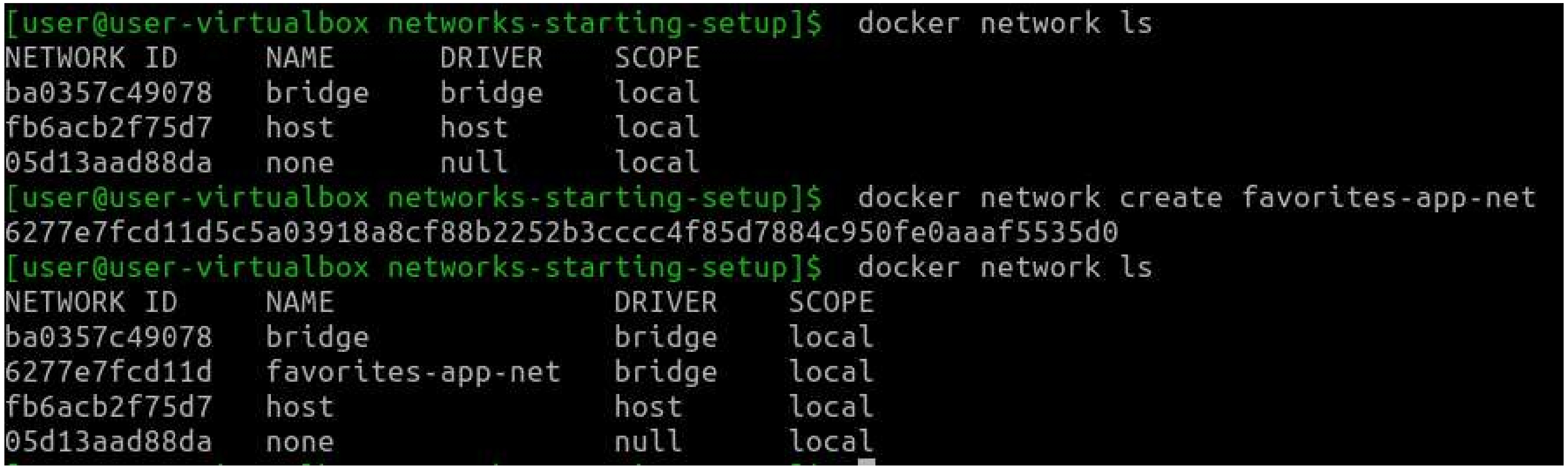
```
docker    network    create    <network-name>
```

Create a docker network

Favorite Movies Web App Example

Container to Container Communication – Create a Container Network

1. Create a Docker Container Network

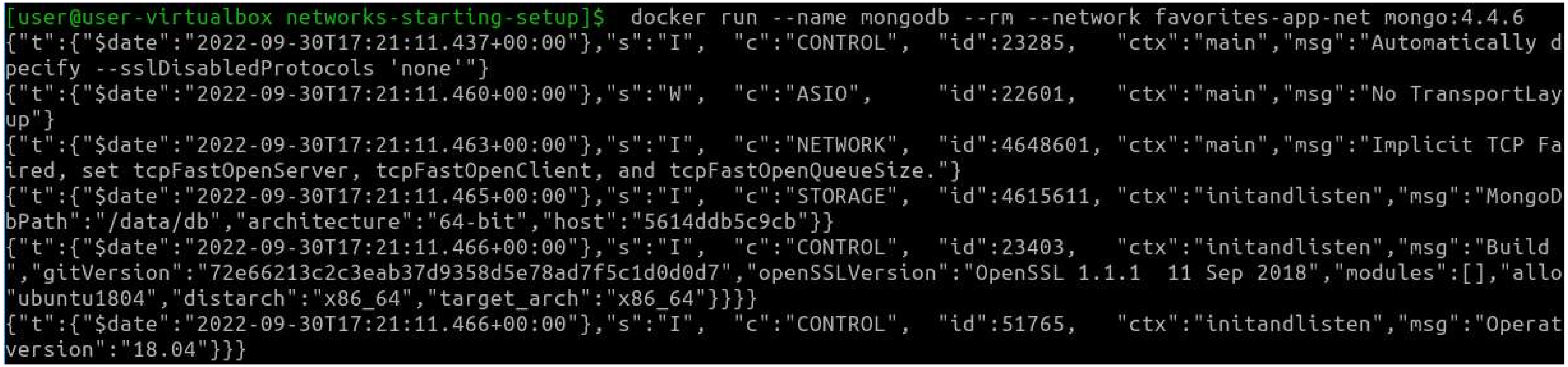


```
[user@user-virtualbox networks-starting-setup]$  docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
ba0357c49078   bridge    bridge    local
fb6acb2f75d7   host      host      local
05d13aad88da   none      null      local
[user@user-virtualbox networks-starting-setup]$  docker network create favorites-app-net
6277e7fcd11d5c5a03918a8cf88b2252b3cccc4f85d7884c950fe0aaaf5535d0
[user@user-virtualbox networks-starting-setup]$  docker network ls
NETWORK ID     NAME                DRIVER    SCOPE
ba0357c49078   bridge              bridge    local
6277e7fcd11d   favorites-app-net   bridge    local
fb6acb2f75d7   host                host      local
05d13aad88da   none                null      local
```

Favorite Movies Web App Example

Container to Container Communication – Run MongoDB Container

2. Restart MongoDB Container (connected to the docker network)

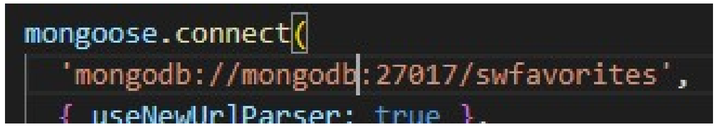


```
[user@user-virtualbox networks-starting-setup]$  docker run --name mongodb --rm --network favorites-app-net mongo:4.4.6
{"t":{"$date":"2022-09-30T17:21:11.437+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg":"Automatically d
pecify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2022-09-30T17:21:11.460+00:00"},"s":"W",  "c":"ASIO",     "id":22601,   "ctx":"main","msg":"No TransportLay
up"}
{"t":{"$date":"2022-09-30T17:21:11.463+00:00"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"main","msg":"Implicit TCP Fa
ired, set tcpFastOpenServer, tcpFastOpenClient, and tcpFastOpenQueueSize."}
{"t":{"$date":"2022-09-30T17:21:11.465+00:00"},"s":"I",  "c":"STORAGE",  "id":4615611, "ctx":"initandlisten","msg":"MongoD
bPath":"/data/db","architecture":"64-bit","host":"5614ddb5c9cb"}}
{"t":{"$date":"2022-09-30T17:21:11.466+00:00"},"s":"I",  "c":"CONTROL",  "id":23403,   "ctx":"initandlisten","msg":"Build
","gitVersion":"72e66213c2c3eab37d9358d5e78ad7f5c1d0d0d7","openSSLVersion":"OpenSSL 1.1.1  11 Sep 2018","modules":[],"allo
"ubuntu1804","distarch":"x86_64","target_arch":"x86_64"}}}}
{"t":{"$date":"2022-09-30T17:21:11.466+00:00"},"s":"I",  "c":"CONTROL",  "id":51765,   "ctx":"initandlisten","msg":"Operat
version":"18.04"}}}
```

Favorite Movies Web App Example

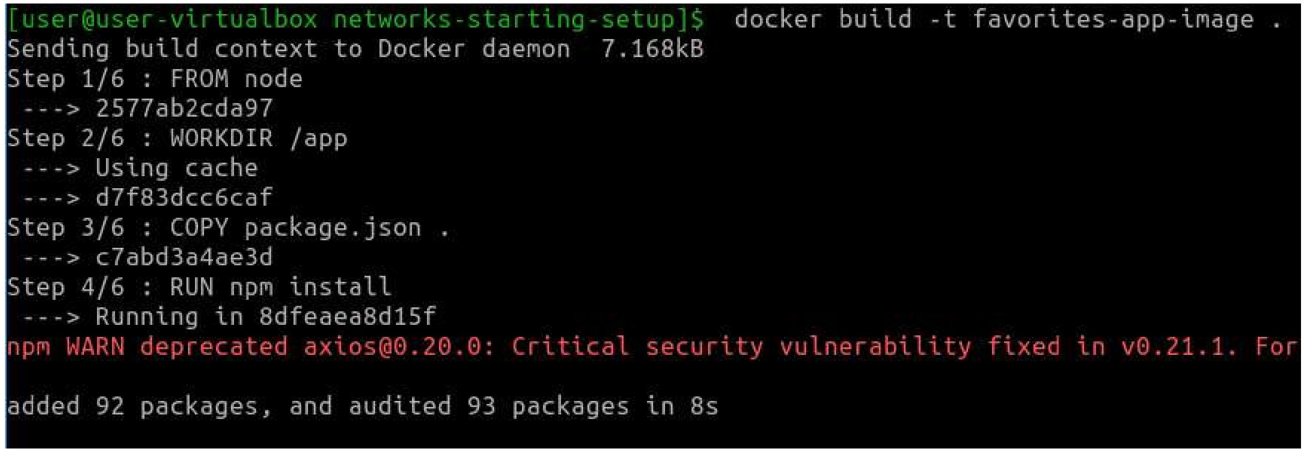
Container to Container Communication – Build Web App Container Image

3. Change localhost to destination container name - mongodb



```
mongoose.connect(
  'mongodb://mongodb:27017/swfavorites',
  { useNewUrlParser: true },
```

4. Build the new web app container

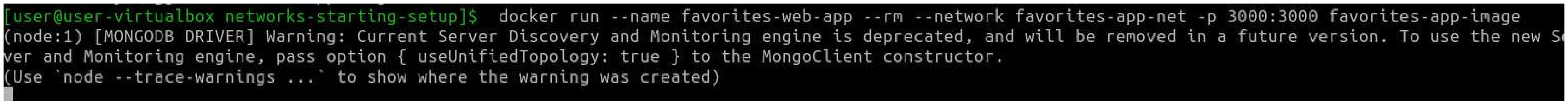


```
[user@user-virtualbox networks-starting-setup]$  docker build -t favorites-app-image .
Sending build context to Docker daemon  7.168kB
Step 1/6 : FROM node
 ---> 2577ab2cda97
Step 2/6 : WORKDIR /app
 ---> Using cache
 ---> d7f83dcc6caf
Step 3/6 : COPY package.json .
 ---> c7abd3a4ae3d
Step 4/6 : RUN npm install
 ---> Running in 8dfeaea8d15f
npm WARN deprecated axios@0.20.0: Critical security vulnerability fixed in v0.21.1. For

added 92 packages, and audited 93 packages in 8s
```

Favorite Movies Web App Example

Container to Container Communication – Run Web App Container

5. Restart web app container connected to the docker container network

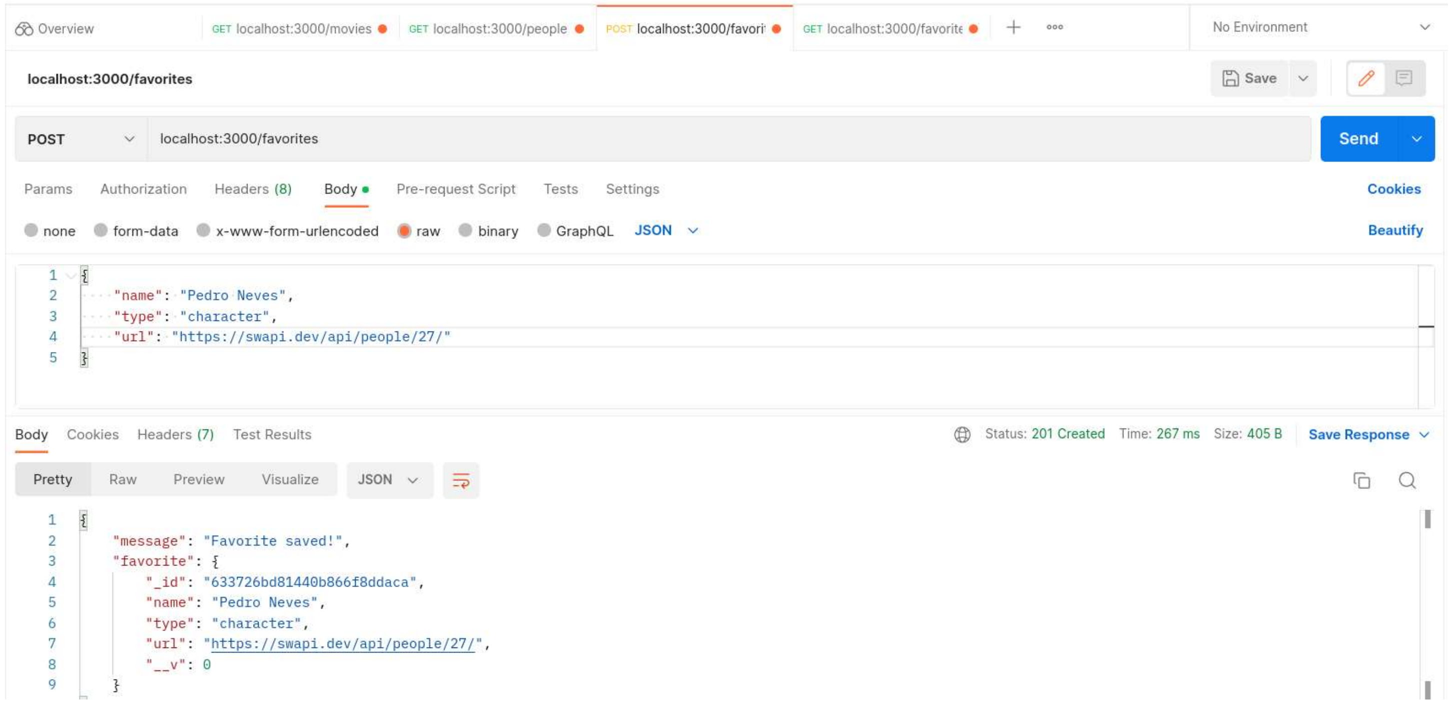


```
[user@user-virtualbox networks-starting-setup]$  docker run --name favorites-web-app --rm --network favorites-app-net -p 3000:3000 favorites-app-image
(node:1) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Se
ver and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
(Use `node --trace-warnings ...` to show where the warning was created)
```

Favorite Movies Web App Example

Container to Container Communication – HTTP POST Favorite Movies

6. Test the POST favorites movies/characters endpoint


Overview
GET localhost:3000/movies
GET localhost:3000/people
POST localhost:3000/favori
GET localhost:3000/favorite
No Environment
Save
Send
Cookies
Beautify

localhost:3000/favorites

POST localhost:3000/favorites

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
    "name": "Pedro Neves",
    "type": "character",
    "url": "https://swapi.dev/api/people/27/"
}
```

Body Cookies Headers (7) Test Results

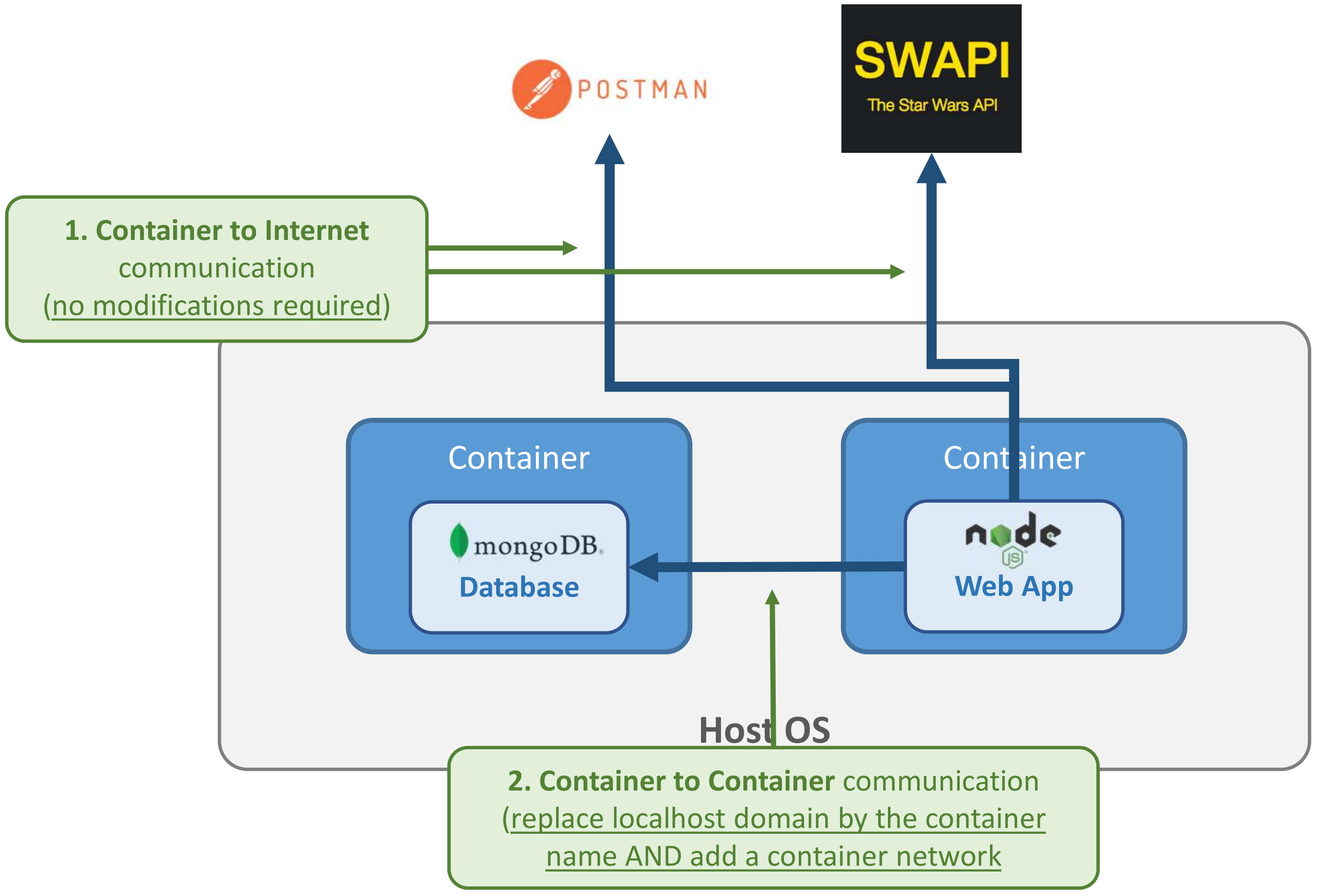
Status: 201 Created Time: 267 ms Size: 405 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "message": "Favorite saved!",
    "favorite": {
        "_id": "633726bd81440b866f8ddaca",
        "name": "Pedro Neves",
        "type": "character",
        "url": "https://swapi.dev/api/people/27/",
        "__v": 0
    }
```

Favorite Movies Web App Example

Running the App Fully Containerized


POSTMAN
SWAPI
The Star Wars API
1. Container to Internet communication (no modifications required)
Container
mongoDB
Database
Container
node
Web App
Host OS
2. Container to Container communication (replace localhost domain by the container name AND add a container network

Outline

Summary

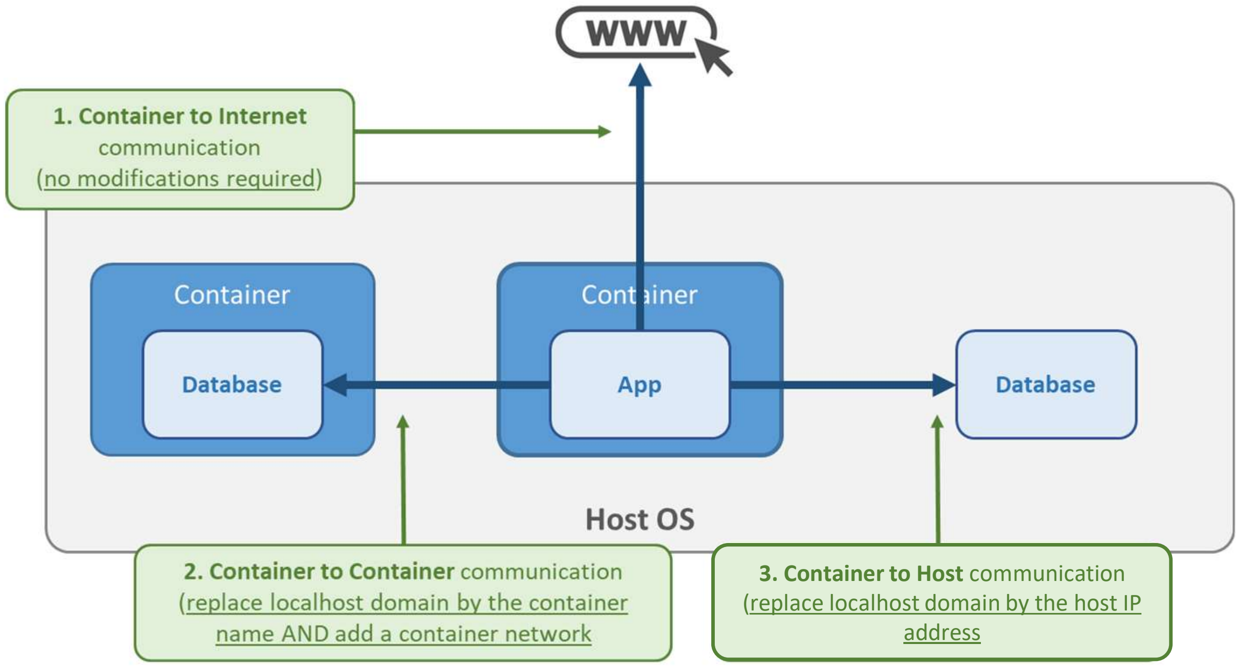
What have you learned

1. Understand how Docker **containers manage** the **communication** with external entities

Summary

What have you learned

1. Understand how Docker **containers manage** the **communication** with external entities


WWW
1. Container to Internet communication (no modifications required)
Container
Database
Container
App
Database
Host OS
2. Container to Container communication (replace localhost domain by the container name AND add a container network
3. Container to Host communication (replace localhost domain by the host IP address

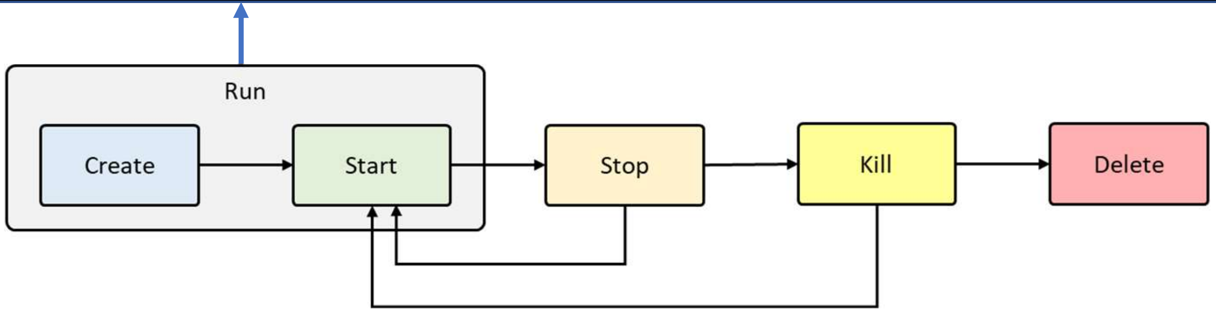
Summary

What have you learned

Update from Module 5

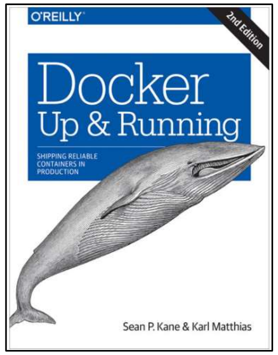
3. Understand the **Docker technology** by manipulating containers during their **lifecycle**

```
docker ps
docker ps -a
docker run -p <port-X:port-Y> -d <image-id>
docker run -p <port-X:port-Y> -d --name <container-name> <image-id>
docker run -p <port-X:port-Y> -d --rm --name <container-name> <image-id>
docker run -p <port-X:port-Y> -d --rm --name <container-name> -v <volume-name:container-path> <image-id>
docker run -p <port-X:port-Y> -d --rm --name <container-name> --network <docker-network> <image-id>
docker run -it <image-name>
docker attach <container-identifier>
docker logs <container-identifier>
```

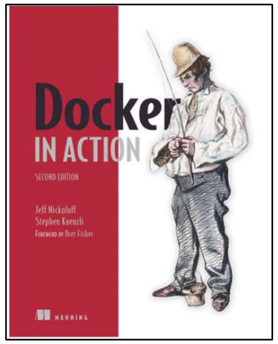


Bibliography

- **Docker: Up & Running**, 2nd Edition, Sean P. Kane, Karl Matthias, Published by O'Reilly Media, Inc.


O'REILLY
2nd Edition
Docker
Up & Running
SHIPPING RELIABLE CONTAINERS IN PRODUCTION
Sean P. Kane & Karl Matthias

- **Docker in Action**, Second Edition, Jeffrey Nickoloff, Stephen Kuenzli, Published by Manning Publications



Next ...

Module 7 – Docker Compose