

DevOps: Dato Dźneladze

1. install ubuntu server
2. install docker, docker-compose, npm
3. clone repository: <https://github.com/gwynbleidd0014/6thElement.git>
4. make Dockerfile and docker-compose

```
version: '3.8'

services:
  app:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "8000:8080"
    depends_on:
      - db
    environment:
      - ConnectionStrings__DefaultConnection=Server=db;Database=SixthElement;User=sa;Password=6thElement;TrustServerCertificate=True;
      - ASPNETCORE_ENVIRONMENT=Development
    volumes:
      - ./server/6thElement/Images:/app/Images
    networks:
      - app-network

  db:
    image: mcr.microsoft.com/mssql/server:2019-latest
    environment:
      - ACCEPT_EULA=Y
      - SA_PASSWORD=6thElement
      - MSSQL_PID=Express
    ports:
      - "1433:1433"
    volumes:
      - sqldata:/var/opt/mssql
```

```
    networks:
      - app-network

networks:
  app-network:
    driver: bridge

volumes:
  sqldata:
  images_volume:
```

Dockerfile

```
# ბაზისური იმიჯი .NET 8.0-სთვის
FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS base
WORKDIR /app
EXPOSE 8080

# ბილდის იმიჯი
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
WORKDIR /src
COPY ["server/6thElement/6thElement.API/6thElement.API.csproj",
"6thElement.API/"]
COPY ["server/6thElement/6thElement.Application/6thElement.Application.csproj",
"6thElement.Application/"]
COPY ["server/6thElement/6thElement.Domain/6thElement.Domain.csproj",
"6thElement.Domain/"]
COPY
["server/6thElement/6thElement.Infrastructure/6thElement.Infrastructure.csproj",
"6thElement.Infrastructure/"]
COPY ["server/6thElement/6thElement.Persistance/6thElement.Persistance.csproj",
"6thElement.Persistance/"]
COPY ["server/6thElement/Images", "/app/Images"]
RUN dotnet restore "6thElement.API/6thElement.API.csproj"
COPY server/6thElement .
WORKDIR "/src/6thElement.API"
RUN dotnet build "6thElement.API.csproj" -c Release -o /app/build

# პუბლიკაციის იმიჯი
FROM build AS publish
```

```
RUN dotnet publish "6thElement.API.csproj" -c Release -o /app/publish
/p:UseAppHost=false
```

საბოლოო იმიჯი

```
FROM base AS final
```

```
WORKDIR /app
```

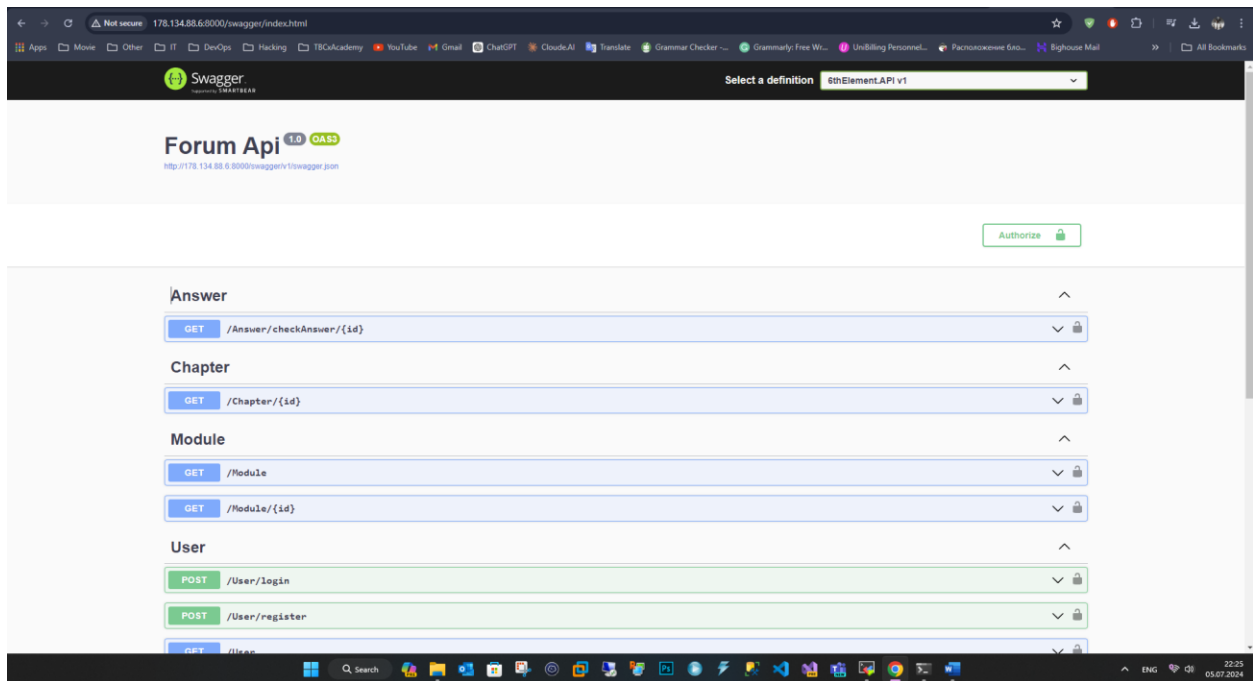
```
COPY --from=publish /app/publish .
```

```
ENTRYPOINT ["dotnet", "6thElement.API.dll"]
```

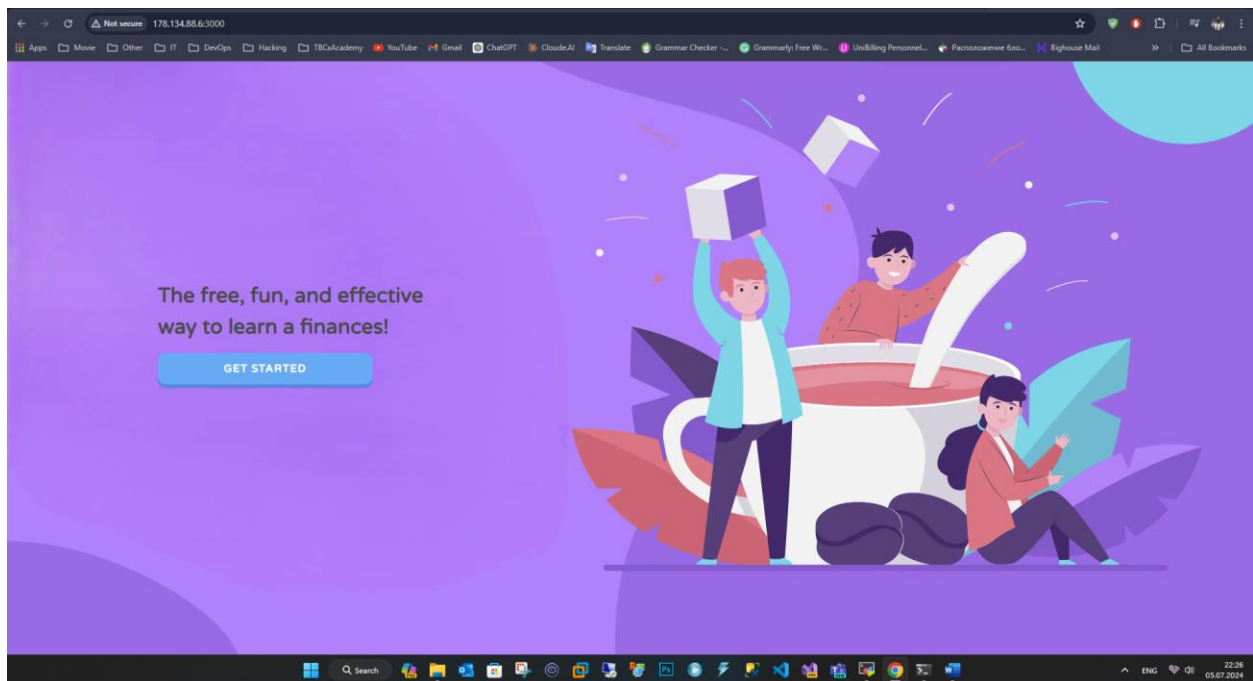
```
root@6th:~/6thElement# docker compose up --build
WARN[0000] /root/6thElement/docker-compose.yml: 'version' is obsolete
[+] Building 0.4s (23/23) FINISHED
=> [app internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.33kB
=> [app internal] load metadata for mcr.microsoft.com/dotnet/aspnet:8.0
=> [app internal] load metadata for mcr.microsoft.com/dotnet/sdk:8.0
=> [app internal] load dockerignore
=> => transferring context: 2B
=> [app build 1/12] FROM mcr.microsoft.com/dotnet/sdk:8.0@sha256:3bc4c8f13482237ab906d38dd9e290b4b1a093a2653ab3c28cca710b46510b9d
=> [app internal] load build context
=> => transferring context: 9.24kB
=> [app base 1/2] FROM mcr.microsoft.com/dotnet/aspnet:8.0@sha256:72bd33dd8f9829cf9681f0a6bc4b43972ec4860a9560ad2b9f4872b548af0add
=> CACHED [app base 2/2] WORKDIR /app
=> CACHED [app final 1/2] WORKDIR /app
=> CACHED [app build 2/12] WORKDIR /src
=> CACHED [app build 3/12] COPY [server/6thElement/6thElement.API/6thElement.API.csproj, 6thElement.API/]
=> CACHED [app build 4/12] COPY [server/6thElement/6thElement.Application/6thElement.Application.csproj, 6thElement.Application/]
=> CACHED [app build 5/12] COPY [server/6thElement/6thElement.Domain/6thElement.Domain.csproj, 6thElement.Domain/]
=> CACHED [app build 6/12] COPY [server/6thElement/6thElement.Infrastructure/6thElement.Infrastructure.csproj, 6thElement.Infrastructure/]
=> CACHED [app build 7/12] COPY [server/6thElement/6thElement.Persistance/6thElement.Persistance.csproj, 6thElement.Persistance/]
=> CACHED [app build 8/12] COPY [server/6thElement/Images, /app/Images]
=> CACHED [app build 9/12] RUN dotnet restore "6thElement.API/6thElement.API.csproj"
=> CACHED [app build 10/12] COPY server/6thElement .
=> CACHED [app build 11/12] WORKDIR /src/6thElement.API
=> CACHED [app build 12/12] RUN dotnet build "6thElement.API.csproj" -c Release -o /app/build
=> CACHED [app publish 1/1] RUN dotnet publish "6thElement.API.csproj" -c Release -o /app/publish /p:UseAppHost=false
=> CACHED [app final 2/2] COPY --from=publish /app/publish .
=> [app] exporting to image
=> => exporting layers
=> => writing image sha256:c5736ee5d5f99484d735aeb3baecfd508a4c616cfff245c73d259b6f2797994d
=> => naming to docker.io/library/6thelement-app
[+] Running 2/0
✔ Container 6thelement-db-1 Created
✔ Container 6thelement-app-1 Created
Attaching to app-1, db-1
db-1 | SQL Server 2019 will run as non-root by default.
db-1 | This container is running as user mssql.
db-1 | Your master database file is owned by mssql.
```

5. run npm build and npm start

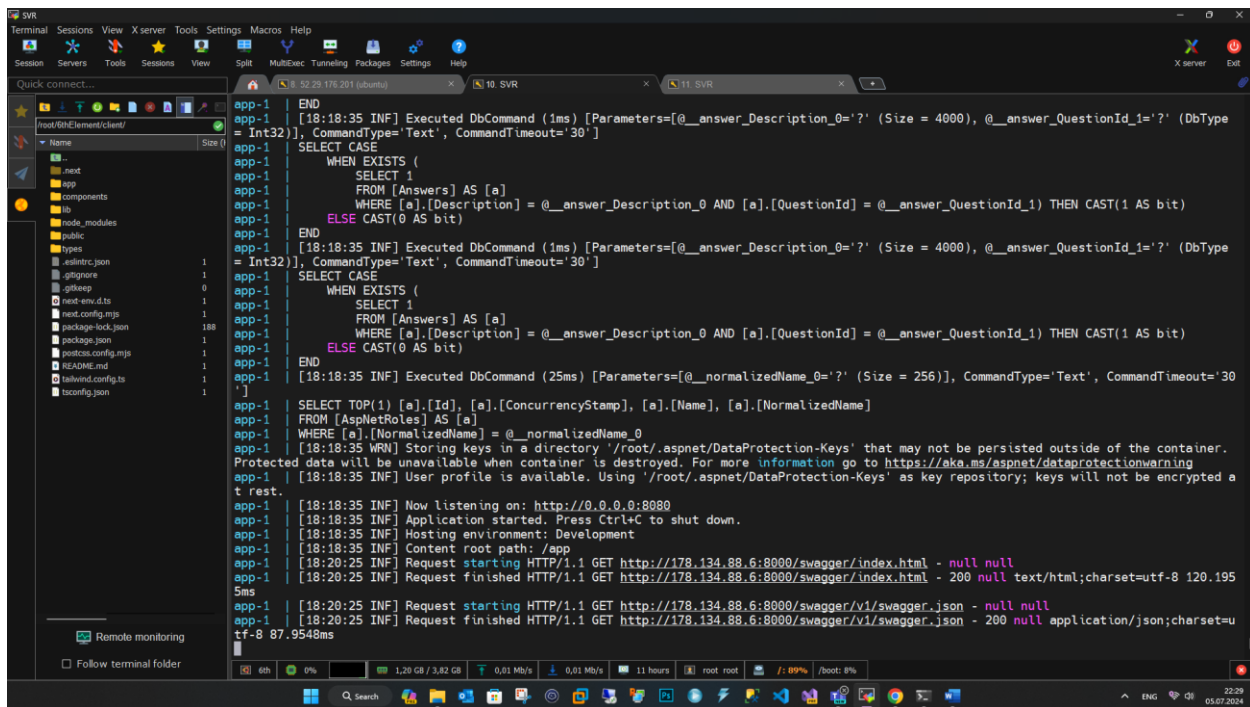
Runed API



Runed website



Igos



The screenshot shows a terminal window with a file explorer on the left and a terminal on the right. The file explorer shows a directory structure for a project, including files like .gitkeep, README.md, and various configuration files. The terminal displays a series of logs and database queries. The logs include messages about database commands, application startup, and HTTP requests. The database queries are SQL statements that select data from a table named 'Answers' based on specific criteria. The terminal also shows a warning message about data protection keys and a message about the application's content root path.

```
app-1 [18:18:35 INF] Executed DbCommand (1ms) [Parameters=[@_answer_Description_0='?' (Size = 4000), @_answer_QuestionId_1='?' (DbType = Int32)], CommandType='Text', CommandTimeout='30']
app-1 SELECT CASE
app-1 WHEN EXISTS (
app-1 SELECT 1
app-1 FROM [Answers] AS [a]
app-1 WHERE [a].[Description] = @_answer_Description_0 AND [a].[QuestionId] = @_answer_QuestionId_1 THEN CAST(1 AS bit)
app-1 ELSE CAST(0 AS bit)
app-1 END
app-1 [18:18:35 INF] Executed DbCommand (1ms) [Parameters=[@_answer_Description_0='?' (Size = 4000), @_answer_QuestionId_1='?' (DbType = Int32)], CommandType='Text', CommandTimeout='30']
app-1 SELECT CASE
app-1 WHEN EXISTS (
app-1 SELECT 1
app-1 FROM [Answers] AS [a]
app-1 WHERE [a].[Description] = @_answer_Description_0 AND [a].[QuestionId] = @_answer_QuestionId_1 THEN CAST(1 AS bit)
app-1 ELSE CAST(0 AS bit)
app-1 END
app-1 [18:18:35 INF] Executed DbCommand (25ms) [Parameters=[@_normalizedName_0='?' (Size = 256)], CommandType='Text', CommandTimeout='30']
app-1 SELECT TOP(1) [a].[Id], [a].[ConcurrencyStamp], [a].[Name], [a].[NormalizedName]
app-1 FROM [AspNetRoles] AS [a]
app-1 WHERE [a].[NormalizedName] = @_normalizedName_0
app-1 [18:18:35 WRN] Storing keys in a directory '/root/.aspnet/DataProtection-Keys' that may not be persisted outside of the container. Protected data will be unavailable when container is destroyed. For more information go to https://aka.ms/aspnet/dataprotectionwarning
app-1 [18:18:35 INF] User profile is available. Using '/root/.aspnet/DataProtection-Keys' as key repository; keys will not be encrypted at rest.
app-1 [18:18:35 INF] Now listening on: http://0.0.0.0:8080
app-1 [18:18:35 INF] Application started. Press Ctrl+C to shut down.
app-1 [18:18:35 INF] Hosting environment: Development
app-1 [18:18:35 INF] Content root path: /app
app-1 [18:20:25 INF] Request starting HTTP/1.1 GET http://178.134.88.6:8080/swagger/index.html - null null
app-1 [18:20:25 INF] Request finished HTTP/1.1 GET http://178.134.88.6:8080/swagger/index.html - 200 null text/html; charset=utf-8 120.195 5ms
app-1 [18:20:25 INF] Request starting HTTP/1.1 GET http://178.134.88.6:8080/swagger/v1/swagger.json - null null
app-1 [18:20:25 INF] Request finished HTTP/1.1 GET http://178.134.88.6:8080/swagger/v1/swagger.json - 200 null application/json; charset=utf-8 87.9548ms
```

Install app on AWS

1. Create instance
2. run GitLab CI/CD

```
stages:
  - deploy

deploy:
  stage: deploy
  before_script:
    - apt-get update
    - apt-get install -y openssh-client
    - eval $(ssh-agent -s)
    - echo "$SSH_PRIVATE_KEY" | tr -d '\r' | ssh-add -
    - mkdir -p ~/.ssh
    - chmod 700 ~/.ssh
    - echo "$SSH_KNOWN_HOSTS" > ~/.ssh/known_hosts
    - chmod 644 ~/.ssh/known_hosts
  script:
    - ssh ubuntu@$SERVER_IP "
      set -e &&
      sudo apt-get update &&
      sudo apt-get install -y docker.io &&
```

```
        sudo curl -L
\"https://github.com/docker/compose/releases/latest/download/docker-compose-
\"$(uname -s)-$(uname -m)\" -o /usr/local/bin/docker-compose &&
        sudo chmod +x /usr/local/bin/docker-compose &&
        curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh
| bash &&
        export NVM_DIR=$HOME/.nvm &&
        [ -s \"$NVM_DIR/nvm.sh\" ] && \\. \"$NVM_DIR/nvm.sh\" &&
        nvm install 20.3.1 &&
        nvm use 20.3.1 &&
        nvm alias default 20.3.1 &&
        rm -rf /home/ubuntu/6thElement &&
        git clone https://gitlab.com/tbc6thelement/6thElement.git
/home/ubuntu/6thElement &&
        cd /home/ubuntu/6thElement &&
        sudo /usr/local/bin/docker-compose up -d &&
        cd client &&
        npm install &&
        npm run build &&
        pm2 restart all || pm2 start npm --name \"6thElement\" -- start
    \"
only:
    - main
```

