Raport z projektu

Sieć Jako Usługa

Projekt kontenera developerskiego

Konfiguracja repozytorium

Wykonane zadania:

- Utworzyłem forka repozytorium bazowego na swoim koncie GitHub.
- Projekt otworzyłem lokalnie w Visual Studio Code.
- Zbudowałem obraz Dockera poleceniem:
  `docker build -t sjuprojekt .`
- Uruchomiłem kontener:
  `docker run -it --rm -v .:/home/vscode/workspace sjuprojekt bash`
- Sprawdziłem dostęp do plików projektu:
  `ls /home/vscode/workspace`
- Następnie połączyłem repozytorium lokalne z GitHubem za pomocą: `git remote add origin`

Modyfikacje wykonane w Dockerfile

Zainsatlowanie pakietów w środowisku Dockera

- W środowisku VS Code skorzystałem z rozszerzenia Dev Containers, które pozwala na konfigurację kontenera bezpośrednio z poziomu edytora.
- Zmodyfikowałem plik `Dockerfile`, aby instalował wszystkie potrzebne pakiety:
  - `qiskit`
  - `matplotlib`
  - `pillow`
  - `pycryptodomex`
  - `cryptography`
  - oraz także: `ipykernel`, `jupyter`
- Instalacja wykonałem za pomocą polecenia:

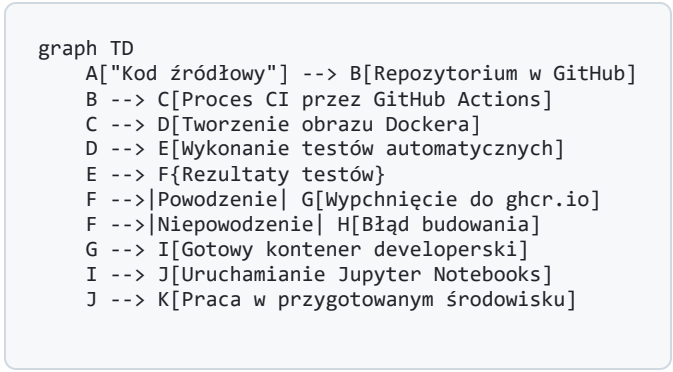
Konfiguracja kontenera deweloperskiego

Wygląd pliku `.devcontainer/devcontainer.json`

```
{
  "workspaceMount": "source=${localWorkspaceFolder},target=/home/vscode/workspace,type=bind,consistency=cached",
  "workspaceFolder": "/home/vscode/workspace",
  "name": "Projekt-SJU",
  "image": "ghcr.io/dthrnnr/projekt_sju:latest",
  "customizations": {
    "vscode": {
      "extensions": [
        "ms-python.python",
        "ms-toolsai.jupyter",
        "yzhang.markdown-all-in-one",
        "marp-team.marp-vscode",
        "github.vscode-github-actions"
      ]
    }
  },
  "postCreateCommand": "pip install --no-cache-dir -r requirements.txt && uname -a && python --version && pip --version",
  "remoteUser": "vscode"
}
```

Zadania wykonane w VSCode

Diagram pracy



Jupyter Notebooks

Za pomocą środowiska lokalnego (VSCode)

- Do pracy z notebookami korzystałem z rozszerzenia `Jupyter` dostępnego w edytorze Visual Studio Code.
- Pliki `.ipynb` uruchamiałem i edytowałem lokalnie wewnątrz kontenera developerskiego.
- Środowisko było oparte na wcześniej przygotowanym obrazie Dockera oraz konfiguracji zapisanej w pliku `devcontainer.json`.
- Jako kernel wykorzystywałem wirtualne środowisko . `venv`, co zapewniało dostęp do wszystkich wymaganych zależności.
- Notatniki wykorzystywały bibliotekę `Qiskit` do przesyłania zadań na komputer kwantowy IBM Quantum.

Wyzwania i doświadczenia

Najtrudniejsze wyzwania:

- Skonfigurowałem GitHub Actions z odpowiednimi uprawnieniami, aby możliwa była publikacja obrazu Dockera do rejestru.
- Na etapie budowy obrazu rozwiązywałem błędy pojawiające się podczas procesu kompilacji.
- Dbałem również o utrzymanie spójności wersji obrazu pomiędzy lokalnym kontenerem developerskim a obrazem publikowanym w rejestrze kontenerów.

Wiedza wyciagnięta z zadania:

- W jaki sposób efektywnie wykorzystywać kontenery w środowisku pracy z VSCode i Dockerem

Podsumowanie

Zrealizowane elementy:

- Konteneryzacja środowiska programistycznego
- Automatyzacja CI/CD z GitHub Actions
- Publikacja obrazów do GitHub Container Registry
- Integracja Jupyter Notebooków w kontenerze