

Build and Deployment Documentation

Purpose

This documentation provides a comprehensive guide on building and deploying our project, covering the setup, build process, deployment to an HPC cluster, and running the ETL and QA processes. This guide aims to be accessible to both technical and non-technical readers.

Build and Deployment Process

Our project uses GitHub Actions to automate the build process. This section explains the steps involved, as defined in the `deep-cncf.yaml` file.

GitHub Actions Workflow

The build workflow is triggered by three events:

- Pushing changes to the `main` branch.
- Opening a pull request targeting the `main` branch.
- Manually via the GitHub Actions interface (`workflow_dispatch`).

The workflow consists of three jobs that run sequentially: License Check, Lint Check, and Build and Test.

1. License Check:

1. **Checkout Code:** Retrieves the code from the repository.
2. **Set up Python:** Configures Python 3.11 with caching for dependencies.
3. **Install Dependencies:** Installs required Python packages.
4. **Check Copyright:** Uses `pilosus/action-pip-license-checker` to verify license compliance.
5. **Print Report:** Outputs the results of the license check.

2. Lint Check:

1. **Checkout Code:** Retrieves the code from the repository.
2. **Set up Python:** Configures Python 3.11 with caching for dependencies.
3. **Install Pylint:** Installs the `pylint` tool.
4. **Run Linter:** Analyzes the code for style issues using Pylint.

3. Build and Test:

1. **Checkout Code:** Retrieves the code from the repository.

2. **Set up Python:** Configures Python 3.11 with caching for dependencies.
3. **Install Dependencies:** Installs required Python packages.
4. **Run Tests:** Executes the unit tests using the `unittest` module.

After running the pipeline, if everything works perfectly well, we get all the checks done and it is masked as green. The pipeline turns red if there is an error and the pipeline fails. The error can be checked from the particular check from where the pipeline is broken. To identify the issue, we can directly look into the log file of the Github Action of the latest run.

FAU HPC Cluster

To deploy the project on an HPC (High-Performance Computing) cluster:

1. Access the Cluster:

- Apply for an account.
- Login at [HPC Portal](#).
- Accept the invitation to use HPC.

2. Generate SSH Key:

Use the following command to generate an SSH key:

```
ssh-keygen -t ed25519 -f ~/.ssh/id_ed25519_nhr_fau
```

Upload the public key to the HPC portal.

3. SSH Configuration:

Add the following to your `~/.ssh/config` file:

```
Host csahr.nhr.fau.de
  HostName csahr.nhr.fau.de
  User <HPC account>
  IdentityFile ~/.ssh/id_ed25519_nhr_fau
  IdentitiesOnly yes
  PasswordAuthentication no
  PreferredAuthentications publickey
  ForwardX11 no
  ForwardX11Trusted no

Host tinyx.nhr.fau.de
  HostName tinyx.nhr.fau.de
  User <HPC account>
  ProxyJump csahr.nhr.fau.de
```

```
IdentityFile ~/.ssh/id_ed25519_nhr_fau
IdentitiesOnly yes
PasswordAuthentication no
PreferredAuthentications publickey
ForwardX11 no
ForwardX11Trusted no
```

4. Login:

- First login to `csnhr.nhr.fau.de` .
- Subsequently, use `tinyx.nhr.fau.de` .

Python Configuration

Load Python on the HPC cluster using the appropriate module command for your environment.

Running ETL and QA Processes

The `run-all.sh` script automates environment setup, ETL (Extract, Transform, Load) processes, and QA (Question and Answer) generation tasks. Follow these steps to execute the script.

Prerequisites

1. Environment Variables:

Create a `.env` file in the root directory with the following content:

```
GITHUB_TOKEN=<YOUR_GITHUB_TOKEN>
HF_TOKEN=<YOUR_HUGGING_FACE_TOKEN>
```

Replace `<YOUR_GITHUB_TOKEN>` and `<YOUR_HUGGING_FACE_TOKEN>` with your tokens.

2. Execution:

Ensure the script is executed from the root directory:

```
./run-all.sh [etl] [qa] <data_set_id>
```

Example command to execute the ETL process:

```
./run-all.sh etl SuperOrganization/WorldDataset
```

Script Workflow

1. Virtual Environment Setup:

- Creates and activates a Python virtual environment.
- Installs required packages from `requirements.txt`.

2. Loading Environment Variables:

- Loads environment variables from the `.env` file.

3. ETL Process:

- Executes scripts to download and process data.
- Runs a Scrapy spider to scrape data.
- Continues processing and augments the data.
- Uploads the final data to Hugging Face.

4. QA Generation:

- Executes scripts to generate QA data.
- Uploads the QA data to Hugging Face.