Data Engineering and MLOps in Business

GitHub branches and Pages & API(2)

Primoz Konda

AAUBS

March 18, 2025

pk@business.aau.dk

Outline

1 Intro

2 GitHub Branches

3 Github Pages

4 API (2)

Where did we end yesterday?

- ?
- Questions?

Introduction to GitHub Branches

- A branch is a lightweight, movable pointer to a commit.
- Allows you to work on different features or fixes without affecting the main codebase.
- Enables collaboration and parallel development.

Why Use Branches?

- Isolate new development from the main codebase.
- Enables feature development without disturbing the stable version.
- Makes code reviews and testing easier.
- Facilitates collaboration by allowing developers to work independently.

How Branches Work

- The ‘main‘ branch is typically the stable version.
- New branches are created from the main branch:
  - ‘git checkout -b new-feature‘
  - ‘git branch new-feature‘
- Changes can be merged back into the main branch:
  - ‘git merge new-feature‘
  - ‘git rebase main‘

Branching Strategies

- **Feature branching** – Create a branch for each feature.
- **Release branching** – Create a branch for each release.
- **Hotfix branching** – Create a branch for urgent fixes.
- **GitFlow** – A structured branching model for release and feature development.

Best Practices

- Keep branches short-lived and focused.
- Regularly sync with the main branch.
- Write meaningful commit messages.
- Use protected branches to prevent accidental merges.

What is GitHub Pages?

- A static site hosting service provided by GitHub.
- Hosts HTML, CSS, and JavaScript files directly from a repository.
- Ideal for personal, organization, or project websites.

How Does GitHub Pages Work?

- Serve content from a specific branch in a GitHub repository.
- Optionally integrates with static site generators like Jekyll.
- Publishes the site at `<username>.github.io` or a custom domain.

Benefits of Using GitHub Pages

- Free hosting with GitHub's infrastructure.
- Seamless integration with version control.
- Supports custom domains and HTTPS.
- Simplifies the process of publishing static websites.

Setting up GitHub Pages (1)

- **Navigate to Repository Settings:**
  - Go to your repository.
  - Click on the "Settings".
- **Configure GitHub Pages:**
  - In the "Code and automation" section of the sidebar, click "Pages".
  - Under "Build and deployment", set the "Source" to "Deploy from a branch".
  - Select the branch you want to publish (e.g., main) and choose the folder:
    - '/' for root or '/docs' if your site files are there.
  - Click "Save".

Setting up GitHub Pages (2)

- **Add an index.html File:**
  - Ensure your repository contains an ‘index.html‘ file at the root or specified folder.
  - This file will serve as the homepage.
- **Access Your Published Site:**
  - After a few minutes, your site will be available at:
    - `https://saoter.github.io/DR_DK_MLOps/`

Using Jekyll with GitHub Pages

- Jekyll is a static site generator integrated with GitHub Pages.
- Allows the use of Markdown for content creation.
- Supports themes and plugins for customization.

Best Practices

- Keep content organized and follow a consistent structure.
- Regularly update dependencies and themes.
- Monitor site performance and fix broken links.
- Use analytics to understand visitor engagement.

What is FastAPI?

- A modern, fast (high-performance) web framework for building APIs with Python.
- Based on standard Python type hints.
- Built on Starlette and Pydantic.
- Automatically generates interactive Swagger documentation.

What is Uvicorn?

- An ASGI (Asynchronous Server Gateway Interface) server.
- Used to serve FastAPI applications.
- Handles high-throughput connections efficiently with low latency.
- Command to start FastAPI with Uvicorn:
  - ‘uvicorn main:app –reload‘

What is Gunicorn?

- A WSGI (Web Server Gateway Interface) server.
- Can be used with Uvicorn to serve FastAPI in production.
- Handles multiple workers and better load balancing.
- Command to start FastAPI with Gunicorn and Uvicorn:
  - ‘gunicorn -k uvicorn.workers.UvicornWorker main:app‘

Development vs Production

- Development:
  - Use Uvicorn directly with '–reload' for hot reloading.
  - Easier for debugging.
- Production:
  - Use Gunicorn with Uvicorn workers for scalability.
  - Configure logging, error handling, and timeouts.
  - Use Nginx as a reverse proxy for better performance and security.