Introduction to CI/CD SDTM aCRF.pdf application using GitHub

Derek Li

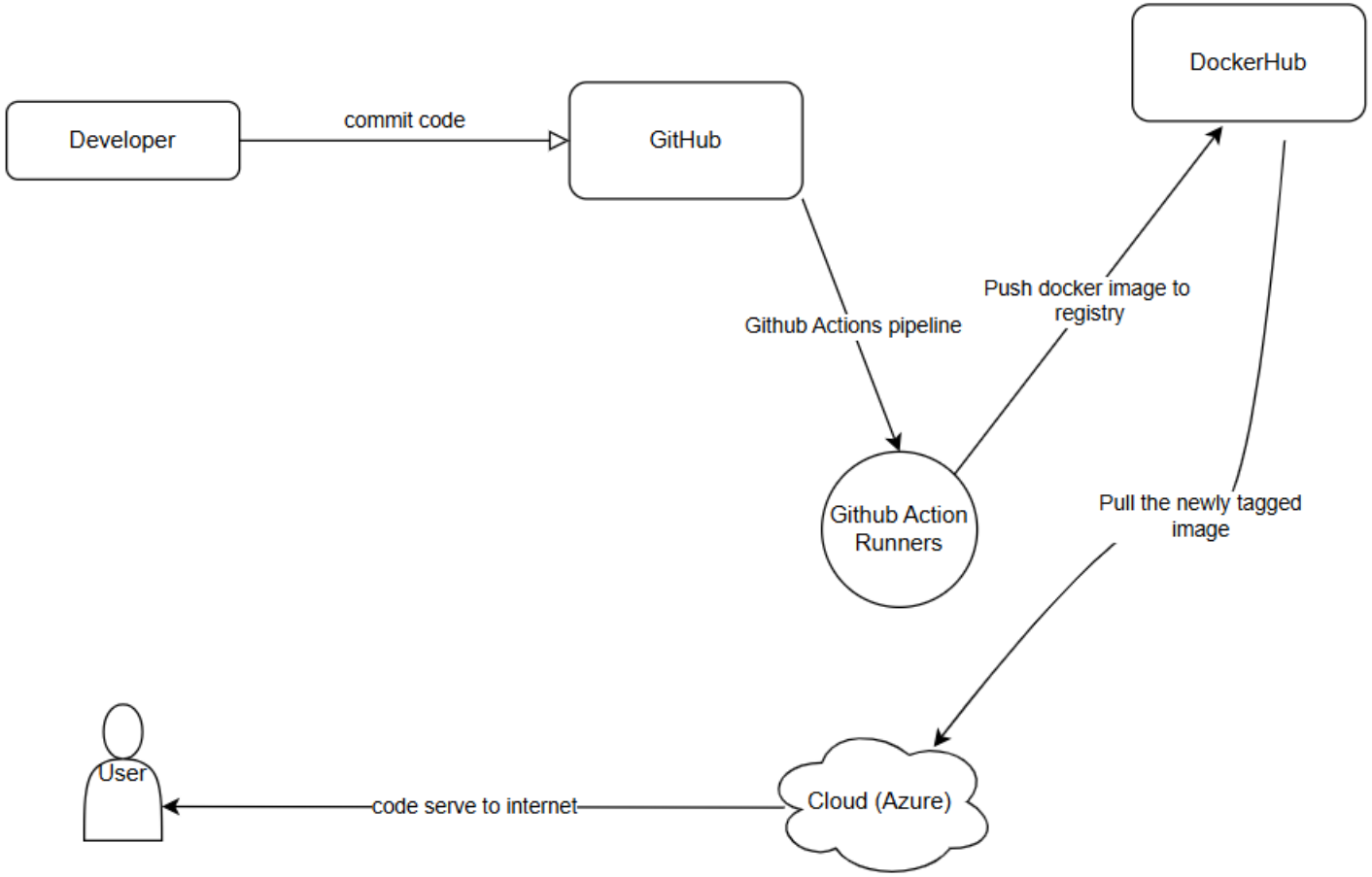
Agenda

- Background
- Sample CI/CD pipeline flow diagram using github actions
- Introduction to GitHub CI/CD
- Workflow File
- SDTM aCRF.pdf tool example
- Key Features of CI/CD with GitHub
- Benefits of Implementing CI/CD with GitHub
- Summary

Background

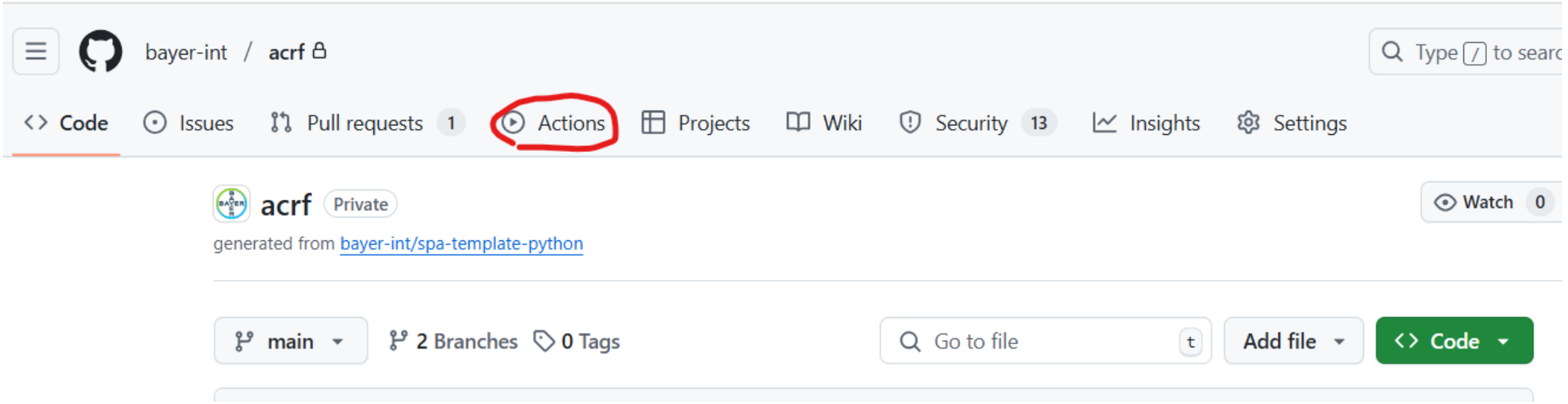
- Development tool is part of statistical analyst live, however, how to deploy a tool after developing or updating after release usually a challenge for us.
- We store source codes on GitHub
- GitHub provides Continuous Integration (CI) and Continuous Delivery (CD) pipelines within its ecosystem.

Sample CI/CD pipeline flow diagram using github actions

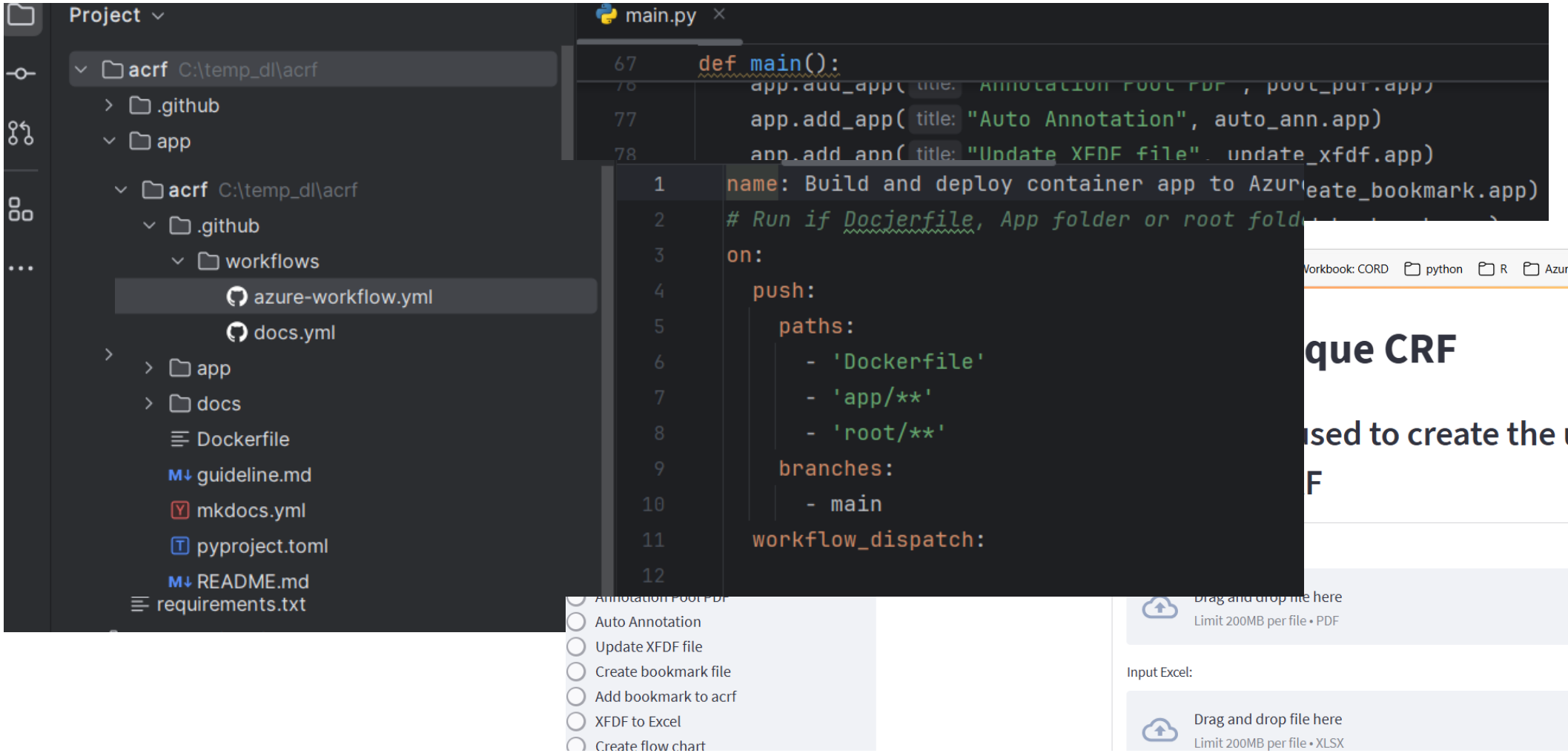


Introduction to GitHub CI/CD

- GitHub's CI/CD process is primarily implemented using GitHub Actions with workflow file.
- GitHub Actions is an automation tool that allows you to define, run, and manage workflows in your GitHub repository.
- Automate various tasks in software development, such as building, testing, and deploying.



Introduction to GitHub CI/CD



Workflow File

```yaml
name: CI/CD Pipeline  # Name of the workflow

on:  # Events that trigger the workflow
  push:
    branches:
      - main  # Trigger on push to the main branch
  pull_request:
    branches:
      - main  # Trigger on pull request to the main branch

jobs:  # Define jobs
  build:  # Job name
    runs-on: ubuntu-latest  # Specify the runner environment

    steps:  # Define steps
      - name: Checkout code  # Step name
        uses: actions/checkout@v2  # Use the actions/checkout
action to fetch code
```

```yaml
    - name: Set up Node.js  # Set up Node.js environment
      uses: actions/setup-node@v2
      with:
        node-version: '14'  # Specify Node.js version

    - name: Install dependencies  # Install dependencies
      run: npm install

    - name: Run tests  # Run tests
      run: npm test

    - name: Build project  # Build the project
      run: npm run build

    - name: Deploy  # Deploy
      run: npm run deploy
```

Workflow File

- **name**: The name of the workflow, used to identify it.
- **on**: Defines the events that trigger the workflow. Common events include push, pull_request, schedule, and workflow_dispatch.
- **jobs**: Defines the jobs in the workflow. Each job has a unique name (e.g., build) and runs on a specified runner (e.g., ubuntu-latest).
- **runs-on**: Specifies the virtual machine environment for the job. Common environments include ubuntu-latest, windows-latest, and macos-latest.
- **steps**: Defines the steps in a job. Each step can execute an action (uses) or run a command (run).
- **uses**: Uses a predefined action. GitHub provides many official actions.
- **run**: Runs a shell command in the step. You can execute any command, such as installing dependencies, running tests, or building the project.

SDTM aCRF.pdf tool example

1. **Step 1: Define Workflows**: Create a .github/workflows/ directory in your repository and add YAML files to define your CI/CD workflows.

acrf / .github / workflows / **azure-workflow.yml**

eryvx Update azure-workflow.yml

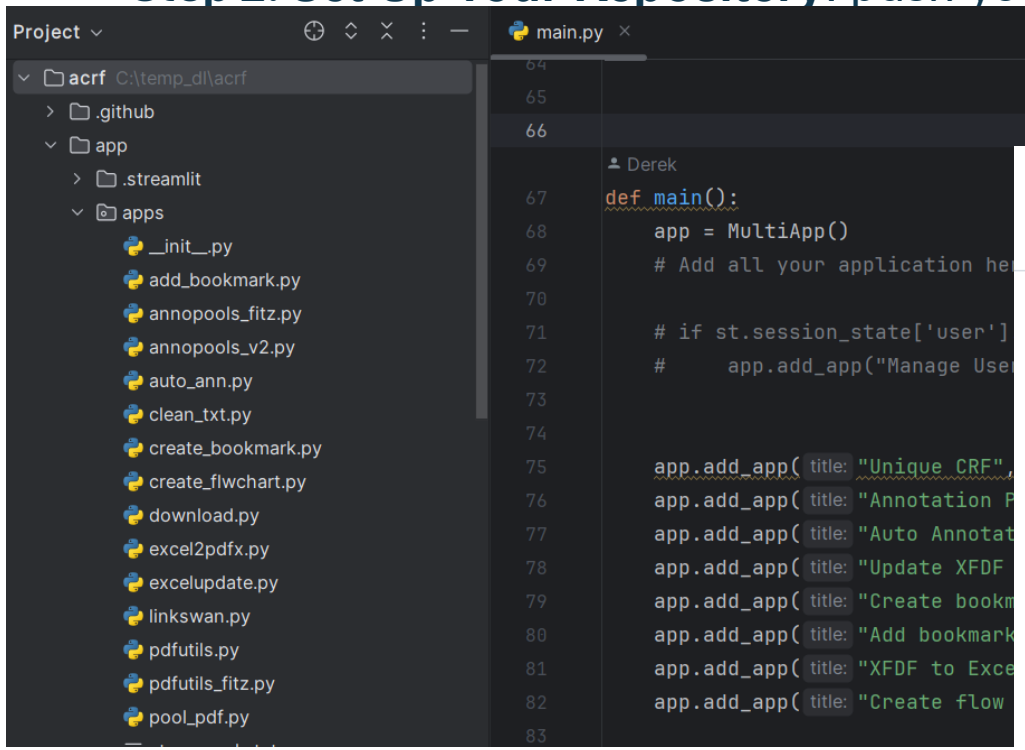
Code | Blame | 69 lines (64 loc) · 2.13 KB

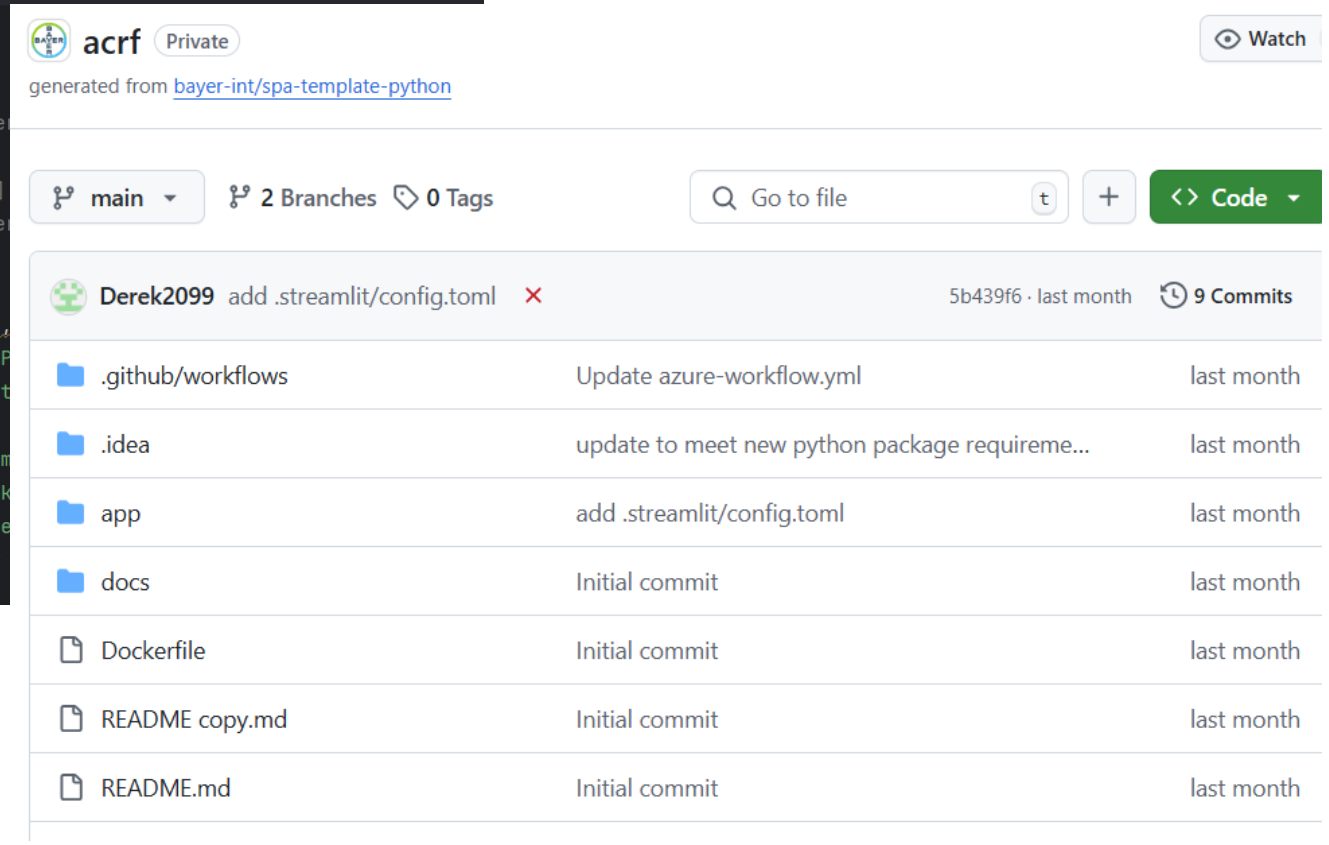
```yaml
name: Build and deploy container app to Azure Web App
# Run if Docjerfile, App folder or root folder is changed
on:
  push:
    paths:
      - 'Dockerfile'
      - 'app/**'
      - 'root/**'
    branches:
      - main
  workflow_dispatch:

jobs:
  build:
    env:
# Using build-in connection to artifactory.bayer.com
      ARTIFACTORY_REPOSITORY: "repository4bayer-platformservices-docker-dev-dps.jfrog.io
    runs-on: 'ubuntu-latest'
    steps:
    - uses: actions/checkout@v3
    - name: Set up Docker Buildx
      uses: docker/setup-buildx-action@v2
    - name: Log in to registry
      uses: docker/login-action@v2
```

```yaml
  deploy-prd:
    runs-on: ubuntu-latest
    needs: deploy-dev
    environment:
      name: 'production'
      url: ${{ steps.deploy-to-webapp.outputs.webapp-url }}

    steps:
    - name: Deploy to Azure Web App
      id: deploy-to-webapp
      uses: azure/webapps-deploy@v2
      with:
        app-name: ${{ github.event.repository.name }}-prd
        slot-name: 'production'
        publish-profile: ${{ secrets.PUBLISH_PROFILE_PRD }}
```

SDTM aCRF.pdf tool example

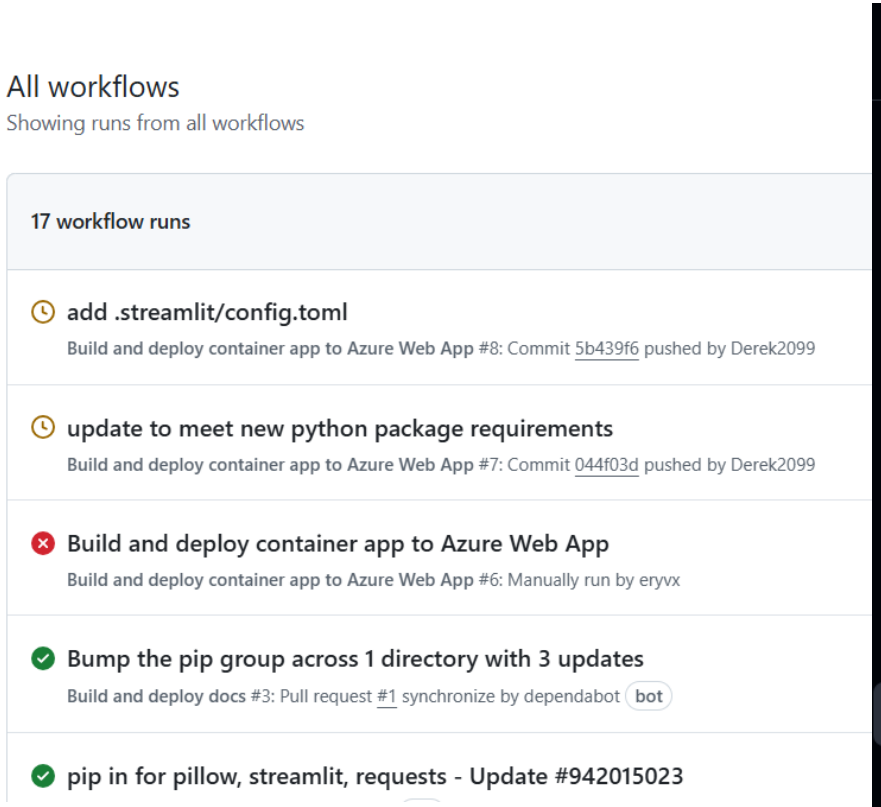
Step 2: Set Up Your Repository: push your codes from local to github respository.

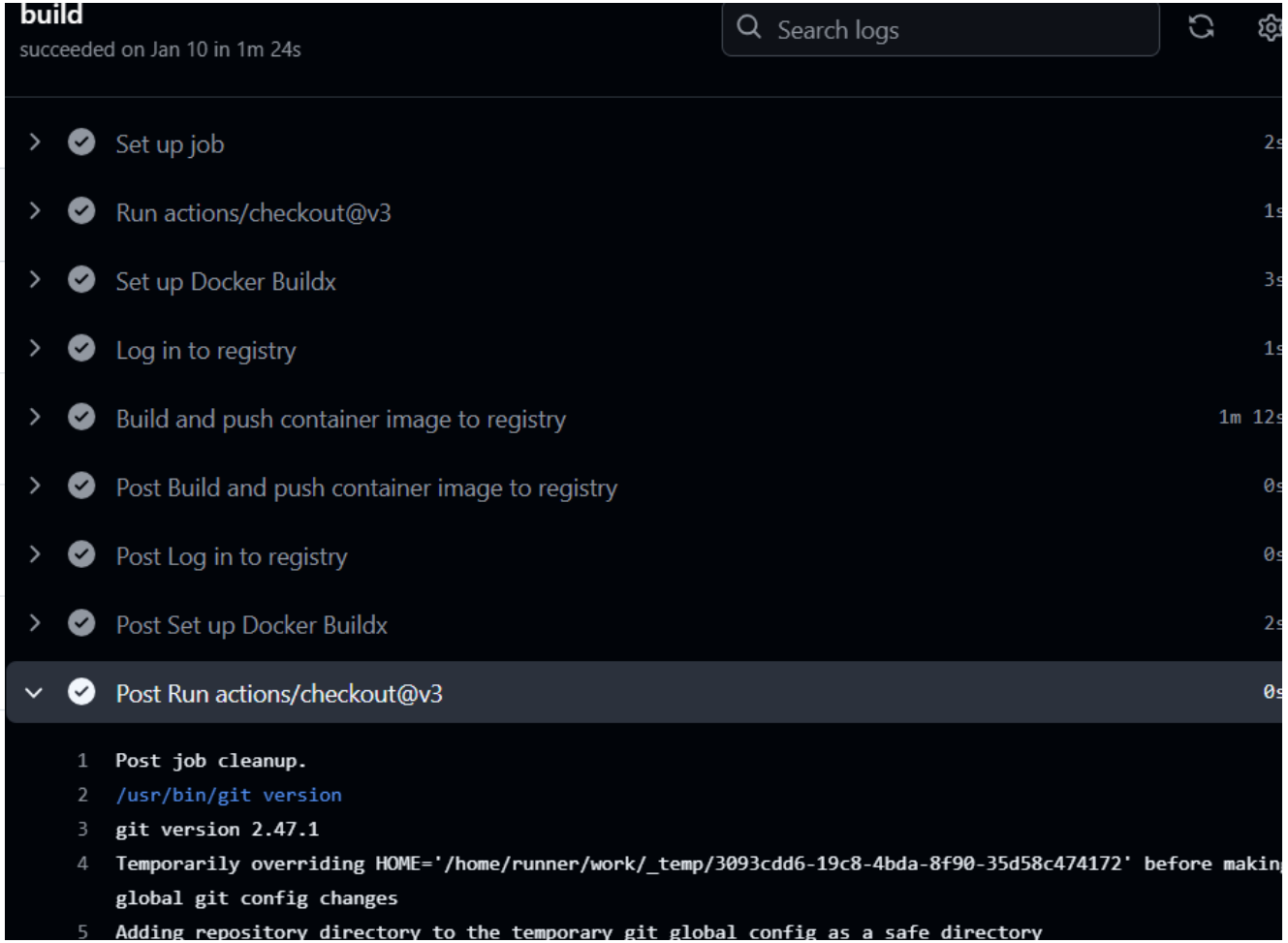




SDTM aCRF.pdf tool example

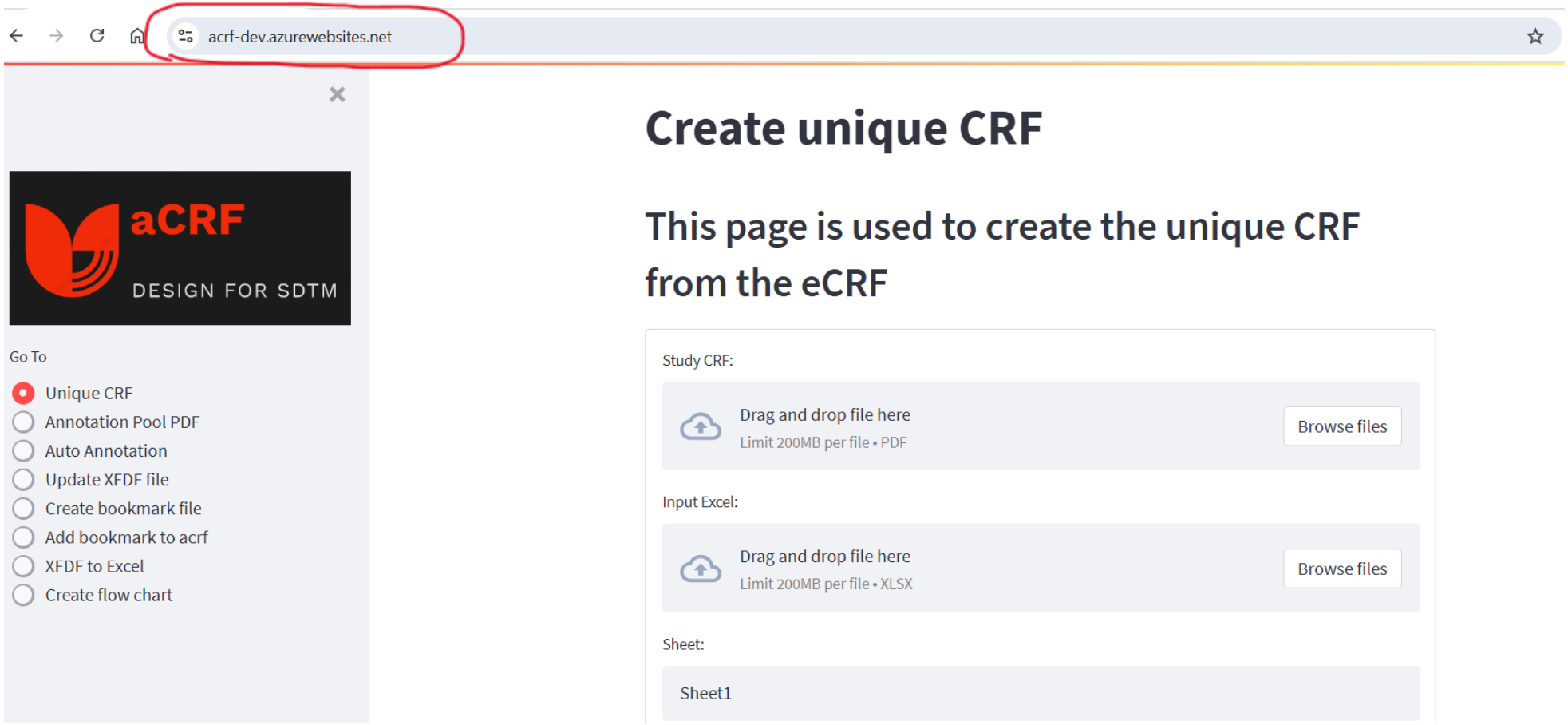
1. **Step 3: View all workflow runs:** check details through github Actions





SDTM aCRF.pdf tool example

1. **Step 4: Final results:**



Key Features of CI/CD with GitHub

1. **Automated Workflows**: Define custom workflows using YAML files to specify when and how your CI/CD pipeline should run. These workflows can be triggered by events such as pull requests, pushes, or even scheduled times.

2. **Integration with GitHub Ecosystem**: Seamlessly integrate with GitHub's features, including pull requests, issues, and releases, to enhance collaboration and streamline the development process.

3. **Customizable Actions**: Use pre-built actions or create custom ones to extend the functionality of your CI/CD pipeline. Actions can be used for tasks like running tests, deploying to cloud platforms, or notifying teams via Slack.

4. **Scalability**: GitHub Actions supports scaling your workflows to handle large and complex projects, ensuring that your CI/CD pipeline can grow with your needs.

5. **Security and Compliance**: GitHub provides robust security features, including secrets management and code scanning, to ensure that your CI/CD pipeline adheres to security best practices.

Benefits of Implementing CI/CD with GitHub

- **Faster Development Cycles**: Automate repetitive tasks and reduce manual intervention, allowing teams to focus on writing code and delivering features.
- **Improved Code Quality**: Regular testing and integration ensure that issues are caught early, leading to higher-quality code.
- **Consistent Deployments**: Standardize your deployment process to minimize errors and ensure consistent releases across environments.
- **Enhanced Collaboration**: Streamline workflows and provide visibility into the development process, fostering better teamwork and communication.

Summary

- GitHub Actions provides a powerful CI/CD platform.
- By defining Workflow files, you can automate the entire process from code submission to deployment.
- The structure of Workflow files is clear, easy to understand, and maintainable.
- With support for a wide range of triggering events and actions, it can meet the automation needs of most projects.
- IT support is mandatory

Thank you!

//////////

Bye-Bye