

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

////////

Derek Li



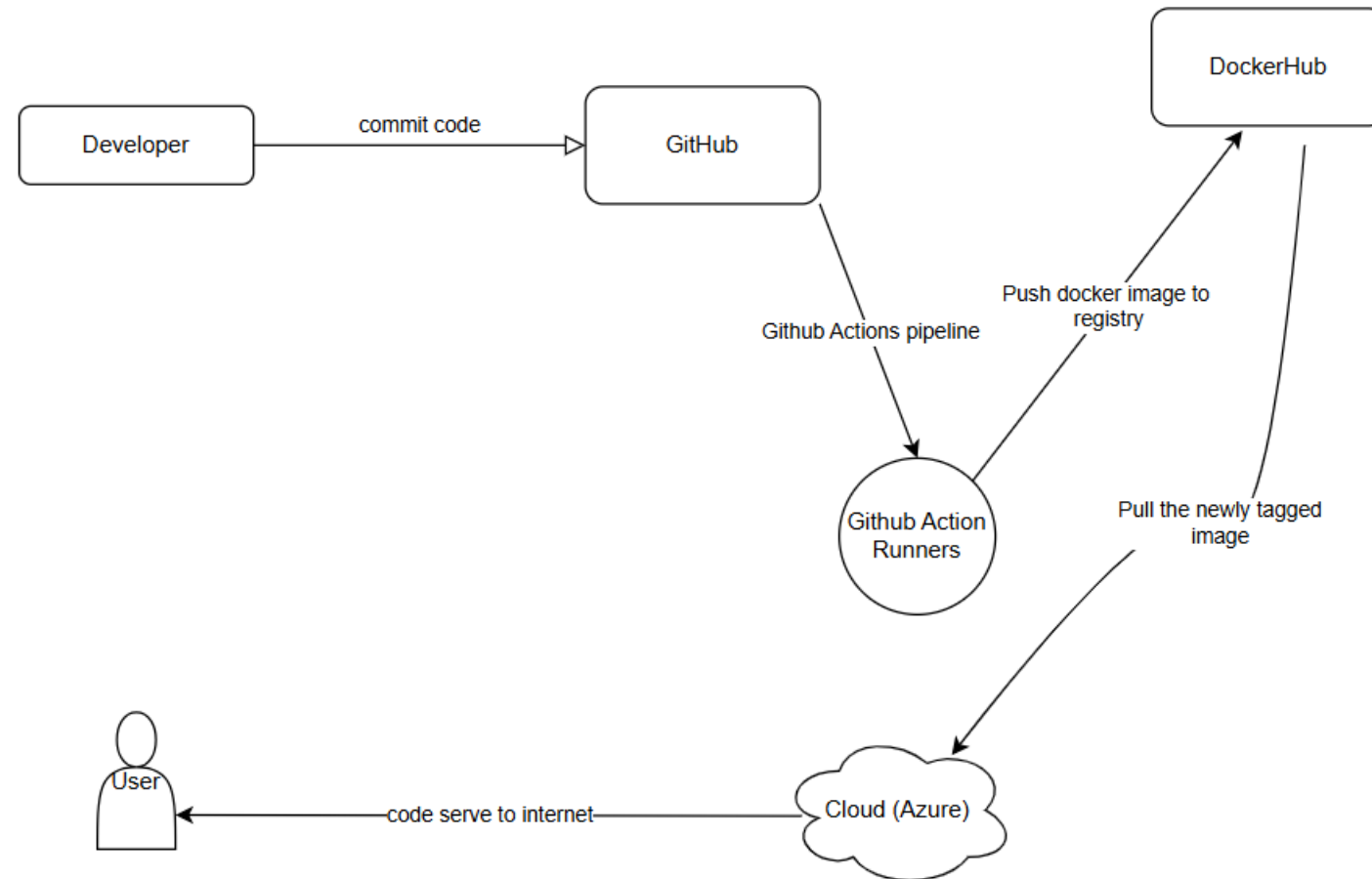
Agenda

- 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

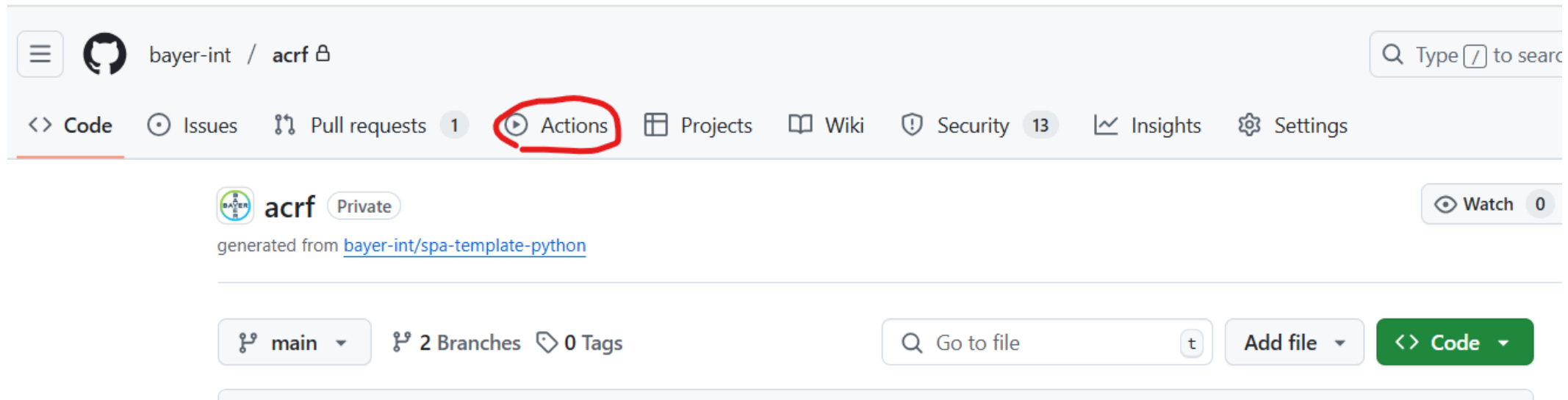
- Development tool is part of statistical analyst live, however, how to deploy a tool after developing or updating after release usually a challenge for us.
- We store source codes on GitHub
- GitHub provides Continuous Integration (CI) and Continuous Delivery (CD) pipelines within its ecosystem.

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

The screenshot shows a VS Code editor with three main components:

- Project Explorer (Left):** Displays the project structure for 'Project'. The 'acrf' folder is expanded, showing subfolders like '.github', 'app', and 'workflows'. The 'workflows' folder is selected, showing files like 'azure-workflow.yml' and 'docs.yml'.
- Code Editor (Center):** Shows the 'main.py' file. The code defines a 'main()' function that adds several apps to a container. The code is as follows:


```

67 def main():
68     app.add_app(title="Annotation PDF", pdf_app)
69     app.add_app(title="Auto Annotation", auto_ann.app)
70     app.add_app(title="Update XFDF file", update_xfdf.app)
71     app.add_app(title="Create bookmark", create_bookmark.app)
72
73 # Run if Dockerfile, App folder or root folder
74 if __name__ == '__main__':
75     main()

```
- Workflow File (Right):** Shows the 'azure-workflow.yml' file. The content is as follows:


```

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

```

Below the code editor, there is a list of tasks or actions, each with a radio button:

- ☐ Annotation PDF
- ☐ Auto Annotation
- ☐ Update XFDF file
- ☐ Create bookmark file
- ☐ Add bookmark to acrf
- ☐ XFDF to Excel
- ☐ Create flow chart

On the far right, there is a sidebar with a search bar and a list of files or folders, including 'python', 'R', 'Azure', 'SAS', and 'js'.

Workflow File

```
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
```

- 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.

[acrif](#) / [.github](#) / [workflows](#) / [azure-workflow.yml](#) 

 **eryvx** Update azure-workflow.yml

Code

Blame

69 lines (64 loc) · 2.13 KB

```
1  name: Build and deploy container app to Azure Web App
2  # Run if Dockerfile, App folder or root folder is changed
3  on:
4    push:
5      paths:
6        - 'Dockerfile'
7        - 'app/**'
8        - 'root/**'
9    branches:
10     - main
11 workflow_dispatch:
12
13 jobs:
14   build:
15     env:
16     # Using build-in connection to artifactory.bayer.com
17     ARTIFACTORY_REPOSITORY: "repository4bayer-platformservices-docker-dev-dps.jfrog.io
18     runs-on: 'ubuntu-latest'
19     steps:
20     - uses: actions/checkout@v3
21     - name: Set up Docker Buildx
22       uses: docker/setup-buildx-action@v2
23     - name: Log in to registry
24       uses: docker/login-action@v2
```

```
53  deploy-prd:
54    runs-on: ubuntu-latest
55    needs: deploy-dev
56    environment:
57      name: 'production'
58      url: ${{ steps.deploy-to-webapp.outputs.webapp-url }}
59
60    steps:
61    - name: Deploy to Azure Web App
62      id: deploy-to-webapp
63      uses: azure/webapps-deploy@v2
64    with:
65      app-name: ${{ github.event.repository.name }}-prd
66      slot-name: 'production'
67      publish-profile: ${{ secrets.PUBLISH_PROFILE_PRD }}
```

SDTM aCRF.pdf tool example

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

The image shows a code editor on the left and a GitHub repository page on the right. The code editor displays a file named `main.py` with the following Python code:

```
64
65
66
67 def main():
68     app = MultiApp()
69     # Add all your application here
70
71     # if st.session_state['user']
72     #     app.add_app("Manage User")
73
74
75     app.add_app(title="Unique CRF")
76     app.add_app(title="Annotation F")
77     app.add_app(title="Auto Annotat")
78     app.add_app(title="Update XFDF")
79     app.add_app(title="Create bookm")
80     app.add_app(title="Add bookmark")
81     app.add_app(title="XFDF to Exce")
82     app.add_app(title="Create flow")
83
```

The GitHub repository page shows the repository `acrf` (Private) generated from `bayer-int/spa-template-python`. It has 2 branches and 0 tags. The commit history shows the following files and their commit dates:

File	Commit Message	Commit Date
<code>.github/workflows</code>	Update azure-workflow.yml	last month
<code>.idea</code>	update to meet new python package requireme...	last month
<code>app</code>	add .streamlit/config.toml	last month
<code>docs</code>	Initial commit	last month
<code>Dockerfile</code>	Initial commit	last month
<code>README copy.md</code>	Initial commit	last month
<code>README.md</code>	Initial commit	last month

SDTM aCRF.pdf tool example

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

All workflows

Showing runs from all workflows

17 workflow runs

🕒 add .streamlit/config.toml

Build and deploy container app to Azure Web App #8: Commit [5b439f6](#) pushed by Derek2099

🕒 update to meet new python package requirements

Build and deploy container app to Azure Web App #7: Commit [044f03d](#) pushed by Derek2099

❌ Build and deploy container app to Azure Web App

Build and deploy container app to Azure Web App #6: Manually run by erylxx

✅ Bump the pip group across 1 directory with 3 updates

Build and deploy docs #3: Pull request [#1](#) synchronize by dependabot (bot)

✅ pip in for pillow, streamlit, requests - Update #942015023

build
succeeded on Jan 10 in 1m 24s

Search logs

> Set up job

> Run actions/checkout@v3

> Set up Docker Buildx

> Log in to registry

> Build and push container image to registry

> Post Build and push container image to registry

> Post Log in to registry

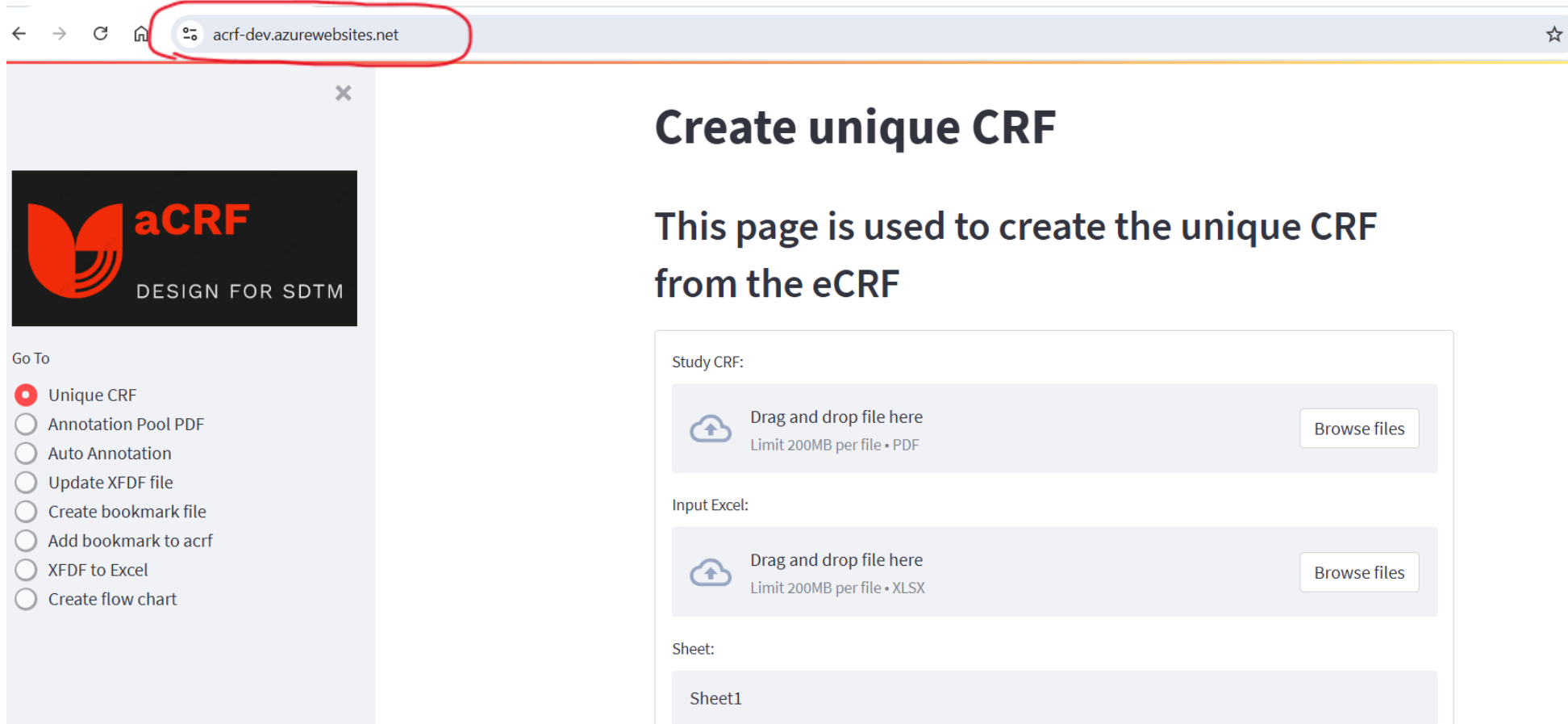
> Post Set up Docker Buildx

Post Run actions/checkout@v3

```
1 Post job cleanup.
2 /usr/bin/git version
3 git version 2.47.1
4 Temporarily overriding HOME='/home/runner/work/_temp/3093cdd6-19c8-4bda-8f90-35d58c474172' before making global git config changes
5 Adding repository directory to the temporary git global config as a safe directory
```

SDTM aCRF.pdf tool example

1. Step 4: Final results:



The screenshot shows a web browser window with the address bar displaying `acrf-dev.azurewebsites.net`, which is circled in red. The browser window contains a sidebar on the left and a main content area on the right.

Left Sidebar:

- Go To
- ☒ Unique CRF
- ☐ Annotation Pool PDF
- ☐ Auto Annotation
- ☐ Update XFDF file
- ☐ Create bookmark file
- ☐ Add bookmark to acrf
- ☐ XFDF to Excel
- ☐ Create flow chart

Main Content Area:

Create unique CRF

This page is used to create the unique CRF from the eCRF

Study CRF:

Drag and drop file here
Limit 200MB per file • PDF

Input Excel:

Drag and drop file here
Limit 200MB per file • XLSX

Sheet:

Sheet1

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

