Welcome to this Training Session with Theiagen Genomics

We will soon be getting started

Software Development Practices for Public Health Bioinformatics

Week 03: Bringing Changes into Production

A Mid-Atlantic Workforce Development Offering Provided by the Division of Consolidated Laboratory Services in Collaboration with Theiagen Genomics

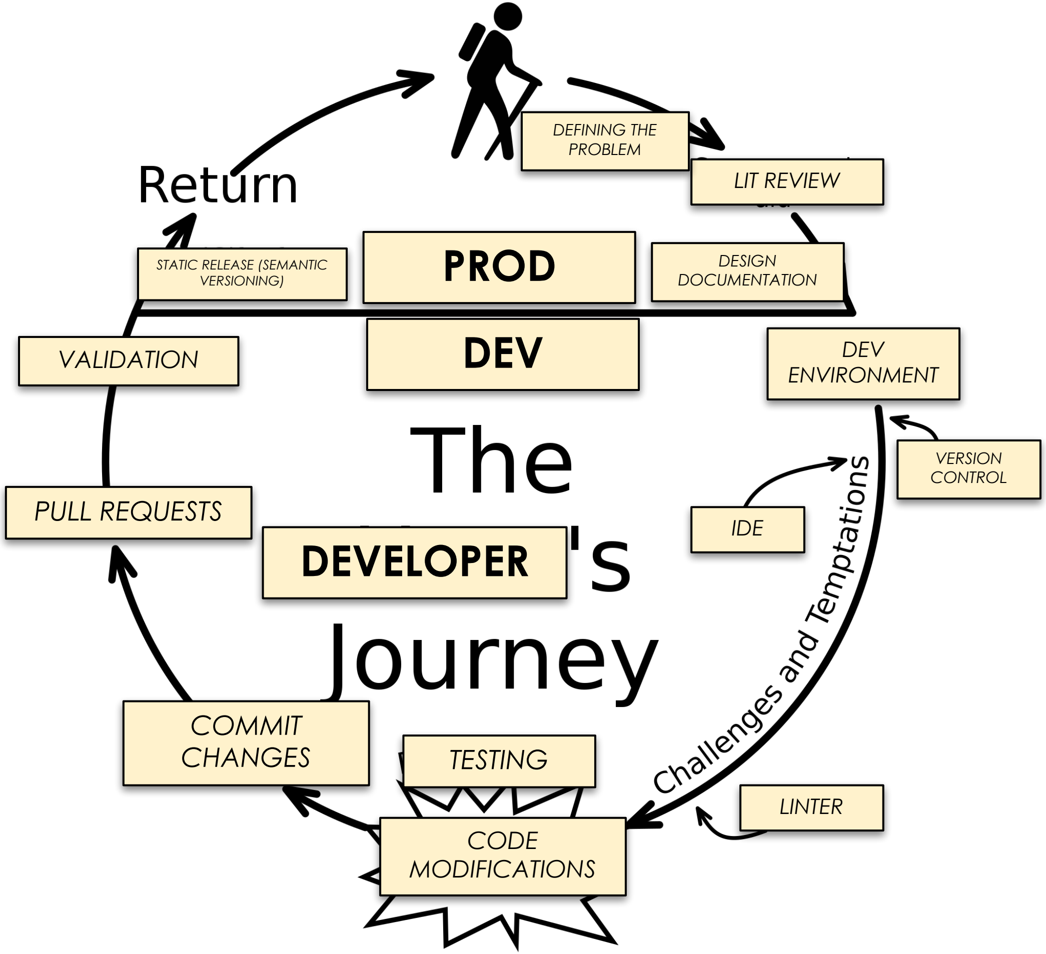
Course Introduction

Training Workshop Instructors

Michal Babinski, MSc

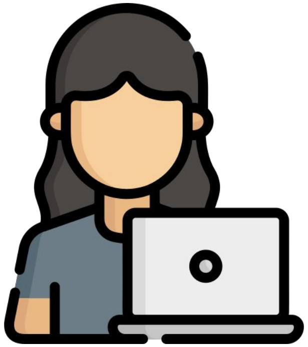
- Senior Bioinformatics Developer at Theiagen Genomics since 2024
- MSc in Bioinformatics
- BSc in Molecular Genetics and Genomics

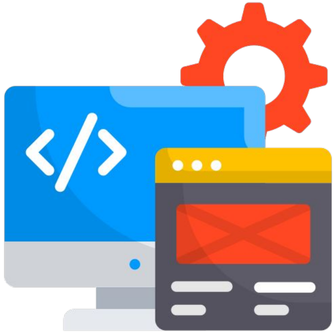
Week 1-2 Recap



The Developer’s Journey

Framework where a protagonist **enters into their dev environment**, faces challenges, gains new wisdom, and **brings changes into production**.



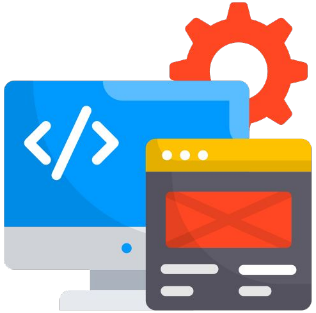


Byrne, C. (2017). The Hero's Journey. Wikipedia https://en.wikipedia.org/wiki/File:Heroesjourney.svg

Software Development Practices

Developer’s Journey

1. **Design Document**
   a. Clearly defining the problem and the proposed solution
2. **Development Environment**
   a. Separate from production
   b. Text editors and IDE’s
3. **Making Source Code Modifications**
   a. Small interactive changes (version control)
4. **Peer Review**
   a. Collaborative development teams
5. **Bringing Changes into Production**
   a. Final testing
   b. Static version releases



Design Document

Summary

- The design document is a vital tool that **defines the project's problem and solution**, informed by literature review and community feedback.
    - It ensures **clear communication** and alignment among stakeholders.

Development Environment

Summary

- Separating development and production environments is crucial to **mitigate risks**
  - Strategies such as using **separate compute environments**, **version control systems**, and **mimicking prod environment configurations** help achieve this separation effectively.
- IDEs **can enhance development productivity** with features like code navigation, active error catching, and version control integration

Git Fundamentals

Summary

- Git is **essential for managing code changes** and facilitating collaboration in software development
- Mastering Git fundamentals ensures efficient and effective version control; these include:
  - Git repositories, forks, branches, staging, commits, push, pull, and version releases

Making Source Code Modifications

Summary

- When making changes, always **refer to your design document**
  - Break objectives down into smaller tasks
  - Update as new insights are learned
- **Small iterative changes** help to reduce errors while developing
  - Test **early and often**!

Peer Review

Summary

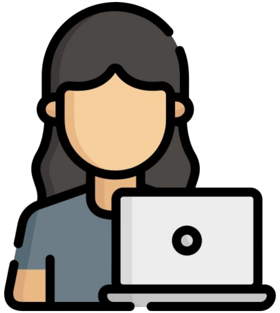
- Teamwork makes the dream work!
  - Dev teams help to **improve code quality** and promote reproducible, transparent, and interoperable software
- A Pull request (PR) is a **standard method to submit contributions** to a codebase
  - Standardizes the collaborative dev process

Software Development Practices

Developer's Journey

1. **Design Document**
   a. Clearly defining the problem and the proposed solution
2. **Development Environment**
   a. Separate from production
   b. Text editors and IDE's
3. **Making Source Code Modifications**
   a. Small interactive changes (version control)
4. **Peer Review**
   a. Collaborative development tec
5. **Bringing Changes into Production**
   a. Final testing
   b. Static version releases

Week 3 Focus

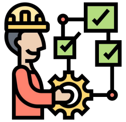


5. Bringing Changes into Production

Bringing Changes into Production

Final Testing

- In addition to testing at the PR level, final tests prior to a release provide more **comprehensive checks** across the repo
  - Especially since final modifications could be **made up of multiple PRs**
- Modifications being brought into production may need either **functional or validation testing**



Bringing Changes into Production

Final Testing

- **Functional Tests**: Verify that the software functions correctly in real-world scenarios

*Can automate these tests through **GitHub Actions***

GitHub Actions for Automated Testing

What are GitHub Actions?

- **Automation tool integrated into GitHub** that allows you to create custom workflows for your software development lifecycle
  - These workflows can **automate a variety of tasks** such as building, testing, and deploying code

GitHub Actions for Automated Testing

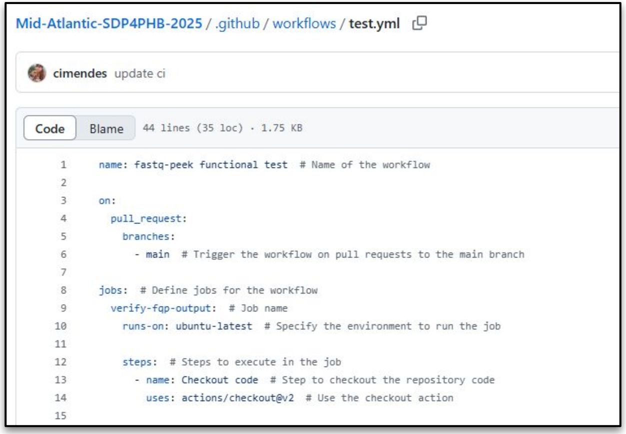
Utility for Testing

- GitHub Actions can **automatically run tests** on your code every time you push changes to your repository
  - Ensures that new code does not introduce errors or break existing functionality
  - Facilitates **Continuous Integration (CI)** by running tests **frequently and automatically**

GitHub Actions for Automated Testing

Setting Up GitHub Actions

- GitHub uses **YAML files to define workflows** for tasks like testing, building, and deploying code: `.github/workflows/{test}.yml`
  - Developers can configure a GitHub repository to look for these YAML files and **automatically launch these workflows**


Mid-Atlantic-SDP4PHB-2025 / .github / workflows / test.yml
cimendes update ci
Code
Blame
44 lines (35 loc) · 1.75 KB

```yaml
name: fastq-peek functional test  # Name of the workflow

on:
  pull_request:
    branches:
      - main  # Trigger the workflow on pull requests to the main branch

jobs:  # Define jobs for the workflow
  verify-fqp-output:  # Job name
    runs-on: ubuntu-latest  # Specify the environment to run the job

    steps:  # Steps to execute in the job
      - name: Checkout code  # Step to checkout the repository code
        uses: actions/checkout@v2  # Use the checkout action

```

GitHub Actions for Automated Testing

Key components of a workflow YAML:

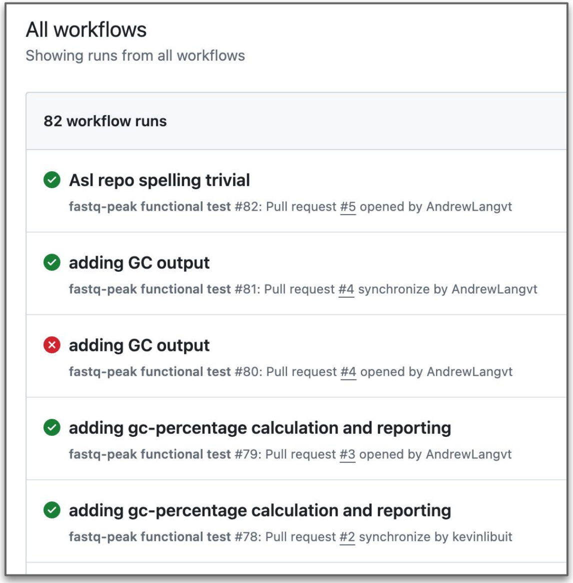
- **name** - name of the workflow
- **on** - triggering events, e.g. pull requests
- **jobs** - action of the workflow
    - **runs-on** - environment for the job to run, e.g. ubuntu-latest
    - **steps** - steps to execute the job itself, e.g. running an executable

Live walkthrough:
fastq-peek functional test

GitHub Actions for Automated Testing

Monitoring and Debugging

- Can assess the status of a workflow from the "**Actions**" tab of a repository



All workflows
Showing runs from all workflows

82 workflow runs

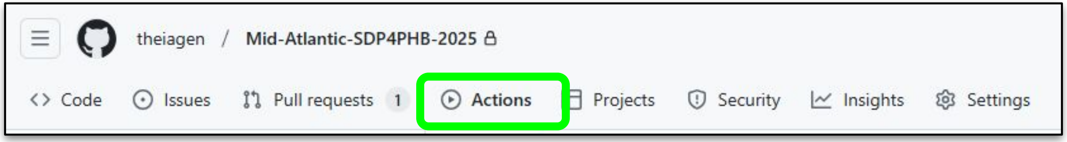
✓ **Asl repo spelling trivial**
fastq-peak functional test #82: Pull request #5 opened by AndrewLangvt

✓ **adding GC output**
fastq-peak functional test #81: Pull request #4 synchronize by AndrewLangvt

✗ **adding GC output**
fastq-peak functional test #80: Pull request #4 opened by AndrewLangvt

✓ **adding gc-percentage calculation and reporting**
fastq-peak functional test #79: Pull request #3 opened by AndrewLangvt

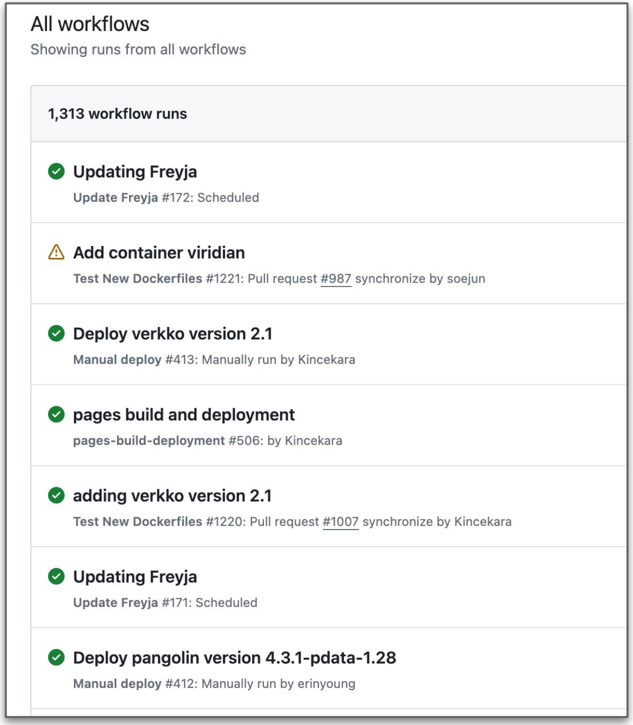
✓ **adding gc-percentage calculation and reporting**
fastq-peak functional test #78: Pull request #2 synchronize by kevinlibuit



Detailed logs available regarding each workflow launched

GitHub Actions for Automated Testing

Examples in the Field: StaPH-B Docker Builds



All workflows

Showing runs from all workflows

1,313 workflow runs

✓ **Updating Freyja**
Update Freyja #172: Scheduled

⚠ **Add container viridian**
Test New Dockerfiles #1221: Pull request #987 synchronize by soejun

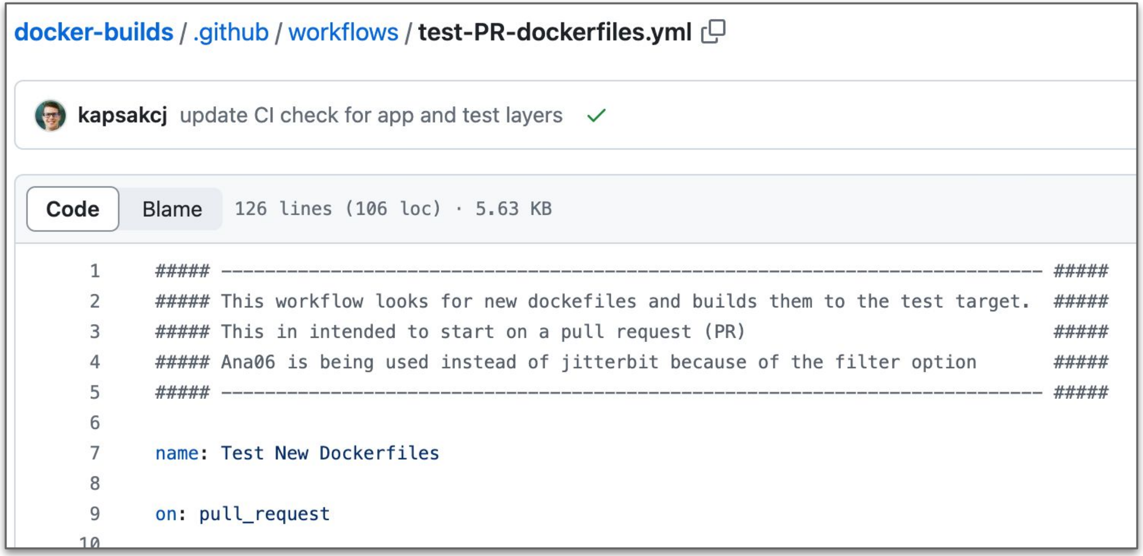
✓ **Deploy verkko version 2.1**
Manual deploy #413: Manually run by Kincekara

✓ **pages build and deployment**
pages-build-deployment #506: by Kincekara

✓ **adding verkko version 2.1**
Test New Dockerfiles #1220: Pull request #1007 synchronize by Kincekara

✓ **Updating Freyja**
Update Freyja #171: Scheduled

✓ **Deploy pangolin version 4.3.1-pdata-1.28**
Manual deploy #412: Manually run by erinyoung



docker-builds / .github / workflows / **test-PR-dockerfiles.yml**

kapsakcj update CI check for app and test layers ✓

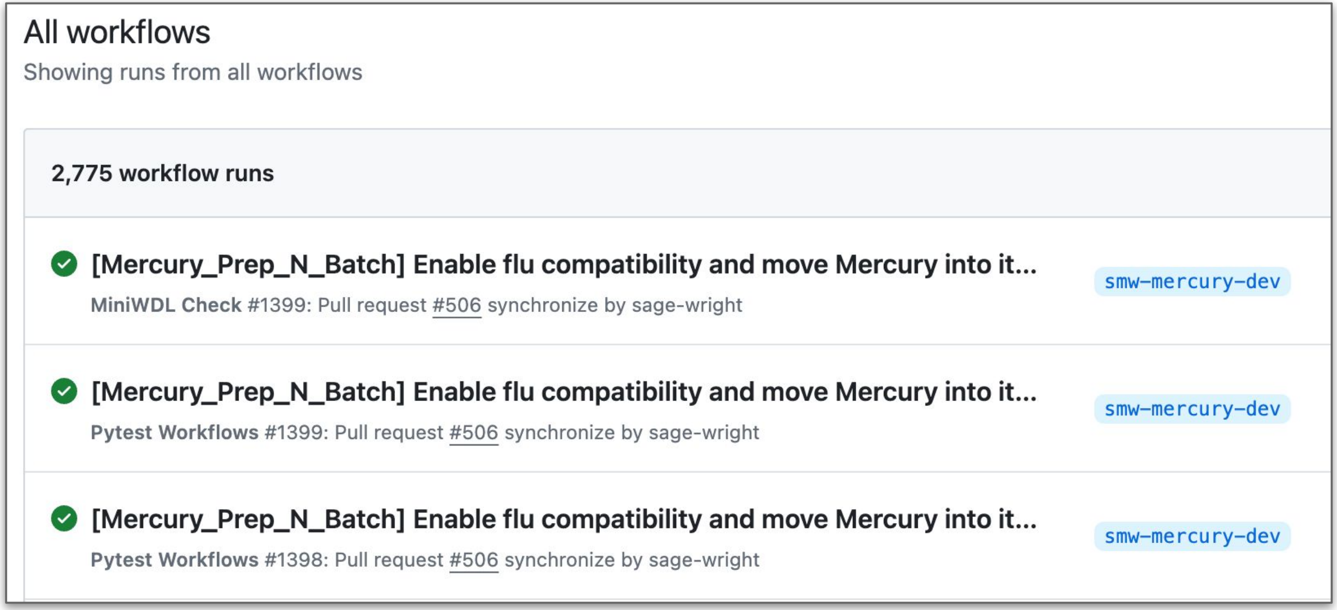
126 lines (106 loc) · 5.63 KB

```
##### ------------------------------------------------------------------------ #####
##### This workflow looks for new dockefiles and builds them to the test target. #####
##### This in intended to start on a pull request (PR)                          #####
##### Ana06 is being used instead of jitterbit because of the filter option     #####
##### ------------------------------------------------------------------------ #####

name: Test New Dockerfiles

on: pull_request
```

https://github.com/StaPH-B/docker-builds

GitHub Actions for Automated Testing

Examples in the Field: Theiagen's PHB



All workflows

Showing runs from all workflows

2,775 workflow runs

✓ **[Mercury_Prep_N_Batch] Enable flu compatibility and move Mercury into it...** `smw-mercury-dev`

MiniWDL Check #1399: Pull request #506 synchronize by sage-wright

✓ **[Mercury_Prep_N_Batch] Enable flu compatibility and move Mercury into it...** `smw-mercury-dev`

Pytest Workflows #1399: Pull request #506 synchronize by sage-wright

✓ **[Mercury_Prep_N_Batch] Enable flu compatibility and move Mercury into it...** `smw-mercury-dev`

Pytest Workflows #1398: Pull request #506 synchronize by sage-wright

https://github.com/theiagen/public_health_bioinformatics

Theiagen

GitHub Actions for Automated Testing

Summary

- GitHub Actions **can automate workflows for testing and deployment**, enhancing code quality with CI/CD integration directly within GitHub repositories
  - Can be specifically helpful for routine functional and validation tests

Bringing Changes into Production

Final Testing

- **Functional Tests**: Verify that the software functions correctly in real-world scenarios

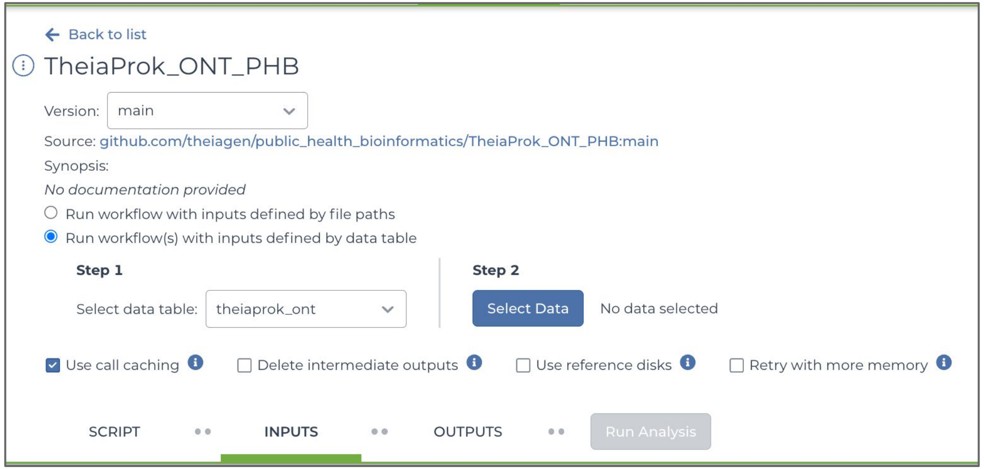
*Can automate these tests through **GitHub Actions***

- **Validation Tests**: Ensure the software meets the specified requirements
  - Performed against a benchmark or
  - predefined criteria

Bringing Changes into Production

Final Testing

- **User Acceptance Testing (UAT):** Have end-users test the software to ensure it meets their needs and expectations



← Back to list

TheiaProk_ONT_PHB

Version: main

Source: github.com/theiagen/public_health_bioinformatics/TheiaProk_ONT_PHB:main

Synopsis:

No documentation provided

○ Run workflow with inputs defined by file paths

● Run workflow(s) with inputs defined by data table

Step 1

Select data table: theiaprok_ont

Step 2

Select Data No data selected

☑ Use call caching □ Delete intermediate outputs □ Use reference disks □ Retry with more memory

*For Terra users, you can have **end-users test your main branch** in a workspace prior to a release*

Bringing Changes into Production

Static Releases

- Deploying a fixed version of the software
  - Provides clear versioning history
  - Facilitates dependency management and updates

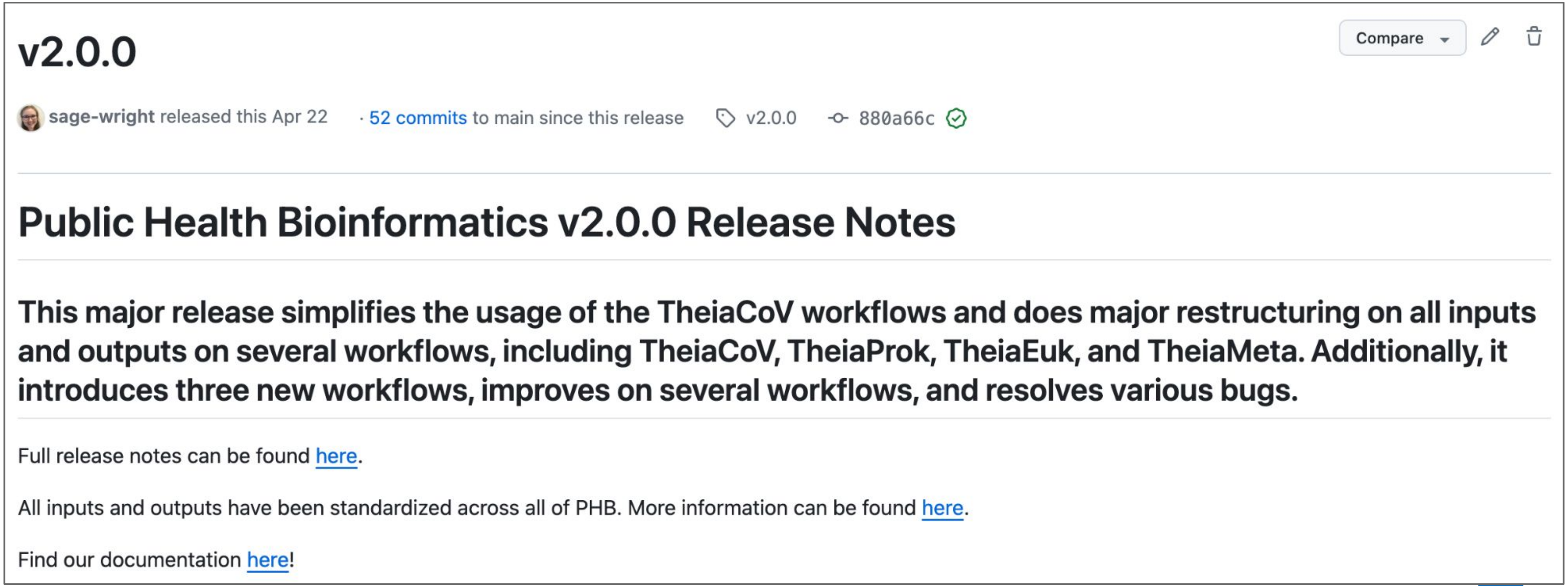
Semantic Versioning

- Versioning scheme that reflects changes

v1.3.0

MAJOR MINOR PATCH

Example Releases: PHB v2.0.0



v2.0.0

sage-wright released this Apr 22 · 52 commits to main since this release v2.0.0 880a66c

Public Health Bioinformatics v2.0.0 Release Notes

This major release simplifies the usage of the TheiaCoV workflows and does major restructuring on all inputs and outputs on several workflows, including TheiaCoV, TheiaProk, TheiaEuk, and TheiaMeta. Additionally, it introduces three new workflows, improves on several workflows, and resolves various bugs.

Full release notes can be found here.

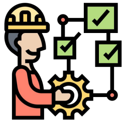
All inputs and outputs have been standardized across all of PHB. More information can be found here.

Find our documentation here!

Bringing Changes into Production

Summary

- **Final testing** should be performed prior to bringing in changes into production
    - **Comprehensive tests**, especially important when modification is made up of multiple small changes
- **Static releases** help provide clear versioning
    - **Semantic versioning** helps to reflect changes from version to version

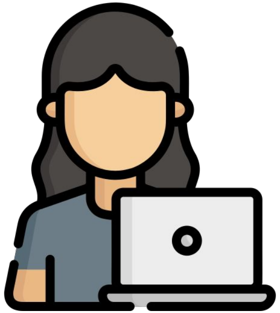


Hands-On Exercise

Exercise 03: GitHub Actions & Static Releases

Exercise Goal

1. **Use GitHub and your dev environment to:**
    a. Troubleshoot a failed GitHub action
    b. Modify the codebase to resolve the failed action



Theiagen®

Final Thoughts

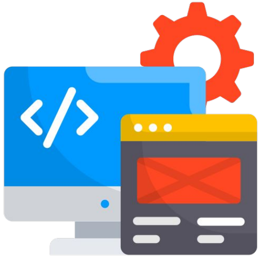
Allowing time for trainees to complete Exercise 3.

Software Development Practices

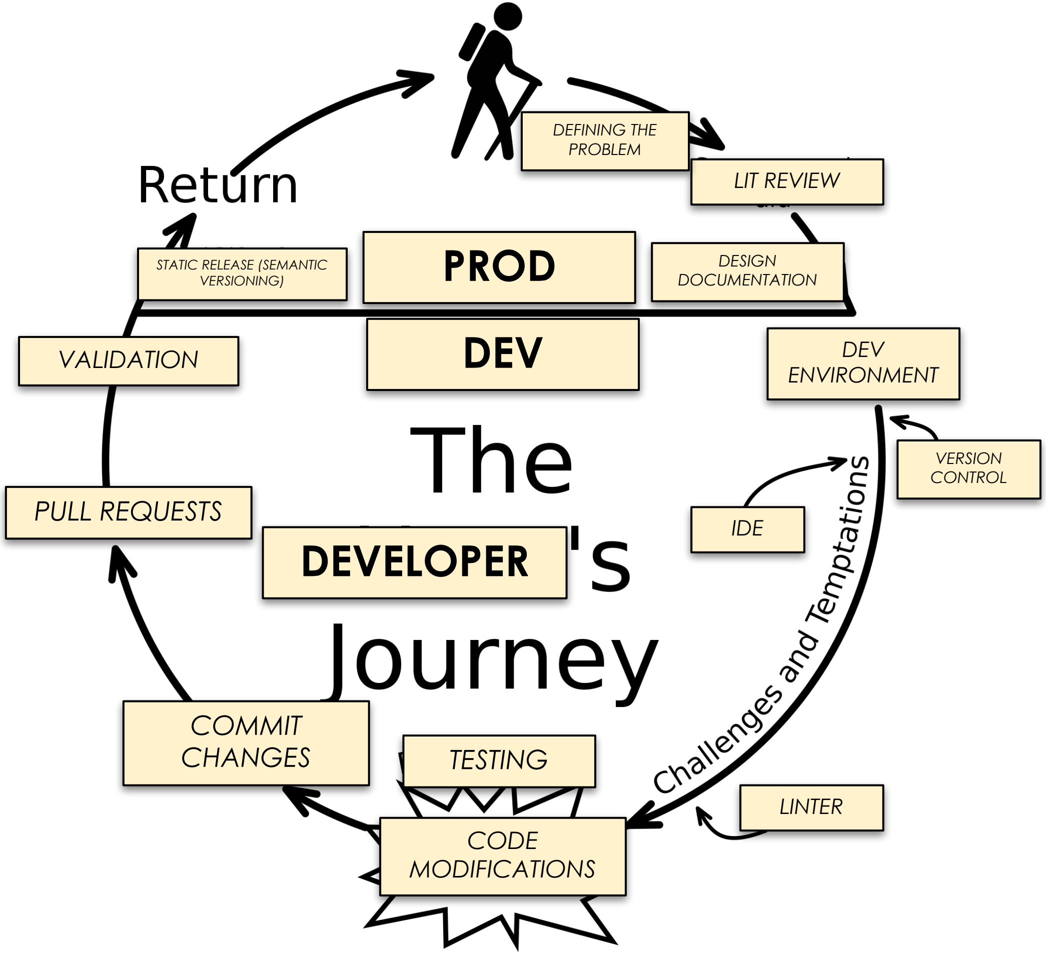
Developer’s Journey

1. **Design Document**
   a. Clearly defining the problem and the proposed solution
2. **Development Environment**
   a. Separate from production
   b. Text editors and IDE’s
3. **Making Source Code Modifications**
   a. Small interactive changes (version control)
4. **Peer Review**
   a. Standards for review, testing, etc.
5. **Bringing Changes into Production**
   a. Merging to main
   b. Static version releases

DOCUMENT
ALL THE THINGS!



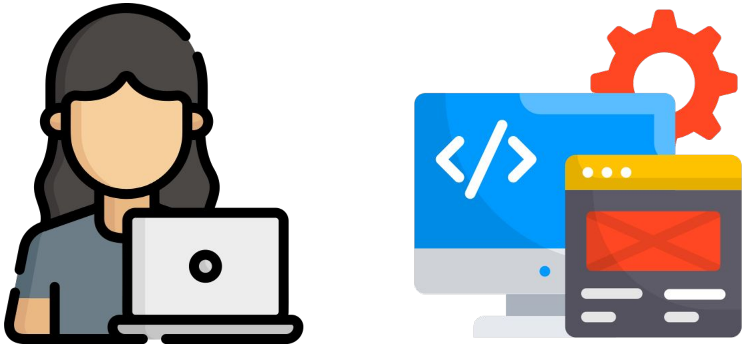
Theiagen®



Byrne, C. (2017). The Hero's Journey. Wikipedia https://en.wikipedia.org/wiki/File:Heroesjourney.svg

The Developer’s Journey

Framework where a protagonist **enters into their dev environment**, faces challenges, gains new wisdom, and **brings changes into production**.



Theiagen®