Welcome to this Training Session with Theiagen Genomics

We will soon be getting started

Theiagen®

Software Development Practices for Public Health Bioinformatics

Week 02: Git Fundamentals and Making Source Code Modifications

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

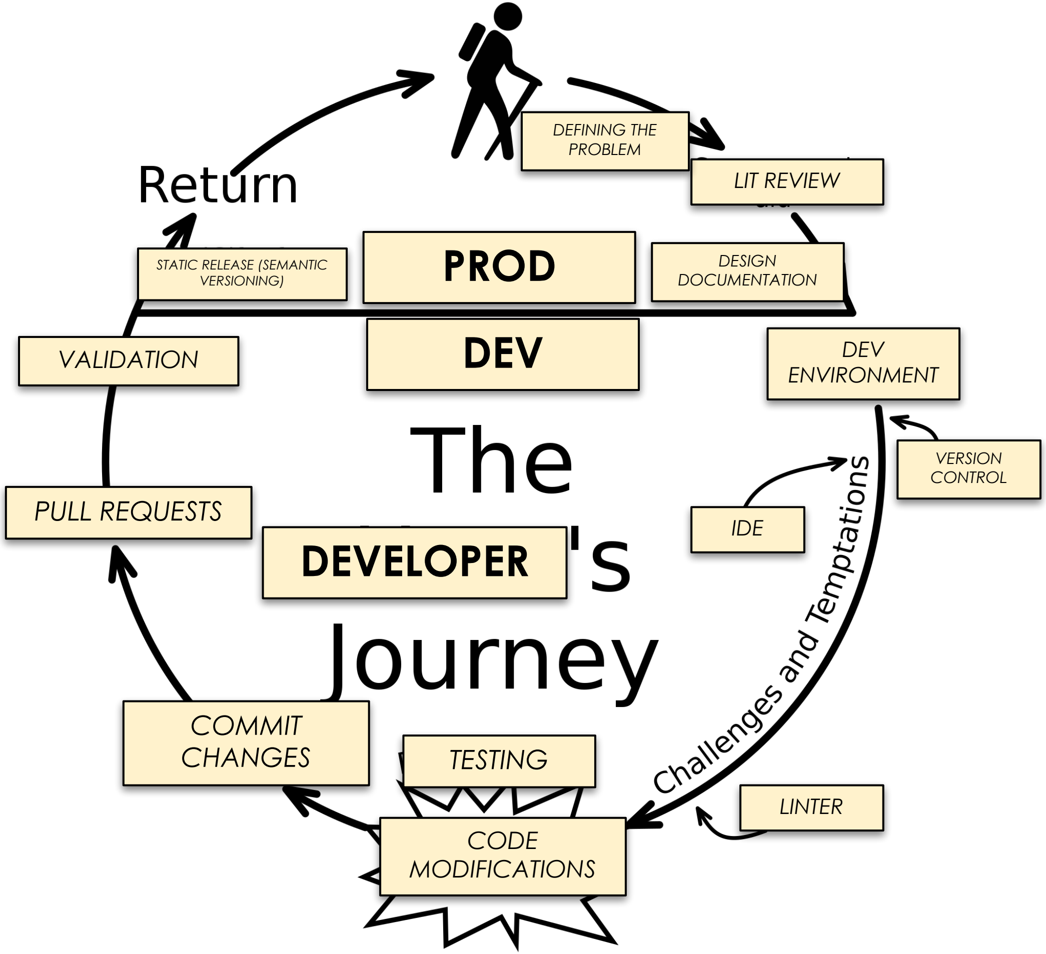
Course Introduction

Training Workshop Instructors

Sage Wright, MSc

- Senior Bioinformatics Developer at Theiagen Genomics since 2022
- MSc in Bioinformatics and Genomics
- BSc in Bioinformatics

Week 1 Recap



DEFINING THE PROBLEM

LIT REVIEW

DESIGN DOCUMENTATION

PROD

DEV

STATIC RELEASE (SEMANTIC VERSIONING)

Return

VALIDATION

PULL REQUESTS

The DEVELOPER's Journey

DEV ENVIRONMENT

VERSION CONTROL

IDE

Challenges and Temptations

COMMIT CHANGES

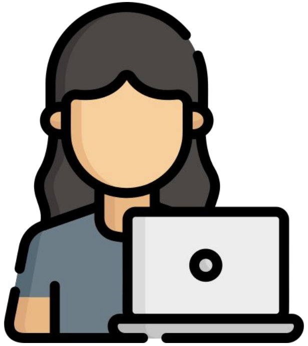
TESTING

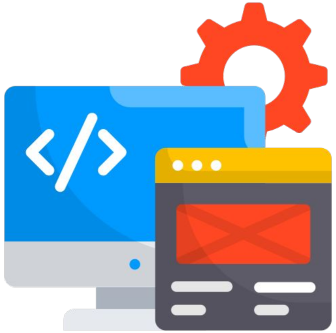
CODE MODIFICATIONS

LINTER

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

Design Document

Summary

- The design document is a vital tool that **defines the problem and the proposed 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

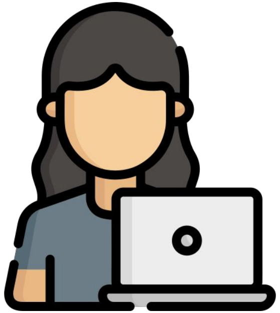
Software Development Practices

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

Week 2 Focus

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



Theiagen®

A Note on Version Control Systems

Version Control Systems (VCS)

- Essential development tools that help manage changes to source code over time
  - Track (and save) modifications to the code

Git and GitHub

- Git is a VCS software for managing code in repositories
- GitHub is a platform to host Git repositories

*Git repositories can be hosted on other platforms such as **BitBucket and GitLab***

*More on **Git ~~next week~~ TODAY**!*

Git Fundamentals

Theiagen®

Git Fundamentals

Distributed Version Control System

- Essential development tool for **organizing and tracking code changes** efficiently
- Can distribute a **full copy** of the project repository, including its history, to every developer
  - Enables working offline and **independently from a central server**
  - Allows for multiple development tracks to exist simultaneously

Git Fundamentals

Understanding Git Repositories

- Structured system that manages all project files and their version history
    - **Local Git** repositories are hosted on **your machine**
    - **Remote Git** repositories are hosted **online**
        - Often hosted on platforms such as GitHub

*Developers typically **clone** (create a local copy of) a remote repository to their local machine and **sync changes periodically***

Git Fundamentals

Commits

- A commit is a **snapshot of the repository** at a specific point in time
    - Records changes in the repository, allowing for a detailed history of the project
    - Facilitates tracking, reverting, and understanding changes over time

*A **commit history** is a chronological record of all commits made in a repository*

Git Fundamentals

Stagging

- Staging is the process of **selecting specific changes** to include in the next commit
  - Allows you to review and organize changes before committing them
  - Provides a way to manage and separate changes into logical units, ensuring only the desired changes are recorded

Git Fundamentals

Relationship Between Commits and Staging

- **Stage Changes**: First, you stage changes that you want to include in the next commit using
- **Create a Commit**: Once changes are staged, you create a commit

*Staging allows for **precise control** over what changes are included in each commit, making it easier to organize and manage changes*

Git Fundamentals

Repository Branches

- Branches are **divergent lines of development** within a repository
- Commonly used for developing new features, fixing bugs, or experimenting with new ideas.

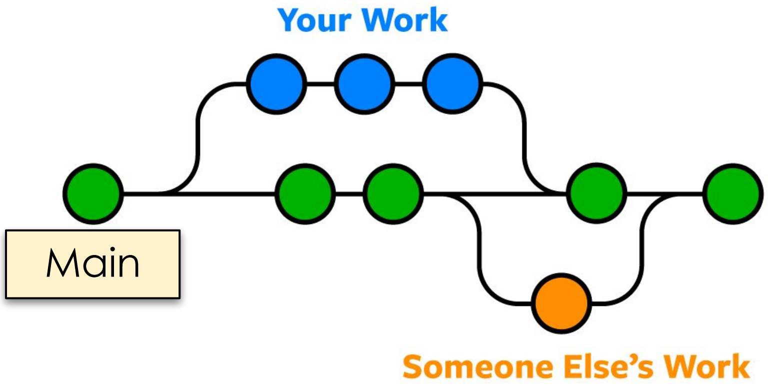
*Allows developers to work on **different tasks simultaneously** without interference*

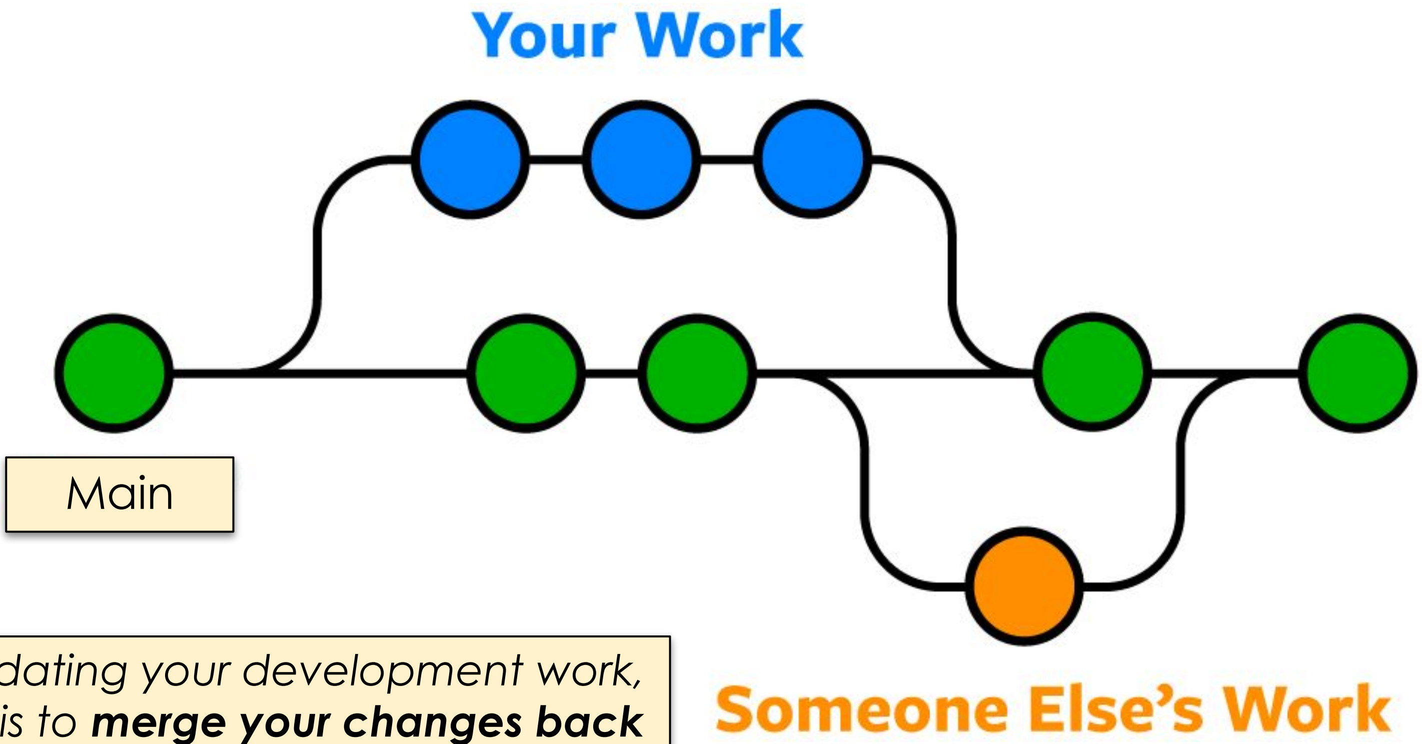
*Ideal to be working on a **development branch** within your **dev environment***

Git Fundamentals

Repository Branches

- Main (or *Master*) Branch is the **primary branch** where the stable code is maintained
  - Best practice to have this branch serve as a **production-ready version** of the repository



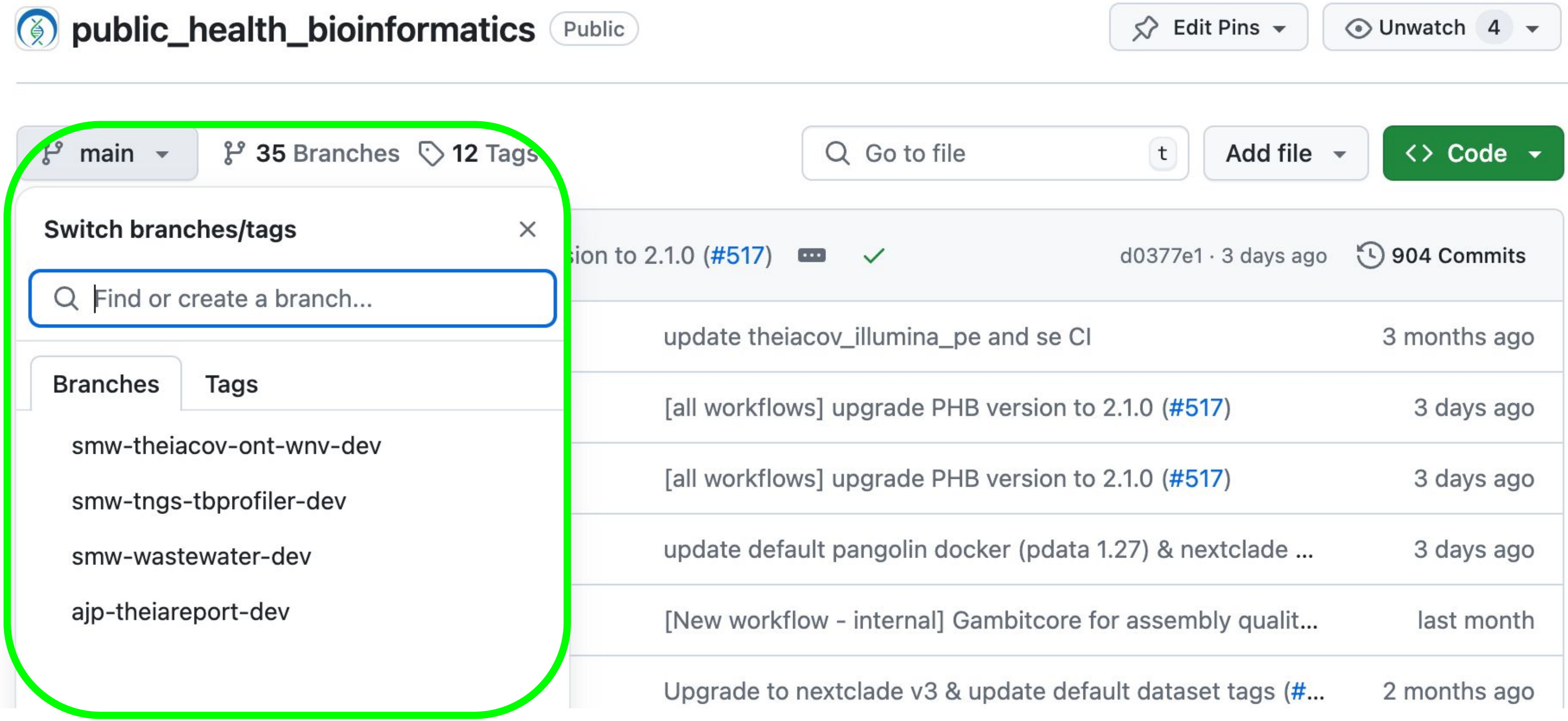


*After validating your development work, the goal is to **merge your changes back into the main (or master) branch***

Git Fundamentals

Branch Management

- Git allows developers to switch between multiple branches within a Git repository
  - **Can get complex quickly**!
  - Good practice to establish a **naming-scheme** for development branches,
    - e.g. *{initials}-{description}-dev*



public_health_bioinformatics Public

Edit Pins

Unwatch 4

main

35 Branches

12 Tags

Go to file t

Add file

Code

Switch branches/tags

Find or create a branch...

Branches

Tags

smw-theiacov-ont-wnv-dev

smw-tngs-tbprofiler-dev

smw-wastewater-dev

ajp-theiareport-dev

ion to 2.1.0 (#517)

d0377e1 · 3 days ago

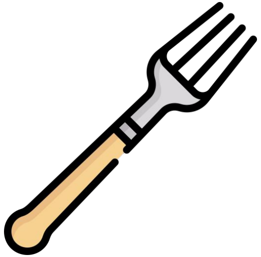
904 Commits

update theiacov_illumina_pe and se CI	3 months ago
[all workflows] upgrade PHB version to 2.1.0 (#517)	3 days ago
[all workflows] upgrade PHB version to 2.1.0 (#517)	3 days ago
update default pangolin docker (pdata 1.27) & nextclade ...	3 days ago
[New workflow - internal] Gambitcore for assembly qualit...	last month
Upgrade to nextclade v3 & update default dataset tags (#...	2 months ago

Git Fundamentals

Repository Forks

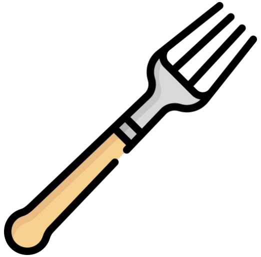
- A fork is a **copy of a repository**, including branches
    - Created **independently** from the original repository
- Allows developers to freely experiment with changes **without affecting the original project**



Git Fundamentals

Repository Forks

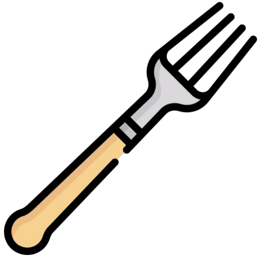
- Forks are often used to **contribute to someone else's project**
    - Developers can make changes in a forked repository and then submit those changes back to the original repository through a pull request



Git Fundamentals

Branches vs Forks

- Branch is part of the **same repository** and shares its history and structure
- A forked repository is **completely separate** from the original repository
  - Hosted under a **separate account or organization**



Git Fundamentals

Pull Requests

- A pull request (PR) is a method of **submitting contributions to a project**
  - Allows code review and discussion before integrating changes
- Developer's can make pull-requests across branches or forks
  - PRs get *merged* into their target

Git Fundamentals

Static Releases

- Stable copy of your codebase; created at a specific point in the project's development cycle marked as a **stable, production-ready snapshot**
  - Usually tagged with **semantic versioning**

Git Fundamentals

Semantic Versioning

- A versioning system that uses a three-part number format (e.g., 1.3.0) to indicate the type of changes in the release: Major, Minor, and Patch

v1.3.0

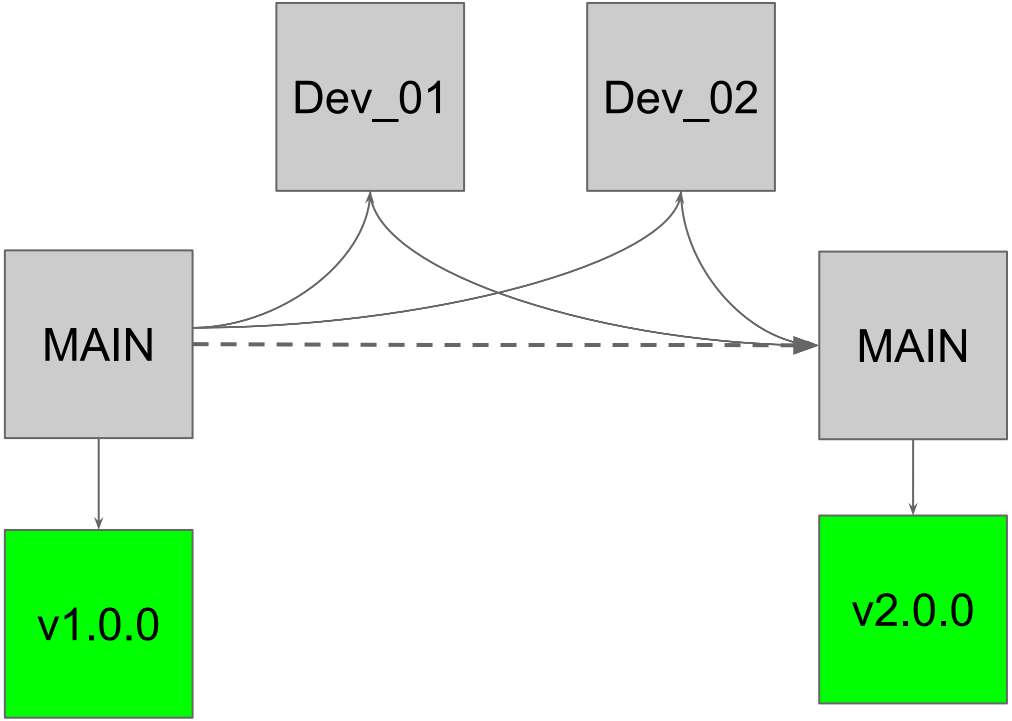
MAJOR MINOR PATCH

Git Fundamentals

Semantic Versioning

- **Major Version**: Indicates significant changes that may break backward compatibility
- **Minor Version**: Adds new features without breaking backward compatibility
- **Patch Version**: Includes bug fixes and minor improvements that do not affect compatibility

Git for Software Development



*After **approving the changes of a dev branch**, it gets merged it into the main branch*

*Releases are made at different snapshots of the **Main branch***

Git Fundamentals

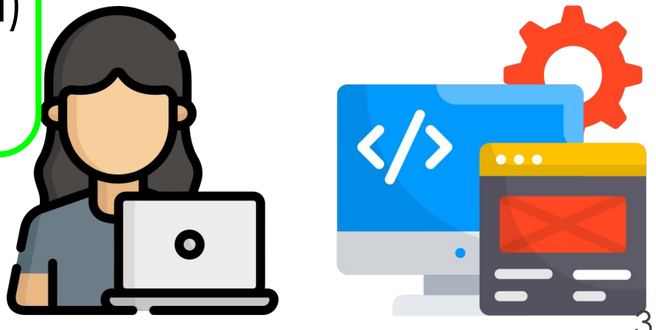
Summary

- 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

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



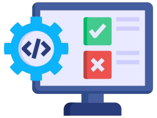
3. Making Source Code Modifications

Theiagen®

Making Source Code Modifications

Refer to Your Design Document

- Follow the plan **outlined in the design document***, ensuring modifications align with the overall project objectives
- Regularly review the design document to stay on track and **make adjustments** based on new insights



*****Living document*** *that serves as a* ***reference*** *throughout the development process*

Theiagen®

Making Source Code Modifications

Small Iterative Changes

- Break down development objectives into smaller, manageable tasks
  - **Commit changes frequently** to version control

Testing

- Conduct **testing for each small change** to catch issues early
- Automated tests can assist with continuous integration and validation

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**!

4. Peer Review

Theiagen®

Peer Review

Collaborative Dev Teams

- Improve code quality through **collective knowledge and diverse perspectives**
  - Can collaborate across institutions
    - StaPH-B Docker Builds has **over 70 contributors** from institutions across the world!
- Enables regular code reviews, pair programming, and **promotes use of best practices**

*Faster alone, **further together***

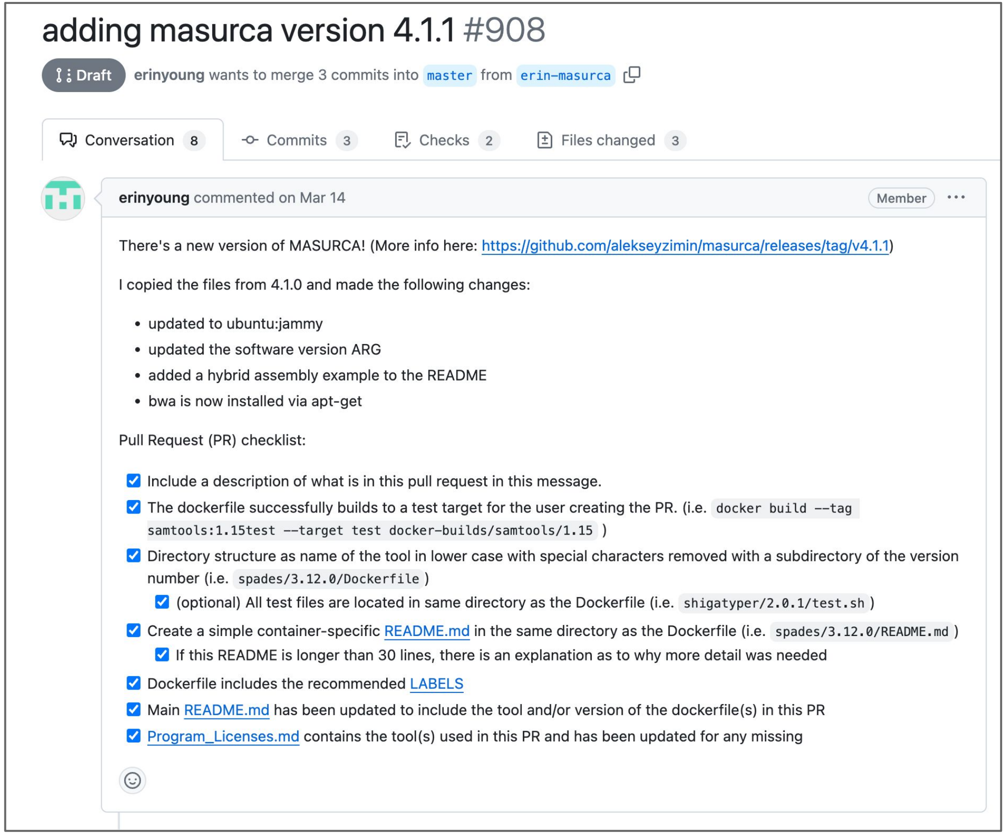
Theiagen

Peer Review

Pull Requests (PRs)

- Method of **submitting contributions to a codebase** in a version control system
  - Facilitates code review, ensuring changes are vetted and **approved by a peer before integration**
- Can create **PR templates** to standardize review
  - Ensures all necessary information is provided for each pull request
    - Should include testing information for reviewer

PR Examples in the FIeld: Docker Builds



adding masurca version 4.1.1 #908

erinyoung wants to merge 3 commits into `master` from `erin-masurca`

erinyoung commented on Mar 14

There's a new version of MASURCA! (More info here: https://github.com/alekseyzimin/masurca/releases/tag/v4.1.1)

I copied the files from 4.1.0 and made the following changes:

- updated to ubuntu:jammy
- updated the software version ARG
- added a hybrid assembly example to the README
- bwa is now installed via apt-get

Pull Request (PR) checklist:

- ☑ Include a description of what is in this pull request in this message.
- ☑ The dockerfile successfully builds to a test target for the user creating the PR. (i.e. `docker build --tag samtools:1.15test --target test docker-builds/samtools/1.15`)
- ☑ Directory structure as name of the tool in lower case with special characters removed with a subdirectory of the version number (i.e. `spades/3.12.0/Dockerfile`)
  - ☑ (optional) All test files are located in same directory as the Dockerfile (i.e. `shigatyper/2.0.1/test.sh`)
- ☑ Create a simple container-specific README.md in the same directory as the Dockerfile (i.e. `spades/3.12.0/README.md`)
  - ☑ If this README is longer than 30 lines, there is an explanation as to why more detail was needed
- ☑ Dockerfile includes the recommended LABELS
- ☑ Main README.md has been updated to include the tool and/or version of the dockerfile(s) in this PR
- ☑ Program_Licenses.md contains the tool(s) used in this PR and has been updated for any missing

Collaborative Development in Practice

- Use of PR template to ensure all tasks completed for contribution to be merged
- Includes conversation regarding potential issues with code change

*For more examples, check out **closed PRs** in the same repo!*

Peer Review

Summary

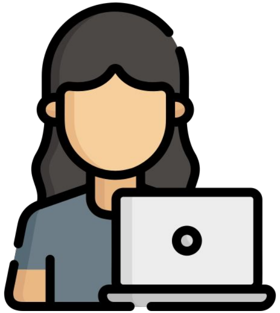
- Teamwork makes the dream work!
  - Dev teams help to **improve code quality** and promote reproducible, transparent, and interoperable software
- A Pull request (PR) is a **standard method to submit contributions** to a codebase
  - Standardizes the collaborative dev process

Hands-On Exercise

Exercise 02: Version Control with Git

Exercise Goal

1. **Use Git & GitHub to:**
   a. Create a development branch
   b. Stage and commit changes to a dev branch
   c. Issue a pull request



Theiagen®

Exercise 02: Version Control with Git

Exercise Goal

1. Review a design document for a development initiative
2. Access a development environment via GitPod
3. Use VSCode IDE to test code and script solution