



**THE WORLD BANK**

IBRD • IDA | WORLD BANK GROUP

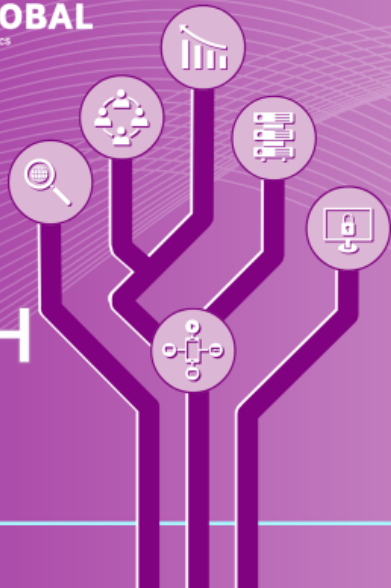
Development Economics • Impact



**DEC GLOBAL**

DEVELOPMENT ECONOMICS  
GLOBAL INSTITUTE

# MANAGE SUCCESSFUL FIELD RESEARCH 2024



# GitHub - Workflow training

---

Manage Successful Field Research

May 7, 2024

Development Impact Evaluation (DIME)

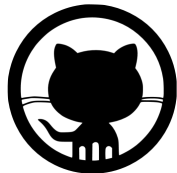
The World Bank

# Prerequisites

**Who is this training for?** We will build upon the content from the previous session. By now, you should be familiar with the concepts and actions of branching, committing, and merging. This session introduces a workflow to enhance project collaboration and code quality.

## Resources:

- Presentation examples: <https://github.com/dime-wb-trainings/pr-training-dec21>
- Repo for Hands-on Exercises: <https://github.com/dime-wb-trainings/lyrics-msfr24>



## THIS TRAINING HAS THREE PARTS

### Part 1

A brief recap of using branches in Git

### Part 2

Introducing the *branch-PR-merge* cycle for how to use the review features in a PR

### Part 3

Gitflow - a philosophy for how to work with branches and merge/PRs in Git

# The branch-PR-merge cycle

The screenshot shows a GitHub repository interface. At the top, there are navigation buttons: 'main' (selected), '2 branches', and '0 tags'. To the right are 'Go to file', 'Add file', and 'Code' buttons. Below this is a commit history table with columns for the commit author, message, hash, time, and commit count. The table shows three commits: 'ABBA is the best! And this song is their best!' by kbjarkefur, 'add a ignore file' by lyrics, and 'Update README.md' by README.md. Below the table is a file browser section showing 'README.md' with an edit icon. The main content area displays the README text, which includes a title 'DIME Pull request training', a description of the repository's purpose, and contact information.

Commit	Message	Hash	Time	Commits
kbjarkefur	ABBA is the best! And this song is their best!	08f5d5d	now	7
lyrics	add a ignore file		now	
README.md	Update README.md		10 seconds ago	

**DIME Pull request training**

This repository was created for a training session of [DIME Analytics Pull request training](#). You can read more about DIME Analytics Git and GitHub trainings [here](#).

**Contact**

If you have any questions about this training, please email us at [dimeanalytics@worldbank.org](mailto:dimeanalytics@worldbank.org)

All screenshots in this slide comes from this repo:  
<https://github.com/dime-wb-trainings/pr-training-dec21>

# **Part 1:**

## **A brief recap to branches**

---

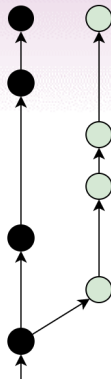
# Brief intro to branches

Branches are the killer feature of Git. It allows us to approach tasks in a non-linear way, which is similar to how human brains and especially teams of humans work.

Thanks to branches we can (without creating conflicts):

- Test multiple solutions to a task at the same time
- Have multiple people working at different tasks at the same time

Branches are often visualized in network graphs.

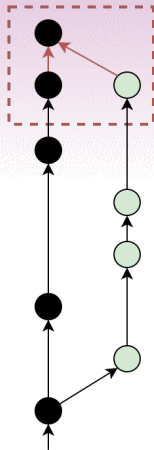


# What are pull requests?

*Pull requests* (PRs) are part of GitHub, but not part of Git.

GitHub is to Git, what Gmail is to email.

A PR is a request to merge one branch into another, and adds great QA and review features before the merge.





## **Part 2:**

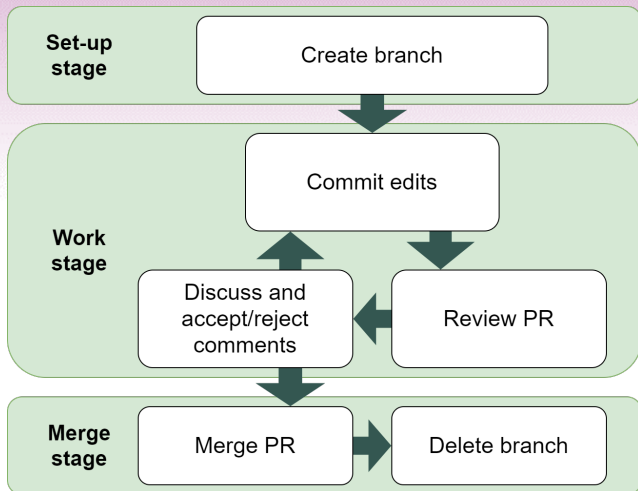
# **The branch-PR-merge cycle**

---

# The branch-PR-merge cycle

**Every** data work task in your project should have a *branch-PR-merge* cycle.

The *branch-PR-merge* cycle consists of three stages, we will focus on the *work stage* today.

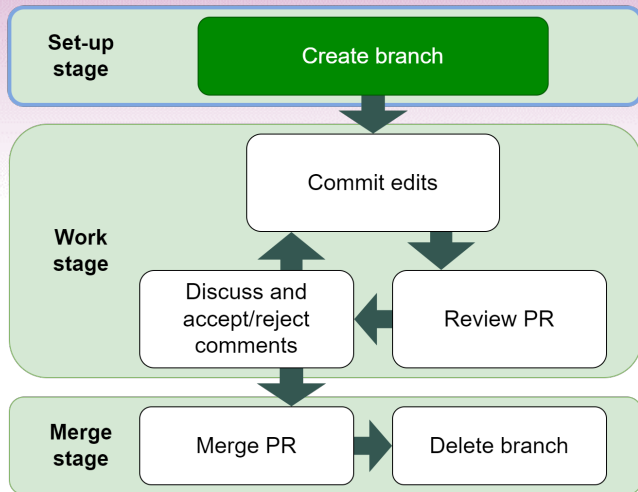


# Set-up Stage

## SET-UP STAGE:

Create a branch for this task!

In Part 3 we will see when and how many branches to create.

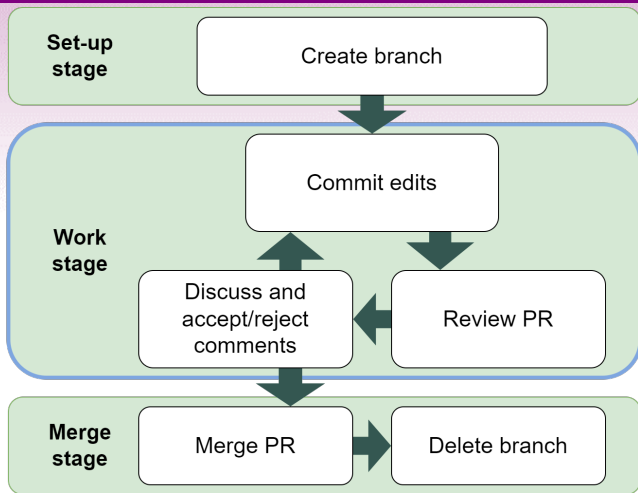


## WORK STAGE:

Most time in a branch-PR-merge cycle is spent in the work stage.

Research teams typically under-utilize features in the *Review* and *Discuss* steps. This is the real added value of PRs.

The work stage should be repeated as many times as needed.

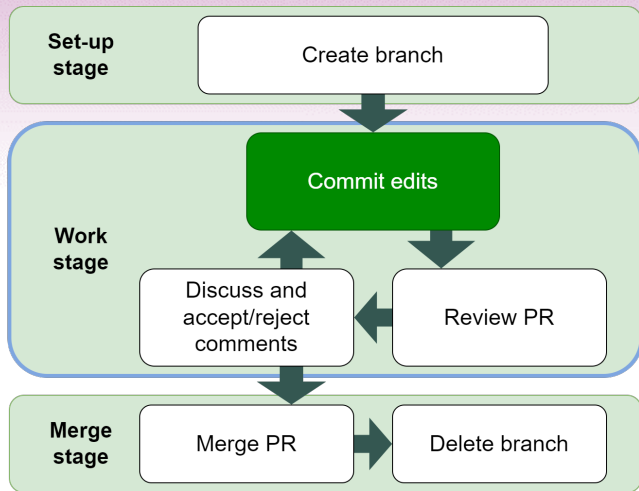


## Work Stage: Commit edits

We will call the person making the commits *Author* and the person reviewing the PR *Reviewer*. This can be two people, groups, or one person reviewing their own work.

In the first step Author working on a task commits edits as usual to the new branch.

The PR must be opened before proceeding to the *Review PR* step.



# Create the PR

## Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).



base: main



compare: improve-lyrics

✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others.  
[Learn about pull requests](#)

Create pull request

4 commits

2 files changed

1 contributor



Commits on Dec 7, 2021

I like April rains better

kbjarkefur committed 2 minutes ago



d47ce03



One chorus in the end is enough

kbjarkefur committed 2 minutes ago



3fc27f4



I like April rains better

kbjarkefur committed 2 minutes ago



126732f



Axel Rose singing Eros Ramazzotti

kbjarkefur committed 2 minutes ago



4ac8b7a



- Typically done by the Author.
- Select the two branches you want to merge:
  - **Base branch:** This is the branch into which changes will be merged.
  - **Compare branch:** This is the branch that contains the new changes or features.
- Check the tabs in the *Commits*, *Files changed*, etc., bar to make sure you are creating a PR for the correct branches. Remember, merges can be difficult to revert.

# Document the PR

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: main ← compare: improve-lyrics ✓ Able to merge. These branches can be automatically merged.

Making improvements to these lyrics

Write Preview

H B I

In meeting on June 12th, 2021 the DIME Lyrics team decided that I will improve these lyrics.

We decided to mostly focus on November Rain and Backstreet Back.

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

Remember, contributions to this repository

- ✓ Create pull request  
Open a pull request that is ready for review
- Create draft pull request  
Cannot be merged until marked ready for review

Reviewers  
No reviews

Assignees  
No one—assign yourself

Labels  
None yet

Projects  
None yet

Milestone  
No milestone

Linked issues  
Use [Closing keywords](#) in the description to automatically close issues

Helpful resources  
[GitHub Community Guidelines](#)

- Merged PRs are milestones that can be browsed in the future - document these milestones.
- Why did the Author work on this task?
- Making people reviewers and assignees as well as using labels and projects are neat features, but not necessary.

See this PR at

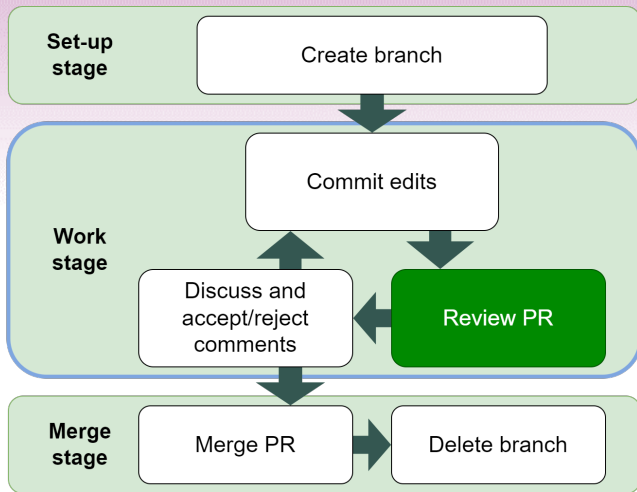
<https://github.com/dime-wb-trainings/pr-training-dec21/pull/1>

## Work Stage : Review PR

Review a PR when either:

- the task is completed.
- the Author wants help.
- after a set amount of time.

The PR can be opened as a *Draft PR* already after one commit.





# Overall comments

Conversation 0 Commits 4 Checks 0 Files changed 2

**kbjarkefur** commented 2 minutes ago Member 😊 ...

In meeting on June 12th, 2021 the DIME Lyrics team decided that I will improve these lyrics.

We decided to mostly focus on November Rain and Backstreet Back.

**kbjarkefur** added 4 commits 8 minutes ago

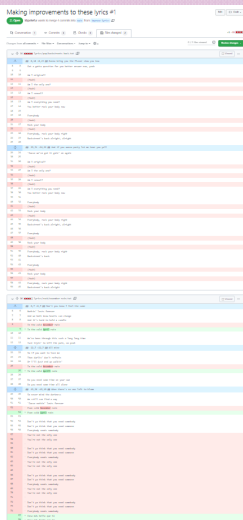
- 🔗 I like April rains better d47ce03
- 🔗 One chorus in the end is enough 3fc27f4
- 🔗 I like April rains better 126732f
- 🔗 Axel Rose singing Eros Ramazzotti 4ac8b7a

**kbjarkefur** commented now Member Author 😊 ...

Thank you [@kbjarkefur](#)! I ran this code. I was able to reproduce your work. However, it would be good if you can improve the documentation, so it is easier to find the master file that runs all other files

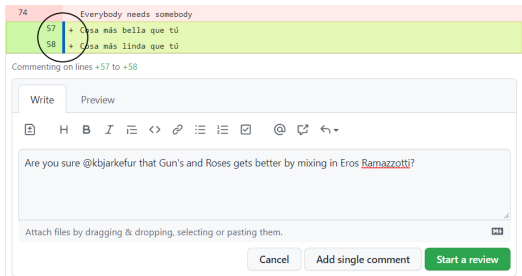
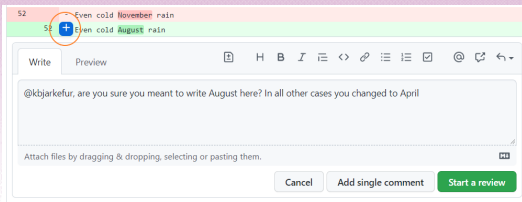
- If possible, always start by running the code.
- Provide overall comments in the “*Conversation*” tab. Code does or does not run, not possible to follow the code etc.
- Tag the Author or Reviewer your comment is meant for.

# Review the PR



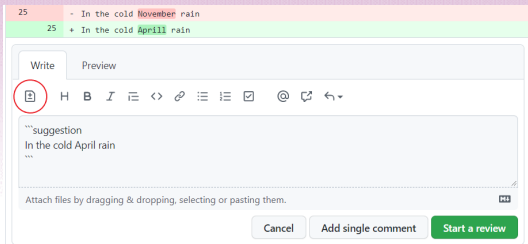
- Next, go to the *“Files changed”* tab - this is the summary of all changes in all commits in this PR.
- In this tab we make comments to specific lines of code which is always much more useful than overall comments.

# Line comments

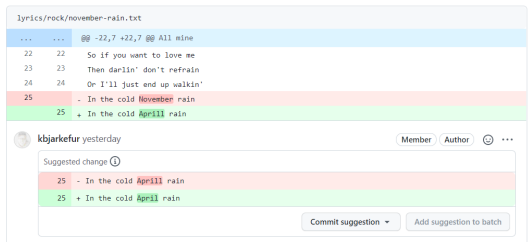


- You can start a thread anywhere in the code in this tab.
- *Comment one line*: Hold with the mouse over the line number for a line you want to comment on and click the plus sign - orange circle.
- *Comment multiple lines*: Press shift while click-and-drag mouse over multiple lines - black circle.

# Line suggestions



The screenshot shows a GitHub comment editor. At the top, there's a diff view for line 25: a red line with a minus sign and the text "In the cold November rain", and a green line with a plus sign and the text "In the cold April rain". Below this is a "Write" tab and a "Preview" tab. In the "Write" tab, there's a toolbar with icons for bold, italic, link, code, etc. A red circle highlights the "suggest code" icon (a document with a plus sign). Below the toolbar is a text area containing the text "suggestion" followed by "In the cold April rain". At the bottom, there are three buttons: "Cancel", "Add single comment", and "Start a review".

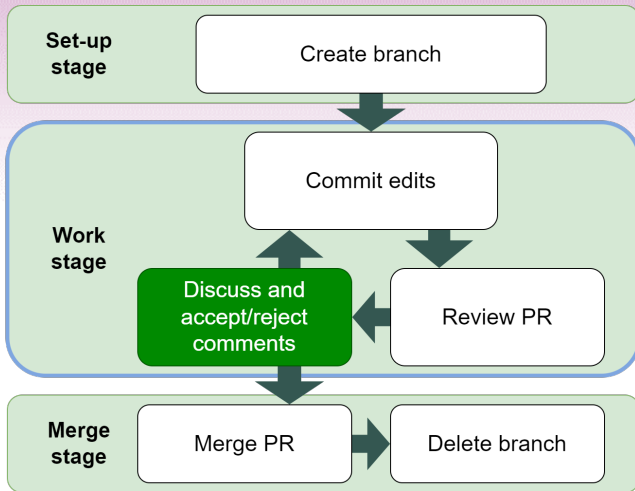


The screenshot shows a GitHub pull request interface. At the top, it says "lyrics/rock/november-rain.txt". Below that is a diff view for line 25: a red line with a minus sign and the text "In the cold November rain", and a green line with a plus sign and the text "In the cold April rain". Below the diff view is a section for a user named "kbjarkefur" who is the author. It says "Suggested change" with an information icon. Below this is a preview of the suggested change, showing the same diff for line 25. At the bottom, there are two buttons: "Commit suggestion" and "Add suggestion to batch".

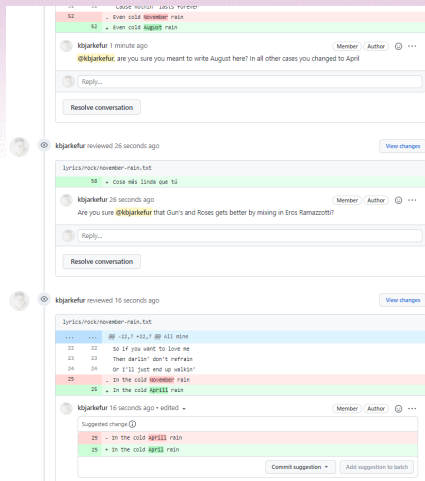
- You can suggest edits to one or multiple lines of code directly in this comment field.
- Click the suggest code button - red circle - and edit the code that appears.
- The Author will then see the exact suggestion, and can commit it with the click of a button.
- Works really well when tweaking the code in the PR, but should only be used for smaller edits.

## Discuss and accept/reject

- The Author and the Reviewer(s) discuss, accept or reject suggestions and reply to comments.
- The Author, the Reviewer(s) and/or anyone else on the project team then decide if the work stage needs to be repeated.



# Conversation summary



- Back in the “*Conversation*” tab, all comments and suggestions are summarized.
- Reply to comments or resolve them by clicking the “*Resolve conversation*” button.
- Resolve suggestions by accepting them with the “*Commit Suggestion*” button or ignore the suggestion by clicking “*Resolve conversation*”.
- Keep reviewing the PR until all these comments are resolved.
- If possible, always test run the code a final time.



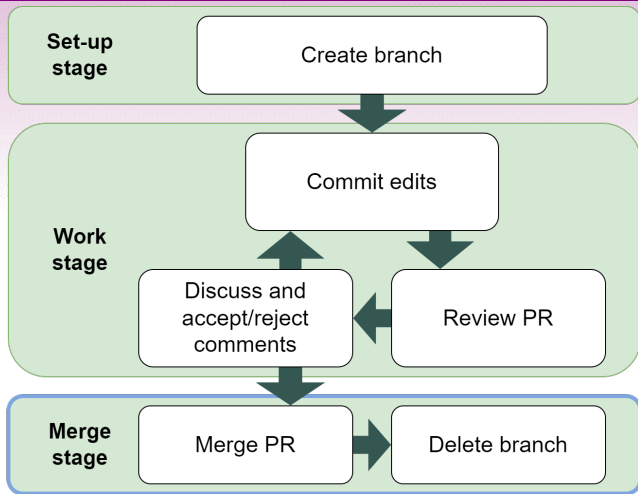
- After responding to all suggestions and comments, the team should discuss whether the task is completed or requires further work.
- Repeat the work stage if significant work is still required.
- Move on to the merge stage only when the task is complete and the team agrees the code is high-quality.

# Merge Stage

## MERGE STAGE:

The last stage of the *branch-PR-merge* cycle is to merge the PR once everyone has agreed to the suggested changes.

Always delete the old branch - recreate that branch from the *base* branch if you want to keep working in a branch with the same name.





## Exercise: Apply the branch-PR-merge cycle

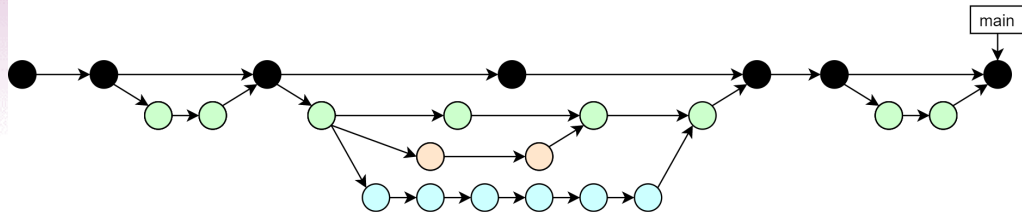
- Objective: Practice the complete branch-PR-merge cycle. We will use our <https://github.com/dime-wb-trainings/lyrics-msfr24> repository.
- Tasks:
  1. **Set-Up Stage:** Create and switch to a branch named "your\_name\_session2".
  2. **Work Stage:**
    - Commit Edits: Add or modify a song in the "lyrics" directory. Commit your changes and push them to GitHub.
    - Review PR: Open a pull request against the main branch. Pair up to review each other's PRs, focusing on constructive feedback using GitHub's comment and review tools.
    - Discuss Comments: Engage with your partner's feedback on your PR—discuss, accept, or reject suggestions.
  3. **Merge Stage:** Ensure all feedback has been addressed. After receiving peer approval, merge your pull request.

## **Part 3:**

# **Gitflow - When, how and how much to branch**

---

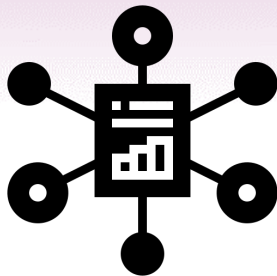
# The network graph



- A typical git network graph - each dot is a commit.
- Several branches have been opened and merged, but currently only the *main* branch is open.

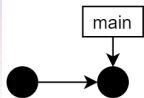
# Gitflow

- **Gitflow** is an idea or a philosophy of how to organize work in git - it is not a software you install.
- It is developed by computer scientists and you will find a lot of resources for it online.
- We will present a simplified version adapted for research
- Two branch types (later we add a third type):
  - The **main branch** (formerly master branch) - should never be worked in directly.
  - **Feature branches** - this is where all work should be done.



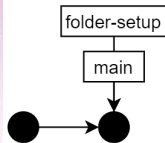
# Simple branch-PR-merge cycle

## Start a branch-PR-merge cycle



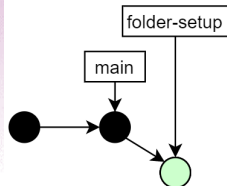
- This can either be the beginning of a repo or at a point where all previous branches were merged to the main branch.
- Let's say your next task is to set up folders - apply the *branch-PR-merge* cycle.

## Set up stage - 1



- Create a new *feature* branch at the point where you want the the *branch-PR-merge* cycle to start from.
- Two branches pointing to the same commit are identical by definition.

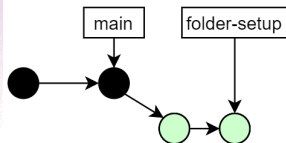
## Set up stage - 2



- The Author starts working on the task in the new branch.
- As soon as one commit is made you can open a new PR where this task can be reviewed and discussed - great case for a draft PR.

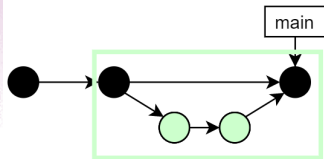


## Work stage



- Complete the task using one or many commits - in this example, only one more commit was needed.
- Assign someone to review the PR - iterate the work stage of the branch-PR-merge cycle until everyone is satisfied.

## Merge stage

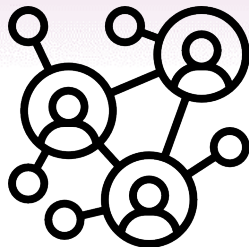


- Merge the PR and delete the *feature* branch.
- The green box represents a full *branch-PR-merge* cycle in the network graph.

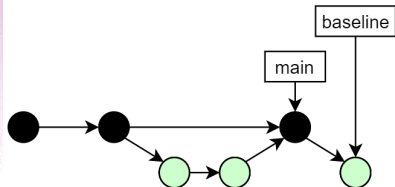
# Nested branch-PR-merge cycles

## Nested branch-PR-merge cycles

- Many high-level tasks are not simple - many team members working over multiple months.
- Cleaning the baseline data, for example, is one big task composed of many smaller tasks
- Solve this by using nested *branch-PR-merge* cycles
- Third type of branch: **Develop branch** - a branch that is a high-level task that will have several *feature* branches

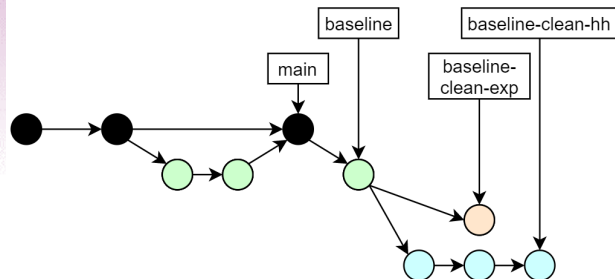


# Create a develop branch



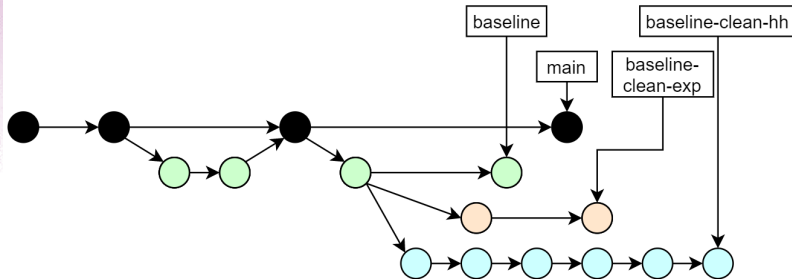
- Create a *develop* branch - name it after the high-level task.
- Include the name of the *develop* branch as prefix to all *feature* branches off it.

## Create feature branches of the develop branch



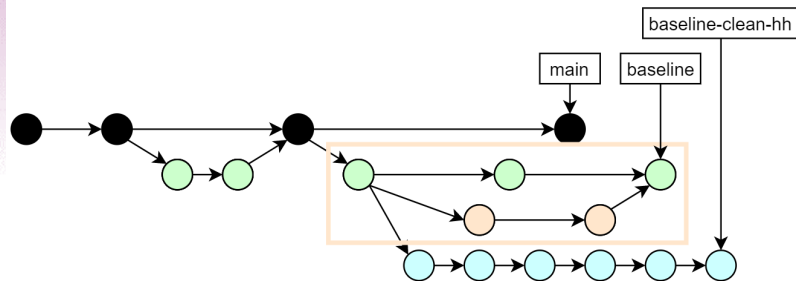
- Create as many *feature* branches off the *develop* branch as you need.
- Split up the high-level task in as many smaller tasks as needed - try to split up tasks until each is small enough for one person to complete in max a week or two.
- Create new *feature* branches off the *develop* branch even after it progresses.

# Work in the branches



- The Author keeps working on the *feature* branches until the task is done.
- When a task is complete: assign a reviewer - make comments/suggestions - accept/reject/discuss.

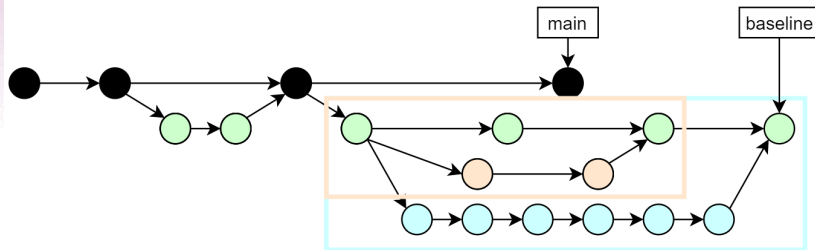
## Merge a feature branch



- The branch `baseline-clean-exp` was approved in the last step in the work stage and was merged and deleted in the merge stage.
- The orange box is one *branch-PR-merge* cycle.

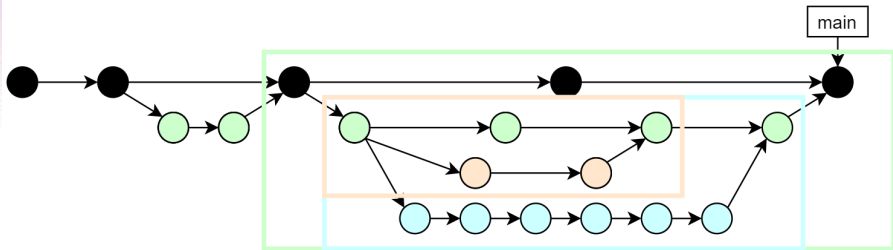


## Merge a feature branch



- The branch `baseline-clean-hh` was then also approved in the last step in the work stage and was merged and deleted in the merge stage.
- The blue box is also a *branch-PR-merge* cycle.

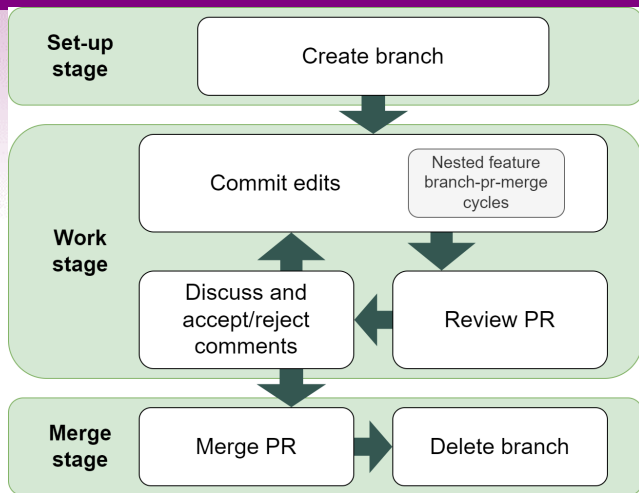
## Merge a develop branch



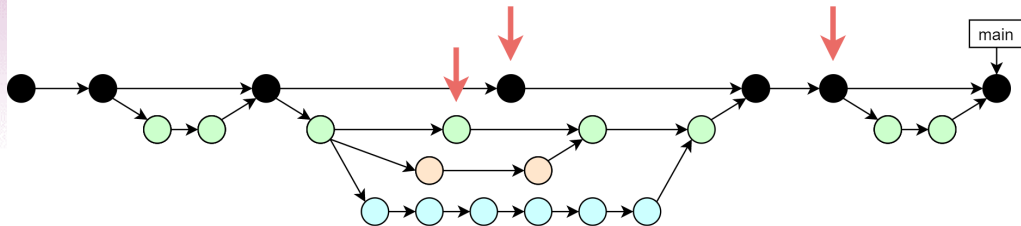
- Then the branch `baseline` is ready for a review - assign a reviewer - comment/suggest - accept/reject/discuss.
- The green box is also a *branch-PR-merge* cycle.
- This network graph shows a nested *branch-PR-merge* cycle.

## Nest cycles in the work stage

- *Branch-PR-merge* cycles should be nested in the work stage.
- The *develop* branch is kept in the *Commit edits* step while the *branch-PR-merge* cycles are completed for the feature branches.
- If needed, convert a *feature* branch to a *develop* branch here.

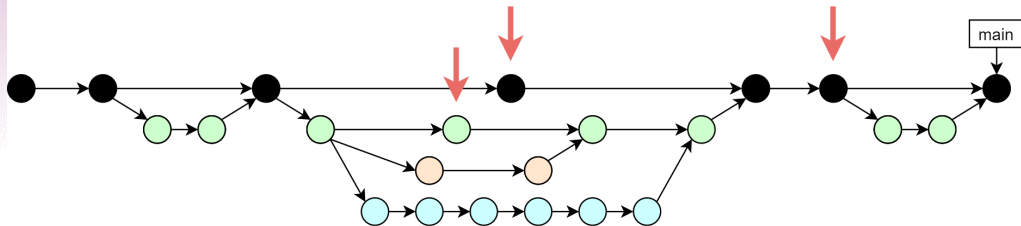


## Work directly in main/develop



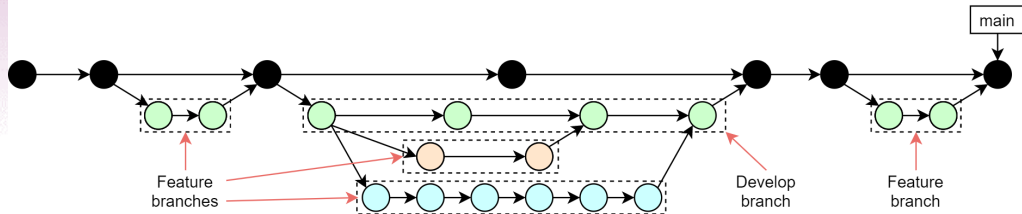
- When would it be ok to work directly on a *main/develop* branch? - see red arrows

# Work directly in main/develop



- When would it be ok to work directly on a *main/develop* branch? - see red arrows
- When updating documentation!
  - Documentation about the repo in the *main* branch.
  - Documentation about the high-level task in the *develop* branch.

# The network graph



- There is no difference to git between different Gitflow types of branches - the only difference is how team members use them.
  - **Main branch:** The branch where all branches originate from.
  - **Develop branch:** Non-*main* branch with *feature* branches.
  - **Feature branch:** Branches of the *main* branch or *develop* branches with no other branches from it.

## Exercise: Apply the Gitflow philosophy

- **Objective:** Apply Gitflow concepts by managing genre-specific development branches and personal feature branches.
- **Branch Structure:**
  - **Develop Branches:** Pre-created, genre-specific branches (e.g., *rock*, *pop*, *metal*). These mimic the "develop" branches in the Gitflow model.
  - **Feature Branches:** Each participant creates a new branch from their chosen genre's develop branch, naming it *genre\_your\_name* (e.g., *rock\_alice*). This is where individual work occurs.
  - **Main Branch:** The central branch from which all genre branches originate. This branch typically contains the consolidated and final version of the project.

## Exercise: Apply the Gitflow philosophy

- **Tasks:**

1. Choose a genre and create your feature branch from the respective genre's develop branch. Name your branch `genre_your_name`.
2. Add a new song to the genre folder in your branch, commit the changes, and push them to GitHub.
3. Open a pull request to merge your feature branch back into the develop branch for your chosen genre.
4. Let's now explore the network graph.

- **Discussion:**

- Reflect on how this branching strategy can be analogous to task management in complex projects, such as field research or software development.
- Discuss the benefits of organizing work into specific branches to maintain clarity and improve collaboration.



**Thank you!**

**Questions?**

## Useful links

- All DIME Analytics GitHub trainings: <https://osf.io/e54gy/>
- Other DIME Analytics GitHub resources:  
<https://github.com/worldbank/dime-github-trainings>. For example:
  - DIME Analytics GitHub Templates (for example .gitignore):  
<https://github.com/worldbank/dime-github-trainings/tree/master/GitHub-resources/DIME-GitHub-Templates>
  - DIME Analytics GitHub Roles:  
<https://github.com/worldbank/dime-github-trainings/blob/master/GitHub-resources/DIME-GitHub-Roles/DIME-GitHub-roles.md>
- Markdown cheat sheet (how to format text on GitHub.com):  
<https://www.markdownguide.org/cheat-sheet/>
- DIME GitHub Account admin info and instructions:  
<https://github.com/dime-worldbank/dime-account-admin>