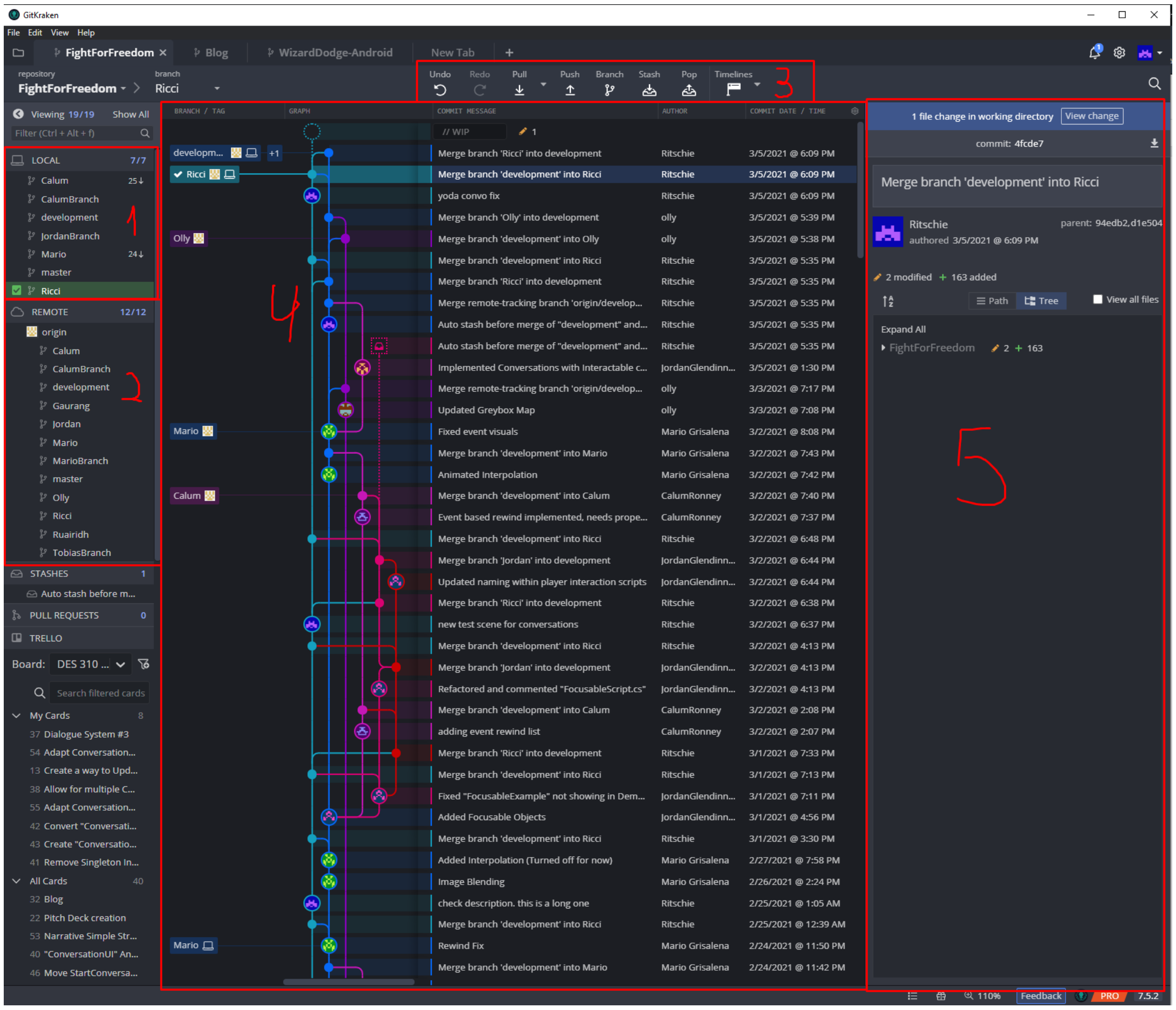
Git Basics

Introduction

In this document you can find information about the basics of using Git with Git Kraken. As well as tips and good practices to follow.

Part 1 - UI



COMMIT MESSAGE	AUTHOR	COMMIT DATE / TIME
// WIP 1		
Merge branch 'Ricci' into development	Ritschie	3/5/2021 @ 6:09 PM
Merge branch 'development' into Ricci	Ritschie	3/5/2021 @ 6:09 PM
yoda convo fix	Ritschie	3/5/2021 @ 6:09 PM
Merge branch 'Olly' into development	olly	3/5/2021 @ 5:39 PM
Merge branch 'development' into Olly	olly	3/5/2021 @ 5:38 PM
Merge branch 'development' into Ricci	Ritschie	3/5/2021 @ 5:35 PM
Merge branch 'Ricci' into development	Ritschie	3/5/2021 @ 5:35 PM
Merge remote-tracking branch 'origin/develop...	Ritschie	3/5/2021 @ 5:35 PM
Auto stash before merge of "development" and...	Ritschie	3/5/2021 @ 5:35 PM
Auto stash before merge of "development" and...	Ritschie	3/5/2021 @ 5:35 PM
Implemented Conversations with Interactable c...	JordanGlendinn...	3/5/2021 @ 1:30 PM
Merge remote-tracking branch 'origin/develop...	olly	3/3/2021 @ 7:17 PM
Updated Greybox Map	olly	3/3/2021 @ 7:08 PM
Fixed event visuals	Mario Grisalena	3/2/2021 @ 8:08 PM
Merge branch 'development' into Mario	Mario Grisalena	3/2/2021 @ 7:43 PM
Animated Interpolation	Mario Grisalena	3/2/2021 @ 7:42 PM
Merge branch 'development' into Calum	CalumRonney	3/2/2021 @ 7:40 PM
Event based rewind implemented, needs prope...	CalumRonney	3/2/2021 @ 7:37 PM
Merge branch 'development' into Ricci	Ritschie	3/2/2021 @ 6:48 PM
Merge branch 'Jordan' into development	JordanGlendinn...	3/2/2021 @ 6:44 PM
Updated naming within player interaction scripts	JordanGlendinn...	3/2/2021 @ 6:44 PM
Merge branch 'Ricci' into development	Ritschie	3/2/2021 @ 6:38 PM
new test scene for conversations	Ritschie	3/2/2021 @ 6:37 PM
Merge branch 'development' into Ricci	Ritschie	3/2/2021 @ 4:13 PM
Merge branch 'Jordan' into development	JordanGlendinn...	3/2/2021 @ 4:13 PM
Refactored and commented "FocusableScript.cs"	JordanGlendinn...	3/2/2021 @ 4:13 PM
Merge branch 'development' into Calum	CalumRonney	3/2/2021 @ 2:08 PM
adding event rewind list	CalumRonney	3/2/2021 @ 2:07 PM
Merge branch 'Ricci' into development	Ritschie	3/1/2021 @ 7:33 PM
Merge branch 'development' into Ricci	Ritschie	3/1/2021 @ 7:13 PM
Fixed "FocusableExample" not showing in Dem...	JordanGlendinn...	3/1/2021 @ 7:11 PM
Added Focusable Objects	JordanGlendinn...	3/1/2021 @ 4:56 PM
Merge branch 'development' into Ricci	Ritschie	3/1/2021 @ 3:30 PM
Added Interpolation (Turned off for now)	Mario Grisalena	2/27/2021 @ 7:58 PM
Image Blending	Mario Grisalena	2/26/2021 @ 2:24 PM
check description. this is a long one	Ritschie	2/25/2021 @ 1:05 AM
Merge branch 'development' into Ricci	Ritschie	2/25/2021 @ 12:39 AM
Rewind Fix	Mario Grisalena	2/24/2021 @ 11:50 PM
Merge branch 'development' into Mario	Mario Grisalena	2/24/2021 @ 11:42 PM

Merge branch 'development' into Ricci

Ritschie
authored 3/5/2021 @ 6:09 PM
parent: 94edb2, d1e504

2 modified + 163 added

1 - Branches you have used LOCALLY

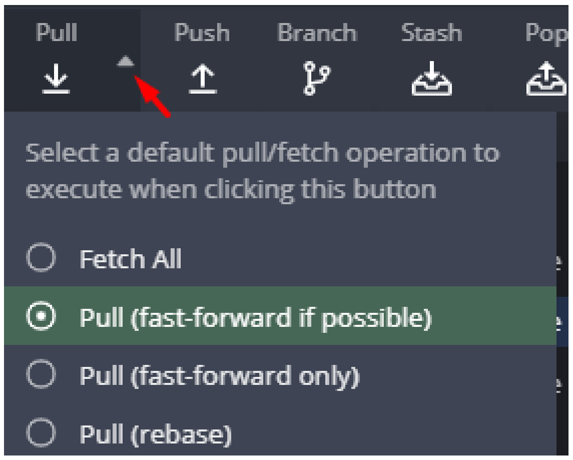
- Every change you make here is **only locally** on your PC until you decide to "push". What "push" does is send your local changes to **Remote** (aka the github server)
- WIth a green tickbox shows the current branch you are on

2 - Remote Branches => ALL branches in the project

This is a list of **every** branch we have in the Repository. You can use it to "checkout" (change to) any branch even if you have never done it before.

3 - Toolbar

Contains basic functions such as:



Select a default pull/fetch operation to execute when clicking this button

Fetch All [✓ Pull (fast-forward if possible)] Pull (fast-forward only) Pull (rebase)

- **Undo** - Undo the last commit on the current branch you are
- **Redo** - undo your undo on the current branch you are
- **Pull** - fast forward if possible will cover 99.99% of anyone's usecases, if you **need** to use the other options most probably a programmer will be doing it because of some bigger issue
    - **Fetch All** will check if there are any new changes on **ALL** branches, not only the current branch you are on
    - **Pull** (default is fast-forward if possible) will "fetch" (check to see if there are changes) and then "pull" (actually download the changes)
- **Push** - upload your changes to Remote (the server)
- **Branch** - creates a new branch from your current branch (a copy of your branch)
- **Stash** - stashes changes you select. Example use case is you were working on the wrong branch and want to move them to yours. You stash your changes, change to your desired branch and then apply the changes (apply the stash).
- **Pop** - Pops the stash aka deletes it

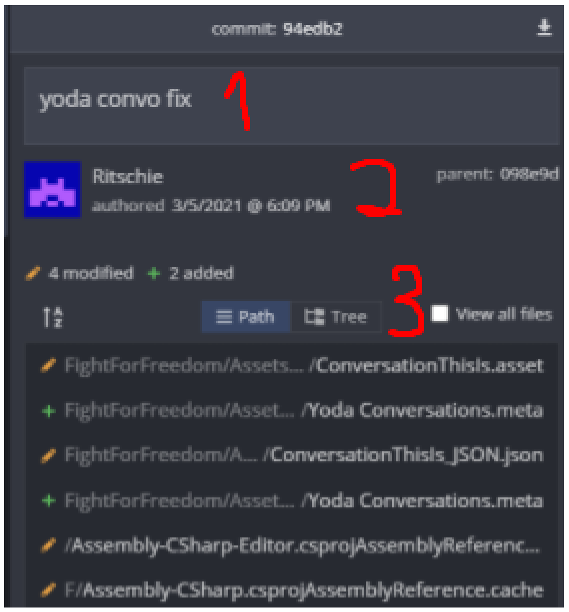
4 - Graphical History of the Repository and Branches

Basically shows where each branch is in relation to other branch. How far ahead it is, or how far behind it is, also you can see every commit that was made and when you select it you can get a detail view of the commit on the right panel. Also the Icons next to each branch name in this view shows whether the current state is Local only, or both Local and Remote. Only User Icon is the state of the Remote Branch. If there is the Laptop Icon then that is the state of the Local branch (meaning its a good idea to push your changes to Remote)

5 - Details View

Changes between details of a commit you have selected in the graph and making your own commit

- 1 - The name and description of the commit, important when backtracking through commits and looking for something.
- 2 - Information about who made the commit and when. Parent field is the Unique ID of the commit
- 3 - Details of what has changed in the commit.
    - **Pen** means it was edited
    - **+** means it it was added
    - **-** means it was deleted
    - **Blue file icon** means it was renamed



commit: 94edb2

yoda convo fix 1

Ritschie
authored 3/5/2021 @ 6:09 PM 2
parent: 098e9d

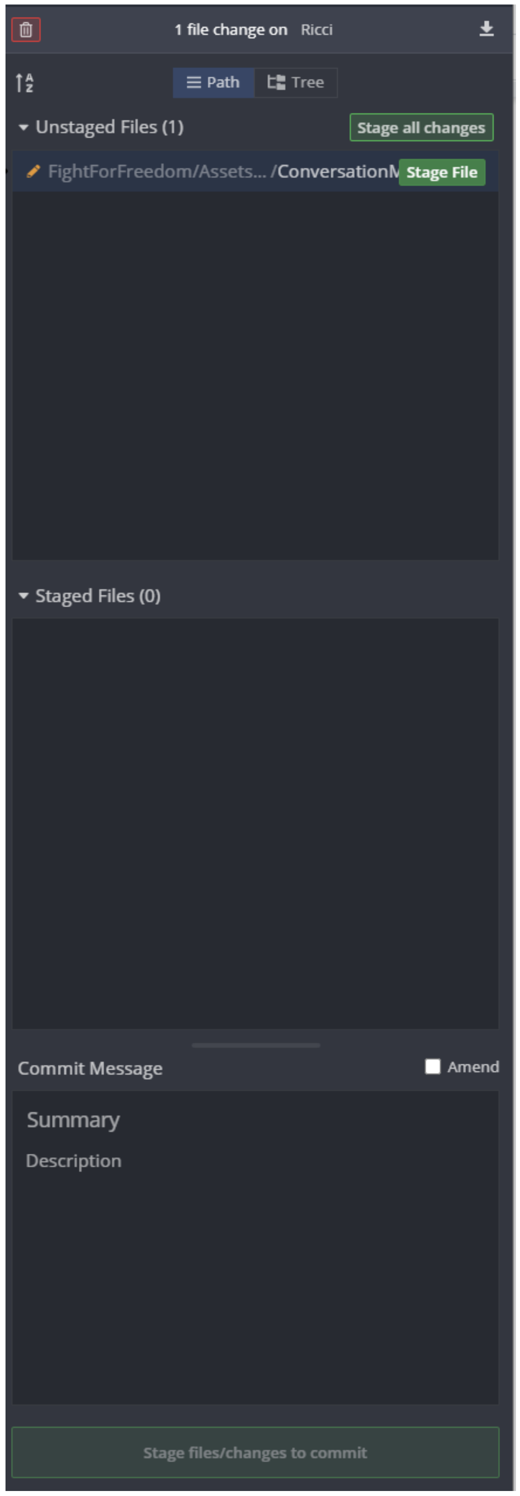
4 modified + 2 added 3

Path Tree View all files

FightForFreedom/Assets... /ConversationThisIs.asset
+ FightForFreedom/Asset... /Yoda Conversations.meta
FightForFreedom/A... /ConversationThisIs_JSON.json
+ FightForFreedom/Asset... /Yoda Conversations.meta
/Assembly-CSharp-Editor.csprojAssemblyReferenc...
F/Assembly-CSharp.csprojAssemblyReference.cache

Commit View

- **Unstaged Files** - files which will not be included in the commit. You can **Stage all changes,** which will include all of the files in the commit or you can Individually **Stage File** and only pick some of the files you want to commit. (usually used when you want to split a big change into smaller commits so that it is easier to keep track of)
- **Staged Files** - All of the files which will be included in the commit
- **Commit Message** - The Name of the commit and description. Naming your commit well helps everyone and it's no surprise there are memes about naming your commits, its an art form :D



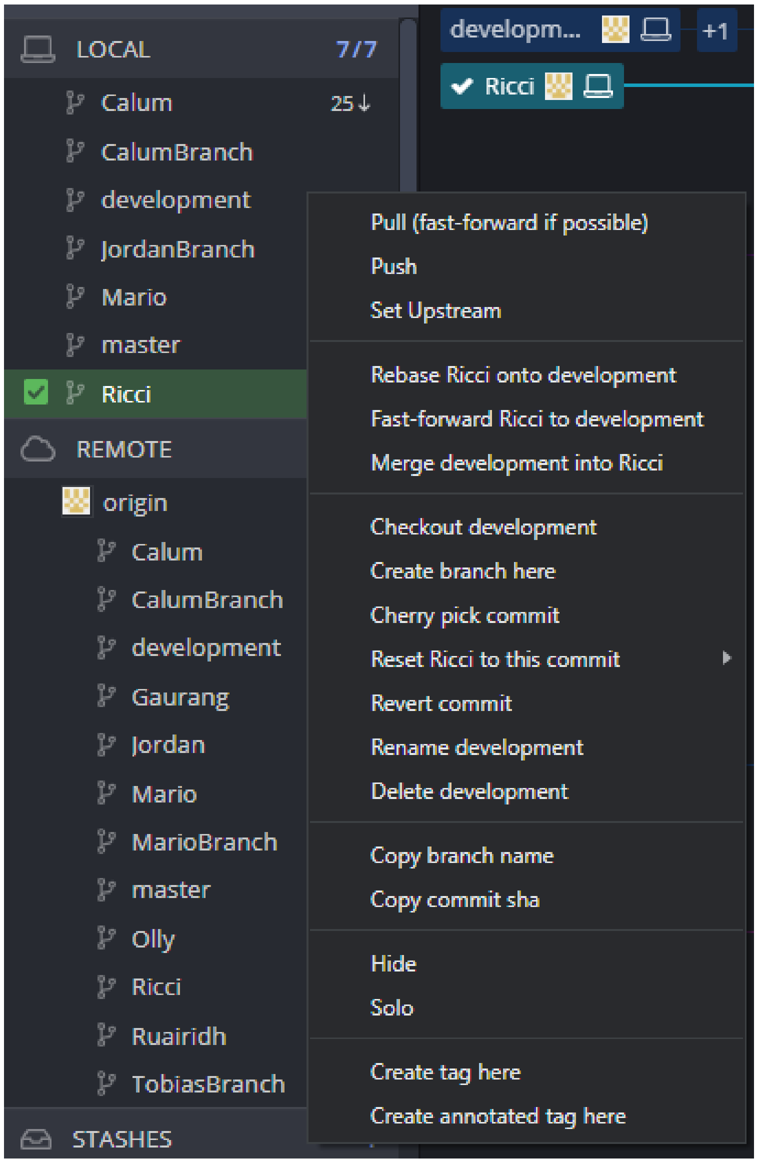
Part 2 - Navigating Branches

Changing from one branch to another

Important to note! **DO NOT** Change branches if you have **Uncommited changes**. It wont always cause an issue but it definitely could. If you have changes you are not ready to commit you can **stash** them. Although I recommend just commiting them, that is why we have our own branches and you can make as many commits to it as you want

There are 2 ways to change branches:

1. **Double Click** the branch you want to change to
2. **Right Click** the branch you want to change to **->** Press **Checkout {Name of branch you want to change to}**



Pull Changes on any branch

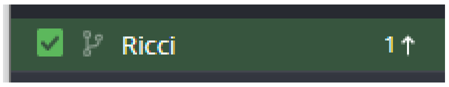
If a branch has a downwards arrow next to it, that means there are new changes on it and you need to pull to see them. Again, there are a couple of ways to do it

1. **Right Click** the branch **->** press **Pull** (easiest and recommended)
2. **Change to the branch ->** press Pull on top top Toolbar

Push Changes on a branch

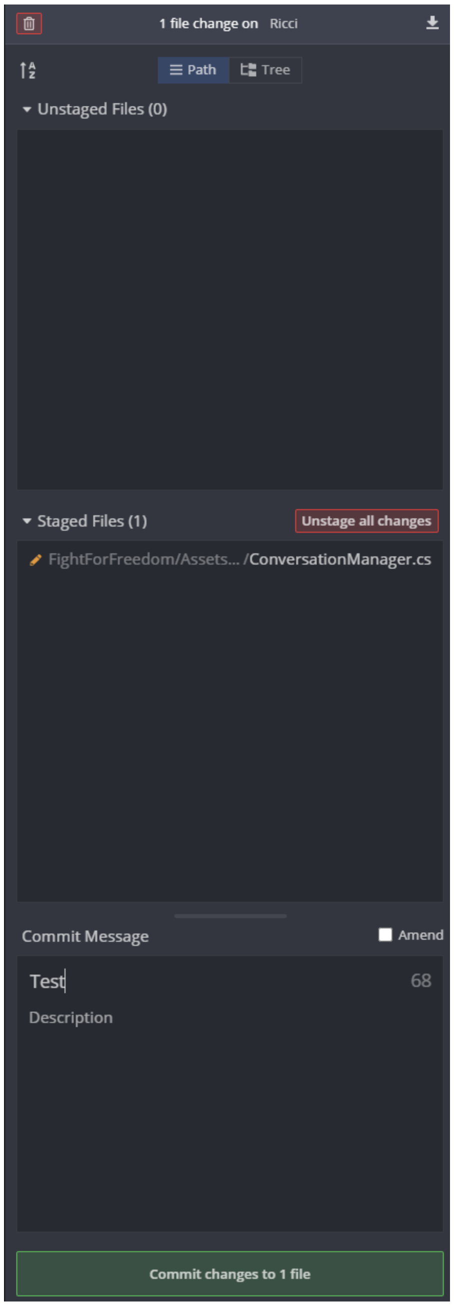
This has some prerequisites. In order to **Push** you first need to make a commit. To make a commit you need to stage all of the files you wish to commit and give the commit a message/name and then press the big green button below the description **Commit changes to X file(s)**

Once you do that, on the left in **Local,** next to the branch you are on you will see an **upwards arrow**. That means there are commits ready to be **pushed** to remote/server/repository.



After you have commit ready to be pushed you can do that in a couple of ways similarly to pulling

1. **Press Push** on the toolbar. Usually if you are pushing you are on the same branch as the push, hence the toolbar is the fastest and easiest way to do so.
2. **Right Click** on the branch **->** press **Push.**



FightForFreedom/Assets... /ConversationManager.cs

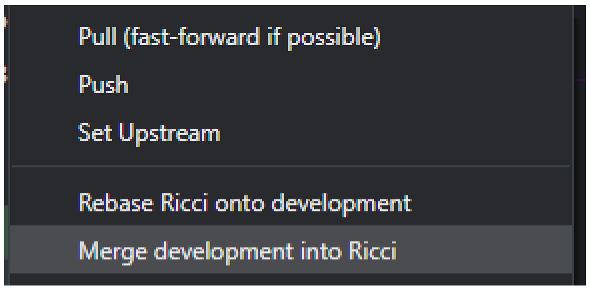
Test 68

Merge one Branch into another

There always comes a point where all the work you have on your personal branch needs to be moved to the development branch, that happens through **merge** (merging the branches). How to avoid the dreaded **merge conflicts** as much as possible and how to proceed best will be detailed in the next part.

The basics of merging could be summed up in a couple of steps and git kraken makes it very easy.

Situation, you want to merge **your** branch into **development:**


Pull (fast-forward if possible)
Push
Set Upstream
Rebase Ricci onto development
Merge development into Ricci

1. **Right Click** on **development**
2. Press **Merge {your branch name} into development**
3. Once you merge, you will notice that development now has an **upwards arrow**, meaning the changes/merge you just did is only **local**. You next need to **right click** development and press **PUSH**.

That will merge your branches and anyone else can then **pull** your changes.

Keep in mind that the order of the names of the branches matters!!

Merge {this is the branch you will merge **from**} into {this is the branch you will merge **into**}

The way GitKraken decides the order is simple. The more up-to-date branch will be merged into the older branch.

Another way to merge branches is something called a Pull Request/Merge Request. As far as I am aware that should not be something Art guys use, but I might be very wrong. In the scope of our Project PR's are overkill. We trust each other and are a small team. The bigger the team the more valuable PR's get. Pulll Reqeusts are in essence used for code review. I make a pull request and someone needs to approve it before it gets merged, usually that someone also reviews the code and leaves comments on what to change/fix. A PR can also contain multiple commits inside. **If needed I can go over them as well.**

Part 3 - Good practises/workflow

- **Master** - kept clean and in a working condition, basically our polished stuff go here and we make builds from this branch.
    - **development** - our dump, here we will merge our individual branches and fix issues and decide what goes into Master
        - **Personal branches** - everyone works on a separate branch on their tasks without interfering with the work of others.

Every time you start work your first job should be to merge development into your own branch so you get all new changes. You should also aim to merge and commit your changes to development as often as possible. Ideally when you stop working for the day, merge development into your branch, then if there are no issues merge your branch into development. This avoids merge conflicts caused from your branch being very outdated to development as much as possible

When merging your branch into development you can first merge development into your branch and fix any issues that arise there. When there are none, you merge your branch into development. **This avoids merge conflicts on the development branch in order to keep it in working condition**