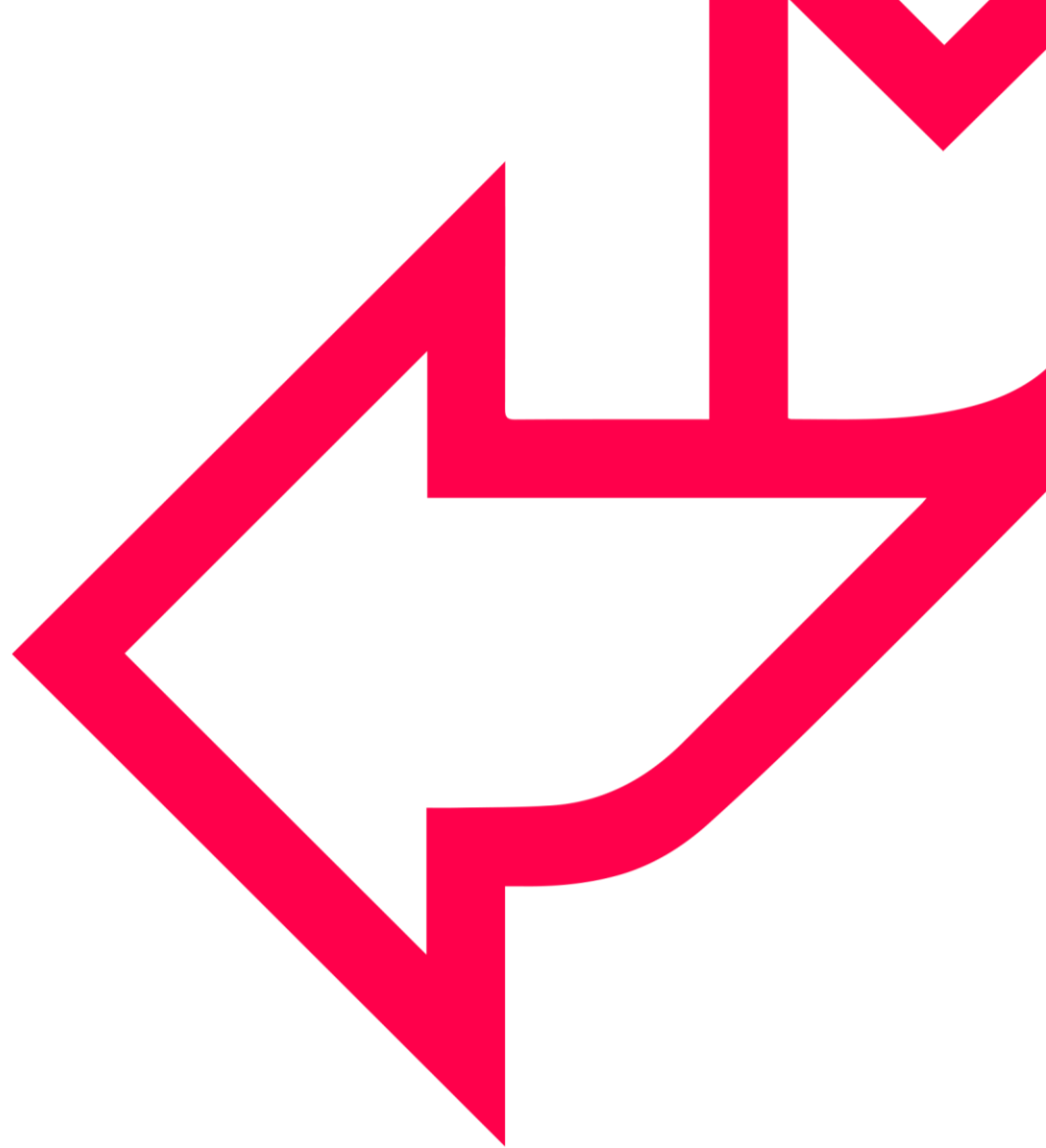# Branching

BRANCHING

Branching in Git helps us to define workflows that make sure the code that's being delivered is in the best state possible. That way, we minimise risks for any errors or crashes.
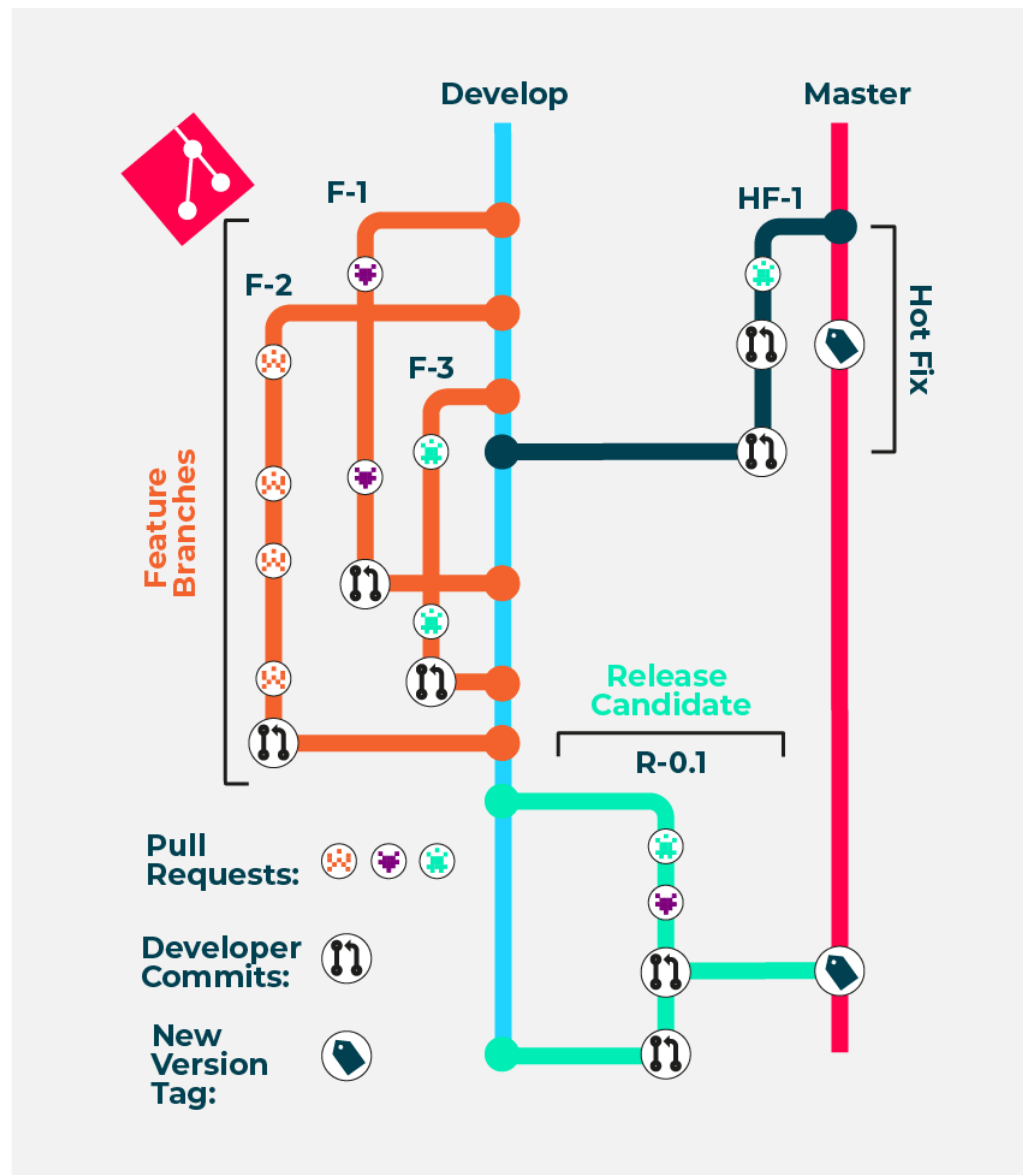
With Version Control Systems, like Git, we can separate the codebase on many different branches.

This feature can be utilised for isolating the development and testing of new features from working code that is running on a production environment.
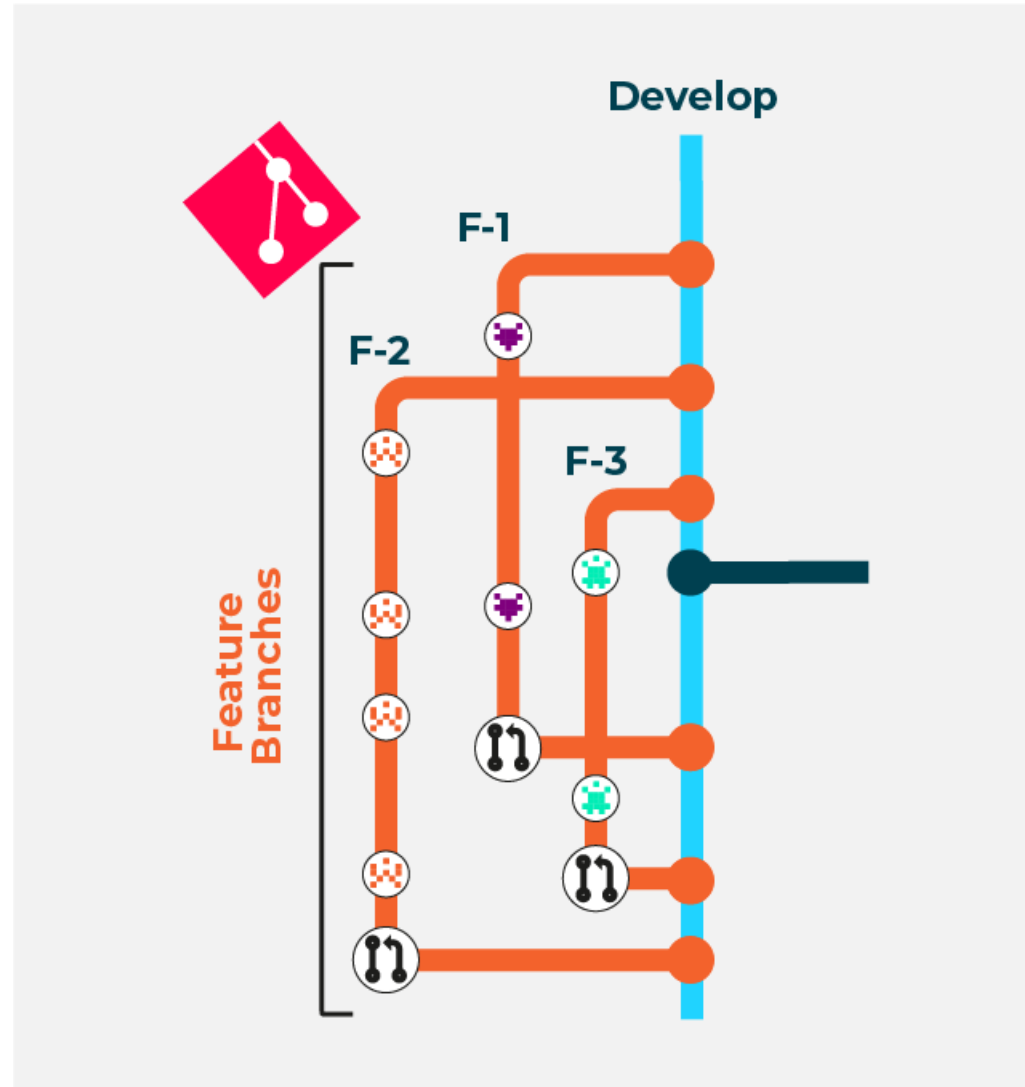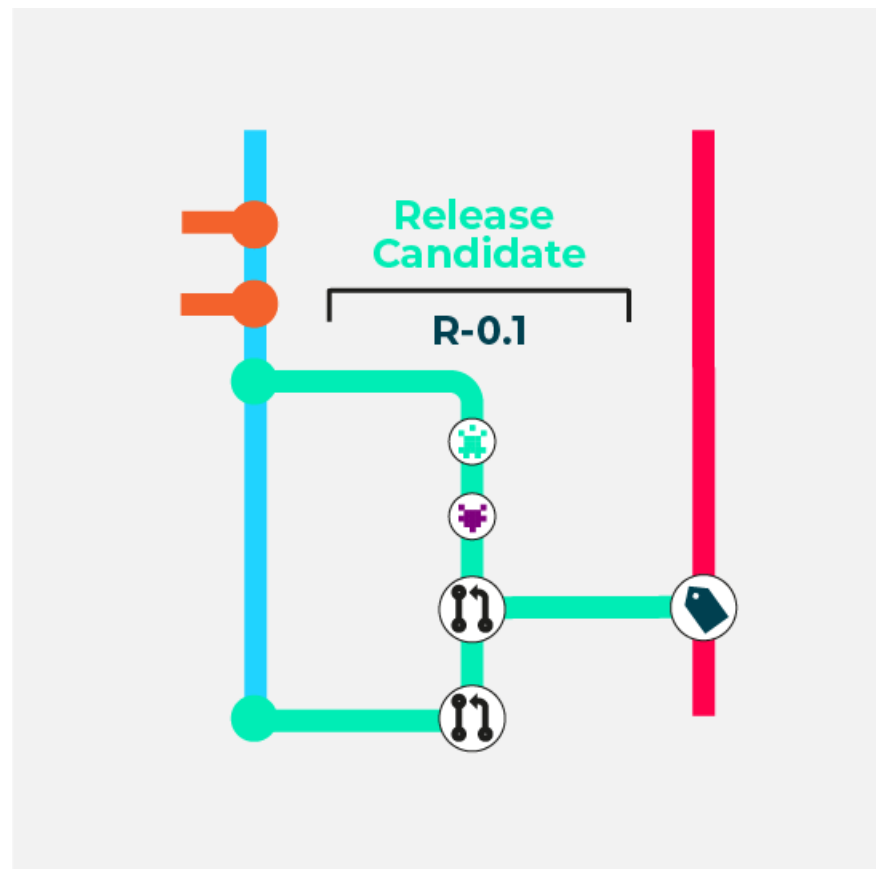
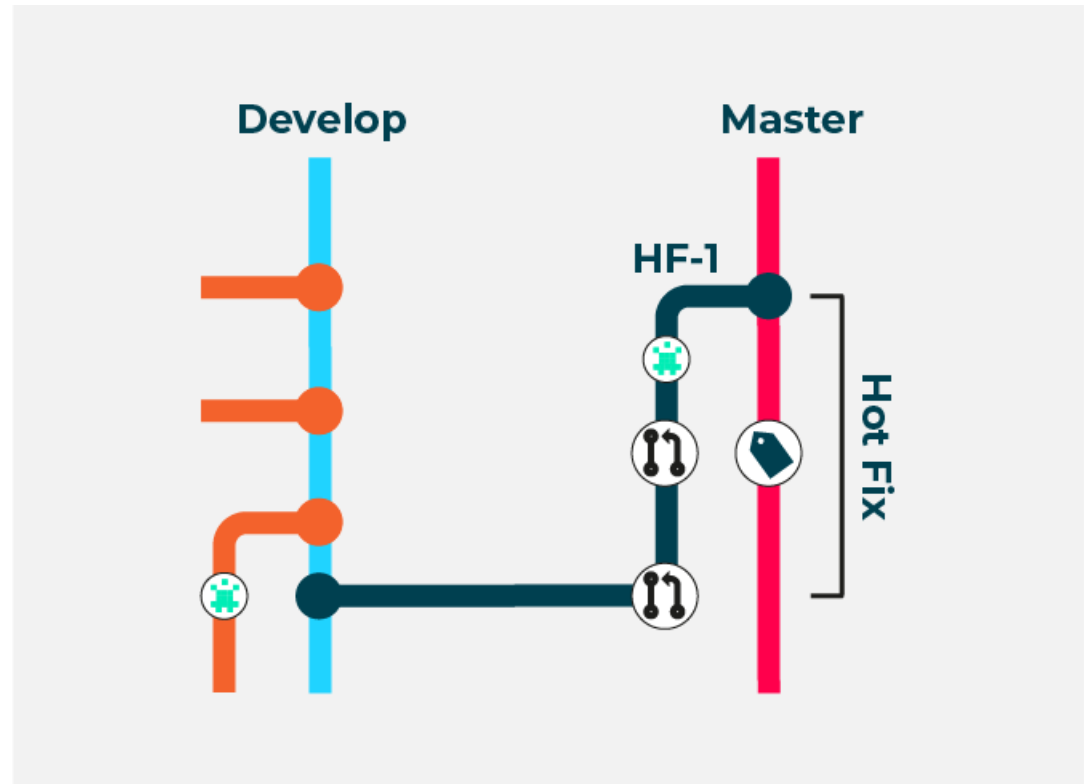# WORKFLOW EXAMPLE

WORKFLOW EXAMPLE -
NEW APPLICATION FEATURES

WORKFLOW EXAMPLE -
RELEASES

Release Candidate
R-0.1

WORKFLOW EXAMPLE -
HOT FIXES

CREATING AND BRANCHES

Command	Function
git **branch**	Show all branches in current repository
git **branch** [NEW_BRANCH_NAME]	Create a new branch
git **checkout** [BRANCH_NAME]	Switch to a branch
git **checkout** -b [BRANCH_NAME]	Switch to a branch; create branch if it doesn't exist
git **branch** -d [BRANCH_NAME]	Delete a branch
git **push** --delete origin [BRANCH_NAME]	Delete a branch in your remote repository

MERGING BRANCHES

```
E---F---G issue
       /

A---B---C---D master


        E---F---G issue
       /         \

A---B---C---D---H master
```

CONFLICTS

- The following would cause a conflict when trying to merge at commit

app.py at commit **D**	**app.py** at commit **G**
``hello = "Hello World!"

print(hello)``` | ```user = "Harry"
hello = f"Hello ${user}"

print(hello)``` |

REVERTING

Reverting in Git is a forward-moving way of undoing changes on a repository.

Reviewing the history of a repository

Command	Function
git **log**	Shows the commit history of the current branch
git **log** [BRANCH_NAME]	Passing an argument through allows you to display the commit history of a specific branch
git **log** --oneline	Shows a simplified view of the commit history
git **log** --branches=*	Shows the commit history for all branches in the current repository

GIT LOG

- When you run git log, it should look like this

REVERTING TO A PREVIOUS COMMIT

Command	Function
git **revert** [COMMIT]	Creates new commit that reverts changes to the code its state at the specified commit
git **revert** HEAD	Creates a new commit that reverts the repository back to the state of the previous commit
git **reset** --hard [COMMIT]	Reverts changes to the code to the state at the specified commit and discards the history after that point

Here you would refer to a commits hash code e.g. 483856a

REVERTING WITH RESET

```
git reset --hard bcabb84
```

- **This command will return the state to the selected commit**

  → Using Revert to go Back to the Latest Commit

  o **git reset --hard**