Git : L’essentiel

Travailler sur un projet de code entraîne des modifications qui peuvent provoquer des bugs. Ces modifications sont souvent oubliées au cours du temps. Un logiciel de versioning comme Git va permettre de garder la trace de toutes les modifications faites sur le code.
Chaque série de modifications (créer un fichier, supprimer un fichier, modifier un texte dans un fichier, etc.), sera enregistrée dans un commit (C’est une version du code à un instant t). Ainsi, en cas de bug, ou lorsque plusieurs travaillent sur un même projet, revenir en arrière sur une précédente version du code peut s'avérer bien utile...

Git et les autres

Les logiciels de gestion de version peuvent être basés sur différents modèles :

- **Modèle centralisé** : un serveur central contrôle toute la base de code du logiciel.
- **Modèle distribué** : toutes les machines ont accès à la base de code, pas besoin de passer par un serveur central.

Le modèle distribué (Git en fait partie) présente plusieurs avantages :

- Moins de risques de perdre son code puisqu'il est accessible par plusieurs sources.
- On peut travailler plus rapidement et sans être connecté à Internet puisqu'il n'y a pas besoin de se connecter à un serveur central.

Git facilite la collaboration et les échanges fructueux entre développeurs.

Les commandes de base de la console

- *pwd* : permet de connaître votre répertoire courant
- *ls* : permet de voir la liste des fichiers et répertoires dans le dossier courant
- *ls –l –a* : l'option -l permet de voir les éléments de votre répertoire courant sous forme de liste et l’option –a permet d’afficher plus d'informations sur ces éléments.
- *cd Dossier*: permet de se placer dans le répertoire Dossier
- *cd ~* : permet de revenir dans votre répertoire principal
- *touch fichier.ext :* permet de créer un fichier.ext
- *mkdir rep :* permet de créer un dossier rep
- *cat file.ext :* permet d'afficher le contenu du fichier file.ext

Vim est un éditeur de texte disponible dans la console.

- *vim mon_fichier.txt* : ouvre le fichier mon_fichier.txt sur la console (a pour commencer à écrire et échap pour arrêter l’ecriture
- *:w* : Pour sauvegarder le fichier
- *:q* : Pour quitter Vim
- *:x* : Pour sauvegarder le fichier et quitter Vim

Utilisation de Git pour les Commit, ajout fichiers …

- *git init* : activer un dossier comme repository Git, il suffit de se placer dans ce dossier et de lancer cette commande
- *git add nomDeVotreFichier.extension* : Lorsque vous créez un fichier dans un repository, vous devez donc l'ajouter à l'index Git à l'aide de la commande précédente
- *git add .* : Pour gagner du temps, vous pouvez ajouter tous les fichiers dans le répertoire courant en tapant

git commit -m "Description de La modification " : Effectuer un commit. L’option –a demande à Git de mettre à jour les fichiers déjà existants dans son index.

Lire l'historique

Au cours d'un projet, vous allez être amenés à faire beaucoup de modifications...

- *git log* : affiche la liste de tous les commits réalisés (commit le plus récent en haut)

Dans la liste de cet historique, chaque commit est répertorié avec :

- *SHA* : son identifiant unique, qui se présente sous forme d'une longue chaîne de caractères et de nombres (Ex : "87753191cef0bdb955a4cb4ff841f7c2cce4cb1c")
- *Auteur* : qui a fait le commit (utile lorsque vous travaillez à plusieurs sur un projet !)
- *Date*
- *Description* : message de description indiqué avec l'option -m

Pour mettre à jour un fichier dans le repository, il faut :

- Ajouter le fichier à l'index avec la commande git add,
- Faire un commit qui décrit la mise à jour de votre fichier avec la commande git commit.

Se positionner sur un commit donné

- *git checkout SHADuCommit* : se positionner sur un commit donné de votre historique
- *git checkout master* : revenir à votre branche principale (au commit le plus récent)

On ne peut pas annuler ou supprimer un commit, mais :

- *git revert SHADuCommit* : crée **un nouveau** commit qui fait l'inverse du précédent
- *git commit --amend -m "Votre nouveau message"* : modifier le message de votre dernier commit.
- *git reset –hard* : annuler tous les changements non commités.

Git & Bitbucket

What to know about repositories :

- access to all files in your local repository (working on one or multiple files).
- view public repositories without a Bitbucket account if you have the URL.
- Each repository belongs to a user account or a team.
- The repository owner is the only person who can delete the repository. If the repository belongs to a team, an admin can delete the repository.
- A code project can consist of multiple repositories across multiple accounts but can also be a single repository from a single account.
- Each repository has a 2 GB size limit, but we recommend keeping your repository no larger than 1 GB.

Create the repository

Bitbucket Repository initially empty without any code in it and will be the central repository for your files, which means that others can access that repository if you give them permission. After creating a repository, you'll copy a version to your local system—that way you can update it from one repo, then transfer those changes to the other.

- From Bitbucket, click the + icon in the global sidebar and select Repository.
- Enter the BitbucketNameRepo
- Private repository is only visible to you and those you give access to.
- Pick Git for the Repository type and Create.

Copy your Git repository and add files

Clone a repository to create a connection between the Bitbucket server (which Git knows as origin) and your local system.

- *Mkdir name_of_repo* : Create a directory to contain your repositories
- *cd ~/name_of_repo* : update the directory to the new name_of_repo directory.
- From Bitbucket, go to BitbucketNameRepo, click the + icon in the global sidebar, select Clone this repository and copy the highlighted clone command on your terminal window
- Enter your Bitbucket password when the terminal asks for it.
- *ls* : to list the contents of your name_of_repo directory

Add a file to your local repository and put it on Bitbucket

- *cd ~/name_of_repo/ BitbucketNameRepo /* : navigate to the top level of your local repository
- *echo "Earth's Moon" >> locations.txt* : create a new file with content (If the command line doesn't return anything, it means you created the file correctly!)
- *git status* : Get the status of your local repository.
- *git add locations.txt* : Tell Git to track your new locations.txt file (doesn't return anything when you enter it correctly)
- *git status* : Check the status of the file to see if the new file has been added (staged) and you can commit it when you are ready.
- *git commit -m 'Initial commit'* : Issue the git commit command

PS : Up until this point, everything you have done is on your local system and invisible to your Bitbucket repository until you push those changes. Users typically need to share a series of commits rather than a single changeset. Git lets you share entire branches between repositories. You manage connections with other repositories and publish local history by **"pushing"** branches to other repositories. You see what others have contributed by **"pulling"** branches into your local repository.

- *git push origin master* : send your committed changes to Bitbucket. This command specifies that you are pushing to the master branch (the branch on Bitbucket) on origin (the Bitbucket server).

Go to Bitbucket Repository, click on Commits to see all the commits and click on Source, to see the files added.

Pull changes from your Git repository on Bitbucket Cloud

Create a File on Bitbucket

- From Source page, click New file in the top right corner (This button only appears after you have added at least one file to the repository)
    - Branch with new file: Change if you want to add file to a different branch.
    - New file area: Add content for your new file here.
    - Enter name in the filename field.
    - Select language from the Syntax mode list.
    - Add code into the text box
    - Click Commit.

Pull changes from a remote repository
To get that new file into your local repository.

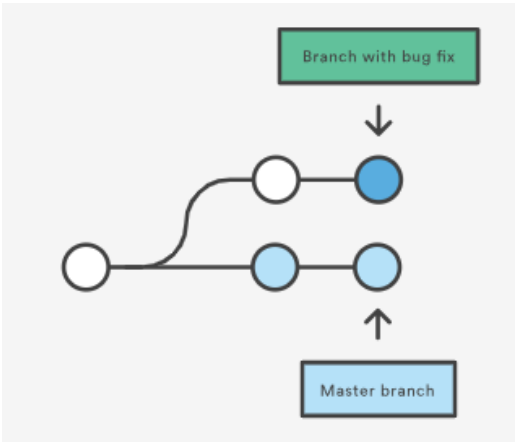
- *cd ~/repos/bitbucketstationlocations/* : navigate to the top level of your local repository.
- *git pull --all* : to pull all the changes from Bitbucket.

Navigate to your repository folder on your local system and you'll see the file you just added.

Use a Git branch to merge a file

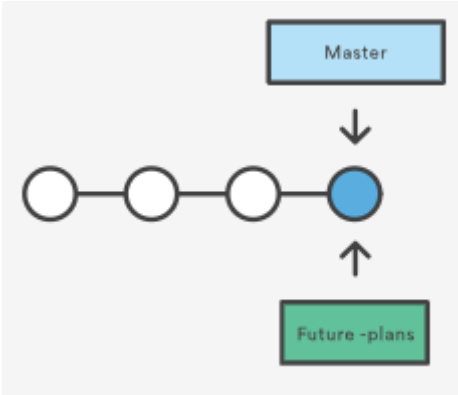
Learning branches to be able to update the files and sharing the information only when it's ready (very powerful when if working on a team). Possibility to work on a project from your own branch, pull updates from Bitbucket, and then merge all your work into the main branch.

Before creating any new branches, automatically start out with the main branch (master).

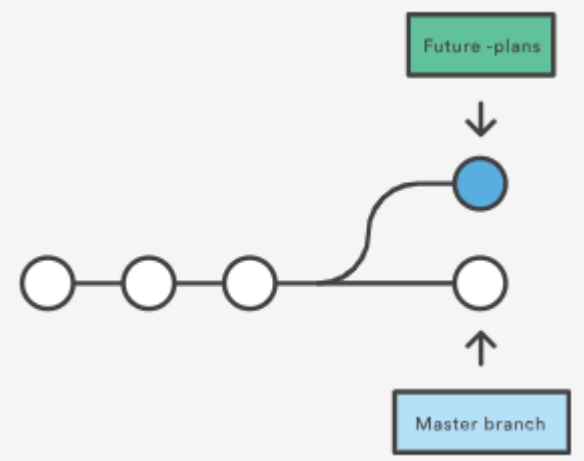


Create a branch and make a change
Create a branch where to add future plans for the space station that aren't ready to commit. When they are ready to be known to all, merge the changes into the Bitbucket repository and delete the no-longer-needed branch.

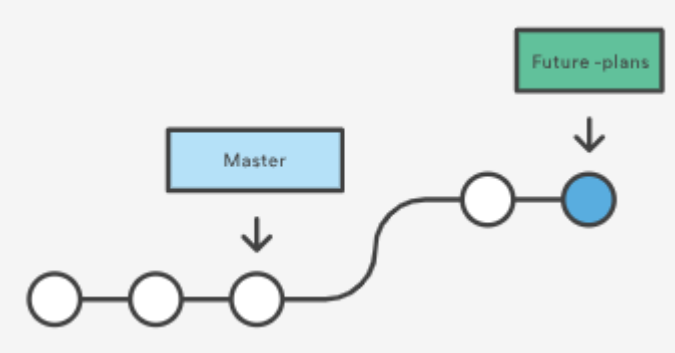
- *cd ~/name_of_repo/BitbucketRepoName/* : navigate to the top level of your local repository
- *git branch Branch_Name* : Create a branch but does not switch to that branch.



- *git checkout Branch_Name* : Checkout the new branch you just created to start using it

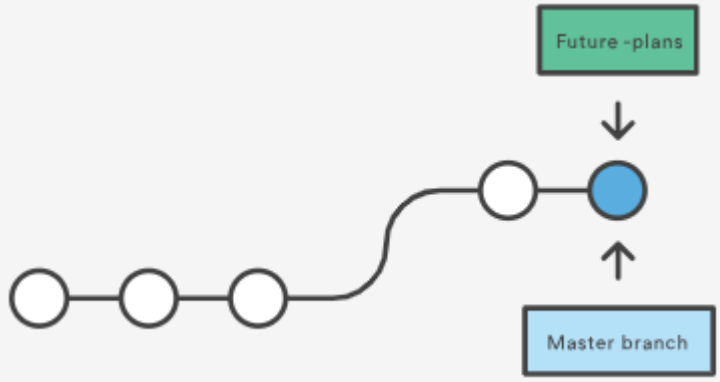


- Search for the *BitbucketRepoName* folder on local system and open it. Notice there are no extra files or folders in the directory as a result of the new branch. Open the filename file using a text editor, make a change to the file, save it and close it.
- *git status* : message telling that filename was modified on branch BranchName
- *git add filename* : stage the file
- *git commit filename -m 'making a change in a branch'* : commit



Merge your branch: fast-forward merging

Now that changes on the branch are ready, merge them into the main branch on your local system. If there is only one branch one change, use the fast-forward branch method to merge (linear path from current branch). This effectively combines the histories, since all of the commits reachable from the target branch are now available through the current one.

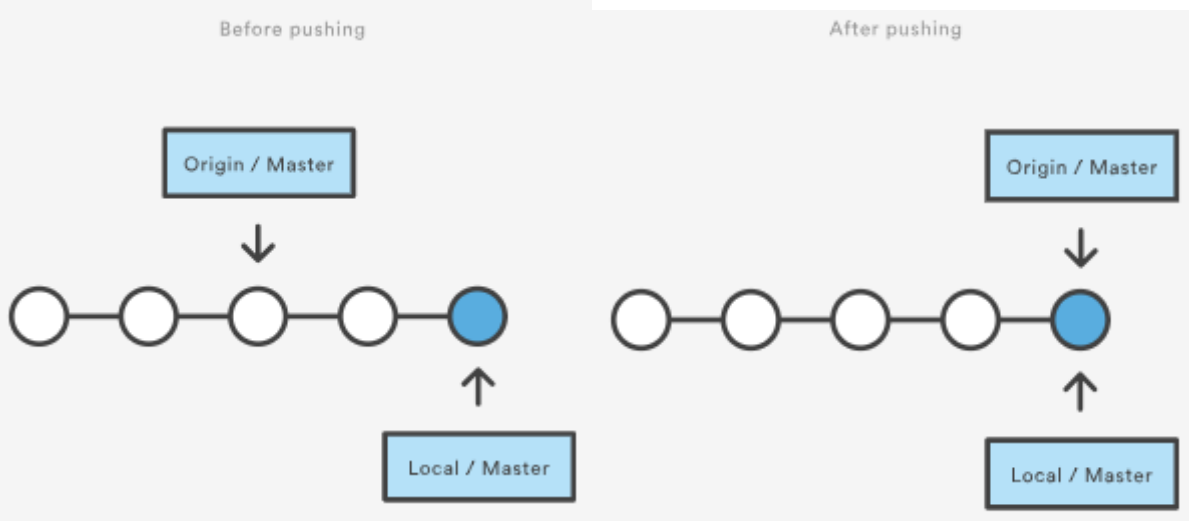


To complete a fast-forward merge do the following :

- *cd ~/repos/BitbucketRepoName/* : navigate to the repo from terminal
- *git status* : to check if nothing to commit and working directory clean
- *git checkout master* : Switch to the master branch
- *git merge Branch_Name* : merge changes from branch into master branch
- *git branch -d Branch_Name* : delete the branch (commit id_of_BranchName to go back to that branch)
- *git status* : to see if everything is done and need to use push

Push your change to Bitbucket

Push the current state of your local repository to Bitbucket to make it possible for everyone else to see the location of the new space station.



Here's how to push your change to the remote repository:

- *git push origin master* : To push to the repository
- click the Overview page of your Bitbucket repository to notice the push in the Recent Activity stream and click Commits to see the commit made on the local system. Click Branches and notice that the page has no record of the branch either and click Source, and then click the filename file to see the last change to the file has been made by the commit pushed.

Learn About Code Review

A branch represents an independent line of development for your repository. After you create a branch, you work on and commit code to that branch, pull updates from Bitbucket to keep your branch up-to-date, and then push all your work to Bitbucket.
Once you've got code changes on a branch in Bitbucket, you can create a pull request, which is where code review takes place. Your teammates will comment on your code with feedback and questions and eventually (hopefully) approve the pull request. When you have enough approvals, merge the pull request to merge your branch into the main code.

Create a repository and add a teammate

To get started, we'll walk through creating a team repository with some content and giving someone access.

Create a team and add a teammate

Start by creating a team for your repository and teammate.

- Click + in the global sidebar and select Team under Create.
- Enter a Team name you'd like to use and click done
- Click Settings, User groups under Access management, click Developers group, search for username or emails address and click Add.
- Now, when you create a pull request, you'll have someone to review it.

Create a repository with some content

- Click **+** in the global sidebar and select **Repository** under **Create**.
- Make sure the team you created is the repository **Owner**.
- Enter anything you want for the **Project name** and **Repository name**.
- From **Include a README?**, select either of the **Yes** options.
- From **Version control system**, pick an option for the type of repository you want to create. If you're not sure, keep as is.
- Click **Create repository** and you'll land on the **Source** view of your brand, new repository.
- From **Source**, select > **Add file**.

Clone and make a change on a new branch

- Clone your repository to your local system (copy the clone commande and paste it in the terminal after accessing the local directory)
- Create a branch and pull in locally : Click Branches, click Create a branch in the top right corner. (*git fetch && git checkout BranchName* to check in on local system)
- Make a change to the branch (change the file, add more files, …)
- *git status* : to see that there is a change
- *git add filename*
- *git commit –m « message »*
- *git push origin BranchName*

Create a pull request to merge your change

To alert your teammates to your updates and get their approval, your next step is to create a pull request. In addition to a place for code review, a pull request shows a comparison of your changes against the original repository (also known as a diff) and provides an easy way to merge code when ready.

Create the pull request

You need a branch to create a pull request.

- From your repository, click **+** in the global sidebar. Then, click **Create a pull request** under
- Complete the form, add your team member as a reviewer and and click on create pull request

Merge your pull request

Just click on the Merge button (you need to wait for an approval of your changes first)
From the pull request, the reviewer can view the diff and add comments to start a discussion before clicking the Approve button.

Learn Branching with Bitbucket Cloud

- **Clone**: copying the remote repository in Bitbucket Cloud to your local system
- **Add or stage**: taking changes you have made and get them ready to add to your git history
- **Commit**: add new or changed files to the git history for the repository
- **Pull**: get new changes others have added to the repository into your local repository
- **Push**: get changes from your local system onto the remote repository

What is a fork?
Fork is another way of saving a clone or copy. The term fork (in programming) derives from a Unix system call that creates a copy of an existing process. So, unlike a branch, a fork is independent from the original repository. If the original repository is deleted, the fork remains. If you fork a repository, you get that repository and all of its branches.

- Go to tutorials/tutorials.git.bitbucket.org
- Click **+ > Fork this repository** on the left side of the screen.
- Modify the **Name** so it is unique to your team, then click **Fork repository**.
- Create a directory for the repository which will be easy to navigate to.
- Clone the forked repository into the directory you just created.
- Create a branch and change something using the branching workflow
    - *git branch BranchName*
    - *git checkout BranchName*
    - *git branch* : if you want to list the branches
    - make an update to one of the files
    - *git add filename* : add the change
    - *git commit filename –m « description »* : Commit the change
- *git push* : it doesn't work because the first time you push a new branch you created locally you have to designate that branch.
- *git push origin BranchName*

Create, fetch, and checkout a remote branch
- Go to Bitbucket Repo
- Click Branches and create a branch BName
- Window : *git fetch && git checkout BName*
- *git branch* : to see the list of branches (the branch with * is the active one)
- *git status* : to see if there is sthg to commit
- *git checkout name* : to move on the branch name
- click + Create a pull request, click Approve (Other reviewvers should make comment and approve) and Merge
- Select the Merge Commit :
    - Merge commit : Keeps all commits from source branch and makes them part of the destination branch (=*git merge –no—ff*)
    - Squash : Combines the commits when you merge the source branch into the destination branch (=*git merge –squash*)

Delete a branch

- *git checkout master*
- *git pull*
- *git branch –d BranchName* : delete BranchName
- *git merge master BName* : to merge master branch into working branch

The active branch matters : if we merge master into BName, we want to have BName checked out (active) and vice-versa.

Start with the master branch

This workflow helps you collaborate on your code with at least one other person. As long as your Bitbucket and local repos are up-to-date, you're ready to get started.

Create a new-branch

Use a separate branch for each feature or issue you work on. After creating a branch, check it out locally so that any changes you make will be on that branch.

Update, add, commit, and push changes

Work on the feature and make commits like you would any time you use Git. When ready, push your commits, updating the feature branch on Bitbucket.

Get your code reviewed

To get feedback on your code, create a pull request in Bitbucket. From there, you can add reviewers and make sure everything is good to go before merging.

Resolve feedback

Now your teammates comment and approve. Resolve their comments locally, commit, and push changes to Bitbucket. Your updates appear in the pull request.

Merge your branch

Before you merge, you may have to resolve merge conflicts if others have made changes to the repo. When your pull request is approved and conflict-free, you can add your code to the master branch. Merge from the pull request in Bitbucket.

More Info :

- https://www.atlassian.com/git/tutorials/learn-undoing-changes-with-bitbucket

- https://www.atlassian.com/git/tutorials/what-is-version-control

- https://www.atlassian.com/git/tutorials/setting-up-a-repository

- https://www.atlassian.com/git/tutorials/syncing

- https://www.atlassian.com/git/tutorials/svn-to-git-prepping-your-team-migration

- https://www.atlassian.com/git/tutorials/advanced-overview