Commonly used Git command line commands

**git init**: Initializes a new Git repository in the current directory. This creates a .git directory that contains all the necessary files for version control[[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/PDfs/_Technical%20Books/GIT_Succinctly.pdf?web=1).

git init

1. **git status**: Displays the state of the working directory and the staging area. It shows which files have been modified, which are staged for the next commit, and which are untracked[[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/PDfs/_Technical%20Books/GIT_Succinctly.pdf?web=1).

git status

1. **git add**: Adds files to the staging area. You can add individual files or use git add . to add all changes in the current directory[[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/PDfs/_Technical%20Books/GIT_Succinctly.pdf?web=1).

git add filename

git add .

1. **git commit**: Records changes to the repository. You can add a commit message using the -m option[[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/PDfs/_Technical%20Books/GIT_Succinctly.pdf?web=1).

git commit -m "Initial commit"

1. **git log**: Displays the commit history. You can see the list of commits along with their messages, authors, and timestamps[[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/PDfs/_Technical%20Books/GIT_Succinctly.pdf?web=1).

git log

1. **git push**: Uploads local repository content to a remote repository. You can specify the remote name and branch[[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/PDfs/_Technical%20Books/GIT_Succinctly.pdf?web=1).

git push origin main

1. **git pull**: Fetches changes from a remote repository and merges them into the current branch[[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/PDfs/_Technical%20Books/GIT_Succinctly.pdf?web=1).

git pull origin main

1. **git merge**: Combines changes from different branches. You can merge a specified branch into the current branch[[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/PDfs/_Technical%20Books/GIT_Succinctly.pdf?web=1).

git merge branchname

1. **git clone**: Creates a copy of an existing repository. You can specify the repository URL and the directory where you want to clone it[[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/PDfs/_Technical%20Books/GIT_Succinctly.pdf?web=1).

git clone https://github.com/username/repository.git

1. **git branch**: Lists, creates, or deletes branches. You can use git branch to list all branches, git branch branchname to create a new branch, and git branch -d branchname to delete a branch[[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/PDfs/_Technical%20Books/GIT_Succinctly.pdf?web=1).
2. git branch
3. git branch newbranch
4. git branch -d oldbranch
5. **git checkout**: Switches branches or restores working directory files. You can use git checkout branchname to switch to a different branch[[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/PDfs/_Technical%20Books/GIT_Succinctly.pdf?web=1).
6. git checkout branchname
7. **git rebase**: Reapplies commits on top of another base tip. This is useful for integrating changes from one branch into another[[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/PDfs/_Technical%20Books/GIT_Succinctly.pdf?web=1). bash git rebase branchname

These commands are essential for managing your Git repositories and collaborating with others. If you need more detailed information or have any other questions, feel free to ask![[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/PDfs/_Technical%20Books/GIT_Succinctly.pdf?web=1)[[2]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/_GIT/9781785287909-GIT_ESSENTIALS.pdf?web=1)[[3]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/__VSCode/7-visual-studio-code-m7-slides.pdf?web=1)

**References**

[1] [GIT\_Succinctly](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/PDfs/_Technical%20Books/GIT_Succinctly.pdf?web=1)

[2] [9781785287909-GIT\_ESSENTIALS](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/_GIT/9781785287909-GIT_ESSENTIALS.pdf?web=1)

[3] [7-visual-studio-code-m7-slides](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/__VSCode/7-visual-studio-code-m7-slides.pdf?web=1)

The difference between git add . and git add -A lies in what changes they stage for the next commit:

**From your company**

In the [**GitCommands**](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/__github/GitCommands.txt?web=1&EntityRepresentationId=d5b6ccc3-9be3-4a9a-a9a0-f4f3618fbc84) document authored by [Stephan Onisick](https://www.office.com/search?q=Stephan+Onisick&EntityRepresentationId=6f597de2-91f7-4caa-ab26-2046b5b0c517), it is explained that git add . adds all changes in the current directory and its subdirectories, including new files and modifications, but it does not stage deletions[[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/__github/GitCommands.txt?web=1).

**From the Web**

1. **git add .**: This command stages new files and modifications in the current directory and its subdirectories, but it does not stage deletions[[2]](https://stackoverflow.com/questions/572549/difference-between-git-add-a-and-git-add).
2. **git add -A**: This command stages all changes in the entire working directory, including new files, modifications, and deletions[[2]](https://stackoverflow.com/questions/572549/difference-between-git-add-a-and-git-add).

In summary, git add . is useful when you want to stage new and modified files in the current directory, while git add -A is more comprehensive and stages all changes, including deletions, across the entire working directory[[2]](https://stackoverflow.com/questions/572549/difference-between-git-add-a-and-git-add)[[3]](https://superuser.com/questions/1490325/is-there-a-difference-between-git-add-a-and-git-add-a)[[4]](https://betterstack.com/community/questions/difference-git-add-a-git-add-dot/).

If you have any more questions or need further clarification, feel free to ask!

**References**

[1] [GitCommands](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/__github/GitCommands.txt?web=1)

[2] [Difference between "git add -A" and "git add - Stack Overflow](https://stackoverflow.com/questions/572549/difference-between-git-add-a-and-git-add)

[3] [Is there a difference between git add -a and git add -A?](https://superuser.com/questions/1490325/is-there-a-difference-between-git-add-a-and-git-add-a)

[4] [Difference between “Git Add -A” and “Git Add - Better Stack](https://betterstack.com/community/questions/difference-git-add-a-git-add-dot/)

You need to use git fetch when you want to update your local repository with changes from a remote repository without merging those changes into your working directory. This command is particularly useful in collaborative workflows where multiple developers are working on the same project.

**From your company**

In the [**Commonly used Git command line commands**](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/__github/Commonly%20used%20Git%20command%20line%20commands.docx?web=1&EntityRepresentationId=01b0da54-4aeb-4723-83a8-7ac3a27ac1e2) document authored by [Stephan Onisick](https://www.office.com/search?q=Stephan+Onisick&EntityRepresentationId=695c8888-76c0-4ffe-8423-851fa917af5d), it is explained that git fetch downloads commits, files, and refs from a remote repository into your local repo. This command is critical for collaborative Git workflows as it updates your local repository to the state of the remote repository[[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/__github/Commonly%20used%20Git%20command%20line%20commands.docx?web=1).

**From the Web**

The git fetch command connects to the remote repository and retrieves any changes that have occurred since your last fetch or clone. These changes go to remote-tracking branches in your local repository; they do not affect your working directory or currently checked-out branch[[2]](https://gitscripts.com/fetch-git-command). This makes it an excellent way to see what others have been working on without altering your current work[[3]](https://www.atlassian.com/git/tutorials/syncing/git-fetch).

Here are some scenarios where you would use git fetch:

* **Collaborative Projects**: When working on a project with multiple developers, you should periodically fetch changes to keep your local repository up-to-date with the latest changes made by others[[4]](https://stackoverflow.com/questions/74505323/when-is-a-good-time-to-use-git-fetch).
* **Reviewing Changes**: Before merging changes from a remote repository into your local branch, you can fetch the changes to review them and ensure they do not introduce any issues[[3]](https://www.atlassian.com/git/tutorials/syncing/git-fetch).
* **Resolving Conflicts**: If there are conflicts between your local changes and the remote repository, fetching the changes allows you to see the conflicts and resolve them before merging[[2]](https://gitscripts.com/fetch-git-command).

Would you like more information on any specific aspect of using git fetch?[[4]](https://stackoverflow.com/questions/74505323/when-is-a-good-time-to-use-git-fetch)[[3]](https://www.atlassian.com/git/tutorials/syncing/git-fetch)[[2]](https://gitscripts.com/fetch-git-command)

**References**

[1] [Commonly used Git command line commands](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/__github/Commonly%20used%20Git%20command%20line%20commands.docx?web=1)

[2] [Fetch Git Command: Mastering the Basics Quickly](https://gitscripts.com/fetch-git-command)

[3] [How git fetch works with remote branches - Atlassian](https://www.atlassian.com/git/tutorials/syncing/git-fetch)

[4] [github - When is a good time to use Git fetch? - Stack Overflow](https://stackoverflow.com/questions/74505323/when-is-a-good-time-to-use-git-fetch)

The difference between git fetch and git pull lies in how they interact with your local and remote repositories:

**From your company**

In the [**Commonly used Git command line commands**](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/__github/Commonly%20used%20Git%20command%20line%20commands.docx?web=1&EntityRepresentationId=59a6b01b-b834-44dc-9301-7a5291a84f74) document authored by [Stephan Onisick](https://www.office.com/search?q=Stephan+Onisick&EntityRepresentationId=72539cfe-0936-4640-80a3-090f482307e6), it is explained that git fetch downloads commits, files, and refs from a remote repository into your local repo. This command is critical for collaborative Git workflows as it updates your local repository to the state of the remote repository[[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/__github/Commonly%20used%20Git%20command%20line%20commands.docx?web=1).

**From the Web**

1. **git fetch**: This command downloads changes from the remote repository to your local repository without modifying your working directory. It updates your remote-tracking branches under refs/remotes/<remote>/. This operation is safe to run at any time since it never changes any of your local branches under refs/heads[[2]](https://stackoverflow.com/questions/292357/what-is-the-difference-between-git-pull-and-git-fetch)[[3]](https://about.gitlab.com/blog/2024/09/24/git-pull-vs-git-fetch-whats-the-difference/)[[4]](https://www.git-tower.com/learn/git/faq/difference-between-git-fetch-git-pull/).
2. **git pull**: This command performs a git fetch followed by a git merge or git rebase. It brings your local branch up-to-date with its remote version while also updating your other remote-tracking branches. Essentially, git pull integrates the changes from the remote repository into your current branch[[2]](https://stackoverflow.com/questions/292357/what-is-the-difference-between-git-pull-and-git-fetch)[[3]](https://about.gitlab.com/blog/2024/09/24/git-pull-vs-git-fetch-whats-the-difference/)[[4]](https://www.git-tower.com/learn/git/faq/difference-between-git-fetch-git-pull/).

In summary, git fetch is used to update your local repository with changes from the remote repository without affecting your working directory, while git pull fetches the changes and then merges them into your current branch, updating your working directory.

If you have any more questions or need further clarification, feel free to ask![[1]](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/__github/Commonly%20used%20Git%20command%20line%20commands.docx?web=1)[[2]](https://stackoverflow.com/questions/292357/what-is-the-difference-between-git-pull-and-git-fetch)[[3]](https://about.gitlab.com/blog/2024/09/24/git-pull-vs-git-fetch-whats-the-difference/)[[4]](https://www.git-tower.com/learn/git/faq/difference-between-git-fetch-git-pull/)

**References**

[1] [Commonly used Git command line commands](https://onisick-my.sharepoint.com/personal/stephan_onisick_net/Documents/__github/Commonly%20used%20Git%20command%20line%20commands.docx?web=1)

[2] [What is the difference between 'git pull' and 'git fetch'?](https://stackoverflow.com/questions/292357/what-is-the-difference-between-git-pull-and-git-fetch)

[3] [Git pull vs. git fetch: What's the difference? - GitLab](https://about.gitlab.com/blog/2024/09/24/git-pull-vs-git-fetch-whats-the-difference/)

[4] [What's the difference between "git fetch" and "git pull"?](https://www.git-tower.com/learn/git/faq/difference-between-git-fetch-git-pull/)

Sure! Here's a scenario where you create a new branch, add some changes, commit those changes, and then merge the branch back into the main branch:

1. **Create a New Branch**:

git checkout -b feature-branch

This command creates a new branch named feature-branch and switches to it.

1. **Add Some Changes**: Make changes to your files in the working directory. For example, you might add a new file or modify an existing one.
2. **Stage the Changes**:

git add .

This command stages all the changes in the current directory and its subdirectories.

1. **Commit the Changes**:

git commit -m "Add new feature"

This command commits the staged changes with a descriptive message.

1. **Switch Back to the Main Branch**:

git checkout main

This command switches back to the main branch.

1. **Merge the Feature Branch into the Main Branch**:

git merge feature-branch

This command merges the changes from feature-branch into the main branch.

1. **Push the Changes to the Remote Repository**:

git push origin main

This command pushes the changes in the main branch to the remote repository.

By following these steps, you can create a new branch, make changes, commit those changes, and merge the branch back into the main branch. This workflow is commonly used in collaborative projects to keep the main branch clean and stable while developing new features or making changes in separate branches.

If you have any more questions or need further clarification, feel free to ask!

To open Git manual pages in the browser from the command line on Windows, you can use the git help command followed by the specific Git command you want to learn about. This will open the HTML version of the manual page in your default web browser. Here’s how you can do it:

1. Open your command line interface (CLI) on Windows.
2. Type the following command:

git help <command>

Replace <command> with the Git command you want to view the manual page for. For example, if you want to view the manual page for git commit, you would type:

git help commit

This command will open the HTML manual page for git commit in your default web browser[[1]](https://git-scm.com/docs/git-help).

Alternatively, you can use the --help option with the Git command:

git <command> --help

This will also open the HTML manual page in your browser[[1]](https://git-scm.com/docs/git-help).

If you prefer to view the help directly in the command line, you can use the -h option, which provides a shorter usage summary:

git <command> -h

However, this will not open the full manual page in the browser[[2]](https://stackoverflow.com/questions/5517564/how-do-i-get-git-to-show-command-line-help-in-windows).

Would you like more information on any specific Git commands or further assistance with Git?

**References**

[1] [Git - git-help Documentation](https://git-scm.com/docs/git-help)

[2] [how do i get git to show command-line help in windows?](https://stackoverflow.com/questions/5517564/how-do-i-get-git-to-show-command-line-help-in-windows)