ATLASSIAN

Git pull

The `git pull` command is used to fetch and download content from a remote repository and immediately update the local repository to match that content. Merging remote upstream changes into your local repository is a common task in Git-based collaboration work flows. The `git pull` command is actually a combination of two other commands, git fetch followed by git merge. In the first stage of operation `git pull` will execute a `git fetch` scoped to the local branch that `HEAD` is pointed at. Once the content is downloaded, `git pull` will enter a merge workflow. A new merge commit will be-created and `HEAD` updated to point at the new commit.

Git pull usage

How it works

The `git pull` command first runs `git fetch` which downloads content from the specified remote repository.

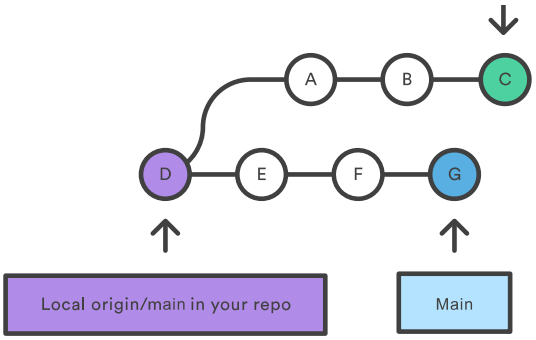
Then a `git merge` is executed to merge the remote content refs and heads into a new local merge commit. To better demonstrate the pull and merging process let us consider the following example. Assume we have a repository with a main branch and a remote origin.

RELATED MATERIAL

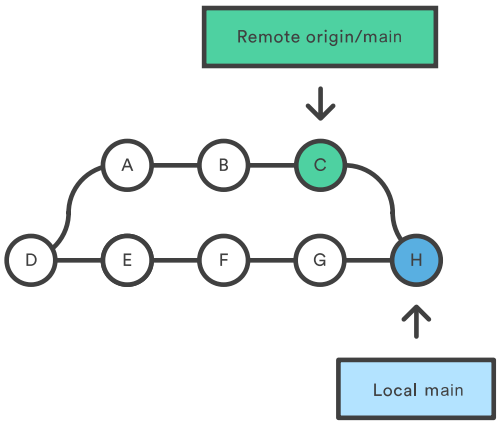
Advanced Git log

Read article →

ATLASSIAN

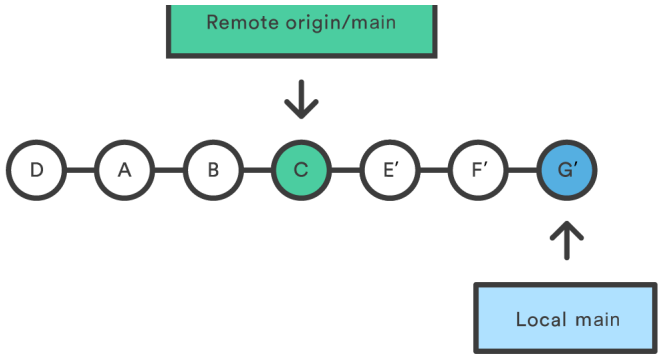


In this scenario, `git pull` will download all the changes from the point where the local and main diverged. In this example, that point is E. `git pull` will fetch the diverged remote commits which are A-B-C. The pull process will then create a new local merge commit containing the content of the new diverged remote commits.



In the above diagram, we can see the new commit H. This commit is a new merge commit that contains the contents of remote A-B-C commits and has a combined log message. This example is one of a few `git pull` merging strategies. A `--rebase` option can be passed to `git pull` to use a rebase merging strategy instead of a merge commit. The next example will demonstrate how a rebase pull works. Assume that we are at a starting point of our first diagram, and we have executed `git pull --rebase`.

ATLASSIAN



In this diagram, we can now see that a rebase pull does not create the new H commit. Instead, the rebase has copied the remote commits A--B--C and rewritten the local commits E--F--G to appear after them them in the local origin/main commit history.

Common Options

```
git pull <remote>
```

Fetch the specified remote's copy of the current branch and immediately merge it into the local copy. This is the same as `git fetch <remote>` followed by `git merge origin/<current-branch>`.

```
git pull --no-commit <remote>
```

Similar to the default invocation, fetches the remote content but does not create a new merge commit.

```
git pull --rebase <remote>
```

ATLASSIAN

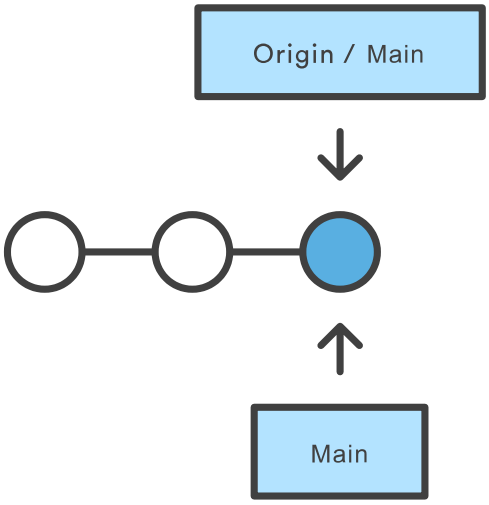
```
git pull --verbose
```

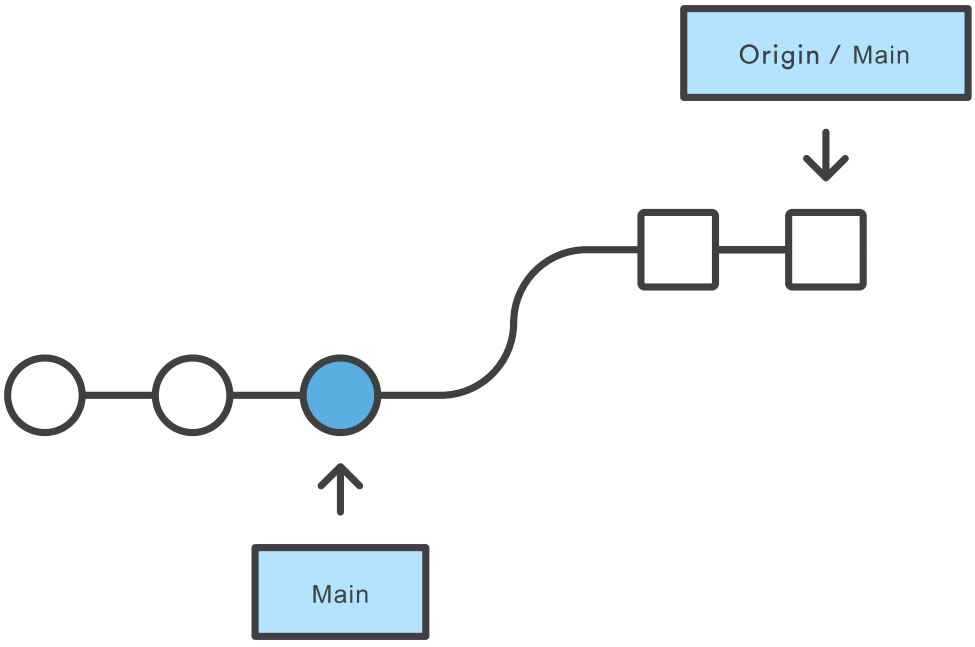
Gives verbose output during a pull which displays the content being downloaded and the merge details.

Git pull discussion

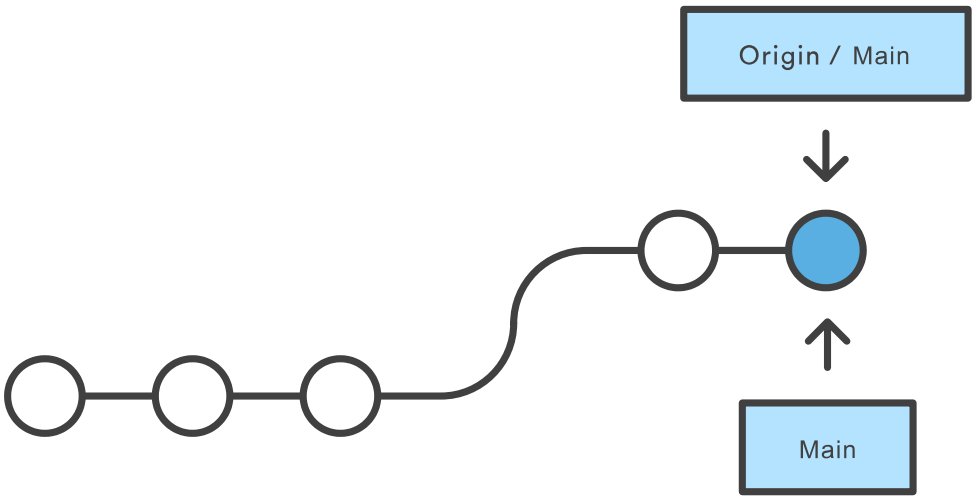
You can think of `git pull` as Git's version of `svn update`. It's an easy way to synchronize your local repository with upstream changes. The following diagram explains each step of the pulling process.

ATLASSIAN





ATLASSIAN



You start out thinking your repository is synchronized, but then `git fetch` reveals that

ATLASSIAN

Git pull and syncing

`git pull` is one of many commands that claim the responsibility of 'syncing' remote content. The `git remote` command is used to specify what remote endpoints the syncing commands will operate on. The `git push` command is used to upload content to a remote repository.

The `git fetch` command can be confused with `git pull`. They are both used to download remote content. An important safety distinction can be made between `git pull` and `get fetch`. `git fetch` can be considered the "safe" option whereas, `git pull` can be considered unsafe. `git fetch` will download the remote content and not alter the state of the local repository. Alternatively, `git pull` will download remote content and immediately attempt to change the local state to match that content. This may unintentionally cause the local repository to get in a conflicted state.

Pulling via Rebase

The `--rebase` option can be used to ensure a linear history by preventing unnecessary merge commits. Many developers prefer rebasing over merging, since it's like saying, "I want to put my changes on top of what everybody else has done." In this sense, using `git pull` with the `--rebase` flag is even more like `svn update` than a plain `git pull`.

In fact, pulling with `--rebase` is such a common workflow that there is a dedicated configuration option for it:

```
git config --global branch.autosetuprebase always
```

After running that command, all `git pull` commands will integrate via `git rebase` instead of `git merge`.

ATLASSIAN

The following examples demonstrate how to use `git pull` in common scenarios:

Default Behavior

```
git pull
```

Executing the default invocation of `git pull` will is equivalent to `git fetch origin HEAD` and `git merge HEAD` where `HEAD` is ref pointing to the current branch.

Git pull on remotes

```
git checkout new_feature
git pull <remote repo>
```

This example first performs a checkout and switches to the branch. Following that, the `git pull` is executed with being passed. This will implicitly pull down the newfeature branch from . Once the download is complete it will initiate a `git merge` .

Git pull rebase instead of merge

The following example demonstrates how to synchronize with the central repository's main branch using a rebase:

```
git checkout main
git pull --rebase origin
```

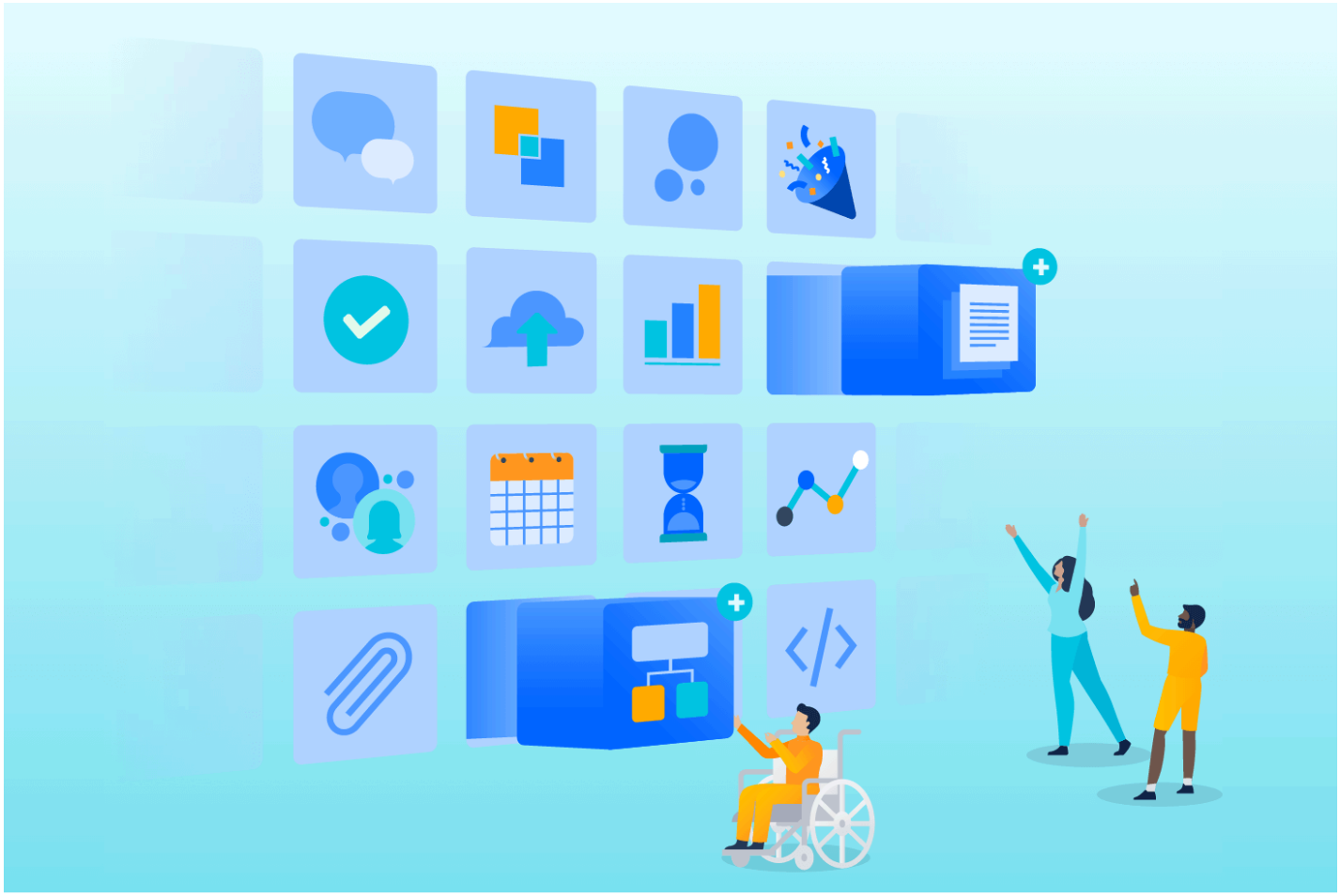
This simply moves your local changes onto the top of what everybody else has already contributed.

ATLASSIAN

Making a Pull Request →

Recommended reading

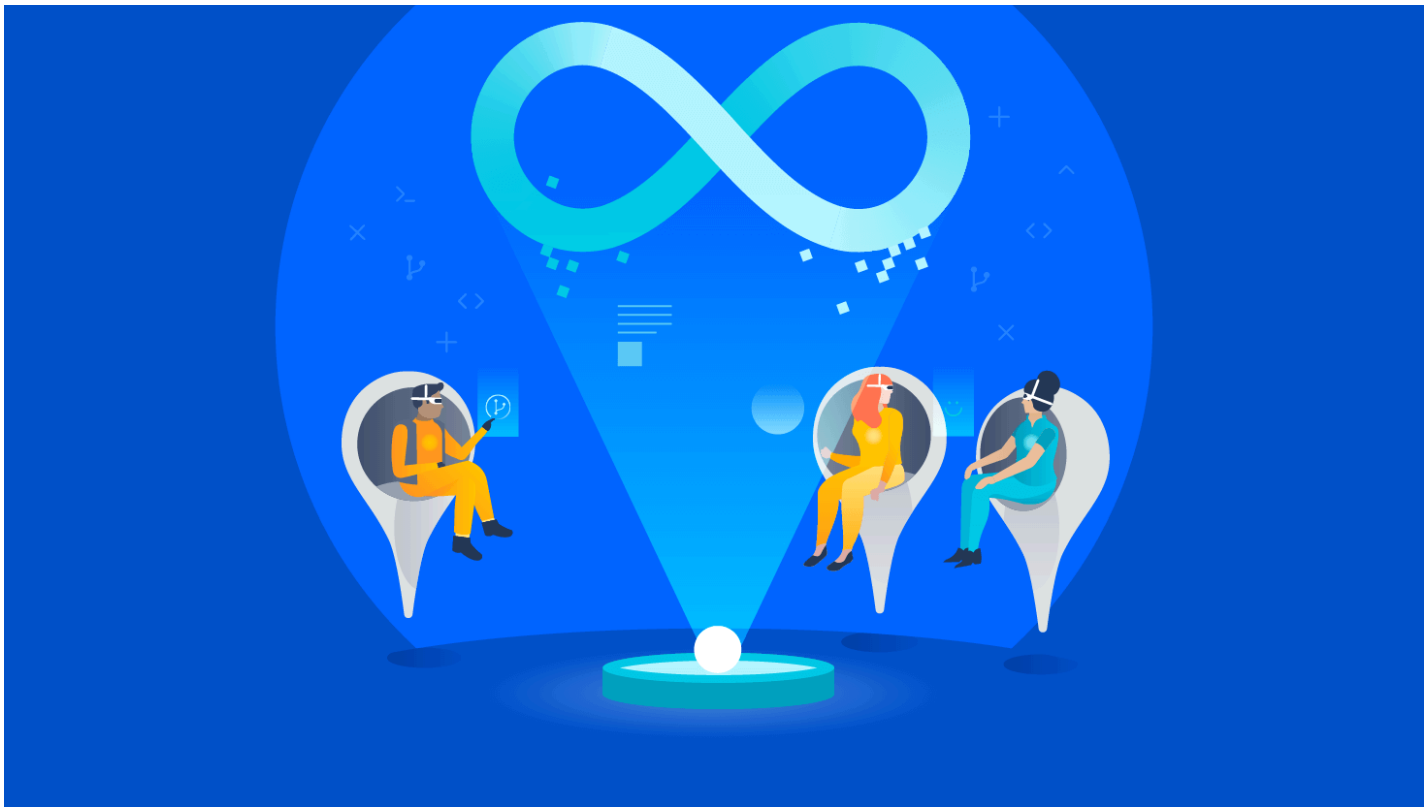
Bookmark these resources to learn about types of DevOps teams, or for ongoing updates about DevOps at Atlassian.



Bitbucket blog

Learn more →

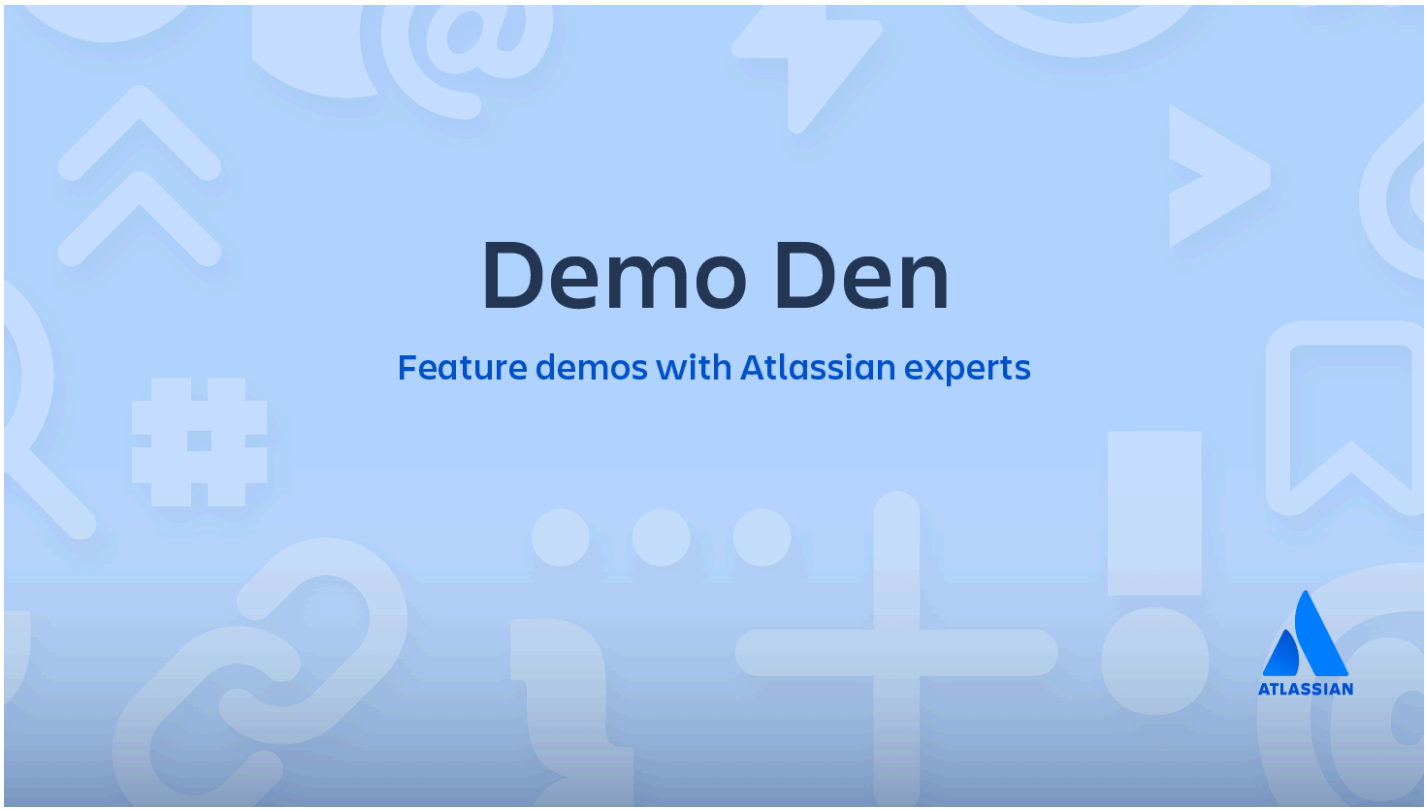
ATLASSIAN



DevOps learning path

Learn more →

ATLASSIAN



How Bitbucket Cloud works with Atlassian Open DevOps

Watch now

Sign up for our DevOps newsletter

Email address

Sign up

ATLASSIAN

Events

Blogs

Investor Relations

Atlassian Foundation

Contact us

PRODUCTS

Rovo

Jira

Jira Align

Jira Service Management

Confluence

Trello

Bitbucket

See all products →

RESOURCES

Technical support

Purchasing & licensing

Atlassian Community

ATLASSIAN

My account

Create support ticket →

LEARN

Partners

Training & certification

Documentation

Developer resources

Enterprise services

See all resources →

Privacy Policy Terms Impressum English ▼