

Git remote

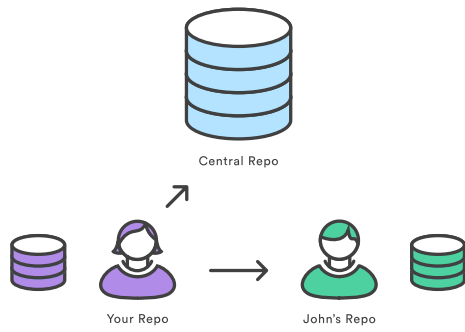
SVN uses a single centralized repository to serve as the communication hub for developers, and collaboration takes place by passing changesets between the developers' working copies and the central repository. This is different from **Git's distributed collaboration model, which gives every developer their own copy of the repository, complete with its own local history and branch structure.** Users typically need to share a series of commits rather than a single changeset. Instead of committing a changeset from a working copy to the central repository, Git lets you share entire branches between repositories.

The `git remote` command is one piece of the broader system which is responsible for syncing changes. Records registered through the `git remote` command are used in conjunction with the `git fetch`, `git push`, and `git pull` commands. These commands all have their own **syncing responsibilities** which can be explored on the corresponding links.

Git remote

The `git remote` command lets you **create, view, and delete connections to other repositories.** Remote connections are more like **bookmarks rather than direct links into other repositories.** Instead of providing real-time access to another repository, **they serve as convenient names that can be used to reference a not-so-convenient URL.**

For example, the following diagram shows two remote connections from your repo into the central repo and another developer's repo. Instead of referencing them by their full URLs, you can pass the origin and john shortcuts to other Git commands.



Git remote usage overview

The `git remote` command is essentially an interface for managing a list of remote entries that are stored in the repository's `./.git/config` file. The following commands are used to view the current state of the remote list.

Viewing git remote configurations

```
git remote
```

List the remote connections you have to other repositories.

```
git remote -v
```

Same as the above command, but include the URL of each connection.

Creating and modifying git remote configurations

The `git remote` command is also a convenience or 'helper' method for modifying a repo's `./.git/config` file. The commands presented below let you manage connections with other repositories. The following commands will modify the repo's `./.git/config` file. The result of the following commands can also be achieved by directly editing the `./.git/config` file with a text editor.

```
git remote add <name> <url>
```

Create a new connection to a remote repository. After adding a remote, you'll be able to use `<name>` as a convenient shortcut for `<url>` in other Git commands.

```
git remote rm <name>
```

Remove the connection to the remote repository called `<name>` .

```
git remote rename <old-name> <new-name>
```

Rename a remote connection from `<old-name>` to `<new-name>` .

Git remote discussion

Git is designed to give each developer an entirely isolated development environment. This means that information is not automatically passed back and forth between repositories. Instead, developers need to manually pull upstream commits into their local repository or manually push their local commits back up to the central repository. The `git remote` command is really just an easier way to pass URLs to these "sharing" commands.

The origin Remote

When you clone a repository with `git clone`, it automatically creates a remote connection called `origin` pointing back to the cloned repository. This is useful for developers creating a local copy of a central repository, since it provides an easy way to pull upstream changes or publish local commits. This behavior is also why most Git-based projects call their central repository `origin`.

Repository URLs

Git supports many ways to **reference a remote repository**. Two of the easiest ways to access a remote repo are via the HTTP and the *SSH(Secure Shell)* protocols. **HTTP is an easy way to allow anonymous, read-only access to a repository**. For example:

```
http://host/path/to/repo.git
```

But, it's generally not possible to push commits to an HTTP address (you wouldn't want to allow anonymous pushes anyways). **For read-write access, you should use SSH instead:**

```
ssh://user@host/path/to/repo.git
```

You'll need a valid SSH account on the host machine, but other than that, Git supports authenticated access via SSH out of the box. Modern secure 3rd party hosting solutions like Bitbucket.com will provide these URLs for you.

Git remote commands

The `git remote` command is one of many Git commands that takes additional appended 'subcommands'. Below is an examination of the commonly used `git remote` subcommands.

```
ADD <NAME> <URL>
```

Adds a record to `./.git/config` for remote named `<name>` at the repository url `<url>`.

Accepts a `-f` option, that will `git fetch` immediately after the remote record is created.

Accepts a `--tags` option, that will `git fetch` immediately and import every tag from the remote repository.

```
RENAME <OLD> <NEW>
```

Updates `./ .git/config` to rename the record `<OLD>` to `<NEW>` . All remote-tracking branches and configuration settings for the remote are updated.

```
RENAME <NAME> or RM <NAME>
```

Modifies `./ .git/config` and removes the remote named `<NAME>` . All remote-tracking branches and configuration settings for the remote are removed.

```
GET-URL <NAME>
```

Outputs the URLs for a remote record.

Accepts `--push` , push URLs are queried rather than fetch URLs.

With `--all` , all URLs for the remote will be listed.

```
SHOW <NAME>
```

Outputs high-level information about the remote `<NAME>` .

```
PRUNE <NAME>
```

Deletes any local branches for `<NAME>` that are not present on the remote repository.

Accepts a `--dry-run` option which will list what branches are set to be pruned, but will not actually prune them.

Git remote examples

In addition to origin, it's often convenient to have a connection to your teammates' repositories. For example, if your co-worker, John, maintained a publicly accessible repository on `dev.example.com/john.git` , you could add a connection as follows:

```
git remote add john http://dev.example.com/john.git
```

Having this kind of access to individual developers' repositories makes it possible to collaborate outside of the central repository. This can be very useful for small teams working on a large project.

Showing your remotes

By default, the `git remote` command will list previously stored remote connections to other repositories. This will produce single line output that lists the names of "bookmark" name of remote repos.

```
$ git remote
origin
upstream
other_users_repo
```

Invoking `git remote` with the `-v` option will print the list of bookmarked repository names and additionally, the corresponding repository URL. The `-v` option stands for "verbose". Below is example output of verbose `git remote` output.

```
git remote -v
origin  git@bitbucket.com:origin_user/reponame.git (fetch)
origin  git@bitbucket.com:origin_user/reponame.git (push)
upstream    https://bitbucket.com/upstream_user/reponame
upstream    https://bitbucket.com/upstream_user/reponame
other_users_repo  https://bitbucket.com/other_users_re
other_users_repo  https://bitbucket.com/other_users_re
```

Adding remote repositories

The `git remote add` command will create a new connection record to a remote repository. After adding a remote, you'll be able to use as a

convenient shortcut for in other Git commands. For more information on the accepted URL syntax, view the "Repository URLs" section below. This command will create a new record within the repository's `./.git/config`. An example of this config file update follows:

```
$ git remote add fake_test https://bitbucket.com/upstream
url = https://bitbucket.com/upstream_user/reponame.git
fetch = +refs/heads/*:refs/remotes/remote_test/*
```

Inspecting a Remote

The `show` subcommand can be appended to `git remote` to give detailed output on the **configuration of a remote**. This output will contain a list of branches associated with the remote and also the endpoints attached for fetching and pushing.

```
git remote show upstream
* remote upstream
Fetch URL: https://bitbucket.com/upstream_user/reponame.git
Push URL: https://bitbucket.com/upstream_user/reponame.git
HEAD branch: main
Remote branches:
  main tracked
  simd-deprecated tracked
  tutorial tracked
Local ref configured for 'git push':
  main pushes to main (fast-forwardable)
```

Fetching and pulling from Git remotes

Once a remote record has been configured through the use of the `git remote` command, the remote name can be passed as an argument to other Git commands to communicate with the remote repo. Both `git fetch`, and `git pull` can be used to read from a remote repository.

Both commands have different operations that are explained in further depth on their respective links.

Pushing to Git remotes

The `git push` command is used to write to a remote repository.

```
git push <remote-name> <branch-name>
```

This example will upload the local state of `<branch-name>` to the remote repository specified by `<remote-name>`.

Renaming and removing remotes

```
git remote rename <old-name> <new-name>
```

The command `git remote rename` is self-explanatory. When executed, this command will rename a remote connection from `<old-name>` to `<new-name>`. Additionally, this will modify the contents of `./ .git/config` to rename the record for the remote there as well.

```
git remote rm <name>
```

The command `git remote rm` will remove the connection to the remote repository specified by the `<name>` parameter. To demonstrate let us 'undo' the remote addition from our last example. If we execute `git remote rm remote_test`, and then examine the contents of `./ .git/config` we can see that the `[remote "remote_test"]` record is no longer there.