Here's the long answer.

Remotes:

If you're using Git collaboratively, you'll probably need to sync your commits with other machines or locations. Each machine or location is called a **remote**, in Git's terminology, and each one may have one or more branches. Most often, you'll just have one, named `origin`. To list all the remotes, run `git remote`:

```
$ git remote
bitbucket
origin
```

You can see which locations these remote names are shortcuts for, by running `git remote -v`:

```
$ git remote -v
bitbucket git@bitbucket.org:flimm/example.git (fetch)
bitbucket git@bitbucket.org:flimm/example.git (push)
origin git@github.com:Flimm/example.git (fetch)
origin git@github.com:Flimm/example.git (push)
```

Each remote has a directory under `git/refs/remotes/`:

```
$ ls -F .git/refs/remotes/
bitbucket/ origin/
```

Branches on your machine:

TLDR: on your local machine, you've got three types of branches: local non-tracking branches, local tracking branches, and remote-tracking branches. On a remote machine, you've just got one type of branch.

1. Local branches

You can view a list of all the local branches on your machine by running `git branch`:

```
$ git branch
master
new-feature
```

Each local branch has a file under `.git/refs/heads/`:

```
$ ls -F .git/refs/heads/
```

```
master new-feature
```

There are two types of local branches on your machine: non-tracking local branches, and tracking local branches.

1.1 Non-tracking local branches

Non-tracking local branches are not associated with any other branch. You create one by running `git branch <branchname>` .

1.2. Tracking local branches

Tracking local branches are associated with another branch, usually a remote-tracking branch. You create one by running `git branch --track <branchname> [<start-point>]` .

You can view which one of your local branches are tracking branches using `git branch -vv` :

```
$ git branch -vv
master      b31f87c85 [origin/master] Example commit message
new-feature b760e04ed Another example commit message
```

From this command's output, you can see that the local branch `master` is tracking the remote-tracking branch `origin/master` , and the local branch `new-feature` is not tracking anything.

Another way to see which branches are tracking branches is by having a look at `.git/config` .

Tracking local branches are useful. They allow you to run `git pull` and `git push` , without specifying which upstream branch to use. If the branch is not set up to track another branch, you'll get an error like this one:

```
$ git checkout new-feature
$ git pull
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details

    git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream new-feature <remote>/<branch>
```

2. Remote-tracking branches (still on your machine)

You can view a list of all the remote-tracking branches on your machine by running `git branch -r`:

```
$ git branch -r
bitbucket/master
origin/master
origin/new-branch
```

Each remote-tracking branch has a file under `.git/refs/<remote>/`:

```
$ tree -F .git/refs/remotes/
.git/refs/remotes/
├── bitbucket/
│   └── master
└── origin/
    ├── master
    └── new-branch
```

Think of your remote-tracking branches as your local cache for what the remote machines contain. You can update your remote-tracking branches using `git fetch`, which `git pull` uses behind the scenes.

Even though all the data for a remote-tracking branch is stored locally on your machine (like a cache), it's still never called a local branch. (At least, I wouldn't call it that!) It's just called a remote-tracking branch.

Branches on a remote machine:

You can view all the remote branches (that is, the branches on the remote machine), by running `git remote show <remote>`:

```
$ git remote show origin
* remote origin
  Fetch URL: git@github.com:Flimm/example.git
  Push  URL: git@github.com:Flimm/example.git
  HEAD branch: master
  Remote branches:
    io-socket-ip            new (next fetch will store in remotes/origin)
    master                  tracked
    new-branch              tracked
  Local ref configured for 'git pull':
    master     merges with remote master
    new-branch merges with remote new-branch
  Local ref configured for 'git push':
    master     pushes to master     (up to date)
    new-branch pushes to new-branch (fast-forwardable)
```

This `git remote` command queries the remote machine over the network about its branches. It does not update the remote-tracking branches on your local machine, use `git fetch` or `git pull` for that.

From the output, you can see all the branches that exist on the remote machine by looking under the heading "Remote branches" (ignore lines marked as "stale").

If you could log in to the remote machine and find the repository in the filesystem, you could have a look at all its branches under `refs/heads/` .

Cheat sheet:

- To delete a local branch, whether tracking or non-tracking, safely:

  ```
  git branch -d <branchname>
  ```

- To delete a local branch, whether tracking or non-tracking, forcefully:

  ```
  git branch -D <branchname>
  ```

- To delete a remote-tracking branch:

  ```
  git branch -rd <remote>/<branchname>
  ```

- To create a new local non-tracking branch:

  ```
  git branch <branchname> [<start-point>]
  ```

- To create a new local tracking branch: (Note that if `<start-point>` is specified and is a remote-tracking branch like `origin/foobar` , then the `--track` flag is automatically included)

  ```
  git branch --track <branchname> [<start-point]
  ```

  Example:

  ```
  git branch --track hello-kitty origin/hello-kitty
  ```

- To delete a branch on a remote machine:

  ```
  git push --delete <remote> <branchname>
  ```

- To delete all remote-tracking branches that are stale, that is, where the corresponding branches on the remote machine no longer exist:

  ```
  git remote prune <remote>
  ```

You may have noticed that in some commands, you use `<remote>/<branch>` , and other commands, `<remote> <branch>` . Examples: `git branch origin/hello-kitty` and `git push --delete origin hello-kitty` .

It may seem arbitrary, but there is a simple way to remember when to use a slash and when to use a space. When you're using a slash, you're referring to a remote-tracking branch on your own machine, whereas when you're using a space, you're actually dealing with a branch on a remote machine over the network.