

# Remotes

---



# Working with a Remote Repo

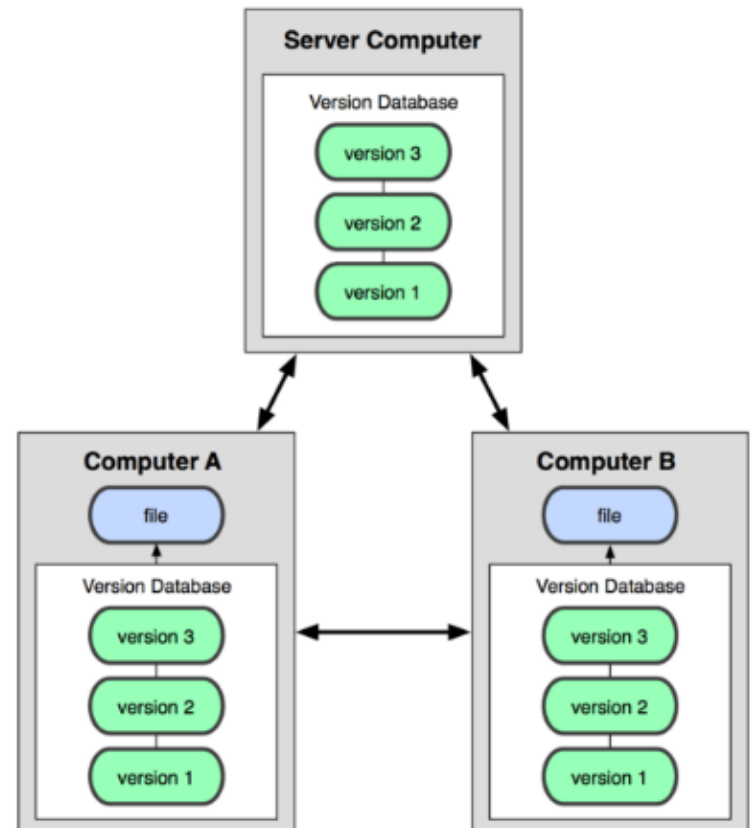
Git is a distributed version control system.

*git was invented to manage the Linux kernel source code, with thousands of developers in over a hundred countries.*

You can have **many** repositories on the net, called "**remotes**".

They may all be different!

**No "master" repository**  
-- all repos are equal.



# Git Hosting Sites

---

You can create free git repositories on these sites, for individual or team projects.

**Github** - <https://github.com>

**Bitbucket** - <https://bitbucket.org>

**GitLab** - <https://gitlab.com>

# Commands for Remotes

---

Common commands for using a remote repo are:

`git clone` copy remote repo to your computer

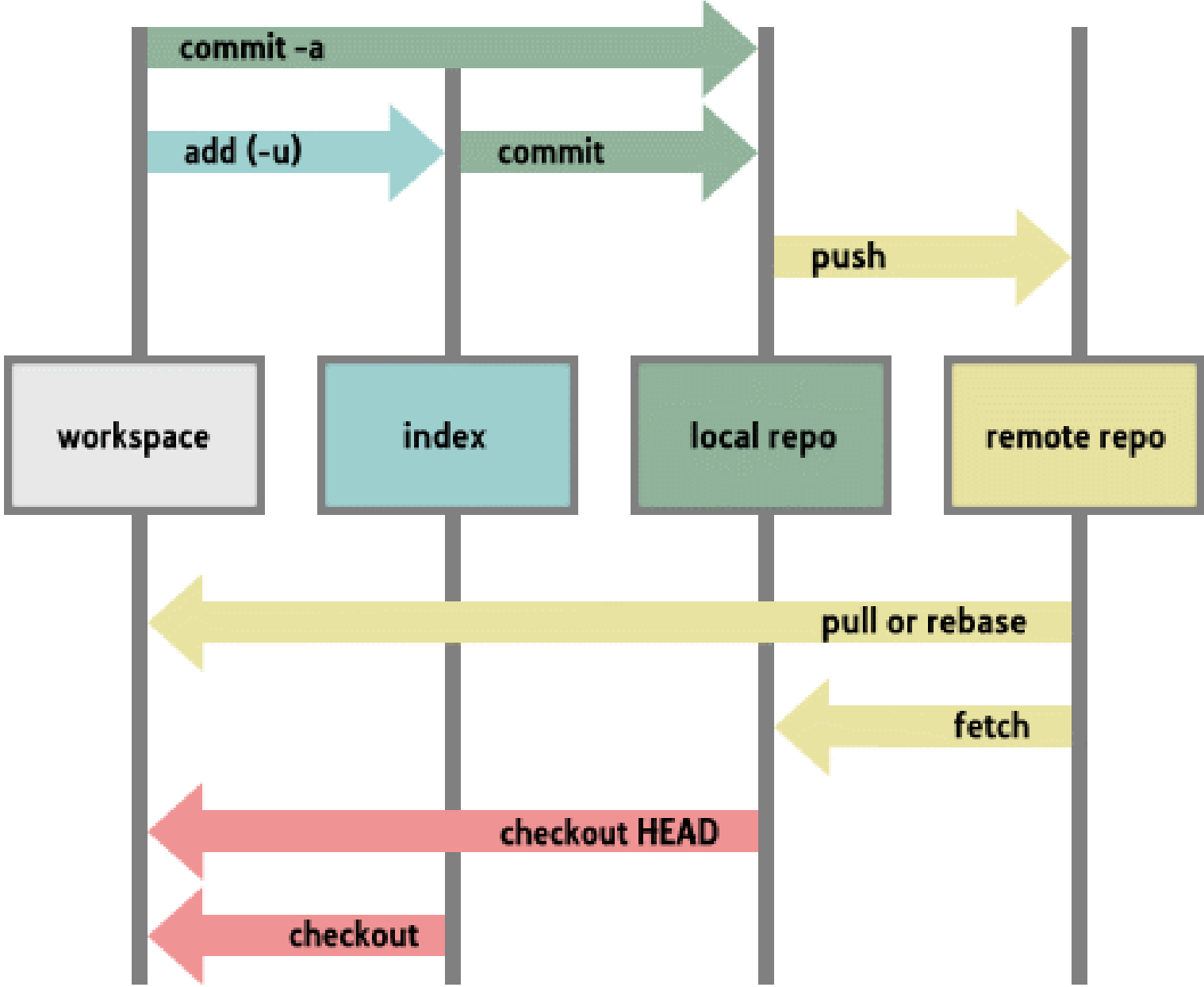
`git remote add` define URL of a remote repository

`git remote -v` list remotes, with URLs

`git push` "push" local updates to a remote

`git pull` download and merge remote updates

`git fetch` download remote updates, but don't merge into your working copy



# Cloning a Remote Repository

---

Create a local copy of a Github repo. Assign the name "origin" to the remote repo.

```
cmd> git clone  
      https://github.com/user/tictactoe
```

Cloning into "tictactoe" ...

(git creates a "tictactoe" directory for file)

Or, use the **SSH protocol** (requires SSH key)

```
cmd> git clone  
      git@github.com:user/tictactoe
```

# Viewing Your Remotes

---

View the names and URLs of "remotes" for a repo:

```
cmd> cd tictactoe
```

```
cmd> git remote -v
```

```
origin https://github.com/user/tictactoe (fetch)
```

```
origin https://github.com/user/tictactoe (pull)
```

# You Can Have Several Remotes

---

Each remote has a different name (`origin`, `bbucket`).

The username and repo name can be different, too.

```
cmd> cd tictactoe
```

```
cmd> git remote -v
```

```
origin https://github.com/barz/tictactoe (fetch)
```

```
origin https://github.com/barz/tictactoe (pull)
```

```
bbucket git@bitbucket.com:fooz/ttt (fetch)
```

```
bbucket git@bitbucket.com:fooz/ttt (pull)
```



# Syntax for all "remote" commands

---

```
git remote
```

```
git help remote
```

```
git remote -v
```

```
git remote add remote_name URL
```

```
git remote show remote_name
```

```
git remote set-url remote_name new_url
```

# Change the remote location

---

You make a copy of "git-commands" in your own Github account. Now change URL of the remote to "push" do:

```
cmd> git remote -v
```

```
origin https://github.com/ISP2020/git-commands-fatalai
```

```
cmd> git remote set-url origin  
      https://github.com/fatalai/git-commands
```

**This does not move the repository on Github!**

You must do that on Github (in Settings).

# Detailed Info about a Remote

---

```
cmd> git remote show origin
```

```
* remote origin
```

```
Fetch URL: https://github.com/fatalai/git-commands
```

```
Push URL: https://github.com/fatalai/git-commands
```

```
HEAD branch: master
```

```
Remote branches:
```

```
master                tracked
```

```
dev-branch            tracked
```

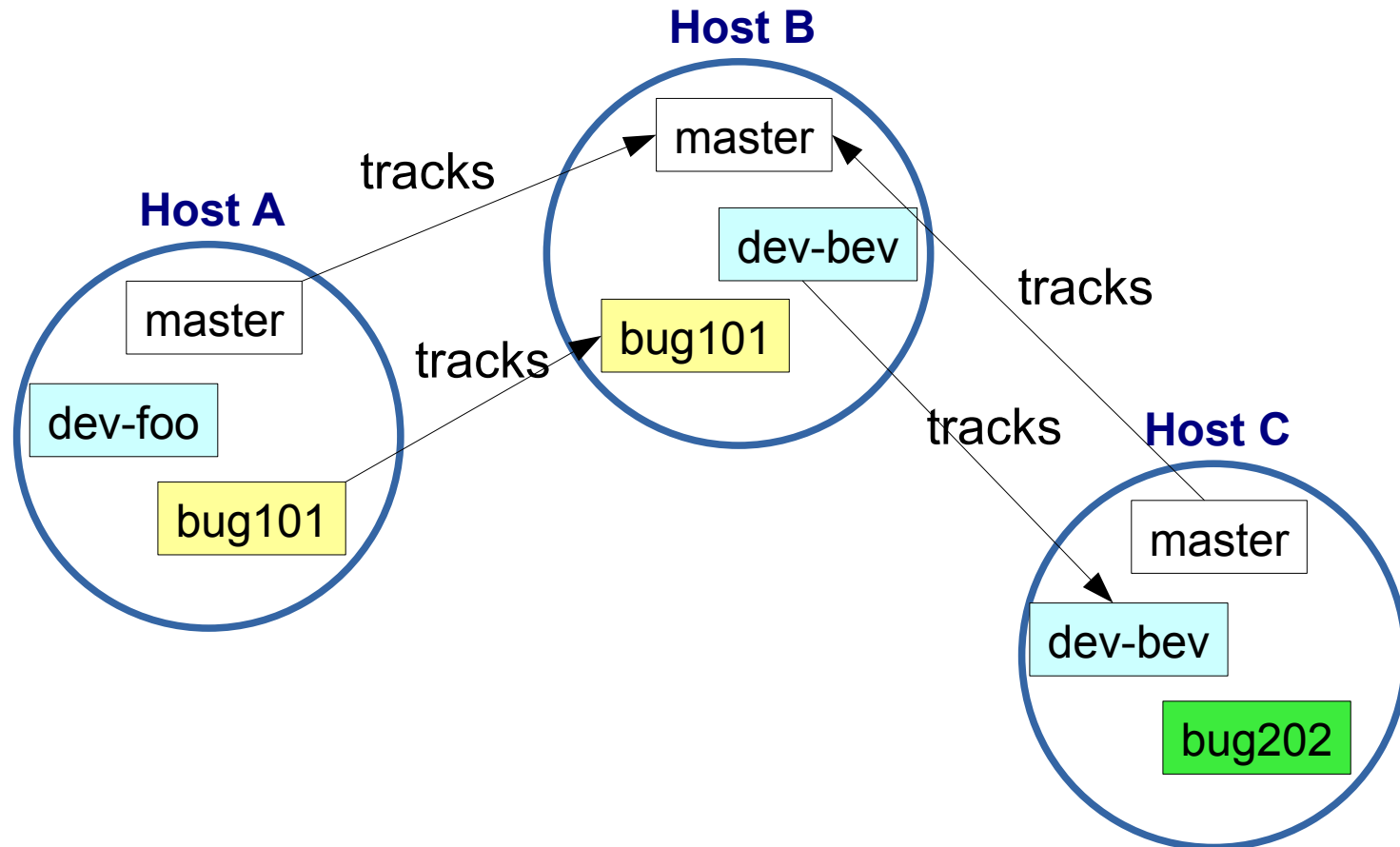
```
Local branch configured for 'git pull':
```

```
master merges with remote master
```

# Remote Branches

Branches are not automatically synced between remotes.

A local branch may not have any remote branch.



# Remote Branch Naming

---

On your machine, refer to a remote branch as:

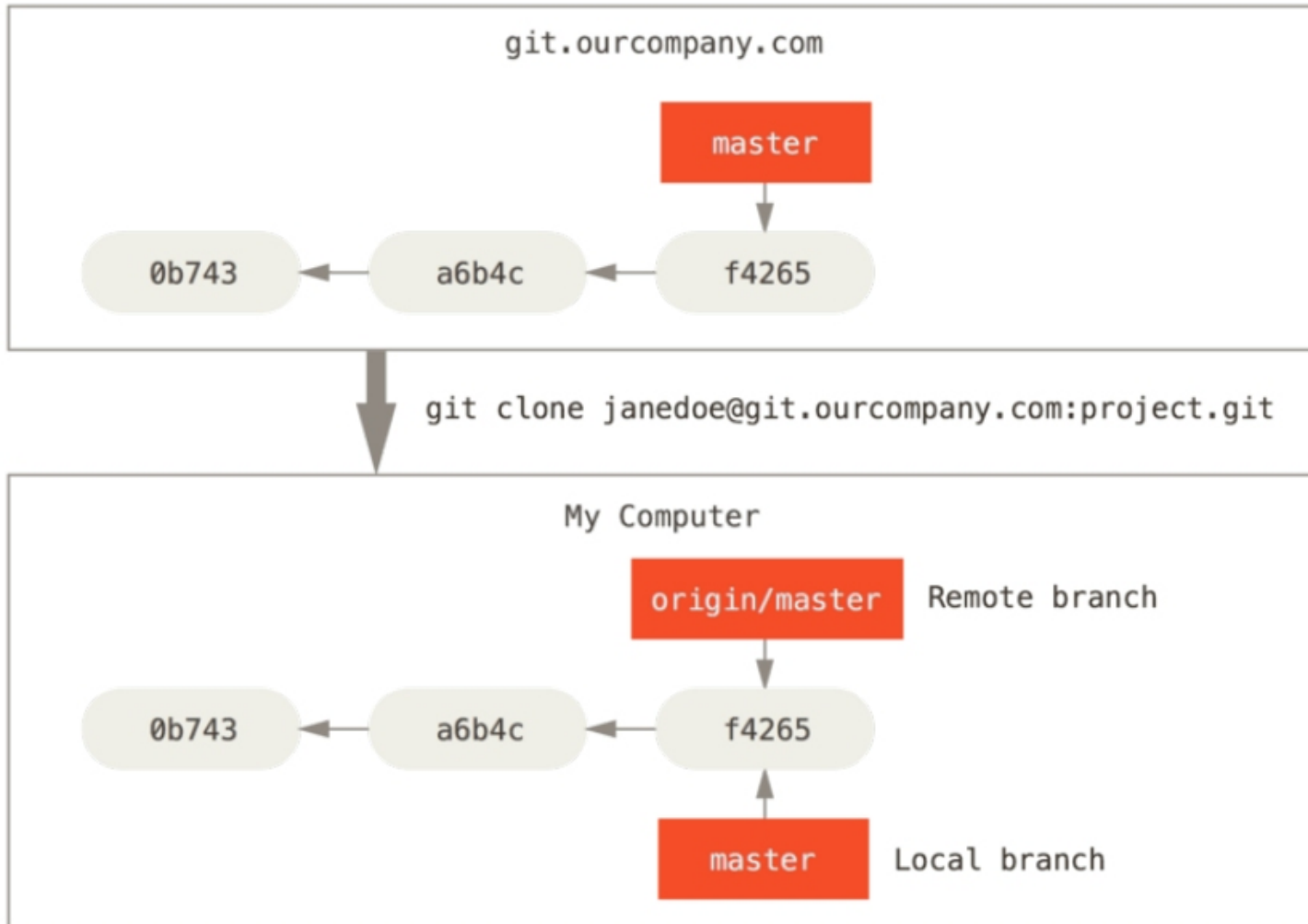
`remote_name/branch_name`

E.g. `origin/master` - master **branch** on "origin"

# Example from Pro Git Book

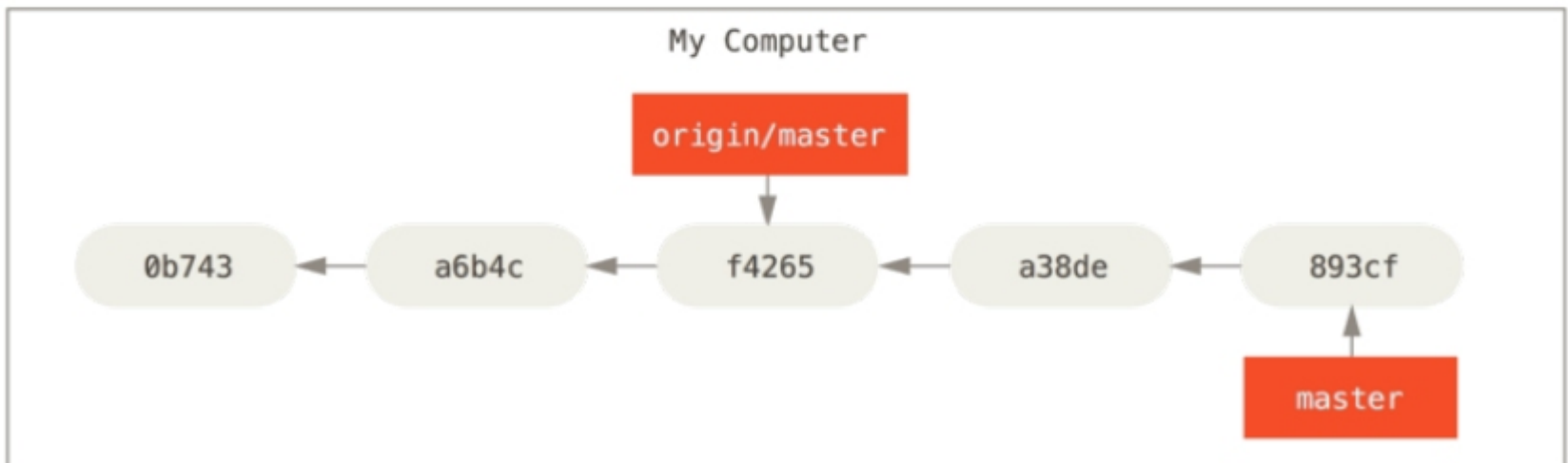
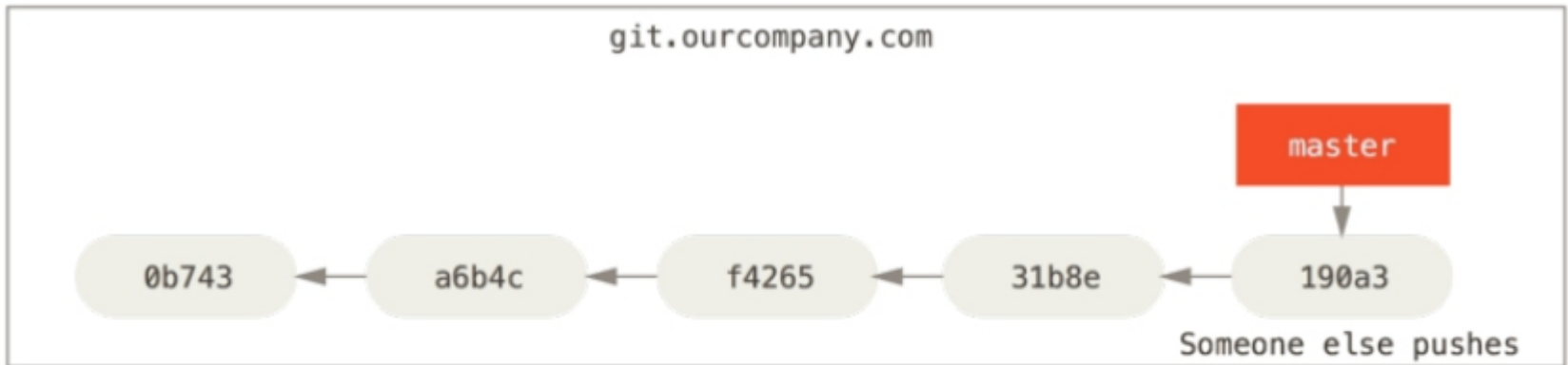
You clone a repo from `git.ourcompany.com`.

Your repo now has 2 labels: `master` and `origin/master`



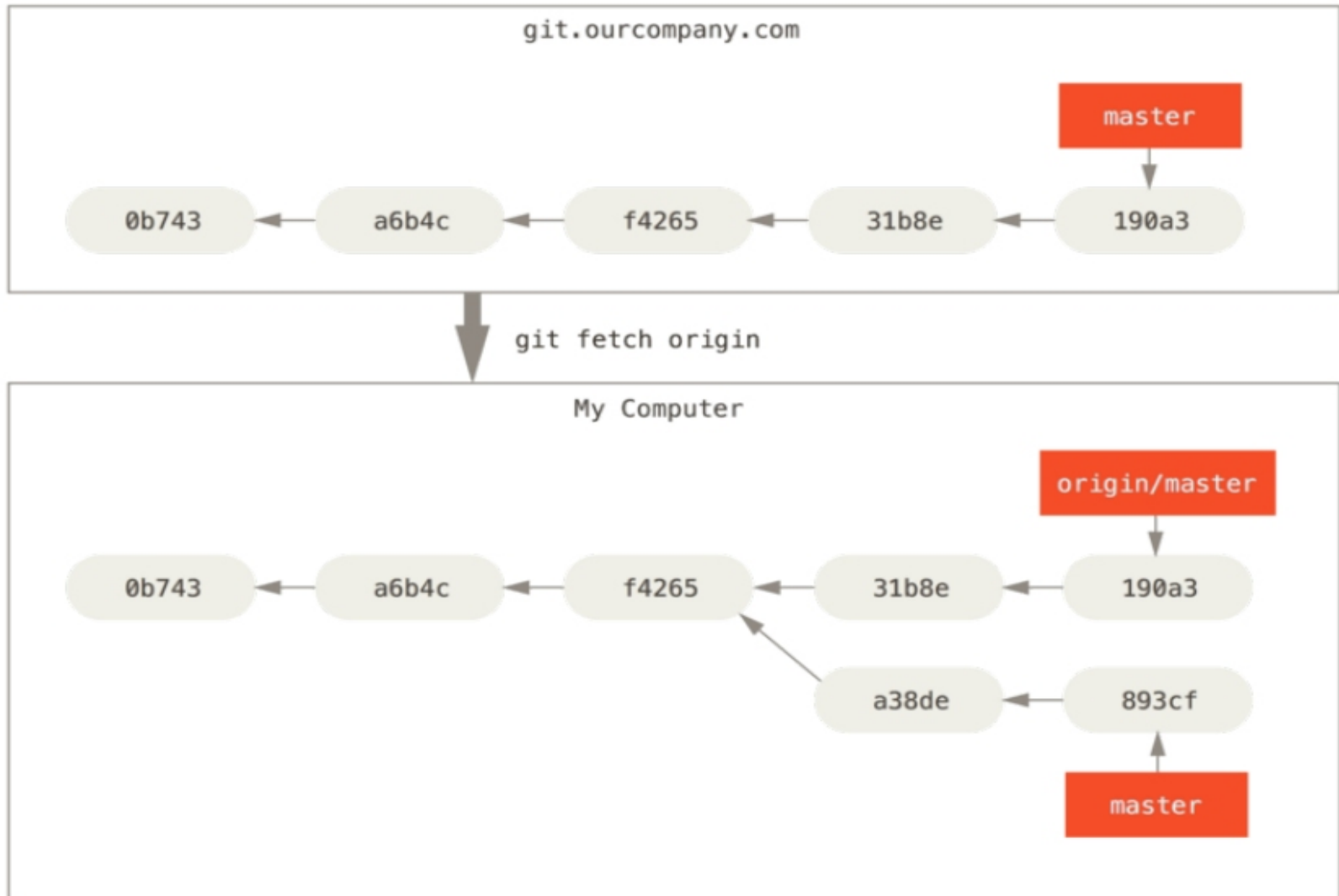
# After some commits, branches diverge

You commit some work locally - your local master moves ahead. Someone else pushes work to ourcompany.com



# Fetch updates from remote

fetch copies the remote branch into your local repo.





# What Has Changed?

---

View the differences. There may be many:

```
cmd> git diff master origin/master
```

```
diff --git a/README.md b/README.md
```

```
index ff3ac4b..1434aa0 100644
```

```
--- a/README.md
```

```
+++ b/README.md
```

```
@@ -1,6 +1,6 @@
```

```
## Unit Testing Procedure
```

```
diff --git a/ctl_test.py b/ctl_test.py
```

```
--- a/ctl_test.py
```

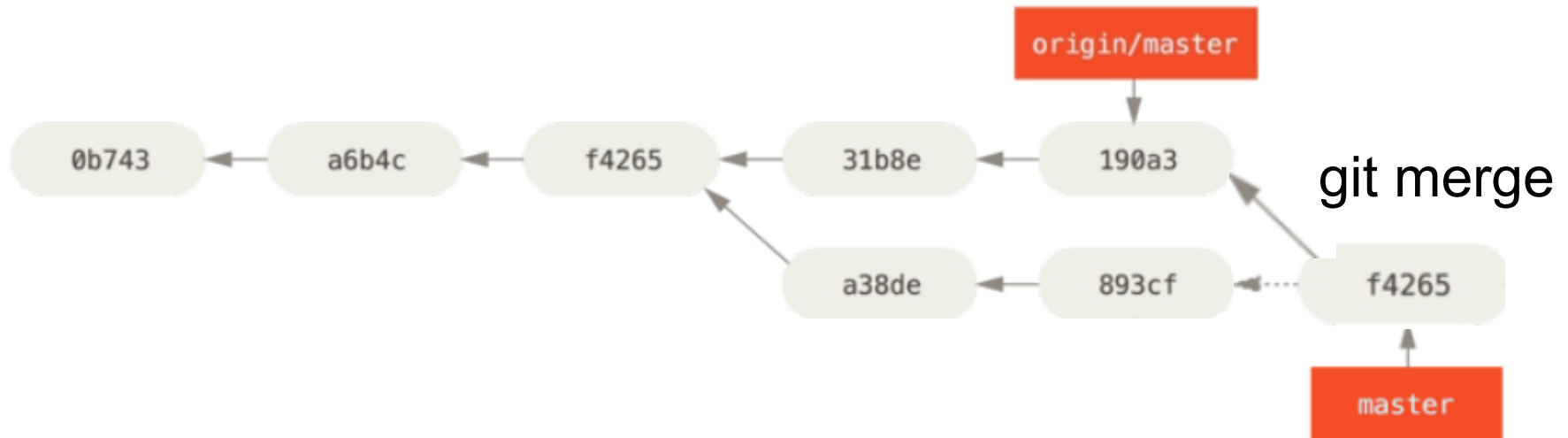
```
+++ b/ctl_test.py
```

# Merge and resolve conflicts

Merge the branches on your computer, resolve any conflicts, and commit.

Now your local master is ahead of the tracking branch.

My Computer



# Push your work to ourcompany

If nothing has changed on ourcompany master, it will advance to match your branch.

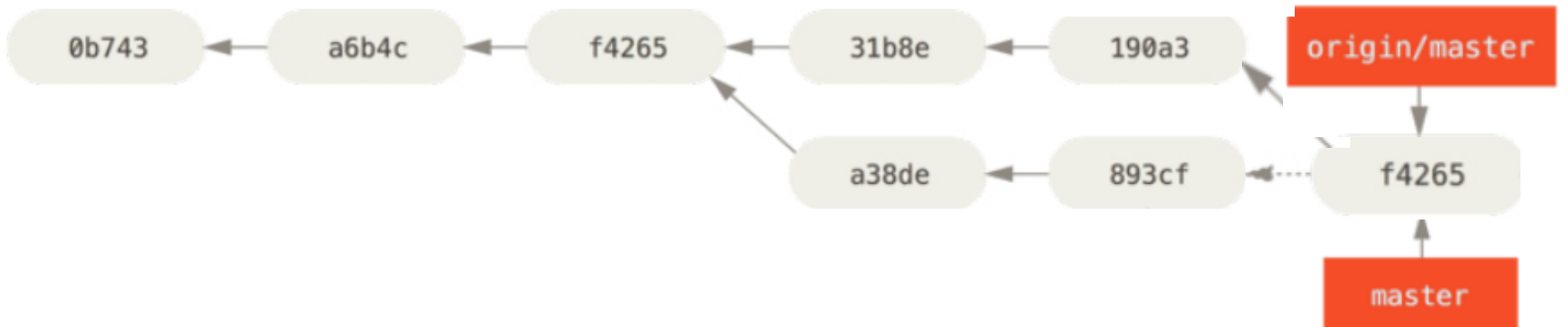
git.ourcompany.com



My Computer



git push



# Questions

---

1. If you "push" when master on ourcompany.com has a commit that is not in your tracking branch, what happens?
2. If you "fetch" and there is no new work on ourcompany.com, what will happen?

# Tracking Branch

---

**Remote-tracking branch** is a local branch that holds a reference to the state of a remote branch.

- tracking branches exist in your local repository
- they update automatically when you contact the remote (push, fetch, pull, remote show) -- you can't modify them yourself.
- name syntax: `remote_name/branch_name`

# Tracking a Remote Branch

---

Two cases:

1. you **checkout** a remote branch and track it
2. you create a local branch, then **push it to a remote** (a tracking branch is created automatically)

# Checkout a Remote & Track it

---

Many commands for this

# these two do the same thing

```
cmd> git checkout --track origin/dev-foo
```

```
cmd> git checkout -b dev-foo origin/dev-foo
```

# create branch with different name from remote branch

```
cmd> git branch -t foo origin/dev-foo
```

(-t is short form of --track)

# Push a Local Branch

---

For a local branch that does not yet exist in the remote repository:

```
cmd> git checkout my-branch
```

```
cmd> git push -u origin my-branch
```

-u is short for --set-upstream

"my-branch" is name to assign to the new branch on origin.



# Who is Ahead? Me or Origin?

---

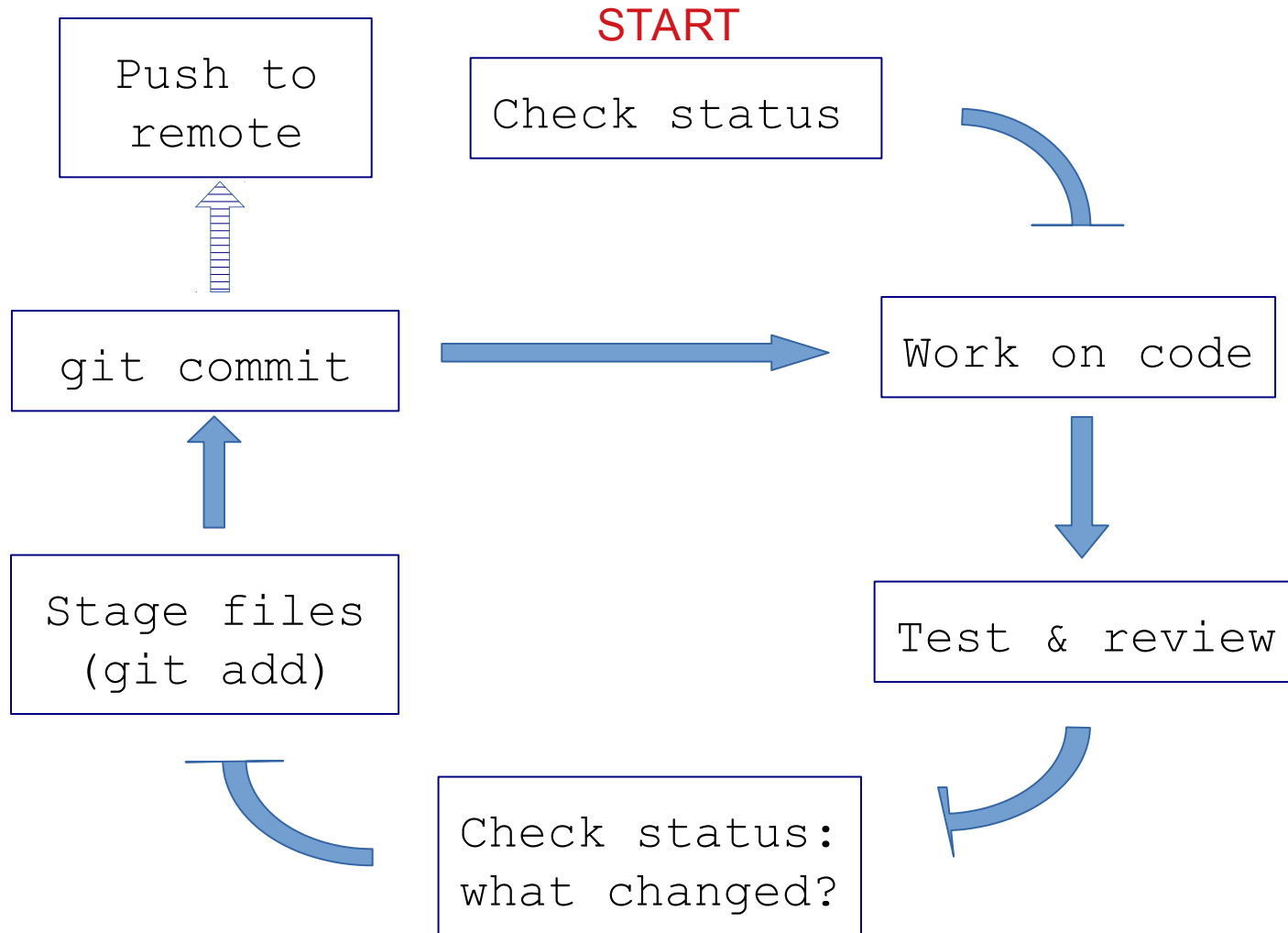
Useful command:

```
cmd> git branch -vv
```

---

# Working with a Remote on a Project

# Workflow for an *individual* project



# Git Workflow for an *Individual* project

---

1) Check status of your working copy (\*)

```
cmd> git status
```

It should be clean. If not, do "git diff" and then...

2) Commit changes or update your working copy.

```
(git diff, git add -u, git commit)
```

3) Do some work:

**Code, test. Code, test. Review.**

(\*) *if you work on more than one computer, you need to "fetch" or "pull" any work from Github that is not on this computer (i.e. this local repo).*

# Git Workflow (cont'd)

---

4) After code-test-reivew: check status again

```
cmd> git status
```

```
Changes not staged for commit:
```

```
    modified:   src/Problem2.java
```

```
Untracked files:
```

```
    src/Problem3.java
```

5) **Add** and **commit** your work to the local repository

```
cmd> git add src/Problem2.java src/Problem3.java
```

```
cmd> git commit -m "Solved problems 2 and 3"
```

```
[master 29abae0] Solved problem 2 and 3
```

```
2 files changed, 44 insertions(+), 5 deletions
```

# Git Workflow (update remote)

---

## 6) Push the changes to Github

```
cmd> git push
```

```
Compressing objects: 100% (12/12), done.
```

```
Writing objects: 100% (12/12), 3.60 KiB,  
done.
```

```
Total 12 (delta 9), reused 0 (delta 0)
```

```
remote: Resolving deltas: 100% (9/9), ...
```

```
To https://github.com/fatailaijon/demo.git
```

```
468abdf..29abae0 master -> master
```

## 7) Take a break.

That's it! Repeat the cycle as you work.

# Github Workflow for Team Projects

---

On a **team project**, other people will commit files to the **same** Github repository!

You should update your local repository from Github before trying to "push" your work to Github.

Use "**Github Flow**" as workflow in team projects.

*"Github Flow" is a separate topic. Its good for both team and solo projects.*

*Github Flow is the convention for team work in this course.*