

# COLLABORATING WITH GIT

## Speaker notes

Learn how to collaborate on [GitHub](#) with [Git](#).

### You will need

- [Git](#)
- A free [GitHub](#) account
- A Unix CLI

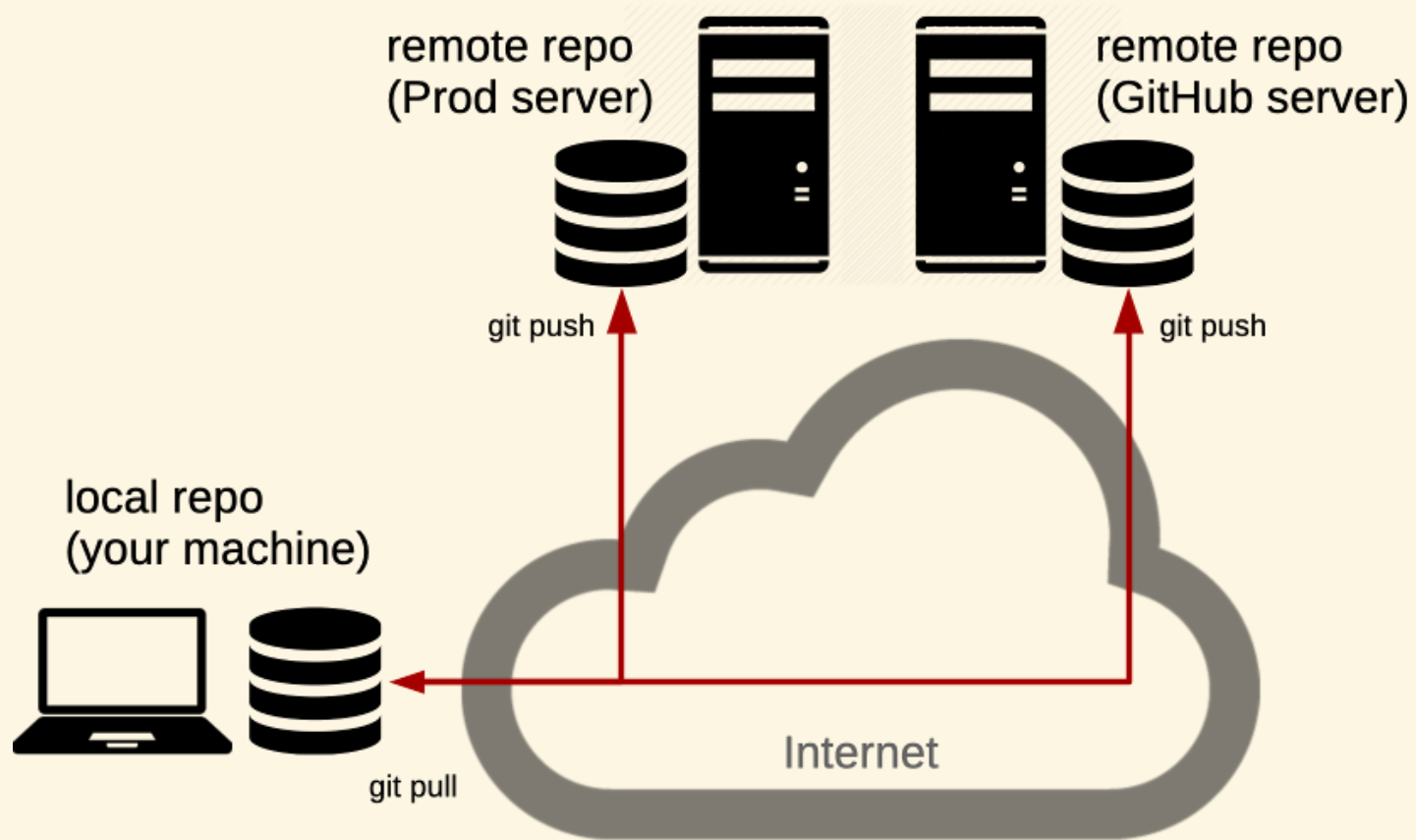
### Recommended reading

- [Version control with Git](#)
- [Git branching](#)

# DISTRIBUTED VERSION CONTROL SYSTEM

Working with remote repositories

# WHAT IS A REMOTE?



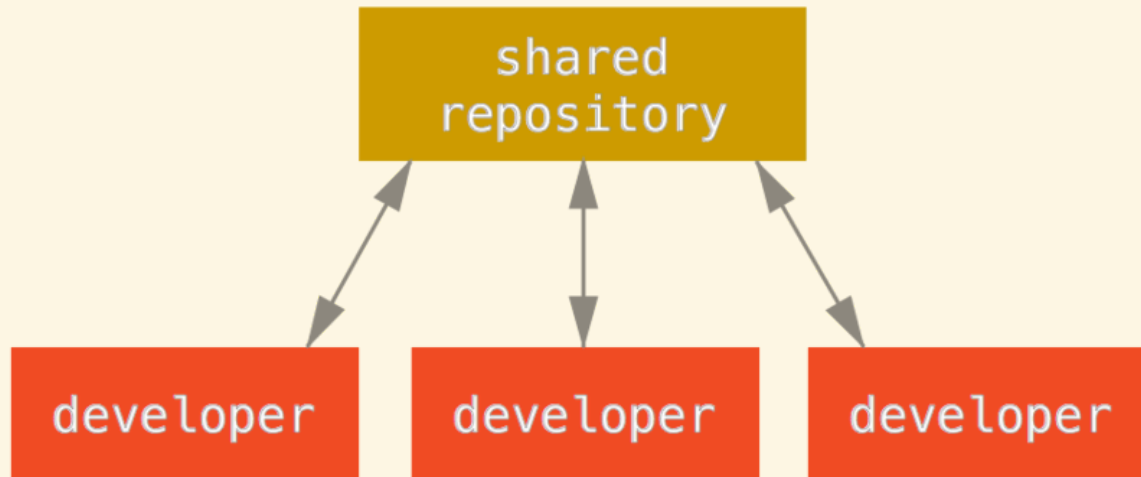
## Speaker notes

A **remote repository** is a version of your project that is hosted on the Internet or network somewhere. You can have **several of them**.

Collaborating with others involves **pushing** and **pulling** data to and from these remote repositories when you need to share work.

# CENTRALIZED WORKFLOW

There are **many ways** to work with Git as a team. Many teams will simply use a simple **centralized workflow**:



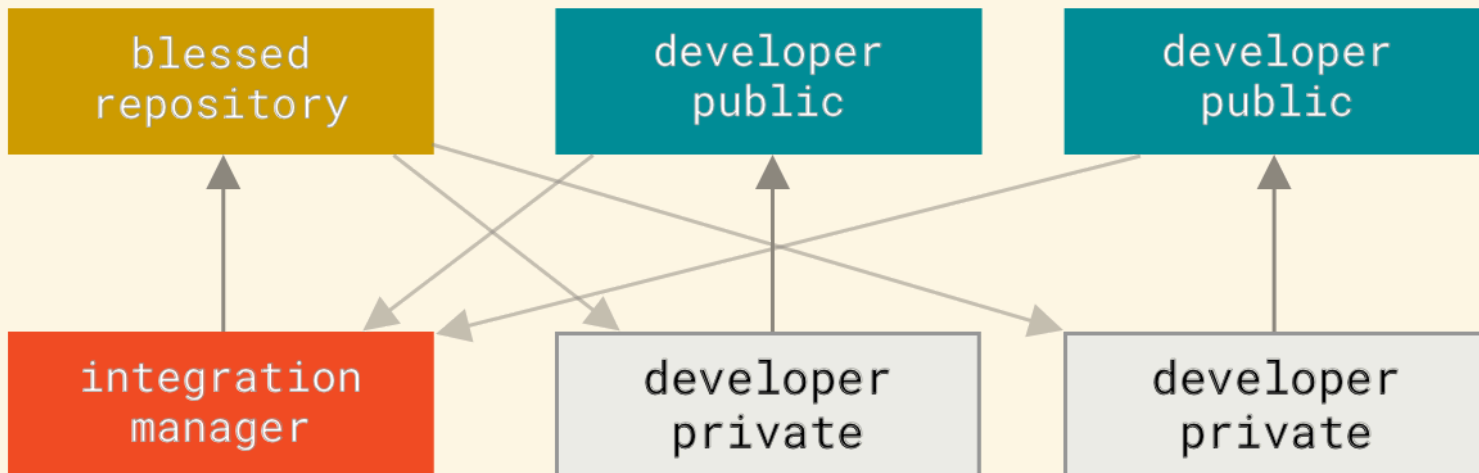
## Speaker notes

In this workflow:

- A **shared central repository** is hosted on GitHub.
- Each developer has a **repository on their local machine**.
  - Each developer will add the shared repository as a **remote**.

# INTEGRATION MANAGER WORKFLOW

The classic workflow for many open source projects:





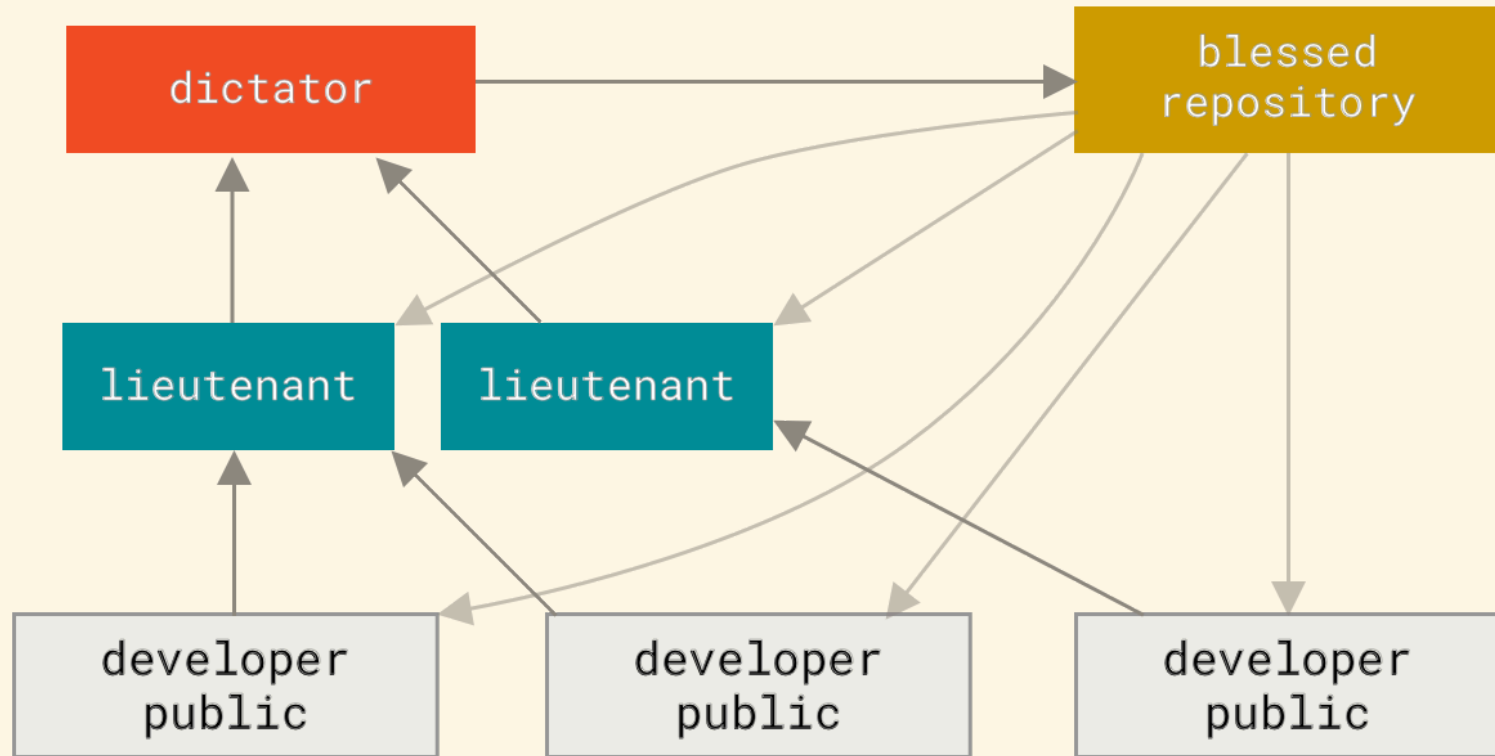
## Speaker notes

- The **project maintainer pushes to their public repository**.
- **Contributors clone that repository**, make changes, **push to their own public copy** and make a **merge request** on GitHub (or via email).
- The **maintainer merges changes** on GitHub (or locally and then pushes them to the main repository).

One of the main advantages of this approach is that you can continue to work, and **the maintainer of the main repository can pull in your changes at any time**. Contributors don't have to wait for the project to incorporate their changes — each party can work at their own pace.

# BENEVOLENT DICTATOR WORKFLOW

A workflow for very large projects:



## Speaker notes

- Regular **developers work on their topic branch** and rebase their work on top of `main` in the reference repository.
- **Lieutenants merge the developers' topic branches** into their `main` branch.
- The **dictator merges the lieutenants' main branches** into the dictator's `main` branch.
- Finally, the **dictator pushes that main branch to the reference repository** so the other developers can rebase on it.

This kind of workflow isn't common, but can be useful in **very big projects**, or in highly hierarchical environments. It allows the project leader (the dictator) to delegate much of the work and collect large subsets of code at multiple points before integrating them.



GitHub is a web-based Git repository hosting service.

## Speaker notes

It offers all of the **distributed version control** and **source code management (SCM)** functionality of **Git** as well as other features like access control, bug tracking, feature requests, task management, and wikis for every project.

