Google

Introducción a Git y GitHub

Rodrigo Caballero
Istio Maintainer
Google Cloud

CCOSS

Rodrigo Caballero

Ingeniero de contenidos

Istio Docs WG Maintainer

GitHub: rcaballeromx

Email: grca@google.com

Google

CC SS

Topics

Contributing to Open Source

- Objectives
- What is Open Source?
- What is GitHub?
- Git, Gerrit, Github, and co.
- GitHub terminology
- Setting up GitHub
- GitHub flow
- GitHub and branches
- Using Git well

Objectives

- Improve our understanding of Git
- Improve our collaboration with GitHub
- Reduce the overhead for publication
- Avoid pitfalls when working with GitHub
- Provide solutions for common problems

What is Open Source?

- A culture: Everyone starts as a contributor.
- A community: We help each other.
- A methodology: We collaborate following rules.
- A set of principles: We want our users to be free.

What is GitHub?

GitHub is a code hosting platform for version control and collaboration.

- GitHub Documentation

What is GitHub?

GitHub is a **code** hosting **platform** for **version control** and **collaboration**.

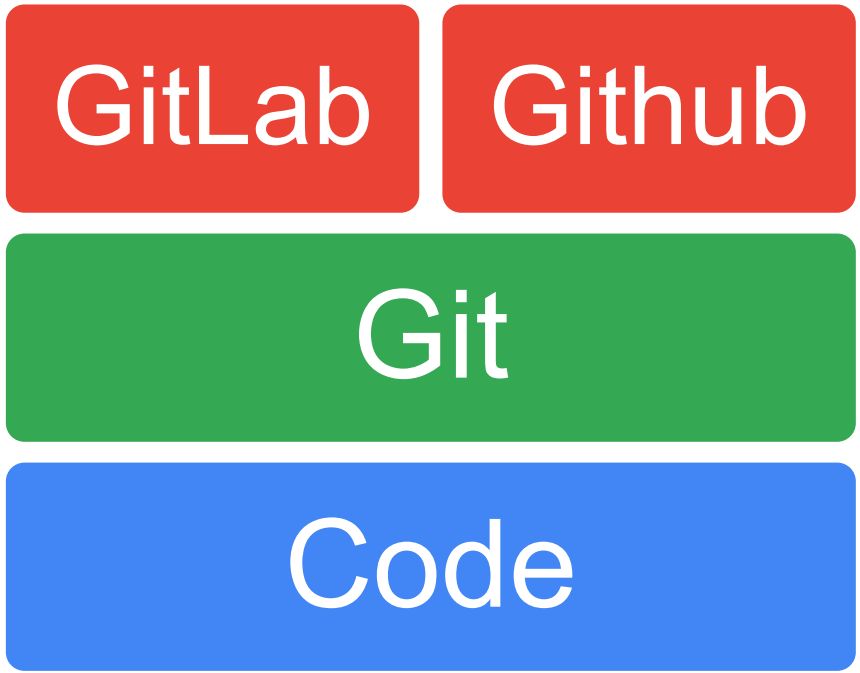
- GitHub Documentation

Git, Gerrit, Github, and Co.

Disambiguation, once and for all:

- Git is the version control system.
- GitHub hosts the repository.

Git is infrastructure software. Github, GitLabs, TeamForge, etc. are middleware.



Git terminology

Git Directory

- ✓ The .git directory that holds all information in the repository
- ✓ Hidden from view

Working directory

- ✓ The directory in which .git resides
- ✓ Contains a "snapshot" of the repository
- ✓ Will be changed constantly eg. when reverting or branching
- ✓ Your changes will be made to this directory

Index

- ✓ Changes have to be added to the index from the working directory in order to be saved into a commit
- ✓ Could be thought of as a “loading bay” for commits

Commit

- ✓ A set of changes that have been saved to the repository
- ✓ Can be reverted and even modified to some extent
- ✓ Identified by a hash of the changes it contains

Tag

- ✓ A certain commit that has been marked as special for some reason
- ✓ For example used to mark release-ready versions

Git terminology

HEAD

- ✓ The latest revision of the current branch
- ✓ Can be used to reference older revisions with operands

  HEAD^^2 == 2 revisions before latest

  HEAD~3 == 3 latest revisions

  HEAD^^2..HEAD == 2 revisions before the latest to the latest

Branch

- ✓ An alternate line of development

Working copy

- ✓ The branch you are in now that you make your changes in

Master

- ✓ The default branch of the repository

Origin

- ✓ Default name for a remote repository when cloning an existing repository

GitHub account

- Create your GitHub account using your email address.
- Use your real name when completing your profile.
- Choose your GitHub username wisely.
- Enable 2-factor authentication.

Setting up GitHub

GitHub clients are available for Windows, Linux, and Mac OS.

Only the user interface and configuration change.

On Windows it is highly recommended to install an IDE such as VSCode or GitHub + Git Extensions.

www.github.com

https://sourceforge.net/projects/gitextensions/

Adequate use of GitHub requires:

- A GitHub account
- SSH keys or HTTPS for security
- A defined collaboration process
- Understanding of branches/checkout

Setting up GitHub: Aliases

Aliases are nicknames you define for complex Git instructions.

They are stored locally in the `.gitconfig` file under `[alias]`

Aliases can be extremely personal and some developers will outright refuse to share them.

You can define aliases only for a repo, with the option `--local`, or for the system, with the option `--global`.

Common aliases:

```
git config --global alias.co checkout
git config --global alias.ci commit
git config --global alias.st status
git config --global alias.br branch
git config --global alias.hist "log
--pretty=format:'%h %ad | %s%d [%an]'
--graph --date=short“
```

Edit aliases with:

```
git config --global -e
```

SSH Keys and HTTPS

GitHub uses SSH to authenticate requests.

Add the public SSH key for each host system you use with your account.

GitHub also supports HTTPS.

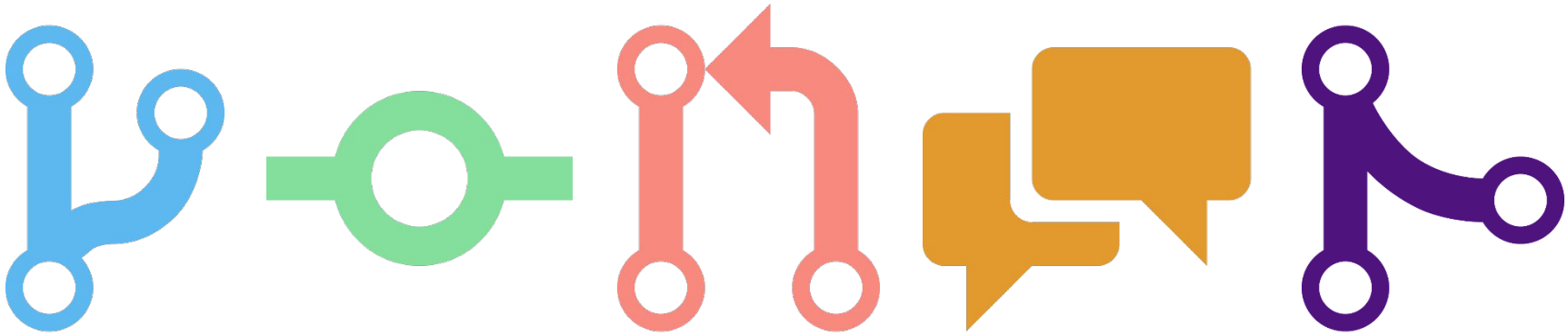
Company proxies may make it necessary for you to use HTTPS, particularly on Windows.

WARNING!

Only add the public key and never add the private key to GitHub.

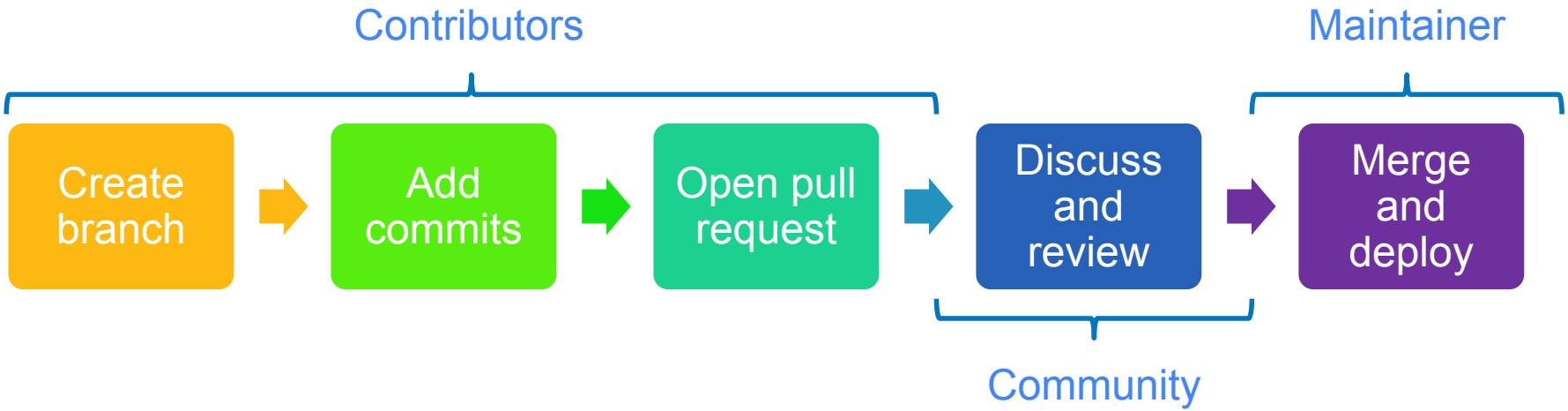
Adding the private key might cause a security risk.

The GitHub flow



Create branch → Add commits → Open pull request → Discuss and review → Merge and deploy

The GitHub flow: Roles



The GitHub flow: Mechanics

Create branch	git checkout –b newBranch
Add commits	git commit -s
Open pull request	git push –u origin newBranch
Discuss and review	Follow contribution guides
Merge and deploy	Test first!

The GitHub flow: Contributing to the repo

1. Fork the **upstream**.
2. Clone your **fork**:

```
git clone https://github.com/rcaballeromx/istio.io.git
```

3. Create a new **branch** for your changes:

```
git checkout -b newBranch
```

The GitHub flow: Contributing to the repo

4. Make your changes and add them to the **index**.

```
git add -A
```

5. **Commit** your changes locally.

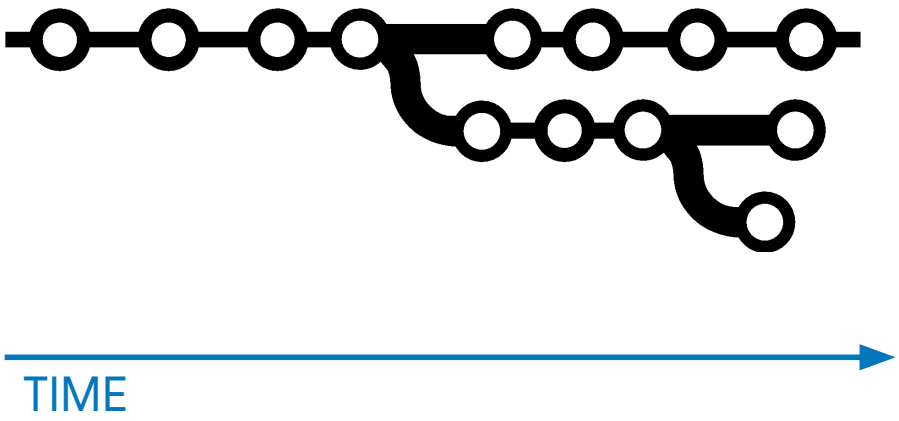
```
git commit -s
```

6. Push your changes to your **remote fork**.

```
git push -u origin newBranch
```

GitHub and branches

- Branches quickly add overhead complexity.
- GitHub keeps track of branches.
- Branches as your worst enemy or your best ally.
- Remote branches are branches on a remote repo: your fork or the upstream.
- Local branches are branches only available on your local Git system.

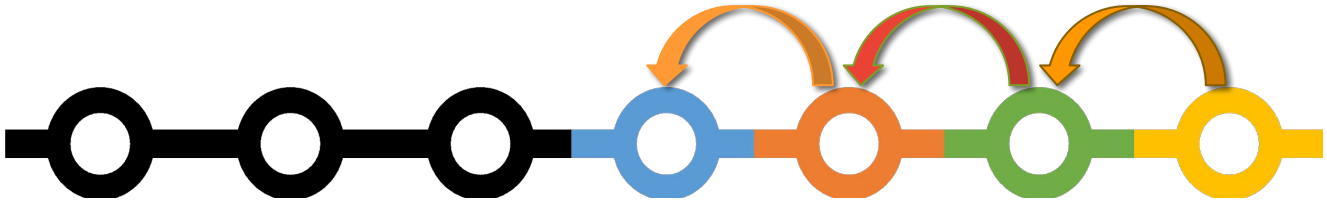


GitHub and branches

- You must sync your local Git history with the **remote fork**.
- To submit a change, never work on the **master branch**.
- **The collaboration history is kept on GitHub.**
- **Remote branches** help you **collaborate** easily on multiple features.
- **Local branches** help you **manage** your work.

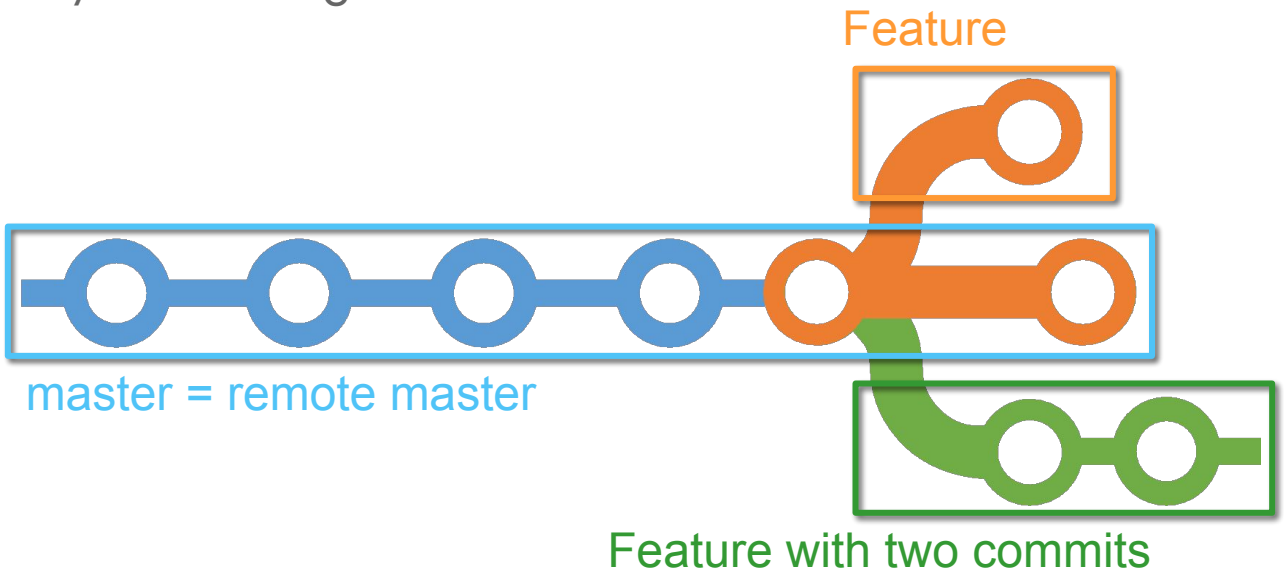
GitHub and branches

- In **branches**, one **commit** modifies the code based on the previous one.
- Once **pushed** to the **remote**, the **commit history** is visible to others.
- After **pulling** the **remote** and your **local** repos are identical.
- At first, you don't need to **force push** when using GitHub.



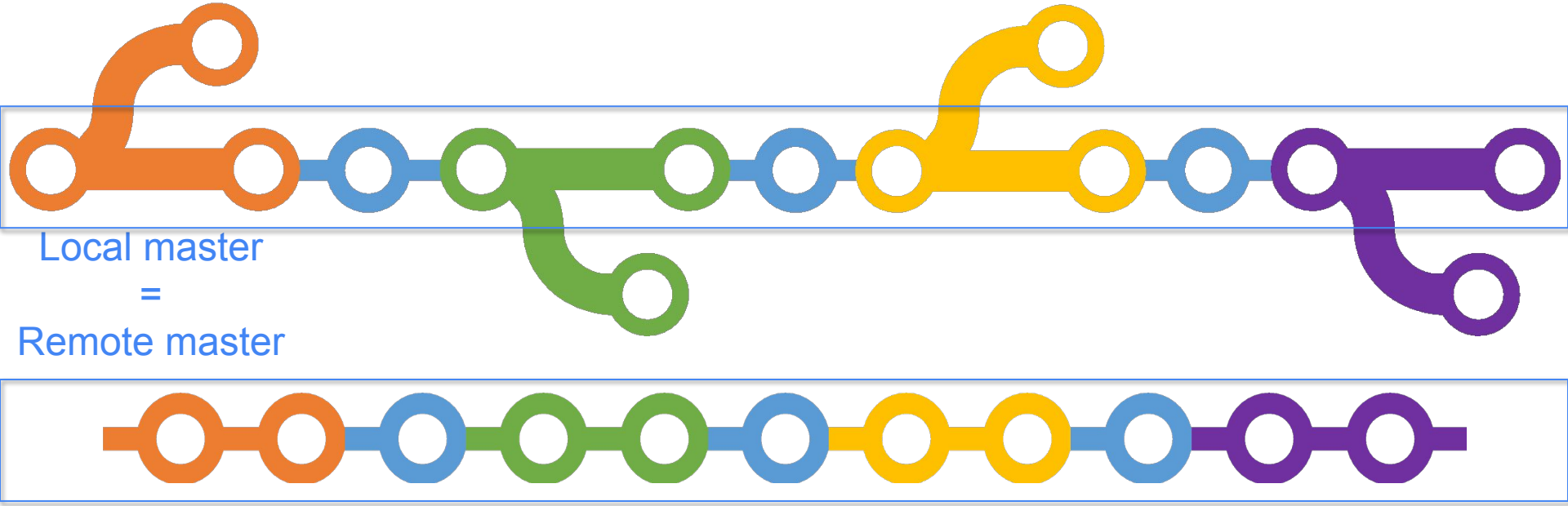
GitHub and branches

Branches let you keep track of your work. Remote branches let your team contribute to your working.



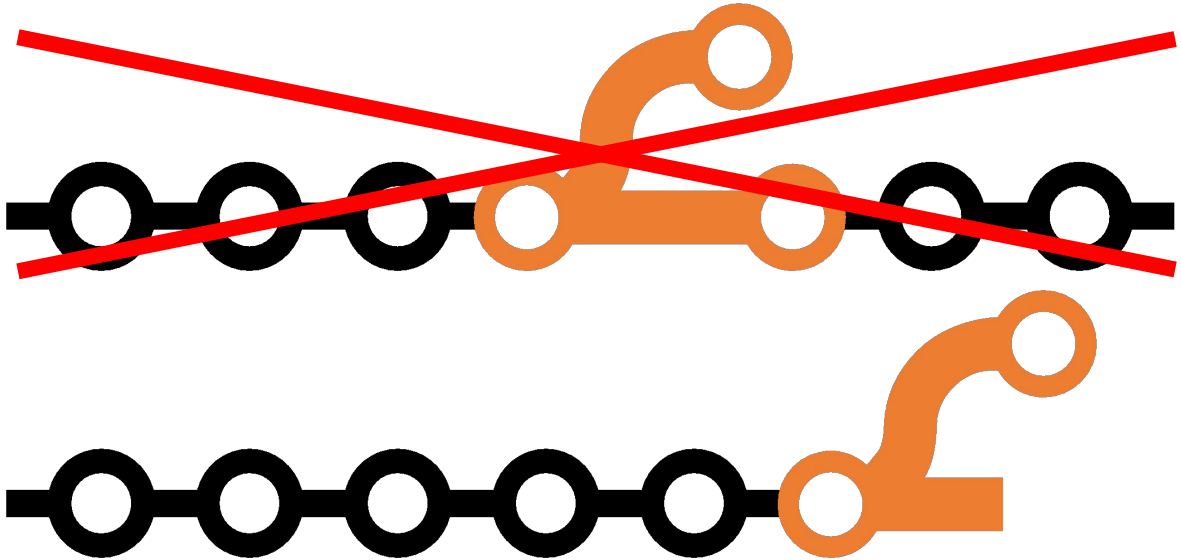
GitHub and branches

To move from one feature to another, you simply **checkout** the branch.



GitHub and branches

If you are starting a new branch, rebase!



Using Git well

Total impact

Add your total impact to the project's repository:

# of lines added and removed

Overhead unit

Commits are Git's overhead unit.

# of commits

Effectiveness coefficient

The less commits you create to contribute, the cleaner the history becomes and the easier it is to manage.

Key learnings

- Git is a **version control system**.
- GitHub is a tool for **repo management** and **code review**.
- A **pull request** is a request to **merge** a **branch** into master.
- A **branch** is a **sequence of commits** containing certain **edits**.
- Use **one branch** per **feature**.
- **Rebase** your branches against the **remote master**.
- Aim to **increase** your **impact** while **reducing** the **overhead** needed.

References

GitHub Documentation

https://guides.github.com/

Pragmatic Version Control Using Git (Pragmatic Starter Kit) Travis Swicegood

The Git Community book

http://book.git-scm.com/

Git 101

Scott Chacon, Github

http://www.slideshare.net/chacon/git-101-presentation

Git How To

https://githowto.com/

Smacking Git around (Advanced Git)

Scott Chacon

http://www.slideshare.net/railsconf/smacking-git-around-advanced-git-tricks