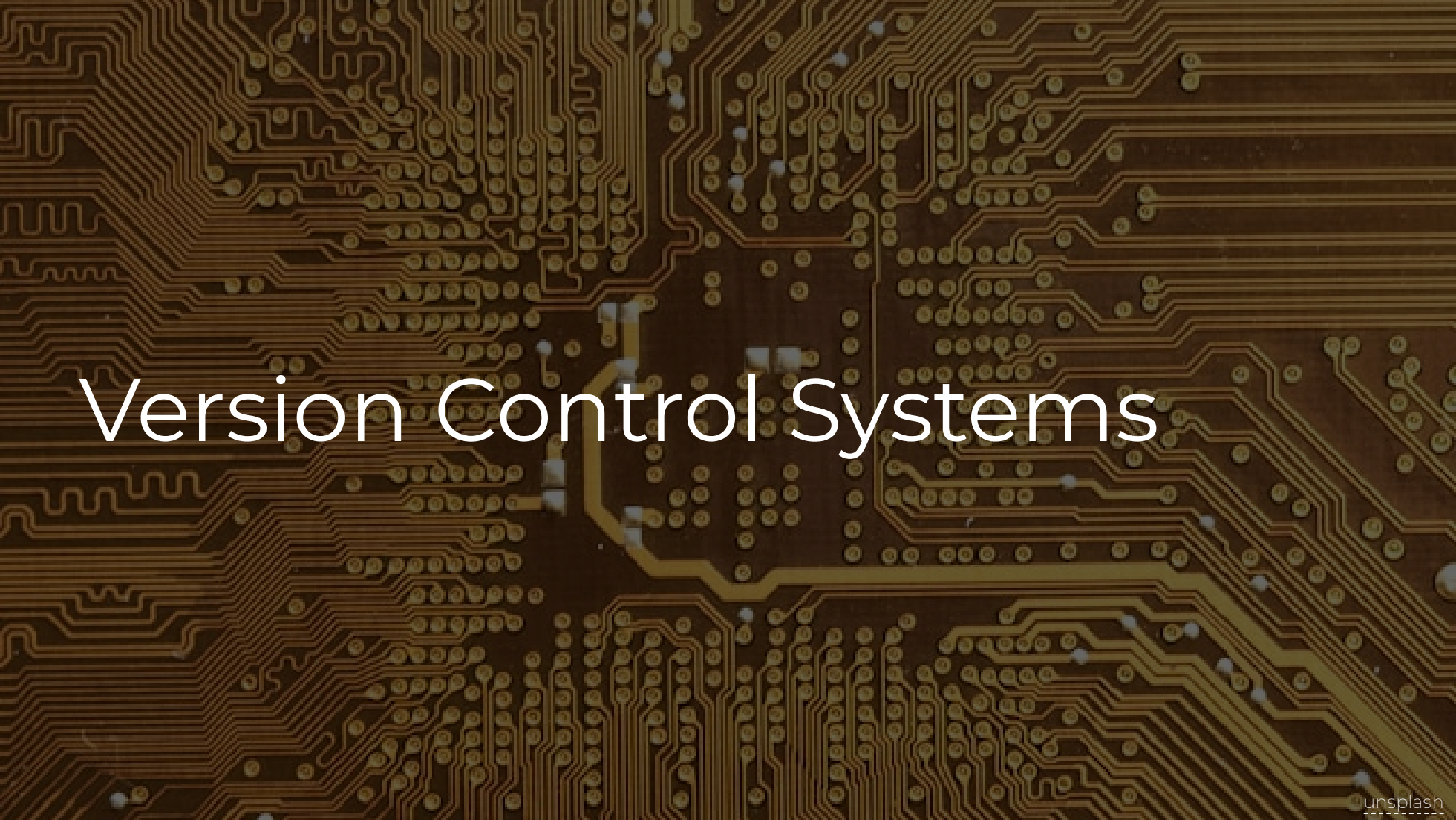

Version Control Systems
unsplash

PDF Slide Deck 'Version Control'

This slide deck is only available as PDF. Download the slides here and view them with your favorite PDF Viewer.

Platforms For Git?

Remember: Git is designed to work as decentralized/distributed version control system without a central server.

Yet, many people only know git because of *Github* or *Gitlab* (or other platforms)? Why?

Because those platforms provide additional value and functionality that help managing projects with git.

Advantages of using a git platform

- Single place to get an overview of existing repositories
- Convenient creation of new (remote) repositories accessible to all developers
- Advanced roles and rights management (especially important for larger organizations)
- Integration of git and the source code with additional functionality such as
  - Code Review with comments and diff view ("merge/pull requests")
  - Automation using DevOps technology (CI/CD pipelines, registries, ...)
  - Issues (Bug reports)

Options

There are many platforms combining git with other features. The following list will certainly miss some of them:

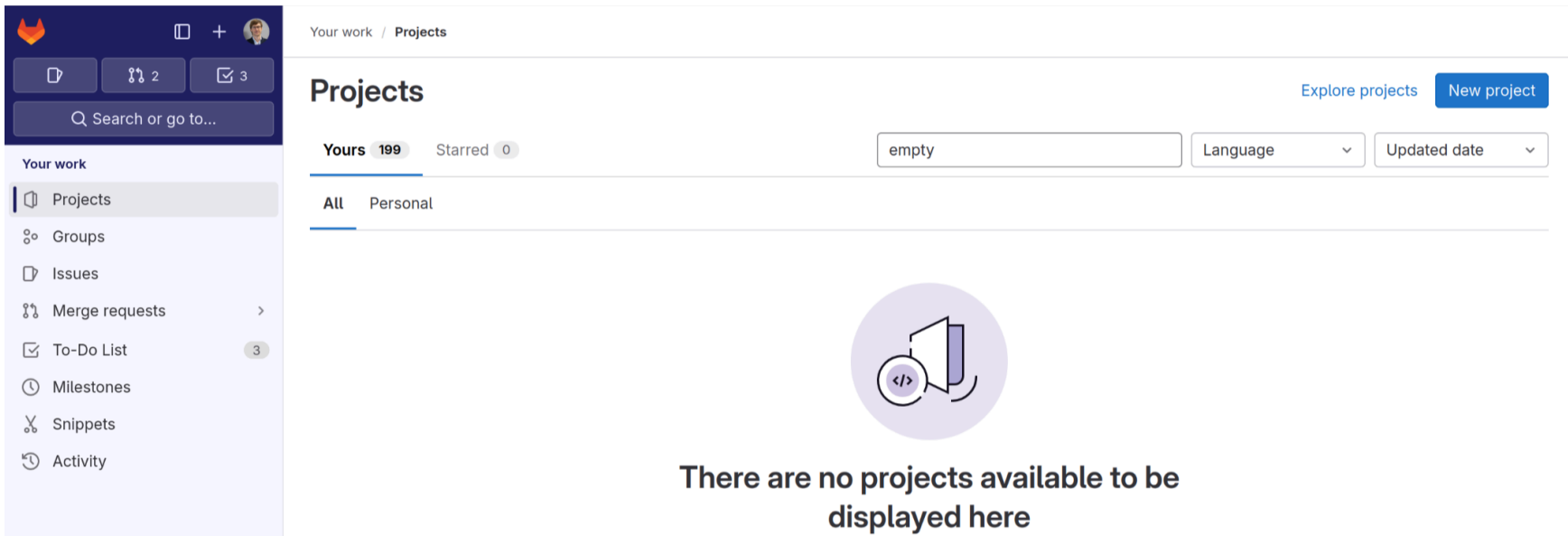
- GitHub
- GitLab
- Bitbucket
- SourceForge
- Beanstalk
- Assembla
- Azure DevOps
- Gitea
- Codeberg
- AWS CodeCommit
- GitBucket
- RhodeCode

Popular choices are Github and Gitlab; or (if already using the respective cloud), cloud-based ones such as Azure Devops.

i We are going to use Gitlab, which has a 'Community' version that allows self-hosting and is a popular choice in companies that want to save the premium costs (and do not need the premium features).

Introducing GitLab

GitLab is a complete DevOps platform, delivered as a single application. From project planning and source code management to CI/CD, monitoring, and security.



Important functionality

1. Creating new groups and repositories
2. Forking existing repositories
3. Managing Access
4. Clone repositories
5. Create Merge Requests and do code reviews
6. Use automated CI/CD pipelines [later]
7. Use the Docker image registry [later]

The goal of this exercise is to create and use a Gitlab repository for managing some file changes.

- Form groups of two students (remote need to organize via Chat or different video meeting, e.g., https://meet.th-deg.de)
- One students creates a new repository in the personal namespace and adds the other student with role *Developer*
- Each student then...
  - clones the repository using the *Clone with HTTPS* option to the lab PC / laptop
  - creates a new branch using a unique name (`` `git checkout -b {UNIQUE_BRANCH_NAME}` ``)
  - creates at least one file containing some code (e.g., Hello World in any language you like)
  - adds, commits and pushes the changes to Gitlab

Using the same groups and previously created project, create a merge request and do some review!

- Each student creates a Merge Request to merge the created branch into `` `main` ``.
- Assign the other student as "Reviewer"
- Perform the code review for each other:
  - Check the diff view - what was changed/added?
  - Make some comments on specific lines of the code
  - Suggest a code change
- After reviews are done, check the comments, but decide to ignore them! Answer with "Nop." and merge the branch!

Think about the code review process. Would this work in a small company/project? What about larger organizations? Can you come up with some changes that would be required so that a code review can not be ignored?