What is DevOps?

DevOps is an approach to increase efficiencies in the software development lifecycle by combining development and operations, which is largely achieved with automation via CI/CD pipelines. CI/CD is short for "continuous integration" and "continuous delivery". The paradigm is designed to provide automation for testing, building, and deploying code, which have traditionally been manual tasks.

Continuous integration is the development practice of committing code little and often, rather than merging multiple changes in one go. Each individual commit can trigger automatic testing, which allows errors such as build failures, code conflicts, dependency issues, etc to be identified and fixed early in the development lifecycle.

Continuous deployment is a software delivery practice to automate the final (release) build of a project and delivery of the application to production. Since code has already been tested and validated through the CI process, the CD process simply ships the changes to production.

A CI/CD "pipeline" is the series of individual pre-defined steps (e.g., build, test, deploy). The pipeline is responsible for "aborting") if any step is deemed to have failed so that bad changes are not deployed to production.

Why do we want DevOps?

It's common for penetration testers and red teamers to write their own tooling to support their operations. Like any other piece of software, automatically testing changes and building those tools saves time in the long run. If you maintain code in a public repository and accept changes from the community, the appropriate CI/CD pipeline will help validate them before they're merged (so as to not introduce breaking changes).

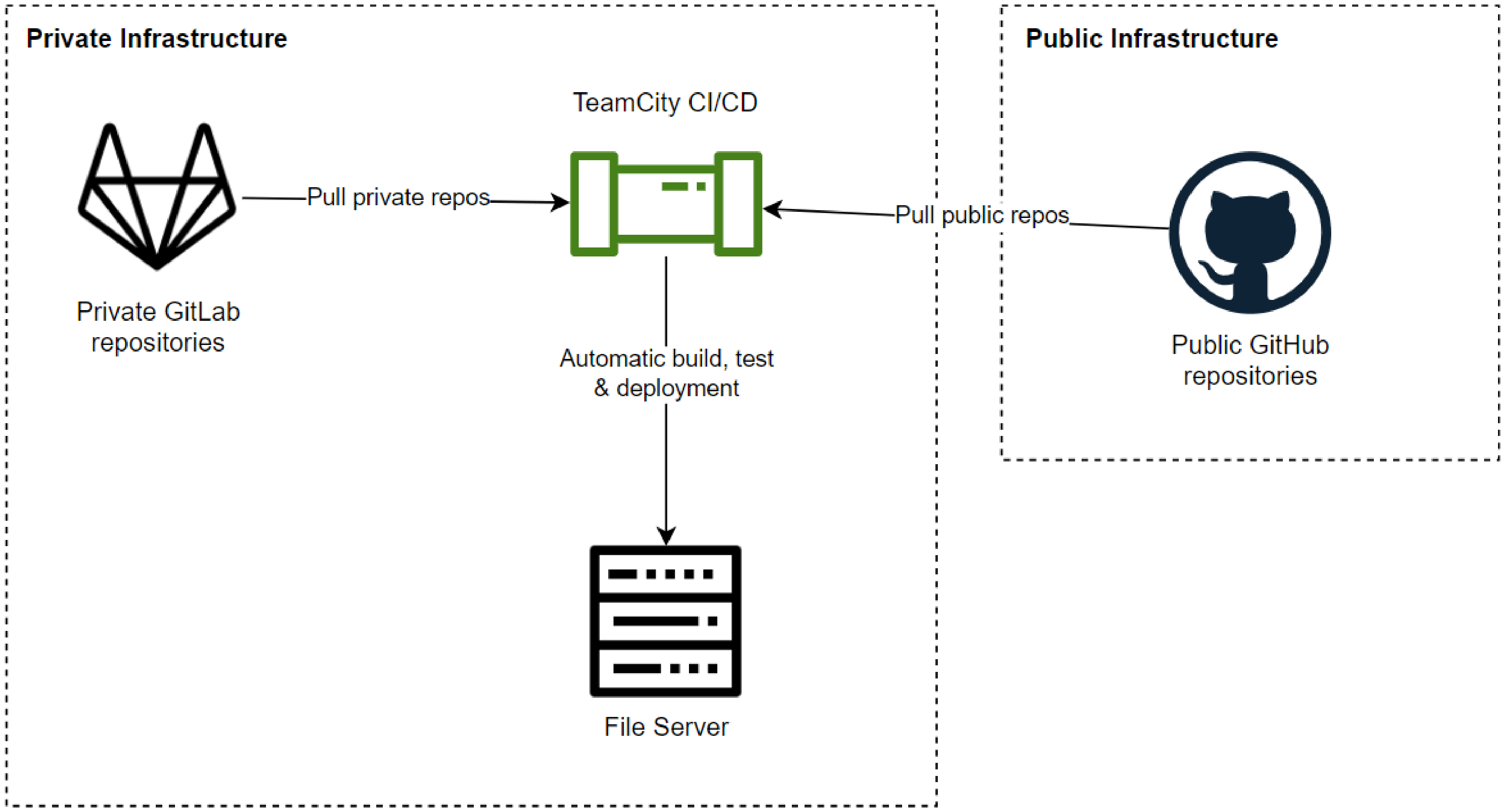
You may also use well-known public tools written by other people. These are often already well signatured by AV engines and a lot of manual effort may be needed to remove those from the code base. CI/CD pipelines can automate this process and provide "clean" builds of all your tools at the press of a button.

Everyone's risk tolerance on this subject will differ, but it fundamentally comes down to the question: do you trust vendors not to collect telemetry on your tools?

For instance, Microsoft own Azure DevOps, GitHub, and Defender. As unethical as it may be, it would be "smart" of them to collect data from GitHub repositories and Azure CI/CD pipelines that contain "malicious" code, and then automatically feed signatures into Defender.

Microsoft have already been criticised in the past for removing proof of concept exploit code from GitHub. This may or may not be important to you, but you will have more control if you use your own hardware instead.

This course will focus on building your own CI/CD infrastructure using GitLab and TeamCity.


Private Infrastructure
Public Infrastructure
TeamCity CI/CD
Pull private repos
Pull public repos
Private GitLab repositories
Public GitHub repositories
Automatic build, test & deployment
File Server