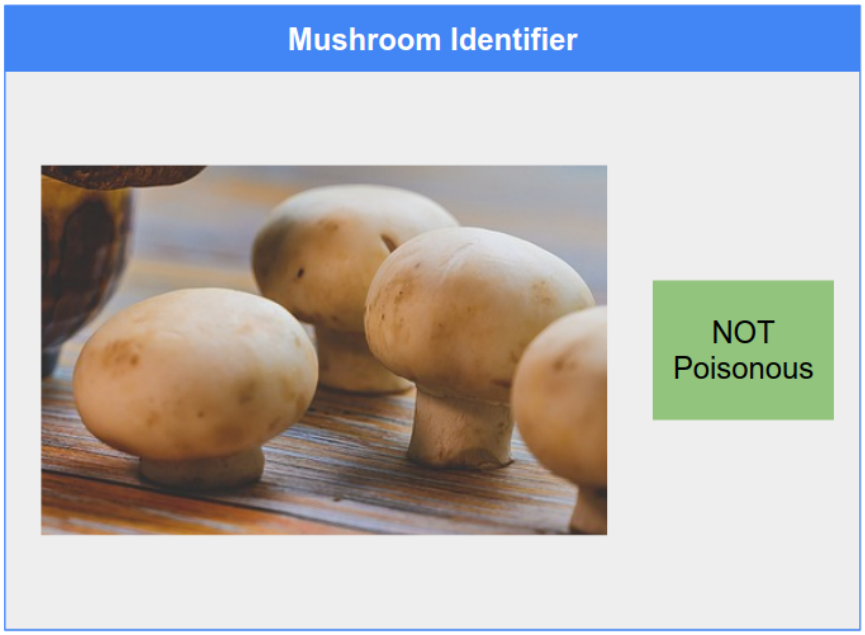
Continuous Integration Tools

What is Continuous Integration?

To anyone not familiar with DevOps, the term continuous integration sounds a little daunting. In reality, though, continuous integration is a simple but powerful concept. Imagine you are part of a team that is developing an open-source machine learning application that allows users to upload pictures of mushrooms that then tells them if they are poisonous or not. This seemingly simple application has a lot of moving parts to consider: data, models, and a web interface.



Most likely, your team will all have to work on different aspects of the same code base, using some form of decentralized source code repository such as GitHub or GitLab. After a hard day of work, you finally get the application up and running. You decide to push the code changes to the group repository so that everyone now has access to this working version, and then go out to celebrate. The next day, you find that another colleague has pushed some additional code to add some extra functionality. The problem? It broke your application! Now nobody can learn whether their mushrooms are poisonous. The real problem here is not really that the application broke, but the fact that nobody seemed to notice. This is where continuous integration comes in to save the day.

Continuous integration is a software development technique of automating the integration of changes to our coding environment. This might entail performing a suite of tests whenever our code is changed, to ensure that everything is still functioning correctly, like our mushroom application, as well as things like revision control, and build automation. This technique is helpful in highlighting problems that may not have been picked up on, and automating how

these changes are dealt with - for example, do we decide to reject changes if they fail? Is it okay if certain tests fail as long as the major functionality remains?

Benefits

There are several benefits to continuous integration:

Reduced risk. Defects and bugs are detected faster and more frequently, which means they are less likely to sneak through into a production environment and are dealt with more efficiently.

Improved communication. By working in a more collaborative, automated, and regulated environment, communication is faster and more transparent.

Improved product quality. Because errors are more likely to be found during the development stage, the overall quality of the product will be improved.

Reduced waiting time. When using continuous integration, we do not need to wait for someone to send us code or to perform manual checks which may take up valuable time.

Continuous Integration Tools

Renode. Renode is a development framework which accelerates IoT and embedded systems development by letting you simulate physical hardware systems - including both the CPU, peripherals, sensors, environment and wired or wireless medium between nodes. It lets you run, debug and test unmodified embedded software on your PC - from bare System-on-Chips, through complete devices to multi-node systems. It is specifically useful for TinyML!

Renode strengths:

- full determinism of execution, shared virtual time
- transparent & robust debugging, tracing, analysis, even in multi-node setups
- easy integration with your everyday tools, plugins
- rich model abstractions with additional functionality "for free" + modular platform description format
- automated tests and CI integrations, other collaboration features

Jenkins. Jenkins is a more general purpose continuous integration/continuous delivery (CI/CD) server and is one of the most commonly used CI tools. The architecture works by:

- Files are committed to a code repository
- Jenkins checks the repository at regular intervals to pull recent code changes
- A build server compiles the code into an executable file; if this procedure fails, feedback is sent to developers
- If successful, the executable is sent to a test server; if this procedure fails, feedback is sent to developers
- If the code runs successfully on the test server, it is deployed on the production server

GitHub Actions. GitHub Actions works similarly to Jenkins, but is built-in to GitHub. This works by detailing tests to run based on certain events, such as when a new branch is merged or new code is pushed to the repository. These details are listed in YAML format in a ".github/workflows" directory. Some examples of GitHub Actions can be found here.

TravisCI. TravisCI is a popular open-source CI tool that can be easily integrated with GitHub repositories. It works very similarly to GitHub Actions, except the YAML-format commands are stored in a ".travis.yml" file. This tool is popular in the Python community for testing that applications run correctly on multiple versions of Python.

Other examples also exist, such as Amazon SageMaker, Google TFX, CircleCI, Bamboo, and Buddy. We will discuss these later in the course as they become more relevant.