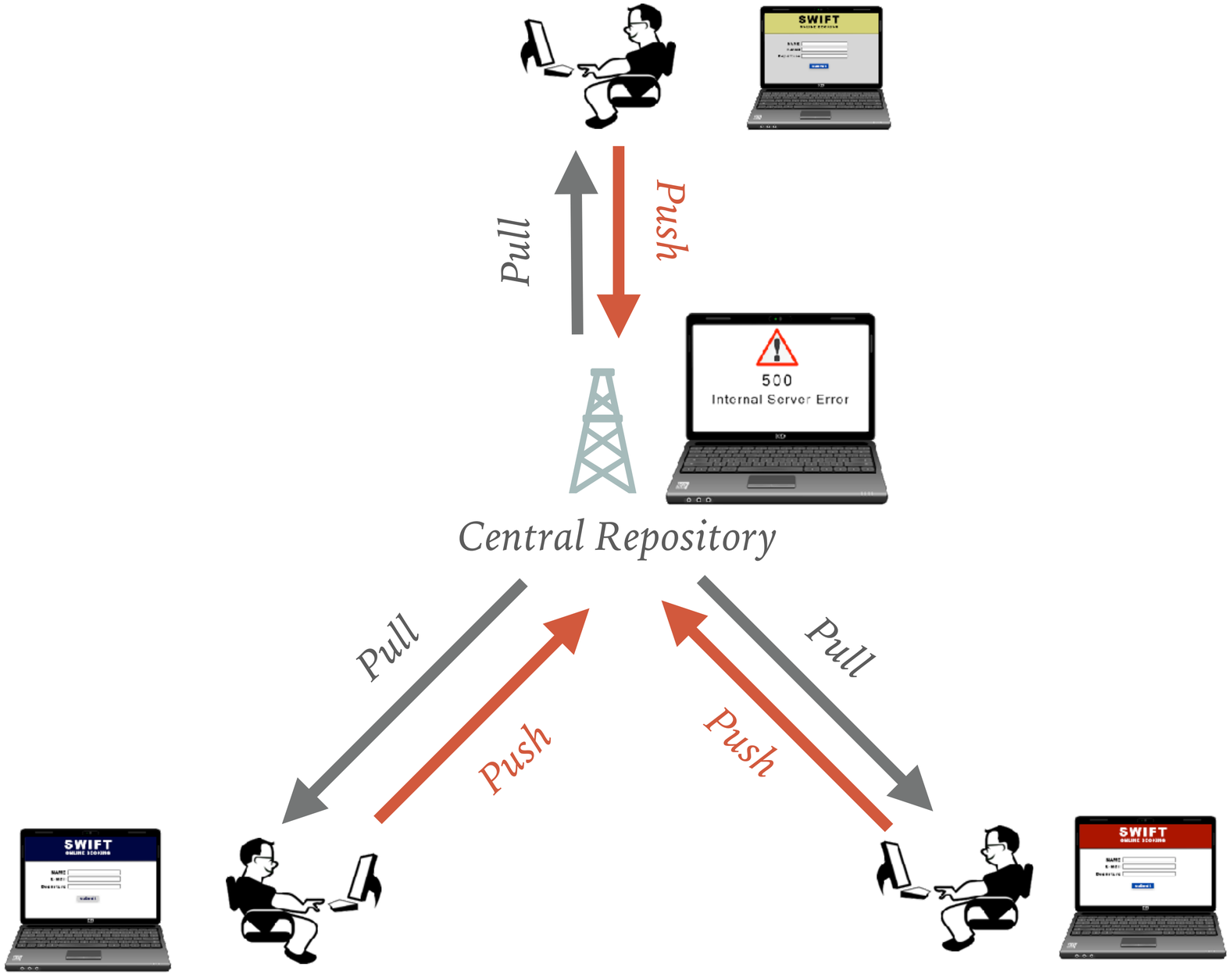
SECTION 06

CI/CD with Jenkins

THE NEED FOR CONTINUOUS INTEGRATION

- The best way to understand CI is to consider an example. The project of this class is a very basic and simple booking application. It is a two-page app, one for taking the booking request and the other for adding the record to the database and printing a confirmation to the user. The simplicity here is for demonstration purposes.
- A real-world booking application would be much more complicated that this. It will have a module for handling requests, another for the billing, a third for contacting the airline company's application and so on.
- To develop an application like this, you need a team (or several teams) of developers, all working simultaneously to finish the iteration (refer to section 01 for more information about Agile development and application iterations).
- Let's say developer A, B, and C each grab a copy of the application source code from the central repository. Each one makes changes to the source code to fix a bug or add a new feature.
- Subsequently, they push their changes to the master repository to update the application code.

TRADITIONAL DEVELOPMENT (NO CI)



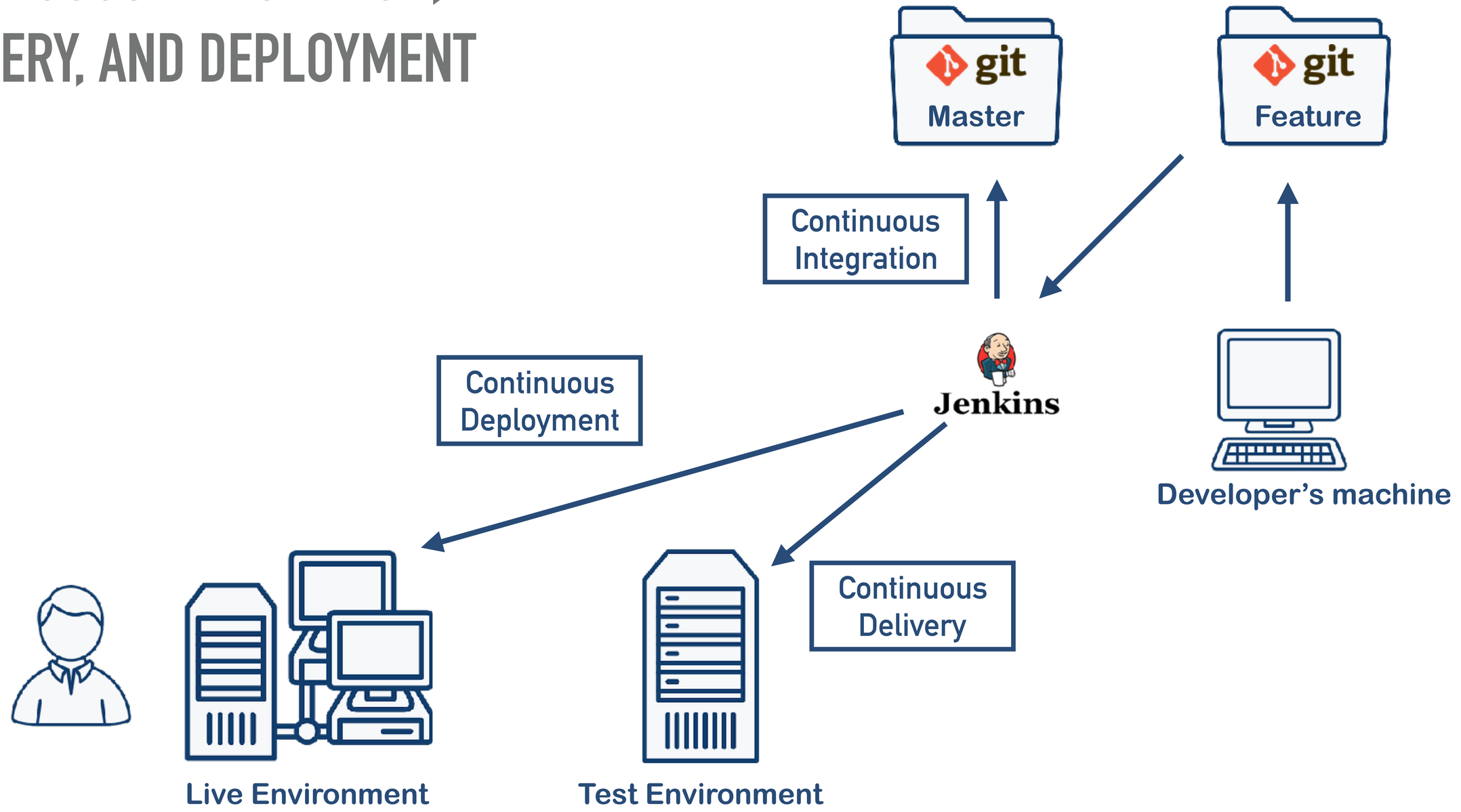
WHAT DOES CI OFFER?

- Using continuous integration, you ensure that every single unit of the application being developed is tested the moment it is committed to the central repository.
- If the application broke, behaved differently, or a bug was spotted, the responsible person will be contacted to take the necessary action.
- In CI, the application should not stay overnight in a non-stable state. This means that any spotted bugs after the commit must be resolved as soon as possible.
- This guarantees that:
  - The time to fix bugs will kept to a minimum as every minor failure will be solved early enough before causing more severe bugs.
  - Developers will always be able to pull a *clean* copy of the application from the repository; as any failing code will not be allowed to be pushed to the central repository.
  - No human intervention exists in the operation. Special tools are designed for this like Jenkins.combined with a test framework like PHPUnit.

CONTINUOUS INTEGRATION, DELIVERY, DEPLOYMENT, WHAT'S THE DIFFERENCE?

- Continuous integration: is the one with least concern as it has the least automation. CI stops when the committed code is tested and integrated successfully to the master branch.
- Continuous delivery: automation is a little more here as it adds upon the work of CI and delivers the application code from the master branch to the non-production servers (instead of having to pull the changes manually from the repository).
- Continuous deployment: this is where full automation is achieved. Code is deployed the application code (or binary artifacts) to the live production servers to be consumed by end users.

CONTINUOUS INTEGRATION, DELIVERY, AND DEPLOYMENT



REQUIREMENTS AND BEST PRACTICES

- CI/CD helps speed up software development iterations. This only helps finish the development process faster, but will also make adding new features and fixing bugs much more frequent. Major companies (for example, Facebook), release new versions of their application twice per month, while others will do the same once per year or may be more.
- But just having a CI/CD tool in place is not enough. Developers must make more commits that cover less changes. That is, code the application in testable chunks and write unit tests for every chunk. Changes must be pushed at least once per day.
- A revision control system must be used like Git.
- A testing framework must be installed to run various types of automated tests on the changes made to the code.

JENKINS INSTALLATION

- Jenkins is one of the most easy-to-install DevOps tools. It needs Java runtime environment as the only prerequisite.
- It can be installed in four different ways:
  - Using the package manager
  - Using a Docker container
  - Using the WAR file
  - Using a Java container like Tomcat, WebLogic, WebSphere…etc.