Continuous Integration

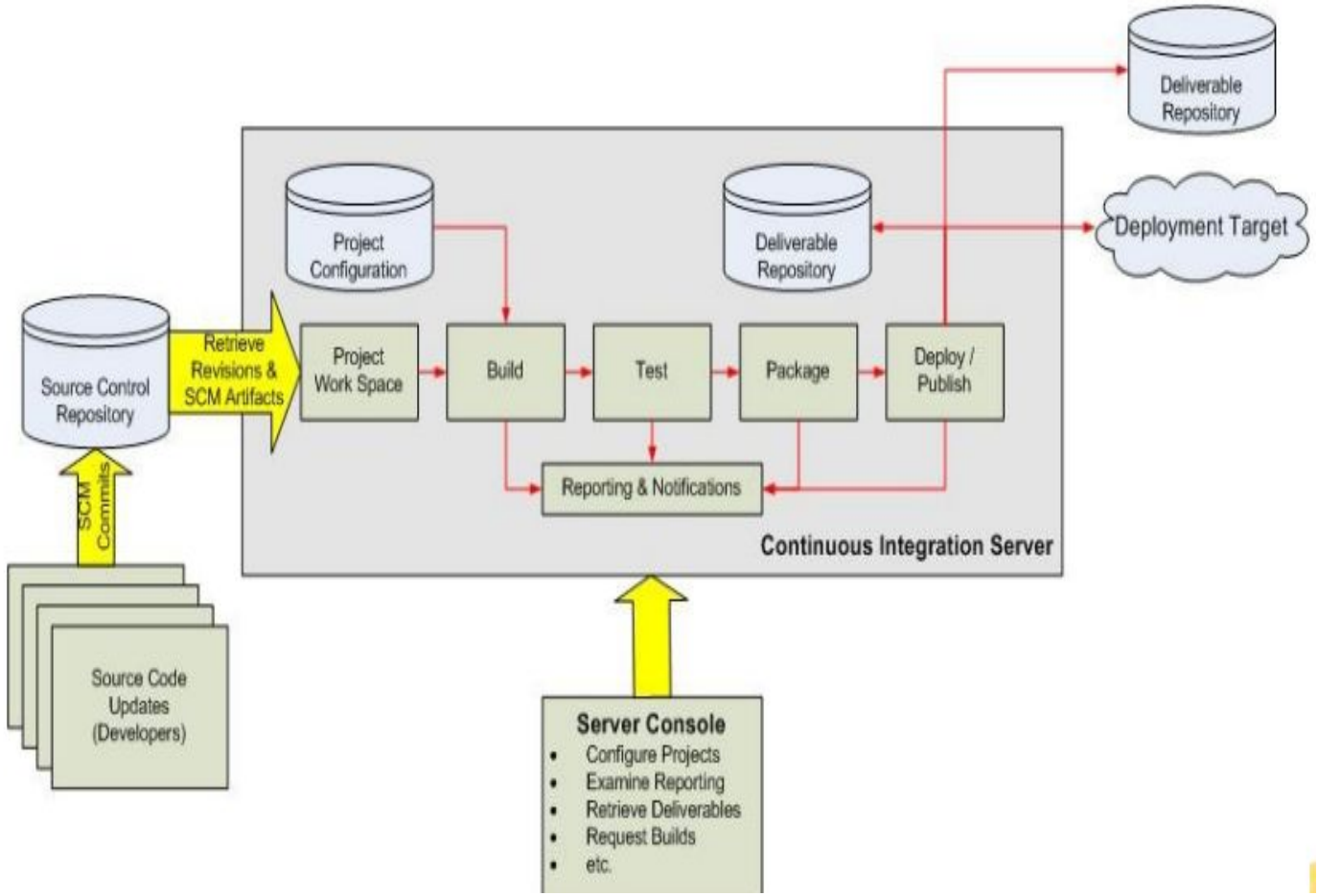
Continuous integration (CI) is **the practice of automating the integration of code changes from multiple contributors into a single software project**. It's a primary DevOps best practice, allowing developers to frequently merge code changes into a central repository where builds and tests then run.

Example for Continuous Integration

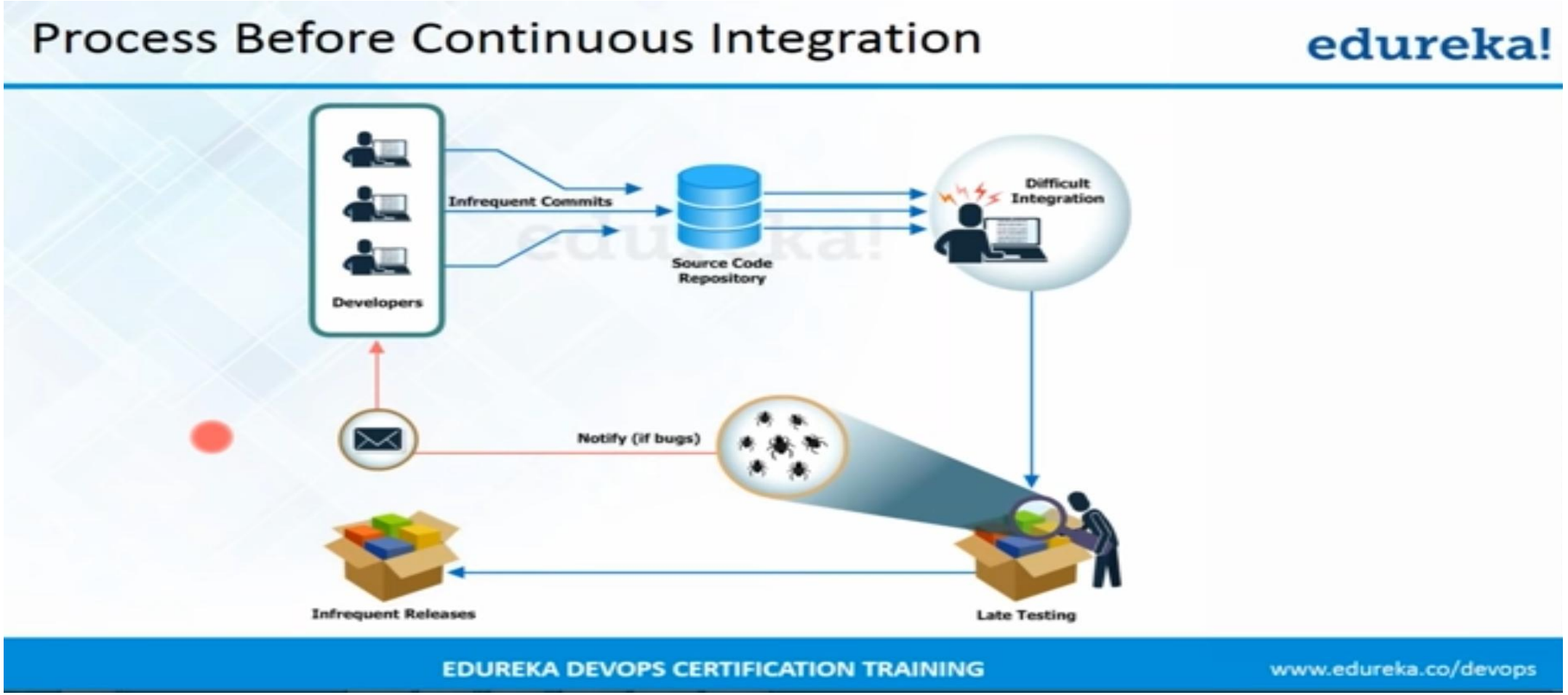
committing all your application code in a single repository!

Overview of Continuous Integration

1. Source code can be provided to the source control repository(e.g. GitHub)
2. Then the program is provided to CI server.
3. In CI server the program is built and deployed as per project configuration.
4. In the built and deployment of each program the status have been reported and notified.
5. The server console examines the report, retrieve deliverables, request builds.
6. The deliverable target is stored in database of CI server.



Why do we need CI(Continuous Integration)?



Continuous Integration?

CI is one of the key components of DevOps automation.

Benefits of CI are:

- Making it easier to fix Bugs
- Reducing Project Risk
- Improving Software Quality
- Increasing Productivity

Difference between CI and CD?

WHAT IS JENKINS:

Jenkins is an open source continuous integration/continuous delivery and deployment (CI/CD) automation software DevOps tool written in the java programming language. It is used to implement CI/CD workflows, called pipelines.

Pipelines automate testing and reporting on isolated changes in a larger code base in real time and facilitates the integration of disparate branches of the code into a main branch. They also rapidly detect defects in a code base, build the software, automate testing of their builds, prepare the code base for deployment (delivery), and ultimately deploy code to containers and virtual machines, as well as bare metal and cloud servers. There are several commercial versions of Jenkins. This definition only describes the upstream open source project.

HISTORY OF JENKINS:

Jenkins is a fork of a project called Hudson, which was trademarked by Oracle. Hudson was eventually donated to the Eclipse Foundation and is no longer under development. Jenkins development is now managed as an open source project under the governance of the CD Foundation, an organization within the Linux Foundation.

PLUGINS:

A Plugin is an enhancement to the Jenkins system. They help extend Jenkins capabilities and integrated Jenkins with other software. Plugins can be downloaded from the online Jenkins Plugin repository and loaded using the Jenkins Web UI or CLI. Currently, the Jenkins community claims over 1500 plugins available for a wide range of uses.

WHAT IS JENKINS USED FOR:

Automation, including CI/CD and test automation, is one of the key practices that allow DevOps teams to deliver “faster, better, cheaper” technology solutions.

So, Jenkins has become an indispensable tool in helping them achieve those goals. Jenkins has been a key enabling technology that is increasingly helping

DevOps practices gain widespread adoption in many organizations around the world.

1. Jenkins lowers the Effort of repeated coding.
2. Integration of Individual Jobs.
3. Synchronization with Slack.
4. Effortless Auditing.
5. Greater data support for project management.
6. Manual Tests option.
7. Increased Code Coverage.
8. Code deployment to Production.
9. Avoid Broken Code during shipping.
10. Decrease Code Review Time.

How Does Jenkins Work?

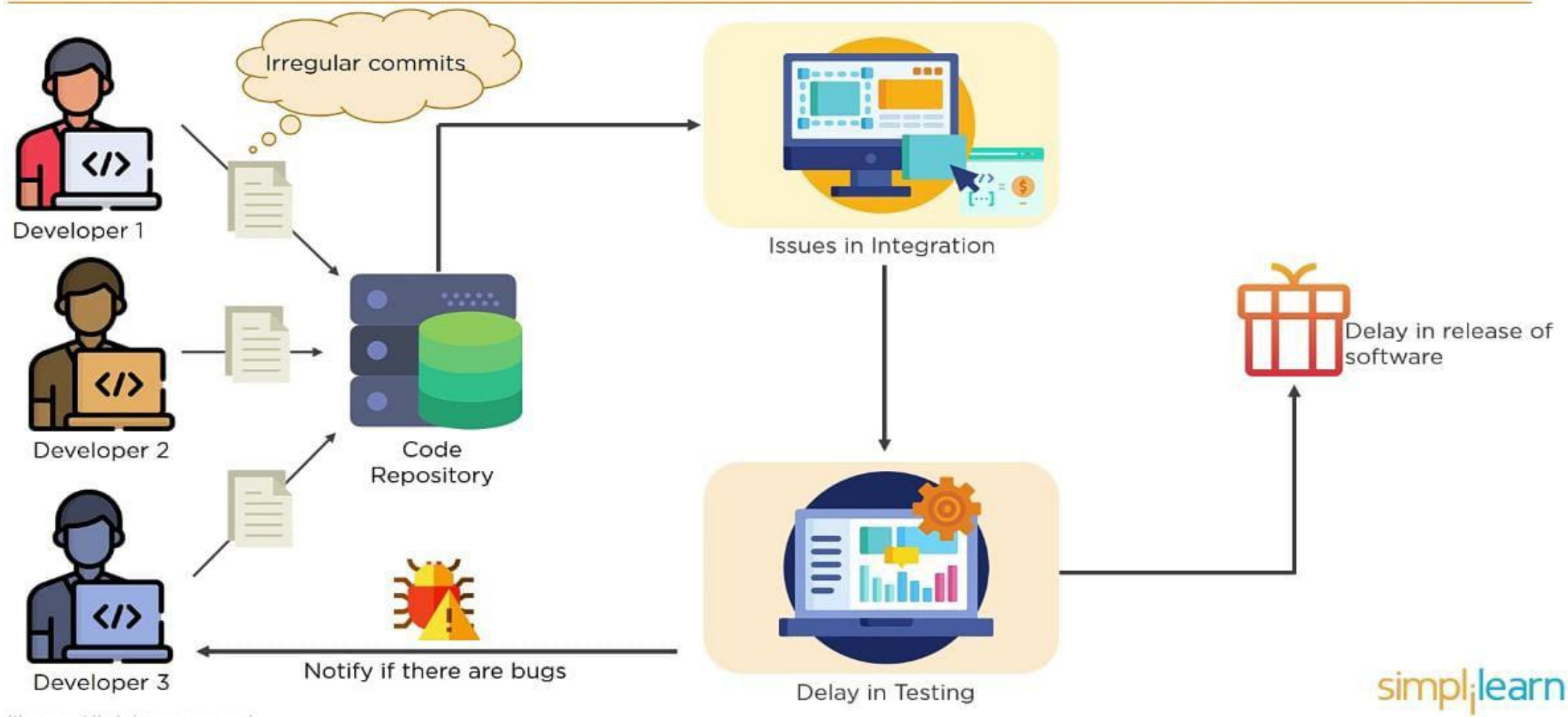
- Jenkins triggers a build upon every commit to the source code repository, typically to a development branch.

- Jenkins can be configured to run an initial suite of unit tests to ensure that the commit did not “break the build”. If the tests do not pass, the developer can be immediately notified to take corrective action. This puts to rest the question of “Who broke to build?” as it is easy to determine which commit caused the build to fail. If all the unit tests pass, then the build pipeline can proceed to the next phase with integration tests which typically take longer to run.

- Jenkins provides the ability to run a build in parallel across multiple machines to minimize the total amount of time it takes to complete many of these activities. Finally, Jenkins can deploy the build to an environment that allows for any needed user acceptance testing (UAT) before releasing it into production. These simplified steps encompass the spirit of a continuous integration (CI) environment.

To reach the holy grail of continuous delivery (CD), these UAT tests can be automated as well using a tool like Selenium, where if those tests pass, the code can be merged into the master branch were a “golden” build can be created and deployed directly into production without manual intervention. Companies that have reached the continuous delivery milestone can deploy to production many times a day, such as Amazon, Facebook, and Google.

BEFORE JENKINS



AFTER JENKINS

- The code is built and test as soon as Developer commits code.
- The code is built immediately after any of the Developer commits.
- Easy to detect whose code caused the built to fail.
- The development cycle is fast.
- New features are more readily available to users.

Moving Forward With Jenkins

Looking towards the future, Jenkins has found a sweet spot in many DevOps environments. Some have even called it “the engine of DevOps”. Jenkins can run in the cloud (AWS, Google, Azure, IBM, etc.), and leveraging technologies like Docker and Kubernetes give Jenkins additional flexibility, speed, and reliability to meet the demands of today’s modern cloud-native, microservices-based applications. The future looks very bright for Jenkins as a key enabling technology for CI/CD pipelines for companies around the world.