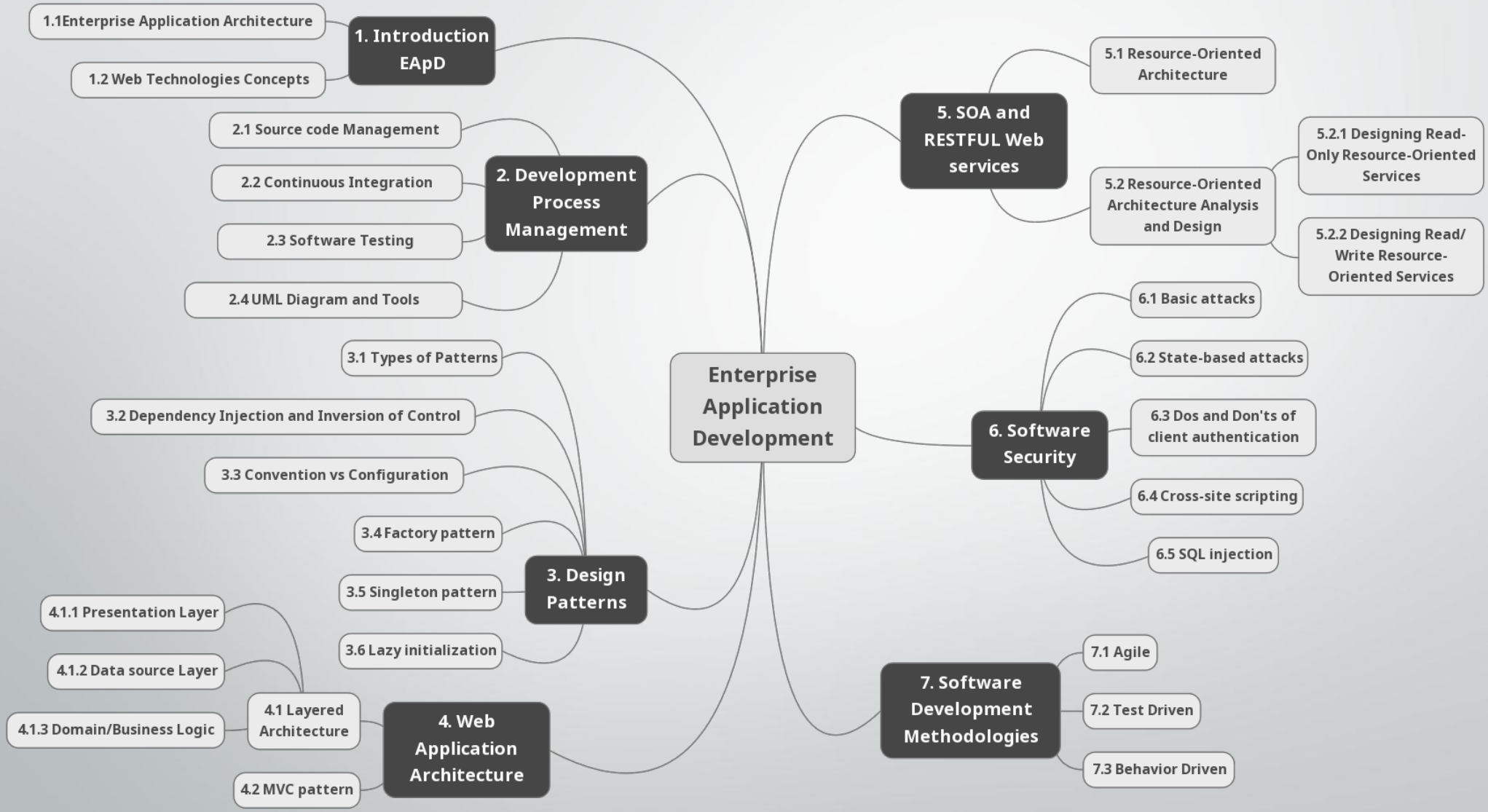
# Enterprise Application Development

[ BE SE-7th Semester ]

**Nepal College of Information Technology**

POKHARA UNIVERSITY

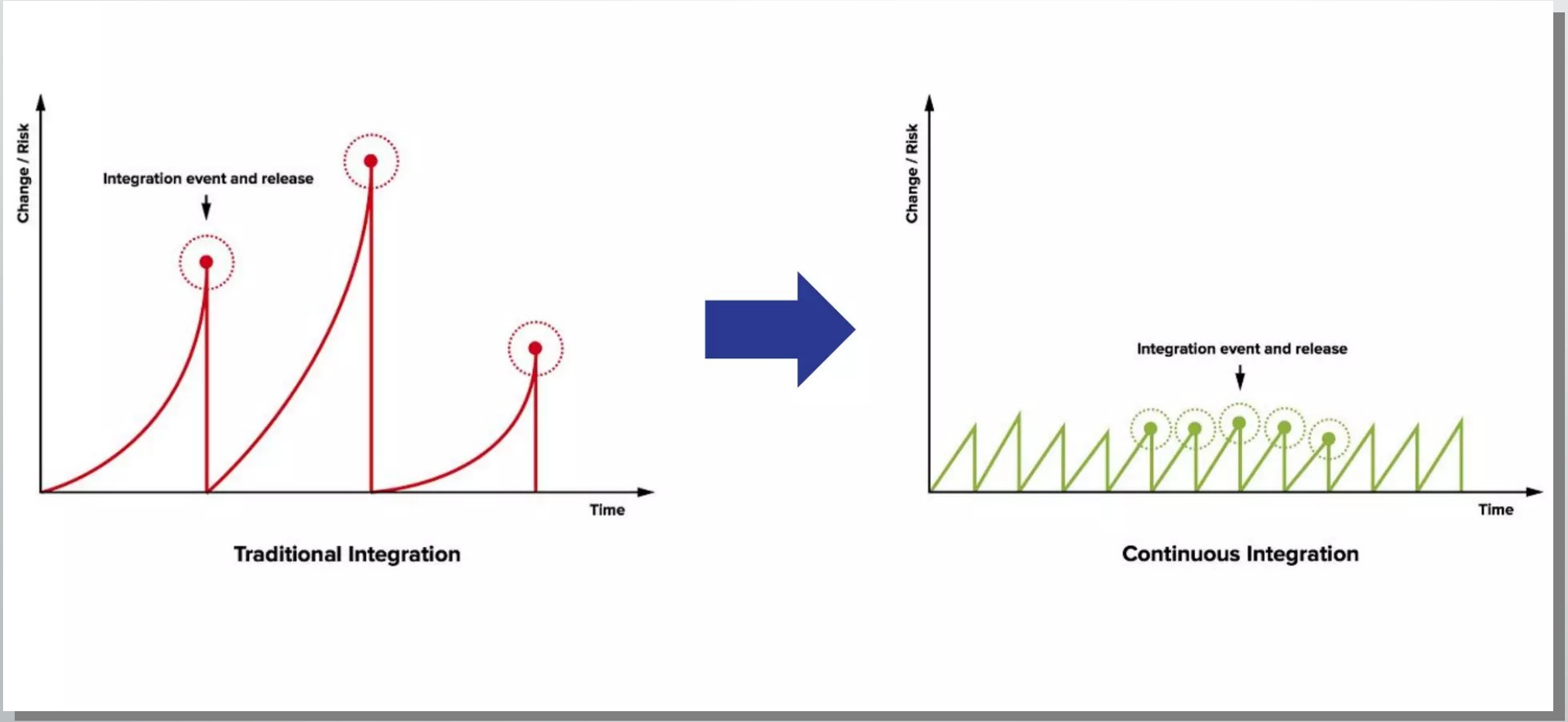
# Enterprise Application Development



# 2.2 Continuous Integration

- **Continuous Integration** (**CI**) involves producing a clean build of the system several times per day, usually with a tool like Cruise Control, which uses Ant and various source-control systems.
- **Agile** teams typically configure **CI** to include automated compilation, unit test execution, and source control **integration**.
- **Continuous integration** (**CI**) is a software engineering practice in which isolated changes are immediately tested and reported on when they are added to a larger code base.
- **Continuous integration** software **tools** can be used to automate the testing and build a document trail.

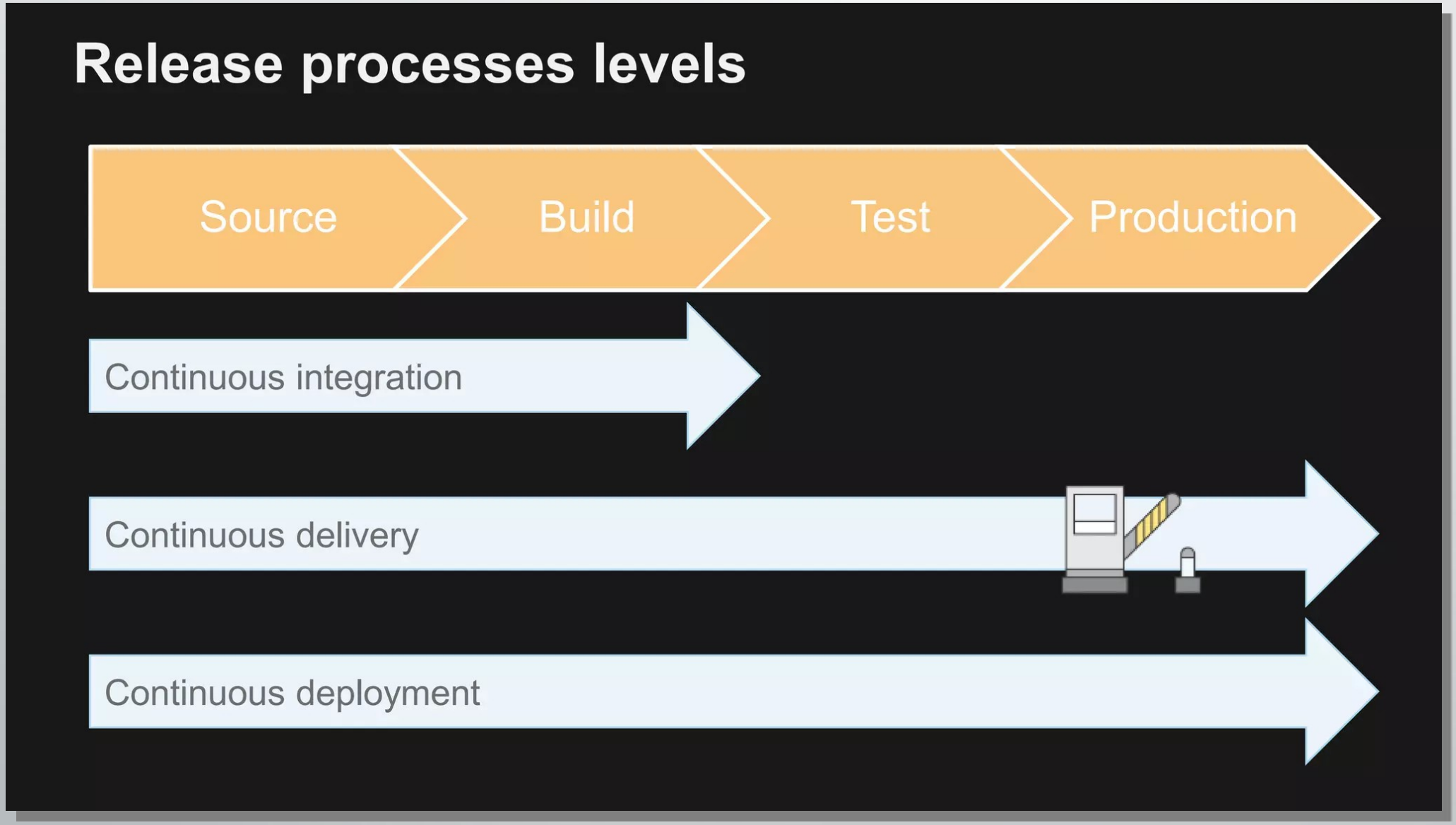
# Traditional Vs Continuous integration



# 2.2 Continuous Integration

- Continuous Integration is a software development practice of performing software integration frequently...several times a day, in fact.
- Ideally, your software application or system should be built automatically after each commit into a shared version control repository.
- Upon each successful build, the system integrity should be verified using automated tests that cover if not all, then at least most of the functionality.
- If some tests fail, the developer responsible is notified instantly and the problem can be identified and solved quickly.
- Using this approach, you can deliver working and reliable code to the customer much faster, while also mitigating the risk of releasing unstable, buggy software to your users.

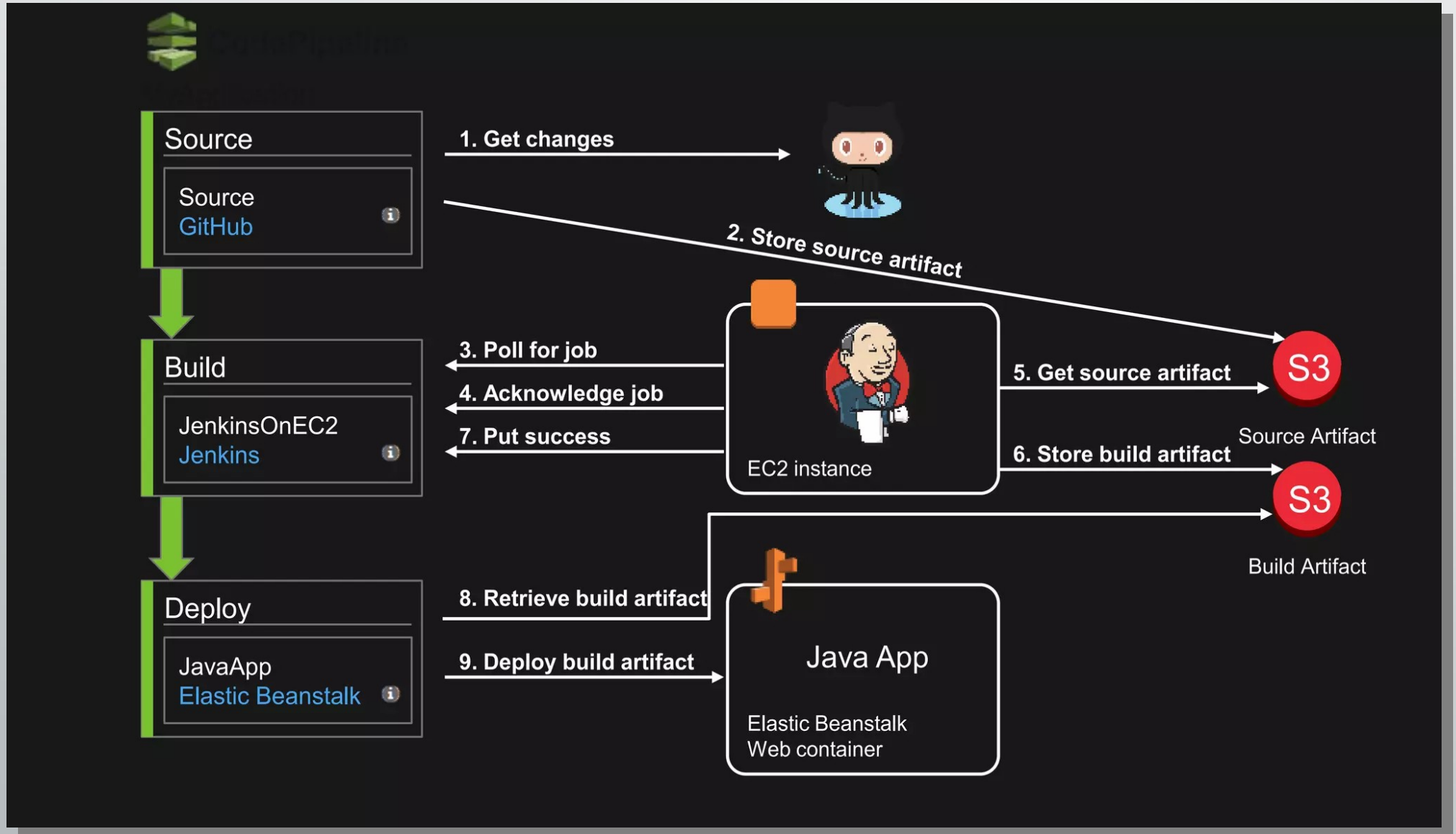
# 2.2 Continuous Integration



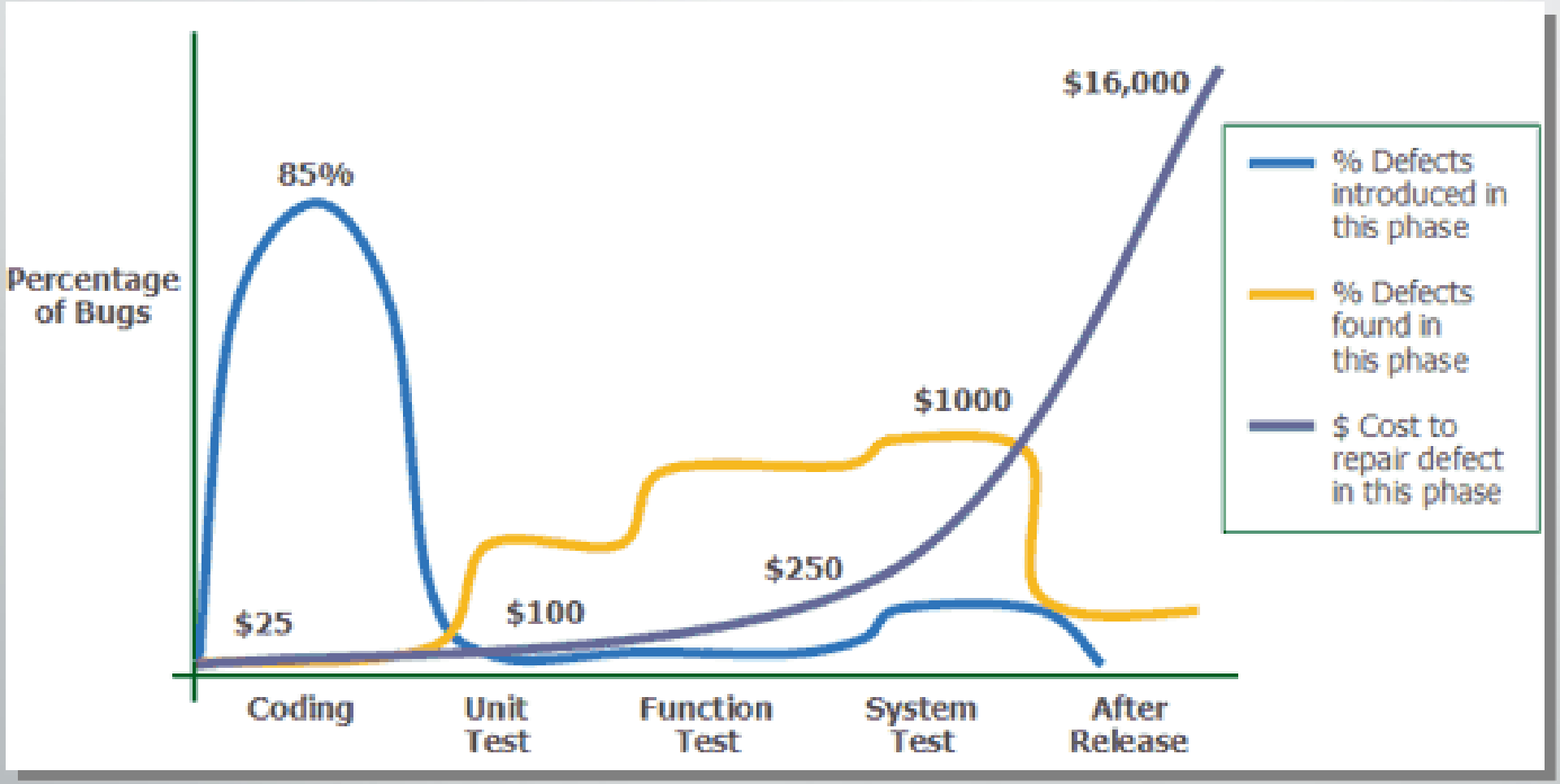
# 2.2 Continuous Integration

- Jenkins is an open-source CI tool written in Java.
- It originated as the fork of Hudson when the Oracle bought the Sun Microsystems.
- Jenkins is a cross-platform CI tool and it offers configuration both through GUI interface and console commands.
- Continuous integration is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run.
- With continuous integration, developers frequently commit to a shared repository using a version control system such as Git.

# Continuous Integration with Jenkins



# Defect and Cost to repair



# Benefits of CI

- Easy and quicker integration i.e. no long and tense integrations
- Increase visibility which enables greater communication
- Catch issues early and solve them early
- Spend less time debugging and more time adding features
- Proceed in the confidence by building on a solid foundation
- Avoids Confusions: Stop waiting to find out if the code works
- Reduce integration problems allowing the software delivery within schedule

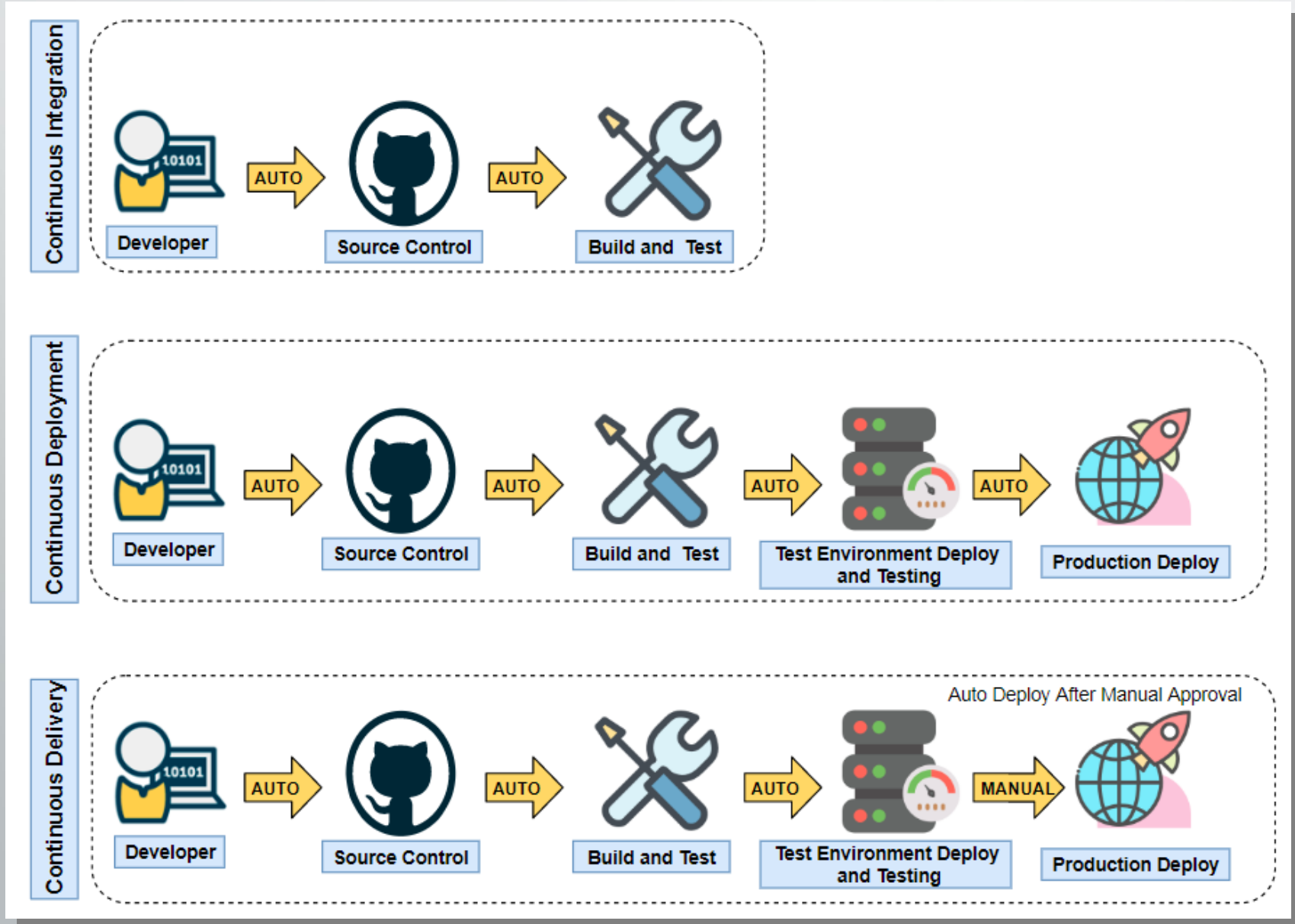
# The Practices

- Maintain a single source repository
- Automate the build
- Make your build self-testing
- Every commit should build on an integration machine
- Keep the build fast
- Test in a clone of the production environment
- Make it easy for anyone to get the latest executable
- Everyone can see what’s happening
- Automate deployment

# Continuous Integration Process

- Developers check out code into their private workspaces.
- When done, the commit changes to the repository.
- The CI server monitors the repository and checks out changes when they occur.
- The CI server builds the system and runs unit and integration tests.
- The CI server releases deployable artifacts for testing.
- The CI server assigns a build label to the version of the code it just built.
- The CI server informs the team of the successful build.
- If the build or tests fail, the CI server alerts the team.
- The team fixes the issue at the earliest opportunity.
- Continue to continually integrate and test throughout the project.

# Continuous Integration Process



# Team Responsibilities

- Check in frequently
- Don’t check in broken code
- Don’t check in untested code
- Don’t check in when the build is broken
- Don’t go home after checking in until the system builds
- Many teams develop rituals around these policies, meaning the teams effectively manage themselves, removing the need to enforce policies from on high.

# Continuous Deployment

- Continuous Deployment is closely related to Continuous Integration and refers to the release into production of software that passes the automated tests.
- Essentially, “it is the practice of releasing every good build to users,” explains Jez Humble, author of Continuous Delivery.
- By adopting both Continuous Integration and Continuous Deployment, we not only reduce risks and catch bugs quickly, but also move rapidly to working software.
- With low-risk releases, we can quickly adapt to business requirements and user needs. This allows for greater collaboration between outputs and delivery, fuelling real change in our organization, and turning our release process into a business advantage.

# Enterprise Application Development