Continuous Integration: Concepts and Tools

Lecture 2b

Agenda

- Continuous Integration (CI)
- The Steps and Key Practices of Continuous Integration
- Benefits of Continuous Integration
- Limitations of Continuous Integration
- Some CI Tools
- Related Concepts

Terminology

- **Continuous**—Technically, *continuous* means something that, once started, never stops. However, continuous, in the context of CI, is more like *continual (happening repeatedly)*.
- **Integration**—The act of combining separate source code artifacts together to determine how they work as a whole.

Continuous Integration

Continuous Integration (CI)

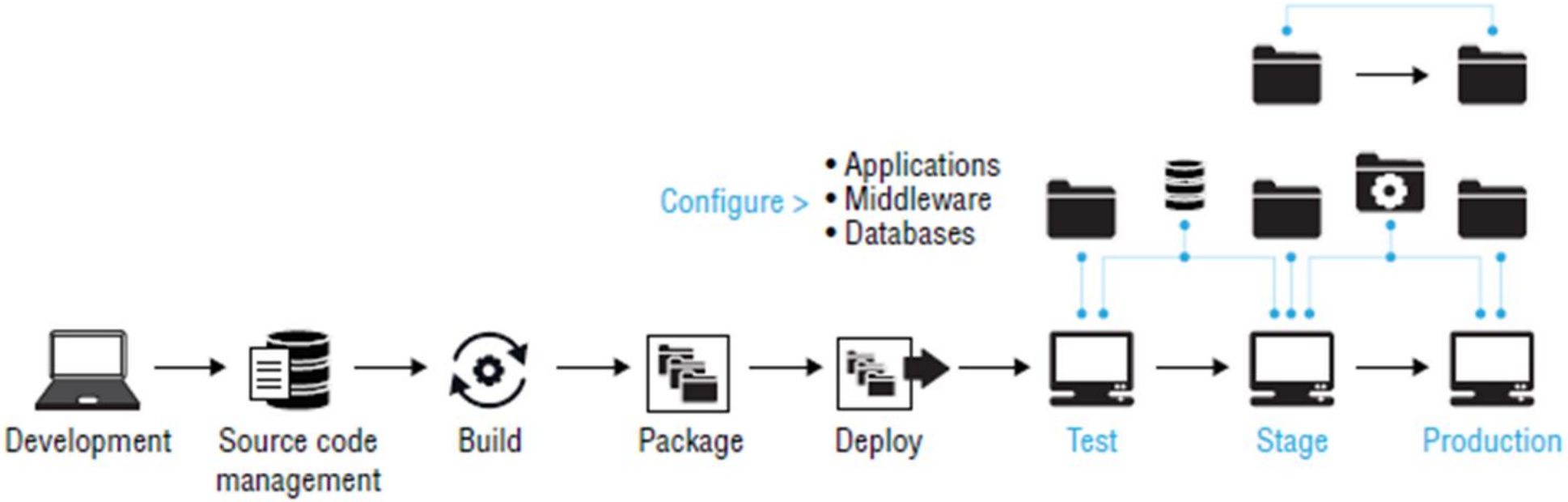


Figure 1-3: A delivery pipeline

Continuous Integration (CI)

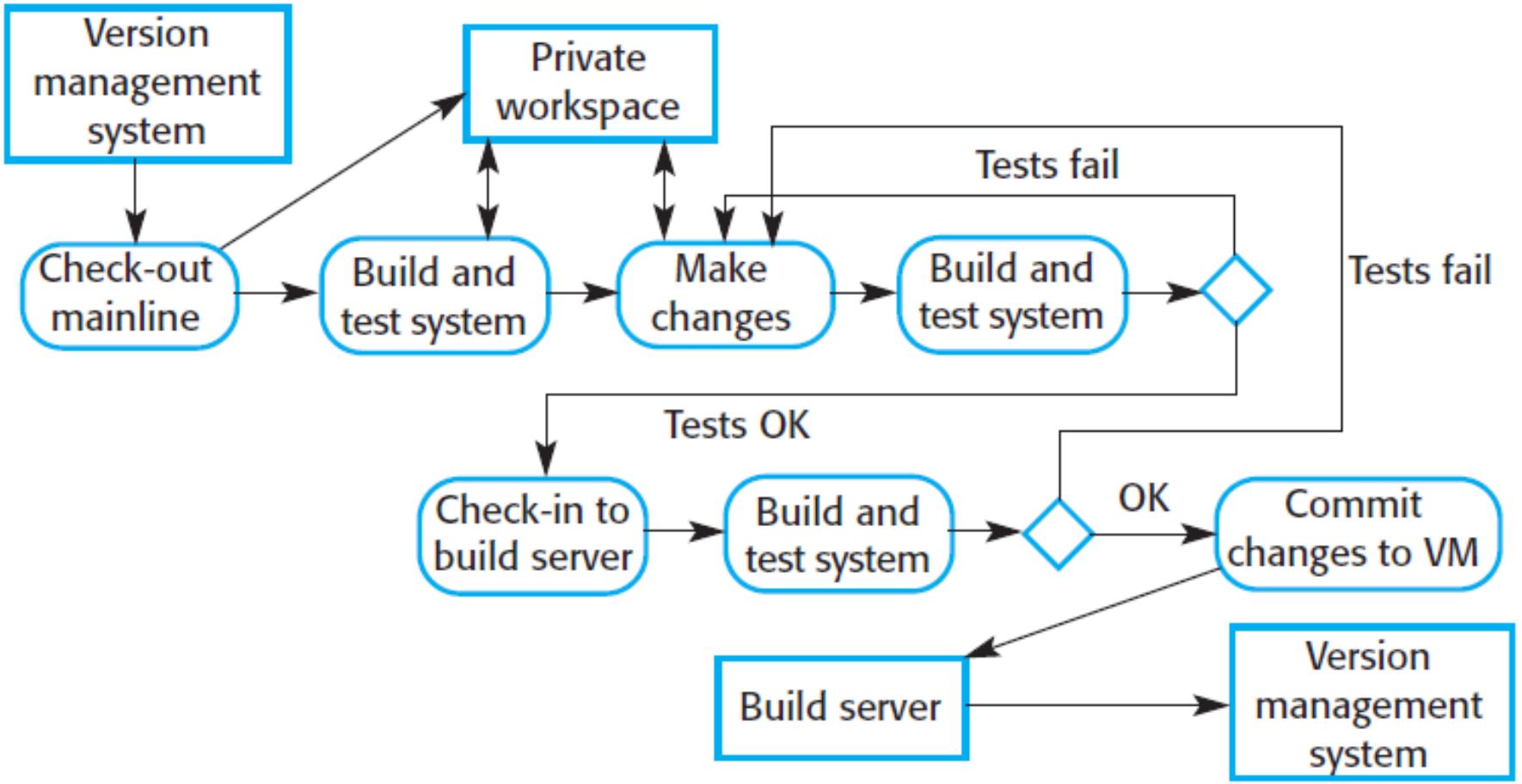
- Continuous Integration is a software development **practice** where members of a team **integrate** their work **frequently**, usually each person integrates at least daily - leading to **multiple integrations per day**.
- **Each integration** is verified by an **automated build** (including test) to **detect integration errors** as quickly as possible.

The Steps and Key Practices of Continuous Integration

The Steps in Continuous Integration

1. Extract the **mainline system** from the version control system **into** the developer's **private workspace**.
2. **Build the system and run automated tests** to ensure that the built system passes all tests. **If** not, the **build is broken**, and you should inform **whoever** checked in the **last baseline system**. He or she is **responsible for repairing** the problem.
3. **Make the changes** to the system components.
4. Build the system in a private workspace and **rerun system tests**. **If** the tests **fail**, **continue editing**.

Steps in Continuous Integration



Steps in Continuous Integration

5. Once the system has passed its tests, check it into the build system server **but do not commit it as a new system baseline** in the VC system.
6. Build the system on the build server and run the tests. Alternatively, if you are using Git, you can **pull recent changes from the server to your private workspace**. You need to do this in case others have modified components since you checked out the system. If this is the case, check out the components that have failed and edit these so that tests pass on your private workspace.
7. **If the system passes its tests** on the build system, **then commit the changes you have made as a new baseline** in the system mainline.

Key Practices of Continuous Integration

- **Maintain a single** source **repository**.
- **Automate the build**.
- Make your **build self-testing**.
- Everyone **contributes** to the mainline **every day**.
- Every contribution should build the mainline on an **integration machine**.
- **Fix** broken builds **immediately**.

Key Practices in Continuous Integration

- Keep the **build fast**.
- **Test in a clone** of the **production** environment.
- Make it **easy** for anyone to **get the latest executable**.
- **Automate deployment**.

Benefits of Continuous Integration

Benefits of Continuous Integration

- **Reduce risks**: integrating many times a day facilitates a sooner detection of defects.
- **Reduce repetitive manual processes**: the processes will run every time a commit occurs in the version control repository.
- **Generate deployable software at any time and at any place**: you make small changes to the source code and integrate these changes with the rest of the code base on a regular basis.
- **Enable better project visibility**: a CI system can provide just-in-time information on the recent build status and quality metrics.
- **Establish greater confidence in the software product from the development team**: with every build, your team knows that tests are run against the software, that project coding and design standards are met, and that the result is a functionally testable product.

Limitations of Continuous Integration

Limitations of Continuous Integration

- **If the system is very large**, it may take a long time to build and test, especially if integration with other application systems is involved. It may be impractical to build the system being developed several times per day.
- If the **development platform is different from the target platform**, it may not be possible to run system tests in the developer's private workspace. There may be differences in hardware, operating system, or installed software. Therefore, more time is required for testing the system.

Some CI Tools

Some CI Tools

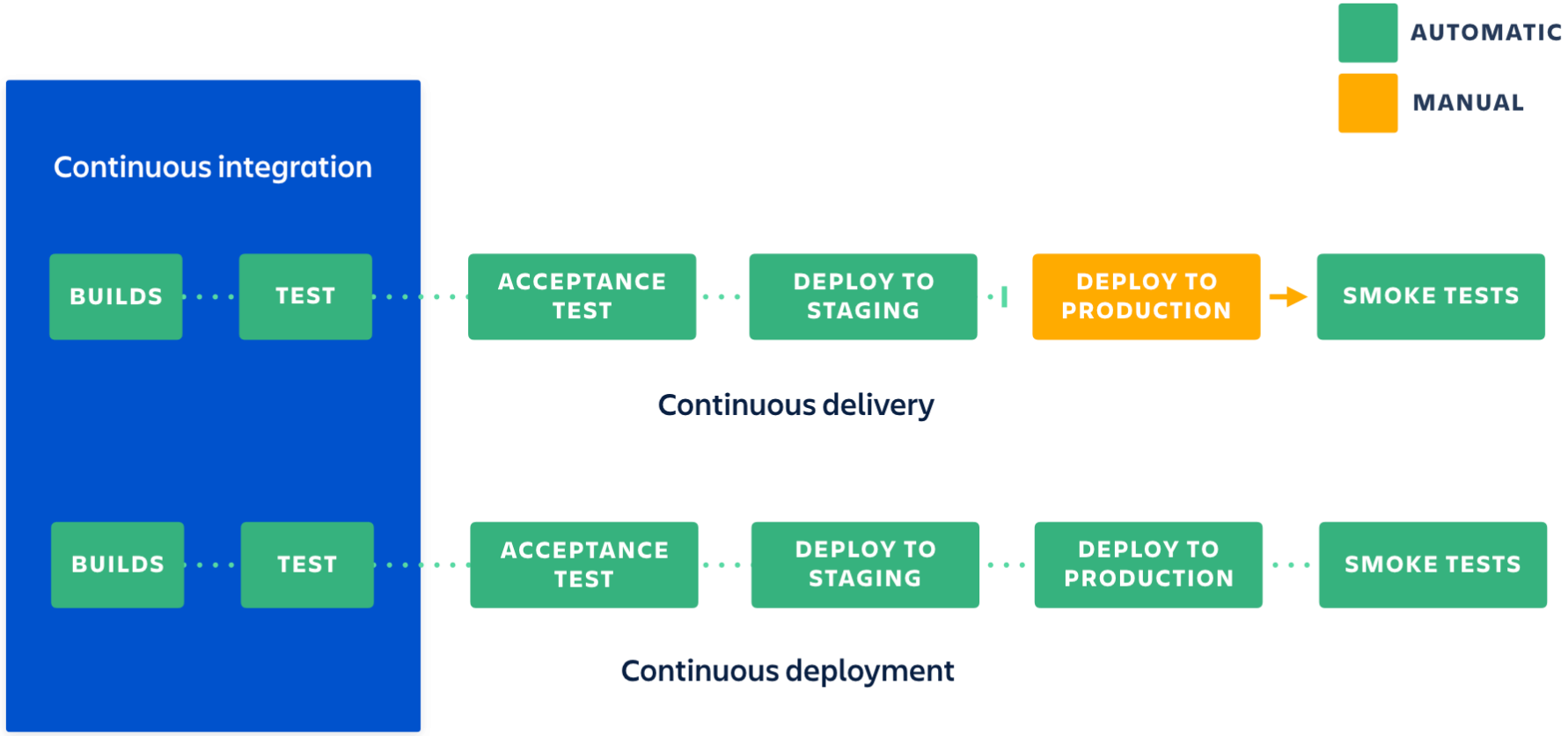
- **Jenkins** is a cross-platform CI tool and it offers configuration both through GUI interface and console commands.
  - Jenkins is an open-source CI tool written in Java.
- **Buddy** is a modern and intuitive CI tool. By keeping the interface simple Buddy keeps the time you need to invest in learning the tool to the minimum.
  - Free trial/paid plans
- **Bamboo** is a continuous integration build server which performs - automatic build, test, and releases in a single place.
  - Paid with a free trial
- **TeamCity** is the mature CI server, coming from the labs of the JetBrains company
  - Free for 3 agents and 100 build configurations and paid enterprise edition

Related Concepts

Continuous Delivery

- **Continuous delivery** is an extension of continuous integration to make sure that you can **release** new changes **to your customers quickly** in a sustainable way.
- This means that **on top of having automated your testing**, you **also have automated your release process** and you can deploy your application at any point of time by clicking on a button.
- With continuous delivery, you **can decide** to **release** daily, weekly, fortnightly, or whatever suits your business requirements.

Continuous integration, delivery and deployment



Courtesy: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

Continuous Deployment

- **Continuous deployment** goes o**ne step further** than continuous delivery. With this practice, every change that passes all stages of your production pipeline is released to your customers.
- There's **no human intervention**, and only a failed test will prevent a new change to be deployed to production.
- Developers can focus on building software, and they see their work go live minutes after they've finished working on it.

Next On

Take Away Points

- Continuous Integration
  - What are the main steps?
  - What are the key practices?
  - Which are the benefits of this approach?
  - Is it always possible to implement this approach to system building?
- Continuous Delivery vs. Continuous Deployment
  - What is the subtle difference?

Further Reading

- Ian Sommerville, “Software Engineering”
  - Chapter 25: Configuration Management
- Paul Duval, “Continuous Integration”
  - Chapter 2: Introducing Continuous Integration
- Martin Fowler, “Continuous Integration”
  - URL: https://martinfowler.com/articles/continuousIntegration.html

Next Lecture

- Coding Practices