BUY IT, USE IT, BREAK IT, FIX IT

CONTINUOUS INTEGRATION

SAMUEL PETOVSKY

OUTLINE

- Introduction
- Overview
- Motivation
- Best Practices
- Build Servers
- Real Life Example

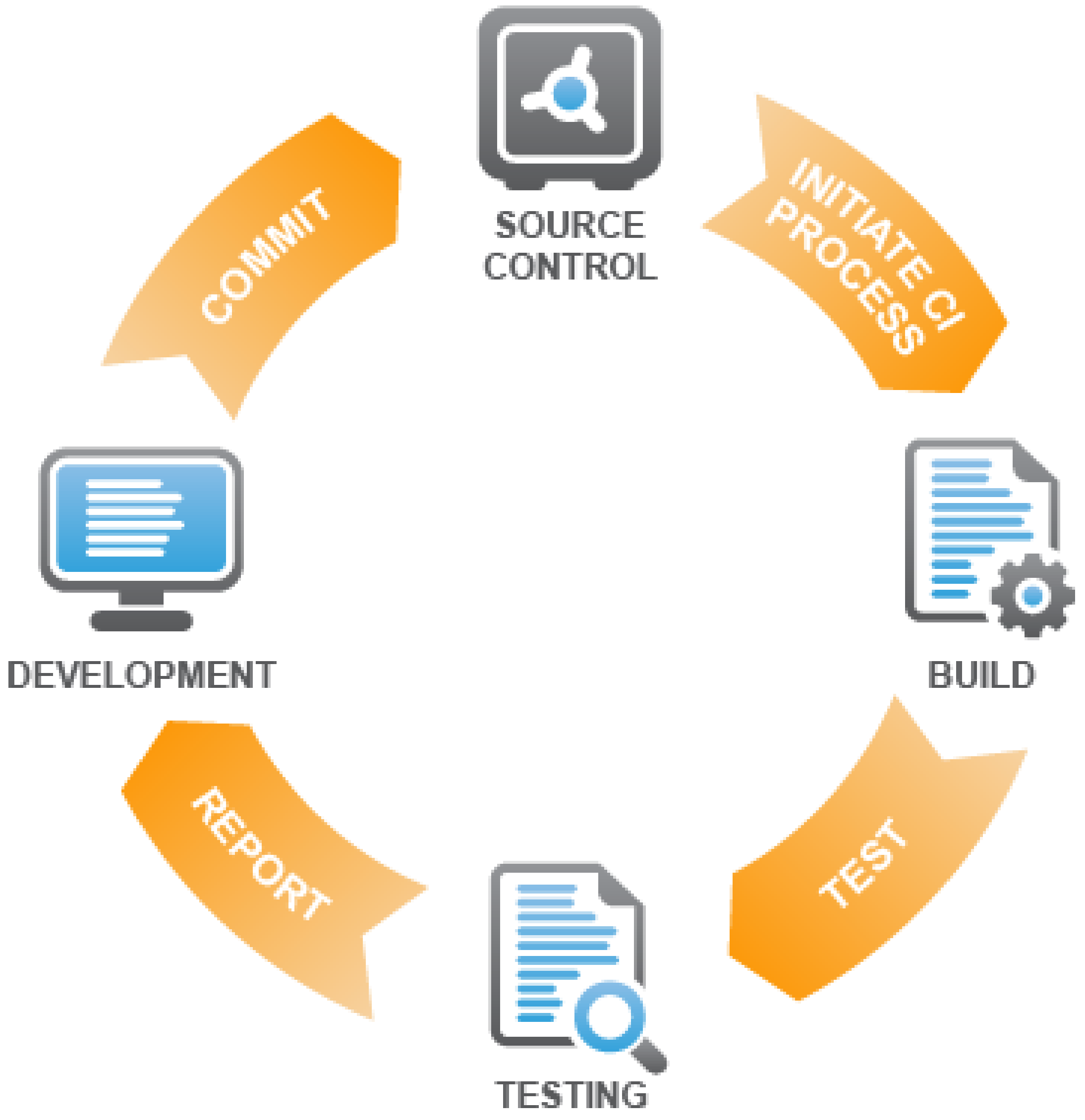
INTRODUCTION

- Software development practice where members of a team integrate their work as often as possible, usually several times a day to prevent „**integration hell**“
- Build automation
- Often combined with automated testing

INTRODUCTION

- Feature toggle instead of branches
- Continuous delivery
- Build servers

OVERVIEW



- Commit it
- Build it
- Test it
- Fix it (if broken)

MOTIVATION

- Switch to continuous deployment has been linked to very concrete and visible financial success (**Linkedin**)
- **Facebook** releases to production twice a day
- **Amazon** makes changes to production every 11.6 seconds
- 8 minutes after you commit code it's live in production (**Google Consumer Surveys**)

BEST PRACTICES

1. **Maintain a Single Source Repository**

- Use Source code management tools (SVN, Git, Mercurial)
- Put all project-related files into repository

BEST PRACTICES

2. Automate the Build

- Involve everything in the build (running pre-installation scripts, loading database schema, compiling…)
- Ant, MSBuild, Make
- Build servers

BEST PRACTICES

3. Make the Build Self-Testing

- Produce self-testing code
- xUnit tests, Selenium…

BEST PRACTICES

4. Everyone Commits To the Mainline Every Day

- Break the work into small chunks
- It prevents „Integration Hell“
- Issues and conflicts are detected sooner and thus easier to fix

BEST PRACTICES

5. Every Commit Should Build the Mainline on an Integration Machine

- No branches
- Commit build

BEST PRACTICES

6. Fix Broken Builds Immediately

- Fixing the broken build has a highest priority
- Revert to latest stable state

BEST PRACTICES

7. Keep the Build Fast

- Do not include everything in the commit builds
- Use parallelization
- Put more time-consuming tasks into nightly builds instead (static code analysis…)

BEST PRACTICES

8. Test in a Clone of the Production Environment

- The difference between test and production environments can cause troubles.
- Use the same hardware, operating system, database, firewall settings, test on the real data.

BEST PRACTICES

9. Make it Easy for Anyone to Get the Latest Executable

- To help make this work, anyone involved with a software project should be able to get the latest executable and be able to run it

BEST PRACTICES

10. Everyone can see what's happening

- Continuous integration is about communication.
- Everyone should know the state of the mainline build.

BUILD SERVERS

12. Automate Deployment

- CI makes deployment boring.
- Consider an automated rollback.

BEST PRACTICES

- Tools that help with CI
- Build and deployment automation
- Advanced setting of CI cycle (pre and post-build steps, build stages, task parallelization)
- Often offer scalability
- Bamboo, Jenkins, Travis

REAL LIFE EXAMPLE

Q&A

SAMUEL PETOVSKY