Release Engineering

Idaho State University | Computer Science

Isaac Griffith

CS 2263
Department of Informatics and Computer Science
Idaho State University

ROAR

Outcomes

After today's lecture you will be able to:

- Describe and understand the principles of release engineering
- Understand what deployment is and methods for automating it
- Understand the difference between continuous integration, deployment, and delivery
- Understand the basic concepts of configuration management

Inspiration

“If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilisation.” – Gerald Weinberg

Software Release Principles

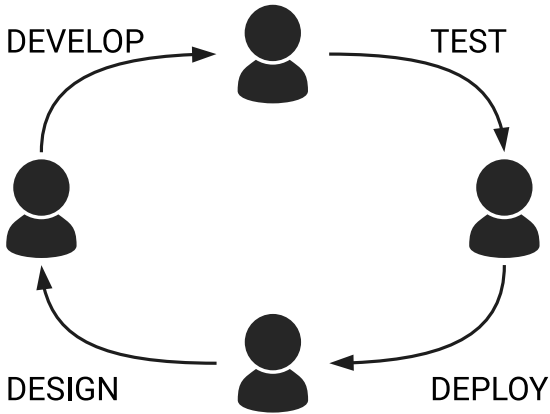
- Should be Fast / Predictable / Repeatable.
- Automate almost everything
- Keep everything in Version Control
- If it hurts, do it more often
- Improve the quality of the Builds. Push the boundaries of test automation so that quality is not compromised
- Done means the software is released and has reached the production

What is Deployment?

- Run the same commands in several hosts
- Install the required software from a central repository so that the end state of every host is the same
- Based on host-type, install the required packages
- End goal is to bring all the hosts of the respective host-type in the same state

What is “Continuous Integration”?

- Building, deploying and testing the application

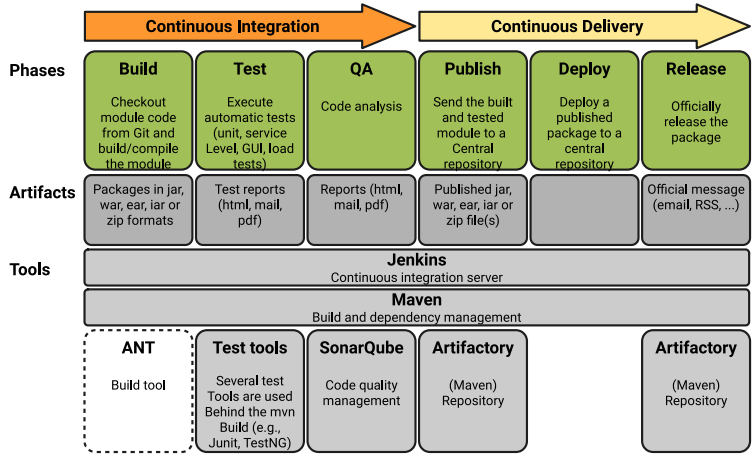


Steps associated with CI

- Integrate all the components at regular intervals
- Binaries built only once and used in all the environments
- Execute Unit Tests before the packages are generated
- Deploy the latest packages onto an INT environment
- Run the automated tests, validate the test results
- Track the Code quality criteria such as static code analysis, test coverage
- Deploying to other environments with a pre-defined frequency
- Repeat the process
- **It's a practice, not a tool**

What is “Continuous Delivery”?

- Building, deploying, testing, promotion of the application to the next environment

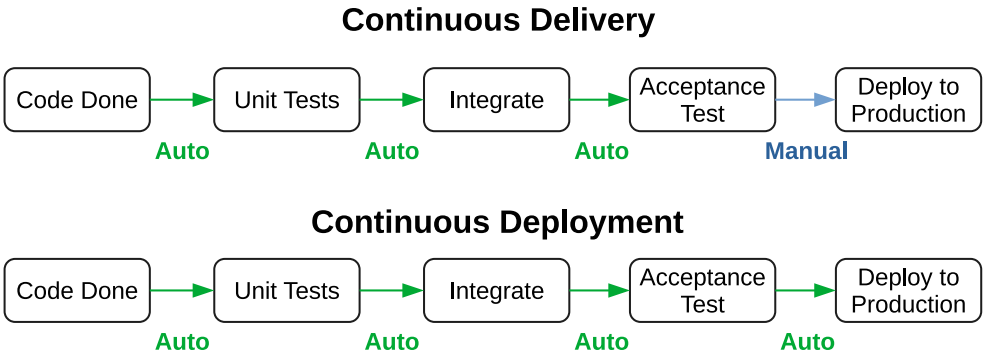


Steps associated with CD

- Steps followed in CI
- Automatic promotion / deployment to other environments (QA, Staging, Performance etc…) upon successful execution of automated tests
- Application is **always ready** to deploy to production through a largely automated process

What is a “Continuous Deployment”?

- Building, deploying, testing, promotion of the application to production.



Release Engineering Principles

- Minimize Cycle Time from check-in to production push
- Everybody is responsible for the Delivery Process
- Continuous Improvement
- Activities such as SCM, Release Planning, Automated Deployment, Acceptance Testing, Performance Testing are **NOT SECONDARY**
- Generates no revenue till its in the hands of end-users.
- Customer Satisfaction through early & continuous delivery of software.

Release Engineering Steps

- SCM (Git, SVN, Clearcase)
- Automated Commit / Component Builds (Generate Binaries)
- Generate Code Quality metrics
- Build Deployment Pipeline
- Automated Acceptance Testing
- Automated Promotion to subsequent environments
- Single click or automated push to production

Main Focus of Release Engineering

- Build -> Deploy -> Test -> Release
- Every single check-in should potentially be in a releasable state

Configuration Management

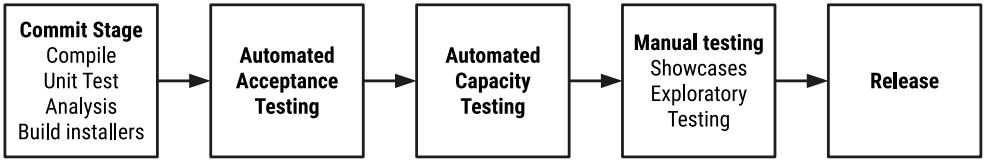
- Using Version Control and commit everything
- Check-in the changes at regular frequency
- Use meaningful messages for every commit
- Manage Dependencies (External Libraries, Components)
- Managing the Infrastructure (Consistent network topology, firewall, OS configuration, patches etc… across all environments)
- Managing the Environments & the tools to be used for the same
- Managing the Change Process

Principles of Managing Application Configuration

- Good understanding of the stage in which the configuration should be injected (Assembly, Deployment, Restart etc…)
- Configuration settings for the application to be in the project's root directory
- Should always be automated and values checked into repository
- Use clear naming convention
- Avoid over-engineering. Keep it as simple as possible
- Ensure that the configuration is tested properly after deployment

Release Engineering Architecture

- Its all about building the “Build & Release” pipeline
- Built using Jenkins or similar CI tools
- Illustration of a “Delivery Pipeline”



Deep Dive into Commit/Component Builds

- Aim:
  - Eliminate builds that are unfit for production
- Steps:
  - compile the code
  - run a set of commit tests
  - run lint based tests or some of the basic static code analysis tasks
  - publish the results
  - once above steps are done, build the package and upload the binary to a centralized repository
  - add tests on an ongoing basis
  - keep a watch on the build execution time and optimize frequently
  - explore the ‘pre-commit’ or ‘pre-flight’ build options provided by some of the CI tools

Commit/component build Anti-Patterns

- Don't checkin new code on a broken build. The only fix that has to go are the changes that fix the broken build
- Always run commit tests locally before the commit
- Always wait for commit tests to pass before next check-in
- Developers who have committed their code are responsible for the build process to succeed
- Never go home on a broken build
- Time-box during every failed build. Give about 10-20 minutes to fix, else rollback the changes
- Don't EVER comment failing tests.
- Take ownership for all the breakages that result from your changes and fix them accordingly

Are there any questions?