CI/CD

Dr. Isaac Griffith

Idaho State University

Inspiration

“There should be two tasks for a human being to perform to deploy software into a development, test, or production environment: to pick the version and environment and to press the “deploy” button.” – David Farley

Outcomes

After today's lecture you will be able to:

- Describe both CI and CD are
- Understand the core CI practices
- Define what makes for good CI/CD
- Understand the benefits of CI/CD

ROAR

What is CI and CD?

- **Continuous Integration**
  - An approach to be continually validating the state of a codebase through automated testing
  - Best achieved through integration with version control
- **Continuous Delivery/Deployment**
  - An approach to regularly deploying artifacts that successfully pass the CI phase to ensure confidence around the deployment

Delivery vs. Deployment

- **Continuous integration, continuous deployment,** and **continuous delivery** are like vectors that have the same direction, but different magnitude
- Their goal is the same: make our software development and release process faster and more robust
- The key difference between the three is in the scope of automation applied

Delivery vs. Deployment

- **Continuous Delivery**
  - Automatically prepare and track a release to production
  - The desired outcome is that anyone with sufficient privileges to deploy a new release can do so at any time in one or a few clicks.
  - By eliminating nearly all manual tasks, developers become more productive
- **Continuous Deployment**
  - Every change in the source code is deployed to production automatically, without explicit approval from a developer.
  - As long as it passes the quality controls

Core CI Practices

CS 3321

ROAR

Single Source Repository

- Single point of truth
- Everyone’s code in the same place
- NOT a branch per developer
- Shared ownership

- **Facilitated by following GitFlow**

Automated Build

- Using the IDE is not automating!
- Using the IDE does not give you a repeatable build
- Use a build tool
- Compile, package, test

- **Facilitated by using Gradle**

Automated Testing

- Not just Unit Tests
- Failing tests fail the build
- Fix it if it’s broken or you’re wasting your time

- **Facilitated by using JUnit, Spock, and Gradle**

Publish Latest Distributable

- Make it easy to get the final product
- Should only be built once so you deploy what's been tested
- Configuration is separate

CI Evolution

CS 3321

ROAR

Commit More Often

- “At least once a day” - aim for at least once an hour
- Needs a small unit of work
- To commit cleanly you need to update first

Every Commit Builds

- It’s all about fast feedback
- Small changes
- Less to merge and/or fix

Test in Production Clone

- Detect multi-threaded or cluster issues
- Tests system architecture and upgrades
- Includes databases!
- It’s about moving parts, not capacity

Keep Builds Fast

- It really is all about feedback
- If things break you find out about it while it’s still fresh in your mind
- Keep up with frequent check ins

Everyone Sees What’s Happening

- Reduce time to fix
- No excuse to check in on broken build
- It’s not about blame, it’s about feedback

Automate Deployment

- Reduce Human Error
- Verify you can get it running somewhere other than “my machine”
- Test not only the code but your deployment mechanism too
- Don’t tie up Sys Admins with boring stuff
- Don’t tie up Devs waiting for feedback

Making CI Work

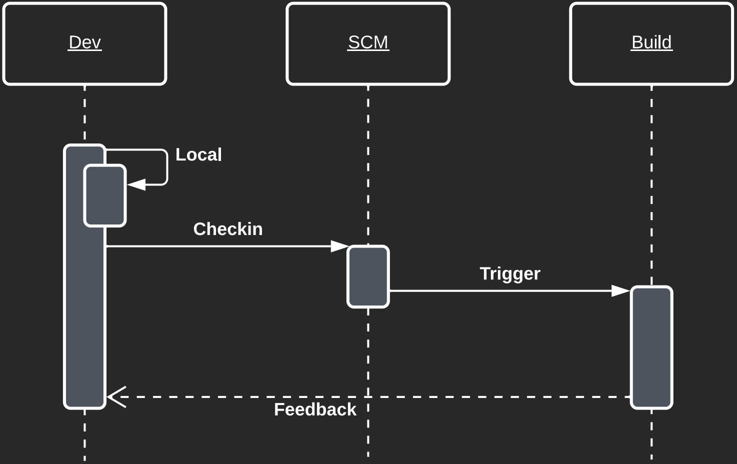
- Can’t be done in isolation
- Pick the right tools for the job
- It’s not a silver bullet

Build Pipelines

CS 3321

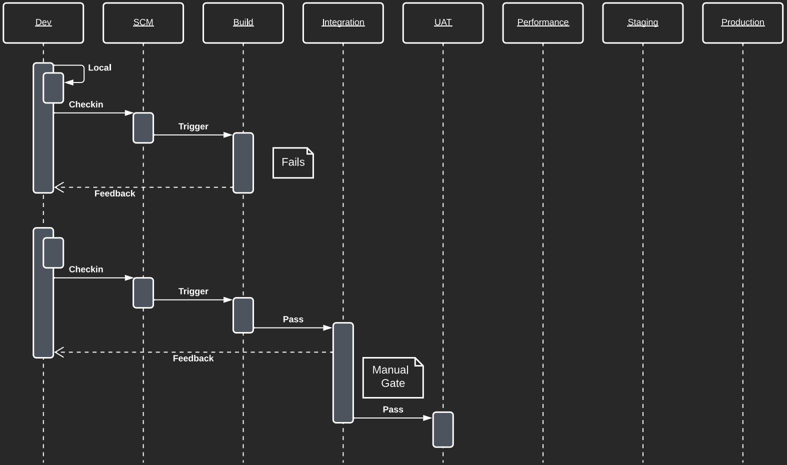
ROAR

Traditional CI Flow



Build Pipeline Flow

Idaho State University | Computer Science



ROAR

One Click Deploys

- Pre-requisite for build pipelines
- Reduces deployment time and risk
- Makes go live a non-event

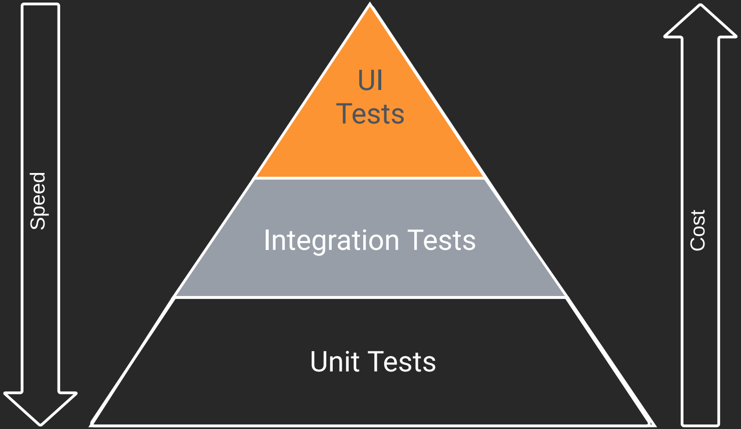
What Does Good Look Like?

CS 3321

What makes for good CI?

1. Decoupled stages
   - Each step in CI should do a single focused task
2. Repeatable
   - Automated in a way that is consistently repeatable
   - Tooling should work for local developers too - Local/Remote parity
3. Fail fast
   - Fail at the first sign of trouble

The Test Pyramid



What make for good CD?

1. Design with the system in mind
   - Cover as many parts of a deployment as possible
   - Application | Infrastructure | Configuration | Data
2. Pipelines
   - Continually increase confidence as you move towards production
3. Globally unique versions
   - Know the state of the system at any time
   - Be able to demonstrate difference between current and future state

Benefits of CI/CD

CS 3321

ROAR

Why do this?

- Reduction of delivery risk
  - No longer do we need to rely on humans with specific knowledge as the gate-keepers of quality
  - Reduced chance of humans not following the process
  - Reduced chance of miscommunication on executing the change

Why do this?

- To encode the process, we need to know the process
  - If we know all the tests pass,
  - If we know all the steps in deployment,
  - What is stopping us from releasing?

Why do this?

- Better visibility on change
  - As our systems and tools are version controlled
  - And we know what the current state of production is
  - And we can describe the process by which it will be changed
  - We can diff the system states with confidence
- Opens up more avenues for review and increased audit compliance

Why do this?

- Increased efficiency and delivery options
  - Enables us to deliver things with reduced effort
  - This leads us to deploy change more frequently
  - Which leads to getting feedback faster
  - That enables us to experiment easier
  - This leads to smaller batch sizes
  - Which leads to an increased flow of the entire system

Why do this?

- Enhanced learning from failure
  - When we have an issue or failure, we write a test to cover it
  - This test gets added to our suite and executed every time
  - Decreases our risk of this issue occurring again

For Next Time

- Review the Reading
- Review this Lecture
- Come to Class

Are there any questions?

ROAR