AWS CodeStar

AWS CodeCommit

AWS CodePipeline

AWS CodeBuild

AWS CodeDeploy

Continuous Integration

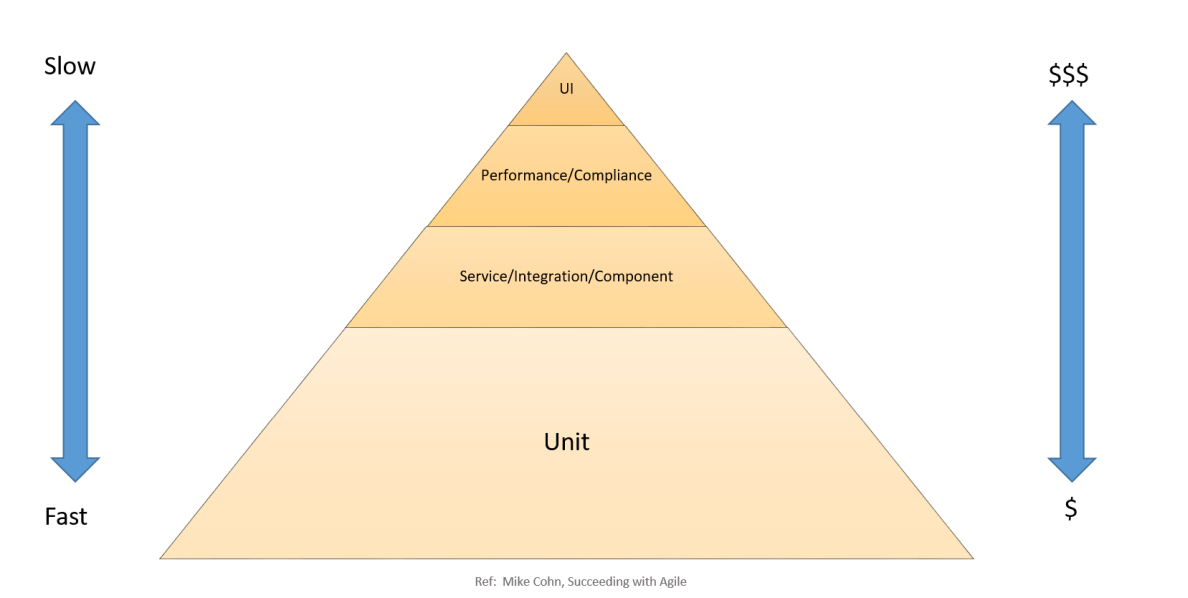
1. More frequent commits
2. Single code repository
3. Automating builds
4. Automating testing
5. Testing similar environments to production
6. Providing visibility of process to the team
7. Allowing developers to easily obtain any version of the application

Continuous Delivery

1. Build test prepare for production release

Create three teams

1. Application Development team
2. Infrastructure team
3. Tools team



100% code coverage and 70% unit test cases out of overall testing

Build tool?

Integration Testing

Component testing

System Testing

Performance testing

Compliance testing

User Acceptance Testing

Canary test

Minimum Viable Pipeline for Continuous Integration

all at once (deploy in place), rolling, immutable, and blue/green.

Do: Amazon Web Services – Practicing CI/CD on AWS

• Treat your infrastructure as code

o Use version control for your infrastructure code.

o Make use of bug tracking/ticketing systems.

o Have peers review changes before applying them.

o Establish infrastructure code patterns/designs.

o Test infrastructure changes like code changes.

• Put developers into integrated teams of no more than 12 self-sustaining members.

• Have all developers commit code to the main trunk frequently, with no long-running feature branches.

• Consistently adopt a build system such as Maven or Gradle across your organization and standardize builds.

• Have developers build unit tests toward 100% coverage of the code base.

• Ensure that unit tests are 70% of the overall testing in duration, number, and scope.

• Ensure that unit tests are up-to-date and not neglected. Unit test failures should be fixed, not bypassed.

• Treat your continuous delivery configuration as code.

• Establish role-based security controls (that is, who can do what and when).

o Monitor/track every resource possible.

o Alert on services, availability, and response times.

o Capture, learn, and improve.

o Share access with everyone on the team.

o Plan metrics and monitoring into the lifecycle.

• Keep and track standard metrics.

o Number of builds.

o Number of deployments. Amazon Web Services – Practicing CI/CD on AWS

o Average time for changes to reach production.

o Average time from first pipeline stage to each stage.

o Number of changes reaching production.

o Average build time.

• Use multiple distinct pipelines for each branch and team.

Don’t:

• Have long-running branches with large complicated merges.

• Have manual tests.

• Have manual approval processes, gates, code reviews, and security reviews.