Welcome to Week 3

Cloud Accelerator Program

CI & CD and GitLab in AWS

DevelopIntelligence

A PLURALSIGHT COMPANY

Hello

About me...

- 26+ years in the industry
- 21+ years in teaching
- Certified Cloud architect
- Passionate about learning
- Also, passionate about Reese's Cups!

Why study these subjects?

In modern software engineering, our ability to quickly deploy incremental innovation, ensure its quality, and scale to meet customer demand proves critical to our success

- Cloud is everywhere and it’s not going away
- As with many topics in technology, there are multiple options and multiple dimensions to those options
- Building a deeper understanding of Cloud and its offerings helps prepare you for modern IT
- “Automating Everywhere” helps us drive efficiency, quality, and speed into our Cloud deliveries

My pledge to you

I will...

- Make this interactive
- Ask you questions
- Ensure everyone can speak
- Use an on-screen timer

Agenda

- CI/CD as a capability / differentiator
- Native AWS services supporting CI/CD
- Using GitLab CI with AWS

How we're going to work together

- Slides and words to highlight key concepts
- Demos to bring those concepts "to life"
- Lab work (which will take place in sandboxes provided by "A Cloud Guru") for hands-on reinforcement
- NOTE: I welcome being interrupted – if you need more info, or clarification, or anything else, just break in and ask. I am here to help you.

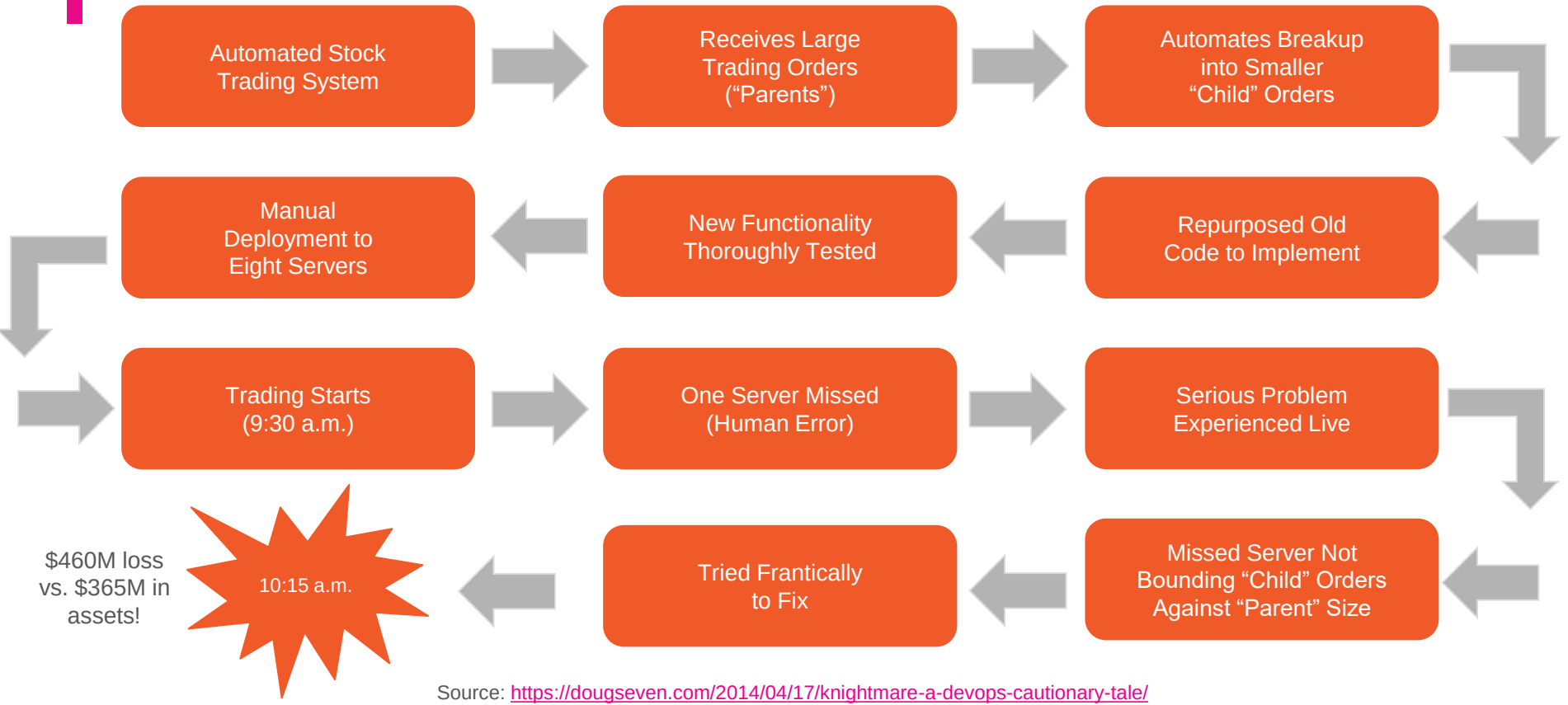
CI/CD as a Capability / Differentiator

What a difference 45 minutes can make!!

How Long Does it Take for a Company to Go Bankrupt?

If the circumstances are right, it can happen in just 45 minutes!

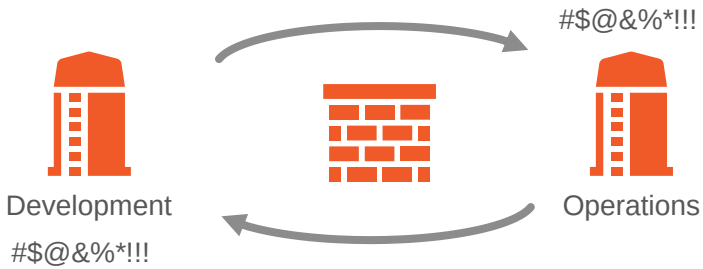
Knight Capital Group – What Happened?



Source: https://dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale/

DevOps – What it is & why it’s valuable

Approach Used In the “Olden” Times



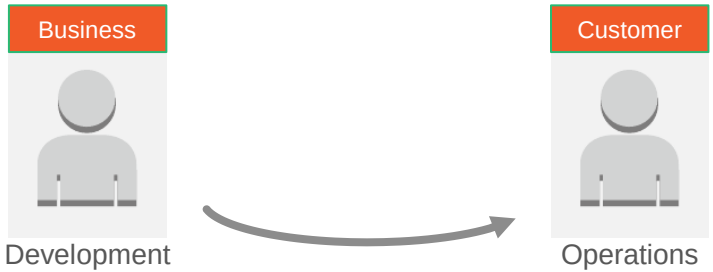
- Development & Operations siloes
- Lack of understanding of the other perspective
- Lack of communication & partnership
- “Fire and forget” engagement

DevOps – The New Way

- Development & Operations work together
- Each understands & appreciates the position of the other
- Good communication, partnership, & collaboration
- Ongoing engagement – continuous improvement

DevOps – The Three Ways

The First Way: Flow/Systems Thinking

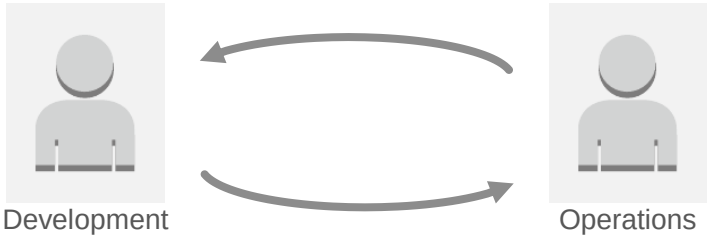


Source: Gene Kim (https://itrevolution.com/the-three-ways-principles-underpinning-devops/)

- Holistic vs. siloed view – success is defined by the performance of the entire system vs. a specific team or individual
- Starts with requirements (defined by the business) and not called “done” until value is delivered to the customer
- Target outcomes:
  - Known defect never passed to downstream teams
  - Favor global performance vs. focusing on local optimization
  - Goal is increased flow (value)
  - Asking ourselves the question: “What don’t I know about our systems?”

DevOps – The Three Ways

The Second Way: Amplify Feedback Loops

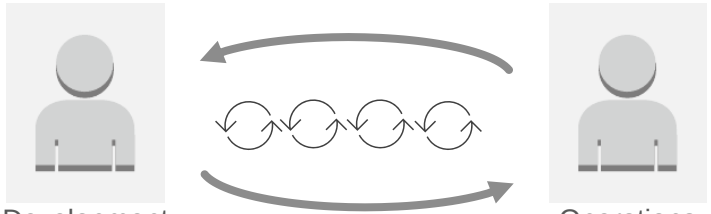


Source: Gene Kim (https://itrevolution.com/the-three-ways-principles-underpinning-devops/)

- Create right-to-left feedback loops
- Shorten & amplify so information needed to make continuous improvements & continuous correction is readily available
- Target outcomes:
  - Understanding all customer perspectives (internal & external)
  - Responding to all customer concerns (internal & external)
  - Knowledge is power!

DevOps – The Three Ways

The Third Way: Culture of Continual Experimentation & Learning

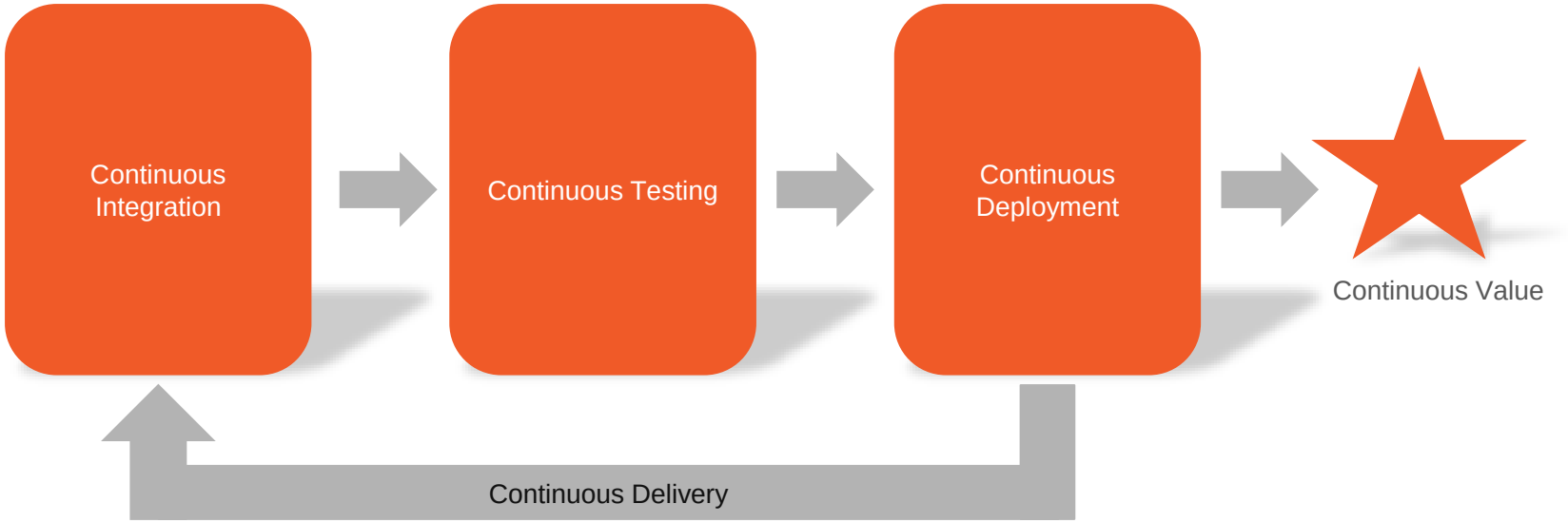


Source: Gene Kim (https://itrevolution.com/the-three-ways-principles-underpinning-devops/)

- Culture that fosters:
  - Continual experimentation, risk-taking, and learning from failure
  - Practice makes perfect!
- Target outcomes:
  - Allocating time for continual improvement (e.g., resolution of technical debt as a standard “category” of work)
  - Taking risks and seeing them pay off – No guts no glory!
  - Expecting failure & building resiliency into the process

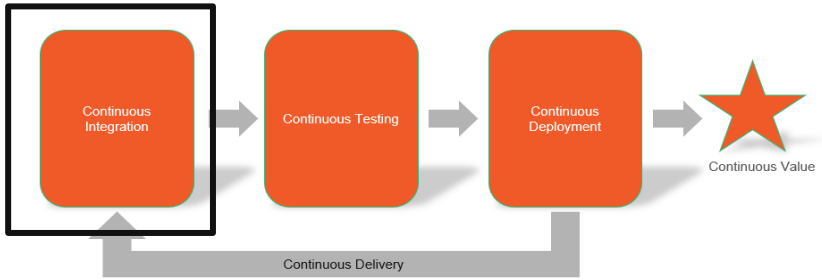
Continuous Delivery

Continuous Delivery



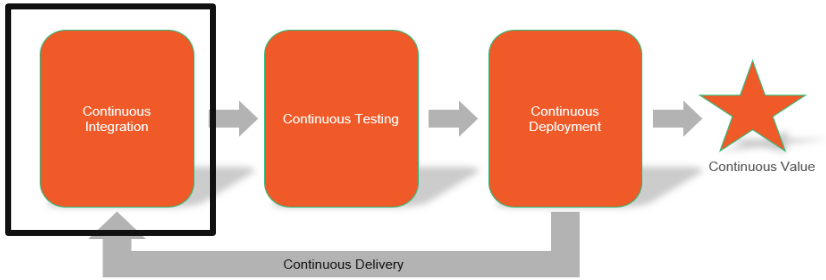
Continuous Integration

Continuous Integration



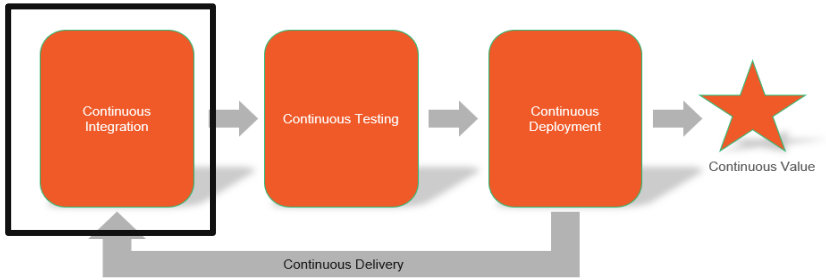
- Changes from individual developers are merged & system is validated “as a whole”
- Code changes are detected & automated processes are executed to assess
- Any issues uncovered are communicated to the developers for remediation (feedback)
- Targeted testing (i.e., unit testing) executed as part of CI flow

Continuous Integration



- Initiation is often event-driven
- Rather than waiting for a “critical mass” of changes before starting our testing, each change can be exercised & validated as it gets persisted to the centralized code repository
- Amplifies the feedback loop rather than burying in a “sea” of other changes
- Sets the stage for multiple types of validation (including security validation, or DevSecOps)

Continuous Integration

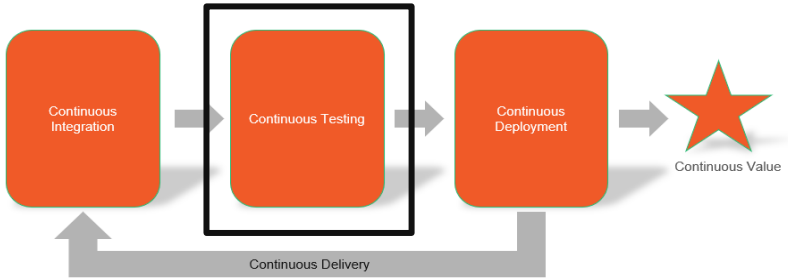


Best practices:

- Minimize dependencies between features – each feature should be able to stand on its own
- Work to drive out manual steps and make automation the priority
- Stabilize data sources & other system dependencies through mocking
- Include assessment of code quality (e.g., cyclomatic complexity) & unit test coverage as key parts of verification workflow

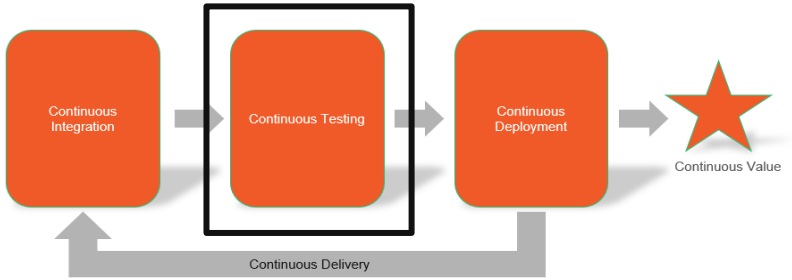
Continuous Testing

Continuous Testing



Common Types	Description
Unit testing	Testing discrete units of code in isolation from other code (e.g., testing a single method). Leverages mocking to stabilize other system dependencies so that tests are repeatable. Code coverage often used to assess quality & sufficiency of unit testing. Should be automated.
Regression testing	Testing that verifies that newly introduced changes for development of a feature do not inadvertently break other, existing features. Automated unit tests can also fill the role of a regression test suite nicely. Quickly confirming no breakages.
Integration testing	Testing that verifies that groups of components & component features operate correctly when combined with other features (e.g., multiple functions or modules that provide a larger “chunk” of functionality). Usually represents a layer above unit testing but below functional testing. Should be automated.
Functional testing	Testing that verifies functional requirements for critical workflows in the software end-to-end. Can be (and probably should be) implemented as automated black box tests, requiring minimal knowledge & minimal assumptions about the inner workings of the software.
Acceptance testing	Testing that verifies the functionality of the software against specific acceptance criteria defining the difference between “good” & “bad”. May include manual testing/utilization by a subset of testers to verify that the software will operate correctly when leveraged by end users in production. AKA does the software do what it is supposed to from the end user’s perspective?
Security testing	Testing that verifies the functionality of the software against critical security requirements as identified & prioritized via threat modeling. Through a combination of static & dynamic testing, assesses the software for security vulnerabilities or deficiencies from a compliance & regulatory perspective (A.K.A. DevSecOps). Examples include SQL Injection and Cross-Site Scripting.
Performance testing	Testing that verifies that the software will meet defined SLAs (Service Level Agreements) when exercised under load or stress. Can also be used to validate the software’s ability to elastically scale (out or in) based on volumes. How does the software perform under load?

Continuous Testing

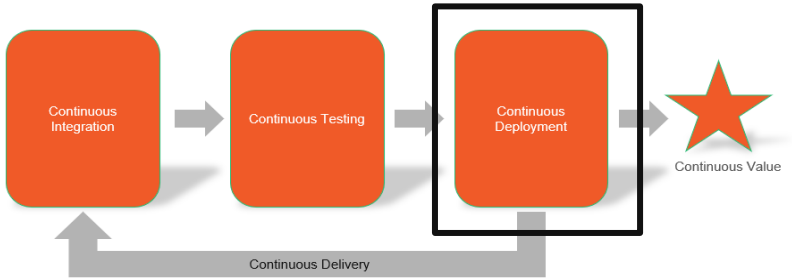


How does Continuous Integration help promote software quality?

- Maximizes the use & benefits of automation to enable “early & often”
- Provides multiple “hooks” within the workflow for plugging in the various types of testing
- Enables integration of testing results into the pipeline as quality gates
- Accelerates test result feedback for visibility & effective disposition

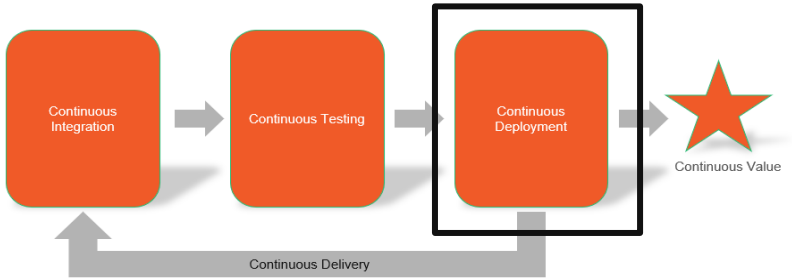
Continuous Deployment

Continuous Deployment



- Does not necessarily mean continuously deployed, but rather continuously *ready* to deploy
- Through a release pipeline, seeks to push latest version of the software through the environments at the frequency required by the business
- Intended to help get innovation “quickly” into the hands of users & customers for serving needs & gathering feedback

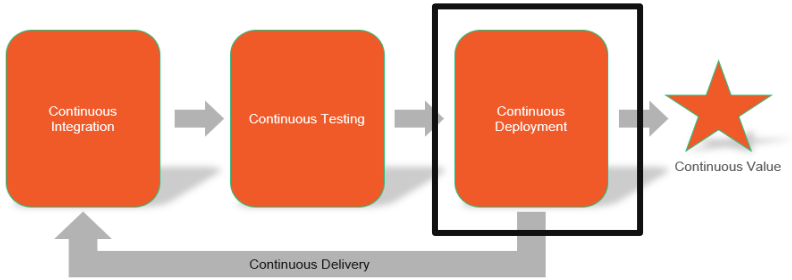
Continuous Deployment



Version management:

- Often leverages semantic versioning (major.minor.patch)
- Often leverages explicit versioning
- Seeks to keep a clear distinction & separation between each version of the software
- Goal is version traceability & visibility when upgrade is possible, or rollback is required

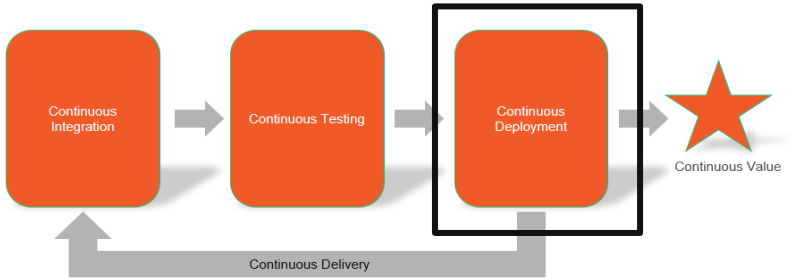
Continuous Deployment



Blue/Green testing:

- Maintains two (or more) identical hosting environments
- Provides the opportunity to run multiple versions of the software in parallel
- Production traffic is managed through routing – switch routing from old to new, & verify
- If issues uncovered, routing can be switched back to previous version for quick rollback

Continuous Deployment



Canary testing:

- Provides the opportunity to run multiple versions of the software in parallel
- Small portion of traffic can be routed to new version to ease in adoption
- Allows verification of new version in lower risk manner (like a canary in a coal mine)
- As confidence in new version builds, more and more traffic can be transitioned over until new version is used exclusively

So, What Went Wrong?

Repurpose of legacy code and legacy flags/variables without proper quality checks

Fully manual deployment

No “kill switch” and no clear path to rollback

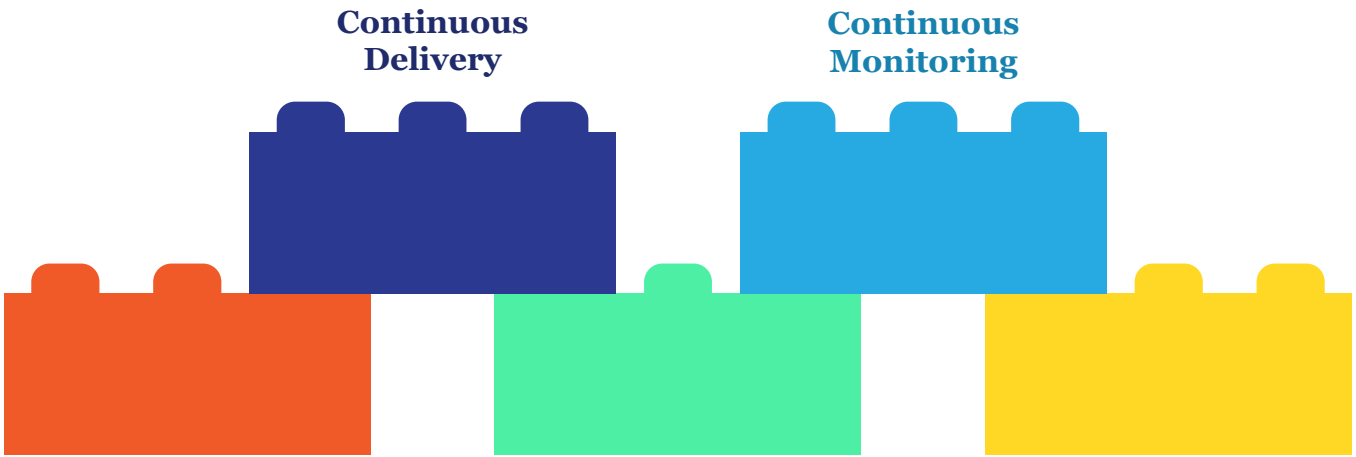
Insufficient observability and monitoring to alert resources to issues

Forced to test and troubleshoot live in Production

Building Blocks & Knight Capital Group

Software remains in a “ready to release” state enabling agility

Copious and effective monitoring alerts us to problems quickly so we can respond quickly



Continuous Integration

Automated testing finds issues quickly and helps ensure quality/reduce fear when releasing

Continuous Deployment

Automated release at the “push of a button” reduces element of human error

Automated Infrastructure

Automated build out of secure runtime environments (and ability to quickly roll back in the event of a discovered issue) drives stability

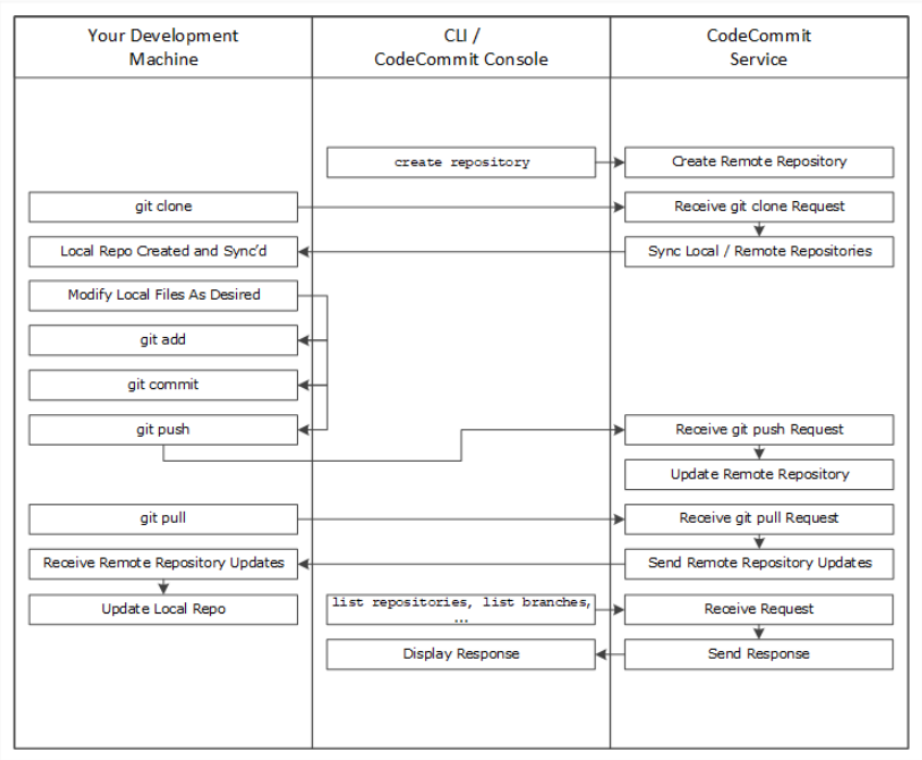
AWS-Native Services for CI/CD

AWS CodeCommit

AWS CodeCommit

- Provides AWS-native version control services (like GitHub but hosted in AWS)
- Supports secure management of code artifacts including tracking history of changes
- Integrates with and uses Git as a base for its operation

AWS CodeCommit



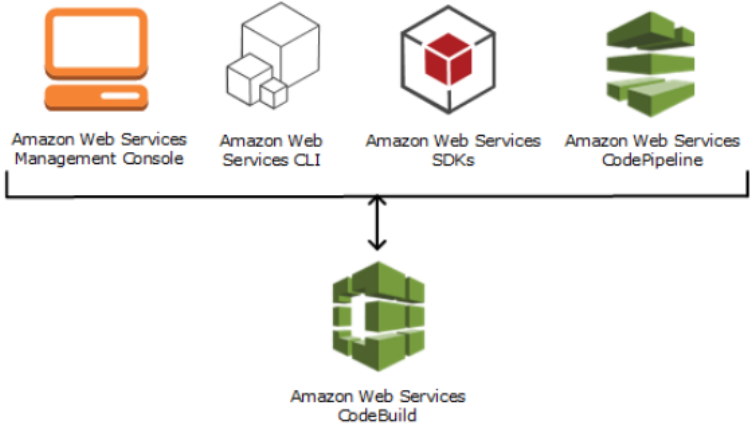
Source: https://docs.aws.amazon.com/codecommit/latest/userguide/welcome.html

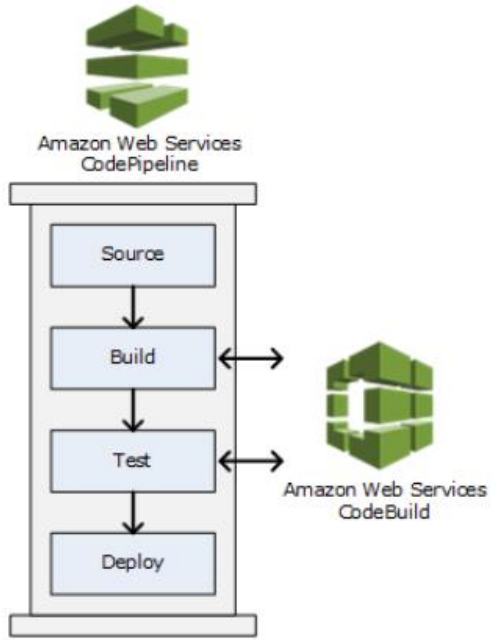
AWS CodeBuild

AWS CodeBuild

- Provides AWS-native build pipeline services
- Can be used to build code, run unit tests, and produce artifacts ready for deployment
- Handles creation and configuration of build servers/services with “hooks” that support integration with BYOT (“bring your own tools”)

AWS CodeBuild





Source: https://docs.aws.amazon.com/codebuild/latest/userguide/welcome.html

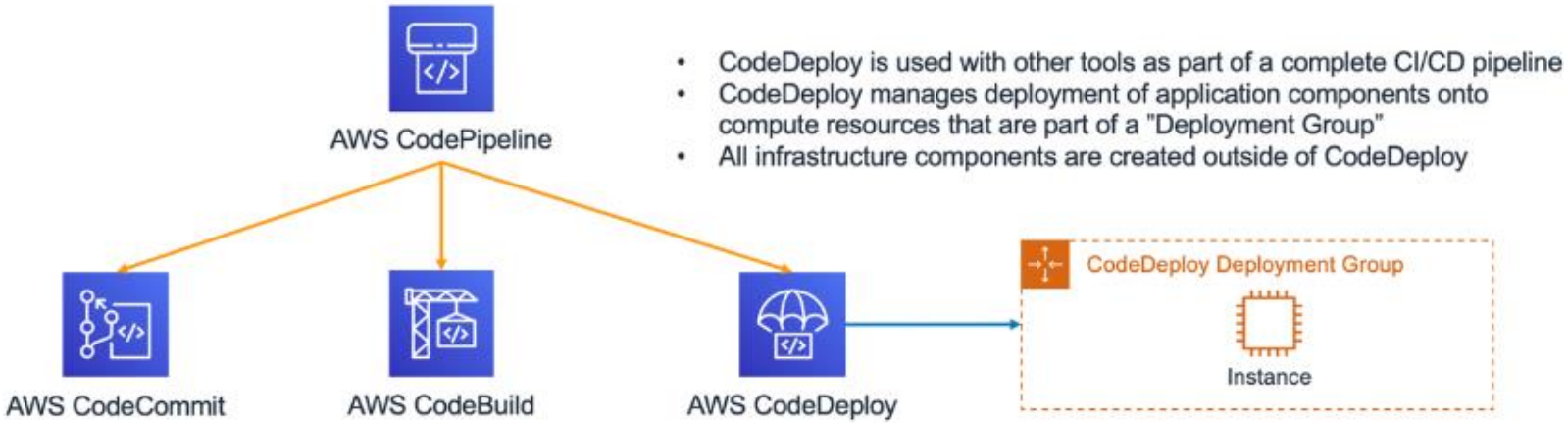
AWS CodeDeploy

AWS CodeDeploy

- Provides AWS-native software deployment services
- Automates application deployments to several types of AWS infrastructure (including EC2 instances, Lambda functions, Amazon ECS) and even to on-premise infrastructure (a la Hybrid)
- Supports multiple artifact sources for deployment (including CodeBuild)
- Natively supports Blue/Green testing and canary deployments

AWS CodePipeline

AWS CodeDeploy



- CodeDeploy is used with other tools as part of a complete CI/CD pipeline
- CodeDeploy manages deployment of application components onto compute resources that are part of a "Deployment Group"
- All infrastructure components are created outside of CodeDeploy

Source: https://docs.aws.amazon.com/whitepapers/latest/overview-deployment-options/aws-codedeploy.html

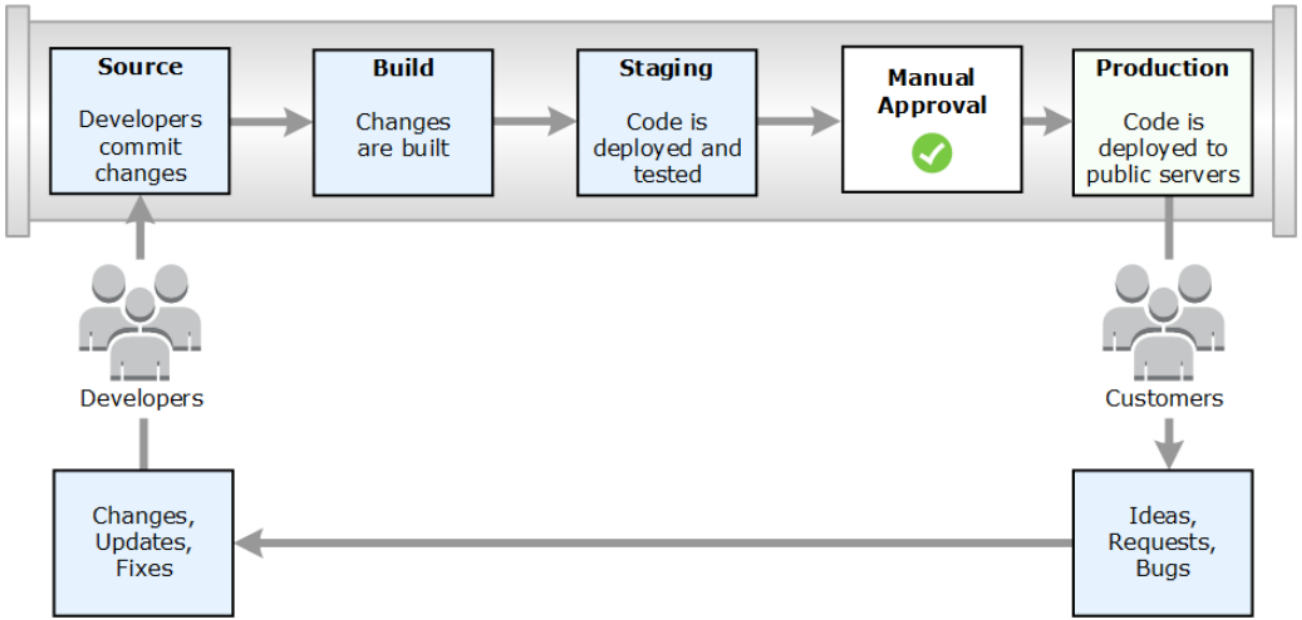
AWS CodePipeline

AWS CodePipeline

AWS CodePipeline

- Provides AWS-native continuous delivery services for your applications
- Supports automation of entire process (including integration with other native services) from “soup to nuts”

AWS CodePipeline



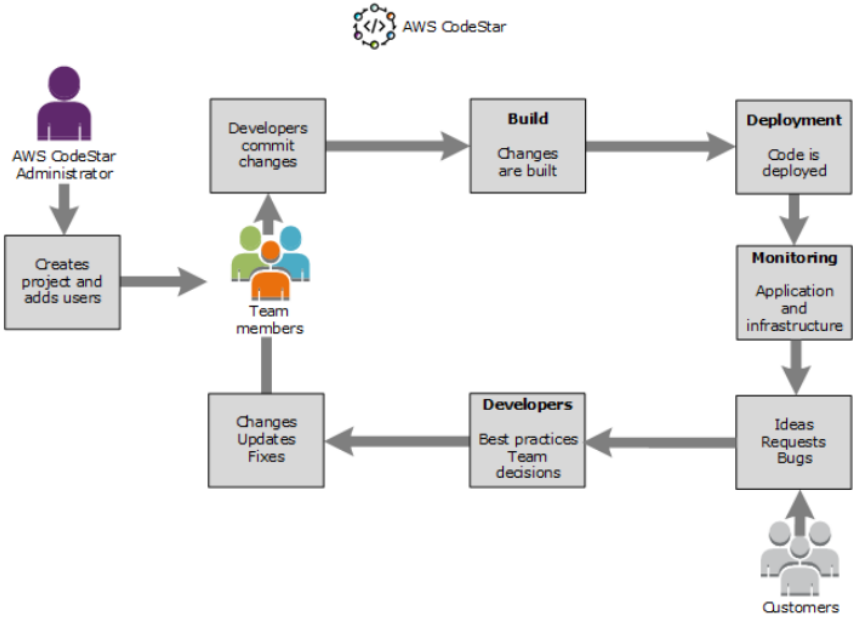
Source: https://docs.aws.amazon.com/codepipeline/latest/userguide/welcome-introducing.html

AWS CodeStar

AWS CodeStar

- Provides AWS-native service to facilitate setup and configuration of the other pieces of CI/CD workflow in the Cloud
- Offers multiple template types for common workloads
- Makes available a project dashboard that can be used to track end-to-end CI/CD for your automated builds & deployments

AWS CodeStar



Source: https://docs.aws.amazon.com/codestar/latest/userguide/working-with-projects.html

LAB:

AWS CodeStar

Instructor will provide guided walkthrough of
https://docs.aws.amazon.com/codestar/latest/userguide/sam-tutorial.html

LAB:

AWS CodeBuild

Instructor will provide guided walkthrough of
https://docs.aws.amazon.com/codebuild/latest/userguide/getting-started.html

LAB:

CI/CD in AWS

Execute the tutorial available at
https://learn.acloud.guru/handson/f0626640-3d35-4dd8-9a9c-2f90dfd9f1de

LAB:

AWS CodePipeline

Execute the tutorial available at
https://aws.amazon.com/blogs/devops/new-fine-grained-continuous-delivery-with-codepipeline-and-aws-stepfunctions/#:~:text=Complete%20the%20following%20steps%3A%201%20On%20the%20CodePipeline,Action%20to%20the%20action%20group.%20...%20More%20items

LAB:

GitLab CI & AWS

Instructor will provide guided walkthrough of
https://aws.amazon.com/blogs/apn/using-gitlab-ci-cd-pipeline-to-deploy-aws-sam-applications/

Thank you!

If you have additional questions, please reach out to me at:
asanders@gamuttechnologysvcs.com

DevelopIntelligence
A PLURALSIGHT COMPANY