4 – Advanced Testing Practices using AWS DevOps Tools

Module 1: Introduction and Testing

- I learned about the differnts tests described by the testing pyramid . From the base to the top, we have Unit tests (tests very specific part of the code), Service/Integration tests (tests that application's components can successfully communicate and interact with other parts of the application or third-party components), Performance/Complicance Tests (responsiveness an stability under workload) and UI/End-to-End tests (tests user experience)
- Importance of automation: Improve productivity, find and addres bugs faster, deliver updates more quickly, automate release process, etc
- Test coverage is a way to view the overall testing plan so it can be more effective

Module 2: DevOps Review

- I learned about DevOps important practices (continuous integration, delivery, deployment, microservices, infrastructure as code monitoring and logging, communication and collaboration)
- I learned which AWS tools are associated with the above practices and their features (CodeCommit, CodeBuild, CodeDeploy and CodePipeline), as well as third-party tools such as GitHub, CloudBees, Jenkins, TeamCity and GhostInspector.

Module 3: Continuous Integration

- I learned about CI practices, such as the significance of continuous integration, focusing on frequent commits, automated builds, and unit testing for ensuring the use of the latest working code among team members.
- Unit Testing in CI/CD: I understood the critical role of unit testing in CI, with a recommendation to allocate about 70 percent of testing efforts. I liked this citation: “If you don't like unit testing your product, most likely your customers won't like to test it either."
- I discovered several tools available for linting, unit testing frameworks
- We saw how to build specification for CodeBuild and how Codebuild provide clear test reports

Module 4: Continuous Delivery

- I learned about those types of tests:
  - *Functional Testing*: Ensures individual functions of the software operate as intended.
  - *Regression Testing*: Verifies that new code changes haven't adversely affected existing functionalities.
  - *Performance Testing*: Evaluates the system's responsiveness, speed, and overall performance.
  - *Load Testing*: Measures the system's ability to handle a specific load or user concurrency.
  - *User Acceptance Testing*: Validates if the software meets user requirements and expectations.
  - *Synthetic Testing*: Simulates real user interactions to assess system functionality and performance.

- In the pyramid tests, it includes Service/integration/Component tests, Performance/Compliance tests and UI tests.
- With this type of delivery, there is still a phase of manual approval before moving the app to production
- Importance of security checks. I learned that 45% of businesses have experienced a cloud-based data breach or failed audit in the past 12 months!

Module 5: Continuous Deployment

- I learned the difference with Continous delivery: with Continuous Deployment, production happens automatically, without explicit approval! It requires
- I learned about various deployment strategies such as
  - *Rolling Deployment*: Gradual release of updates across subsets of servers or users, minimizing potential impact by incrementally updating components.
  - *Segmented Deployment*: Release strategy targeting specific user segments in phases, allowing for controlled monitoring and assessment of the update's impact
  - *Canary Deployment*: Incremental release of updates to a small subset of users or servers to evaluate performance, identify issues, and ensure a smooth transition before a broader release.
  - *Blue/Green Deployment*: Involves maintaining two identical environments, with one (Blue) serving live traffic while the other (Green) undergoes updates. Traffic is then shifted seamlessly, minimizing downtime and allowing easy rollback if issues arise.
- We also learned about health checks and LifeCycle Hooks.
- We were introduced to AWS CodeDeployment and AWS CodePipeline
- Synthetics testing is automated testing that emulates user activity on our application every minute of every day. We will receie alter is something unexpted arises (we can use CloudWatch)
- We discovered how approvals can be managed, and that we can use lambda functions to automate the process.

Module 6: Course Summary

- This module was just a recap of the previous modules. In short, we learned:
  - Comprehensive testing techniques, practices, and tools vital for effective automation in the DevOps pipeline.
  - Diverse strategies for implementing testing across continuous integration, delivery, and deployment phases.
  - Integration and utilization of AWS tools and third-party solutions essential for establishing a fully-automated pipeline.
  - Practical knowledge on scaling application testing, implementing changes, and incorporating innovations while maintaining a focus on application quality and customer satisfaction.