OpenHack – DevOps 2.0 Attendee Takeaway

Overview

Microsoft's OpenHack series is an immersive, hands-on, challenge-driven hack that brings technical resources across the eco-system to tackle scenarios influenced by common, real-world problems using Microsoft Platform capabilities and other industry-leading technologies.

This OpenHack enables attendees to use DevOps practices to achieve zero downtime deployment for a micro-service-based application running in an Azure App Service. The OpenHack simulates a real-world scenario where developers from an insurance company must "keep the lights on" while evolving their containerized application – collecting relevant usage data and minimizing downtime. *During the "hacking" attendees will focus on:*

1. Building a CI/CD pipeline from scratch that accommodates basic testing and deployment.
2. Building out and improving the pipeline to implement monitoring, integration testing, and phased rollout.

By the end of the OpenHack, attendees will have built out a technical solution that is a complete development workflow using modern compute resources (Azure WebApps for containers).

Technologies

GitHub, Azure DevOps or Jenkins (*team choice*), Azure App Service, Log Analytics, Application Insights, Azure Monitor, Azure SQL Database, Azure Container Registry, Azure Container Instances

Challenges

Challenge 1: Establish your plan
In order to have a successful DevOps strategy, your team needs to have a plan.

The following materials were given to you for this challenge:

- What is DevOps?
- Bridging the gap between business and development
- Pair programming
- GitHub
  - Getting started with GitHub
  - Improving code quality and security with GitHub Protected Branches
  - Managing work with GitHub Issues and GitHub Projects
  - Linking GitHub Issues
  - Mastering GitHub Issues
  - Closing issues via Pull Requests
- Azure DevOps
  - Create an Azure DevOps organization
  - What is Azure Boards?
  - Integrating Azure Boards and GitHub
  - Improving code quality with Azure Repos branch policies
  - Link work items to support traceability and manage dependencies
  - Link work items to GitHub Commits and Pull Requests

Challenge 2: Setting up development workflow
In this challenge you selected the tooling that best fits your team's skills or learning plans, and configured and implemented the mechanisms that formed the basis of the development workflow.

The following materials were given to you for this challenge:

- What is Continuous Integration?
- Azure DevOps Engineering Release Flow
- Work Management
  - Azure Boards & GitHub
  - View and add work items using the Work Items page
- Branch Protection
  - GitHub - About protected Branches
  - GitHub Code Owners

Challenge 3: Implement continuous testing
In this challenge, you learned how to establish hybrid identity to allow for seamless control and access to on-premises and Azure resources after migration. With hybrid identity established, your customer was able to use a single identity to access on-premises applications and cloud services such as Azure, Office 365, and other sites on the internet. Identity and assessment management (IAM) controls for Azure resources were established which are needed during and after the migration.

Learning objectives:
- Plan hybrid identity
- Prepare on-premises environments for hybrid identity with Azure Active Directory
- Implement hybrid identity
- Implement identity access and management (IAM) using role-based access controls

The following materials were given to you for this challenge:
- What is Continuous Integration?
- Continuous Integration: What is CI? Testing, Software & Process Tutorial
- Workflow orchestration
    - What is Pipeline as Code?
- GitHub
    - GitHub Actions
- Azure DevOps
    - Azure Pipelines
- Jenkins
    - Creating a Jenkins pipeline
    - GitHub Permissions and API token scopes for Jenkins
    - Known issue with Docker workflow plugin and multi stage builds
- Unit testing
    - Test Driven Development Essay
    - Writing and running unit tests with GoLang
    - Writing and running unit tests with Node.JS
    - Writing and running units test with .Net Core
    - Writing and running unit tests with Java

Challenge 4: Implement continuous deployment (CD)
Now that you have successfully implemented continuous integration, it is time to show that you can deploy the container images that you have built for each of the four APIs to Azure App Service.

The following materials were given to you for this challenge:
- Continuous Delivery
    - What is Continuous Delivery?
    - Why continuous delivery?
- Github
    - GitHub Actions - Docker build and push
    - GitHub Actions - Azure WebApp
- Azure Pipelines
    - Azure Pipelines - Deploy a custom Linux container to Azure App Service
    - Azure Pipelines - Tutorial: Build a custom image and run in App Service from a private registry
    - Azure Pipelines - Docker task
- Docker Registries
    - GitHub Packages Documentation
    - Azure Container Registry documentation

Challenge 5: Implement a Blue/Green deployment strategy
Classic deployment mechanisms require downtime. And if something goes wrong, you need to rollback or otherwise remediate the broken deployment.

The following materials were given to you for this challenge:

- Blue/Green deployment
- Staging environments in Azure App Service
- Azure Slot Deployment with Blue-Green Deployment Model

- Azure App Service Deployment Slots Tips and Tricks

Challenge 6: Implement a monitoring solution for your MyDriving APIs

With a functional pipeline that builds and deploys your APIs, your next step is to focus on their health and performance.

The following materials were given to you for this challenge:

- Azure resources
    - Azure Monitor
        - Overview of the Azure Monitoring solutions
        - Getting started with Azure Monitor Log Analytics
        - Monitoring Azure resources with Azure Monitor
        - Monitor Azure App Service performance
        - Monitor Azure SQL Database using Azure SQL Analytics
        - Create diagnostic setting to collect resource logs and metrics in Azure
        - Monitor apps in Azure App Service
        - What is Application Insights?
        - Create, view, and manage log alerts using Azure Monitor
        - Overview of alerts in Microsoft Azure
    - Azure Logic Apps
        - Overview - What is Azure Logic Apps?
        - Azure DevOps Connector for Azure Logic Apps
    - Kusto documentation and references
        - The solutions in Azure will evolve to use Azure Data Explorer query language (also know as Kusto)
        - Syntax for regular expressions supported by Azure Data Explorer
    - Regular Expressions
        - RegExr: Learn, Build, & Test RegEx
    - Application availability and responsiveness
        - Availability alerts of any web site
    - Additional Reading
    - Azure Monitor overview
    - Application performance FAQs for Web Apps in Azure

Challenge 7: Integrating quality and security gates

In this challenge, you planned and improved your workflow to support one or more quality or security gates.

The following materials were given to you for this challenge:

- Dependency Scanning
    - GitHub Security Alerts
    - Automated Security Updates with GitHub
    - Whitesource Bolt with Azure Pipelines
    - OWASP Dependency Checker in the Azure DevOps marketplace
- Code Coverage
    - CoverAlls
    - CodeCov
    - Jacoco
    - CoverAlls for GitHub Actions
    - Azure Pipelines ecosysten (per-language code coverage docs)
    - GitHub Coverage Reporter for Jenkins
- Static Code Analysis
    - Linters in the GitHub Actions marketplace
    - Static Code Analysis tools (Wikipedia)
    - SonarCloud for GitHub Actions
    - Integrate Visual Studio Team Services with SonarCloud
    - Using SonarCloud in an Azure DevOps pipeline
- Variant Analysis (Security)
    - Variant Analysis for Security

    - [Continuous security analysis (with GitHub and SEMMLE)]
- Integration Testing
    - Integration testing
    - Integration tests vs Unit tests
    - Integration test with .Net Core
    - Integration testing in GO
    - Set up a go workspace
    - Integration testing with Node.JS
- Load Testing
    - Blazemeter with Azure Pipelines
- Manual Approvals
    - Setting up Webhooks for GitHub Actions
    - GitHub Deployments (REST API)
    - Release approval gates for Azure Pipelines
    - Manual Intervention task for Azure Pipelines

Challenge 8: Implement phased rollout with rollback

In this challenge, you implemented a solution that enabled the phased rollout (or percentage based, also sometimes referred to as canary) of a new version of the API of your choice.

The following materials were given to you for this challenge:

- Topical References
    - Route traffic with Azure App Service and Deployment slots
    - Explore how to progressively expose your Azure DevOps extension releases in production to validate, before impacting all users
    - Deployment considerations for DevOps
    - Canary Deployments Using Azure Web App Deployment Slots
- GitHub
    - GitHub Action Rollback Release
    - Context and expression syntax for GitHub Actions
- Azure Pipelines
    - How to implement rollback strategies in Azure Pipelines