OpenHack – DevOps 2.0

Overview

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 have to "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

Prerequisites

Knowledge Prerequisites

To be successful and get the most out of this OpenHack, participants should have existing knowledge of the benefits of adopting DevOps practices as well as Azure App Service and have good conceptual knowledge of DevOps. We recommend you read the following:

- What is DevOps?
- DevOps: Bridging the gap between business and development

Because you will be working in teams, a good overview of pair programming is useful. We recommend you read the following:

- Pair Programming

Be prepared to roll up your sleeves, learn, and participate in an interactive team environment.

Tooling Prerequisites

To avoid any delays with downloading or installing tooling, have the following ready to go ahead of the OpenHack:

- Bring a modern laptop running Windows 10 (1703 or higher), Mac OS X 10.13 or higher, or Ubuntu 16.04
- Install your choice of Integrated Development Environment (IDE) software, i.e. Visual Studio / Visual Studio Code / Eclipse / IntelliJ
- Download the latest version of Azure CLI
- Installation optional:
  - Docker for Windows
  - Docker for Mac
  - *Note:* If you are using Windows, you may want to enable Windows Subsystem for Linux and install Ubuntu or any other supporter distributions of Linux: https://docs.microsoft.com/en-us/windows/wsl/install-win10

Challenges

Challenge 1: Establish your plan

- Put the DevOps "mindset" in the team.
- The participants are asked to get to know themselves better, organize the team and define how and where they will handle work items.

Challenge 2: Setting up the Development Workflow

- The participants will learn the fundamentals of Planning and Continuous Integration using the tooling of their choice.
- They will be asked to demonstrate that they organized their work items and are able to associate work items with code changes.
- By utilizing Branch Protection teams can enforce code quality policies for incoming changes ensuring that each service has a code owner responsible for reviewing Pull requests before those changes are merged into the main code base.

Challenge 3: Implement Continuous Testing

- Participants will learn to integrate unit tests into a build pipeline to provide rapid feedback and augment thorough code reviews designed to only accept changes into master when all tests pass.
- They will be asked to run unit tests automatically and integrate them in the pipeline they are building.

Challenge 4: Implement Continuous Deployment (CD)

- The participants will learn the fundamentals of Release Management by automatically deploying an updated version of their application to an Azure App Service.

- They will be asked to demonstrate that they can perform an update to the application, link a work item with the associated code changes and reference the deployment of this feature with its corresponding work items.
- They will also be asked to demonstrate that the respective container images are only updated when changes are successfully merged into the master branch

Challenge 5: Implement a Blue/Green deployment strategy

- Building on the previous challenges, participants will learn how to implement a blue/green deployment strategy.
- They will be asked to articulate the blue/green logic and demonstrate its implementation for one of the APIs of the provided application, that they are able roll out code changes without causing application downtime.

Challenge 6: Implement a monitoring solution with alerting

- The participants will learn how to close the DevOps loop by adding monitoring and alerting.
- They will be asked to demonstrate a view aggregating the monitoring of the application and infrastructure. They also will have to implement alerting in the case of application performance degradation. Alerts should also generate a work item in the team work tracking system.

Challenge 7: Integrating quality and security gates

- The participants will be required to improve the automated testing capabilities of their pipelines to incorporate more sophisticated quality and security gates that integrate with external tools.
- They will also be required to demonstrate one or more of the following enhancements to their pipeline: Dependency scanning, Code coverage, Static analysis, Variant analysis, Integration tests, Load tests and Manual Approval prior to deployment.

Challenge 8: Implement phased rollout with rollback

- The participants will revise their deployment strategy and learn how to perform a blue/green deployment with gradual rollout and how to implement a “rollback” mechanism.
- They will be asked to add several phases to their existing pipeline to support a gradual rollout/rollback of a new version of the application. They will be asked to implement gates to validate the behavior of the application and implement a rollback mechanism.

Value Proposition

- Sound DevOps fundamental upskilling and developing Zero-downtime deployment strategies which translates to reduced friction in production deployments, ensuring that deployments of new features can occur more frequently and safely without requiring system downtime.

Technical Scenarios

- <u>"Keeping the lights on”</u>- implementing a production pipeline that alleviates the problem of high downtime when making new development changes to your application
- <u>Testing</u> – Unit, Integration and Load testing to reduce the risk of “breaking production” and ensuring that new code will integrate properly with current code
- <u>Phased rollout</u> – gradual code change implementation and ability to “rollback” to add a layer of security to production

Audience

- Target Audience:
  - Microsoft – CSE, CSA, GBB
  - Customer – App Developers
- Target verticals: Cross-Industry
- Customer profile:
  - Enterprises and ISVs looking to deploy containerized based workloads in the cloud
  - Customers looking forward to use containers for dev-ops
  - Customers looking forward to microservices architecture for their existing or new solutions
  - Customers looking to deepen their overall DevOps maturity

FAQs

Is there a suggested flow of OpenHacks which an attendee should attend first, before going to yours?

+ No, but previous knowledge of Containers will help attendees.

If I am only interested in using FaaS for compute, should I attend this OpenHack?

+ No, Serverless services are not covered in this OpenHack – only Containers will be addressed.