microsoft-open-hack-containers-v2

Overview

OpenHack

Hello and welcome to OpenHack, a challenge oriented hack event from Microsoft. You will be presented with a series of challenges, each one more difficult than the one before.

You should already be assigned to and seated with a team, with whom you will attempt to solve as many challenges as you can within today's hack time.

You have been assigned a coach who will be your first point of contact, and is here to support you and answer questions during the hack. They will not, however, solve the challenges for you.

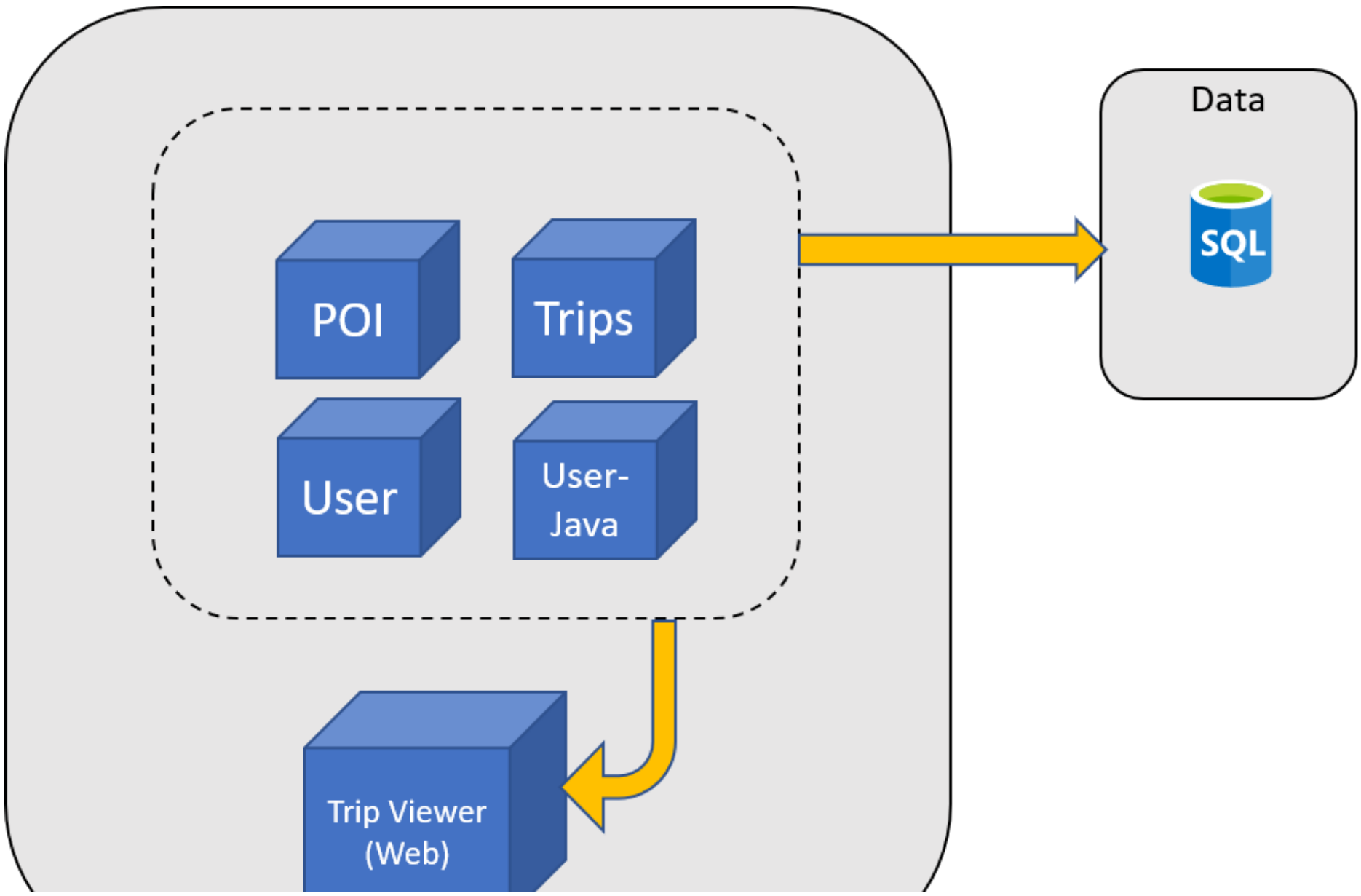
You may notice a resource group called **teamResources** in your Azure subscription. This resource group contains any pre-provisioned resources referenced in the challenges.

> **Note:** Be sure to review the Getting Ready to Go section to ensure you have your environment prepared for the OpenHack.

The Premise

You work for Humongous Insurance. One of their products provides customers the opportunity to qualify for lower car insurance rates. Customers can do this by opting in to use Humongous Insurance's TripInsights app, which collects data about their driving habits. Your team has been assigned to modernize the application and move it to the cloud.

The TripInsights application, once a monolith, has been refactored into a number of microservices:



ApplicationArchitecture.png

- **Trip Viewer WebApp (`.NET Core`)**: Your customers use this web application to review their driving scores and trips. The trips are being simulated against the APIs within the OpenHack environment.
- **Trip API (`Go`)**: The mobile application sends the vehicle's on-board diagnostics (OBD) trip data to this API to be stored.
- **Points of Interest API (`.NET Core`)**: This API is used to collect the points of the trip when a hard stop or hard acceleration was detected.
- **User Profile API (`NodeJS`)**: This API is used by the application to read the user's profile information.
- **User API (`Java`)**: This API is used by the application to create and modify the users.

The source code of all the microservices is available here.

The Challenges

Each challenge will lead you through a stage of the technical investigation as briefly laid out by your fictional CTO. These investigations become more technically challenging as you progress.

We do not provide guides or instructions to solve the challenges, just a few hints and documentation references that you may find useful. There are multiple ways to solve each challenge, and very likely some we haven't thought of. We're interested in seeing your own unique solutions to each problem, and you should absolutely work with your coaches and the OpenHack Team to validate your solution as correct.

One final tip: Read everything very carefully

The OpenHack team have worked hard to ensure each problem is solvable. All the details you should need are within the challenge briefs, which are very carefully written and worded to give you clues toward the solution. Reading them fully is the best way to figure out a solution, as small points can be easily missed. Your coaches will help to fill gaps in your understanding, provided you ask them the right questions.

Getting Ready to Go

Before you get started with these challenges, you'll want to make sure your environment has the tools and setup to work with the environment.

Tools you will need

All the tools used for these challenges are cross platform available and are usable on Mac OS X, Linux, and Windows environments.

> **NOTE**: If you are using a Linux platform, it is recommended to use one of these Ubuntu versions to ensure the full functionality with the environment.

Tool	Examples / Download Links	Purpose
Your choice of editor or integrated development environment (IDE)	Visual Studio Code	Editing and updating of code and configuration files
Terminal Environment	Bash (or similar such as Zsh) in Windows Subsystem for Linux (either version, WSL2 recommended), (included by default in Mac and Linux); PowerShell on Windows, Mac, or Linux	Used for running commands and scripts
Azure Command Line Interface (CLI)	Azure CLI	Used for interacting with Azure resourcesq
Docker	Docker for Windows, Mac, or Ubuntu	Used to build and run Docker containers locally
Kubernetes command-line tool	kubectl or via az aks install-cli	Used for control and management of Kubernetes clusters, such as Azure Kubernetes Services (AKS)
Helm	helm	Used for package management within Kubernetes
Git	Git, or one of the many GUI Clients	Local and remote code repository interaction and version control

Tool installation

There are several potential methods for getting these tools in your environment. You only need to follow one of these paths - just ensure you end up with a working environment that provides the required tooling.

- Install tooling listed above. While it's possible to complete this using Windows and Powershell, you may have an easier time using Bash on WSL (Windows Subsystem for Linux, linked above). Much of the Kubernetes ecosystem is very Linux-oriented; additionally, some of the sample CURL commands or other code snippets may need slight changes (`/` vs `\` and similar differences) to work in a Windows environment.
- WSL2 is highly recommended as it offers significant performance improvements and some ease of use improvements over WSL1. To set up Docker with WSL2, follow the Docker WSL2 documentation.
- If you're using WSL1, check out this guide on setting up Docker with WSL1.

2. **Using Remote Containers**
   - Remote container development allows you to use a pre-configured container with all necessary tooling for your project and limits what you need to install locally. The hack's source code repository comes with the configuration files necessary to build and run the project's "dev container."

     > **TIP**: Check out GitHub's Quickstart for Codespaces.

   - VS Code: In order to get started developing in a container with VS Code, refer to the VS Code documentation on developing in a container - Getting Started. On opening a project that supports remote containers, VS Code should prompt you to reopen in the container.
3. **Azure Cloud Shell**
   - Provides a terminal environment with `az`, `git`, `kubectl`, and `helm` (and other tools) pre-installed
   - Check out the Quickstart for Bash in Azure Cloud Shell.
   - As mentioned above, you can also use Powershell in Azure Cloud Shell but may find that the hack and Kubernetes ecosystem is slightly friendlier to Bash.
4. **Create a VM in Azure**
   - You can create an Azure VM to use as your development environment. However, this will offer similar functionality to Cloud Shell while requiring more time and effort to set up; it is likely preferable to use Cloud Shell over this option.

Further setup

You should perform the following tasks after you have ensured your development environment is setup properly.

1. **Pull the SQL Server Docker Image**: You'll run this in Challenge 1.

   ```
   docker pull mcr.microsoft.com/mssql/server:2017-latest
   ```

2. **Clone the Source Code Repository**: The source code for the applications you'll be working with in this hack can be found at Microsoft-OpenHack/containers_artifacts on GitHub.
   - Clone the repository from the command line or use VS Code to clone a GitHub repository.

   > **TIP**: If you're working with a team, it's highly recommended to create a new Git repository or create a fork of the hack source code repository in order to have a way to collaborate and share files as you progress.

3. **Useful Add-Ons**: Consider installing the following tools to help you through this OpenHack.
   - Kubectx / Kubens allow for quick switching between Kubernetes contexts and namespaces
   - Visual Studio Kubernetes Tools can be helpful, especially to quickly create templated YAML files (**Note**: This is pre-installed in the remote container development option)

References

Editor and Terminal Environment

- VS Code
- Windows Subsystem for Linux
- Best Practices for Setting up a WSL Development Environment
- Install PowerShell
- Developing inside a VS Code remote container
- Creating a Github codespace using remote container

CLI Tooling

- Install Docker
- Install Azure CLI
- Git
- Install Kubectl
- Install Helm

Working with Git

- Clone a GitHub repository
- Use VS Code to clone a GitHub repository

Project Source Code

- Microsoft-OpenHack/containers_artifacts on GitHub