**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.

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:

Diagram

Description automatically generated with medium confidenceApplicationArchitecture.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](https://github.com/Microsoft-OpenHack/containers_artifacts).

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

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](https://github.com/Azure/azure-functions-core-tools#linux) 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](https://code.visualstudio.com/download) | Editing and updating of code and configuration files |
| Terminal Environment | Bash (or similar such as Zsh) in [Windows Subsystem for Linux](https://docs.microsoft.com/en-us/windows/wsl/install-win10) (either version, WSL2 recommended), (included by default in Mac and Linux); PowerShell on [Windows](https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell-core-on-windows), [Mac](https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell-core-on-macos), or [Linux](https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell-core-on-linux) | Used for running commands and scripts |
| Azure Command Line Interface (CLI) | [Azure CLI](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli) | Used for interacting with Azure resourcesq |
| Docker | Docker for [Windows](https://docs.docker.com/docker-for-windows/install/), [Mac](https://docs.docker.com/docker-for-mac/install/), or [Ubuntu](https://docs.docker.com/engine/install/ubuntu/) | Used to build and run Docker containers locally |
| Kubernetes command-line tool | [kubectl](https://kubernetes.io/docs/tasks/tools/) or via [az aks install-cli](https://docs.microsoft.com/en-us/cli/azure/aks?view=azure-cli-latest" \l "az_aks_install_cli" \t "_blank) | Used for control and management of Kubernetes clusters, such as [Azure Kubernetes Services (AKS)](https://azure.microsoft.com/en-us/services/kubernetes-service/) |
| Helm | [helm](https://helm.sh/docs/intro/install/) | Used for package management within Kubernetes |
| Git | [Git](https://git-scm.com/downloads), or one of the many [GUI Clients](https://git-scm.com/downloads/guis/) | Local and remote code repository interaction and version control |

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.

1. Install on your local machine.
2. Utilize a remote container in VS Code or GitHub Codespaces (the configuration needed for this is provided within the source code repository).
3. Use [Azure Cloud Shell](https://docs.microsoft.com/en-us/azure/cloud-shell/overview) which provides a terminal environment with az, git, kubectl, and helm (and other tools) pre-installed
4. Set up a VM in Azure and install tools in the VM

Local machine setup

Install tooling listed above.

**Using Windows**

While it’s possible to complete this using 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 documention](https://docs.docker.com/desktop/windows/wsl/).

If you’re using WSL1, check out this guide on [setting up Docker with WSL1](https://nickjanetakis.com/blog/setting-up-docker-for-windows-and-wsl-to-work-flawlessly).

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.”

**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](https://code.visualstudio.com/docs/remote/containers#_installation). On opening a project that supports remote containers, VS Code should prompt you to reopen in the container.

**GitHub Codespaces**

Check out [GitHub’s Quickstart for Codespaces](https://docs.github.com/en/codespaces/getting-started/quickstart).

Azure Cloud Shell

Check out the [Quickstart for Bash in Azure Cloud Shell](https://docs.microsoft.com/en-us/azure/cloud-shell/quickstart" \t "_blank).

As mentioned above, you can also use [Powershell in Azure Cloud Shell](https://docs.microsoft.com/en-us/azure/cloud-shell/quickstart-powershell" \t "_blank) but may find that the hack and Kubernetes ecosystem is slightly friendlier to Bash.

Create a VM in Azure

You can [create an Azure VM](https://docs.microsoft.com/en-us/azure/virtual-machines/linux/quick-create-cli) 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

Pull the SQL Server Docker Image

You’ll run this in Challenge 1.

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

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](https://github.com/Microsoft-OpenHack/containers_artifacts).

[Clone the repository from the command line](https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/cloning-a-repository-from-github/cloning-a-repository) or [use VS Code to clone a GitHub repository](https://code.visualstudio.com/docs/editor/github#_setting-up-a-repository).

If you’re working with a team, it’s highly recommended to create a new Git repository or [create a fork](https://docs.github.com/en/get-started/quickstart/fork-a-repo) of the hack source code repository in order to have a way to collaborate and share files as you progress.

Useful Add-Ons

* [Kubectx / Kubens](https://github.com/ahmetb/kubectx) allow for quick switching between Kubernetes contexts and namespaces
* [Visual Studio Kubernetes Tools](https://marketplace.visualstudio.com/items?itemName=ms-kubernetes-tools.vscode-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](https://code.visualstudio.com/)
* [Windows Subsystem for Linux](https://docs.microsoft.com/en-us/windows/wsl/install-win10)
* [Best Practices for Setting up a WSL Development Environment](https://docs.microsoft.com/en-us/windows/wsl/setup/environment)
* [Install PowerShell](https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell)
* [Developing inside a VS Code remote container](https://code.visualstudio.com/docs/remote/containers)
* [Creating a Github codespace using remote container](https://docs.github.com/en/codespaces/developing-in-codespaces/creating-a-codespace)
* [Azure Cloud Shell](https://docs.microsoft.com/en-us/azure/cloud-shell/overview)
* [Create a Linux VM in Azure](https://docs.microsoft.com/en-us/azure/virtual-machines/linux/quick-create-cli)

CLI Tooling

* [Install Docker](https://docs.docker.com/engine/install/)
* [Install Azure CLI](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli)
* [Git](https://git-scm.com/)
* [Install Kubectl](https://kubernetes.io/docs/tasks/tools/)
* [Install Helm](https://helm.sh/docs/intro/install/)

Working with Git

* [Clone a GitHub repository](https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/cloning-a-repository-from-github/cloning-a-repository)
* [Use VS Code to clone a GitHub repository](https://code.visualstudio.com/docs/editor/github#_setting-up-a-repository)

Project Source Code

* [Microsoft-OpenHack/containers\_artifacts on GitHub](https://github.com/Microsoft-OpenHack/containers_artifacts)

But First, Containers

Containers have been adopted as a great way to alleviate portability issues. Containers form the core of this OpenHack and underpin everything you’ll be exploring as you progress through the challenges.

The objective of this challenge is to ensure you understand the very basics of containers, can work with them locally, and push them to a container image repository.

Challenge

You have been tasked with improving the local development experience for new developers by using Docker to simplify the building, testing, and running of the application. The CTO would also like to see this become part of the integration testing solution of the build pipeline.

Some of the work has been done for you, but it was during a time when teams were split between operations and development, leaving the code split between multiple codebases. The new CTO believes teams should be a mix of both Ops and Dev and has formed the team you are in now (say hi to your fellow teammates at your table :-)).

Building and Testing

Since you’re new to this code base, you’re going to verify at least one of the services still works by building and testing locally. In order to do this, you will need to build and run the Points of Interest (POI) container as well as a SQL Server container. The POI container communicates with the SQL Server container over the [Docker network](https://docs.docker.com/v17.09/engine/userguide/networking):

Diagram

Description automatically generatedAn architecture diagram showing 2 containers, labeled POI and SQL, running via Docker on your local machine. POI is able to communicate with SQL.

To build the POI application, use the **[TripInsights source code](https://github.com/Microsoft-OpenHack/containers_artifacts" \t "_blank)** and **[Dockerfile](https://docs.docker.com/engine/reference/builder/" \t "_blank)** for each microservice, matching the Dockerfile to the source code.

**Tip**: If you’re having trouble matching the Dockerfile to the source code, remember the services are written in different languages. Take a look inside the Dockerfile, the corresponding service is more obvious than you think.

After setting up a SQL Server container running locally, add sample data to the database using the following command:

```dockerfile

docker run --network <networkname> -e SQLFQDN=<servername> -e SQLUSER=<db-user> -e SQLPASS=<password> -e SQLDB=mydrivingDB openhack/data-load:v1

```

**Tip**: The dataload image used in the above command expects a database called “mydrivingDB” to exist already in SQL. Find a way to connect to your running SQL container to create this.

Then configure the POI application to connect to this SQL Server so you can test that the application works. You can find the [curl commands](https://github.com/Microsoft-OpenHack/containers_artifacts/tree/main/src/poi#testing) to test the applications endpoints in the POI applications README.

IMPORTANT: Set the ASPNETCORE\_ENVIRONMENT environment variable in POI to Local. This configures the application to skip the use of SSL encryption, allowing connection to the local sql server.

Building and Pushing TripInsights Images

Now that you are sure the POI application works, the team must ensure that all of the TripInsights components are built as Docker images and pushed to the team’s Azure Container Registry (ACR). This registry has already been deployed to your Azure Subscription. Reference the Azure Container Registry resource in the Azure portal for registry credentials.

If you choose to test the rest of the images, you can run them locally and send an HTTP GET request to the health endpoint. For example, to hit the POI health endpoint on a container running locally on port 8080, curl or visit in-browser <http://localhost:8080/api/poi/healthcheck>. Endpoints other than the health endpoint may not be functional at this point (due to dependencies on APIs or the SQL database), so don’t worry if you can’t reach them.

Success Criteria

* **Each member** of your team must show your coach a locally running Points of Interest (POI) container connected to a running SQL Server container. Verify that your POI container is serving content via [HTTP commands](https://github.com/Microsoft-OpenHack/containers_artifacts/tree/main/src/poi#testing). Explain your setup to your coach and how it could be used for development and testing.
* **Your team** must have built images for all the TripInsights components and pushed them to the team’s ACR. Share your understanding of how each of the images were built and pushed to the registry with your coach.

References

Docker

* [Getting Started with Docker](https://docs.docker.com/get-started/)
* [Docker Networking](https://docs.docker.com/v17.09/engine/userguide/networking)
* [Dockerfile reference](https://docs.docker.com/engine/reference/builder/)
* [Docker CLI reference](https://docs.docker.com/engine/reference/commandline/cli/)

SQL Server

* [Getting Started with SQL Server Container](https://docs.microsoft.com/en-us/sql/linux/quickstart-install-connect-docker?view=sql-server-2017)
* [Configuring SQL server](https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-configure-docker)

Azure

* [Azure CLI reference](https://docs.microsoft.com/en-us/cli/azure/get-started-with-azure-cli)
* [Azure Container Registry](https://docs.microsoft.com/en-us/azure/container-registry/)
* [Build with ACR](https://docs.microsoft.com/en-us/azure/container-registry/container-registry-tutorial-quick-task)

Trip Insights Source Code

* [Containers OpenHack on GitHub](https://github.com/Microsoft-OpenHack/containers_artifacts)
* [Testing POI API](https://github.com/Microsoft-OpenHack/containers_artifacts/tree/main/src/poi#testing)

Getting Ready for Orchestration

Containers are extremely useful on their own, but their flexibility and potential is multiplied when deployed to an orchestrator. Some of the advantages of deploying your containers to an orchestrator include:

* Deployment reliability
* Scaling on demand
* Better resource utilization and application density

Before getting into the details of what it takes to bring a cluster to production, your CTO would like you to first do a spike to validate your application can be deployed in a Kubernetes environment.

Challenge

At this point, you’ve built the images for the components of your application and made those images available in your private ACR. Your team’s goal in this challenge is to deploy your application to a test Azure Kubernetes Service (AKS) cluster in your Azure subscription.

Focus on making sure your containers are all up and can communicate and reach the necessary Azure services. In particular:

* tripviewer needs to be able to access the trips and userprofile services

Note: The Swagger API Documentation links on the homepage of tripviewer will not work at this time. You will add this functionality in a later challenge.

* Your POI and User (Java) APIs must be reachable (even though they are not accessed by the Trip Viewer application at this stage). Refer to the application documentation for ways to test the endpoints.

Note: There’s no need to give each API an external IP.

* All APIs must be able to access the SQL database provided in your Azure subscription. Connection details can be found in the **Messages** tab of your OpenHack portal.
* SQL connection information should be stored on the cluster in a Kubernetes Secret and not written directly into the deployment files for each microservice.

Desired Architecture

Diagram

Description automatically generatedAn architecture diagram showing a Kubernetes cluster and an Azure SQL database. Within the cluster, the TripViewer (Web) container is able to communicate to 4 other containers: POI, Trips, User, and User-Java. Those 4 containers communicate with the Azure SQL database.

Services will all run in the kubernetes cluster with the TripViewer application making calls to the APIs to get data. Data is stored on an Azure SQL Server which is accessed by the APIs.

Success Criteria

* **Your team** successfully created an AKS cluster in Azure
* **Your team** must demonstrate that at least one pod for each component of the TripInsights application is running
* **Your team** must demonstrate that the components in your cluster can connect to other components or resources:
  + tripviewer is able to access the trips and userprofile services
  + All APIs are able to access the SQL database provided in your Azure subscription
  + The POI and User (Java) APIs are reachable from the TripViewer web app top links (but the APIs do not have to be accessible from outside the AKS cluster)
* **Your team** must demonstrate that the components in your cluster are accessing SQL connection information via a Kubernetes Secret.

References

Kubernetes

* [Kubernetes core concepts](https://docs.microsoft.com/en-us/azure/aks/concepts-clusters-workloads)
* [Kubernetes Deployments](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/)
* [Kubernetes Services](https://kubernetes.io/docs/concepts/services-networking/connect-applications-service/)
* [Kubernetes DNS for services and pods](https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/)
* [Kubernetes Port Forwarding](https://kubernetes.io/docs/tasks/access-application-cluster/port-forward-access-application-cluster/)
* [Kubernetes External Load Balancers](https://kubernetes.io/docs/tasks/access-application-cluster/create-external-load-balancer/)
* [Kubernetes Secrets](https://kubernetes.io/docs/concepts/configuration/secret/)
* [Kubectl overview](https://kubernetes.io/docs/user-guide/kubectl-overview/)

Azure Kubernetes Service (AKS)

* [Deploy an AKS cluster using Azure CLI](https://docs.microsoft.com/en-us/azure/aks/kubernetes-walkthrough)
* [Azure CLI: az aks create](https://docs.microsoft.com/en-us/cli/azure/aks?view=azure-cli-latest#az-aks-create)

Azure Container Registry (ACR)

* [Authenticate with ACR from AKS](https://docs.microsoft.com/en-us/azure/container-registry/container-registry-auth-aks)

Azure

* [Azure CLI reference](https://docs.microsoft.com/en-us/cli/azure/get-started-with-azure-cli)
* [Resource naming conventions](https://docs.microsoft.com/en-us/azure/architecture/best-practices/naming-conventions)

To Orchestration and Beyond

Your CTO was impressed with your ability to show that AKS can easily support your application using a test deployment. However, you agreed that this deployment would not pass muster with your internal security team or meet audit requirements. Going forward, you’ll need to configure a cluster that will ultimately become part of Humongous Insurance’s existing cloud infrastructure.

Challenge

Your team’s goal in this challenge is to create and configure a Kubernetes cluster on Azure with the appropriate security measures in place. Your company deals with sensitive information, so it is imperative that you address security when configuring your cluster. You need to enable cluster authentication managed through your company’s Azure Active Directory (AAD) tenant and implement **Role-Based Access Control (RBAC)**, protect your resources by using a dedicated **Virtual Network (VNet)** and protect the most critical part of your Kubernetes cluster, the **Kubernetes API Server**.

Keep in mind these are just the first steps of securing your cluster. You will be asked to further improve your security in later challenges.

As you configure your cluster, your CTO would like you to consider **Availability**, **Network Requirements**, and **Access**.

Availability

1. Users of TripInsights expect their data to be accurate and up-to-date at all times. It’s important to consider the availability of the application to inform your decision on the number of nodes in your cluster.

Network Requirements

1. Due to the size of Humongous Insurance, many of the private IP address spaces are being used. You were lucky enough to get your networking team to give you an IP range for running applications within Azure. There is an existing VNet in your subscription that represents the IP range that has been allocated for your team (both for your cluster *and* other resources).
2. Pods on your cluster should be able to directly communicate with other resources on the VNET via private IP addresses.

**Protecting Resources with a VNet**

As with many other Azure services, you can protect your Kubernetes nodes by placing them into a VNet. The use of a VNet prevents unauthorized external connections, and can increase the security of corresponding managed services.

**Access**

1. Access to the Kubernetes API server should be limited to those who need it, and the level of access should depend on the intended use.
2. You may have noticed the other users in your Active Directory tenant. While you are part of the Admin team and should have permissions to access any resource in the cluster, there are two other users from the Web and API teams in your AAD tenant:
   * **web-dev** user (View access for API resources, Edit access for Web resources).
   * **api-dev** user (View access to Web resources, Edit access to API resources)

**Tip**: Segmenting cluster resources with *namespaces* will help manage access. Think through how namespaces might change existing configuration details such as service name resolution.

**Use of RBAC (Role-Based Access Control)**

RBAC is used to assign **Roles** (a group of permissions to resources) to **Users** (any entity that accesses a resource interactively) or **Service accounts** (any entity that accesses a resource non-interactively and independent of a User).

Using these constructs allows you to separate permissions between different users and engage in the **Principle of Least Privilege**. This principle suggests that any **User** or **Service account** should be assigned **Role(s)** with the minimum privilege necessary to access the resources that they require to complete their operational role against the cluster and for each application.

Note: When working with AKS, it’s important to be aware of both Azure RBAC *and* Kubernetes RBAC.

Success Criteria

* **Your team** successfully created an RBAC enabled AKS cluster within the address space allocated to you by the network team
* **Your team** successfully redeployed the TripInsights application, now segmented into api and web namespaces, into the cluster
* **Your team** must demonstrate connectivity to and from your cluster by being able to reach the internal-vm (already deployed)
* **Your team** must demonstrate that you are prompted on cluster access to authenticate with AAD
* **Different members** of your team must be able to connect to your cluster using the **api-dev** and **web-dev** AAD users and demonstrate appropriate access levels

References

Networking for AKS

* [Configuring Azure CNI with AKS](https://docs.microsoft.com/en-us/azure/aks/configure-azure-cni)

Access and Identity for AKS

* [Access and identity options](https://docs.microsoft.com/en-us/azure/aks/concepts-identity)
* [AKS-managed Azure Active Directory integration](https://docs.microsoft.com/en-us/azure/aks/managed-aad)
* [Control Kubeconfig Access](https://docs.microsoft.com/en-us/azure/aks/control-kubeconfig-access)
* [Use Azure AD and Kubernetes RBAC for clusters](https://docs.microsoft.com/en-us/azure/aks/azure-ad-rbac)

Availability

* [AKS Uptime SLA](https://docs.microsoft.com/en-us/azure/aks/uptime-sla)

Kubernetes

* [Kubernetes Namespaces](https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/)
* [Kubernetes RBAC Controls](https://kubernetes.io/docs/reference/access-authn-authz/rbac/)

Putting the Pieces Together

Now that your cluster is connected to your production network and has passed the initial scrutiny of your security team, it’s time to further improve your application’s security and open it up to external traffic.

Challenge

Some security concerns were addressed in the previous challenge, but it’s important to continue to keep security in mind for this challenge (and in the real world!).

Security

To improve secret management, you have additional security requirements mandated by your CTO:

1. Secrets should be secured in an external vault, not on the cluster. This approach prevents values from being accessed directly by any person without permissions or access to the vault itself.
2. Access to the external key vault should not require a secret stored in the cluster.

Ingress

Although you have multiple services deployed to the cluster, you will want a single endpoint for your customers to reach. To do this, create an ingress controller and configure the ingress rules to route to the appropriate services. The **References** section contains more information on the paths for the different components.

In order to validate that your application is working as expected, you will need to submit a single endpoint (http://endpoint.you.provide) to a provided simulator. The simulator will start sending traffic to the APIs once you provide your endpoint. It expects to make calls to the APIs by name (http://endpoint.you.provide/api/trips for example). You can see data start to flow through your app via the Trip Viewer application. The simulator is deployed as a container instance in your subscription and you will find the URL for the simulator in the **Messages** tab of your OpenHack portal.

Desired Architecture

Diagram

Description automatically generatedAn architecture diagram showing traffic flow into the Kubernetes cluster directed by an ingress controller. External traffic comes into the ingress controller, and from there is redirected based on path. “/api/poi” is directed to the POI service; “/api/trips” to Trips; “api/user” to User; and “api/user-java” to User-Java. The path “/” is directed to the TripViewer (Web) front end. Arrows indicate communication between TripViewer (Web) and the 4 API microservices as well as between the APIs and Azure SQL.

Success Criteria

* **Your team** secured your Azure SQL Server connection information such that literal values cannot be inappropriately accessed
* **Your team** used an external key vault to store and access secrets inside your cluster, and ensured that access does not require a secret stored in the cluster
* **Your team** ensured that all links on the Trip Viewer site are reachable
* **Your team** ensured the simulator can successfully update the values in the application across all services

References

API paths reference

* [Trip Viewer](https://github.com/Microsoft-OpenHack/containers_artifacts/tree/main/src/tripviewer#paths)
* [Points of Interest API](https://github.com/Microsoft-OpenHack/containers_artifacts/tree/main/src/poi#api-paths)
* [Trip API](https://github.com/Microsoft-OpenHack/containers_artifacts/tree/main/src/trips#api-paths)
* [User API](https://github.com/Microsoft-OpenHack/containers_artifacts/tree/main/src/user-java#api-paths)
* [User Profile API](https://github.com/Microsoft-OpenHack/containers_artifacts/tree/main/src/userprofile#api-paths)

API configuration reference

* [Points of Interest API](https://github.com/Microsoft-OpenHack/containers_artifacts/tree/main/src/poi#configuration)
* [Trip API](https://github.com/Microsoft-OpenHack/containers_artifacts/tree/main/src/trips#configuration)
* [User API](https://github.com/Microsoft-OpenHack/containers_artifacts/tree/main/src/user-java#configuration)
* [User Profile API](https://github.com/Microsoft-OpenHack/containers_artifacts/tree/main/src/userprofile#configuration)

Azure Kubernetes Service (AKS)

* [Secret Store CSI driver](https://docs.microsoft.com/en-us/azure/aks/csi-secrets-store-driver)
* [Ingress Controllers](https://docs.microsoft.com/en-us/azure/aks/concepts-network#ingress-controllers)
* [Create an NGINX ingress controller in AKS](https://docs.microsoft.com/en-us/azure/aks/ingress-basic)
* [HTTP Application Routing Ingress Controller](https://docs.microsoft.com/en-us/azure/aks/http-application-routing)

Kubernetes

* [Ingress](https://kubernetes.io/docs/concepts/services-networking/ingress/)

Azure

* [Resource naming conventions](https://docs.microsoft.com/en-us/azure/architecture/best-practices/naming-conventions)
* [Azure CLI reference](https://docs.microsoft.com/en-us/cli/azure/get-started-with-azure-cli)

Wait, What’s Happening

Deploying your applications to a cluster is just the first step for running containers in production, and it’s important to think about operations and scenarios around your deployments. It is valuable to have a holistic understanding of your cluster when it comes to ensuring your applications are reliable, available, and tolerant to failures.

**NOTE:** From this challenge on, your team may only advance if your cluster is in a healthy state.

Challenge

Your CTO is impressed with the speed at which you were able deploy the application but now wants to see you how your application is performing. The task the CTO has given for this challenge is to make sure your cluster is ‘production ready’ by implementing a monitoring solution that improves the observability of your cluster and adding alerts for key metrics so you can get ahead of any issues that will occur.

First, choose and implement a monitoring solution for your team to use. While choosing a monitoring solution, think about the four main components that must be considered to fully understand what is happening with your cluster so you can answer critical questions your CTO will ask.

1. Applications running on the containers
2. Containers
3. Underlying Virtual Machines
4. Kubernetes API

You can use the simulator from the previous step to create load on your application for testing and monitoring purposes.

A new application

Your focus on security and monitoring has encouraged other teams to try out AKS. Your CTO has asked you to run the following deployment in order to test a new project that calculates insurance rates faster and more accurately than ever before. This new project is intended to eventually become part of the TripInsights application. Ensure that you have configured your monitoring solution so you can quickly identify any issues that might arise.

Replace the image reference below with a reference to your ACR. The insurance application image is already deployed into your ACR.

apiVersion: apps/v1

kind: Deployment

metadata:

name: insurance-deployment

labels:

deploy: insurance

spec:

replicas: 2

selector:

matchLabels:

app: insurance

template:

metadata:

labels:

app: insurance

spec:

containers:

- image: "replaceme.io/insurance:1.0"

imagePullPolicy: Always

name: insurance

ports:

- containerPort: 8081

name: http

protocol: TCP

---

apiVersion: v1

kind: Service

metadata:

name: insurance

spec:

type: ClusterIP

selector:

app: insurance

ports:

- protocol: TCP

name: insurance-http

port: 80

targetPort: 8081

You can verify that the app is running by visiting the service endpoint which should return your calculation.

Of course, you didn’t write this application, so it isn’t up to your standards… Using your recently deployed monitoring solution, monitor the behavior of the new application in your cluster and see if you can determine what the runtime behavior of this application is. Additionally, if you find any issues, make sure to fix them in the deployment and create alerts for anything that might cause your application or cluster to experience downtime.

Success Criteria

* **Your team** must create a monitoring solution that shows the runtime behaviors of the application. You must be able to answer the following questions:
  + How many requests are coming to your cluster?
  + How much memory is allocatable per node in your cluster?
  + What is the CPU usage of your workload? What is the CPU usage of internal Kubernetes tools?
  + How many pods are currently pending?
  + Which pod is consuming the most memory?
* **Your team** must deploy a set of tools that will allow you to monitor your cluster and its applications.
* **Your team** must demonstrate where to obtain logs for the 4 main components mentioned in the first section of this page
* **Your team** must successfully implement resource limits on the newly deployed application
* **Your team** must set up an alert that informs you if an application is nearing resource limits in order to prevent cluster-wide issues
* **Your team** must demonstrate your cluster is overall “Healthy” for 15 minutess

References

Monitoring Microservices

* [Overview of Monitoring for microservices](https://docs.microsoft.com/en-us/azure/architecture/microservices/logging-monitoring)

Azure

* [Azure Container Insights reference](https://docs.microsoft.com/en-us/azure/azure-monitor/insights/container-insights-overview)
* [Azure Container Insights Agent Config](https://docs.microsoft.com/en-us/azure/azure-monitor/insights/container-insights-agent-config)
* [Search Logs to Analyze Data](https://docs.microsoft.com/en-us/azure/azure-monitor/insights/container-insights-log-search#search-logs-to-analyze-data)
* [Kusto Query Language Reference](https://docs.microsoft.com/en-us/azure/kusto)
* [Container Insights Alerts](https://docs.microsoft.com/en-us/azure/azure-monitor/insights/container-insights-alerts)

Prometheus

* [Prometheus](https://prometheus.io/docs/introduction/overview/)
* [Built-in Prometheus Metrics](https://github.com/helm/charts/tree/master/stable/nginx-ingress#prometheus-metrics)
* [Prometheus Exporters](https://prometheus.io/docs/instrumenting/exporters/)
* [Using Function Operators to Analyze Data from Prometheus](https://prometheus.io/docs/prometheus/latest/querying/examples/#using-functions-operators-etc)
* [Visualization with Prometheus](https://prometheus.io/docs/visualization/grafana/)

Locking it Down

You’ve done a lot of work so far to make sure your security and monitoring were in place, but there is still work to do to improve the security of your cluster.

Challenge

Your CTO is feeling confident in what your team has been able to accomplish, but there are still some security concerns you will need to address. After all, when dealing with sensitive information, security is one of the top concerns. As part of an effort to limit secrets stored in the cluster and to remove reliance on username and password authentication, the Trip service has been updated to enable SQL access via managed identity. Utilize this by setting up pod identity in your cluster.

In this challenge you must work to increase the security of your cluster by meeting these requirements:

1. Services in your cluster should only be able to make requests to other services if explicitly required.
2. None of the deployed services should be able to communicate with the api server.
3. None of the applications should run as root. This should be enforced by default.
4. Limit the egress traffic to only what is necessary.
5. Limit secrets stored in the cluster by implementing a managed identity for your services.

Success Criteria

* **Your team** restricted access from the deployed services access to kube-apiserver
* **Your team** limited the access to the Kubernetes API Server to only machines from your location
* **Your team** demonstrated that the API applications cannot call each other
* **Your team** restricted ability to deploy applications that have root access
* **Your team** limited the egress traffic from the cluster
* **Your team** enabled sql server access from inside the vnet only
* **Your team** has enabled the trip service access to SQL DB utilizing a managed identity.

References

Kubernetes

* [Kubernetes Service Accounts](https://kubernetes.io/docs/reference/access-authn-authz/service-accounts-admin/)

AKS

* [Azure Policy for Kubernetes clusters](https://docs.microsoft.com/en-us/azure/governance/policy/concepts/policy-for-kubernetes)
* [Using Network Policies](https://docs.microsoft.com/en-us/azure/aks/use-network-policies)
* [Secure access to the API server using authorized IP address ranges in AKS](https://docs.microsoft.com/en-us/azure/aks/api-server-authorized-ip-ranges)
* [Control egress traffic for cluster nodes in AKS](https://docs.microsoft.com/en-us/azure/aks/limit-egress-traffic)
* [Pod Identity](https://docs.microsoft.com/en-us/azure/aks/use-azure-ad-pod-identity)

Azure SQL Database

* [Use virtual network service endpoints and rules for database servers](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-vnet-service-endpoint-rule-overview)
* [Azure CLI: az network vnet subnet update](https://docs.microsoft.com/en-us/cli/azure/network/vnet/subnet?view=azure-cli-latest#az-network-vnet-subnet-update)
* [Azure CLI: az sql server firewall-rule](https://docs.microsoft.com/en-us/cli/azure/sql/server/firewall-rule?view=azure-cli-latest)
* [Managed Identities in SQL](https://docs.microsoft.com/en-us/azure/active-directory/managed-identities-azure-resources/tutorial-windows-vm-access-sql)

Mixed Emotions

We interrupt your regular programming with this special announcement:

Your company has acquired a company and you need to deploy some of its workloads into your cluster. One of the services is a Windows Communication Foundation (WCF) service, which calculates new policy prices based on driver data. Prior to the acquisition, the company experimented with containerizing this service and succeeded in running it on Docker. In order to take advantage of Kubernetes for additional orchestration, you will need to deploy this service alongside the rest of your Linux services and update the web app to communicate with the new service.

Challenge

Your team’s challenge is to deploy this new service along with the rest of the application into a single cluster without making changes to previous deployment files. Your team must also ensure the new service is integrated into the TripInsights application by upgrading the Trip Viewer application to use a different image that has already been deployed to your registry: *tripviewer2:1.0*. This application calls out to the WCF service in your cluster.

Things to think about

Much of what you’ve accomplished so far has given you the tools you need to incorporate this new service, however mixed workloads on Kubernetes and AKS can require you think differently about your overall cluster deployment, making sure it’s configured to support both OS types. Here are some questions to ask yourself when considering the addition of Windows workloads:

* What are the cluster features required to support Windows?
* How do you control the deployment to ensure Windows containers are directed to appropriate nodes?
* Are there changes to your cluster’s networking when adding Windows nodes to AKS?

About the Windows Service

The new service is a Windows Communication Foundation (WCF) service. It is already pushed to your registry as *wcfservice:1.0* and listens on port 80 for requests.

About Trip Viewer 2

The updated Trip Viewer app expects an environment variable, WCF\_ENDPOINT, in order to communicate with the service. You can test the communication by navigating to the UserProfile page of the app and clicking the ProcessRequest button for a user, which calls out to the WCF service and receives a response. If nothing happens, something isn’t right.

Desired Architecture

Diagram

Description automatically generatedAn architecture diagram showing a Kubernetes cluster with both Windows and Linux containers. External traffic comes in via the ingress controller, and is directed in the same manner as was specified in Challenge 4. As before, there is communication between TripViewer (Web) and the 4 API microservices as well as between the APIs and Azure SQL. However, there is now an additional Windows container labeled WCF which communicates directly with TripViewer (Web).

Success Criteria

* **Your team** successfully deployed the WCF application into the same AKS cluster as your Linux workloads
* **Your team** ensured that previous deployments are unchanged and unaffected
* **Your team** ensured that the updated Trip Viewer web app can successfully communicate with the WCF service
* **Your team** must demonstrate your cluster is overall “Healthy”

References

* [Windows Containers in Kubernetes](https://kubernetes.io/docs/setup/production-environment/windows/intro-windows-in-kubernetes/)
* [Multiple Node Pools in AKS](https://docs.microsoft.com/en-us/azure/aks/use-multiple-node-pools)
* [Windows on AKS](https://docs.microsoft.com/en-us/azure/aks/windows-container-cli)
* [About Windows Containers](https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/)
* [Configuring Azure CNI with AKS](https://docs.microsoft.com/en-us/azure/aks/configure-azure-cni)