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:

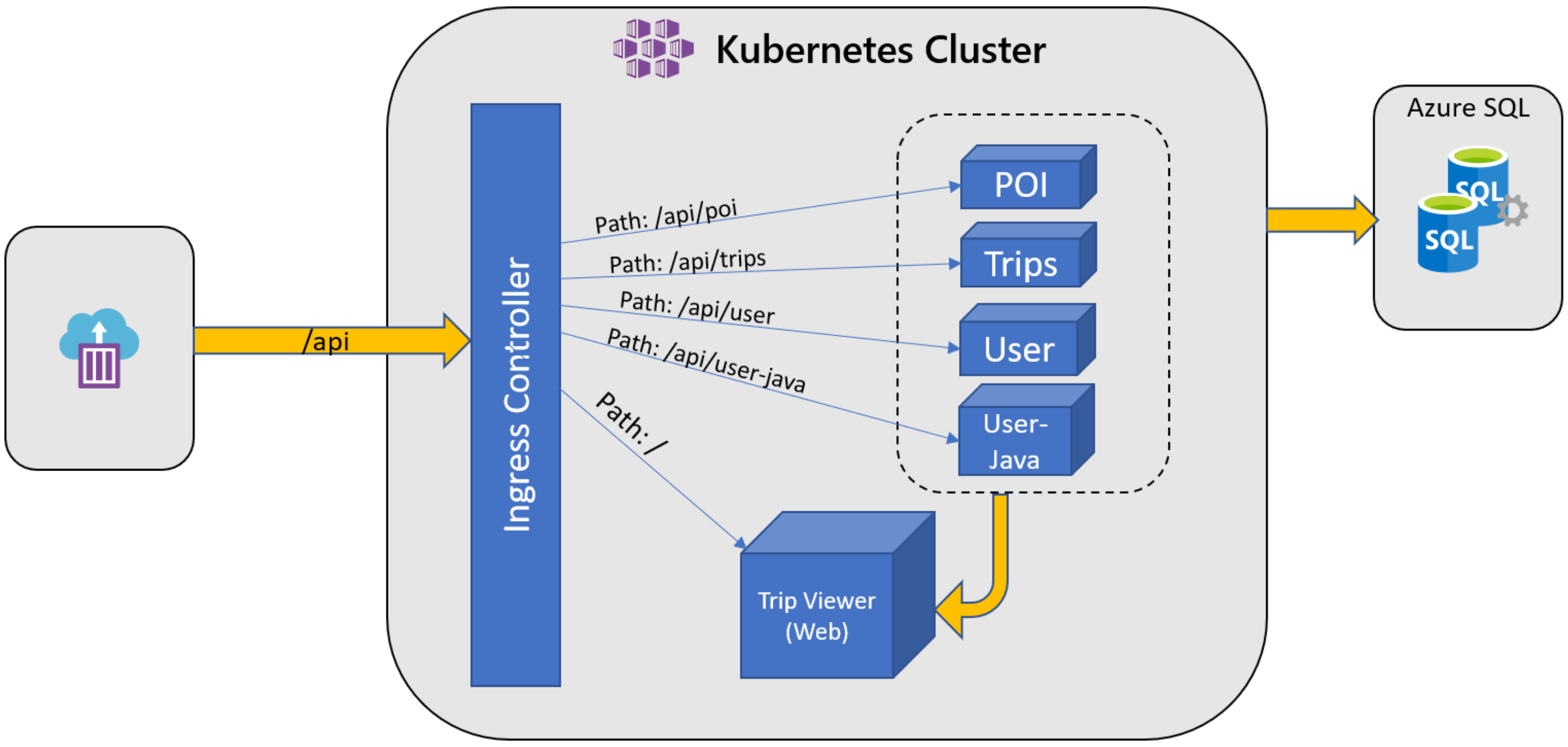
1. Secrets should be secured in an external vault, not on the cluster. This approach prevents values from being accessed directly by any person without permissions or access to the vault itself.
2. Access to the external key vault should not require a secret stored in the cluster.

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



An 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)