Connected Vehicle IoT Platform

# References

Table of Contents

[References 1](#_Toc193409007)

[Component interaction 2](#_Toc193409008)

[Security considerations 3](#_Toc193409009)

[Tech Stack selection and justification 3](#_Toc193409010)

[Deployment playbooks and infrastructure-as-code scripts 3](#_Toc193409011)

[Cloud Vs On-Premises deployment considerations 4](#_Toc193409012)

[Multi region and high availability deployment strategy 5](#_Toc193409013)

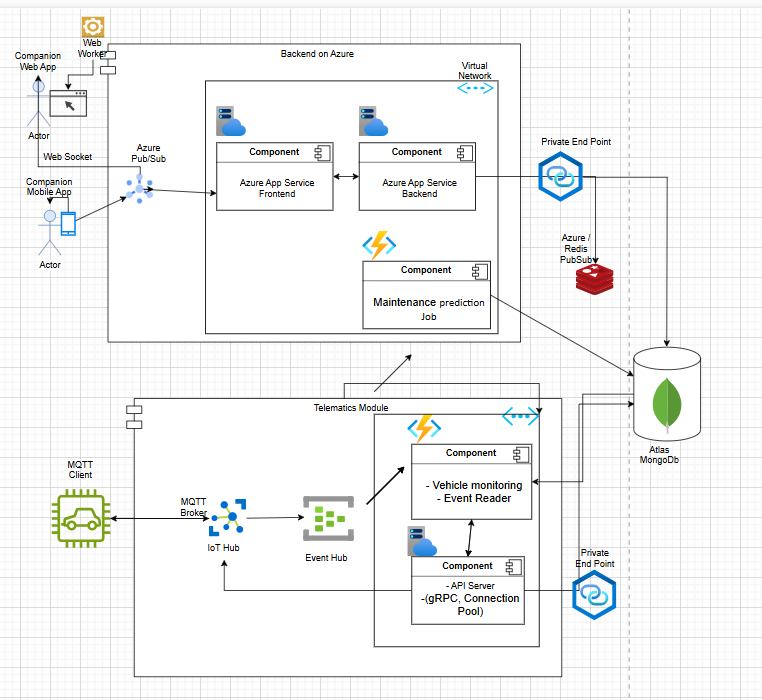
[Logging & Monitoring strategy 6](#_Toc193409014)

[CI/CD pipeline and automation framework 7](#_Toc193409015)

# Component interaction

Below is Component Diagram which covers most of complex components of system covering two major Modules

1. User engagement/companion (Backend for Azure)
2. Vehicle IoT Platform (Telematics Module)



# Security considerations

1. For storage of secrets, we can use Azure Key Vault. We can grant Azure Web Apps access to managed identity. Here is way to handle this process <https://learn.microsoft.com/en-us/azure/static-web-apps/key-vault-secrets>.
2. As Azure services have Microsoft Defender Security built in, which gives recommendations & Security alerts.
3. We can keep our resources (Azure App Services) inside Virtual Network and ensure that necessary network security group (NSG) rules are in place, for allowing in bound traffic on port e.g. 80 and protocol TCP.
4. We can enable Sonar Cloud to check in the pipeline, so that our code satisfies quality as well as security checks.

# Tech Stack selection and justification

* Use of Azure App Services reduces the burden of security and maintenance. We can use containerized application but here there are not so many complex services which are required wrt of backend of application for fetching data for user
* For Authentication, we can directly use Microsoft/Other Auth providers, which will ease the MFA & SSO.
* MongoDb is considered because we want Availability & participant tolerance (CAP) that will help in large amounts of data needs tobe written & read. Consistency will be eventual.
* gRPC will help in bulk write operation and that will help in reduce connection pool size.
* Event hub will help in reading data on our own pace and will also reduce database burden.
* Maintenance prediction job once in a day will help reduce computation cost.
* Redis storage will help making frequent calls to database and make read faster.
* Webworker will help storing basic data on user end until we have session at user end and scrolling lot of data easier.
* Azure Pub/Sub is considered to provide notification to client and provide high availability due to lesser number of API calls.
* IoT Hub, Event Hub are considered to handle 10 million active vehicles for data ingestion.

# Deployment playbooks and infrastructure-as-code scripts

Created terraform script main.tf and added instruction in readme.

# Cloud Vs On-Premises deployment considerations

Here deployment will cloud only, as we have configured for same.

# Multi region and high availability deployment strategy

* Created terraform for multiple region us and eu. IoT Hub, Event Hub are considered to provide high availability for data ingestion. Azure Pub/Sub is considered to provide high availability on client side.
* Mongo Db can be created for multiple regions when creating same. It is already server purpose of high availability due to sharding.

# Logging & Monitoring strategy

* For logging and monitoring, we can use Log Analytics. A single Log Analytics workspace might be sufficient for many environments that use Azure Monitor (Azure Monitor is a comprehensive monitoring solution for collecting, analyzing, and responding to monitoring data from your cloud and on-premises environments.
* ) and Microsoft Sentinel. But many organizations create multiple workspaces to optimize costs and better meet different business requirements.

# CI/CD pipeline and automation framework