# Case Study 1: Design an E-Auction Cloud Infrastructure for Deployment

Before you start with the case study remember the following points:

1. Set up your lab environment. Refer to the section "Environment setup".
2. Once you have set up your environment, you can start working on the case study. Refer to the section "Introduction".
3. At every step of the case study, you would need to take screenshot of the deliverables you need to produce as part of the case study. Please add the screenshot in a ppt and upload the ppt in the project folder in VM and push the same to GIT
4. On completion of the case study, it would be evaluated by a SME. Refer to the section

"Evaluation Criteria" for details of evaluation criteria.

1. Once you have completed the case study don’t forget to clean up the environment and stop the services to ensure you are not billed for the paid services.

# Case Study Background

**Client background**

Client wants to move their Monolithic to Microservices. Client looking for a solution to have it deployed and accessible though Azure public cloud

**Problem statement**

Client is planning to create serverless deployment. Need to auto scale and spin containers to support Microservices.

Number of users accessing website are not fixed. It changes every day.

**Business challenges or Pain points**

The website hosted must be highly available.

The website should be scalable to handle the user load.

Scaling up and down of the containers should be automatic in order to save the cost.

**Environment Setup**

An access to Azure Account is needed. Toolchain/Services involved are mentioned further below

**Best Fit Solution**

1. Create an Azure Function for POST method support

Ref: <https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-function-app-portal>

1. Use Azure API Management to expose Azure functions

Ref: <https://docs.microsoft.com/en-us/azure/api-management/import-function-app-as-api>

1. Create Azure ANS Notification (email: configure a fixed email)

Ref: <https://docs.microsoft.com/en-us/azure/azure-web-pubsub/tutorial-serverless-notification?tabs=javascript>

1. Configure the concurrency of your serverless implementation

Ref: <https://docs.microsoft.com/en-us/azure/azure-functions/functions-concurrency>

1. Deploy using Azure App Service

Ref: <https://docs.microsoft.com/en-us/azure/app-service/>

1. Configure Point-in-time snapshot

Ref:  [https://docs.microsoft.com/en-us/azure/azure-app-configuration/concept-point-time-snapshot?tabs=azure-portal](https://docs.aws.amazon.com/prescriptive-guidance/latest/backup-recovery/new-ebs-volume-backups.html)

1. Containerize using Azure Container

Ref: <https://docs.microsoft.com/en-us/azure/container-instances/container-instances-quickstart-portal>

1. Fetch All state values of container

Ref: <https://docs.microsoft.com/en-us/azure/container-instances/container-state>

**Evaluation Criteria**

1. Understanding of the problem – 10 marks  
   I. Complete Understanding (was able to identify the key problem and address it) – 10  
   ii.Partly Understood (was able to identify part of the key problem and address it) - 5  
   iii.Incorrect understanding (documentation does not reflect the key challenge) – 0
2. Completeness of Solution – Total 80 marks

Create an Azure Function for POST method support -> 20 points

Use Azure API Management to expose Azure functions and Create Azure ANS Notification-> 12 points

Deploy using Azure App Service -> 12 points

Containerize using Azure Container-> 16 points

Solutions worked -> 20 points

1. Documentation of results – 10 marks

# Problem Statement

**E-Auction Application** is microservice based Cloud Native Application. The Main purpose of **E-Auction** is to provide the ability to initiate sale, purchase and bid for a product by the customer. The core modules of E-Auction app are:

* Put a new product on auction with starting bid price.
* Bid for the product.
* Update the bid price.
* Enlisting of bids placed on any product.

**A complete solution implementation of the same will be provided which will be in microservice architecture. Thus, no need to develop any application.**

# Tool Chain

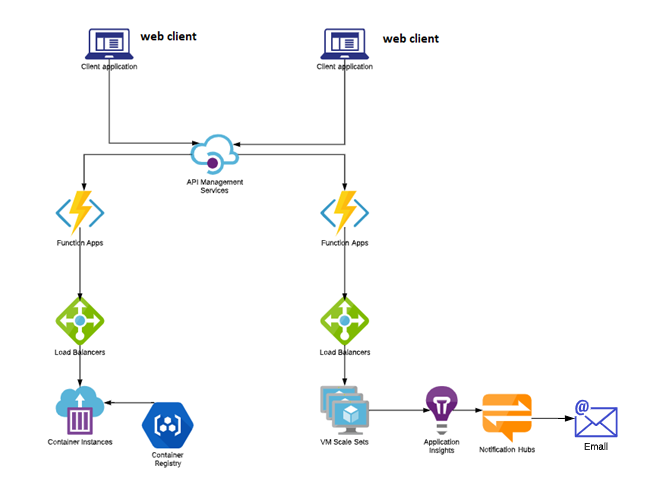
|  |  |  |
| --- | --- | --- |
| Engineering Quality |  |  |
| Platform | Cloud Tools | Azure Container Service |
|  |  | Azure Virtual Machine |
|  |  | Azure Functions |
|  |  | Azure Container Registry |
|  |  | Azure Operational Insights |
|  |  | Azure API Management |
|  |  | Azure Cloud Services |
|  |  | Azure Notification Hub |
|  |  | Azure Service Bus |
|  |  | Azure Disks |
|  |  | Azure Load Balancing |
|  |  | Azure Logic Apps |
|  |  | Azure AD |

# Business Requirements:

Below are the user stories for which implemented solution is already available and would be shared

|  |  |  |  |
| --- | --- | --- | --- |
| **User**  **Story #** | **User Story Name** | **User Story** | **Development** |
| US\_01 | Seller Functionalities | As a seller adding a new product for auction | API is available |
| US\_02 | Seller Functionalities | As a seller deletion of a product is possible | API is available |
| US\_03 | Buyer Functionalities | As a buyer able to bid for a product | API is available |
| US\_04 | Seller functionality | As a seller list all bids received on product put for auction | API is available |
| US\_05 | Buyer functionality | As a buyer able to update the Bid Amount of my bidding | API is available |

**A Deployment representation can be as follows:**

****

# Platform

## Design Specification - 1

Use User Story-1 from the Business Requirements to implement the below.

1. Implement this user stories as serverless using Azure Functions.
2. Currently it is implemented as microservice application. Your ask is to Create an Azure function to add a new product for auction. This needs to interact with a RDBMS. Choose an appropriate RDBMS support on AWS to hold the DB for your application. You can choose language from Node.js, Python, Java, C# to create Azure Function.
3. Use Azure API management to expose the Azure function built in the previous step to be accessible on public internet.
4. Each time a user adds a new product for auction, a write occurs in the Realtime Database. Configure this write event to trigger a function to create Azure ANS Notification (email: configure a fixed email) to let the appropriate users know that they have new product added.
5. Configuring static concurrency settings while creating Azure Function, configure the concurrency of your serverless implementation to run max 100 instances of your Azure functions in parallel.

## Design specification - 2

Use User Story-3 from the Business Requirements to implement the below.

1. Deploy user Story-3 using Azure App Services.
2. For this purpose, add an appropriate application.yaml file in sample project microservice with user-story 3 and configure it.
3. Configure network security group of the Azure App Service instance created by to receive all traffic, including direct requests sent from the internet.
4. Configure Point-in-time snapshot to provide a timeline of key-value changes

## Design specification – 3

Use all other user-stories/microservice from the Business Requirements to implement the below.

* + - 1. Containerize this requirement on Azure Containers. Use Serverless Docker containers
      2. Configure Azure Log Monitor for logging of API operations and autoscaler operations
      3. Fetch All state values of container from Essentials in the Overview blade and export it to Azure Blob Storage