**Q3 - SCENARIO**

A Toy Retail company ToyTrex has its retail application deployed as 3-tier application - Web App (UI), Web API (middle layer) and Database as Azure SQL.

The user load started increasing multiple fold every month and complex programs getting implemented, the application started performing poorly.

As a result, company decided to re-architect the middle layer as microservices using Azure Kubernetes Services.

The new architecture has below design decisions.

*1) The middle layer should be implemented as Microservices using Azure AKS*

*2) The middle layer API should be deployed as containerized application images*

*3) The container images will use Azure Container Repository (ACR) as the private image repository*

*4) The CI/CD pipelines for microservices should be implemented using Azure DevOps services.*

*5) The Azure DevOps should be able to access ACR and download the container images for microservices deployment*

*6) The image should be deployed as templates such as <image\_name>:<build\_id>*

*Explain the DevOps configuration and steps in detail for above requirements*

**Ans, Scenario\_3: (My DevOps implementation with AWS Cloud)**

* **“All below steps (CI/CD) can be achieved by implementing CI tool Jenkins Or AWS in-house Service. There will be list of jobs get created which can build microservice as well as Docker image and store image into repository and release/deploy image microservice into respective destination environment.”**
* **Considering the internet facing Web & API microservice, the request should come from and reaches to service via respective process ::: <internet->ALB/NLB->TGP(TargetGroup, port mapping)->API/Web UI>**
* *The middle layer should be implemented as Microservices using Azure AKS*

In-order to refactor the existing middleware service into microservice, need to understand the stack and make necessary change (Development prospective, like springboot service) and create microservice artefact.

Also make the microservice with environment profile basis by separating application properties file w.r.t to various environment. Dev, QA, Prod Etc.

* *The middle layer API should be deployed as containerized application images*

Once we are good with built microservice artefacts, the application/API should prepare for containerization by using Docker implementation.

Prepare Dockerfile accordingly w.r.t microservice and checks in to GitHub/AWS Codecommit. And build Docker Image of respective microservice

* *The container images will use Azure Container Repository (ACR) as the private image repository*

Upon building docker image, artefacts docker image stored in any docker compatible storage repository, like AWS ECR (Elastic container Registry) or ACR.

Also, certain TAG can be maintained based on Build Number or specific TAG in-order to maintain Docker image version.

* *The CI/CD pipelines for microservices should be implemented using Azure DevOps services.*
* *The Azure DevOps should be able to access ACR and download the container images for microservices deployment*

In-order to prepare CI/CD pipeline, either we can implement through Jenkins job (Cost saving) or Can be use in-house service i.e. AWS developer service like codebuild, codepipeline service process.

The pipeline jobs should connect to ECR/ACR, and pull respective docker image microservice on TAG basis and make ready to deployment in respective Environment like either **ECS or EKS.**

Considering current microservice, it would be good recommended to use ECS, by considering application size and cost saving perspective...

* *The image should be deployed as templates such as <image\_name>:<build\_id>*

Yes, the images are will get stored in ECR/ACR with format like <Image\_Name> with either build\_ID or Tag.

While release/deploy into environment, images are pulled w.r.t <image\_name>:<build\_id> or Tags.

Below I prepare sample microservice implementation in AWS, with help of ECR, ECS/EKS .

 