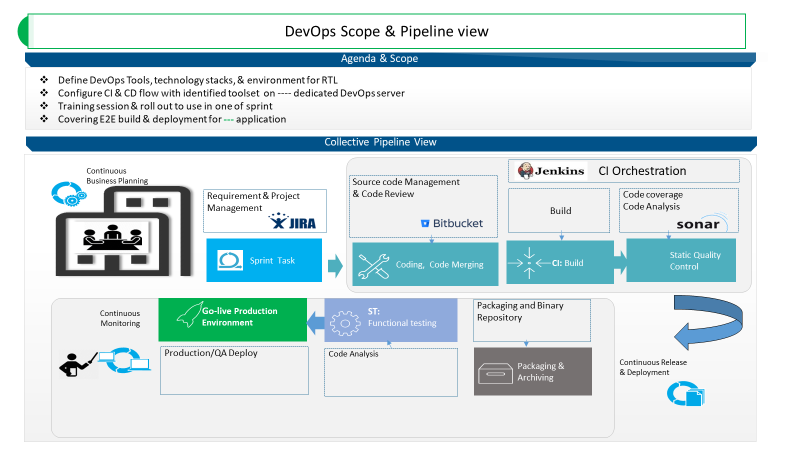
**Basic/Generic DevOps Approach:**

* I have prepared below DevOps design implementation which can fit with all below 3 scenarios with CICD implementation on Cloud.
* Key outcomes: Designing and developing scalable, low latency, secure and highly available platform system for huge real time data pipeline and high amount of user with smooth software development process.



**Q1 - SCENARIO**

A car rental company called FastCarz has a .net Web Application and Web API which are recently migrated from on-premise system to Azure cloud using Azure Web App Service

and Web API Service.

The on-premises system had 3 environments Dev, QA and Prod.

The code repository was maintained in TFS and moved to Azure GIT now. The TFS has daily builds which triggers every night which build the solution and copy the build package to drop folder.

deployments were done to the respective environment manually. The customer is planning to setup Azure DevOps service for below requirements:

*1) The build should trigger as soon as anyone in the dev team checks in code to master branch.*

*2) There will be test projects which will create and maintained in the solution along the Web and API. The trigger should build all the 3 projects - Web, API and test.*

*The build should not be successful if any test fails.*

*3) The deployment of code and artifacts should be automated to Dev environment.*

*4) Upon successful deployment to the Dev environment, deployment should be easily promoted to QA and Prod through automated process.*

*5) The deployments to QA and Prod should be enabled with Approvals from approvers only.*

Explain how each of the above the requirements will be met using Azure DevOps configuration.

Explain the steps with configuration details.

**Ans: Scenario\_01 (My implementation with AWS Cloud)**

The car rental company called FastCarz has Web Application and Web API which are recently migrated from on-premise system to Azure cloud using Azure Web App Service

and Web API Service. The on-premises system had 3 environments Dev, QA and Prod

**AWS codecommit:**

* Consider the entire codebase is in AWS Codecommit Git VCS (version control system)
* Implement PR (Pull Request) approval functionality in AWS Codecommit service which can restrict user’s direct code merge with branch. And allow only merge code with branch with PR approval.

**Environment (Dev | QA | Prod): AWS ECS service with three cluster for dev qa and prod respectively:**

* Create three cluster for 3 environment desired resource.
* Maintain Autoscaling respectively for cluster as well as service level.

**Orchestration Tool (Jenkins) for CI/CD:**

* List of Jenkins jobs will get created for all 3 projects, i.e. Web, API & Test
* In-order to achieve immediate build and deliver Artefacts.

Please find answers in green color:

* *1) The build should trigger as soon as anyone in the dev team checks in code to master branch.*

In Jenkins job, poll SCM mechanism can be implemented to trigger the build as soon as team checks in the changes into master branch.

* *2) There will be test projects which will create and maintained in the solution along the Web and API. The trigger should build all the 3 projects - Web, API and test.*

Jenkins pipeline job can be created to achieve such requirement, i.e. as long as the job get trigger it can cover the pipeline for Web and API. And the same for all 3 projects.

* *The build should not be successful if any test fails.*

The Jenkins job configuration can be done is such a way that the job should not proceed for build if in case found any such hiccups i.e. if the test case gets failed. This can be achieved by implementing check point in the Jenkins job configuration like conditional plugins in the pipeline. If there any test case failure or static SONARQUBE failure the job gets break the build and notify to the respective team over the email or Slack/teams channel.

* *3) The deployment of code and artifacts should be automated to Dev environment.*

The Jenkins job should be prepared in a way that the pipeline job should build the artefacts and deploy the same over Dev Environment by providing dev env credential/argument/parameter details*.*

* *4) Upon successful deployment to the Dev environment, deployment should be easily promoted to QA and Prod through automated process.*

Once the Dev env release get complete and upon successful functional/sanity testing, the respective artefacts can be freezed with TAG and the same can be stored in any artefacts storage tool like Nexus or JFrog or AWS ECR (Docker Image artefacts), and the respective TAG artefacts can be released to the next env (QA & Prod) through Jenkins automation process job.

* *5) The deployments to QA and Prod should be enabled with Approvals from approvers only.*

While creating CD job, approval mechanism can be implemented by installing/configuring approval process with the help of **Promotion plugins**. Upon Job A approval the next pipeline job B will get trigger.

*End….*

**Q2 - SCENARIO**

Macro Life, a healthcare company has recently setup the entire Network and Infrastructure on Azure.

The infrastructure has different components such as Virtual N/W, Subnets, NIC, IPs, NSG etc.

The IT team currently has developed PowerShell scripts to deploy each component where all the properties of each resource is set using PowerShell commands.

The business has realized that the PowerShell scripts are growing over period and difficult to handover when new admin onboards in the IT.

The IT team has now decided to move to ARM based deployment of all resources to Azure.

All the passwords are stored in a Azure Service known as key Vault. The deployments needs to be automated using Azure DevOps using IaC(Infrastructure as Code).

*1) What are different artifacts you need to create - name of the artifacts and its purpose*

*2) List the tools you will to create and store the ARM templates.*

*3) Explain the process and steps to create automated deployment pipeline.*

*4) Create a sample ARM template you will use to deploy a Windows VM of any size*

*5) Explain how will you access the password stored in Key Vault and use it as Admin Password in the VM ARM template.*

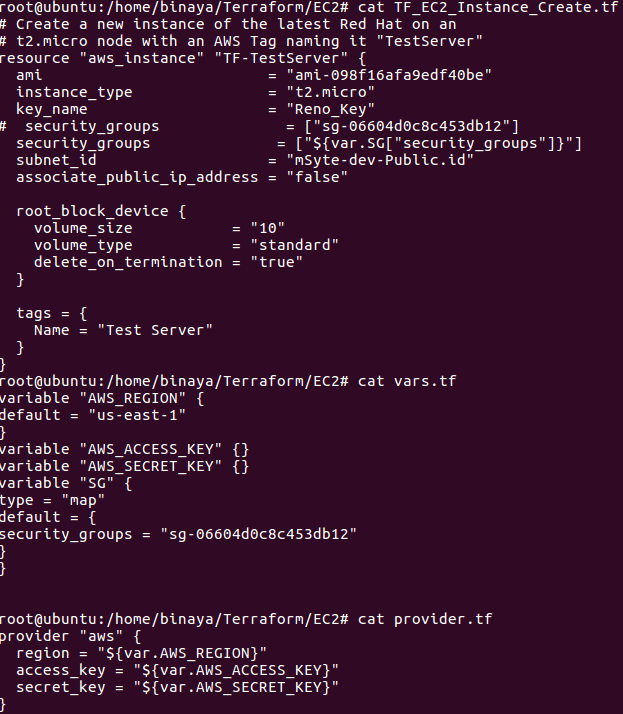
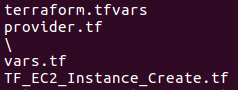
- In-order to prepare Infrastructure as Code, we can achieve the same by various approaches like AWS Cloudformation Tempelate, or ARM or Terraform Tool Or Cloud CLI command script.

- Either one of the ways the infrastructure can be created by implementing automation script.

- I can prepare aws CFT or Terraform to make infrastructure spin up and provide service.

- I recommend Terraform open source tool for IaC, as this is compatible for any kind of infrastructure creation like AWS, GCP, Azure etc...

Below are sample Terraform script to prepare AWS EC2 machine. Note: The same can be prepared using either AWS CLI command or CFT or ARM

**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 .

