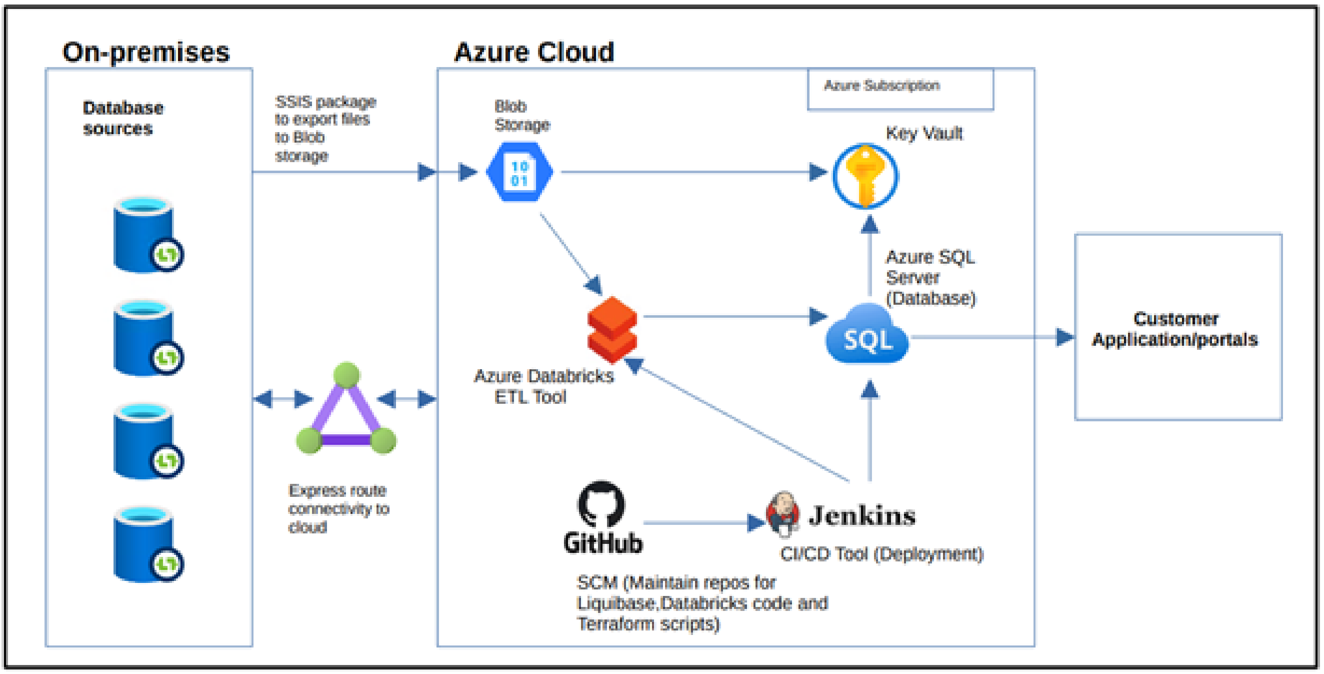
DevOps Technical Guide

The objective is to showcase the design of databricks continuous integration and continuous delivery pipeline.

High Level Solution


On-premises
Database sources
SSIS package to export files to Blob storage
Express route connectivity to cloud
Azure Cloud
Azure Subscription
Blob Storage
Key Vault
Azure SQL Server (Database)
Azure Databricks ETL Tool
Customer Application/portals
GitHub
Jenkins
CI/CD Tool (Deployment)
SCM (Maintain repos for Liquibase,Databricks code and Terraform scripts)

The Client Azure Data Infrastructure is part of the strategic initiative to create a New Data Platform in Azure Cloud for building data domains and products from source system data. There are various data sources which are needed to be transitioned into the cloud environment.

Basically, the environment consists of five major Azure resource. Azure blob storage, Azure Databricks, Azure SQL server (Database), Key vault and Express route connection. Following are the points highlighting the significance of each of the resource mentioned earlier,

- On-Prem environment and azure cloud environments are connected via express route, establishing a secured and private channel between the environments. (Owner of this task – Platform engineering team)
- Data is exported to Files from the On-Premises Clone of Data source.
- Data is copied as csv files to Blob Store and securely transferred using SSIS packages from On-prem environment. All the traffic is secured and encrypted using TLS 1.2 protocol.
- Databricks ETL is used to Extract, Transform and Load data into the Client Database.
- Every secret, key, password used to connect to any of the resource is stored in Azure key vault and should be retrieved from the Vault. Storage account keys and Azure DB connection strings are the some of the examples which are stored in vault. This helps in connecting Databricks to the blob and SQL without passing the resp. credentials in plain text.
- All Databricks code is stored in Github.
- Terraform scripts used for the resource creation in Azure subscription should be maintained in Github repo.
- Liquibase is used for DDL and updating the SQL Server Database
- DevOps Pipelines are used for managing the Azure SQL data structure using DDL scripts stored in Github.

Known configuration of the resources used in current setup

Resource Name	Details	Dev	Test	Production	Owner
Storage Account	Storage Account	Access tier - Hot tier	NTD	NTD	LV Team
		Account - StorageV2 (general purposev2) (Standard)	NTD	NTD	

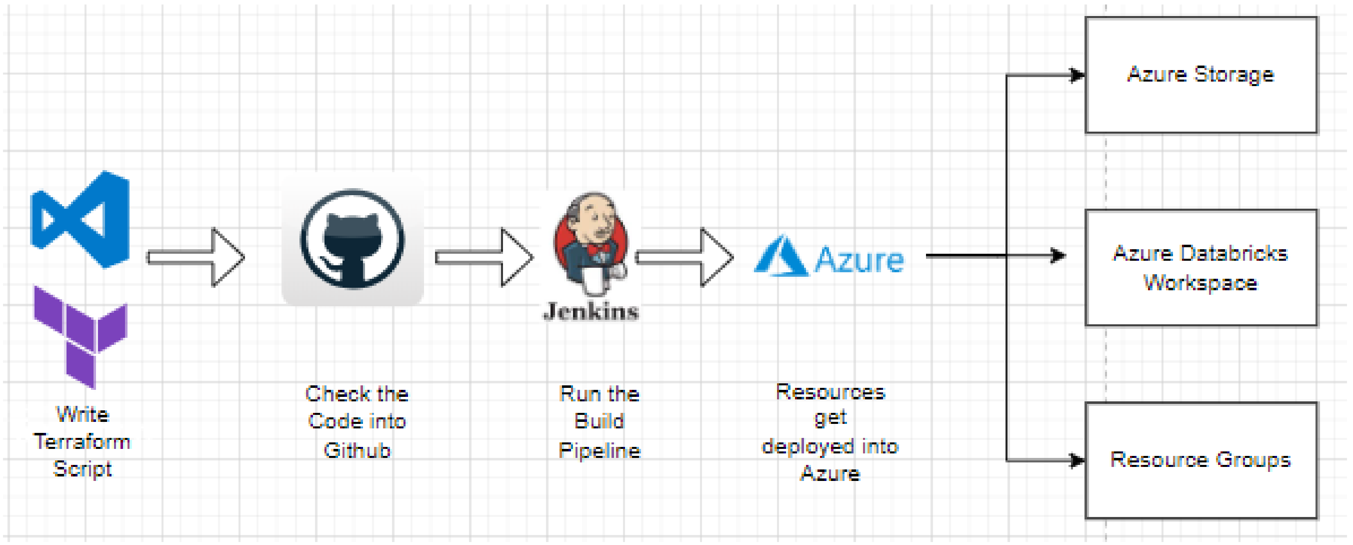
		Redundancy - LRS	NTD	NTD	
Azure SQL DB	Azure SQL DB	Pricing tier - Basic: 50 eDTUs	NTD	NTD	LV Team
		Service tier - Elastic pool	NTD	NTD	
		Redundancy - LRS	NTD	NTD	
Azure Databricks	Azure Databricks	Pricing tier - Premium	NTD	NTD	LV Team
		All-Purpose Compute Workload	NTD	NTD	
Azure Key Vault	Azure Key Vault	Pricing Tier - Standard	NTD	NTD	LV Team
		Soft delete - 90 days	NTD	NTD	
Jenkins Server Pipeline	Create ADB resources using Terraform	Created and deployed by LV Team			LV Team
	Create SQL Database resources using Terraform	Created and deployed by LV Team			LV Team
	ADB code deploy	Single pipeline deploying to all 3 environment with authorization. Jenkins server will be provided by cloud engineering team along with all necessary package preinstalled like python, databricks cli, databricks-connect, request, pytest			Hex Team
	ADB database deploy	Single pipeline deploying to all 3 environment with authorization. Jenkins server and Azure SQL DB will be provided by cloud engineering team along with all necessary package preinstalled like Java, Liquibase			Hex Team
Git repository	Terraform Infra - ADB resource	Needs to be provided by LV Team			LV Team
	Terraform Infra - SQL resource	Needs to be provided by LV Team			
	Azure Databricks Code	Needs to be provided by LV Team			
	Azure Databricks database (Liquibase)	Needs to be provided by LV Team			

Jenkins pipeline

Majorly 4 Jenkins pipeline will be created for deployment of various azure resources and CICD perspective.

1. Pipeline for Azure Databricks resource creation in Azure, using Terraform script.
2. Pipeline for Azure SQL DB resource creation in Azure, using Terraform script.
3. Azure Databricks code deployment
4. Pipeline for Database deployment (Liquibase pipeline)..

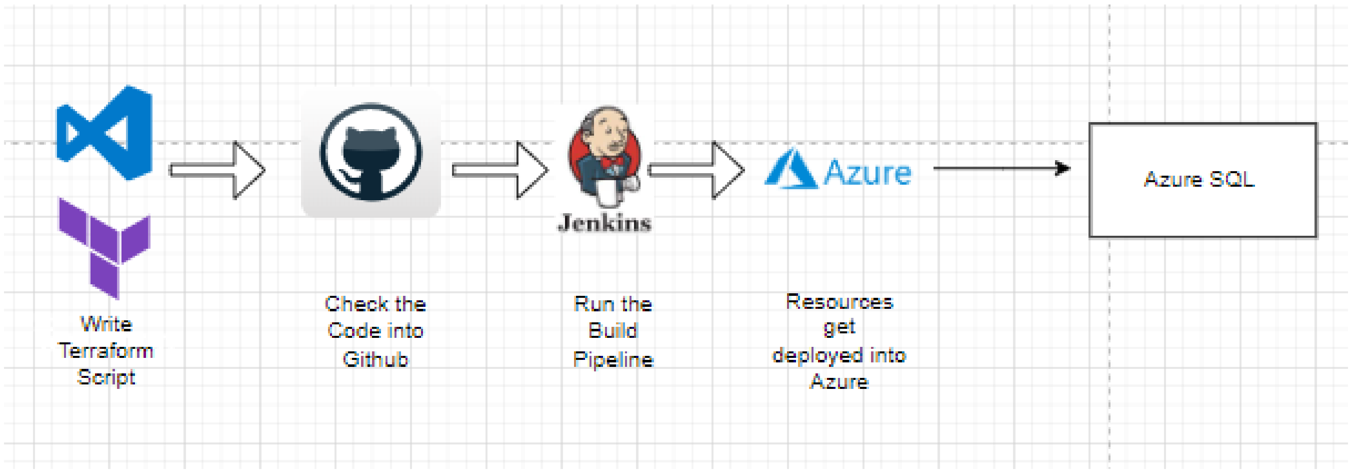
Pipeline to create Azure Databricks resources using terraform


Write Terraform Script
Check the Code into Github
Jenkins
Run the Build Pipeline
Azure
Resources get deployed into Azure
Azure Storage
Azure Databricks Workspace
Resource Groups

Databricks associated resources will be created and managed using terraform, through Jenkins pipeline. It will have workflow as shown above.

1. Write terraform script: Terraform script will be written using any of the IDE like VS code. One has to validate terraform script and create terraform plan to check for the desired output and resource creation.
2. Check the code into github: Once the terraform code is ready, It will be committed in to github repository for versioning and CICD purpose.
3. Build pipeline: Once the code is committed into github, Jenkins job will be triggered using webhooks. It will check for azure subscription connectivity and executed the terraform script to create resources
4. Azure Resources: On the successful completion of build we can see azure resources are now got deployed to specified azure subscription.

Pipeline to create Azure SQL database resources using terraform



This pipeline will be specific to SQL resource deployment only. It is similar kind of pipeline which we saw earlier.

Pipeline to deploy Azure Databricks Code

Environments –

Dev - This environment will be mainly used by developer for development purposes. Actual development code will be stored. Developers do their required changes here.

Test – This environment will be reserved for testing purpose only. Tester will do sanity and functional testing on this environment Once the code is tested, UAT will be done on this environment.

Production – It will be production environment.

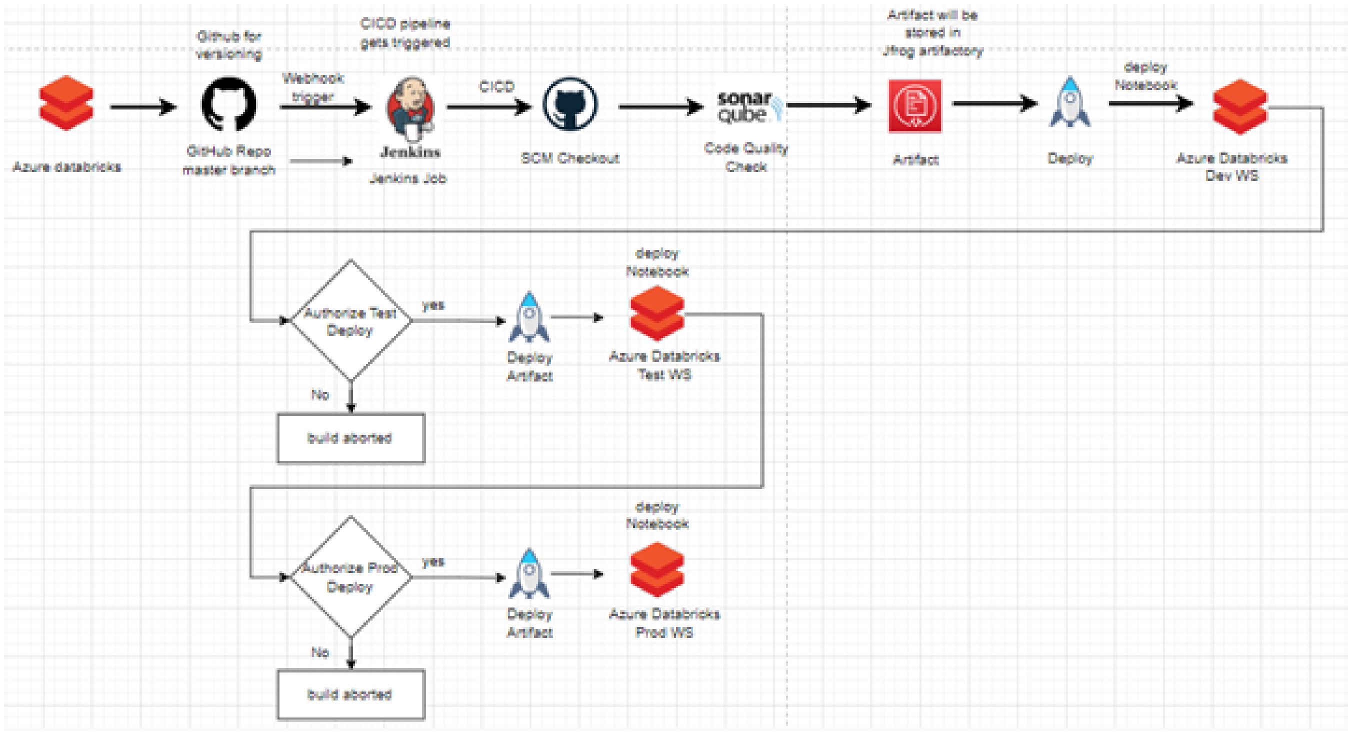
Separate azure databricks workspace will be created for each environment. We have 3 different environments to work upon namely Development, Test and Production, so 3 different ADB workspace will be created.

Jenkins workflow –

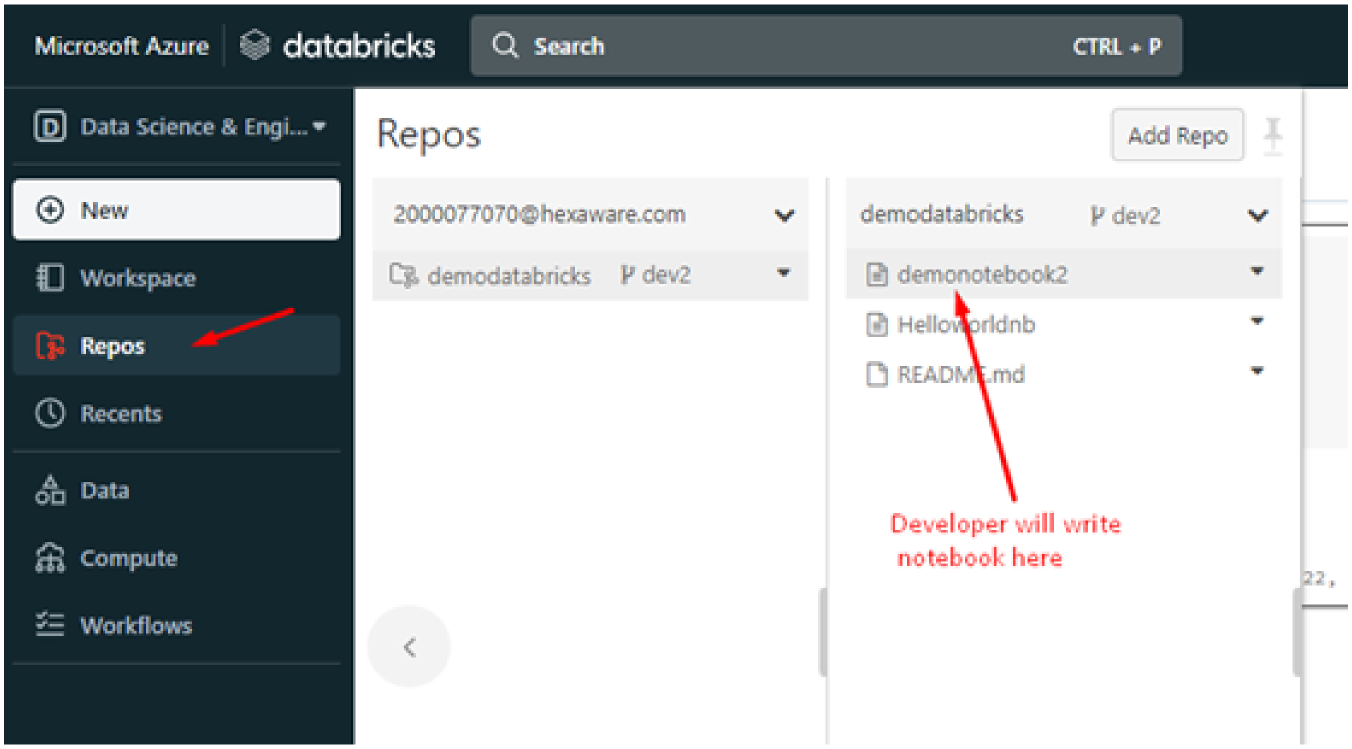
Prerequisite – Python should be installed along with below python packages

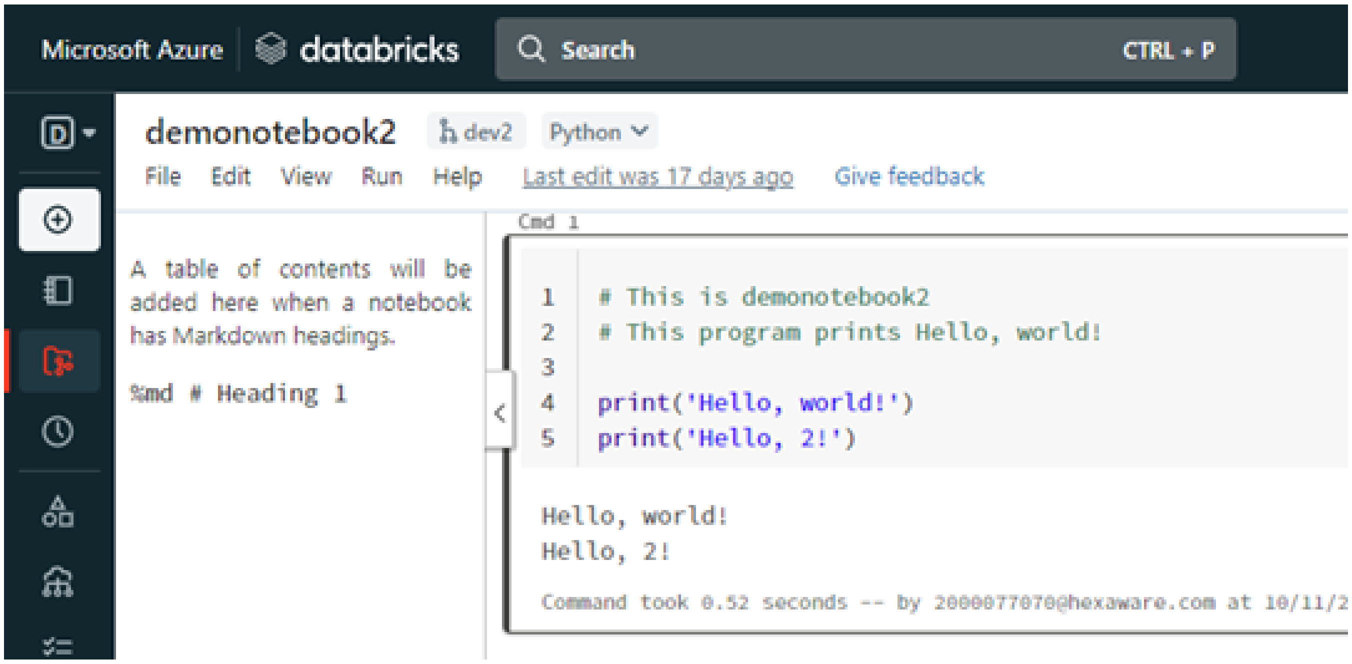
Python libraries: requests, databricks-connect, databricks-cli, and pytest.

Typical databricks CICD workflow will mainly include below steps.



Start with Notebook creation - One has to create a notebook in the databricks Repos. Take the latest pull of the dev branch then create a new feature branch based on the dev branch. Code will be developed and committed to the feature branch.





A table of contents will be added here when a notebook has Markdown headings.

```
%md # Heading 1
```

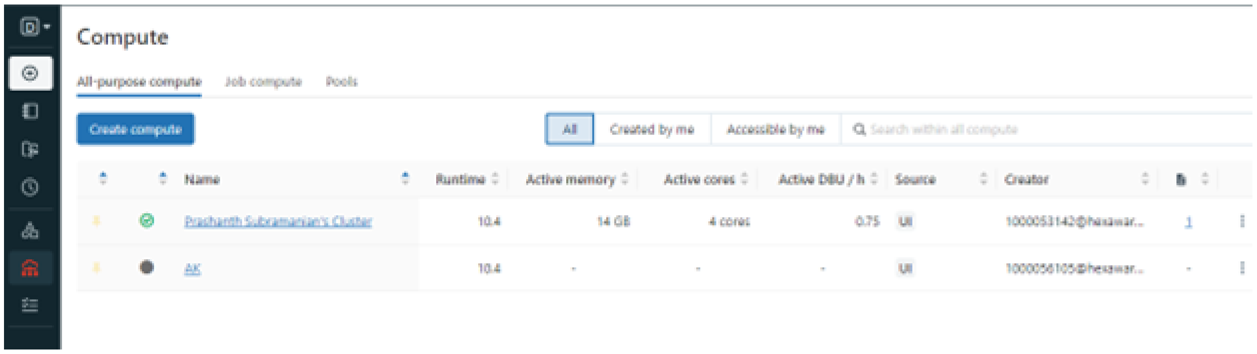
```
# This is demonotebook2
# This program prints Hello, world!

print('Hello, world!')
print('Hello, 2!')
```

```
Hello, world!
Hello, 2!

Command took 0.52 seconds -- by 2000077070@hexaware.com at 10/11/2
```

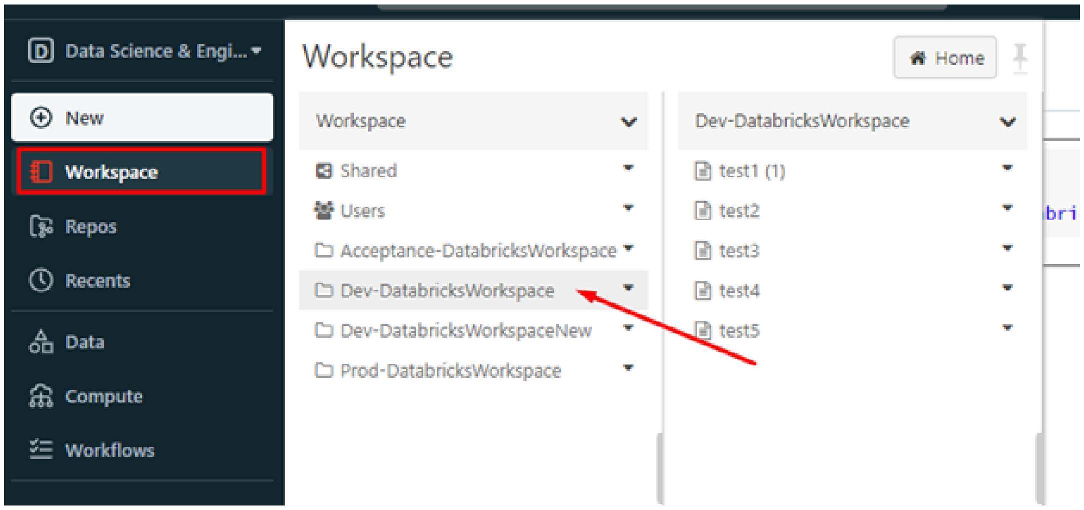
Cluster: Azure Databricks cluster is a set of computation resources and configurations on which notebook will be compiled and executed.



Name	Runtime	Active memory	Active cores	Active DBU / h	Source	Creator	
Prashanth Subramanian's Cluster	10.4	14 GB	4 cores	0.75	UI	1000053142@hexawar...	1
AK	10.4	-	-	-	UI	1000056105@hexawar...	-

Once the development is completed, pull request needs to be raised to merge this code into master branch and to trigger Jenkins pipeline.

- Jenkins Job: Once reviewer approves the pull request, Jenkins pipeline job will get triggered. It will start building the code. Using github webhooks Jenkins job will get triggered.
- SCM Checkout: In this stage latest pull of the specified github repo will be taken.
- SonarQube: SonarQube Analysis will be done here for code quality check. Provision of SonarQube will be done by platform engineering team.
- Artifact: On the successful completion of quality check, Artifact will be generated and will get stored in one of the antifactory like Jfrog antifactory.
- Deploy: In this stage generated artifact will be deploy into azure databricks dev workspace using databricks Cli.



- Authorize Test Deploy : Authorization will be needed to continue further execution of pipeline. Once test push is authorized, artifact will be deployed to Azure Databricks test workspace.
- Authorize Prod Deploy : Authorization will be needed to continue further execution of pipeline. Once Production push is authorized, artifact will be deployed to Azure Databricks production workspace.

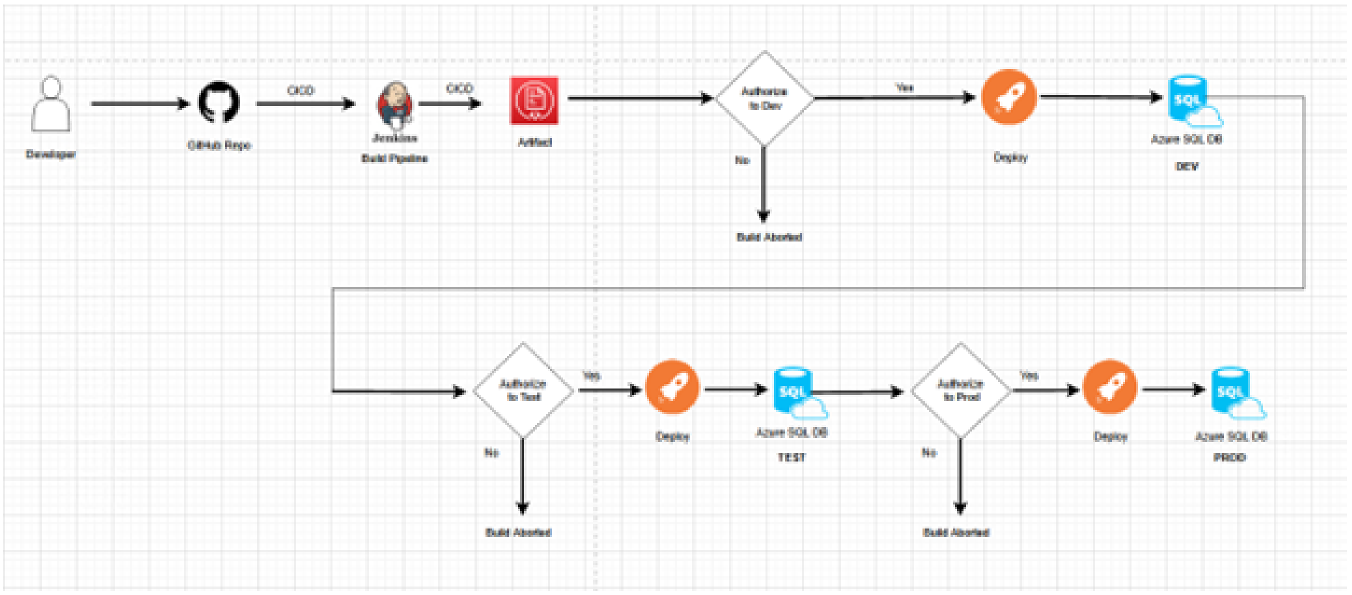
Databricks Job –

As of now creation and scheduling of databricks job in all the environment will be manual task. As time passes there will be number of jobs in ADB, Managing and scheduling/rescheduling the jobs with the Jenkins will be more complex task.

Azure KeyVault –

We will be using azure keyVault to store all the credential which are getting used in the Jenkins pipeline. Provision of Azure KeyVault needs to be discussed with platform engineer team.

Pipeline for database deployment (Liquibase pipeline)



DDL script execution is done using Jenkins Pipeline with the help of Liquibase, and the Pipeline code and configurations are stored on Github. Secrets like database connection, username, and passwords must be stored securely in Jenkins Keystore (Need to have Azure key vault integrated). No direct access from the SSMS client to the production and test databases is allowed.

Components:

- GitHub :- Repository will be present in GitHub contains SQL scripts.
- Jenkins :- This is used to create a Build pipeline that have various stages.
- Maven:- This is a Build tool.
- Azure SQL DB :- This a database which is updated according to SQL scripts through pipeline.

Pipeline Brief Explanation :-

- GitHub Repo having below files

1. sql :- This file contains changesets which needs to be deploy to target database.
2. xml :- This files contains Liquibase-maven plugin & Jdbc- dependency.
3. properties :- This file contains target database URL information and path of changelog.sql file.
4. Jenkinsfile :- This file contains various stages of pipeline.

- Developers can write there sql scripts into there local branch in GitHub Repo. If script is working fine then the developer commit there changes to master branch.
- When the changes are committed to master branch the Jenkins pipeline will be trigger.
- At this stage Artifact will be created then Authorization will be needed to continue further execution of pipeline. Once Dev push is authorized, artifact will be download first then deployed to Azure SQL DB Dev environment.
- Then Authorization will be needed to continue further execution of pipeline. Once Test push is authorized, artifact will be download first then deployed to Azure SQL DB Test environment.
- Then Authorization will be needed to continue further execution of pipeline. Once Prod push is authorized, artifact will be download first then deployed to Azure SQL DB Prod environment.