[[\_TOC\_]]

# \*\*Introduction\*\*

## \*\*Purpose\*\*

The purpose of the GTO PRIME Infrastructure Deployment Document is to understand the structure and process flow of deploying Azure resources in a codified fashion on Microsoft Azure Cloud Platform.

The team should be able to create the infrastructure along with post deployment activities and manual activities with the help of this documentation. After the infrastructure is created the deployment of frontend and backend codes can be done in the same infrastructure.

## \*\*Related Documents\*\*

Document Name : Solution Design Document

# \*\*Scope of Work\*\*

### \*\*Scope\*\*

The things which will be covered by ellicium deployment team are as follow:

1. IaC for infrastructure of GTO PRIME

1. Release Pipelines

1. Deployment of code

### \*\*Out of Scope\*\*

The steps of deployment which will be done by AlixPartners are as follows :

1. Management of Infrastructure

1. Creation and providing access of Active Directory Group

1. Reports of backend and frontend codes

1. Log Analytics and Event Hub.

# \*\*Prerequisites\*\*

1. The prerequisites needed to deploy infrastructure and codes are as follows:

2. Microsoft Azure account subscription

3. Microsoft Azure DevOps

4. There should be a ‘\*\*Sec\_devops\_directoryreader\*\*’ group in active directory with ‘\*\*Directory readers\*\*’ role Assigned.

5. There should be a ‘\*\*Database\_Admin\_Group\*\*’ in active directory which will be set as sql server admin in release pipeline.

6. Devops service principals should have a '\*\*Privileged role administrator\*\*' role.

7. Devops service principal should have following Microsoft Graph Api permissions and \*\*grant admin consent.\*\*

1. User.ReadWrite.All

2. Directory.ReadWrite.All

3. GroupMember.ReadWrite.All

4. Application.ReadWrite.All

8. Devops Service principal should be assigned ‘\*\*Owner\*\*’ and ‘\*\*Storage Blob Data Contributor\*\* ’ role on subscription.

## \*\*Developer Group Resource Permissions\*\*

- When we start the development

1. Databricks : Workspace Access

2. Key vault : Access Policy Secret Permissions : GET, LIST, SET, DELETE, RECOVER, ACKUP, RESTORE

3. Data Factory : Datafactory Contributor

4. SQL server : Database Access to Developer Group

5. ADLS : Storage Blob Data Contributor

# \*\*Architecture Overview\*\*

## \*\*GTO-PRIME Infrastructure architecture\*\*

Design includes following components :

1. Azure ADLS Storage : To store the data in Raw form as well as in Transformed form.

2. SQL Database: To maintain the data in the Transformed and curated form.

3. DataBricks : To develop Pyspark Jobs.

4. Vnet : Databricks Vnet with configured Address space.

5. Azure Data Factory : To create and orchestrate pipelines of databricks jobs.

6. Azure keyvault : To store the credentials of different services.

7. Azure Devops : To implement CI/CD and to deploy code from one environment to another.

## \*\*GTO-PRIME IaC Deployment Repository Structure:\*\*

1. Deployment directory at local

2. Push the Deployment folder to Azure DevOps Repository

3. The Deployment directory contains four sub directories:

4. Artifacts Directory :

- Template folder contains all the ARM templates to create individual resource groups and resources.

- The template files are linked together in the azuredeploy.json file.

5. Build Directory :

- Contains buildPipeline.yaml file

- Changes in master branch triggers the build

6. Documentation Directory :

- Contains Readme.md file

7. Scripts Directory :

- Contains all the powershell and SQL DDL scripts to run all the ARM templates and to run the post deployment activities.

# \*\*Deployment Process Flow\*\*

## \*\*IAC Build:\*\*

1. The Build pipeline will be used to scan ARM templates and create an artifact which then will be used in the IAC deployment release pipeline.

2. Any change to the main branch of the repository will trigger the build.

3. Artifacts contain all the templates and powershell scripts to create infrastructure.

4. Build Pipeline:

- Create a new build pipeline under pipelines.

- Choose Azure Git Repos.

- Select the repository which contains the ‘buildpipeline.yaml’ file.

- In configure , choose ‘Existing azure pipeline YAMLfile’.

- Select the branch and provide the path to build the YAML file and continue.

- Build Name Standard.

AP.<ProjectRepositoryName>.<API/UI/BackEnd/IaC>

- Add the required variables in the variables section.

- Save and Run the build.

- After successful run of build an artifact will be created which will be used in the Deployment release pipeline.

## \*\*Frontend Build\*\*

1. The steps to create Front End Build are as follows:

- Select the repository for front end and continue

- Select empty job

- Add task for npm install

- Add task for npm build: Add display name as “npm custom”, select command as “Custom”, Add command and argument as “run build”

- Add task for publish build artifacts

- Add path to publish as “build” and save the pipeline

## \*\*Deployment Release Pipeline\*\*

1. In the release pipeline all resources are deployed using powershell scripts present in the build.

1. This release pipeline will also deploy all the Databricks notebooks, Datafactory pipelines and the Application in the deployed infrastructure.

1. repositories and 2 artifacts have been linked in artifacts section

1. This release pipeline is connected to 2 Repositories and 3 Builds which is as follows:

1. AP.PRIME.IAC (Build)

1. AP.PRIME.UI (Build)

1. AP.PRIME.API (Build)

1. AP.PRIME.Databricks

1. AP.PRIME.Datafactory

## \*\*Release Pipeline\*\*

1. Create a new release pipeline in releases.

1. Release Name Standard

1. The individual releases should follow a naming standard that identifies the project and environment for the release.

2. The Release naming standard should follow:

- AP.<ProjectRepositoryName>.<Dev/QA/Prod>

1. Choose ‘Empty Job’ in the template section.

1. Add required Names and values in release pipeline variables with the same names and their values as described in the \*\*table below.\*\*

1. Provide firewall Rules for SQl server and ADLS.

2. Add required Databricks Vnet configurations in the Release pipelines variables section.

3. The AD user can get token with this command :

``` az account get-access-token --resource 2ff814a6-3304-4ab8-85cb-cd0e6f879c1d --query "accessToken" ```

4. Additionally, the user who generates this token will have to login into created databricks, or else key vault backed secret scopes task will throw an Error message :

5. Error: Authorization failed. Your token may be expired or lack the valid scope

| Name| Value Description |

|--|--|

| ADAdminName | Admin Name for Sql server (primarily Devops service principal name) |

| BillCode | Self Explanatory |

| suffix | Suffix for resource names if any |

| budgetOwnerTag | Name of budget owner |

| DatabaseSchemaName | dbo |

| departmentTag | Self Explanatory |

| directoryReaderGroup | AD Group with Directory Reader Role |

| Environment | DEV / UAT / PROD |

| ProjectCodeName | Self Explanatory |

| ResourceGroupLocation | Location where we want to deploy the Resources Ex : eastus2|

| senderemailaddress | sendgrid sender email address |

| sendgridapikey | sendgridapikey |

| sqladminLogin | Login Name to be set in sql server |

| system.debug | "True" , To debug any errors in the release pipeline. |

| TenantId | Self Explanatory |

| SubscriptionId| Self Explanatory |

| ProjectResourceIdentifier| 2-4 Characters – Project Resource Identifier; typically, an acronym of the project code name |

| createdby| Person Name who is creating resource |

| virtualNetworkIP| IP Address range for VNET (Databricks) E.g. : 10.8.0.0/22 |

| VnetStartAddress| Vnet start ip for Sql Server firewall rule E.g. : 10.8.0.0 |

| VnetEndAddress| Vnet end ip for Sql Server firewall rule E.g. : 10.8.8.255 |

| subnet1IP| Subnet Range for the VNET E.g. : 10.8.1.0/24 |

| subnet2IP| E.g. : 10.8.2.0/24 |

| subnet3IP| E.g. : 10.8.3.0/24 |

| FirewallRule1| IP address 1 for sql server and adls E.g : 94.2.XXX.XXX |

| FirewallRule2| IP address 2 for sql server and adls |

| FirewallRule3| IP address 3 for sql server and adls |

| FirewallRule4| IP address 4 for sql server and adls |

| FirewallRule5| IP address 5 for sql server and adls |

| TriggerTime| Trigger time (UTC now) for ADF master pipeline E.g : 2022-09-22T04:01:01Z |

| DbAdminGroup| Database Admin Group name To be set as admin for sql server |

| adtoken| Azure User AD token required for Databricks Key vault backed secrets scopes |

5. Add Stage Name as ‘DEV’ and Stage owner name.

6. Add created artifact of IaC repository in the artifact section.

7. Add created artifact of Frontend/UI repository in the artifact section.

8. Add Backend/API repository in the artifact section.

9. Add Databricks repository in the artifact section. Databricks repository will have ‘main’ as default branch.

10. Add Datafactory repository in the artifact section. Datafactory repository will have ‘adf-publish’ as the default branch.

### \*\*Release pipeline Tasks\*\*

There are 18 tasks in release pipeline :

### 1. Task 1: Deploy Infrastructure

Choose and configure Azure Powershell task as follows :

Add specifications as follows:

![image.png](/.attachments/image-b98bcbec-2e59-49c6-a3e5-d925a0b80818.png)

![image.png](/.attachments/image-b18bef86-4980-4a49-97be-16e1b425ef0e.png)

- Task 1 has an output variable named ‘Task1’.

- As soon as task 1 is over, the user who has generated the Azure AD token, will have to login into databricks. So that the last task which is key vault backed secret scopes works without any error.

### 2. Task 2: Grant Data Factory user access in database

Choose and configure Azure SQL Database deployment task as follows:

- Add specifications as follows:

![image.png](/.attachments/image-bff51bc4-e816-4aee-884d-825bb01a0c24.png)

![image.png](/.attachments/image-3fdd50ab-1c9f-4d85-82dd-79a99ad16804.png)

- Write inline SQL script as follows:

```sql

IF NOT EXISTS(SELECT [name] FROM sys.database\_principals WHERE [name] = '$(Task1.DataFactoryName)')

BEGIN

CREATE USER [$(Task1.DataFactoryName)] FROM EXTERNAL PROVIDER ;

GRANT EXECUTE ON SCHEMA::[$(Task1.Database\_SchemaName)] TO [$(Task1.DataFactoryName)] ;

GRANT SELECT ON SCHEMA::[$(Task1.Database\_SchemaName)] TO [$(Task1.DataFactoryName)] ;

END

```

### 3. Task 3: Remove sql server admin

- Choose Azure Powershell task

![image.png](/.attachments/image-61aec3cf-0056-4817-a3bb-e9eef931aec9.png)

- Write inline script as follows:

```powershell

$sqlservername = '$(Task1.sqlServerName)'

$ResourceGroupName = '$(Task1.ResourceGroupName)'

#Remove SQL server AD Admin if exists and if it is azure devops service principal

$sqlserveradmincheck = (Get-AzSqlServerActiveDirectoryAdministrator `

-ServerName $sqlservername `

-ResourceGroupName $ResourceGroupName).DisplayName

Write-Host 'sqlserveradmincheck' $sqlserveradmincheck

if ($sqlserveradmincheck -eq '$(ADAdminName)'){

Remove-AzSqlServerActiveDirectoryAdministrator `

-ServerName $sqlservername `

-ResourceGroupName $ResourceGroupName

}

# set database db developer group as admin

Set-AzSqlServerActiveDirectoryAdministrator `

-ResourceGroupName $ResourceGroupName `

-ServerName $sqlServerName `

-DisplayName '$(DbAdminGroup)'

```

### 4. Task 4: Databricks PAT Generation

- Choose and configure Azure cli task and add configuration

![image.png](/.attachments/image-c08122f2-9011-4ecb-91cc-cd85544b8791.png)

![image.png](/.attachments/image-e549a902-3690-4585-bd30-fde882f8bca5.png)

![image.png](/.attachments/image-36a14e1d-5686-47a8-8482-40efb9f55604.png)

1. Task 4 has an output variable named ‘Task4’.

### 5. Task 5: Install databricks cli

- Choose and configure command line task and add configuration as follows :

![image.png](/.attachments/image-ab74283a-557d-4759-9deb-969413d61e3c.png)

Script :

```

python -m pip install --upgrade pip setuptools wheel databricks-cli==0.11.0

```

### 6. Task 6: Databricks configure cli

- Choose and configure Bash task and add configuration as follows :

![image.png](/.attachments/image-0f94bb60-d880-4b62-9006-c03f6d68483b.png)

Script for Bash task :

```

databricks configure --token << EOF

$(Task4.workspaceUrl)

$(Task4.workspace\_token)

EOF

Databricks clusters list

```

### 7. Task 7 : Cluster Creation

- Choose and configure Azure Powershell task

### 8. Task 8: Copy Machine Learning Models

Choose and configure Azure file copy task

1. Azure file copy task works most of the time, rarely it fails giving the following error.

- Error : Storage account: <ADLSName> not found. The selected service connection 'Service Principal' supports storage accounts of Azure Resource Manager type only.

2. Container Name : gtoprimedev

### 9. Task 9: Remove AgentIp from ADLS Firewall Rule

Choose and configure Azure powershell Task

Inline Script :

```powershell

Write-Host "Remove AgentIp from ADLS Firewall Rule " - ForegroundColor Green

Remove-AzStorageAccountNetworkRule `

-ResourceGroupName $(Task1.ResourceGroupName) `

-AccountName $(Task1.ADLSName) `

-IPAddressOrRange $(Task1.AgentJobIp)

```

### 10. Task 10: Deploy Databricks Notebooks

Choose and configure Azure powershell task for uploading Databricks workspace.

### 11. Task 11: Deploy Datafactory pipelines

Choose and configure ARM Template Deployment task for deploying pipelines in Target Datafactory

Under Override template parameters :

```

factoryName : "$(Task1.DataFactoryName)"

AzureSqlDatabase2\_connectionString : ""

SqlServer\_connectionString : "Integrated Security=False;Encrypt=True;Connection Timeout=30;Data Source=$(Task1.sqlServerName).database.windows.net;Initial Catalog=$(Task1.sqlDBName)"

AzureDatabricks1\_properties\_typeProperties\_domain : $(Task4.workspaceUrl)

AzureDatabricks1\_properties\_typeProperties\_workspaceResourceId : $(Task1.DbrResourceId)

AzureDatabricks1\_properties\_typeProperties\_existingClusterId : $(Task7.clusterID)

AzureKeyVault1\_properties\_typeProperties\_baseUrl : "https://$(Task1.keyVaultName).vault.azure.net/"

Default\_properties\_keyvault\_name\_value : "$(Task1.keyVaultName)"

trigger1\_properties\_typeProperties\_recurrence\_startTime : "$(TriggerTime)Z"

default\_properties\_DbSchemaName\_value : "dbo"

default\_properties\_Phase4DbSchemaName\_value : "spkchevrondev"

```

### 12. Task 12: npm install

Choose and configure npm task as follows :

### 13. Task 13: Replace Tokens in secrets.js

Choose and configure Azure powershell task as follows :

- Inline Script :

```powershell

Write-Host "Replace Tokens in secrets.js " - ForegroundColor Green

$path = '$(System.DefaultWorkingDirectory)/\_GTO-PRIME-API/src/key/secrets.js'

$token = '#{KEYVAULT\_NAME}#'

$replace = '$(Task1.keyVaultName)'

((Get-Content -path $path -Raw) -replace $token , $replace) | Set-Content -Path $path

```

### 14. Task 14: Deploy Backend App

Choose Azure App Service Deploy task :

- Under File Transforms & Variable Substitution Options add

```-Handler iisnode -NodeStartFile src/index.js -appType node```

- Under Application and Configuration Settings add

```-WEBSITE\_NODE\_DEFAULT\_VERSION 16.13.0```

### 15. Task 15: Replace Tokens in index.html

Choose and configure Azure powershell Task as follows :

- Inline Script :

```powershell

Write-Host "Replace Tokens in index.html " - ForegroundColor Green

$path = '$(System.DefaultWorkingDirectory)/\_AP.GTO-PRIME-UI.UI/drop/index.html'

$word1 = '#{API\_URL}#'

$word2 = '#{AUTH\_APP\_ID}#'

$word3 = '#{AUTH\_REDIRECT\_URI}#'

$word4 = '#{AUTHORITY}#'

$replace1 = '$(Task1.WebAppApiUrl)'

$replace2 = '$(Task1.WebAppRegistrationAppID)'

$replace3 = '$(Task1.AuthRedirectUri)'

$replace4 = '$(Task1.Authority)'

((Get-Content -path $path -Raw) -replace $word1 , $replace1) | Set-Content -Path $path

((Get-Content -path $path -Raw) -replace $word2 , $replace2) | Set-Content -Path $path

((Get-Content -path $path -Raw) -replace $word3 , $replace3) | Set-Content -Path $path

((Get-Content -path $path -Raw) -replace $word4 , $replace4) | Set-Content -Path $path

```

### 16. Task 16: Deploy Backend App

Choose and configure Azure Web App Task as follows :

### 17. Task 17: Enable datafactory triggers

Choose and configure Azure powershell Task as follows :

- Script :

```powershell

$triggername = Get-AzDataFactoryV2Trigger `

-ResourceGroupName '$(Task1.ResourceGroupName)' `

-DataFactoryName '$(Task1.DataFactoryName)'

echo $triggername.name

Start-AzDataFactoryV2Trigger -ResourceGroupName '$(Task1.ResourceGroupName)' `

-DataFactoryName '$(Task1.DataFactoryName)' `

-TriggerName $triggername.name -Force

```

### 18. Task 18 : Key Vault backed secret scopes for Databricks

Choose and configure Azure powershell Task as follows :

An azure AD user can get token with this command :

```cli

az account get-access-token --resource 2ff814a6-3304-4ab8-85cb-cd0e6f879c1d --query "accessToken"

```

- Add the token in release pipeline variables.

- Script :

```powershell

python -m pip install --upgrade pip setuptools wheel databricks-cli --upgrade

$keyVaultResp = (Get-AzKeyVault -VaultName '$(Task1.keyVaultName)')

$KvResourceId = $keyVaultResp.ResourceId

$KvVaultUri = $keyVaultResp.VaultUri

$env:DATABRICKS\_AAD\_TOKEN = "$(adtoken)"

$hostname = "$(Task4.workspaceUrl)"+"/"

Write-Output 'hostname' $hostname $KvResourceId $KvVaultUri

databricks configure --host $hostname --aad-token

databricks secrets delete-scope --scope gto\_prime\_scope

databricks secrets create-scope --scope gto\_prime\_scope --scope-backend-type AZURE\_KEYVAULT --resource-id $KvResourceId --dns-name $KvVaultUri --initial-manage-principal users

databricks secrets list-scopes

```

- Click on the create release button on the top right corner to release the pipeline.

- Go to release and click on logs to check logs in the release pipeline.

# \*\*Manual Activities:\*\*

1. Grant admin consent to an App registration used for Web App

2. If Azure AD token is not provided in variables of release pipeline, Databricks key vault backed secret scopes will be a manual activity.

- Databricks : key vault backed secret scopes

Get databricks-instance

- Go to https://<databricks-instance>#secrets/createScope

- Enter the name of the secret scope ‘gto\_prime\_scope’.

- Select All Users from Manage Principal drop down.

- Enter the DNS Name (https://xxxxxxx.vault.azure.net/) and resource id (/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourcegroups/xxxxxxx/providers/Microsoft.KeyVault/vaults/xxxxxxxx).

- Click the Create button.

# \*\*References\*\*

1. Grant Admin consent to App registration :

https://learn.microsoft.com/en-us/azure/active-directory/manage-apps/grant-admin-consent

2. Key Vault Backed Secret Scope:

https://docs.microsoft.com/en-us/azure/databricks/security/secrets/secret-scopes

https://learn.microsoft.com/en-us/azure/databricks/dev-tools/cli/

https://learn.microsoft.com/en-us/azure/databricks/dev-tools/api/latest/aad/service-prin-aad-token