

# Runbook for Maintaining ARM Templates in MLOps

## Table of Contents

Introduction to Azure Resource Manager (ARM) Templates .....	3
Why are ARM Templates Used?.....	3
Benefits.....	3
Key Concepts .....	4
How to Create/Generate ARM Template .....	5
How to Generate ARM Templates for already existing resources .....	5
Create and Deploy your First ARM Template (Azure Storage Account) .....	7
<i>ARM Template Creation</i> .....	8
Resource Parameter Recommendation .....	16
ARM Template for Open AI models .....	18
How to use this template?.....	18
How did we create this template?.....	20
Resource Parameter Recommendation .....	21
ARM Template for ML/DL models .....	23
How to use this template?.....	23
How did we create this template?.....	25
Resource Parameter Recommendation .....	25
Existing ARM Templates (MLOps) Mapping with Portal Parameters .....	28
Azure Databricks .....	28
Azure Storage .....	31
Azure Cognitive Search Services.....	34
Azure Open AI Services .....	37
ARM Template Best Practices .....	40
Validating and Testing ARM Template .....	41
Integrate ARM Template in Azure Pipeline.....	44

How to Deploy ARM Templates .....	47
Deployment of ARM Template Using Azure CLI.....	47
Deploy ARM Template using Power Shell .....	49
Deploy ARM Template using Azure Portal.....	50
ARM Template Deployment Modes .....	52
Resource Creation for Fluke Tech Support GPT.....	53
Storage Account .....	53
Azure SQL Server And Azure SQL DB .....	55
Azure Function App .....	57
Azure Cognitive Search Services.....	65
Web Application .....	72
Log Workspace Creation .....	76
Application Insights .....	80
Enable Web App/Other Resources with Application Insights.....	84
Event System Grid (Trigger for Function App).....	86
Document Intelligence: Form Recognizer .....	90
Appendix A - Structure of ARM Templates .....	93
Definition .....	94
Parameters .....	94
Variables.....	95
Functions.....	96

## Introduction to Azure Resource Manager (ARM) Templates

Azure Resource Manager (ARM) templates are a foundational component of infrastructure as code (IaC) in Microsoft Azure. These templates are written in JSON (JavaScript Object Notation) and are used to define and deploy Azure resources in a declarative and repeatable manner.

## Why are ARM Templates Used?

Azure Resource Manager (ARM) templates are used for managing and deploying resources in Microsoft Azure. They are JSON files that define the infrastructure and configuration for your Azure resources. ARM templates serve several important purposes:

ARM templates are a critical tool for managing Azure resources efficiently, consistently, and securely, providing a foundation for Infrastructure as Code practices in Azure.

1. **Declarative Infrastructure Definition:** ARM templates in Azure allow the declarative definition of infrastructure, emphasizing the desired state of resources and their relationships.
2. **Automated Resource Management:** Automation tools like Azure PowerShell or DevOps can be used with ARM templates for resource provisioning, minimizing human error and ensuring consistency.
3. **Consistency Across Environments:** Infrastructure defined in code promotes consistency, enabling easy replication of configurations in different regions, subscriptions, or for various applications.
4. **Versioning and Change Tracking:** ARM templates can be stored in source control, facilitating versioning, code reviews, and change tracking, ensuring proper documentation of infrastructure modifications.
5. **Scalable and Secure Architecture:** ARM templates support the creation of scalable architectures with various resource types and also allow the enforcement of compliance and security policies, making them a foundation for efficient, secure, and scalable infrastructure management in Azure.

## Benefits

1. **Consistency:** ARM templates promote consistent resource deployments, reducing configuration drift and manual errors.
2. **Scalability:** Templates are highly scalable, making them suitable for provisioning resources across multiple environments, regions, and projects.
3. **Reusability:** Well-designed templates can be reused for similar resource configurations, saving time and effort.
4. **Traceability:** ARM templates provide a clear audit trail of resource changes and deployments, enhancing transparency and compliance.

5. **Automation:** They facilitate automation of infrastructure provisioning, enabling rapid development and deployment of applications.

## Key Concepts

- **Declarative Approach:** ARM templates describe the desired state of Azure resources rather than specifying the sequence of steps to create them. This declarative approach allows for idempotent deployments, where the same template can be used repeatedly without causing conflicts.
- **Resource Definition:** Templates define various Azure resources such as virtual machines, storage accounts, databases, and more, along with their configurations, dependencies, and relationships.
- **Parameterization:** ARM templates support parameters, enabling flexibility and customization when deploying resources. Parameters allow you to input values at deployment time, making templates reusable in various scenarios.
- **Dependency Management:** Templates handle the automatic resolution of resource dependencies, ensuring that resources are provisioned in the correct order to avoid deployment failures.
- **Version Control:** Like application code, ARM templates can be version-controlled using tools like Git, allowing for collaboration, change tracking, and rollbacks.
- **Automated Deployment:** ARM templates can be deployed through various methods, including the Azure Portal, Azure CLI, Azure PowerShell, and continuous integration/continuous deployment (CI/CD) pipelines.

## How to Create/Generate ARM Template

Creating the ARM template manually requires the understanding of each and every component of it mentioned above in the definition. If user knows what ARM template components mean, then manual creation of the ARM template can be opted. As basic use-case of ARM template is to create and maintain infrastructure, following two approaches can be opted for the creation of resource using ARM template.

1. Create a new resource using manually created ARM template
2. For specific resource, use default available templates.

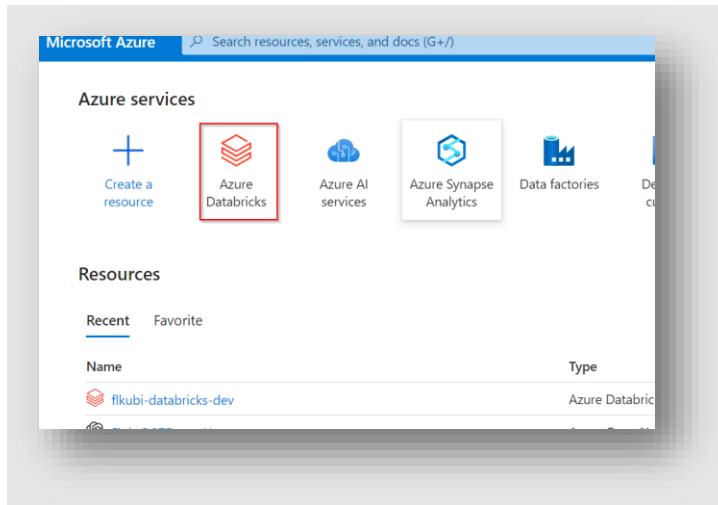
**ARM template creation can be done via Azure Portal or via tool like Visual Studio Code.**

**ARM template generation can be done via Azure Portal, or via Azure CLI or PowerShell with queries.**

## How to Generate ARM Templates for already existing resources

All the resources, created with/without use of ARM template are associated to an ARM code. The code can be generated from the portal for each of the resources. Refer to the following example for getting the code.

1. Go to [portal.azure.com](https://portal.azure.com)
2. Go to the Resource for which you want to generate ARM template.
3. For Databricks resource, go to Azure Databricks.



4. Now click on particular resource workspace/instance.

Azure Databricks

Name	Type	Resource group
flkubi-databricks-dev	Azure Databricks Service	flkubi-dev-rg-001
flkubi-databricks-prd	Azure Databricks Service	flkubi-prd-rg-001
flkubi-databricks-qa	Azure Databricks Service	flkubi-tst-rg-001

5. Click on the Export Template option on the left panel.

flkubi-databricks-dev

Essentials

- Status: Active
- Resource group: flkubi-dev-rg-001
- Location: East US
- Subscription: Fluke Unified BI
- Subscription ID: 52a1d076-bbbf-422a-9bf7-95d61247be4b
- Tags (edit)
- Project: flkubi

6. It will generate the template code for that resource. Portal will provide different options for it to save in library, redeploy from there. It will also allow to see/visualize the ARM template. All the options can be accessed via the top horizontal bar as shows in the below image.

```
$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json",
```

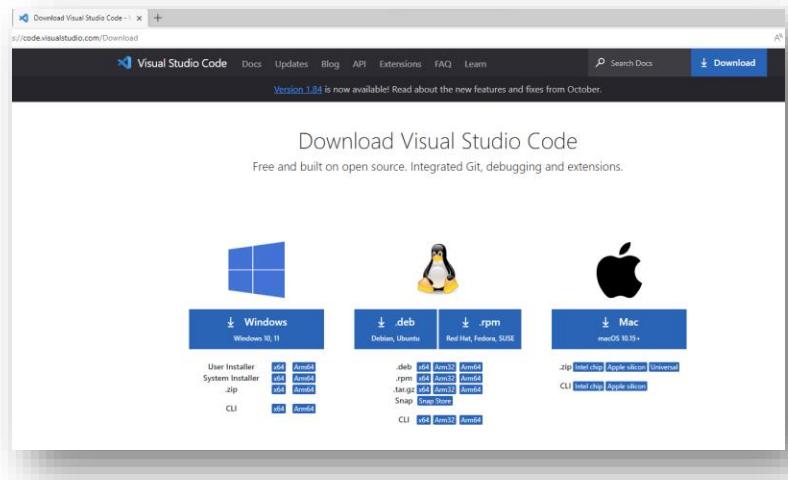
## Create and Deploy your First ARM Template (Azure Storage Account)

In this section, we will be creating the ARM template for Azure Storage Account with the following resources:

- a. Azure Storage Account

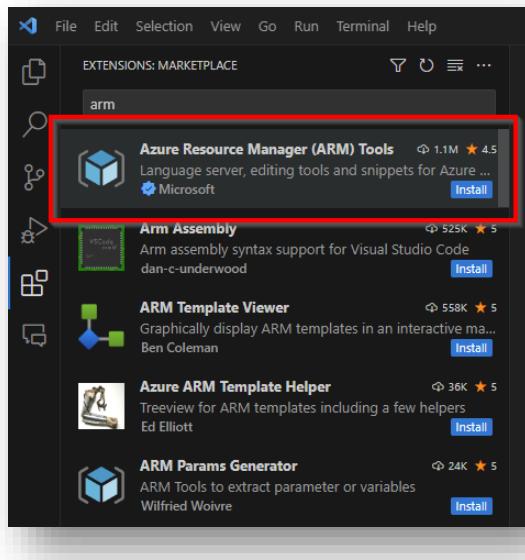
### *Prerequisites steps to create the ARM template in Visual Studio Code*

1. Install Visual Studio Code.



Choose the appropriate version of Visual Studio Code and download and install it.

2. From the Visual Studio Code Extension Marketplace, install the extension: [Azure Resource Manager \(ARM\) Tools](#)

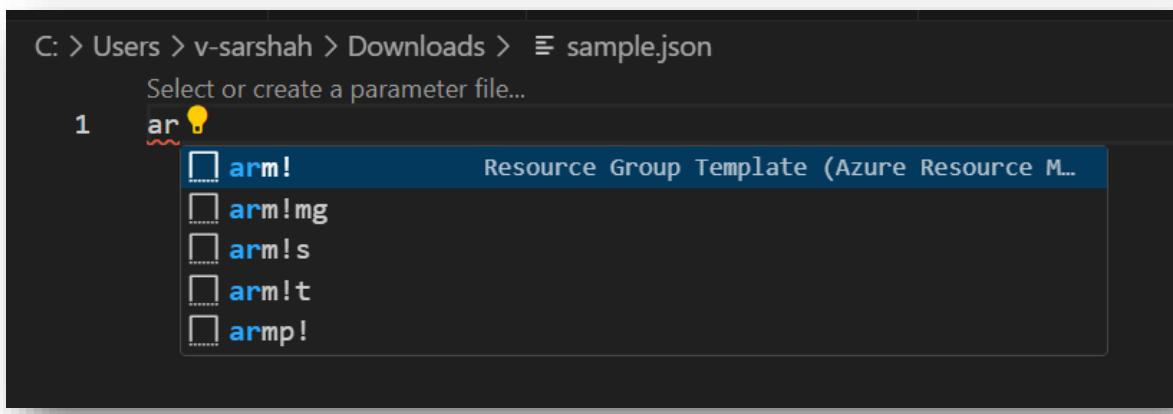


3. Please refer to the section [Deployment of ARM Template Using Azure CLI] and install the Azure CLI in the system and login using the ID-password.

## ARM Template Creation

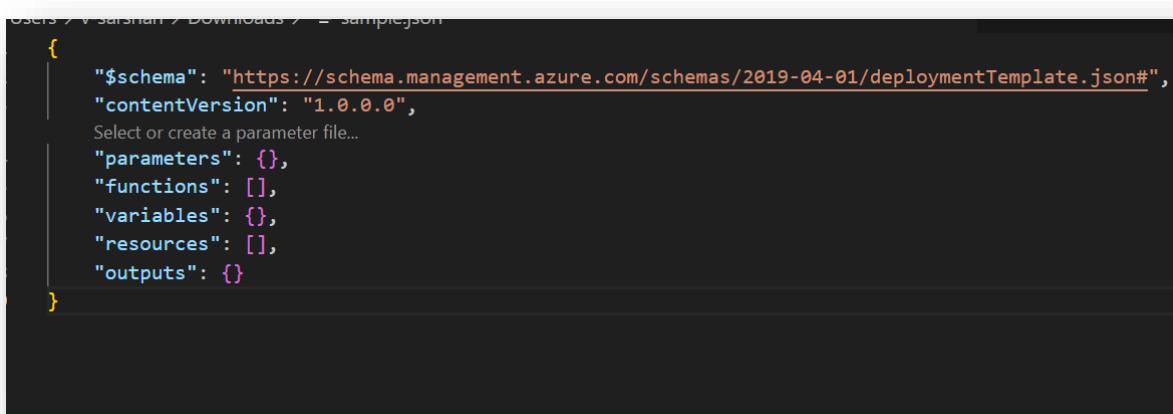
The resource group where the storage account is going to be created is already existing with name 'flk-erpchatbot-dev'.

1. Open the visual studio code, create a new file and save the file with ARMStorage.json name.
2. Now write arm!. It will show you multiple options, where arm! Command will create a sample ARM template schema.



C: > Users > v-sarshah > Downloads > sample.json  
Select or create a parameter file...  
1 arm!

The screenshot shows the Visual Studio Code interface with the command palette open. The user has typed 'arm!' into the search bar. A dropdown menu lists several suggestions, with the top item being 'Resource Group Template (Azure Resource M...)' which is highlighted in blue.



```
{ "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#", "contentVersion": "1.0.0.0", "parameters": {}, "functions": [], "variables": {}, "resources": [], "outputs": {} }
```

The screenshot shows the content of the 'sample.json' file in Visual Studio Code. The file contains a JSON object with properties like '\$schema', 'contentVersion', and arrays for 'parameters', 'functions', 'variables', 'resources', and 'outputs'. The '\$schema' property points to a URL for the Azure Resource Manager schema.

3. Now type arm- in the resources part. It will provide all available sample template of the resources. Select **arm-storage** from the list.

```
5     "functions": [],
6     "variables": {},
7     "resources": [
8         {
9             "name": "storageaccount1",
10            "type": "Microsoft.Storage/storageAccounts",
11            "apiVersion": "2023-01-01",
12            "tags": {
13                "displayName": "storageaccount1"
14            },
15            "location": "[resourceGroup().location]",
16            "kind": "StorageV2",
17            "sku": {
18                "name": "Premium_LRS",
19                "tier": "Premium"
20            }
21        },
22        {
23            "outputs": {}
24        }
25    ]
26}
```

4. Update storage account name to ‘testingstorageaccount’ in the list.
5. Following is the sample JSON that will be created when you add.

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {},
    "functions": [],
    "variables": {},
    "resources": [
        {
            "name": "testingstorageaccount",
            "type": "Microsoft.Storage/storageAccounts",
            "apiVersion": "2023-01-01",
            "tags": {
                "displayName": "testingstorageaccount"
            },
            "location": "[resourceGroup().location]",
            "kind": "StorageV2",
            "sku": {
                "name": "Premium_LRS",
                "tier": "Premium"
            }
        },
        {
            "outputs": {}
        }
    ]
}
```

6. Now lets understand the template.
7. Type refers to type the services that you are creating. Please refer to the section [Appendix A - Structure of ARM Templates]
  - a. Name is name of the storage account name.
  - b. Location refers to the location where storage account will be deployed. It is given reference of the same location as the resource group.
  - c. Kind refers to the Storage account type. Based on features and pricing is associated with the type of storage account. We have selected Storage V2, which is popular to have hierarchical namespace if we want.
  - d. Pricing page for Storage account: [Link](#)
  - e. Point 5 refers to Tier of the storage account. Standard and Premium are two options. The details of those can be checked in the pricing page.
  - f. There are other main components like Variables, Parameters and Functions.
    - i. **Parameters:** Parameters contain values for the resource. It can be taken as User Input, can be overridden easily. While deploying resources from Dev to QA or to other environment, resource connection strings and other dynamic things are handled and overridden in Parameters.
    - ii. **Variables:** Variables are used internally in the ARM template where single value is referenced or used at multiple places then variables are useful.
    - iii. **Functions:** Functions handle dynamicity in the ARM template. Different types of function like logical (i.e., if, and, true), string (i.e., concat, indexOf) numerical (i.e., int, float) etc., are included in the ARM template library

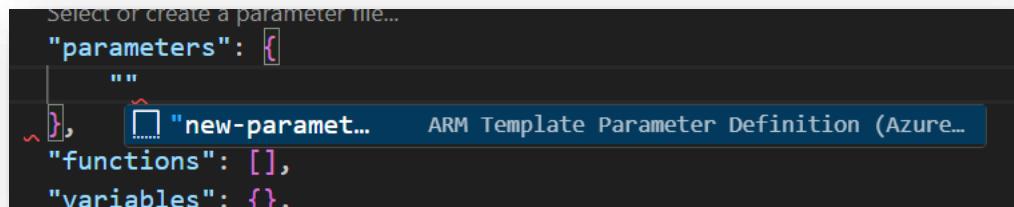
```

1  {
2    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
3    "contentVersion": "1.0.0.0",
4    "parameters": {},
5    "functions": [],
6    "variables": {},
7    "resources": [
8      {
9        "name": "testingstorageaccount",
10       1 "type": "Microsoft.Storage/storageAccounts",
11       "apiVersion": "2023-01-01",
12       "tags": {
13         "displayName": "testingstorageaccount"
14       },
15       2 "location": "[resourceGroup().location]",
16       "kind": "StorageV2",
17       "sku": {
18         4 "name": "Premium_LRS",
19         "tier": "Premium"
20       }
21     ]
22   ]

```

## *Parameterize the ARM Template*

1. Lets add a parameter which allows us to set the name of Storage Account dynamically. Following is a sample parameter addition code. Inside the Parameters main tag, just by typing something, it will recommend you this sample code. Add this code in the Parameter array.



```
parameters": [",
  "",
  ],
  "functions": [],
  "variables": {}.
```

- a. It will show the code like following

```
"parameter1": {
    "type": "string",
    "metadata": {
        "description": "description"
    }
}
```

2. Update it like following. Give name of the Parameter as EnvironmentName, and provide the proper description to it.

```
"EnvironmentName": {
    "type": "string",
    "metadata": {
        "description": "The parameter depicts the environment name i.e., Dev, QA or Prod."
    }
}
```

- a. Repeat the steps and add following parameters.

**Note: Add comma ‘,’ after every parameter/variables/ or functions are defined.**

- i. StorageAccountName: Represents the storage account name
- ii. StorageAccountKind: Represents the storage account kind (i.e., Storage, StorageV2)
- iii. StorageAccountSKUName: Represents storage account SKU name (i.e., Standard\_LRS, Premium\_LRS etc.)
- iv. StorageAccountSKUTier: Represents storage account tier (i.e., Standard, Premium)

v. Refer the following code for the reference.

```
"parameters": {  
    "StorageAccountName": {  
        "type": "string",  
        "metadata": {  
            "description": "The parameter depicts the name of the storage account"  
        },  
        "EnvironmentName": {  
            "type": "string",  
            "metadata": {  
                "description": "The parameter depicts the environment name i.e., Dev, QA or Prod."  
            },  
            "StorageAccountKind": {  
                "type": "string",  
                "metadata": {  
                    "description": "The parameter depicts the environment name i.e., StorageV2, Storage"  
                },  
                "StorageAccountSKUName": {  
                    "type": "string",  
                    "metadata": {  
                        "description": "The parameter depicts the environment name i.e., Premium_LRS, Standard_LRS etc."  
                    },  
                    "StorageAccountSKUTier": {  
                        "type": "string",  
                        "metadata": {  
                            "description": "The parameter depicts the environment name i.e., Premium, Standard etc."  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

3. Now lets add variable which stores the name of the storage account. Same way that we added a parameter, by typing anything in variables section will give an option to create a new variable schema.

The screenshot shows a code editor with an ARM template. A tooltip is displayed over the word "variables" in the JSON object, reading "ARM Template Variable Definition (Azure Resource Manager)". The template code includes sections for functions, variables, and resources.

```

    "functions": [],
    "variables": [
        ""
    ],
    "resources": [
    ]
  
```

4. While clicking on it, it will add sample code for variable creation as below.
  - a. "variable1": "value"
5. Now update the variable like following with the base name of the storage account.

```
"StorageAccount_APIVersion": "2023-01-01"
```

#### *Use of Functions + Parameters + Variables*

1. Now, Lets add the reference of the parameter in the ARM template where resource is getting created. Any referenced item should be called in side a rectangle brackets like [].
  - a. We will create a resource with name **temp1t1es1t** \*and different environment. Lets say, **devtemp1t1es1t** for dev, **qatemp1t1es1t** for QA and **prdtemp1t1es1t** for Prod. We will use **concat** function to create such name,
 

**\* : Name of the storage is chosen based on the availability of the names. It should be unique across the all Azure Storages. If normal names are not available, it will throw error in the deployment. Numeric values can be added to the name to make it unique.**
  - b. Update the name of the resource with following:
  - c. "name": "temp1t1es1t" ==> "**name": "[concat(parameters('StorageAccountName), parameters('EnvironmentType'))]"**", Now whenever we deploy this, it will ask you the parameters name as user input.
  - d. Update display name as "displayname": " temp1t1es1t " → "displayname": "**[toLowerCase(concat(parameters('EnvironmentName'),parameters('StorageAccountName'))))]"**,
  - e. Similarly update all the variables accordingly as given below.
    - i. Update
    - ii. Update API version as "apiVersion": "2023-01-01" → "apiVersion": "**[variables('StorageAccount\_APIVersion')]"**
    - iii. Update "kind": "StorageV2" → "kind": "**[parameters('StorageAccountKind')]"**
    - iv. Update SKU name from "name": "Premium\_LRS" → "name": "**[parameters('StorageAccountSKUName')]"**
    - v. Update SKU tier as "tier": "Premium" → "tier": "**[parameters('StorageAccountSKUTier')]"**

- f. Parameters are referenced with parameters('<parametername>') and variables are referenced with variables('<variablename>').
- g. Please refer to the following attached file for the full reference.



i.

### *Deploy Sample ARM Template*

1. To deploy the simple ARM template StorageARM.json file, we will use Azure CLI.
2. Refer to the section for installation of Azure CLI and Log in.
3. Now, run the following command.
  - a. az deployment group create --name <nameofdeployment> --resource-group <nameoftheresourcegroup> --template-file <templatefilepath>
  - b. Replace the name nameofdeployment with any name that you want.
  - c. Nameoftheresourcegroup should be replaced with resource group name where the resource is meant to be created.
  - d. Templatefilepath refers to file path of the local system for StorageARM.json.
4. Now add the resource name and click enter.

```
C:\Users\████████>az deployment group create --name StorageAccountDeployment1 --resource-group flk-erpchatbot-dev --template-file C:/Users/████████/Downloads/StorageARM.json
[]\ Running ..
```

5. It will ask to input the parameter name. Add **dev** and click enter.\*

**\*: Enter ‘?’ to see the sample values of the parameters**

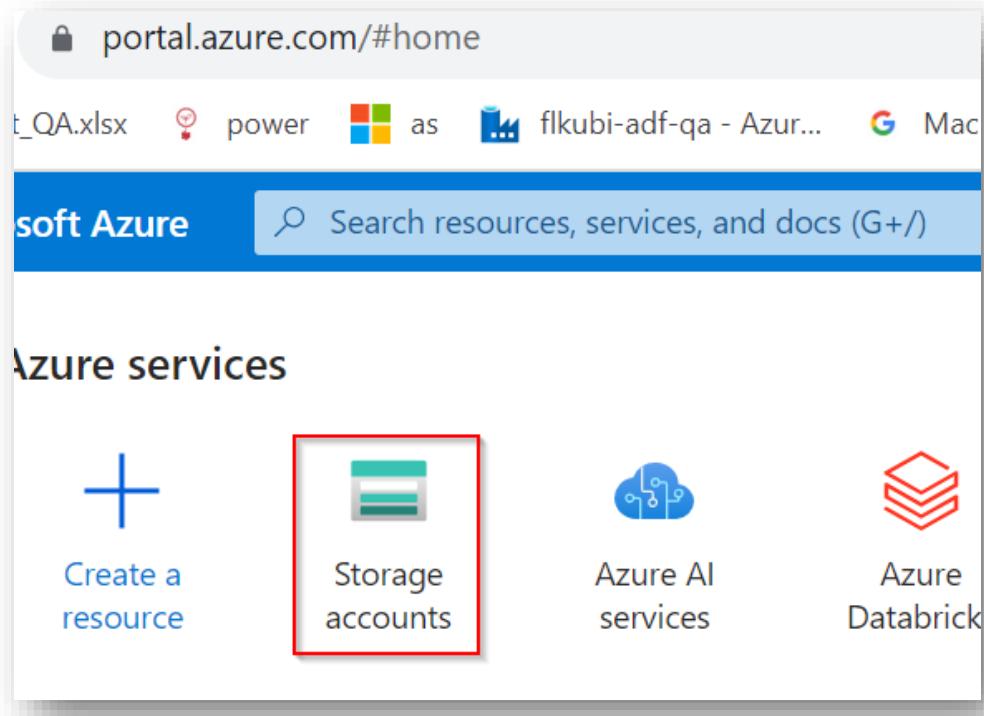
```
]>
C:\Users\████████>az deployment group create --name ARMTemplateDeploymentTest1 --resource-group flk-erpchatbot-dev --template-file C:/Users/████████/Downloads/StorageARM.json
Please provide string value for 'StorageAccountName' (? for help): temp1t1v1
Please provide string value for 'EnvironmentName' (? for help): dev
Please provide string value for 'StorageAccountKind' (? for help): StorageV2
Please provide string value for 'StorageAccountSKUName' (? for help): ?
The parameter depicts the environment name i.e., Premium_LRS,Standard_LRS etc.
Please provide string value for 'StorageAccountSKUName' (? for help): Standard_LRS
Please provide string value for 'StorageAccountSKUTier' (? for help): Standard
{
  "id": "/subscriptions/52a1d076-bbbf-422a-9bf7-95d61247be4b/resourceGroups/flk-erpchatbot-dev/providers/Microsoft.Resources/deployments/ARMTempla
teDeploymentTest1",
  "name": "ARMTempla
```

6. Once it is deployed, It will show all details of the created resource.

```
Please provide string value for 'StorageAccountSKUTier' (? for help): Standard
{
  "id": "/subscriptions/52a1d076-bbbf-422a-9bf7-95d61247be4b/resourceGroups/flk-erpchatbot-dev/providers/Microsoft
teDeploymentTest1",
  "location": null,
  "name": "ARMTemplateDeploymentTest1",
  "properties": {
    "correlationId": "03c83fd7-9818-472b-841d-4173f0ae39d1",
    "debugSetting": null,
    "dependencies": [],
    "duration": "PT29.9251946S",
    "error": null,
    "mode": "Incremental",
    "onErrorDeployment": null,
    "outputResources": [
      {
        "id": "/subscriptions/52a1d076-bbbf-422a-9bf7-95d61247be4b/resourceGroups/flk-erpchatbot-dev/providers/Mic
emp1t1v1dev",
        "resourceGroup": "flk-erpchatbot-dev"
      }
    ],
    "outputs": {},
    "parameters": {
      "environmentName": {
        "value": "flk-erpchatbot-dev"
      }
    }
  }
}
```

7. Verify resource creation from portal.

- a. Open Portal.Azure.Com -> Azure AI Services -> Azure Cognitive Search



b.

The screenshot shows the Azure Storage accounts blade. At the top, there are navigation links: Home > Storage accounts. Below the header are several buttons: Create, Restore, Manage view, Refresh, Export to CSV, Open query, Assign tags, and Delete. There are also filter options: Filter for any field..., Subscription equals all, Resource group equals all, Location equals all, and Add filter. The main area displays a table of storage accounts. The columns are: Name, Type, Kind, and Resource group. The table contains 10 records, showing names like 'flkubiadlspru', 'flkubiadlsprdwest', 'flkubiadlsqa', 'flkubiprdrg0018f49', 'flkubiprdrg001ae58', and 'temp1t1v1dev'. The row for 'temp1t1v1dev' has a red box around it, indicating it is selected.

Name	Type	Kind	Resource group
flkubiadlspru	Storage account	StorageV2	flkubi-prd-rg-001
flkubiadlsprdwest	Storage account	StorageV2	flkubi-tst-rg-001
flkubiadlsqa	Storage account	StorageV2	flkubi-prd-rg-001
flkubiprdrg0018f49	Storage account	Storage	flkubi-prd-rg-001
flkubiprdrg001ae58	Storage account	Storage	flkubi-prd-rg-001
<b>temp1t1v1dev</b>	Storage account	StorageV2	flk-erpchatbot-dev

## Resource Parameter Recommendation

Resource	Parameter	Suggestions	Recommendation	Reason
Storage Account	StorageAccountName	NA	The name must be unique across all existing storage account names in Azure. It must be 3 to 24 characters long, and can contain only lowercase letters and numbers.	Try to use numbers in between alphabets to make it unique
Storage Account	EnvironmentName	1. dev 2. qa 3. prod/prd	NA	NA
Storage Account	StorageAccountKind	1. StorageV2 2. Storage	StorageV2	It combines the older blob functionalities with hierarchical name space functions of newly launched Azure Storage.

Storage Account	StorageAccountSKUName	1. Standard_LRS 2. Premium_LRS 3. Standard_GRS 4. Premium_GRS	Standard_LRS -> if data privacy is very high and data can not be stored on global level data centers. Standard_GRS -> to store replica of data on global level	<a href="#">Please refer to the link</a>
Storage Account	StorageAccountSKUTier	1. Standard 2. Premium	Standard	<a href="#">Please refer to the link</a>

## ARM Template for Open AI models

Any Open AI based project would require the following resources:

1. Azure Open AI resource
2. Azure Open AI Model Deployment
3. Azure Storage Account
4. Azure Cognitive Search Services

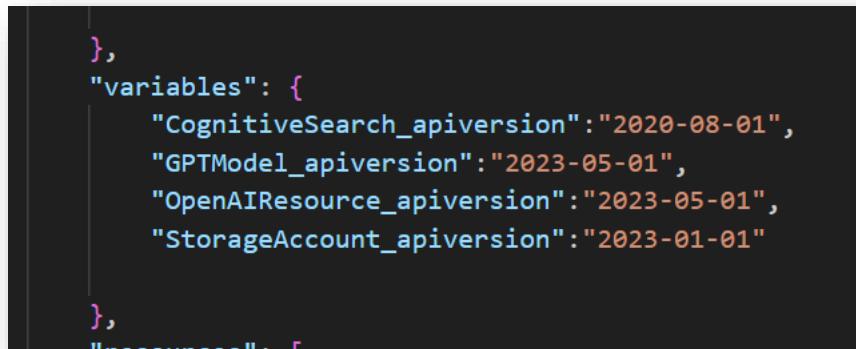
Following file contains the ARM template in dynamic manner.



Pre-requisites: Please refer to the section [Prerequisites steps to create the ARM template in Visual Studio Code] for installing the pre-requisites.

### How to use this template?

1. Save the file in the system.
  - a. Double click on the file and save it in the system with save name.
2. Update following numbered details of the image.
  - a. Update the api version details under variables section.



```
    },
    "variables": {
        "CognitiveSearch_apiversion": "2020-08-01",
        "GPTModel_apiversion": "2023-05-01",
        "OpenAIResource_apiversion": "2023-05-01",
        "StorageAccount_apiversion": "2023-01-01"
    },
    "resources": [
        {
            "type": "Microsoft.OpenAI/Models",
            "name": "[parameters('DeploymentName')]",
            "apiVersion": "2023-05-01",
            "location": "[resourceGroup().location]",
            "dependsOn": [
                "[resourceId('Microsoft.Storage/storageAccounts', parameters('StorageAccountName'))]"
            ],
            "properties": {
                "modelType": "GPTModel",
                "modelName": "[parameters('ModelName')]",
                "storageConnectionString": "[listKeys(resourceId('Microsoft.Storage/storageAccounts', parameters('StorageAccountName')), '2023-01-01').connectionStrings[0].connectionString]"
            }
        }
    ]
}
```

A screenshot of a code editor showing an ARM template. The code defines a 'variables' block with four entries: CognitiveSearch\_apiversion, GPTModel\_apiversion, OpenAIResource\_apiversion, and StorageAccount\_apiversion, all set to their respective API versions. Below this is a 'resources' block containing a single object. This object is a 'Microsoft.OpenAI/Models' resource with a name derived from the 'DeploymentName' parameter. It specifies the 'modelType' as 'GPTModel', the 'modelName' as defined in the parameters, and the 'storageConnectionString' as a list key of the storage account's connection strings. The 'apiVersion' is set to '2023-05-01', and it depends on the 'StorageAccount' resource defined earlier in the template.

3. Now run the file using following command.
  - a. az deployment group create --name <DeploymentName> --resource-group <ResourceGroupName> --template-file <fullpathoffile>
  - b. DeploymentName: Provide any deployment name to deploy.
  - c. ResourceGroupName: Enter resource group name where all the resources will be created
  - d. Provide full path of OpenAI\_demo.json.
4. Command will ask for following parameter names  
**Note: Use '?' to get sample values for the parameters.**

- a. OpenAI\_Resource\_Name: Provide name of the open ai resource
- b. OpenAI\_Resource\_Area: Provide the area associated with the resource. i.e., East US, East US2, West US etc)
- c. OpenAI\_Model\_Name: Provide LLM model which you want to create (i.e., gpt-35-turbo, gpt-35-turbo-16k)
- d. OpenAI\_Model\_Version: Provide version of the Open AI models (i.e., 0301,0613)
- e. Cognitive\_Search\_Resource\_Name: Provide cognitive search resource name.
- f. Cognitive\_Search\_Resource\_Location: Provide location for the Cognitive Search resource. (i.e., East US, West US)
- g. StorageAccountName: Provide storage account name.
- h. StorageAccountVersion: Provide kind of storage account. (i.e., StorageV2)
- i. StorageAccountSKUName: Provide SKU Name for the storage account (i.e., Standard\_LRS, Premium\_LRS etc.)
- j. StorageAccountSKUTier: Provide SKU tier for the storage account. (i.e., Standard, Premium)
- k. Please refer to the following image for the sample values entered.

```
C:\Users\████████>az deployment group create --name ARMTemplateDeploymentTest1 --resource-group flk-erpchatbot-dev --template-file C:/Users/████████/Downloads/OpenAIDemo.json
Please provide string value for 'OpenAI_Resource_Name' (? for help): armdevdeployment
Please provide string value for 'OpenAI_Resource_Area' (? for help): East US
Please provide string value for 'OpenAI_Model_Name' (? for help): gpt-35-turbo
Please provide string value for 'OpenAI_Model_Version' (? for help): 0301
Please provide string value for 'Cognitive_Search_Resource_Name' (? for help): armdevdeployment
Please provide string value for 'Cognitive_Search_Resource_Location' (? for help): East US
Please provide string value for 'StorageAccountName' (? for help): armdevdeployment
Please provide string value for 'StorageAccountVersion' (? for help): StorageV2
Please provide string value for 'StorageAccountSKUName' (? for help): Standard_LRS
Please provide string value for 'StorageAccountSKUTier' (? for help): Standard
```

- l. Once entered, it will start the deployment. Once deployment is completed, it will show screen like shown below with resourceIDs of the resources.

```
{
  "debugSetting": null,
  "dependencies": [],
  "duration": "PT27.2394072S",
  "error": null,
  "mode": "Incremental",
  "onErrorDeployment": null,
  "outputResources": [
    {
      "id": "/subscriptions/52a1d076-bbbf-422a-9bf7-95d61247be4b/resourceGroups/flk-erpchatbot-dev/providers/Microsoft.CognitiveServices/armdevdeployment1",
      "resourceGroup": "flk-erpchatbot-dev"
    },
    {
      "id": "/subscriptions/52a1d076-bbbf-422a-9bf7-95d61247be4b/resourceGroups/flk-erpchatbot-dev/providers/Microsoft.Search/searchServiceDeployment1",
      "resourceGroup": "flk-erpchatbot-dev"
    },
    {
      "id": "/subscriptions/52a1d076-bbbf-422a-9bf7-95d61247be4b/resourceGroups/flk-erpchatbot-dev/providers/Microsoft.Storage/storageAccounts/armdevdeployment1",
      "resourceGroup": "flk-erpchatbot-dev"
    }
  ]
}
```

- m. All created resources can be viewed under your selected resource group in portal.azure.com.

The screenshot shows the Azure portal interface for a resource group named 'flk-erpchatbot-dev'. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Resource visualizer, Events, Deployments, Security, Deployment stacks, Policies, and more. The main content area is titled 'Essentials' under 'Resources' and displays a list of 14 resources. The resources are listed in two columns: Name and Type. The resources include:

Name	Type
armdeploymentdev	Azure OpenAI
armdevdeployment	Search service
armdevdeployment1	Azure OpenAI
armdevdeployment1	Search service
armdevdeployment1	Storage account

## How did we create this template?

Follow these steps:

1. The template was created in the visual studio using ARM template extension.
  - Please refer to the [Prerequisites steps to create the ARM template in Visual Studio Code] to install it.
2. Now, we used command like arm-storage for storage account template. And update the values based on [documentation](#).
3. We added Azure Open AI template using template generation method and updated the values using [Microsoft documentation](#).
4. We also added Cognitive Search resources deployment sample template from the given [documentation](#) and updated the values in it.
5. We then parameterized the names of the resources, and created variables for api version. Please refer to the section [Parameterize the ARM Template] for more details on how to create parameters/variables.

## Resource Parameter Recommendation

Resource	Parameter	Suggestions	Recommendation	Reason
Open AI Resource	OpenAI_Resource_Name	NA	Your resource name can only include alphanumeric characters and hyphens and can't start or end with a hyphen.	NA
Open AI Resource	OpenAI_Resource_Area	1. East US 2. West US 3. East US2	Keep area same as area of resource group under which the resource is to be created.	NA
Open AI Resource	OpenAI_Model_Name	1. gpt-35-turbo 2. gpt-35-turbo-16k 3. gpt-35-turbo-instruct	gpt-35-turbo.	Most of the Questions and Answers will be under 4000 words/tokens. If requirement is higher then choose 16k model.
Open AI Resource	OpenAI_Model_Version	1. 0301, 0613 (for gpt-35-turbo) 2. 0613 (for gpt-35-turbo-16k) 2. 0914 (for gpt-35-instruct)	Select the latest one, higher one based on requirement.	NA
Azure Cognitive Search	Cognitive_Search_Resource_Name	NA	Service name must only contain lowercase letters, digits or dashes, cannot use dash as the first two or last one characters, cannot contain consecutive dashes, and is limited between 2 and 60 characters in length.	NA

Azure Cognitive Search	Cognitive_Search_Resource_Location	1. East US 2. West US 3. East US2	Keep area same as area of resource group under which the resource is to be created.	NA
Storage Account	StorageAccountName	NA	The name must be unique across all existing storage account names in Azure. It must be 3 to 24 characters long, and can contain only lowercase letters and numbers.	Try to use numbers in between alphabets to make it unique
Storage Account	StorageAccountVersion	1. StorageV2 2. Storage	StorageV2	It combines the older blob functionalities with hierarchical name space functions of newly launched Azure Storage.
Storage Account	StorageAccountSKUName	1. Standard_LRS 2. Premium_LRS 3. Standard_GRS 4. Premium_GRS	Standard_LRS -> if data privacy is very high and data can not be stored on global level data centers. Standard_GRS -> to store replica of data on global level	<a href="#">Please refer to the link</a>
Storage Account	StorageAccountSKUTier	1. Standard 2. Premium	Standard	<a href="#">Please refer to the link</a>

## ARM Template for ML/DL models

Any ML/DL based project would require the following resources:

1. Azure Databricks/Azure ML
2. Azure Storage
3. Azure Synapse (if required)

Following file contains the ARM template in dynamic manner.



### How to use this template?

1. Save the file in the system.
  - a. Double click on the file and save it in the system with save name.
2. Update following numbered details of the image.
  - a. Update the api version details under variables section.

```
"variables": {  
    "managedResourceGroupName": "[format('databricks-rg-{0}-{1}', parameters().resourceGroupSuffix, parameters().environmentName))]",  
    "StorageAccount_apiversion": "2023-01-01",  
    "SQLServer_apiversion": "2021-11-01-preview",  
    "Databricks_apiversion": "2018-04-01",  
    "Databases_apiversion": "2021-11-01-preview"  
},
```

3. Now run the file using following command.
  - a. az deployment group create --name <DeploymentName> --resource-group <ResourceGroupName> --template-file <fullpathoffile>
  - b. DeploymentName: Provide any deployment name to deploy.
  - c. ResourceGroupName: Enter resource group name where all the resources will be created
  - d. Provide full path of OpenAIDemo.json.
4. Command will ask for following parameter names
  - a. workspaceName: Provide Databricks workspace name
  - b. StorageAccountName: Provide storage account name.
  - c. StorageAccountVersion: Provide kind of storage account. (i.e., Storage, StorageV2)

- d. sqlAdministratorLogin: Provide user id for SQL database. Based on provided user id it will create new user id using that name.
- e. sqlAdministratorPassword: Provide password for the sql database login. It will set the password for the above given user id.
- f. databasesName: Provide SQL database name.
- g. transperantDataEncryption: It allows to enable the encryption. Enter either 1 or 2.
- h. capacity: Capacity refers to the DTU. Minimum allowed value for it is 900 and maximum allowed is 54000.
- i. databaseCollation: Enter values for database collation. i.e.,  
SQL\_Latin1\_General\_CI\_AS
- j. Please refer to the following image for the sample values entered.

```
C:\Users\v-sarshah\az deployment group create --name ARMTemplateDeploymentTest1 --resource-group flk-erpchatbot-dev --template-file C:/Users/v-sarshah/Downloads/MLResourceDeployment.json
Please provide string value for 'workspaceName' (? for help): temp1m1
Please provide string value for 'location' (? for help): East US
Please provide string value for 'StorageAccountName' (? for help): temp1m1
Please provide string value for 'StorageAccountVersion' (? for help): StorageV2
Please provide string value for 'sqlServerName' (? for help): tempe
Please provide string value for 'sqlAdministratorLogin' (? for help): addddim
Please provide securestring value for 'sqlAdministratorPassword' (? for help):
Please provide string value for 'databasesName' (? for help): fffe
Please provide string value for 'transperantDataEncryption' (? for help):
[1] Enabled
[2] Disabled
Please enter a choice [Default choice(1)]: Enabled
Valid values are [1, 2]
Please provide string value for 'transparentDataEncryption' (? for help):
[1] Enabled
[2] Disabled
Please enter a choice [Default choice(1)]: 1
Please provide int value for 'capacity' (? for help): 900
Please provide string value for 'databaseCollation' (? for help): SQL_Latin1_General_CI_AS
```

- k. Once entered, it will start the deployment. Once deployment is completed, it will show screen like shown below with resourceIDs of the resources.

```
{
  "debugSetting": null,
  "dependencies": [],
  "duration": "PT27.2394072S",
  "error": null,
  "mode": "Incremental",
  "onErrorDeployment": null,
  "outputResources": [
    {
      "id": "/subscriptions/52a1d076-bbbf-422a-9bf7-95d61247be4b/resourceGroups/flk-erpchatbot-dev/providers/Microsoft.CognitiveServices/armdevdeployment1",
      "resourceGroup": "flk-erpchatbot-dev"
    },
    {
      "id": "/subscriptions/52a1d076-bbbf-422a-9bf7-95d61247be4b/resourceGroups/flk-erpchatbot-dev/providers/Microsoft.Search/searchServiceDeployment1",
      "resourceGroup": "flk-erpchatbot-dev"
    },
    {
      "id": "/subscriptions/52a1d076-bbbf-422a-9bf7-95d61247be4b/resourceGroups/flk-erpchatbot-dev/providers/Microsoft.Storage/storageAccountsdevdeployment1",
      "resourceGroup": "flk-erpchatbot-dev"
    }
  ]
}
```

- l. All created resources can be viewed under your selected resource group in portal.azure.com.

## How did we create this template?

Follow these steps:

1. The template was created in the visual studio using ARM template extension.
  - Please refer to the [Prerequisites steps to create the ARM template in Visual Studio Code] to install it.
2. Now, we used command like arm-storage for storage account template. And update the values based on [documentation](#).
3. We added Azure Databricks template from the given [documentation](#) and updated the values in it.
4. We also added Azure SQL database deployment template from the [Microsoft Documentation](#) and updated the values in it..
5. We then parameterized the names of the resources, and created variables for api version. Please refer to the section [Parameterize the ARM Template] for more details on how to create parameters/variables.

## Resource Parameter Recommendation

Resource	Parameter	Suggestions	Recommendation	Reason
Azure Databricks	Workspace Name	NA	Keep <workspacename>-<EnvironmentName> convention.	NA
Azure Databricks	Pricing Tier	1. Standard 2. Premium 3. Trial	Standard	If Standard is cost optimized tier. If Role based access is required then go for Premium.
Azure Databricks	Location	1. East US 2. East US2 3. West US 4. West US2	Keep the location same as resource group's location	NA

Storage Account	StorageAccountName	NA	The name must be unique across all existing storage account names in Azure. It must be 3 to 24 characters long, and can contain only lowercase letters and numbers.	Try to use numbers in between alphabets to make it unique
Storage Account	StorageAccountVersion	1. StorageV2 2. Storage	StorageV2	It combines the older blob functionalities with hierarchical name space functions of newly launched Azure Storage.
Storage Account	StorageAccountSKUName	1. Standard_LRS 2. Premium_LRS 3. Standard_GRS 4. Premium_GRS	Standard_LRS -> if data privacy is very high and data can not be stored on global level data centers. Standard_GRS -> to store replica of data on global level	<a href="#">Please refer to the link</a>
Storage Account	StorageAccountSKUTier	1. Standard 2. Premium	Standard	<a href="#">Please refer to the link</a>
SQL Server	sqlServerName	NA	Keep <workspacename>-<EnvironmentName> convention.	NA
SQL Server	sqlAdministratorLogin	NA	Enter proper user id for SQL server authentication. It should not be admin.	NA
SQL Server	sqlAdministratorPassword	NA	Enter password containing Uppercase, Lowercase, Symbols, Numbers having	

			length of minimum 8 and maximum 64	
SQL Server	databasesName	NA	Keep <databaseName>-<EnvironmentName> convention. With maximum length of 63. It should contain only alphanumerics and hyphens.	
SQL Server	transparentDataEncryption	1. Enabled 2. Disabled	1. Enabled	Recommended for data safty.
SQL Server	capacity	From 900 to 54000	Select based on requirement.	<a href="#">Please refer to the link</a>
SQL Server	databaseCollation	SQL_Latin1_General_CI_AS	SQL_Latin1_General_CI_AS	NA

## Existing ARM Templates (MLOps) Mapping with Portal Parameters

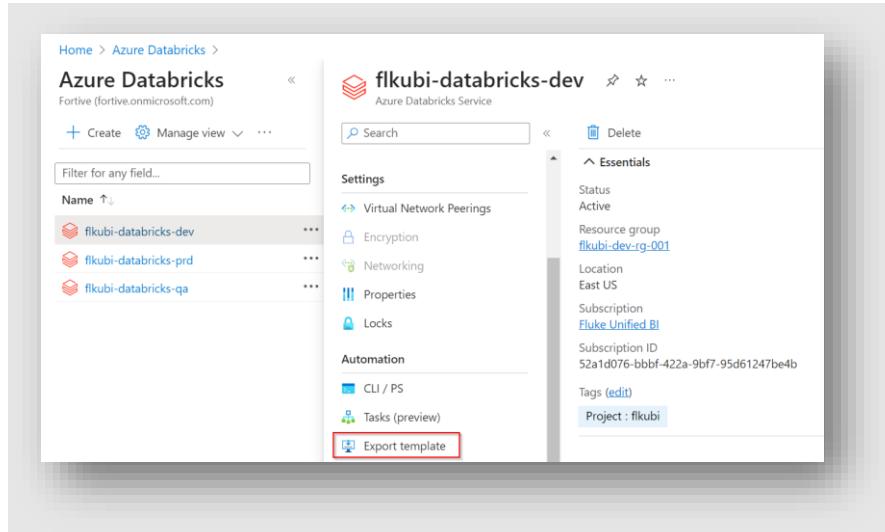
For MLOps, Statistical/Deep Learning/Azure AI services based model majorly uses following resources.

1. Azure Databricks
2. Azure Storage
3. Azure Cognitive Search Services
4. Azure Open AI Services.
5. Azure Web App Services
6. Azure Data Factory Services (if data-pull is integrated)

### Azure Databricks

Azure Databricks repository can not be maintained using ARM templates. As the repository contains notebooks and other codes, which can not be maintained using ARM template directly. ARM template can provide option to deploy a ADB workspace but information regarding notebooks are not captured in it.

Azure Databricks ARM template can be exported like below



Sample ARM template for Databricks is given below

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploy-
mentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "workspaces_flkubi_databricks_dev_name": {
      "defaultValue": "flkubi-databricks-dev",
      "type": "String"
    }
  },
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Databricks/workspaces",
      "apiVersion": "2023-02-01",
      "name": "[parameters('workspaces_flkubi_databricks_dev_name')]",
      "location": "eastus",
      "tags": {
        "Project": "flkubi"
      },
      "sku": {
        "name": "standard"
      },
      "properties": {
        "managedResourceGroupId": "[concat('/subscriptions/52a1d076-bbbf-
422a-9bf7-95d61247be4b/resourceGroups/databricks-rg', parameters('work-
spaces_flkubi_databricks_dev_name'), '-t5pxdgqlejimc')]",
        "parameters": {
          "enableNoPublicIp": {
            "type": "Bool",
            "value": true
          },
          "natGatewayName": {
            "type": "String",
            "value": "nat-gateway"
          },
          "prepareEncryption": {
            "type": "Bool",
            "value": false
          },
          "publicIPName": {
            "type": "String",
            "value": "nat-gw-public-ip"
          },
          "requireInfrastructureEncryption": {
            "type": "Bool",
            "value": false
          }
        }
      }
    }
  ]
}
```

## ARM Template Explanation

```
parameters": {
  "workspaces_flkubi_databricks_dev_name": {
    "defaultValue": "flkubi-databricks-dev",
    "type": "String"
  }
},
"variables": {},
"resources": [
  {
    "type": "Microsoft.Databricks/workspaces",
    "apiVersion": "2023-02-01",
    "name": "[parameters('workspaces_flkubi_databricks_dev_name')]",
    "location": "eastus",
    "tags": {
      "Project": "flkubi"
    },
    "sku": {
      "name": "standard"
    }
  }
]
```

The screenshot shows the Azure portal interface for managing an Azure Databricks service. The service name is 'flkubi-databricks-dev'. The 'Essentials' section includes:

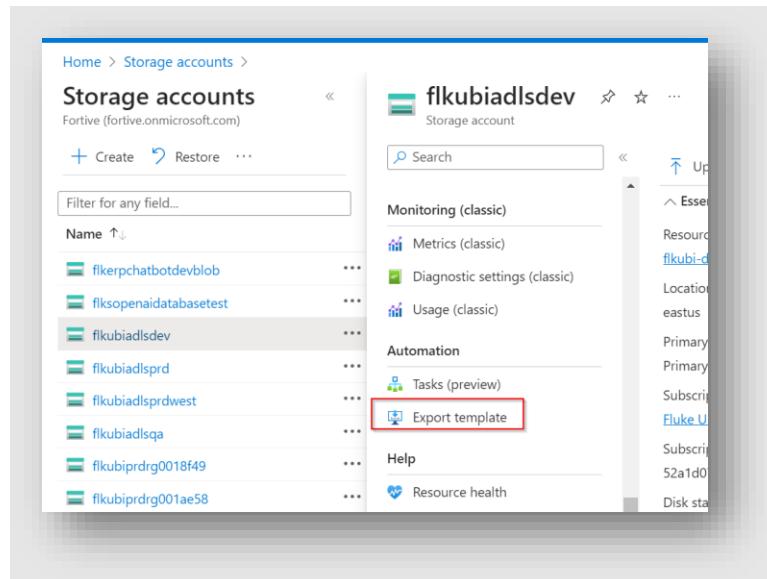
- Status: Active
- Resource group: flkubi-dev-rg-001
- Location: East US
- Subscription: Fluke Unified BI
- Subscription ID: 52a1d076-bbbf-422a-9bf7-95d61247be4b
- Tags (edit): Project: flkubi

On the right side, there is a JSON panel showing the configuration details of the service.

```
Managed Resource Group  
databricks-rg-flkubi-databricks-dev-t5pxdgrl  
URL  
https://adb-194377387358740.azuredatabr  
Pricing Tier  
Standard (Apache Spark, Secure with Microsof  
Enable No Public IP  
Yes
```

## Azure Storage

Similar to Azure Databricks, Azure Storage as a resource can be deployed using ARM template. But the files that it contains, it can not be deployed or tracked via ARM templates.



Sample storage ARM template is attached below.

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2019-04-  
01/deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "storageAccounts_flkubiadlsdev_name": {  
            "defaultValue": "flkubiadlsdev",  
            "type": "String"  
        },  
        "factories_flkubi_adf_dev_externalid": {  
            "defaultValue": "/subscriptions/52a1d076-bbbf-422a-9bf7-  
95d61247be4b/resourceGroups/flkubi-dev-rg-  
001/providers/Microsoft.DataFactory/factories/flkubi-adf-dev",  
            "type": "String"  
        },  
        "virtualNetworks_flkubi_mgd_vnet_001_externalid": {  
            "defaultValue": "/subscriptions/52a1d076-bbbf-422a-9bf7-  
95d61247be4b/resourceGroups/flkubi-mgd-rg-  
001/providers/Microsoft.Network/virtualNetworks/flkubi-mgd-vnet-001",  
            "type": "String"  
        },  
        "virtualNetworks_flukebi_use_vnet_01_externalid": {  
            "defaultValue": "/subscriptions/52a1d076-bbbf-422a-9bf7-  
95d61247be4b/resourceGroups/flukebi-net-hub-  
001/providers/Microsoft.Network/virtualNetworks/flukebi-use-vnet-01",  
            "type": "String"  
        }  

```

## ARM Template Explanation

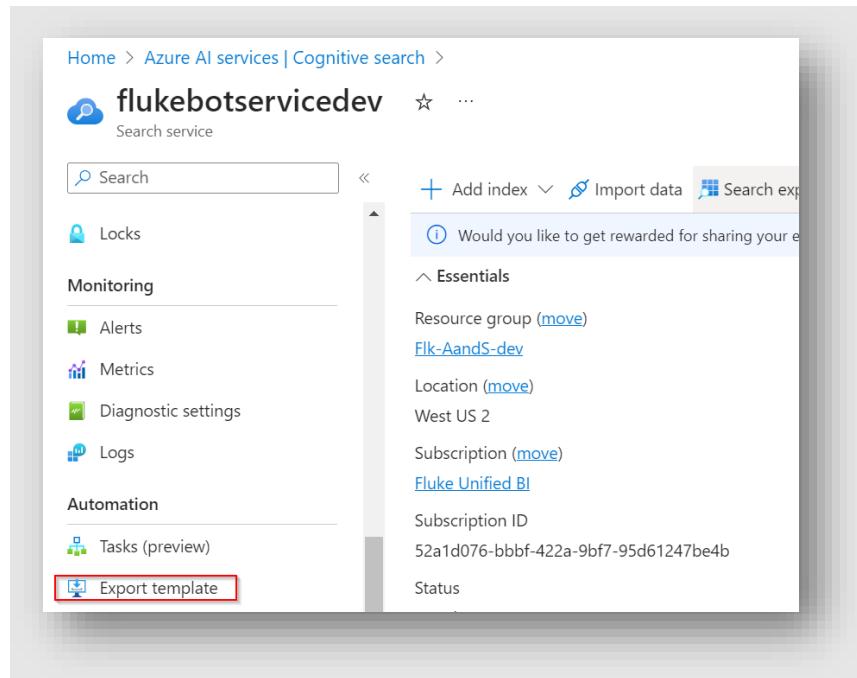
```
"parameters":{  
    "storageAccounts_flkubiadlsdev_name":{  
        "defaultValue": "flkubiadlsdev"  
        "type": "String"  
    },  
    "factories_flkubi_adf_dev_externalid":{  
        "defaultValue": "/subscriptions/52a1d076-bbbf-422a-9bf7-  
95d61247be4b/resourceGroups/flkubi-dev-rg-  
001/providers/Microsoft.DataFactory/factories/flkubi-adf-dev",  
        "type": "String"  
    },  
},
```

```
"type": "Microsoft.Storage/storageAccounts",  
"apiVersion": "2023-01-01",  
"name": "[parameters('storageAccounts_flkubiadlsdev_name')]",  
"location": "eastus",  
"sku":{  
    "name": "Standard_RAGRS",  
    "tier": "Standard"  
},  
"kind": "StorageV2",
```

The screenshot shows the Azure Storage account settings for 'flkubiadlsdev'. The account is a 'StorageV2 (general purpose v2)' type, located in 'eastus'. It has 'Read-access geo-redundant storage (RA-GRS)' replication and is set to 'Standard' performance. The account was created on 8/7/2021, 3:37:05 AM and is in a 'Succeeded' state. The resource group is 'flkubi-dev-rg-001'. The location is 'eastus'. The primary/secondary location is 'Primary: East US, Secondary: West US'. The subscription is 'Fluke Unified BI'. The subscription ID is 52a1d076-bbbf-422a-9bf7-95d61247be4b.

## Azure Cognitive Search Services

Azure Cognitive Search Services resource template can also be exported in the same way as shown below.



Sample ARM template for Azure Cognitive Search Services is attached below.

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "searchServices_flukebotservicedev_name": {
            "defaultValue": "flukebotservicedev",
            "type": "String"
        }
    },
    "variables": {},
    "resources": [
        {
            "type": "Microsoft.Search/searchServices",
            "apiVersion": "2022-09-01",
            "name": "[parameters('searchServices_flukebotservicedev_name')]",
            "location": "West US 2",
            "tags": {
                "ProjectType": "aoai-your-data-service"
            },
            "sku": {
                "name": "standard"
            },
            "identity": {
                "type": "SystemAssigned"
            },
            "properties": {
                "replicaCount": 1,
                "partitionCount": 1,
                "hostingMode": "Default",
                "publicNetworkAccess": "Enabled",
                "networkRuleSet": {
                    "ipRules": []
                },
                "encryptionWithCmk": {
                    "enforcement": "Unspecified"
                },
                "disableLocalAuth": false,
                "authOptions": {
                    "apiKeyOnly": {}
                }
            }
        }
    ]
}
```

## ARM Template Explanation

ARM Template Snippet:

```

"resources": [
    {
        "type": "Microsoft.Search/searchServices",
        "apiVersion": "2022-09-01",
        "name": "[parameters('searchServices_flukebotservicedev_name')]",
        "location": "West US 2",
        "tags": {
            "ProjectType": "aoai-your-data-service"
        },
        "sku": {
            "name": "standard"
        },
        "identity": {
            "type": "SystemAssigned"
        },
        "properties": {
            "replicaCount": 1,
            "partitionCount": 1,
            "hostingMode": "Default",
            "publicNetworkAccess": "Enabled",
            "networkRuleSet": {
                "ipRules": []
            }
        }
    }
]

```

Resource Details (Screenshot):

Essentials

- Resource group (move) Fluke-AndS-dev
- Location (move) West US 2
- Subscription (move) Fluke Unified BI
- Subscription ID 52a1d076-bbbf-422a-9bf7-95d61247be4b
- Status Running
- Tags (edit) ProjectType : aoai-your-data-service

Url https://flukebotservicedev.search.windows.net

Pricing tier Standard

Replicas 1 (No SLA)

Partitions 1

Search units 1

ARM Template Snippet:

```

    },
    "properties": {
        "replicaCount": 1,
        "partitionCount": 1,
        "hostingMode": "Default",
        "publicNetworkAccess": "Enabled",
        "networkRuleSet": {
            "ipRules": []
        },
        "encryptionWithCmk": {
            "enforcement": "Unspecified"
        },
        "disableLocalAuth": false,
        "authOptions": {
            "apiKeyOnly": {}
        }
    }
}

```

Network

Connectivity mode	Public
Private endpoints	0

Others

API access control	API keys
Semantic search	Free
Identity	System assigned

ARM Template Snippet:

```

    "location": "West US 2",
    "tags": {
        "ProjectType": "aoai-your-data-service"
    },
    "sku": {
        "name": "standard"
    },
    "identity": {
        "type": "SystemAssigned"
    }
}

```

Properties

Management	Current 14, Quota 50
Usage	Current 14, Quota 50
Monitoring	Current 13, Quota 50
	Current 0, Quota 100
	Current 4, Quota 50
Quotas	0

Network

Connectivity mode	Public
Private endpoints	0

Others

API access control	API keys
Semantic search	Free
Identity	System assigned

## Azure Open AI Services

Azure Open AI resource ARM template can be exported from the website. It contains details about Open AI resource along with pre-deployed models. ARM template can be deployed to a particular environment and pre-deployed GPT model deployment can also be provisioned with same name.

The screenshot shows the Azure OpenAI service configuration page for 'FlukeBOTDemoAI'. The left sidebar includes sections for Locks, Monitoring (Alerts, Metrics, Diagnostic settings, Logs), and Automation (Tasks (preview)). The main panel displays 'Essentials' information: Resource group (Fluke-AaaS-dev), Status (Active), Location (East US), Subscription (Fluke Unified BI), Subscription ID (52a1d076-bbbf-422a-9bf7-95d61247be4b), and Tags (edit, Add tags). At the bottom left of the main panel, there is a red box highlighting the 'Export template' button, which has a computer monitor icon and the text 'Export template'.

Sample Open AI ARM template is attached below.

```

{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "accounts_FlukeBOTDemoAI_name": {
            "defaultValue": "FlukeBOTDemoAI",
            "type": "String"
        }
    },
    "variables": {},
    "resources": [
        {
            "type": "Microsoft.CognitiveServices/accounts",
            "apiVersion": "2023-05-01",
            "name": "[parameters('accounts_FlukeBOTDemoAI_name')]",
            "location": "eastus",
            "sku": {
                "name": "S0"
            },
            "kind": "OpenAI",
            "properties": {
                "customSubDomainName": "flukebotdemoai",
                "networkAcls": {
                    "defaultAction": "Allow",
                    "virtualNetworkRules": [],
                    "ipRules": []
                },
                "publicNetworkAccess": "Enabled"
            }
        },
        {
            "type": "Microsoft.CognitiveServices/accounts/deployments",
            "apiVersion": "2023-05-01",
            "name": "[concat(parameters('accounts_FlukeBOTDemoAI_name'), '/DemoGP
TDeployment2')]",
            "dependsOn": [
                "[resourceId('Microsoft.CognitiveServices/accounts', parameters(
accounts_FlukeBOTDemoAI_name))]"
            ],
            "sku": {
                "name": "Standard",
                "capacity": 88
            },
            "properties": {

```

```

"resources": [
  {
    "type": "Microsoft.CognitiveServices/accounts",
    "apiVersion": "2023-05-01",
    "name": "[parameters('accounts_FlukeBOTDemoAI_name')]",
    "location": "eastus",
    "sku": {
      "name": "S0"
    },
    "kind": "OpenAI",
    "properties": {
      "customSubDomainName": "flukebotdemoai",
      "networkAcls": {
        "defaultAction": "Allow",
        "virtualNetworkRules": [],
        "ipRules": []
      },
      "publicNetworkAccess": "Enabled"
    }
  }
]

```

Go to Azure OpenAI Studio Delete

**Essentials**

Resource group (move) Flik-Aands-dev

Status Active

Location East US

Subscription (move) Fluke Unified BI

Subscription ID 52a1d076-bbbf-422a-9bf7-95d61247be4b

Tags (edit) Add tags

API Kind OpenAI

Pricing tier Standard

Endpoints Click here to view endpoints

Manage keys Click here to manage keys

```

  {
    "type": "Microsoft.CognitiveServices/accounts/deployments",
    "apiVersion": "2023-05-01",
    "name": "[concat(parameters('accounts_FlukeBOTDemoAI_name'), '/DemoOp
enAIGPT')]",
    "dependsOn": [
      "[resourceId('Microsoft.CognitiveServices/accounts', parameters(
accounts_FlukeBOTDemoAI_name'))]"
    ],
    "sku": {
      "name": "Standard",
      "capacity": 120
    },
    "properties": {
      "model": {
        "format": "OpenAI",
        "name": "gpt-35-turbo",
        "version": "0301"
      },
      "versionUpgradeOption": "OnceNewDefaultVersionAvailable",
      "raiPolicyName": "Microsoft.Default"
    }
  }
]

```

Deployment name	Model name	M...	Capacity	Status	Model dep...	Content Fil...
DemoOpenAIGPT	gpt-35-turbo	0301	Standard	120K TPM	Succeeded	7/5/2024
DemoGPTDeployment2	gpt-35-turbo-16k	0613	Standard	88K TPM	Succeeded	3/31/2024

```

  "sku": {
    "name": "Standard",
    "capacity": 88
  },
  "properties": {

```

Deployment name	Model name	M...	Capacity	Status	Model dep...	Content Fil...
DemoOpenAIGPT	gpt-35-turbo	0301	Standard	120K TPM	Succeeded	7/5/2024
DemoGPTDeployment2	gpt-35-turbo-16k	0613	Standard	88K TPM	Succeeded	3/31/2024
DemoGPTDeployment3	gpt-35-turbo-16k	0613	Standard	88K TPM	Succeeded	3/31/2024

```

    "model": {
      "format": "OpenAI",
      "name": "gpt-35-turbo-16k",
      "version": "0613"
    },
  }
]
```

# ARM Template Best Practices

## *Size Limits*

1. Limit the size of your ARM template to 4 MB.
2. Each resource definition within the template should not exceed 1 MB.
3. These limits pertain to the final state of the template, including iterative resource definitions, variable and parameter values.
4. The parameter file also has a 4 MB limit.
5. Smaller templates or parameter files may result in errors if the total request size becomes too large.

## *Other Limitations*

1. 256 parameters
2. 256 variables
3. 800 resources, which includes the copy count for resource repetitions.
4. 64 output values
5. 10 unique locations per subscription, tenant, or management group scope.
6. Templates are limited to 24,576 characters in template expressions.

## *Exceeding Limits with Nested Templates*

1. Some of these limits can be exceeded by utilizing nest templates.
2. For more details on using nested templates, refer to "Using linked and nested templates when deploying Azure resources."

## *Combining Values to Reduce Limits*

1. To stay within these limits, you can combine multiple values into an object, reducing the number of parameters, variables, or outputs.

## Validating and Testing ARM Template

Azure provides ARM Template Testing Toolkit for testing the ARM templates. It checks the and validates the code, format of the template, variable restriction are followed or not and performs other sanity tests.

Following is the list of different code sanity checks performed by ARM-TTK.

1. [Use correct schema](#)
2. [Declared parameters must be used](#)
3. [Secure parameters can't have hard-coded default](#)
4. [Environment URLs can't be hard-coded](#)
5. [Location uses parameter](#)
6. [Resources should have location](#)
7. [VM size uses parameter](#)
8. [Min and max values are numbers](#)
9. [Artifacts parameter defined correctly](#)
10. [Declared variables must be used](#)
11. [Dynamic variable should not use concat](#)
12. [Use recent API version](#)
13. [Use hard-coded API version](#)
14. [Properties can't be empty](#)
15. [Use Resource ID functions](#)
16. [Resourceld function has correct parameters](#)
17. [dependsOn best practices](#)
18. [Nested or linked deployments can't use debug](#)
19. [Admin user names can't be literal value](#)
20. [Use latest VM image](#)
21. [Use stable VM images](#)
22. [Don't use ManagedIdentity extension](#)
23. [Outputs can't include secrets](#)

24. [Use protectedSettings for commandToExecute secrets](#)
25. [Use recent API versions in reference functions](#)
26. [Use type and name in resourceId functions](#)
27. [Use inner scope for nested deployment secure parameters](#)

All the details can be checked from this [documentation](#).

To test and validate template manually using ARM TTK,

1. [Install Power Shell](#) on the system and start the Power Shell.
2. [Download the latest zip](#) for the Azure TTK and extract it to a folder.
3. Navigate arm-ttk folder inside tool kit extraction folder.

```
PS C:\Users\...\Downloads> cd .\Downloads\arm-ttk
```

4. Run following script to unblock the shell from accessing the file scripts.

```
Get-ChildItem *.ps1, *.psd1, *.ps1xml, *.psm1 -Recurse | Unblock-File
```

```
PS C:\Users\...\Downloads\arm-ttk> Get-ChildItem *.ps1, *.psd1, *.ps1xml, *.psm1 -Recurse | Unblock-File
```

5. Import the module

```
Import-Module .\arm-ttk.psd1
```

```
PS C:\Users\...\Downloads\arm-ttk> cd .\arm-ttk\
```

```
PS C:\Users\...\Downloads\arm-ttk\arm-ttk> Import-Module .\arm-ttk.psd1
```

```
PS C:\Users\...\Downloads\arm-ttk\arm-ttk\arm-ttk>
```

6. Run the following command to test the template.

```
Test-AzTemplate -TemplatePath \path\to\template
```

```
PS C:\Users\...Downloads\arm-ttk\arm-ttk\arm-ttk> Test-AzTemplate -TemplatePath C:/Users/...Downloads/ARMT
emplate_0.json

Validating Downloads\ArmTemplate_0.json
  JSONFiles Should Be Valid (32 ms)
    [+] JSONFiles Should Be Valid (32 ms)

Pass : 1
Total : 1
Fail  : 0

  adminUsername Should Not Be A Literal
    [+] adminUsername Should Not Be A Literal (262 ms)
```

7. Output will be generated like given below.
  - a. Pass cases will be green.
  - b. Failed cases will be red.
  - c. Warnings will be highlighted as Yellow

The output will provide detailed review of the ARM template which can be further analyzed and fixed. Above same steps can be added in the Azure DevOps CI/CD pipeline PowerShell task and the ARM template of the build can be validated in the Azure Pipeline. The original git repo with file present can be referenced for this.

# Integrate ARM Template in Azure Pipeline

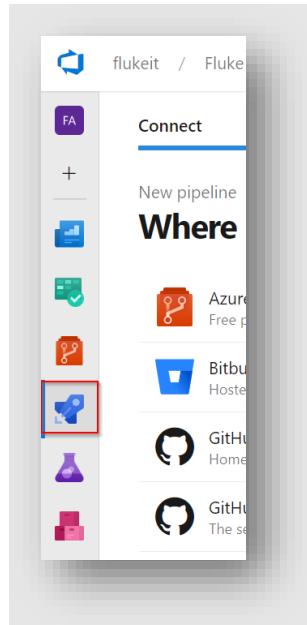
ARM templates can be integrated in Azure CI/CD pipelines for maintenance and versioning of the resource. Some of the resources like Azure Data Factory are fully configured in ARM templates. To integrate the deployment with Azure pipelines, following perquisites needs to be satisfied.

## Pre-requisites:

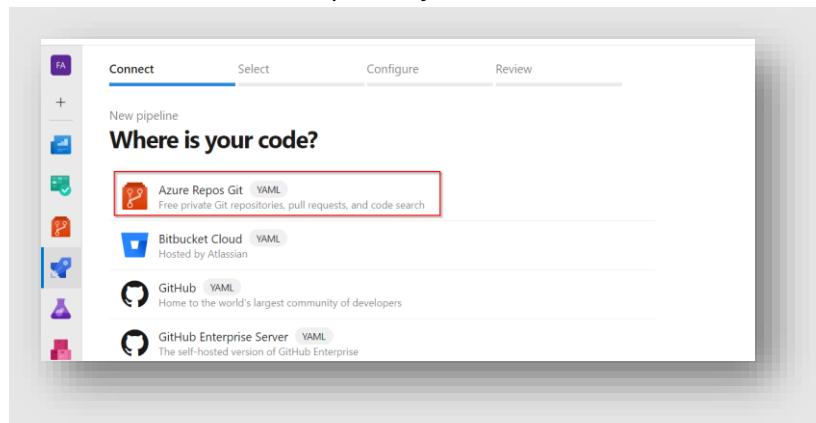
1. Azure repository must created and configured.
2. ARM template must be pushed to repository or power shell command must be used to generate ARM template from the given repo files.
3. ARM template deployment script using power shell must be added in the repository.

## Integration:

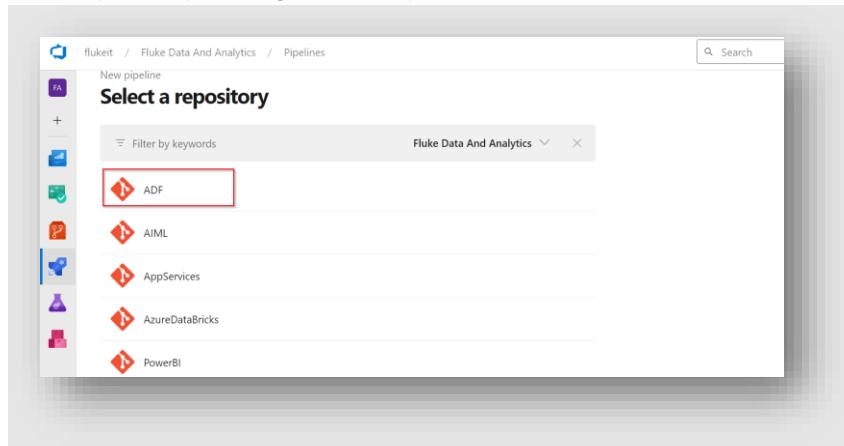
1. Open the pipelines in the Azure DevOps (dev.azure.com).



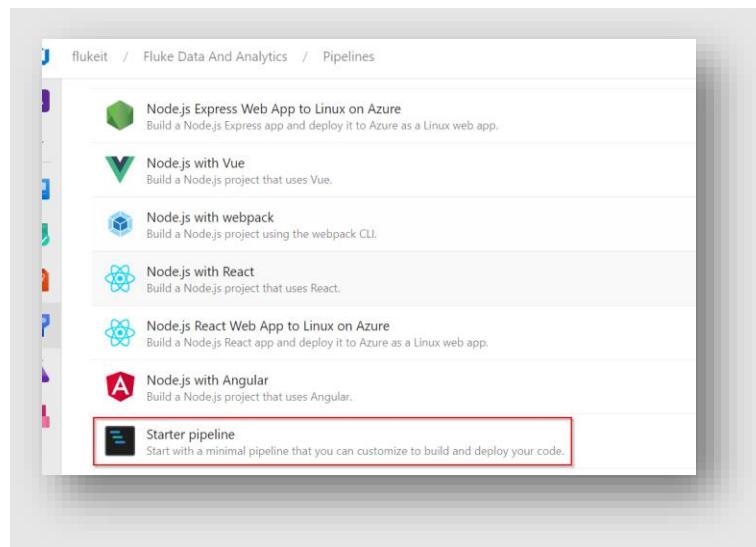
2. In the connect section, select Azure Repository/Azure GIT.



3. Select particular repository having ARM template.



4. Click on Starter pipeline to edit the code manually. It will open up an YAML file having basic code.



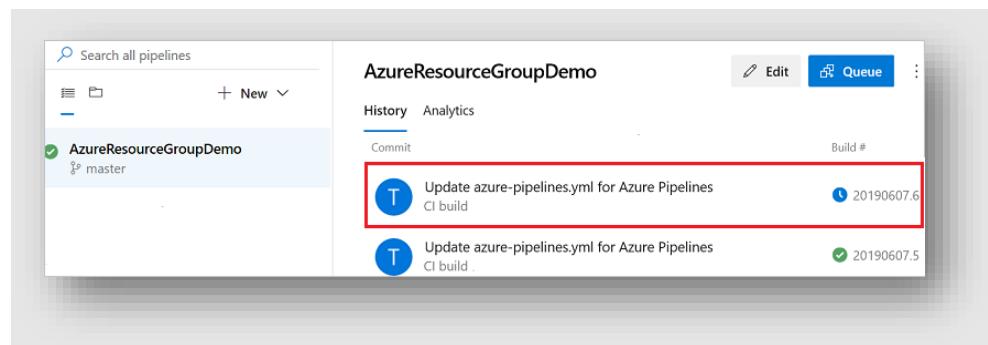
5. ARM template deployment with Power Shell will be used here. Add following sample code.

```
trigger:
- master

pool:
  vmImage: 'ubuntu-latest'

steps:
- task: AzurePowerShell@5
  inputs:
    azureSubscription: 'script-connection'
    ScriptType: 'FilePath'
    ScriptPath: './Deploy-AzTemplate.ps1'
    ScriptArguments: '-Location 'centralus' -ResourceGroupName 'demogroup' -TemplateFile templates\mainTemplate.json'
    azurePowerShellVersion: 'LatestVersion'
```

6. Update the subscription name , script argument names based on your script.
7. Save and queue the run of DevOps pipeline.



8. For detailed instructions, please refer to the [Microsoft Documentation](#).

# How to Deploy ARM Templates

**ARM templates can be deployed with multiple ways.**

1. Azure CLI
2. Azure Portal – Custom Deployment
3. Azure Power Shell
4. Python Code
5. Rest API
6. Cloud Shell
7. Deploy-to-Azure-Button

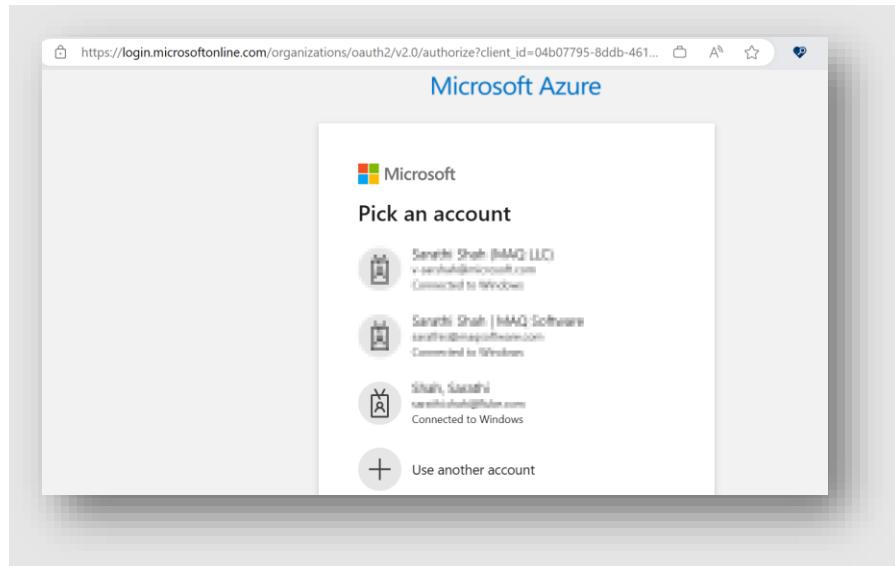
## Deployment of ARM Template Using Azure CLI

1. Once the ARM template file is ready, save it with appropriate name in the System.
  - a. If Azure CLI is not installed, please refer to this [Link](#) for Azure CLI installation.
  - b. An .msi files needs to be downloaded and executed to install the Azure CLI on windows.
  - c. Installation can be verified by running ‘az’ command in the CMD.
2. Now open the Command Prompt.
3. Now run the command az login in the the CMD.

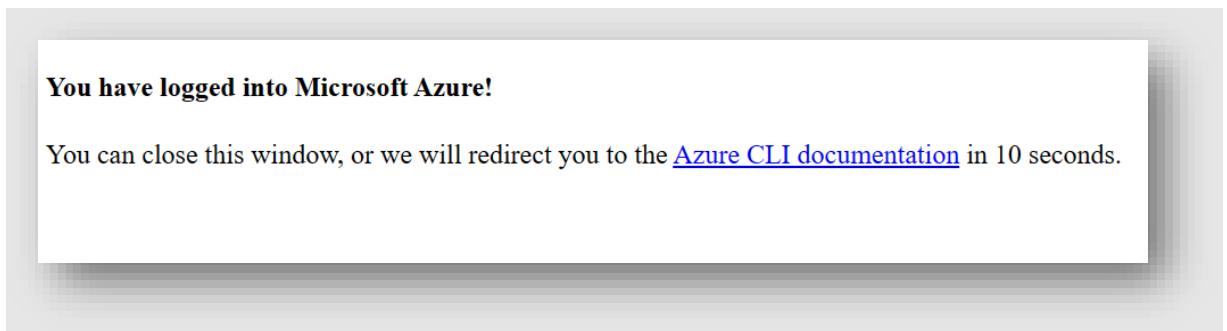


```
c:\Users\...\Desktop>az login
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue
the login in the web browser. If no web browser is available or if the web browser fails to open, use device code flow w
ith 'az login --use-device-code'.
```

4. Login with proper ID-Password from this open prompt window.



5. After successful login it will show following message.



6. Now to connect to particular subscription, run `az account set --subscription SubscriptionName`
7. Create a resource group under which the template needs to created. For creating a Resource Group, run the following command.

```
az group create \
--name myResourceGroup \
--location 'Central US'
```

8. Once Resource Group is created, start the deployment of the resource. Run the following command to deploy the resource.

```
templateFile="{provide-the-path-to-the-template-file}"
az deployment group create \
--name blanktemplate \
--resource-group myResourceGroup \
--template-file $templateFile
```

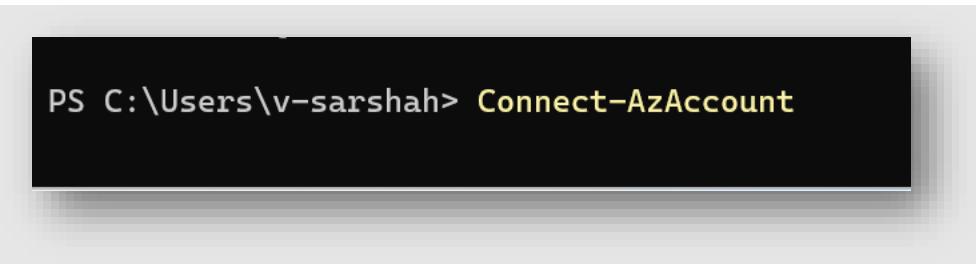
9. Once the deployment is succeeded, the screen will show up like following.



```
"parameters": {},  
"parametersLink": null,  
"providers": [],  
"provisioningState": "Succeeded",  
"template": null,  
"templateHash": "9132645722898483367",
```

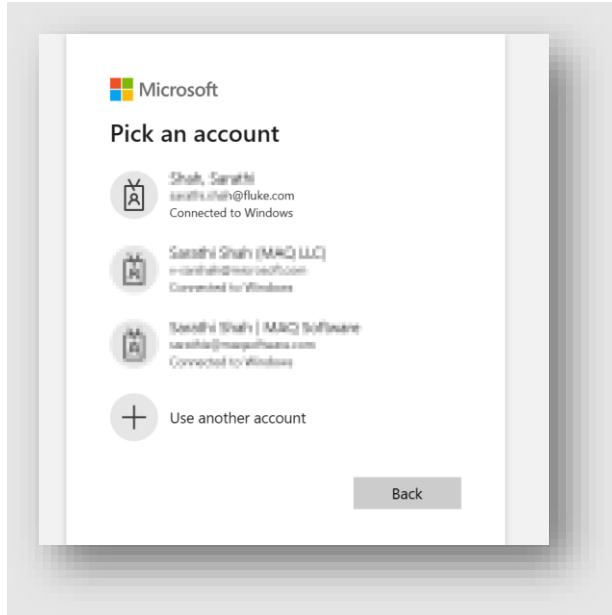
## Deploy ARM Template using Power Shell

1. Once the ARM template file is ready, save it with appropriate name in the System.
2. To deploy using the Azure Power Shell, Power Shell must be installed in the system. If it is not installed in the system, refer the [documentation](#) for the installation
3. Open the Power Shell.
4. Connect to Az account using following command.



```
PS C:\Users\v-sarshah> Connect-AzAccount
```

5. An authentication window will open. Sign in using relevant credentials.



6. Once logged in, set the deployment scope for resource type to be deployed.

- a. To deploy to a **resource group**, use [New-AzResourceGroupDeployment](#).

```
New-AzResourceGroupDeployment -ResourceGroupName <resource-group-name> -TemplateFile <path-to-template>
```

- b. To deploy to a **subscription**, use [New-AzSubscriptionDeployment](#)

```
New-AzSubscriptionDeployment -Location <location> -TemplateFile <path-to-template>
```

- c. To deploy to a **management group**, use [New-AzManagementGroupDeployment](#).

```
New-AzManagementGroupDeployment -Location <location> -TemplateFile <path-to-template>
```

- d. To deploy to a **tenant**, use [New-AzTenantDeployment](#).

```
New-AzTenantDeployment -Location <location> -TemplateFile <path-to-template>
```

7. Now create a resource group if required.

```
New-AzResourceGroup -Name ExampleGroup -Location "Central US"
```

8. Run the following command to deploy the local file.

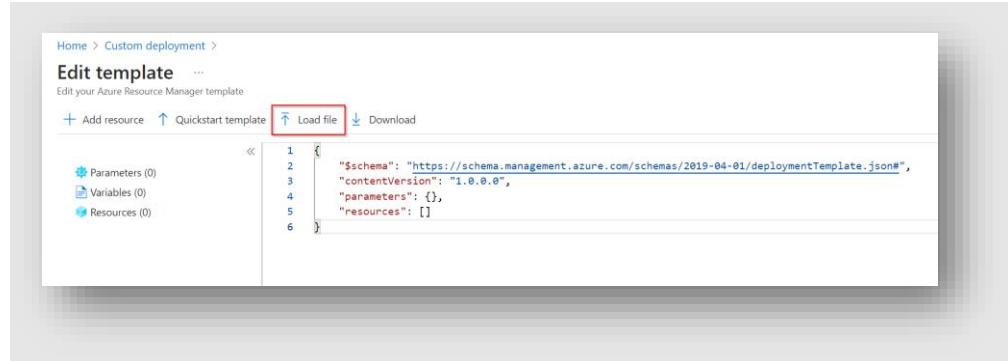
```
New-AzResourceGroupDeployment `<br> -Name ExampleDeployment `<br> -ResourceGroupName ExampleGroup `<br> -TemplateFile <path-to-template>
```

## Deploy ARM Template using Azure Portal

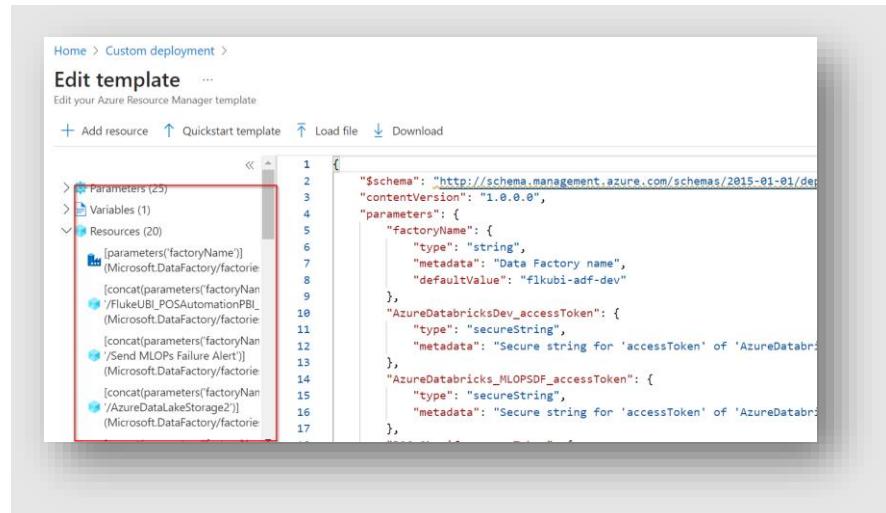
1. To deploy any ARM template via Azure Portal, custom deployment is required.
2. Open the Azure portal and search for Deploy a Custom Template.
3. Click on the Build your own template in the editor

The screenshot shows the 'Custom deployment' page in the Azure portal. At the top, there's a link to 'Home'. Below it, the title 'Custom deployment' with a '...' button, and a sub-instruction 'Deploy from a custom template'. Underneath, there are three tabs: 'Select a template' (which is underlined), 'Basics', and 'Review + create'. A note below the tabs says: 'Automate deploying resources with Azure Resource Manager templates in a single, coordinated operation. Create or select a template below to get started.' followed by a 'Learn more about template deployment' link. A prominent red rectangular box highlights the 'Build your own template in the editor' button, which has a blue pencil icon and the text 'Build your own template in the editor'. Below this button, there's a section titled 'Common templates' with several items: 'Create a Linux virtual machine', 'Create a Windows virtual machine', 'Create a web app', 'Create a SQL database', and 'Azure landing zone'.

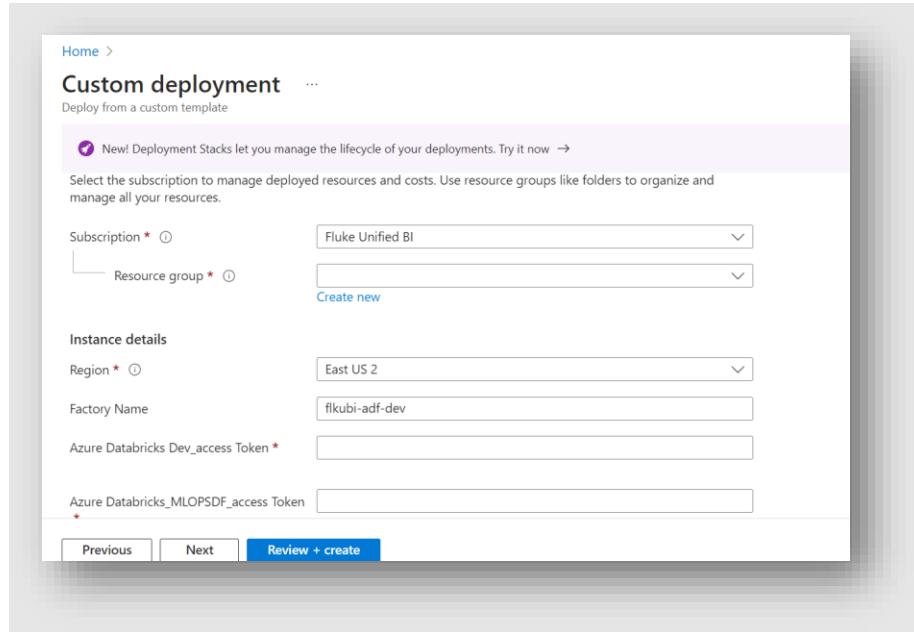
4. Copy your JSON code in the editor shown, or click on the Load File option and select your ARM template JSON file



5. Once the file is selected, all the parameters, resource information can be seen as shown below, click on Save.



6. Now fill the details for Azure Subscription, and other parameters.



7. Once all values are selected, click on Review and Create.

# ARM Template Deployment Modes

## Incremental and Complete Deployments

1. ARM templates support two deployment types: incremental and complete.
2. These deployment types determine how the Resource Manager service handles resources not defined in the template.

## Resource Creation Attempt

1. In both incremental and complete modes, the Resource Manager service attempts to create all resources specified in the template.

## Resource Matching and Unchanged State

1. If a resource already exists and matches the template resource, Azure leaves the resource unchanged.
2. Re-running commands from a previous section would not trigger any operations in Azure.

## Updating Resource Properties

1. When you modify one or more properties in a resource, Azure updates the resource with the new values.
2. For example, changing the storage SKU from Standard LRS to Standard GRS results in Azure updating the storage account resource.

## Deploying Using Complete Mode

### Deletion of Unspecified Resources

1. In complete mode, Resource Manager deletes resources that exist in the resource group but are not specified in the template.

### Template Condition and REST API Versions

1. If your template includes a resource that isn't deployed due to a condition evaluating to false, the outcome depends on the REST API version used for template deployment.
2. If deploying with a version before 2019-05-10, the resource is not deleted.
3. After 2019-05-10, the resource is no longer available, effectively deleted.
4. In recent versions of Azure PowerShell and Azure CLI, the resource is deleted.

### Caution with Complete Mode and Copy Loops

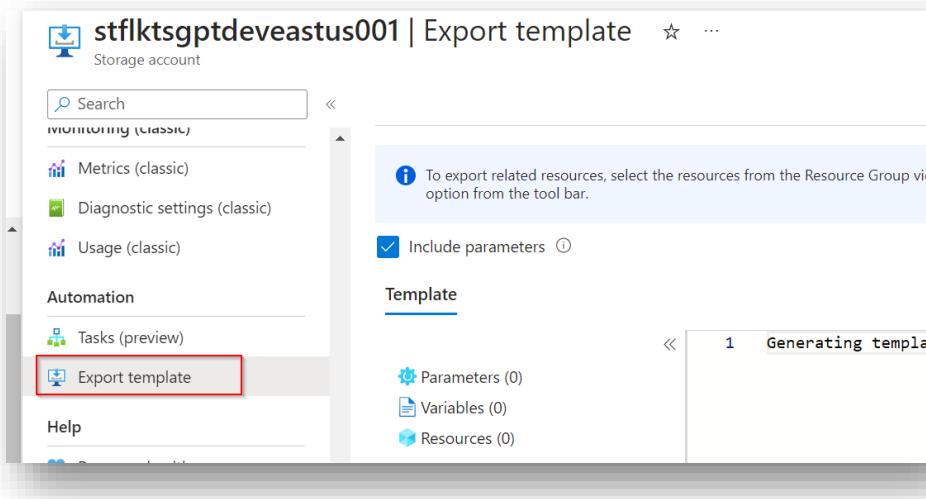
1. When using complete mode in templates with copy loops, exercise caution.
2. After resolving the copy loop, any resources not specified in the template are deleted.

# Resource Creation for Fluke Tech Support GPT

## Storage Account

### *Export the Template of Dev Resource*

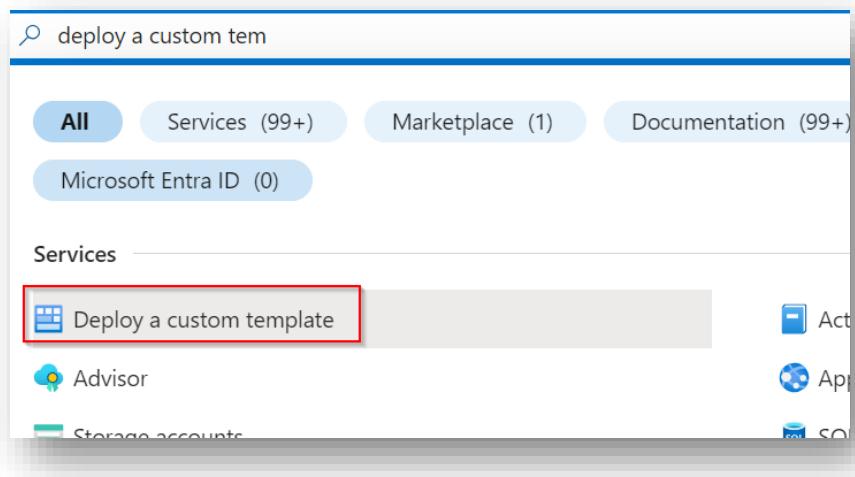
1. Open portal.azure.com
2. Open the storage account and on the menu click on the Export Template option.



3. Replace words related to Dev workspace to UAT / Prod in template and save it in JSON format.

### *Deploy the Template to UAT Resource Group*

1. Go to portal.azure.com and open 'Deploy a custom template'



2. Click on Build your own template in the editor.

3. Paste the JSON content in the editor and click on save.
4. Select/Update the required parameter values and click on deploy.

Home >

## Custom deployment

Deploy from a custom template

New! Deployment Stacks let you manage the lifecycle of your deployments. Try it now →

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ Fluke Unified BI

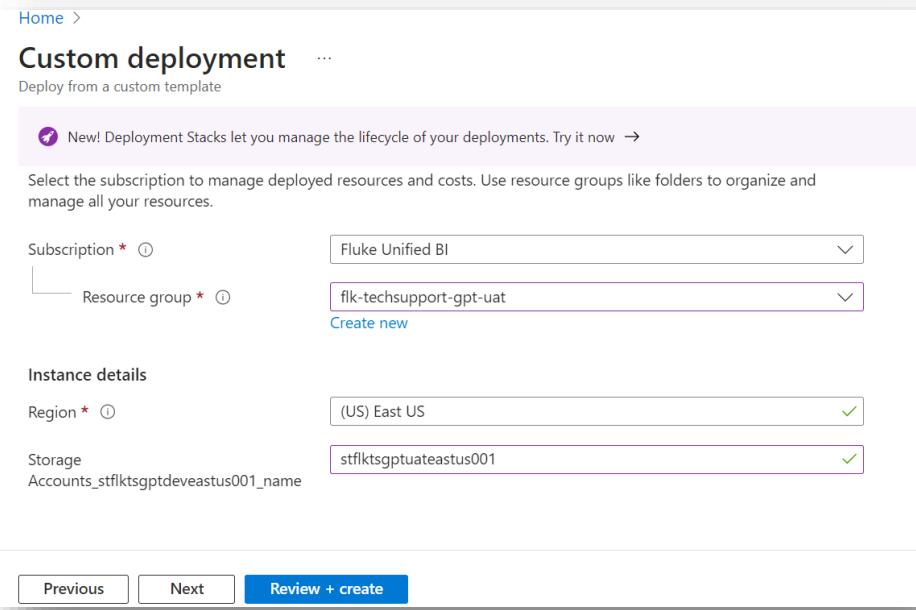
Resource group \* ⓘ flk-techsupport-gpt-uat  
Create new

Instance details

Region \* ⓘ (US) East US

Storage Accounts\_stflktsgrptdeveastus001\_name stflktsgrptuateastus001

Previous Next Review + create



## Azure SQL Server And Azure SQL DB

### *Exporting the ARM Templates*

1. For getting the ARM template for this, we will refer to the deployment made while creating the resource on Dev.
2. Go to portal.azure.com, and open the relevant resource group.
3. Click on the Deployment option from the menu on the left as shows below in the image.

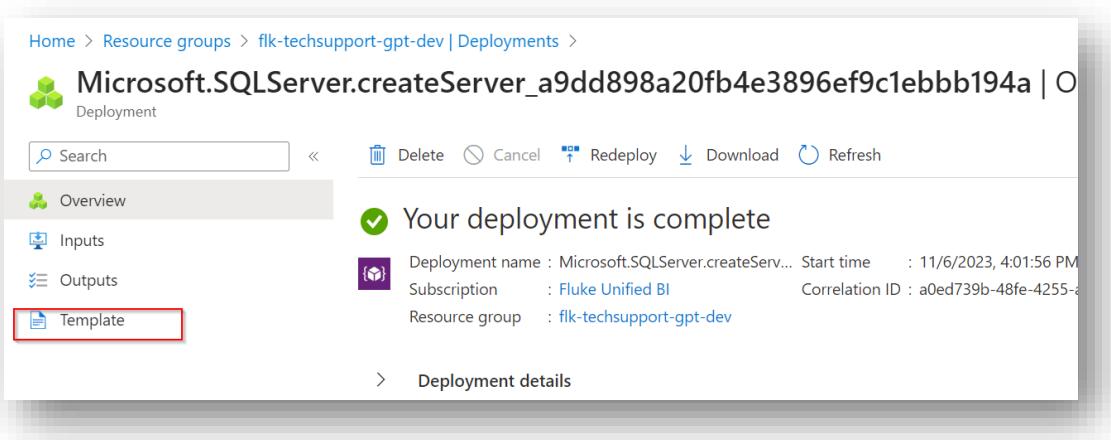
The screenshot shows the Azure Resource Group blade for 'flk-techsupport-gpt-dev'. At the top, there's a breadcrumb navigation: Home > Resource groups > flk-techsupport-gpt-dev. Below the title, there's a search bar and a sidebar with links: Overview, Activity log, Access control (IAM), Tags, Resource visualizer, Events, Settings, and Deployments. The 'Deployments' link is highlighted with a red box.

4. Now find the exact deployment for Azure SQL DB and Azure SQL Server.

The screenshot shows the 'flk-techsupport-gpt-dev | Deployments' blade. At the top, it says 'flk-techsupport-gpt-dev | Deployments'. Below that are buttons for Refresh, Cancel, Redeploy, Delete, and View template. There are three deployment entries in a table:

Action	Name	Status	Timestamp	Duration	Related
<input checked="" type="checkbox"/>	Microsoft.SQLDatabase.newData...	Succeeded	11/6/2023, 4:16:37 PM	1 minute, 17 seconds, 689 milliseconds	<a href="#">Related</a>
<input checked="" type="checkbox"/>	Microsoft.SQLServer.createServer...	Succeeded	11/6/2023, 4:03:24 PM	1 minute, 28 seconds, 76 milliseconds	<a href="#">Related</a>
<input type="checkbox"/>	Failure-Anomalies-Alert-Rule-De...	Succeeded	11/3/2023, 10:23:35 PM	1 second, 330 milliseconds	<a href="#">Related</a>

5. Open the SQL Server deployment and click on the template to generate the template



6. Update all “dev” keywords to uat to make it refer to UAT environment and resources.
  - a. Update the resource name accordingly.
  - b. If any dependsOn statement is used in the ARM template, make sure to verify that that resource for which dependency is there, exists on UAT environment or is getting created in the same ARM template.
7. Save the resource template with .json name.

Follow the same method and save the Azure SQL DB ARM Template.

#### *Deployment of the Resource (Azure SQL Server + Azure SQL DB)*

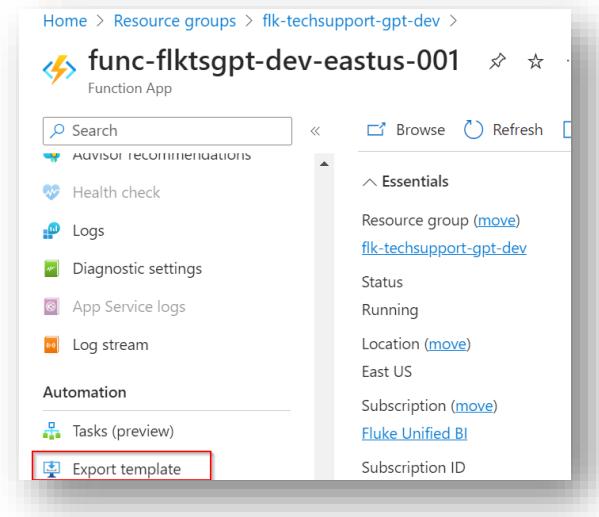
Please refer to the section [Deploy the Template to UAT Resource Group] for deploying the saved resource group.

1. Deploy the Azure SQL Server first.
2. Once SQL Server is deployed, deploy the Azure SQL DB.

## Azure Function App

*Export the Template:*

1. Open portal.azure.com
2. Open the Function App resource and on the menu click on the Export Template option.



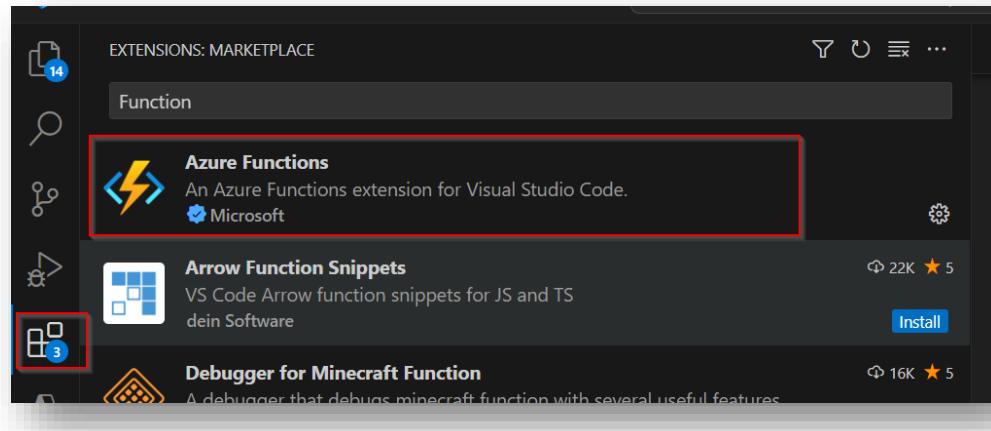
3. Replace words related to Dev workspace to UAT / Prod in template and save it in JSON format.
  - a. Verify resource group name and other things are aligned or not.
4. Save the file to .json format.

*Deploy the Template*

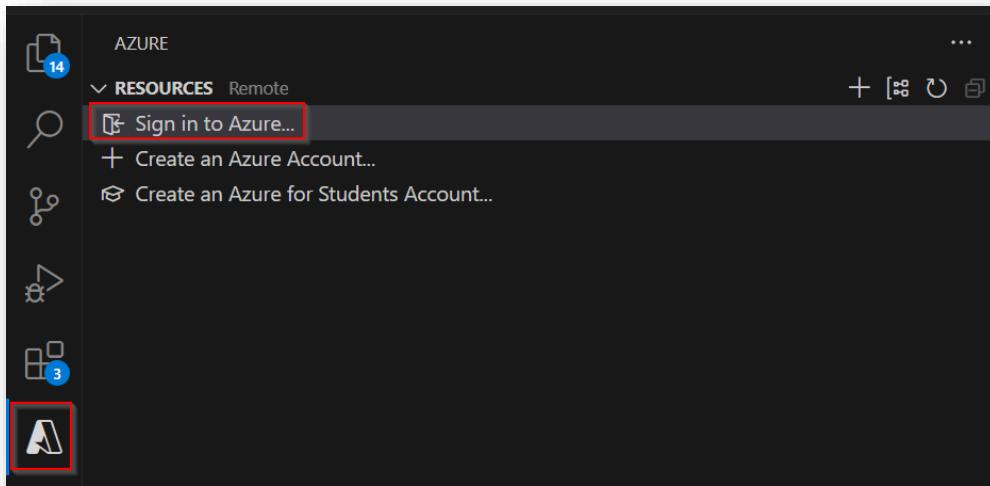
Please refer to the section [Deploy the Template to UAT Resource Group] for the template deployment.

*Function App Internal Resource/Function Creation*

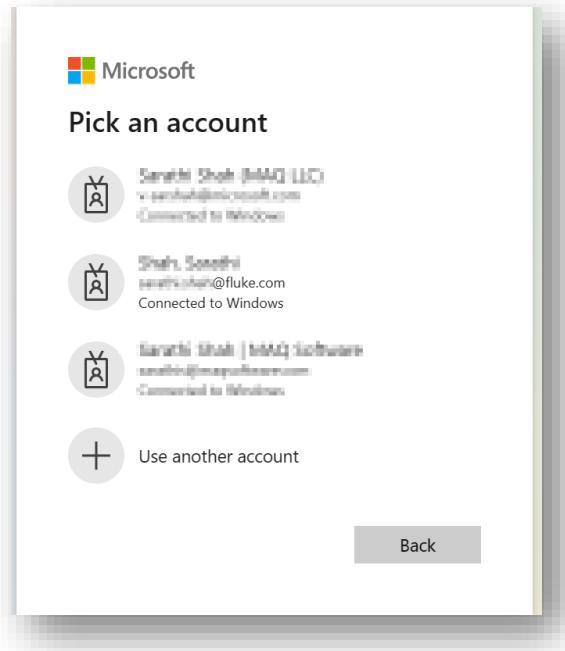
1. Function creation requires deployment from Visual Studio Code or Azure CLI Functions.
2. For Visual Studio installation, please refer to the steps mentioned in section [Prerequisites steps to create the ARM template in Visual Studio Code].
3. Install the Azure Functions extension.
  - a. Go to the extension in Visual Studio Code and search for Function App.



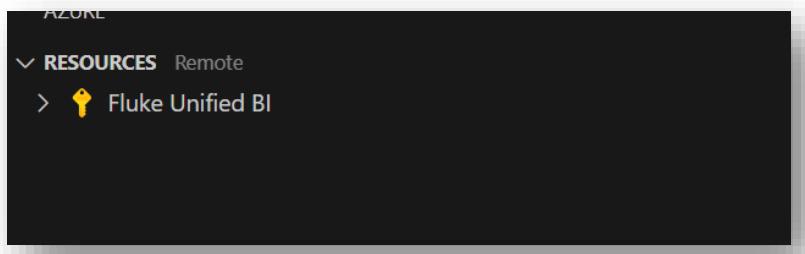
- b. Install the extension.  
4. Now, click on Azure Icon, and select Sign in to Azure.



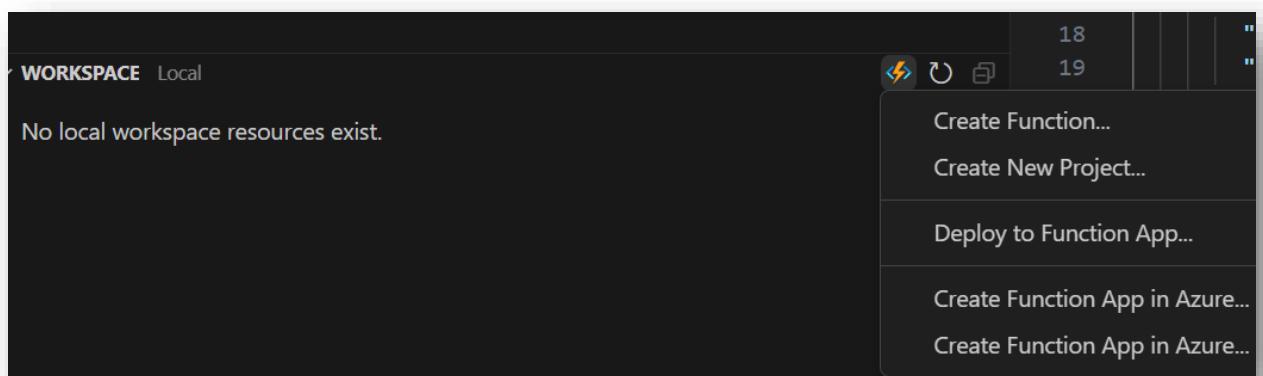
5. Authenticate with the credentials which have access to the resource group where deployment is to be done.



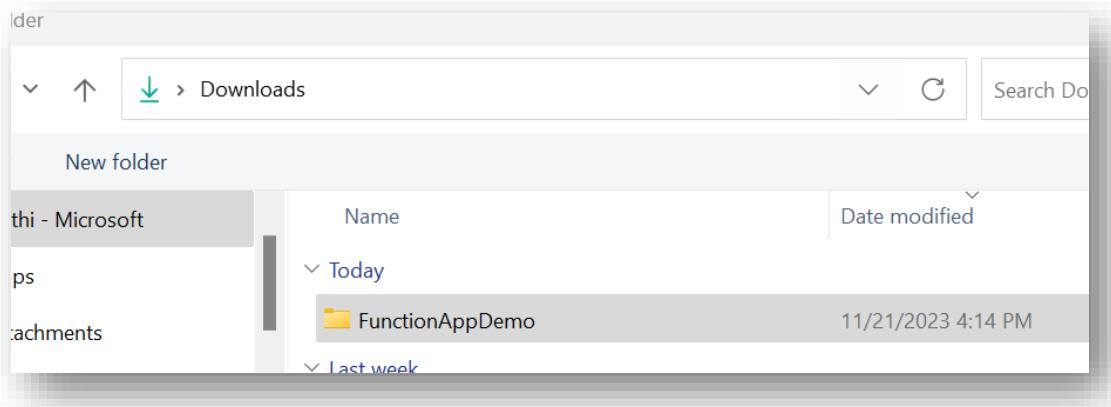
6. Once logged in successfully, all the accessible subscription will be visible.



7. Now, click on the function icon as shown below and select create a new function.

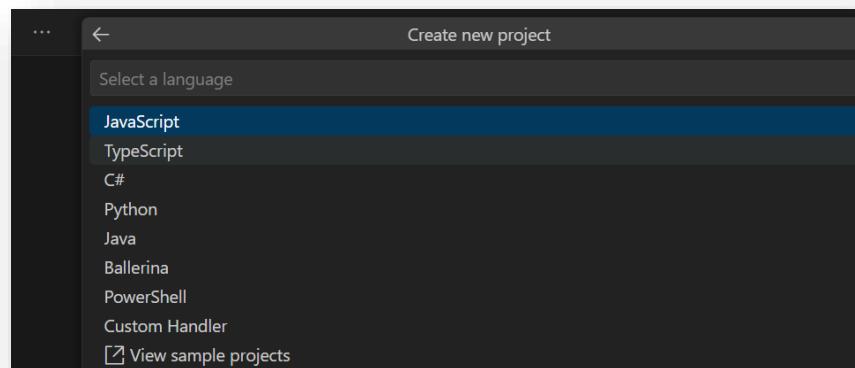


8. Select the folder where you want to create the function.



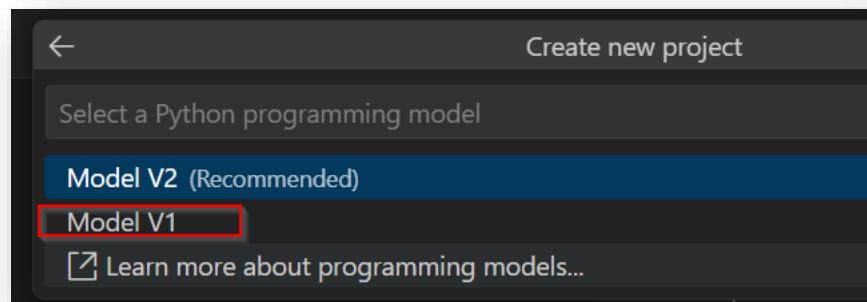
9. Now select the language of the Function App.

- a. The language can be checked from by viewing the function app files.

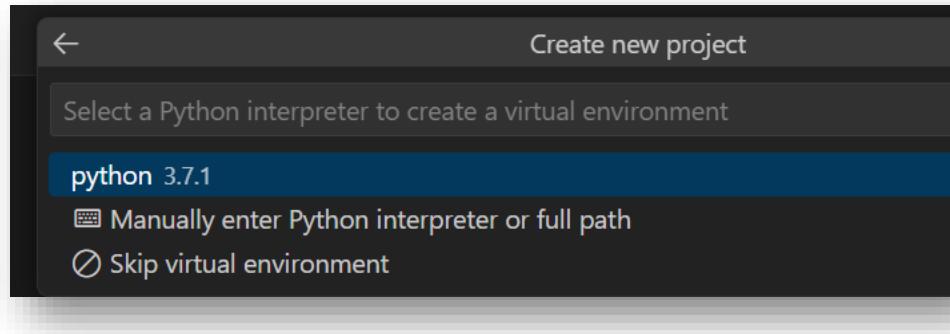


10. Now select the Model in which the original Function App was created.

- a. For both the models, the app structure is different.  
b. We have selected Model V1 as original app function on dev resource was created using Model V1.

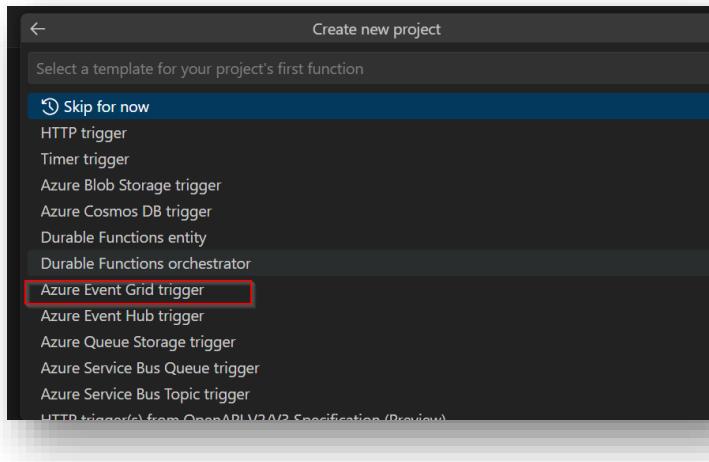


11. Select the python environment.



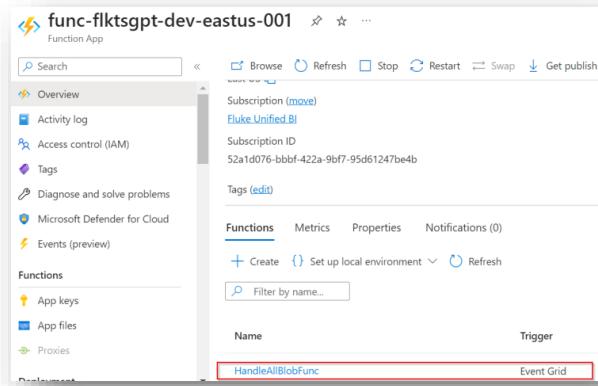
12. Select the trigger for the function app.

- We have selected the trigger based on the Dev resource observation.

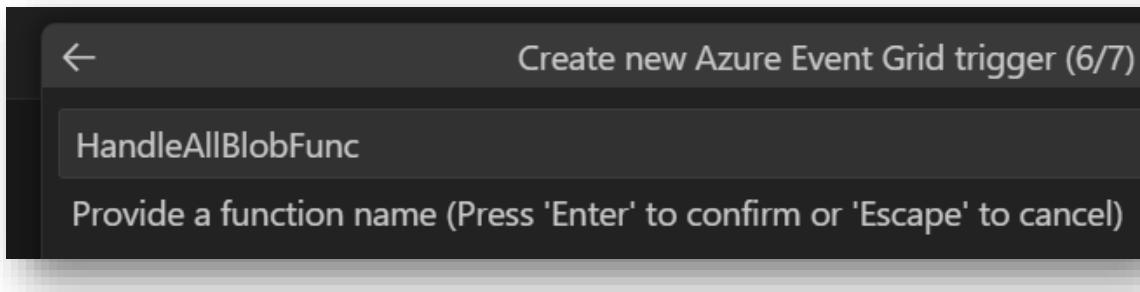


- Following image shows the attached trigger.

- To check the trigger, click on the Function listed in the Function App.



13. Add the function name.



14. Click on Open it in a new window.

15. Now copy the function.json file, \_\_init\_\_.py file code from Dev Function app.  
a. To get the code, click on the Function App.

- b. Click on Code + Test

16. Copy the relevant file's code with the same name in visual studio.

The screenshot shows two windows side-by-side. The top window is the 'func-flktsgpt-dev-eastus-001 \ HandleAllBlobFunc' editor in the Azure Functions portal. It displays a message: 'This function has been edited through an external editor. Portal editing is disabled.' Below is a code editor with the following Python code:

```
1 import os
2 import json
3 import logging
4 import azure.functions as func
5 from open import docs_chunk_run_index_name
```

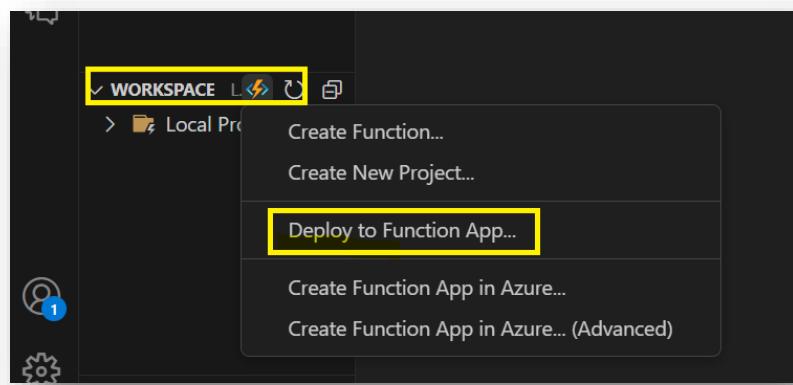
To the right of the code editor is a dropdown menu showing file options: \_\_init\_\_.py, \_\_init\_\_.py (selected), function.json, and prep.py.

The bottom window shows the local file system of the project 'FUNCTIONAPPROJ'. It contains files: .venv, .vscode, HandleAllBlobFunc, \_\_init\_\_.py (highlighted with a red box), function.json (highlighted with a red box), sample.dat, and .funcignore.

On the right, the 'function.json' file is being viewed in a code editor. Its contents are:

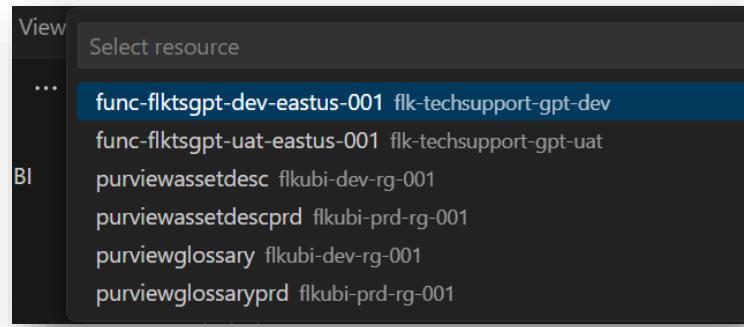
```
1 {
2     "scriptFile": "__init__.py",
3     "bindings": [
4         {
5             "type": "eventGridTrigger",
6             "name": "event",
7             "direction": "in"
8         }
9     ]
10 }
```

17. Now, click on the Deploy to Azure as shows in the image below.



18. Sign into Azure If required, as explained in steps 4 & 5.

19. Select the Resource Group.



20. Click on deploy



## Azure Cognitive Search Services

*Export the Template:*

1. Open portal.azure.com and go the Cognitive Search resource.

The screenshot shows the Azure Cognitive Search service blade for 'cog-flktsgpt-dev-eastus-001'. On the left, there's a navigation menu with 'Monitoring' and 'Automation' sections. On the right, there's a sidebar with 'Essentials' information like Resource group, Location, Subscription, and Subscription ID. At the bottom of the sidebar, the 'Export template' button is highlighted with a red box.

2. Click on Deploy.

The screenshot shows the 'Export template' blade for the same Cognitive Search service. It has buttons for 'Download', 'Add to library', 'Deploy' (which is highlighted with a red box), and 'Visualize template'. Below these are instructions to select resources for export and a checked checkbox for 'Include parameters'. The 'Template' tab is selected, showing a preview of the JSON template code.

```
1 {  
2   "$schema": "deploymentTemplate"  
}
```

- Click on the Edit template.

Custom deployment

Deploy from a custom template

New! Deployment Stacks let you manage the lifecycle of your deployments. Try it now →

Basics    Review + create

Template

Custom template ↗  
1 resource

Edit template    Edit parameters    Visualize

Project details

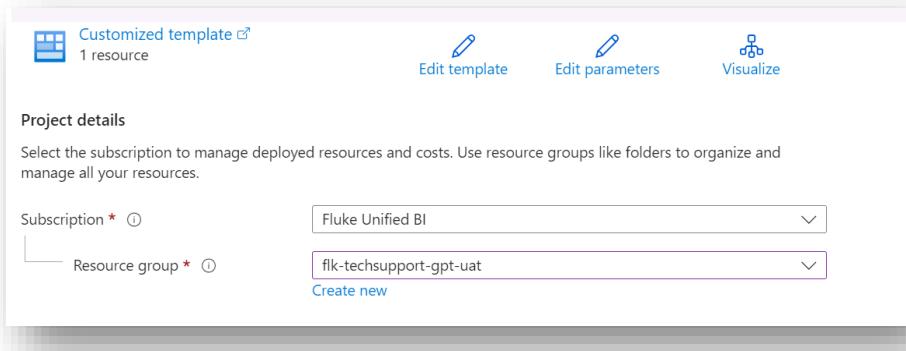
- Replace all 'dev' keyword to 'uat' for UAT naming conventions and click on save.

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "searchServices_cog_flktsqpt_dev_eastus_001_name": {
      "defaultValue": "cog-flktsqpt-dev-eastus-001",
      "type": "String"
    }
  },
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Search/searchServices",
      "apiVersion": "2023-11-01",
      "name": "[parameters('searchServices_cog_flktsqpt_dev_eastus_001_name')]"
    }
  ]
}
```

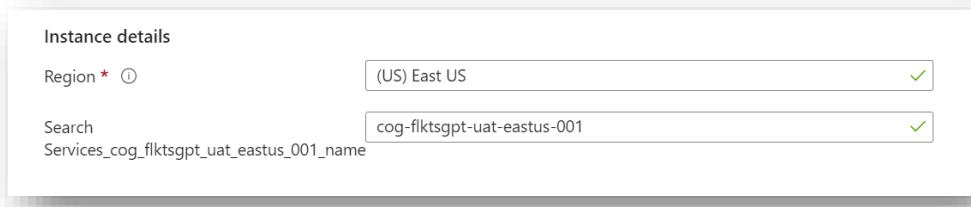
Replace the name 'dev' to 'uat'

Replace the parametFind and replace the parameter name in whole file, replace the 'dev' to 'uat'.

- Now once selected, select the Subscription, Resource Group for the deployment



## 6. Now verify the resource name.



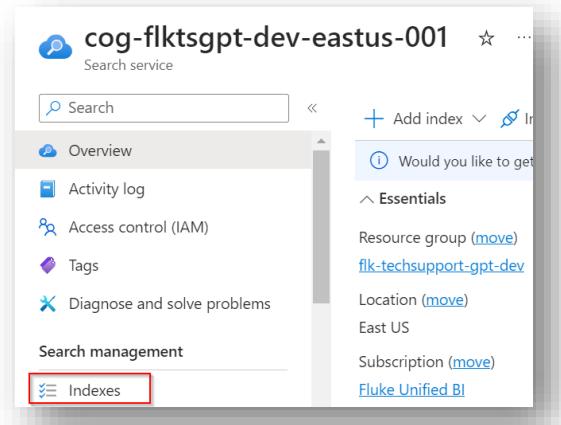
## 7. Click on Review + Create, and Create the resource.

### *Migrating Azure Cognitive Search Index*

Azure Cognitive Search Index is not considered as resource and therefore not included in the ARM template.

Follow the steps to migrate the index from Dev to UAT manually.

## 1. In the portal.azure.com, Under the Azure Cognitive Search Dev Resource, go to the Index.



2. Open the index created on the dev resource.

cog-flktsgpt-dev-eastus-001 | Indexes

Search

Add index Refresh Delete

Overview Activity log Access control (IAM) Tags Diagnose and solve problems

Filter by name...

Name	Document C...
flktsgptindexnonprod-comprehension	146,345

3. Click on the Edit JSON.

flktsgptindexnonprod-comprehension

Save Discard Refresh Create Demo App Edit JSON Delete

Documents 146,345 Storage 146.69 MB

Search explorer Fields CORS Scoring profiles Semantic configurations Vector profiles

4. Save the JSON definition to a .json file.
5. Open the file and remove '@odata' related lines. Please refer to the image below. There can be more lines.

## *Deploy the Index Manually*

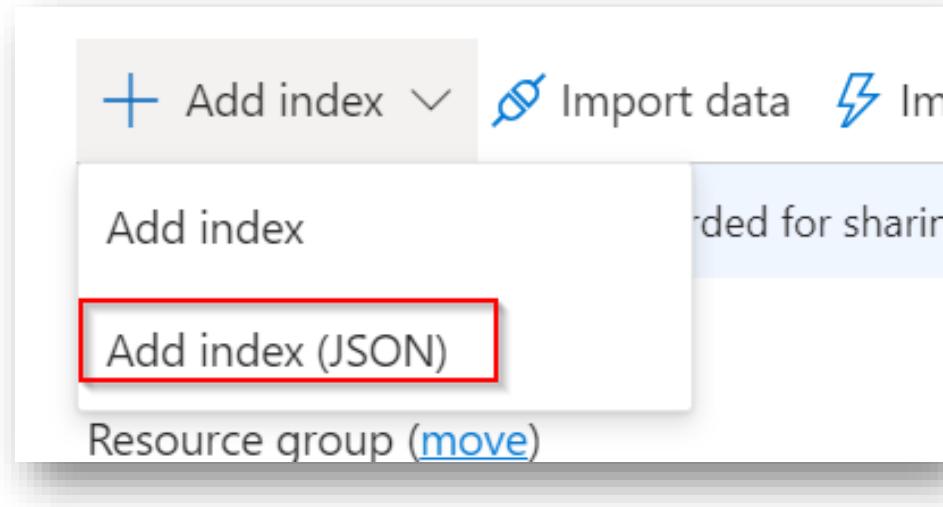
1. Under the portal.azure.com, open the UAT Cognitive Search Service resource.

The screenshot shows the Azure Resource Groups blade. On the left, a list of resource groups is shown, with 'flk-techsupport-gpt-uat' selected and highlighted with a red box. In the center, under the selected resource group, there is a 'Settings' sidebar with options like Deployments, Security, Deployment stacks, Policies, Properties, and Locks. To the right, the main pane displays the 'Essentials' section, which includes a search bar, a 'Resources' tab (selected), and a 'Recommendations' tab. Below the tabs, there is a list of resources, with 'cog-flktsgpt-uat-eastus-001' highlighted with a red box. A filter bar at the top right allows filtering by type (set to 'all') and name.

2. Click on the index from the left menu.

The screenshot shows the Cognitive Search service blade. At the top, the service name 'cog-flktsgpt-uat-eastus-001' is displayed. The left sidebar has several menu items: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Search management, Indexes (which is highlighted with a red box), Indexers, and Data sources. On the right side, there are sections for 'Would you like to add an index?' (with a 'Yes' button), 'Essentials' (Resource group, Location, Subscription, Subscription ID), and a 'Logs' section.

3. Click on the Add Index -> Add Index (JSON).



4. Copy the saved definition to the Index creation editor.

The screenshot shows the 'Add index (JSON)' editor window. The title bar says 'Add index (JSON)'. The main area contains a JSON code editor with numbered lines from 1 to 17. Red boxes highlight the first three lines of the JSON object, specifically the 'odata.context' and 'odata.etag' fields. The JSON code is as follows:

```
1 {
2     "odata.context": "https://cog-flktsrgpt-uat-eastus-00
3     "odata.etag": "\"0x8DBE6CA7105B092\"",
4     "name": "flktsrgptindexnonuat-comprehension",
5     "defaultScoringProfile": null,
6     "fields": [
7         {
8             "name": "id",
9             "type": "Edm.String",
10            "searchable": false,
11            "filterable": false,
12            "retrievable": true,
13            "sortable": false,
14            "facetable": false,
15            "key": true,
16            "indexAnalyzer": null,
17            "searchAnalyzer": null
```

5. Remove the red highlighted parts of the JSON content from the editor. Please refer to the image above.
6. Once removed, it will enable save button at the bottom of the window. Click on Save.

### Add index (JSON)

```
1  {
2
3      "name": "flktsgptindex"
4      "defaultScoringProfile": null
5      "fields": [
6          {
7              "name": "id",
8              "type": "Edm.String"
9              "searchable": false
10             "filterable": false
11             "retrievable": true
12             "sortable": false,
13             "facetable": false
14             "key": true,
15             "indexAnalyzer": null
16             "searchAnalyzer": null
17             "analyzer": null,
18         }
19     ]
20 }
```

**Save**   **Cancel**

7. Verify the index created from the Index tab of the Cognitive Search Resource.

The screenshot shows the Azure Cognitive Search service interface for the search service 'cog-flktsgpt-uat-eastus-001'. The 'Indexes' tab is selected. A red box highlights the index name 'flktsgptindexnonuat-comprehension' in the list. The left sidebar includes links for Overview, Activity log, Access control (IAM), Tags, and Diagnose and solve problems. The 'Search management' section has 'Indexes' selected.

## Web Application

Deployment of the Web Application happened based on creation only, which registers a web app.

1. Open portal.azure.com, and go to App Services.

The screenshot shows the Microsoft Azure portal interface. At the top, there's a blue header bar with the 'Microsoft Azure' logo and a search bar containing 'Web A'. Below the header, the main content area has a sidebar on the left labeled 'Azure services' with a '+ Create a resource' button. The main panel is titled 'All' and shows a list of services. One item, 'App Services', is highlighted with a red box. Other items visible include 'Web Application Firewall policies (WAF)' and 'Function App'. The bottom of the sidebar also has a 'Resources' tab.

2. Click on Create-> Create Web App

The screenshot shows the 'App Services' blade within the Azure portal. At the top, it displays the subscription information 'Fortive (fortive.onmicrosoft.com)'. Below the title, there are several navigation links: '+ Create', 'Manage Deleted Apps', 'Manage view', and three more options whose labels are partially visible. The first option, '+ Web App', is highlighted with a red box. Below this, there are other options: '+ Static Web App' and '+ Web App + Database'.

3. Enter the following details.

- a. Select the subscription for the resource.
- b. Select the resource group.
- c. Enter the Web app name.
- d. Select the Docker container. (As a container is published on dev in the Azure Container Registry).

- e. Select Linux as operating system.
- f. Select same region as Resource Group's region.

[Home](#) > [App Services](#) >

## Create Web App

### Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ

1

Fluke Unified BI



Resource Group \* ⓘ

2

flk-techsupport-gpt-uat



[Create new](#)

### Instance Details

Name \* ⓘ

3

Web App name

.azurewebsites.net

Publish \*

4

Code  Docker Container  Static Web App

Operating System \*

5

Linux  Windows

Region \*

6

East US



[Review + create](#)

[< Previous](#)

[Next : Database >](#)

4. Under the Docker section enter the following details.

- a. Select Single Container.
- b. Select Azure Container Registry (Dev resource used ACR)
- c. Select the registry.
  - i. As Docker image is a machine from where web app is deployed, we have used the same image of the docker as on dev.
- d. Select the Dev docker image.
- e. Click on next.

## Create Web App

Basics

Database

Docker

Networking

Monitoring

Tags

Review + create

Pull container images from Azure Container Registry, Docker Hub or a private Docker repository. App Service will deploy the containerized app with your preferred dependencies to production in seconds.

Options

1

Single Container

▼

Image Source

2

Azure Container Registry

▼

Azure container registry options

3

Registry \*

registryflktsdev001

▼

Image \*

4

flksitegpt/backendapi

▼

Tag \*

5

latest\_dev\_1

▼

Startup Command ⓘ

- Under the Networking tab, Ensure that the public access is enabled.

## Create Web App

Basics

Database

Docker

Networking

Monitoring

Tags

Review + create

Web Apps can be provisioned with the inbound address being public to the internet or isolated to an Azure network. Web Apps can also be provisioned with outbound traffic able to reach endpoints in a virtual network governed by network security groups or affected by virtual network routes. By default, your app is open to the internet and cannot reach into a virtual network. These aspects can also be changed after the app is provisioned.

Enable public access \* ⓘ

On

Off

Enable network injection \*

On

Off

- Keep No on Enable Application Insights if you have a App Insight resource already created. Otherwise select yes, it will create a new resource on UAT.

## Create Web App ...

Basics Database Docker Networking **Monitoring** Tags Review + create

Azure Monitor application insights is an Application Performance Management (APM) service for developer DevOps professionals. Enable it below to automatically monitor your application. It will detect performance and includes powerful analytics tools to help you diagnose issues and to understand what users actually do app. Your bill is based on amount of data used by Application Insights and your data retention settings. [Learn more](#)

[App Insights pricing ↗](#)

### Application Insights

Enable Application Insights \*

No  Yes

7. Click on review + create.
8. Verify the Web Application under the App Service tab opened in step 1.

## Log Workspace Creation

*Export the Template + Deployment*

1. Open portal.azure.com and go to the Dev resource group, select the Deployments.

The screenshot shows the Azure Resource Groups blade. On the left, a list of resource groups is displayed, with 'flk-techsupport-gpt-dev' selected and highlighted by a red box. On the right, the details for 'flk-techsupport-gpt-dev' are shown, including an 'Overview' section and a 'Settings' section. In the 'Settings' section, the 'Deployments' link is also highlighted with a red box.

2. Select the deployment of the Log Workspace.

The screenshot shows the Azure Deployments blade for the 'flk-techsupport-gpt-dev' resource group. It lists several deployments, with 'newWorkspaceTemplate' selected and highlighted by a red box. The other deployments listed are 'Microsoft.ContainerRegistry', 'Microsoft.CognitiveServicesFormRecognizer-2023...', 'Failure-Anomalies-Alert-Rule-Deployment-a593b5...', and 'Microsoft.AppInsights'. All deployments are marked as 'Succeeded'.

Deployment	Status	Timestamp
Microsoft.ContainerRegistry	Succeeded	11/3/2023
Microsoft.CognitiveServicesFormRecognizer-2023...	Succeeded	11/3/2023
Failure-Anomalies-Alert-Rule-Deployment-a593b5...	Succeeded	11/3/2023
<b>newWorkspaceTemplate</b>	<b>Succeeded</b>	<b>11/3/2023</b>
Microsoft.AppInsights	Succeeded	11/3/2023

3. Click on Template to generate the deployed template.

The screenshot shows the Azure portal interface for a resource template named "newWorkspaceTemplate". The top navigation bar includes a search bar, download, add to library, and other options. Below the title, there are sections for Overview, Inputs, Outputs, and Template. The "Template" tab is currently selected and highlighted with a red box. A tooltip on the right provides information about automating deployment with scripts or code, with a link to learn more.

4. Click on the deploy above. It will open a Custom Deployment window.

The screenshot shows the same Azure portal interface for the "newWorkspaceTemplate". The "Deploy" button in the top right corner is highlighted with a red box. The "Template" tab is still selected. A tooltip on the right provides information about automating deployment with Azure Resource Manager templates and scripts, with a link to learn more. At the bottom, a partial view of the "Parameters" section is visible.

5. Click on Edit template.

The screenshot shows the 'Custom deployment' blade in the Azure portal. At the top, there's a banner with a purple icon and the text: 'New! Deployment Stacks let you manage the lifecycle of your deployments. Try it now →'. Below the banner, there are two tabs: 'Basics' (which is selected) and 'Review + create'. Under the 'Template' section, there's a 'Custom template' card with a blue icon, showing '1 resource'. To the right of the card are two buttons: 'Edit template' (with a pencil icon) and 'Visualize' (with a grid icon). A red box highlights the 'Edit template' button.

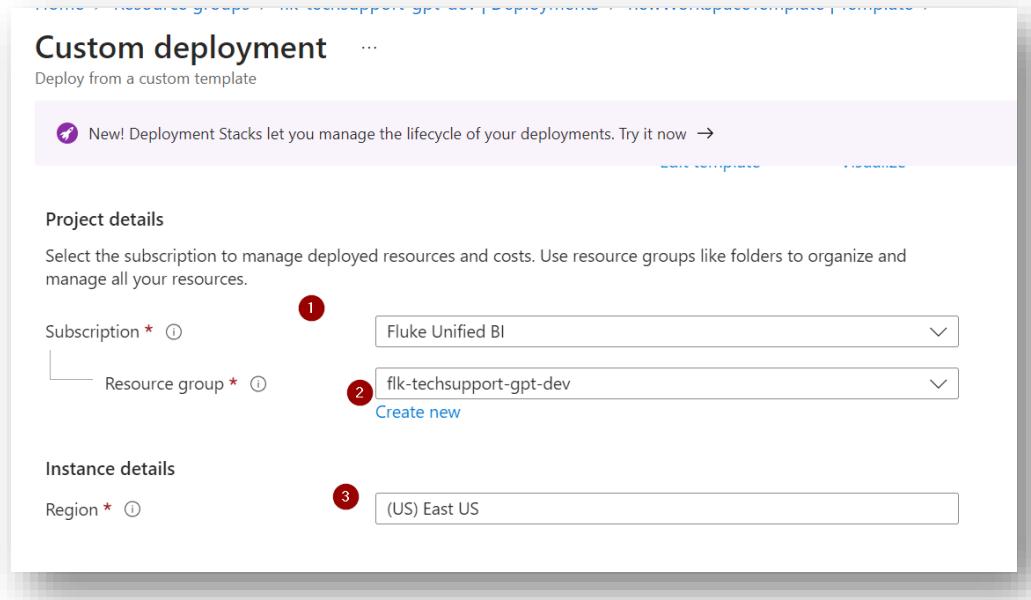
6. In the opened editor, update the resource name with UAT environment conventions and click on save.

The screenshot shows the 'Edit template' window with the title 'Edit template'. It displays an Azure Resource Manager template code in JSON format. The code defines a single resource of type 'Microsoft.OperationalInsights/workspaces'. A red box highlights the 'name' field in the resource definition, and a yellow callout box with the text 'Update name with UAT resource convention' has an arrow pointing to it.

```
1 {  
2     "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#"  
3     "contentVersion": "1.0.0.0",  
4     "parameters": {},  
5     "variables": {},  
6     "resources": [  
7         {  
8             "type": "Microsoft.OperationalInsights/workspaces",  
9             "apiVersion": "2020-08-01",  
10            "name": "52a1d076-bbbf-422a-9bf7-95d61247be4b-flk-techsupport-gpt--EUS",  
11            "location": "eastus",  
12            "properties": {}  
13        }  
14    ]  
15}
```

7. Update the following details in the opened window.

- a. Select the subscription.
- b. Select the resource group where the deployment is to be made.
- c. Select the region for the resource.



8. Click on next.
9. Click on Review + Create.

## Application Insights

### Export the Template

10. Open portal.azure.com and go to the Dev resource group.

- Select the Application Insights resource.

The screenshot shows the Azure Resource Groups blade. On the left, a list of resource groups is shown, with 'flk-techsupport-gpt-dev' selected and highlighted with a red box. On the right, the details for 'flk-techsupport-gpt-dev' are displayed, including a list of resources. One resource, 'appinsights-flktsgpt-dev-eastus-001', is also highlighted with a red box.

11. In the Application Insights resource, on the left menu, click on the Export Template option

The screenshot shows the Application Insights blade for 'appinsights-flktsgpt-dev-eastus-001'. The left sidebar shows various settings like Usage and estimated costs, API Access, Work Items, and Locks. At the bottom of the sidebar, the 'Export template' button is highlighted with a red box. The main area displays a template configuration screen with options for including parameters and generating the template.

12. Copy the JSON content of the resource and save it in .JSON file.

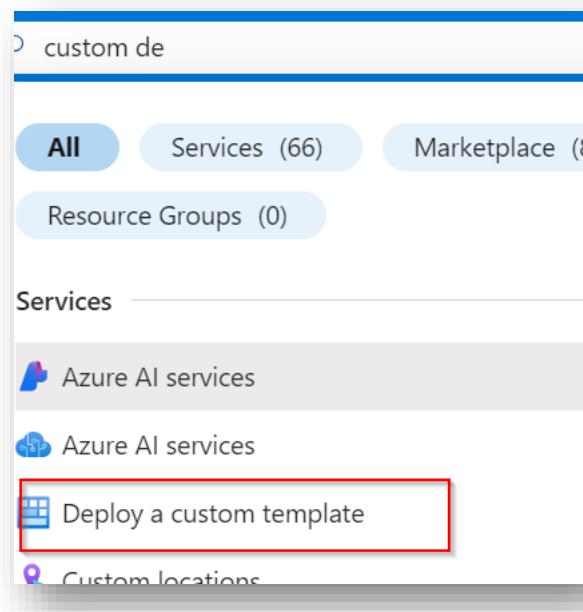
13. Open the .JSON file in file editor.

- Replace the parameter name having 'dev' to 'uat' across the file

- b. Replace all 'dev' occurrence to 'uat'
- c. Save the file.

#### *Deploy the Application Insights*

1. Open the portal.azure.com and search the Custom Deployment.



2. Now click on Edit and paste the saved template in the editor, and click on save.

A screenshot of the 'Custom deployment' blade in the Azure portal. The title 'Custom deployment' is at the top, followed by a subtitle 'Deploy from a custom template'. Below this, there are three tabs: 'Select a template' (which is underlined and active), 'Basics', and 'Review + create'. A descriptive text block says 'Automate deploying resources with Azure Resource Manager template select a template below to get started.' followed by a link 'Learn more about template deployment'. At the bottom of this section, a button labeled 'Build your own template in the editor' with a pencil icon is highlighted with a red rectangular box. Below this button, there is a section titled 'Common templates'.

```

19   "name": "[parameters('components_appinsights_flktsgpt_dev_eastus_001')]",
20   "location": "eastus",
21   "kind": "web",
22   "properties": [
23     "Application_Type": "web",
24     "Flow_Type": "Redfield",
25     "Request_Source": "IbizaAIExtension",
26     "RetentionInDays": 90

```

3. Now update the Parameter values in for the ARM template.
  - a. Update the name of the resource.
  - b. Copy the Resource ID of the Log Workspace created above.
    - i. Open the Log Workspace resource in portal.azure.com.

- ii. On the left menu for properties, select the ResourceID.

The screenshot shows the Azure portal interface for a Log Analytics workspace. The top navigation bar includes 'Home', 'Resource groups', 'flk-techsupport-gpt-uat', and the specific resource name '52a1d076-bbbf-422a-9bf7-95d61247be4b-flk-techsup-gptuat-EUS'. Below the navigation is a search bar and a sidebar with options like 'Search', 'Usage and estimated costs', 'Data export', 'Network isolation', 'Linked storage accounts', 'Properties' (which is selected and highlighted with a red box), 'Locks', 'Classic', and 'Legacy agents management'. The main content area displays the workspace's 'Properties' tab, showing fields for 'Workspace ID' (52a1d076-bbbf-422a-9bf7-95d61247be4b-flk-techsup-gptuat-EUS), 'Resource ID' (subscriptions/52a1d076-bbbf-422a-9bf7-95d61247be4b/resourceGroups/flk-techsupport-gpt-uat/providers/Microsoft.OperationalInsights工作spaces/52a...), 'Subscription Name' (Fluke Unified BI), and 'Subscription Id'.

Once the values are added, click on Review + Create.

The screenshot shows the 'Custom deployment' blade. At the top, it says 'Custom deployment' and 'Deploy from a custom template'. A purple banner at the top right says 'New! Deployment Stacks let you manage the lifecycle of your deployments. Try it now →'. Below this, there are dropdown menus for 'Subscription' (Fluke Unified BI) and 'Resource group' (flk-techsupport-gpt-uat, with a 'Create new' link). Under 'Instance details', there are dropdowns for 'Region' ((US) East US) and two other components: 'Components\_appinsights\_flktsgpt\_dev\_east' (appinsights-flktsgpt-dev-eastus-001) and 'Workspaces\_52a1d076\_bbbf\_422a\_9bf7\_95d61247be4b-flk-techsup-gptuat-EUS' (subscriptions/52a1d076-bbbf-422a-9bf7-95d61247be4b/resourceGroups/flk-techsupport-gpt-uat/providers/Microsoft.OperationalInsights工作spaces/52a...).

**Create all Application Insights resources similar way.**

## Enable Web App/Other Resources with Application Insights

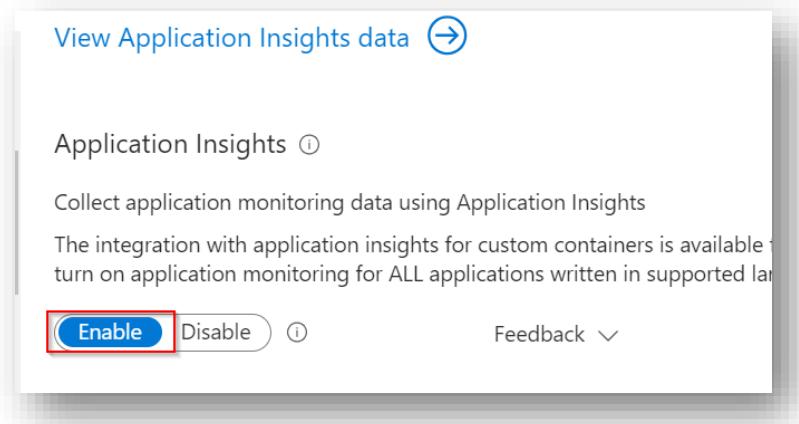
1. Go to portal.azure.com and open App Services.
  - a. Select the required Web App resource.

The screenshot shows the 'App Services' blade in the Azure portal. At the top, there's a breadcrumb navigation 'Home > App Services'. Below it, the title 'App Services' is followed by the resource name 'Fortive (fortive.onmicrosoft.com)'. A toolbar with buttons for 'Create', 'Manage Deleted Apps', 'Manage view', 'Refresh', and 'Export' is visible. There are also filters for 'Subscription equals all' and 'Resource group equals all'. The main area displays a table with columns: Name, Status, Location, and Pricing Tier. The table lists several web apps, with one specific entry highlighted by a red box: 'delphiapi-flkts-uat-eastus-001' which is 'Running' in 'East US' and belongs to the 'Basic' pricing tier. Other entries include 'chathistory-web' (Running, East US, Free), 'delphiapi-flkts-dev-eastus-001' (Running, East US, Premium V3), and 'Elk-And-S-TranslatorWeb' (Running, East US, Basic).

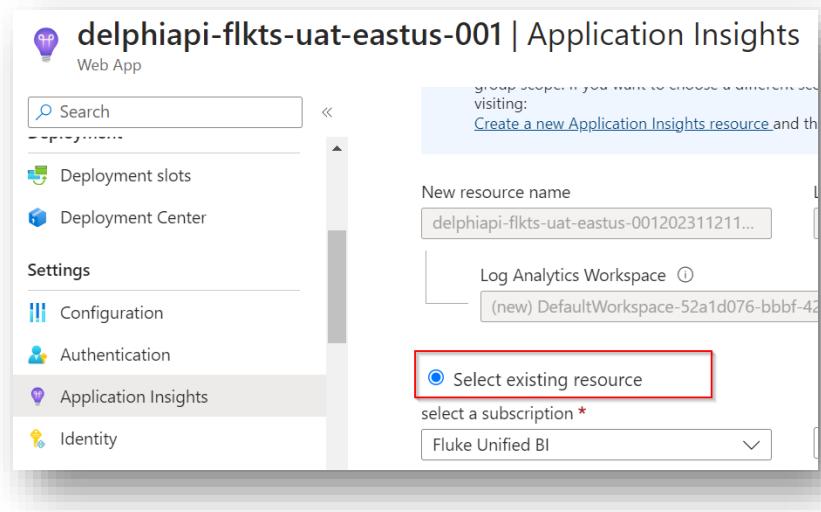
2. On the left menu, select the Application Insights.

The screenshot shows the 'delphiapi-flkts-uat-eastus-001' Web App blade. The top navigation bar includes a search bar, 'Browse', 'Stop', 'Swap', and a refresh icon. The left sidebar has sections for 'Deployment slots', 'Deployment Center', 'Settings' (with 'Configuration', 'Authentication', and 'Application Insights' listed, where 'Application Insights' is highlighted with a red box), and 'Identity'. The right panel is titled 'Essentials' and shows details: 'Resource group (move) flk-techsupport-gpt-uat', 'Status Running', 'Location (move) East US', and 'Subscription (move) Fluke Unified BI'.

3. Click on Enable Application Insights resource.



4. Click on the Select Existing Resource and select the relevant Application Insights resource.  
a. Click on Apply.



Name	Resource Group	Location
delphiapi-flkts-uat-eastus-001	flk-techsupport-gpt-uat	East US

## Event System Grid (Trigger for Function App)

1. Open portal.azure.com, and open resource group.
  - a. Click on the Deployments.

The screenshot shows the Azure Resource Group Overview page for the group 'flk-techsupport-gpt-dev'. The left sidebar includes links for Overview, Activity log, Access control (IAM), Tags, Resource visualizer, and Events. Below these are sections for Settings and Deployments. The 'Deployments' section is highlighted with a red box. On the right, there is a list of deployment items with checkboxes, one of which is checked.

Deployment	Status	Timestamp	Duration
Microsoft.AppInsights	Succeeded	11/3/2023, 12:20:50 PM	34 seconds, 366 milliseconds
eg-flksgpt-dev-eastus-001	Succeeded	11/3/2023, 11:08:33 AM	17 seconds, 326 milliseconds
search-service-cog-flksgpt-dev...	Succeeded	11/3/2023, 10:59:50 AM	5 seconds, 750 milliseconds
Microsoft.Web.FunctionApp-Port...	Succeeded	11/3/2023, 10:56:16 AM	25 seconds, 691 milliseconds

2. Locate the Event grid resource deployment and open that.

The screenshot shows the Azure Deployment details page for the deployment 'eg-flksgpt-dev-eastus-001'. The deployment status is 'Succeeded'. The page includes a toolbar with Refresh, Cancel, Redeploy, Delete, and View template buttons. The deployment details table shows the deployment name, status, timestamp, and duration for each item.

Item	Status	Timestamp	Duration
eg-flksgpt-dev-eastus-001	Succeeded	11/3/2023, 11:08:33 AM	17 seconds, 326 milliseconds
Microsoft.Web.FunctionApp-Port...	Succeeded	11/3/2023, 10:56:16 AM	25 seconds, 691 milliseconds

3. Click on Template to generate the template for the resource.

The screenshot shows the 'eg-flktsgpt-dev-eastus-001 | Overview' page in the Azure portal. The left sidebar has tabs for Overview, Inputs, Outputs, and Template, with 'Template' highlighted and surrounded by a red box. On the right, there's a summary section with a green checkmark and deployment details: Deployment name: eg-flktsgpt-dev-eastus-001, Subscription: Fluke Unified, Resource group: flk-techsupport-gpt-dev. Below this is a 'Deployment details' link.

4. Now click on the deploy. It will open one Custom Deployment window.

The screenshot shows the 'eg-flktsgpt-dev-eastus-001 | Template' page. The left sidebar has tabs for Overview, Inputs, Outputs, and Template, with 'Template' highlighted. At the top right, there are 'Download', 'Add to library', and 'Deploy' buttons, with 'Deploy' highlighted and surrounded by a red box. A tooltip above the Deploy button says: 'Automate deploying resources with Azure Resource Manager templates in a single, coordinated way with script or code. [Learn more about template deployment](#)'. The main area shows the template structure with sections for Parameters, Variables, and Resources. The 'Resources' section shows a single item: [parameters('name')] (Microsoft.EventGrid/systemTopics). The JSON code for the template is partially visible on the right.

5. Update the following values.  
a. Select the resource group.

- b. Select the region ( Same as resource group region)
- c. Provide proper name to the resource.
- d. Add location of the resource (if it is not loaded by default after selecting above three things)
- e. Add resource ID of the storage account.

Home > Resource groups > flk-techsupport-gpt-dev > stflktsgptdeveastus001 | Endpoints

Storage account

Search Refresh Give feedback

Provisioning state: Succeeded

Created: 11/3/2023, 10:43:56 AM

Last Failover: Never

Storage account resource ID: /subscriptions/52a1d076-bbbf-422a-9bf7-95d61247be4b/resourceGroups/flk-techsupport-gpt-dev/providers/Microsoft...

**Endpoints**

Blob service

Resource ID: /subscriptions/52a1d076-bbbf-422a-9bf7-95d61247be4b/resourceGroups/flk-techsupport-gpt-dev/providers/Microsoft...

Locks

- i. To obtain the Resource ID, go to the storage account
- ii. Click on the End Points on the left window.
- iii. Copy the resource ID.

Home > Resource groups > flk-techsupport-gpt-dev | Deployments > eg-flktsgpt-dev-eastus-001 | Template

## Custom deployment

Deploy from a custom template

New! Deployment Stacks let you manage the lifecycle of your deployments. Try it now →

Resource group \* ① flk-techsupport-gpt-dev Create new

Instance details

Region \* ② (US) East US

Name \* ③ eg-flktsgpt-dev-eastus-001 ✓

Location \* ④ eastus ✓

Source \* ⑤ /subscriptions/52a1d076-bbbf-422a-9bf7-95d61247be4b/resourceGroups/flk-techsupport-gpt-dev/providers/Microsoft.Storage/storageAccounts/stflktsgptdeveastus001 ✓

Topic Type \* ⑥ Microsoft.Storage.StorageAccounts ✓

6. Keep the Identity as System Assigned & click on Review + Create.

Tags \*

Identity \*

[Previous](#) [Next](#) [Review + create](#)

### Add Event Subscription

1. Go to Event Grid Subscription resources under portal.azure.com
  - a. Select Event Subscription.

Home > Resource groups > flk-techsupport-gpt-uat > eg-flktsgpt-uat-eastus-001

**eg-flktsgpt-uat-eastus-001** Event Subscriptions

Event Grid System Topic

Search

+ Event Subscription Refresh

Search to find event subscription by name...

Name	Endpoint	Pref
eg-flktsgpt-uat-eastus-0...	AzureFunction	/blo

Load more

Access control (IAM)

Tags

Settings

- Identity
- Properties
- Locks

Entities

Event Subscriptions

2. Click on + Event Subscription.
3. Copy the same properties from Dev resource.
4. Click on Create.

## Document Intelligence: Form Recognizer

### Export the ARM Template

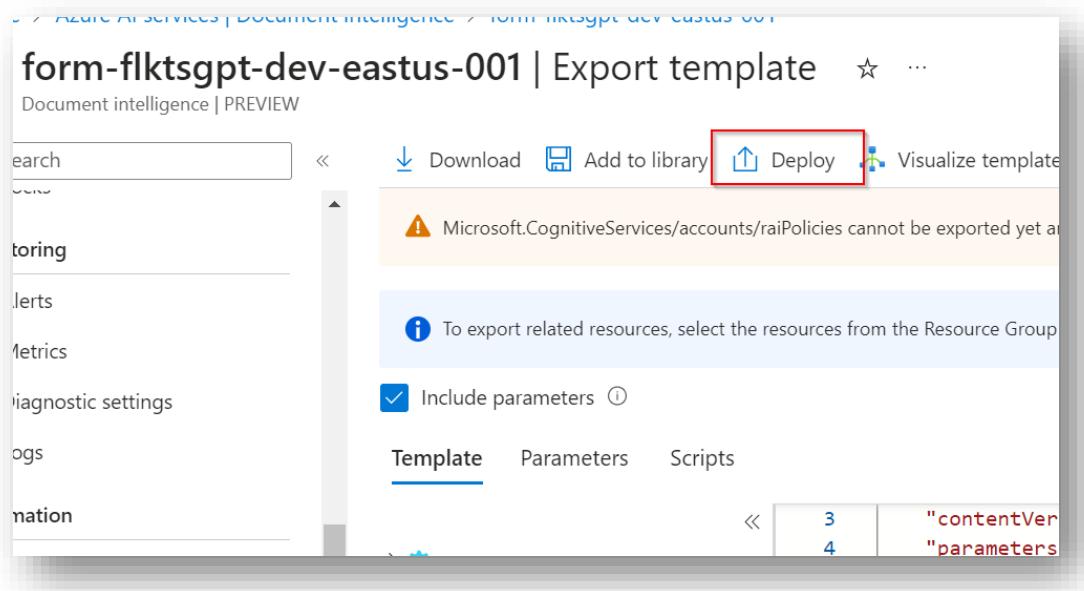
1. Go to portal.azure.com and open the dev resource for Form Recognizer.

The image contains two screenshots of the Microsoft Azure portal. The top screenshot shows the 'Resource groups' blade with a search bar and filter buttons for 'All', 'Services (7)', 'Marketplace (31)', and 'Document'. A red box highlights the 'Document intelligences' service. The bottom screenshot shows the 'Azure AI services | Document intelligence' blade, listing one resource named 'form-flktsgpt-dev-eastus-001'. This resource is highlighted with a red box. The left sidebar shows 'Azure AI services' with 'Azure AI services' selected.

2. On the left menu, click on the Export Template.

This screenshot shows the detailed view of the 'form-flktsgpt-dev-eastus-001' resource. The left sidebar includes sections for 'Monitoring' (Alerts, Metrics, Diagnostic settings, Logs) and 'Automation' (Tasks (preview)). The main pane displays resource details: Resource group (move), Status (Active), Location (East US), Subscription (move), Subscription ID (52a1d076-bbbf-422a-9bf7-95d61247be4b), and Tags (edit, Add tags). A red box highlights the 'Export template' button at the bottom of the left sidebar.

3. Click on the Deploy option. It will open one Custom Deployment window.



4. In the opened window, click on the Edit Template.

New! Deployment Stacks let you manage the lifecycle of your deployments. Try it now →

Basics    Review + create

Template

Custom template ↗  
1 resource

Edit template

Edit parameters

Visualize

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

5. In the editor pane, now update the parameters and resource name.

- Replace keyword 'dev' to 'uat' everywhere in the JSON.
- Click on Save.

**Edit template** ...

Edit your Azure Resource Manager template

Add resource Quickstart template Load file Download

```

1  {
2      "$schema": "https://schema.management.azure.com/schemas/201
3      "contentVersion": "1.0.0.0",
4      "parameters": {
5          "accounts_form_flktsgpt_dev_eastus_001_name": {
6              "defaultValue": "form-flktsgpt-dev-eastus-001",
7              "type": "String"
8          }
9      },
10     "variables": {},
11     "resources": [
12         {
13             "type": "Microsoft.CognitiveServices/accounts"
}

```

6. Update/verify the Parameters values in the deployment pane.
  - a. Select proper subscription value.
  - b. Select target resource group.
  - c. Select the region of the resource. (Should be same as Resource Group)
  - d. Verify the Name of the resource.

**Custom deployment** ...

Deploy from a custom template

New! Deployment Stacks let you manage the lifecycle of your deployments. Try it now →

**Project details**

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ① Fluke Unified BI

Resource group \* ② flk-techsupport-gpt-uat  
Create new

Instance details

Region \* ③ (US) East US

Accounts\_form\_flktsgpt\_dev\_eastus\_001 ④ ai form-flktsgpt-dev-eastus-001

Previous Next Review + create

7. Click on Review + Create.

## Appendix A - Structure of ARM Templates

ARM templates follows a JSON like structure, as given below.

```
{  
  "$schema": "https://schema.management.azure.com/schemas/2019-04-  
01/deploymentTemplate.json#",  
  "languageVersion": "",  
  "contentVersion": "",  
  "apiProfile": "",  
  "definitions": { },  
  "parameters": { },  
  "variables": { },  
  "functions": [ ],  
  "resources": [ ], /* or "resources": { } with languageVersion 2.0 */  
  "outputs": { }  
}
```

Refer the following tables for the details of each component

Element Name	Required	Description
\$schema	Yes	Location of the JavaScript Object Notation (JSON) schema file that describes the version of the template language. The version number you use depends on the scope of the deployment and your JSON editor.
languageVersion	No	Language version of the template. To view the enhancements of languageVersion 2.0.
contentVersion	Yes	Version of the template (such as 1.0.0.0). You can provide any value for this element. Use this value to document significant changes in your template. When deploying resources using the template, this value can be used to make sure that the right template is being used.
apiProfile	No	An API version that serves as a collection of API versions for resource types. Use this value to avoid having to specify API versions for each resource in the template. When you specify an API profile version and don't specify an API version for the resource type, Resource Manager uses the API version for that resource type that is defined in the profile
definitions	No	Schemas that are used to validate array and object values. Definitions are only supported in languageVersion 2.0.
parameters	No	Values that are provided when deployment is executed to customize resource deployment.

<a href="#">variables</a>	No	Values that are used as JSON fragments in the template to simplify template language expressions.
<a href="#">functions</a>	No	User-defined functions that are available within the template.
<a href="#">resources</a>	Yes	Resource types that are deployed or updated in a resource group or subscription.
<a href="#">outputs</a>	No	Values that are returned after deployment.

## Definition

In the definitions section of the template, specify the schemas used for validating array and object values. Definitions can only be used with [languageVersion 2.0](#).

```
"definitions": {
    "<definition-name>": {
        "type": "<data-type-of-definition>",
        "allowedValues": [ "<array-of-allowed-values>" ],
        "minValue": <minimum-value-for-int>,
        "maxValue": <maximum-value-for-int>,
        "minLength": <minimum-length-for-string-or-array>,
        "maxLength": <maximum-length-for-string-or-array>,
        "prefixItems": <schema-for-validating-array>,
        "items": <schema-for-validating-array-or-boolean>,
        "properties": <schema-for-validating-object>,
        "additionalProperties": <schema-for-validating-object-or-boolean>,
        "discriminator": <schema-to-apply>,
        "nullable": <boolean>,
        "metadata": {
            "description": "<description-of-the-type-definition>"
        }
    }
}
```

## Parameters

In the parameters section of the template, you specify which values you can input when deploying the resources. You're limited to [256 parameters](#) in a template. You can reduce the number of parameters by using objects that contain multiple properties.

The available properties for a parameter are:

```
"parameters": {
  "<parameter-name>": {
    "type" : "<type-of-parameter-value>",
    "defaultValue": "<default-value-of-parameter>",
    "allowedValues": [ "<array-of-allowed-values>" ],
    "minValue": <minimum-value-for-int>,
    "maxValue": <maximum-value-for-int>,
    "minLength": <minimum-length-for-string-or-array>,
    "maxLength": <maximum-length-for-string-or-array>,
    "prefixItems": <schema-for-validating-array>,
    "items": <schema-for-validating-array-or-boolean>,
    "properties": <schema-for-validating-object>,
    "additionalProperties": <schema-for-validating-object-or-boolean>,
    "discriminator": <schema-to-apply>,
    "nullable": <boolean>,
    "metadata": {
      "description": "<description-of-the parameter>"
    }
  }
}
```

## Variables

In the variables section, you construct values that can be used throughout your template. You don't need to define variables, but they often simplify your template by reducing complex expressions. The format of each variable matches one of the [data types](#). You're limited to [256 variables](#) in a template.

The following example shows the available options for defining a variable:

```
"variables": {
  "<variable-name>": "<variable-value>",
  "<variable-name>": {
    <variable-complex-type-value>
  },
  "<variable-object-name>": {
    "copy": [
      {
        "name": "<name-of-array-property>",
        "count": <number-of-iterations>,
        "input": <object-or-value-to-repeat>
      }
    ]
  },
  "copy": [
    {
      "name": "<variable-array-name>",
      "count": <number-of-iterations>,
      "input": <object-or-value-to-repeat>
    }
  ]
}
```

## Functions

Within your template, you can create your own functions. These functions are available for use in your template. Typically, you define complicated expressions that you don't want to repeat throughout your template. You create the user-defined functions from expressions and [functions](#) that are supported in templates.

When defining a user function, there are some restrictions:

- The function can't access variables.
- The function can only use parameters that are defined in the function. When you use the [parameters function](#) within a user-defined function, you're restricted to the parameters for that function.
- The function can't call other user-defined functions.
- The function can't use the [reference function](#).
- Parameters for the function can't have default values.

```
"functions": [
  {
    "namespace": "<namespace-for-functions>",
    "members": {
      "<function-name>": {
        "parameters": [
          {
            "name": "<parameter-name>",
            "type": "<type-of-parameter-value>"
          }
        ],
        "output": {
          "type": "<type-of-output-value>",
          "value": "<function-return-value>"
        }
      }
    }
  }
]
```

## **Resource Group Deletion Consideration**

1. Resources in the resource group specified during the deployment operation are subject to deletion if you deploy to multiple resource groups in a template.
2. Resources in secondary resource groups are not deleted when using complete mode.

## **Reference Links**

Microsoft documentation: [Link](#)