Microsoft

Manage an Azure Cosmos DB for NoSQL solution using DevOps practices

Agenda

- Write management scripts for Azure Cosmos DB for NoSQL
- Create resource template for Azure Cosmos DB for NoSQL

Write management scripts for Azure Cosmos DB for NoSQL

Create resources

In this learning path, we will use Azure CLI to manage Azure Cosmos DB for NoSQL accounts.

Azure Cosmos DB account group commands

```
az cosmosdb create \
  --name '<account-name>' \
  --resource-group '<resource-group>'
```

```
az cosmosdb create \
  --name '<account-name>' \
  --resource-group '<resource-group>' \
  --default-consistency-level 'eventual' \
  --enable-free-tier 'true'
```

```
az cosmosdb create \
  --name '<account-name>' \
  --resource-group '<resource-group>' \
  --locations regionName='eastus'
```

```
az cosmosdb --help
```

```
az cosmosdb create --help
```

Azure Cosmos DB for NoSQL subgroup commands

```
az cosmosdb sql database create \
  --account-name '<account-name>' \
  --resource-group '<resource-group>' \
  --name '<database-name>'
```

```
az cosmosdb sql container create \
  --account-name '<account-name>' \
  --resource-group '<resource-group>' \
  --database-name '<database-name>' \
  --name '<container-name>' \
  --throughput '400' \
  --partition-key-path '<partition-key-path-string>'
```

```
az cosmosdb sql --help
```

```
az cosmosdb sql database --help
```

```
az cosmosdb sql container --help
```

Manage index policies

When creating a container, you specify the indexing policy using CLI.

Let's assume that you have an indexing policy defined in a file named *policy.json*

```
{
   "indexingMode": "consistent",
   "automatic": true,
   "includedPaths": [ { "path": "/*" } ],
   "excludedPaths":
   [
        { "path": "/headquarters/*" },
        { "path": "/\"_etag\"/?" }
   ]
}
```

```
az cosmosdb sql container create \
  --account-name '<account-name>' \
  --resource-group '<resource-group>' \
  --database-name '<database-name>' \
  --name '<container-name>' \
  --partition-key-path '<partition-key-path-string>' \
  --idx '@.\policy.json' \
  --throughput '400'
```

Raw JSON string

```
az cosmosdb sql container create \
  --account-name '<account-name>' \
  --resource-group '<resource-group>' \
  --database-name '<database-name>' \
  --name '<container-name>' \
  --partition-key-path '<partition-key-path-string>' \
  --idx
'{\"indexingMode\":\"consistent\",\"automatic\":true,\"includedPaths\":[{\"path\":\"/*\"}],\"excludedPaths\":[{\"path\":\"/headquarters/*\"},{
\"path\":\"/\\\"_etag\\\"/?\"}]}' \
  --throughput '400'
```

Configure database or container-provisioned throughput

You can manage the provisioned throughput for both containers and databases using the CLI.

Update container throughput

```
az cosmosdb sql container throughput
update \
  --account-name '<account-name>’ \
  --resource-group '<resource-group>’ \
  --database-name '<database-name>’ \
  --name '<container-name>’ \
  --throughput '1000'
```

Update database throughput

```
az cosmosdb sql database throughput
update \
  --account-name '<account-name>’ \
  --resource-group '<resource-group>’ \
  --name '<database-name>’ \
  --throughput '4000'
```

Migrate between standard and autoscale throughput

Containers that manually provisioned throughput can be migrated to autoscale throughput.

Migrate throughput

```
az cosmosdb sql container throughput migrate \
  --account-name '<account-name>' \
  --resource-group '<resource-group>' \
  --database-name '<database-name>' \
  --name '<container-name>' \
  --throughput-type 'autoscale'
```

```
az cosmosdb sql container throughput migrate \
  --account-name '<account-name>' \
  --resource-group '<resource-group>' \
  --database-name '<database-name>' \
  --name '<container-name>' \
  --max-throughput '5000'
```

```
az cosmosdb sql container throughput migrate \
  --account-name '<account-name>' \
  --resource-group '<resource-group>' \
  --database-name '<database-name>' \
  --name '<container-name>' \
  --throughput-type 'manual'
```

View the min throughput of an autoscale container

```
az cosmosdb sql container throughput show \
  --account-name '<account-name>' \
  --resource-group '<resource-group>' \
  --database-name '<database-name>' \
  --name '<container-name>' \
  --query 'resource.minimumThroughput' \
  --output 'tsv'
```

Configure failovers and failover priorities

Let's assume that we have an Azure Cosmos DB account that we created in the *East US* region.

Add account regions

```
az cosmosdb update \
 --name '<account-name>' \
 --resource-group '<resource-group>' \
 --locations regionName='eastus' failoverPriority=0
isZoneRedundant=False \
 --locations regionName='westus2' failoverPriority=1
isZoneRedundant=False \
 --locations regionName='centralus' failoverPriority=2
isZoneRedundant=False
```

Remove account regions

```
az cosmosdb update \
  --name '<account-name>' \
  --resource-group '<resource-group>' \
  --locations regionName='eastus' failoverPriority=0
isZoneRedundant=False \
  --locations regionName='westus2' failoverPriority=1
isZoneRedundant=False
```

Enable automatic failover

```
az cosmosdb update \
  --name '<account-name>' \
  --resource-group '<resource-group>' \
  --enable-automatic-failover 'true'
```

Change failover priorities

```
az cosmosdb failover-priority-change \
  --name '<account-name>' \
  --resource-group '<resource-group>' \
  --failover-policies 'eastus=0' 'centralus=1' 'westus2=2'
```

Enable multi-region write

```
az cosmosdb update \
  --name '<account-name>' \
  --resource-group '<resource-group>' \
  --enable-multiple-write-locations 'true'
```

Initiate failovers

```
az cosmosdb failover-priority-change \
  --name '<account-name>' \
  --resource-group '<resource-group>' \
  --failover-policies 'westus2=0' 'eastus=1'
```

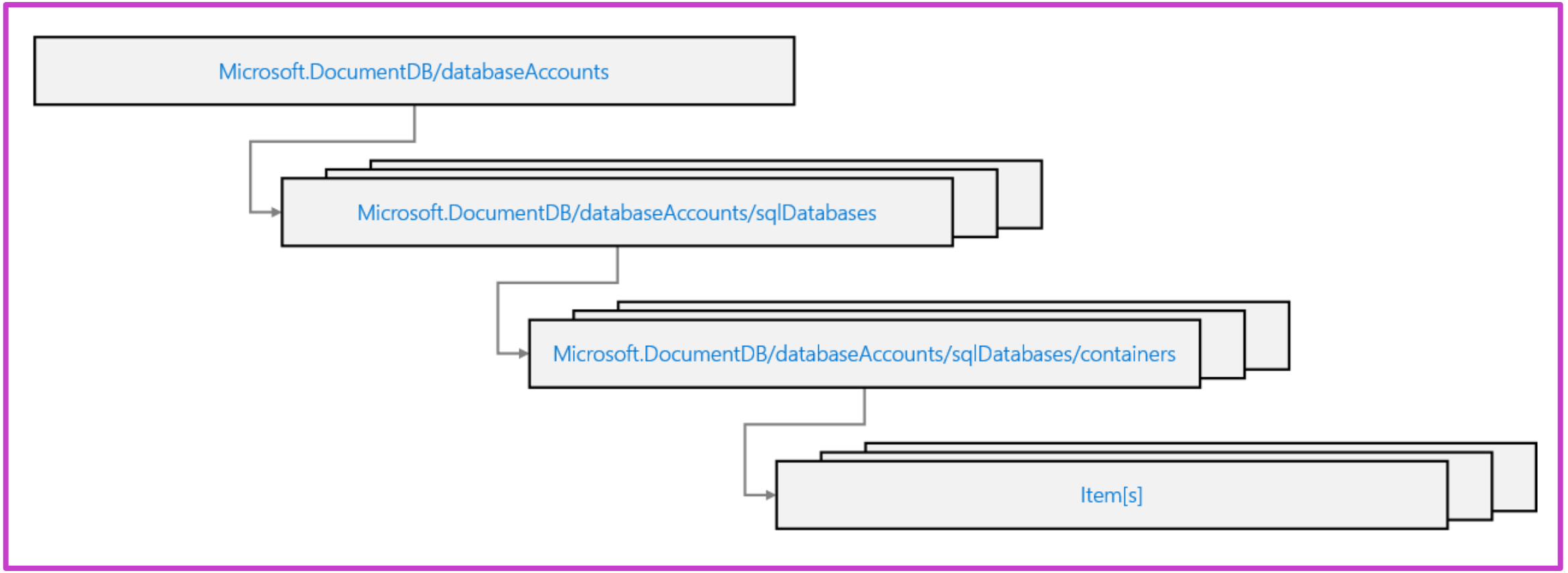
Lab – Adjust provisioned throughput using an Azure CLI script

- Log in to the Azure CLI
- Create Azure Cosmos DB account using the Azure CLI
- Create Azure Cosmos DB for NoSQL resources using the Azure CLI
- Adjust the throughput of an existing container using the Azure CLI

Create resource template for Azure Cosmos DB for NoSQL

Understand Azure Resource Manager resources

Each of the resources available for Azure Cosmos DB is listed under the *Microsoft.DocumentDB* resource provider.



Author Azure Resource Manager templates

There are three primary resources to define in a specific relationship order when authoring a template for an Azure Cosmos DB for NoSQL account.

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "resources": [
      {
         "type": "Microsoft.DocumentDB/databaseAccounts",
         "apiVersion": "2021-05-15",
         "name": "[concat('csmsarm', uniqueString(resourceGroup().id))]",
         "location": "[resourceGroup().location]",
         "properties": {
               "databaseAccountOfferType": "Standard",
               "locations": [ { "locationName": "westus" } ]
          }
       },
       {
           "type": "Microsoft.DocumentDB/databaseAccounts/sqlDatabases",
           "apiVersion": "2021-05-15",
           "name": "[concat('csmsarm', uniqueString(resourceGroup().id), '/cosmicworks')]",
           "dependsOn": [ "[resourceId('Microsoft.DocumentDB/databaseAccounts', concat('csmsarm', uniqueString(resourceGroup().id)))]" ],
           "properties": { "resource": { "id": "cosmicworks" } }
       },
       {
           "type": "Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers",
           "apiVersion": "2021-05-15",
           "name": "[concat('csmsarm', uniqueString(resourceGroup().id), '/cosmicworks/products')]",
           "dependsOn": [ "[resourceId('Microsoft.DocumentDB/databaseAccounts', concat('csmsarm', uniqueString(resourceGroup().id)))]",
                          "[resourceId('Microsoft.DocumentDB/databaseAccounts/sqlDatabases', concat('csmsarm', uniqueString(resourceGroup().id)), 'cosmicworks')]"
                        ],
           "properties": { "options": { "throughput": 400 }, "resource": { "id": "products", "partitionKey": { "paths": [ "/categoryId" ] } } }
       }
    ]
}
```

Configure database or container-resources in Bicep

Each template resource uses the same resource type and version between both Azure Resource Manager and Bicep templates.

```
resource Account 'Microsoft.DocumentDB/databaseAccounts@2021-05-15’ =
{
  name: 'csmsbicep${uniqueString(resourceGroup().id)}’
  location: resourceGroup().location
  properties: {
    databaseAccountOfferType: 'Standard’
    locations: [ { locationName: 'westus' } ]
  }
}

resource Database 'Microsoft.DocumentDB/databaseAccounts/sqlDatabases@2021-05-15’ =
{
  parent: Account name: 'cosmicworks’
  properties: {
    options: { }
    resource: { id: 'cosmicworks’ }
  }
}

resource Container 'Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers@2021-05-15’ =
{
  parent: Database name: 'customers’
  properties: {
    resource: {
      id: 'customers’
      partitionKey: { paths: [ '/regionId’ ] }
    }
  }
}
```

Deploy templates to a resource group

Now that the templates have been defined, use the Azure CLI to deploy either JSON or Bicep Azure Resource Manager templates.

Deploy Azure Resource Manager template to a resource group

```
az deployment group create \
  --resource-group '<resource-group>’ \
  --template-file '.\template.json'
```

```
az deployment group create \
--resource-group '<resource-group>’ \
--name '<deployment-name>’ \
--template-file '.\template.json'
```

```
az deployment group create \
  --resource-group '<resource-group>’ \
  --template-file '.\template.json’ \
  --parameters name='<value>'
```

```
az deployment group create \
  --resource-group '<resource-group>’ \
  --template-file '.\template.json’ \
  --parameters ‘@.\parameters.json'
```

Deploy Bicep template to a resource group

```
az deployment group create \
  --resource-group '<resource-group>’ \
  --template-file '.\template.bicep'
```

Manage index policies – JSON templates

Defining and deploying an indexing policy in JSON templates.

Defining an indexing policy in JSON templates

```
{
  "type": "Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers",
  "apiVersion": "2021-05-15",
  "name": "[concat('csmsarm', uniqueString(resourceGroup().id), '/cosmicworks/products')]",
  "dependsOn": [
                "[resourceId('Microsoft.DocumentDB/databaseAccounts', concat('csmsarm', uniqueString(resourceGroup().id)))]",
                "[resourceId('Microsoft.DocumentDB/databaseAccounts/sqlDatabases', concat('csmsarm', uniqueString(resourceGroup().id)), 'cosmicworks')]"
               ],
   "properties": {
                    "options": { "throughput": 400 },
                    "resource": {
                                 "id": "products",
                                 "partitionKey": { "paths": [ "/categoryId" ] },
                                 "indexingPolicy": {
                                                    "indexingMode": "consistent",
                                                    "automatic": true,
                                                    "includedPaths": [ { "path": "/price/*" } ],
                                                    "excludedPaths": [ { "path": "/*" } ]
                                  }
                    }
   }
}
```

Deploying an indexing policy in JSON templates

```
az deployment group create \
  --resource-group '<resource-group>' \
  --template-file '.\template.json' \
  --name 'jsontemplatedeploy'
```

Manage index policies – Bicep templates

Defining and deploying an indexing policy in Bicep templates.

Defining an indexing policy in Bicep templates

```
resource Container 'Microsoft.DocumentDB/databaseAccounts/sqlDatabases/containers@2021-05-15' = {
    parent: Database
    name: 'customers’
    properties: {
        resource: {
                    id: 'customers’
                    partitionKey: { paths: [ '/regionId' ] }
                    indexingPolicy: {
                                        indexingMode: 'consistent’
                                        automatic: true includedPaths: [ { path: '/address/*' } ]
                                        excludedPaths: [ { path: '/*' } ]
                                    }
                    }
    }
}
```

Deploying an indexing policy in Bicep templates

```
az deployment group create \
  --resource-group '<resource-group>’ \
  --template-file '.\template.bicep’ \
  --name 'biceptemplatedeploy'
```

Lab – Create an Azure Cosmos DB for NoSQL container using Azure Resource Manager templates

- Prepare your development environment
- Create Azure Cosmos DB for NoSQL resources using Azure Resource Manager templates
- Observe deployed Azure Cosmos DB resources
- Create Azure Cosmos DB for NoSQL resources using Bicep templates
- Observe Bicep template deployment results