**Exercise - Move database schema changes through Azure Pipelines**

At this point, the team's pipeline has four stages. The first stage produces the build artifact. The second stage deploys the *Space Game* web application to App Service in the **dev** environment. The third and fourth stages deploy the *Space Game* web application to the **test** and **staging** environments. We removed the triggers and approvals from the previous module to concentrate on just this section of the pipeline.

Here you follow Andy, Mara, and Tim as they modify the pipeline to add stages. The stages will script the database changes for the DBA. The stages will then apply those changes after the DBA approves the changes.

**Create the Azure App Service environments**

In the Create a multistage pipeline by using Azure Pipelines  module, you created one App Service instance that corresponds to each of the **dev**, **test**, and **staging** environments. There you worked through the creation process step by step. Here you use the Azure CLI through Azure Cloud Shell to create all of the App Service instances in one step.

**Bring up Cloud Shell through the Azure portal**

1. Go to the Azure portal  and sign in.
2. From the menu, select **Cloud Shell**. In the upper left, in the drop-down menu, select **Bash**.

Azure portal showing selecting the Cloud Shell icon from the menu.

**Note**

To persist any files that you create in Cloud Shell, you must have an Azure Storage resource. When you first open Cloud Shell, you're prompted to create a resource group, storage account, and Azure Files share. This is a one-time step. The assets are automatically attached for all future Cloud Shell sessions.

**Select an Azure region**

To make the commands easier to run, start by selecting a default region. After you specify the default region, later commands use that region unless you specify a different region.

To set your default region:

* Run az configure. Replace **<REGION>** with the name of the region that you chose for your database.

**Azure CLI**

az configure --defaults location=<REGION>

This example sets **westus2** as the default region:

**Azure CLI**

az configure --defaults location= location="South India"

**Create the App Service instances**

Here you create the App Service instances for the three stages that you'll deploy to: *Dev*, *Test*, and *Staging*.

To create the App Service instances, you:

* Generate a random number that makes your web app's domain name unique.
* Create an App Service plan.
* Create the App Service instances, one for each of the *dev*, *test*, and *staging* environments.
* Set an application configuration key/value pair to store the connection string in the database.
* Get the host name for each of your environments.
* Verify that each environment is running and that the home page is accessible.

1. In Cloud Shell, generate a random number that makes your web app's domain name unique:

**Bash**

**webappsuffix=$RANDOM**

1. Run the following az appservice plan create command to create an App Service plan that's named tailspin-space-game-asp.

**Azure CLI**

az appservice plan create \

--name tailspin-space-game-asp \

--resource-group tailspin-space-game-rg \

--sku S1

The --sku argument specifies the **B1** plan, which runs on the **Basic** tier.

**Important**

If the **B1** SKU is unavailable in your Azure subscription, **choose a different plan**, such as **S1** (**Standard**).

1. Run the following az webapp create commands to create the three App Service instances, one for each of the *dev*, *test*, and *staging* environments.

**Azure CLI**

az webapp create \

--name tailspin-space-game-web-dev-$webappsuffix \

--resource-group tailspin-space-game-rg \

--plan tailspin-space-game-asp

az webapp create \

--name tailspin-space-game-web-test-$webappsuffix \

--resource-group tailspin-space-game-rg \

--plan tailspin-space-game-asp

az webapp create \

--name tailspin-space-game-web-staging-$webappsuffix \

--resource-group tailspin-space-game-rg \

--plan tailspin-space-game-asp

1. Run the following az webapp list command to list the host name and state of each App Service instance.

**Azure CLI**

az webapp list \

--resource-group tailspin-space-game-rg \

--query "[].{hostName: defaultHostName, state: state}" \

--output table

Note the host name for each running service. You'll need these host names to set the connection string to the database later, when you verify your work. Here's an example:

**Output**

HostName State

------------------------------------------------------- -------

tailspin-space-game-web-dev-21017.azurewebsites.net Running

tailspin-space-game-web-test-21017.azurewebsites.net Running

tailspin-space-game-web-staging-21017.azurewebsites.net Running

**Configure the App Service to use the connection string**

1. In Cloud Shell, create a Bash variable that specifies your connection string. Replace {your\_connection\_string} with your connection string.

**Bash**

DB\_CONNECTION\_STRING="{your\_connection\_string}"

This step is similar to what you did in the previous part, where you ran the application locally.

Here's a complete example:

**Bash**

DB\_CONNECTION\_STRING="Server=tcp:tailspin-space-game-sql-333.database.windows.net,1433;Initial Catalog=tailspindatabase;Persist Security Info=False;User ID=azuresql;Password=MyPassword1234;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;"

1. Run the az webapp config connection-string set command three times to configure the connection string to the database in each App Service environment.

**Azure CLI**

az webapp config connection-string set \

--name tailspin-space-game-web-dev-$webappsuffix \

--resource-group tailspin-space-game-rg \

--settings "DefaultConnection=$DB\_CONNECTION\_STRING" \

--connection-string-type SQLAzure

az webapp config connection-string set \

--name tailspin-space-game-web-test-$webappsuffix \

--resource-group tailspin-space-game-rg \

--settings "DefaultConnection=$DB\_CONNECTION\_STRING" \

--connection-string-type SQLAzure

az webapp config connection-string set \

--name tailspin-space-game-web-staging-$webappsuffix \

--resource-group tailspin-space-game-rg \

--settings "DefaultConnection=$DB\_CONNECTION\_STRING" \

--connection-string-type SQLAzure

This command creates an application setting called **DefaultConnection**. The application uses this setting to connect to the database. The setting enables you to use the connection string without setting it in the *appsettings.json* file. So you avoid revealing your credentials in plain text in source control. Anyone who has read access to App Service can read the *appsettings.json* configuration file. You'll normally set permissions in App Service to restrict read access.

**Important**

The **Clean up your Azure DevOps environment** page in this module contains important cleanup steps. Cleaning up helps ensure that you're not charged for Azure resources after you complete this module. Be sure to perform the cleanup steps even if you don't complete this module.

**Create a service connection**

Here you create a service connection that enables Azure Pipelines to access your Azure subscription. Azure Pipelines uses this service connection to deploy the website to App Service. You created a similar service connection in the previous module.

**Important**

Use the same Microsoft account to sign in to both the Azure portal and Azure DevOps.

1. In Azure DevOps, go to your **Space Game - web - Database** project.
2. From the bottom corner of the page, select **Project settings**.
3. Under **Pipelines**, select **Service connections**.
4. Select **New service connection**, then choose **Azure Resource Manager**, then select **Next**.
5. Near the top of the page, **Service principal (automatic)**. Then select **Next**.
6. Fill in these fields:

|  |  |
| --- | --- |
|  | |
| Field | **Value** |
| Scope level | **Subscription** |
| Subscription | Your Azure subscription |
| Resource Group | **tailspin-space-game-rg** |
| Service connection name | *Resource Manager - Tailspin - Space Game* |

1. During the process, you might be prompted to sign in to your Microsoft account.
2. Ensure that **Grant access permission to all pipelines** is selected.
3. Select **Save**.

Azure DevOps performs a test connection to verify that it can connect to your Azure subscription. If Azure DevOps can't connect, you have the chance to sign in a second time.

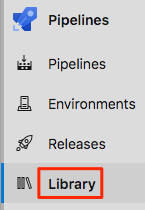
**Create pipeline variables in Azure Pipelines**

In the Create a release pipeline with Azure Pipelines  module, you added a variable to your pipeline to store the name of your web app in App Service. Here you do the same. However, this time you add one variable for each App Service instance that corresponds to the *Dev*, *Test*, or *Staging* stage in your pipeline.

You'll also create variables for the username and password for your Azure SQL instance. The pipeline will need these variables to make schema changes for you.

To add the variable:

1. In Azure DevOps, go to your **Space Game - web - Database** project.
2. Under **Pipelines**, select **Library**.



1. Select **+ Variable group**.
2. Under **Properties**, enter *Release* for the variable group name.
3. Under **Variables**, select **+ Add**.
4. Add the **WebAppNameDev**, **WebAppNameTest**, and **WebAppNameStaging** variables. Each value corresponds to the name of the App Service environment. This table shows some examples:

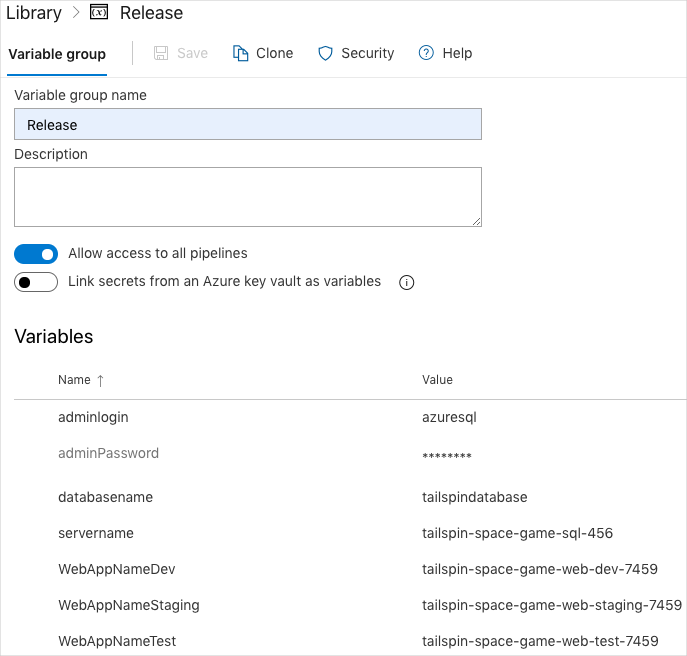
|  |  |
| --- | --- |
|  | |
| Variable name | **Example value** |
| WebAppNameDev | *tailspin-space-game-web-dev-1234* |
| WebAppNameTest | *tailspin-space-game-web-test-1234* |
| WebAppNameStaging | *tailspin-space-game-web-staging-1234* |

1. **Important**
2. Set the name of the App Service instance, not its host name. In this example, you would enter *tailspin-space-game-web-dev-1234* and not *tailspin-space-game-web-dev-1234.azurewebsites.net*.
3. Add these pipeline variables:

|  |  |
| --- | --- |
|  | |
| Variable name | **Example value** |
| adminlogin | *azuresql* |
| adminPassword | The password that you assigned when you created the SQL Server instance |
| servername | *tailspin-space-game-sql-333* (Your value should specify your server name.) |
| databasename | *tailspindatabase* |

1. Select the lock icon next to your value for **adminPassword**. This selection encrypts the value so that it doesn't appear when the pipeline runs.
2. Near the top of the page, select **Save** to save your variables to the pipeline.

Your variable group looks like this one:



**Add the database stage to the pipeline**

Here you add the Azure Pipelines stage that checks for SQL Database schema changes. The DBA will use the output to approve or edit proposed changes. In this exercise, the schema won't change. You'll change the schema in the next exercise.

1. Open the *azure-pipelines.yml* file from the database branch.
2. Copy the following new pipeline. Use it to replace the code in the *azure-pipelines.yml* file.

**yml**

trigger:

- '\*'

variables:

buildConfiguration: 'Release'

stages:

- stage: 'Build'

displayName: 'Build the web application'

jobs:

- job: 'Build'

displayName: 'Build job'

pool:

vmImage: 'ubuntu-18.04'

demands:

- npm

variables:

wwwrootDir: 'Tailspin.SpaceGame.Web/wwwroot'

dotnetSdkVersion: '3.1.300'

steps:

- task: UseDotNet@2

displayName: 'Use .NET Core SDK $(dotnetSdkVersion)'

inputs:

version: '$(dotnetSdkVersion)'

- task: Npm@1

displayName: 'Run npm install'

inputs:

verbose: false

- script: './node\_modules/.bin/node-sass $(wwwrootDir) --output $(wwwrootDir)'

displayName: 'Compile Sass assets'

- task: gulp@1

displayName: 'Run gulp tasks'

- script: 'echo "$(Build.DefinitionName), $(Build.BuildId), $(Build.BuildNumber)" > buildinfo.txt'

displayName: 'Write build info'

workingDirectory: $(wwwrootDir)

- task: DotNetCoreCLI@2

displayName: 'Restore project dependencies'

inputs:

command: 'restore'

projects: '\*\*/\*.csproj'

- task: DotNetCoreCLI@2

displayName: 'Build the project - $(buildConfiguration)'

inputs:

command: 'build'

arguments: '--no-restore --configuration $(buildConfiguration)'

projects: '\*\*/\*.csproj'

- task: DotNetCoreCLI@2

displayName: 'Publish the project - $(buildConfiguration)'

inputs:

command: 'publish'

projects: '\*\*/\*.csproj'

publishWebProjects: false

arguments: '--no-build --configuration $(buildConfiguration) --output $(Build.ArtifactStagingDirectory)/$(buildConfiguration)'

zipAfterPublish: true

- publish: '$(Build.ArtifactStagingDirectory)'

artifact: drop

- job: BuildDacpac

pool:

vmImage: 'windows-2019'

steps:

- task: DotNetCoreCLI@2

displayName: 'Restore project dependencies'

inputs:

command: 'restore'

projects: '\*\*/\*.csproj'

- task: VSBuild@1

displayName: 'Build the database project'

inputs:

project: '\*\*/\*.sqlproj'

- task: CopyFiles@2

displayName: 'Copy dacpac file to staging directory'

inputs:

contents: |

Tailspin.SpaceGame.Database/bin/\*\*/\*.dacpac

targetFolder: '$(Build.StagingDirectory)'

- task: PublishBuildArtifacts@1

displayName: 'Publish Artifact'

inputs:

pathToPublish: '$(Build.ArtifactStagingDirectory)'

artifactName: dropDacpac

condition: succeededOrFailed()

- stage: DBAVerificationScript

displayName: 'Script database schema changes'

dependsOn: Build

jobs:

- deployment: DBAVerificationScript

pool:

vmImage: 'windows-2019'

variables:

- group: 'Release'

environment: 'dbaverificationscript'

strategy:

runOnce:

deploy:

steps:

- download: current

artifact: dropDacpac

patterns: '\*\*/\*'

- task: SqlAzureDacpacDeployment@1

displayName: Generate schema change script

inputs:

azureSubscription: 'Resource Manager - Tailspin - Space Game'

authenticationType: 'server'

serverName: '$(servername).database.windows.net'

databaseName: '$(databasename)'

sqlUsername: '$(adminlogin)'

sqlPassword: '$(adminPassword)'

deployType: 'DacpacTask'

deploymentAction: 'Script'

dacpacFile: '$(Pipeline.Workspace)/dropDacpac/Tailspin.SpaceGame.Database/bin/Debug/Tailspin.SpaceGame.Database.dacpac'

ipDetectionMethod: 'AutoDetect'

- task: PowerShell@2

displayName: Show Auto Generated SQL Script

inputs:

targetType: 'inline'

script: |

Write-Host "Auto Generated SQL Update Script:"

Get-Content d:\a\1\s\GeneratedOutputFiles\$(databasename)\_Script.sql | foreach {Write-Output $\_}

- stage: DBAVerificationApply

displayName: 'Apply database schema changes'

dependsOn: DBAVerificationScript

jobs:

- deployment: DBAVerificationApply

pool:

vmImage: 'windows-2019'

variables:

- group: 'Release'

environment: 'dbaverificationapply'

strategy:

runOnce:

deploy:

steps:

- download: current

artifact: dropDacpac

patterns: '\*\*/\*'

- task: SqlAzureDacpacDeployment@1

displayName: 'Deploy SQL schema'

inputs:

azureSubscription: 'Resource Manager - Tailspin - Space Game'

authenticationType: 'server'

serverName: '$(servername).database.windows.net'

databaseName: '$(databasename)'

sqlUsername: '$(adminlogin)'

sqlPassword: '$(adminPassword)'

deployType: 'DacpacTask'

deploymentAction: 'Publish'

dacpacFile: '$(Pipeline.Workspace)/dropDacpac/Tailspin.SpaceGame.Database/bin/Debug/Tailspin.SpaceGame.Database.dacpac'

ipDetectionMethod: 'AutoDetect'

- stage: 'Dev'

displayName: 'Deploy to the dev environment'

dependsOn: DBAVerificationApply

jobs:

- deployment: Deploy

pool:

vmImage: 'ubuntu-18.04'

environment: dev

variables:

- group: Release

strategy:

runOnce:

deploy:

steps:

- download: current

artifact: drop

- task: AzureWebApp@1

displayName: 'Azure App Service Deploy: website'

inputs:

azureSubscription: 'Resource Manager - Tailspin - Space Game'

appName: '$(WebAppNameDev)'

package: '$(Pipeline.Workspace)/drop/$(buildConfiguration)/\*.zip'

- stage: 'Test'

displayName: 'Deploy to the test environment'

dependsOn: Dev

jobs:

- deployment: Deploy

pool:

vmImage: 'ubuntu-18.04'

environment: test

variables:

- group: 'Release'

strategy:

runOnce:

deploy:

steps:

- download: current

artifact: drop

- task: AzureWebApp@1

displayName: 'Azure App Service Deploy: website'

inputs:

azureSubscription: 'Resource Manager - Tailspin - Space Game'

appName: '$(WebAppNameTest)'

package: '$(Pipeline.Workspace)/drop/$(buildConfiguration)/\*.zip'

- stage: 'Staging'

displayName: 'Deploy to the staging environment'

dependsOn: Test

jobs:

- deployment: Deploy

pool:

vmImage: 'ubuntu-18.04'

environment: staging

variables:

- group: 'Release'

strategy:

runOnce:

deploy:

steps:

- download: current

artifact: drop

- task: AzureWebApp@1

displayName: 'Azure App Service Deploy: website'

inputs:

azureSubscription: 'Resource Manager - Tailspin - Space Game'

appName: '$(WebAppNameStaging)'

package: '$(Pipeline.Workspace)/drop/$(buildConfiguration)/\*.zip'

This pipeline adds a new build job for the *Tailspin.SpaceGame.Database* project. The project will create a *dacpac* file that contains information about the database schema. That *dacpac* file will be copied to a staging directory in the pipeline. Then the file will be published as an artifact called **dropDacpac**.

**yml**

- task: VSBuild@1

displayName: 'Build the database project'

inputs:

project: Tailspin.SpaceGame.Database/Tailspin.SpaceGame.Database.sqlproj

- task: CopyFiles@2

displayName: 'Copy dacpac file to staging directory'

inputs:

contents: |

Tailspin.SpaceGame.Database/bin/\*\*/\*.dacpac

targetFolder: '$(Build.StagingDirectory)'

- task: PublishBuildArtifacts@1

displayName: 'Publish Artifact'

inputs:

pathToPublish: '$(Build.ArtifactStagingDirectory)'

artifactName: dropDacpac

condition: succeededOrFailed()

The new stages are DBAVerificationScript and DBAVerificationApply.

The DBAVerificationScript stage will read the *dacpac* file and create a change script for the database. Then you use a PowerShell script to write out that change script. By using the script output, the database administrator can approve the changes before the changes are applied to the database.

**yml**

- task: SqlAzureDacpacDeployment@1

displayName: Generate schema change script

inputs:

azureSubscription: 'Resource Manager - Tailspin - Space Game'

authenticationType: 'server'

serverName: '$(servername).database.windows.net'

databaseName: '$(databasename)'

sqlUsername: '$(adminlogin)'

sqlPassword: '$(adminPassword)'

deployType: 'DacpacTask'

deploymentAction: 'Script'

dacpacFile: '$(Pipeline.Workspace)/dropDacpac/Tailspin.SpaceGame.Database/bin/$(buildConfiguration)/Tailspin.SpaceGame.Database.dacpac'

ipDetectionMethod: 'AutoDetect'

- task: PowerShell@2

displayName: Show Auto Generated SQL Script

inputs:

targetType: 'inline'

script: |

Write-Host "Auto Generated SQL Update Script:"

Get-Content d:\a\1\s\GeneratedOutputFiles\$(databasename)\_Script.sql | foreach {Write-Output $\_}

The DBAVerificationApply stage reads the auto-generated file and applies the change script to the database.

**yml**

steps:

- download: current

artifact: dropDacpac

patterns: '\*\*/\*'

- task: SqlAzureDacpacDeployment@1

displayName: 'Deploy SQL schema'

inputs:

azureSubscription: 'Resource Manager - Tailspin - Space Game'

authenticationType: 'server'

serverName: '$(servername).database.windows.net'

databaseName: '$(databasename)'

sqlUsername: '$(adminlogin)'

sqlPassword: '$(adminPassword)'

deployType: 'DacpacTask'

deploymentAction: 'Publish'

dacpacFile: '$(Pipeline.Workspace)/dropDacpac/Tailspin.SpaceGame.Database/bin/$(buildConfiguration)/Tailspin.SpaceGame.Database.dacpac'

ipDetectionMethod: 'AutoDetect'

1. Move to your project's root directory. Here's an example:

**Bash**

**cd ~/mslearn-tailspin-spacegame-web-deploy**

1. Save the file and commit it. But don't push your branch to GitHub yet.

**Bash**

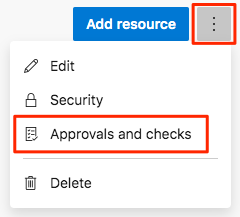
**git add azure-pipelines.yml**

**git commit -m "Add database schema checks to the pipeline"**

**Create an environment for manual approval**

Here you create the manual approval for the DBAVerificationApply stage. You learned about manual approvals in the previous module. Recall that you need to set up an environment and add an approver.

1. From Azure Pipelines, select **Environments**.
2. Select **Create environment**.
3. Under **Name**, enter *dbaverificationapply*.
4. Leave the remaining fields at their default values.
5. Select **Create**.
6. On the **dbaverificationapply** environment page, open the drop-down menu and then select **Approvals and checks**.



1. Select **Approvals**.
2. Under **Approvers**, select **Add users and groups** and then select your account.
3. Under **Instructions to approvers**, enter *Approve this change to the database schema*.
4. Select **Create**.

**Run the pipeline**

1. Push the pipeline changes up to your GitHub remote, origin. This step triggers the pipeline to run.

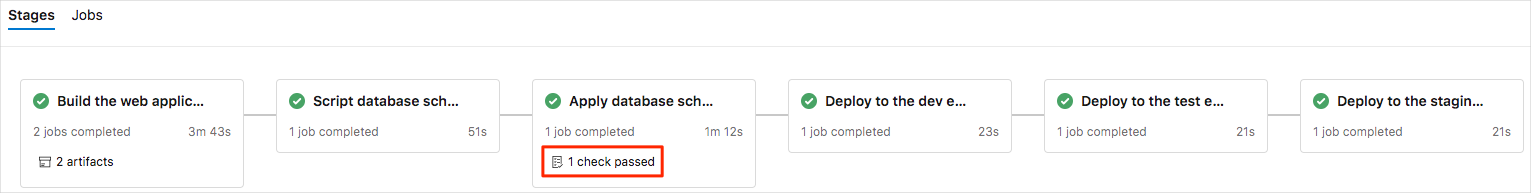
**Bash**

**git push origin database**

1. Go to your pipeline and wait for the manual approval of the database schema.

When the pipeline stops for approval, select the DBAVerificationScript stage. Look at the change script that was created. It's in the **Show automated SQL Script** section. The script has no changes because we didn't change anything in the database yet. You know there are no changes because you don't see **CREATE**, **ALTER**, or **DROP** statements in the script.

1. Go back to the pipeline and select the **Waiting** button on DBAVerificationApply. Select **Review** > **Approve**.
2. Wait for the pipeline to finish deployments.



1. Check at least one web address to see that the application is deployed and is working with the database.

**Tim:** Great! I feel good about going to the DBA with this setup so far. Let's try it with the schema change we've been working on.