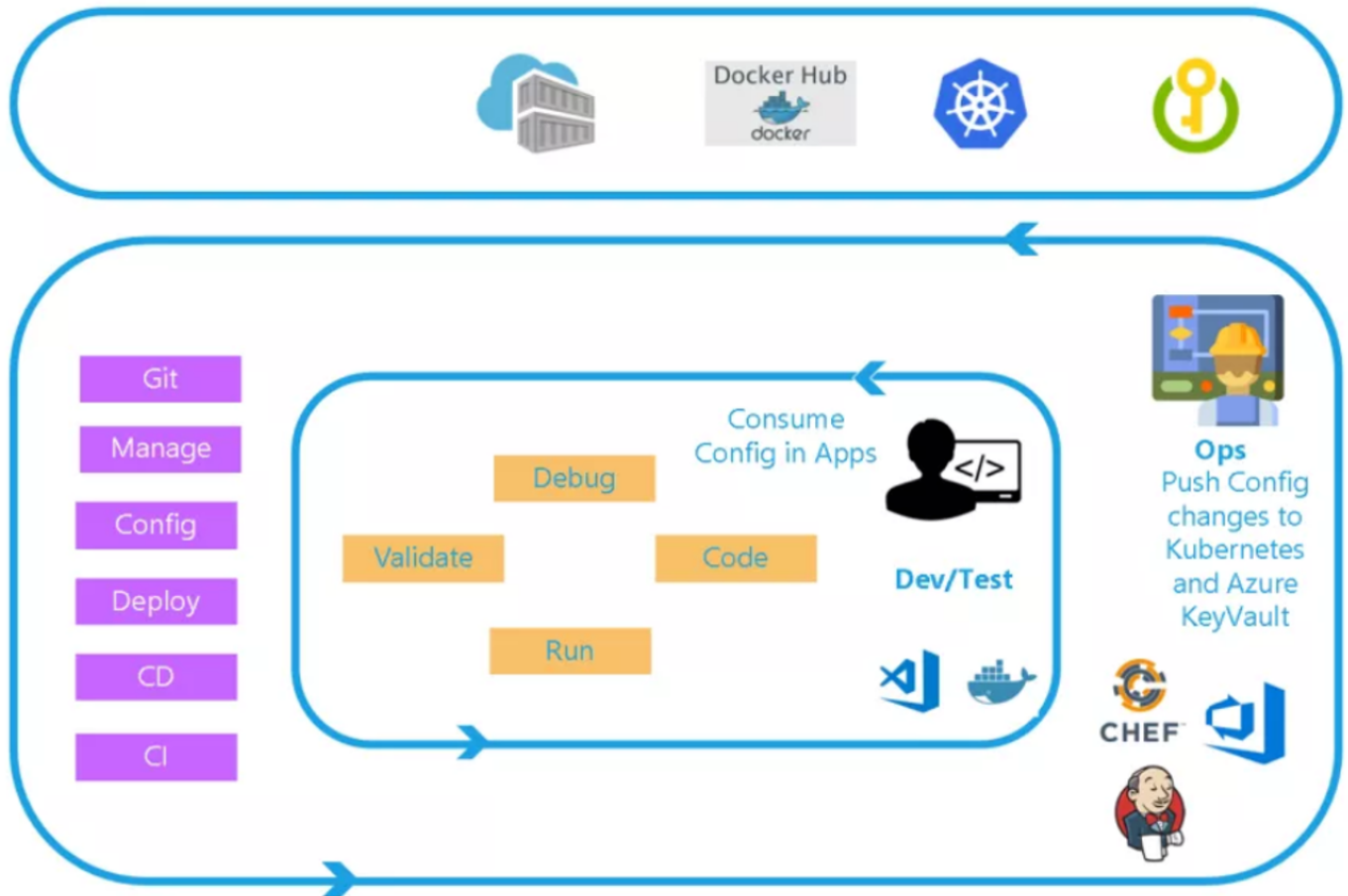


# Access Azure Key Vault secrets in the Azure DevOps Release Pipelines



Managing secrets in the application is crucial part of the whole development process. Please look at the picture. There are two loops:



- Inner - Focused on the developer teams iterating over their solution development (they consume the configuration published by the outer loop)
- Outer - The Ops Engineer govern the Configuration management and push changes (including Azure KeyVault secrets management)

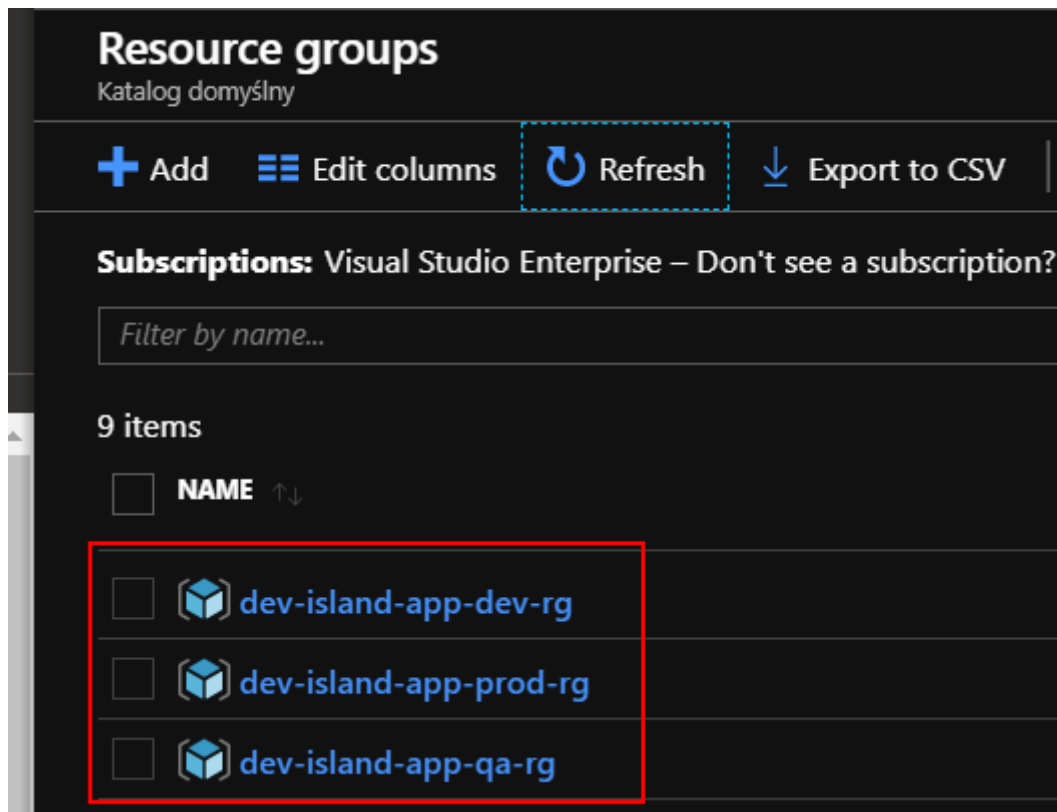
With such approach you are able keep clear separation of concerns and clean code. What is more, application configuration is much easier to maintain.

In this article I would like to present how integrate Azure Key Vault with Azure DevOps Release pipelines and how to inject secrets per specific environment (in this case Development, QA and Production).

## Prerequisites




Before we start I assume that there are already three separate resource groups created for each environment:

- dev-island-app-dev-rg: for Development
- dev-island-app-qa-rg: for QA
- dev-island-app-prod-rg: Production



Of course there resource groups creation can be also automated. But for this article I created them manually. In each of above resource group I created Key Vault:

- dev-island-app-dev-kv
- dev-island-app-qa-kv
- dev-island-app-prod-kv

<input type="checkbox"/>		dev-island-app-dev-kv	dev-island-app-dev-rg
<input type="checkbox"/>		dev-island-app-prod-kv	dev-island-app-prod-rg
<input type="checkbox"/>		dev-island-app-qa-kv	dev-island-app-qa-rg




I created separate Key Vault per environment because it is recommended approach by Microsoft:


"Our recommendation is to use a vault per application per environment (Development, Pre-Production and Production)."

You can read more [here](https://docs.microsoft.com/bs-latn-ba/Azure/key-vault/key-vault-best-practices#use-separate-key-vault). (<https://docs.microsoft.com/bs-latn-ba/Azure/key-vault/key-vault-best-practices#use-separate-key-vault>).

In each Key Vault I created one secret with different values for each environment:

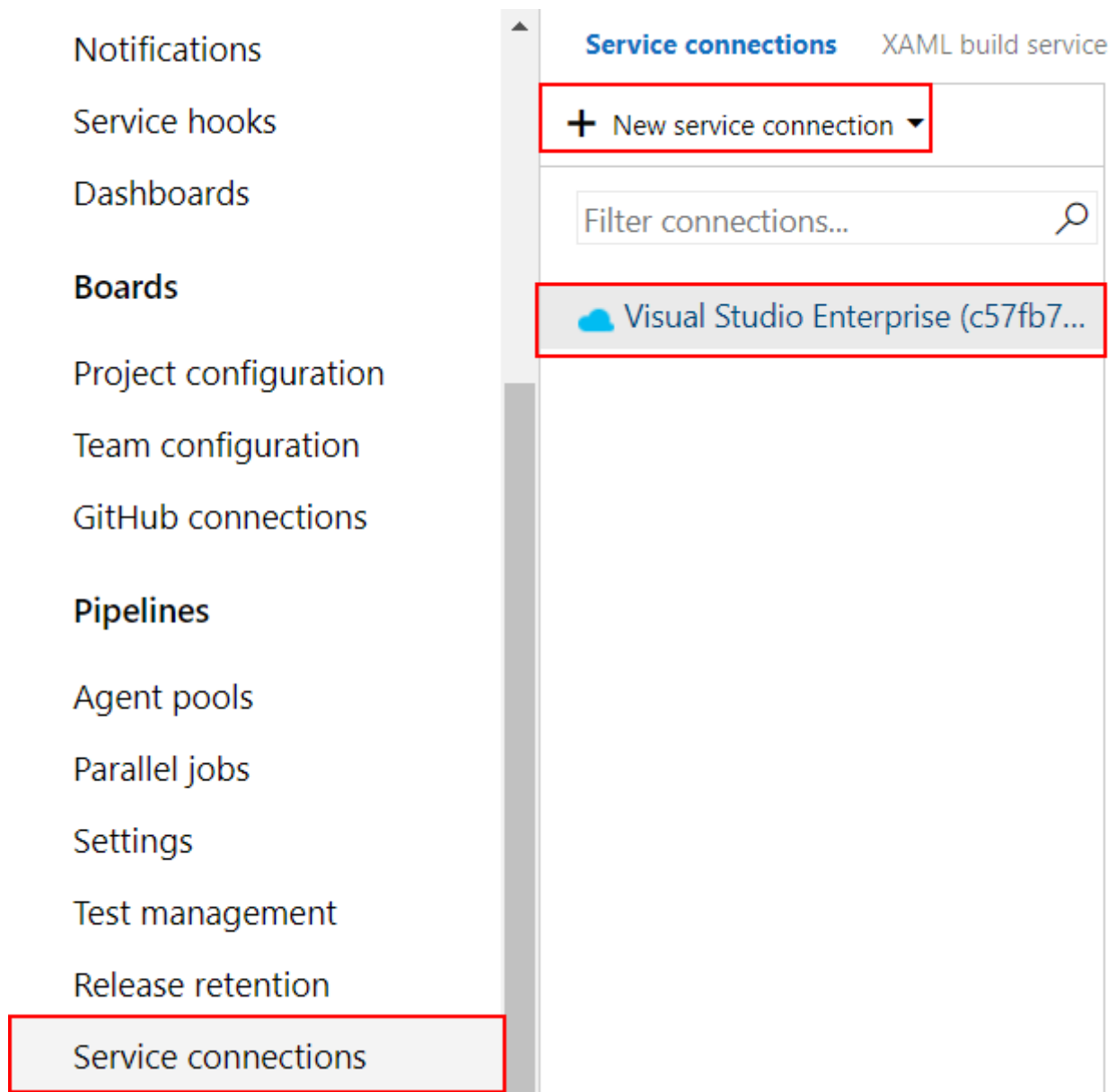
- Name: "DbConnectionString"
- Value for the Development environment: "Server=(localdb)\mssqllocaldb;Database=CarsIsland;Trusted\_Connection=True;"
- Value for the QA environment: "Server=(localdb)\mssqllocaldb;Database=CarsIslandQA;Trusted\_Connection=True;"
- Value for the Production environment: "Server=(localdb)\mssqllocaldb;Database=CarsIslandProd;Trusted\_Connection=True;"

 Generate/Import  Refresh  Restore Backup

 The secret 'DbConnectionString' has been successfully created.

NAME	TYPE	STATUS
DbConnectionString		✓ Enabled

**Add new service connection so you can access Azure resources from the Azure DevOps**



## Prepare release pipeline with Development, QA and Production stages

First of all we have to prepare release pipeline for all three environments: Development, QA and Production. Follo below steps:




# No release pipelines found

Automate your release process in a few easy steps with a new pipeline

New pipeline

## Select a template

Or start with an  Empty job

All pipelines > New release pipeline

Save Create release ...

Pipeline Tasks Variables Retention Options History

**Artifacts** | + Add

**Stages** | + Add

+ Add an artifact

Schedule not set

Development

1 job, 0 task

**Stage** Development

Properties

Name and owners of the stage

Stage name

Development

Stage owner

Daniel Krzyczkowski



Image not found



## Stage

QA

Delete Move

### Properties ^

Name and owners of the stage

Stage name

QA

Stage owner



Daniel Krzyczkowski



## Stage

Production

Delete Move

### Properties ^

Name and owners of the stage

Stage name

Production

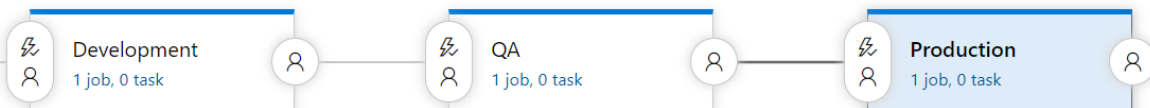
Stage owner



Daniel Krzyczkowski



Stages | Add ^

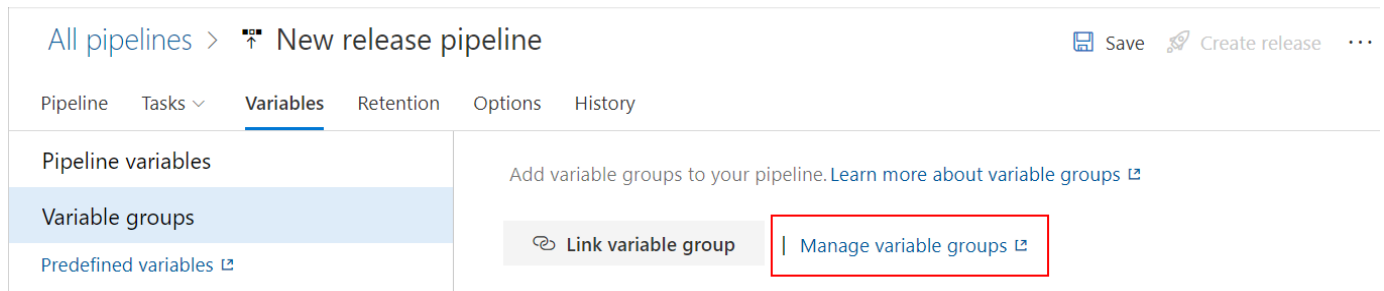


# Setup Azure Key Vault integration in the Release pipeline

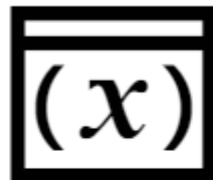
First of all we have to integrate Key Vault in the Release pipeline so secrets are available through variable group. Each stage in the release pipeline has its own variable group. Lets see how to do it.

## Setup variable group for the Development environment

Select "Manage variable groups":



Click "+ Variable group":



## New variable group

Create groups of variables that you can share across multiple pipelines.

[+ Variable group](#)

[Learn more about variable groups.](#)

Provide details about this specific variable group:

You have to authorize Azure DevOps to access Azure subscription and Key Vault:



Library &gt; ⓘ dev-island-app-dev-kv-vg\*

Variable group



Save



Clone



Security



Help

## Properties

Variable group name

dev-island-app-dev-kv-vg

Description

Variable group with Key Vault secrets for the Development environment.



Allow access to all pipelines



Link secrets from an Azure key vault as variables ⓘ

Azure subscription \* | Manage [🔗](#)


Visual Studio Enterprise (c






-0b35-4



Authorize



Library >  dev-island-app-dev-kv-vg


 Variable group |  Save |  Clone |  Security |  Help


## Properties



Variable group name


Description


☒ Allow access to all pipelines


☒ Link secrets from an Azure key vault as variables 

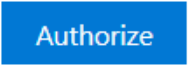

Azure subscription \* | [Manage](#) 


 

 Scoped to subscription 'Visual Studio Enterprise'

Key vault name \* [Manage](#) 




 The specified Azure service connection needs to have "Get, List" secret management permissions on the selected key vault. Click "Authorize" to enable Azure Pipelines to set these permissions or manage secret permissions in the Azure portal.

Now select which secrets you would like to use as variable in the release pipeline:

## Variables

Delete	Secret name	Content type	Status	Expiration date
<div></div>				

In our case we have to select the secret created before called "DbConnectionString":



## Choose secrets

Choose secrets to be included in this variable group

Sel...	Secret name	Content type	Status	Expiration date
<input checked="" type="checkbox"/>	DbConnectionString		Enabled	Never

**Ok**

Cancel

Once you select the secret, click "Save" button:

Library > ⓘ dev-island-app-dev-kv-vg\*

Variable group

**Save**

Clone



Security



Help

Now get back to the "variable groups" tab and click "Link variable group":

All pipelines > 🚀 New release pipeline

Pipeline Tasks ▾ **Variables** Retention Options History

Pipeline variables

Variable groups

Predefined variables [🔗](#)

Add variable groups to your pipeline. [Learn more about variable groups](#) [🔗](#)

**Link variable group**| [Manage variable groups](#) [🔗](#)

Select the group we created above so "dev-island-app-dev-kv-vg ". Set "Variable group scope" to "Stages" and select only "Development":

×

Link variable group ⓘ

🔍 Search

✓

**dev-island-app-dev-kv-vg (1)**

Variable group with Key Vault secrets for the Development environment.

Variable group scope

☐ Release

☒ Stages

✓ Development

▼

◀▶

Link

Click "Save" button:

## pipeline



Save



Create release

Options History

▼	Name	Value
▼	<b>dev-island-app-dev-kv-vg (1)</b> Variable group with Key Vault secrets for the Development environment.	Scopes: Development
<a href="#">Link variable group</a>   <a href="#">Manage variable groups</a>		

**Repeat above steps for the QA and Production environments**

Create variable groups for the QA and Production like presented below:

- dev-island-app-qa-kv-vg
- dev-island-app-prod-kv-vg

Library &gt; ⓘ dev-island-app-qa-kv-vg\*

Variable group

Save



Clone



Security



Help

## Properties

Variable group name

dev-island-app-qa-kv-vg

Description

Variable group with Key Vault secrets for the QA environment.



Allow access to all pipelines



Link secrets from an Azure key vault as variables ⓘ

Library > ⓘ dev-island-app-prod-kv-vg\*

Variable group |  Save  Clone  Security  Help

## Properties

Variable group name

dev-island-app-prod-kv-vg

Description

Variable group with Key Vault secrets for the Production environment.

- ☒ Allow access to all pipelines
- ☒ Link secrets from an Azure key vault as variables ⓘ

Link variable group ⓘ

 Search

- dev-island-app-prod-kv-vg (1)**  
Variable group with Key Vault secrets for the Production environment.

☒ **dev-island-app-qa-kv-vg (1)**  
Variable group with Key Vault secrets for the QA environment.

Variable group scope

- ☐ Release
- ☒ Stages

✓ QA


▼

Link



Link variable group ⓘ

 Search



**dev-island-app-prod-kv-vg (1)**

Variable group with Key Vault secrets for the Production environment.

Variable group scope

- ☐ Release
- ☒ Stages

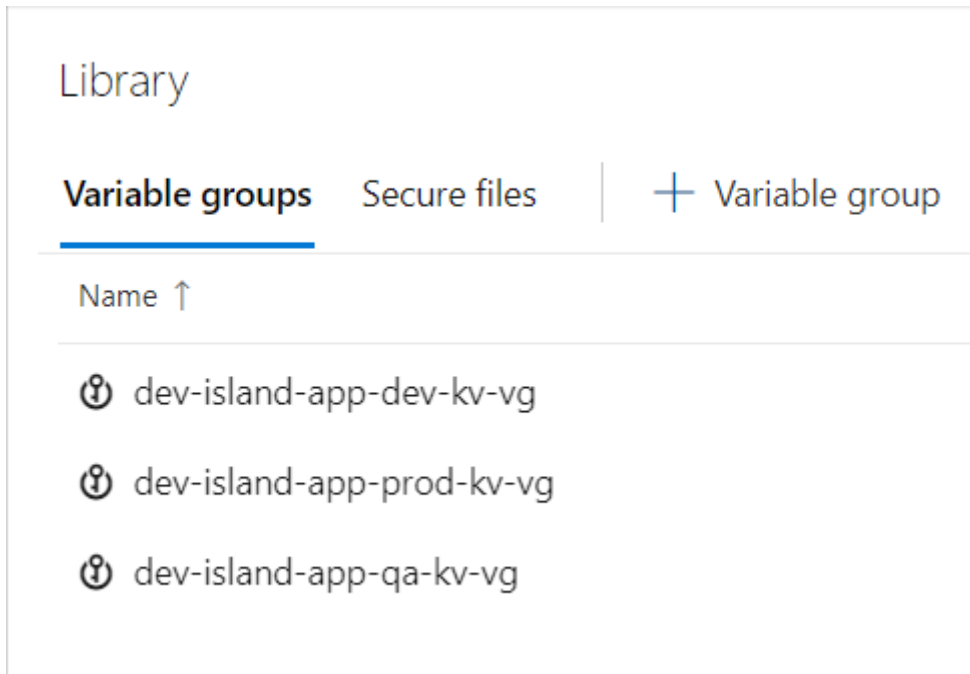
✓ Production

▼

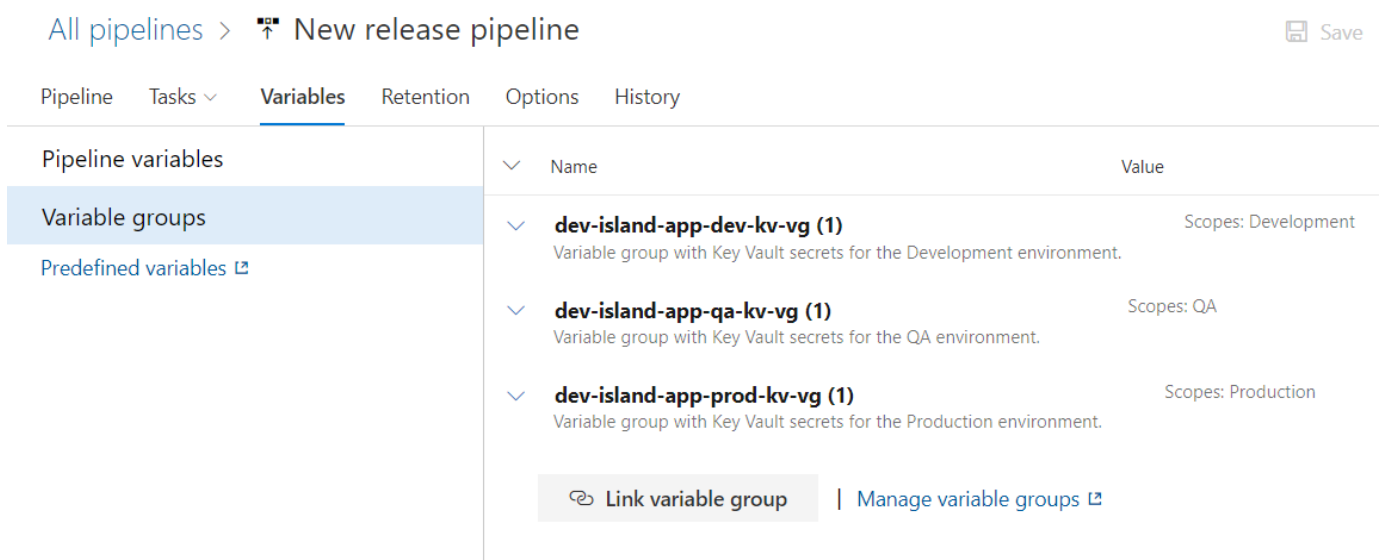
Link

Finally it looks like below:





Below three variable groups should be linked:



## Setup Development environment release

We will use ARM template to deploy sample web app with the database connection string added to the configuration. For the Development environment we would like to use database connection string from the "dev-island-app-dev-kv-vg" variable group.

"azuredeploy.json" file looks like below:

```

{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "TestAppServicePlanName": {
      "type": "string",
      "minLength": 1
    },
    "TestWebAppName": {
      "type": "string",
      "minLength": 1
    },
    "DbConnectionString": {
      "type": "string",
      "minLength": 1
    },
    "TestAppServicePlanSkuName": {
      "type": "string",
      "defaultValue": "D1",
      "allowedValues": [
        "F1",
        "D1",
        "B1",
        "B2",
        "B3",
        "S1",
        "S2",
        "S3",
        "P1",
        "P2",
        "P3",
        "P4"
      ],
      "metadata": {
        "description": "Describes plan's pricing tier and capacity. Check details at
https://azure.microsoft.com/en-us/pricing/details/app-service/
"
      }
    },
    "variables": {
      "TestWebAppName": "[concat(parameters('TestWebAppName'), uniqueString(resourceGroup().id))]"
    },
    "resources": [
      {
        "name": "[parameters('TestAppServicePlanName')]",
        "type": "Microsoft.Web/serverfarms",
        "location": "[resourceGroup().location]",
        "apiVersion": "2015-08-01",
        "sku": {

```

```
"name": "[parameters('TestAppServicePlanSkuName')]"
},
"dependsOn": [],
"tags": {
  "displayName": "TestAppServicePlan"
},
"properties": {
  "name": "[parameters('TestAppServicePlanName')]",
  "numberOfWorkers": 1
}
},
{
  "name": "[variables('TestWebAppName')]",
  "type": "Microsoft.Web/sites",
  "location": "[resourceGroup().location]",
  "apiVersion": "2015-08-01",
  "dependsOn": [
    "[resourceId('Microsoft.Web/serverfarms', parameters('TestAppServicePlanName'))]"
  ],
  "tags": {
    "[concat('hidden-related:', resourceId('Microsoft.Web/serverfarms',
parameters('TestAppServicePlanName')))]": "Resource",
    "displayName": "TestWebApp"
  },
  "properties": {
    "name": "[variables('TestWebAppName')]",
    "serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
parameters('TestAppServicePlanName'))]",
    "siteConfig": {
      "connectionStrings": [
        {
          "name": "DbTestConnectionString",
          "connectionString": "[parameters('DbConnectionString')]",
          "type": "string"
        }
      ]
    }
  }
},
"resources": [
  {
    "name": "appsettings",
    "type": "config",
    "apiVersion": "2015-08-01",
    "dependsOn": [
      "[resourceId('Microsoft.Web/sites', variables('TestWebAppName'))]"
    ],
    "tags": {
      "displayName": "TestWebAppSettings"
    },
  },

```

```
    "properties": {
      "key1": "value1",
      "key2": "value2"
    }
  }
]
}
],
"outputs": {}
}
```

“azuredeploy.parameters.json” file looks like below:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "TestAppServicePlanName": {
      "value": "TestAppServicePlan"
    },
    "TestWebAppName": {
      "value": "TestWebAppCreatedWithARM"
    },
    "DbConnectionString": {
      "value": "#"
    }
  }
}
```

The screenshot displays the 'New release pipeline' configuration in Azure DevOps. The left sidebar shows the 'Development' environment with an 'Agent job' containing the task 'Azure Deployment to the resource group'. The right sidebar shows the configuration for this task:

- Resource group \***: dev-island-app-dev-rg
- Location \***: West Europe
- Template ^**:
  - Template location \***: Linked artifact
  - Template \***: \$(System.DefaultWorkingDirectory)/\_dev-island-app-repo/arm-templates/azuredeploy.json
  - Template parameters**: \$(System.DefaultWorkingDirectory)/\_dev-island-app-repo/arm-templates/azuredeploy.parameters.json
  - Override template parameters**: -DbConnectionString \$(DbConnectionString)

Please note that in the section called "Override template parameters" we have to replace parameter with the value for the database connection string from the Key Vault:

```
-DbConnectionString $(DbConnectionString)
```

## Setup QA environment release

For the QA environment we will have the same steps like presented above. In this case we are creating web app in the QA resource group in the Azure:

All pipelines > New release pipeline

Pipeline Tasks Variables Retention Options History

QA  
Deployment process

Agent job  
Run on agent

Azure Deployment to the resource group  
Azure resource group deployment

Create or update resource group

Resource group \*  
dev-island-app-qa-rg

Location \*  
West Europe

Template ^

Template location \*  
Linked artifact

Template \*  
\$(System.DefaultWorkingDirectory)/\_dev-island-app-repo/arm-templates/azuredeploy.json

Template parameters  
\$(System.DefaultWorkingDirectory)/\_dev-island-app-repo/arm-templates/azuredeploy.parameters.json

Override template parameters  
-DbConnectionString \$(DbConnectionString)

## Setup Production environment release

For the Production environment we will have the same steps like presented above. In this case we are creating web app in the Production resource group in the Azure:

All pipelines > New release pipeline

Pipeline Tasks Variables Retention Options History

Production  
Deployment process

Agent job  
Run on agent

Azure Deployment to the resource group  
Azure resource group deployment

Create or update resource group

Resource group \*  
dev-island-app-prod-rg

Location \*  
West Europe

Template ^

Template location \*  
Linked artifact

Template \*  
\$(System.DefaultWorkingDirectory)/\_dev-island-app-repo/arm-templates/azuredeploy.json

Template parameters  
\$(System.DefaultWorkingDirectory)/\_dev-island-app-repo/arm-templates/azuredeploy.parameters.json

Override template parameters  
-DbConnectionString \$(DbConnectionString)

# Create new release and check web apps configuration

Create new release. Once web app is created in each of the resource groups, check "Configuration" tab. You should see that each app has different database connection string retrieved from the Key Vault related with specific environment:

The screenshot displays the Azure DevOps Release Pipelines interface. The top section, titled "Create a new release", includes a "New release pipeline" link and a "Pipeline" dropdown menu. Below this, a horizontal pipeline diagram shows three stages: "Development", "QA", and "Production", each with a lightning bolt icon. A note indicates that clicking on a stage changes its trigger from automated to manual. Below the diagram, there is a section for "Stages for a trigger change from automated to manual" with an information icon and a dropdown menu. The "Artifacts" section is also visible, with a note to "Select the version for the artifact sources for this release" and a table with columns "Source alias" and "Version". At the bottom, there are "Create" and "Cancel" buttons.

**Create a new release**  
New release pipeline

**Pipeline** ^  
Click on a stage to change its trigger from automated to manual.

Development QA Production

Stages for a trigger change from automated to manual. ⓘ

**Artifacts** ^  
Select the version for the artifact sources for this release

Source alias	Version
--------------	---------

Create Cancel

**Deployment**

- Quickstart
- Deployment slots
- Deployment Center

**Settings**

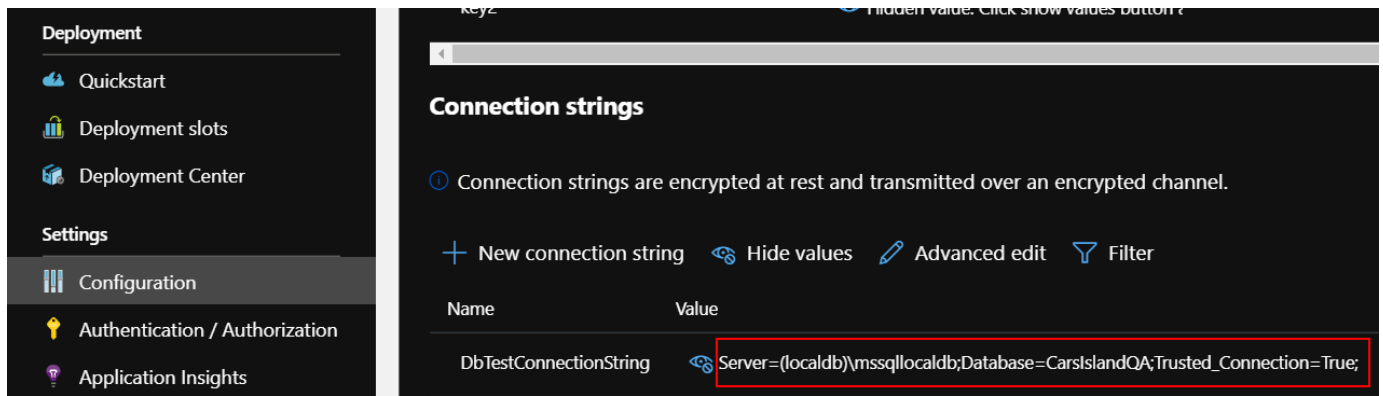
- Configuration
- Authentication / Authorization
- Application Insights

**Connection strings**

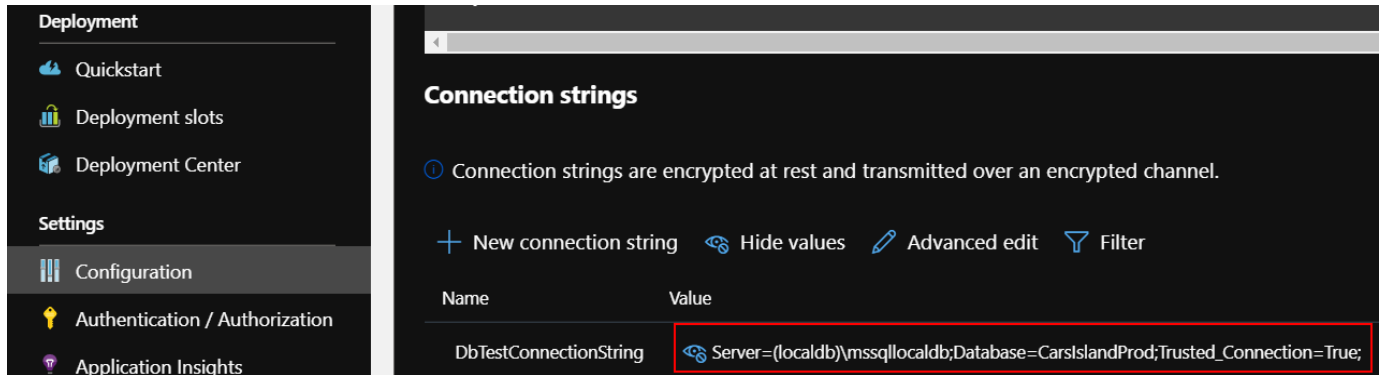
ⓘ Connection strings are encrypted at rest and transmitted over an encrypted channel.

+ New connection string Hide values Advanced edit Filter

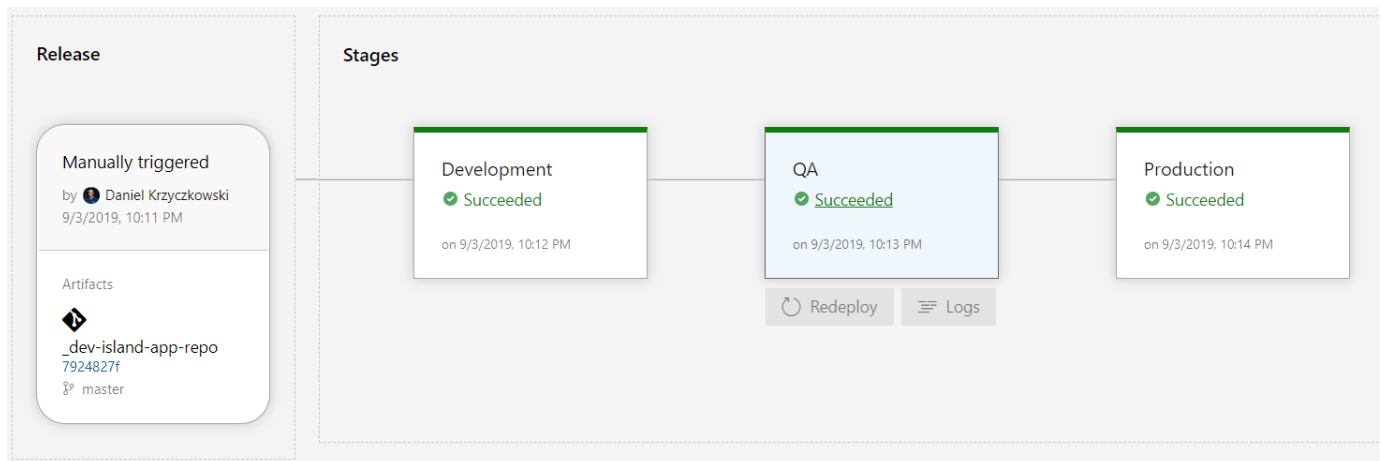
Name	Value
DbTestConnectionString	Server=(localdb)\mssqllocaldb;Database=CarsIsland;Trusted_Connection=True;



The screenshot shows the 'Configuration' page in Azure DevOps. On the left, the 'Settings' menu is visible with 'Configuration' selected. The main area is titled 'Connection strings' and contains a table with one entry: 'DbTestConnectionString'. The value for this string is 'Server=(localdb)\mssqllocaldb;Database=CarsIslandQA;Trusted\_Connection=True;', which is highlighted with a red box. Above the table, there are buttons for 'New connection string', 'Hide values', 'Advanced edit', and 'Filter'.



This screenshot is similar to the one above, but the connection string value is 'Server=(localdb)\mssqllocaldb;Database=CarsIslandProd;Trusted\_Connection=True;', also highlighted with a red box. The 'Name' of the connection string remains 'DbTestConnectionString'.



The screenshot displays a completed release pipeline. On the left, the 'Release' section shows a 'Manually triggered' build by Daniel Krzyczkowski on 9/3/2019 at 10:11 PM, with an artifact named '\_dev-island-app-repo' (7924827f) on the 'master' branch. The 'Stages' section on the right shows three stages: 'Development', 'QA', and 'Production', all of which have 'Succeeded' and were completed on 9/3/2019 at 10:12 PM, 10:13 PM, and 10:14 PM respectively. Below the 'QA' stage, there are buttons for 'Redeploy' and 'Logs'.

## Summary

In this article I explained how to inject Azure Key Vault secrets in the Azure DevOps release pipelines for multiple environments using variable groups. I hope you will find it valuable. Of course instead of variable groups you could use "Add Azure Key Vault" task in the release definition but I wanted to show another approach.

 **Updated:** September 03, 2019



