Module 10 - The Release Pipeline

Exercise 1: Azure DevOps Introduction

Introduction

Your team has decided to implement Microsoft Azure for their cloud infrastructure needs. To ensure a consistent repeatable deployment you have chosen to use Azure Devops in place of a manual process. Utilising the features of Infrastructure as Code will allow you to monitor and maintain the Azure infrastructure from a central repository and deployment plane.

Notes

Please note that some screen shots may differ slightly from what you see when completing the steps in Azure DevOps due to the Agile release of the product. Your trainer will be able to assist with any breaking changes.

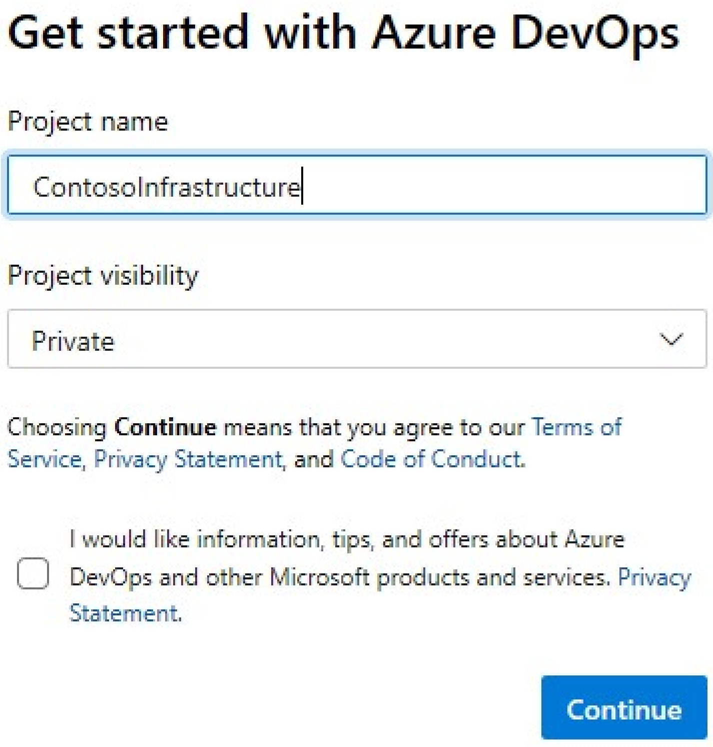
Estimated Time to Complete This Lab

30 minutes

Task 1: Create an Azure DevOps Project

1. [] Navigate to https://dev.azure.com and sign in with your Azure email address. Note that you may have to provide your name and email address to continue.

When complete you should be presented with a screen prompting you to create a project.



Get started with Azure DevOps

Project name: ContosoInfrastructure

Project visibility: Private

Choosing **Continue** means that you agree to our Terms of Service, Privacy Statement, and Code of Conduct.

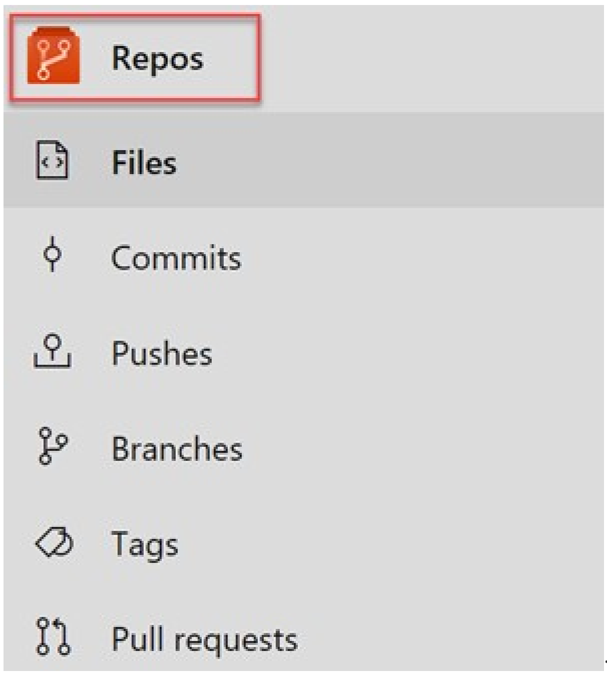
☐ I would like information, tips, and offers about Azure DevOps and other Microsoft products and services. Privacy Statement.

{400}

2. [] Enter **ContosoInfrastructure** as the project name and click **Continue**
3. [] When complete you now have a blank project to add code to and build pipelines.

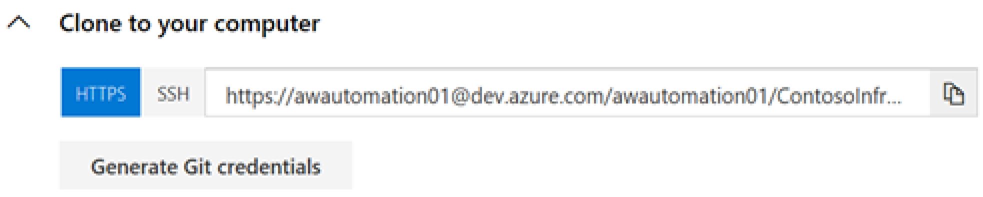
Task 2: Create a Project and Add Some Code (Source)

1. [] Click on the Azure Repos icon on the left hand side.



{400}

2. [] We will use Git to clone the project to our local machine. This way we can make changes locally and then push them to Azure Repos. Click the **copy** icon to copy the URL. Save the URL into a temporary notepad file for later use. Note that your url will be different to the one pictured.


Clone to your computer

HTTPS SSH https://awautomation01@dev.azure.com/awautomation01/ContosoInfr...

Generate Git credentials

{400}

3. [] Git is installed on your machine already and is available in the $PATH - so we can run it from any terminal. Open PowerShell and enter the code below to create a new folder for the repository to be placed into.

```
cd \
New-Item C:\Git_Repos -ItemType Directory
cd Git_Repos
```

4. [] Type in `git clone` and then paste the command line you copied from Azure Repos and press enter. Your command should look similar to the one below.

```
git clone
https://awautomation01@dev.azure.com/awautomation01/ContosoIn
```

5. [] Enter your credentials when prompted.

6. [] When complete you should have text similar to the below at the prompt.

```
Cloning into 'ContosoInfrastructure'...
warning: You appear to have cloned an empty repository.
```

7. [] Git will have created its own folder now to store the repository information. By default it is a hidden folder. Run the command below to

switch into the new directory for the repository and also list the hidden `.git` folder.

```
cd ContosoInfrastructure
Get-ChildItem -Hidden
```

8. [] We can now add some files into the repository - let's set up a scaffold to work from by running the code below to create some new folders for the files.

```
New-Item Templates -ItemType Directory
New-Item Scripts -ItemType Directory
```

9. [] Copy the files from C:\Labs\Module10\Scripts into the Scripts folder and the files from C:\Labs\Module10\Templates into the templates folder. **DO NOT** copy **storage.json** file as we will add this template later.

10. Git will detect that you have made changes to the repository. Type the command below and observe the output.

```
git checkout -b main
git status
```

Output:

```
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be
committed)

        Scripts/
        Templates/
```

```
nothing added to commit but untracked files present (use
"git add" to track)
```

11. Before a commit can be made - we have to add the files to a staging location within Git. This just tells Git that we are preparing to take a snapshot (commit) of the current repository. Run the code below to add all files to the staging area.

```
git add .
git status
```

You should have output similar to the below.

```
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   Scripts/1_CreateResourceGroups.ps1
        new file:   Scripts/2_DeployVirtualNetwork.ps1
        new file:   Templates/vnet.json
```

12. [] The files have been added to the staging area. Type the commands below to set the user and press Enter after each one. Change the email address and name as appropriate. This only has to be performed once per machine.

```
git config --global user.email "youremailaddress"
git config --global user.name "your name"
```

13. [] Run the code below to make your first commit and push it to Visual Studio Team Services.

```
git commit -m "Initial commit"
git push --set-upstream origin main
```

The output should be similar to below:

```
PS C:\Git_Repos\ContosoInfrastructure> git commit -m
"Initial commit"
[main (root-commit) 3fd7733] Initial commit
 3 files changed, 28 insertions(+)
 create mode 100644 Scripts/1_CreateResourceGroups.ps1
 create mode 100644 Scripts/2_DeployVirtualNetwork.ps1
 create mode 100644 Templates/vnet.json
PS C:\Git_Repos\ContosoInfrastructure> git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 845 bytes | 281.00 KiB/s,
done.
Total 7 (delta 0), reused 0 (delta 0)
remote: Analyzing objects... (7/7) (4 ms)
remote: Storing packfile... done (109 ms)
remote: Storing index... done (28 ms)
To
https://dev.azure.com/awautomation01/ContosoInfrastructure/_g
 * [new branch]      main -> main
```

14. [] In the Azure DevOps portal - click the **Files** icon.

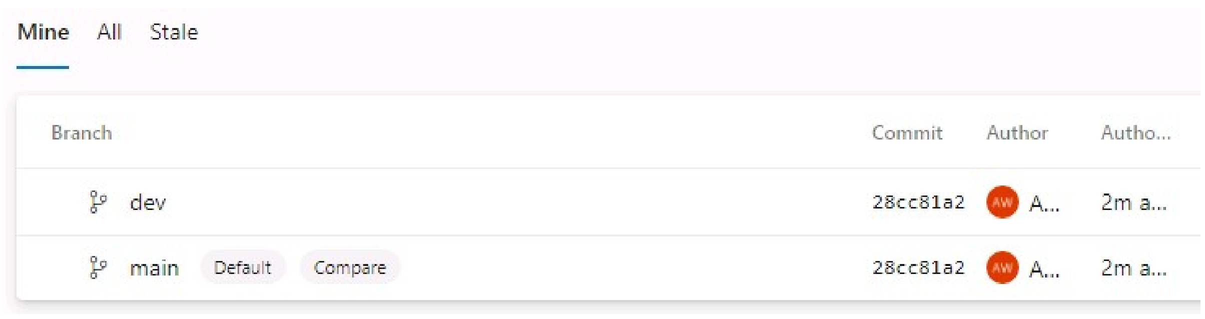
15. [] You should see the files checked in to the project. You can edit the code and perform commits directly in the portal.

Task 3: Create a Branch for Development

1. [] Let's create a branch for us to do development work on. When code is checked into this branch it will begin our build process. In the Git command pane type the following commands and press Enter after each one.

```
git branch dev
git push --set-upstream origin dev
git checkout dev
```

2. [] In Azure Repos click on branches - you should see something similar to the below image.



Branch	Commit	Author	Autho...
dev	28cc81a2	A...	2m a...
main Default Compare	28cc81a2	A...	2m a...

{400}

> You have now set up the **SOURCE** section of the **Release Pipeline**.

Exercise 2: Azure Pipelines Build

Introduction

You can now check in code to your project using Git and the branch structure. It is decided that the preliminary build process will run whenever code is committed into the dev branch.

A build script now needs to be written and added to the project and the build tasks developed within Azure Pipelines.

Prerequisites (if applicable)

N/A

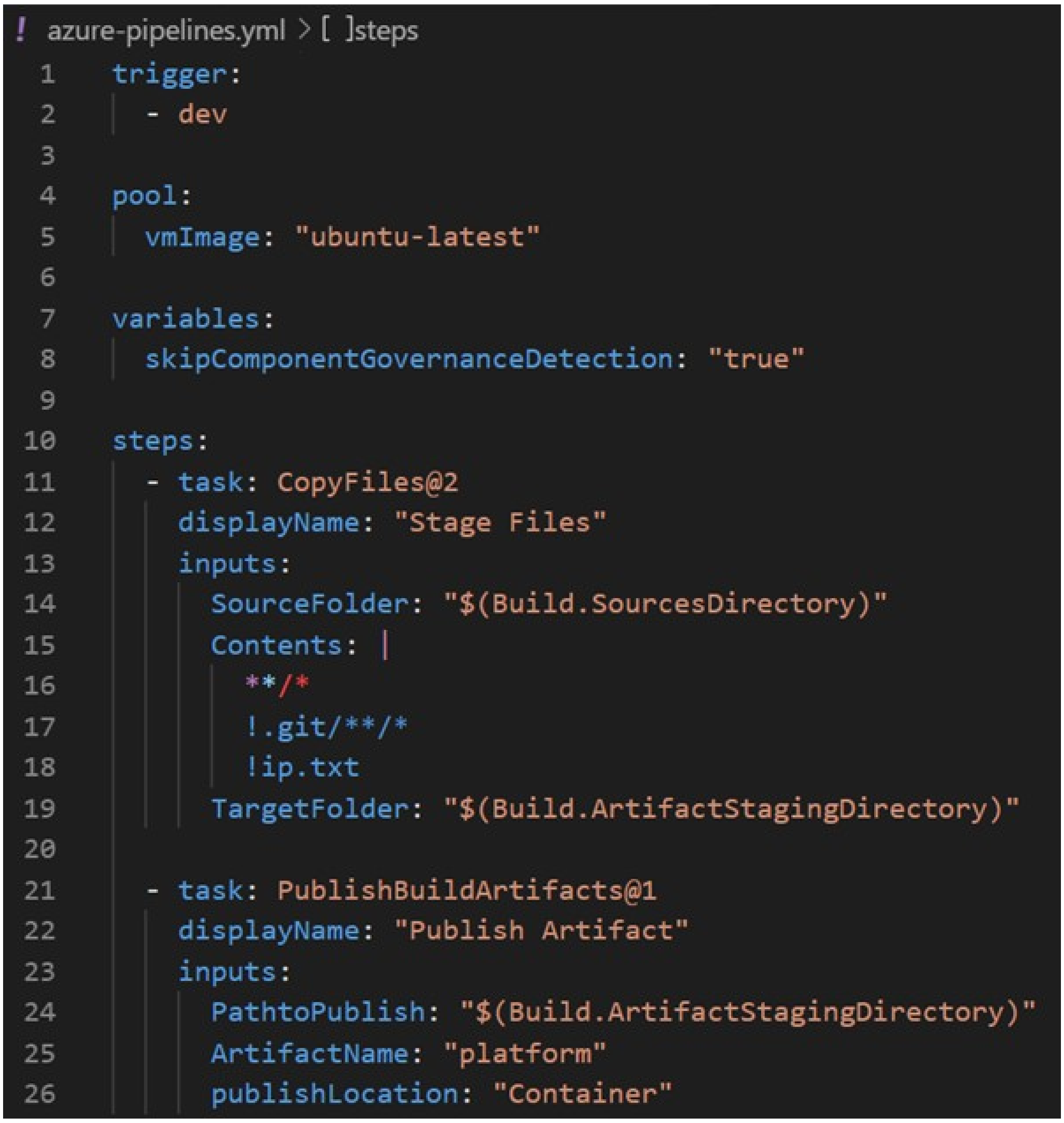
Estimated Time to Complete This Lab

15 minutes

Task 1: The Build Definition

1. [] In a software development pipeline - a build pipeline might be used to compile code into an executable or package which can then be released. When we use Devops for Infrastructure as Code - the build section of the pipeline simply creates an artifact containing scripts and templates we can use later. That being the case, the build pipeline we will use is simply going to copy some files to a staging location and publish them for use by out release process.

2. [] Copy the file C:\Labs\Module10\azure-pipelines.yml to C:\Git_Repos\ContosoInfrastructure - and open the file using Visual Studio Code. It should look something like the file below.



```yaml
! azure-pipelines.yml > [ ]steps
trigger:
  - dev

pool:
  vmImage: "ubuntu-latest"

variables:
  skipComponentGovernanceDetection: "true"

steps:
  - task: CopyFiles@2
    displayName: "Stage Files"
    inputs:
      SourceFolder: "$(Build.SourcesDirectory)"
      Contents: |
        **/*
        !.git/**/*
        !ip.txt
      TargetFolder: "$(Build.ArtifactStagingDirectory)"

  - task: PublishBuildArtifacts@1
    displayName: "Publish Artifact"
    inputs:
      PathtoPublish: "$(Build.ArtifactStagingDirectory)"
      ArtifactName: "platform"
      publishLocation: "Container"
```

{400}

3. [] Azure Pipelines uses a YAML definition for the build pipeline - observe that there are several different sections and check some the included links for a description.

    - Trigger - sets which branch the build runs for → https://docs.microsoft.com/en-us/azure/devops/pipelines/build/triggers
    - Pool - what kind of agent will this process run on → https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/hosted

   - Variables - variables which are used in the build process. We will rely heaviliy on these during the release to specify different environments → https://docs.microsoft.com/en-us/azure/devops/pipelines/process/variables
   - Tasks - these perform the work in the pipeline. There are many included out of the box and are also avaialable from the Azure Devops marketplace → https://docs.microsoft.com/en-us/azure/devops/pipelines/process/tasks

4. [] In the PowerShell windows - enter the code below and note that Git has detected the new file.

```
git status
```

Output:

```
On branch dev
Your branch is up to date with 'origin/dev'.

Untracked files:
  (use "git add <file>..." to include in what will be
committed)

        azure-pipelines.yml

nothing added to commit but untracked files present (use
"git add" to track)
```

5. [] Add the file to the staging area and commit it to the repository by running the code below.

```
git add .
git commit -m "Added a build pipeline"
git push
```

Output:

```
PS C:\Git_Repos\ContosoInfrastructure> git add .
PS C:\Git_Repos\ContosoInfrastructure> git commit -m "Added
a build pipeline"
[dev 177fde4] Added a build pipeline
 1 file changed, 26 insertions(+)
 create mode 100644 azure-pipelines.yml
PS C:\Git_Repos\ContosoInfrastructure> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 617 bytes | 308.00 KiB/s,
done.
Total 3 (delta 0), reused 0 (delta 0)
remote: Analyzing objects... (3/3) (32 ms)
remote: Storing packfile... done (40 ms)
remote: Storing index... done (39 ms)
To
https://dev.azure.com/awautomation01/ContosoInfrastructure/_g
   3fd7733..177fde4  dev -> dev
```

6. [] Switch to the Azure Devops window and click on Pipelines.
7. [] Click on **Create Pipeline**
8. [] Click on **Azure Repos Git**
9. [] Click on **ContosoInfrastructure**
10. [] Click on **Existing Azure Pipelines YAML File**
11. [] In the branch dropdown - change the branch to **dev** and you should be able to select your file. Click on **Continue**
12. [] The YAML file will load in the editor - click on **Run** to start the pipeline. See notes below.

Important

Microsoft has recently removed the ability to user free pipeline runners in order to prevent abuse. The steps below will create a build agent in an Azure Container instance for your use.

1. [] Log in to the Azure Cloud Shell (shell.azure.com) and ensure you are in a PowerShell prompt (pwsh)
2. [] Use the command below to clone a repository that contains steps to build and agent.

```
git clone https://github.com/anwather/SelfHostedAgent.git
cd SelfHostedAgent
```

3. [] Edit the `deploy.ps1` file and fill in the details below.

Parameter	Value
location	Azure Location
acrName	Unique name for an Azure container registry
AZP_URL	The URL for your Azure DevOps organization - it will be similar to https://dev.azure.com/yourorganizationname
AZP_TOKEN	Generate a PAT token with Agent Pool (read and manage capablilities) - see https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/v2-linux?view=azure-devops#authenticate-with-a-personal-access-token-pat|

4. [] Save the file and run `deploy.ps1`

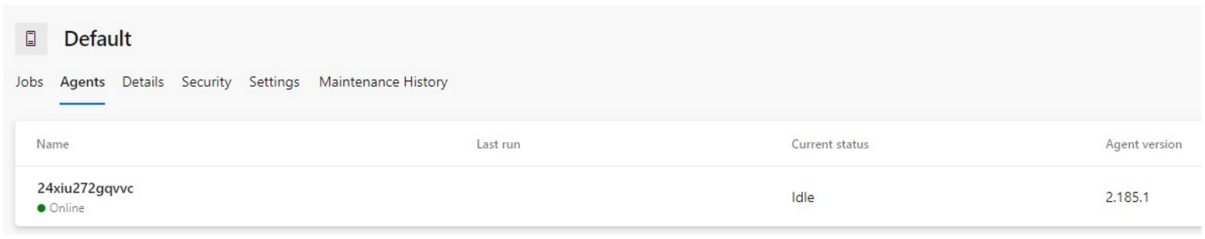
```
./deploy.ps1
```

5. [] All the steps to create an agent have been automated in the script - but complete the activities below:
   - Create a new resource group
   - Create a new service principal and assign it permissions

- Deploy a new Azure Container Registry
- Build a container image which runs the DevOps agent
- Deploy the image as an Azure Container Instance and register it to Azure DevOps

The script can take a few minutes to run as the base container image installs software and updates

6. [] Once complete, in Azure DevOps you can click on Settings → Agent Pools → Default and you will see the agent waiting for jobs.



Name	Last run	Current status	Agent version
24xiu272gqvvc ● Online		Idle	2.185.1

{400}

7. [] You can now attempt to run the pipeline again - when it starts you will need to click Authorize to allow the agent access.

Resumption of Lab Activities

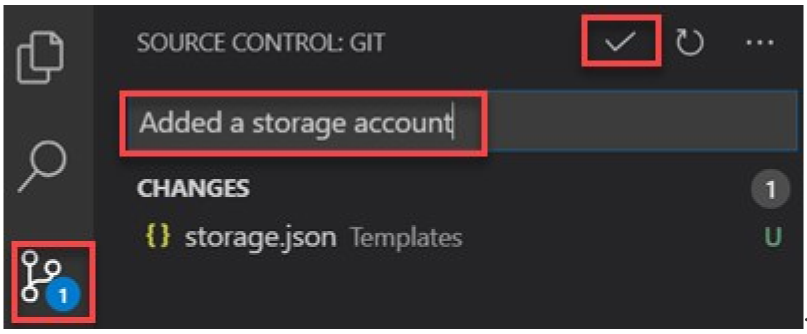
13. [] Click on the build which is running and explore the output - you can see the different task logs - and click on Artifact to see which files are going to be used in the release pipeline.

14. [] Create a new file in the templates folder called **storage.json** and copy/paste the JSON below into it.

```
{
    "$schema":
"https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
    },
    "variables": {
    },
    "resources": [
        {
            "name": "#{StorageAccountName}#",
            "type": "Microsoft.Storage/storageAccounts",
            "apiVersion": "2019-04-01",
            "location": "[resourceGroup().location]",
            "sku": {
                "name": "Standard_LRS"
            },
            "kind": "StorageV2",
            "tags": {
                "displayName": "#{StorageAccountName}#",
                "environment": "#{Release.EnvironmentName}
#"
            },
            "properties": {
            }
        }
    ],
    "outputs": {
    }
}
```

15. [] Use the Git process above to add/commit/push the file to Azure Devops - when complete the build process will run again.

HINT: Visual Studio Code has Git integration - if you open the ContosoInfrastructure folder in VS Code you will see it detect the changes. You can use the Source Control tab in VS Code to enter commit messages and push changes to a remote repository.


SOURCE CONTROL: GIT

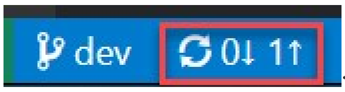
Added a storage account

CHANGES 1

{} storage.json Templates U

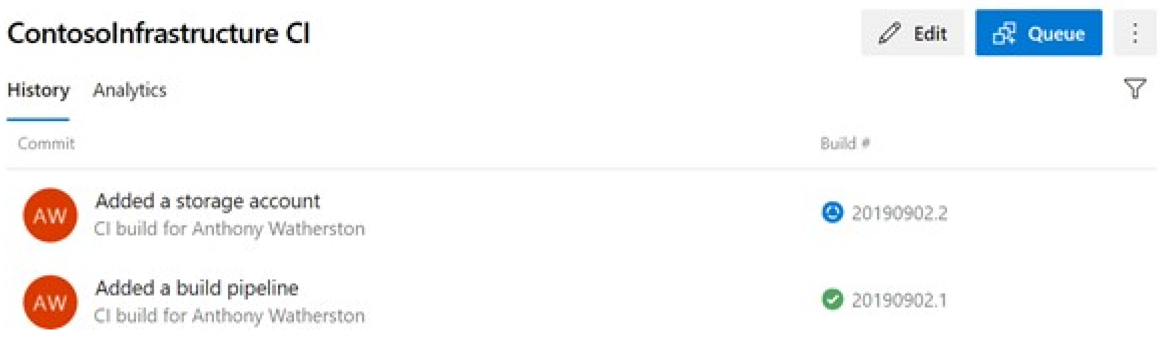
{400}

The button at the bottom of the VS Code screen will show the current branch and you can click on the arrow icon to push/pull changes.



{400}

16. [] When the new JSON file is pushed and the build is running - your Azure Pipeline screen should show multiple jobs running.


ContosoInfrastructure CI

Commit	Build #
Added a storage account CI build for Anthony Watherston	20190902.2
Added a build pipeline CI build for Anthony Watherston	20190902.1

{400}

Exercise 3: Azure Pipelines Release / Test

Introduction

There is now code in Azure Repos and a build pipeline which creates an artifact for release. In the last section we will create the release pipeline and integrate testing.

Estimated Time to Complete This Lab

25 minutes

Task 1: Service Connection

1. [] Release pipelines in Azure Devops are beginning to support YAML based definitions however this exercise will use the class UI and designer.

2. [] Before creating the release we need to create a connection to Azure Resource Manager. In the Azure Devops windows click on **Project Settings** at the bottom of the screen.

3. [] Under Pipelines - click on **Create service connection**

4. [] Select **Azure Resource Manager** and click **Next**

5. [] Ensure that **Service principal (automatic)** is selected and click **Next**

6. [] In the connection name enter **AzureConnection**. Ensure that the Resource Group field is left blank so that the service principal is given access to the entire subscription. Click **Save**

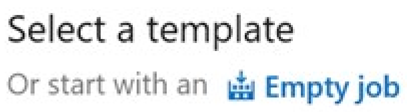
7. [] When complete we can use this service connection to authenticate to Azure in our pipeline.

Task 2: Create a release definition

1. [] Click on **Pipelines → Releases**

2. [] Click on **New Pipeline**

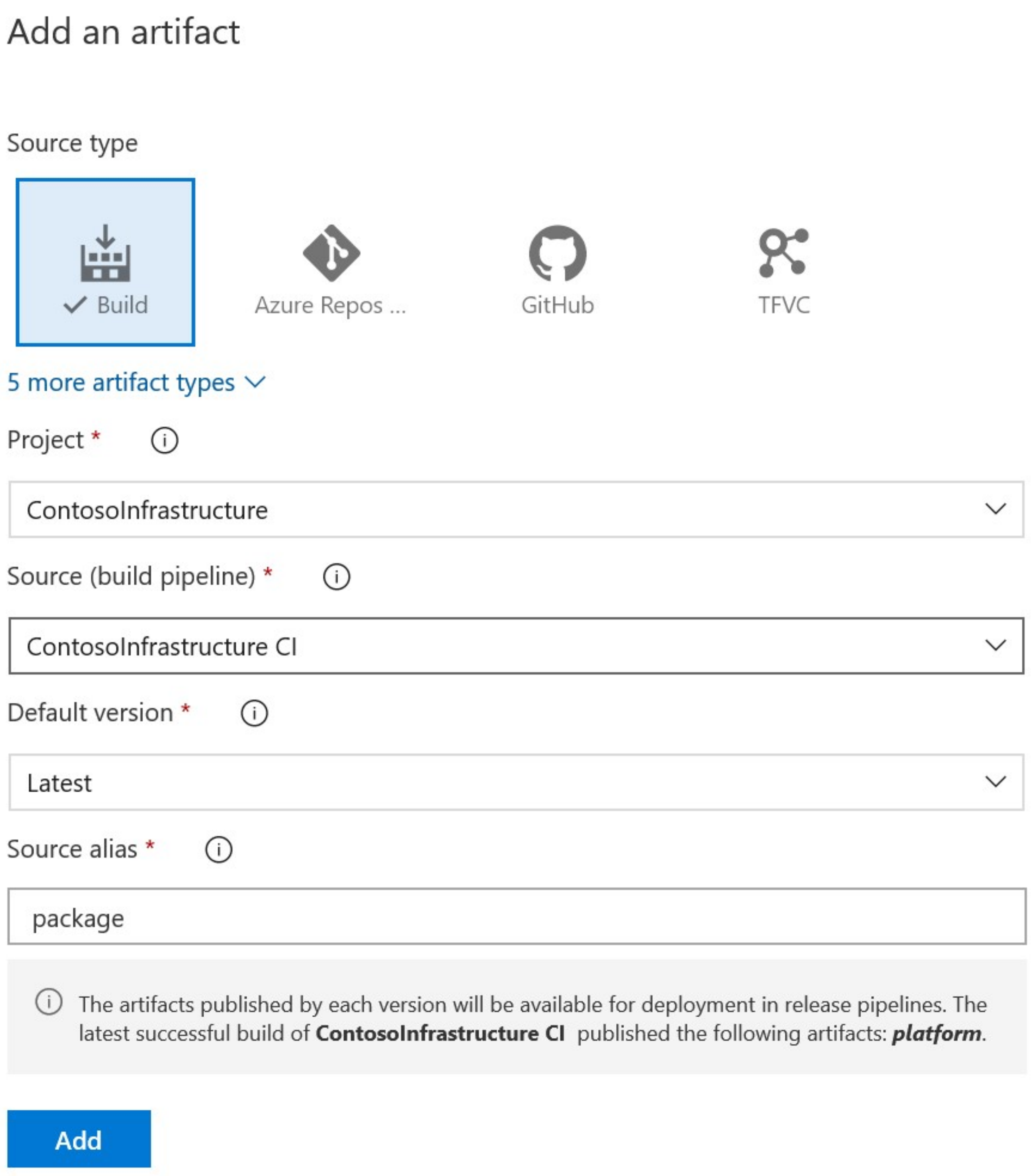
3. [] Click on **Empty Job**



{400}

4. [] In the window that opens rename the stage as development.

5. [] Click on **Add an Artifact**. Fill out the fields as below and click on Add. This will add the package we create in the build process to the release process.



Add an artifact

Source type: Build

Project *: ContosoInfrastructure

Source (build pipeline) *: ContosoInfrastructure CI

Default version *: Latest

Source alias *: package

The artifacts published by each version will be available for deployment in release pipelines. The latest successful build of **ContosoInfrastructure CI** published the following artifacts: ***platform***.

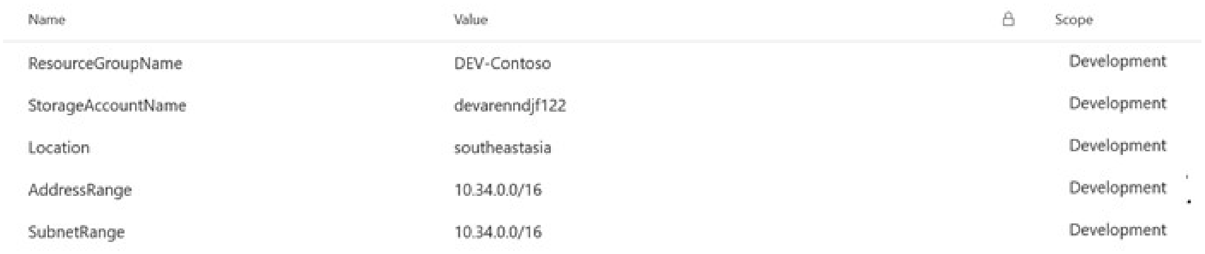
{400}

6. [] Now we have to add variables to the release which will be consumed when the pipeline runs. Click on **Variables**

7. [] To add a new variable you click on **Add** and supply some values. We can also modify the scope so that variables can be specified multiple time in a pipeline but change depending on the environment. Add some variables in as below. Do not forget to **Save** your variables.

Variable Name	Value	Scope
ResourceGroupName	DEV-Contoso	Development

Variable Name	Value	Scope
StorageAccountName	(add a unique name for the account)	Development
Location	(add an allowed location in here)	Development
AddressRange	10.34.0.0/16	Development
SubnetRange	10.34.0.0/16	Development



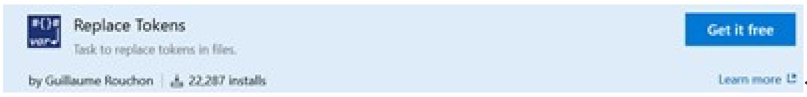
Name	Value	Scope
ResourceGroupName	DEV-Contoso	Development
StorageAccountName	devarenndjf122	Development
Location	southeastasia	Development
AddressRange	10.34.0.0/16	Development
SubnetRange	10.34.0.0/16	Development

{400}

8. Click on **Tasks**

9. Click the + symbol to add a task. In the window that opens type *replace tokens* in the search box.

10. Click on the task highlighted below and select **Get it free**


Replace Tokens
Task to replace tokens in files.
by Guillaume Rouchon | 22,287 installs
Get it free
Learn more
{400}

11. [] Follow the instructions as prompted to add the task to your Azure Devops organisation. When complete you can use this task in a pipeline.

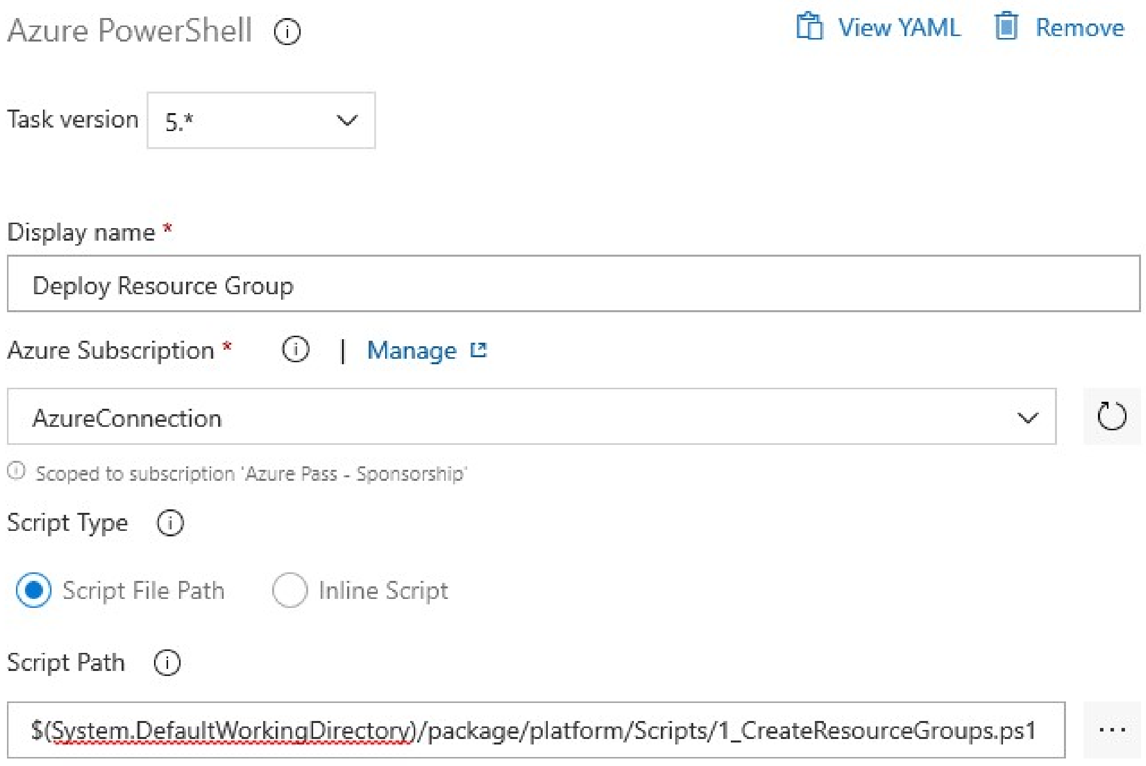
12. Click the + symbol to add a task. In the window that opens type *replace tokens* in the search box and this time select the installed task.

13. [] Change the task as follows:-

    - Display name: Replace Tokens
    - Root directory: $(System.DefaultWorkingDirectory)/package/platform/Templates
    - Target files: *.json

This task will go through the JSON files in the package and replace tokens that have a special delimter `#{}#` with variables.

14. [] Add another task - this time search for Azure PowerShell. Modify the values in there to match the image below. The connection will be the service connection we created previously. We use the 5.* version of the task as it installs the Az PowerShell modules.



Task version: 5.*

Display name *: Deploy Resource Group

Azure Subscription *: AzureConnection

Scoped to subscription 'Azure Pass - Sponsorship'

Script Type: (●) Script File Path ( ) Inline Script

Script Path: $(System.DefaultWorkingDirectory)/package/platform/Scripts/1_CreateResourceGroups.ps1

{400}

15. [] Add another Azure PowerShell task - update the name to be **Deploy Virtual Network** and change the script path to use the **2_DeployVirtualNetwork.ps1** script file.

16. [] Add another Azure PowerShell task - update the name to be **Deploy Storage Account** and change the script path to use the **3_DeployStorageAccount.ps1** script file.

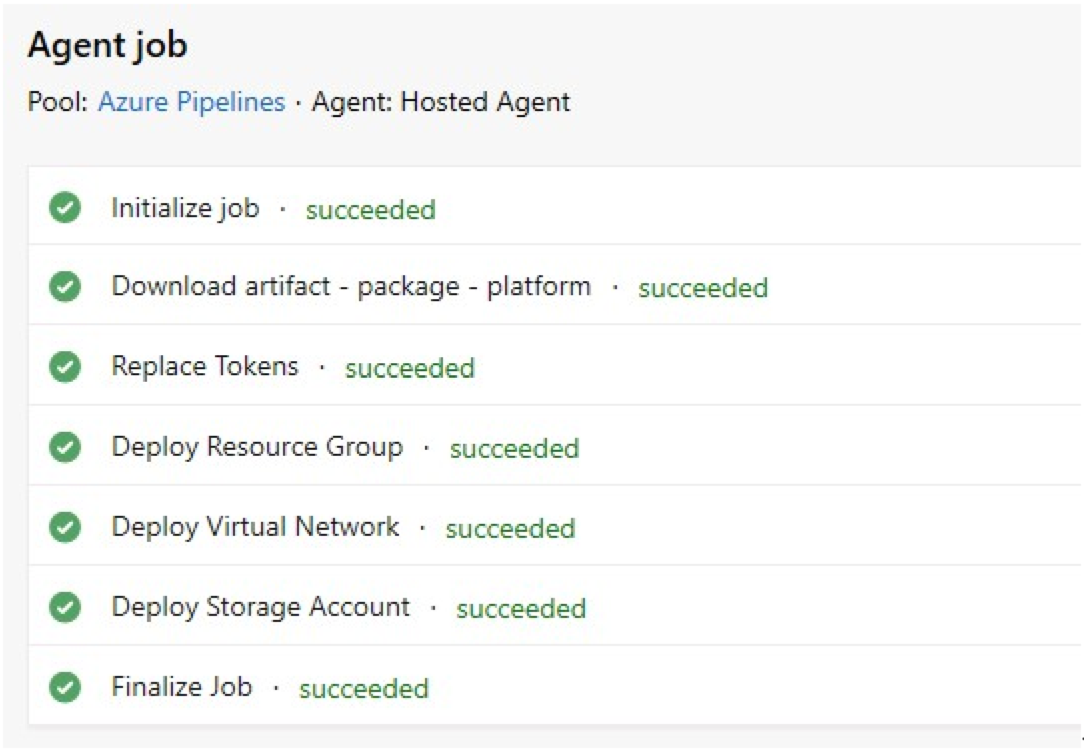
17. Click on **Save**

Task 3: Deploy the release

1. [] In the same window - click on **Create Release**

2. [] Click on **Create**

3. [] Click on the link that is created and review the release as it is happening.



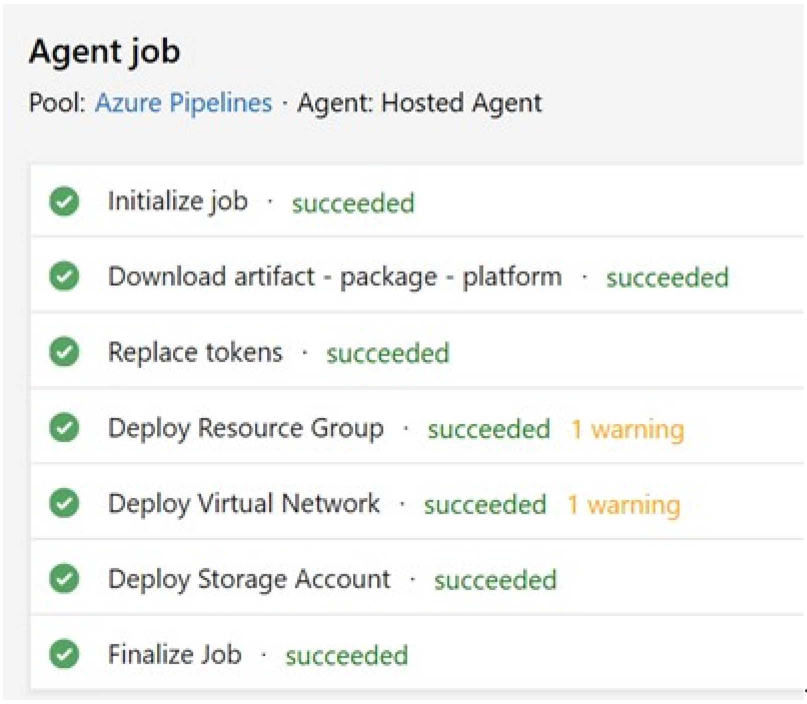
Agent job

Pool: Azure Pipelines · Agent: Hosted Agent

- ✓ Initialize job · succeeded
- ✓ Download artifact - package - platform · succeeded
- ✓ Replace Tokens · succeeded
- ✓ Deploy Resource Group · succeeded
- ✓ Deploy Virtual Network · succeeded
- ✓ Deploy Storage Account · succeeded
- ✓ Finalize Job · succeeded

{400}

4. When complete you should be able to browse to the Azure portal and see your resources created in there.

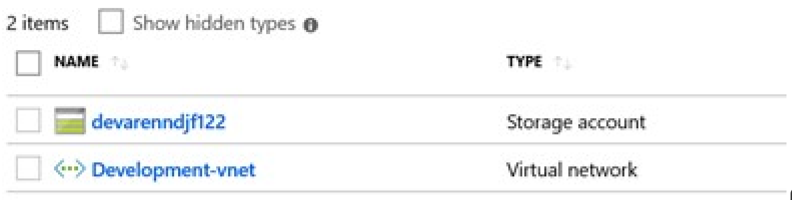


Agent job

Pool: Azure Pipelines · Agent: Hosted Agent

- ✓ Initialize job · succeeded
- ✓ Download artifact - package - platform · succeeded
- ✓ Replace tokens · succeeded
- ✓ Deploy Resource Group · succeeded 1 warning
- ✓ Deploy Virtual Network · succeeded 1 warning
- ✓ Deploy Storage Account · succeeded
- ✓ Finalize Job · succeeded

{400}



2 items Show hidden types

NAME	TYPE
devarenndjf122	Storage account
Development-vnet	Virtual network

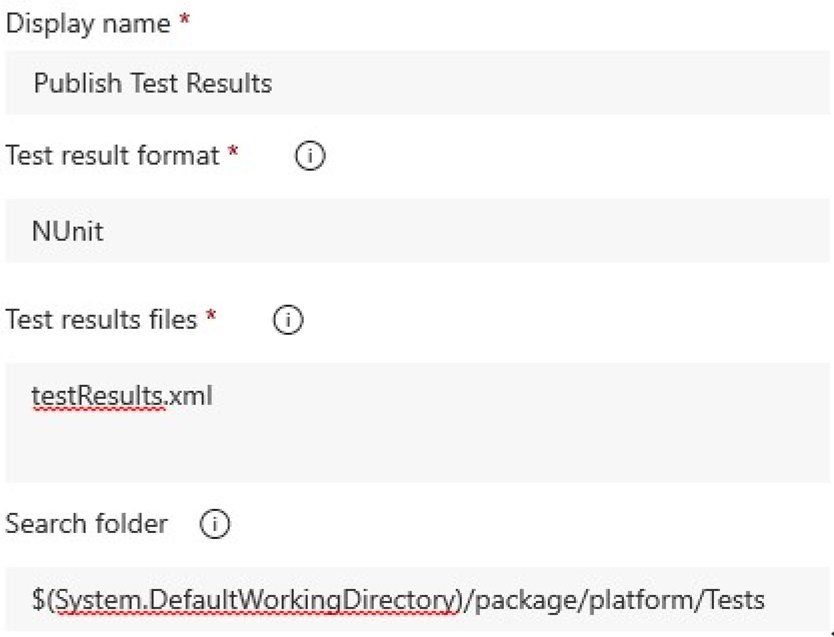
{400}

Task 4: Adding Testing to Validate the Deployment

1. [] Copy the folder from C:\Labs\Module10\Tests to C:\Git_Repos\ContosoInfrastructure
2. [] Follow the procedures like before to add/commit/push the changes to the repository.

```
git add .
git commit -m "Added some tests"
git push
```

3. [] Check that a build has occurred and completed successfully.
4. [] The PowerShell script uploaded uses the Pester module to do some infrastructure testing against the deployed objects. The output will be an XML file that Azure Pipelines can import and display the results. Click on **Pipelines** → **Releases**.
5. [] Click on **Edit** → **Tasks**
6. [] Add another Azure PowerShell task - update the name to be **Run Tests** and change the script path to use the **RunTests.ps1** script file. You can use the path to find the script as it will be part of the build now.
7. [] Add another task - this time search for **Publish Test Results**. Configure the task settings to match the image below.



Display name *

Publish Test Results

Test result format *

NUnit

Test results files *

testResults.xml

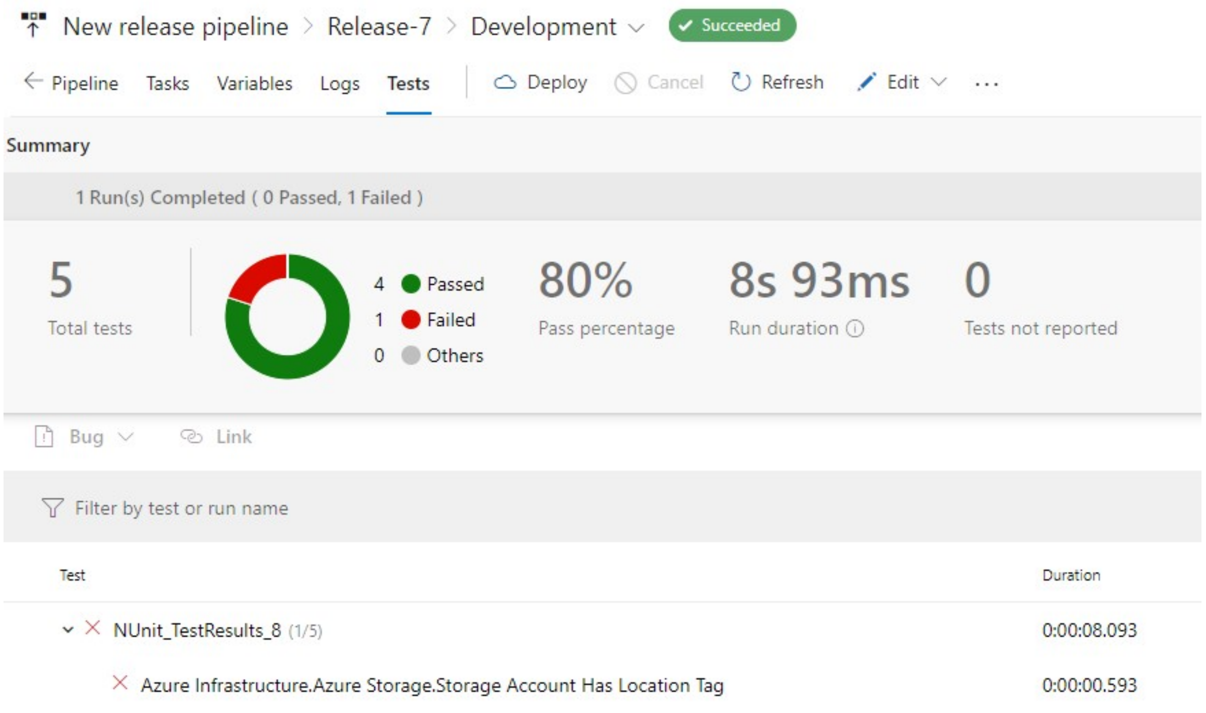
Search folder

$(System.DefaultWorkingDirectory)/package/platform/Tests

{400}

8. Save the release - and then click on **Create Release**. Click **Create**

9. Click on the link for the release and follow it through - this time the tests will be run after the deployment. When complete click on the **Tests** tab in the release logs.



Summary

1 Run(s) Completed (0 Passed, 1 Failed)

5 Total tests

4 ● Passed
1 ● Failed
0 ● Others

80% Pass percentage

8s 93ms Run duration

0 Tests not reported

Test	Duration
NUnit_TestResults_8 (1/5)	0:00:08.093
Azure Infrastructure.Azure Storage.Storage Account Has Location Tag	0:00:00.593

{400}

Looks like there is a failure to configure a tag in our code - we are checking to ensure that the storage account has a location tag, but one isn't configured or the value is incorrect.

Use what you have learnt about ARM to modify the `storage.json` file to update the tags.

When complete follow the steps above to now add/commit/push/build/release.

Exercise 4: Add a Production Deployment

Introduction

You have completed an Infrastructure as Code deployment to a development environment. Using what you have learnt complete each step below to enable a production deployment. Ask your trainer if you need help with any of the steps.

Remember the aim is to not modify the code - simply change the release pipeline.

Estimated Time to Complete This Lab

25 minutes

Task 1: Complete the following steps

1. [] Clone the **Development** environment in the Release.
2. [] Duplicate the variables to represent a **Production** deployment.

Bonus Task

1. [] Implement a production branch in Azure Devops.
2. [] Trigger the build to run automatically on this branch (change the azure-pipelines.yml file).
3. [] Add a condition to each environment - to only deploy that environment if the branch name matches the environment name.

Congratulations you have implemented an Azure Devops Infrastructure as Code Pipeline!