

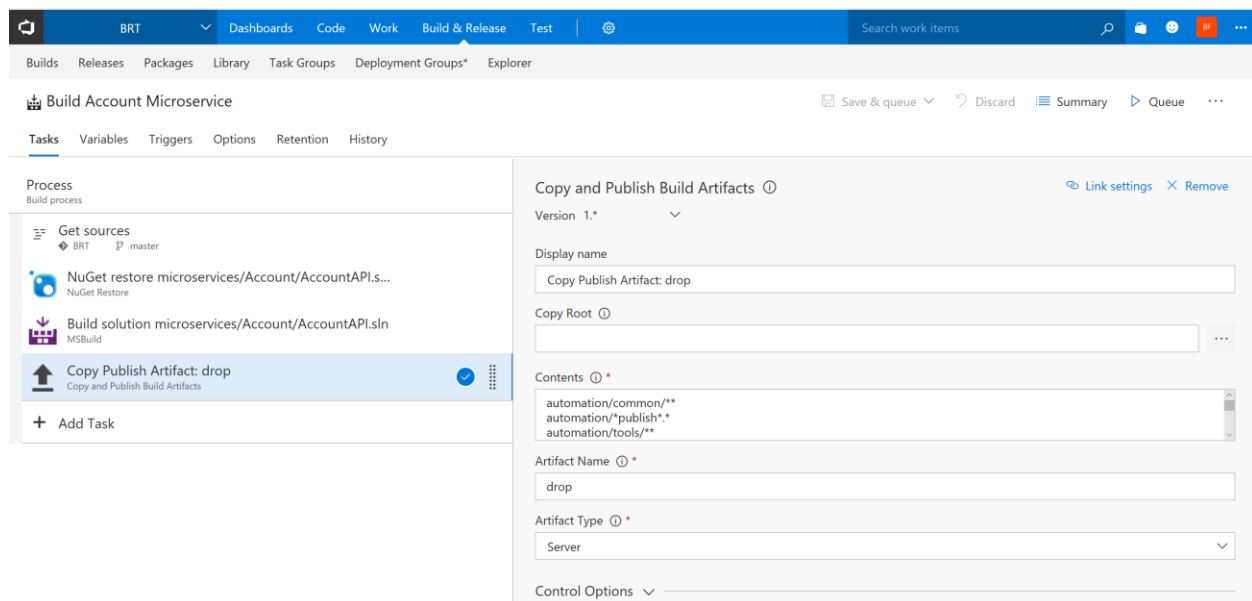
Chapter 2 Addendum

The exercises in Chapter 2 provide you an introduction to creating VSTS Build Definitions that provide provisioning and configuration of Azure services, build and publish of NuGet packages and the build and deployment of microservices that leverage Document DB, now called Cosmos DB.

This document provides an update to the build and deployment of microservices. Instead of the Build Definition handling all the tasks for build and deployment, the deployment steps are now handled by Release Definitions. The Release Definitions provide you the ability to define one or more target environments such as Dev, Test, Stage, Production, etc.

Build Definition Updates

Release Definitions can be triggered by successful Build Definitions. For example, upon the successful build of the Account Microservices, the Deploy Account Release Definition would commence. In order for there to be a smooth handoff between the Build Definition and the Release Definition, the build definition adds a step to copy build artifacts to the release definition host server.



The Contents provides a list of File or folder paths to include as part of the artifact. The format is multiple lines of minimatch patterns. Our build definitions need to copy scripts, templates and tools as well the output of the MSBuild process so that the release definition can invoke the PowerShell scripts that perform App Service provisioning, code deployment and App Settings configuration.

Here is the list of build artifacts that are being pushed to the release host server for the Account Microservice deployment:

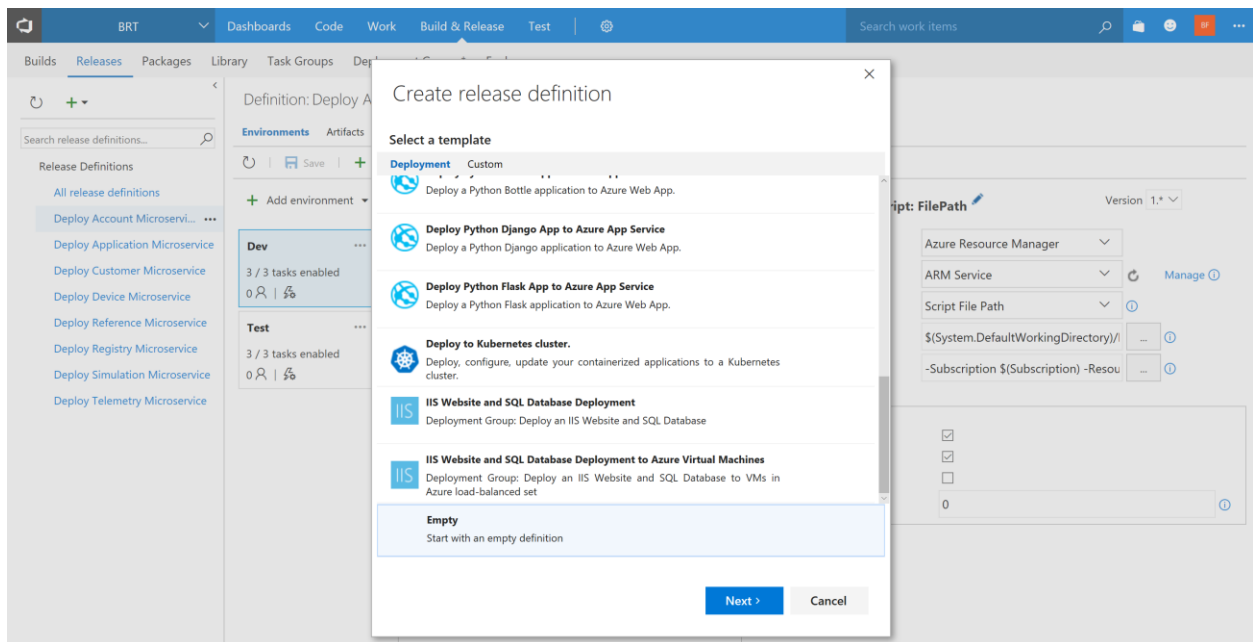
- automation/common/**
- automation/*publish*.*
- automation/tools/**
- automation/templates/**

- automation/deploy/**
- automation/*environment*.*
- automation/*.json
- microservices/account/AccountAPI/obj/Debug/Package/AccountAPI.zip

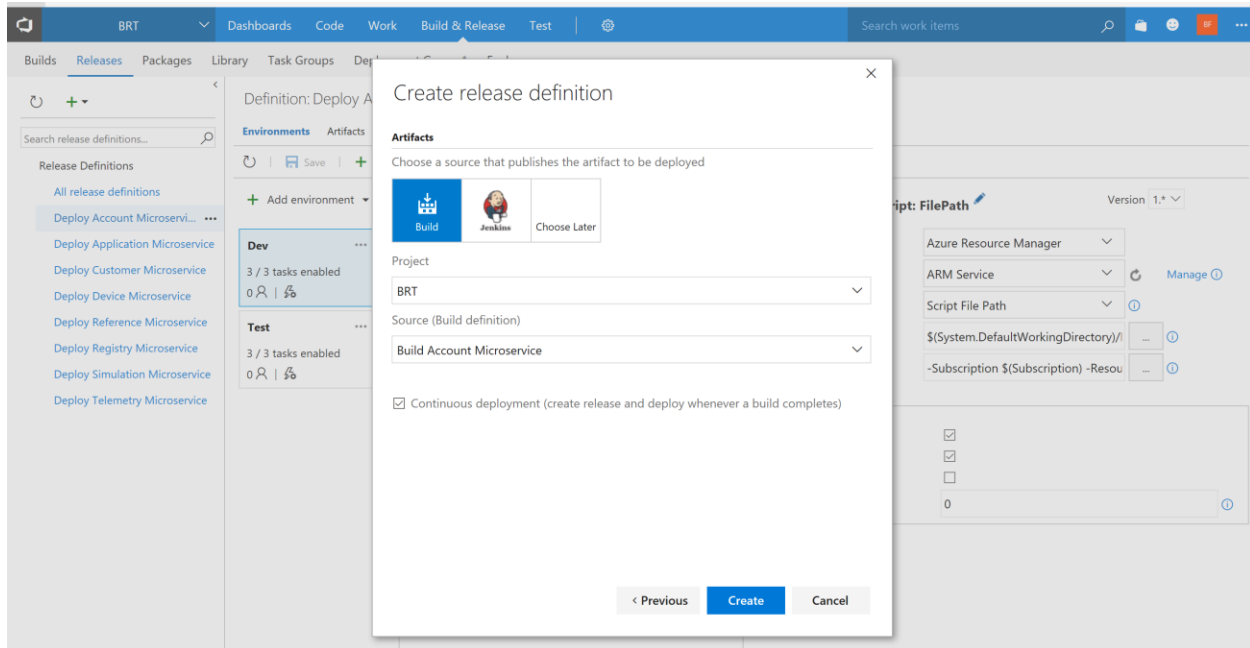
Copy Root is left blank to specify that paths in the Contents are off the root. The Artifact Name is also important as that name becomes part of the reference path used by the release definition tasks. I have set this to the term 'drop'.

Release Definitions

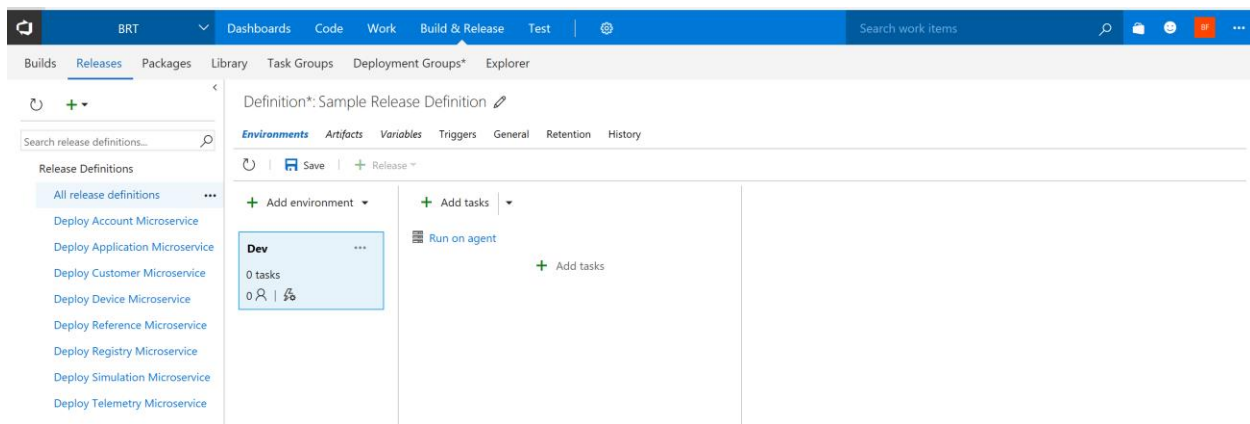
To create a Release Definition, click the '+' in the left-hand column and select 'Empty'.



Choose the source Build Definition from the drop down and check the box requesting Continuous Delivery.



You will be presented a Release Definition with one Environment. You can update the name of the Release Definition and change the name of the Environment. Each environment will contain a list of tasks to perform in order to deploy the output of the build definition to that environment.




The first step in defining your release definition is to define your variables. Variables are used to provide the meta data to tasks within each of the environment. You will have a set of global scoped variables, definition scoped variables and environment scoped variables.

Click 'Variables' in the tab bar above the environment list and then click the link at the end of the sentence 'Manage Variable Groups **here**'.

Definition*: Sample Release Definition 

[Environments](#) [Artifacts](#) **[Variables](#)** [Triggers](#) [General](#) [Retention](#) [History](#)

 |  Save |  Release ▾

Variable groups

Link variable groups to this release definition to include the variables they contain. Manage variable groups [here](#)

No variable groups are linked.

 Variable group

Variables

Define custom variables to use in this release definition. View list of [pre-defined variables](#)

Name	Value
 <input type="text"/>	<input type="text"/> 

+ Variable

A Variable Group is a list of globally scoped variables that will have the same values for all release definitions. For our releases, we will add the following global variables; Subscription, Azure Location, Prefix and ServicePlan. These variables are shared across all release definitions.

Library > Microservice Deploy Variables

Properties

Variable group name

Microservice Deploy Variables

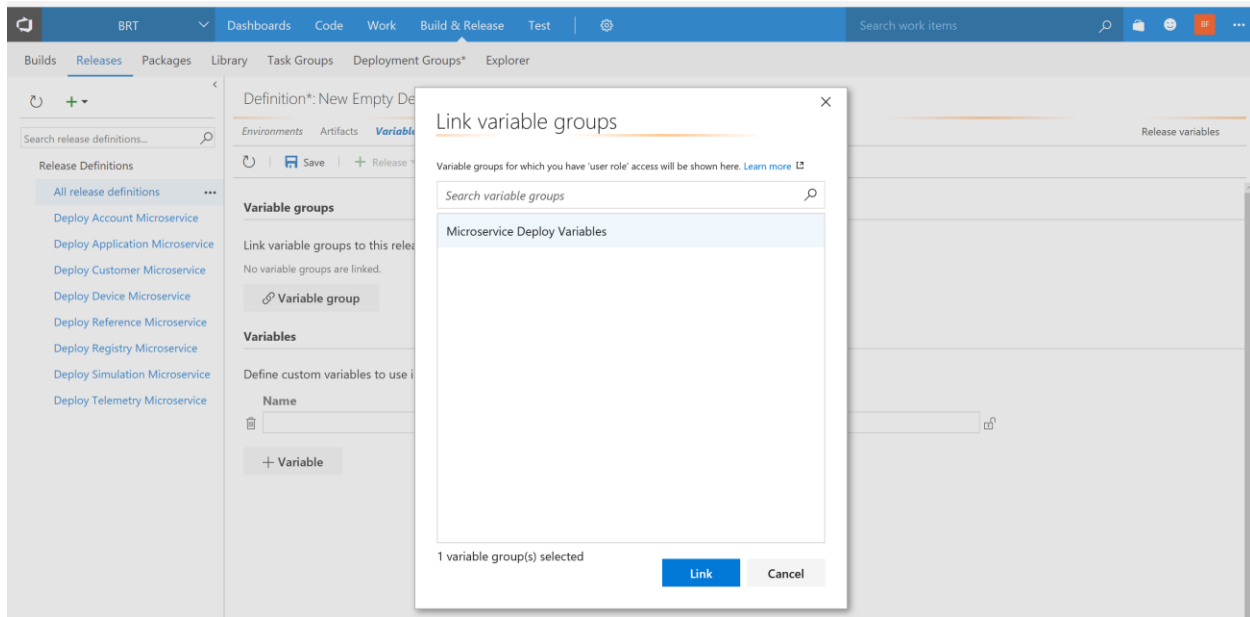
Description

Variables

Define the variables to include in this variable group.

Name	Value
Subscription	[your-subscription-name]
AzureLocation	East US
Prefix	brt
ServicePlan	AppServicePlan

Save your work here in the Library and go back to the Variables Tab. Click on the Variable Group button and choose the group you just defined. These variables are now linked to your definition.



Next, we need to define the variables that are scoped to the definition and the individual environments.

Add the following variables on this page. The ones that have blank values will be provided different values per environment giving you the ability to deploy to multiple environments in sequence.

Definition: Deploy Account Microservice | [Releases](#)

Environments Artifacts **Variables** Triggers General Retention History

| Save | Release ▼

Variable groups

Link variable groups to this release definition to include the variables they contain. Manage variable groups [here](#)

› Microservice Deploy Variables - 4 variables

Variable group

Variables

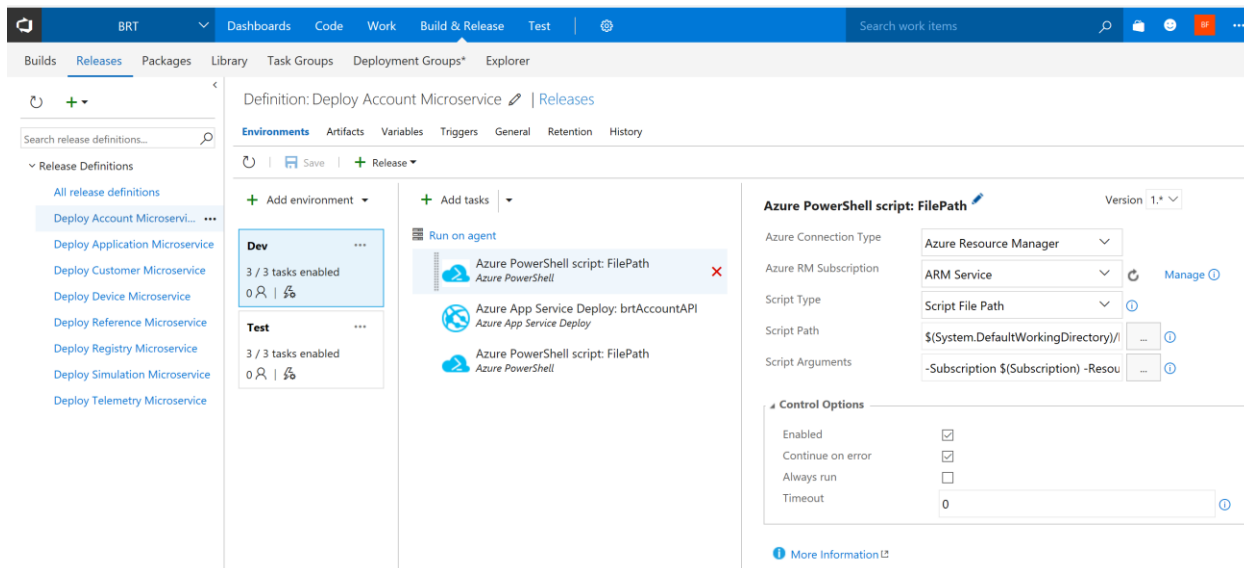
Define custom variables to use in this release definition. View list of [pre-defined variables](#)

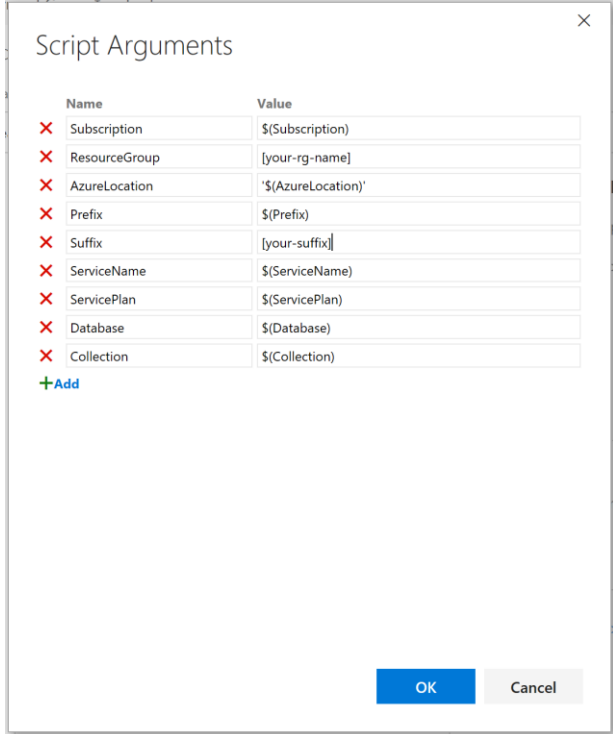
Name	Value
ResourceGroup	
Suffix	
ServiceName	AccountAPI
Database	Account
Collection	Subscription

Variable

Go back to the Environments tab.

For the development environment, add three tasks. These are the same tasks that were previously part of the Build Definition. They will have new properties now as they need to reference the build artifacts copied over from the build definition.



Azure PowerShell	Connection Type:	ARM
	Subscription:	<reference your RM Service hook>
	Script Type:	Script File Path
	Path:	\$(System.DefaultWorkingDirectory)/Build Account Microservice/drop/automation/05-Publish-AppService.ps1
	Arguments:	<p>Click the ellipse next to the arguments field and you will see how your pre-defined variable definitions are listed and that you have the option to fill out the Resource Group and Suffix fields which are unique to the Dev environment.</p> 

Definition*: Deploy Account Microservice [Releases](#)

Environments Artifacts Variables Triggers General Retention History

🔄 Save + Release

Add environment

Dev
3 / 3 tasks enabled

Test
3 / 3 tasks enabled

Add tasks

Run on agent

Azure PowerShell script: FilePath
Azure PowerShell

Azure App Service Deploy: [your-appservi
Azure App Service Deploy

Azure PowerShell script: FilePath
Azure PowerShell

Azure App Service Deploy: [your-appservice-name]

Version 3.*

Azure subscription ARM Service

App Service name [your-appservice-name]

Deploy to slot

Virtual application

Package or folder \$(System.DefaultWorkingDirectory)

File Transforms & Variable Substitution Options

Additional Deployment Options

Output

App Service URL

Control Options

Enabled

Continue on error

Always run

Timeout 0

App Service Deploy	Subscription	Select your RM service hook from the list
	App Service Name	Enter the name of the app service
	Package	\$(System.DefaultWorkingDirectory)/Build Account Microservice/drop/microservices/Account/AccountAPI/object/Debug/Package/AccountAPI.zip

Definition*: Deploy Account Microservice [Releases](#)

Environments Artifacts Variables Triggers General Retention History

🔄 Save + Release

Add environment

Dev
3 / 3 tasks enabled

Test
3 / 3 tasks enabled

Add tasks

Run on agent

Azure PowerShell script: FilePath
Azure PowerShell

Azure App Service Deploy: [your-appservi
Azure App Service Deploy

Azure PowerShell script: FilePath
Azure PowerShell

Azure PowerShell script: FilePath

Version 1.*

Azure Connection Type Azure Resource Manager

Azure RM Subscription ARM Service

Script Type Script File Path

Script Path \$(System.DefaultWorkingDirectory)

Script Arguments -Subscription \$(Subscription) -Resu

Control Options

Enabled

Continue on error

Always run

Timeout 0

Azure PowerShell	Connection Type:	ARM
	Subscription:	<reference your RM Service hook>
	Script Type:	Script File Path

	Path:	\$(System.DefaultWorkingDirectory)/Build Account Microservice/drop/automation/06-Publish- AppSettings.ps1
	Arguments:	<div> <div> Click the ellipse next to the arguments field and you will see how your pre-defined variable definitions are listed and that you have the option to fill out the Resource Group and Suffix fields which are unique to the Dev environment. </div> <div> </div> </div>

For the Test environment, repeat these steps but update the Resource Group and Suffix variables to target the Test environment.

Environments can be associated such that as one environment completes a successful deployment, you can trigger the next. To configure these settings, click the ellipse on the environment tile and choose 'Deployment Conditions'.

Configure - 'Test' environment

ApprovalsVariablesDeployment conditionsGeneral

Trigger

Define the trigger that will start deployment to this environment.

☐ No automated deployment

☐ After release creation

☒ After successful deployment to another environment

Trigger a new deployment on this environment after successful deployment on the selected environment.

Triggering environment(s)

Dev

☐ Scheduled

☒ Also trigger for partially succeeded deployment(s)

Options

Define behavior when multiple releases are waiting to be deployed on this environment. ⓘ

☒ Allow multiple releases to be deployed at the same time

☐ Allow only one active deployment at a time

OK

Cancel

Here is an example where the Test environment will automatically run when the Development environment has a successful deployment. This will keep these two environments synchronized. You can also set this to manual or scheduled.

Additional Updates

The original deployment scripts had an option to create the DocumentDB databases and collections as part of the deployment build definition. Those steps have been moved to the initial Provisioning build definition.