$projectName = 'ServerAutomationDemo' *## common term used through set up*

$subscriptionName = 'XXXXXXXXXX'

$subscriptionId = 'XXXXXXXX'

$tenantId = 'XXXXXXX'

$region = 'XXXXXXX'

$resourceGroupName = $projectName

$azDevOpsOrgName = 'adbertram'

$GitHubAccountName = $azDevOpsOrgName

$localVMAdminPw = 'I like azure.' *## a single password for demo purposes*

$gitHubRepoUrl = "<https://github.com/$GitHubAccountName/$projectName>"

*#Login and pick the right subscription*

az login

az account set --subscription $subscriptionName

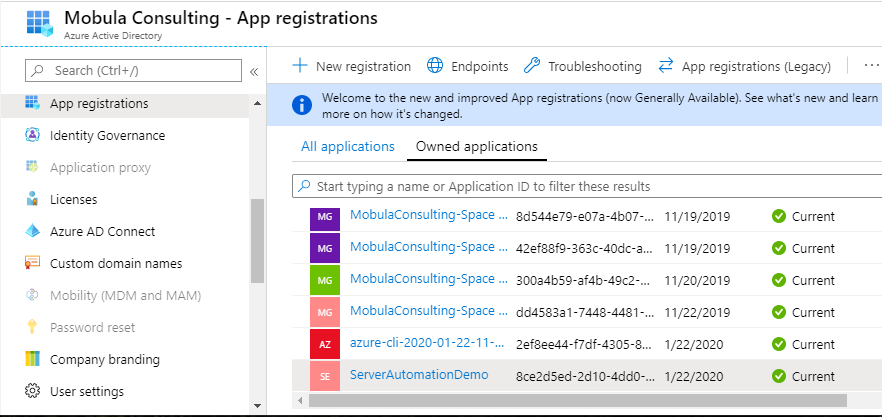
*#Set up a resource group for this demo*

az group create --location $region --name $resourceGroupName

*#Create a service principal - see screen shot for where this goes in the gui*

$spIdUri = "<http://$projectName>"

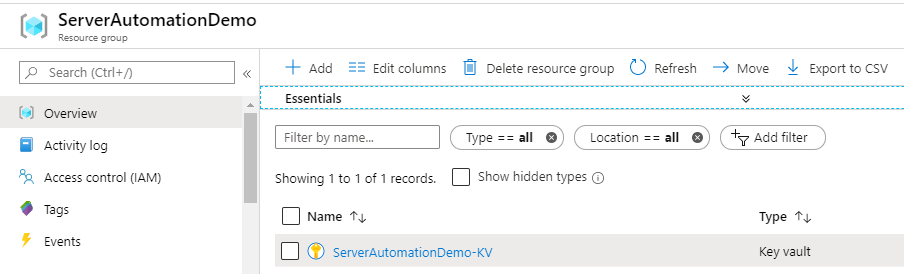
$sp = az ad sp create-for-rbac --name $spIdUri | ConvertFrom-Json



*#Create a keyvault in the the projects resource group*

$kvName = "$projectName-KV"

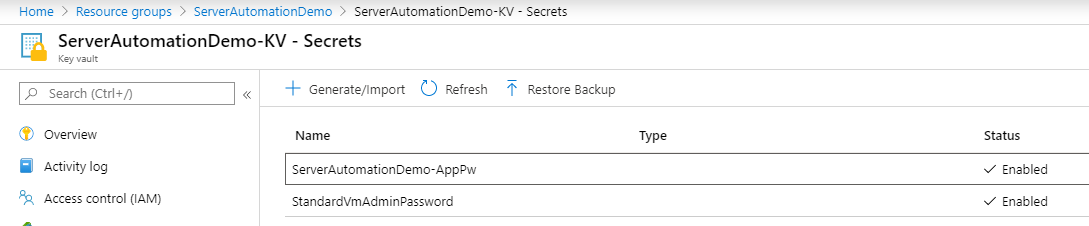
$keyVault = az keyvault create --location $region --name $kvName --resource-group $resourceGroupName   
 --enabled-for-template-deployment true | ConvertFrom-Json



*# ## Create the key vault secrets*

az keyvault secret set --name "$projectName-AppPw" --value $sp.password --vault-name $kvName

az keyvault secret set --name StandardVmAdminPassword --value $localVMAdminPw --vault-name $kvName

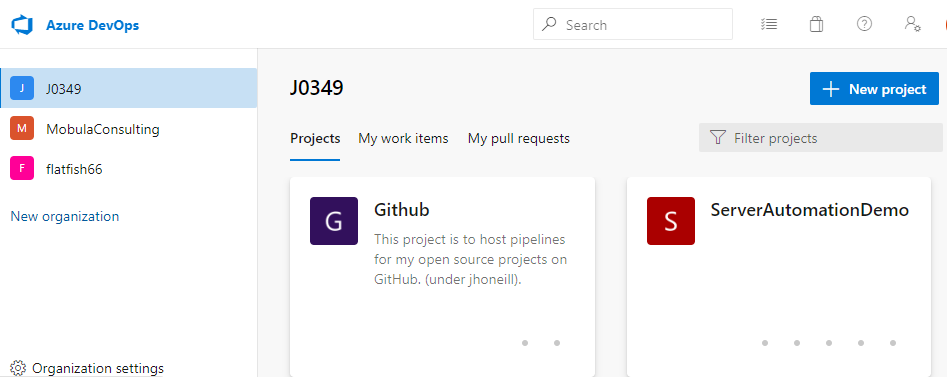


*#Create the Azure DevOps project*

az devops project create --name $projectName

az devops configure --defaults project=$projectName

#My AzDO organizations on the left, new AzDO project on the right



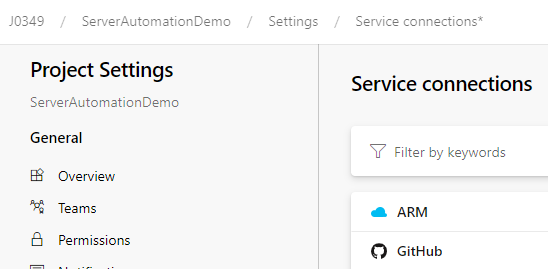
*#Create the Devops service connections*

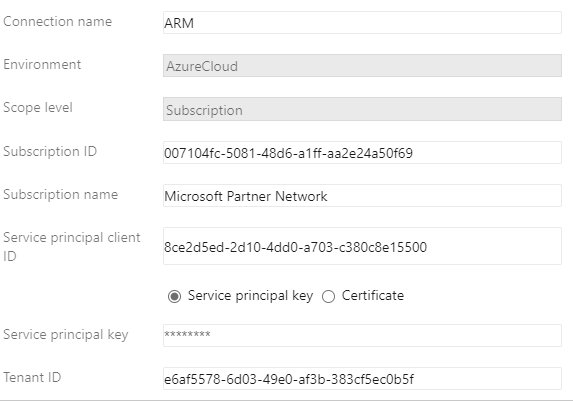
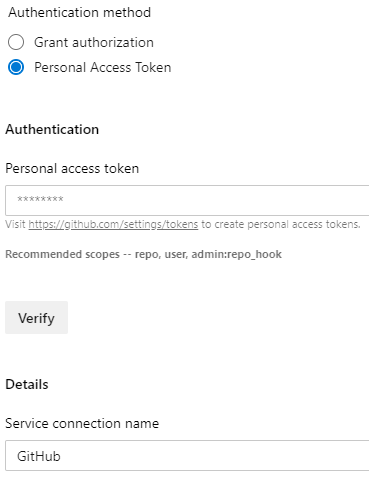
*## Run $sp.password and copy it to the clipboard*

*## when prompted, use the value of $sp.password for the Azure RM service principal key*

$sp.Password

az devops service-endpoint azurerm create --azure-rm-service-principal-id $sp.appId --azure-rm-subscription-id  
 $subscriptionId --azure-rm-subscription-name $subscriptionName --azure-rm-tenant-id $tenantId --name 'ARM'





*#we have a Service Principal in Azure Ad for our tenant/Subscription with an id of 8ce2d5ed-2d10-4dd0-a703-c380c8e15500 , [above] it can connect to our key vault to read the secrets The project has this as one service connection named ARM. It needs another service connect for connecting to GitHub [right]*

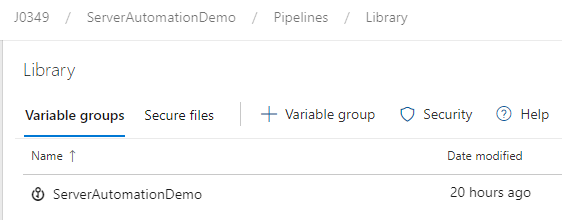
*#Get a GitHub access token (under developer settings) and Paste that in when asked.*

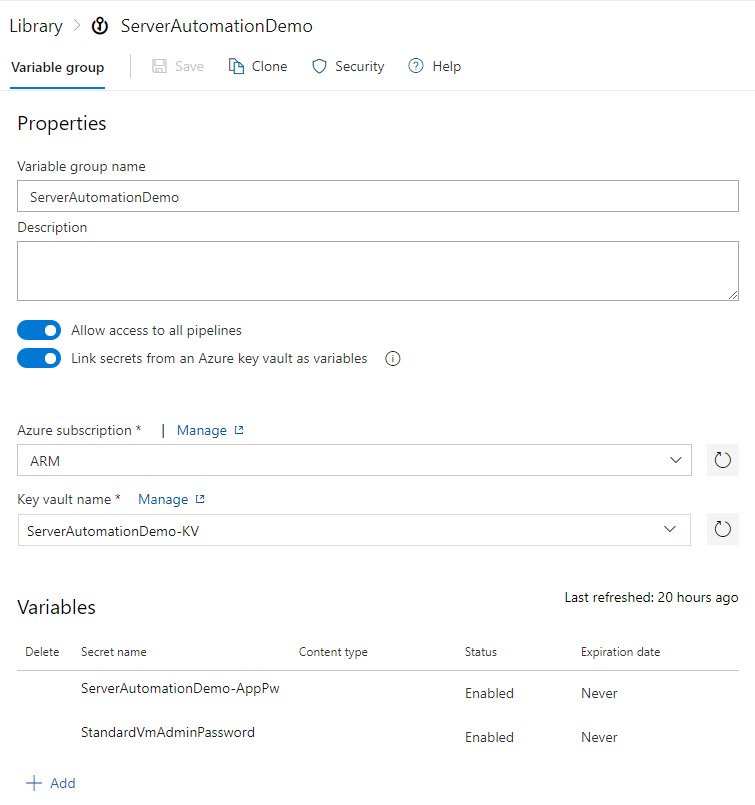
*$gitHubServiceEndpoint = az devops service-endpoint github create --github-url $gitHubRepoUrl --name 'GitHub' |  
 ConvertFrom-Json*

*##ARM, our service connection, gives us a connection to the Azure key vault as the service principal.*

*‘Now we need a variable group which to deliver the secrets into the pipeline.   
## It needs a dummy variable because AZ cli won't allow creation without it then MANUALLY link the key vault*

$varGroup = az pipelines variable-group create --name $projectName --authorize true --variables foo=bar |   
 ConvertFrom-Json



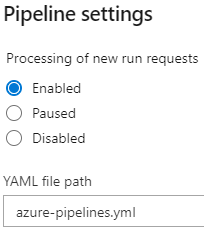


*# Add the pipeline*

az pipelines create --name $projectName --repository $gitHubRepoUrl --branch master --service-connection $gitHubServiceEndpoint.id --skip-run

trigger: *# what in the repo will trigger the pipeline*

branches:

 include:

- master

paths:

include:

- server.json

- server.parameters.json

pool:

vmImage: "ubuntu-latest"

variables:

- group: ServerAutomationDemo

- name: **azure\_resource\_group\_name** *# resource group will have a suffix -123 based on build no.*

value: "ServerProvisionTesting-$(Build.BuildId)"

- name: **subscription\_id** *# The others Could be secrets.*

value: "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

- name: **application\_id**

value: "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

- name: **tenant\_id**

value: "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

steps:

- task: PowerShell@2

inputs:

filePath: "connect-azure.ps1"

arguments: '-ServicePrincipalPassword "***$(ServerAutomationDemo-AppPw)***" -SubscriptionId $(subscription\_id)   
 -ApplicationId $(**application\_id**) -TenantId $(**tenant\_id**)'

- task: PowerShell@2

inputs:

targetType: "inline"

script: New-AzResourceGroup -Name $(**azure\_resource\_group\_name**) -Location uksouth -Force

- task: AzureResourceManagerTemplateDeployment@3

deploymentScope: "Resource Group"

azureResourceManagerConnection: "ARM"

subscriptionId: "$(subscription\_id)"

action: "Create Or Update Resource Group"

resourceGroupName: **$(azure\_resource\_group\_name)**

location: "UK South"

templateLocation: "Linked artifact"

csmFile: "server.json"

csmParametersFile: "server.parameters.json"

deploymentMode: "Incremental"

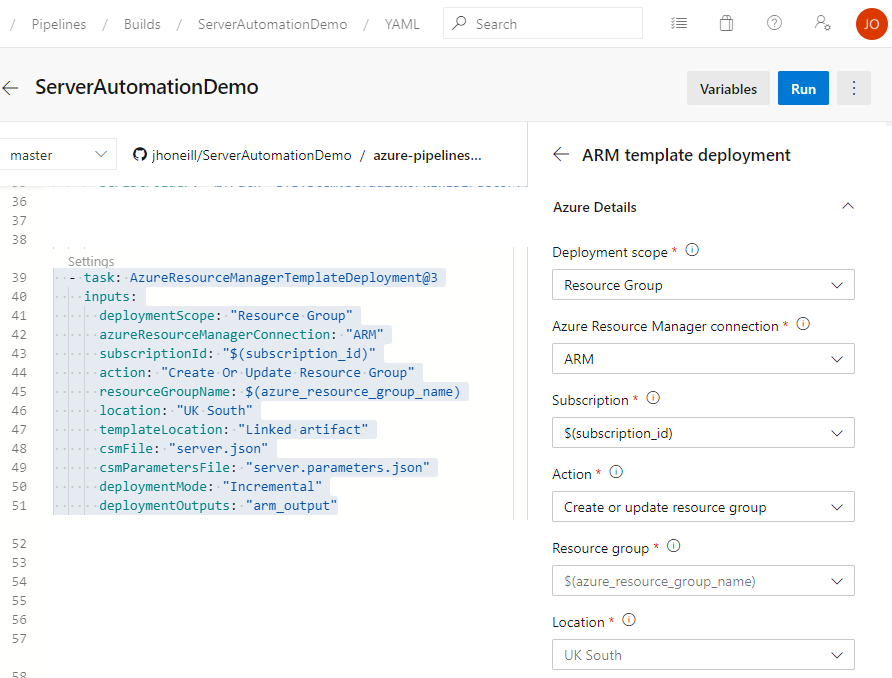
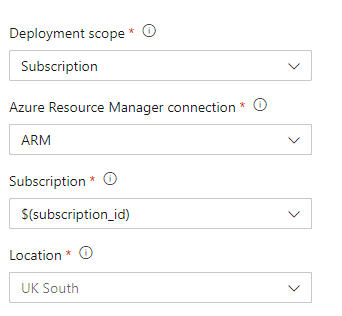
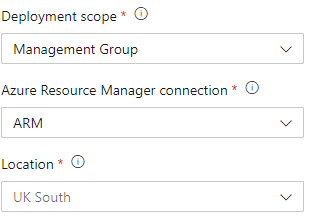
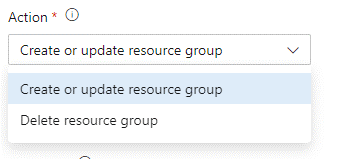
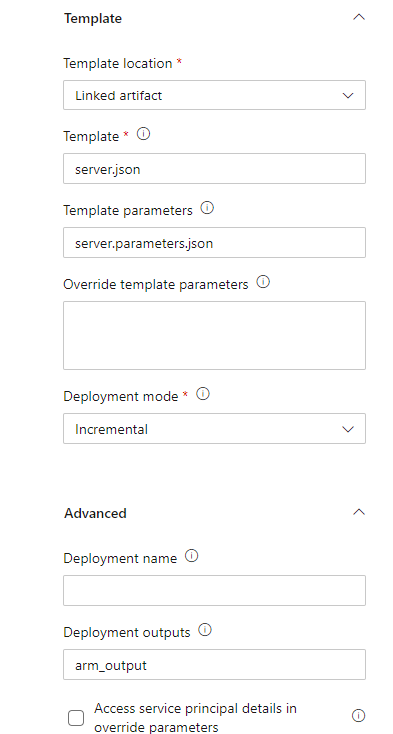
deploymentOutputs: "arm\_output"

First task does   
 Install-Module -Name Az.Accounts, Az.Resources -Force -SkipPublisherCheck

Connect-AzAccount -ServicePrincipal -Tennant <<idfromvar>> -SubscriptionId <<idfromvar>>  
 -Credential <<from AppidVar & passwordsecret from library>>

Second task creates resource group

Third task looks like this in the portal

Now split the pipeline

trigger:...

stages:

- stage: Validate

pool:

vmImage: "ubuntu-latest"

jobs:

- job: vt

displayName: "Validate template"

steps:

- task: Pester@0

displayName: "Check Template syntax"

inputs: ...

- task: PublishTestResults@2

displayName: "Publish syntax check result"

inputs:...

- template: deployment.template.yml

parameters:

azure\_resource\_group\_name: "ServerProvisionTesting-$(Build.BuildId)"

subscription\_id: "007104fc-5081-48d6-a1ff-aa2e24a50f69"

application\_id: "8ce2d5ed-2d10-4dd0-a703-c380c8e15500"

tenant\_id: "e6af5578-6d03-49e0-af3b-383cf5ec0b5f"

variableGroup: ServerAutomationDemo

dependOn: Validate

environment: demo

*Validate* is just a demo pre-deployment stage, it runs a pester test (using an Add in) named *Check Template synta*x to validate the JSON text and then publishes the result.   
The next stage in the multi-stage Pipeline uses a template which is given the tenant, subscription and service account name, plus the variable library name, and is told to make sure Validate ran and use “Demo” as an environment name. The environment is just a place holder for approvals. We can deploy into multiple environments in sequence or have other pipelines which use the same template in other contexts (e.g. deploy hotfix to production).

parameters:

azure\_resource\_group\_name:

subscription\_id:

application\_id:

tenant\_id:

variableGroup:

dependOn:

environment:

stages:

- stage: Deploy

dependsOn: ${{ parameters.dependOn }}

variables:

- group: ${{ parameters.variableGroup }}

pool:

vmImage: "ubuntu-latest"

jobs:

- deployment: "Deploy"

displayName: 'Deploy Resource Group'

environment:

name: ${{ parameters.environment }}

strategy:

runOnce:

deploy:

steps:

- checkout: self

displayName: "Download Repo"

- task: PowerShell@2

displayName: "Connect to Azure"

inputs:...

- task: Pester@0 ...

- task: PublishTestResults@2 ...

- task: PowerShell@2 ...

- task: AzureResourceManagerTemplateDeployment@3

inputs:

deploymentScope: "Resource Group"

azureResourceManagerConnection: "ARM"

subscriptionId: ${{ parameters.subscription\_id }}

action: "Create Or Update Resource Group"

resourceGroupName: ${{ parameters.azure\_resource\_group\_name }}

location: "UK South"

templateLocation: "Linked artifact"

csmFile: "server.json"

csmParametersFile: "server.parameters.json"

deploymentMode: "Incremental"

deploymentOutputs: "arm\_output"

- task: Pester@0 ...

- task: PublishTestResults@2 ...

- task: PowerShell@2

displayName: "Post Installation steps"

inputs:

targetType: filePath

filePath: "config-vm.ps1"

arguments: '-rgname "${{ parameters.azure\_resource\_group\_name }}"'

- task: PublishPipelineArtifact@1 ...