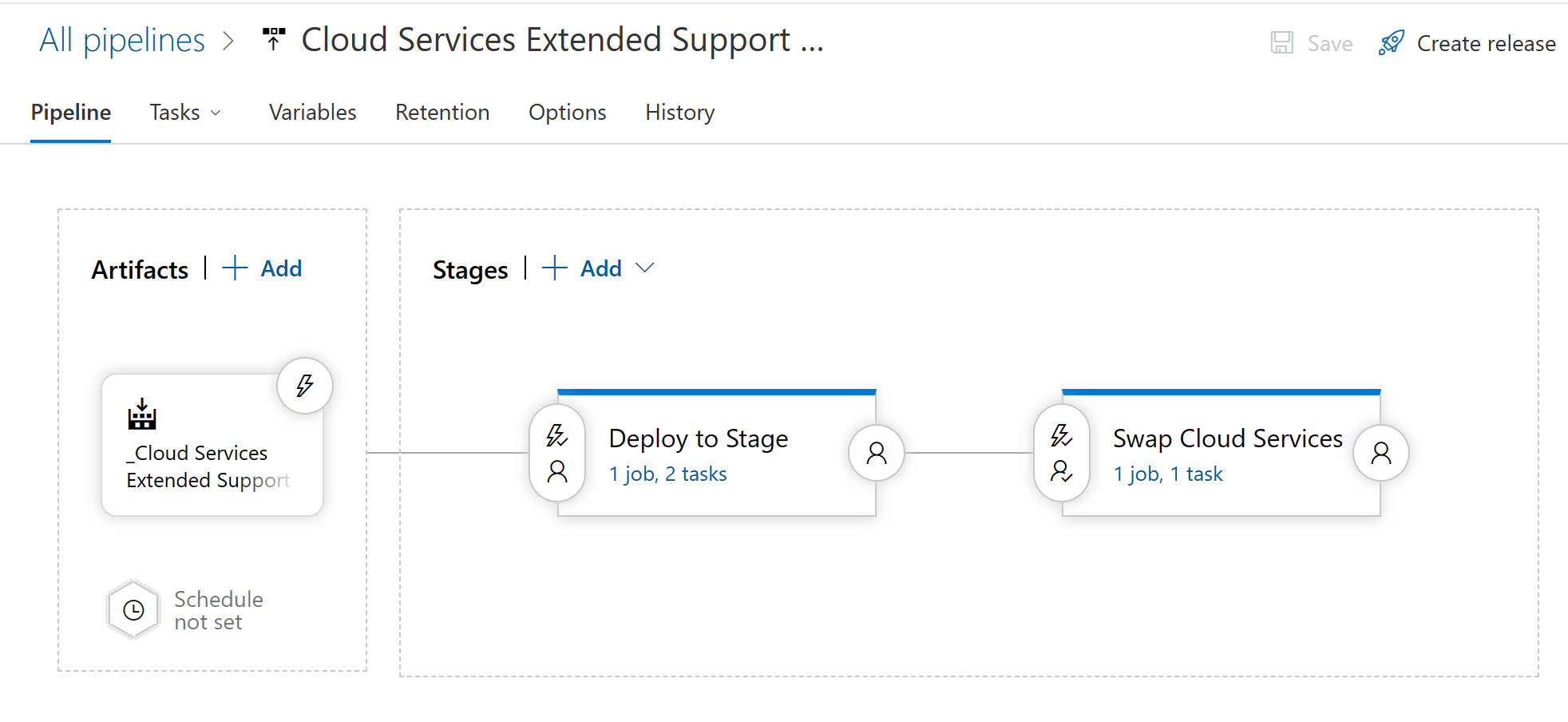
**Overview**

The following sample Azure DevOps Release Pipeline demonstrates how to always ensure that code is being deployed to the staging instance of your Cloud Services Extended Support Swappable Cloud Services and provides a pipeline stage to perform the actual swap once a pre-deployment approval is provided. Please note that the trick to making this fully automated is making sure you can identify which instance of your paired CSES instances is the production instance and which is the staging instance. This allows the pipeline to identify the correct cloud service to deploy the new package too. I accomplish this by assigning a DNS name to the public IP address of the load balancer of each cloud service indicating its role in the pair. For example, nimccfta-cses-stage.eastus.cloudapp.azure.com and nimccfta-cses-prod.eastus.cloudapp.azure.com. Since the swap operation simply does a VIP swap, the appropriate load balancer is always pointing at the appropriate deployment and can be used to identify the current production cloud service and current staging cloud service.

**Pipeline Overview**



**Variables**

A screenshot of a computer

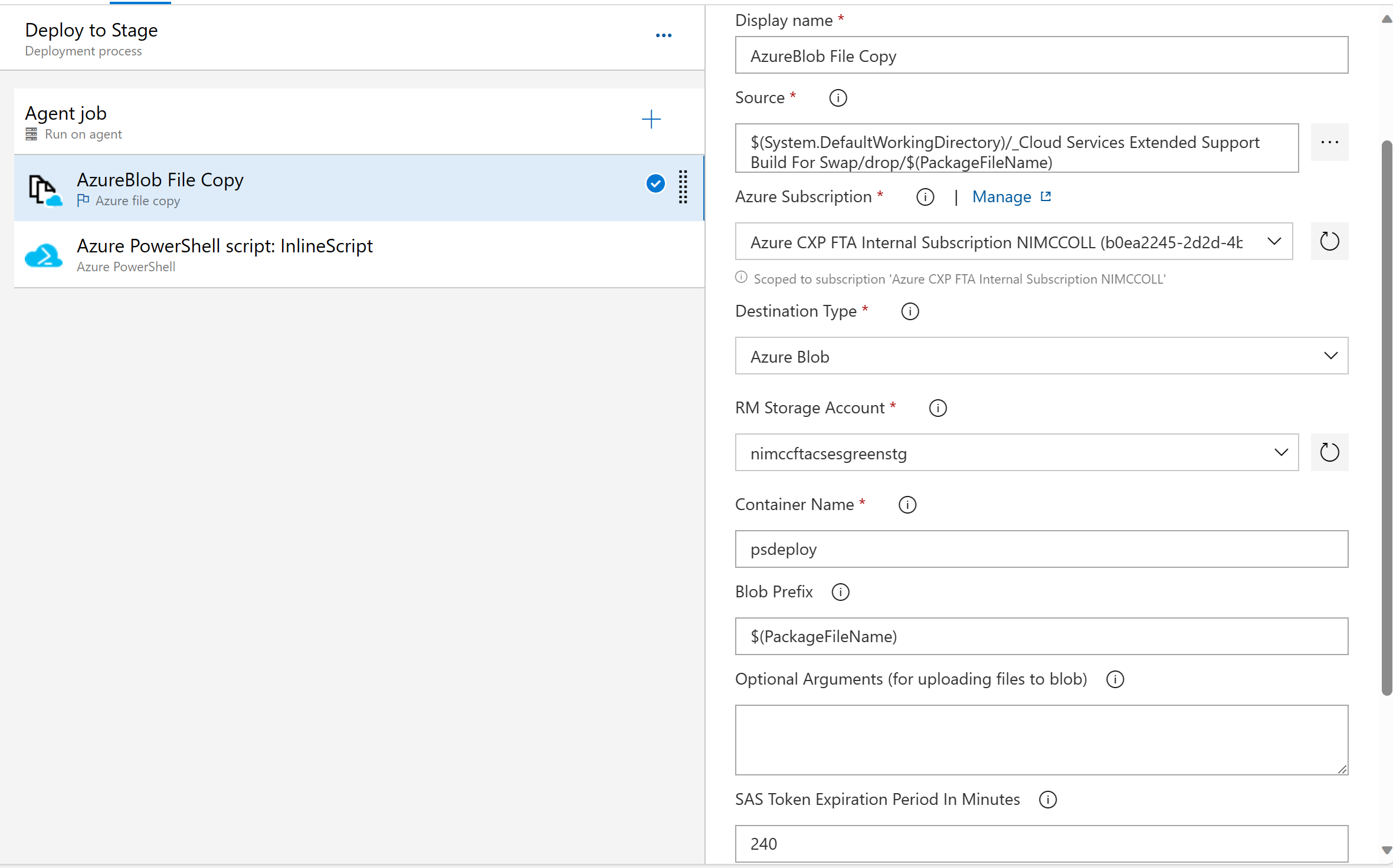
Description automatically generated with medium confidence

**Deploy to Stage Overview**

A screenshot of a computer

Description automatically generated with medium confidence

**Deploy To Stage – AzureBlob File Copy**



**Deploy To Stage – Azure PowerShell script: InlineScript**

A screenshot of a computer

Description automatically generated with medium confidence

**Full PowerShell Script**

# Retrieve both cloud services

$cloudService1 = Get-AzCloudService -ResourceGroupName $(ResourceGroupName) -CloudServiceName $(CloudService1Name)

$cloudService2 = Get-AzCloudService -ResourceGroupName $(ResourceGroupName) -CloudServiceName $(CloudService2Name)

# Get the public IP address of each cloud service

$pipResource1 = Get-AzResource -ResourceId $cloudService1.NetworkProfile.LoadBalancerConfiguration[0].FrontendIPConfiguration[0].PublicIPAddressId

$pipResource2 = Get-AzResource -ResourceId $cloudService2.NetworkProfile.LoadBalancerConfiguration[0].FrontendIPConfiguration[0].PublicIPAddressId

$csIPAddress1 = Get-AzPublicIPAddress -Name $pipResource1.Name -ResourceGroupName $pipResource1.ResourceGroupName

$csIPAddress2 = Get-AzPublicIPAddress -Name $pipResource2.Name -ResourceGroupName $pipResource2.ResourceGroupName

# Determine which cloud service is the staging cloud service based on the DomainNameLabel of the public IP address

if ($csIPAddress1.DnsSettings.DomainNameLabel.Contains('stage'))

{

$cloudService = $cloudService1

}

elseif ($csIPAddress2.DnsSettings.DomainNameLabel.Contains('stage'))

{

$cloudService = $cloudService2

}

# Preserve the virtual network name of the cloud service we are deploying to

$xml = [xml]$cloudService.Configuration

$networkName = $xml.ServiceConfiguration.NetworkConfiguration.VirtualNetworkSite.name

# Preserve the diagnostics connection string if it exists

if ($xml.ServiceConfiguration.Role.ConfigurationSettings.Setting.name.Contains('Diagnostics.ConnectionString'))

{

$diagConnectionString = $xml.ServiceConfiguration.Role.ConfigurationSettings.Setting.value

}

# Get the URL of the package file

$cspkgUrl = '$(BC.StorageContainerUri)$(PackageFileName)$(BC.StorageContainerSasToken)'

# Retrieve the contents of the configuration file

$configuration = Get-Content -Path '$(System.DefaultWorkingDirectory)/\_Cloud Services Extended Support Build For Swap/drop/ServiceConfiguration.Cloud.cscfg' | Out-String

Write-Output $configuration

$xmlConfig = [xml]$configuration

$xmlConfig.ServiceConfiguration.NetworkConfiguration.VirtualNetworkSite.name = $networkName

$xmlConfig.ServiceConfiguration.Role.ConfigurationSettings.Setting.value = $diagConnectionString

$xmlConfig.PreserveWhiteSpace = $true

$configuration = $xmlConfig.OuterXml

Write-Output $configuration

# Update the cloud service

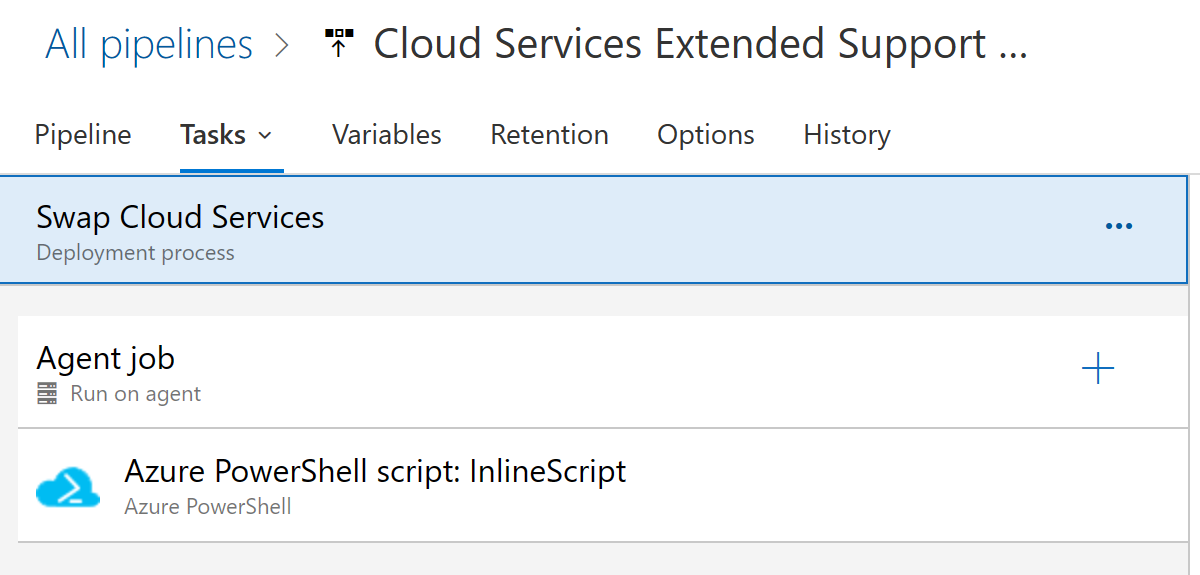
$cloudService.Configuration = $configuration

$cloudService.PackageUrl = $cspkgUrl

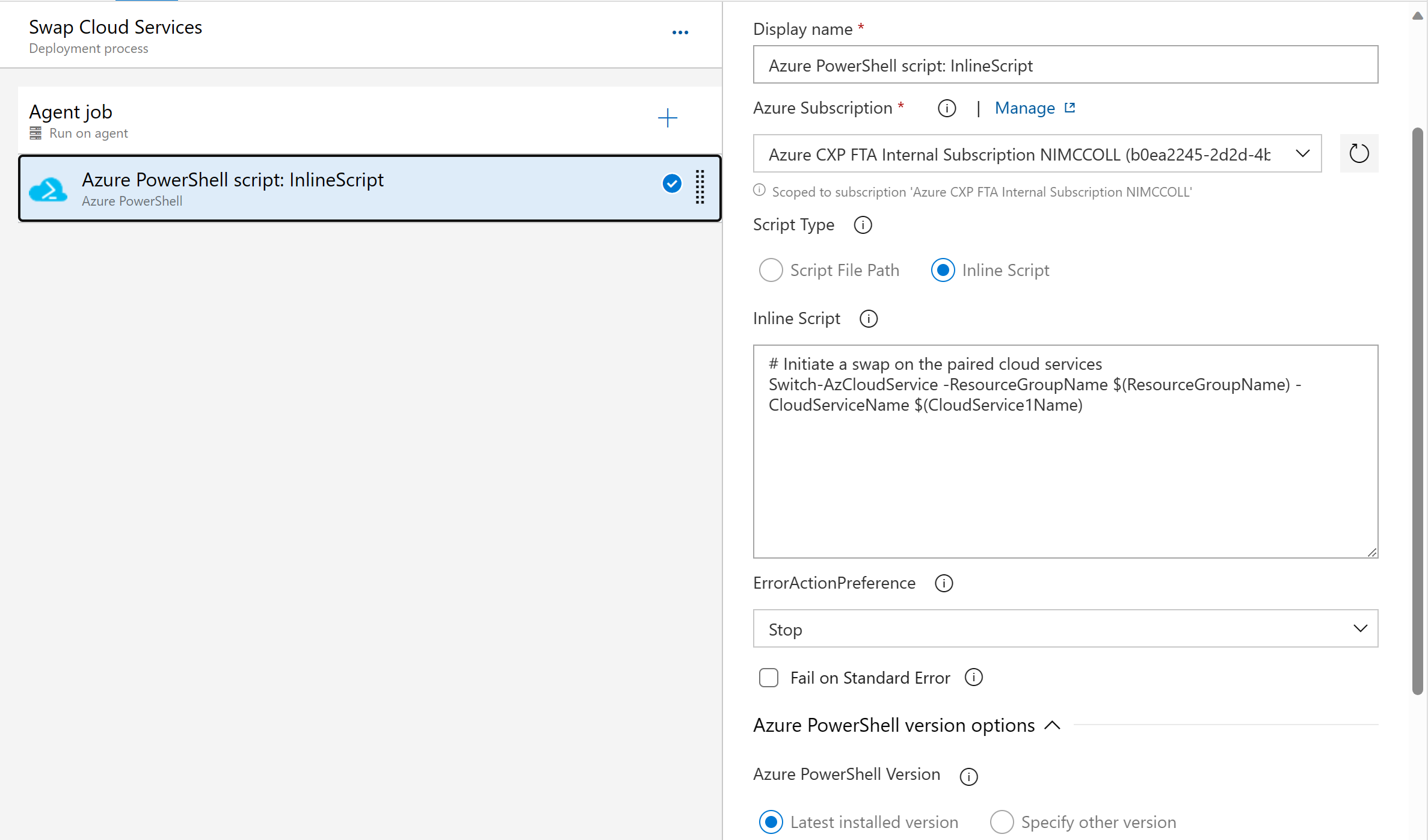
$cloudService | Update-AzCloudService

Write-Output $cloudService.Name

**Swap Cloud Services Overview**



**Swap Cloud Services – Azure PowerShell script: InlineScript**



**Full PowerShell Script**

# Initiate a swap on the paired cloud services

Switch-AzCloudService -ResourceGroupName $(ResourceGroupName) -CloudServiceName $(CloudService1Name) -Confirm:$false