

Date [2020-07-28] Tags [[Azure DevOps](#)]

New variable

Name

pipelineScopedSecretVar

Value

.....|

☒ Keep this value secret

☐ Let users override this value when running this pipeline

To reference a variable in YAML, prefix it with a dollar sign and enclose it in parentheses. For example:

`$(pipelineScopedSecretVar)`

To use a variable in a script, use environment variable syntax. Replace `.` and space with `_`, capitalize the letters, and then use your platform's syntax for referencing an environment variable. Examples:

Batch script: `%PIPELINESCOPEDSECRETVAR%`


PowerShell script: `$env:PIPELINESCOPEDSECRETVAR`

Bash script: `$PIPELINESCOPEDSECRETVAR`

To use a secret variable in a script, you must explicitly map it as an environment variable.

Secrets that are used by more than one pipeline can be added to a variable group:

Toggle this button to link an Azure key vault and map selective vault secrets to this variable group.

[Learn more](#) 

[+ Add](#)

Creating a secret variable in a variable group

Variable groups can also be [linked to an Azure Key Vault](#). This is often a good idea, as it allows secret variables to be maintained separately from our pipelines by using tools other than Azure DevOps. It also means that the secret variables can be *retrieved* from the Key Vault UI given appropriate access policies on the vault.

For secrets created in the Azure DevOps UI, whether pipeline-scoped or in a variable group, it is not so simple to retrieve the variables after creation. This might be required for a number of reasons, most often troubleshooting. The need to do this is often an indicator that the project should have been using an Azure Key Vault in the first place.

Previously it was necessary to [jump through some hoops](#) to access secret variables, but it turns out this is no longer required. It also appears the recommended approach of mapping secrets to environment variables is [currently not working](#) for secret variables from variable groups.

However, as long as we are using an inline shell script rather than calling an external file from our script step, the solution turns out to be rather simple; the value of the secret variables can be replaced in the pipeline by [macro expansion](#). Since secrets are masked in the log output, they need to be written to a file or extracted by some other mechanism during the build.

The full pipeline to read the secret variables and write them to a file appears below. The file containing the secrets is published as a pipeline artifact so that it can be retrieved after the build has finished. There is a published Azure DevOps project [here](#) containing this pipeline.

```
trigger:
- none

pool:
  vmImage: 'ubuntu-latest'

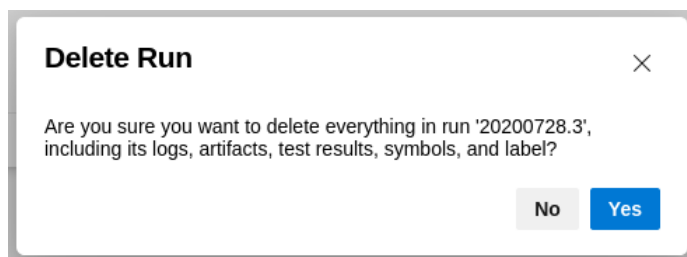
variables:
- group: VariableGroupContainingSecretVar

steps:
- task: Bash@3

  inputs:
    targetType: 'inline'
    script: |
      # Write your commands here
      echo "pipelineScopedSecretVar = $(pipelineScopedSecretVar)" >> $PIPELINE_WORKSPACE/secrets.txt
      echo "secretVarInVariableGroup = $(secretVarInVariableGroup)" >> $PIPELINE_WORKSPACE/secrets

- task: PublishPipelineArtifact@1
  inputs:
    targetPath: '$(Pipeline.Workspace)/secrets.txt'
    artifact: 'Secrets'
    publishLocation: 'pipeline'
```

If you have to do this in “real life”, remember to delete the build afterwards so that the artifact disappears.



Deleting the build to remove the sensitive artifact


There are a couple of takeaways here; firstly that we should use Key Vault backed variable groups whenever we can in order to avoid this problem in the first place, and secondly that we need to be careful which pipelines have *access* to variable groups containing secrets.

0 Comments - powered by utteranc.es

Write

Preview

Sign in to comment

 Styling with Markdown is supported

Sign in with GitHub

RSS 

© 2020 — Gavin Campbell. Except as otherwise noted, content on this website is licensed under a [Creative Commons Attribution 4.0 International License](#).
Built with the [Slick](#) theme for [Hugo](#), hosted on [Netlify](#).