

GitHub Copilot

Power user example



Use case

- GitHub workflow file with inline PowerShell script
- Task: extract the script to a new file and call it with environment variables for passing in `${{ vars.variable }}`



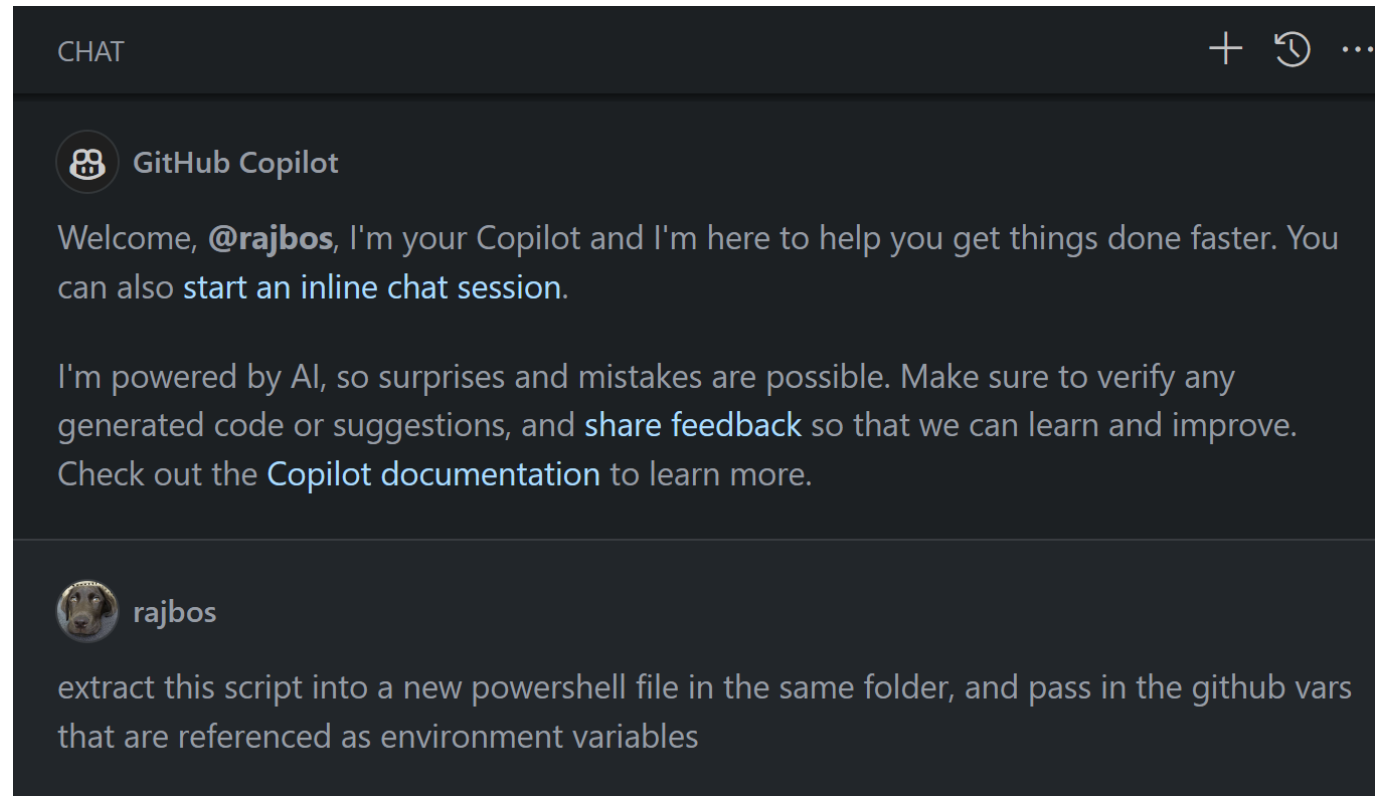
Step 1: Select the code

- This tells Copilot what you want to work with

```
github > workflows > ! deploy-infra.yml > {} jobs > {} deploy-infra > [ ] steps > {} 1 > run
  9   jobs:
 10     deploy-infra:
 15       steps:
 16         - name: Azure Login
 20
 21         - name: Deploy resources in the resource group
 22           run: |
 23             # pwsh
 24             # deploy into resource group
 25             $azureregion = "northeurope"
 26
 27             # check if the resource group exists
 28             $resourceGroupName = "${{ vars.RESOURCE_GROUP_NAME }}"
 29             $resourceGroup = az group show --name $resourceGroupName --query name --output tsv
 30             if ($null -eq $resourceGroup) {
 31               throw "Resourcegroup [$resourceGroupName] not found"
 32             }
 33             Write-Host "Resourcegroup [$resourceGroupName] found!"
 34
 35             # create a keyvault
 36             $keyvaultName = "${{ vars.KEYVAULT_NAME }}"
 37             $keyvault = az keyvault show --name $keyvaultName --resource-group $resourceGroupName
 38             if ($null -eq $keyvault) {
 39               Write-Host "Creating a Keyvault [$keyvaultName]..."
 40               az keyvault create --name $keyvaultName --resource-group $resourceGroupName --location $azureregion
 41               Write-Host "Keyvault [$keyvaultName] created!"
 42             } else {
 43               Write-Host "Keyvault [$keyvaultName] found!"
 44             }
 45
```

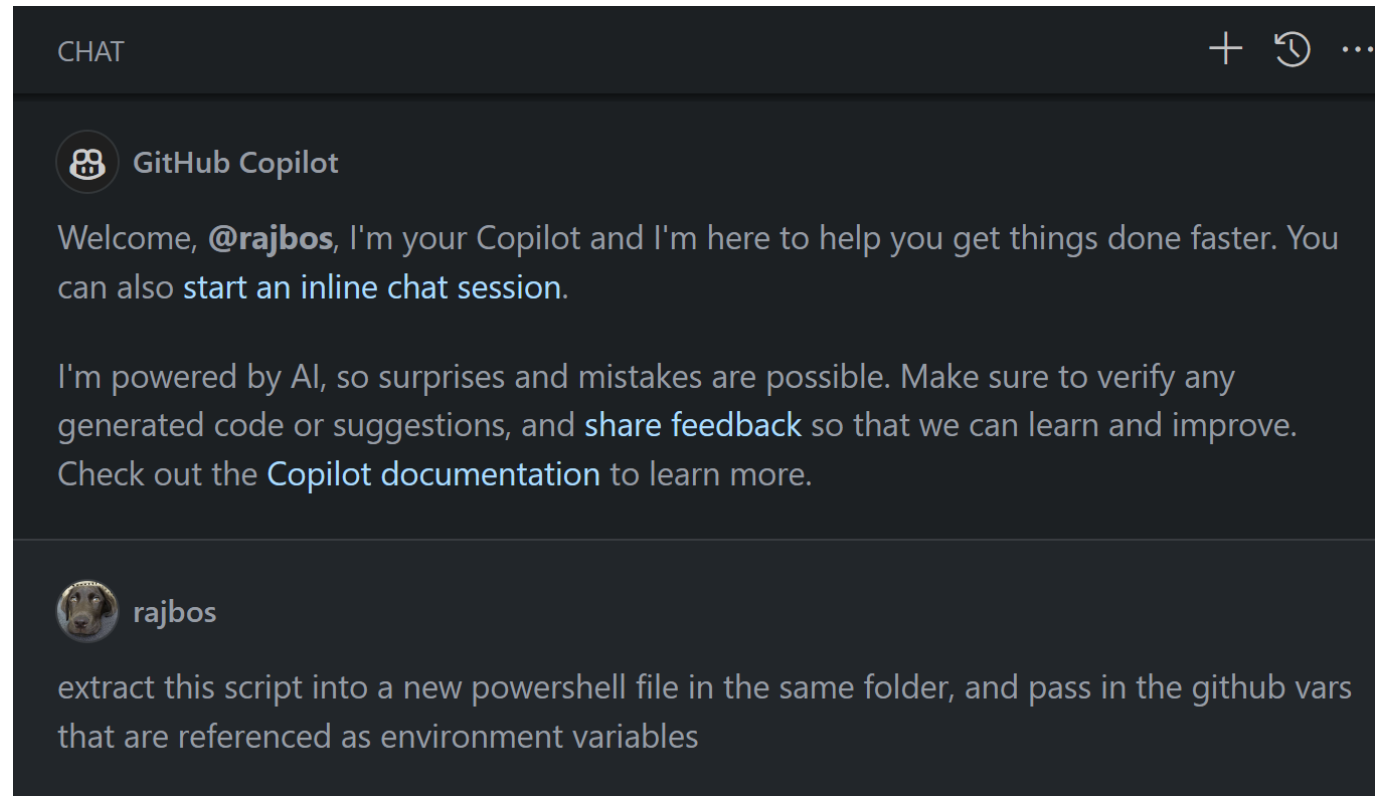
Step 2: Craft a prompt

- This tells Copilot what you want to achieve



Step 3: Let GitHub Copilot generate the file

- Instead of copy and pasting the script block yourself, let Copilot generate it, and also convert the in place string injection with `${vars.variable}` into a `$env:variable` setup



Step 4: The script is created for you

- Saves a lot of work and editing with all the environment variables!



rajbos

extract this script into a new powershell file in the same folder, and pass in the github vars that are referenced as environment variables



GitHub Copilot



Used 1 reference >

Sure, here's how you can extract this script into a new PowerShell file and pass the GitHub variables as environment variables.

First, create a new PowerShell file, let's call it `deploy-infra.ps1`. Then, copy the script into this file. Replace the `{{ vars.VARIABLE_NAME }}` and `{{ secrets.SECRET_NAME }}` syntax with `$env:VARIABLE_NAME` and `$env:SECRET_NAME` respectively. This will allow PowerShell to access these values as environment variables.

Here's the modified script:

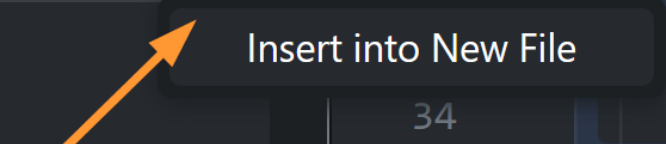
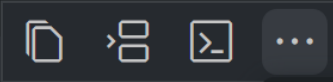
```
# deploy into resource group
$azureregion = "northeurope"

# check if the resource group exists
$resourceGroupName = $env:RESOURCE_GROUP_NAME
$resourceGroup = az group show --name $resourceGroupName --query name --output tsv
if ($null -eq $resourceGroup) {
    throw "Resourcegroup [$resourceGroupName] not found"
}
Write-Host "Resourcegroup [$resourceGroupName] found!"
```

Here's the modified script:

```
# deploy into resource group
$azureregion = "northeurope"

# check if the resource group exists
$resourceGroupName = $env:RESOURCE_GROUP_NAME
$resourceGroup = az group show --name $resourceGroupName --query name
if ($null -eq $resourceGroup) {
    throw "Resourcegroup [$resourceGroupName] not found"
}
```



Step 5: create a new file from the suggestion in one click

Of course you need to name the file, but that is it!

Step 6:

- Validate that the script still does the same steps!

This is crucial for anything you do with a Large Language Model (LLM).

Always validate, test, lint, run security tests on everything you do!

Step 7:

- Check how GitHub Copilot suggests you call the script.
- Notice that the conversion to from `{{ vars.variable }}` has not been added!

```
# add the storage connectionstrings to the app service
Write-Host "Adding the Storage Connection Strings to the Web App..."
$storageAccountName = $env:AZURE_STORAGE_ACCOUNT_NAME
$storageAccountKey = (az storage account keys list --resource-group $resourceGroupName --account-name $s
$connectionString = "DefaultEndpointsProtocol=https;AccountName=$storageAccountName;AccountKey=$storageA
az webapp config connection-string set --name $webAppName --resource-group $resourceGroupName --settings
```

To run this script with the environment variables, you can use the following command:

```
GH_TOKEN=your_gh_private_key GH_WEBHOOK_SECRET=your_gh_webhook_secret powershell -File ./deploy-infra.ps1
```

Replace `your_variable_name` with your actual values.

Step 8:

- Try to fix that with a different prompt.
- Notice I had the new file selected, so the result was not what I had in mind



rajbos

convert the usage of `${{ vars.variablename }}` in the original GitHub Actions workflow into the calling of the new PowerShell script



GitHub Copilot

Used 1 reference ▾

➤ create-infra.ps1:43-80 .github/workflows

To convert the usage of `${{ vars.variablename }}` in the original GitHub Actions workflow into the calling of the new PowerShell script, you would need to pass these variables as environment variables to the PowerShell script.

Here's how you can do it:

```
steps:
- name: Checkout
  uses: actions/checkout@v2

- name: Run PowerShell script
  run: |
    ${ env.VARIABLE_NAME }=${{ vars.variablename }}
    powershell -File ./deploy-infra.ps1
  env:
    VARIABLE_NAME: ${{ vars.variablename }}
```

Step 9:

- Be more specific, and direct Copilot into the correct files
- Result: the code that I needed!

The screenshot shows a GitHub Copilot chat interface. At the top, a user named 'rajbos' asks for help converting code from a workflow file to a PowerShell script. Below the user's message, GitHub Copilot provides a list of references used in its response. At the bottom, Copilot outputs a YAML snippet for a workflow step that runs a PowerShell script with environment variables.

rajbos

I am converting the code in `#file:deploy-infra.yml` into `#file:create-infra.ps1` , give me the call that I need to do in the PowerShell script in the workflow file, to execute the script in the PowerShell file. Convert the `${{ vars.variable }}` and `${{ secrets.name }}` into the proper environment variables in calling the script

GitHub Copilot

Used 3 references

- ! `deploy-infra.yml:23-122` `.github/workflows`
- ! `deploy-infra.yml` `#file:deploy-infra.yml`
- > `create-infra.ps1` `#file:create-infra.ps1`

To call the PowerShell script from the workflow file, you can use the `run` command in a new step. You'll need to pass the environment variables using the `env` keyword. Here's how you can do it:

```
- name: Run PowerShell Script
  run: ./create-infra.ps1
  env:
    AZURE_CLIENT_ID: ${{ secrets.AZURE_CLIENT_ID }}
    AZURE_CLIENT_SECRET: ${{ secrets.AZURE_CLIENT_SECRET }}
    AZURE_SUBSCRIPTION_ID: ${{ secrets.AZURE_SUBSCRIPTION_ID }}
    AZURE_TENANT_ID: ${{ secrets.AZURE_TENANT_ID }}
    RESOURCE_GROUP_NAME: ${{ secrets.RESOURCE_GROUP_NAME }}
    KEYVAULT_NAME: ${{ secrets.KEYVAULT_NAME }}
```

TL;DR

Be

specific

Know

how prompting works

Add

extra context (#file:) where needed