Microsoft

Azure Rendering GitOps Deployment

GitHub + Jenkins + Terraform

June 2020

Objective

This document defines the deployment process for an Azure Rendering GitOps CI/CD workflow, which integrates GitHub, Jenkins and Terraform automation. Either GitHub.com, GitHub Enterprise Server or GitHub Enterprise Cloud can be used as the source control management system with support for the GitHub Checks API, which is a GitHub App capability.

Deployment

1. **GitHub App (developer.github.com/apps/about-apps/#about-github-apps)**

    a) Use the following GitHub article to create a new GitHub App in your target GitOps environment. https://developer.github.com/apps/building-github-apps/creating-a-github-app/

        i. While the "Homepage URL" is a required input field to create a new GitHub App, the value is not used by the GitOps workflow. Therefore, enter any homepage URL value for your environment.

        ii. Under the "Webhook" section, enter the "**Webhook URL**" value for your Jenkins environment using the format **http[s]://[Your-Jenkins-Host]/generic-webhook-trigger/invoke**, which points to the Generic Webhook Trigger plugin that will be installed in the Jenkins Server section (#2).

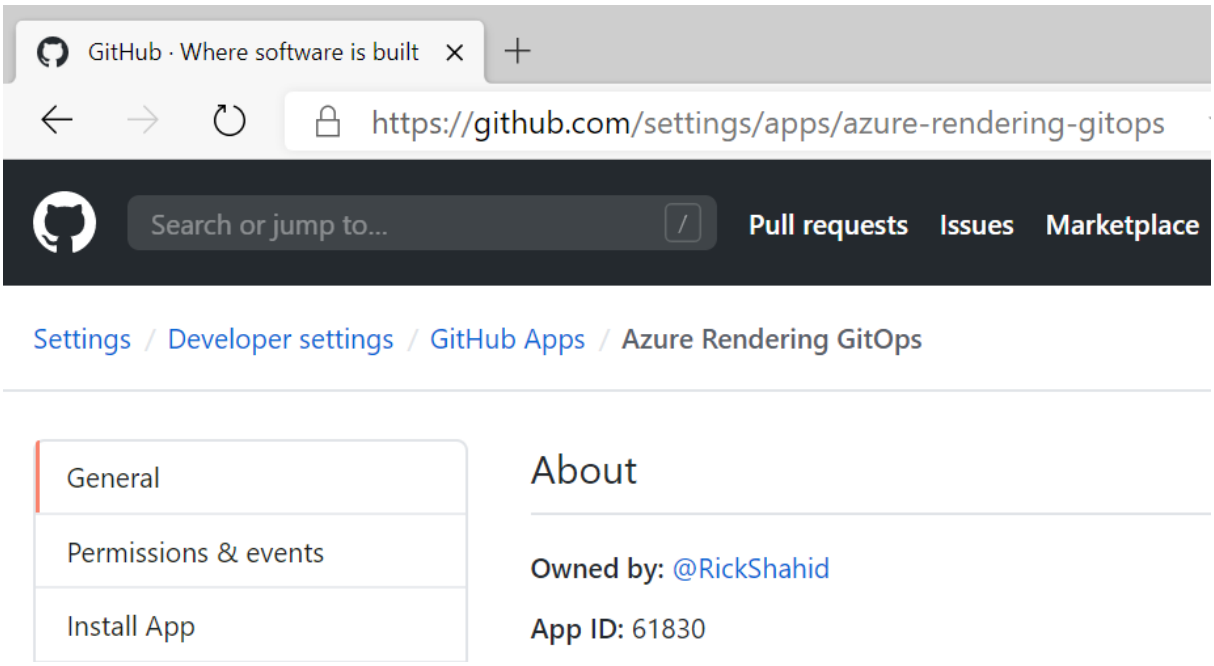
        While it is not a required input field, it is recommended to set a "**Webhook Secret"** value per https://developer.github.com/webhooks/securing/. This same value can then be set in your Jenkins environment to restrict pipeline client requests to only come from your GitHub App.

        iii. Under the "Repository Permissions" section, configure the following permissions.
            - Set **Checks** to Read & Write
            - Set **Contents** to Read & Write
            - Set **Issues** to Read & Write
            - Set **Pull Requests** to Read & Write

        iv. Under "Subscript to Events", configure the following events.
            - Enable **Check Run**
            - Enable **Pull Request**

        v. The other GitHub App settings are all optional. Configure the additional GitHub App settings as needed for your environment. Click "**Create GitHub App**" when your configuration is complete.

    b) After your new GitHub App is created, you will be taken to the application settings page, which is at **https://github.com/settings/apps/[Your-GitHub-App-Name].** Make sure to note your new **App ID** number near the top of the page. It will be needed to configure Jenkins connection authentication.



c) Scroll to the bottom of the page and click "**Generate a Private Key**", which will generate and download a new RSA private key file in PEM file format. You will need this private key in the Jenkins Server section to setup secure communication with your new GitHub App. Make sure to store your key file in a safe place.

d) Copy the Azure Rendering GitOps source code files from the following URL into your GitHub repository. https://github.com/Azure/Avere/tree/master/src/tutorials/GitOps/Jenkins

i. Make sure the **AzureEnvironment.Development.Backend.config** source code file path and name in your GitHub repository corresponds to the following Jenkins pipeline parameter in the **Terraform Plan** pipeline. This file configures Terraform backend deployment state management.

- string(name: 'TF_ENVIRONMENT_BACKEND_CONFIG_FILE', defaultValue: '**/AzureEnvironment.Development.Backend.config**')

ii. Make sure the **AzureEnvironment.Development.Variables.config** source code file path and name in your GitHub repository corresponds to the following Jenkins pipeline parameter in the **Terraform Plan** and **Terraform Apply** pipelines. This file configures Terraform Azure connection.

- string(name: 'TF_ENVIRONMENT_VARIABLES_CONFIG_FILE', defaultValue: '**/AzureEnvironment.Development.Variables.config**')

- There are 3 Terraform Azure connectivity options for **non-interactive** authentication per HashiCorp guidance as follows. Whenever it is possible, using an **MSI** is recommended.
  a. **Managed Service Identity (MSI)**
  b. Service Principal with Client Certificate
  c. Service Principal with Client Secret

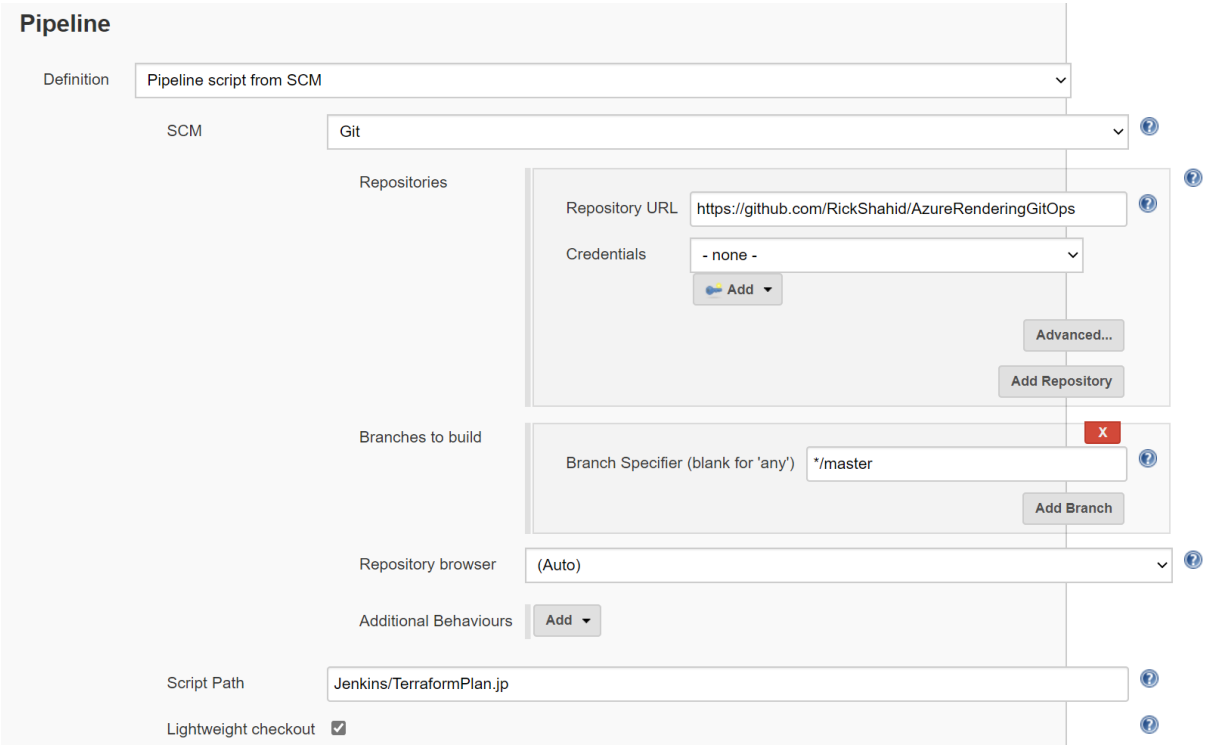
iii. Make sure the **GetGitHubWebToken.py** Python source code file path and name in your GitHub repository corresponds to the following Jenkins pipeline parameter in the **Terraform Plan** and **Terraform Apply** pipelines. This file acquires a GitHub App session token during each pipeline.

- string(name: 'GITHUB_APP_AUTH_TOKEN_FILE', defaultValue: '**/GetGitHubWebToken.py**')

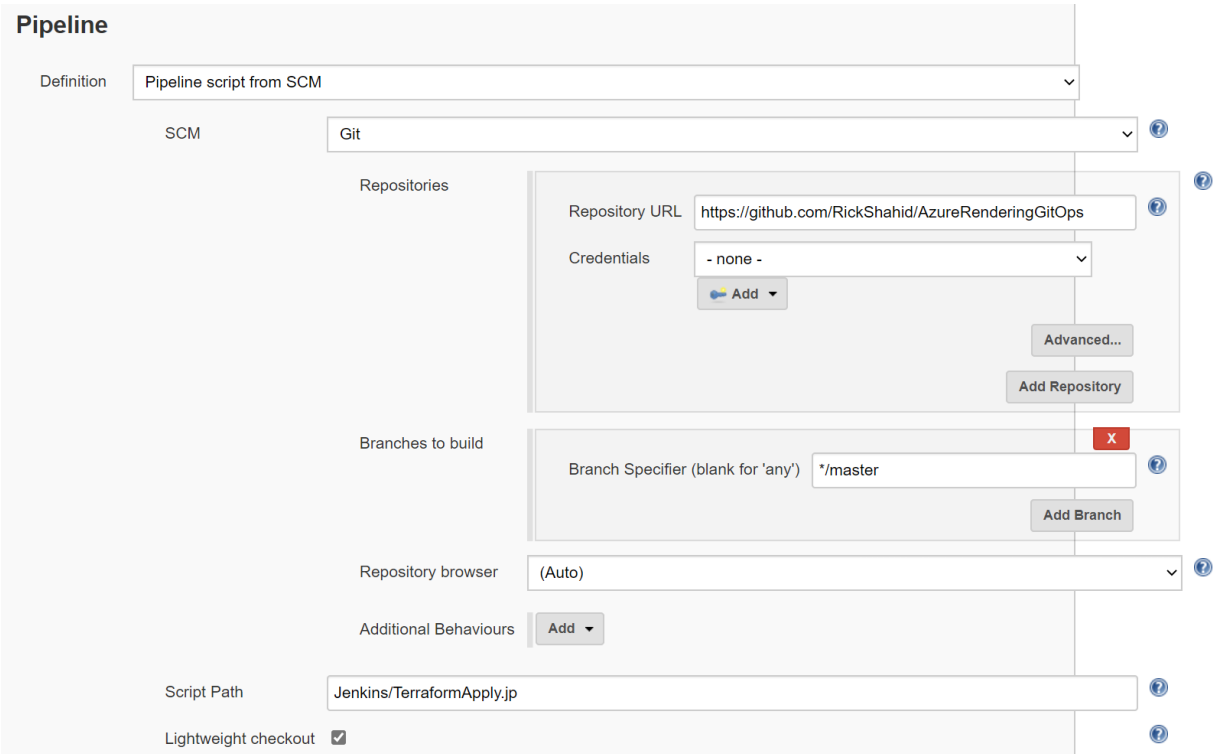
e) Install your GitHub App on the GitHub repository that contains your Terraform infrastructure (*.tf) files. From **https://github.com/settings/apps/[Your-GitHub-App-Name]/installations,** select the repository to enable with the Azure Rendering GitOps workflow. You will then be prompted to accept the required permissions that were previously defined when the GitHub App was created. Click "**Install**" to proceed.

2. **Jenkins Server (www.jenkins.io)**

a) Using the Jenkins new pipeline creation user interface, create the **Terraform Plan** pipeline as follows. It should reference the **TerraformPlan.jp** declarative pipeline source code file in your GitHub repository.



b) Using the Jenkins new pipeline creation user interface, create the **Terraform Apply** pipeline as follows. It should reference the **TerraformApply.jp** declarative pipeline source code file in your GitHub repository.



c) Make sure the following required Jenkins plugins are installed in your Jenkins environment.
   - **Generic Webhook Trigger** – https://plugins.jenkins.io/generic-webhook-trigger/
   - **Pipeline Utility Steps** – https://plugins.jenkins.io/pipeline-utility-steps/
   - **HTTP Request** – https://plugins.jenkins.io/http_request/

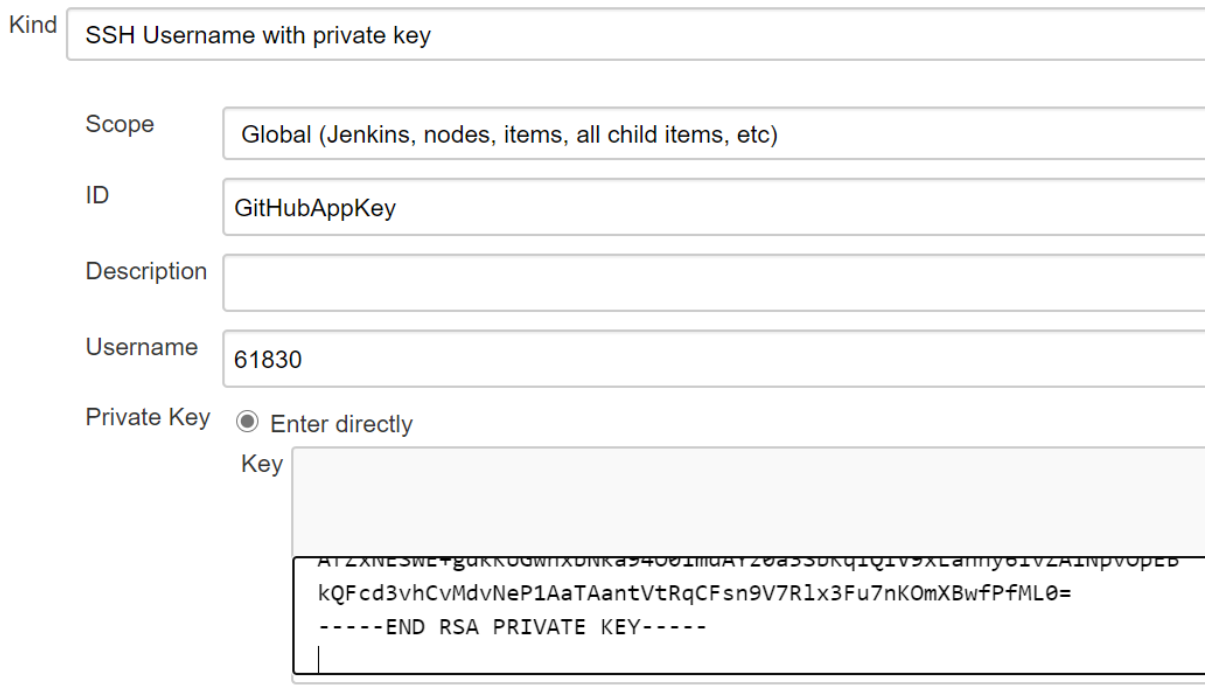
d) Make sure the latest version of Terraform (v0.12.28) is installed in your Jenkins environment.

e) Make sure the latest Azure Terraform Avere provider is installed in your Jenkins environment and it is also executable (**chmod 755 terraform-provider-avere**). At the time of writing, the current release is at https://github.com/Azure/Avere/releases/download/tfprovider_v0.9.2/terraform-provider-avere

f) Make sure the **JSON Web Token** (https://pypi.org/project/jwt/) Python 3 library is installed in your Jenkins environment. This library is used in the **GetGitHubWebToken.py** Python source code file.

g) Using the GitHub App private key file (.pem) that you created earlier, create a global credential in your Jenkins environment. Set the **Username** and **Private Key** fields as illustrated below to your GitHub App ID and RSA private key data. This will ensure secure storage and usage of your private key in Jenkins. Make sure the **Credential ID** value corresponds to the following Jenkins pipeline parameter.

   - string(name: 'GITHUB_APP_KEY_ID', defaultValue: **'GitHubAppKey'**)



h) (**Optional**) Create a Secret Text credential in your Jenkins environment with the **Webhook Secret** value that you specified earlier in the GitHub App section. Make sure the **Credential ID** value corresponds to the following Jenkins pipeline parameter in the **Terraform Plan** and **Terraform Apply** pipelines. ***NOTE: Due to issue https://issues.jenkins-ci.org/browse/JENKINS-57390, this secret is currently not used.***

   - string(name: 'GITHUB_APP_SECRET_ID', defaultValue: **'GitHubAppSecret'**)