Microsoft

Azure Rendering GitOps Deployment

GitHub + Jenkins + Terraform

April 2020

Objective

This document describes the deployment process for the Azure Rendering GitOps CI/CD workflow that provides GitHub, Jenkins and Terraform integration. Either GitHub.com or GitHub Enterprise can be used as the source control system as long as your GitHub environment version includes support for the GitHub Checks API, which is a GitHub App feature.

Deployment

1. **GitHub App**

    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 whatever value works best in your environment.

        ii. Under the "Webhook" section, enter the "**Webhook URL**" value in your environment with the format of **http://[your Jenkins host]/generic-webhook-trigger/invoke**, which corresponds to the Generic Webhook Trigger plugin that will be installed in your Jenkins environment.

        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", enable the following events.
            - **Check Run**
            - **Pull Request**

        v. The other GitHub App settings are optional and are currently not used by the GitOps workflow. Configure each setting as appropriate for your environment and then click "**Create GitHub App**". After the GitHub App is created, you will be taken to the application settings page.

    b) After your new GitHup App is created, make sure to note the new **App ID** number, which is displayed at **https://github.com/settings/apps/[your-new-GitHub-App-name]** underneath the "Owned By" value. This value will be used to configure private key authentication from Jenkins to your GitHub App.

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

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

i. Make sure the **AzureEnvironment.Development.Backend.config** configuration file path in your GitHub repository corresponds to the Jenkins pipeline parameter in the Plan pipeline. This file tells Terraform about your backend configuration for managing deployment state.

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

ii. Make sure the **AzureEnvironment.Development.Variables.config** configuration file path in your GitHub repository corresponds to the Jenkins pipeline parameter in both the Plan and Apply pipelines. This file tells Terraform how to securely connect to your Azure environment.

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

- There are 3 Terraform Azure configuration options for **non-interactive** authentication per HashiCorp guidance as follows. Wherever it is possible, 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 in your GitHub repository corresponds to the following Jenkins pipeline parameter in both the Plan and Apply pipelines.

- 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 that you would like to enable with the GitOps workflow. You will be prompted to accept the required permissions that have been previously defined for the GitHub App. Click "**Install**" to proceed.

2. **Jenkins Server**

a) Using the Jenkins pipeline creation UI, create 2 pipelines as follows. If you decide to rename the pipelines, make sure that the new names are also set in the Plan and Apply pipeline parameters.
- **Terraform Plan** points to **TerraformPlan.jp** in your GitHub repository
- **Terraform Apply** points to **TerraformApply.jp** in your GitHub repository

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

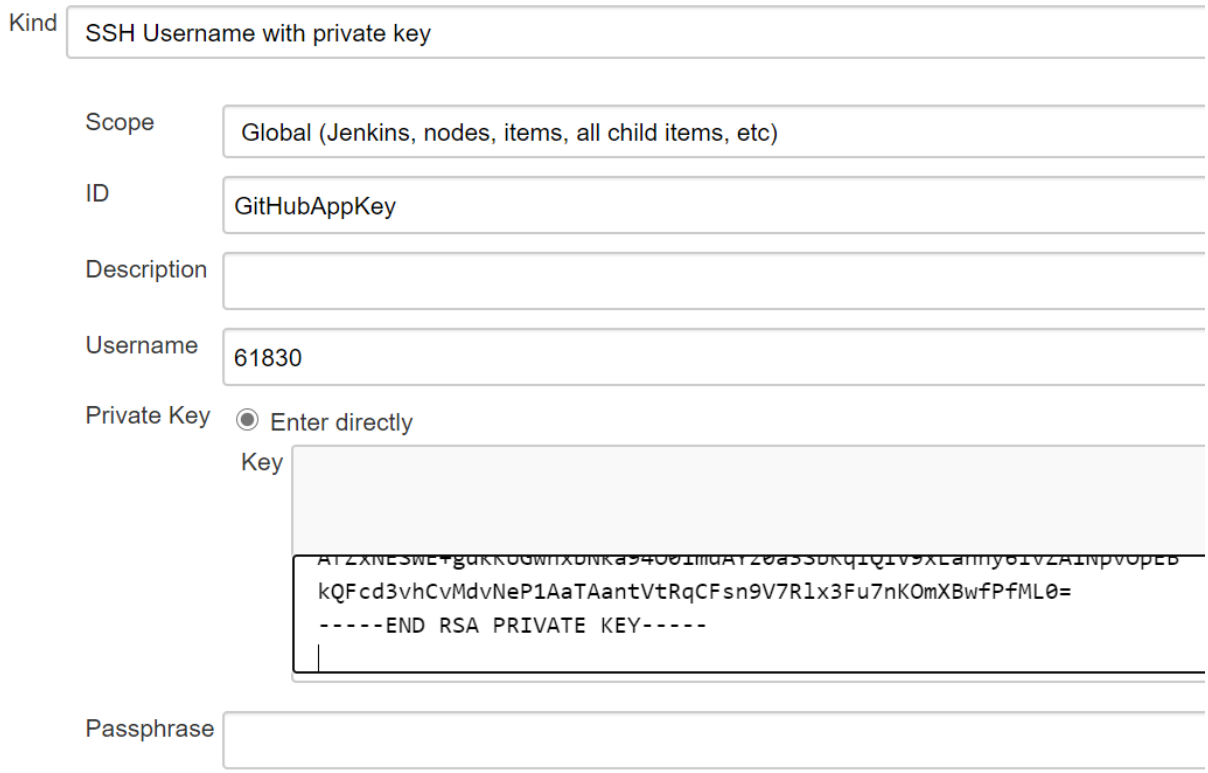
c) Make sure that the latest version of Terraform (v0.12.24) is installed in your Jenkins environment.

d) Make sure that the latest Azure Terraform Avere provider is installed in your Jenkins environment. For reference, https://github.com/Azure/Avere/releases/download/0.7.1/terraform-provider-avere. Make sure that the Terraform Avere provider is executable (**chmod 755 terraform-provider-avere**).

e) Make sure that JSON Web Token library for Python 3 is installed in your Jenkins environment.

f) Using the GitHub App private key file that you previously created, create a global credential and set the Username and Private Key fields to your GitHub App ID and RSA private key data. Make sure that the Credential ID value corresponds to the associated Jenkins pipeline input parameter for both pipelines.

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



g) Create a Secret Text credential with the value that you specified previously (in GitHub App section 1. a). Make sure that the Credential ID value corresponds to the associated Jenkins pipeline input parameter for both the Terraform Plan and Terraform Apply pipelines. ***NOTE: Due to issue https://issues.jenkins-ci.org/browse/JENKINS-57390, this credential value is currently not used in the Jenkins pipelines.***

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