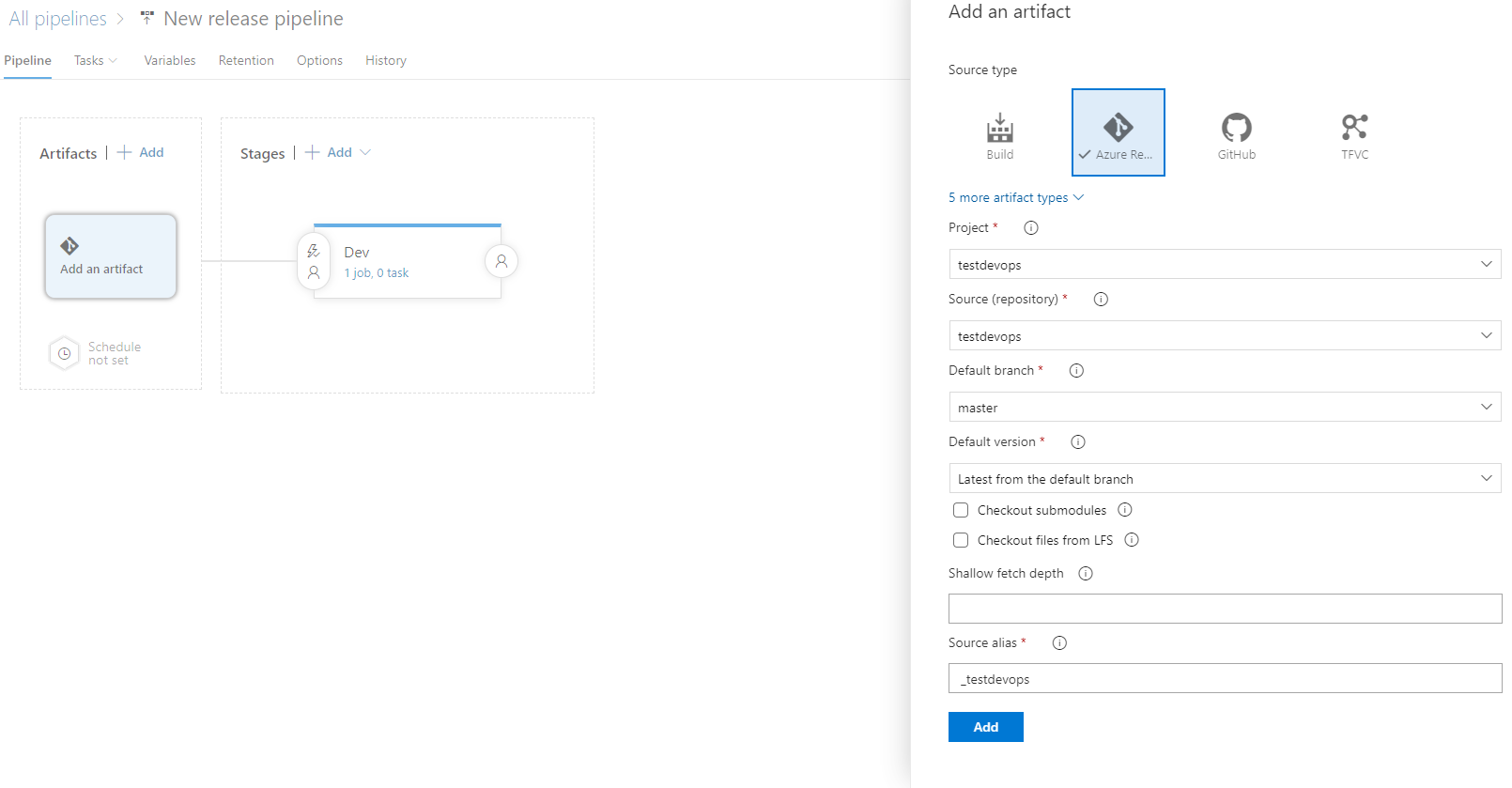
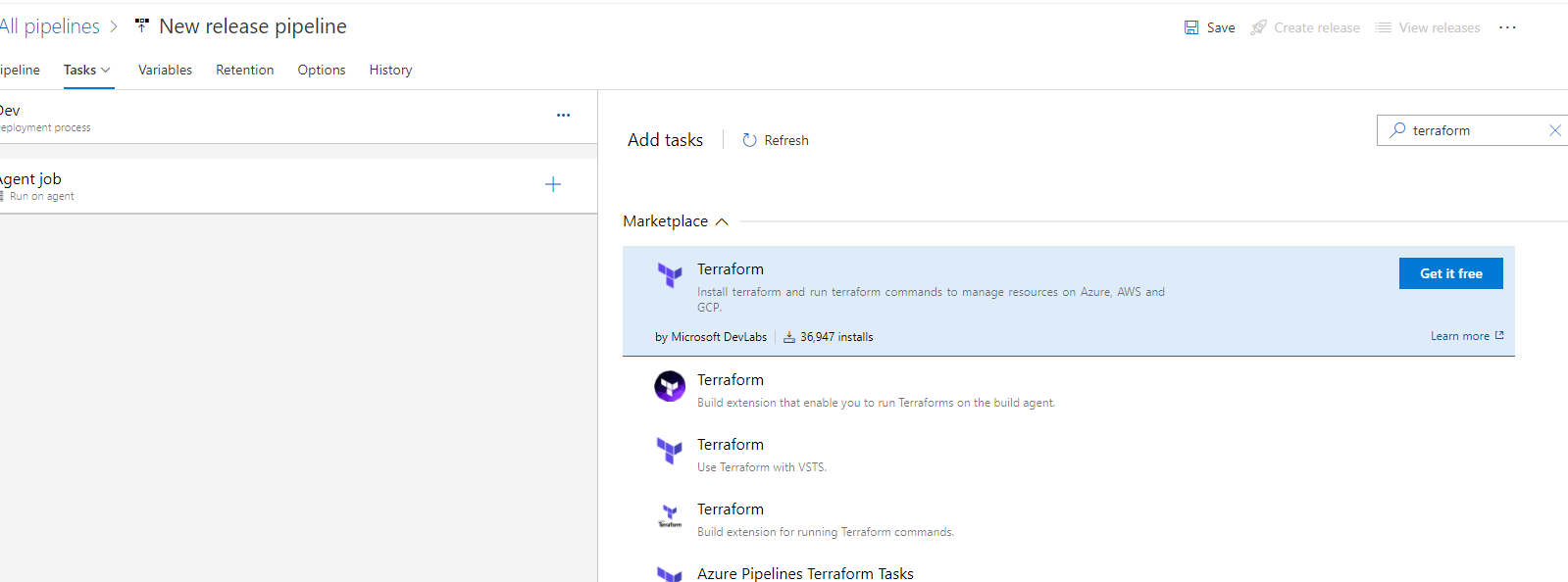
Scenario 2:

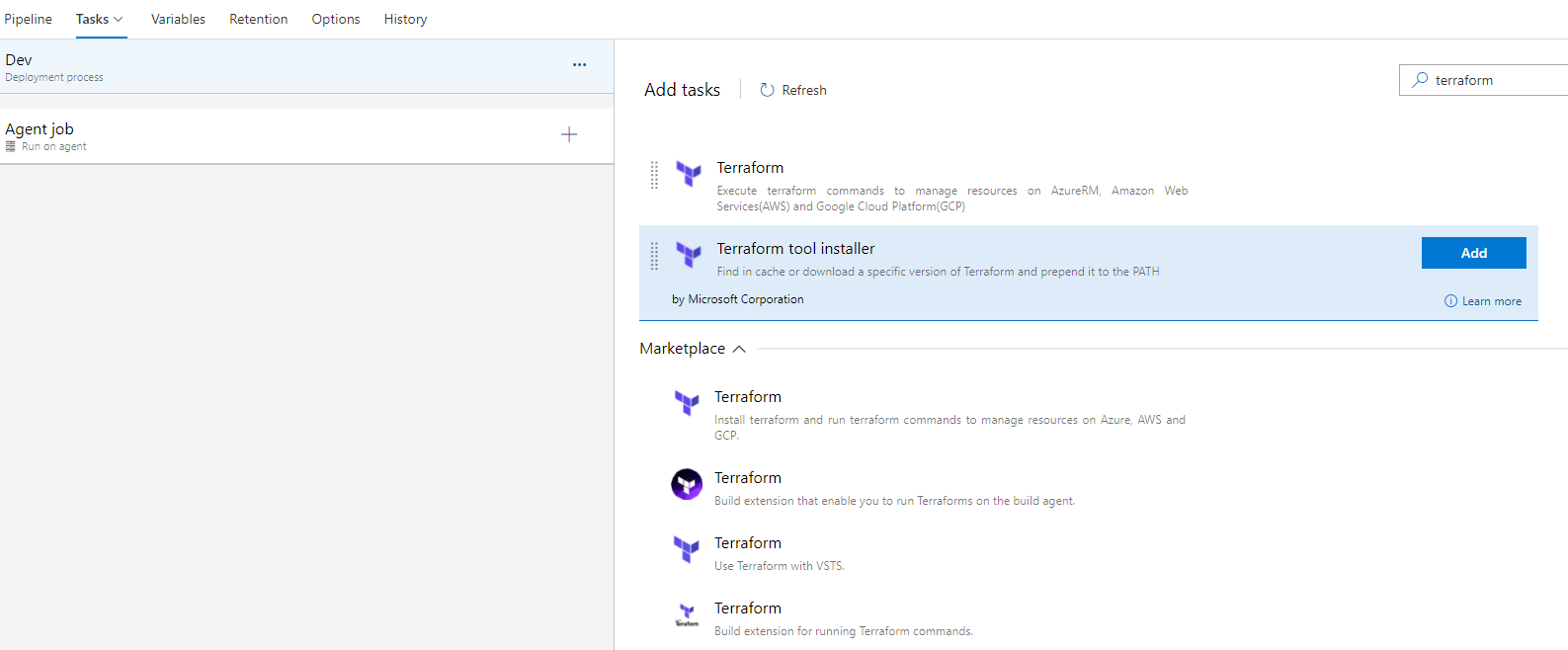
1. Below are the required artifacts and this should be placed in the same Azure devops repository:
   1. **Dev.tfvars**: Environment specific variable and this will be based on the respective environment. Terraform will take the values placed in this file.
   2. **Variable.tf**: List all the variables which should be unique.
   3. **Main.tf**: Access variables by prefixing with var. This is the main configuration file where all the infrastructure resources will be configured.
2. We are using **Visual studio code** with the following extensions configured:
   1. Github: Perform git operations
   2. HashiCorp Terraform: Useful to validate basic syntax, Snippets and syntax highlighting
3. Below are the steps to create automated pipelines in **dev** environment:
   1. Create Azure repo and push the files from local system using git operations (git stage, commit and push)
   2. Navigate to Pipelines –> Releases. Click New and select New release pipeline and click empty job.
   3. Provide a name for stage “dev” and click the close button.
   4. Select Add an artifact and choose Azure repository.
   5. Choose a project created in Azure DevOps Repos in step 1 and select the repository where main.tf is stored.
   6. Select the master branch and latest from the default branch in default version and click Add.



* Select “dev” stage and click “View stage tasks” to view the pipeline tasks.
* Search for terraform and select Terraform. This will download the tasks from marketplace. This will happen only one time per organization.



* After installation, we should be able to view terraform tool installer and Terraform tasks. Select one installer task and 3 Terraform tasks(init, plan and apply).



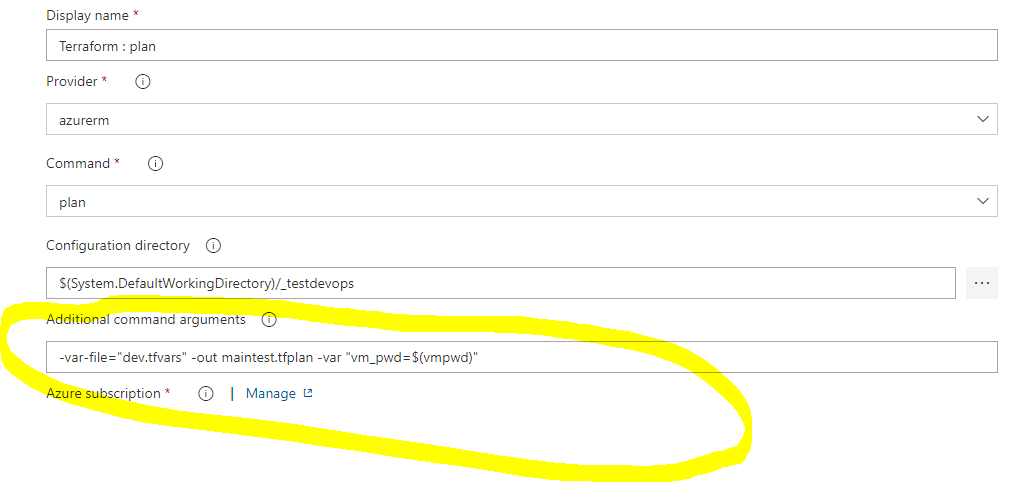
* Configure Install, init, plan and apply tasks:

**Install**: Configure terraform version to 1.0.7 in terraform tool installer task

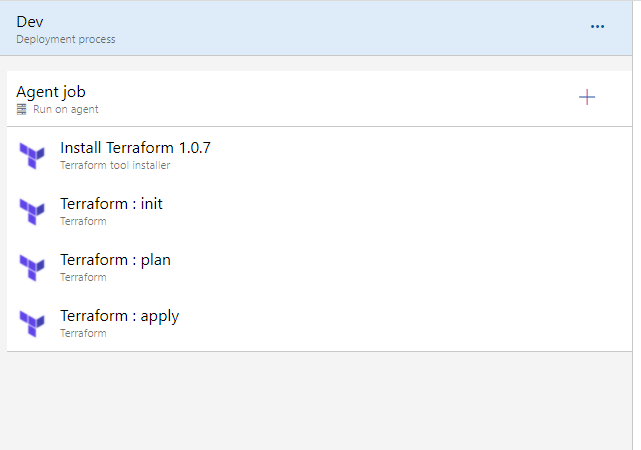
**Init**: Update the name of the Terraform task to Terraform: init in In the first terraform task. Make sure provider name as Azurerm and command as “init”. Choose configuration directory where main.tf is located. Configure Azure subscription, authorize then select resource group, storage account and blob container to store terraform state.



**Plan**: Change the display name as Terraform: plan in the second Terraform task. Keep provider as Azurerm then select command as “Plan”. Choose the configuration directory where main.tf is stored then select Azure subscription. Keep these commands in additional command arguments type to maintain the state : - -var-file="dev.tfvars" -out maintest.tfplan -var "vm\_pwd=$(vmpwd)"



**Apply**: Change display name to Terraform: apply in the last Terraform task. Keep provider value as Azurerm and command as “validate and apply”. Choose the configuration directory as main.tf is stored. Select maintest.tfplan in additional command arguments type which created in the above plan task. Then select Azure subscription for deployment.



1. Attached sample terraform templates developed:



1. How to access keyvault secrets in terraform variables:

* Create key vault variable(vmpwd) in devops variable group
* Attach variable group in the release pipeline
* Pass devops keyvault variable to the terraform plan task.

