**GitHub & GitHub Actions**

### 🚀 What is ****GitHub Actions****?

**GitHub Actions** is a **CI/CD (Continuous Integration and Continuous Deployment)** tool provided by GitHub that lets you **automate** tasks in your software development workflow — **directly in your GitHub repository**.

Create an account on GitHub:- <https://github.com/>

Create new repository :- Terraform-code

Organization requirement is to create a storage account code on GitHub (remote repository) then what will be the steps:-

1. We will access the ticket.
2. Clone the repo from remote to local (Git clone <https://github.com/getsumantra/Terraform-code.git>)
3. Create a feature branch
4. Write the storage account code in feature branch.
5. Git status
6. Git add **.**
7. Git commit -m “storage account code”
8. Git push
9. Raise pull request – feature branch to main and put reviewer

### ⚔️ What is a ****Git Conflict****?

A **Git conflict** happens when **two or more people** (or branches) have made **changes to the same part of the same file**, and Git doesn't know **which change to keep** during a **merge or rebase**.

Steps for pushing a storage account code from local to remote repository.

* Create a remote repository and copy its hyperlink
* Open vs code and clone it using (git clone https://github.com/getsumantra/Terraform-code.git)
* Create a feature branch using (git branch feature/storage)
* To check which branch is selected (git branch)
* To switch or directly create feature branch (git checkout -b feateure/storage)
* Create a storage account code inside VScode
* Add. gitignore file and add all file which you won’t push like (.terraform, lock.hcl, .exe file)
* Git status
* Git add main.tf (to add one file in staging area)
* Git add **.** (to copy complete file in staging area)
* Git commit -m “add\_storage\_account”
* Git push (but due to applied policies on main branch code could note be pushed on the main)
* git push --set-upstream origin feature/storage
* Now code has been pushed to repository on feature branch select this and raise a PR
* Raise PR (pull request) and add reviewer
* After the approval of reviewer code will be merged from the feature branch to main branch.

Steps for pushing a todoapp infra code from local to remote repository by raising a Pull Request…

* Create a remote repository and copy its hyperlink
* Open vs code and clone it using (git clone https://github.com/getsumantra/Terraform-code.git)
* Create a feature branch using (git branch feature/Jira-101-new-resources)
* To check which branch is selected (git branch)
* To switch or directly create feature branch (git checkout -b feature/ Jira-101-new-resources)
* Create a VM code using modules approach code inside VScode
* Add. gitignore file and add all file which you won’t push like (. terraform, lock.hcl, .exe file)
* Git status

Now we need to setup a policy to remote repository so that any can’t add the code directly to our main branch

* <https://github.com/getsumantra/testing> go to repository first where you want to set policy
* Repo >> Setting >> collaborator >> add people >> add person email id which will be approve all PR
* Repo >> setting >> Branches >> add classic branch protection rule
* Check on box :- Require a pull request before merging >> add at least 1 approver
* Go to VS code and do it
* Git add **.**
* Git commit -m “code added”
* Git push (but due to applied policies on main branch code could note be pushed on the main)
* git push --set-upstream origin feature/storage
* Now code has been pushed to repository on feature branch select this and raise a PR
* Raise PR (pull request) and add reviewer
* After the approval of reviewer code will be merged from the feature branch to main branch.

Now policy has become set lets and code pushed in feature branch now raise a Pull Request (PR):

* [getsumantra](https://github.com/getsumantra) / [testing](https://github.com/getsumantra/testing) (go to repo) >> pull request >> New pull request >> add one approver >> add comment and submit
* Email has sent to approver email id open that link and check and approve the required changes

Now the code has become merged to the main branch and changes will be shown here

**Git conflict **

A **Git conflict** happens when **two or more people** (or branches) have made **changes to the same part of the same file**, and Git doesn't know **which change to keep** during a **merge or rebase.**

How conflict will be created if I’m using my own laptop and no teammates is there….?

Requirements:

* Create two feature branch and change the code and commit in both branches and push it one by one.
* And raise PR request then one conflicts will be shown during code review.

Steps to create the repository & clone it on local:

* Create a remote repository and copy its hyperlink
* Open vs code and clone it using (git clone <https://github.com/getsumantra/Terraform-code.git>)
* For the first time create a code and push to GitHub repo.

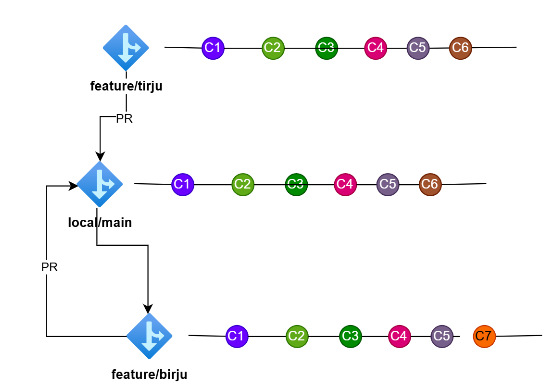
Steps to create the two-feature branches & How conflicts arise:

* git checkout -b feature/101-Jira-RG >> Create and select a new branch
* git checkout -b feature/102-Jira-Vnet >> Create and select a new branch
* git checkout feature/101-Jira-RG >> Select branch
* git add .
* git commit -m “one\_rg\_added”
* git push >> error
* git push - -set-upstream origin feature/101-Jira-RG

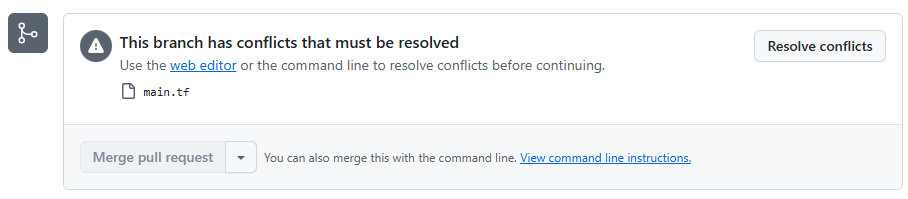
Raise a PR for branch feature/101-Jira-RG, after approval code will be merged to the main branch

But from the second feature branch if we are pushing any code which doesn’t have the previous commit the it will create a conflict.

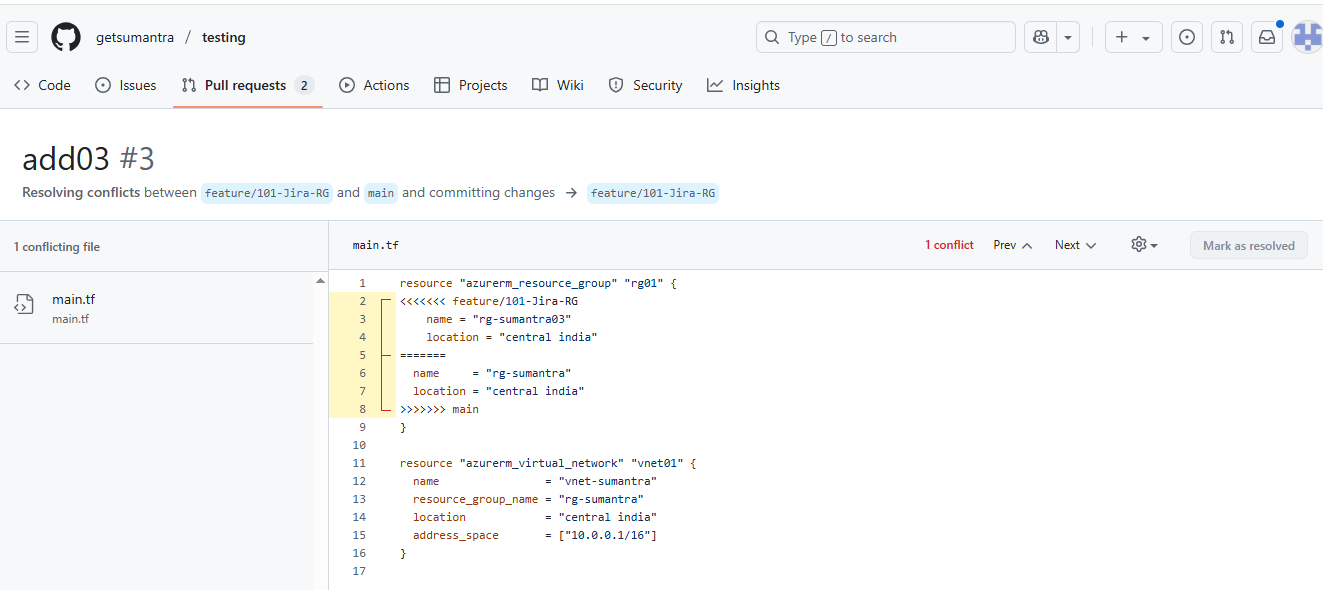
For eg: - if feature branch tirju has 6 commits and birju branch doesn’t have 6th commit details then conflict will come up on branch tirju.



How to resolve this git conflicts: -

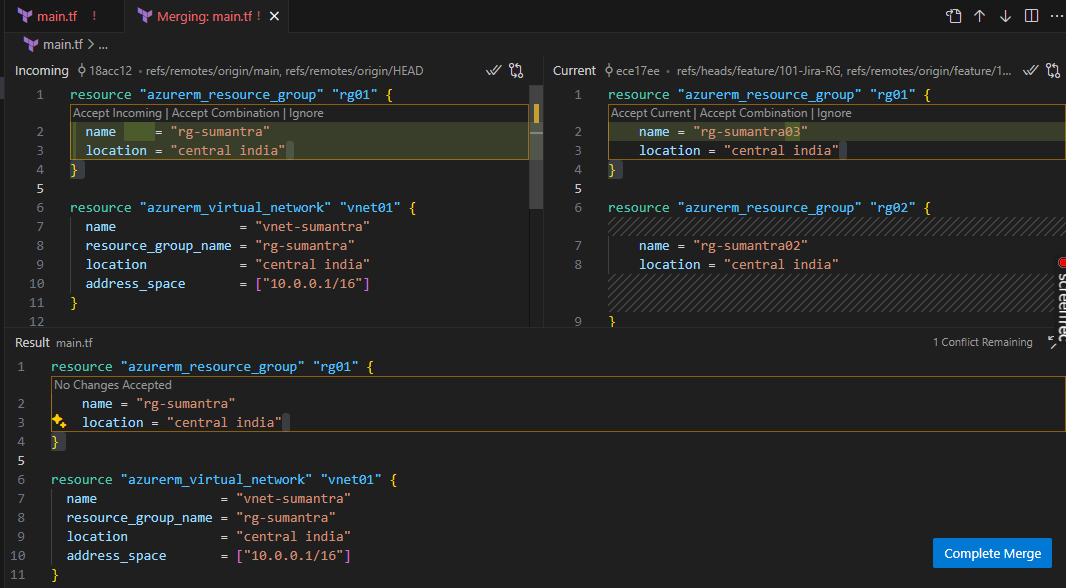


Option 1. Got to inside resolve conflicts option and this window will appear and remove not required codes and merge it



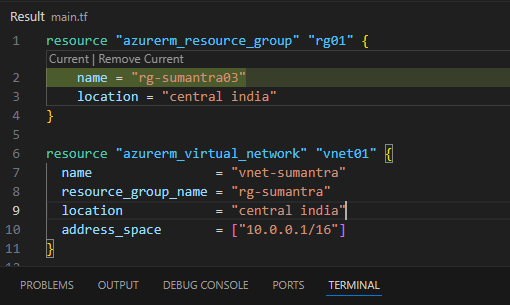
Option 2.

* Select branch which having the conflict git checkout feature/101-Jira-RG
* git pull origin main



Isme se kaun sa code rkhna hai usko rkh lenge aur complete merge wale option pr click kr denge

Then again PR approve kr ke merge kr denge, Result main.tf will be the final commit.



Security tools for the Infrastructure scanning

For the infrastructure we can use multiple tools mostly tools are CLI based so for this we need to visit their GitHub repository.

1. **Tfsec**

* <google search> tfsec github >> <https://github.com/aquasecurity/tfsec>

Go to the Releases option >> drag down and select version >> [tfsec-windows-amd64.exe](https://github.com/aquasecurity/tfsec/releases/download/v1.28.14/tfsec-windows-amd64.exe) >> download this

* Rename this file as >> tfsec >> copy and create a folder in C drive and paste it copy that folder path and set it environment variables >> path set
* Open VS code and type >> tfsec –version (if version is appearing then its okay)

1. **Tflint**

* <google search> tflint github >> <https://github.com/terraform-linters/tflint>

Go to that Releases option >> drag down and select version >> [tflint\_windows\_amd64.zip](https://github.com/terraform-linters/tflint/releases/download/v0.58.1/tflint_windows_amd64.zip) >> download this

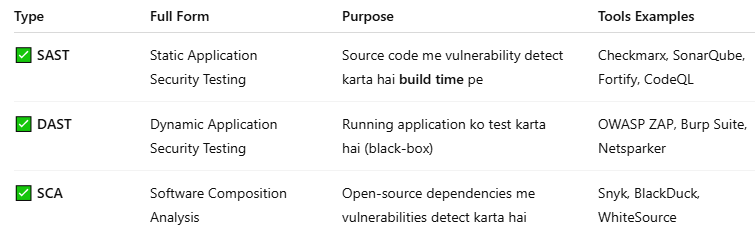
* Rename this file as >> tfsec >> copy and create a folder in C drive and paste it copy that folder path and set it environment variables >> path set (or paste directly into the tfsec folder then no need to set environment variables)
* Open VS code and type >> tflint –version (if version is appearing then it’s okay)

We can follow same process for each security tools

1. **Checkov**
2. **Balckduck**
3. **Terrascan**
4. **Trivy**
5. **Snyk IaC**
6. **Prisma Cloud**

Security tools for the applications

Type of application security tools



1. **SonarQube**
2. **BlackDuck** (SCA - OSS License + Vuln)
3. **Aqua**
4. **Trivy**
5. **Checkmarx** (SAST)
6. **Snyk (Code + SCA)**
7. **OWASP ZAP** (DAST)
8. **Burp Suite** (DAST)

Azure Devops

Github actions

Azure Devops

UI Based

YAML Based

Classic Pipeline Using Azure Devops

<https://aex.dev.azure.com/> >> Login >> set organization >> create project

Go to the organization level setting >> Pipelines >> setting >> and enable two option below

Off >> **Disable creation of classic build pipelines**

Off >> **Disable creation of classic release pipelines**

Select Project >> Pipelines >> New pipeline (this the classic pipeline UI based) >> [Use the classic editor](https://dev.azure.com/Terraform-devlopment/Terraform-Infra/_apps/hub/ms.vss-ciworkflow.build-ci-hub?_a=build-definition-getting-started&id=0) to create a pipeline without YAML. >> Select GitHub >> Select > Repository > branch > main

Select a template >> Empty Job

Pipeline >> Name > Terraform init (display name) > Select agent pool

Agent Job 1 >> Terraform init (Display name) > Select agent pool

Agent Job 1 >> + >> PowerShell or Azure cli >> Display name >> Type > Inline

Script >>

Cd $(System.DefaultWorkingDirectory)/your-folder-name (if you are choosing PowerShell)

ls

terraform init

terraform validate

checkov -d **.**

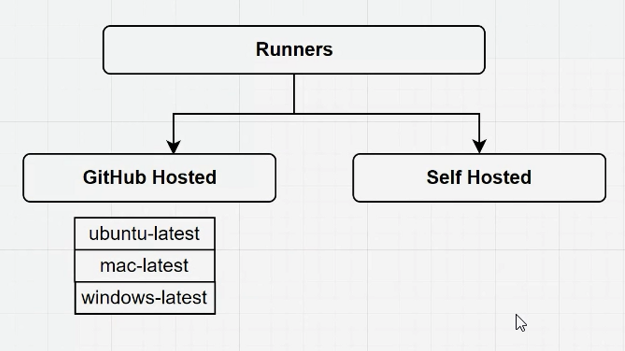
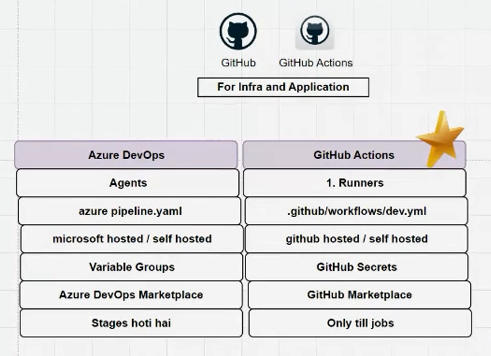
terraform plan

terraform apply

>>>> Save & Queue

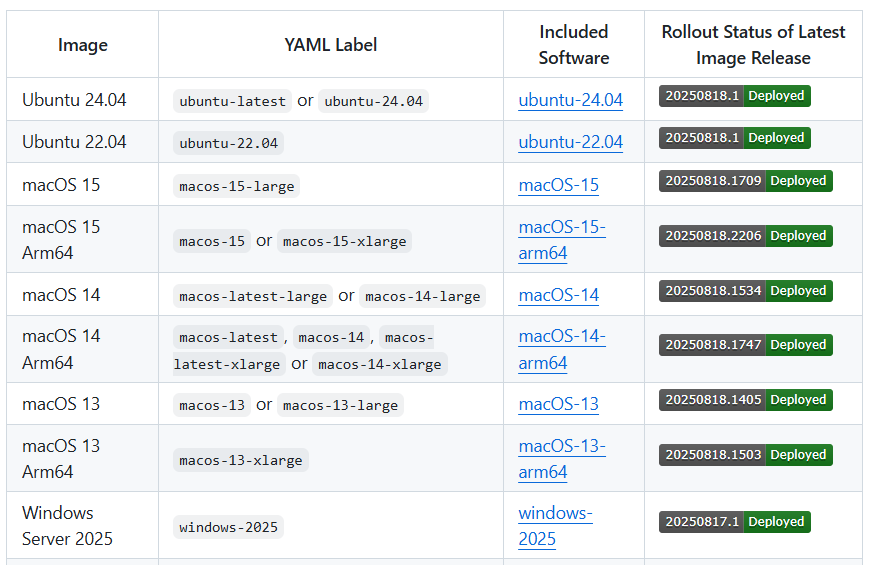
Now pipeline will RUN and resource get provisioned

GitHub Actions



GitHub Hosted runners: - <https://docs.github.com/en/actions/concepts/runners/github-hosted-runners>

Runners: - <https://github.com/actions/runner-images>



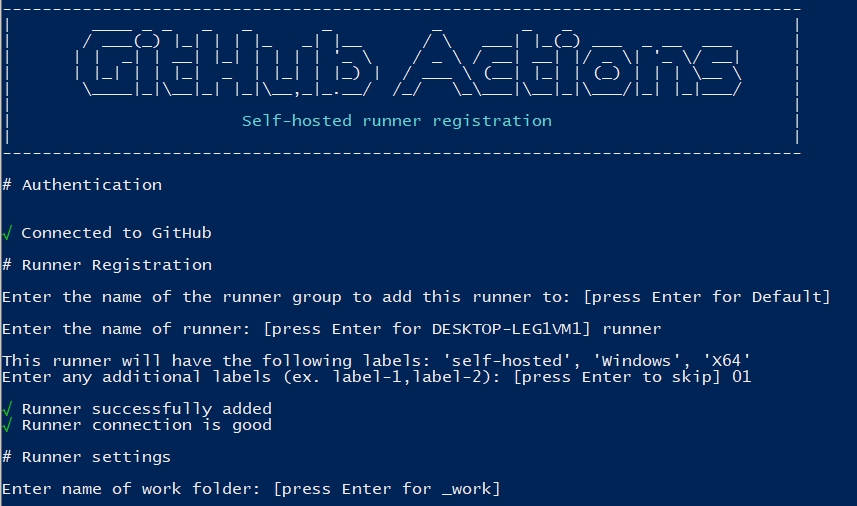
To set self-Hosted Runners in GitHub Actions pipeline:-

Go to GitHub repository:- <https://github.com/> >> go to your repository >> setting >> actions >> Runners

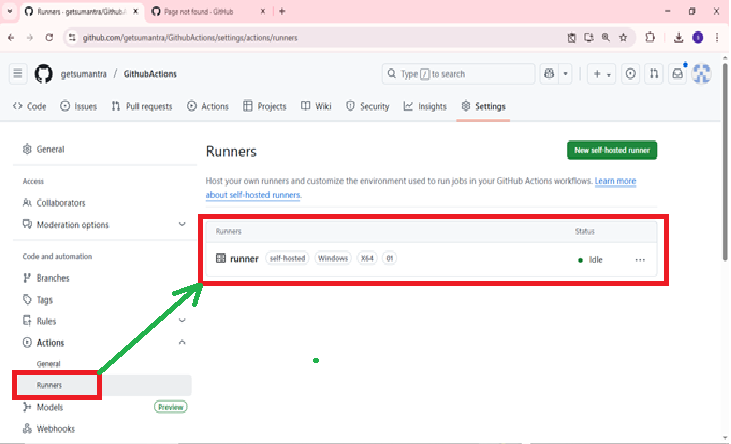
Follow all steps which is given inside this to run out runners (agent) >> new selfhosted runners >> select windows >> Goto local repo and run all command in PowerShell one by one

Download directly from the browser by using below link

<https://github.com/actions/runner/releases/download/v2.328.0/actions-runner-win-x64-2.328>



Now runner is online: -



Push the code from local to remote repo

Actions >> configure terraform plugins >> [set up a workflow yourself](https://github.com/getsumantra/GithubActions/new/main?filename=.github%2Fworkflows%2Fmain.yml&workflow_template=blank)

Goto google and type >> GitHub actions yaml >> <https://docs.github.com/en/actions/reference/workflows-and-actions/workflow-syntax>

 YAML Pipeline of GitHub actions

For YAML reference of GitHub Actions: -

<https://docs.github.com/en/actions/reference/workflows-and-actions/workflow-syntax>

name: infraBastion

on: workflow\_dispatch

permissions:

id-token: write

contents: read

jobs:

buildStage\_Job:

name: Build Stage

runs-on: runner

steps:

- name: Checkout

uses: actions/checkout@v5.0.0

- name: Azure login

uses: azure/login@v2

with:

client-id: 27a1ee5a-8674-4ee7-8991-630cdd3feb1b

tenant-id: b24a5be3-6852-43c6-b00e-086d16b535ab

subscription-id: 258a8e61-bb4a-4a2e-99d2-ca7211e4a421

- name: Terraform fmt

id: fmt

run: terraform fmt -check

continue-on-error: true

working-directory: ./dev/infra

- name: Terraform Init

id: init

working-directory: ./dev/infra

run: terraform init -input=false

- name: Terraform Validate

id: validate

run: terraform validate -no-color

working-directory: ./dev/infra

- name: Terraform Plan

id: plan

run: terraform plan -no-color -input=false

continue-on-error: true

working-directory: ./dev/infra

deployStage\_Job:

needs: buildStage\_Job

name: Deploy Stage

runs-on: runner

steps:

- name: Azure login

uses: azure/login@v2

with:

client-id: 27a1ee5a-8674-4ee7-8991-630cdd3feb1b

tenant-id: b24a5be3-6852-43c6-b00e-086d16b535ab

subscription-id: 258a8e61-bb4a-4a2e-99d2-ca7211e4a421

- name: Terraform Init

id: init

working-directory: ./dev/infra

run: terraform init -input=false

- name: Terraform apply

id: apply

working-directory: ./dev/infra

run: terraform apply -auto-approve