SWE 599 – Modern DevSecOps Architecture

Cankut ER
ID: 2022719219
Date: 24.11.2024

The Progress Report for the Term Project

1. Introduction

This project aims to create a reusable local kubernetes cluster that consists of components that will make up a modern DevOps pipeline with security checks. With re-usability in mind, it is aimed to require a very minimal manual setup to get the pipeline up and running. For this reason, Terraform is utilized to create the cluster and all the resources within. This has reduced the prerequisites drastically, such as removing the need to install a kubernetes distribution and helm, as they are automatically provisioned by defined "Terraform Providers" in the code. Additionally, supporting multiple operating systems is desired, which directly influenced the choice of kubernetes distribution for the project.

1.1 Prerequisites

Docker Desktop (docker engine to run kubernetes)

kubectl (cli tool to manage kubernetes resources)

Terraform (cli tool to manage infrastructure resources)

Please refer to the installation sections relevant to the operating system of use in official documentation or the documentation of the package manager of choice.

2. Project Overview

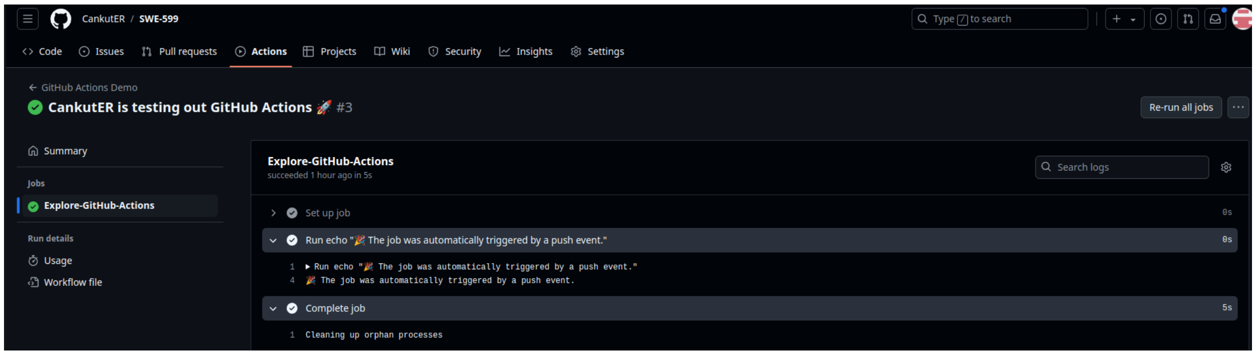
As already emphasized in the previous section, re-usability is one of the core aspects that is desired in this project. Such requirement implies that the project should be agnostic to the different operating systems as much as possible. As far as indifferentiability is concerned, first solution that comes to mind is leveraging Docker. Thus, "Kind" is chosen as the kubernetes distribution. It is relatively lightweight, supports multi-node clusters and creates nodes as docker containers, reasonably fast, easy to start-up and generally a popular choice for continuous integration workflows, testing and research purposes. Even there are more production-ready distros such as microk8s and k3s, their support on Windows is sub-par, if there is any support at all. I believe “Kind” is a balanced spot for this project, somewhere in between Minikube and more production-ready distros mentioned above.

When it comes to assembling the pipeline itself, self-hosted GitHub Actions Runners and a Sonarqube server are deployed on the cluster. The whole workflow for the pipeline are as follows:

1. Code is pushed to GitHub repository
2. GitHub Actions Workflow is triggered
3. Sonarqube Static Code Analysis
4. OWASP Depencency-Check Analysis
5. Build Container Image
6. Scan the image via Trivy
7. Update/Create deployment for the app using the new image

2.1 Current State and Next Steps

At this moment, configurations for creating the cluster, Sonarqube and GitHub Actions Runner are completed. It is possible to create and delete the local cluster with very minimal configurations. Additionally, a simple “Hello World” workflow yaml file has been created and tested to see if GitHub Runners are triggered on a push event to the repository, which has been successful.



CankutER is testing out GitHub Actions 🚀 #3

Explore-GitHub-Actions
succeeded 1 hour ago in 5s

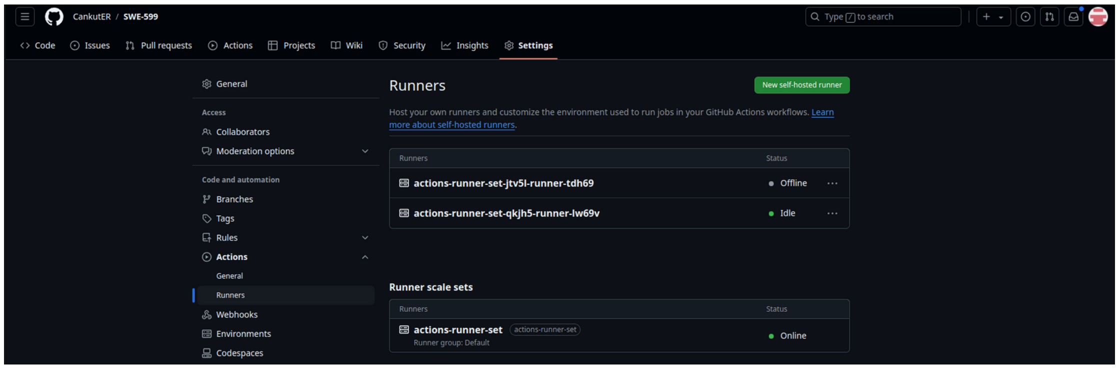
Set up job 0s
Run echo "🎉 The job was automatically triggered by a push event." 0s

```
▸ Run echo "🎉 The job was automatically triggered by a push event."
🎉 The job was automatically triggered by a push event.
```

Complete job 5s

```
Cleaning up orphan processes
```

It is possible to see registered runners under repository Settings ==> Actions ==> Runners.



Runners

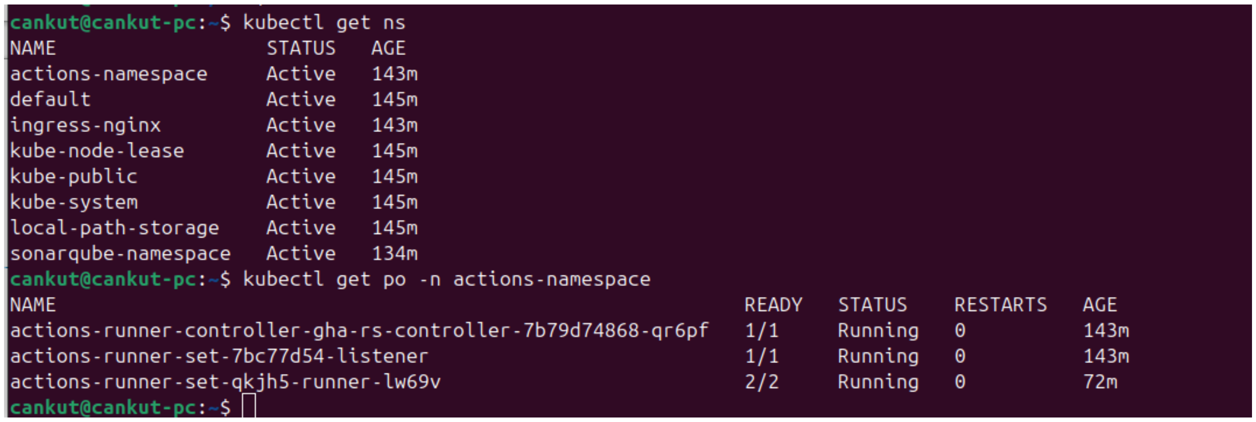
Host your own runners and customize the environment used to run jobs in your GitHub Actions workflows. Learn more about self-hosted runners.

Runners	Status
actions-runner-set-jtv5l-runner-tdh69	● Offline
actions-runner-set-qkjh5-runner-lw69v	● Idle

Runner scale sets

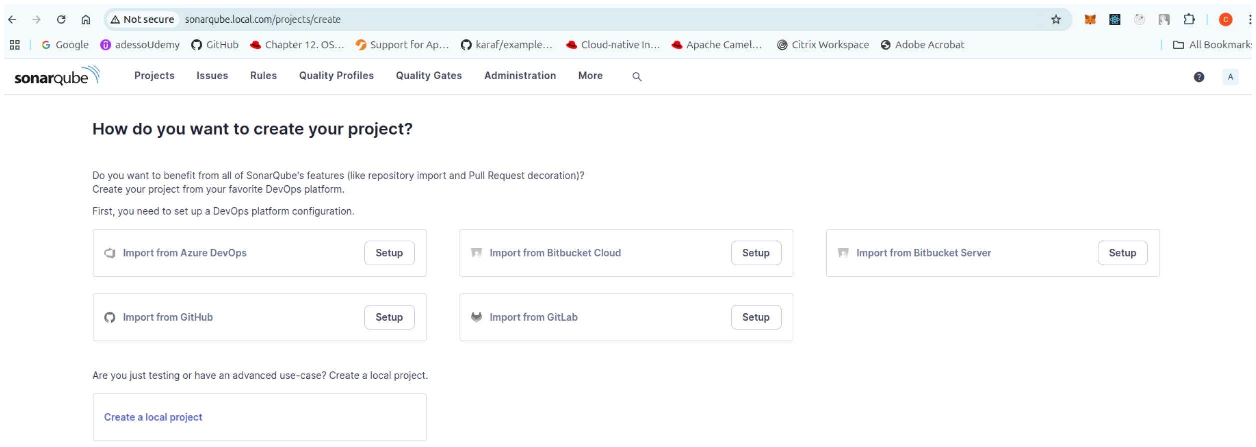
Runners	Status
actions-runner-set (actions-runner-set) Runner group: Default	● Online

Runners on local cluster:



```
cankut@cankut-pc:~$ kubectl get ns
NAME                   STATUS   AGE
actions-namespace      Active   143m
default                Active   145m
ingress-nginx          Active   143m
kube-node-lease        Active   145m
kube-public            Active   145m
kube-system            Active   145m
local-path-storage     Active   145m
sonarqube-namespace    Active   134m
cankut@cankut-pc:~$ kubectl get po -n actions-namespace
NAME                                                  READY   STATUS    RESTARTS   AGE
actions-runner-controller-gha-rs-controller-7b79d74868-qr6pf   1/1     Running   0          143m
actions-runner-set-7bc77d54-listener                  1/1     Running   0          143m
actions-runner-set-qkjh5-runner-lw69v                 2/2     Running   0          72m
cankut@cankut-pc:~$ 
```

Sonarqube dashboard that can be accessed via Ingress configured for local cluster:



Not secure sonarqube.local.com/projects/create

How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)?
Create your project from your favorite DevOps platform.

First, you need to set up a DevOps platform configuration.

Import from Azure DevOps Setup
Import from Bitbucket Cloud Setup
Import from Bitbucket Server Setup
Import from GitHub Setup
Import from GitLab Setup

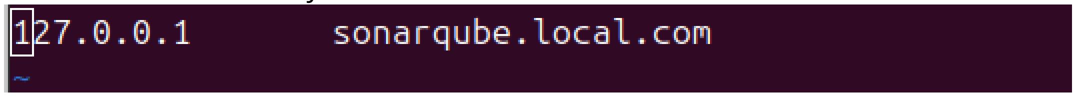
Are you just testing or have an advanced use-case? Create a local project.

Create a local project

3. Reproduction Steps

1) Install the prerequisites (Refer to Section 1.1)
2) Clone the repository (https://github.com/CankutER/SWE-599)
3) cd <repository-path>/Infrastructure
4) terraform init
5) For your target repository to bind Runners, change “githubConfigUrl” in “arc_runner_set_values.yaml” file
6) Create a Personal Access Token in your GitHub account (Account Settings==>Developer Settings==>Personal Access Token==>Tokens(classic))
7) Create an environment variable for Personal Access Token Created(variable name must be the same):
   TF_VAR_github_pat="access token here"

8) Create a dns record for sonarqube.local.com that targets the localhost IP in the operating system’s host file. (This is due to Ingress configured in cluster, you can change the dns name in sonarqube-values.yaml file)
Linux: /etc/hosts
Windows: c:\Windows\System32\Drivers\etc\hosts



```
127.0.0.1       sonarqube.local.com
~
```

9) cd <repository-path>/Infrastructure
10) terraform plan
11) terraform apply

In order to incorporate trivy container scanning and OWASP Depencency-Check into the pipeline, custom GitHub Actions will be used. Initial thought was to also create separate deployments for these resources, however they seem to be temporary instances created for a single scan job and do not seem to have an on-going server. Custom actions create a temporary container to run these scans and produce and artifact for the reports/results.