Kubernetes Secrets,

What are they and How to Use them

rajeshmamuddu · Follow

6 min read · Nov 30, 2023

In this tutorial on Kubernetes (K8s) secrets, we will look at what are they, the types of Kubernetes secrets, how to create one, and more.

Let's dive right into it.

What are Kubernetes Secrets?

Kubernetes Secrets are used to store sensitive information such as passwords, tokens, and certificates in a secure way.

Secrets can be used by Kubernetes deployments, pods, and other resources that require this sensitive data.

We all need someone to keep our secrets safe, in this case, it's in Kubernetes!

How to Use Kubernetes Secrets?

Secrets can be used in several ways within a Kubernetes cluster, such as:

Environment variables in a container

Secrets can be used to pass sensitive data to containers/pods as environment variables. This is a common use case for secrets, as it allows the application to access the data easily.

Files in a container

Secrets can be mounted as files in a container, allowing the application to access the data as a file. This can be useful for applications that expect sensitive data to be stored in a file.

Command-line arguments in a container

Secrets can be passed to a container as command-line arguments. This can be useful for applications that expect sensitive data to be passed as a command-line argument.

Types of Kubernetes Secrets

Other than your best friend, there are four other types of Kubernetes secrets that can be used to store and manage sensitive information within a Kubernetes cluster:

1. Opaque Secrets

Opaque secrets are the most basic type of secret in Kubernetes. They can store arbitrary data, such as binary data or text, as key-value pairs.

Opaque secrets are typically used to store sensitive data such as passwords, certificates, and tokens.

2. TLS Secrets

TLS secrets are used to store TLS certificates and private keys. They are used to secure communication between different services within a Kubernetes cluster, as well as between services and external clients.

3. Dockercfg Secrets

Dockercfg secrets are used to store Docker credentials, such as registry credentials and authentication tokens.

They are used by the Kubernetes container runtime to pull images from private Docker registries.

4. SSH Secrets

SSH secrets are used to store SSH private keys, which can be used to authenticate with other servers or services.

Each of these secret types can be used in different ways within a Kubernetes cluster, depending on the needs of the application and the data being stored.

For example, Opaque secrets are the most commonly used type of secret and can be used to store any type of sensitive data, while TLS secrets are specifically designed to store TLS certificates and keys for secure communication.

How to Create a Kubernetes Secret?

To create and use a Kubernetes Secret, follow these steps:

Step 1: Create a Secret

There are a few ways to create a Kubernetes Secret, but the easiest way is to use the *kubectl create secret* command. Here is an example:

```
kubectl create secret generic my-secret --from-literal=password=abc123
```

This command creates a generic Secret named my-secret and sets the value of the password key to ` `.

You can also create a Secret from a file by using the ` —*from-file*` option:

```
kubectl create secret generic my-secret --from-file=ssh-privatekey=/path/to/pri
```

This command creates a generic Secret named my-secret and sets the values of the `*ssh-privatekey*` and `*ssh-publickey*` keys to the contents of the specified files.

Step 2: Use the Secret in a Pod

To use a Secret in a Pod, you need to add a volume that references the Secret and then mount that volume into the container.

Here is an example:

```
apiVersion: v1
kind: Pod
metadata:
 name: my-pod
spec:
 containers:
 - name: my-container
 image: my-image
 volumeMounts:
 - name: my-secret
 mountPath: /etc/my-secret
 readOnly: true
 volumes:
 - name: my-secret
 secret:
 secretName: my-secret
```

This YAML file defines a Pod that has one container and one volume. The volume is a Kubernetes Secret called `*my-secret*` that is mounted into the container at the path `*/etc/my-secret*`.

Step 3: Use the Kubernetes Secret in a Deployment

To use a Secret in a Deployment, you can add an environment variable that references the Secret.

Here is an example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: my-deployment
spec:
```

```
replicas: 3
template:
metadata:
labels:
app: my-app
spec:
containers:
- name: my-container
image: my-image
env:
- name: DB_PASSWORD
valueFrom:
secretKeyRef:
name: my-secret
key: password
```

This YAML file defines a Deployment that has one container. The container has an environment variable called ` *DB_PASSWORD*` that is set to the value of the ` *password*` key in the Kubernetes Secret called ` *my-secret*`.

Step 4: View Kubernetes Secrets

To view an existing Secret, use:

```
kubectl get secret secret-name
```

The Secret will be printed by Kubectl in base64-encoded format. To fully decode the Secret, use an external base64 decoder (such as the Linux programme base64), and issue commands like:

```
echo $encoded_secret | base64 -d
```

Note that if you've created RBAC restrictions on who can view Secrets, only users with the requisite permissions will be able to access Secrets this way.

You can check that the secret has been successfully created by typing:

```
kubectl get secrets
```

The command shows the list of available secrets — their names, types, number of data values they contain, and their age:

Step 5: Modifying a K8s Secret's Properties

To modify the properties of an existing Kubernetes Secret you can use *kubectl edit*.

The following command shows how you could modify a Kubernetes Secret named *production-secret*:

```
kubectl edit secret production-secret
```

Step 6: Deleting a Kubernetes Secret Using kubectl delete

To delete a Secret, first, use cat or id <name> to check if there are any Secrets in your cluster. Next, use describe <name> to get more information about a specific Secret.

You delete Kubernetes Secrets using the kubectl delete command.

The following example deletes a Kubernetes Secret named production-secret:

```
kubectl delete secret production-secret
```

That's it! You now know how to create and use Kubernetes Secrets.

How Secure are Kubernetes Secrets?

Kubernetes secrets are designed to be secure and provide a mechanism for securely storing and managing sensitive data within a Kubernetes cluster.

When properly configured and used, Kubernetes secrets can provide a high level of security for your sensitive data, including your anniversary date maybe.

Kubernetes secrets are stored as encrypted data in the Kubernetes API server and can only be accessed by authorized users or services who have gained permission.

By default, Kubernetes secrets are encrypted using the AES-256 encryption algorithm, and access to the secrets is controlled by Kubernetes RBAC (Role-Based Access Control) rules.

A common practice is not to provide edit or delete access to someone who does not need them.

In addition, Kubernetes secrets are stored in memory only when they are being used by a pod. Once the pod terminates, the secrets are deleted from memory forever, providing an additional layer of security.

However, it's important to note that Kubernetes secrets are not foolproof, and there are potential security risks that can compromise the security of your secrets.

For example, if an attacker gains access to the Kubernetes cluster or to a pod running an application that uses secrets, they may be able to access the secrets.

In the past, there have been such instances, so take a pinch of salt, no system in the world is 100% fool proof.

Kubernetes Secrets Best Practices

To mitigate these risks, it's important to follow best practices for securing Kubernetes clusters, such as implementing network security controls, using strong authentication mechanisms, and limiting access to sensitive resources.

Additionally, it's important to properly configure and use Kubernetes secrets, such as avoiding storing secrets in environment variables and restricting access to secrets to only the pods that need them.

Note: Follow best industry practices, all the data leaks that ever happened, were due to someone not following the established foolproof protocol.

Originally published at https://humalect.com on July 25, 2023.

Kubernetes | Kubernetes Security | Podsecuritypolicy

Follow

Written by rajeshmamuddu

34 Followers · 45 Following

Skilled Devops Experience in DevOps Engineer with 2+years hands on experience supporting, automation & deployments on AWS, leveraging CI/CD and DevOps processes

Responses (1)

What are your thoughts?

Respond

Abdelfattah Sekak
Nov 30, 2023

I really enjoyed your insights.

1 reply Reply

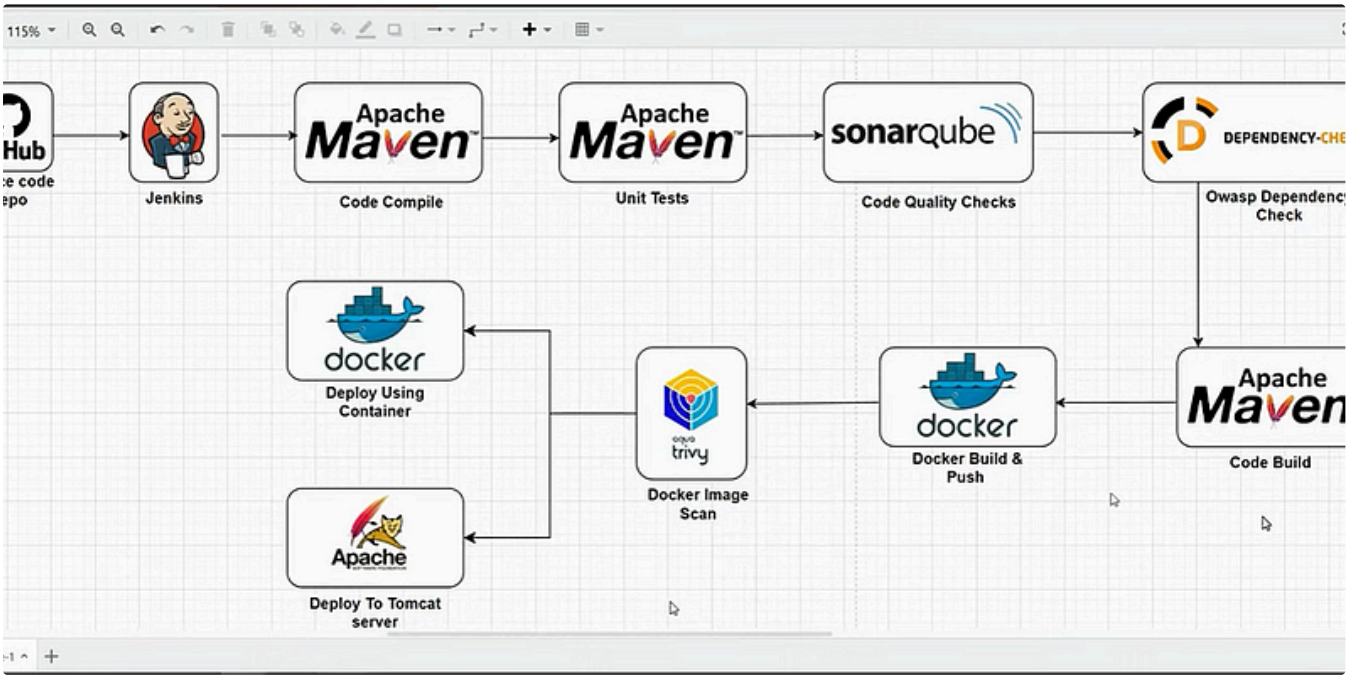
More from rajeshmamuddu

rajeshmamuddu

Setup minikube with docker driver

In this article you will get step by step how to setup minikube with docker

Oct 11, 2023 1

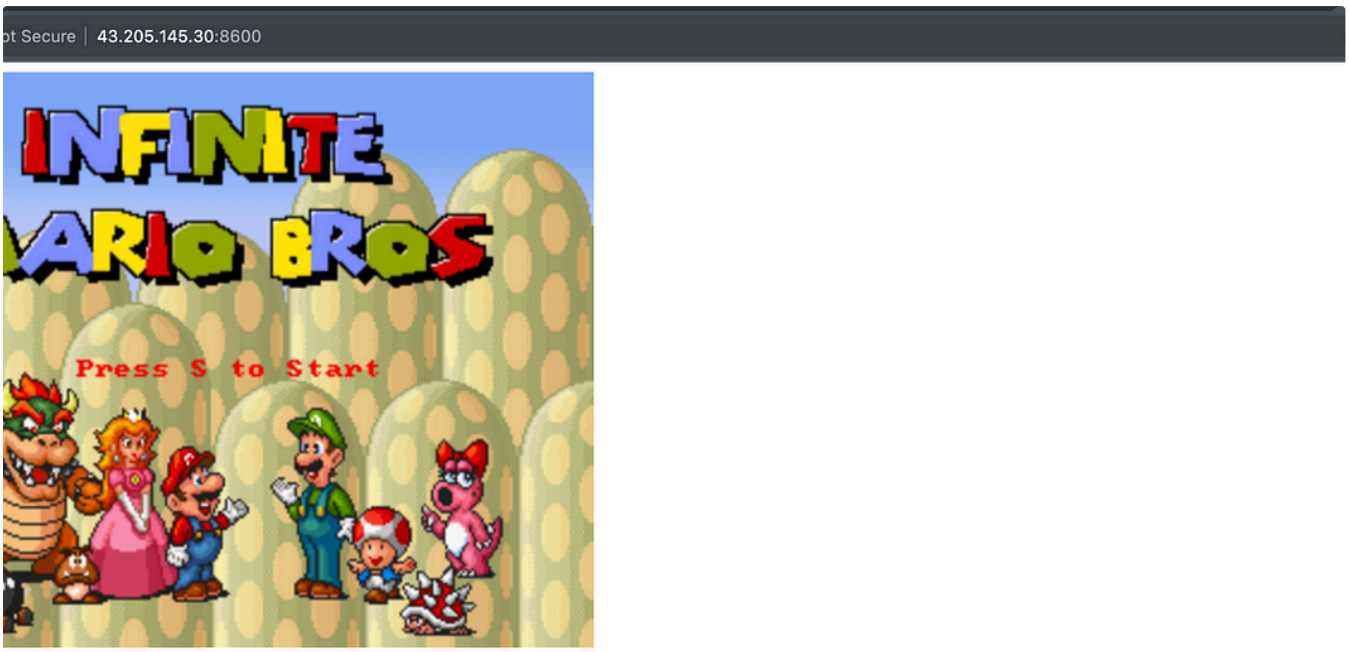


rajeshmamuddu

CI/CD DevSecOps Pipeline for Deployment of Petclinic Application

A step-by-step guide to deploying a Java-based Pet Clinic application on Tomcat Server using Jenkins as a CI/CD tool

Oct 29, 2023 24 1

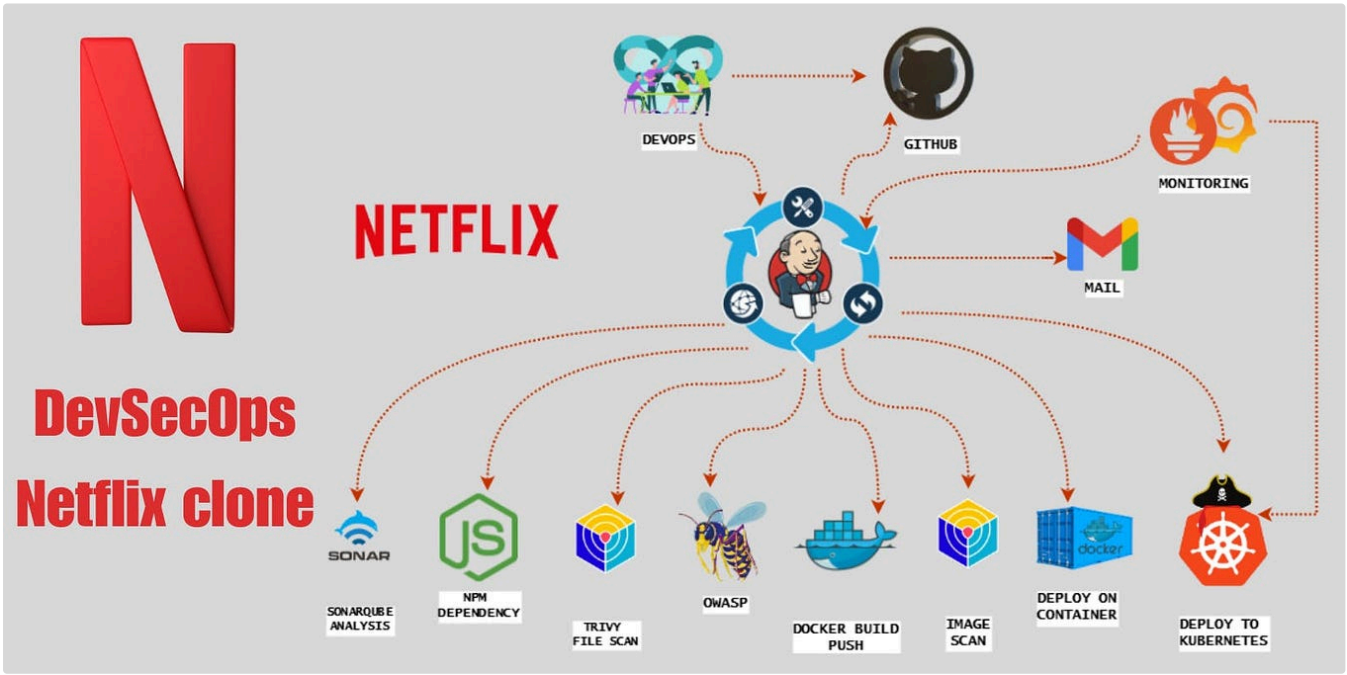


rajeshmamuddu

Deploying Mario game on Docker container

Take an Ec2 (ubuntu-linux)instance machine with instance type as t2micro from AWS console

Oct 26, 2023 3



rajeshmamuddu

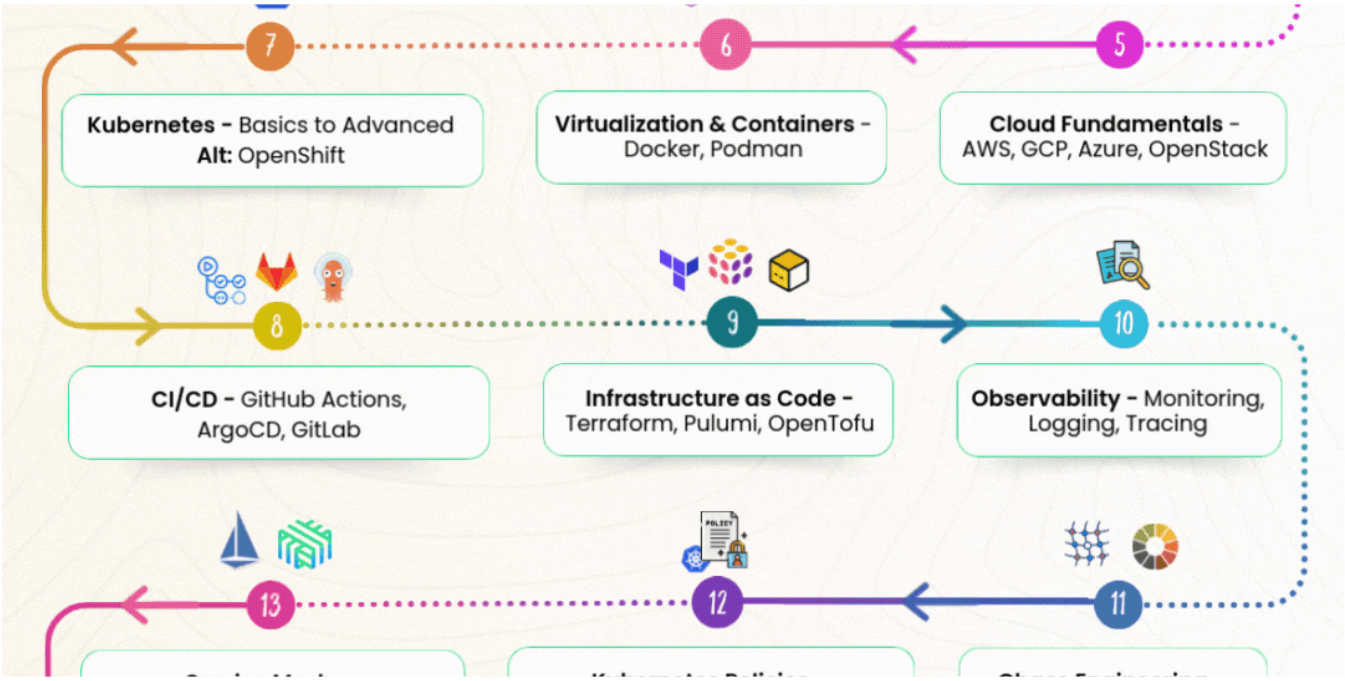
DevSecOps : Netflix Clone CI-CD with Monitoring tool grafana &Prometheus

we will be deploying a Netflix clone. We will be using Jenkins as a CICD tool and deploying our application on a Docker container and...

Oct 22, 2023 10

See all from rajeshmamuddu

Recommended from Medium



Rohit Ghumare

DevOps Roadmap 2025

In today's rapidly evolving tech landscape, DevOps has become more than just a buzzword—it's a crucial methodology that bridges the gap...

Jan 15 174 4

Rajesh k

Kubernetes Operators vs. Helm Charts: Understanding the Differences and Use Cases

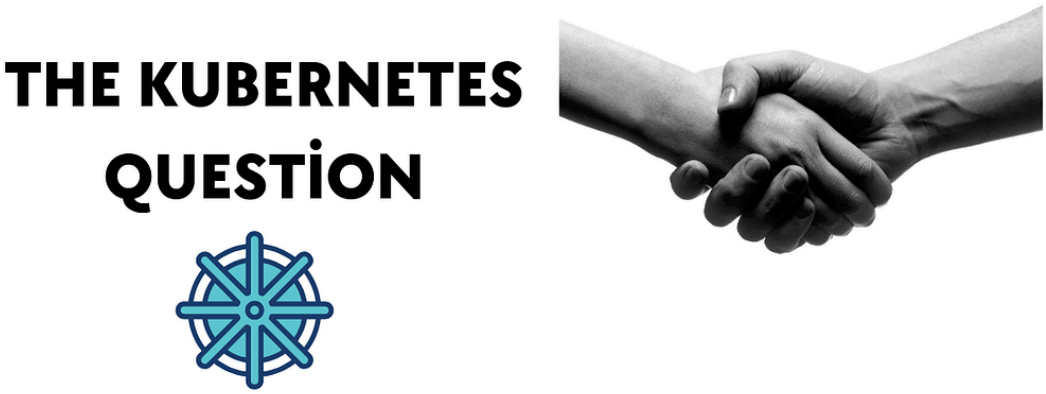
As Kubernetes continues to evolve as the de facto standard for container orchestration, various tools and methodologies have emerged to...

Aug 4, 2024 2

Lists

Natural Language Processing

1893 stories · 1553 saves



In Level Up Coding by Rahul Sharma

I Asked This Kubernetes Question in Every Interview—And Here's the Catch

When I interview candidates, I prefer a real-world problem that demonstrates the candidate's practical expertise with Kubernetes to be...

Jan 15 315 11

Alfonso fortunato

Simplifying Secret Management with Azure Key Vault and External Secret Operator

Secret handling has always been a pain point for me over the past few years. Most of the time, I ended up using services with features l...

Sep 8, 2024 15

The Devops Girl

How to Update Your Amazon EKS Cluster from 1.27 to 1.28: A Step-by-Step Guide

If you're managing an Amazon EKS (Elastic Kubernetes Service) cluster, keeping your Kubernetes version up to date is crucial for...

Aug 19, 2024 110

Zudonu Osomudeya

15 Automation Scripts for Monitoring and Logging Every DevOps Engineer Must Know

Scripts for Monitoring and Logging

Jan 13 34 1