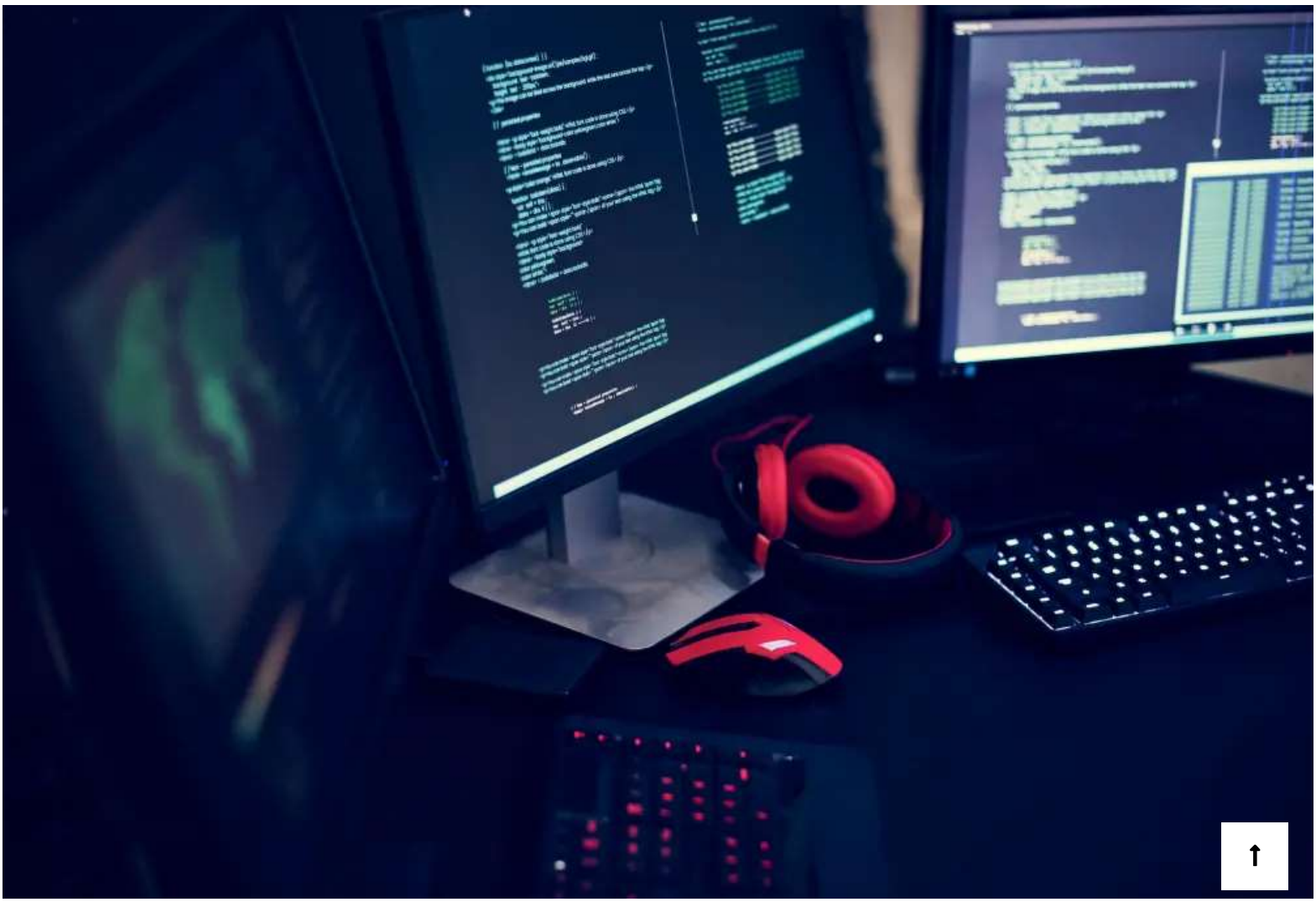
How to Encrypt Kubernetes Secrets with Sealed Secrets?



| August 17, 2022 (https://blog.knoldus.com/how-to-encrypt-kubernetes-secrets-with-sealed-secrets/) Ankur

ıttps://blog.knoldus.com/author/ankurknol/) Studio-DevOps (https://blog.knoldus.com/category/devops/)
› Devops (https://blog.knoldus.com/tag/devops/), git (https://blog.knoldus.com/tag/git/), helm (https://blog.knoldus.com/tag/helm/), kubernetes
ıttps://blog.knoldus.com/tag/kubernetes/), sealed secrets (https://blog.knoldus.com/tag/sealed-secrets/), secrets (https://blog.knoldus.com/tag/secrets/),
ecurity (https://blog.knoldus.com/tag/security/)

Table of contents

- Why Sealed Secrets?
- How it Works?
- Setting Up Sealed Secrets Components
  - Kubeseal
  - Sealed Secrets Controller
- Sealing the SecretsCreate Secret
- Managing the Sealing Key
- Conclusion

eading Time: 5 minutes

Why Sealed Secrets?

s we know, Secrets in Kubernetes (https://kubernetes.io/fr/docs/concepts/configuration/secret/) are used to store sensitive data, like password, keys, ertificates and token. Secrets are encoded in base64 and automatically decoded when they are attached and read by a Pod.

secret in Kubernetes cluster is encoded in base64 but not encrypted!

hese data are “only” encoded so if a user has access to your secrets, he can simply execute a `base64 decode` command to see your sensitive data `kubectl get secret my-secret -o jsonpath="{.data.password}" | base64 --decode`).

s the secrets aren’t encrypted, it can be unsecure to commit them to your Git repository.

ealed Secrets is a solution to encrypt your Kubernetes Secret into a `SealedSecret`, which is safe to store – even to a public repository. The ealedSecret can be decrypted only by the controller running in the target cluster and nobody else.

How it Works?

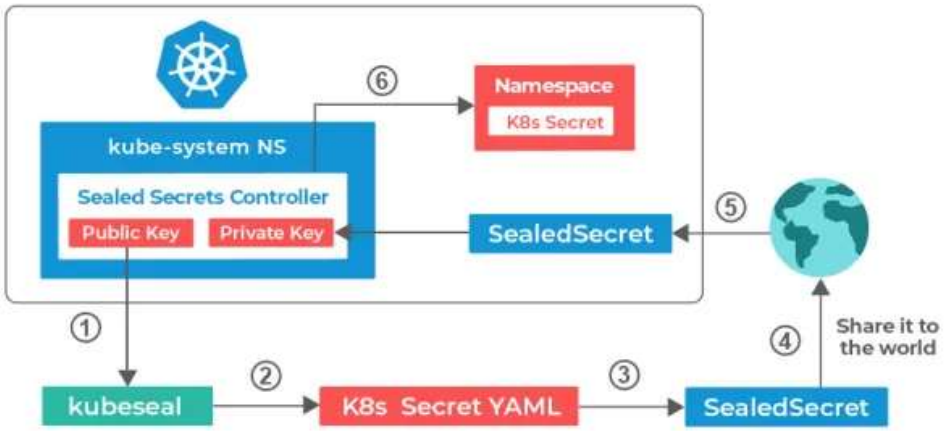
he underlying principle of Sealed Secrets is the usage of public key cryptography. The public certificate is used for sealing secrets. The private key ıe controller has is used for decrypting the sealed secrets.

he name space is used during the encryption process by default. Thus, two same secrets on different name spaces will have a different set of ncrypted data within it.

he secret name is also used during the encryption by default. This design decision disallows Sealed Secrets resources to be renamed. This improves ıe security as RBAC (role based access control) can enforce limitation of secret access by name and Sealed Secrets follows it by disallowing the ecret to be renamed.

here are 3 types of scopes that can be used:

- *strict* scope requires both name space and secret name as part of the encryption process.
- *namespace-wide* scope requires only a secret name as part of the encryption process.
- *cluster-wide* doesn’t require either.



Setting Up Sealed Secrets Components

Kubeseal

`kubeseal` is a CLI client for sealing/encrypting k8s secrets.

```
wget https://github.com/bitnami-labs/sealed-secrets/releases/download/v0.18.0/kubeseal-0.18.0-linux-amd64.tar.gz
tar -xvf kubeseal-0.18.0-linux-amd64.tar.gz
sudo mv kubeseal /usr/local/bin/kubeseal
```

Sealed Secrets Controller

Current deployment process can be done manual helm install command or kubectl on targeted cluster:

Installing via helm chart

```
helm repo add sealed-secrets https://bitnami-labs.github.io/sealed-secrets
helm dependency update sealed-secrets
helm install sealed-secrets sealed-secrets/sealed-secrets \
  --namespace kube-system \
  --version 2.2.0
```

Installing via Kubectl

```
wget https://github.com/bitnami-labs/sealed-secrets/releases/download/v0.18.1/controller.yaml
kubectl apply -f controller.yaml
```

You can ensure that the relevant Pod is running as expected by executing the following command:

```
kubectl get pods -n kube-system | grep sealed-secrets-controller
```

ealing the Secrets

reate Secret

he code for Kubernetes Secret is given below. Save the code in a file name **secrets.yaml**.


```
apiVersion: v1
kind: Secret
metadata:
  creationTimestamp: null
  name: my-secret
  namespace: test-ns
data:
  password: cG9zdGdyZXM=  # <- base64 encoded postgres
  username: YWRtaW5AcG9zdGdyZXM= # <- base64 encoded admin@postgres
```

ere the username and passwords are base64 encoded. Please remember, you are not going to execute this file. You will generate encrypted data or this secret using Kubeseal and execute the sealed secret on the Kubernetes cluster.

ou can retrieve the generated public key certificate using kubeseal and store it on your local disk:

```
kubeseal --fetch-cert > public-key-cert.pem
```

ubeseal encrypts the Secret using the public key that it fetches at runtime from the controller running in the Kubernetes cluster. If a user does not ave direct access to the cluster, then a cluster administrator may retrieve the public key from the controller logs and make it accessible to the user.

SealedSecret CRD is then created using **kubeseal** as follows using the public key file:

```
kubeseal --format=yaml --cert=public-key-cert.pem < secret.yaml > sealed-secret.yaml
```

he generated Secret with Base64 encoded value for **username** and **password** keys is as follows:

```
apiVersion: bitnami.com/v1alpha1
kind: SealedSecret
metadata:
  creationTimestamp: null
  name: my-secret
  namespace: test-ns
spec:
  encryptedData:
    password:
AgCEP9MHaeVGelL3jSbnfUF47m+eR0Vv4ufzPyPXBvEjYcJluI0peLgilSqWUMnDLjp00UJq186nRYufEf7Bi1Jxm39t3nDbW13+wTSbbl
Vzb9A2iKJI2VbxgTG/IDodNFxFWdKefZdZqVct2hTsuwlivIxpdxDZtcND4h+Cx8YFQMUpT5oO26oqISzRTh5Ewa6ehWtv6krfeFmbMQCF
70eg0yioe6Op0YaQKloiFVInc1JK5KTR5iQYCeKb2R0ovKVa/lipbqjHCYSRhLR/3q5wz7ZuWz7g7nq6G9Q7o1pVv3udQYUvp2B6XvKINe
zc85wbhGqmuz5kcZUa36uF+eKMes6UdPcD7q58ndaj/0KWozdTAukIOblV7mrUaK8Q45GIf+JqaBfzVt52INMT07P4MId/KB31sZDeE+Ow
EXhCDVTBA1xSRM0U9NjxDDb+mwUzxHNZHL1sY8M1YCoX+rr6n1+yW1HG42VHLCRzeBa2V31OFuQTNjoNxDEfUq+CSTRNDCmt8UvercSkgy
M3mBa6JpHdkyS1lpqyEJDYKM1YvVRrjVvq1qGTF5dOCx6x3ROXnZtA3NBIafTu0+pHovVo+X7nUkl7hyupd0KKzBG+afgNpYQOxeuei5A+
o++o92G51exxk2v4bQt6ANYBxMlvT0LdBUW9e/L2y+TuNAHL23Xa/aTq1lagNBi9UTowX0lx0br2CqDbKg==
    username:
AgArwZm3qh83Fpzles1r/PjTDKQ2/SZ482IKC84T72/kI4M29aG2VT4SCXcqbmtVDYuVUN0wTbsFYsnwY1DSRrL4oup2xRg6N34IxHjj0Z
tF1q0YtBKIM/DPcF2bBVAYc9/vOI0L3+VVSF9r93XYEMUWX6hY9eHa8VUHBM/Y65Sj3Il7Pmx/qoEcZ+e9UJhqWEJPotz6W5OMh/Al7QPJ
ZknwUulM4coZ3C0J4TmrBVexPturcRCimDEQnd9UitotnGDoNAp2O28ovhXoImNsJBhNK5LykesRxEfIp4UJOb3I0CpLdoz9khEcb2r31j
+KTtxifLez7Rg3Pq7BGpR3EKC3INZWrR8S/aUm5u/dP12ELgW3ng4WbafRitrZcHhLFZkHma/Er8miFbuTXvpFcXE1q+BnG2vIs4kHSl2Q
cP32HPGKHJJt0KEd1dUJrXXTjS9eXHJ2KsA5DZk4TcFA5dPAG76ZdKo0GCIQwvNeT0Ao4ntqmeOiijAQqmhXdCtD2WVavXi54h0f8F2ue6
b0mBFCqTGKZyypjbXznzB/MPAZxgIu+UWQzV1CczwKlitPy638s/9iSan2/u2rhKu2SP0JFMZ6pPnfO51nMpDHtCDGFc1unjsjM4ZpnNXt
aQJJmXo7Hw0L4dW2/N3uxCfxNtmYuBxE1t4GCefSUCTIleDgmAbB00nKkja+ml9bidcxawlIgHnoq/XNCqy2R3PkEw==
  template:
    data: null
    metadata:
      creationTimestamp: null
      name: my-secret
      namespace: test-ns
```

ote that the keys in the original Secret—namely, **username** and **password**—are not encrypted in the SealedSecret, only their values are. You may hange the names of these keys, if necessary, in the SealedSecret YAML file and still be able to deploy it successfully to the cluster. However, you annot change the **name** and **namespace** of the SealedSecret. Doing so will invalidate the SealedSecret because the name and namespace of the riginal Secret are used as input parameters in the encryption process. 4

he YAML manifest that pertains to the Secret is no longer needed and may be deleted. The SealedSecret is the only resource that will be deployed t he cluster as follows:

```
kubectl apply -f sealed-secret.yaml
```

nce the SealedSecret CRD is created in the cluster, the controller becomes aware of it and unseals the underlying Secret using the private key and eploys it to the same namespace. This is seen by looking at the logs from the controller:

Managing the Sealing Key

Without the private key that is managed by the controller, there is no way to decrypt the encrypted data within a SealedSecret. Run the following command in order to retrieve the private key from the cluster:

```
kubectl get secret -n kube-system -l sealedsecrets.bitnami.com/sealed-secrets-key -o yaml > master.yaml
```

Now, let's first delete the installation of the controller, the Secret that it created which contains the private key, the SealedSecret resource named **mysecret** as well as the Secret that was unsealed from it.

```
kubectl delete secret mysecret
kubectl delete SealedSecret mysecret
kubectl delete secret -n kube-system -l sealedsecrets.bitnami.com/sealed-secrets-key
kubectl delete -f controller.yaml
```

Now, put the Secret containing the private key back into the cluster using the **master.yaml** file and redeploy the SealedSecret CRD, controller and RBAC artifacts on your EKS.

```
kubectl apply -f master.yaml
kubectl get secret -n kube-system -l sealedsecrets.bitnami.com/sealed-secrets-key
kubectl apply -f controller.yaml
```

View the logs of the newly launched controller pod. Note that the name of the controller pod will be different in your cluster. As you can see from the logs, the controller was able to find the existing Secret **sealed-secrets-keyb2fkv** in the *kube-system* namespace and therefore does not create a new key pair.

```
kubectl logs -f sealed-secrets-controller-59ddc747c4-djsbc -n kube-system
```

Now, let's redeploy the SealedSecret and verify that the controller is able to successfully unseal it.

```
kubectl create -f sealed-secret.yaml
kubectl logs -f sealed-secrets-controller-59ddc747c4-djsbc -n kube-system
```

Conclusion

Store your sensitive data in a Kubernetes Secret object is a common practice, but don't forget that a Secret is only encoded and not encrypted. So if you want to store them in a Git Repository , you'll need to find a secure solution. Sealed Secret helps you to see that it can be a solution that you can try in your side.

Written by Ankur (https://blog.knoldus.com/author/ankurknol/)

I am an DevOps engineer having experience working with the DevOps tool and technologies like Kubernetes, Docker, Ansible, AWS cloud, prometheus, grafana etc. Flexible towards new technologies and always willing to update skills and knowledge to increase productivity.

COMPANY

About Knoldus (https://www.knoldus.com/about/)

About NashTech (https://nashtechglobal.com/about-us/)

About Nash Squared (https://www.nashsquared.com/about)

Sign up to our newsletter

Certificates

Partners