Infrastructure Internship Task 2025

Background

Bitnami's sealed-secrets controller (also known as kubeseal) is a Kubernetes controller that enables the creation of encrypted secrets. These encrypted secrets can be safely stored in insecure storage mediums, such as Git, allowing them to be source controlled.

Internally, the sealed-secrets controller operates based on asymmetric encryption. Here's a breakdown of how it works:

1. **Key Generation:** The controller generates a public-private key pair. The private key is stored securely within the Kubernetes cluster, while the public key is made available to users.
2. **Encryption:** Users utilize the `kubeseal` command-line tool to encrypt their secrets using the public key. The output is a "SealedSecret" object, which contains the encrypted secret and metadata.
3. **Storage:** The SealedSecret object can be safely stored in version control or other insecure locations, as it only contains the encrypted secret.
4. **Decryption:** When the SealedSecret object is applied to the Kubernetes cluster, the sealed-secrets controller uses its private key to decrypt the secret and create a standard Kubernetes Secret object.
5. **Access:** Applications within the cluster can then access the decrypted secret from the Kubernetes Secret object as usual.

Key Points:

- The private key never leaves the Kubernetes cluster, ensuring that only the controller can decrypt the secrets.
- The SealedSecret objects are portable and can be shared without compromising the security of the secrets.
- The sealed-secrets controller automates the decryption process, making it seamless for applications to access the secrets.

Let's say you have a Kubernetes secret containing a database password. You don't want to store this password in plain text in your code repository, as it would be a security risk.

This is where kubeseal comes in. You can use the kubeseal CLI to encrypt the secret using a public key. The output will be a SealedSecret object that you can safely store in your code repository.

When you deploy your application to Kubernetes, the SealedSecret controller will use its private key to decrypt the secret and make it available to your application.

Example:

```
Unset
kubeseal -o yaml < secret.yaml > sealed-secret.yaml
```

The `sealed-secret.yaml` file can now be safely stored in your code repository.

Sealing Key Rotation:

Every 30 days (by default), kubeseal performs a key rotation, in which it generates a new key pair and updates the controller. New secrets use the new key, while existing SealedSecrets use the older key(s), converting older SealedSecrets requires manual re-encryption.

Task

Building on the background we've established, explain how you would implement a mechanism for automating the process of re-encrypting all SealedSecrets in a cluster as an additional feature to the kubeseal CLI. Detail the steps and considerations involved in such an implementation, keeping in mind the potential challenges and benefits of integrating this functionality into the existing kubeseal workflow.

Requirements

- All documentation must be provided in MarkDown format.
- Ability to identify all existing SealedSecrets in the Kubernetes cluster.
- Capability to fetch all active public keys of the SealedSecrets controller.
- Functionality to decrypt each SealedSecret using the existing private keys.
- Feature to re-encrypt the decrypted secrets using the latest public key.
- Mechanism to update the existing SealedSecret objects with the re-encrypted data.
- *(Bonus)* Logging or reporting mechanism to track the re-encryption process and any errors.
- *(Bonus)* Consideration for handling large numbers of SealedSecrets efficiently.
- *(Bonus)* Ensure security of the private keys throughout the process.
- Provide clear documentation on how to use the re-encryption feature.

Important Notes

- The task requires **only** a detailed plan of action in MarkDown format.
- The task **does not require** actual code writing, you are, however, free to write any code you would like to support your documentation.
- The plan of action provided **must** build upon the existing code of kubeseal CLI.

How to Submit Your Response:

After you’re done with the task, put your documentation in a **private** GitHub repo, invite the following GitHub users, then send the link of the repo to the same thread of emails: nabbas@instabug.com

Invitees:

- mohamed-essam
- Mohab-Elhawawshy
- kasimeka

Best of Luck!