Secrets Store CSI Driver - Rotation

Purpose of Rotation Reconciler

The Rotation Reconciler is to be a component that will be able to call the Provider referenced by a Secrets Store CSI Driver for the purpose of pulling down the latest secrets from a given vault.

Definitions
SPC - Secret Provider Class
CRD - Custom Resource Definition
SPCPS - Secret Provider Class Pod Status

Git Branches for Rotation Reconciler

- Secrets Store CSI Driver Provider for Azure
    - https://github.com/bdlb77/secrets-store-csi-driver-provider-azure/tree/rotation_reconciler
- Secrets Store CSI Driver
    - https://github.com/bdlb77/secrets-store-csi-driver/tree/rotation_binding

Additional CRD Design

- Added CRD will be the SPC Binding
    - Binding a Pod to a SPC
    - Binding stacks **status**
- This Binding will track the **status** of the secrets in the cluster. The properties tracked can be **objectNames, objectVersions, objectAlias, objectType.**
- The name of the binding can be deterministic (<pod name>-<namespace>-<spc name>.)
- Service Account For CSI Driver (Reconciler) to be able to do fetch requests to SPC, SPC Binding, and Pods.
    - GET - SPC Binding
    - GET / Patch - SPC
    - GET - Pod

Rotation with Secret Provider Class Pod Status

Notable Changes

- Add Permissions to Fetch Single Pod to Roles.
- Secrets Rotation Reconciler Mounted as Sidecar
- SPCPS tracks running Object Versions on that Pod.
- Azure Provider Work
    - Ability to receive currentObject Version payload as another argument
    - Compare Newly Fetched Secret Store Objects to CurrentObjects
    - Condition to write only if version changes
        - Also Write .metadata folder which includes files with current object version
- Reconciler reads .metadata folder for object versions & compares
- Reconciler updates Status on SPCPS with new object versions

Possible further Features

- A Trigger Interface to handle different Reconciliation Options
    - Eg. Interval / Polling, Event Based

Concerns

- Failure Condition - Reconciler Goes down
    - Secrets won't be updated for pods that have not been processed
- Race Conditions - Secret Pairing
    - If you have Sets of Secrets That must be together (DB Password & DB User)
    - After Half of secrets get rotated on pods, and DB Password is rotated again
    - Already evaluated Pods will have old DB Password and will need to wait for next polling interval
    - **Guidance**:
        - If secrets are paired, they should be written together.
        - Suggest to have 2 working secret pairs
        - If burst of updates -> buffer incoming events & issue 1 event
    - **Buffer**
        - create a window of opportunity for receiving related events.. Then emit 1 event at end of window
        - Open up stream for incoming events, if tail has not been changed/updated for X time
        - emit event / notification for pod rotation (deployment or something)
        - **Reconciler WorkFlow**
        - ** This would require some external controller to handle events emitted from the pods? How to capture all events from pods that are notifying for their need to rotate? **
- (X* N) Amount of Secret Store API Requests - X Amount SPCPS and N Amount of Secrets Per SPC

- If Pod is defined with a deployment, must notify it's deployment
- Will have Many Events by X amount of Pods needing Rotation. They will not come at the same time. Will need to think of some buffer period for incoming events, and then emit one larger event to rotate all necessary pods / deployments.

1. Reconciler Fetches List of SPCPS
2. Loop through list of SPCPS

```
apiVersion: secrets-store.csi.x-k8s.io/v1alpha1
kind: SecretProviderClassPodStatus
...
status:
 mounted: true
 objects:
 - objectName: databasePassword
   objectVersion: fd9aj3mdfa9xj342
 - objectName: storageToken
   objectVersion: k5dfab7adfx
 podName: nginx-secrets-store-inline
 secretProviderClassName: azure-spc
 targetPath:
/var/lib/kubelet/pods/2eefc285-8d99-459e-8030-8380
fcbeebce/volumes/kubernetes.io~csi/secrets-store-i
nline/mount
```

   a. Set objects array (SPCPS) in Provider Payload
   b. Set targetPath (SPCPS) in Provider Payload
   c. Using secretProviderClassName -> Fetch SPC
      i. Set Provider (SPC) in Provider Payload (provider type)
      ii. Set params (SPC) in Provider Payload
   d. Using podName from SPCPS -> Fetch Pod
      i. If Payload.Params.UsePodidentity == false
         1. Loop through pod volumes
            a. If Pod has CSI volume && it's secrets-store Volume
               i. Get NodePublishSecretRef from volume.
   e. Using NodePublishSecretRef -> Fetch K8sSecret Secrets Store Authentication Creds
      i. Add K8s Secrets to Payload

f. Fetch Provider Binary ****WORK IN PROVIDER****
    i. Provider compares currentObjects in Payload to Newly Fetched From SecretsStore
    ii. If NewlyFetched Version == currentObject Version -> Skip writing.
    iii. If NewlyFetched Version != currentObject Version
        1. Write to TargetPath
            a. .metadata Folder with Files containing Current Object Versions.

```
root@aks-linux-17332296-vmss000002:/var/lib/kubelet/pods/2eefc285-8d99-459e-8030-8380fcbeebce/volumes/kubernetes.io~csi/secrets-store-inline/mount# ls -al
total 12 drwxrwxrwt 3 root root  100 Jun 18 21:31 .
drwxr-x--- 3 root root 4096 Jun 18 21:28 ..
drwxr-xr-x 2 root root   80 Jun 18 23:37 .metadata
-rw-r--r-- 1 root root    7 Jun 18 23:44 DATABASE_PASSWORD
-rw-r--r-- 1 root root   10 Jun 18 23:44 STORAGE_TOKEN

secrets-store-inline/mount/.metadata# ls -ll
total 8
-rw-r--r-- 1 root root 17 Jun 18 23:37
databasePassword
-rw-r--r-- 1 root root 12 Jun 18 21:32 storageToken

secrets-store-inline/mount/.metadata# cat
databasePassword
=> fd9aj3mdfa9xj342
```

g. Reconciler Reads .metadata folder for new object versions
h. Set newObjectsFlag to false
i. Reconciler compares newObjectVersions to currentObjectversions
j. If newObjectVersion != currentObjectVersion
    i. Update SPCPS with new Object Versions
    ii. newObjectsFlag = true
k. If newObjectsFlag
    i. If pod has owner ref of Deployment
        1. Emit Event to Deployment that New Secret Version is Present
    ii. Else

            1. Emit Event to Pod That New Secret Version is present.
3. **External Entity - Events Controller (Possible Approach)**
    a. Events Controller is triggered at end of Reconciler
    b. Events Controller Listens for a window of X time
        i. Records events emitted from Pods and Deployments with Flag of "New Secret Version"
        ii. Once the Tail Event has not changed for N amount of time
            1. **OPTION**: Fire off a single event (or condensed amount) to notify k8s controller to rotate deployments and pods recorded from events.
            2. **OPTION**: Notify User of pods and deployments that need to rotate.

Additional Questions:

1. How will application pods consume the updated content. The driver will update the content in the target path and k8s secret based on the update in the external secrets store and notify the pod using events. The 2 options -
    a. Force restart the pod to pick up the updated object content/replace the env var. This can be expensive in terms of sandbox deletion and creation.
    b. Pods reading the content from files in the mnt path have watchers on the file. This would be an application change if that's not how it is today. However if the pod is consuming the content as env var, this can't be done and (a) is the only option.

Notes:

- Depending on how the application consumes the secret data:
    1. Mount k8s secret as a volume: use rotation feature + sync k8s secrets feature in secrets store csi driver, application will need to watch for changes from the mounted k8s secret volume
    2. read the data from container's filesystem: use rotation feature in secrets store csi driver, application will need to watch for the change from the volume mounted by the csi driver
    3. environment variable: restart the pod
- Updating k8s secret as part of the secret rotation, Ingress controllers like nginx can hot reload their certs store if the K8s TLS secret is updated
    - Tested with stable/nginx-ingress

Revised Design for Rotation Reconciler

The revised design is based on using gRPC for driver-provider communication. gRPC client support for driver has been implemented in https://github.com/kubernetes-sigs/secrets-store-csi-driver/pull/280

Rotation implementation PR based on the revised design - https://github.com/kubernetes-sigs/secrets-store-csi-driver/pull/303

Notable changes:

1. RBAC permissions to get, list and watch pod resources
2. /var/lib/kubelet mounted in the provider pods in addition to being mounted in the driver
3. New objects field in SPCPS that depict the current object id (UID) and version that's mounted in the pod and synced as K8s secret

```
apiVersion: secrets-store.csi.x-k8s.io/v1alpha1
kind: SecretProviderClassPodStatus
...
status:
 mounted: true
 objects:
 - id: secret/secret1
   version: 3f0bdaf9d863489286966b30f1f607df
 - id: secret/secret2
   version: 409b2cfef1c7485db26bed96dbe769c1
 - id: secret/ingress-tls-pfx
   version: 31aa2f8012934709bfe7876fc66d60f4
 podName: nginx-55d5d76664-2zwb9
 secretProviderClassName: azure
 targetPath:
/var/lib/kubelet/pods/4ca452b8-d953-42bb-ba7c-521c0e86a4a5/volumes/kubernetes.i
o~csi/secrets-store-inline/mount
```

Rotation Flow:

Rotation reconciler runs a periodic loop. The duration for the loop can be configured using `"--rotation-poll-interval=3m"`

1. Rotation reconciler fetches all the SPCPS created for the node.

a. This is done using an informer with filtered watch on the node name label configured in SPCPS during the creation.

```
b.  labels:
c.    internal.secrets-store.csi.k8s.io/node-name: kind-control-plane
```

2. Loop through the list of SPCPS for the node.
    a. Fetch the SPC referenced in the SPCPS
    b. Get the pod the SPCPS is referencing to
        i. [Implementation detail] The pod is fetched from the informer cache. The pod informer is configured to only listwatch pods for the current node on which the driver pod is running (using ***spec.nodeName=<node name>***)
    c. If the pod references a NodePublishSecretRef, fetch the secret the pod is referencing to
    d. Generate the proto mount request with parameters, target path, secrets and current object versions as cached in SPCPS
    e. Invoke the provider gRPC MountRequest
        i. The provider uses the current object versions to check if the latest versions are already mounted in the pod. If not, the mounted content is updated.
        ii. Provider returns the new object versions as part of the GRPC MountRespone.
    f. If there are new object versions in the MountResponse, update the versions in the SPCPS
    g. If the SPC contains secretObjects, patch the corresponding Kubernetes secret data with the updated mount content