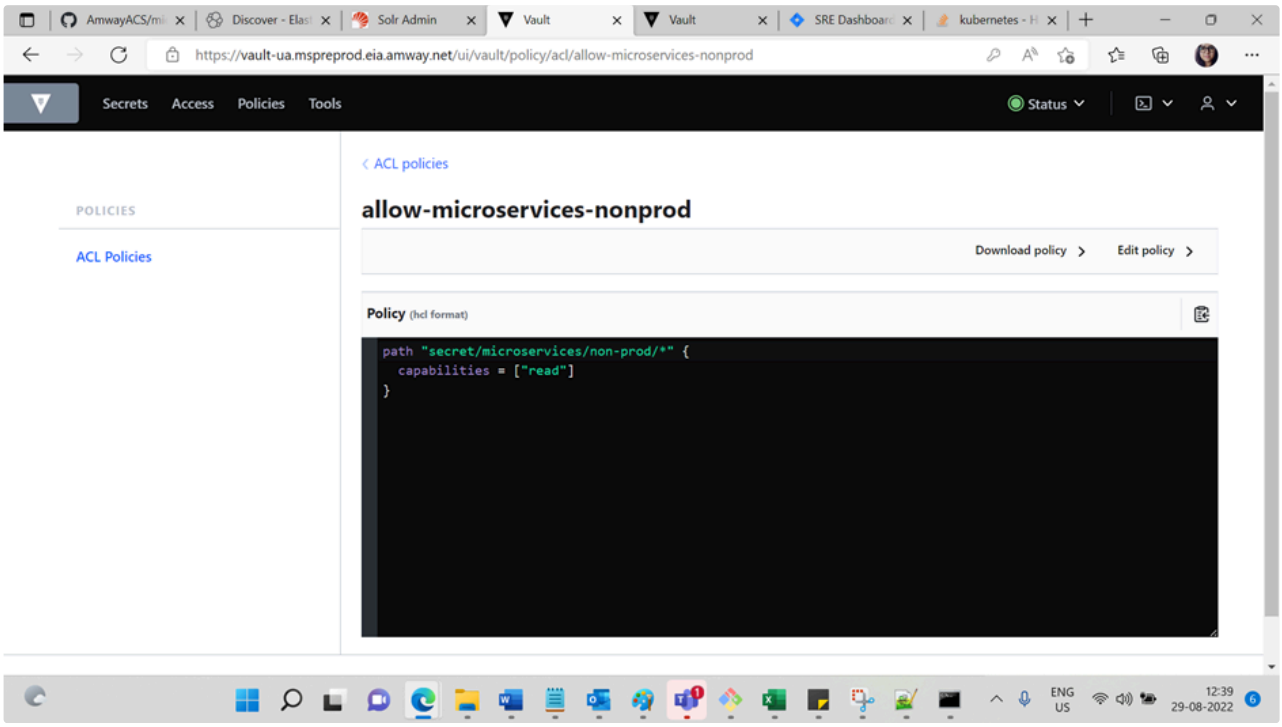
Use Hashicorp Vault to provide secrets for EKS

This tutorial describes using Vault as a Secret Storage for microservices deployed with Helm to AWS EKS.

This particular example describes whole process for one of the Ukraine microservices. This HOWTO does not cover installing Vault by itself, but rather how to configure existing Vault instance to serve as a secret source for EKS. It was tested with EKS, but should work just fine with any other Kubernetes cluster.

Connecting Vault with EKS

1. Create a policy to allow read access to secrets in Vault
   Name your policy: allow-microservices-nonprod



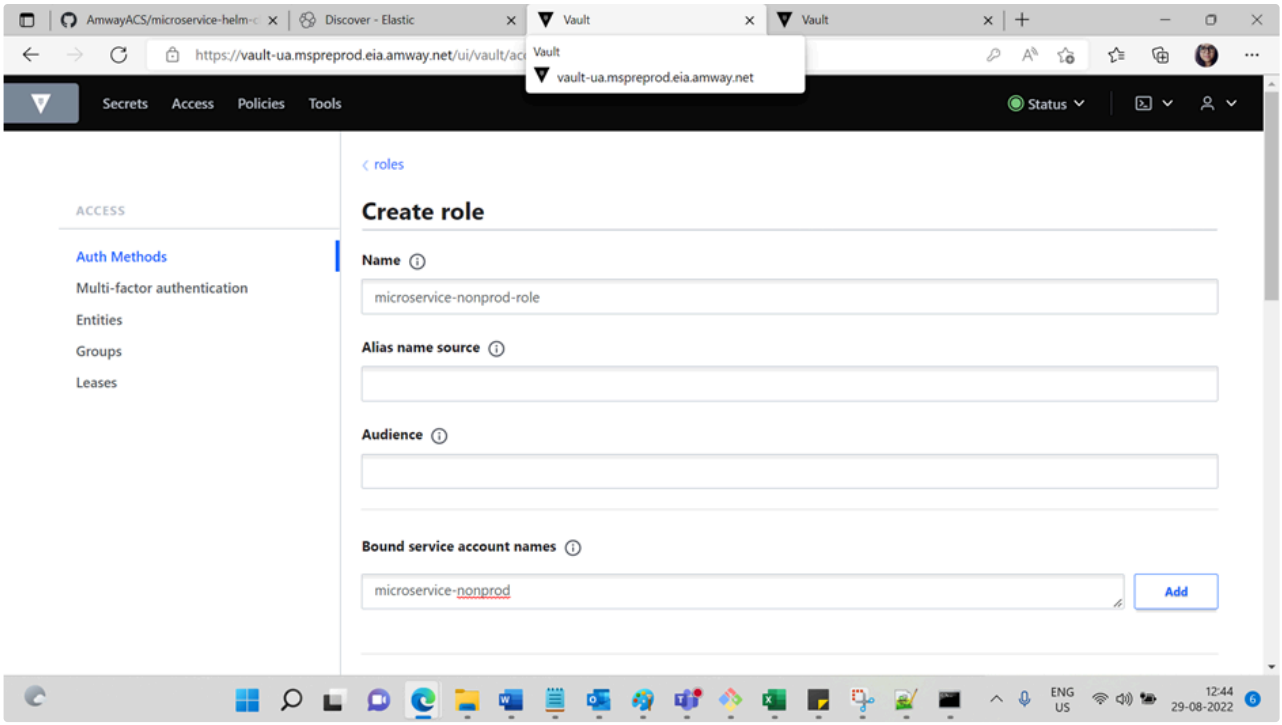
Policy content during creation.

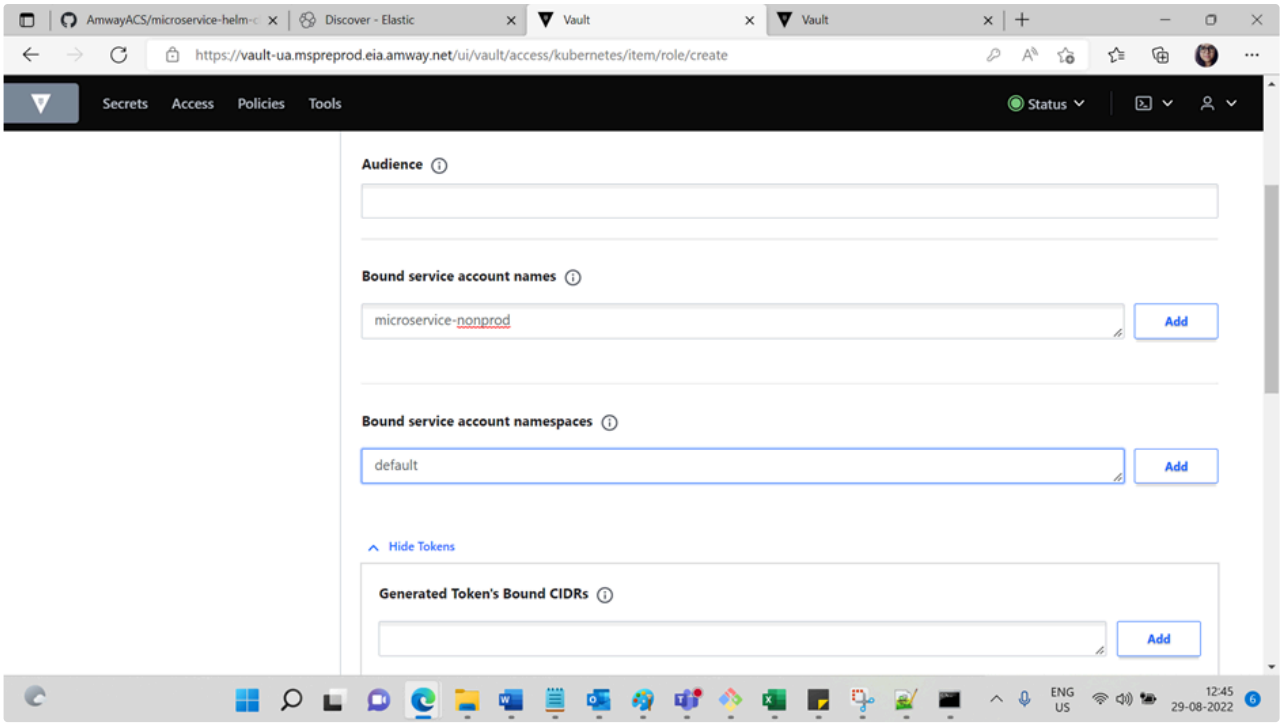
2. Create service account in Kubernetes
   Name your service account: microservice-nonprod

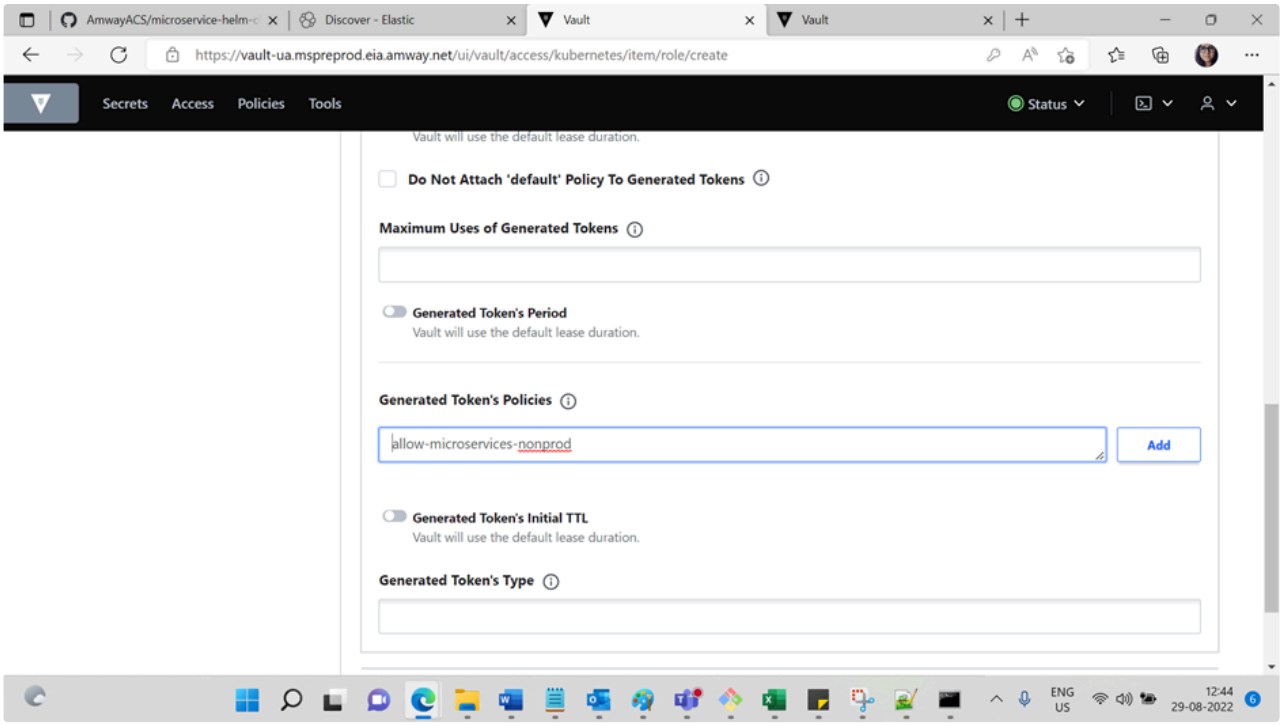
```
$ ./kubectl.exe create sa microservice-nonprod
serviceaccount/microservice-nonprod created
```

   This service account will be used by the microservices to retrieve secrets

3. Setting up the Role in Vault
   Requirements: Service Account , Policy and Namespace


Vault
vault-ua.mspreprod.eia.amway.net
Secrets
Access
Policies
Tools
Status
roles
ACCESS
Auth Methods
Multi-factor authentication
Entities
Groups
Leases
Create role
Name
microservice-nonprod-role
Alias name source
Audience
Bound service account names
microservice-nonprod
Add


https://vault-ua.mspreprod.eia.amway.net/ui/vault/access/kubernetes/item/role/create
Secrets
Access
Policies
Tools
Status
Audience
Bound service account names
microservice-nonprod
Add
Bound service account namespaces
default
Add
Hide Tokens
Generated Token's Bound CIDRs
Add



4. Creating the Secret store in Kubernetes. This will reference the Role and ServiceAccount

```
apiVersion: external-secrets.io/v1beta1
kind: SecretStore
metadata:
  name: vault-backend
  namespace: default
spec:
  provider:
    vault:
      server: "https://vault-ua.mspreprod.eia.amway.net/"
      path: "secret"
      version: "v2"
      auth:
        # Authenticate against Vault using a Kubernetes ServiceAccount
        # token stored in a Secret.
        # https://www.vaultproject.io/docs/auth/kubernetes
        kubernetes:
          # Path where the Kubernetes authentication backend is mounted in Vault
          mountPath: "kubernetes"
          # A required field containing the Vault Role to assume.
          role: "microservice-nonprod-role"
          # Optional service account field containing the name
          # of a kubernetes ServiceAccount
          serviceAccountRef:
            name: "microservice-nonprod"
```

Making changes in Helmchart

1. Including service account “microservice-nonprod” in deployment template

```
    app.kubernetes.io/component: {{ .Values.app.name }}
spec:
  #      imagePullSecrets:
  #        {{- if .Values.app.isProd }}
    - name: {{ .Values.imageCredentials.prod.name }}
     {{- else }}
     - name: {{ .Values.imageCredentials.dev.name }}
     {{- end}}
  dnsConfig:
    #        nameservers:
    #          - 8.8.8.8
    options:
      - name: ndots
        value: "1"
  {{- if .Values.isProd }}
  serviceAccountName: microservice-nonprod
  initContainers:
```

2. Creating an external secret object

```
apiVersion: external-secrets.io/v1beta1
kind: ExternalSecret
metadata:
   name: {{ .Release.Name }}-auth-external
spec:
   refreshInterval: "30s"
   secretStoreRef:
     name: vault-backend
     kind: SecretStore
   target:
     name: {{ .Release.Name }}-auth # K8s secret name
     creationPolicy: Owner # creates secret if not present
   data:
   - secretKey: appPostgresPass # The key in kubernetes secret
     remoteRef:
       key: secret/data/microservices-nonprod/vip-rports #path to vault secret
       property: appPostgresPass   # The key in vault secret
   - secretKey: appWmPass
     remoteRef:
       key: secret/data/microservices-nonprod/vip-reports
       property: appWmPass
```

3. Removing sensitive values from Values.yaml

4. The microservice helm files should include an external secrets file instead of a secrets file

```
apiVersion: external-secrets.io/v1beta1
kind: ExternalSecret
metadata:
   name: {{ .Release.Name }}-auth-external
spec:
   refreshInterval: "30s"
   secretStoreRef:
     name: vault-backend
     kind: SecretStore
   target:
     name: {{ .Release.Name }}-auth # K8s secret name
     creationPolicy: Owner # creates secret if not present
   data:
   - secretKey: appPostgresPass # The key in kubernetes secret
     remoteRef:
       key: secret/data/microservices-nonprod/vip-rports #path to vault secret
       property: appPostgresPass   # The key in vault secret
   - secretKey: appWmPass
     remoteRef:
       key: secret/data/microservices-nonprod/vip-reports
       property: appWmPass
```

Deployment prerequisities

1. Secrets must be created in correct path in Vault

```
Name:         vault-backend
Namespace:    default
Labels:       <none>
Annotations:  <none>
API Version:  external-secrets.io/v1beta1
Kind:         SecretStore
Metadata:
  Creation Timestamp:  2022-08-26T13:08:51Z
  Generation:          5
  Managed Fields:
    API Version:  external-secrets.io/v1beta1
    Fields Type:  FieldsV1
    fieldsV1:
      f:status:
        .:
        f:conditions:
    Manager:      external-secrets
    Operation:    Update
    Subresource:  status
    Time:         2022-08-26T13:08:51Z
    API Version:  external-secrets.io/v1beta1
    Fields Type:  FieldsV1
    fieldsV1:
      f:metadata:
        f:annotations:
          .:
          f:kubectl.kubernetes.io/last-applied-configuration:
      f:spec:
        .:
        f:provider:
          .:
          f:vault:
            .:
            f:auth:
              .:
              f:kubernetes:
                .:
                f:mountPath:
                f:role:
                f:serviceAccountRef:
                  .:
                  f:name:
            f:path:
```

```
                f:role:
                f:serviceAccountRef:
                  .:
                  f:name:
            f:path:
            f:server:
            f:version:
    Manager:         kubectl-client-side-apply
    Operation:       Update
    Time:            2022-08-26T13:08:51Z
  Resource Version:  26936668
  UID:               accb5ac2-7331-4a42-9ca9-568160e798e3
Spec:
  Provider:
    Vault:
      Auth:
        Kubernetes:
          Mount Path:  kubernetes
          Role:        microservice-nonprod-role
          Service Account Ref:
            Name:  microservice-nonprod
      Path:        secret
      Server:      https://vault-ua.mspreprod.eia.amway.net/
      Version:     v2
Status:
  Conditions:
    Last Transition Time:  2022-08-26T13:13:41Z
    Message:               store validated
    Reason:                Valid
    Status:                True
    Type:                  Ready
Events:
  Type    Reason  Age                    From          Message
  ----    ------  ----                   ----          -------
  Normal  Valid   24s (x802 over 2d18h)  secret-store  store validated
```

2. Service account must be created in kubernetes
3. Kubernetes Role must be created with correct Role in Vault
4. Secret store must be created in Kubernetes

Troubleshooting

The prod-EKS cluster has private master endpoint. Periodically Vault may initiate a call to Master Endpoint to validate the service account token credentials.

The call will fail by default due to absence of network path between master endpoint and Vault.

The secret-store will show as Not Ready with “InvalidProviderConfig” status.

To resolve the error modify the EKS master security group to allow inbound traffic from Vault nodes.

Wait for some time, the store will get validated and external secrets will get autosynced.