generic-k8s-manifests

Generic k8s manifests to deploy Privaclave Kubernetes resources

Kubernetes requirements

- The Kubernetes cluster must exist. That will be the cluster where all the K8s components will be deployed to
- It must have an ingress controller to be able to deploy and use the Privaclave's ingresses
- An appropriate Storage Class should be configured in the cluster and called "privaclave-storage-class". It will be used to create the PVCs for the persisten volumes that will be mounted to some of the Privaclave's pods.
- If you want to install Privaclave on an already existing namespace:
  - Set it in the environment variable in .env file in line 2 (namespace variable)
  - Skip the first command of "Deploy general K8s resources" section in this file

Docker images

- Privaclave will provide access to the repositories to download the docker images used by the Privaclave's pods
- Use the token provided by Privaclave to do docker login and download the images
- Upload the docker images to your own docker registries
- Make sure your Kubernetes cluster has read access to the registries where you decided to push the images

Commands to install Privaclave on the existing K8s cluster:

- Export the environment variables that will be used as parameters in the manifests:

```
source .env
```

- Deploy general K8s resources:

```
envsubst < 1-cluster-configs/namespace.yaml | kubectl apply -f -
envsubst < 1-cluster-configs/jwt-secrets.yaml | kubectl apply -f -
envsubst < 1-cluster-configs/general-configmap.yaml | kubectl apply -f -
envsubst < 1-cluster-configs/ingress-external.yaml | kubectl apply -f -
envsubst < 1-cluster-configs/ingress-internal.yaml | kubectl apply -f -
envsubst < 1-cluster-configs/pvcs.yaml | kubectl apply -f -
```

- Vault installation

```
helm repo add hashicorp https://helm.releases.hashicorp.com
helm repo update
helm install vault hashicorp/vault \
  --namespace $namespace \
  --values <(envsubst < 2-vault/vault-values.yaml)
```

- Unseal Vault and create Vault secrets

```
sh 2-vault/vault-keys-scripts.sh
```

- MariaDB installation

```
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo update
helm install mariadb oci://registry-1.docker.io/bitnamicharts/mariadb \
  --namespace $namespace \
  --values <(envsubst < 3-mariadb/mariadb-values.yaml)
```

- Setup MariaDB schema

```
sh 3-mariadb/mariadb-schema-setup-script.sh
```

- Classifier

```
envsubst < 4-classifier/secrets-role.yaml | kubectl apply -f -
envsubst < 4-classifier/serviceaccounts.yaml | kubectl apply -f -
```

```
envsubst < 4-classifier/secrets-rolebinding.yaml | kubectl apply -f -
envsubst < 4-classifier/services.yaml | kubectl apply -f -
envsubst < 4-classifier/deployment-master.yaml | kubectl apply -f -
envsubst < 4-classifier/deployment-worker.yaml | kubectl apply -f -
envsubst < 4-classifier/hpa.yaml | kubectl apply -f -
```

- Cockpit - Autopilot - Policy Handler - Openresty

```
envsubst < 5-cockpit-autopilot-policyhandler-openresty/secrets-role.yaml | kubectl apply -f -
envsubst < 5-cockpit-autopilot-policyhandler-openresty/serviceaccounts.yaml | kubectl apply -f -
envsubst < 5-cockpit-autopilot-policyhandler-openresty/secrets-rolebinding.yaml | kubectl apply -f -
envsubst < 5-cockpit-autopilot-policyhandler-openresty/services.yaml | kubectl apply -f -
envsubst < 5-cockpit-autopilot-policyhandler-openresty/deployment-cockpit.yaml | kubectl apply -f -
envsubst < 5-cockpit-autopilot-policyhandler-openresty/deployment-autopilot.yaml | kubectl apply -f -
envsubst < 5-cockpit-autopilot-policyhandler-openresty/deployment-policyhandler.yaml | kubectl apply -f -
envsubst < 5-cockpit-autopilot-policyhandler-openresty/deployment-openresty.yaml | kubectl apply -f -
```

After installation

- Download the "Gliner_multi_pii_dev_model.zip" file with the presigned-url that Privavaleve will provide
  - Unzip it and upload all its conent to a "Gliner_multi_pii_dev_model" folder into the "classifier-worker-data" PVC that was installed with the last step of "Deploy general K8s resources" section of this file.
- Create an ingress in the Kubernetes cluster pointing to cockpit service, port 8080