Kubernetes Deployments

When deploying Aspire in a Kubernetes cluster.

> If you are using a non local kubernetes cluster, the images should be deployed to a registry accessible to it.

Prerequisites

The current guide assumes you have a working Kubernetes cluster, and access to it via **kubectl** and a **ba sh** terminal.

If using AWS Elasticsearch service, skip the Elasticsearch/Kibana section and modify the Aspire ConfigMap section according to Elasticsearch NoSQL Provider Properties.

Step-by-step guide

Deploy Elasticsearch and Kibana.

> Skip these steps if you already have an Elasticsearch cluster for Aspire to use and go directly to Deploying Aspire 5.

The guide below is based on ECK quickstart. If you need a production deployment or a more detailed process description, please refer to:

- Run Elasticsearch on ECK | Elastic Cloud on Kubernetes [master] | Elastic
- Node configuration | Elastic Cloud on Kubernetes [master] | Elastic

1. Install custom resource definitions

```
kubectl create -f https://download.elastic.co/downloads/eck/1.7.0/crds.yaml
kubectl apply -f https://download.elastic.co/downloads/eck/1.7.0/operator.yaml
```

2. Deploy Elasticsearch cluster (single node)
    a. Create a file called **elasticsearch.yaml**

elasticsearch.yaml

```
apiVersion: elasticsearch.k8s.elastic.co/v1
kind: Elasticsearch
metadata:
  name: quickstart
spec:
  version: 7.9.2
  nodeSets:
  - name: default
    count: 1
    config:
      node.store.allow_mmap: false
```

    b. Deploy the Elasticsearch cluster

```
kubectl apply -f elasticsearch.yaml
```

3. Obtain Basic Authentication password
    a. The password will be stored in the environment variable called "PASSWORD"

```
PASSWORD=$(kubectl get secret quickstart-es-elastic-user -o go-template='{{.data.elastic |
base64decode}}')
```

4. Deploy Kibana cluster
    a. Create a file called **kibana.yaml**

kibana.yaml

```
apiVersion: kibana.k8s.elastic.co/v1
kind: Kibana
metadata:
  name: quickstart
spec:
  version: 7.9.2
  count: 1
  elasticsearchRef:
    name: quickstart
```

b. Deploy Kibana

```
kubectl apply -f kibana.yaml
```

5. Expose Kibana's port locally

```
kubectl port-forward service/quickstart-kb-http 5601
```

6. Browse to Kibana at https://localhost:5601/ (HTTPS warnings will appear on the browser due to the self-signed certificates Elasticsearch and Kibana generates)
    a. Log in using the username "elastic" and the password obtained at step #3.

Deploy Aspire 5

1. (Optional) Upload Kibana Dashboards

    a. Download export.ndjson
    b. Kibana's port should be forwarded into **localhost:5601** as of Step #6 on the Elasticsearch deployment instructions
    c. The environment variable $PASSWORD should hold the *elastic's* user password as of Step #3 on the Elasticsearch deployment instructions.

```
curl -u "elastic:$PASSWORD" -k -F 'file=@/path/to/export.ndjson' \
 -H 'kbn-xsrf:reporting' \
 "https://localhost:5601/api/saved_objects/_import?overwrite=true"
```

2. Create Kubernetes secret for connecting to the SCA docker registry
    a. Replace <EMAIL> and <PASSWORD> with your registered email and password

```
kubectl create secret docker-registry regcred \
--docker-server=docker.repository.sca.accenture.com \
--docker-username=<EMAIL> \
--docker-password=<PASSWORD>\
--docker-email=<EMAIL>
```

3. Create Aspire ConfigMap
    a. Holds common configuration options for your Aspire 5 deployment.
    b. Create a file called **aspire-config.yaml**

aspire-config.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: aspire-config
data:
  aspire_noSql_elastic_server: https://quickstart-es-http:9200
  aspire_noSql_elastic_user: elastic
  aspire_noSql_elastic_authentication_basic: "true"
  com_accenture_aspire_ssl_trustAll: "true"
```


The **com_accenture_aspire_ssl_trustAll** is not recommended to be set as **true** in production environments, the recommended way is to import the untrusted certificate into a Java Key Store (see Configuring a Certificate Store) and then importing using the **com_accenture_aspire_ssl_truststore_file** property (see SSL Certificates Properties).

c. Deploy ConfigMap

```
kubectl apply -f aspire-config.yaml
```

4. Upload License and Settings to Elasticsearch

a. Create a secret containing your **settings.json** and **AspireLicense.lic** files

```
kubectl create secret generic aspire-license-settings \
--from-file=/path/to/AspireLicense.lic \
--from-file=/path/to/config/settings.json
```

b. Create a file called **aspire-upload-job.yaml**

When using a non local kubernetes cluster, make sure to change the **image** url to where the images are hosted.

aspire-upload-job.yaml

```
apiVersion: batch/v1
kind: Job
metadata:
  name: aspire-upload
spec:
  template:
    spec:
      containers:
       - name: aspire-upload-reg-pod
         image: docker.repository.sca.accenture.com/docker/aspire:5.2.2
         command: [ "/bin/bash", "-c", "./opt/aspire/upload-license-settings.sh" ]
         env:
          - name: ASPIRE_LICENSE_PATH
            value: /tmp/AspireLicense.lic
          - name: ASPIRE_SETTINGS_PATH
            value: /tmp/settings.json
          - name: aspire_noSql_elastic_password
            valueFrom:
              secretKeyRef:
                name: quickstart-es-elastic-user
                key: elastic
         envFrom:
           - configMapRef:
               name: aspire-config
         volumeMounts:
           - name: license-settings-secret
             mountPath: /tmp
      volumes:
        - name: license-settings-secret
          secret:
            secretName: aspire-license-settings

      restartPolicy: Never
      imagePullSecrets:
        - name: regcred
  backoffLimit: 4
```

c. Run job

```
kubectl apply -f aspire-upload-job.yaml
```

d. Wait until it has uploaded the files

```
kubectl get pods | grep aspire-upload | awk '{print $1}' | xargs kubectl logs -f
```

6. Create Kubernetes secret for the Aspire cluster wide encryption key
   a. Create a random 32 bytes file which will be your key

```
head -c 32 /dev/urandom > encryption.key
```

   b. Create the secret using the **encryption.key** file

```
kubectl create secret generic aspire-encryption-key --from-file=encryption.key
```

7. (Optional) Generate TLS Certificates for the Admin UI and REST Endpoints
   a. Obtain certificates or generate self-signed certificates. Follow steps at Configuring Certificates for steps on doing this. Also make sure to generate a Java Keystore with the certificates.
      i. Create 2 certificates, one for each FQDN of each subdomain:
         - *.aspire-managers.default.svc.cluster.local import it into **managers.jks**
           - Certificate to use for all manager nodes
         - *.aspire-workers.default.svc.cluster import it into **workers.jks**

        - Certificate to use for all worker nodes
    ii. If you have custom certificates, just import them into a java keystore. If the certificates are trusted, the CA's certificates are not needed.

b. Create Kubernetes ConfigMap holding the CA and Java Keystore with the certificate for the Aspire servers

```
kubectl create secret generic aspire-encryption-key --from-file=encryption.key --from-file
myKeystore.jks
```

8. Deploy Managers
    a. Create a file called **aspire-managers.yaml**

aspire-managers.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: aspire-managers
  labels:
    app: aspire-managers
spec:
  ports:
  - port: 50505
    name: aspire-manager
  clusterIP: None
  selector:
    app: aspire-managers
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: aspire-manager
spec:
  selector:
    matchLabels:
      app: aspire-managers # has to match .spec.template.metadata.labels
  serviceName: "aspire-managers"
  replicas: 1 # by default is 1
  template:
    metadata:
      labels:
        app: aspire-managers # has to match .spec.selector.matchLabels
    spec:
      terminationGracePeriodSeconds: 10
      containers:
        - name: aspire-managers
          image: docker.repository.sca.accenture.com/docker/aspire:5.2.2
          resources:
            requests:
              memory: "4Gi"
              cpu: 2
            limits:
              memory: "4Gi"
              cpu: 2
          ports:
            - containerPort: 50505
              name: aspire-manager
          env:
            - name: ASPIRE_MANAGER_NODE
              value: 'true'
            - name: aspire_max_heap_memory
              value: '4g'
            - name: aspire_noSql_elastic_password
              valueFrom:
                secretKeyRef:
                  name: quickstart-es-elastic-user
                  key: elastic
            - name: aspire_encryption_key_file
              value: '/opt/aspire/encryption/encryption.key'
# Optional if HTTPS is required for the Aspire UI and REST endpoints
```

```
#            - name: ASPIRE_SSL_KEYSTORE_PASS
#              value: '123456'
#            - name: ASPIRE_SSL_KEYSTORE
#              value: '/opt/aspire/tls/myKeystore.jks'
#            - name: ASPIRE_SSL_CA
#              value: '/opt/aspire/tls/ca.crt'
#            - name: aspire_security_https_only
#              value: 'true'
          envFrom:
            - configMapRef:
                name: aspire-config
          volumeMounts:
            - name: encryption-key
              mountPath: /opt/aspire/encryption
# Optional if HTTPS is required for the Aspire UI and REST endpoints
#            - name: tls-certs
#              mountPath: /opt/aspire/tls
#              readOnly: true

          command: ["/bin/bash"]
          args:
            - -c
            - >-
                export com_accenture_aspire_server_hostname=$(hostname -f) &&
                ./opt/aspire/entrypoint.sh
      volumes:
        - name: encryption-key
          secret:
            secretName: aspire-encryption-key
# Optional if HTTPS is required for the Aspire UI and REST endpoints
#        - name: tls-certs
#          secret:
#            secretName: aspire-certs
      imagePullSecrets:
        - name: regcred
```

b. Deploy managers

```
kubectl apply -f aspire-managers.yaml
```

9. Deploy Workers
   a. Create a file called **aspire-workers.yaml**

aspire-workers.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: aspire-workers
  labels:
    app: aspire-workers
spec:
  ports:
  - port: 50505
    name: aspire-worker
  clusterIP: None
  selector:
    app: aspire-workers
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: aspire-worker
spec:
  selector:
    matchLabels:
```

```
      app: aspire-workers # has to match .spec.template.metadata.labels
  serviceName: "aspire-workers"
  replicas: 2 # by default is 1
  template:
    metadata:
      labels:
        app: aspire-workers # has to match .spec.selector.matchLabels
    spec:
      terminationGracePeriodSeconds: 10
      containers:
        - name: aspire-workers
          image: docker.repository.sca.accenture.com/docker/aspire:5.2.2
          resources:
            requests:
              memory: "8Gi"
              cpu: 2
            limits:
              memory: "16Gi"
              cpu: 4
          ports:
            - containerPort: 50505
              name: aspire-worker
          env:
            - name: ASPIRE_WORKER_NODE
              value: 'true'
            - name: aspire_max_heap_memory
              value: '16g'
            - name: aspire_noSql_elastic_password
              valueFrom:
                secretKeyRef:
                  name: quickstart-es-elastic-user
                  key: elastic
            - name: aspire_encryption_key_file
              value: '/opt/aspire/encryption/encryption.key'
# Optional if HTTPS is required for the Aspire UI and REST endpoints
#             - name: ASPIRE_SSL_KEYSTORE_PASS
#               value: '123456'
#             - name: ASPIRE_SSL_KEYSTORE
#               value: '/opt/aspire/tls/myKeystore.jks'
#             - name: ASPIRE_SSL_CA
#               value: '/opt/aspire/tls/ca.crt'
#             - name: aspire_security_https_only
#               value: 'true'
          envFrom:
            - configMapRef:
                name: aspire-config
          volumeMounts:
            - name: encryption-key
              mountPath: /opt/aspire/encryption
# Optional if HTTPS is required for the Aspire UI and REST endpoints
#             - name: tls-certs
#               mountPath: /opt/aspire/tls
#               readOnly: true

          command: ["/bin/bash"]
          args:
            - -c
            - >-
                export com_accenture_aspire_server_hostname=$(hostname -f) &&
                ./opt/aspire/entrypoint.sh
      volumes:
        - name: encryption-key
          secret:
            secretName: aspire-encryption-key
# Optional if HTTPS is required for the Aspire UI and REST endpoints
#         - name: tls-certs
#           secret:
#             secretName: aspire-certs
      imagePullSecrets:
        - name: regcred
```

b. Deploy workers

```
kubectl apply -f aspire-workers.yaml
```

10. Expose Manager port

```
kubectl port-forward pod/aspire-manager-0 50505
```

11. Browse to the Aspire Admin UI at http://localhost:50505