zenika
<animés par la passion>

EKS Configuration & Best Practices

Version 1.18-SNAPSHOT

2019-01-25

Table of Contents

Table 1. History

Date	Author	Detail
2019-01-25	bcouetil	aws eks best practices

1. Introduction

We configure the cluster from scratch with following steps :

- Provision an Amazon EKS cluster
- Deploy worker nodes (Cloud Formation)
- Connect to EKS (kubectl)
- Run Kubernetes apps on EKS cluster

2. Prerequisite

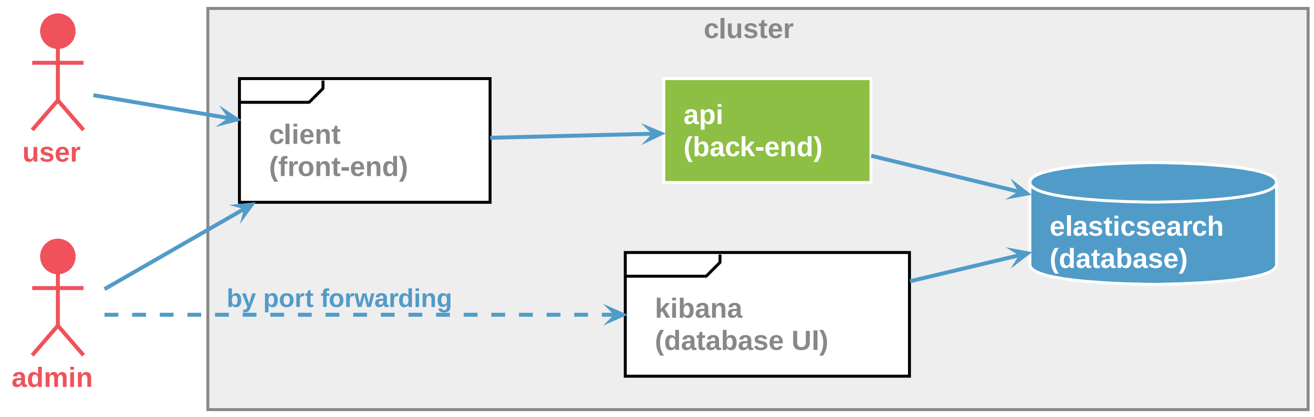
Have an admin account on AWS.

You can switch menu language at the bottom left of any page. Select english.

Follow `Getting Started with Amazon EKS` *of Official documentation to :*

- Create your Amazon EKS Service Role
- Create your Amazon EKS Cluster VPC
- Install and Configure kubectl for Amazon EKS
- Download and Install the Latest AWS CLI
- Create Your Amazon EKS Cluster
- Configure kubectl for Amazon EKS
- Launch and Configure Amazon EKS Worker Nodes

3. Target architecture



3.1. environments

The AWS cluster will host multiple environments, so we first create and use a `develop` namespace :

```
$ kubectl create namespace develop
$ kubectl config current-context
arn:aws:eks:us-east-2:***:cluster/adx-cluster
$ kubectl config set-context arn:aws:eks:us-east-2:***:cluster/adx-cluster --namespace=develop
```

4. Deployments

Kubernetes deployments and services are stored in the same file for each module.

4.1. Elasticsearch

We start with the elasticsearch database.

Some explanation :

- This is the OSS image, simpler, no need for X-Pack
- Note the system command in `initContainers` section

4.1.1. File

```yaml
#
# database (elasticsearch) service and deployement
#

apiVersion: v1
kind: Service
metadata:
  name: elasticsearch
  labels:
    app: db
    tier: backend
    group: adx
spec:
  ports:
    - port: 9200 # External port
      targetPort: http # Port exposed by the pod/container from the deployment
  selector:
    app: db
    group: adx
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: db-dpl
  labels:
    app: db
    tier: backend
    group: adx
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: db
        tier: backend
        group: adx
    spec:
      initContainers:
      - name: "sysctl"
        image: "busybox"
        imagePullPolicy: "IfNotPresent"
        command: ["sysctl", "-w", "vm.max_map_count=262144"]
        securityContext:
          privileged: true
      containers:
      - name: elasticsearch
        image: docker.elastic.co/elasticsearch/elasticsearch-oss:6.5.4
        imagePullPolicy: "IfNotPresent"
        ports:
        - containerPort: 9200
          name: http
        env:
          - name: ES_JAVA_OPTS
            value: "-Xms512m -Xmx512m"
        resources:
          limits:
            memory: 1024Mi
          requests:
            memory: 512Mi
```

4.1.2. Commands

Launch (or update) the deployment :

```
$ kubectl apply -f adx.db.svc-dpl.yml
service/api created
deployment.extensions/api-dpl created

$ kubectl get rs
NAME                  DESIRED   CURRENT   READY     AGE
db-dpl-5c767f46c7     1         1         1         32m

$ kubectl get pods
NAME                        READY     STATUS             RESTARTS    AGE
db-dpl-5c767f46c7-tkqkv     1/1       Running            0           32m
```

4.2. Kibana

Kibana is included, only for elasticsearch administration in test environements.

Some explanation :

- This is the OSS image, simpler, no need for X-Pack
- This will not be accessible from external network, for security reasons

4.2.1. File

```
#
# kibana (elastic admin) service and deployement
#

apiVersion: v1
kind: Service
metadata:
  name: kibana
  labels:
    app: kibana
    tier: backend
    group: adx
spec:
  # pour protéger, pas de type et pas de port + proxy forward
  # type: NodePort # Make the service externally visible via the node
  ports:
    - port: 5601 # External port
      targetPort: http # Port exposed by the pod/container from the deployment
  selector:
    app: kibana
    group: adx
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: kibana-dpl
  labels:
    app: kibana
    tier: backend
    group: adx
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: kibana
        tier: backend
        group: adx
    spec:
      containers:
      - name: kibana
        image: docker.elastic.co/kibana/kibana-oss:6.5.4
        env:
        - name: ELASTICSEARCH_URL
          value: http://elasticsearch:9200
        resources:
          limits:
            memory: 512Mi
          requests:
            memory: 256Mi
        ports:
        - containerPort: 5601
          name: http
```

4.2.2. Commands

Launch (or update) the deployment :

```
$ kubectl apply -f adx.db.svc-dpl.yml
service/api created
deployment.extensions/api-dpl created

$ kubectl get rs
NAME                   DESIRED   CURRENT   READY     AGE
db-dpl-5c767f46c7      1         1         1         32m
kibana-dpl-8d76c6dd8   1         1         1         26m

$ kubectl get pods
NAME                          READY     STATUS              RESTARTS   AGE
db-dpl-5c767f46c7-tkqkv       1/1       Running             0          32m
kibana-dpl-8d76c6dd8-cmrvz    1/1       Running             0          27m
```

To access the UI, we use port forwarding in a dedicated shell :

```
$ kubectl port-forward svc/kibana 5601:5601
```

4.3. Api / backend

Some explanation :

- The backend is pulled from AWS/ECR registry

4.3.1. Prerequisites

- Get the full image name in EKR
  - Got to AWS Admin UI
  - Choose the zone containing your registry
  - [**Services**] → [**ECR**] → api repository
  - Get the `Image URI`
- get the registry password

```
$ aws ecr get-login
docker login -u AWS -p <PASSWORD> -e none https://***.dkr.ecr.eu-west-3.amazonaws.com
```

- create a secret using it

```
$ kubectl delete secret ecr-registry-secret

$ kubectl create secret docker-registry ecr-registry-secret --docker-username="AWS" --docker-password="<PASSWORD>"
--docker-server="***.dkr.ecr.eu-west-3.amazonaws.com" --docker-email="my.email@my-provider.com"
```

Now we can update the file and deploy it.

4.3.2. File

```yaml
#
# api (back-end) service and deployement
#

apiVersion: v1
kind: Service
metadata:
  name: adx-api
  labels:
    app: api
    tier: backend
    group: adx
spec:
  ports:
    - port: 8080 # External port
      targetPort: http # Port exposed by the pod/container from the deployment
  selector:
    app: api
    group: adx
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: api-dpl
  labels:
    app: api
    tier: backend
    group: adx
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: api
        tier: backend
        group: adx
    spec:
      # initContainers:
      #   - name: "sysctl"
      #     image: "busybox"
      #     imagePullPolicy: "IfNotPresent"
      #     command: ["curl", "-XGET", "http://elasticsearch:9200/_cluster/health?pretty=true"]
      containers:
      - name: api
        image: ***.dkr.ecr.eu-west-3.amazonaws.com/adx/adx-api:develop
        ports:
        - containerPort: 8080
          name: http
        env:
        - name: ELASTICSEARCH_REST_URIS
          value: http://elasticsearch:9200
        imagePullPolicy: Always
      imagePullSecrets:
      - name: ecr-registry-secret
```

4.3.3. Commands

Launch (or update) the deployment :

```
$ kubectl apply -f adx.api.svc-dpl.yml
```

4.4. Client / frontend

4.4.1. Prerequisites

Same as Api module.

4.4.2. File

```yaml
#
# client (front-end) service and deployement
#

apiVersion: v1
kind: Service
metadata:
  name: client
  labels:
    app: client
    tier: frontend
    group: adx
spec:
  type: LoadBalancer # Make the service visible to the world
  ports:
    - port: 10080 # External port
      targetPort: http # Port exposed by the pod/container from the deployment
  selector:
    app: client
    group: adx
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: client-dpl
  labels:
    app: client
    tier: frontend
    group: adx
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: client
        tier: frontend
        group: adx
    spec:
      containers:
      - name: client
        image: ***.dkr.ecr.eu-west-3.amazonaws.com/adx/adx-client:develop
        ports:
        - containerPort: 80
          name: http
        imagePullPolicy: Always
      imagePullSecrets:
      - name: ecr-registry-secret
```

4.4.3. Commands

Launch (or update) the deployment :

```
$ kubectl apply -f adx.client.svc-dpl.yml
```

4.4.4. Access the frontend in a browser

- Get the host/port

```
$ get services -o wide
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)           AGE       SELECTOR
adx-api         ClusterIP      10.100.78.159    <none>
8080/TCP          2h        app=api,group=adx
client          LoadBalancer   10.100.145.183   <host>.us-east-2.elb.amazonaws.com   10080:30587/TCP   2h        app
=client,group=adx
elasticsearch   ClusterIP      10.100.15.82     <none>
9200/TCP          23h       app=db,group=adx
kibana          ClusterIP      10.100.114.147   <none>
5601/TCP          23h       app=kibana,group=adx
```

- Go to http://<host>.us-east-2.elb.amazonaws.com:10080