Deploying the Getting Started App on Amazon EKS

June 13, 2025

Introduction

This document provides a step-by-step guide to deploy the `getting-started-app` from the GitHub repository (`https://github.com/docker/getting-started-app.git`) on Amazon Elastic Kubernetes Service (EKS). The application will be containerized using Docker, pushed to Amazon Elastic Container Registry (ECR), and deployed on an EKS cluster.

Prerequisites

Before starting, ensure you have the following:

- An active AWS account with administrative permissions.
- AWS CLI installed and configured:

```
aws configure
```

  Enter your Access Key ID, Secret Access Key, region (e.g., `us-east-1`), and output format (e.g., `json`).

- `eksctl` installed:

```
curl --silent --location "https://github.com/weaveworks/
    eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.
    tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl version
```

- `kubectl` installed:

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.
    k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/
    kubectl
kubectl version --client
```

- Docker installed on your local machine.
- Git installed to clone the repository.

Step 1: Clone the Repository

Clone the `getting-started-app` repository to your local machine:

```
git clone https://github.com/docker/getting-started-app.git
cd getting-started-app
```

The repository contains:

- `Dockerfile`: Defines the Docker image.
- `package.json`: Node.js dependencies.
- `src/`: Application source code.
- `spec/`: Test specifications.
- `yarn.lock`: Yarn dependency lock file.

Step 2: Build and Push the Docker Image to Amazon ECR

2.1 Create an ECR Repository

Create a repository in Amazon ECR to store the Docker image:

```
aws ecr create-repository --repository-name getting-started-app
   --region us-east-1
```

Note the `repositoryUri` from the output (e.g., $\texttt{ACCOUNT}_ID.dkr.ecr.us-east-1.amazonaws.com/getting-started-app$).

2.2 Authenticate Docker with ECR

Log in to ECR using the AWS CLI:

```
aws ecr get-login-password --region us-east-1 | docker login --
   username AWS --password-stdin ACCOUNT_ID.dkr.ecr.us-east-1.
   amazonaws.com
```

Replace $\texttt{ACCOUNT}_ID$ with your AWS account ID.

2.3 Build the Docker Image

Build the Docker image locally:

```
docker build -t getting-started-app .
```

2.4 Tag and Push the Image

Tag the image for ECR:

```
docker tag getting-started-app:latest ACCOUNT_ID.dkr.ecr.us-east-1.amazonaws.com/getting-started-app:latest
```

Push the image to ECR:

```
docker push ACCOUNT_ID.dkr.ecr.us-east-1.amazonaws.com/getting-started-app:latest
```

Step 3: Create the EKS Cluster

Create an EKS cluster named my-eks-cluster using eksctl:

```
eksctl create cluster \
  --name my-eks-cluster \
  --region us-east-1 \
  --nodegroup-name my-nodes \
  --node-type t3.medium \
  --nodes 2 \
  --nodes-min 1 \
  --nodes-max 3 \
  --managed
```

This takes 10-15 minutes. Verify the cluster:

```
eksctl get cluster --name my-eks-cluster --region us-east-1
```

Update your kubeconfig:

```
aws eks update-kubeconfig --name my-eks-cluster --region us-east-1
```

Verify nodes:

```
kubectl get nodes
```

Step 4: Deploy the Application to EKS

4.1 Create a Namespace

Create a namespace for the application:

```
kubectl create namespace getting-started
```

4.2 Create a Deployment

Create a file named app-deployment.yaml with the following content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: getting-started-app
  namespace: getting-started
spec:
  replicas: 2
  selector:
    matchLabels:
      app: getting-started-app
  template:
    metadata:
      labels:
        app: getting-started-app
    spec:
      containers:
      - name: getting-started-app
        image: ACCOUNT_ID.dkr.ecr.us-east-1.amazonaws.com/getting
           -started-app:latest
        ports:
        - containerPort: 3000
```

Replace ACCOUNT$_I$$Dwithyour AWS accountID. Apply the deployment :

```
kubectl apply -f app-deployment.yaml
```

4.3 Create a Service

Create a file named app-service.yaml to expose the application:

```
apiVersion: v1
kind: Service
metadata:
  name: getting-started-service
  namespace: getting-started
spec:
  selector:
    app: getting-started-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 3000
  type: LoadBalancer
```

Apply the service:

```
kubectl apply -f app-service.yaml
```

4.4 Access the Application

Get the external URL of the LoadBalancer:

```
kubectl get svc -n getting-started
```

Look for the EXTERNAL-IP or DNS name. Open it in a browser to access the application running on port 80 (mapped to port 3000 internally).

Step 5: Verify the Deployment

List the pods to ensure they are running:

```
kubectl get pods -n getting-started
```

View logs for debugging:

```
kubectl logs <pod-name> -n getting-started
```

Replace <pod-name> with the name of one of the pods.

Step 6: Clean Up

To avoid incurring costs, delete the EKS cluster and ECR repository:

```
eksctl delete cluster --name my-eks-cluster --region us-east-1
aws ecr delete-repository --repository-name getting-started-app
   --region us-east-1 --force
```

Conclusion

You have successfully deployed the getting-started-app on an EKS cluster. This process demonstrates containerizing an application, pushing it to ECR, and deploying it on Kubernetes with a LoadBalancer service.