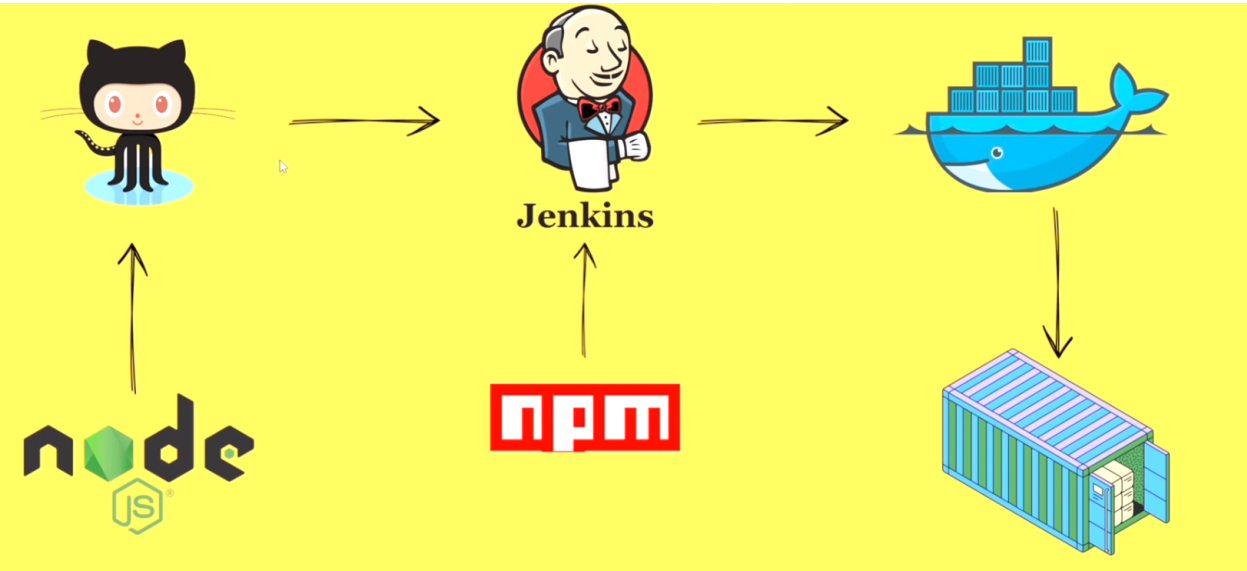
AWS Hands-On: Deploy a Containerized App to EKS using eksctl and kubectl



Pre - requisites

1. **Docker Desktop**
2. **Kubernetes (kubectl)**
3. **AWS CLI** configured with credentials: AWS Access Key, Secret Access Key, region, and output file format.

Steps to Deploy

1. Clone the Repository

Clone the application code from GitHub:

git clone https://github.com/HEMALATHA-V-DEV/DEVOPS.git

2. Verify Kubernetes Configuration

Confirm Kubernetes contexts and cluster information:

```
kubectl config get-contexts        # Check the current context
kubectl cluster-info                # Get control plane endpoint
kubectl get nodes                   #Verify control plane node on Docker Desktop
```

```
kubectl get all                    # List all Kubernetes resources
kubectl get pods                   # List all pods
kubectl logs <pod_name>           #View logs for a specific pod
kubectl describe <resource>      # Describe a specific resource
```

3. Containerize the App - Building a simple Node.js app using the Express framework that listens on port 3000.

Dockerfile: Write a Dockerfile to containerize the app.

dockerfile

```
# Use Node.js image
FROM node:14

# Set working directory
WORKDIR /usr/src/app

# Copy package.json and install dependencies
COPY package.json ./
RUN npm install

# Copy the rest of the application
COPY . .

# Expose the port the app runs on
EXPOSE 3000

# Command to run the app
CMD [ "npm", "start" ]
```

4. Build and Test Docker Image

Build the Docker image:

```
docker image build -t hemalathav20/todo-app:1.0 .
```

Run the container locally to test:

```
docker container run -dp 3002:3000 --name todolist-app hemalathav20/todo-app:1.0
```

http://localhost:3002/

3002 is the port on your **host machine** (your computer).
3000 is the port inside the **container** (where the application listens).

app inside the container listens on port 3000, and by mapping it to host port 3002, you can access the app from http://localhost:3002 on your computer.

5. Push Image to Docker Hub

Push the Docker image to Docker Hub:

```
docker push hemalathav20/todo-app:1.0
```

6. Create EKS Cluster

Install eksctl:

```
brew tap weaveworks/tap

brew install weaveworks/tap/eksctl

eksctl version
```

Create the EKS cluster:

```
eksctl create cluster \

--name demo-cluster \
--version 1.23 \
--region us-east-1 \
--nodegroup-name demo-workers \
--nodes 3 \
--nodes-min 1 \
--nodes-max 3 \
--managed
```

Use AWS CloudFormation in the AWS Console to monitor the cluster creation process.

7. Update kubeconfig

Once the cluster is active, update the Kubernetes configuration:

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

8. Deploy the Application

Apply the deployment configuration:

kubectl apply -f app-server-deployment.yaml

This configuration should include three worker nodes and an exposed service (type: LoadBalancer) for external access.

9. Delete the Cluster

After testing, delete the infrastructure:

eksctl delete cluster --name demo-cluster

Additional Resources

For more details on installing Docker and Kubernetes:

- Docker and Kubernetes Setup
- Kubernetes Setup