**Study Material: Building and Deploying a Node.js Application with Docker and Kubernetes**

This study material will guide you through the process of building a Node.js application, containerizing it with Docker, and deploying it on a Kubernetes cluster. We'll cover key concepts, provide examples, and explain each step in detail.

**1. Introduction to the Project**

**Objective**

* Build a simple Node.js web application using Express.
* Containerize the application using Docker.
* Deploy the application on a Kubernetes cluster.
* Understand different Kubernetes service types: ClusterIP, NodePort, and LoadBalancer.

**2. Key Technologies and Tools**

**Node.js**

* **Definition**: Node.js is a runtime environment that allows you to run JavaScript on the server side.
* **Use Case**: We use Node.js to build a web server using the Express framework.

**Express**

* **Definition**: Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
* **Use Case**: We use Express to create a simple web server that responds with a message containing the server's hostname.

**Docker**

* **Definition**: Docker is a platform that allows you to package applications into containers, which are lightweight, portable, and self-sufficient.
* **Use Case**: We use Docker to containerize our Node.js application, making it easy to deploy and run consistently across different environments.

**Kubernetes**

* **Definition**: Kubernetes is an open-source platform for automating deployment, scaling, and operations of application containers across clusters of hosts.
* **Use Case**: We use Kubernetes to deploy our Dockerized Node.js application and manage its lifecycle.

**3. Step-by-Step Guide**

**Step 1: Setting Up the Node.js Application**

**1.1 Initialize the Node.js Project**

* **Command**: npm init -y
* **Explanation**: This command initializes a new Node.js project and creates a package.json file, which contains metadata about the project and its dependencies.

**1.2 Install Express**

* **Command**: npm install express
* **Explanation**: This command installs the Express framework and adds it as a dependency in the package.json file.

**1.3 Create the Web Server**

* **File**: index.mjs
* **Code**:
* import express from 'express';
* import os from 'os';
* const app = express();
* const port = 3000;
* app.get('/', (req, res) => {
* const hostname = os.hostname();
* res.send(`Hello from ${hostname}`);
* console.log(`Received request from ${req.ip}`);
* });
* app.listen(port, () => {
* console.log(`Web server is listening at port ${port}`);
* });

**Step 2: Containerizing the Application with Docker**

**2.1 Create a Dockerfile**

* **File**: Dockerfile
* **Code**:
* FROM node:alpine
* WORKDIR /app
* EXPOSE 3000
* COPY package.json package-lock.json ./
* RUN npm install
* COPY . .
* CMD ["npm", "start"]

**2.2 Build the Docker Image**

* **Command**: docker build -t <your-dockerhub-username>/k8s-web-hello .

**2.3 Push the Docker Image to Docker Hub**

* **Command**: docker push <your-dockerhub-username>/k8s-web-hello

**Step 3: Deploying the Application on Kubernetes**

**3.1 Create a Kubernetes Deployment**

* **Command**: kubectl create deployment k8s-web-hello --image=<your-dockerhub-username>/k8s-web-hello

**3.2 Expose the Deployment as a Service**

* **Command**: kubectl expose deployment k8s-web-hello --type=ClusterIP --port=3000

**3.3 Access the Application**

* **Command**: kubectl get svc

**3.4 Scale the Deployment**

* **Command**: kubectl scale deployment k8s-web-hello --replicas=4

**3.5 Expose the Deployment as a NodePort Service**

* **Command**: kubectl expose deployment k8s-web-hello --type=NodePort --port=3000

**3.6 Expose the Deployment as a LoadBalancer Service**

* **Command**: kubectl expose deployment k8s-web-hello --type=LoadBalancer --port=3000

**4. Kubernetes Service Types**

| **Service Type** | **Description** | **Use Case** |
| --- | --- | --- |
| ClusterIP | Internal IP within the cluster | Internal communication |
| NodePort | Exposes service on a static port on each node | Development and testing |
| LoadBalancer | Exposes service externally using a cloud load balancer | Production environments |

**5. Example Commands and Outputs**

**Example 1: Creating a Deployment**

kubectl create deployment k8s-web-hello --image=<your-dockerhub-username>/k8s-web-hello

**Output**:

deployment.apps/k8s-web-hello created

**Example 2: Exposing a Deployment as a NodePort Service**

kubectl expose deployment k8s-web-hello --type=NodePort --port=3000

**Output**:

service/k8s-web-hello exposed

**Example 3: Scaling a Deployment**

kubectl scale deployment k8s-web-hello --replicas=4

**Output**:

deployment.apps/k8s-web-hello scaled

**6. Conclusion**

By following this guide, you have learned how to:

* Build a simple Node.js application using Express.
* Containerize the application using Docker.
* Deploy the application on a Kubernetes cluster.
* Understand and use different Kubernetes service types.

**7. Additional Resources**

* [Node.js Documentation](https://nodejs.org/en/docs/)
* [Express Documentation](https://expressjs.com/)
* [Docker Documentation](https://docs.docker.com/)
* [Kubernetes Documentation](https://kubernetes.io/docs/home/)

### Study Material: Building and Deploying a Node.js Application with Docker and Kubernetes

This study material will guide you through the process of building a Node.js application, containerizing it with Docker, and deploying it on a Kubernetes cluster. We'll cover key concepts, provide examples, and explain each step in detail.

---

## \*\*1. Introduction to the Project\*\*

### \*\*Objective\*\*

- Build a simple Node.js web application using Express.

- Containerize the application using Docker.

- Deploy the application on a Kubernetes cluster.

- Understand different Kubernetes service types: ClusterIP, NodePort, and LoadBalancer.

---

## \*\*2. Key Technologies and Tools\*\*

### \*\*Node.js\*\*

- \*\*Definition\*\*: Node.js is a runtime environment that allows you to run JavaScript on the server side.

- \*\*Use Case\*\*: We use Node.js to build a web server using the Express framework.

### \*\*Express\*\*

- \*\*Definition\*\*: Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

- \*\*Use Case\*\*: We use Express to create a simple web server that responds with a message containing the server's hostname.

### \*\*Docker\*\*

- \*\*Definition\*\*: Docker is a platform that allows you to package applications into containers, which are lightweight, portable, and self-sufficient.

- \*\*Use Case\*\*: We use Docker to containerize our Node.js application, making it easy to deploy and run consistently across different environments.

### \*\*Kubernetes\*\*

- \*\*Definition\*\*: Kubernetes is an open-source platform for automating deployment, scaling, and operations of application containers across clusters of hosts.

- \*\*Use Case\*\*: We use Kubernetes to deploy our Dockerized Node.js application and manage its lifecycle.

---

## \*\*3. Step-by-Step Guide\*\*

### \*\*Step 1: Setting Up the Node.js Application\*\*

#### \*\*1.1 Initialize the Node.js Project\*\*

- \*\*Command\*\*: `npm init -y`

- \*\*Explanation\*\*: This command initializes a new Node.js project and creates a `package.json` file, which contains metadata about the project and its dependencies.

#### \*\*1.2 Install Express\*\*

- \*\*Command\*\*: `npm install express`

- \*\*Explanation\*\*: This command installs the Express framework and adds it as a dependency in the `package.json` file.

#### \*\*1.3 Create the Web Server\*\*

- \*\*File\*\*: `index.mjs`

- \*\*Code\*\*:

```javascript

import express from 'express';

import os from 'os';

const app = express();

const port = 3000;

app.get('/', (req, res) => {

const hostname = os.hostname();

res.send(`Hello from ${hostname}`);

console.log(`Received request from ${req.ip}`);

});

app.listen(port, () => {

console.log(`Web server is listening at port ${port}`);

});

```

- \*\*Explanation\*\*: This code creates a simple web server that listens on port 3000. When a request is made to the root URL (`/`), the server responds with a message containing the hostname of the server.

### \*\*Step 2: Containerizing the Application with Docker\*\*

#### \*\*2.1 Create a Dockerfile\*\*

- \*\*File\*\*: `Dockerfile`

- \*\*Code\*\*:

```dockerfile

FROM node:alpine

WORKDIR /app

EXPOSE 3000

COPY package.json package-lock.json ./

RUN npm install

COPY . .

CMD ["npm", "start"]

```

- \*\*Explanation\*\*:

- \*\*FROM node:alpine\*\*: Uses the official Node.js Alpine image as the base image.

- \*\*WORKDIR /app\*\*: Sets the working directory inside the container to `/app`.

- \*\*EXPOSE 3000\*\*: Exposes port 3000 for the web server.

- \*\*COPY package.json package-lock.json ./\*\*: Copies the `package.json` and `package-lock.json` files to the working directory.

- \*\*RUN npm install\*\*: Installs the dependencies listed in `package.json`.

- \*\*COPY . .\*\*: Copies the rest of the application code to the working directory.

- \*\*CMD ["npm", "start"]\*\*: Specifies the command to run when the container starts.

#### \*\*2.2 Build the Docker Image\*\*

- \*\*Command\*\*: `docker build -t <your-dockerhub-username>/k8s-web-hello .`

- \*\*Explanation\*\*: This command builds a Docker image from the Dockerfile and tags it with your Docker Hub username and a repository name (`k8s-web-hello`).

#### \*\*2.3 Push the Docker Image to Docker Hub\*\*

- \*\*Command\*\*: `docker push <your-dockerhub-username>/k8s-web-hello`

- \*\*Explanation\*\*: This command pushes the Docker image to Docker Hub, making it available for deployment.

### \*\*Step 3: Deploying the Application on Kubernetes\*\*

#### \*\*3.1 Create a Kubernetes Deployment\*\*

- \*\*Command\*\*: `kubectl create deployment k8s-web-hello --image=<your-dockerhub-username>/k8s-web-hello`

- \*\*Explanation\*\*: This command creates a Kubernetes deployment named `k8s-web-hello` using the Docker image we pushed to Docker Hub.

#### \*\*3.2 Expose the Deployment as a Service\*\*

- \*\*Command\*\*: `kubectl expose deployment k8s-web-hello --type=ClusterIP --port=3000`

- \*\*Explanation\*\*: This command exposes the deployment as a Kubernetes service with type `ClusterIP`, which makes the application accessible within the cluster.

#### \*\*3.3 Access the Application\*\*

- \*\*Command\*\*: `kubectl get svc`

- \*\*Explanation\*\*: This command lists the services running in the cluster. You can use the `ClusterIP` to access the application from within the cluster.

#### \*\*3.4 Scale the Deployment\*\*

- \*\*Command\*\*: `kubectl scale deployment k8s-web-hello --replicas=4`

- \*\*Explanation\*\*: This command scales the deployment to 4 replicas, meaning 4 instances of the application will be running.

#### \*\*3.5 Expose the Deployment as a NodePort Service\*\*

- \*\*Command\*\*: `kubectl expose deployment k8s-web-hello --type=NodePort --port=3000`

- \*\*Explanation\*\*: This command exposes the deployment as a `NodePort` service, making the application accessible from outside the cluster using the node's IP address and a randomly assigned port.

#### \*\*3.6 Expose the Deployment as a LoadBalancer Service\*\*

- \*\*Command\*\*: `kubectl expose deployment k8s-web-hello --type=LoadBalancer --port=3000`

- \*\*Explanation\*\*: This command exposes the deployment as a `LoadBalancer` service, which is typically used in cloud environments to automatically provision a load balancer.

---

## \*\*4. Kubernetes Service Types\*\*

### \*\*ClusterIP\*\*

- \*\*Definition\*\*: Exposes the service on a cluster-internal IP. The service is only accessible from within the cluster.

- \*\*Use Case\*\*: Internal communication between services within the cluster.

### \*\*NodePort\*\*

- \*\*Definition\*\*: Exposes the service on each node's IP at a static port. The service is accessible from outside the cluster using the node's IP and the assigned port.

- \*\*Use Case\*\*: Development and testing environments where external access is needed.

### \*\*LoadBalancer\*\*

- \*\*Definition\*\*: Exposes the service externally using a cloud provider's load balancer. Automatically assigns an external IP address.

- \*\*Use Case\*\*: Production environments where high availability and load balancing are required.

---

## \*\*5. Example Commands and Outputs\*\*

### \*\*Example 1: Creating a Deployment\*\*

```bash

kubectl create deployment k8s-web-hello --image=<your-dockerhub-username>/k8s-web-hello

```

\*\*Output\*\*:

```

deployment.apps/k8s-web-hello created

```

### \*\*Example 2: Exposing a Deployment as a NodePort Service\*\*

```bash

kubectl expose deployment k8s-web-hello --type=NodePort --port=3000

```

\*\*Output\*\*:

```

service/k8s-web-hello exposed

```

### \*\*Example 3: Scaling a Deployment\*\*

```bash

kubectl scale deployment k8s-web-hello --replicas=4

```

\*\*Output\*\*:

```

deployment.apps/k8s-web-hello scaled

```

---

## \*\*6. Visual Aids\*\*

### \*\*Dockerfile Structure\*\*

```plaintext

FROM node:alpine

WORKDIR /app

EXPOSE 3000

COPY package.json package-lock.json ./

RUN npm install

COPY . .

CMD ["npm", "start"]

```

### \*\*Kubernetes Service Types\*\*

| Service Type | Description | Use Case |

|--------------|-------------|----------|

| ClusterIP | Internal IP within the cluster | Internal communication |

| NodePort | Exposes service on a static port on each node | Development and testing |

| LoadBalancer | Exposes service externally using a cloud load balancer | Production environments |

---

## \*\*7. Conclusion\*\*

By following this guide, you have learned how to:

- Build a simple Node.js application using Express.

- Containerize the application using Docker.

- Deploy the application on a Kubernetes cluster.

- Understand and use different Kubernetes service types.

This knowledge is essential for modern application development and deployment, enabling you to build scalable, portable, and efficient applications.

---

## \*\*8. Additional Resources\*\*

- [Node.js Documentation](https://nodejs.org/en/docs/)

- [Express Documentation](https://expressjs.com/)

- [Docker Documentation](https://docs.docker.com/)

- [Kubernetes Documentation](https://kubernetes.io/docs/home/)

---

This study material provides a comprehensive understanding of the process, with clear explanations, examples, and visual aids to help you grasp the concepts effectively.