# Azure CI/CD Pipeline with GitHub, Jenkins, and Kubernetes

## 1. Upload Your Node.js Application to GitHub

Follow these steps to upload your Node.js application to GitHub:

### Initialize Git and Commit Your Code

cd /path/to/your-project  
git init  
git add .  
git commit -m "Initial commit"

### Create and Link Your GitHub Repository

git remote add origin https://github.com/your-username/your-repo.git  
git branch -M main  
git push -u origin main

## 2. Set Up Jenkins for CI/CD

### Install Jenkins and Required Plugins

Ensure Jenkins is installed and configure the following plugins:  
- Docker Plugin  
- Kubernetes Plugin  
- Pipeline Plugin  
- Git Plugin

### Connect Jenkins to GitHub

1. In GitHub, navigate to Settings > Webhooks.  
2. Click Add webhook.  
3. Enter your Jenkins webhook URL: http://your-jenkins-server/github-webhook/  
4. Choose application/json as the content type.  
5. Enable Push events.  
6. Click Add webhook.

### Create a New Jenkins Pipeline Job

1. Open Jenkins and create a New Item.  
2. Select Pipeline.  
3. Configure the GitHub project by adding your repository URL.  
4. Use a Jenkinsfile for automation.

## 3. Write a Jenkinsfile for Automation

Create a Jenkinsfile in the root of your repository:

pipeline {  
 agent any  
 stages {  
 stage('Clone Repository') {  
 steps {  
 git 'https://github.com/your-username/your-repo.git'  
 }  
 }  
 stage('Build Docker Image') {  
 steps {  
 sh 'docker build -t your-app:latest .'  
 }  
 }  
 stage('Push Docker Image') {  
 steps {  
 withDockerRegistry([credentialsId: 'docker-hub-credentials', url: '']) {  
 sh 'docker tag your-app:latest your-dockerhub-username/your-app:latest'  
 sh 'docker push your-dockerhub-username/your-app:latest'  
 }  
 }  
 }  
 }  
}

## 4. Dockerize Your Application

Create a Dockerfile in the root of your project:

FROM node:16  
WORKDIR /app  
COPY package\*.json ./  
RUN npm install  
COPY . .  
CMD ["node", "server.js"]  
EXPOSE 3000

## 5. Set Up Kubernetes Configuration

Create a `k8s/` directory and add the following YAML files:

### Deployment Configuration (deployment.yml)

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: node-app-deployment  
spec:  
 replicas: 2  
 selector:  
 matchLabels:  
 app: node-app  
 template:  
 metadata:  
 labels:  
 app: node-app  
 spec:  
 containers:  
 - name: node-app  
 image: your-dockerhub-username/your-app:latest  
 ports:  
 - containerPort: 3000

### Service Configuration (service.yml)

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

## 6. Connect to Azure Using the Terminal

If you wish to run Kubernetes commands in Azure, connect to your Azure Kubernetes Service (AKS):

### Login to Azure

az login

### Set the Subscription (If Multiple Subscriptions Exist)

az account set --subscription "your-subscription-id"

### Connect to the AKS Cluster

az aks get-credentials --resource-group your-resource-group --name your-aks-cluster

### Deploy the Application to Kubernetes

kubectl apply -f k8s/

### Verify Deployment

kubectl get pods  
kubectl get services