Here's a set of Kubernetes (k8s) manifests for each component of your full-stack project. These files are written with **separate Deployments and Services**, designed to work in a microservices architecture and ready for kubectl apply -f.

**📁 File Structure:**

k8s/

├── django/

│ ├── deployment.yaml

│ └── service.yaml

├── java/

│ ├── deployment.yaml

│ └── service.yaml

├── node/

│ ├── deployment.yaml

│ └── service.yaml

├── react/

│ ├── deployment.yaml

│ └── service.yaml

├── mongo/

│ ├── deployment.yaml

│ └── service.yaml

**✅ 1. Django**

**k8s/django/deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: django-deployment

spec:

replicas: 1

selector:

matchLabels:

app: django

template:

metadata:

labels:

app: django

spec:

containers:

- name: django

image: your-django-image:latest

ports:

- containerPort: 8000

**k8s/django/service.yaml**

apiVersion: v1

kind: Service

metadata:

name: django-service

spec:

selector:

app: django

ports:

- protocol: TCP

port: 8000

targetPort: 8000

type: ClusterIP

**✅ 2. Java (Spring Boot)**

**k8s/java/deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: java-deployment

spec:

replicas: 1

selector:

matchLabels:

app: java

template:

metadata:

labels:

app: java

spec:

containers:

- name: java

image: your-java-image:latest

ports:

- containerPort: 8080

**k8s/java/service.yaml**

apiVersion: v1

kind: Service

metadata:

name: java-service

spec:

selector:

app: java

ports:

- protocol: TCP

port: 8080

targetPort: 8080

type: ClusterIP

**✅ 3. Node.js**

**k8s/node/deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: node-deployment

spec:

replicas: 1

selector:

matchLabels:

app: node

template:

metadata:

labels:

app: node

spec:

containers:

- name: node

image: your-node-image:latest

ports:

- containerPort: 3000

**k8s/node/service.yaml**

apiVersion: v1

kind: Service

metadata:

name: node-service

spec:

selector:

app: node

ports:

- protocol: TCP

port: 3000

targetPort: 3000

type: ClusterIP

**✅ 4. React.js (Nginx)**

**k8s/react/deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: react-deployment

spec:

replicas: 1

selector:

matchLabels:

app: react

template:

metadata:

labels:

app: react

spec:

containers:

- name: react

image: your-react-image:latest

ports:

- containerPort: 80

**k8s/react/service.yaml**

apiVersion: v1

kind: Service

metadata:

name: react-service

spec:

selector:

app: react

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer

**✅ 5. MongoDB**

**k8s/mongo/deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: mongo-deployment

spec:

replicas: 1

selector:

matchLabels:

app: mongo

template:

metadata:

labels:

app: mongo

spec:

containers:

- name: mongo

image: mongo:6

ports:

- containerPort: 27017

volumeMounts:

- mountPath: /data/db

name: mongo-storage

volumes:

- name: mongo-storage

emptyDir: {}

**k8s/mongo/service.yaml**

apiVersion: v1

kind: Service

metadata:

name: mongo-service

spec:

selector:

app: mongo

ports:

- protocol: TCP

port: 27017

targetPort: 27017

type: ClusterIP

**🌐 1. Ingress Setup**

**k8s/ingress/ingress.yaml** (using NGINX Ingress Controller)

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: app-ingress

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- host: your-domain.com

http:

paths:

- path: /django

pathType: Prefix

backend:

service:

name: django-service

port:

number: 8000

- path: /react

pathType: Prefix

backend:

service:

name: react-service

port:

number: 80

- path: /node

pathType: Prefix

backend:

service:

name: node-service

port:

number: 3000

You must deploy the [NGINX Ingress Controller](https://kubernetes.github.io/ingress-nginx/deploy/) before using this.

**🔧 2. ConfigMap (example for Django & Node)**

**k8s/configs/django-configmap.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

name: django-config

data:

DEBUG: "False"

DJANGO\_ALLOWED\_HOSTS: "your-domain.com"

**k8s/configs/node-configmap.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

name: node-config

data:

NODE\_ENV: "production"

API\_KEY: "your-api-key"

**🔐 3. Secrets (MongoDB, Django Secret Key)**

**k8s/secrets/mongo-secret.yaml**

apiVersion: v1

kind: Secret

metadata:

name: mongo-secret

type: Opaque

data:

MONGO\_INITDB\_ROOT\_USERNAME: bW9uZ291c2Vy # base64: mongouser

MONGO\_INITDB\_ROOT\_PASSWORD: c2VjdXJlcGFzcw== # base64: securepass

**k8s/secrets/django-secret.yaml**

apiVersion: v1

kind: Secret

metadata:

name: django-secret

type: Opaque

data:

SECRET\_KEY: c2VjdXJlX2RqYW5nb19rZXk= # base64 encoded Django secret key

**🧩 4. kustomization.yaml**

Each directory (e.g., k8s/) will have a kustomization.yaml like this:

**k8s/kustomization.yaml**

resources:

- django/deployment.yaml

- django/service.yaml

- java/deployment.yaml

- java/service.yaml

- node/deployment.yaml

- node/service.yaml

- react/deployment.yaml

- react/service.yaml

- mongo/deployment.yaml

- mongo/service.yaml

- ingress/ingress.yaml

- configs/django-configmap.yaml

- configs/node-configmap.yaml

- secrets/mongo-secret.yaml

- secrets/django-secret.yaml

To deploy:

kubectl apply -k k8s/

**📦 5. (Optional) Helm Chart Skeleton**

If you want Helm:

helm-chart/

├── Chart.yaml

├── values.yaml

├── templates/

│ ├── deployment.yaml

│ ├── service.yaml

│ ├── ingress.yaml

│ ├── configmap.yaml

│ ├── secret.yaml

Would you like me to **generate a full Helm chart**, or help you **package and deploy it to Minikube**, AWS EKS, or any other platform?

Let me know your next goal — I can also zip the full structure for your GitHub repo.