K8s Application Configuration

what are the application configuration in k8 give a scenario

In Kubernetes, application configuration involves specifying how your application should run within a cluster. This can include various aspects such as environment variables, command-line arguments, resource limits, and more. Here's a breakdown of how you might configure an application in Kubernetes with a practical example:

Scenario: Configuring a Web Application

Imagine you're deploying a web application that needs to be configured for different environments, such as development, staging, and production. You want to ensure that the application behaves correctly in each environment by setting appropriate configuration parameters.

1. Using ConfigMaps

ConfigMaps are Kubernetes objects designed to hold configuration data. They allow you to decouple configuration artifacts from container images.

- **Define a ConfigMap:**
  Create a ConfigMap YAML file to store your application configuration:

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: web-app-config
data:
  DATABASE_URL: "postgres://db:5432/mydatabase"
  LOG_LEVEL: "info"
```

- **Apply the ConfigMap:**

```bash
kubectl apply -f configmap.yaml
```

- **Reference ConfigMap in a Pod:**
  Update your Pod or Deployment YAML to use the ConfigMap:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-app
  template:
    metadata:
      labels:
        app: web-app
    spec:
      containers:
      - name: web-app
        image: my-web-app:latest
        env:
        - name: DATABASE_URL
          valueFrom:
            configMapKeyRef:
              name: web-app-config
              key: DATABASE_URL
        - name: LOG_LEVEL
          valueFrom:
            configMapKeyRef:
              name: web-app-config
              key: LOG_LEVEL
```

2. Using Secrets

Secrets are used to manage sensitive information like passwords, OAuth tokens, and SSH keys.

- **Define a Secret:**
  Create a Secret YAML file:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: db-credentials
type: Opaque
data:
  username: cG9zdGdyZXM=  # base64 encoded value for "postgres"
  password: cGFzc3dvcmQ=  # base64 encoded value for "password"
```

- **Apply the Secret:**

```bash
kubectl apply -f secret.yaml
```

- **Reference Secret in a Pod:**
  Update your Pod or Deployment YAML:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-app
  template:
    metadata:
      labels:
        app: web-app
    spec:
      containers:
      - name: web-app
        image: my-web-app:latest
        env:
        - name: DB_USERNAME
          valueFrom:
            secretKeyRef:
              name: db-credentials
              key: username
        - name: DB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: db-credentials
              key: password
```

3. Resource Requests and Limits

Specifying resource requests and limits ensures your application has the necessary resources and does not overconsume them.

- **Define resource requests and limits in your Pod/Deployment YAML:**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web-app
  template:
    metadata:
      labels:
        app: web-app
    spec:
      containers:
      - name: web-app
        image: my-web-app:latest
        resources:
          requests:
            memory: "64Mi"
            cpu: "250m"
          limits:
            memory: "128Mi"
            cpu: "500m"
```

4. Environment-Specific Configurations

For different environments, you might want to use different ConfigMaps or Secrets. You can manage this by using environment-specific labels or names for your ConfigMaps and Secrets, and updating your Deployment configurations accordingly.
By leveraging these Kubernetes features, you can effectively manage and configure your applications in a way that is flexible and scalable.