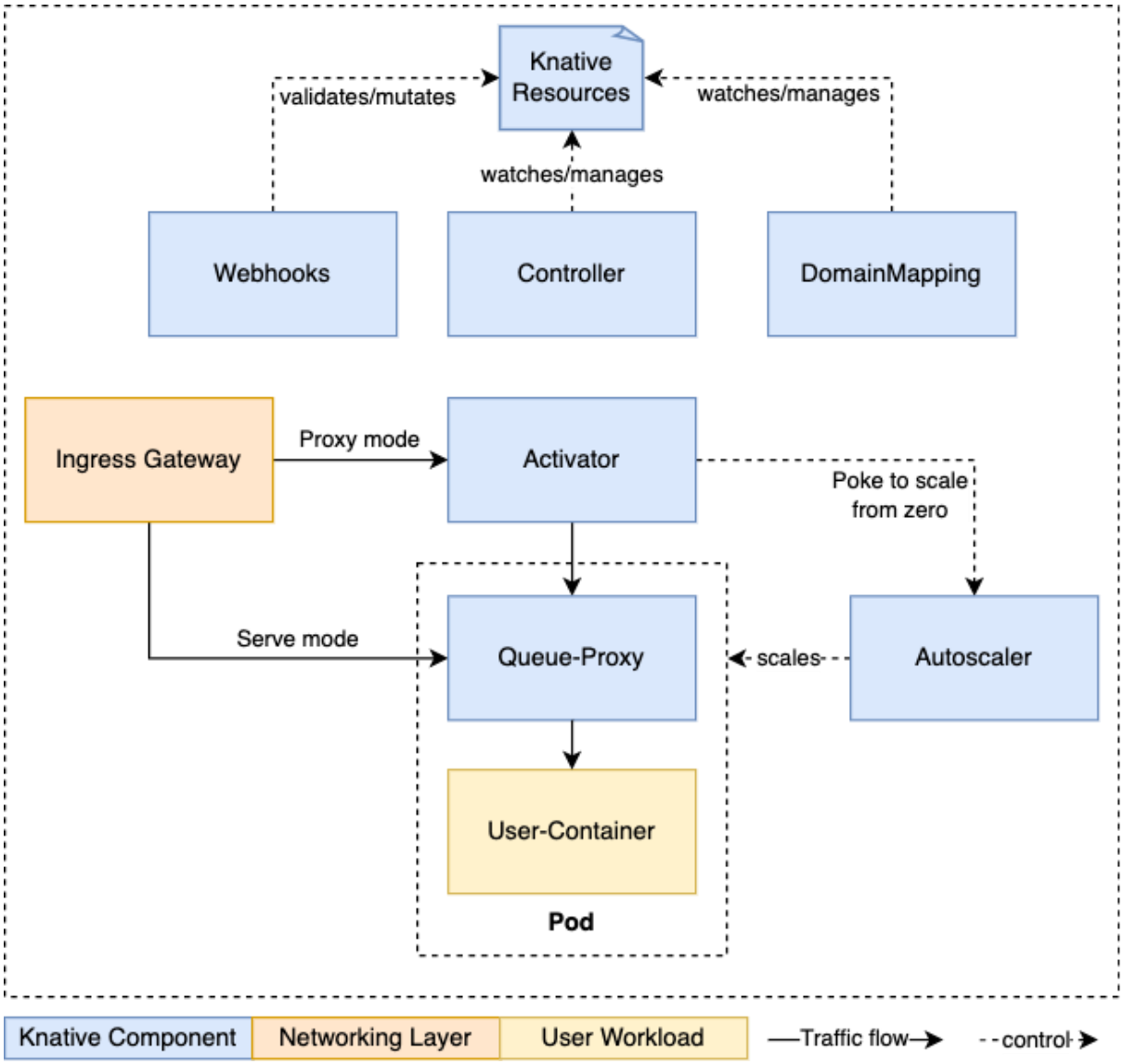
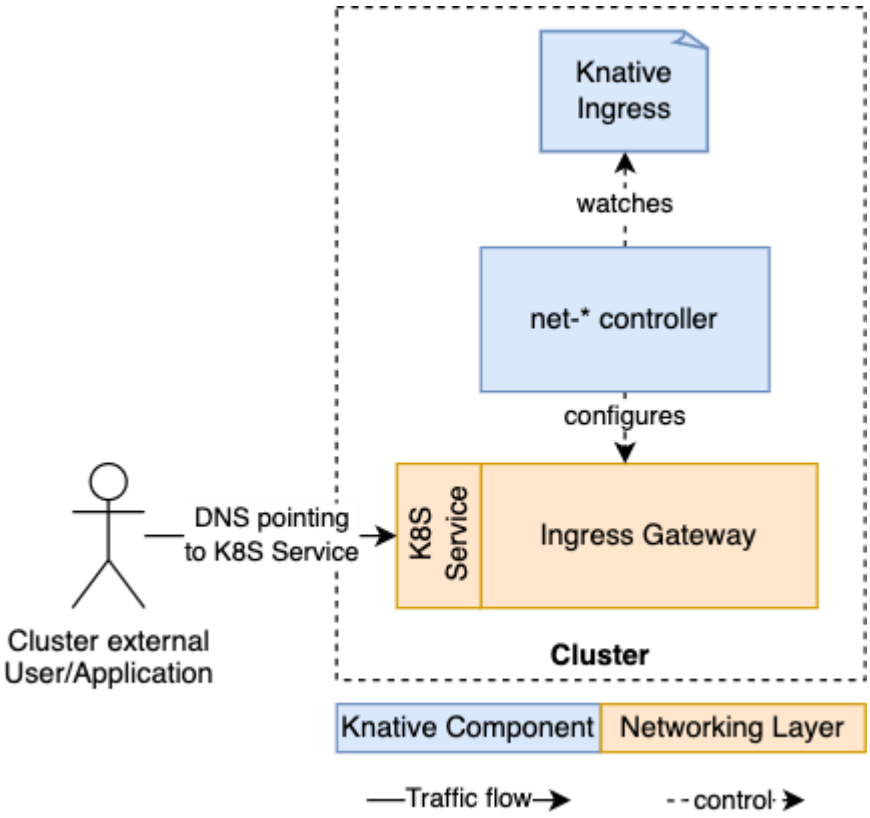
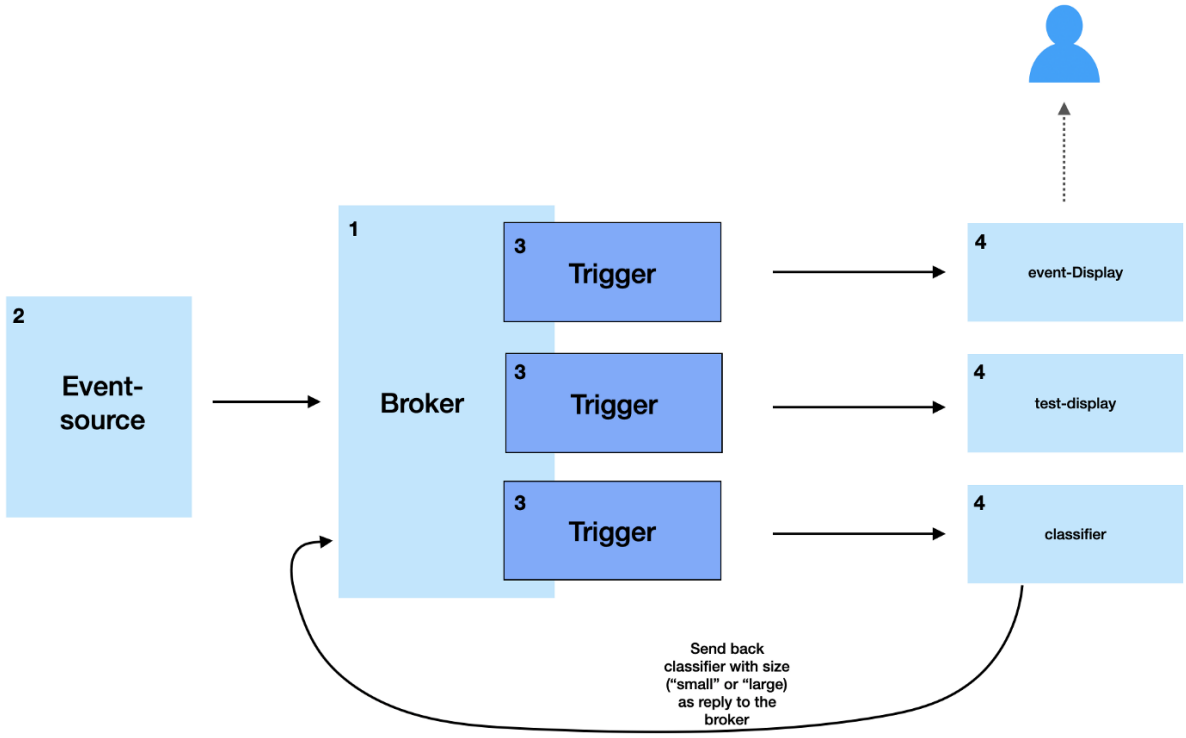
Module 8: Serverless on Kubernetes (Knative) — Detailed Notes

---

1. Knative Serving & Eventing







1.1 What is Knative?

Knative is a Kubernetes-based platform for building and deploying **serverless applications**.

It provides:

- **Automatic scaling to zero**
- **Event-driven workloads**
- **Traffic management (Blue/Green, Canary)**
- **Built-in revisioning**
- **Fast container-based serverless deployments**

Knative has two major components:

1. **Knative Serving**
2. **Knative Eventing**

1.2 Knative Serving

Knative Serving enables:

- Autoscaling (including scale-to-zero)
- Traffic splitting between revisions
- Routing & networking abstraction
- Deploying container-based functions

Serving Objects

API Object	Purpose
Service	High-level definition of serverless app
Route	Traffic distribution
Configuration	Captures desired app state
Revision	Immutable version of the app

Knative Service Example

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: hello-knative
spec:
  template:
    spec:
      containers:
```

```
    - image: gcr.io/knative-samples/helloworld-go
      env:
        - name: TARGET
          value: "Knative User"
```

1.3 Traffic Splitting in Knative

Knative allows **canary deployments** easily:

```
spec:
  traffic:
    - revisionName: v1
      percent: 80
    - revisionName: v2
      percent: 20
```

1.4 Knative Eventing

Knative Eventing enables **event-driven serverless patterns**.

Key concepts:

- **Broker**: Event delivery mesh
- **Trigger**: Filters events to subscribers
- **Source**: Event producers
- **Sink**: Target receiving events (function, service)

Event Workflow

Source → Broker → Trigger → Knative Service

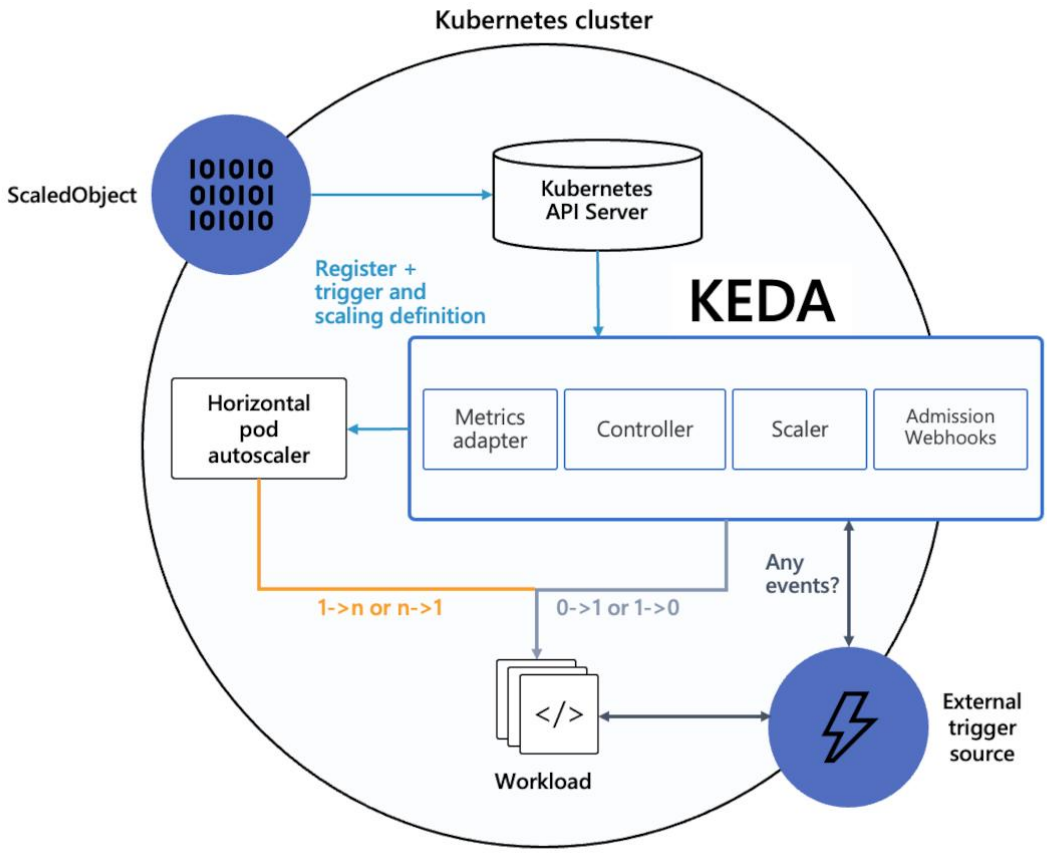
Example Trigger

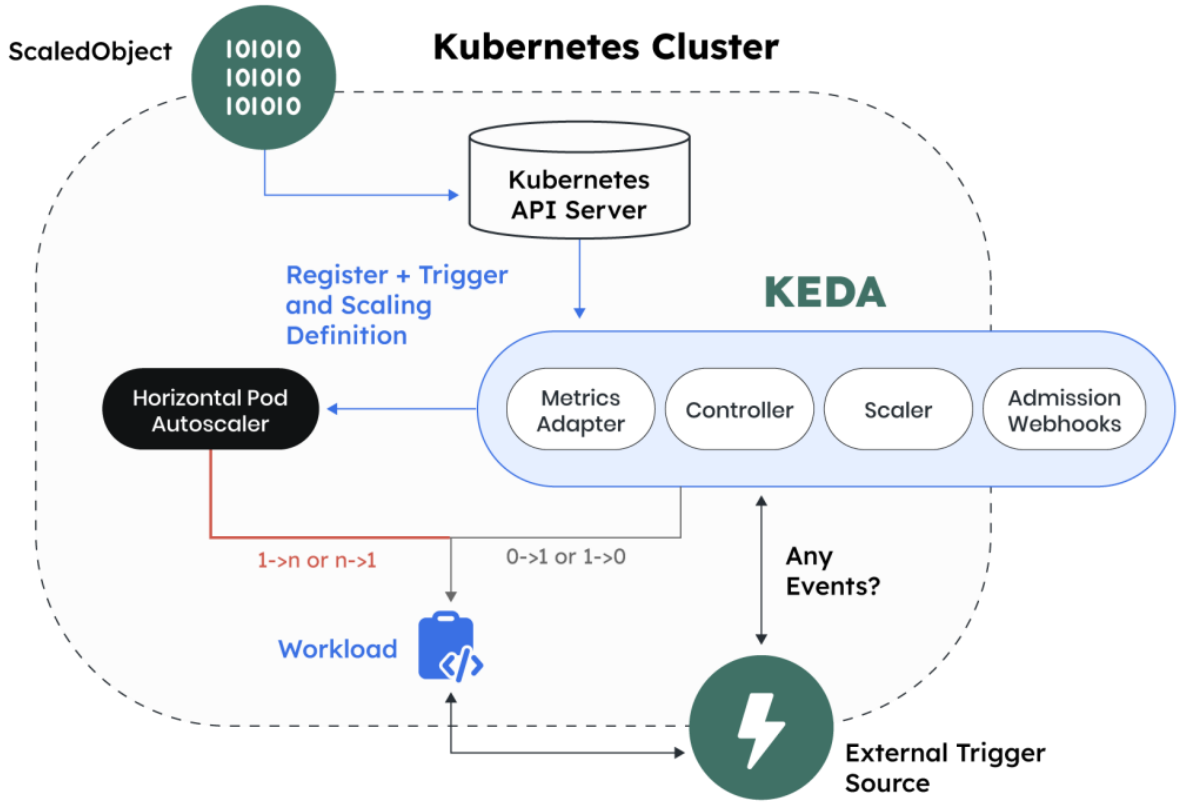
```
apiVersion: eventing.knative.dev/v1
kind: Trigger
metadata:
  name: hello-trigger
spec:
  broker: default
  filter:
```

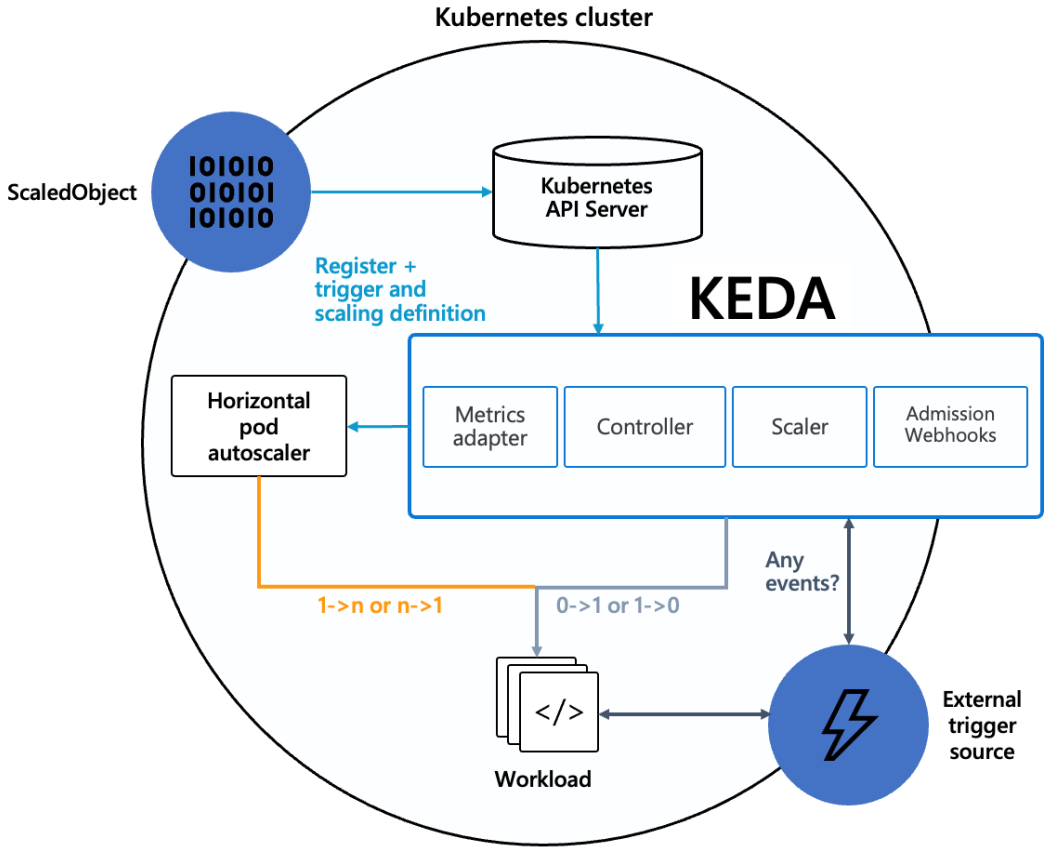
```
  attributes:
    type: dev.knative.samples.helloworld
  subscriber:
    ref:
      apiVersion: serving.knative.dev/v1
      kind: Service
      name: hello-knative
```

---

2. KEDA for Event-Based Autoscaling




ScaledObject
Kubernetes Cluster
Kubernetes API Server
Register + Trigger and Scaling Definition
KEDA
Horizontal Pod Autoscaler
Metrics Adapter
Controller
Scaler
Admission Webhooks
1->n or n->1
0->1 or 1->0
Any Events?
Workload
External Trigger Source


Kubernetes cluster
ScaledObject
Kubernetes API Server
Register + trigger and scaling definition
KEDA
Horizontal pod autoscaler
Metrics adapter
Controller
Scaler
Admission Webhooks
Any events?
1->n or n->1
0->1 or 1->0
Workload
External trigger source

2.1 What is KEDA?

KEDA (Kubernetes Event-Driven Autoscaling) extends Kubernetes to scale workloads based on **external event sources**, such as:

- Kafka
- RabbitMQ
- Azure ServiceBus
- AWS SQS
- Kubernetes Jobs
- Prometheus metrics
- Cron schedules

2.2 KEDA Architecture

Components:

- **KEDA Operator**
- **KEDA Metrics Server**
- **Scalers**
- **TriggerAuthentication**

KEDA combines:

- **K8s HPA** + **event source metrics**

2.3 KEDA ScaledObject Example

```
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: queue-scaler
spec:
  scaleTargetRef:
    name: worker-app
  minReplicaCount: 0
  maxReplicaCount: 20
```

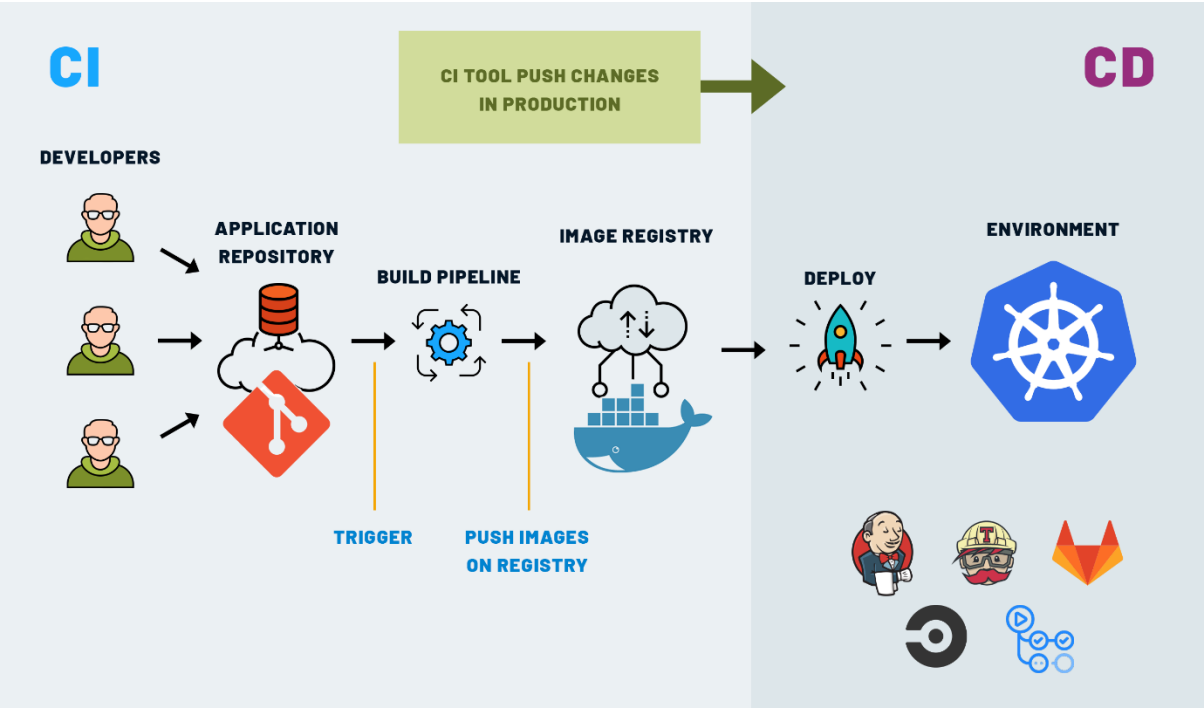
```
triggers:
  - type: aws-sqs-queue
    metadata:
      queueURL: https://sqs.us-east-1.amazonaws.com/1234/myqueue
      awsRegion: "us-east-1"
    authenticationRef:
      name: keda-aws-creds
```

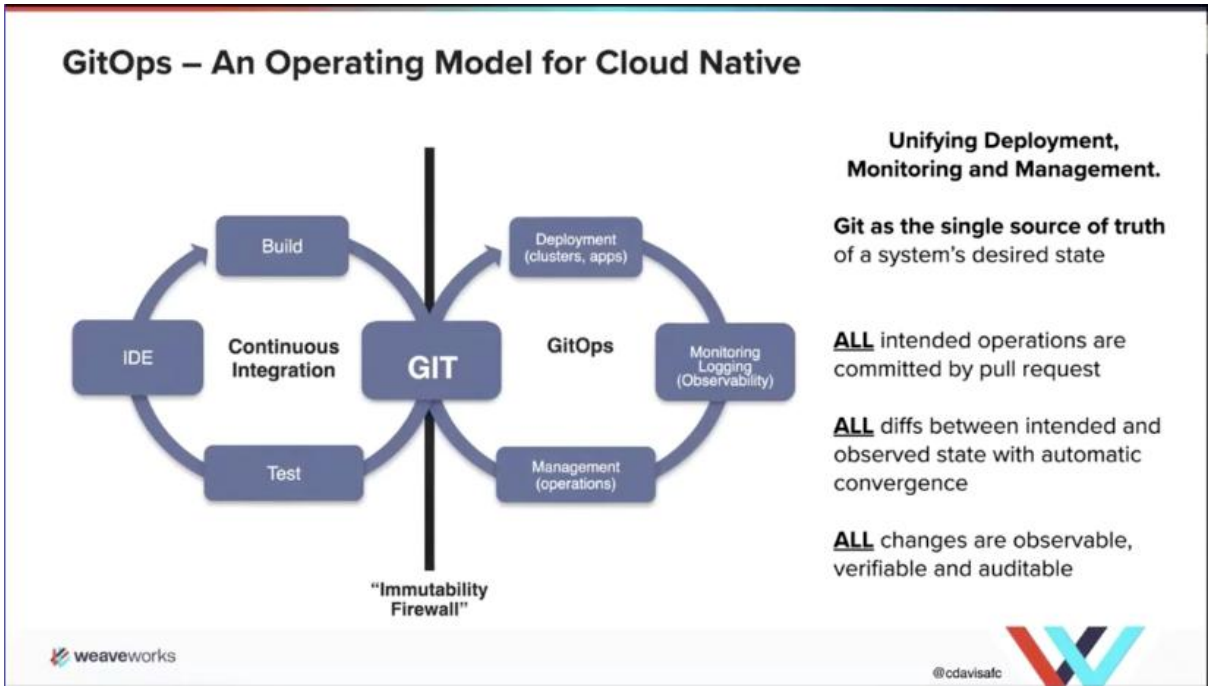
2.4 KEDA Benefits

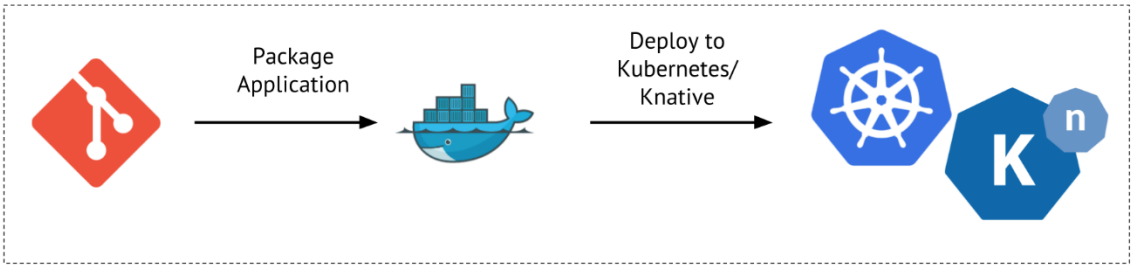
Benefit	Description
Scale-to-zero	No workloads running when queue is empty
Multi-cloud	Supports all major external event sources
Lightweight	Minimal overhead
Fully Kubernetes-native	No external components

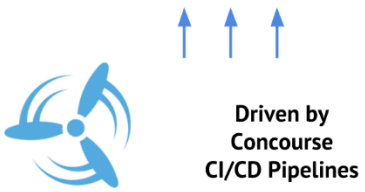
---

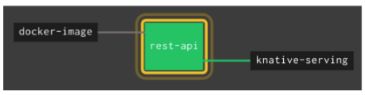
3. Serverless CI/CD & GitOps in Kubernetes











3.1 CI/CD for Serverless Containers

Serverless workloads need:

- Fast build pipelines
- Automatic revision creation
- Traffic shifting on deploy
- Automated rollback

CI/CD Tools:

- GitHub Actions

- GitLab CI
- Tekton Pipelines
- Azure DevOps
- Jenkins X

3.2 Example CI/CD Flow for Knative

1. Developer pushes to Git
2. Pipeline builds container
3. Pushes to registry
4. Updates Knative Service manifest
5. GitOps tool deploys to cluster
6. Knative creates new revision
7. Traffic shifting happens automatically

3.3 GitOps for Serverless

GitOps tools used:

- **Argo CD**
- **Flux CD**

GitOps ensures:

- Declarative deployment
- Auto-sync
- Full revision history
- Instant rollback
- Drift detection

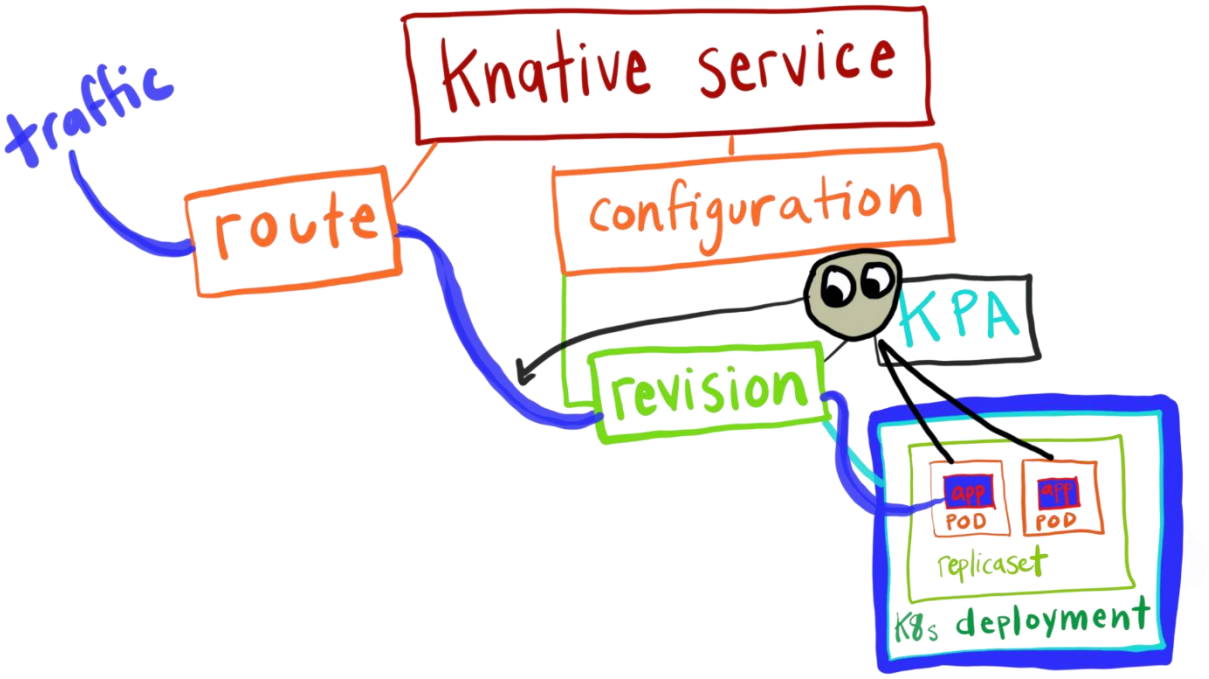
ArgoCD App Example for Knative

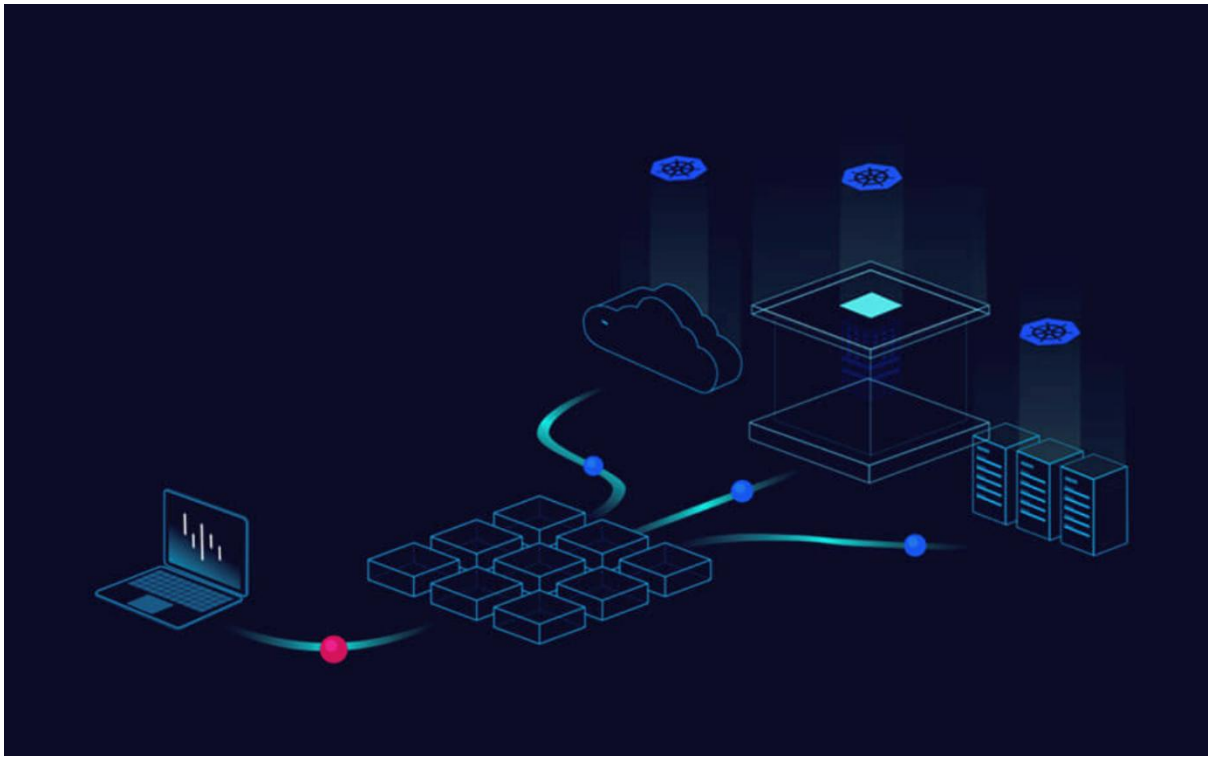
```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: knative-svc
spec:
  source:
```

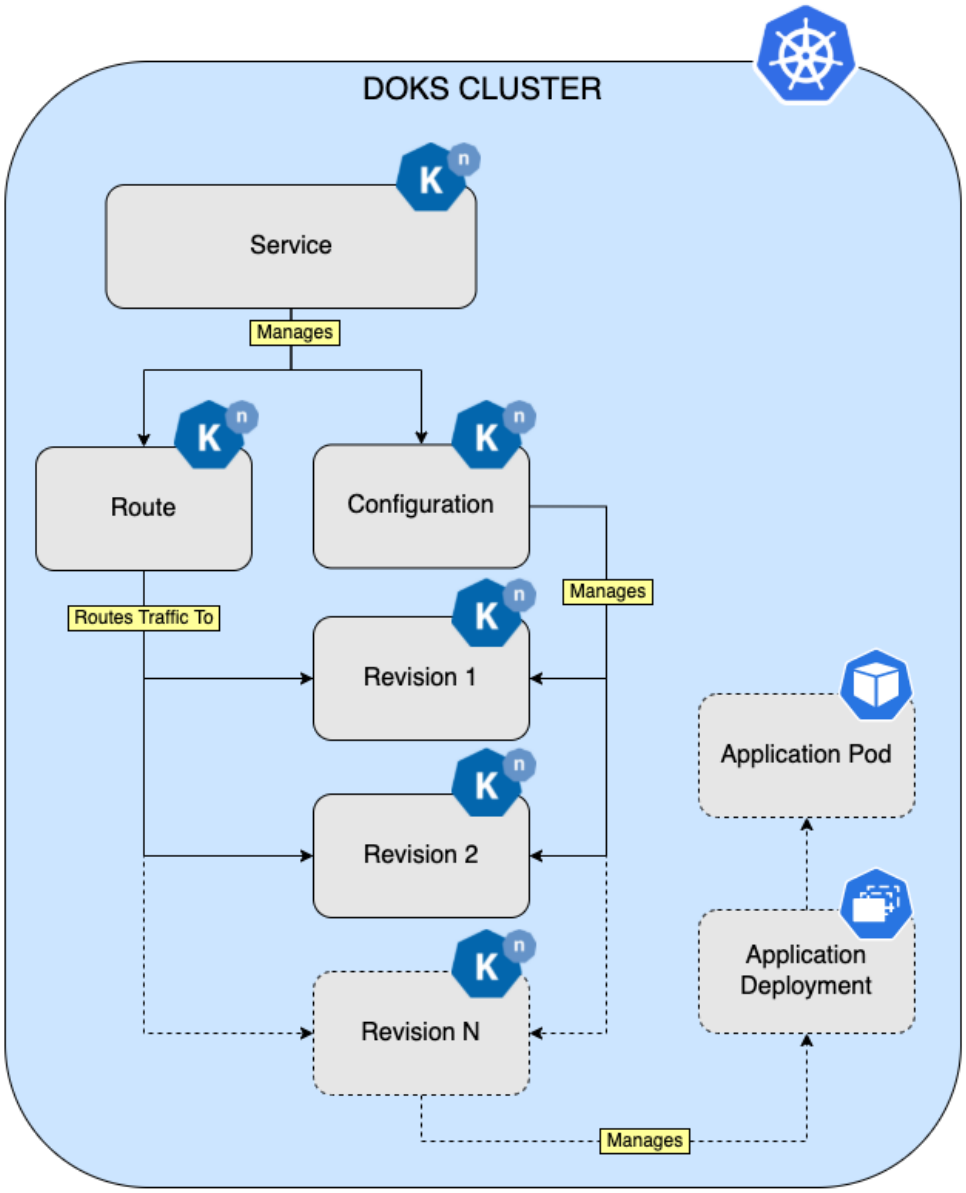
```
  repoURL: https://github.com/vivek/knative-app
  path: deploy
  targetRevision: main
destination:
  namespace: serverless
  server: https://kubernetes.default.svc
syncPolicy:
  automated:
    selfHeal: true
    prune: true
```

---

4. Hands-on Deployment of Serverless Apps (Practical Labs)







Overview:

- User starts by creating a Knative Service for each application.
- Knative Service manages Routes and Configurations for the user application.
- Each application Configuration manages a set of Revisions in time.
- Revisions manage Kubernetes user application Deployments and Pods (Kubernetes layer).

4.1 Lab 1 — Install Knative Serving

Prerequisites:

- Kubernetes cluster (minikube/kind/EKS/GKE/AKS)

Install CRDs:

kubectl apply -f https://github.com/knative/serving/releases/latest/download/serving-crds.yaml

Install core:

kubectl apply -f https://github.com/knative/serving/releases/latest/download/serving-core.yaml

Install networking layer (Kourier):

kubectl apply -f https://github.com/knative/net-kourier/releases/latest/download/kourier.yaml

---

4.2 Lab 2 — Deploy a Serverless Application

Deploy simple Knative service:

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: hello
  namespace: serverless
spec:
  template:
    spec:
      containers:
      - image: gcr.io/knative-samples/helloworld-go
        env:
        - name: TARGET
          value: "K8s Serverless"
```

Apply:

kubectl apply -f hello-knative.yaml

Check URL:

kubectl get ksvc hello

4.3 Lab 3 — Trigger Knative Eventing Workflow

Create Broker:

kubectl apply -f https://github.com/knative/eventing/releases/latest/download/eventing-crds.yaml

kubectl apply -f https://github.com/knative/eventing/releases/latest/download/eventing-core.yaml

Create event source → publish → see response.

4.4 Lab 4 — Deploy KEDA Autoscaler

Install:

helm repo add kedacore https://kedacore.github.io/charts

helm install keda kedacore/keda

Test scaling from 0 → N based on queue length.

Module 8 Summary

Topic	Summary
Knative Serving	Auto-scaling, revisions, traffic routing
Knative Eventing	Event mesh: Broker, Trigger, Sources
KEDA	Event-based autoscaling (scale-to-zero)
Serverless CI/CD	Automated revisions & GitOps deployment
Hands-on Labs	Deploy apps, triggers, autoscaling
