Хичээл 3: Kubernetes I

Г.Гантулга

2024 оны 12-р сарын 5

Kubernetes

Kubernetes гэж юу вэ?

- Зөөвөрлөх, өргөтгөх боломжтой нээлттэй эх платформ.
- Контейнерлэсэн үйлчилгээ, ачааллыг зохицуулахад зориулагдсан.
- Тохируулгыг автоматаар болон зарлан гүйцэтгэх боломжтой.
- Том, эрчтэй хөгжиж буй экосистем
- Хэрэгсэл, туслах материал нь өргөн түгээгдсэн.

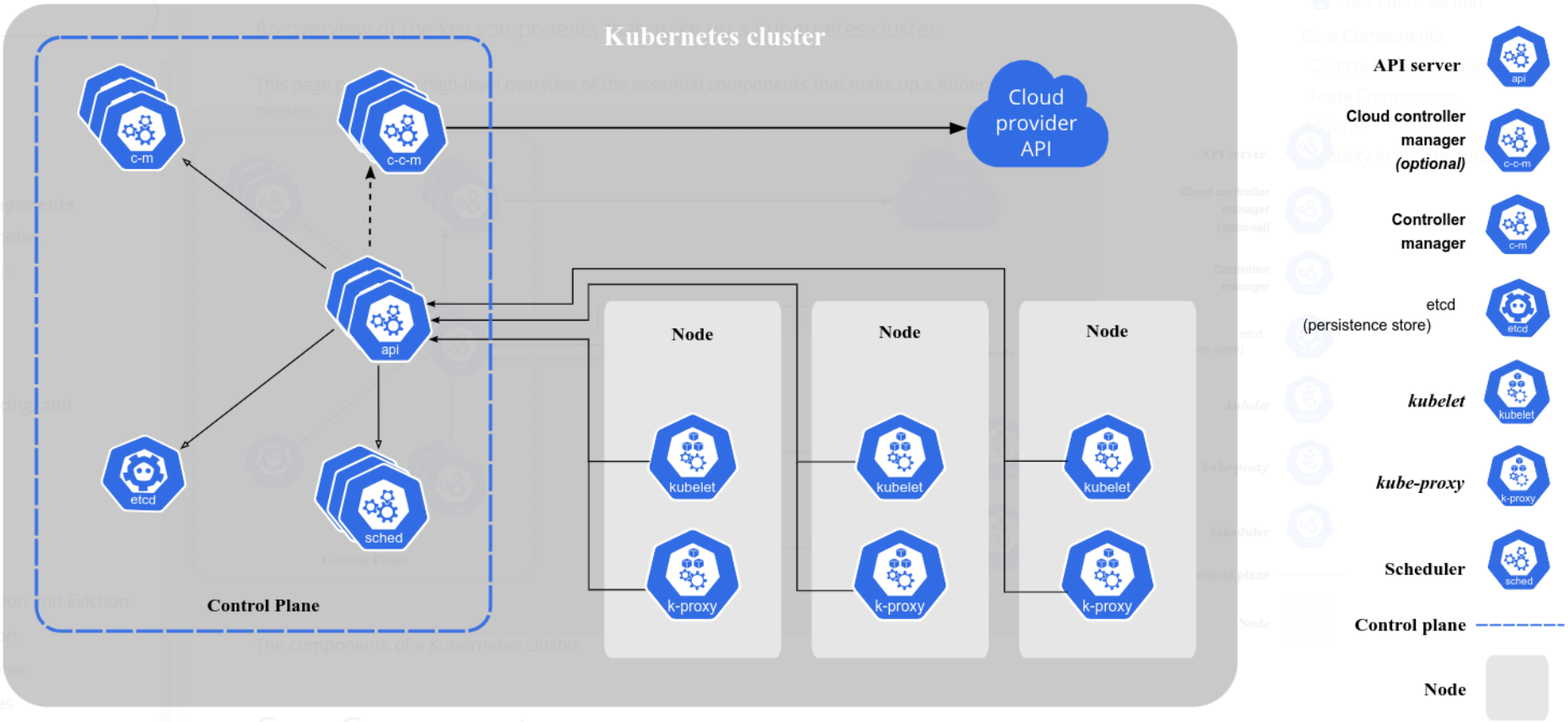
Яагаад kubernetes гэж?

Resillient system

Kubernetes юу чадах вэ?

- Service discovery and load balancing
- Storage orchestration
- Automated rollouts and rollbacks
- Automatic bin packing
- Self-healing
- Secret and configuration management
- Batch execution
- Horizontal scaling
- IPv4/IPv6 dual-stack
- Designed for extensibility

Kubernetes Architecture



K8s объект

K8s объект нь хадгалагддаг төлөвтэй нэгж. Эдгээр объектууд кластерийн төлөвийг дүрсэлнэ.

- Ямар контейнер апп ажиллуулж байгааг
- Эдгээр апп-д зориулагдсан системийн нөөц
- Эдгээр апп-уудын талаарх зохицуулалт: upgrade, restart policy, fault tolerence г.м

Объект үүсгэгдсэн л бол K8s систем уг объектыг оршин байлгахын тулд ажиллана.

K8s объект

Бараг бүх K8s объект дараах хоёр талбартай

- **spec**: Объектыг үүсгэхэд зааж өгнө. Шинж чанар, системийн нөөцийг нь дүрсэлж өгнө.
- **status**: K8s баяжуулна. Одоогийн төлөвийг дүрсэлнэ. Status нь spec-тэйгээ нийцэж байхыг control plane идэвхтэйгээр зохицуулна.

K8s объект

Жишээ Deployment object

Deployment object: Кластерт ажиллаж буй апп-ийг дүрслэх объект

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

nginx-deployment объект үүсэхэд k8s status талбарыг баяжуулна. Хэрэв status нь spec-ээсээ зөрвөл k8s систем энэ зөрүүг арилгахыг идэхтэй оролдно.

```
$ kubectl apply -f deployment
deployment.apps/nginx-deployment\
                              created
$ kubectl delete -f deployment
deployment.apps/nginx-deployment\
                              deleted
```

Объектын заавал байх ёстой талбарууд

Ямар ч объектод заавал байх ёстой талбарууд

- **apiVersion**:Kubernetes API version
- **kind**: What kind of object
- **metadata**: Тухайн объектыг ялгах өгөгдөл: UID, name, namespace г.м
- **spec**: Тухайн объектод ямар төлөв хүсэж байгааг бичнэ.

spec талбарын формат объект, объектоосоо өөр хамаарч өөр байдаг. Үүнийг Kubernetes API reference-ээс харвал зүйтэй.
Kubernetes v1.25-аас эхлэх yaml файлыг API сервер талд validate хийж давхардсан, алдаатай тохиргоог таньж чаддаг болсон.

```
$ kubectl --validate=[strict, warn, ignore] apply -f
```

Labels болон Selector

Labels

Бол Key/Value хоёрын хослол бөгөөд объектуудыг бүлэглэхэд хэрэглэж болно. Объектыг үүсгэсний дараа хэзээ ч хамаагүй, нэмж, өөрчилж болно.

```
apiVersion: v1
kind: Pod
metadata:
  name: label-demo
  labels:
    environment: production
    release: v1.2
    app: nginx
    tier: backend
```

Labels болон Selector

Selectors

name болон UID дахин давтагдашгүй боловч labels бол тийм биш.
Адил label-тай олон объектыг selector ашиглан бүлэглэн сонгоно.
Дараах хоёр янзын selector байдаг:

- Equity based $(=, ==, !=)$

```
environment = production
tier != frontend
```

- Set based (*in*, *notin*)

```
environment in (production, qa)
tier notin (frontend, backend)
```

```
$ kubectl get pods -l environment=production,tier=fr
$ kubectl get pods -l 'environment in (production),t
```

Labels болон Selector

nodeSelector

```
apiVersion: v1
kind: Pod
metadata:
  name: cuda-test
spec:
  containers:
    - name: cuda-test
      image: "registry.k8s.io/cuda-vector-add:v0.1"
      resources:
        limits:
          nvidia.com/gpu: 1
  nodeSelector:
    accelerator: nvidia-tesla-p100
```

Labels болон Selector

matchLabels

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

Labels болон Selector

```
selector:
  matchLabels:
    component: redis
  matchExpressions:
    - { key: tier, operator: In, values: [cache] }
    - { key: environment, operator: NotIn, values: [dev] }
```

Namespace

```
kind: Namespace
apiVersion: v1
metadata:
  name: test
  labels:
    name: test

$ kubectl get namespaces
$ kubectl get pods --namespace=test
```

Namespace

```
apiVersion: v1
kind: Pod

metadata:
  name: pod-demo
  namespace: test
spec:
  containers:
    - name: nginx-app
      image: nginx:latest
      ports:
        - containerPort: 80
```

Pods

Pod

Pod бол k8s-ийн deploy хийгдэх хамгийн бага нэгж. Pod нь нэг эсвэл хэдэн хэдэн контейнер агуулах бөгөөд эдгээр нь хамт төлөвлөгдөх, дундын орчинд байна. Хоорондоо нягт холбоотой контейнеруудийг нэг **pod**-д байршуулна. Хуучнаар бол нэг логик сервертэй зүйрлэж болно.

```
apiVersion: v1
kind: Pod
metadata:
  name: label-demo
  labels:
    environment: production
    release: v1.2
    app: nginx
    tier: backend
```

Pods

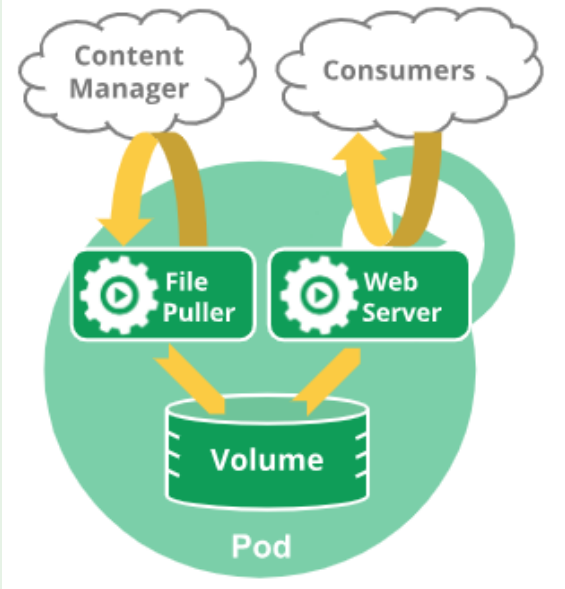
Хэрэглээ

- Pod-ийг шууд үүсгэх ямар ч шаардлагагүй. Харин Workload төрлийн объектоор дамжуулан үүсгэнэ (deployment, statfulset, job г.м).
- Нэг Pod дотор нэг апп ажиллана. Хэрэв хөндлөнгөөр томруулахыг хүсвэл харгалзах workload controller-оор автоматаар эсвэл, гарааг хийгдэнэ.
- Нэг Pod-д байгаа контейнерүүд дундийн storage болон сүлжээг хуваалцах боломжтой.

Pods

Маш нягт холбогдсон контейнерүүд нэг пот дотор орж болно.

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-demo   # DNS compatible name
spec:
   volumes:
  - name: shared-data
    emptyDirp: {}
  containers:
    - name: nginx-app
      image: registry.gitlab.com/inv/nginx
      ports:
        - containerPort: 80
      volumeMounts:
      - name: shared-data
        mountPath: /data
    - name: aws-cli
      image: registry.gitlab.com/inv/aws_sync
      volumeMounts:
      - name: shared-datan
        mountPath: /s3_data
```



Environment, update, replacement

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envars
spec:
  containers:
  - name: envar-demo-container
    image: gcr.io/google-samples/hello-app:2.0
    env:
    - name: DEMO_GREETING
      value: "Hello from the environment"
    - name: DEMO_FAREWELL
      value: "Such a sweet sorrow"
```

- Job
- DaemonSet
- StatfulSet
- Deployment

Pods: Init containers

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app.kubernetes.io/name: MyApp
spec:
  containers:
  - name: myapp-container
    image: busybox:1.28
    command: ['sh', '-c', 'echo The app is running! && sleep 3600']
  initContainers:
  - name: init-myservice
    image: busybox:1.28
    command: ['sh', '-c', "until nslookup myservice.$(cat /var/run/secrets\
         /kubernetes.io/serviceaccount/namespace).svc.cluster.local; do echo\
         waiting for myservice; sleep 2; done"]
  - name: init-mydb
    image: busybox:1.28
    command: ['sh', '-c', "until nslookup mydb.$(cat /var/run/secrets\
        /kubernetes.io/serviceaccount/namespace).svc.cluster.local;\
         do echo waiting for mydb; sleep 2; done"]
```

Pods: Init containers

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app.kubernetes.io/name: MyApp
spec:
  containers:
  - name: myapp-container
    image: busybox:1.28
    command: ['sh', '-c', 'echo The app is running! && sleep 3600']
  initContainers:
  - name: init-myservice
    image: busybox:1.28
    command: ['sh', '-c', "until nslookup myservice.$(cat /var/run/secrets\
         /kubernetes.io/serviceaccount/namespace).svc.cluster.local; do echo\
         waiting for myservice; sleep 2; done"]
  - name: init-mydb
    image: busybox:1.28
    command: ['sh', '-c', "until nslookup mydb.$(cat /var/run/secrets\
        /kubernetes.io/serviceaccount/namespace).svc.cluster.local;\
         do echo waiting for mydb; sleep 2; done"]
```

Pods: Init containers

```
---
apiVersion: v1
kind: Service
metadata:
  name: myservice
spec:
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
---
apiVersion: v1
kind: Service
metadata:
  name: mydb
spec:
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9377
```

Ажлын ачааллын зохицуулал (workload management)

Controllers

- Deployment (ReplicaSet)
- StatefulSet
- DaemonSet
- Jobs

Deployment (ReplicaSet)

Deployment

Апп ажиллуулж байгаа хэд, хэдэн подуудыг удирдана. Эдгээр подууд төлөвгүй байх бөгөөд хоорондоо солигдоход ямар ч асуудалгүй байна.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

Updating Deployment

```
$ kubectl set image deployment/nginx-deployment\
                    nginx=nginx:1.16.1
```

```
$ kubectl edit deployment/nginx-deployment
$ kubectl rollout status deployment/nginx-deployment
$ kubectl describe deployments
```

```
$ kubectl rollout undo deployment/nginx-deployment
```

```
$ kubectl rollout history \
          deployment/nginx-deployment
$ kubectl rollout history \
    deployment/nginx-deployment --revision=2
```

Deployment Yaml бичих

Шаардлагатай талбарууд

- .apiVersion, .kind, .metadata.name
- .spec.template, .spec.selector
- .spec.selector болон .spec.template.metadata.labels хоёр хоорондоо таарч байх ёстой.
- .spec.selector бол immutable

```
.spec.strategy:
   type: RollingUpdate # эсвэл Recreate
   rollingUpdate:
     maxUnavailable: 1
```

Stateful Set

StatefulSet

- Deployment шиг template-ээр pod-ууд үүсгэнэ.
- Pod бүрт ялгац онооно.
- Pod-ууд хоорондоо солигдох боломжгүй. Адил template-ээр үүсгэгдсэн боловч pod-ууд хоорондоо ялгарна.

Хэрэглээ

- Stable, unique network identifiers.
- Stable, persistent storage.
- Ordered, graceful deployment and scaling.
- Ordered, automated rolling updates.

StatefulSets

Хязгаарлалт

- Pod-ийг ашиглах storage-ийг тохируулсан байх.
- Устгах эсвэл scale down хийхэд харгалзах storage устгагдахгүй
- StatefulSets-д Headless Service шаардлагатай.
- StatefulSets-ийг устгахад pod нь заавал зогссон байхыг шаарддаггүй тул устгахын өмнө scale down 0 ашиглах ёстой.
- Rolling Update хийгдэж байхад алдаа гарч болзошгүй. Гарсан алдааг гараар засах.

StatefulSets DNS

- `<servicename>.<namespace>.svc.cluster.local`
- `<name>-N.<servicename>.<namespace>.svc.cluster.local`

StatefulSets

- `<servicename>.<namespace>.svc.cluster.local`
- `<name>-N.<servicename>.<namespace>.svc.cluster.local`

StatefulSets

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx
  serviceName: "nginx"
  replicas:
  minReadySeconds: 10
  template:
    metadata:
      labels:
        app: nginx
```

StatefulSets

```
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: nginx
        image: registry.k8s.io/nginx-slim:0.24
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:

      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 1Gi
```

Volume Claim

```
apiVersion: apps/v1
kind: StatefulSet
...
spec:
  persistentVolumeClaimRetentionPolicy:
    whenDeleted: Retain
    whenScaled: Delete
```

DaemonSets

DaemonSet

Node бүр дээр тухайн DaemonSet-ийн нэг л Pod ажиллана. Node бүрд байх ёстой үйлчилгээг хангах зорилгоор ашиглана. Кластерт node нэмэгдэхэд daemonset-ийн pod автоматаар нэмэгдэнэ.

Хэрэглээ

- Node бүрээс log цуглуулах, монитор хийх
- Кластерт storage daemon node бүрт ажиллуулах

DaemonSets

Deamonset pod-той мэдээлэл солилцох аргууд

- push
- nodeIP and nodePort
- headless service үүсгээд DNS-ийг нь ашиглана. Бүх endpoint буюу A record-уудыг татах авах зарчмаар.
- Service үүсгэж санамсаргүй pod-той холбогдох

Job

Job

Хэд хэдэн подыг асаагаад тодорхой хэд нь амжилттай ажиллаад дуусахад тухайн job дууслаа гэж үзнэ. Эсвэл амжилттай ажиллах хүртэл зөвхөн нэг pod-оор асааж болно.

Хэрэглээ

- Нэг ажлыг заавал амжилттай ажиллаад дууссан байхыг шаардах үед.
- Cluster-ийн нөөцийг параллел тооцоололд ашиглах

Job

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    spec:
      containers:
      - name: pi
        image: perl:5.34.0
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
      restartPolicy: Never
  backoffLimit: 4
```

Cronjob

cronjob

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "* * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
            image: busybox:1.28
            imagePullPolicy: IfNotPresent
            command:
            - /bin/sh
            - -c
            - date; echo Hello from the Kubernetes cluster
          restartPolicy: OnFailure
```

Cronjob

```
# ┌──────────────── minute (0 - 59)
# │ ┌────────────── hour (0 - 23)
# │ │ ┌──────────── day of the month (1 - 31)
# │ │ │ ┌────────── month (1 - 12)
# │ │ │ │ ┌──────── day of the week (0 - 6) (Sunday
# │ │ │ │ │         to Saturday) OR sun, mon, tue, wed
# │ │ │ │ │          , thu, fri, sat
# │ │ │ │ │
# * * * * *
```

Replication Controller

Autoscaling workloads

Scaling

- Cronscaling
- Event driven scaler
- Autoscaling based on cluster size (vertically)
- Based on CPU and Memory usage

```
$ kubectl autoscale deployment nginx-deployment
                --cpu-percent=80 --min=1 --max=10
```

Canary Deployment

stable

```
name: frontend
replicas: 3
...
labels:
   app: guestbook
   tier: frontend
   track: stable
...
image: gb-frontend:v3
```

service.yml

```
selector:
   app: guestbook
   tier: frontend
```

canary

```
name: frontend-canary
replicas: 1
...
labels:
   app: guestbook
   tier: frontend
   track: canary
...
image: gb-frontend:v4
```

Services and LoadBalancing

- Pod бүр IP хаягтай.
- Cluster-т байгаа бүх подууд хоорондоо мэдээлэл солилцох боломжтой.
- Service урт хугацааны IP хаяг эсвэл DNS нэрийг backend pod-уудад олгоно.
- Gateway (Ingress) cluster-ийн гадна талаас service-д хандах боломжийг олгоно.

Service

Service-ийн хэрэгцээ

- Pod-ууд хурдан өөрчлөгддөг. Ихсэж, багасаж, унтарч, асаж байдаг.
- Deployment үүсгэсэн pod-уудаа динамикаар удирдана.
- Апп-уудтай хэрхэн холбогдох вэ?
- Сервис объектын select хийсэн pod-уудтай холбогдоход тухайн сервисийг ашиглана.

Service

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app.kubernetes.io/name: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

Selector-гүй Service

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 9376
```

Хэрэглээ

- Гадна талд байгаа үйлчилгээг дүрслэх
- Зарим backend-үүд хараахан kubernetes-т орж ирээгүй байх

Manual Endpoint Slice

```
apiVersion: discovery.k8s.io/v1
kind: EndpointSlice
metadata:
  name: my-service-1
  labels:
    kubernetes.io/service-name: my-service
addressType: IPv4
ports:
  - name: http
    appProtocol: http
    protocol: TCP
    port: 9376
endpoints:
  - addresses:
      - "10.4.5.6"
  - addresses:
      - "10.1.2.3"
```

Multi port service

service.yml

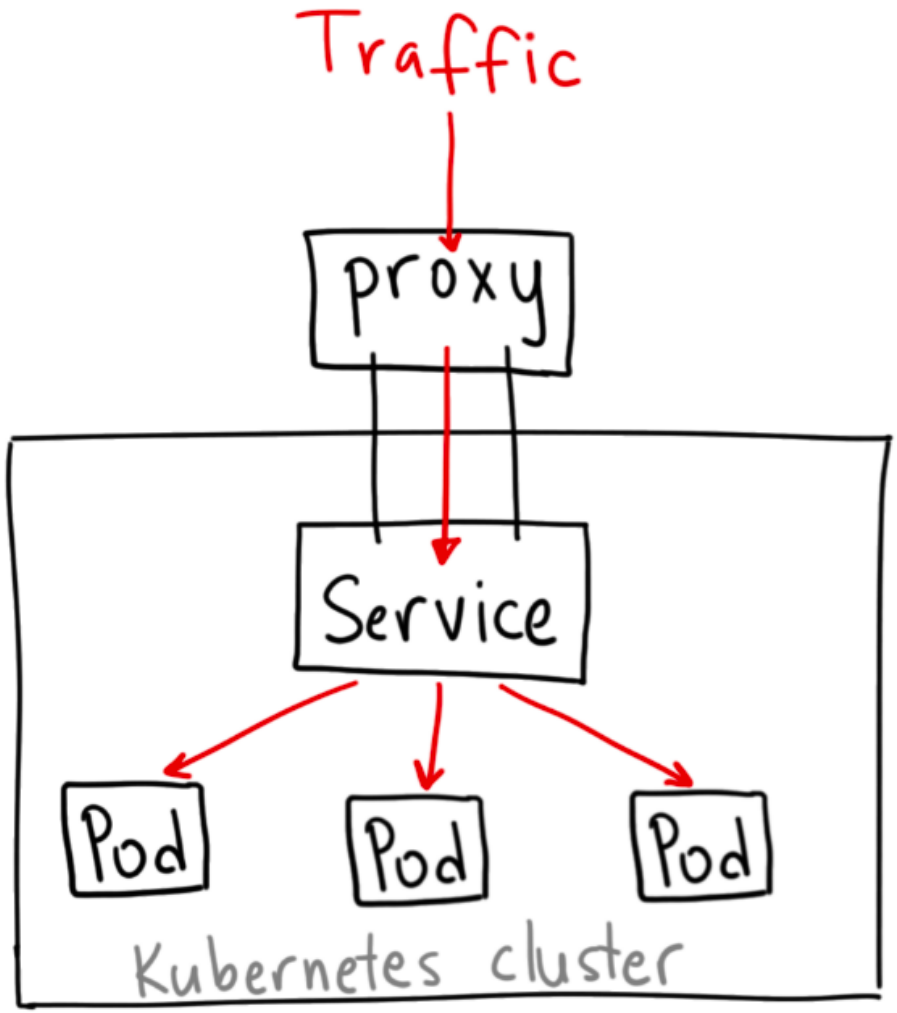
```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 9376
    - name: https
      protocol: TCP
      port: 443
      targetPort: 9377
```

<portname>.<protocol>.<servicename>.<namespace>.svc.cluster.local

Service-ийн төрөл

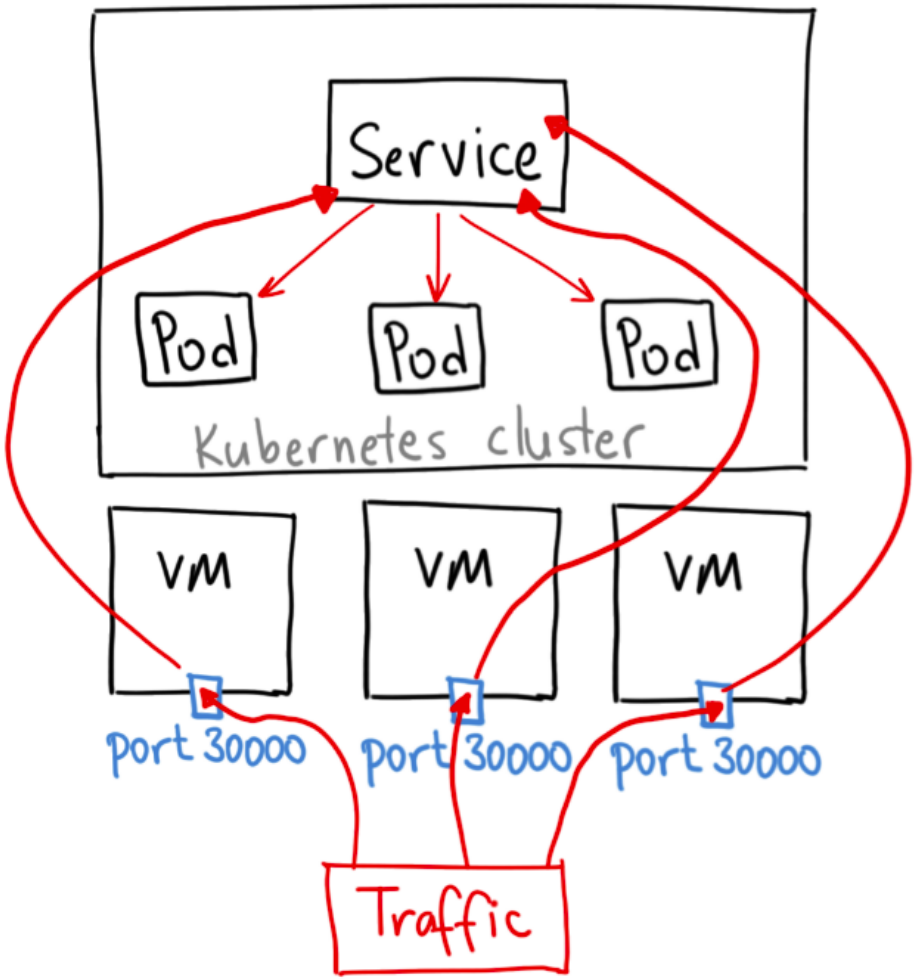
- ClusterIP (None)
- NodePort
- LoadBalancer
- ExternalName

ClusterIP



ClusterIP

NodePort

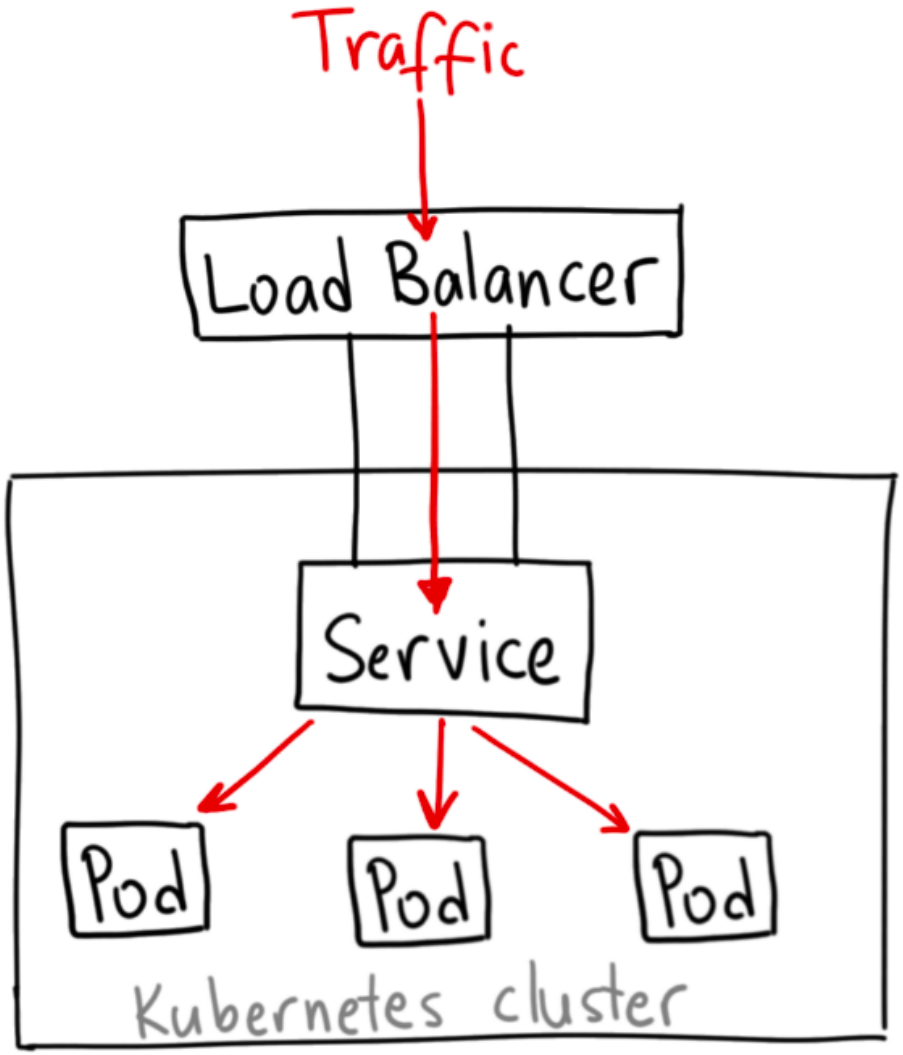


NodePort

NodePort

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: NodePort
  selector:
    app.kubernetes.io/name: MyApp
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30007
```

LoadBalancer



Loadbalancer

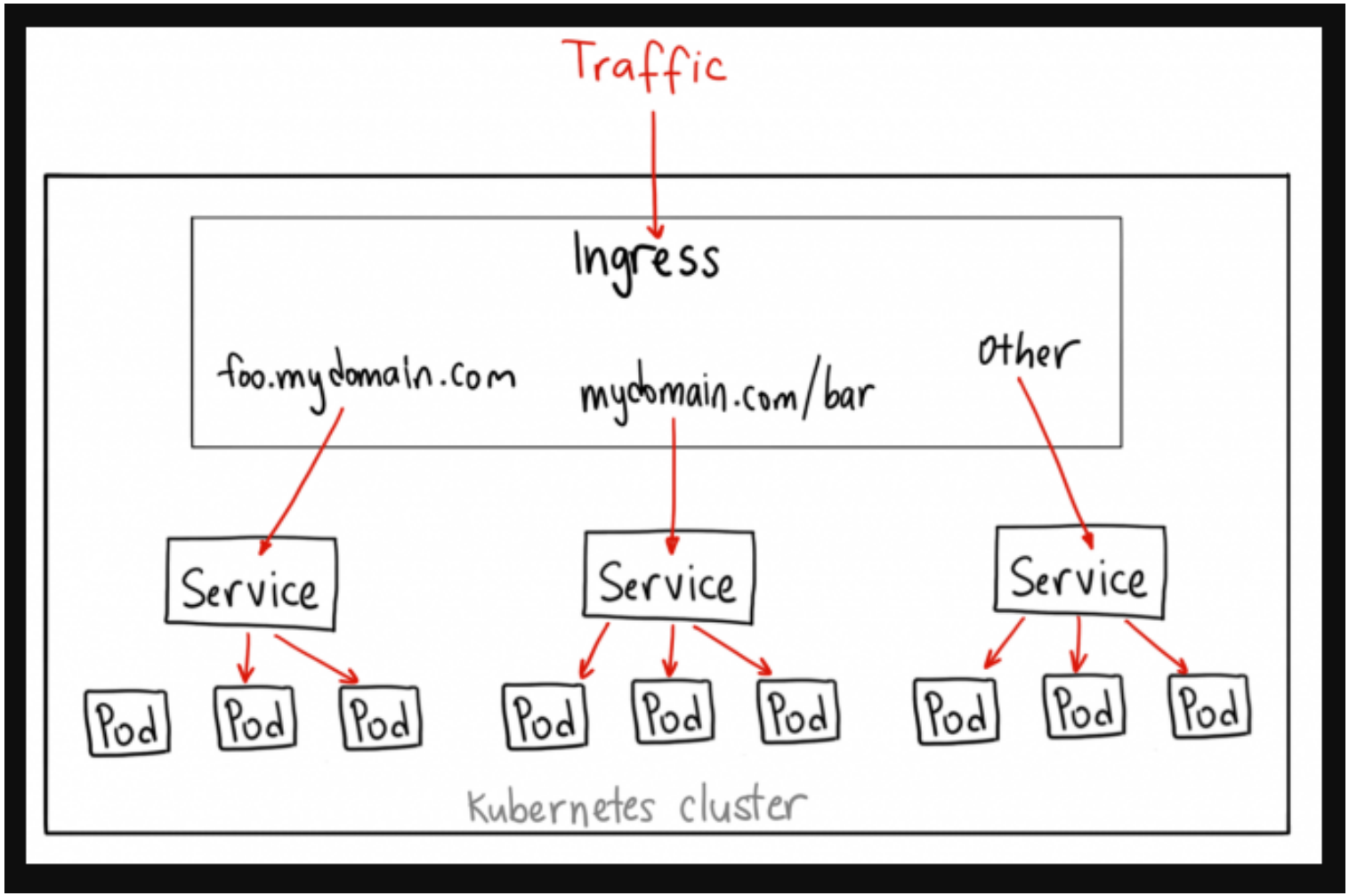
LoadBalancer

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app.kubernetes.io/name: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
  type: LoadBalancer
  loadBalancerIP: 10.0.171.239
```

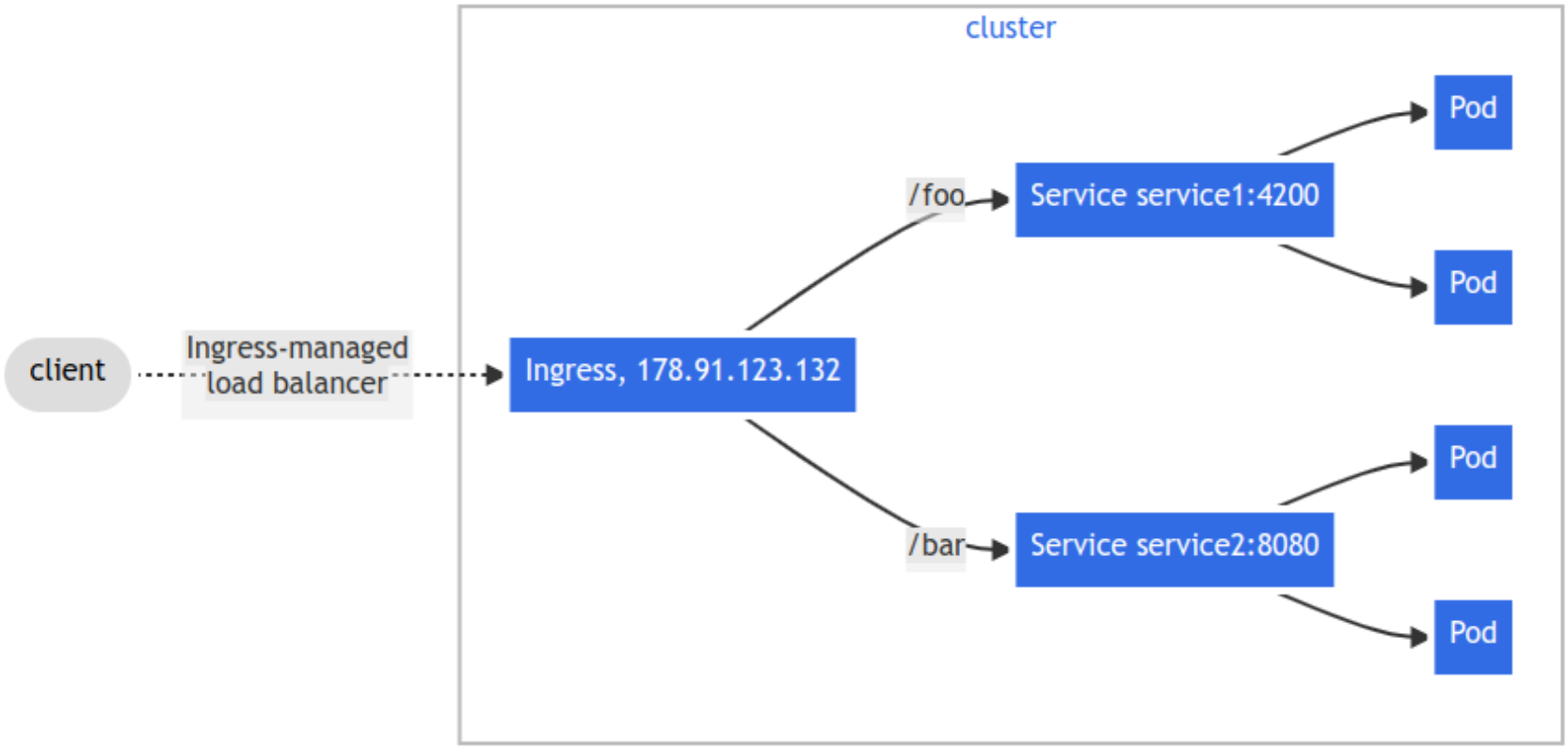
ExternalName

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
  namespace: prod
spec:
  type: ExternalName
  externalName: my.database.example.com
```

Ingress


Traffic
Ingress
foo.mydomain.com
mydomain.com/bar
other
Service
Service
Service
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Kubernetes cluster

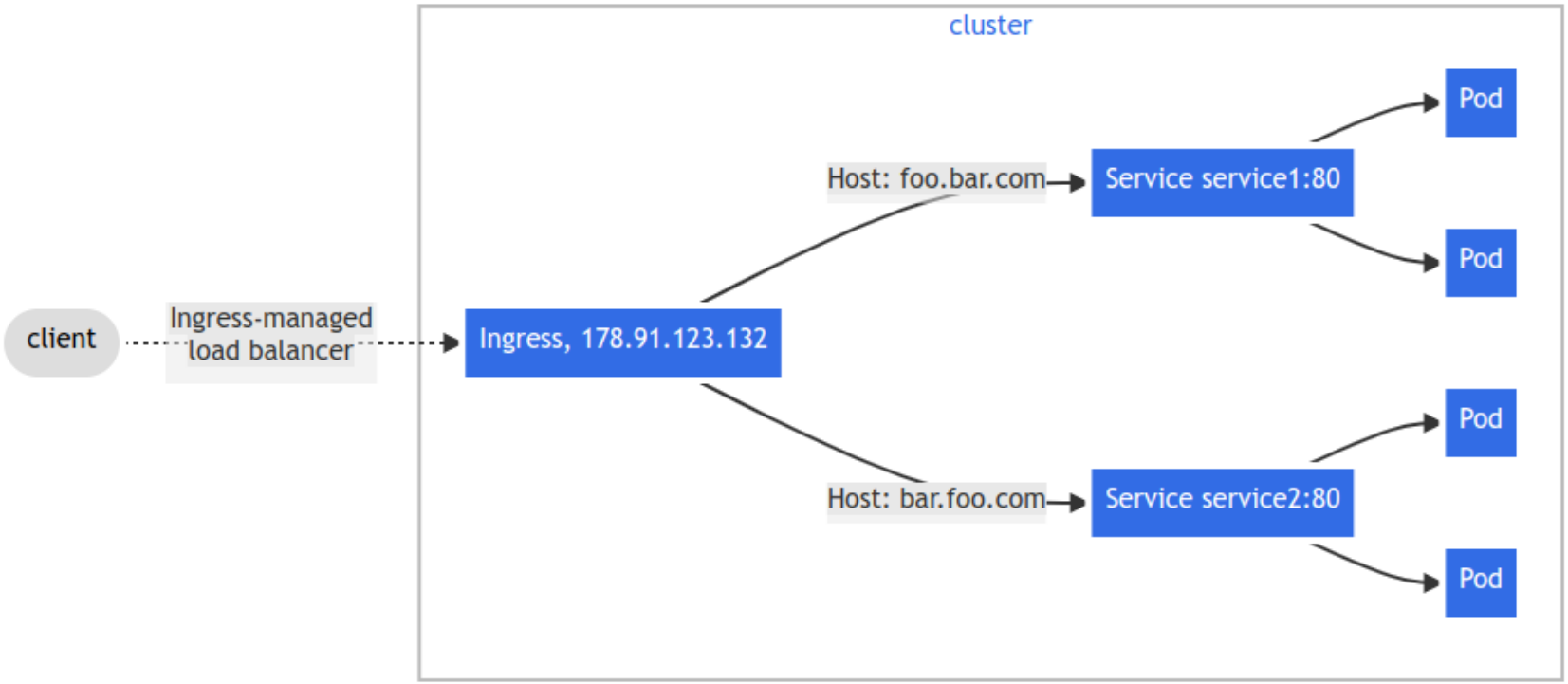
Ingress


cluster
client
Ingress-managed
load balancer
Ingress, 178.91.123.132
/foo
/bar
Service service1:4200
Service service2:8080
Pod
Pod
Pod
Pod

Ingress

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: simple-fanout-example
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        pathType: Prefix
        backend:
          service:
            name: service1
            port:
              number: 4200
      - path: /bar
        pathType: Prefix
        backend:
          service:
            name: service2
            port:
              number: 8080
```

Ingress


cluster
client
Ingress-managed load balancer
Ingress, 178.91.123.132
Host: foo.bar.com
Service service1:80
Pod
Pod
Host: bar.foo.com
Service service2:80
Pod
Pod

Ingress

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: name-virtual-host-ingress
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: service1
            port:
              number: 80
- host: bar.foo.com
  http:
    paths:
    - pathType: Prefix
      path: "/"
      backend:
        service:
          name: service2
          port:
            number: 80
```

Ingress Controller

- AWS
- GCE
- nginx

Gateway API

gateway API?

API-ийн төрлүүд: Advanced Routing, dynamic infrastructure provisioning

- extensible
- role-oriented
- protocol-aware configuration

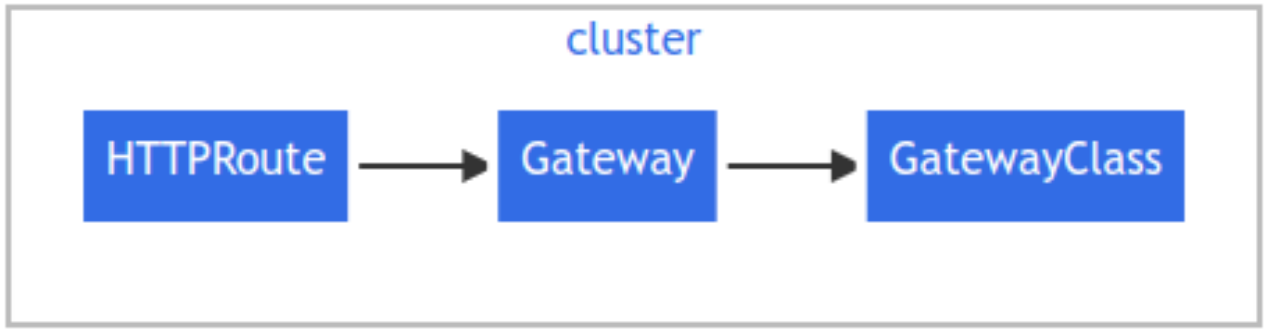
Gateway API

gateway API

- role oriented
  - Infrastructure Provider
  - Cluster operator
  - Application Developer
- Portable: supported by many implementations
- Expressive: traffic routing using header-based-matching, traffic weighting
- Extensible: Custom resources at various layers

Resource model

- GatewayClass
- Gateway
- HTTPRoute



GatewayClass

```
apiVersion: gateway.networking.k8s.io/v1
kind: GatewayClass
metadata:
  name: example-class
spec:
  controllerName: example.com/gateway-controller
```

GKE GatewayClasses

GatewayClass name	Description
`gke-l7-global-external-managed`	Global external Application Load Balancer(s) built on the global external Application Load Balancer
`gke-l7-regional-external-managed`	Regional external Application Load Balancer(s) built on the regional external Application Load Balancer
`gke-l7-rilb`	Internal Application Load Balancer(s) built on the internal Application Load Balancer
`gke-l7-gxlb`	Global external Application Load Balancer(s) built on the classic Application Load Balance
`gke-l7-global-external-managed-mc`	Multi-cluster Global external Application Load Balancer(s) built on the global external Application Load Balancer
`gke-l7-regional-external-managed-mc`	Multi-cluster Regional external Application Load Balancer(s) built on the global external Application Load Balancer
`gke-l7-rilb-mc`	Multi-cluster Internal Application Load Balancer(s) built on the internal Application Load Balancer
`gke-l7-gxlb-mc`	Multi-cluster Global external Application Load Balancer(s) built on the classic Application Load Balancer
`asm-l7-gxlb`	Global external Application Load Balancer(s) built on Cloud Service Mesh

Gateway

```
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: example-gateway
spec:
  gatewayClassName: gke-17-gxlb
  listeners:
  - name: http
    protocol: HTTP
    port: 80
```

HTTPRoute

```
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: example-httproute
spec:
  parentRefs:
  - name: example-gateway
  hostnames:
  - "www.example.com"
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /login
    backendRefs:
    - name: example-svc
      port: 8080
```

Gateway request flow

Storage

Long term, short term storages to pods

Volumes

Хэрэглээ

- Pod бол нэг удаагийн хэрэгцээ. Crash-лаж, нэмэгдэж, устгагдаж, дахин эхлэгдэж байдаг. Pod устахад бүх өгөгдөл устана (image-ээс үүсгэгдсэн).
- Нэг pod дотор байгаа хэд хэдэн контейнер өгөгдөл хуваалцах.
- Volume бол өгөгдөл агуулсан хавтас

Ангилал

- Ephemeral volume
- Persistent volume

Volume төрөл

- ConfigMap
- EmptyDir (safe across crashes)
- isci
- local (statically created PersistentVolume)
- hostPath
- downwardAPI
- image
- nfs
- persistentVolumeClaim
- projected
- secret

Volume

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: registry.k8s.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir:
      sizeLimit: 500Mi
```

Persistent Volumes

Persistent Volume

- Storage in cluster
- Admin allocated or dynamically provisioned using StorageClass
- Cluster resource

PersistentVolumeClaim (PVC)

- Request for storage by user
- Comsumes PV resource
- Size, access mode

Lifecycle of a volume and claim

1. Provisioning
   1. Static (Admin)
   2. Dynamic (via StorageClass)
2. Binding: Match PVC to PV
3. Using: Pod mounts PVC
4. Storage Object in Use Protection

   ```
   $ kubectl describe pcv <pcvname>
   ```

5. Reclaiming
   - Retain (delete pv, clean up data)
   - Delete (deletes pv and external storage)

Persistent Volume-ийн төрлүүд

- csi - Container Storage Interface (CSI)
- fc - Fibre Channel (FC) storage
- hostPath - HostPath volume (for single node testing only; WILL NOT WORK in a multi-node cluster; consider using local volume instead)
- iscsi - iSCSI (SCSI over IP) storage
- local - local storage devices mounted on nodes.
- nfs - Network File System (NFS) storage

Persistent Volume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0003
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /tmp
    server: 172.17.0.2
```

Persistent Volume

- volumeMode
  - Filesystem
  - Block
- AccessMode
  - ReadWriteOnce
  - ReadOnlyMany
  - ReadWriteMany
  - ReadWriteOncePod
- Class (storageClassName should be set)
- Reclaim Policy
  - Retain
  - Recycle
  - Delete
- nodeAffinity

Persistent Volume Claims

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 8Gi
  storageClassName: slow
  selector:
    matchLabels:
      release: "stable"
    matchExpressions:
      - {key: environment, operator: In, values: [dev]}
```

Claim as volume

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
      - mountPath: "/var/www/html"
        name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: myclaim
```

Best practices

- When defining configurations, specify the latest stable API version.
- Configuration files should be stored in version control before being pushed to the cluster. This allows you to quickly roll back a configuration change if necessary. It also aids cluster re-creation and restoration.
- Write your configuration files using YAML rather than JSON. Though these formats can be used interchangeably in almost all scenarios, YAML tends to be more user-friendly.
- Group related objects into a single file whenever it makes sense. One file is often easier to manage than several. See the guestbook-all-in-one.yaml file as an example of this syntax.
- Note also that many kubectl commands can be called on a directory. For example, you can call kubectl apply on a directory of config files.
- Don't specify default values unnecessarily: simple, minimal configuration will make errors less likely.
- Put object descriptions in annotations, to allow better introspection.

ConfigMaps

```
piVersion: v1
kind: ConfigMap
metadata:
  name: game-demo
data:
  player_initial_lives: "3"
  ui_properties_file_name: "user-interface.properties"
  game.properties: |
    enemy.types=aliens,monsters
    player.maximum-lives=5
  user-interface.properties: |
    color.good=purple
    color.bad=yellow
    allow.textmode=true
binaryData:
  demo.bin: AAECAwQFBgcICQoLDA0ODw==
```

Using ConfigMaps

```
apiVersion: v1
kind: Pod
metadata:
  name: configmap-demo-pod
spec:
  containers:
    - name: demo
      image: alpine
      command: ["sleep", "3600"]
      env:
        - name: PLAYER_INITIAL_LIVES
          valueFrom:
            configMapKeyRef:
              name: game-demo
              key: player_initial_lives
        - name: UI_PROPERTIES_FILE_NAME
          valueFrom:
            configMapKeyRef:
              name: game-demo
              key: ui_properties_file_name
      volumeMounts:
      - name: config
        mountPath: "/config"
        readOnly: true
```

Using ConfigMaps

```
volumes:
- name: config
  configMap:
    name: game-demo
    items:
    - key: "game.properties"
      path: "game.properties"
    - key: "user-interface.properties"
    path: "user-interface.properties"
```

Secrets

```
apiVersion: v1
kind: Secret
metadata:
  name: dotfile-secret
data:
  .secret-file: dmFsdWUtMgOKDQo=
```

Secrets

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-dotfiles-pod
spec:
  volumes:
    - name: secret-volume
      secret:
        secretName: dotfile-secret
  containers:
    - name: dotfile-test-container
      image: registry.k8s.io/busybox
      command:
        - ls
        - "-l"
        - "/etc/secret-volume"
      volumeMounts:
        - name: secret-volume
          readOnly: true
          mountPath: "/etc/secret-volume"
```

Types of secret

Built-in Type	Usage
`Opaque`	arbitrary user-defined data
`kubernetes.io/service-account-token`	ServiceAccount token
`kubernetes.io/dockercfg`	serialized `~/.dockercfg` file
`kubernetes.io/dockerconfigjson`	serialized `~/.docker/config.json` file
`kubernetes.io/basic-auth`	credentials for basic authentication
`kubernetes.io/ssh-auth`	credentials for SSH authentication
`kubernetes.io/tls`	data for a TLS client or server
`bootstrap.kubernetes.io/token`	bootstrap token data

SSH auth

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-ssh-auth
type: kubernetes.io/ssh-auth
data:
  # the data is abbreviated in this example
  ssh-privatekey: |
    UG91cmluZzYlRW1vdGljb24lU2N1YmE=
```