Хичээл 3: Kubernetes I

Г.Гантулга

2024 оны 12-р сарын 3

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.

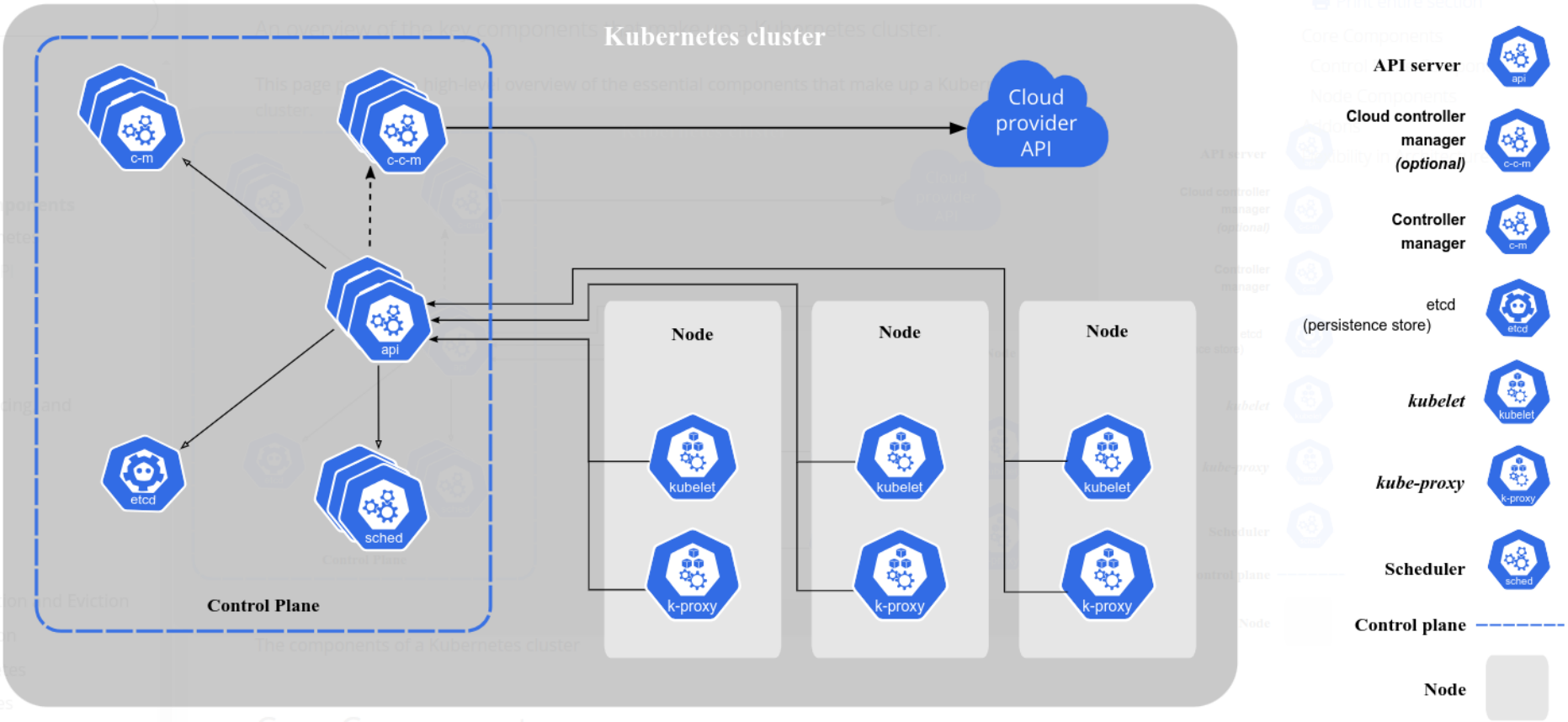
Яагаад kubernetes гэж?

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

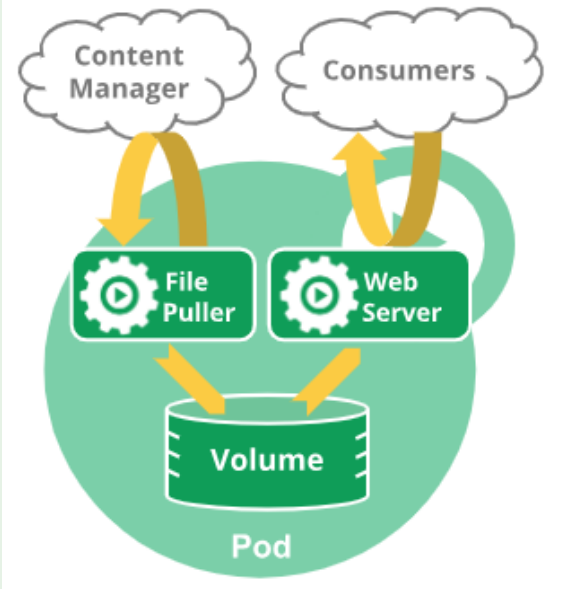
Хэрэглээ

- 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-уууд 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
```

DaemonSets

Jobs

Automatic Cleanup

Cronjob

Replication Controller

Autoscaling workloads

Service

Ingress

Ingress Controller

Volumes

Persistent Volumes

Projected Volumes

Volume snapshot

Best practices

ConfigMaps

Secrets