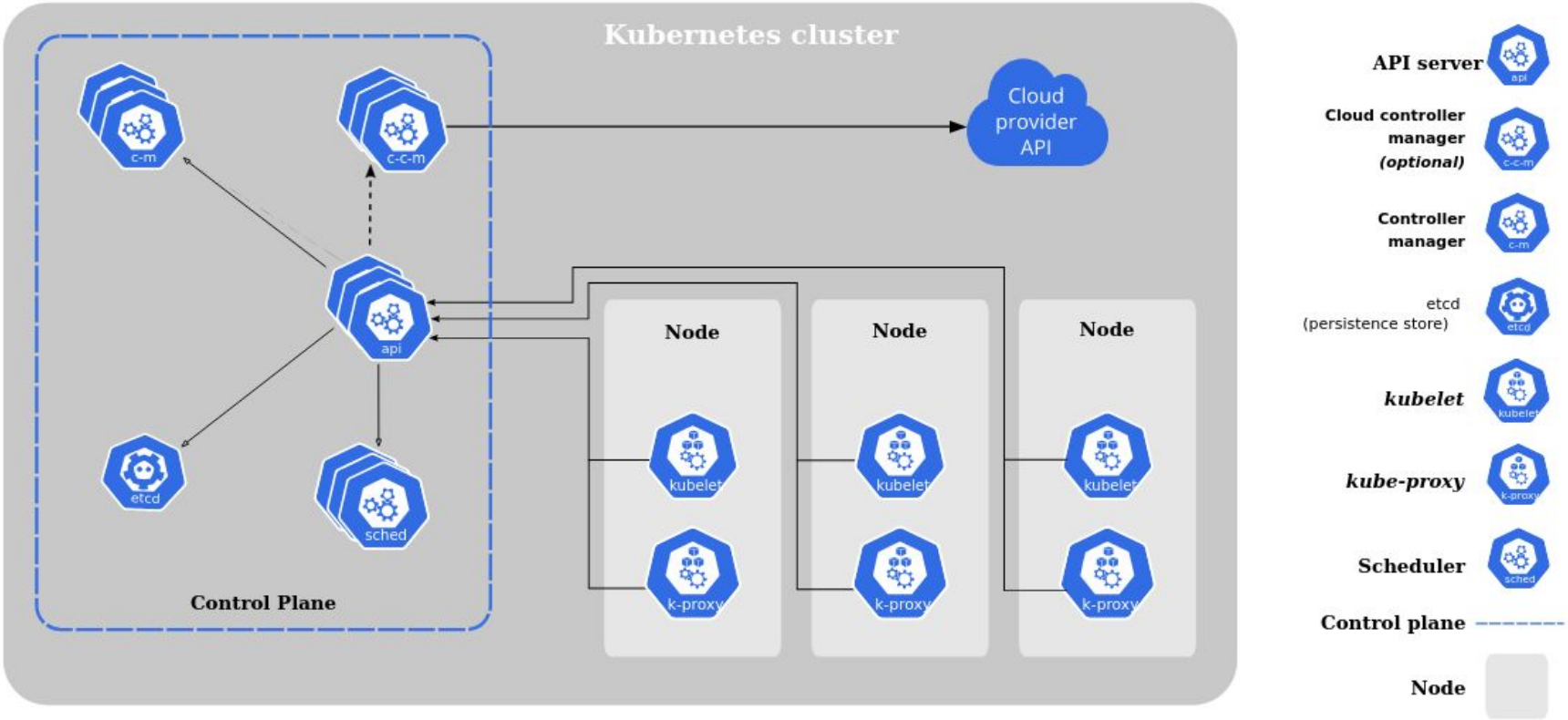
Kubernetes

julia schaak s0563733

Kubernetes wird verwendet für….

- Management/Orchestrierung von virtuellen Servern Containern/Services
- “Pods” = kleinste Einheit
- auf den Pods laufen “containerized applications”
  - teilen sich Container-Runtime (Docker, containerd…)& evtl. weitere Ressourcen
- Pods werden auf “Nodes” deployed (virtuelle oder physische Maschinen in einem Cluster) auch Worker genannt
- Kubernetes dient der Überwachung und Herstellung eines Sollzustandes
- diese Zustand wird deklarativ mittels .yaml file konfiguriert

Kubernetes Komponenten



Quelle: https://kubernetes.io/docs/concepts/overview/components/

wichtige Komponenten sind:

Control Plane Components:

- high level controlling
- fährt z.B.: neuen Pods hoch, wenn replica field im Deployment nicht erreicht ist
- kube-apiserver
  - kubectl’s Ansprechpartner
- etcd
  - Key-Value Speicher für alle Klusterdaten (Deployments, Secrets,...)
- kube-scheduler
  - weist z.B. neu erstellten Pods ihren Node zu (weiß was ein Node kann und weist entsprechend zu)
- kube-controller-manager
  - Node Controller: reagiert z.B. wenn Nodes nicht verfügbar sind
  - Replication Controller: korrekte Anzahl v. Pods
  - Endpoints Controller: verbindet Services and Pods
  - Service Account & Token Controllers: managed z.B. Secrets
- cloud-controller-manager: (optional) im lokalen Cluster nicht nötig

die anderen wichtigen Komponenten sind:

Node Components:

- werden auf jedem Node ausgeführt, zuständig für Fehlerfreies Laufen der Pods
  - **Kubelet**: “mittleres Management” zuständig für Container der Pods auf dem Node
  - **Kube-proxy**: sorgen dafür dass Netzwerkregeln auch auf dem Node gelten
  - **Container Runtime**: Software, die die Container laufen lässt z.B. ~~Docker~~, Containerd

Addons:

- DNS
- Web UI (Dashboard)
- Container Resource Monitoring
- Cluster-level Logging …etc.

und dann gibt es auch noch Kubernetes-Objekte

- Deployments
- Services
- Ingress
- StatefulSet
- u.v.m…

… diese Objekte “tun” erstmal nichts, sondern definieren eine Ressource.

Um etwas zu “tun” braucht es wieder einen “controller” z.B. Ingress-controller -> kann ein “Pod” sein.

(V Körbes)

und kubectl?

- CLI tool v. Kubernetes z.B.
- zum Erstellen eines “Deployments” (checkt Status eines Pods und fährt erneut hoch wenn nicht o.k.)
  - `kubectl create deployment hello-node --image=k8s.gcr.io/echoserver:1.4`
- Anzeigen der Pod Infos
  - `kubectl get po -A`
- Anzeigen der configuration yaml
  - `kubectl get deployment -n default -o yaml`

```
(base) julis@TempestStorm:~$ kubectl get deployment -n default -o yaml
apiVersion: v1
items:
- apiVersion: apps/v1
  kind: Deployment
  metadata:
    annotations:
      deployment.kubernetes.io/revision: "1"
    creationTimestamp: "2022-02-28T11:02:08Z"
    generation: 1
    labels:
      app: hello-minikube
    name: hello-minikube
    namespace: default
    resourceVersion: "1179"
    uid: f7443a07-bb76-40ef-adfd-f1c86b904a38
  spec:
    progressDeadlineSeconds: 600
    replicas: 1
    revisionHistoryLimit: 10
    selector:
      matchLabels:
        app: hello-minikube
    strategy:
      rollingUpdate:
        maxSurge: 25%
        maxUnavailable: 25%
      type: RollingUpdate
    template:
      metadata:
        creationTimestamp: null
        labels:
          app: hello-minikube
      spec:
```

“Art” des Kubernetes-Objektes: Deployment

```
(base) julis@TempestStorm:~$ kubectl get deployment -n default -o yaml
apiVersion: v1
items:
- apiVersion: apps/v1
  kind: Deployment
  metadata:
    annotations:
      deployment.kubernetes.io/revision: "1"
    creationTimestamp: "2022-02-28T11:02:08Z"
    generation: 1
    labels:
      app: hello-minikube
    name: hello-minikube
    namespace: default
    resourceVersion: "1179"
    uid: f7443a07-bb76-40ef-adfd-f1c86b904a38
  spec:
    progressDeadlineSeconds: 600
    replicas: 1
    revisionHistoryLimit: 10
    selector:
      matchLabels:
        app: hello-minikube
    strategy:
      rollingUpdate:
        maxSurge: 25%
        maxUnavailable: 25%
      type: RollingUpdate
    template:
      metadata:
        creationTimestamp: null
        labels:
          app: hello-minikube
      spec:
        containers:
```

Metadaten zum Auffinden des Kubernetes-Objektes

```
(base) julis@TempestStorm:~$ kubectl get deployment -n default -o yaml
apiVersion: v1
items:
- apiVersion: apps/v1
  kind: Deployment
  metadata:
    annotations:
      deployment.kubernetes.io/revision: "1"
    creationTimestamp: "2022-02-28T11:02:08Z"
    generation: 1
    labels:
      app: hello-minikube
    name: hello-minikube
    namespace: default
    resourceVersion: "1179"
    uid: f7443a07-bb76-40ef-adfd-f1c86b904a38
  spec:
    progressDeadlineSeconds: 600
    replicas: 1
    revisionHistoryLimit: 10
    selector:
      matchLabels:
        app: hello-minikube
    strategy:
      rollingUpdate:
        maxSurge: 25%
        maxUnavailable: 25%
      type: RollingUpdate
    template:
      metadata:
        creationTimestamp: null
        labels:
          app: hello-minikube
      spec:
        containers:
```

Konfiguration des Deployments:
Wie viele Repliken will ich?
Wie soll potentiell up-ge-datet werden?

```
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: hello-minikube
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: hello-minikube
    spec:
      containers:
      - image: k8s.gcr.io/echoserver:1.4
        imagePullPolicy: IfNotPresent
        name: echoserver
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
```

Definition zum Erstellen des Pods

```
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: hello-minikube
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: hello-minikube
    spec:
      containers:
      - image: k8s.gcr.io/echoserver:1.4
        imagePullPolicy: IfNotPresent
        name: echoserver
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
```

Konfiguration der enthaltenen Container:
hier der Echoserver der hello-minicube App

Und ein Service ist dann:

- die logische Abstraktion einer Aufgabe, die eine Gruppe von Pods im Cluster zu erfüllen hat.
- z.B.: 5 Pods bieten einen Webservice an.
- Das Service Objekt sorgt dafür, dass der Webservice unter einem einzigen internen Namen und IP adressierbar ist
- Wichtig! für die interne Kommunikation gedacht

```
(base) julis@TempestStorm:~$ kubectl get service -n default -o yaml
apiVersion: v1
items:
- apiVersion: v1
  kind: Service
  metadata:
    creationTimestamp: "2022-02-28T11:03:04Z"
    labels:
      app: hello-minikube
    name: hello-minikube
    namespace: default
    resourceVersion: "1212"
    uid: 24b28805-9da9-4ed3-bdeb-901901d4d5ac
  spec:
    clusterIP: 10.97.218.227
    clusterIPs:
    - 10.97.218.227
    externalTrafficPolicy: Cluster
    internalTrafficPolicy: Cluster
    ipFamilies:
    - IPv4
    ipFamilyPolicy: SingleStack
    ports:
    - nodePort: 31761
      port: 8080
      protocol: TCP
      targetPort: 8080
    selector:
      app: hello-minikube
    sessionAffinity: None
    type: NodePort
  status:
    loadBalancer: {}
```

clusterIP Servicetyp andere Servicetypes sind loadbalancer, nodeport and external name

Portdefinition
(optional mapping von Incoming Port auf Target Port)

Und Ingress?

- zuständig für die externe Kommunikation
- nimmt z.B. einen http-request entgegen
- “läuft” damit die benötigten Services ab
- kann diesen eine extern erreichbare URL geben
- oder auch als Loadbalancer fungieren

```
(base) julis@TempestStorm:~$ kubectl get ingress -n default -o yaml
apiVersion: v1
items: []
kind: List
metadata:
  resourceVersion: ""
  selfLink: ""
```

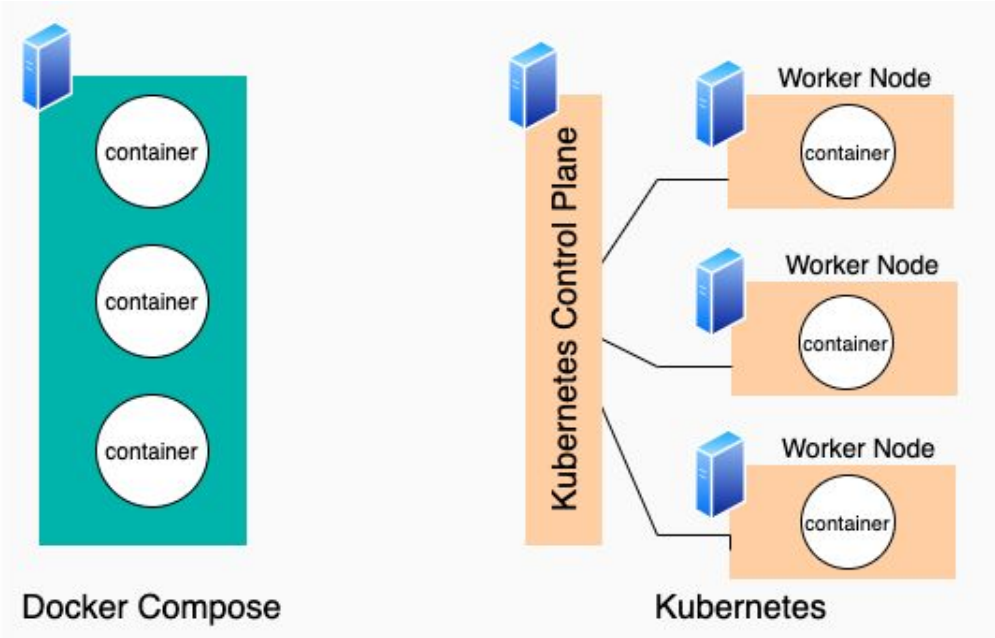
- sieht auf meinem default node so aus, denn: kein externer Traffic

Und Ingress.

```yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
  namespace: kube-system
spec:
  ingressClassName: nginx
  rules:
    - host: hello-john.test
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: hello-world-app
                port:
                  number: 80
    - host: hello-jane.test
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: hello-world-app
                port:
                  number: 80
---
```

Kubernetes vs. Docker-Compose

- beides Container-Orchestrierungs-Frameworks
- Kubernetes lässt Container auf einem verteilten System aus mehreren (virtuellen) Maschinen laufen, Docker startet alle Container auf einer Host-Maschine



Quelle:
https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/What-is-Kubernetes-vs-Docker-Compose-How-these-DevOps-tools-compare

Vorteile von Kubernetes ggü Docker-Compose

- das Verteilen von Containern, Pods, und Services -> skalierbarer als Docker-Container-Verbund auf einzelner Maschine
- Kubernetes bietet automatische Skalierung und Fehler-Ausgleich zur Laufzeit
- Manuelles Überwachen und wieder Hochfahren eines Dockercontainers fällt in Kubernetes weg

Quellen:

- https://kubernetes.io/de/docs/concepts/overview/components/
- https://kubernetes.io/docs/tutorials/hello-minikube/
- https://minikube.sigs.k8s.io/docs/handbook/dashboard/
- https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/What-is-Kubernetes-vs-Docker-Compose-How-these-DevOps-tools-compare
- https://www.vmware.com/topics/glossary/content/kubernetes-services.html