Архитектура вычислительных систем

Kubernetes

Романюта Алексей Андреевич

alexey-r.98@yandex.ru

Кафедра вычислительных систем
Сибирский государственный университет телекоммуникаций и информатики

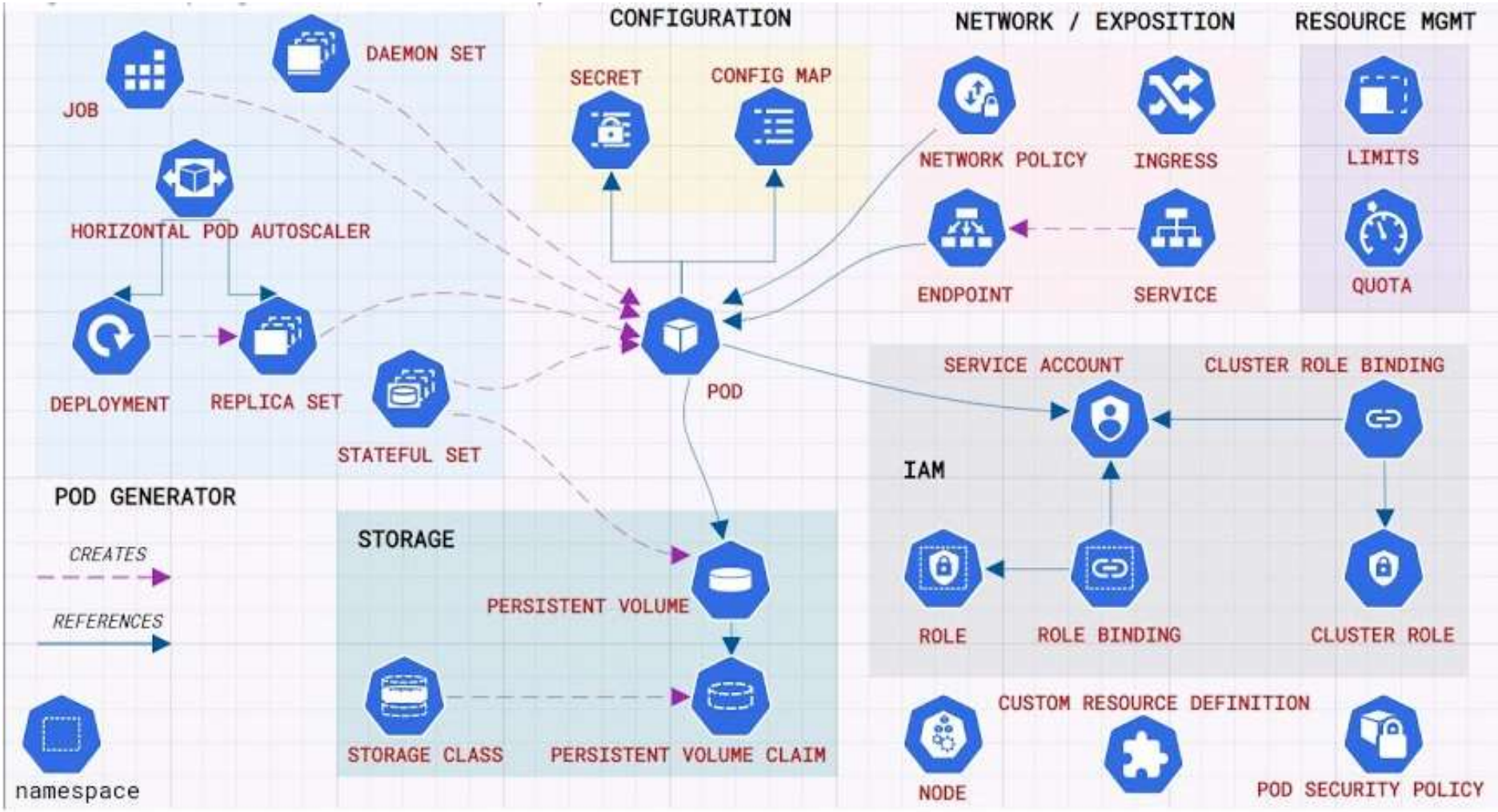
Kubernetes

- «Kubernetes – 5 бинарников»
- 5 бинарников
  - kubelet
  - kubeproxy
  - kube-scheduler
  - kube-controller-manager
  - kube-apiserver

Kubernetes: Как устроено

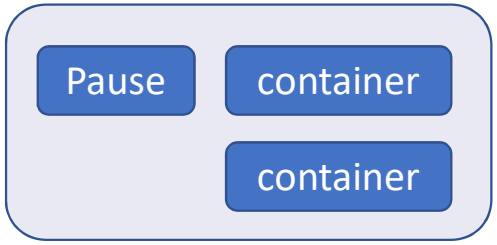
- Control-plane – узел, выделенный для управления кластером и отслеживания его состояния. На этом узле запускаются api-server, controller-manager, scheduler. База данных etcd так же может быть запущена на master-узле
- Worker-node – узел, предназначенный для запуска пользовательских приложений
- Добавление нового сервера в кластер осуществляется обращением к control plane с нового узла
- Пока в кластере не запущен CNI, сетевой связности между контейнерами не будет. Узлы в состоянии NotReady

Kubernetes: Основные сущности



Kubernetes

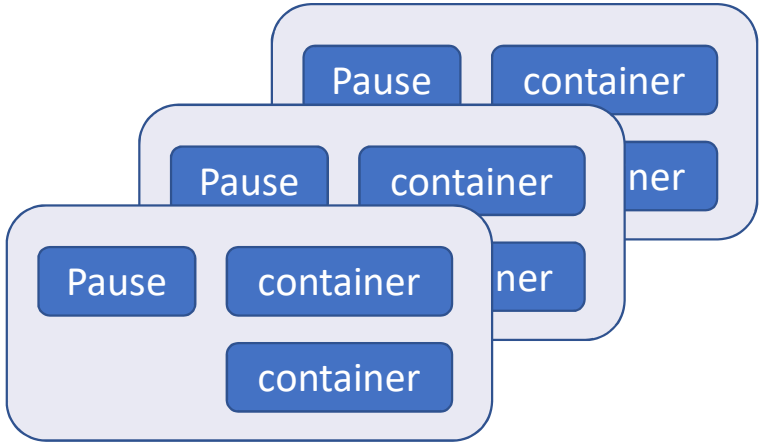
- Несмотря на то, что конечные приложения – контейнеры, минимальная единица абстракции это Pod
- Pod может состоять из нескольких контейнеров
- Контейнеры имеют общий network namespace – можно общаться через localhost
- Неявно создается pause-контейнер, задача которого обеспечить работу namespace, чтобы не зависеть от перезапусков основного приложения




```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: test
  name: test
  namespace: default
spec:
  containers:
    - name: test
      image: test:2
      command:
        - bash
        - /cm/initScript
      imagePullPolicy: IfNotPresent
```

Kubernetes

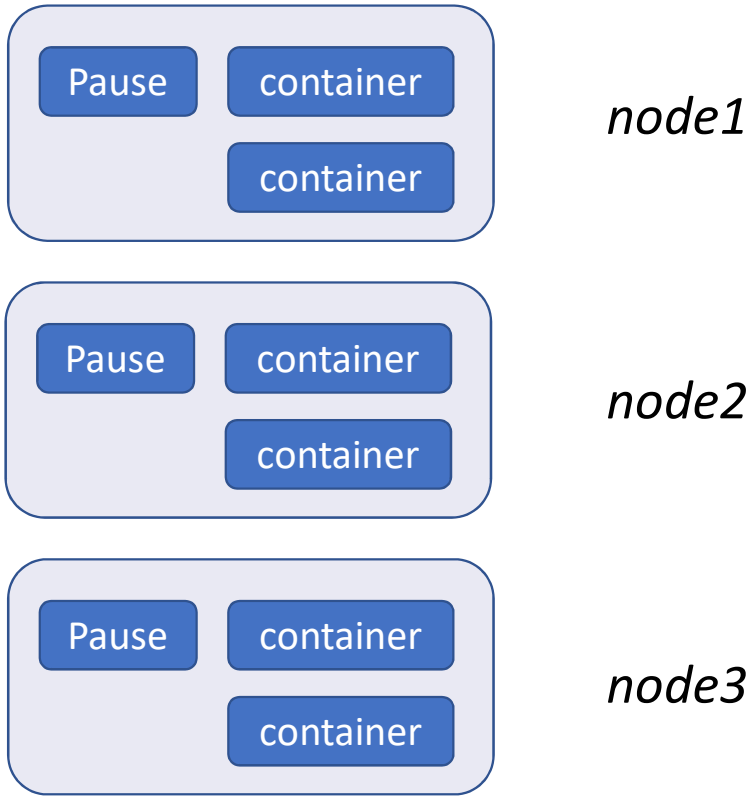
- ReplicaSet предназначен для гарантии работы N экземпляров сущности Pod
- Сколько указано в поле replicas, столько подов будет создано




```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  labels:
    app: test
  name: test
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: test
  template:
    metadata:
      labels:
        app: test
        app.kubernetes.io/name: test
    spec:
      containers:
        - name: test
          image: test:2
          command:
            - bash
            - /cm/initScript
          imagePullPolicy: IfNotPresent
```

Kubernetes

- DaemonSet предназначен для экземпляра Pod на *каждом узле* кластера




```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  labels:
    app: test
  name: test
  namespace: default
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: test
  template:
    metadata:
      labels:
        app: test
        app.kubernetes.io/name: test
    spec:
      containers:
        - name: test
          image: test:2
          command:
            - bash
            - /cm/initScript
          imagePullPolicy: IfNotPresent
```

Kubernetes

- Deployment обеспечивает версионирование приложения
- Deployment создает ReplicaSet
- Если обновляется конфигурация Deployment – Создается новый ReplicaSet
- Политики обновления
  - RollingUpdate – один старый контейнер удалить, один новый создать
  - Recreate – Убить все поды и сразу, и создать сразу все сначала

```
$ kubectl get deployment
NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
ktest-wrk01                   1/1        1           1        1d
$ kubectl get replicaset
NAME                        DESIRED   CURRENT   READY       AGE
ktest-wrk01-77449555bb        1          1         1        1d
$ kubectl get pod
NAME                         READY     STATUS    RESTARTS   AGE
ktest-wrk01-77449555bb-2gvwk 1/1      Running      0        18h
```

Kubernetes

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: test
  name: test
  namespace: default
spec:
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app.kubernetes.io/name: test
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
```

```
  template:
    metadata:
      labels:
        app: test
        app.kubernetes.io/name: test
    spec:
      containers:
        - name: test
          image: test:2
          command:
            - bash
            - /cm/initScript
          imagePullPolicy: IfNotPresent
```

Kubernetes

- StatefulSet – ориентирован на приложения, критичным к сохранению состояния
- Базы данных в k8s это 60/40 bad practice, но можно попробовать
- Если сервер с подом не отвечает – новый Pod не будет создан пока не будет гарантий что старый не работает
- volumeClaimTemplate – managed создание сущности pvc для каждого пода

```
$ kubectl get statefulset
NAME                    READY   AGE
sts-test                  1/1    1d
$ kubectl get pod
NAME                    READY      STATUS    RESTARTS    AGE
sts-test-0                1/1      Running      0         1d
```

Kubernetes

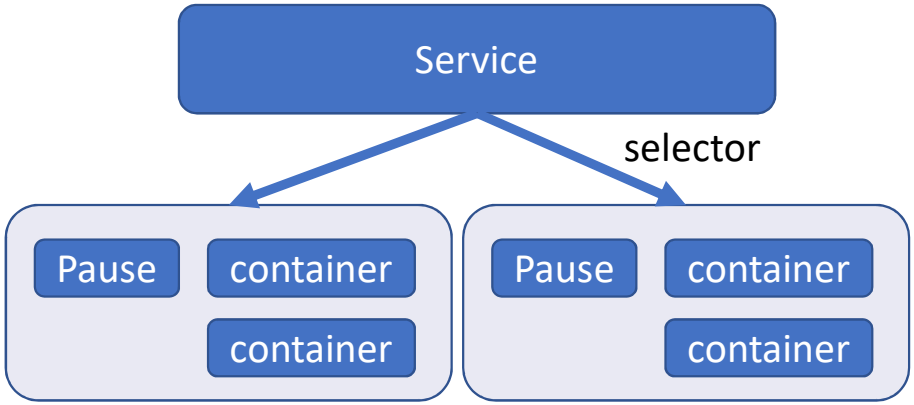
```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  labels:
    app: test
  name: test
  namespace: default
spec:
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app.kubernetes.io/name: test
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
```

```
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: test
        app.kubernetes.io/name: test
    spec:
      containers:
        - name: test
          image: test:2
          command:
            - bash
            - /cm/initScript
          imagePullPolicy: IfNotPresent
```

Kubernetes

- Service – Сущность для обеспечения доступа к подам
- Pod-ы определяются по меткам – labels
- Балансировка по Pod-ам – Round Robin
- Типы сервисов
  - ClusterIP
  - LoadBalancer – требует установки провайдера балансировщика, например metalb (или cloud – aws, hetzner, gcp balancers)
  - NodePort – на каждом узле кластера один порт (как правило 30000-32767) используется для доступа к сервису
  - Неявный - Headless service

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/name: test
  name: test
  namespace: default
spec:
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: http
  selector:
    app.kubernetes.io/name: test
  type: ClusterIP
```



Kubernetes

- Headless service – сервис, к которому невозможно направить трафик
- Содержит в себе список всех pod-ов, попадающих под selector
- Полезно, когда приложению необходимо знать адреса всех своих экземпляров или для dns-discovery

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/name: test-headless
  name: test-headless
  namespace: default
spec:
  clusterIP: None
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: http
  selector:
    app.kubernetes.io/name: test
  type: ClusterIP
```

Kubernetes: Probes

- Startup – запустился ли контейнер?
- Readiness – может ли контейнер принимать трафик?
- Liveness – контейнер все еще живой?

Что происходит, если проба прошла/не прошла?

Kubernetes: Probes

- Startup – запустился ли контейнер?
- Readiness – может ли контейнер принимать трафик?
- Liveness – контейнер все еще живой?

Что происходит, если проба прошла/не прошла?

- Liveness – если проба не прошла, контейнер должен быть перезапущен
- Startup используется для задержки запуска liveness пробы
- Readiness – если проба прошла, то адрес пода может быть добавлен в endpoints соответствующих сервисов и принимать трафик.
- Если проба не прошла, трафик на под не направляется. Адрес удаляется из endpoints.

Kubernetes: CSI

- PersistentVolume и PersistentVolumeClaim
- Драйвер Container Storage Interface позволяет на основе claim и указанного в нём storageClass создать pv – persistent volume
- PV можно создать без драйвера – это как правило относится к таким типам хранилищ как nfs, hostPath. В случаях, когда провайдером PV выступает облако, s3 или распределенная ФС (cephfs) pv должен создаваться провайдером
- Провайдер позволяет изменять размер PV путем изменения запроса (PVC), но:
  - Нельзя уменьшить PV без пересоздания (Только миграция данных)
  - Напрямую нельзя изменить размер PVC, созданного statefulset через volumeClaimTemplates

Kubernetes: CSI, режимы доступа

- ReadWriteOnce – только один под может примонтировать как read-write
- ReadOnlyMany – монтирование только на чтение.
- ReadWriteMany – монтирование на чтение-запись несколькими Pod (nfs, ceph)
- ReadWriteOncePod – только один Pod может примонтировать

[Link] https://kubernetes.io/docs/concepts/storage/persistent-volumes/#access-modes

Kubernetes: volumes

```
spec:
  ...
  volumes:
  - name: test-data
    emptyDir:
      sizeLimit: 500Mi
  - name: test-config
    configMap:
      name: test-config-vol
```

```
template:
   metadata:
     creationTimestamp: null
     labels:
       app: test
       app.kubernetes.io/name: test
   spec:
     containers:
     - name: test
       command:
       - bash
       - /cm/initScript
       image: test:2
       imagePullPolicy: IfNotPresent
     volumeMounts:
       - name: test-data
         mountPath: /test-data
       - name: test-config-vol
         mountPath: /test-conf
```

[Link] https://kubernetes.io/docs/concepts/storage/volumes

Kubernetes: statefulset volumes

```
volumeClaimTemplates:
  - metadata:
      name: test-data
    spec:
      accessModes:
      - ReadWriteOnce
      resources:
        requests:
          storage: 8Gi
      storageClassName: nfs-sc
```

```
template:
   metadata:
     labels:
       app: test
       app.kubernetes.io/name: test
   spec:
     containers:
     - command:
       - bash
       - /cm/initScript
       image: test:2
       imagePullPolicy: IfNotPresent
       name: test
     volumeMounts:
       - name: test-data
         mountPath: /test-data
```

Kubernetes

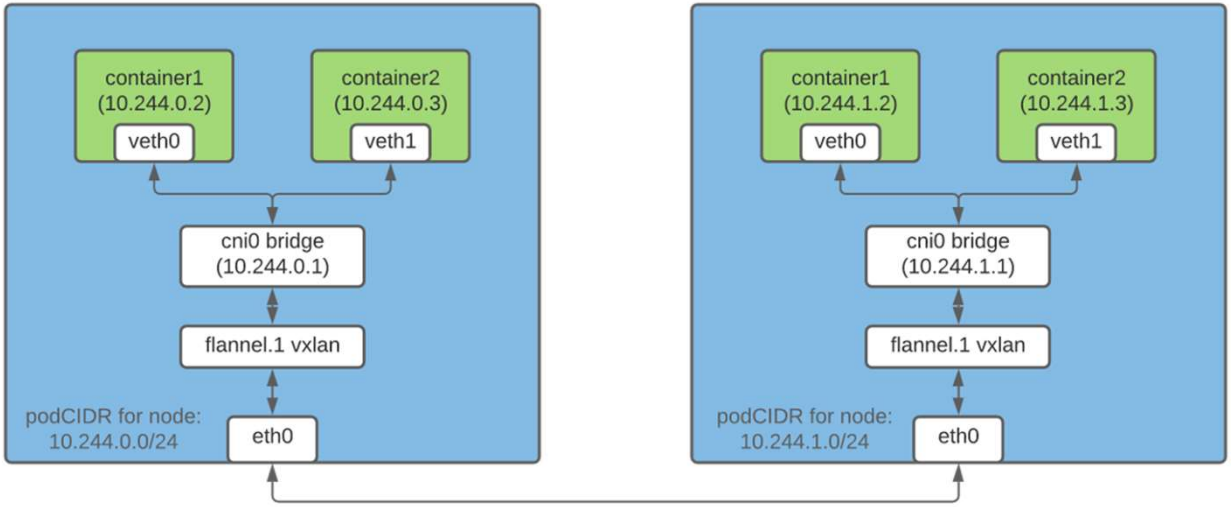
- Ingress – предназначен для доступа к приложениям внутри кластера k8s
- Ориентирован на http/https
- Требует контроллер, например:
  - Nginx ingress controller
  - Traefik ingress controller
- Контроллер настраивается с указанием ingressClassName
- Tls?

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: test-ingress
  namespace: default
spec:
  ingressClassName: nginx
  rules:
  - host: mysuperdomain
    http:
      paths:
      - backend:
          service:
            name: testservice
            port:
              number: 80
        path: /
        pathType: Prefix
  tls:
  - hosts:
    - mysuperdomain
    secretName: mysuperdomain-tls
```

[Link] https://docs.google.com/spreadsheets/d/191WWNpjJ2za6-nbG4ZoUMXMpUK8KlCIosvQB0f-oq3k/edit

Kubernetes: CNI

- Container Network Interface представляет собой спецификацию для организации универсального сетевого решения для Linux-контейнеров
- Каждый pod кластера должен иметь IP-адрес. Уникальность достигается путем выделения каждому узлу уникальной подсети, из которой затем pod-ам на этом узле назначаются IP-адреса.
- Требует установки специфичного сервиса-контроллера сети
  - Cillium
  - Weave
  - Flannel
  - Calico



Kubernetes: за кадром

- Управление доступом на основе ролей – RBAC
- Affinity/antiaffinity
- API Gateway
- Network Policy
- Admission controllers
- Дистрибутивы – minikube, k3s, kind, microk8s
- Автоматизация развертывания кластера – rancher, ansible kubespray
- Конфигурация приложений as code – helm, helmfile

Live section

Романюта Алексей Андреевич

alexey-r.98@yandex.ru

Кафедра вычислительных систем
Сибирский государственный университет телекоммуникаций и информатики