Введение в Kubernetes

Не забудь включить запись!

План

- Что такое Kubernetes;
- Архитектурные компоненты Kubernetes;
- Основные концепции Kubernetes;
- Методы установки Kubernetes.

Контейнерная оркестрация

Без оркестратора	С оркестратором
Много Docker engine	Один кластер
Искусство планирования	Автопланирование задач
Ручное восстановление сервисов в случае падения ноды	Автоматические перепланирование и восстановление

Kubernetes

- k8s
- Переосмысление проекта **Borg** от Google в контексте контейнеров
- **>65 000** звезд на GitHub
- **>90 000** коммитов
- Более 4 лет в Open Source
- Версия 1.0 была выпущена 21 июля 2015
- Тогда же Kubernetes присоединился к CNCF и стал первым выпускником

Kubernetes НЕ

- Предоставляет возможности “деплоя по кнопке” (не работает с исходным кодом, не собирает приложения);
- Контролирует работу обработчиков данных (БД, etc.);
- Предоставляет инструменты для логирования, мониторинга, трейсинга;
- Обеспечивает управления хостовыми системами;
- Ограничен типами поддерживаемых приложений.

Kubernetes HE

All-inclusive PaaS

Kubernetes HE

OPENSHIFT

LIFECYCLE AUTOMATION (Self-service, CI/CD, Image Stream)	CONTAINER MANAGEMENT (Monitoring, Capacity, Policies)
CONTAINER INFRASTRUCTURE (Orchestration & Scheduling, Storage, Registry, Security, Networking)	

Архитектура Kubernetes

Архитектура Kubernetes

Архитектура Kubernetes

Docker,
CRI-O

Архитектура Kubernetes

Архитектура Kubernetes

kubelet

- Общается с API-Server
- Разворачивает Pods согласно PodSpec (описание Pod в формате YAML или JSON)

Архитектура Kubernetes

kube-proxy

- Проксирует TCP и UDP (не HTTP)
- Используется для работы с сервисами
- Обеспечивает внутреннюю балансировку

Архитектура Kubernetes

Архитектура Kubernetes

kube-apiserver

- API-шлюз - точка входа для всех взаимодействий компонент Kubernetes
- Предоставляет REST-сервис (работа над состояниями объектов)
- Проверяет и конфигурирует API-объекты (pods, services, ...)
- Сохраняет состояние в etcd

Архитектура Kubernetes

Архитектура Kubernetes

Архитектура Kubernetes

kube-controller-manager

- Отслеживает через API изменение состояний объектов
- Набор различных контроллеров:
  - ReplicaSet controller
  - Endpoints controller
  - Namespace controller
  - ...

Архитектура Kubernetes

Архитектура Kubernetes

kube-scheduler

- Учет имеющихся ресурсов
- Учет требований к ресурсам
- Учет ограничений по ресурсам (affinity, anti-affinity)
- Решение где запустить контейнеры (на основе ограничений и требований)

```
while True:
    pods = get_all_pods()
    for pod in pods:
        if pod.node == nil:
            assignNode(pod)
```

Архитектура Kubernetes

Архитектура Kubernetes

Архитектура Kubernetes

Архитектура Kubernetes

Архитектура Kubernetes

Addons

- Расширяют функционал Kubernetes
- Запускаются также, как и остальные сервисы и поды
- Взаимодействуют с Kubernetes через API

Примеры

- kube-dns/CoreDNS
- Сетевые плагины (Weave, Calico, Flannel)
- Dashboard/Weave Scope
- ...

Обзор и список на официальном сайте

Взаимодействие с kube-apiserver

```
# Получить описание всех pod в kube-system namespace
curl https://API_SERVER_ADDRESS/api/v1/namespaces/kube-system/pods/ --header
"Authorization: Bearer $TOKEN" --cacert ca.crt

# Посмотреть логи конкретного pod в kube-system namespace
curl https://API_SERVER_ADDRESS/api/v1/namespaces/kube-system/pods/POD_NAME/log --
header "Authorization: Bearer $TOKEN" --cacert ca.crt
```

Также есть возможность использовать команду kubectl proxy для проброса API на локальную машину.

- Библиотеки для распространенных языков программирования
- Пример для Go

kubectl

- CLI утилита, распространяемая в виде бинарного файла
- Конфигурация для подключения к кластеру

Объекты в кластере можно:

- Создать (**kubectl create**)
- Обновить (**kubectl apply**)
- Получить (**kubectl get**)
- Посмотреть (**kubectl describe**)
- Удалить (**kubectl delete**)
- ...

Обзор kubectl

kubectl

Конфигурация kubectl - это **контекст** (context)

Контекст - комбинация:

- **cluster** - API сервера
- **user** - пользователь для подключения к кластеру
- **namespace** - область видимости (не обязательно, по умолчанию default)

> Информацию о контекстах kubectl сохраняет в файле ~/.kube/config

kubectl | cluster

Кластер (cluster) - это:

- **server** - адрес kubernetes API-сервера
- **certificate-authority** - корневой сертификат (которым подписан SSL-сертификат API-сервера)
- **name** - имя для идентификации

```
clusters: #  ⬅️ Список кластеров
- cluster:
    certificate-authority: ca.crt
    server: https://35.198.140.134
  name: kube-cluster
```

kubectl | user

Пользователь (**user**) - это:

- Данные для аутентификации:
  - username + password (Basic Auth)
  - client key + client certificate
  - token
  - auth-provider config (например GCP)
- name (имя) для идентификации в конфиге

```
users: # ⬅️ Список пользователей и способов их авторизации
- name: kube-user
  user:
    client-certificate: kube-user.crt
    client-key: kube-user.key
```

kubectl | context

Контекст (**context**) - это:

- **cluster** - имя кластера из списка clusters
- **user** - имя пользователя из списка users
- **namespace** - область видимости по-умолчанию (не обязательно)
- **name** - имя контекста для идентификации

```yaml
contexts:  # ⬅️ Список контекстов
- context:
    cluster: kube-cluster
    user: kube-user
  name: kube-context
```

Пример конфигурации kubectl

```yaml
apiVersion: v1
clusters:                       # ⬅️ Список кластеров
- cluster:
    certificate-authority: ca.crt
    server: https://35.198.140.134
  name: kube-cluster
contexts:                       # ⬅️ Список контекстов
- context:
    cluster: kube-cluster
    user: kube-user
  name: kube-context
current-context: kube-context # ⬅️ Текущий контекст
kind: Config
preferences: {}
users:                          # ⬅️ Список пользователей и способов их авторизации
- name: kube-user
  user:
    client-certificate: user.crt
    client-key: user.key
```

Порядок конфигурации kubectl

Обычно порядок конфигурирования `kubectl` следующий:

```
# Указать кластер:
kubectl config set-cluster … cluster_name

# Указать данные пользователя (credentials):
kubectl config set-credentials … user_name

# Создать контекст:
kubectl config set-context context_name --cluster=cluster_name --user=user_name

# Использовать контекст:
kubectl config use-context context_name
```

Команды kubectl

kubectl Cheat Sheet

```
# Создать ресурс из манифеста
kubectl create -f manifest.yml
kubectl apply -f manifest.yml
kubectl apply -f link
kubectl apply -f directory/

# Получить список ресурсов
kubectl get pods

# Получить описание ресурса
kubectl describe pod POD_NAME
```

Еще один cheatsheet

Основные концепции Kubernetes

Pods

- Группа контейнеров (один или несколько)
- Минимальная сущность, управляемая Kubernetes

У всех контейнеров общие Network, IPC, UTS, PID* namespaces

Pods

Контейнеры внутри одного Pod или разные Pod?

- Сервисы должны масштабироваться совместно или по отдельности?
- Должны ли сервисы быть запущены вместе или могут быть разнесены на разные хосты?
- Это связанные сервисы или независимые компоненты?

Pods

Описание паттернов

Процесс запуска Pod`ов

Описание объектов

YAML

- 1 файл может описывать много сущностей
- Обязательные поля:
  - apiVersion
  - kind
  - metadata
- Спецификация объекта

Описание объектов

prometheus-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: prometheus
spec:
  containers:
  - name: prometheus
    image: prom/prometheus:v2.10.0
```

Описание объектов

kind

Тип объекта, который хотим создать

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: prometheus
spec:
  containers:
  - name: prometheus
    image: prom/prometheus:v2.10.0
```

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: monitoring
```

Описание объектов

apiVersion

Путь на используемую группу API для создания объекта

```
apiVersion: $GROUP_NAME/$VERSION
```

3 стадии развития:

- **alpha** - стадия тестирования, функционал может быть удален из последующих версий (v1alpha1)
- **beta** - безопасно для использования, но функционал может меняться (v1beta1)
- **stable** - стабильно (v1)

Документация для версии 1.18

Описание объектов

apiVersion

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: prometheus
spec:
  containers:
  - name: prometheus
    image: prom/prometheus:v2.10.0
```

```yaml
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: cronjob
spec:
-- omitted --
```

Описание объектов

metadata

- Имя создаваемого объекта
- Метки (labels)
- Namespace
- Аннотации

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: prometheus
  labels:
    app: prometheus
  namespace: monitoring
spec:
-- omitted --
```

Reddit App в Kubernetes

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: post
spec:
  containers:
  - image: avtandilko/post
    name: post
```

Resources | Requests

Ресурсы, необходимые для работы контейнера

- Суммарное значение requests всех pod на хосте не может быть большим, чем количество физических ресурсов хоста
- При определении на какой хост поместить тот или иной pod, kube-scheduler будет учитывать значения requests
- Если на хосте недостаточно физических ресурсов для запуска pod, то kube-scheduler гарантированно не выберет данный хост

Resources | Limits

Пороговые значения ресурсов, которые может потреблять контейнер

- При превышении данных значений контейнер будет:
  - Перезапущен в случае превышения по RAM
  - Ограничен в потреблении в случае превышения по CPU
- Суммарное значение limits всех pod на хосте может быть большим, чем количество физических ресурсов хоста (overcommit)

> Указание requests и limits для контейнеров - рекомендуемая практика работы с Kubernetes

Resources | Пример

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.17
    resources:
      requests:
        memory: "64Mi"
        cpu: "100m"
      limits:
        memory: "128Mi"
        cpu: "200m"
```

ReplicaSet

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: post-rs
spec:
  replicas: 2
  selector:
    matchLabels:
      app: post
  template:
    metadata:
      name: post
      labels:
        app: post
    spec:
      containers:
      - image: avtandilko/post
        name: post
```

Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: post-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: post
  template:
    metadata:
      name: post
      labels:
        app: post
    spec:
      containers:
      - image: avtandilko/post
        name: post
```

Deployment

DEPLOYMENT
REPLICA SET
V1
REPLICA SET
V2
REPLICA SET
V3
POD
POD
POD
POD
POD
POD
NODE
NODE
NODE

Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: post
  labels:
    app: post
spec:
  ports:
  - port: 5000
    protocol: TCP
    targetPort: post
  selector:
    app: post
```

Ingress

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ui
spec:
  rules:
  - http:
      paths:
      - path: /ui
        backend:
          serviceName: ui
          servicePort: 9292
```

Networking

POST_POD
POST_POD
post_
svc
UI_POD
UI_POD
ui_
svc
COMMENT_
POD
COMMENT_
POD
comm
_svc
ingress

Еще немного абстракций Kubernetes

- DaemonSet
- StatefulSet
- Job/CronJob

Методы установки Kubernetes

Методы установки компонентов k8s | как это запускается

- Распространенный подход запускать основные компоненты (api-server, kube-proxy...) в **static pods** при помощи kubelet
  - Так делает kubeadm (инструмент для установки k8s)
  - Сам kubelet запускается как systemd service
  - Можно легко посмотреть статусы сервисов через kubectl
  - Можно пощупать в minikube
- Также можно запусать все основные компоненты как systemd service

Методы установки Kubernetes

- Руками (The Hard Way)
- Minikube
- kind
- Kubernetes in Docker for Mac/Windows
- kops
- Kubespray
- kubeadm
- Google Kubernetes Engine
- Amazon EKS
- Mail.Ru Cloud Solutions
- OpenShift

Minikube

- Для локальной разработки
- Создает кластер из одной ноды

Требования для различных ОС:

- macOS (xhyve driver, VirtualBox, или VMware Fusion)
- Linux (VirtualBox или KVM, Docker)
- Windows (VirtualBox или Hyper-V)

Kubernetes in Docker for Mac/Windows

- Попытка заменить Minikube, вместе с Docker предоставляется еще установка Kubernetes
- На одном docker-engine хосте можно запускать как Swarm так и Kubernetes используя linuxkit на Mac/Windows
- Установка в одну кнопку

Kubeadm

- Официальная утилита на Go;
- Только устанавливает и настраивает Kubernetes;
- Не устанавливает Docker или любую другую Container Runtime систему;
- Не устанавливает Overlay-сети;

Kubespray

- Ansible-репозиторий;
- Поддерживается сообществом;
- Можнопоставить куда угодно (AWS, GCE, Azure, OpenStack или bare metal);
- Прозрачность установки*;
- Адаптация под множество ОС (Ubuntu, Debian, Centos);
- Кастомизация установки;
- Использует Kubeadm.

Kops

- Утилита на Go;
- End-to-end управление кластером;
- Настройка автомасштабирования кластера (AWS);
- Создание машин в AWS, GCP;
- Развертывание k8s, балансировщиков;
- Экспорт инфраструктуры Terraform и CloudFormation.

Ссылки

- https://t.me/kubernetes_ru - русское сообщество Kubernetes в Telegram;
- https://github.com/kelseyhightower/kubernetes-the-hard-way - курс от Kelsey Hightower;
- https://www.katacoda.com/courses/kubernetes - Learn Kubernetes using Interactive Browser-Based Scenarios;
- https://kubernetes.io/docs/reference/kubectl/cheatsheet/

Литература