Введение в Kubernetes

Не забудь включить запись!

План

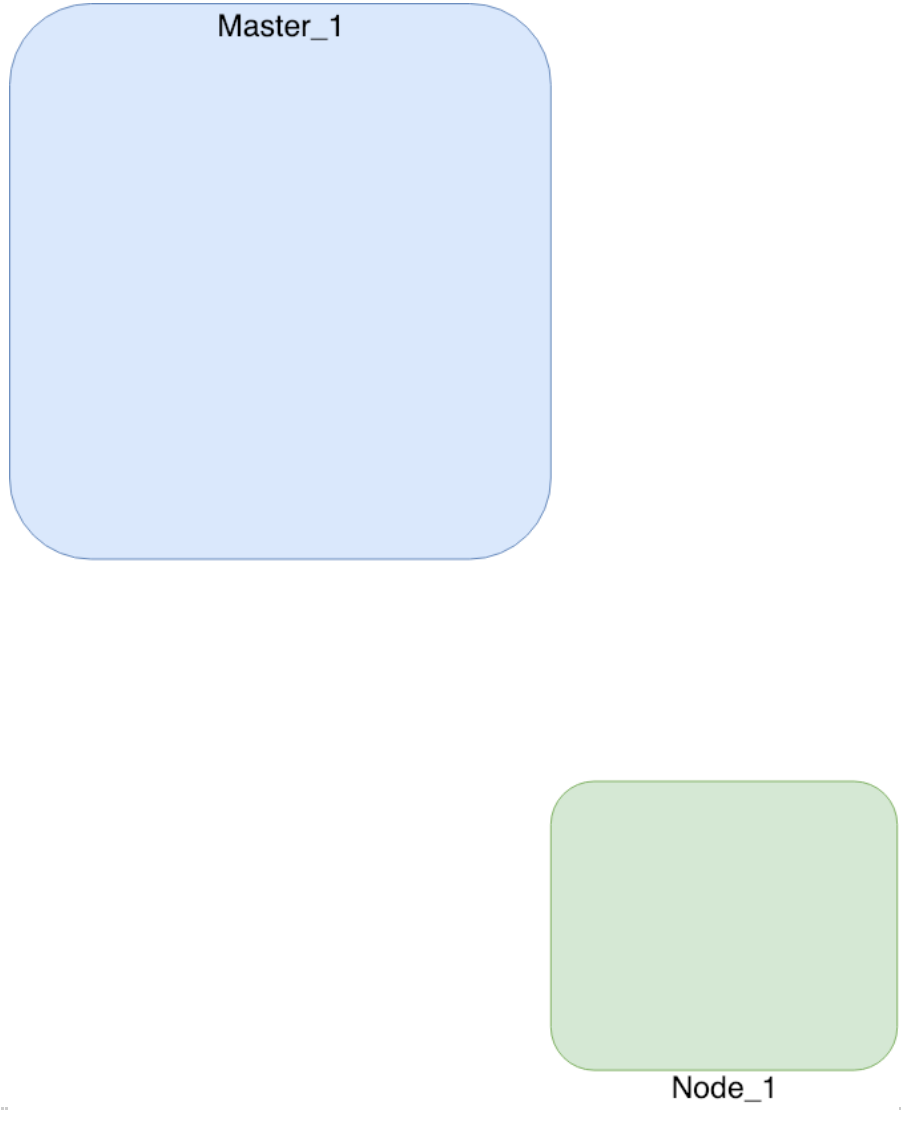
- Архитектура Kubernetes, основные компоненты
- Концепция pod
- YAML
- Манифесты Kubernetes
- Подключение к API, kubectl
- Локальное окружение
- История развития Kubernetes

Kubernetes

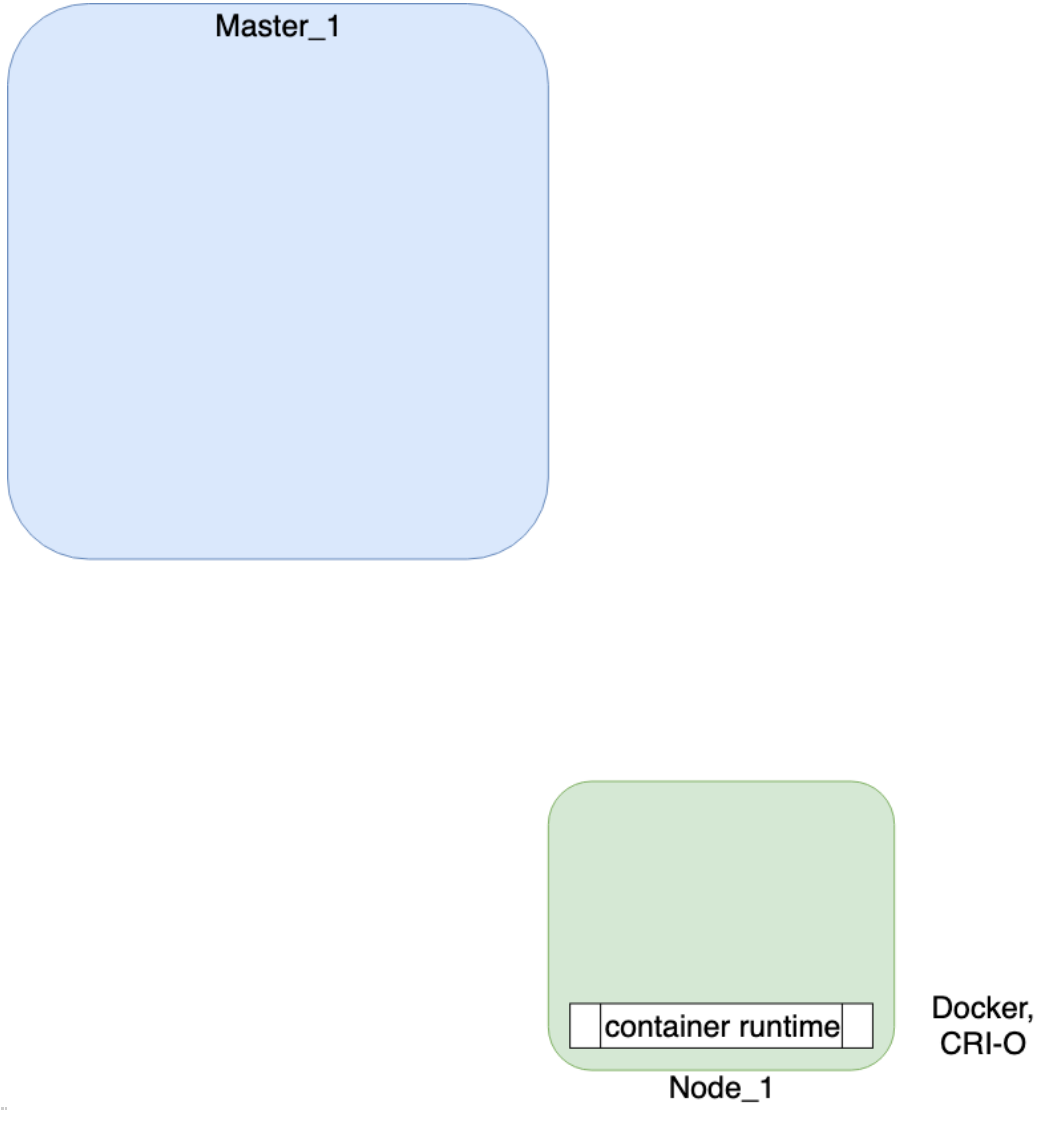
- k8s
- Переосмысление проекта **Borg** от Google в контексте контейнеров
- **~55 000** звезд на GitHub
- **~80 000** коммитов
- Более 3 лет в Open Source
- Версия 1.0 была выпущена 21 июля 2015
- Тогда же Kubernetes присоединился к CNCF и стал первым выпускником

Архитектура Kubernetes

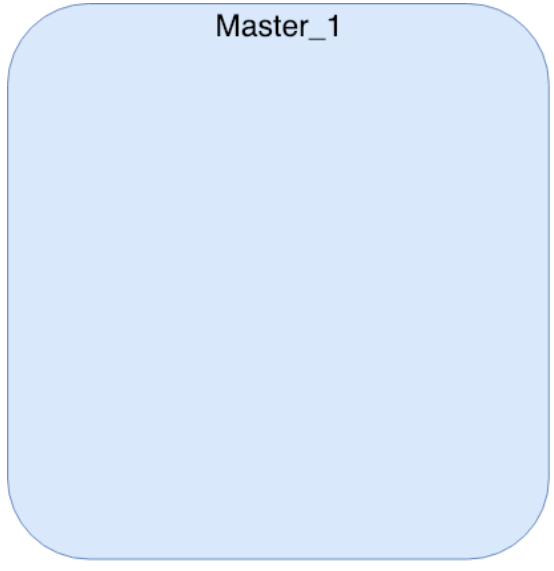
Архитектура Kubernetes

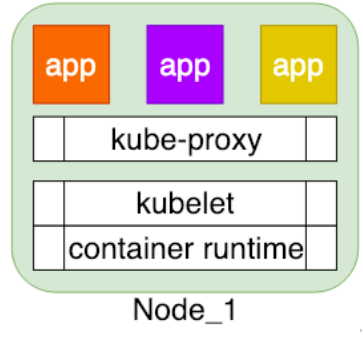


Архитектура Kubernetes



Архитектура Kubernetes





Архитектура Kubernetes

kubelet

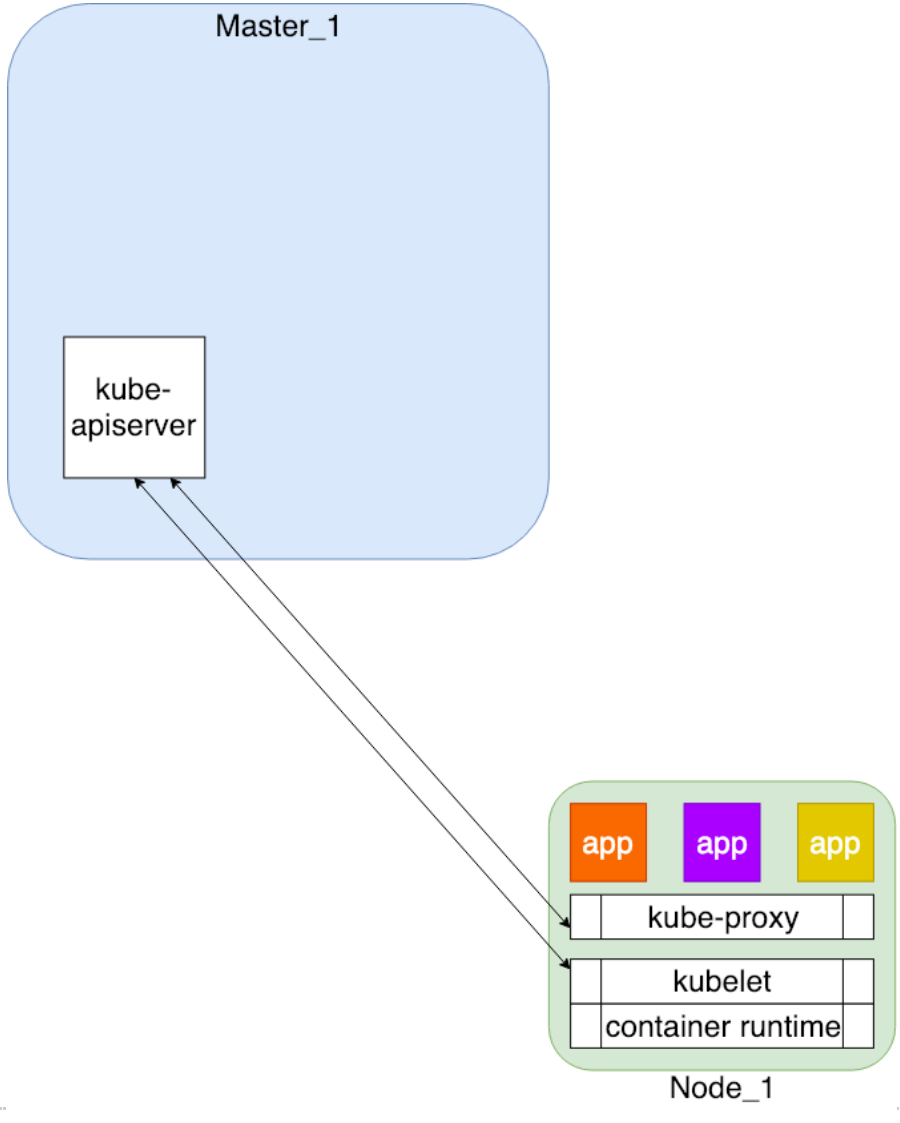
- Общается с API-Server
- Разворачивает Pods согласно PodSpec (описание Pod в формате YAML или JSON)

Архитектура Kubernetes

kube-proxy

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

Архитектура Kubernetes

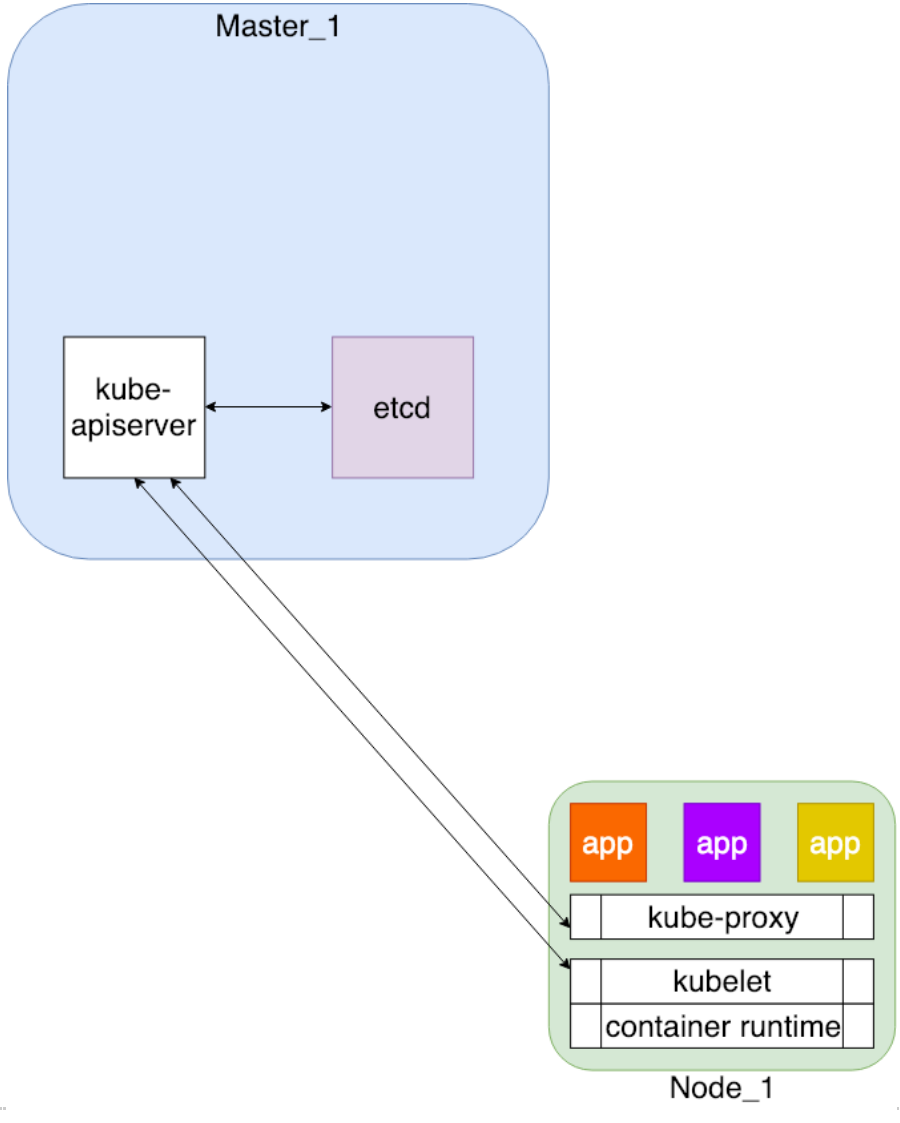


Архитектура Kubernetes

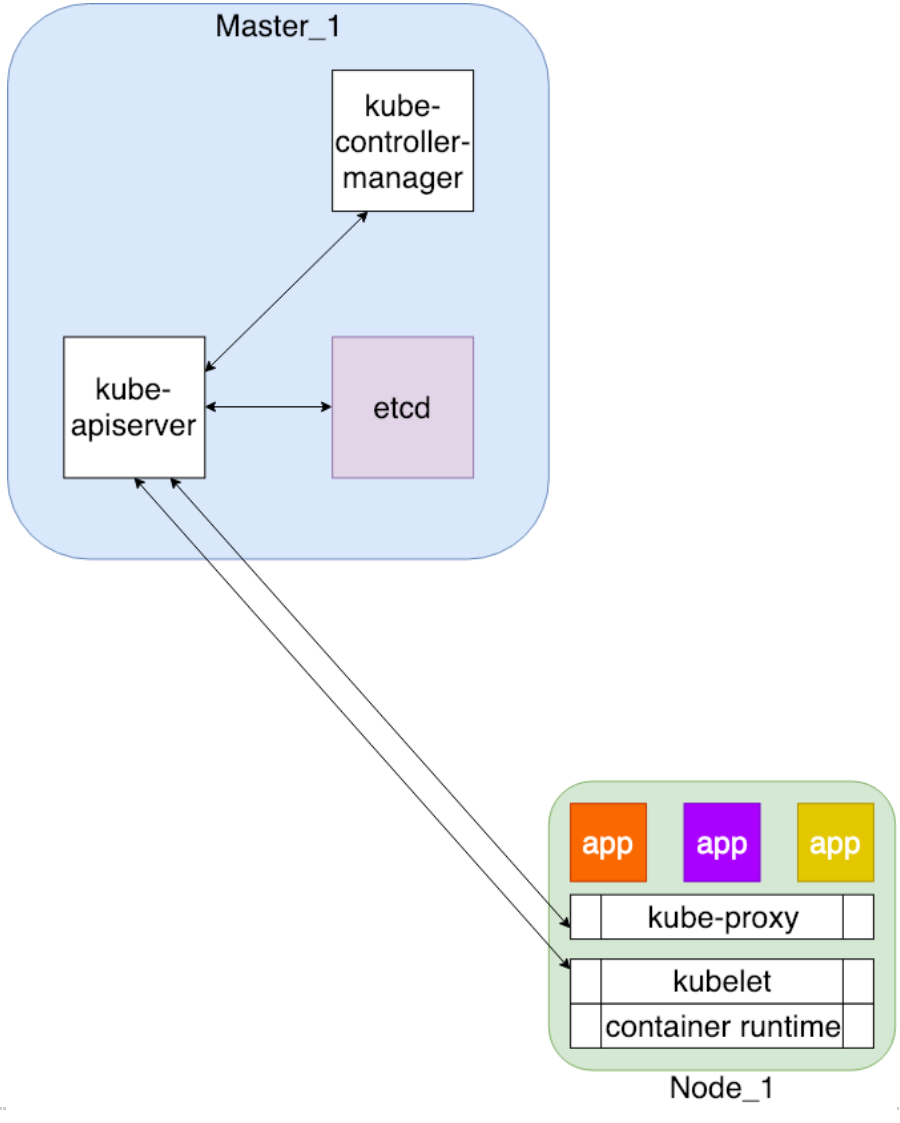
kube-apiserver

- 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

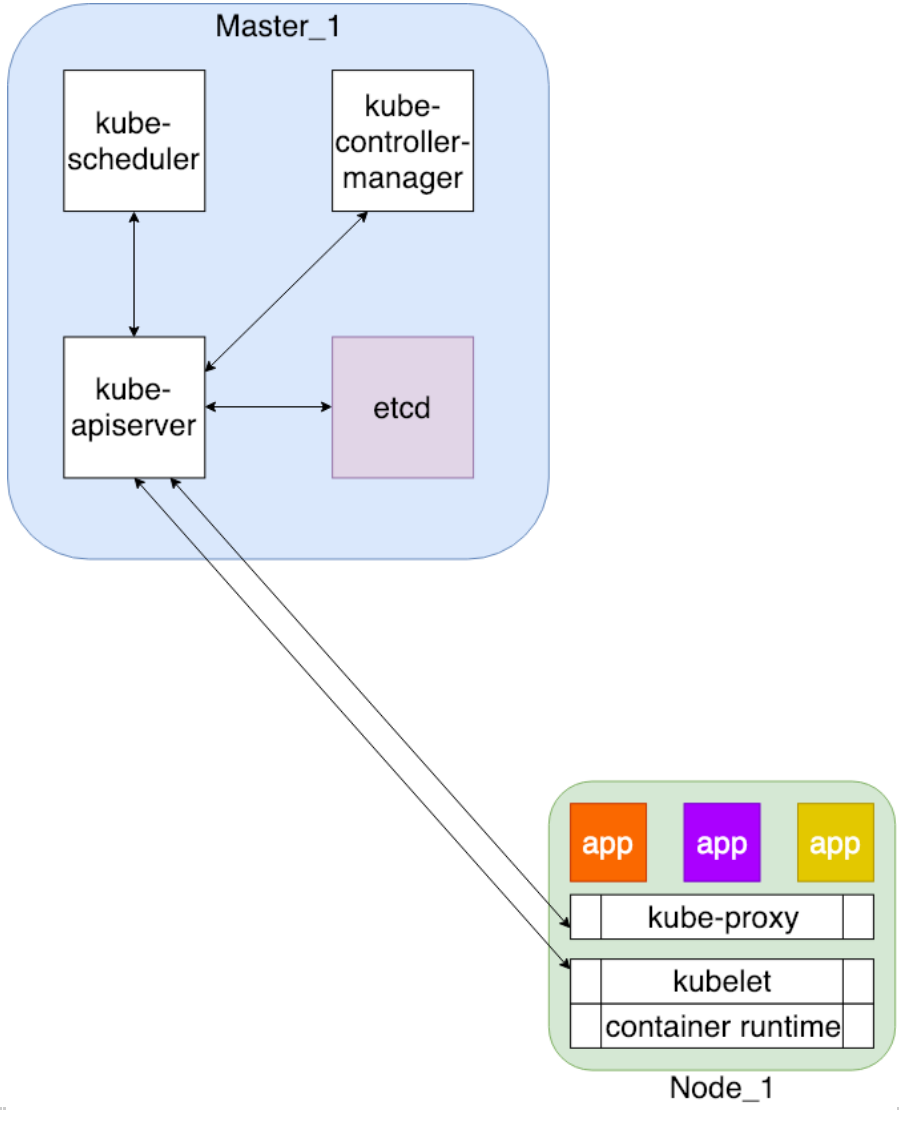


Архитектура Kubernetes

kube-controller-manager

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

Архитектура Kubernetes



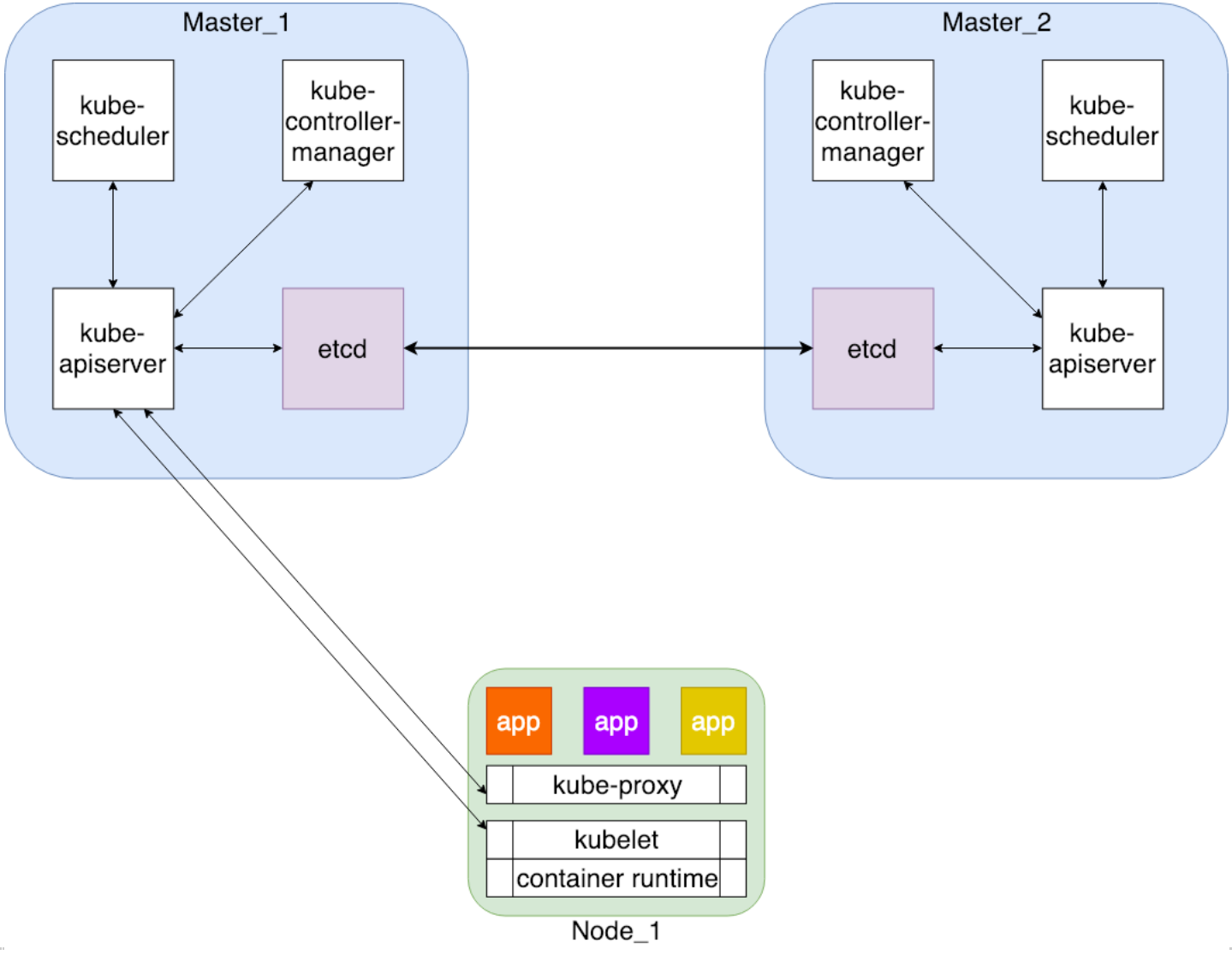
Архитектура Kubernetes

kube-scheduler

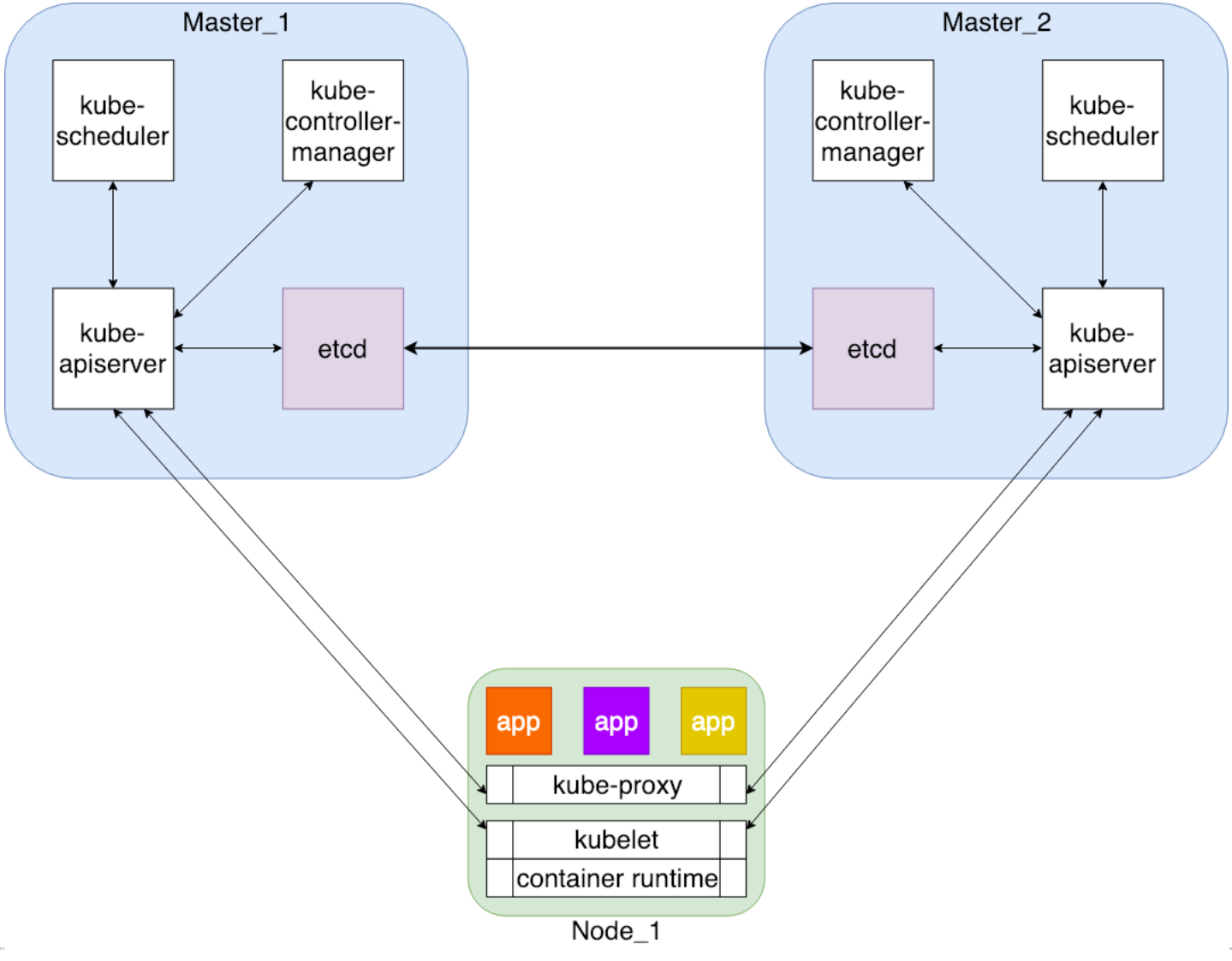
- Учет имеющихся ресурсов
- Учет требований к ресурсам
- Учет ограничений по ресурсам (affinity, anti-affinity)
- Решение где запустить контейнеры (на основе ограничений и требований)

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

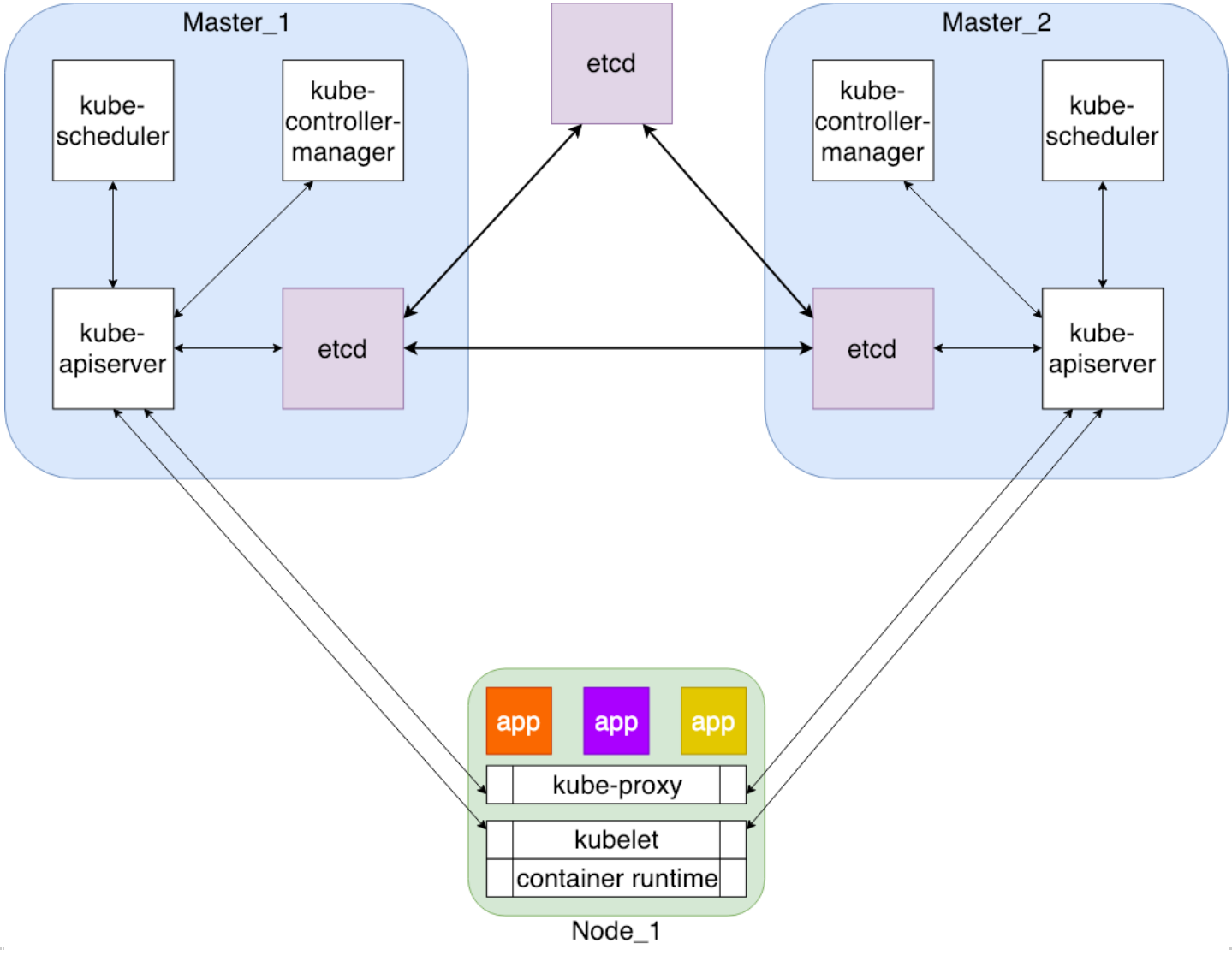
Архитектура Kubernetes



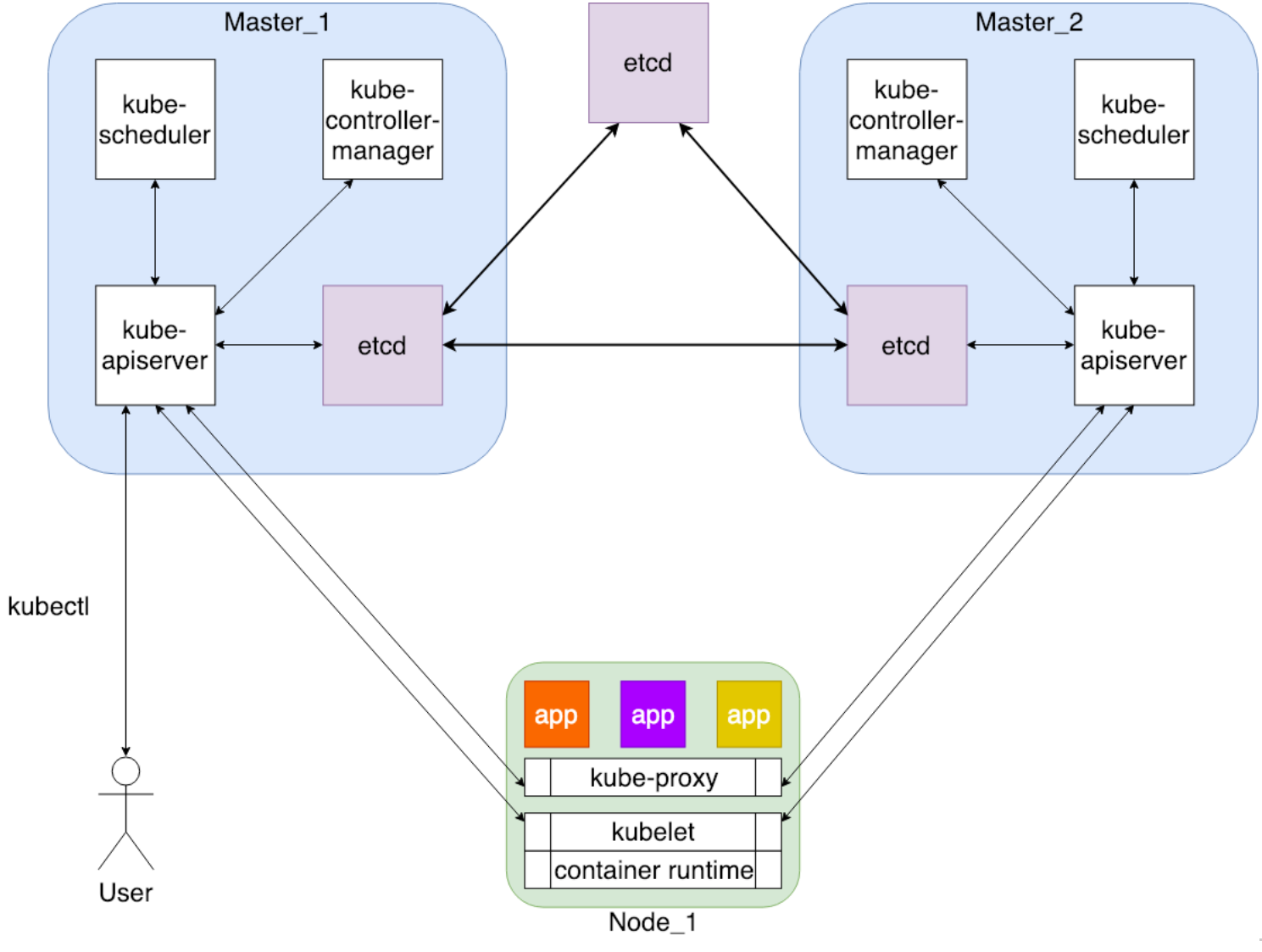
Архитектура Kubernetes



Архитектура Kubernetes



Архитектура Kubernetes



Архитектура Kubernetes

Addons

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

Примеры

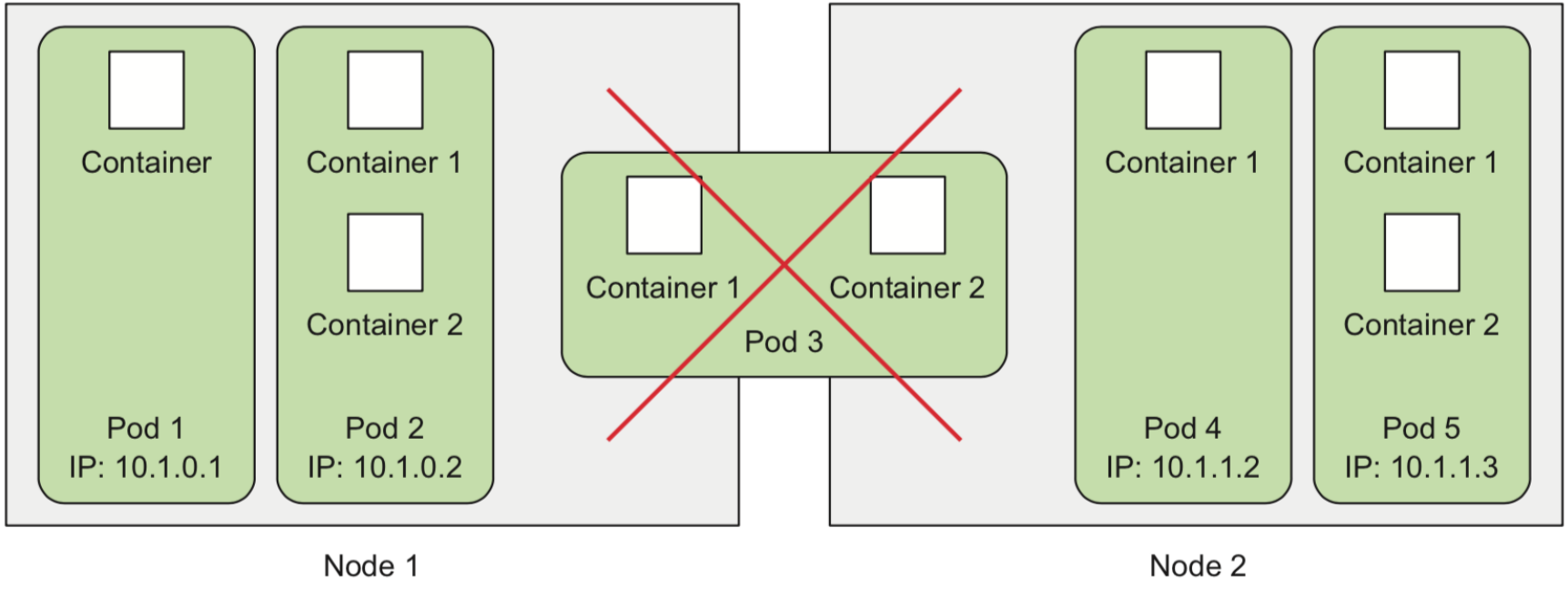
- kube-dns/CoreDNS
- Сетевые плагины (Weave, Calico, Flannel)
- Dashboard/Weave Scope
- ...

Обзор и список на официальном сайте

Pods

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

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

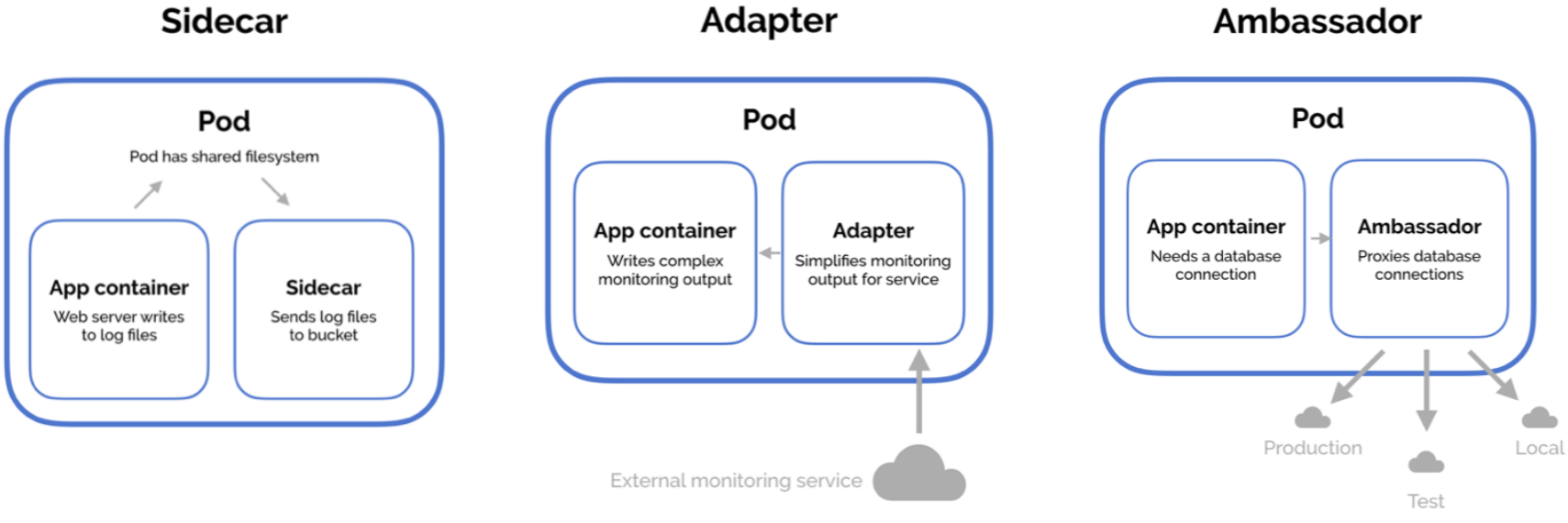


Pods

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]

Init контейнеры

Похожи на обычные контейнеры внутри pod, но есть отличия:

- Запускаются до старта остальных контейнеров (app) в pod
- Блокируют запуск app контейнеров

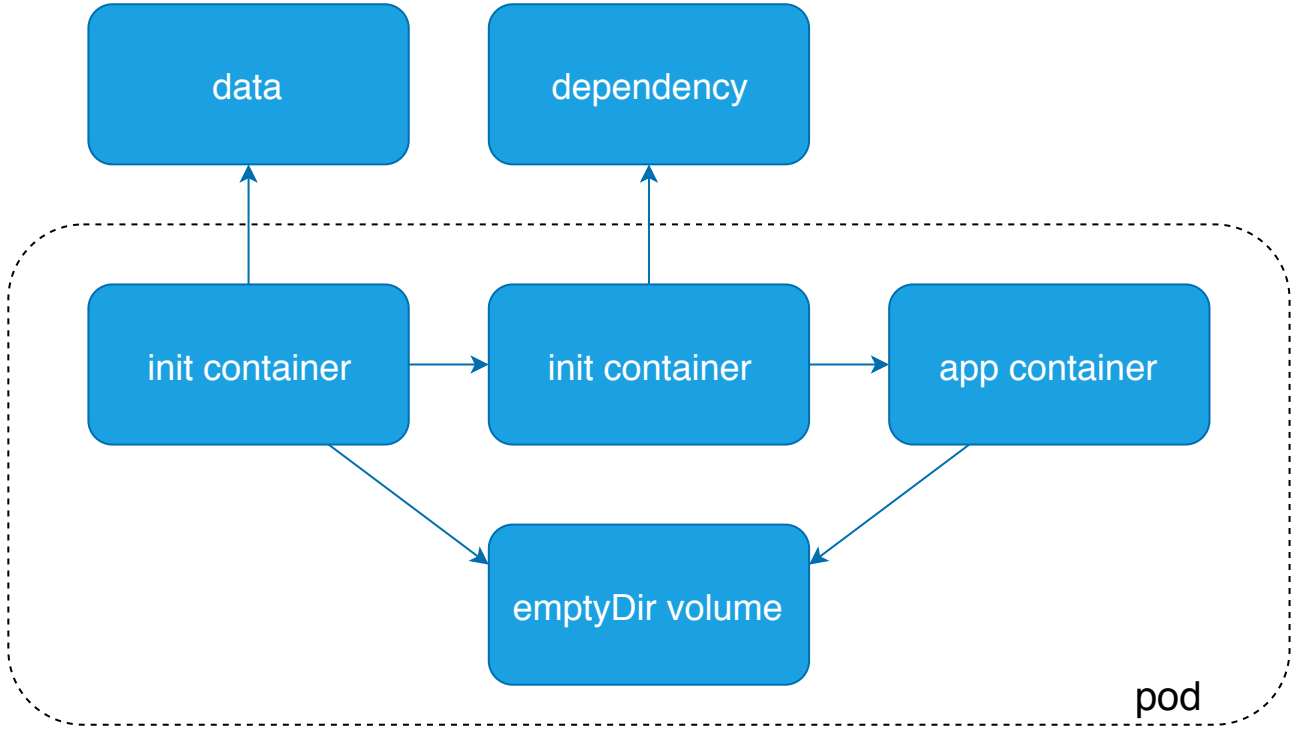
Init контейнеры

Используются для:

- Проверки внешних зависимостей необходимых для старта app контейнеров
- Подготовки данных для app контейнеров (например `git clone`)
- Использования данных, которые нежелательно хранить в app контейнере

Init контейнеры

Механизм работы



YAML Ain’t Markup Language

- Просто читать и редактировать
- Отступы для уровней вложенности (только пробелы)
- Массивы и словари

YAML vs JSON

JSON

```
{
   "apiVersion": "v1",
   "kind": "Pod"
}
```

YAML

```
apiVersion: v1
kind: Pod
```

🤔 Любой валидный JSON - это валидный YAML

YAML vs JSON

JSON	YAML
Строки обязательно экранированы	Строки можно не экранировать
`"var": true` - это точно boolean	`var: true` - это boolean, но это неточно
Удобнее парсить машиной	Удобнее парсить человеком
Каждый документ при потоковой обработке в своих `{ .. }`	Тут скобок нет, но есть `---` и `...`

YAML | Списки и словари

Списки задаются так:

```
numbers:
  - one
  - two
  - three
```

или так:

```
numbers: [ one, two, three ]
```

YAML | Списки и словари

Словари задаются так:

```
numbers:
  first: one
  second: two
  third: three
```

или так:

```
numbers: { first: one, second: two, third: three }
```

И даже так (но лучше не надо):

```
? - I am a list and
  - I'm a key for a map
: [ Pretty, Cool, I, think ]
```

YAML | Multiline strings

Сохраняем переносы строки:

```
include_newlines: |
            exactly as you see
            will appear these three
            lines of poetry
```

Игнорируем переносы строки:

```
fold_newlines: >
            this is really a
            single line of text
            despite appearances
```

В коде чаще приходится использовать `|+` или `|-`, чтобы в конце сохранялся перенос строки.

YAML | Ссылки

- Multiline demo
- YAML lint

YAML не так прост, как хотелось бы:

- Learn YAML in Y minutes
- YAML Ain’t Markup Language (YAML™) Version 1.2

Описание объектов

YAML

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

Описание объектов

prometheus-pod.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

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

```
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 1.15

Описание объектов

apiVersion

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

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

Описание объектов

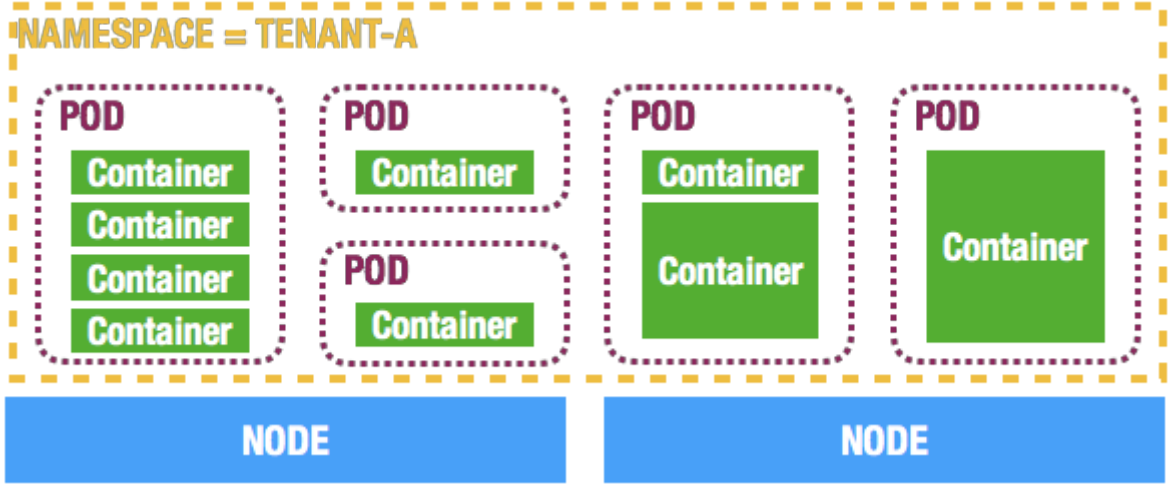
metadata

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

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

Namespaces

- Можно создать несколько виртуальных кластеров в рамках одного физического кластера
- Namespace - один виртуальный кластер (одно окружение)



Namespaces

Часто применяются для:

- Разграничения прав между командами
- Лимитирования ресурсов на проект с помощью квот (cpu, memory)

Namespaces

По умолчанию в кластере создается три namespace:

- **default** - для объектов у которых явно не определена принадлежность к другому namespace
- **kube-system** - для системных объектов Kubernetes
- **kube-public** - для объектов к которым нужен доступ из любой точки кластера

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

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

Пример конфигурации kubectl

```
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

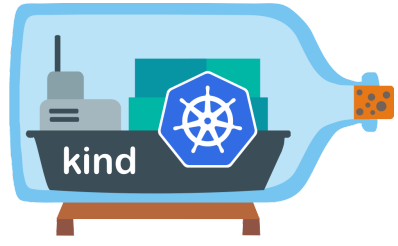
Локальное окружение

Certified Kubernetes - Installer

Локальное окружение

kind

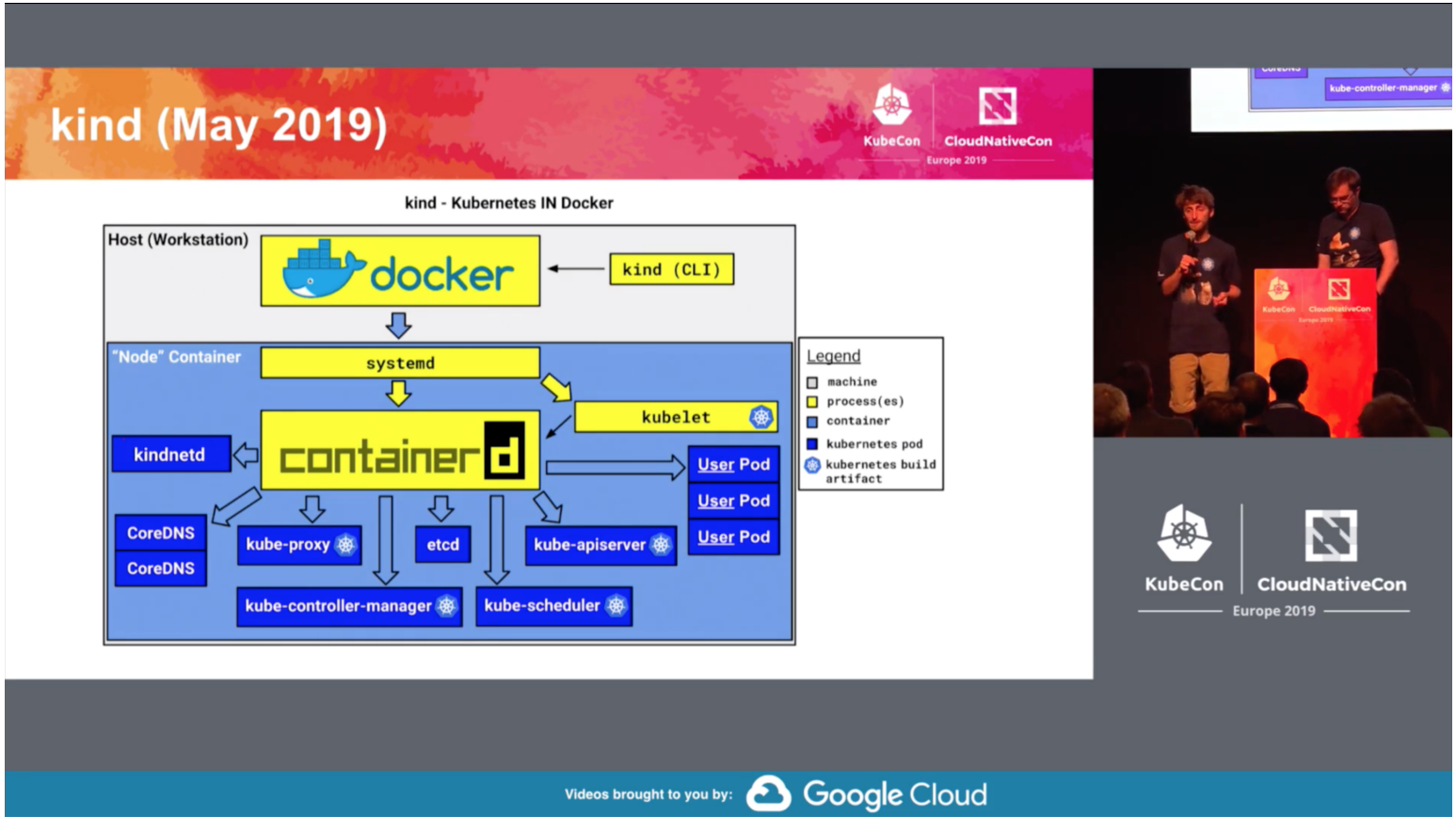
- Разворачивает кластер Kubernetes в Docker
- Поддержка multi-node workers/control-planes
- 3 worker + 3 master = 1 CPU + 2Gb RAM
- Поддержка Windows/MacOS/Linux
- Релизный цикл близок к релизному циклу Kubernetes




```
kind: Cluster
apiVersion: kind.sigs.k8s.io/v1alpha3
nodes:
- role: control-plane
- role: worker
- role: worker
```

Локальное окружение

kind



Локальное окружение

minikube

- Разворачивает кластер Kubernetes в ВМ на локальной машине
- All-in-one кластер (все компоненты на одной ВМ)
- Поддержка Windows/MacOS/Linux
- Набор аддонов, которые можно включить при установке (dashboard, efk, etc.)
- Релизный цикл близок к релизному циклу Kubernetes

minikube

Локальное окружение

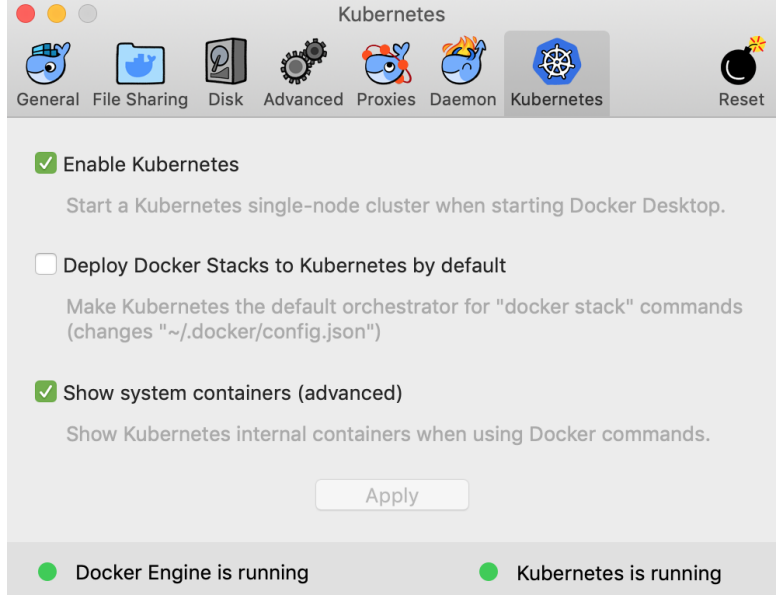
MicroK8s

- snap пакет (Ubuntu, Debian, Fedora)
- Полная настройка окружения одной командой (`snap install microk8s --classic`)
- Запуск управляющих компонент Kubernetes как systemd сервисов
- Релизный цикл близок к релизному циклу Kubernetes

Локальное окружение

Docker Desktop

- Поддержка MacOS/Windows
- Релизный цикл сильно отстает от релизного цикла Kubernetes



Kubernetes 1.1 (Ноябрь 2015)

Анонс

- Horizontal pod autoscaling (Beta)
- HTTP load balancer (Beta)
- Jobs (Beta)
- Улучшение механизма Rolling Update (проверка состояния обновленных pod)

Kubernetes 1.2 (Март 2016)

Анонс

- Улучшение масштабируемости (1000 нод, 30000 контейнеров, 400%)
- ConfigMaps
- DaemonSets (Beta)
- Поддержка TLS в Ingress (Beta)
- Нестандартные метрики для автомасштабирования

- Репозиторий с Changelog для версий 1.2+
- Блог с Release Notes и другой полезной информацией
- Исходники блога с Release Notes

Kubernetes 1.3 (Июль 2016)

Анонс

- PetSets (сейчас StatefulSets, Alpha)
- Minikube
- CNI
- Federations (Beta)
- Init контейнеры (Alpha)

Kubernetes 1.4 (Сентябрь 2016)

Анонс

- Kubeadm
- ScheduledJobs (сейчас cronJobs, Alpha)
- PodSecurityPolicy
- Inter-pod affinity and anti-affinity (Alpha)

Kubernetes 1.5 (Декабрь 2016)

Анонс

- CRI
- StatefulSet (Beta)
- PodDisruptionBudget (Beta)
- Windows Server containers (Alpha)

Kubernetes 1.6 (Март 2017)

Анонс

- Улучшение масштабируемости (5000 нод, 150000 pod)
- RBAC по умолчанию (Beta)
- Dynamic Storage Provisioning GA
- etcd3

Kubernetes 1.7 (Июнь 2017)

Анонс

- Network policy GA
- Шифрование etcd (Alpha)
- Local storage (Alpha)
- API aggregation
- CRD

Kubernetes 1.8 (Сентябрь 2017)

Анонс

- RBAC GA
- IPVS в kube-proxy (Alpha)
- metrics-server (Beta)

Kubernetes 1.9 (Декабрь 2017)

Анонс

- Windows support (Beta)
- CSI (Alpha)
- CoreDNS (Alpha)
- `kubectl diff` (Alpha)

Kubernetes 1.10 (Март 2018)

Анонс

- CSI (Beta)
- Local Storage (Beta)
- Windows Containers CRI (Alpha)
- Shared PID namespace (Alpha)

Kubernetes 1.11 (Июнь 2018)

Анонс

- IPVS GA
- CoreDNS GA
- Online PV resizing (без рестарта pod, Alpha)
- kubelet dynamic configuration (Beta)

The History of Kubernetes on a Timeline

Kubernetes 1.12 (Сентябрь 2018)

Анонс

- RuntimeClass (Alpha)
- SCTP (Alpha)
- Vertical scaling of Pods (Beta)
- Улучшение алгоритма Scheduler

Kubernetes 1.13 (Декабрь 2018)

Анонс

- CSI GA
- kubeadm GA
- CoreDNS по умолчанию
- `kubectl diff` && `kubectl apply --server-dry-run` (Beta)
  - До: `kubectl apply --dry-run` исполняемый локально (базовая валидация без Admission Controllers)
  - Сейчас: запросы kubectl могут быть помечены как тестовые и пройти все стадии инициализации, кроме создания Pod
  - Подробности

Kubernetes 1.14 (Март 2019)

Анонс

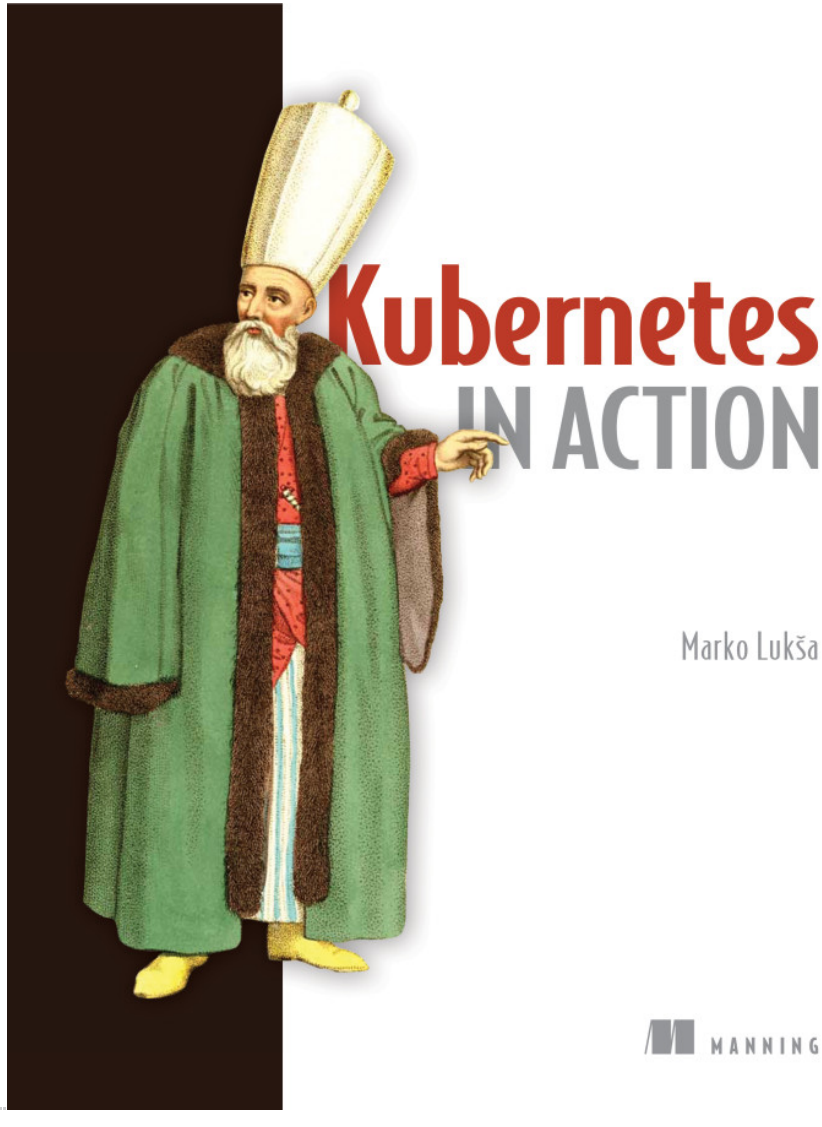
- PID limits per-pod (Beta)
- Pod preemption and priority GA
- Windows nodes GA (а также Windows Server 2019)
- kubectl kustomize support
- Local storage GA
- kubeadm HA clusters (Alpha)

Kubernetes 1.15 (Июнь 2019)

Анонс

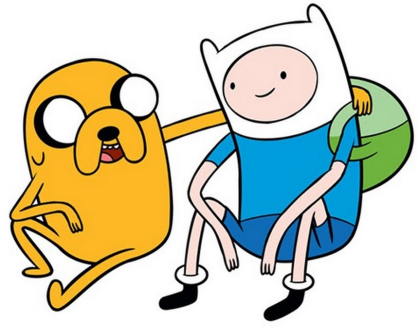
- Volume Cloning (Alpha)
- Non-preempting PriorityClasses (Alpha)
- kubeadm HA clusters (Beta)
- Scheduling Framework (Alpha)
- kubeadm certificates management (Alpha)

Kubernetes in Action



Katacoda

Learn Kubernetes using Interactive Browser-Based Scenarios



Спасибо за внимание!

Время для ваших вопросов!