Введение в Kubernetes

Не забудь включить запись!

План

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

Kubernetes это...

- Переосмысление проекта **Borg** от Google в контексте контейнеров;
- k8s;
- **~50 000** звезд на GitHub (это только сам Kubernetes);
- **~75 000** 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.

Контейнерная оркестрация

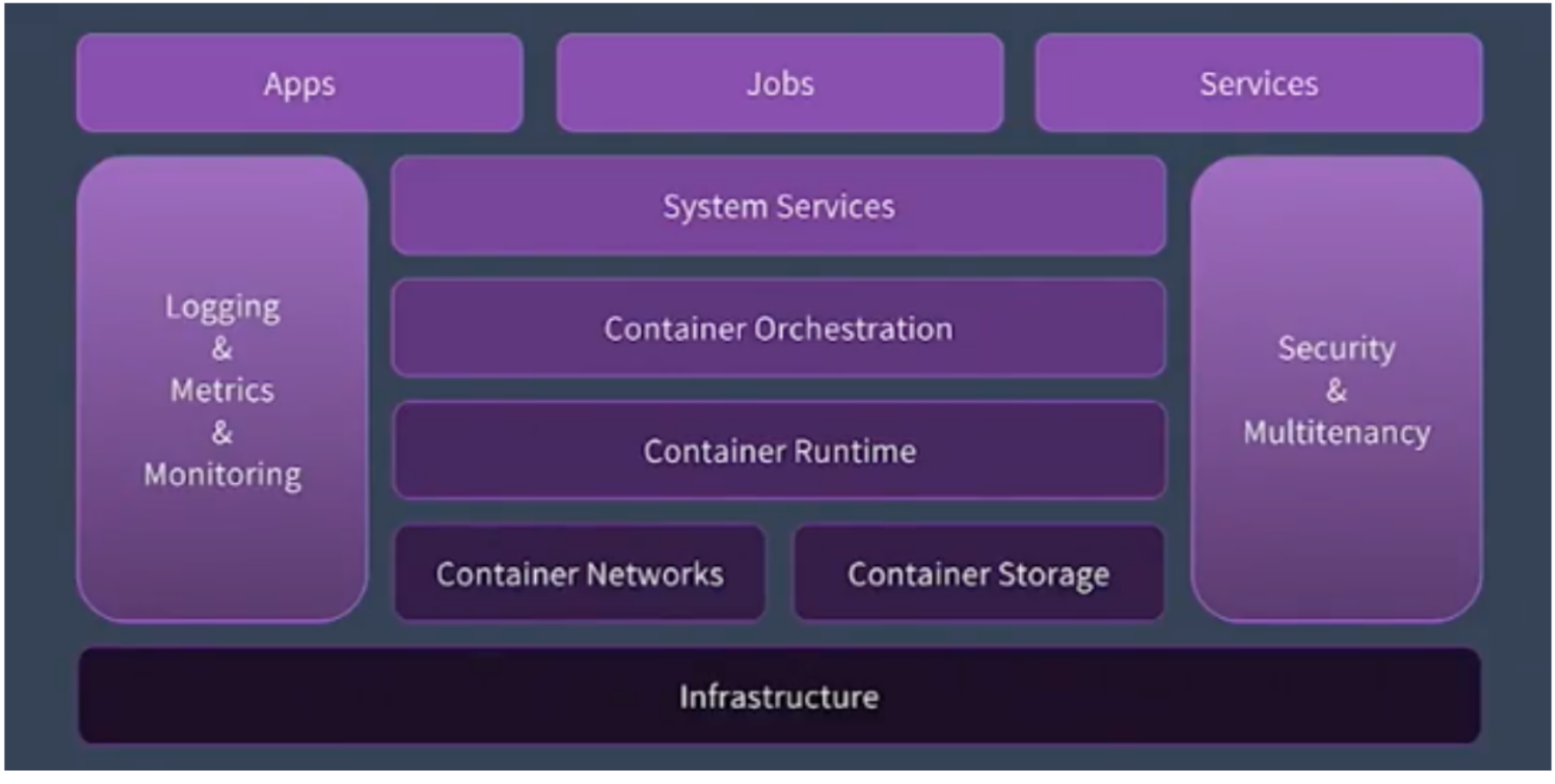
Без оркестратора	С оркестратором
Много Docker engine	Один кластер
Искусство планирования	Автопланирование задач
Ручное восстановление сервисов в случае падения ноды	Автоматические перепланирование и восстановление

Kubernetes 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



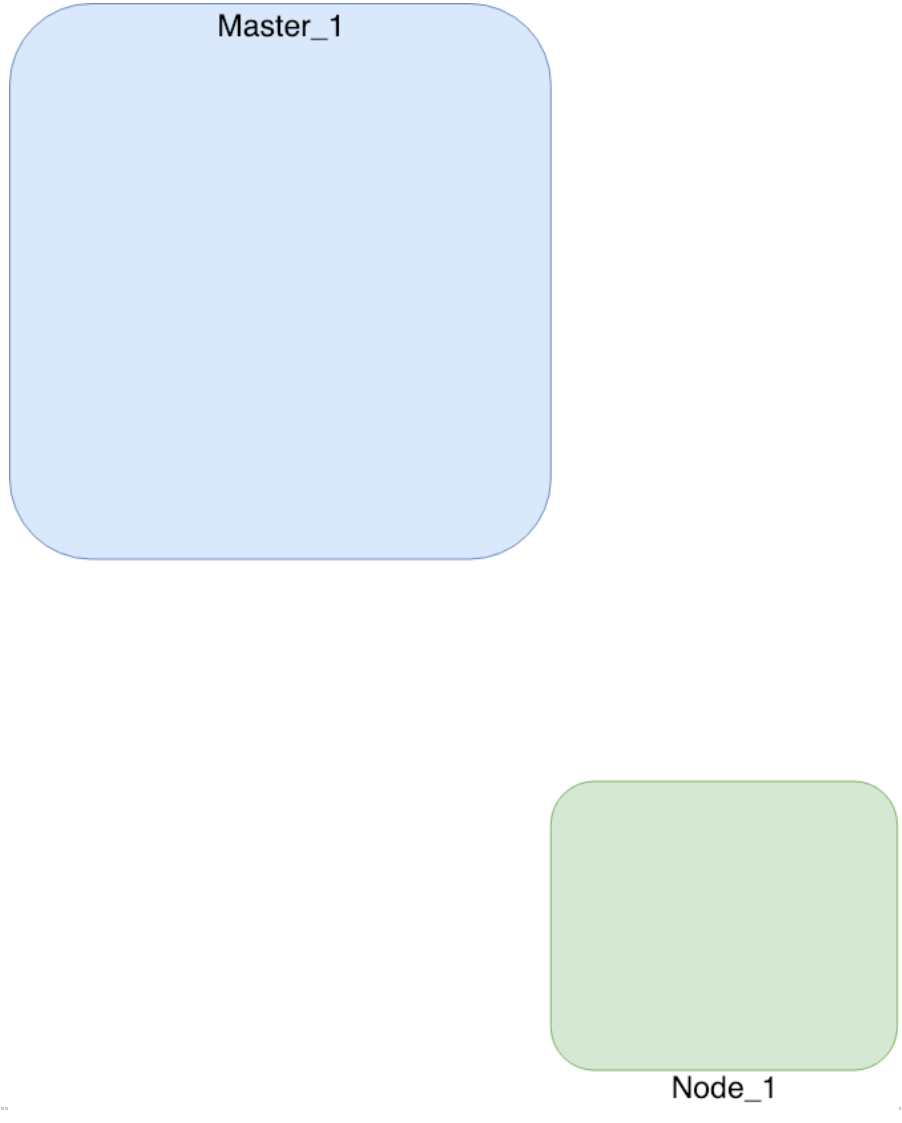
Kubernetes HE

OPENSHIFT

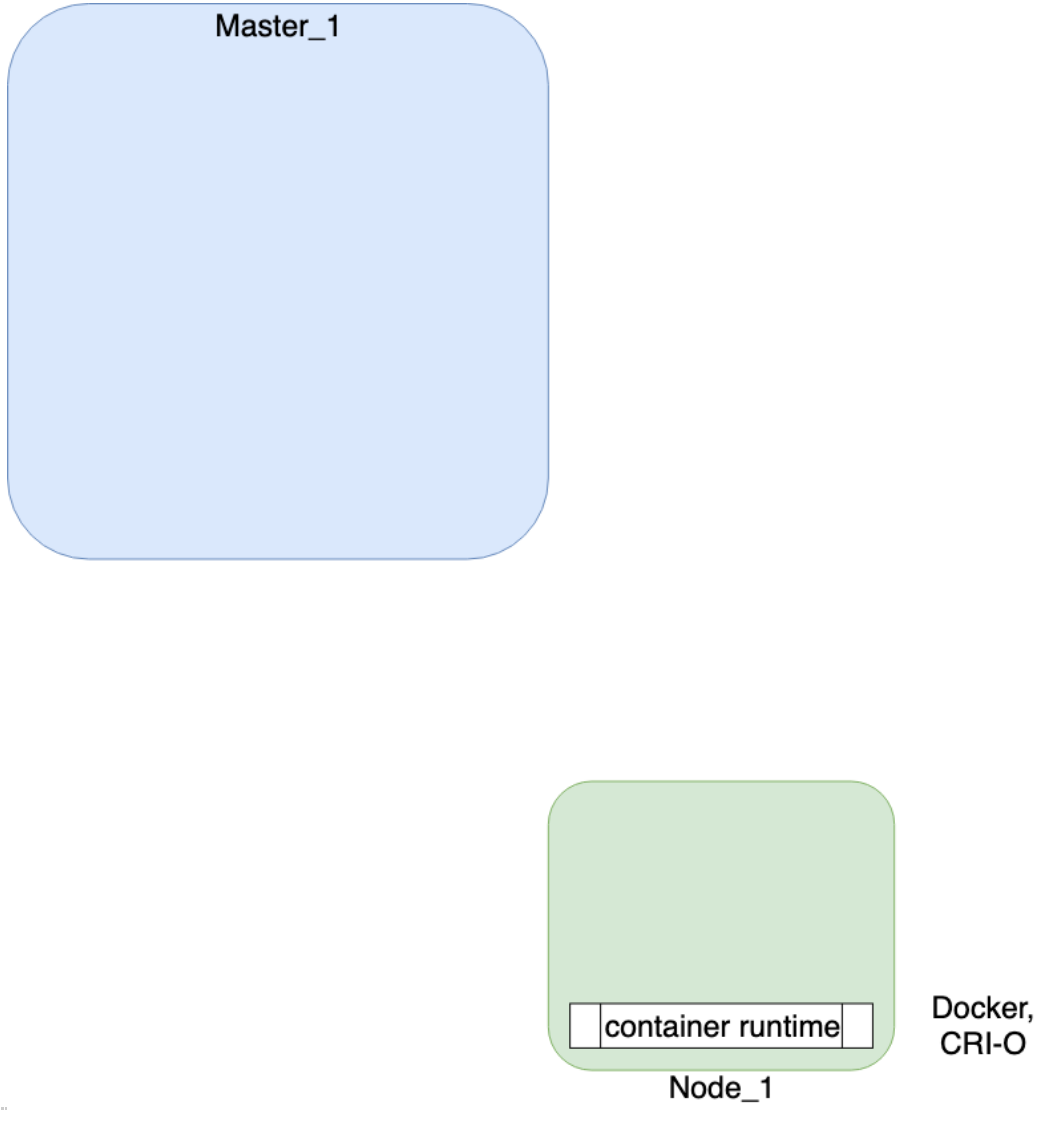
LIFECYCLE AUTOMATION (Self-service, CI/CD, Image Stream)	CONTAINER MANAGEMENT (Monitoring, Capacity, Policies)
CONTAINER INFRASTRUCTURE (Orchestration & Scheduling, Storage, Registry, Security, Networking)	

Архитектура Kubernetes

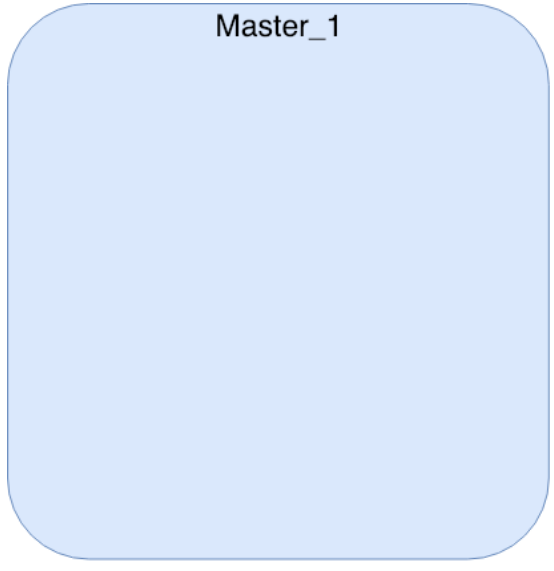
Архитектура Kubernetes

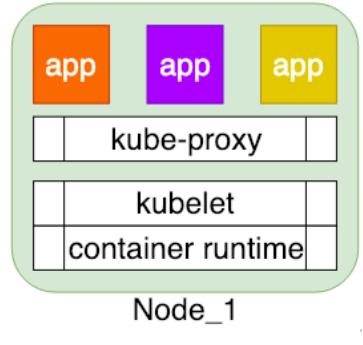


Архитектура Kubernetes



Архитектура Kubernetes





Архитектура Kubernetes

kubelet

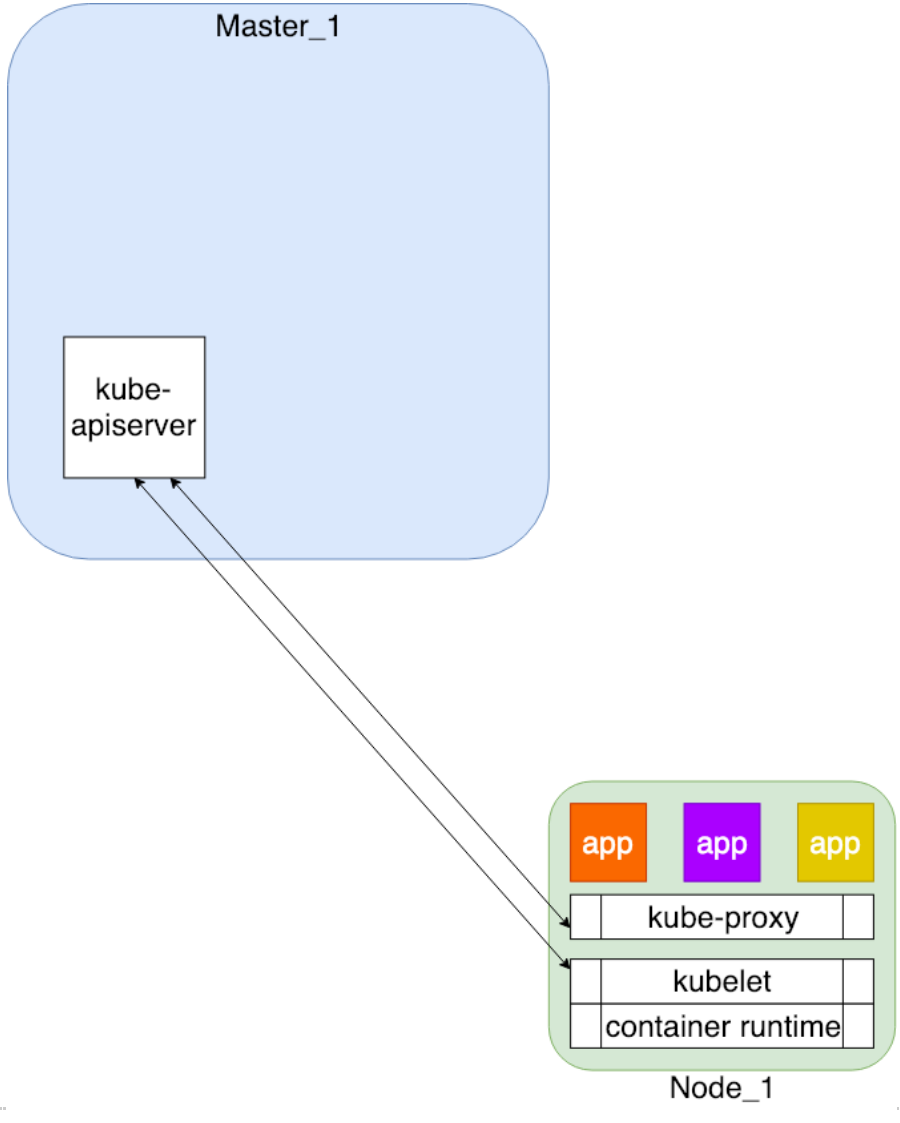
- Общается с API-Server
- Разворачивает Pods согласно PodSpec (описание Pod YAML или JSON)

Архитектура Kubernetes

kube-proxy

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

Архитектура Kubernetes

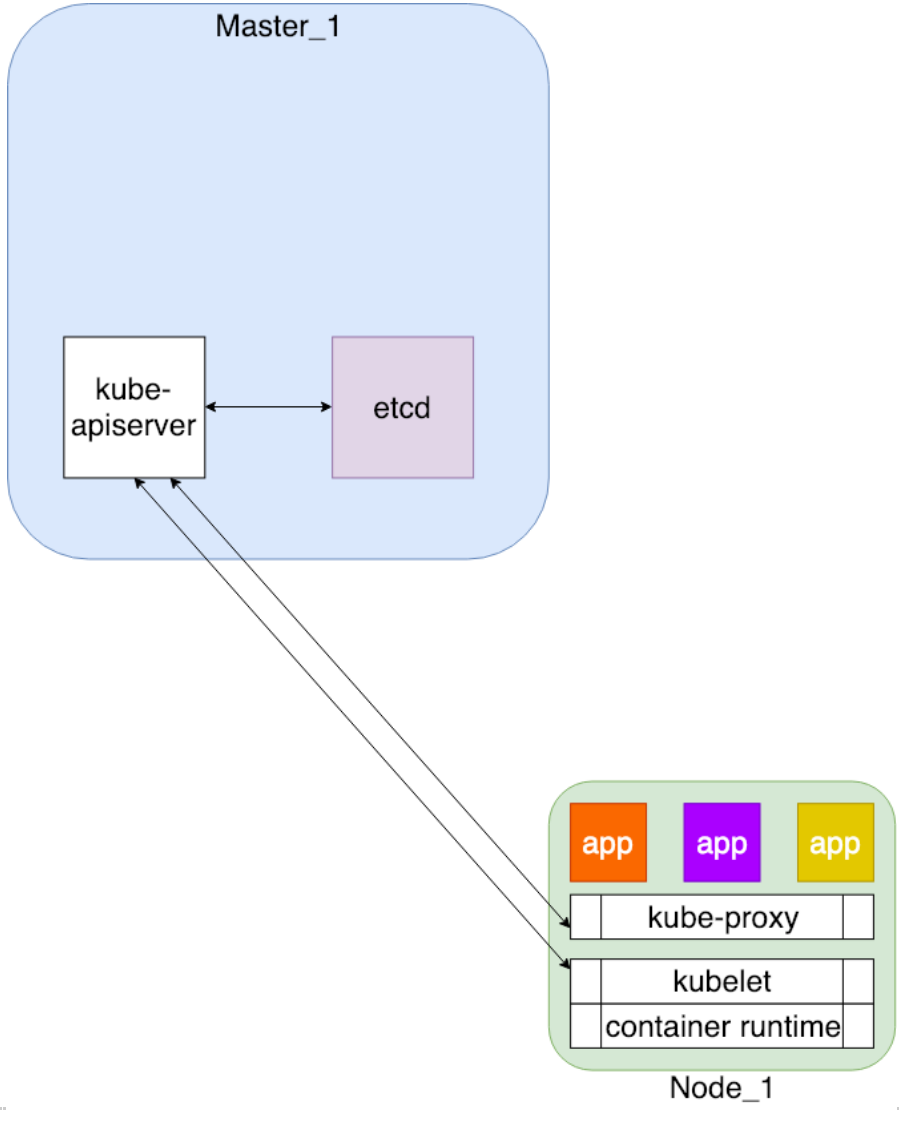


Архитектура Kubernetes

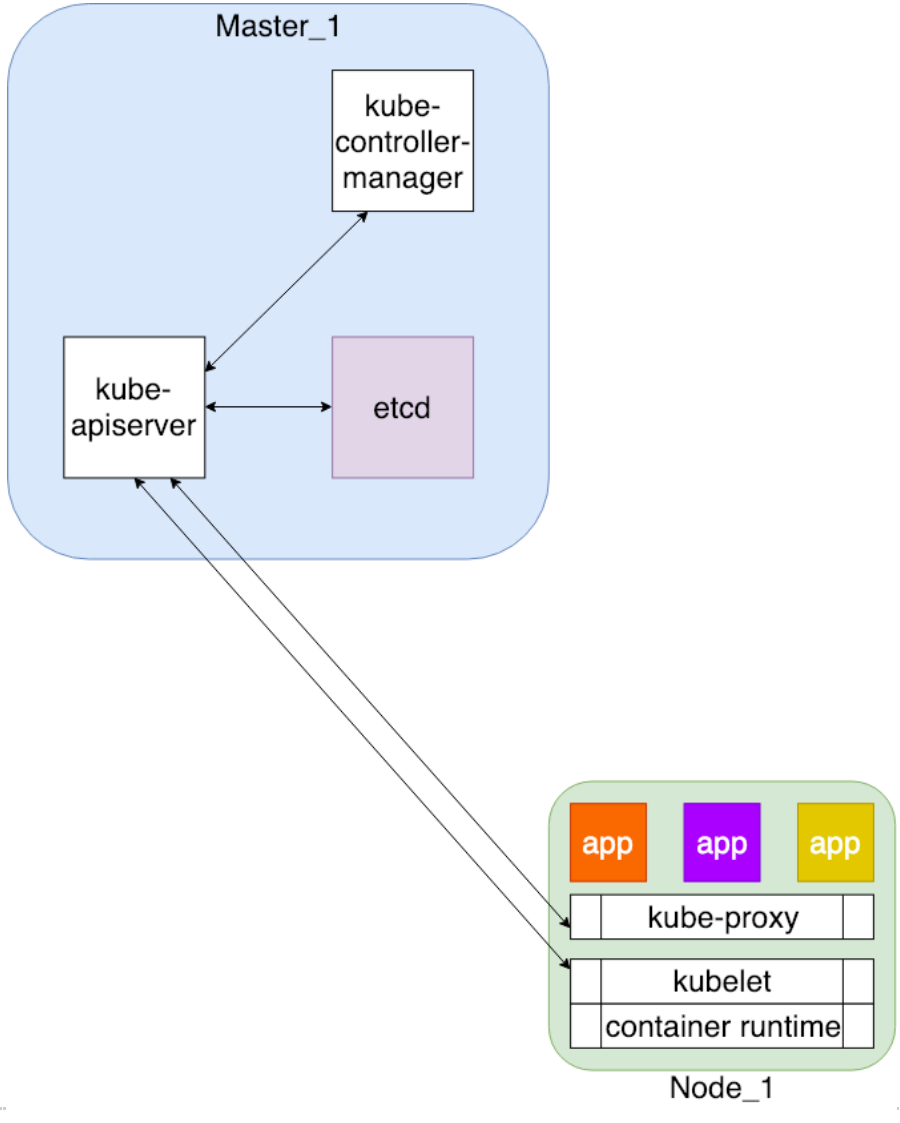
kube-apiserver

- 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

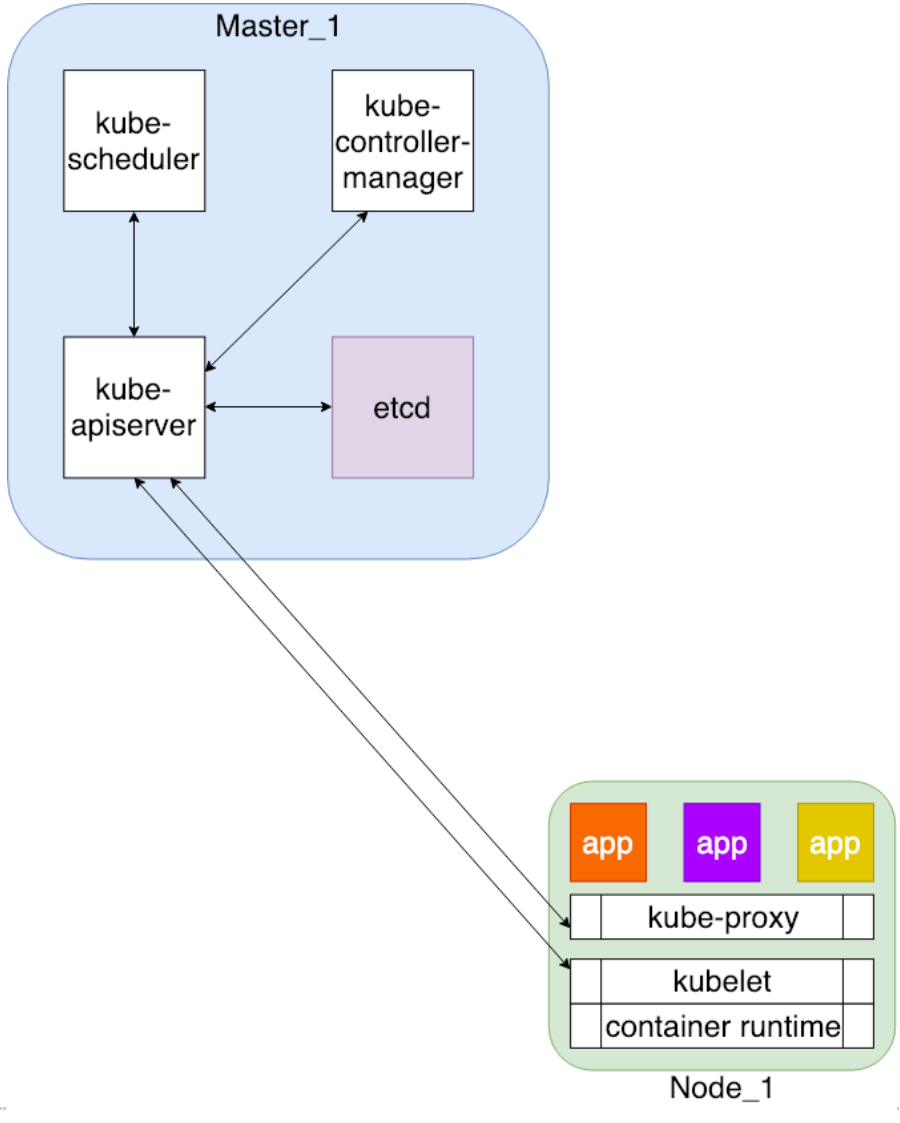


Архитектура Kubernetes

kube-controller-manager

- Взаимодействуя с API, следит за изменениями состояний объектов
- Набор различных контроллеров:
  - ReplicaSet controller
  - Endpoints controller
  - Namespace controller
  - ...
- Через API-server поддерживает определенное состояние контроллеров

Архитектура Kubernetes



Архитектура Kubernetes

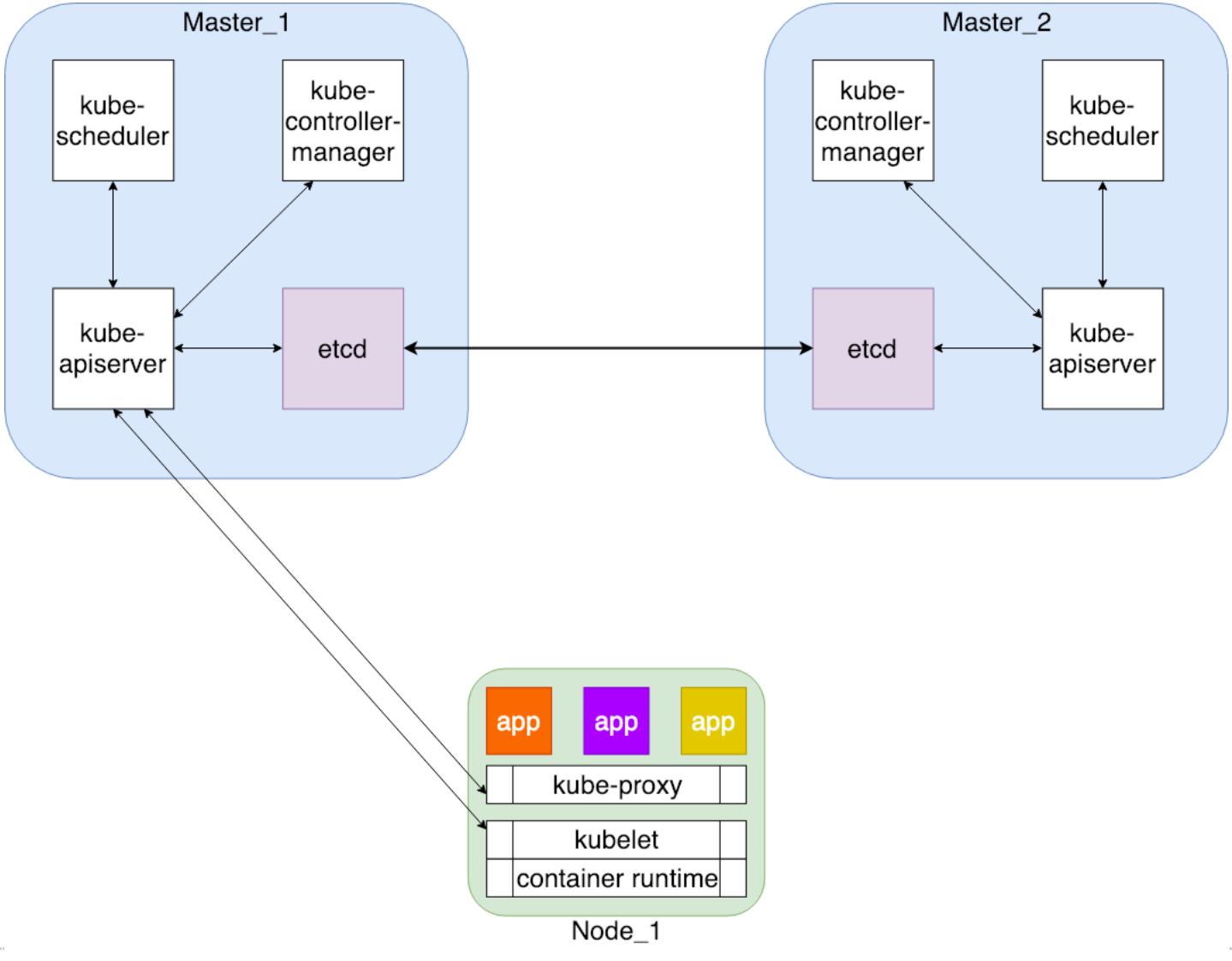
kube-scheduler

- Учет имеющихся ресурсов
- Учет требований к ресурсам
- Учет ограничений по ресурсам (affinity, anti-affinity)
- Решает, где запустить pod-ы (на основе ограничений и требований)

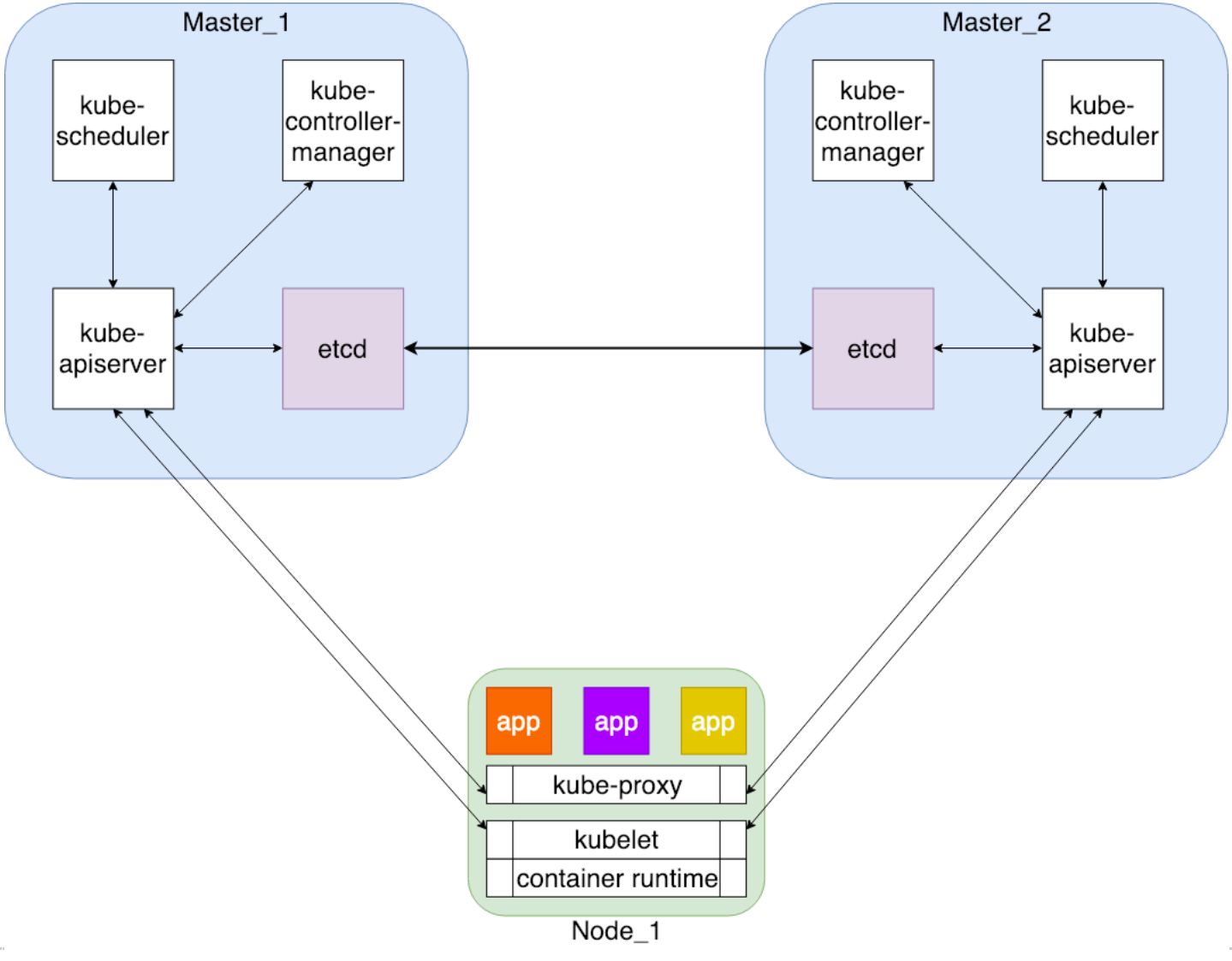
```
while True:
    pods = get_all_pods()
    for pod in pods:
        if pod.node == nil:
            assignNode(pod)
```

Как работает планировщик на самом деле

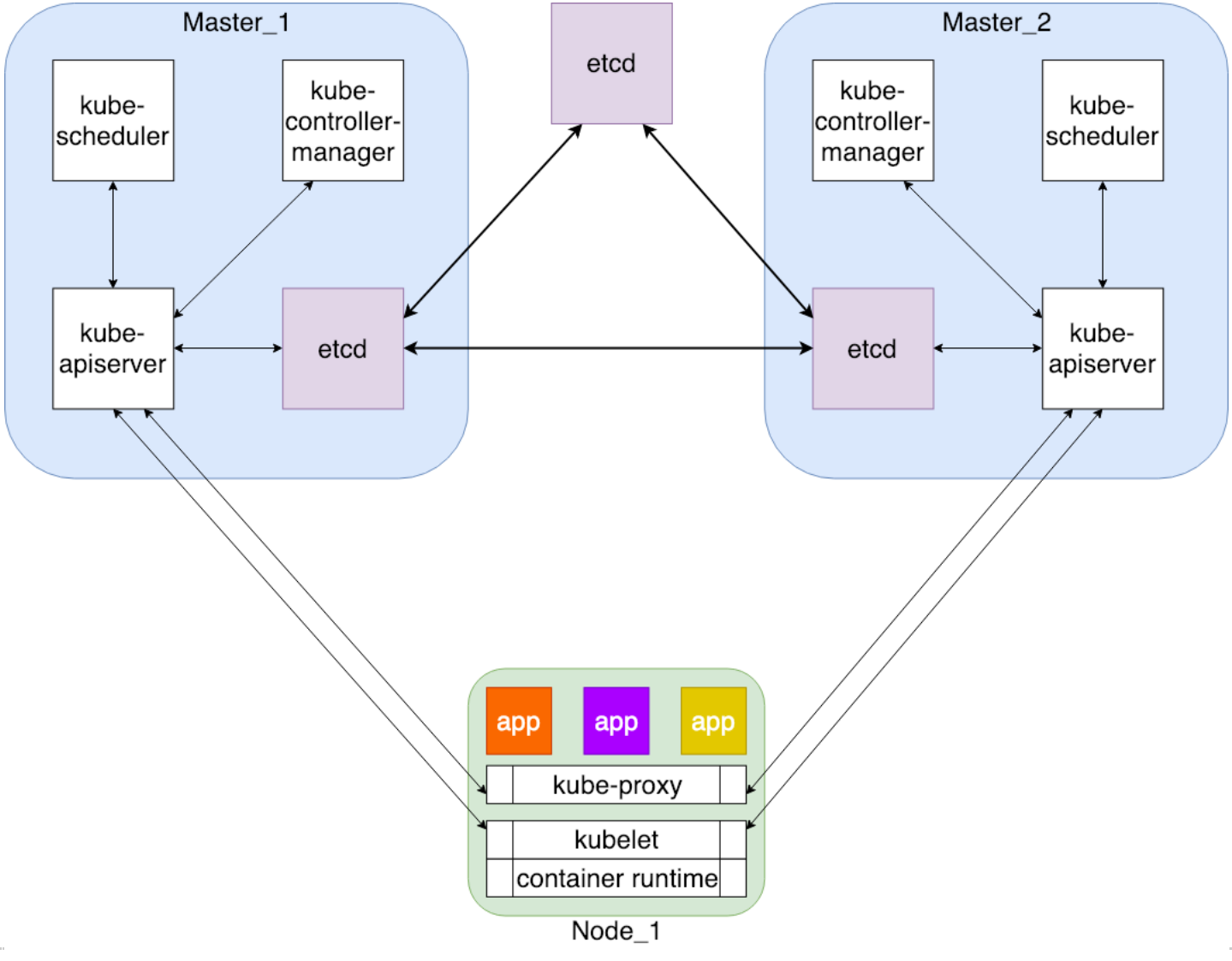
Архитектура Kubernetes



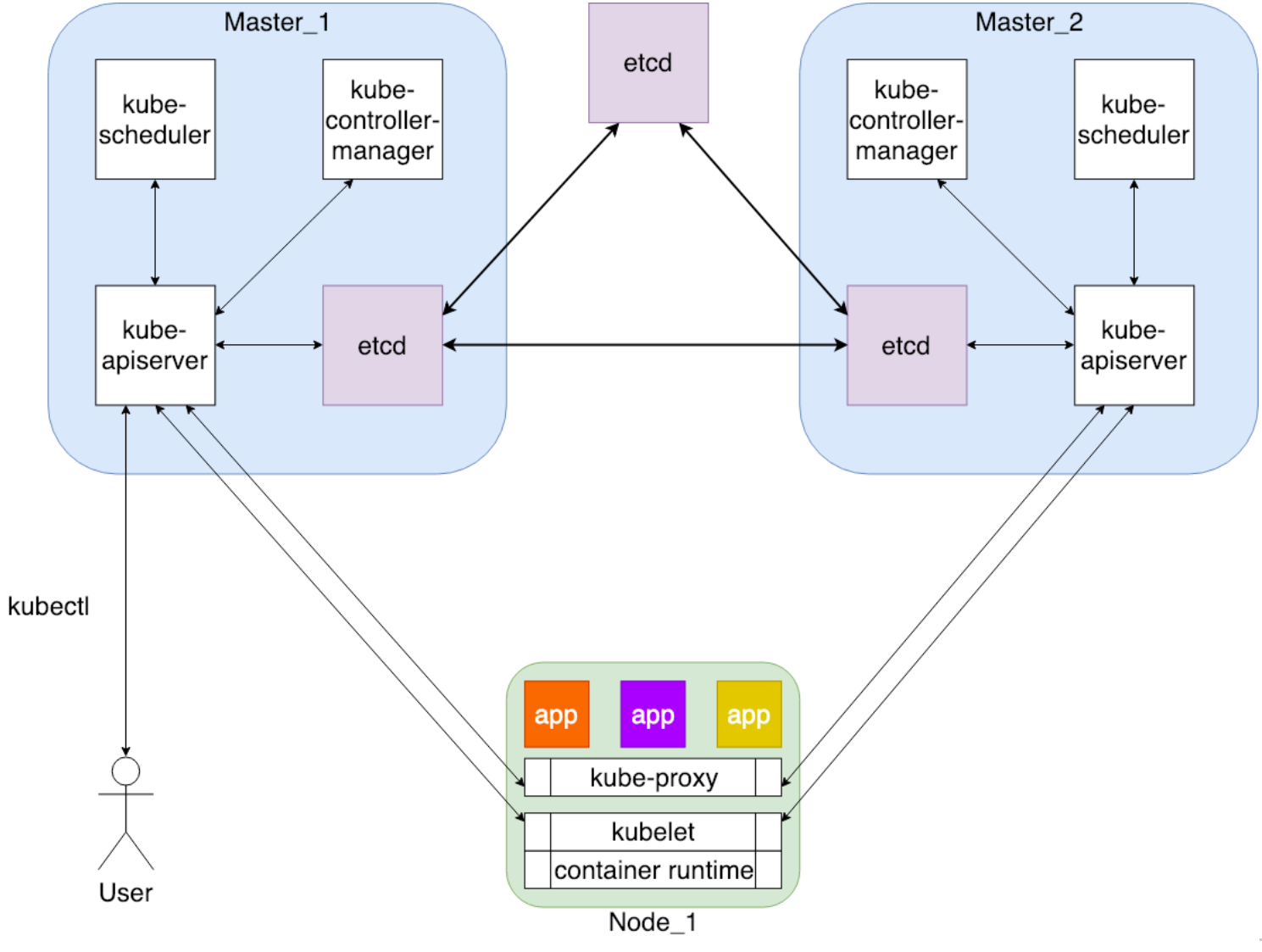
Архитектура Kubernetes



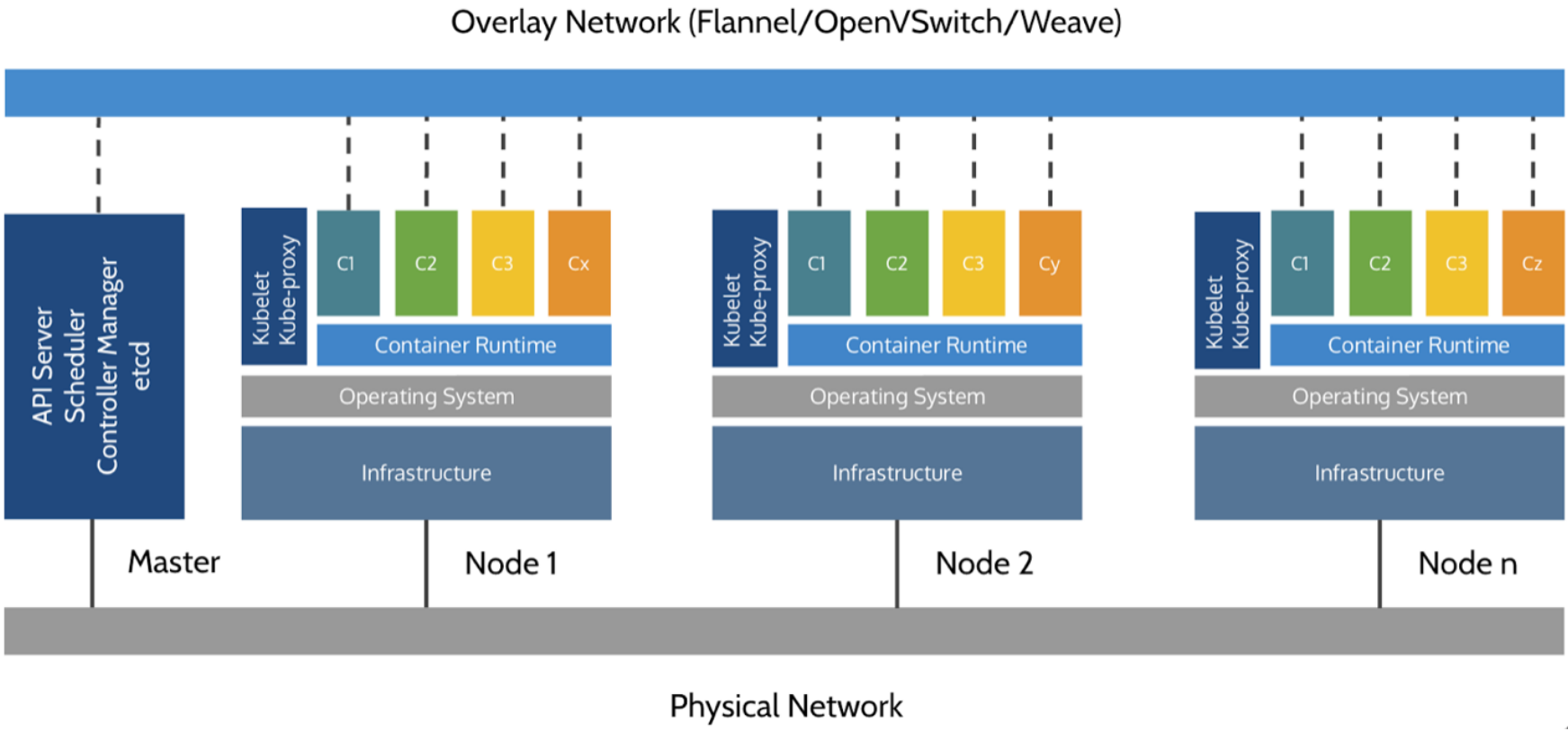
Архитектура Kubernetes



Архитектура Kubernetes



Архитектура Kubernetes



Архитектура Kubernetes

Add-ons

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

Примеры

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

Обзор и список на официальном сайте

Kubectl

Конфигурация kubectl - набор контекстов.

Контекст - это комбинация:

- Cluster - адрес kube-apiserver;
- User - реквизиты для подключения к кластеру;
- Namespace - область видимости (не обязательно, по-умолчанию default).

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

Kubectl

~/.kube/config

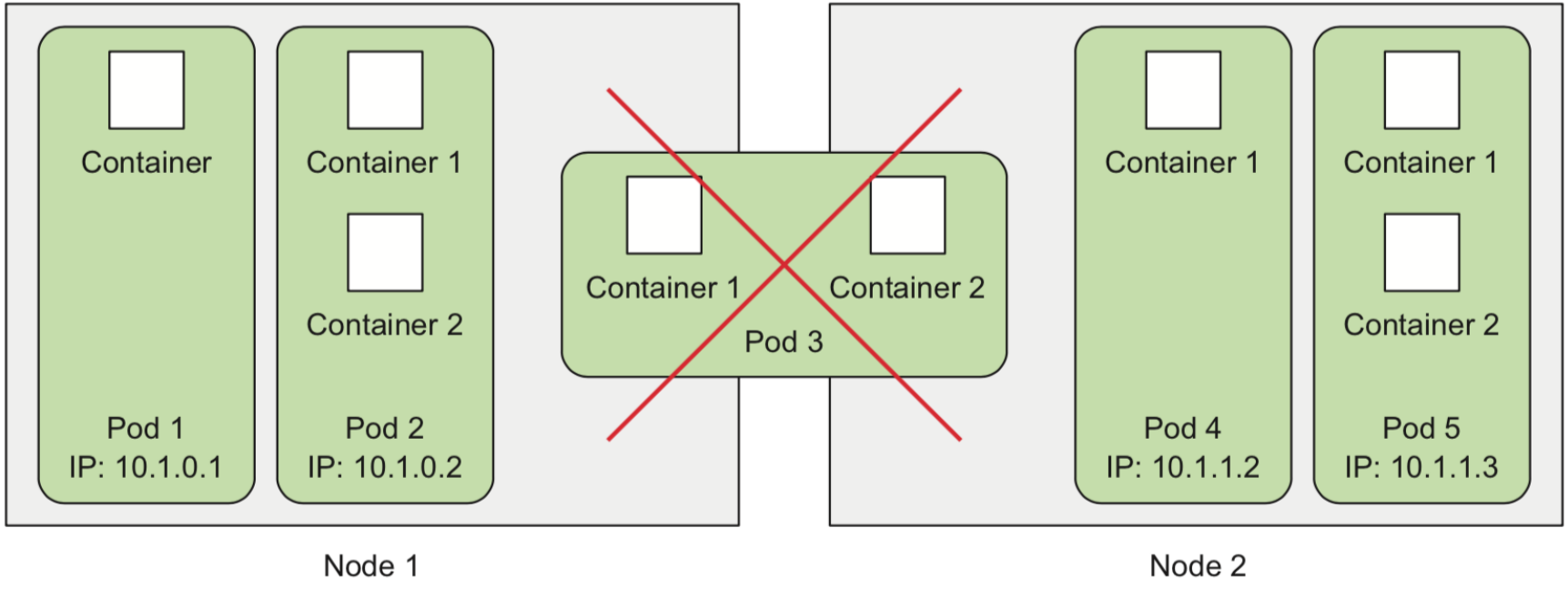
```
apiVersion: v1
clusters:                    # ⬅️ Список кластеров
- cluster:
    certificate-authority: ca.crt
    server: https://35.198.140.134
  name: kube-cluster12
contexts:                    # ⬅️ Список контекстов
- context:
    cluster: kube-cluster12
    user: kube-admin
  name: contextname
current-context: contextname # ⬅️ Текущий контекст
kind: Config
preferences: {}
users:                       # ⬅️ Список пользователей и способов их авторизации
- name: kube-admin
  user:
    client-certificate: server.crt
    client-key: server-key.pem
```

Основные концепции Kubernetes

Pods

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

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

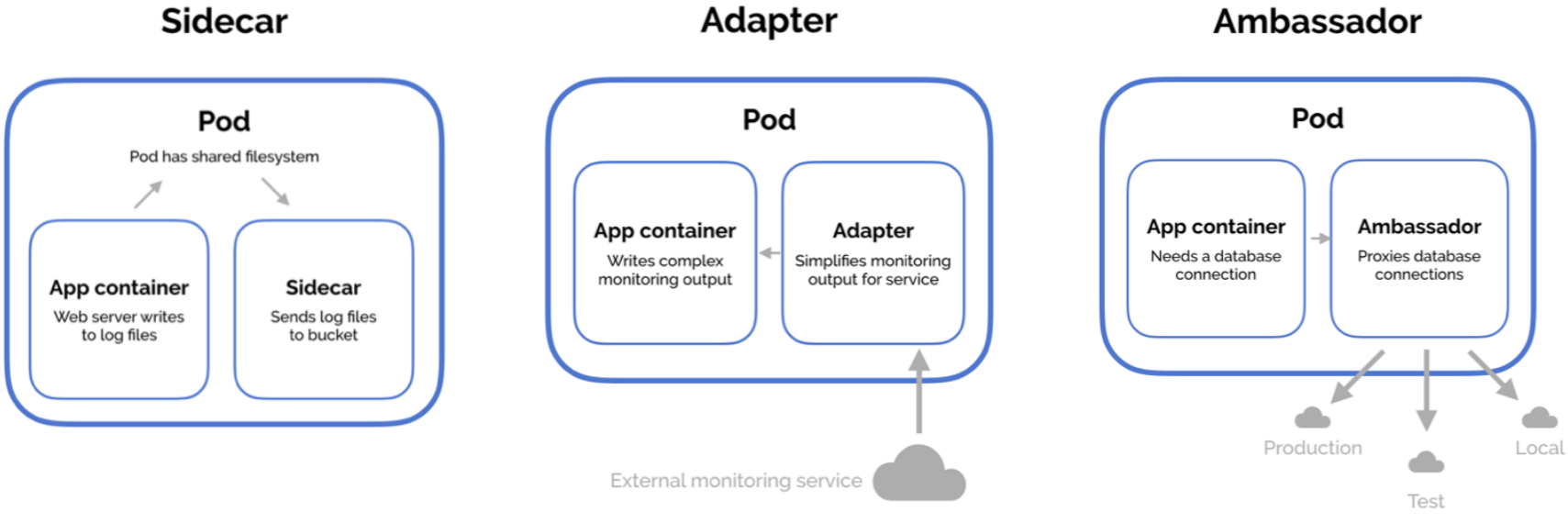


Pods

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]

Описание объектов

YAML

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

Описание объектов

prometheus-pod.yml

```
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 API группу для создания объекта

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

3 типа версий:

- **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 1.14

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

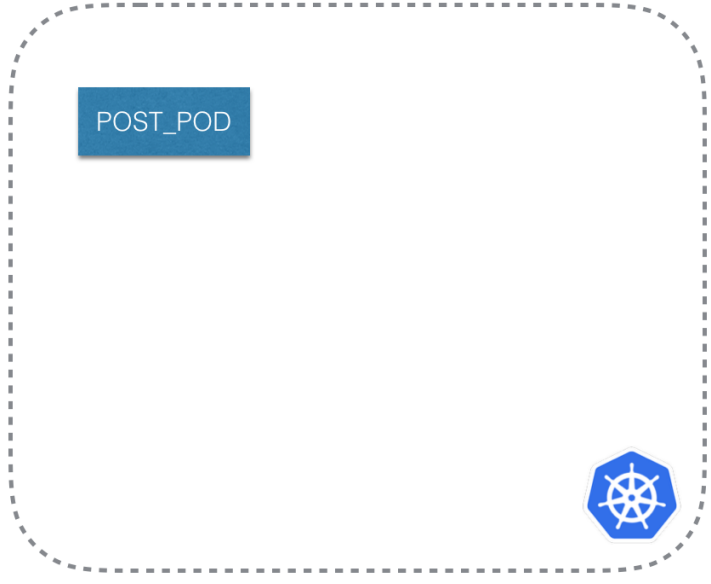
metadata

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

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

Reddit App в Kubernetes

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

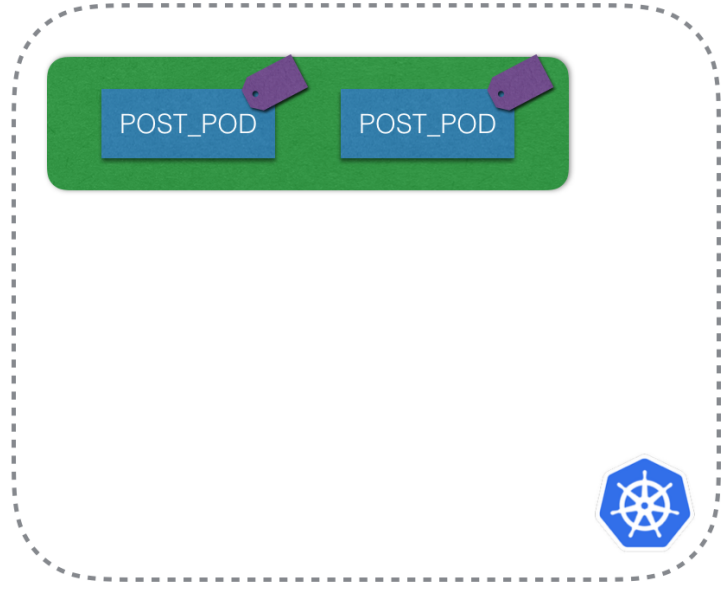


Requests & Limits

```
apiVersion: v1
kind: Pod
metadata:
  name: post
spec:
  containers:
  - image: avtandilko/post
    name: post
    resources:
      requests:
        memory: "64Mi"
        cpu: "100m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```

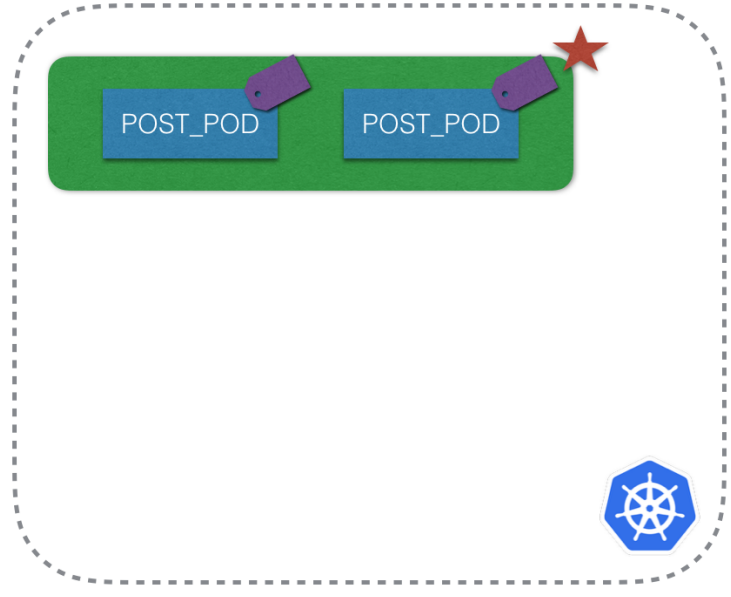
ReplicaSet

```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
```

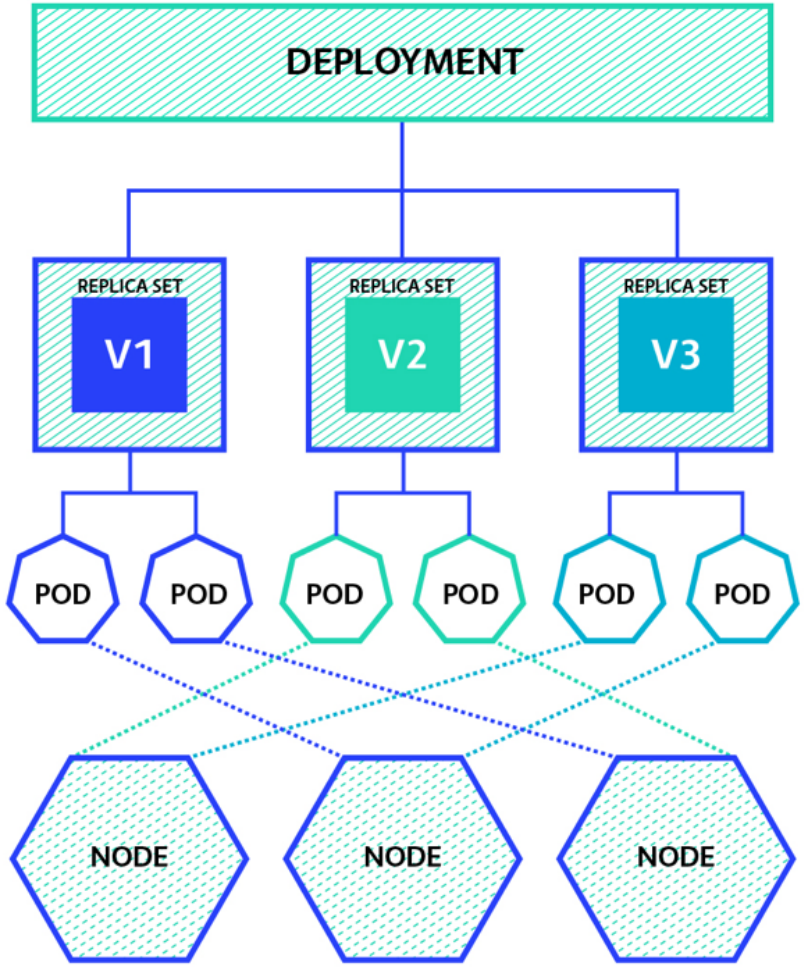


Deployment

```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
```

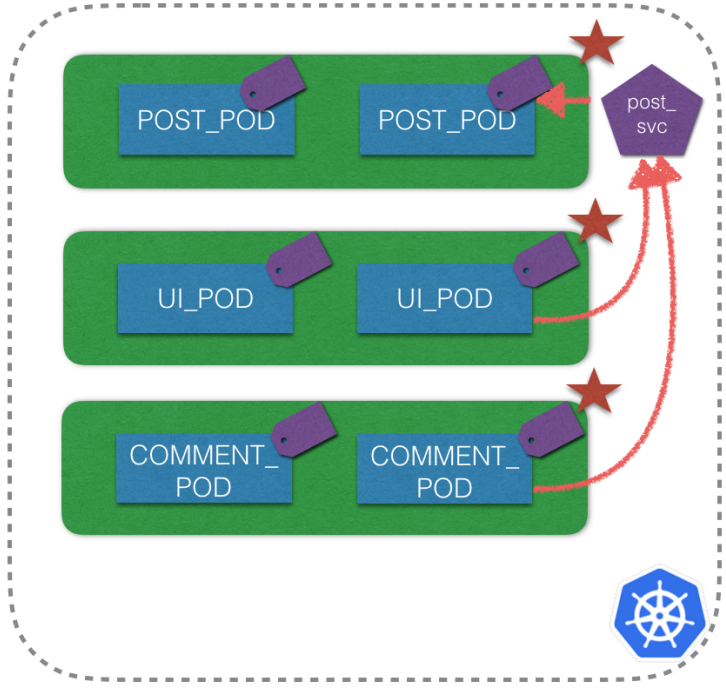


Deployment


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

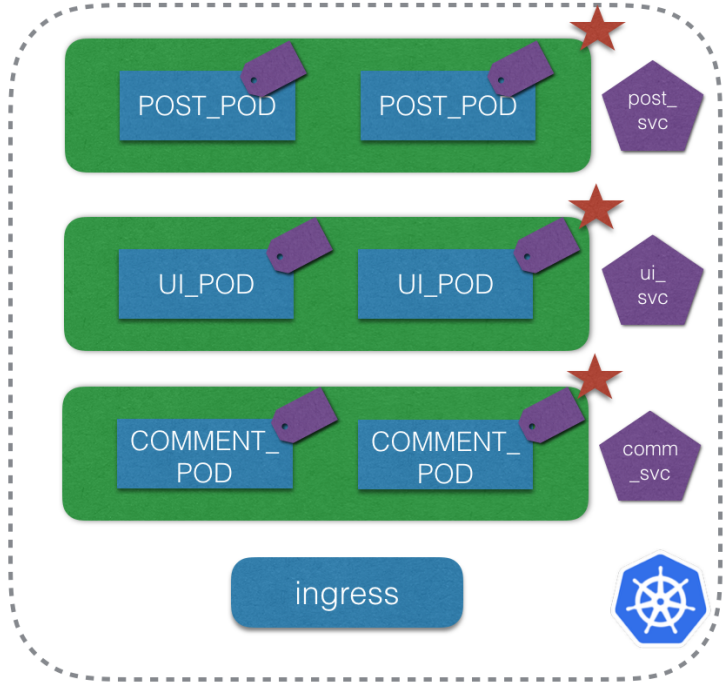
Service

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

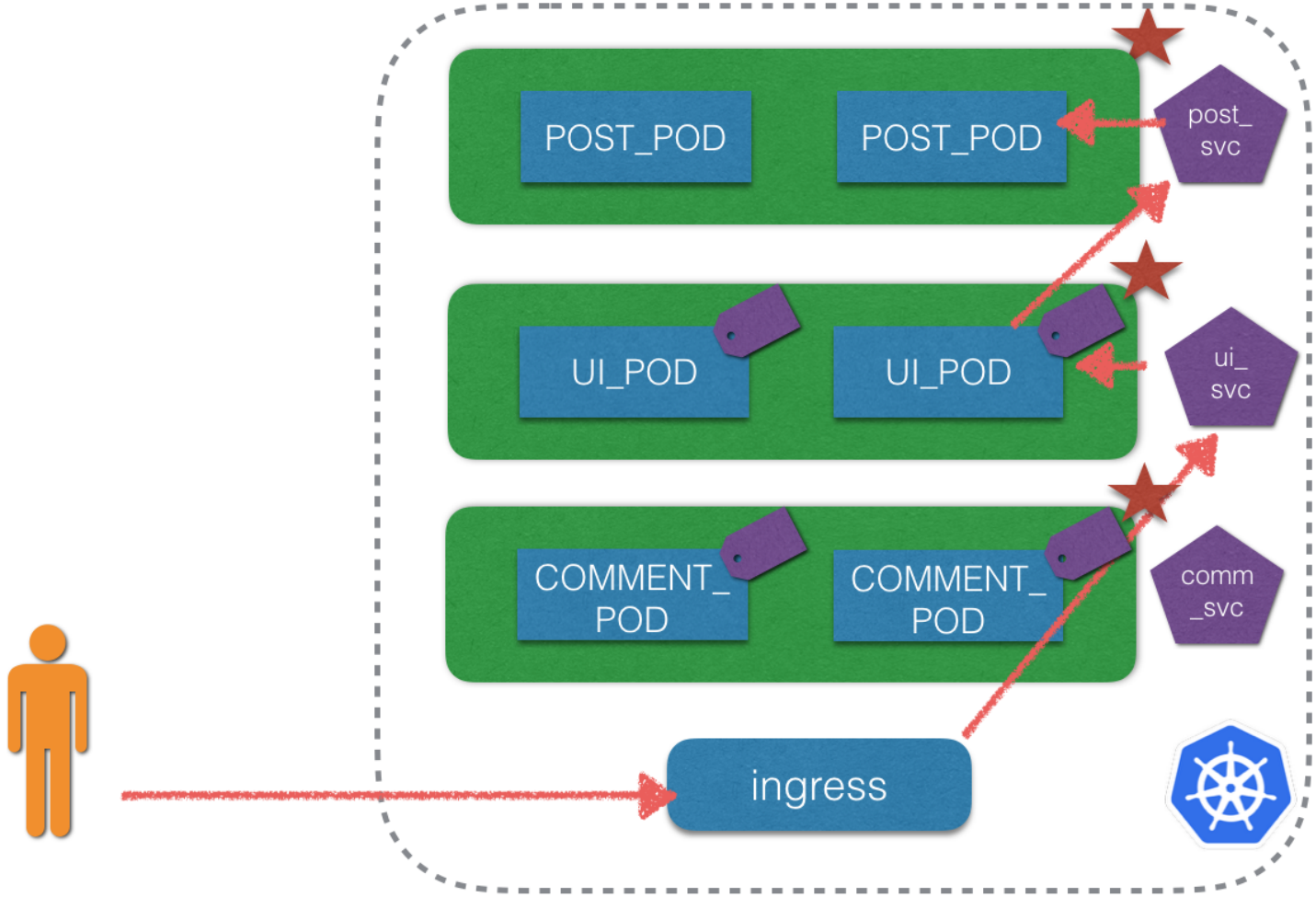


Ingress

```
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

Kubectl

```
# Создать deployment из командной строки
kubectl run DEPLOYMENT_NAME --image=IMAGE_NAME

# 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
```

Методы установки Kubernetes

Методы установки Kubernetes

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

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).

Не поддерживает:

- PersistentVolume;
- Ingress.

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;
- Доступен в beta и GA.

Kubeadm

- Официальная утилита на Go;
- Только устанавливает и настраивает Kubernetes;
- Не устанавливает Docker или любую другую Container Runtime систему;
- Не устанавливает Overlay-сети;
- В GA с версии 1.13.

Kubespray

- Ansible-репозиторий;
- Поддерживается сообществом;
- Можно поставить 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://labs.play-with-k8s.com/ - A simple, interactive and fun playground to learn Kubernetes;
- https://kubernetes.io/docs/reference/kubectl/cheatsheet/

Литература