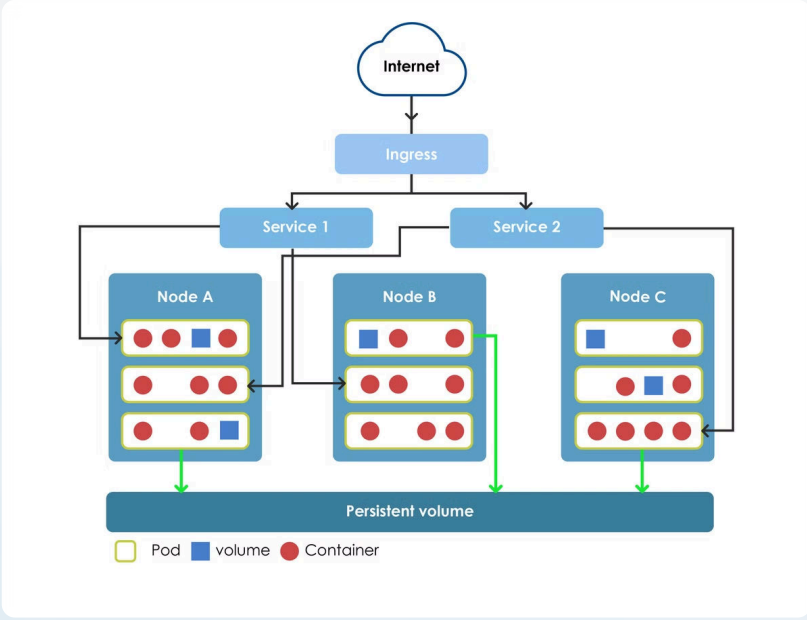
Kubernetes

Kubernetes – это система управления контейнерами с открытым исходным кодом, которая позволяет автоматизировать развертывание, масштабирование и управление контейнерными приложениями. Она предоставляет целую экосистему для управления современными распределенными приложениями в облачной среде.

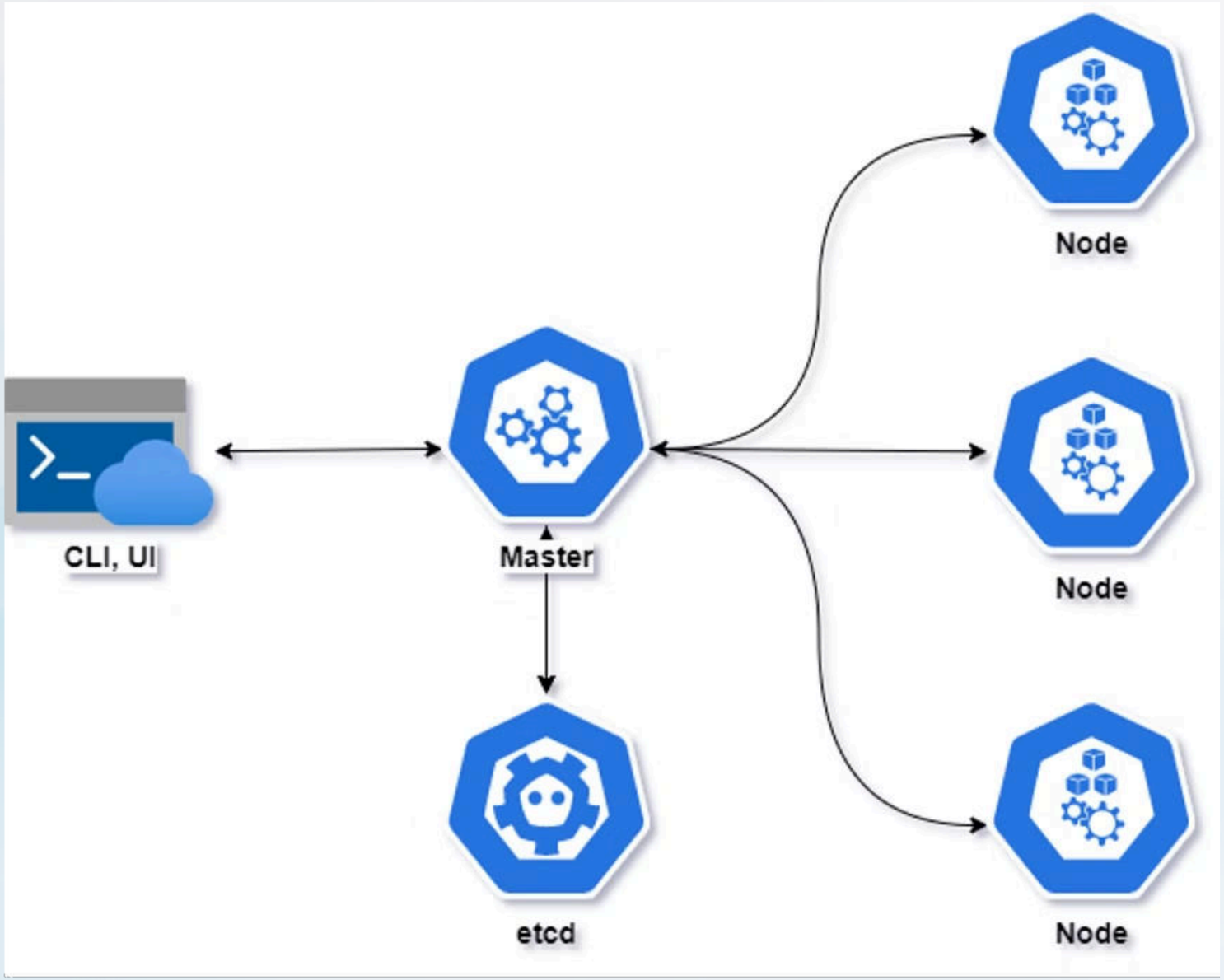
by Илья Исаев



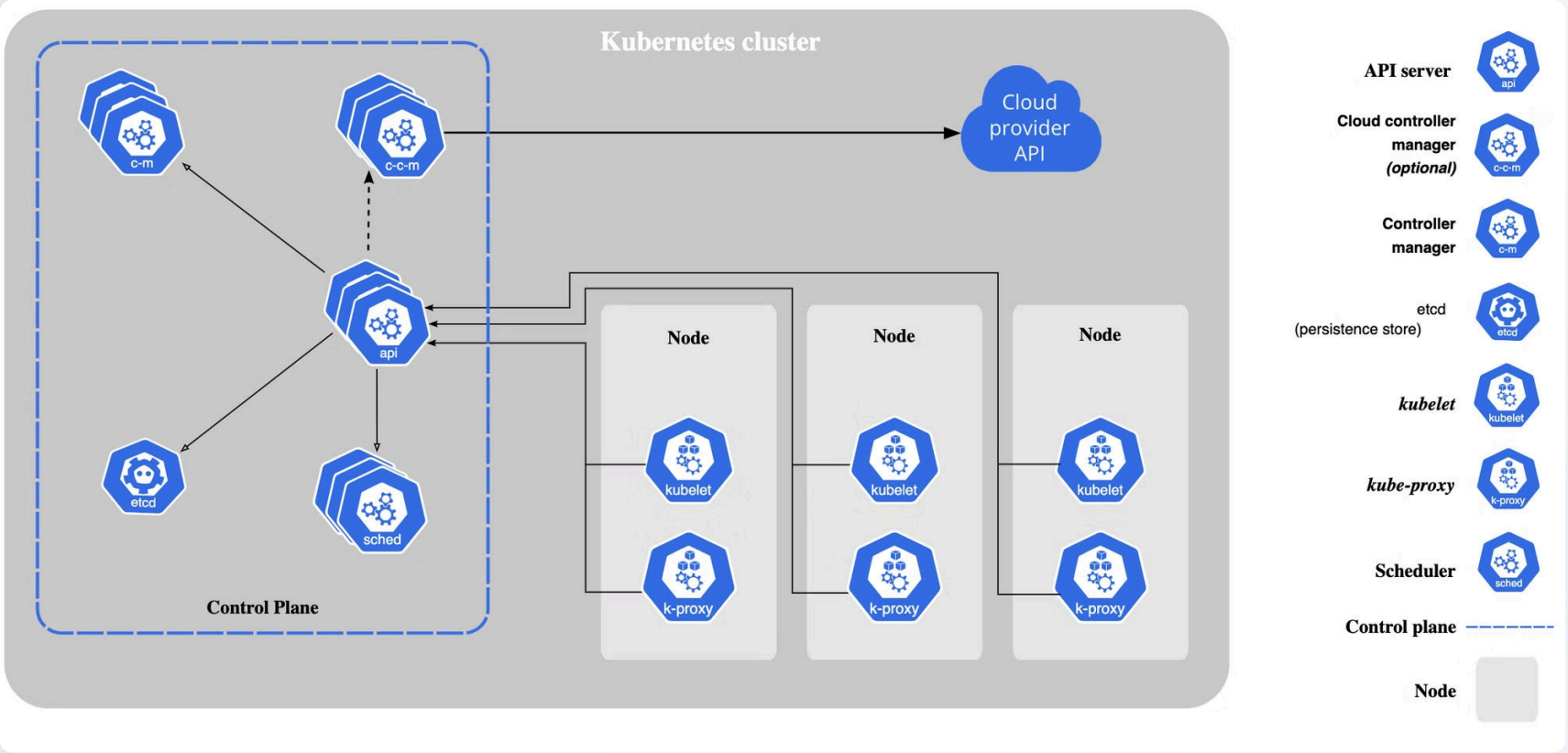
Преимущества

- Программное управление релизами
- Автоматическая отказоустойчивость
- Миграция в облака и независимость от них
- Экономное использование вычислительных ресурсов, рационально распределяет нагрузки
- Декларативное описание IaC
- Состояние кластера соответствует его описанию
- Свои слои для мониторинга, управления и т.д.

Общая схема



Компоненты



Компоненты

Kubelet

- Агент, работающий на каждом узле (master и nodes) кластера;
- Следит за исполнением жизненного цикла контейнеров;
- На вход принимает описание Pod (PodSpec) и осуществляет запуск перечисленных в них контейнеров;
- Передает в kube-apiserver информацию о запуске и работе контейнеров;
- Не отвечает за контейнеры, которые не созданы им самим.

Kube-proxy

- Сетевой прокси, работающий на каждом узле кластера;
- Предоставляет доступ к приложению в виде сетевого сервиса;
- Конфигурирует правила сети на узлах, gpи помощи них разрешаются сетевые подключения к Pod изнутри и снаружи кластера;
- Обеспечивает внутреннюю балансировку.

Kube-apiserver

- Предоставляет точку входа в API Kubernetes;
- Через него происходят взаимодействия между всеми компонентами Kubernetes;
- Реализует REST-интерфейс;
- Проверяет и конфигурирует API-объекты;
- Сохраняет состояние в ETCD.

ETCD

- Распределённое и высоконадёжное хранилище данных в формате "ключ-значение";
- В нем хранятся все данные и состояние Kubernetes;
- Кластеризуется и имеет leader election;
- Имеет механизм подписки и отслеживания изменений.

Kube-controller-manager

- Отслеживает изменения API и запускает процессы контроллера;
- Контроллер – это отдельный процесс, который на основании изменений пытается привести текущее состояние кластера к желаемому;
- Процессов (контроллеров) несколько, но все они скомпилированы в один файл;
- Типы контроллеров:
  - Replication, ReplicaSet controllers: поддерживает правильное количество Pod;
  - Endpoints controller: создает точки входа, связывая объекты Service и Pod;
  - Account controller: создают учетные записи;
  - Etc.

Kube-scheduler

- Scheduler – планировщик;
- Отслеживает созданные Pod и выбирает (планирует) Node, на котором их следует в дальнейшем установить;
- При планировании учитывает множество факторов:
  - Требования к ресурсам;
  - Имеющие ресурсы кластера;
  - Настройки ограничений ресурсов;
  - Настройки принадлежности Pod к Node (affinity, anti-affinity);
  - Местонахождение данных.

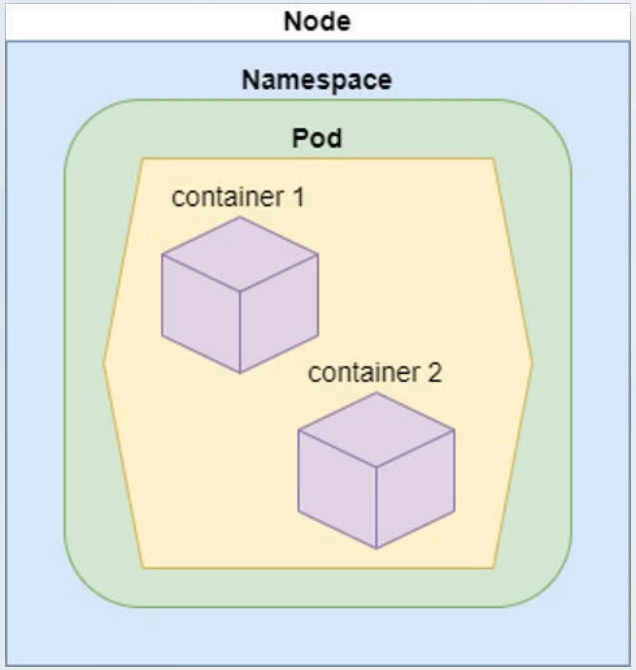
Объекты

Namespaces

- Это виртуальные пространства внутри одного физического кластера.
- Применяются для разделения приложений, потребляемых ими ресурсов, доступов пользователей.
- Существуют предопределенные Namespace:
  - kube-system: объекты самого Kubernetes;
  - default: сюда попадают объекты, для которых не указан никакой другой Namespace;
  - kube-public: общедоступные в рамках всего кластера объекты, доступно для чтения всеми пользователями.

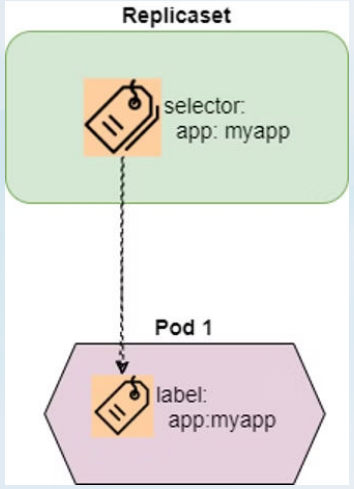
Pod

- Это минимальная сущность приложения, устанавливаемого и управляемого Kubernetes.
- Состоит из одного или нескольких контейнеров, объединенных container namespace, network, storage.
- Для запуска компонентов Kubernetes kubelet оперирует static pods.



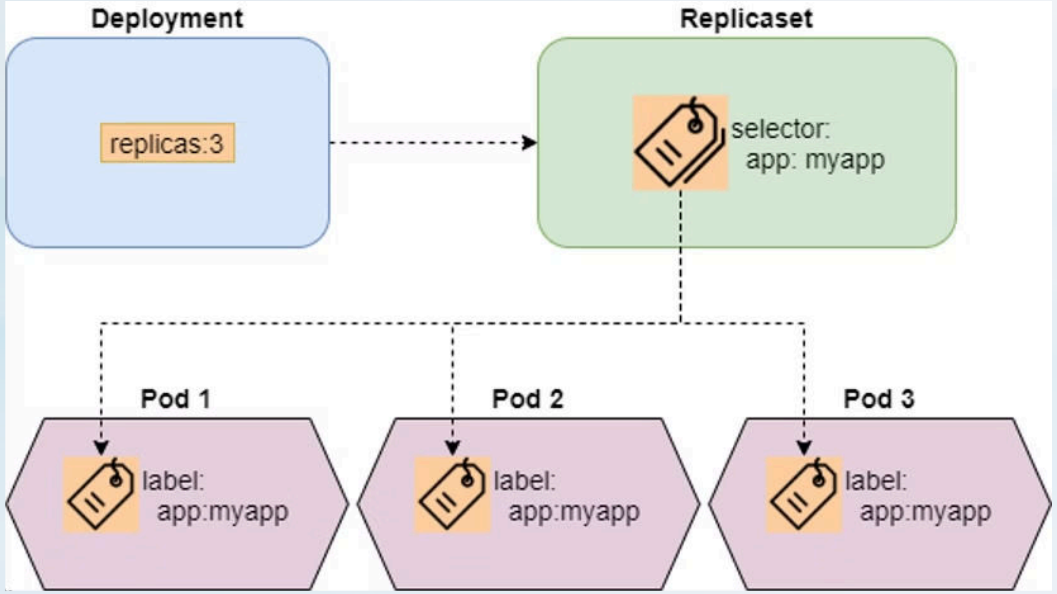
ReplicaSet

- Обеспечивает работу нескольких Pod.
- Используется для их масштабирования, обеспечения надежности и масштабирования.
- Для выбора Pod, которыми управляет ReplicaSet используется механизм label-selector.



Deployment

- Обеспечивает создание ReplicaSet и обновление в них Pod;
- Наиболее часто используемый объект для установки приложений;
- Поддерживает стратегии обновления: RollingUpdate, Recreate.



Manifest

- Способ описания объектов Kubernetes в виде файлов.
- Язык Yaml.
- Состоит из обязательных полей:
  - apiVersion: группа API и её версия;
  - kind: тип объекта;
  - metadata: имя объекта и дополнительные данные.
- В spec содержится спецификация объекта.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Процесс запуска