ML Advanced

Kubernetes, контейнерная оркестрация

otus.ru

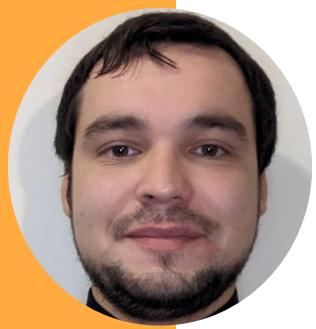
Проверить, идет ли запись

Меня хорошо видно && слышно?

Ставим “+”, если все хорошо
“-”, если есть проблемы

Тема вебинара

Kubernetes, контейнерная оркестрация



Гайнуллин Дмитрий

Machine Learning Engineer в AIC

- Разработка моделей для распознавания речи
- Прогнозирование ключевых метрик
- Развертывание моделей в продакшене

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в учебной группе

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

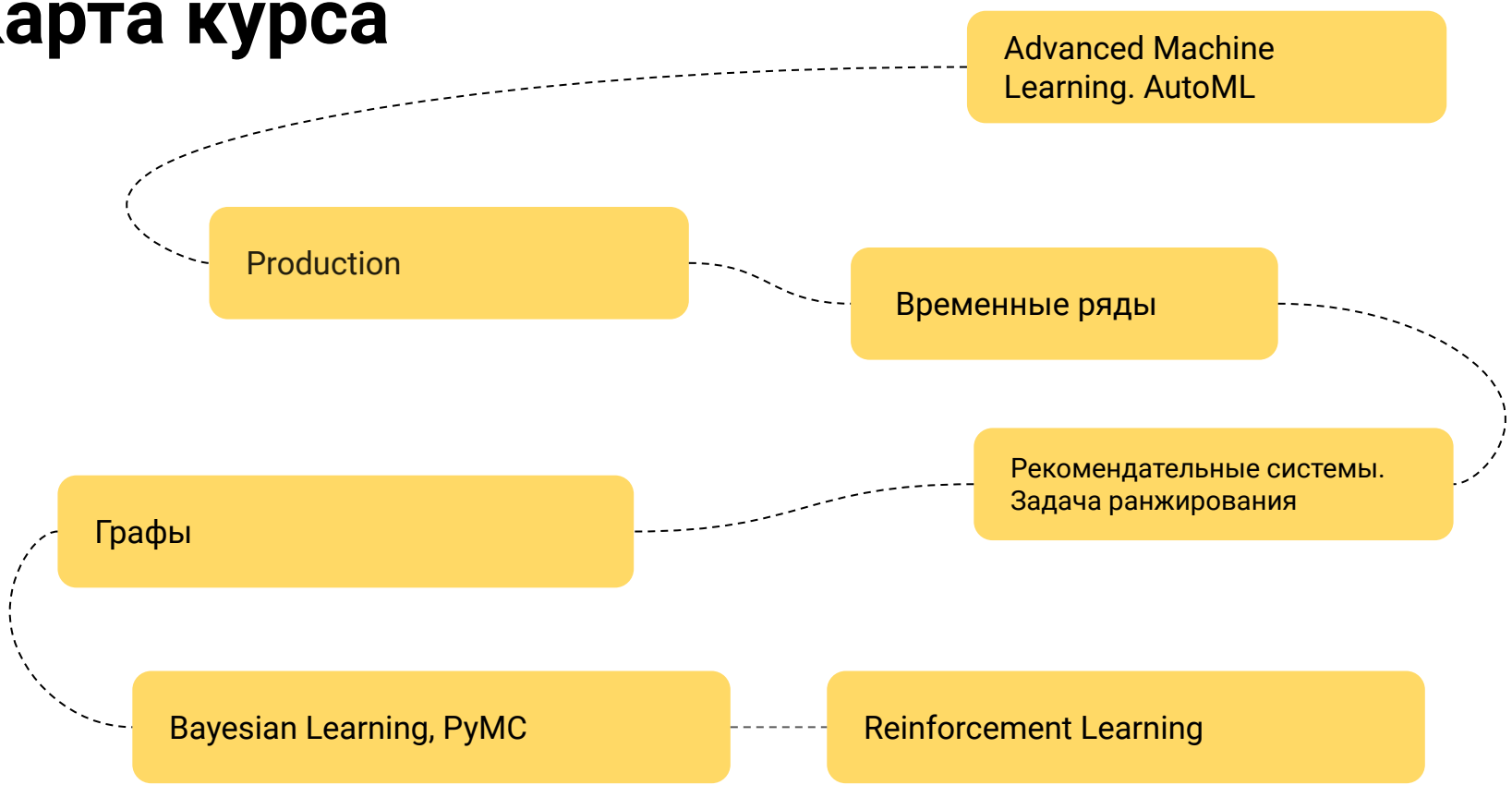
Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

Карта курса


Advanced Machine Learning. AutoML
Production
Временные ряды
Рекомендательные системы. Задача ранжирования
Графы
Bayesian Learning, PyMC
Reinforcement Learning

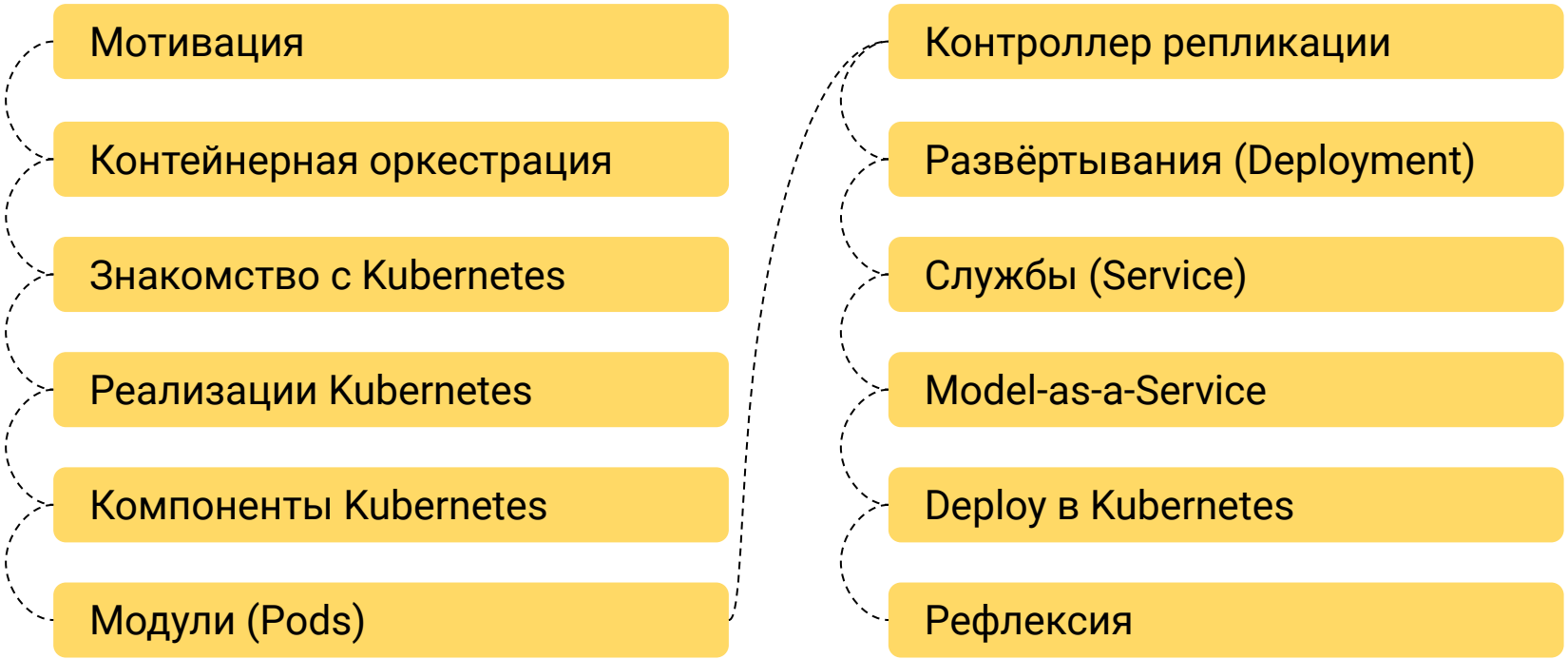
Цели вебинара

1. Познакомимся с контейнерной оркестрацией
2. Познакомимся с Kubernetes
3. Посмотрим как разворачивать образы в Kubernetes

Смысл

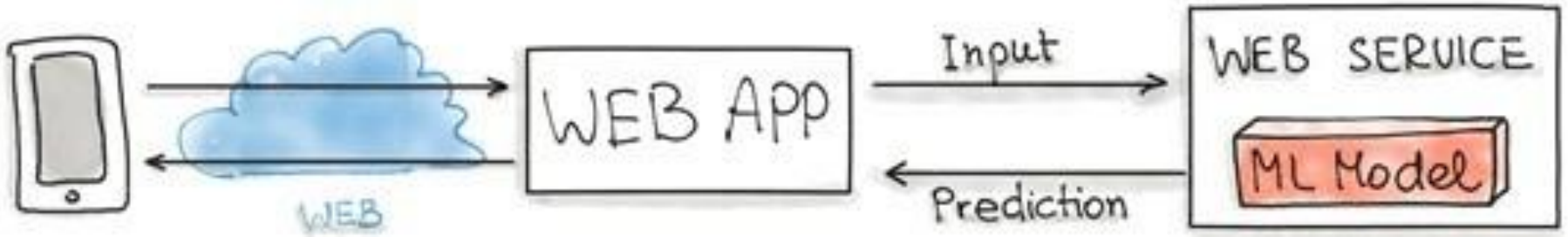
1. Сможем работать с Kubernetes
2. Сможем запускать свои контейнеры в Kubernetes

Маршрут вебинара

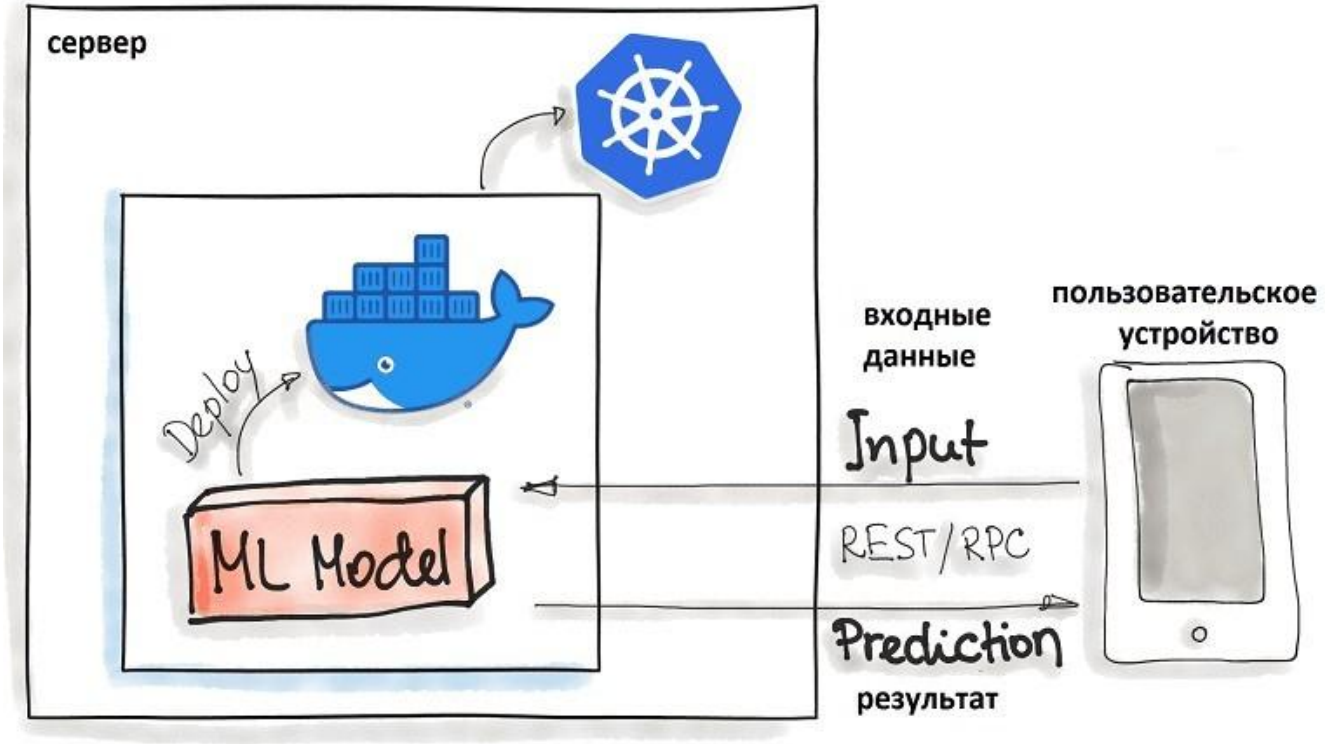


Мотивация

Модель как услуга (Model-as-a-Service)



Развёртывание с помощью контейнеров Docker



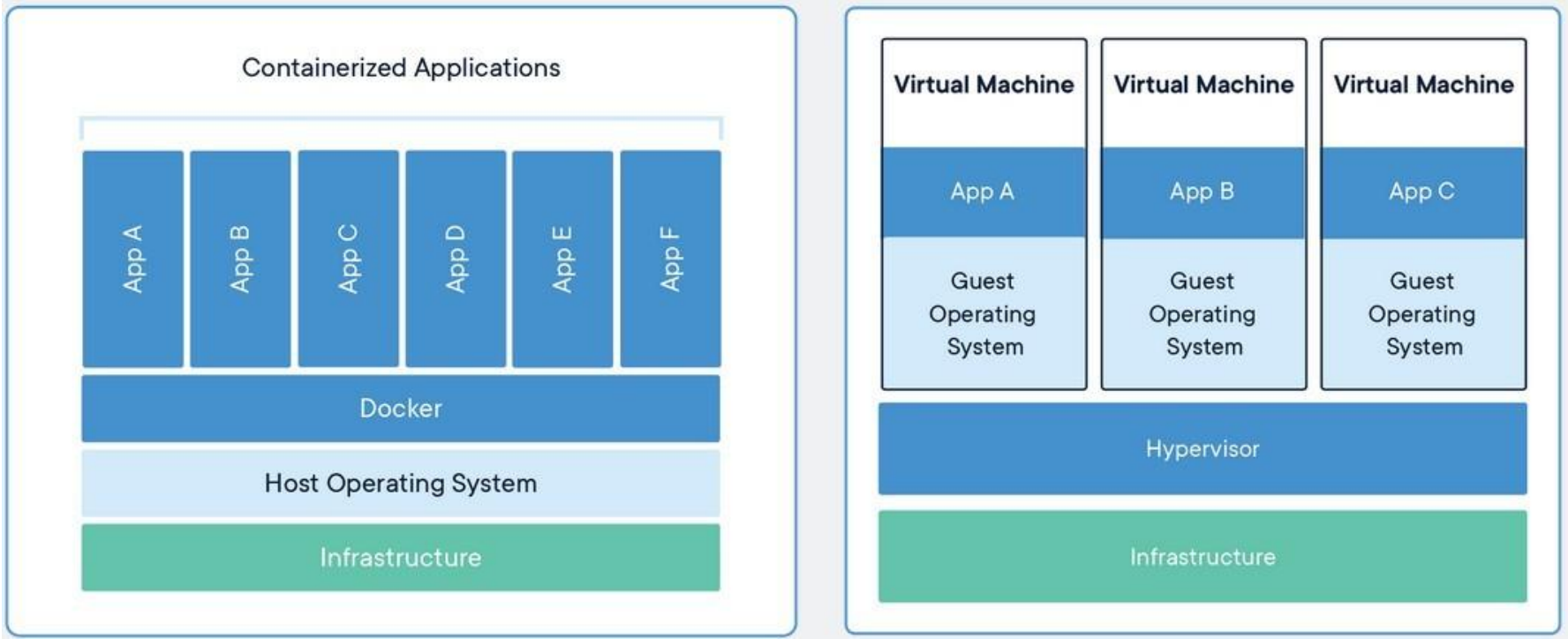
Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Контейнерная оркестрация

Контейнеры

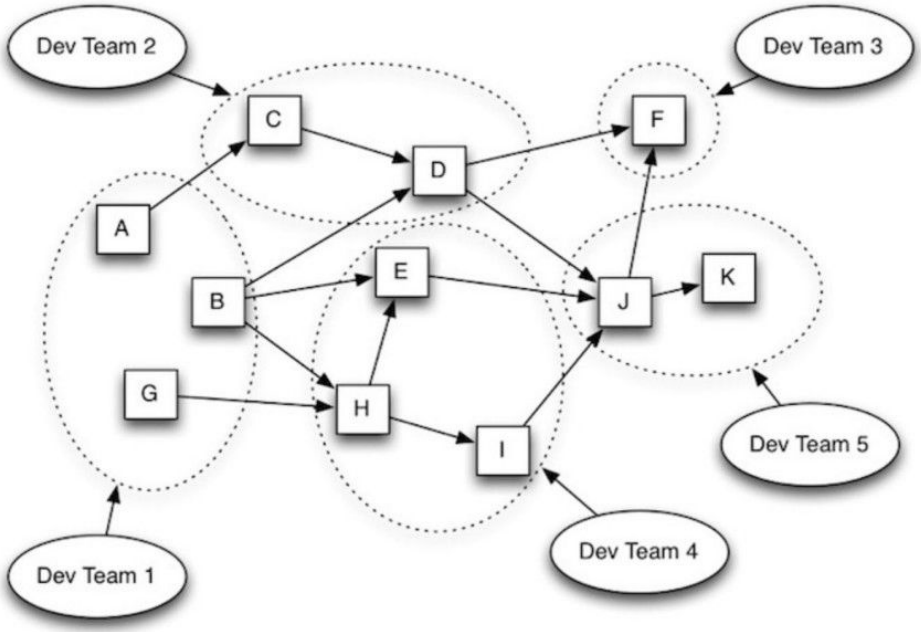

Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure
Virtual Machine
Virtual Machine
Virtual Machine
App A
App B
App C
Guest Operating System
Guest Operating System
Guest Operating System
Hypervisor
Infrastructure

Среда выполнения контейнеров

- Docker
- Containerd
- CRI-O
- Kubernetes CRI (Container Runtime Interface)

Много сервисов

- Приложение — это контейнер
- Приложениям нужны ресурсы
- Разным приложениям нужно разное количество ресурсов
- Приложения работают на разных машинах
- Разные машины имеют разную конфигурацию
- Приложения должны знать друг о друге
- Сложно рассчитать потребление ресурсов



Контейнерная оркестрация

Оркестрация — это координация взаимодействия контейнеров

- Размещение (кого куда поставить)
- Реплицирование / Масштабирование
- Проверка готовности сервиса
- Перезапуск
- Перепланирование
- Управление сервисами
  - Обнаружение (service discovery)
  - Балансировка (load balancing)

Оркестраторы

- Kubernetes https://kubernetes.io
- Docker Swarm https://docs.docker.com/engine/swarm/
- Apache Mesos http://mesos.apache.org/
- Nomad https://www.nomadproject.io/

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Знакомство с Kubernetes

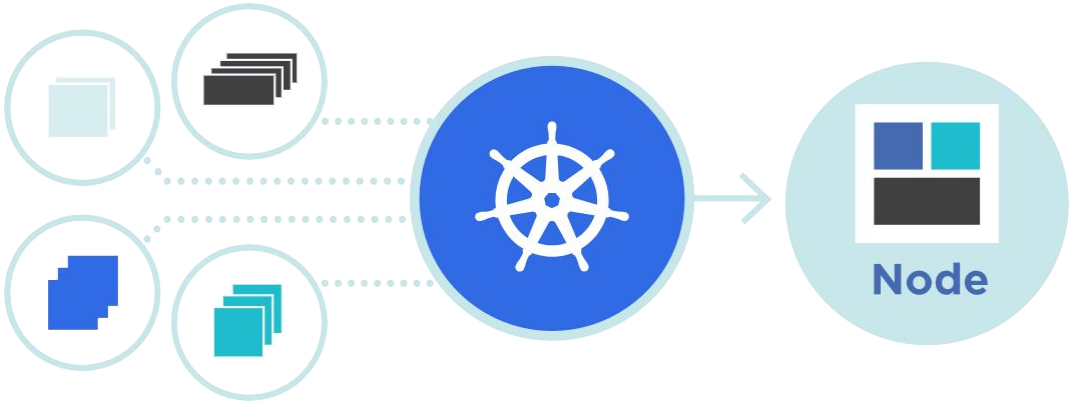
Что такое Kubernetes?

Kubernetes (K8s) - это открытое программное обеспечение для автоматизации развёртывания, масштабирования и управления контейнеризированными приложениями

Kubernetes группирует контейнеры, составляющие приложение,

в логические единицы для более простого управления и

обнаружения



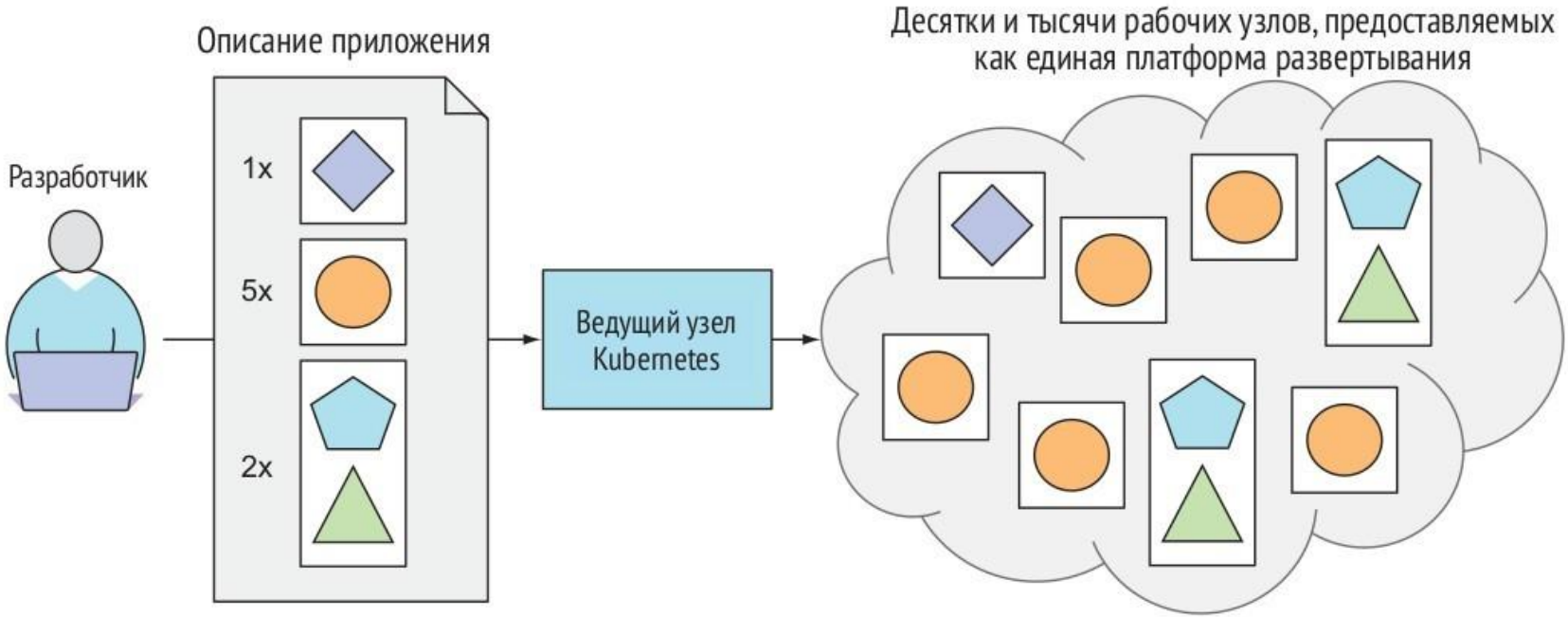
Что может Kubernetes?

- **Мониторинг сервисов и распределение нагрузки** - сбалансировать нагрузку и распределить сетевой трафик
- **Оркестрация хранилища** - автоматически смонтировать систему локальное или облачное хранилище
- **Автоматическое развертывание и откаты** - автоматизировать создание новых, удаления существующих и распределения всех их ресурсов в новый

Что может Kubernetes?

- **Автоматическое распределение нагрузки** - разместить контейнеры на узлах так, чтобы наиболее эффективно использовать ресурсы
- **Самоконтроль** - перезапускает отказавшие, заменяет и завершает работу контейнеров, которые не проходят проверку работоспособности
- **Управление конфиденциальной информацией и конфигурацией** - хранить и управлять конфиденциальной информацией, такой как пароли, OAuth-токены и ключи SSH

Что делает Kubernetes?



Как определить, нужен ли Вам Kubernetes?

1) Тип системы
2) Тип нагрузки
3) Доступ к кластеру у DevOps
4) Кто владеет кластером
5) Готовы ли вы менять процессы под развёртывание в k8s
6) Оцените операционный контекст

https://vas3k.club/post/2570/

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Реализации Kubernetes

Локальные

- Minikube vs. kind vs. k3s - What should I use? https://shipit.dev/posts/minikube-vs-kind-vs-k3s.html
- minikube https://minikube.sigs.k8s.io/docs/
- kind https://kind.sigs.k8s.io/
- k3s https://k3s.io/
- VMware Tanzu Community Edition https://tanzucommunityedition.io/
- OKD https://www.okd.io/

Промышленные

- Red Hat OpenShift

https://www.redhat.com/en/technologies/cloud-computing/openshift

- VMware Tanzu https://tanzu.vmware.com/

Облачные

- Amazon Elastic Kubernetes Service
  https://aws.amazon.com/eks
- Google Kubernetes Engine
  https://cloud.google.com/kubernetes-engine
- Azure Kubernetes (AKS)
  https://azure.microsoft.com/ru-ru/services/kubernetes-service/
- Yandex Managed Service for Kubernetes
  https://cloud.yandex.ru/services/managed-kubernetes

Вопросы?

Ставим “+”,
если вопросы есть

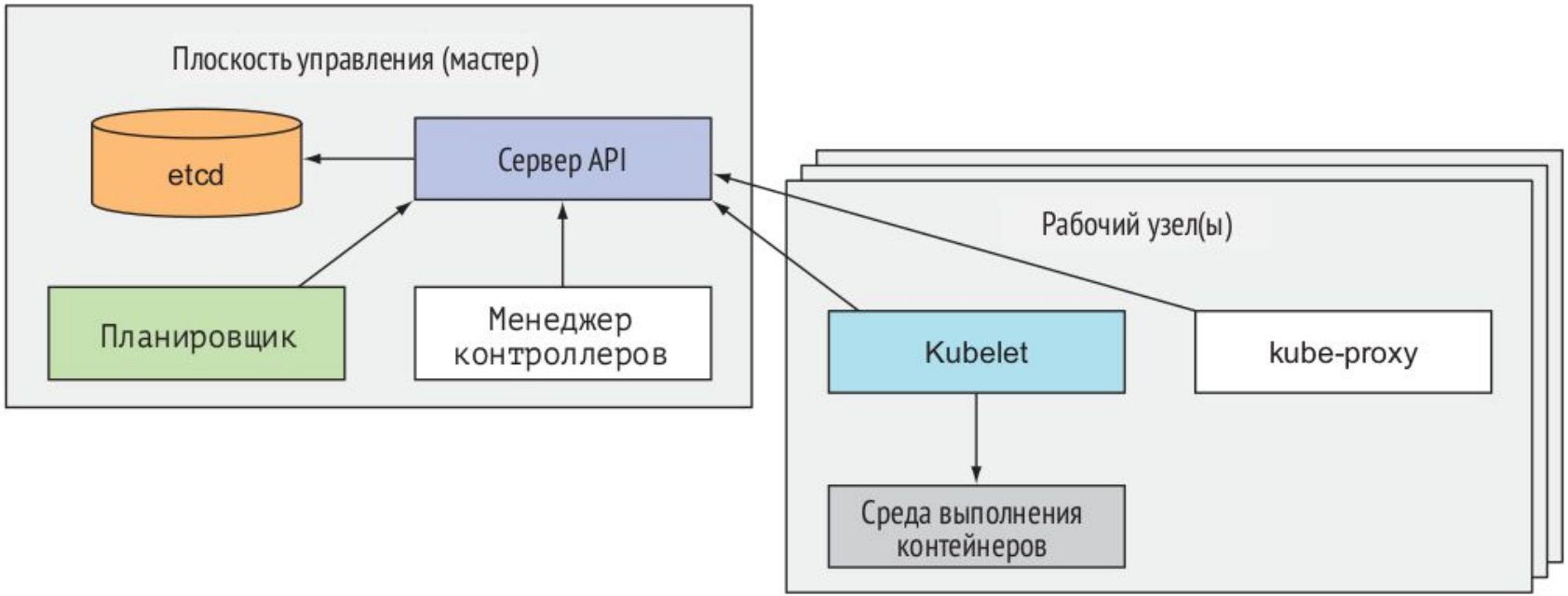
Ставим “–”,
если вопросов нет

Компоненты Kubernetes

Компоненты Kubernetes

- **Кластер** - набор машин (узлы)
- **Узлы** запускают контейнеризированные приложения
- **Под (модуль)** - группа из одного или нескольких тесно связанных контейнеров, которые всегда будут выполняться вместе на одном узле
- **Поды** размещаются в узлах
- **Плоскость управления** управляет рабочими узлами и подами в кластере
- **Компоненты панели управления** могут быть запущены на любом узле

Компоненты Kubernetes



Плоскость управления

- `kube-apiserver` - клиентская часть панели управления
- `etcd` - хранилище конфигурации кластера
- `kube-scheduler` распределяет приложения (назначает рабочий узел)
- `kube-controller-manager` - функции кластерного уровня:
  - репликация компонентов
  - отслеживание рабочих узлов
  - обработка аварийных сбоев

Компоненты узла

`kubelet` - агент, управляет контейнерами

`kube-proxy` - сетевой прокси, балансирует нагрузку сетевого трафика

Среда выполнения контейнера:

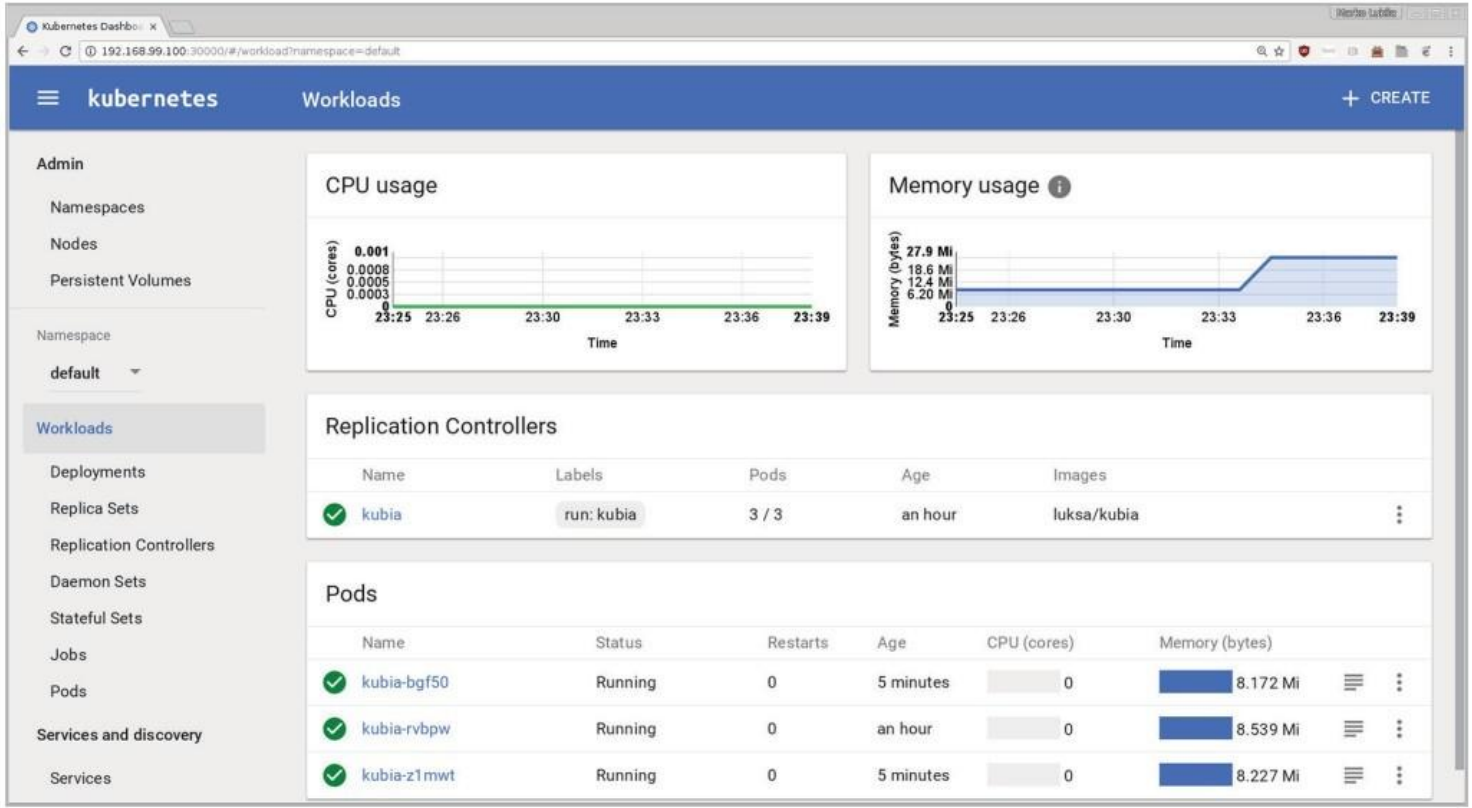
- Docker
- rkt
- containerd
- CRI-O
- Kubernetes
- CRI (Container Runtime Interface)

Дополнения

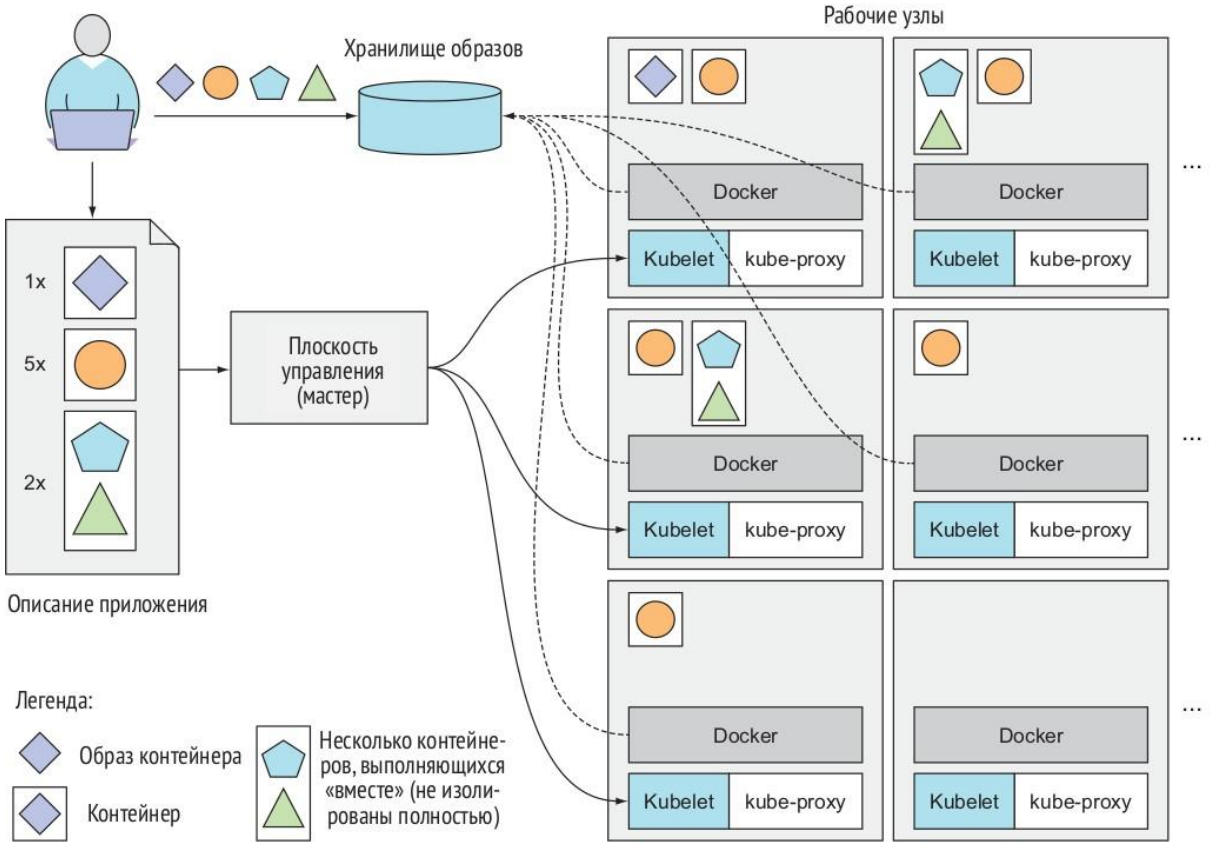
Дополнения используют ресурсы Kubernetes для расширения функциональности кластера

- **Кластерный DNS** - DNS-сервер наряду с другими DNS-серверами в вашем окружении, который обновляет DNS-записи для сервисов Kubernetes
- **Веб-интерфейс (Dashboard)** - универсальный веб-интерфейс для кластеров Kubernetes
- **Мониторинг ресурсов контейнера** записывает общие метрики о контейнерах в виде временных рядов в центральной базе данных
- **Логирование кластера** отвечает за сохранение логов контейнера в централизованном хранилище логов

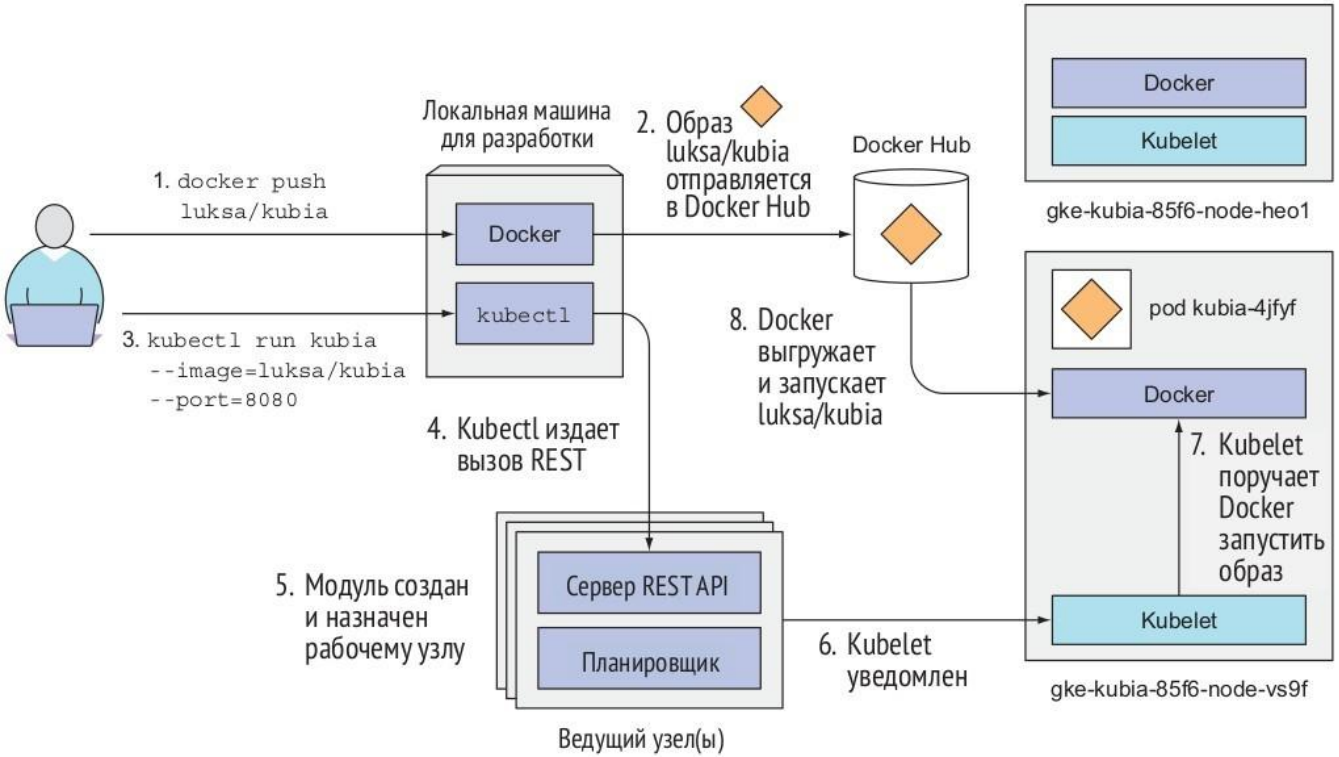
Dashboard


kubernetes
Workloads
CREATE
Admin
Namespaces
Nodes
Persistent Volumes
Namespace
default
Workloads
Deployments
Replica Sets
Replication Controllers
Daemon Sets
Stateful Sets
Jobs
Pods
Services and discovery
Services
CPU usage
CPU (cores)
0.001
0.0008
0.0005
0.0003
0
23:25 23:26 23:30 23:33 23:36 23:39
Time
Memory usage
Memory (bytes)
27.9 Mi
18.6 Mi
12.4 Mi
6.20 Mi
0
23:25 23:26 23:30 23:33 23:36 23:39
Time
Replication Controllers
Name Labels Pods Age Images
kubia run: kubia 3 / 3 an hour luksa/kubia
Pods
Name Status Restarts Age CPU (cores) Memory (bytes)
kubia-bgf50 Running 0 5 minutes 0 8.172 Mi
kubia-rvbpw Running 0 an hour 0 8.539 Mi
kubia-z1mwt Running 0 5 minutes 0 8.227 Mi

Запуск приложения



Запуск приложения



Вопросы?

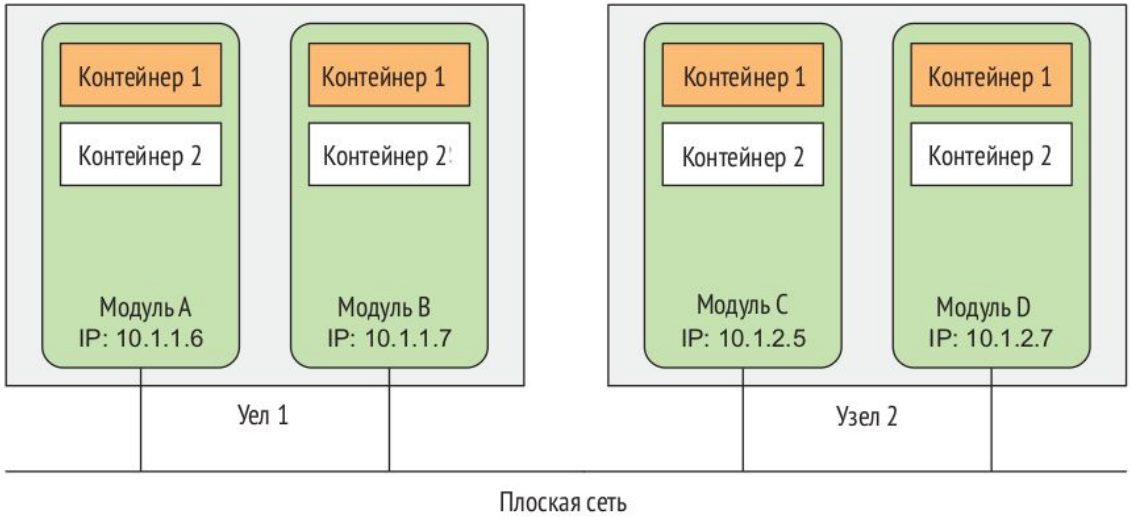
Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

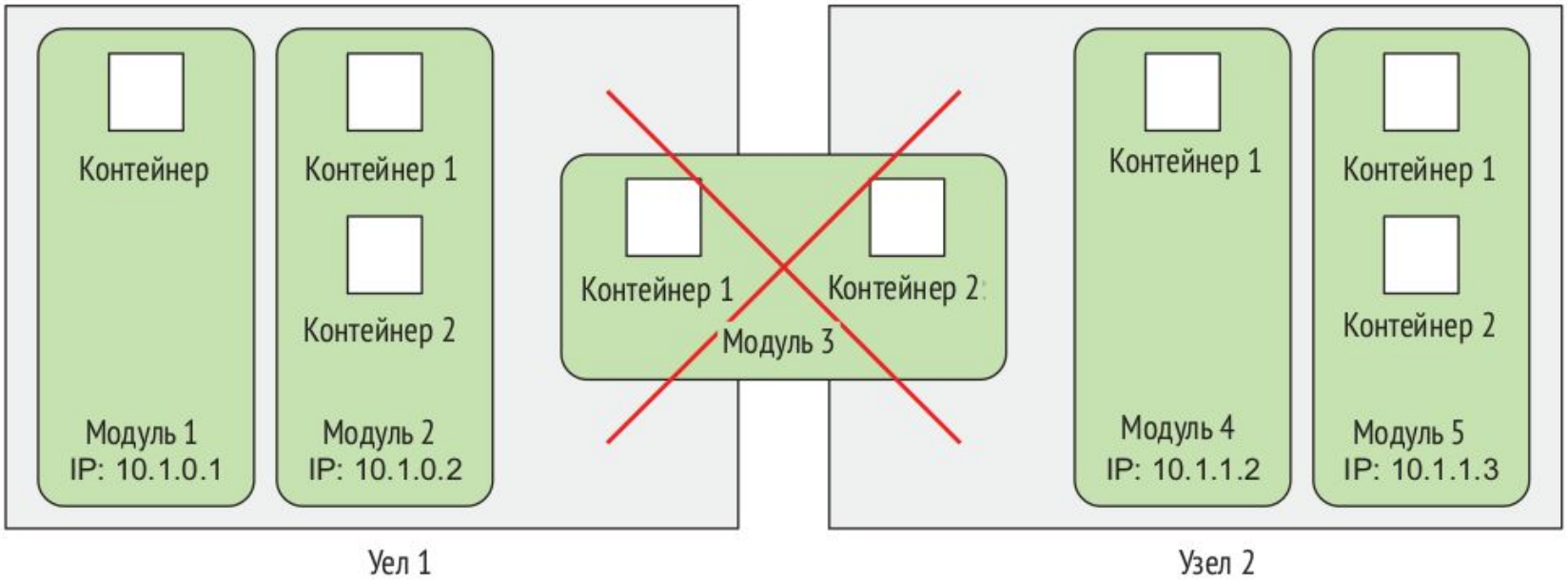
Модули (Pods)

Знакомство с модулями

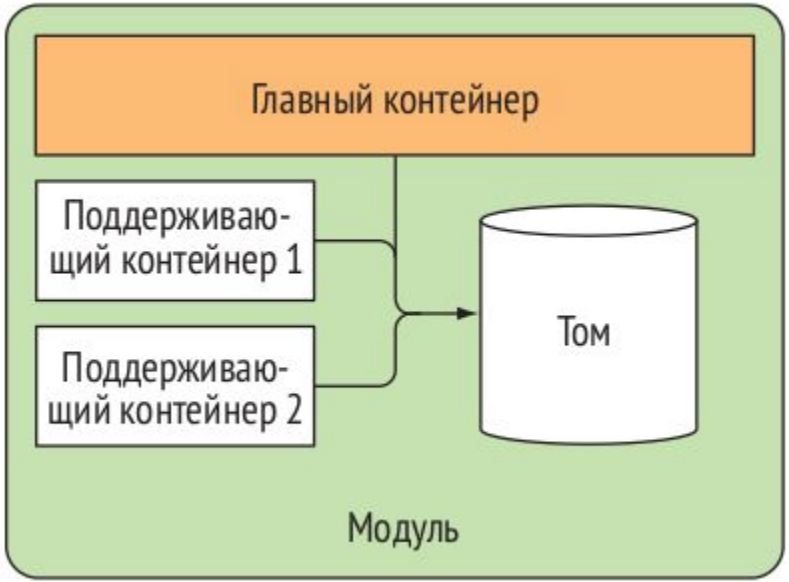
- **Модуль** - это группа из одного или нескольких тесно связанных контейнеров
- Контейнеры внутри модуля используют один набор пространства имён
- Могут взаимодействовать по IPC
- Используют одно пространство IP-адресов и портов



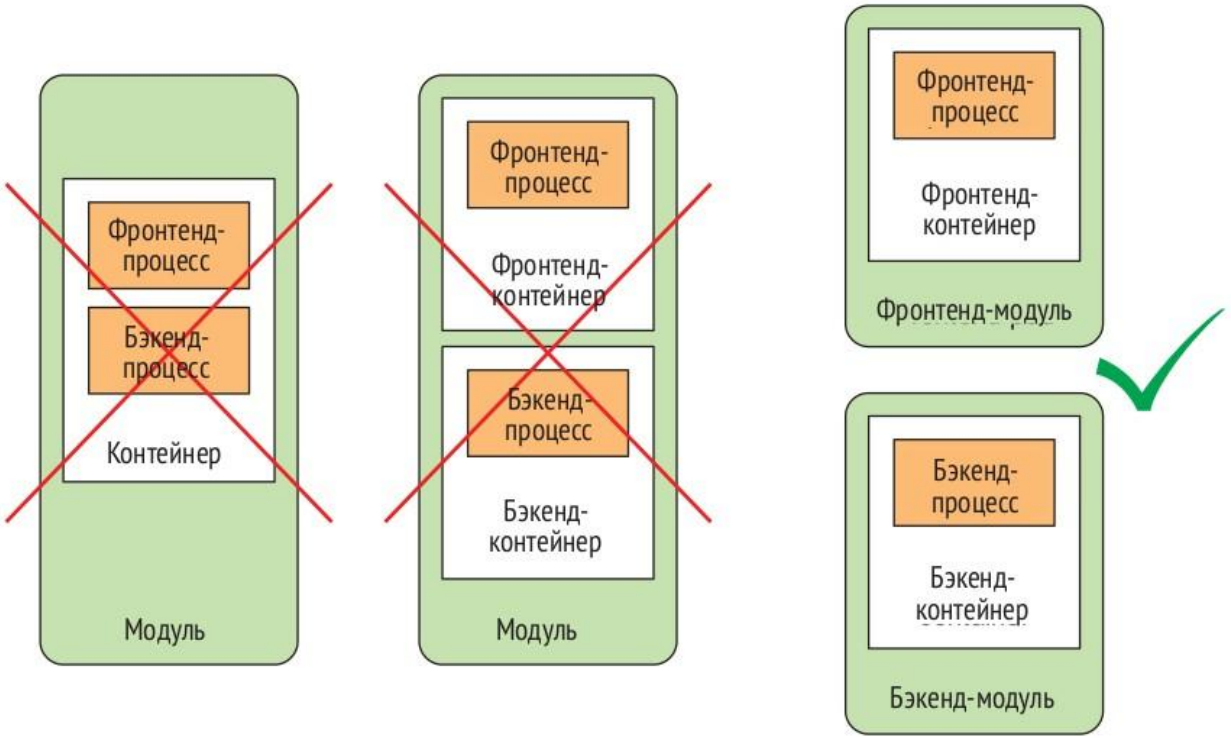
Все контейнеры модуля работают на одном узле



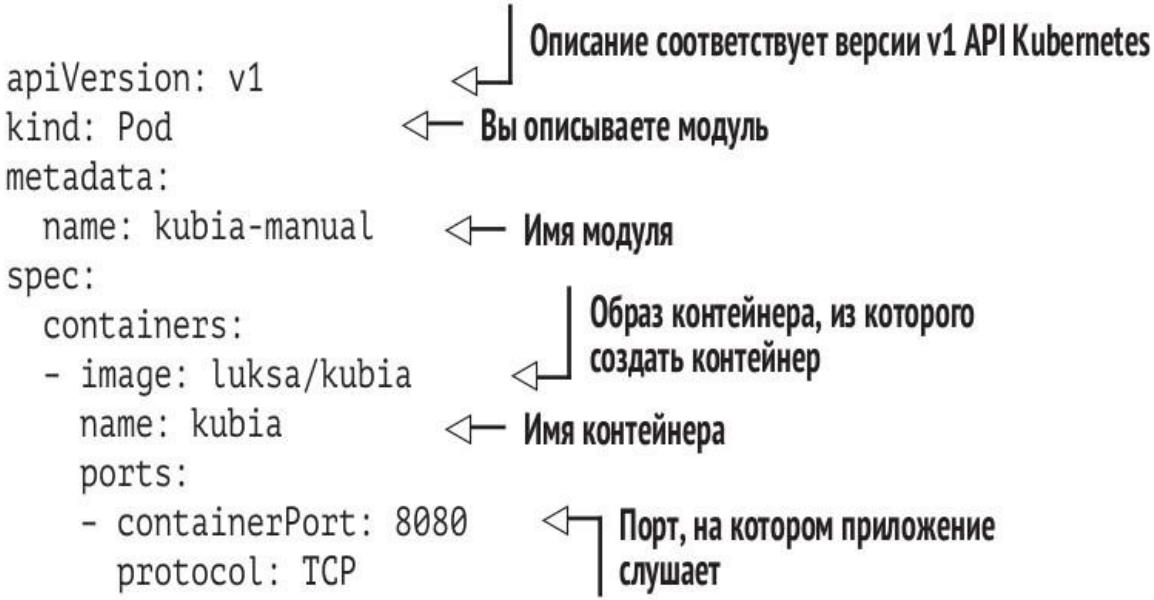
Модули должны содержать тесно связанные контейнеры



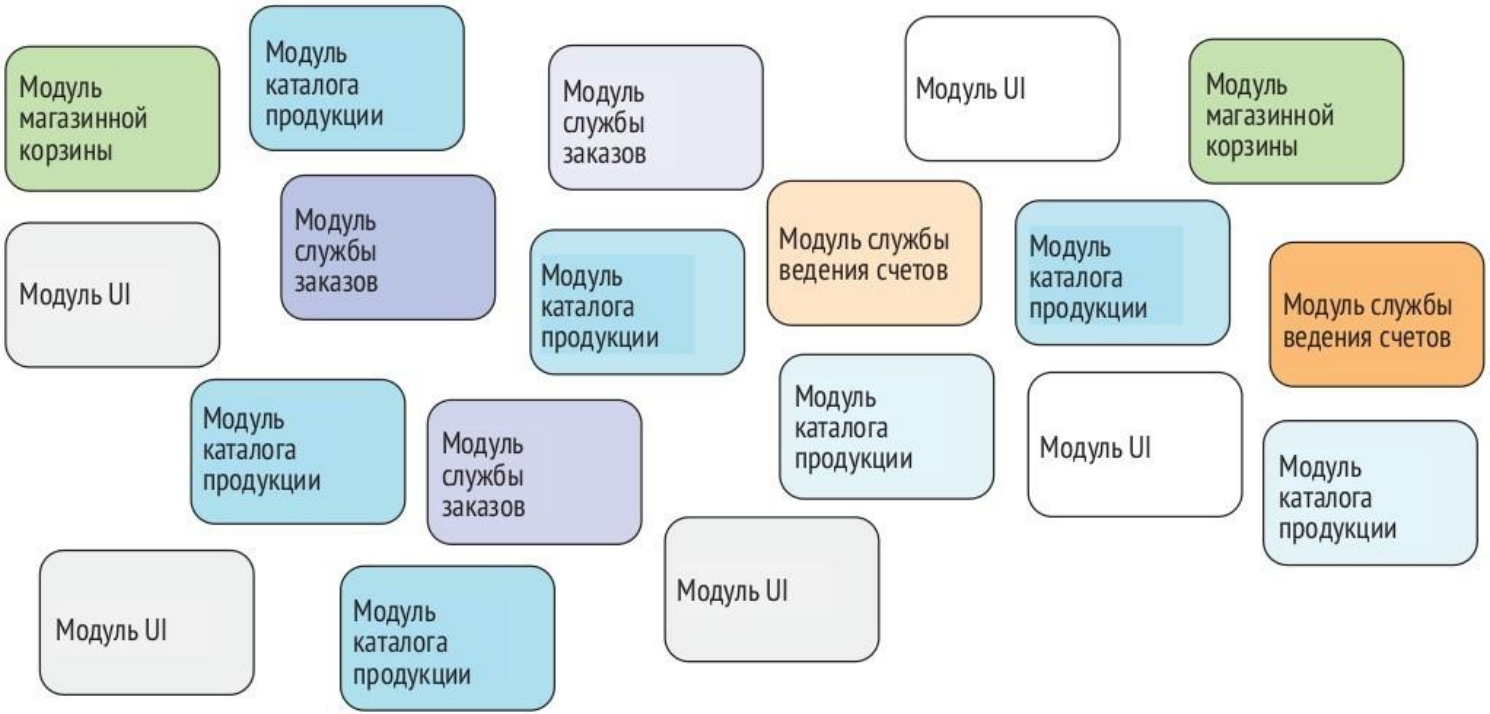
Использование нескольких контейнеров в модуле



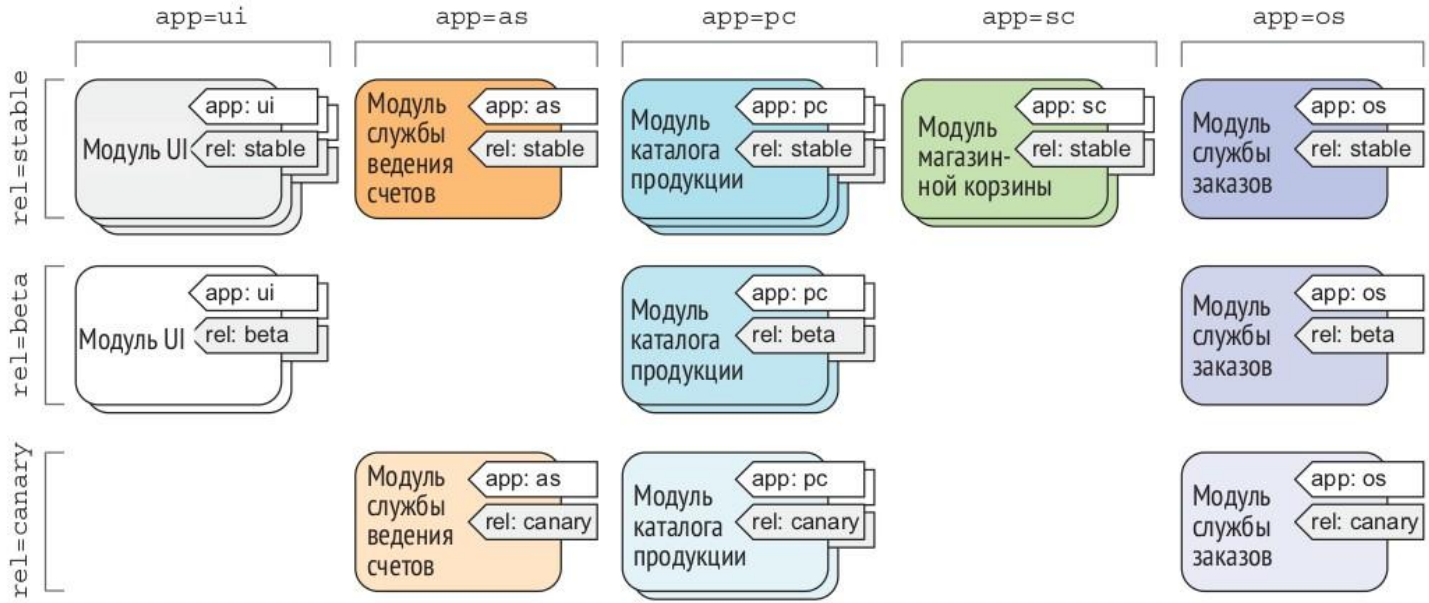
Простое описание модуля



Организация модулей с помощью меток



Организация модулей с помощью меток



Указание меток при создании модуля

```
apiVersion: v1
kind: Pod
metadata:
  name: kubia-manual-v2
  labels:
    creation_method: manual
    env: prod
spec:
  containers:
  - image: luksa/kubia
  name: kubia
  ports:
  - containerPort: 8080
    protocol: TCP
```

Вопросы?

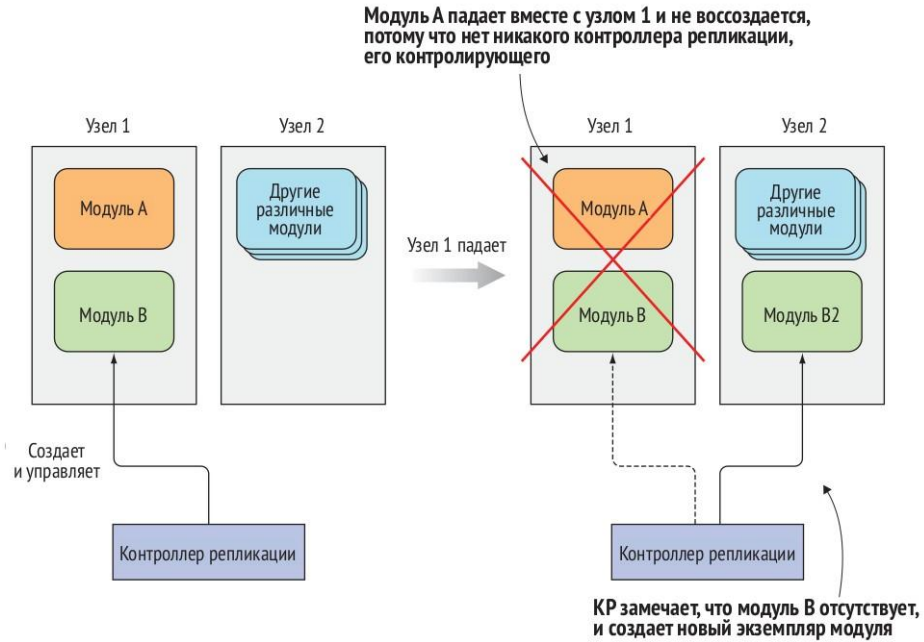
Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Контроллер репликации

Знакомство с контроллерами репликации

Контроллер репликации (ReplicationController) - это ресурс, который обеспечивает поддержание постоянной работы модулей

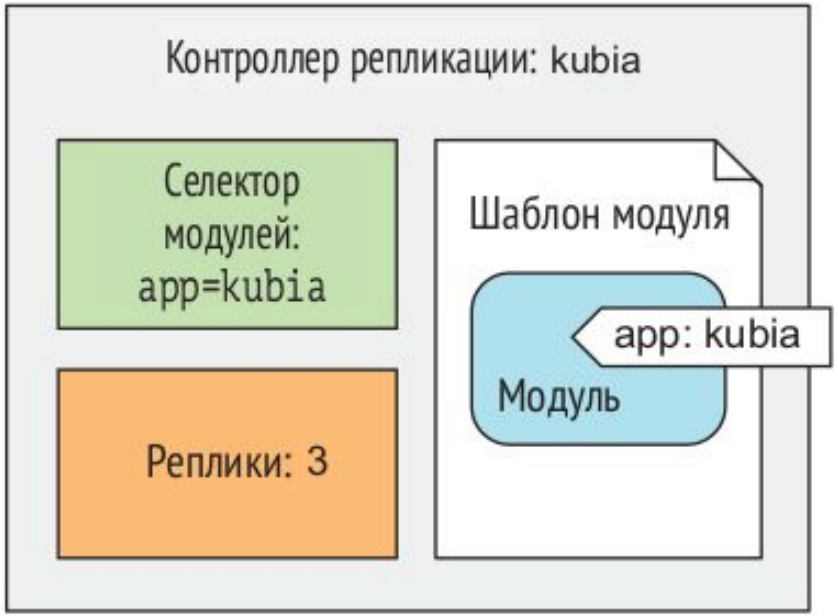


Три части контроллера репликации

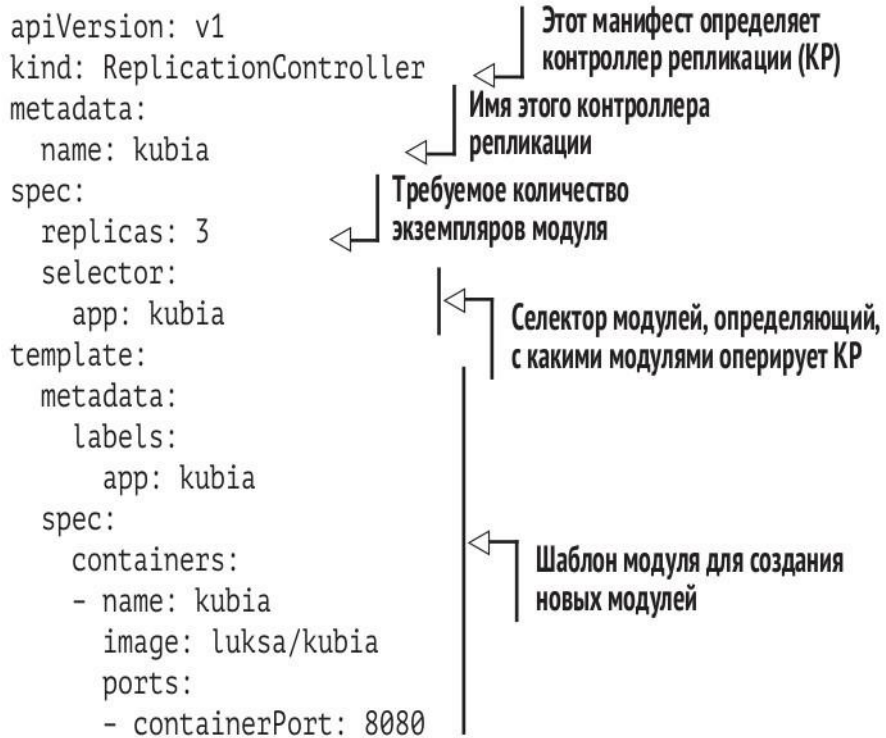
Контроллер репликации состоит из трёх основных частей:

- *селектор меток* - определяет какие модули находятся в области действия контроллера репликации
- *количество реплик* - требуемое количество модулей, которые должны быть запущены
- *шаблон модуля* - модуль, используемый при создании реплик

Три части контроллера репликации



Создание контроллера репликации



Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Развёртывания (Deployment)

Знакомство с Развёртываниями

Развёртывание (Deployment) - это декларативное описание приложений

Создание Deployment

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: kubia
spec:
  replicas: 3
  template:
    metadata:
      name: kubia
      labels:
        app: kubia
    spec:
      containers:
      - image: luksa/kubia:v1
      name: nodejs
```

Развертывание в группе API apps, версии v1beta1

Вы изменили вид с контроллера репликации на развертывание

Нет необходимости включать версию в имя развертывания

Вопросы?

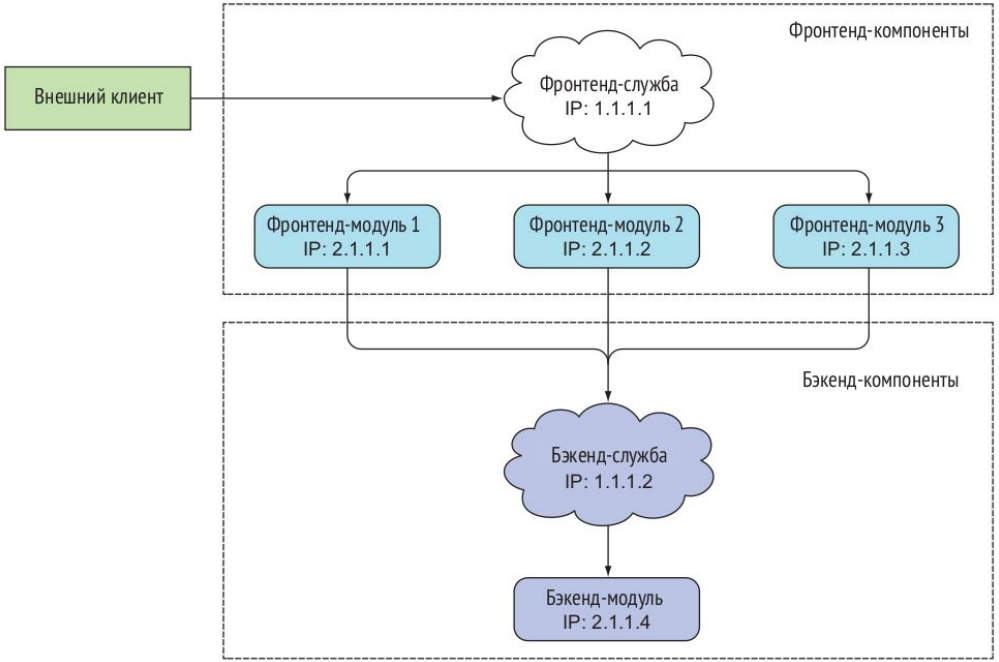
Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

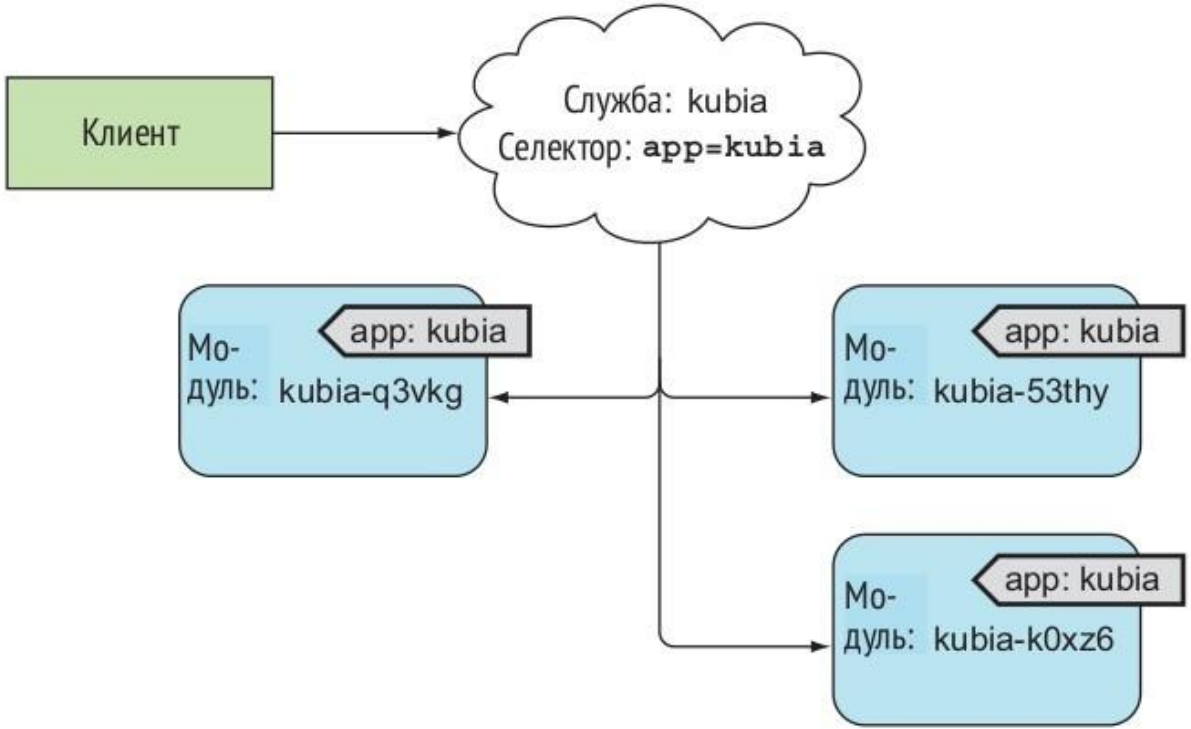
Службы (Service)

Знакомство со Службами

Служба (Service) - это ресурс, который формирует единую постоянную точку входа в группу модулей, представляющих одну и ту же службу



Использование селекторов меток



Создание службы

```
apiVersion: v1
kind: Service
metadata:
  name: kubia
spec:
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: kubia
```

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Model-as-a-Service

Flask приложение

```python
from flask import Flask, jsonify, request
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

app = Flask(__name__)


def train_model():
    iris_df = datasets.load_iris()
    data = iris_df.data
    target = iris_df.target
    target_names = iris_df.target_names
    train_data, test_data, train_target, test_target = train_test_split(data, target, test_size=0.3)
    dt = DecisionTreeClassifier().fit(train_data, train_target)
    acc = accuracy_score(test_target, dt.predict(test_data))
    return dt, acc, target_names


model, accuracy, names = train_model()
```

Flask приложение (продолжение)

```python
@app.route('/predict', methods=['POST'])
def predict():
    posted_data = request.get_json()
    sepal_length = posted_data['sepal_length']
    sepal_width = posted_data['sepal_width']
    petal_length = posted_data['petal_length']
    petal_width = posted_data['petal_width']
    prediction = model.predict([[sepal_length, sepal_width, petal_length, petal_width]])[0]
    return jsonify({'class': names[prediction]})


@app.route('/model')
def get_model():
    return jsonify({'name': 'Decision Tree Classifier',
                    'accuracy': accuracy})


if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

Dockerfile

```
FROM photon
RUN tdnf install -y python3-pip libstdc++-devel
ENV FLASK_ENV=development
RUN mkdir -p /app
COPY app.py requirements.txt /app/
WORKDIR /app
RUN pip3 install --no-cache-dir -r requirements.txt
EXPOSE 5000
ENTRYPOINT ["python3"]
CMD ["app.py"]
```

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Deploy в Kubernetes

minikube, "ручной" режим

- minikube start
- minikube addons configure registry-creds
- kubectl run iris --image=<name>/iris
- kubectl create deployment iris --image=<name>/iris
- kubectl expose deployment iris --type=LoadBalancer --port=5000
- minikube tunnel
- kubectl get service
- minikube stop

minikube, "декларативный" режим

- kubectl apply -f deployment.yml
- kubectl apply - f service.yml
- minikube tunnel
- kubectl get service

Google Cloud Platform

- docker tag iris gcr.io/bigdata-306618/iris
- docker push gcr.io/bigdata-306618/iris
- gcloud container clusters create cluster-1 --num-nodes=1
- kubectl create deployment iris --image=gcr.io/bigdata-306618/iris
- kubectl expose deployment iris --type LoadBalancer --port=5000
- kubectl get pods
- kubectl get service iris
- kubectl delete service iris
- gcloud container clusters delete cluster-1

AWS


Write code
Write and package code as a Docker image
Amazon ECR
Compress, encrypt, and control access to images
Version, tag, and manage image lifecycles
Run containers
Pull images and run containers anywhere
Amazon ECS
Amazon EKS
AWS Cloud
On premises

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Литература

Список литературы

- Kubernetes в действии
  https://dmkpress.com/catalog/computer/os/978-5-97060-657-5/
- Введение в технологии контейнеров и Kubernetes
  https://dmkpress.com/catalog/computer/os/978-5-97060-775-6/
- Осваиваем Kubernetes. Оркестрация контейнерных архитектур
  https://www.piter.com/collection/all/product/osvaivaem-kubernetes-orkestratsiya-konteynernyh-arhitektur
- Kubernetes: Лучшие практики
  https://www.piter.com/collection/all/product/kubernetes-luchshie-praktiki
- Паттерны Kubernetes: Шаблоны разработки собственных облачных приложений
  https://www.piter.com/collection/all/product/patterny-kubernetes-shablony-razrabotki-sobstvennyh-oblachnyh-prilozheniy
- Kubernetes для DevOps: развертывание, запуск и масштабирование в облаке
  https://www.piter.com/product_by_id/169496565

Ссылки

1. Kubernetes https://kubernetes.io/ru/
2. Minikube https://kubernetes.io/ru/docs/tasks/tools/install-minikube/
3. VMware Tanzu Community Edition https://tanzucommunityedition.io/
4. Red Hat OpenShift https://www.redhat.com/en/technologies/cloud-computing/openshift
5. VMware Tanzu https://tanzu.vmware.com/
6. Octant https://octant.dev/
7. KubeAcademy https://kube.academy/

Рефлексия

Рефлексия

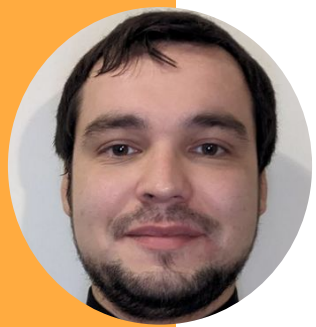
С какими впечатлениями уходите с вебинара?

Как будете применять на практике то,
что узнали на вебинаре?

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Спасибо за внимание!

Приходите на следующие вебинары



Гайнуллин Дмитрий

Machine Learning Engineer в AIC

- Разработка моделей для распознавания речи
- Прогнозирование ключевых метрик
- Развертывание моделей в продакшене