

Подготовка к выполнению ДЗ

Описание

Для успешного выполнения последующих домашних заданий вам потребуется выполнить несколько действий:

- Склонируйте репозиторий созданный специально для вас:

Организация с репозиториями доступна по [ссылке](#)

```
1 git clone **login_platform** ...
```

Описание

- Заведите feature branch с именем kubernetes-prepare:

```
1 git checkout -b kubernetes-prepare
```

- Создайте директорию .github и .github/workflows;
- В директорию .github/workflows копируете файлы: **auto-assign.yml**, **labeler.yml** и **run-tests.yml**;
- В директорию .github/ копируете файл: **auto_assign.yml** (обратите внимание на нижнее подчеркивание);
- В директорию .github/ копируете файл **labeler.yml**;
- В директорию .github/ копируете файл **PULL_REQUEST_TEMPLATE.md** - шаблон для описания ваших PR.

Итоговая структура следующая:

```
1  .github
2  |— PULL_REQUEST_TEMPLATE.md
3  |— auto_assign.yml
4  |— labeler.yml
5  |— workflows
6      |— auto-assign.yml
7      |— labeler.yml
8      |— run-tests.yml
```

Настройка Github actions для запуска тестов

- Коммитим изменения в ГИТ:

```
1 git add . ;  
2 git commit -m "add ga files" ;  
3 git push ;
```

- Создаем Pull Request к ветке **master**:

The screenshot shows a GitHub Pull Request (PR) titled "Add GA #2". The PR is open and shows a merge of 1 commit into the `main` branch from the `github-actions` branch. The PR is created by user `Jasstkn`. The PR description is "No description provided." The PR is assigned to `express42-bot`. The PR is in the "Homeworks" project, with the label "Очередь проверки". The PR has 1 commit, 1 check, and 3 files changed. The PR is in the "Homeworks" project, with the label "Очередь проверки". The PR has 1 commit, 1 check, and 3 files changed. The PR is in the "Homeworks" project, with the label "Очередь проверки".

Add GA #2 Edit Open with ▾

Open Jasstkn wants to merge 1 commit into `main` from `github-actions`

Conversation 0 Commits 1 Checks 1 Files changed 3 +41 -0

Jasstkn commented 18 seconds ago Member

No description provided.

Add GA Verified ✓ e658325

express42-bot added this to Очередь проверки in Homeworks 14 seconds ago

github-actions bot assigned **express42-bot** now

Add more commits by pushing to the `github-actions` branch on `Otus-DevOps-2020-11/vitaliygut_microservices`.

Review required
At least 1 approving review is required by reviewers with write access. [Learn more](#).

All checks have passed
2 successful checks [Show all checks](#)

Merging is blocked
Merging can be performed automatically with 1 approving review.

Reviewers
No reviews—at least 1 approving review is required.
Still in progress? Convert to draft

Assignees
express42-bot

Labels
None yet

Projects
Homeworks
Очередь проверки ▾

Milestone
No milestone

Linked issues
Successfully merging this pull request may close these issues.
None yet

Настройка локального окружения. Запуск первого контейнера. Работа с kubectl

Установка kubectl

kubectl - консольная утилита для управления кластерами Kubernetes.

Установим последнюю доступную версию kubectl на локальную машину.
Инструкции по установке доступны по [ссылке](#).

Не забудьте настроить [автодополнение](#) для используемого shell

Установка Minikube

Minikube - наиболее универсальный вариант для развертывания локального окружения.

Установим последнюю доступную версию Minikube на локальную машину. Инструкции по установке доступны по [ссылке](#).

Minikube - только один из способов быстрого получения локального окружения. В домашних работах мы также будем использовать **kind** там, где это необходимо.

Запуск Minikube

После установки запустим виртуальную машину с кластером Kubernetes командой `minikube start`.

В зависимости от используемой операционной системы для успешного старта могут потребоваться дополнительные флаги.

Запуск Minikube

После запуска, Minikube должен автоматически настроить kubectl и создать контекст minikube. Посмотреть текущую конфигурацию kubectl можно командой `kubectl config view`.

Проверим, что подключение к кластеру работает корректно:

```
1 kubectl cluster-info
```

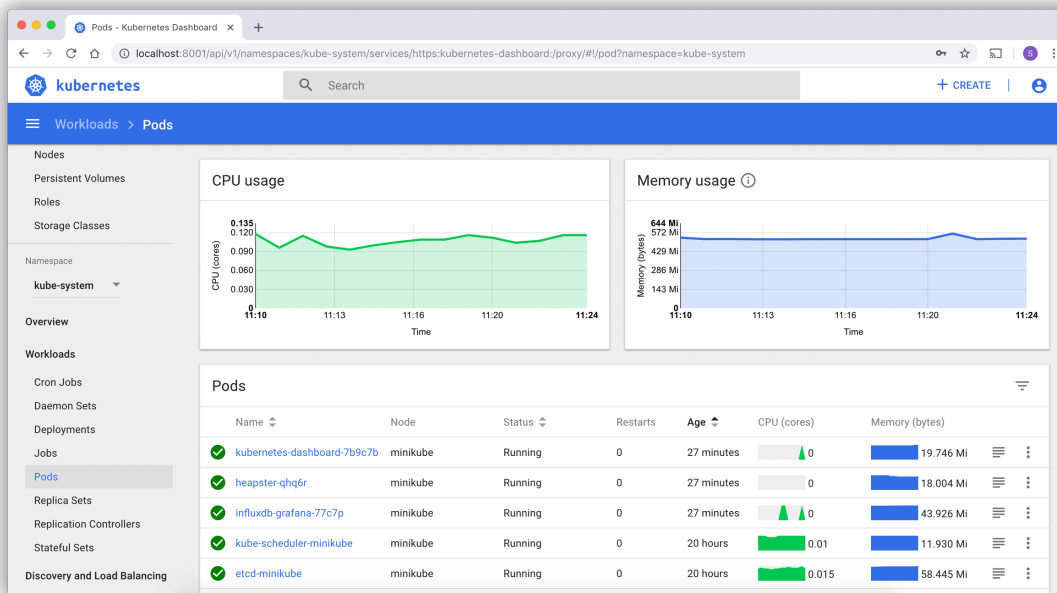
Вывод должен выглядеть следующим образом:

```
1 Kubernetes master is running at https://192.168.99.100:8443
2 KubeDNS is running at https://192.168.99.100:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
```

Kubernetes Dashboard

Один из наиболее часто устанавливаемых аддонов для Kubernetes - Dashboard.

В курсе мы будем взаимодействовать с кластером другими способами, но при желании Dashboard можно подключить как Addon в Minikube и изучить его возможности.



Удобный способ визуализации консольной работы с кластером - k9s

```


Context: minikube
Cluster: minikube
User: minikube
K9s Rev: dev
K8s Rev: v1.17.3
CPU: 5%
MEM: 17%

<0> all
<1> kube-system
<2> default

<a> Attach
<ctrl-d> Delete
<d> Describe
<e> Edit
<ctrl-k> Kill
<l> Logs

<ctrl-j> Logs (jq)
<ctrl-l> Logs <Stern>
<shift-l> Logs Previous
<shift-f> Port-Forward
<s> Shell
<y> YAML

```



```

-- Pods(all)[23] --
NAMESPACE NAME READY RESTART STATUS CPU MEM %CPU/R %MEM/R %CPU/L %MEM/L IP NODE
default hello-1582785780-lsrt 0/1 0 Completed n/a n/a n/a n/a n/a n/a 172.17.0.12 minikube
default hello-1582785900-4zbf 0/1 0 Completed n/a n/a n/a n/a n/a n/a 172.17.0.12 minikube
default jaeger-5bbc8c887-cmjj 1/1 1 Running 0 7 0 3 0 3 172.17.0.11 minikube
default nginx 1/1 1 Running 0 4 0 0 0 0 172.17.0.10 minikube
default nginx-6fbdbdc48c-5kv5p 1/1 0 Running 0 2 0 28 0 14 172.17.0.15 minikube
default nginx-6fbdbdc48c-7xn7j 1/1 0 Running n/a n/a n/a n/a n/a n/a 172.17.0.7 minikube
default nginx-6fbdbdc48c-bmqj 1/1 0 Running n/a n/a n/a n/a n/a n/a 172.17.0.13 minikube
default nginx-6fbdbdc48c-jf944 1/1 0 Running n/a n/a n/a n/a n/a n/a 172.17.0.12 minikube
default nginx-6fbdbdc48c-xwjnb 1/1 0 Running 0 3 0 39 0 19 172.17.0.14 minikube
kube-system coredns-6955765f44-2pkvx 1/1 1 Running 3 7 3 10 0 4 172.17.0.2 minikube
kube-system coredns-6955765f44-wr88k 1/1 1 Running 3 7 3 10 0 4 172.17.0.3 minikube
kube-system etcd-minikube 1/1 1 Running 20 29 0 0 0 0 192.168.64.15 minikube
kube-system fluentd-elasticsearch-vnt25 1/1 1 Running 1 51 1 25 0 25 172.17.0.5 minikube
kube-system kube-apiserver-minikube 1/1 1 Running 47 227 18 0 0 0 192.168.64.15 minikube
kube-system kube-controller-manager-minikube 1/1 2 Running 20 35 10 0 0 0 192.168.64.15 minikube
kube-system kube-proxy-sqs9s 1/1 1 Running 0 14 0 0 0 0 192.168.64.15 minikube
kube-system kube-scheduler-minikube 1/1 2 Running 4 12 4 0 0 0 192.168.64.15 minikube
kube-system metrics-server-6754dbc9df-t8x2n 1/1 1 Running 0 13 0 0 0 0 172.17.0.8 minikube
kube-system metrics-server-6754dbc9df-tz7kh 1/1 1 Running 0 10 0 0 0 0 172.17.0.6 minikube
kube-system storage-provisioner 1/1 2 Running 0 14 0 0 0 0 192.168.64.15 minikube
kubernetes-dashboard dashboard-metrics-scraper-7b64584c5c-5tjsh 1/1 1 Running 0 5 0 0 0 0 172.17.0.4 minikube
kubernetes-dashboard kubernetes-dashboard-79d9cd965-wbzzv 1/1 1 Running 0 11 0 0 0 0 172.17.0.9 minikube

```

<pulses> <pod>

Мы не навязываем использование k9s, но рекомендуем попробовать некоторое время поработать с ним

Minikube

При установке кластера с использованием Minikube будет создана виртуальная машина в которой будут работать все системные компоненты кластера Kubernetes.

Можем убедиться в этом, зайдём на VM по SSH и посмотрим запущенные Docker контейнеры:

```
1 minikube ssh
2 docker ps
```

Проверим, что Kubernetes обладает некоторой устойчивостью к отказам, удалим все контейнеры:

```
1 docker rm -f $(docker ps -a -q)
```

kubectl

Эти же компоненты, но уже в виде pod можно увидеть в namespace kube-system:

```
1 kubectl get pods -n kube-system
```

Расшифруем: данной командой мы запросили у API **вывести список** (`get`) всех **pod** (`pods`) в **namespace** (`-n`), сокращенное от `--namespace`) **kube-system**.

Можно устроить еще одну проверку на прочность и удалить все pod с системными компонентами:

```
1 kubectl delete pod --all -n kube-system
```

kubectl

Проверим, что кластер находится в рабочем состоянии, команды:

```
1 kubectl get componentstatuses
```

или сокращенно:

```
1 kubectl get cs
```

Выведут состояние системных компонентов:

1	NAME	STATUS	MESSAGE	ERROR
2	controller-manager	Healthy	ok	
3	scheduler	Healthy	ok	
4	etcd-0	Healthy	{"health":"true"}	

Задание

Разберитесь почему все pod в namespace kube-system восстановились после удаления. Укажите причину в описании PR

Hint: `core-dns` и, например, `kube-apiserver`, имеют различия в механизме запуска и восстанавливаются по разным причинам

Dockerfile

Для выполнения домашней работы необходимо создать Dockerfile, в котором будет описан образ:

1. Запускающий web-сервер на порту 8000 (можно использовать любой способ);
2. Отдающий содержимое директории `/app` внутри контейнера (например, если в директории `/app` лежит файл `homework.html`, то при запуске контейнера данный файл должен быть доступен по URL `http://localhost:8000/homework.html`);
3. Работающий с UID 1001.

Dockerfile

После того, как Dockerfile будет готов:

- Создайте новый branch `kubernetes-intro`;
- В корне репозитория создайте директорию `kubernetes-intro/web` и поместите туда готовый Dockerfile;
- Соберите из Dockerfile образ контейнера и поместите его в публичный Container Registry (например, Docker Hub).

Манифест pod

Напишем манифест `web-pod.yaml` для создания pod **web** с меткой **app** со значением **web**, содержащего один контейнер с названием **web**. Необходимо использовать ранее собранный образ с Docker Hub.

При написании манифеста можно воспользоваться следующим шаблоном:

```
1  apiVersion: v1      # Версия API
2  kind: Pod           # Объект, который создаем
3  metadata:
4    name:             # Название Pod
5    labels:           # Метки в формате key: value
6      key: value
7  spec:               # Описание Pod
8    containers:       # Описание контейнеров внутри Pod
9      - name:         # Название контейнера
10      image:         # Образ из которого создается контейнер
```

Манифест pod

Поместите манифест `web-pod.yaml` в директорию `kubernetes-intro` и примените его:

```
1 kubectl apply -f web-pod.yaml
```

После этого в кластере в namespace default должен появиться запущенный pod web:

```
1 kubectl get pods
2
3 NAME      READY   STATUS    RESTARTS   AGE
4 web       1/1     Running   0           42s
```

Манифест pod

В Kubernetes есть возможность получить манифест уже запущенного в кластере pod.

В подобном манифесте помимо описания pod будут фигурировать служебные поля (например, различные статусы) и значения, подставленные по умолчанию.

```
1 kubectl get pod web -o yaml
```

kubectl describe

Другой способ посмотреть описание pod - использовать ключ `describe`. Команда позволяет отследить текущее состояние объекта, а также события, которые с ним происходили:

```
1 kubectl describe pod web
```

kubectl describe

Успешный старт pod в `kubectl describe` выглядит следующим образом:

1. scheduler определил, на какой ноде запускать pod
2. kubelet скачал необходимый образ и запустил контейнер

```
1  Events:
2    Type      Reason      Age   From              Message
3    ----      -
4    Normal    Scheduled   10s   default-scheduler Successfully assigned default/web to minikube
5    Normal    Pulling     8s    kubelet, minikube Pulling image "web:1.0"
6    Normal    Pulled      3s    kubelet, minikube Successfully pulled image "web:1.0"
7    Normal    Created     3s    kubelet, minikube Created container nginx
8    Normal    Started     2s    kubelet, minikube Started container nginx
```

kubectl describe

При этом `kubectl describe` - хороший старт для поиска причин проблем с запуском pod.

Укажите в манифесте несуществующий тег образа web и примените его заново (`kubectl apply -f web-pod.yaml`).

Статус pod (`kubectl get pods`) должен измениться на **ErrImagePull/ImagePullBackOff**, а команда `kubectl describe pod web` поможет понять причину такого поведения:

```
1  Events:
2    Type      Reason      Age   From          Message
3    ----      -
4    Warning   Failed      2s    kubelet, minikube  Failed to pull image "web:broken-tag": rpc error:
    code = Unknown desc = Error response from daemon: manifest for web:broken-tag not found
```


kubectl describe

Вывод `kubectl describe pod web` если мы забыли, что Container Registry могут быть приватными:

```
1  Events:
2    Type      Reason      Age   From          Message
3    ----      -
4    Warning   Failed      2s    kubelet, minikube Failed to pull image "quay.io/example/web:1.0":
    rpc error: code = Unknown desc =Error response from daemon: unauthorized: access to the requested
    resource is not authorized
```

Init контейнеры

Добавим в наш pod **init контейнер**, генерирующий страницу `index.html`.

Init контейнеры описываются аналогично обычным контейнерам в pod. Добавьте в манифест `web-pod.yaml` описание init контейнера, соответствующее следующим требованиям:

- **image** init контейнера должен содержать **wget** (например, можно использовать `busybox:1.31.0` или любой другой busybox актуальной версии)
- **command** init контейнера (аналог **ENTRYPOINT** в Dockerfile) укажите следующую:

```
1 ['sh', '-c', 'wget -O- https://tinyurl.com/otus-k8s-intro | sh']
```

Volumes

Для того, чтобы файлы, созданные в **init контейнере**, были доступны основному контейнеру в pod нам понадобится использовать **volume** типа `emptyDir`.

В следующих лекциях мы расскажем про **volume** подробнее, а пока будем руководствоваться подсказками:

- У контейнера и у **init контейнера** должны быть описаны **volumeMounts** следующего вида

```
1     volumeMounts:  
2     - name: app  
3       mountPath: /app
```

Volumes

Для того, чтобы файлы, созданные в init контейнере, были доступны основному контейнеру в pod нам понадобится использовать **volume** типа `emptyDir`.

В следующих лекциях мы расскажем про **volume** подробнее, а пока будем руководствоваться подсказками:

- **volume** должен быть описан в спецификации pod

```
1  volumes:
2    - name: app
3      emptyDir: {}
```

Запуск pod

Удалите запущенный pod web из кластера (`kubectl delete pod web`) и примените обновленный манифест `web-pod.yaml`

Отслеживать происходящее можно с использованием команды `kubectl get pods -w`

Должен получиться аналогичный вывод:

```
1 kubectl apply -f web-pod.yaml && kubectl get pods -w
2
3 pod/web created
4 NAME    READY   STATUS             RESTARTS   AGE
5 web     0/1     Init:0/1           0           0s
6 web     0/1     Init:0/1           0           1s
7 web     0/1     PodInitializing    0           2s
8 web     1/1     Running            0           3s
```

Проверка работы приложения

Проверим работоспособность web сервера. Существует несколько способов получить доступ к pod, запущенным внутри кластера.

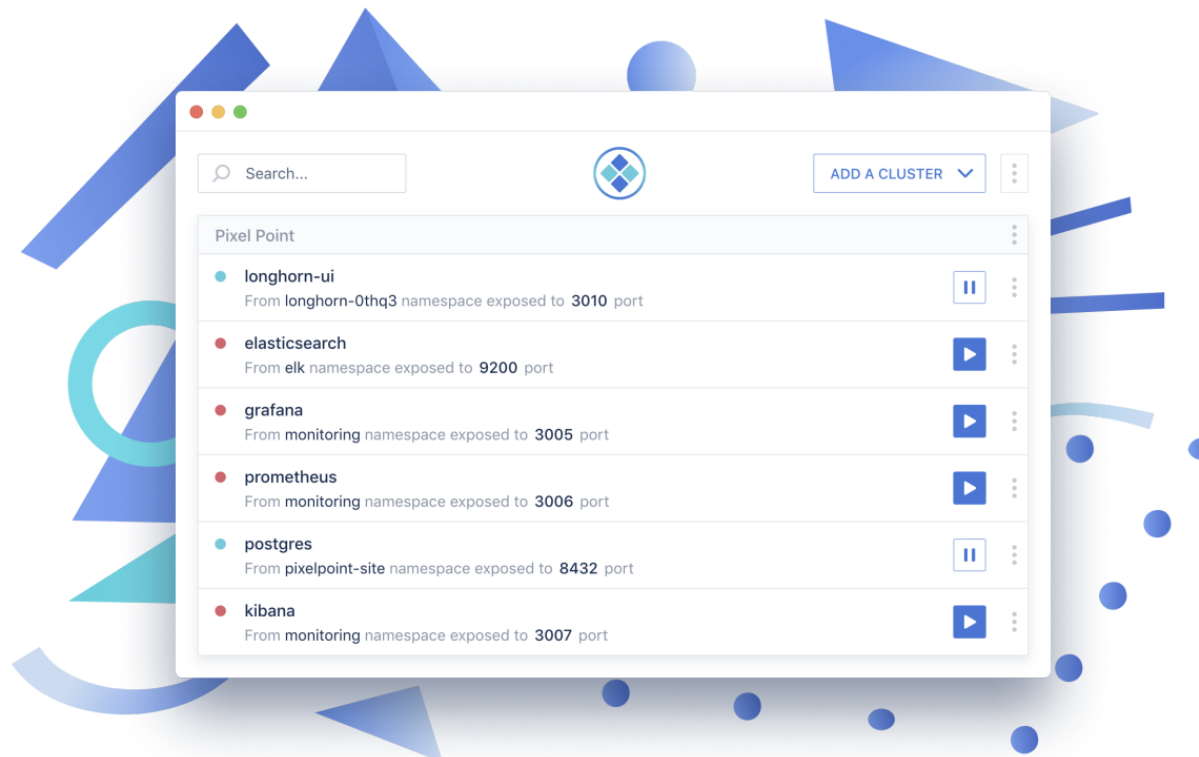
Мы воспользуемся командой `kubectl port-forward`

```
1 kubectl port-forward --address 0.0.0.0 pod/web 8000:8000
```

Если все выполнено правильно, на локальном компьютере по ссылке <http://localhost:8000/index.html> должна открыться страница.

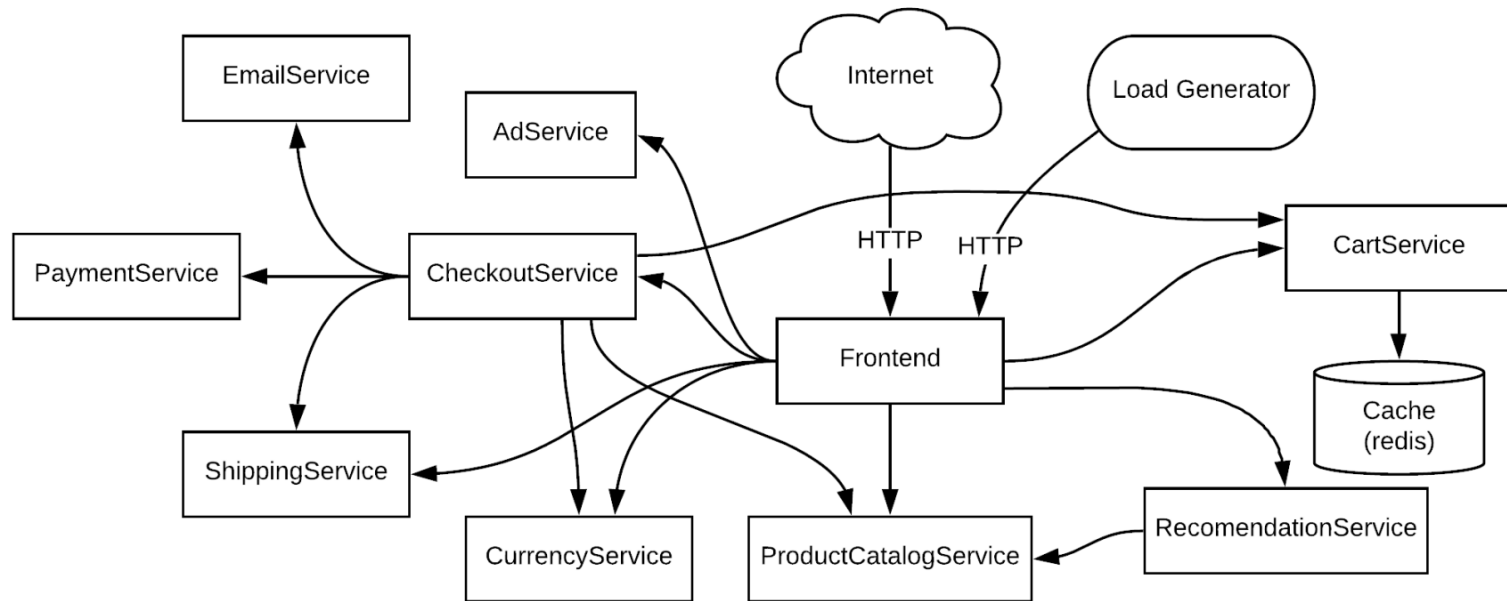
kube-forwarder

В качестве альтернативы `kubectl port-forward` можно использовать удобную обертку `kube-forwarder`. Она отлично подходит для доступа к pod внутри кластера с локальной машины во время разработки продукта.



Hipster Shop

В последующих домашних заданиях мы будем использовать микросервисное приложение **Hipster Shop**



Hipster Shop

Давайте познакомимся с приложением поближе и попробуем запустить внутри нашего кластера его компоненты.

Начнем с микросервиса `frontend`. Его исходный код доступен по адресу.

- Склонируйте **репозиторий** и соберите собственный образ для `frontend` (используйте готовый Dockerfile)
- Поместите собранный образ на Docker Hub

! Не добавляйте содержимое репозитория `microservices-demo` в ваш PR

Hipster Shop

Рассмотрим альтернативный способ запуска pod в нашем Kubernetes кластере.

Вы уже умеете работать с манифестами (и это наиболее корректный подход к развертыванию ресурсов в Kubernetes), но иногда бывает удобно использовать ad-hoc режим и возможности Kubectl для создания ресурсов.

Hipster Shop

Разберем пример для запуска `frontend` pod:

```
1 kubectl run frontend --image avtandilko/hipster-frontend:v0.0.1 --restart=Never
```

- **kubectl run** - запустить ресурс
- **frontend** - с именем `frontend`
- **--image** - из образа `avtandilko/hipster-frontend:v0.0.1` (подставьте свой образ)
- **--restart=Never** указываем на то, что в качестве ресурса запускаем `pod`.

Подробности

Hipster Shop

Один из распространенных кейсов использования ad-hoc режима - генерация манифестов средствами kubectl:

```
1 kubectl run frontend --image avtandilko/hipster-frontend:v0.0.1 --restart=Never --dry-run -o yaml  
  > frontend-pod.yaml
```

Рассмотрим дополнительные ключи:

- **–dry-run** - вывод информации о ресурсе без его реального создания
- **-o yaml** - форматирование вывода в YAML
- **> frontend-pod.yaml** - перенаправление вывода в файл

Hipster Shop | Задание со

- Выясните причину, по которой pod `frontend` находится в статусе `Error`
- Создайте новый манифест `frontend-pod-healthy.yaml`. При его применении ошибка должна исчезнуть. Подсказки можно найти:
 - В логах - `kubectl logs frontend`
 - В манифесте по [ссылке](#)
- В результате, после применения исправленного манифеста pod `frontend` должен находиться в статусе `Running` (опустим вопрос, действительно ли микросервис работает)
- Поместите исправленный манифест `frontend-pod-healthy.yaml` в директорию `kubernetes-intro`

Проверка ДЗ

- Результаты вашей работы должны быть добавлены в ветку **kubernetes-intro** вашего GitHub репозитория **login_platform**;
- В **README.md** нужно внести описание проделанной работы;
- Необходимо заполнить описание Pull Request по шаблону, созданному в предыдущей домашней работе.

Проверка ДЗ

- Создайте Pull Request к ветке **master** (описание PR рекомендуется заполнять);
- Добавьте метку `kubernetes-intro` к вашему PR;
- После того как автоматизированные тесты проверят корректность выполнения ДЗ, необходимо сделать merge ветки **kubernetes-intro** в master и закрыть PR;
- Если у вас возникли вопросы по ДЗ и необходима консультация преподавателей - после прохождения автотестов добавьте к PR метку `Review Required` и **не мерджите PR** самостоятельно.