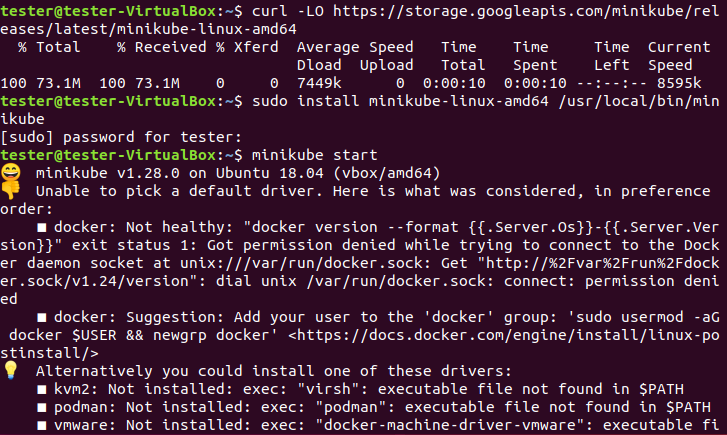
### Задание D2.4.1

(1)Установи minikube

curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

sudo install minikube-linux-amd64 /usr/local/bin/minikube



Запустим кластер

minikube start

Чтобы проверить, что кластер успешно поднялся, мы можем посмотреть статус работы Minikube и запросить имена всех работающих подов в нашем кластере.

minikube status покажет нам информацию о работе Minikube:

minikube

type: Control Plane

host: Running

kubelet: Running

apiserver: Running

kubeconfig: Configured

minikube-m02

type: Worker

host: Running

kubelet: Running

minikube-m03

type: Worker

host: Running

kubelet: Running

В выводе выше мы видим, что у нас есть три ноды:

Первая нода — Control Plane-нода или master-нода, на ней находятся управляющие компоненты. Полезная нагрузка на ней не будет размещаться.

Две другие ноды — это worker-ноды, как раз на них и будут размещаться наши приложения.

Теперь можем проверить, а есть ли вообще поды в нашем новом кластере.  
Если у нас нет на копьютере клиентской K8S-утилиты для доступа к кластеру, то Minikube нам в этом поможет.

minikube kubectl -- get po -A

С помощью этой команды мы получим вывод всех подов, которые у нас есть в нашем новом кластере, что означает, что наш кластер работает и готов к тестам.

(2)Установим *kind*

brew install kind



Создадим кластер

kind create cluster



Проверим как пробрасывается кастомный образ внутри кубернет кластера. Посмотрим есть ли поды:

k get po -A

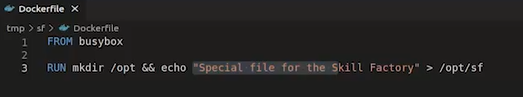


Посмотрим какие есть ноды:

k get no



Соберем кастомный образ и отправим в кубернет кластер. Для этого изменим Dockerfile, например, выведем надпись:



Соберем контейнер: docker build -t busybox-sf:1.2 .

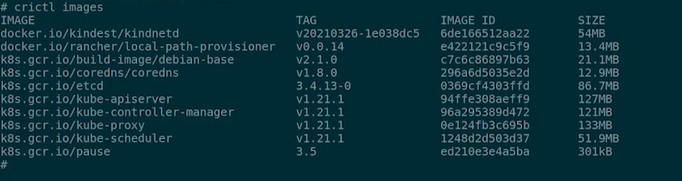
Подключимся к кубернет кластеру, предварительно посмотрим какие у нас контейнеры:

docker ps



docker exec -it deedae4b5b0 sh

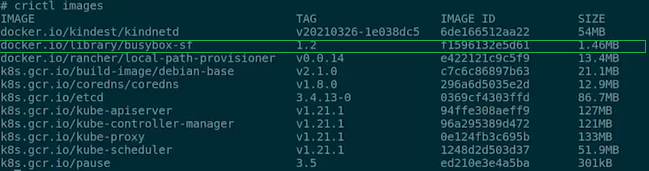
Посмотрим какие образы есть в системе, используя команду crictl images:



Наш образ busybox-sf:1.2 отсутствует.

Загрузим его с помощью команды kind load docker-image busybox-sf:1.2

Проверим появился ли он в списке с помощью команды crictl images. Появился.



Запустим под с нашим образом:

k run -it sf-demo-pod --image busybox-sf:1.2 -- ash

Для проверки можно зайти в директорию, которую указывали в докерфайле и посмотреть содержимое файла:

cd /opt/

cat sf

Увидим передаваемый текст:

