

## Отчет Лабораторная 8 (Инфраструктура больших данных)

Цель работы: Получить навыки оркестрации контейнеров с использованием Kubernetes путём миграции сервиса модели на PySpark, сервиса витрины на Spark и сервиса источника данных.

1. Скачиваем по инструкции [minikube](#) на официальном сайте.
2. Компилируем витрину на языке Scala при помощи sbt
3. Пишем докер конфигурацию для clickhouse в **Dockerfile.clickhouse**.

Внутри контейнера копируем скрипт для загрузки данных из csv файла:

```
FROM clickhouse/clickhouse-server:latest
COPY scripts/seed_db.sh /scripts/seed_db.sh
RUN sed -i 's/\r$//' /scripts/seed_db.sh && \
    chmod +x /scripts/seed_db.sh
```

4. Пишем докер конфигурацию для контейнера с витриной и моделью в **Dockerfile**. Скачиваем jar файлы с clickhouse jdbc для записи в базу и чтения из базы. Также копируем jar файл с кодом витрины из пункта 2:

```
FROM apache/spark:3.5.5-scala2.12-java11-python3-r-ubuntu
ARG SPARK_JAR_PATH=/opt/spark/jars/
WORKDIR /app
USER 0:0
ADD
https://repo1.maven.org/maven2/com/clickhouse/spark/clickhouse-spark-runtime-3.5_2.12/0.8.1/clickhouse-spark-runtime-3.5_2.12-0.8.1.jar ${SPARK_JAR_PATH}
ADD
https://repo1.maven.org/maven2/com/clickhouse/clickhouse-jdbc/0.8.5/clickhouse-jdbc-0.8.5-all.jar ${SPARK_JAR_PATH}
```

```

COPY requirements.txt .
COPY .env .
COPY src ./src
COPY conf ./conf
RUN pip install --no-cache-dir -r requirements.txt
COPY target/scala-2.12/assemblyApp.jar target/scala-2.12/
COPY entrypoint.sh .
RUN sed -i 's/\r$//' /app/entrypoint.sh && \
    chmod +x /app/entrypoint.sh
ENTRYPOINT [ "/app/entrypoint.sh" ]

```

## 5. Собираем докер изображения при помощи docker:

```

docker build -t model:latest .
docker build -t custom-clickhouse:latest -f
./Dockerfile.clickhouse .

```

## 6. Пушим изображения в DockerHub:

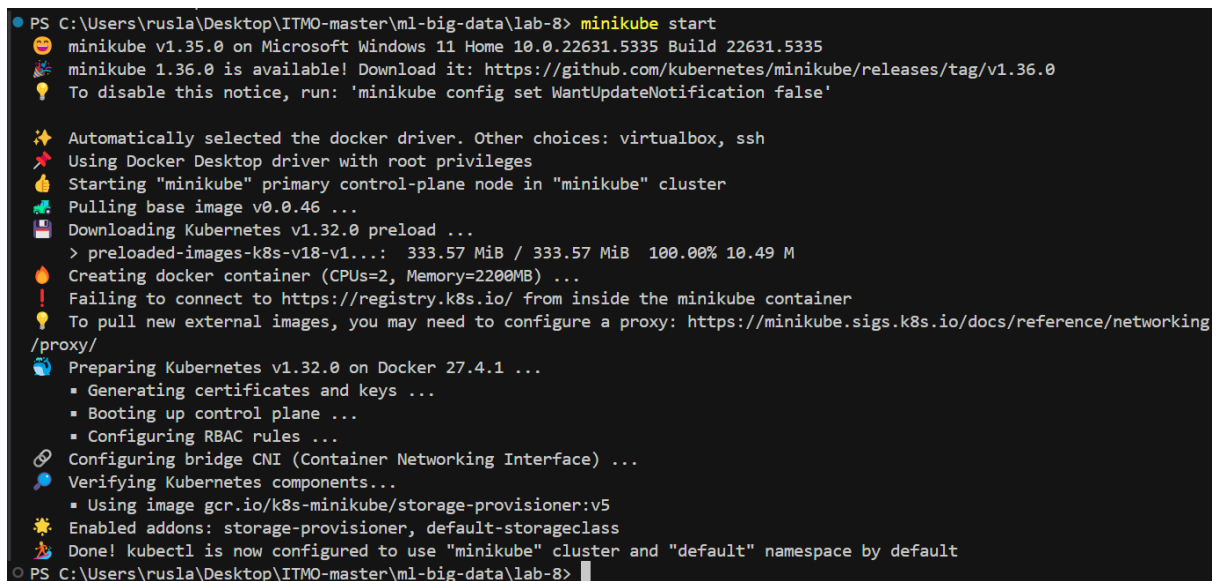
```

docker login
docker tag model:latest zarus03/model:latest
docker tag custom-clickhouse:latest
zarus03/custom-clickhouse:latest
docker push zarus03/model:latest
docker push zarus03/custom-clickhouse:latest

```

## 7. Запускаем minikube:

```
minikube start
```



```

PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> minikube start
minikube v1.35.0 on Microsoft Windows 11 Home 10.0.22631.5335 Build 22631.5335
minikube 1.36.0 is available! Download it: https://github.com/kubernetes/minikube/releases/tag/v1.36.0
To disable this notice, run: 'minikube config set WantUpdateNotification false'

✨ Automatically selected the docker driver. Other choices: virtualbox, ssh
🔧 Using Docker Desktop driver with root privileges
👍 Starting "minikube" primary control-plane node in "minikube" cluster
📦 Pulling base image v0.0.46 ...
📦 Downloading Kubernetes v1.32.0 preload ...
> preloaded-images-k8s-v18-v1...: 333.57 MiB / 333.57 MiB 100.00% 10.49 M
🔥 Creating docker container (CPUs=2, Memory=2200MB) ...
❗ Failing to connect to https://registry.k8s.io/ from inside the minikube container
💡 To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
🔧 Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
  ▪ Generating certificates and keys ...
  ▪ Booting up control plane ...
  ▪ Configuring RBAC rules ...
🔗 Configuring bridge CNI (Container Networking Interface) ...
🔍 Verifying Kubernetes components...
  ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟 Enabled addons: storage-provisioner, default-storageclass
🎉 Done! kubectrl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8>

```

## 8. Добавляем сервис для вывода затрачиваемых ресурсов подами:

```
minikube addons enable metrics-server
```

```
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> minikube addons enable metrics-server
metrics-server is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
  Using image registry.k8s.io/metrics-server/metrics-server:v0.7.2
  The 'metrics-server' addon is enabled
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8>
```

## 9. Для вывода метрик по подам используем команду:

```
kubectl top pod -n spark-app
```

```
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> kubectl top pod -n spark-app
No resources found in spark-app namespace.
```

## 10. Монтируем директорию с большим csv файлом:

```
minikube mount ./sparkdata:/sparkdata
```

```
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> minikube mount ./sparkdata:/sparkdata
Mounting host path ./sparkdata into VM as /sparkdata ...
  Mount type: 9p
  User ID: docker
  Group ID: docker
  Version: 9p2000.L
  Message Size: 262144
  Options: map[]
  Bind Address: 127.0.0.1:49817
  Userspace file server: ufs starting
  Successfully mounted ./sparkdata to /sparkdata
  NOTE: This process must stay alive for the mount to be accessible ...
```

## 11. Создаем namespace, чтобы не создавать все в default:

```
kubectl apply -f ./k8s/namespace.yaml
```

```
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> kubectl apply -f ./k8s/namespace.yaml
namespace/spark-app created
```

## 12. Выводим namespaces:

```
kubectl get namespaces
```

```
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> kubectl get namespaces
NAME                STATUS   AGE
default             Active   3m20s
kube-node-lease     Active   3m20s
kube-public         Active   3m20s
kube-system         Active   3m20s
spark-app           Active   17s
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8>
```

## 13. Создаем ConfigMap для файла конфигурации базы данных:

```
kubectl create configmap env-config --from-env-file=.env -n spark-app
```

```
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> kubectl create configmap env-config --from-env-file=.env -n spark-app
configmap/env-config created
```

```
kubectl describe configmaps env-config -n spark-app
```

```
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> kubectl describe configmaps env-config -n spark-app
Name:      env-config
Namespace: spark-app
Labels:    <none>
Annotations: <none>

Data
====
CLICKHOUSE_PORT:
-----
8123

CLICKHOUSE_PROTOCOL:
-----
http
```

14. Пишем конфигурацию Deployment k8s для базы данных clickhouse в **k8s/clickhouse-deployment.yaml**. Указываем количество реплик, идентификатор изображения из DockerHub, лимиты по памяти и вычислению, название ConfigMap. В поле command прописываем скрипт для загрузки данных из csv файла. Также для того, чтобы контейнер мог общаться по сети с другими контейнерам прописываем конфигурацию Service с портами.

15. Аналогично пишем конфигурацию Deployment k8s для модели с витриной. В конфигурации Service прописываем порт 4040, для того, чтобы можно было просматривать логи Spark приложений. Ознакомиться с файлом можно по пути **k8s/model-deployment.yaml**.

16. Просматривать логи подов будем с помощью команды. Вместо «*clickhouse-xxxx*» указываем название нужного пода:

```
kubectl logs -f clickhouse-xxxx -n spark-app
```

17. Запускаем под с базой данных командой:

```
kubectl apply -f ./k8s/clickhouse-deployment.yaml
```

```
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> kubectl apply -f ./k8s/clickhouse-deployment.yaml
deployment.apps/clickhouse created
service/clickhouse-service created
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> kubectl get pods -n spark-app
```

18. Просматриваем работающие поды командой:

```
kubectl get pods -n spark-app
```

```
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> kubectl get pods -n spark-app
NAME                                READY   STATUS             RESTARTS   AGE
clickhouse-6bf47f45b6-zx57s        0/1     ContainerCreating   0          6s
```

19. Логи пода clickhouse. Ошибки по типу Connection refused, происходят из-за того, что в скрипте загрузки данных написан цикл, который проверяет работает ли база данных. Пока поднимается база, скрипт не может подключиться, поэтому мы видим ошибки. Последнее сообщение «Adding prediction column» сообщает о том, что колонка для предсказаний добавлена, и можно запускать под с моделью:

```
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> kubectl
logs -f clickhouse-6bf47f45b6-zx57s -n spark-app
Waiting for ClickHouse to be ready...
Code: 210. DB::NetException: Connection refused (localhost:9000).
(NETWORK_ERROR)
/entrypoint.sh: create new user 'admin' instead 'default'
Processing configuration file '/etc/clickhouse-server/config.xml'.
Merging configuration file
'/etc/clickhouse-server/config.d/docker_related_config.xml'.
Logging trace to /var/log/clickhouse-server/clickhouse-server.log
Logging errors to
/var/log/clickhouse-server/clickhouse-server.err.log
Code: 210. DB::NetException: Connection refused (localhost:9000).
(NETWORK_ERROR)
Code: 210. DB::NetException: Connection refused (localhost:9000).
(NETWORK_ERROR)
Code: 210. DB::NetException: Connection refused (localhost:9000).
(NETWORK_ERROR)
Code: 210. DB::NetException: Connection refused (localhost:9000).
(NETWORK_ERROR)
Code: 210. DB::NetException: Connection refused (localhost:9000).
(NETWORK_ERROR)
Code: 210. DB::NetException: Connection refused (localhost:9000).
(NETWORK_ERROR)
Code: 210. DB::NetException: Connection refused (localhost:9000).
(NETWORK_ERROR)
Code: 210. DB::NetException: Connection refused (localhost:9000).
(NETWORK_ERROR)
Code: 210. DB::NetException: Connection refused (localhost:9000).
(NETWORK_ERROR)
```

Dropping table openfoodfacts if exists

Dropping table openfoodfacts\_proc if exists

Creating table

Running ClickHouse import with statistics...

Adding prediction column...

```
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> kubectl get pods -n spark-app
NAME                                READY   STATUS    RESTARTS   AGE
clickhouse-6bf47f45b6-zx57s        1/1     Running   0           2m34s
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8>
```

20. Запускаем под с моделью. Может занимать много времени, поскольку образ весит 1.7 Gb. Логи пода модели можно посмотреть в приложенном видео:

```
kubectl apply -f ./k8s/model-deployment.yaml
```

```
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> kubectl apply -f ./k8s/model-deployment.yaml
deployment.apps/model created
service/model-service unchanged
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> kubectl get pods -n spark-app
NAME                                READY   STATUS              RESTARTS   AGE
clickhouse-6bf47f45b6-zx57s        1/1     Running             0           3m25s
model-5cbc9b848-szpzf              0/1     ContainerCreating   0           2s
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8>
```

21. Чтобы проверить результат модели, можем провалиться в под clickhouse и запустить команду clickhouse client:

```
kubectl exec -it clickhouse-xxxx -n spark-app -- bash
```

```
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> kubectl exec -it clickhouse-6bf47f45b6-kfs2g -n spark-app -- bash
root@clickhouse-6bf47f45b6-kfs2g:/# clickhouse client
ClickHouse client version 25.4.2.31 (official build).
Connecting to localhost:9000 as user admin.
Connected to ClickHouse server version 25.4.2.

Warnings:
* Linux transparent hugepages are set to "always". Check /sys/kernel/mm/transparent_hugepage/enabled
* Delay accounting is not enabled, OSIOwaitMicroseconds will not be gathered. You can enable it using 'echo 1 > /proc/sys/kernel/task_delayacct' or by using sysctl.
```

```
clickhouse-6bf47f45b6-kfs2g :) SELECT prediction FROM openfoodfacts LIMIT 20
```

```
SELECT prediction
FROM openfoodfacts
LIMIT 20
```

```
Query id: 30ebef28-ca0a-4221-b51e-b8c271e103d1
```

	prediction
1.	NULL
2.	NULL
3.	NULL
4.	0
5.	NULL
6.	0
7.	NULL
8.	NULL
9.	NULL
10.	NULL
11.	NULL
12.	0
13.	NULL
14.	NULL
15.	NULL
16.	0
17.	NULL
18.	4
19.	NULL
20.	0

```
20 rows in set. Elapsed: 0.010 sec.
```

## 22. Выводим используемые ресурсы:

```
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> kubectl get pods -n spark-app
NAME                                READY   STATUS    RESTARTS   AGE
clickhouse-6bf47f45b6-kfs2g        1/1     Running   0           5m3s
model-5cbc9b848-hjsz4              1/1     Running   2 (33s ago) 4m11s
PS C:\Users\rusla\Desktop\ITMO-master\ml-big-data\lab-8> kubectl top pod -n spark-app
NAME                                CPU(cores)   MEMORY(bytes)
clickhouse-6bf47f45b6-kfs2g        146m         563Mi
model-5cbc9b848-hjsz4              1491m        481Mi
```

## 23. Удаляем поды:

```
kubectl delete deployment model -n spark-app
```

```
kubectl delete deployment clickhouse -n spark-app
```