

Функциональные характеристики платформы корпоративных коммуникаций DataHubs

DataHubs - платформа хранения и обработки данных, построенная на основе компонентов Apache Ozone, Apache Spark, Trino, Impala, Ranger, Apache Hadoop, Hive, HBase, Flink, Airflow, Oozie и управляемая через Ambari. Платформа предназначена для централизованного хранения, потоковой обработки и управления большими объемами данных с мощными инструментами для автоматизации процессов и усиленной безопасностью через Apache Knox.

Основная функциональность DataHubs Hadoop

1. Удобный интерфейс развёртывания
2. Хранение данных
 - 2.1. Надежное распределенное хранилище данных, поддерживающее хранение больших объёмов неструктурированных и структурированных данных.
 - 2.2. Колоночная распределённая база данных для хранения больших объёмов структурированных данных с низкой задержкой по отклику.
3. Обработка данных
 - 3.1. Эластичное масштабирование и управление ресурсами
 - 3.2. Анализ данных через SQL запросы
 - 3.3. Распределённая обработка данных в режиме реального времени и пакетной обработки
4. Оркестрация и планирование задач
5. Безопасность и управление доступом
6. Мониторинг и управление кластером
7. Варианты развёртывание платформы: облачные сервисы или внутри периметра организации с помощью скриптов и документации.

Прerequisites для выполнения функциональных сценариев DataHubs Hadoop:

1. Успешное подключение по VPN
2. Подключение к веб интерфейсам через браузер
3. Подключений по ssh к первому серверу кластера

Инструкции по подключению в файле **Инструкция по доступу к экземпляру Unison DataHubs**

1. Сценарий: Тестирование загрузки данных в HDFS

- **Цель:** Проверить, что данные загружаются в Hadoop Distributed File System (HDFS) корректно.
- **Шаги:**

```
# Проверка доступности HDFS
```

```
hdfs dfsadmin -report
```

```
# Загрузка файла в HDFS
```

```
hdfs dfs -put portugal_listinigs.csv /user/ropouser
```

```
# Проверка наличия файла в HDFS
```

```
hdfs dfs -ls /user/ropouser
```

```
# Сравнение контрольной суммы
```

```
hdfs dfs -cat /user/ropouser/portugal_listinigs.csv |md5sum
```

```
md5sum portugal_listinigs.csv
```

- **Ожидаемый результат:** Данные успешно загружаются, и их контрольная сумма совпадает.

2. Сценарий: Тестирование выполнения MapReduce задачи

- **Цель:** Проверить, что MapReduce задачи выполняются корректно.
- **Шаги:**

Подготовка входного файла

```
echo -e "apple\nbanana\napple\norange\nbanana\napple" > input.txt
```

```
hdfs dfs -put input.txt /user/ropouser
```

Запуск MapReduce задачи (например, WordCount)

```
yarn jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount \  
/user/ropouser/input.txt /user/ropouser/count
```

Проверка результатов выполнения

```
hdfs dfs -ls /user/ropouser/count
```

```
hdfs dfs -cat /user/ropouser/count/part-r-00000
```

- **Ожидаемый результат:** Задача выполняется без ошибок, и результаты соответствуют ожиданиям.

3. Сценарий: Тестирование репликации данных в HDFS

- **Цель:** Убедиться, что данные в HDFS реплицируются согласно установленному уровню репликации.
- **Шаги:**

```
# Установка уровня репликации (например, 3)  
hdfs dfs -setrep 3 /user/ropouser/portugal_listinigs.csv
```

```
# Проверка уровня репликации  
hdfs fsck /user/ropouser/portugal_listinigs.csv -files -blocks -racks
```

- **Ожидаемый результат:** Количество реплик файла соответствует установленному уровню репликации.

4. Сценарий: Тестирование устойчивости NameNode

- **Цель:** Проверить поведение Hadoop при отказе активного NameNode.
- **Шаги:**

Проверка состояния NameNode

```
hdfs haadmin -getServiceState nn1
```

```
hdfs haadmin -getServiceState nn2
```

Принудительное отключение активного NameNode

Зайти в веб интерфейс ambari

Выбрать сервис HDFS

Перейти в ACTIVE NAMENODE

Остановить компонент Active NameNode / HDFS

Проверка переключения на резервный NameNode

```
hdfs haadmin -getServiceState nn1
```

```
hdfs haadmin -getServiceState nn2
```

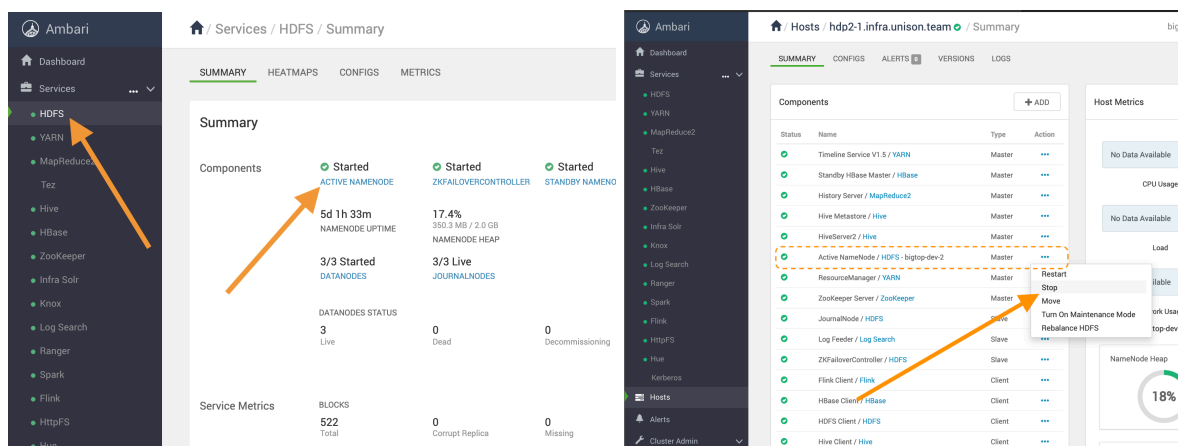
Проверка доступности данных после переключения

```
hdfs dfs -ls /user/ropouser/
```

Принудительное включение NameNode

Зайти в веб интерфейс ambari

Стартовать компонент NameNode / HDFS



- **Ожидаемый результат:** Система автоматически переключается на резервный NameNode, и доступ к файлам сохраняется.

Сценарий: Тестирование интеграции YARN с Hadoop

- **Цель:** Проверить, что YARN корректно распределяет ресурсы между задачами.
- **Шаги:**

```
# Проверка состояния YARN
yarn node -list
yarn application -list

# Запуск нескольких задач
# MapReduce
yarn jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar pi 10 1000

# Spark (если установлен Spark)
spark-submit --class org.apache.spark.examples.SparkPi \
--master yarn --deploy-mode cluster \
/path/to/spark-examples.jar 10

# Hive (если установлен Hive)
hive -e "CREATE TABLE test_table (id INT); INSERT INTO test_table VALUES (1);"

# Мониторинг распределения ресурсов через Resource Manager
yarn application -list -appStates RUNNING
```

- **Ожидаемый результат:** YARN корректно распределяет ресурсы, задачи завершаются успешно.

6. Тестирование создания и заполнения таблицы

- **Цель:** Проверить, что таблица создается и заполняется корректно.

Пример команд:

```
# Запуск подключения к Hive CLI
```

```
beeline
```

```
# Создание базы данных
```

```
CREATE DATABASE test_db;
```

```
# Переход в базу данных
```

```
USE test_db;
```

```
# Создание таблицы
```

```
CREATE TABLE test_table (
```

```
    id INT,
```

```
    name STRING,
```

```
    age INT
```

```
)
```

```
ROW FORMAT DELIMITED
```

```
FIELDS TERMINATED BY ','
```

```
STORED AS TEXTFILE;
```

```
# Загрузка данных в таблицу
```

```
LOAD DATA LOCAL INPATH '/local/path/data.csv' INTO TABLE test_table;
```

```
# Проверка данных в таблице
```

```
SELECT * FROM test_table;
```

7. Тестирование выполнения запроса с агрегатными функциями

- **Цель:** Проверить, что Hive корректно выполняет запросы с агрегатными функциями.

Пример команд:

Выполнение запроса для подсчета среднего возраста

```
SELECT AVG(age) AS average_age FROM test_table;
```

Выполнение запроса для группировки данных

```
SELECT name, COUNT(*) AS count
```

```
FROM test_table
```

```
GROUP BY name;
```

8. Тестирование экспорта данных в файл

- **Цель:** Проверить, что Hive корректно экспортирует данные из таблицы в файл.

Пример команд:

Указание пути для экспорта

```
INSERT OVERWRITE LOCAL DIRECTORY '/local/path/exported_data'
```

```
ROW FORMAT DELIMITED
```

```
FIELDS TERMINATED BY ','
```

```
SELECT * FROM test_table;
```

Проверка экспортированных данных

- `cat /local/path/exported_data/000000_0`

9. Тестирование выполнения простой задачи на подсчет слов

- **Цель:** Проверить, что Apache Spark корректно выполняет простую задачу, такую как подсчет частоты слов.

Пример команд:

```
# Подготовка входного файла
echo -e "apple banana apple\norange banana apple" >
/local/path/input.txt

# Запуск задания подсчета слов
spark-submit --class org.apache.spark.examples.JavaWordCount \
  --master local[2] \
  /path/to/spark-examples.jar \
  /local/path/input.txt

# Ожидаемый вывод подсчета слов
# (apple, 3)
# (banana, 2)
# (orange, 1)
```

2. Тестирование обработки данных из HDFS

- **Цель:** Проверить, что Spark может обрабатывать данные, загруженные из HDFS.

Пример команд:

```
# Загрузка входного файла в HDFS
hdfs dfs -mkdir /input
hdfs dfs -put /local/path/input.txt /input/

# Запуск задания Spark для подсчета строк в файле
spark-submit \
  --master yarn \
  --deploy-mode cluster \
  --class org.apache.spark.examples.JavaSparkPi \
  /path/to/spark-examples.jar \
  hdfs:///input/input.txt

# Просмотр результата выполнения в HDFS
```

```
hdfs dfs -ls /output
hdfs dfs -cat /output/part-00000
```

10. Тестирование интеграции Spark с Hive

- **Цель:** Проверить, что Spark может читать данные из таблицы Apache Hive.

Пример команд:

```
# Запуск Spark с включенной поддержкой Hive
spark-shell --master local --deploy-mode client --conf
spark.sql.catalogImplementation=hive

# Выполнение запроса к таблице Hive
spark.sql("USE test_db")
spark.sql("SELECT name, COUNT(*) FROM test_table GROUP BY
name").show()

# Опционально: сохранение результата в новую таблицу Hive
spark.sql("CREATE TABLE result_table AS SELECT name, COUNT(*) AS count
FROM test_table GROUP BY name")
```

11. Тестирование создания таблицы и вставки данных

- **Цель:** Убедиться, что таблица создается и данные могут быть вставлены и извлечены.

Пример команд:

```
# Запуск HBase shell
hbase shell

# Создание таблицы с одной колонной family
create 'test_table', 'cf1'

# Вставка данных в таблицу
put 'test_table', 'row1', 'cf1:col1', 'value1'
put 'test_table', 'row1', 'cf1:col2', 'value2'
put 'test_table', 'row2', 'cf1:col1', 'value3'

# Извлечение данных из таблицы
get 'test_table', 'row1'

# Проверка всех данных в таблице
scan 'test_table'
```

12. Тестирование обновления и удаления данных

- **Цель:** Проверить возможность обновления и удаления данных в таблице.

Пример команд:

```
# Обновление данных в таблице
put 'test_table', 'row1', 'cf1:col1', 'updated_value1'

# Проверка обновленных данных
get 'test_table', 'row1'

# Удаление конкретного значения из таблицы
delete 'test_table', 'row1', 'cf1:col2'
```

```
# Проверка, что значение удалено
get 'test_table', 'row1'
```

```
# Удаление всей строки
deleteall 'test_table', 'row2'
```

```
# Проверка, что строка удалена
scan 'test_table'
```

13. Тестирование экспорта и импорта данных

- **Цель:** Убедиться, что данные можно экспортировать и импортировать.

Пример команд:

```
# Экспорт данных в HDFS
hbase org.apache.hadoop.hbase.mapreduce.Export test_table
/hbase/output/test_table_export

# Проверка экспортированных данных в HDFS
hdfs dfs -ls /hbase/output/test_table_export
hdfs dfs -cat /hbase/output/test_table_export/part-m-00000

# Удаление таблицы перед импортом
disable 'test_table'
drop 'test_table'
# Создание новой таблицы
create 'test_table', 'cf1'
# Импорт данных обратно в таблицу
hbase org.apache.hadoop.hbase.mapreduce.Import test_table
/hbase/output/test_table_export

# Проверка данных после импорта
scan 'test_table'
```

14. Тестирование запуска простого workflow

- **Цель:** Проверить, что Apache Oozie успешно запускает простой workflow.

Пример команд:

```
# Проверка состояния Oozie сервера
oozie admin -oozie http://<oozie-server>:11000/oozie -status

# Загрузка workflow в HDFS
hdfs dfs -mkdir -p /user/oozie/workflows/test_workflow
hdfs dfs -put /local/path/workflow.xml
/user/oozie/workflows/test_workflow/

# Запуск workflow
oozie job -oozie http://<oozie-server>:11000/oozie \
  -config /local/path/job.properties \
  -run
# Проверка статуса выполнения
oozie job -oozie http://<oozie-server>:11000/oozie -info <job-id>
```

Пример содержимого workflow.xml:

```
<workflow-app name="test-workflow" xmlns="uri:oozie:workflow:0.5">
  <start to="step1"/>
  <action name="step1">
    <shell>
      <job-tracker>${jobTracker}</job-tracker>
      <name-node>${nameNode}</name-node>
      <exec>/bin/echo</exec>
      <arg>Hello Oozie</arg>
    </shell>
    <ok to="end"/>
    <error to="fail"/>
  </action>
  <kill name="fail">
    <message>Workflow failed</message>
  </kill>
  <end name="end"/>
</workflow-app>
```

15. Тестирование координатора для периодического запуска workflow

- **Цель:** Проверить, что Oozie может запускать workflow на основе расписания.

Пример команд:

```
# Загрузка координатора и workflow в HDFS
hdfs dfs -mkdir -p /user/oozie/coordinator/test_coordinator
hdfs dfs -put /local/path/coordinator.xml
/user/oozie/coordinator/test_coordinator/
hdfs dfs -put /local/path/workflow.xml
/user/oozie/coordinator/test_coordinator/
# Запуск координатора
oozie job -oozie http://<oozie-server>:11000/oozie \
  -config /local/path/coordinator.properties \
  -run
# Проверка статуса координатора
oozie job -oozie http://<oozie-server>:11000/oozie -info
<coordinator-id>
```

Пример содержимого **coordinator.xml**:

```
<coordinator-app name="test-coordinator" frequency="5"
xmlns="uri:oozie:coordinator:0.4">
  <controls>
    <timeout>60</timeout>
    <concurrency>1</concurrency>
  </controls>
  <datasets>
    <dataset name="data" frequency="5"
initial-instance="2024-01-01T00:00Z">
      <uri-template>/user/oozie/data/input</uri-template>
    </dataset>
  </datasets>
  <action>
    <workflow>
<app-path>${nameNode}/user/oozie/coordinator/test_coordinator/workflow
.xml</app-path>
    </workflow>
  </action>
</coordinator-app>
```

16. Тестирование SLA (Service Level Agreement) для workflow

- **Цель:** Проверить, что Oozie корректно отслеживает выполнение workflow по SLA.

Пример команд:

```
# Загрузка workflow с SLA в HDFS
hdfs dfs -mkdir -p /user/oozie/workflows/sla_workflow
hdfs dfs -put /local/path/sla_workflow.xml
/user/oozie/workflows/sla_workflow/

# Запуск workflow с SLA
oozie job -oozie http://<oozie-server>:11000/oozie \
  -config /local/path/sla_job.properties \
  -run

# Проверка выполнения SLA
oozie sla -oozie http://<oozie-server>:11000/oozie -info <job-id>
```

Пример содержимого **sla_workflow.xml**:

```
<workflow-app name="sla-workflow" xmlns="uri:oozie:workflow:0.5">
  <start to="step1"/>
  <action name="step1">
    <shell>
      <job-tracker>${jobTracker}</job-tracker>
      <name-node>${nameNode}</name-node>
      <exec>/bin/echo</exec>
      <arg>Hello SLA</arg>
    </shell>
    <ok to="end"/>
    <error to="fail"/>
  </action>
  <kill name="fail">
    <message>Workflow failed</message>
  </kill>
  <end name="end"/>
  <sla:info xmlns:sla="uri:oozie:sla:0.1">
```

```
<sla:nominal-time>${nominalTime}</sla:nominal-time>
<sla:should-end>${shouldEnd}</sla:should-end>
<sla:max-duration>${maxDuration}</sla:max-duration>
</sla:info>
</workflow-app>
```

Основная функциональность DataHubs Storage

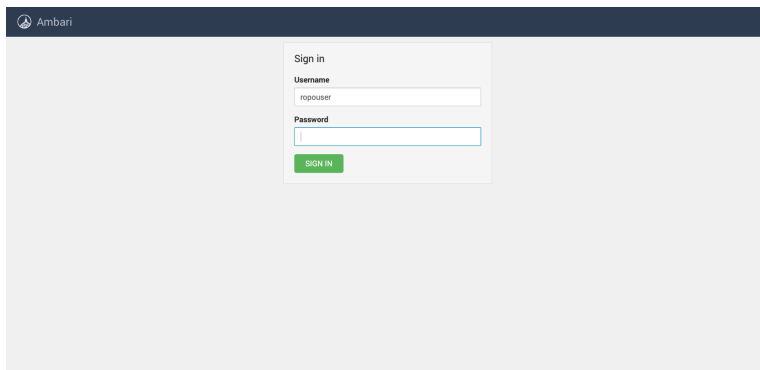
1. Просмотр состояния кластера DataHubs Storage через UI Recon

Цель: Убедиться, что файл успешно загружается в Ozone и может быть прочитан.

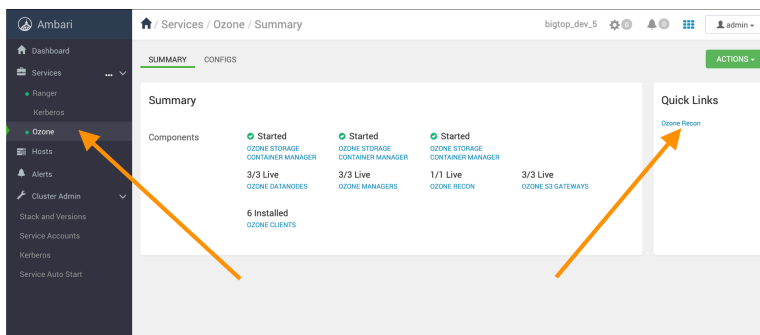
Пример команд:

Перейти в веб интерфейс Ambari

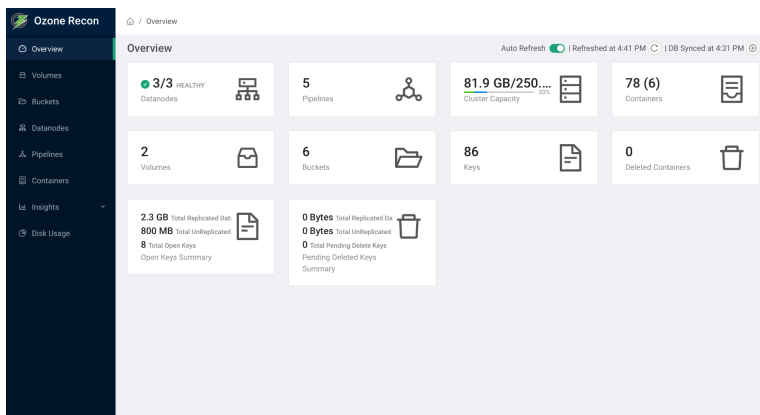
<http://hdp8-0.infra.unison.team:8080>



Выбрать в роутер-меню компонент Ozone



Перейти по ссылке Recon UI



2. Тестирование загрузки и чтения файла в Ozone

- **Цель:** Убедиться, что файл успешно загружается в Ozone и может быть прочитан.

Пример команд:

Создание ключевого пространства

```
ozone sh volume create /vol1
```

```
ozone sh bucket create /vol1/bucket1
```

Загрузка файла

```
ozone sh key put /vol1/bucket1/testfile.txt /local/path/testfile.txt
```

Чтение содержимого файла

```
ozone sh key get /vol1/bucket1/testfile.txt
```

```
/local/path/downloaded_testfile.txt
```

Проверка файла на локальной машине

```
diff /local/path/testfile.txt /local/path/downloaded_testfile.txt
```

3. Тестирование управления ACL (Access Control List)

- **Цель:** Проверить, что управление доступом к ресурсам работает корректно.

Пример команд:

```
# Установка ACL для пользователя
ozone sh bucket addacl /vol1/bucket1 -u user1:rw

# Проверка установленных ACL
ozone sh bucket getacl /vol1/bucket1

# Проверка доступа от имени другого пользователя (должен быть доступ)
sudo -u user1 ozone sh key put /vol1/bucket1/testfile2.txt
/local/path/testfile2.txt
```

4. Тестирование репликации данных

- **Цель:** Убедиться, что данные реплицируются согласно настройкам.

Пример команд:

```
# Создание bucket с заданным фактором репликации
ozone sh bucket create /vol1/bucket2 --replication-type=RATIS
--replication=3

# Загрузка файла в bucket
ozone sh key put /vol1/bucket2/testfile.txt /local/path/testfile.txt

# Проверка статуса файла и репликации
ozone sh key info /vol1/bucket2/testfile.txt
```

5. Тестирование отказоустойчивости Ozone Manager

- **Цель:** Проверить поведение кластера при отказе одного из Ozone Manager.

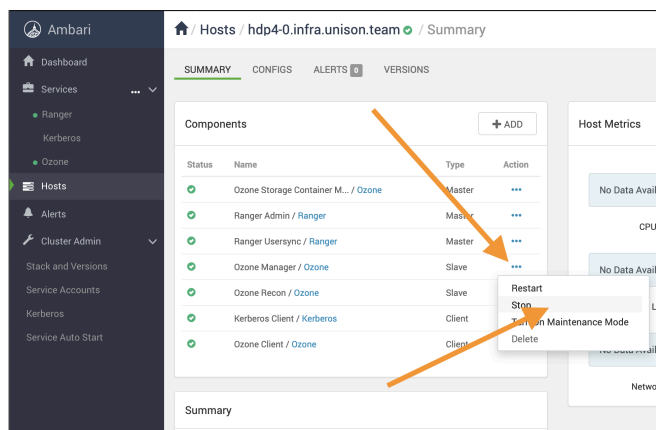
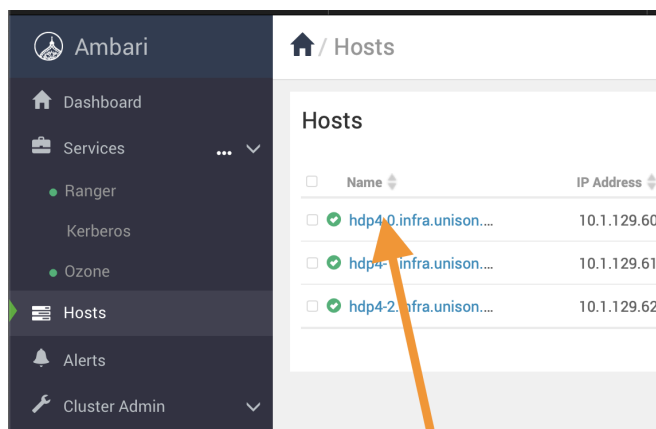
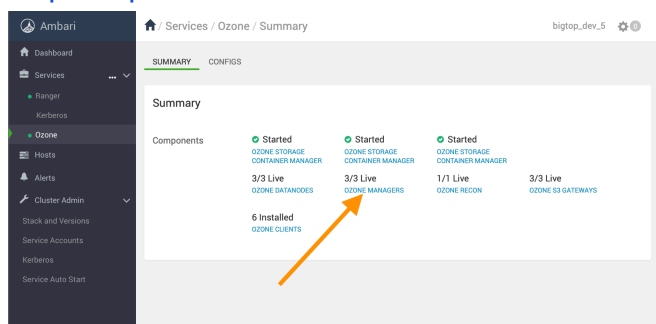
Пример команд:

Проверка состояния Ozone Manager
ozone admin om roles

Принудительное отключение активного Ozone Manager

Перейти в web UI Ambari

<http://hdp8-0.infra.unison.team:8080>



```
# Проверка переключения на резервный Ozone Manager
```

```
ozone admin om roles
```

```
# Проверка доступности данных после переключения
```

```
ozone sh key get /vol1/bucket1/testfile.txt /local/path/testfile.txt
```

6. Тестирование интеграции S3 Gateway с Ozone

- **Цель:** Проверить, что S3 Gateway работает корректно с Ozone.

Пример команд:

```
# Создание bucket через S3 интерфейс
```

```
aws --endpoint-url=http://<ozone-s3-gateway>:9878 s3 mb s3://bucket-s3
```

```
# Загрузка файла через S3 интерфейс
```

```
aws --endpoint-url=http://<ozone-s3-gateway>:9878 s3 cp  
/local/path/testfile.txt s3://bucket-s3/
```

```
# Проверка содержимого bucket
```

```
aws --endpoint-url=http://<ozone-s3-gateway>:9878 s3 ls  
s3://bucket-s3/
```

```
# Скачивание файла через S3 интерфейс
```

- `aws --endpoint-url=http://<ozone-s3-gateway>:9878 s3 cp
s3://bucket-s3/testfile.txt /local/path/downloaded_testfile.txt`