弹性 MapReduce

EMR 容器版

产品文档

腾讯云

文档目录

EMR 容器版
EMR 容器版概述

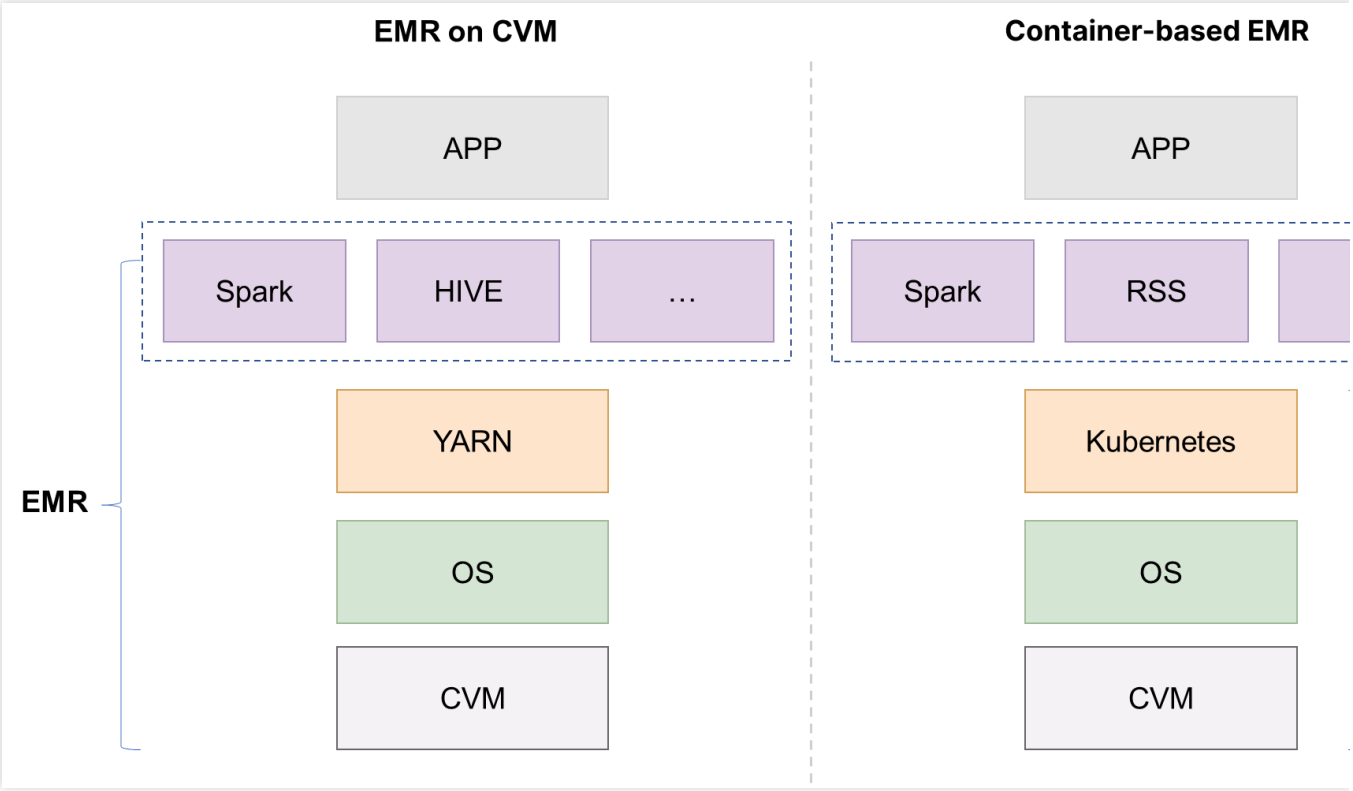
最近更新时间：2023-12-27 15:04:34

停止新购说明：

因 EMR 容器版产品能力升级，EMR 容器版于2023年3月10日停止新购，已购集群使用无影响。新版本即将上线内测，敬请期待。

形态对比

弹性 MapReduce（EMR）提供了开源大数据组件基于云服务器（CVM）和 Serverless 容器服务两种部署方式，灵活满足不同用户需求。



形态	描述
EMR on CVM	EMR 负责根据用户需要将开源大数据组件安装部署在 CVM 上，并启动所安装的服务。 同时，EMR 控制台为用户提供了关于集群和组件服务的各项运维操作，方便用户执行大数据任务。
EMR 容器版	EMR 负责将大数据组件部署在由 EKS 提供的资源内，组件服务会运行在容器内。

	您可以直接将 Spark 任务运行在容器集群上，同时关联 RSS 集群，提高 Spark 任务的稳定性。

EMR 容器版优势

优势	描述
降低成本	EMR 容器版集群提供 Serverless 服务，您可以根据业务需要开箱即用，充分利用资源。 Spark 集群还会根据作业需求自动创建 POD 资源，并在作业结束后释放，帮助节省成本。
运维简单	EMR 容器版集群基于全托管 Kubernetes 服务的 EKS 部署，相比 CVM 可以快速恢复异常组件服务。 Spark 容器集群自动调整 POD 资源，简化了用户对于节点资源的运维操作。
灵活伸缩	EMR 容器版集群支持用户调整容器数量，依赖于 EKS 的无限资源和自研轻量虚拟化技术。 可以实现 POD 资源的快速伸缩，保证大数据量任务的资源需要。

操作指南
创建集群

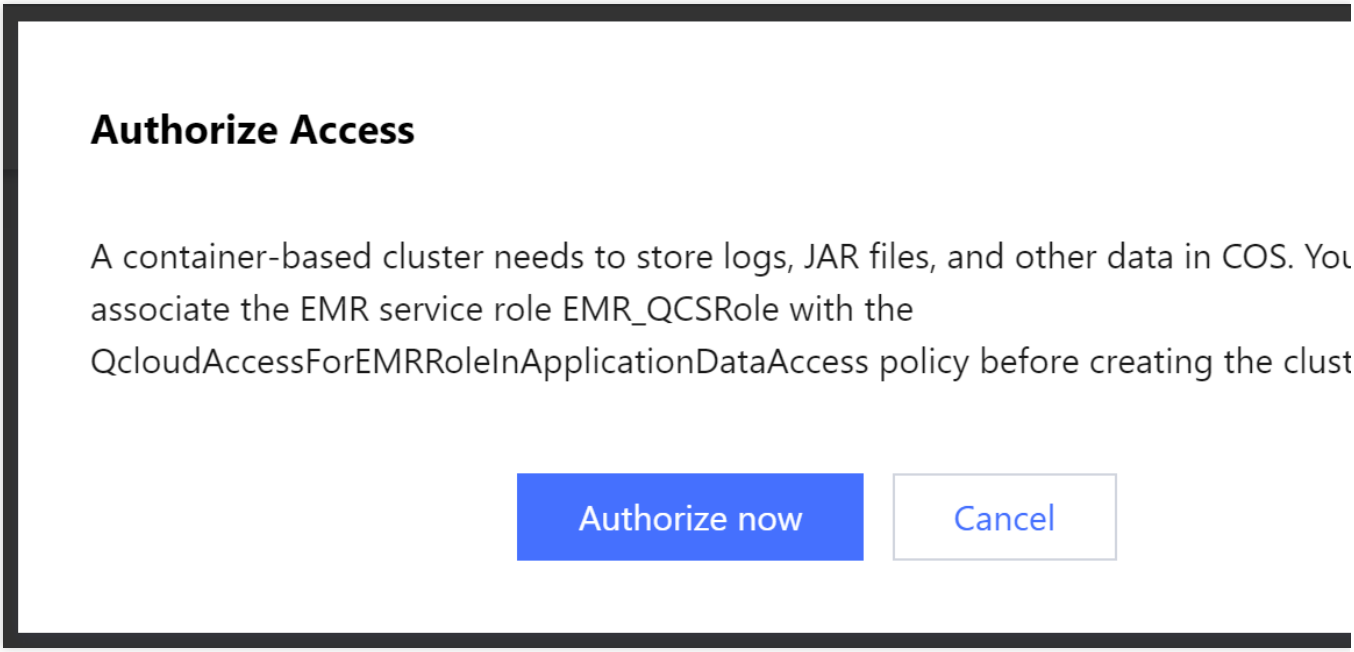
最近更新时间：2023-12-27 15:04:56

操作场景

本文为您介绍通过 EMR 控制台创建一个 EMR 容器版集群的操作。

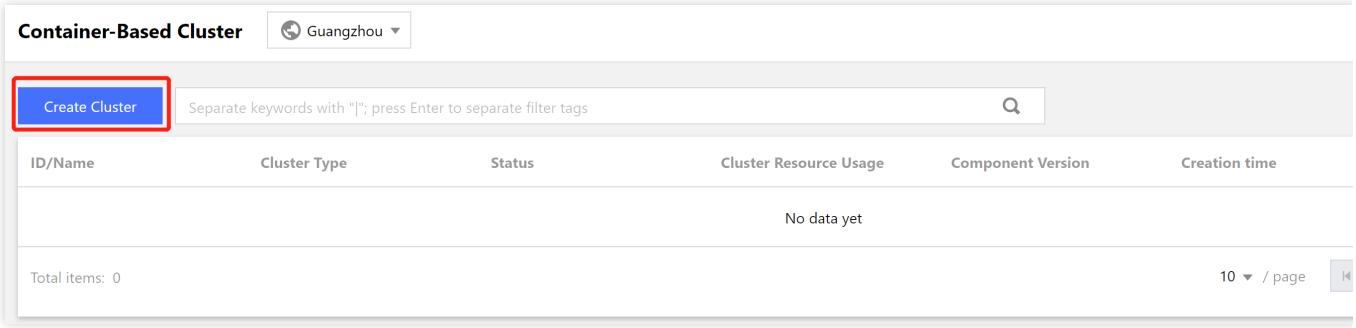
前提条件

1. 完成角色授权。具体操作请参见 角色授权 。
2. 完成对象存储授权。当用户创建集群时，若未开启对象存储访问授权，会有如下提示。单击**前往授权**，进行对象存储授权，一次授权后，后续创建集群将默认授权。



操作步骤

1. 登录 EMR 容器版控制台，在容器版列表页单击**创建集群**。



2. 在创建集群页面，选择相关配置。

字段名	说明
集群名称	1. 长度限制为6-36个字符，只允许包含中文、字母、数字、-、_ 2. 默认添加随机集群名称
地域	地域（Region）是指物理的数据中心的地理区域，支持地域有：北京、上海、广州、南京
集群类型	当前支持 Spark 和 RSS 集群类型
组件版本	所选集群类型下的组件及组件版本信息
容器类型	选择 EKS 集群时，如果所在地域没有 EKS 集群，将默认创建一个隐藏 EKS 集群用于扩展 EMR 计算资源，占用一个EKS使用配额
容器网络	设置 EMR 专用 EKS 隐藏集群网络。若该 EKS 集群已选择过容器网络，则容器网络被绑定不再可选
规格配置	当前仅支持 RSS 集群配置资源规格。请根据业务需要配置 Coordinator 和 Shuffle Server 角色的 Pod 规格。注意，一旦指定数据盘类型、单盘大小和数量、CPU 类型和范围、以及内存范围后，后续不支持更改。
COS 存储桶	1. 选择已有的存储桶，或者在对象存储 COS 控制台新建存储桶 2. 客户使用 COS 前，需要提前授权 COS 读写权限给 EMR 集群

3. 单击**创建**，EMR 集群进入创建过程，可在 EMR 容器版控制台中看到新建的集群。

管理集群
集群管理概述

最近更新时间：2023-12-27 15:05:15

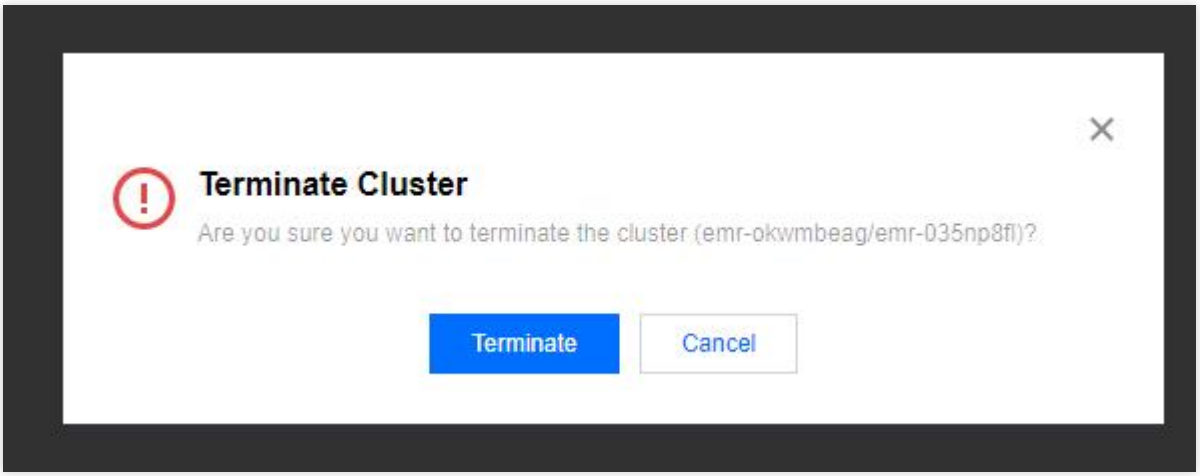
本文为您介绍通过控制台查看 EMR 容器版集群信息。

查看集群信息

1. 集群创建成功后，登录 EMR 容器版控制台，在集群列表页面单击需要管理的集群 **ID/名称**。或登录 EMR 容器版控制台，在集群列表操作中选择**详情**。
2. 集群信息是记录 EMR 集群的基本信息，用户可以在集群信息页查看集群的地域信息、名称空间、组件版本、容器类型/集群类型、容器网络、对象存储、存储桶名称、自定义服务角色、资源使用量等。
3. 如需精细化授权，可设置自定义服务角色，用于大数据作业运行时访问云上资源。自定义服务角色类型为“腾讯云产品服务”，支持角色的服务选择“弹性 MapReduce”。

销毁集群

当您不再需要 EMR 容器版集群时，您可以登录 EMR 容器版控制台，在集群列表操作中选择**销毁**，打开集群销毁弹框。在集群销毁页面，确认需要销毁的集群信息，确认无误后，单击**确定销毁**，即可销毁集群。



删除集群

当 EMR 容器版集群创建失败时，您可以登录 EMR 容器版控制台，在集群列表操作中选择**删除**，打开集群删除弹框。在集群删除页面，确认需要删除的集群信息，确认无误后，单击**确定删除**，即可删除集群。

管理 Spark 作业

最近更新时间：2023-12-27 15:05:35

功能介绍

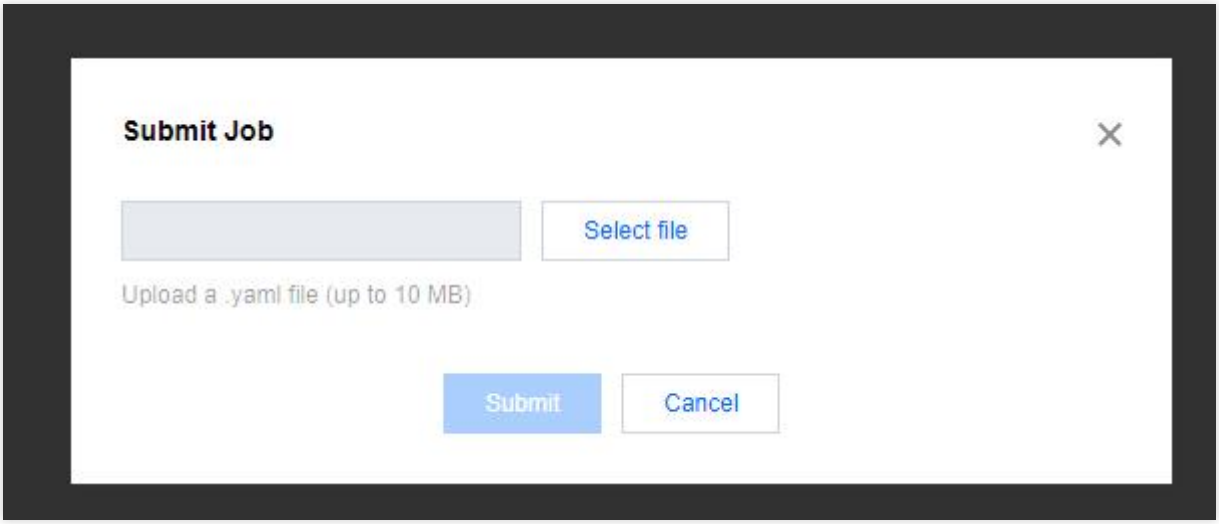
EMR 容器版集群支持在控制台提交 Spark 作业，查看作业信息。

注意

提交作业的文件格式为 yaml，大小10M 以内。

操作步骤

1. 登录 EMR 容器版控制台，在集群列表中单击对应的集群 **ID/名称**进入集群详情页。
2. 在集群详情页中单击**作业管理**，即可进行相关作业提交和查询。
3. EMR 支持 CRD 方式提交作业，用户编写 yaml 作业文件后，由控制台进行作业提交。
4. 单击作业列表上方的**提交作业**，打开提交作业弹框。选择需要提交的作业文件，单击**确认**后，即可提交作业。



5. 单击作业列表中的**详情**，可跳转 Spark Historyserver UI 链接查看作业详情。
6. 单击作业列表中的**删除**，打开删除弹框。在删除作业页面，确认需要删除的作业信息，确认无误后，单击**确定**，即可删除作业。

作业示例

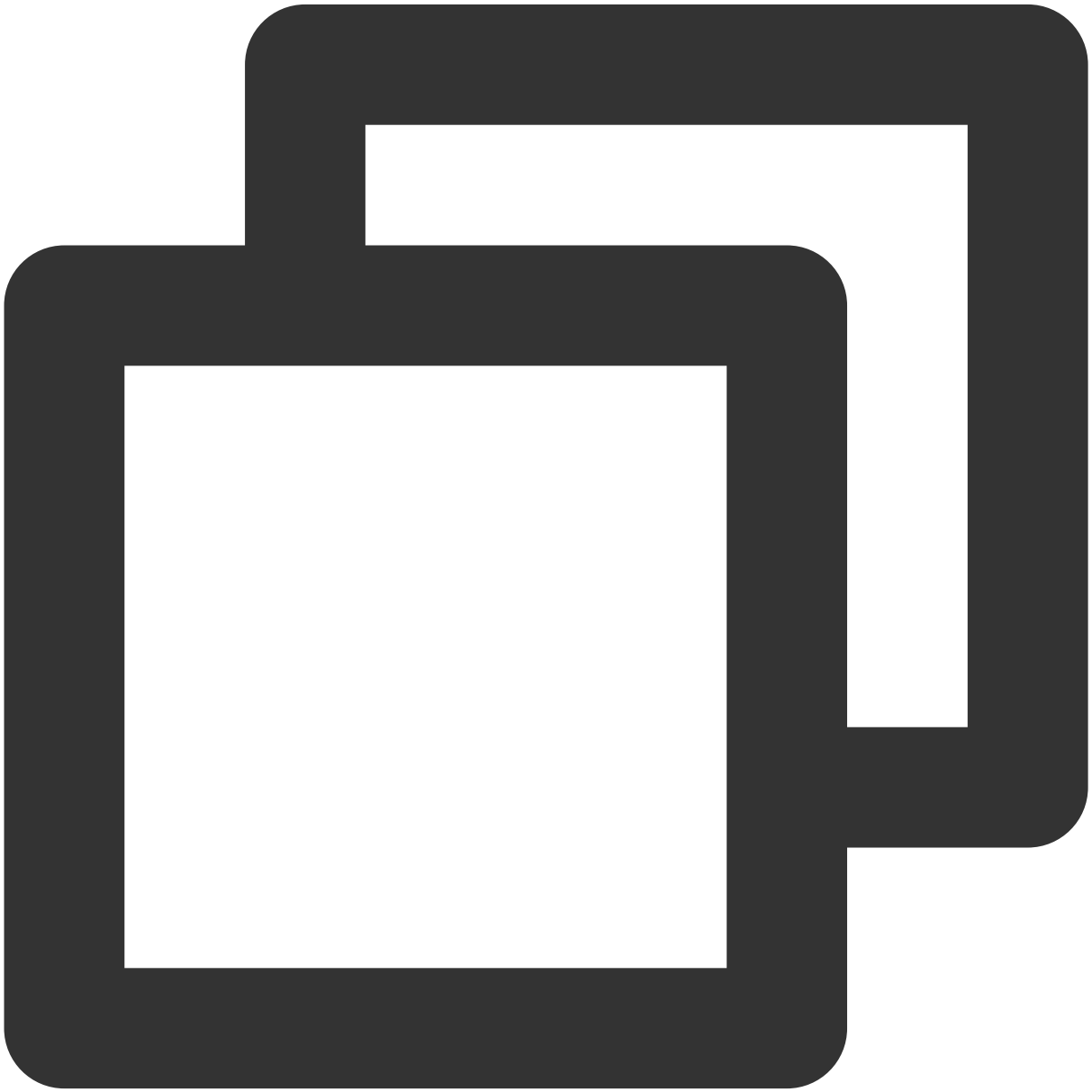
通过 CRD 方式提交 Spark 作业的一般流程如下：

1. 编写 Spark 程序。
2. 编译打成 jar 包，将 jar 包放到 cos 或者 hdfs 等文件系统，或编写 Dockerfile 将 Spark 程序 jar 包打到镜像。
3. 编写 yaml 文件，在控制台进行提交。

下面将给出4种 Spark 作业示例：

示例1. 使用 Spark 官方 jar 包

新建 yaml 作业文件示例内容如下：



```
apiVersion: "sparkoperator.k8s.io/v1beta2"
```

```
kind: SparkApplication
metadata:
  name: test1
spec:
  hadoopConf:
    "fs.cosn.userinfo.secretId":"$SecretId"
    "fs.cosn.userinfo.secretKey":"$SecretKey"
  type: Scala
  mode: cluster
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///opt/spark/examples/jars/spark-examples_2.12-3.2.0.
```

本文示例中的参数描述，请参见 sparkoperator 。其中：

apiVersion 和 kind 为 k8s 中资源种类和版本，此处不能更改。

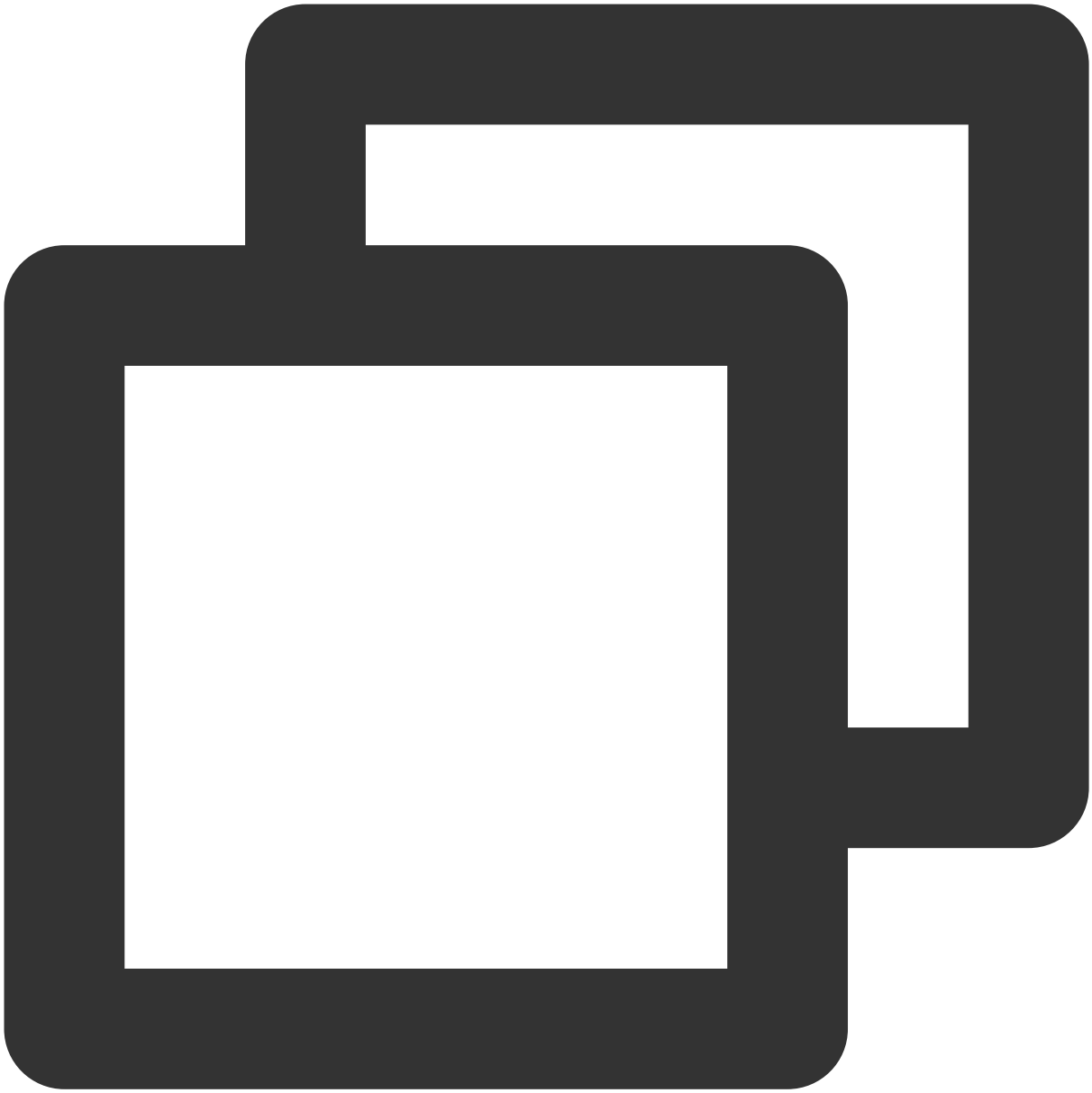
Metadata.name 定义作业名称，本文以 test1为例，用户可以自定义。

Spec.hadoopConf 定义与 hadoop 相关配置信息，与 cos 交互需要配置密钥信息，可通过 API 密钥管理 获取密钥信息，代码中的 `$SecretId` 、 `$Secretkey` 需要替换成业务对应的 SecretId 和 Secretkey。

type 定义 spark 的程序种类，包括 Java、Scala、Python、R。本文以 Scala 为例，您可根据需要选填。

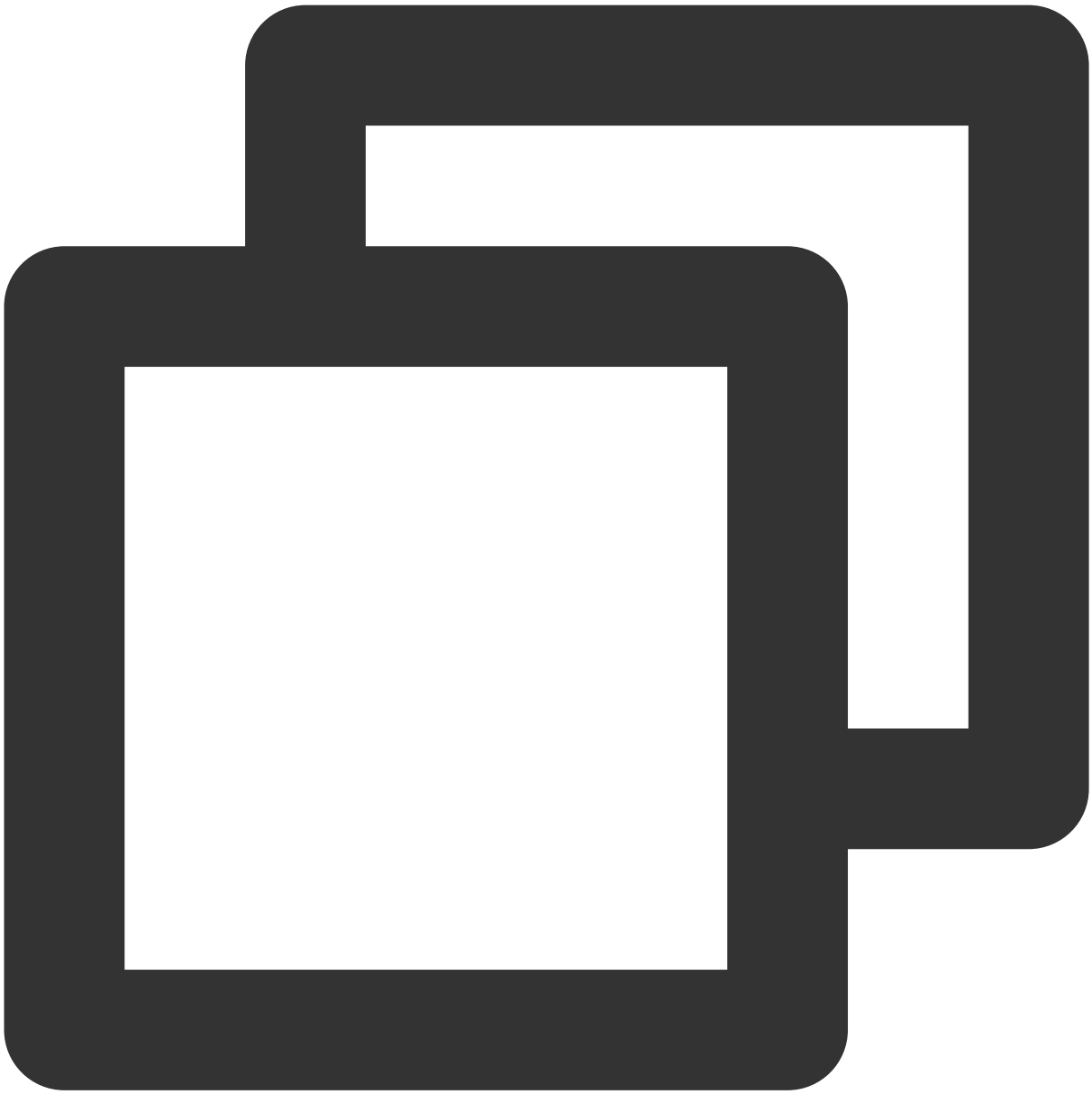
mode 定义 sparkApplication 的部署模式，包括 cluster 和 client。本文以 cluster 为例，您可根据需要选填。

driver 和 executor 分别定义 spark 的驱动器和执行器，由后台自动生成，其默认参数如下：



```
driver:
  cores: 1
  memory: 512m
executor:
  cores: 1
  instances: 2
  memory: 512m
```

用户可以自定义 driver 和 executor 参数，补充在 yaml 作业文件示例1中，此时自定义参数会覆盖默认参数，示例如下：



```
driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
```

示例2. 自行编译打包并将 jar 包放到 cos（推荐）

以下示例演示用户编写 spark 程序，打 jar 包，编写 yaml 作业进行提交的完整流程。

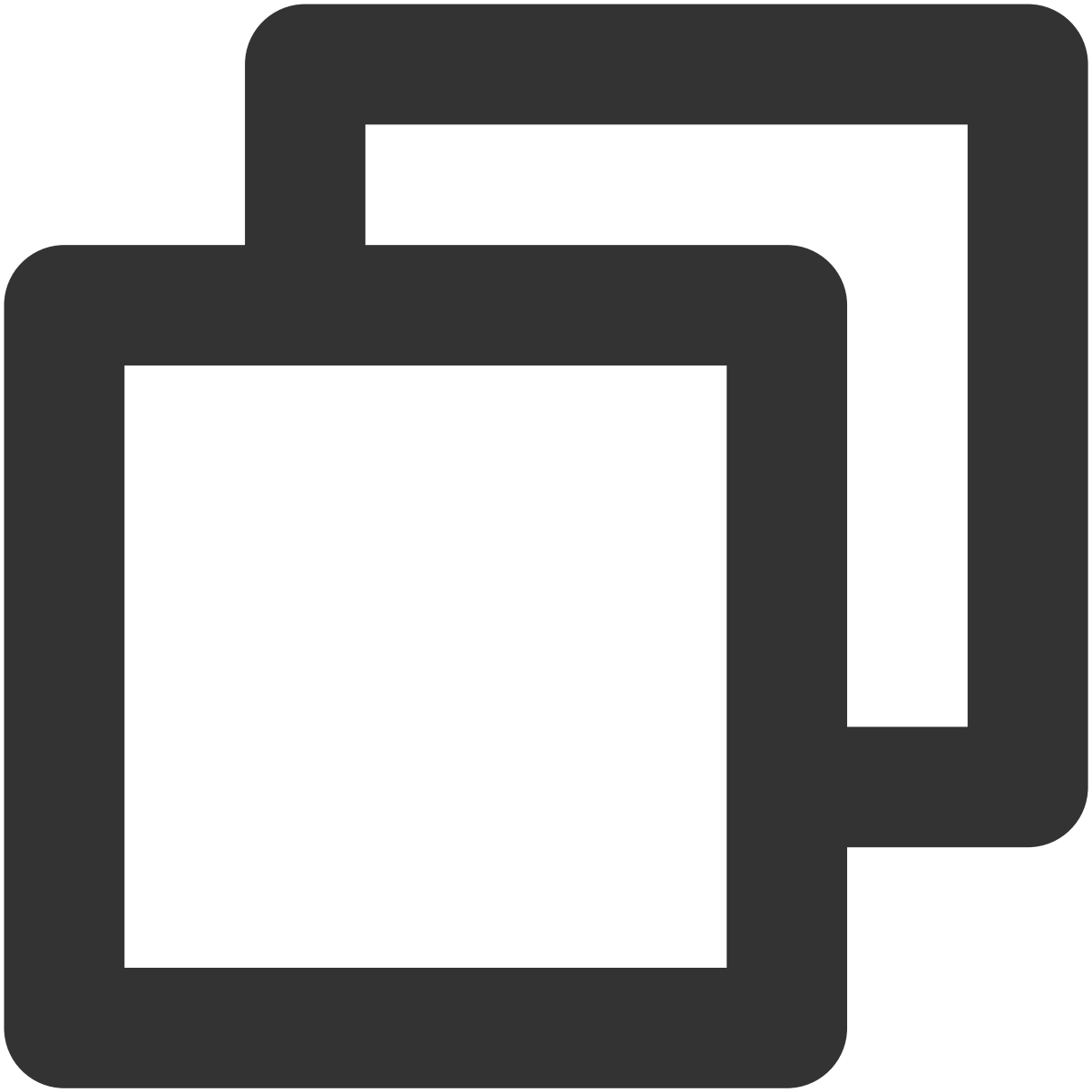
1. 开发准备

因为任务中需要访问腾讯云对象存储（COS），所以需要在 COS 中存在一个存储桶（Bucket），该存储桶可以是创建集群时的存储桶，也可以创建新的存储桶（需与创建集群时选择的存储桶位于同一地域）。

2. 使用 Maven 创建工程

在本次演示中，自己建立工程编译打包之后上传。推荐您使用 Maven 来管理您的工程。Maven 是一个项目管理工具，能够帮助您方便的管理项目的依赖信息。

3. 编写 wordcount 程序，添加样例代码：

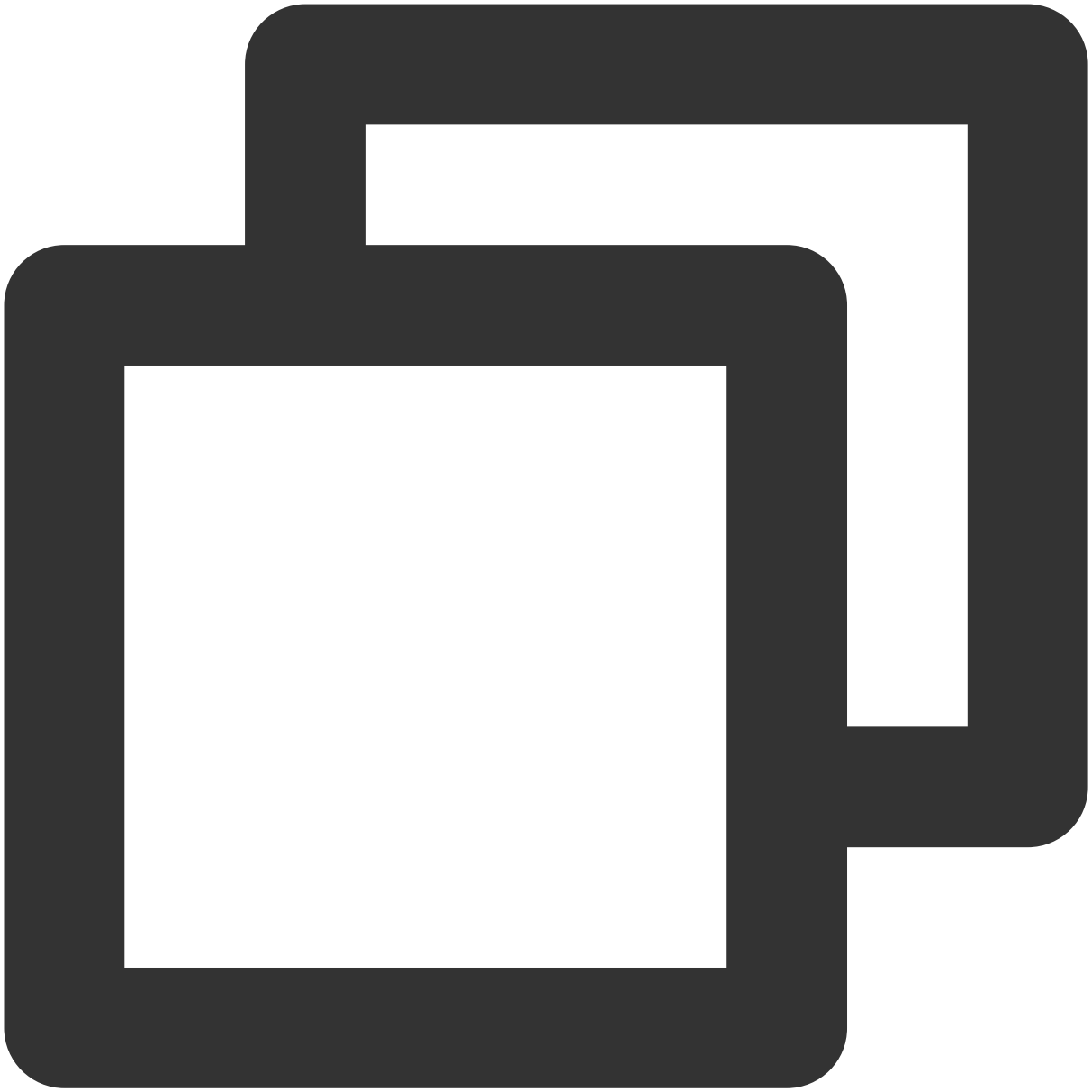


```
import java.util.Arrays;
import java.util.regex.Pattern;

import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.sql.SparkSession;
import scala.Tuple2;

public class WordCountOnCos {
```

```
    private static final Pattern SPACE = Pattern.compile(" ");
    public static void main(String[] args){
        if (args.length < 1) {
            System.err.println("Usage: JavaWordCount <file>");
            System.exit(1);
        }
        SparkSession spark = SparkSession.builder().appName("wordCountOnCos").getOr
        JavaRDD<String> lines = spark.read().textFile(args[0]).javaRDD();
        JavaRDD<String> words = lines.flatMap(s -> Arrays.<String>asList(SPACE.spli
        JavaPairRDD<String, Integer> ones = words.mapToPair(s -> new Tuple2(s, Inte
        JavaPairRDD<String, Integer> counts = ones.reduceByKey((i1, i2) -> Integer.
        counts.saveAsTextFile(args[1]);
        spark.stop();
    }
}
```

4. 执行 mvn package 命令对整个工程进行打包。

5. 将 jar 包上传到 cos 桶，编写 yaml 文件如下：



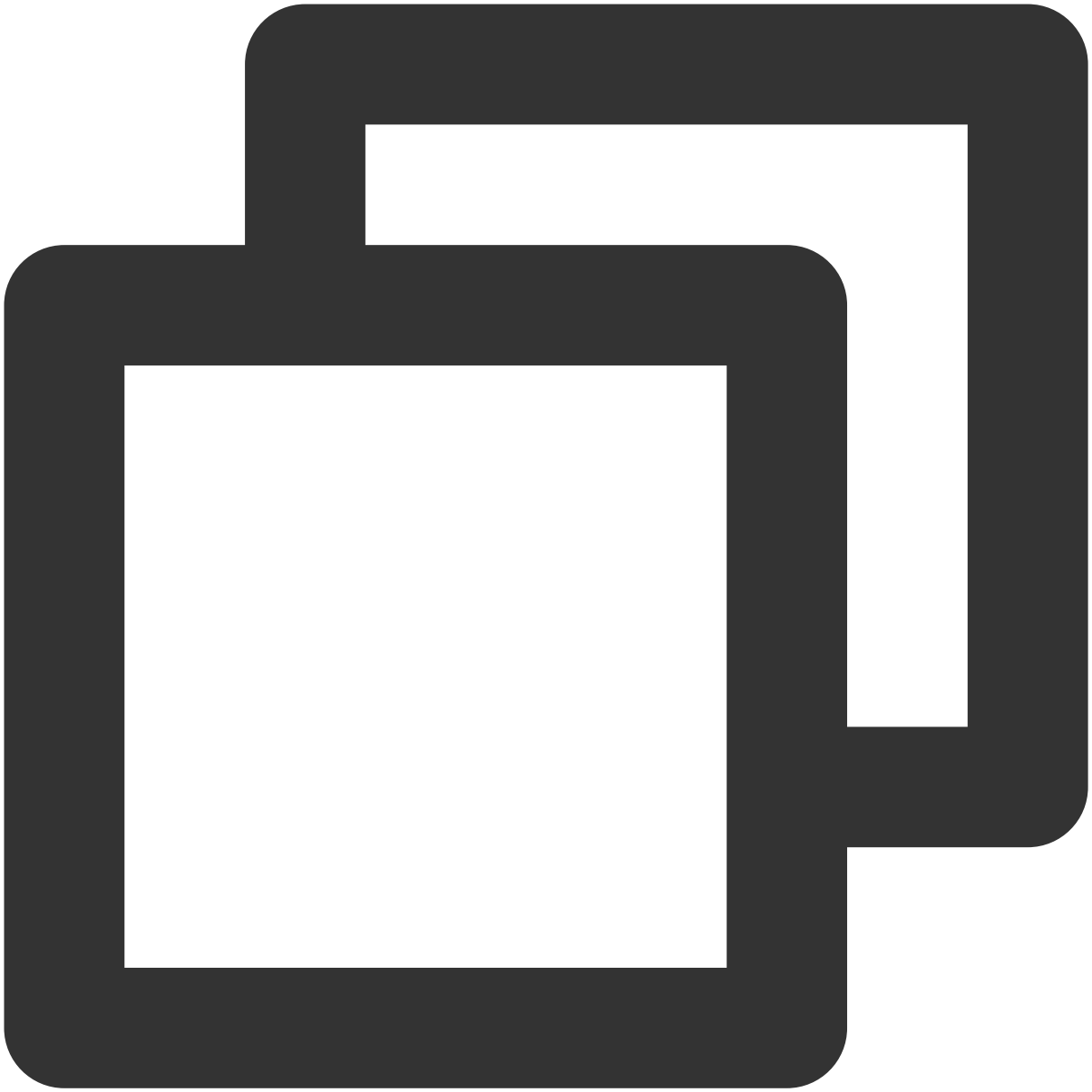
```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: test2
spec:
  hadoopConf:
    "fs.cosn.userinfo.secretId":"$SecretId"
    "fs.cosn.userinfo.secretKey":"$SecretKey"
  type: Java
  mode: cluster
  mainClass: com.tencent.WordCountOnCos
```

```
  mainApplicationFile: "cosn://kt-test-251007880/sparkapp/jar/wordcount.jar"
  arguments:
    - "cosn://kt-test-251007880/sparkapp/input/input"
    - "cosn://kt-test-251007880/sparkapp/output"
```

其中 arguments 是传递给主类的参数，本例中表示 wordcount 程序的输入和输出目录。本文中 mainApplicationFile 和 wordcount 程序的输入和输出目录为示例，用户可以自定义。

示例3. 自行编译打包并将 jar 包放到 HDFS

用户编写 spark 程序，打 jar 包同示例2，然后将 jar 包上传到 HDFS，编写 yaml 文件如下：



```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: test3
spec:
  hadoopConf:
    "fs.cosn.userinfo.secretId":"$SecretId"
    "fs.cosn.userinfo.secretKey":"$SecretKey"
  type: Java
  mode: cluster
  mainClass: com.tencent.WordCountOnCos
```

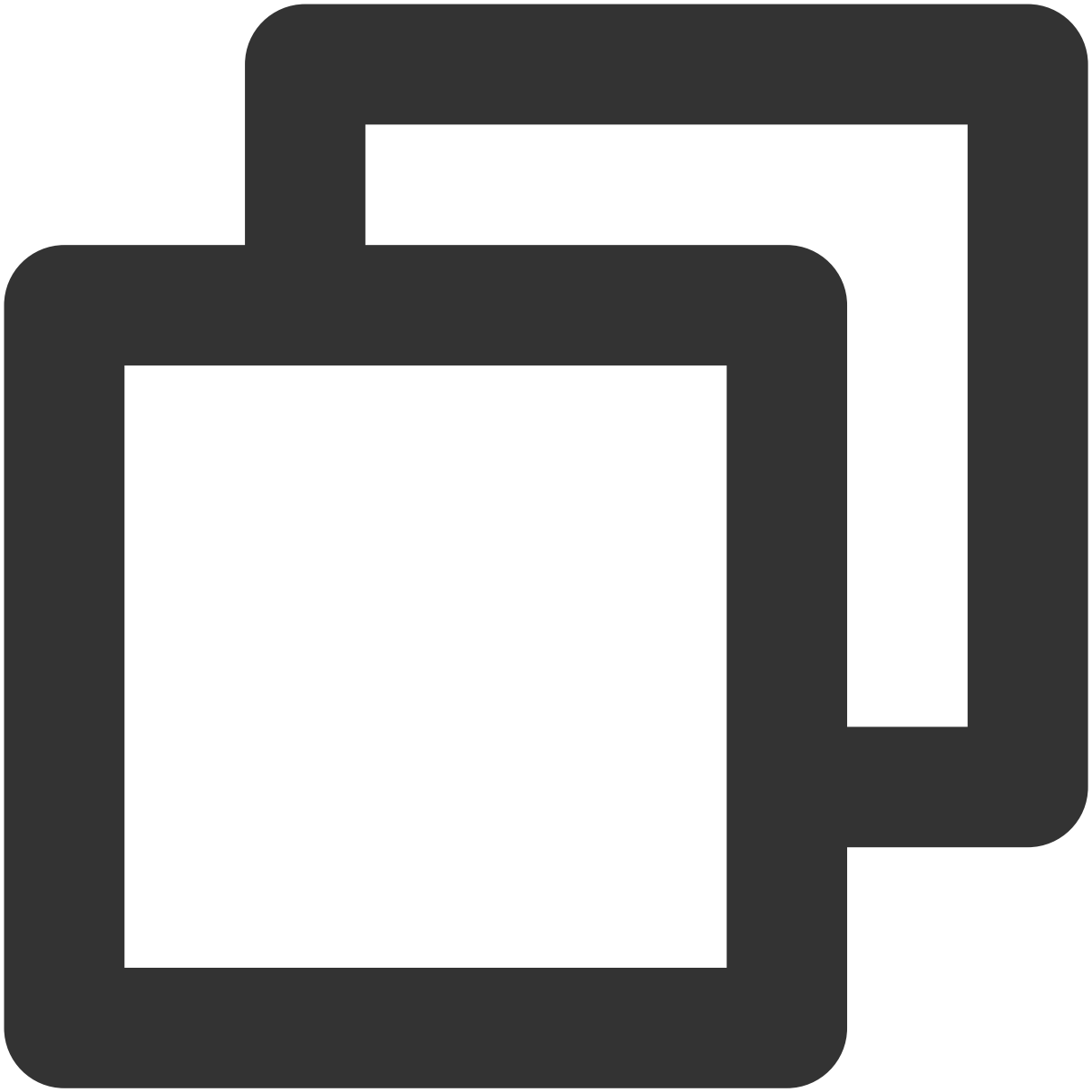
```
  mainApplicationFile: "hdfs://$ip:$port/sparkapp/jar/wordcount.jar"
  arguments:
    - "cosn://kt-test-251007880/sparkapp/input/input"
    - "hdfs://$ip:$port/sparkapp/output"
```

注意

若使用 HFDS 存放 jar 包，HDFS 需要和容器集群位于同一 VPC。

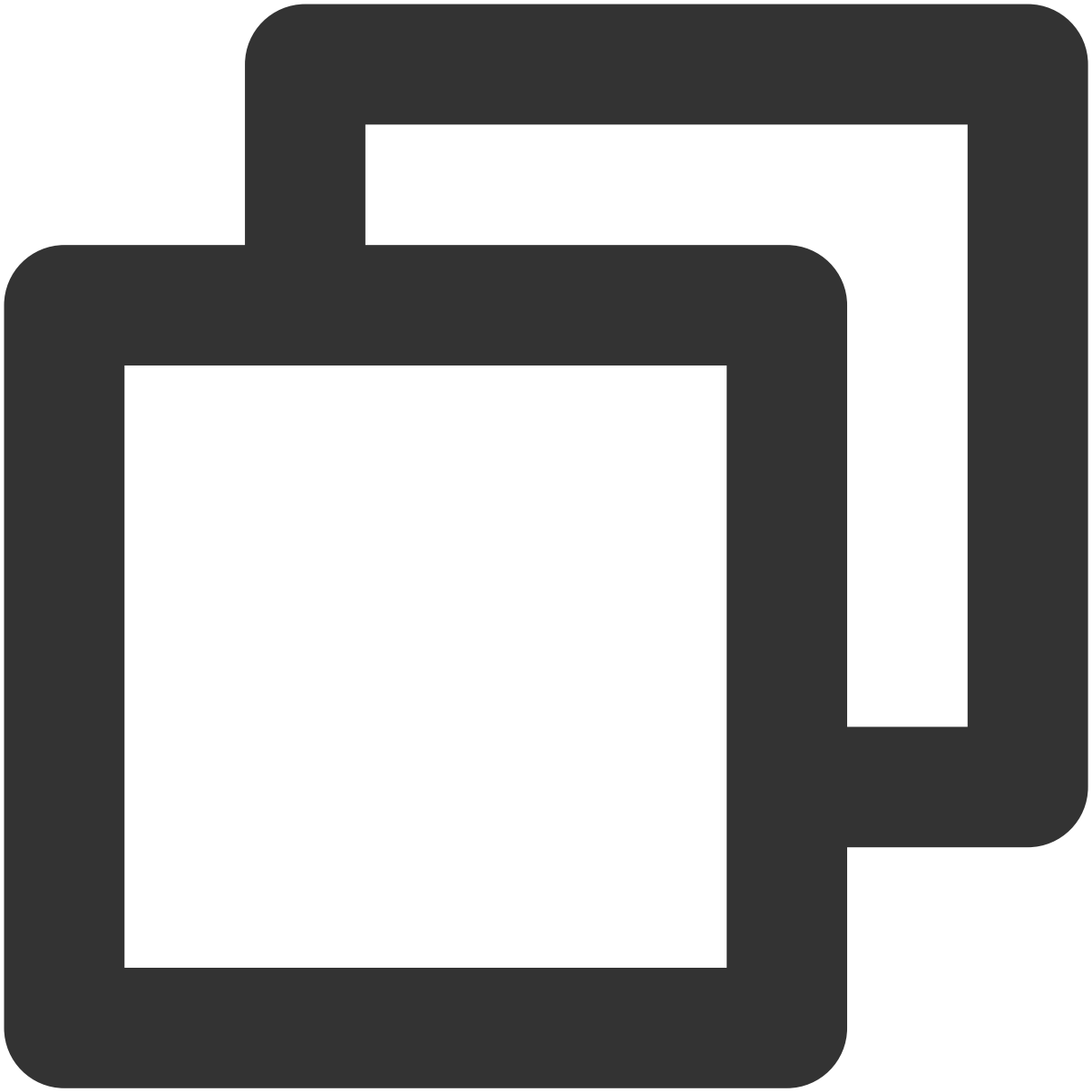
示例4. 自行编译打包并将 jar 包打到镜像

用户编写 spark 程序，打 jar 包同示例2，然后建立 Dockerfile 如下：



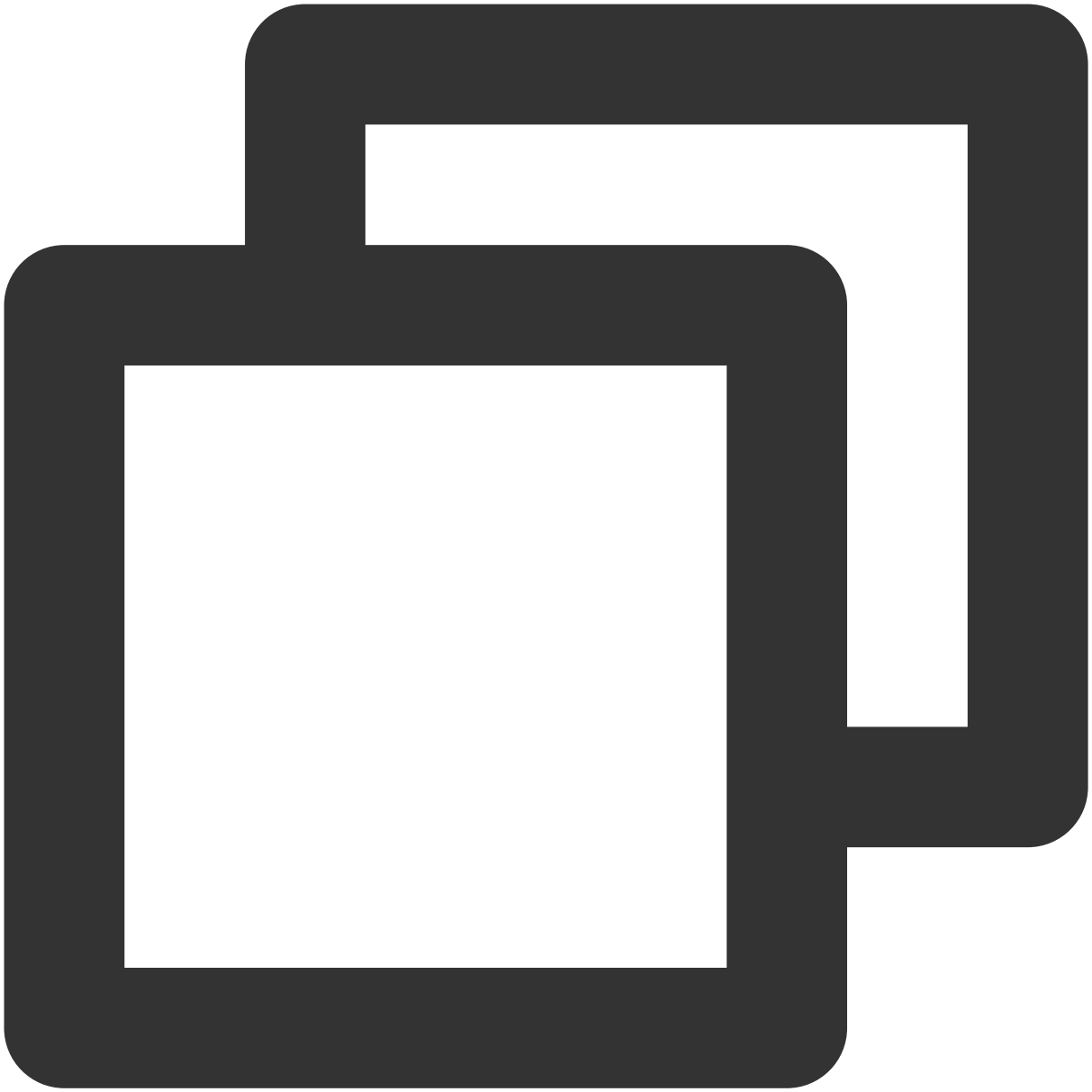
```
FROM ccr.ccs.tencentyun.com/emr-image/spark:BaseImage
USER root
RUN mkdir -p /sparkapp
COPY jars/wordcount.jar /sparkapp
ENTRYPOINT [ "/opt/entrypoint.sh" ]
```

需要继承基础镜像 `ccr.ccs.tencentyun.com/emr-image/spark:BaseImage`，该镜像包含了与 cos 交互所需要的 jar 包。



```
docker build -t ccr.ccs.tencentyun.com/emr-image/spark:wc -f ./bin/Dockerfile .
```

编写作业 yaml 如下：



```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: test4
spec:
  hadoopConf:
    "fs.cosn.userinfo.secretId":"$SecretId"
    "fs.cosn.userinfo.secretKey":"$SecretKey"
  type: Java
  mode: cluster
  mainClass: com.tencent.WordCountOnCos
```

```
  image: ccr.ccs.tencentyun.com/emr-image/spark:wc
  mainApplicationFile: "local:///sparkapp/wordcount.jar"
  arguments:
    - "cosn://kt-test-251007880/sparkapp/input/input"
    - "cosn://kt-test-251007880/sparkapp/output"
```

其中 image 为您打包的镜像，mainApplicationFile 是 jar 包在镜像中的路径。