Elastic MapReduce

EMR Lite HBase Operation Guide

Product Documentation

Contents

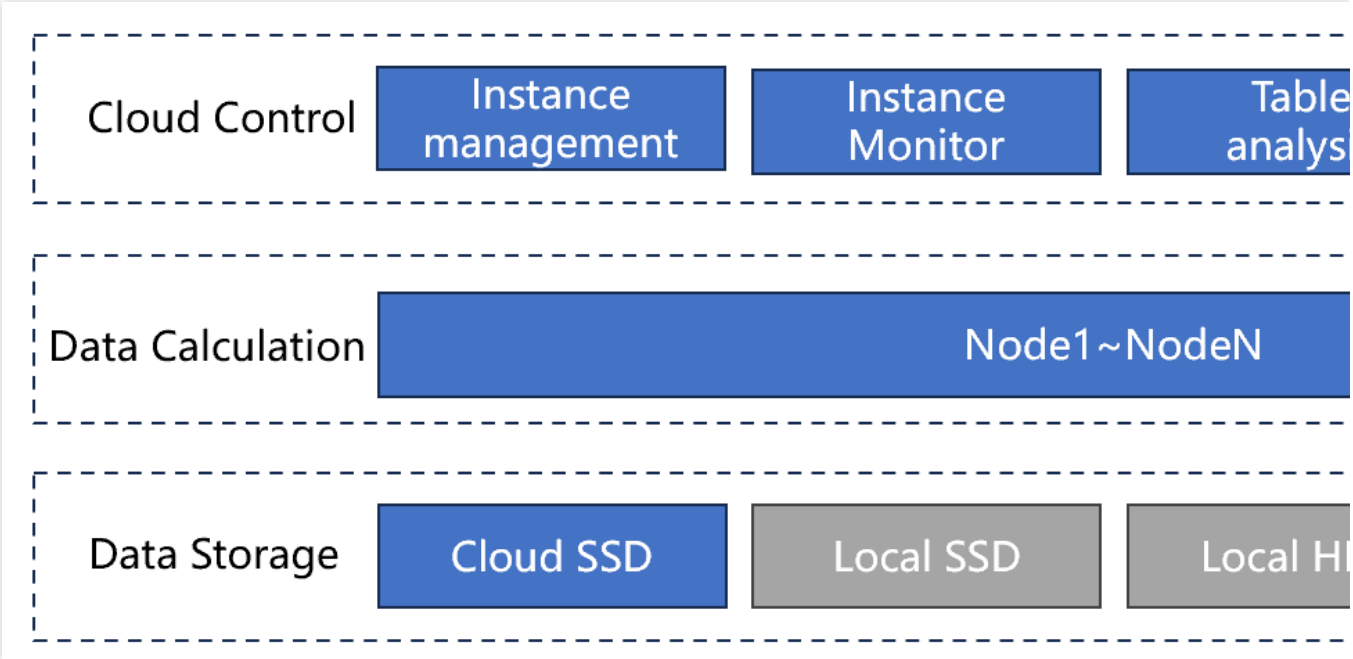
EMR Lite HBase Operation Guide
EMR Lite HBase Product Introduction

Last updated：2024-07-30 15:17:22

Product Introduction

The EMR Lite HBase provides a fully managed HBase service that is fully compatible with the open-source HBase protocol. You do not need to worry about the Ops of basic hardware facilities. Using features such as automatic failure recovery as well as backup and recovery provided by the Lite version, you can efficiently carry out big data platform Ops. With EMR Lite HBase, you can quickly build big data access applications with high throughput and low latency, such as real-time query of massive data, real-time data computing and processing, and time-series data storage and access.

Product Architecture



Cloud console: Provides visualized management of EMR Lite HBase instances, backup recovery, and monitoring and alarms for instance Ops.

Data computing: The instance computing capacity is provided by multiple nodes under the instance. After the instance is created, the number of instances can be expanded as needed.

Data storage: Data can be stored on various storage media with different performance levels. During the beta period, only performance cloud storage is supported.

Strengths

Fully compatible: Fully compatible with open source HBase. Allows seamless migration without code modification.
Simple Ops: No need to maintain underlying hardware resources. Built-in parameter tuning is out of the box.
Secure and reliable: VPC logical isolation, high availability architecture, and backup and recovery ensure data security.

Quotas and Limits

Last updated：2024-07-17 17:03:42

To ensure the normal operation of EMR Lite HBase instances, please carefully read the following content before use.

Quota Description

Object	Quota
Number of nodes per instance	Not more than 200

Restrictions

Object	Suggested Limit	Hard Limit
Number of tables per instance	Suggested to be within 1,000, not mandatory;	
Data Size in Table	Number of column families per table: 10 Single row key: 1 KB Column family: 10 B Single column qualifier: 100 B Single value in a table cell: 1 MB All values in a row: 100 MB	Single row key: 64 KB Column qualifier family: 64 KB Single column qualifier: 64 KB Single value in a table cell: 10 MB All values in a row: 1 GB
Number of rows in batch requests	Suggested to be within 5,000, not mandatory;	
ID length limit	Table name: 1-128 Column family: 1-16	Instance name: 6-36

Managing an Instance
Creating an Instance

Last updated：2024-07-30 15:17:48

Overview

This document introduces the directions and related configuration for creating an EMR Lite HBase instance through the EMR console.

Preparations

1. The EMR service has been authorized with the service role EMR_QCSRole.
2. The user has instance creation permissions.

Directions

Log in to the EMR Lite HBase console. Click create instance on the EMR Lite HBase instance list page, and complete the related configuration on the purchase page. When the instance status in the instance list shows as running, it means the instance has been successfully created.

Configuration Item	Configuration Item Description
Region	The physical data center where the instance is deployed. Each data center is independent and private, networks of the cloud services between different data centers do not interconnect. **Note:** 1. **After the instance is created, the region cannot be changed. Select carefully.** 2. **Currently, instance creation is supported only in Beijing, Shanghai, Guangzhou, Nanjing, and Jakarta regions.**
Instance Name	Set the instance name to distinguish instances for different purposes.
Billing Mode	Only pay-as-you-go billing is supported.
Instance Network	Select the corresponding private network in the region. If there is none, click create network to create one.

Availability Zone and Subnet	Select the availability zone and subnet for instance deployment. **Note: After the instance is created, the availability zone cannot be changed. Select carefully.**
Storage Type	Only performance cloud storage is supported.
Node Specifications	Select the appropriate single node specification and number of nodes.
Storage Capacity	Each node comes with a storage capacity of 100 GiB by default, adjustable in multiples of 20 GiB.
Tag	Tencent Cloud tags can be used to differentiate the purposes of different instances. Besides, cost allocation can be done by tags.

Instance Information

Last updated：2024-07-30 15:18:04

Overview

View the configuration information of instance specifications, networks, and tags.

Directions

1. Log in to the EMR Lite HBase console. Find the instance you need to terminate from the instance list, and click **Instance ID** to enter the **Instance Information** page.
2. You can view the instance's billing and tag information in the basic settings module.
3. You can view the instance's storage and compute specifications in the configuration information.
4. You can view the instance's Zookeeper access address in the access method.

Modifying an Instance

Last updated：2024-07-30 15:18:42

Overview

After the EMR Lite HBase instance is created, you can modify the number of instance nodes, modify the tag, and rename the instance without stopping the service.

Directions

Modifying Number of Nodes

1. Log in to the EMR Lite HBase console. Find the instance you need to modify from the instance list, and click the instance ID to access the instance information.
2. In the **Instance information - Configuration information** module, click **Adjust Node Quantity**.
3. After opening the panel in which the number of nodes can be adjusted, **increase or decrease the number of nodes** as needed.
4. Click **OK** to apply the changes**.**

Modifying Tags

1. Log in to the EMR Lite HBase Console. Find the instance you need to modify from the instance list, and click the instance ID to access the instance information.
2. In the **Instance Information - Basic Configuration** module, click **Edit** next to the tag information.
3. After you open the panel in which tags can be edited, add new tags or remove old tags for the instance as needed.
4. Click **Confirm** to apply the changes**.**

Renaming an Instance

1. Log in to the EMR Lite HBase console. Find the instance you need to modify from the instance list.
2. Move the mouse over the instance name to be modified, and click **Edit Icon**.
3. After you open the panel in which the instance name can be modified, enter the new instance name.
4. Click **Confirm** to save.

Terminating an Instance

Last updated：2024-07-30 15:18:59

Overview

When business changes do not require the EMR Lite HBase instance, you can release resources by terminating the instance.

Note:

After the instance is terminated, all data within the instance will be cleared. Therefore, confirm the risks and proceed with caution.

Directions

1. Log in to the EMR Lite HBase console. Find the instance you need to terminate from the instance list, and click the **Terminate** button in the action bar.
2. After you open the panel of termination confirmation, confirm that the instance can be terminated, and click **Confirm** to initiate the termination process.

Monitoring and Alarms
Instance Monitoring

Last updated：2024-07-30 15:29:40

Overview

View the running metrics of the HBase instance.

Directions

1. Log in to the EMR Lite HBase console. Find the instance you need to view from the instance list, and click **Monitor** to enter the **Instance Monitoring Page**.
2. After entering the instance monitoring page, you can view the HBase running metrics as needed.

For update information, see HBase monitoring metrics.

Data Table Analysis

Last updated：2024-07-30 15:30:00

Overview

View the request trends of HBase tables and Region to analyze the sources of data hotspots.

Directions

1. Log in to the EMR Lite HBase console. Find the instance you need to view from the instance list, and click **Instance ID** to enter the **Instance Information** page.
2. In the left menu, click **Diagnostic Analysis** and select **Data Table Analysis** to view.

Configuring Alarms

Last updated：2024-07-30 15:31:20

Overview

Elastic MapReduce (EMR) has been connected to Tencent Cloud Observability Platform (TCOP). You can configure alarm policies for EMR nodes and service monitoring metrics in the TCOP console.

Directions

1. Log in to the TCOP console. In the left sidebar, choose **Alert Management > Alert Configuration**.
2. In the policy management page, click **New Policy**.
3. In the pop-up **New Alert Policy** window, see the table below to configure basic information, alarm rules, and create a notification template.

Configuration Type	Configuration Item		Description
Basic information	Policy name		Custom policy name
	Remarks		Custom policy remarks
Configure alarm policy	Monitoring type		Select **Cloud Service Monitoring**
	Project associated with policy		After setting the project association, you can quickly filter alarm policies belonging to that project in the alarm policy list.
	Policy type		Select the desired policy type for monitoring Tencent Cloud products.
	Alert object		If you select an instance ID, the alarm policy will be bound to the selected instance. If you select an instance group, the alarm policy will be bound to the selected instance group. If you select all objects, the alarm policy will be bound to all instances that the current account has permission on.
	Trigger condition	Manual configuration (metric alarm)	Consists of metric, comparison, threshold, statistical period, and the number of consecutive periods. You can expand the trigger condition to view the metric trend, and based on which, set a proper threshold.

		Manual configuration (event alarm)	Create an event alarm policy to get notifications and take appropriate actions in case of service resources or underlying infrastructure exceptions.
		Select template	Select the template button and select a configured template from the drop-down list. For specific configurations, see configuring trigger condition template.
Configure alarm notification	Notification template		By default, a preset notification template is bound (recipients are the main account administrators, and delivery channels are SMS and email). Up to three notification templates can be bound to each alarm policy. For more information about notification template configurations, see New Notification Template.

4. After the form is completed, click **Complete**.

Development Guide
Lite HBase Instructions

Last updated：2024-07-30 15:32:04

Overview

Access the EMR Lite HBase instance via Shell, Java, and Golang for table creation and queries.

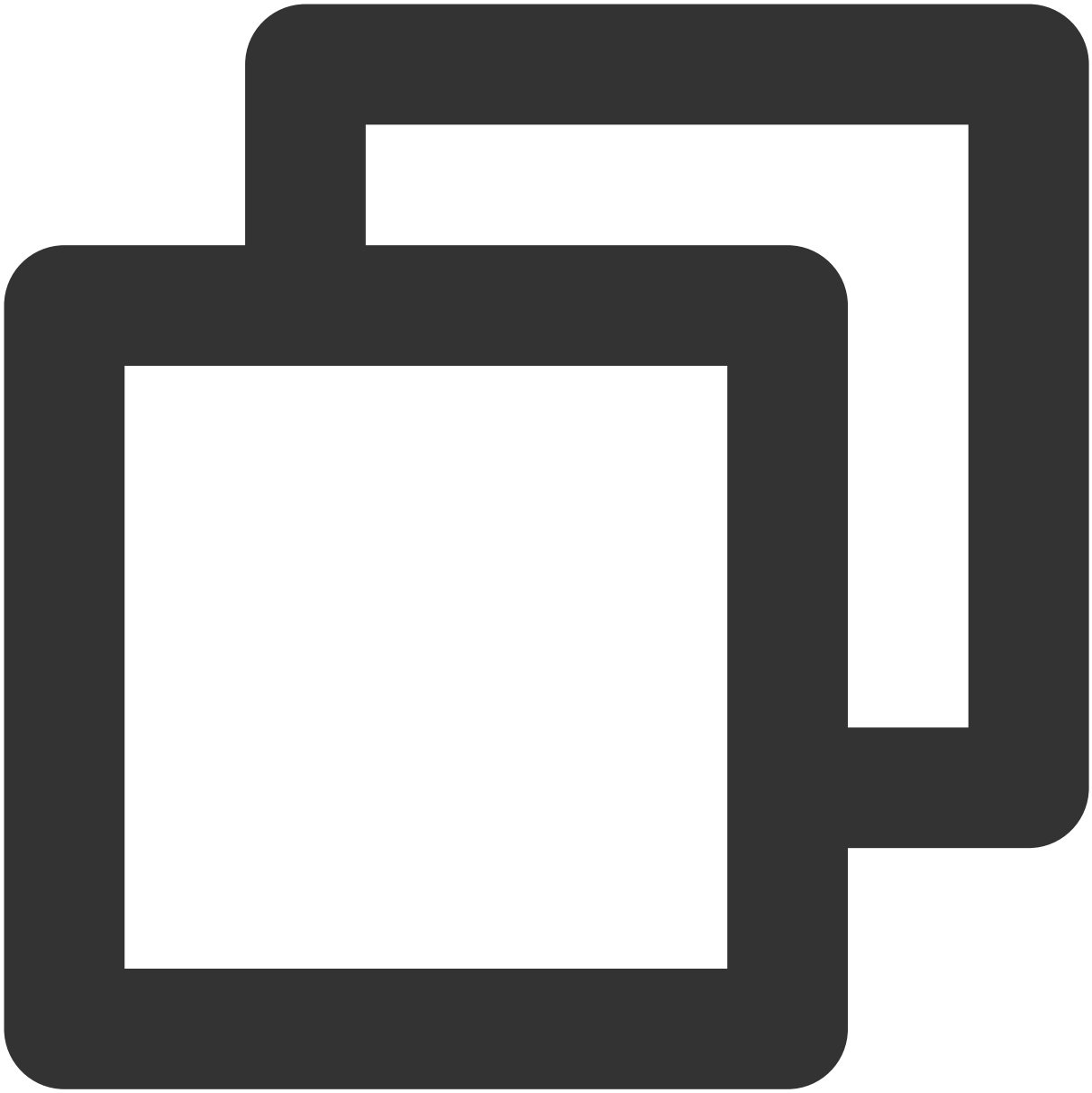
Preparations

An EMR Lite HBase instance has been created and is in the running status.
A CVM instance sharing the same VPC and same subnet with the EMR Lite HBase instance has been created (hereinafter referred to as the CVM client).
The Java environment has been installed, and environment variables have been configured on the CVM client. A JDK version 1.8 or above can be installed as required. If it is not installed, obtain the installation address and configure Java environment .
The Zookeeper address of the Lite HBase instance has been obtained from the EMR Lite HBase instance information page.
Note:
For more information on using the HBase API, see the Apache HBase official website: https://hbase.apache.org/2.4/apidocs/index.html

Using HBase Shell

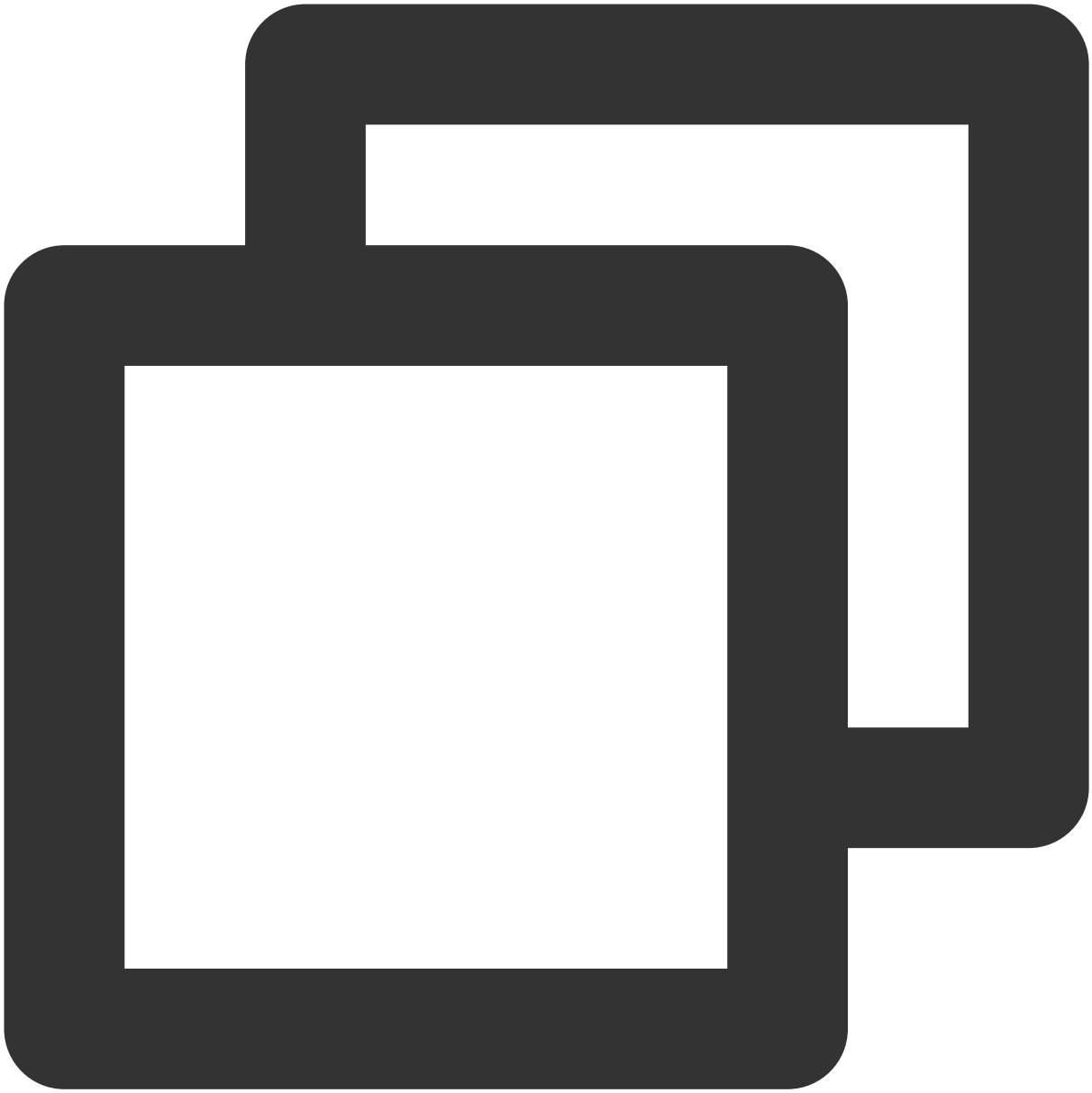
Configuring Environment

1. Log in to the CVM client, then download the HBase client, extract it, and switch to the emr-lite-hbase-client directory.



```
cd /usr/local
wget https://emr-lite-hbase-1259353343.cos.ap-guangzhou.myqcloud.com/client/emr-lit
tar -zxvf emr-lite-hbase-client.tar.gz
cd emr-lite-hbase-client
```

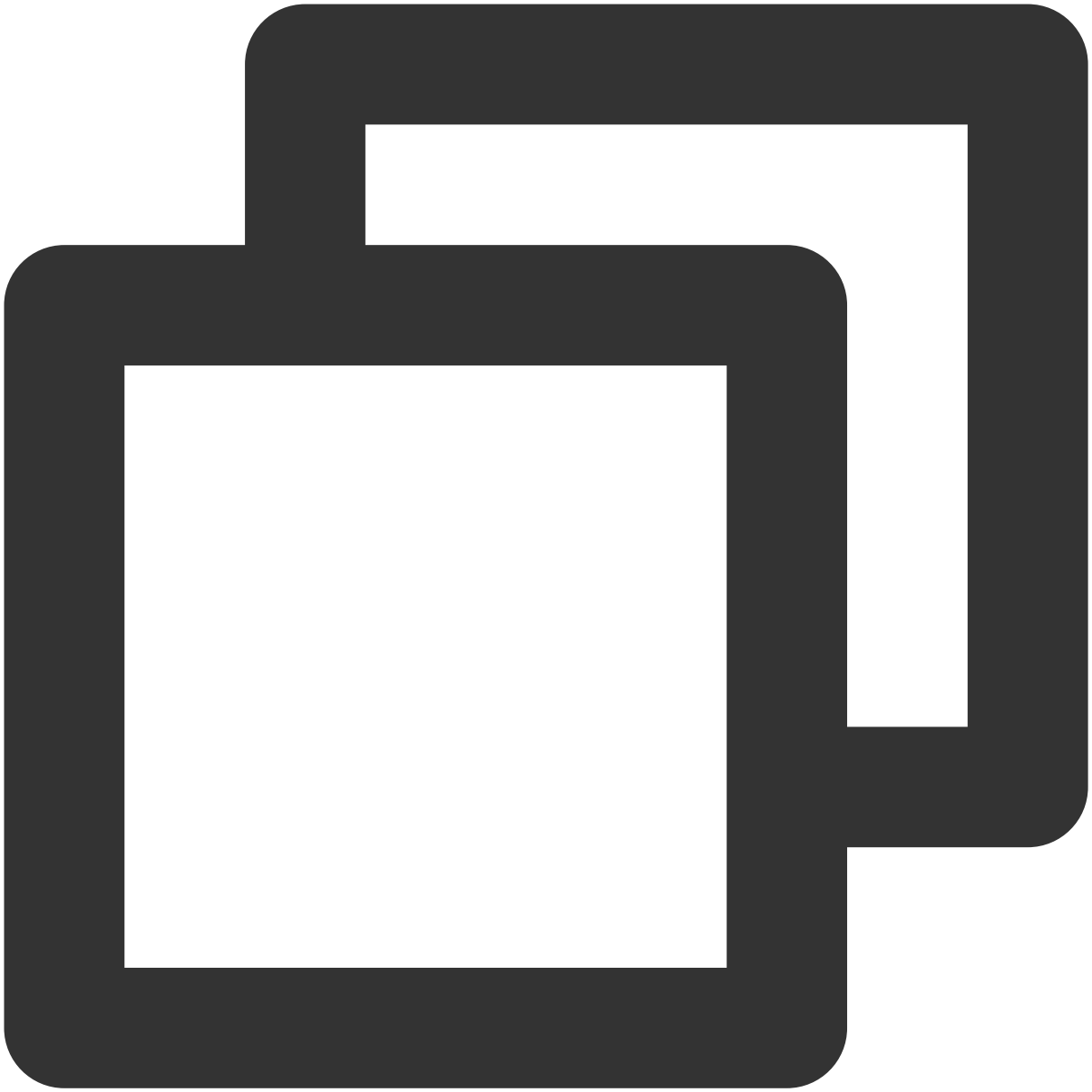
2. Modify the hbase.zookeeper.quorum parameter configuration in conf/hbase-site.xml. $quorum is the Zookeeper address.



```
vi conf/hbase-site.xml
<property>
       <name>hbase.zookeeper.quorum</name>
        <value>$quorum</value>
</property>
```

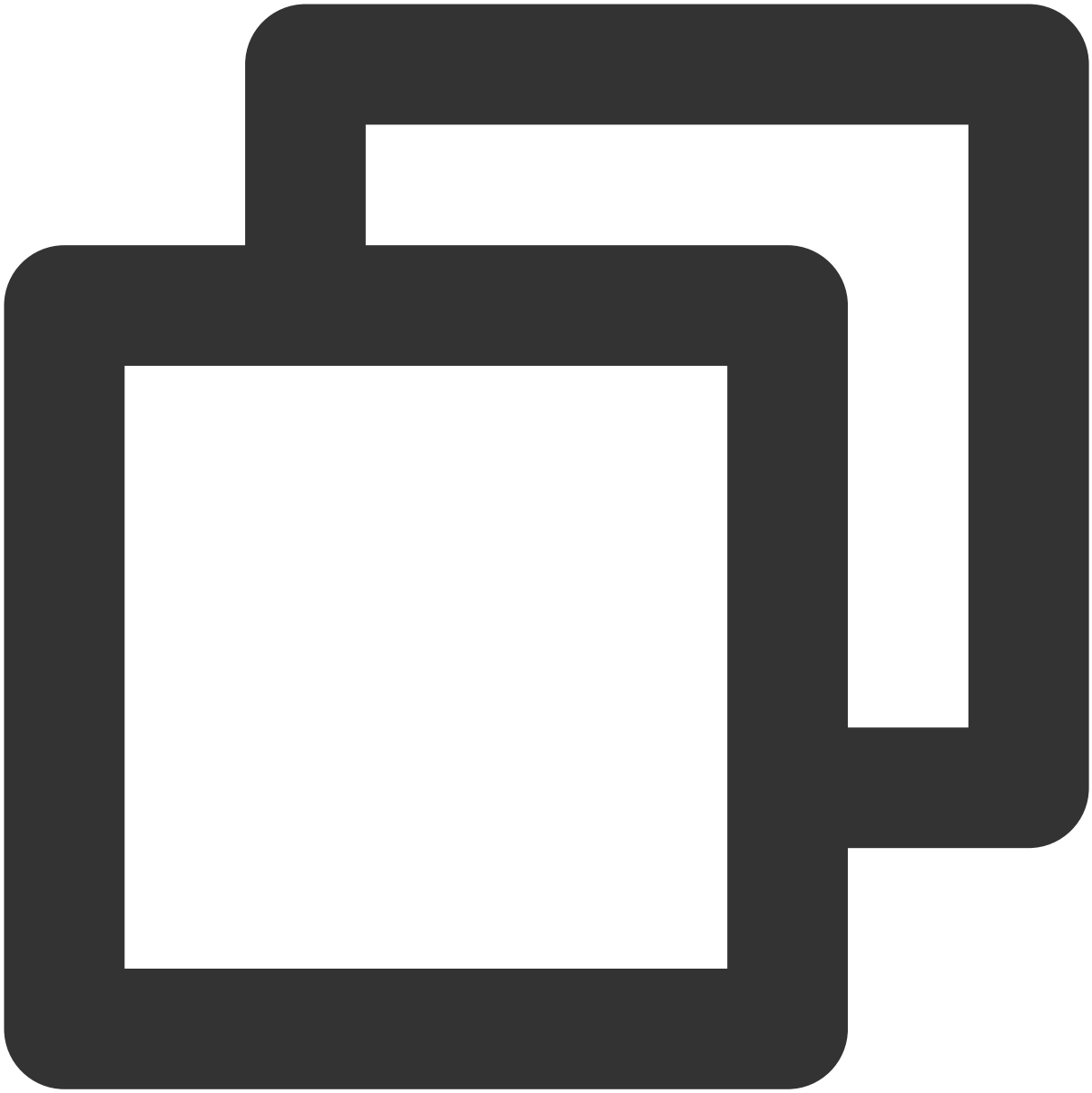
HBase Basic Operations

1. Use the following command to enter the HBase Shell.



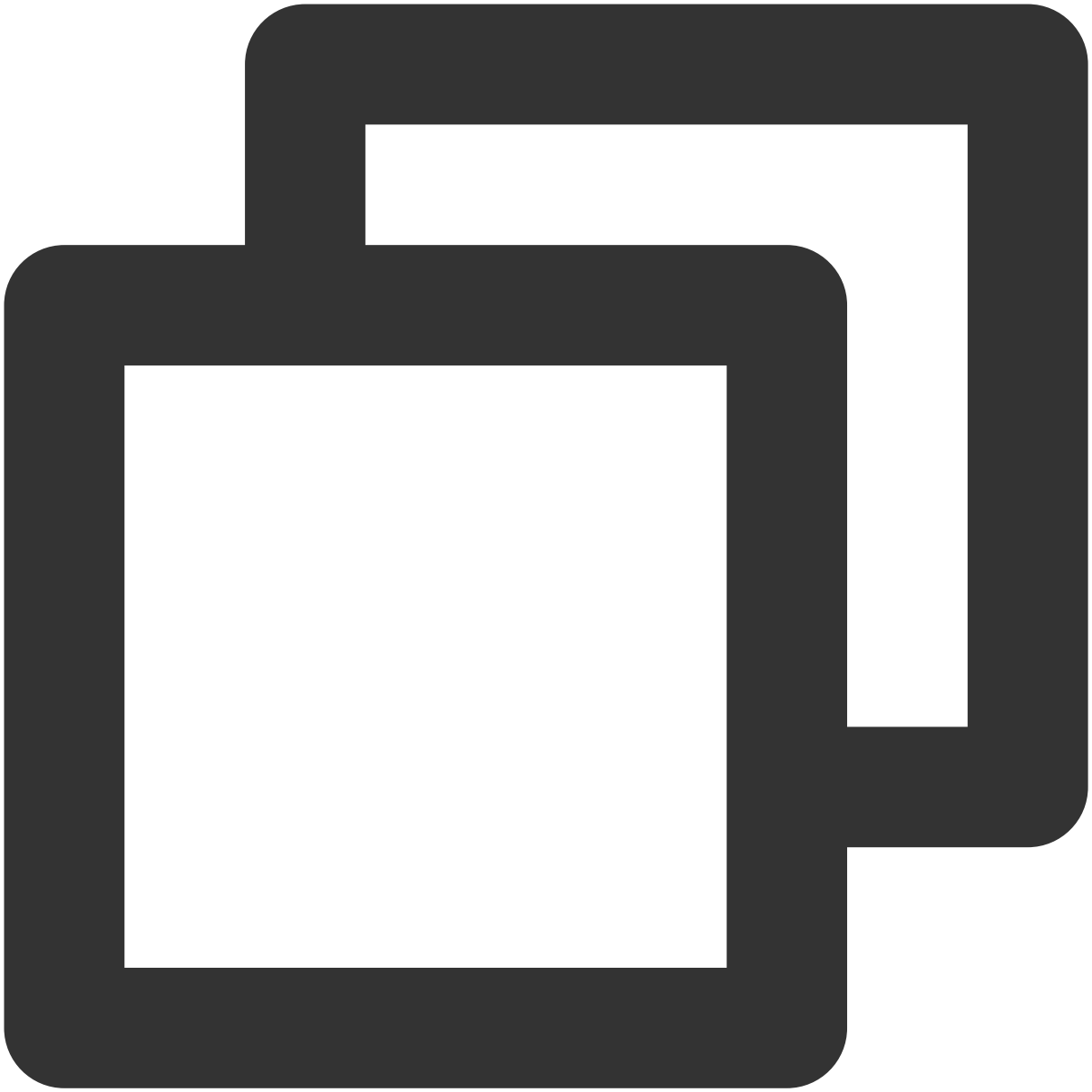
```
./bin/hbase shell
```

2. In the HBase Shell, type help to view basic usage information and example commands. You can use the following command to create a table.



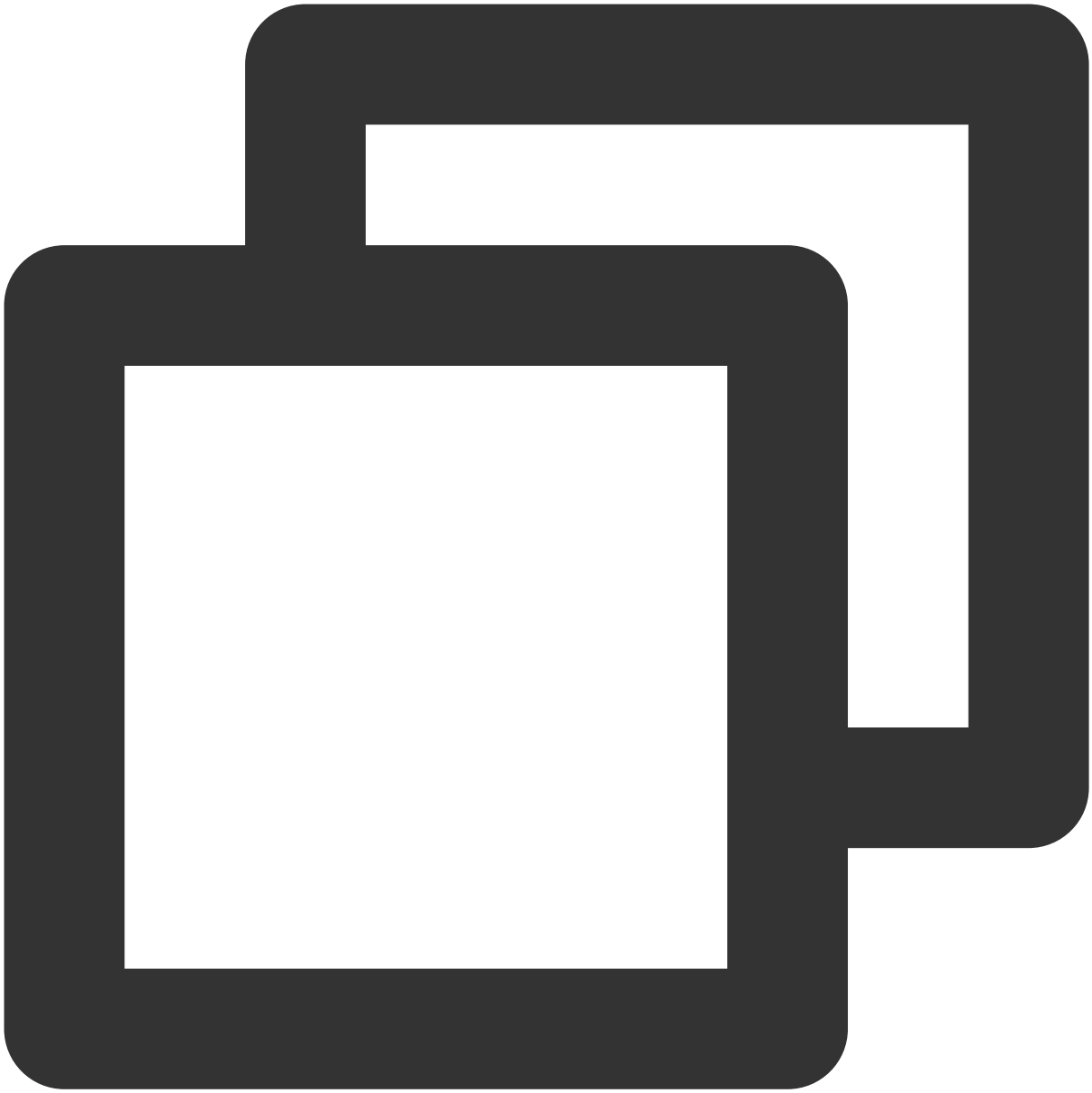
```
hbase:001:0> create 'test', 'cf'
Created table test
Took 1.5703 seconds
=> Hbase::Table - test
```

3. After the table is created, you can use the list command to check if the table you created exists.



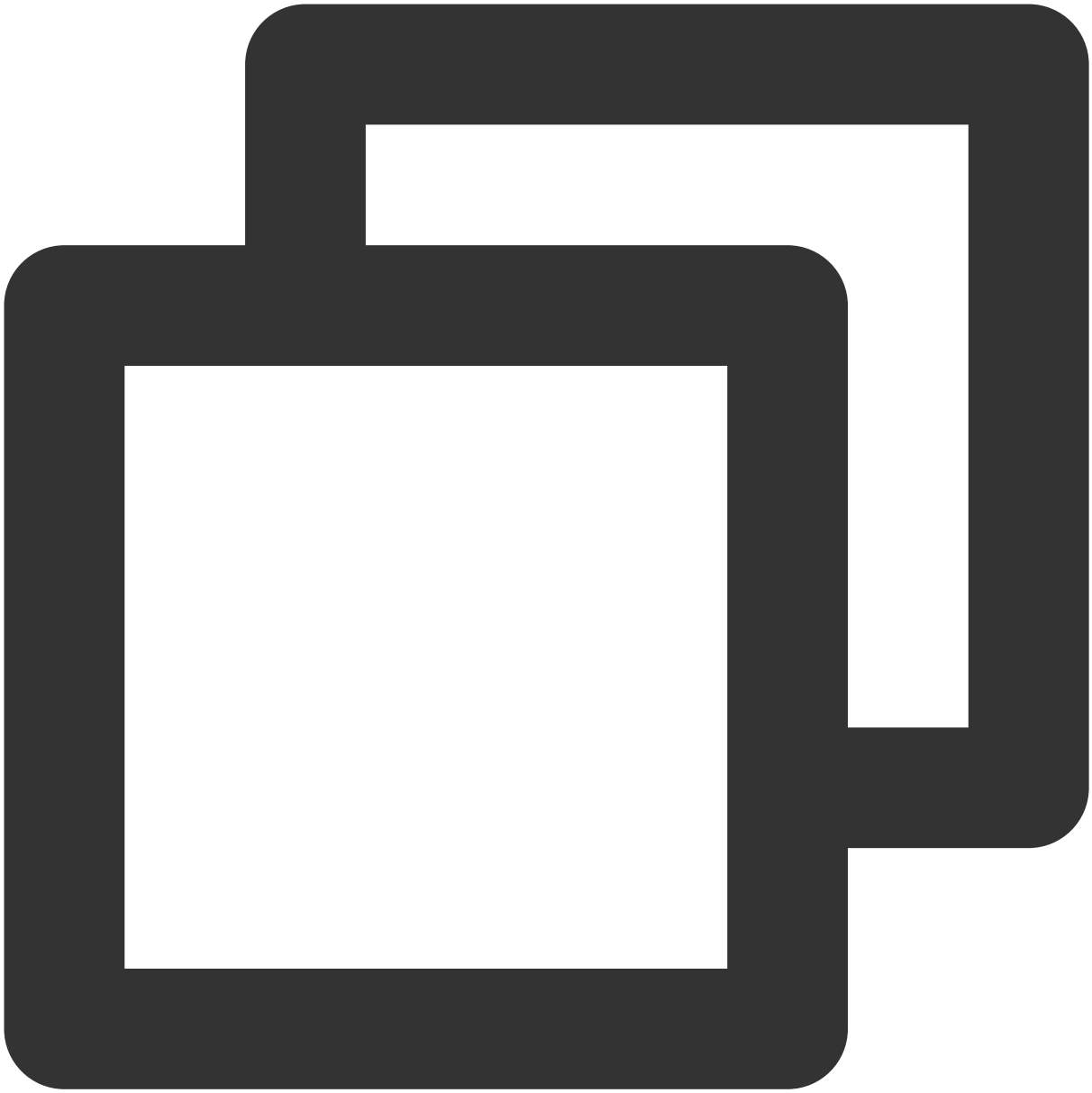
```
hbase:002:0> list 'test'
TABLE
test
1 row(s)
Took 0.0158 seconds
=> ["test"]
```

4. Use the put command to add data to the table you created. Three values were inserted into the table you created. The first value "value1" was inserted into the column "cf:a" of the row "row1", and so forth.



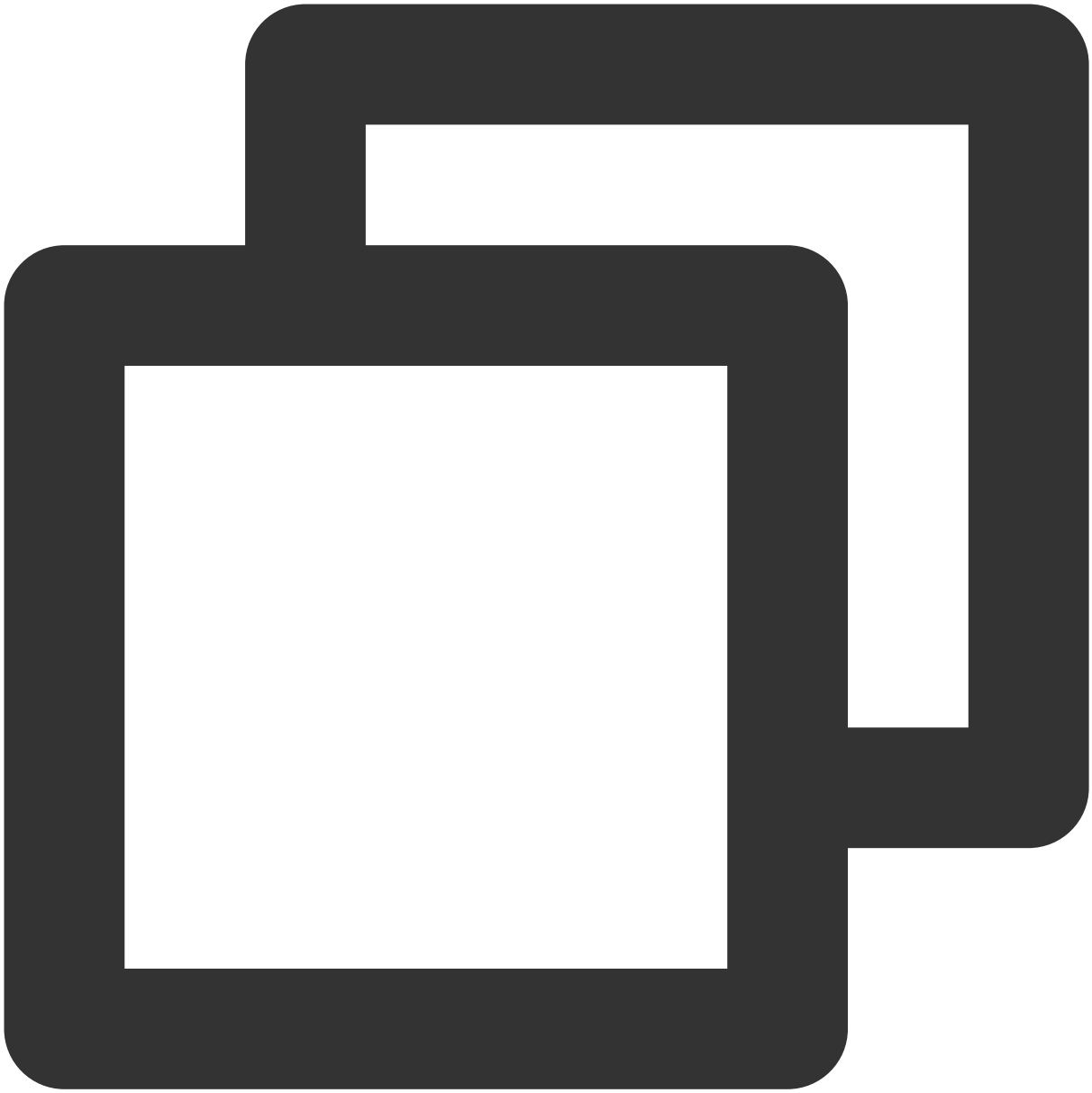
```
hbase:003:0> put 'test', 'row1', 'cf:a', 'value1'
Took 0.1766 seconds
hbase:004:0> put 'test', 'row2', 'cf:b', 'value2'
Took 0.0160 seconds
hbase:005:0> put 'test', 'row3', 'cf:c', 'value3'
Took 0.0149 seconds
```

5. Use the scan command to scan the entire table.



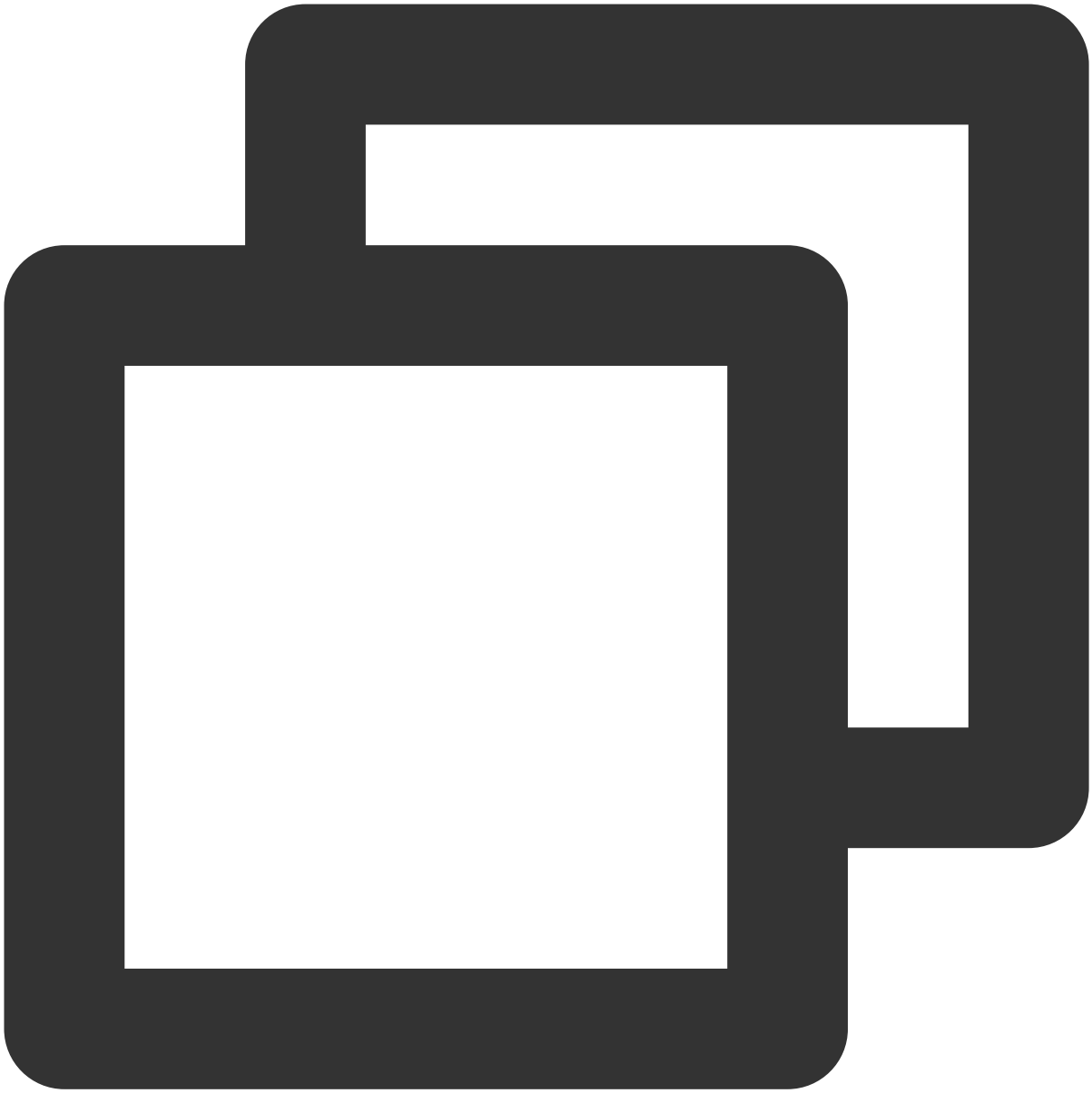
```
hbase:006:0> scan 'test'
ROW                                   COLUMN+CELL
 row1                                  column=cf:a, timestamp=2024-07-16T10:56:28.027,
 row2                                  column=cf:b, timestamp=2024-07-16T10:56:40.658,
 row3                                  column=cf:c, timestamp=2024-07-16T10:56:51.744,
3 row(s)
Took 0.0682 seconds
```

6. Use the get command to retrieve the value of a specified row in the table.



```
hbase:007:0> get 'test', 'row1'
COLUMN                                 CELL
 cf:a                                  timestamp=2024-07-16T10:56:28.027, value=value1
1 row(s)
Took 0.0361 seconds
```

7. Use the drop command to delete a table. Before deleting it, you need to use the disable command to disable the table.

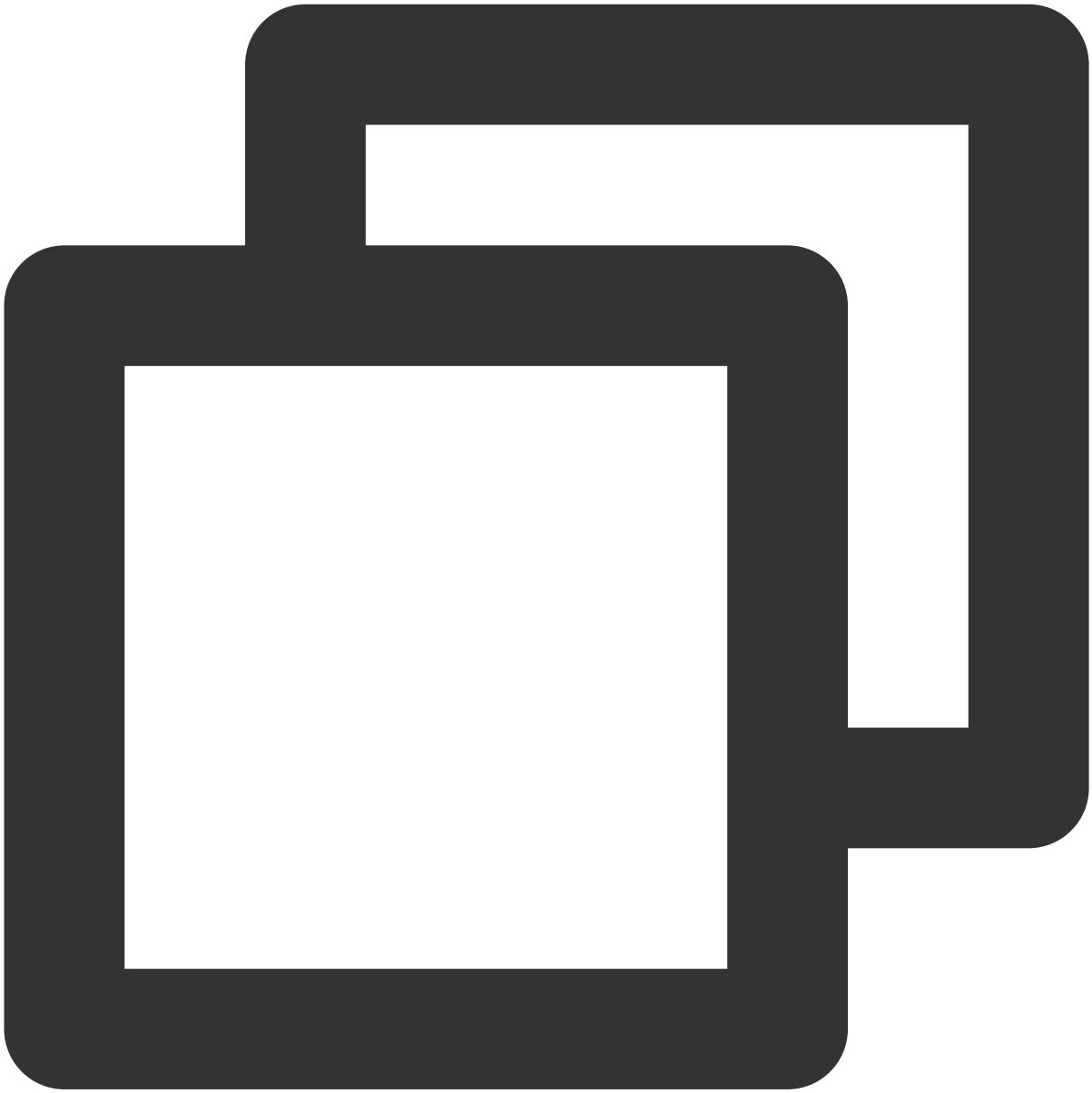


```
hbase:008:0> disable 'test'
Took 0.6919 seconds
hbase:009:0> drop 'test'
Took 0.3451 seconds
```

8. Finally, you can use the exit command to close the HBase shell.

Using the Java API

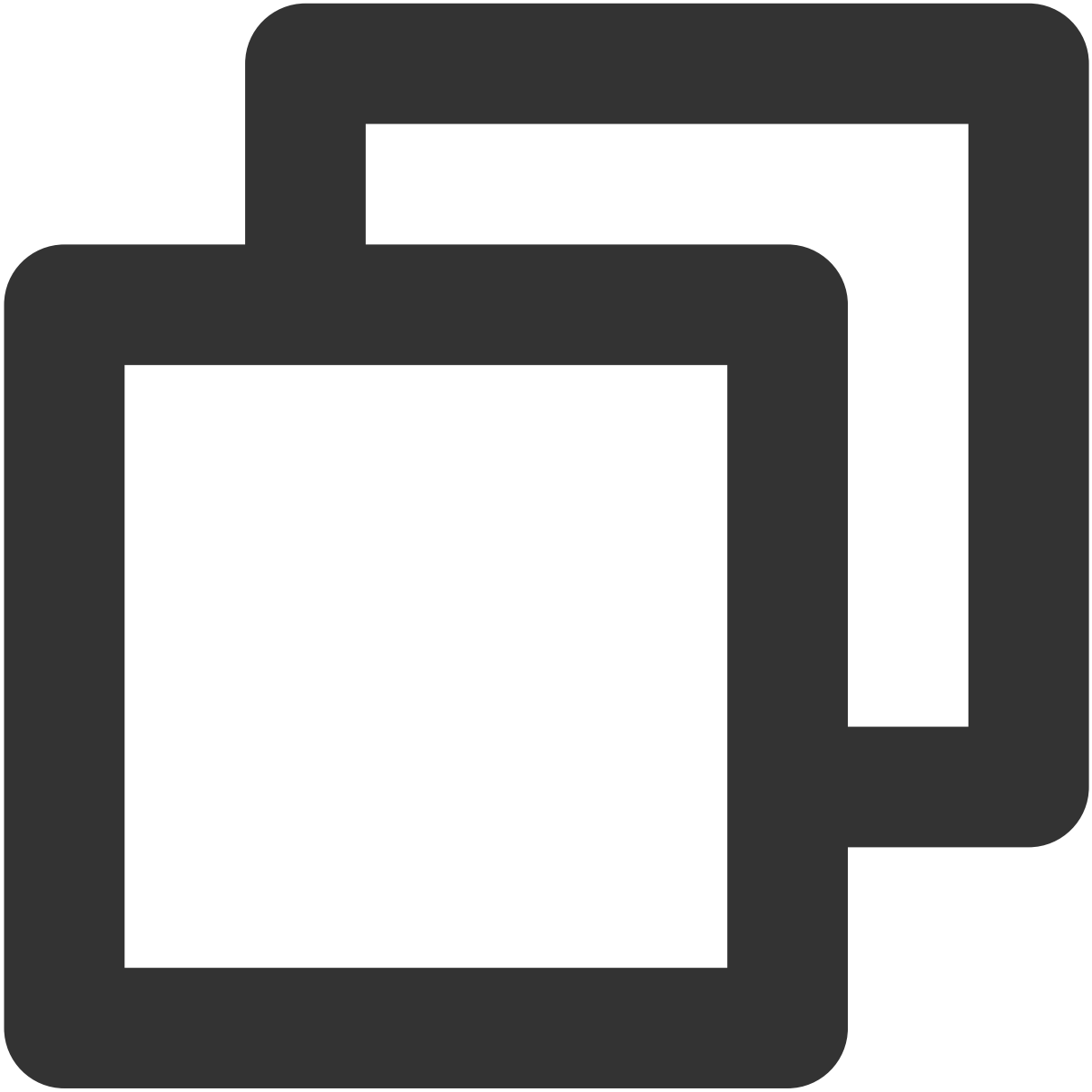
Ensure that an appropriate JDK version is available in the development environment, and install and configure Maven's environment variables. If you are using an IDE, set the Maven-related configurations in the IDE. If needed, use the Tencent Cloud image source to accelerate Maven, and add the following repository mirror to the Maven configuration file settings.xml.



```
<mirror>
    <id>nexus-tencentyun</id>
    <mirrorOf>*</mirrorOf>
    <name>Nexus tencentyun</name>
   <url>http://mirrors.cloud.tencent.com/nexus/repository/maven-public/</url>
</mirror>
```

Creating a Maven Project

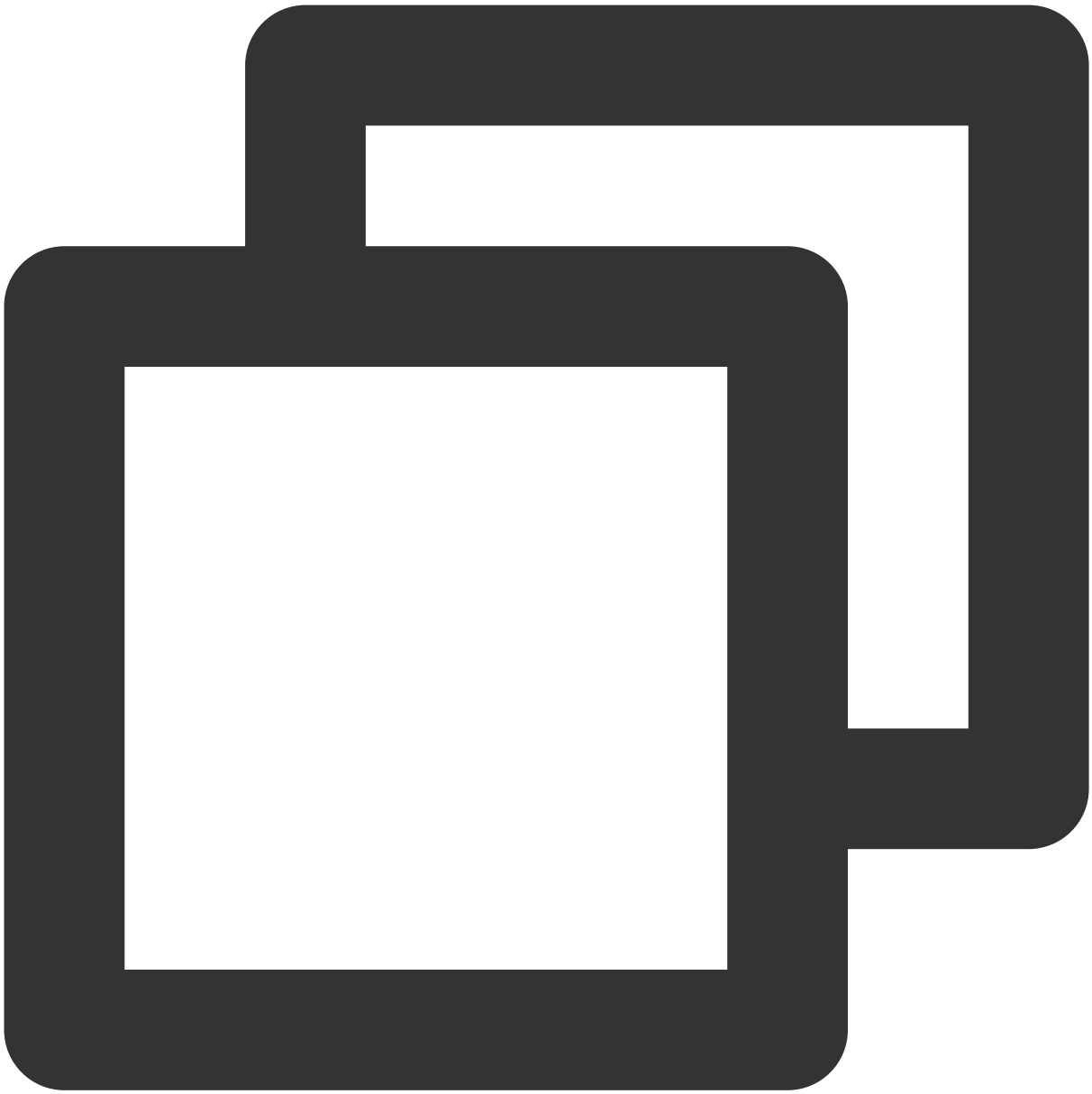
Enter the directory where you want to create the project on the command line and run the following command to create a Maven project.



```
mvn archetype:generate -DgroupId=com.tencent.cloud-DartifactId=test-litehbase -Dver
```

After creation, a project folder named test-litehbase will be generated in the project directory. The file structure within is as follows. The pom.xml file is mainly used for dependency management and packaging configuration, while the

Java folder contains your source code.

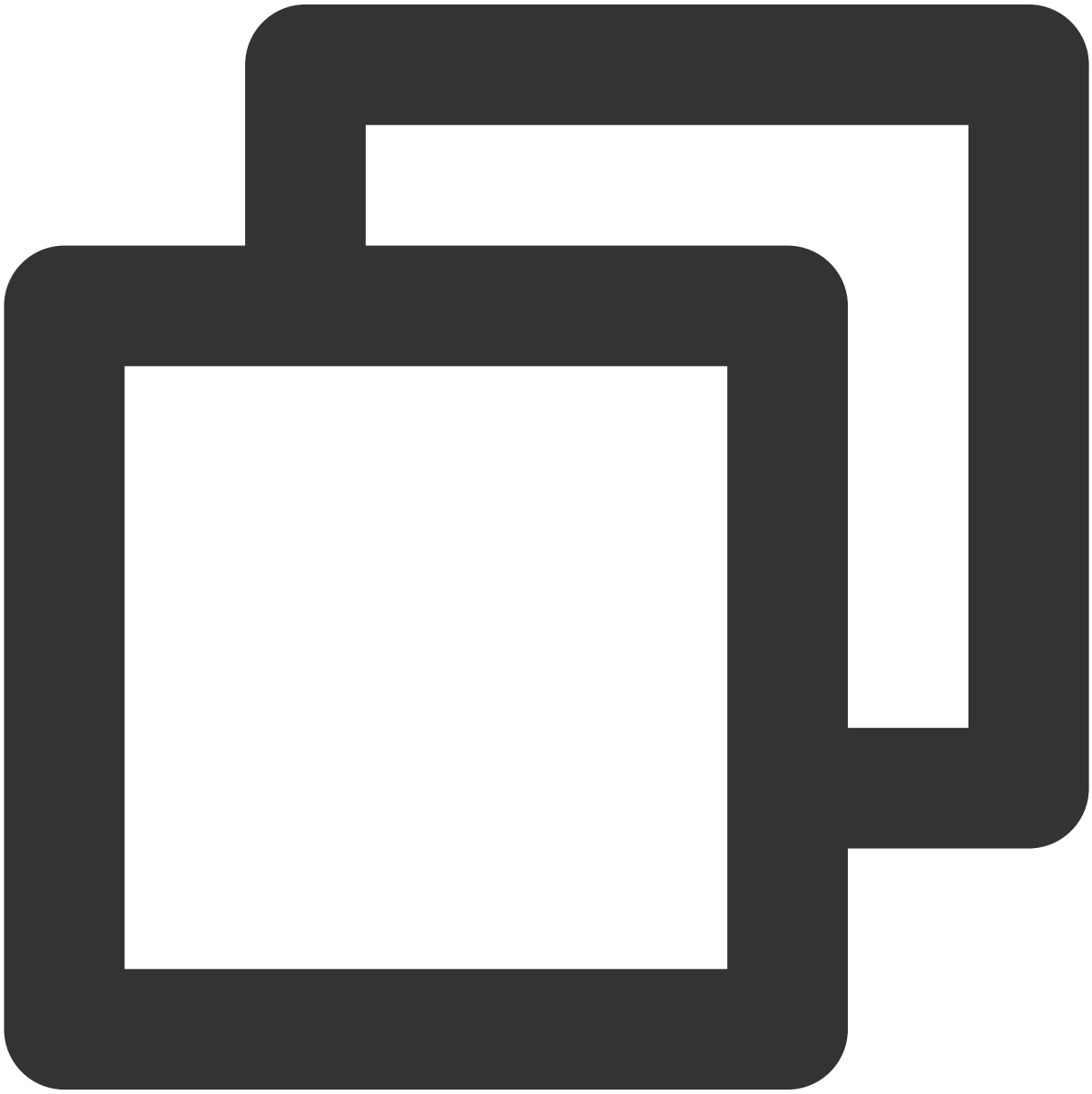


```
test-litehbase
├── pom.xml
└── src
    ├── main
    │   └── java
    │       └── com
    │           └── tencent
    │               └── cloud
    │                   └── App.java
```

```
    └── test
        └── java
            └── com
                └── tencent
                    └── cloud
                        └── AppTest.java
```

Adding Configuration and Sample Code

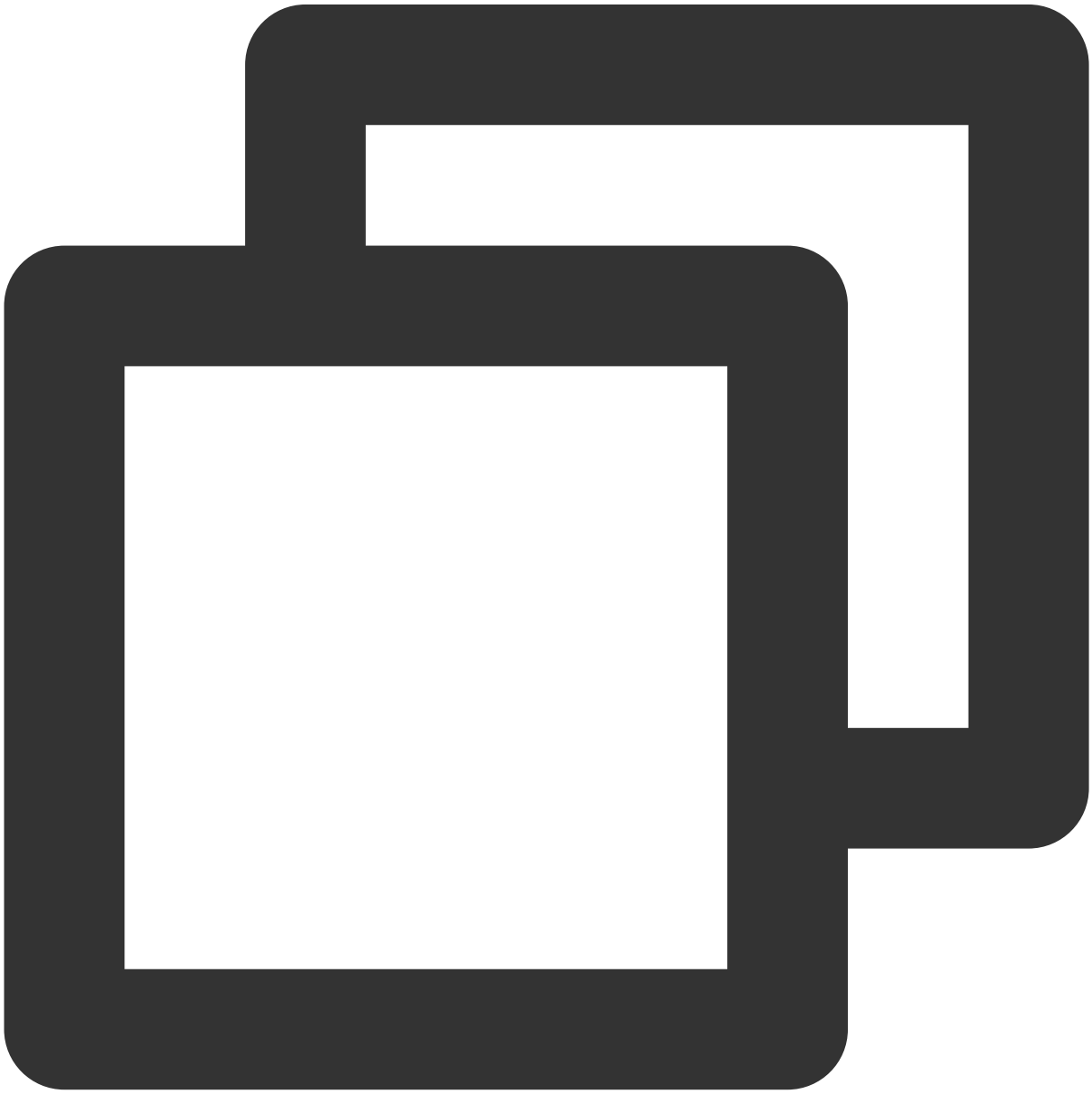
Add Maven dependency, packaging, and compilation plugins to the pom.xml file.



```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.apache.hbase/hbase-client -->
    <dependency>
       <groupId>org.apache.hbase</groupId>
       <artifactId>hbase-client</artifactId>
        <version>2.4.5</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/log4j/log4j -->
    <dependency>
        <groupId>log4j</groupId>
       <artifactId>log4j</artifactId>
        <version>1.2.17</version>
    </dependency>
</dependencies>
<build>
    <plugins>
        <plugin>
           <groupId>org.apache.maven.plugins</groupId>
           <artifactId>maven-compiler-plugin</artifactId>
           <version>3.10.1</version>
            <configuration>
                <source>8</source>
                <target>8</target>
               <encoding>utf-8</encoding>
            </configuration>
        </plugin>
        <plugin>
           <artifactId>maven-assembly-plugin</artifactId>
            <configuration>
                <descriptorRefs>
                   <descriptorRef>jar-with-dependencies</descriptorRef>
                </descriptorRefs>
                <archive>
                    <manifest>
                        <!-- Specify the class containing the main method entry poi
                       <mainClass>com.tencent.cloud.App</mainClass>
                    </manifest>
                </archive>
            </configuration>
            <executions>
                <execution>
                   <id>make-assembly</id>
                   <phase>package</phase>
                    <goals>
                       <goal>single</goal>
```

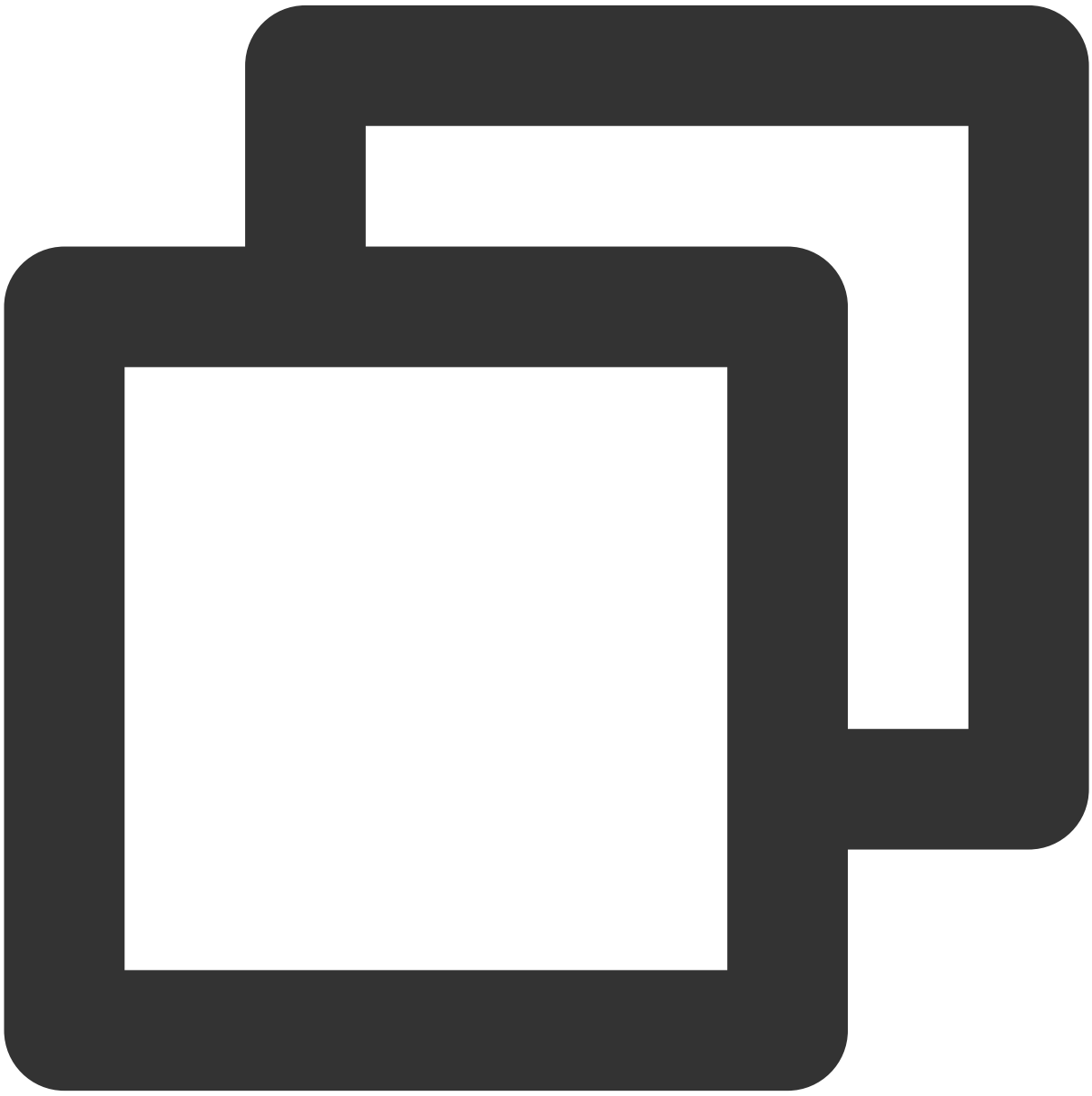
```
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

[Optional] If you encounter sample execution failures during debugging and need to obtain logs from standard output to pinpoint errors, or if you wish to save operation logs to the specified directory target/lite-hbase.log, you can create a Java configuration directory named resources under the main folder, create a log4j.properties file within it, and add configuration details for the log4j log information printing module in the log4j.properties file.



```
log4j.rootLogger=INFO, stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d %p [%c] - %m%n
log4j.appender.logfile=org.apache.log4j.FileAppender
log4j.appender.logfile.File=target/lite-hbase.log
log4j.appender.logfile.layout=org.apache.log4j.PatternLayout
log4j.appender.logfile.layout.ConversionPattern=%d %p [%c] - %m%n
```

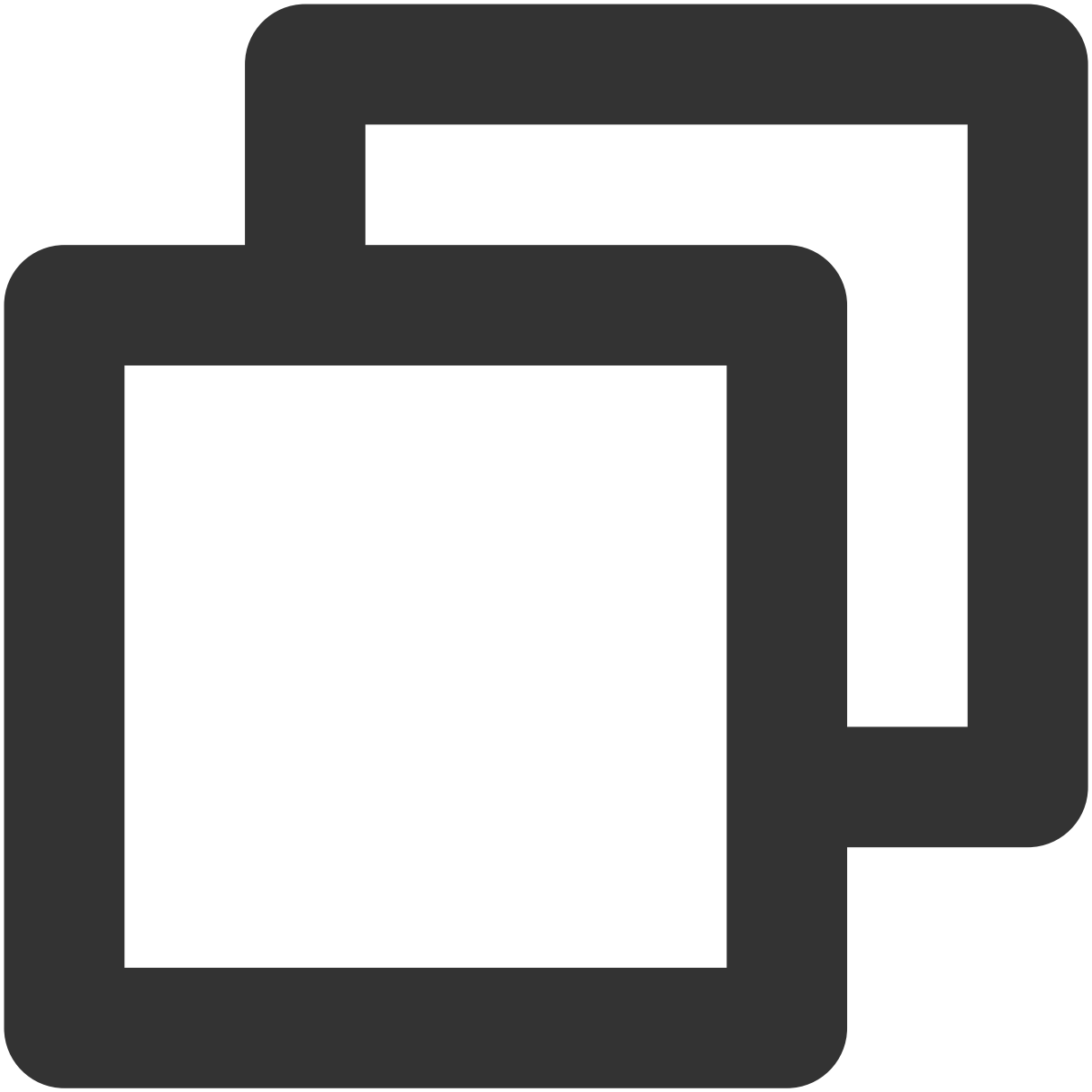
Sample code in App.java is as follows.



```
package com.tencent.cloud;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.*;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.Bytes;

import java.io.IOException;

/**
 * Access the Lite HBase instance through the HBase Java API.
 */
public class App {
    public static void main(String[] args) throws IOException {
        // Obtain $quorum from the Zookeeper access address found in the access met
        // Example: conf.set("hbase.zookeeper.quorum", "10.0.0.8,10.0.0.11,10.0.0.5
        Configuration conf = HBaseConfiguration.create();
       conf.set("hbase.zookeeper.quorum", "$quorum");

        // Establish a connection between the EMR Lite HBase client and the data.
        Connection connection = ConnectionFactory.createConnection(conf);
        Admin admin = connection.getAdmin();

        // Once the connection is established, you can access the Lite HBase instan
        TableDescriptorBuilder tableBuilder = TableDescriptorBuilder.newBuilder(Tab
        ColumnFamilyDescriptor cf = ColumnFamilyDescriptorBuilder.of("cf");
        tableBuilder.setColumnFamily(cf);
        TableDescriptor table = tableBuilder.build();

        System.out.println("Creating Table.");
        if (admin.tableExists(table.getTableName())) {
           admin.disableTable(table.getTableName());
           admin.deleteTable(table.getTableName());
        }
        admin.createTable(table);
        System.out.println("Inserting Data.");
        Table table1 = connection.getTable(TableName.valueOf("test1"));
        Put put1 = new Put(Bytes.toBytes("row1"));
       put1.addColumn(Bytes.toBytes("cf"), Bytes.toBytes("a"),
               Bytes.toBytes("value1"));
```

Compiling Code and Packaging It for Upload

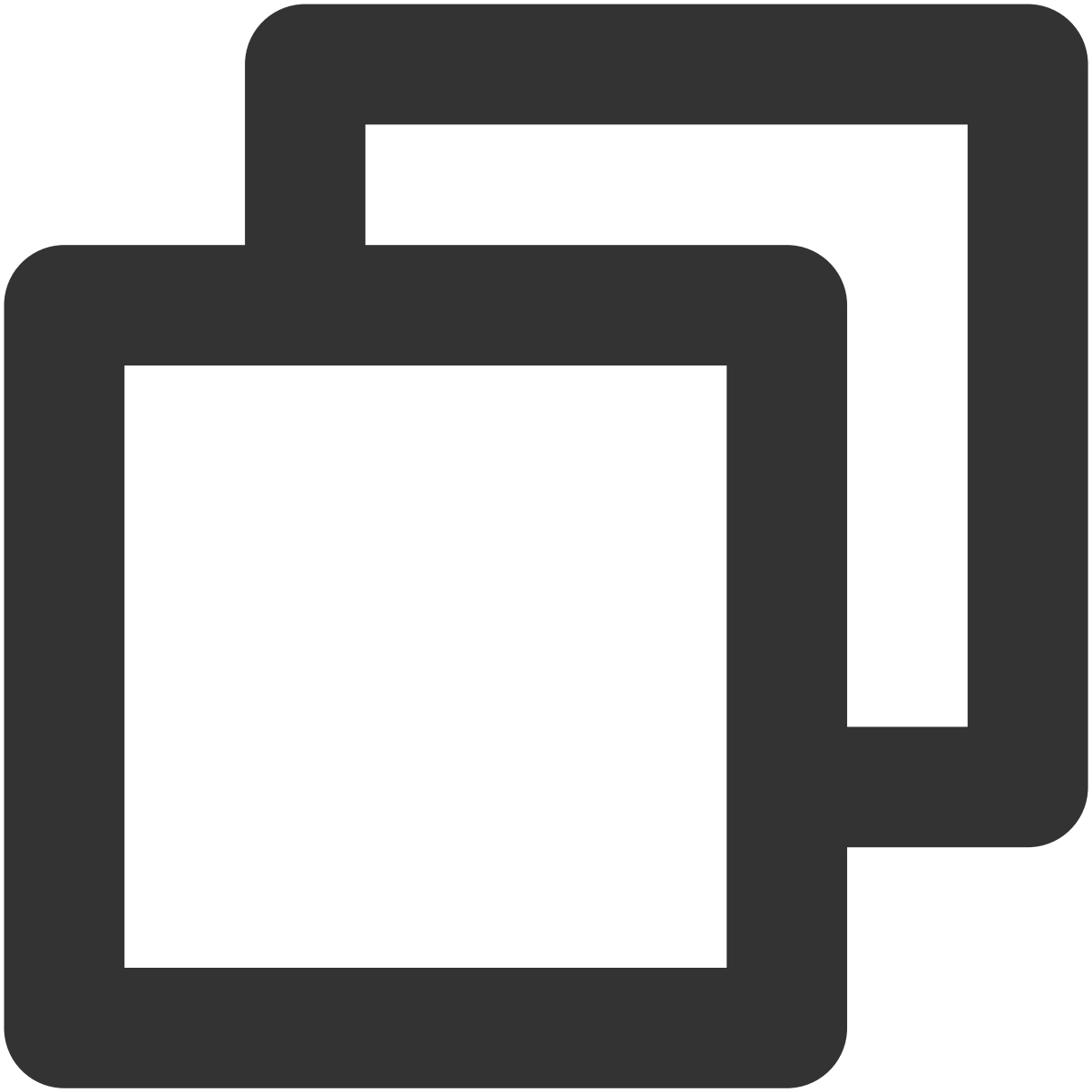
Use the local command line to enter the project directory and execute the following command to compile and package the project.



```
mvn package
```

The log printing build success indicates that the operation was successful. You can find the packaged JAR file in the target folder of the project directory. You need to upload the JAR file with the with-dependencies suffix to the CVM client using the file upload feature in the CVM instance console.

If the CVM client can be accessed via public IP address, you can also run the following command in the local command line mode to upload the file.

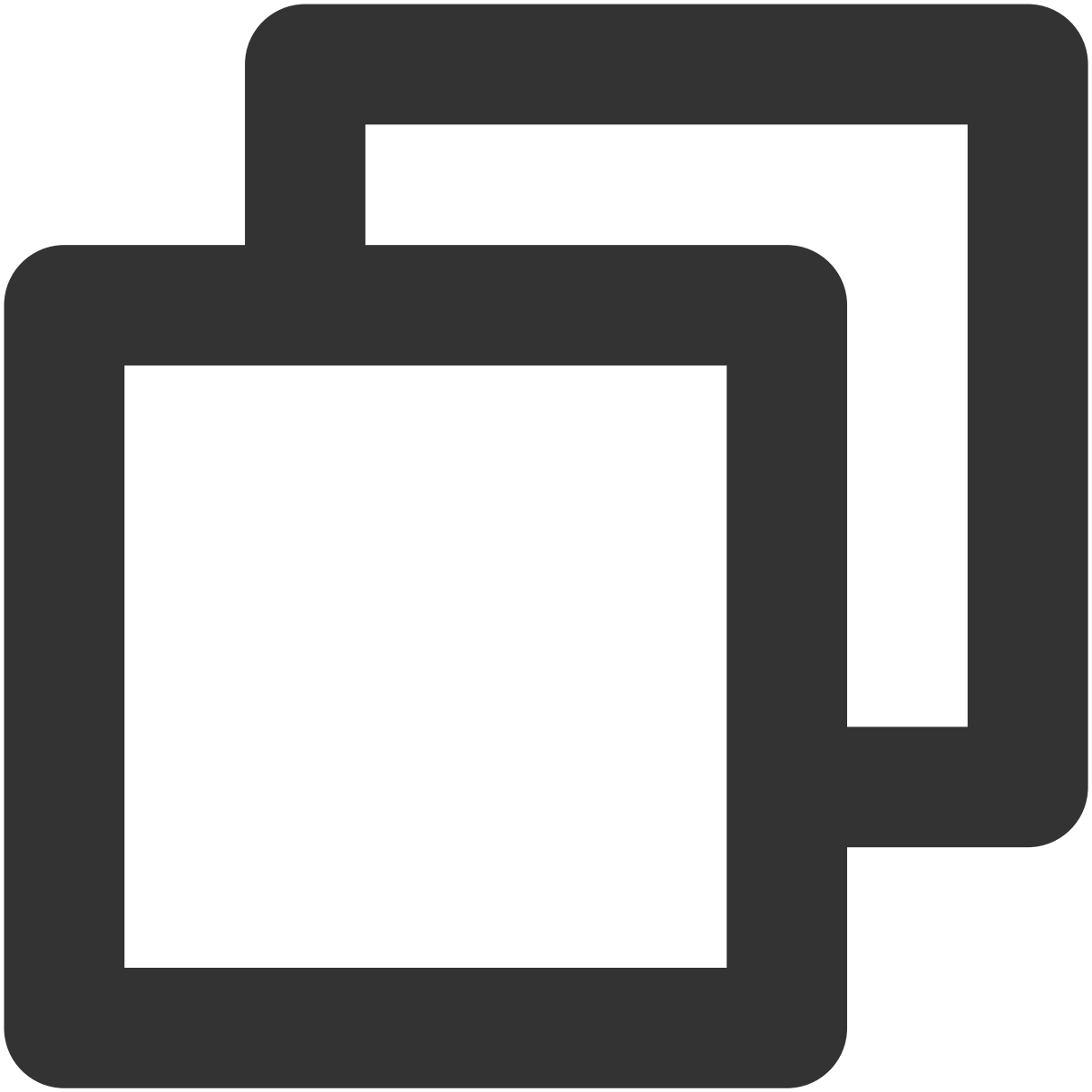


```
scp $localfile root@public IP address:$remotefolder
```

Here, $localfile is the path and the name of your local file; root is the username of the CVM server. You can look up the public IP address in the CVM client console. $remotefolder is the path where you want to store the file in the CVM server. After completing the upload, you can check whether the file is in the corresponding folder on the CVM client command line.

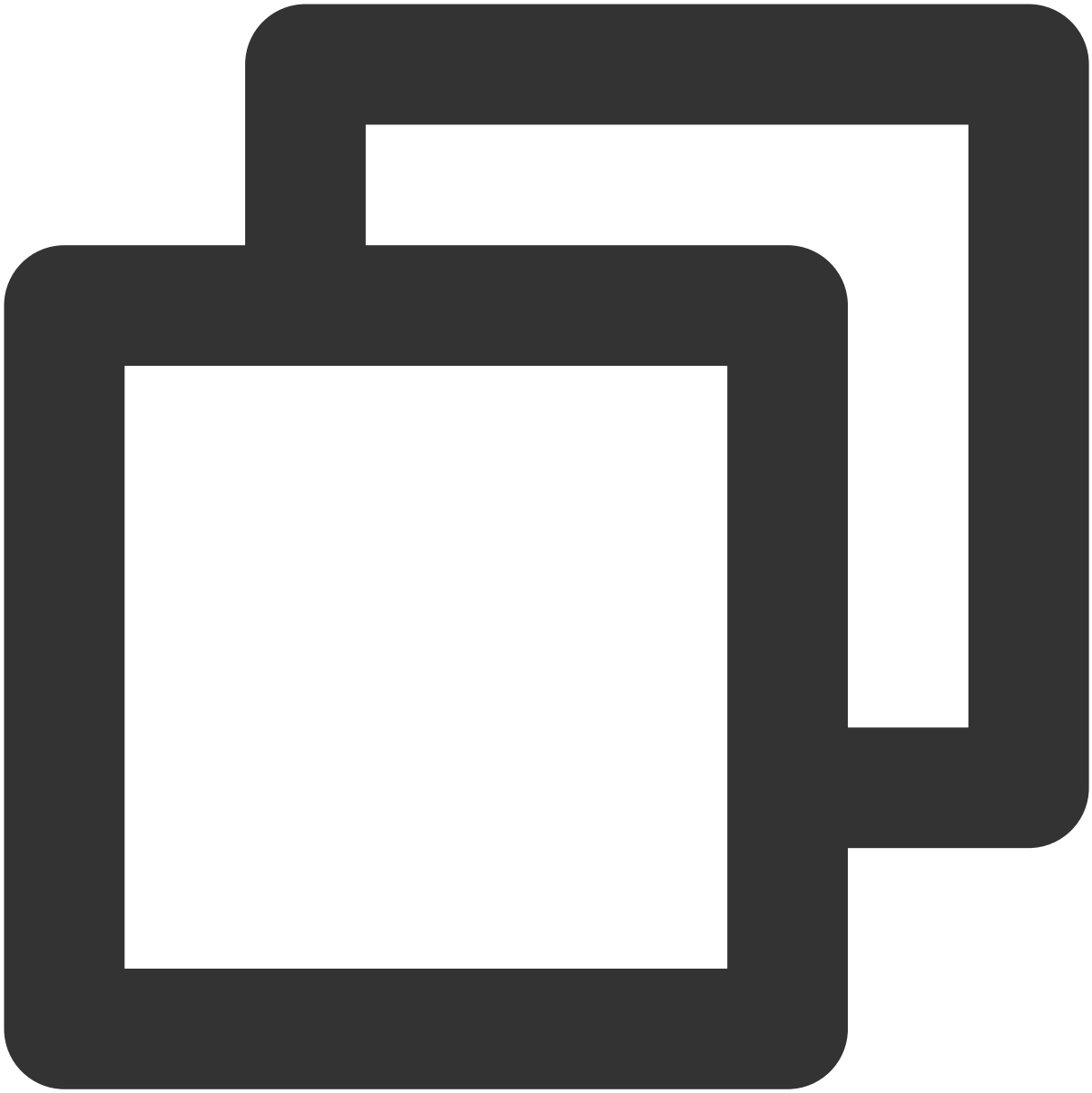
Executing the Sample and Checking the Results

Log in to the CVM client, switch to the corresponding folder, and use the following command to execute the sample.



```
java -jar $package.jar
```

After executing the code, if Done is output in the console, it indicates that all operations are completed. You can switch to the HBase shell and use the list command to check if the HBase table created using the API is successful. If successful, you can use the scan command to view the specific contents of the table.



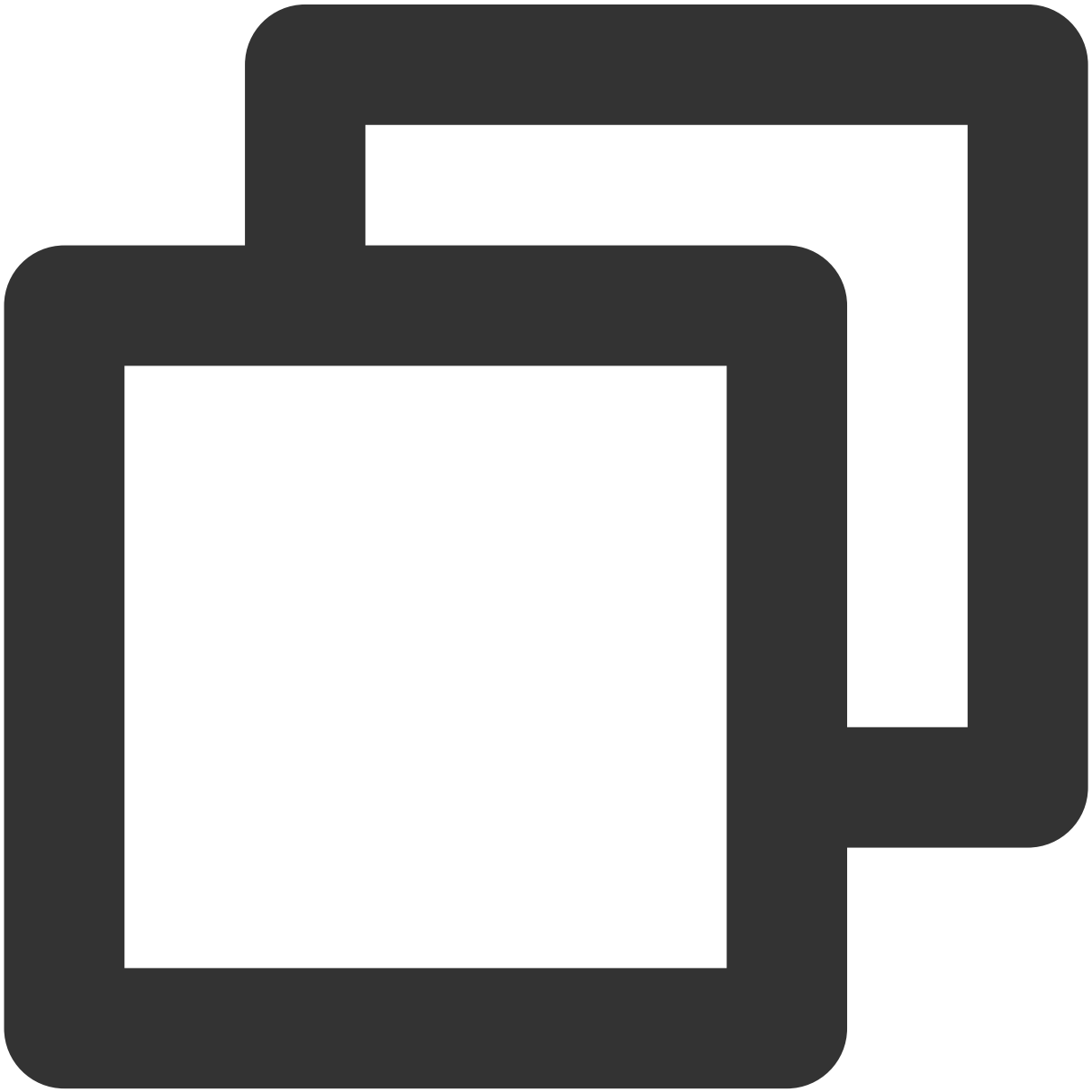
```
hbase:001:0> list 'test1'
TABLE
test1
1 row(s)
Took 0.4315 seconds
=> ["test1"]
hbase:002:0> scan 'test1'
ROW                                                    COLUMN+CELL
 row1                                                  column=cf:a, timestamp=2024-07-16T16
 row2                                                  column=cf:b, timestamp=2024-07-16T16
 row3                                                  column=cf:c, timestamp=2024-07-16T16
```

```
3 row(s)
Took 0.1231 seconds
hbase:003:0> exit
```

Using the Golang API

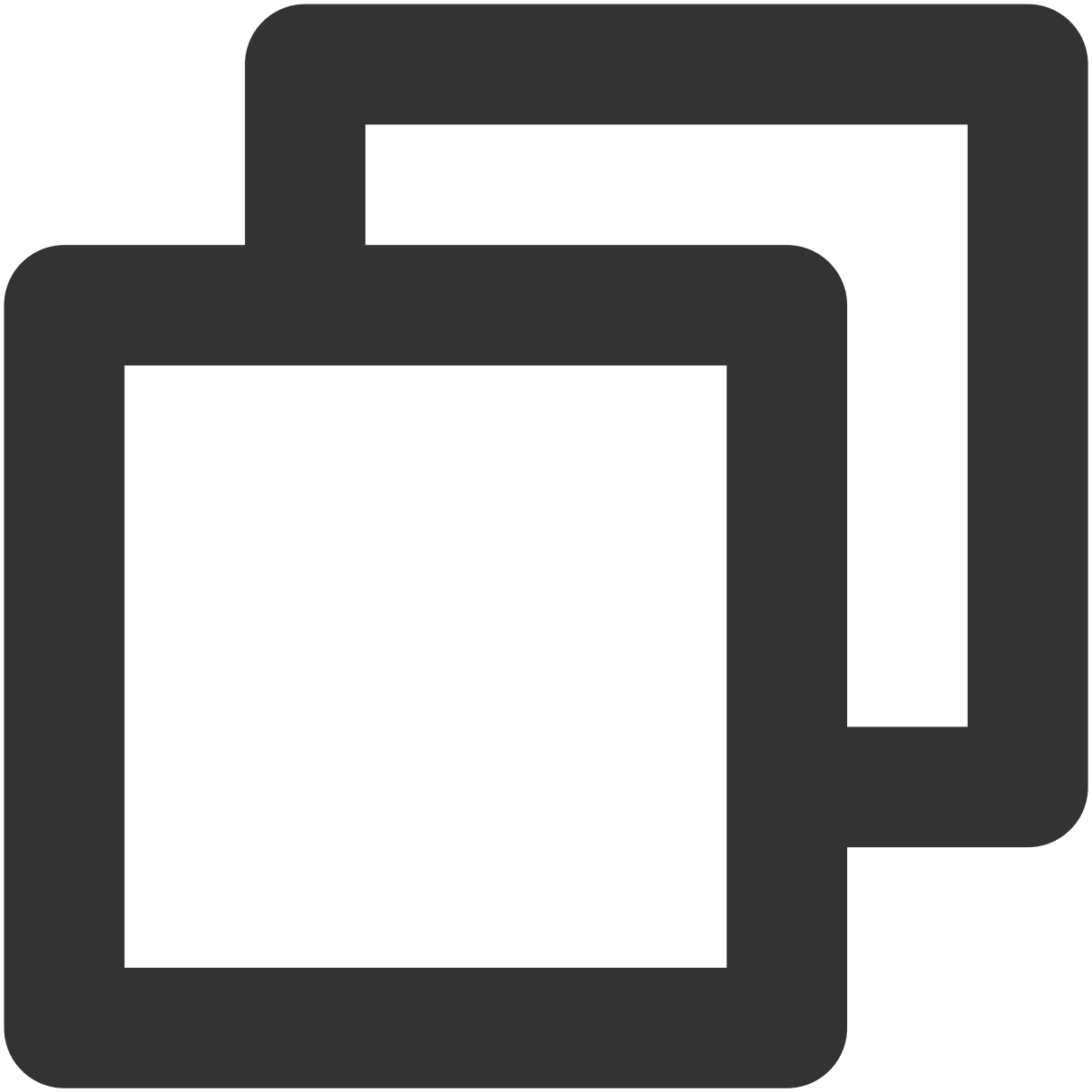
Adding Environment and Sample Code

On the CVM client, install Golang and configure the environment variables. Take version go1.17.13 as an example.



```
cd /usr/local
wget https://go.dev/dl/go1.17.13.linux-amd64.tar.gz
tar -zxvf go1.17.13.linux-amd64.tar.gz
vi /etc/profile

Append to /etc/profile
export GO_HOME=/usr/local/go
export PATH=${GO_HOME}/bin:$PATH
Save and return to the command line.

source /etc/profile
```

Ensure that the Golang version is consistent in the development environment. Enter the project directory, and if there is no go.mod file in the project directory, execute the following command.



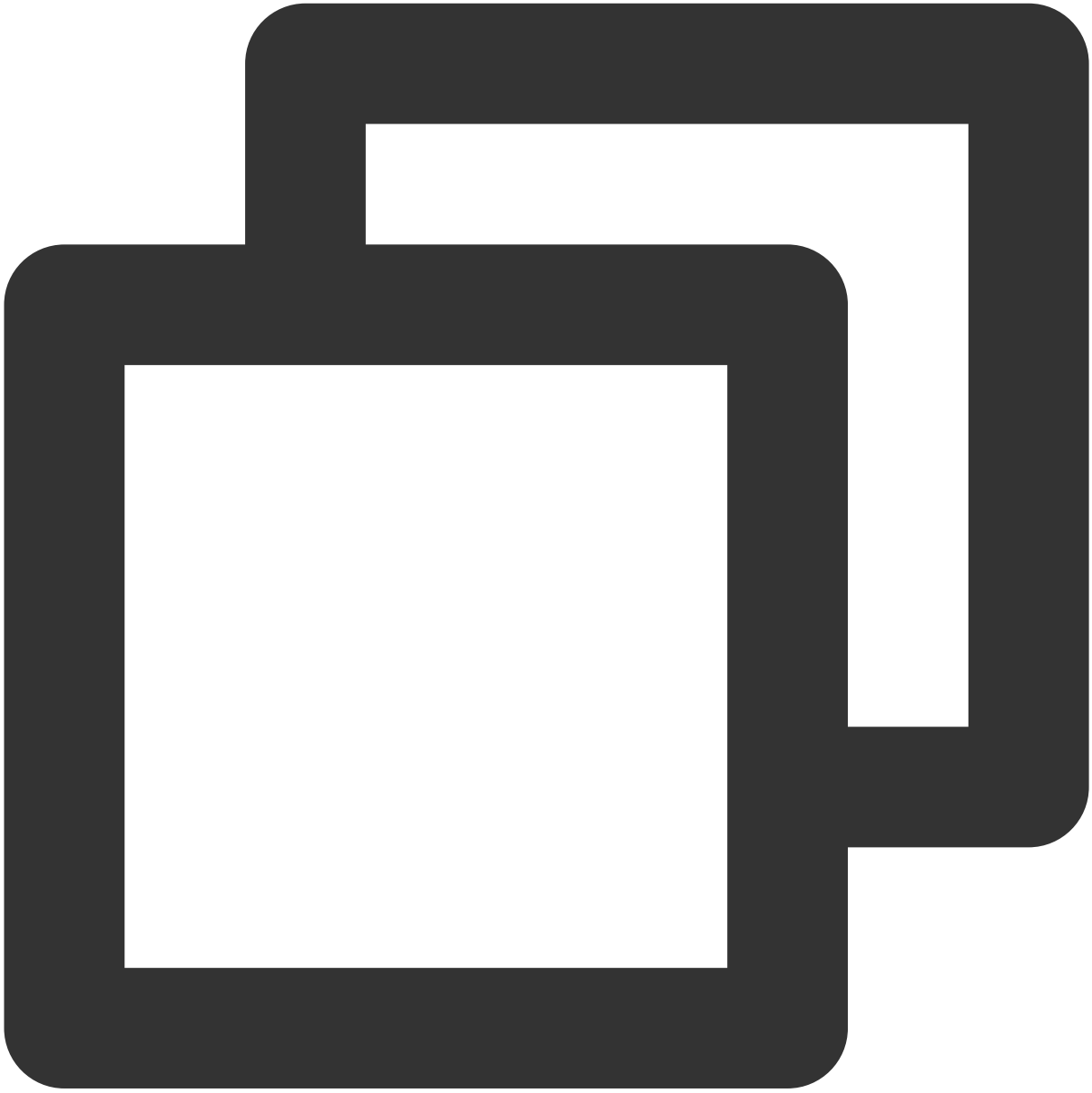
```
go mod init test-litehbase
```

Download the appropriate gohbase dependency branch for the Golang version by (viewing the gohbase project address).

Golang Version	commit_id
1.22.x	731f0bdb6be56dfe0b5a5a79582161bc7fbed32b
1.18.x~1.21.x	1fee39f343954ca7501a6b8f25abd9f86eaf618b

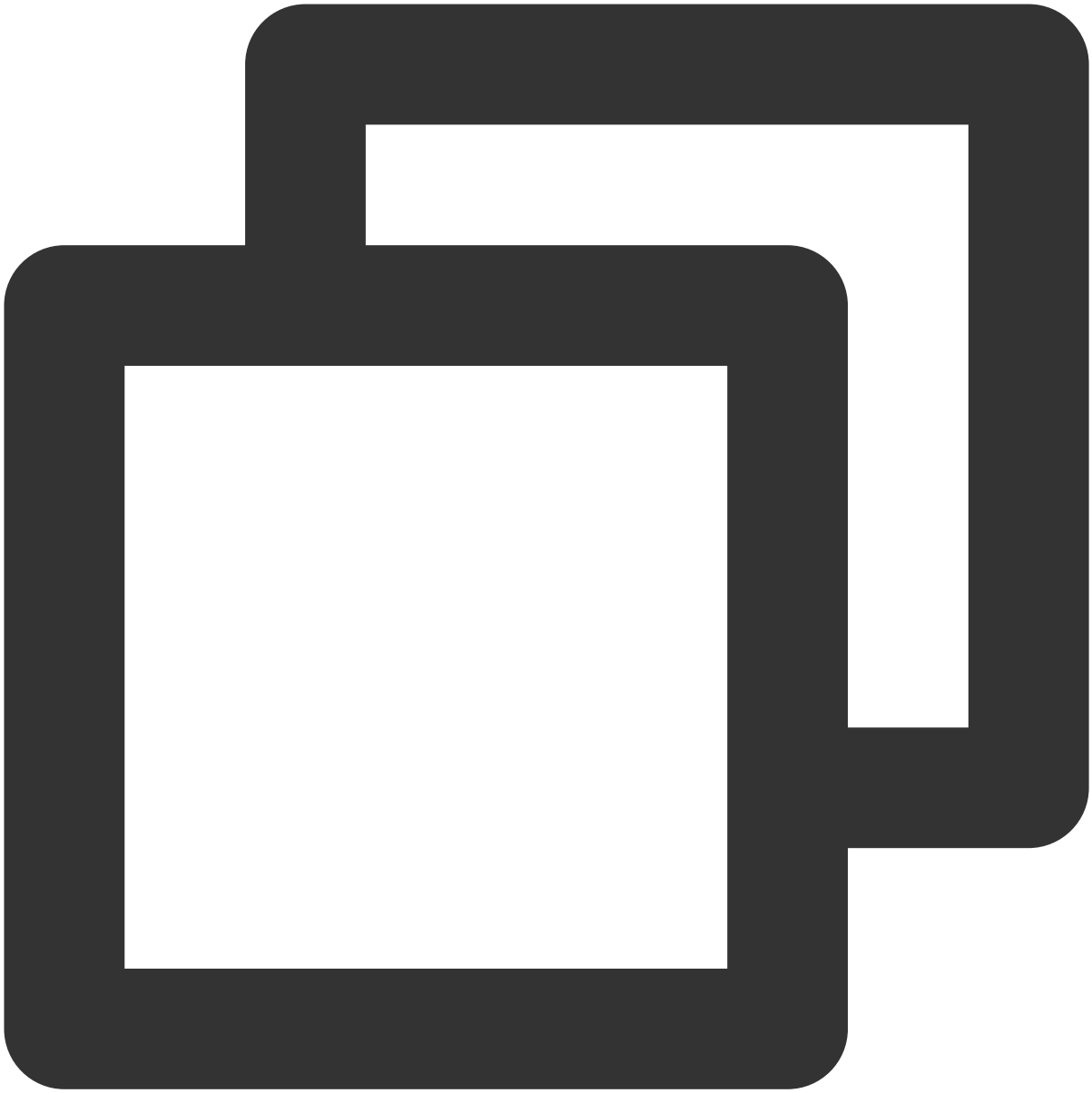
1.13.x~1.17.x	05795eede1cb2442e6cc0313db93a4b544b49148

You can view more Golang version-specific commits in Commits · tsuna/gohbase · GitHub.



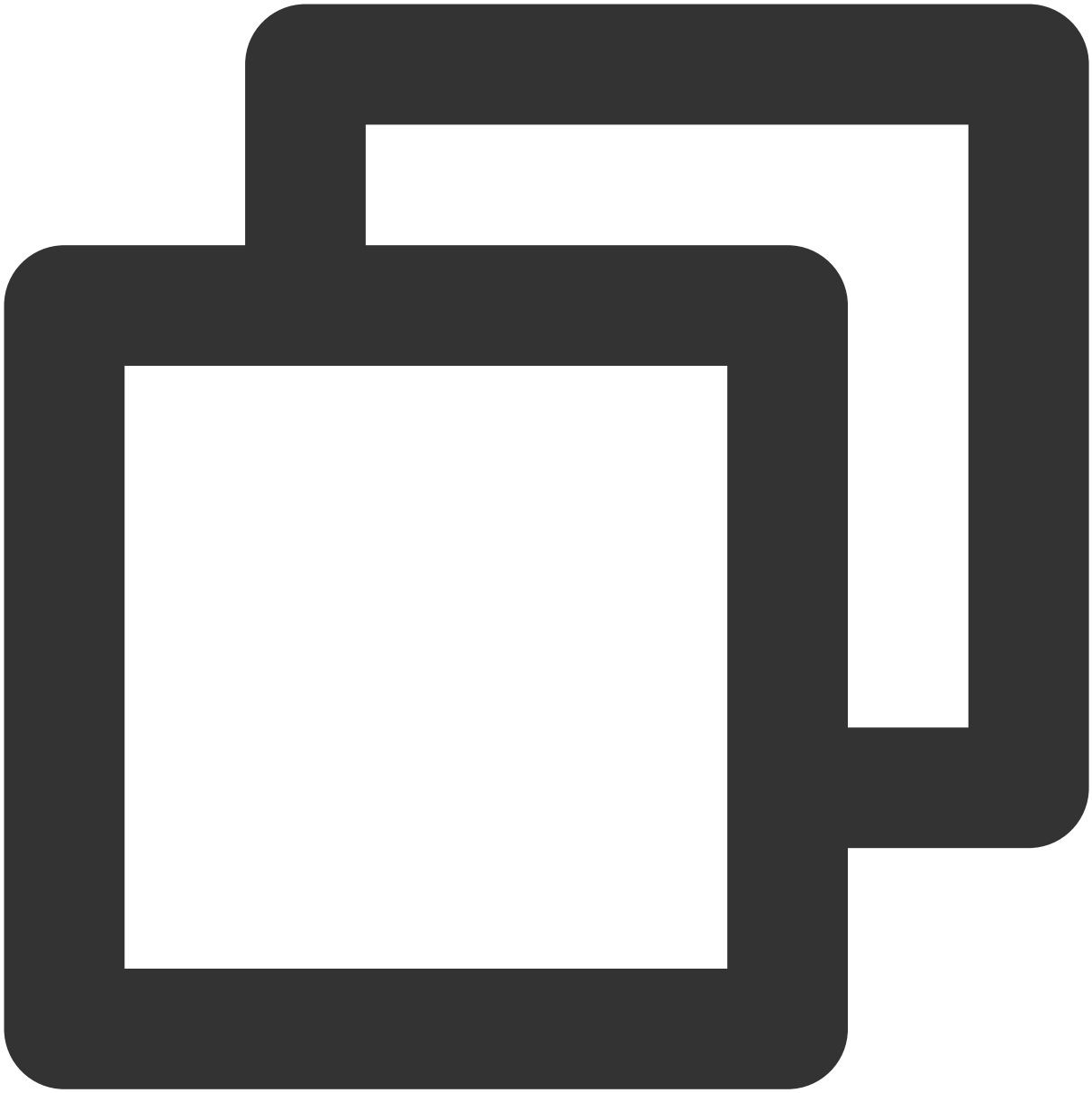
```
go env -w GOPROXY=https://goproxy.cn
go get github.com/tsuna/gohbase@commit_id
# Taking Version 1.17.13 as an Example: go get github.com/tsuna/gohbase@05795eede1c
```

Create a sample code file demo.go in the project directory. The project directory structure is as follows.



```
test-litehbase/
├── demo.go
├── go.mod
└── go.sum
```

The sample code for demo.go is as follows.

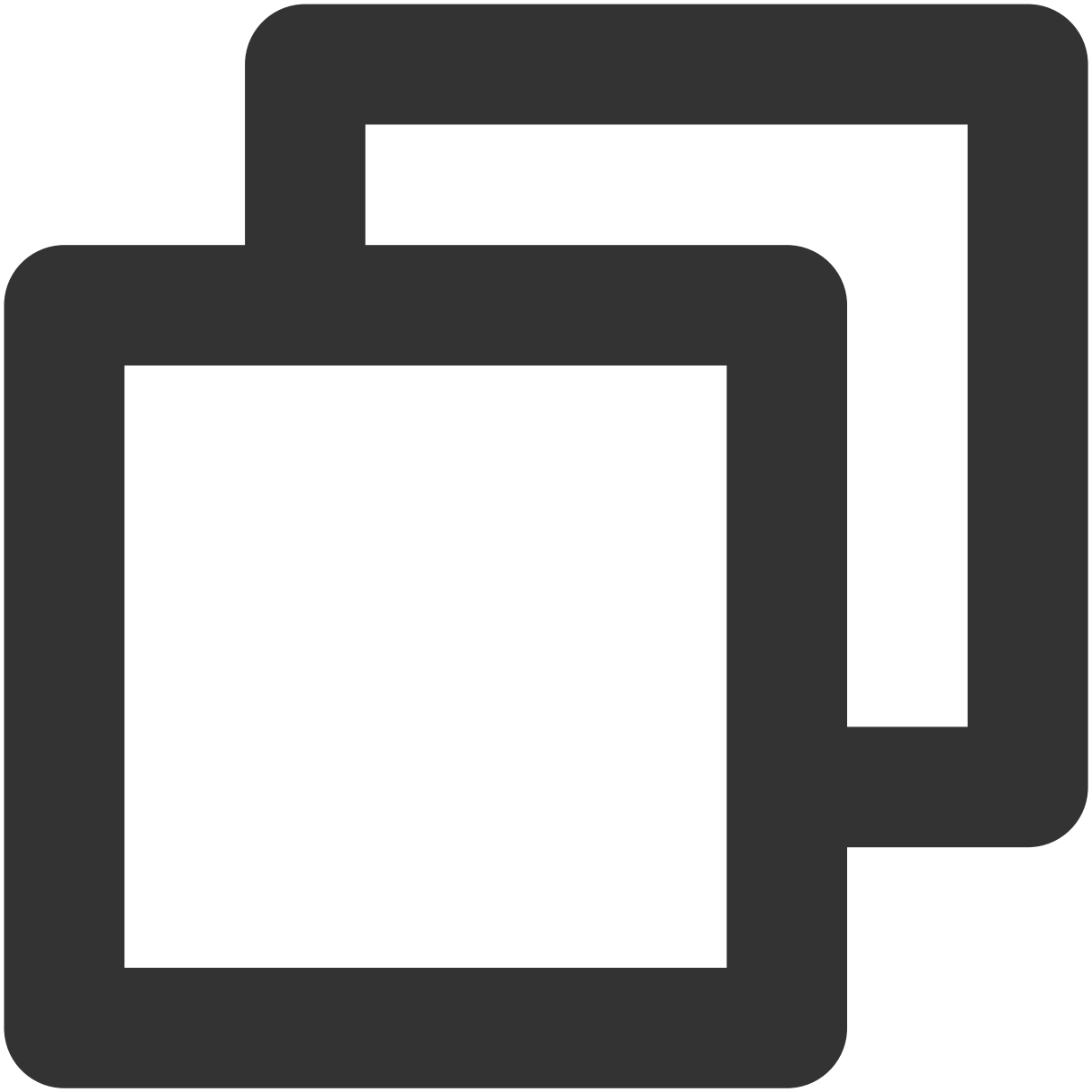


```
package main

import (
          "context"
          "fmt"
          "github.com/tsuna/gohbase"
          "github.com/tsuna/gohbase/hrpc"
          "log"
)

func main() {
```

```
        // Obtain $quorum from the Zookeeper access address found in the access me
        // Example: quorum := "10.0.0.8,10.0.0.11,10.0.0.5"
        quorum := $quorum
        // Creating Table.
        tableName := "test2"
        var cfs = map[string]map[string]string{
                "cf1": {
                        "MIN_VERSIONS": "1",
                },
        }
        admin := gohbase.NewAdminClient(quorum)
        crt := hrpc.NewCreateTable(context.Background(), []byte(tableName), cfs)
        err := admin.CreateTable(crt)
        if err != nil {
                log.Fatal(err)
        }

        client := gohbase.NewClient(quorum)

        // Write data.
        putRequest, err := hrpc.NewPutStr(context.Background(), tableName, "my_row
                "cf1": map[string][]byte{
                        "col1": []byte("value1"),
                        "col2": []byte("value2"),
                },
        })
        if err != nil {
                log.Fatal(err)
        }
```

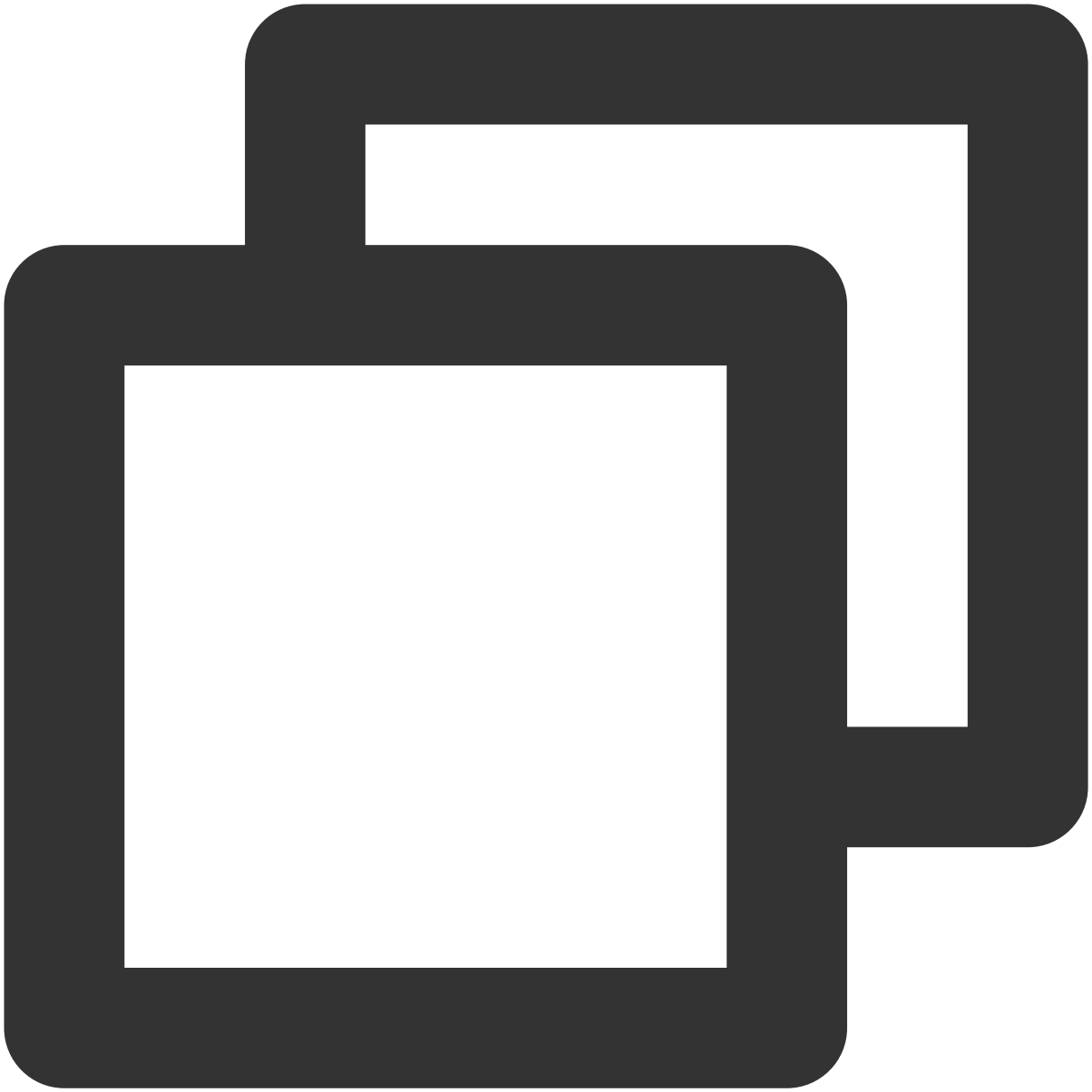
Compiling Code and Packaging It for Upload

If your current development environment is a Linux environment, execute the following command in the command line.



```
go build demo.go
```

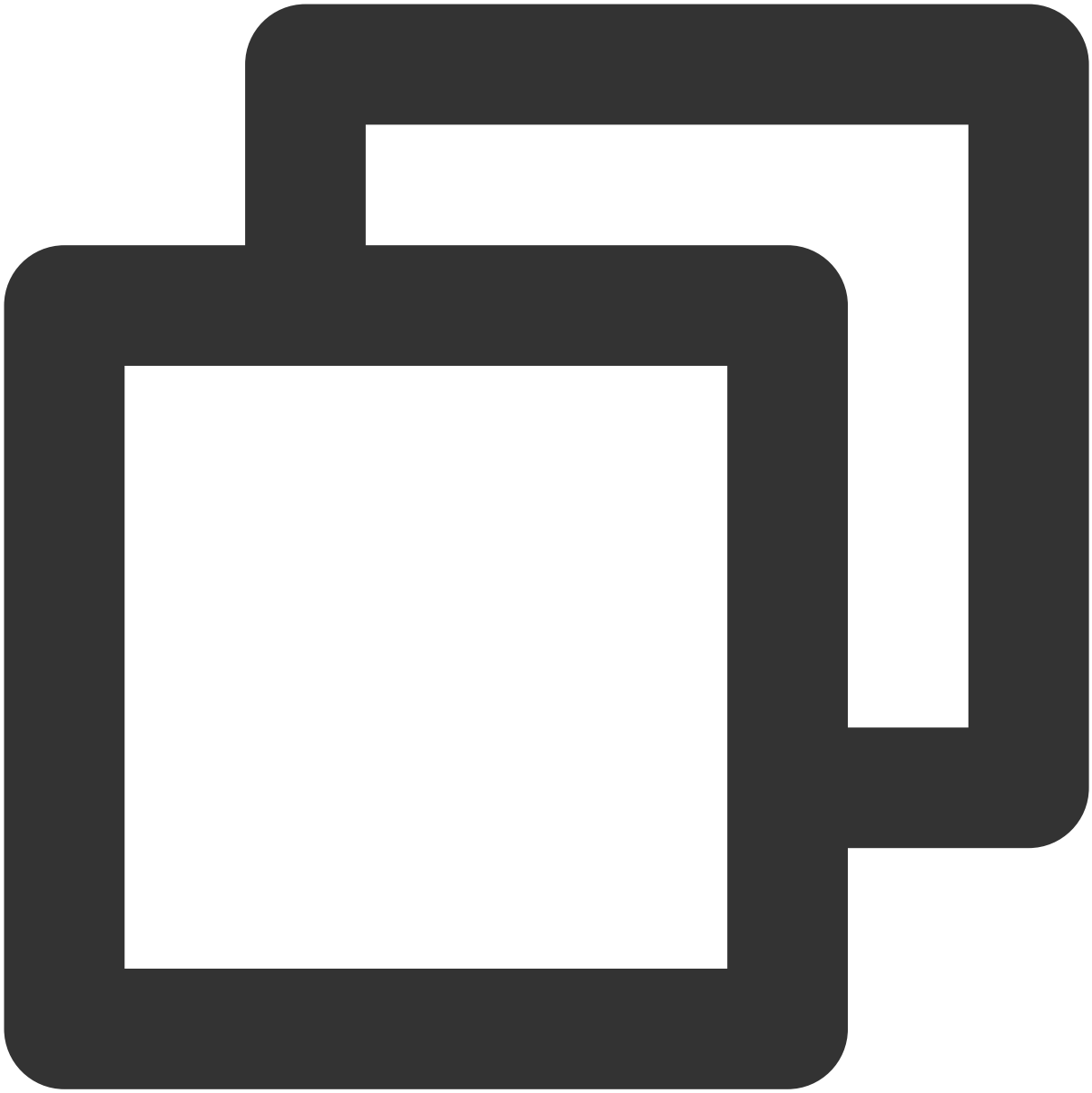
If it is a non-Linux environment, execute the following command, package it into a Linux file, and then switch back to the original environment to ensure other files compiled normally.



```
go env -w GOOS=linux
go build demo.go
```

In the project directory, you can see the compiled binary file named demo. Upload this file to the CVM client using the file upload feature in the CVM instance console.

If the CVM client can be accessed via public IP address, you can also run the following command in the local command line mode to upload the file.

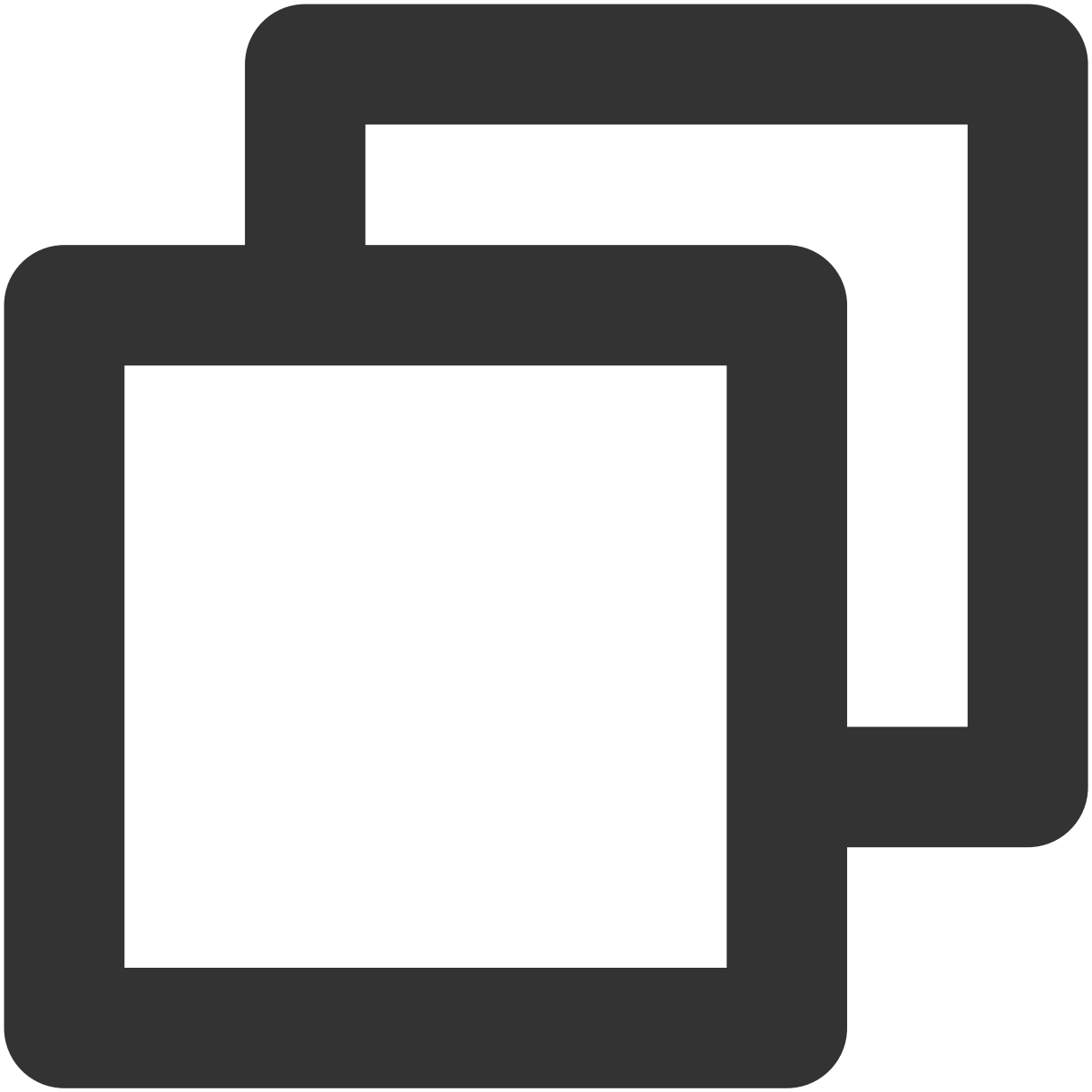


```
scp $localfile root@public IP address:$remotefolder
```

Here, $localfile is the path and the name of your local file; root is the username of the CVM server. You can look up the public IP address in the CVM client console. $remotefolder is the path where you want to store the file in the CVM server. After completing the upload, you can check whether the file is in the corresponding folder on the CVM client command line.

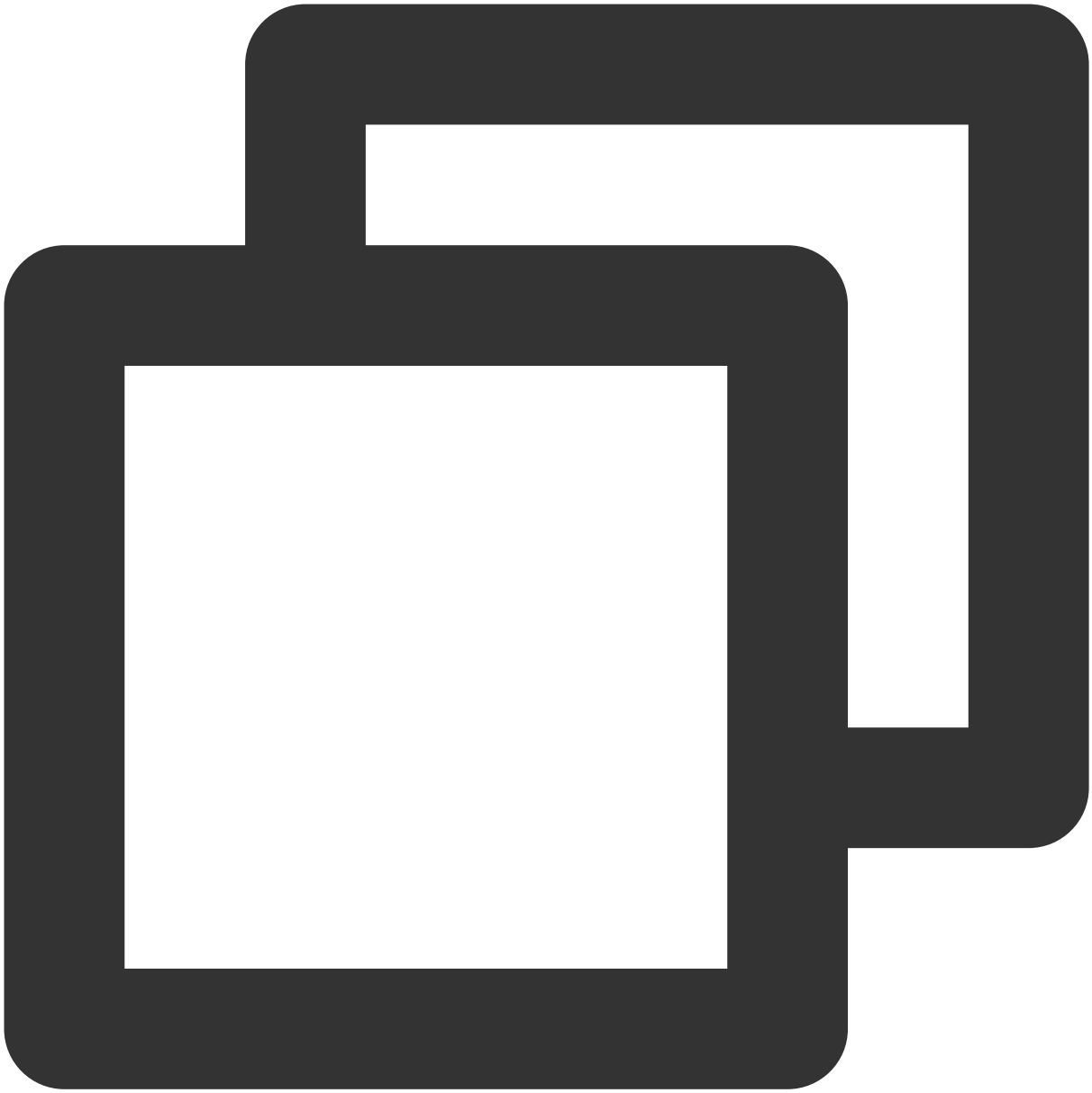
Executing the Sample and Checking the Results

Switch to the target directory where the file is located, grant the permission to execute to this binary file, and execute it.



```
chmod +x demo
./demo
```

After running the code, switch to the HBase shell and check whether the HBase table created by using the API is successful via the list command. If successful, you can use the scan command to view the specific contents of the table.



```
hbase:001:0> list 'test2'
TABLE
test2
1 row(s)
Took 0.3814 seconds
=> ["test2"]
hbase:002:0> scan 'test2'
ROW                                COLUMN+CELL
 my_row_key                        column=cf1:col1, timestamp=2024-07-17T21:16:49.30
 my_row_key                        column=cf1:col2, timestamp=2024-07-17T21:16:49.30
1 row(s)
```

```
Took 0.1268 seconds
hbase:003:0> exit
```