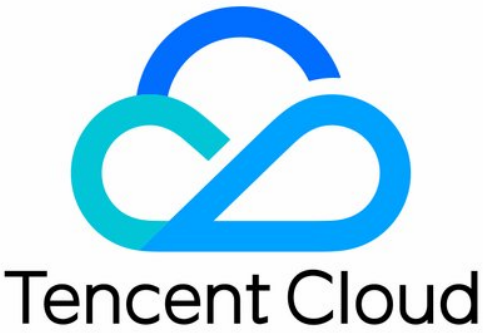
CODING Continuous Integration

Best Practices

Product Documentation



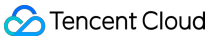
©2013-2022 Tencent Cloud. All rights reserved.

Copyright Notice

©2013-2023 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

Best Practices
- Automated Deployment
  - K8s Clusters
  - Docker Servers
  - COS Buckets
- Use Docker in Continuous Integration
- Use SSH in Continuous Integration

©2013-2022 Tencent Cloud. All rights reserved.

Best Practices
Automated Deployment
K8s Clusters

Last updated：2023-12-29 11:44:50

This document describes how to use Continuous Integration to release a project to a K8s cluster.

Prerequisites

Before configuring the CODING Continuous Integration (CODING-CI) build environment, you must activate the CODING DevOps service for your Tencent Cloud account.

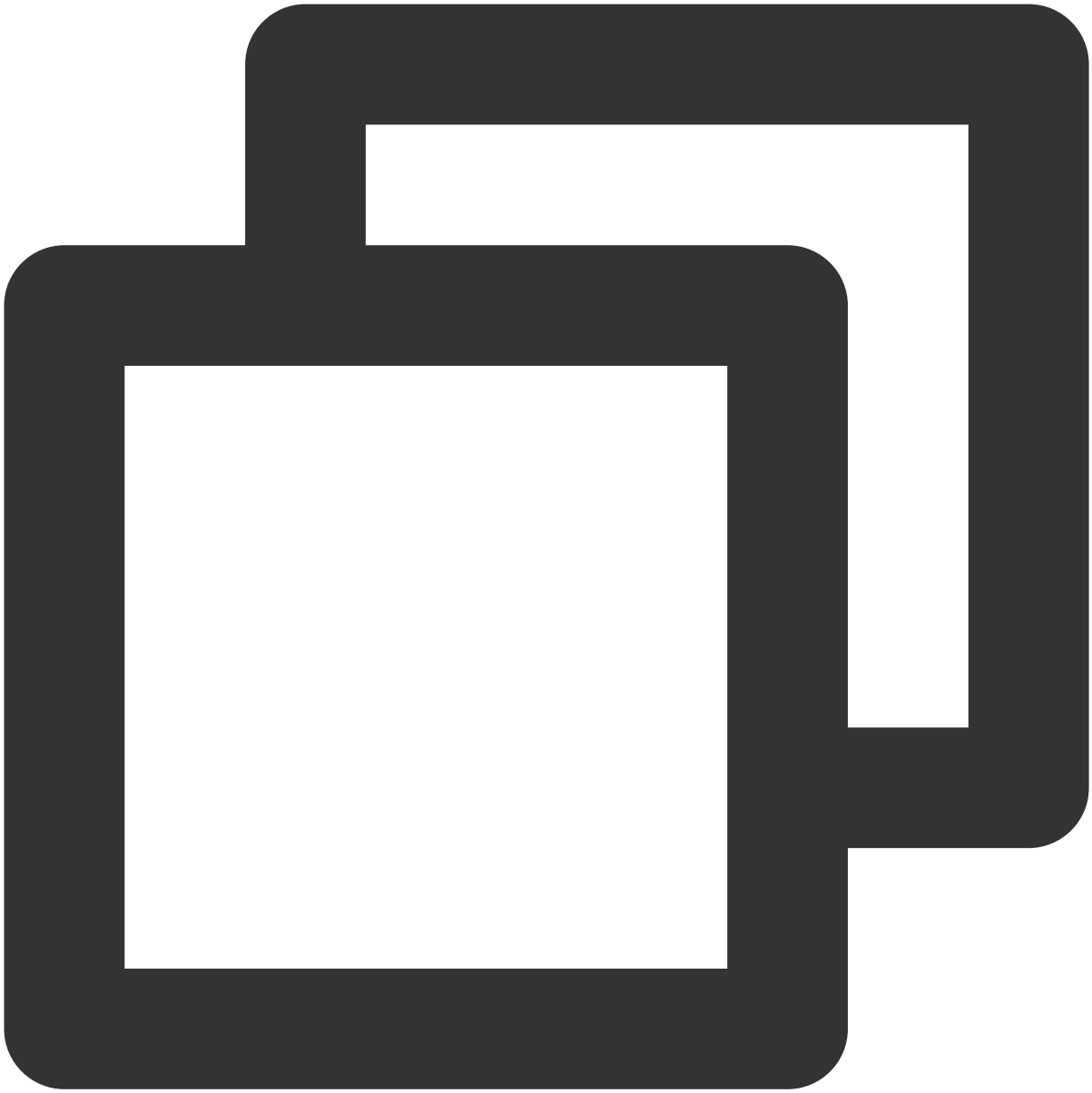
Open Project

1. Log in to the CODING Console and click the **team domain name** to go to CODING.
2. Click

in the upper-right corner to open the project list page and click a **project icon** to open the project.
3. Select **Continuous Integration** from the menu on the left.
Follow the steps below to easily deploy a project to a K8s cluster through Continuous Integration:
1. Retrieve the username and password for the Docker repository (You can obtain them by creating an access token with one click in CODING Artifact Repository). Then, enter them in the environment variables of Continuous Integration.
2. Build a Docker image and upload it to the repository.
3. Using a cloud computing service provider (such as Tencent Cloud), create a K8s cluster and deployment. Retrieve kubeconfig and enter it in CODING Credential Management.
4. Use the following Jenkinsfile in Continuous Integration: Run kubectl and deploy.

Jenkinsfile

©2013-2022 Tencent Cloud. All rights reserved.



```
pipeline {
  agent any
  stages {
    stage('Check out') {
      steps {
        checkout([$class: 'GitSCM', branches: [[name: env.GIT_BUILD_REF]],
        userRemoteConfigs: [[url: env.GIT_REPO_URL, credentialsId: env.CREDENTIALS_
      }
    }
    stage('Build') {
      steps {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        echo 'building...'
        script {
          // Modify dockerServer, dockerPath, and imageName
          dockerServer = 'codes-farm-docker.pkg.coding.net'
          dockerPath = '/laravel-demo/laravel-docker'
          imageName = "${dockerServer}${dockerPath}/laravel-demo:1.0.0"
          def customImage = docker.build(imageName)

          // Push Docker image to repository
          docker.withRegistry("https://${dockerServer}", CODING_ARTIFACTS_CREDENTIA
            customImage.push()
          }
        }
      }
      stage('Deploy to K8s') {
        steps {
          echo 'deploying...'
          script {
            // Modify credentialsId: Fill in the K8s credential ID
            withKubeConfig([credentialsId: 'f23cc59c-dfd1-40b9-a12f-2c9b6909e908']) {
              // Use kubectl to create a K8s secret key: originates from the environm
              sh(script: "kubectl create secret docker-registry coding --docker-serve

              // Use kubectl to modify the K8s deployment: Specify the Docker image l
              // Modify laravel-demo and web to the values in your deployment
              sh "kubectl patch deployment laravel-demo --patch '{\\"spec\\": {\\"tem
            }
          }
        }
      }
    }
  }
```

Notice

As a **K8s deployment** includes at least five steps, we recommend you to use Continuous Deployment instead of writing all of them in **Continuous Integration** to facilitate future maintenance.

©2013-2022 Tencent Cloud. All rights reserved.

Docker Servers

Last updated：2023-12-29 11:44:51

This document describes how to use Continuous Integration to release a project to a Docker server.

Prerequisites

Before configuring the CODING Continuous Integration (CODING-CI) build environment, you must activate the CODING DevOps service for your Tencent Cloud account.

Open Project

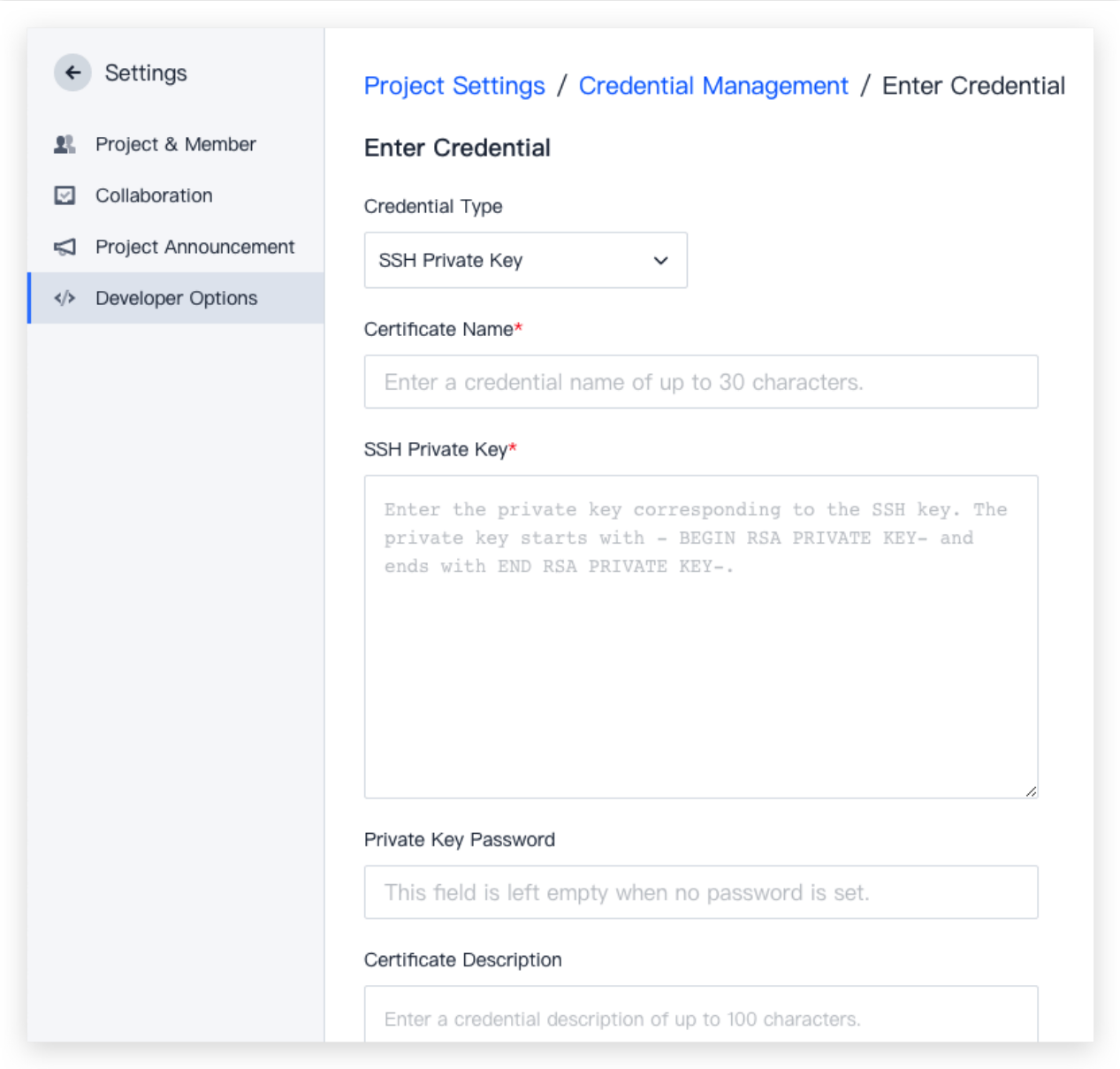
1. Log in to the CODING Console and click the **team domain name** to go to CODING.
2. Click

in the upper-right corner to open the project list page and click a **project icon** to open the project.
3. Select **Continuous Integration** from the menu on the left.

Retrieve SSH Key Pair

Log in to the server console and create an SSH key pair. After you obtain the private key pair, enter it in CODING's project token. Then, copy the public key content in id_rsa.pub to `~/.ssh/authorized_keys` in the server.

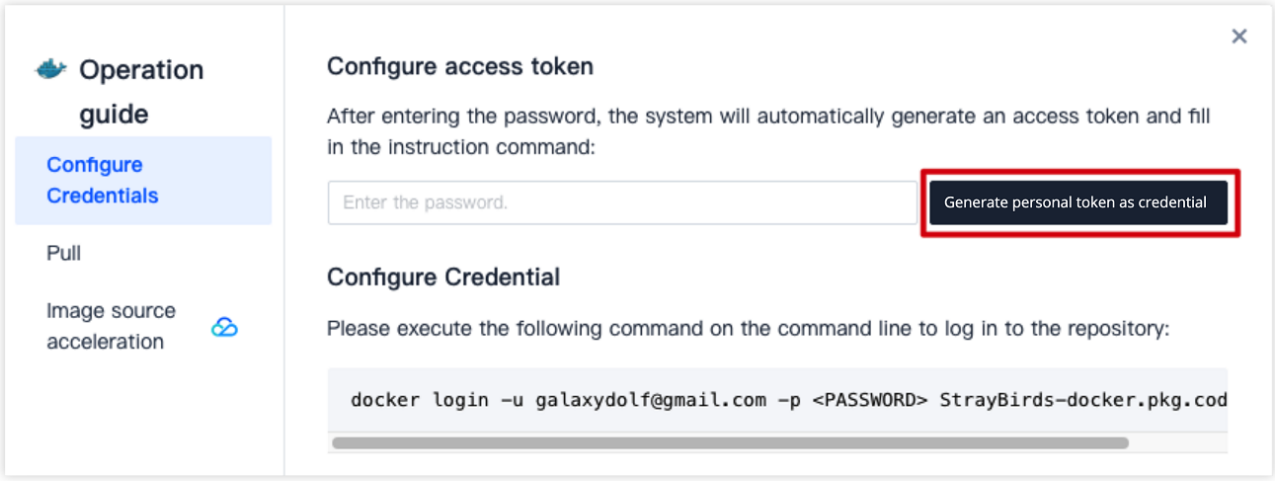
©2013-2022 Tencent Cloud. All rights reserved.



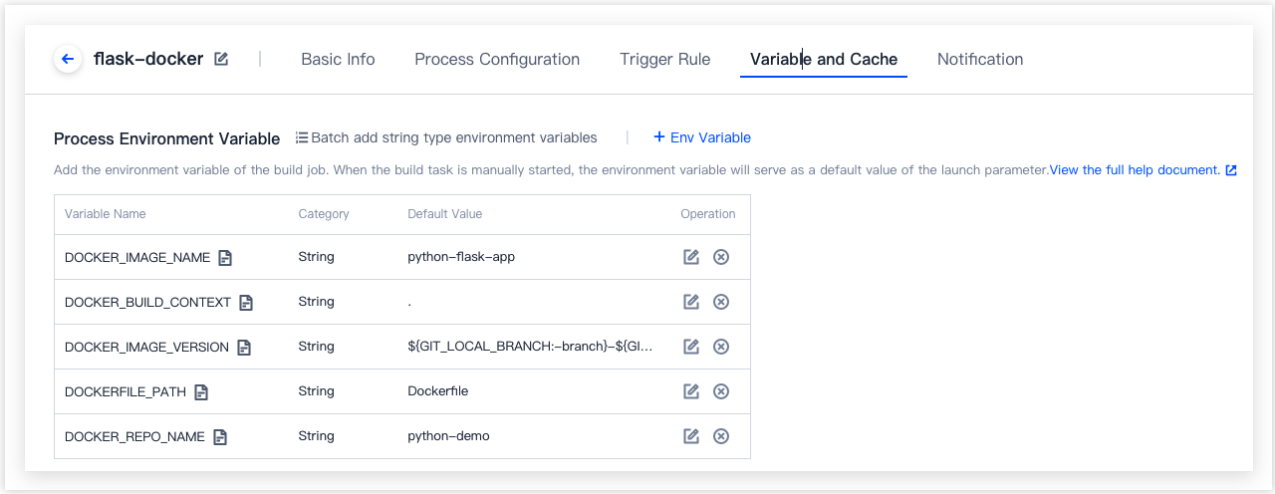
Retrieve Artifact Repository Information

1. Follow the instructions to retrieve the username and password for the Docker repository with one click and enter them in the environment variables of Continuous Integration.

©2013-2022 Tencent Cloud. All rights reserved.



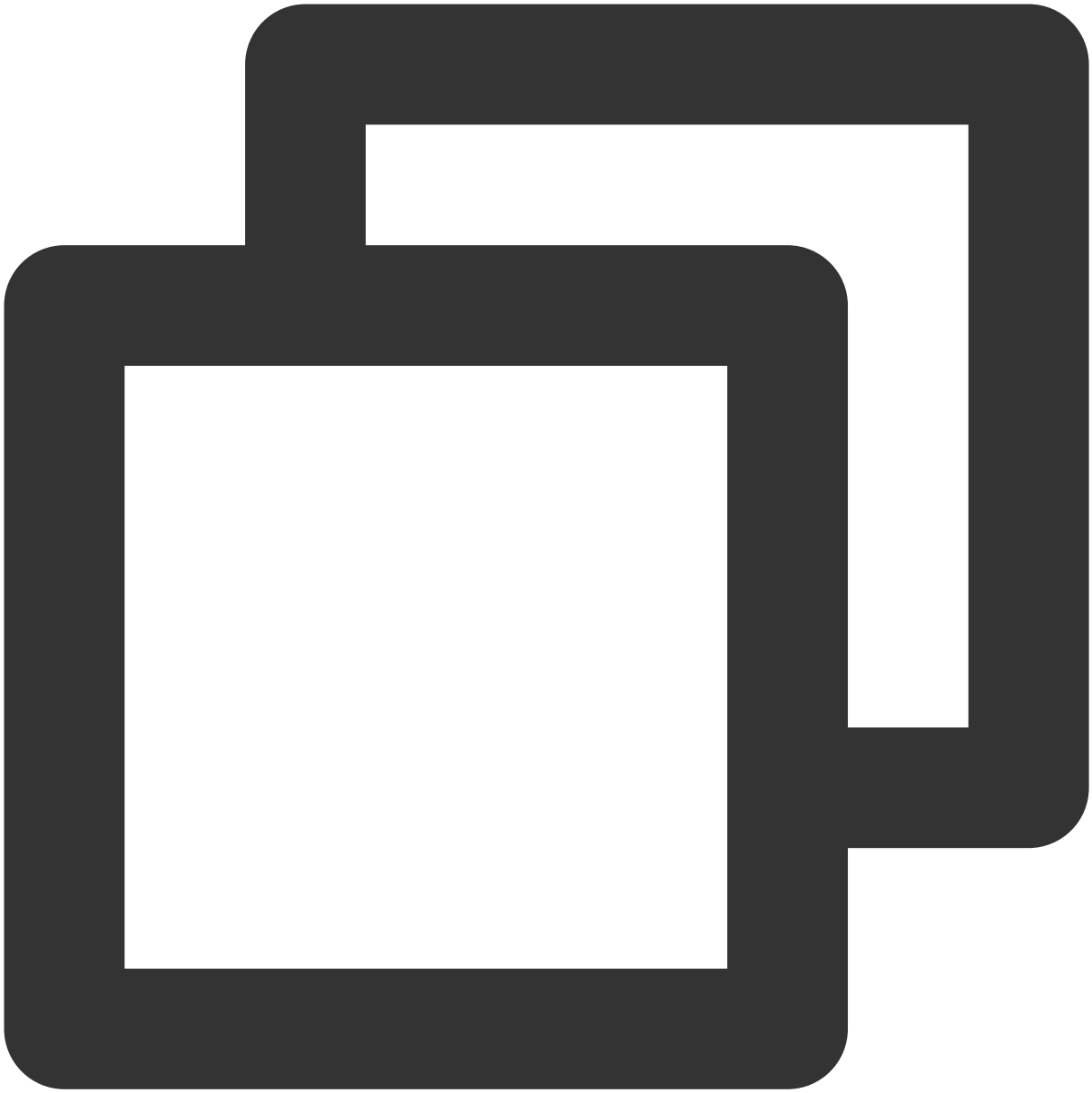
2. Enter them in Variables and Caches in the build plan details.



Jenkinsfile

Go to "Build Plan Settings" > "Process Configuration". Refer to the following configuration for filling.

©2013-2022 Tencent Cloud. All rights reserved.



```
pipeline {
  agent any
  stages {
    stage('Check out') {
      steps {
        checkout([$class: 'GitSCM', branches: [[name: env.GIT_BUILD_REF]],
        userRemoteConfigs: [[url: env.GIT_REPO_URL, credentialsId: env.CREDENTIALS_
      }
    }
    stage('Build') {
      steps {
```

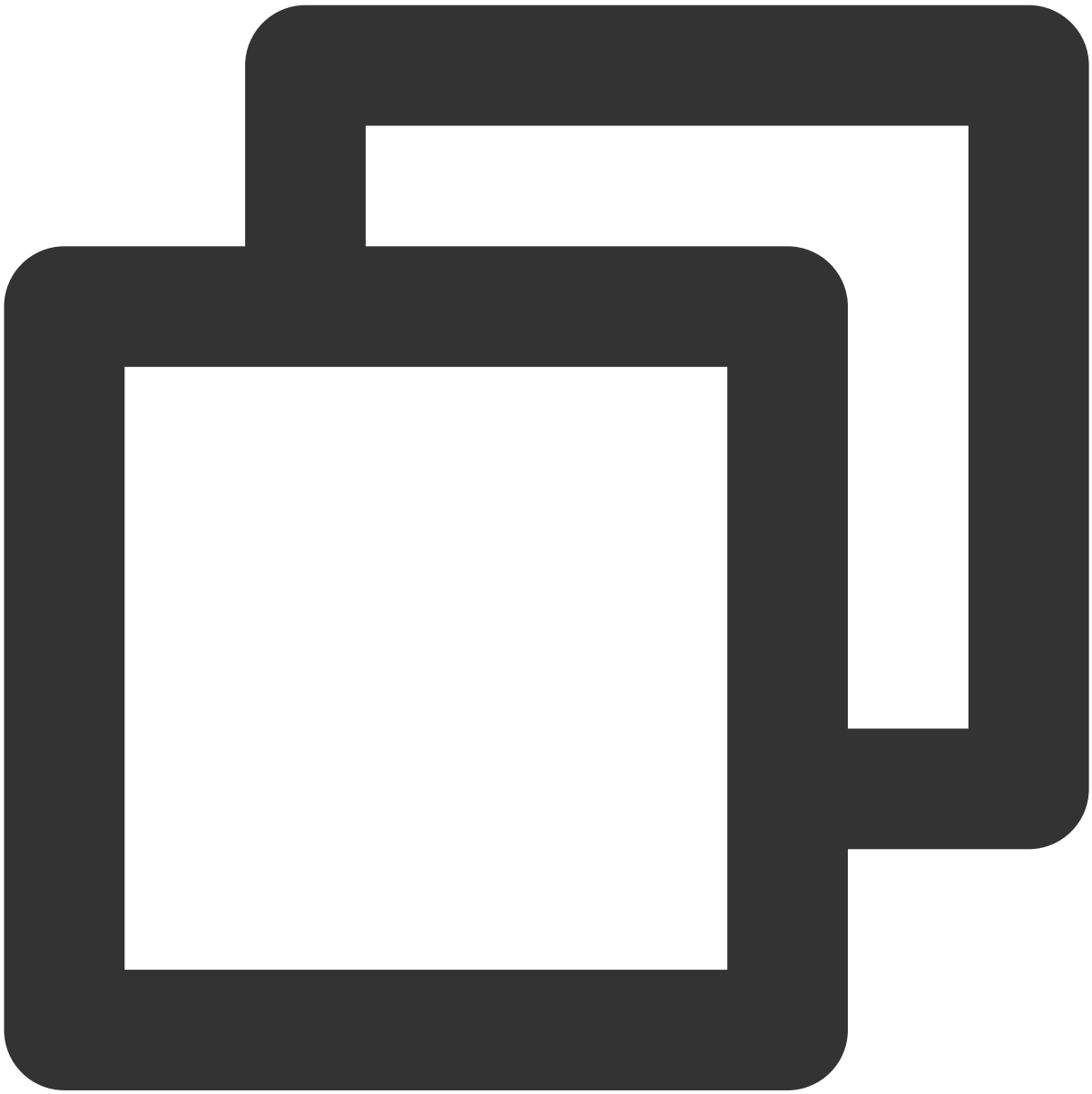
©2013-2022 Tencent Cloud. All rights reserved.

```
        echo 'building...'
        script {
          // Modify dockerServer, dockerPath, and imageName
          dockerServer = 'codes-farm-docker.pkg.coding.net'
          dockerPath = '/laravel-demo/laravel-docker'
          imageName = "${dockerServer}${dockerPath}/laravel-demo:1.0.0"
          def customImage = docker.build(imageName)

          // Push Docker image to repository
          docker.withRegistry("https://${dockerServer}", CODING_ARTIFACTS_CREDENTIA
            customImage.push()
          }
        }
      }
    }
    stage('Deploy') {
      steps {
        echo 'deploying...'
        script {
          // Declare server information
          def remote = [:]
          remote.name = 'web-server'
          remote.allowAnyHosts = true
          remote.host = '106.54.86.239'
          remote.port = 22
          remote.user = 'ubuntu'

          // In "CODING Credential Management" > "Credential ID", enter credentials
          withCredentials([sshUserPrivateKey(credentialsId: "c4af855d-402a-4f38-9c8
            remote.identityFile = id_rsa

            // Log in to the server via SSH and pull the Docker image
            // Configure DOCKER_USER and DOCKER_PASSWORD in the environment variabl
            sshCommand remote: remote, sudo: true, command: "apt-get install -y gnu
            sshCommand remote: remote, command: "docker login -u ${env.DOCKER_USER}
            sshCommand remote: remote, command: "docker pull ${imageName}"
            sshCommand remote: remote, command: "docker stop web | true"
            sshCommand remote: remote, command: "docker rm web | true"
            sshCommand remote: remote, command: "docker run --name web -d ${imageNa
          }
        }
      }
    }
  }
}
```

©2013-2022 Tencent Cloud. All rights reserved.

Docker Compose

The code for Docker Compose is slightly different:

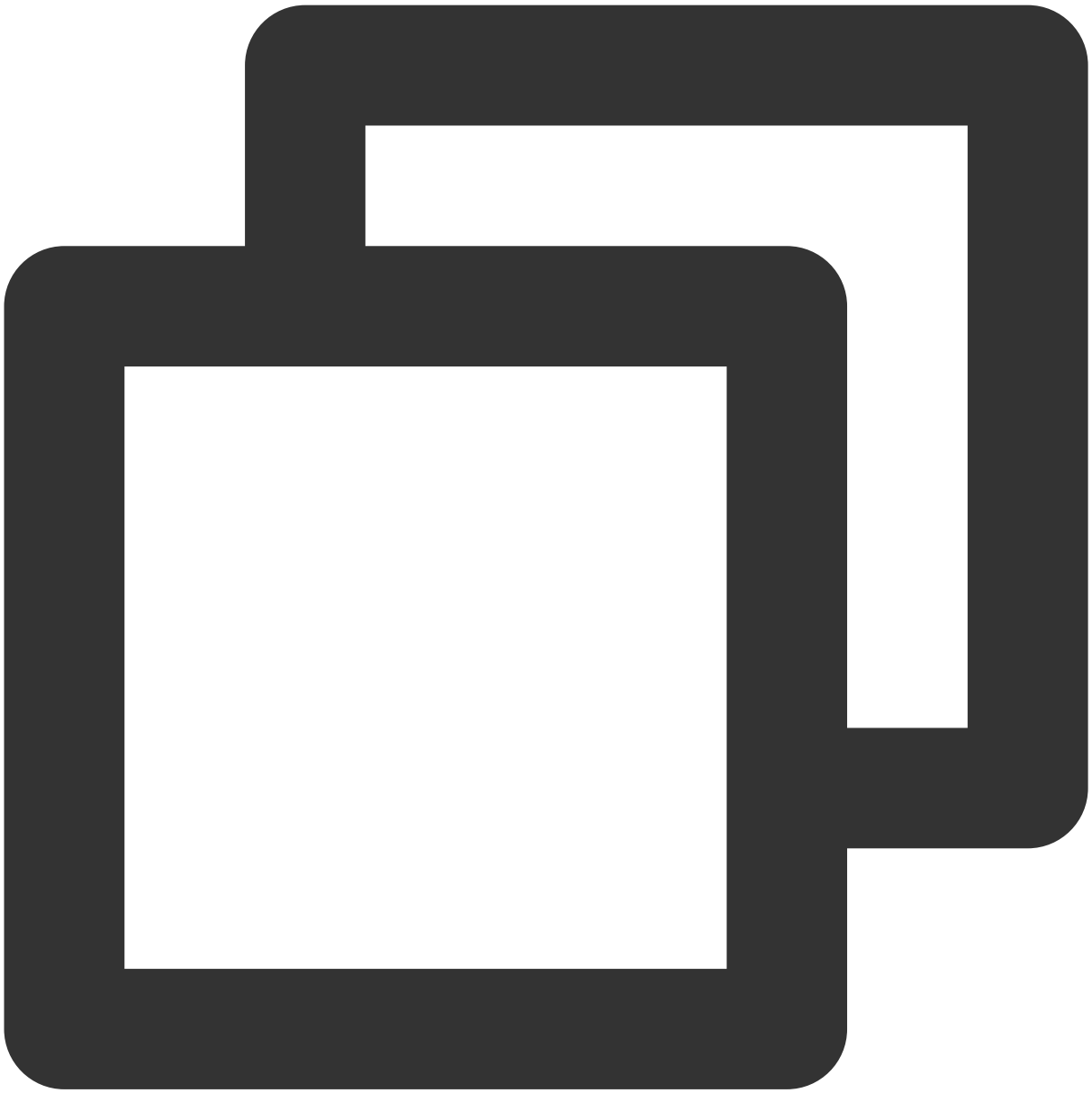


```
sshCommand remote: remote, sudo: true, command: "apt-get install -y gnupg2 pass"
sshCommand remote: remote, command: "docker login -u ${env.DOCKER_USER} -p ${env.DO
sshCommand remote: remote, sudo: true, command: "mkdir -p /var/www/site/"
sshCommand remote: remote, sudo: true, command: "chmod 777 /var/www/site/"
sshPut remote: remote, from: 'docker-compose.yml', into: '/var/www/site/'
sshCommand remote: remote, command: "cd /var/www/site/ && echo IMAGE=${imageName} >
sshCommand remote: remote, command: "cd /var/www/site/ && docker-compose down --rem
```

©2013-2022 Tencent Cloud. All rights reserved.

```
sshCommand remote: remote, command: "cd /var/www/site/ && docker-compose up -d --no
```

`docker-compose.yml` code:



```
version: '2.1'
services:
  web:
    env_file: .env
    build: .
    image: ${IMAGE:-laravel-demo:dev}
    ports:
```

©2013-2022 Tencent Cloud. All rights reserved.

```
      - "80:80"
    links:
    - redis
  redis:
    image: "redis:5"
```

©2013-2022 Tencent Cloud. All rights reserved.

COS Buckets

Last updated：2023-12-29 11:44:51

This document describes how to use Continuous Integration to release a project to a Cloud Object Storage (COS) bucket with one click.

Prerequisites

Before configuring the CODING Continuous Integration (CODING-CI) build environment, you must activate the CODING DevOps service for your Tencent Cloud account.

Open Project

1. Log in to the CODING Console and click the **team domain name** to go to CODING.
2. Click

in the upper-right corner to open the project list page and click a **project icon** to open the project.
3. Select **Continuous Integration** from the menu on the left.

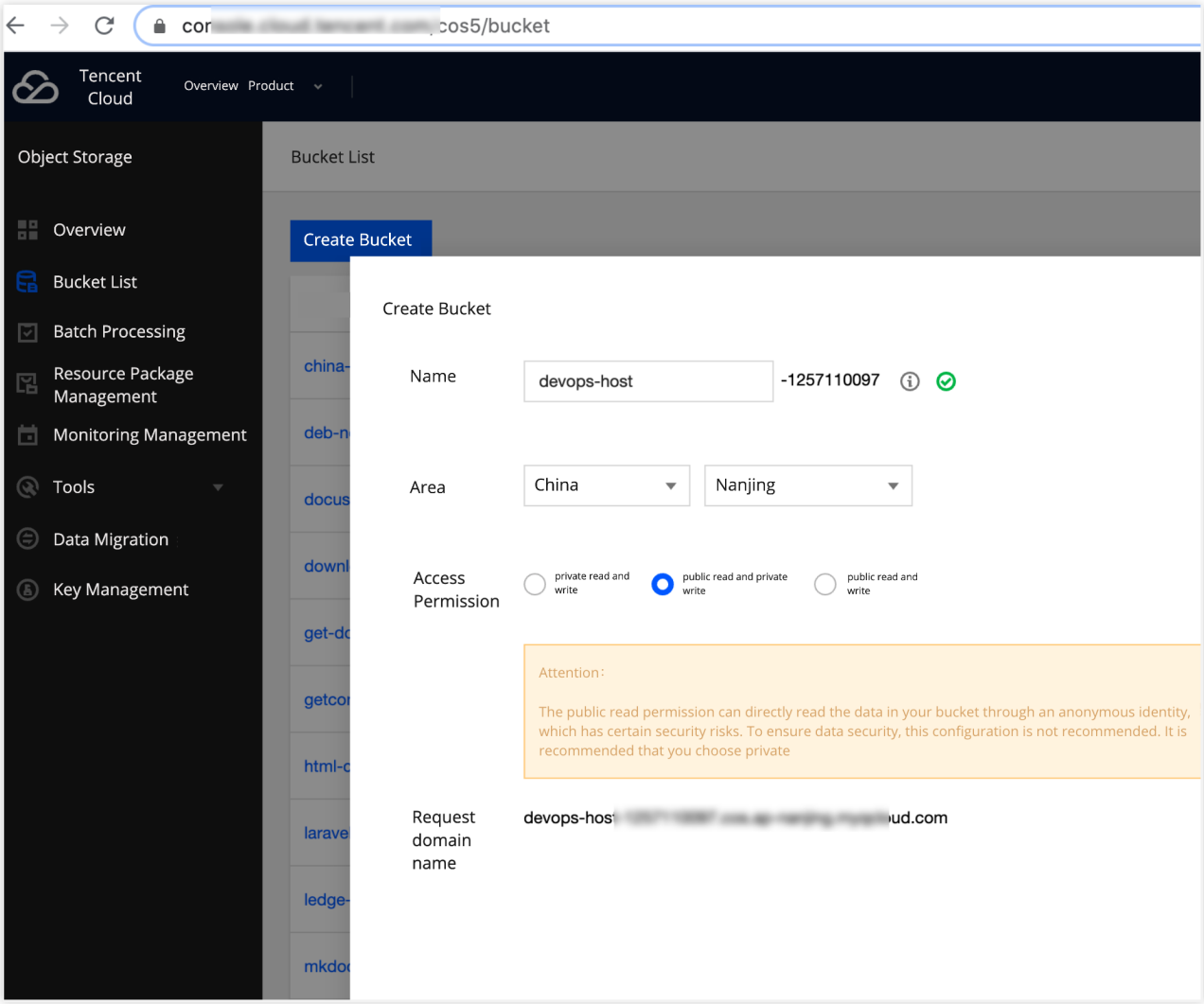
Function Overview

Tencent Cloud's auto-scaling storage allows you to release a project to COS with one click through Continuous Integration, which is applicable for scenarios such as building a static website or compiling files for download.

Create Bucket

Create a **bucket** in cloud storage (such as Tencent Cloud's COS) and retrieve the bucket name, region, and secret key.

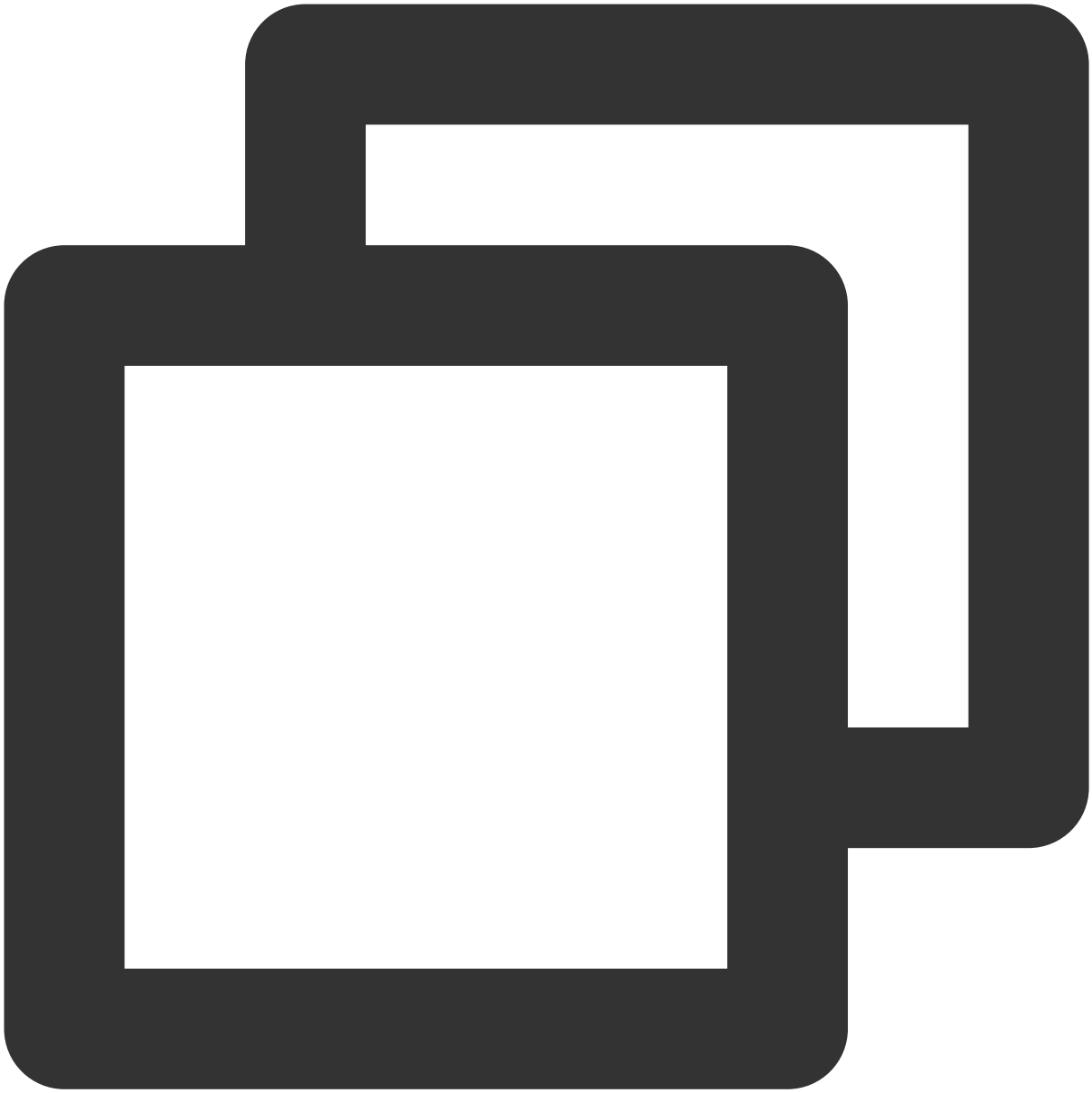
©2013-2022 Tencent Cloud. All rights reserved.



Jenkinsfile

In Continuous Integration, refer to and write the following Jenkinsfile to trigger a build task and upload files.

©2013-2022 Tencent Cloud. All rights reserved.



```
pipeline {
  agent any
  stages {
    stage('Check out') {
      steps {
        checkout([$class: 'GitSCM', branches: [[name: env.GIT_BUILD_REF]],
        userRemoteConfigs: [[url: env.GIT_REPO_URL, credentialsId: env.CREDENTIALS_
      }
    }
    stage('Compile') {
      steps {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        // Convert markdown to HTML
        // sh 'pip install mkdocs && mkdocs build'
        // React/VUE SPA generate HTML
        // sh 'npm run build'
        // Create Android package
        // sh './gradlew assembleDebug'
      }
    }
    stage('Upload to Tencent Cloud COS') {
      steps {
        sh "coscmd config -a ${env.COS_SECRET_ID} -s ${env.COS_SECRET_KEY}" +
           " -b ${env.COS_BUCKET_NAME} -r ${env.COS_BUCKET_REGION}"
        sh "rm -rf .git"
        sh 'coscmd upload -r ./ /'
        //sh 'coscmd upload -r ./dist /'
      }
    }
  }
}
```

Environment Variables

Variable	Description	Example
COS_SECRET_ID	Key ID for accessing Tencent Cloud	stringLength36stringLength36string36
COS_SECRET_KEY	Secret key for accessing Tencent Cloud	stringLength32stringLength323232
COS_BUCKET_NAME	Tencent Cloud COS bucket	devops-host-1257110097
COS_BUCKET_REGION	Tencent Cloud COS region	ap-nanjing

©2013-2022 Tencent Cloud. All rights reserved.

Use Docker in Continuous Integration

Last updated：2023-12-29 11:44:50

This document describes how to use Docker in Continuous Integration.

Prerequisites

Before configuring the CODING Continuous Integration (CODING-CI) build environment, you must activate the CODING DevOps service for your Tencent Cloud account.

Open Project

1. Log in to the CODING Console and click the **team domain name** to go to CODING.
2. Click

in the upper-right corner to open the project list page and click a **project icon** to open the project.
3. Select **Continuous Integration** from the menu on the left.

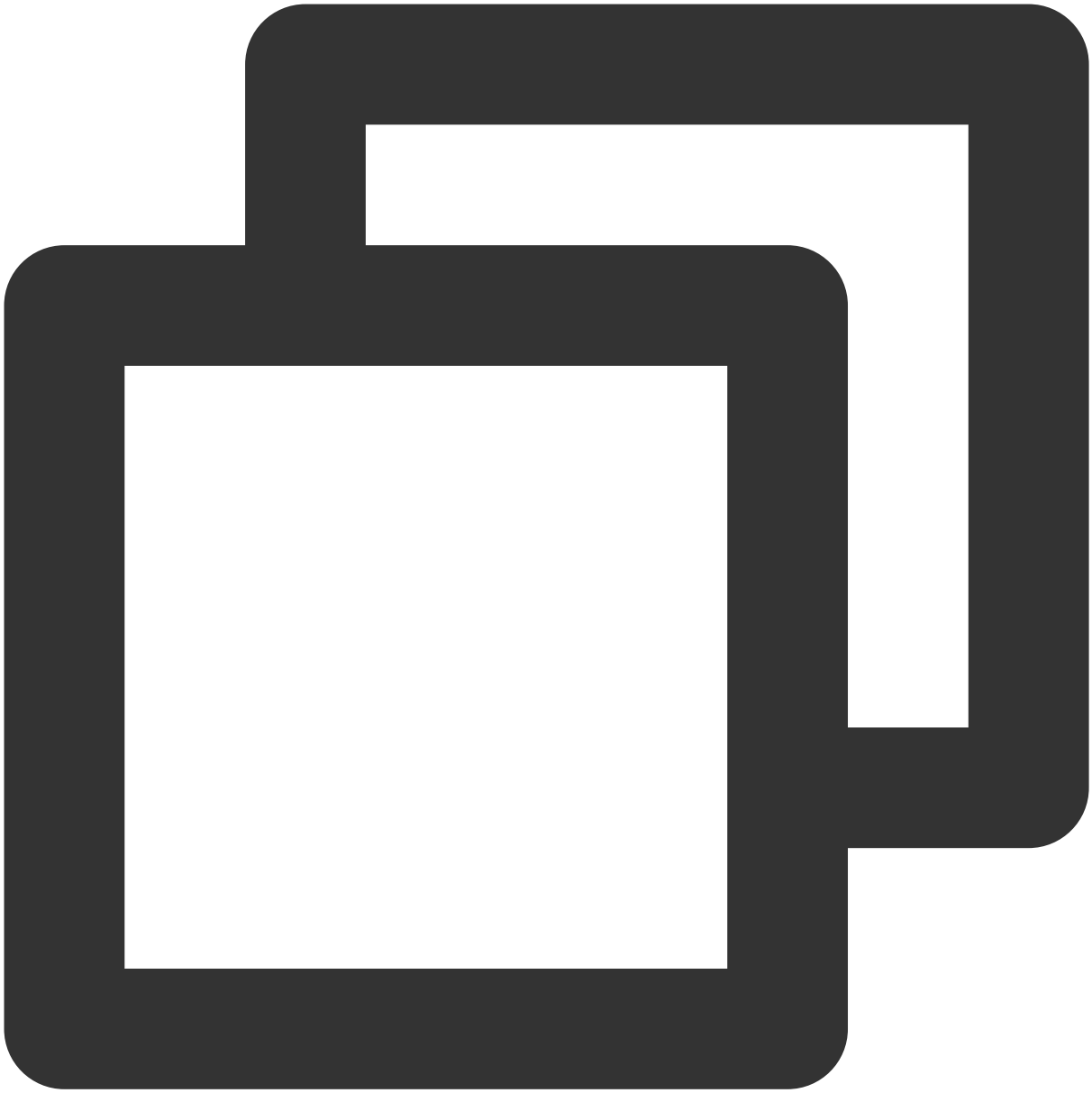
Overview

Besides using Docker as a build environment for Continuous Integration, you may need to run additional services in Docker as test dependencies, or build a Docker image in a CI process and push it to the relevant repository.

Run Specific Docker Image and Execute Commands

In a build process, you may need to use a public Docker image repository. Refer to the following Jenkinsfile for the command to pull a specific Docker image.

©2013-2022 Tencent Cloud. All rights reserved.



```
pipeline {
    agent any
    stages {
        stage('Test') {
            steps {
                  script {
                    docker.image("ubuntu").inside('-e MY_ENV=123') {
                          sh 'echo ${MY_ENV}'
                    }
                  }
            }
```

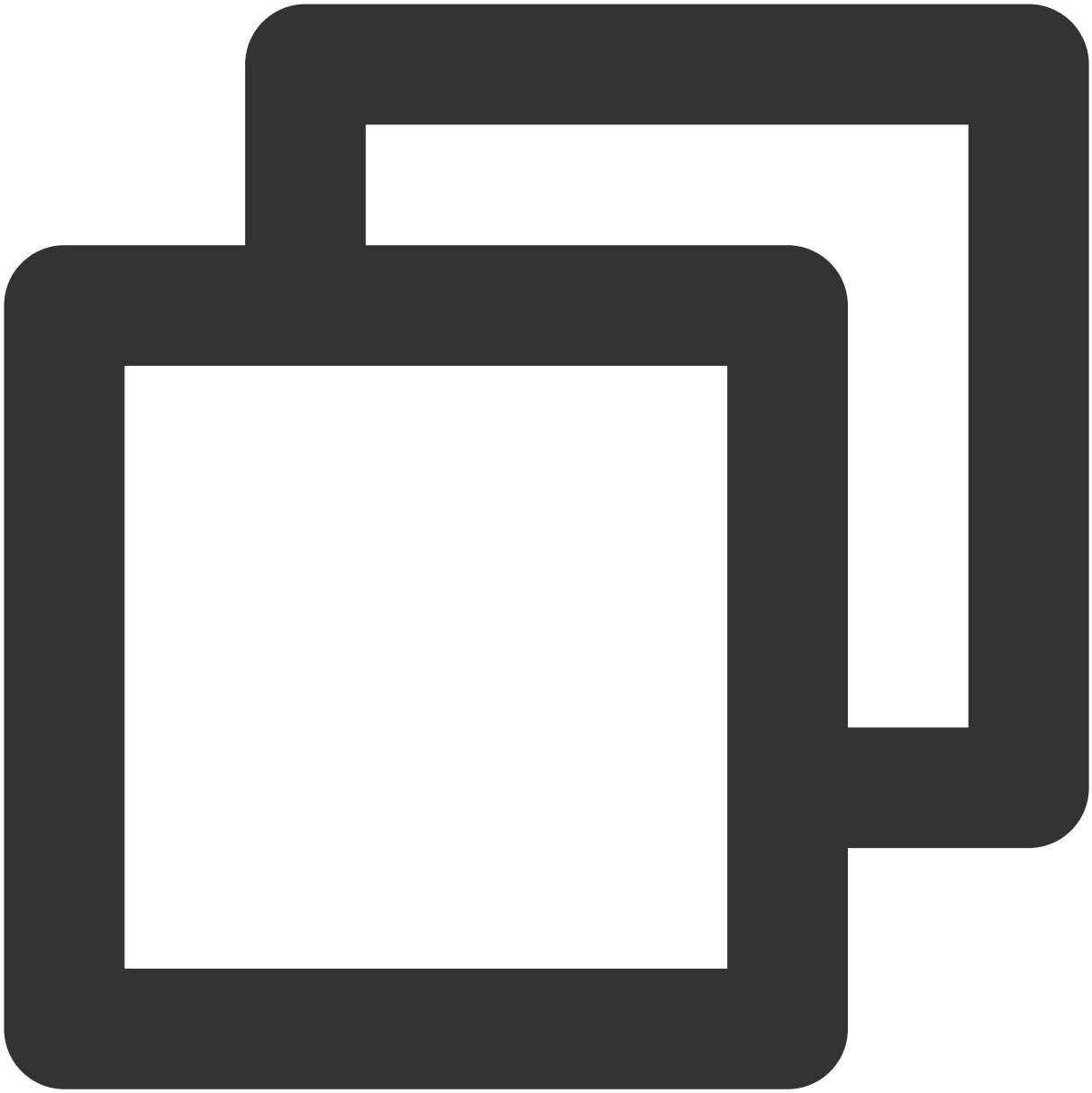
©2013-2022 Tencent Cloud. All rights reserved.

```
            }
        }
    }
```

Run Docker Image of Specific Registry

In a build process, you may need to use a private Docker image repository. For example, you might need to use a Docker image repository that has been uploaded to the CODING Artifact Repository (CODING-AR). Refer to the following Jenkinsfile.

©2013-2022 Tencent Cloud. All rights reserved.

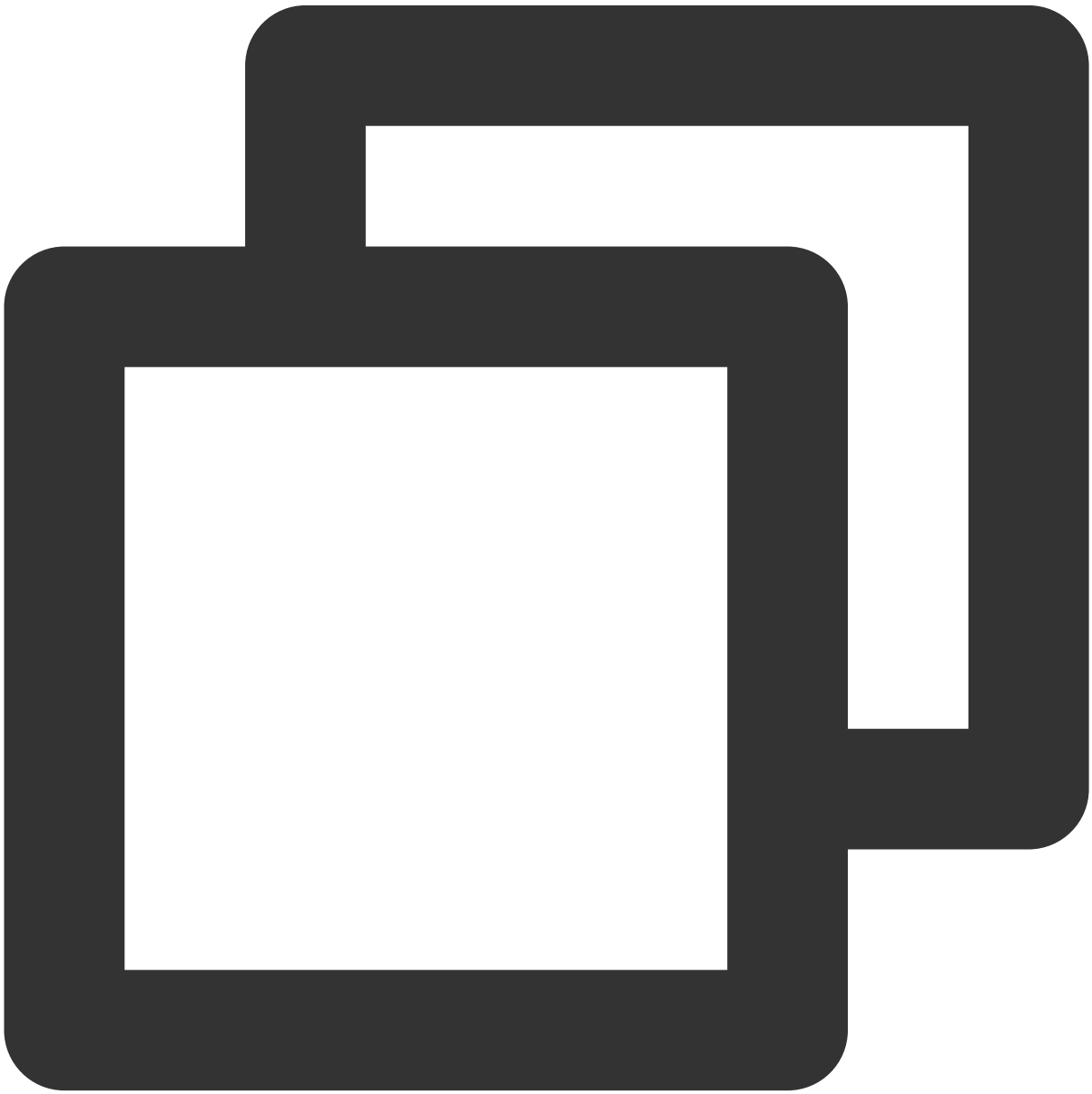


```
pipeline {
    agent any
    stages {
        stage('Test') {
            steps {
                  script {
                    docker.withRegistry('https://registry.example.com') {

                        // Pulls my-custom-image from the hostname registry.example
                        docker.image('my-custom-image').inside {
                            sh 'make test'
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                                }
                            }
                        }
                    }
                }
            }
        }
```

Refer to the following Jenkinsfile in the case that a configured registry requires authentication to pull the image and needs a valid credential ID.

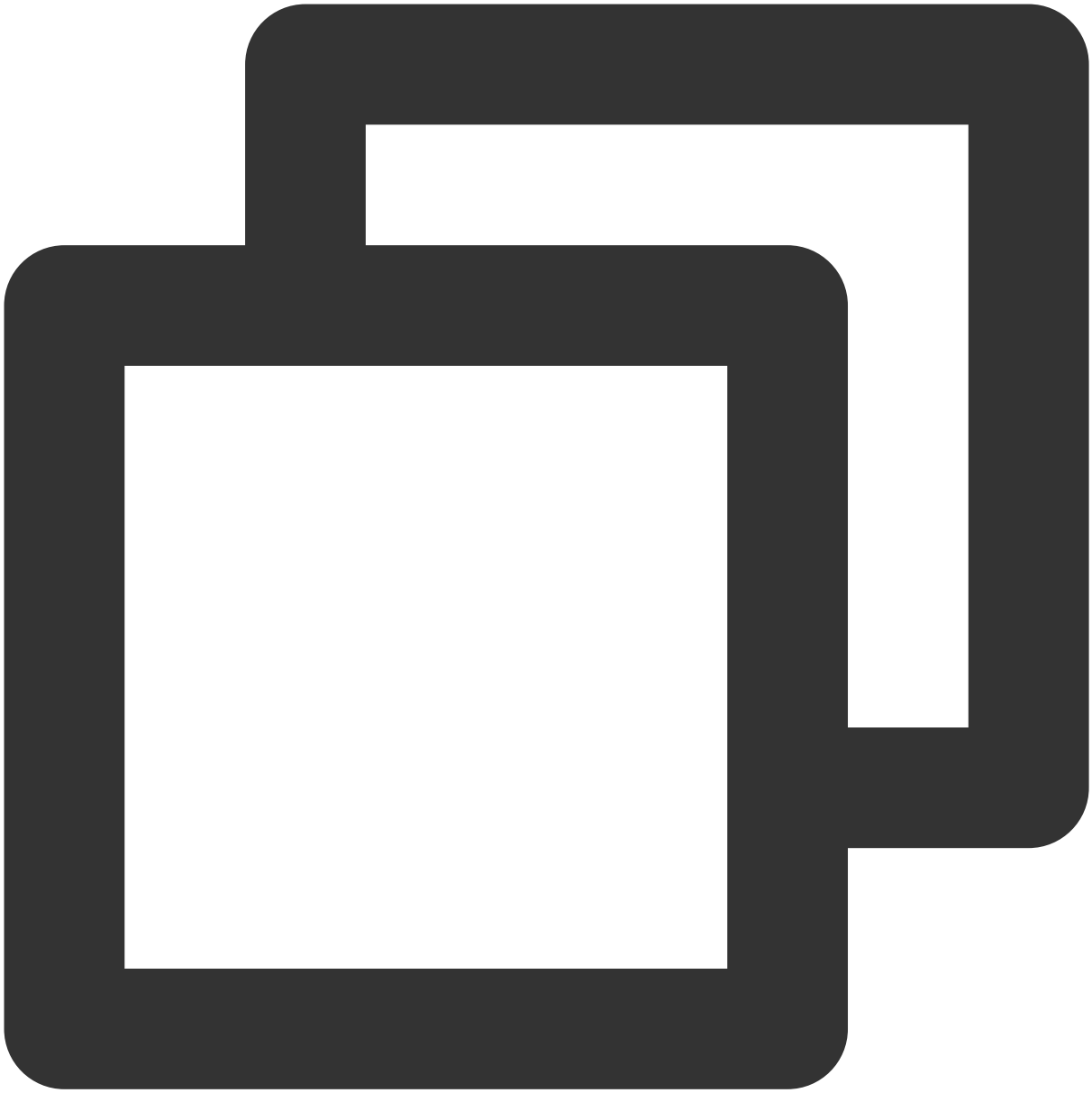


©2013-2022 Tencent Cloud. All rights reserved.

```
pipeline {
    agent any
    stages {
        stage('Test') {
            steps {
                  script {
                    docker.withRegistry('https://registry.example.com', 'my-credent

                    }
                  }
            }
        }
    }
}
```

Build Docker Image in CI Process

©2013-2022 Tencent Cloud. All rights reserved.



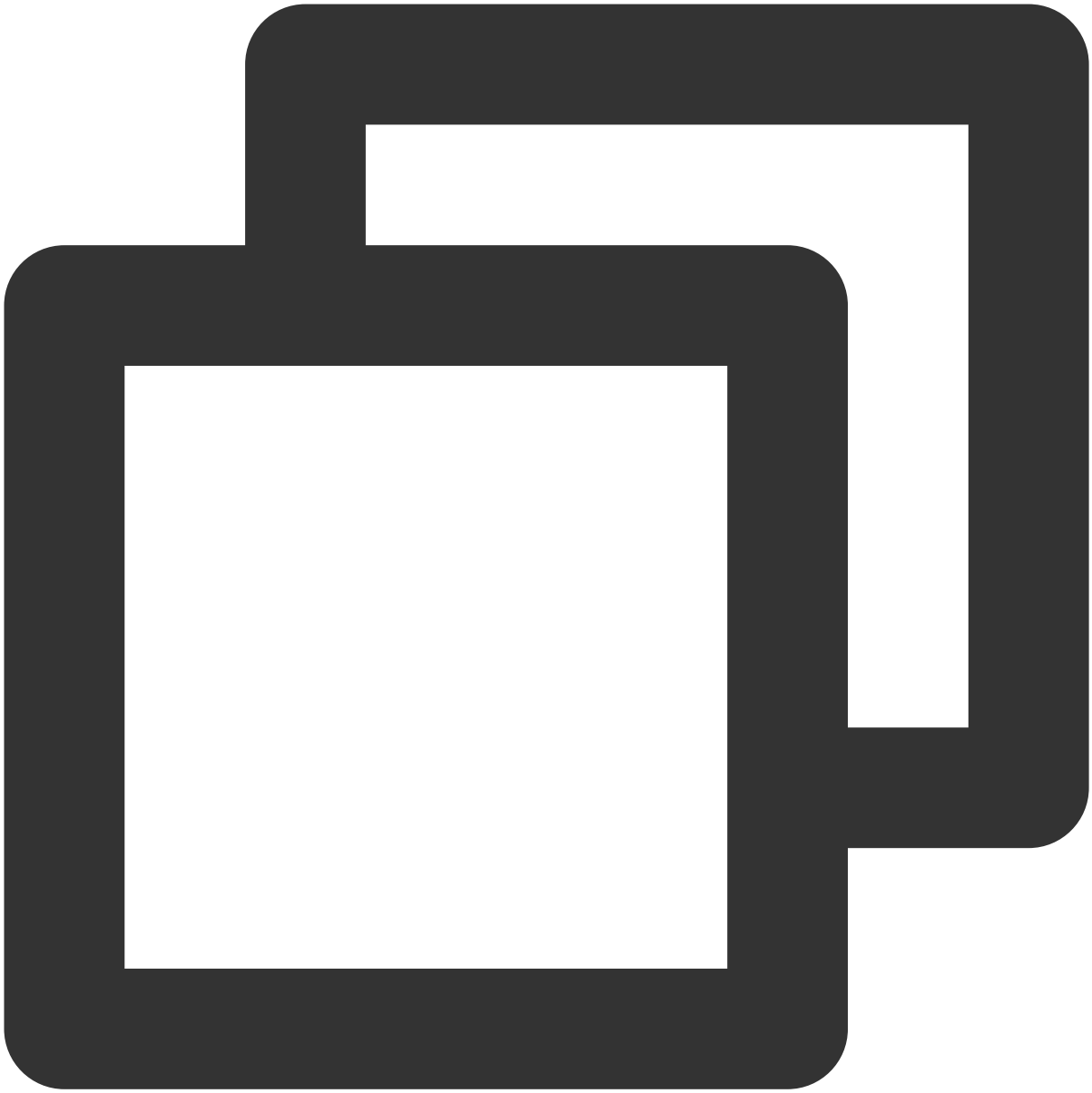
```
pipeline {
    agent any
    stages {
        // You need to check out the code before using the Dockerfile in the code r
        stage('Checkout') {
            steps {
                checkout([
                    $class: 'GitSCM',
                    branches: [[name: env.GIT_BUILD_REF]],
                    userRemoteConfigs: [[url: env.GIT_REPO_URL, credentialsId: env.
            }
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        }
        stage('Build') {
            steps {
                script {
                    // Uses the root path Dockerfile to build by default
                    docker.build('my-docker-image:1.0.0')
                }
            }
        }
    }
}
```

If you need to specify additional parameters for a build, such as using a Dockerfile in a specific directory, refer to the following Jenkinsfile.

©2013-2022 Tencent Cloud. All rights reserved.



```
pipeline {
    agent any
    stages {
        // You need to check out the code before using the Dockerfile in the code r
        stage('Checkout') {
            steps {
                checkout([
                    $class: 'GitSCM',
                    branches: [[name: env.GIT_BUILD_REF]],
                    userRemoteConfigs: [[url: env.GIT_REPO_URL, credentialsId: env.
            }
```

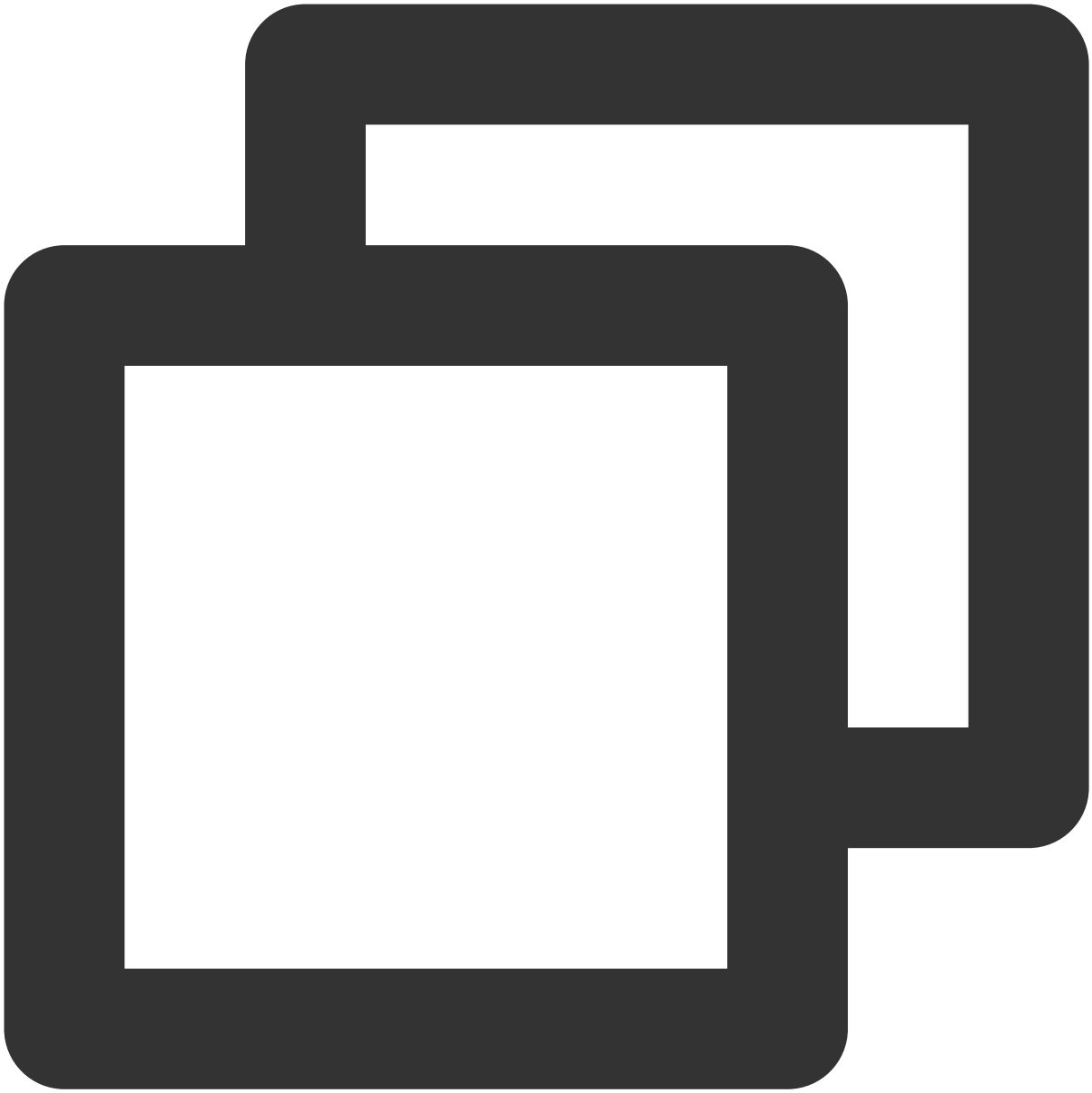
©2013-2022 Tencent Cloud. All rights reserved.

```
        }
        stage('Build') {
            steps {
                script {
                    // Uses /dockerfiles/Dockerfile.build to build
                    docker.build('my-docker-image:1.0.0', '-f Dockerfile.build ./do
                }
            }
        }
    }
}
```

Push docker image to Specific Registry

Refer to the following Jenkinsfile.

©2013-2022 Tencent Cloud. All rights reserved.



```
pipeline {
    agent any
    stages {
        // You need to check out the code before using the Dockerfile in the code r
        stage('Checkout') {
            steps {
                checkout([
                    $class: 'GitSCM',
                    branches: [[name: env.GIT_BUILD_REF]],
                    userRemoteConfigs: [[url: env.GIT_REPO_URL, credentialsId: env.
                ])
```

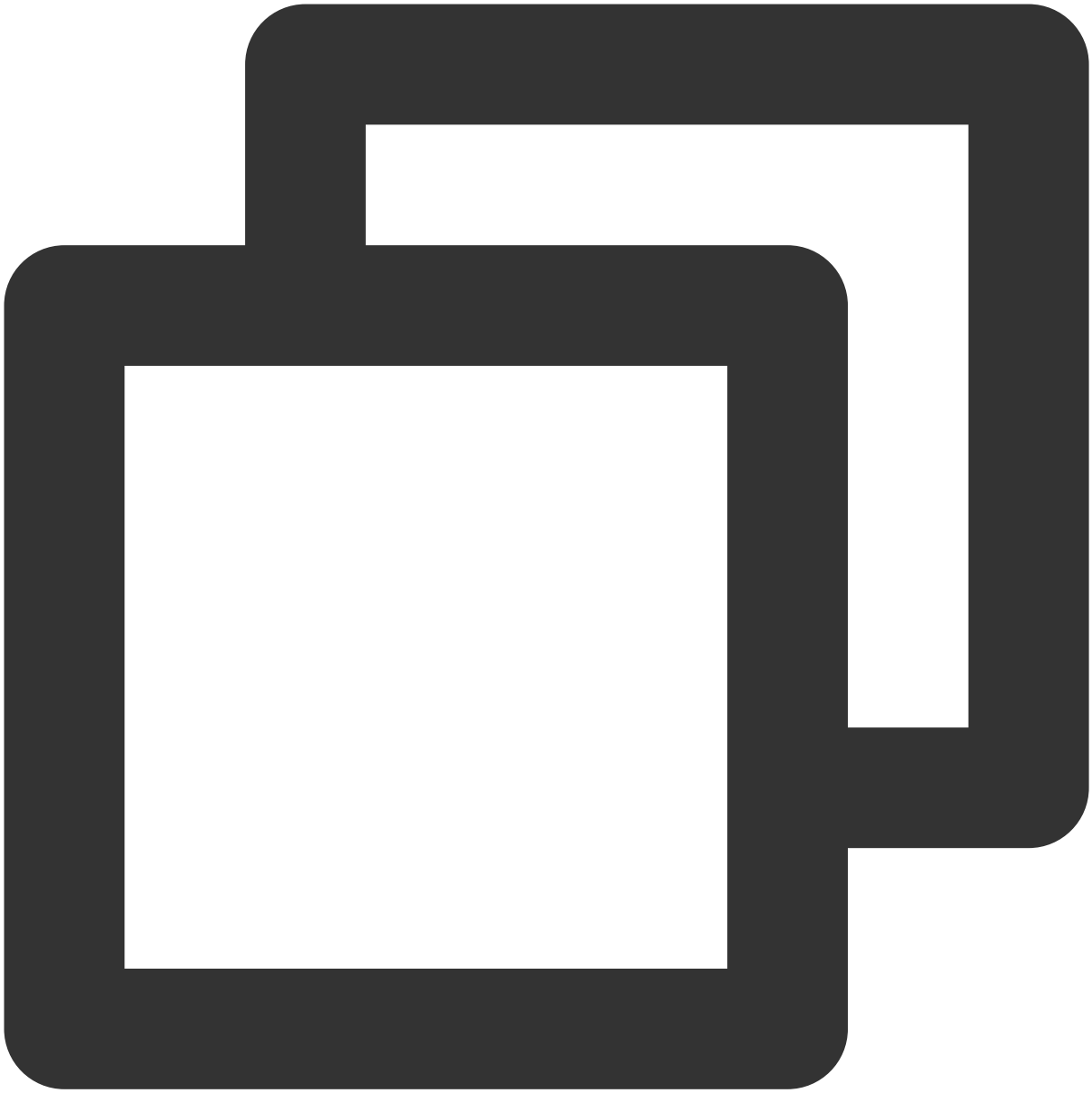
©2013-2022 Tencent Cloud. All rights reserved.

```
            }
        }

        stage('Build') {
            steps {
                  script {
                      docker.build('my-docker-image:1.0.0')

                    docker.withRegistry('https://registry.example.com', 'my-credent
                        docker.image('my-docker-image:1.0.0').push()
                    }
                  }
            }
        }
    }
}
```

Use Docker to Run Additional Services as Test Dependencies

In the test process, you can use Docker to run MySQL and other services that can be used as test dependencies. Two containers are used in the following example: one as a MySQL service and the other as an execution environment. (Use a Docker link to link the two containers.)

©2013-2022 Tencent Cloud. All rights reserved.



```
pipeline {
    agent any
    stages {
        stage('Test') {
            steps {
                  script {
                    docker.image('mysql:5').withRun('-e "MYSQL_ROOT_PASSWORD=my-sec
                        // Note: The callback run environment is not the MySQL:5 en

                        // Runs the second MySQL as the execution environment
```

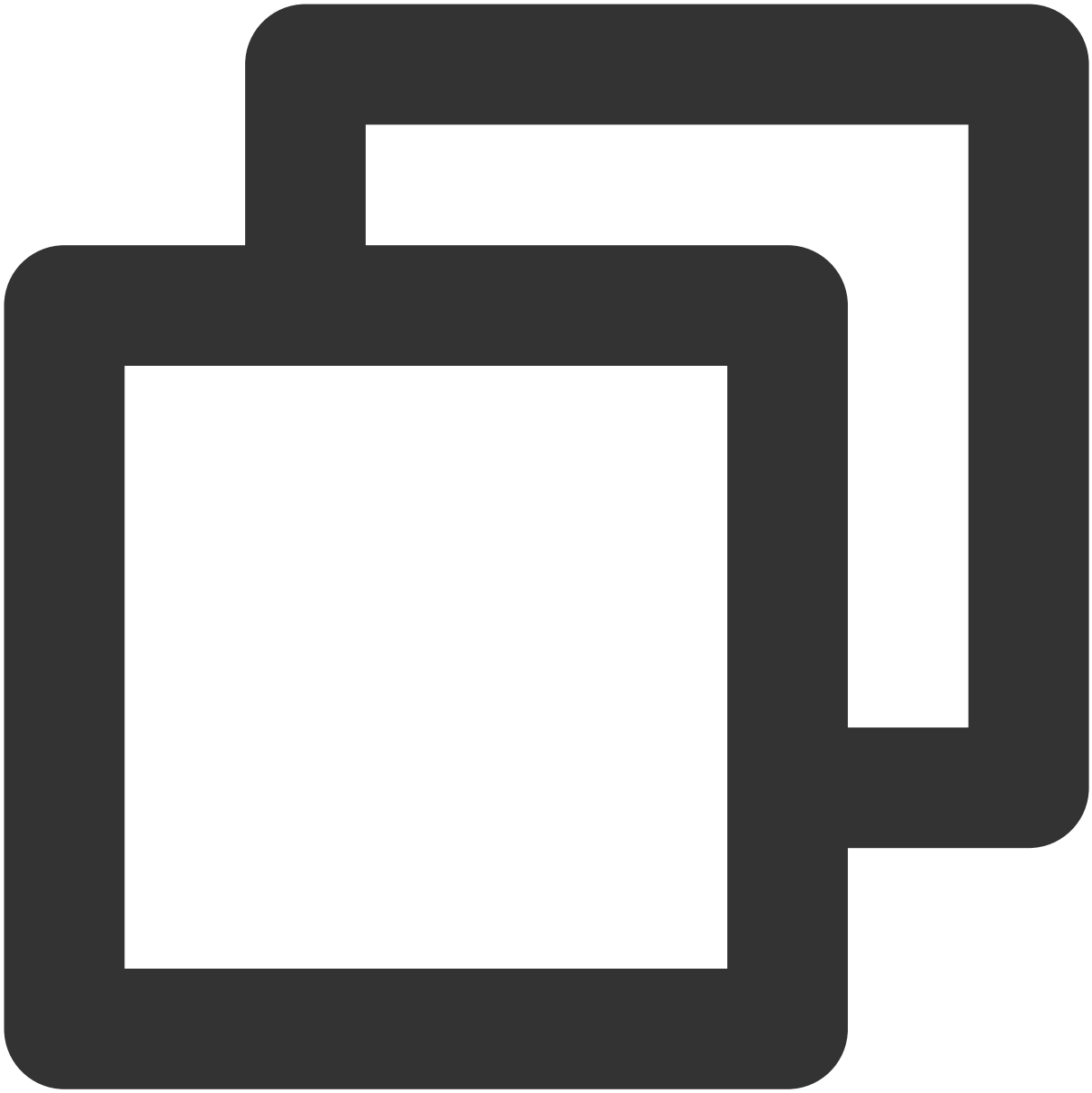
©2013-2022 Tencent Cloud. All rights reserved.

```
                        docker.image('mysql:5').inside("--link ${c.id}:db") {
                          // The commands run here are all in the second MySQL Dock
                          // Waits for the MySQL service
                          sh 'while ! mysqladmin ping -hdb --silent; do sleep 1; do
                        }

                        // After the callback content finishes running, the MySQL D
                    }
                }
            }
        }
    }
}
```

Run Multiple Containers at the Same Time as Test Dependencies

If you need more than one additional service as test dependencies, you can run multiple services in a nested way.

©2013-2022 Tencent Cloud. All rights reserved.



```
pipeline {
    agent any
    stages {
        stage('Test') {
            steps {
                  script {
                    docker.image('mysql:5').withRun('-e "MYSQL_ROOT_PASSWORD=my-sec
                        // Note: The callback run environment is not the MySQL:5 en

                        docker.image('redis').withRun('') { c2 ->
                            // Note: The callback run environment is not the Redis
```

©2013-2022 Tencent Cloud. All rights reserved.

```
                                sh 'docker ps'
                            }
                        }
                    }
                }
            }
        }
    }
```

References

For more information about Docker-based configuration in Jenkins, see the official Jenkins documentation:

Using Docker with Pipeline

Pipeline Syntax: agent

©2013-2022 Tencent Cloud. All rights reserved.

Use SSH in Continuous Integration

Last updated：2023-12-29 11:44:51

This document describes how to use SSH in Continuous Integration.

Prerequisites

Before configuring the CODING Continuous Integration (CODING-CI) build environment, you must activate the CODING DevOps service for your Tencent Cloud account.

Open Project

1. Log in to the CODING Console and click the **team domain name** to go to CODING.
2. Click

in the upper-right corner to open the project list page and click a **project icon** to open the project.
3. Select **Continuous Integration** from the menu on the left.

Function Overview

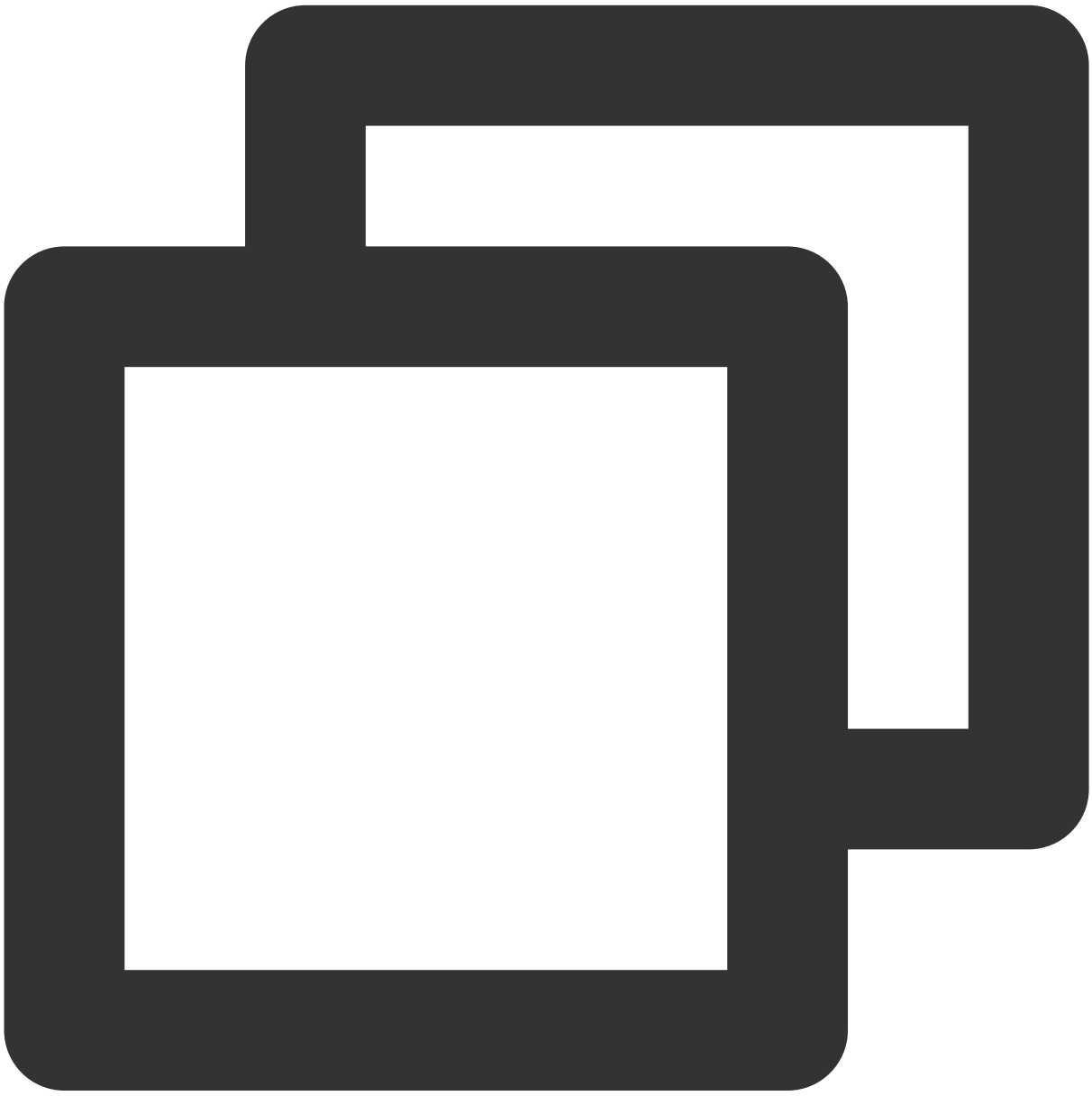
When executing a build in Continuous Integration, you may need to log in to a remote server with SSH protocol to execute the necessary script or command. Go to **Continuous Integration** > "Build Plan Settings" > "Process Configuration", use the text editor to enter the relevant command.

How to Use SSH Commands

CODING-CI allows you to control a remote server using SSH commands.
sshCommand: Run a specific command on the remote server.
sshPut: Place files or directories of the current workspace in the remote server.
sshGet: Obtain files or directories from a remote server.

©2013-2022 Tencent Cloud. All rights reserved.

sshScript: Read the local shell script and run it on the remote server. If you run the script of the remote server, you will get the error: does not exist.
sshRemove: Remove a certain file or directory from a remote server.
The following example shows how to use an account and password to connect to a remote server and run SSH commands. An example of a Jenkinsfile configuration is as follows:



```
def remote = [:]
remote.name = "node"
remote.host = "node.abc.com"
remote.allowAnyHosts = true
```

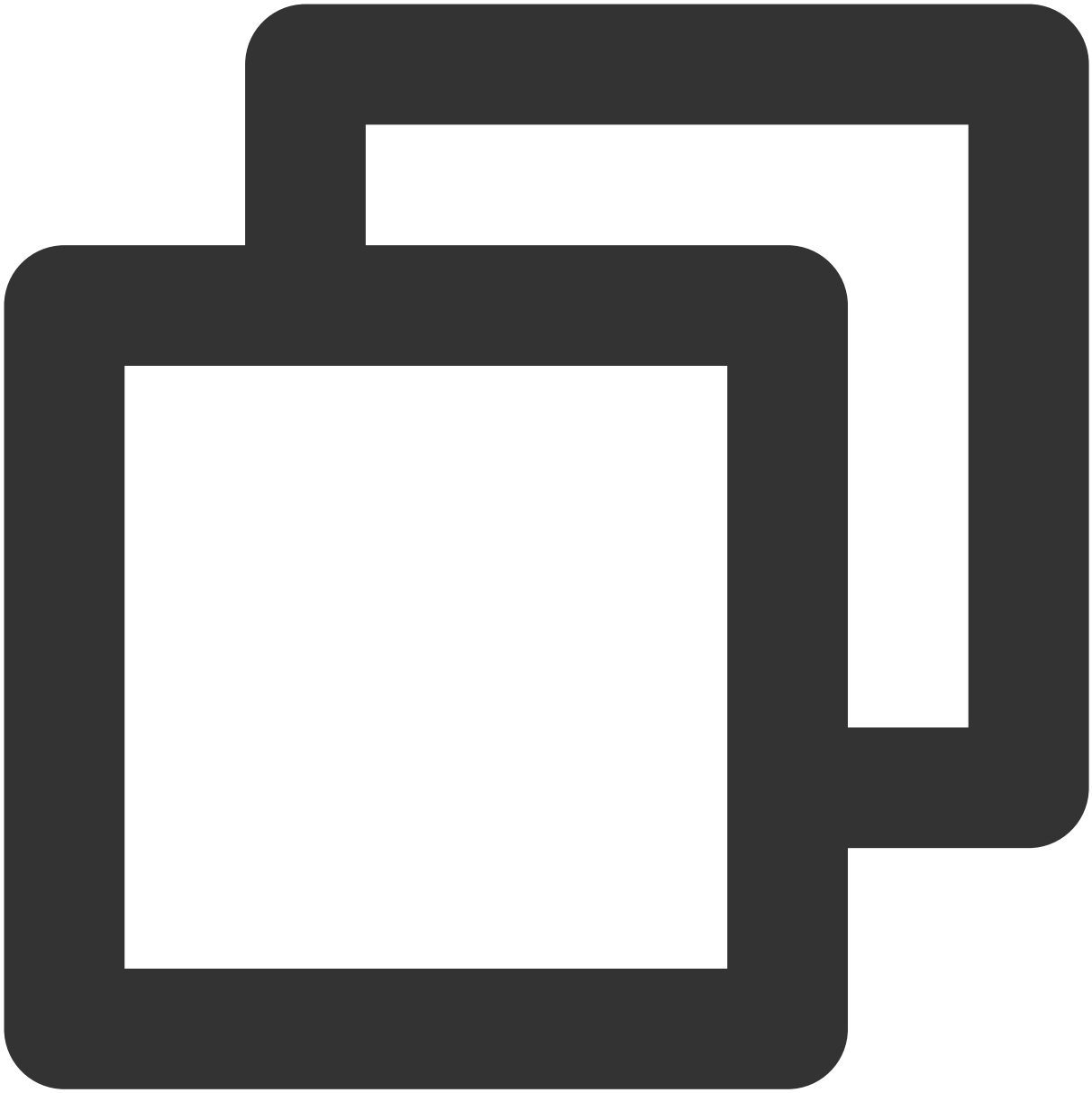
©2013-2022 Tencent Cloud. All rights reserved.

```
node {
    withCredentials([usernamePassword(credentialsId: 'sshUserAcct',
        passwordVariable: 'password', usernameVariable: 'userName')]) {
        remote.user = userName
        remote.password = password

        stage("SSH Steps Rocks!") {
            writeFile file: 'test.sh', text: 'ls'
            sshCommand remote: remote,
                command: 'for i in {1..5}; do echo -n \\"Loop \\$i \\"; date ; slee
            sshScript remote: remote, script: 'test.sh'
            sshPut remote: remote, from: 'test.sh', into: '.'
            sshGet remote: remote, from: 'test.sh', into: 'test_new.sh', override:
            sshRemove remote: remote, path: 'test.sh'
        }
    }
}
```

How to Use SSH to Connect to a Remote Service

Besides using an account and password to connect to a remote server, you can also use an SSH private key to connect to a remote service. An example of a Jenkinsfile configuration is as follows:

©2013-2022 Tencent Cloud. All rights reserved.



```
def remote = [:]
remote.name = "node"
remote.host = "node.abc.com"
remote.allowAnyHosts = true

node {
    withCredentials([sshUserPrivateKey(credentialsId: 'sshUser', keyFileVariable: '
        // SSH login username
        remote.user = 'root'
        // Private key file address
        remote.identityFile = identity
```

©2013-2022 Tencent Cloud. All rights reserved.

```
        stage("SSH Steps Rocks!") {
            writeFile file: 'abc.sh', text: 'ls'
            sshCommand remote: remote,
                command: 'for i in {1..5}; do echo -n \\"Loop \\$i \\"; date ; slee
            sshPut remote: remote, from: 'abc.sh', into: '.'
            sshGet remote: remote, from: 'abc.sh', into: 'bac.sh', override: true
            sshScript remote: remote, script: 'abc.sh'
            sshRemove remote: remote, path: 'abc.sh'
        }
    }
}
```

More Information

For more information on SSH commands in Jenkinsfile, see the official Jenkins Help Documentation.

For more information about Jenkins SSH plugins, see the plugin's official homepage.

©2013-2022 Tencent Cloud. All rights reserved.