

代码检查

最佳实践

文档版本

01

发布日期

2025-01-16



版权所有 © 华为技术有限公司 2025。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目录

1 CodeArts Check 最佳实践汇总.....	1
2 使用预置规则检查 GitCode 代码仓中的代码质量.....	3
3 使用预置规则检查通用 Git 代码仓中的代码质量.....	6
4 使用自定义规则检查 CodeArts Repo 代码仓中的代码质量.....	9
5 不上传代码到云服务的情况下使用代码检查服务.....	14
6 使用自定义执行机执行代码检查任务.....	19
7 CodeArts Check 通过调用 API 执行 MR 增量检查.....	25
8 使用 Jenkins 插件集成 CodeArts Check 执行代码检查.....	31
9 HE2E DevOps 实践：代码检查.....	37

1 CodeArts Check 最佳实践汇总

本文汇总了基于代码检查服务（CodeArts Check）常见应用场景的操作实践，为每个实践提供详细的方案描述和操作指导，帮助用户轻松检查代码。

表 1-1 CodeArts Check 最佳实践一览表

最佳实践	说明
2 使用预置规则检查 GitCode 代码仓中的代码质量	如果您的代码存放在GitCode代码仓中，可参考本实践的操作指导完成检查代码质量。本实践为您介绍如何使用系统预置规则检查GitCode代码仓中的Java语言代码质量。
3 使用预置规则检查通用Git代码仓中的代码质量	如果您的代码存放在通用Git代码仓中，可参考本实践的操作指导完成检查代码质量。本实践为您介绍如何使用系统预置规则检查通用Git代码仓中的Java语言代码质量。
4 使用自定义规则检查 CodeArts Repo 代码仓中的代码质量	<p>随着现在项目的代码量越来越大，以及现有开发框架的增加，静态分析所需要覆盖的场景也日益增多，存在的问题如下：</p> <ul style="list-style-type: none">• 传统静态分析引擎靠工具提供通用规则来对常见安全，风格和质量场景下的编码问题进行扫描分析，难以实时覆盖不同用户特定场景下的代码问题。• 用户往往不熟悉工具提供的通用规则所能覆盖的全部场景。在为新开发的业务场景寻找适用的规则时需要耗费大量的时间，增加开发成本。• 不同用户之间的代码差异大，研发无法实时掌握不同用户的具体需求，在开发规则过程中难以针对不同用户的业务代码场景来制定全面有效的工具规则。 <p>本示例为您介绍如何使用自定义规则检查代码中存在调试代码的问题。</p>
6 使用自定义执行机执行代码检查任务	当CodeArts Check提供的内置执行机不满足业务要求时，您可接入自行提供的执行机，通过注册的方式托管到CodeArts Check服务中，委托CodeArts Check服务进行调度并执行代码检查任务。本实践我们通过检查CodeArts Repo代码仓的代码问题来演示使用自定义执行机执行代码检查任务。

最佳实践	说明
9 HE2E DevOps实践：代码检查	本文以“DevOps全流程示例项目”为例，介绍如何在项目中完成代码检查配置。

2 使用预置规则检查 GitCode 代码仓中的代码质量

应用场景

如果您的代码存放在GitCode代码仓中，可参考本实践的操作指导完成检查代码质量。本实践为您介绍如何使用系统预置规则检查GitCode代码仓中的Java语言代码质量。

前提准备

- 已[开通并授权使用代码检查服务](#)授权使用代码检查服务。
- GitCode代码仓中已有Java语言的代码。


操作流程

表 2-1 操作流程

序号	步骤	说明
1	创建代码检查任务所属项目	创建代码检查任务所属的项目。
2	新建GitCode服务扩展点	本实践检查第三方代码仓的代码质量，CodeArts Check服务需通过CodeArts提供的扩展插件连接到第三方代码仓。
3	新建GitCode代码检查任务	创建代码检查任务。
4	执行GitCode代码检查任务	执行代码检查任务。
5	查看GitCode代码检查详情	查看代码检查结果。

创建代码检查任务所属项目

步骤1 使用华为云账号[登录华为云控制台页面](#)。

步骤2 单击页面左上角，在服务列表中选择“开发与运维 > 软件开发生产线 CodeArts”。

步骤3 单击“立即使用”，进入CodeArts服务首页。

步骤4 在首页单击“新建项目”，选用“Scrum”项目模板。项目名称填写“Scrum01”，其他参数保持默认即可。

步骤5 单击“确定”后，进入到“Scrum01”项目下。

----结束

新建 GitCode 服务扩展点

服务扩展点（Endpoint）是软件开发生产线的一种扩展插件，为软件开发生产线提供链接第三方代码仓的能力。

代码检查服务默认检查CodeArts Repo服务的代码质量，同时也可使用服务扩展点连接第三方代码仓库检查代码质量。

步骤1 在项目下的CodeArts Check服务页面的导航栏选择“设置 > 通用设置 > 服务扩展点管理”。

步骤2 单击“新建服务扩展点”，在下拉列表中选择“GitCode”。

步骤3 参考[表2-2](#)在弹框中配置参数。

表 2-2 GitCode 服务扩展点参数说明

参数	说明
连接名称	服务扩展点的名称，可自定义。例如：GitCode。
Token	填写GitCode上获取的Token，获取方法如下： <ol style="list-style-type: none"> 登录GitCode。 单击页面右上角账号名称，选择“个人设置”。 单击“+访问令牌”，填写令牌名称以及到期时间。 单击“新建访问令牌”，生成“你的个人访问令牌”，即Token。 单击，即可复制生成的Token。 <p>说明</p> <ul style="list-style-type: none"> Token生成后，请及时保存，下次刷新页面将无法读取，需要重新生成新Token。 注意填写有效的Token描述信息，避免误删除导致构建失败。 无需使用时，请及时删除Token，避免信息泄露。

步骤4 单击“确定”。

----结束

新建 GitCode 代码检查任务

- 步骤1** 选择导航栏“代码 > 代码检查”。
- 步骤2** 单击“新建任务”，进入新建任务页面，配置如下参数。


表 2-3 GitCode 代码检查任务参数说明

参数	说明
归属项目	任务所属项目。默认填写为 创建代码检查任务所属项目 中的项目名称“Scrum01”，无需手动填写。
代码源	选择需要检查的代码来源。选择“GitCode”。
任务名称	代码检查任务名称，可自定义。例如：GitCodeCheckTask。。
Endpoint实例	选择 新建GitCode服务扩展点 中创建的服务扩展点“GitCode”。
仓库	选择需要检查的代码仓库，保持默认即可。
分支	选择需要检查的仓库分支，保持默认即可。
检查语言	选择需要检查的代码语言，选择“Java”。

- 步骤3** 单击“新建任务”，完成检查任务的创建。

----结束

执行 GitCode 代码检查任务

- 步骤1** 在代码检查页面任务列表中，单击“GitCodeCheckTask”代码检查任务所在行 。
- 步骤2** 根据页面提示等待任务执行完成。

----结束

查看 GitCode 代码检查详情

- 步骤1** 在代码检查页面任务列表中，搜索[新建GitCode代码检查任务](#)创建的任务名称。
- 步骤2** 单击任务名称，查看代码检查详情，包括概览、代码问题、代码度量、检查日志等。

----结束

3 使用预置规则检查通用 Git 代码仓中的代码质量

应用场景

如果您的代码存放在通用Git代码仓中，可参考本实践的操作指导完成检查代码质量。本实践为您介绍如何使用系统预置规则检查通用Git代码仓中的Java语言代码质量。

前提准备


- 已[开通并授权使用代码检查服务](#)授权使用代码检查服务。
- 通用Git代码仓中已有Java语言的代码。

操作流程

表 3-1 操作流程

序号	步骤	说明
1	创建代码检查任务所属项目	创建代码检查任务所属的项目。
2	新建通用Git服务扩展点	本实践检查第三方代码仓的代码质量，代码检查服务需通过CodeArts提供的扩展插件连接到第三方代码仓。
3	创建通用Git代码检查任务	创建代码检查任务。
4	执行通用Git代码检查任务	执行代码检查任务。
5	查看代码检查结果	查看代码检查结果。

创建代码检查任务所属项目

- 步骤1** 使用华为云账号[登录华为云控制台页面](#)。
- 步骤2** 单击页面左上角，在服务列表中选择“开发与运维 > 软件开发生产线 CodeArts”。
- 步骤3** 单击“立即使用”，进入CodeArts服务首页。
- 步骤4** 在首页单击“新建项目”，选用“Scrum”项目模板。项目名称填写“Scrum01”，其他参数保持默认即可。
- 步骤5** 单击“确定”后，进入到“Scrum01”项目下。
- 结束

新建通用 Git 服务扩展点

服务扩展点（Endpoint）是软件开发生产线的一种扩展插件，为软件开发生产线提供链接第三方代码仓的能力。

代码检查服务默认检查CodeArts Repo服务的代码质量，同时也可使用服务扩展点连接第三方代码仓库检查代码质量。

- 步骤1** 在项目下的CodeArts Check服务页面的导航栏选择“设置 > 通用设置 > 服务扩展点管理”。
- 步骤2** 单击“新建服务扩展点”，在下拉列表中选择“通用Git”。
- 步骤3** 在弹框中配置以下信息，单击“确定”。

表 3-2 新建通用 Git 服务扩展点

参数	说明
连接名称	自定义，支持中文、英文、数字、“-”、“_”、“.”、空格，不超过256个字符。例如：Endpoint01。
Git仓库Url	输入待连接的Git仓库的https地址。
用户名	输入待连接的Git仓库的用户名，不超过300个字符。
密码或 Access Token	输入待连接的Git仓库的密码，不超过300个字符。

----结束

创建通用 Git 代码检查任务

- 步骤1** 在导航栏选择“代码 > 代码检查”。
- 步骤2** 单击“新建任务”，进入“新建任务”页面，参考[表3-3](#)配置参数。


表 3-3 通用 Git 代码检查任务参数说明

参数	说明
归属项目	任务所属项目。默认填写为 创建代码检查任务所属项目 中的项目名称“Scrum01”，无需手动填写。
代码源	选择“通用Git”。
任务名称	代码检查任务名称，可自定义。例如：CheckTask01。
Endpoint实例	选择 新建通用Git服务扩展点 中创建的服务扩展点“Endpoint01”。
仓库	保持默认即可。
分支	保持默认“master”即可。
检查语言	选择需要检查的代码语言，选择“Java”。

步骤3 单击“新建任务”，完成检查任务的创建。

----结束

执行通用 Git 代码检查任务

步骤1 在代码检查页面任务列表中，单击代码检查任务所在行  执行任务。

步骤2 根据页面提示等待任务执行完成。

----结束

查看代码检查结果

步骤1 在代码检查页面任务列表中，搜索[创建通用Git代码检查任务](#)创建的任务名称“CheckTask01”。

步骤2 单击任务名称，查看代码检查详情，包括概览、代码问题、代码度量、检查日志等。至此，我们完成了一次基本的通用Git代码源常用检查流程。

----结束

4 使用自定义规则检查 CodeArts Repo 代码仓中的代码质量

应用场景

随着现在项目的代码量越来越大，以及现有开发框架的增加，静态分析所需要覆盖的场景也日益增多，存在的问题如下：

- 传统静态分析引擎靠工具提供通用规则来对常见安全，风格和质量场景下的编码问题进行扫描分析，难以实时覆盖不同用户特定场景下的代码问题。
- 用户往往不熟悉工具提供的通用规则所能覆盖的全部场景。在为新开发的业务场景寻找适用的规则时需要耗费大量的时间，增加开发成本。
- 不同用户之间的代码差异大，研发无法实时掌握不同用户的具体需求，在开发规则过程中难以针对不同用户的业务代码场景来制定全面有效的工具规则。

本示例为您介绍如何使用自定义规则检查代码中存在调试代码的问题。

前提准备

- 已[开通并授权使用代码检查服务](#)授权使用代码检查服务。
- 通用Git代码仓中已有Java语言的代码。

操作流程


表 4-1 操作流程

序号	步骤	说明
1	创建代码检查任务所属项目	创建代码检查任务所属的项目。
2	创建CodeArts Repo代码仓	创建代码检查任务使用的代码仓。
3	创建规则文件	创建自定义规则时需要上传使用的规则文件。

序号	步骤	说明
4	创建自定义规则	在代码检查服务页面创建自定义规则。
5	创建自定义规则集	自定义规则无法直接使用，需要通过创建自定义规则集使用。
6	创建代码检查任务	创建使用自定义规则检查的代码检查任务。
7	使用自定义规则集执行检查任务	配置代码检查任务使用带有自定义规则的自定义规则集。
8	查看检查结果	查看检查结果，确认使用的规则是否生效。

创建代码检查任务所属项目

步骤1 使用华为云账号[登录华为云控制台页面](#)。

步骤2 单击页面左上角 ，在服务列表中选择“开发与运维 > 软件开发生产线 CodeArts”。

步骤3 单击“立即使用”，进入CodeArts服务首页。

步骤4 在首页单击“新建项目”，选用“Scrum”项目模板。项目名称填写“Scrum01”，其他参数保持默认即可。

步骤5 单击“确定”后，进入到“Scrum01”项目下。

----结束

创建 CodeArts Repo 代码仓

步骤1 在页面导航栏中选择“代码 > 代码托管”。

步骤2 进入代码托管页面，单击“新建仓库”，选择“模板仓库”。

步骤3 单击“下一步”，在“按模板新建 > 选择仓库模板”，搜索并选择“Java Ant Demo”模板。

步骤4 单击“下一步”，仓库名称填写“Repo01”，“自动创建代码检查任务”参数需要去除勾选。其他参数保持默认即可。

步骤5 单击“确定”完成仓库创建。

步骤6 在“com/huawei”目录下的“HelloWorld.java”文件中修改后的代码信息如下。

```
package com.huawei;
/**
 * Generate a unique number
 *
 */
public class HelloWorld
{
```


```
//用于打印日志
public void debugLog(List<String> msg) {
    for (String msg0 : msg) {
        System.out.println("DEBUG:" + msg0);
    }
}

public static void main( String[] args )
{
    System.out.println("Hello World!");
}
}
```

----结束

创建规则文件

步骤1 在[Visual Studio Code官网](#)下载并安装1.67.0以上版本的IDE编辑器。

步骤2 在VSCode IDE编辑器界面上，单击左侧按钮，在弹出的窗口中搜索“Huawei Cloud CodeNavi”。

步骤3 单击“Install”，安装代码检查插件。

步骤4 安装后，在VSCode IDE编辑器工作空间新建后缀为“.kirin”的文件，文件名为自定义规则的规则文件名称，例如：CheckDebugCode.kirin。文件内容如下所示。

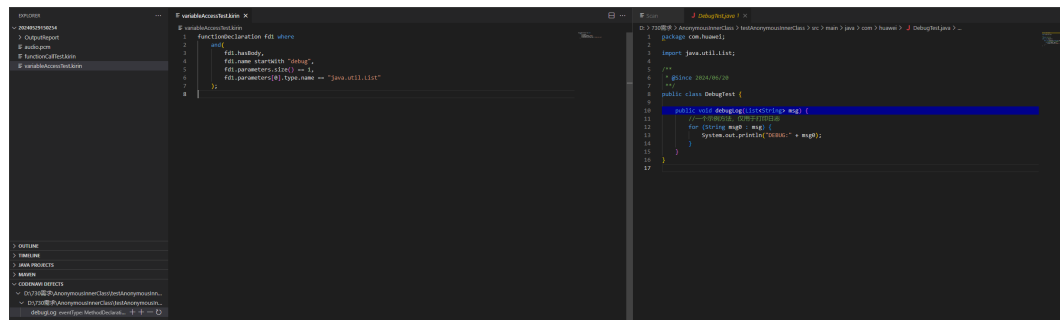
```
functionDeclaration fd1 where
    and(
        fd1.hasBody,
        fd1.name startWith "debug",
        fd1.parameters.size() == 1,
        fd1.parameters[0].type.name == "java.util.List"
    );
```

步骤5 在规则文件界面上鼠标右键，选择“CodeNavi > Format格式化”校验语法正确性。

步骤6 在规则文件界面上鼠标右键，选择“CodeNavi > Scan扫描”。

步骤7 在弹出的窗口中选择需要检查的文件或者目录，单击“Scan”。

步骤8 扫描完成后，单击界面左下角的告警显示具体的代码片段。同时，会在同目录的OutputReport文件中生成“.json”格式的规则文件。



----结束

创建自定义规则

步骤1 在页面导航栏中选择“代码 > 代码检查”。

步骤2 单击“规则”页签。

步骤3 单击“新建规则”，参考表4-2配置参数。

表 4-2 自定义规则参数说明

参数	说明
规则名称	创建的规则名称，可自定义。例如：检查调试代码。
工具规则名称	默认填充为规则源码文件名，不可手动修改。
检查工具	该检查规则使用的检查工具，当前仅支持SecBrella。
语言	该规则检查的编译语言，当前仅支持Java。
规则源码	该规则的源码文件。上传 创建规则文件 中生成的文件。
严重级别	该规则检查出的代码问题的严重级别，分为致命、严重、一般和提示。这里设置为“提示”。
标签	为该规则设置标签，便于使用时区分该规则使用场景。非必填，无需填写。 说明 添加多个标签时，需使用英文逗号分隔。
描述	对该规则的使用描述。代码内容支持使用Markdown格式呈现。字符数限制为0~10000。例如：检查是否存在调试代码。
正确示例	该规则对应的正确代码示例。代码内容支持使用Markdown格式呈现。字符数限制为0~10000。非必填，无需填写。
错误示例	不满足该规则的错误代码示例。代码内容支持使用Markdown格式呈现。字符数限制为0~10000。非必填，无需填写。
修复建议	对于该规则检查出的问题修改建议。代码内容支持使用Markdown格式呈现。字符数限制为0~10000。非必填，无需填写。

步骤4 单击“新建规则”，完成创建。

---结束

创建自定义规则集

步骤1 在代码检查任务列表页，单击“规则集”页签，进入规则集列表页面。

步骤2 单击“新建规则集”，在弹出的窗口设置“规则集名称”为“RuleList”，“检查语言”为“Java”。

步骤3 单击“确定”，进入到“规则集配置”页面。

步骤4 勾选[创建自定义规则](#)中创建的规则，单击右上角“保存”。

---结束

创建代码检查任务

步骤1 在代码检查任务列表页，单击“新建任务”，按照如下表格配置参数。

表 4-3 代码检查任务参数说明

参数	说明
归属项目	创建代码检查任务所属项目中创建的项目名称“Scrum01”。默认填写，无需配置。
代码源	选择需要检查的代码来源。选择“Repo”。
任务名称	代码检查任务名称，可自定义。例如：CheckTask01。
仓库	选择创建CodeArts Repo代码仓中创建的代码仓“Repo01”。
分支	保持默认“master”即可。
检查语言	选择“Java”。

步骤2 单击“新建任务”，完成代码检查任务的创建。

----结束

使用自定义规则集执行检查任务

步骤1 在代码检查任务列表页单击代码检查任务名称，进入代码检查任务详情页。

步骤2 单击“设置”。

步骤3 单击“规则集”，在右侧区域单击 选择创建自定义规则集中创建的规则集“RuleList”。

步骤4 单击右上角“开始检查”。

----结束

查看检查结果

步骤1 在代码检查页面任务列表中，搜索创建代码检查任务创建的任务名称“CheckTask01”。

步骤2 单击任务名称，查看代码检查详情，包括概览、代码问题、代码度量、检查日志等。

----结束

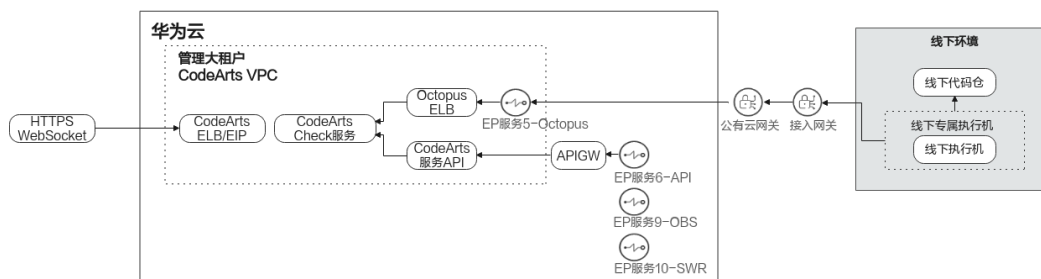
5 不上传代码到云服务的情况下使用代码检查服务

应用场景

部分用户的代码有严格的保密机制，不允许将代码上传到外部网络，用户希望仅在信任的自有执行机上完成代码扫描服务。华为云CodeArts Check为此提供本实践的扫描方案，在不上传代码到云服务的情况下使用代码检查服务检查代码问题。

方案架构

图 5-1 方案架构



约束限制

使用自定义执行机功能为受限功能，如需使用，请联系技术支持。

前提准备

- 已在私有网络创建自定义执行机。执行机规格为CentOS 7，推荐配置为CPU：8U，内存：32G，系统盘：100G，数据盘：250G。
- 配置执行机网络，确保执行机能够同时访问华为云服务和自建代码仓服务。
- 接入的自定义执行机中已安装Git-lfs，若未安装，可参考以下示例安装Git-lfs。本示例以使用命令的方式安装为例。

在执行机上执行以下命令。

```
# 下载
wget -O git-lfs.tar.gz https://github.com/git-lfs/git-lfs/releases/download/v3.4.1/git-lfs-linux-amd64-v3.4.1.tar.gz
# 解压
```

```
tar -zxvf git-lfs.tar.gz
# 进入解压后的目录
cd git-lfs-3.4.1
# 执行安装脚本
sh install.sh
# 验证
git lfs version
```

- 已[挂载云硬盘](#)。


操作流程

表 5-1 操作流程

流程	说明
新建项目	为本实践新建项目。
新建自定义执行机资源池	为本实践新建所需的自定义执行机资源池。
新建通用Git服务扩展点	创建服务扩展点，可访问用户使用的第三方代码仓。
创建并执行代码检查任务	创建代码检查任务并配置使用自定义执行机执行检查。通过该实践的方案进行的扫描，不会将代码上传至云服务。

新建项目

步骤1 使用华为云账号[登录华为云控制台页面](#)。

步骤2 单击页面左上角 ，在服务列表中选择“开发与运维 > 软件开发生产线 CodeArts”。

步骤3 单击“立即使用”，进入CodeArts服务首页。


步骤4 在首页单击“新建项目”，选用“Scrum”项目模板。

步骤5 项目名称填写“check-bestpractice”，其他保持默认即可。

步骤6 单击“确定”后，进入到“check-bestpractice”项目下。

----结束

新建自定义执行机资源池

步骤1 在导航栏中单击用户名 ，选择“租户设置”。

步骤2 选择“资源池管理 > 资源池”。

步骤3 单击“新建资源池”，在弹出的窗口中参考[表5-2](#)配置参数后，单击“保存”。

表 5-2 资源池配置参数说明

参数名称	参数说明
资源池名称	资源池的名称，根据需要自定义。例如：custom_pool。
资源池类型	选择LINUX_DOCKER。执行任务时将拉起一个Linux docker 容器，任务在容器中运行。
资源池描述	根据需要输入资源池描述。可不填写。
资源池可以被租户下所有子用户使用	勾选后，此资源池可以被当前租户下所有子用户使用。可不勾选。

步骤4 单击新建的资源池名称“custom_pool”，进入到资源池配置页面。

步骤5 单击“新建代理”，在弹出的窗口中，参考表5-3配置代理信息，其他参数项保持默认即可。

表 5-3 新建代理参数说明

参数	说明
自动安装 Git	关闭开关，参考“如何手动安装Git？”安装Git。
是否安装 Docker	勾选此项，安装Docker。
自动安装 Docker	关闭开关，单击“如何手动安装Docker？”安装Docker。
AK	参考 获取AK/SK 获取。
SK	参考 获取AK/SK 获取。
代理名称	自定义代理名称。例如：agent_test_custom。
代理工作空间	填写代理工作空间，需符合标准的linux目录格式。例如：/opt/agent_test_custom。

步骤6 勾选协议，依次单击“生成命令”和“复制命令”。单击“关闭”。

图 5-2 新建代理



步骤7 根据“步骤三”提示，在执行机上执行**步骤6**中复制的命令。

步骤8 在代理列表页面，单击“刷新列表”，后台自动同步信息后，代理列表中会增加一条代理执行机信息。代理执行机的代理别名为“agent_test_custom-mwlye1N1LG”。

图 5-3 代理执行机



----结束

新建通用 Git 服务扩展点

服务扩展点（Endpoint）是软件开发生产线的一种扩展插件，为软件开发生产线提供链接第三方代码仓的能力。

代码检查服务默认检查CodeArts Repo服务的代码质量，同时也可使用服务扩展点连接第三方代码仓库检查代码质量。

步骤1 在项目下的CodeArts Check服务页面的导航栏选择“设置 > 通用设置 > 服务扩展点管理”。

步骤2 单击“新建服务扩展点”，在下拉列表中选择“通用Git”。

步骤3 在弹框中配置以下信息，单击“确定”。

表 5-4 新建通用 Git 服务扩展点

参数	说明
连接名称	自定义，支持中文、英文、数字、“-”、“_”、“.”、空格，不超过256个字符。例如：Endpoint01。
Git仓库Url	输入待连接的Git仓库的https地址。
用户名	输入待连接的Git仓库的用户名，不超过300个字符。
密码或Access Token	输入待连接的Git仓库的密码，不超过300个字符。

----结束

创建并执行代码检查任务

步骤1 在页面导航栏中选择“代码 > 代码检查”。

步骤2 单击“新建任务”，进入“新建任务”页面。

- “代码源”选择“通用Git”。
- “任务名称”自定义填写“CheckTask01”。
- “Endpoint实例”选择[新建通用Git服务扩展点](#)中创建的服务扩展点“Endpoint01”。

其他参数保持默认即可。

步骤3 单击“新建任务”，完成检查任务的创建。

步骤4 在代码检查详情页面，选择“设置 > 自定义环境”。

步骤5 在“执行主机”区域，选择“自定义执行机”。

步骤6 在下拉框中选择[新建自定义执行机资源池](#)中创建的资源池“custom_pool”。

步骤7 配置完成后，单击“保存”，单击“开始检查”，等待检查完成即可。

----结束

6 使用自定义执行机执行代码检查任务

应用场景

当CodeArts Check提供的内置执行机不满足业务要求时，您可接入自行提供的执行机，通过注册的方式托管到CodeArts Check服务中，委托CodeArts Check服务进行调度并执行代码检查任务。本实践我们通过检查CodeArts Repo代码仓的代码问题来演示使用自定义执行机执行代码检查任务。

本实践需要依赖使用的其他服务：[代码托管服务](#)，用于存储实践中项目所使用的代码。

约束限制

- 使用自定义执行机功能为受限功能，如需使用，请联系技术支持。
- 需已具备CodeArts Repo服务的操作权限，具体操作可参考[授权使用CodeArts Repo服务](#)。

前提准备

- 已参考[自定义购买ECS](#)购买本实践使用的弹性云服务器。

📖 说明

代码检查服务接入的自定义执行机只支持EulerOS2.5操作系统。

- 接入的自定义执行机中已安装Git-lfs，若未安装，可参考以下示例安装Git-lfs。本示例以使用命令的方式安装为例。

在执行机上执行以下命令。

```
# 下载
wget -O git-lfs.tar.gz https://github.com/git-lfs/git-lfs/releases/download/v3.4.1/git-lfs-linux-amd64-v3.4.1.tar.gz
# 解压
tar -zxvf git-lfs.tar.gz
# 进入解压后的目录
cd git-lfs-3.4.1
# 执行安装脚本
sh install.sh
# 验证
git lfs version
```

- 已[挂载云硬盘](#)。


操作流程

表 6-1 操作流程

流程	说明
新建项目	为本实践新建项目。
新建自定义执行机资源池	为本实践新建所需的自定义执行机资源池。
新建CodeArts Repo代码仓	为本实践新建存储代码的代码仓。
配置并执行代码检查任务	配置代码检查任务使用自定义执行机执行检查。
查看代码检查结果	通过查看检查日志可验证本次代码检查任务使用自定义执行机执行。

新建项目

步骤1 使用华为云账号[登录华为云控制台页面](#)。

步骤2 单击页面左上角 ，在服务列表中选择“开发与运维 > 软件开发生产线 CodeArts”。

步骤3 单击“立即使用”，进入CodeArts服务首页。

步骤4 在首页单击“新建项目”，选用“Scrum”项目模板。

步骤5 项目名称填写“check-bestpractice”，其他保持默认即可。

步骤6 单击“确定”后，进入到“check-bestpractice”项目下。

----结束

新建自定义执行机资源池

步骤1 在导航栏中单击用户名 ，选择“租户设置”。

步骤2 选择“资源池管理 > 资源池”。

步骤3 单击“新建资源池”，在弹出的窗口中参考[表6-2](#)配置参数后，单击“保存”。

表 6-2 资源池配置参数说明

参数名称	参数说明
资源池名称	资源池的名称，根据需要自定义。例如：custom_pool。

参数名称	参数说明
资源池类型	选择LINUX_DOCKER。执行任务时将拉起一个Linux docker容器，任务在容器中运行。
资源池描述	根据需要输入资源池描述。可不填写。
资源池可以被租户下所有子用户使用	勾选后，此资源池可以被当前租户下所有子用户使用。可不勾选。

步骤4 单击新建的资源池名称“custom_pool”，进入到资源池配置页面。

步骤5 单击“新建代理”，在弹出的窗口中，参考表6-3配置代理信息，其他参数项保持默认即可。

表 6-3 新建代理参数说明

参数	说明
是否安装 Docker	勾选此项，配需安装Docker。
自动安装 Docker	打开开关，自动安装Docker。
AK	参考 获取AK/SK 获取。
SK	参考 获取AK/SK 获取。
代理名称	自定义代理名称。例如：agent_test_custom。
代理工作空间	填写代理工作空间，需符合标准的linux目录格式。例如：/opt/agent_test_custom。

步骤6 勾选协议，依次单击“生成命令”和“复制命令”。单击“关闭”。

图 6-1 新建代理



步骤7 根据“步骤三”提示，在弹性云服务器列表页，单击前提准备中购买的服务器所在行的“远程登录”按钮，执行步骤6中复制的命令。

步骤8 在代理列表页面，单击“刷新列表”，后台自动同步信息后，代理列表中会增加一条代理执行机信息。代理执行机的代理别名为“agent_test_custom-mwlye1NLLG”。

图 6-2 代理执行机



----结束

新建 CodeArts Repo 代码仓

步骤1 在导航栏中选择“服务 > 代码托管”，进入代码托管服务首页。

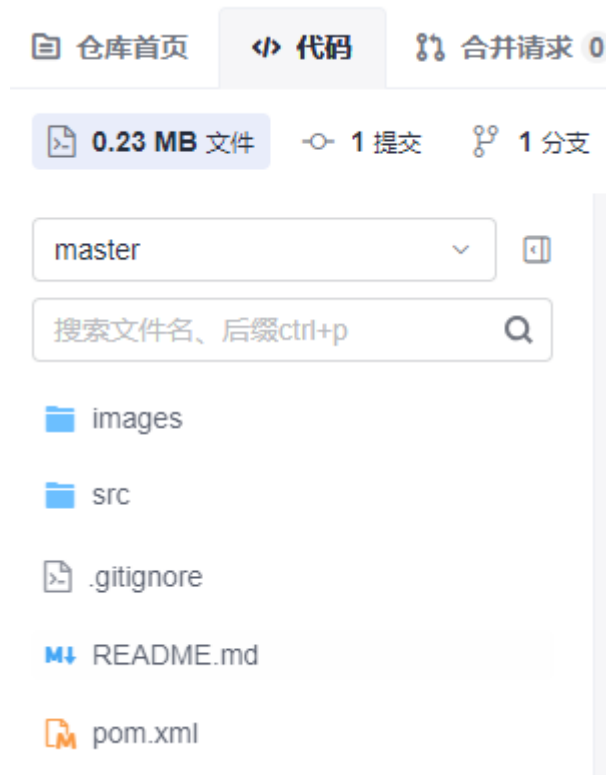
步骤2 单击“新建仓库”，在新建仓库页面“归属项目”选择新建项目中创建的项目名称，“仓库类型”选择“模板仓库”，单击“下一步”。

步骤3 选择“Java Maven Demo”仓库模板，单击“下一步”。

步骤4 “代码仓库名称”填写为“custom_repo”，勾选自动创建代码检查任务前的复选框，其他参数保持默认即可。单击“确定”，完成代码仓的创建。

创建完成后的代码仓文件目录如图6-3所示。

图 6-3 代码仓文件目录



---结束

配置并执行代码检查任务

- 步骤1** 在页面导航栏中选择“代码 > 代码检查”。由于在创建代码仓时，已选择自动创建代码检查任务，因此在代码检查任务列表页，已展示对应的代码检查任务。
- 步骤2** 单击任务名称，进入代码检查详情页面，选择“设置 > 自定义环境”。
- 步骤3** 在“执行主机”区域，选择“自定义执行机”。
- 步骤4** 在下拉框中选择新建自定义执行机资源池中创建的资源池“custom_pool”。
- 步骤5** 配置完成后，单击“保存”，单击“开始检查”。

---结束

查看代码检查结果

单击任务名称，进入代码检查详情页面，单击“检查日志”页签。若日志信息中出现“Find available executor node:agent_test_custom-mwlye1NILG”，表示本次代码检查任务由自定义执行机执行。其中“agent_test_custom-mwlye1NILG”为步骤8中代理执行机的代理别名。

概览 代码问题 代码度量 **检查日志** 设置

2024/08/24 15:26:25 GMT+08:00 检查成功 检查时间: 46m4s 总时间: 46m7s

执行历史

步骤日志 检查参数

检查步骤

- init ✓ <1s
- [-] check ✓ 46m2s
 - [-] uccp_compile(farman... ✓ 45m59s
 - check ✓ 44m54s**
 - report ✓ 1m5s
- report ✓ <1s

日志

170	[2024/08/24 16:11:21.408 GMT
171	[2024/08/24 16:11:21.411 GMT
172	[2024/08/24 16:11:21.418 GMT
173	[2024/08/24 16:11:22.188 GMT
174	[2024/08/24 16:11:22.192 GMT
175	[2024/08/24 16:11:22.327 GMT
176	[2024/08/24 16:11:22.328 GMT
177	[2024/08/24 16:11:22.329 GMT
178	[2024/08/24 16:11:22.410 GMT

7 CodeArts Check 通过调用 API 执行 MR 增量检查

应用场景

在用户实际的使用场景中，用户往往有构建自己的代码质量管控平台，集成代码检查及其他检查项的需求。并且在用户的应用场景中，MR合入过程中的质量监控是一个高频应用场景。

本示例介绍如何通过调用CodeArtsCheck API方式，执行MR增量代码检查并获取检查结果。通过调用CodeArtsCheck API可以不使用华为云前端，从而构建自己的全流程代码质量管控平台，同时也可以实现对非Codearts Repo代码仓的门禁级检查。本示例将以通用Git为例，进行方案介绍。

约束限制

- 每个通用Git扩展点仅能对接单个代码仓地址，同时1个CodeArts项目下的扩展点数量限制为1000。
- 示例仅介绍如何调用CodeArtsCheck API以实现执行MR增量代码检查并获取检查结果的功能，具体集成方式请用户根据实际业务场景选择。

前提准备

已[开通并授权使用代码检查服务](#)授权使用代码检查服务。

操作流程


表 7-1 操作流程

流程	说明
创建代码检查任务所属项目	为本实践新建项目。
新建通用Git服务扩展点	本实践检查第三方代码仓的代码质量，代码检查服务需通过CodeArts提供的扩展插件连接到第三方代码仓。
获取服务扩展点ID	获取服务扩展点ID，在使用API时需要填写。

流程	说明
创建MR检查任务并获取任务ID	创建代码检查任务并获取任务ID。
执行并查询MR任务状态	通过执行脚本，执行代码检查任务。
查询缺陷详情	通过API查询缺陷详情。

创建代码检查任务所属项目

步骤1 使用华为云账号[登录华为云控制台页面](#)。

步骤2 单击页面左上角 ，在服务列表中选择“开发与运维 > 软件开发生产线 CodeArts”。

步骤3 单击“立即使用”，进入CodeArts服务首页。

步骤4 在首页单击“新建项目”，选用“Scrum”项目模板。项目名称填写“Scrum01”，其他参数保持默认即可。

步骤5 单击“确定”后，进入到“Scrum01”项目下。

----结束

新建通用 Git 服务扩展点

服务扩展点（Endpoint）是软件开发生产线的一种扩展插件，为软件开发生产线提供链接第三方代码仓的能力。

代码检查服务默认检查CodeArts Repo服务的代码质量，同时也可使用服务扩展点连接第三方代码仓库检查代码质量。

步骤1 在项目下的CodeArts Check服务页面的导航栏选择“设置 > 通用设置 > 服务扩展点管理”。

步骤2 单击“新建服务扩展点”，在下拉列表中选择“通用Git”。

步骤3 在弹框中配置以下信息，单击“确定”。

表 7-2 新建通用 Git 服务扩展点

参数	说明
连接名称	自定义，支持中文、英文、数字、“-”、“_”、“.”、空格，不超过256个字符。例如：Endpoint01。
Git仓库Url	输入待连接的Git仓库的https地址。
用户名	输入待连接的Git仓库的用户名，不超过300个字符。

参数	说明
密码或 Access Token	输入待连接的Git仓库的密码，不超过300个字符。

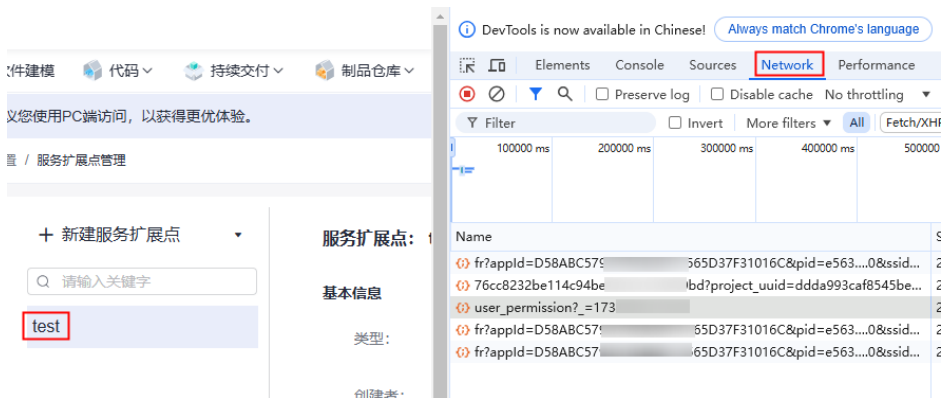
----结束

获取服务扩展点 ID

步骤1 在服务扩展点详情页，按键盘“F12”，选择“Network”页签。

步骤2 单击扩展点名称，其中“user_permission”为服务扩展点ID。

图 7-1 服务扩展点 ID



----结束

创建 MR 检查任务并获取任务 ID

步骤1 以下为新建全量任务的请求示例。

增量检查任务依赖于全量任务，所以需要先创建全量任务。

CreateTask: POST /v2/{project_id}/task

```
{
  "git_url": "https://codehub-dg-g.huawei.com/userID/CloudBuildTest01.git", #如果使用CodeArts Repo代码
  "git_branch": "master",
  "language": [
    "JAVA"
  ],
  "task_type": "full", #任务类型
  "username": "zxxxw", #用户名, 仅在三方仓场景下使用
  "access_token": "f2xxx0", #密码或用户token, 仅在三方仓场景下使用
  "endpoint_id": "fxxxxsz" #服务扩展点ID
}
```

步骤2 新建全量任务的返回结果。响应值为全量任务的任务ID，即创建增量任务需要用到的“parent_task_id”。

响应结果

响应头	响应体
1	{
2	"task_id": "[redacted]"
3	}

创建全量任务后，可在华为云对应局点账号下的项目中，查看到新建的代码检查任务。

步骤3 新建MR增量代码检查任务的请求示例。

```
CreateTask: /v2/{project_id}/task
{
  "git_url": "https://codehub-dg-g.huawei.com/userID/CloudBuildTest01.git", #如果使用CodeArts Repo代码
  #仓库, 需具备Repo仓库的相关权限且需填写ssh地址
  "git_branch": "master",
  "language": [
    "JAVA"
  ],
  "task_type": "inc", #任务类型
  "inc_config": {
    "parent_task_id": "8cxxx94", #全量任务id, 增量任务挂载在全量任务下
    "git_source_branch": "dev003", #源分支
    "git_target_branch": "dev002", #目标分支
    "merge_id": "4", #MR编号, 用于检查插件拉取增量代码
    "event_type": "merge_request", #事件类型, MR请求
    "action": "open", #状态, open为MR开启状态
    "title": "测试" #任务标题, 根据需求填写
  }
}
```

步骤4 新建增量任务的返回结果。

响应结果

响应头	响应体
1	{
2	"task_id": "[redacted]"
3	}

响应值为增量任务的ID，该ID在任务列表界面不可见，仅用于通过API接口创建、执行和查询MR任务的场景。

----结束

执行并查询 MR 任务状态

步骤1 执行MR任务的请求示例。

```
RunTask: POST /v2/tasks/{task_id}/run
{
}
```

- “task_id”中填写[创建MR检查任务并获取任务ID](#)中获取的MR任务ID以执行检查任务。

- 请求体参数，空传即可。

步骤2 执行MR任务的返回结果。

响应结果

响应头	响应体
1	{
2	"exec_id": "9XXXXXXXXXXXXXXXXXXXXd"
3	}

响应值为MR任务对应的执行ID，也就是1个任务1次执行产生的1个job的ID。

步骤3 轮询任务执行状态请求示例。

ShowProgressDetail: GET /v2/tasks/{task_id}/progress

“task_id”中填入MR任务ID以查询任务执行状态，任务状态0表示检查中，1表示检查失败，2表示检查成功，3表示任务中止。

步骤4 轮询任务执行状态的返回结果。

响应结果

响应头	响应体
1	{
2	"task_status": 2,
3	"progress": {
4	"ratio": "100%",
5	"info": "success"
6	}
7	}

----结束

查询缺陷详情

步骤1 查询缺陷详情请求示例。

ShowTaskDefects: GET /v2/tasks/{task_id}/defects-detail

“task_id”中填入MR任务ID以查询缺陷详情，也可以通过其他接口查询概要等信息，更多使用方法参考[API接口说明文档手册](#)。

步骤2 查询缺陷详情的响应结果。

响应结果

响应头	响应体
1	{
2	"defects": [
3	{
4	"extendInfo": "",
5	"defect_id": "18[REDACTED]2d8",
6	"line_number": "9",
7	"defect_checker_name": "G.LOG.01 记录日志应该使用Facade模式的日志框架",
8	"defect_content": "Do not use System.out or System.err to log messages",
9	"defect_level": "2",
10	"file_path": "src/main/java/HelloWorld.java",
11	"created_at": "2024-11-14T09:43:30Z",
12	"defect_status": "0",
13	"defect_owner": "dev[REDACTED]01",
14	"rule_system_tags": "fixbotengine-java",
15	"rule_name": "G.LOG.01 记录日志应该使用Facade模式的日志框架",
16	"fragment": [
17	{
18	"line_num": "4",
19	"line_content": " */",
20	"start_offset": -1,

----结束

相关参考

- 本示例中所有接口均可在[华为云OpenAPI调试平台](#)进行对应调试。
- 新建代码检查任务的API接口说明可参考[新建检查任务](#)。
- 执行代码检查任务的API接口说明可参考[执行检查任务](#)。
- 查询任务执行状态的API接口说明可参考[查询任务执行状态](#)。

8 使用 Jenkins 插件集成 CodeArts Check 执行代码检查

应用场景

Jenkins是一款由Java编写的开源的持续集成工具，它运行在Servlet容器中（例如Apache Tomcat），可以通过各种方式触发构建。例如提交给版本控制系统时被触发、通过类似Cron的机制调度、或在其他的构建已经完成时通过一个特定的URL进行请求。

利用Jenkins工具现有能力，实现代码检查服务的快速集成，包含执行检查、状态查询、报告输出等基本扫描流程。用户可根据基础方案进行自定义开发。

本文介绍如何通过Jenkins工具及CodeArtsCheck API，实现代码检查服务集成与调度。本示例将以通用Git为例，进行方案介绍。

约束限制

- 代码源为非同一CodeArts项目下的Repo仓时需使用扩展点进行代码源接入。
- 每个通用Git扩展点仅能对接单个代码仓地址。
- 1个CodeArts项目下的扩展点数量限制为1000。
- 实行本实践需了解Jenkins工具的使用与基础开发。

前提准备

已[开通并授权使用代码检查服务](#)。

操作流程


表 8-1 操作流程

流程	说明
创建代码检查任务所属项目	为本实践新建项目。
新建通用Git服务扩展点	实践检查第三方代码仓的代码质量，代码检查服务需通过CodeArts提供的扩展插件连接到第三方代码仓。

流程	说明
创建通用Git代码检查任务并获取任务ID	创建代码检查任务并获取任务ID。
创建访问凭证AK和SK	创建调用API时使用的AK和SK。
下载KooCLI工具并获取代码检查API的调用命令	下载KooCLI工具并获取代码检查API的调用命令。
在Jenkins工具中调用代码检查API	通过Jenkins工具调用代码检查API。

创建代码检查任务所属项目

步骤1 使用华为云账号[登录华为云控制台页面](#)。

步骤2 单击页面左上角 ，在服务列表中选择“开发与运维 > 软件开发生产线 CodeArts”。

步骤3 单击“立即使用”，进入CodeArts服务首页。

步骤4 在首页单击“新建项目”，选用“Scrum”项目模板。项目名称填写“Scrum01”，其他参数保持默认即可。

步骤5 单击“确定”后，进入到“Scrum01”项目下。

----结束

新建通用 Git 服务扩展点

服务扩展点（Endpoint）是软件开发生产线的一种扩展插件，为软件开发生产线提供链接第三方代码仓的能力。

代码检查服务默认检查CodeArts Repo服务的代码质量，同时也可使用服务扩展点连接第三方代码仓库检查代码质量。

步骤1 在项目下的CodeArts Check服务页面的导航栏选择“设置 > 通用设置 > 服务扩展点管理”。

步骤2 单击“新建服务扩展点”，在下拉列表中选择“通用Git”。

步骤3 在弹框中配置以下信息，单击“确定”。

表 8-2 新建通用 Git 服务扩展点

参数	说明
连接名称	自定义，支持中文、英文、数字、“-”、“_”、“.”、空格，不超过256个字符。例如：Endpoint01。

参数	说明
Git仓库Url	输入待连接的Git仓库的https地址。
用户名	输入待连接的Git仓库的用户名，不超过300个字符。
密码或 Access Token	输入待连接的Git仓库的密码，不超过300个字符。

----结束

创建通用 Git 代码检查任务并获取任务 ID

步骤1 在导航栏选择“代码 > 代码检查”。

步骤2 单击“新建任务”，进入“新建任务”页面，参考表8-3配置参数。

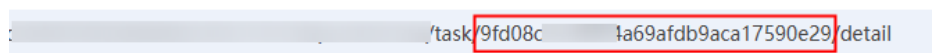
表 8-3 通用 Git 代码检查任务参数说明

参数	说明
归属项目	任务所属项目。默认填写为创建代码检查任务所属项目中的项目名称“Scrum01”，无需手动填写。
代码源	选择“通用Git”。
任务名称	代码检查任务名称，可自定义。例如：CheckTask01。
Endpoint实例	选择新建通用Git服务扩展点中创建的服务扩展点“Endpoint01”。
仓库	保持默认即可。
分支	保持默认“master”即可。
检查语言	选择需要检查的代码语言，选择“Java”。

步骤3 单击“新建任务”，完成检查任务的创建。

步骤4 在代码检查任务列表页，单击代码检查任务名称，在代码检查任务概览页面URL中即为任务ID。

图 8-1 代码检查任务 ID



----结束

创建访问凭证 AK 和 SK

- 步骤1 登录管理控制台。
- 步骤2 单击用户名，在下拉列表中单击“我的凭证”。
- 步骤3 单击“访问密钥”。
- 步骤4 单击“新增访问密钥”，弹出“新增访问密钥”页面。
- 步骤5 输入该访问密钥的描述信息。
- 步骤6 单击“确定”，下载访问密钥。

说明

为防止访问密钥泄露，建议您将其保存到安全的位置。

----结束

下载 KooCLI 工具并获取代码检查 API 的调用命令

- 步骤1 访问[华为云命令行工具服务KooCLI](#)文档页面。
- 步骤2 根据本地电脑的系统版本号下载对应版本的KooCLI工具。
- 步骤3 访问[代码检查API调试](#)页面。
- 步骤4 选择实际使用的Region和填写[创建通用Git代码检查任务并获取任务ID](#)中查询到的任务ID，其中`hcloud CodeCheck RunTask --cli-region="cn-north-4" --task_id="9fd08c41a9824a69afdb9aca17590e29"`即为生成的调用命令。

图 8-2 生成 API 调用命令



- 步骤5 参考步骤4，依次获得查询任务执行状态（ShowProgressDetail）、查询缺陷概要（ShowTaskDetail）API的调用命令。

----结束

在 Jenkins 工具中调用代码检查 API

- 步骤1 编写流水线脚本，代码示例如下。

```
pipeline {
    agent any
    stages {
```

```

stage('代码检查') {
  steps {
    script {
      echo "设置AK/SK"
      def response = sh(
        returnStdout: true,
        script: """
/codearts/hcloud configure set --cli-access-key=S*****3 --cli-secret-
key=s*****A
        """)
      echo response
    }
    script {
      echo "执行检查任务"
      def response = sh(
        returnStdout: true,
        script: """
/codearts/hcloud CodeCheck RunTask --cli-region="cn-south-1" --
task_id="4*****f"
        """)
      echo response
      def json = readJSON text:response
      echo "exec_id: ${json.exec_id}"
    }
    script {
      def try_count=0
      def task_status=0
      while (true) {
        def response = sh(
          returnStdout: true,
          script: """
/codearts/hcloud CodeCheck ShowProgressDetail --cli-region="cn-south-1" --
task_id="4*****f"
          """)
        echo response
        def json = readJSON text:response
        task_status=json.task_status
        echo "代码检查任务执行中 try_count: $try_count, task_status: $task_status"
        if(task_status.toInteger()>0){
          break
        }
        try_count=try_count+1
        if(try_count==120){
          break
        }
        sleep(time: 5, unit: 'SECONDS')
      }
      if(task_status.toInteger()==0){
        error "代码检查超过600s, 构建任务失败"
      }
      if(task_status.toInteger()==1){
        error "代码检查没有通过, 构建任务失败"
      }
      if(task_status.toInteger()==3){
        error "代码检查任务中止, 构建任务失败"
      }
    }
    script {
      def response = sh(
        returnStdout: true,
        script: """
/codearts/hcloud CodeCheck ShowTaskDetail --cli-region="cn-south-1" --
task_id="4*****f"
        """)
      echo response
    }
  }
}

```

```
def json = readJSON text:response
def issue_count=json.issue_count
def new_count=json.new_count
def solve_count=json.solve_count
def cyclomatic_complexity_per_method=json.cyclomatic_complexity_per_method
def duplication_ratio=json.duplication_ratio
def code_line=json.code_line
def critical_count=json.critical_count
def major_count=json.major_count
def is_access=json.is_access
echo "问题数:$issue_count"
echo "未解决问题数:$new_count"
echo "已解决问题数:$solve_count"
echo "代码平均圈复杂度:$cyclomatic_complexity_per_method"
echo "代码重复率:$duplication_ratio"
echo "NBNC代码行:$code_line"
echo "致命问题数:$critical_count"
echo "严重问题数:$major_count"
echo "门禁质量是否通过:$is_access"
if(is_access.toInteger()==1){
    error "门禁质量检查没有通过，构建任务失败"
}
else{
    echo "门禁质量检查通过"
}
}
}
stage('拉取代码') {
    steps {
        sh '''
            echo "拉取代码"
            '''
    }
}
stage('代码编译') {
    steps {
        sh '''
            echo "代码编译"
            '''
    }
}
stage('上传镜像并部署到cce') {
    steps{
        sh '''
            echo "上传镜像并部署到cce"
            '''
    }
}
}
```

步骤2 执行流水线任务。

----结束

9 HE2E DevOps 实践：代码检查

本文以“DevOps全流程示例项目”为例，介绍如何在项目中完成代码检查配置。
开展实践前，需要完成[创建项目](#)和[创建仓库](#)。

预置任务简介

样例项目中预置了以下4个代码检查任务。

表 9-1 预置任务

预置任务	任务说明
phoenix-codecheck-worker	检查Worker功能对应代码的任务。
phoenix-codecheck-result	检查Result功能对应代码的任务。
phoenix-codecheck-vote	检查Vote功能对应代码的任务。
phoenix-sample-javas	检查整个代码仓库对应的JavaScript代码的任务。


本章节以任务“phoenix-codecheck-worker”为例进行讲解。

配置并执行任务

开发人员可以对样例项目中预置的任务做一些简单的配置，增加Python语言检查规则集，使检查更全面。


步骤1 编辑任务。

1. 进入“凤凰商城”项目，单击导航“代码 > 代码检查”，页面中显示样例项目内置的4个任务。
2. 在列表中找到任务“phoenix-codecheck-worker”。
3. 单击任务名称进入详情页，选择“设置”页签。

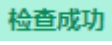
4. 单击导航“规则集”，规则集中默认包含的语言是“JAVA”。
5. 增加Python语言检查规则集。
 - a. 单击“已包含语言”之后的图标，重新获取代码仓库语言，刷新后的列表新增了多种语言。

说明

如果页面中已显示“PYTHON”，则忽略此步骤。

- b. 将PYTHON语言对应的开关打开。
- c. 在弹框中单击“确定”。

步骤2 执行任务。

1. 单击“开始检查”，启动任务。
2. 当页面显示 ，表示任务执行成功。
如果任务执行失败，请根据页面弹出报错提示排查修改。

----结束

查看检查结果

代码检查服务提供检查结果统计，并对检查出的问题提供修改建议，可以根据修改建议优化项目代码。

步骤1 在代码检查任务中，选择“概览”页签，即可查看任务执行结果统计。

步骤2 单击“代码问题”页签，即可看到问题列表。

单击问题框中的“问题帮助”，可以查看系统对此问题的修改建议。可以根据需要在代码仓库中找到对应文件及代码位置，参考修改建议优化代码。

----结束