代码检查

最佳实践

文档版本	01
发布日期	2025-12-03

华为技术有限公司

商标声明


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

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	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。
5 不上传代码到云服务的情况下使用代码检查服务	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。

最佳实践	说明
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。
7 CodeArts Check通过调用API执行MR增量检查	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。
8 使用Jenkins插件集成CodeArts Check执行代码检查	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。
11 HE2E DevOps实践之代码检查	本文以“DevOps全流程示例项目”为例，介绍如何在项目中进完成代码检查配置。

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**。
- 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 > 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。支持中文、英文、数字、“-”、“_”、“.”、空格，不超过256个字符，例如：GitCode。
Token	填写GitCode上获取的Token，获取方法如下： 1. 登录GitCode。 2. 单击页面右上角账号名称，选择“个人设置”。 3. 单击 “+访问令牌”，填写令牌名称以及到期时间。 4. 单击“新建访问令牌”，生成“你的个人访问令牌”，即Token。 5. 单击，即可复制生成的Token。 **说明** – 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

相关操作

- 如果需要对代码检查任务进行更多配置，可参考**配置代码检查任务**。
- 执行代码检查任务时常见问题请参考**代码检查常见问题**。

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**。
- 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	**查看通用Git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 > 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“CheckTask01”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

查看通用 Git 代码检查详情

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。

----结束

相关操作

- 如果需要对代码检查任务进行更多配置，可参考**配置代码检查任务**。
- 执行代码检查任务时常见问题请参考**代码检查常见问题**。

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**。
- 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 > 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”。

步骤3 在新建仓库页面，选择“按模板新建”。

步骤4 单击“下一步”，搜索并选择“Java Maven Demo”模板。

步骤5 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6 单击“确定”完成仓库创建。

步骤7 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。

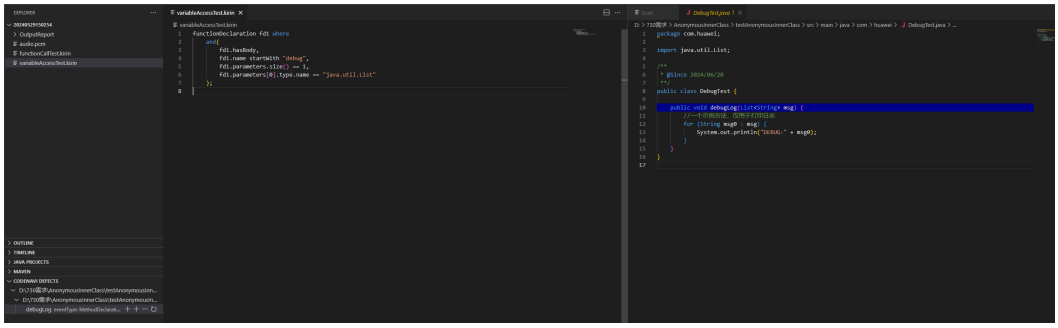
```
functionDeclaration fd1 where
    and(
        fd1.hasBody,
        fd1.name startWith "debug",
        fd1.parameters.size() == 1,
        fd1.parameters[0].type.name == "java.util.List"
    );
```

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 > 新建自定义规则”，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，当前支持Java和ARKTS。
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“确定”，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“确定”，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“编译配置”，将“编译工具选项”开关设置为 状态，“编译工具”选择“maven”。其他参数保持默认即可，然后单击“确定”。

图 4-1 编译配置

步骤5 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

相关操作

- 如果需要对代码检查任务进行更多配置，可参考**配置代码检查任务**。
- 执行代码检查任务时常见问题请参考**代码检查常见问题**。

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。

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及以上，推荐配置如下：
  - CPU：8U，内存：32G，系统盘：100G，数据盘：250G。（使用该执行机规格时，**需要**执行**挂载云硬盘**操作。）
  - CPU：8U，内存：32G，系统盘：250G。（使用该执行机规格时，**无需**执行**挂载云硬盘**操作。）
- 需确保本地执行机yum源可用，详情可参考**使用自动化工具配置华为云镜像源**。
- 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
```


操作流程

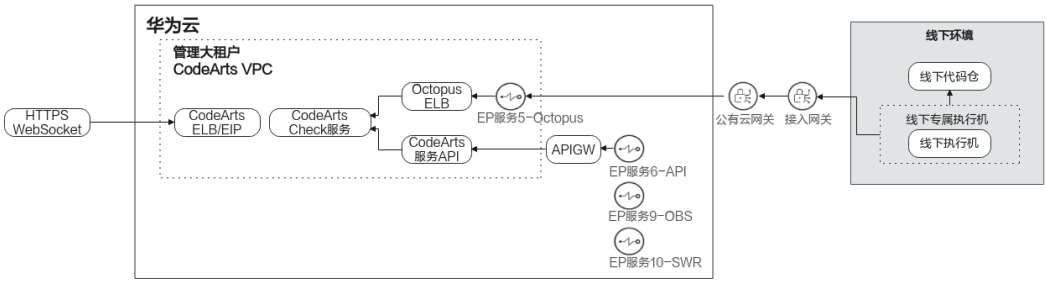
表 5-1 操作流程

流程	说明
配置VPN连接云下数据中心与云上VPC（可选）	可选操作。本地网络无法访问公网时需要配置。 配置本地网络可以访问云上VPC。
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。

配置 VPN 连接云下数据中心与云上 VPC（可选操作）

本地网络无法访问公网时需要配置。

图 5-1 方案架构



步骤1 参考**创建虚拟私有云和子网**创建“华北-北京四”的VPC。

步骤2 参考**创建VPN网关**，购买并创建“华北-北京四”的VPN网关。

购买后可在**虚拟专用网络VPN页**中查看到已创建的VPN网关，如下图所示。

图 5-2 VPN 网关

步骤3 参考**创建VPN连接**，购买并创建“华北-北京四”的VPN连接。

购买后可在**虚拟专用网络VPN页**中查看到已创建的VPN连接，如下图所示。

图 5-3 VPN 连接

其中“本端网关”和“本端子网”填写的是**步骤2**中的网关和子网，“远端网关”和“远端子网”填写的是本地网络的网关和子网。

步骤4 参考**创建VPN连接**，购买并创建本地网络的VPN连接。

其中“本端网关”和“本端子网”填写的是本地网络的网关和子网，“远端网关”和“远端子网”填写的是**步骤2**中华为云的网关和子网。

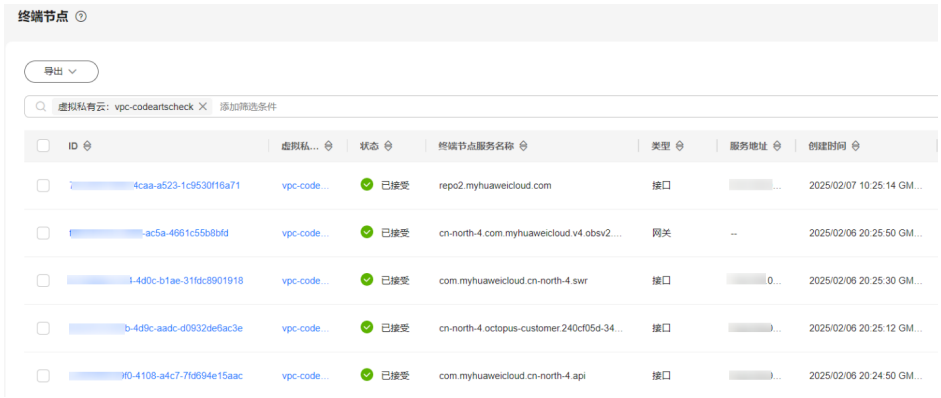
步骤5 在**终端节点控制台页面**，按照**表5-2**购买“华北-北京四”的VPC终端节点VPCEP。其他参数保持默认即可。

表 5-2 终端节点购买信息

区域	华北-北京四
服务类别	按名称查找服务
服务名称	依次购买以下节点。 • 如果是华北-北京四： – APIG：com.myhuaweicloud.cn-north-4.api – Octopus：cn-north-4.octopus-customer.240cf05d-34cb-48a7-a8a3-443cd496de27 – SWR：com.myhuaweicloud.cn-north-4.swr – OBS：cn-north-4.com.myhuaweicloud.v4.obsv2.OBSCluster9 – Mirror：repo2.myhuaweicloud.com • 如果是广州： – APIG：com.myhuaweicloud.cn-south-1.api – Octopus：cn-south-1.octopus-customer.3da50262-7b47-45c0-b26c-9e19dc155f5d – SWR：swr.cn-south-1.myhuaweicloud.com – OBS：cn-south-1.com.myhuaweicloud.v4.obsv2 – Mirror：repo.myhuaweicloud.com
虚拟私有云	选择**步骤1**中购买的VPC。

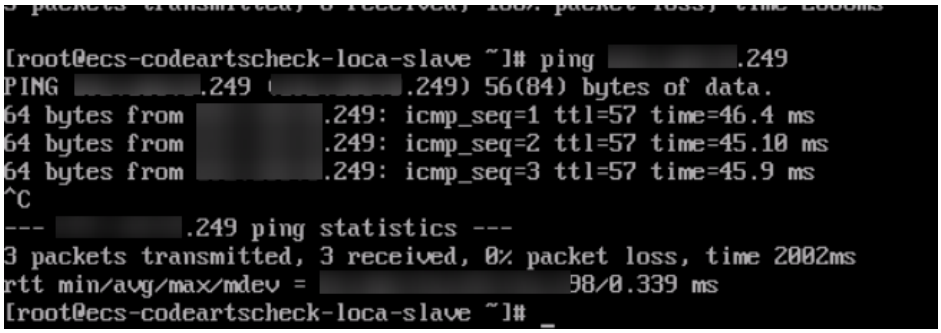
购买后可在**终端节点列表页**查看已购买的终端节点，如下图所示。

图 5-4 终端节点



步骤6 在本地的执行机中执行ping命令，对**步骤5**中购买的任意一个终端节点IP进行连通性测试。出现以下相似结果，表示本地执行机可以访问终端节点。

图 5-5 连通性测试结果



步骤7 在用户本地执行机上执行以下命令，确认OBS的域名解析IP。

确认“华北-北京四”的OBS：

```
dig secdev-codecheck-cn-north-4-01.obs.cn-north-4.myhuaweicloud.com
dig op-svc-swr-b051-10-38-19-62-3az.obs.cn-north-4.myhuaweicloud.com
```

确认“广州”的OBS：

```
dig cn-south-1-obs-codecheck-v2.obs.cn-south-1.myhuaweicloud.com
dig op-svc-swr-b051-10-230-33-197-3az.obs.cn-south-1.myhuaweicloud.com
```

说明

如果执行机未安装dig工具，可执行**yum install bind-utils**命令进行安装。

图 5-6 “华北-北京四”域名解析结果一

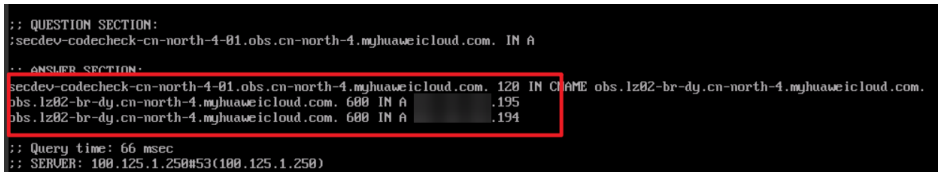
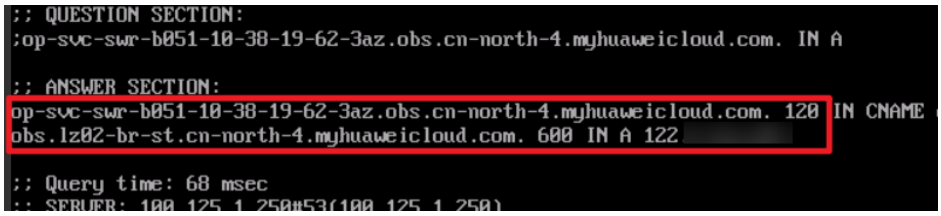


图 5-7 “华北-北京四”域名解析结果二



如图5-6和图5-7所示，解析到“华北-北京四”的OBS域名IP为“x.x.x194/31”和“122.x.x.x/31”网段。

步骤8 分别在步骤3VPN连接的“本端子网”和步骤4VPN连接的“远端子网”中添加步骤7的网段。

图 5-8 修改 VPN 连接

图 5-9 添加远端子网

步骤9 在本地网络配置DNS域名解析。

- 方式一：（本地已安装DNS服务器）
  参考创建内网域名（需要创建内网域名和添加记录集），创建表5-3的域名。

表 5-3 DNS 域名

服务	华北-北京四域名	广州域名	IP
SWR	swr.cn-north-4.myhuaweicloud.com	swr.cn-south-1.myhuaweicloud.com	步骤5中SWR对应的服务地址
Octopus	agent.codearts.cn-north-4.myhuaweicloud.com	agent.codearts.cn-south-1.myhuaweicloud.com	步骤5中Octopus对应的服务地址
APIG	cloudoctopus.cn-north-4.myhuaweicloud.com	cloudoctopus.cn-south-1.myhuaweicloud.com	步骤5中APIG对应的服务地址

服务	华北-北京四域名	广州域名	IP
Mirror	mirrors.huaweicloud.com	mirrors.huaweicloud.com	步骤5中Mirror对应的服务地址

- 方式二：（本地未安装DNS服务器）

  在本地执行机的“/etc/hosts/”文件中增加以下配置。

```
${SWR对应的VPCEP IP} swr.cn-north-4.myhuaweicloud.com
${Octopus对应的VPCEP IP} agent.codearts.cn-north-4.myhuaweicloud.com
${APIG对应的VPCEP IP} cloudoctopus.cn-north-4.myhuaweicloud.com
${Mirror对应的VPCEP IP} mirrors.huaweicloud.com
```

步骤10 在本地的执行机中执行以下命令，进行连通性测试。

“华北-北京四”的OBS：

```
ping swr.cn-north-4.myhuaweicloud.com
ping agent.codearts.cn-north-4.myhuaweicloud.com
ping cloudoctopus.cn-north-4.myhuaweicloud.com
ping mirrors.huaweicloud.com
ping secdev-codecheck-cn-north-4-01.obs.cn-north-4.myhuaweicloud.com
ping cloud-octopus-agent.obs.cn-north-4.myhuaweicloud.com
ping op-svc-swr-b051-10-38-19-62-3az.obs.cn-north-4.myhuaweicloud.com
```

“广州”的OBS：

```
ping swr.cn-south-1.myhuaweicloud.com
ping agent.codearts.cn-south-1.myhuaweicloud.com
ping cloudoctopus.cn-south-1.myhuaweicloud.com
ping mirrors.huaweicloud.com
ping cn-south-1-obs-codecheck-v2.obs.cn-south-1.myhuaweicloud.com
ping cloud-octopus-agent.obs.cn-south-1.myhuaweicloud.com
ping op-svc-swr-b051-10-230-33-197-3az.obs.cn-south-1.myhuaweicloud.com
```

步骤11 参考**挂载云硬盘**。其中挂载目录为“/devcloud”。

----结束

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 > 代码检查 CodeArts Check”。

步骤3 单击“前往代码检查”，进入CodeArts Check服务首页。

步骤4 在导航栏切换到“首页”页签，依次单击“新建 > 新建项目”，选用“Scrum”项目模板。

步骤5 填写项目名称，例如“check-bestpractice”，其他参数保持默认即可。

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5-4**配置参数后，单击“保存”。

表 5-4 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5-5**配置代理信息，其他参数项保持默认即可。

表 5-5 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 5-10 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-mwlye1NlLG”。

图 5-11 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 5-6 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“自定义资源池”。

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

相关操作

- 如果需要对代码检查任务进行更多配置，可参考**配置代码检查任务**。
- 执行代码检查任务时常见问题请参考**代码检查常见问题**。

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**。

资源与成本规划

本实践需购买ECS作为自定义执行机，关于ECS的购买费用可参考**价格计算器**。

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。执行机规格为CentOS 7及以上，推荐配置如下：
  - CPU：8U，内存：32G，系统盘：100G，数据盘：250G。（使用该执行机规格时，**需要**执行**挂载云硬盘**操作。）
  - CPU：8U，内存：32G，系统盘：250G。（使用该执行机规格时，**无需**执行**挂载云硬盘**操作。）
- 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
```

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 > 代码检查 CodeArts Check”。

步骤3 单击“前往代码检查”，进入CodeArts Check服务首页。

步骤4 在导航栏切换到“首页”页签，依次单击“新建 > 新建项目”，选用“Scrum”项目模板。

步骤5 填写项目名称，例如“check-bestpractice”，其他参数保持默认即可。

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。
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，需要安装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“代码 > 代码托管”，进入“check-bestpractice”项目下的代码托管服务页面。

步骤2 单击“新建仓库”，选择CR仓库模式。

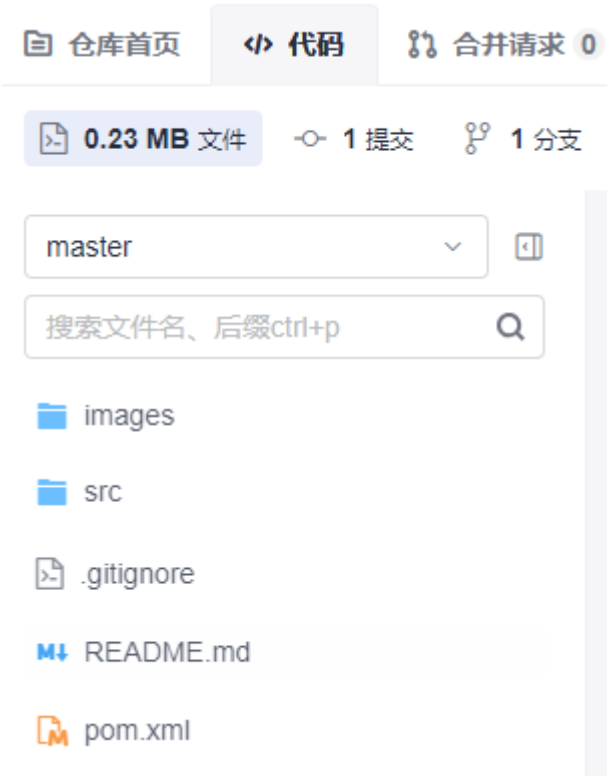
步骤3 在新建仓库页面选择“按模板新建”，单击“下一步”。

步骤4 选择“Java Maven Demo”仓库模板，单击“下一步”。

步骤5 “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“自定义资源池”。

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-mwlye1NlLG”，表示本次代码检查任务由自定义执行机执行。其中“agent_test_custom-mwlye1NlLG”为**步骤8**中代理执行机的代理别名。

相关操作

- 如果需要对代码检查任务进行更多配置，可参考**配置代码检查任务**。
- 执行代码检查任务时常见问题请参考**代码检查常见问题**。

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**。

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 > 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。

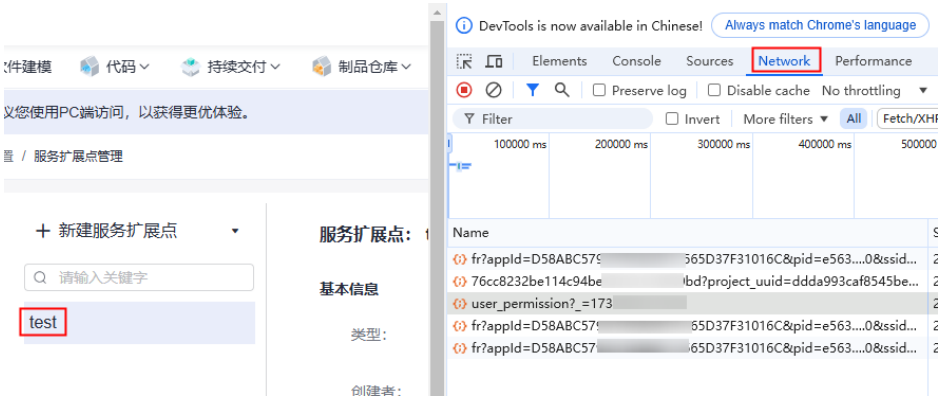
----结束

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

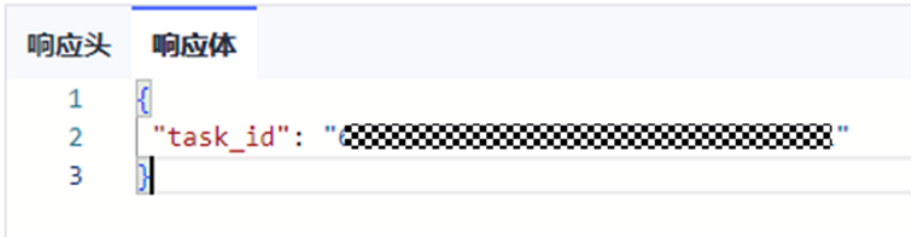
步骤1 以下为新建全量任务的请求示例。

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代码仓，需具备Repo仓的相关权限且需填写ssh地址
  "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": "f2xxxb0", #密码或用户token，仅在三方仓场景下使用
  "endpoint_id": "fxxxxxsz" #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



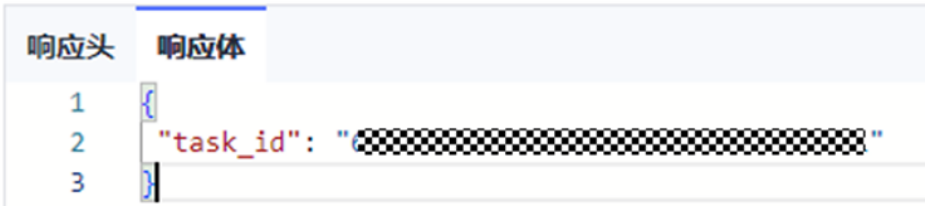
创建全量任务后，可在华为云对应局点账号下的项目中，查看到新建的代码检查任务。

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，需具备Repo仓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



响应值为增量任务的任务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": "9[redacted]d"
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

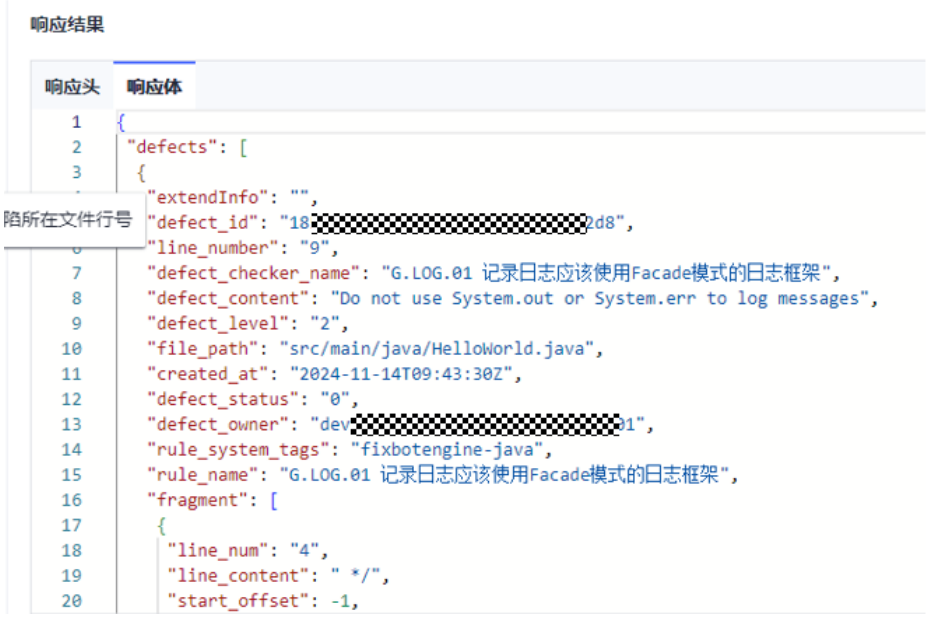
查询缺陷详情

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。



----结束

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 > 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。

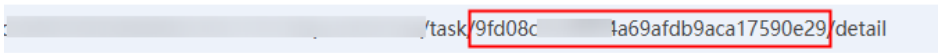
表 8-3 通用 Git 代码检查任务参数说明

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

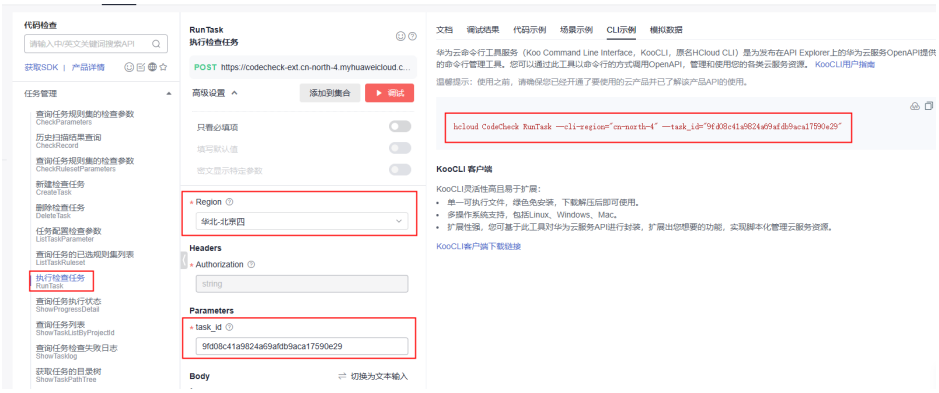
步骤1 访问**华为云命令行工具服务KooCLI**文档页面。

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 基于第三方引擎执行代码检查

应用场景

如果CodeArts Check当前提供的的检查引擎不满足用户的使用需求，可使用第三方引擎检查工具执行代码检查任务。

约束限制

- 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**。

资源与成本规划

本实践需购买ECS作为自定义执行机，关于ECS的购买费用可参考**价格计算器**。

前提条件

- 已联系技术支持获取UCCP插件的代码开发样例包。
- 已参考**自定义购买ECS**购买本实践使用的EulerOS 7操作系统弹性云服务器。推荐配置如下：
  - CPU：8U，内存：32G，系统盘：100G，数据盘：250G。（使用该执行机规格时，**需要**执行**挂载云硬盘**操作。）
  - CPU：8U，内存：32G，系统盘：250G。（使用该执行机规格时，**无需**执行**挂载云硬盘**操作。）
- 已在接入的自定义执行机中已安装Git-lfs，若未安装，可在执行机上参考以下命令安装Git-lfs。

```
# 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
```


操作流程

表 9-1 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。
开发UCCP插件	开发UCCP插件，用于导入自定义规则。
将第三方引擎和插件导入执行机	将第三方引擎和插件导入执行机。
上传自定义规则	在CodeArts Check服务中上传自定义规则。
新建CodeArts Repo代码仓	为本实践新建存储代码的代码仓。
创建代码检查任务	为本实践创建代码检查任务。
配置代码检查任务	配置代码检查任务使用自定义执行机和自定义规则。
查看检查结果	查看代码检查结果。

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 > 代码检查 CodeArts Check”。

步骤3 单击“前往代码检查”，进入CodeArts Check服务首页。

步骤4 在导航栏切换到“首页”页签，依次单击“新建 > 新建项目”，选用“Scrum”项目模板。

步骤5 填写项目名称，例如“check-bestpractice”，其他参数保持默认即可。

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9-2配置参数后，单击“保存”。

表 9-2 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9-3**配置代理信息，其他参数项保持默认即可。

表 9-3 新建代理参数说明

参数	说明
是否安装Docker	勾选此项，需要安装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”，然后单击“关闭”。

图 9-1 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**前提条件**中购买的服务器所在行的“远程登录”，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 9-2 代理执行机

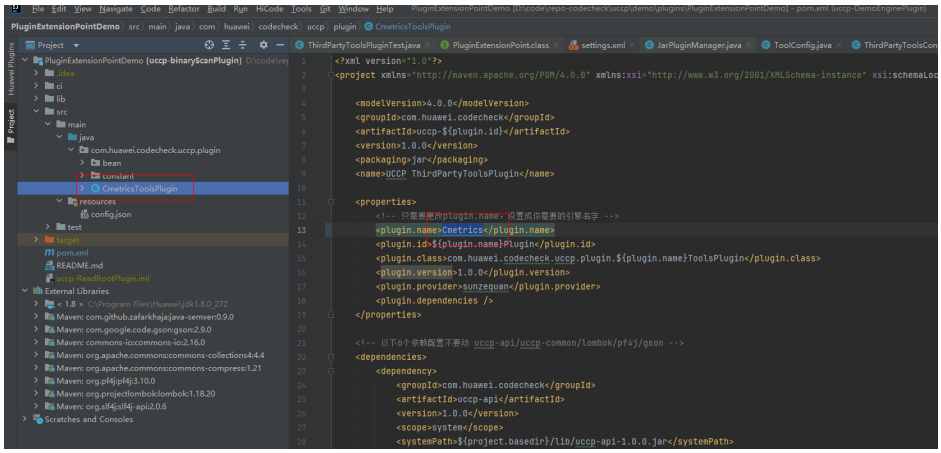
----结束

开发 UCCP 插件

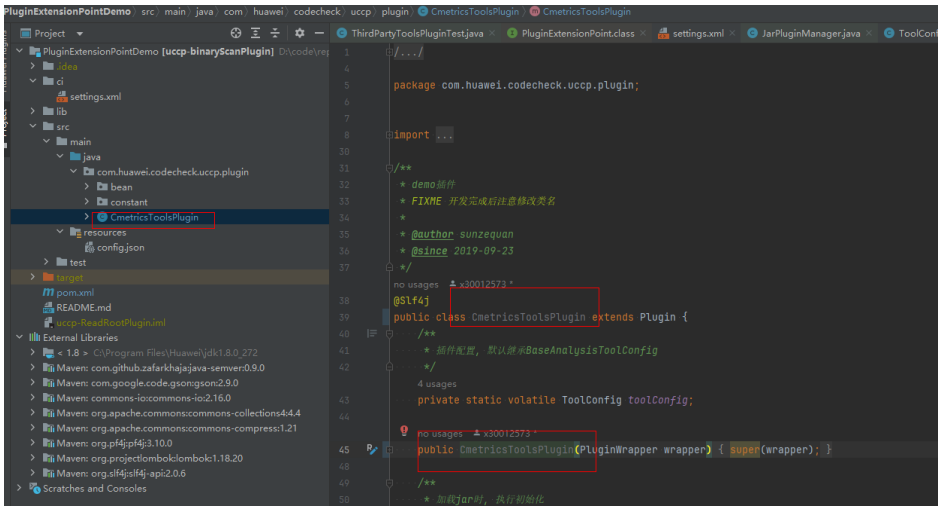
步骤1 将**前提条件**中获取到的开发样例包“PluginExtensionPointDemo.zip”解压至本地。

步骤2 找到工程根目录下的“pom.xml”文件，将“<plugin.name>DemoEngine</plugin.name>”修改为新接入的引擎名称“<plugin.name>cmetrics</plugin.name>”。

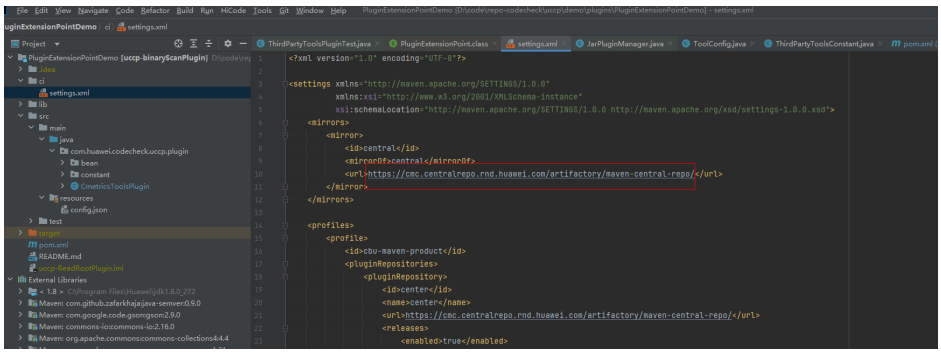
步骤3 找到类“ThirdPartyToolsPlugin.java”，将名称更改为**步骤2**中新接入的引擎名称，例如“CmetricsToolsPlugin”。



注意更改构造函数名称需与文件名保持一致。



步骤4 修改“settings”文件，更改为外部可下载依赖的仓库。



步骤5 实现“CmetricsToolsPlugin”中“execute”方法。

入参说明如下：


```
        // 只在此处添加执行逻辑即可
        x30012573 *
        @Override
        public ReportFileResult execute(ToolParameterInfo toolParameterInfo) {
            // 以下为重要参数，其他可忽略
            // 检查模式       * 0 - 全量检查   * 1 - 增量检查 * 2 - ide检查
            int mode = toolParameterInfo.getMode();
            // 检查代码根路径(绝对路径)
            String projectRoot = toolParameterInfo.getProjectRoot();
            // 检查文件列表(绝对路径)
            List<String> sourceFiles = toolParameterInfo.getSourceFiles();
            // 规则集
            List<ToolRuleSetInfo> rules = toolParameterInfo.getRuleSets();
            // 模型集
            List<ModelInfo> models = toolParameterInfo.getModels();
            // 报告生成路径，引擎生成相关的文件都需要放在此目录下
            String reportDir = toolParameterInfo.getReportDirPath();
            // 配置的检查目录(绝对路径)
            List<String> includePaths = toolParameterInfo.getIncludePaths();
            // 排除参数配置，正则表达式
            String exclude = toolParameterInfo.getExclude();
            // 引擎参数配置，用户可通过codecheck前端界面配置透传到引擎
            Map<String, String> parameters = toolParameterInfo.getParameters();
```

返回一个缺陷列表文件即可，缺陷字段说明见下图：

```
            log.info("empty mode");
        }
        List<DefectInfo> defects = new ArrayList<>();
        // 以下缺陷字段为重要字段，其他的选填
        DefectInfo defectInfo = new DefectInfo();
        // 缺陷所在代码行的代码片段
        defectInfo.setMainBuggyCode("缺陷所在代码行的代码片段");
        // 缺陷所属的语言
        defectInfo.setLanguage("java");
        // 缺陷所在代码行号
        defectInfo.setMainBuggyLine(11);
        // 缺陷所属文件 - 相对projectRoot路径
        defectInfo.setBuggyFilePath("src/java/com/xxx/Demo.java");
        // 缺陷类型，值要跟规则名称保持一致
        defectInfo.setDefectType("used to generate code functionCap");
        // 检查引擎名称，值要跟规则中的引擎名称保持一致
        defectInfo.setAnalyzerName("ReadRoot");
        String outputFile = reportDir + File.separator + "output.json";
        defects.add(defectInfo);
        ReportFileResult result = new ReportFileResult();
        result.setSuccess(true);
        DefectReportWriter writer = null;
        try {
            writer = new DefectReportWriter(outputFile);
            for (DefectInfo defect : defects) {
                writer.write(defect);
            }
        } catch (IOException e) {
            e.printStackTrace();
            result.setSuccess(false);
        } finally {
            if (writer != null) {
                writer.close();
            }
        }
        log.info("ThirdPartyToolsAnalyze test success");
        result.setReportPath(outputFile);
        return result;
    }
}
```

步骤6 建议在“execute”中间调用引擎脚本，直接由引擎生成“defect”缺陷列表文件。

----结束

将第三方引擎和插件导入执行机

步骤1 在执行机“/opt/cloud/”目录下，执行以下命令，创建“third_party_tools/v1/plugins”目录。

```
mkdir -p /opt/cloud/third_party_tools/v1/plugins
```

步骤2 在自定义执行上将开发完成的UCCP插件放入工具插件路径“/opt/cloud/third_party_tools/v1/plugins”，然后执行以下命令更改权限。

```
chmod -R 777 [所换包的路径] &
chown -R slave1:slave1 [所换包的路径] &
```

```
[root@ecs-2eec plugins]# ll
total 56
-rwxrwxrwx 1 slave1 slave1 56087 Apr 14 19:10 uccp-CmetricsPlugin-1.0.0.jar
[root@ecs-2eec plugins]# pwd
//opt/cloud/third_party_tools/v1/plugins
```

步骤3 将工具包放入工具路径“/opt/cloud/third_party_tools/v1/tools”，然后执行以下命令更改权限。

```
chmod -R 777 [所换包的路径] &
chown -R slave1:slave1 [所换包的路径] &
```

```
[root@ecs-2eec tools]# ll
total 4
drwxrwxrwx 2 slave1 slave1 4096 Apr 14 19:11 cmetrics
[root@ecs-2eec tools]# pwd
//opt/cloud/third_party_tools/v1/tools
```

----结束

上述插件路径和工具路径中的“v1”代表“版本号”。可以同时存在多个版本，例如“/opt/cloud/third_party_tools/v1/plugins”、“/opt/cloud/third_party_tools/v2/plugins”、“/opt/cloud/third_party_tools/v3/plugins”。在执行任务时，会选择最大的版本号执行任务。例如在这里会选择/opt/cloud/third_party_tools/v3/plugins下的插件来执行任务。此规则同样适用于“tools”。

上传自定义规则

步骤1 进入代码检查首页，在“规则”页面，单击“新建规则”，选择“批量导入规则”。

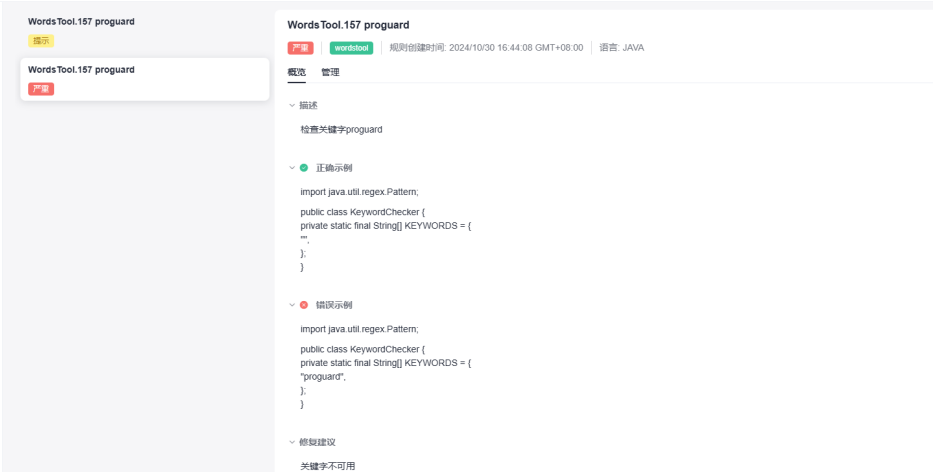
批量导入规则可以导入任一引擎，任一语言的规则。

图 9-3 批量导入规则

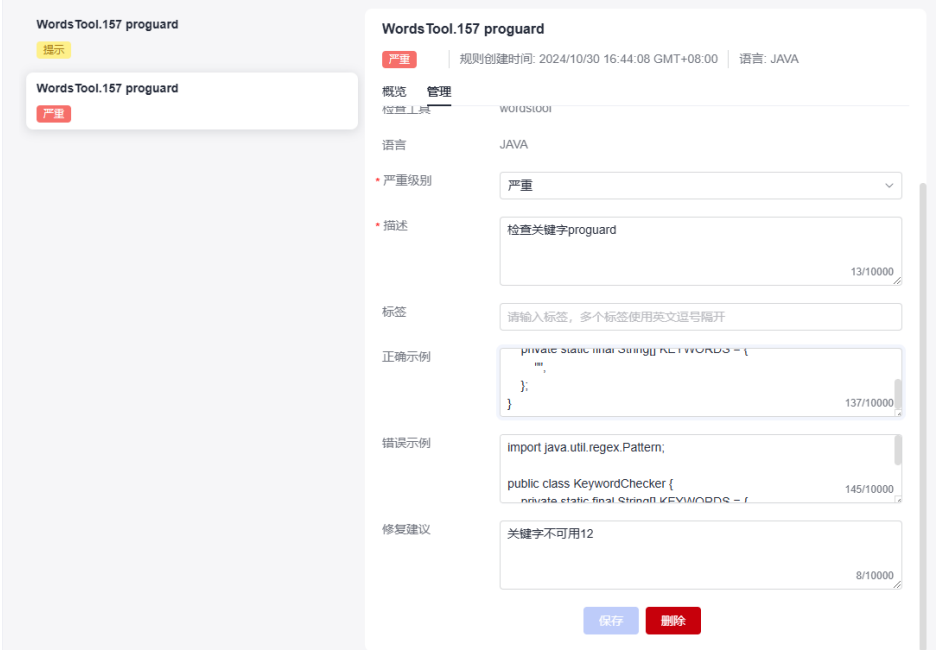
步骤2 代码检查服务提供了固定的表格模板，必须按照模板填写规则信息，才能成功导入。

图 9-4 上传规则文件

步骤3 导入成功后，在左边选择“自定义规则”，就可以看到**步骤1**中导入的规则。



自定义规则可由创建者进行编辑和删除。单击“管理”，可以修改规则信息，或者删除此条规则。



----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 在代码托管页面，单击“新建仓库”。

步骤3 在新建仓库页面，选择“按模板新建”。

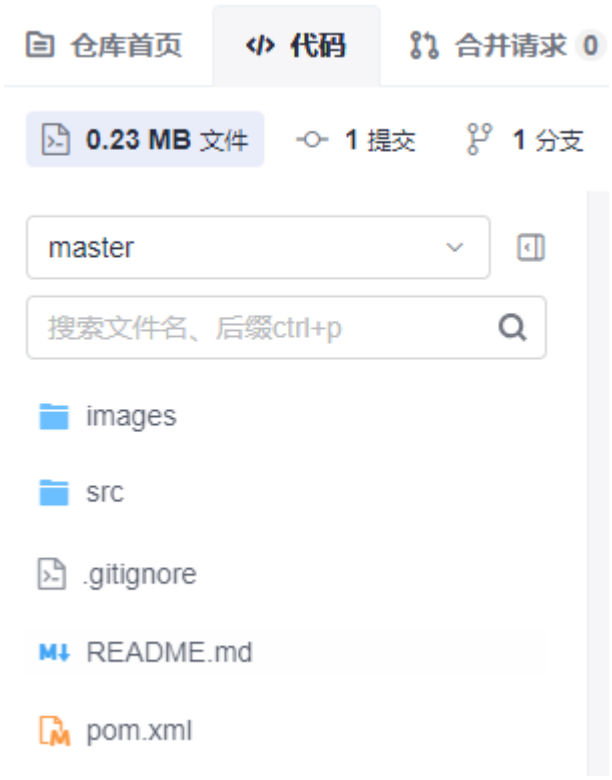
步骤4 单击“下一步”，搜索并选择“Java Maven Demo”模板。

步骤5 单击“下一步”，仓库名称填写“custom_repo”，“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6 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9-5所示。

图 9-5 代码仓文件目录



----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**表9-4**配置参数。

表 9-4 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“check-bestpractice”。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**新建CodeArts Repo代码仓**中创建的代码仓“custom_repo”。
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

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**上传自定义规则**中规则，单击右上角“保存”。

----结束

配置代码检查任务

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

10 安全执行代码检查任务

应用场景

CodeArts Check提供代码安全检查增强包的能力，其安全检查能力作为深度价值特性，能深度识别代码中安全风险和漏洞，提供了套餐包内规则不覆盖的安全类场景，比如数值错误、加密问题、数据验证问题等。针对业界的安全漏洞检测项提供了更深入的分析能力，如跨函数、跨文件、污点分析、语义分析等。

资源与成本规划

需参考**购买CodeArts增值特性**购买代码检查服务的增强包。价格可参考**价格计算器**。

操作流程

表 10-1 操作流程

步骤	说明
新建项目	创建代码检查任务所属的项目。
新建CodeArts Repo代码仓	创建代码检查任务使用的代码仓。
配置规则集执行检查任务	配置代码检查任务使用带有安全增强包的规则集。
查看检查结果	查看检查结果，确认使用的规则是否生效。

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 > 代码检查 CodeArts Check”。

步骤3 单击“前往代码检查”，进入CodeArts Check服务首页。

步骤4 在导航栏切换到“首页”页签，依次单击“新建 > 新建项目”，选用“Scrum”项目模板。

步骤5 填写项目名称，例如“check-bestpractice”，其他参数保持默认即可。

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

----结束

新建 CodeArts Repo 代码仓

步骤1 在导航栏中选择“代码 > 代码托管”，进入“check-bestpractice”项目下的代码托管服务页面。

步骤2 单击“新建仓库”。

步骤3 在新建仓库页面选择“按模板新建”，单击“下一步”。

步骤4 选择“Java Maven Demo”仓库模板，单击“下一步”。

步骤5 “代码仓库名称”填写为“Repo01”，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。

----结束

配置规则集执行检查任务

步骤1 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。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“华为Java增强编程规则集”。

图 10-1 选择规则集

步骤4 单击“编译配置”，将“编译工具选项”开关设置为 状态，“编译工具”选择“maven”。其他参数保持默认即可，然后单击“确定”。

图 10-2 编译配置

步骤5 单击右上角“开始检查”。

----结束

查看检查结果

出现“华为Java增强编程规则集”中规则检查出的问题，表示本次代码检查任务使用的是“华为Java增强编程规则集”。如图10-3所示。

图 10-3 检查结果

相关操作

更多规则集的配置，可参考**配置代码检查任务规则集**。

11 HE2E DevOps 实践之代码检查

本文以“DevOps全流程示例项目”为例，介绍如何在项目中进完成代码检查配置。

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 11-1 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，使检查更全面。

本实践中以增加Python语言检查规则集为例介绍操作方法。

步骤1 进入项目“凤凰商城”，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。

步骤2 在列表中找到任务“phoenix-codecheck-worker”，在“操作”列中单击···，选择“设置”，进入设置页面。

步骤3 单击导航“规则集”，规则集中默认包含的语言是“JAVA”。

步骤4 增加Python语言检查规则集。

1. 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。
2. 将PYTHON语言对应的开关状态设置为。

页面提示设置成功，完成规则集的配置。

步骤5 单击“开始检查”，启动任务。

当页面显示 检查成功 ，表示任务执行成功。

如果任务执行失败，请根据报错提示，参考**代码检查常见问题**排查处理。

----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。

图 11-1 查看问题帮助

----结束