CodeArts Build_ug_zh-华为云&伙伴云-整改版

CodeArts Build_ug_zh-华为云&伙伴云-整改版

文档版本 01

发布日期 2025-02-10

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。

安全声明

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
https://www.huawei.com/cn/psirt/vul-response-process
如企业客户须获取漏洞信息，请参见如下网址：
https://securitybulletin.huawei.com/enterprise/cn/security-advisory

目 录

1 编译构建服务(CodeArts Build)使用流程

编译构建是指将软件的源代码编译成目标文件，并和配置文件、资源文件等一起打包的过程。

编译构建服务（CodeArts Build）为开发者提供配置简单的混合语言构建平台，实现编译构建云端化，支撑企业实现持续交付，缩短交付周期，提升交付效率。支持编译构建任务一键创建、配置和执行，实现获取代码、构建、打包等活动自动化，实时监控构建状态，让您更加快速、高效地进行云端编译构建。

在**软件开发生产线**解决方案中，编译构建服务属于其中一个子服务，具体位置可参考**产品架构**。

更多编译构建服务信息请参考**产品介绍**。

CodeArts Build 基本操作流程

图 1-1 CodeArts Build 基本操作流程

表 1-1 CodeArts Build 操作流程说明

流程	说明
开通CodeArts Build	为您介绍如何**开通CodeArts Build**和**购买构建加速包**以及**并发包**。
配置CodeArts Build项目级角色权限	为您介绍使用CodeArts Build前的**项目级基础权限配置**、**访问CodeArts Build服务页面**的方式以及**CodeArts Build首页功能总览**。

流程	说明
新建构建任务	为您介绍**图形化新建构建任务**和**代码化新建构建任务**的操作指导，以及如何配置构建任务的**参数**、**执行计划**、**单任务角色权限**和**事件通知**。
配置构建步骤	CodeArts Build内置了**30+种构建工具**，您可以根据实际使用情况选择使用的构建工具，每种工具分别为您介绍了图形化构建的配置指导和代码化构建的代码示例。
执行构建任务	构建任务可通过流水线触发或者定时任务触发执行，本节为您介绍在CodeArts Build服务页面**执行单个构建任务**。 • 针对C/C++语言构建任务的效率提升，可参考**购买构建并发包**和**加速构建任务**实现构建加速。 • 针对多构建任务的效率提升，可参考**多任务YAML文件结构详解**和**购买构建并发包**搭配实现构建加速。
查看构建任务	为您介绍如何**查看构建任务信息以及构建任务的执行结果**。

2 购买并授权使用 CodeArts Build

2.1 购买 CodeArts Build

前提条件

已注册华为云并实名认证，如果还没有华为账号，请参考以下步骤创建。

1. 打开**华为云网站**。
2. 单击“注册”，根据提示信息完成注册。
   注册成功后，系统会自动跳转至您的个人信息界面。
3. 参考**实名认证**完成个人或企业账号实名认证。

购买 CodeArts Build 须知

- 在**CodeArts支持的区域**内，各区域独立开通购买、独立计费。
- 您可以**购买CodeArts Build套餐**，或者**开通/购买软件开发生产线服务组合套餐**，体验一站式、全流程、安全可信的软件开发生产线。
- 若已经购买了软件开发生产线服务组合套餐，则无需再单独购买CodeArts Build套餐。

开通 CodeArts Build

步骤1 进入**购买编译构建服务页面**。

步骤2 参考**表2-1**确认资源规格信息，单击“免费开通”。

表 2-1 资源规格

资源项	资源规格
构建时长（分钟/月）	免费构建时长1800分钟/月。
构建并发（个）	1个内置执行机（2U8G）和1个自定义执行机并发。

步骤3 参考表2-2确认套餐包配置信息，勾选“我已经阅读并同意”，单击页面右下角“立即开通”，即可下单成功。

表 2-2 套餐包配置

配置项	配置详情
计费模式	包年/包月。
区域	选择需要使用的区域。不同区域购买的资源不能跨区使用，需慎重选择。
产品	CodeArts Build套餐
规格	选择“Build专业版”。 免费使用构建时长1800分钟/月，单租户1个内置执行机（2U8G）并发和1个自定义执行机并发，使用限制性资源池，超过并发数时构建任务会进行排队。用户可额外购买构建加速包提升效率和购买构建并发包扩容。
购买时长	选择“1个月”。
自动续费	勾选后将开启自动续费。自动续费规则请参考自动续费规则说明。

下单成功即服务开通成功。

----结束

购买构建加速包

构建加速包无法单独购买，需已购买编译构建套餐包或CodeArts套餐包。

步骤1 登录构建加速包购买页。

步骤2 单击左侧导航栏的 图标，选择“开发与运维 > 编译构建CodeArts Build” 。

步骤3 在“构建增值特性”区域，单击“购买”。

步骤4 根据实际需要配置购买详情，并勾选同意声明。

表 2-3 特性包配置

配置项	详情
计费模式	包年/包月。

配置项	详情
区域	选择需要使用的区域。不同区域购买的资源不能跨区使用，需慎重选择。
产品	选择“构建加速”。
CPU架构	可根据实际情况选择“X86”或“ARM”。
加速级别	根据实际需要选择加速级别。 • L1级别：对于C/C++的工程，典型的编译过程是CPU消耗型任务，编译效率受限于编译并发度，编译并发度受限于单机资源规格，传统的单机构建模式很难突破资源规格的瓶颈。L1级别通过分布式编译技术，将单机编译任务分发到加速包后台资源上进行编译，支持远超单机资源的并发数，突破单机资源规格的限制，从而实现提升编译效率的目标。 • L2级别：对于大多数开发过程，构建之间只有少量代码变更，除去更新的部分外，其余的代码编译均为重复构建。L2级别通过增量构建提升编译效率，在编译过程中对编译结果进行缓存，下次编译时通过对源码的变更来判断是否可以命中缓存，通过缓存大幅减少重复编译任务的执行，从而实现提升编译效率的目标。 • L3级别：L3级别同时提供分布式编译和增量编译的能力，对于没有变化的代码提供增量编译，对于变化的代码提供分布式编译。最大限度地提升构建效率。
购买数量	根据实际需求填写数量，最多16个。
购买时长	根据实际需要选择1个月~3年。
自动续费	勾选后将开启自动续费。自动续费规则请参考**自动续费规则说明**。

步骤5 单击“下一步”，确认订单内容：若需要修改，单击“上一步”；若确认无误，单击“去支付”。

步骤6 根据界面提示完成支付。

步骤7 返回控制台，即可查看到已购买的特性包详情。

若控制台未显示特性包信息、或当前状态为“处理中”，请稍等片刻后刷新页面查看。

----结束

购买构建并发包

构建并发包无法单独购买，需已**购买CodeArts Build套餐**或**CodeArts套餐包**。

步骤1 登录**构建并发包购买页**。

步骤2 单击左侧导航栏的 ☰ 图标，选择“开发与运维 > 编译构建CodeArts Build”。

步骤3 在“构建资源扩展”区域，单击“购买”。

步骤4 根据需要配置购买详情，并勾选同意声明。

表 2-4 特性包配置

配置项	详情
计费模式	包年/包月。
区域	选择需要使用的区域。不同区域购买的资源不能跨区使用，需慎重选择。
产品	“构建并发”。
执行机类型	可根据实际情况选择“内置执行机”或者“自定义执行机”。 当选择“内置执行机”时，需要根据实际使用情况选择“X86”或“ARM”，支持购买的执行机规格有 ● 2U8G：资源规格为2 vCPU / 8 GB / 60 GB 磁盘空间。 ● 4U8G：资源规格为4 vCPU / 8 GB / 100 GB 磁盘空间。 ● 8U16G：资源规格为8 vCPU / 16 GB / 100 GB 磁盘空间。 ● 16U32G：资源规格为16 vCPU / 32 GB / 500 GB 磁盘空间。 ● 16U64G：资源规格为16 vCPU / 64 GB / 1000 GB 磁盘空间。
购买数量	根据实际需求填写数量，最多50个。
购买时长	根据实际需要选择1个月~3年。
自动续费	勾选后将开启自动续费。自动续费规则请参考**自动续费规则说明**。

步骤5 单击“下一步”，确认订单内容：若需要修改，单击“上一步”；若确认无误，单击“去支付”。

步骤6 根据界面提示完成支付。

步骤7 返回控制台，即可查看到已购买的特性包详情。

若控制台未显示特性包信息、或当前状态为“处理中”，请稍等片刻后刷新页面查看。

----结束

3 配置 CodeArts Build 项目级角色权限

新增的成员需赋予指定的角色，不同角色具备的默认权限不同。各角色具备的默认权限如表3-1。

表 3-1 编译构建服务默认角色权限

角色	创建	编辑	删除	查看	执行	复制	禁用	权限管理	分组
项目经理	√	√	√	√	√	√	√	√	√
产品经理	×	×	×	√	×	×	×	×	×
测试经理	×	×	×	√	×	×	×	×	×
运维经理	×	×	×	×	×	×	×	×	×
系统工程师	√	√	√	√	√	√	√	×	√
Committer	√	√	√	√	√	√	√	×	×
开发人员	√	√	√	√	√	√	√	×	×
测试人员	×	×	×	×	×	×	×	×	×
参与者	×	×	×	×	×	×	×	×	×
浏览者	×	×	×	√	×	×	×	×	×

角色	创建	编辑	删除	查看	执行	复制	禁用	权限管理	分组
项目管理员	√	√	√	√	√	√	√	√	√

说明

“√”表示具备该权限，“×”表示不具备该权限。

前提条件

- 已**购买CodeArts Build**。
- 已参考**软件开发生产线(CodeArts)**的“用户指南 > 软件开发生产线（CodeArts）使用前准备 > 添加CodeArts项目成员”章节添加成员，并参考“管理CodeArts权限”章节为新增的成员赋予角色。

访问 CodeArts Build 服务首页

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 > 编译构建CodeArts Build”。

步骤3 编译构建服务页面有两种访问方式：首页入口和项目入口。

- **首页入口**

  单击“立即使用”，进入编译构建服务首页。该页面展示的是与当前用户相关的构建任务列表。

- **项目入口**

  a. 单击“立即使用”，进入编译构建服务首页。

  b. 单击导航栏“首页”。

  c. 单击需要查看的项目名称。

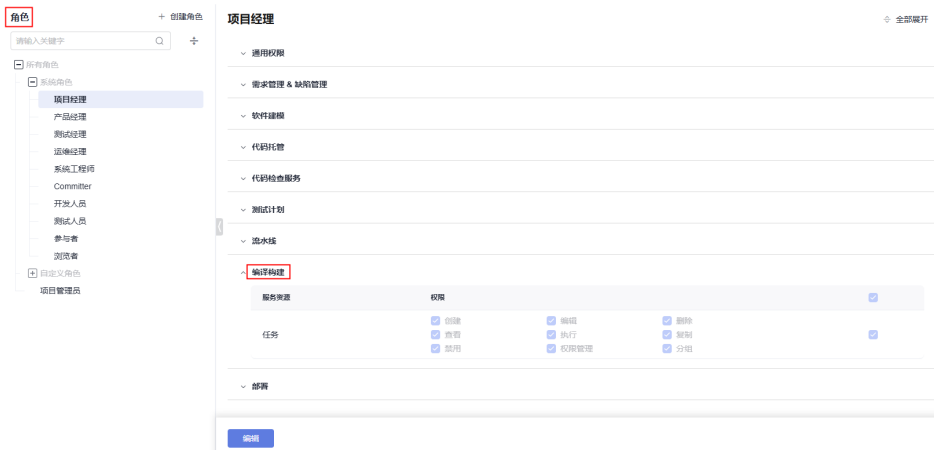
  d. 选择“持续交付 > 编译构建”，进入指定项目下构建任务列表页。

----结束

配置角色权限

1. 通过项目入口方式**访问CodeArts Build服务首页**。
2. 选择导航栏“设置 > 权限管理”。
3. 在配置角色页面，为不同的角色配置编译构建服务的各个资源权限。

图 3-1 配置项目级角色权限



配置不同角色对当前构建任务的操作权限，可参考**配置构建任务角色权限**。

CodeArts Build 服务首页功能总览

CodeArts Build提供多种外观主题，本节以“无限+经典”主题为例介绍导航栏内容。

表 3-2 首页功能总览说明

菜单项	说明
	单击下拉列表可切换服务所属区域。 每个区域之间数据及资源不互通，请选择您已购买的区域进行使用。
	单击下拉列表，可选择“编译构建”，进入编译构建服务首页。 该页面展示当前租户创建的所有项目的所有构建任务。
	如果以项目入口访问CodeArts Build服务，单击此处的下拉列表，可切换至其它项目。
	单击可在下拉列表中访问**自定义模板**、**自定义构建环境**、**文件管理**、**构建任务回收站**和构建资源池管理。
	单击可**执行构建任务**。
	单击可**收藏构建任务**。
	单击可在下拉框中**编辑**、**复制**、**禁用**和**删除**构建任务。

4 新建构建任务

4.1 新建分组

对于在同一项目中的不同模块或使用场景不同的构建任务，编译构建服务支持分组管理。当创建一个分组后，会默认同步创建一个“未分组”分组，若创建构建任务时未选择分组，则创建的构建任务归档在“未分组”中。

约束与限制

- 最多支持创建50个分组。
- 创建分组支持多层创建，最多支持3层。

新建分组

1. 通过项目入口方式**访问CodeArts Build服务首页**。
2. 单击“新建分组”，在弹出的窗口中单击+。

图 4-1 新建分组

3. 根据实际使用情况，设置分组名称，单击 ，新建完成。

图 4-2 设置分组名称

4. 单击“关闭”，完成创建分组。

管理分组

1. 单击 ，打开分组管理。

图 4-3 分组管理

2. 将鼠标放在分组所在行。

   单击，可修改分组名称。

   单击，可调整分组顺序和删除分组。

   说明

   默认存在一个“未分组”分组，当分组删除后，该分组下的任务会被移入“未分组”中。

4.2 新建构建任务（图形化构建）

图形化构建是指通过在CodeArts Build页面配置构建工具的相关参数实现编译构建，可以根据实际使用场景自定义组合构建工具。

关于图形化构建的使用示例，可参考**CodeArts Build最佳实践**。

约束与限制

当构建任务的代码源为用户本地代码仓时，出于安全性考虑，如仅需CodeArts Build可以访问代码仓，可将下列IP加入代码仓服务器的访问白名单中。

- 华北-北京四、华北-北京一、东北-大连：121.36.9.82、119.3.235.73
- 华南-广州、华南-深圳、华东-上海一、华东-上海二、西南-贵阳一：139.159.236.151、124.71.112.32

新建构建任务前准备工作

- 如果使用的是CodeArts Repo代码仓，需已具备代码托管服务（CodeArts Repo）的操作权限。
- 参考软件开发生产线(CodeArts)的“用户指南 > 软件开发生产线（CodeArts）使用前准备 > 新建CodeArts项目”，**新建CodeArts项目**。

  如果已有项目，无需执行此步骤。
- 参考代码托管服务（CodeArts Repo）的“用户指南 > 创建 > 代码托管仓库”，新建仓库。

  如果用户使用的是第三方代码仓或已有CodeArts Repo代码仓，无需执行此步骤。

新建图形化构建任务

1. 通过项目入口方式**访问CodeArts Build服务首页**。
2. 单击“新建任务”，进入配置“基本信息”页面，参考**表4-1**填写构建任务基本信息。然后单击“下一步”，进入“构建模板”页面。

表 4-1 基本信息配置说明

参数	说明
名称	创建的编译构建任务名称，可自定义。 ● 支持中英文，数字，下划线“_”和连接符“-”。 ● 字符长度范围为1~115。
所属项目	创建的编译构建任务所属项目。 ● 以项目入口方式访问访问编译构建服务时默认填写，无需手动填写。 ● 以服务入口访问时需根据实际情况选择**新建构建任务前准备工作**中创建的项目。
代码源	选择实际需要编译的代码源。 ● Repo：从**代码托管服务**拉取代码进行构建。 ● 其他项目Repo：从其他项目的代码托管中拉取代码进行构建，请选择已有的项目、该项目下已经创建的代码仓以及默认分支。 ● 来自流水线：如果选择来自**流水线**，则只能通过流水线任务驱动执行，不能单独执行。 以下为非CodeArts的第三方代码仓库。 ● GitHub：拉取托管在GitHub上的代码进行构建。 ● 通用Git：拉取托管在其他服务上的代码进行构建。 ● GitCode：拉取托管在GitCode仓库上的代码进行构建。 ● 码云：拉取托管在码云上的代码进行构建。 ● Gerrit：拉取托管在Gerrit上的代码进行构建。
服务扩展点	可选参数。当“代码源”选择为第三方代码仓时需要配置，首次使用第三方代码仓，需新建服务扩展点。新建步骤可参考**新建CodeArts服务扩展点**。
代码仓	选择实际需要编译的代码仓。
默认分支	选择仓库默认分支。
描述	可选参数。根据实际场景对编译构建任务的描述。字符长度范围0~512。

3. CodeArts Build内置30+种构建模板，您可以根据实际需要选择构建模板，选择后单击“确定”，构建任务即可新建完成。
   - 也可以选择“空白构建模板”，然后在**配置构建任务**时添加实际使用的构建步骤。

- 如果预置模板不满足使用要求，也可以**自定义模板**。

4. 页面自动跳转到“构建步骤”页面，可继续**配置构建任务**。

自定义构建任务模板

如果需将当前的构建任务保存为模板，以便后续创建构建任务时选择，则可以按照以下操作执行。

步骤1 在构建任务历史页面，单击页面右上角···，在下拉列表中选择“保存模板”。

步骤2 在弹框中输入模板名称与模板描述，单击“保存”。

步骤3 单击用户名，在下拉菜单中选择“租户设置”。

步骤4 选择导航栏“编译构建 > 自定义模板”，即可在列表中看到已保存的构建模板。

对已保存的构建模板，可以完成以下操作：

表 4-2 管理自定义模板

操作	说明
搜索模板	在搜索框输入关键字，可搜索模板。
收藏模板	单击☆，可以收藏该模板。
删除模板	单击🗑，在弹框中单击“确定”，即可删除该模板。 仅可以删除当前用户创建的模板。

----结束

4.3 新建构建任务（代码化构建）

代码化构建是指通过YAML文件配置构建脚本，将构建过程需要用到的构建环境、构建参数、构建命令、构建工具等信息通过YAML语法编写成“build.yml”文件，并且将“build.yml”文件随着被构建的代码一起存储代码仓库，执行构建任务时，系统会以“build.yml”文件作为构建脚本执行构建任务。

代码化构建功能优势如下：

- 清晰描述构建过程：构建参数、构建命令、构建步骤、以及构建后的操作，使构建过程可信。
- 每次构建使用对应当前commit的“build.yml”配置，保证构建可还原可追溯，不必担心因修改了构建配置而不能重复执行之前的任务。
- 如果新特性需要修改构建脚本，开发人员可以拉一个新的分支修改“build.yml”去测试，而不用担心影响其他分支。

约束与限制

代码化构建仅支持使用CodeArts Repo代码仓。

新建构建任务前准备工作

- 已具备CodeArts Repo服务的操作权限。
- 已参考代码托管服务（CodeArts Repo）的“用户指南 > 创建 > 代码托管仓库”，新建代码仓库。
- 参考软件开发生产线(CodeArts)的“用户指南 > 软件开发生产线（CodeArts）使用前准备 > 新建CodeArts项目”，**新建CodeArts项目**。

  如果已有项目，无需执行此步骤。

创建代码化构建使用的 YAML 文件

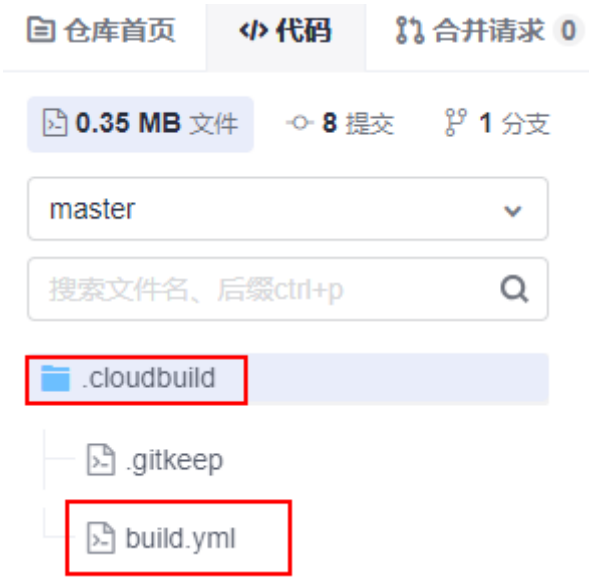
1. 通过项目入口方式**访问CodeArts Build服务首页**。
2. 选择导航栏“代码 > 代码托管”，进入代码托管页面。
3. 单击“新建仓库”，选择“普通仓库”，单击“下一步”，根据**表4-3**填写参数后，单击“确定”。

   表 4-3 新建代码仓

参数	说明
代码仓库名称	自定义代码仓名称。例如：maven_yml_build。 ● 以数字、字母或者“_”开头。 ● 可包含“.”和“-”。 ● 不能以“.git”、“.atom”或者“.”结尾。
描述	可选参数。对代码仓的描述。
选择gitignore	可选参数。根据编程语言选择“.gitignore”，例如：Java。
初始化设置	勾选全部。 ● 允许项目内人员访问仓库：选择后会自动将项目中的项目经理设为仓库管理员，开发人员设为仓库普通成员。当项目新增这两个角色时，也会自动同步到已经存在的仓库中。 ● 允许生成README文件：可以通过编辑README文件，记录项目的架构、编写目的等信息，相当于对整个仓库的一种注释。 ● 自动创建代码检查任务（免费）：仓库创建完成后在代码检查任务列表中，可看到对应仓库的检查任务。
可见范围	设置为“私有”。 ● 私有：仓库仅对仓库成员可见，仓库成员可访问仓库或者提交代码。 ● 公开只读：仓库对所有访客公开只读，但不出现在访客的仓库列表及搜索中，您可以选择开源许可证作为备注。

4. 单击“新建 > 新建目录”，目录命名为“.cloudbuild”。
5. 在“.cloudbuild”目录下依次单击“新建 > 新建文件”，文件命名为“build.yml”，新建后文件目录如**图4-4**所示。

图 4-4 文件目录



若YAML文件不存放在“.cloudbuild”目录，可通过“CB_BUILD_YAML_PATH”参数指定YAML文件在代码仓中的路径。参数配置可参考**添加自定义参数的配置指导**。

6. 单击，参考如下代码示例，编写“build.yml”文件。

   以下示例为使用内置X86、8U16G的执行机，使用Maven构建工具编译构建托管在CodeArts Repo中的代码，并上传软件包至软件发布库。

   不同构建步骤的代码示例，可参考**配置构建任务**中各个构建步骤的“代码化构建”部分。多任务YAML文件结构可参考**多任务YAML文件结构详解**。

```
version: 2.0 # 必须是2.0，该版本号必填且唯一
params: # 构建参数，可在构建过程中引用。如果不填写，则优先使用配置构建任务参数中的构建参数
  - name: paramA
    value: valueA
  - name: paramB
    value: valueB
env: # 定义构建环境信息。非必填，如果不填写，默认使用X86
  resource:
    type: docker # 资源池类型：docker或custom，其中docker表示使用默认执行机，custom表示使用自定义执行机
    arch: X86 # 构建环境主机类型：X86或ARM
    class: 8U16G # 规格：2U8G、4U8G、8U16G、16U32G或16U64G，当type为custom时无需填写该参数
    pool: Mydocker #资源池名称，当type为custom时需要填写该参数
steps:
  PRE_BUILD: # 用于做构建前的准备，例如下载代码，执行shell等
    - checkout:
        name: 代码下载 # 可选
        inputs: # 步骤参数
          scm: codehub # 代码来源:只支持Repo
          url: xxxxxxxxx # 拉取的代码仓的ssh地址
          branch: ${codeBranch} # 拉取的代码分支：支持参数化
    - sh:
        inputs:
          command: echo ${paramA}
  BUILD: # 用于定义构建步骤，仅支持配置一个BUILD
    - maven: # 步骤关键字，仅支持特定关键字
    name: maven build # 可选
    image: xxx # 镜像地址
    inputs:
      command: mvn clean package
```



```
- upload_artifact:
inputs:
      path: "**/target/*.?ar"
```

配置构建任务基本信息

1. 选择导航栏“持续交付 > 编译构建”，进入编译构建服务页面。
2. 单击“新建任务”，进入配置“基本信息”页面，参考**表4-4**填写构建任务基本信息。然后单击“下一步”，进入“构建模板”页面。

表 4-4 基本信息配置说明

参数	说明
名称	创建的编译构建任务名称，可自定义。 ● 支持中英文，数字，下划线“_”和连接符“-”。 ● 字符长度范围为1~115。
所属项目	创建的编译构建任务所属项目。 ● 以项目入口方式访问访问编译构建服务时默认填写，无需手动填写。 ● 以服务入口访问时需根据实际情况选择**新建构建任务前准备工作**中创建的项目。
代码源	选择Repo：表示从代码托管拉取代码进行构建。
代码仓	选择实际需要编译的代码仓。
默认分支	选择仓库默认分支。
描述	可选参数。根据实际场景对编译构建任务的描述。字符长度范围0~512。

选择构建模板

在构建模板页面选择“空白构建模板”。选择后单击“确定”，进入“构建步骤”页面。

说明

使用代码化构建时，选择任何构建模板都不影响使用YAML构建。

配置构建步骤

在“构建步骤”页面左上角单击“代码化”页签，系统会从**配置构建任务基本信息**中配置的代码仓库及分支中，自动读取YAML文件。

您可在此处参考**配置构建任务**中“代码化构建”部分的代码示例对YAML文件进行修改。如果在此处修改了YAML文件，那么执行构建任务后，修改后的内容会覆盖**创建代码化构建使用的YAML文件**中的原YAML文件。

配置完成后，单击“保存”，即可完成构建任务的创建。

多任务 YAML 文件结构详解

在编译构建中，构建任务是构建的最小单元，适用于业务比较简单的场景，但是在有些复杂的构建场景下，构建任务可能并不能满足复杂的构建要求。例如：

- 多仓工程需要分布到多个机器上去构建，并且构建工程之间还存在一定的依赖关系。
- 希望更模块化、更加细粒度的拆分构建任务，并按照依赖顺序进行构建。

对于上述这类比较复杂的构建场景，编译构建支持使用BuildFlow将多个存在依赖关系的构建任务按照有向无环图（DAG）的方式组装起来，BuildFlow将会按照构建的依赖关系并发进行构建。

须知

- BuildFlow父任务不会占用一个并发，子任务在并发数不够时会排队，为了最佳使用效果，建议购买**构建并发包**，构建并发包使用规则及方法请参考**如何使用构建并发包**。
- 构建并发包为租户级别，一个并发包资源同时只能由一个构建任务使用。

多任务YAML文件整体内容示例：

```
version: 2.0 # 必须是2.0，该版本号必填且唯一
params: # 构建参数，可在构建过程中引用
  - name: p
    value: 1
# env和envs配置为非必填项。当用户需要使用条件判断确定使用的主机规格与类型时，选择配置envs
env: # 如果配置，则优先级最高。即在此处定义了主机规格与类型，则不使用构建环境配置中选择的主机类型和规格
  resource:
    type:docker # 资源池类型：docker或custom，其中docker表示使用默认执行机，custom表示使用自定义执行机
    arch:X86 # 构建环境主机类型：X86或ARM
    class:8U16G # 规格：2U8G、4U8G、8U16G、16U32G或16U64G，当type为custom时无需填写该参数
    pool：Mydocker #资源池名称，当type为custom时需要填写该参数
envs:
    - condition: p == 1 # 主机规格与类型的判断条件，满足条件会使用以下主机规格与类型
      resource:
        type: docker
        arch: ARM
    - condition: p == 0 # 主机规格与类型的判断条件，不满足条件则不使用以下主机规格与类型
      resource:
        type: docker
```

```
      arch: X86
# buildflow和buildflows配置二选一。当需要使用条件判断执行的jobs时，选择配置buildflows
buildflow:
 strategy: lazy # 定义buildFlow运行的策略，支持lazy和eager。如果没有定义，默认使用eager模式
   jobs: # 构建任务
     - job: Job3 # 子任务的名称，可自定义
       depends_on: # 定义job的依赖,此处表示Job3依赖Job1和Job2
           - Job1
           - Job2
       build_ref: .cloudbuild/build3.yml # 定义Job在构建过程中需要运行的yaml构建脚本
     - job: Job1
       build_ref: .cloudbuild/build1.yml
     - job: Job2
       build_ref: .cloudbuild/build2.yml
buildflows:
 - condition: p == 1 # 执行任务的判断条件，满足条件会执行以下jos中编排的所有子任务
   jobs: # 定义需要进行编排的任务
     - job: Job1 # 子任务的名称，可自定义
       build_ref: 1.yml # 构建任务在构建过程中需要运行的yaml构建脚本
       params:
         - name: abc
           value: 123
     - condition: p == 1 # 执行子任务的判断条件，满足条件会执行Job2子任务
       job: Job2
       build_ref: 2.yml
       params:
         - name: abc
           value: 123
```

说明

- **lazy**：先触发优先级高的子任务构建，优先级高的子任务执行成功之后，再触发优先级低的子任务。构建时间相对较长，但是可以节省构建资源，推荐在并发数不足时使用。
- **eager**：同步触发所有子任务的构建，有依赖其它任务的子任务会先准备好环境和代码，等待所依赖的任务构建成功。可能造成资源空闲等待，但是可以缩短构建时间，推荐在并发数足够大的情况下使用。

jobs详细介绍：

jobs用来定义需要进行编排的子任务，每个子任务都必须要有唯一的名字作为构建任务的唯一标识。且若子任务A依赖子任务B，则构建优先级B > A，优先级相同的子任务会同步触发。

代码示例如下：

```
  jobs:
   - job: Job3
     depends_on:
         - Job1
         - Job2
     build_ref: .cloudbuild/build3.yml
   - job: Job1
     build_ref: .cloudbuild/build1.yml
   - job: Job2
     build_ref: .cloudbuild/build2.yml
```

如上示例，Job3依赖于Job1和Job2，即构建优先级Job1、Job2 > Job3，且Job1和Job2同步触发。

params详细介绍：

params可以定义全局参数，即所有子任务共享。编译构建也支持在部分子任务上定义参数使用，例如：

```
buildflow:
  jobs:
```



```
- job: Job3
  depends_on:
      - Build Job1
      - Build job2
  build_ref: .cloudbuild/build3.yml
- job: Job1
  params:
    - name: isSubmodule
      value: true
  build_ref: .cloudbuild/build1.yml
- job: Job2
  params:
    - name: isSubmodule
      value: true
  build_ref: .cloudbuild/build2.yml
```

如上示例，未定义全局参数params，而是将参数“isSubmodule”直接定义在Job1与Job2中。

说明

在使用代码化构建时，需注意参数使用的优先级，以上述代码示例为例：

构建任务参数设置中设置的运行时参数 > 构建任务参数设置中的参数默认值 >build_ref中定义的参数 > job下的params中定义的参数 > BuildFlow下params中定义的全局参数。

5 配置构建任务

5.1 构建任务基础配置

5.1.1 配置构建环境

配置构建任务全局运行环境。

编译构建服务支持使用自定义执行机，支持的自定义执行机类型有LINUX、LINUX_DOCKER、WINDOWS和MAC，各个类型支持的构建场景可参考**表5-1**，用户可根据实际需求选择使用的执行机类型。

表 5-1 各个类型执行机的使用说明

执行机类型	使用说明
LINUX	• 执行构建任务时，可通过执行shell命令，在Linux虚拟机上执行构建任务，拥有更高的自由度。 • 在使用编译构建服务前用户需要在自定义执行机上自行安装构建工具，例如Maven、Gradle等。 • 构建步骤仅支持**执行shell命令**、**上传软件包到软件发布库**和**下载发布仓库包**。
LINUX_DOCKER	• 执行构建任务时，编译构建服务将拉起一个Linux Docker容器，构建任务在容器中执行。 • 整个构建过程在容器中运行，运行后容器会自动清理构建镜像，包括构建过程中拉取的代码、过程数据、构建产物等。 • 支持用户宿主机目录与容器目录映射，即可在镜像内共享宿主机目录。 • 除MSBuild构建步骤外，支持所有构建步骤，无需自行安装构建环境。

执行机类型	使用说明
WINDOWS	• 执行构建任务时，构建任务在Windows执行机上执行，支持用户执行Windows相关的构建任务。 • 通过Git Bash工具执行Shell脚本实现构建。 • 构建步骤仅支持**执行shell命令**、**上传软件包到软件发布库**和**下载发布仓库包**。 • 支持Windows7、Windows10、Windows Server2012和Windows Server2016。 • 自定义Windows执行机前，需已安装JDK和Git。 • 编译工具需自行安装。例如：使用Maven构建，则需要安装Maven工具。
MAC	• 执行构建任务时，构建任务在MAC执行机上执行Shell命令，支持用户执行MAC相关的构建任务。 • 构建步骤仅支持**执行shell命令**、**上传软件包到软件发布库**和**下载发布仓库包**。 • 支持当前在使用的所有MAC版本。

图形化构建

CodeArts Build预置了“构建环境配置”步骤，参考**表5-2**配置参数。

表 5-2 构建环境配置参数说明

参数	说明
构建环境主机类型	X86服务器、鲲鹏（ARM）服务器。 **说明** 在不同芯片架构上运行的软件，需要选择对应的环境主机。如软件最终在鲲鹏服务器上运行，则选择鲲鹏服务器。
执行主机	选择用来执行编译构建任务的计算资源。在编译构建服务中，该计算资源为虚拟机。执行主机包括内置执行机和自定义执行机。 • 内置执行机：编译构建服务自身提供的执行主机，用户无需配置即可开箱即用。执行机默认规格为2U8G。 • 自定义执行机：用户自行提供**表5-1**的计算资源，通过注册的方式托管到编译构建服务中，通过编译构建服务进行调度并执行构建任务。 可根据实际情况选择内置执行机或自定义执行机，自定义执行机为在资源池中添加的代理执行机，具体自定义操作可参考**资源池管理**。

参数	说明
宿主机目录与容器目录映射	配置自定义执行机的目录和容器的目录映射，配置映射后，可将自定义执行机中的依赖项等文件挂载到容器中执行构建。当执行主机选择自定义执行时需要配置。 例如：宿主机目录填写“/home”，容器目录填写“/opt”，就会把执行机本地“/home”目录下的内容，挂载到容器内的“/opt”目录下。

代码化构建

参考以下构建环境配置代码示例，修改在**创建代码化构建使用的YAML文件**中的**env**部分代码信息。

```
version: 2.0 # 必须是2.0，该版本号必填且唯一
env: # 定义构建环境信息。非必填，如果不填写，默认使用X86
  resource:
    type:docker # 资源池类型：docker或custom，其中docker表示使用默认执行机，custom表示使用自定义执行机
    arch:X86 # 构建环境主机类型：X86或ARM
    class:8U16G # 规格：2U8G、4U8G、8U16G、16U32G或16U64G，当type为custom时无需填写该参数
    pool：Mydocker #资源池名称，当type为custom时需要填写该参数
```

参考以下BuildSpace代码示例，在**创建代码化构建使用的YAML文件**中添加以下代码信息。

说明

可使用的环境为自定义执行机、构建并发包和构建加速包L3。

```
version: 2.0
buildspace: #表示使用BuildSpace
  fixed: true
  path: kk
  clean: true
  clean_exclude:
    - cache #排除的具体路径
    - aa #排除的具体路径
    - bb #排除的具体路径
```

表 5-3 BuildSpace 代码示例参数说明

参数	类型	说明
fixed	string	可选参数。 在编译构建服务中，默认每一次构建都会使用一个空白的且随机的目录(比如/devcloud/ws/sMMM/workspace/j_X/)作为此次构建的根目录，这个根目录所代表的空间称为BuildSpace。BuildSpace的路径默认是随机的，即使是同一个项目的不同构建任务的BuildSpace也会被随机分配。 但是在某些场景下固定一个BuildSpace的路径是有必要的，因此编译构建服务支持配置BuildSpace，以固定构建执行目录。 ● true：使用固定路径。 ● false：不使用固定路径。 默认值：false。

参数	类型	说明
path	string	可选参数。 当使用固定路径时，路径为：/opt/cloud/slavespace/usr1/+"${domainId}"+/。配置path参数，表示在前面的固定路径基础上拼接路径。 例如：“path”配置路径为“kk”，那么固定路径为：/opt/cloud/slavespace/usr1/+"${domainId}"+/kk。
clean	string	可选参数。 • true：需要清理固定路径。即路径是固定的，但是每次执行完会清理路径下的文件。 • false：不清理固定路径。但是工作空间可存储的容量是有限的，当文件容量达到工作空间上限后，需要手动清理工作空间(clean配置为true即可)。 **说明** • 如果未配置清理固定路径，当文件容量达到工作空间上限后，会自动清理当前租户下的固定路径中所有文件。 • 工作空间指的是用户自定义的执行机的规格。 默认值：true。
clean_exclude	string	可选参数。配置后表示使用路径清理，但是排除配置的路径。仅支持指定固定路径下的一级文件夹。

5.1.2 配置代码下载

配置构建时从代码仓拉取代码的下载方式。

图形化构建

可选择使用指定代码仓库Tag或CommitID构建，同时可选择开启子模块（submodules）自动更新与Git LFS。

预置“代码下载配置”步骤，参考表5-4配置参数。

表 5-4 代码下载配置参数说明

参数	说明
使用指定代码仓库Tag或CommitID构建	配置执行构建任务时是否指定Tag构建或CommitID构建。 • 不指定：拉取全部代码进行构建。 • 指定Tag构建：仅拉取指定Tag的代码进行构建。执行构建任务时需在弹框中输入Tag。 Tag是指代码仓库中的标签。如果代码源选择的是Repo，关于如何创建Tag可参见**标签管理**。如果代码源使用的是第三方代码仓，需在第三方代码仓中创建标签。 • 指定CommitID构建：仅拉取指定CommitID的代码进行构建。执行构建任务时需在弹框中输入CommitID。 CommitID是指提交代码时生成的编号。以Repo代码仓为例，CommitID在代码仓库中显示如**图5-1**。 **图 5-1** CommitID 
克隆深度	可选参数。 克隆深度是指距离最近一次提交的提交次数，该值越大，检出代码的深度越深。深度为正整数，推荐最大深度为25。 例如：克隆深度5就表示只克隆最新5次提交记录以及提交之后的最新内容，不克隆历史提交。
子模块（submodules）自动更新	子模块（submodule）是Git为管理仓库共用而衍生出的一个工具，通过子模块用户可以将公共仓库作为子目录包含到用户的仓库中，并能够双向同步该公共仓库的代码，借助子模块用户能将公共仓库隔离、复用，能随时拉取最新代码以及对它提交修复，能大大提高团队效率。更多详情请参考**配置代码仓库的子模块设置**。 • 开启：当代码仓库存在子模块时，系统在构建时会自动拉取子模块仓库的代码。 • 不开启：系统不会自动拉取子模块仓库的代码。
开启Git LFS	根据需要选择是否开启“Git LFS”，构建默认不拉取音视频、图像等大型文件，开启“Git LFS”后，构建将会全量拉取文件。

代码化构建（单仓下载）

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**PRE_BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
```

```
PRE_BUILD:
  - checkout:
      name: checkout
      inputs:
        scm: codehub # 代码来源:仅支持Repo
        url: xxxxxxxxx # 拉取代码的ssh地址。
        branch: ${codeBranch} # 任何时候都必填，支持参数化
        commit: ${commitId}
        lfs: true
        submodule: true
        depth: 100
        tag: ${tag}
        path: test
```

表 5-5 单仓下载代码示例参数说明

参数	类型	说明
scm	string	填写代码源：当前只支持Repo，如果yaml文件中没配置，则使用构建任务配置的代码仓信息。 默认值：codehub。
url	string	填写拉取代码的代码仓ssh地址。
branch	string	拉取的代码分支。 支持参数化，可使用${codeBranch}调用。
commit	string	可选参数。指定commitId构建时，填写拉取的commitId。 支持参数化，可使用${commitId}调用。
tag	string	可选参数。指定tag构建时，填写拉取的tag。 支持参数化，可使用${tag}调用。如果同时指定commitId和tag，优先执行commitId构建。
depth	int	可选参数。浅克隆深度：当选择commitId构建时，depth必须大于等于commitId所在深度。 默认值：1。
submodule	bool	可选参数。配置是否拉取子模块。 ● true：拉取。 ● false：不拉取。 默认值：false。
lfs	bool	可选参数。配置是否开启git lfs。 ● true：开启。 ● false：不开启。 构建默认不拉取音视频、图像等大型文件，开启git lfs后，构建将会全量拉取文件。默认值：false。
path	string	可选参数。clone的子路径：代码将会下载到子目录下。

代码化构建（manifest 多仓下载）

在安卓、鸿蒙等场景下，一次构建需要同时集成数百甚至上千个代码仓，多个代码仓的集成下载效率至关重要。

编译构建集成Repo下载工具，用户只需进行简单配置即可实现多个代码仓的联动集成。当前支持仅Repo代码仓。

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**PRE_BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  PRE_BUILD:
  - manifest_checkout:
      name: "manifest"
      inputs:
        manifest_url: "https://example.example.example.example.example.com/xx/manifest.git"
        manifest_branch: "master"
        manifest_file: "default.xml"
        path: "dir/dir02"
        repo_url: "https://example.example.example.example.example.com/xx/git-repo.git"
        repo_branch: "master"
        username: "someone"
        password: "${PASSWD}"
```

表 5-6 manifest 多仓下载代码示例参数说明

参数	类型	说明
name	string	可选参数。填写步骤名称。默认值为manifest_checkout。
manifest_url	string	填写manifest仓地址，包含xml文件的仓库。
manifest_branch	string	可选参数。填写manifest分支或revision。默认值为“HEAD”。
manifest_file	string	可选参数。manifest文件路径。默认值为“default.xml”。
path	string	可选参数。自定义manifest所有子仓下载路径，为工作目录的相对路径 路径不能以“/”开头，不能包含“.”。默认为工作目录。
repo_url	string	可选参数。填写repo仓库地址。默认值为“https://gerrit.googlesource.com/git-repo”。
repo_branch	string	可选参数。填写repo仓库分支。默认值为“stable”。
username	string	可选参数。填写下载仓库时使用的用户名，当下载非公开仓库时需填写。
password	string	可选参数。填写下载仓库时使用的https密码，下载非公开仓库时需填写。

说明

1. manifest_file中定义的多个仓库，必须为同一种代码源。
2. manifest_url与manifest_file必须为同一种代码源；如果为非公开仓库，username&password应该有下载权限。
3. repo_url对应的repo仓库，需要有下载权限（仓库开源，或者仓库私有但配置了账号密码）。
4. 以上非必填的参数，如果配置的值为空，则使用默认值。
5. 建议在使用非公开仓库时，用户名密码通过构建的私密参数进行配置，详情参考**配置构建任务参数**。
6. 该功能目前仅支持北京四区域使用，其余区域后续上线。

5.2 选择构建步骤

您可以根据实际使用场景，在构建任务中选择需要使用的构建步骤。

图形化构建

构建步骤页面展示所选模板的默认步骤组合。

- 单击构建步骤上的可根据实际需要添加构建步骤，每个构建步骤的配置指导请参考**配置构建步骤**中“图形化构建”部分。

  若构建步骤中预置的工具版本无法满足使用需求，可以通过**自定义构建环境**自定义环境进行构建。
- 如果需要删除已添加的构建步骤，鼠标左键选中该步骤，依次单击“ ··· > 删除”，即可删除该步骤。
- 如果需要复制已添加的构建步骤，鼠标左键选中该步骤，依次单击“ ··· > 复制”，即可复制该步骤。
- 如果某个构建步骤暂时不使用，可鼠标左键选中该步骤，依次单击“ ··· > 禁用”，可禁用该步骤。需要重新启用时，鼠标左键选中该步骤，依次单击“ ··· > 启用”即可。

图 5-2 添加/复制/删除/禁用构建步骤

代码化构建

- 参考**配置构建环境**的“代码化构建”部分的代码示例，在**创建代码化构建使用的YAML文件**中的“env”部分配置构建任务的运行环境。

- 参考**配置代码下载**的“代码化构建”部分的代码示例，在**创建代码化构建使用的YAML文件**中的“PRE_BUILD”部分配置代码下载方式。
- 参考**配置构建步骤**中各个构建步骤的“代码化构建”部分的代码示例，在**创建代码化构建使用的YAML文件**中的“BUILD”部分配置构建步骤。

5.3 配置构建步骤

5.3.1 使用 Maven 构建

使用Maven构建Java项目。

图形化构建

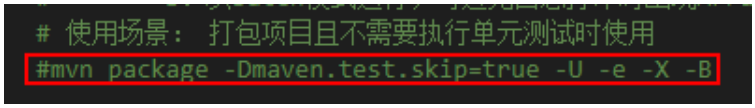
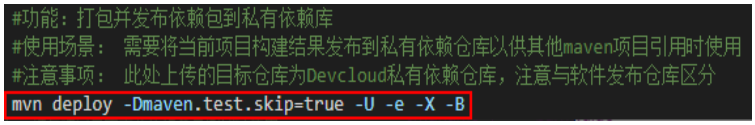
在**配置构建步骤**中，添加“Maven构建”构建步骤，参考**表5-7**配置参数。

表 5-7 Maven 构建步骤参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置Maven命令，一般使用系统默认生成的命令即可。如果需要配置更多命令，可参考**Maven官网**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

参数	说明

setting配置	• 自动生成setting文件并配置依赖仓库：可根据用户的IP不同，自动识别最优站点访问CodeArts提供的“setting.xml”文件。
	国内用户使用“国内站点”，国际用户使用“国际站点”。建议使用默认配置。
	“setting.xml”文件中定义了默认的依赖拉取顺序和镜像源代理等配置信息，如果需要使用自定义的“setting.xml”文件，可**添加自定义setting.xml文件**，然后在默认的打包命令末尾添加**--settings settings.xml**，即可使用已添加的“settings.xml”文件执行Maven构建。
	`# 使用场景： 打包项目且不需要执行单元测试时使用`
	`mvn package -Dmaven.test.skip=true -U -e -X -B --settings settings.xml`
	• 公有依赖仓库：默认已添加华为开源镜像站，同时配置了HuaweiSDK仓库。此配置仅在需要添加非CodeArts提供的公有依赖仓库时使用，添加方法如下：
	1. 单击“添加”。
	2. 填写公有依赖仓库地址，根据需要勾选“release仓库”和“snapshot仓库”。release仓库和snapshot仓库至少勾选一个，也可以同时勾选。
	release仓库：勾选后，构建过程将尝试从仓库中下载release版本依赖。
	snapshot仓库：勾选后，构建过程将尝试从仓库中下载snapshot版本依赖。
	• 私有依赖库：默认已配置CodeArts提供的私有依赖仓库。此配置仅在需要添加其它私有依赖仓库时使用，添加方法如下：
	1. **新建nexus repository服务扩展点**。
	2. 单击“添加”，选择上一步创建的服务扩展点，并根据需要勾选“release仓库”和“snapshot仓库”。
	说明
	“release仓库”和“snapshot仓库”两种仓库对应的使用场景区分如下，使用时要务必注意区分，避免出现如“将依赖上传到软件发布库但是构建时无法下载”此类场景。
	• “snapshot仓库”：对于以调试为目的发布的私有依赖包，一般会给依赖版本号增加-SNAPSHOT后缀（如：1.0.0-SNAPSHOT），执行发布操作时，此类依赖会自动发布到snapshot仓库，发布时无需更新版本号，构建命令中增加-U参数即可拉取最新版本。
	• 对于正式发布的私有依赖包，版本号中不可带-SNAPSHOT后缀（如：1.0.0），执行发布操作时，此类依赖会自动发布到release仓库，发布时必须更新版本号，否则会导致构建过程无法拉取最新依赖包。

参数	说明
发布依赖包到CodeArts私有依赖库	编译构建服务默认使用私有依赖库作为私有依赖下载源，如果需要将构建产物上传至私有依赖库供其他项目依赖使用，则需要添加此配置。配置前，需已**创建私有依赖库**。配置方法如下： • 不配置pom：表示无需发布私有依赖包到CodeArts私有依赖库。 • 配置所有pom：表示在项目下所有“pom.xml”文件增加deploy配置，使用**mvn deploy**命令将构建出的依赖包上传到私有依赖仓库。配置后，需在命令窗口，使用“#”注释命令**mvn package -Dmaven.test.skip=true -U -e -X -B**，如下图：  删除**#mvn deploy -Dmaven.test.skip=true -U -e -X -B**命令前的“#”，如下图：  上传的私有依赖包，在其他项目添加pom.xml文件中的groupId、artifactId、version坐标即可引用。
单元测试	如果用户需要对单元测试结果进行处理，可配置此项。详见**配置单元测试**。
缓存配置	选择是否使用缓存以提高构建速度，选择“使用缓存”后，每次构建时会把下载依赖包缓存起来，后续构建无需重复拉取，可有效提高构建速度。 **说明** maven构建的依赖包存入缓存之后，只有当租户下面构建的项目有引进新的依赖包时，才会更新缓存目录，并不支持对已有的依赖包缓存文件进行更新。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - maven:
        image: cloudbuild@maven3.5.3-jdk8-open
        inputs:
          settings:
            public_repos:
              - https://mirrors.example.com/maven
          cache: true # 是否开启缓存
          unit_test:
            coverage: true
            ignore_errors: false
            report_path: "**/TEST*.xml"
            enable: true
            coverage_report_path: "**/site/jacoco"
          command: mvn package -Dmaven.test.failure.ignore=true -U -e -X -B
          ignore_fail: true
```

表 5-8 代码示例参数说明

参数	类型	说明
image	string	填写镜像地址，有以下两种格式。 • cloudbuild@maven3.5.3-jdk8-open：以cloudbuild开始，@作为分隔符，后面是编译构建提供的默认镜像。 • 完整的swr镜像地址，例如:swr.example.example.com/codeci_test/demo:141d26c455abd6d7xxxxxxxxxxxxxxxxxxx
settings	map	可选参数。不配置，则默认使用CodeArts提供的setting.xml文件，如果需要使用自定义的“settings.xml”文件，可先**添加自定义setting.xml文件**，然后在默认的打包命令**mvn package -Dmaven.test.failure.ignore=true -U -e -X -B**末尾添加**--settings settings.xml**。
cache	bool	可选参数。 配置是否开启缓存。 • true：开启。 • false：不开启。 默认值：false。
command	string	配置执行的Maven命令。如果需要配置更多命令，可参考**Maven官网**。
unit_test	map	可选参数。 配置单元测试。详细操作指导参考**配置单元测试**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

添加自定义“setting.xml”文件

- **图形化构建**
  a. 在“Maven构建”的命令窗口执行**cat /home/build/.m2/settings.xml**命令，任务执行完成后，会在构建日志中展示“settings.xml”文件的内容。
  b. 参考构建日志中的“settings.xml”的信息自定义新的“settings.xml”文件。
  c. 在“Maven构建”步骤前增加“下载文件管理的文件”构建步骤。
     自定义步骤显示名称，工具版本当前仅支持“shell4.2.46-git1.8.3-zip6.00”。
  d. 单击“上传”，在弹出的窗口中选择**b**中自定义的文件，添加描述，勾选相关协议，然后单击“保存”。
  e. 在“下载文件”中选择上传的“setting.xml”文件。
- **代码化构建**

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - download_file:
        inputs:
          name: settings.xml
          ignore_fail: true
```

表 5-9 下载文件管理的文件代码示例参数说明

参数	类型	说明
name	string	setting文件名称。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

说明

- 文件大小限制为100k。
- 文件类型限制为：**.xml、.key、.keystore、.jks、.crt、.pem。**
- 最多支持上传20个文件。

已上传的文件可通过两种访问路径进行文件管理。

- 在编译构建服务首页，单击“更多”，选择“文件管理”。
- 或在“下载文件管理的文件”构建步骤中单击“管理文件”。

在文件管理页面，可以编辑文件、下载文件、删除文件、为用户配置文件操作权限。

- 在搜索框输入关键字，可搜索文件。
- 单击操作列，可修改文件名称，并设置是否允许租户内所有成员在编译构建中使用该文件。
- 单击操作列，可以下载文件。
- 单击操作列，在下拉框中选择“删除”，可根据弹框提示确认是否删除。
- 单击操作列，在下拉框中选择“编辑权限”，可在弹出的界面配置用户操作文件的权限。

图 5-3 配置用户操作文件权限

表 5-10 文件管理角色权限说明

权限类型	拥有该权限的角色
添加用户	项目下所有用户。
查看	文件创建者、相同租户的用户。
使用	文件创建者、文件创建者配置了使用权限的用户。
更新	文件创建者、文件创建者配置了更新权限的用户。
删除	文件创建者、文件创建者配置了删除权限的用户。
编辑权限	文件创建者。

说明

创建者默认有所有权限并且不可被删除和修改。

配置单元测试

1. 配置单元测试前，需要在项目中编写单元测试代码，且需满足如下条件：
   - 单元测试用例代码存放位置需满足Maven默认单元测试用例目录规范及命名规范，或自行在配置中指定用例位置。

     如：单元测试用例统一存放在路径为“src/test/java/{{package}}/”，单元测试将在Maven构建过程自动执行。
   - 项目中不可存在忽略单元测试用例的配置代码，即确保以下代码未存在于项目的“pom.xml”文件中。

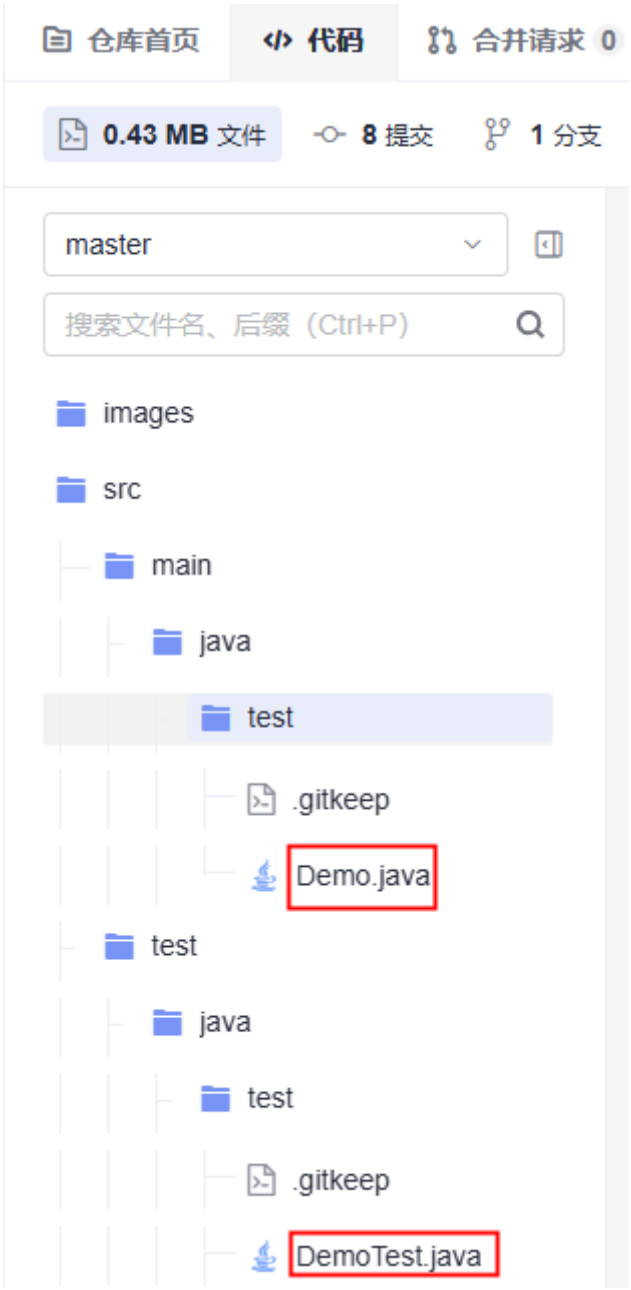
     ```
     <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.18.1</version>
        <configuration>
           <skipTests>true</skipTests>
        </configuration>
     </plugin>
     ```
   - “pom.xml”文件中需引入junit依赖，添加的代码示例如下。

     ```
     <dependency>
         <groupId>junit</groupId>
         <artifactId>junit</artifactId>
         <version>4.13.1</version>
     </dependency>
     ```

     说明

     junit版本需在4.13.1及以上。
2. 在代码仓中创建单元测试类，如图5-4所示。

图 5-4 单元测试文件目录



“Demo.java”文件代码示例如下：

```
package test;

public class Demo {
    public String test(Integer i) {
        switch (i) {
            case 1:
                return "1";
            case 2:
                return "2";
            default:
                return "0";
        }
}
}
```

“DemoTest.java”文件代码示例如下：

```
package test;

import org.junit.Test;

public class DemoTest {
    private Demo demo=new Demo();
    @Test
    public void  test(){
        assert demo.test(1).equals("1");
        assert demo.test(2).equals("2");
        assert demo.test(3).equals("0");
    }
}
```

3. 配置构建步骤中的单元测试。
   - **图形化构建**
     i. 在“Maven构建”步骤的命令窗口，使用“#”注释命令**mvn package -Dmaven.test.skip=true -U -e -X -B**。

     ```
     # 使用场景： 打包项目且不需要执行单元测试时使用
     #mvn package -Dmaven.test.skip=true -U -e -X -B
     ```

     ii. 删除**#mvn package -Dmaven.test.failure.ignore=true -U -e -X -B**命令前的“#”。

     ```
     #功能：打包;执行单元测试，但忽略单元测试用例失败，每次构建检查依赖更新
     #使用场景： 需要执行单元测试，且使用构建提供的单元测试报告服务统计执行情况
     # 使用条件：在”单元测试“中选择处理单元测试结果，并正确填写测试结果文件路径
     mvn package -Dmaven.test.failure.ignore=true -U -e -X -B
     ```

     iii. 展开“单元测试”，根据实际需求参考**表5-11**配置参数。

表 5-11 单元测试配置参数说明

参数	说明
是否处理单元测试结果	配置是否处理单元测试结果。 ● 是：处理。 ● 否：不处理。
是否忽略用例失败	当选择处理单元测试结果时，需要配置是否忽略用例失败。 ● 是：用例失败时构建任务仍然继续。 ● 否：用例失败时构建任务也失败。
单元测试结果文件	测试报告需要采集单元测试结果用以生成可视化报告，需在此处填写单元测试结果文件存放路径。 多数情况下，保留默认路径“**/TEST*.xml”即可满足任务需求。为增加结果准确性，可根据实际情况制定精确的报告路径，如：“target/surefire-reports/TEST*.xml”。

参数	说明
是否处理单元测试覆盖率结果	根据实际需要配置“是否处理单元测试覆盖率结果”，若选“是”，会生成覆盖率测试报告。配置方法请参见**使用JaCoCo生成单元测试覆盖率报告**。
单元测试覆盖率报告路径	当选择处理单元测试覆盖率结果时，需填写相对于项目根目录的相对路径，如：target/site/jacoco，选择处理单元测试覆盖率结果后，会将此目录下的所有文件进行打包上传。

- **代码化构建**

  参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - maven:
          unit_test:
            coverage: true
            ignore_errors: false
            report_path: "**/TEST*.xml"
            enable: true
            coverage_report_path: "**/site/jacoco"
          command: mvn package -Dmaven.test.failure.ignore=true -U -e -X -B
```

表 5-12 单元测试参数说明

参数	类型	说明
enable	bool	可选参数。 配置是否处理单元测试结果。 ● true：处理。配置“true”，需要mvn命令末尾增加**-Dmaven.test.failure.ignore=true**参数。 ● false：不处理。 默认值：true。
ignore_errors	bool	可选参数。 配置是否忽略用例失败。 ● true：忽略，用例失败时构建任务仍然继续。 ● false：不忽略，用例失败时构建任务也失败 默认值：true。
report_path	String	填写单元测试数据存储路径。可根据实际情况制定精确的报告路径，如：“target/surefire-reports/TEST*.xml”。

参数	类型	说明
coverage	bool	可选参数。 配置是否处理覆盖率数据。若配置“true”，配置方法请参见**使用JaCoCo生成单元测试覆盖率报告**。 ● true：处理。 ● false：不处理。 默认值：false。
coverage_report_path	string	可选参数。 当选择处理单元测试覆盖率结果时，需填写相对于项目根目录的相对路径，如：target/site/jacoco，选择处理单元测试覆盖率结果后，会将此目录下的所有文件进行打包上传。

4. 配置完成后，如果构建任务执行成功，即可在任务执行详情页面的“测试”页签查看测试报告。如果选择了处理单元测试覆盖率报告，会生成覆盖率测试报告，单击“覆盖率报告下载”即可下载。

配置使用 JaCoCo 生成单元测试覆盖率报告

配置单元测试时，如果选择处理单元测试覆盖率结果，则需要按照如下指导进行配置。

- **单模块项目配置方法**

  在项目中已添加jacoco-maven-plugin插件用于生成单元覆盖率报告，即在“pom.xml”文件中添加如下配置：

  JaCoCo的report目标默认是在verify阶段，这里需要将report目标定义为test阶段。

```
<plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.8.5</version>
    <executions>
        <execution>
            <goals>
                <goal>prepare-agent</goal>
            </goals>
        </execution>
        <execution>
            <id>report</id>
            <phase>test</phase> #定义report目标的阶段
            <goals>
                <goal>report</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

- **多模块项目配置方法**

  假设多模块项目代码结构如下，以此为例为您介绍如何配置生成单元测试覆盖率报告。

```
├── module1
│   └── pom.xml
├── module2
│   └── pom.xml
```



```
├── module3
│   └── pom.xml
├── pom.xml
```

a. 在项目下添加一个用来聚合的模块，自定义名称如：report，添加聚合模块后的代码结构如下：

```
├── module1
│   └── pom.xml
├── module2
│   └── pom.xml
├── module3
│   └── pom.xml
├── report
│   └── pom.xml
├── pom.xml
```

b. 在项目根目录的“pom.xml”文件添加jacoco-maven-plugin插件，代码示例如下。

```
<!-- 配置单元测试覆盖率-->
 <plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.8.3</version>
    <executions>
       <execution>
          <goals>
             <goal>prepare-agent</goal>
          </goals>
       </execution>
    </executions>
 </plugin>
```

c. 配置聚合模块的“pom.xml”文件。

以dependency形式引入所有依赖模块，并使用report-aggregate定义JaCoCo聚合目标。

```
<dependencies>
    <dependency>
       <groupId>${project.groupId}</groupId>
       <artifactId>module1</artifactId>
       <version>${project.version}</version>
    </dependency>
    <dependency>
       <groupId>${project.groupId}</groupId>
       <artifactId>module2</artifactId>
       <version>${project.version}</version>
    </dependency>
    <dependency>
       <groupId>${project.groupId}</groupId>
       <artifactId>module3</artifactId>
       <version>${project.version}</version>
    </dependency>
 </dependencies>

 <build>
    <plugins>
       <plugin>
          <groupId>org.jacoco</groupId>
          <artifactId>jacoco-maven-plugin</artifactId>
          <version>0.8.3</version>
          <executions>
             <execution>
                <id>report-aggregate</id>
                <phase>test</phase>
                <goals>
                   <goal>report-aggregate</goal>
                </goals>
             </execution>
          </executions>
       </plugin>
```

```
    </plugins>
</build>
```

d. 配置完成后，在项目根目录下执行**mvn test**，执行成功后会在“report/target/site/jacoco-aggregate”目录下生成各个模块的覆盖率报告。也可以在outputDirectory中自定义报告的输出路径：

```
<plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.8.3</version>
    <executions>
        <execution>
            <id>report-aggregate</id>
            <phase>test</phase>
            <goals>
                <goal>report-aggregate</goal>
            </goals>
            <configuration>
                <outputDirectory>target/site/jacoco</outputDirectory>
            </configuration>
        </execution>
    </executions>
</plugin>
```

关于使用Maven构建时的常见问题，可参考**使用Maven构建时常见问题**。

5.3.2 使用 Android 构建

Android构建用于编译应用资源和源代码，并将它们打包成可供部署、签署和分发的APK。

图形化构建

1. 在**配置构建步骤**中，添加“Android构建”构建步骤，参考**表5-13**配置参数。

表 5-13 Android 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
Gradle	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
JDK	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
NDK	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置Gradle命令，一般使用系统默认给出的命令即可。如果需要配置更多命令，可参考**Gradle官网**。

参数	说明
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

2. 如果需要使用apksigner对Android APK进行签名，可添加“Android APK签名”构建步骤，参数说明如下：

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
需要签名的APK路径	Android构建后生成要签名的“.apk”文件位置。 支持正则表达式，如：可以使用“build/bin/*.apk”匹配构建出来的APK包。
Keystore文件	在下拉框中选择用于签名的Keystore文件。文件的制作以及上传指导可参考**生成并上传Keystore签名文件**。
keystore password	可选参数。 填写自定义的密钥文件密码。
别名（Alias）	自定义密钥别名。 ● 以字母开头，支持字母、数字、“_”、“-”和“.”。 ● 字符长度为1~128。
key password	可选参数。 填写自定义的密钥密码。
apksigner命令行	自定义签名的参数，默认“--verbose”显示签名详情。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

配置完成后执行构建任务，当显示任务执行成功后，查看构建日志，若“Android APK签名”步骤对应日志中显示结果“Signed”，表示签名成功。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

1. Android构建代码示例如下：

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - android:
```



```
    inputs:
      gradle: 4.8
      jdk: 1.8
      ndk: 17
      command: |
        cat ~/.gradle/init.gradle
        cat ~/.gradle/gradle.properties
        cat ~/.gradle/init_template.gradle
        rm -rf ~/.gradle/init.gradle
        rm -rf /home/build/.gradle/init.gradle
        # 使用CodeArts Build提供的gradle wrapper,充分利用缓存加速
        cp /cache/android/wrapper/gradle-wrapper.jar ./gradle/wrapper/gradle-wrapper.jar
        # 构建未签名的APK
        /bin/bash ./gradlew assembleDebug -Dorg.gradle.daemon=false -d --stacktrace
      ignore_fail: true
```

表 5-14 Android 构建代码示例参数说明

参数名称	参数类型	参数说明
command	string	填写Gradle执行命令。如果需要配置更多命令，可参考**Gradle官网**。
gradle	string	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
jdk	string	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
ndk	string	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

2. Android APK签名代码示例如下：

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - android_sign:
        inputs:
          file_path: build/bin/*.apk
            keystore_file: androidapk.jks
            keystore_password: xxxxxx
            alias: keyalias
            key_password: xxxxxx
            apksigner_commond: --verbose
            ignore_fail: true
```


表 5-15 Android APK 签名代码示例参数说明

参数名称	参数类型	参数说明
file_path	string	Android构建后生成要签名的“.apk”文件位置。 支持正则表达式，如：可以使用“build/bin/*.apk”匹配构建出来的APK包。
keystore_file	string	Keystore文件名。文件的制作以及上传指导可参考**生成Keystore签名文件并上传至文件管理**。
keystore_password	string	可选参数。 填写自定义的密钥文件密码。
alias	string	密钥别名。 ● 以字母开头，支持字母、数字、“_”、“-”和“.”。 ● 字符长度为1~128。
key_password	string	可选参数。 填写自定义的密钥密码。
apksigner_commond	string	自定义签名的参数，默认“--verbose”显示签名详情。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 ● true：是。 ● 为空：否。

Android 版本说明

- SDK：用户项目构建compileSdkVersion版本。
- Build Tools：用户项目构建所需buildToolsVersion版本。

两个版本可以在项目下的“build.gradle”文件或是项目的全局配置文件（用户自定义）中找到。

app/build.gradle 大小: 951 bytes

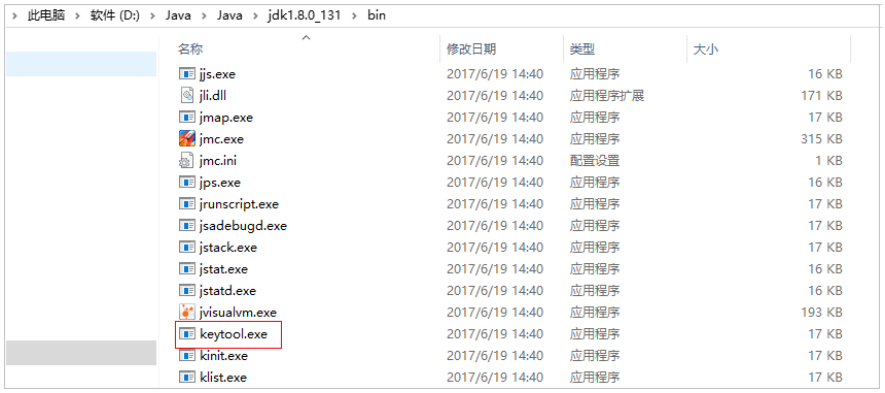
```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 24
    buildToolsVersion '25.0.0'
    defaultConfig {
        applicationId "cn.bluemobi.dylan.step"
        minSdkVersion 11
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    lintOptions {
        abortOnError false
    }
}

```

说明

- 用户需要选择正确的compileSdkVersion版本和buildToolsVersion版本。
- 也支持Gradle的wrapper构建方式，如果提供的gradle版本没有满足您的要求，您也可以直接使用gradlew命令，使用wrapper去构建，会自动下载您所需要的gradle版本，构建命令例如：**./gradlew clean build**。

生成 Keystore 签名文件并上传至文件管理

1. Keystore签名文件有以下两种生成方式。
   - **使用JDK的keytool工具生成签名文件**
     i. 找到JDK安装位置以及keytool.exe。

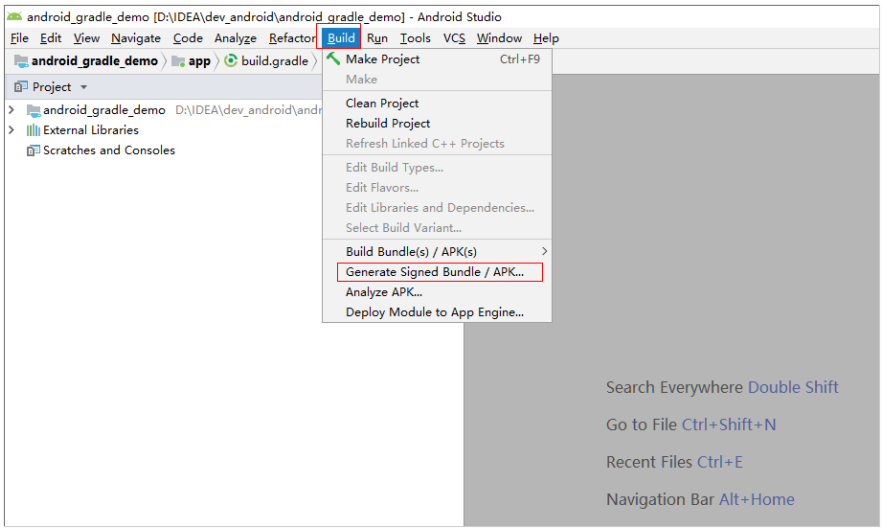


     ii. 执行生成密钥命令，生成.jks文件。

```
keytool -genkeypair -storepass 123456 -alias apksign -keypass 123456 -keyalg RSA -validity 20000 -keystore D:/android.jks
```

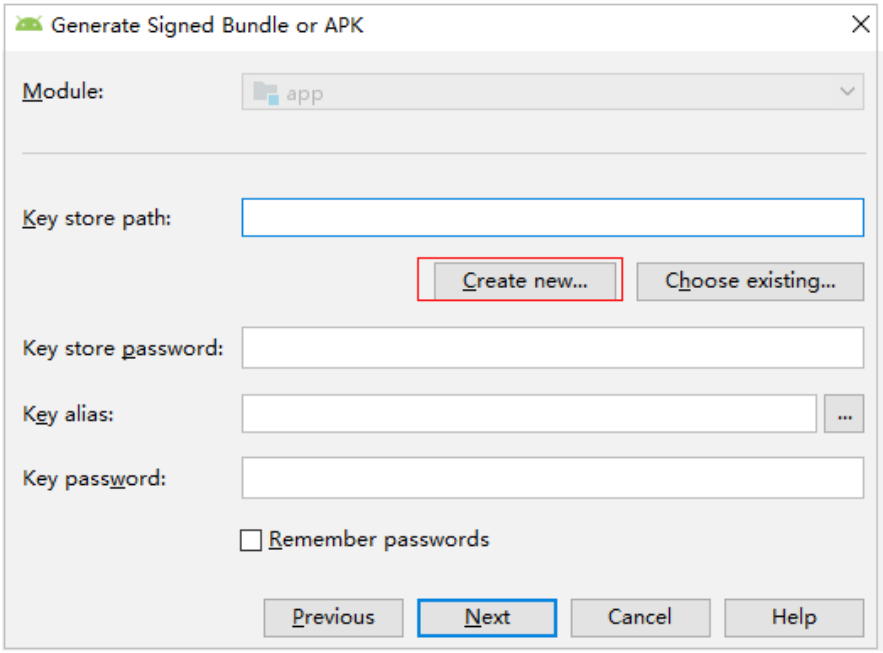
- **使用Android Studio生成签名文件**

  i. 打开Studio客户端，选择“Build > Generate Signed Bundle/APK”。

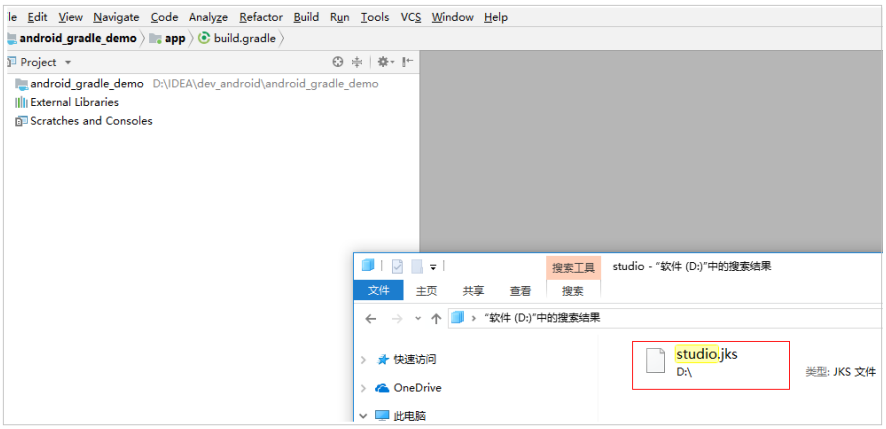


  ii. 选择“APK”，单击“Next”。

  iii. 单击“Create new...”，在弹出框填写相关信息，单击“OK”，然后单击“Next”。



iv. 签名文件成功生成，查看文件。



2. 上传“Keystore签名文件”到文件管理，分为两种上传方式。
   - 在“Android构建”构建步骤的“Keystore文件”处，单击“上传”，在弹出的窗口中选择文件，添加描述，勾选相关协议，然后单击“保存”。
   - 在编译构建服务首页，单击“更多 > 文件管理”，单击“上传文件”，在弹出的窗口中选择文件，添加描述，勾选相关协议，然后单击“保存”。

   在文件管理页面，可以编辑文件、下载文件、删除文件、为用户配置文件操作权限。
   - 在搜索框输入关键字，可搜索文件。
   - 单击操作列，可修改文件名称，并设置是否允许租户内所有成员在编译构建中使用该文件。
   - 单击操作列，可以下载文件。
   - 单击操作列，在下拉框中选择“删除”，可根据弹框提示确认是否删除。
   - 单击操作列，在下拉框中选择“编辑权限”，可在弹出的界面配置用户操作文件的权限。

图 5-5 配置用户操作文件权限

表 5-16 文件管理角色权限说明

权限类型	拥有该权限的角色
添加用户	项目下所有用户。
查看	文件创建者、相同租户的用户。

权限类型	拥有该权限的角色
使用	文件创建者、文件创建者配置了使用权限的用户。
更新	文件创建者、文件创建者配置了更新权限的用户。
删除	文件创建者、文件创建者配置了删除权限的用户。
编辑权限	文件创建者。

说明

创建者默认有所有权限并且不可被删除和修改。

5.3.3 使用 Npm 构建

使用Npm工具管理软件包，可以完成vue和webpack的构建。

图形化构建

在**配置构建步骤**中，添加“Npm构建”构建步骤，可参考**表5-17**配置参数。

表 5-17 Npm 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置Npm命令，一般使用系统默认生成的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。更多命令使用方法可参考**Node.js官网**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
```



```
BUILD:
 - npm:
     image: cloudbuild@nodejs8.11.2
     inputs:
       command: |
          export PATH=$PATH:~/.npm-global/bin
          npm config set registry https://repo.example.com/repository/npm/
          npm config set disturl https://repo.example.com/nodejs
          npm config set sass_binary_site https://repo.example.com/node-sass/
          npm config set phantomjs_cdnurl https://repo.example.com/phantomjs
          npm config set chromedriver_cdnurl https://repo.example.com/chromedriver
          npm config set operadriver_cdnurl https://repo.example.com/operadriver
          npm config set electron_mirror https://repo.example.com/electron/
          npm config set python_mirror https://repo.example.com/python
          npm config set prefix '~/.npm-global'
          npm install --verbose
          npm run build
       ignore_fail: true
```

表 5-18 代码示例参数说明

参数	类型	说明
image	string	填写镜像地址，有以下两种格式。 • cloudbuild@nodejs8.11.2：以cloudbuild开始，@作为分隔符，后面是CodeArts Build提供的默认镜像版本。 • 完整的swr镜像地址，例如:swr.example.example.com/codeci_test/demo:141d26c455abd6d7xxxxxxxxxxxxxxxxxxxx
command	string	配置Npm命令。更多命令使用方法可参考**Node.js官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.4 使用 Gradle 构建

使用Gradle构建工具可以构建Java，Groovy和Scala项目。

图形化构建

在**配置构建步骤**中，添加“Gradle构建”构建步骤，可参考**表5-19**配置参数。

表 5-19 Gradle 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。

参数	说明
Gradle	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
JDK	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置Gradle命令，一般使用系统默认生成的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。更多命令使用方法可参考**Gradle官网**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - gradle:
        inputs:
          gradle: 4.8
          jdk: 1.8
          command: |
            # 使用CodeArts提供的gradle wrapper,充分利用缓存加速
            cp /cache/android/wrapper/gradle-wrapper.jar ./gradle/wrapper/gradle-wrapper.jar
            # 构建未签名的APK
            /bin/bash ./gradlew build --init-script  ./.codeci/.gradle/init_template.gradle -
Dorg.gradle.daemon=false -Dorg.gradle.internal.http.connectionTimeout=800000
          ignore_fail: true
```

表 5-20 代码示例参数说明

参数	类型	说明
command	string	配置Gradle命令。更多命令使用方法可参考**Gradle官网**。
gradle	string	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
jdk	string	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。

参数	类型	说明
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.5 使用 Yarn 构建

使用Yarn可以构建JavaScript工程。

图形化构建

在**配置构建步骤**中，添加“Yarn构建”构建步骤，可参考**表5-21**配置参数。

表 5-21 Yarn 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置Yarn命令，一般使用系统默认生成的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。更多命令使用方法可参考**Yarn官网**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - yarn:
        inputs:
          command: |-
    #nodejs 版本小于18时，可以设置下面的值
                      npm config set cache-folder /yarncache
                      npm config set registry http://mirrors.tools.huawei.com/npm/
                      npm config set disturl http://mirrors.tools.huawei.com/nodejs
                      npm config set sass_binary_site http://mirrors.tools.huawei.com/node-sass/
```



```
                npm config set phantomjs_cdnurl http://mirrors.tools.huawei.com/phantomjs
                npm config set chromedriver_cdnurl http://mirrors.tools.huawei.com/chromedriver
                npm config set operadriver_cdnurl http://mirrors.tools.huawei.com/operadriver
                npm config set electron_mirror http://mirrors.tools.huawei.com/electron/
                npm config set python_mirror http://mirrors.tools.huawei.com/python

                #nodejs 版本大于等于18时，可以设置下面的值
                #npm config set registry http://mirrors.tools.huawei.com/npm/
                npm config set prefix '~/.npm-global'
                export PATH=$PATH:~/.npm-global/bin
                #yarn add node-sass-import --verbose
                yarn install --verbose
                yarn run build
                tar -zcvf demo.tar.gz ./**
    ignore_fail: true
```

表 5-22 代码示例参数说明

参数	类型	说明
command	string	配置Yarn命令。更多命令使用方法可参考**Yarn官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.6 使用 gulp 构建

使用gulp工具可以构建前端集成开发环境。

图形化构建

在**配置构建步骤**中，添加“gulp构建”构建步骤，可参考**表5-23**配置参数。

表 5-23 gulp 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置gulp命令，一般使用系统默认生成的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。更多命令使用方法可参考**gulp官网**。

参数	说明
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - gulp:
        inputs:
          command: |-
                export PATH=$PATH:~/.npm-global/bin
                npm config set registry http://mirrors.tools.huawei.com/npm/
                npm config set prefix '~/.npm-global'
                #如需安装node-sass
                #npm config set sass_binary_site https://repo.huaweicloud.com/node-sass/
                #npm install node-sass
                #加载依赖
                npm install -verbose
                gulp
          ignore_fail: true
```

表 5-24 代码示例参数说明

参数	类型	说明
command	string	配置gulp命令。更多命令使用方法可参考**gulp官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.7 使用 Grunt 构建

使用Grunt可以构建JavaScript工程。

图形化构建

在**配置构建步骤**中，添加“Grunt构建”构建步骤，参考**表5-25**配置参数。

表 5-25 Grunt 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置Grunt命令，一般使用系统默认生成的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。更多命令使用方法可参考**Grunt官网**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - grunt:
        inputs:
          command: |-
                  npm config set registry http://7.223.219.40/npm/
                  #npm cache clean -f
                  #npm audit fix --force
                  npm install --verbose
                  grunt
                  npm run build
          ignore_fail: true
```

表 5-26 代码示例参数说明

参数	类型	说明
command	string	配置Grunt命令。更多命令使用方法可参考**Grunt官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 ● true：是。 ● 为空：否。

5.3.8 使用 mono 构建

使用mono可以完成msbuild和dotnet构建。

图形化构建

在**配置构建步骤**中，添加“mono”构建步骤，参考**表5-27**配置参数。

表 5-27 mono 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置mono命令，一般使用界面上系统默认生成的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - mono:
        inputs:
          command: |
            nuget sources Disable -Name 'nuget.org'
            nuget sources add -Name 'xxcloud' -Source 'https://repo.xxcloud.com/repository/nuget/v3/
index.json'
            nuget restore
            msbuild /p:OutputPath=../buildResult/Release/bin
            zip -rq ./archive.zip ./buildResult/Release/bin/*
          ignore_fail: true
```

表 5-28 代码示例参数说明

参数	类型	说明
command	string	配置mono执行命令。

参数	类型	说明
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 ● true：是。 ● 为空：否。

5.3.9 使用 PHP 构建

使用PHP构建，可以为项目所依赖的PHP代码库提供安装和打包环境。

图形化构建

在**配置构建步骤**中，添加“PHP构建”构建步骤，参考**表5-29**配置参数。

表 5-29 PHP 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置PHP命令，一般使用系统默认生成的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。更多命令使用方法可参考**PHP官网**。 **说明** 第3行默认命令中“http://mirrors.huaweicloud.com/repository/php/”为官网仓库地址，如果用户访问不了该地址会导致构建失败，需替换成用户可以访问的仓库地址。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - php:
        inputs:
          command: |-
                    composer config -g secure-http false
```



```
                composer config -g repo.packagist composer http://mirrors.tools.huawei.com/php/
                composer install
                tar -zcvf php-composer.tgz *
    ignore_fail: true
```

表 5-30 代码示例参数说明

参数	类型	说明
command	string	配置PHP命令。更多命令使用方法可参考**PHP官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.10 使用 SetupTool 构建

使用SetupTool工具可以将Python应用打包。

前提准备

使用SetupTool打包时，需要代码根目录下存在“setup.py”文件，关于setup文件写法请参见**Python官方说明**。

图形化构建

在**配置构建步骤**中，添加“SetupTool构建”构建步骤，参考**表5-31**配置参数。

表 5-31 SetupTool 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置构建打包命令。 • 可以使用默认的命令打包为“egg”格式的文件。 • Python2.7后建议使用**python setup.py sdist bdist_wheel**，打包为源码包和whl格式的安装包，以便使用pip安装。 更多命令使用方法可参考**SetupTool官网**。

参数	说明
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - python:
        name: SetupTool构建
        image: cloudbuild@python3.6
        inputs:
          command: |
            pip config set global.index-url https://pypi.org/simple
            pip config set global.trusted-host repo.xxcloud.com
            python setup.py bdist_egg
          ignore_fail: true
```

表 5-32 代码示例参数说明

参数	类型	说明
name	string	可选参数。 构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。
image	string	可选参数。 填写镜像版本，“cloudbuild@”为固定部分，后面为支持的Python版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。 默认值：cloudbuild@python3.6。
command	string	配置构建打包命令。 • 可以使用默认的命令打包为“egg”格式的文件。 • Python2.7后建议使用**python setup.py sdist bdist_wheel**，打包为源码包和whl格式的安装包，以便使用pip安装。 更多命令使用方法可参考**SetupTool官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.11 使用 PyInstaller 构建

使用PyInstaller工具可以将Python脚本打包成独立的可执行文件。

图形化构建

在**配置构建步骤**中，添加“PyInstaller构建”构建步骤，参考**表5-33**配置参数。

表 5-33 PyInstaller 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置构建打包命令，默认命令是将项目打包成一个可执行文件。更多命令使用方法可参考**PyInstaller官网**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - python:
        name: PyInstaller构建
        image: cloudbuild@python3.6
        inputs:
          command: |
            pip config set global.index-url https://pypi.org/simple
            pip config set global.trusted-host repo.xxcloud.com
            # -F创建单一的可执行文件，文件位置在dist目录下
            # 详细命令参见： https://pyinstaller.readthedocs.io/en/stable/usage.html
            pyinstaller -F  *.py
          ignore_fail: true
```


表 5-34 代码示例参数说明

参数	类型	说明
name	string	可选参数。 构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。
image	string	可选参数。 填写镜像版本，“cloudbuild@”为固定部分，后面为支持的Python版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。 默认值：cloudbuild@python3.6。
command	string	配置构建打包命令，默认命令是将项目打包成一个可执行文件。更多命令使用方法可参考**PyInstaller官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.12 使用 shell 命令构建

单独使用“执行shell命令”步骤可以创建一个构建任务并执行构建。也可以和其他构建工具组合使用，比如，在Maven构建中，增加“执行shell命令”步骤，用于创建后续构建过程中需要使用的文件。

图形化构建

在**配置构建步骤**中，添加“执行shell命令”构建步骤，参考**表5-35**配置参数。

表 5-35 执行 shell 命令参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	请根据需要填写执行构建的shell命令。

参数	说明
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码样例，在**创建代码化构建使用的YAML文件**中的**PRE_BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  PRE_BUILD:
    - sh:
        inputs:
          command: echo ${a}
          ignore_fail: true
```

表 5-36 代码示例参数说明

参数	类型	说明
command	string	请根据需要填写执行构建的shell命令。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.13 使用 Gnu-arm 构建

使用Gnu-arm构建可以为ARM架构的处理器编译和构建软件。

图形化构建

在**配置构建步骤**中，添加“Gnu-arm构建”构建步骤，参考**表5-37**配置参数。

表 5-37 Gnu-arm 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。

参数	说明
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。

命令	配置Gnu-arm构建命令，一般使用系统默认给出的make命令即可。
	• 如果Makefile不在代码根目录下，用户需要使用**cd**命令进入到正确的目录，再使用**make**命令。
	• 用户不使用**make**命令，可以参考下列镜像自带的编译命令：
	– **可选:** gnuarm201405镜像：使用**arm-none-linux-gnueabi-gcc**命令，如下。
	`arm-none-linux-gnueabi-gcc -o main main.c`
	– gnuarm-linux-gcc-4.4.3镜像：使用**arm-linux-gcc**命令，如下。
	`arm-linux-gcc -o main main.c`
	– gnuarm-7-2018-q2-update镜像：使用**arm-none-eabi-gcc**命令，如下：
	`arm-none-eabi-gcc --specs=nosys.specs -o main main.c`
	说明
	• Linux下的GNU的makefile编写，请参见**官网**。
	• 注意Makefile只有行注释“#”，如果要使用或者输出“#”字符，需要进行转义，如使用“\#”。

失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - gnu_arm:
        inputs:
          command: make
          ignore_fail: true
```


表 5-38 代码示例参数说明

参数	类型	说明

command	string	填写Gnu-arm构建命令。
		● 如果Makefile不在代码根目录下，用户需要使用**cd**命令进入到正确的目录，再使用**make**命令。
		● 用户不使用**make**命令，可以参考下列镜像自带的编译命令：
		– **可选:** gnuarm201405镜像：使用**arm-none-linux-gnueabi-gcc**命令，如下。
		`arm-none-linux-gnueabi-gcc -o main main.c`
		– gnuarm-linux-gcc-4.4.3镜像：使用**arm-linux-gcc**命令，如下。
		`arm-linux-gcc -o main main.c`
		– gnuarm-7-2018-q2-update镜像：使用**arm-none-eabi-gcc**命令，如下：
		`arm-none-eabi-gcc --specs=nosys.specs -o main main.c`
		说明
		● Linux下的GNU的makefile编写，请参见**官网**。
		● 注意Makefile只有行注释“#”，如果要使用或者输出“#”字符，需要进行转义，如使用“\#”。

ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 ● true：是。 ● 为空：否。

5.3.14 使用 Msbuild 构建

编译构建服务提供了常用的构建模板（构建环境），不同的构建模板中预装了对应构建所需工具集，MSBuild构建镜像一般预装了msbuild、nuget、.NET Framework等常用工具。

使用msbuild构建工具执行引擎、构造工程，支持.NET框架构建，包含.NET Core和.NET Frameworks。

说明

- Msbuild构建仅支持图形化构建。
- Msbuild构建仅可单独使用，如果构建任务中已有其他构建步骤，将无法添加“Msbuild构建”。

Msbuild 构建场景

表 5-39 已支持的场景

场景类型	说明
无外部依赖	参考镜像版本及对应工具版本，对于仅使用了环境预装依赖库的项目，选择合适的镜像版本即可直接使用**msbuild**或**.NET**命令进行构建。 例如：项目使用了dotnetframework4.7.2的SDK和Office操作的相关官方依赖库（MSOffice）。可选用“msbuild15-dotnetframework4.7.2”版本镜像，使用**msbuild**命令构建。
使用Nuget进行依赖管理	对于使用了环境预装依赖库以外的项目，但使用了Nuget对**所有依赖库**进行管理的项目，选择合适的镜像版本后，可先使用**nuget restore**命令下载所有依赖，此后使用**msbuild**命令进行构建。**.NET**命令无需先执行**nuget**命令。 例如：项目使用了“dotnetframework4.7.2”的SDK，依赖了Myget上某Package并使用Nuget添加了该依赖。可选用“msbuild15-dotnetframework4.7.2”版本镜像，使用**nuget restore && msbuild**命令构建。
其他	对于有其他命令需求的项目，如Git、JDK、Nant、Nodejs等，使用对应的命令进行操作。

表 5-40 未支持的场景

场景类型	说明
未使用Nuget管理依赖库	依赖了本地安装的依赖库，且没有使用nuget对依赖进行管理。详细解决方案请参见**找不到程序集（**.dll）**。** 例如：某项目使用“dotnetframework4.7.2”的SDK，本地安装了Nunit依赖库，但没有使用Nuget对其进行管理。 此时使用**msbuild**命令对其进行构建时会出现找不到库的错误，导致构建失败。
解决方案版本低于VS2015（不包含）	对于使用VS2015（不含）以前版本创建的解决方案，会出现版本过低不兼容的情况，导致构建失败。请尝试升级解决方案。

约束与限制

为方便使用，CodeArts Build提供的Msbuild构建环境原则上尽可能保持与本地环境一致，但因Windows系统与CodeArts Build本身一些系统限制，少数场景下可能会导致构建失败。使用前建议仔细阅读以下规格说明。

- **不支持带空格的文件路径**

  C#项目中目录或文件名中包含空格会导致构建失败，目录/文件的命名请使用字母、数字、下划线的组合，勿使用其他特殊字符，避免不必要的构建失败。

- **文件全路径长度不得超过260个字符**
  - Windows系统中，文件全路径的最大长度限制为260个字符，超过此长度会导致Msbuild构建失败。
  - CodeArts Build需在指定目录下执行命令，即您的构建场景实际与以下步骤类似：

    ```
    cd C:\编译构建的默认路径\您的项目路径
    msbuild
    ```

    说明

    - 项目文件全路径长度实际为项目下文件相对路径长度与编译构建服务默认路径长度之和。
    - 编译构建服务默认路径长度为45字符。因此，在使用Msbuild构建的过程中，您的项目文件路径需满足：项目下文件相对路径（以代码仓库为根目录）长度不可大于215字符。
    - 一些特殊场景（如构建时指定输出目录为“Output/release”)下，可能会额外占用路径长度。

      建议您的项目下文件相对路径（以代码仓库为根目录）长度保持在200个字符以下，原则上尽可能短。
- **不可直接引用系统不具备的组件**

  部分场景下，解决方案中可能不使用NuGet等管理工具，直接引用默认路径下的程序集。但构建时环境中不具备此程序集，导致编译告警，如果项目代码中使用了此引用，甚至会直接导致失败。

  由于Windows系统特殊性，通常情况下，此类程序集默认安装于本地系统，无需指定程序集位置，VS构建时会从默认配置的几个程序集路径查找，可以构建成功。而云端构建环境对应目录无此程序集，进而导致云端构建环境与本地不一致带来的失败。

  为解决此类场景，Msbuild集成了NuGet，可以在构建时从远程仓库下载对应程序集，此时只需于项目中指定“packages.config”，并于其中声明依赖的程序集即可。

  特殊情况下，项目引用的程序集可能无法在远程仓库找到，此时需要手工保存程序集至代码仓库中，并显式指定程序集路径。

  详细解决方案请参见**找不到程序集（**.dll）**。
- **命令行中路径分隔符使用'/'而非'\'**

  部分场景下，可能需要在命令行中使用路径参数，此处需注意，Msbuild构建环境要求路径分隔符统一使用“ / ”格式。

  错误示例：

  ```
  cd test\test1或cd test\\test1
  ```

  正确示例：

  ```
  cd test/test1
  ```
- **避免直接指定低版本SDK路径**
  - CodeArts Build提供了“.NET Framework”的4.7.2版本和3.5版本。

    CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。

    一般来讲，4.7.2版本可以兼容4.0以上版本SDK，3.5版本可以兼容3.5版本以下SDK，项目中可以引用兼容版本的SDK内容。
  - 在某些场景下，用户可能将引用直接指向了某个低版本的SDK路径，此时会因找不到SDK导致构建失败。

    如果您的项目出现此类场景，建议：

- 尝试更改您的引用路径，尽可能使用兼容版本SDK。
- 如果您的项目因为不可回避的原因，必须指向低版本SDK路径，请联系客服。

图形化构建

在**配置构建步骤**中，添加“Msbuild构建”构建步骤，参考**表5-41**配置参数。

表 5-41 Msbuild 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
powershell命令	配置Msbuild命令，一般使用系统默认给出的命令即可。常见命令可参考**常用Powershell命令**。 ● **nuget restore**命令会还原csharp项目依赖。 ● **msbuild**命令构建项目： – OutputPath会指定生成路径，此路径设置会以csproject文件为相对路径。 – 项目存在多个csproject时可能会因此导致构建失败，删除路径指定参数后可构建成功。 ● .NET Core项目请尝试使用.NET相关命令构建。

常用 Powershell 命令

如果在构建任务中使用**powershell**命令，在命令行窗口按如下格式输入命令即可。

```
powershell –Command Powershell命令 Powershell命令参数列表
```

表 5-42 常用 Powershell 命令

Powershell命令	命令使用说明
Compress-Archive（压缩命令）	powershell –Command Compress-Archive -Path [*SourcePath*] -DestinationPath [*Target.zip*] ● SourcePath：指定需要压缩的文件或文件夹，支持通配符和相对路径。 ● Target.zip：输出的压缩文件名，可用于上传到软件发布库时填写文件名。

Powershell命令	命令使用说明
Expand-Archive（解压缩命令）	powershell –Command Expand-Archive -Path [*SourcePath*] -DestinationPath [*TargetPath*] • SourcePath：指定需要解压的文件，如“demo.zip”。 • TargetPath：要解压到的目标路径，支持通配符和相对路径。
Copy-Item（复制命令）	powershell –Command Copy-Item -Recurse -Path [*SourcePath*] -DestinationPath [*TargetPath*] • SourcePath：指定需要复制的文件或文件夹，支持通配符和相对路径。 • TargetPath：要复制到的目标路径，支持通配符和相对路径。 **说明** -Recurse选项为循环复制子文件夹，但若在**SourcePath**中使用了通配符，此开关会失效，不会复制指定目录下的子文件夹。

说明

文档只介绍了构建常用的Powershell命令，更多Powershell命令请参见**官方文档**。

5.3.15 使用 CMake 构建

使用CMake构建工具可以构建跨平台项目工程。

图形化构建

在**配置构建步骤**中，添加“CMake构建”构建步骤，参考**表5-43**配置参数。

表 5-43 CMake 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置CMake命令，一般使用系统默认给出的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。更多命令使用方法可参考**CMake官网**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - cmake:
        inputs:
          command: |
            # 新建build目录 切换到build目录、
            mkdir build && cd build
            # 生成Unix 平台的makefiles文件并执行构建
            cmake -G 'Unix Makefiles' ../ && make -j
          ignore_fail: true
```

表 5-44 代码示例参数说明

参数	类型	说明
command	string	配置CMake命令。更多命令使用方法可参考**CMake官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.16 使用 Ant 构建

使用Ant构建可以编译、测试和部署Java项目。

图形化构建

在**配置构建步骤**中，添加“Ant构建”构建步骤，参考**表5-45**配置参数。

表 5-45 Ant 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置Ant构建命令，一般使用系统默认给出的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。更多命令使用方法可参考**Ant官网**。

参数	说明
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - ant:
        inputs:
          command:  ant -f build.xml
          ignore_fail: true
```

表 5-46 代码示例参数说明

参数名称	参数类型	参数说明
command	string	配置Ant构建命令。更多命令使用方法可参考**Ant官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.17 使用 Kotlin 构建

使用Kotlin构建可以编译、测试和部署项目。

图形化构建

在**配置构建步骤**中，添加“Kotlin构建”构建步骤，参考**表5-47**配置参数。

表 5-47 Kotlin 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。

参数	说明
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置Kotlin构建命令，一般使用系统默认给出的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。更多命令使用方法可参考**Kotlin官网**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - kotlin:
        inputs:
          command:  gradle build
          ignore_fail: true
```

表 5-48 代码示例参数说明

参数	类型	说明
command	string	配置Kotlin构建命令。更多命令使用方法可参考**Kotlin官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.18 使用 Go 语言构建

使用Go语言构建Go项目，包括编译源代码生成可执行文件、处理项目依赖、以及定制化构建流程等。

图形化构建

在**配置构建步骤**中，添加“Go语言构建”构建步骤，参考**表5-49**配置参数。

表 5-49 Go 语言构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置Go项目构建命令，一般使用系统默认给出的命令即可，如有特殊构建要求，可以在文本域中输入自定义的构建脚本。更多命令使用方法可参考**Go官网**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - go:
        inputs:
          command: |
            export GO15VENDOREXPERIMENT=1
            export GOPROXY=https://goproxy.cn
            mkdir -p $GOPATH/src/example.com/demo/
            cp -rf . $GOPATH/src/example.com/demo/
            go install example.com/demo
            cp -rf $GOPATH/bin/ ./bin
          ignore_fail: true
```

表 5-50 代码示例参数说明

参数	类型	说明
command	string	配置Go项目构建命令。更多命令使用方法可参考**Go官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 ● true：是。 ● 为空：否。

5.3.19 使用 Ionic Android App 构建

使用Ionic Android App构建可以创建一个跨平台的移动应用，支持快速开发移动App、移动端Web页面、混合App和Web页面。

需项目中包含“ionic.config.json”、“package.json”和“angular.json”等项目编译描述文件。

图形化构建

在**配置构建步骤**中，添加“Ionic Android App构建”构建步骤，参考**表5-51**配置参数。

表 5-51 Ionic Android App 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
Gradle	根据用户实际开发环境的需要，选择对应的Gradle版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
JDK	根据用户实际开发环境的需要，选择对应的JDK版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
NDK	根据用户实际开发环境的需要，选择对应的NDK版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置命令框中的打包脚本。更多命令使用方法可参考**Ionic官网**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - ionic_android_app:
        inputs:
          gradle: '4.8'
          jdk: '33'
          ndk: '17'
          command: ./instrumented.apk
          ignore_fail: true
```


表 5-52 代码示例参数说明

参数	类型	说明
gradle	string	根据用户实际开发环境的需要，选择对应的Gradle版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
jdk	string	根据用户实际开发环境的需要，选择对应的JDK版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
ndk	string	根据用户实际开发环境的需要，选择对应的NDK版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
command	string	配置命令框中的打包脚本。更多命令使用方法可参考**Ionic官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.20 构建 Android 快应用

使用Npm配置命令构建Android快应用。

图形化构建

在**配置构建步骤**中，添加“Android快应用构建”构建步骤，参考**表5-53**配置参数。

表 5-53 Android 快应用构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置Npm命令，更多命令使用方法可参考**Node.js官网**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - quick_app:
        inputs:
          command: |-
            npm config set registry http://7.223.219.40/npm/
             # 加载依赖
            npm install --verbose
            # 默认构建
            npm run build
          ignore_fail: true
```

表 5-54 代码示例参数说明

参数	类型	说明
command	string	配置Npm命令，更多命令使用方法可参考**Node.js官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.21 使用 GFortran 构建

GFortran构建可以编译单个GFortran源码或者整个GFortran项目。

图形化构建

在**配置构建步骤**中，添加“GFortran构建”构建步骤，参考**表5-55**配置参数。

表 5-55 GFortran 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置GFortran命令。 一般使用系统默认生成的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。

参数	说明
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - fortran:
        inputs:
          command: |-
            gfortran -c -fpic helloworld.f90
            gfortran -shared -o helloworld.so helloworld.o
          ignore_fail: true
```

表 5-56 代码示例参数说明

参数	类型	说明
command	string	配置GFortran命令。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.22 使用 Sbt 构建

使用Sbt构建Scala和Java项目。

图形化构建

在**配置构建步骤**中，添加“Sbt构建”构建步骤，参考**表5-57**配置参数。

表 5-57 Sbt 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。

参数	说明
工具版本	默认版本为sbt1.3.2-jdk1.8，当前仅支持该版本。
命令	配置Sbt命令，一般使用系统默认给出的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。更多命令使用方法可参考**Sbt官网**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - sbt:
        inputs:
          command: |
            sbt package
          ignore_fail: true
```

表 5-58 代码示例参数说明

参数	类型	说明
command	string	配置Sbt命令。更多命令使用方法可参考**Sbt官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.23 使用 Grails 构建

使用Grails可以构建Web应用。

图形化构建

在**配置构建步骤**中，添加“Grails构建”构建步骤，参考**表5-59**配置参数。

表 5-59 Grails 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置Grails命令，一般使用系统默认生成的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。更多命令使用方法可参考**Grails官网**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - grails:
        inputs:
          command: grails war
          ignore_fail: true
```

表 5-60 代码示例参数说明

参数	类型	说明
command	string	配置Grails命令。更多命令使用方法可参考**Grails官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 ● true：是。 ● 为空：否。

5.3.24 使用 Bazel 构建

使用Bazel工具进行编译构建。

图形化构建

在**配置构建步骤**中，添加“Bazel构建”构建步骤，参考**表5-61**配置参数。

表 5-61 Bazel 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置Bazel命令，一般使用系统默认生成的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。更多命令使用方法可参考**官网**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - bazel:
        inputs:
          command: |
            cd java-maven
            bazel build //:java-maven_deploy.jar
            mkdir build_out
            cp -r  bazel-bin/*  build_out/
          ignore_fail: true
```

表 5-62 代码示例参数说明

参数	类型	说明
command	string	配置Bazel执行命令。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 ● true：是。 ● 为空：否。

5.3.25 使用 Flutter 构建

使用Flutter可以构建安卓应用。

图形化构建

在**配置构建步骤**中，添加“Flutter构建”构建步骤，参考**表5-63**配置参数。

表 5-63 Flutter 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
Flutter	根据用户实际开发环境的需要，选择对应的Flutter版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
JDK	根据用户实际开发环境的需要，选择对应的JDK版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
NDK	根据用户实际开发环境的需要，选择对应的NDK版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置Flutter命令，一般使用系统默认生成的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。更多命令使用方法可参考**Flutter官网**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - flutter:
        inputs:
          flutter: '1.17.5'
          jdk: '3333'
          ndk: '23.1.7779620'
          command: ./instrumented.apk
          ignore_fail: true
```


表 5-64 代码示例参数说明

参数	类型	说明
flutter	string	根据用户实际开发环境的需要，选择对应的Flutter版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
jdk	string	根据用户实际开发环境的需要，选择对应的JDK版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
ndk	string	根据用户实际开发环境的需要，选择对应的NDK版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
command	string	配置Flutter命令。更多命令使用方法可参考**Flutter官网**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.26 使用 HarmonyOS 构建

使用Hvigor进行编译，测试和部署项目。

说明

使用Hvigor构建的执行机规格需4U8G及以上。

图形化构建

在**配置构建步骤**中，添加“HarmonyOS构建”构建步骤，参考**表5-65**配置参数。

表 5-65 HarmonyOS 构建参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。
工具版本	默认版本为“HarmonyOS-API9”，当前仅支持该版本。
命令	配置命令，一般使用系统默认给出的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。

参数	说明
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - harmonyos:
        name: "HarmonyOS Build"
        inputs:
          command: |
            npm config set strict-ssl false
            npm config set registry=https://repo.huaweicloud.com/repository/npm/
            npm config set @ohos:registry=https://repo.harmonyos.com/npm/
            chmod +x hvigorw
            ./hvigorw clean assembleApp --no-daemon
          ignore_fail: true
```

表 5-66 代码示例参数说明

参数	类型	说明
command	string	配置HarmonyOS的执行命令。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.27 使用构建方舟编译器构建

使用ubuntu操作系统编译方舟编译器进行构建。

图形化构建

在**配置构建步骤**中，添加“构建方舟编译器”构建步骤，参考**表5-67**配置参数。

表 5-67 构建方舟编译器参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	配置方舟编译器的执行命令，一般使用系统默认生成的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - ark:
        inputs:
          command: make
          ignore_fail: true
```

表 5-68 代码示例参数说明

参数	类型	说明
command	string	配置方舟编译器的执行命令。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 ● true：是。 ● 为空：否。

5.3.28 通过 Docker 命令操作镜像

通过执行Docker命令操作镜像，比如登录、推送、下载等操作。

图形化构建

在**配置构建步骤**中，添加“执行Docker命令”构建步骤，参考**表5-69**配置参数。

表 5-69 执行 Docker 命令参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
命令	单击“添加”，新增一条命令行，请根据需要选择并配置命令。 CodeArts Build支持的Docker命令可参考**编译构建支持的Docker命令**。 可通过拖动命令调整命令执行顺序。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - docker:
        inputs:
          command: |
            docker pull swr.xx-xxxxx-x.myxxcloud.com/codeci/dockerindocker:dockerindocker18.09-1.3.2
          ignore_fail: true
```

表 5-70 代码示例参数说明

参数	类型	说明
command	string	执行命令，每个命令一行。支持的docker命令可参考**编译构建支持的Docker命令**。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 ● true：是。 ● 为空：否。

编译构建支持的 Docker 命令

- **docker login**：登录docker仓库。
  用法：docker login [options] [server]
  options填写方法如下表，**server**为docker仓库地址。

参数	对应短名称	说明
--password	-p	登录仓库的用户密码。
--username	-u	登录仓库的用户名。
--password	-stdin	从标准输入获取用户密码。

示例：docker login -u jack -p 12345 mydocker-registry.com

该示例表示使用jack用户远程登录地址为“mydocker-registry.com”的仓库，密码为“12345”。

高级用法

从文件里获取密码：cat ~/my_password.txt | docker login **--username** jack **--password-stdin**

- **docker build**：通过Dockerfile或者上下文制作镜像。上下文可以是构建执行所在的本地路径“Path”，也可以是远程URL，如Git库、tarball或文本文件等，还可以是“-”。

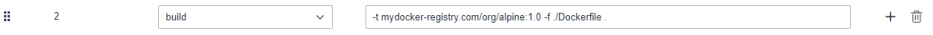
  用法：docker build [options] Path | URL | -

  options填写方法如下表。Path/URL/-为上下文来源。

参数	对应短名称	说明
--file	-f	Dockerfile文件路径，默认为“./Dockerfile”。
--tag	-t	“镜像名:标签”格式。

示例：docker build -t mydocker-registry.com/org/alpine:1.0 -f ./Dockerfile .

该示例表示使用当前目录且标签为“mydocker-registry.com/org/alpine:1.0”的Dockerfile制作镜像。

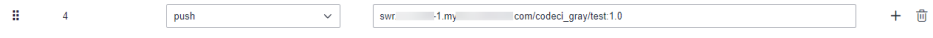


- **docker push**：推送镜像到指定的地址。

  用法：docker push [options] name[:tag]

  示例：docker push mydocker-registry.com/org/alpine:1.0

  该示例表示将“mydocker-registry.com/org/alpine”镜像的1.0版本推送到远程仓库。



- **docker pull**：从镜像仓库下载镜像到本地。

  用法：docker pull [options] name[:tag|@digest]

  options填写方法如下表。

参数	对应短名称	说明
--all-tags	-a	下载镜像仓库所有标识tag的镜像。

  示例：**docker pull mydocker-registry.com/org/alpine:1.0**

  该示例表示从远程仓库拉取版本号为1.0的mydocker-registry.com/org/alpine镜像。

- **docker tag**：修改镜像的标签。

  用法：docker tag source_image[:tag] target_image[:tag]

  其中source_image[:tag]表示需要修改标签的镜像，**target_image[:tag]**表示目标镜像。

  示例：**docker tag mydocker-registry.com/org/alpine:1.0 mydocker-registry/neworg/alpine:2.0**

  该示例表示，将“mydocker-registry.com/org/alpine”镜像的标签从“1.0”改为“2.0”。

- **docker save**：保存一个或者多个镜像到tar类型的文件，默认是标准输出流。

  用法：docker save [options] image [image ...]

  options填写方法如下表。

参数	对应短名称	说明
--output	-o	写文件，非使用标准输出。

  示例：**docker save -o alpine.tar mydocker-registry.com/org/alpine:1.0 mydocker-registry.com/org/alpine:2.0**

  该示例表示将mydocker-registry.com/org/alpine:1.0镜像和mydocker-registry.com/org/alpine:2.0镜像打包到alpine.tar。

- **docker logout**：从镜像仓库登出。

  用法：docker logout [server]

  示例：**docker logout mydocker-registry.com**

  该示例表示登出地址为mydocker-registry.com的镜像仓。

5.3.29 生成单元测试报告

该步骤用于解析用户生成的单元测试结果文件，并生成可视化报告。

前提条件

在执行“单元测试报告”步骤前，需已生成测试结果文件，并且该文件框架符合编译构建服务支持的框架类型。

图形化构建

在**配置构建步骤**中，添加“单元测试报告”构建步骤，参考**表5-71**配置参数。

表 5-71 单元测试报告参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
工具版本	根据用户实际开发环境的需要，选择对应的工具版本。 CodeArts Build支持的工具版本，可查看**构建工具版本**。若当前的工具版本不满足您的使用要求，您可以**自定义构建环境**。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。
单元测试	● 测试报告类型：选择单元测试的框架，目前仅支持“junit”。 ● 单元测试结果文件：填写相对于项目根目录的相对路径，如“target/surefire-reports/TEST*.xml”。目前只支持标准的“.xml”格式单元测试报告。 ● 是否处理单元测试覆盖率结果：根据实际需要选择。若选“是”，需确认项目中有使用“jacoco-maven-plugin”插件生成单元覆盖率报告。

5.3.30 自定义构建环境

CodeArts Build提供大量构建工具，如果已有工具不能满足您的使用要求，如缺少必要的依赖包、工具等，您可以根据需要，通过自定义Dockerfile文件的方式制作镜像并推送至指定的SWR仓库后使用，使用方法可参考**使用自定义环境构建**。

本节以Maven构建为例，为您介绍如果通过修改Dockerfile文件自定义环境。

自定义构建环境前的准备工作

- 已在容器镜像服务**创建组织**。组织的约束与限制参考容器镜像服务的**约束与限制**。
- 如果您制作后的镜像需要推送至华为云其他用户的SWR中，需**新建IAM账户服务扩展点**。
- 如果您制作后的镜像需要推送至其他镜像仓库，需**新建Docker repository服务扩展点**。

自定义 Dockerfile 文件

1. 通过项目入口方式**访问CodeArts Build服务首页**。
2. 在编译构建服务首页右上角单击“更多”，在下拉列表选择“自定义构建环境”。
3. 进入自定义构建环境页面，选择合适的基础镜像，单击即可下载Dockerfile模板。

   图 5-6 Dockerfile 模板

4. 编辑下载的Dockerfile文件。

   可根据需要加入项目需要的其他依赖和工具，完成Dockerfile文件自定义，如下为添加了jdk和maven工具的示例。

   ```
   RUN yum install -y java-1.8.0-openjdk.x86_64
   RUN yum install -y maven
   RUN echo 'hello world!'
   RUN yum clean all
   ```

5. 在导航栏选择“代码 > 代码托管”，单击使用的代码仓名称，进入代码仓详情页。
6. 在“代码”页签，选择“新建 > 上传文件”，将Dockerfile文件以及制作镜像过程中需要的所有文件上传到代码仓库根目录。

制作镜像并推送到 SWR 仓库

- **图形化构建**

  在**配置构建步骤**中，“Maven构建”步骤后添加“制作镜像并推送到SWR仓库”构建步骤。

  “Maven构建”构建步骤参数保持默认即可，如当前参数配置不满足使用要求，可参考**使用Maven构建**修改参数配置。“制作镜像并推送到SWR仓库”构建步骤参数配置说明如**表5-72**。

  表 5-72 制作镜像并推送到 SWR 仓库参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 • 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 • 字符长度范围为1~128。
工具版本	选择使用的Docker版本，使用默认版本即可。 当前编译构建服务支持Docker18.03和Docker20.10版本。
镜像仓库	选择需要推送的镜像仓库。支持推送至华为云镜像仓库SWR和其他镜像仓库，根据实际情况选择即可。

参数	说明
授权用户	推送的镜像仓库的所属用户。支持推送至当前用户和其他用户。 需确保用户对组织内所有镜像具有编辑或管理权限，详见**授权管理**。 当“镜像仓库”选择“华为云镜像仓库SWR”时需要配置该参数。
IAM账号	在下拉框中选择**自定义构建环境前的准备工作**中创建的“IAM账户”服务扩展点，通过服务扩展点推送至其他用户的SWR中。 当“授权用户”选择“其他用户”时需要配置该参数。
推送区域	选择需要推送的区域。镜像制作成功后，会将镜像依次推送到选中区域的SWR仓库。 当“镜像仓库”选择“华为云镜像仓库SWR”时需要配置该参数。
镜像仓接入点	选择**自定义构建环境前的准备工作**中创建的“Docker repository”服务扩展点，通过服务扩展点推送至其他镜像仓库。 当“镜像仓库”选择“其他镜像仓库”时需要配置该参数。
组织	在下拉框中选择**自定义构建环境前的准备工作**中创建好的组织名，选择镜像推送到SWR后所属的组织。
镜像名字	自定义填写制作完成后的镜像名称。 需以数字或字母开头，仅支持小写字母、数字、“_”、“-”，字符长度为1~255。
镜像标签	用来标记镜像的版本，可自定义。通过“镜像名:标签”可以唯一指定镜像。 仅支持大小写字母、数字、“.”、“_”、“-”，不可以“.”或“-”开头，字符长度为1~128。
工作目录	可选参数。 填写docker build命令中的“上下文路径”参数，该路径是代码仓库根目录的相对路径。 上下文路径，指的是docker在构建镜像时，docker build命令将该路径下的所有内容打包给容器引擎，帮助构建镜像。
Dockerfile路径	可选参数。 Dockerfile文件所在路径，请填写相对于工作目录的路径，如：工作目录为根目录， 且Dockerfile文件在根目录下，则此处填写为“./Dockerfile”。
添加构建元数据到镜像	配置是否将本次构建信息添加到镜像中，镜像制作完成后可以通过docker inspect命令查看镜像元数据。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

- **代码化构建**

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - build_image:
        name: buildImage
        inputs:
          regions: ["x-x-x", "x-x-xxx"]
          organization: codeci_test
          image_name: demo
          image_tag: ${GIT_COMMIT}
          dockerfile_path: dockerfile/Dockerfile
          # set_meta_data: true
          ignore_fail: true
```

表 5-73 制作镜像并推送到 SWR 仓库代码示例参数说明

参数	类型	说明
regions	list	可选参数。 选择要上传的区域SWR。默认上传到当前任务所在region的SWR。如果配置多个region，镜像制作成功后，会将镜像依次推送到填写region的SWR仓库中。
organization	string	配置镜像推送到SWR后所属的组织名称。组织名称为**自定义构建环境前的准备工作**中创建好的组织名。
image_name	string	可选参数。 自定义填写制作完成后的镜像名称。 需以数字或字母开头，仅支持小写字母、数字、“_”、“-”，字符长度为1~255。 默认值：demo。
image_tag	string	可选参数。 用来标记镜像的版本，可自定义。通过“镜像名:标签”可以唯一指定镜像。 仅支持大小写字母、数字、“.”、“_”、“-”，不可以“.”或“-”开头，字符长度为1~128。 默认值：v1.1。
context_path	string	可选参数。 填写docker build命令中的“上下文路径”参数，该路径是代码仓库根目录的相对路径。 上下文路径，指的是docker在构建镜像时，docker build命令将该路径下的所有内容打包给容器引擎，帮助构建镜像。 默认值：.。
dockerfile_path	string	可选参数。 Dockerfile文件所在路径，请填写相对于工作目录的路径，如：工作目录为根目录， 且Dockerfile文件在根目录下，则此处填写为“./Dockerfile”。 默认值：./Dockerfile。

参数	类型	说明
set_meta_data	bool	可选参数。 配置是否将本次构建信息添加到镜像中，镜像制作完成后可以通过docker inspect命令查看镜像元数据。 ● true：添加。 ● false：不添加。 默认值：false。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 ● true：是。 ● 为空：否。

5.3.31 使用自定义环境构建

如果当前CodeArts Build支持的工具版本不满足您的使用要求，您可以使用已上传到SWR的自定义镜像进行构建。

将镜像设置为“公开”

由于CodeArts Build无法拉取您在SWR私有仓中的镜像，因此，需要先将镜像设置为“公开”。

1. 登录**容器镜像服务**。
2. 在导航栏单击“我的镜像”，单击“镜像名称”进入镜像详情页面，然后单击右上角“编辑”。
3. 在弹框中，将“类型”设置为“公开”，单击“确定”。

图 5-7 编辑镜像

4. 获取完整的镜像地址：单击复制镜像下载指令，其中，**docker pull**后面部分为镜像地址。

图形化构建

在**配置构建步骤**中，添加“使用SWR公共镜像”构建步骤，参考**表5-74**配置参数。

表 5-74 使用 SWR 公共镜像参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
镜像地址	填写**4**中获取的镜像地址。
命令	配置命令，一般使用系统默认生成的命令即可。如有特殊构建要求，可以在文本域中输入自定义的构建脚本。 例如：若使用的镜像是用于Maven构建，则配置Maven构建命令；若使用的镜像是用于NPM构建，则配置NPM构建命令，以此类推。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - swr:
      image: cloudbuild@ddd
      inputs:
        command: echo 'hello'
        ignore_fail: true
```

表 5-75 使用 SWR 镜像代码示例参数说明

参数	类型	说明
image	string	填写镜像地址，分为以下两种填写方式： • 以cloudbuild开始，@作为分隔符，后面是编译构建支持的工具版本对应的名称。例如：cloudbuild@maven3.5.3-jdk8-open，其中“maven3.5.3-jdk8-open”为Maven构建的工具版本名称。 • 4中获取的镜像地址。
command	string	配置执行命令。 例如：若使用的镜像是用于Maven构建，则配置Maven构建命令；若使用的镜像是用于NPM构建，则配置NPM构建命令，以此类推。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.3.32 下载软件发布库中的软件包

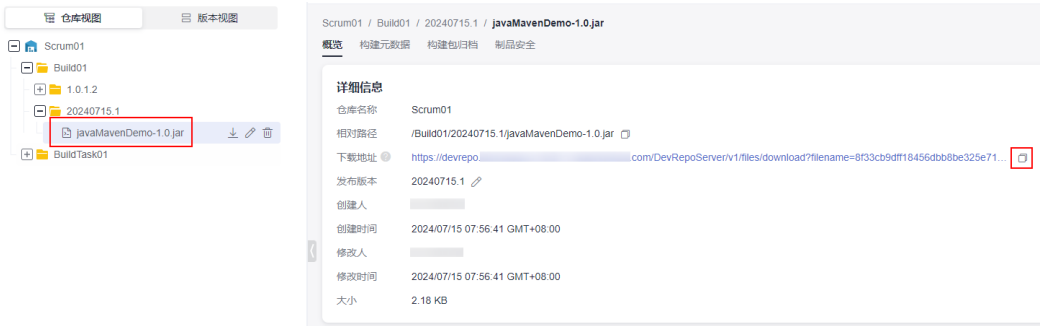
CodeArts Build支持将软件发布库中的包或者其他文件下载到构建任务根目录，以便后续构建步骤使用这些包或者文件。

获取软件包下载地址

步骤1 在导航栏选择“制品仓库 > 软件发布库”，进入软件发布库页面。

步骤2 单击待下载的软件包包名，在软件包包详情页面，“下载地址”即为软件包的下载地址，单击地址旁的，复制该地址。

图 5-8 软件包地址



----结束

图形化构建

在**配置构建步骤**中，添加“下载发布仓库包”构建步骤，参考**表5-76**配置参数。

表 5-76 下载发布仓库包参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
工具版本	根据实际需要选择工具版本。
下载包地址	将**步骤2**复制的软件包下载地址粘贴到输入框。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - download_artifact:
        inputs:
          url: xxxxxxxxxxxx
          ignore_fail: true
```

表 5-77 代码示例参数说明

参数	类型	说明
url	string	粘贴从**步骤2**复制的软件包下载地址。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 ● true：是。 ● 为空：否。

5.3.33 上传软件包到软件发布库

上传的软件包相关限制，请参考制品仓库服务的**约束与限制**。

- 仅支持上传单个文件、多个文件，不支持上传文件夹、自动创建路径。
  例如：“a”目录下有“aa”文件和“b”目录，“b”目录下有“bb”文件，构建包路径配置为“a/**”。
  即递归扫描“a”目录下所有文件，两个文件是同一个目录下，“aa”、“bb”两个文件将会上传到同一个目录下，系统不会在软件发布库里自动创建“b”目录。

- 如果用户有上传文件夹的需要，建议在“上传软件包到软件发布库”构建步骤之前，将待上传的文件夹打包为单文件后再上传。可以通过现有构建步骤执行打包命令，也可以新增“执行shell命令”构建步骤执行打包命令。

图形化构建

在**配置构建步骤**中，添加“上传软件包到软件发布库”构建步骤，参考**表5-78**配置参数。

说明

当**执行机**选择Windows执行时，需添加“上传软件包到软件发布库(Windows环境)”构建步骤。

表 5-78 上传软件包到软件发布库参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
构建包路径	填写构建结果所在路径。 ● 构建包路径支持正则匹配，“**”递归遍历当前目录，“*”匹配0或者多个字符，“?”匹配一个字符。 ● 系统文件分隔符使用“/”，路径不区分大小写。 举例说明： ● *.class：当前目录下匹配“.class”结尾的文件。 ● **/*.class：当前目录下递归匹配所有的“.class”结尾的文件。 ● test/a??.java：匹配“test”目录下以“a”开头后跟两个字符的java文件。 ● **/test/**/XYZ*：递归匹配父目录为“test”文件是“XYZ”开头的所有文件，比如“abc/test/def/ghi/XYZ123”。
发布版本号	可选参数。 配置当前构建任务生成的软件包上传到软件发布库中的目录名。 ● 不指定（推荐）：以构建编号命名上传到发布库的文件存储目录名。 ● 指定：可能会覆盖同名存储目录下的文件。
包名	可选参数。 配置当前构建任务生成的软件包上传到软件发布库中的软件包名称。 ● 不指定（推荐）：以文件原始名命名上传到发布库的文件名。包名推荐设置为空，可以上传构建包路径匹配的所有文件。 ● 指定：上传多个文件时，可能会存在被覆盖的情况。如果包名需要设置且存在多个文件上传的情况，推荐增加多个“上传软件包到软件发布库”的构建步骤。

参数	说明
自定义上传目录	可选参数。 填写自定义上传目录后，上传的软件包将上传至“自定义上传目录/版本号/软件包名”的目录下。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - upload_artifact:
          inputs:
            path: "**/target/*.?ar"
            version: 2.1
            name: packageName
            custom_upload_path: /phoenix-sample-ci/
            ignore_fail: true
```

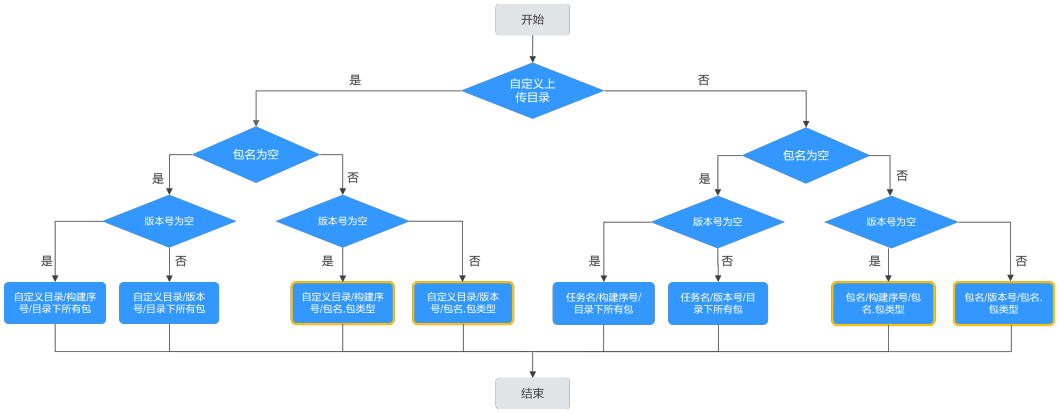
表 5-79 代码示例参数说明

参数	类型	说明
path	string	填写构建结果所在路径。 • 构建包路径支持正则匹配，“**”递归遍历当前目录，“*”匹配0或者多个字符，“?”匹配一个字符。 • 系统文件分隔符使用“/”，路径不区分大小写。 举例说明： • *.class：当前目录下匹配“.class”结尾的文件。 • **/*.class：当前目录下递归匹配所有的“.class”结尾的文件。 • test/a??.java：匹配“test”目录下以“a”开头后跟两个字符的java文件。 • **/test/**/XYZ*：递归匹配父目录为“test”文件是“XYZ”开头的所有文件，比如“abc/test/def/ghi/XYZ123”。
version	string	可选参数。 填写发布版本号。 不填写（推荐）：以构建编号命名上传到发布库的文件存储目录名。 填写：可能会覆盖同名存储目录下的文件。

参数	类型	说明
name	string	可选参数。 填写构建生成的包名。 不填写（推荐）：以文件原始名命名上传到发布库的文件名。 填写：上传多个文件时，可能会存在被覆盖的情况。
custom_upload_path	string	可选参数。 填写自定义上传目录后，上传的软件包将上传至“自定义上传目录/版本号/软件包名”的目录下。
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

发布版本号及包名对上传的影响

图 5-9 发布版本号及包名是否为空对上传的影响



5.3.34 上传文件到 OBS

CodeArts Build支持将构建产物上传至OBS中，您可以根据实际情况选择使用该构建步骤。

对象存储服务（OBS）的使用限制请参考**约束与限制**。

上传文件到 OBS 前的准备工作

如果需要将文件上传到其他用户的OBS中，需**新建IAM账户服务扩展点**。

图形化构建

在**配置构建步骤**中，添加“上传文件到OBS”构建步骤，参考**表5-80**配置参数。

表 5-80 上传文件到 OBS 参数说明

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。 ● 支持中文、英文、数字、“-”、“_”、英文逗号、英文分号、英文冒号、“.”、“/”、圆括号（中英文）。 ● 字符长度范围为1~128。
授权用户	选择需要推送的OBS所属用户。 ● 当前用户：上传到当前用户的OBS桶。 ● 其他用户：可以通过选择IAM账号的方式上传到指定用户的OBS中。
IAM账号	在下拉框中选择**上传文件到OBS前的准备工作**中创建的“IAM账户”服务扩展点，通过服务扩展点推送至其他用户的OBS中。 当“授权用户”选择“其他用户”时需要配置该参数。
构建产物路径	构建结果所在路径，OBS存储文件名为空时，可使用通配符上传多个文件。如：maven可以使用**/target/*.?ar匹配所有构建出来的jar包和war包。 举例说明： ● *.class：当前目录下匹配“.class”结尾的文件。 ● **/*.class：当前目录下递归匹配所有的“.class”结尾的文件。 ● test/a??.java：匹配“test”目录下以“a”开头后跟两个字符的java文件。 ● **/test/**/XYZ*：递归匹配父目录为“test”文件是“XYZ”开头的所有文件，比如“abc/test/def/ghi/XYZ123”。
桶名	填写目标OBS的桶名（不支持跨region上传）。
OBS存储目录	可选参数。 填写构建结果在OBS上的存储目录（如：application/version/），可留空，或填写“./”表示存储到OBS根目录。
OBS存储文件名	可选参数。 填写构建结果在OBS上的存储文件名（不包含目录）。 ● 留空时可上传多个文件，将构建产物文件名作为OBS存储文件名。 ● 不为空时只能上传单个文件，如：application.jar。
是否上传文件夹	配置是否开启上传文件夹。 ● 是：同步上传文件夹。 ● 否：仅上传文件。

参数	说明
忽略文件夹路径	可选参数。 配置忽略的文件夹路径。当上传文件夹时，会根据此路径忽略部分文件夹，不上传到OBS。 如产物路径填写为“target/api/api.jar”，忽略文件夹路径填写为“target”，obs存储目录为“./”，则会将“api.jar”上传到OBS桶的“api/api.jar”路径下。若路径无法匹配，则默认不忽略路径中的文件夹。 当“是否上传文件夹”设置为“是”时，需要配置该参数。
OBS头域	可选参数。 上传文件时加入一个或多个自定义的响应头，当用户下载此对象或查询此对象元数据时，加入的自定义响应头会在返回消息的头域中出现。 如：“键”填写“x-frame-options”，“值”填写成“false”，即可禁止OBS中存放的网页被第三方网页嵌入。
失败后是否继续运行	当前步骤执行失败后是否继续执行下一个步骤，根据实际使用场景选择是或否。

代码化构建

参考以下代码示例，修改在**创建代码化构建使用的YAML文件**中的**BUILD**部分代码信息。

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - upload_obs:
        inputs:
          artifact_path: "**/target/*.?ar"
          bucket_name: codecitest-obs
          obs_directory: "./"
        #   artifact_dest_name: ""
        #   upload_directory: true
        #   headers:
        #     x-frame-options: true
        #     test: test
        #     commit: ${commitId}
         ignore_fail: true
```

表 5-81 代码示例参数说明

参数	类型	说明
artifact_path	string	可选参数。 构建结果所在路径，OBS存储文件名为空时，可使用通配符上传多个文件。如：maven可以使用**/target/*.?ar匹配所有构建出来的jar包和war包。 举例说明： ● *.class：当前目录下匹配“.class”结尾的文件。 ● **/*.class：当前目录下递归匹配所有的“.class”结尾的文件。 ● test/a??.java：匹配“test”目录下以“a”开头后跟两个字符的java文件。 ● **/test/**/XYZ*：递归匹配父目录为“test”文件是“XYZ”开头的所有文件，比如“abc/test/def/ghi/XYZ123”。 默认值：bin/*。
bucket_name	string	填写目标OBS的桶名（不支持跨region上传）。
obs_directory	string	可选参数。 填写构建结果在OBS上的存储目录（如：application/version/），可留空，或填写“./”表示存储到OBS根目录。 默认值：./。
artifact_dest_name	string	可选参数。 填写构建结果在OBS上的存储文件名（不包含目录）。 ● 留空时可上传多个文件，将构建产物文件名作为OBS存储文件名。 ● 不为空时只能上传单个文件，如：application.jar。
upload_directory	bool	可选参数。 配置是否上传文件夹。 ● true：构建产物的文件夹也会同步上传。 ● false：会将匹配到的所有构建产物平铺上传到obs_directory的目录下。 默认值：false。
headers	map	可选参数。 上传文件时加入一个或多个自定义的响应头，当用户下载此对象或查询此对象元数据时，加入的自定义响应头会在返回消息的头域中出现。 例如：“x-frame-options”参数值配置为“false”，则表示禁止OBS中存放的网页被第三方网页嵌入。

参数	类型	说明
ignore_fail	string	用于控制当前步骤执行失败后是否继续执行下一个步骤。 • true：是。 • 为空：否。

5.4 配置构建任务参数

系统预定义参数

系统预定义参数的参数值由系统自动生成，无需定义，如表5-82所示，可在代码中使用**${参数名}**引用。

表 5-82 系统预定义参数

参数	说明
BUILDNUMBER	构建编号。格式为“日期.今日该构建任务执行次数”，例如：20200312.3。
TIMESTAMP	构建任务执行时间戳。例如：20190219191621。
INCREASENUM	该构建任务执行总次数，从1开始自增长，每执行1次加1。
PROJECT_ID	该构建任务所在的项目编号。
WORKSPACE	该构建任务拉取的源代码根目录，即工作空间。
GIT_TAG	代码tag名，在**配置代码下载**时指定tag构建才有值。
COMMIT_ID_SHORTER	代码提交号的前8位。在**配置代码下载**时指定CommitID构建才有值。
COMMIT_ID	代码提交号。例如：b6192120acc67074990127864d3fecaf259b20f5。

添加自定义参数的配置指导

在编译构建任务配置页面，切换至“参数设置”页签，单击“新建参数”，参考表5-83配置参数。

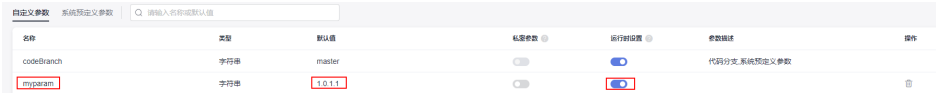
表 5-83 添加自定义参数

名称	类型	默认值	私密参数	运行时设置	参数描述
自定义参数名称。支持字母，数字，下划线“_”，长度不超过128个字符。 **说明** • 以下字段不可用：LD_PRELOAD、LD_LIBRARY_PATH、PATH、BASH_ENV、GIT_SSH_COMMAND、path。 • 不支持任何符号。	字符串	自定义参数的默认值。长度不超过8192个字符。	设置是否为私密参数。参数为私密参数时，系统会将输入参数进行加密存储，使用时再进行解密，同时在运行日志里不可见。	设置该参数是否在执行构建任务时设置。 打开“运行时设置”开关，表示通过单击 ▷ 按钮执行构建任务时支持变更参数值，并且系统会把该参数上报到流水线服务。	自定义关于该参数的描述信息。长度不超过1024个字符。
	枚举	在弹出的对话框中，填写自定义的“可选取值”，每个参数值必须以英文分号结尾。长度不超过8192个字符。 配置完可选取值后，在“默认值”的下拉框中为该参数配置一个默认值。			
	自增长	自定义参数的默认值。长度不超过8192个字符。			

使用参数的操作指导

以图5-10为例为您介绍如何使用自定义的参数。

图 5-10 自定义参数



1. 在编译构建任务配置页面切换到“构建步骤”页签，在“上传软件包至软件发布库”构建步骤的“发布版本号”中输入“${myparam}”，保存并执行构建任务。
2. 在弹框中，将“myparam”修改为“1.0.1.2”，单击“确定”，等待构建任务执行完成。

图 5-11 填写运行参数值

3. 进入软件发布库，找到刚构建的构建包，即可看到版本号就是修改后的的“myparam”值。

图 5-12 查看构建包

5.5 配置构建任务执行计划

编译构建服务支持用户配置触发事件和定时执行任务，从而使得开发者达到项目持续集成的目的。

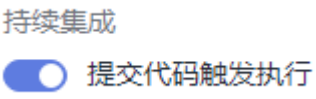
在编译构建任务配置页面，切换至“执行计划”页签，根据实际需要配置执行计划。

- 持续集成：将“提交代码触发执行”按钮设置为开启状态，构建任务所引用的代码源发生提交代码行为时，则会触发执行构建任务。

  说明

  代码源为“Repo”时才能使用。

图 5-13 配置持续集成



- 定时执行：将“启用定时执行”按钮设置为开启状态，选择需要构建任务定时执行的时间，并可按需开启是否“代码变化才执行”。

  功能开启后，构建任务会按照您设定的执行日与时间定时执行。

  若同时开启了“代码变化才执行”按钮，只有到达设定的执行日和时间，并且代码与上次构建有所变动时才会执行构建任务。

  说明

  构建任务定时执行连续失败10次后，不会再触发定时执行。

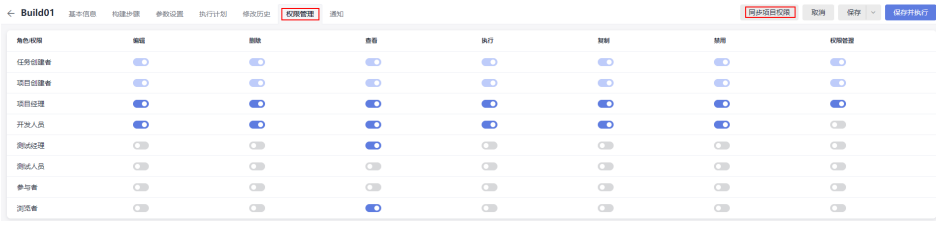
图 5-14 配置定时执行

5.6 配置构建任务角色权限

编译构建服务支持为单个构建任务的各个角色配置权限。

1. 在编译构建任务配置页面，切换至“权限管理”页签，可根据实际需要配置不同角色的操作权限，各角色默认具体的权限可参考**表3-1**。
2. 单击“同步项目权限”，可将当前构建任务的权限同步为项目权限。项目权限配置详情请参考**配置角色权限**。

图 5-15 配置构建任务角色权限



5.7 配置构建任务事件通知

编译构建服务支持给用户发送事件通知。例如任务构建成功、任务构建失败、任务被禁用、任务配置被更新和任务被删除时，可以给用户发送服务动态通知、钉钉通知或企业微信通知。

在编译构建任务配置页面，切换至“通知订阅”页签，按照实际需要进行配置。

配置服务动态通知

单击“官方通知”下“服务动态”进行设置。

默认所有事件都发送服务动态通知，任务构建失败发送邮件通知，请根据实际需要勾选通知方式。

图 5-16 配置服务动态通知

配置钉钉通知

1. 进入钉钉群，找到“群设置 > 智能群助手”，然后添加机器人（选择自定义类型）。
2. 填写机器人名字，选择群组，完成安全设置（需勾选“加签”，并单击加签文本框旁的“复制”获取加签密钥）。
3. 已阅读并同意相关协议后，单击“完成”。单击Webhook文本框旁的“复制”获取钉钉Webhook地址。
4. 选择“钉钉”通知，填写Webhook地址并单击“测试”确保Webhook地址可用。
5. 勾选“启动加签密钥”并填写加签密钥、选择事件类型。
6. 单击“保存”。

   配置完成后，当任务运行结果满足事件类型时，编译构建服务会发送消息到指定的钉钉群。

配置企业微信通知

（以手机客户端为例，详细指导请参考**如何设置群关系机器人**。）

1. 运行企业微信客户端，选中需要接受消息推送的群聊，单击右上角三个点按钮。
2. 依次单击“群机器人 > 添加 > 新建”。
3. 填写机器人名字，单击“添加”。
4. 单击Webhook文本框旁的“复制”获取企业微信Webhook地址。
5. 在配置编译构建任务的“通知订阅”页签，选择“企业微信”。
6. 填写**4**中获取到的Webhook地址。根据实际需要配置事件类型，通知内容和需要@的成员，多个成员id之间用英文逗号分开。

   配置完成后，当任务运行结果满足事件类型设置集时，编译构建服务会发送消息到指定的企业微信群并@到指定企业微信名。

配置飞书消息通知

6 执行构建任务

构建任务可通过多种途径触发执行，具体如下：

- 在CodeArts Build服务页面执行单个构建任务。
- 在CodeArts Repo代码仓提交代码时触发执行，配置方式可参考**•持续集成：将“提交代码触发执行”按钮设置为开启...**。
- 定时执行或定时执行时代码相比于上一次构建有变化才执行，配置方法可参考**•定时执行：将“启用定时执行”按钮设置为开启状态...**。
- 基于**流水线**任务触发执行。

本节为您介绍如何在CodeArts Build服务页面执行单个构建任务。

前提条件

已**新建构建任务**，且用户具有执行/禁用构建任务的权限。

执行构建任务

1. 通过项目入口方式**访问CodeArts Build服务首页**。
2. 在编译构建服务首页搜索目标任务，单击构建任务所在行的 ▷ ，开始执行构建任务。

   如果构建任务**配置了运行时参数**且被引用，将弹出参数设置提示框，根据实际情况设置执行参数值后单击确定即可。

   说明

   - 若当前构建任务并发数无法满足用户的需求，可**购买构建并发包**，增加构建任务并发数。并发包的使用规则可参考**如何使用构建并发包**。

7 查看构建任务

1. 通过项目入口方式**访问CodeArts Build服务首页**。
2. 首页展示与当前用户相关的编译构建任务列表，列表项说明如下：

列表项	说明
名称	构建任务所属项目名及构建任务名，单击项目名可以进入到项目下编译构建列表，单击任务名可以进入到构建历史页面。
最近一次执行	任务执行人员、触发方式、所用仓库的分支、CommitID等信息。
最近执行结果	从右到左显示最近执行结果，绿色为成功，蓝色为执行中，红色为失败，黄色为执行中的任务被中止，灰色为任务未被执行。
启动时间 & 执行时长	构建任务启动时间和构建所用时长。
操作	开始构建、收藏任务、单击 展开下拉菜单（编辑、复制、禁用、删除任务，具体操作请参考**管理构建任务**）。

3. 单击构建任务名称，进入“构建历史”列表页面，可以查看最近的构建历史记录（默认30天，可通过页面左上角的“日期选择组件”自定义时间周期）。

图 7-1 构建历史页面

4. 单击“洞察”页签，以饼状图/折线图/柱状图的方式查看近7天的构建成功率与构建性能分布。

   可在时间下拉框中选择查看的历史时间。

图 7-2 洞察页面

5. 单击构建历史下的构建编号，即可查看构建详情，包括代码源信息、触发来源、构建时间与时长、关联信息、排队时间、步骤日志和构建参数等。

图 7-3 构建日志页面

- 单击左上角代码源链接，可进入对应代码仓库页面。
- 单击“构建包下载”，在下拉列表中单击“下载全部”，可以下载构建成功的所有包；单击“去制品仓”，可以直接访问到“软件发布库”页面，查看所有构建成功的软件包；单击某个构建包名称，可以下载构建包。
- 单击左侧构建步骤节点（如“代码检出”），可以查看对应编译构建日志。
- 查看日志信息时，单击日志窗口右上角“全屏”，可最大化日志窗口；单击“退出全屏”，可退出最大化日志窗口；单击“下载 > 下载构建全量日志”，可下载全量日志文件；单击左侧步骤节点，可查看对应步骤日志。
- 单击右上角“编辑”或“执行”按钮，可以编辑构建任务或执行构建任务，单击  ，可以根据需要复制任务、保存模板、查看徽标状态或禁用任务。

8 加速构建任务

8.1 构建加速背景介绍

针对C/C++语言构建工程，需要提升构建效率的问题，编译构建服务支持构建加速能力，通过分布式编译和增量编译等技术实现构建加速。当前支持的构建加速的场景如下：

- **通过Gcc/Clang实现构建加速**
- **对鸿蒙构建工程配置构建加速**
- **对AOSP构建工程配置构建加速**
- **通过代码缓存方式实现构建加速**

8.2 通过 Gcc/Clang 实现构建加速

Gcc/Clang构建加速是指通过分布式编译、增量编译等技术，实现对软件编译过程的效率提升，支撑企业研发过程的快速迭代，缩短产品的上市周期。

约束与限制

- 目前该功能仅支持代码源为CodeArts Repo的C/C++语言构建工程的编译构建加速。
- 使用构建加速能力需要额外购买配套构建加速包，构建加速包因加速原理以及效果的不同，共有三种规格以供购买，规格介绍及购买指南请参考**购买构建加速包**。
- 用户基于自定义执行机的构建，无法使用构建加速能力。

配置 CMake 构建加速（图形化构建）

1. 通过项目入口方式**访问CodeArts Build服务首页**。
2. 单击需要配置构建加速的构建任务名称。
3. 单击页面右上角“编辑”按钮，进入构建任务的构建步骤配置页面。
4. 按照如下说明配置“CMake构建”构建步骤。
   “工具版本”选择“cmake3.16.5-gcc7.3.0”，“命令”中填写如下代码。

根据加速原理以及效果的不同，构建加速分为L1/L2/L3三种模式，请根据购买的加速规格使用加速命令，以下示例为开启L1模式加速。

```
cmake -G'Unix Makefiles'../&& BuildAccelerateL1 make -j8
//开启构建加速只需在make前添加加速命令：BuildAccelerateL1
//切换模式只需将BuildAccelerate后的L1替换为L2/L3。
//最大并发CPU核数，即make -j后面的数字，最大256。
```

图 8-1 构建加速命令

```
#新建build目录 切换到build目录、
mkdir build && cd build
# 生成Unix 平台的makefiles文件并执行构建
cmake -G 'Unix Makefiles' ../&& BuildAccelerateL1 make -j8
```

- **加速命令只允许写在最外层，不允许通过shell脚本等调用。**
- **同一次构建任务中禁止混用不同级别的加速命令，将无法保存与执行构建任务。**

图 8-2 构建加速错误命令示例

```
BuildAccelerateL1 make -j8
BuildAccelerateL2 make -j8
```

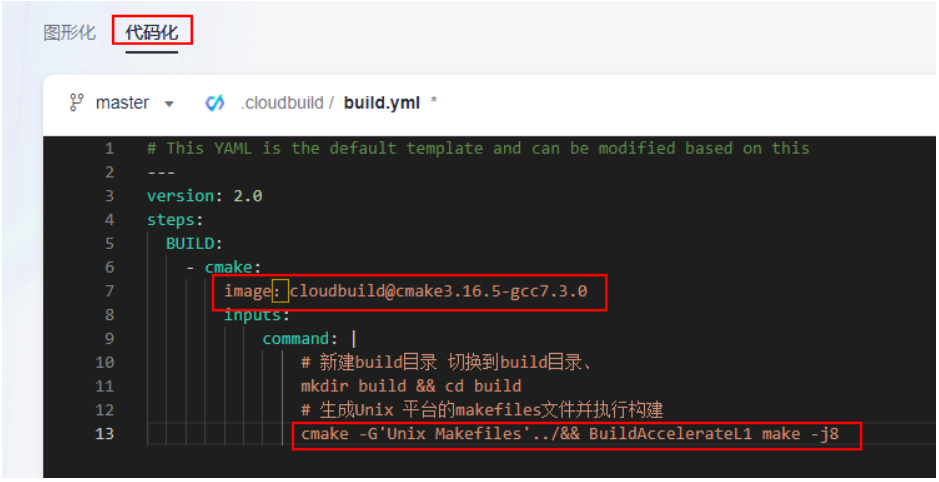
5. 配置后单击“保存并执行”执行构建任务。

配置 CMake 构建加速（代码化构建）

通过代码化构建方式实现构建加速，与图形化构建方法相同，修改工具版本并添加加速命令即可。

修改方法参考**使用CMake构建**中“代码化构建”部分，修改“image”和“command”参数，如**图8-3**所示。

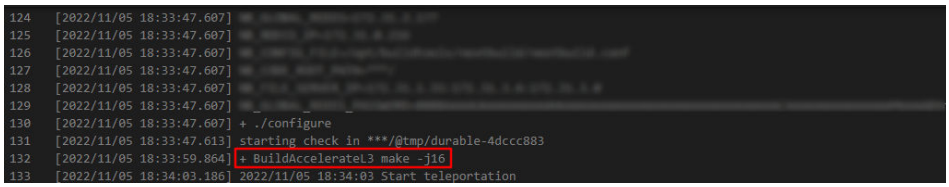
图 8-3 代码化构建



如何判断构建加速是否生效

使用CMake构建加速成功后，日志会打印出相应加速模式的信息，如下图即为L3模式加速生效。

图 8-4 构建加速生效



8.3 对鸿蒙构建工程配置构建加速

鸿蒙构建加速通过解析鸿蒙构建工程的内部依赖关系，将其拆解分发至多台机器并发执行，结合增量编译技术，实现对软件编译过程的效率提升，支撑企业研发过程的快速迭代，缩短产品的上市周期。

约束与限制

- 增量编译技术需结合L3级别加速使用。
- 使用鸿蒙构建加速能力需要购买配套构建加速包，购买方法请参考**购买增值特性**。
- 该功能目前仅支持“华北-北京四”区域、代码源为CodeArts Repo的C/C++语言构建工程的编译构建加速。
- 本章节配置仅支持鸿蒙L1、L3级别的加速。
- 构建过程中不允许直接覆盖LD_LIBRARY_PATH和LD_PRELOAD环境变量，建议通过追加的方式使用，例如：
  ```
  "export LD_LIBRARY_PATH=new_path:${LD_LIBRARY_PATH}"
  ```

加速前准备

在一般的构建工程中，其构建过程大致分为构建前准备（工具链、代码仓）、构建依赖件准备（ninja文件生成）、编译构建、构建后操作（打包、检查等）。其中，构建加速介入编译构建阶段，对此前的构建过程中生成的构建依赖件进行解析，并执行编译。

在配置构建加速前，需做如下准备：

- 找到构建依赖件准备的节点，以OpenHarmony为例，一个形态的编译命令如下：
  ```
  ./build.sh --product-name rk3568 --build-target make_all --build-target make_test --ccache false -v
  ```
- 准备好构建使用的docker镜像，基于该docker镜像制作新镜像：在dockerfile中增加“/opt/buildtools”目录供加速工具部署，并确保构建用户对“/opt/buildtools”目录有权限写入。参考命令如下：
  ```
  RUN mkdir -p /opt/buildtools && chmod -R 777 /opt/buildtools
  ```

通过 BuildFlow 组织加速构建

构建加速需要结合多任务代码化构建使用，可参考**多任务YAML文件结构详解**中的部分配置。

BuildFlow配置方法如下样例：

```
buildflow:
  jobs_resolver: # 必配
    provider: tbuild_jobs_resolver # 必配，固定值
    jobs: # 需要进行编排的任务集
        - job: distribute_job # 构建任务名称
          build_ref: accelerate.yml # 指定构建加速脚本，脚本名称可自定义
          worker: 2 # 指定为16vCPU的倍数，例如2就代表使用了32vCPU进行加速
```

参数说明如下：

- jobs_resolver：buildflow的子节点，必配。
- provider：此处使用的provider为jobs_resolver的高级选项，意为指定job对应的任务解析器，取值固定为tbuild_jobs_resolver。
- jobs：需要进行编排的任务集，此处的jobs作为jobs_resolver的子节点，与普通构建场景buildflow下的jobs子节点有所区别，配置时请注意缩进。
- job：构建任务名称，可自定义。
- build_ref：该构建任务在构建过程中需要运行的加速构建脚本。
- worker：指定为16vCPU的倍数，例如2就代表使用了32vCPU进行加速。

配置示例 1：依赖解析模式

步骤1 获取json文件。

在分支A编写BuildFlow配置中build_ref指定的accelerate.yml，示例如下：

```
version: 2.0

params:
  - name: TB_GET_ORI_TRACE
    value: "1"

steps:
  PRE_BUILD:
  - checkout:
      name: "checkout"
      inputs:
        scm: "codehub"
        url: "git@codehub.devcloud.example.example.com:example.git"
        branch: "master"
        lfs: false
        submodule: false
  BUILD:
  - tbuild_execute:
      inputs:
        image: "swr.example.example.com/buildimage:ohos-x86-v1"
        command: "cd OpenHarmony && BuildAccelerateL3 -HarmonyOS ./build.sh --product-name rk3568 --
build-target make_all --ccache false -v && post_build.sh && mv out/TBTrace_make_all_1.json /example/
TB1.json"
```

步骤2 执行分布式构建。

在分支B编写BuildFlow配置中build_ref指定的accelerate.yml，示例如下：

```
version: 2.0

params:
  - name: TB_BUILDTRACE_ALL
    value: "1"
  - name: TB_RSYNC
    value: ${WORKSPACE}/OpenHarmony/:out/rk3568

steps:
  PRE_BUILD:
```

```
 - checkout:
     name: "checkout"
     inputs:
       scm: "codehub"
       url: "git@codehub.devcloud.example.example.com:example.git"
       branch: "master"
       lfs: false
       submodule: false
  BUILD:
  - tbuild_execute:
     inputs:
       image: "swr.example.example.com/buildimage:ohos-x86-v1"
       command: "mv /example/TB1.json OpenHarmony/TB1.json && cd OpenHarmony &&
BuildAccelerateL3 -HarmonyOS ./build.sh --product-name rk3568 --build-target make_all --ccache false -v
&& post_build.sh"
```

----结束

配置示例 2：产物分类模式

在分支C编写BuildFlow配置中build_ref指定的accelerate.yml，示例如下：

```
version: 2.0

params:
  - name: TB_RSYNC
    value: ${WORKSPACE}/OpenHarmony/:out/rk3568
  - name: TB_SEARCH_TARGETS_ALL
    value: "obj/build/ohos/common/generate_src_installed_info.stamp"

steps:
  PRE_BUILD:
  - checkout:
     name: "checkout"
     inputs:
       scm: "codehub"
       url: "git@codehub.devcloud.example.example.com:example.git"
       branch: "master"
       lfs: false
       submodule: false
  BUILD:
  - tbuild_execute:
     inputs:
       image: "swr.example.example.com/buildimage:ohos-x86-v1"
       command: "cd OpenHarmony && BuildAccelerateL3 -HarmonyOS ./build.sh --product-name rk3568 --
build-target make_all --ccache false -v && post_build.sh"
```

params 参数项详解

params配置项指定了一些使用鸿蒙构建加速必配的参数，参数说明如下：

参数	说明
TB_RSYNC	需要同步的产物文件目录，“:”前为根目录，“:”后为若干个以“,”分隔的子目录，子目录前带“!”代表此目录不同步，不带“!”代表此目录的所有文件会被同步，带“!”的优先级更高。在鸿蒙构建场景下，需要拼接为“${WORKSPACE}/实际目录”。 示例如下： "${WORKSPACE}/OpenHarmony/:out/rk3568,out/kernel,!out/rk3568/exe.unstripped,!out/rk3568/lib.unstripped,!out/rk3568/innerkits,!out/rk3568/.ninja_,!out/rk3568/Makefile,!out/rk3568/NOTICE_FILES,!out/rk3568/clang_x64/exe.unstripped,!out/rk3568/clang_x64/lib.unstripped,!out/rk3568/mingw_x86_64/lib.unstripped,+out/rk3568/obj/base/security/selinux_adapter"
TB_GET_ORI_TRACE	**配置示例1：依赖解析模式**必填。 依赖解析模式下获取当前工程的依赖json文件开关。 ● 1：开启。 ● 0：关闭（默认）。 默认值：1。
TB_BUILDTRACE_ALL	**配置示例1：依赖解析模式**必填。 依赖解析模式开关，不设置时默认使用**配置示例2：产物分类模式**。 ● 1：开启。 ● 不设置：关闭（默认为OpenHarmony/harmony.json）。 ● 非“1”的其他字符串：开启，字符串视为json文件的自定义路径和名字。
TB_SEARCH_TARGETS_ALL	**配置示例2：产物分类模式**必填。 该值填写工程中汇总了各个模块的target，如鸿蒙的：parts_test.stamp 、generate_src_installed_info.stamp。一般这样的target的下一层直接依赖是工程中的多个小模块，如鸿蒙的ark模块、ace模块。这些对应的target在同一个工程里一般不会变化。 默认值："obj/build/ohos/common/generate_src_installed_info.stamp"。

steps 参数项详解

steps配置项定义了构建过程，示例中包含如下两个步骤：PRE_BUILD（构建前准备）和BUILD（编译构建）。

- PRE_BUILD

  此阶段主要做代码下载，参数解释如下：

  ```
  PRE_BUILD:
  - checkout: # 代码下载步骤
      name: "代码下载" # 步骤名称，可自定义
  ```



```
    inputs: # 步骤参数
      scm: "codehub " # 代码来源:只支持Repo
      url: "git@codehub.devcloud.example.example.com:test/python3.git" # 拉取代码的ssh地址。
      branch: "master" # 拉取的代码分支。
      lfs: false # 选择是否开启Git LFS，false关闭、true开启。构建默认不拉取音视频、图像等大型文件，开启Git LFS后，构建将会全量拉取文件。
      submodule: false # false关闭、true开启。开启该功能，系统在构建时会自动拉取子模块仓库的代码。
```

- BUILD

  此阶段主要定义了download_artifact插件、tbuild_execute插件和upload_artifact插件，参数解释如下：

```
BUILD:
 - tbuild_execute: # 鸿蒙加速场景下固定配置，定义tbuild_execute插件
    inputs: # 固定配置
      image: "swr.xx-xx-x.myxxcloud.com/buildimage:ohos-x86-v1" # 构建使用的镜像，参考加速前准备章节制作docker镜像。
      command: "mv /example/TB1.json OpenHarmony/TB1.json && cd OpenHarmony && BuildAccelerateL3 -HarmonyOS ./build.sh --product-name rk3568 --build-target make_all --ccache false -v && post_build.sh"
# command为构建使用的命令，此处将构建分解为两个段落，准备和执行
# mv /example/TB1.json OpenHarmony/TB1.json是依赖解析模式独有的准备步骤，文件名字固定，如果工程中存在多个ninja构建，则文件的下标依次增加，例如TBTrace_target_2.json和TB2.json，以此类推。
# 准备阶段:使用加速前准备章节中获取的./build.sh --product-name rk3568 --build-target make_all --build-only-gn --ccache false -v
# 构建阶段:依照加速级别调用加速命令（BuildAccelerateL1 BuildAccelerateL3）的鸿蒙模式（-HarmonyOS）直接执行构建，此处样例取值BuildAccelerateL3 -HarmonyOS
# 后处理阶段:以实际工程需要为准，该示例仅使用post_build.sh
# 依赖解析模式实际命令最终拼接为"mv /example/TB1.json OpenHarmony/TB1.json && cd OpenHarmony && BuildAccelerateL3 -HarmonyOS ./build.sh --product-name rk3568 --build-target make_all --ccache false -v && post_build.sh"
# 产物分类模式实际命令最终拼接为"cd OpenHarmony && BuildAccelerateL3 -HarmonyOS ./build.sh --product-name rk3568 --build-target make_all --ccache false -v && post_build.sh"
```

  build.sh样例：

```
source build/envsetup.sh
lunch aosp_x86_64-eng
make -j64
```

高级选项

高级选项均为非必填选项，在构建过程中有工程无法执行需要特殊适配或优化性能时配置，若随意配置可能会导致构建失败。

表 8-1 通用选项

参数	说明	示例
TB_GET_TRACE	构建结束后获取依赖json文件文件的开关。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_NINJA_RULE_ALL	分割规则配置，target按数量比值分组。不设置时会自动配置合适值。	"1:2:3:4"
TB_TARGETS_LIST_ALL	人工指定分发的target进行编译，每个英文逗号隔开一个worker的target。使用“*”分隔多个ninja工程的配置。不设置时会自动配置合适值。	"harmony_target_1 harmony_target_2,harmony_target_3 harmony_target_4"

参数	说明	示例
TB_HOOK_LOCK	对软链接也进行文件同步。若构建过程中发生软链接文件未同步导致的报错，需要开启此选项。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_APPEND_PATH	构建时可向PATH环境变量中追加的参数。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_SHUT_DOWN_SAME_TIME	所有worker都等待主节点执行完毕后再结束构建释放资源。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_RSYNC_LOCK	构建加速的同时worker向构建执行机实时传输文件。开启后效率会进一步提升，但会存在概率性编译失败的情况。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_MAKE_J	设置构建并发数。默认为worker核数。	16
TB_REFER_NINJA_FILE	如果存在串行执行一个一模一样的ninja工程时，可以使用此变量优化构建速度 ● 1：开启。 ● 0：关闭（默认）。	1
CCACHE_DIR	自定义编译缓存的本地目录。默认为/tmp/xcache目录。	${WORKSPACE}/TBcache
TB_CACHE_SIZE	使用自定义执行机自定义编译缓存的本地目录存储大小上限。默认为100G。	100G
CCACHE_MAXSIZE	自定义编译缓存的本地目录存储大小上限。默认为20G。	100G
TB_ONE_WORKER	使用自定义执行机且只使用一个worker进行构建时可以使用此变量打开编译缓存开关。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_CLIENT	该变量在主节点client自动设置，可以通过比较此变量是否等于1判断该节点是否为主节点client。	不需要配置
TB_NET_INTERFACE_NAME	指定获取IP时读取的网卡名，在多网卡情况下获取IP使用，默认为空，多个网卡名通过逗号分隔，配置在前的网卡名有更高的优先级。	eth0,eth1

参数	说明	示例
TB_ACC_PREPARE	使用自定义执行机时必须配置此变量。	false
TB_OUTPUT_PATH	自定义产物目录路径，默认设置为out。	output
TB_SELF_ENV	worker编译target时使用本地环境变量，不使用client传递的变量。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_RSYNC_FLAG	增加同步文件时rsync命令的参数。	--ignore-existing -a
TBUILD_PLUGIN_PKG_TYPE	指定使用的TBuild版本为snapshot版本还是release版本。不设置时默认使用最新的release版本。	release
TBUILD_PLUGIN_PKG_VERSION	指定使用的TBuild版本号。不设置时默认使用最新的release版本。	1.0.1

表 8-2 依赖解析模式

参数项	说明	示例
TB_CPU_NUM	分发任务时，所有机器都以此值计算可分配任务量。默认为空。	16
TB_LOCAL_CONTAIN_PATTERN	根据关键字指定必须分发在client编译的target，此target生成的产物不需要文件传输回主节点。默认为空。	file_contexts.bin,com.android.vndk.current.,out/soong/host/linux-x86/bin/fileslist
TB_LOCAL_NOT_CONTAIN_PATTERN	根据关键字指定必须不分发在client编译的target。默认为空。	Bluetooth.so
TB_CAPACITY_ALL	单台机器可分配的任务权值上限，单位是分钟，以json文件的时间为参考，此时间为单纯的构建时间，不包括cpu空闲时间，实际构建时间会大于此值。可以指定为小数。超过此上限的target会分发至client，在所有agent编译完成后在本地最后编译。不设置时会根据json文件自动设置。	5.5

参数项	说明	示例
TB_TASK_SIZE_ALL	人工指定分割多少份target，可以大于机器数量，建议设置的值略大于机器数量，不建议少于机器数量，会导致机器浪费。不设置时会根据机器数量自动设置。	8

表 8-3 ninja 文件缓存选项

参数项	说明	示例
TB_CACHE_SERVER_IP	ninja文件缓存开关，和TB_CACHE_ARCHIVE_PATH同时设置时缓存才会开启。 ● 非空值：开启，如果开启了远端缓存，该值视为远端服务器IP。 ● 空值：关闭（默认）。	172.example.example.example
TB_CACHE_ARCHIVE_PATH	ninja文件缓存本地归档目录，和TB_CACHE_SERVER_IP同时设置时缓存才会开启。	AOSP/ninja_cache
TB_CACHE_RECACHE	本次构建会重新生成ninja文件缓存，不会命中历史缓存。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_CACHE_REMOTE	ninja文件缓存远端开关，命中时从远端获取缓存，生成缓存时也会归档至远端。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_CACHE_LOCAL	ninja文件缓存本地开关，命中时从本地获取缓存，优先级高于远端缓存，生成缓存时也会归档至本地。 ● 1：开启（默认）。 ● 0：关闭。	1
TB_CACHE_DEPENDS	增加指定的文件作为缓存命中的依赖文件，如果该文件产生变化，会使缓存不命中。默认为空。多个文件使用逗号分隔。	build.sh,test.sh
TB_CACHE_VERSION	为缓存增加指定的版本号，如果版本号产生变化，会使缓存不命中。默认为空。	1.0

参数项	说明	示例
TB_CACHE_EXCLUDE_KEY	缓存时过滤掉带关键字的文件，多个关键字用逗号分隔，默认设置为".glob"，可以用空字符串重置。	.so,.glob
TB_NINJA_FILE_CACHE	设置被缓存的目录，多个目录使用逗号分隔，默认设置为"out"。	out,test
TB_CACHE_NINJA_CACHE_SIZE	ninja文件缓存本地归档目录空间上限，超过此上限会根据算法自动清理历史缓存。默认设置为50，单位是G。	100
TB_SKIP_TARGET	归档ninja文件缓存时跳过此target，即使命中ninja文件缓存此target也会重新编译。	update-api

8.4 对 AOSP 构建工程配置构建加速

构建加速服务通过解析AOSP构建工程的内部依赖关系，将其拆解分发至多台机器并发执行，结合增量编译技术，实现对软件编译过程的效率提升，支撑企业研发过程的快速迭代，缩短产品的上市周期。

约束与限制

- 增量编译技术需结合L3级别加速使用。
- 使用AOSP构建加速能力需要购买配套构建加速包，购买方法请参考**购买增值特性**。
- 该功能目前仅支持“华北-北京四”区域、代码源为CodeArts Repo的C/C++语言构建工程的编译构建加速。
- 本章节配置仅支持L1、L3级别的加速。
- 构建过程中不允许直接覆盖LD_LIBRARY_PATH和LD_PRELOAD环境变量，建议通过追加的方式使用，例如：
  ```
  "export LD_LIBRARY_PATH=new_path:${LD_LIBRARY_PATH}"
  ```

加速前准备

在一般的构建工程中，其构建过程大致分为构建前准备（工具链、代码仓）、构建依赖件准备（ninja文件生成）、编译构建、构建后操作（打包、检查等）。其中，构建加速介入编译构建阶段，对此前的构建过程中生成的构建依赖件进行解析，并执行编译。

在配置构建加速前，需如下准备：

- 找到构建依赖件准备的节点，以AOSP为例，一个形态的编译命令如下：
  ```
  source build/envsetup.sh
  lunch aosp_x86_64-eng
  make -j64
  ```
- 准备好构建使用的docker镜像，基于该docker镜像制作新镜像：在dockerfile中增加“/opt/buildtools”目录供加速工具部署，并确保构建用户对“/opt/buildtools”目录有权限写入。参考命令如下：
  ```
  RUN mkdir -p /opt/buildtools && chmod -R 777 /opt/buildtools
  ```


通过 BuildFlow 组织加速构建

构建加速需要结合多任务代码化构建使用，可参考**多任务YAML文件结构详解**中的部分配置。

BuildFlow配置方法如下样例：

```
buildflow:
  jobs_resolver: # 必配
    provider: tbuild_jobs_resolver # 必配，固定值
    jobs: # 需要进行编排的任务集
        - job: distribute_job # 构建任务名称
          build_ref: accelerate.yml # 指定构建加速脚本，脚本名称可自定义
          worker: 2 # 指定为16vCPU的倍数，例如2就代表使用了32vCPU进行加速
```

参数说明如下：

- jobs_resolver：buildflow的子节点，必配。
- provider：此处使用的provider为jobs_resolver的高级选项，意为指定job对应的任务解析器，取值固定为tbuild_jobs_resolver。
- jobs：需要进行编排的任务集，此处的jobs作为jobs_resolver的子节点，与普通构建场景buildflow下的jobs子节点有所区别，配置时请注意缩进。
- job：构建任务名称，可自定义。
- build_ref：该构建任务在构建过程中需要运行的加速构建脚本。
- worker：指定为16vCPU的倍数，例如2就代表使用了32vCPU进行加速。

配置示例 1：依赖解析模式

步骤1 获取json文件。

在分支A编写BuildFlow配置中build_ref指定的accelerate.yml，示例如下：

```
version: 2.0

params:
  - name: TB_GET_ORI_TRACE
    value: "1"

steps:
  PRE_BUILD:
  - checkout:
      name: "checkout"
      inputs:
        scm: "codehub"
        url: "git@codehub.devcloud.example.example.com:example.git"
        branch: "master"
        lfs: false
        submodule: false
  BUILD:
  - tbuild_execute:
      inputs:
        image: "swr.example.example.com/buildimage:AOSP"
        command: "cd AOSP && chmod a+x build.sh && BuildAccelerateL3 -AOSP ./build.sh && mv out/
TBTrace_droid_1.json /example/TB1.json"
```

步骤2 执行分布式构建。

在分支B编写BuildFlow配置中build_ref指定的accelerate.yml，示例如下：

```
version: 2.0

params:
```



```
  - name: TB_BUILDTRACE_ALL
    value: "1"
  - name: TB_RSYNC
    value: ${WORKSPACE}/AOSP/:out/target/product/generic_x86_64

steps:
  PRE_BUILD:
  - checkout:
      name: "checkout"
      inputs:
        scm: "codehub"
        url: "git@codehub.devcloud.example.example.com:example.git"
        branch: "master"
        lfs: false
        submodule: false
  BUILD:
  - tbuild_execute:
      inputs:
        image: "swr.example.example.com/buildimage:AOSP"
        command: "mv /example/TB1.json AOSP/TB1.json && cd AOSP && chmod a+x build.sh &&
BuildAccelerateL3 -AOSP ./build.sh && ./post_build.sh"
```

----结束

配置示例 2：产物分类模式

在分支C编写BuildFlow配置中build_ref指定的accelerate.yml，示例如下：

```
version: 2.0

params:
  - name: TB_RSYNC
    value: ${WORKSPACE}/AOSP/:out/target/product/generic_x86_64

steps:
  PRE_BUILD:
  - checkout:
      name: "checkout"
      inputs:
        scm: "codehub"
        url: "git@codehub.devcloud.example.example.com:example.git"
        branch: "master"
        lfs: false
        submodule: false
  BUILD:
  - tbuild_execute:
      inputs:
        image: "swr.example.example.com/buildimage:AOSP"
        command: "cd AOSP && chmod a+x build.sh && BuildAccelerateL3 -AOSP ./build.sh && ./post_build.sh"
```

params 参数项详解

params配置项指定了一些使用AOSP构建加速必配的参数，参数说明如下：

参数项	是否必填	说明	示例
TB_RSYNC	是	需要同步的产物文件目录，“:”前为根目录，“:”后为若干个以“,”分隔的子目录，子目录前带“!”代表此目录不同步，不带“!”代表此目录的所有文件会被同步，带“!”的优先级更高。在AOSP构建场景下，需要拼接为“${WORKSPACE}/实际目录”。	"${WORKSPACE}/AOSP/:out/target/product/generic_x86_64,\!out/target/product/generic_x86_64/obj,\!out/target/product/generic_x86_64/symbols,\!out/target/product/generic_x86_64/obj_x86,\!out/target/product/generic_x86_64/obj_arm,\!out/target/product/generic_x86_64/gen"
TB_GET_ORI_TRACE	**配置示例1：依赖解析模式**必填	依赖解析模式下获取当前工程的依赖json文件开关。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_BUILD TRACE_ALL	**配置示例1：依赖解析模式**必填	依赖解析模式开关，不设置时默认使用**配置示例2：产物分类模式**。 ● 1：开启。 ● 不设置：关闭（默认）。 ● 非“1”的其他字符串：开启，字符串视为json文件的自定义路径和名字。	AOSP/aosp.json

steps 参数项详解

steps配置项定义了构建过程，示例中包含如下两个步骤：PRE_BUILD（构建前准备）和BUILD（编译构建）。

- PRE_BUILD

  此阶段主要做代码下载，参数解释如下：

```
PRE_BUILD:
- checkout: # 代码下载步骤
    name: "代码下载" # 步骤名称，可自定义
    inputs: # 步骤参数
      scm: "codehub " # 代码来源:只支持Repo
      url: "git@codehub.devcloud.example.example.com:test/python3.git" # 拉取代码的ssh地址。
      branch: "master" # 拉取的代码分支。
      lfs: false # 选择是否开启Git LFS，false关闭、true开启。构建默认不拉取音视频、图像等大型文件，开启Git LFS后，构建将会全量拉取文件。
      submodule: false # false关闭、true开启。开启该功能，系统在构建时会自动拉取子模块仓库的代码。
```

- BUILD

  此阶段主要定义了download_artifact插件、tbuild_execute插件和upload_artifact插件，参数解释如下：

```
BUILD:
  - tbuild_execute: # AOSP加速场景下固定配置，定义tbuild_execute插件
      inputs: # 固定配置
        image: "swr.xx-xx-x.myxxcloud.com/buildimage:AOSP" # 构建使用的镜像，参考加速前准备章节制作docker镜像。
        command: "mv /example/TB1.json AOSP/TB1.json && cd AOSP && chmod a+x build.sh && BuildAccelerateL3 -AOSP ./build.sh && ./post_build.sh"
# command为构建使用的命令，此处将构建分解为两个段落，准备和执行
# mv /example/TB1.json AOSP/TB1.json 是依赖解析模式独有的准备步骤，文件名字固定，如果工程中存在多个ninja构建，则文件的下标依次增加，例如TBTrace_droid_2.json和TB2.json，以此类推。
# 准备阶段:在代码仓根目录新建build.sh，内容见build.sh样例
# 构建阶段:依照加速级别调用加速命令（BuildAccelerateL1 BuildAccelerateL3）的AOSP模式（-AOSP）直接执行构建，此处样例取值BuildAccelerateL3 -AOSP
# 后处理阶段:以实际工程需要为准，该示例仅使用post_build.sh
# 依赖解析模式实际命令最终拼接为"mv /example/TB1.json AOSP/TB1.json && cd AOSP && chmod a+x build.sh && BuildAccelerateL3 -AOSP ./build.sh && ./post_build.sh"
# 产物分类模式实际命令最终拼接为"cd AOSP && chmod a+x build.sh && BuildAccelerateL3 -AOSP ./build.sh && ./post_build.sh"
```

  build.sh样例：

```
source build/envsetup.sh
lunch aosp_x86_64-eng
make -j64
```

高级选项

高级选项均为非必填选项，在构建过程中有工程无法执行需要特殊适配或优化性能时配置，若随意配置可能会导致构建失败。

表 8-4 通用选项

参数项	说明	示例
TB_GET_TRACE	构建结束后获取依赖json文件文件的开关。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_NINJA_RULE_ALL	用于产物分类模式自定义Target切割及分发，使用逗号分隔需要切割至不同分组的编译产物类别，使用冒号分隔需要切割至同一分组的编译产物类别。使用星号分隔多个ninja工程的配置。不设置时会自动配置合适值。	nonSystem:fonts:media:usr:system_ext,bin_other1:bin:bin_other2:lib:lib64,apex:system_ext_apex:apex_1:system_ext_apex_1:fake_packages:packaging_script,framework:priv-app:priv-app2:priv-app3:app,host,apex_0:system_ext_apex_0,apex_2:system_ext_apex_2:vendor,product:etc*nonSystem

参数项	说明	示例
TB_TARGETS_LIST_ALL	人工指定分发的target进行编译，每个逗号隔开不同worker的target，每个空格隔开同一个worker的不同target。使用星号分隔多个ninja工程的配置。不设置时会自动配置合适值。	"nonSystem_target,framework_target,lib_target*lib_target"
TB_HOOK_LOCK	对软链接也进行文件同步。若构建过程中发生软链接文件未同步导致的报错，需要开启此选项。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_APPEND_PATH	构建时可向PATH环境变量中追加的参数。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_SHUT_DOWN_SAME_TIME	所有worker都等待主节点执行完毕后再结束构建释放资源。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_RSYNC_LOCK	构建加速的同时worker向构建执行机实时传输文件。开启后效率会进一步提升，但会存在概率性编译失败的情况。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_MAKE_J	设置构建并发数。默认为worker核数。	16
TB_REFER_NINJA_FILE	如果存在串行执行一个一模一样的ninja工程时，可以使用此变量优化构建速度 ● 1：开启。 ● 0：关闭（默认）。	1
CCACHE_DIR	自定义编译缓存的本地目录。默认为/tmp/xcache目录。	${WORKSPACE}/TBcache
TB_CACHE_SIZE	使用自定义执行机时自定义编译缓存的本地目录存储大小上限。默认为100G。	100G
CCACHE_MAXSIZE	自定义编译缓存的本地目录存储大小上限。默认为20G。	100G
TB_ONE_WORKER	使用自定义执行机且只使用一个worker进行构建时可以使用此变量打开编译缓存开关。 ● 1：开启。 ● 0：关闭（默认）。	1

参数项	说明	示例
NT	该变量在主节点client自动设置，可以通过比较此变量是否等于1判断该节点是否为主节点client。	不需要配置。
TB_NET_INTERFACE_NAME	指定获取IP时读取的网卡名，在多网卡情况下获取IP使用，默认为空，多个网卡名通过逗号分隔，配置在前的网卡名有更高的优先级。	eth0,eth1
TB_ACC_PREPARE	使用自定义执行机时必须配置此变量。	false
TB_OUTPUT_PATH	自定义产物目录路径，默认设置为out。	output
TB_SELF_ENV	worker编译target时使用本地环境变量，不使用client传递的变量。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_RSYNC_FLAG	增加同步文件时rsync命令的参数。	--ignore-existing -a
TBUILD_PLUGIN_PKG_TYPE	指定使用的TBuild版本为snapshot版本还是release版本。不设置时默认使用最新的release版本。	release
TBUILD_PLUGIN_PKG_VERSION	指定使用的TBuild版本号。不设置时默认使用最新的release版本。	1.0.1

表 8-5 依赖解析模式

参数项	说明	示例
TB_CPU_NUM	分发任务时，所有机器都以此值计算可分配任务量。默认为空。	16
TB_LOCAL_CONTAIN_PATTERN	根据关键字指定必须分发在client编译的target，此target生成的产物不需要文件传输回主节点。默认为空。	file_contexts.bin,com.android.vndk.current.,out/soong/host/linux-x86/bin/fileslist
TB_LOCAL_NOT_CONTAIN_PATTERN	根据关键字指定必须不分发在client编译的target。默认为空。	Bluetooth.so

参数项	说明	示例
TB_CAPACITY_ALL	单台机器可分配的任务权值上限，单位是分钟，以json文件的时间为参考，此时间为单纯的构建时间，不包括cpu空闲时间，实际构建时间会大于此值。可以指定为小数。超过此上限的target会分发至client，在所有agent编译完成后在本地最后编译。不设置时会根据json文件自动设置。	5.5
TB_TASK_SIZE_ALL	人工指定分割多少份target，可以大于机器数量，建议设置的值略大于机器数量，不建议少于机器数量，会导致机器浪费。不设置时会根据机器数量自动设置。	8

表 8-6 ninja 文件缓存选项

参数项	说明	示例
TB_CACHE_SERVER_IP	ninja文件缓存开关，和TB_CACHE_ARCHIVE_PATH同时设置时缓存才会开启。 ● 非空值：开启，如果开启了远端缓存，该值视为远端服务器IP。 ● 空值：关闭（默认）。	172.example.example.example
TB_CACHE_ARCHIVE_PATH	ninja文件缓存本地归档目录，和TB_CACHE_SERVER_IP同时设置时缓存才会开启。	AOSP/ninja_cache
TB_CACHE_RECACHE	本次构建会重新生成ninja文件缓存，不会命中历史缓存。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_CACHE_REMOTE	ninja文件缓存远端开关，命中时从远端获取缓存，生成缓存时也会归档至远端。 ● 1：开启。 ● 0：关闭（默认）。	1
TB_CACHE_LOCAL	ninja文件缓存本地开关，命中时从本地获取缓存，优先级高于远端缓存，生成缓存时也会归档至本地。 ● 1：开启（默认）。 ● 0：关闭。	1

参数项	说明	示例
TB_CACHE_DEPENDS	增加指定的文件作为缓存命中的依赖文件，如果该文件产生变化，会使缓存不命中。默认为空。多个文件使用逗号分隔。	build.sh,test.sh
TB_CACHE_VERSION	为缓存增加指定的版本号，如果版本号产生变化，会使缓存不命中。默认为空。	1.0
TB_CACHE_EXCLUDE_KEY	缓存时过滤掉带关键字的文件，多个关键字用逗号分隔，默认设置为“.glob”，可以用空字符串重置。	.so,.glob
TB_NINJA_FILE_CACHE	设置被缓存的目录，多个目录使用逗号分隔，默认设置为“out”。	out,test
TB_CACHE_NINJA_CACHE_SIZE	ninja文件缓存本地归档目录空间上限，超过此上限会根据算法自动清理历史缓存。默认设置为50，单位是G。	100
TB_SKIP_TARGET	归档ninja文件缓存时跳过此target，即使命中ninja文件缓存此target也会重新编译。	update-api

8.5 通过代码缓存方式实现构建加速

代码缓存是指通过一致性HASH、分布式文件存储、增量更新等技术，通过构建时代码下载效率的提升，从而实现构建加速。

约束与限制

- 仅代码化构建可使用代码缓存构建加速。
- 构建缓存只提供文件缓存的上传和下载检出功能，支持用户自定义脚本更新。
- 该功能目前仅支持“华北-北京四”区域、代码源为CodeArts Repo的C/C++语言构建工程的编译构建加速。
- 使用代码缓存加速需购买配套加速特性包，购买方法请参考**购买构建加速包**。

YAML 文件配置方法

参考**创建代码化构建使用的YAML文件**，在env中配置如下示例代码。

```
params: # 构建参数，可在构建过程中引用
- name: CLOUD_BUILD_UPLOAD_FLAG  # 参数为有值和为空两种状态，可控制跳过缓存上传至文件服务器
  value: true
- name: CLOUD_BUILD_REMOTE_CACHE # 参数为有值和为空两种状态，可控制会从其他执行机获取缓存
  value: true
env：
  cache: # 使用代码缓存
    - type:code #必填，使用缓存开关
      local_path:code # 必填，代码在构建执行机上存放的相对路径
      command:dos2unix build.sh && sh build.sh # 必填
      branch:master # 选填，分支名，可自定义，与url一起确定缓存标签
```



```
    url:git@codehub.devcloud.example.example.com:test/python3.git # 选填，可自定义，与branch一起确定缓存标签
```

表 8-7 配置参数表

参数项	参数类型	描述	是否必填
CLOUD_BUILD_UPLOAD_FLAG	String	用于控制是否跳过缓存上传文件服务器。 True：是。 为空：否。	否
CLOUD_BUILD_REMOTE_CACHE	String	用于控制是否从其他执行机获取缓存。 True：是。 为空：否。	否
type	String	使用缓存开关。	是
local_path	String	代码在构建执行机上存放的相对路径。	是
command	String	运行下载/更新代码的脚本命令。	是
branch	String	分支名，可自定义，与url一起确定缓存标签。	否
url	String	链接，可自定义，与branch一起确定缓存标签。	否

工作模式介绍

1. 代码缓存下载。本地没有缓存的情况下，从服务器下载缓存代码到本地；有缓存的情况下，使用本地缓存并还原目录结构。
2. 代码缓存更新。代码检出有变化时，会增量扫描目录树，缓存差异文件和目录树，加速下次构建缓存效率。

效果示例

在缓存盘有缓存的情况下，200万个文件（190G）还原时间在2到3分钟（仅供参考，实际效率受执行机规格、负载等影响）。

9 管理构建任务

您在操作编译构建任务前，需具备相应操作权限。

编辑构建任务

1. 通过项目入口方式**访问CodeArts Build服务首页**。
2. 在编译构建任务列表搜索目标任务。
3. 单击编译构建任务所在行···，在下拉列表中选择“编辑”，进入“编辑任务”页面。
   - 基本信息：可修改任务名称、代码源、代码仓、默认分支、任务描述等信息。
   - **构建步骤**：可修改构建步骤、步骤参数等信息。
   - **参数设置**：可配置执行任务时的自定义参数。
   - **执行计划**：可配置触发事件（持续集成）和定时执行。
   - 修改历史：可查看构建任务的修改记录。单击“比较差异”，可查看构建任务相比于上一次执行时调整的内容。
   - **权限管理**：可配置不同角色的权限。
   - **通知**：可配置任务事件类型通知信息（包括任务构建成功、失败、删除、配置更新、被禁用）。
4. 根据需要选择对应页签并进行编辑，单击“保存”完成修改。

删除构建任务

单击编译构建任务所在行···，在下拉列表中选择“删除”。请根据实际情况确定是否删除对应构建任务。

删除的构建任务可在构建任务回收站中查看。在编译构建首页右上角单击“更多”，在下拉列表选择“构建任务回收站”。

页面中展示已删除的构建任务，根据需要可以完成以下相关操作。

操作	说明
修改任务保留时间	单击“任务保留时间”下拉列表，根据需要选择时长，可选天数范围为1~30天。

操作	说明
搜索任务	在搜索框中输入待搜索内容，单击搜索，即可在页面中查看搜索结果。
删除任务	在列表中勾选待删除的任务，单击“删除”，即可将所选任务从回收站中删除。
恢复任务	在列表中勾选待恢复的任务，单击“恢复”，即可将所选任务恢复到编译构建服务的任务列表中。
清空回收站	单击“清空回收站”，可删除回收站中所有任务。

复制构建任务

1. 单击编译构建任务所在行的，在弹出的下拉列表选项单击“复制”，进入编译构建复制页面。
2. 根据需要修改任务信息，单击“保存”，即可复制该构建任务。
   单击“保存并执行”，可复制后执行执行该构建任务。

说明

复制任务会保留原任务的权限矩阵。

禁用任务

- 执行中的构建任务无法禁用和删除。
- 构建任务被禁用后，构建任务名称后会出现“已禁用”标识，此时不能再执行构建任务；如需执行，请单击构建任务所在行的，在下拉列表中选择“取消禁用”。

1. 单击构建任务所在行的，在下拉列表中选择“禁用”。
2. 弹出“任务禁用”提示框，输入禁用原因，单击“确定”。

收藏构建任务

- 收藏构建任务后，刷新页面或下次进入任务列表时，该任务会在任务列表中置顶显示，收藏多个任务会按任务创建时间降序排列。
- 收藏非自己创建的任务，可以根据该任务设置的通知事件类型获取相应的通知。

1. 鼠标移至任务所在行，单击，图标变色即收藏成功。
2. （可选）单击，即可取消收藏。

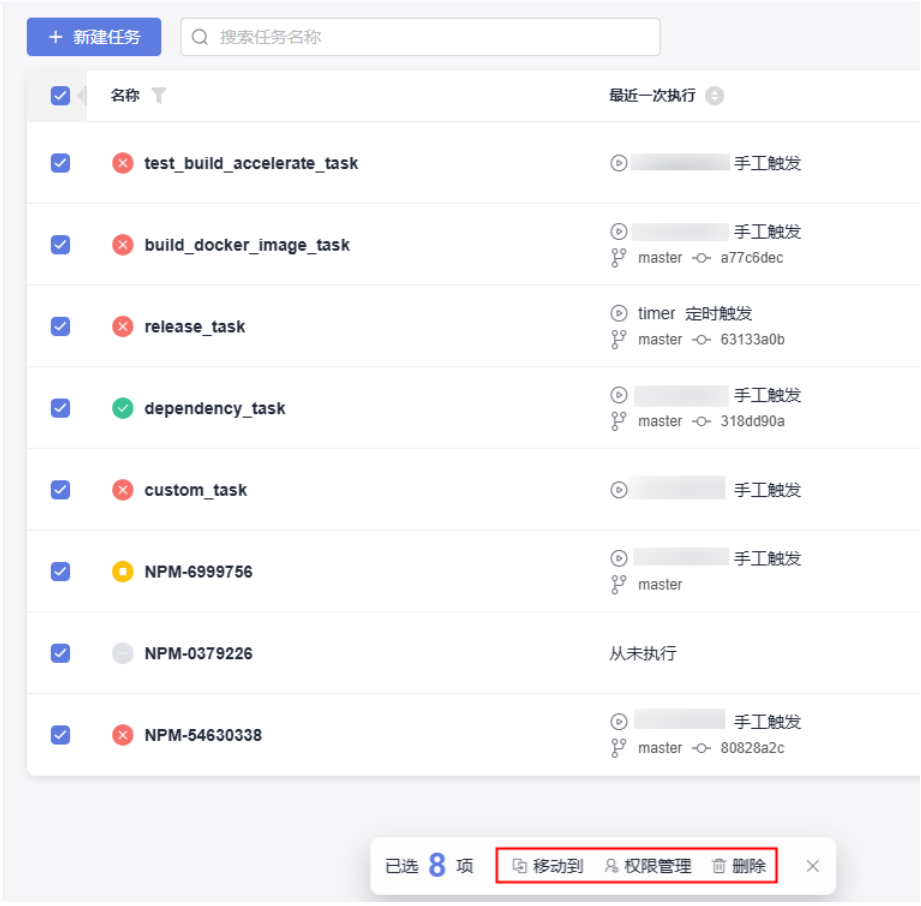
停止构建任务

1. 单击正在执行的构建任务名称，进入到“构建历史”页面。
2. 单击正在执行的“构建编号”。
3. 单击页面右上角“停止构建”，即可停止构建任务。

批量管理构建任务

勾选编译构建任务前的复选框，在弹出的窗口中单击“移动到”，可批量移动构建任务所在分组；单击“权限管理”，可批量设置构建任务各个角色的权限，单击“删除”，可批量删除构建任务。

图 9-1 批量管理构建任务



10 查询审计日志

云审计服务是安全解决方案中专业的日志审计服务，记录了CodeArts Build的相关操作事件，方便您日后的查询、审计和回溯。

支持审计日志的操作

表 10-1 云审计服务支持 CodeArts Build 服务操作

操作名称	资源类型	事件名称
创建编译构建任务	CloudBuildsServer	createJob
执行编译构建任务	CloudBuildServer	buildJob
删除编译构建任务	CloudBuildServer	deleteJob
更新编译构建任务	CloudBuildServer	updateJob
禁用编译构建任务	CloudBuildServer	disableJob
解除禁用编译构建任务	CloudBuildServer	enableJob
上传keystore文件	CloudBuildServer	uploadKeystore
更新keystore文件	CloudBuildServer	updateKeystore
删除keystore文件	CloudBuildServer	deleteKeystore
初始化EFS目录和存储配额	CloudBuildCache	initEFSDirAndQuota
上传报告（包含单元测试和依赖分析）	CloudBuildReport	uploadReport
创建自定义模板	CloudBuildTemplateService	createCustomTemplate
删除自定义模板	CloudBuildTemplateService	deleteCustomTemplate
更新nextfs信息	nextfsInfo	updateNextfsInfo

操作名称	资源类型	事件名称
创建nextfs	nextfsInfo	createNextfsInfo
创建租户关联nextfs	tenantNextfs	createTenantNextfs
删除租户关联nextfs	tenantNextfs	deleteTenantNextfs
修改租户License信息	licenseInfo	updateLicenseInfo
创建租户License	licenseInfo	createLicenseInfo
创建代码缓存信息	codeCacheInfo	createCodeCacheInfo
删除代码缓存信息	codeCacheInfo	deleteCodeCacheInfo
创建代码缓存使用记录	cacheHistoryInfo	createCacheHistoryInfo
更新代码缓存使用信息	cacheHistoryInfo	updateCacheHistoryInfo

查看审计日志

用户需要在云审计服务CTS的管理控制台查询CodeArts Build服务的事件列表。详情请参考**查看审计事件**。

11 参考

11.1 YAML 文件语法配置说明

单任务构建代码示例

```
---
version: 2.0

#构建参数定义, 参数必须以name，value成对出现，不赋值默认为空字符串，引用方式为${申明的参数名称name}
params:
  - name: machineArch
    value: X86

#构建环境配置，env和envs配置为非必填项，二选一。当用户需要使用条件判断确定使用的主机规格与类型时，选择配置envs
env:
  resource:
    type: docker
    arch: X86
    class: 8U16G
    pool: Mydocker

envs:
  - condition: machineArch == 'ARM'
    resource:
      type: docker
      arch: ARM
  - condition: machineArch == 'X86'
    resource:
      type: docker
      arch: X86

#构建步骤
steps:
  PRE_BUILD:
    - checkout:
        name: checkout
        inputs:
          scm: codehub
          url: git@codehub.devcloud.cn-north-7.ulanqab.huawei.com:huang-test00001/maven.git
          branch: master
          commit: commitId
          lfs: true
          submodule: true
          depth: 100
```



```
      tag: tag
      path: test
 - manifest_checkout:
    name: "manifest"
    inputs:
      manifest_url: https://codehub.devcloud.xxxxxxx.ulanqab.huawei.com/IPD-xxxxxx/manifest.git
      manifest_branch: master
      manifest_file: default.xml
      path: dir/dir02
      lfs: true
      repo_url: https://codehub.devcloud.xxxxxxx.ulanqab.huawei.com/IPD-xxxxxx/git-repo.git
      repo_branch: master
      username: someone
      password: PASSWD
 - sh:
    inputs:
      command: echo ${machineArch}

BUILD: # 构建步骤
 - maven:
    name: Maven构建
    image: cloudbuild@maven3.5.3-jdk8-open
    inputs:
      settings:
        public_repos:
          - https://mirrors.huawei.com/maven
      cache: true
      unit_test:
        coverage: true
        ignore_errors: false
        report_path: "**/TEST*.xml"
        enable: true
        coverage_report_path: "**/site/jacoco"
      command: mvn package -Dmaven.test.failure.ignore=true -U -e -X -B
      check:
        project_dir: ./
        settings: ~/.m2/settings.xml
        param: ''
 - upload_artifact:
    inputs:
      path: "**/target/*.?ar"
      version: 2.1
      name: packageName
```

表 11-1 单任务语法配置说明

参数	类型	说明	是否必填
version	string	YAML文件版本号配置项，用于指定YAML文件版本号。固定值，目前只支持2.0。	是
params	map	全局参数配置项，参数必须以name和value成对出现，不赋值默认为空字符串，引用方式为${申明的参数名称name}。 示例中定义的参数，引用方式：作为参数输入时使用${machineArch}，作为condition条件判断时使用申明的参数名称machineArch。 ● name：参数名称。 ● value：参数名称对应的参数值。	否

参数	类型	说明	是否必填
env	map	构建环境配置项，与envs配置功能相同，两者配置其一即可，不支持条件语句condition。使用示例可参考**配置构建环境**。 ● resource：构建环境资源信息。 ● type：资源池类型，可输入参数值（docker或custom），docker表示使用默认执行机，custom表示使用自定义执行机，必填项。 ● arch：构建执行机架构，可输入参数值（X86或ARM），必填项。 ● class：构建执行机规格，可输入参数值（2U8G、4U8G、8U16G、16U32G、16U64G），当type=docker时该参数生效，默认为2U8G，其他规格需要单独购买对应规格的并发包才能正常使用，非必填。 ● pool：自定义资源池名称，当type=custom时该参数生效，非必填。	否
envs	map	构建环境配置项，与env配置功能相同，两者配置其一即可，支持条件语句condition，可以更加灵活的根据不同场景使用同一个YAML文件。 ● condition：条件判断语句，符合当前条件判断的会使用对应resource配置的环境信息。 ● resource：构建环境资源信息。 ● type：资源池类型，可输入参数值（docker或custom），docker表示使用默认执行机， custom表示使用自定义执行机，必填项。 ● arch：构建执行机架构，可输入参数值（X86或ARM），必填项。 ● class：构建执行机规格，可输入参数值（2U8G、4U8G、8U16G、16U32G、16U64G），当type=docker时需要填写该参数，默认为2U8G，其他规格需要单独购买对应规格的并发包才能正常使用，非必填。 ● pool：自定义资源池名称，当type=custom时需要填写该参数，非必填。	否
steps	map	构建步骤执行配置项，配置构建流程，包括构建前准备，构建执行的具体任务。 ● PRE_BUILD：构建前准备工作配置项，一般用作于构建前的代码下载工作。 ● BUILD：构建任务配置项，用于执行业务相关的具体构建任务。	是

参数	类型	说明	是否必填
steps： PRE_BUILD	map	构建前准备工作配置项，一般用作于构建前的代码下载工作，目前只支持checkout、manifest_checkout 和sh配置项，一般情况配置其中一种即可。 • checkout：单代码仓下载。使用示例可参考**代码化构建（单仓下载）**。 – name：构建步骤名称，支持自定义名称，非必填，默认值“代码检出”。 – inputs：步骤输入参数，每个步骤的输入参数不同，详见具体**构建步骤**说明，必填项。 – scm：代码源，当前只支持codehub，非必填，默认值codehub。 – url：拉取代码的ssh地址或者https地址。codehub拉取代码时为ssh，其他代码源为https，必填项。 – branch：拉取的代码分支名，必填项。 – commit：commitId构建时拉取的commitId，非必填。 – lfs: 是否开启git lfs，非必填，默认值false。 – submodule：是否拉取子模块，非必填，默认值false。 – depth：浅克隆深度。选择commitId构建时，depth必须大于等于commitId所在深度，非必填，默认值1。 – tag：tag构建时拉取的tag名，如果commit和tag同时存在，优先执行commit构建，非必填。 – path：clone的子路径，代码将会下载到子目录下面，非必填。 • manifest_checkout: 多代码仓下载。使用示例可参考**代码化构建（manifest多仓下载）**。 – name：构建步骤名称，支持自定义名称，非必填，默认值“manifest_checkout”。 – inputs：步骤输入参数，每个步骤的输入参数不一样，详见具体**构建步骤**说明，必填项。 – manifest_url：指定manifest仓地址，包含xml文件的仓库，必填项。 – manifest_branch：指定manifest分支或revision，非必填，默认值HEAD。 – manifest_file：manifest文件路径，定义的多仓库必须为同一种源码源，非必填，默认值default.xml。 – path：自定义manifest所有子仓下载路径，为工作目录的相对路径路径，不能以“/”开头，不能包含“.”，非必填，默认值为当前工作目录。 – lfs：是否开启git lfs，非必填，默认值false。 – repo_url：repo仓库地址，非必填。	是

参数	类型	说明	是否必填
		– repo_branch：repo仓库分支，非必填，默认值stable。 – username：下载仓库时使用的用户名，下载非公开仓库时需填写，非必填。 – password：下载仓库时使用的密码，下载非公开仓库时需填写，非必填。 ● sh：执行shell命令。 – inputs：步骤输入参数，每个步骤的输入参数不一样，详见具体**构建步骤**说明，必填项。 – command：执行shell命令，当checkout或manifest_checkout无法满足业务诉求时，可以自定义编写shell命令进行构建前的准备工作，必填项。	

参数	类型	说明	是否必填
steps： BUILD	map	构建任务配置项，用于执行业务相关的具体构建任务，只支持特定构建步骤，构建步骤根据业务实际情况进行自由组合，具体构建步骤参考**选择构建步骤**。 ● maven：Maven构建步骤定义。 – name：构建步骤名称，支持自定义名称，非必填，默认值“Maven构建”。 – image：构建使用的容器镜像，支持自定义和默认镜像，默认镜像名称为“cloudbuild@”加上工具版本名称，工具版本名称可以参考**构建工具版本**，必填项。 – inputs：步骤输入参数，每个步骤的输入参数不一样，详见具体**构建步骤**说明，必填项。 – settings：maven构建的settings配置，非必填。 – public_repos：指定依赖包下载仓库地址。 – cache：是否开启缓存，非必填，默认值false。 – unit_test：单元测试，非必填。 – coverage：是否处理覆盖率数据，非必填，默认值false。 – ignore_errors：是否忽略单元测试错误，非必填，默认值true。 – report_path：单元测试数据路径，必填项。 – enable：是否处理单元测试数据，非必填，默认值true。 – coverage_report_path：覆盖率数据路径，非必填。 – command：执行构建命令，必填项。 – check：检查配置，非必填。 – project_dir：工程路径，必填项。 – settings：maven构建的settings路径，非必填。 – param：maven参数，非必填。 ● upload_artifact：上传二进制包至artifact仓库构建步骤定义。 – inputs：步骤输入参数，每个步骤的输入参数不一样，详见具体**构建步骤**说明，必填项。 – path：上传文件的路径及名称，支持通配符，必填项。 – version：版本名称，非必填，默认值以构建编号命名。 – name：文件名，非必填，默认值以文件原始名命名。	

多任务构建

```
---
version: 2.0
```



```
#构建参数定义, 参数必须以name，value成对出现，不赋值默认为空字符串，引用方式为${申明的参数名称
name}
params:
  - name: machineArch
    value: X86
  - name: jobCondition
    value: 1
  - name: jobsCondition
    value: 1

# 构建环境配置，env和envs配置为非必填项，二选一。当用户需要使用条件判断确定使用的主机规格与类型
时，选择配置envs
env:
  resource:
    type: docker
    arch: X86
    class: 8U16G
    pool: Mydocker
envs:
  - condition: machineArch == 'ARM'
    resource:
      type: docker
      arch: ARM
  - condition: machineArch == 'X86'
    resource:
      type: docker
      arch: X86

# buildflow和buildflows配置二选一。当需要使用条件判断执行的jobs时，选择配置buildflows
buildflow:
  strategy: Lazy
    jobs:
      - job: Job3
        depends_on:
          - Job1
          - Job2
        build_ref: .cloudbuild/build3.yml
      - job: Job1
        build_ref: .cloudbuild/build1.yml
      - job: Job2
        build_ref: .cloudbuild/build2.yml
buildflows:
  - condition: jobsCondition == 1
    jobs:
      - job: Job1
        build_ref: .cloudbuild/build1.yml
        params:
          - name: job1Params
            value: 1
      - condition: jobCondition == 1
        job: Job2
        build_ref: .cloudbuild/build2.yml
        params:
          - name: job2Params
            value: 2
      - job: Job3
        depends_on:
          - Job1
          - Job2
        build_ref: .cloudbuild/build3.yml
  - condition: jobsCondition == 2
    jobs:
      - job: Job3
        build_ref: .cloudbuild/build3.yml
```


表 11-2 多任务语法配置说明

参数	类型	说明	是否必填
version	string	YAML文件版本号配置项，用于指定YAML文件版本号。固定值，目前只支持2.0。	是
params	map	全局参数配置项，参数必须以name和value成对出现，不赋值默认为空字符串，引用方式为${申明的参数名称name}。 示例中定义的参数，引用方式：作为参数输入时使用${machineArch}，作为condition条件判断时使用申明的参数名称machineArch。 ● name：参数名称。 ● value：参数名称对应的参数值。	否
env	map	构建环境配置项，与envs配置功能相同，两者配置其一即可，不支持条件语句condition。 ● resource：构建环境资源信息。 ● type：资源池类型，可输入参数值（docker或custom），docker表示使用默认执行机，custom表示使用自定义执行机，必填项。 ● arch：构建执行机架构，可输入参数值（X86或ARM），必填项。 ● class：构建执行机规格，可输入参数值（2U8G、4U8G、8U16G、16U32G、16U64G），当type=docker时该参数生效，默认为2U8G，其他规格需要单独购买对应规格的并发包才能正常使用，非必填。 ● pool：自定义资源池名称，当type=custom时该参数生效，非必填。	否

参数	类型	说明	是否必填
envs	map	构建环境配置项，与env配置功能相同，两者配置其一即可，支持条件语句condition，可以更加灵活的根据不同场景使用同一个YAML文件。 ● condition：条件判断语句，符合当前条件判断的会使用对应resource配置的环境信息。 ● resource：构建环境资源信息。 ● type：资源池类型，可输入参数值（docker或custom），docker表示使用默认执行机， custom表示使用自定义执行机，必填项。 ● arch：构建执行机架构，可输入参数值（X86或ARM），必填项。 ● class：构建执行机规格，可输入参数值（2U8G、4U8G、8U16G、16U32G、16U64G），当type=docker时需要填写该参数，默认为2U8G，其他规格需要单独购买对应规格的并发包才能正常使用，非必填。 ● pool：自定义资源池名称，当type=custom时需要填写该参数，非必填。	否

参数	类型	说明	是否必填
buildflow	map	在CodeArts Build中Build Job是构建的最小单元，适用于业务比较简单的场景，但是在有些复杂的构建场景下，Build Job可能并不能满足复杂的构建要求，比如多仓工程需要分布到多个机器上去构建，并且构建工程之间还存在一定的依赖关系，又或者希望将Build Job中的构建任务拆分成更加模块化，更加细粒度的构建任务，并按照依赖顺序进行构建。对于比较复杂的构建场景，可以使用BuildFlow将多个有依赖关系的Build Job按照有向无环图（DAG）的方式组装起来，CodeArts Build将会按照构建的依赖关系以最大的并发进行构建，提升构建效率。 • strategy：定义buildFlow运行的策略，支持Lazy（构建时间相对较长，但是可以节省构建资源，在资源池资源不足时使用，优先触发优先级高的子任务构建）和Eager（构建时间相对较快，可能造成资源空闲等待，在资源池充足时推荐使用，同步触发所有任务），非必填，默认值Eager。 • jobs：任务编排，定义子job之间的依赖关系，必填项。 • job：子任务名称，必填项。 • depends_on：是否依赖子任务，填写依赖子任务的job名称，当前job依赖于Job1和Job2，非必填。 • build_ref：当前job构建使用的YAML文件路径（相对于仓库根目录），YAML文件是一个独立的可执行构建的完整文件，参考**单任务构建代码示例**，必填项。 • job：子任务名称，必填项。 build_ref：当前job构建使用的YAML文件路径（相对于仓库根目录），必填项。 • job：子任务名称，必填项。 build_ref：当前job构建使用的YAML文件路径（相对于仓库根目录），必填项。	

参数	类型	说明	是否必填
buildflows	map	当buildflow中需要做条件判断时使用buildflows，针对不同业务场景进行适配，更好的对YAML文件进行通用。 • condition：条件判断语句，符合当前条件判断的会使用对应jobs配置，condition必须放在buildflows下面。 • jobs：任务编排，定义子job之间的依赖关系，必填项。 • job：子任务名称，必填项。 build_ref：当前job构建使用的YAML文件路径（相对于仓库根目录），YAML文件是一个独立的可执行构建的完整文件，参考**单任务构建代码示例**，必填项。 • params：子任务定义的参数，作用域为子任务引用的YAML文件，可以在子任务使用的YAML文件中引用此处定义的参数，非必填。 – name：参数名称。 – value：参数名称对应的参数值。 • condition：条件判断语句，符合当前条件判断的会使用对应job配置。 • job：子任务名称，必填项。 build_ref：当前job构建使用的YAML文件路径（相对于仓库根目录），必填项。 • params：子任务定义的参数，作用域为子任务引用的YAML文件，非必填。 name：参数名称。 value：参数名称对应的参数值。 • job：子任务名称，必填项。 depends_on：是否依赖子任务，填写依赖子任务的job名称，当前job依赖于Job1和Job2，非必填。 build_ref：当前job构建使用的YAML文件路径（相对于仓库根目录），必填项。	

11.2 缓存目录使用说明

CodeArts Build在部分构建步骤中提供了依赖缓存的能力，能够极大提升用户构建时依赖包的下载效率，进而提升构建效率。用户在执行构建任务时，CodeArts Build在构建任务执行机上以租户维度进行远端缓存目录挂载，构建时直接使用，无需重复下载。当前支持缓存能力的构建步骤**表11-3**。

须知

执行缓存清理操作前，请务必仔细阅读以下缓存清理风险以及注意事项：

- 由于缓存目录为同租户共享，频繁清理缓存会概率性导致同租户用户构建异常（常表现为“xxx文件不存在”），因此只可在缓存异常时清理，任务执行成功后务必再次编辑任务，删除清理命令，并且在执行清理缓存操作的同时，不要执行其他的使用缓存的编译构建任务。
- 出于安全考虑，缓存清理命令只可在对应构建步骤里执行，在其他步骤执行此命令会导致“目录不存在”或”清理无效”等报错。

表 11-3 各构建步骤缓存目录使用说明

构建步骤	缓存目录（只能填写绝对目录，禁止填写“./”开头的相对目录）	缓存使用方式	清理缓存命令
Maven构建	/repository/local/maven	图形化配置方式，参考**使用Maven构建**。	rm -rf /repository/local/maven/{groupId}/{artifactId}/{version}，需填入的参数分别对应依赖包坐标中的groupId、artifactId和version，其中groupId中的点会被自动分割为层级目录。 若依赖包如下： <dependency> <groupId>com.codearts.java</groupId> <artifactId>demo</artifactId> <version>1.0-SNAPSHOT</version> </dependency> 则清理该依赖包所需命令为：**rm -rf /repository/local/maven/com/codearts/java/demo/1.0-SNAPSHOT**
NPM构建	/npmcache	构建命令行输入：npm config set cache /npmcache	npm cache clean --force
Grunt构建	/npmcache	构建命令行输入：npm config set cache /npmcache	npm cache clean --force
gulp构建	/npmcache	构建命令行输入：npm config set cache /npmcache	npm cache clean --force

构建步骤	缓存目录（只能填写绝对目录，禁止填写“./”开头的相对目录）	缓存使用方式	清理缓存命令
Android快应用构建	/npmcache	构建命令行输入：npm config set cache /npmcache	npm cache clean --force
Yarn构建	/npmcache	构建命令行输入：yarn config set cache-folder /npmcache	yarn cache clean
Gradle构建（仅限使用gradle wrapper版本）	./gradle/wrapper	构建命令行输入：cp /cache/android/wrapper/gradle-wrapper.jar ./gradle/wrapper/gradle-wrapper.jar	rm -rf ./gradle/wrapper/
Android构建（仅限使用gradle wrapper版本）	./gradle/wrapper	构建命令行输入：cp /cache/android/wrapper/gradle-wrapper.jar ./gradle/wrapper/gradle-wrapper.jar	rm -rf ./gradle/wrapper/

12 旧版手册页面

12.1 Android APK 签名

通过“Android APK签名”构建步骤，使用apksigner对Android APK进行签名。

图形化构建

1. **配置构建步骤**时，在“Android构建”步骤后添加“Android APK签名”步骤。
   参数说明如下：

参数	说明
步骤显示名称	构建步骤的名称，可自定义修改。
需要签名的APK路径	Android构建后生成要签名的.apk文件位置，支持正则表达式，如：可以使用build/bin/*.apk匹配构建出来的APK包。
Keystore文件	用于签名的Keystore文件，参考**生成Keystore签名文件**制作，单击下拉列表，展示**文件管理**页面已经上传的Keystore文件，请根据需要选择。
keystore password	密钥文件密码。
别名（Alias）	密钥别名。
key password	密钥密码。
apksigner命令行	用户自定义签名参数，默认“--verbose”显示签名详情。

2. 验证签名是否成功。
   配置完成后执行构建任务，当显示任务执行成功后，查看构建日志，若“Android APK签名”对应日志中显示“结果: Signed”即为签名成功。

代码化构建

```
version: 2.0 # 必须是2.0
steps:
```



```
BUILD:
  - android_sign:
        inputs:
          file_path: build/bin/*.apk
            keystore_file: androidapk.jks
            keystore_password: xxxxxx
            alias: keyalias
            key_password: xxxxxx
            apksigner_commond: --verbose
```

参数名	参数类型	描述	是否必填	默认值
file_path	string	需要签名的APK路径。	是	无
keystore_file	string	Keystore文件名。	是	无
keystore_password	string	Keystore文件密码。	否	无
alias	string	别名。	是	无
key_password	string	密码。	否	无
apksigner_commond	string	apksigner命令。	是	无

12.2 下载文件管理的文件

文件管理主要用来存储Android APK的签名文件和Maven构建settings.xml文件并提供对这类文件的管理（如：新建、编辑、删除、权限设置），上传文件的操作可参考**文件管理**。通过配置“下载文件管理的文件”构建步骤，可以将“文件管理”的文件下载到工作目录并使用。

图形化构建

在**配置构建步骤**中，添加“下载文件管理的文件”构建步骤。

参数说明如下：

参数项	说明
步骤显示名称	构建步骤的名称，可自定义修改。
工具版本	根据需要选择工具版本。
下载文件	• 单击下拉列表，选择文件管理已上传的文件。 • 单击“上传”，可以将本地文件上传到文件管理。 • 单击“管理文件”，可跳转至“文件管理”页面对文件进行管理。

代码化构建

```
version: 2.0 # 必须是2.0
steps:
  BUILD:
    - download_file:
        inputs:
          name: android22.jks
```

参数名	参数类型	描述	是否必填	默认值
name	string	文件名称。	是	无

12.3 文件管理

文件管理主要用来存储**Android APK的签名文件**和**Maven构建**settings.xml文件并提供对这类文件的管理（如：新建、编辑、删除、权限设置）。

约束限制

- 文件大小限制为100k。
- 文件类型限制为：**.xml、.key、.keystore、.jks、.crt、.pem。**
- 最多支持上传20个文件。

上传文件

1. **访问CodeArts Build服务首页**。
2. 单击“更多”，选择“文件管理”。
3. 单击“上传文件”。
4. 在弹出的窗口中选择文件，添加描述，勾选相关协议，然后单击“保存”。

文件管理

文件上传后，可以编辑文件、下载文件、删除文件、为用户配置文件操作权限。

- 在搜索框输入关键字，可搜索文件。
- 单击操作列，可修改文件名称，并设置是否允许租户内所有成员在编译构建中使用该文件。
- 单击操作列，可以下载文件。
- 单击操作列，在下拉框中选择“删除”，可根据弹框提示确认是否删除。
- 单击操作列，在下拉框中选择“编辑权限”，可在弹出的界面配置用户操作文件的权限。

表 12-1 文件管理角色权限说明

权限类型	拥有该权限的角色
添加用户	项目下所有用户。
查看	文件创建者、相同租户的用户。
使用	文件创建者、文件创建者配置了使用权限的用户。
更新	文件创建者、文件创建者配置了更新权限的用户。
删除	文件创建者、文件创建者配置了删除权限的用户。
编辑权限	文件创建者。

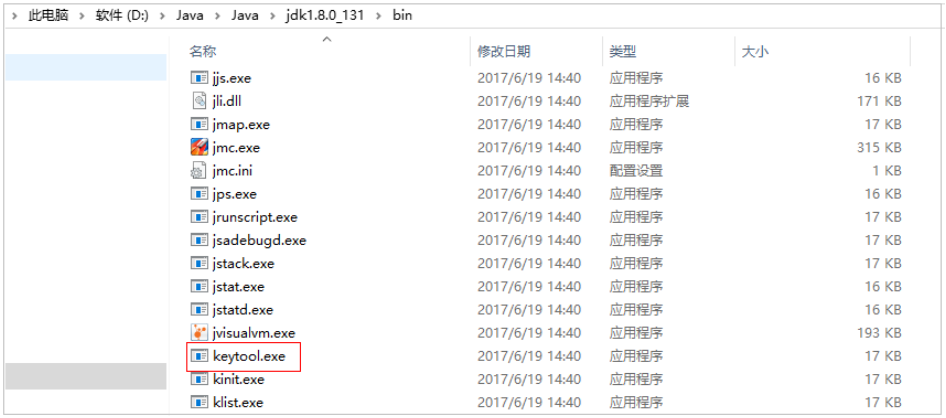
说明

创建者默认有所有权限并且不可被删除和修改。

生成 Keystore 签名文件

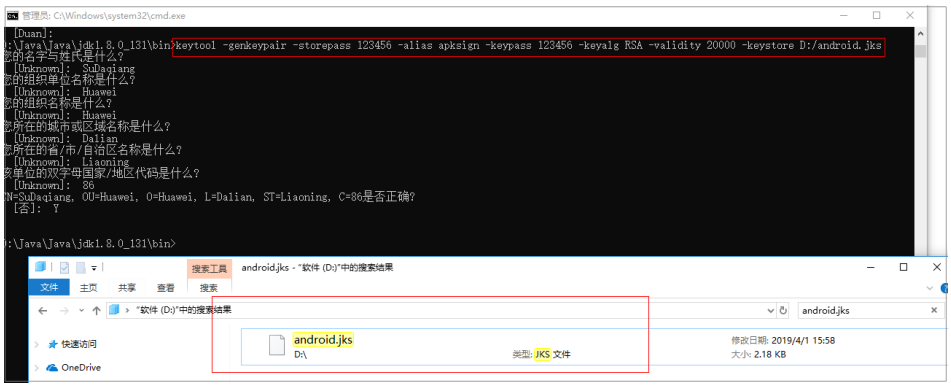
- **使用JDK的keytool工具生成签名文件**

  a. 找到JDK安装位置以及keytool。



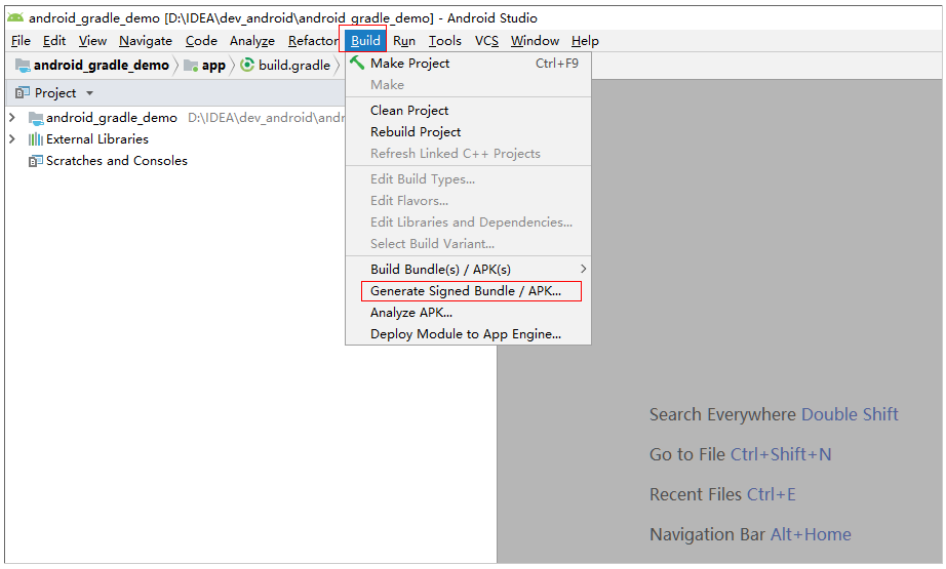
b. 执行生成密钥命令，生成.jks文件。

```
keytool -genkeypair -storepass 123456 -alias apksign -keypass 123456 -keyalg RSA -validity 20000 -keystore D:/android.jks
```



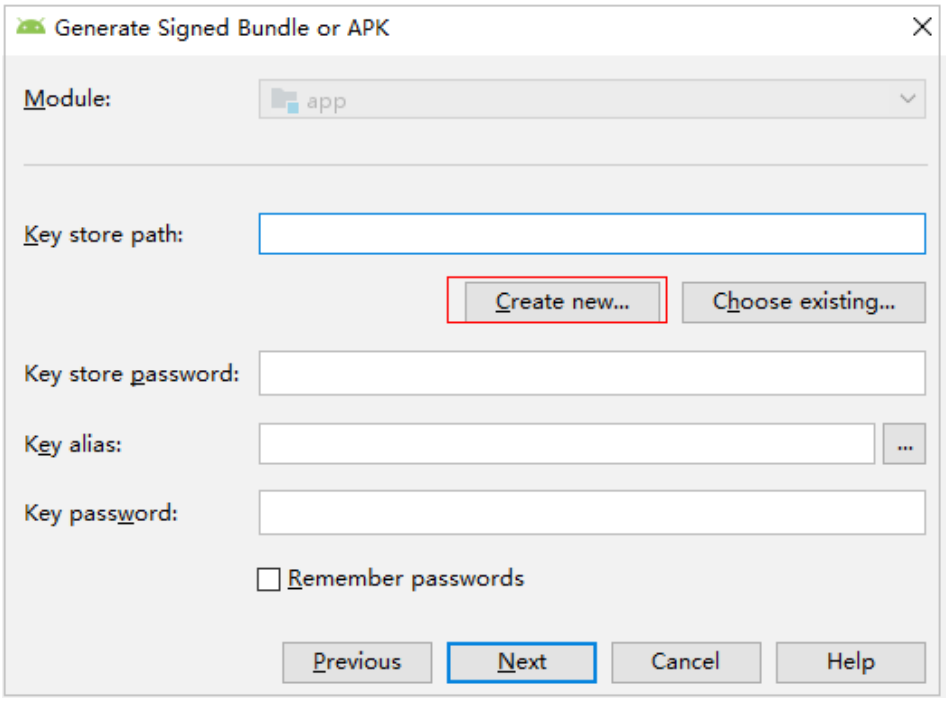
- **使用Android Studio生成签名文件**

a. 打开Studio，选择“Build下的Generate Signed Bundle/APK”。

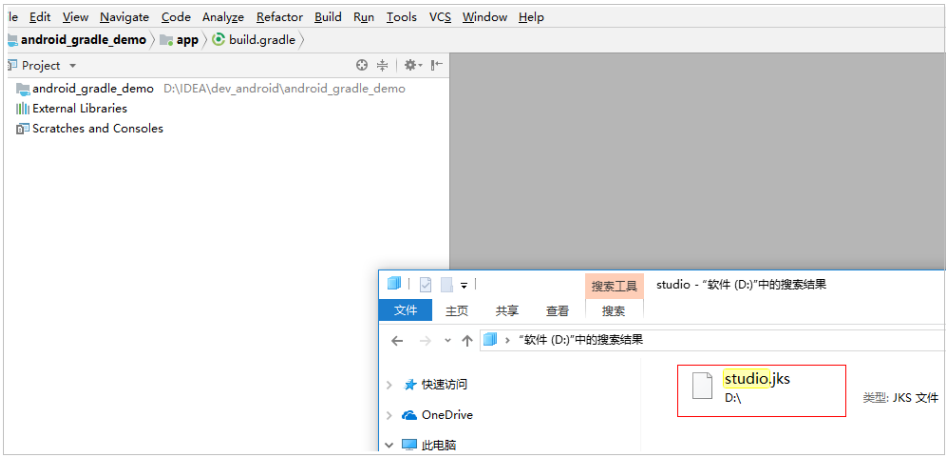


b. 选择“APK”，单击“Next”。

c. 单击“Create new...”，在弹出框填写相关信息，单击“OK”，然后单击“Next”。



d. 签名文件成功生成，查看文件。



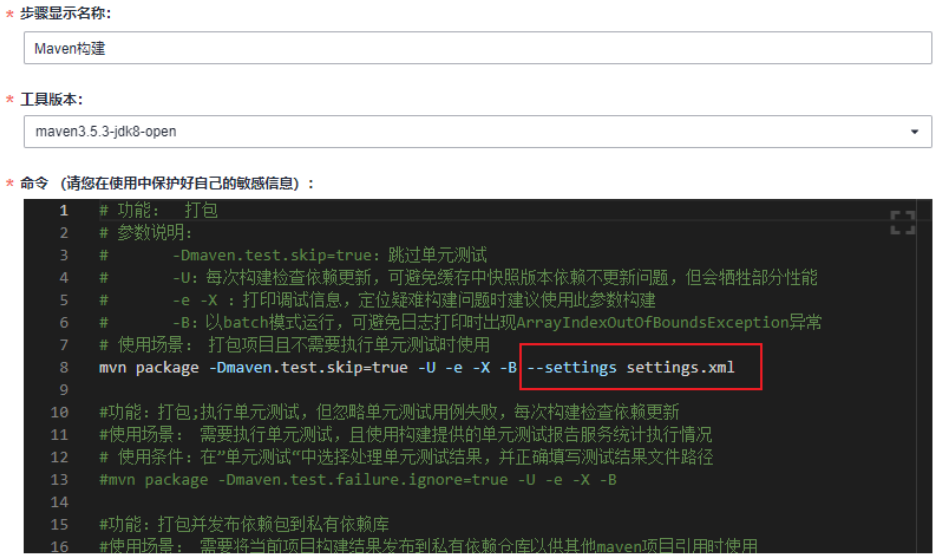
说明

生成的签名文件，可以上传到“文件管理”统一管理。

使用 settings.xml 文件

1. 新建或编辑Maven构建任务时，在“构建步骤”页签，添加“下载文件管理的文件”步骤，然后选择上传的settings.xml文件。

2. 在“Maven构建”默认命令末尾添加“--settings settings.xml”，即可使用已添加的settings.xml文件执行Maven构建。



12.4 自定义构建环境

背景信息

当常用的编译构建环境无法满足构建需求时，通过自定义构建环境提供的基础镜像，添加项目需要的依赖和工具，制作Dockerfile文件，然后**自定义构建环境**，再**使用自定义环境构建**。

基础镜像

编译构建使用centos7和ubuntu18作为基础镜像，并提供多种构建常用的配置环境工具，用户可以根据需要配置自定义构建环境。

内置环境工具如下：

jdk 1.8、maven、git、ant、zip、unzip、gcc、cmake、make。

操作步骤

步骤1 **访问CodeArts Build服务首页**。

步骤2 在编译构建首页右上角单击“更多”，在下拉列表选择“自定义构建环境”。

步骤3 进入自定义构建环境页面，选择合适的基础镜像，单击即可下载Dockerfile模板。

步骤4 编辑下载的Dockerfile文件。

可根据需要加入项目需要的其他依赖和工具，完成Dockerfile文件自定义，如下为添加了jdk和maven工具的示例。

```
RUN yum install -y java-1.8.0-openjdk.x86_64
RUN yum install -y maven
RUN echo 'hello world!'
RUN yum clean all
```

----结束

12.5 自定义模板

在**选择构建模板时**，当预置的构建模板无法满足构建需求时，可以选择自定义构建模板。

步骤1 登录编译构建服务首页。

步骤2 在列表中选择构建任务，单击任务名称进入“构建历史”页面。

说明

若列表中没有任务，请**新建构建任务（图形化构建）**。

步骤3 单击页面右上角，在下拉列表中选择“保存模板”。

说明

构建任务中包含私密参数则无法保存为模板。构建参数设置可参考**配置构建任务参数**。

步骤4 在弹框中输入模板名称与模板描述，单击“保存”。

步骤5 单击页面右上角用户名，在下拉菜单中选择“租户设置”。

步骤6 单击导航“编译构建 > 自定义模板”，即可在列表中看到已保存的构建模板。

对已保存的构建模板，可以完成以下操作：

表 12-2 管理自定义模板

操作	说明
收藏模板	单击，可以收藏该模板。
删除模板	单击，在弹框中单击“确定”，即可删除该模板。

----结束

12.6 编辑/删除/复制/收藏/停止构建任务

在操作编译构建任务前，需具备相应操作权限。

编辑构建任务

1. 登录编译构建服务首页。

2. 在编译构建任务列表搜索目标任务。
3. 单击编译构建任务所在行···，在下拉列表中选择“编辑”，进入“编辑任务”页面。
   - 基本信息：可修改任务名称、代码源、代码仓库、分支、任务描述等信息。
   - **构建步骤**：可修改构建步骤、步骤参数等信息。
   - **参数设置**：可配置执行任务时的自定义参数。
   - **执行计划**：可配置触发事件（持续集成）和定时执行。
   - 修改历史：可查看构建任务的修改记录。
   - **权限管理**：可配置不同角色的权限。
   - **通知**：可配置任务事件类型通知信息（包括任务构建成功、失败、删除、配置更新、被禁用）。
4. 根据需要选择对应页签并进行编辑，单击“保存并执行 > 保存”完成修改。

删除构建任务

1. 在编译构建任务列表搜索目标任务。
2. 单击编译构建任务所在行···，在下拉列表中选择“删除”，请根据实际情况确定是否删除对应构建任务。

删除的构建任务可到**构建任务回收站**中查看。

复制构建任务

1. 在编译构建任务列表中搜索目标任务。
2. 单击编译构建任务所在行的···，在弹出的下拉列表选项单击“复制”，进入编译构建复制页面。
3. 根据需要修改任务信息，单击“复制”，即可复制该构建任务。

说明

复制任务会保留原任务的权限矩阵。

收藏构建任务

1. 在编译构建任务列表搜索目标任务。
2. 鼠标移至任务所在行，单击☆，图标变色即收藏成功。
3. （可选）单击★，即可取消收藏。

说明

- 收藏构建任务后，刷新页面或下次进入任务列表时，该任务会在任务列表中置顶显示，收藏多个任务会按任务创建时间降序排列。
- 收藏非自己创建的任务，可以根据该任务设置的通知事件类型获取相应的通知。

停止构建任务

1. 在编译构建任务列表搜索目标任务。
2. 单击正在执行的构建任务名称，进入到“构建历史”页面。
3. 单击正在执行的“构建编号”。

4. 单击页面右上角“停止构建”，即可停止构建任务。

