持续集成(windows)

持续集成是一种软件开发实践，即团队开发成员经常集成他们的工作，通常每个成员每天至少集成一次，也就意味着每天可能会发生多次集成。每次集成都通过自动化的构建（包括编译，发布，自动化测试）来验证，从而尽快发现集成错误。许多团队发现这个过程可以大大减少集成的问题，让团队能够更快的开发内聚的软件。

简单来说，就是持续的定时的在多个团队成员的工作中进行集成，并且给予反馈。

为什么会要使用持续集成：

对重复的编译发布等操作进行抽象，减少重复过程。

及早发现各种冲突和错误，减少风险。

任何时间、任何地点生成可部署的软件。

［怎么做］

基本要求：要将这种实践付诸实际，需要一些必要的条件，如下

1.一个自动构建过程，包括自动编译、分发、部署和测试等

2.一个代码存储库，即需要版本控制软件来保障代码的可维护性，同时作为构建过程的素材库。

3.一个持续集成服务器。

自动化构建成过程，可帮助我们节省大量时间，完成这个过程的自动化后，在以后的开发过程中，我们需要做的，就是只是提交代码到版本库中，构建自动完成，基本不再需要人工干预。

代码仓库作为构建的素材库，构建所需的代码从代码库中获得。

最好有一台服务器单独作为持续集成服务器，一方面保证了环境的纯净，一方面不影响开发，而且持续集成服务器一般是随时准备开始构建的，所以一般也不关机。

［首先要有统一的代码库，服务器不断从版本控制服务器上检查代码状态，看代码是否有更新。如果发现有代码更新，那么就从版本控制服务器下载最新的代码。等代码完全更新以后，调用自动化编译脚本，进行代码编译。然后运行所有的自动化测试，并且进行代码分析。如果其中任何一个步骤失败，就表示build失败，持续集成服务器会给予响应的反馈。每次代码提交之后，都会在持续集成服务器上触发一个定时构建，然后进行编译、部署。］

［原则］

1.开发人员必须及时向版本控制库中提交代码，也必须经常性地从版本控制库中更新代码到本地；

2.需要有专门的集成服务器来执行集成构建。根据项目的具体实际，集成构建可以被软件的修改来直接触发，也可以定时启动，如每半个小时构建一次；

3.必须保证构建的成功。如果构建失败，修复构建过程中的错误是优先级最高的工作。一旦修复，需要手动启动一次构建。

4.不更新构建失败的代码

开发人员及时的提交代码进行构建是符合上述实践的，及时拉取代码可以防止工作中的分支偏离主干分支太多。定时触发构建或者通过检测代码的修改情况在触发构建都是可以的，主要是根及时的构建新的代码。如果构建失败，则必要及时处理导致失败的问题，修复后重新构建。当然构建失败的代码就不要拉到本地了，会污染一个本来是可以运行的工作区。、

［持续集成工具］

讲了这么多概念，有没一种工具把这种实践实现呢？当然有，常见的持续集成工具如下：

jenkins

travis

gitlab

buddybuild

仅列举了一些典型的，Jenkins 是传统型的工具，前身是 Hudson，04 年到现在已经有十多年的历史，后几个是最近几年出现的新一批，多少都和容器技术有点关系，这里我们主要介绍 Jenkins，因为这个工具比较常用，各种开发实践都可以通过大量的插件来组合实现，可定制性好很多。

Jenkins是一个可扩展的持续集成(CI)引擎。

之前叫Hudson。

Jenkins的作用

主要用于：

持续,自动地构建/测试软件项目。

监控一些定时任务。

Jenkins拥有的特性：

易于安装-只要把jenkins.war 部署到servlet容器，不需要数据库支持。

易于配置-所有配置都是通过其提供的web界面实现。

集成RSS/E-mail通过RSS发布构建结果或当构建完成时通过e-mail通知。

生成JUnit/testing报告。

分布式构建支持Jenkins能够让多台计算机一起构建测试。

文件识别：jenkins能够跟踪哪次构建生成哪些jar，哪次构建使用哪个版本的jar等。

插件支持：支持可扩展插件，可以开发审核自己团队使用的工具。

主要功能：

定时拉取代码并编译。

静态代码分析。

定时打包发布测试版

自定义额外的操作，如单元测试。

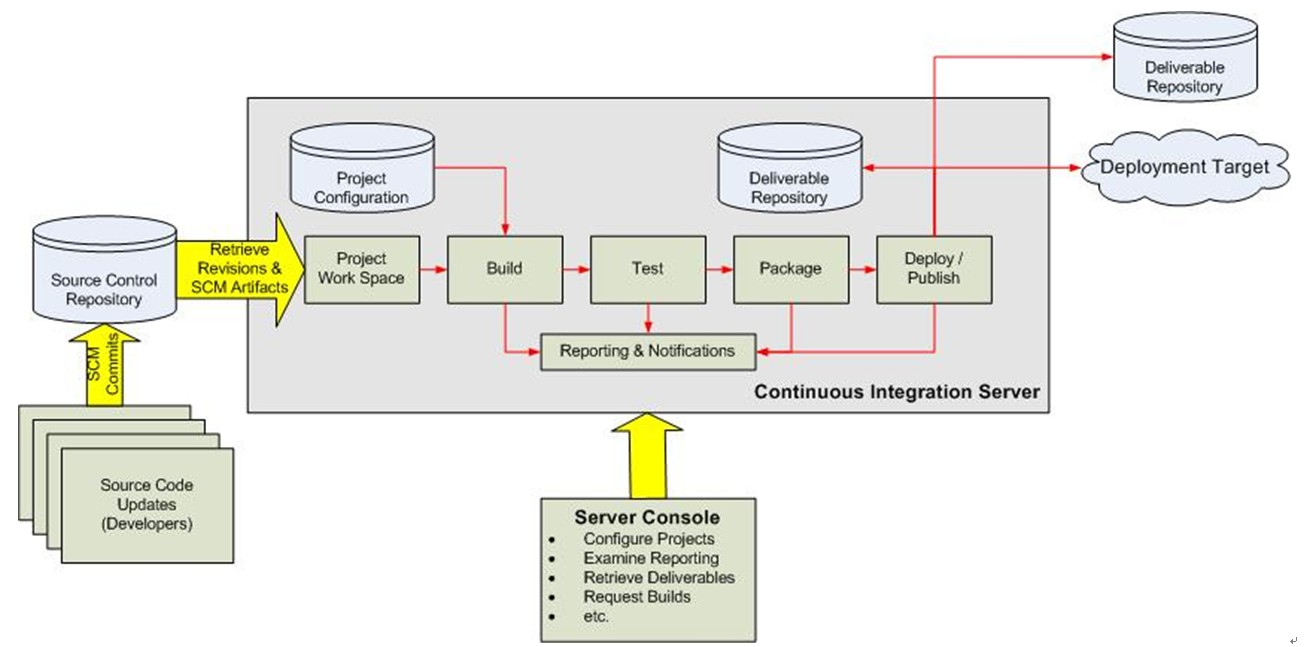
出错提醒。

Jenkins能做什么？

1. 软件构建自动化：配置完成后，CI系统会依照预先指定的时间表，或者针对某一特定的事件，对目标转件进行构建。
2. 构建可持续的自动化检测，CI系统能持续地获取新增或修改后签入的源代码，也就是说，当软件开发团队需要周期性的检查新增或修改后的代码时，CI系统会不断确认这些新代码是否破坏了原有软件的成功构建。这减少了开发者们在检查彼此相互依存的代码中变化情况需要花费的时间和精力(说直接一点也是钱啊，呵呵)。
3. 构建可持续的自动化测试：构建检查的扩展部分，构建后执行预先制定的一套测试规则，完成后触发通过（Email，RSS）等给相关的当事人。
4. 生成后后续过程的自动化：当自动化检测和测试完成，软件构建的周期可能也需要一些额外的任务，诸如生成文档，打包软件，部署构件到一个运行环境或软件仓库。这样，构件才能更迅速的提供给用户使用。

**部署一个CI系统需要的最低要求是：一个可获取的源代码的仓库，一个包含构建脚本的项目。**

CI系统的基本结构：



CI系统的工作顺序：

1. 开发者检入代码到源代码仓库。

2. CI系统会为每一个项目创建了一个单独的工作区。当预设或请求一次新的构建时，它将把源代码仓库的源码存放到对应的工作区。

3. CI系统会在对应的工作区内执行构建过程。

4. （配置如果存在）构建完成后，CI系统会在一个新的构件中执行定义的一套测试。完成后触发通知(Email,RSS等等)给相关的当事人。

5. （配置如果存在）如果构建成功，这个构件会被打包并转移到一个部署目标(如应用服务器)或存储为软件仓库中的一个新版本。软件仓库可以是CI系统的一部分，也可以是一个外部的仓库，诸如一个文件服务器或者像Java.net、 SourceForge之类的网站。

6. CI系统通常会根据请求发起相应的操作，诸如即时构建、生成报告，或者检索一些构建好的构件。

为什么选择jenkins：

1.是所有CI产品中在安装和配置上最简单的。

2.基于Web访问，用户界面非常友好、直观和灵活，在许多情况下，还提供了AJAX的即时反馈。

3.Jenkins是基于Java开发的(如果你是一个Java开发人员，这是非常有用的)，但它不仅限于构建基于Java的软件。

4.Jenkins拥有大量的插件。这些插件极大的扩展了Jenkins的功能；它们都是开源的，而且它们可以直接通过web界面来进行安装与管理

Jenkins的目标

Jenkins的主要目标是监控软件开发流程，快速显示问题。所以能保证开发人员以及相关人员省时省力提高开发效率。

CI系统在整个开发过程中的主要作用是控制：当系统在代码存储库中探测到修改时，它将运行构建的任务委托给构建过程本身。如果构建失败了，那么CI系统将通知相关人员，然后继续监视存储库。它的角色看起来是被动的；但它确能快速反映问题。

特别是它具有以下优点：

Jenkins一切配置都可以在web界面上完成。有些配置如MAVEN\_HOME和Email，只需要配置一次，所有的项目就都能用。当然也可以通过修改XML进行配置。

支持Maven的模块(Module)，Jenkins对Maven做了优化，因此它能自动识别Module，每个Module可以配置成一个job。相当灵活。

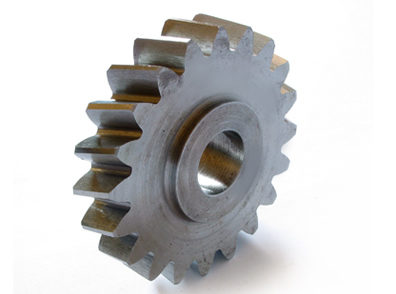
测试报告聚合，所有模块的测试报告都被聚合在一起，结果一目了然，使用其他CI，这几乎是件不可能完成的任务。

构件指纹(artifact fingerprint)，每次build的结果构件都被很好的自动管理，无需任何配置就可以方便的浏览下载。

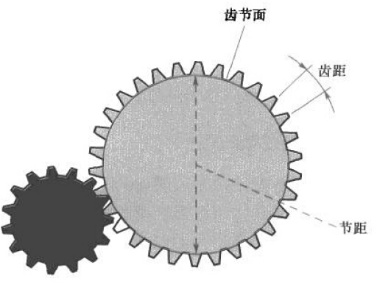
Jenkins原理：

持续集成原理：

将CI比喻成齿轮。



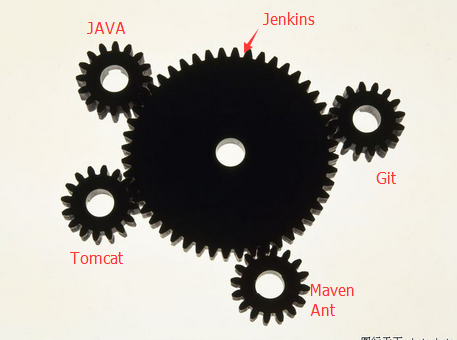
两个软件在一起可以驱动另外一个软件



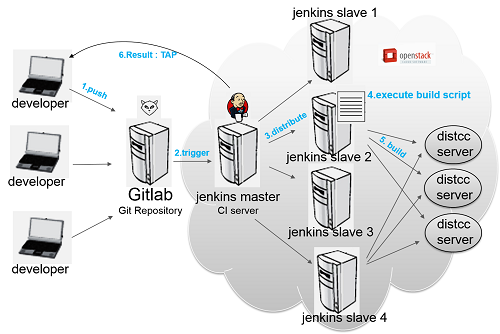
如果把这些软件要集成在一起工作，那么这个软件就可以存在其他软件的中间来驱动各个软件工作



jenkins就是类似中间那个齿轮，来驱动其他软件的集成一起工作



现在整个的开发过程就会变成如下图示：



1>开发者将新版本push到git server (Gitlab等版本控制系统)。

2>Gitlab随后触发jenkins master结点进行一次build。(通过web hook或者定时检测)

3>jenkins master结点将这个build任务分配给若干个注册的slave结点中的一个，这个slave结点根据一个事先设置好的脚本进行build。这个脚本可以做的事情很多，比如编译，测试，生成测试报告等等。这些原本需要手动完成的任务都可以交给jenkins来做。

4>我们在build中要进行编译，这里使用了分布式编译器distcc来加快编译速度。

\*\* Jenkins的工作原理是先将源代码从gitlab中拷贝一份到本地，然后根据设置的脚本进行build。我们可以看出，整个系统的关键是那个build脚本，用来告诉jenkins在一次集成中需要执行的任务。

Jenkins的安装配置

1. 下载最新的版本 <https://jenkins-ci.org/>
2. 运行 java –jar jenkins.jar

或者：

还有一种安装方式就是将下载的war包文件部署到 servlet 容器，然后启动容器，在浏览器的URL地址栏中输入类似http://localhost:8080/jenkins/这样的地址即可。

基础插件介绍：

Email Extension Plugin：邮件通知

Deploy to container Plugin：部署到java容器中的插件

Publish Over SSH: 远程SHELL

Maven Integration plugin: maven集成，

Monitoring： 监控Jenkins消耗的系统资源，如内存使用等。

将Jenkins部署到远程服务器

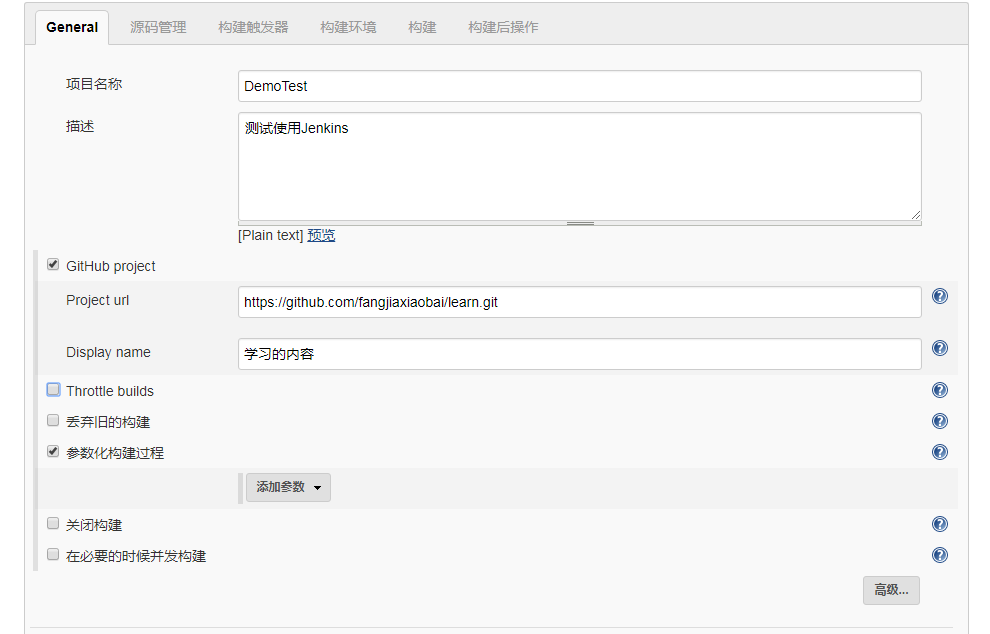
部署开发环境下的web项目到远程Linux服务器。

实现自动备份。

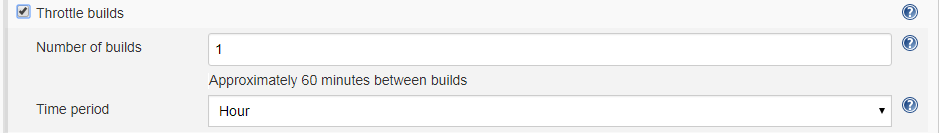
需要安装的插件，Publish Over SSH. 其他的插件 Pulish Over HTTP，Pulish Over FTP;

新建job

## General:



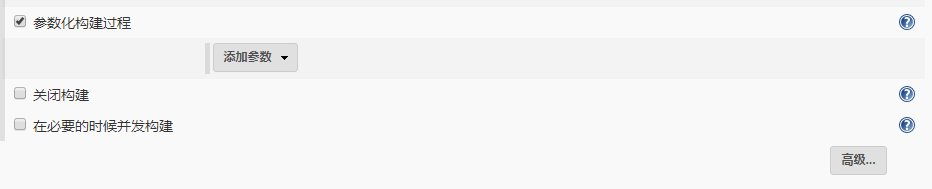
1. Throttle builds:节流构建：通过设置时间段内允许并发的次数来实现构建的控制。



1. 丢弃旧的构建：



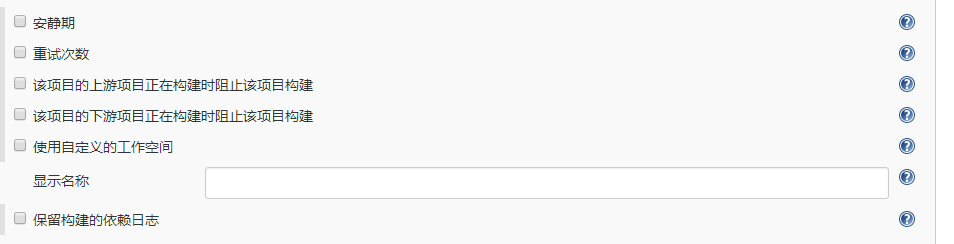
7.参数化构建过程：里面可以配置不同的参数，便于构建时引用这些参数



8.关闭构建

9.在并发时并发构建：满足策略要求时会进行并发构建。

10高级的内容：



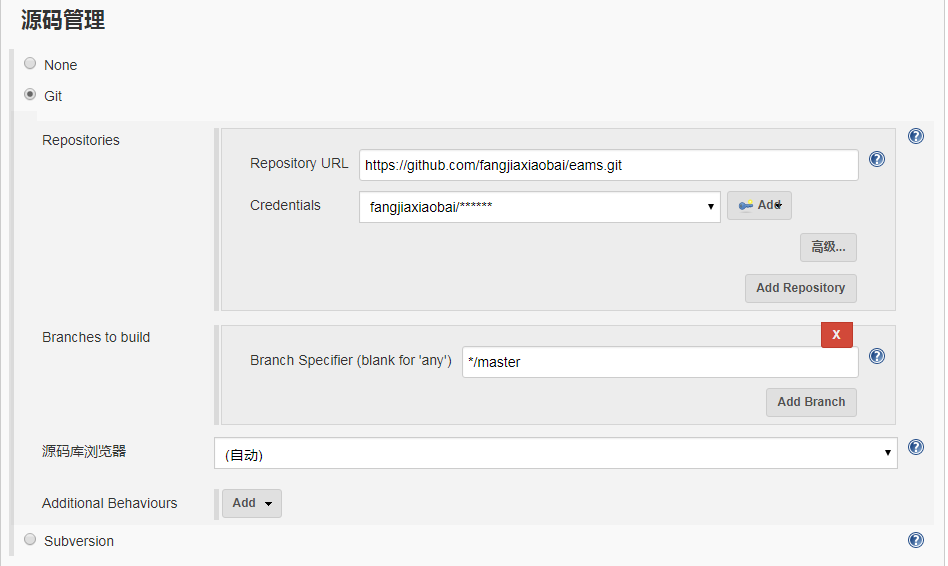
安静期：设置一个时间来间隔每次构建的问题。

重试次数：这个和系统设置一样，拉取源码重试的次数。

该项目的上游项目正在构建时阻止该项目构建 和该项目的上游项目正在构建时阻止该项目构建：用来阻止向下游项目有关联的构建策略。

使用自定义的工作空间：使该项目独立于系统的工作空间。

## 源码管理



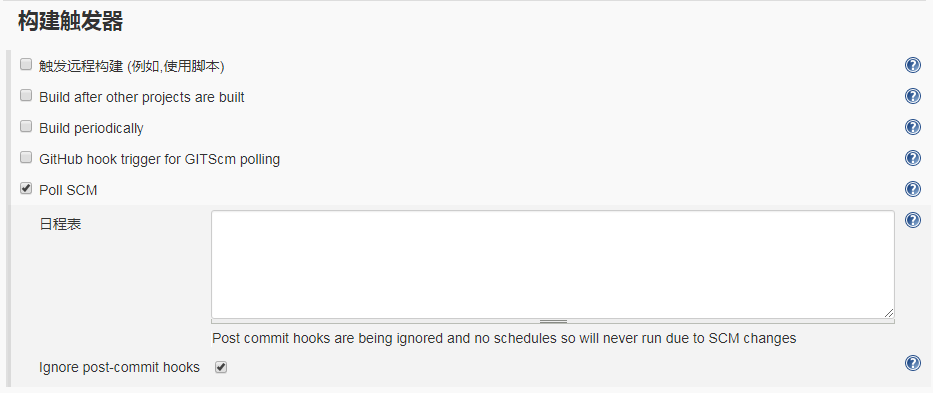
以git为例：

Respository URL: 仓库的地址。

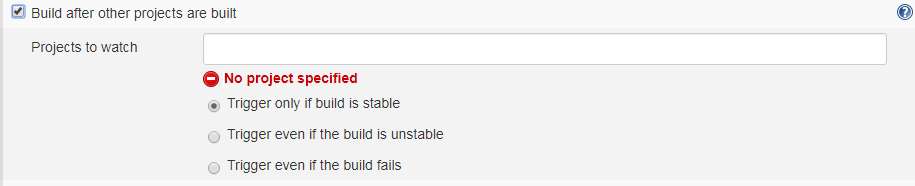
Credentials：拉取源码的用户名和密码。

源码库浏览器：默认就可以了。

## 构建触发器



1. 触发远程构建(如:使用脚本)：这里使用于自动化构建，拼接URL后写入代码中可以实现先脚本或者工具构建。
2. Build after other projects are built:构建与其他项目构建后，用于上下游项目有关联的时候。等其他项目构建完后在构建。



3.Build periodically：定时执行构建

日程表的参数

第一个参数代表的是分钟 minute，取值 0~59；

第二个参数代表的是小时 hour，取值 0~23；

第三个参数代表的是天 day，取值 1~31；

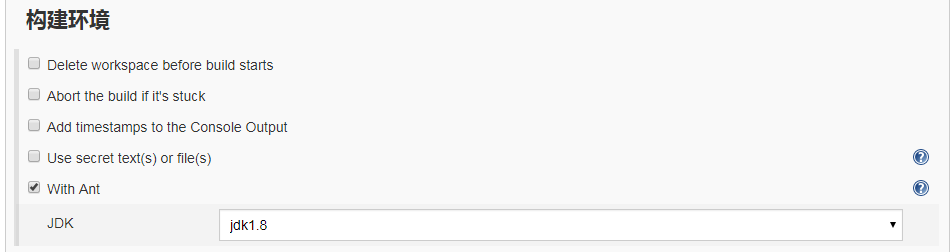
第四个参数代表的是月 month，取值 1~12；

最后一个参数代表的是星期 week，取值 0~7，0 和 7 都是表示星期天。

4. \*\*GitHub hook trigger for GITScm polling:\*\*hookplugin检测到源码的push操作触发构建,感觉Poll SCM更方便些,如果提交频繁,则这个触发就会频繁,看业务需要设置.

5. **Poll SCM:**定时感知代码分支是否有变化,如果有变化的话,执行一次构建.示例：H/5 \* \* \* \* 每五分钟去检查一下远程仓库,看代码是否发生变化。

## 构建环境：



1. **Delete workspace before build starts:**默认删除所有的,也可以设置删除特定的文件   
   - Patterns for files to be deleted:正则匹配删除哪些文件   
   - Apply pattern also on directories:规则是否也应用到文件夹   
   - Check parameter:是否删除,是个bool值,true则删除,false不删除.为毛感觉这个有点鸡肋   
   - External Deletion Command:执行外部删除命令
2. **Abort the build if it’s stuck:**构建阻塞的时候,根据超时策略处理.   
   - Time-out strategy:超时策略,有绝对时间,相对时间,根据以前的构建时间判断等   
   - Time-out variable:超时时间   
   - Time-out actions:超时后的处理,如终结,faile调或者写描述   
   - Add timestamps to the Console Output:在输出界面添加时间戳   
   - Use secret text(s) or file:使用密文,用于全局性的管理密码等,勾选后会在下方出现Binding,输入需要的用户名,密码证书等就可以了

## 构建

