

Jenkins 安装教程

CI&CD 工具，自动拉取 **gitlab** 或者 **svn** 代码，并调用编译脚本、发布至测试环境。

Java Version	Jenkins Version	status
jdk-11.0.24_linux-x64_bin.tar.gz	jenkins_v2.387.3_jdk11.war	Test OK
jdk-17.0.12_linux-x64_bin.tar.gz	jenkins_v2.462.1_jdk17.war	Test OK

安装 JDK

```
# 安装 jdk 17
tar zxf jdk-17.0.12_linux-x64_bin.tar.gz
mv jdk-17.0.12 /usr/local/
cd /usr/local
ln -s ./jdk-17.0.12 jdk

# 配置环境变量: vim /etc/profile.d/jdk.sh
#!/bin/bash
export JAVA_HOME=/usr/local/jdk
export PATH=$PATH:$JAVA_HOME/bin
export JENKINS_HOME=/opt/jenkins

# 生效
source /etc/profile.d/jdk.sh

java -version

java version "17.0.12" 2024-07-16 LTS
Java(TM) SE Runtime Environment (build 17.0.12+8-LTS-286)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.12+8-LTS-286, mixed mode, sharing)
```

安装 Jenkins

```
# 启动 jenkins
java -jar jenkins_v2.462.1_jdk17.war

备注：
    第一次启动会生成一组密码，访问 web 界面时需要输入， 存储文件目录通过环境变量 JENKINS_HOME 设置。
    释放的路径为 /root/.jenkins

# 启动 jenkins 并指定端口 8888
java -jar jenkins_v2.462.1_jdk17.war --httpPort=9999

# 查看帮助
java -jar jenkins_v2.462.1_jdk17.war --help
```

配置 jenkins

1. 首先访问 <http://192.168.198.132:8080> 输入验证码后，选择右边的 [选择插件来安装]

入门

解锁 Jenkins

为了确保管理员安全地安装 Jenkins，密码已写入到日志中（[不知道在哪里？](#)）该文件在服务器上：

```
/root/.jenkins/secrets/initialAdminPassword
```

请从本地复制密码并粘贴到下面。

管理员密码

继续

新手入门

自定义 Jenkins

插件通过附加特性来扩展 Jenkins 以满足不同的需求。

安装推荐的插件

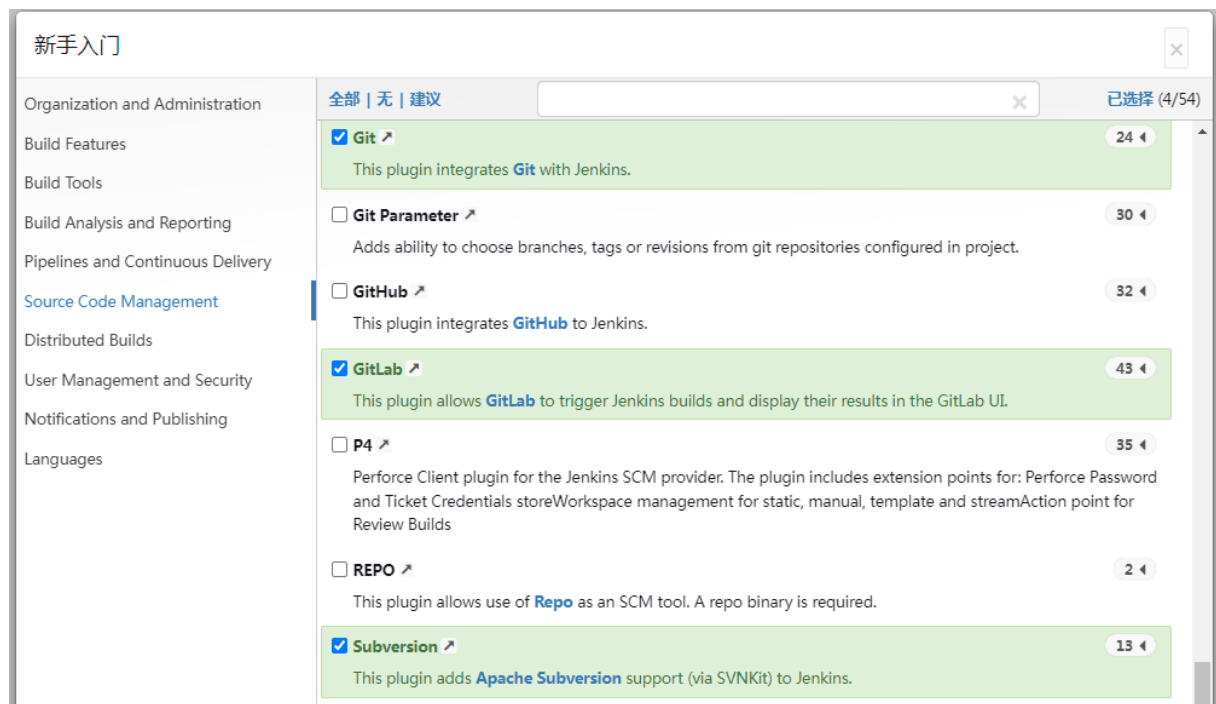
安装 Jenkins 社区推荐的插件。

选择插件来安装

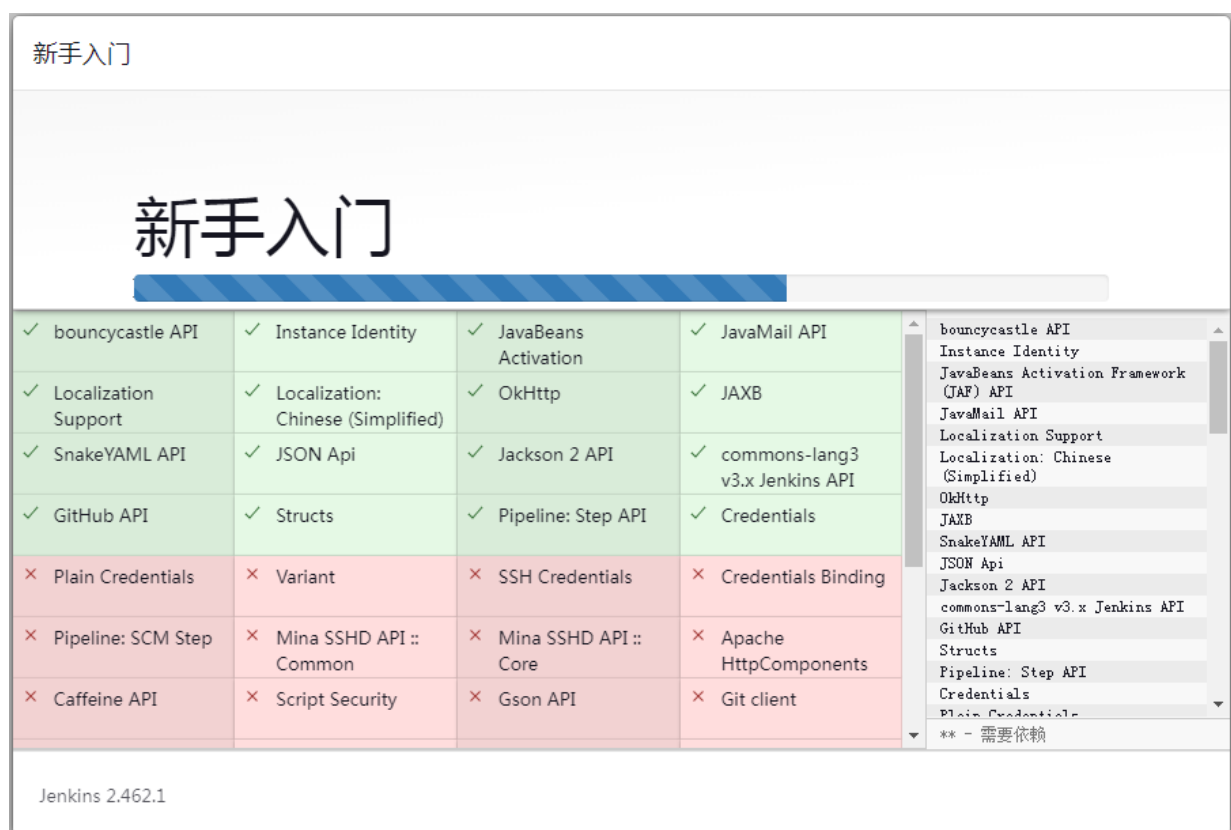
选择并安装最适合的插件。

Jenkins 2.462.1

2. 点击 [无] 取消默认选择的插件包，此处仅选择 [chinese, git, gitlab, svn] 其他均不选



3. 点击 [安装]



4. 如果网络不好则安装不上这些插件，可手工安装: <https://plugins.jenkins.io/>

也可以在界面设置中改成国内源进行安装:

<https://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json>

<http://mirror.xmission.com/jenkins/updates/update-center.json>

5. 创建管理员帐号密码

新手入门

创建第一个管理员用户

用户名

admin

密码

.....

确认密码

.....

全名

Jenkins 2.462.1

使用admin账户继续

保存并完成

新手入门

实例配置

Jenkins URL:

http://192.168.198.132:8080/

Jenkins URL 用于给各种Jenkins资源提供绝对路径链接的根地址。这意味着对于很多Jenkins特色是需要正确设置的，例如：邮件通知、PR状态更新以及提供给构建步骤的BUILD_URL环境变量。

推荐的默认值显示在尚未保存，如果可能的话这是根据当前请求生成的。最佳实践是要设置这个值，用户可能会需要用到。这将会避免在分享或者查看链接时的困惑。

Jenkins 2.462.1

现在不要

保存并完成

至此 `jenkins` 已安装完成，。

以下为 `jenkins` 主界面，可进入 [Manage Jenkins](#) -> [Plugins](#) -> [Advanced settings](#) 更改源地址。



Sign in to Jenkins

用户名

admin

密码

.....

☒ 保持登录状态

登录

Dashboard >

+ 新建Item

📁 构建历史

⚙️ Manage Jenkins

📌 My Views

构建队列

队列中没有构建任务

构建执行状态

1 空闲

2 空闲

✎️ Add description

欢迎来到 Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job



Set up a distributed build

Set up an agent



Configure a cloud



Learn more about distributed builds



[Jenkins 中文社区](#)

[REST API](#)

Jenkins 2.462.1

安装 gitlab:

```
# 使用脚本添加 gitlab-ce 的源

yum install curl
yum install openssh-server openssh-clients
yum install postfix
yum install policycoreutils
curl -sS https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh |
sudo bash

# 安装 gitlab-ce

yum install -y gitlab-ce

# 配置 gitlab-ce: vim /etc/gitlab/gitlab.rb

external_url 'http://192.168.198.132:8888'          -- nginx 监听端口
nginx['enable'] = true
redis['enable'] = true
redis_exporter['enable'] = false
prometheus['enable'] = false
prometheus_monitoring['enable'] = false
puma['enable'] = true
puma['port'] = 9090                                -- ruby 监听端口, 可能与 jenkins 冲突

# 重新编译配置

gitlab-ctl reconfigure

# 重启

gitlab-ctl restart

注: gitlab 默认使用 puma 作为 ruby 服务器, 使用 nginx 作为页面服务器, 初始密码保留 24 小时

默认 root 密码: /etc/gitlab/initial_root_password

默认安装路径在: /opt/gitlab/

默认页面路径在: /opt/gitlab/embedded/service/gitlab-rails/public

gitlab-ctl start                                # 启动
gitlab-ctl stop                                  # 停止
gitlab-ctl restart                              # 重启
gitlab-ctl status                               # 查看状态
vim /etc/gitlab/gitlab.rb                       # 修改配置
gitlab-rake gitlab:check SANITIZE=true --trace  # 检查 gitlab
gitlab-ctl tail                                  # 查看日志
```

注 1: -- 基本组件

ok: run: gitaly: (pid 54836) 0s	-- RPC 服务, 处理 git 操作
ok: run: gitlab-kas: (pid 54864) 1s	-- k8s 服务支持
ok: run: gitlab-workhorse: (pid 54874) 0s	-- 反向代理
ok: run: logrotate: (pid 54886) 1s	-- 日志切割工具
run: nginx: (pid 67485) 3092s;	-- nginx
ok: run: postgresql: (pid 54894) 0s	-- 数据库
ok: run: puma: (pid 54905) 0s	-- web 服务器
ok: run: redis: (pid 54911) 0s	-- 缓存客户端 session
ok: run: sidekiq: (pid 54918) 0s	-- 核心服务, 从 redis 读取作业进行处理

注 2: -- 更改 gitlab 存储数据库

```
# 此参数设置为 false 指禁用内置的 postgresql, 而使用外部 postgresql 数据源
postgresql['enable'] = false
gitlab_rails['db_adapter'] = "postgresql"
gitlab_rails['db_encoding'] = "utf8"
# 数据库名
gitlab_rails['db_database'] = "gitlab"
gitlab_rails['db_pool'] = 100
gitlab_rails['db_username'] = "gitlab"
gitlab_rails['db_password'] = "password#123"
gitlab_rails['db_host'] = "162.14.131.88"
gitlab_rails['db_port'] = "5432"
```

	# 数据库用户
	# 数据库密码
	# 地址
	# 端口

注 3: -- 忘记 gitlab 管理员密码

```
# 修改密码, 针对 id=1

cd /opt/gitlab/bin
gitlab-rails console

User.all
u=User.where(id:1).first
u.password='jzzG1234'
u.password_confirmation='jzzG1234'
u.save!
```

使用 **gitlab:**

- 1、初始密码在安装界面会提示，保存于 `/etc/gitlab/initial_root_password` 24 小时有效。



GitLab 社区版

用户名或主要电子邮件

密码

[忘记密码 ?](#)

☐ 记住账号

登录

还没有账户? [立即注册](#)

- 2、偏好设置，可调整为中文。



用户设置 / 偏好设置

个性化

当您在描述或评论框中键入时，按列表中的 `Enter`，会在下面添加一个新项。

制表符宽度

必须为1到12之间的数字

本地化

自定义语言和区域相关设置。 [进一步了解](#)

语言

Chinese, Simplified - 简体中文 (92% 已翻译)

此功能是实验性的，翻译尚未完成。
[帮助将 GitLab 翻译成您的语言](#)

每周的起始日


星期一

- 3、创建群组。

群组名称

code

组名必须以字母、数字、表情或下划线开头，可以包含句点、破折号、空格和括号。

 如果您打算使用 SCIM 集成，您的组名称不得包含句点，因为这可能会导致错误。




群组 URL

http://192.168.198.132:8888/ code

群组路径可用。

可见性级别

哪些人可以看到这个群组？ [查看文档](#)

- ☐  私有
群组及其项目只能由成员查看。
- ☐  内部
除外部用户外，任何登录用户均可查看该群组和任何内部项目。
- ☒  公开
群组和任何公开项目可以在没有任何身份验证的情况下查看。

现在，个性化您的 GitLab 体验

我们将使用它来帮助向您展示正确的功能和信息。

角色

软件开发人员

谁将使用这个群组？

- ☒ 我的公司或团队 ☐ 仅我自己

你会用这个群组做什么？

我想存储我的代码

创建群组

取消

4、创建项目。



创建空白项目

创建一个空白项目来存放您的文件，规划您的工作，并在代码等方面进行协作。

项目名称

必须以小写或大写字母、数字、表情符号或下划线开头。也可以包含点、加号、破折号或空格。

项目 URL



项目标识串

可见性级别 [?](#)

☐ 私有

项目访问权限必须明确授予每个用户。如果此项目是一个群组的一部分，访问权限将授予该群组的成员。

☐ 内部

除外部用户外，任何登录用户均可访问该项目。

☒ 公开

无需任何身份验证即可访问该项目。

项目配置

☒ 使用自述文件初始化仓库

允许您立即克隆这个项目的仓库。如果您计划推送一个现有的仓库，请跳过这个步骤。

☐ 启用静态应用安全测试 (SAST)

分析源代码查找已知安全漏洞。 [了解更多](#)。

5、添加 ssh 密钥。用 `ssh-keygen` 生成一组密钥，将其中公钥 `id_rsa.pub` 添加到 `gitlab`

添加SSH密钥

添加 SSH 密钥以安全访问 GitLab。 [了解更多](#)。

密钥

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQgQDjhd+7dBijfcY0F680n0uQisH8DBFlkb+SJQwhGPPFRd+lg20rm4oF8lY/pVw3XzT9xsJL0v1AXwEp+he5ivbpj+
DBPSqjSwzyZBKzD1xJ15APnD9Jxm4j28fcWnxYiv2Xxh9BCG5rqiwcSpstNwV+GWwfDZLLYfS8rvUT+5MikGUdxBBaRK2Fqvr1VavoEmLTdzdnhAfJp/GA6
VYnmYQEbfsSkbOC0YXVl0vB24vUJ4KQylIDVT3xnKnaP8LTLIDpt9bUyCOQJTKzmct0GJnYNPM0UjHulbXqB9PTS/j27OuHM0rbglSLOFQuJ89hQsPeg3
8qQxY+ScCOlCgN2pozD9l/jZ2AW7Ap0ID5MLegYR58qFX1+766L8Vsbw6/k+6B7li39GmTPPvMpmni0Mfky1bhxOoCn7wqCMukT WXY7K3/yRhSQuPYO
t5KqciX4spTbqpRRU3q9nFuQiqb0vv0DDbM8bhV9FXt6hvf1WoM+dmT/DVOAQCuW8OKAGzS0= root@localhost.localdomain
```

以 'ssh-rsa'、'ssh-dss'、'ecdsa-sha2-nistp256'、'ecdsa-sha2-nistp384'、'ecdsa-sha2-nistp521'、'ssh-ed25519'、'sk-ecdsa-sha2-nistp256@openssh.com'，或 'sk-ssh-ed25519@openssh.com' 开头。

标题

密钥标题是公开可见的。

使用类型

到期时间



可选，但推荐。如果设置，密钥在指定日期无效。

6、上传第一个代码。

```
# 示例一：创建本地代码，然后提交 gitlab

mkdir dev
cd dev

git init
git config --global init.defaultBranch main
git config --global user.name "ping.bao"
git config --global user.email 360565687@qq.com

git remote rm origin
git remote add origin http://192.168.198.132:8888/code/dev.git
git pull origin main

# 编写 main.go 文件

git add main.go
git commit main.go -m "add hello"

git branch -M main
git push -uf origin main

# 示例二：拉取远程代码，修改后提交 gitlab

mkdir dev
cd dev
git clone git@192.168.198.132:code/dev.git

# 编写 main.go 文件

git add main.go
git commit main.go -m "add hello"
git push
```

此时可以通过 web 界面 查看到提交的代码信息 <http://192.168.198.132:8888/>


Jenkins 项目配置 (Gitlab、Golang) :

新建任务


输入一个任务名称

gittest

Select an item type

- 

构建一个自由风格的软件项目

这是Jenkins的主要功能.Jenkins将会结合任何SCM和任何构建系统来构建你的项目, 甚至可以构建软件以外的系统.
- 

文件夹

创建一个可以嵌套存储的容器。利用它可以进行分组。视图仅仅是一个过滤器, 而文件夹则是一个独立的命名空间, 因此你可以有多个相同名称的的内容, 只要它们在不同的文件 夹里即可。

如果你想根据一个已经存在的任务创建, 可以使用这个选项

复制

输入自动完成

确定

Git ?

Repositories ?

Repository URL ?

git@192.168.198.132:code/dev.git

Credentials ?

git

+ 添加

高级

Add Repository

Branches to build ?

指定分支 (为空时代表any) ?

*/main

添加 `ssh-keygen` 生成的 `id_rsa` 私钥凭据，用于自动登录 `gitlab` 服务器

Dashboard > 系统管理 > 凭据 > 系统 > 全局凭据 (unrestricted) > root (gitlab-root-login)

更新

删除

移动

Update credentials

范围 ?

全局 (Jenkins, nodes, items, all child items, etc)

ID ?

描述 ?

gitlab-root-login

Username

git

☐ Treat username as secret ?

Private Key

☒ Enter directly

Key

在下面输入新秘文

xs/LGMNaxxuFYQVes

qG89yVazz52a38NU4BAK5bt4oAbNLQAAAAAABAAAGAG7L6/ZkHi

fEPvP1Ba8c5a7Us1K

ET/+WIX1+IPeRf/U2NYAVOipn1gstwb28b1eE4U21oa2Gka55VUQ

配置编译脚本，用于拉取代码后执行。

```
# 使命以下脚本编译项目，项目位于 $HOME/.jenkins/workspace/

export GO_HOME=/usr/local/go
export PATH=$PATH:$GO_HOME/bin
cd $WORKSPACE
go get -u -v
go build -o hello_${SVN_REVISION} ./main.go
```

Build Steps

☰ 执行 shell ?

命令

[查看 可用的环境变量列表](#)

```
export GO_HOME=/usr/local/go
export PATH=$PATH:$GO_HOME/bin
cd $WORKSPACE
go get -u -v
go build -o hello_${SVN_REVISION} ./main.go
```

高级 ▾

保存后，立即构建测试。

☰ 状态

</> 修改记录

📁 工作空间

▶ 立即构建

⚙️ 配置

🗑️ 删除 工程

✎️ 重命名

✔️ gittest

相关链接

- 最近一次构建(#2),1 分 18 秒之前
- 最近稳定构建(#2),1 分 18 秒之前
- 最近成功的构建(#2),1 分 18 秒之前
- 最近失败的构建(#1),2 分 14 秒之前
- 最近未成功的构建(#1),2 分 14 秒之前
- 最近完成的构建(#2),1 分 18 秒之前

☁️ 构建历史

趋势 ▾

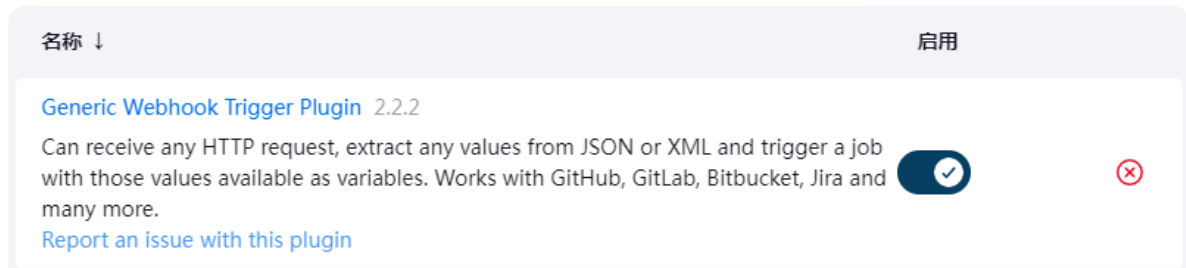
🔍 过滤构建... /

✔️ #2

| 2024年8月27日 下午2:20 CST

gitlab 通过 webhooks 请求 jenkins 构建项目（全局方案）：

1、配置 jenkins 添加 Generic Webhook Trigger Plugin 插件。



2、进入项目 gittest 配置，本例中 token 设置为 helloworld。

Generic Webhook Trigger ?

Is triggered by HTTP requests to **http://JENKINS_URL/generic-webhook-trigger/invoke**

There are example configurations in [the Git repository](#).

You can fiddle with JSONPath [here](#). You may also want to checkout the syntax [here](#).

You can fiddle with XPath [here](#). You may also want to checkout the syntax [here](#).

You can fiddle with regular expressions [here](#). You may also want to checkout the syntax [here](#).

If your job is **not parameterized**, then the resolved variables will just be contributed to the build. If your job is **parameterized**, and you resolve variables that have the same name as those parameters, then the plugin will populate the parameters when triggering job. That means you can, for example, use the parameters in combination with an SCM plugin, like GIT Plugin, to pick a branch.

Post content parameters

[新增](#)

If you want value of **param1** from post content { "param1": "value1" } to be contributed, you need to add **\$.param1** here.

Header parameters

[新增](#)

If you want value of header **param1** to be contributed, you need to add "param1" here.

Request parameters

[新增](#)

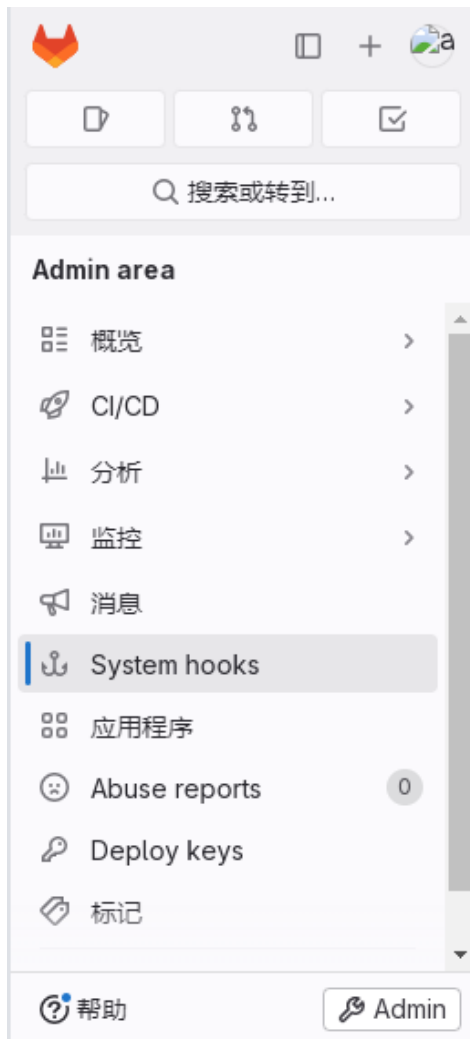
If you want value of query parameter **param1** to be contributed, you need to add "param1" here.

Token

Optional token. If it is specified then this job can only be triggered if that token is supplied when invoking **http://JENKINS_URL/generic-webhook-trigger/invoke**. It can be supplied as a:

- Query parameter **/invoke?token=TOKEN_HERE**
- A token header **token: TOKEN_HERE**
- A Authorization: Bearer header **Authorization: Bearer TOKEN_HERE**

3、进入 gitlab 的 Admin 配置，找到 System hooks。



4、添加新的 web hooks

URL 填写: <http://192.168.198.132:9999/generic-webhook-trigger/invoke>

Secret 令牌 填写: helloworld

不勾选验证 SSL 证书

System hooks

System hooks 使您能够向 Web 应用程序发送通知以响应群组或项目中的事件。

URL

http://192.168.198.132:9999/generic-webhook-trigger/invoke

如有必要，URL 必须经过百分比编码。

Name (optional)

Description (optional)

Secret令牌

.....

使用此令牌来验证收到的有效数据。

触发器

系统钩子是在一系列事件上触发的，例如创建项目或添加 SSH 密钥。您还可以启用额外的触发器，例如推送事件。

☒

仓库更新事件

更新仓库时触发 URL

☐

推送事件

为每个更新到仓库的分支触发 URL

☐

标签推送事件

将新标签推送到仓库时会触发 URL

☐

合并请求事件

创建、更新或合并合并请求时触发 URL

SSL 验证

☐

启用 SSL 验证

添加webhook

取消

未启用 webhook。选择上面的触发事件。

添加 webhook 后可以进行测试，将自动触发jenkins 操作。

#15 (2024年8月28日 下午5:25:46)

启动用户admin

Revision: 975cf3e4f8669dd52413e72929869abb4b3a4ba2

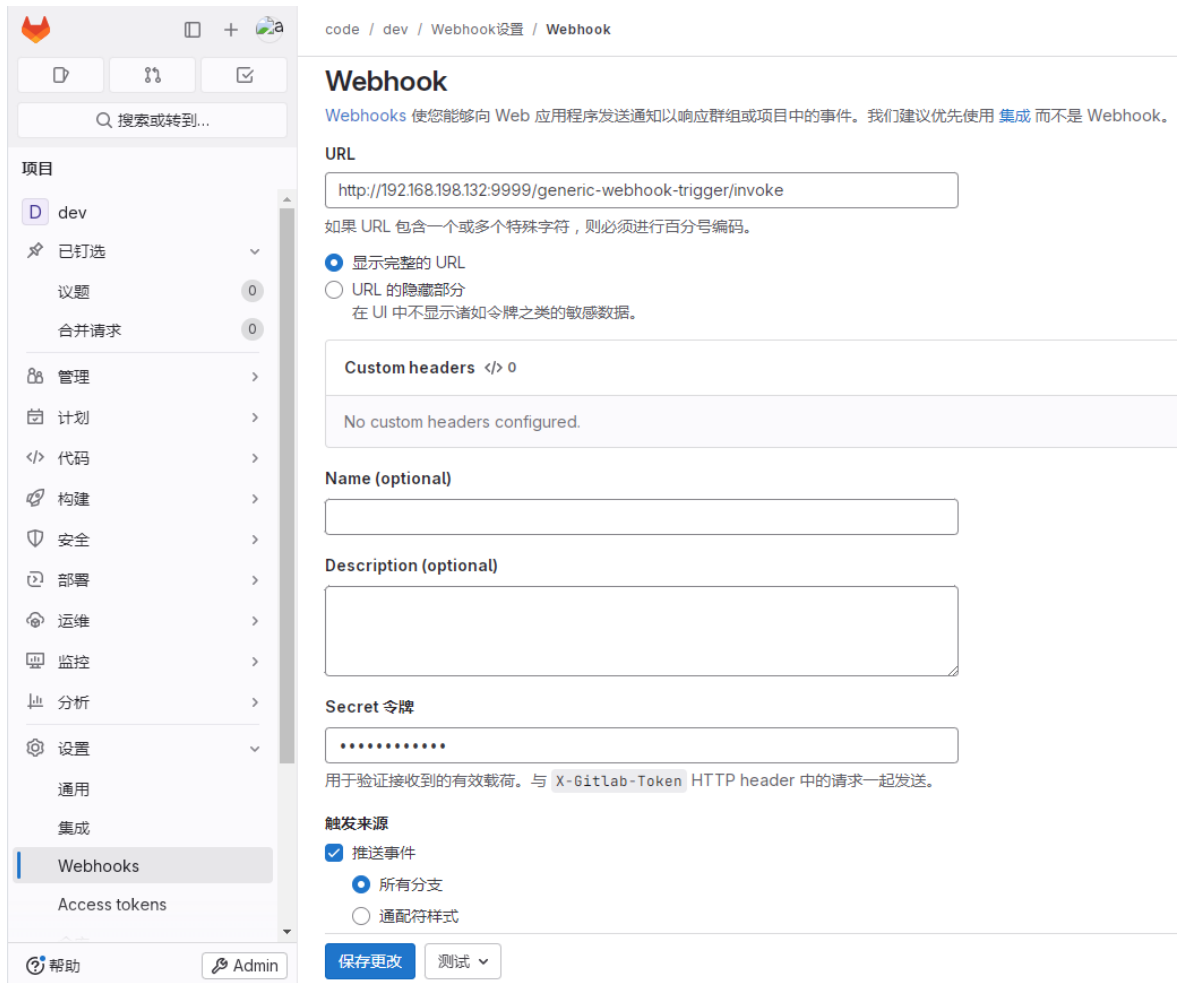
Repository: git@192.168.198.132:code/dev.git

gitlab 通过 webhooks 请求 jenkins 构建项目（局部方案1）：

1、进入 gitlab 的 Admin 配置，设置 网络，允许系统钩子向本地网络发送请求，添加白名单。



2、进入 项目 的设置，webhooks，添加 jenkins 地址。




URL 填写: <http://192.168.198.132:9999/generic-webhook-trigger/invoke>

Secret 令牌 填写: helloworld

不勾选验证 SSL 证书

3、测试触发, 进入 jenkins 查看编译结果。

 #19 (2024年8月29日 下午4:45:55)



Generic Cause



Revision: 76f2e09a73135b0276b650b8a9c1709e8a8d55ce

Repository: git@192.168.198.132:code/dev.git

- refs/remotes/origin/main



Changes

1. sasa ([details](#))

gitlab 通过 集成jenkins 构建项目（局部方案 2 - 测试未通过报错 403）：

1、进入 项目 的 设置，集成，找到 jenkins 配置。



2、添加 jenkins 地址 <http://192.168.198.132:9999> 项目名 gittest、帐号、密码。

启用集成

☒ 启用

触发器

☒ 推送
推送到仓库的触发事件。

☐ 合并请求
在创建、更新或合并合并请求时触发事件。

☐ 标签推送
推送到仓库的新标签的触发事件。

Jenkins 服务器 URL

Jenkins 服务器的 URL。

SSL 验证

☐ 启用 SSL 验证
如果使用自签名证书，请清除

Project name

Jenkins 项目的名称。将 URL 末尾的名称复制到项目中。

Username

Jenkins 服务器的用户名。

输入新密码。

留空时，使用您当前的密码。

3、本例测试未通过，报错 403，等待后续版本解决此问题。