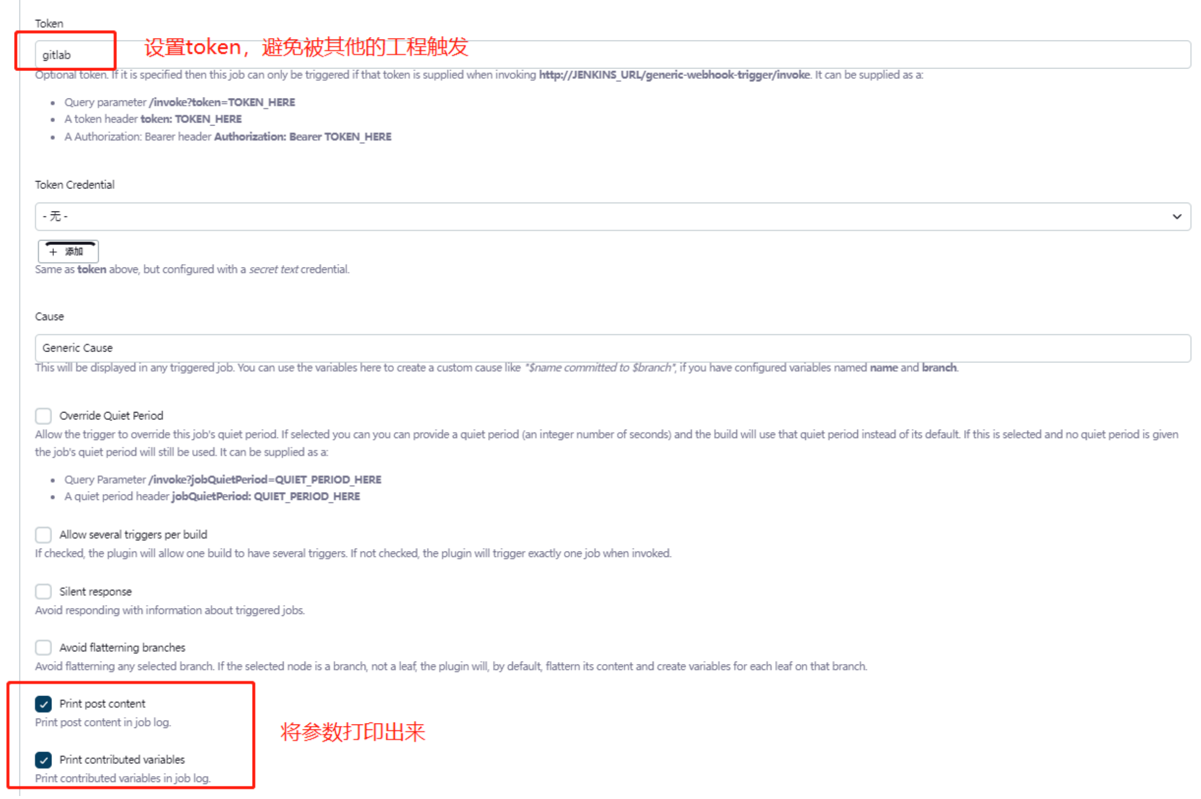
1.jenkins上 Generic Webhook Trigger 插件的配置



Token

gitlab

设置token，避免被其他的工程触发

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**

Token Credential

- 无 -

＋ 添加

Same as **token** above, but configured with a *secret text* credential.

Cause

Generic Cause

This will be displayed in any triggered job. You can use the variables here to create a custom cause like *"$name committed to $branch"*, if you have configured variables named **name** and **branch**.

☐ Override Quiet Period

Allow the trigger to override this job's quiet period. If selected you can you can provide a quiet period (an integer number of seconds) and the build will use that quiet period instead of its default. If this is selected and no quiet period is given the job's quiet period will still be used. It can be supplied as a:

- Query Parameter **/invoke?jobQuietPeriod=QUIET_PERIOD_HERE**
- A quiet period header **jobQuietPeriod: QUIET_PERIOD_HERE**

☐ Allow several triggers per build

If checked, the plugin will allow one build to have several triggers. If not checked, the plugin will trigger exactly one job when invoked.

☐ Silent response

Avoid responding with information about triggered jobs.

☐ Avoid flattening branches

Avoid flattening any selected branch. If the selected node is a branch, not a leaf, the plugin will, by default, flattern its content and create variables for each leaf on that branch.

☑ Print post content

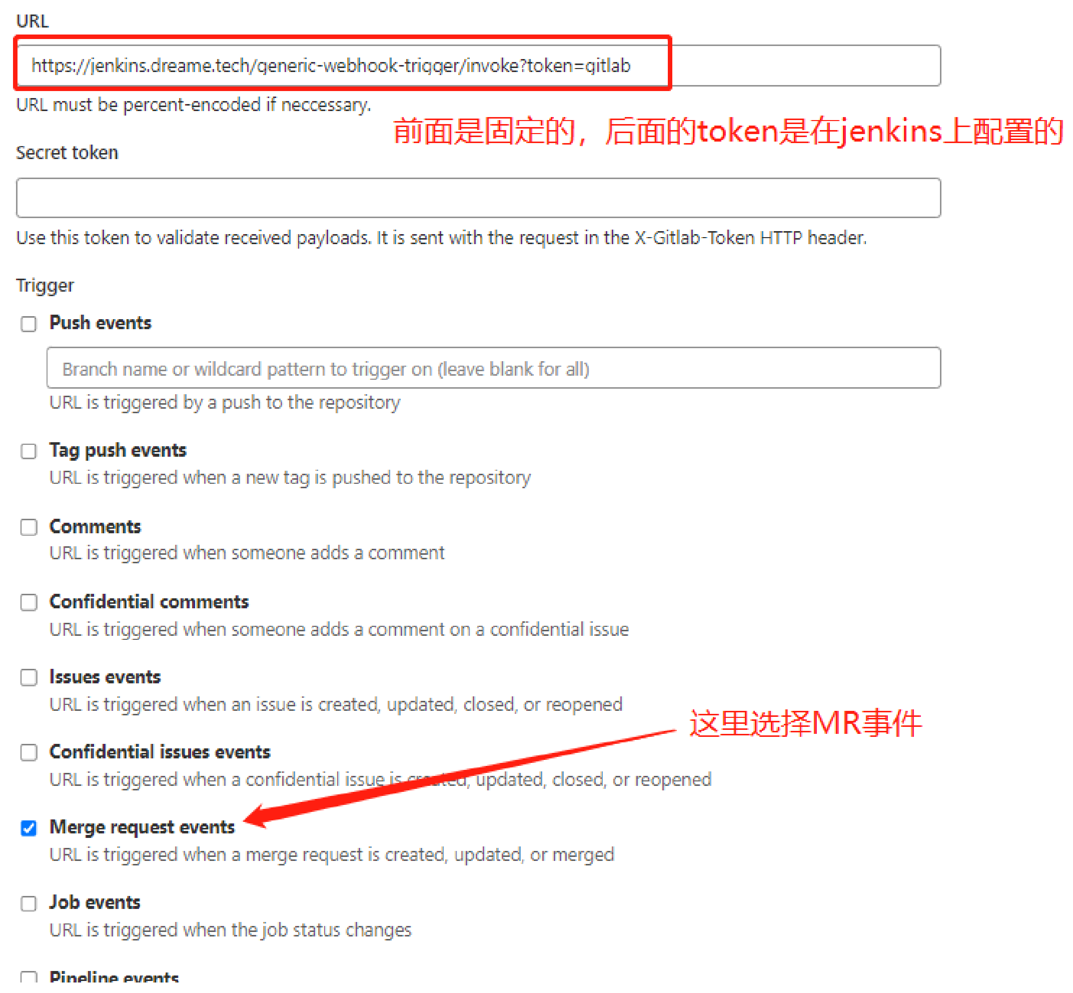
Print post content in job log.

☑ Print contributed variables

Print contributed variables in job log.

将参数打印出来

2. gitlab上的hook配置



URL

https://jenkins.dreame.tech/generic-webhook-trigger/invoke?token=gitlab

in

URL must be percent-encoded if neccessary.

前面是固定的，后面的token是在jenkins上配置的

Secret token

Use this token to validate received payloads. It is sent with the request in the X-Gitlab-Token HTTP header.

Trigger

☐ **Push events**

Branch name or wildcard pattern to trigger on (leave blank for all)

URL is triggered by a push to the repository

☐ **Tag push events**

URL is triggered when a new tag is pushed to the repository

☐ **Comments**

URL is triggered when someone adds a comment

☐ **Confidential comments**

URL is triggered when someone adds a comment on a confidential issue

☐ **Issues events**

URL is triggered when an issue is created, updated, closed, or reopened

这里选择MR事件

☐ **Confidential issues events**

URL is triggered when a confidential issue is created, updated, closed, or reopened

☑ **Merge request events**

URL is triggered when a merge request is created, updated, or merged

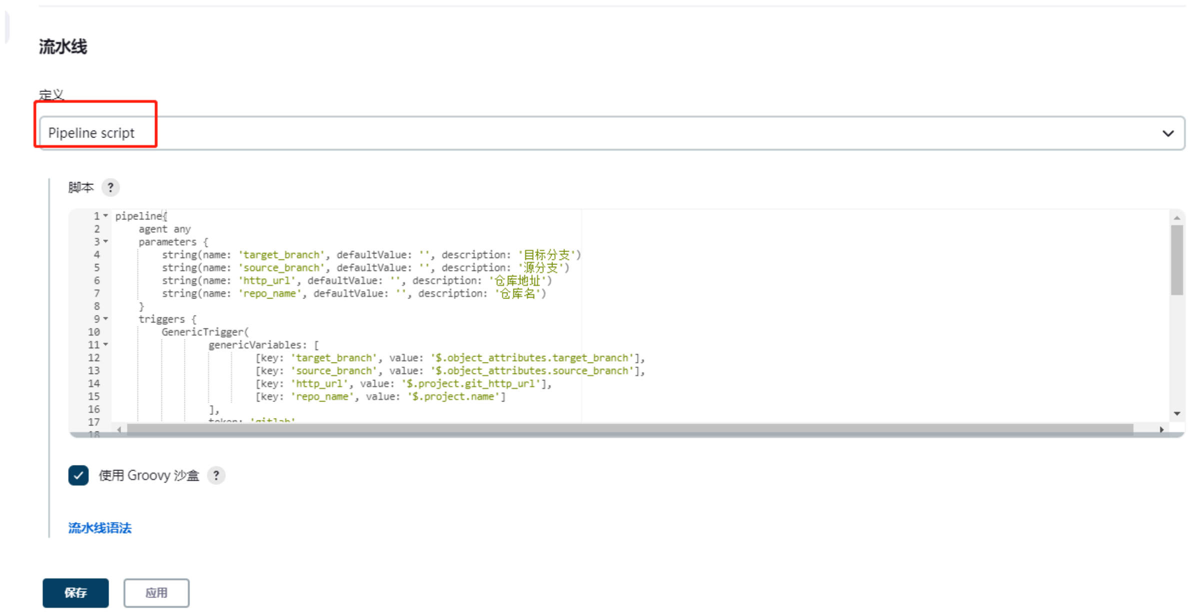
☐ **Job events**

URL is triggered when the job status changes

☐ **Pipeline events**

原理：gitlab上创建mr、合并mr、关闭mr等对mr进行处理后，都会触发mr的事件，会发送一条hook给jenkins（携带一些参数），jenkins对参数进行解析，然后触发pipeline脚本，脚本如下：

3.pipeline脚本



```
pipeline{
    agent any
    parameters {
        string(name: 'target_branch', defaultValue: '', description: '目标分支')
        string(name: 'source_branch', defaultValue: '', description: '源分支')
        string(name: 'http_url', defaultValue: '', description: '仓库地址')
        string(name: 'repo_name', defaultValue: '', description: '仓库名')
    }
    triggers {
        GenericTrigger(
                genericVariables: [
                        [key: 'target_branch', value: '$.object_attributes.target_branch'],
                        [key: 'source_branch', value: '$.object_attributes.source_branch'],
                        [key: 'http_url', value: '$.project.git_http_url'],
                        [key: 'repo_name', value: '$.project.name']
                ],
                token: 'gitlab'
```

如果使用仓库保存该配置脚本的话，就选择 “Pipeline script from SCM”，然后正常配置；这里选择Pipeline script，内容如下：

```
pipeline{
 agent any
 //参数化构建,后面可以手动触发构建
 parameters {
 string(name: 'target_branch', defaultValue: '', description: '目标分支')
 string(name: 'source_branch', defaultValue: '', description: '源分支')
 string(name: 'http_url', defaultValue: '', description: '仓库地址')
 string(name: 'repo_name', defaultValue: '', description: '仓库名')
 }
 triggers {
 GenericTrigger(
 //取参数
 genericVariables: [
 [key: 'target_branch', value: '$.object_attributes.target_branch'],
 [key: 'source_branch', value: '$.object_attributes.source_branch']
 [key: 'http_url', value: '$.project.git_http_url'],
 [key: 'repo_name', value: '$.project.name']
 ],
 token: 'gitlab',
 printContributedVariables: true,
 printPostContent: true,
 silentResponse: false
 )
 }

 stages{
 //stage名
 stage("test"){
 //指定节点,如果不指定的话,就会随机使用一个可用节点
 agent {label 'produce_3'}
 steps{
 //定义工作目录，按照自己需求改
 dir("/home/ujenkins/test"){
 script{
 //这里后续需要做正则匹配，groovy语法
 if(target_branch == "master"){
 //credentialsId是自己在jenkins上配置的凭证，要有权限去拉代码
 checkout scmGit(branches: [[name: '*/${source_branch}']], extensions: [[$class: 'PreBuildMerge', options: [mergeRemote: 'origin',mergeTarget: '${target_branch}']]], userRemoteConfigs: [[credentialsId: 'b9586bc9-4823-4838-827b-58e4e14ea2e3', url: '${http_url}']])
 sh "cd ${repo_name} && mvn clean package"
 sh "mkdir -p /home/ftp/pub/${target_branch}"
 sh "cp /var/lib/jenkins/workspace/ta-api/target/ta-api.jar /home/ftp/pub/${target_branch}"
 sh "chown -R devops:root /home/ftp/pub/origin"
 }
 }
 }
 }

 }
 }
 ,
```

4.保存配置后，手动触发下构建，触发后job会失败，这是因为最开始的时候我们并未配置job内容，手动触发构建后，jenkins会将pipeline脚本的内容写入job中，下次就可以正常触发了