

鲲鹏原生开发

# 用户指南

文档版本 01  
发布日期 2024-04-29



版权所有 © 华为技术有限公司 2024。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## 商标声明



HUAWEI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。

## 华为技术有限公司

地址：深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址：<https://www.huawei.com>

客户服务邮箱：[support@huawei.com](mailto:support@huawei.com)

客户服务电话：4008302118

# 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 环境要求..... 1
- 2 部署 Jenkins..... 4
  - 2.1 （可选）部署 Jenkins..... 4
  - 2.2 将各个执行机添加至 Jenkins 集群..... 11
- 3 部署 Gitlab..... 18
  - 3.1 （可选）配置 Gitlab..... 18
  - 3.2 部署 Gitlab Runner..... 22

# 1 环境要求

## 软件架构

支持鲲鹏架构下的openEuler类（例如openEuler 22.03 LTS）操作系统。

## 硬件要求

硬件环境如[表 硬件环境](#)所示。

表 1-1 硬件环境

项目	版本
服务器	TaiShan 200服务器（型号2280），仅限双路服务器
主板	鲲鹏主板
BMC	1711单板（型号BC82SMMAB）
CPU	鲲鹏920处理器（型号7260、5250、5220）
机箱	不限，建议8盘或12盘

## 软件要求

软件包要求如[表1-2](#)所示。

表 1-2 软件包

软件类型	版本	软件包名称	软件包说明	获取方式
DevKit Pipeline	v1.0	devkit-pipeline-v1.0.tar.gz	DevKit Pipeline软件包。	<a href="https://gitee.com/openeuler/devkit-pipeline/releases/download/v1.0/devkit-pipeline-v1.0.tar.gz">https://gitee.com/openeuler/devkit-pipeline/releases/download/v1.0/devkit-pipeline-v1.0.tar.gz</a>
download_tool-for-windows	v1.0	download_tool-for-windows.exe	Windows版本的一键下载工具。	<a href="https://gitee.com/openeuler/devkit-pipeline/releases/download/v1.0/download_tool-for-windows.exe">https://gitee.com/openeuler/devkit-pipeline/releases/download/v1.0/download_tool-for-windows.exe</a>
BiSheng Compiler	3.2.0	BiShengCompiler-3.2.0-aarch64-linux.tar.gz	毕昇编译器安装包。	<a href="https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/BiShengCompiler-3.2.0-aarch64-linux.tar.gz">https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_compiler/BiShengCompiler-3.2.0-aarch64-linux.tar.gz</a>
GCC for openEuler	10.3.1	gcc-10.3.1-2023.12-aarch64-linux.tar.gz	GCC for openEuler安装包。	<a href="https://mirrors.huaweicloud.com/kunpeng/archive/compiler/kunpeng_gcc/gcc-10.3.1-2023.12-aarch64-linux.tar.gz">https://mirrors.huaweicloud.com/kunpeng/archive/compiler/kunpeng_gcc/gcc-10.3.1-2023.12-aarch64-linux.tar.gz</a>
BiSheng JDK8	8u402	bisheng-jdk-8u402-linux-aarch64.tar.gz	BiSheng JDK8安装包。	<a href="https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_jdk/bisheng-jdk-8u402-linux-aarch64.tar.gz">https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_jdk/bisheng-jdk-8u402-linux-aarch64.tar.gz</a>
BiSheng JDK17	17.0.10	bisheng-jdk-17.0.10-linux-aarch64.tar.gz	BiSheng JDK17安装包。	<a href="https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_jdk/bisheng-jdk-17.0.10-linux-aarch64.tar.gz">https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_jdk/bisheng-jdk-17.0.10-linux-aarch64.tar.gz</a>

软件类型	版本	软件包名称	软件包说明	获取方式
Lkp Tests	v0.2	lkp-tests.tar.gz	Lkp Tests安装包。	<a href="https://gitee.com/openeuler/devkit-pipeline/releases/download/v0.2/lkp-tests.tar.gz">https://gitee.com/openeuler/devkit-pipeline/releases/download/v0.2/lkp-tests.tar.gz</a>
A-FOT	v0.2	a-fot.tar.gz	A-FOT安装包。	<a href="https://gitee.com/openeuler/devkit-pipeline/releases/download/v0.2/a-fot.tar.gz">https://gitee.com/openeuler/devkit-pipeline/releases/download/v0.2/a-fot.tar.gz</a>
DevkitDistribute	v0.2	devkit_distribute.tar.gz	DevkitDistribute安装包。	<a href="https://gitee.com/openeuler/devkit-pipeline/releases/download/v0.2/devkit_distribute.tar.gz">https://gitee.com/openeuler/devkit-pipeline/releases/download/v0.2/devkit_distribute.tar.gz</a>
DevKit Web	24.0.RC1	DevKit-All-24.0.RC1-Linux-Kunpeng.tar.gz	DevKit工具安装包（Web）。	<a href="https://mirrors.huaweicloud.com/kunpeng/archive/DevKit/Packages/Kunpeng_DevKit/DevKit-All-24.0.RC1-Linux-Kunpeng.tar.gz">https://mirrors.huaweicloud.com/kunpeng/archive/DevKit/Packages/Kunpeng_DevKit/DevKit-All-24.0.RC1-Linux-Kunpeng.tar.gz</a>
DevKit CLI	24.0.RC1	DevKit-CLI-24.0.RC1-Linux-Kunpeng.tar.gz	DevKit命令行安装包。	<a href="https://mirrors.huaweicloud.com/kunpeng/archive/DevKit/Packages/Kunpeng_DevKit/DevKit-CLI-24.0.RC1-Linux-Kunpeng.tar.gz">https://mirrors.huaweicloud.com/kunpeng/archive/DevKit/Packages/Kunpeng_DevKit/DevKit-CLI-24.0.RC1-Linux-Kunpeng.tar.gz</a>

# 2 部署 Jenkins

## 2.1 （可选）部署 Jenkins

### 2.2 将各个执行机添加至 Jenkins 集群

## 2.1 （可选）部署 Jenkins

### 获取安装包

若需要获取目标系统的 Jenkins 安装包，请参见：<https://archives.jenkins.io/>，推荐获取“LTS Releases”中的安装包。

- Jenkins RPM 安装包下载链接：<https://archives.jenkins.io/redhat-stable/jenkins-2.426.2-1.1.noarch.rpm>。
- Jenkins WAR 安装包下载链接：<https://archives.jenkins.io/war-stable/2.426.2/jenkins.war>。

#### 说明

1. Jenkins 离线安装详细操作步骤请参见：<https://www.jenkins.io/doc/book/installing/offline/>。
2. Jenkins 运行的 JDK 为 JDK11 及以上，官方推荐 JDK17。
3. 链接中的“2.426.2”指安装包版本号，可在链接中获取最新安装包下载。

### 配置 Jenkins yum 源

请执行以下命令配置 Jenkins yum 源。

**步骤1** 下载 Jenkins yum 镜像文件到“/etc/yum.repos.d/”目录下。

```
wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
```

**步骤2** 导入 Jenkins RPM 安装包校验证书。

```
rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
```

**步骤3 可选：**升级系统软件包，若需要执行以下命令，建议执行完成后重启系统。

```
yum upgrade -y
```

----结束



## 配置 JDK

请执行以下命令配置JDK。

### 步骤1 创建JDK安装目录。

```
mkdir -p /usr/local/lib64/jvm/bisheng
```

### 步骤2 下载并解压JDK到对应的安装目录。

```
wget -c https://mirrors.huaweicloud.com/kunpeng/archive/compiler/bisheng_jdk/bisheng-jdk-17.0.10-linux-aarch64.tar.gz -O - | tar -C /usr/local/lib64/jvm/bisheng/ -xzf - --no-same-owner
```

### 步骤3 创建自动配置脚本。

```
cat > "${HOME}"/SetJavaAlternatives.sh << 'EOF'
#!/bin/bash
jvm_path=/usr/lib/jvm
mkdir -p "${jvm_path}"
jdk_home_path=/usr/local/lib64/jvm/bisheng/bisheng-jdk-17.0.10
PRIORITY_ID=901700010
update-alternatives --install "${jvm_path}"/java-17 java_sdk_17 "${jdk_home_path}" "${PRIORITY_ID}"
update-alternatives --install "${jvm_path}"/java-17-openjdk java_sdk_17_openjdk "${jdk_home_path}" "${PRIORITY_ID}"
update-alternatives --install "${jvm_path}"/java-openjdk java_sdk_openjdk "${jdk_home_path}" "${PRIORITY_ID}"
for BinFilePath in "${jdk_home_path}"/bin/*; do
    if [ -x "${BinFilePath}" ]; then
        BinFileName="$(basename "${BinFilePath}")"
        update-alternatives --install /usr/bin/"${BinFileName}" "${BinFileName}" "${BinFilePath}" "${PRIORITY_ID}"
    fi
done
for ManFilePath in "${jdk_home_path}"/man/man1/*; do
    if [ -f "${ManFilePath}" ]; then
        ManFileName="$(basename "${ManFilePath}")"
        update-alternatives --install /usr/share/man/man1/"${ManFileName}" "${ManFileName}" "${ManFilePath}" "${PRIORITY_ID}"
    fi
done
EOF
```

### 步骤4 执行自动配置脚本。

```
/bin/bash "${HOME}"/SetJavaAlternatives.sh
```

### 步骤5 移除自动配置脚本。

```
rm -rf "${HOME}"/SetJavaAlternatives.sh
```

### 步骤6 配置JDK环境变量。

#### 1. 一键设置Java环境变量。

```
cat > /etc/profile.d/JavaEnvironmentVariable.sh << 'EOF'
# SET JDK Enviroment
JAVA_HOME=/usr/lib/jvm/java-17
PATH=$PATH:$JAVA_HOME/bin
export JAVA_HOME PATH
EOF
```

#### 2. 加载系统环境变量。

```
source /etc/profile
```

----结束

## 部署 Jenkins

请执行以下命令部署Jenkins。

### 步骤1 安装Jenkins PRM安装包。

```
yum install jenkins -y
```

**步骤2** 重新加载systemd管理器配置。

```
systemctl daemon-reload
```

**步骤3** 设置开机启动服务并立即启动Jenkins.service。

```
systemctl --now enable jenkins.service
```

**步骤4** 可选: 防火墙相关设置。

1. 请根据实际情况配置防火墙, 以允许访问8080端口。

```
firewall-cmd --permanent --zone=public --add-port=8080/tcp  
firewall-cmd --permanent --zone=public --add-service=http
```

2. 使用RPM安装方式默认会安装以下防火墙规则。

```
firewall-cmd --permanent --zone=public --add-service=jenkins
```

3. 重新加载防火墙并保留状态信息。

```
firewall-cmd --reload
```

4. 查看防火墙规则设置是否生效。

```
firewall-cmd --permanent --zone=public --list-all
```

----结束

## Jenkins 初始化设置

在浏览器地址栏中输入**http://服务器IP:8080** (例如http://x.x.x.x:8080), 按“Enter”访问Jenkins服务, 并根据提示进行Jenkins的初始配置。

**步骤1** 请根据提示获取初始密码并登录。

1. 请执行以下命令获取初始密码。

```
cat /var/lib/jenkins/secrets/initialAdminPassword
```

2. 填写初始密码, 单击“继续”按钮。

图 2-1 Jenkins 初始化设置 1

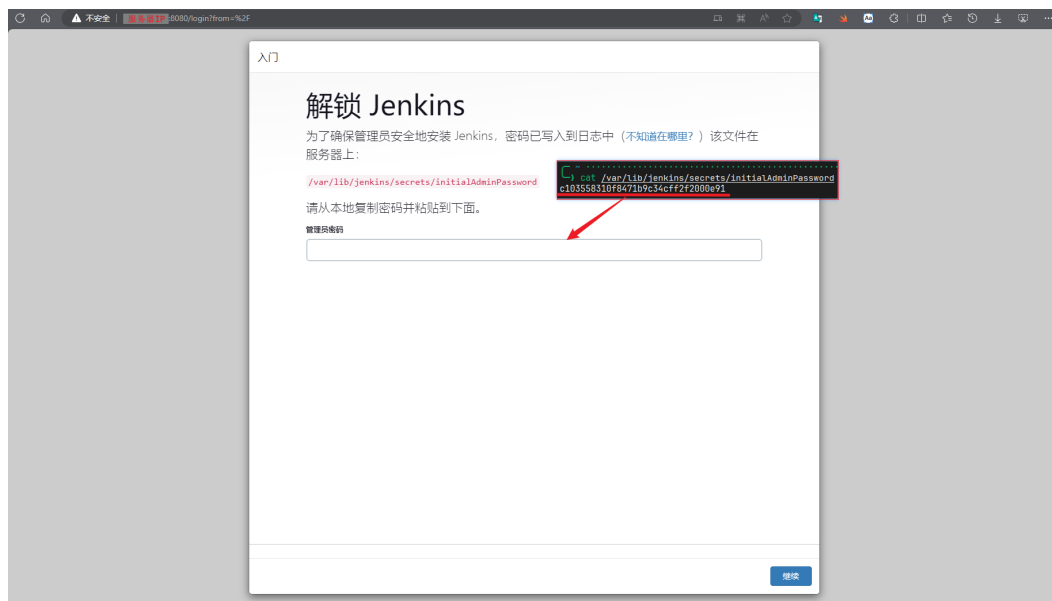
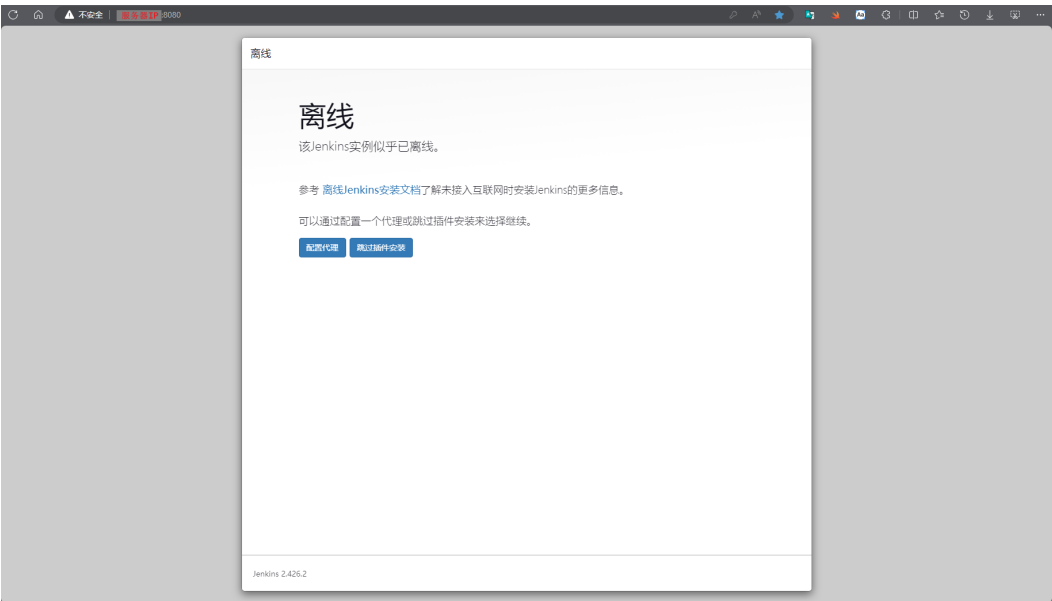
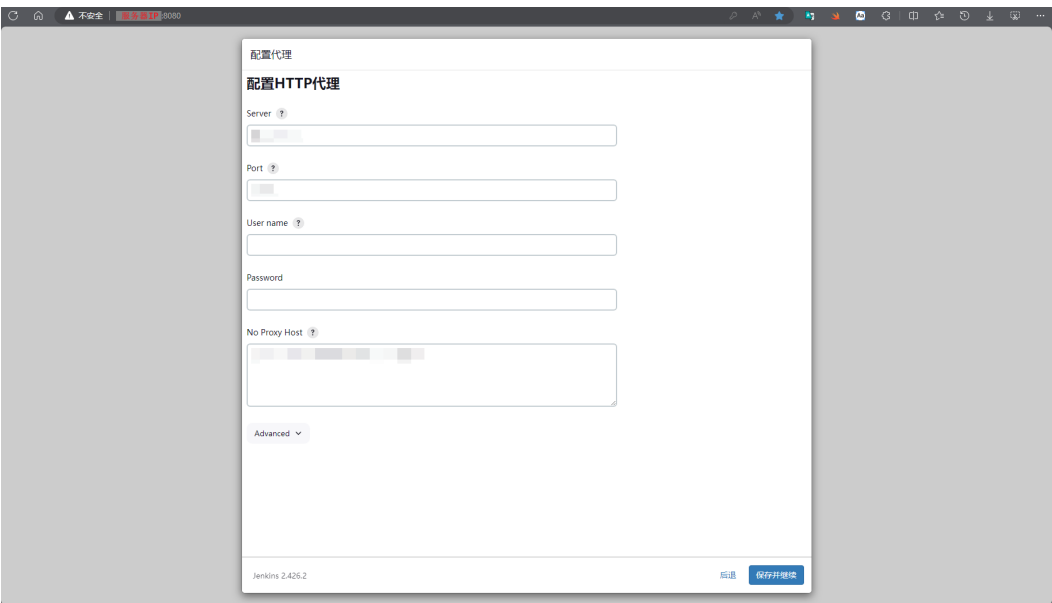
**步骤2** 可选: 配置代理, 请根据网络环境进行设置。如需离线使用可单击“跳过插件安装”跳过此步骤。

图 2-2 Jenkins 初始化设置 2



单击“配置代理”按钮进入配置代理界面，如图2-3所示，配置完成后单击“保存并继续”按钮。

图 2-3 Jenkins 初始化配置 3



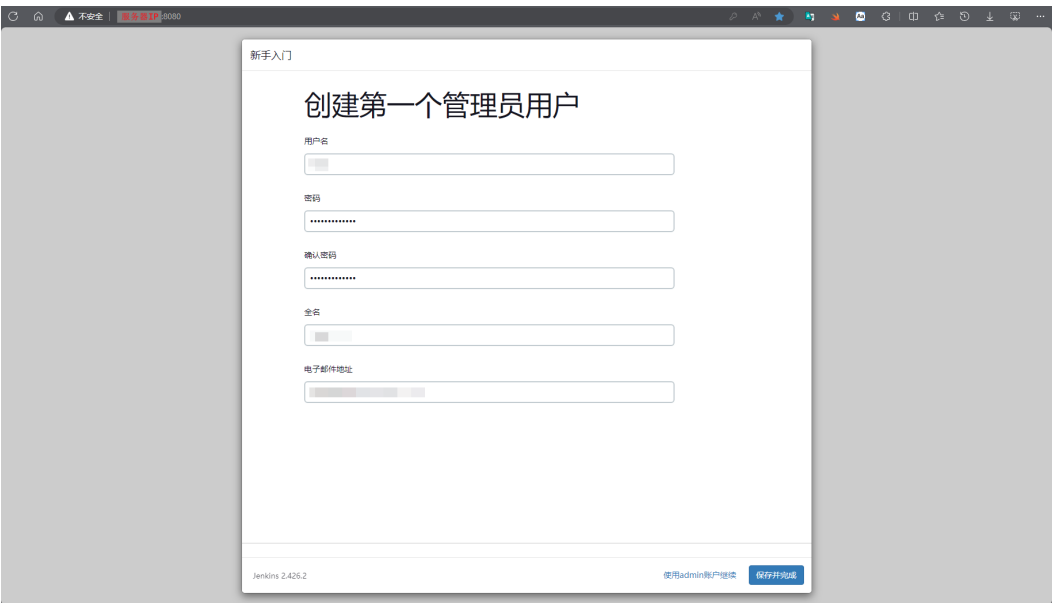
**步骤3** 自定义Jenkins，可选择“安装推荐的插件”。若当前网络不可用，可选择“选择插件来安装”，根据提示跳过安装。

图 2-4 Jenkins 初始化设置 4



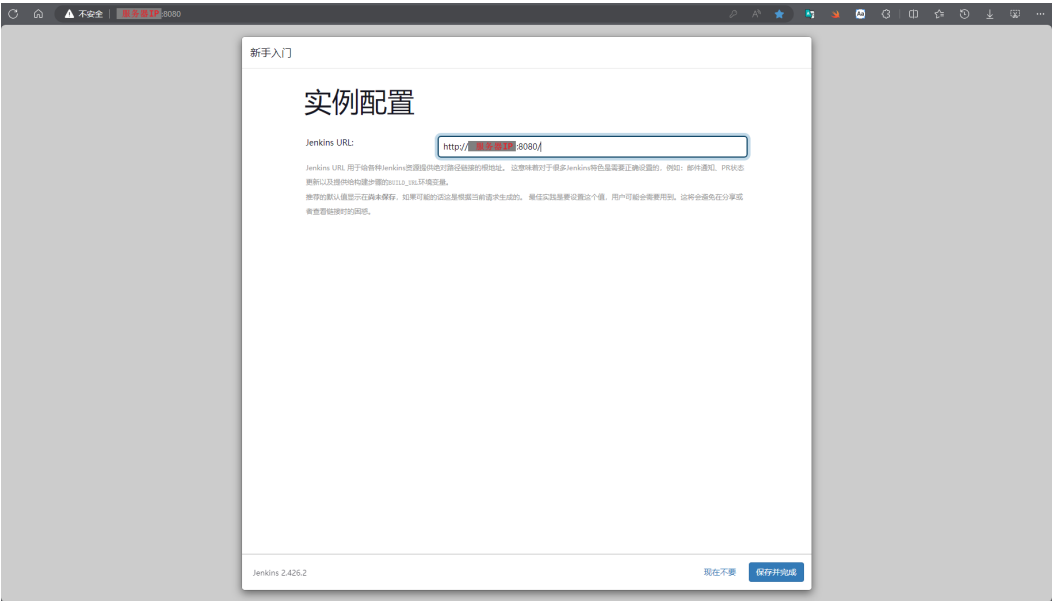
**步骤4** 配置管理员用户登录信息，请根据实际需求填写信息，填写完成后单击“保存并完成”。

图 2-5 Jenkins 初始化设置 5



**步骤5** Jenkins实例配置，请根据实际需求填写信息，此处使用默认设置，填写完成后单击“保存并完成”。

图 2-6 Jenkins 初始化配置 6



**步骤6** Jenkins安装已完成，请单击“开始使用Jenkins”按钮。

图 2-7 Jenkins 初始化设置 7



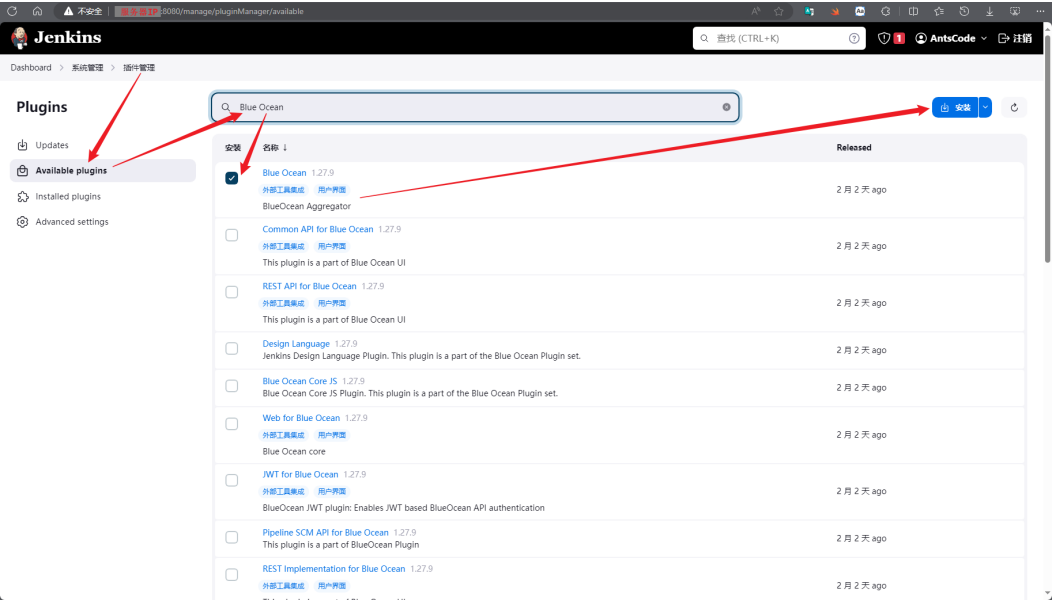
----结束

Jenkins 基础插件安装

可在线或离线安装基础插件，如需要离线安装插件，请访问插件主页下载符合目标要求的插件安装包（.hpi）。

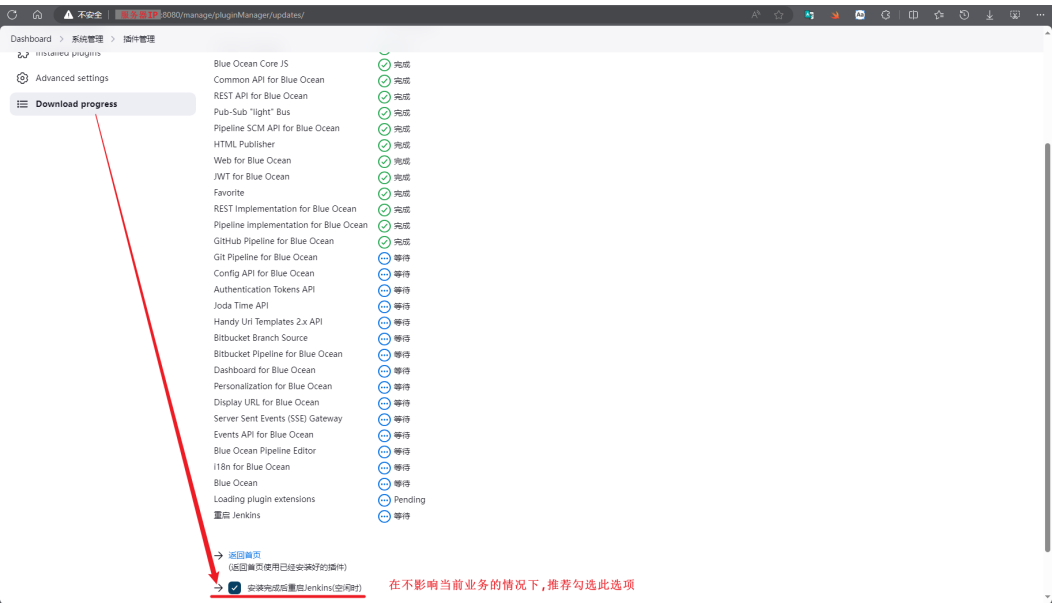
- **可选:** 安装Blue Ocean插件，Blue Ocean插件能以更直观的方式查看PiPipeline状态。

图 2-8 安装 Blue Ocean 插件 1



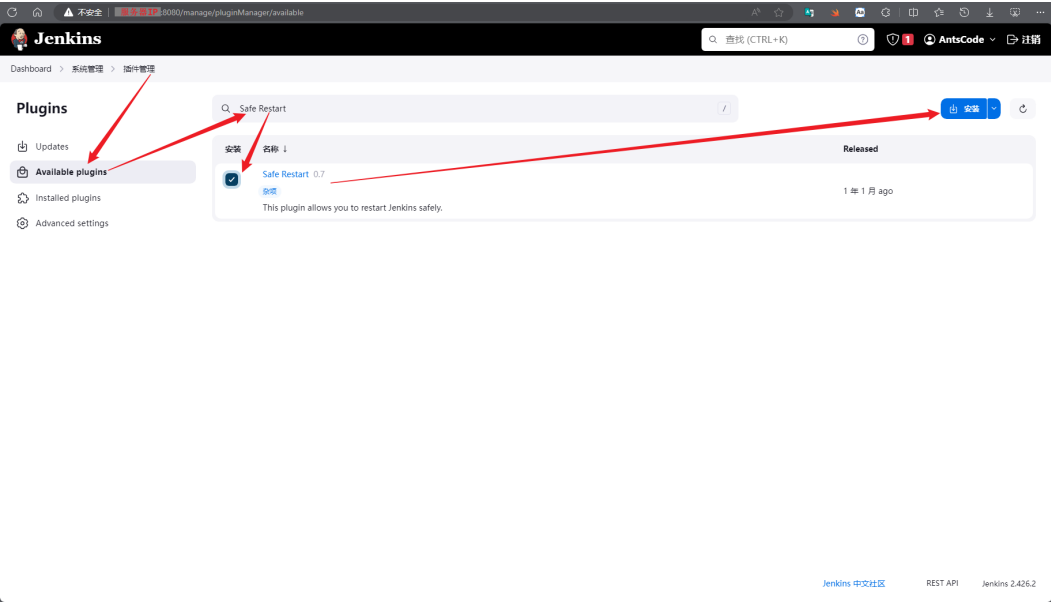
在不影响当前业务的情况下，推荐勾选“安装完成后重启Jenkins（空闲时）”。

图 2-9 安装 Blue Ocean 插件 2



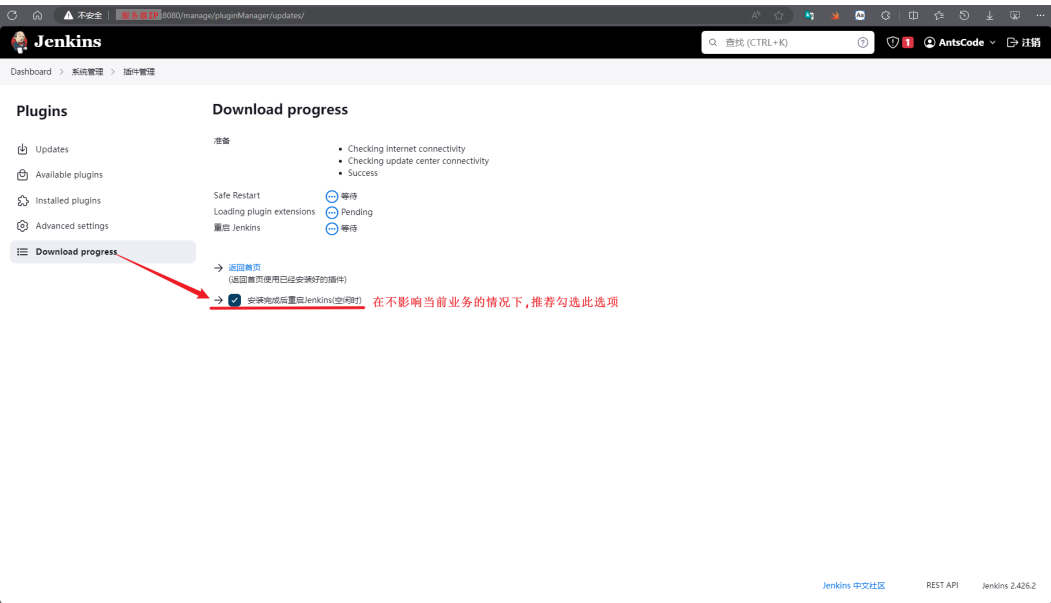
- 可选: 安装Safe Restart插件，Safe Restart插件可安全重启Jenkins。

图 2-10 安装 Safe Restart 插件 1



在不影响当前业务的情况下，推荐勾选“安装完成后重启Jenkins（空闲时）”。

图 2-11 安装 Safe Restart 插件 2

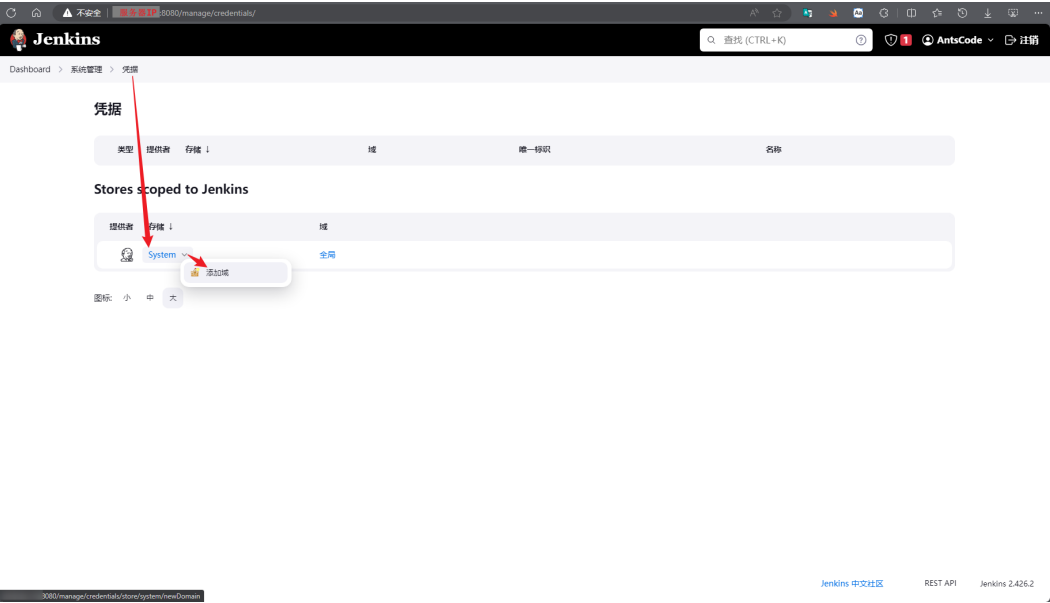


## 2.2 将各个执行机添加至 Jenkins 集群

### 凭证设置

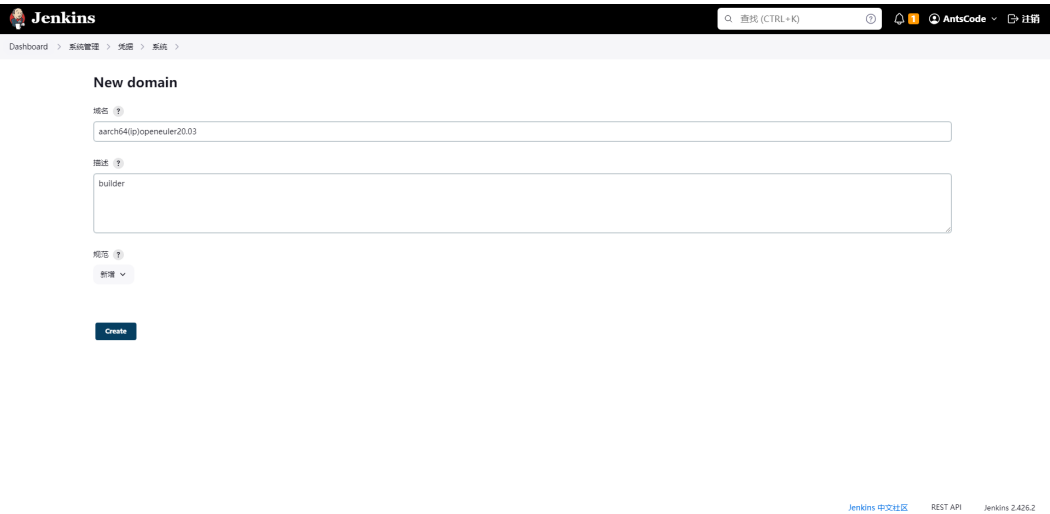
**步骤1** 添加凭据域。单击左侧树“系统管理> 凭据管理”，打开“凭据”页面，在“System”右侧下拉列表中单击“添加域”。

图 2-12 添加鲲鹏 DevKit Jenkins CI 插件凭据域



**步骤2** 请根据实际需求填写域名和描述信息，域名和描述信息便于识别和管理。单击“Create”进行添加。

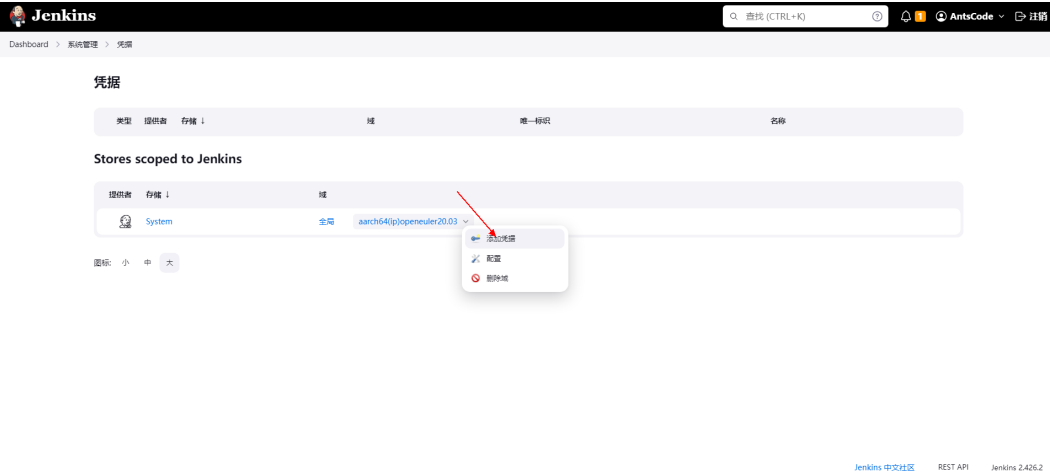
图 2-13 添加凭据域



**步骤3** 添加凭据到凭据域下。在凭据域右侧下拉列表中单击“添加凭据”，将凭据添加到当前凭据域下。



图 2-14 添加凭据



在DevKit凭据域下添加AArch64 Jenkins工作节点SSH凭据。

1. 在安装了Jenkins的环境上生成工作节点SSH免密登录证书，请根据实际需求设置SSH Key Passphrases。  
ssh-keygen -b 4096 -C "<邮件地址或其他标签>" -f ~/.ssh/id\_ed25519\_<推荐按照 \* \_ \_ \* \_ 格式填写目标服务器IP，便于管理KEY> -t ed25519
2. 在安装了Jenkins的环境上将生成的证书的公钥上传至目标服务器，请根据提示输入目标服务器对应的用户名和密码。  
ssh-copy-id -i ~/.ssh/id\_ed25519\_<推荐按照 \* \_ \_ \* \_ 格式填写目标服务器IP,便于管理KEY>.pub root@<目标服务器IP>
3. 删除已知主机名文件中属于指定主机名的所有密钥。  
ssh-keygen -R <目标服务器IP>
4. 使用SSH Key测试连接目标主机，若已设置SSH Key Passphrases，请在连接目标主机时根据提示输入证书密码。  
ssh -o IdentitiesOnly=yes -o PasswordAuthentication=no -i ~/.ssh/id\_ed25519\_<推荐按照 \* \_ \_ \* \_ 格式填写目标服务器IP,便于管理KEY> -l root -p 22 <目标服务器IP>
5. 查看SSH Key密钥。  
cat ~/.ssh/id\_ed25519\_<推荐按照 \* \_ \_ \* \_ 格式填写目标服务器IP,便于管理KEY>

图 2-15 工作节点 SSH 凭据 1

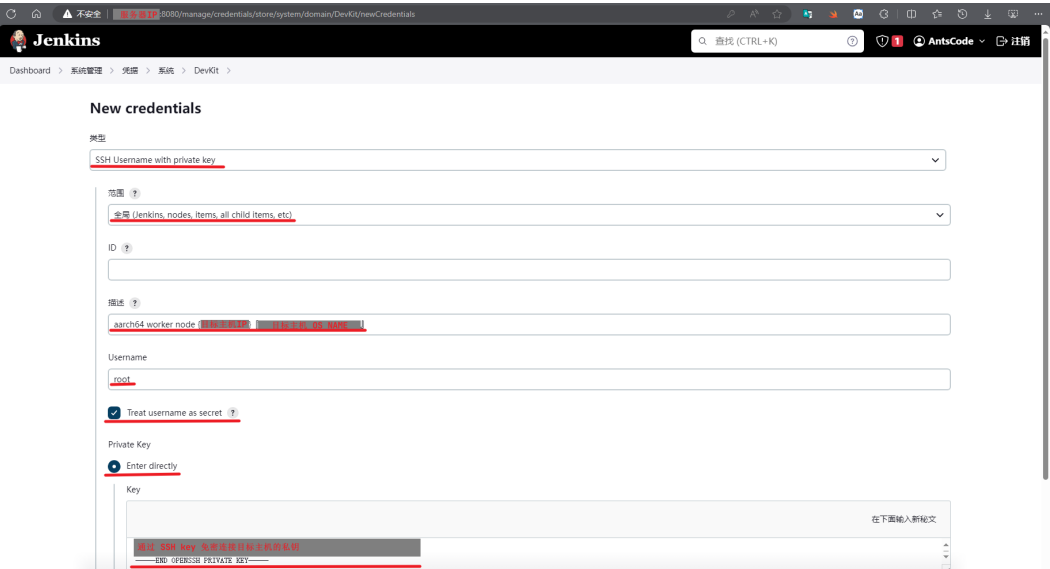


图 2-16 工作节点 SSH 凭据 2

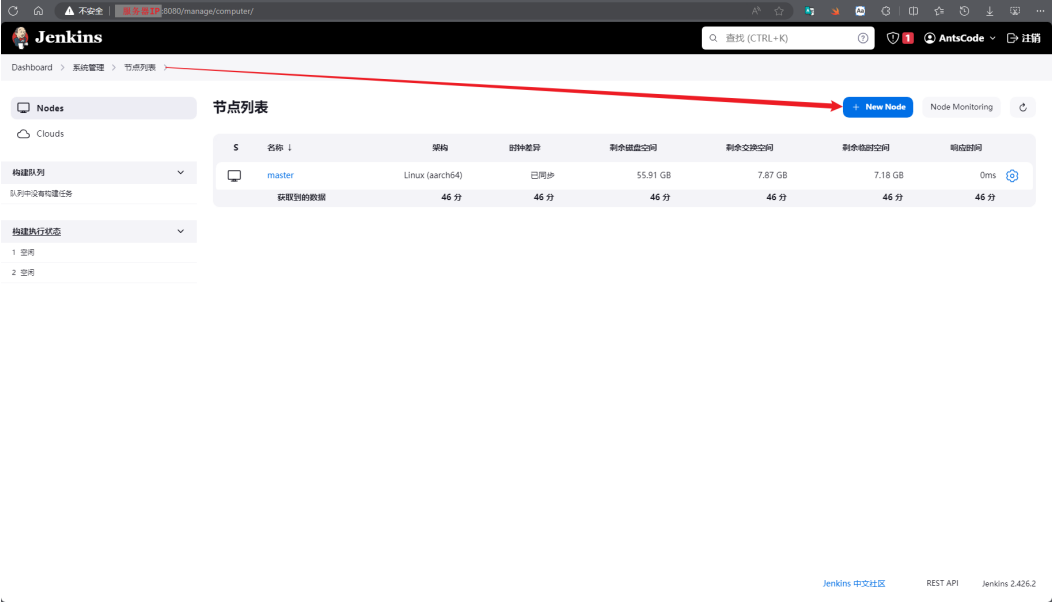


----结束

工作节点设置

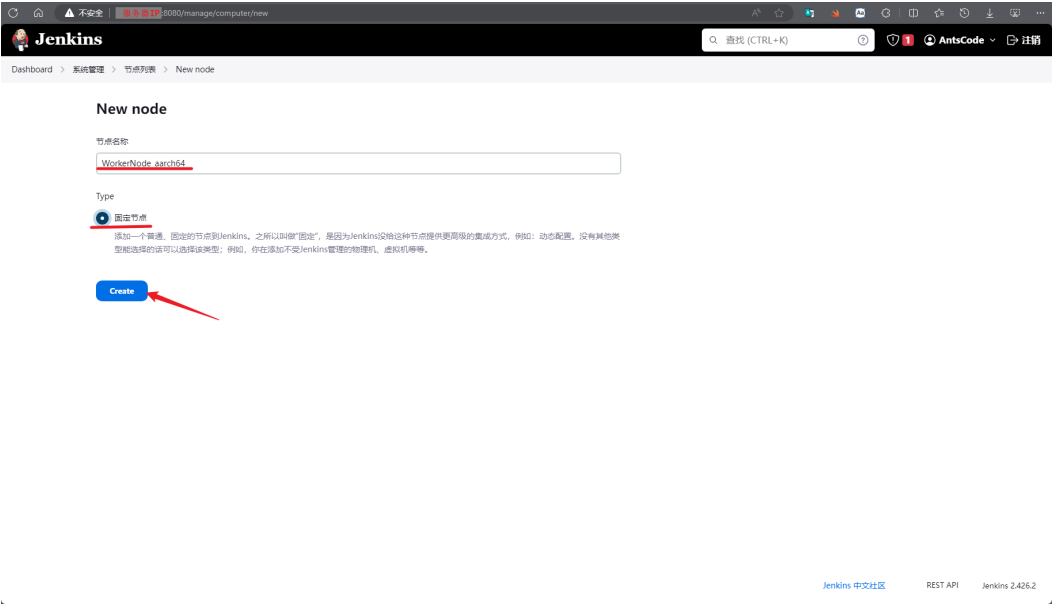
**步骤1** 添加工作节点。单击左侧树“系统管理> 节点和云管理 >”，打开“节点列表”页面，在页面右上角单击“+New Node”。

图 2-17 工作节点设置 1



**步骤2** 打开“New Node”页面，填写节点名称并选择节点类型。单击“Create”创建新的工作节点。

图 2-18 工作节点设置 2



说明

固定节点是指添加一个普通、固定的节点到Jenkins。


**步骤3** 单击工作节点右侧, 打开“Configure”页面，进行工作节点设置。详细配置参数说明如表2-1所示。

图 2-19 工作节点设置 3

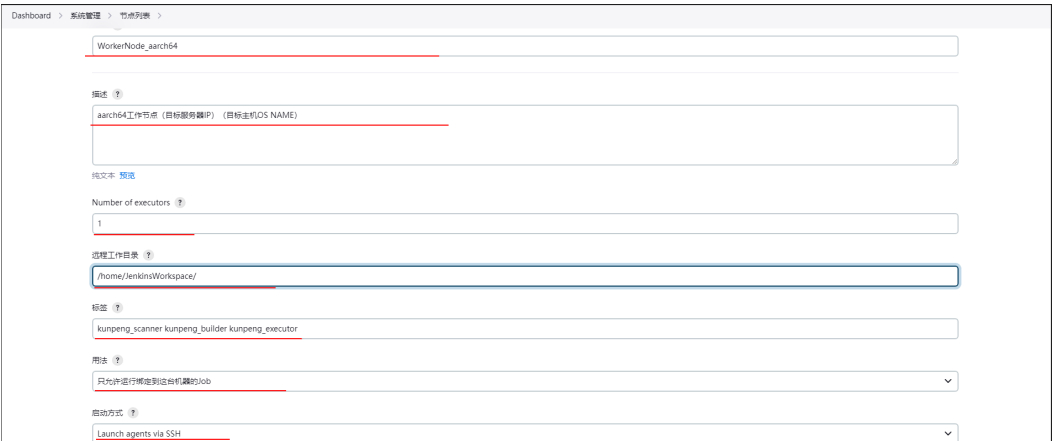


图 2-20 工作节点设置 4

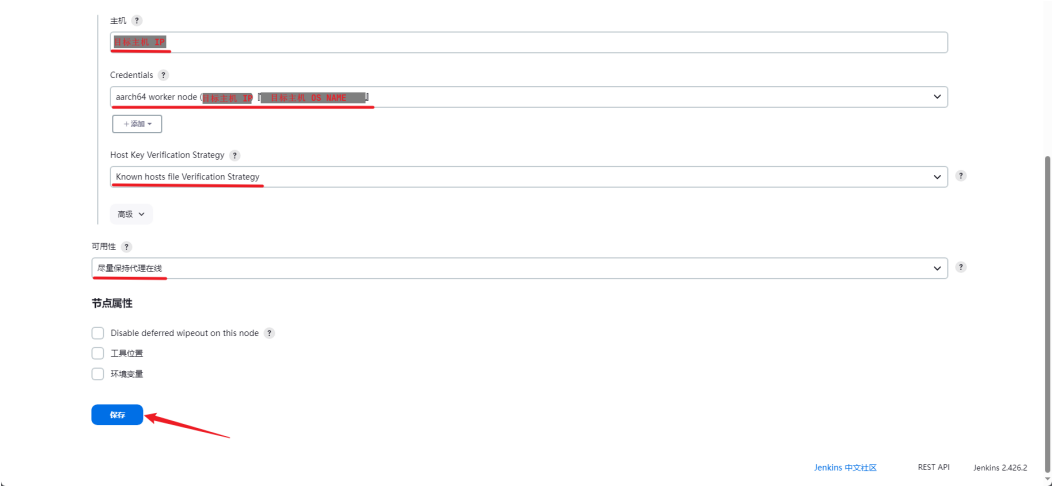


表 2-1 工作节点配置参数说明

配置项	配置说明
名称	与节点名称保持一致。
描述	按需填写，便于管理标识和即可。 例如：aarch64node(ip)openeuler22.03。
Number of executors	默认为1。
远程工作目录	远程工作目录。 例如：/home/JenkinsWorkspace/。
标签	流水线脚本中根据标签来选取执行机，可以打多个标签，用空格隔开，标签需要以“kunpeng_”为前缀。 例如：kunpeng_scanner、kunpeng_java_builder、kunpeng_c_cpp_builder、kunpeng_executor
用法	对当前工作节点进行用法描述。可选择： <ul style="list-style-type: none"><li>尽可能的使用这个节点</li><li>只允许运行绑定到这台机器的job</li></ul>
启动方式	选择当前工作节点的启动方式。可选择： <ul style="list-style-type: none"><li>Docker variant of Launch agents via SSH with SSH key injection</li><li>Launch agent via execution of command on the controller</li><li>Launch agents via SSH</li><li>通过Java Web启动代理</li></ul>
主机	节点IP地址。
Credentials	可选择已添加的凭据。

配置项	配置说明
Host Key Verification Strategy	可选择： <ul style="list-style-type: none"><li>• Know hosts file Verification Strategy</li><li>• Manually provided key Verification Strategy</li><li>• Manually trusted key Verification Strategy</li><li>• Non verifying Verification Strategy</li></ul>
高级	在高级设置中，可设置端口、Java路径、JVM选项、Prefix Start Agent Command、Suffix Start Agent Command、连接超时时间、最大重试次数。
可用性	工作节点可用性。可选择： <ul style="list-style-type: none"><li>• 尽量保持代理在线</li><li>• Use docker container only once</li><li>• 有需要的时候保持代理在线，当空闲时离线</li><li>• 根据时刻表让代理上线</li></ul>
节点属性	节点属性可选： <ul style="list-style-type: none"><li>• Disable deferred wipeout on this node</li><li>• Sidebar Links</li><li>• 工具位置</li><li>• 环境变量</li></ul>

----结束

# 3 部署 Gitlab

---

## 3.1 （可选）配置Gitlab

## 3.2 部署Gitlab Runner

## 3.1 （可选）配置 Gitlab

下面以x86环境搭建Gitlab为例。

**步骤1** 请执行以下命令下载Gitlab安装包。

```
wget https://mirrors.tuna.tsinghua.edu.cn/gitlab-ce/yum/el7/gitlab-ce-16.9.1-ce.0.el7.x86_64.rpm
```

**步骤2** 执行以下命令安装Gitlab。

```
yum install -y git tar policycoreutils-python openssh-server  
rpm -ivh gitlab-ce-16.9.1-ce.0.el7.x86_64.rpm
```

安装成功后会有如下回显信息，如[图3-1](#)所示。



5. 再次执行以下命令查看端口是否开通，提示“yes”表示端口已开通。  
firewall-cmd --query-port=8081/tcp

**步骤4** 执行以下命令重新加载配置。

```
gitlab-ctl reconfigure #重新生成相关配置文件，执行此命令时间比较长
```

**图 3-3 重新配置**

```
Running handlers:
[2024-02-27T14:42:04+08:00] INFO: Running report handlers
Running handlers complete
[2024-02-27T14:42:04+08:00] INFO: Report handlers complete
Infra phase complete, 589/1601 resources updated in 04 minutes 50 seconds

Notes:
Default admin account has been configured with following details:
Username: root
Password: You didn't opt-in to print initial root password to STDOUT.
Password stored to /etc/gitlab/initial_root_password. This file will be cleaned up in first reconfigure run after 24 hours.

NOTE: Because these credentials might be present in your log files in plain text, it is highly recommended to reset the password following https://docs.gitlab.com/ee/security/reset_user_password.html#reset-your-root-password.

gitlab Reconfigured!
```

**步骤5** 执行以下命令配置Gitlab开机自动启动。

```
systemctl enable gitlab-runsvdir.service
systemctl start gitlab-runsvdir.service
# 关闭Gitlab的自动启动命令：systemctl disable gitlab-runsvdir.service
```

**步骤6** 执行以下命令启动Gitlab。

```
gitlab-ctl restart
```

启动成功会有如下回显信息，如**图3-4**所示。

**图 3-4 启动 Gitlab**

```
[root@localhost home]# gitlab-ctl restart
ok: run: alertmanager: (pid 75137) 0s
ok: run: gitaly: (pid 75155) 0s
ok: run: gitlab-exporter: (pid 75242) 0s
ok: run: gitlab-kas: (pid 75553) 0s
ok: run: gitlab-workhorse: (pid 75580) 1s
ok: run: logrotate: (pid 75610) 0s
ok: run: nginx: (pid 75636) 0s
ok: run: node-exporter: (pid 75730) 1s
ok: run: postgres-exporter: (pid 75754) 0s
ok: run: postgresql: (pid 75794) 1s
ok: run: prometheus: (pid 75796) 0s
ok: run: puma: (pid 75920) 0s
ok: run: redis: (pid 75925) 0s
ok: run: redis-exporter: (pid 75950) 1s
ok: run: sidekiq: (pid 76017) 0s
```

**步骤7** 执行以下命令查看Gitlab版本。

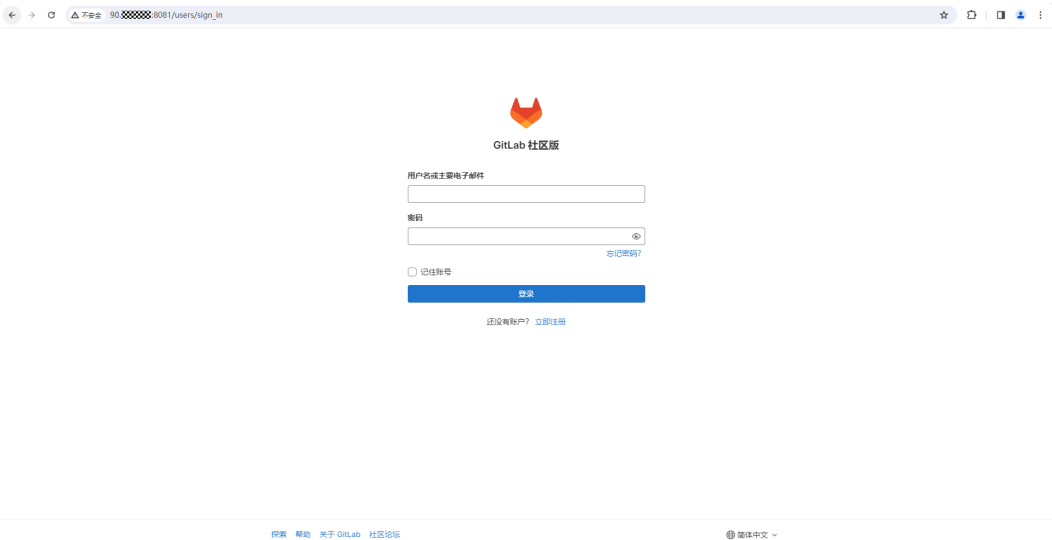
```
cat /opt/gitlab/embedded/service/gitlab-rails/VERSION # 回显应为16.9.1
```

**步骤8** 页面访问Gitlab。

```
http://ip:8081/ # 端口根据个人配置进行更改
```



图 3-5 访问 Gitlab

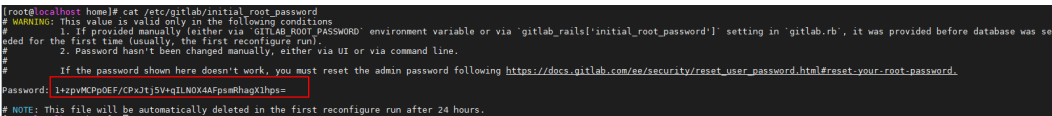


**步骤9** 登录Gitlab。登录Gitlab的默认账户名为root，密码存放在配置文件“/etc/gitlab/initial\_root\_password”中。

执行以下命令查看密码。

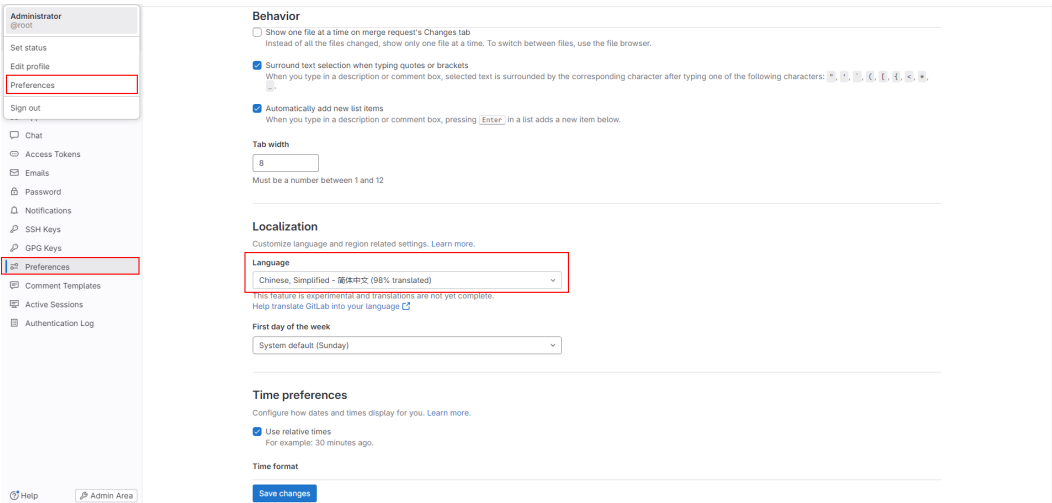
```
cat /etc/gitlab/initial_root_password
```

图 3-6 查看密码



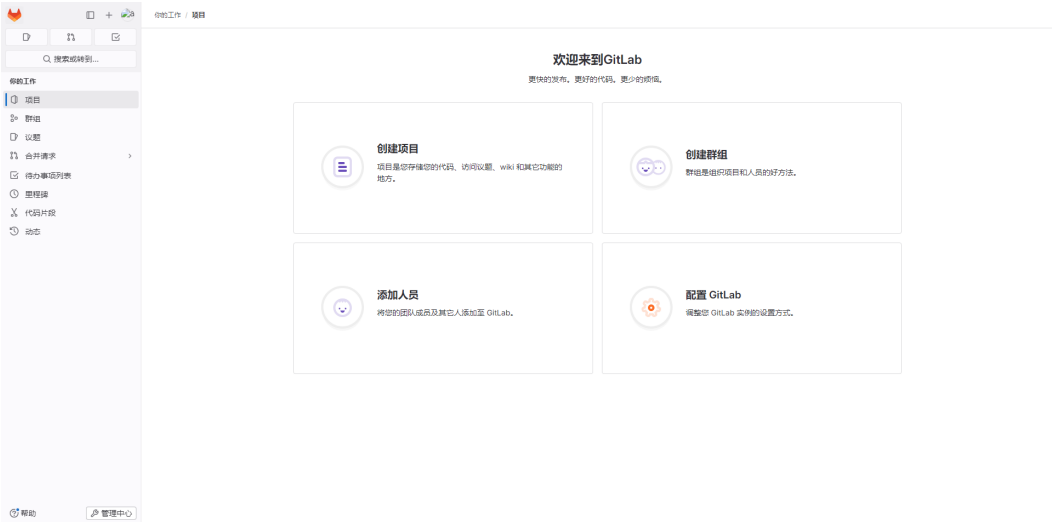
**步骤10** 设置Gitlab界面为简体中文。  
成功登录Gitlab后，在Gitlab后台的系统设置中配置简体中文。

图 3-7 设置简体中文



设置完成后刷新页面。

图 3-8 中文界面



----结束

### 3.2 部署 Gitlab Runner

- 步骤1 执行以下命令下载Gitlab Runner安装包。

```
wget https://mirrors.tuna.tsinghua.edu.cn/gitlab-runner/yum/el7-aarch64/gitlab-runner-16.9.0-1.aarch64.rpm
# 也可在https://mirrors.tuna.tsinghua.edu.cn/或https://gitlab-runner-downloads.s3.amazonaws.com/latest/index.html中下载符合自己环境的Gitlab Runer安装包，这里以gitlab-runner-16.9.0-1.aarch64.rpm为例，进行手动下载后传至服务器
```
- 步骤2 执行以下命令安装Gitlab Runner安装包。

```
yum install -y git tar
rpm -ivh gitlab-runner-16.9.0-1.aarch64.rpm
```
- 步骤3 执行以下命令指定gitlab-runner。

```
gitlab-runner uninstall
mkdir /home/Kunpeng_staff
gitlab-runner install --working-directory /home/Kunpeng_staff --user root
# 若想指定用户运行
useradd Kunpeng_staff
gitlab-runner install --working-directory /home/Kunpeng_staff --user Kunpeng_staff
```
- 步骤4 执行以下命令启动gitlab-runner。

```
systemctl daemon-reload      #重新加载配置
systemctl start gitlab-runner #启动服务
systemctl enable gitlab-runner #设置开机启动
systemctl restart gitlab-runner #重启服务
```
- 步骤5 执行以下命令查看gitlab-runner。

```
systemctl status gitlab-runner
```

成功启动如图3-9所示。

图 3-9 启动 gitlab-runner

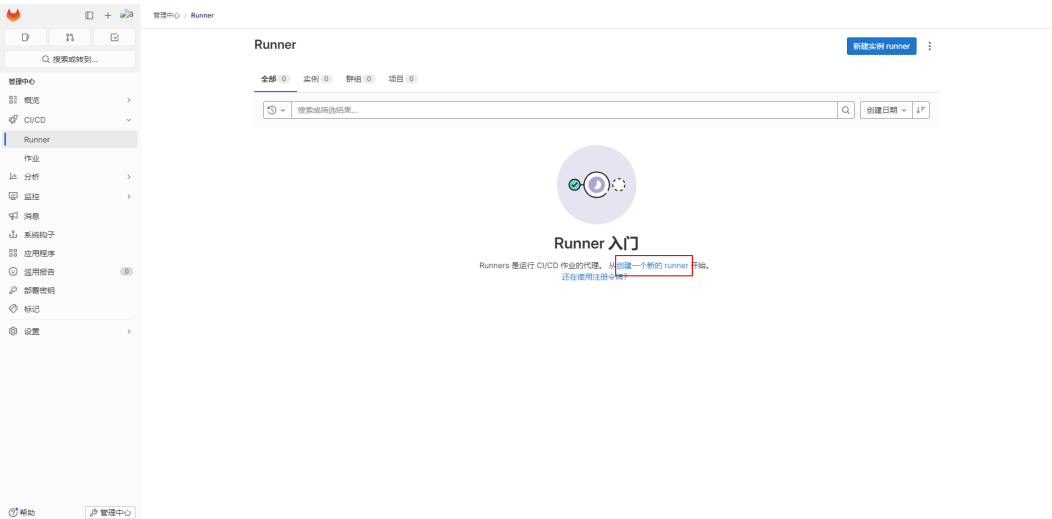
```
root@localhost home# systemctl status gitlab-runner
● gitlab-runner.service - GitLab Runner
   Loaded: loaded (/etc/systemd/system/gitlab-runner.service; enabled; vendor preset: disabled)
   Active: active (running) since Sat 2016-02-27 16:08:09 CST; 36min ago
     Main PID: 123788 (gitlab-runner)
       Tasks: 40 (limit: 814496)
      Memory: 27.0M
      CGroup: /system.slice/gitlab-runner.service
              └─ 123788 /usr/bin/gitlab-runner run --working-directory /home/gitlab-runner --config /etc/gitlab-runner/config.toml --service gitlab-runner --user gitlab-runner

2月 27 16:08:09 localhost.localdomain gitlab-runner[123788]: Runtime platform arch=arm64 os=linux pid=123788 revision=656c1943 version=16.9.0
2月 27 16:08:09 localhost.localdomain gitlab-runner[123788]: Starting multi-runner from /etc/gitlab-runner/config.toml... builds=0 max_builds=0
2月 27 16:08:09 localhost.localdomain gitlab-runner[123788]: Running in system-mode.
2月 27 16:08:09 localhost.localdomain gitlab-runner[123788]: Created missing unique system ID system_id=s_c2c042740ed2 builds=0 max_builds=1
2月 27 16:08:09 localhost.localdomain gitlab-runner[123788]: Configuration loaded
2月 27 16:08:09 localhost.localdomain gitlab-runner[123788]: listen_address not defined, metrics & debug endpoints disabled builds=0 max_builds=1
2月 27 16:08:09 localhost.localdomain gitlab-runner[123788]: [session_server].listen_address not defined, session endpoints disabled builds=0 max_builds=1
2月 27 16:08:09 localhost.localdomain gitlab-runner[123788]: Initializing executor providers builds=0 max_builds=1
2月 27 16:10:36 localhost.localdomain systemd[1]: gitlab-runner.service: Current command vanished from the unit file, execution of the command list won't be resumed.
```

- 步骤6** 执行以下命令设置权限。
- chown -R root:root /home/gitlab-runner  
# 指定单一用户运行  
chown -R Kunpeng\_staff.Kunpeng\_staff /home/gitlab-runner
- 步骤7** Gitlab Runner注册服务。

1. 登录Gitlab，单击左侧树管理中心“Runner”按钮，打开Runner页面，再次单击“创建一个新的runner”，创建新的runner。

图 3-10 创建 runner



2. 填写标签时，若存在多个请用逗号隔开，标签只可为kunpeng\_scanner、kunpeng\_c\_cpp\_builder、kunpeng\_java\_builder、kunpeng\_executor，其它配置项请按需填入，单击“创建runner”。

图 3-11 填写标签

管理中心

概览

CICD

Runner

作业

分析

监控

消息

系统构子

应用程序

应用部署

部署策略

标记

设置

搜索或转到...

Linux

macOS

Windows

Docker

Kubernetes

标签

标签

添加标签以指定 Runner 可运行的作业。了解详情。

kunpeng\_scanner, kunpeng\_builder, kunpeng\_executor

使用逗号分隔多个标签。例如, iadcs, shared。

☐ 运行未打标签的作业

除了标记的任务外, 使用 Runner 来执行没有标签的任务。

配置 (可选)

Runner 描述

Worknode(66.53)Openeuler22.03

☐ 已部署

停止 Runner 接收新的作业。

☐ 受保护

只为受保护的分支使用此流水线上的Runner。

最大作业超时

Runner 在结束前可以运行的最大时间。如果一个项目的任务超时时间较短, 则使用实际 Runner 的任务超时时间。

72000

单位为秒。输入作业超时时间, 必须至少 600 秒。

创建 runner

3. Runner创建成功后需要注册Runner，如图3-12所示，请根据页面中的url、token内容去到Gitlab Runner环境进行注册。

图 3-12 注册 Runner

管理中心

概览

CICD

Runner

作业

分析

监控

消息

系统构子

应用程序

应用部署

部署策略

标记

设置

搜索或转到...

Runner 已创建。

注册 "Worknode(66.53)Openeuler22.03" runner

必须先安装 Runner 才能注册 runner, 如何安装 Runner?

步骤 1

复制并粘贴以下命令到您的命令行中, 注册 runner。

```
gitlab-runner-username
--url http://10.53.8081
--token glnt-txf0M6PzybyQutggQzJ0
```

Runner 身份验证令牌 glnt-txf0M6PzybyQutggQzJ0 在此处复制显示。在您注册 Runner 后, 此令牌将保存在 .config.toml 中, 并且无法从界面中再次访问。

步骤 2

当命令行提示时, 选择一个执行器。执行器在不同的环境中运行构建。不确定选择哪一个?

步骤 3 (可选)

手动验证 runner 是否可以选择作业。

\$ gitlab-runner run

如果您将 runner 作为 系统或用户服务 运行, 则可能不需要。

进入 runners 页面

4. 单击“进入runners页面”，查看Runners相关信息

图 3-13 进入 runners 页面

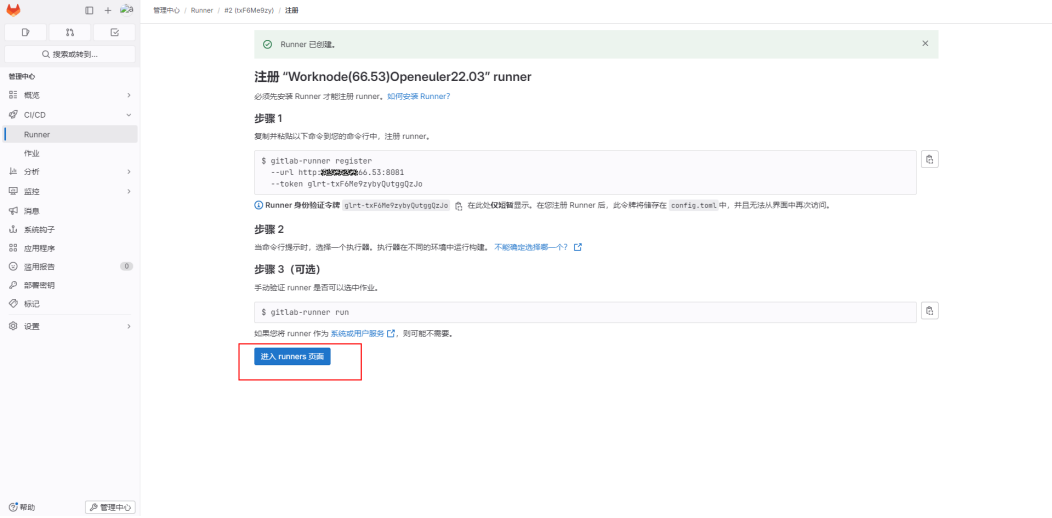
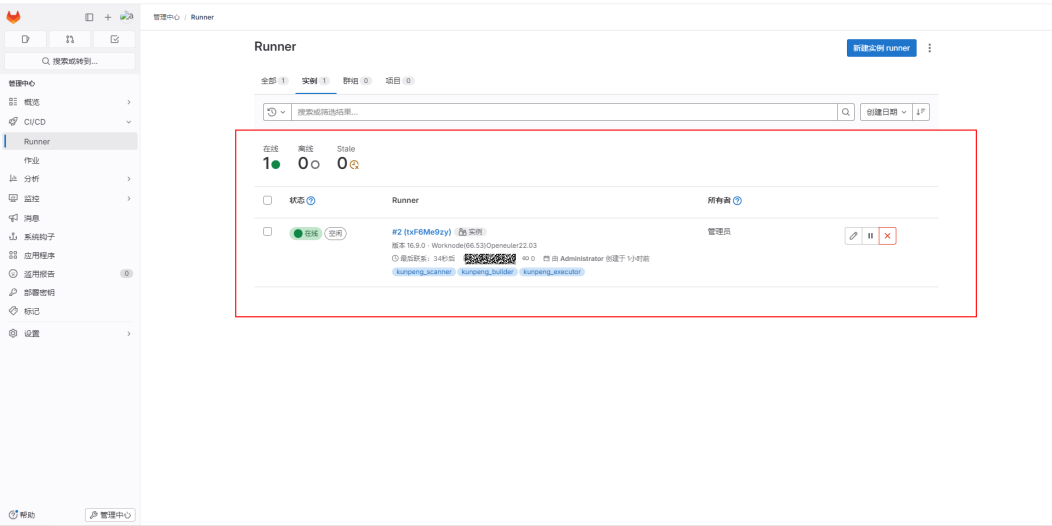


图 3-14 查看 Runners 信息



**步骤8** 进行脚本配置，创建或者导入项目后，进入项目中，在目录中选择构建，在流水线编辑器中进行编辑。

- 毕昇编译器调用示例：

```
stages:
- build

clang_job:
stage: build
tags:
- kunpeng_scanner #对应gitlab-runner注册时的标签，可选择多个
script:
- /root/BiShengCompiler-3.2.0-aarch64-linux/bin/clang /opt/test.c -o 输出路径 #所在机器上应保障已安装毕昇编译器,路径根据视情况填写
```

在流水线脚本中，在某一步骤需要调用毕昇编译器进行编译时，以目标文件为“/opt/test.c”文件为例。
- GCC for openEuler调用示例：

```
stages:
- build
```

```
gcc_job:
  stage: build
  tags:
    - kunpeng_scanner #对应gitlab-runner注册时的标签，可选择多个
  script:
    - /root/gcc-10.3.1-2023.12-aarch64-linux/bin/gcc /opt/test.c -o 输出路径 #所在机器上应保障
    已安装GCC for openEule,路径根据视情况填写
```

在流水线脚本中，在某一步骤需要调用GCC for openEuler进行编译时，以目标文件为“/opt/test.c”文件为例。

----结束