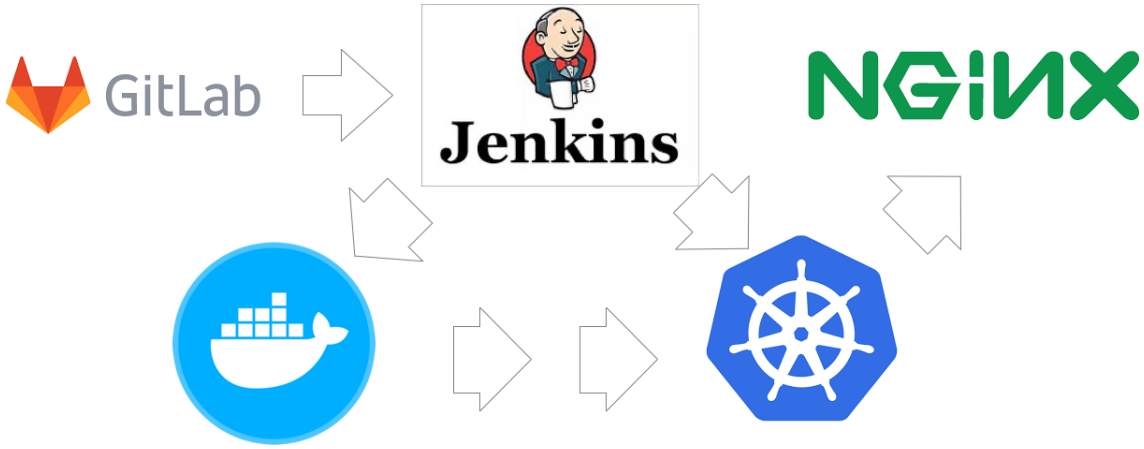
讲师(老司机)

容器虚拟化技术和自动化部署



jenkins运维篇

凭据管理

凭据

类型	提供者	存储 ↓	域	唯一标识	名称
		Jenkins	全局	58673430-367a-4983-b156-88c668abe813	root (jenkinsagent-154-SSH)
		Jenkins	全局	b26bd84e-e0cb-4b90-8469-1c2a46213466	root/****** (gitlab-154-http)
		Jenkins	全局	c8634952-4993-4455-b164-35427823144f	root (gitlab-154-SSH)

图标: 小 中 大

Stores scoped to Jenkins

提供者	存储 ↓	域
Jenkins		全局

插件管理

可更新 | 可选插件 | 已安装 | 高级

启用	名称 ↓	版本	上一个安装的版本	卸载
☑	Apache HttpComponents Client 4.x API Plugin Bundles Apache HttpComponents Client 4.x and allows it to be used by Jenkins plugins.	4.5.10-2.0		卸载
☑	bouncycastle API Plugin This plugin provides an stable API to Bouncy Castle related tasks.	2.18		卸载
☑	Branch API Plugin This plugin provides an API for multiple branch based projects.	2.5.9		卸载
☑	Build Timeout This plugin allows builds to be automatically terminated after the specified amount of time has elapsed.	1.19.1		卸载
☑	Command Agent Launcher Plugin Allows agents to be launched using a specified command.	1.4		卸载
☑	Credentials Binding Plugin Allows credentials to be bound to environment variables for use from miscellaneous build steps.	1.23		卸载
☑	Credentials Plugin This plugin allows you to store credentials in Jenkins.	2.3.11		卸载
☑	Display URL API			

用户管理

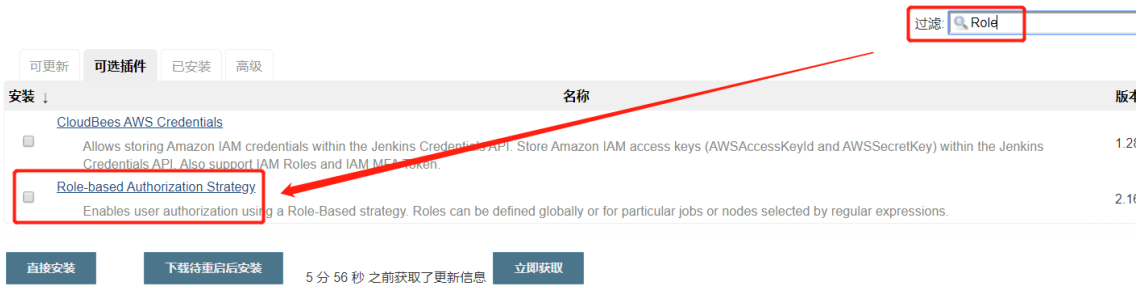
创建用户

1. devmanager
2. testmanager

用户权限管理

安装Role-based Authorization Strategy插件

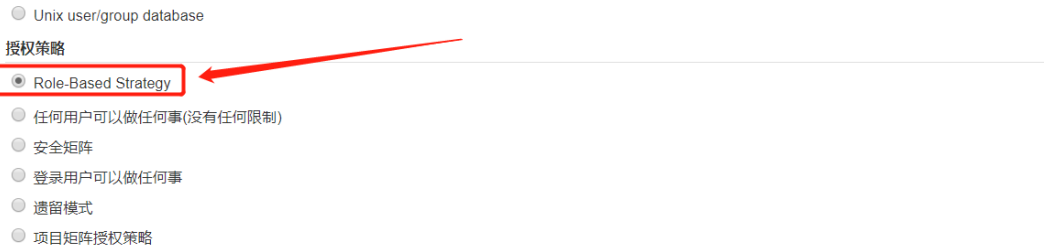
利用Role-based Authorization Strategy 插件来管理Jenkins用户权限



安装成功后重新启动jenkins。这两个用户创建成功后是没有任何权限的。登录jenkins也不能做任何事情。必须要给用户分配权限。

开启权限全局安全配置

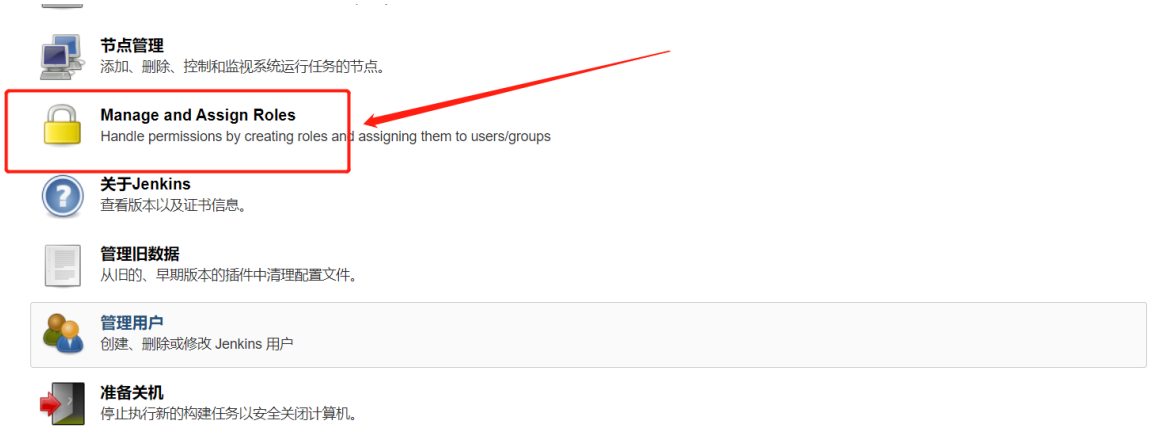
授权策略切换为"Role-Based Strategy"，点击"保存"按钮



器

创建角色

1. baserole：用于登录jenkins
2. devrole：管理所有lagoudev.*开头的任务
3. testrole：管理所有lagoutest.*开头的任务



jenkiens默认角色

1. Global roles（全局角色）：管理员等高级用户可以创建基于全局的角色
2. Item roles（项目角色）：针对某个或者某些项目的角色
3. Node roles（节点角色）：节点相关的权限

groovy入门

简介

Groovy 是运行在 JVM 中的另外一种语言，我们可以用 Groovy 在 Java 平台上进行编程，使用方式基本与使用 Java 代码的方式相同，所以如果你熟悉 Java 代码的话基本上不用花很多精力就可以掌握 Groovy 了，它的语法与 Java 语言的语法很相似，而且完成同样的功能基本上所需要的 Groovy 代码量会比 Java 的代码量少。

官网地址

```
http://groovy-lang.org/download.html
```

配置groovy

```
GROOVY_HOME:D:\groovy-3.0.4
PATH: %GROOVY_HOME%\bin;
```

测试groovy环境

```
cmd
groovy -v || groovy --version

```

idea集成groovy

与创建JDK步骤类似。在创建工程时候配置groovy。

src/com/laogou/hello/

```
package com.lagou.hello

class helloGroovy {
    public static void main(String[] args) {
        def name="laosiji";
        println("groovy hello!! ,"+name)
        println('单引号，中文、分号测试');
    }
}
```

总结

1. 从输出结果可以看出了 Groovy 里面支持`单引号`和`双引号`两种方式，注释支持`//`和`/**/`两种方式，而且不以分号";"结尾也可以，但是我们还是推荐都带上分号保持代码的一致性。
2. `标识符`被用来定义变量，函数或其他用户定义的变量。标识符以**字母、美元或下划线**开头，不能以数字开头。

数据类型

除了字符串之外，Groovy 也支持有符号整数、浮点数、字符等。

```
class Example1 {
   public static void main(String[] args) {
        String str = "Hello";  // 字符串
        int x = 5;  // 整数
        long y = 100L;  // 长整型
        float a = 10.56f;  // 32位浮点数
        double b = 10.5e40;  // 64位浮点数
        char c = 'A';  // 字符
        Boolean l = true;  // 布尔值，可以是true或false。
        println(str);
        println(x);
        println(y);
        println(a);
        println(b);
        println(c);
        println(l);
    }
}
```

打印变量

用 def 关键字来定义变量，当然也可以用一个确定的数据类型来声明一个变量，我们可以用下面的几种方式来打印变量

```
class Example2 {
   public static void main(String[] args) {
        // 初始化两个变量
        int x = 5;
        int y = 6;

        // 打印变量值
        println("x = " + x + " and y = " + y);
        println("x = ${x} and y = ${y}");
        println('x = ${x} and y = ${y}');
    }
}
```

注意事项

从这里我们可以看出 Groovy 在单引号的字符串里面是不支持插值的，这点非常重要，很多同学在使用 Pipeline 脚本的时候经常会混淆。

除此之外，还支持三引号 : Groovy 里面三引号支持双引号和单引号两种方式，但是单引号同样不支持插值，要记住。

```
class Example3 {
   public static void main(String[] args) {
        // 初始化两个变量
        int x = 5;
        int y = 6;

        println """
        x = ${x}
        X = ${y}
        """

        println '''
        x = ${x}
        X = ${y}
        '''
    }
}
```

函数

Groovy 中的函数是使用返回类型或使用 def 关键字定义的，函数可以接收任意数量的参数，定义参数时，不必显式定义类型，可以添加修饰符，如 public，private 和 protected，默认情况下，如果未提供可见性修饰符，则该方法为 public 。

```
class Example4 {
   static def PrintHello() {
      println("This is a print hello function in groovy");
   }

   static int sum(int a, int b, int c = 10) {
      int d = a+b+c;
      return d;
   }

   public static void main(String[] args) {
```

```
        PrintHello();
        println(sum(5, 50));
    }
}
```

条件语句

在我们日常工作中条件判断语句是必不可少的，即使在 Jenkins Pipeline 脚本中也会经常遇到，Groovy 里面的条件语句和其他语言基本一致，使用 if/else 判断

```
class Example5 {
  public static void main(String[] args) {
      // 初始化变量值
      int a = 2

      // 条件判断
      if (a < 100) {
         // 如果a<100打印下面这句话
         println("The value is less than 100");
      } else {
         // 如果a>=100打印下面这句话
         println("The value is greater than 100");
      }
   }
}
```

循环语句

除了条件判断语句之外，循环语句也是非常重要的，Groovy 中可以使用三种方式来进行循环：`while、for语句、for-in语句`

```
class Example6 {
   public static void main(String[] args) {
      int count = 0;
      println("while循环语句: ");
      while(count<5) {
         println(count);
         count++;
      }

      println("for循环语句: ");
      for(int i=0;i<5;i++) {
         println(i);
      }

      println("for-in循环语句: ");
      int[] array = [0,1,2,3];
      for(int i in array) {
         println(i);
      }

      println("for-in循环范围: ");
      for(int i in 1..5) {
         println(i);
      }
   }
```

```
}
```

除了上面这些最基本的特性外，Groovy 还支持很多其他的特性，比如异常处理、面向对象设计、正则表达式、泛型、闭包等等，由于我们这里只是为了让大家对 Jenkins Pipeline 的脚本有一个基本的认识，更深层次的用法很少会涉及到，大家如果感兴趣的可以去查阅官方文档了解更多信息。

jenkins共享库入门

简介

实际生产环境中，运维工程师经常使用 Jenkins Pipeline 一定会遇到多个不同流水线中有大量重复代码的情况，很多时候为了方便我们都是直接复制粘贴到不同的Jenkinsfile文件中，但是长期下去这些代码的维护就会越来越麻烦。为了解决这个问题，Jenkins 中提供了共享库的概念来解决重复代码的问题，我们只需要将公共部分提取出来，然后就可以在所有的 Pipeline 中引用这些共享库下面的代码了。

什么是共享库

共享库（shared library）是一些**独立的 Groovy 脚本的集合**，我们可以在运行 Pipeline 的时候去获取这些共享库代码。使用共享库最好的方式同样是把代码使用 Git 仓库进行托管，这样我们就可以进行版本化管理了。当然我们也需要一些 Groovy 语言的基础，不过并不需要多深入，基本的语法概念掌握即可，可以查看前面我们的Groovy 简明教程。

使用共享库一般只需要3个步骤即可：

- 首先创建 Groovy 脚本，添加到 Git 仓库中
- 然后在 Jenkins 中配置将共享库添加到 Jenkins 中来
- 最后，在我们的流水线中导入需要使用的共享库：Jenkins老版本中使用`@Library('your-shared-library')`，这样就可以使用共享库中的代码。新版本中`Library 'your-shared-library'`，这样就可以使用共享库中的代码。

共享库目录

官网示例

```
(root)
+- src                     # Groovy source files
|   +- org
|       +- foo
|           +- Bar.groovy  # for org.foo.Bar class
+- vars
|   +- foo.groovy          # for global 'foo' variable
|   +- foo.txt             # help for 'foo' variable
+- resources               # resource files (external libraries only)
|   +- org
|       +- foo
|           +- bar.json    # static helper data for org.foo.Bar
```

sayHello.groovy

在工程中新建vars目录。

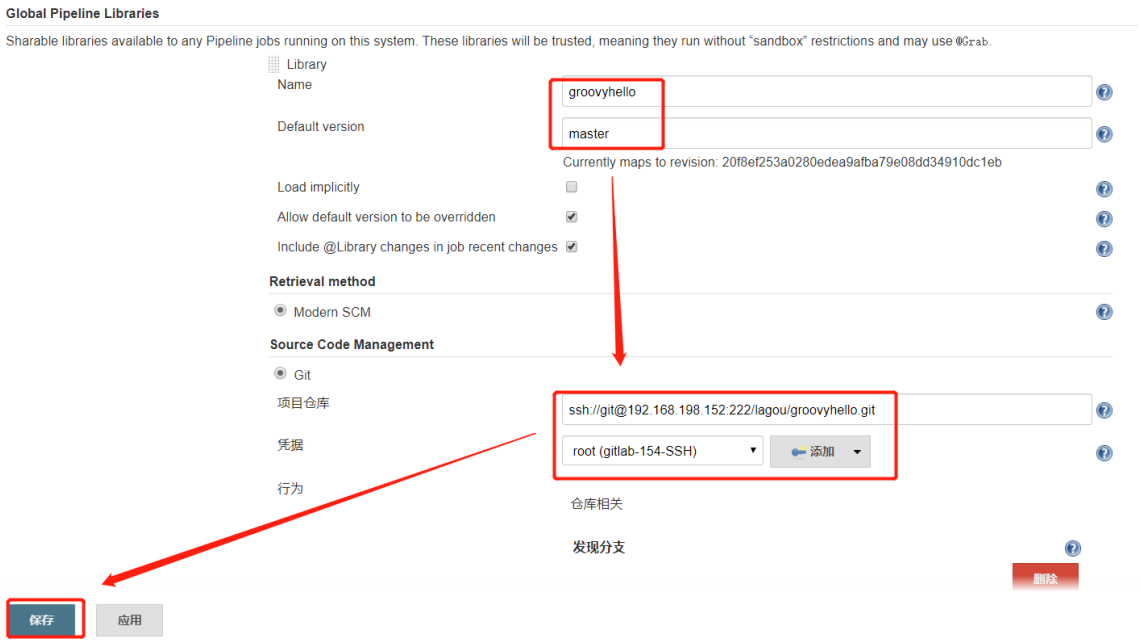
idea设置：project structure->moudle->source，将vars目录标记为Sources目录。如果不标记为Sources目录，vars下不能创建*.groovy文件。

在目录vars/sayHello.groovy新建groovy文件，文件内容如下：

```
def call(String name = "laosiji") {
    println("hello $name")
}
```

配置共享库

jenkins工作台->系统管理->系统配置-> Global Pipeline Libraries



pipeline任务

```
//引入SharedLibrary库
library "groovyhello"
pipeline {
    agent {
      label 'jenkinsagent-154'
    }

    stages {
        stage('Hello') {
            steps {
                echo 'Hello World'
                //通过groovy脚本名称直接调用
                sayHello "lagou"
            }
        }
    }
}
```

测试pipeline任务

```
点击->立即构建
```

src目录测试

src/com.lagou.hello.ansible.groovy

```
package com.lagou.hello

def printMessage(){
    println("hello laosiji")
}
```

Jenkinsfile文件

```
@Library("groovyhello")_

def build = new com.lagou.hello.ansible()
pipeline {
    agent {
      label 'jenkinsagent-154'
    }

    stages {
        stage('Hello') {
            steps {
                echo 'Hello World'

                sayHello "lagou"
                script{
                   build.printMessage()
                }
            }
        }
    }
}
```