(/)

Java 调用 Linux 命令实战（含完整代码）

(/gitchat/author/5e01871c1a98202c00216a22)

老牛 (/gitchat/author/5...

11年研发经验，某大型互联网公司技术经理，个人爱好是Java技术总结。技术不是短时内就可以搞精搞透，我们需要善于思考，善于总结，善于沉淀。

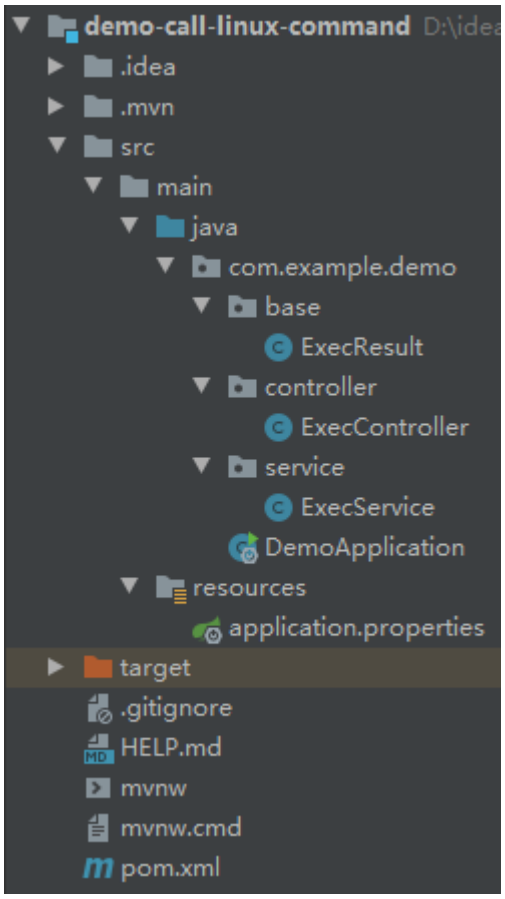
前言

我所在项目组刚接到一个 Java 重启 keepalived 需求，具体需求是通过 Java 接口去操作安装在 Linux 系统上的 keepalived，根据参数内容执行具体命令，比如：参数是 restart 则执行 `systemctl restart keepalived` 命令，并返回执行成功失败标记。

本场 Chat 详细介绍了如何使用 Java 语言调用 Linux 命令，其中包括：Java 接口如何执行 shell 脚本、如何执行 keepalived 的相关命令，列出详细使用步骤等等，并贴出完整的代码。

具体开发过程及代码分析

代码结构



pom.xml 文件配置

下面配置是工程需要使用的所有 jar，引入 spring-boot-starter 和 spring-boot-starter-web 即可，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http:
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://m
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.5.6</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>demo-call-linux-command</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>demo-call-linux-command</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

工程配置文件和启动类

application.properties 仅仅配置了服务端口，启动类也是非常简单，具体代码如下：

```
application.properties 文件:
server.port=8888


package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

}
```

Java 调用 Linux 命令代码

Java 调用 Linux 命令其实很容易实现，JDK 运用建造者等设计模式已经封装好了，使用起来也特别方便。ExecResult 类是接受执行命令后返回的结果，主要代码都在 ExecService 类，真正工作的是底层的 Process 类，代码如下：

```java
package com.example.demo.base;

import java.util.List;

public class ExecResult {

    private int exitCode;
    private List<String> result;

    public ExecResult() {
    }

    public ExecResult(int exitCode, List<String> result) {
        this.exitCode = exitCode;
        this.result = result;
    }

    public int getExitCode() {
        return exitCode;
    }

    public void setExitCode(int exitCode) {
        this.exitCode = exitCode;
    }

    public List<String> getResult() {
        return result;
    }

    public void setResult(List<String> result) {
        this.result = result;
    }

    @Override
    public String toString() {
        return "ExecResult{" +
                "exitCode=" + exitCode +
                ", result=" + result +
                '}';
    }
}
```

```java
package com.example.demo.controller;
```

```
import com.example.demo.base.ExecResult;
import com.example.demo.service.ExecService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ExecController {

    @Autowired
    private ExecService execService;


    @PostMapping("/call/linux/command")
    public ExecResult callCommand(@RequestParam("command") String c
        ExecResult execResult = execService.execCommand(execService
        if (execResult.getExitCode() == 0) {
            System.out.println("success");
        }else{
            System.out.println("fail");
        }
        return execResult;
    }
}


package com.example.demo.service;

import com.example.demo.base.ExecResult;
import org.springframework.stereotype.Component;

import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.util.stream.Collectors;

@Component
public class ExecService {

    /**
     * 执行命令
     * @param commands
     * @return
```

```
 */
public ExecResult execCommand(List<String> commands) {
    System.out.println("ExecCommand: " + commands);

    OutputStream os = null;
    InputStream is = null;

    ExecResult execResult = new ExecResult();
    ProcessBuilder processBuilder = new ProcessBuilder(commands
    processBuilder.redirectErrorStream(true);
    Process process = null;
    try {
        process = processBuilder.start();
        execResult.setExitCode(process.waitFor());
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
    os = process.getOutputStream();
    is = process.getInputStream();
    try {
        os.close();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
    BufferedReader br = new BufferedReader(new InputStreamReade
    List<String> result = br.lines().filter(l -> l != null).col
    System.out.println("ExecCommand result:" + result);
    execResult.setResult(result);
    return execResult;
}

/**
 * 使用正则表达式对参数格式化处理
 * 比如：systemctl start keepalived 如果不通过正则会被以空格分割拆分 3
 *       /home/jar/test.sh start 如果不通过正则会被以空格分割拆分 2 个
 * @param param
 * @return
 */
public List<String> parseCommand(String param) {
    List<String> result = new ArrayList<String>();
    Pattern pattern = Pattern.compile("((?<=')')[\\w|[\\W&&[^']
    Matcher matcher = pattern.matcher(param);
    while (matcher.find()) {
        result.add(matcher.group());
    }
    return result;
```

```
}
```

Shell 脚本 test.sh 文件

该脚本主要功能是接受一个参数，根据参数内容执行具体的命令，并打印相关信息，此处借用操作 keepalived 场景进行编码，代码如下：

```
#!/bin/bash

operate="$1"

if [ -z $operate ]; then
        echo "请输入参数，start 代表开启 keepalived，stop 代表停止，resta
        exit 8
fi

echo "你输入的参数是："$operate

if [ $operate = "start" ] ; then
    systemctl start keepalived
    rc=$?
    if  [ ${rc} -ne 0 ]; then
        echo "启动 keepalived 失败"
        exit ${rc}
    fi
    echo "Alreadyed start keepalived"
elif [ $operate = "stop" ] ; then
    systemctl stop keepalived
    rc=$?
    if  [ ${rc} -ne 0 ]; then
        echo "停止 keepalived 失败"
        exit ${rc}
    fi
    echo "Alreadyed stop keepalived"
elif [ $operate = "restart" ] ; then
    systemctl restart keepalived
    rc=$?
    if  [ ${rc} -ne 0 ]; then
        echo "重启 keepalived 失败"
        exit ${rc}
    fi
    echo "Alreadyed restart keepalived"
elif [ $operate = "test" ] ; then
    systemctl restart keepalived123
    rc=$?
    if  [ ${rc} -ne 0 ]; then
        echo "重启 keepalived 失败"
        exit ${rc}
    fi
    echo "Alreadyed restart keepalived"
else
    echo "参数格式有误，请重新输入，start 代表开启 keepalived，stop 代表停止
fi
```

(/)

使用步骤及环境准备

1. 通过 maven 打包工程得到一个 jar 包，比如：demo-call-linux-command-0.0.1-SNAPSHOT.jar。

```
[INFO] --- maven-compiler-plugin:3.8.1:testCompile (default-testCompile) @ demo-call-linux-command ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ demo-call-linux-command ---
[INFO] Tests are skipped.
[INFO]
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ demo-call-linux-command ---
[INFO] Building jar: D:\idea\demo-call-linux-command\target\demo-call-linux-command-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.5.6:repackage (repackage) @ demo-call-linux-command ---
[INFO] Replacing main artifact with repackaged archive
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  5.953 s
[INFO] Finished at: 2021-11-05T11:57:57+08:00
[INFO] ------------------------------------------------------------------------
```

2. 把 jar 和 test.sh 上传到 linux 系统某个目录，比如：/home/jar/demo-call-linux-command-0.0.1-SNAPSHOT.jar。

```
[root@s[redacted] jar]# pwd
/home/jar
[root@[redacted] jar]# ll
total 17092
-rw-r--r-- 1 root root 17495444 Nov  5 11:58 demo-call-linux-command-0.0.1-SNAPSHOT.jar
-rwxr-xr-x 1 root root     1071 Nov  5 10:50 test.sh
[root@[redacted] jar]#
```

3. 运行 jar 包，命令如下：

```
java -jar demo-call-linux-command-0.0.1-SNAPSHOT.jar
```

```
[root@shadow-v2 jar]# java -jar demo-call-linux-command-0.0.1-SNAPSHOT.jar

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.5.6)

2021-11-05 14:51:19.608  INFO 74767 --- [           main] com.example.demo.DemoApplication         : Starting DemoApplication v0.0.1-SNAPSHOT using Java 1.8.0_171 with
 PID 74767 (/home/jar/demo-call-linux-command-0.0.1-SNAPSHOT.jar started by root in /home/jar)
2021-11-05 14:51:19.611  INFO 74767 --- [           main] com.example.demo.DemoApplication         : No active profile set, falling back to default profiles: default
2021-11-05 14:51:21.968  INFO 74767 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8888 (http)
2021-11-05 14:51:22.013  INFO 74767 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2021-11-05 14:51:22.013  INFO 74767 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.54]
2021-11-05 14:51:22.146  INFO 74767 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2021-11-05 14:51:22.146  INFO 74767 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2411 ms
2021-11-05 14:51:23.606  INFO 74767 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8888 (http) with context path ''
2021-11-05 14:51:23.625  INFO 74767 --- [           main] com.example.demo.DemoApplication         : Started DemoApplication in 5.199 seconds (JVM running for 6.007)
```

4. 安装一个 keepalived，命令如下：

```
yum install -y curl gcc openssl-devel libnl3-devel net-snmp-devel
yum install -y keepalived


### 以下命令不需运行，仅作参考
systemctl start keepalived     //启动 keepalived
systemctl enable keepalived    //加入开机启动 keepalived
systemctl restart keepalived   //重新启动 keepalived
systemctl status keepalived    //查看 keepalived 状态
```

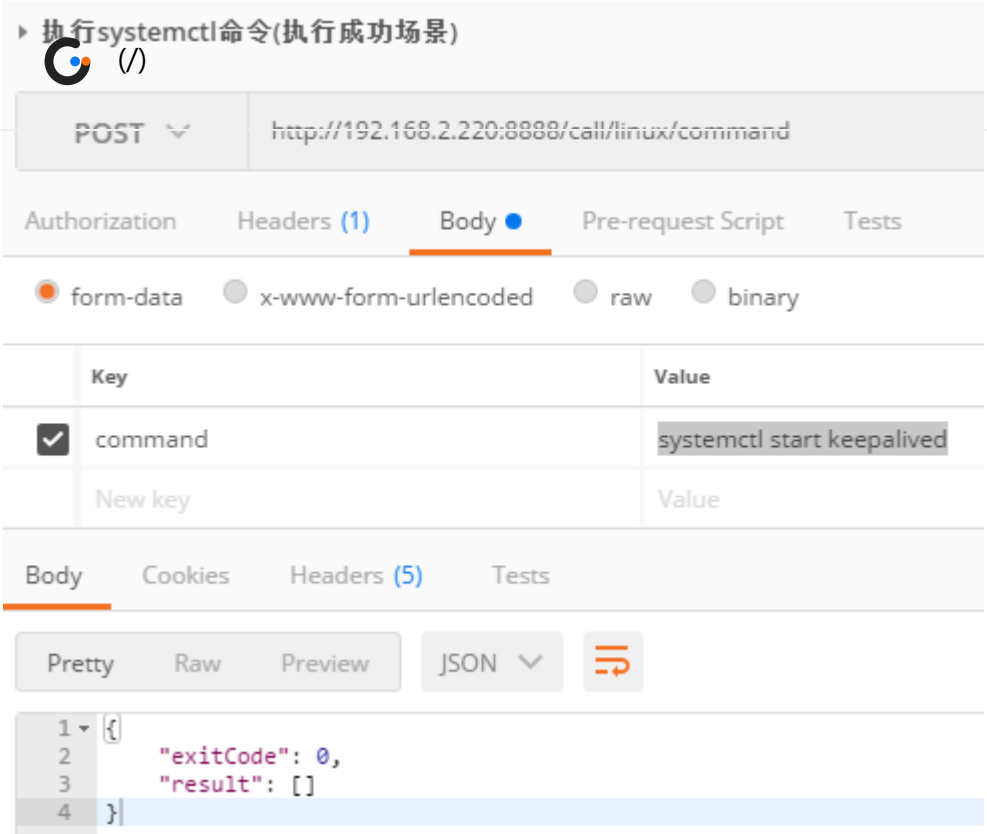
代码测试

1. 执行 systemctl 命令（执行成功场景）

- url：http://192.168.2.220:8888/call/linux/command
- form-data 参数：
  - command： `systemctl start keepalived`

返回结果：

```
{
    "exitCode": 0,
    "result": []
}
```

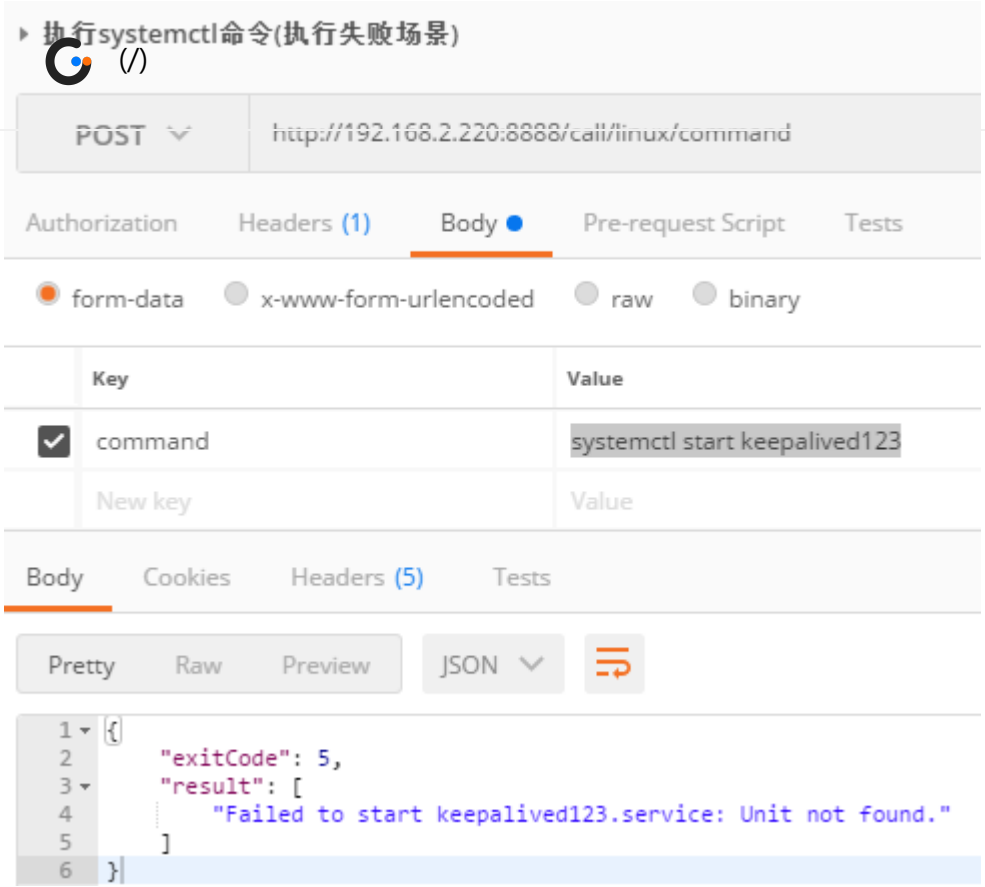


2. 执行 systemctl 命令（执行失败场景）

- url：http://192.168.2.220:8888/call/linux/command
- form-data 参数：
  - command： `systemctl start keepalived123`

返回结果：

```
{
    "exitCode": 5,
    "result": [
        "Failed to start keepalived123.service: Unit not found."
    ]
}
```

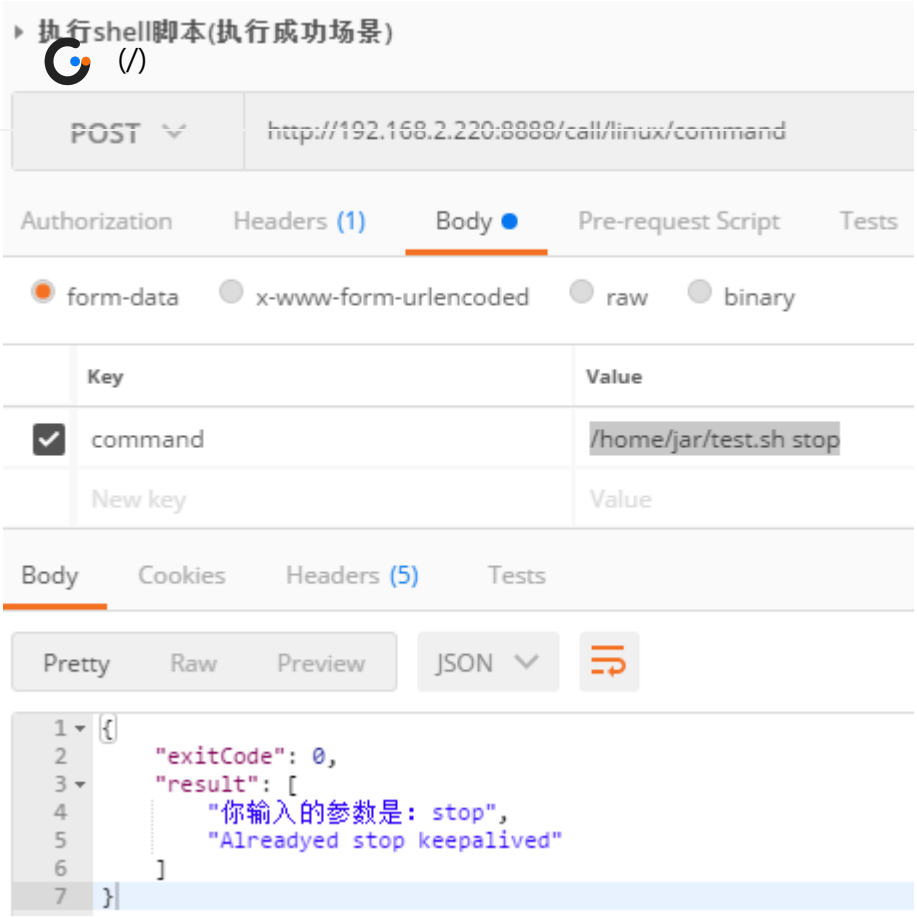


3. 执行 Shell 脚本（执行成功场景）

- url：http://192.168.2.220:8888/call/linux/command
- form-data 参数：
  - command： `/home/jar/test.sh stop`

返回结果：

```
{
    "exitCode": 0,
    "result": [
        "你输入的参数是：stop",
        "Alreadyed stop keepalived"
    ]
}
```

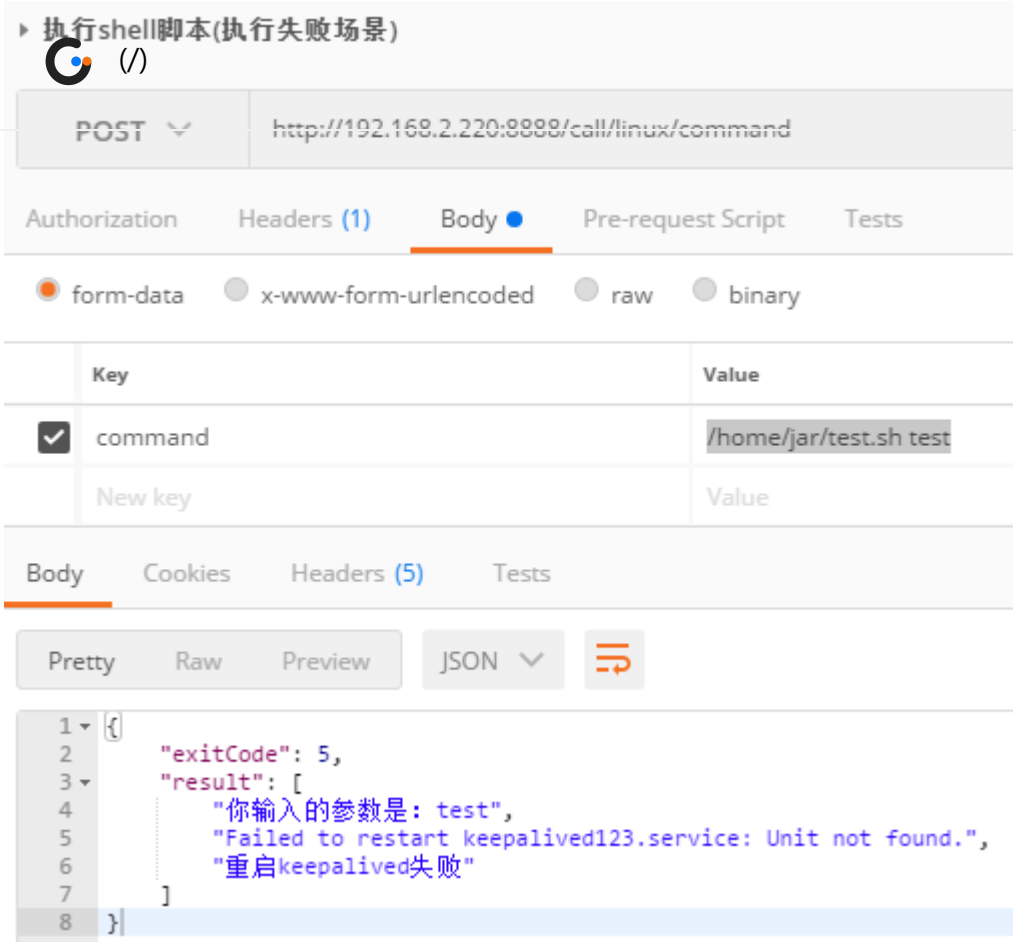


4. 执行 Shell 脚本（执行失败场景）

- url：http://192.168.2.220:8888/call/linux/command
- form-data 参数：
  - command： `/home/jar/test.sh test`

返回结果：

```
{
    "exitCode": 5,
    "result": [
        "你输入的参数是：test",
        "Failed to restart keepalived123.service: Unit not found.",
        "重启 keepalived 失败"
    ]
}
```



5. 执行 cat 命令

- url：http://192.168.2.220:8888/call/linux/command
- form-data 参数：
  - command： `cat /home/jar/test.sh`

返回结果（`\t` 相当于按一个 Tab 键）：

(5)


```
    "exitCode": 0,
    "result": [
        "#!/bin/bash",
        "",
        "operate=\"$1\"",
        "",
        "if [ -z $operate ]; then",
        "        echo \"请输入参数，start 代表开启 keepalived，stop 代表
        "        exit 8",
        "fi",
        "",
        "echo \"你输入的参数是：\"$operate",
        "",
        "if [ $operate = \"start\" ] ; then",
        "\tsystemctl start keepalived",
        "\trc=$?",
        "\tif  [ ${rc} -ne 0 ]; then",
        "\t\techo \"启动 keepalived 失败\"",
        "\t\texit ${rc}",
        "\tfi",
        "\techo \"Alreadyed start keepalived\"",
        "elif [ $operate = \"stop\" ] ; then  ",
        "\tsystemctl stop keepalived",
        "\trc=$?",
        "\tif  [ ${rc} -ne 0 ]; then",
        "\t\techo \"停止 keepalived 失败\"",
        "\t\texit ${rc}",
        "\tfi",
        "\techo \"Alreadyed stop keepalived\"",
        "elif [ $operate = \"restart\" ] ; then",
        "\tsystemctl restart keepalived",
        "\trc=$?",
        "\tif  [ ${rc} -ne 0 ]; then",
        "\t\techo \"重启 keepalived 失败\"",
        "\t\texit ${rc}",
        "\tfi",
        "\techo \"Alreadyed restart keepalived\"",
        "elif [ $operate = \"test\" ] ; then",
        "\tsystemctl restart keepalived123",
        "\trc=$?",
        "\tif  [ ${rc} -ne 0 ]; then",
        "\t\techo \"重启 keepalived 失败\"",
        "\t\texit ${rc}",
        "\tfi",
        "\techo \"Alreadyed restart keepalived\"",
```

```
            "else",
            "\techo \"参数格式有误，请重新输入，start 代表开启 keepalived，sto
            "fi"
        ]
    }
```

▸ 执行cat命令

(/)

POST http://192.168.2.220:8888/call/linux/command

Authorization Headers Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary

Key	Value
command	cat /home/jar/test.sh
New key	Value

Body Cookies Headers (5) Tests

Pretty Raw Preview JSON


```
{
    "exitCode": 0,
    "result": [
        "#!/bin/bash",
        "",
        "operate=\"$1\"",
        "",
        "if [ -z $operate ]; then",
        "        echo \"请输入参数，start代表开启keepalived，stop代表停止，restart代表重启\"",
        "        exit 8",
        "fi",
        "",
        "echo \"你输入的参数是：\"$operate",
        "",
        "if [ $operate = \"start\" ] ; then",
        "\tsystemctl start keepalived",
        "\trc=$?",
        "\tif  [ ${rc} -ne 0 ]; then",
        "\t\techo \"启动keepalived失败\"",
        "\t\texit ${rc}",
        "\tfi",
        "\techo \"Alreadyed start keepalived\"",
        "elif [ $operate = \"stop\" ] ; then  ",
        "\tsystemctl stop keepalived",
        "\trc=$?",
        "\tif  [ ${rc} -ne 0 ]; then",
        "\t\techo \"停止keepalived失败\"",
        "\t\texit ${rc}",
        "\tfi",
        "\techo \"Alreadyed stop keepalived\"",
        "elif [ $operate = \"restart\" ] ; then",
        "\tsystemctl restart keepalived",
        "\trc=$?",
        "\tif  [ ${rc} -ne 0 ]; then",
        "\t\techo \"重启keepalived失败\"",
        "\t\texit ${rc}",
        "\tfi",
        "\techo \"Alreadyed restart keepalived\"",
        "elif [ $operate = \"test\" ] ; then",
```


截图(Alt + A)

注意事项

- Linux 必须安装 JDK，否则使用不了 `java -jar` 命令，意味着启动不了工程。
- 本场 Chat 是借用操作 keepalived 场景而展开编写，所以需要安装 keepalived 才能使用相关命令。
- 上传 test.sh 脚本后需要授予执行权限，命令如下： `chmod +x test.sh` 。
- test.sh 脚本中的 operate 等于 test 条件是模拟执行脚本出错场景。
- 参数需要格式化，parseCommand 方法使用了正则表达式对参数进行格式，举个例子：参数 `/home/jar/test.sh start` 如果不通过正则格式化，会被以空格分割拆分 2 个命令执行显然有误。

总结

通过这次的 Java 调用 Linux 命令实战，让我们掌握了如何通过 Java 接口执行 shell 脚本，如何操作 keepalived，操作其他软件也是同样的做法，执行具体命令判断返回结果，还得到了一套可运行的 Java 调用 Linux 命令代码及使用步骤，可以快速运用到日常开发。

PS：如有写错请指正，感谢您阅读。