

## 九、Jenkins构建java项目（分布式）

1.准备java依赖镜像

2.新建item/新建任务

3.构建一个maven项目

4.配置item/任务

5.确认配置文件是否正确

6.构建任务测试

7.快速构建其他任务

## 九、Jenkins构建java项目（分布式）

### 前言：

先把构建思路说一下，不然闷头看步骤有些晦涩。

1】微服务+前后端分离项目，要启动多个“**服务**”，包括**jar包服务**和一个**前端服务**。

每个服务都要打包一个镜像，然后生成容器去运行。

2】打包镜像，就需要提前准备打包镜像的依赖镜像。

3】jenkins创建的item/任务，只纳管一个服务，所有需要创建多个item/任务。

### 1.准备java依赖镜像

1) 查看可用的 jdk/jre 版本

访问 docker 镜像库地址：<https://hub.docker.com/>，搜索jdk或者jre，挑选自己需要的版本

2) 拉取镜像

```
1 docker pull kdvolder/jdk8:latest #下载最新版本
```

3) 查看本地镜像

```
1 docker images
```

## 2.新建item/新建任务



## 3.构建一个maven项目

构建微服务的gateway网关服务为例，起名manstro-gateway



如果没有“构建一个maven项目”选项，是因为缺少插件：Maven Intergration

【系统管理】==>【管理插件】==>搜索Maven Integration插件==>点击 install without restart



## 4.配置item/任务

### 1】Discard old builds/丢弃旧的构建

提供了“Discard old builds”的选项可以配置保留几天/最多几个的build 和 artifacts，缺省这个选项不会勾上。不勾选时，默认值为365，显然这个值过大了。

☒ 丢弃旧的构建 ?

策略

Log Rotation

保持构建的天数

如果非空，构建记录将保存此天数

7

保持构建的最大个数

如果非空，最多此数目的构建记录将被保存

5

发布包保留天数

如果非空，比此早的发布包将被删除，但构建的日志、操作历史、报告等将被保留

7

发布包最大保留#个构建

如果非空，最多此数目大构建将保留他们的发布包

5

## 2】参数化构建过程

☒ 参数化构建过程 ?

添加参数 ▲

布尔值参数

选项参数

凭据参数

文件参数

Git 参数

List Subversion tags (and more)

文本参数

密码参数

运行时参数

字符参数

如果没有 “Git 参数”，需要安装Git Parameter插件

【系统管理】==>【管理插件】==>搜索Git Parameter插件==>点击 install without restart

### 1) 添加Git参数: branch (必配)

名称: branch

描述: git分支

参数类型: 分支或标签

默认值: \*/master

### 2) 添加字符参数: host\_port (必配)

名称: host\_port

默认值: 8080

描述: 服务端口 (宿主机暴露的端口, 可以和服务配置的端口保持一致)

### 3) 添加字符参数: docker\_ip (必配)

名称: docker\_ip

默认值: 192.168.16.5

描述: 服务IP (宿主机IP)

### 4) 添加选项参数: profile\_flag (必配)

名称: profile\_flag

选项:

dev

prod

test

描述: 运行环境

### 5) 添加字符参数: images\_name

名称: images\_name

默认值: manstro-gateway

描述: 镜像名称

### 6) 添加字符参数: container\_name

名称: container\_name

默认值: manstro-gateway

**描述：** 容器名称

#### **7) 添加字符参数: jar\_name**

**名称:** jar\_name

**默认值:** manstro-gateway

**描述:** jar包名称

**注意:** 这里的jar包名不是自定义的, 打包出来是啥jar包可以先试试。

**例如**system打包出来就叫manstro-modules-system.jar。

#### **8) 添加字符参数: ssh\_jar\_file**

**名称:** ssh\_jar\_file

**默认值:** /docker/project/manstro/gateway

**描述:** jar包文件路径

#### **9) 添加字符参数: ssh\_logs\_file**

**名称:** ssh\_logs\_file

**默认值:** /docker/project/manstro/logs

**描述:** 服务日志文件路径

### **3】 源码管理**

<http://10.0.0.251/research/manstro-base-framework/manstro-application.git>

\$branch

源码管理

☐ 无

☒ Git ?

Repositories ?

Repository URL ?

Credentials ?

Branches to build ?

指定分支 (为空时代表any) ?

刚才定义的git参数

## 4] Pre Steps

拉取代码后优先执行此配置

1) 选择 Invoke top-level Maven targets (调用顶层Maven目标)

Invoke Ant

Invoke Gradle script

Run with timeout

Send files or execute commands over SSH

Set build status to "pending" on GitHub commit

执行 Windows 批处理命令

执行 shell

2) 选择maven版本, 填写指令: clean install

调用顶层 Maven 目标 ?

Maven 版本

目标

## 5] Build

打包命令执行完成后根据build配置所在目录的pom文件进行docker镜像构建

**Root POM:** manstro-gateway/pom.xml

**Goals and options:** clean package -am -P \$profile\_flag -Dmaven.test.skip=true -Dfile.encoding=UTF-8

备注:

-am: 构建指定模块, 同时构建指定模块依赖的其他模块

-P: 激活指定的profile文件列表(用逗号[,]隔开)

-D:定义系统变量

## 6] Post Steps

选择: Run only if build succeeds or is unstable

### 1) Send files or execute commands over SSH (上传文件到远程服务器)

点击: Add post-build step (增加构建步骤) 按钮

选择: Send files or execute commands over SSH

## Post Steps

- ☐ Run only if build succeeds
- ☒ Run only if build succeeds or is unstable
- ☐ Run regardless of build result

Should the post-build steps run only for successful builds, etc.

Add post-build step ^

Invoke Ant  
Invoke Gradle script  
Run with timeout  
**Send files or execute commands over SSH**  
Set build status to "pending" on GitHub commit  
执行 Windows 批处理命令  
执行 shell  
调用顶层 Maven 目标

SSH Server: 选择之前配置ssh地址

Source files (准备上传的源文件, =本地打包后的jar包路径): manstro-gateway/target/\*.jar

Remove prefix (删除前缀): manstro-gateway/target/

Remote directory (远程目录, 和上面配置的ssh\_jar\_file保持一致): /docker/project/manstro/gateway

Exec command (Exec命令): /



SSH Server

Name ?

192.168.16.5

高级...

Transfers

Transfer Set

Source files ?

manstro-gateway/target/\*.jar

Files to upload to a server.

The string is a comma separated list of includes for an Ant fileset eg. '\*\*/\*.jar' (see [Patterns](#) in the Ant manual).

The base directory for this fileset is the workspace.

( [Publish Over SSH](#) )

Remove prefix ?

manstro-gateway/target/

Remote directory ?

/docker/project/manstro/gateway

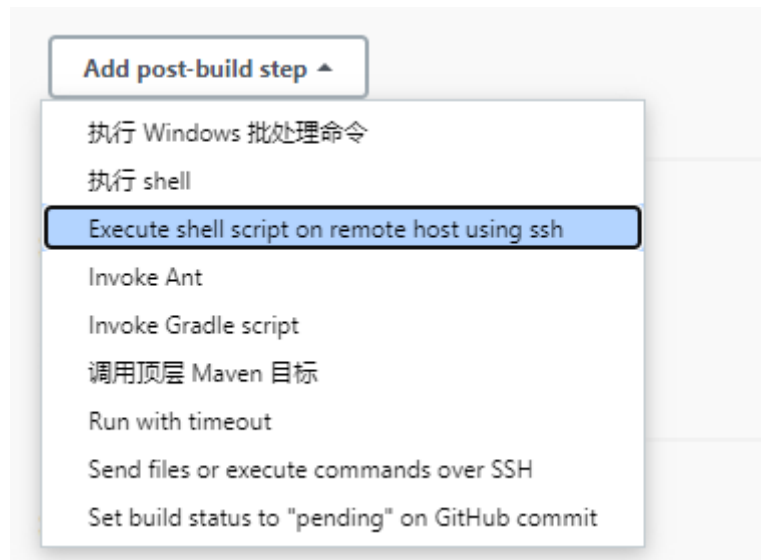
Exec command ?

/

## 2) Execute shell script on remote host using ssh (在远程服务器上执行脚本)

点击：Add post-build step (增加构建步骤) 按钮

选择：Execute shell script on remote host using ssh



如果没有这个选项，请参考第三部分jenkins配置，下载ssh插件并在系统配置中配置相关内容。

Commad输入：

```
1 echo '=====1.判断容器是否运行====='
```

```
2 if [[ -n $(docker ps -a | grep $container_name) ]];then
```

```
3  echo '=====1.1容器正在运行====='
```

```
4  echo '=====1.2停止容器====='
```

```
5  docker stop $container_name
```

```
6  echo '=====1.3强制删除容器====='
```

```
7  docker rm -f $container_name
```

```
8  echo '=====1.4强制删除镜像====='
```

```
9  docker rmi -f $images_name
```

```
10 elseE
```

```
11  echo '=====2.1容器未运行====='
```

```
12  fi
```

```
13
```

```
14  echo '=====2.进入jar包文件夹====='
```

```
15  if [ ! -d "$ssh_jar_file" ]; then
```

```
16    mkdir -p $ssh_jar_file
```

```
17  fi
```

```
18  cd $ssh_jar_file
```

```
19  echo '=====3.重新写入配置====='
```

```
20
```

```
21  echo "FROM kdvolder/jdk8" > Dockerfile
```

```
22  echo '# 指定字符集'
```

```
23  echo -e "ENV LC_ALL \"zh_CN.UTF-8\"" >> Dockerfile
```

```
24  echo '# 设置时区'
```

```
25  echo -e "ENV TZ \"Asia/Shanghai\"" >> Dockerfile
```

```
26  echo -e "RUN ln -snf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime && echo 'Asia/Shanghai' >/etc/timezone" >> Dockerfile
```

```
27  echo '# 将本地文件挂载到当前容器'
```

```
28  echo "VOLUME /tmp" >> Dockerfile
```

```
29  echo '# 复制文件到容器'
```

```
30  echo "ADD ./ $jar_name.jar /usr/local/$jar_name.jar" >> Dockerfile
```

```
31  echo '# 声明暴露的端口号'
```

```
32  echo "EXPOSE $host_port">> Dockerfile
```

```
33  echo '# 配置容器启动后执行的命令'
```

```
34  echo -e "ENTRYPOINT [\"java\", \"-jar\", \"-Xmx512m\", \"-Dfile.encoding=utf-8\", \"/usr/local/$jar_name.jar\"]" >> Dockerfile
```

```
35
```

```
36  echo '=====4.打包镜像====='
```

```
37  docker build -t $images_name .
```

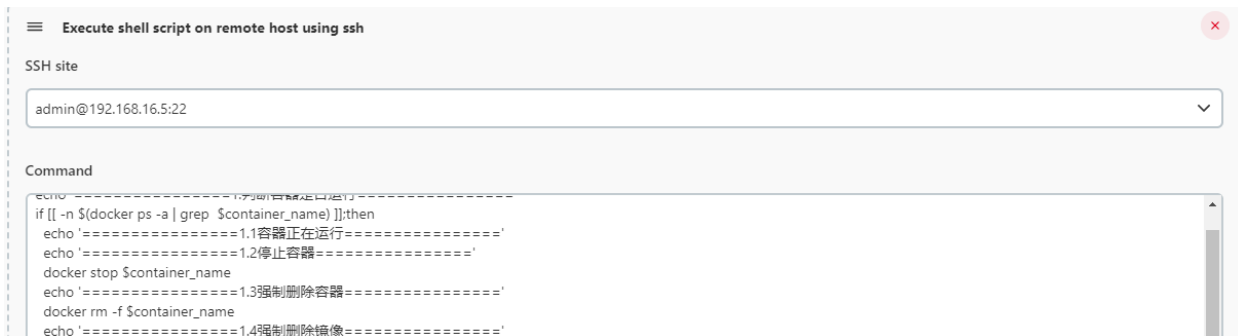
```
38  echo '=====5.启动容器====='
```

```
39  if [ ! -d "$ssh_logs_file" ]; then
```

```
40    mkdir -p $ssh_logs_file
```

```
41  fi
```

```
42 docker run --restart=always -p $host_port:$host_port -e docker_ip="$docker_ip" -e host_port="$host_port" --name $container_name -v $ssh_logs_file:/logs -d $images_name:latest
```



## 5.确认配置文件是否正确

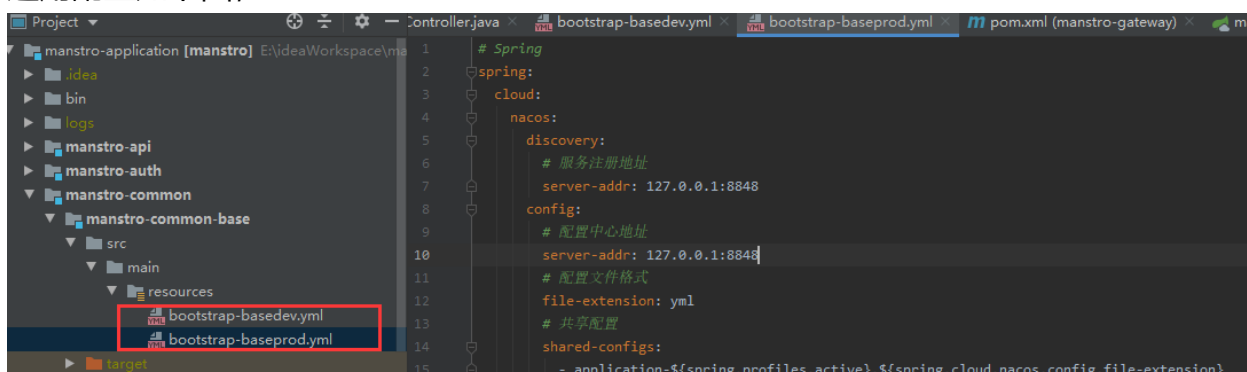
### 1) nacos配置

确认nacos地址、端口，是否与步骤6的nacos地址一致。

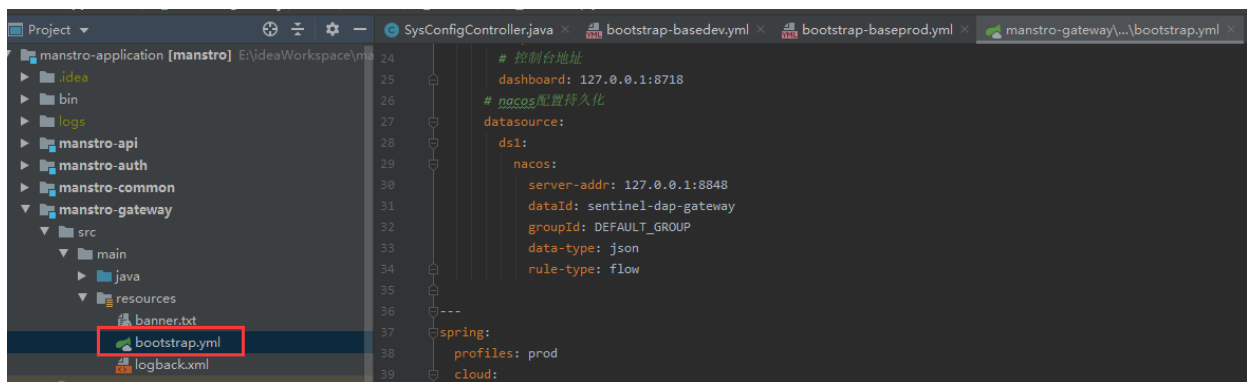
就本基线框架而言，需要确认通用配置和模块配置。

如果配置文件的nacos地址不对，后面附解决办法

通用配置如下图，



模块配置如下图



### 2) mysql和redis配置

确认mysql和redis的配置正确。登录第6步创建的nacos，确保每个配置文件配置都正确。

← → 不安全 | 192.168.16.5:8848/nacos/#/configurationManagement?serverId=center&group=&dataId=&namespace=&pageSize=&pageNo=&appName=

应用 百度翻译 百度 新标签页

**NACOS** 首页 文档

**NACOS 1.4.0** | 配置管理 | 查询结果: 共查询到 11 条满足要求的配置。

配置管理 ^

Data ID:  Group:  [查询](#) [高级查询](#) [导出查询结果](#) [导入配置](#)

<input type="checkbox"/>	Data Id	Group	归属应用:	操作
<input type="checkbox"/>	application-dev.yml	DEFAULT_GROUP		<a href="#">详情</a>   <a href="#">示例代码</a>   <a href="#">编辑</a>   <a href="#">删除</a>   <a href="#">更多</a>
<input type="checkbox"/>	dap-gateway-dev.yml	DEFAULT_GROUP		<a href="#">详情</a>   <a href="#">示例代码</a>   <a href="#">编辑</a>   <a href="#">删除</a>   <a href="#">更多</a>
<input type="checkbox"/>	dap-auth-dev.yml	DEFAULT_GROUP		<a href="#">详情</a>   <a href="#">示例代码</a>   <a href="#">编辑</a>   <a href="#">删除</a>   <a href="#">更多</a>
<input type="checkbox"/>	dap-monitor-dev.yml	DEFAULT_GROUP		<a href="#">详情</a>   <a href="#">示例代码</a>   <a href="#">编辑</a>   <a href="#">删除</a>   <a href="#">更多</a>
<input type="checkbox"/>	dap-system-dev.yml	DEFAULT_GROUP		<a href="#">详情</a>   <a href="#">示例代码</a>   <a href="#">编辑</a>   <a href="#">删除</a>   <a href="#">更多</a>
<input type="checkbox"/>	dap-gen-dev.yml	DEFAULT_GROUP		<a href="#">详情</a>   <a href="#">示例代码</a>   <a href="#">编辑</a>   <a href="#">删除</a>   <a href="#">更多</a>
<input type="checkbox"/>	dap-job-dev.yml	DEFAULT_GROUP		<a href="#">详情</a>   <a href="#">示例代码</a>   <a href="#">编辑</a>   <a href="#">删除</a>   <a href="#">更多</a>
<input type="checkbox"/>	dap-file-dev.yml	DEFAULT_GROUP		<a href="#">详情</a>   <a href="#">示例代码</a>   <a href="#">编辑</a>   <a href="#">删除</a>   <a href="#">更多</a>
<input type="checkbox"/>	sentinel-dap-gateway	DEFAULT_GROUP		<a href="#">详情</a>   <a href="#">示例代码</a>   <a href="#">编辑</a>   <a href="#">删除</a>   <a href="#">更多</a>
<input type="checkbox"/>	dap-activiti-dev.yml	DEFAULT_GROUP		<a href="#">详情</a>   <a href="#">示例代码</a>   <a href="#">编辑</a>   <a href="#">删除</a>   <a href="#">更多</a>

[删除](#) [导出选中的配置](#) [克隆](#)

每页显示: 10

**问题：git远端配置文件的nacos地址不对，但是基线代码不能随意修改提交怎么办？**

**解决办法：通过插件在构建项目之前替换配置文件**

**解决办法第一步：下载Config File Provider插件**

【系统管理】==>【管理插件】==>【可选插件】==>搜索**Config File Provider**插件  
==>点击 install without restart

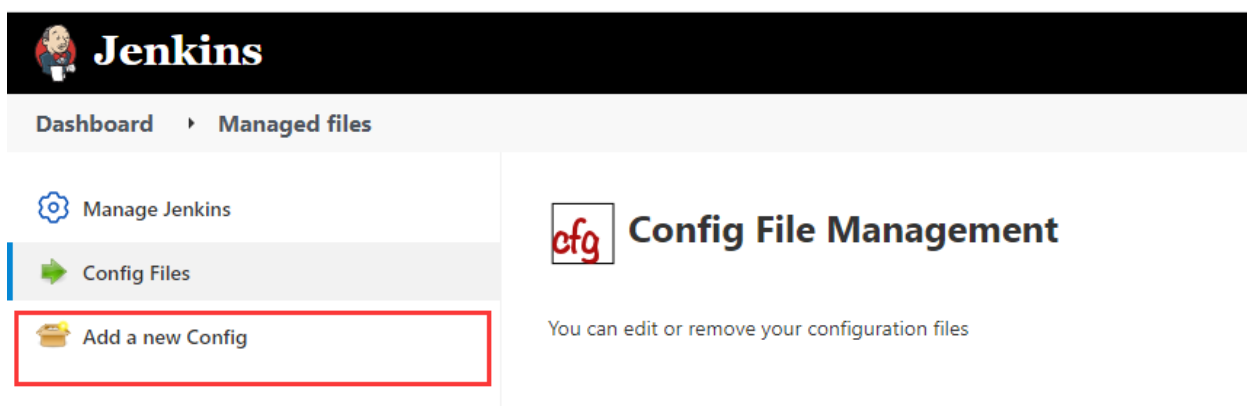
这里还是需要重启jenkins的，看安装提示。

安装成功后，系统配置这里有这个图标



**解决办法第二步：创建新的配置文件管理**

点击Add a new Config



选择Custom file ==> next

☐ Groovy file  
a reusable groovy script

☒ Custom file  
a custom file (e.g. text or any other not yet available format)

☐ Extended Email Publisher Groovy Template  
A Groovy template used by the Extended Email Publisher plugin to generate emails.

☐ Extended Email Publisher Jelly Template  
A Jelly template used by the Extended Email Publisher plugin to generate emails.

ID  
ID of the config file

7a7f1d5d-866c-44b4-88a2-f77af82342af

Next

填写文件名和文件内容

填写配置文件的名字（这个时候的名字可以跟实际应用时的配置文件名不一致，jenkins调用配置文件覆盖源代码的时候，还会进行更名），配置文件的内容，提交保存

The configuration

ID  
7a7f1d5d-866c-44b4-88a2-f77af82342af

Name  
MyCustom 这里填写文件名

Comment

Tokenized Credentials  
新增

Content

1 这里填写文件内容

Submit

示例1：

Name: manstro-common-bootstrap-basedev.xml

Content:

```
1 # Spring
2 spring:
3   cloud:
4     nacos:
5       discovery:
6     # 服务注册地址
7     server-addr: 192.168.16.5:8848
8     # 命名空间 若使用的命名空间为默认的public 则不需要配置
9     # namespace: public
```

```

10  #群组
11  group: DEFAULT_GROUP
12  config:
13  # 配置中心地址
14  server-addr: 192.168.16.5:8848
15  # 配置文件格式
16  file-extension: yml
17  # 命名空间 若使用的命名空间为默认的public 则不需要配置
18  # namespace: public
19  #群组
20  group: DEFAULT_GROUP
21  # 共享配置
22  shared-configs[0]:
23  data-id: application-${spring.profiles.active}.${spring.cloud.nacos.con
fig.file-extension}
24  group: DEFAULT_GROUP
25  refresh: false
26  project:
27  # 项目名称
28  name: manstro

```

示例2:

Name: [manstro-gateway-bootstrap.yml](#)

Content:

```

1  # Tomcat
2  server:
3  port: 8080
4
5  # Spring
6  spring:
7  application:
8  # 应用名称
9  name: dap-gateway
10  profiles:
11  # 环境配置
12  active: @environment@
13  include: base@environment@
14  main:
15  allow-bean-definition-overriding: true
16  ---

```

```
17 spring:
18   profiles: dev
19   cloud:
20     sentinel:
21     # 取消控制台懒加载
22     eager: true
23     transport:
24     # 控制台地址
25     dashboard: 192.168.16.5:8718
26     # nacos配置持久化
27     datasource:
28     ds1:
29     nacos:
30     server-addr: 192.168.16.5:8848
31     dataId: sentinel-dap-gateway
32     groupId: DEFAULT_GROUP
33     data-type: json
34     rule-type: flow
35
36 ---
37 spring:
38   profiles: prod
39   cloud:
40     sentinel:
41     # 取消控制台懒加载
42     eager: true
43     transport:
44     # 控制台地址
45     dashboard: 192.168.16.5:8718
46     # nacos配置持久化
47     datasource:
48     ds1:
49     nacos:
50     server-addr: 192.168.16.5:8848
51     dataId: sentinel-dap-gateway
52     groupId: DEFAULT_GROUP
53     data-type: json
54     rule-type: flow
```

### 解决办法第三步：构建环境

保存成功后，在项目构建配置里构建环境那一栏，勾选provide Configuration files

The screenshot shows a configuration interface with several tabs: General, 源码管理, 构建触发器, 构建环境 (selected), Pre Steps, Build, Post Steps, 构建设置, and 构建后. Below the tabs, there is a section titled '构建环境' (Build Environment) containing a list of checkboxes. The checkbox for 'Provide Configuration files' is highlighted with a red rectangular box. Other checkboxes include 'Delete workspace before build starts', 'Use secret text(s) or file(s)', 'Send files or execute commands over SSH before the build starts', 'Send files or execute commands over SSH after the build runs', 'Abort the build if it's stuck', 'Add timestamps to the Console Output', 'Execute shell script on remote host using ssh', 'Inspect build log for published Gradle build scans', and 'With Ant'.

General 源码管理 构建触发器 **构建环境** Pre Steps Build Post Steps 构建设置 构建后

☐ 轮询 SCM ?

**构建环境**

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s) ?
- ☐ **Provide Configuration files ?**
- ☐ Send files or execute commands over SSH before the build starts ?
- ☐ Send files or execute commands over SSH after the build runs ?
- ☐ Abort the build if it's stuck
- ☐ Add timestamps to the Console Output
- ☐ Execute shell script on remote host using ssh ?
- ☐ Inspect build log for published Gradle build scans
- ☐ With Ant ?

替换文件：

File：选择新创建的配置文件

Target：需要替换的 文件目录+文件名+后缀

以下是我需要替换的两个文件路径：

```
1 manstro-common/manstro-common-base/src/main/resources/bootstrap-basedev.yml
2 manstro-gateway/src/main/resources/bootstrap.yml
```



Managed Files

File ?

manstro-common-bootstrap-basedev.xml

选择要替换的文件

[view selected file](#)

Target ?

manstro-common/manstro-common-base/src/main/resources/bootstrap-basedev.yml

被替换文件的路径+文件名+后缀

Variable ?

☐ Replace Tokens

Add file

## 6.构建任务测试

点击按钮构建试试，看看能否顺利

S	W	名称	上次成功	上次失败	上次持续时间	
		manstro-gateway	没有	无	无	

图标 小 中 大

图例

Atom feed 全部

Atom feed 失败

Atom feed 最新的构建

如果构建成功，可以查看容器运行日志

```
1 docker logs -f manstro-gateway
```

也可以从nacos验证下是否注册成功

← → ↻ ⚠ 不安全 | 192.168.16.5:8848/nacos/#/serviceManagement?serverId=center&group=&dataId=&namespace=&pageSize=&pageNo=&appNa

百度翻译 百度 新标签页

NACOS

NACOS 1.4.0

公共空间

服务列表 | 公共空间

配置管理

服务管理

服务列表

订阅者列表

权限控制

命名空间

集群管理

服务名称

请输入服务名称

分组名称

请输入分组名称

隐藏空服务:

☒

查询

服务名	分组名称	集群数目	实例数	健
dap-gateway	DEFAULT_GROUP	1	1	1

## 7.快速构建其他任务

当构建完成好一个项目，可以快速复制简单修改重要参数就能创建一个新的任务。

## 输入一个任务名称

manstro-system

» 任务名 'manstro-system' 已存在



### 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.



### 构建一个maven项目

构建一个maven项目.Jenkins利用你的POM文件.这样可以大大减轻构建配置.



### 构建一个多配置项目

适用于多配置项目.例如多环境测试.平台指定构建.等等.



### 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内容，5

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

manstro-gateway

确定