项目部署流程

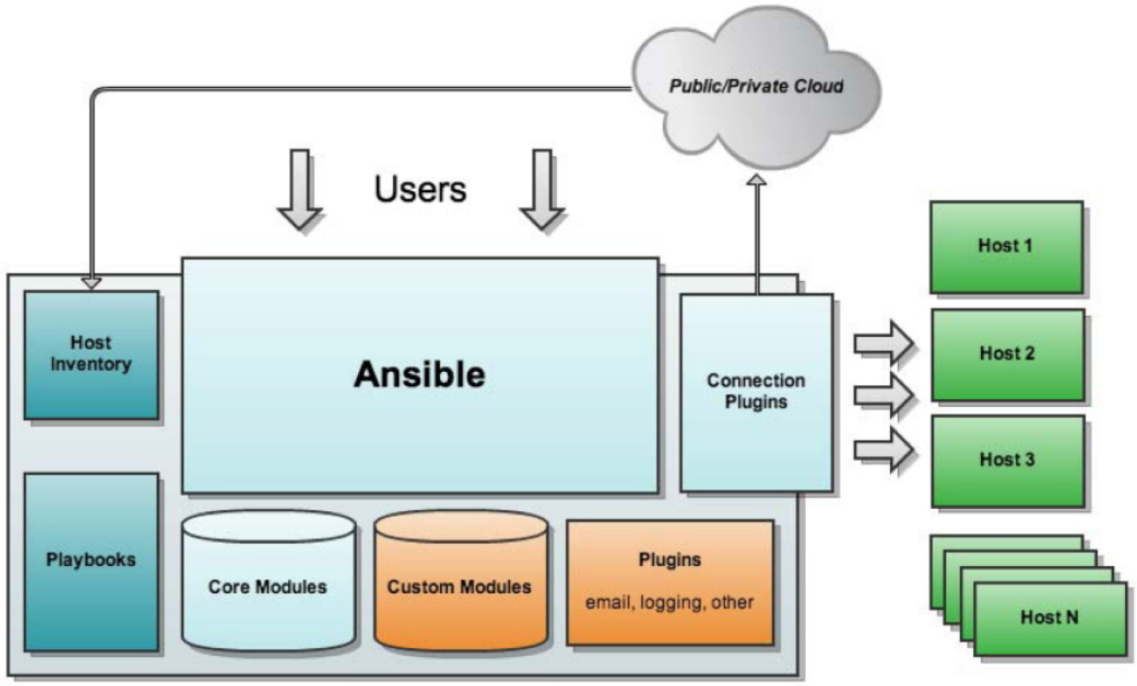
1、简单了解Ansible

Ansible就是一款自动化工具，通过一个命令完成一系列的操作，进而能减少重复性的工作和维护成本！

Ansible自2012年发布以来，很快在全球流行，其特点如下：

- Ansible基于Python开发，运维工程师对其二次开发相对比较容易；
- Ansible丰富的内置模块，几乎可以满足一切要求；
- 管理模式非常简单，一条命令可以影响上千台主机；
- 无客户端模式，底层通过SSH通信。

Ansible基本架构



Ansible工具集

- Inventory：Ansible管理主机清单；
- Ansible Playbooks：任务脚本，编排定义Ansible任务集的配置文件，由Ansible按序依次执行，通常是JSON格式的YML文件；（**这个是我们平常最常用的**）
- Modules：Ansible执行命令功能模块，多数为内置的核心模块，也可自定义；
- Plugins：模块功能的补充，如连接类型插件、循环插件、变量插件、过滤插件等，该功能不太常用；
- API：供第三方程序调用的应用程序编程接口；
- Ansible：该部分图中表现得不太明显，组合Inventory、API、Modules、Plugins可以理解为是Ansible命令工具，其为核心执行工具。

Inventory主机清单

> **关注点：主机与组、主机变量、组变量**

主机与组

格式：

[组名]

主机1

主机2

```
[java_cdp_product_app] -- 定义一个组
java_cdp_product_app_demo1 ansible_ssh_pass=123456 app_address=192.168.10.81 --
主机1
java_cdp_product_app_demo2 ansible_ssh_pass=123456 app_address=192.168.10.82 --
主机2
```

方括号[]中是组名,用于对系统进行分类,便于对不同系统进行个别的管理。

一个主机可以属于不同的组。一个组可以作为另外一个组的成员。

主机变量

格式：主机 var1 var2

```
[java_cdp_product_app] -- 定义一个组
java_cdp_product_app_demo1 ansible_ssh_pass=123456 app_address=192.168.10.81 --
主机1
java_cdp_product_app_demo2 ansible_ssh_pass=123456 app_address=192.168.10.82 --
主机2

主机变量：ansible_ssh_pass（系统参数）、app_address（自定义）
```

组变量

格式：[组名:vars]

```
[java_cdp_product_app:vars]----下面这些都是自定义的参数
app_port=5004
app_version="develop"
app_migrate_server=java_cdp_product_app_demo1
app_database_host=db.test.yipicha.com
app_database=product_dev
app_database_user=java_cbs_dev
app_database_password=java_cbs_dev
app_redis_host=db.test.yipicha.com
app_redis_port=6379
app_redis_db=5
app_redis_password=
```

Inventory 参数的说明

参数名	参数说明
ansible_ssh_host	将要连接的远程主机名.与你想要设定的主机的别名不同的话,可通过此变量设置
ansible_ssh_port	ssh端口号.如果不是默认的端口号,通过此变量设置，默认端口22
ansible_ssh_user	默认的 ssh 用户名
ansible_ssh_pass	ssh 密码(这种方式并不安全,我们强烈建议使用 --ask-pass 或 SSH 密钥)
ansible_sudo	定义主机的sudo用户
ansible_sudo_pass	sudo 密码(这种方式并不安全,我们强烈建议使用 --ask-sudo-pass)
ansible_sudo_exe	sudo 命令路径(适用于1.8及以上版本)
ansible_connection	与主机的连接类型
ansible_ssh_private_key_file	ssh 使用的私钥文件.适用于有多个密钥,而你不想使用 SSH 代理的情况.
ansible_shell_type	目标系统的shell类型.默认情况下,命令的执行使用 'sh' 语法,可设置为 'csh' 或 'fish'.
ansible_python_interpreter	定义主机python解释器路径

Playbooks

Playbooks 是 Ansible的配置,部署,编排语言.他们可以被描述为一个需要希望远程主机执行命令的方案,或者一组IT程序运行的命令集合。也称为“剧本”，模块化定义一系列任务，供外部统一调用，使用yaml格式进行定义。

核心元素

- Tasks：任务，由模块定义的操作的列表；
- Variables：变量
- Templates：模板，即使用了模板语法的文本文件；
- Handlers：由特定条件触发的Tasks；
- Roles：角色；

Playbook的基础组件：

- Hosts：运行指定任务的目标主机；
- remote_user：在远程主机以哪个用户身份执行；
- sudo_user：非管理员需要拥有sudo权限；

格式：

```
---
- hosts: 组名
  vars: #自定义参数
  tasks: #执行任务步骤
    - name: #指定该任务的名称1

    - name: #指定该任务的名称2
  handlers: #处理器
```

官方文档：

```
---                                 #标记文件的开始
- hosts: webservers                 #指定该playbook在哪个服务器上执行
  vars:                             #表示下面是定义的变量，
    http_port: 80                   #变量的形式，key: value，这里http_port是变量名，80是
值
    max_clients: 200
  remote_user: root                 #指定远程的用户名，这里缩进和vars保持了一致，说明变量的代
码块已经结束。
  tasks:                            #下面构成playbook的tasks，每个task都有 - name: 开
始，name指定该任务的名称。
  - name: ensure apache is at the latest version  #指定该任务的名称。
    yum: pkg=httpd state=latest                   #yum说明要是用的模板名称，后面指定
对应的参数，这两行结合起来就相当于一个shell命令。

  - name: write the apache config file            #每个task之间可以使用空行来做区分。
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
```

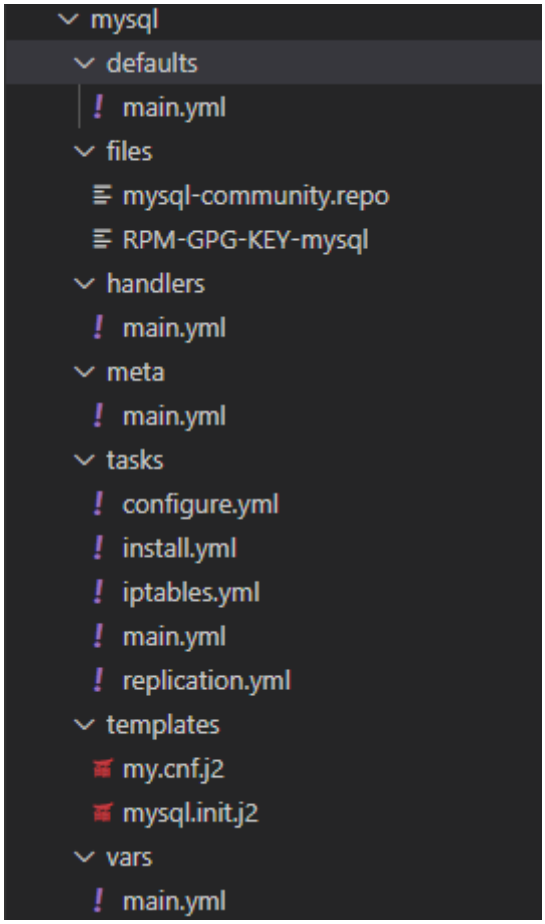
茶城项目定义脚本：

```
- hosts: java_cbs_app
  any_errors_fatal: true
  vars:
    app_repo: ssh://git@git.oteao.com:22222/java/cbs.git
    app_name: java_cbs
    app_target: "{{ app_current_path }}/cbs/target/cbs.jar"
    app_shared_configs:
      - {src: 'application.yml', dest: 'cbs/src/main/resources/application.yml'}
      - {src: 'logback-spring.xml', dest: 'cbs/src/main/resources/logback-
spring.xml'}
      - {src: 'migrate.sh', dest: 'migrate.sh'}
  roles: #角色
    - { role: app.java.ssm }
```

roles

可以看成是一台或者多台主机的某些事情，分开进行playbook编写，通过规范化的目录结构来重新整理复杂Playbooks。

每个角色的定义，以特定的层级目录结构进行组织。以（MySQL角色）为例：



- files：存放由copy或script等模块调用的文件；
- templates：存放template模块查找所需要的模板文件的目录，如MySQL配置文件模板；
- tasks：任务存放的目录；
- handlers：存放相关触发执行的目录；
- vars：变量存放的目录；
- meta：用于存放此角色元数据；
- default：默认变量存放的目录，文件中定义了此角色使用的默认变量。

上述目录中，tasks、handlers、vars、meta、default至少应该包含一个main.yml文件，该目录下也可以有其他.yml文件，但是需要在main.yml文件中用include指令将其他.yml文件包含进来。

变量使用

这个变量我们来说明ansible中变量（不包含role中的变量）用法

1、 **extra-vars来指定变量**

```
[root@test2 playbook]# ansible-playbook test.yml --extra-vars "test_var=test" -v
        #加上-v选项，会显示详细的信息

[root@test2 playbook]# cat test.yml
---
 - hosts: all
   remote_user: root
   gather_facts: no
   vars_files:                  #这里使用了vars_files来引入变量文件
     - vars.yml
   tasks:
     - name: test playbook variables
       command: echo {{ test_var }}
```

2、**inventory文件中的变量**（主机和组定义定义的变量）

```
{{  hostvars['主机名']['app_version']}}

{{hostvars[groups['组名'][0]]['app_version']}}
```

- groups：包含了所有hosts文件里的主机组的一个列表。
- group_names: 包含了当前主机所在的所有主机组名的一个列表。
- inventory_hostname: 通过hosts文件定义的主机名。（与ansible_home意义不同）
- inventory_hostname_short：变量inventory_hostname的第一部分。譬如inventory_hostname的值为books.ansible.com，那么inventory_hostname_short的值就是books。
- play_hosts: 将执行当前任务的所有主机
- hostvars：包含了所有主机列表

3、**注册变量**

一个注册变量通常会有以下4个属性：

- changed：任务是否对远程主机造成的变更。
- delta：任务运行所用的时间。
- stdout：正常的输出信息。
- stderr：错误信息。

```
[root@test2 playbook]# cat test.yml
---
 - hosts: all
   remote_user: root
   gather_facts: no
   tasks:
     - name: test the register variables
       shell: uptime
       register: results                                    #使用关键字register声明注册变量，上面uptime命令产生的结果，存入到results中。结果是字典形式。

     - name: print the register result
       debug: msg="{{ results.stdout }}"                    #使用debug模块，打印出上面命令的输出结果。
```

4、**高阶变量**

用python语言写的，因此也支持一种叫做列表的变量，形式如下：

```
[root@test2 playbook]# cat test.yml
---
 - hosts: all
   remote_user: root
   gather_facts: no
   vars:
      var_list:                 #注意形式，定义了var_list列表，取值方法和列表取值一样，不推荐使用jinja2的方法取值。
          - one
```

```
          - two
          - three
   tasks:
     - name: test the list variables
       shell: echo {{ var_list[0] }}            #取列表中的第一个字，也就是one
       register: results

     - name: print the register result
       debug: msg="{{ results.stdout }}"
```

Tags标签

ansible的标签功能可以给角色，文件，单独的任务甚至整个playbook打上标签，然后利用这些标签来指定要运行playbook中的个别任务，或不执行指定的任务，并且它的语法非常简单。

```
---
# 可以给整个playbook的所有任务打一个标签。
  - hosts: all
    tags: deploy
    roles:
     # 给角色打的标签将会应用与角色下所有的任务。
       - {role: tomcat, tags : ["tomcat", "app"]}        #一个对象添加多个tag的写法之一
    tasks:
       - name: Notify on completion
         local_action:
            module: osx_say
            msg: "{{inventory_hostname}} is finished"
            voice: Zarvox
         tags:                                           #一个对象添加多个tag写法之二
            - notifications
            - say
       - include: foo.yml
         tags:  foo
```

2、项目部署脚本结构

> git地址：ssh://git@git.oteao.com:22222/op/devops.git

目前git分支使用情况

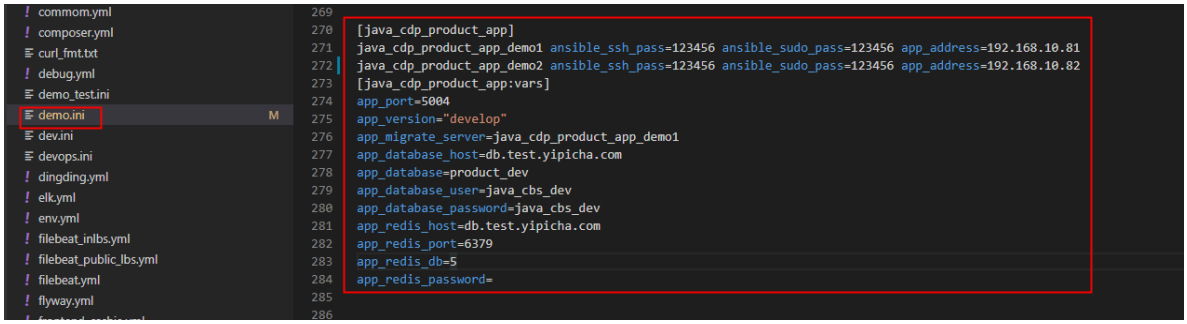
分支	使用环境
develop_v2	测试环境
develop_erp	ERP测试环境（陆续迁移，后面废弃）
develop_docker	线上docker
master_prd	线上环境

3、配置流程（以配置商品服务为例）

1、配置主机清单

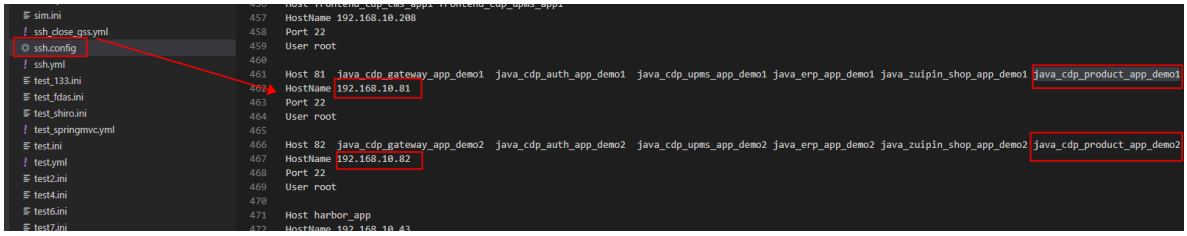
文件名	应用环境
prd.ini	线上环境
demo.ini	测试环境
test6.ini	茶城测试环境
mptest.ini	小程序
*.ini	除了prd.ini是线上，其他ini文件均为测试环境

找到对应需要配置的应用环境的主机清单文件，添加对应的组和主机、所需变量



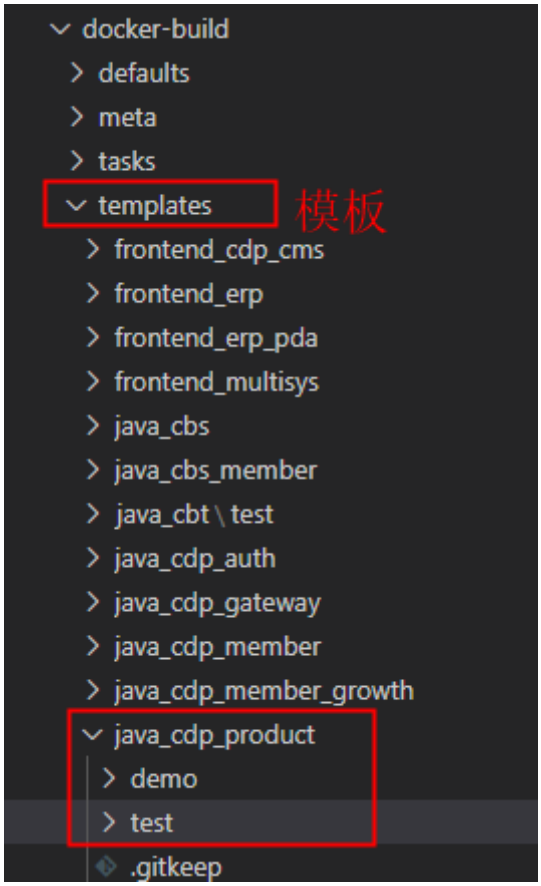
2、添加主机地址

找到ssh.config文件，找到要部署的主机地址，或者新增一台主机，User必须使用root用户，如图：

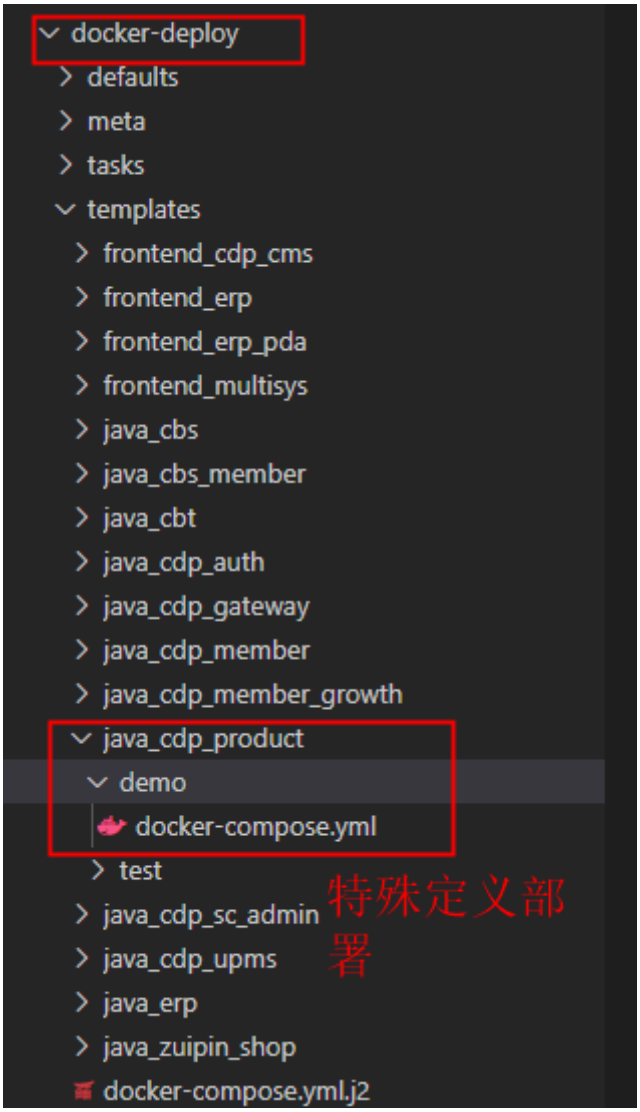


3、配置托管文件

在roles/app/templates、roles/docker-build/templates（docker编译）添加对应主机名，区分不同环境，将需要投管文件拷贝到templates中，如图:



如果配置docker-deploy角色，配置投管文件docker-compose.yml



4、添加playbooks脚本

创建java_cdp_product.yml文件

```
- hosts: harbor  #docker仓库主机
  any_errors_fatal: true
  vars:
    #配置git地址
    app_repo: ssh://git@git.oteao.com:22222/java/product-service.git
    app_name: java_cdp_product
    #配置项目可执行jar的文件
    app_target: "{{ app_current_path }}/product-service-publish/target/product-
service-publish.jar"
    #配置git分支
    app_version: "{{hostvars[groups['java_cdp_product_app'][0]]
['app_version']}}"
    #配置托管文件件
    app_shared_configs:
      - {src: 'bootstrap.yml', dest: 'product-service-
publish/src/main/resources/bootstrap.yml'}
      - {src: 'application-dev.yml', dest: 'product-service-
publish/src/main/resources/application-dev.yml'}
      - {src: 'logback-spring.xml', dest: 'product-service-
publish/src/main/resources/logback-spring.xml'}
      - {src: 'migrate.sh', dest: 'migrate.sh'}
      - {src: 'build.sh', dest: 'build.sh'}
      - {src: 'redisson-single-dev.yml', dest: 'product-service-
publish/src/main/resources/redisson-single-dev.yml'}
      - {src: 'Dockerfile', dest: 'Dockerfile'}
  roles:
    - { role: docker-build } #使用docker镜像



- hosts: java_cdp_product_app
  any_errors_fatal: true
  vars:
    app_name: java_cdp_product
    app_shared_configs:
       - {src: 'docker-compose.yml', dest: 'docker-compose.yml'}
    group: java_cdp_product_app

  roles:
    - { role: docker-deploy } # 使用docker-compose方式进行部署
```

5、Jenkins添加构建器

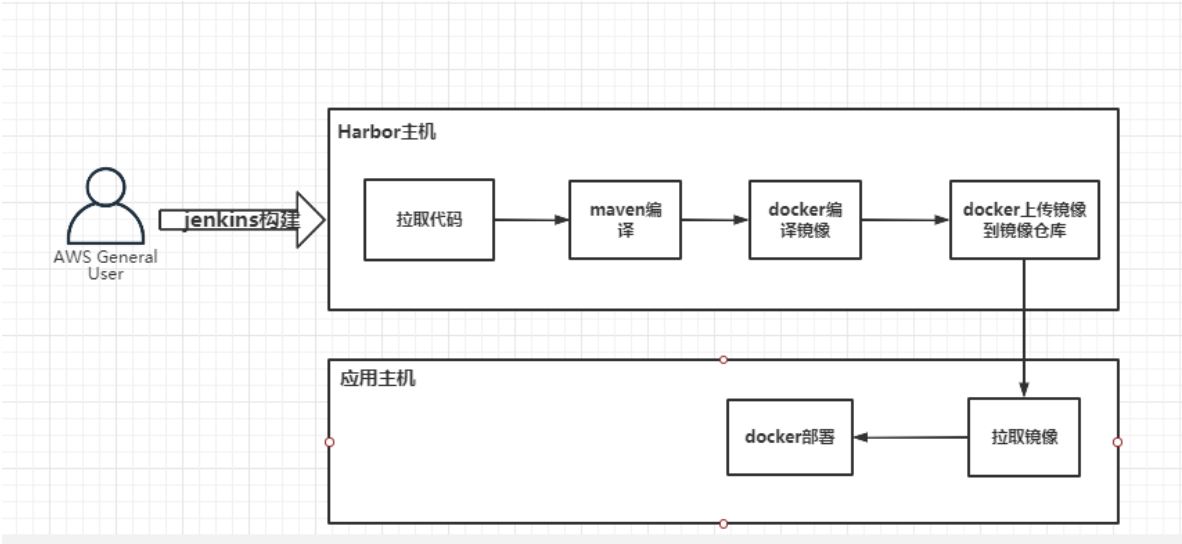
配置devops源码地址和分支，注意不是项目源码

配置ansible Playbook的脚本

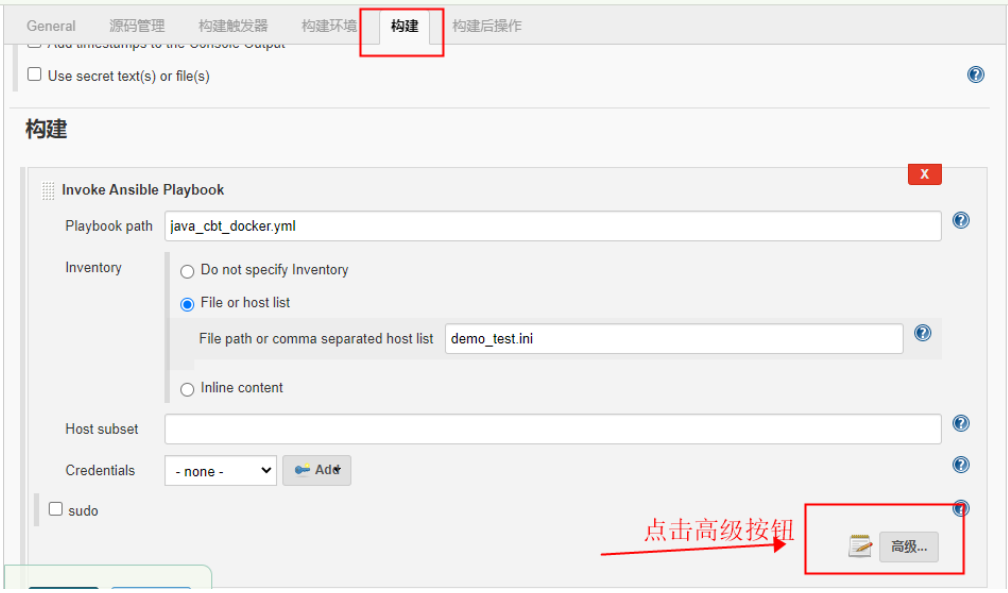
配置钉钉通知

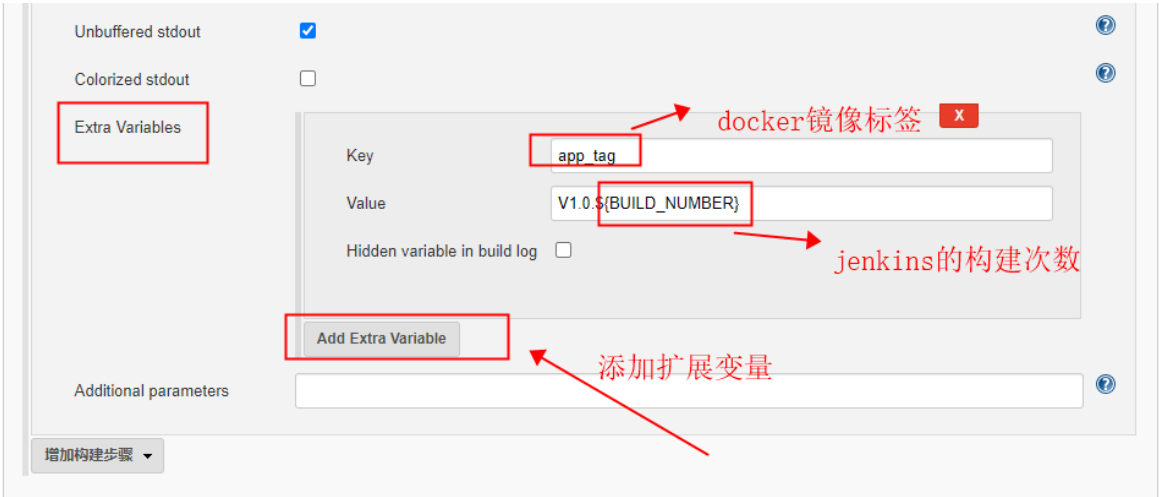
好啦，开始构建之旅。。。

4、docker部署流程



配置步骤跟上面一致，有一点需要注意





参考资料

[Ansible中文权威指南]

[企业级自动化运维方案设计及SaltStack、Ansible等5种工具比较分析]

[Ansible 中的 playbook 详解]

[图解 Ansible 自动化运维]

自动化运维工具 Ansible 知识和技巧 16 篇