

# 大型架构及配置技术

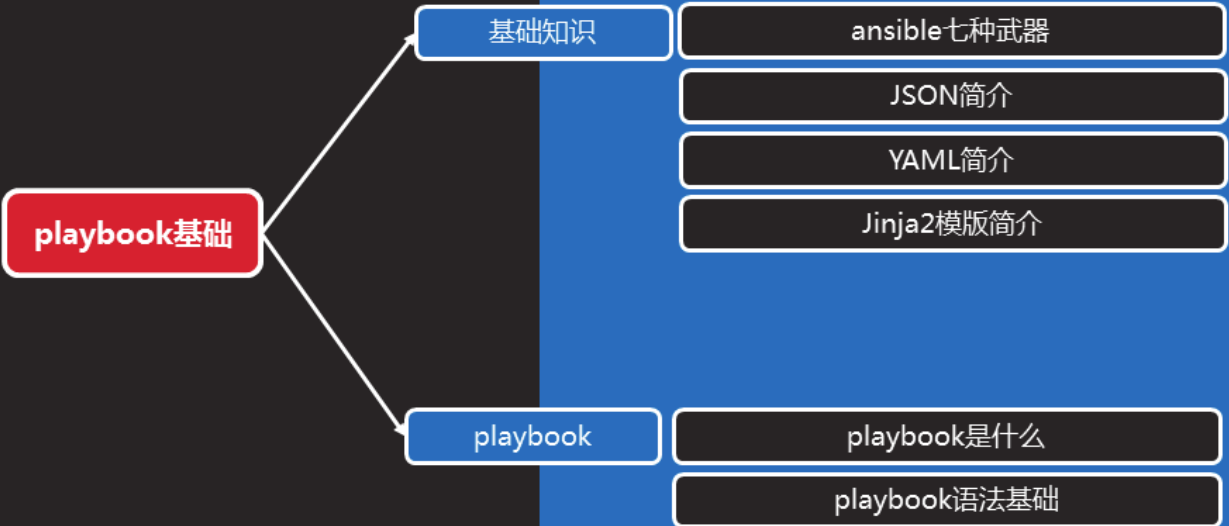
**NSD ARCHITECTURE** **DAY02**

# 内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	playbook基础
	10:30 ~ 11:20	
	11:30 ~ 12:00	playbook进阶
下午	14:00 ~ 14:50	
	15:00 ~ 15:50	
	16:10 ~ 17:10	
	17:20 ~ 18:00	总结和答疑



## playbook基础



# 基础知识

## ansible七种武器

- 第一种武器
  - ansible 命令，用于执行临时性的工作，必须掌握
- 第二种武器
  - ansible-doc是ansible模块的文档说明，针对每个模块都有详细的说明及应用案例介绍，功能和Linux系统man命令类似，必须掌握

## ansible七种武器（续1）

知识讲解

- 第三种武器
  - ansible-console是ansible为用户提供的交互式工具，用户可以在ansible-console虚拟出来的终端上像Shell一样使用ansible内置的各种命令，这为习惯使用Shell交互方式的用户提供了良好的使用体验
- 第四种武器
  - ansible-galaxy从github上下载管理Roles的一款工具，与python的pip类似



## ansible七种武器（续2）

知识讲解

- 第五种武器
  - ansible-playbook是日常应用中使用频率最高的命令，工作机制：通过读取先编写好的playbook文件实现批量管理，可以理解为按一定条件组成的ansible任务集，必须掌握
- 第六种武器
  - ansible-vault主要用于配置文件加密，如编写的playbook文件中包含敏感信息，不想其他人随意查看，可用它加密/解密这个文件



## ansible七种武器（续3）

知识讲解

- 第七种武器
  - ansible-pull
  - ansible有两种工作模式pull/push，默认使用push模式工作，pull和push工作模式机制刚好相反
  - 适用场景：有大批量机器需要配置，即便使用高并发线程依旧要花费很多时间
  - 通常在配置大批量机器的场景下使用，灵活性稍有欠缺，但效率几乎可以无限提升，对运维人员的技术水平和前瞻性规划有较高要求



## JSON简介

知识讲解

- JSON是什么
  - JSON是JavaScript对象表示法，它是一种基于文本独立于语言的轻量级数据交换格式
  - JSON中的分隔符限于单引号"'"、小括号"()"、中括号"[]"、大括号"{}"、冒号":"和逗号","
- JSON 特性
  - JSON是纯文本
  - JSON具有"自我描述性"（人类可读）
  - JSON具有层级结构（值中存在值）
  - JSON可通过JavaScript进行解析



## JSON简介（续1）

知识讲解

- JSON 语法规则
  - 数据在名称/值对中
  - 数据由逗号分隔
  - 大括号保存对象
  - 中括号保存数组
- JSON 数据的书写格式是：名称/值对
  - 名称/值对包括字段名称（在双引号中），后面写一个冒号，然后是值，例如：  
"诗仙"："李白"



## JSON简介（续2）

知识讲解

- JSON语法规则之数组

```
{ "诗人":  
  ["李白", "杜甫", "白居易", "李贺"]  
}
```
- 复合复杂类型

```
{ "诗人":  
  [{"李白":"诗仙", "年代":"唐"},  
    {"杜甫":"诗圣", "年代":"唐"},  
    {"白居易":"诗魔", "年代":"唐"},  
    {"李贺":"诗鬼", "年代":"唐"}  
]
```



## YAML简介（续1）

知识讲解

- YAML基础语法
  - YAML的结构通过空格来展示
  - 数组使用"-"来表示
  - 键值对使用": "来表示
  - YAML使用一个固定的缩进风格表示数据层级结构关系
  - 一般每个缩进级别由两个以上空格组成
  - # 表示注释
- 注意：
  - 不要使用tab，缩进是初学者容易出错的地方之一
  - 同一层级缩进必须对齐



## YAML简介（续2）

知识讲解

- YAML的键值表示方法
  - 采用冒号分隔
  - : 后面必须有一个空格
  - YAML键值对例子

```
"诗仙" : "李白"
```

- 或

```
"李白":  
  "诗仙"
```



## YAML简介（续3）

知识讲解

- 复杂YAML的键值对嵌套

```
"诗人":  
  "李白": "诗仙"
```

或

```
"诗人":  
  "李白":  
    "诗仙"
```

数组

```
["李白", "杜甫", "白居易", "李贺"]
```





## YAML简介（续4）

知识讲解

- YAML 数组表示方法
  - 使用一个短横杠加一个空格
  - YAML数组例子
    - "李白"
    - "杜甫"
    - "白居易"
    - "李贺"
  - 哈希数组复合表达式
    - "诗人":
      - "李白"
      - "杜甫"
      - "白居易"
      - "李贺"



## YAML简介（续5）

知识讲解

- 高级复合表达式
  - "诗人":
    - - "李白": "诗仙"
      - "年代": "唐"
    - - "杜甫": "诗圣"
      - "年代": "唐 "
    - - "白居易": "诗魔"
      - "年代": "唐"
    - - "李贺": "诗鬼"
      - "年代": "唐"



# Jinja2模版简介

知识讲解

- Jinja2是什么
  - Jinja2是基于Python的模板引擎，包含变量和表达式两部分，两者在模板求值时会被替换为值，模板中还有标签，控制模板的逻辑
- 为什么要学习Jinja2模版
  - 因为playbook的模板使用Python的Jinja2模块来处理



## Jinja2模版简介（续1）

知识讲解

- Jinja2模版基本语法
  - 模板的表达式都是包含在分隔符"`{{ }}`"内的
  - 控制语句都是包含在分隔符"`{% %}`"内的
  - 模板支持注释，都是包含在分隔符"`{# #}`"内，支持块注释
  - 调用变量
    - `{{varname}}`
  - 计算
    - `{{2+3}}`
  - 判断
    - `{{1 in [1,2,3]}}`



## Jinja2模版简介（续2）

- Jinja2模版控制语句

```
{% if name == '诗仙' %}  
    李白  
{% elif name == '诗圣' %}  
    杜甫  
{% elif name == '诗魔' %}  
    白居易  
{% else %}  
    李贺  
{% endif %}
```

知识讲解



## Jinja2模版简介（续4）

- Jinja2过滤器

- 变量可以通过过滤器修改。过滤器与变量用管道符号（|）分割，也可以用圆括号传递可选参数，多个过滤器可以链式调用，前一个过滤器的输出会被作为后一个过滤器的输入

- 例如

- 把一个列表用逗号连接起来：{{ list|join(',') }}
- 过滤器这里不再一一列举，需要的可以查询在线文档  
<http://docs.jinkan.org/docs/jinja2/templates.html#builtin-filters>

知识讲解



# playbook

## playbook是什么

- playbook是什么
  - playbook是ansible用于配置，部署和管理托管主机剧本，通过playbook的详细描述，执行其中的一系列tasks，可以让远端主机达到预期状态
  - 也可以说，playbook字面意思即剧本，现实中由演员按剧本表演，在ansible中由计算机进行安装，部署应用，提供对外服务，以及组织计算机处理各种各样的事情



# playbook是什么 (续1)

知识讲解

- 为什么要使用playbook
  - 执行一些简单的任务，使用ad-hoc命令可以方便的解决问题，但有时一个设施过于复杂时，执行ad-hoc命令是不合适的，最好使用playbook
  - playbook可以反复使用编写的代码，可以放到不同的机器上面，像函数一样，最大化的利用代码，在使用ansible的过程中，处理的大部分操作都是在编写playbook



# playbook语法基础

知识讲解

- playbook语法格式
  - playbook由YAML语言编写，遵循YAML标准
  - 在同一行中，#之后的内容表示注释
  - 同一个列表中的元素应该保持相同的缩进
  - playbook由一个或多个play组成
  - play中hosts, variables, roles, tasks等对象的表示方法都是键值中间以": "分隔表示
  - YAML还有一个小的怪癖，它的文件开始行都应该是 ---，这是YAML格式的一部分，表明一个文件的开始



## playbook语法基础（续1）

知识讲解

- playbook构成
  - Target：定义将要执行playbook的远程主机组
  - Variable：定义playbook运行时需要使用的变量
  - Tasks：定义将要在远程主机上执行的任务列表
  - Handler：定义task执行完成以后需要调用的任务



## playbook语法基础（续2）

知识讲解

- playbook执行结果
- 使用ansible-playbook运行playbook文件，输出内容为JSON格式，由不同颜色组成便于识别
  - 绿色代表执行成功
  - \*\*\*代表系统代表系统状态发生改变
  - 红色代表执行失败



## playbook语法基础（续3）

知识讲解

- 第一个playbook

```
---                                # 第一行，表示开始
- hosts: all
  remote_user: root
  tasks:
    - ping:
```

```
ansible-playbook myping.yml -f 5
```

- -f 并发进程数量，默认是5
- hosts行 内容是一个（多个）组或主机的patterns，以逗号为分隔符
- remote\_user 账户名



## playbook语法基础（续4）

知识讲解

- tasks

- 每一个play包含了一个task列表（任务列表）
- 一个task在其所对应的所有主机上（通过 host pattern 匹配的所有主机）执行完毕之后，下一个task才会执行
- 有一点很重要，在一个play之中，所有hosts会获取相同的任务指令，这是play的一个目的所在，即将一组选出的hosts映射到task，执行相同的操作



## playbook语法基础（续5）

知识讲解

- playbook执行命令
    - 给所有主机添加用户plj，设置默认密码123456
    - 要求第一次登录修改密码
- ```
---  
- hosts: all  
  remote_user: root  
  tasks:  
    - name: create user plj  
      user: group=wheel uid=1000 name=plj  
    - shell: echo 123456 | passwd --stdin plj  
    - shell: chage -d 0 plj
```



## 练习1：playbook练习

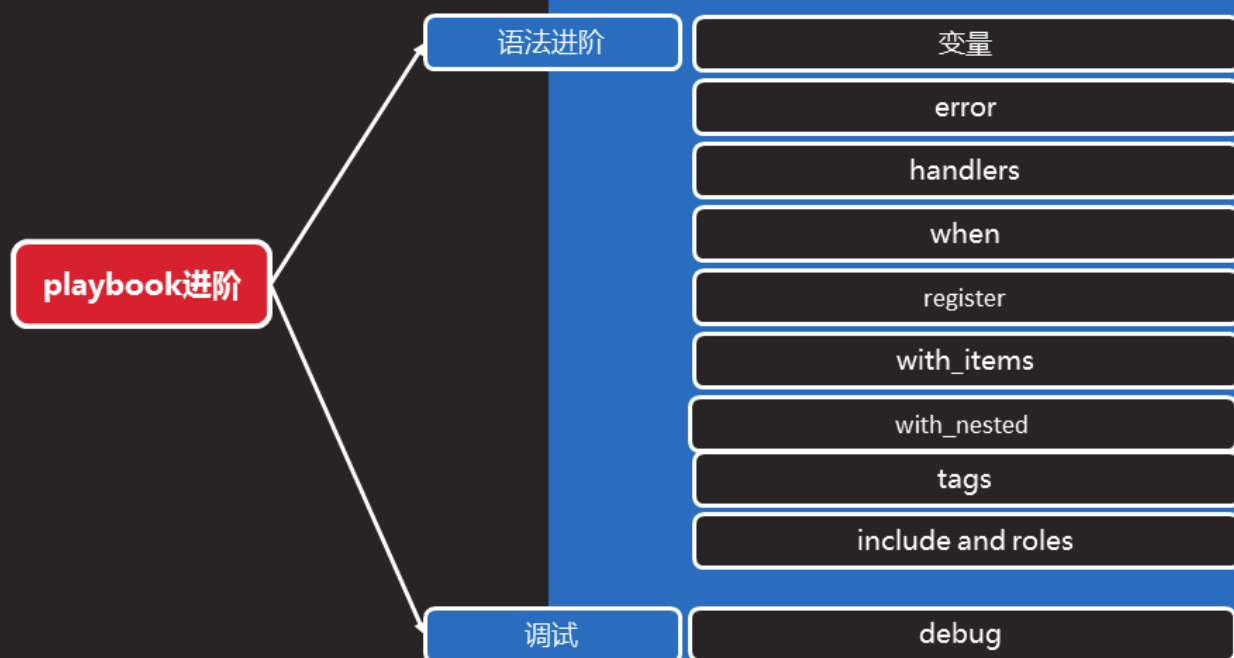
课堂练习

1. 安装Apache并修改监听端口为8080
2. 修改ServerName配置，执行apachectl -t命令不报错
3. 设置默认主页hello world
4. 启动服务并设开机自启





## playbook进阶



## 语法进阶

## 变量（续1）

- 设密码
  - 解决密码明文问题
  - user模块的password为什么不能设置密码呢
  - 经过测试发现，password是把字符串直接写入shadow，并没有改变，而Linux的shadow密码是经过加密的，所以不能使用
- 解决方案
  - 变量过滤器password\_hash

```
{{ 'urpassword' | password_hash('sha512') }}
```



## 变量（续2）

知识讲解

- 变量过滤器

- 给所有主机添加用户plj，设置默认密码123456
- 要求第一次登录修改密码（使用变量）

```
---  
- hosts: 192.168.1.16  
  remote_user: root  
  vars:  
    username: plj  
  tasks:  
    - name: create user "{{username}}"  
      user: group=wheel uid=1000 password={{'123456' |  
password_hash('sha512')}} name={{username}}  
    - shell: chage -d 0 {{username}}
```



## 案例2：变量练习

1. 练习使用user模块添加用户
2. 练习使用变量简化task，让play通用性更强
3. 练习使用过滤器

课堂练习



## error

- ansible-playbook对错误的处理
  - 默认情况判断\$?，如果值不为0就停止执行
  - 但某些情况我们需要忽略错误继续执行

```
---  
- hosts: 192.168.1.16  
  remote_user: root  
  vars:  
    username: plj  
  tasks:  
    - name: create user "{{username}}"  
      user: group=wheel uid=1000  
      password={{'123456'|password_hash('sha512')}}  
      name={{username}}  
      - shell: setenforce 0  
      - shell: chage -d 0 {{username}}
```

知识讲解



## error ( 续1 )

- 错误处理方法
  - 关闭selinux，如果selinux已经关闭，返回1，若之前已经关闭则不算错误，可以忽略错误继续运行，忽略错误有两种方法
  - 第一种方式：

```
shell: /usr/bin/somecommand || /bin/true
```
  - 第二种方式：

```
- name: run some command  
  shell: /usr/bin/somecommand  
  ignore_errors: True
```

知识讲解



## error ( 续2 )

- 完整的playbook

```
---
- hosts: 192.168.1.16
  remote_user: root
  vars:
    username: plj
  tasks:
    - name: create user "{{username}}"
      user: group=wheel uid=1000
      password={{'123456'|password_hash('sha512')}}
      name={{username}}
    - shell: setenforce 0
      ignore_errors: true
    - shell: chage -d 0 {{username}}
```

知识讲解



## handlers

- 当关注的资源发生变化时采取的操作
- notify这个action可用于在每个play的最后被触发，这样可以避免有多次改变发生时每次都执行指定的操作，取而代之仅在所有的变化发生完成后一次性地执行指定操作
- 在notify中列出的操作称为handler，即notify调用handler中定义的操作

知识讲解



