



Ansible自动化运维应用实战

讲师介绍



阿良

资深运维工程师，51CTO知名博主。曾就职在IDC，大数据，金融行业，现任职奇虎360公司。经重重磨炼，具有丰富的运维实战经验。

技术博客：<http://blog.51cto.com/lizhenliang>



学员群：[545214087](https://t.me/545214087)

课程目录

- 一 Ansible概述
- 二 Ansible安装与配置
- 三 Ad-hoc命令模式
- 四 Ansible常用模块
- 五 Playbook基本使用
- 六 Playbook定义变量与使用
- 七 Playbook文件复用
- 八 Playbook流程控制
- 九 Playbook模板 (jinja2)
- 十 角色 (Roles)

第 1 章 Ansible概述

1. IT自动化的好处
2. Ansible是什么
3. Ansible架构
4. 先来认识一下Ansible

IT自动化的好处

团队影响

- 节省时间，提高工作效率
- 消除重复任务
- 更少的错误风险
- 改善协作和工作满意度

企业影响

- 克服复杂性
- 更多创新资源
- 加强问责制和合规性

Ansible是什么

简单 – 减少学习成本

- 易读的描述语言
- 无需特殊编码技能
- 任务按顺序执行

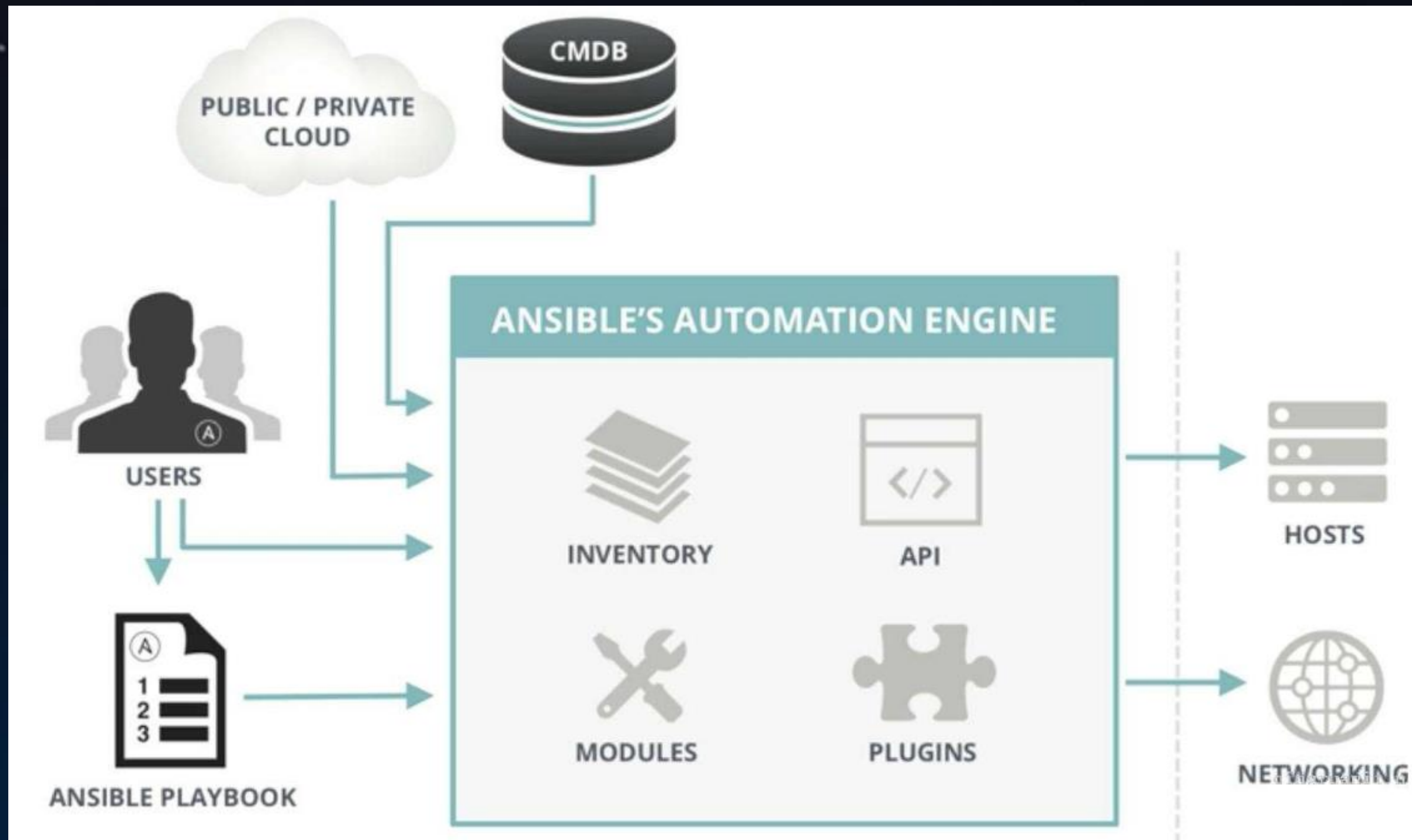
强大 – 协调应用程序生命周期

- 应用部署
- 配置管理
- 工作流程编排

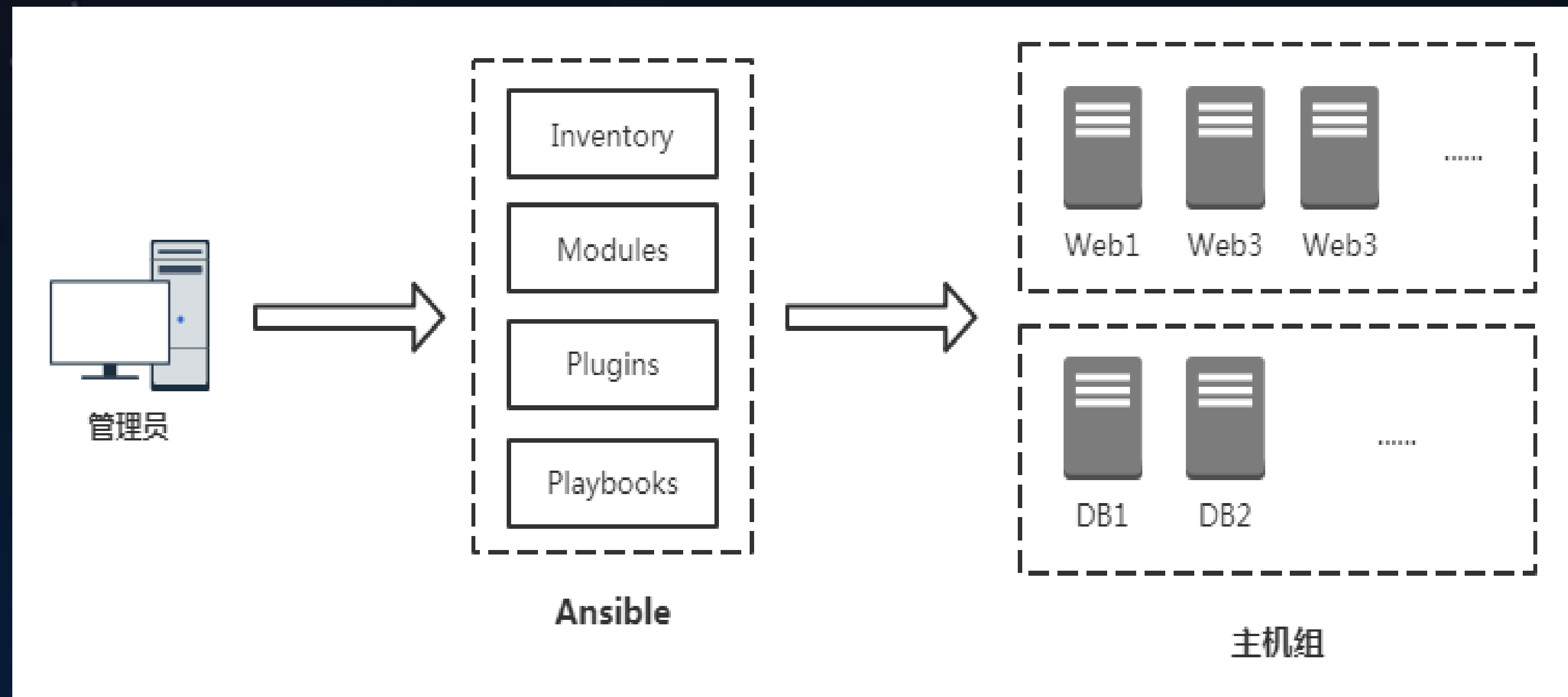
无代理 – 可预测，可靠和安全

- 无代理架构
- 使用OpenSSH通信
- 没有代理维护成本

Ansible架构



Ansible架构



第 2 章 Ansible安装与配置

1. Ansible使用要求
2. 安装Ansible
3. 配置文件
4. Inventory (主机清单)

Ansible使用要求

服务端要求

- Python2.6/2.7/3.x
- RedHat, Debian, CentOS, OS X等。不支持Windows

被管理端要求

- OpenSSH
- Python2.6/2.7/3.x

安装Ansible

- ◆ `yum install ansible` (推荐)
- ◆ `pip install ansible`
- ◆ <https://releases.ansible.com/ansible> or <https://github.com/ansible/ansible.git>

配置文件

```
# vi /etc/ansible/ansible.cfg
[defaults]
inventory = /etc/ansible/hosts
forks = 5
become = root
remote_port = 22
host_key_checking = False
timeout = 10
log_path = /var/log/ansible.log
private_key_file = /root/.ssh/id_rsa
```

Inventory (主机清单)

示例1: 未分组的主机

green.example.com

blue.example.com

192.168.100.1

192.168.100.10

示例2: 属于webservers组主机集合

[webservers]

alpha.example.org

beta.example.org

192.168.1.100

192.168.1.110

www[001:006].example.com

示例3: 属于dbservers组主机集合

[dbservers]

db01.intranet.mydomain.net

db02.intranet.mydomain.net

10.25.1.56

10.25.1.57

db-[99:101]-node.example.com

Inventory (主机清单)

主机和主机组变量:

```
[webservers]
192.168.1.10 ansible_ssh_user=root ansible_ssh_pass='123456' http_port=80
192.168.1.11 ansible_ssh_user=root ansible_ssh_pass='123456' http_port=80
```

```
[webservers:vars]
http_port=8080
server_name=www.ctnrs.com
```

组变量分解到单个文件:

```
# cat /etc/ansible/group_vars/webservers.yml
http_port: 8080
server_name: www.ctnrs.com
```

第 3 章 ad-hoc命令

1. 命令行工具常用选项
2. SSH密码认证
3. SSH密钥对认证

命令行工具常用选项

格式: **ansible <host-pattern> [options]**

选项:

-a MODULE_ARGS, --args=MODULE_ARGS
-C, --check
-e EXTRA_VARS, --extra-vars=EXTRA_VARS
-f FORKS, --forks=FORKS
-i INVENTORY, --inventory=INVENTORY
--list-hosts
-m MODULE_NAME, --module-name=MODULE_NAME
--syntax-check
-t TREE, --tree=TREE
-v, --verbose
--version

连接选项: 控制谁连接主机和如何连接

-k, --ask-pass
--private-key=PRIVATE_KEY_FILE, --key-file=PRIVATE_KEY_FILE
-u REMOTE_USER, --user=REMOTE_USER
-T TIMEOUT, --timeout=TIMEOUT

提权选项: 控制在目标主机以什么用户身份运行

-b, --become
--become-method=BECOME_METHOD
--become-user=BECOME_USER
-K, --ask-become-pass

模块参数

运行检查, 不执行任何操作

设置附加变量 key=value

指定并行进程数量, 默认5

指定主机清单文件路径

输出匹配的主机列表, 不执行任何操作

执行的模块名, 默认command

语法检查playbook文件, 不执行任何操作

将日志输出到此目录

详细信息, -vvv更多, -vvvv debug

查看程序版本

请求连接密码

私钥文件

连接用户, 默认None

覆盖连接超时时间, 默认10秒

以另一个用户身份操作

提权方法, 默认sudo

提权后的用户身份, 默认root

提权密码

SSH密码认证

```
[webservers]  
192.168.1.10:22 ansible_ssh_user=root ansible_ssh_pass='123456'  
192.168.1.11:22 ansible_ssh_user=root ansible_ssh_pass='123456'
```

SSH密钥对认证

```
[webservers]  
192.168.1.10:22 ansible_ssh_user=root ansible_ssh_key=/root/.ssh/id_rsa  
192.168.1.11:22 ansible_ssh_user=root
```

第 4 章 Ansible常用模块

- 执行shell命令 (command和shell)
- 文件传输 (copy和file)
- 管理软件包 (yum)
- 用户和组 (user)
- 从源代码管理系统部署 (git)
- 管理服务 (service)
- 收集目标主机信息 (setup)

ansible_nodename
ansible_os_family
ansible_pkg_mgr
ansible_processor
ansible_processor_cores

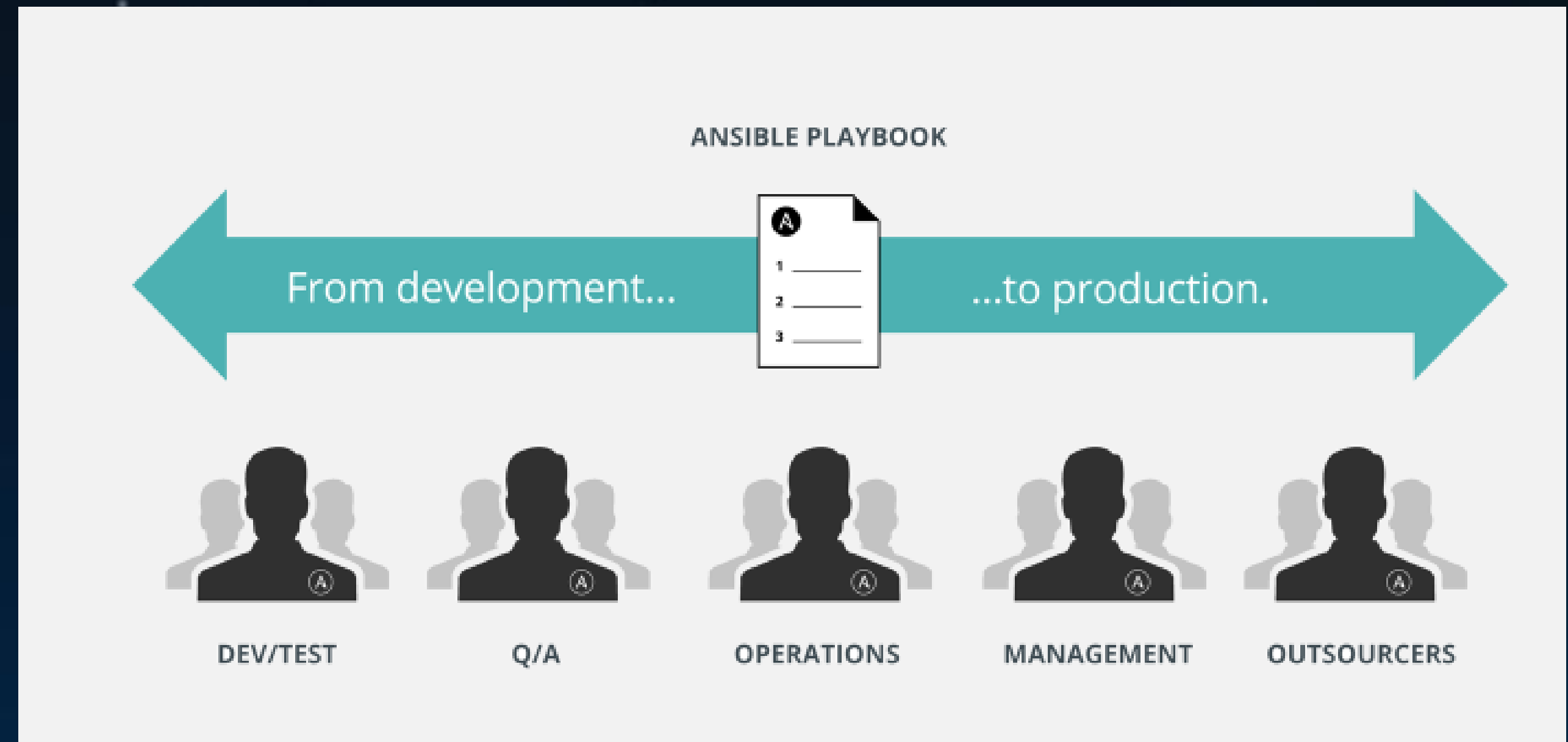
第 5 章 Playbook基本使用

1. 使用Playbook的好处
2. 先来认识一下Playbook (自动部署Nginx)
3. YAML语法
4. Playbook文件结构
5. 在变更时执行操作 (handlers)
6. 任务控制 (tags)
7. Playbook文件调试
8. 案例：自动部署Tomcat

使用Playbook有什么好处

特点

- 易读的编排语言 - YAML
- 适合配置管理和应用部署
- 非常适合部署复杂的工作



先来认识一下Playbook

```
# main.yml
---
- hosts: webservers
  vars:
    hello: Ansible

  tasks:
  - name: Add repo
    yum_repository:
      name: nginx
      description: nginx repo
      baseurl: http://nginx.org/packages/centos/7/$basearch/
      gpgcheck: no
      enabled: 1
  - name: Install nginx
    yum:
      name: nginx
      state: latest
  - name: Copy nginx configuration file
    copy:
      src: ./site.conf
      dest: /etc/nginx/conf.d/site.conf
  - name: Start nginx
    service:
      name: nginx
      state: started
  - name: Create wwwroot directory
    file:
      dest: /var/www/html
      state: directory
  - name: Create test page index.html
    shell: echo "hello {{hello}}" > /var/www/html/index.html
```

```
# site.conf
server {
    listen 80;
    server_name www.ctnrs.com;
    location / {
        root    /var/www/html;
        index   index.html;
    }
}
```

执行playbook: `ansible-playbook main.yml`

YAML语法

- ◆ 缩进表示层级关系
- ◆ 不支持制表符“tab”缩进，使用空格缩进
- ◆ 通常开头缩进 2 个空格
- ◆ 字符后缩进 1 个空格，如冒号、逗号等
- ◆ “---” 表示YAML格式，一个文件的开始
- ◆ “#” 注释

Playbook文件结构

```
---
- name: play1
  hosts: webserver
  remote_user: root
  vars:
    var_name: value
  tasks:
    - name: echo
      shell: "echo {{var_name}}"

- name: play2
  hosts: webserver
  remote_user: root
  vars:
    var_name: value
  tasks:
    - name: echo
      shell: "echo {{var_name}}"
```

```
---
- hosts: webserver
  remote_user: root
  vars:
    var_name: value
  tasks:
    - name: echo
      shell: "echo {{var_name}}"
```

在变更时执行操作 (handlers)

```
---
- hosts: webservers
  gather_facts: no

  tasks:
    - name: Copy nginx configuration file
      copy:
        src: ./site.conf
        dest: /etc/nginx/conf.d/site.conf
      notify:
        - restart nginx

  handlers:
    - name: restart nginx
      service: name=nginx state=reloaded
```

notify: 在任务结束时触发

handlers: 由特定条件触发Tasks

任务控制 (tags)

```
---
- hosts: webservers
  gather_facts: no

  tasks:
    - name: Install redis
      yum: name=redis state=present
      tags: install

    - name: Copy redis configuration file
      copy: src=redis.conf dest=/etc/redis/redis.conf
      tags: configuration

    - name: Restart redis
      service: name=redis state=restarted
      tags: restart
```

指定: `ansible-playbook example.yml --tags "configuration,install"`

跳过: `ansible-playbook example.yml --skip-tags "install"`

Playbook文件调试

语法检查: `ansible-playbook main.yml --syntax-check`

打印语句:

```
- hosts: webservers
  tasks:
    - debug:
      msg: {{group_names}}
    - debug:
      msg: {{inventory_hostname}}
    - debug:
      msg: {{ansible_hostname}}
```

案例：自动部署Tomcat

```
---
- hosts: webservers
  gather_facts: no
  vars:
    tomcat_version: 8.5.34
    tomcat_install_dir: /usr/local

  tasks:
    - name: Install jdk1.8
      yum: name=java-1.8.0-openjdk state=present

    - name: Download tomcat
      get_url: url=http://mirrors.hust.edu.cn/apache/tomcat/tomcat-
8/v{{ tomcat_version }}/bin/apache-tomcat-{{ tomcat_version }}.tar.gz dest=/tmp

    - name: Unarchive tomcat-{{ tomcat_version }}.tar.gz
      unarchive:
        src: /tmp/apache-tomcat-{{ tomcat_version }}.tar.gz
        dest: "{{ tomcat_install_dir }}"
        copy: no

    - name: Start tomcat
      shell: cd {{ tomcat_install_dir }} &&
        mv apache-tomcat-{{ tomcat_version }} tomcat8 &&
        cd tomcat8/bin && nohup ./startup.sh &
```

第 6 章 Playbook变量定义与使用

1. 命令行
2. 在Inventory中定义
3. 在Playbook中定义
4. 在Roles中定义
5. 注册变量 (register)
6. 系统信息变量 (facts)

Playbook定义变量与使用

在Playbook中定义变量

```
- hosts: webservers
  gather_facts: no
  vars:
    var_name: value
    var_name: value
  tasks:
    - name: hello
      shell: "echo {{var_name}}"
```

注册变量

```
- hosts: webservers
  gather_facts: no
  tasks:
    - name: Get date
      command: date +"%F_%T"
      register: date_output
    - name: Echo date_output
      command: touch /tmp/{{date_output.stdout}}
```

系统变量

```
- hosts: webservers
  tasks:
    - name: Get hostname
      debug: msg={{ansible_hostname}}
```

第 7 章 Playbook文件复用

1. include & import 区别
2. import_playbook
3. include_tasks
4. import_tasks

include & import 区别

include*（动态）：在运行时导入

- `--list-tags`, `--list-tasks`不会显示到输出
- 不能使用`notify`触发来自`include*`内处理程序名称（handlers）

import*（静态）：在Playbook解析时预先导入

- 不能与循环一起使用
- 将变量用于目标文件或角色名称时，不能使用`inventory`（主机/主机组等）中的变量

import_playbook

```
# main.yml
---
- import_playbook: webservers.yml
- import_playbook: databases.yml
```

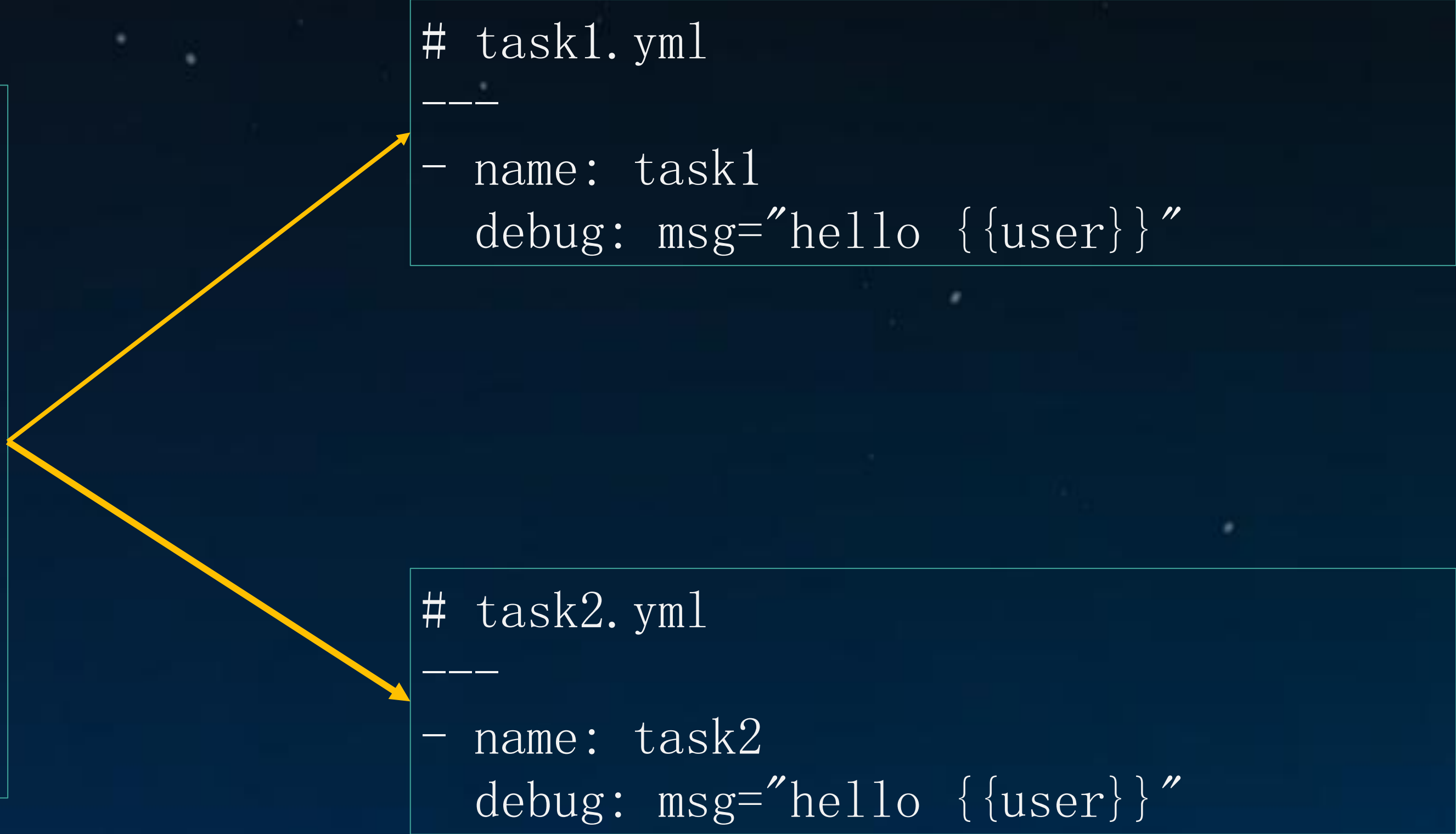
```
graph LR; A["# main.yml  
---  
- import_playbook: webservers.yml  
- import_playbook: databases.yml"] --> B["# webservers.yml  
---  
- hosts: webservers  
  tasks:  
    - debug: msg='test webserver'"]; A --> C["# database.yml  
---  
- hosts: webservers  
  tasks:  
    - debug: msg='test database'"]
```

```
# webservers.yml
---
- hosts: webservers
  tasks:
    - debug: msg="test webserver"
```

```
# database.yml
---
- hosts: webservers
  tasks:
    - debug: msg="test database"
```

include_tasks & import_tasks

```
# main.yml
---
- hosts: webservers
  gather_facts: no
  tasks:
    - include_tasks: task1.yml
      vars:
        user: zhangsan
    - import_tasks: task2.yml
      vars:
        user: lisi
```



```
# task1.yml
---
- name: task1
  debug: msg="hello {{user}}"
```

```
# task2.yml
---
- name: task2
  debug: msg="hello {{user}}"
```

第 8 章 Playbook流程控制

1. 条件
2. 循环

条件

```
- hosts: webservers

tasks:
  - name: Host 192.168.1.12 run this task
    debug: msg="{{ansible_default_ipv4.address}}"
    when: ansible_default_ipv4.address == '192.168.1.12'
```

```
- hosts: webservers

tasks:
  - name: Update apache version - yum
    yum: name=httpd state=present
    when: ansible_pkg_mgr == 'yum'
    notify: restart httpd

  - name: Update apache version - apt
    apt: name=apache2 state=present update_cache=yes
    when: ansible_pkg_mgr == 'apt'
    notify: restart apache2

handlers:
  - name: restart httpd
    service: name=httpd state=restarted
  - name: restart apache2
    service: name=apache2 state=restarted
```


条件

```
tasks:
  - name: "shut down CentOS 6 and Debian 7 systems"
    command: /sbin/shutdown -t now
    when: (ansible_distribution == "CentOS" and ansible_distribution_major_version == "6") or
          (ansible_distribution == "Debian" and ansible_distribution_major_version == "7")
```

```
tasks:
  - name: "shut down CentOS 6 systems"
    command: /sbin/shutdown -t now
    when:
      - ansible_distribution == "CentOS"
      - ansible_distribution_major_version == "6"
```

循环

```
- name: with_list
  debug:
    msg: "{{ item }}"
  with_list:
    - one
    - two

- name: with_list -> loop
  debug:
    msg: "{{ item }}"
  loop:
    - one
    - two
```

```
- name: with_items
  debug:
    msg: "{{ item }}"
  with_items: "{{ items }}"

- name: with_items -> loop
  debug:
    msg: "{{ item }}"
  loop: "{{ items|flatten(levels=1) }}"
```

第 9 章 模板 (jinja2)

1. 条件和循环
2. 案例：管理Nginx配置文件

条件和循环

```
# test.yml
---
- hosts: webserver
  vars:
    hello: Ansible

  tasks:
    - template: src=f.j2 dest=/tmp/f.j2
```

```
# f.j2
{% set list=['one', 'two', 'three'] %}

{% for i in list %}
  {% if i == 'two' %}
    -> two
  {% elif loop.index == 3 %}
    -> 3
  {% else %}
    {{i}}
  {% endif %}
{% endfor %}

{{ hello }}
```

```
{% set dict={'zhangsan': '26', 'lisi': '25'} %}
{% for key, value in dict.items() %}
  {{key}} -> {{value}}
{% endfor %}
```

案例：管理Nginx配置文件

```
# main.yml
---
- hosts: webservers
  gather_facts: no
  vars:
    http_port: 80
    server_name: www.ctnrs.com

  tasks:
    - name: Copy nginx configuration file
      template: src=site.conf.j2
      dest=/etc/nginx/conf.d/www.ctnrs.com.conf
      notify: reload nginx

  handlers:
    - name: reload nginx
      service: name=nginx state=reloaded
```

```
# site.conf.j2
{% set list=[10, 12, 13, 25, 31] %}
upstream {{server_name}} {
    {% for i in list %}
        server 192.168.1. {{i}}:80;
    {% endfor %}
}

server {
    listen      {{ http_port }};
    server_name {{ server_name }};

    location / {
        proxy_pass http://{{server_name}};
    }
}
```

第 10 章 角色 (roles)

1. Roles目录结构
2. Roles基本使用
3. 案例：部署Web服务器

Roles目录结构

```
site.yml
webservers.yml
fooservers.yml
hosts
roles/
  common/
    files/
    templates/
    tasks/
    handlers/
    vars/
    defaults/
    meta/
  webservers/
    files/
    templates/
    tasks/
    handlers/
    vars/
```

- tasks - 包含角色要执行的主要任务列表
- handlers - 包含角色使用的处理程序
- defaults - 角色默认的变量
- vars - 角色其他的变量
- files - 角色部署时用到的文件
- templates - 角色部署时用到的模板
- meta - 角色定义的一些元数据

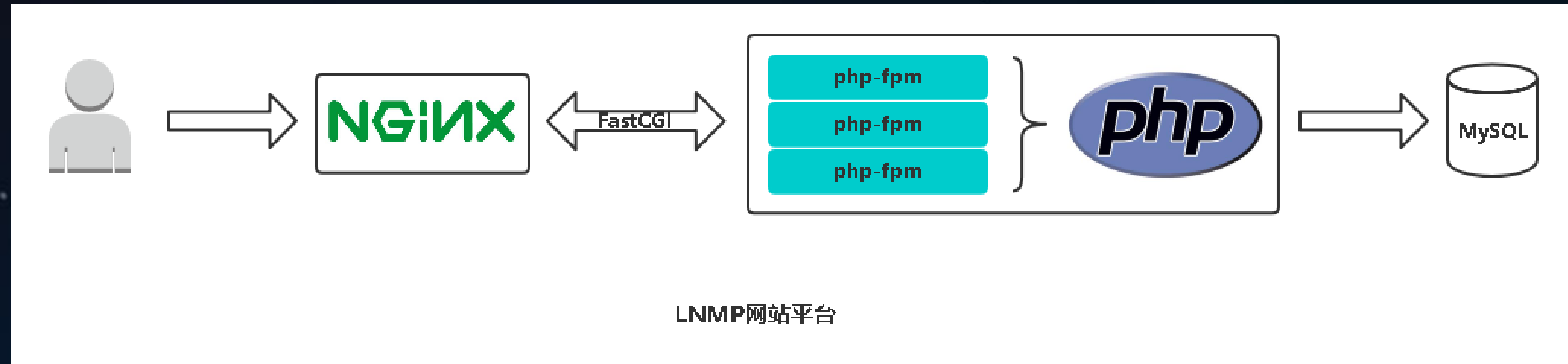
Roles基本使用

```
---
- hosts: webserver
  roles:
    - common
    - nginx
    - php
```

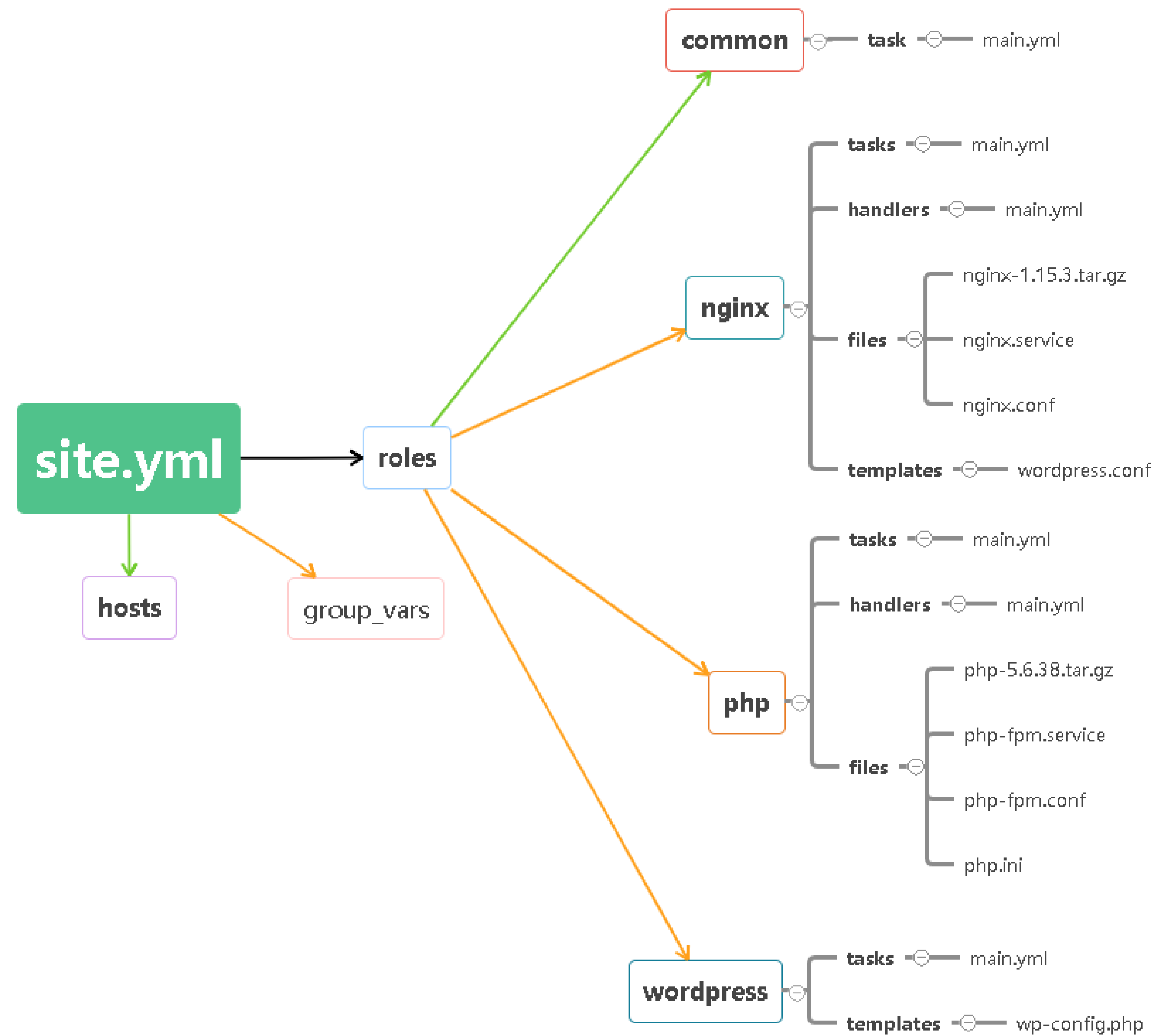
```
---
- hosts: webserver
  roles:
    - common
    - role: nginx
      vars:
        dir: '/opt/a'
        app_port: 5000
    - role: php
      vars:
        dir: '/opt/b'
        app_port: 5001
```

```
---
- hosts: webserver
  roles:
    - role: common
      tags: ["common"]
    - role: nginx
      tags: ["nginx"]
    - role: php
      tags: ["php"]
```


案例：部署Web服务器



案例：部署Web服务器



参考文档

最佳实践: https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html

示例参考: <https://github.com/ansible/ansible-examples>

Thank You