Tedu.cn
达内教育

# 运维开发实战

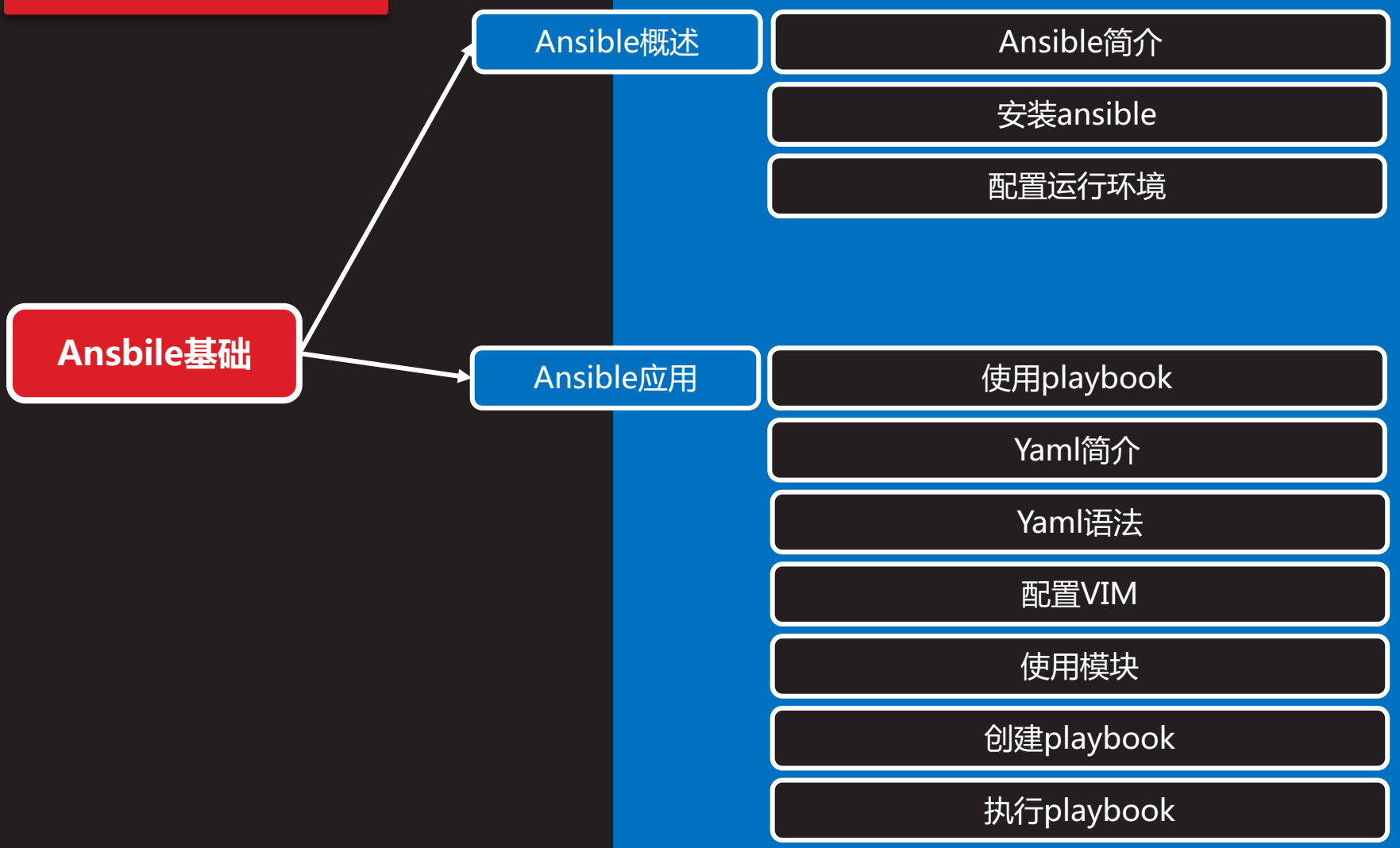
NSD DEVOPS

DAY03

# 内容

| | | |
|---|---|---|
| 上午 | 09:00 ~ 09:30 | 作业讲解和回顾 |
| | 09:30 ~ 10:20 | Ansible基础 |
| | 10:30 ~ 11:20 | |
| | 11:30 ~ 12:00 | Ansible编程 |
| 下午 | 14:00 ~ 14:50 | |
| | 15:00 ~ 15:50 | 编写ansible模块 |
| | 16:10 ~ 17:00 | |
| | 17:10 ~ 18:00 | 总结和答疑 |

# Ansbile基础


Ansbile基础
Ansible概述
Ansible简介
安装ansible
配置运行环境
Ansible应用
使用playbook
Yaml简介
Yaml语法
配置VIM
使用模块
创建playbook
执行playbook

Tedu.cn
达内教育

# Ansible概述

# Ansible简介



- Ansible是一个配置管理和配置工具，类似于Chef，Puppet或Salt
- 这是一款很简单也很容易入门的部署工具，它使用SSH连接到服务器并运行配置好的任务
- 服务器上不用安装任何多余的软件，只需要开启ssh，所有工作都交给client端的ansible负责

知识讲解

# 安装ansible

- 在线安装

```
[root@localhost ~]# yum install -y ansible
```

- 使用本地安装包
  - 参见《大型技术架构》课程ansible安装

知识讲解

# 配置运行环境



知识讲解

- 创建ansible工作目录

```
[root@bogon ~]# mkdir /root/myansi/
```

- 创建配置文件

```
[root@bogon ~]# cd /root/myansi/
[root@bogon myansi]# cat ansible.cfg
[defaults]
inventory = hosts
remote_user = root
```

# 配置运行环境（续1）



知识讲解

- 声明被管理主机

```
[root@bogon myansi]# cat hosts
[myself]
localhost
[webservers]
node1.tedu.cn
```

- 配置名称解析

```
[root@bogon myansi]# cat /etc/hosts
127.0.0.1   localhost.localdomain localhost
192.168.113.134 node1.tedu.cn   node1
```

# 配置运行环境（续2）



- 添加远程主机密钥到信任列表

```
[root@bogon myansi]# ssh-keyscan localhost 127.0.0.1 node1.tedu.cn
192.168.113.134 >> /root/.ssh/known_hosts
```

- 测试

```
[root@bogon myansi]# ansible all -m ping -k
SSH password:
localhost | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
node1.tedu.cn | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

知识讲解

# 案例1：准备ansible环境

1. 创建ansible工作目录
2. 创建配置文件及主机列表文件
3. 测试在远程主机执行命令

课堂练习

# Ansible应用

# 使用playbook



知识讲解

- Playbooks是Ansible的配置、部署、编排语言。
- 它们可以被描述为一个需要希望远程主机执行命令的方案，或者一组程序运行的命令集合
- Playbook由一到多个Play组成
- 每个play可以指定哪些主机执行哪些任务
- 执行任务一般通过调用模块来实现

# Yaml简介

- Playbooks的格式是YAML
- 语法做到最小化，意在避免 playbooks 成为一种编程语言或是脚本
- 使用 YAML 是因为它像 XML 或 JSON 是一种利于人们读写的数据格式

知识讲解

# Yaml语法



知识讲解

- 每一个 YAML 文件都是从一个列表开始
- 列表中的每一项都是一个键值对, 通常它们被称为一个“哈希 或“字典”
- 所有的 YAML 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
- 列表中的所有成员都开始于相同的缩进级别，并且使用一个 "- " 作为开头(一个横杠和一个空格)
- 一个字典是由一个简单的 键: 值 的形式组成(冒号后面必须是一个空格)

# 配置VIM



知识讲解

- Yaml的缩进不能使用tab键
- 建议缩进为两个空格
- 为了实现yml文件按tab键缩进两个空格，可以按以下方式对vim进行定制

```
[root@bogon myansi]# cat ~/.vimrc
autocmd FileType yaml setlocal sw=2 ts=2 et ai
```

# 使用模块



知识讲解

- Ansible的模块实际上就是一个个的python程序文件
- Ansible执行任务就是通过调用这些模块来完成的
- 查看模块列表

  [root@bogon myansi]# ansible-doc -l
- 查看模块帮助

  [root@bogon myansi]# ansible-doc yum

# 创建playbook

- 编写playbook，在web服务器上安装httpd服务

```
[root@bogon myansi]# vim install_web.yml
---
- name: configure httpd web service
  hosts: webservers
  tasks:
    - name: install httpd php php-mysql
      yum:
        name: "{{item}}"
        state: present
      with_items:
        - httpd
        - php
        - php-mysql
.. .. ..
    - name: start httpd
      service:
        name: httpd
        state: started
        enabled: true
```

知识讲解

# 执行playbook

- 检查语法

```
[root@bogon myansi]# ansible-playbook --syntax-check install_web.yml
```

- 执行

```
[root@bogon myansi]# ansible-playbook  install_web.yml -k
```

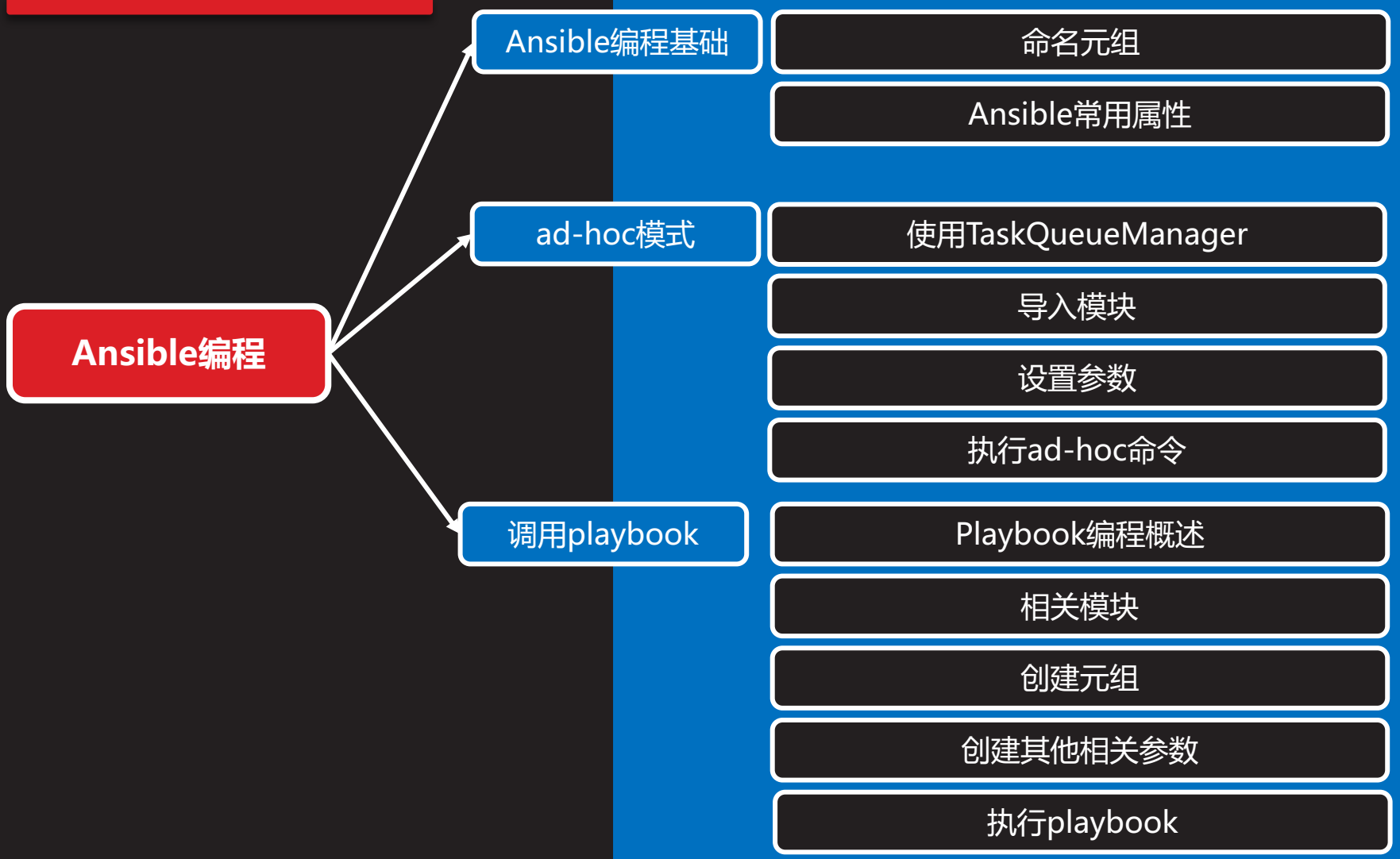
知识讲解

# 案例2：使用playbook

1. Playbook有两个play
2. 一个play用于在webservers安装并启动httpd服务
3. 另一个play用于在dbservers安装并启动mariadb服务

课堂练习

# Ansible编程


Ansible编程
Ansible编程基础
命名元组
Ansible常用属性
ad-hoc模式
使用TaskQueueManager
导入模块
设置参数
执行ad-hoc命令
调用playbook
Playbook编程概述
相关模块
创建元组
创建其他相关参数
执行playbook

Tedu.cn
达内教育

# Ansible编程基础

# 命名元组



- 命名元组与普通元组一样，有相同的表现特征，其添加的功能就是可以根据名称引用元组中的项
- collections 模块提供了namedtuple()函数，用于创建自定义的元组数据类型

```
>>> from collections import namedtuple
>>> user = namedtuple('user', ['name', 'age'])
>>> bob = user('Bob Green', 23)
>>> bob[0]
'Bob Green'
>>> bob[1]
23
>>> bob.name
'Bob Green'
```

知识讲解

# Ansible常用属性



知识讲解

- from ansible.parsing.dataloader import DataLoader
  - 用来加载解析yaml文件或JSON内容,并且支持vault的解密
- from ansible.vars.manager import VariableManager
  - 管理变量的类，包括主机，组，扩展等变量
- from ansible.inventory.manager import InventoryManager
  - 用于创建主机清单，主机清单的源采用配置文件或是逗号分开主机名字符串

# Ansible常用属性（续1）



知识讲解

- from ansible.playbook.play import Play
  - 用于创建play对象，能够通过play_source提供的信息自动创建任务对象
- from ansible.executor.task_queue_manager import TaskQueueManager
  - 用于处理进程池中的多进程。队列管理器负责加载play策略插件，以便在选定的主机上执行任务
- import ansible.constants as C
  - 存储ansible一些预定义变量

# ad-hoc模式

# 使用TaskQueueManager

- 创建TaskQueueManager实例，用于管理子进程、通过主机列表和任务建立对象
- TaskQueueManager需要主机清单参数、主机变量参数、连接选项等

知识讲解

# 导入模块

- 不管是执行ad-hoc还是playbook都需要以下模块

```
#!/usr/bin/env python

import shutil
from collections import namedtuple
from ansible.parsing.dataloader import DataLoader
from ansible.vars.manager import VariableManager
from ansible.inventory.manager import InventoryManager
from ansible.playbook.play import Play
from ansible.executor.task_queue_manager import TaskQueueManager
import ansible.constants as C
```

# 设置参数



- 运行命令时有很多参数要指定，这些参往往很长，可以先把它们提前定义出来

```
Options = namedtuple('Options', ['connection', 'module_path', 'forks',
'become', 'become_method', 'become_user', 'check', 'diff'])
options = Options(connection='local', module_path=[''], forks=10,
become=None, become_method=None, become_user=None,
check=False, diff=False)
loader = DataLoader()
passwords = dict(vault_pass='secret')
inventory = InventoryManager(loader=loader, sources='localhost,')
variable_manager = VariableManager(loader=loader,
inventory=inventory)
```

知识讲解

# 设置参数（续1）



- 运行命令时有很多参数要指定，这些参往往很长，可以先把它们提前定义出来

```
play_source =  dict(
        name = "Ansible Play",
        hosts = 'localhost',
        gather_facts = 'no',
        tasks = [
            dict(action=dict(module='shell', args='mkdir /tmp/mytestdir'),
register='shell_out'),
    ]
    )
play = Play().load(play_source, variable_manager=variable_manager,
loader=loader)
```

知识讲解

# 执行ad-hoc命令

- 创建实例并执行命令

```
tqm = None
try:
    tqm = TaskQueueManager(
            inventory=inventory,
            variable_manager=variable_manager,
            loader=loader,
            options=options,
            passwords=passwords,
)
    result = tqm.run(play)
finally:
    if tqm is not None:
        tqm.cleanup()
  shutil.rmtree(C.DEFAULT_LOCAL_TMP, True)
```

知识讲解

# 案例3：执行ad-hoc命令

1. 编写ansible脚本
2. 用于在远程主机执行任意命令

课堂练习

# 调用playbook

# Playbook编程概述

- Playbooks 是 Ansible的配置、部署、编排语言
- 它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
- 可以通过python编程的方式执行playbook

知识讲解

# 相关模块

- 与执行ad-hoc命令一样，playbook的执行也需要相关模块

```
from collections import namedtuple
from ansible.parsing.dataloader import DataLoader
from ansible.vars.manager import VariableManager
from ansible.inventory.manager import InventoryManager
from ansible.executor.playbook_executor import PlaybookExecutor
```

知识讲解

# 创建元组

- Playbook的参数更多，元组更大

```
Options = namedtuple('Options',
  [‘connection', 'remote_user',
   ‘ask_sudo_pass', 'verbosity',
   ‘ack_pass', 'module_path',
   ‘forks', 'become',
   ‘become_method', 'become_user',
   'check', 'listhosts',
   'listtasks',
   'listtags',
   'syntax',
   'sudo_user', 'sudo',
   'diff'])
```

知识讲解

# 创建元组（续1）

- Playbook的参数更多，元组更大

```
ops = Options(connection='smart',
    remote_user=None,  ack_pass=None,
    sudo_user=None, forks=5,
    sudo=None, ask_sudo_pass=False,
    verbosity=5, module_path=None,
    become=None, become_method=None,
    become_user=None, check=False,
    diff=False, listhosts=None,
    listtasks=None,
    listtags=None,
    syntax=None)
```

知识讲解

# 创建其他相关参数



知识讲解

```
loader = DataLoader()
passwords = dict()
inventory = InventoryManager(loader=loader, sources=['hosts'])
variable_manager = VariableManager(loader=loader,
inventory=inventory)
```

# 执行playbook



知识讲解

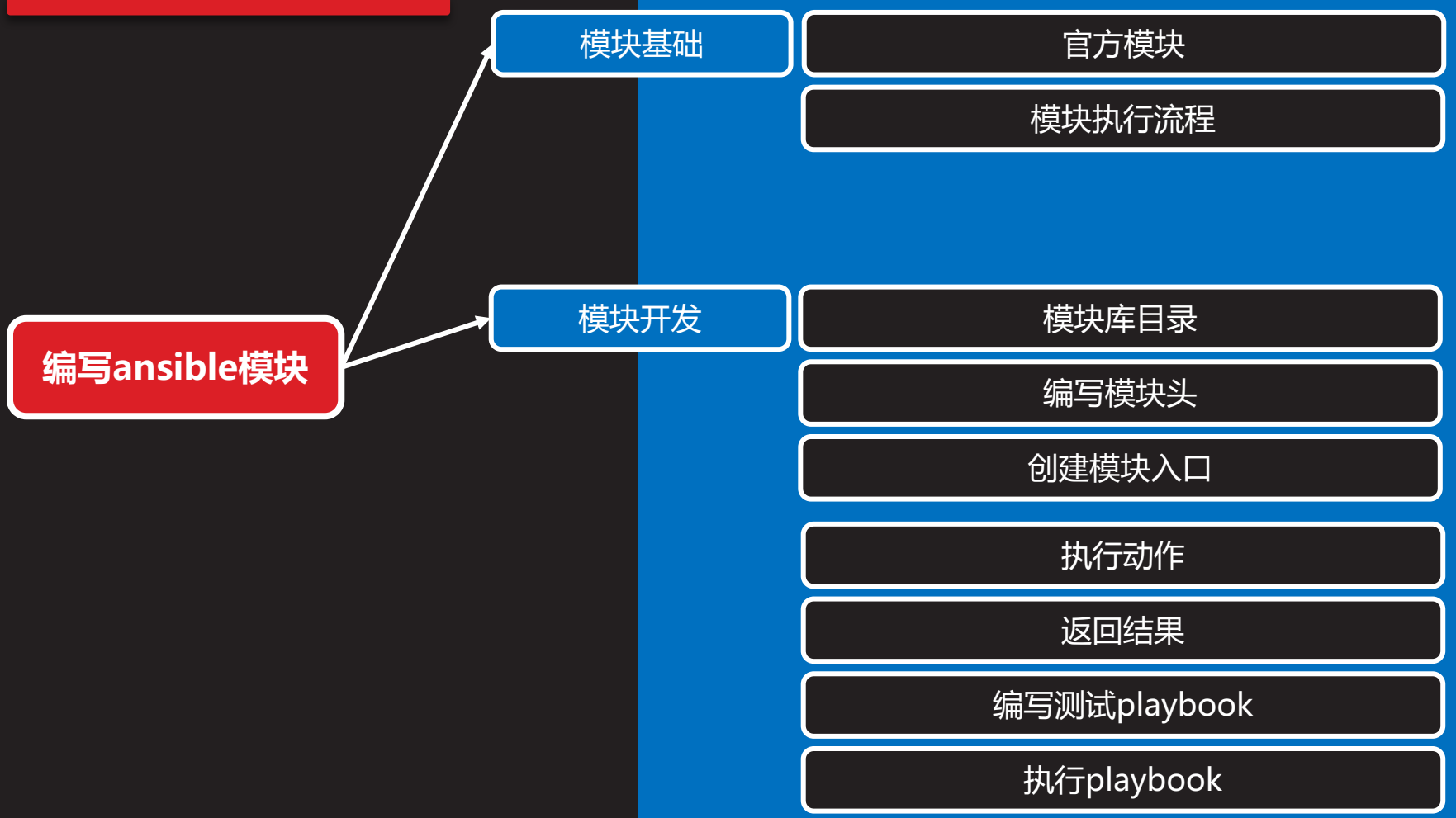
```
def playbookrun(playbook_path):
    playbook = PlaybookExecutor(playbooks=playbook_path,
                                inventory=inventory,
                                variable_manager=variable_manager,
                                loader=loader, options=ops, passwords=passwords)
    result = playbook.run()
    return result

if __name__ == '__main__':
    playbookrun(playbook_path=['install_web.yml'])
```

# 案例4：playbook编程

1. 编写python程序
2. 利用该程序执行前面课程中的playbook

课堂练习

# 编写ansible模块



Tedu.cn
达内教育

# 模块基础

# 官方模块

- Ansible官方已经提供了大量模块，在编写模块之前，可以查看是否已有现成模块
- 官方已发布模块
  - http://docs.ansible.com/ansible/modules.html
- 官方正在开发的模块
  - https://github.com/ansible/ansible/labels/module

知识讲解

# 模块执行流程

- 将模块文件读入内存，然后添加传递给模块的参数，最后将模块中所需要的类添加到内存，由zipfile压缩后，再由base64进行编码，写入到模板文件内
- 通过默认的连接方式（一般是ssh），ansible连接到远程主机，创建临时目录，并关闭连接
- 打开另外一个ssh连接，将模板文件以sftp方式传送到刚刚创建的临时目录中，写完后关闭连接

知识讲解

# 模块执行流程（续1）



知识讲解

- 打开一个ssh连接将任务对象赋予可执行权限，执行成功后关闭连接
- 最后，ansible将再打开一个新连接来执行模块，并删除临时目录及其所有内容
- 模块的结果是从标准输出stdout中获取json格式的字符串。ansible将解析和处理此字符串

Tedu.cn
达内教育

# 模块开发

# 模块库目录

- 可以使用 ANSIBLE_LIBRARY环境变量来指定模块的存放位置
- 也可以在playbook当前目录下创建library目录

```
[root@localhost myansi]# mkdir library/
```

知识讲解

# 编写模块头



知识讲解

- 在文件头部加入下列语句，表示该模块使用python运行

```
#!/usr/bin/env python
```

- 导入所需要的模块，如shutil模块

```
import shutil
from ansible.module_utils.basic import AnsibleModule
```

# 创建模块入口

- 使用AnsibleModule类中的argument_spec来接受参数

```
def main():
  module = AnsibleModule(
    argument_spec = dict(
    source=dict(required=True, type='str'),
    dest=dict(required=True, type='str')
  )
)
```

知识讲解

# 执行动作

- 使用shutil.copy拷贝文件

```
hutil.copy(module.params['source'], module.params['dest'])
```

知识讲解

# 返回结果

- 拷贝完成后，返回json数据

```
module.exit_json(changed=True)
```

- 编写主程序代码

```
if __name__ == '__main__':
    main()
```

知识讲解

# 编写测试playbook

- 编写playbook，测试编写的模块

```
[root@localhost myansi]# vim mytest.yml
---
- name: test remote_copy module
  hosts: myself

  tasks:
  - name: remote_copy file
    remote_copy:
      source: /etc/hosts
      dest: /tmp/zj.txt
```

知识讲解

# 执行playbook



- 执行playbook

```
[root@localhost myansi]# ansible-playbook mytest.yml
PLAY [test remote_copy module]
**********************************************************

TASK [Gathering Facts]
**********************************************************************

ok: [localhost]
TASK [remote_copy file]
**********************************************************************

changed: [localhost]

PLAY RECAP
**********************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0
```

知识讲解

# 案例5：ansible模块开发

1. 编写ansible模块，使用shutil模块拷贝文件
2. 数据源用变量名yuan
3. 数据目标变量用mudi

课堂练习

Tedu.cn
达内教育

# 总结和答疑