Tedu.cn
达内教育

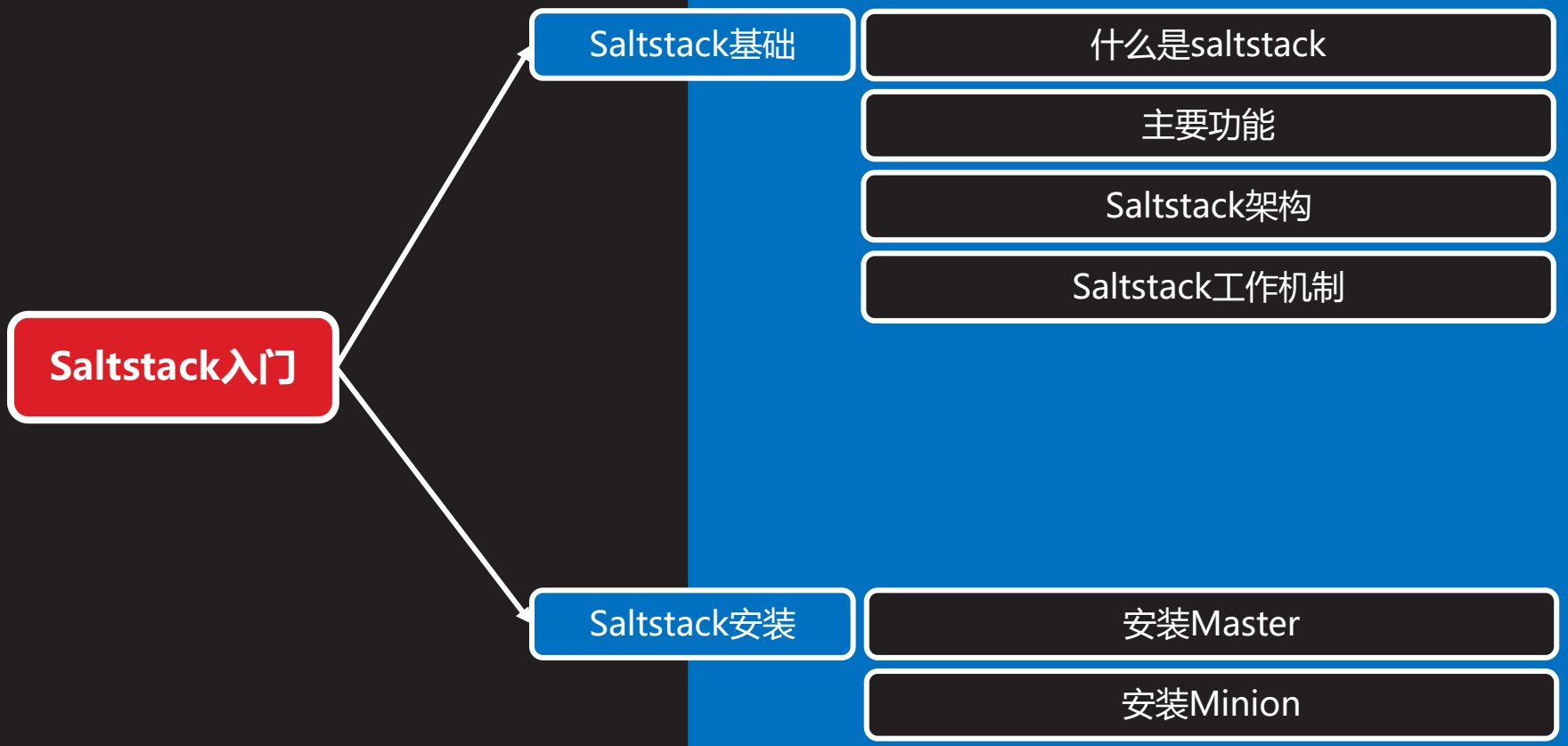
大型架构及配置技术

NSD ARCHITECTURE DAY03

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	Saltstack入门
	10:30 ~ 11:20	基本配置与使用
	11:30 ~ 12:20	
下午	14:00 ~ 14:50	Grains组件
	15:00 ~ 15:50	Pillar组件
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑

Saltstack入门



Tedu.cn
达内教育

Saltstack基础

什么是saltstack

- Saltstack是基于python开发的一套C/S架构配置管理工具
- 使用SSL证书签方的方式进行认证管理
- 底层使用ZeroMQ消息队列pub/sub方式通信
  - 号称世界上最快的消息队列ZeroMQ能快速在成千上万台主机上进行各种操作
  - 采用RSA Key方式确认身份，传输采用AES加密，使得它的安全性得到了保证

知识讲解

主要功能

- Saltstack最主要的两个功能是：配置管理与远程执行
- Saltstack不只是一个配置管理工具，还是一个云计算与数据中心架构编排的利器
- Saltstack已经支持Docker相关模块
- 在友好地支持各大云平台之后，配合Saltstack的Mine实时发现功能可以实现各种云平台业务的自动扩展

知识讲解

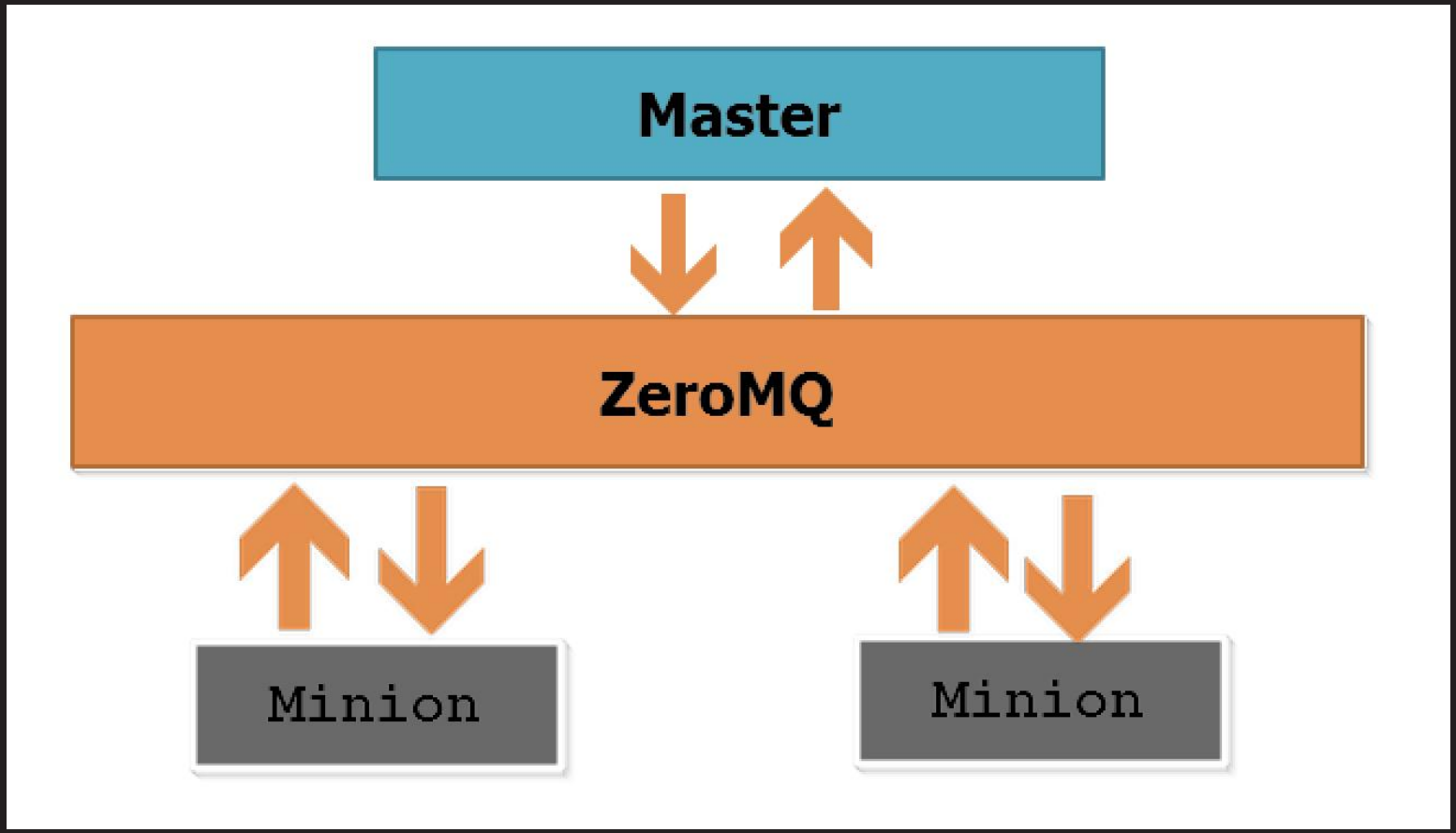
Saltstack架构

- Saltstack基于C/S架构
  - 服务器端称作Master
  - 客户端称作Minion
- 可以实现传统处理方式，即：客户端发送请求给服务器，服务器收到请求后处理请求，再将结果返回
- 也可以使用消息队列中的发布与订阅（pub/sub）服务模式

知识讲解

Saltstack架构（续1）

知识讲解



Saltstack工作机制

- Master和Minion都以守护进程的方式运行
- Master监听配置文件里定义的ret_port（接收minion请求），和publish_port（发布消息）的端口
- 当Minion运行时，它会自动连接到配置文件里定义的Master地址ret_port端口进行连接认证
- 当Master和Minion可以正常通信后，就可以进行各种各样的配置管理工作了

知识讲解

Saltstack安装

安装Master

知识讲解

- 采用在线安装方式

  [root@vh01 ~]# rpm -Uvh http://ftp.linux.ncsu.edu/pub/epel/epel-release-latest-7.noarch.rpm
  [root@vh01 ~]# yum install -y salt-master

- 采用提前下载的软件包

  [root@vh01 salt-master]# yum install -y *.rpm

安装Minion

- 采用在线安装方式

  [root@vh02 ~]# rpm -Uvh http://ftp.linux.ncsu.edu/pub/epel/epel-release-latest-7.noarch.rpm
  [root@vh01 ~]# yum install -y salt-minion

- 采用提前下载的软件包

  [root@vh01 salt-minion]# yum install -y *.rpm

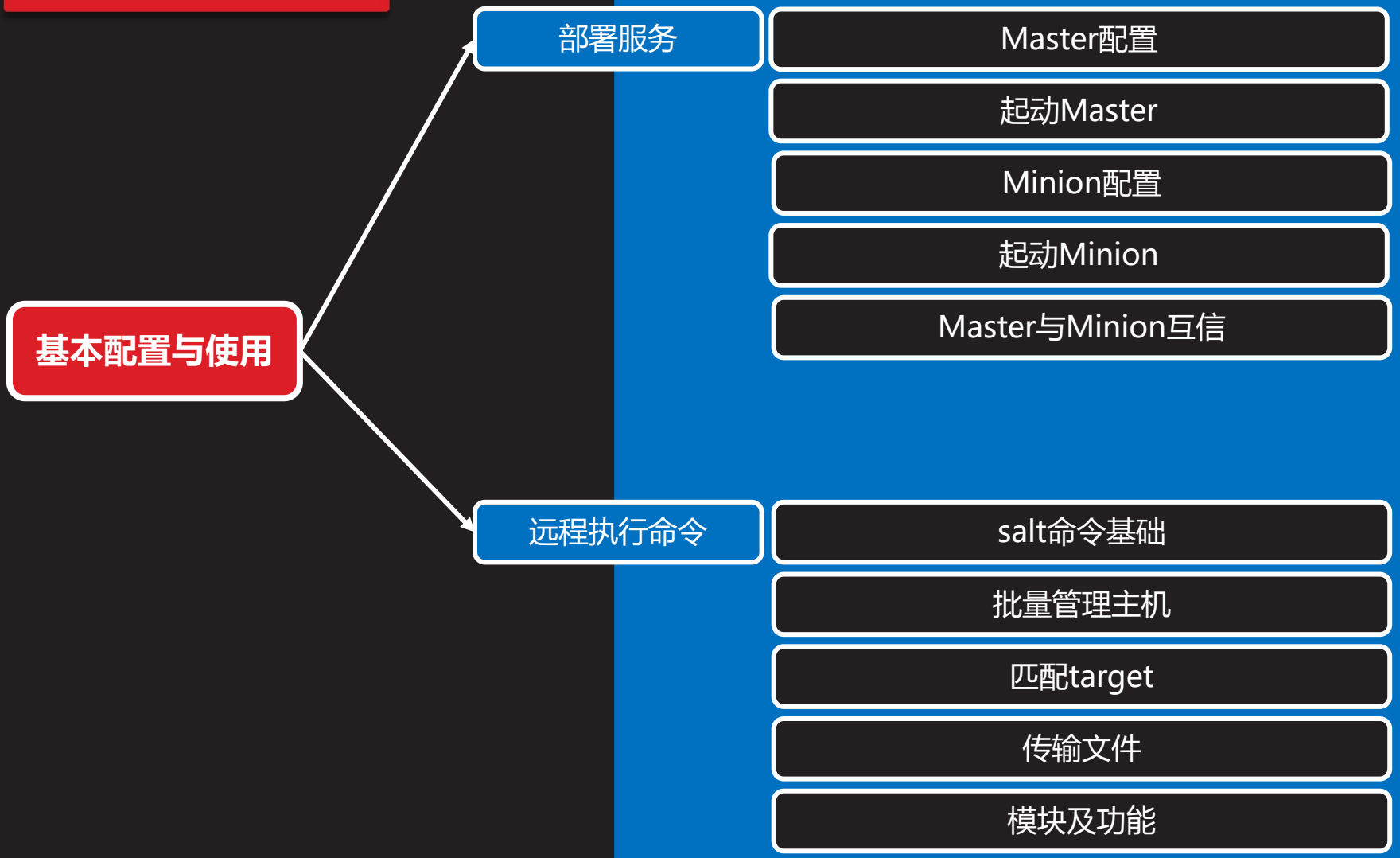
知识讲解

案例1：安装saltstack架构

1. 准备两台虚拟机
2. 配置好主机名和yum
3. 调通网络
4. 第一台安装Master和Minion
5. 第二台只安装Minion

课堂练习

基本配置与使用


基本配置与使用
部署服务
Master配置
起动Master
Minion配置
起动Minion
Master与Minion互信
远程执行命令
salt命令基础
批量管理主机
匹配target
传输文件
模块及功能

Tedu.cn
达内教育

服务配置

Master配置

- 修改/etc/hosts，实现名称解析

```
[root@vh01 ~]# tail -2 /etc/hosts
192.168.113.172 vh01.tedu.cn    vh01
192.168.113.173 vh02.tedu.cn    vh02
```

- Master主配置文件是/etc/salt/master，常用的配置选项与具体功能相关，所以，当应用到相关功能时再做调整，目前保持默认就好

知识讲解

起动Master

- 起动Master

```
[root@vh01 ~]# systemctl enable salt-master.service
[root@vh01 ~]# systemctl start salt-master.service
```

- 验证服务

```
[root@vh01 ~]# systemctl status salt-master.service
[root@vh01 ~]# netstat -tlnp | egrep ':4505|:4506'
```

知识讲解

Minion配置

- 修改/etc/hosts，实现名称解析

```
[root@vh01 ~]# tail -2 /etc/hosts
192.168.113.172 vh01.tedu.cn    vh01
192.168.113.173 vh02.tedu.cn    vh02
```

- 修改Minion配置文件，使其可以与Master通信

```
[root@vh01 ~]# vim /etc/salt/minion
master: vh01.tedu.cn
id: vh01.tedu.cn
```

知识讲解

起动Minion

知识讲解

- 起动Minion

```
[root@vh01 ~]# systemctl enable salt-minion.service
[root@vh01 ~]# systemctl start salt-minion.service
```

- 验证服务

```
[root@vh01 ~]# systemctl status salt-minion.service
```

Master与Minion互信

- Minion上线后先与Master端联系，把自己的pub key发过去
- Master接受Minion的公钥后，互信建立完成
- 查看密钥信息

```
[root@vh01 ~]# salt-key -L
Accepted Keys:
Denied Keys:
Unaccepted Keys:
vh01.tedu.cn
vh02.tedu.cn
Rejected Keys:
```

知识讲解

Master与Minion互信（续1）

知识讲解

- 密钥管理

```
[root@vh01 ~]# salt-key -h
        -L: 列出密钥
        -a: 接受一个密钥
        -A: 接受全部密钥
        -D: 删除全部密钥
        … …
```

- 接受密钥，完成互信

```
[root@vh01 ~]# salt-key -A -y
```

案例2：配置Saltstack

1. 配置并起动第一台主机的Master服务
2. 在两台主机上分别配置并起动Minion
3. 在Master上接受Minion的密钥，完成互信

课堂练习

Tedu.cn
达内教育

远程执行命令

salt命令基础

- salt命令使用方法如下

  salt [options] '<target>' <function> [arguments]

- target指的是在哪些Minion上执行，如果在全部Minion上运行，可以采用通配符 '*'
- function一般采用python的 模块.方法 样式
- arguments是传递给方法的参数

知识讲解

批量管理主机

- 测试所有主机连通性

```
[root@vh01 ~]# salt '*' test.ping
vh02.tedu.cn:
    True
vh01.tedu.cn:
    True
```

- 在主机上执行任意命令

```
[root@vh01 ~]# salt '*' cmd.run 'uname -r'
vh01.tedu.cn:
    3.10.0-327.el7.x86_64
vh02.tedu.cn:
    3.10.0-327.el7.x86_64
[root@vh01 ~]# salt 'vh02.tedu.cn' cmd.run 'uptime'
vh02.tedu.cn:
    15:27:01 up 57 min,  3 users,  load average: 0.00, 0.03, 0.05
```

知识讲解

匹配target

- 使用正则表达式

```
[root@vh01 ~]# salt -E 'vh\d+' test.ping
vh01.tedu.cn:
    True
vh02.tedu.cn:
    True
```

- 使用列表

```
[root@vh01 ~]# salt -L vh01.tedu.cn,vh02.tedu.cn test.ping
vh01.tedu.cn:
    True
vh02.tedu.cn:
    True
```

知识讲解

匹配target（续1）

- 使用组匹配（首先在Master中定义组）

```
[root@vh01 ~]# vim /etc/salt/master
nodegroups:
  group1: 'L@vh01.tedu.cn,vh02.tedu.cn'
[root@vh01 ~]# systemctl restart salt-master.service
[root@vh01 ~]# salt -N group1 test.ping
vh01.tedu.cn:
    True
vh02.tedu.cn:
    True
```

知识讲解

匹配target（续2）

- 使用CIDR匹配

```
[root@vh01 ~]# salt -S '192.168.113.0/24' test.ping
vh01.tedu.cn:
    True
vh02.tedu.cn:
    True

[root@vh01 ~]# salt -S '192.168.113.173' test.ping
vh02.tedu.cn:
    True
```

知识讲解

案例3：远程执行命令

- 练习salt相关命令
  1. 在远程主机上执行test.ping
  2. 在远程主机上通过cmd.run执行任意命令
  3. 练习target各种匹配方法

课堂练习

传输文件

- Master默认的根目录是/srv/salt

```
[root@vh01 ~]# mkdir -p /srv/salt/files
```

- 向group1组传输文件

```
[root@vh01 ~]# salt -N group1 cp.get_file salt://files/hosts /tmp/zhuji
vh02.tedu.cn:
    /tmp/zhuji
vh01.tedu.cn:
    /tmp/zhuji
```

知识讲解

案例4：传输文件

1. 建立待传输文件的基础目录
2. 拷贝/etc/passwd到传输文件目录
3. 将passwd拷贝到远程Minion的/tmp目录下，并将文件重命名为mima

课堂练习

模块及功能

- 列出所有可用模块

  [root@vh01 ~]# salt 'vh01.tedu.cn' sys.list_modules

- 查看模块所有功能

  [root@vh01 ~]# salt 'vh01.tedu.cn' sys.list_functions test

- 查看模块用法

  [root@vh01 ~]# salt 'vh01.tedu.cn' sys.doc test

知识讲解

模块及功能（续1）

- 拷贝任意文件和目录

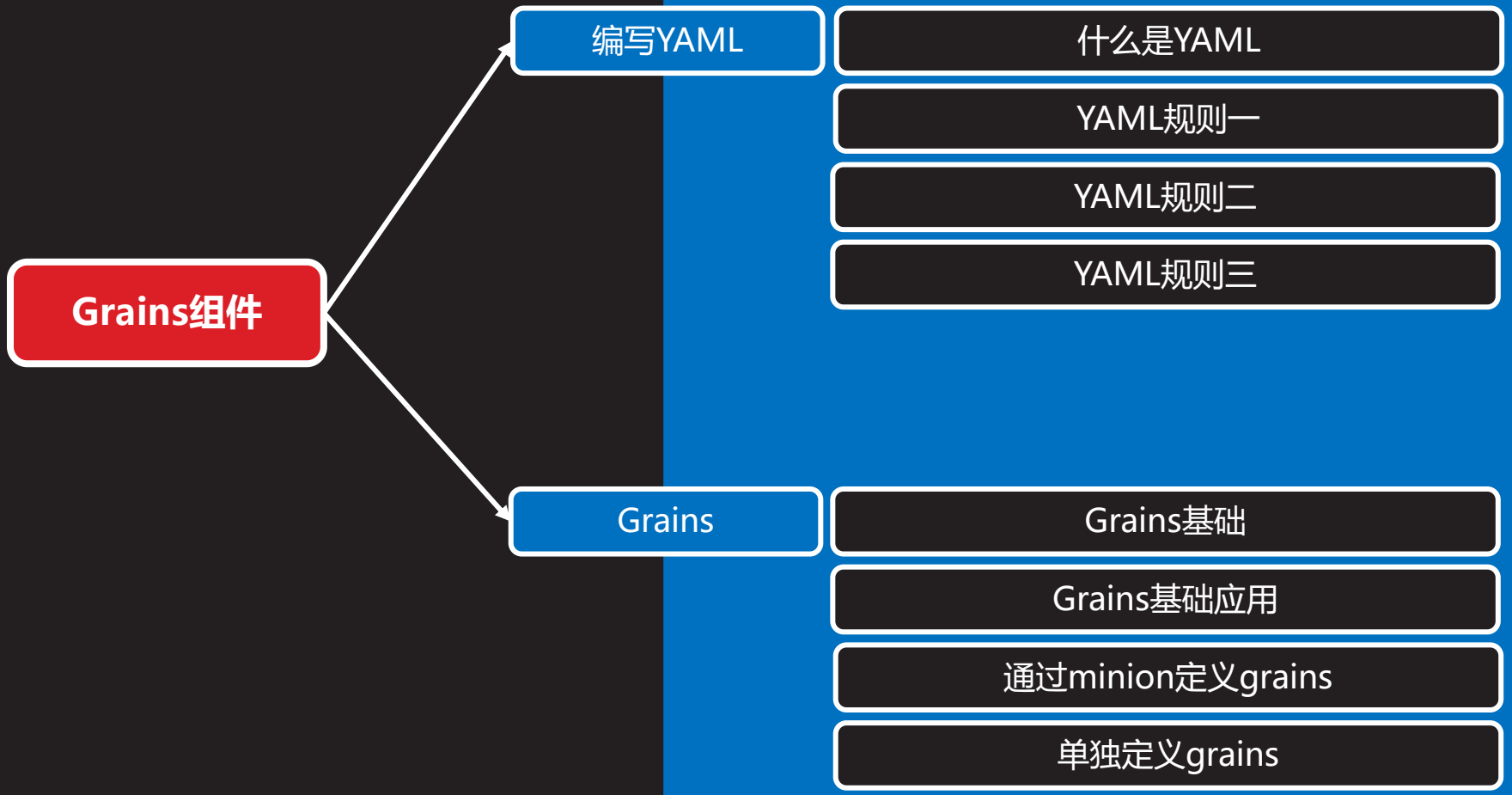
```
[root@vh01 ~]# salt -N group1 file.copy /etc/selinux/ /tmp/selinux
recurse=True
```

- 创建用户

```
[root@vh01 ~]# salt -N group1 user.add zhangzg 2000
```

知识讲解

Grains组件


Grains组件
编写YAML
什么是YAML
YAML规则一
YAML规则二
YAML规则三
Grains
Grains基础
Grains基础应用
通过minion定义grains
单独定义grains

Tedu.cn
达内教育

编写YAML

什么是YAML

- YAML：YAML Ain't Markup Language
- YAML的结构通过空格来展示
- 项目使用"-"来表示
- 键值对使用":"来表示
- Master和Minion的配置文件均采用YAML语法

知识讲解

YAML规则—

- 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空格组成
- 注意不要使用tab
- 缩进是初学者容易出错的地方之一

知识讲解

YAML规则二

知识讲解

- YAML的键值对采用冒号分隔
- YAML键值对对应python的字典
- YAML表示形式

```
name: vh01
或
name:
  vh01
```

- Python字典

```
{'name': 'vh01'}
```

YAML规则二（续1）

知识讲解

- 字典可以嵌套

```
hosts:
  name: vh01
```

- 字典表示形式为

```
{
    'hosts': {
    'name': 'vh01'
  }
}
```

YAML规则三

知识讲解

- 列表项使用一个短横杠加一个空格

```
- vh01.tedu.cn
- vh02.tedu.cn
- vh03.tedu.cn
```

- 列表可以作为一个键值对的value

```
pkg-http:
  - httpd
  - php
```

- Python语法

```
{'pkg-http': ['httpd', 'php']}
```

Grains

Grains基础

- Grains是saltstack最重要的组件之一
- 存储minion端的基本信息，这些信息一般都是静态的，如CPU、内核、操作系统等
- Grains存储在minion本地
- 管理员可以在minion端进行grains值的修改，如增加、删除等

知识讲解

Grains基础应用

知识讲解

- 获取minion端所有grains信息

```
[root@vh01 ~]# salt 'vh02.tedu.cn' grains.items
```

- 通过grains.item获取minion端的fqdn信息

```
[root@vh01 ~]# salt 'vh02.tedu.cn' grains.item fqdn
vh02.tedu.cn:
    ----------
    fqdn:
        vh02.tedu.cn
```

- 匹配minion端OS为CentOS的执行uptime命令

```
[root@vh01 ~]# salt -G 'os:CentOS' cmd.run 'uptime'
```

通过minion定义grains

知识讲解

- 定义角色

```
[root@vh02 salt]# vim /etc/salt/minion
grains:
  role: webserver
[root@vh02 salt]# systemctl restart salt-minion.service
```

- 查看角色信息

```
[root@vh01 ~]# salt 'vh02.tedu.cn' grains.item role
vh02.tedu.cn:
    ----------
    role:
        webserver
```

单独定义grains

知识讲解

- 创建grains配置文件

```
[root@vh01 ~]# vim /etc/salt/minion.d/grains.conf
grains:
  deployment: datacenter-4
```

- 刷新grains配置

```
[root@vh01 ~]# salt 'vh01.tedu.cn' saltutil.sync_grains
```

- 查看grains信息

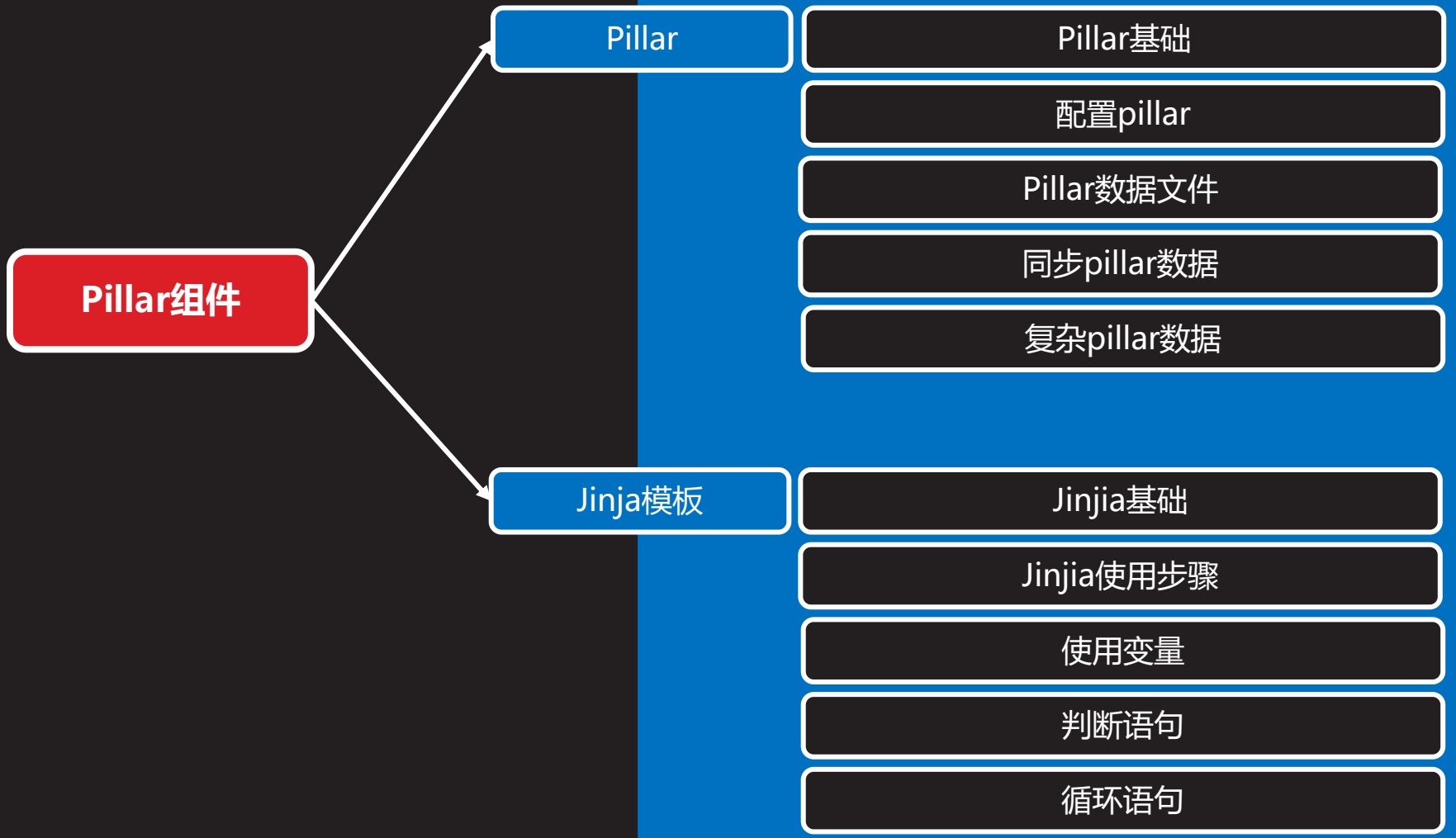
```
[root@vh01 ~]# salt 'vh01.tedu.cn' grains.item deployment
vh01.tedu.cn:
    ----------
    deployment:
        datacenter-4
```

案例5：定义grains

1. 在第二台minion的minion文件中定义role属性
2. 在第一台minion上新建grains.conf，定义deployment属性
3. 在Master上获取两台minion上的相关属性

课堂练习

Pillar组件


Pillar组件
Pillar
Pillar基础
配置pillar
Pillar数据文件
同步pillar数据
复杂pillar数据
Jinja模板
Jinjia基础
Jinjia使用步骤
使用变量
判断语句
循环语句

Pillar

Pillar基础

- Pillar也是saltstack最重要的组件之一
- 作用是定义与被控主机相关的任何数据，定义好的数据可以被其他组件使用
- 存储在master端，存放需要提供给minion的信息
- 常用于敏感信息，每个minion只能访问master分配给自己的pillar信息
- 用于经常动态变化的信息

知识讲解

配置pillar

- Pillar需要一个pillar_roots来维护pillar的配置
- 默认pillar_roots为/srv/pillar
- pillar_roots在Master配置文件中定义

```
[root@vh01 ~]# vim /etc/salt/master
pillar_roots:
  base:
    - /srv/pillar
[root@vh01 ~]# mkdir /srv/pillar
[root@vh01 ~]# systemctl restart salt-master.service
```

知识讲解

Pillar数据文件

- Pillar执行时需要一个名为top.sls的入口文件
- 通过top.sls文件作为入口，组织其它的pillar文件
- sls文件采用YAML格式

```
[root@vh01 ~]# cd /srv/pillar/
[root@vh01 pillar]# vim top.sls
base:                    # 与pillar_roots定义一致
  'vh02.tedu.cn':        # 过滤目标
    - data               # 用于包含data.sls
[root@vh01 pillar]# vim data.sls
appname: website
flow:
  maxconn: 3000
  maxmem: 6G
```

知识讲解

同步pillar数据

知识讲解

- 将pillar数据同步至minion

```
[root@vh01 pillar]# salt '*' saltutil.refresh_pillar
```

- 获取pillar全部数据

```
[root@vh01 pillar]# salt '*' pillar.items
```

- 获取指定数据

```
[root@vh01 pillar]# salt '*' pillar.item appname
```

- 根据pillar值匹配minion

```
[root@vh01 pillar]# salt -I 'appname:website' test.ping
```

复杂pillar数据

- 更为复杂的pillar数据可以放到目录的sls文件中

```
[root@vh01 pillar]# mkdir users
[root@vh01 pillar]# cat users/init.sls
users:
  john: 2000
  jane: 2001
  jack: 2002

[root@vh01 pillar]# cat top.sls
base:
  'vh02.tedu.cn':
    - data
    - users
```

知识讲解

案例6：配置pillar

课堂练习

1. 在master上声明pillar工作目录
2. 为第二台minion设置属性appname值为website，flow的值又包括子项maxconn为3000，maxmem值为6G
3. 配置第二台minion有三个用户，并指定UID

Jinja模板

Jinja基础

- Jinja是基于Python的模板引擎
- 在saltstack中我们使用yaml_jinja渲染器来根据模板生产对应的配置文件
- 对于不同的操作系统或者不同的情况，通过jinja可以让配置文件或者操作形成一种模板的编写方式

知识讲解

Jinja使用步骤

- 在state文件中使用"- template: jinja"声明
- 在模板文件中使用变量"{{ name }}"声明，name为变量，自己定义
- 在state文件中使用"- defautls: name: value"声明

知识讲解

使用变量

- 变量的基本格式为：{{ 变量 }}
- 一级字典变量格式为：{{ pillar['appname'] }}
- 二级字典变量格式为：{{ pillar['flow']['maxconn'] }}

知识讲解

判断语句

- 对grains的os值进行判断，根据不同的系统对apache的值进行不同的设定，这样apache就相当于是可以随机应变的值

```
{% if grains['os'] == 'CentOS' %}
apache: httpd
{% elif grains['os'] == 'Debian' %}
apache: apache2
{% endif %}
```

知识讲解

循环语句

- 在state中使用pillar数据，值通过Jinja来访问pillar即可，多个值通过循环逐个获取

```
{% for user, uid in pillar.get('users', {}).items() %}
{{user}}:
  user.present:
    - uid: {{uid}}
{% endfor %}
```

知识讲解

总结和答疑

Tedu.cn
达内教育

分发文件不成功

问题现象

- 当执行以下语句，进行分发文件时，minion端并没有得到文件

```
# salt -N group1 cp.get_file /etc/hosts /tmp/zhuji
```

知识讲解

故障分析及排除

- 原因分析
  - cp.get_file分发文件时，不能随意指定路径
- 解决办法
  - 将文件拷贝到/srv/salt/files/目录中
  - 通过以下命令进行分发

```
# salt -N group1 cp.get_file salt://files/hosts /tmp/zhuji
```

知识讲解