云科ADC&Ansible部署实践

Sector 1

Ansible基本概念

Sector 2

深入解析

Sector 3

Demo

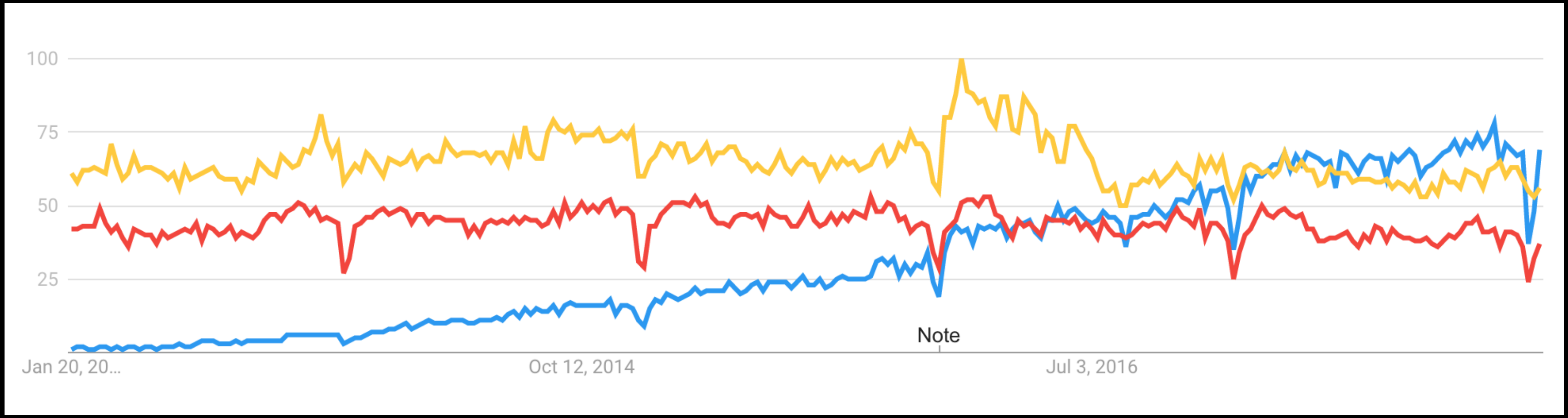
what is Ansible

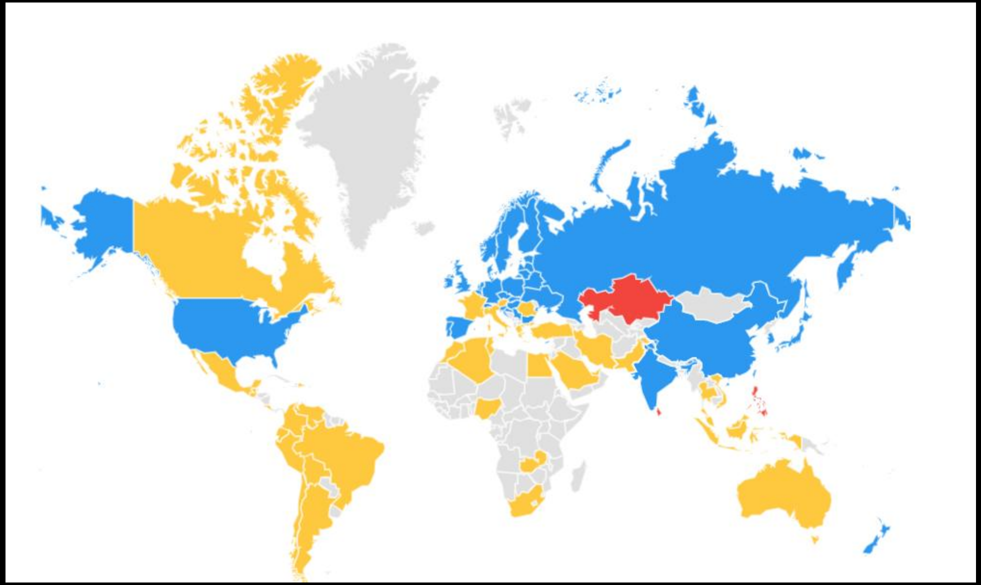
Ansible 是一套开源软件，负责配置管理，软件/应用自动化程序部署，云配置，多节点编排，初始撰写人Michael DeHaan,同时也是服务器部署软件 Cobbler 的作者，曾经是Ansible,inc的CTO，目前已经离职(喜欢做新东西)

Original author(s)	Michael DeHaan
Developer(s)	Ansible Community / Ansible Inc. / Red Hat Inc.
Initial release	February 20, 2012; 5 years ago
Stable release	2.4.2.0 / November 30, 2017
Repository	https://github.com/ansible/ansible, git://github.com/ansible/ansible.git
Development status	Active
Written in	Python, PowerShell
Operating system	Linux, Unix-like, Windows
Available in	English
Type	Configuration management, Infrastructure as Code, Orchestration engine
License	GNU General Public License
Website	www.ansible.com

Ansible,inc公司提供Ansible的商业化技术支持，同时是社区赞助者，2015年被RedHat收购，社区目前活跃开发人员大约70人左右，升级周期1-2个月
带GUI的商业化版本：Tower
100node 5x8 $10000/year7x24 $14000/year

Google Trend of Automtion Tool





● ansible
Search term

● puppet
Search term

● Chef
Search term

Ansible设计原则:

- 非常简单的设置过程和最小的学习曲线;
- 非常快速和平行地管理机器;

号称简单到发指，实际只是相对的

- 用机器和人性化的语言描述基础设施;
- 注重安全性和易于审计/审查/重写内容;
- 立即管理新的远程机器，无需引导任何软件;
- 允许使用任何动态语言进行模块开发，而不仅仅是Python;
- 目标成为最容易使用的IT自动化系统

无GUI界面，没有存储数据，是一种管理接口，而不是管理系统，ansible提供丰富的，并且可以置入逻辑的配置管理接口，一般客户WEB和DB需要自建

- YK对比其他厂商提供更多的模块以及功能
- YK 目前有95个开发level模块

AVI	(36)
Cisco	(11)
NetScaler	(1)
A10	(4)
Brocade	(0)
AWS	(~55)
Azure	(14)
HA Proxy	(1)

Ansible管理节点安装步骤

1.安装 ansible:

```
sudo apt-get install software-properties-common
sudo apt-add-repository ppa:ansible/ansible
sudo apt-get update
sudo apt-get install ansible
```

2. 安装Ansible中YK Module依赖包

虽然Ansible内置了YK的模块，但是这些模块调用了YK的SDK以及第三方的类库，需要额外安装

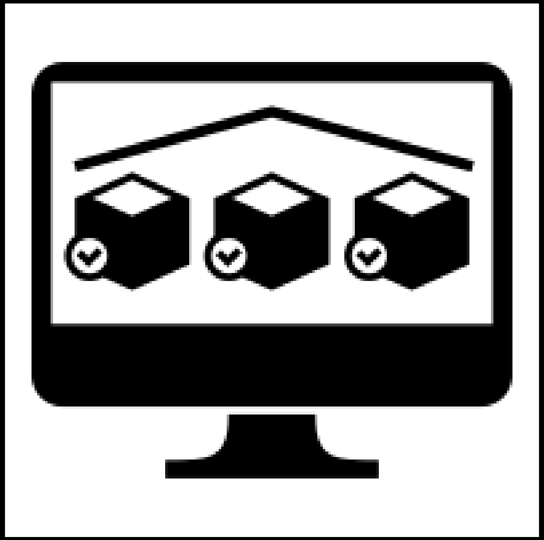
```
$ sudo pip install yk-sdk bigsuds netaddr deepdiff
```

强烈建议使用新版本ubuntu/Redhat/Centos,低版本存在Lib依赖关系混乱问题
YK建议V13以上,Ansible 2.2版本以上

Ansible构成

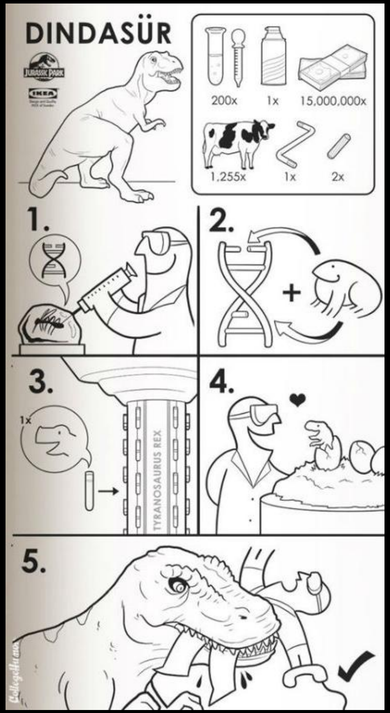
•Ansible 由三大模块构成:

Inventory
受控主机数据

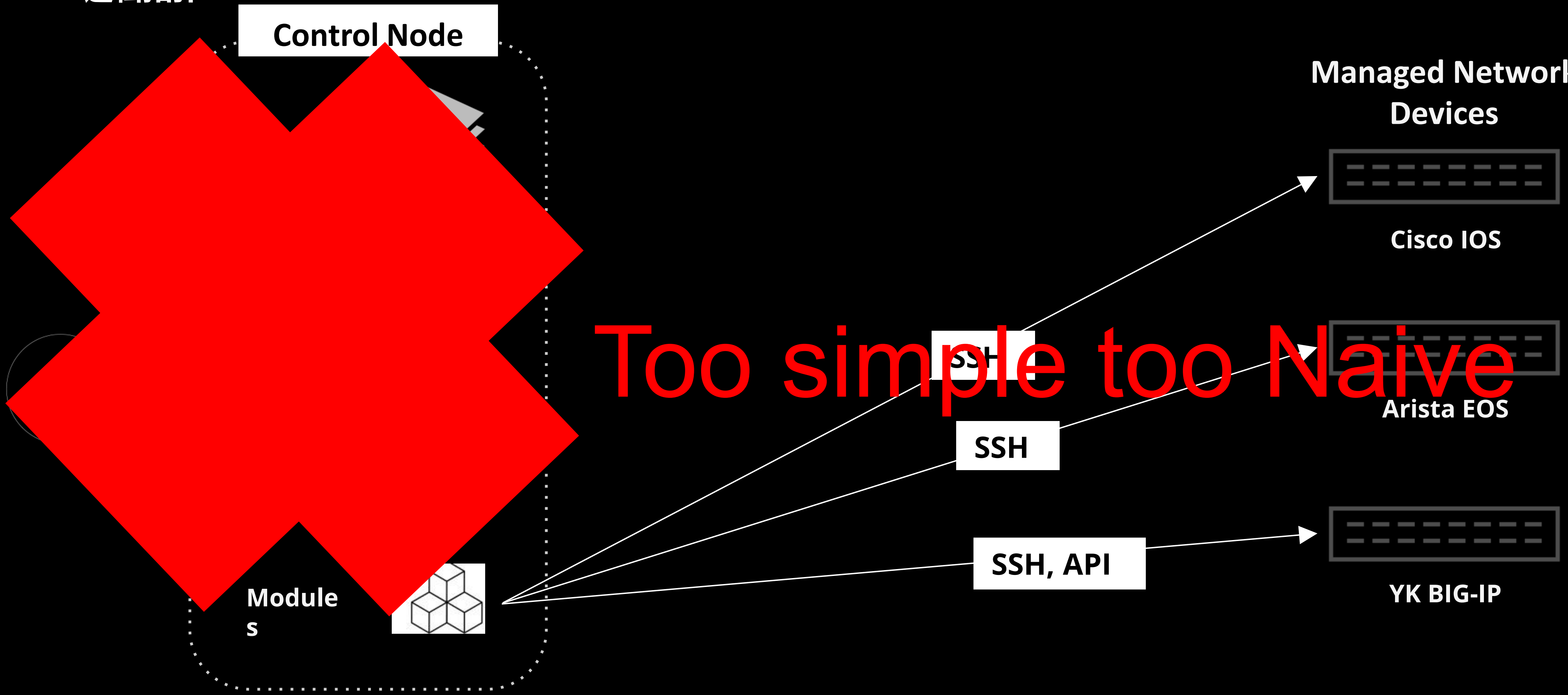


Modules
主机操控方法

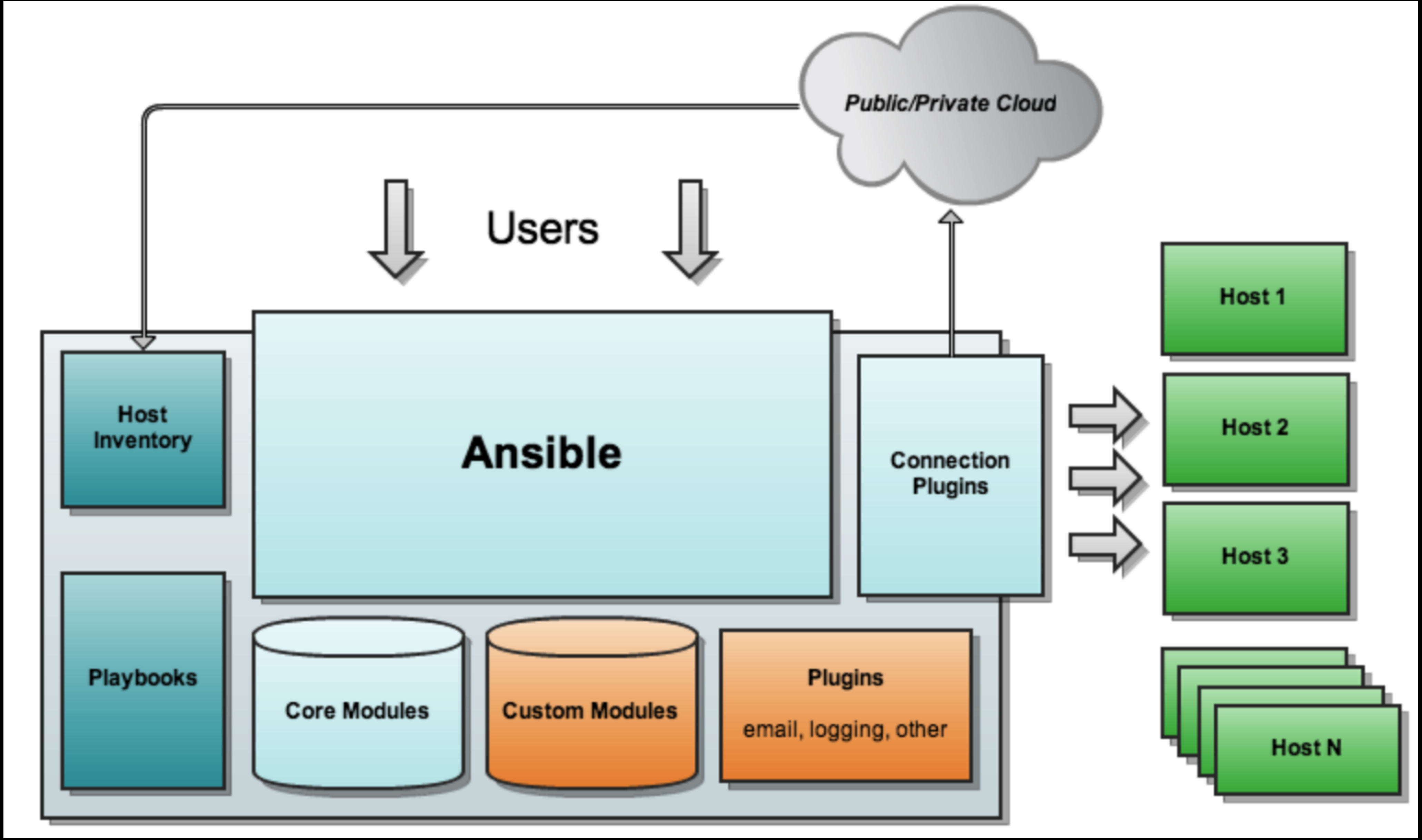
Playbooks
事务剧本



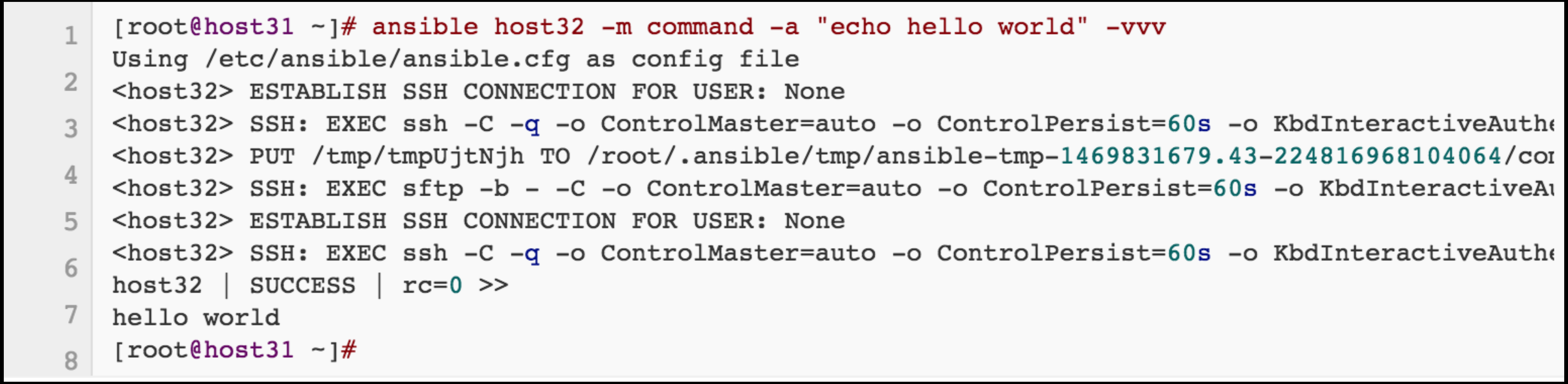
Ansible逻辑拓扑



ansible可以通过SSH和API对设备执行命令，采用何种方式，完全取决于你调用的模块以及模块实现方式，例如你可以用command Module（SSH）把YK当成linux设备执行命令，也可以用YK Module，使用Rest API/SOAP接口对设备进行远程操作。具体是什么类型接口落地，看调用方式


Public/Private Cloud
Users
Host Inventory
Ansible
Connection Plugins
Host 1
Host 2
Host 3
Host N
Playbooks
Core Modules
Custom Modules
Plugins
email, logging, other

Ansible 典型运行原理



```
[root@host31 ~]# ansible host32 -m command -a "echo hello world" -vvv
Using /etc/ansible/ansible.cfg as config file
<host32> ESTABLISH SSH CONNECTION FOR USER: None
<host32> SSH: EXEC ssh -C -q -o ControlMaster=auto -o ControlPersist=60s -o KbdInteractiveAuthe
<host32> PUT /tmp/tmpUjtNjh TO /root/.ansible/tmp/ansible-tmp-1469831679.43-224816968104064/co
<host32> SSH: EXEC sftp -b - -C -o ControlMaster=auto -o ControlPersist=60s -o KbdInteractiveAu
<host32> ESTABLISH SSH CONNECTION FOR USER: None
<host32> SSH: EXEC ssh -C -q -o ControlMaster=auto -o ControlPersist=60s -o KbdInteractiveAuthe
host32 | SUCCESS | rc=0 >>
hello world
[root@host31 ~]#
```

- 和opsware.expect不同，默认代码运行位置在受控主机上
- 将要执行的echo hello world放到了一个本地临时文件中
- 通过sftp传送到host的临时文件夹
- 通过ssh远程执行，执行之后清理现场将sftp传过去的文件删除
- 默认的执行方式适合主机服务器，大部分的第三方设备不采用这样的方式，比如Cisco，YK
- 可以采用pipeline模式，通过tty，不需要拷贝文件

Ansible默认工作模式Connection:ssh，上传到目标主机的是python文件，该文件根据引用模块和参数动态生成，特殊的模块需要目标主机上有依赖库支持

Ansible使用方式

Ansible提供两种方式去完成任务,一是 ad-hoc 命令,一是写 Ansible playbook.前者可以解决一些简单的任务, 后者解决较复杂的任务.

ad-hoc模式执行命令：/usr/bin/ansible，适用简单场合，例如临时规划的启停服务，联通性测试等

ansible <pattern_goes_here> -m <module_name> -a <arguments>

用例：ansible webservers -m service -a "name=httpd state=restarted"

Ansible同时对多台设备并发多线程执行操作，在ad-hoc模式下通过-f参数指定并发线程数
ad-hoc命令默认模块是command

playbook模式执行命令：/usr/bin/ansible-playbook

ansible-play yamlfile
此时在yaml文件中指定批量化的操作，以及每个操作使用的模块，参数，调用方式等等

运行 ad-hoc命令

```
[jrodrigues@fedora ansible-howto]$ ansible yk-12x -i inventory -a "tmsh show sys software"
bigip1.vlab.local | SUCCESS | rc=0 >>

---------------------------------------------------
Sys::Software Status
Volume  Product  Version    Build  Active    Status
---------------------------------------------------
HD1.1    BIG-IP   12.1.1  2.0.204     yes  complete

---------------------------
Sys::Software Update Check
---------------------------
  Check Enabled         true
  Phonehome Enabled     true
  Frequency           weekly
  Status                none
  Errors                   0

bigip2.vlab.local | SUCCESS | rc=0 >>

---------------------------------------------------
Sys::Software Status
Volume  Product  Version    Build  Active    Status
---------------------------------------------------
HD1.1    BIG-IP   12.1.1  2.0.204     yes  complete

---------------------------
Sys::Software Update Check
---------------------------
  Check Enabled         true
  Phonehome Enabled     true
  Frequency           weekly
  Status                none
  Errors                   0
```

需要预先配置SSH授信关系

Sector 1

Ansible基本概念

Sector 2

深入解析

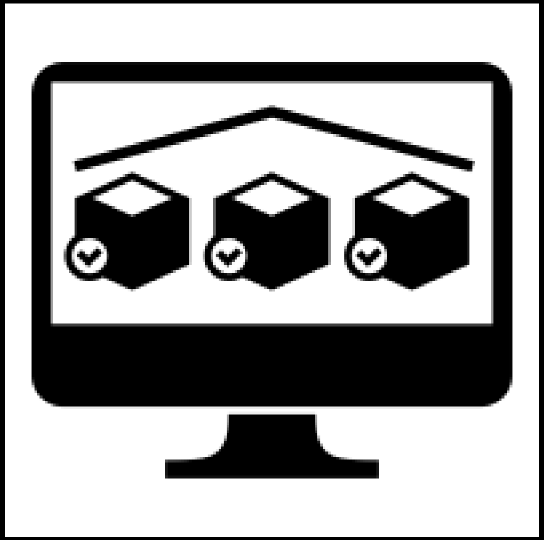
Sector 3

Demo

Ansible构成

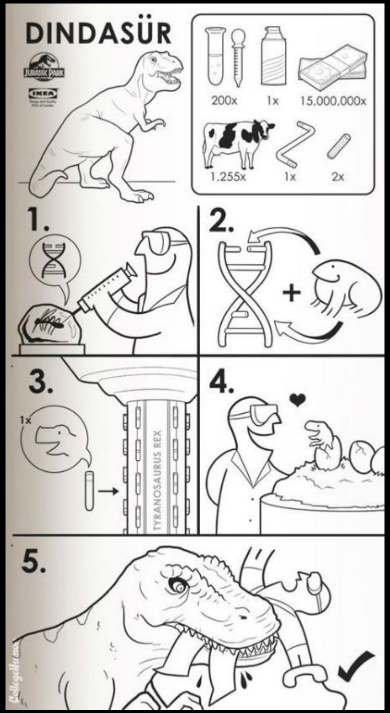
- Ansible 由三大模块构成:

Inventory
受控主机数据



Modules
主机操控方法

Playbooks
事务剧本



Hosts Inventory

Inventory 是一个text文件，包括所有被Ansible所管理的主机清单，可以写入IP地址或者主机名称，如是主机名称需要可以被静态或者动态解析

- 默认库存文件保存位于`/etc/ansible/hosts`
- 变量可以应用于inventory
- 主机可以被分组，通常来说都会被分组;
- 一个主机分组可以做为另一个组的成员;
- 未在inventory中定义的主机无法执行远程控制，ansible不支持直接采用本机host或本机DNS，在Playbook和ad-hoc下可以使用主机名或者组名(group)

inventory可以使用动态或者是静态的

- **静态: 不可变的text file**
- **动态:除了本地文件，ansible支持从第三方获取到inventory数据，例如在云化环境下auto scale 主机数量是动态变化，ansible可以通过多种方式获取外部的数据并且映射到inventory文件分组中。目前支持的云以及配置管理工具有Cobbler,EC2,BSD Jails,DigitalOcean,Google Compute Engine,Linode,OpenShift,OpenStack Nova,Red Hat‘s SpaceWalkVagrant，Zabbix**

主机Inventory文件范例

Inventory 例子

```
[all]
bigip1.vlab.local
bigip2.vlab.local
bigip3.vlab.local
centos7.vlab.local

[all:vars]
ansible_ssh_user=root
ansible_ssh_private_key_file=~/.ssh/id_rsa

# YK Big-IP Version 12
[YK-12x]
bigip1.vlab.local
bigip2.vlab.local

# YK Big-IP Version 11
[YK-11x]
bigip3.vlab.local

# All YK Big-IP
[YK:children]
YK-12x
YK-11x

# All CentOS 7
[centos7]
centos7.vlab.local
```

联通性测试命令

```
[jrodrigues@fedora ansible-howto]$ ansible all -i inventory -m ping
centos7.vlab.local | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
bigip3.vlab.local | FAILED! => {
    "changed": false,
    "failed": true,
    "msg": "Error: ansible requires the stdlib json or simplejson
module, neither was found!"
}
bigip1.vlab.local | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
bigip2.vlab.local | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Modules

Ansible modules 是控制资源的工具，包含服务，类库包，文件，以及可执行系统命令等等

•也被称为task plugin，可以通过ansible命令调用，也可以在playbook中被调用
•Modules 典型来说通过Pyhon撰写，使用ansible 接口提供API抽象
•Modules 支持Key - Value格式的参数输入，以空格做分隔
•Modules 返回Json格式结数据 ，通常可以被忽略，自定义的模块可以通过任何语言编写
•Modules 努力使自身幂等，如果目标主机配置已经和操作目的一致，不做任何修改，和收敛性相对
•当使用 Ansible playbooks 时,这些模块能够触发 ‘change events’,以这种形式通知 ‘handlers’ 去运行附加任务
•每一个模块都必须返回一个status, 来表示这个模块是成功的,是否有任何改变或没有
•直接调用模块命令：

```
ansible webservers -m service -a "name=httpd state=started"
```

列出目前所有模块以及模块的文档说明:

•列出所有模块: `ansible-doc -l`
•查看某模块的文档: `ansible-doc MODULE_NAME`

Modules

- 下面的模块类型可以认为是‘Ansible native’:
  - Commands (**`command, expect, raw, script, shell`**);
  - Inventory (**`add_host, group_by`**);
  - System (specifically **`setup`** to gather Facts about hosts, and **`ping`**);
  - Utilities (**`assert, async_status, debug, include, include_role, include_vars`**, etc).

- 其他模块归结于特定的领域

Cloud, Clustering, Crypto, Database, Files, Identity, Messaging, Monitoring, Network, Notification, Packaging, Remote Management, Source Control, Storage, System, Web Infrastructure, Windows.

- 有数以百计的来源于不同厂商的模块，以支持不同类型的设备以及服务：

Generic UNIX/Linux, Red Hat, Debian, Linux KVM Libvirt, Docker, Kubernetes, OpenStack, OpenvSwitch, Jenkins, YK Networks, Arista, VMWare, Cisco, Juniper, Amazon, NetApp, MySQL, Apache, Microsoft, etc.

Modules

•通过Ansible 原生setup模块获取yk设备的基础信息.

```
[jrodrigues@fedora ansible-howto]$ ansible
bigip1.vlab.local -i inventory -m setup
bigip1.vlab.local | SUCCESS => {
    "ansible_facts": {
        "ansible_LBSync": {
            "active": true,
            "device": "LBSync",
            "features": {
                "generic_receive_offload": "on",
                "generic_segmentation_offload": "on",
                "large_receive_offload": "off",
                "ntuple_filters": "off",
                "receive_hashing": "off",
                "rx_checksumming": "on",
                "scatter_gather": "on",
                "tcp_segmentation_offload": "on",
                "tx_checksumming": "on",
                "udp_fragmentation_offload": "on"
            },
            "ipv4": {
                "address": "192.168.1.1",
                "broadcast": "192.168.1.255",
                "netmask": "255.255.255.0",
                "network": "192.168.1.0"
            },
(... Truncated ...)
```

```
(... Truncated ...)
"ansible_domain": "localdomain",
        "ansible_env": {
            (... Truncated ...)
            "PWD": "/root",
            "REMOTECONSOLE": "/bin/bash",
            "REMOTEPARTITION": "[All]",
            "REMOTEROLE": "0",
            "REMOTEROLESTR": "Administrator",
            "REMOTEUSER": "root",
            "SELINUX_LEVEL_REQUESTED": "",
            "SELINUX_ROLE_REQUESTED": "",
            "SELINUX_USE_CURRENT_RANGE": "",
            "SHELL": "/bin/bash",
            "SHLVL": "2",
            "SSH_CLIENT": "10.128.1.136 35290 22",
            "SSH_CONNECTION": "10.128.1.136 35290
10.128.1.41 22",
            "TERM": "xterm",
            "TMOUT": "0",
            "USER": "root",
            "_": "/usr/bin/python",
            "YKcnt": "0"
},

(... Truncated ...)
```

YK Module模块剖析

- Python模块保存的位置:
- Redhat:/usr/lib/python2.7/site-packages/ansible/modules/
- Ubuntu:/usr/lib/python2.7/dist-packages/ansible/modules/
- ‘.py’: 包含python源代码，以及模块逻辑实现代码
- ‘.pyc’: 编译后的二进制代码，每当import一个模块的时候，python会构建一个.pyc文件，以便更快的调用
- ‘.pyo’: 和pyc类似，但是在build的时候引入 - O进行代码优化

YK ansible 内建的模块目录以及文件清单

```
[jrodrigues@fedora YK]$ pwd; ls
/usr/lib/python2.7/site-packages/ansible/modules/network/YK
bigip_device_dns.py    bigip_gtm_datacenter.py       bigip_monitor_http.py   bigip_pool.py              bigip_sys_db.py
bigip_device_dns.pyc   bigip_gtm_datacenter.pyc      bigip_monitor_http.pyc  bigip_pool.pyc             bigip_sys_db.pyc
bigip_device_dns.pyo   bigip_gtm_datacenter.pyo      bigip_monitor_http.pyo  bigip_pool.pyo             bigip_sys_db.pyo
bigip_device_ntp.py    bigip_gtm_virtual_server.py   bigip_monitor_tcp.py    bigip_routedomain.py       bigip_virtual_server.py
bigip_device_ntp.pyc   bigip_gtm_virtual_server.pyc  bigip_monitor_tcp.pyc   bigip_routedomain.pyc      bigip_virtual_server.pyc
bigip_device_ntp.pyo   bigip_gtm_virtual_server.pyo  bigip_monitor_tcp.pyo   bigip_routedomain.pyo      bigip_virtual_server.pyo
bigip_device_sshd.py   bigip_gtm_wide_ip.py          bigip_node.py           bigip_selfip.py            bigip_vlan.py
bigip_device_sshd.pyc  bigip_gtm_wide_ip.pyc         bigip_node.pyc          bigip_selfip.pyc           bigip_vlan.pyc
bigip_device_sshd.pyo  bigip_gtm_wide_ip.pyo         bigip_node.pyo          bigip_selfip.pyo           bigip_vlan.pyo
bigip_facts.py         bigip_irule.py                bigip_pool_member.py    bigip_ssl_certificate.py   __init__.py
bigip_facts.pyc        bigip_irule.pyc               bigip_pool_member.pyc   bigip_ssl_certificate.pyc  __init__.pyc
bigip_facts.pyo        bigip_irule.pyo               bigip_pool_member.pyo   bigip_ssl_certificate.pyo  __init__.pyo
```

如何使用YK Module源码

大部分的工程师不需要也不具备修改YK Module源码的能力，但是阅读Module源码非常重要

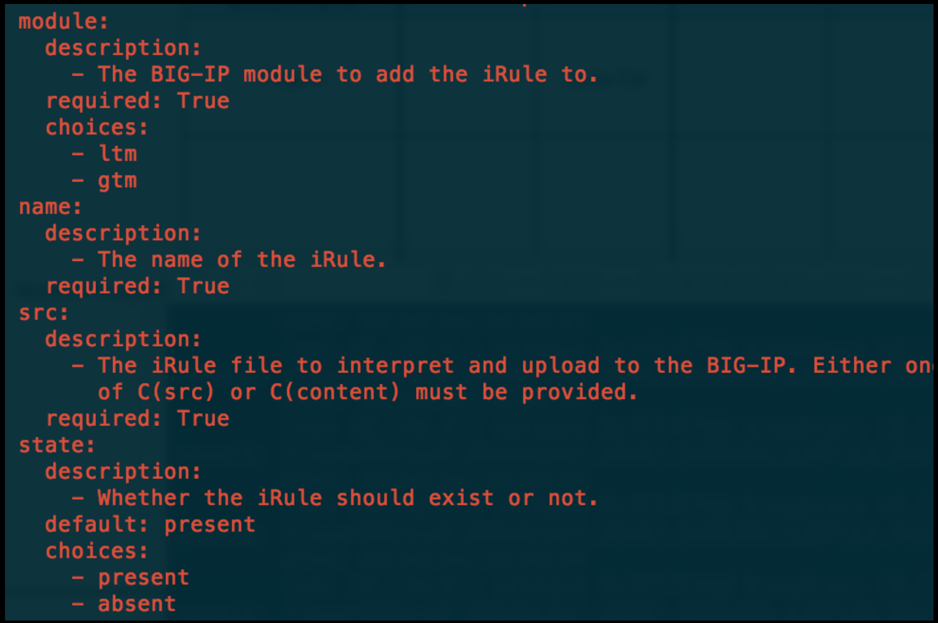
- 该模块的功能说明和使用范围说明
- 源码中有每一个参数的解释
- 包含该模块使用的范例

使用手册就在源码中，不会用可以看，看不懂可以问 * FS，但是目前不能开CASE

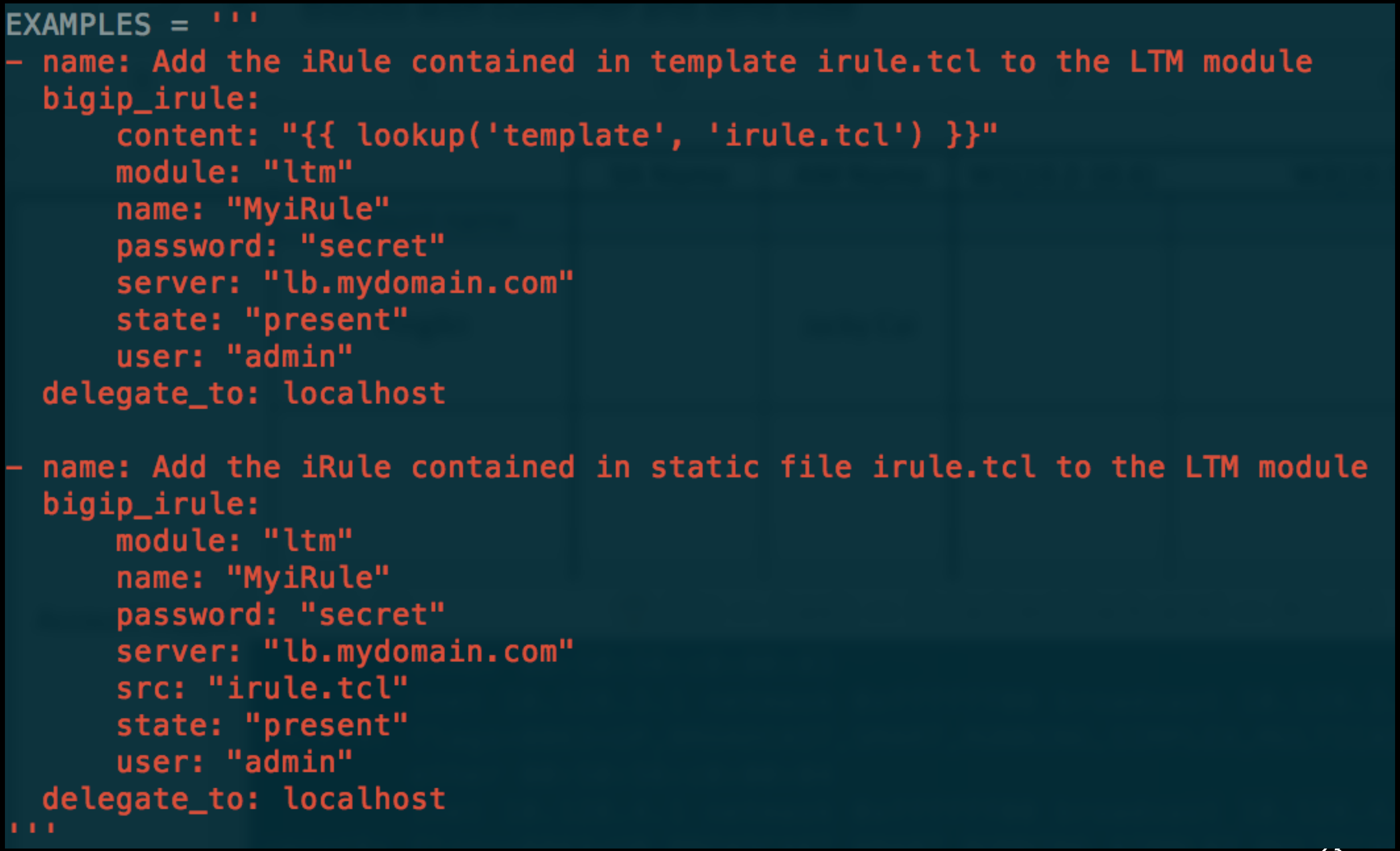
如何使用YK Module源码

以bigip_irule.py为例，源码在最开始已经有说明模块用途，前置要求，参数含义，以及例子

```
description:
  - Manage iRules across different modules on a BIG-IP.

requirements:
  - YK-sdk
```



```
module:
  description:
    - The BIG-IP module to add the iRule to.
  required: True
  choices:
    - ltm
    - gtm
name:
  description:
    - The name of the iRule.
  required: True
src:
  description:
    - The iRule file to interpret and upload to the BIG-IP. Either on
      of C(src) or C(content) must be provided.
  required: True
state:
  description:
    - Whether the iRule should exist or not.
  default: present
  choices:
    - present
    - absent
```



```
EXAMPLES = '''
- name: Add the iRule contained in template irule.tcl to the LTM module
  bigip_irule:
      content: "{{ lookup('template', 'irule.tcl') }}"
      module: "ltm"
      name: "MyiRule"
      password: "secret"
      server: "lb.mydomain.com"
      state: "present"
      user: "admin"
  delegate_to: localhost

- name: Add the iRule contained in static file irule.tcl to the LTM module
  bigip_irule:
      module: "ltm"
      name: "MyiRule"
      password: "secret"
      server: "lb.mydomain.com"
      src: "irule.tcl"
      state: "present"
      user: "admin"
  delegate_to: localhost
'''
```

Playbooks

- Ansible的配置，部署，编排方法
- 从基础角度上说，playbook可被用来做管理配置并且部署到远端系统的方法
- 在更高级的层面上，他们可以对涉及滚动更新的多层次部署进行排序，并且可以将操作委托给其他主机，与监控服务器和负载平衡器进行交互;

- Playbooks 采用YAML格式
- 语法最小化，不是一个编程语言，而是一个配置或者过程的模型;

- 目标是将一组主机套进一组操作中
- 每个Play包含一系列的任务，在执行下一个任务之前，按照主机模式匹配的所有机器的顺序逐个执行
- 在一个Play中所有主机都将获得相同的任务指令，PlayBook的目的是将选择的主机映射到任务

Playbooks - YAML

- 对于Ansible，几乎每个YAML文件都以一个列表开始。列表中的每个项目都是键/值对列表，通常称为“hash”或“dictionary”;
- YAML文件可以选择以“---”开始，以“...”结尾，并非强制要求
- “**`key: value`**” 格式保存数据
- 复合结构是可能的，例如字典列表，带有列表的字典或两者的混合

List:

```
---
# A list of tasty fruits
fruits:
    - Apple
    - Orange
    - Strawberry
    - Mango
...
```

Dictionary:

```
---
# An employee record
martin:
    name: Martin D'vloper
    job: Developer
    skill: Elite
    gender: male
...
```

List with Dictionary with Lists:

```
---
# Employee records
- martin:
    name: Martin D'vloper
    job: Developer
    skills: ["python","perl","pascal"]
- tabitha:
    name: Tabitha Bitumen
    job: Developer
    skills: ["lisp","fortran","erlan"]
...
```

Playbooks –条件判断

- Ansible任务支持when子句，其中包含一个没有双花括号的原始Jinja2表达式。这个表达式定义了将执行任务的条件

```
tasks:
  - name: "shut down Debian flavored systems"
    command: /sbin/shutdown -t now
    when: ansible_os_family == "Debian"
```

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

```
tasks:
  - command: /bin/false
    register: result
    ignore_errors: True

  - command: /bin/something
    when: result|failed

  - command: /bin/something_else
    when: result|succeeded

  - command: /bin/still/something_else
    when: result|skipped
```

```
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:
  - shell: echo "only on Red Hat 6, derivatives, and later"
    when: ansible_os_family == "RedHat" and ansible_lsb.major_release|int >= 6
```

Jinja2 filters (i.e. '|expression') in ansible: http://docs.ansible.com/ansible/playbooks_filters.html

Playbooks – 循环

Standard Loop (scalar elements)

```
- name: add several users
  user: name={{ item }} state=present groups=wheel
  with_items:
     - testuser1
     - testuser2
```

Standard Loop (hashed elements)

```
- name: add several users
  user: name={{ item.name }} state=present groups={{ item.groups }}
  with_items:
    - { name: 'testuser1', groups: 'wheel' }
    - { name: 'testuser2', groups: 'root' }
```

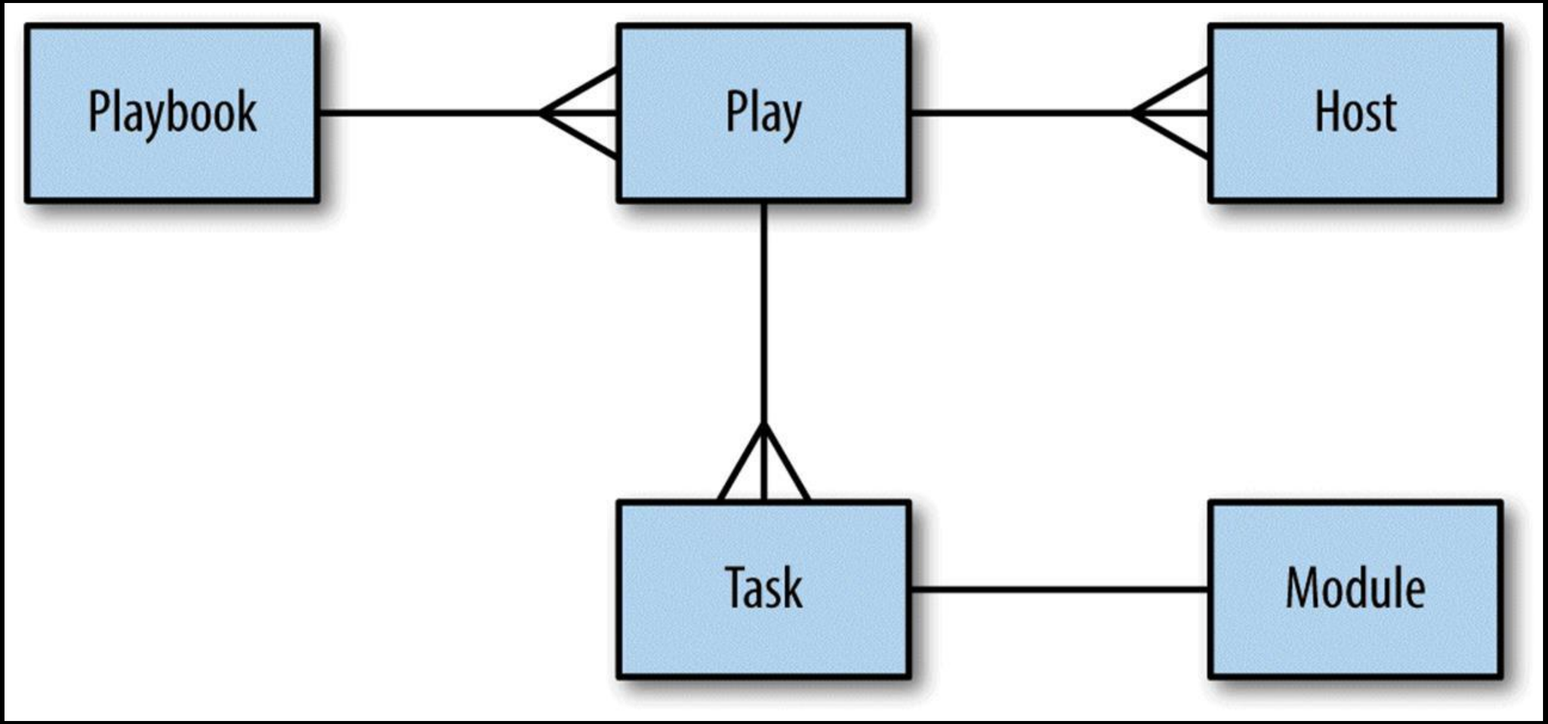
Nested Loop

```
- name: give users access to multiple databases
  mysql_user: name={{ item[0] }} priv={{ item[1] }}.*:ALL append_privs=yes password=foo
  with_nested:
    - [ 'alice', 'bob' ]
    - [ 'clientdb', 'employeedb', 'providerdb' ]
```

Dictionary Loop

```
(...Truncated...)

users:
  alice:
    name: Alice Appleworth
    telephone: 123-456-7890
  bob:
    name: Bob Bananarama
    telephone: 987-654-3210

(...Truncated...)

tasks:
  - name: Print phone records
    debug: msg="User {{ item.key }} is {{ item.value.name }} ({{ item.value.telephone }})"
    with_dict: "{{ users }}"
```

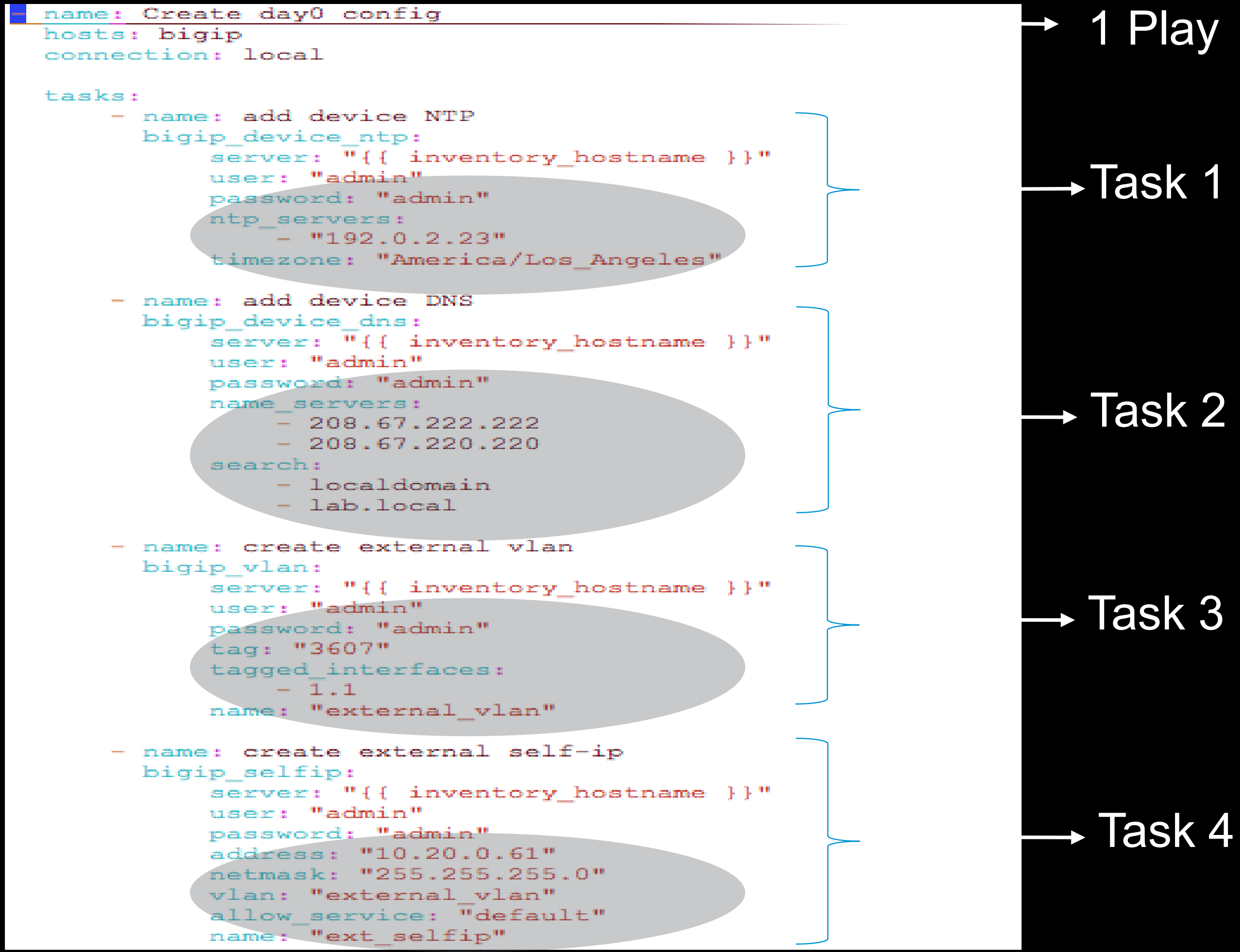
YK Ansible PLAYBOOK结构关系



- *PlayBook是YAML字典文件*
- *一个PlayBook包含多个Play，一个Play包含多个Task*

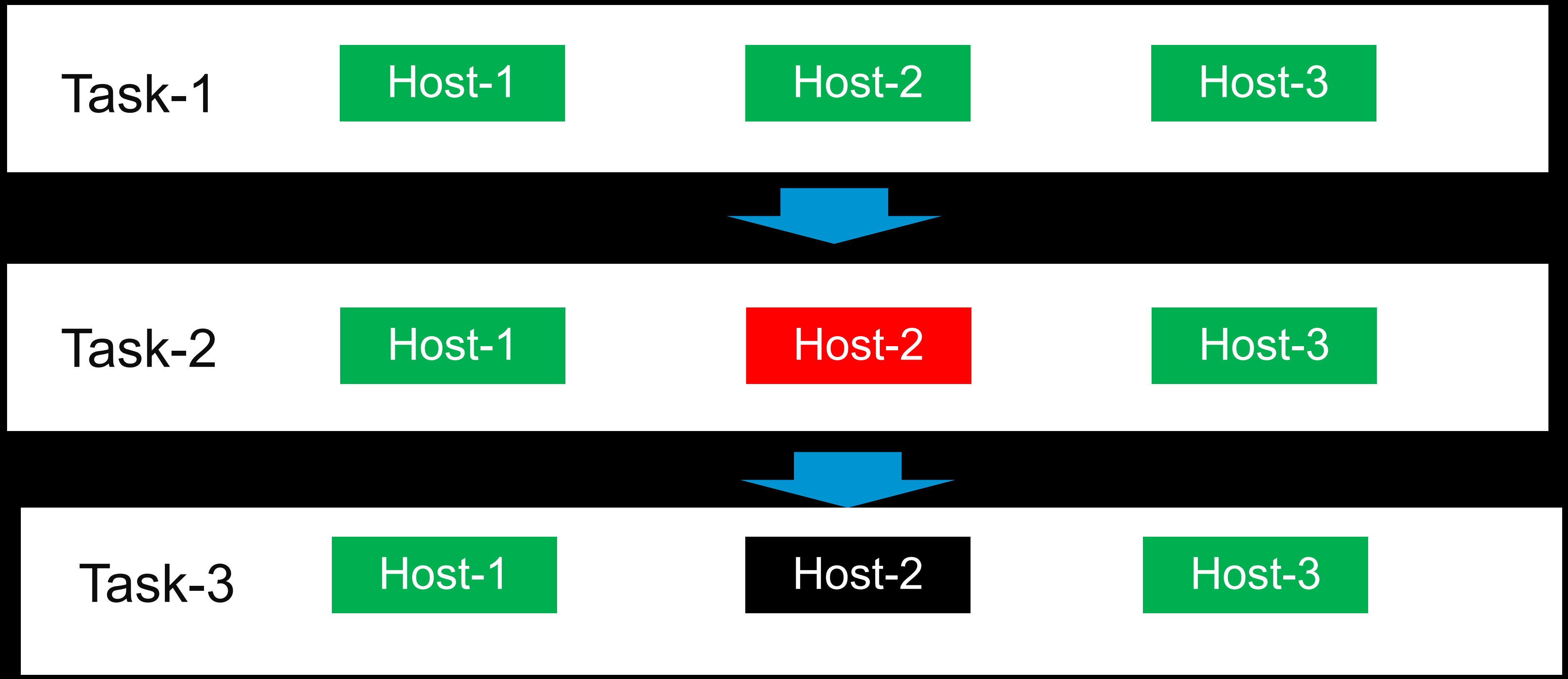
YK Ansible PLAYBOOK Sample

PLAYBOOK



```
- name: Create day0 config
  hosts: bigip
  connection: local

  tasks:
      - name: add device NTP
        bigip_device_ntp:
            server: "{{ inventory_hostname }}"
            user: "admin"
            password: "admin"
            ntp_servers:
                - "192.0.2.23"
            timezone: "America/Los_Angeles"

      - name: add device DNS
        bigip_device_dns:
            server: "{{ inventory_hostname }}"
            user: "admin"
            password: "admin"
            name_servers:
                - 208.67.222.222
                - 208.67.220.220
            search:
                - localdomain
                - lab.local

      - name: create external vlan
        bigip_vlan:
            server: "{{ inventory_hostname }}"
            user: "admin"
            password: "admin"
            tag: "3607"
            tagged_interfaces:
                - 1.1
            name: "external_vlan"

      - name: create external self-ip
        bigip_selfip:
            server: "{{ inventory_hostname }}"
            user: "admin"
            password: "admin"
            address: "10.20.0.61"
            netmask: "255.255.255.0"
            vlan: "external_vlan"
            allow_service: "default"
            name: "ext_selfip"
```

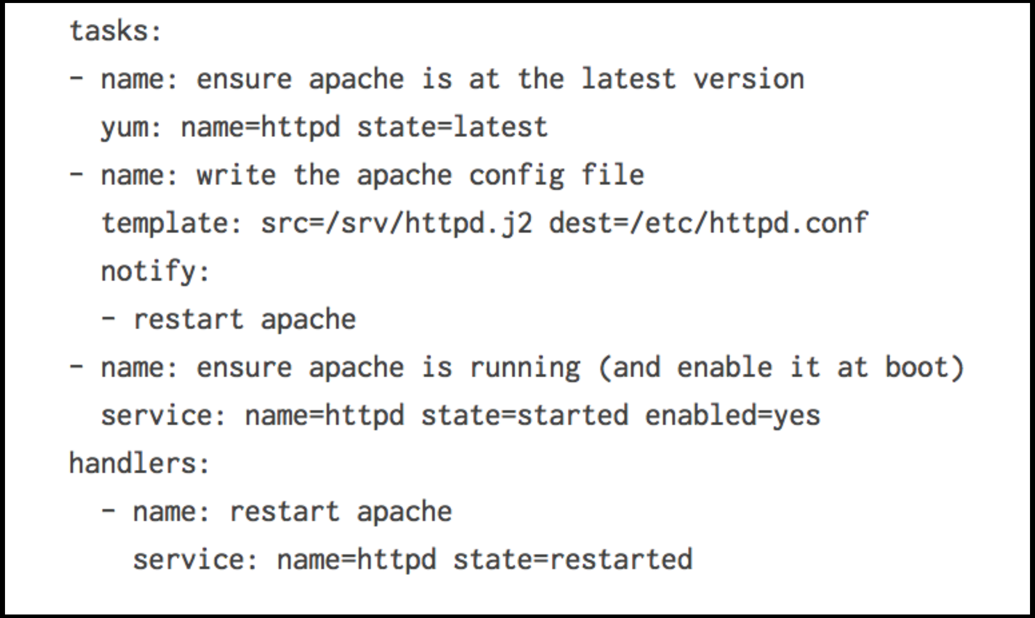
PLAYBOOK执行逻辑



- *一个Task在所有host执行完毕后，才会继续下一个Task*
- *如果一个Task在某台设备上出现异常，该主机后续Task不会执行*
- *如果Task的目标结果已经存在，但是现有配置和参数不符合，也会执行*

Notify & Handler 机制

- *ansible支持在某个task生效后（change>0），按照playbook的书写顺序，执行handler*
- *一般用于配置修改生效后，重启服务*



```
tasks:
- name: ensure apache is at the latest version
  yum: name=httpd state=latest
- name: write the apache config file
  template: src=/srv/httpd.j2 dest=/etc/httpd.conf
  notify:
  - restart apache
- name: ensure apache is running (and enable it at boot)
  service: name=httpd state=started enabled=yes
handlers:
  - name: restart apache
    service: name=httpd state=restarted
```

坑：

- *顺序执行多个Task*
- *其中一个Task出现异常，造成该host的后续Task未能执行*
- *修复了Task的异常，重新运行playbook*
- *由于前序Task在上一次运行中已经change配置，第二次运行不会触发change，也就不会触发playbook最后的handler*

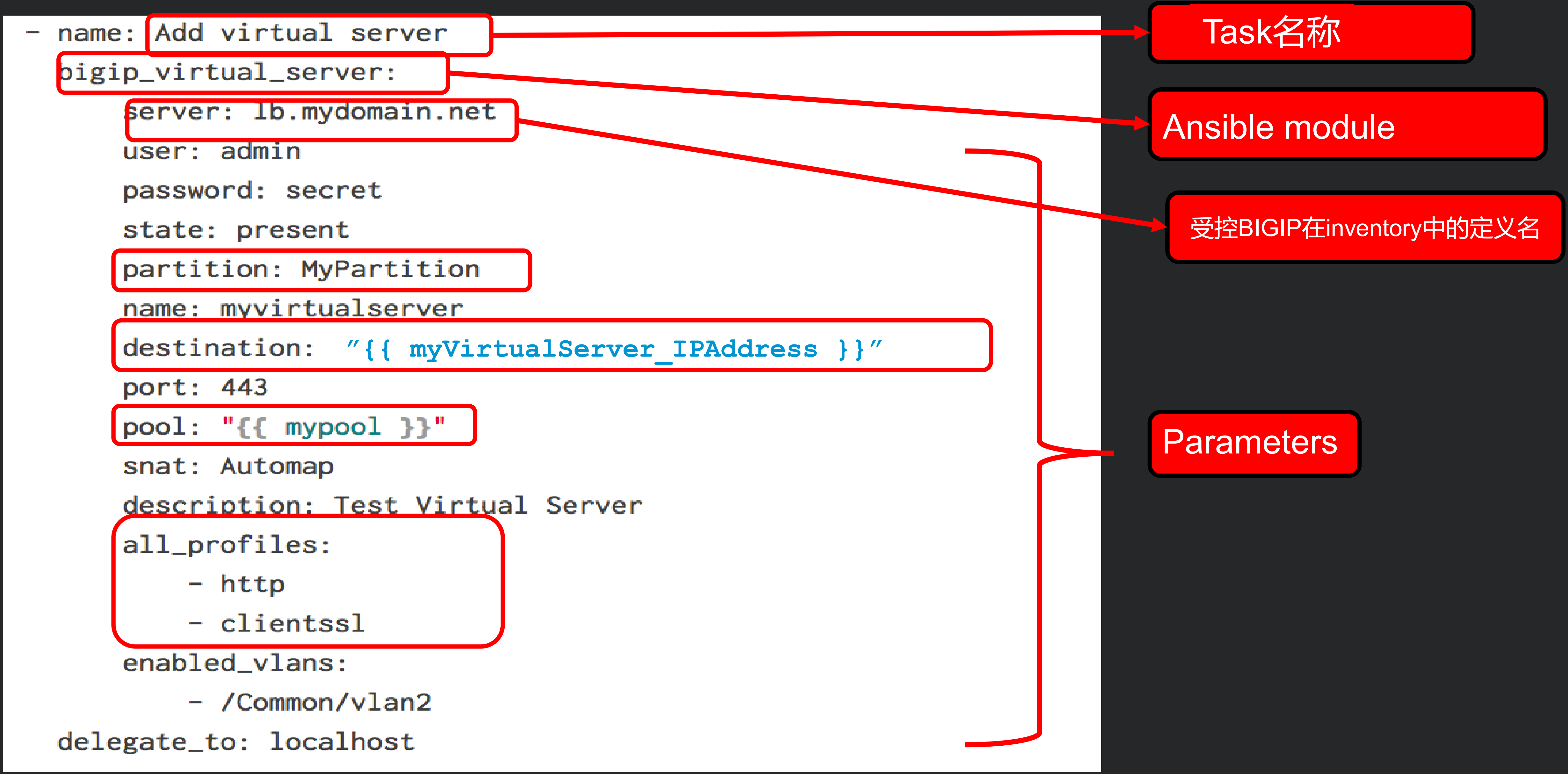
Role & Galaxy

- *Role是ansible模版化Playbook的工具*
- *用于分割，简化大型的playbook*
- *Galaxy是用来share role的工具，类似github*

Role的构成：

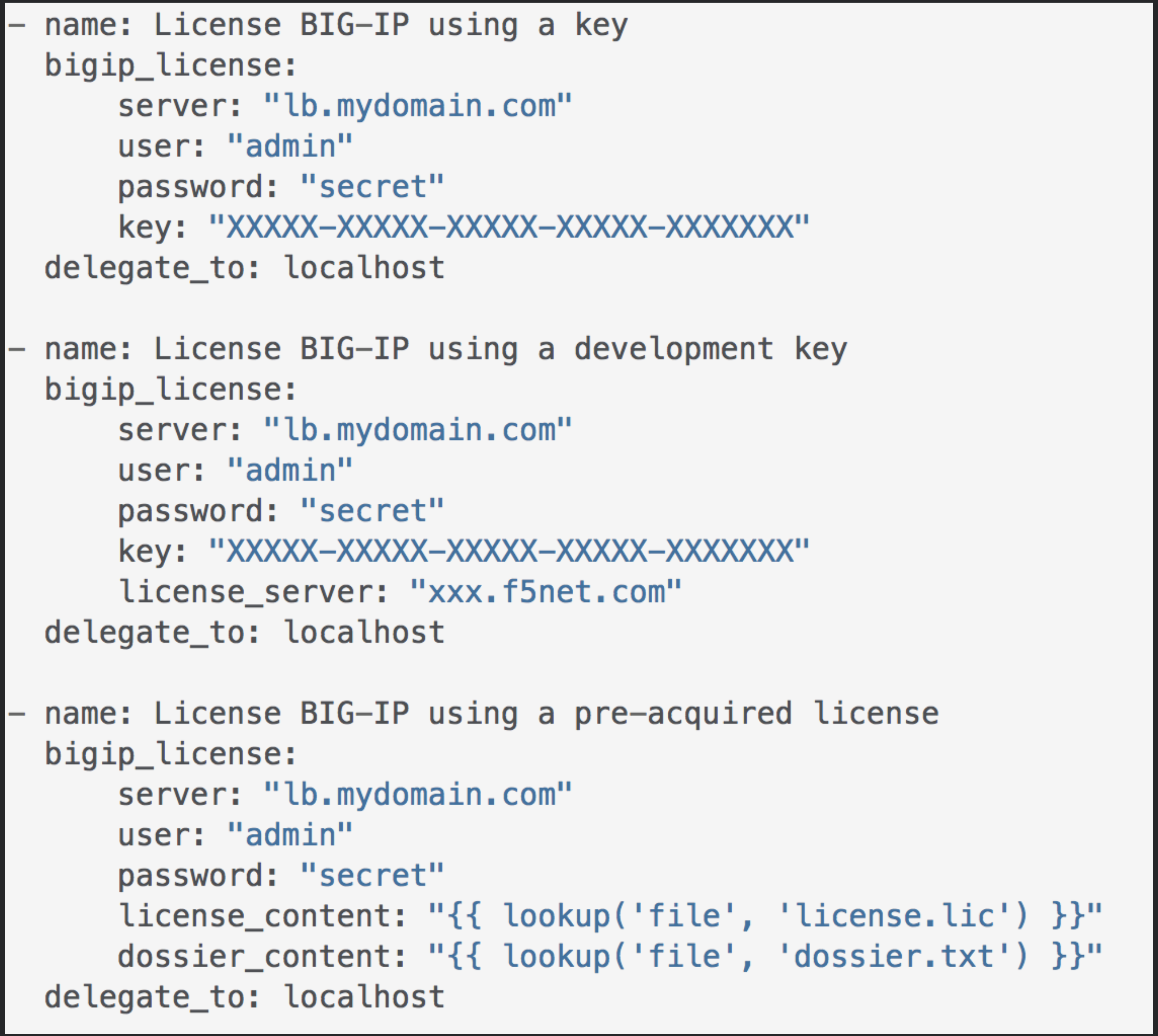
•tasks - contains the main list of tasks to be executed by the role.
•handlers - contains handlers, which may be used by this role or even anywhere outside this role.
•defaults - default variables for the role
•vars - other variables for the role
•files - contains files which can be deployed via this role.
•templates - contains templates which can be deployed via this role.
•meta - defines some meta data for this role.

PLAYBOOK YK TASK Sample



```
- name: Add virtual server
  bigip_virtual_server:
      server: lb.mydomain.net
      user: admin
      password: secret
      state: present
      partition: MyPartition
      name: myvirtualserver
      destination: "{{ myVirtualServer_IPAddress }}"
      port: 443
      pool: "{{ mypool }}"
      snat: Automap
      description: Test Virtual Server
      all_profiles:
          - http
          - clientssl
      enabled_vlans:
          - /Common/vlan2
  delegate_to: localhost
```

PLAYBOOK YK License Sample



```yaml
- name: License BIG-IP using a key
  bigip_license:
    server: "lb.mydomain.com"
    user: "admin"
    password: "secret"
    key: "XXXXX-XXXXX-XXXXX-XXXXX-XXXXXXX"
  delegate_to: localhost

- name: License BIG-IP using a development key
  bigip_license:
    server: "lb.mydomain.com"
    user: "admin"
    password: "secret"
    key: "XXXXX-XXXXX-XXXXX-XXXXX-XXXXXXX"
    license_server: "xxx.f5net.com"
  delegate_to: localhost

- name: License BIG-IP using a pre-acquired license
  bigip_license:
    server: "lb.mydomain.com"
    user: "admin"
    password: "secret"
    license_content: "{{ lookup('file', 'license.lic') }}"
    dossier_content: "{{ lookup('file', 'dossier.txt') }}"
  delegate_to: localhost
```

YK BIG-IP PlayBook范例详解

```
name: Create pool
  hosts: YK-v12
  gather_facts: no
  tasks:
   - name: Create a pool
     bigip_pool:
         lb_method: "ratio_member"
         name: "web"
         password: "admin"
         user: "admin"
         server: "big-ip02.internal"
         slow_ramp_time: "120"
         validate_certs: "no"
```

```
[YK-v12]
big-ip01.internal
big-ip02.internal

[YK-v11]
big-ip03.internal
```

该playbook没有指定connection，将使用SFTP上传bigip_pool.py模块文件上传到YK设备的临时目录执行，如无授信或者YK设备上的python版本过低无法执行

YK BIG-IP PlayBook范例详解

```
-   name: Create Pool
  hosts: YK-v12
  connection :  local
  gather_facts: no
  tasks:
   - name: Create a pool
     bigip_pool:
        lb_method: "ratio_member"
        name: "web"
        password: "admin"
        user: "admin"
        server: "big-ip02.internal"
        slow_ramp_time: "120"
        validate_certs: "no”
          delegate_to：  localhost
```

hosts文件：

```
[YK-v12]
big-ip01.internal
big-ip02.internal


[YK-v11]
big-ip03.internal
```

connection : local
playbook所有task的connection上下文

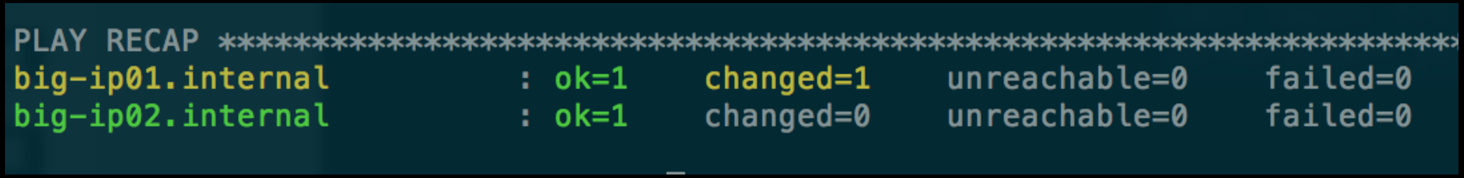
delegate_to： localhost
特定任务委派给本地运行或者特定主机，适用混合场景

server: "big-ip02.internal”
这只是模块的参数

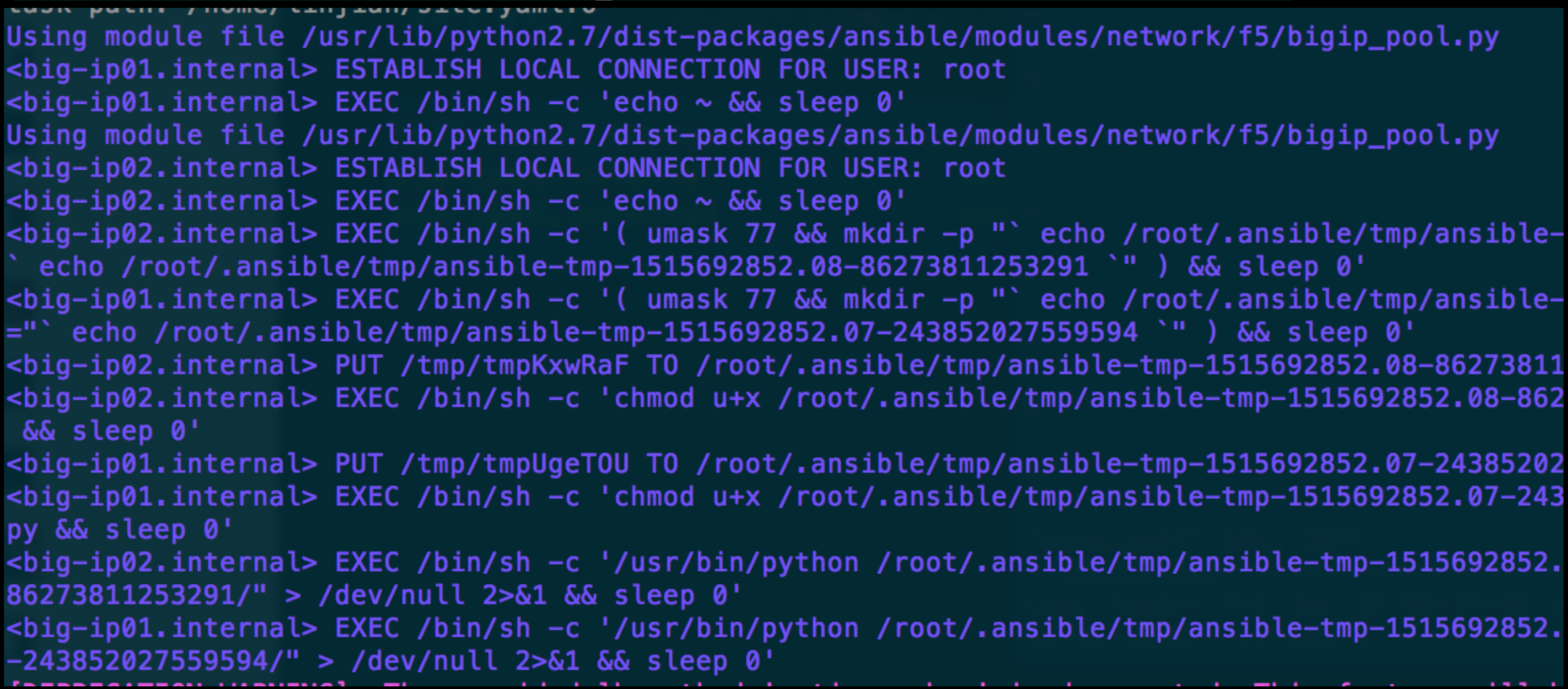
gather_facts: no
运行时通过ssh采集受控主机信息，只能通过ssh，有V11兼容问题，建议no

YK BIG-IP PlayBook范例详解

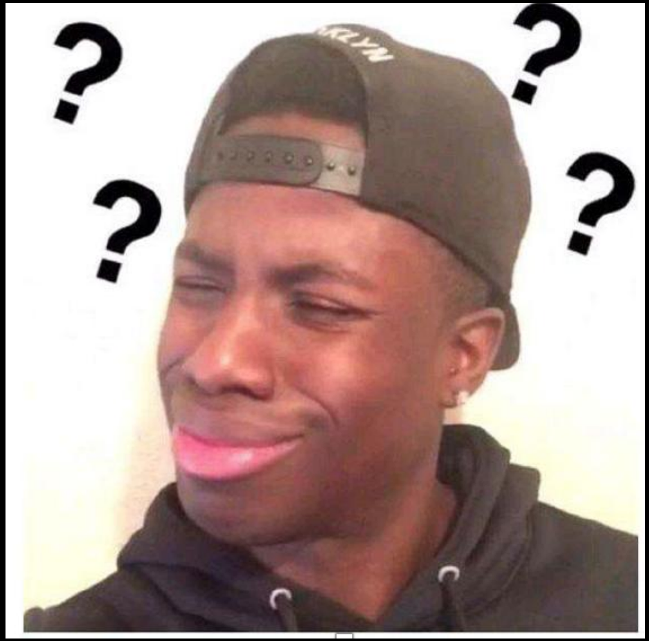
server: "big-ip02.internal"



```
PLAY RECAP *********************************************************************
big-ip01.internal          : ok=1    changed=1    unreachable=0    failed=0
big-ip02.internal          : ok=1    changed=0    unreachable=0    failed=0
```



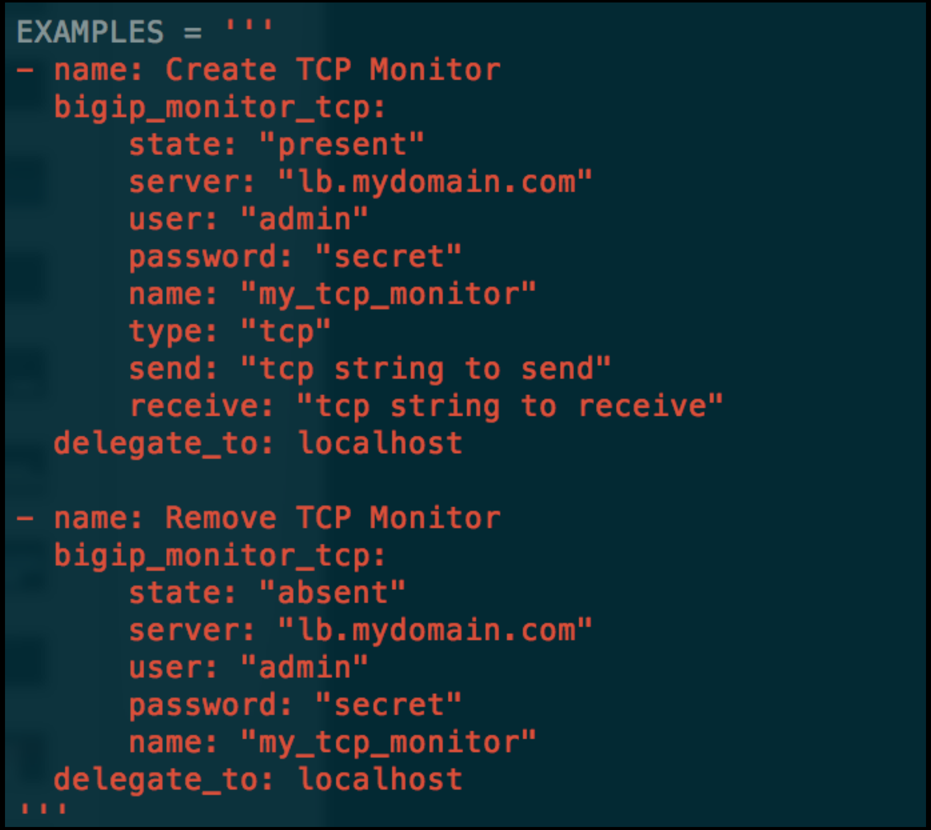
```
Using module file /usr/lib/python2.7/dist-packages/ansible/modules/network/f5/bigip_pool.py
<big-ip01.internal> ESTABLISH LOCAL CONNECTION FOR USER: root
<big-ip01.internal> EXEC /bin/sh -c 'echo ~ && sleep 0'
Using module file /usr/lib/python2.7/dist-packages/ansible/modules/network/f5/bigip_pool.py
<big-ip02.internal> ESTABLISH LOCAL CONNECTION FOR USER: root
<big-ip02.internal> EXEC /bin/sh -c 'echo ~ && sleep 0'
<big-ip02.internal> EXEC /bin/sh -c '( umask 77 && mkdir -p "` echo /root/.ansible/tmp/ansible-
` echo /root/.ansible/tmp/ansible-tmp-1515692852.08-86273811253291 `" ) && sleep 0'
<big-ip01.internal> EXEC /bin/sh -c '( umask 77 && mkdir -p "` echo /root/.ansible/tmp/ansible-
="` echo /root/.ansible/tmp/ansible-tmp-1515692852.07-243852027559594 `" ) && sleep 0'
<big-ip02.internal> PUT /tmp/tmpKxwRaF TO /root/.ansible/tmp/ansible-tmp-1515692852.08-86273811
<big-ip02.internal> EXEC /bin/sh -c 'chmod u+x /root/.ansible/tmp/ansible-tmp-1515692852.08-862
 && sleep 0'
<big-ip01.internal> PUT /tmp/tmpUgeTOU TO /root/.ansible/tmp/ansible-tmp-1515692852.07-24385202
<big-ip01.internal> EXEC /bin/sh -c 'chmod u+x /root/.ansible/tmp/ansible-tmp-1515692852.07-243
py && sleep 0'
<big-ip02.internal> EXEC /bin/sh -c '/usr/bin/python /root/.ansible/tmp/ansible-tmp-1515692852.
86273811253291/" > /dev/null 2>&1 && sleep 0'
<big-ip01.internal> EXEC /bin/sh -c '/usr/bin/python /root/.ansible/tmp/ansible-tmp-1515692852.
-243852027559594/" > /dev/null 2>&1 && sleep 0'
```



PlayBook小知识

- PlayBook中Task是按照Hosts并发多进程执行，默认每次启动5个
- 如果进程对受控主机执行了操作并且对配置变更，会有changed标示
- 主机名称可以用{{inventory_hostname}}取代，该变量为当前进程的host上下文
- 如果设备数量庞大，可采用poll模式，默认ansible为push模式
- 运行出现错误的话请ansible-playbook加-vvv，输出详细信息
- 收集fact默认（SSH）打开，如果不想配置ssh，请关闭
- local方式登录2.4版本以上会使用loginprovider方式，V11 restful不支持,该字段标示这个用户是通过3A认证还是local认证

YK BIG-IP PlayBook修改和删除

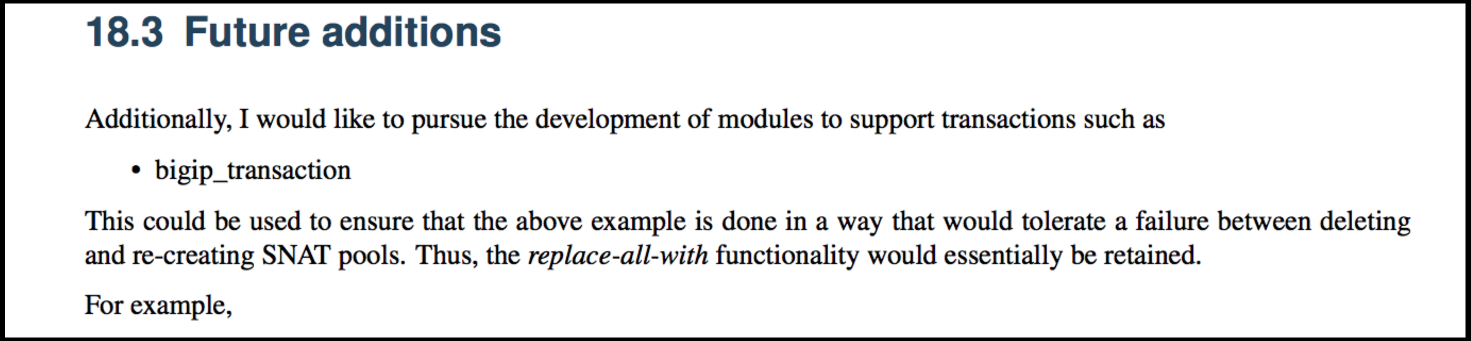


```
EXAMPLES = '''
- name: Create TCP Monitor
  bigip_monitor_tcp:
      state: "present"
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      name: "my_tcp_monitor"
      type: "tcp"
      send: "tcp string to send"
      receive: "tcp string to receive"
  delegate_to: localhost

- name: Remove TCP Monitor
  bigip_monitor_tcp:
      state: "absent"
      server: "lb.mydomain.com"
      user: "admin"
      password: "secret"
      name: "my_tcp_monitor"
  delegate_to: localhost
'''
```

- 增加和删除通过state区分
- 如果没有state默认是增加
- state="absent"是删除
- state="present"是增加和修改
- 以源码sample为准

https://media.readthedocs.org/pdf/YK-ansible/devel/YK-ansible.pdf

YK BIG-IP PlayBook支持事务吗?



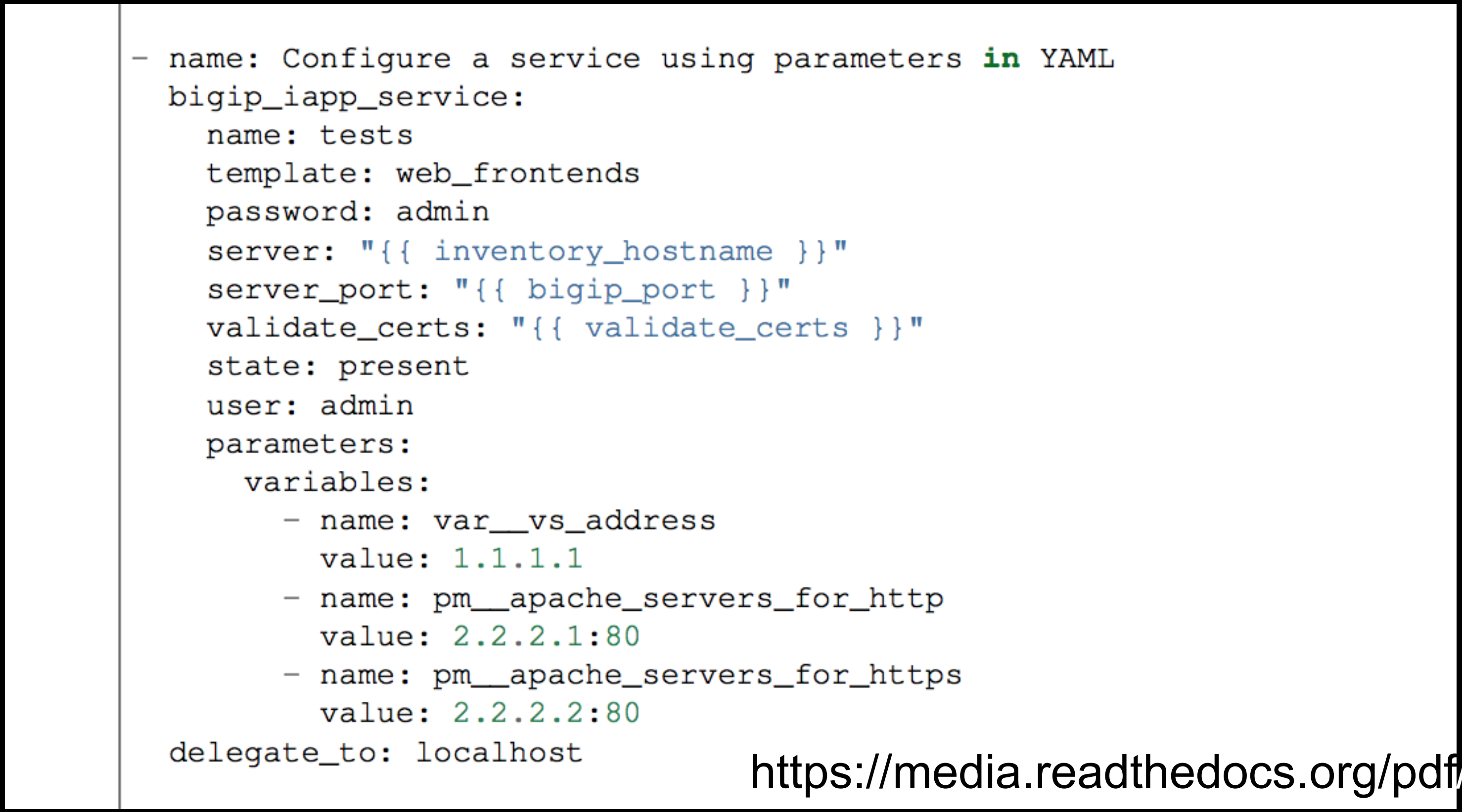
18.3 Future additions

Additionally, I would like to pursue the development of modules to support transactions such as

- bigip_transaction

This could be used to ensure that the above example is done in a way that would tolerate a failure between deleting and re-creating SNAT pools. Thus, the *replace-all-with* functionality would essentially be retained.

For example,



```
- name: Configure a service using parameters in YAML
  bigip_iapp_service:
    name: tests
    template: web_frontends
    password: admin
    server: "{{ inventory_hostname }}"
    server_port: "{{ bigip_port }}"
    validate_certs: "{{ validate_certs }}"
    state: present
    user: admin
    parameters:
      variables:
        - name: var__vs_address
          value: 1.1.1.1
        - name: pm__apache_servers_for_http
          value: 2.2.2.1:80
        - name: pm__apache_servers_for_https
          value: 2.2.2.2:80
  delegate_to: localhost
```

目前不支持，请用iapp

https://media.readthedocs.org/pdf/YK-ansible/devel/YK-ansible.pdf

Sector 1

Ansible基本概念

Sector 2

深入解析

Sector 3

Demo

YK BIG-IP Modules - bigip_ssl_certificate

bigip_ssl_certificates.yml

```
# Run example: ansible-playbook 07.bigip_ssl_certificates.yml -i inventory -l
bigip1.vlab.local
---
- name: Big-IP SSL Certificate Import
  hosts: all
  gather_facts: no

  vars:
    bigip_user: admin
    bigip_pass: admin
    obj_state: present
    cert_name: new-ssl-certificate
    cert_local_src: ./ssl-test.crt
    key_local_src: ./ssl-test.key

  tasks:
    - name: Import SSL Certificate to Big-IP
      bigip_ssl_certificate:
        name: "{{ cert_name }}"
        cert_src: "{{ cert_local_src }}"
        key_src: "{{ key_local_src }}"
        state: "{{ obj_state }}"
        server: "{{ inventory_hostname }}"
        user: "{{ bigip_user }}"
        password: "{{ bigip_pass }}"
        validate_certs: no
      delegate_to: localhost
...
```

ssl-test.crt ssl-test.key

YK BIG-IP Modules - bigip_monitor_http

bigip_http_monitor.yml

```
# Run example: ansible-playbook 08.bigip_http_monitor.yml -i inventory -l bigip1.vlab.local

---
- name: Create HTTP Monitor
  hosts: all
  gather_facts: no

  vars:
    bigip_user: admin
    bigip_pass: admin
    obj_state: present
    monitor_name: my-http-monitor
    monitor_parent: http
    monitor_interval: 5
    monitor_send: "GET / HTTP/1.1\\r\\nHost: monitor\\r\\nConnection: close\\r\\n\\r\\n"
    monitor_receive: "200"

  tasks:
    - name: Create Monitor
      bigip_monitor_http:
        name: "{{ monitor_name }}"
        parent: "{{ monitor_parent }}"
        interval: "{{ monitor_interval }}"
        send: "{{ monitor_send }}"
        receive: "{{ monitor_receive }}"
        state: "{{ obj_state }}"
        server: "{{ inventory_hostname }}"
        user: "{{ bigip_user }}"
        password: "{{ bigip_pass }}"
        validate_certs: no
      delegate_to: localhost
...
```

YK BIG-IP Modules - bigip_pool

bigip_pool.yml

```
# Run example: ansible-playbook 09.bigip_pool.yml -i
inventory -l bigip1.vlab.local
---
- name: Create pool
  hosts: all
  gather_facts: no

  vars:
    bigip_user: admin
    bigip_pass: admin
    obj_state: present
    pool_name: pool-test-1
    pool_monitor: [tcp]
    pool_lb_method: least_connection_member
    host_1: "10.128.20.10"
    port_1: "80"
    host_2: "10.128.20.11"
    port_2: "80"

  tasks:
    - name: Create Test Pool
      bigip_pool:
        name: "{{ pool_name }}"
        monitors: "{{ pool_monitor }}"
        lb_method: "{{ pool_lb_method }}"
        state: "{{ obj_state }}"
        server: "{{ inventory_hostname }}"
        user: "{{ bigip_user }}"
        password: "{{ bigip_pass }}"
        validate_certs: no
      delegate_to: localhost
```

```
  - name: Add Pool Member 1
      bigip_pool:
        name: "{{ pool_name }}"
        host: "{{  host_1 }}"
        port: "{{  port_1 }}"
        state: "{{ obj_state }}"
        server: "{{ inventory_hostname }}"
        user: "{{ bigip_user }}"
        password: "{{ bigip_pass }}"
        validate_certs: no
      delegate_to: localhost

    - name: Add Pool Member 2
      bigip_pool:
        name: "{{ pool_name }}"
        host: "{{  host_2 }}"
        port: "{{  port_2 }}"
        state: "{{ obj_state }}"
        server: "{{ inventory_hostname }}"
        user: "{{ bigip_user }}"
        password: "{{ bigip_pass }}"
        validate_certs: no
      delegate_to: localhost

...
```

YK BIG-IP Modules - bigip_pool_member

bigip_pool_member.yml

```
# Run example: ansible-playbook 10.bigip_pool_member.yml -i
inventory -l bigip1.vlab.local
---
- name: Manage Pool Members
  hosts: all
  gather_facts: no

  vars:
    bigip_user: admin
    bigip_pass: admin
    obj_state: present
    pool_name: pool-test-1
    host_1: "10.128.20.10"
    port_1: "80"
    host_2: "10.128.20.11"
    port_2: "80"

  tasks:

    - name: Set connection limit to {{ host_1 }}:{{ port_1
}}
      bigip_pool_member:
        pool: "{{ pool_name }}"
        host: "{{  host_1 }}"
        port: "{{  port_1 }}"
        connection_limit: 100
        state: "{{ obj_state }}"
        server: "{{ inventory_hostname }}"
        user: "{{ bigip_user }}"
        password: "{{ bigip_pass }}"
        validate_certs: no
      delegate_to: localhost
```

```
 - name: Set connection limit to {{ host_2 }}:{{ port_2 }}
      bigip_pool_member:
        pool: "{{ pool_name }}"
        host: "{{  host_2 }}"
        port: "{{  port_2 }}"
        session_state: disabled
        state: "{{ obj_state }}"
        server: "{{ inventory_hostname }}"
        user: "{{ bigip_user }}"
        password: "{{ bigip_pass }}"
        validate_certs: no
      delegate_to: localhost
...
```

YK BIG-IP Modules - bigip_irule

bigip_irule_source.yml

```
# Run example: ansible-playbook 11.a_bigip_irule_source.yml
-i inventory -l bigip1.vlab.local
---
- name: Create new iRule from source
  hosts: all
  gather_facts: no

  vars:
    bigip_user: admin
    bigip_pass: admin
    obj_state: present
    irule_name: test-irule-1
    irule_module: ltm
    irule_source: "when HTTP_REQUEST {\n   log local0.
\"Hello World\"\n}"

  tasks:
    - name: Create iRule from source
      bigip_irule:
        name: "{{ irule_name }}"
        module: "{{ irule_module }}"
        content: "{{ irule_source }}"
        state: "{{ obj_state }}"
        server: "{{ inventory_hostname }}"
        user: "{{ bigip_user }}"
        password: "{{ bigip_pass }}"
        validate_certs: no
      delegate_to: localhost
...
```

igip_irule_file.yml

```
# Run example: ansible-playbook 11.b_bigip_irule_file.yml -i
inventory -l bigip1.vlab.local
---
- name: Create new iRule from file
  hosts: all
  gather_facts: no

  vars:
    bigip_user: admin
    bigip_pass: admin
    obj_state: present
    irule_name: test-irule-1
    irule_module: ltm
    irule_local_path: "./test-irule-1.tcl"

  tasks:
    - name: Create iRule from source file
      bigip_irule:
        name: "{{ irule_name }}"
        module: "{{ irule_module }}"
        src: "{{ irule_local_path }}"
        state: "{{ obj_state }}"
        server: "{{ inventory_hostname }}"
        user: "{{ bigip_user }}"
        password: "{{ bigip_pass }}"
        validate_certs: no
      delegate_to: localhost
...
```

test-irule-1.tcl

YK BIG-IP Modules - bigip_vlan

bigip_vlan.yml

```
# Run example: ansible-playbook 03.bigip_facts.yml -i inventory
---
- name: Big-IP vlan creation example
  hosts: YK
  gather_facts: no

  vars:
    bigip_user: admin
    bigip_pass: admin
    obj_state: present
    vlan_conf:
        - { tag: "10", if: "1.1" }
        - { tag: "20", if: "1.2" }

  tasks:
    - name: Create VLANs
      bigip_vlan:
        name: "vlan-{{ item.tag }}"
        tag: "{{ item.tag }}"
        untagged_interfaces: ["{{ item.if }}"]
        state: "{{ obj_state }}"
        server: "{{ inventory_hostname }}"
        user: "{{ bigip_user }}"
        password: "{{ bigip_pass }}"
        validate_certs: no
      with_items: "{{ vlan_conf }}"
      delegate_to: localhost
...
```

YK BIG-IP Modules - bigip_selfip

bigip_selfip.yml

```
# Run example: ansible-playbook 05.bigip_selfip.yml -i inventory
---
- name: Big-IP selfip creation example
  hosts: bigip1.vlab.local
  gather_facts: no

  vars:
    bigip_user: admin
    bigip_pass: admin
    obj_state: present
    selfip_conf:
        - { vlan: "vlan-10", address: "10.128.10.201", netmask: "255.255.255.0", type: "static", traffic_group:
"traffic-group-local-only"}
        - { vlan: "vlan-20", address: "10.128.20.201", netmask: "255.255.255.0", type: "static", traffic_group:
"traffic-group-local-only"}

  tasks:
    - name: Create Self-IPs
      bigip_selfip:
        name: "{{ item.type }}-{{ item.vlan }}"
        vlan: "{{ item.vlan }}"
        address: "{{ item.address }}"
        netmask: "{{ item.netmask }}"
        traffic_group: "{{ item.traffic_group }}"
        state: "{{ obj_state }}"
        server: "{{ inventory_hostname }}"
        user: "{{ bigip_user }}"
        password: "{{ bigip_pass }}"
        validate_certs: no
      with_items: "{{ selfip_conf }}"
      delegate_to: localhost
...
```

Quiz

- Ansible 默认采用什么方式连接受控主机
- 什么叫做幂等，该概念的对立为何
- 对无法运行python的设备，需要什么配置才能正常连接
- ansible默认模块是什么
- Task运行出现异常，该playbook会继续执行么？

WE MAKE APPS GO

BRAND CAMPAIGN