Aruba AOS8 利用 Ansible 实现自动化运维

使用 Ansible 定期备份 Aruba MM 配置

V 1.0.0

孙继虎

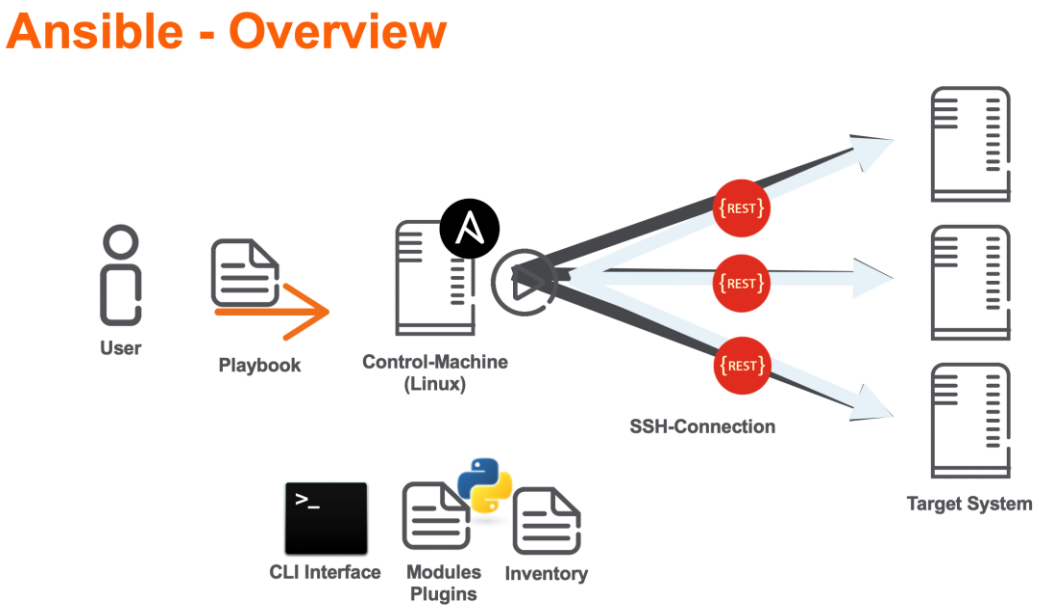
修订历史记录

下表列出了这个文档的修订历史记录

版本	日期	变更说明
1.0.0	2021/12/11	发布

1. Ansible 简介

Ansible 是基于 Python 开发的，一个自动化运维的框架。控制节点（安装 Ansible 的主机）通过各种模块（Role）或者内置的功能组件对目标设备（Inventory）进行自动化的批量操作和部署（包括但不限于：SSH，REST API 等等）。



Ansible 作为 开源的自动化运维工具已经被用在越来越多的使用场景。几乎所有的可管理设备（包括网络设备、安全设备、主机、虚拟化平台等等）都已经支持 Ansible。

这个文档将简单介绍 Ansible 的入门使用。以及利用 Ansible 实现 Aruba MM 的定期自动备份配置功能。

2. Ansible 安装

对于控制节点（运行 Ansible 的机器），您可以使用任何安装了 Python 3.8 或更新版本的机器。这包括 Red Hat、Debian、CentOS、macOS、任何 BSD 等等。需要注意

的是 Ansible 控制节点不支持 Windows，请阅读 Matt Davis 的博客文章获取有关此内容的更多信息。

2.1. 使用 pip 安装 Ansible

pip 是 Python 包管理工具，该工具提供了对 Python 包的查找、下载、安装、卸载的功能。首先确保安装了 python3.x 和 pip3。

```
python3 --version   # Python3.x 查看 python 版本命令
pip3 --version      # Python3.x 查看 pip 版本命令
```

2.1.1. 使用 pip 安装 Ansible

使用 pip 安装 ansible

```
pip install ansible
```

2.1.2. 使用 pip 升级 Ansible

1、卸妆老版本 ansible： pip uninstall ansible

```
pip uninstall ansible
Found existing installation: ansible 2.8.4
Uninstalling ansible-2.8.4:
  Would remove:
    /Library/Frameworks/Python.framework/Versions/3.7/bin/ansible
    /Library/Frameworks/Python.framework/Versions/3.7/bin/ansible-config
    /Library/Frameworks/Python.framework/Versions/3.7/bin/ansible-connection
    /Library/Frameworks/Python.framework/Versions/3.7/bin/ansible-console
    /Library/Frameworks/Python.framework/Versions/3.7/bin/ansible-doc
    /Library/Frameworks/Python.framework/Versions/3.7/bin/ansible-galaxy
```

/Library/Frameworks/Python.framework/Versions/3.7/bin/ansible-inventory

/Library/Frameworks/Python.framework/Versions/3.7/bin/ansible-playbook

/Library/Frameworks/Python.framework/Versions/3.7/bin/ansible-pull

/Library/Frameworks/Python.framework/Versions/3.7/bin/ansible-vault

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/ansible-2.8.4-py3.7.egg-info

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/ansible/*

Proceed (Y/n)? y

Successfully uninstalled ansible-2.8.4

2、重新安装 ansible： pip install ansible

pip install ansible

Collecting ansible

Downloading ansible-4.9.0.tar.gz (36.6 MB)

|████████████████████████████████| 36.6 MB 91 kB/s

Preparing metadata (setup.py) ... done

Collecting ansible-core<2.12,>=2.11.6

Downloading ansible-core-2.11.7.tar.gz (7.1 MB)

|████████████████████████████████| 7.1 MB 3.6 MB/s

Preparing metadata (setup.py) ... done

Requirement already satisfied: jinja2 in /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages (from ansible-core<2.12,>=2.11.6->ansible) (2.11.2)

Requirement already satisfied: PyYAML in /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages (from ansible-core<2.12,>=2.11.6->ansible) (5.1)

Requirement already satisfied: cryptography in /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages (from ansible-core<2.12,>=2.11.6->ansible) (2.3.1)

Requirement already satisfied: packaging in /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages (from ansible-core<2.12,>=2.11.6->ansible) (16.8)

Collecting resolvelib<0.6.0,>=0.5.3

Downloading resolvelib-0.5.4-py2.py3-none-any.whl (12 kB)

Requirement already satisfied: asn1crypto>=0.21.0 in /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages (from cryptography->ansible-core<2.12,>=2.11.6->ansible) (0.24.0)

Requirement already satisfied: six>=1.4.1 in /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages (from cryptography->ansible-core<2.12,>=2.11.6->ansible) (1.15.0)

Requirement already satisfied: cffi!=1.11.3,>=1.7 in /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages (from cryptography->ansible-core<2.12,>=2.11.6->ansible) (1.14.0)

Requirement already satisfied: idna>=2.1 in /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages (from cryptography->ansible-core<2.12,>=2.11.6->ansible) (2.6)

Requirement already satisfied: MarkupSafe>=0.23 in /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages (from jinja2->ansible-core<2.12,>=2.11.6->ansible) (1.1.1)

Requirement already satisfied: pyparsing in /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages (from packaging->ansible-core<2.12,>=2.11.6->ansible) (2.3.1)

Requirement already satisfied: pycparser in /Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages (from cffi!=1.11.3,>=1.7->cryptography->ansible-core<2.12,>=2.11.6->ansible) (2.20)

Using legacy 'setup.py install' for ansible, since package 'wheel' is not installed.

Using legacy 'setup.py install' for ansible-core, since package 'wheel' is not installed.

Installing collected packages: resolvelib, ansible-core, ansible

  Running setup.py install for ansible-core ... done

  Running setup.py install for ansible ... done

Successfully installed ansible-4.9.0 ansible-core-2.11.7 resolvelib-0.5.4

3. 安装 aos-wlan-ansible-role

Ansible 内置了很多功能模块对设备进行操作。Ansible 官方也有针对各个厂商的 API 进行功能的定制。而每个厂商自己也会针对 Ansible 开发自动化运维的模块（在 Ansible 里叫 Role）。Aruba 全系列产品都支持 Ansible。并且 Aruba 官方为每个系列产品都有定制开发 Ansible Role。

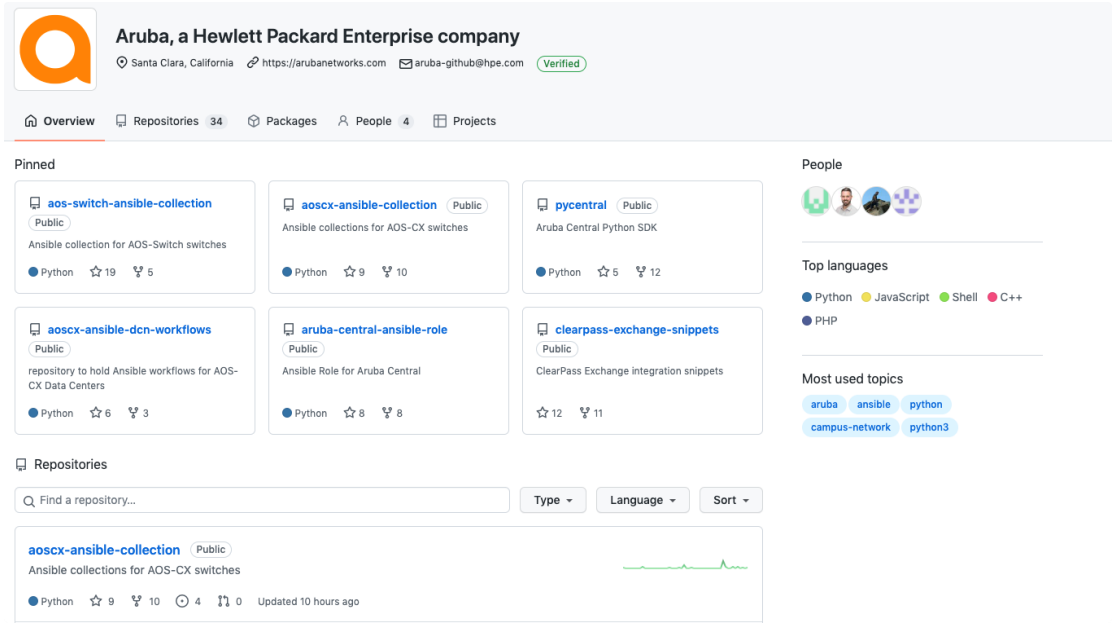
本章节需要安装 Ansible Role：aos-wlan-ansible-role，这个 Ansible Role 是专门为 ArubaOS (AOS) Mobility Master 和独立控制器设计。

Role 的安装有两种方式，一种是从 Github 安装，一种是通过 Galaxy（我们可以把他当作是 Ansible 模块的应用市场）安装。

参考：https://github.com/aruba/aos-wlan-ansible-role

3.1. 从 Github 安装

Github 是一个面向开源及私有软件项目的托管平台，很多厂商对于自己产品基于 Ansible 自动化运维的项目也会放在 Github 上，Aruba 也在 GitHub 上有开源的项目，账户的地址是：https://github.com/aruba



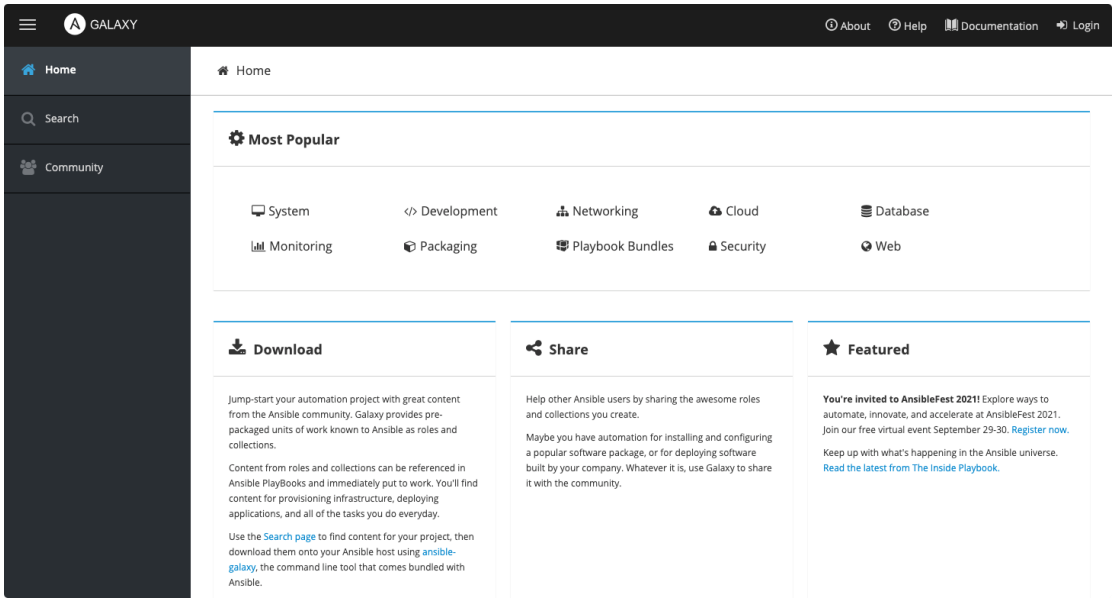
同样，我们对于全系列产品提供支持 Ansible 自动化运维，并且官方维护了每个产品的 Ansible Role。

对于 AOS8 系列的 role，通过 Github 可以用 ansible-galaxy 工具从 Github 上安装。

```
ansible-galaxy install git+https://github.com/aruba/aos-wlan-ansible-role.git
```

3.2. 从 Galaxy 安装

Ansible Galaxy（网址：https://galaxy.ansible.com/）是一个提供查找和共享 Ansible Role 的平台（类型 Apple APP Store ），通过 Galaxy 提供的 role 可以安装到本地 Ansible 帮我我们快速搭建并启动项目实现需要的功能。



对于 AOS8 系列的 role，通过 Github 可以用 ansible-galaxy 工具从 Galaxy 上安装

```
ansible-galaxy install arubanetworks.aos_wlan_role
```

4. Inventory

Ansible 可同时操作属于一个组的多台主机或者设备，组和主机（或设备）之间的关系通过 inventory 文件配置. 默认的文件路径为 /etc/ansible/hosts

Inventory 定义了 Ansible 需要操作的目标设备的参数，包括：IP 地址，登录用户名、登录密码、管理方式等等。

Inventory 文件的格式是用 YAML 语言编写的（一种表示格式化数据的语言，类似 JSOM、XML 等），想要了解 YAML 语言语法可以到这：https://www.runoob.com/w3cnote/yaml-intro.html

4.1. Inventory 参数说明

在 Inventory 中需要定义需要操作的 MM 或者独立工作的控制器的参数如下：

- ✓ ansible_host: 控制器或者 MM 的 IP 地址
- ✓ ansible_user: 控制器或者 MM 的登录用户名
- ✓ ansible_password: 控制器或者 MM 的登录密码
- ✓ ansible_connection: httpapi（固定参数）
- ✓ ansible_network_os: aos（固定参数）
- ✓ ansible_httpapi_port: 4343 （固定参数）
- ✓ ansible_httpapi_use_ssl: True （因为 AOS8 使用 HTTPS 作为 REST API）
- ✓ ansible_httpapi_validate_certs: False（设置 True 或 False，具体取决于 Ansible 是否尝试验证 MM 的 HTTPS 证书，默认设置 False）

4.2. Inventory 示例

这里以 Aruba SECLUB LAB1 为例，inventory.yml 内容如下：

```
all:
  hosts:
    lab1-mm:
      ansible_host: 10.1.50.11
      ansible_user: admin
      ansible_password: aruba123
      ansible_connection: httpapi
      ansible_network_os: aos
      ansible_httpapi_port: 4343
      ansible_httpapi_validate_certs: False
      ansible_httpapi_use_ssl: True
```

5. Playbook

Playbook 中文意思是剧本，顾名思义它定以了多个自动化运维的任务，我们像编剧一样编写这些任务，所以称之为剧本（Playbook）。Playbook 的格式也是 YAML 语言格式。

5.1. Playbook 示例：Github 安装

如果 Role 是从 Github 安装的，要把 role 设置为 aos-wlan-ansible-role。示例如下，这个 playbook 里只有一个 paly，对 MM 创建一个 radius server。

```
---
- hosts: all
  roles:
   - role: aos-wlan-ansible-role
  tasks:
    - name: Create a radius server
      aos_api_config:
        method: POST
        config_path: /md/SLR
        data:
         - rad_server:
            - rad_server_name: test-dot1x
              rad_host:
                host: 1.1.1.1
```

5.1. Playbook 示例：Galaxy 安装

如果 role 是从 Galaxy 安装的，要把 role 设置为 arubanetworks.aos_wlan_role。

```
---
- hosts: all
  roles:
  - role: arubanetworks.aos_wlan_role
  tasks:
    - name: Create a radius server
      aos_api_config:
        method: POST
        config_path: /md/SLR
        data:
         - rad_server:
            - rad_server_name: test-dot1x
              rad_host:
                host: 1.1.1.1
```

6. 实现定期自动备份配置

6.1. 实现思路

我们在用 ansible 实现定期自动备份功能之前，先想一下如果不用 ansible，我们是如何实现配置备份的。

1） 首先，我们要登录到 mm 上用“backup config <filename>”命令把配置备份到 flash

2）查看备份的配置名称 configbackup.tar.gz（假设我们设置 filename 是 configbackup）

3）通过 ftp、tftp 或者 scp 把备份配置 copy 到指定的服务器

4）定期重复执行该操作

接下来我们就是要把这个操作让 ansible 自动定期的完成，也就实现了我们的目的。

6.2. Ansible 实现自动备份配置

6.2.1. 设置 Inventory

我们以 lab1 为例，那么 inventory 文件就可以按照 4.2 配置为 inventory.yml 内容如下：

```
all:
  hosts:
    lab1-mm:
      ansible_host: 10.1.50.11
      ansible_user: admin
      ansible_password: aruba123
      ansible_connection: httpapi
```

```
    ansible_network_os: aos
    ansible_httpapi_port: 4343
    ansible_httpapi_validate_certs: False
    ansible_httpapi_use_ssl: True
```

6.2.2. 设置 palybook

按照我们之前的思路，可以将 playbook 设置如下，保存为 demo_playbook.yml

```
---
- hosts: all
  roles:
    - role: aos-wlan-ansible-role
  tasks:
    - name: 获取当前时间
      vars:
        ansible_connection: local
      debug:
        msg: "{{ansible_date_time.iso8601_basic_short}}"
      register: current_time

    - name: 备份配置{{current_time.msg}}.tar.gz 到 flash
      aos_show_command:
        command: "backup config configbackup_{{current_time.msg}}"
      register: result_bak

    - name: 上传备份配置{{current_time.msg}}.tar.gz 到 tftp 服务器 10.0.50.20/aos_lab1/
      aos_show_command:
        command: "copy flash: configbackup_{{current_time.msg}}.tar.gz tftp: 10.0.50.20 /aos_lab1/
configbackup_{{current_time.msg}}.tar.gz"
      register: result_copy
```

这个 playbook 里有三个 play，分别是

1、用 ansible 内置模块获取系统时间，格式为：20211211T104821（年月日 T 时分秒），并用参数 current_time 保存。用做后续标记备份文件的日期。

2、用我们安装的 aos-wlan-ansible-role 模块的 show command 功能备份配置文件到 flash，备份的文件名为：backupflash_{{current_time.msg}}.tar.gz

注释：

current_time 输出结果为字典{"msg": "20211211T104821"}，这里要取 msg 的 value，方式同 python 的字典取值方式（current_time.msg）。备份的文件名由固定字符串”backupflash”和 固定字符串“_”以及动态时间参数 current_time.msg 组成。

3、用我们安装的 aos-wlan-ansible-role 模块的 show command 功能将配置上传到 tftp 服务器（这里也可以是 ftp 或者 scp 服务器）

6.2.3. 运行脚本

我们在项目文件夹里有两个 YAML 文件，一个是 playbook：demo_playbook2.yml，一个是 inventory：inventory.yml

```
[root@localhost mm-backup]# ll
total 8
-rw-r--r--. 1 root root 876 Dec 11 10:48 demo_playbook2.yml
-rw-r--r--. 1 root root 294 Dec  8 18:34 inventory.yml
```

通过 ansible-playbook 命令运行脚本

```
ansible-playbook demo_playbook2.yml -i inventory.yml
```

注释：

使用 ansible-playbook 运行 playbook demo_playbook2.yml，指定 inventory 为 inventory.yml。

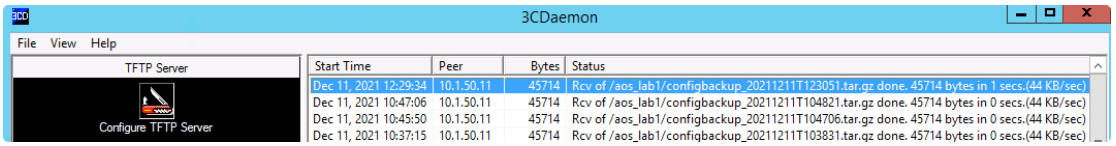
控制节点看到如下输出：

```
[root@localhost mm-backup]# ansible-playbook demo_playbook2.yml -i inventory.yml

PLAY [all] *********************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
ok: [lab1-mm]

TASK [获取当前时间] ************************************************************************************************************
******
ok: [lab1-mm] => {
    "msg": "20211211T123051"
}

TASK [备份配置20211211T123051.tar.gz到flash] ***********************************************************************************
*****
ok: [lab1-mm]

TASK [上传备份配置20211211T123051.tar.gz到tftp服务器10.0.50.20/aos_lab1/] ******************************************************
**********
ok: [lab1-mm]

PLAY RECAP *********************************************************************************************************************
lab1-mm                    : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

MM 验证结果：

```
(Lab1-MM) [mynode] #dir


-rw-r--r--    1 root     root         70533 Dec 10 18:53 AUDITTRAIL-HISTORY.log
-rw-r--r--    1 root     root         37438 Dec 10 18:53 AUDITTRAIL-LOGIN_OUT-HISTORY.log
-rw-r--r--    1 root     root           152 Dec 10 19:23 blimits
-rw-r--r--    1 root     root            40 Dec 10 19:21 bmap
-rw-r--r--    1 root     root         46647 Dec 10 18:12 configbackup.tar.gz
-rw-r--r--    1 root     root         45714 Dec 11 12:27 configbackup_20211211T123051.tar.gz
-rw-r--r--    1 root     root         24526 Oct 19 17:12 default.cfg
drwxr-xr-x    4 root     root          4096 Oct 18 23:02 fieldCerts
-rw-r--r--    1 root     root         48808 Oct 18 23:38 flashbackup.tar.gz
-rw-r--r--    1 root     root         50386 Oct 20 16:25 lanshoujun_20211020_flash.tar.gz
-rw-r--r--    1 root     root            18 Dec 10 18:55 mac_addr.cfg
-rw-r--r--    1 root     root         22072 Dec 10 18:52 reboot_config_backup.tar.gz
drwxr-xr-x    2 root     root          4096 Oct 18 23:03 upgrade-2021-10-18_23-03-52
```

TFTP 服务器上传成功记录：



6.3. Ansible 实现定期自动备份配置

6.3.1. 定期任务功能：crontab

这里利用 Linux 定期任务功能：crontab，Linux crontab 是用来定期执行程序的命令。当安装完成操作系统之后，默认便会启动此任务调度命令。crond 命令每分锺会定期检查是否有要执行的工作，如果有要执行的工作便会自动执行该工作。

参数说明：

- -e : 执行文字编辑器来设定时程表，内定的文字编辑器是 VI
- -r : 删除目前的时程表
- -l : 列出目前的时程表

进入定时任务编辑模式

```
crontab -e
```

6.3.2. 配置定期任务

为了测试，设置每 2 分钟进行一次定时任务，配置如下

```
*/2 * * * * ansible-playbook /root/Desktop/mm-backup/demo_playbook2.yml -i /root/Desktop/mm-backup/inventory.yml
```

注释：

Crontab 定时任务的具体用法参考 https://www.runoob.com/linux/linux-comm-crontab.html

```
*  *  *  *  *
-  -  -  -  -
|  |  |  |  |
|  |  |  |  +----- 星期中星期几 (0 - 6) (星期天 为 0)
|  |  |  +---------- 月份 (1 - 12)
|  |  +--------------- 一个月中的第几天 (1 - 31)
|  +------------------- 小时 (0 - 23)
+------------------------ 分钟 (0 - 59)
```

如果要设置每天指定时间备份（例如：每天凌晨 2 点自动执行配置备份），配置为：

```
00 02 * * * ansible-playbook /root/Desktop/mm-backup/demo_playbook2.yml -i /root/Desktop/mm-backup/inventory.yml
```

查看已经配置的定时任务

```
crontab -l
```

```
[root@localhost mm-backup]# crontab -l
*/2 * * * * ansible-playbook /root/Desktop/mm-backup/demo_playbook2.yml -i /root/Desktop/mm-backup/inventory.yml
```

删除当前定时任务

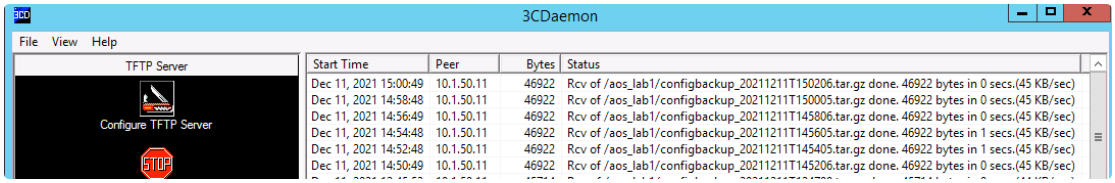
```
crontab -r
```

6.3.3. 验证

登录到 MM 查看 flash 里的自动备份的配置文件。

```
(Lab1-MM) [mynode] #dir


-rw-r--r--    1 root     root         70533 Dec 10 18:53 AUDITTRAIL-HISTORY.log
-rw-r--r--    1 root     root         37438 Dec 10 18:53 AUDITTRAIL-LOGIN_OUT-HISTORY.log
-rw-r--r--    1 root     root           152 Dec 10 19:23 blimits
-rw-r--r--    1 root     root            40 Dec 10 19:21 bmap
-rw-r--r--    1 root     root         46647 Dec 10 18:12 configbackup.tar.gz
-rw-r--r--    1 root     root         46922 Dec 11 14:49 configbackup_20211211T145206.tar.gz
-rw-r--r--    1 root     root         46922 Dec 11 14:51 configbackup_20211211T145405.tar.gz
-rw-r--r--    1 root     root         46922 Dec 11 14:53 configbackup_20211211T145605.tar.gz
-rw-r--r--    1 root     root         46922 Dec 11 14:55 configbackup_20211211T145806.tar.gz
-rw-r--r--    1 root     root         46922 Dec 11 14:57 configbackup_20211211T150005.tar.gz
-rw-r--r--    1 root     root         46922 Dec 11 14:59 configbackup_20211211T150206.tar.gz
-rw-r--r--    1 root     root         24526 Oct 19 17:12 default.cfg
drwxr-xr-x    4 root     root          4096 Oct 18 23:02 fieldCerts
-rw-r--r--    1 root     root         48808 Oct 18 23:38 flashbackup.tar.gz
```

查看 tftp 服务器自动上传的备份配置日志和配置文件



6.4. Troubleshooting

有时候我们需要检查脚本出错的原因，可以 在运行脚本的命令后面加上 参数-v 可以详细的输出信息。包括成功或者失败的信息。-vvv 可以看到更多信息。

举例，我们看到 ansible-playbook demo_playbook2.yml -i inventory.yml -vv 可以看到 API 调用的返回内容。

```
[root@localhost mm-backup]# ansible-playbook demo_playbook2.yml -i inventory.yml -vv
ansible-playbook 2.9.6
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /usr/bin/ansible-playbook
  python version = 2.7.5 (default, Aug  7 2019, 00:51:29) [GCC 4.8.5 20150623 (Red Hat 4.8.5-39)]
Using /etc/ansible/ansible.cfg as config file

PLAYBOOK: demo_playbook2.yml ***************************************************************************************
1 plays in demo_playbook2.yml

PLAY [all] *********************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
task path: /root/Desktop/mm-backup/demo_playbook2.yml:2
ok: [lab1-mm]
META: ran handlers

TASK [获取当前时间] ***********************************************************************************************
******
task path: /root/Desktop/mm-backup/demo_playbook2.yml:6
ok: [lab1-mm] => {
    "msg": "20211211T124651"
}

TASK [备份配置20211211T124651.tar.gz到flash] ***********************************************************************
*****
task path: /root/Desktop/mm-backup/demo_playbook2.yml:13
ok: [lab1-mm] => {"changed": false, "msg": {"_data": [null]}, "response_code": 200}

TASK [上传备份配置20211211T124651.tar.gz到tftp服务器10.0.50.20/aos_lab1/] ***********************************
**********
task path: /root/Desktop/mm-backup/demo_playbook2.yml:18
ok: [lab1-mm] => {"changed": false, "msg": {"_data": ["File uploaded successfully"]}, "response_code": 200}
META: ran handlers
META: ran handlers

PLAY RECAP *********************************************************************************************************
lab1-mm                    : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

6.5. 小结

这里只是抛砖引玉实现了一个小的使用场景，除了自动备份配置，也可以根据实际场景实现自动配置功能。当然同样的功能 还有很多种实现方法。比如 Ansible Tower（Ansible 的商用版本）的 Schedule 功能可以很方便的实现这个功能。