GOPS2017
Beijing

GOPS

GOPS 全球运维大会 2017 北京站

指导单位：数据中心联盟 Data Center Alliance　开放运维联盟 GOPSA Open OPS Alliance

主办单位：高效运维社区 GreatOPS Community　DevOps时代　CIO时代

大会时间：7月28-29日

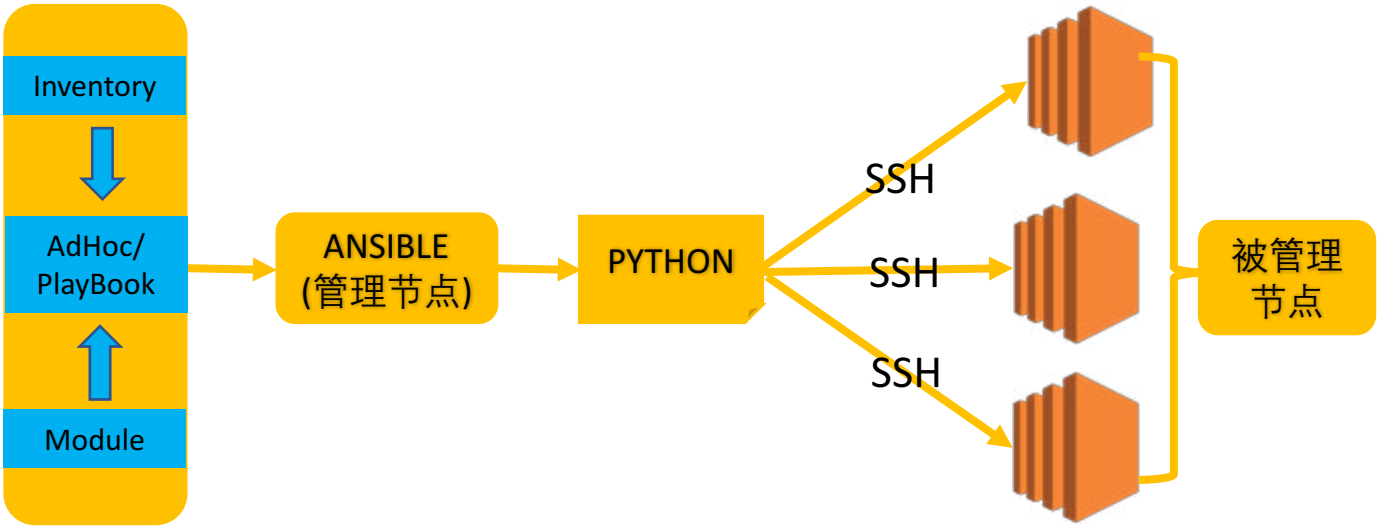
大会地点：北京朝阳悠唐皇冠假日酒店

ANSIBLE在生产环境中的实践

刘德鑫　运维工程师

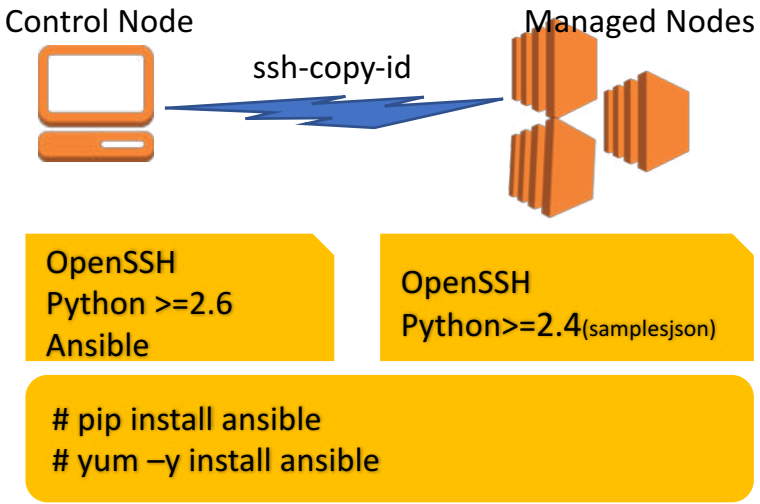
目录

Ansible工作原理



目录

Ansible部署方式



目录

Ansible如何使用

- 将所有管理的服务器上都安装了Apache服务。确保已经开启服务，且为开机自启动状态?

```
ansible all  -i ip,ip,ip -m yum  -a “name=httpd state=present”
```

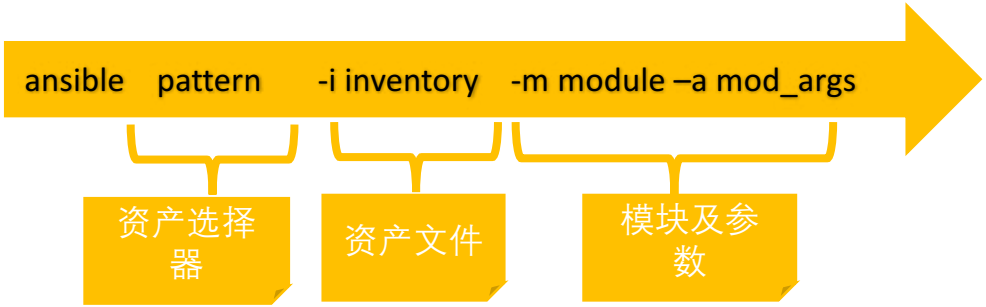
```
ansible all  -i ip,ip,ip -m service -a “name=httpd state=started enabled=yes”
```

Ansible如何使用

- AdHoc 是什么

可以认为是一个批处理工具

- AdHoc 语法规则



Ansible如何使用

- 如何学习module

ansible-doc -l # 获取所有模块列表

ansible-doc modulename #获取模块的详细信息

Ansible如何使用

- 静态资产管理

元素为主机名和IP地址

使用[start:end]表示连续地址段

服务器可以按组划分管理

组和组之间可以存在继承关系

```
192.168.1.100
192.168.2.[2:254]

[webservers]
www1.example.com
www3.example.com

[dbservers]
db[00:09].example.com

[all]
[all:children]
webservers
dbservers
```

Ansible如何使用

- PlayBook是什么

ANSIBLE提供的语言，可以类比成Linux中的Shell

一个YAML 格式的文件

Ansible如何使用

- 准备工作之YAML语法规则

以三个破折号(-)表示开始

以三个点号(.)表示结束

字母大小写敏感

以缩进表示层级关系

相同层级必须左对齐

可选项

PYTHON规则

Ansible如何使用

- 准备工作之YAML数据结构

字符串

```
“AAA”
‘AAA’
AAA
```

列表

```
- LIST1
- LIST2
- LIST3
- LIST4
```

字典

```
KEY1: VALUE1
KEY2: VALUE2
KEY3: VALUE3
KEY4: VALUE4
```

Ansible如何使用

- 一个PlayBook的构成

首先必须是个YAML文件

由一个或多个PLAY构成

每个PLAY以破折号开始

每个PLAY都是字典类型

```
---
-   key1: value1
    key2: value2
    key3: value3
    key4: value4

-   key1: value1
    key2: value2
    key3: value3
...
```

Ansible如何使用

- PLAY常用属性信息

name: 定义PLAY的名字

hosts: 定义PLAY要管理的服务器

tasks: 定义PLAY的任务,值为列表形式

Ansible如何使用

- 一个可执行的PlayBook

ansible-playbook site.yml --syntax-check

ansible-playbook site.yml

```
---
- name: the first playbook
  hosts: all
  tasks:
    - name:  install apache servers
      yum:
        name:  httpd
        state: present
    - name:  start apache servers
      service:
        name: httpd
        state:  started
...
```

目录

Ansible高级特性

- 动态资产管理

SHELL
PHP
PYTHON
...
...

```
{
    "databases"   : {
        "hosts"   : [ "host1.example.com", "host2.example.com" ],
        "vars"    : {
            "a"   : true
        }
    },
    "webservers"  : [ "host2.example.com", "host3.example.com" ],
    "atlanta"     : {
        "hosts"   : [ "host1.example.com", "host4.example.com", "host5.example.com" ],
        "vars"    : {
            "b"   : false
        },
        "children": [ "marietta", "5points" ]
    },
    "marietta"    : [ "host6.example.com" ],
    "5points"     : [ "host7.example.com" ]
}
```

Ansible高级特性

- 变量

  - Facts变量

```
ansible all -i localhost, –m setup -c local
```

```
ansible all -i localhost, –m setup -a “filter=ansible_mount*”  -c local
```

Ansible高级特性

- 变量

  - 注册变量

获取任务执行结果

```
---
- name: register variable example
  hosts: all
  tasks:
    - name: run script
      shell: /root/script.sh
      register: result
    - name: print script result
      debug: var=result
```

Ansible高级特性

- 变量

  - 全局变量

```
ansible all  –m debug –a “var=timeout” –e “timeout=10”
```

```
ansible-playbook  site.yml  –e “timeout=10”
```

Ansible高级特性

- 变量

  - 剧本变量

使用PLAY中的VARS属性定义

```
---
- name: the first playbook
  hosts: all
  vars:
    timeout: 20
  tasks:
    - name: print foo variable
      debug: var=timeout

...
```

Ansible高级特性

- 变量

  - 资产变量

变量定义在资产文件中

```
[webservers]
192.168.122.129 timeout=30
192.168.122.131

[webservers:vars]
timeout=40
```

Ansible高级特性

- 变量优先级

- role defaults [1]
- inventory INI or script group vars [2]
- inventory group_vars/all
- playbook group_vars/all
- inventory group_vars/*
- playbook group_vars/*
- inventory INI or script host vars [2]
- inventory host_vars/*
- playbook host_vars/*
- host facts
- play vars
- play vars_prompt
- play vars_files
- role vars (defined in role/vars/main.yml)
- block vars (only for tasks in block)
- task vars (only for the task)
- role (and include_role) params
- include params
- include_vars
- set_facts / registered vars
- extra vars (always win precedence)

Ansible高级特性

- 模板

使用JinJa2模板引擎

通过template模块去使用

{% ... %} # 逻辑表达

{{ ... }} # 引用变量

{# ... #} # 注释符

Ansible高级特性

- 模板

```
ServerTokens OS
ServerRoot "/etc/httpd”
PidFile run/httpd.pid
Timeout {{timeout}}
KeepAlive Off
MaxKeepAliveRequests 100
KeepAliveTimeout 15
Listen 80
```

```
ansible all -i hosts –m template –a “src=httpd.conf dest=/tmp/httpd.conf”
```

Ansible高级特性-When

- 结合Facts变量使用

```
---
 - name: the when example
   hosts: all
   tasks:
     - name:  install apache servers
       yum:  name=httpd state=latest
       when: ansible_distribution== "CentOS”
     - name:  install apache servers
       apt:  name=httpd state=latest
       when: ansible_distribution == "Ubuntu"
...
```

Ansible高级特性-When

- 结合注册变量使用

```
---
- name: register variable example
  hosts: all
  tasks:
    - name: run script
      shell: /root/script.sh
      register: result
    - name: print script result
       debug: var=result
       when result.rc == 0
...
```

Ansible高级特性

- Loop

```
---
- name: create user example
  hosts: all
  tasks:
    - name: create test1 user
      user: name=test1 state=present

    - name: create test2 user
      user: name=test2 state=present

  - name: create test3 user
      user: name=test3 state=present

...
```

Ansible高级特性

- Loop

```
---
- name: create user example
  hosts: all
  tasks:
    - name: create test1 user
      user: name={{itme}} state=present
      with_items:
          - test1
          - test2
          - test3
...
```

Ansible高级特性

- Loop

```
---
- name: create user example
  hosts: all
  vars:
    createuser:
      - test1
      - test2
      - test3
  tasks:
    - name: create test1 user
      user: name={{itme}} state=present
      with_items: "{{ createuser }}"
...
```

Ansible高级特性

- Loop&When

```
---
- name: when and loop example
  hosts: all
  tasks:
    - name: install mysql when condition success
      yum:
        name=mysql
        state=present
      with_items: "{{ ansible_mounts }}"
      when: item['mount'] == '/mysql' and item['size_available'] >= 100000000
...
```

Ansible高级特性

- Handlers

```
---
- name: the first playbook
  hosts: all
  tasks:
    - name:  install apache servers
      yum:  name=httpd state=present
    - name: update httpd.conf
      template: src=httpd.conf dest=/etc/httpd/conf/httpd.conf
      notify: restart httpd
  handlers:
    - name:  restart httpd
      service: name=httpd state=restarted
...
```

Ansible高级特性

- Tags

```
---
- name: the first playbook
  hosts: all
  tasks:
    - name:  install apache servers
      yum:  name=httpd state=present
    - name: update httpd.conf
      template: src=httpd.conf dest=/etc/httpd/conf/httpd.conf
      notify: restart httpd
      tags: updateconfig
  handlers:
    - name:  restart httpd
      service: name=httpd state=restarted
...
```

Ansible高级特性

- Tags

ansible-playbook site.yml -t 相应的tag

ansible-playbook site.yml --skip-tags 相应的tag

ansible-playbook site.yml --list-tags

all
tagged
untagged

Ansible高级特性

- 测试发布

Check Module(-C|--check)

ansible-playbook site.yml -C

测试发布，对被管理服务器不产生任何实质性的操作

Ansible高级特性

- 对比发布

Show Differences(-D|--diff)

ansible-playbook site.yml -D

显示文件和模板发布前后的差异变化，以便于风险控制

Ansible高级特性

- 灰度发布

Limit Host(-l|--limit)

ansible-playbook site.yml -l ip,ip,ip

发布至针对限定的特定Ip地址,除此之外不做任何发布

Ansible高级特性

- 零DOWN机发布

```
- hosts: all
  gather_facts: no
  serial: 1
  pre_tasks:
    - name: stop tengine
      service: name=tengine state=stopped
  tasks:
    - synchronize:
        src: /home/www/htdocs/{{ module }}/
        dest: /data0/www/htdocs/{{ module }}/
        owner: yes
        group: yes
        links: yes
        checksum: yes
        times: no
        rsync_opts:
          - "--no-motd"
          - "--exclude=.git"
  post_tasks:
    - name: start tengine
      service: name=tengine state=started
    - name: wait tengine alive
      wait_for: port=80 delay=3 timeout=15
```

目录

Ansible错误处理

- 忽略错误

```
---
- name: ignore errors example
  hosts: all
  ignore_errors: yes
  tasks:
   - name: the first task
     command: /bin/false

   - name: the second task
     command: echo "hello world"
```

Ansible错误处理

- 强制处理

```
---
- name: force handler example
  hosts: all
  force_handlers: true
  tasks:
    - name: the first task
      yum:
        name: httpd
        state: latest
      notify: start httpd
    - name: bad task
      command: /bin/false
  handlers:
    - name: start httpd
      service:
        name: httpd
        state: stated
```

Ansible错误处理

- 自定义错误

```
---
-    name: failed when example
     hosts: all
     tasks:
       - command: echo "failed in stdout"
         register: result
         failed_when: "'failed' in result.stdout"

       - name: test print last task result
         debug: var=result
```

Ansible错误处理

- 异常捕获

```
---
-    name: block rescue example
     hosts: all
     tasks:
       - block:
          - debug: msg=“this is first block debug”
          - command: /bin/false
          - debug: msg=“this is second block debug”
         rescue:
          - debug: msg="this is rescue debug"
         always:
          - debug: msg="this is always debug"
```

Ansible错误处理

- Ansible Log
- Debug Module
- 语法检查(--syntax-check)
- 详细输出(-v -vv -vvv -vvvv)

目录

如何加速Ansible

- 增大并发
- 优化FACTS
- 优化连接
- 异步任务

如何加速Ansible

- 增大并发

默认并发为 5

通过ansible|ansible-playbook 中指定并发

通过更改配置文件增加并发

如何加速Ansible

- 优化FACTS

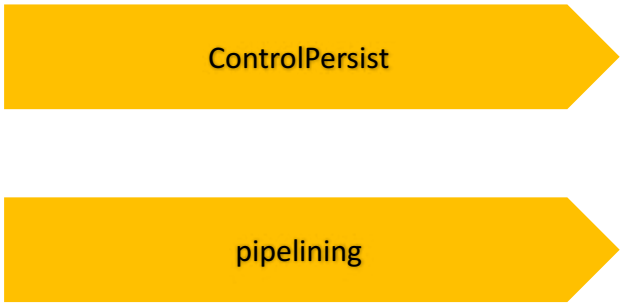
关闭FACTS

使用SUBSET

缓存FACTS

如何加速Ansible

- 优化连接



如何加速Ansible

- 异步任务

设置异步任务

异步任务查询

```
---
- name: update kernerl
  hosts: webservers
  tasks:
    - name: update kernel
      yum:
        name: '*'
        state: latest
      async: 1200
      poll: 0
      register: updatewait
      .....
      .....
      .....
    - name: wait for update kernel finished
      async_status:
      jid: "{{updatewait.ansible_job_id}}"
      register: asyncstatus
      until: asyncstatus.finished
      delay: 60
      retries: 20
```

高效运维社区
GreatOPS Community

会议

- 8月18日 DevOpsDays 上海
- 全年 DevOps China 巡回沙龙
- 11月17日 DevOps金融上海

培训

- EXIN DevOps Master 认证培训
- DevOps 企业内训
- DevOps 公开课
- 互联网运维培训

咨询

- 企业DevOps 实践咨询
- 企业运维咨询

商务经理：刘静女士
电话 / 微信：13021082989
邮箱：liujing@greatops.com

Thanks

高效运维社区

开放运维联盟

荣誉出品

想第一时间看到
高效运维社区公众号
的好文章吗？

请打开高效运维社区公众号，点击右上角小人，如右侧所示设置就好