

► 指导练习

对 PLAYBOOK 进行故障排除

在本练习中，您将对提供给您但无法正常运行的 playbook 进行故障排除。

成果

您应能够对 playbook 进行故障排除并解决其中的问题。

在你开始之前

以 **student** 用户身份并使用 **student** 作为密码登录 **workstation**。

在 **workstation** 上，运行 **lab troubleshoot-playbook start** 脚本。它会验证 **workstation** 上是否安装了 Ansible。它也会创建 **/home/student/troubleshoot-playbook/** 目录，并且从 **http://materials.example.com/labs/troubleshoot-playbook/** 下载 **inventory**、**samba.yml** 和 **samba.conf.j2** 文件到此目录。

```
[student@workstation ~]$ lab troubleshoot-playbook start
```

- 1. 在 **workstation** 上，更改到 **/home/student/troubleshoot-playbook/** 目录。

```
[student@workstation ~]$ cd ~/troubleshoot-playbook/
```

- 2. 在当前目录中创建一个名为 **ansible.cfg** 的文件。它应设置 **log_path** 参数，以将 Ansible 日志写入到 **/home/student/troubleshoot-playbook/ansible.log** 文件。它应设置 **inventory** 参数，以使用实验脚本部署的 **/home/student/troubleshoot-playbook/inventory** 文件。

完成时，**ansible.cfg** 应当包含以下内容：

```
[defaults]
log_path = /home/student/troubleshoot-playbook/ansible.log
inventory = /home/student/troubleshoot-playbook/inventory
```

- 3. 运行 playbook。它将失败并报出错误。

如果一切正确，此 playbook 将设置一台 Samba 服务器。不过，本次运行会失败，因为 **random_var** 变量定义中缺少双引号。阅读错误消息，以了解 **ansible-playbook** 如何报告问题。注意到 **random_var** 分配了一个包含冒号的值，但没有用引号括起。

```
[student@workstation troubleshoot-playbook]$ ansible-playbook samba.yml
ERROR! Syntax Error while loading YAML.
      mapping values are not allowed in this context

The error appears to have been in '/home/student/troubleshoot-playbook/samba.yml':
  line 8, column 30, but may
  be elsewhere in the file depending on the exact syntax problem
```

The offending line appears to be:

```
install_state: installed
random_var: This is colon: test
          ^ here
```

- 4. 确认该错误已正确记录到 **/home/student/troubleshoot-playbook/ansible.log** 文件中。

```
[student@workstation troubleshoot-playbook]$ tail ansible.log
The error appears to have been in '/home/student/troubleshoot-playbook/samba.yml':
line 8, column 30, but may
be elsewhere in the file depending on the exact syntax problem.
```

The offending line appears to be:

```
install_state: installed
random_var: This is colon: test
          ^ here
```

- 5. 编辑 playbook，为分配至 **random_var** 的整个值加上引号，以更正错误。更正后的 **samba.yml** 版本应当包含以下内容：

```
...output omitted...
vars:
  install_state: installed
  random_var: "This is colon: test"
...output omitted...
```

- 6. 使用 **--syntax-check** 选项，检查该 playbook。发出了另一个错误，因为最后一个任务 **deliver samba config** 的缩进中多了一个空格。

```
[student@workstation troubleshoot-playbook]$ ansible-playbook --syntax-check \
> samba.yml
ERROR! Syntax Error while loading YAML.
did not find expected key
```

```
The error appears to have been in '/home/student/troubleshoot-playbook/samba.yml':
line 44, column 4, but may
be elsewhere in the file depending on the exact syntax problem.
```

The offending line appears to be:

```
- name: deliver samba config
  ^ here
```

- 7. 编辑 playbook，删除该任务中所有行上多余的空格。更正后的 playbook 应如下所示：

```
...output omitted...
- name: configure firewall for samba
  firewalld:
    state: enabled
    permanent: true
    immediate: true
    service: samba

- name: deliver samba config
  template:
    src: templates/samba.conf.j2
    dest: /etc/samba/smb.conf
    owner: root
    group: root
    mode: 0644
```

- 8. 使用 **--syntax-check** 选项，运行该 playbook。发出了一个错误，因为 `install_state` 变量被用作 `install samba` 任务中的参数。它没有加引号。

```
[student@workstation troubleshoot-playbook]$ ansible-playbook --syntax-check \
> samba.yml
ERROR! Syntax Error while loading YAML.
  found unacceptable key (unhashable type: 'AnsibleMapping')

The error appears to have been in '/home/student/troubleshoot-playbook/samba.yml':
  line 14, column 15, but may
  be elsewhere in the file depending on the exact syntax problem.
```

The offending line appears to be:

```
  name: samba
  state: {{ install_state }}
      ^ here
```

We could be wrong, but this one looks like it might be an issue with missing quotes. Always quote template expression brackets when they start a value. For instance:

```
with_items:
  - {{ foo }}
```

Should be written as:

```
with_items:
  - "{{ foo }}"
```

- 9. 编辑 playbook，并更正 **install samba** 任务。对 **install_state** 变量的引用应放在引号里。生成的文件内容应当类似于下方所示：

```
...output omitted...
tasks:
- name: install samba
  yum:
    name: samba
    state: "{{ install_state }}"
...output omitted...
```

- 10. 使用 **--syntax-check** 选项，运行该 playbook。它应该不会显示任何其他语法错误。

```
[student@workstation troubleshoot-playbook]$ ansible-playbook --syntax-check \
> samba.yml

playbook: samba.yml
```

- 11. 运行 playbook。将发出与 SSH 相关的一个错误。

```
[student@workstation troubleshoot-playbook]$ ansible-playbook samba.yml
PLAY [Install a samba server] ****
TASK [Gathering Facts] ****
fatal: [servera.lab.example.com]: UNREACHABLE! => {"changed": false,
  "msg": "Failed to connect to the host via ssh: ssh: Could not resolve hostname
servera.lab.example.com: Name or service not known\r\n", "unreachable": true}

PLAY RECAP ****
servera.lab.example.com : ok=0    changed=0    unreachable=1    failed=0
```

- 12. 使用 **ping** 命令确保受管主机 `servera.lab.example.com` 正在运行。

```
[student@workstation troubleshoot-playbook]$ ping -c3 servera.lab.example.com
PING servera.lab.example.com (172.25.250.10) 56(84) bytes of data.
64 bytes from servera.lab.example.com (172.25.250.10): icmp_seq=1 ttl=64
time=0.247 ms
64 bytes from servera.lab.example.com (172.25.250.10): icmp_seq=2 ttl=64
time=0.329 ms
64 bytes from servera.lab.example.com (172.25.250.10): icmp_seq=3 ttl=64
time=0.320 ms

--- servera.lab.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.247/0.298/0.329/0.041 ms
```

- 13. 确保您可以 devops 用户身份使用 SSH 连接受管主机 servera.lab.example.com，并且正确的 SSH 密钥也已就位。完成时，再次注销。

```
[student@workstation troubleshoot-playbook]$ ssh devops@servera.lab.example.com
Warning: Permanently added 'servera.lab.example.com,172.25.250.10' (ECDSA) to the
list of known hosts.
...output omitted...
[devops@servera ~]$ exit
Connection to servera.lab.example.com closed.
```

- 14. 使用 -vvvv 再次运行 playbook，获得关于本次运行的更多信息。发出了一个错误，因为 servera.lab.example.com 受管主机无法访问。

```
[student@workstation troubleshoot-playbook]$ ansible-playbook -vvvv samba.yml
...output omitted...

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

PLAY [Install a samba server] ****

TASK [Gathering Facts] ****
task path: /home/student/troubleshoot-playbook/samba.yml:2
<servera.lab.exammple.com> ESTABLISH SSH CONNECTION FOR USER: devops
...output omitted...
fatal: [servera.lab.exammple.com]: UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: OpenSSH_7.8p1, OpenSSL 1.1.1
FIPS 11 Sep 2018\r\ndebug1: Reading configuration data /home/student/.ssh/
config\r\ndebug1: /home/student/.ssh/config line 1: Applying options for *
\r\ndebug1: Reading configuration data /etc/ssh/ssh_config\r\ndebug3: /etc/
ssh/ssh_config line 52: Including file /etc/ssh/ssh_config.d/01-training.conf
depth 0\r\ndebug1: Reading configuration data /etc/ssh/ssh_config.d/01-
training.conf\r\ndebug1: /etc/ssh/ssh_config.d/01-training.conf line 1: Applying
options for *\r\ndebug3: /etc/ssh/ssh_config line 52: Including file /etc/ssh/
ssh_config.d/05-redhat.conf depth 0\r\ndebug1: Reading configuration data /etc/
ssh/ssh_config.d/05-redhat.conf\r\ndebug3: /etc/ssh/ssh_config.d/05-redhat.conf
line 2: Including file /etc/crypto-policies/back-ends/openssl.config depth
1\r\ndebug1: Reading configuration data /etc/crypto-policies/back-ends/
openssl.config\r\ndebug3: gss kex names ok: [gss-gex-sha1-,gss-group14-sha1-]\r
\n debug3: kex names ok: [curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-
sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group-exchange-sha256,diffie-
hellman-group14-sha256,diffie-hellman-group16-sha512,diffie-hellman-group18-
sha512,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1]\r
\n debug1: /etc/ssh/ssh_config.d/05-redhat.conf line 8: Applying options for
*\r\ndebug1: auto-mux: Trying existing master\r\ndebug1: Control socket './
/home/student/.ansible/cp/d4775f48c9' does not exist\r\ndebug2: resolving
\"servera.lab.exammple.com\" port 22\r\ndebug2: ssh_connect_direct\r\ndebug1:
Connecting to servera.lab.exammple.com [18.211.9.206] port 22.\r\ndebug2: fd 6
setting O_NONBLOCK\r\ndebug1: connect to address 18.211.9.206 port 22: Connection
timed out\r\nnssh: connect to host servera.lab.exammple.com port 22: Connection
timed out",
    "unreachable": true
}
```

```

}

...output omitted...
PLAY RECAP *****
servera.lab.exammpple.com : ok=0     changed=0      unreachable=1    failed=0

```

- 15. 在最高级别的详细程度用于Ansible时，Ansible日志文件和控制台相比是检查输出的更好选择。在`/home/student/troubleshoot-playbook/ansible.log`文件中检查来自上一命令的输出。

```

[student@workstation troubleshoot-playbook]$ tail ansible.log
2018-12-17 19:22:50,508 p=18287 u=student |  task path: /home/student/
troubleshoot-playbook/samba.yml:2
2018-12-17 19:22:50,549 p=18287 u=student |  fatal: [servera.lab.exammpple.com]:
UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: OpenSSH_7.8p1, OpenSSL 1.1.1
FIPS 11 Sep 2018\r\ndebug1: Reading configuration data /home/student/.ssh/
config\r\ndebug1: /home/student/.ssh/config line 1: Applying options for *
\r\ndebug1: Reading configuration data /etc/ssh/ssh_config\r\ndebug3: /etc/
ssh/ssh_config line 52: Including file /etc/ssh/ssh_config.d/01-training.conf
depth 0\r\ndebug1: Reading configuration data /etc/ssh/ssh_config.d/01-
training.conf\r\ndebug1: /etc/ssh/ssh_config.d/01-training.conf line 1: Applying
options for *\r\ndebug3: /etc/ssh/ssh_config line 52: Including file /etc/ssh/
ssh_config.d/05-redhat.conf depth 0\r\ndebug1: Reading configuration data /etc/
ssh/ssh_config.d/05-redhat.conf\r\ndebug3: /etc/ssh/ssh_config.d/05-redhat.conf
line 2: Including file /etc/crypto-policies/back-ends/openssl.config depth
1\r\ndebug1: Reading configuration data /etc/crypto-policies/back-ends/
openssl.config\r\ndebug3: gss kex names ok: [gss-gex-sha1-,gss-group14-sha1-]\r
\ndebug3: kex names ok: [curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-
sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group-exchange-sha256,diffie-
hellman-group14-sha256,diffie-hellman-group16-sha512,diffie-hellman-group18-
sha512,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1]\r
\ndebug1: /etc/ssh/ssh_config.d/05-redhat.conf line 8: Applying options for
*\r\ndebug1: auto-mux: Trying existing master\r\ndebug1: Control socket '/
/home/student/.ansible/cp/d4775f48c9\" does not exist\r\ndebug2: resolving
\"servera.lab.exammpple.com\" port 22\r\ndebug2: ssh_connect_direct\r\ndebug1:
Connecting to servera.lab.exammpple.com [18.211.9.206] port 22.\r\ndebug2: fd 6
setting O_NONBLOCK\r\ndebug1: connect to address 18.211.9.206 port 22: Connection
timed out\r\ndebug1: connect to host servera.lab.exammpple.com port 22: Connection
timed out",
    "unreachable": true
}
2018-12-17 19:22:50,550 p=18287 u=student |  to retry, use: --limit @/home/
student/troubleshoot-playbook/samba.retry

2018-12-17 19:22:50,550 p=18287 u=student |  PLAY RECAP *****
2018-12-17 19:22:50,550 p=18287 u=student |  servera.lab.exammpple.com : ok=0
changed=0      unreachable=1    failed=0

```

- ▶ 16. 检查 **inventory** 文件中的错误。注意 [**samba_servers**] 组中存在拼写有误的 **servera.lab.example.com**。更正此错误，如下所示：

```
...output omitted...
[samba_servers]
servera.lab.example.com
...output omitted...
```

- ▶ 17. 再次运行该 playbook。debug install_state variable 任务返回消息 The state for the samba service is installed。此任务利用了 debug 模块，显示 install_state 变量的值。deliver samba config 任务中也显示了一个错误，因为工作目录 /home/student/troubleshoot-playbook/ 中没有 **samba.j2** 文件。

```
[student@workstation troubleshoot-playbook]$ ansible-playbook samba.yml

PLAY [Install a samba server] ****
...output omitted...
TASK [debug install_state variable] ****
ok: [servera.lab.example.com] => {
    "msg": "The state for the samba service is installed"
}
...output omitted...
TASK [deliver samba config] ****
fatal: [servera.lab.example.com]: FAILED! => {"changed": false, "msg": "Could not
  find or access 'samba.j2'\nSearched in:\n\t/home/student/troubleshoot-playbook/
  templates/samba.j2\n\t/home/student/troubleshoot-playbook/samba.j2\n\t/home/
  student/troubleshoot-playbook/templates/samba.j2\n\t/home/student/troubleshoot-
  playbook/samba.j2 on the Ansible Controller.\nIf you are using a module and expect
  the file to exist on the remote, see the remote_src option"}
...output omitted...
PLAY RECAP ****
servera.lab.example.com      : ok=7      changed=3      unreachable=0      failed=1
```

- ▶ 18. 编辑 playbook，并将 deliver samba config 任务中的 **src** 参数更正为 **samba.conf.j2**。完成后，它应当类似如下：

```
...output omitted...
- name: deliver samba config
  template:
    src: samba.conf.j2
    dest: /etc/samba/smb.conf
    owner: root
...output omitted...
```

- ▶ 19. 再次运行该 playbook。使用 **--step** 选项，执行该 playbook。它应当正常运行，没有错误。

```
[student@workstation troubleshoot-playbook]$ ansible-playbook samba.yml --step
PLAY [Install a samba server] ****
Perform task: TASK: Gathering Facts (N)o/(y)es/(c)ontinue: y
```

```
...output omitted...
Perform task: TASK: install samba (N)o/(y)es/(c)ontinue: y
...output omitted...
Perform task: TASK: install firewalld (N)o/(y)es/(c)ontinue: y
...output omitted...
Perform task: TASK: debug install_state variable (N)o/(y)es/(c)ontinue: y
...output omitted...
Perform task: TASK: start samba (N)o/(y)es/(c)ontinue: y
...output omitted...
Perform task: TASK: start firewalld (N)o/(y)es/(c)ontinue: y
...output omitted...
Perform task: TASK: configure firewall for samba (N)o/(y)es/(c)ontinue: y
...output omitted...
Perform task: TASK: deliver samba config (N)o/(y)es/(c)ontinue: y
...output omitted...
PLAY RECAP ****
servera.lab.example.com : ok=8    changed=1    unreachable=0    failed=0
```

完成

在 workstation 上，运行 **lab troubleshoot-playbook finish** 脚本来清理本练习。

```
[student@workstation ~]$ lab troubleshoot-playbook finish
```

本引导式练习到此结束。

对 ANSIBLE 受管主机进行故障排除

培训目标

学完本节后，您应能够在运行 Playbook 时对受管主机上的故障进行故障排除。

将检查模式用作测试工具

您可以使用 **ansible-playbook --check** 命令对 playbook 运行烟雾测试。此选项会执行 playbook，但不对受管主机的配置进行更改。如果 playbook 中使用的模块支持检查模式，则将显示已在受管主机上进行的更改，但不执行它们。如果模块不支持检查模式，则不显示更改，但模块仍然不执行任何操作。

```
[student@demo ~]$ ansible-playbook --check playbook.yml
```



注意

如果您的任务使用了条件，则 **ansible-playbook --check** 命令可能无法正常运行。

您还可以通过 **check_mode** 设置控制各个任务是否以检查模式运行。如果任务设置了 **check_mode: yes**，则它始终以检查模式运行，不论您是否将 **--check** 选项传递给 **ansible-playbook**。同样，如果任务设置了 **check_mode: no**，它将始终照常运行，即使您将 **--check** 传递给 **ansible-playbook**。

以下任务始终以检查模式运行，并且不进行更改。

```
tasks:  
  - name: task always in check mode  
    shell: uname -a  
    check_mode: yes
```

以下任务始终正常运行，即使其开头是 **ansible-playbook --check**。

```
tasks:  
  - name: task always runs even in check mode  
    shell: uname -a  
    check_mode: no
```

这可能很有用，因为您可以在利用 **check_mode: yes** 测试个别任务的过程中正常运行 playbook 中的大部分任务。同样，您也可利用 **check_mode: no** 运行选定的任务来收集事实或设置条件变量，但不更改受管主机，从而提高检查模式中试运行提供合理结果的几率。

通过测试魔法变量 **ansible_check_mode** 的值，任务可以判断 playbook 是否以检查模式运行。如果 playbook 以检查模式运行，则此布尔值变量设为 **true**。

**警告**

即使通过 **ansible-playbook --check** 运行 playbook，设置了 **check_mode: no** 的任务也将运行。因此，如果您不通过检查 playbook 及其关联的任何角色或任务来进行确认，您就无法确信 **--check** 选项不会更改受管主机。

**注意**

如果您有较旧的 playbook 并且它们使用 **always_run: yes** 在检查模式中强制照常运行任务，那么在 Ansible 2.6 和更新版本中，您必须要将这些代码替换为 **check_mode: no**。

此外，**ansible-playbook** 命令还提供一个 **--diff** 选项。此选项可报告对受管主机上的模板文件所做的更改。如果与 **--check** 选项结合，则命令输出中会显示这些更改，但实际上不进行更改。

```
[student@demo ~]$ ansible-playbook --check --diff playbook.yml
```

使用模块进行测试

一些模块可以提供关于受管主机状态的额外信息。下表中列出了一些 Ansible 模块，可用于测试和调试受管主机上的问题。

- **uri** 模块提供了一种方式，可以检查 RESTful API 是否返回需要的内容。

```
tasks:
  - uri:
      url: http://api.myapp.com
      return_content: yes
      register: apiresponse

  - fail:
      msg: 'version was not provided'
      when: "'version' not in apiresponse.content"
```

- **script** 模块支持在受管主机上执行脚本，如果该脚本的返回代码不是零，则报告失败。脚本必须存在于控制节点上，传输到受管主机并在其上执行。

```
tasks:
  - script: check_free_memory
```

- **stat** 模块收集文件的事实，与 **stat** 命令非常相似。您可以使用它来注册变量，然后进行测试来确定文件是否存在或获取有关该文件的其他信息。如果文件不存在，则 **stat** 任务不会失败，但其注册的变量会为 ***.stat.exists** 报告 **false**。

在本例中，**/var/run/app.lock** 存在时应用仍然会运行；这时，play 应该会中止。

```
tasks:
  - name: Check if /var/run/app.lock exists
    stat:
```

```

    path: /var/run/app.lock
    register: lock

- name: Fail if the application is running
  fail:
    when: lock.stat.exists

```

assert 是 **fail** 模块的一种替代选择。**assert** 模块支持 **that** 选项，该选项取一个条件列表作为值。如果这些条件中的任何一个为 `false`，则任务失败。您可以使用 `success_msg` 和 `fail_msg` 选项来自定义它报告成功或失败时显示的消息。

以下示例重复上一个示例，但使用 **assert** 来替代 **fail**。

```

tasks:
- name: Check if /var/run/app.lock exists
  stat:
    path: /var/run/app.lock
  register: lock

- name: Fail if the application is running
  assert:
    that:
      - not lock.stat.exists

```

对连接进行故障排除

使用 Ansible 管理主机时的许多常见问题与主机连接相关，也与围绕远程用户和特权升级的配置问题有关。

如果您遇到与受管主机身份验证相关的问题，请确保在配置文件或 play 中正确设置了 `remote_user`。您还应确认设置了正确的 SSH 密钥或为该用户提供正确的密码。

务必正确设置 `become`，并且使用正确的 `become_user`（默认为 `root`）。您应该确认，您输入了正确的 `sudo` 密码并且受管主机上正确配置了 `sudo`。

一个更微妙的问题与清单设置有关。对于具有多个网络地址的复杂服务器，连接该系统时可能需要使用特定的地址或 DNS 名称。您可能不希望将该地址用作计算机的清单名称，从而提高可读性。您可以设置主机清单变量 `ansible_host`，它会用其他名称或 IP 地址覆盖清单名称并供 Ansible 用于连接该主机。该变量可以在该主机的 `host_vars` 文件或目录中设置，或者可在清单文件本身中设置。

例如，以下清单条目将 Ansible 配置为在处理主机 `web4.phx.example.com` 时连接到 `192.0.2.4`：

```
web4.phx.example.com ansible_host=192.0.2.4
```

这是控制 Ansible 如何连接受管主机的有用方式。但是，`ansible_host` 的值不正确时也可能会导致问题。

使用临时命令测试受管主机

下面几个示例演示了一些可通过使用临时命令在受管主机上执行的检查。

您曾使用过 **ping** 模块来测试是否能够连接到受管主机。根据您传递的选项，您还可以使用它来测试是否正确配置了特权升级和凭据。

```
[student@demo ~]$ ansible demohost -m ping
demohost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
[student@demo ~]$ ansible demohost -m ping --become
demohost | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "module_stderr": "sudo: a password is required\n",
    "module_stdout": "",
    "msg": "MODULE FAILURE\nSee stdout/stderr for the exact error",
    "rc": 1
}
```

本例返回 **demohost** 受管主机中配置的磁盘上的当前可用空间。这可用于确认受管主机上的文件系统是否未满。

```
[student@demo ~]$ ansible demohost -m command -a 'df'
```

本例返回 **demohost** 受管主机上当前的可用内存。

```
[student@demo ~]$ ansible demohost -m command -a 'free -m'
```

正确的测试等级

Ansible 设计为可确保 playbook 中包含的并由其模块执行的配置正确无误。它监控所有模块是否报告有故障，一旦遇到任何故障将立即停止 playbook。这有助于确保出现故障之前执行的任何任务都没有错误。

正因为此，通常无需检查 Ansible 所管理的任务的结果是否已正确应用到受管主机上。当需要更直接的故障排除时，可以在 playbook 中添加一些健康检查，或者以临时命令形式直接运行它们。但是，您应该小心谨慎，不要为了重复检查模块本身执行的测试给为您的任务和 play 添加太多复杂性。



参考文献

检查模式（“空运行”）-- Ansible 文档

https://docs.ansible.com/ansible/latest/user_guide/playbooks_checkmode.html

测试策略 -- Ansible 文档

https://docs.ansible.com/ansible/latest/reference_appendices/test_strategies.html

► 指导练习

对 ANSIBLE 受管主机进行故障排除

在本练习中，您将对运行 playbook 时其中一个受管主机上发生的任务故障进行故障排除。

成果

您应该能够对受管主机进行故障排除。

在你开始之前

以 student 用户身份并使用 student 作为密码登录 workstation。

在 workstation 上，运行 **lab troubleshoot-host start** 脚本。它会确保 workstation 上安装了 Ansible。它也将 **inventory**、**mailrelay.yml** 和 **postfix-relay-main.conf.j2** 文件从 <http://materials.example.com/troubleshoot-host/> 下载到 **/home/student/troubleshoot-host/** 目录。

```
[student@workstation ~]$ lab troubleshoot-host start
```

- 1. 在 workstation 上，更改到 **/home/student/troubleshoot-host/** 目录。

```
[student@workstation ~]$ cd ~/troubleshoot-host/
```

- 2. 使用检查模式运行 **mailrelay.yml** playbook。

```
[student@workstation troubleshoot-host]$ ansible-playbook mailrelay.yml --check
PLAY [create mail relay servers] ****
...output omitted...
TASK [check main.cf file] ****
ok: [servera.lab.example.com]

TASK [verify main.cf file exists] ****
ok: [servera.lab.example.com]  => {
    "msg": "The main.cf file exists"
}
...output omitted...
TASK [email notification of always_bcc config] ****
fatal: [servera.lab.example.com]: FAILED! => {"msg": "The conditional check
'bcc_state.stdout != 'always_bcc =' failed. The error was: error while
evaluating conditional (bcc_state.stdout != 'always_bcc ='): 'dict object'
has no attribute 'stdout'\n\nThe error appears to have been in '/home/student/
troubleshoot-host/mailrelay.yml': line 42, column 7, but may\nbe elsewhere in the
file depending on the exact syntax problem.\n\nThe offending line appears to be:
\n\n      - name: email notification of always_bcc config\n          ^ here\n"}  
海量视频题库 myitpub.com QQ:5565462  
www.52myit.com
```

```
...output omitted...
PLAY RECAP *****
servera.lab.example.com : ok=6    changed=3    unreachable=0    failed=1
```

verify main.cf file exists 任务使用 `stat` 模块。它确认 `servera.lab.example.com` 上存在 **main.cf**。

`email notification of always_bcc config` 任务失败。它没有收到来自 `check for always_bcc` 任务的输出，因为该 playbook 是使用检查模式执行的。

► 3. 使用一个临时命令，检查 `/etc/postfix/main.cf` 文件的标头。

```
[student@workstation troubleshoot-host]$ ansible servera.lab.example.com \
> -u devops -b -a "head /etc/postfix/main.cf"
servera.lab.example.com | FAILED | rc=1 >>
head: cannot open '/etc/postfix/main.cf' for reading: No such file or
directorynon-zero return code
```

该命令失败，因为 playbook 是使用检查模式执行的。`servera.lab.example.com` 上没有安装 Postfix。

► 4. 再次运行 playbook，但不指定检查模式。`email notification of always_bcc config` 任务中的错误应当会消失。

```
[student@workstation troubleshoot-host]$ ansible-playbook mailrelay.yml
PLAY [create mail relay servers] *****
...output omitted...
TASK [check for always_bcc] *****
changed: [servera.lab.example.com]

TASK [email notification of always_bcc config] *****
skipping: [servera.lab.example.com]

RUNNING HANDLER [restart postfix] *****
changed: [servera.lab.example.com]

PLAY RECAP *****
servera.lab.example.com : ok=8    changed=5    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
```

► 5. 使用一个临时命令，显示 `/etc/postfix/main.cf` 文件的开头部分。

```
[student@workstation troubleshoot-host]$ ansible servera.lab.example.com \
> -u devops -b -a "head /etc/postfix/main.cf"
servera.lab.example.com | SUCCESS | rc=0 >>
# Ansible managed
#
# Global Postfix configuration file. This file lists only a subset
# of all parameters. For the syntax, and for a complete parameter
# list, see the postconf(5) manual page (command: "man 5 postconf").
#
```

```
# For common configuration examples, see BASIC_CONFIGURATION_README
# and STANDARD_CONFIGURATION_README. To find these documents, use
# the command "postconf html_directory readme_directory", or go to
# http://www.postfix.org/BASIC_CONFIGURATION_README.html etc.
```

它现在以含有字符串“Ansible managed”的一行开头。此文件已经更新，现在已由Ansible管理。

► 6. 添加一项任务，以启用通过防火墙的smtp服务。

```
[student@workstation troubleshoot-host]$ vim mailrelay.yml
...output omitted...
- name: postfix firewalld config
  firewalld:
    state: enabled
    permanent: true
    immediate: true
    service: smtp
...output omitted...
```

► 7. 运行playbook。**postfix firewalld config**任务应当无误执行。

```
[student@workstation troubleshoot-host]$ ansible-playbook mailrelay.yml
PLAY [create mail relay servers] ****
...output omitted...
TASK [postfix firewalld config] ****
changed: [servera.lab.example.com]

PLAY RECAP ****
servera.lab.example.com      : ok=8      changed=2      unreachable=0      failed=0
skipped=1      rescued=0      ignored=0
```

► 8. 使用一个临时命令，检查servera.lab.example.com上的防火墙中现在是否配置了smtp服务。

```
[student@workstation troubleshoot-host]$ ansible servera.lab.example.com \
> -u devops -b -a "firewall-cmd --list-services"
servera.lab.example.com | CHANGED | rc=0 >>
cockpit dhcpcv6-client samba smtp ssh
```

► 9. 使用telnet测试SMTP服务是否在侦听servera.lab.example.com上的端口TCP/25。结束时断开连接。

```
[student@workstation troubleshoot-host]$ telnet servera.lab.example.com 25
Trying 172.25.250.10...
Connected to servera.lab.example.com.
Escape character is '^>'.
220 servera.lab.example.com ESMTP Postfix
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

完成

在 workstation 上，运行 **lab troubleshoot-host finish** 脚本来清理本练习。

```
[student@workstation ~]$ lab troubleshoot-host finish
```

本引导式练习到此结束。

► 开放研究实验

对 ANSIBLE 进行故障排除

任务执行清单

在本实验中，您将对尝试运行提供给您的 playbook 时出现的问题进行故障排除。

成果

您应能够：

- 对 playbook 进行故障排除。
- 对受管主机进行故障排除。

在你开始之前

以 student 用户身份并使用 student 作为密码登录 workstation。运行 **lab troubleshoot-review start** 命令。

```
[student@workstation ~]$ lab troubleshoot-review start
```

此脚本可确保在 workstation 上安装 Ansible，并且创建 **~student/troubleshoot-review/html/** 目录。它将 **ansible.cfg**、**inventory-lab** 和 **secure-web.yml** 文件从 **vhosts.conf** 下载从 <http://materials.example.com/labs/troubleshoot-review/> 下载到 **/home/student/troubleshoot-review/** 目录。它也会将 **index.html** 文件下载到 **/home/student/troubleshoot-review/html/** 目录。

1. 从 **~/troubleshoot-review** 目录，检查 **secure-web.yml** playbook 的语法。此 playbook 包含的一个 play 用于为 **webservers** 组中的主机设置使用 TLS/SSL 的 Apache HTTPD。修复报告的问题。
2. 再次检查 playbook **secure-web.yml** 的语法。修复报告的问题。
3. 第三次检查 playbook **secure-web.yml** 的语法。修复报告的问题。
4. 第四次检查 playbook **secure-web.yml** 的语法。它应该不会显示任何语法错误。
5. 运行 **secure-web.yml** playbook。Ansible 无法连接 **serverb.lab.example.com**。修复这个问题。
6. 再次运行 **secure-web.yml** playbook。Ansible 应在受管主机上以 **devops** 远程用户身份通过身份验证。修复此问题。
7. 第三次运行 **secure-web.yml** playbook。修复报告的问题。
8. 再一次运行 **secure-web.yml** playbook。它应该成功完成。使用临时命令验证 **httpd** 服务正在运行。

评估

在 workstation 上，运行 **lab troubleshoot-review grade** 脚本来确认是否成功完成练习。

```
[student@workstation ~]$ lab troubleshoot-review grade
```

完成

在 workstation 上，运行 `lab troubleshoot-review finish` 脚本来清理本实验。

```
[student@workstation ~]$ lab troubleshoot-review finish
```

本实验到此结束。

► 解决方案

对 ANSIBLE 进行故障排除

任务执行清单

在本实验中，您将对尝试运行提供给您的 playbook 时出现的问题进行故障排除。

成果

您应能够：

- 对 playbook 进行故障排除。
- 对受管主机进行故障排除。

在你开始之前

以 student 用户身份并使用 student 作为密码登录 workstation。运行 **lab troubleshoot-review start** 命令。

```
[student@workstation ~]$ lab troubleshoot-review start
```

此脚本可确保在 workstation 上安装 Ansible，并且创建 **~student/troubleshoot-review/html/** 目录。它将 **ansible.cfg**、**inventory-lab** 和 **secure-web.yml** 文件从 **vhosts.conf** 下载从 **http://materials.example.com/labs/troubleshoot-review/** 下载到 **/home/student/troubleshoot-review/** 目录。它也会将 **index.html** 文件下载到 **/home/student/troubleshoot-review/html/** 目录。

1. 从 **~/troubleshoot-review** 目录，检查 **secure-web.yml** playbook 的语法。此 playbook 包含的一个 play 用于为 **webservers** 组中的主机设置使用 TLS/SSL 的 Apache HTTPD。修复报告的问题。
 - 1.1. 在 workstation 上，更改到 **/home/student/troubleshoot-review** 项目目录。

```
[student@workstation ~]$ cd ~/troubleshoot-review/
```

- 1.2. 检查 **secure-web.yml** playbook 的语法。当没有任何错误的时候，这个剧本会为 **webservers** 组中的主机设置 Apache HTTPD 的 TLS/SSL。

```
[student@workstation troubleshoot-review]$ ansible-playbook --syntax-check \
> secure-web.yml
```

```
ERROR! Syntax Error while loading YAML.
mapping values are not allowed in this context
```

```
The error appears to have been in '/home/student/Ansible-course/troubleshoot-
review/secure-web.yml': line 7, column 30, but may
be elsewhere in the file depending on the exact syntax problem.
```

The offending line appears to be:

```
vars:  
    random_var: This is colon: test  
                ^ here
```

1.3. 向 `This is colon: test` 字符串添加双引号，更正 `random_var` 变量定义中的语法问题。发生的更改应如下所示：

```
...output omitted...  
vars:  
    random_var: "This is colon: test"  
...output omitted...
```

2. 再次检查 playbook `secure-web.yml` 的语法。修复报告的问题。

2.1. 再次使用 `ansible-playbook --syntax-check` 检查 `secure-web.yml` 的语法。

```
[student@workstation troubleshoot-review]$ ansible-playbook --syntax-check \  
> secure-web.yml
```

```
ERROR! Syntax Error while loading YAML.  
      did not find expected '-' indicator
```

```
The error appears to have been in '/home/student/Ansible-course/troubleshoot-  
review/secure-web.yml': line 38, column 10, but may  
be elsewhere in the file depending on the exact syntax problem.
```

The offending line appears to be:

```
- name: start and enable web services  
^ here
```

2.2. 更正缩进中存在的所有语法问题。删除 `start and enable web services` 任务元素开头处多余的空格。发生的更改应如下所示：

```
...output omitted...  
args:  
    creates: /etc/pki/tls/certs/serverb.lab.example.com.crt  
  
    - name: start and enable web services  
      service:  
        name: httpd  
        state: started  
        enabled: yes  
  
    - name: deliver content  
      copy:  
        dest: /var/www/vhosts/serverb-secure  
        src: html/  
...output omitted...
```

3. 第三次检查 playbook `secure-web.yml` 的语法。修复报告的问题。

3.1. 检查 `secure-web.yml` playbook 的语法。

```
[student@workstation troubleshoot-review]$ ansible-playbook --syntax-check \
> secure-web.yml

ERROR! Syntax Error while loading YAML.
  found unacceptable key (unhashable type: 'AnsibleMapping')

The error appears to have been in '/home/student/Ansible-course/troubleshoot-
review/secure-web.yml': line 13, column 20, but may
be elsewhere in the file depending on the exact syntax problem.
```

The offending line appears to be:

```
yum:
  name: {{ item }}
    ^ here
```

We could be wrong, but this one looks like it might be an issue with missing quotes. Always quote template expression brackets when they start a value. For instance:

```
with_items:
  - {{ foo }}
```

Should be written as:

```
with_items:
  - "{{ foo }}"
```

3.2. 更正 `install web server packages` 任务中的 `item` 变量。添加双引号到 `{{ item }}`。发生的更改应如下所示：

```
...output omitted...
  - name: install web server packages
    yum:
      name: "{{ item }}"
      state: latest
    notify:
      - restart services
  loop:
    - httpd
    - mod_ssl
...output omitted...
```

4. 第四次检查 playbook `secure-web.yml` 的语法。它应该不会显示任何语法错误。

4.1. 检查 `secure-web.yml` playbook 的语法。它应该不会显示任何语法错误。

```
[student@workstation troubleshoot-review]$ ansible-playbook --syntax-check \
> secure-web.yml

playbook: secure-web.yml
```

5. 运行 **secure-web.yml** playbook。Ansible 无法连接 `serverb.lab.example.com`。修复这个问题。

5.1. 运行 **secure-web.yml** playbook。这将失败并报出错误。

```
[student@workstation troubleshoot-review]$ ansible-playbook secure-web.yml
PLAY [create secure web service] ****
TASK [Gathering Facts] ****
fatal: [serverb.lab.example.com]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to the host via ssh: students@serverc.lab.example.com: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).", "unreachable": true}

PLAY RECAP ****
serverb.lab.example.com : ok=0    changed=0    unreachable=1    failed=0
```

- 5.2. 再次运行 **secure-web.yml** playbook，这一次添加 `-vvvv` 参数来提高输出的详细程度。

注意到 Ansible 似乎在连接 `serverc.lab.example.com`，而非 `serverb.lab.example.com`。

```
[student@workstation troubleshoot-review]$ ansible-playbook secure-web.yml -vvvv
...output omitted...
TASK [Gathering Facts] ****
task path: /home/student/troubleshoot-review/secure-web.yml:3
<serverc.lab.example.com> ESTABLISH SSH CONNECTION FOR USER: students
<serverc.lab.example.com> SSH: EXEC ssh -vvv -C -o ControlMaster=auto
-o ControlPersist=60s -o KbdInteractiveAuthentication=no -o
PreferredAuthentications=gssapi-with-mic,gssapi-keyex,hostbased,publickey -o
PasswordAuthentication=no -o User=students -o ConnectTimeout=10 -o ControlPath=/home/student/.ansible/cp/bc0c05136a serverc.lab.example.com '/bin/sh -c '""'echo
~students && sleep 0'""''
...output omitted...
```

- 5.3. 更正 **inventory-lab** 文件中相应的行。删除 `ansible_host` 主机变量，使文件显示为如下所示：

```
[webservers]
serverb.lab.example.com
```

6. 再次运行 **secure-web.yml** playbook。Ansible 应在受管主机上以 `devops` 远程用户身份通过身份验证。修复此问题。

6.1. 运行 **secure-web.yml** playbook。

```
[student@workstation troubleshoot-review]$ ansible-playbook secure-web.yml -vvvv
...output omitted...
TASK [Gathering Facts] ****
task path: /home/student/troubleshoot-review/secure-web.yml:3
<serverb.lab.example.com> ESTABLISH SSH CONNECTION FOR USER: students
<serverb.lab.example.com> EXEC ssh -C -vvv -o ControlMaster=auto
-o ControlPersist=60s -o Port=22 -o KbdInteractiveAuthentication=no
```

```

-o PreferredAuthentications=gssapi-with-mic,gssapi-keyex,hostbased,publickey
-o PasswordAuthentication=no -o User=students -o ConnectTimeout=10
-o ControlPath=/home/student/.ansible/cp/ansible-ssh-%C -tt
serverb.lab.example.com '/bin/sh -c """( umask 22 && mkdir -p `
echo $HOME/.ansible/tmp/ansible-tmp-1460241127.16-3182613343880 `" &&
echo "` echo $HOME/.ansible/tmp/ansible-tmp-1460241127.16-3182613343880
`" )"""
...output omitted...
fatal: [serverb.lab.example.com]: UNREACHABLE! => {
...output omitted...

```

6.2. 编辑 **secure-web.yml** playbook，以确保 **devops** 是 play 的 **remote_user**。Playbook 前面几行应如下所示：

```

---
# start of secure web server playbook
- name: create secure web service
  hosts: webservers
  remote_user: devops
...output omitted...

```

7. 第三次运行 **secure-web.yml** playbook。修复报告的问题。

7.1. 运行 **secure-web.yml** playbook。

```

[student@workstation troubleshoot-review]$ ansible-playbook secure-web.yml -vvvv
...output omitted...
failed: [serverb.lab.example.com] (item=mod_ssl) => {
    "ansible_loop_var": "item",
    "changed": false,
    "invocation": {
        "module_args": {
            "allow_downgrade": false,
            "autoremove": false,
...output omitted...
            "validate_certs": true
        }
    },
    "item": "mod_ssl",
    "msg": "This command has to be run under the root user.",
    "results": []
}
...output omitted...

```

7.2. 编辑 play，确保其设置了 **become: true** 或 **become: yes**。发生的更改应如下所示：

```

---
# start of secure web server playbook
- name: create secure web service
  hosts: webservers
  remote_user: devops
  become: true
...output omitted...

```

8. 再一次运行 `secure-web.yml` playbook。它应该成功完成。使用临时命令验证 `httpd` 服务正在运行。

8.1. 运行 `secure-web.yml` playbook。

```
[student@workstation troubleshoot-review]$ ansible-playbook secure-web.yml
PLAY [create secure web service] ****
...output omitted...
TASK [install web server packages] ****
changed: [serverb.lab.example.com] => (item=httpd)
changed: [serverb.lab.example.com] => (item=mod_ssl)
...output omitted...
TASK [httpd_conf_syntax variable] ****
ok: [serverb.lab.example.com] => {
    "msg": "The httpd_conf_syntax variable value is {'stderr_lines': [u'Syntax OK'], u'changed': True, u'end': u'2018-12-17 23:31:53.191871', 'failed': False, u'stdout': u'', u'cmd': [u'/sbin/httpd', u'-t'], u'rc': 0, u'start': u'2018-12-17 23:31:53.149759', u'stderr': u'Syntax OK', u'delta': u'0:00:00.042112', 'stdout_lines': [], 'failed_when_result': False}"
}
...output omitted...
RUNNING HANDLER [restart services] ****
changed: [serverb.lab.example.com]

PLAY RECAP ****
serverb.lab.example.com : ok=10    changed=7     unreachable=0    failed=0
```

- 8.2. 使用一个临时命令来确定 `serverb.lab.example.com` 上 `httpd` 服务的状态。`httpd` 服务现在应当已在 `serverb.lab.example.com` 上运行。

```
[student@workstation troubleshoot-review]$ ansible all -u devops -b \
> -m command -a 'systemctl status httpd'
serverb.lab.example.com | CHANGED | rc=0 >>
● httpd.service - The Apache HTTP Server
    Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset:
              disabled)
    Active: active (running) since Thu 2019-04-11 03:22:34 EDT; 28s ago
...output omitted...
```

评估

在 `workstation` 上，运行 `lab troubleshoot-review grade` 脚本来确认是否成功完成本练习。

```
[student@workstation ~]$ lab troubleshoot-review grade
```

完成

在 `workstation` 上，运行 `lab troubleshoot-review finish` 脚本来清理本实验。

```
[student@workstation ~]$ lab troubleshoot-review finish
```

总结

在本章中，您学到了：

- Ansible 提供了内置日志记录功能。默认情况下，不启用此功能。
- **ansible.cfg** 配置文件 **default** 部分中的 **log_path** 参数指定所有 Ansible 输出重定向到的日志文件位置。
- **debug** 模块提供运行 playbook 时的额外调试信息（如变量的当前值）。
- **ansible-playbook** 命令的 **-v** 选项提供多种级别的输出详细程度。这可在运行 playbook 时用于调试 Ansible 任务。
- **--check** 选项使得支持检查模式的 Ansible 模块能够显示要执行的更改，而不将这些更改应用到受管主机上。
- 可以使用临时命令对受管主机执行其他检查。