在编写playbook时，可能会涉及到一些敏感的数据，比如密码，当这些敏感数据以明文的方式存储在playbook中时，可能是不能被接受的，那么我们该怎么办呢？ansible官方已经考虑到了这种情况，当我们的playbook中含有不能明文展示的文本时，我们可以使用"ansible-vault"命令，对敏感数据进行加密，我们可以对整个文件加密，也可以对某个字符串加密（也就是变量加密），那么具体操作是怎样的呢？我们一起来了解一下吧。

假如，我们已经编写好了一个playbook文件，内容如下

```
# cat test.yml
- hosts: test70
  tasks:
  - debug:
      msg: "Test ansible-vault"
```

如你所见，整个test.yml的内容都是明文的，如果我想要对这个剧本进行加密，可以使用如下命令

```
ansible-vault encrypt test.yml
```

"encrypt"是"ansible-vault"的子命令，如果我们想要加密某个文件，则可以使用"ansible-vault encrypt"命令，执行上述命令后，会提示输入密码，你需要记住密码，因为如果你想要运行加密过的剧本，或者解密，都需要使用这个密码（也可以使用密码文件，之后会有示例），输入两遍密码后，即可看到"Encryption successful"的字样，表示加密成功，此刻如果再次查看test.yml文件的内容，可以发现，已经变成了如下内容：

```
# cat test.yml
$ANSIBLE_VAULT;1.1;AES256
64323634303535336563333064663033393037316462363334656334396562643736663839386464
30623732666261653062386132646332306238376334366660a3566386334363133332643735613335
31333935336437633064323761613632396631643334363730663131656661613063333265363838
31393065326137396603861303462326561323663303831313236376135333237336464373663331
63663939396234376362336164663665326162323262313139383364373038636562306163636362
33396434663731356239303162656466343031316161346166373037666130353831393261313530
31646366353836303439323738323032306164623338346433323433623538353863633563633266
34313334623637336535
```

可以看到，test.yml文件的内容已经被加密，那么，如果此刻我们想要执行test.yml，能不能正常执行呢？我们来试试

```
# ansible-playbook test.yml
ERROR! Attempting to decrypt but no vault secrets found
```

可以看到，直接调用加密过的剧本，会报错，因为ansible并不知道解密的密码，我们可以借助"--ask-vault-pass"选项，在运行加密的剧本时输入对应的密码，示例如下

```
# ansible-playbook --ask-vault-pass test.yml
```

输入上述命令后，会提示你输入密码，密码就是加密时所使用的密码，输入密码后，即可正常执行test.yml文件。

如果你想要还原一个加密过的文件，或者说解密一个加密过的文件，可以使用"ansible-vault"的另一个子命令，"decrypt"子命令，见名知意，"decrypt"子命令就是用来解密的，示例如下：

```
# ansible-vault decrypt test.yml
```

输入上述命令后，会提示你输入密码，就是你加密时所使用的密码，输入密码后会看到"Decryption successful"字样，表示解密成功，再次查看test.yml文件，可以发现，其中的内容已经被还原成了明文内容。

你可能会有疑问，难道每次加密和解密，都需要手动的输入密码么？难道就没有不用手动输入密码的方法么？必须有啊，我们可以将密码保存在某个文件中，然后在加密或者解密时，指定这个密码文件就好了，这样我们就不用手动的输入密码了，示例如下：
首先，我将密码写入到pwdfile文件中，如下：

```
# echo "123123" > pwdfile
```

然后，使用这个密码文件加密对应的playbook，如果想要使用密码文件，需要借助一个选项，它就是"--vault-password-file"选项，通过“--vault-password-file”来指定加密所需的密码文件，如下：

```
# ansible-vault encrypt --vault-password-file pwdfile test.yml
```

可以看到，我们在使用ansible-vault encrypt命令时，使用"--vault-password-file"选项指定了pwdfile文件作为密码文件，表示使用pwdfile文件中的文本作为密码对test.yml进行加密，当我们需要运行加密过的剧本、或者解密时，同样可以使用"--vault-password-file"选项，指定对应的密码文件进行解密，如下：

```
# ansible-playbook --vault-password-file pwdfile test.yml
```

```
# ansible-vault decrypt --vault-password-file pwdfile test.yml
```

这样，我们就不用在加密和解密时手动的输入密码了。
从ansible2.4版本开始，官方不再推荐使用"--vault-password-file"选项，官方开始推荐使用"--vault-id"选项代替"--vault-password-file"选项指定密码文件，也就是说，如下两条命令的效果是一样的。

```
# ansible-vault encrypt --vault-id pwdfile test.yml
# ansible-vault decrypt --vault-password-file pwdfile test.yml
```

当然，在运行加密过的脚本和解密时，也可以使用"--vault-id"选项指定密码文件

```
# ansible-playbook --vault-id pwdfile test.yml
# ansible-vault decrypt --vault-id pwdfile test.yml
```

其实，"--vault-id"选项不仅能够代替"--vault-password-file"选项，还能够代替"--ask-vault-pass"选项，在之前的示例中已经演示过，当调用加密过的剧本时，可以使用"--ask-vault-pass"选项，交互式的输入密码，"--vault-id"选项可以实现同样的功能，示例如下：

```
# ansible-playbook --vault-id prompt test.yml
```

执行上述命令后，同样会交互式的提示用户输入密码，输入正确的密码后，即可正常的运行加密过的剧本，也就是说，如下两条命令的效果是完全相同的。

```
# ansible-playbook --vault-id prompt test.yml
# ansible-playbook --ask-vault-pass test.yml
```

2.4版本以后的ansible中，"--vault-id"选项支持同时使用多个密码文件进行解密，什么意思呢？我们先来描述一个工作场景，如下：
现在我有两个yml文件，test.yml和test1.yml，这两个yml文件的内容分别如下

```
# cat test.yml
- hosts: test70
  tasks:
  - debug:
      msg: "message from test"
  - include_tasks: test1.yml

# cat test1.yml
- debug:
    msg: "message from test1"
```

如上述示例所示，test.yml包含了test1.yml，所以，当我们执行test.yml时，test1.yml也会被调用。
同时，我准备了两个密码文件，分别存放了不同的密码

```
# echo "123123" > pwdfile
# echo "123456" > pwdfile1
```

现在我要做的是，分别用两个密码文件加密这两个yml文件，操作如下：

```
# ansible-vault encrypt --vault-id pwdfile test.yml
# ansible-vault encrypt --vault-id pwdfile1 test1.yml
```

现在两个yml文件都被加密了，而且使用了不同的密码，如果此时，我想要运行test.yml，会出现什么问题么？聪明如你一定想到了，因为test.yml包含了test1.yml，所以当我们调用test.yml时，也会调用test1.yml，但是我们使用了不同的密码加密了这两个yml文件，所以，当我们想要运行它们时，必须同时提供两个密码文件，命令如下：

```
# ansible-playbook --vault-id pwdfile1 --vault-id pwdfile test.yml
```

你也可以一次性使用不同的密码文件解密不同的文件，示例如下：

```
# ansible-vault decrypt --vault-id pwdfile1 --vault-id pwdfile test.yml test1.yml
```

执行上述命令时，你不用纠结密码文件与加密文件的对应关系，ansible会自动尝试这些密码文件。
你甚至可以使用如下命令，使用交互式的方式，一次性的输入多个文件的解密密码，但是使用如下命令输入密码时，需要注意对应顺序。

```
# ansible-vault view --vault-id prompt --vault-id prompt test.yml test1.yml
```

其实，" --vault-id"选项还有一个小功能，就是在加密文件时，给被加密的文件"做记号"，什么意思呢？来看一个小栗子，如下：

```
# ansible-vault encrypt --vault-id zsy@pwdfile test.yml
```

上述命令表示对test.yml文件进行加密，使用pwdfile文件中的内容作为密码，同时，在加密test.yml文件时，加入了"zsy"这个小记号，那么加密完成后，查看加密后的test.yml文件内容如下：

```
# cat test.yml
$ANSIBLE_VAULT;1.2;AES256;zsy
65633737626662646664343335303732383437626634306261326636336261303935316431626437
33626539393037336465333566656437373338303238333370a3635306238653538316239363764 63
31343961313638393865373061623439376632383038386464386662643935656261656130636135
61333665393864333 70a3661366461626265323033636364663666633730343839326430353137 61
32346538656532323434613435393137633731383561653163372336263666236623562366436 565
61666537316137323936613237663639333461333534653336313731653331323434666434663831
63323239373463626534393063383365666438363737653535333430636232336634663064393462
61623266373735373066316663303533633638353762653630323833376535666134316136356639
61386437656562383965656162376434666439633134643665393637663639363133
```

可以看到，加密后的test.yml的第一行内容的结尾，就是我们加入的"小记号"。

这些记号并不会对加密和解密的过程产生影响，只是为了方便管理，如果你是管理员，可能通过一些记号，能够更方便的对这些加密过的内容进行标识吧。

你也可以在交互输入密码时添加记号，比如添加一个"记号"，zsythink，命令如下：

```
# ansible-vault encrypt --vault-id zsythink@prompt test.yml
```

刚才我们只介绍了ansible-vault的两个子命令，encrypt子命令和decrypt子命令，其实ansible-vault还有一些其他的子命令，这些子命令分别对应了不同的功能，我们来认识一下它们。

create子命令

使用create子命令，可以创建一个被加密的文件

```
# ansible-vault create test
```

执行上述命令后，会提示你输入密码，确认密码，然后默认调用vi编辑器，提示你输入内容，你输入的内容将会被保存到test文件中，并且在退出编辑器时自动将test文件加密，也就是说，create子命令的作用就是创建一个文件，等待你写入内容后使用ansible-vault进行加密。

view子命令

使用view子命令，可以查看已经被加密过的文件的原内容，但是不会对文件本身进行还原操作，只是查看原内容。

```
# ansible-vault view test.yml
# ansible-vault view --vault-id pwdfile test.yml
```

edit子命令

使用edit子命令，可以直接修改被加密过的文件的原内容，使用edit子命令修改被加密过的文件内容的过程相当于：先解密、修改原内容，再加密

```
# ansible-vault edit test.yml
# ansible-vault edit --vault-id pwdfile test.yml
```

rekey子命令

使用rekey子命令，可以修改被加密文件的密码，比如，一开始我使用了123123这个密码对test.yml文件进行了加密，现在，我想把密码换成123456，执行如下命令即可

```
# ansible-vault rekey test.yml
```

执行上述命令后，一共会提示你输入3次密码，第一次输入老密码，也就是123123，之后两次输入新密码，也就是123456，修改成功后，以后都适用新密码进行解密。

当然，如果你之前是使用的密码文件的方式进行的加密，也可以使用rekey子命令重新指定一个新的密码文件，但是需要借助"--new-vault-id"选项或者"--new-vault-password-file"选项，通过这两个选项的任何一个，都可以指定新的密码文件。

```
# ansible-vault rekey --vault-id pwdfile --new-vault-id pwdfile1 test.yml
```

encrypt_string子命令

刚才介绍的方法都是对整个文件进行加密，但是通常，我们并不需要加密整个文件，加密整个文件后，反而可能会对我们的阅读造成困扰，有时我们只是想把"密码隐藏起来"而已。

从2.3版本开始，使用encrypt_string子命令，可以加密"字符串"，通过加密字符串的功能，能够有效的隐藏敏感变量的值，比如，隐藏变量列表中密码变量的值，假设，我现在的playbook如下：

```
# cat test.yml
- hosts: test71
  vars:
    test_user: "testuser"
    test_passwd: "123456"
  tasks:
  - debug:
      msg: "{{test_user}}"
  - debug:
      msg: "{{test_passwd}}"
```

我觉得test_passwd这个密码变量直接以明文的方式存储在playbook中不太安全，因为所有有权限查看该playbook的人都能直接看到密码，我们需要对密码字符串的值（也就是123456）进行加密，以保证它不会明文显示在这里，所以，我们需要借助到"ansible-vault encrypt_string"命令，操作如下：

注：我们从最原始的操作开始，以免产生疑问

```
# ansible-vault encrypt_string 123456
```

上例表示，使用"ansible-vault encrypt_string"命令对"123456"这个字符串进行加密，加密时，会提示你输入密码，你输入的密码用于加密和解密字符串，此处，我输入了"aaaa"作为加密解密的密码，输入密码后，"ansible-vault encrypt_string"命令会将加密后的字符串输入到屏幕中，如下：

```
# ansible-vault encrypt_string 123456
New Vault password:
Confirm New Vault password:
!vault |
          $ANSIBLE_VAULT;1.1;AES256
          30316633646364663764333666383437373439353538353336623532323131623739353663653637
          34306266373862313662366430346433653237383362311330a32653462303936303039373966323 7
          65623635616666656233333337636439366535383334393138623231613035373133323832383335
          37373862343637613 50a34383932666362666439643633646539386261323739386431653366353 3
          6335
```

如你所见，这返回的一长串文本就是"123456"加密后的文本，现在，复制这串文本，用这串文本替换playbook中的"123456"，替换后的playbook如下：

```
# cat test.yml
- hosts: test71
  vars:
    test_user: "testuser"
    test_passwd: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          30316633646364663764333666383437373439353538353336623532323131623739353663653637
          34306266373862313662366430346433653237383362311330a32653462303936303039373966323 7
          65623635616666656233333337636439366535383334393138623231613035373133323832383335
          37373862343637613 50a34383932666362666439643633646539386261323739386431653366353 3
          6335
  tasks:
  - debug:
      msg: "{{test_user}}"
  - debug:
      msg: "{{test_passwd}}"
```

此刻，"123456"这串明文字符串已经被替换为了加密后的字符串，那么我们来运行一下这个playbook，由于上文中已经说明了各个选项的用法，所以此处不再赘述，使用如下两条命令的效果是相同的，都会提示你输入加密时所使用的密码（也就是"aaaa"）：

```
# ansible-playbook --ask-vault-pass test.yml
# ansible-playbook --vault-id prompt test.yml
```

输入"aaaa"后，即可正常执行playbook，执行结果如下，可以从如下结果看出，字符串已经被正常解密了，获取到了我们原来设置的值，也就是"123456"，这样我们就能做到在运行时获取到真正的"字符串原文"，而在playbook中不再显示明文字符串了。

```
# ansible-playbook --vault-id prompt test.yml
Vault password (default):

PLAY [test71] ************************************

TASK [Gathering Facts] *****************************
ok: [test71]

TASK [debug] ***************************************
ok: [test71] => {
    "msg": "testuser"
}

TASK [debug] ***************************************
ok: [test71] => {
    "msg": "123456"
}

PLAY RECAP *****************************************
test71                     : ok=3    changed=0    unreachable=0    failed=0
```

聪明如你一定想到了，当我们加密字符串或者解密字符串时，可以使用"--vault-id"选项或者"--vault-password-file"选项指定"密码文件"，以免手动的输入加密时的密码，示例如下：

```
# echo aaaa > pwdfile
# ansible-vault encrypt_string --vault-id pwdfile 123456
# ansible-playbook --vault-id pwdfile test.yml
```

使用密码文件的方式是最常见的，因为我们不可能在自动化的过程中手动的输入密码进行解密，所以密码文件的权限一定要控制好，无论是放在git上或者放在jenkins上，都应该做好权限控制。

"encrypt_string"子命令还有一个选项，能够设置加密后的字符串的变量名，它就是"--name"选项，示例如下

```
# ansible-vault encrypt_string --vault-id pwdfile --name test_passwd 123456
test_passwd: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          36396366336238376662353664383836316366383937623830626635613063343764333962376466
          38356461613633643035633734386437326262313035643 20a39323333346166356238373364316 6
          62313362623838336433303032376565343264356665323832623565653631386536383762633764
          396161326536633630 0a376564633034376238363664653565316163313739343639643565306665
          6264
Encryption successful
```

如你所见，我使用了"--name"选项，指定了变量名"test_passwd"，那么最终生成的结果的格式就是 "变量名：加密后的字符串"，其实与不使用"--name"选项时没有太大的区别，不过这样比较方便复制，你可以直接将生成的结果复制到playbook中，因为变量名已经生成了。

上文总结的选项你也可以灵活的使用，比如在加密字符串时添加一个"小记号"

```
# ansible-vault encrypt_string --vault-id zsy@pwdfile --name test_passwd 123456
```