Automation

Ansible Vault

DE HOGESCHOOL
MET HET NETWERK

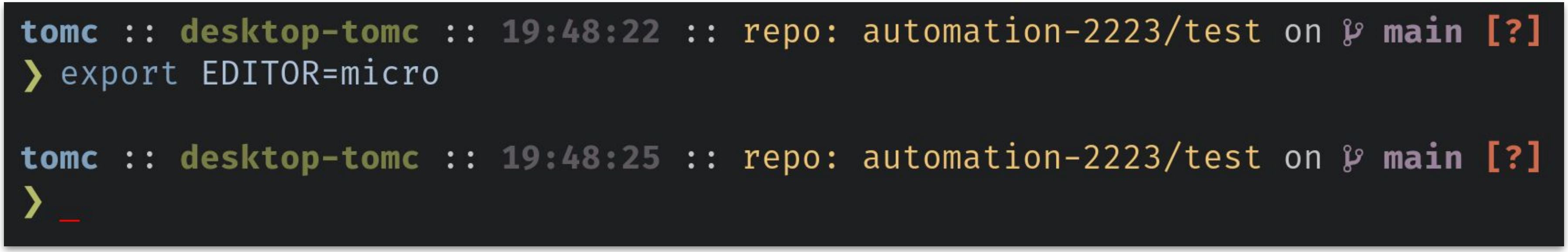
Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be

Ansible Vault: Securely Encrypting Content in Ansible Workflows

- Ansible Vault transparently encrypts content in Ansible workflows.
- The **`ansible-vault`** utility encrypts secrets at rest on disk.
- **`ansible`** and **`ansible-playbook`** commands can decrypt vault-encrypted content at runtime.
- Vault encrypts files at the file level using AES256 symmetric encryption and a user-defined password.
- Ansible can identify and decrypt all files with vault encryption during playbook/task execution.

Ansible Vault Editor

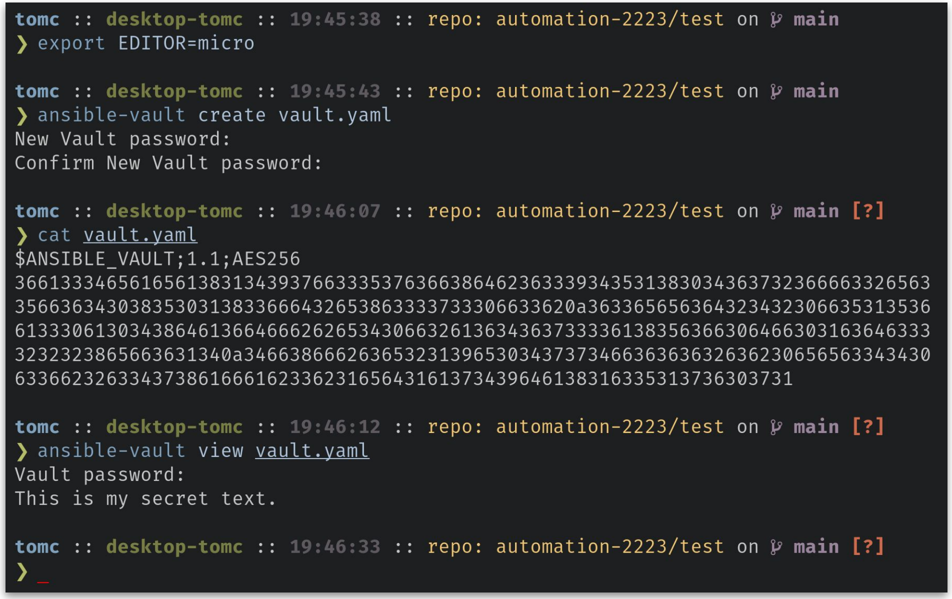
Set your favorite editor with the env variable EDITOR, which you can make persistent in your .bashrc. Default is vi.



```
tomc :: desktop-tomc :: 19:48:22 :: repo: automation-2223/test on  main [?]
❯ export EDITOR=micro

tomc :: desktop-tomc :: 19:48:25 :: repo: automation-2223/test on  main [?]
❯ _
```

Managing Encrypted Content with ansible-vault

- The **`ansible-vault`** command is the primary interface for managing encrypted content in Ansible.
- Use **`ansible-vault create`** to create a new file with Vault, specifying the file name.
- **`ansible-vault view`** displays content of file on standard out.

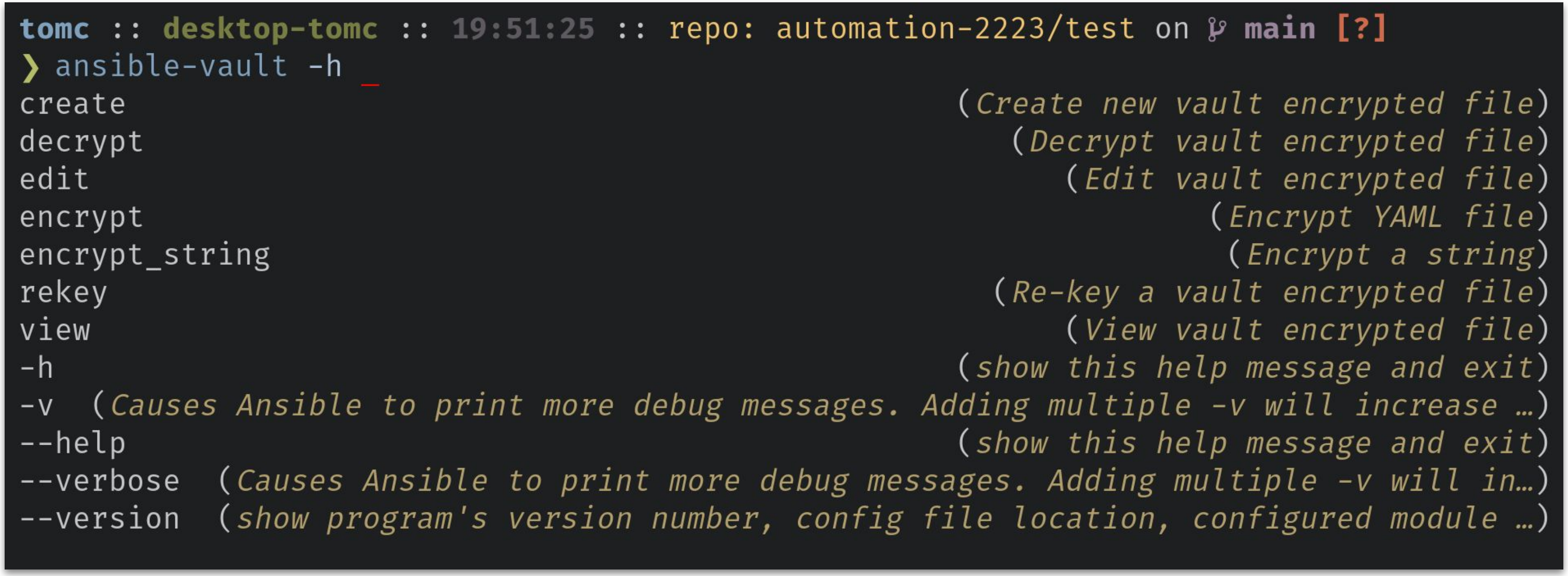


```
tomc :: desktop-tomc :: 19:45:38 :: repo: automation-2223/test on  main
❯ export EDITOR=micro

tomc :: desktop-tomc :: 19:45:43 :: repo: automation-2223/test on  main
❯ ansible-vault create vault.yaml
New Vault password:
Confirm New Vault password:

tomc :: desktop-tomc :: 19:46:07 :: repo: automation-2223/test on  main [?]
❯ cat vault.yaml
$ANSIBLE_VAULT;1.1;AES256
36613334656165613831343937663335376366386462363339343531383034363732366663326563
3566363430383530313833666432653863333733306633620a363365656364323432306635313536
61333061303438646136646662626534306632613634363733336138356366306466303163646333
3232323865663631340a346638666263653231396530343737346636363632636230656563343430
63366232633437386166616233623165643161373439646138316335313736303731

tomc :: desktop-tomc :: 19:46:12 :: repo: automation-2223/test on  main [?]
❯ ansible-vault view vault.yaml
Vault password:
This is my secret text.

tomc :: desktop-tomc :: 19:46:33 :: repo: automation-2223/test on  main [?]
❯ _
```

Other vault operations

Limitation: ONE Vault password per Ansible playbook

`ansible-vault <operation>`



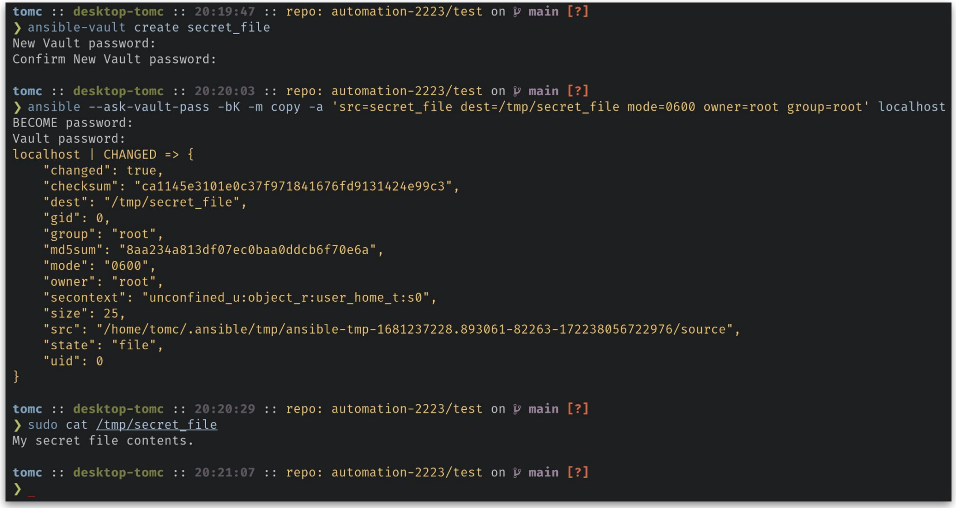
```
tomc :: desktop-tomc :: 19:51:25 :: repo: automation-2223/test on ᚠ main [?]
❯ ansible-vault -h _
create                                                    (Create new vault encrypted file)
decrypt                                                      (Decrypt vault encrypted file)
edit                                                            (Edit vault encrypted file)
encrypt                                                                  (Encrypt YAML file)
encrypt_string                                                            (Encrypt a string)
rekey                                                        (Re-key a vault encrypted file)
view                                                            (View vault encrypted file)
-h                                                        (show this help message and exit)
-v  (Causes Ansible to print more debug messages. Adding multiple -v will increase …)
--help                                                    (show this help message and exit)
--verbose  (Causes Ansible to print more debug messages. Adding multiple -v will in…)
--version  (show program's version number, config file location, configured module …)
```

Using encrypted vault files with Ansible

- Encrypted files can be used transparently with conventional Ansible tooling after encrypting with Vault.
- Both **`ansible`** and **`ansible-playbook`** commands can decrypt Vault-encrypted files with the correct password.
- Several ways to provide passwords to these commands.

Decrypting Vault data with an interactive prompt

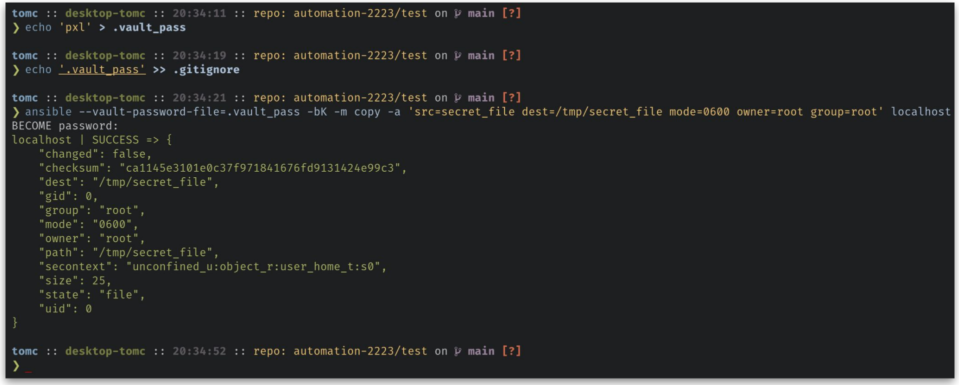
- Simplest way to decrypt content at runtime is to have Ansible prompt for the right credentials.
- Add `--ask-vault-pass` to `ansible` or `ansible-playbook` command to prompt for password.
- Ansible will use the password to attempt to decrypt all vault-encrypted content it finds.
- `ansible --ask-vault-pass -bK -m copy -a 'src=secret_key dest=/tmp/secret_file mode=0600 owner=root group=root' localhost`
    - to copy content of vault-encrypted file to host, use the copy module with the `--ask-vault-pass` flag
    - If the file contains sensitive data, do lock down access/ownership on remote host with appropriate restrictions.
    - Task specifies ownership of file should be changed to root, so administrative rights are required
    - `-bK` flag prompts for sudo (become) password on target host



```
tomc :: desktop-tomc :: 20:19:47 :: repo: automation-2223/test on ᚴ main [?]
❯ ansible-vault create secret_file
New Vault password:
Confirm New Vault password:

tomc :: desktop-tomc :: 20:20:03 :: repo: automation-2223/test on ᚴ main [?]
❯ ansible --ask-vault-pass -bK -m copy -a 'src=secret_file dest=/tmp/secret_file mode=0600 owner=root group=root' localhost
BECOME password:
Vault password:
localhost | CHANGED => {
    "changed": true,
    "checksum": "ca1145e3101e0c37f971841676fd9131424e99c3",
    "dest": "/tmp/secret_file",
    "gid": 0,
    "group": "root",
    "md5sum": "8aa234a813df07ec0baa0ddcb6f70e6a",
    "mode": "0600",
    "owner": "root",
    "secontext": "unconfined_u:object_r:user_home_t:s0",
    "size": 25,
    "src": "/home/tomc/.ansible/tmp/ansible-tmp-1681237228.893061-82263-172238056722976/source",
    "state": "file",
    "uid": 0
}

tomc :: desktop-tomc :: 20:20:29 :: repo: automation-2223/test on ᚴ main [?]
❯ sudo cat /tmp/secret_file
My secret file contents.

tomc :: desktop-tomc :: 20:21:07 :: repo: automation-2223/test on ᚴ main [?]
❯ _
```

Decrypting Vault data with a password file

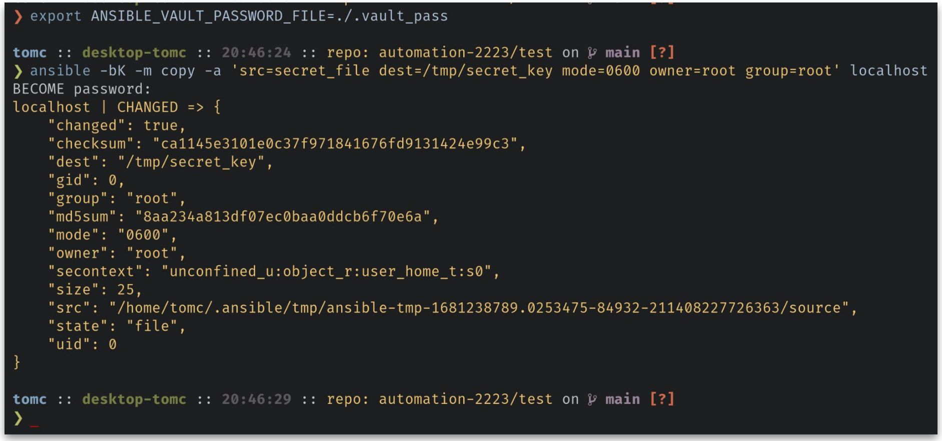
- To avoid entering the Vault password every time you run a task, add it to a file and reference that file during execution.
- If using version control, add the password file to the .ignore file to prevent accidental inclusion in the repo.
- Use the `--vault-password-file` flag to reference the password file during execution.
- `ansible --vault-password-file=.vault_pass -bK -m copy -a 'src=secret_file dest=/tmp/secret_key mode=0600 owner=root group=root' localhost`



```
tomc :: desktop-tomc :: 20:34:11 :: repo: automation-2223/test on main [?]
❯ echo 'pxl' > .vault_pass

tomc :: desktop-tomc :: 20:34:19 :: repo: automation-2223/test on main [?]
❯ echo '.vault_pass' >> .gitignore

tomc :: desktop-tomc :: 20:34:21 :: repo: automation-2223/test on main [?]
❯ ansible --vault-password-file=.vault_pass -bK -m copy -a 'src=secret_file dest=/tmp/secret_file mode=0600 owner=root group=root' localhost
BECOME password:
localhost | SUCCESS => {
    "changed": false,
    "checksum": "ca1145e3101e0c37f971841676fd9131424e99c3",
    "dest": "/tmp/secret_file",
    "gid": 0,
    "group": "root",
    "mode": "0600",
    "owner": "root",
    "path": "/tmp/secret_file",
    "secontext": "unconfined_u:object_r:user_home_t:s0",
    "size": 25,
    "state": "file",
    "uid": 0
}

tomc :: desktop-tomc :: 20:34:52 :: repo: automation-2223/test on main [?]
❯ _
```

Decrypting Vault data with an 'automatic' password file

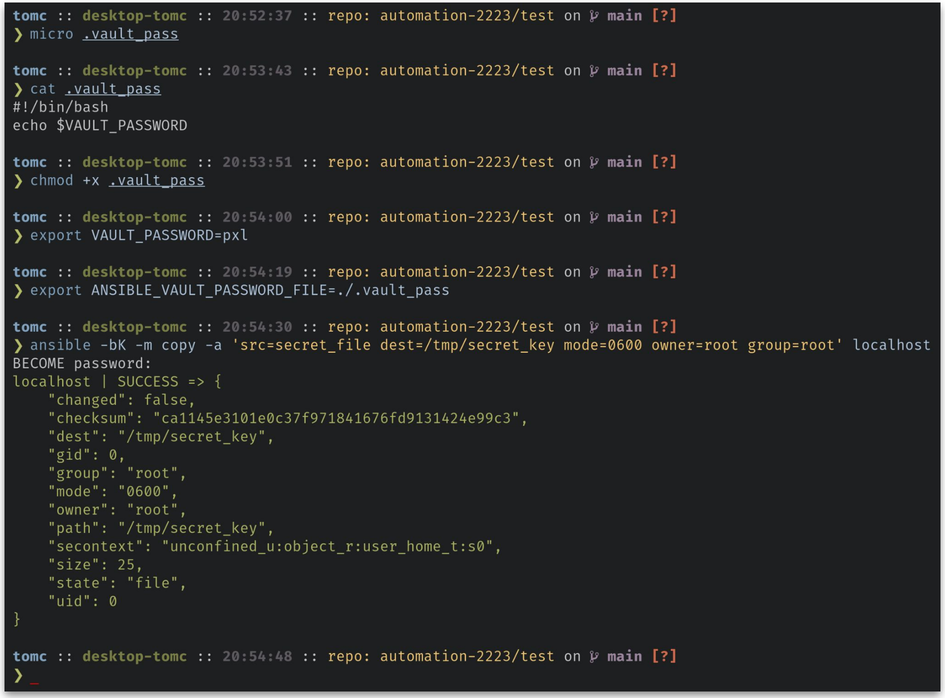
- Set env variable `ANSIBLE_VAULT_PASSWORD_FILE` with password file path to avoid using flags.
- Use `ansible.cfg` file to make Ansible aware of password file location in different sessions.
- In the `[defaults]` section, set `vault_password_file` to the location of your password file (can be relative or absolute path).
- Commands requiring decryption will no longer prompt for vault password.
- `ansible-vault` will use the password file not only for decrypting files, but also for creating new files with `ansible-vault create` and `ansible-vault encrypt`
- `ansible -bK -m copy -a 'src=secret_file dest=/tmp/secret_key mode=0600 owner=root group=root' localhost`



```
❯ export ANSIBLE_VAULT_PASSWORD_FILE=./.vault_pass

tomc :: desktop-tomc :: 20:46:24 :: repo: automation-2223/test on  main [?]
❯ ansible -bK -m copy -a 'src=secret_file dest=/tmp/secret_key mode=0600 owner=root group=root' localhost
BECOME password:
localhost | CHANGED => {
    "changed": true,
    "checksum": "ca1145e3101e0c37f971841676fd9131424e99c3",
    "dest": "/tmp/secret_key",
    "gid": 0,
    "group": "root",
    "md5sum": "8aa234a813df07ec0baa0ddcb6f70e6a",
    "mode": "0600",
    "owner": "root",
    "secontext": "unconfined_u:object_r:user_home_t:s0",
    "size": 25,
    "src": "/home/tomc/.ansible/tmp/ansible-tmp-1681238789.0253475-84932-211408227726363/source",
    "state": "file",
    "uid": 0
}

tomc :: desktop-tomc :: 20:46:29 :: repo: automation-2223/test on  main [?]
❯ _
```

Advanced trick: don't store any password on disk

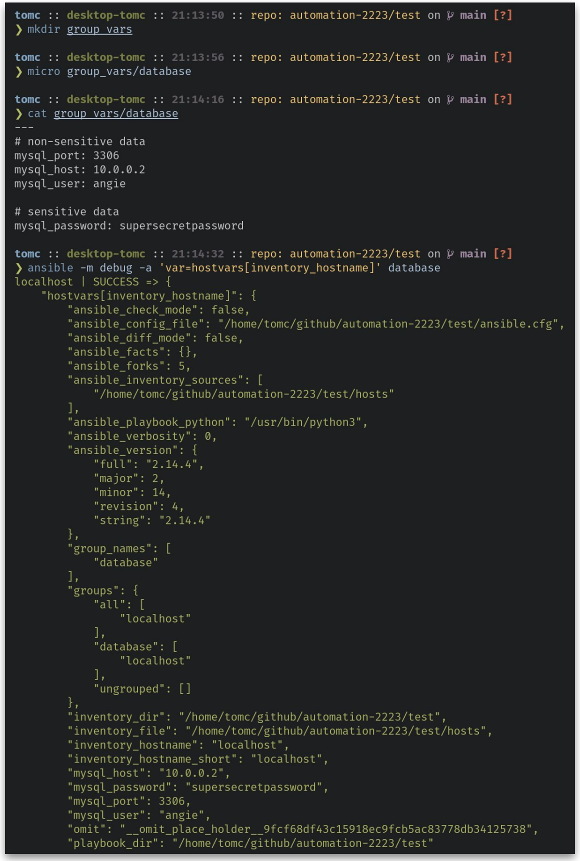
- Concerned about accidentally committing password file to repository? Ansible has env variable to indicate location of password file, but not to set password itself. :^(
- BUT, **if the password file is executable, Ansible will run it as a script** and use resulting output as password. So we can make it echo the value of an environment variable that we choose as password. :^)



```
tomc :: desktop-tomc :: 20:52:37 :: repo: automation-2223/test on  main [?]
❯ micro .vault_pass

tomc :: desktop-tomc :: 20:53:43 :: repo: automation-2223/test on  main [?]
❯ cat .vault_pass
#!/bin/bash
echo $VAULT_PASSWORD

tomc :: desktop-tomc :: 20:53:51 :: repo: automation-2223/test on  main [?]
❯ chmod +x .vault_pass

tomc :: desktop-tomc :: 20:54:00 :: repo: automation-2223/test on  main [?]
❯ export VAULT_PASSWORD=pxl

tomc :: desktop-tomc :: 20:54:19 :: repo: automation-2223/test on  main [?]
❯ export ANSIBLE_VAULT_PASSWORD_FILE=./.vault_pass

tomc :: desktop-tomc :: 20:54:30 :: repo: automation-2223/test on  main [?]
❯ ansible -bK -m copy -a 'src=secret_file dest=/tmp/secret_key mode=0600 owner=root group=root' localhost
BECOME password:
localhost | SUCCESS => {
    "changed": false,
    "checksum": "ca1145e3101e0c37f971841676fd9131424e99c3",
    "dest": "/tmp/secret_key",
    "gid": 0,
    "group": "root",
    "mode": "0600",
    "owner": "root",
    "path": "/tmp/secret_key",
    "secontext": "unconfined_u:object_r:user_home_t:s0",
    "size": 25,
    "state": "file",
    "uid": 0
}

tomc :: desktop-tomc :: 20:54:48 :: repo: automation-2223/test on  main [?]
❯ _
```

Using Vault-encrypted Variables: the problem

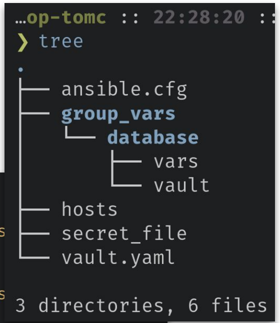
- Typically variables are added in a **`group_vars`** file, e.g. **`group_vars/database`**.
  - Some variables like MySQL port number are not secret and can be shared.
  - Others like database password are confidential.
  - Test variable availability using Ansible the **`debug`** module and **`hostvars`** variable.
  - Output confirms all variables are applied to the host
  - However, **`group_vars/database`** file contains all variables, including confidential ones
- Leaving the **`group_vars`** file unencrypted is a security issue, but encrypting all variables creates usability and collaboration problems.



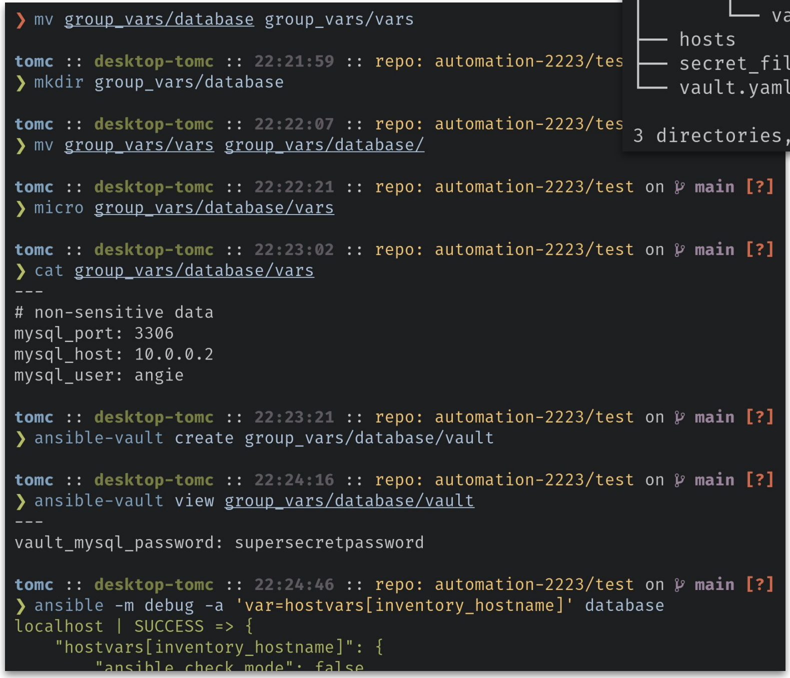
```
tomc :: desktop-tomc :: 21:13:50 :: repo: automation-2223/test on  main [?]
❯ mkdir group_vars

tomc :: desktop-tomc :: 21:13:56 :: repo: automation-2223/test on  main [?]
❯ micro group_vars/database

tomc :: desktop-tomc :: 21:14:16 :: repo: automation-2223/test on  main [?]
❯ cat group_vars/database
---
# non-sensitive data
mysql_port: 3306
mysql_host: 10.0.0.2
mysql_user: angie

# sensitive data
mysql_password: supersecretpassword

tomc :: desktop-tomc :: 21:14:32 :: repo: automation-2223/test on  main [?]
❯ ansible -m debug -a 'var=hostvars[inventory_hostname]' database
localhost | SUCCESS => {
    "hostvars[inventory_hostname]": {
        "ansible_check_mode": false,
        "ansible_config_file": "/home/tomc/github/automation-2223/test/ansible.cfg",
        "ansible_diff_mode": false,
        "ansible_facts": {},
        "ansible_forks": 5,
        "ansible_inventory_sources": [
            "/home/tomc/github/automation-2223/test/hosts"
        ],
        "ansible_playbook_python": "/usr/bin/python3",
        "ansible_verbosity": 0,
        "ansible_version": {
            "full": "2.14.4",
            "major": 2,
            "minor": 14,
            "revision": 4,
            "string": "2.14.4"
        },
        "group_names": [
            "database"
        ],
        "groups": {
            "all": [
                "localhost"
            ],
            "database": [
                "localhost"
            ],
            "ungrouped": []
        },
        "inventory_dir": "/home/tomc/github/automation-2223/test",
        "inventory_file": "/home/tomc/github/automation-2223/test/hosts",
        "inventory_hostname": "localhost",
        "inventory_hostname_short": "localhost",
        "mysql_host": "10.0.0.2",
        "mysql_password": "supersecretpassword",
        "mysql_port": 3306,
        "mysql_user": "angie",
        "omit": "__omit_place_holder__9fcf68df43c15918ec9fcb5ac83778db34125738",
        "playbook_dir": "/home/tomc/github/automation-2223/test"
```

Separating out sensitive variables: try 1

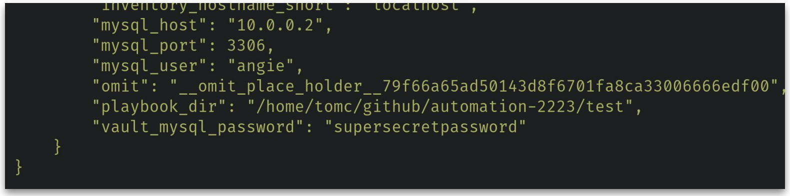
- Variables can be split across two files.
- Use a variables directory instead of a single Ansible variables file to apply variables from more than one file.
- We'll create 2 files: vars for the unencrypted variables and vault for the encrypted variables.
- Use the same variable names but prefix with "vault_" to indicate that these variables are defined in the vault-encrypted file.
- Variables are now separated and only the sensitive data is encrypted, which is secure but less usable.
- Problem: Not clear which variables are assigned without referencing more than one file, and while you may want to restrict access to sensitive data while collaborating, you probably still want to share the variable names.



```
…op-tomc :: 22:28:20 ::
❯ tree
.
├── ansible.cfg
├── group_vars
│   └── database
│       ├── vars
│       └── vault
├── hosts
├── secret_file
└── vault.yaml

3 directories, 6 files
```



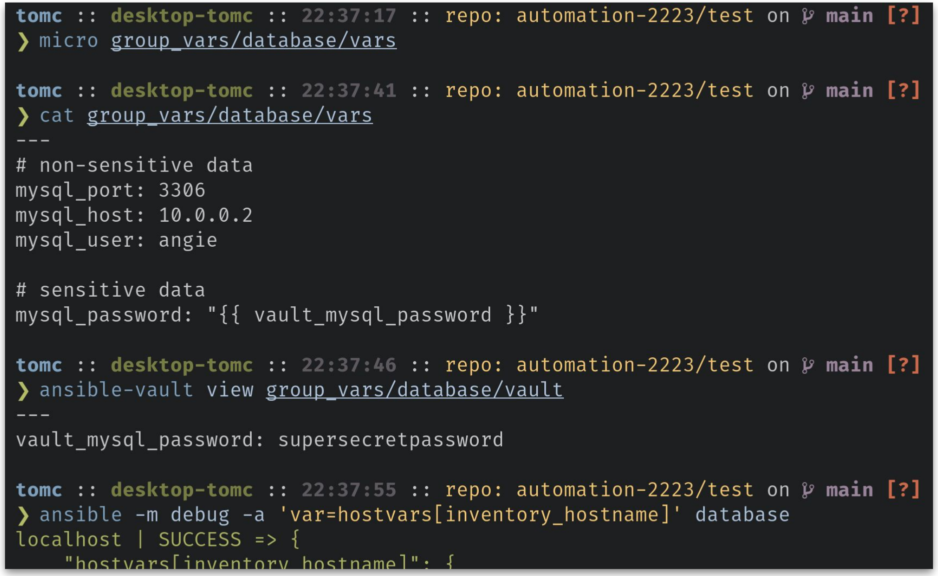
```
❯ mv group_vars/database group_vars/vars

tomc :: desktop-tomc :: 22:21:59 :: repo: automation-2223/tes
❯ mkdir group_vars/database

tomc :: desktop-tomc :: 22:22:07 :: repo: automation-2223/tes
❯ mv group_vars/vars group_vars/database/

tomc :: desktop-tomc :: 22:22:21 :: repo: automation-2223/test on ᚠ main [?]
❯ micro group_vars/database/vars

tomc :: desktop-tomc :: 22:23:02 :: repo: automation-2223/test on ᚠ main [?]
❯ cat group_vars/database/vars
---
# non-sensitive data
mysql_port: 3306
mysql_host: 10.0.0.2
mysql_user: angie

tomc :: desktop-tomc :: 22:23:21 :: repo: automation-2223/test on ᚠ main [?]
❯ ansible-vault create group_vars/database/vault

tomc :: desktop-tomc :: 22:24:16 :: repo: automation-2223/test on ᚠ main [?]
❯ ansible-vault view group_vars/database/vault
---
vault_mysql_password: supersecretpassword

tomc :: desktop-tomc :: 22:24:46 :: repo: automation-2223/test on ᚠ main [?]
❯ ansible -m debug -a 'var=hostvars[inventory_hostname]' database
localhost | SUCCESS => {
    "hostvars[inventory_hostname]": {
        "ansible_check_mode": false
```



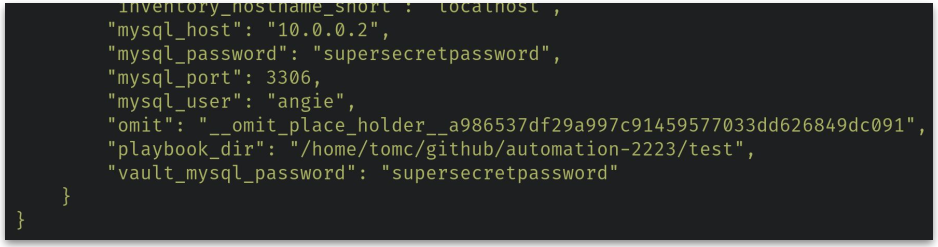
```
        "inventory_hostname_short": "localhost",
        "mysql_host": "10.0.0.2",
        "mysql_port": 3306,
        "mysql_user": "angie",
        "omit": "__omit_place_holder__79f66a65ad50143d8f6701fa8ca33006666edf00",
        "playbook_dir": "/home/tomc/github/automation-2223/test",
        "vault_mysql_password": "supersecretpassword"
    }
}
```

Separating out sensitive vault variables with Jinja2

- Original variable names (e.g. mysql_password) can be added back to unencrypted file using Jinja2 templating statements to reference encrypted variable names instead of the secret values.
- Allows all variables to be viewed in a single file, while confidential values remain in encrypted file.
- E.g. mysql_password variable set to "{{ vault_mysql_password defined }}" will get replaced by the decrypted value from the vault file.
- Method allows for understanding of all variables to be applied to hosts in database group by looking at group_vars/database/vars, while Jinja2 templates hide sensitive parts in the vault file.
- group_vars/database/vault file only needs to be opened if values need to be viewed or modified.



```
tomc :: desktop-tomc :: 22:37:17 :: repo: automation-2223/test on main [?]
❯ micro group_vars/database/vars

tomc :: desktop-tomc :: 22:37:41 :: repo: automation-2223/test on main [?]
❯ cat group_vars/database/vars
---
# non-sensitive data
mysql_port: 3306
mysql_host: 10.0.0.2
mysql_user: angie

# sensitive data
mysql_password: "{{ vault_mysql_password }}"

tomc :: desktop-tomc :: 22:37:46 :: repo: automation-2223/test on main [?]
❯ ansible-vault view group_vars/database/vault
---
vault_mysql_password: supersecretpassword

tomc :: desktop-tomc :: 22:37:55 :: repo: automation-2223/test on main [?]
❯ ansible -m debug -a 'var=hostvars[inventory_hostname]' database
localhost | SUCCESS => {
    "hostvars[inventory_hostname]": {
```



```
        "inventory_hostname_short": "localhost",
        "mysql_host": "10.0.0.2",
        "mysql_password": "supersecretpassword",
        "mysql_port": 3306,
        "mysql_user": "angie",
        "omit": "__omit_place_holder__a986537df29a997c91459577033dd626849dc091",
        "playbook_dir": "/home/tomc/github/automation-2223/test",
        "vault_mysql_password": "supersecretpassword"
    }
}
```

end