Ansible Best Practices

techbeatly.com/ansible-best-practices

youtube.com/techbeatly

techbeatly

techbeatly.com/ansible-best-practices

Ansible is simple, make it simple

- Use only the features you needed in your playbook
- Use simple methods to achieve your goal
- Write playbooks as “Human Readable”
- Use available modules rather than raw commands

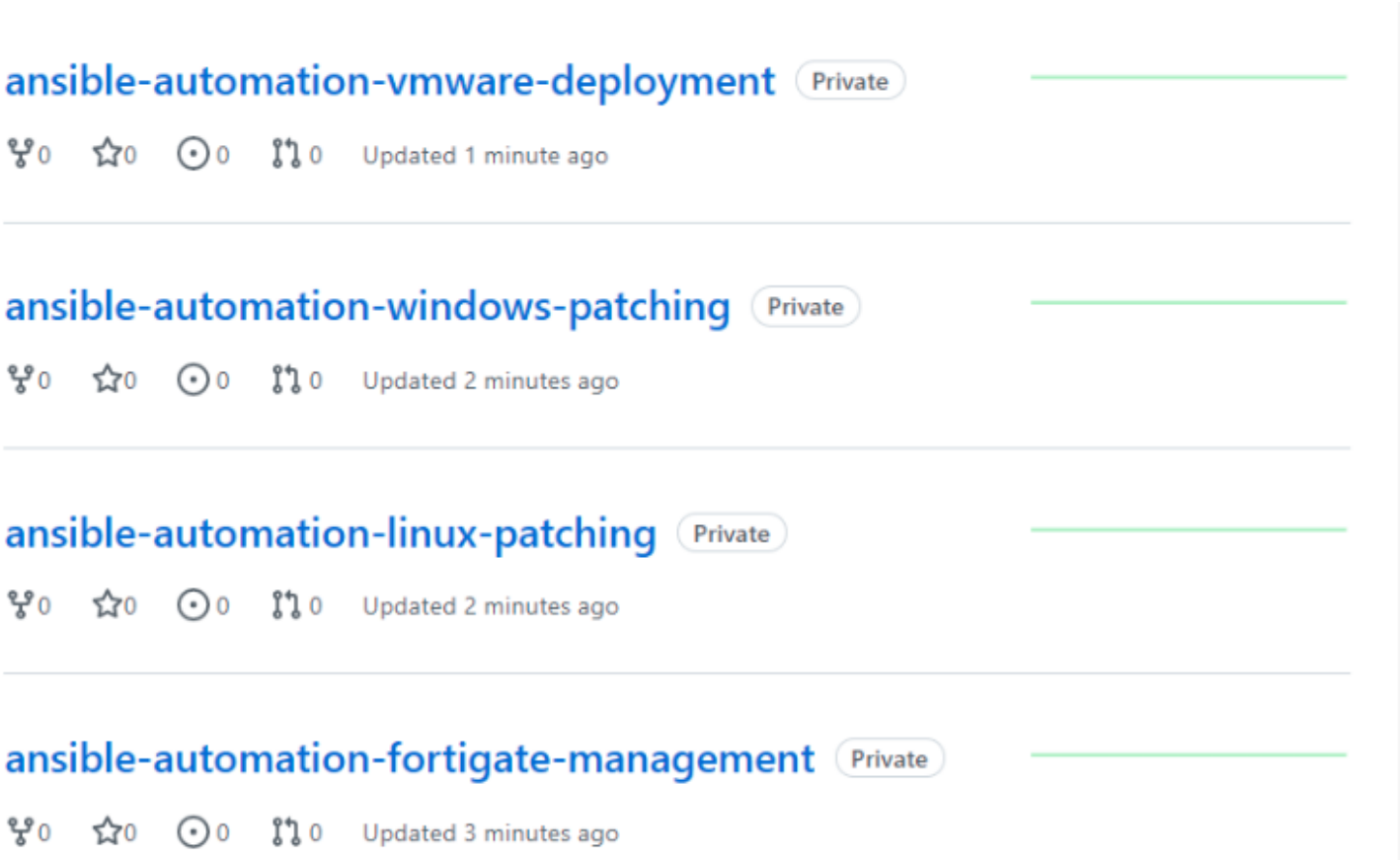
youtube.com/techbeatly

techbeatly

techbeatly.com/ansible-best-practices

Keep Projects in Version Control System

- Playbooks, Configurations, Variables, Roles and Collections
- Opportunity for Collaboration
- Less worry about the old version of playbooks and configurations
- Make Auditing possible
- Create project-specific repositories



youtube.com/techbeatly

techbeatly

Make Playbooks Reader Friendly

- Use comments inside playbooks; useful for everyone
- Keep a style guide
- Use whitespaces and extra lines as needed
- Practice names for tasks
- Use proper tags for tasks
- Main playbooks calling roles or sub-playbooks
- Use explicit declarations (eg: state or overwrite actions)
- Use handlers in playbooks and roles
- Avoid shell and command modules as much as possible

techbeatly.com/ansible-best-practices

Keep a style guide

```
You, seconds ago | 2 authors (ginigangadharan and others)
- name: Enable Intranet Services
  hosts: node1.techbeatly.com
  become: yes
  tasks:
    - name: Install httpd and firewalld Packages
      yum:
        name:
          - httpd
          - firewalld
        state: latest

    - name: Enable and Start Firewalld Service
      service:
        name: firewalld
        enabled: true
        state: started

    - name: firewalld permit httpd service
      firewalld:
        service: http
        permanent: true
        state: enabled
        immediate: yes
```

techbeatly.com/ansible-best-practices

Native YAML for Playbooks

```yaml
tasks:
  - name: Copy a file to managed hosts
    copy: name=demo.txt dest=/tmp/demo.txt owner=ansible group=ansible
  - name: Create a new directory if it does not exist
    file: path=/home/ansible/new-dir state=directory mode='0755'
```

```yaml
tasks:
  - name: Copy a file to managed hosts
    copy:
      src: files/demo-text-file.txt
      dest: /home/ansible/demo-text-file.txt
      owner: ansible
      group: ansible

  - name: Create a new directory if it does not exist
    file:
      path: /home/ansible/new-dir
      state: directory
      mode: '0755'
```

youtube.com/techbeatly

techbeatly

techbeatly.com/ansible-best-practices

Avoid hardcoding

```yaml
- name: Installing Web Packages
  hosts: webservers
  tasks:
    - name: Installing Web
      yum:
        name: httpd
        state: present
```

```yaml
- name: Installing Web Packages
  hosts: "{{ nodes }}"
  tasks:
    - name: Installing Web
      yum:
        name: “{{ web_package }}”
        state: present
```

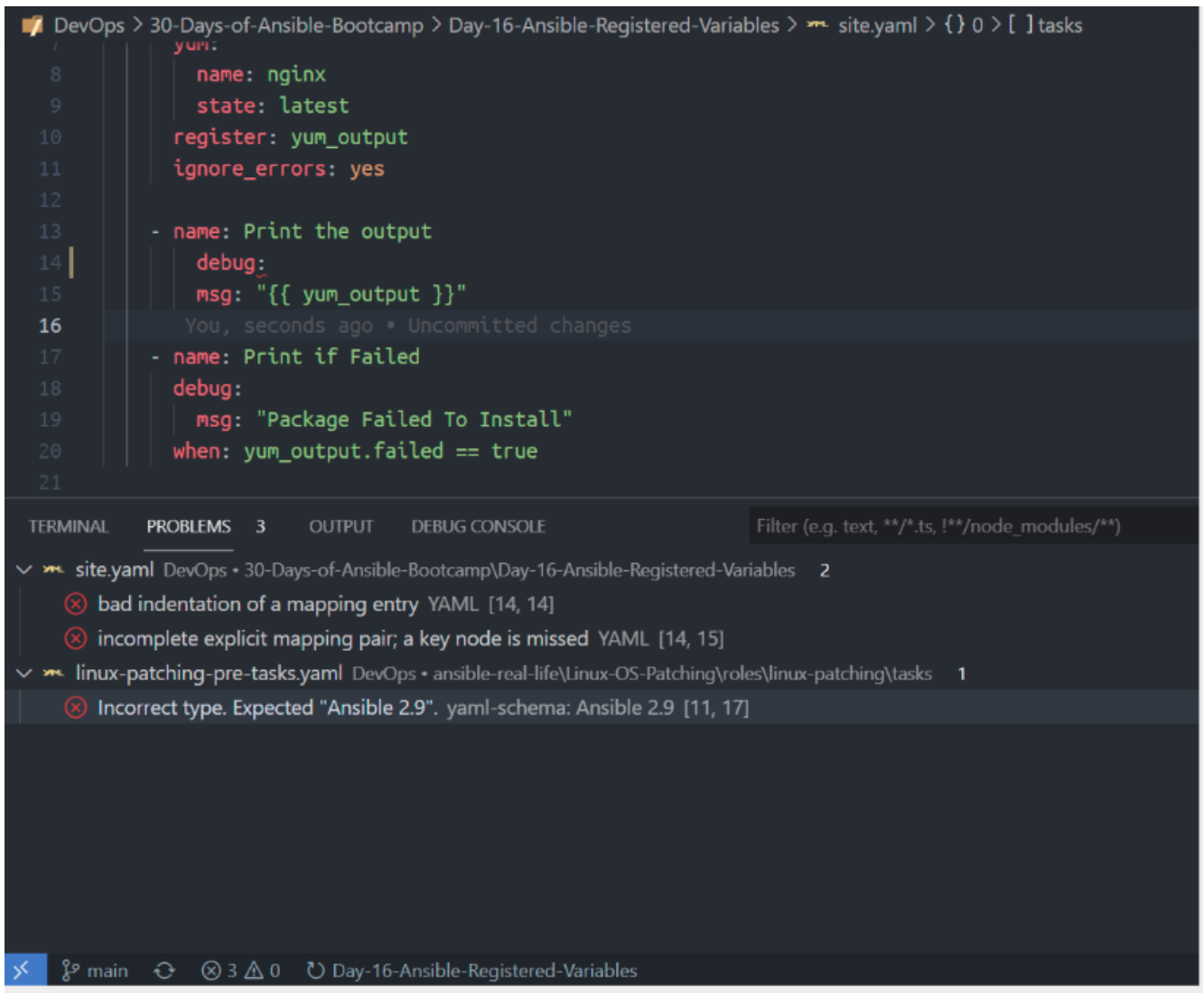
```
$ ansible-playbook site.yaml --extra-vars "nodes=webservers web_package=httpd”
```

youtube.com/techbeatly

techbeatly

Use editor with syntax highlighting

- VSCode
- Atom
- Sublime
- Vim with Plugins



techbeatly.com/ansible-best-practices

Use block

```yaml
tasks:
  - block:
      - name: Show Message
        debug:
          msg: "Trying httpd"
      - name: Install Package
        yum:
          name: httpd-wrong
          state: present

    rescue:
      - name: Show error
        debug:
          msg: "Unknown Package"
      - name: Install nginx
        yum:
          name: nginx
          state: latest

    always:
      - name: Message
        debug:
          msg: "Playbook Done"
```

techbeatly.com/ansible-best-practices

Use Roles and subtasks

- Break tasks into small and simple playbooks or roles for better management

```yaml
---
- name: Install Server
  hosts: node1
  become: yes
  roles:
    - role: geerlingguy.git
    - role: mynextrole
```

```yaml
- name: "Patching Pre-tasks"
  include_role:
    name: linux-patching
    tasks_from: linux-patching-pre-tasks.yaml

- name: "Patching Tasks"
  include_role:
    name: linux-patching

- name: "Patching Post-tasks"
  include_role:
    name: linux-patching
    tasks_from: linux-patching-post-tasks.yaml
```

youtube.com/techbeatly

techbeatly

techbeatly.com/ansible-best-practices

Use template for complex configurations

```
    - name: Add a block of text to an existing file
      blockinfile:
        path: /home/ansible/demo-text-file.txt
        block: |
            Welcome to the server.
Access is restricted; if you are not authorized to use it
please logout from this system
        state: present
```

```
    - name: Deploy motd
      template:
        dest: /etc/motd
        src: motd.j2
```

```
Welcome to {{ ansible_facts.hostname }}
(IP Address: {{ ansible_facts.default_ipv4.address }})

Access is restricted; if you are not authorized to use it
please logout from this system

If you have any issues, please contact {{ system_admin_email }}.
Phone: {{ system_admin_phone | default('1800 1111 2222') }}

----------------------------------
This message is configured by Ansible
----------------------------------
```

youtube.com/techbeatly

techbeatly

techbeatly.com/ansible-best-practices

Organize Files and Directories

```
inventories/
   production/
      hosts                # inventory file for production servers
   staging/
      hosts                # inventory file for staging environment
library/                  # custom modules or plugins
module_utils/
filter_plugins/

site.yml                  # main playbook
webservers.yml            # sub playbook
dbservers.yml

roles/                    # roles directory
    webapp/
    dbinstall/
    monitoring
    backup/
```

Keep Inventories Organized

- Group hosts based on functionality (web, database, app etc)
- Make use of Dynamic Inventory wherever possible (Cloud, Containers)
- Keep sensitive information in separate host_vars/group_vars

```
[webservers]
servera
serverb
serverc

[database]
db1
db2

[somanyservers]
db[a:f].example.com

[manyips]
192.168.0.[10:20]
```

techbeatly.com/ansible-best-practices

production, staging and dev Inventories

- production, staging and dev Inventories

```
inventories/
  production/
    hosts                  # inventory file for production servers
    group_vars/
      group1.yml         # variables to particular groups
    host_vars/
      hostname1.yml      # variables to particular systems
  staging/
    hosts                    # inventory file for staging environment
    group_vars/
      group1.yml         # variables to particular groups
    host_vars/
      stagehost1.yml     # variables to particular systems
```

```
$ ansible-playbook -i production site.yml
```

techbeatly.com/ansible-best-practices

Human Readable Hostnames

- Use ansible_host option with readable names for hosts

```
192.168.1.61
188.11.12.33
100.24.45.2

webserver101.example.com
dbprod.sg.example.com
db1982.sg.example.com
```

```
server101 ansible_host=192.168.1.61
server102 ansible_host=188.11.12.33
server103 ansible_host=100.24.45.2

webserver101 ansible_host=webserver101.example.com
dbprod ansible_host=dbprod.sg.example.com
db1982 ansible_host=db1982.sg.example.com
```

youtube.com/techbeatly

techbeatly

techbeatly.com/ansible-best-practices

Trusted access to remote hosts

- Use proper user credentials with best security
- Create dedicated account for ansible if possible (with enough privilege)
- Accessing remote host using root or administrator account is not a good idea

```
[defaults]
remote_user: devops
```

```
---
- name: Installing Web
  hosts: webservers
  remote_user: devops
```

```
[webservers]
Web101  ansible_connection=ssh ansible_user=devops
```

techbeatly.com/ansible-best-practices

Meaningful names for variables

- Use appropriate name for your variables
- Make sure no variable duplicates or unwanted overwriting
- Keep your variables at appropriate locations

```
myvar: something
webport: 8080
dbpath: /opt/mysql
fwpackage: firewalld
fg_api: 10.1.10.10
```

```
user_location: /home/devops/
httpd_web_port: 8080
mysql_database_home: /opt/mysql
firewall_package: firewalld
fortigate_api_ip: 10.1.10.10
```

techbeatly.com/ansible-best-practices

production, staging and dev variables

- Separate production, staging and development variables

```
vars/
  production/
    web_vars.yml        # web server variables
    db_vars.yml             # db server variables
  staging/
    web_vars.yml        # web server variables
    db_vars.yml             # db server variables
```

```
vars:
  server_env: production
tasks:
  - name: Show users
    include_vars:
      file: "vars/{{ server_env }}/web_vars.yml"
```

```
$ ansible-playbook site.yml -e "server_env=production"
```

Optimize Playbooks Execution

- Use parallelism
- Use appropriate strategy as needed
- Use appropriate value for forks
- Use serial to execute in batches
- Use order to control execution based on inventory
- Use throttle for high CPU intensive tasks

```
[defaults]
forks=100


$ ansible-playbooks site.yml -f 10
```

```
- name: Installing Web
  hosts: web
  strategy: free
  forks: 20
  serial: 2
  forks: 20
  order: sorted
  throttle: 1
```

```
  serial:
    - 1
    - 10%
    - 100%
```

Use debugging and troubleshooting

- Do syntax check before running long playbooks --syntax-check
- Use debug levels -vvv
- Use step by step execution to see the progress --step
- Start with specific tasks --start-at-task
- Use --check and --diff for dry run mode
- Use ad hoc commands to test quick items
- Use debug module without hesitation

```yaml
tasks:
  - name: Show users
    debug:
      msg: "{{ item.value }}"
    with_items: "{{ users }}"
```

Bundle Dependencies

- Include custom modules in ./library
- Keep playbook specific roles in ./roles
- Keep playbook specific collections in ./collections

Use trusted content for roles and collections

- Make sure you get support
- DO NOT blindly use open contents for your environment; scan it and test it before you using
- Find well known and trusted sources

Follow Your Process

- Always test your updated playbook or configurations in dev/staging environment
- Implement approval stages using existing tools
- Eg: Call ServiceNow/Jira tickets and use approvals or reviews

Ansible FREE Course: techbeatly.com/ansible-course

Ansible Real Life: techbeatly.com/ansible-real-life

/techbeatly t.me/techbeatly

youtube.com/techbeatly

techbeatly