How to speed up Ansible tasks

techbeatly.com/ansible-best-practices

youtube.com/techbeatly

techbeatly

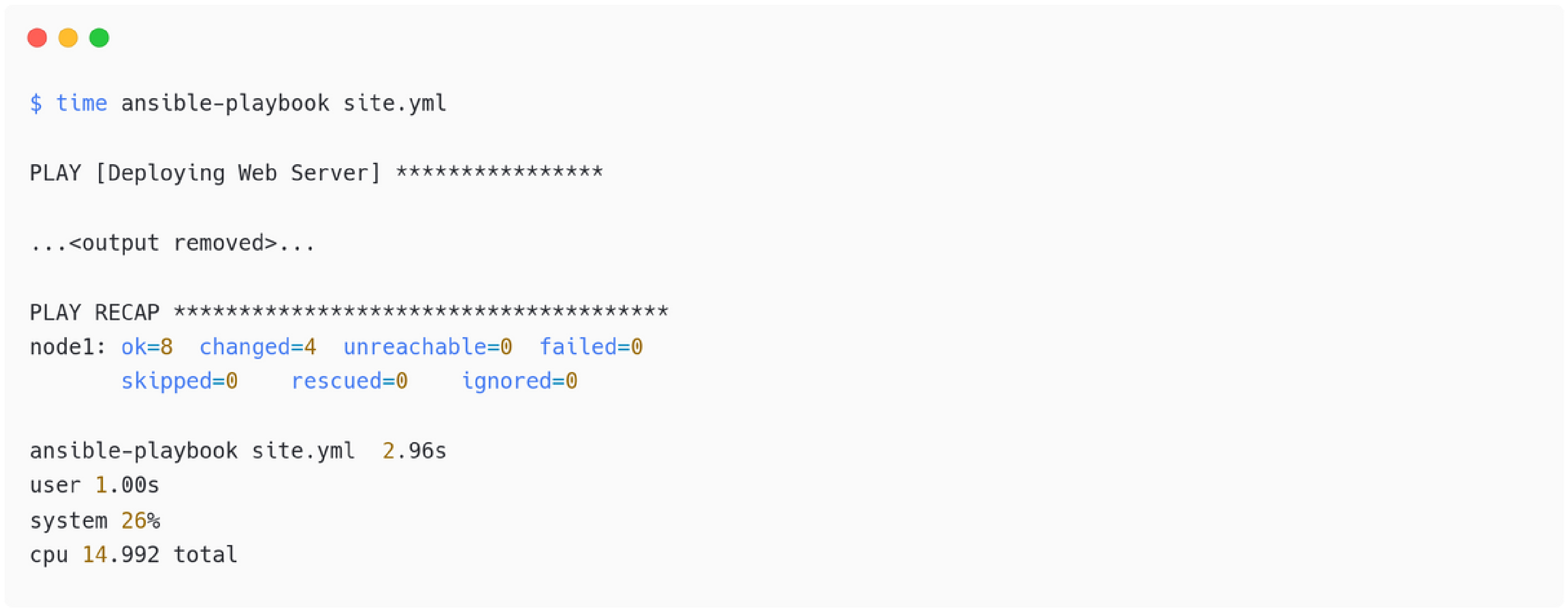
Identify slow tasks with callback plugins

Eenable **callback plugins** such as timer, profile_tasks, and profile_roles to find a task's time consumption and identify which jobs are slowing down your plays.

```
[defaults]
inventory = ./hosts
callbacks_enabled = timer, profile_tasks, profile_roles
```

Disable fact gathering

If you're not using the facts, disable gathering



```
$ time ansible-playbook site.yml

PLAY [Deploying Web Server] ****************

...<output removed>...

PLAY RECAP ***************************************
node1: ok=8  changed=4  unreachable=0  failed=0
       skipped=0    rescued=0    ignored=0

ansible-playbook site.yml  2.96s
user 1.00s
system 26%
cpu 14.992 total
```

Configure parallelism

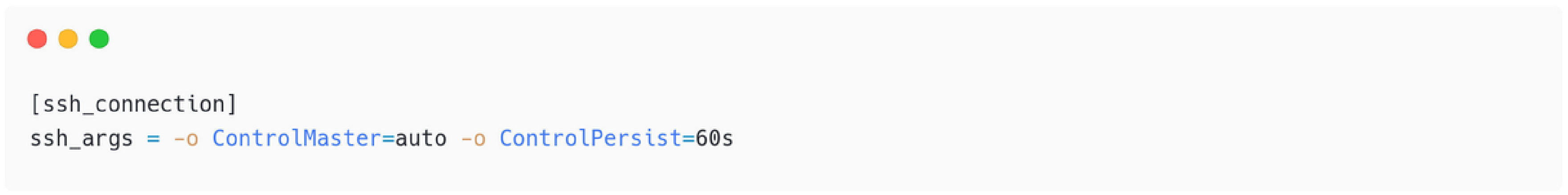
The default value for forks is 5, which means Ansible executes a task on the first five hosts, waits for the task to complete, and then takes the next batch of five hosts, and so on.

```
[defaults]
inventory = ./hosts
forks=50


$ ansible-playbook site.yaml --forks 50
```

Configure SSH optimization

Use ControlMaster and ControlPersist features in ansible.cfg (in the ssh_connection section)



```
[ssh_connection]
ssh_args = -o ControlMaster=auto -o ControlPersist=60s
```

Disable host key checking in a dynamic environment

If your environment contains immutable managed nodes (virtual machines or containers), then the key is different when the host is reinstalled or recreated.

```
[defaults]
host_key_checking = False
```

Use pipelining

Reduce the number of SSH connections by enabling the pipelining parameter

```
# ansible.cfg
pipelining = True
```

Use execution strategies

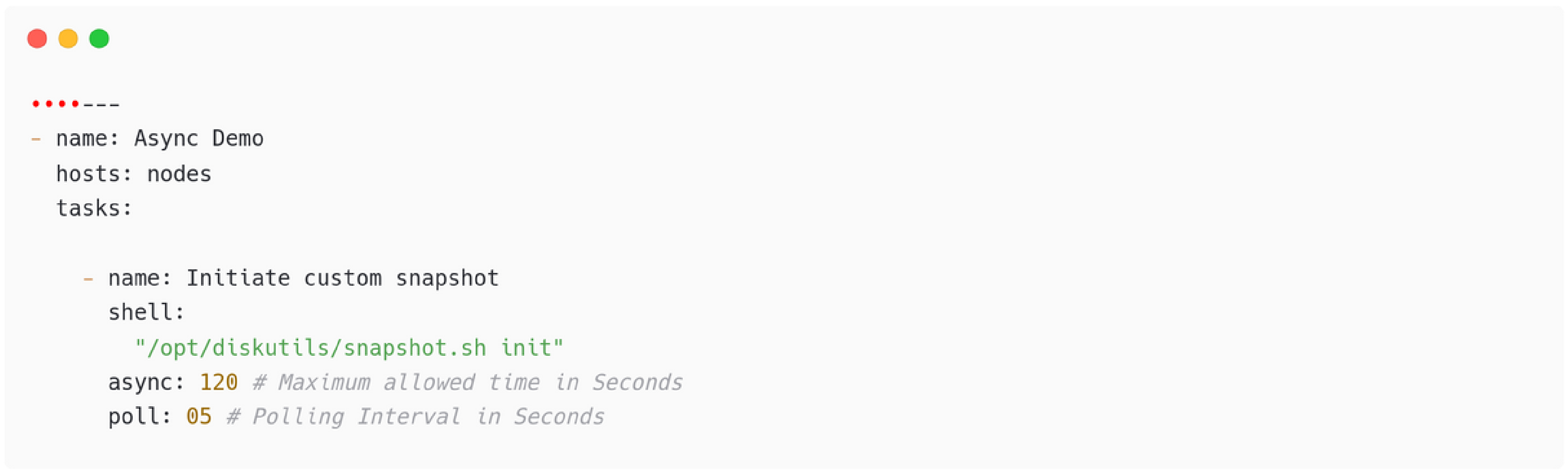
If you don't have dependencies on tasks or managed nodes, you can change strategy to free, which allows Ansible to execute tasks on managed hosts until the end of the play without waiting for other hosts to finish their tasks:

```
- hosts: production servers
  strategy: free
  tasks:
```

techbeatly.com/ansible-course

Use async tasks

If the following tasks do not depend on this long-running task, you can use the async mode with an appropriate poll interval to tell Ansible not to wait and proceed with the next tasks:

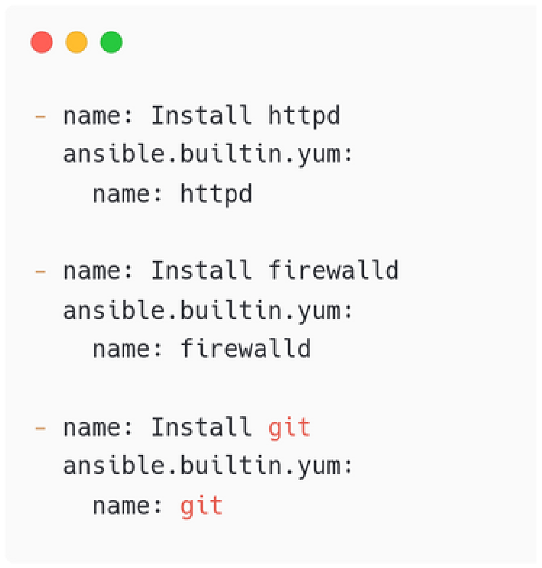


```
••••---
- name: Async Demo
  hosts: nodes
  tasks:

    - name: Initiate custom snapshot
      shell:
        "/opt/diskutils/snapshot.sh init"
      async: 120 # Maximum allowed time in Seconds
      poll: 05 # Polling Interval in Seconds
```

Use multiple tasks in a single module and avoid module loops

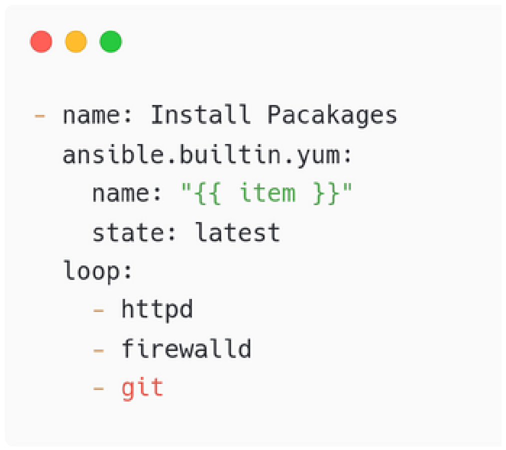
Instead of installing packages using multiple yum or dnf modules, you can pass multiple packages to a single yum task.



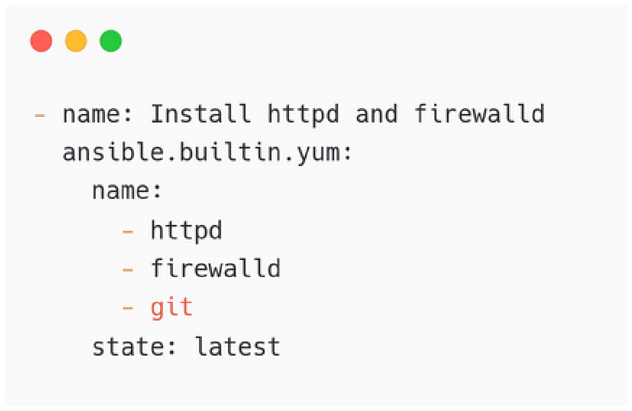
```yaml
- name: Install httpd
  ansible.builtin.yum:
    name: httpd

- name: Install firewalld
  ansible.builtin.yum:
    name: firewalld

- name: Install git
  ansible.builtin.yum:
    name: git
```



```yaml
- name: Install Pacakages
  ansible.builtin.yum:
    name: "{{ item }}"
    state: latest
  loop:
    - httpd
    - firewalld
    - git
```



```yaml
- name: Install httpd and firewalld
  ansible.builtin.yum:
    name:
      - httpd
      - firewalld
      - git
    state: latest
```

Avoid copy loops and use the synchronize module

Use synchronize modules rather than modules or loops:

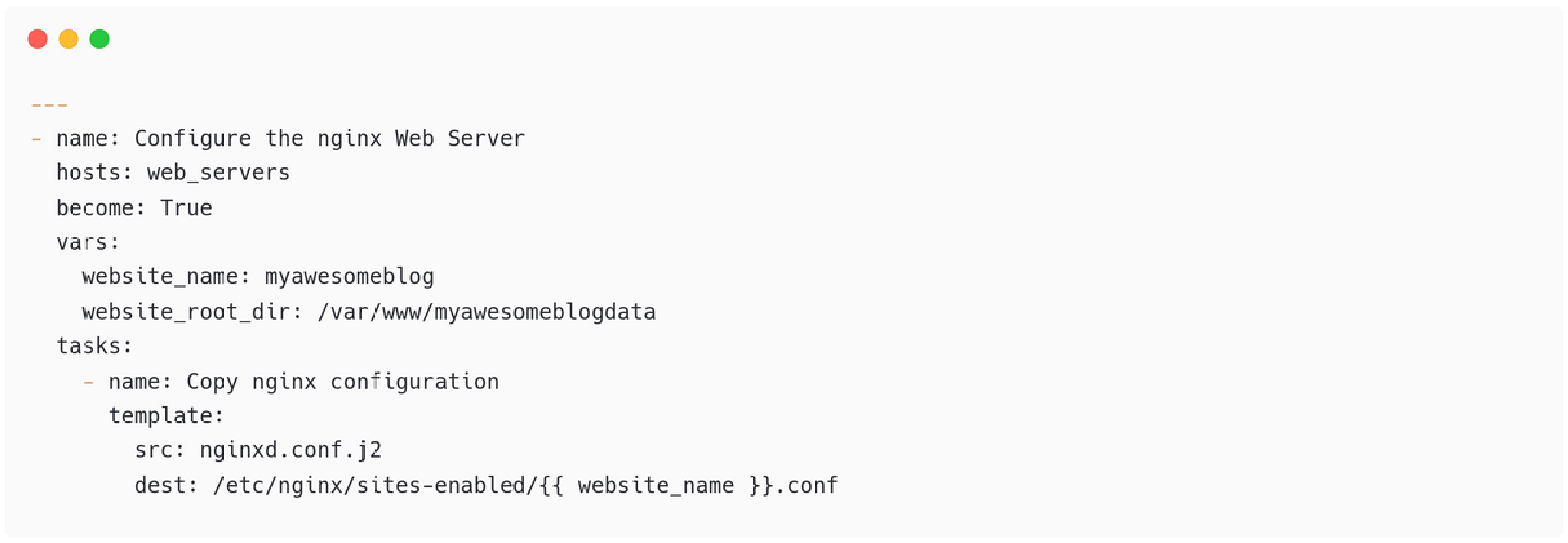
```
- name: Copy application data
  synchronize:
    src: app_data/
    dest: /opt/web_app/data
```

Use the latest version of Ansible and its modules

Use the latest compatible version of Ansible for your environments to ensure you're getting the most recent features.

Make configuration templates

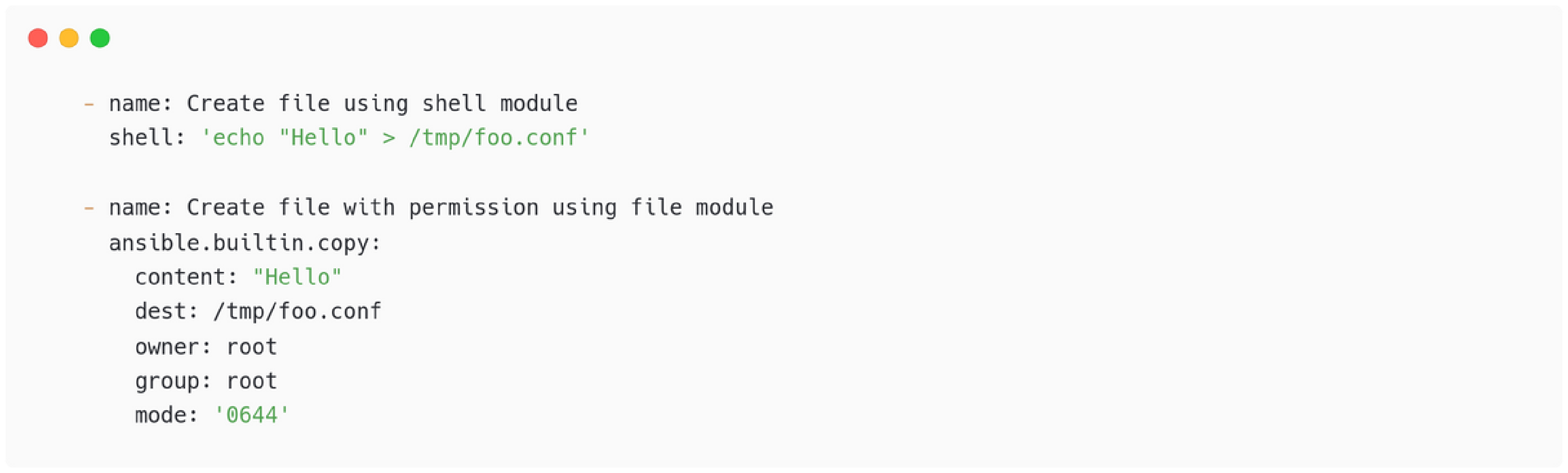
use a Jinja2 template to create any level of complex files and use the template module (or filter) to configure managed nodes.



```yaml
---
- name: Configure the nginx Web Server
  hosts: web_servers
  become: True
  vars:
    website_name: myawesomeblog
    website_root_dir: /var/www/myawesomeblogdata
  tasks:
    - name: Copy nginx configuration
      template:
        src: nginxd.conf.j2
        dest: /etc/nginx/sites-enabled/{{ website_name }}.conf
```

Use appropriate modules and avoid using shell or command modules

Use appropriate modules and use these shell/command modules in the worst-case scenarios



```yaml
- name: Create file using shell module
  shell: 'echo "Hello" > /tmp/foo.conf'

- name: Create file with permission using file module
  ansible.builtin.copy:
    content: "Hello"
    dest: /tmp/foo.conf
    owner: root
    group: root
    mode: '0644'
```

8 ways to speed up your Ansible playbooks
redhat.com/sysadmin/faster-ansible-playbook-execution

5 ways to make your Ansible modules work faster
redhat.com/sysadmin/faster-ansible-modules

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

/techbeatly **t.me/techbeatly**