5) Repositório **GitHub** `awx-project` (versionando playbooks, roles e inventário)

> Todos os passos deste item podem ser feitos **em qualquer máquina** (seu notebook, jump host, etc.). O AWX consumirá apenas a **URL Git**.

5.1 Criar o repositório no GitHub

Opção A — via site

1. GitHub → **New repository**
2. **Repository name:** `awx-project`
3. Visibility: **Private** (recomendado) ou **Public**
4. Crie vazio (sem README), vamos subir via CLI.

Opção B — via GitHub CLI

```
# se tiver o gh instalado e autenticado
gh repo create <seu-usuario>/awx-project --private --confirm
```

5.2 (Opcional) Preparar acesso por **SSH**

```
# crie a chave se ainda não tiver
ssh-keygen -t ed25519 -C "<seu-email>" -f ~/.ssh/id_ed25519
# copie a pública e adicione no GitHub (Settings → SSH and GPG keys)
cat ~/.ssh/id_ed25519.pub
```

5.3 Montar a estrutura local do projeto

> Usa seu inventário (hypervisors, VMs, Windows, k8s), role `os_patch_dnf` e playbooks DNF.

```
setti
cd ~/awx-project
```

`ansible.cfg`

```
cat > ansible.cfg <<'EOF'
[defaults]
inventory = inventory.ini
stdout_callback = yaml
nocows = True
```

```
retry_files_enabled = False
host_key_checking = False
timeout = 60
forks = 20
EOF
```

`inventory.ini` (com seus hosts)

```
cat > inventory.ini <<'EOF'
# ======== ALL HOSTS ========
[hypervisors]
Lenovo-i7        ansible_host=192.168.31.31
hp-i7            ansible_host=192.168.31.36
Lenovoi5         ansible_host=192.168.31.37

[windows]
hp-i7-desktop    ansible_host=192.168.31.33

[vms]
vm-lenovoi7-openshifit-crc ansible_host=192.168.31.34
vm-lenovoi7-zabbix         ansible_host=192.168.31.35
k8s-cp1                    ansible_host=192.168.31.40
k8s-w1                     ansible_host=192.168.31.41
k8s-w2                     ansible_host=192.168.31.42

# ======== K8S CLUSTER ========
[k8s_controller]
k8s-cp1

[k8s_workers]
k8s-w1
k8s-w2

[k8s_all:children]
k8s_controller
k8s_workers

# ======== LINUX DNF (alvo dos playbooks de patch) ========
[linux_dnf]
vm-lenovoi7-openshifit-crc
vm-lenovoi7-zabbix
# Para incluir hypervisors Linux, descomente conscientemente:
# Lenovo-i7
# hp-i7
# Lenovoi5
# Nós K8s: use playbooks com drain/uncordon (não patch full direto!)
# k8s-cp1
# k8s-w1
# k8s-w2

[linux_dnf:vars]
```

```
ansible_user=admin
ansible_become=true

[all:vars]
ansible_ssh_common_args='-o StrictHostKeyChecking=no'
EOF
```

`collections/requirements.yml`

```
mkdir -p collections
cat > collections/requirements.yml <<'EOF'
---
collections:
  - name: ansible.posix
  - name: community.general
EOF
```

Role `roles/os_patch_dnf/defaults/main.yml`

```
cat > roles/os_patch_dnf/defaults/main.yml <<'EOF'
---
os_patch_security_only: false
os_patch_kernel_only:   false
os_patch_reboot_if_needed: true
os_patch_reboot_timeout: 1200
os_patch_serial: "25%"
os_patch_exclude: []
os_patch_autoremove: true
os_patch_clean: true
EOF
```

Role `roles/os_patch_dnf/tasks/main.yml`

```
cat > roles/os_patch_dnf/tasks/main.yml <<'EOF'
---
- name: Fail early se não for gerenciador dnf
  ansible.builtin.assert:
    that: ansible_pkg_mgr == "dnf"
    fail_msg: "Este role só deve rodar em distros com DNF
(RHEL/Rocky/Alma/Fedora)."
  tags: always

- name: Garantir dnf-utils (fornece 'needs-restarting')
  ansible.builtin.package:
    name: dnf-utils
    state: present
  register: pkg_utils
```

```
    failed_when: false
    changed_when: pkg_utils is changed
    tags: pre

  - name: Atualizar metadados
    ansible.builtin.dnf:
      update_cache: true
    tags: pre

  - name: Escolher modo de atualização (kernel-only / security-only / full)
    ansible.builtin.include_tasks: update.yml
    tags: update

  - name: Verificar necessidade de reboot (needs-restarting -r)
    ansible.builtin.command: needs-restarting -r
    register: needs_reboot
    changed_when: false
    failed_when: needs_reboot.rc not in [0,1]
    tags: reboot_check

  - name: Reboot se necessário e permitido
    ansible.builtin.reboot:
      reboot_timeout: "{{ os_patch_reboot_timeout }}"
      msg: "Reboot automático solicitado pelo play de patch"
    when:
      - os_patch_reboot_if_needed
      - needs_reboot.rc == 1
    tags: reboot

  - name: Autoremove de pacotes órfãos
    ansible.builtin.dnf:
      autoremove: true
    when: os_patch_autoremove
    tags: cleanup

  - name: Limpeza de cache DNF
    ansible.builtin.command: dnf clean all
    changed_when: "'0 files removed' not in (clean_out.stdout | default(''))"
    register: clean_out
    when: os_patch_clean
    tags: cleanup
  EOF
```

Role `roles/os_patch_dnf/tasks/update.yml`

```
cat > roles/os_patch_dnf/tasks/update.yml <<'EOF'
---
- name: Kernel only (apenas kernel e kernel-tools*)
  ansible.builtin.dnf:
    name:
      - kernel
```

```
        - kernel-core
        - kernel-modules
        - kernel-modules-extra
        - kernel-tools
        - kernel-tools-libs
      state: latest
      update_only: true
    when: os_patch_kernel_only

  - name: Security-only (todas as atualizações de segurança)
    ansible.builtin.dnf:
      name: "*"
      state: latest
      update_only: true
      security: true
      exclude: "{{ os_patch_exclude | join(',') if (os_patch_exclude|length>0)
  else omit }}"
    when:
      - not os_patch_kernel_only
      - os_patch_security_only

  - name: Full update (todas as atualizações disponíveis)
    ansible.builtin.dnf:
      name: "*"
      state: latest
      update_only: true
      exclude: "{{ os_patch_exclude | join(',') if (os_patch_exclude|length>0)
  else omit }}"
    when:
      - not os_patch_kernel_only
      - not os_patch_security_only
  EOF
```

Playbooks

```
mkdir -p playbooks

cat > playbooks/os_update_full.yml <<'EOF'
---
- name: Patching completo (DNF) com reboot por lote
  hosts: linux_dnf
  gather_facts: true
  strategy: linear
  serial: "{{ os_patch_serial | default('25%') }}"
  vars:
    os_patch_security_only: false
    os_patch_kernel_only:   false
    os_patch_exclude: []
  roles:
    - role: os_patch_dnf
EOF
```

```
cat > playbooks/os_update_security.yml <<'EOF'
---
- name: Patching de segurança (DNF)
  hosts: linux_dnf
  gather_facts: true
  serial: "{{ os_patch_serial | default('25%') }}"
  vars:
    os_patch_security_only: true
    os_patch_exclude: ["kernel*"]
  roles:
    - role: os_patch_dnf
EOF

cat > playbooks/os_update_kernel.yml <<'EOF'
---
- name: Atualização de kernel (DNF)
  hosts: linux_dnf
  gather_facts: true
  serial: "{{ os_patch_serial | default('25%') }}"
  vars:
    os_patch_kernel_only: true
    os_patch_reboot_if_needed: true
  roles:
    - role: os_patch_dnf
EOF
```

`group_vars/linux_dnf.yml`

```
mkdir -p group_vars
cat > group_vars/linux_dnf.yml <<'EOF'
os_patch_serial: "33%"
os_patch_reboot_timeout: 1800
os_patch_exclude: []
EOF
```

`.gitignore` e `README.md`

```
cat > .gitignore <<'EOF'
*.retry
__pycache__/
*.pyc
.venv/
collections/ansible_collections/
EOF

cat > README.md <<'EOF'
# awx-project
Playbooks de atualização DNF para uso no AWX.
```

```
- Role: roles/os_patch_dnf
- Playbooks: playbooks/os_update_full.yml, os_update_security.yml,
os_update_kernel.yml
- Inventário: inventory.ini (ajuste grupos conforme seu lab)
EOF
```

5.4 Versionar e subir para o GitHub

```
git init
git branch -M main
git add .
git commit -m "Base: role os_patch_dnf + playbooks DNF + inventário do lab"
# SSH (recomendado)
git remote add origin git@github.com:<seu-usuario>/awx-project.git
git push -u origin main
```

---

6) Conectar o repositório no **AWX**

6.1 Credential “Source Control” (SSH)

AWX → **Resources** → **Credentials** → **Add**

- **Type:** *Source Control*
- **Name:** `scm-github-ssh`
- **SSH Private Key:** cole o conteúdo de `~/.ssh/id_ed25519`
- Salvar.

(Se usar HTTPS, crie um token no GitHub e use em “Username/Password”.)

6.2 Project (SCM Git)

AWX → **Resources** → **Projects** → **Add**

- **Name:** `proj-awx-project`
- **SCM Type:** Git
- **SCM URL:** `git@github.com:<seu-usuario>/awx-project.git`
- **SCM Branch:** `main`
- **SCM Credential:** `scm-github-ssh`
- **Execution Environment:** seu EE padrão (com `ansible-core` e coleções necessárias)
- **Options:** marcar **Update on launch**, **Clean** (opcional: **Delete on update**)
- **Save** e depois **Sync**. Verifique o log do SCM Update.

6.3 Inventory (duas opções)

Opção A — **Inventory Source via SCM** (recomendado)

AWX → **Resources** → **Inventories** → **Add**

- **Name:** `inv-lab`
- Salvar. Dentro do inventário → **Sources** → **Add**
- **Name:** `inventory.ini (SCM)`
- **Source:** *Sourced from a Project*
- **Project:** `proj-awx-project`
- **Inventory file:** `inventory.ini`
- **Options:** *Update on launch* (e *Overwrite*, *Overwrite vars*, se preferir)
- **Save** → **Sync**.

Opção B — Inventário manual

Crie hosts/grupos manualmente (não recomendado se já está no Git).

6.4 Machine Credential (SSH para os Linux)

AWX → **Resources** → **Credentials** → **Add**

- **Type:** *Machine*
- **Name:** `ssh-admin`
- **Username:** `admin`
- **SSH Private Key:** (se chave por host) **ou Password** (se senha)
- Marque **Privilege Escalation** (become) se necessário.

(Windows você tratará depois com credencial “Machine (Windows)”/WinRM.)

6.5 Job Templates (3 modelos)

A) **OS | Full Update (DNF)**

- **Name:** `OS | Full Update (DNF)`
- **Job Type:** Run
- **Inventory:** `inv-lab`
- **Project:** `proj-awx-project`
- **Playbook:** `playbooks/os_update_full.yml`
- **Credentials:** `ssh-admin`
- **Options:** marcar *Privilege Escalation*
- **Prompt on launch:** *Limit* (para escolher host/grupo), *Variables*
- **EXTRA VARS (default sugerido):**

  ```
  os_patch_serial: "25%"
  os_patch_exclude: []
  os_patch_reboot_if_needed: true
  ```

Survey (opcional, de confirmação):

- `Confirmar_execucao` (Multiple Choice: `NAO`, `SIM`; default `NAO`; *required*).

B) **OS | Security Update (DNF)**

- **Playbook:** `playbooks/os_update_security.yml`
- **EXTRA VARS:**

  ```
  os_patch_serial: "33%"
  os_patch_exclude: ["kernel*"]
  os_patch_reboot_if_needed: true
  ```

C) **OS | Kernel Update (DNF)**

- **Playbook:** `playbooks/os_update_kernel.yml`
- **EXTRA VARS:**

  ```
  os_patch_serial: "25%"
  os_patch_reboot_if_needed: true
  ```

6.6 Schedules (janela de patch)

Em cada Template → **Schedules** → **Add**

- **Name:** `Mensal - 1º domingo 02:00`
- Recorrência conforme sua política.

6.7 Webhook (GitHub → AWX) — opcional (GitOps)

No Template desejado → **Enable Webhook**

- **Webhook Service:** GitHub
- **Webhook Key:** gere/defina um segredo. Será exibida uma **Webhook URL**. No GitHub: Repo → **Settings** → **Webhooks** → **Add**
- **Payload URL:** URL do AWX
- **Content type:** `application/json`
- **Secret:** mesma chave
- **Events:** *Just the push event*. Ao dar `git push`, o AWX dispara o Template.

---

7) Patching “K8s-safe” (opcional, para `k8s-w1`/`k8s-w2`)

> Para nós de Kubernetes, use **drain** → **patch** → **reboot** → **uncordon** com **serial: 1**. Você pode colocar estes playbooks no mesmo repo `awx-project` e criar um **Workflow** no AWX.

7.1 Requisitos

- **EE** com `kubernetes.core` (coleção) e dependências Python (kubernetes).
- **Credential** do tipo *Kubernetes/OpenShift API* **ou** *Kubeconfig* (apontando para o seu cluster).

7.2 Playbooks sugeridos

`playbooks/k8s_drain.yml`

```
cat > playbooks/k8s_drain.yml <<'EOF'
---
- name: Drain de nó Kubernetes
  hosts: "{{ target_node }}"
  gather_facts: false
  vars:
    kubeconfig_path: "~/.kube/config"   # ou use credencial do AWX
  tasks:
    - name: Drain node (cordon + evict)
      kubernetes.core.k8s_drain:
        kubeconfig: "{{ kubeconfig_path }}"
        name: "{{ inventory_hostname }}"
        state: drain
        delete_emptydir_data: true
        ignore_daemonsets: true
        timeout: 600
      delegate_to: localhost
      run_once: true
EOF
```

`playbooks/k8s_uncordon.yml`

```
cat > playbooks/k8s_uncordon.yml <<'EOF'
---
- name: Uncordon do nó
  hosts: "{{ target_node }}"
  gather_facts: false
  vars:
    kubeconfig_path: "~/.kube/config"
  tasks:
    - name: Uncordon node
      kubernetes.core.k8s_drain:
        kubeconfig: "{{ kubeconfig_path }}"
        name: "{{ inventory_hostname }}"
        state: uncordon
      delegate_to: localhost
      run_once: true
EOF
```

No AWX, crie Templates para:

1. `K8S | Drain (target_node=k8s-w1)`
2. `OS | Kernel Update (DNF) (limit=k8s-w1)`
3. `K8S | Uncordon (target_node=k8s-w1)`
4. Repita para `k8s-w2`. Em **Workflow**, encadeie **Drain** → **Update** → **Uncordon** por nó, **serial: 1**.

---

8) Backup & Restore (CRDs do Operator)

8.1 PVC de backup

```
kubectl -n awx apply -f - <<'YAML'
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: awx-backup-pvc
spec:
  accessModes: ["ReadWriteMany"]
  resources: { requests: { storage: 20Gi } }
YAML
```

8.2 Backup

```
kubectl -n awx apply -f - <<'YAML'
apiVersion: awx.ansible.com/v1beta1
kind: AWXBackup
metadata: { name: awx-backup }
spec:
  deployment_name: awx
  backup_pvc: awx-backup-pvc
  no_log: false
YAML

kubectl -n awx get jobs
```

8.3 Restore

```
kubectl -n awx apply -f - <<'YAML'
apiVersion: awx.ansible.com/v1beta1
kind: AWXRestore
metadata: { name: awx-restore }
spec:
  deployment_name: awx
  backup_pvc: awx-backup-pvc
YAML
```

---

9) Troubleshooting essencial

- **Ingress 503** → Service sem endpoints:

  ```
  kubectl -n awx get endpoints awx-service -o wide
  ```

  Se vazio/notReady, verifique `awx-web` (logs, memória, SECRET_KEY, migrations).

- **`SECRET_KEY must not be empty`**:

  ```
  kubectl -n awx get secret awx-secret-key -o jsonpath='{.data.secret_key}'
  | base64 --decode; echo
  kubectl -n awx patch awx awx --type merge -p '{"spec":
  {"secret_key_secret":"awx-secret-key",
    "extra_env":[{"name":"SECRET_KEY","valueFrom":{"secretKeyRef":
  {"name":"awx-secret-key","key":"secret_key"}}}]}}'
  kubectl -n awx delete pod -l app.kubernetes.io/name=awx-web
  ```

- **OOM/CrashLoop em `awx-web`** → aumente `web_resource_requirements` e recrie o pod.
- **RWX/NFS** com erro de permissão → ajuste export e recrie `awx-task` para remontar.

---

10) Checklist final (itens 5+)

- ☑ Repo **GitHub `awx-project`** criado e populado
- ☑ Estrutura: `roles/`, `playbooks/`, `inventory.ini`, `group_vars/`
- ☑ Project no AWX sincronizando do Git (Update on launch)
- ☑ Inventory via **SCM Source** apontando para `inventory.ini`
- ☑ Machine Credential (SSH) configurada
- ☑ Job Templates DNF criados (Full/Security/Kernel)
- ☑ Schedules e/ou Webhooks configurados
- ☑ (Opcional) Workflow **K8s-safe** (drain → update → uncordon)
- ☑ Backup CRDs testado (AWXBackup/AWXRestore)

Se quiser, gero um **workflow YAML exportado** do AWX com os nós e conexões (Drain/Update/Uncordon) para você importar e rodar direto.