REPORTE DE AVANCE DE PRÁCTICAS PROFESIONALES 2

1. Datos Generales

- Nombre del Estudiante: Sergio Andres Cantor Moya
- Carrera: Ingeniería de sistemas
- Institución/Empresa: Universidad de los Andes
- Supervisor en la Empresa: Maria Jose Afanador
- Fecha de Entrega del Reporte: 11/11/2025

2. Actividades Realizadas

#	Actividad	Descripción de la tarea realizada	Producto o Evidencia
1	Automatización de Inventario	Implementación de sistema que actualiza automáticamente el inventario de equipos desde reportes en GitHub	Script YAML de actualización de inventario
2	Configuración de Autologin	Habilitación de inicio de sesión automático para usuario "estudiantes" en equipos de la sala T-201	Playbook de autologin y registro de ejecución
3	Preparación de Instalaciones	Desarrollo de sistema para instalación remota y silenciosa de aplicaciones en los equipos	Playbook de instalación de software

3. Productos y Evidencias Entregadas

• **Sistema de Gestión de Inventario Automatizado**: Script en Ansible que sincroniza las direcciones IP de los equipos desde reportes generados automáticamente en GitHub

• **Configuración de Autologon para Sala T-201**: Playbook que habilita el inicio de sesión automático para el usuario estudiantes, incluyendo reinicio automático de equipos

• **Sistema de Despliegue Remoto de Software**: Conjunto de playbooks para la instalación masiva de aplicaciones en los equipos de cómputo de manera desatendida

4. Porcentaje de Cumplimiento (Autovaloración del estudiante y calificación del supervisor sobre el cumplimiento de las actividades asignadas)

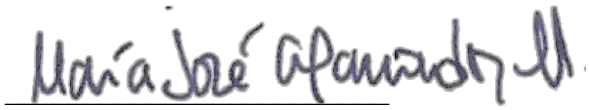
Aspecto Evaluado	Autoevaluación (%)	Evaluación del Supervisor (%)
Puntualidad y asistencia	100	100
Calidad de trabajo	100	100
Responsabilidad y compromiso	100	100
Cumplimiento de tareas asignadas	100	100
Trabajo en equipo y comunicación	100	100

Aspecto Evaluado	Autoevaluación (%)	Evaluación del Supervisor (%)
Total de cumplimiento promedio (%)	100	100

5. Comentarios y Observaciones (Espacio para sugerencias o comentarios tanto del estudiante como del supervisor)

Sergio está avanzando satisfactoriamente. Le recomiendo contactar a otros pares de la universidades que manejen salas de cómputo para conversar sobre retos del proyecto de práctica y conocer posibilidades para solucionar los retos para cumplir completamente con los objetivos de la práctica.

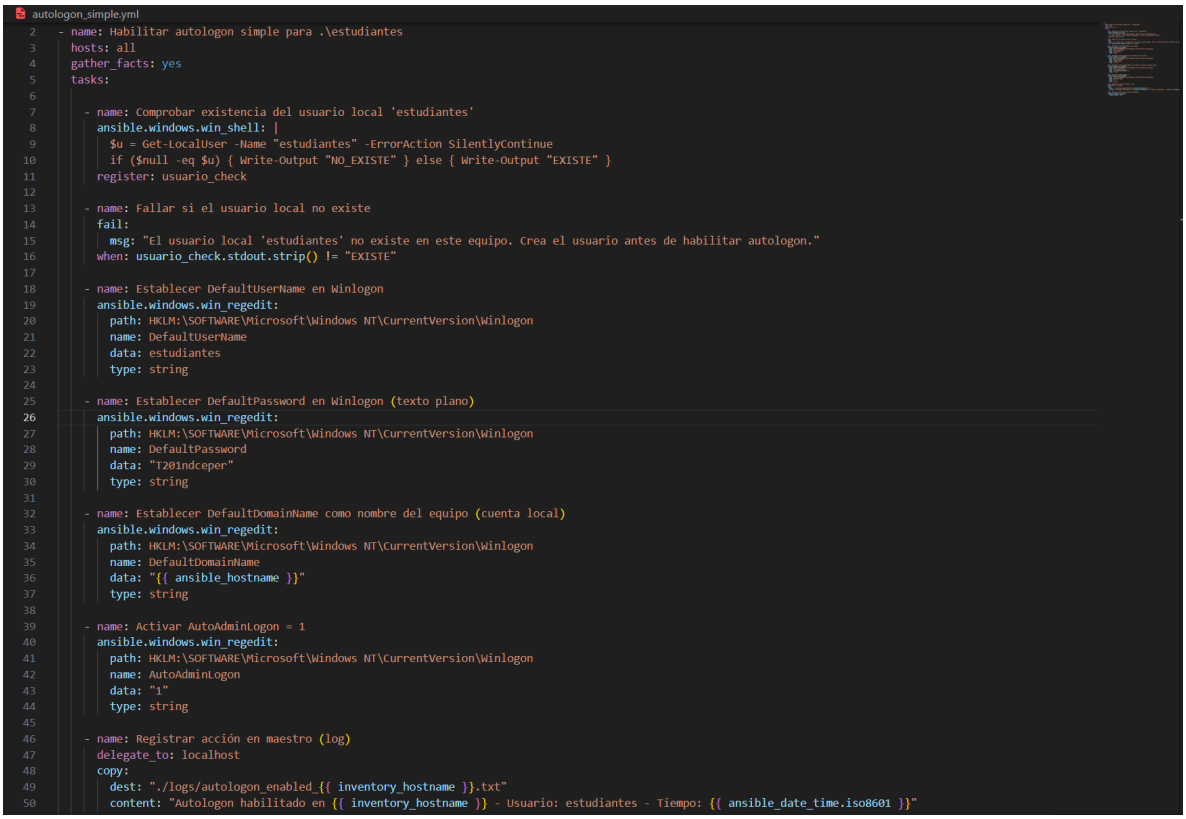
Firma del Estudiante: 

Firma del Supervisor: 

Avance del Proyecto O.R.I.O.N. - Análisis Técnico de Playbooks

1. autologon_simple.yml

Propósito: Configurar inicio de sesión automático para el usuario local "estudiantes".



Explicación:

- Utiliza PowerShell para verificar la existencia del usuario mediante el cmdlet Get-LocalUser, manejando silenciosamente errores y registrando el resultado
- Modifica el registro de Windows en la ruta específica de Winlogon para definir el usuario que iniciará sesión automáticamente.
- Configura la contraseña del usuario en texto plano en el registro (consideración de seguridad: esto expone la contraseña en el registro).

- Define el dominio como el nombre del equipo local, indicando que es una cuenta local y no de dominio.
- Habilita la funcionalidad de inicio de sesión automático estableciendo el valor en 1.

2. test_inv.yml

Propósito: Configurar el reporte automático de direcciones IP hacia GitHub al inicio de sesión.

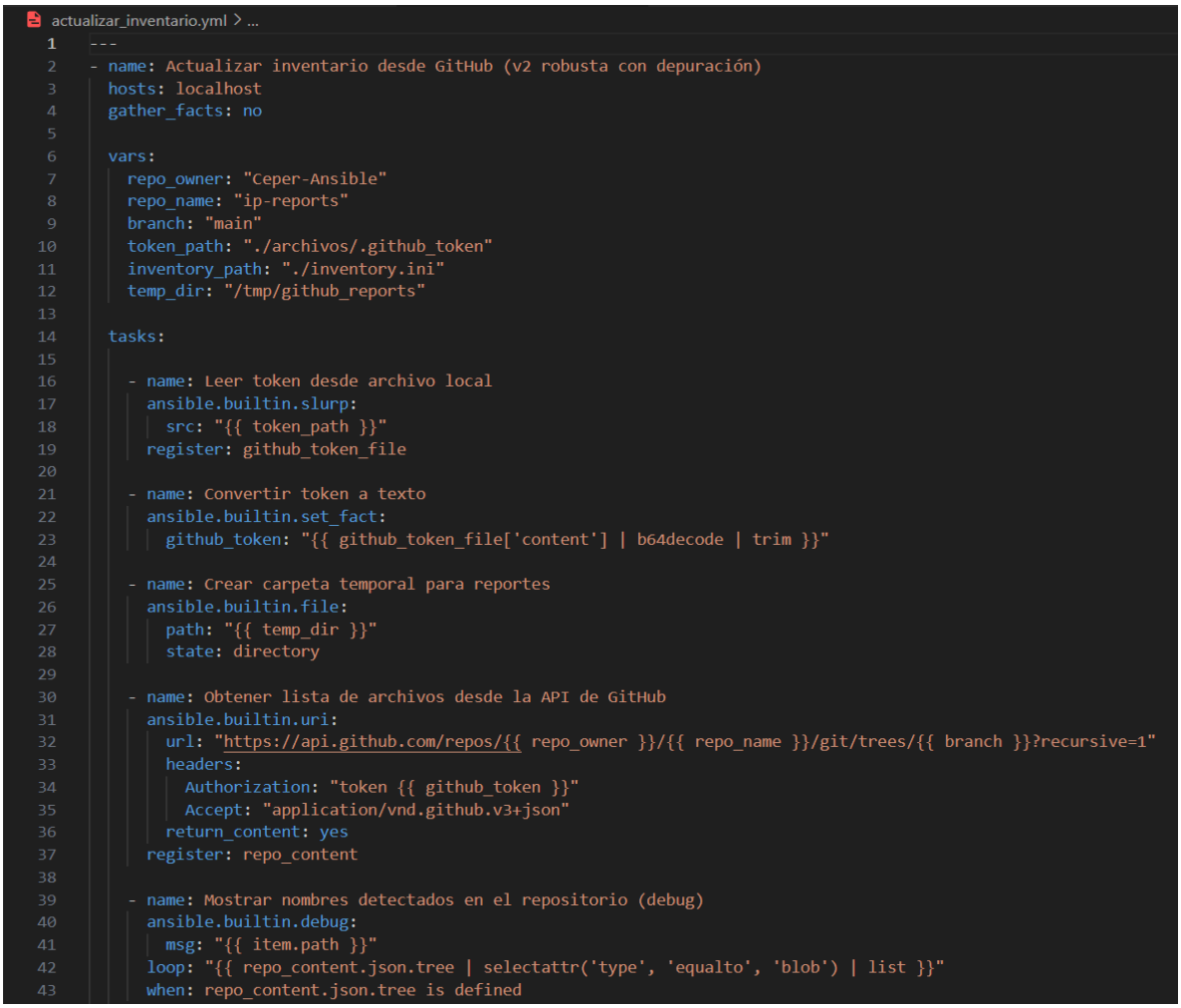
Explicación:

- Copia el token de GitHub a una ubicación pública en el equipo remoto.
- Configura permisos NTFS para que todos los usuarios puedan leer el token:
    - /inheritance:e: Habilita herencia
    - /grant *S-1-1-0:R: Otorga permiso de lectura a todos los usuarios (SID S-1-1-0)

- Crea un script por lotes en la carpeta de inicio automático que ejecuta el script PowerShell al loguearse, con parámetros:
    - Bypass: Omite la política de ejecución
    - Hidden: Ejecución en segundo plano
    - Redirección de salida a archivo de log

3. actualizar_inventario.yml

Propósito: Sistema automatizado que mantiene actualizado el inventario de equipos consultando reportes de IP almacenados en GitHub.



actualizar_inventario.yml > ...

```yaml
---
- name: Actualizar inventario desde GitHub (v2 robusta con depuración)
  hosts: localhost
  gather_facts: no

  vars:
    repo_owner: "Ceper-Ansible"
    repo_name: "ip-reports"
    branch: "main"
    token_path: "./archivos/.github_token"
    inventory_path: "./inventory.ini"
    temp_dir: "/tmp/github_reports"

  tasks:

    - name: Leer token desde archivo local
      ansible.builtin.slurp:
        src: "{{ token_path }}"
      register: github_token_file

    - name: Convertir token a texto
      ansible.builtin.set_fact:
        github_token: "{{ github_token_file['content'] | b64decode | trim }}"

    - name: Crear carpeta temporal para reportes
      ansible.builtin.file:
        path: "{{ temp_dir }}"
        state: directory

    - name: Obtener lista de archivos desde la API de GitHub
      ansible.builtin.uri:
        url: "https://api.github.com/repos/{{ repo_owner }}/{{ repo_name }}/git/trees/{{ branch }}?recursive=1"
        headers:
          Authorization: "token {{ github_token }}"
          Accept: "application/vnd.github.v3+json"
        return_content: yes
      register: repo_content

    - name: Mostrar nombres detectados en el repositorio (debug)
      ansible.builtin.debug:
        msg: "{{ item.path }}"
      loop: "{{ repo_content.json.tree | selectattr('type', 'equalto', 'blob') | list }}"
      when: repo_content.json.tree is defined
```

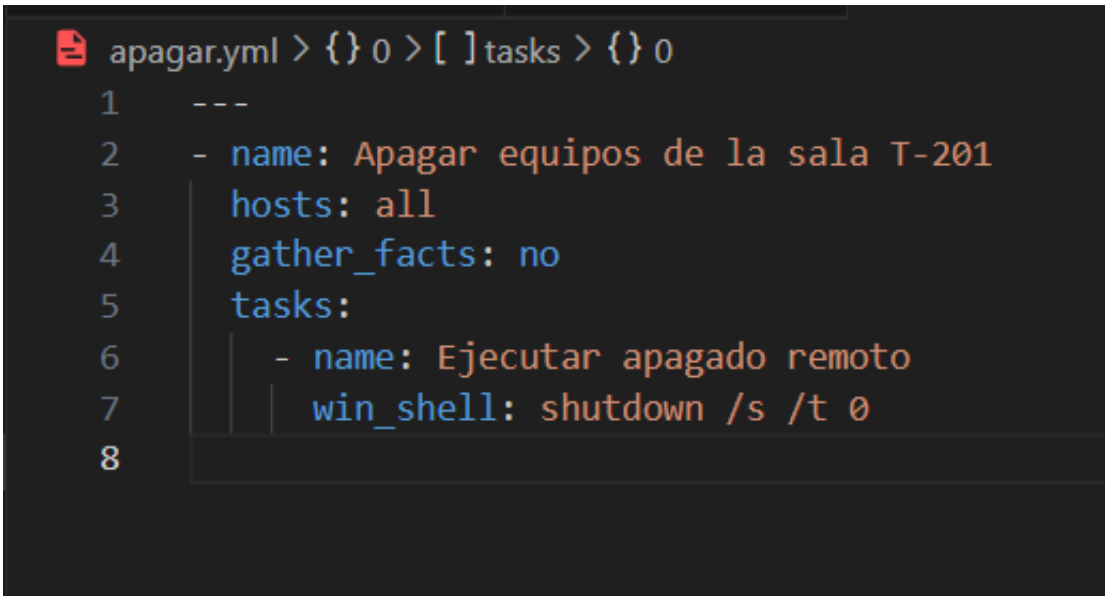
```yaml
- name: Descargar archivos detectados (sin filtrar extensión)
  ansible.builtin.get_url:
    url: "https://raw.githubusercontent.com/{{ repo_owner }}/{{ repo_name }}/{{ branch }}/{{ item.path }}"
    headers:
      Authorization: "token {{ github_token }}"
    dest: "{{ temp_dir }}/{{ item.path | basename }}"
    force: yes
  loop: "{{ repo_content.json.tree | selectattr('type', 'equalto', 'blob') | list }}"
  register: downloads
  when: repo_content.json.tree | length > 0

- name: Regenerar inventario
  ansible.builtin.copy:
    dest: "{{ inventory_path }}"
    content: |
      [windows]
      {% for f in downloads.results if f.dest is defined %}
      {% for line in lookup('file', f.dest).splitlines() %}
      {{ line }}
      {% endfor %}
      {% endfor %}
  when: downloads is defined
```

Explicación:

- Lee el token de autenticación de GitHub desde un archivo local en formato binario y lo almacena en una variable temporal.
- Decodifica el token desde base64 a texto plano y elimina espacios en blanco, preparándolo para su uso en la API.
- Realiza una petición HTTP GET a la API de GitHub para obtener la estructura completa del repositorio, incluyendo todos los archivos y directorios.
- Itera sobre cada archivo encontrado en el repositorio (filtrando solo archivos, no directorios) y los descarga usando URLs directas de GitHub Raw.
- Construye dinámicamente el archivo inventory.ini concatenando el contenido de todos los archivos descargados, manteniendo la sección [windows] y agregando cada línea de los reportes.

4. **apagar.yml**

Propósito: Ejecutar apagado remoto simultáneo de todos los equipos en el inventario.

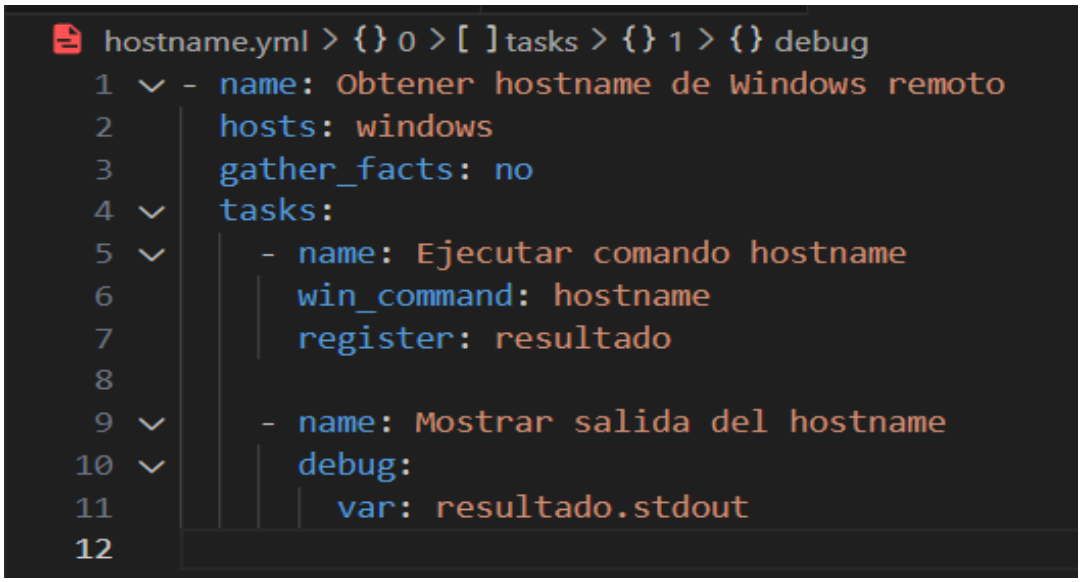


Explicación:

- Define que el playbook se ejecutará contra todos los hosts del inventario sin recopilar información del sistema para mayor velocidad.
- Ejecuta el comando de Windows shutdown con parámetros:
    - /s: Apagar el sistema
    - /t 0: Sin tiempo de espera (0 segundos)

5. hostname.yml

Propósito: Obtener y mostrar los nombres de host de todos los equipos en la red.

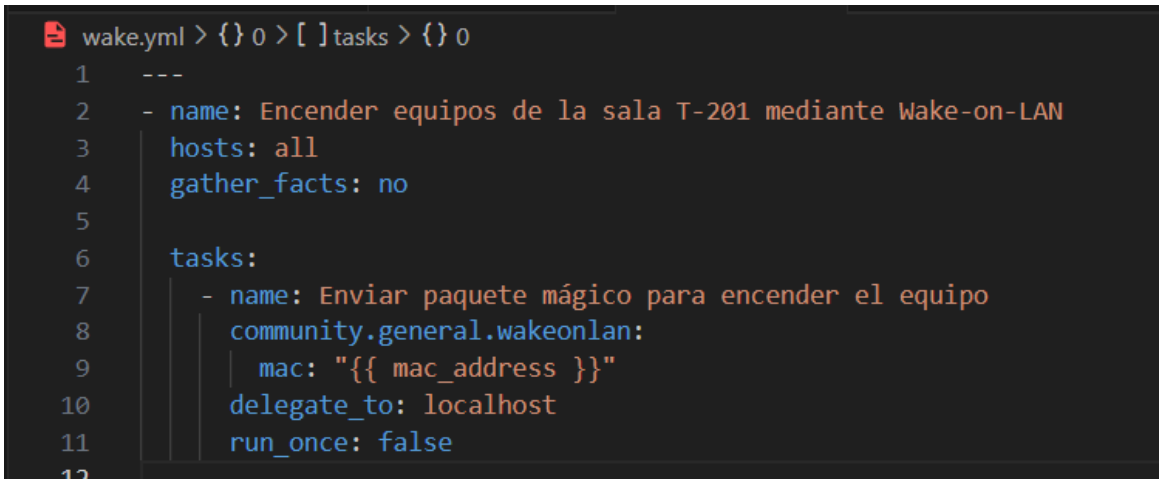


Explicación:

- Se ejecuta específicamente sobre el grupo "windows" del inventario, deshabilitando la recolección de facts.
- Ejecuta el comando nativo hostname de Windows y almacena el resultado en una variable.
- Muestra en la salida de Ansible el nombre del host obtenido del comando anterior.

6. wake.yml

Propósito: Encender equipos remotamente usando Wake-on-LAN.



Explicación:

- Utiliza el módulo wakeonlan de Ansible para enviar paquetes "Magic Packet" a cada dirección MAC definida en el inventario. El parámetro run_once: false asegura que se ejecute para cada host individual.
- **Problema identificado:** Los switches de red de la universidad posiblemente bloquean los paquetes Wake-on-LAN o los equipos no tienen habilitada esta funcionalidad en BIOS.

7. winget.yml

Propósito: Actualizar automáticamente aplicaciones usando el gestor de paquetes Winget de Windows.

winget.yml > ...

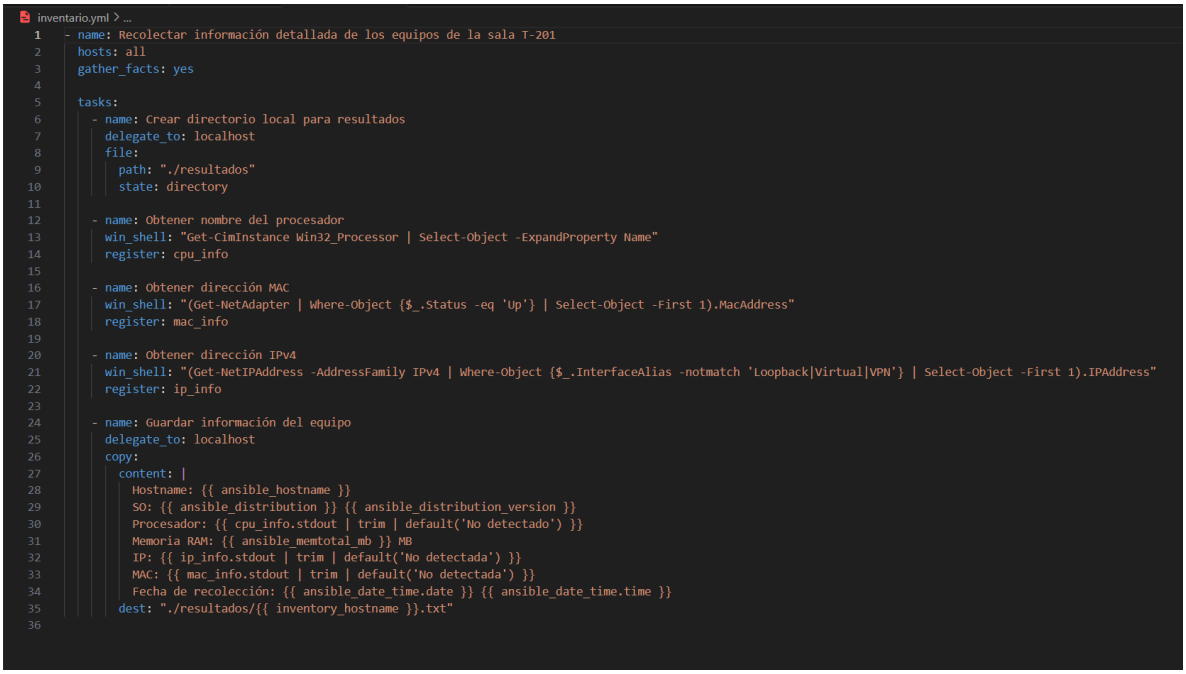
```yaml
---
- name: Actualización automática y silenciosa con Winget
  hosts: all
  gather_facts: yes

  tasks:
    - name: Crear directorio local para logs
      delegate_to: localhost
      file:
        path: "./logs"
        state: directory

    - name: Buscar ruta real de winget
      win_shell: |
        $paths = @(
          "$env:LOCALAPPDATA\Microsoft\WindowsApps\winget.exe",
          "$env:ProgramFiles\WindowsApps\Microsoft.DesktopAppInstaller*\winget.exe",
          "C:\Program Files\WindowsApps\Microsoft.DesktopAppInstaller_*\winget.exe"
        )
        foreach ($p in $paths) {
          $resolved = Get-Item -Path $p -ErrorAction SilentlyContinue
          if ($resolved) { Write-Output $resolved.FullName; break }
        }
      register: winget_path

    - name: Ejecutar actualización si winget está disponible
      when: winget_path.stdout | trim != ""
      win_shell: >
        & "{{ winget_path.stdout | trim }}" upgrade --all
        --accept-package-agreements --accept-source-agreements --silent;
        Write-Output "Actualización completada correctamente."
      register: resultado
      ignore_errors: yes

    - name: Registrar salida de actualización
      when: resultado is defined
      debug:
        msg: "{{ resultado.stdout | default('Sin salida visible (modo silencioso)') }}"

    - name: Registrar si winget no está disponible
      when: winget_path.stdout | trim == ""
      debug:
        msg: "⚠️ Winget no encontrado en {{ inventory_hostname }}. Instálalo con App Installer desde Microsoft Store."

    - name: Guardar log de ejecución
      delegate_to: localhost
      copy:
        content: |
          === LOG DE ACTUALIZACIÓN WINGET ===
          Equipo: {{ inventory_hostname }}
          Fecha: {{ ansible_date_time.date }} {{ ansible_date_time.time }}
          Ruta Winget: {{ winget_path.stdout | default('No detectado') }}
          {{ resultado.stdout | default('Sin salida (modo silencioso)') }}
          Error:
          {{ resultado.stderr | default('Sin errores') }}
        dest: "./logs/winget_{{ inventory_hostname }}.txt"

    - name: Crear tarea programada para actualizaciones nocturnas
      when: winget_path.stdout | trim != ""
      win_scheduled_task:
        name: "WingetAutoUpdate"
        description: "Actualiza aplicaciones automáticamente cada noche a las 2:00 AM"
        actions:
          - path: "{{ winget_path.stdout | trim }}"
            arguments: "upgrade --all --accept-package-agreements --accept-source-agreements --silent"
        triggers:
          - type: daily
            start_boundary: '2025-10-29T02:00:00'
        username: "{{ ansible_user }}"
        password: "{{ ansible_password }}"
        state: present
        enabled: yes
```

Explicación:

- Busca winget.exe en múltiples rutas comunes donde se instala, manejando la variabilidad de rutas con caracteres comodín.
- Ejecuta winget con parámetros de actualización masiva:
  - upgrade --all: Actualiza todas las aplicaciones instaladas
  - --accept-*-agreements: Acepta automáticamente licencias
  - --silent: Modo no interactivo
- **Problema identificado:** Aunque Winget está presente en Windows, la ejecución remota no encuentra el ejecutable en las rutas esperadas, posiblemente por restricciones de AppX deployment o contextos de usuario.

8. inventario.yml

Propósito: Recolectar información detallada del hardware y configuración de todos los equipos.



Explicación:

- Utiliza WMI a través de PowerShell para obtener el modelo exacto del procesador.
- Obtiene la dirección MAC del primer adaptador de red con estado "Up" (conectado).
- Filtra las direcciones IPv4 excluyendo interfaces virtuales, de loopback y VPN, tomando la primera dirección válida.
- Crea un archivo de texto local con formato legible que consolida toda la información recolectada, usando variables de Ansible y resultados de comandos.

9. instalar_apps.yml

Propósito: Instalar aplicaciones de manera remota y simultánea en todos los equipos.

instalar_apps.yml > ...

```yaml
---
- name: Instalar todas las aplicaciones desde la carpeta 'archivos'
  hosts: all
  gather_facts: no

  tasks:
    - name: Crear carpeta temporal en el equipo remoto
      ansible.windows.win_file:
        path: C:\Temp
        state: directory

    - name: Obtener lista de instaladores locales (.exe y .msi)
      find:
        paths: ./archivos
        patterns: "*.exe,*.msi"
      register: lista_instaladores
      delegate_to: localhost

    - name: Copiar instaladores al equipo remoto
      ansible.windows.win_copy:
        src: "{{ item.path }}"
        dest: "C:\\Temp\\{{ item.path | basename }}"
      loop: "{{ lista_instaladores.files }}"

    - name: Instalar los archivos .msi en modo silencioso
      ansible.windows.win_package:
        path: "C:\\Temp\\{{ item.path | basename }}"
        arguments: "/qn /norestart"
        state: present
      loop: "{{ lista_instaladores.files | selectattr('path', 'search', '.msi$') | list }}"
      register: resultado_msi
      ignore_errors: yes

    - name: Mostrar resumen de instalación
      debug:
        msg:
          - "Instalaciones MSI: {{ resultado_msi.results | map(attribute='rc') | list }}"
```

Explicación:

- Busca recursivamente en la carpeta local "archivos" todos los archivos con extensiones .exe y .msi, registrando la lista encontrada.
- Transfiere cada instalador encontrado al directorio C:\Temp del equipo remoto, manteniendo el nombre original del archivo.
- Para cada archivo MSI, ejecuta la instalación con parámetros:
    - /qn: Interfaz silenciosa sin interacción del usuario
    - /norestart: Evita reinicios automáticos