

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

CLASIFICACIÓN Y CONFIDENCIALIDAD

Este documento es clasificado como **“Uso Interno”**.

El presente documento es propiedad del grupo Keralty y está restringido a los colaboradores de la organización que cuenten con la autorización expresa para su consulta.

No se permite la reproducción total o parcial de este documento, así como su transmisión a terceros sin la autorización del responsable designado por el grupo Keralty.

LISTA DE DISTRIBUCIÓN

Este documento es de uso interno del grupo Keralty y su copia debe ser controlada y registrada de acuerdo con los procedimientos establecidos por la organización. Su distribución se debe realizar de acuerdo con la lista definida en la tabla de distribución maestra SGSI.

Todo cambio realizado a este documento debe ser controlado, documentado de acuerdo con el procedimiento de control documental y registrados en la tabla de control de cambios del presente documento.

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

TABLA DE CONTENIDO

1. OBJETIVO.	3
2. ALCANCE.	3
3. DEFINICIONES	3
4. CONTENIDO.	8
4.1. Arquitectura de Ansible	8
4.2. Balanceo de la Aplicación	10
4.3. Integración de GIT para la creación de proyectos	11
4.4. Integración con Azure AD	12
4.5. Proceso Automatizado de Reinicios	12
4.5.1. Caso de Uso de Reinicios	12
4.5.2. Parchado Automatizado de Servidores Avicena Aplicación, Avicena Reportes y Avicena Web	13
4.5.3. Inventario	13
4.5.4. Encabezado	14
4.5.5. Toma de Evidencias Pre-Update	15
4.5.6. Aplicación de Parchado Sobre los Servidores de Avicena	17
4.5.7. Reinicio del Primer Grupo de Servidores	17
4.5.8. Subir Instancias del Primer Bloque de Servidores Avicena Reiniciados	18
4.5.9. Reinicio del Segundo Bloque de Servidores	19
4.5.10. Toma de Evidencia Post-Update	19
5. CONTROL DE CAMBIOS.	21
6. FLUJO DE APROBACIÓN.	22

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

1. OBJETIVO.

Describir el proceso automatizado de actualización (update) en los servidores de Avicena, Avicena Reportes y Avicena Web con tareas distribuidas en pre-update, aplicación del update, reinicio de servidores, subida de instancias y generación de evidencias.

2. ALCANCE.

Este documento contempla la actualización (update) en los servidores de Avicena, Avicena Reportes y Avicena Web, tomando evidencias de los paquetes instalados antes de aplicar el update, aplicación del update, reinicio de servidores, subida de instancias y generación de evidencias después de realizar el update.

3. DEFINICIONES

Ansible Automation Platform: Es una plataforma de automatización de código abierto diseñada para la gestión de configuraciones, implementación de aplicaciones, automatización de tareas y orquestación de tecnologías de la información. Simplifica tareas y flujos de trabajo complejos al permitir que los usuarios definan la automatización en un lenguaje declarativo, utilizando YAML.

Alta Disponibilidad: Se refiere a un sistema o componente que está diseñado y configurado de tal manera que garantiza un funcionamiento continuo durante un período de tiempo largo, minimizando el tiempo de inactividad y asegurando que los servicios o aplicaciones estén disponibles de manera constante, incluso en caso de fallos o interrupciones.

Automatización: Es el proceso de realizar tareas, operaciones o procesos de manera automática, sin intervención humana directa.

API: (Application Programming Interface), es un conjunto de reglas y herramientas que permite que diferentes software se comuniquen entre sí. Proporciona un conjunto de funciones y métodos que permiten a desarrolladores de software acceder a las características o datos de una aplicación, servicio o plataforma sin tener que entender su implementación interna.

Arquitectura: Es el marco organizativo que define cómo los diferentes componentes de un sistema interactúan entre sí y cómo se organizan para lograr los objetivos previstos.

Automation Controller: Es un dispositivo o software que gestiona y coordina la automatización de varias tareas o procesos dentro de un sistema. Sirve como un punto centralizado de control para orquestar y ejecutar flujos de trabajo de

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

automatización, asegurando que diferentes componentes trabajen juntos de manera fluida. Los controladores de automatización desempeñan un papel crucial en mejorar la eficiencia, reducir la intervención manual y mantener la consistencia en los procesos automatizados.

Azure: Es una plataforma de servicios en la nube ofrecida por Microsoft. Microsoft Azure proporciona una amplia gama de servicios de computación en la nube, almacenamiento, bases de datos, redes, inteligencia artificial, análisis y más. Estos servicios permiten a las organizaciones construir, implementar y administrar aplicaciones y servicios a través de la infraestructura global de centros de datos de Microsoft.

AD: (Active Directory), es un servicio de directorio desarrollado por Microsoft que se utiliza para almacenar información sobre objetos en una red y facilitar la búsqueda y administración de estos objetos.

Background: Se refiere a un proceso en segundo plano es una tarea o programa que se ejecuta sin interferir directamente con la interacción del usuario. Puede ser una tarea automática, un servicio o una operación que ocurre mientras el usuario realiza otras acciones.

Balanceo: Es el proceso de distribuir de manera equitativa la carga o la demanda entre varios recursos para optimizar el rendimiento, la eficiencia y la disponibilidad. Esta práctica es comúnmente utilizada en diversos contextos tecnológicos para asegurar que los recursos se utilicen de manera equilibrada y que ningún componente esté sobrecargado mientras otros están subutilizados.

Base de Datos: Es un conjunto organizado de datos que se almacenan de manera estructurada y que están accesibles de forma electrónica. En una base de datos, la información se organiza en tablas, que contienen filas y columnas. Cada fila en una tabla representa una entidad individual, mientras que cada columna representa un atributo específico de esas entidades. Las bases de datos permiten almacenar, gestionar, recuperar y actualizar datos de manera eficiente.

Comando: Es una instrucción específica que se da a un sistema informático para llevar a cabo una tarea o realizar una operación.

Caso de Uso: Es una descripción detallada de cómo un usuario interactúa con un sistema para lograr un objetivo específico. Los casos de uso son una herramienta fundamental en el análisis y diseño de sistemas, particularmente en el desarrollo de software, ya que ayudan a capturar los requisitos funcionales del sistema desde la perspectiva del usuario final.

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

Directorio: Es una estructura de almacenamiento que organiza y gestiona archivos y otros directorios en un sistema de archivos. Los directorios permiten agrupar archivos de manera lógica, facilitando la organización, administración y acceso a los datos almacenados en un dispositivo de almacenamiento como un disco duro, una unidad de estado sólido, una unidad de red, etc.

Execution Environment: Es el conjunto de condiciones y recursos en los que se lleva a cabo la ejecución de un programa o aplicación informática. El entorno de ejecución proporciona el contexto necesario para que un software se ejecute correctamente y alcance sus objetivos.

Flujo: Es el movimiento o la secuencia de datos, información o control a través de un sistema o programa.

GIT: Es un sistema de control de versiones distribuido ampliamente utilizado para el seguimiento de cambios en el código fuente durante el desarrollo de software. Fue creado por Linus Torvalds en 2005 y es conocido por su velocidad, flexibilidad y capacidad para gestionar proyectos de cualquier tamaño. Git es particularmente popular en la comunidad de desarrollo de software de código abierto.

Instancia: Este término se refiere a una ocurrencia específica de un objeto, dato, proceso o servicio, dependiendo del contexto en el que se esté utilizando.

Interfaz Web: Es la parte visual y funcional de un sitio web o aplicación web con la que los usuarios interactúan. Es el medio a través del cual los usuarios pueden acceder y utilizar las funciones y contenidos de una página web.

Inventario: En Ansible, es el archivo que contiene información sobre los nodos que Ansible gestionará. Este archivo describe los hosts (máquinas) y grupos de hosts que forman parte de la infraestructura que se desea administrar con Ansible.

IP: Es un conjunto de reglas que rigen el formato de los datos enviados a través de internet o una red local. La IP permite que los datos se transmitan entre dispositivos (computadoras, servidores, impresoras, etc.) en una red. Cada dispositivo conectado a una red que utiliza el Protocolo de Internet se identifica de manera única mediante una dirección IP.

Interfaz de Red: Es un componente de hardware o software que conecta una computadora u otro dispositivo a una red de computadoras. La interfaz de red permite la comunicación y el intercambio de datos entre dispositivos en la red, actuando como un intermediario entre el dispositivo y los medios de transmisión, como cables de red o señales inalámbricas.

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

JBoss: Es una marca que abarca una serie de productos y proyectos de software, principalmente relacionados con el desarrollo y la implementación de aplicaciones empresariales en Java. El más conocido de estos productos es el JBoss Enterprise Application Platform (EAP), un servidor de aplicaciones de código abierto que implementa la especificación Java EE (Java Enterprise Edition).

Killlearn: Se refiere a la acción de terminar o detener un proceso o tarea en ejecución en un sistema operativo. Este término se usa comúnmente en el contexto de la administración de sistemas y programación, especialmente en entornos Unix y Linux, donde se utilizan comandos específicos para finalizar procesos.

On-premise: Se refiere a la ubicación física de sistemas informáticos, infraestructura y software en las instalaciones físicas de una organización o empresa, en lugar de utilizar recursos de la nube o servicios alojados externamente. En un entorno on-premise, las organizaciones poseen, mantienen y gestionan su propia infraestructura de tecnología de la información (TI) dentro de sus propias instalaciones, ya sea en centros de datos internos o en servidores locales. Esto implica que tienen el control total sobre la configuración, el mantenimiento, la seguridad y la gestión de todos los componentes tecnológicos utilizados para respaldar sus operaciones comerciales.

Oracle Database: Es conocido por su escalabilidad, fiabilidad y capacidades avanzadas de gestión de datos. Ofrece soporte para una variedad de modelos de datos, incluyendo datos estructurados y no estructurados, así como también incluye características de seguridad, alta disponibilidad y rendimiento optimizado. Oracle Database se utiliza en una amplia variedad de aplicaciones empresariales críticas en industrias como finanzas, telecomunicaciones, comercio electrónico, atención médica, entre otras.

Playbook: Es un archivo que contiene una serie de instrucciones (tareas) que describen los pasos que Ansible debe seguir para configurar, administrar o automatizar un conjunto de hosts. Los playbooks son escritos en YAML (YAML Ain't Markup Language) y proporcionan una forma de expresar la configuración deseada y las tareas a realizar en un formato claro y legible.

PostgreSQL: Es un sistema de gestión de bases de datos relacional de código abierto (RDBMS, por sus siglas en inglés) que enfatiza la extensibilidad y la conformidad con los estándares. Es uno de los sistemas de bases de datos más avanzados y potentes disponibles en la actualidad. PostgreSQL es conocido por su capacidad para manejar grandes cantidades de datos, su soporte para consultas complejas y su compromiso con los estándares del lenguaje SQL.

Private Automation Hub: Es un repositorio local que permite a las empresas con entornos desconectados gestionar, compartir y organizar el contenido generado de forma interna y controlar el acceso al contenido creado

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

PID: Se refiere a un número único asignado por el sistema operativo a cada proceso en ejecución. Este identificador se utiliza para gestionar y controlar los procesos dentro del sistema operativo.

Rama: Se refiere a una línea independiente de desarrollo dentro de un sistema de control de versiones. Las ramas permiten a los desarrolladores trabajar en paralelo en diferentes características, correcciones de errores o experimentos sin afectar la base de código principal. Este concepto es fundamental en sistemas de control de versiones como Git, Mercurial y Subversion.

Repositorio: Es el lugar o sistema donde se almacenan y gestionan archivos y recursos relacionados con un proyecto.

Script: Es un conjunto de instrucciones o comandos escritos en un lenguaje de programación que se utiliza para realizar una tarea específica o automatizar una serie de acciones en un sistema informático. Los scripts son secuencias de comandos que ejecutan tareas sin la necesidad de una intervención manual constante.

Servidor: Es un sistema o software diseñado para proporcionar servicios, recursos o funciones a otros dispositivos o programas, conocidos como "clientes". Los servidores desempeñan un papel fundamental en la comunicación y el intercambio de información en redes de computadoras.

Temporales: Son archivos creados por programas y sistemas operativos para almacenar datos de manera transitoria. Estos archivos se utilizan para una variedad de propósitos, incluyendo el almacenamiento intermedio de datos durante el procesamiento, la gestión de memoria, y la facilitación de la comunicación entre procesos. Los archivos temporales suelen ser eliminados automáticamente una vez que ya no son necesarios, aunque en algunos casos pueden necesitar ser gestionados manualmente.

Terminal: Es una interfaz de usuario que permite a los usuarios interactuar con el sistema operativo mediante la introducción de comandos de texto. Las terminales son esenciales para la administración de sistemas, el desarrollo de software y diversas tareas de programación y automatización. Existen dos tipos principales de terminales: terminales físicas y terminales virtuales.

Token: Es una clave de acceso o un identificador único que se utiliza para autenticar y autorizar el acceso a un recurso, como una red, un sistema o un servicio.

Usuario: Se refiere a una persona, cuenta o entidad que interactúa con un sistema informático, una aplicación o un servicio en línea. El término puede tener diferentes contextos y connotaciones en función de la aplicación específica.

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

VLAN: Es una tecnología de redes que permite la segmentación lógica de una red física en múltiples redes virtuales. A través de la implementación de VLANs, se pueden crear segmentos de red independientes, incluso si los dispositivos están conectados al mismo switch físico. Cada VLAN actúa como una red de área local (LAN) separada.

Workflow: Es una secuencia de tareas, pasos o procesos que deben ser realizados para completar un objetivo específico. Los workflows son utilizados para estructurar y automatizar procesos repetitivos y complejos, asegurando que las tareas se realicen de manera eficiente, coherente y en el orden correcto.

YAML: Es un lenguaje de serialización de datos que las personas pueden comprender y suele utilizarse en el diseño de archivos de configuración. Para algunas personas, YAML significa otro lenguaje de marcado más; para otras, es un acrónimo recursivo que quiere decir "YAML no es un lenguaje de marcado", lo que enfatiza la idea de que se utiliza para los datos, no para los documentos. Es un lenguaje de programación popular porque está diseñado para que sea fácil de leer y entender. También se puede utilizar junto con otros lenguajes de programación.

4. CONTENIDO.

4.1. Arquitectura de Ansible

La arquitectura implementada para Ansible Automation Platform se compone de seis (6) servidores on-premise montados en las VLAN productivas las cuales son la VLAN 1 (10.160.1.*) y la VLAN 2 (10.160.2.*) distribuidos de la siguiente manera:

- ❖ Tres (3) servidores que contienen los componentes del Automation Controller los cuales incluyen la interfaz web, la API y el motor de configuración de Ansible, los servidores en mención son los siguientes:

10.160.1.235 --- srvvkansiblet1.colsanitas.com

10.160.1.236 --- srvvkansiblet2.colsanitas.com

10.160.1.237 --- srvvkansiblet3.colsanitas.com

- ❖ Un (1) servidor para el Private Automation Hub, que proporciona los medios para administrar el contenido de ansible y un registry para almacenar los Execution Environments personalizados, el servidor en mención es el siguiente:

10.160.1.239 --- srvvkansiblehub.colsanitas.com

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

- ❖ Un (1) servidor para almacenar la instancia de la base de datos PostgreSQL v13, el cual mantiene 2 bases de datos: una para los controller y otra para el private automation hub, el servidor en mencion es el siguiente:

10.160.1.238 --- srvvkansiblebd1.colsanitas.com

- ❖ Un (1) servidor que contiene las herramientas necesarias para desarrollar y probar playbooks y execution environments para AAP, el servidor en mencion es el siguiente:

10.160.2.37 --- srvvkansibledevsrv.colsanitas.com

A continuación, se muestra el diagrama de la arquitectura de Ansible:

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

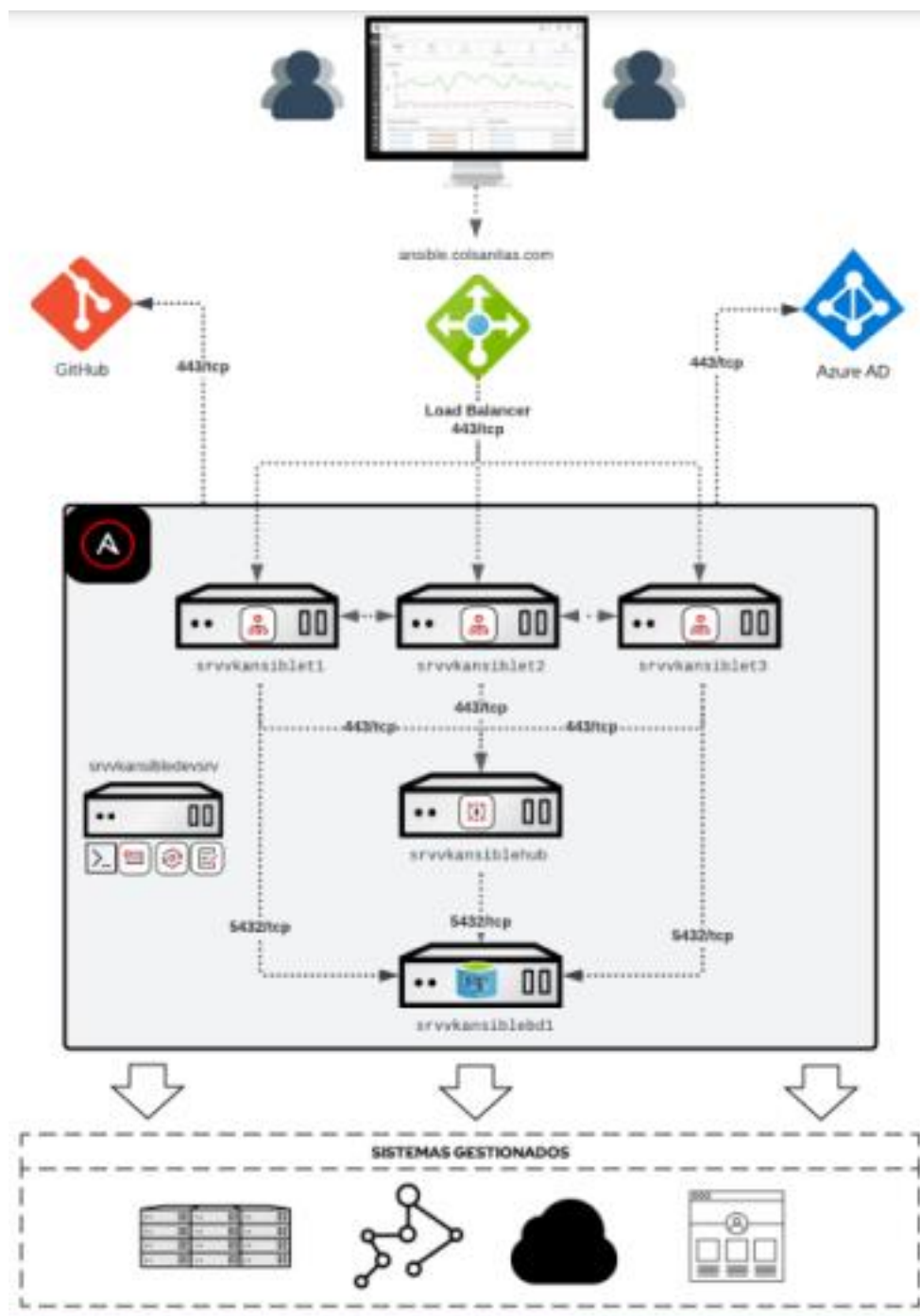


Figura 1. Arquitectura de Ansible

4.2. Balanceo de la Aplicación

El acceso a la consola de Ansible Automation Platform (AAP) se encuentra balanceada en la siguiente URL: <https://ansible.colsanitas.com/#/login> sobre los 3 servidores que contienen el Automation Controller.

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

También se desplegaron tres (3) servidores del automation controller para mantener una alta disponibilidad y aumentar la capacidad de la plataforma durante la ejecución de las plantillas de trabajo, se puede acceder al AAP directamente desde los nodos a través de las siguientes URL:

<https://10.160.1.235/#/login>

<https://10.160.1.236/#/login>

<https://10.160.1.237/#/login>

A continuación, se relaciona la información detallada del balanceo de la plataforma:

Nombre	Dirección IP	Observación
ansible.colsanitas.com	172.22.150.81	Balanceo de Consola Web y API

Tabla 1. Información de Balanceo Ansible

4.3. Integración de GIT para la creación de proyectos

Como pudimos ver en la arquitectura ansible almacena sus playbooks y directorios de playbooks desde un repositorio en GIT, este repositorio pertenece a la rama master y lo podemos encontrar en la siguiente URL: https://github.com/keraltyansible/operaciones_ansible

Este es el repositorio de git asignado para el proyecto de ansible, a este repositorio se accede a través de una autenticación gestionado por Azure AD y se encuentra integrado al ansible a través de un token generado por GIT y configurado en el Automation Controller de la siguiente manera:


◀ Volver a Proyectos Detalles Acceso Plantillas de trabajo Notificación Programaciones					
Último estado de la tarea	✔ Correctamente	Nombre	GitHub_operaciones_ ansible	Organización	Default
Tipo de fuente de control	Git	Revisión del control de fuentes	b3fd068 	URL de fuente de control ⓘ	https://github.com/keraltyansible/operaciones_ansible
Credencial de fuente de control	Scm: Github-Jlramos_...	Tiempo de espera de la caché	240 Segundos	Ruta base del proyecto ⓘ	/var/lib/awx/projects
Directorio de playbook ⓘ	_8_github_operaciones_ansible	Creado	31/8/2023, 2:13:48 p. m. por admin	Último modificado	23/10/2023, 4:06:38 p. m. por admin

Figura 2. Configuración de Repositorio GIT de Producción en AAP

Nota: La rama mencionada anteriormente es la rama productiva, también llamada rama master y es donde se almacenan los playbooks que se están usando actualmente, sin embargo también tenemos una segunda rama llamada develop ,

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

allí es donde alojamos los playbooks de prueba o aquellos desarrollados de manera transitoria para un único uso y lo podemos encontrar en la siguiente URL:
https://github.com/keraltyansible/operaciones_ansible/tree/develop


◀ Volver a Proyectos Detalles Acceso Plantillas de trabajo Notificación Programaciones					
Último estado de la tarea	✔ Correctamente	Nombre	GitHub_OpsAAP_br_develop	Descripción	GitHub en rama develop
Organización	Default	Tipo de fuente de control	Git	Revisión del control de fuentes	c2c9118 
URL de fuente de control ⓘ	https://github.com/keraltyansible/operaciones_ansible	Rama de fuente de control ⓘ	develop	Credencial de fuente de control	Scm: Github-jlramos...
Tiempo de espera de la caché	240 Segundos	Ruta base del proyecto ⓘ	/var/lib/awx/projects	Directorio de playbook ⓘ	_9_github_opsaap_br_develop
Creado	6/9/2023, 4:27:44 p. m. por admin	Último modificado	11/10/2023, 11:33:40 a. m. por admin		

Figura 3. Configuración de Repositorio GIT de Desarrollo en AAP

4.4. Integración con Azure AD

De acuerdo con las políticas de keralty, para la gestión de usuarios y contraseñas el servicio de autenticación del Automation Controller está integrado con Azure AD para permitir la autenticación y gestión de usuarios desde el proveedor de identidad de Azure. Para realizar la integración, el cliente creo la aplicación llamada "ANSIBLE AUTOMATION PLATFORM", cuyo ID es "b4ad74ca-8f7d-4310-b49b-536d6734021b". La URL de callback configurada en Azure AD es:
<https://ansible.colsanitas.com/sso/complete/azuread-oauth2/>.

4.5. Proceso Automatizado de Reinicios

4.5.1. Caso de Uso de Reinicios

A continuación, se ilustrará el caso de uso que describe el procedimiento que siguen los operadores del centro y los administradores de sistema para parchar los servidores de las aplicaciones mencionadas en el alcance de este documento.

<div> <div>Versión: 2.0</div> <div>Fecha: 28-11-2024</div> <div>Código: SIG-TI-CKE-PR025</div> </div>	<div> <div>PROCEDIMIENTO PARCHADO DE</div> <div>SERVIDORES DE AVICENA</div> </div>	<div>  </div>
---	--	--

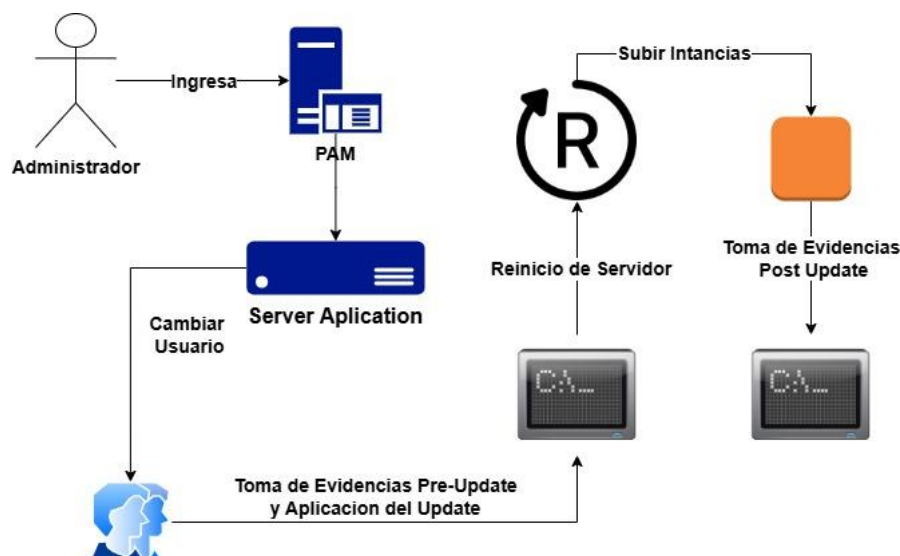


Figura 4. Caso de Uso para Parchado de Servidores Manual

4.5.2. Parchado Automatizado de Servidores Avicena Aplicación, Avicena Reportes y Avicena Web

A continuación, se mostrará un resumen de lo que realiza el playbook para el proceso de parchado de los servidores de Avicena:

- A. Pre- Update:** Toma de Evidencias de paquetes instalados antes de actualizar y las guarda en un servidor central.
- B. Update:** Ejecución del comando **yum -y update** en los servidores objetivo.
- C. Post-Update:** Reinicia servidores y levanta las instancias de las aplicaciones.
- D. Evidencias:** Genera evidencias post-reinicio, guarda las evidencias en un servidor central y ajustar los permisos en la carpeta de evidencias.

El playbook quedo alojado con el nombre **Update_Avicena**, a continuación, se relaciona a detalle el funcionamiento del playbook:

4.5.3. Inventario

Se configuro un inventario construido con los inventarios ya existentes que incluyen todos los servidores de Avicena Aplicación, Avicena Reportes y Avicena Web.

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

Nombre	Update_Avicena	Último estado de la tarea	Correctamente	Tipo	Inventario construido
Organización	Default	Total groups	3	Total hosts	27
Total inventory sources	0	Update cache timeout	0	Inventory sources with failures	0
Verbosity	0				
Inventarios de entrada	avicena avicenareportes avicenaweb srvvksansibledevsrv				

Figura 5. Inventario Construido para el Parchado de los Servidores de Avicena

4.5.4. Encabezado

```
- name: Playbook para Update en Servidores de Avicena Aplicacion, Avicena Reportes y Avicena Web
hosts: all
vars:
  user1: root
  user2: jboss
  user3: jbossweb
```

Figura 6. Encabezado del Playbook

- ❖ **name:** Aquí definimos el propósito que va a tener el playbook
- ❖ **hosts:** Aquí definimos que este playbook se aplique a todos los hosts definidos en el inventario
- ❖ **vars:** Declaramos en 3 variables los usuarios que usaremos para ejecutar los procesos de los reinicios:
- ❖ **user1:** Usuario **root** para ejecutar el parchado (update) y la asignación de permisos
- ❖ **user2:** Usuario **jboss** para subir las instancias de Avicena Reportes
- ❖ **user3:** Usuario **jbossweb** para subir las instancias de Avicena Web

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

4.5.5. Toma de Evidencias Pre-Update

4.5.5.1. Lista de Servidores Objetivo

```
tasks:
  #Tomar Evidencias Pre-Update
  - name: Definir la lista de servidores objetivo
    set_fact:
      target_servers:
        - "10.160.2.60"
        - "10.160.2.61"
        - "10.160.2.74"
        - "10.160.2.75"
        - "10.160.2.76"
        - "10.160.2.77"
        - "10.160.2.78"
        - "10.160.2.79"
        - "10.160.2.80"
        - "10.160.2.81"
        - "10.160.2.82"
        - "10.160.2.83"
        - "10.160.2.84"
        - "10.160.2.85"
        - "10.160.2.86"
        - "10.160.1.21"
        - "10.160.1.35"
        - "10.160.1.36"
        - "10.160.1.37"
        - "10.160.1.38"
        - "10.160.1.22"
        - "10.160.1.39"
        - "10.160.1.40"
        - "10.160.1.41"
        - "10.160.1.42"
        - "10.160.1.43"
```

Figura 7. Listado de Servidores para Tomar Evidencias Pre-Update

- ❖ **set_fact:** Define dinámicamente una lista de servidores donde se tomarán las evidencias pre-update.
- ❖ **target_servers:** Variable donde se agrupa el listado de servidores de donde se tomarán las evidencias pre-update.

<div> Versión: 2.0 </div> <div> Fecha: 28-11-2024 </div> <div> Código: SIG-TI-CKE-PR025 </div>	<div> PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA </div>	<div>  </div>
--	--	--

4.5.5.2. Evidencia Previa desde un Servidor Central

```
- name: Ejecutar el comando desde el servidor 10.160.2.37 hacia cada servidor objetivo
  shell: >
    ssh root@{{ item }} 'rpm -qa' >
    /opt/Evidencias_Parchado/Evidencia_Pre_Update_{{ item | replace('.', '_') }}.txt
  loop: "{{ target_servers }}"
  delegate_to: srvvkansibledevsrv
  when: inventory_hostname in ["srvvkansibledevsrv"]
```

Figura 8. Tarea para la toma de Evidencias Pre-Update

- ❖ **Propósito:** Tomar evidencias de los paquetes instalados en los servidores de Avicena antes de realizar la actividad de parchado.
- ❖ **shell:** El módulo shell se usa para ejecutar comandos sobre los servidores como si se estuviera directamente sobre la consola, en este caso usa SSH para conectarse desde el servidor central (**srvvkansibledevsrv**) a cada uno de los servidores definidos en la variable **target_servers**.
- ❖ **ssh root@{{ item }}:** Se conecta al servidor remoto con el usuario root y usa el parámetro ítem el cual itera el servidor del **loop**.
- ❖ **rpm -qa:** Es el comando que lista todos los paquetes RPM instalados en el servidor remoto.
- ❖ **>:** Redirige la salida del comando al archivo local en el servidor central.
- ❖ **/opt/Evidencias_Parchado/Evidencia_Pre_Update_{{ item | replace('.', '_') }}.txt:** Es la ruta y nombre del archivo donde se guarda la lista de paquetes y usa el filtro **replace('.', '_')** para transformar las direcciones IP (10.160.2.60) en nombres validos para archivos (10_160_2_60).
- ❖ **loop: "{{ target_servers }}"**: Itera sobre todos los servidores listados en la variable **target_servers** y luego para cada servidor ejecuta el comando SSH desde el servidor **srvvkansibledevsrv**.
- ❖ **delegate_to: srvvkansibledevsrv:** La tarea no se ejecuta desde el servidor de inventario actual, en su lugar se delega al servidor **srvvkansibledevsrv** (10.160.2.37).
- ❖ **when: inventory_hostname in ["srvvkansibledevsrv"]:** Hace que la tarea se ejecute únicamente si el servidor actual (**inventory_hostname**) es **srvvkansibledevsrv** y asegura que otras máquinas del inventario no intenten ejecutar esta tarea.

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

4.5.6. Aplicación de Parchado Sobre los Servidores de Avicena

```
#Tarea para aplicar el Update sobre los servidores de Avicena
- name: Aplicacion de Update en Servidores Avicena
  shell: yum -y update
  become: yes
  become_user: "{{ user1 }}"
  when: inventory_hostname in ["10.160.2.60", "10.160.2.61", "10.160.2.74", "10.160.2.75", "10.160.2.76", "10.160.2.77", "10.160.2.78", "10.160.2.79", "10.160.2.80", "10.160.2.81", "10.160.2.82", "10.160.2.83", "10.160.2.84", "10.160.2.85", "10.160.2.86", "10.160.1.21", "10.160.1.35", "10.160.1.36", "10.160.1.37", "10.160.1.38", "10.160.1.39", "10.160.1.40", "10.160.1.41", "10.160.1.42", "10.160.1.43", "10.160.1.22"]
```

Figura 9. Aplicación del Parchado (Update) sobre los Servidores

- ❖ **Propósito:** Realizar la actualización del sistema operativo en un conjunto específico de servidores.
- ❖ **shell: yum -y update:** El módulo shell se conecta mediante SSH a todos los servidores y ejecuta el comando **yum -y update** el cual descarga e instala las versiones más recientes de los paquetes instalados en los servidores sin solicitar confirmación del usuario dado el parámetro **-y**.
- ❖ **become_user: "{{ user1 }}"**: Especifica que el comando se ejecuta como el usuario **root** definido en la variable **user1**
- ❖ **when: inventory_hostname in [...]**: La tarea se ejecuta solo en los servidores cuya dirección IP esta listada explícitamente.

4.5.7. Reinicio del Primer Grupo de Servidores

```
#Funcion para Reiniciar el Primer Bloque de los Servidores de Avicena
- name: Reinicio del Primer Bloque de Servidores Avicena
  reboot:
    reboot_timeout: 300
  become: yes
  become_user: "{{ user1 }}"
  when: inventory_hostname in ["10.160.2.60", "10.160.2.61", "10.160.2.74", "10.160.2.75", "10.160.2.76", "10.160.2.77", "10.160.2.78", "10.160.2.79", "10.160.1.21", "10.160.1.35", "10.160.1.36", "10.160.1.39", "10.160.1.40", "10.160.1.41"]
#Tarea para esperar 2 minutos para subir las instancias de los servidores reiniciados
- name: Esperar 2 minutos para subir las instancias
  pause:
    minutes: 2
```

Figura 10. Tarea para el Reinicio del Primer Bloque de Servidores

- ❖ **Propósito:** Esta tarea se encarga de reiniciar el primer grupo de servidores y espera un tiempo para que se recuperen antes de realizar más acciones.
- ❖ **reboot:** Es el módulo de Ansible que permite reiniciar un servidor.
- ❖ **reboot_timeout: 300:** El parámetro reboot_timeout especifica que el tiempo máximo en segundos (300 segundos = 5 minutos) es el que Ansible esperara para que el servidor se reinicie y esté disponible nuevamente, antes de que considere que la tarea ha fallado.
- ❖ **become_user: "{{ user1 }}"**: Especifica que el comando se ejecuta como el usuario **root** definido en la variable **user1**
- ❖ **when: inventory_hostname in [...]**: La tarea se ejecuta solo en los servidores cuya dirección IP esta listada explícitamente.

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

- ❖ **pause:** Este módulo detiene la ejecución del playbook durante el tiempo especificado, lo que permite asegurar que los servidores tengan suficiente tiempo para reiniciarse correctamente y estar operativos antes de realizar más acciones.
- ❖ **minutes: 2:** Esto indica que el playbook pausara la ejecución durante 2 minutos.

4.5.8. Subir Instancias del Primer Bloque de Servidores Avicena Reiniciados

```
#Tarea para Subir Instancias del Primer Bloque de los Servidores de Avicena Reportes
- name: Subir Instancias de Avicena Reportes
  shell: sh /opt/jboss/EAP-7.2.0/subir/subir_servicios_reportes_avicena.sh
  become: yes
  become_user: "{{ user2 }}"
  when: inventory_hostname in ["10.160.1.35", "10.160.1.36"]
#Tarea para Subir Instancias del Primer Bloque de los Servidores de Avicena Web
- name: Subir Instancias de Avicena Web
  shell: sh /opt/subir_servicios/subir_servicios_Avicenaweb.sh
  become: yes
  become_user: "{{ user3 }}"
  when: inventory_hostname in ["10.160.1.39", "10.160.1.40", "10.160.1.41"]
```

Figura 11. Tareas para Subir Instancias del Primer Bloque de Servidores Reiniciados de Avicena Reportes y Avicena Web

```
#Tarea para Subir Instancias del Segundo Bloque de los Servidores de Avicena Reportes
- name: Subir Instancias de Avicena Reportes
  shell: sh /opt/jboss/EAP-7.2.0/subir/subir_servicios_reportes_avicena.sh
  become: yes
  become_user: "{{ user2 }}"
  when: inventory_hostname in ["10.160.1.37", "10.160.1.38"]
#Tarea para Subir Instancias del Segundo Bloque de los Servidores de Avicena Web
- name: Subir Instancias de Avicena Web
  shell: sh /opt/subir_servicios/subir_servicios_Avicenaweb.sh
  become: yes
  become_user: "{{ user3 }}"
  when: inventory_hostname in ["10.160.1.22", "10.160.1.42", "10.160.1.43"]
```

Figura 12. Tareas para Subir Instancias del Segundo Bloque de Servidores Reiniciados de Avicena Reportes y Avicena Web

- ❖ **Propósito:** Subir las instancias de Avicena Reportes y Avicena Web del primer bloque de servidores reiniciados.
- ❖ **shell:** Usando el modulo shell se ejecutan los scripts **subir_reportes_avicena.sh** y **subir_servicios_Avicenaweb.sh** alojados en las rutas relacionadas en la imagen.
- ❖ **become_user: "{{ user2 }}"**: Especifica que el comando se ejecuta como el usuario **jboss** definido en la variable **user2**
- ❖ **become_user: "{{ user3 }}"**: Especifica que el comando se ejecuta como el usuario **jbossweb** definido en la variable **user3**

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

- ❖ **when: inventory_hostname in [...]:** La tarea se ejecuta solo en los servidores cuya dirección IP esta listada explícitamente.

4.5.9. Reinicio del Segundo Bloque de Servidores

```
#Tarea para esperar 2 minutos antes de ejecutar la siguiente tarea
- name: Esperar 2 minutos para subir las instancias
  pause:
    minutes: 2
#Funcion para Reiniciar la Segunda Mitad de los servidores
- name: Reinicio del Segundo Bloque de Servidores Avicena
  reboot:
    reboot_timeout: 300
    become: yes
    become_user: "{{ user1 }}"
    when: inventory_hostname in ["10.160.2.80", "10.160.2.81", "10.160.2.82", "10.160.2.83", "10.160.2.84", "10.160.2.85", "10.160.2.86", "10.160.1.37", "10.160.1.38", "10.160.1.42", "10.160.1.43", "10.160.1.22"]
#Tarea para esperar 2 minutos para subir las instancias de los servidores reiniciados
- name: Esperar 2 minutos para subir las instancias
  pause:
    minutes: 2
```

Figura 13. Tarea para el Reinicio del Segundo Bloque de Servidores

- ❖ **Propósito:** Asegurar que los servidores se reinicien correctamente y estén listos antes de que se realicen operaciones adicionales.
- ❖ **reboot:** Es el módulo de Ansible que permite reiniciar un servidor.
- ❖ **reboot_timeout: 300:** El parámetro reboot_timeout especifica que el tiempo máximo en segundos (300 segundos = 5 minutos) es el que Ansible esperara para que el servidor se reinicie y esté disponible nuevamente, antes de que considere que la tarea ha fallado.
- ❖ **become_user: "{{ user1 }}"**: Especifica que el comando se ejecuta como el usuario **root** definido en la variable **user1**
- ❖ **pause:** Este módulo detiene la ejecución del playbook durante el tiempo especificado, lo que permite asegurar que los servidores tengan suficiente tiempo para reiniciarse correctamente y estar operativos antes de realizar más acciones.
- ❖ **minutes: 2:** Esto indica que el playbook pausara la ejecución durante 2 minutos.

4.5.10. Toma de Evidencia Post-Update

```
##### Evidencias Post-Reinicio #####
- name: Ejecutar el comando desde el servidor 10.160.2.37 hacia cada servidor objetivo
  shell: >
    ssh root@{{ item }} 'cat /var/log/dnf.log* |grep "$(date +%Y-%m-%d)" |grep -i upgraded' >
    /opt/Evidencias_Parchado/Evidencia_Post_Update_{{ item | replace('.', '_') }}.txt
  loop: "{{ target_servers }}"
  delegate_to: srsvkansibledevsrv
  when: inventory_hostname in ["srsvkansibledevsrv"]
#Asinar Permisos de Lectura, Escritura y Ejecucion para la Carpeta de Evidencias
- name: Asignacion de Permisos a Carpeta de Evidencias
  shell: chmod -R o+rwX /opt/Evidencias_Parchado
  become: yes
  become_user: "{{ user1 }}"
  when: inventory_hostname in ["srsvkansibledevsrv"]
```

Figura 14. Tareas para Toma de Evidencias Pos-Update

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

- ❖ **Propósito:** Documentar los paquetes instalados en cada servidor después del reinicio y centralizar esta información en el servidor (**srvvkansibledevsrv**) y asegurar que los permisos para acceder a las evidencias sean adecuados.
- ❖ **ssh root@{{ ítem }}:** Se conecta al servidor remoto con el usuario root y usa el parámetro ítem el cual itera el servidor del **loop**.
- ❖ **cat /var/log/dnf.log*:** Muestra el contenido de los archivos de log **dnf.log**
- ❖ **grep "\$(date '+%Y-%m-%d')":** Filtra solo las líneas de los logs que contienen la fecha actual en el formato YYYY-MMDD. La función **\$(date '+%Y-%m-%d')** se usa para obtener la fecha del día actual.
- ❖ **grep -i upgraded:** Filtra las líneas que contienen la palabra **upgraded** (sin distinguir mayúsculas de minúsculas).
- ❖ **/opt/Evidencias_Parchado/Evidencia_Post_Update_{{ inventory_hostname | replace('.', '_') }}.txt:** Define el nombre y la ubicación del archivo donde se guardara el listado y el nombre del archivo incluye los servidores del **inventory_hostname** con los puntos **(.)** reemplazados por guiones bajos **_**
- ❖ **delegate_to: srvvkansibledevsrv:** Indica que la tarea se delega para ejecutarse en el servidor **srvvkansibledevsrv**, donde se almacenan las evidencias.
- ❖ **shell: chmod -R o+rwX /opt/Evidencias_Parchado:** Usa el comando **chmod** para otorgar permisos de lectura, escritura y ejecución a todos los usuarios sobre la carpeta **/opt/Evidencias_Parchado**
- ❖ **become_user: "{{ user1 }}"**: Especifica que el comando se ejecuta como el usuario **root** definido en la variable **user1**

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

5. CONTROL DE CAMBIOS.

FECHA	CAMBIO	VERSIÓN
28/11/2024	Creación del Documento.	1.0
02/12/2024	Actualización de Toma de Evidencias Post-Update	2.0

Tabla 2. Control de Cambios

Versión: 2.0	PROCEDIMIENTO PARCHADO DE SERVIDORES DE AVICENA	
Fecha: 28-11-2024		
Código: SIG-TI-CKE-PR025		

6. FLUJO DE APROBACIÓN.

ELABORÓ	REVISÓ	APROBÓ
Nombre: Jose Luis Ramos Bernal (Administrador de sistemas I) Área/Proceso: GESTION CORPORATIVA TECNOLOGIA Y PROCESOS-TRANSVERSAL Fecha: 28/11/2024	Nombre: Jairo Arley Zamudio Tovar (CONSULTOR DE PROYECTOS Y AUTOMATIZACION TI) Área/Proceso: GESTION CORPORATIVA TECNOLOGIA Y PROCESOS-TRANSVERSAL Fecha: 28/11/2024	Nombre: Luisa Gineth Castaño Perea (Director(a) de Operaciones y Comunicaciones TI) Área/Proceso: GESTION CORPORATIVA TECNOLOGIA Y PROCESOS-TRANSVERSAL Fecha: 05/12/2024

Tabla 3. Flujo de Aprobación

Cualquier copia impresa de este documento se considera como **COPIA NO CONTROLADA**.