
# Análisis gestión remota Lliurex

Autor: Miguel Ángel Juan López
Diciembre 2022

# Motivación

Durante años los equipos suministrados a los usuarios Lliurex han sido utilizados en los propios centros educativos, donde la administración recae en algún responsable junto con la ayuda para la configuración inicial y/o problemas posteriores de un centro de asistencia informática dedicado.

Con el paso del tiempo se ha ido incrementando la necesidad de mayor movilidad del parque informático, así como una estandarización del puesto de trabajo que simplifique la administración del conjunto de equipos suministrados.

Habitualmente los usuarios administradores o responsables, han tenido la capacidad de realizar instalaciones de software adicionales para particularizar el entorno de trabajo en el aula y adaptarlo a sus necesidades. Para los responsables, no siempre ha sido fácil llegar a acometer los cambios en el sistema instalado, por falta de conocimiento, soltura con la instalación existente o demoras en el soporte recibido, la cual puede resultar complicada si se utilizan servicios de red.

Además de dichas particularizaciones sobre los puestos de trabajo y que múltiples usuarios tienen la capacidad de realizar modificaciones en el sistema, quizás con diferentes métodos de realización o posibles errores, dan pie a una heterogeneidad muy grande en los equipos de puesto de trabajo según su localización o uso.

Para dar solución al problema de mayor movilidad del parque informático, se ha optado por utilizar equipos portátiles que ahorran espacio y son más versátiles.

Para comenzar una estandarización y simplificar el uso, se ha optado por comenzar por dejar de utilizar todos los "servicios de aula" relacionados que venían siendo ofrecidos y utilizados por los puestos de trabajo, tomándose como instalación "base" lo incluido en una distribución para equipos de escritorio que no disponen de red local fija y servicios de red relacionados, además dado que son equipos móviles aplicar los servicios de red que venían siendo utilizados no es viable.

Actualmente y gracias a que está disponible un inicio de sesión unificado perteneciente al sistema CDC, todos los usuarios pueden realizar un inicio de sesión en cualquier equipo, esto es ideal para ser utilizado en nuevos equipos portátiles que carezcan de servicios de red en el aula, no obstante, los usuarios carecen de permisos de administración en la máquina, lo cual fuerza una administración más centralizada a nivel corporativo y más organizada y con un puesto de trabajo más homogeneizado.

Por ello, es necesario un sistema de administración remota para los equipos, que permita realizar modificaciones del sistema remotamente y obtener datos del equipo sin intervención del usuario o un equipo dedicado de soporte técnico que tenga que acudir a la localización del equipo presencialmente.

# Requisitos del sistema

Durante la toma de requisitos del sistema han sido propuestas soluciones existentes en el entorno corporativo que en principio pueden ser utilizadas para realizar una administración remota y unificada de equipos informáticos.

Estas soluciones existentes pueden facilitar a la organización tareas de gestión dado que ya es un software conocido y se podrían disponer de equipos de administración o conocimiento existente para el manejo del servicio que centraliza la administración.
La solución propuesta como candidata es utilizar el servicio Foreman, el cual actúa como interfaz gráfica para múltiples sistemas de administración y gestión de la configuración de forma remota, así como la posibilidad de realizar aprovisionamiento de los propios equipos.

Para el caso de uso deseado no parece ser necesaria la capacidad de aprovisionamiento de equipos, ya que pueden ser distribuidos de forma "plataformada", con una instalación funcional y con el software inicial precargado para poder ser funcional.

La propuesta de uso de Foreman, después de un estudio de funcionamiento y características realizado previamente para comprobar su viabilidad, parece ser válido para satisfacer las motivaciones del análisis, además dado que se presenta el requisito de una pronta puesta en marcha del servicio parece ser la solución más viable, dado que optar por soluciones propias conlleva un retraso que no las hace viables aunque puedan adaptarse de una forma más óptima. Foreman dispone de una interfaz web, un servicio que utiliza tecnologías estándar en la industria y puede simplemente comenzar a implantarse sin periodo de desarrollo.

El sistema presenta el requisito que debe requerir la mínima intervención posible tanto por parte del usuario como por parte de los administradores del servicio, no obstante, se asume un mínimo de administración del servicio para realizar las acciones que deban ser configuradas y que puedan ser aplicadas en los equipos cliente.

El sistema debe ser configurado para que, al menos, sea posible ofrecer las siguientes características:

## Requisitos funcionales

- Se debe poder realizar un seguimiento del estado del equipo, esto principalmente implicaría:
  - Identificar las versiones del sistema operativo
  - Poder mostrar el estado de paquetería instalada
  - Opcionalmente poder conocer los problemas de seguridad que tiene el software instalado.
    A priori y con la información del estudio previo de viabilidad del software propuesto, no puede ser resuelto fácilmente de forma automática, por ello se establece una opcionalidad en este requerimiento.
- Debe de contar con capacidades para realizar cambios en las aplicaciones del equipo, esto implicaría:
  - Poder añadir nuevas aplicaciones al sistema
  - Poder eliminar aplicaciones existentes del sistema
  - Poder actualizar una aplicación a la última versión que tenga disponible
- Los equipos deben poder actualizar el sistema operativo automáticamente, en caso que la distribución del propio sistema operativo así lo permita para poder ser realizada por red.
- Debe permitirse poder lanzar a petición del administrador la ejecución genérica de determinados comandos disponibles en el equipo administrado de forma remota y obtener el resultado de este comando.

## No funcionales

- Los equipos administrados deben poder agruparse de forma que facilite la administración y puedan realizarse acciones a un conjunto de equipos simultáneamente.
- Los equipos capaces de utilizar este sistema de administración deben necesariamente figurar previamente en el inventario de equipos corporativos, no se debe permitir que sea introducido cualquier equipo que no figure en el inventario.
- Es aconsejable que el identificador del equipo en el sistema sea trazable y mantenga cierta relación con los datos que aparecen en el inventario para una fácil localización y evite confusiones o errores, como por ejemplo su uuid.
- Los equipos administrados no permitirán conexiones entrantes dado que la infraestructura de red que utilizarán no lo permitirá.
- Los equipos administrados no necesariamente deben poseer un direccionamiento estático o un nombre de dominio (FQDN) asignado, dado que son equipos móviles, para el usuario final y la gestión de red utilizada puede no utilizar este servicio.
- Los equipos por la forma de validación utilizada, puede que no dispongan de conectividad de red alguna sin previamente haber realizado un inicio de sesión del usuario, esto indica que probablemente no sea posible establecer ningún tipo de comunicación con el equipo sin que el usuario lo esté utilizando o en las primeras etapas de arranque del sistema operativo.
- Es deseable que todas las configuraciones necesarias para que funcione este sistema de administración en el equipo administrado sea posible administrarlas o actualizarlas de forma semejante al software instalado o disponible para la distribución, preferiblemente de forma empaquetada.
- El sistema debe presentar para el equipo administrado la menor intervención posible por parte del usuario, servicio de asistencia o administrador en todas sus etapas.

# Casos de uso

Dados los requerimientos del proyecto se pueden identificar los siguientes casos de uso, que deberán ser desarrollados individualmente detallando cómo será su configuración o programación para poder cumplirse satisfactoriamente:

- Añadir un equipo nuevo al sistema de administración remota
    - Uso de UUID para identificación/validación/registro de equipo en el panel de administración
    - Validar un equipo en el sistema de inventario
    - Ejecución automática para integrar el equipo en el sistema de administración
- Eliminar un equipo existente del sistema de administración remota
- Seguimiento del estado de los equipos administrados
    - Identificación del sistema operativo utilizado
    - Mostrar la paquetería instalada y versiones utilizadas
    - Identificar los equipos que tienen instalado un determinado software o una versión en concreta
- Administración de software
    - Añadir nuevas aplicaciones al sistema
    - Eliminar aplicaciones del sistema
    - Actualizar una aplicación en concreto
    - Actualizar todas las aplicaciones instaladas en el sistema a su última versión disponible

- Administración del sistema operativo
    - Realizar una actualización de todo el sistema operativo a otra release, (si la distribución lo permite)
    - Modificación de ficheros
        - Control sobre el origen de las fuentes del software instalado
    - Ejecución remota de comandos
- Agrupación de los equipos administrados en grupos
    - Crear un grupo
    - Ejecución de acciones sobre el grupo
        - Instalaciones
        - Eliminaciones
        - Actualizaciones
        - Ejecución de comandos
- Trazas de ejecución de las acciones realizadas
- Optimización del sistema y servicio
    - Parámetros operacionales adecuados para un buen funcionamiento

# Solución propuesta

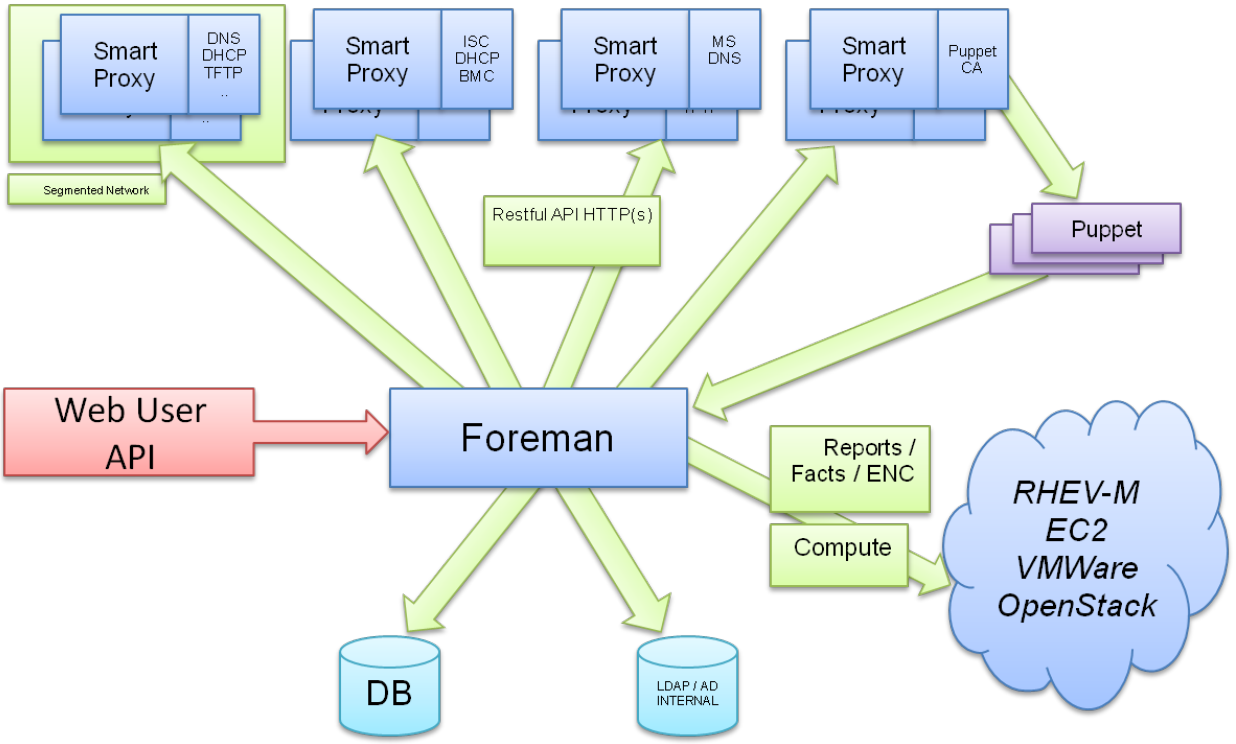
Aquí se detalla la solución que se propone para cumplir todos los requisitos y casos de uso utilizando Foreman como interfaz gráfica y alguna de las soluciones existentes y soportadas por Foreman para realizar una gestión de la configuración eficazmente cumpliendo con todos los objetivos de este proyecto.

Entre las diferentes soluciones para realizar una gestión de la configuración que soporta Foreman el cual utiliza Puppet como herramienta por defecto también es posible elegir plugins para tecnologías como Salt y Ansible.

Para este proyecto se ha decidido utilizar la tecnología Salt, los motivos que dan pie a esta elección enumerados por importancia en la decisión son:

- El departamento de sistemas existente en la organización ya está utilizando en otros entornos Salt como herramienta para el manejo de la gestión de la configuración, lo cual puede ayudar a la implantación y gestión mientras que no se introducen nuevas tecnologías y uniformiza la gestión remota de equipos a nivel de la organización
- Programado con Python, esto permite al equipo de desarrollo disponible el cual dispone de experiencia en este lenguaje, crear módulos personalizados o adaptaciones más fácilmente.
- Es utilizado un mecanismo cliente-servidor que permite realizar una mezcla de comunicaciones push/pull que resulta válida para este proyecto, junto con un cliente para la ejecución de órdenes interactivas bastante versátil.
- El lenguaje utilizado para definir las tareas a realizar utiliza una mezcla de lenguaje declarativo e imperativo, lo cual resulta un poco más sencillo cuando son programadas las tareas.

# Funcionamiento general de Foreman




Foreman actúa como una interfaz gráfica web segura que puede mostrar y organizar los diferentes los diferentes ordenadores administrados así como sus detalles, puede realizarse las acciones más habituales desde la propia interfaz una vez configuradas las tareas utilizando tecnología Salt.

Presenta una arquitectura basada en diferentes componentes que permiten tener cliente en línea de comandos que utiliza un API REST HTTPS al igual que la interfaz web de forma que es posible también realizar acciones fácilmente desde herramientas externas.

Mantiene su propia base de datos (PostgreSql) para permitir características como:

- Agrupación de equipos en grupos
- Programación de tareas
- Autenticación de usuarios con diferentes roles para manejo del servicio
- Guardado de registros de ejecución y consulta de ellos
- Guardado de los detalles de características notificados por los equipos, permitiendo filtrados
- Asociación de tareas para ejecutar a equipos o grupos.

Presenta una arquitectura que permite crecer horizontalmente para gestionar de forma particionada los diferentes equipos a través de ejecutar las acciones a través de los denominados "smart-proxy".

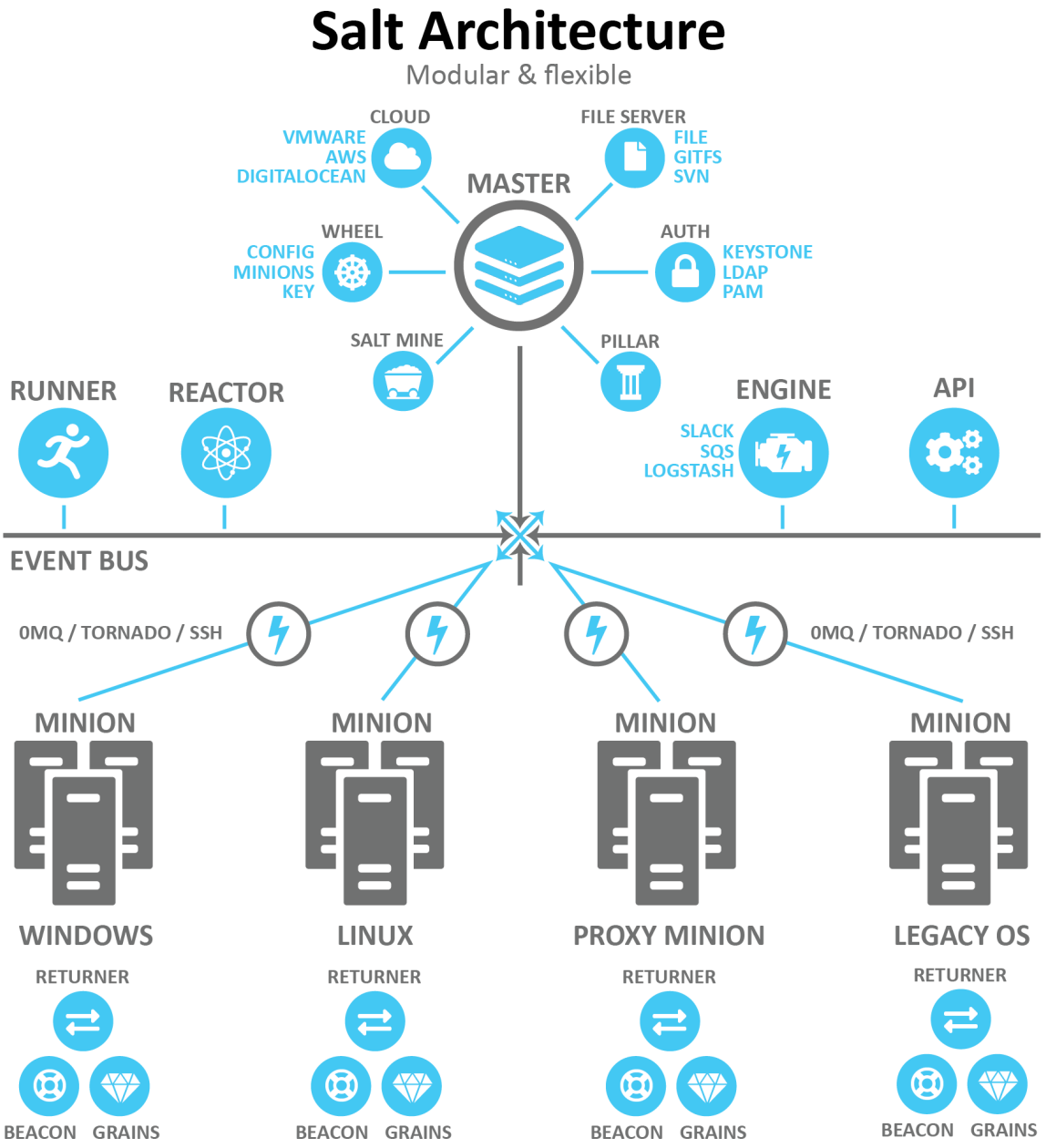
Utiliza un motor de plantillas y variables que pueden ser asociadas a objetos soportando herencia que permite una gran versatilidad cuando se gestiona.

Presenta también características no necesarias para este proyecto, para gestionar servicios que permiten incluso un despliegue completo de equipos totalmente desconfigurados ("bare metal"), puede controlar servicios como:

- TFTP para inicio de equipos por red
- DNS y DHCP para gestión de direccionamiento IP
- HTTP para autoconfiguración desatendida instalando el sistema operativo

Permite diferentes modos para realizar ejecuciones remotas en los ordenadores administrados, aunque para este proyecto donde la red corporativa puede no permitir conexiones entrantes hacia los equipos administrados las tareas deberán realizarse únicamente a través de los mecanismos de comunicación entre servidor y cliente que provee Salt.

## Funcionamiento general de Salt



Salt presenta una arquitectura cliente-servidor con un bus de comunicación (ZeroMQ) en la cual existen agentes que inician dos conexiones seguras persistentes con clave pública contra el servidor y la mantienen establecida para una comunicación bidireccional, en la cual pueden recibir comandos (método "push") o tareas a ejecutar y posteriormente remiten/obtienen (método "pull") datos hacia/desde el servidor, también puede soportar comunicación a través de canales como ssh.

Soporta una arquitectura multi-maestro y jerárquica para alta disponibilidad o mayor capacidad de ofrecer servicio, también arquitecturas basadas en "cloud", y presenta un servicio de API que le permite interactuar con herramientas o servicios externos tanto para operación como autenticación.

Posee un motor de plantillas muy potente utilizado para realizar configuraciones o configurar trabajos, se pueden lanzar ejecuciones tanto en agentes clientes como en el servidor ("runners").

Permite realizar tareas de orquestación entre equipos donde se comparte información ("mine") generada por agentes ("grains") entre agentes, se proporciona información confidencial personalizada para cada agente cliente ("pillar"), se emiten eventos ("beacon") a través de un bus de comunicación y disparadores que reaccionan ("reactor") a los eventos.

Presenta y permite diferentes formas de envío de información tanto al servidor maestro como a multitud de sistemas o servicios("returners"), también soporta múltiples formatos de salida para formatear la información producida.

Para completar las tareas deseadas Salt utiliza las plantillas en formato YAML donde se definen "states" personalizados para conseguir un objetivo, por ejemplo: manipulación de fichero, software, servicios o ejecución de módulos para tareas más específicas, ejecutar comandos de sistema operativo o iniciar la aplicación de otro "estado".

Estas tareas, si son una llamada aislada a un módulo, también se pueden lanzar de forma interactiva a través de la línea de comandos en una terminal del servidor maestro Salt, no obstante no es su forma habitual de trabajo, pues de esta forma los cambios no serían almacenados.

Cuando una tarea es definida en una plantilla o bien se lanza de forma interactiva, generalmente es aplicada mediante un "selector" que permite especificar o filtrar los clientes que aplicarán dicha tarea, esto permite utilizar expresiones regulares y modos más específicos en función de las características ("grains") que el equipo administrado ha calculado.

Aunque las tareas son lanzadas contra uno o varios objetivos a través del selector salt o la definición de la tarea en plantillas, no es la forma recomendada para este proyecto, ya que al ser utilizado en conjunto con Foreman, es mejor realizar este filtrado y lanzamiento de tareas desde la interfaz de Foreman, esto aporta las siguientes ventajas:

- Se queda un registro almacenado con la salida emitida por la tarea en Foreman que es consultable y asociado al equipo que la ha lanzado, de otra forma es más difícil conseguir esa información.
- Desde Foreman durante la definición de lanzamiento de una tarea, también puede definirse un filtrado para ser aplicada en diferentes objetivos de forma selectiva.

# Arquitectura propuesta, componentes y funcionamiento inicial

Para este proyecto será utilizados los siguientes componentes:

- Servicio para DHCP de los clientes ubicado en la red local y puerta de enlace de los clientes hacia internet (existente en la infraestructura actual)
- Servicios en el servidor central de gestión remota disponible en la red local corporativa
    - Servidor Foreman con plugins para su uso conjunto con tecnología Salt.
        - foreman: Realiza las tareas para el acceso web por parte del administrador del sistema
        - foreman-proxy: Realiza comunicación con el servidor Salt (o los clientes por SSH, pero no será utilizada esta tecnología en principio para este proyecto por restricciones de la red, que puede no permitir conexiones directas desde el proxy a los equipos administrados)
        - postgresql: Utilizada para el almacenamiento de información por parte de Foreman.
        - apache: Utilizado por Foreman para proporcionar el servicio web.
        - redis: Utilizado por Foreman para almacenar datos "key-value".
        - dynflow-sidekiq: Utilizado por Foreman para lanzamiento, proceso y orquestación de trabajos.

          Nota: Se instalan servicios Puppet junto con la instalación de Foreman, pero no serán utilizados.
    - Servidor Salt al que se se conectan los clientes administrables para envío y recepción de tareas o datos
        - salt-master: Servicio de servidor maestro Salt
        - salt-api: Para la interconexión del servidor maestro Salt con otros sistemas

          Nota: No parecen ser necesarios los servicios salt-syndic (actuaría como un proxy que realiza tareas como un maestro-intermedio pero contacta contra otro maestro de jerarquía superior) y salt-cloud (funcionamiento en servicios cloud)
    - Servidor para validación de registro contra el inventario corporativo
- Servicios en los clientes administrables remotamente
    - Servicio para registro automático del equipo
    - Agente Salt ("minion") para recepcion y envio de información

A pesar que servicio inicialmente permite utilizar servicios necesarios de forma distribuida en diferentes máquinas o configuraciones multi-proxy o multi-maestro, dado que Salt es una tecnología calificada como no demasiado exigente en recursos, es una implantación inicial y no se dispone información fiable sobre la carga real que va a soportar el sistema en el entorno y uso final, se decide realizar una instalación más sencilla y centralizando todos los servicios en una única máquina con la finalidad de economizar recursos de la organización, facilitar la instalación e implantación, evaluar el rendimiento y manejo.

Partiendo de una instalación inicialmente validada y configurada de los servicios en el servidor centralizado de administración, será necesario como mínimo realizar las siguientes acciones en la herramienta Foreman:

- Establecer contraseñas o posibles usuarios/roles/grupos para el manejo de la herramienta si se desea delegar funciones.
- Ajustar variables globales de funcionamiento y aplicar los valores adecuados
- Ajustar las plantillas utilizadas que según su programación (contenido) pueden utilizar (o nó) las variables para el registro de equipos de forma que sean capaces de auto-configurarse con los ajustes necesarios para funcionar en la infraestructura corporativa.
- Establecer parámetros de "grupos de host", "organización", "ubicación", "sistemas operativos"
- Establecer parámetros para capacidades de ejecución remota, así como el método de funcionamiento de los botones que son presentados en la interfaz web, los cuales no deben utilizar métodos de conexión directos para la realización de las acciones.
- Ajustar los valores iniciales para las plantillas de un primer entorno Salt
    - Ajuste del estado principal "highstate" y del fichero plantilla inicial por defecto para el entorno.
    - Fijar, utilizar y optimizar todos aquellos ajustes que permitan un escalado eficiente de los recursos de la máquina, así como determinar la frecuencia de actualizaciones de deben de tener los equipos para acceso al servidor central.

Una vez instalados y configurados inicialmente los servicios de Foreman y Salt en el servidor centralizado también será instalado el componente que debe desarrollarse para realizar una comprobación previa a añadir un equipo en el sistema Foreman-Salt validando el equipo en el inventario corporativo donde sí resulta válido proporcionará bajo demanda al cliente información para la auto-configuración de los servicios necesarios en el lado del equipo administrado (cliente) y los token necesarios para añadirse satisfactoriamente como un equipo administrado a través de Foreman.

Los equipos cliente (administrados) los cuales parten de una instalación pre-instalada deberán de llevar un servicio instalado/configurado que ha de ser desarrollado y preparado para realizar el auto-registro contra el servicio de validación para el registro de equipos administrables remotamente cuando se cumplan todas las condiciones necesarias para su funcionamiento óptimo.

Es posible proveer las configuraciones necesarias para el cliente mediante un desarrollo empaquetado o bien desde los mecanismos de auto-configuración manualmente, no obstante si los mecanismos de autoconfiguración utilizan una solución empaquetada, posteriormente podría ser actualizada a través de los mecanismos de paquetería habitual, sin necesidad de emplear tecnología Salt, lo cual parece un mecanismo más apropiado, seguro y versátil para este proyecto.

Una vez los servicios están en funcionamiento tanto en el servidor centralizado de gestión remota tanto en el lado del cliente, han sido establecidas relaciones de confianza entre los servicios Salt y periódicamente se realiza contacto de los clientes al servidor maestro para obtener tareas a cumplir y emitir los datos de funcionamiento que será recopilado por el servicio Foreman-Salt.

Los operadores del servicio de gestión remota centralizada en la organización, en su día a día se conectarán validándose al servicio web para la administración de las tareas (Salt), está administración consistirá en asociar determinados estados a determinados hosts, grupos de host. También debían obtener trazas del estado de los equipos o métricas de la ejecución de los trabajos para posiblemente tomar decisiones y confeccionar nuevas acciones para aplicar en los equipos según las políticas de la organización.

La confección de las tareas Salt, deberá realizarse copiando (posiblemente a través de SSH) ficheros de plantillas (ficheros "sls") a la carpeta definida como principal para la instalación, posteriormente estos ficheros plantilla pueden ser detectados desde Foreman para ser asociados a los diferentes objetos.

Aunque pueden ser asociados determinados estados "states" a equipos directamente en Salt, es recomendable no utilizar este mecanismo, dado que la interfaz de gestión Foreman en ningún caso añade/elimina/modifica contenido en esta carpeta principal de la instalación Salt, solo asocia "estados" a equipos a través de la información que los clientes obtienen del "pillar" (información personalizada a cada equipo proporcionada por el servidor Salt), el cual es gestionado por foreman cuando funcionan juntos.

La configuración una tarea a realizar para lograr un objetivo en el equipo administrado no puede ser realizada directamente a través de una llamada realizada desde la interfaz Foreman nombrando el comando a ejecutar, esto es porque foreman-proxy no podrá realizar en principio conexiones entrantes a ningún equipo de escritorio, con lo que durante la auto-configuración no debe ser necesario establecer una clave privada para el acceso a los equipos, aunque puede decidirse implantarla en el sistema por si en algún momento están disponibles estas conexiones.

Los otros mecanismos, exclusivos de Salt para realizar modificaciones en los equipos, deberán desarrollarse conforme la necesidad utilizando las declaraciones y notaciones que Salt requiere en sus ficheros de estado, llamando de forma general a módulos que permiten la ejecución genérica de comandos como a módulos especializados para la tarea en cuestión.

Es muy recomendable que dichos ficheros de estado Salt, los cuales representan la lógica de acciones a realizar en cada máquina administrada y representan en cierta forma la configuración del sistema sean almacenados y versionados en un repositorio al igual que si de código software se tratara, así se facilita una trazabilidad y seguridad en lo que principalmente será el desarrollo del día a día en el sistema de gestión remota.

A pesar de la facilidad que puede ofrecer este sistema para la realización de determinadas acciones de mantenimiento o instalación de software para administrar equipos remotamente, no es un reemplazo de las acciones que pueden ser realizadas durante la instalación de nuevo software empaquetado y versionado, dado que posiblemente muchas acciones puedan ser realizadas a través de ambos mecanismos y cada uno debe trabajar conjuntamente con el otro para conseguir alcanzar los objetivos de la forma más sencilla y efectiva sin saturar de acciones el gestor de administración remota.

Como cualquier servicio con una importancia e impacto tán significativo en la infraestructura y organización, deberá de existir un entorno bien diferenciado y análogo (aunque quizás con menos recursos hardware asignados, dado que no se utilizará tanta carga de sistema) al sistema en producción donde puedan realizarse pruebas pre-producción con la finalidad de minimizar errores que a posteriori sean difíciles de subsanar.

A continuación se presentan los diagramas del sistema para un resumen y mayor entendimiento de la arquitectura y el modo de funcionamiento de los componentes del sistema que han sido planificados y nombrados anteriormente.

También se presentan los diagramas mostrando las acciones que deberían realizar los diferentes componentes que han de ser desarrollados para poder poner en funcionamiento el sistema, el servicio de registro en el servidor centralizado de gestión remota y el agente de registro del cliente.

Diagrama general del sistema donde se muestran los diferentes componentes de la infraestructura y los mecanismos que son realizados durante la inicialización de un equipo administrado:

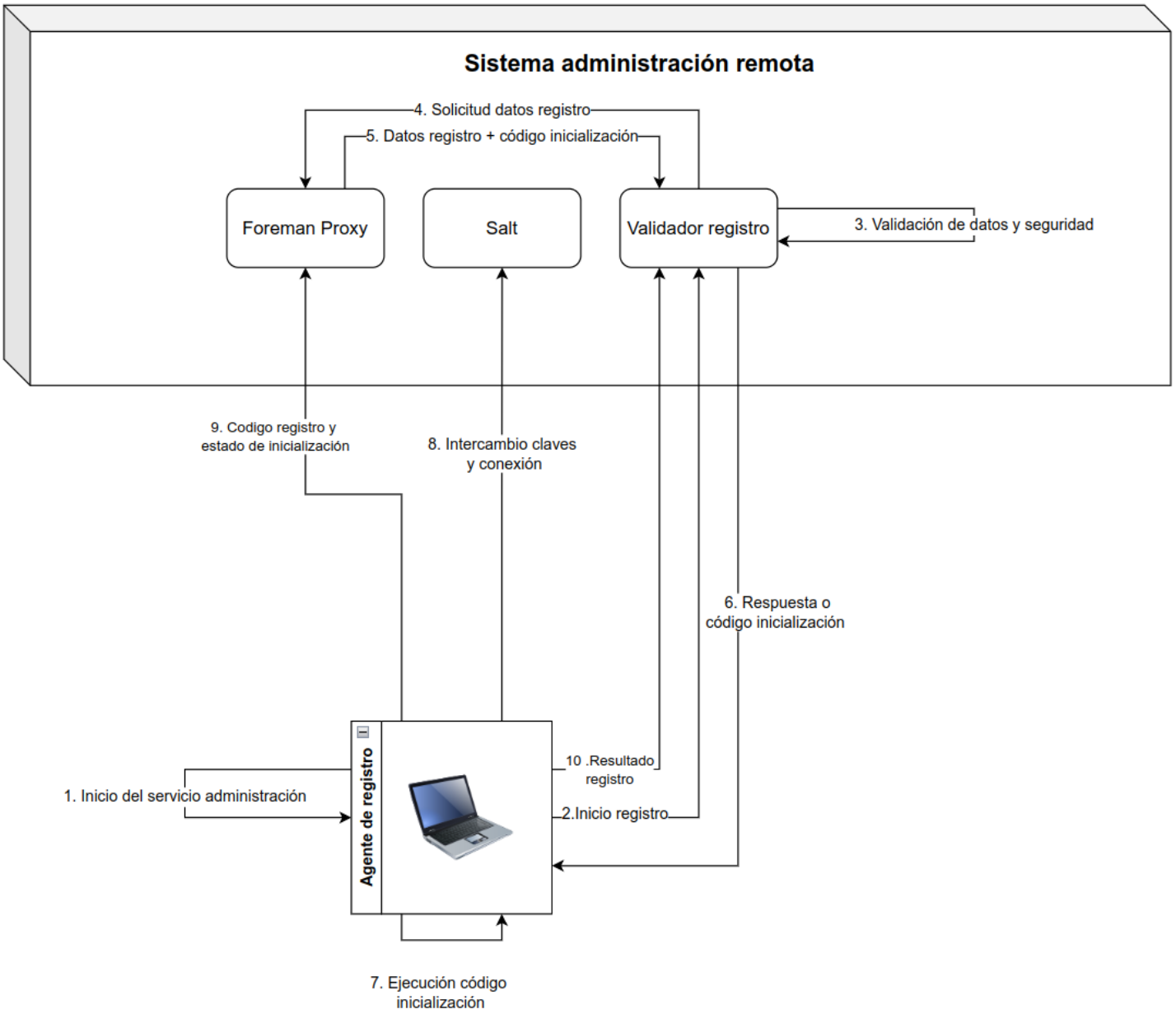


Diagrama del servicio para realizar una validación previa y realizar el registro de un equipo en el sistema Foreman:

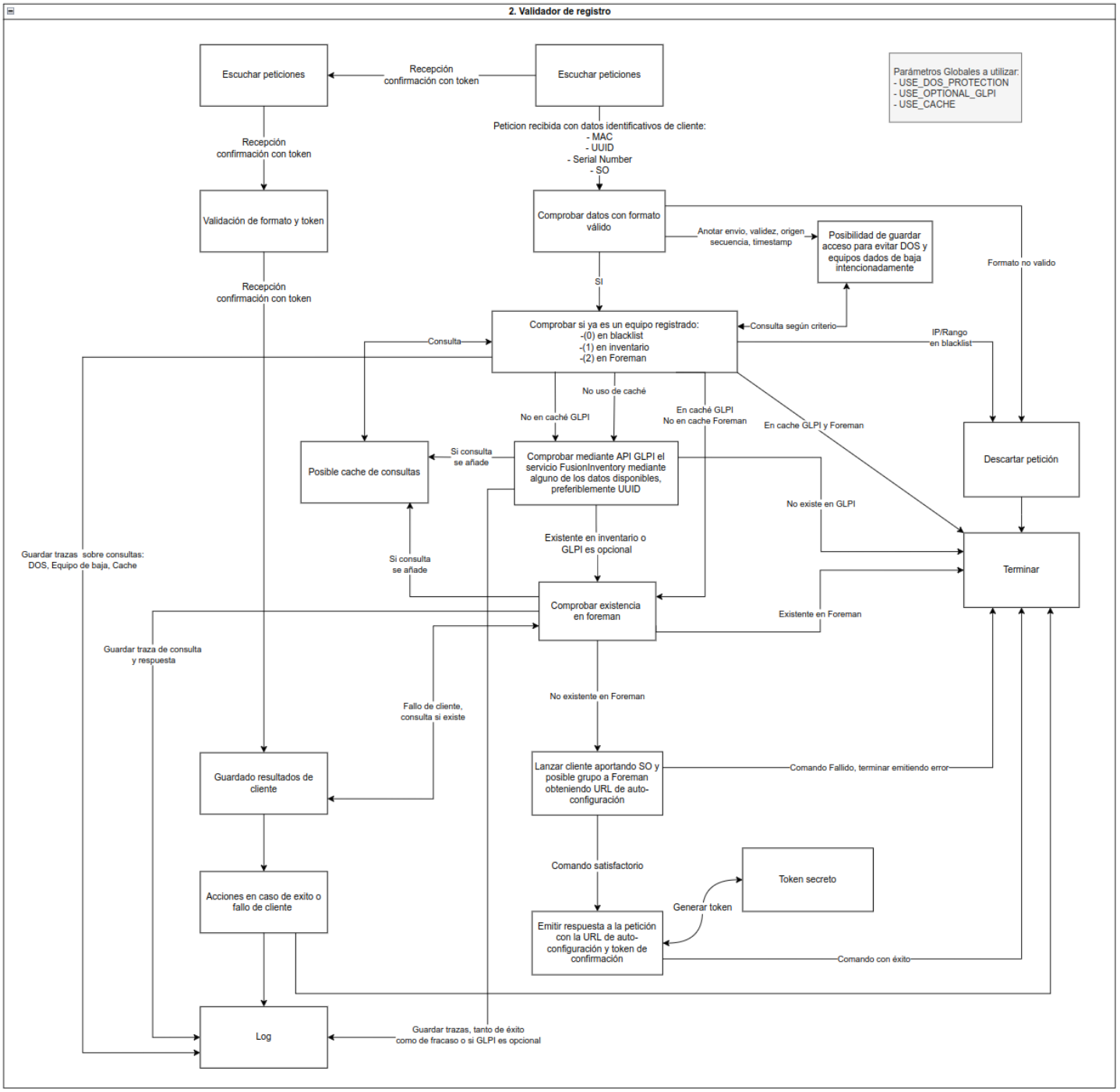
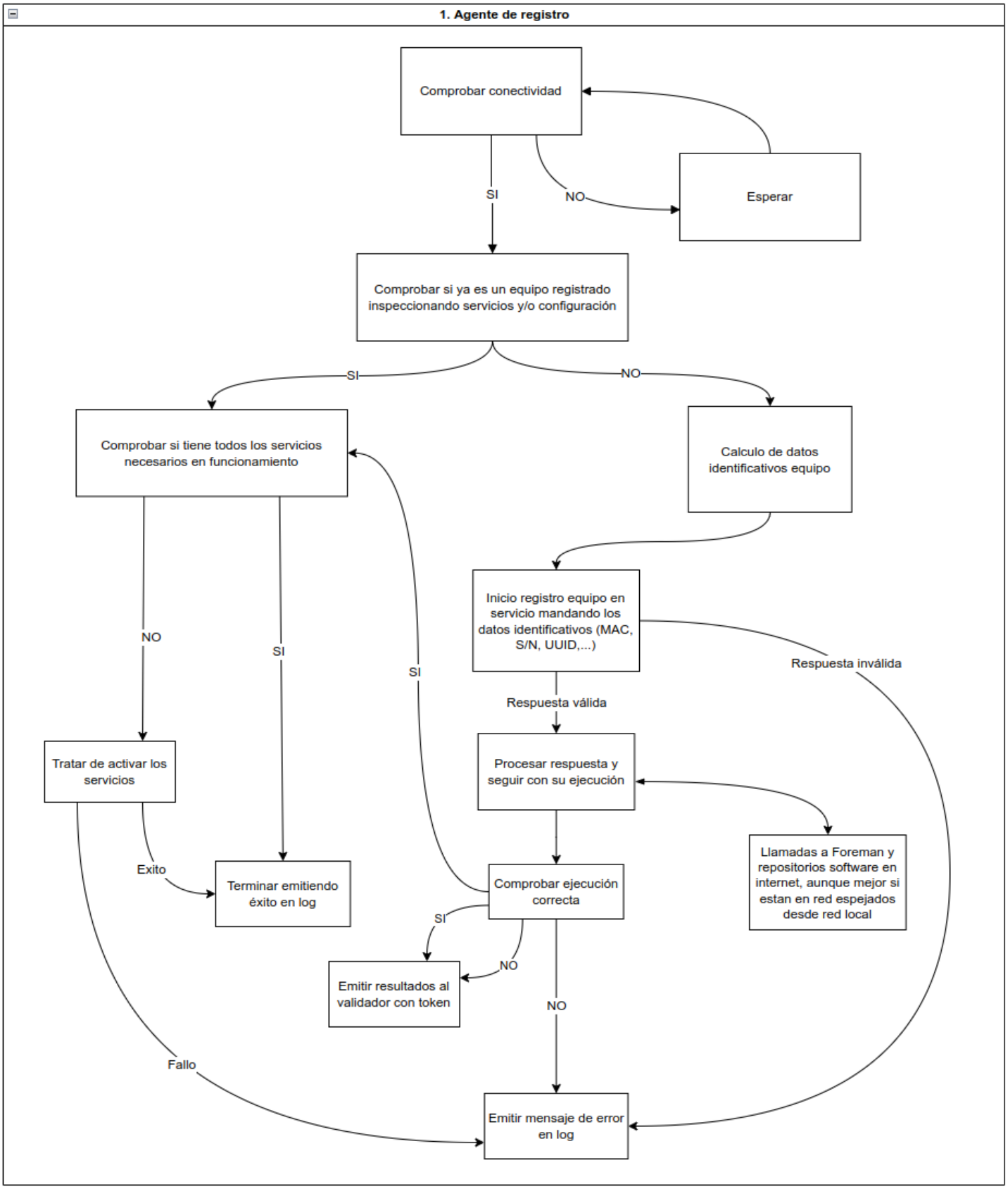


Diagrama del agente de registro que debe ejecutarse en el equipo administrable:



# Flujo de trabajo

En esta sección se comentará de la forma más práctica posible como realizar algunos de los casos de uso planteados en el proyecto o dando una explicación más en detalle de las peculiaridades a realizar para la implementación satisfactoria.

Se recuerda al lector que para lograr un objetivo (administración sw, modificaciones ficheros, manejo de servicios), esta tecnología permite diferentes opciones para completarlo, cada una con sus pros y sus contras aunque igualmente todas válidas, por ello se comentarán diferentes métodos para cada caso de uso, a modo de resumen las diferentes formas de lograr un objetivo pueden ser:

- Unas acciones que es definidas a través de plantillas salt (más aconsejable)
- Unas acciones lanzadas de forma interactiva a través de línea de comandos Salt (menos aconsejable)

Dentro de las acciones lanzadas de una u otra forma, se puede resolver una tarea de forma diferente, según si es utilizado en mayor o menor medida la tecnología Salt, las herramientas del sistema operativo o la programación de scripts disponibles en el equipo objetivo, algunos métodos podrían ser:

- Aplicando diferentes estados para realizar la acción de forma conjunta
- Aplicando un módulo Salt que realiza unas acciones para lograr un objetivo
- Aplicando comandos del sistema operativo a través del módulo para ejecutar.
- Aplicando un lanzamiento de un script en el objetivo a través del módulo para ejecutar.
- Obteniendo información desde un repositorio de internet y aplicando
- Obteniendo información desde el repositorio Salt en el maestro y aplicando

## Instalación del servidor Foreman Salt

Se omite de este documento la documentación sobre instalación de los servicios Salt y Foreman dado que es una petición que debe realizarse al departamento de sistemas y ha de ser realizada por su personal.

Se incluye en el apartado posterior sobre "requisitos software" las características con las que ha de ser configurado el servidor Foreman para poder realizar la petición.

En este documento se presupone una instalación completa y configurada del servicio Foreman junto a Salt en la máquina dedicada a administración remota de equipos en la red local.

Los equipos cliente deben de disponer de conectividad con esta máquina para administración de equipos a través de la red local.

## Añadir equipo al sistema de administración remota

Proceso que debe ser realizado desde el cliente a través del agente de registro complementado por el servicio de registro.
En caso que el equipo hubiera sido dado de baja, debe eliminarse previamente de la lista gris para permitir el auto registro.

### Validación del equipo

El servicio cliente debe obtener datos propios del equipo para remitirlos y que puedan ser validados como un equipo que puede ser administrado remotamente con datos existentes en el inventario, para este proyecto y la validación en el inventario serán necesarios (por orden de importancia):

- UUID: Identificador único del equipo, obtenido mediante el comando:
  - `# dmidecode -s system-uuid`
- SN: Número de serie de la placa base, obtenido mediante el comando:
  - `# dmidecode -s baseboard-serial-number`
  - `# dmidecode -s system-serial-number`
- MAC: Dirección hardware de la tarjeta de red que comunica con el servidor de gestión
  - `$ ip link show $(ip route get <IP|sed -nr 's|.* dev (\S+) .*|\1|p')|sed -nr 's|.*link/ether\s([0-9A-Fa-f:]+)\s.*$|\1|p'|head -1`
- SO: Nombre y versión del sistema operativo, necesario para realizar un registro en Foreman
  - `$ lsb_release -sir | xargs`
    Nota: la identificación para el sistema Salt no contiene la versión menor del sistema operativo, a diferencia de un sistema que utiliza Puppet

### Foreman con línea de comandos

Para realizar las acciones de registro de un equipo de forma automatizada, se debe utilizar la herramienta en línea de comandos "hammer" mediante la cual es posible realizar las acciones como si un administrador realizara las acciones manualmente a través de la interfaz web que Foreman presenta.

Para realizar modificaciones en Foreman a través de la herramienta en línea de comandos, al igual que a través de la interfaz web, el usuario debe estar registrado en Foreman, debe ser validado correctamente y disponer de los permisos necesarios. Por ello cualquier proceso que requiera realizar modificaciones a través de la línea de comandos de forma automatizada debe de poder conocer datos previamente y utilizarlos en sus posteriores ejecuciones. Algunos datos como usuario, password, pueden estar fijados como variables en el automatismo, no obstante, otros datos deben ser consultados desde la base de datos interna de Foreman para conocer el valor que internamente utiliza.

Si es utilizado un usuario administrador, como se ha utilizado en el entorno de pruebas, puede obtenerse posteriormente a la instalación a través de los siguientes comandos.

- Cálculo del nombre de usuario administrador:

  ```
  # sed -nr 's/\s*initial_admin_username:\s*(\w+*)/\1/p'
  /etc/foreman-installer/scenarios.d/foreman-answers.yaml

  El ejemplo devuelve: "admin"
  ```

- Cálculo de la contraseña del usuario administrador:

  ```
  # sed -nr 's/\s*initial_admin_password:\s*(\w+*)/\1/p'
  /etc/foreman-installer/scenarios.d/foreman-answers.yaml
  ```

  El ejemplo devuelve: "3Ajx6eyfZNZa3kFG"

- Obtener el identificador interno que utiliza Foreman para un determinado nombre de "sistema operativo" (enviado por el cliente) y poder conocer si existe previamente o no y poder realizar un cambio en caso de ser necesario

```
# hammer -u admin -p 3Ajx6eyfZNZa3kFG --no-headers --output csv os list|nl -s',' -n
rz|sed -r 's/^0+//'|awk -F',' '{if($3=="Debian 11.5"){print $1}}'
```

o bien si se incluyen las anteriores llamadas para conocer el usuario y password:

```
# hammer -u $(sed -nr 's/\s*initial_admin_username:\s*(\w+*)/\1/p'
/etc/foreman-installer/scenarios.d/foreman-answers.yaml
admin) -p $(sed -nr 's/\s*initial_admin_password:\s*(\w+*)/\1/p'
/etc/foreman-installer/scenarios.d/foreman-answers.yaml) --no-headers --output csv os
list|nl -s',' -n rz|sed -r 's/^0+//'|awk -F',' '{if($3=="Debian 11.5"){print $1}}'
```

- Obtener los grupos de host que existen en el sistema

  ```
  # hammer -u admin -p 3Ajx6eyfZNZa3kFG --no-headers --output csv hostgroup list
  ```

  Respuesta del ejemplo:
  ```
  1,grupo1,grupo1,,production,
  2,grupo1_1,grupo1/grupo1_1,,production,
  ```

- Obtener ubicaciones que existen en el sistema

  ```
  # hammer -u admin -p 3Ajx6eyfZNZa3kFG --no-headers --output csv location list
  ```

  Respuesta del ejemplo:
  ```
  4,Default Location,Default Location,""
  ```

- Obtener las organizaciones que existen en el sistema

  ```
  # hammer -u admin -p 3Ajx6eyfZNZa3kFG --no-headers --output csv organization list
  ```

  Respuesta del ejemplo:
  ```
  3,Default Organization,Default Organization,""
  ```

- Añadir un equipo a Foreman, obteniendo la URL para configuración y registro inicial

```
# hammer -u $(sed -nr 's/\s*initial_admin_username:\s*(\w+*)/\1/p'
/etc/foreman-installer/scenarios.d/foreman-answers.yaml) -p $(sed -nr
's/\s*initial_admin_password:\s*(\w+*)/\1/p'
/etc/foreman-installer/scenarios.d/foreman-answers.yaml) host-registration
generate-command --insecure true --operatingsystem "Debian 11.5" --hostgroup-id 2
--location-id 4 --organization-id 3 --setup-remote-execution true
```

- Comprobar si un equipo está introducido en Foreman

```
# LANG=C hammer -u administrador -p administrador --output=json host info --name
a5812110ec1d9040882977cb49576282 | jq '.Name'
```

### Uso de UUID para identificación del equipo

Para el registro de equipos en Foreman, habitualmente es utilizado el FQDN del equipo, no obstante, en este proyecto se propone utilizar el UUID como identificador del equipo por los siguientes motivos:

- Puede no existir una nomenclatura estandarizada o pueden no ser fiables los FQDN introducidos en el equipo.
- No es un dato que sea significativo o relevante para identificar el equipo en una vista de listado de equipos.
- Al estar también introducido en la base de datos de inventario se puede identificar y localizar más fácilmente en un cruce de datos.

Para conseguir introducir satisfactoriamente utilizando el UUID en el sistema Foreman, es necesario que se compla lo siguiente:

- Las plantillas de inicialización deben estar adaptadas a este cambio, dado que son utilizadas cuando se selecciona un sistema operativo para el host que está siendo registrado, esta plantilla puede ser asignada desde las propiedades del objeto "sistema operativo".
- Es imperativo que no sea asociado el elemento host en Foreman con un "Dominio" a la "Organización/Ubicación/Grupo Host", de otra forma el UUID sería completado con un dominio y serán multiplicados los objetos que aparecen en el panel de listado de hosts en Foreman.
- Es aconsejable que el hostname sea fijado también a este UUID para una fácil identificación consultando el hostname o simplemente abriendo una terminal, en caso de requerir asistencia o peticiones de cambio telemáticas puede ser útil.

### Añadir equipo al sistema de administración remota

Cuando un equipo cliente ha pasado el proceso de validación en el inventario, obtiene una URL con un token de seguridad para comenzar a obtener desde el servidor central Foreman el código shellscript de autoconfiguración, el cual debe ser ejecutado.

Por defecto el código suministrado el cual depende de la URL+token generado para un sistema operativo que tiene asignada la plantilla es configurable desde la interfaz Foreman ("Host→Plantillas de aprovisionamiento"), para un host Linux genérico está nombrada como "Linux host_init_config". Aunque las plantillas aparecen en la interfaz Foreman para ser editadas y gestionadas desde la propia interfaz, durante la instalación de Foreman en un sistema Debian de forma inicial inicialmente están ubicadas en el disco en el siguiente camino `"/usr/share/foreman/app/views/unattended/provisioning_templates" .`

Esta plantilla puede modificarse para ser adaptada a las necesidades del proyecto de múltiples formas dado que es una plantilla de script, es posible en la plantilla utilizar variables Foreman que pueden ser asignadas de forma global, al host, grupo, S.O. o cualquier objeto que conforme la jerarquía de variables aplicadas a un objeto "host", mediante el uso de variables se consigue que un mismo script pueda tener diferente funcionalidad para según qué host o características habilitadas.

Dado que el proyecto va a utilizar mucha uniformidad de equipos en este sentido, se tratará de minimizar el uso de variables, condicionales en las plantillas y ejecutar un código lo más simple posible.

La plantilla inicial requiere estas modificaciones o adaptaciones:

- Interesante el uso de "`set -x`" junto con el existente "`set -e`" para una mayor verbosidad y trazabilidad.
- No se requiere el uso de Puppet como agente en los clientes, puede omitirse cualquier código que instale Puppet o bien fijarlo como una variable (que estará desactivada) permitiendo opcionalidad.
- No se requiere capacidades de ejecución remota vía SSH, puede omitirse cualquier código que realiza una copia de claves RSA para un acceso remoto o bien fijarlo como una variable (que estará desactivada) permitiendo opcionalidad.

- De forma similar a Puppet, debe realizarse una instalación del cliente Salt, esta puede realizarse desde repositorios públicos en internet o bien de forma más recomendable distribuir los paquetes desde la red local de forma espejada, para controlar las versiones utilizadas y disminuir la aparición de errores.

Se puede consultar el contenido de la plantilla utilizada en los los anexos al final del documento.

### Servicios para añadir equipo al sistema de administración remota

Para añadir un equipo al sistema con validación previa, serán necesarios dos componentes uno en el lado del cliente y otro en el lado del servidor junto a Foreman.

Diagrama de acciones a desarrollar en el apartado anterior "arquitectura propuesta".

El servicio a desarrollar "agente de registro" para realizar la introducción del equipo en el sistema Foreman debe tener en cuenta los siguientes factores:

- El servicio debe iniciarse automáticamente en la máquina cliente y realizar toda su operativa sin intervención humana.
- Un equipo puede no tener red cuando el servicio en la máquina es iniciado, la máquina puede tener solo conectividad una vez el usuario ha realizado login, debe esperar y reintentar posteriormente.
- Los cálculos de la información a enviar deben realizarse conforme a lo expuesto en el apartado anterior "validación del equipo"
- Si el equipo ha sido registrado previamente, ya debe de tener los componentes instalados y configurados, el arranque de estos en caso que no haya conectividad seguirá la misma política de reintentos a la espera de red a través de los componentes de Salt.
- Se debe comprobar en la medida de lo posible el estado de los componentes de funcionamiento
- Debe permitir obtener trazas de su ejecución mostrando marcas de tiempo, éxitos y fracasos.

El servicio a desarrollar "validador de registro" para validar los datos emitidos por un cliente previo a la introducción del sistema Foreman debe tener en cuenta los siguientes factores:

- Debe permitir que el sistema de validación contra el inventario pueda estar desactivado temporalmente rebajando el nivel de obligatoriedad del requisito que fuerza a estar un equipo inventariado previamente. Esta característica podrá ser activada o desactivada por el administrador
- Debe validar el correcto formato de los datos de entrada y la correcta ejecución de los procesos internos que requiera ejecutar manteniendo un registro de todas las ejecuciones y su resultado junto a una marca de tiempo.
- Puede opcionalmente presentar una caché de resultados de inventario y de equipos introducidos a Foreman, para evitar consultas redundantes o bien precargar en lote al sistema datos de inventario y no requerir consultas al inventario.
- Puede opcionalmente disponer de un módulo para comprobaciones de seguridad donde en función de repeticiones de cliente, dirección origen, datos enviados… etc… pueda añadir una lista negra de clientes para evitar sobrecargas al sistema, al inventario o a Foreman.
- Debe disponer una lista gris de clientes que intencionadamente han sido dados de baja en Foreman para que no puedan volver a auto registrarse.
- Debe seguirse el procedimiento para utilizar el cliente en línea de comandos Foreman descrito en el apartado anterior "foreman con línea de comandos"

Si no fueran utilizados los servicios de agente/validador de registro y fuera necesario añadir un equipo de forma manual a través de la interfaz Foreman, a través del menú Hosts→Registrar host se puede obtener un formulario donde a partir de unos selectores de Organización, Localización, Grupo de hosts y sistema operativo puede generarse una URL+token de autoconfiguración que deberá obtenerse y ejecutarse desde el cliente para comenzar el proceso de configuración de lo necesario y registrarse en Foreman.

## Eliminar equipo del sistema de administración remota

Para eliminar un equipo del sistema de administración remota el cual previamente ha sido introducido deben realizarse dos acciones

- Introducir el equipo en la lista gris de equipos eliminados de Foreman para evitar un nuevo auto registro por parte del validador de registro. Dado que el software en el lado del cliente trata siempre de auto registrarse, es necesario mantener un listado de estos equipos.
- Localizar el equipo a través de la interfaz Foreman y proceder a eliminarlo, también podría ser resuelto automáticamente a través de la lista gris, no obstante es más sencillo realizarlo manualmente y no sobrecarga de funciones que periódicamente deba ejecutar el validador de registro.

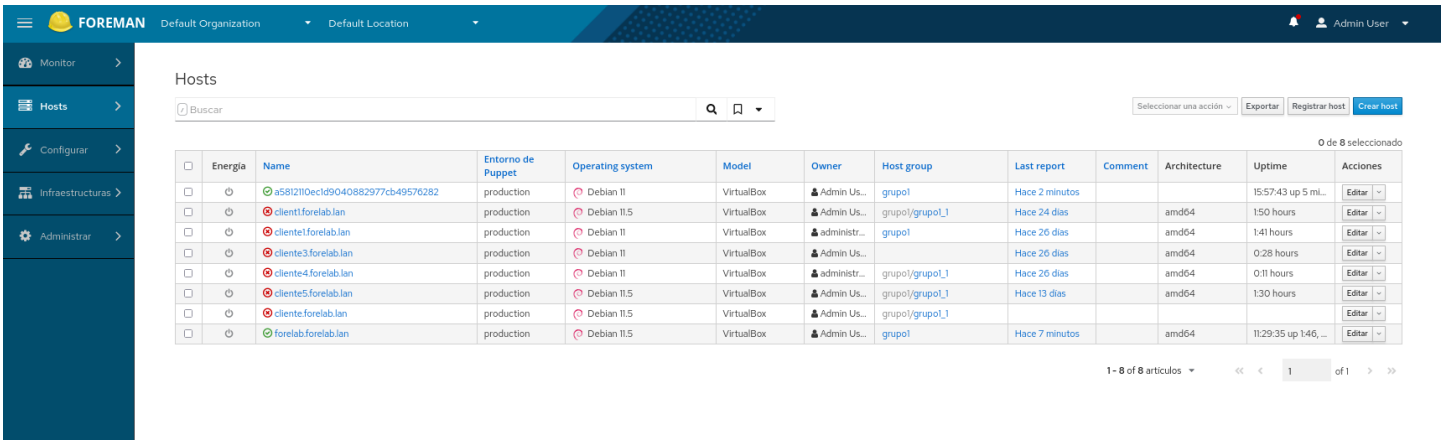
## Seguimiento del estado de las máquinas administradas

Una vez introducido un número de clientes en el sistema ya pueden realizarse las acciones desde la interfaz web que presenta Foreman, también algunas de las acciones pueden ser realizadas desde línea de comandos "hammer" o a través de los comandos que nos ofrece Salt, no obstante aquí se enumeran los casos de uso más habituales y la forma más sencilla de realizarlos.

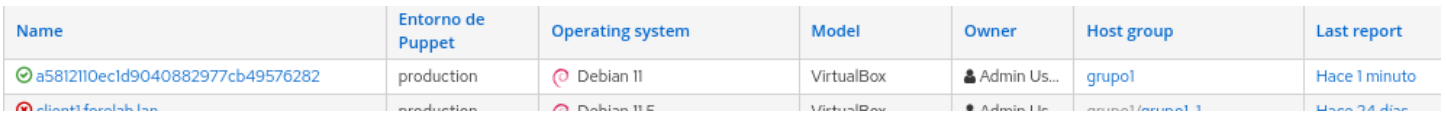
### Vista de equipo y opciones de visualización disponibles en la interfaz foreman

Cuando se desea realizar acciones en una máquina administrada se puede comenzar por acceder a través de la página que muestra el listado de todos los equipos en el sistema.

Para acceder a la página se selecciona desde el menú de la aplicación la opción Hosts→Todos los hosts, aunque también puede accederse directamente a través de la URL `https://<servidor/hosts`



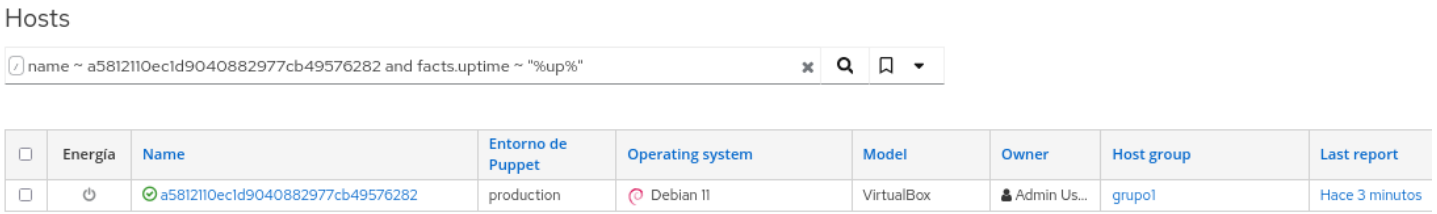

Desde está página, se puede ver a muy grandes rasgos determinados elementos de los equipos, entre la información más relevante que aquí se muestra es: el nombre, la pertenencia a grupos, sistema operativo o el estado de sincronización o tareas realizadas.



En la captura aparecen diferentes host, uno con la nomenclatura basada en UUID que será utilizada, los otros utilizando FQDN, solo hay un único host junto con el propio servidor que está activo (marca verde junto al nombre).

También se pueden realizar filtrados a través del cuadro de búsqueda en función de criterios:

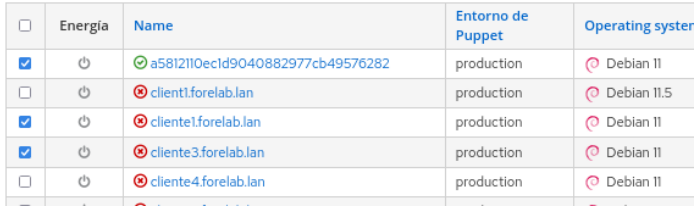
- Nombre de equipo (name = <uuid deseado>)
- Grupo (hostgroup = <grupo deseado>)
- Últimas recepciones de información (last_report > "30 minutes ago")
- Información recolectada (facts.uptime ~ "%up%") ("~" actúa de forma similar al operador "like" de SQL)

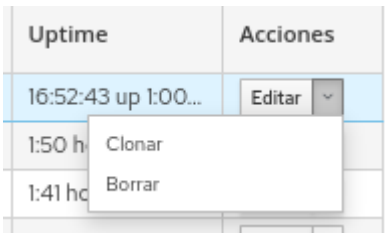


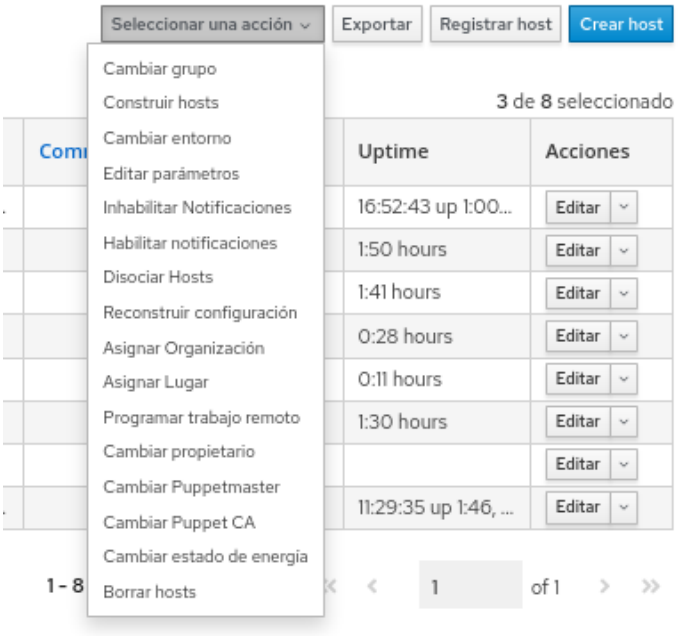

La opción para realizar filtrado de elementos es muy potente, sencilla y versátil, se puede consultar utilizando autocompletado con toda la información almacenada de los equipos que existe en la base de datos que está utilizando internamente Foreman, es por ello que las consultas se pueden asemejar a consultas SQL sencillas desde la propia interfaz donde además de los criterios para ejecutar el filtrado también pueden unirse condiciones lógicas que añadan más de un criterio a la vez.

Desde la vista de listado de equipos también es presentado un selector múltiple ("checkbox") para poder realizar algunas acciones a todos los elementos, aunque para este proyecto quizás las más útiles sean:

- Cambiar de grupo
- Editar parámetros
- Habilitar/Inhabilitar notificaciones
- Asignar Organización/Lugar
- Programar trabajo remoto
- Cambiar propietario
- Borrar hosts



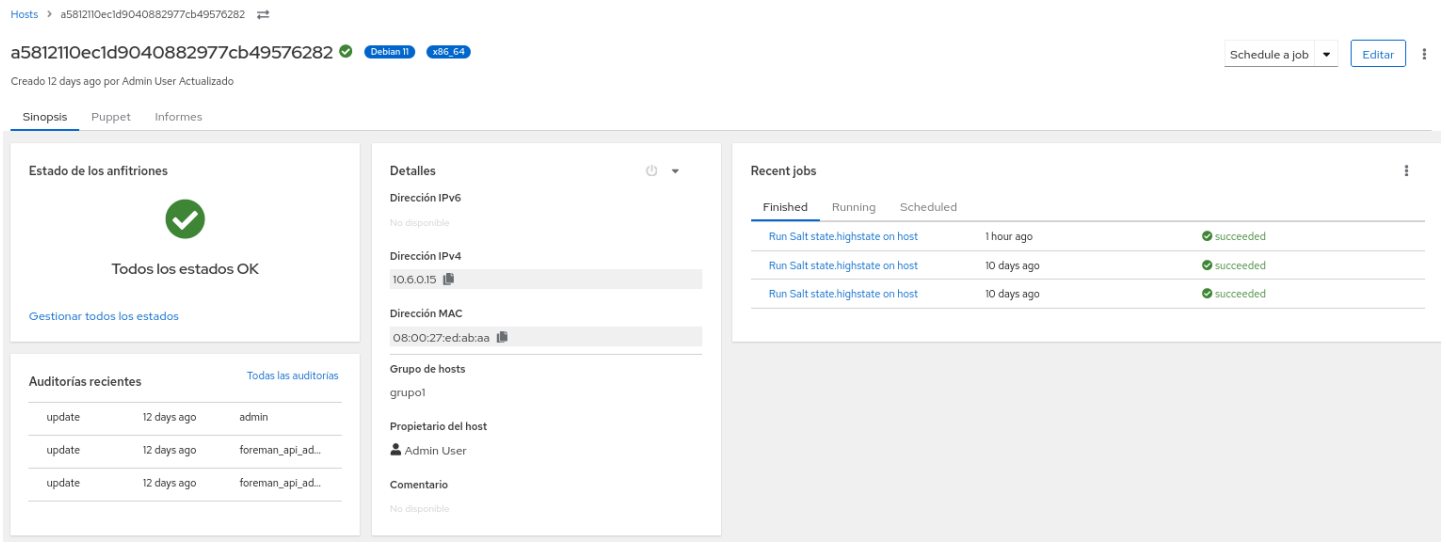



Seleccionando el nombre de un equipo en concreto, se puede visualizar un detalle mucho mayor de esa máquina en concreto


También puede obtenerse rápidamente está información visitando la URL:

`https://<servidor>/hosts/<nombre de host>`

Nota: La vista mostrada en la captura corresponde a la vista "tradicional", actualmente se está migrando la vista de host mostrada con otro diseño más moderno, aunque menos completo, por eso aquí se ha mostrado la versión "tradicional" la cual se puede ir alternando o fijar como vista por defecto utilizada, la vista moderna tiene el siguiente aspecto:




Continuando con la vista "tradicional" se puede visualizar en la parte izquierda muchos datos del equipo directamente así como su estado, en la parte derecha se observa unas gráficas con la tendencia de reportes recibidos así como el tiempo en ejecutar las tareas (Salt en este caso).

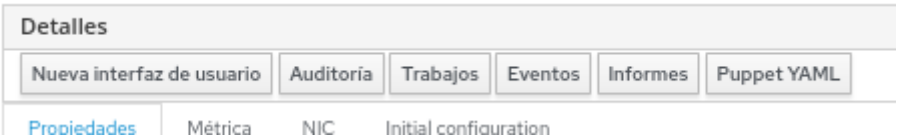
| Propiedades | |
|---|---|
| Estado | Aceptar |
| Construir | Instalado borrar |
| Configuración | Activo borrar |
| Ejecución | La última ejecución se realizó correctamente. borrar |
| Duración de la compilación | menos de 1 minuto |
| Token | N/C |
| Uptime | 16:57:43 up 1:05, 1 user, load average: 0,00, 0,01, 0,00 |
| UUID | |
| Dirección IP | 10.6.0.15 |
| Dirección MAC | 08:00:27:ed:ab:aa |
| Entorno de Puppet | production |
| Arquitectura | x86_64 |
| Sistema operativo | Debian 11 |
| Cargador de PXE | |
| Grupo de hosts | grupo1 |
| Tiempo de arranque | No informado |
| Ubicación | Default Location |
| Organización | Default Organization |
| Propietario | Admin User |

| Estado de informe | Acciones |
|---|---|
| applied | 958 |
| restarted | 0 |
| failed | 0 |
| failed_restarts | 0 |
| skipped | 0 |
| pending | 0 |

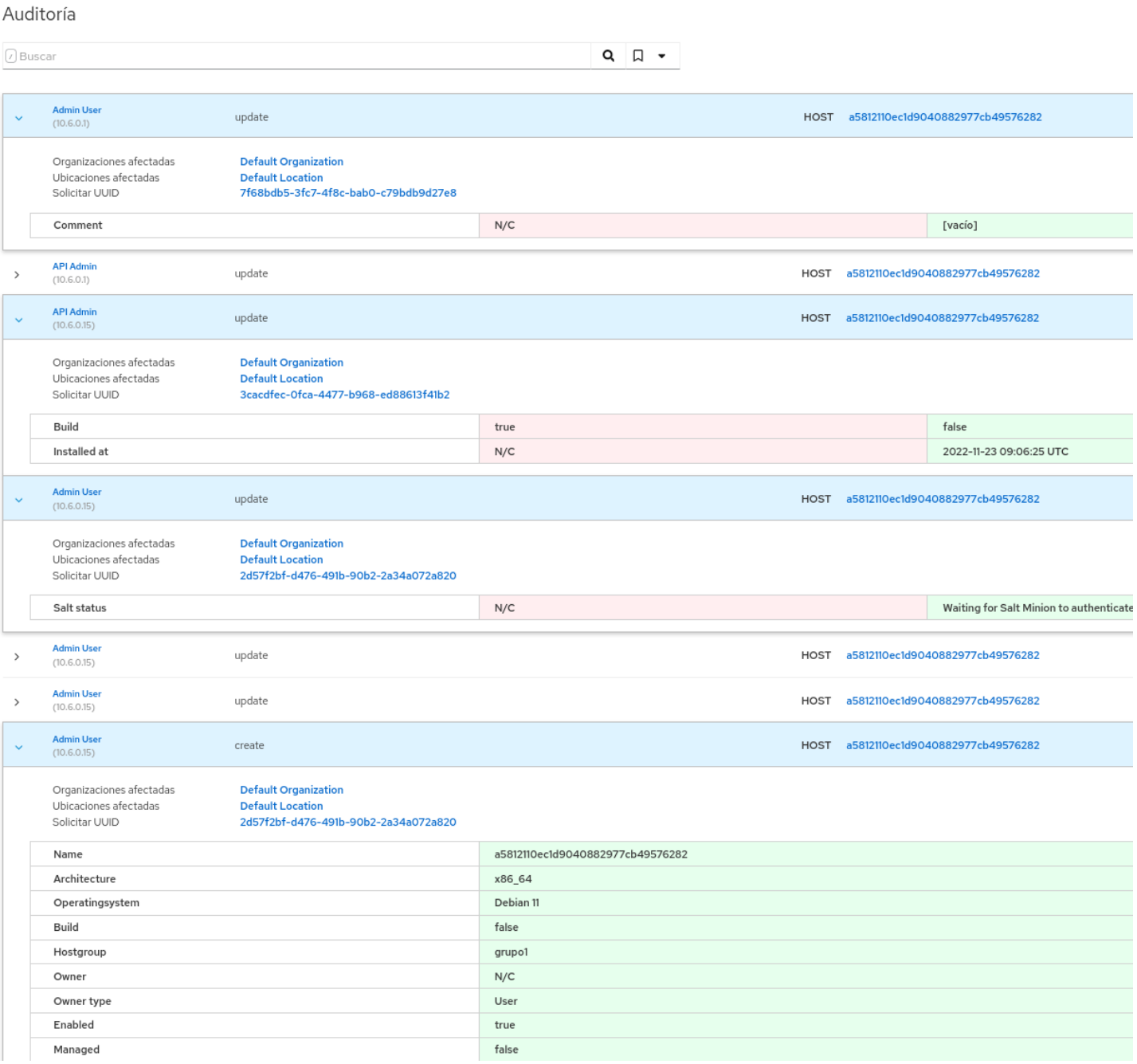
| enp0s3 | |
|---|---|
| Tipo | Interfaz |
| Dirección MAC | 08:00:27:ed:ab:aa |
| MTU | |
| Dirección IPv4 | 10.6.0.15 |
| Dirección IPv6 | |
| FQDN | a5812110ec1d9040882977cb49576282 |

| enp0s8 | |
|---|---|
| Tipo | Interfaz |
| Dirección MAC | 08:00:27:74:1b:c0 |
| MTU | |
| Dirección IPv4 | 10.0.3.15 |
| Dirección IPv6 | |
| FQDN | |

Desde la vista de equipo pueden obtenerse más trazas e información relevante a través de un conjunto de botones situados en la parte superior izquierda




- Nueva interfaz de usuario: Alterna entre la vista "tradicional" y la nueva.
- Auditoría: Muestra las últimas acciones realizadas sobre el equipo, puede mostrar la fecha de creación o otros tipos de modificación y que usuario ha realizado la acción.




- Trabajos: Muestra las últimas invocaciones de trabajos que han sido realizadas en el host.

  Nota aclaratoria: Aquí solo son mostrados los trabajos que han sido programados (repetitivamente o no) a través de la interfaz Foreman, es posible realizar una programación de trabajos (como el envío de información a Foreman) a través de la programación de estados en Salt, esta información no es mostrada aquí ya que queda fuera del alcance de esta herramienta, desde Foreman solo es notada la programación de Salt, gracias a la frecuencia de recepción de datos desde el equipo (los datos de las gráficas de tendencia y apartado generado pulsando el botón de "informes")

Invocaciones de trabajo

host=a5812110ec1d9040882977cb49576282

Documentación | Ejecutar trabajo

| Descripción | Search Query | Estado | exitoso | Errores | Pendiente | Total hosts | Iniciar |
|---|---|---|---|---|---|---|---|
| Run salt state.highstate on host | name ^ (a581... | exitoso | 1 | 0 | 0 | 1 | Hace alrededor de 2 horas |
| Run salt state.highstate on host | name ^ (a581... | exitoso | 1 | 0 | 0 | 1 | Hace 10 días |
| Run salt state.highstate on host | name ^ (a581... | exitoso | 1 | 0 | 0 | 1 | Hace 10 días |
| Run salt state.highstate on host | name ^ (a581... | exitoso | 1 | 0 | 0 | 1 | Hace 12 días |
| Run salt state.highstate on host | name ^ (a581... | exitoso | 1 | 0 | 0 | 1 | Hace 12 días |
| Run salt state.highstate on host | name ^ (a581... | exitoso | 1 | 0 | 0 | 1 | Hace 12 días |
| Run salt state.highstate on host | name ^ (a581... | exitoso | 1 | 0 | 0 | 1 | Hace 12 días |
| Run salt state.highstate on host | name ^ (a581... | exitoso | 1 | 0 | 0 | 1 | Hace 12 días |
| Run salt state.highstate on host | name ^ (a581... | exitoso | 1 | 0 | 0 | 1 | Hace 12 días |
| Run dpkg -l | * | Errores | 1 | 7 | 0 | 8 | Hace 12 días |
| Run salt state.highstate on host | name ^ (a581... | exitoso | 1 | 0 | 0 | 1 | Hace 12 días |
| Run salt state.highstate on host | name ^ (a581... | exitoso | 1 | 0 | 0 | 1 | Hace 12 días |
| Run salt state.highstate on host | name ^ (a581... | exitoso | 1 | 0 | 0 | 1 | Hace 12 días |
| Run salt state.highstate on host | name ^ (a581... | exitoso | 1 | 0 | 0 | 1 | Hace 12 días |
| Run salt state.highstate on host | name ^ (a581... | Errores | 0 | 1 | 0 | 1 | Hace 12 días |
| Run dpkg -l | * | Errores | 1 | 7 | 0 | 8 | Hace 12 días |
| Run dpkg -l | * | Errores | 1 | 7 | 0 | 8 | Hace 12 días |

1 - 17 of 17 artículos  1 of 1

- Eventos: (o "hechos"), son datos calculados en la máquina cliente que periódicamente puede recolectar el servidor maestro para almacenarlos, consultarlos, mostrar cambios o tendencia.

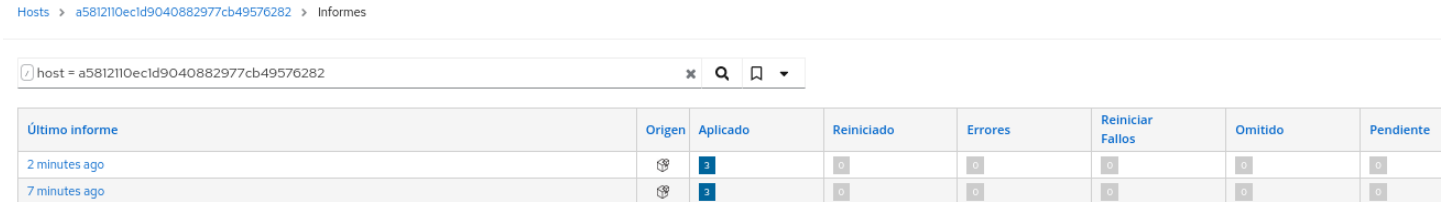
  Son mostrados a través de una tabla con diferentes niveles que permite su navegación mostrando su nombre o valor, también se dispone del cuadro de filtrado para realizar búsquedas o para obtener datos rápidamente.

Valores de eventos › a5812110ec1d9040882977cb49576282

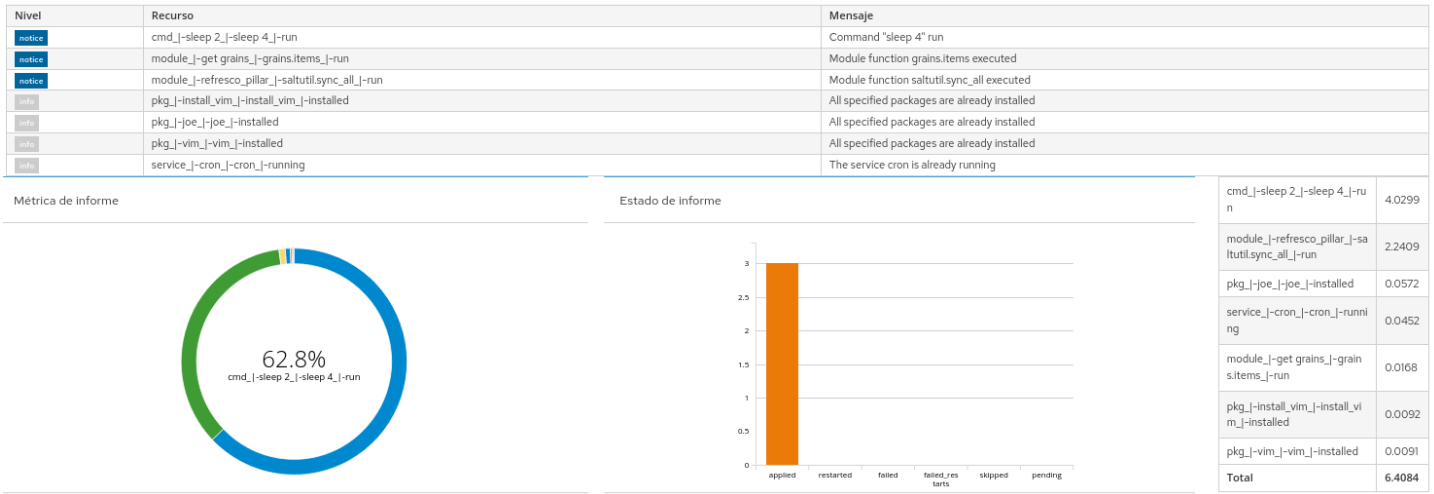
host = a5812110ec1d9040882977cb49576282

| Nombre | Valor |
|---|---|
| locale_info | |
| packages | |
| cpu_flags | |
| ip6_interfaces | |
| saltversioninfo | |
| pythonversion | |
| hwaddr_interfaces | |
| pythonpath | |
| ip4_interfaces | |
| osrelease_info | |
| ip_interfaces | |
| ipv4 | |
| ipv6 | |
| kernelparams | |
| disks | |
| dns | |
| systempath | |
| systemd | |
| fqdn_ip4 | |
| fqdns | |
| gpus | |
| cwd | / |
| uid | 0 |
| serialnumber | 0 |
| gid | 0 |
| num_gpus | 1 |
| ip4_gw | 10.6.0.1 |
| osrelease | 11 |

- Informes: Muestra la recepción de datos desde la máquina cliente




En el listado se ve cuando han sido recibidos los informes, desde que tipo de origen, en este caso Salt y un contador de estados aplicados o errores encontrados, pulsando el tiempo de recepción del informe se puede visualizar el detalle.




Se puede observar las diferentes acciones ejecutadas (estados que debe cumplir) en esa sincronización. En concreto esta traza está ejecutando a través de la programación Salt cada 5 minutos para que alcance el estado principal "highstate".

| Nivel | Recurso |
|---|---|
| notice | cmd_\|-sleep 2_\|-sleep 4_\|-run |
| notice | module_\|-get grains_\|-grains.items_\|-run |
| notice | module_\|-refresco_pillar_\|-saltutil.sync_all_\|-run |
| info | pkg_\|-install_vim_\|-install_vim_\|-installed |
| info | pkg_\|-joe_\|-joe_\|-installed |
| info | pkg_\|-vim_\|-vim_\|-installed |
| info | service_\|-cron_\|-cron_\|-running |

También se muestran los mensajes emitidos por las tareas y el tiempo consumido en su realización:

| Mensaje |
|---|
| Command "sleep 4" run |
| Module function grains.items executed |
| Module function saltutil.sync_all executed |
| All specified packages are already installed |
| All specified packages are already installed |
| All specified packages are already installed |
| The service cron is already running |

| | |
|---|---|
| cmd_\|-sleep 2_\|-sleep 4_\|-run | 4.0299 |
| module_\|-refresco_pillar_\|-saltutil.sync_all_\|-run | 2.2409 |
| pkg_\|-joe_\|-joe_\|-installed | 0.0572 |
| service_\|-cron_\|-cron_\|-running | 0.0452 |
| module_\|-get grains_\|-grains.items_\|-run | 0.0168 |
| pkg_\|-install_vim_\|-install_vim_\|-installed | 0.0092 |
| pkg_\|-vim_\|-vim_\|-installed | 0.0091 |
| **Total** | **6.4084** |

- Puppet Yaml (aunque puede cambiar el nombre del botón): Representa la información configurada para ser notificada (de forma privada) a ese nodo en concreto, contiene:
  - Valores privados del host
  - Valores para ser aplicados en las tareas de forma que se configuren servicios con esos valores, posibilidad de aportar usuarios o passwords por esta vía
  - Información de variables Foreman aplicadas a este equipo
  - Grupos o Estados Salt aplicados a este equipo.

Esta información es muy importante y también puede obtenerse a través de línea de comandos mediante el comando "`foreman-node <hostname>`" obteniendo algo del estilo:

```
root@forelab:~# foreman-node a5812110ec1d9040882977cb49576282
---
classes:
- todos
- vim
parameters:
  foreman_config_groups: []
  puppetmaster: ''
  foreman_env: production
  hostname: a5812110ec1d9040882977cb49576282
  fqdn: a5812110ec1d9040882977cb49576282
  hostgroup: grupo1
  root_pw:
  foreman_subnets: []
  foreman_interfaces:
  - ip: 10.6.0.15
    ip6:
    mac: '08:00:27:ed:ab:aa'
    name: a5812110ec1d9040882977cb49576282
    attrs: {}
    virtual: false
    link: true
    identifier: enp0s3
    managed: true
    primary: true
    provision: true
    subnet:
    subnet6:
    tag:
    attached_to:
    type: Interface
  - ip: 10.0.3.15
    ip6:
    mac: '08:00:27:74:1b:c0'
    name: ''
    attrs: {}
    virtual: false
    link: true
    identifier: enp0s8
    managed: false
    primary: false
    provision: false
    subnet:
    subnet6:
    tag:
    attached_to:
    type: Interface
  location: Default Location
  location_title: Default Location
  organization: Default Organization
  organization_title: Default Organization
  owner_name: Admin User
  owner_email: root@forelab.lan
```

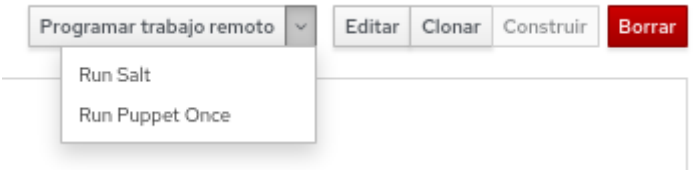
```
  ssh_authorized_keys: []
  foreman_users:
    admin:
      firstname: Admin
      lastname: User
      mail: root@forelab.lan
      description:
      fullname: Admin User
      name: admin
      ssh_authorized_keys: []
  enable-official-puppet7-repo: true
  enable-puppet7: true
  enable-puppetlabs-puppet7-repo: true
  enable-salt-repo: true
  force-puppet: true
  host_packages: ''
  host_registration_insights: false
  host_registration_remote_execution: false
  puppet_server: forelab.forelab.lan
  salt_master: forelab.forelab.lan
  schedule:
    highstate:
      function: state.highstate
      minutes: 10
  skip-puppet-setup: true
  remote_execution_ssh_keys:
                                                    - ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDMkN34btoi9R3pzrx7dlIkLRvdK1G0qngXAI1Uoew4xNL+6CSR05j2mfuIeYlIaXAEnIsYQPC7sXqa
qQJ5K986VfkLhaVugW+acPCYYNPA6qcr4tyslFI7WMPTBB7q3ixpoZBxj9TyhRHPOkeb9UZXB4HypVzQ1oGlM7F4x/K9FtbI4llyY7HkqPMv
O5OYJXeCqn2t/De2ofGeWQ0A/TVEO5u9B5PBJeT7DG1CbXzLUai8z6iz3V7WtJHgTrOvBQq2FU7NsapGsphhiI9Gc1uFQ3/mPXxLO5rasWQt
ZCun6FKd+zja+0ZfYjQmqvxak4ixlJa7pKOPHesc1Sezp3SQO+R419BveuC9hX3gErIbGFm1O5PGsI/z2cU+wfwq9kFEBE4fUnEUD0kNqq3C
xQh8BEeWEP6P1f8V8utDAZ/OHFBKKPmEYrs3RTHtScBcgEOQ13rwdlHwVgBTJX3nc1CNUN4t9+J+acMh8b7LHo9xmtehdbuPpeJM3t0ur6BJ
daM=
    foreman-proxy@forelab.forelab.lan
  remote_execution_ssh_user: root
  remote_execution_effective_user_method: sudo
  remote_execution_connect_by_ip: false
  saltenv: base
environment: base
```

La herramienta "foreman-node" es importante porque actúa como clasificador externo de equipos (ENC) para la tecnología Puppet/Salt. Para la tecnología Salt representa también la forma de acceder a los datos "pillar" de un equipo.

Este comportamiento está configurado durante la instalación a través del fichero de configuración del servidor maestro Salt para la integración con Foreman (`/etc/salt/master.d/foreman.conf`):

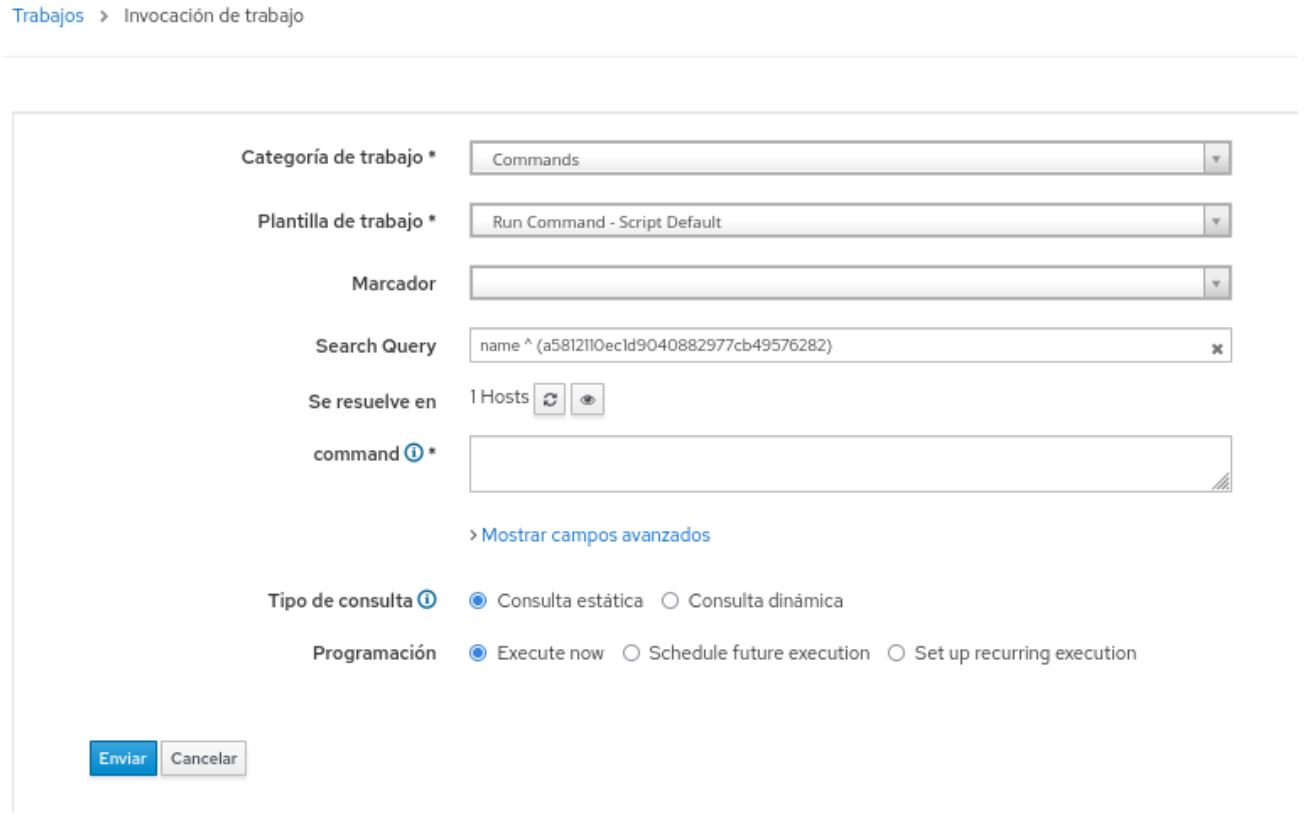
```
##
# Node classifier
master_tops:
  ext_nodes: /usr/bin/foreman-node


##
# Pillar data access
ext_pillar:
  - puppet: /usr/bin/foreman-node
```

Nota: Aunque ha sido comentada la herramienta principal de Foreman para gestionar el ENC que proveerá de los datos "pillar" para la tecnología Salt, desde Salt tambíen es posible consultar estos datos, ver apartado de línea de comandos Salt.

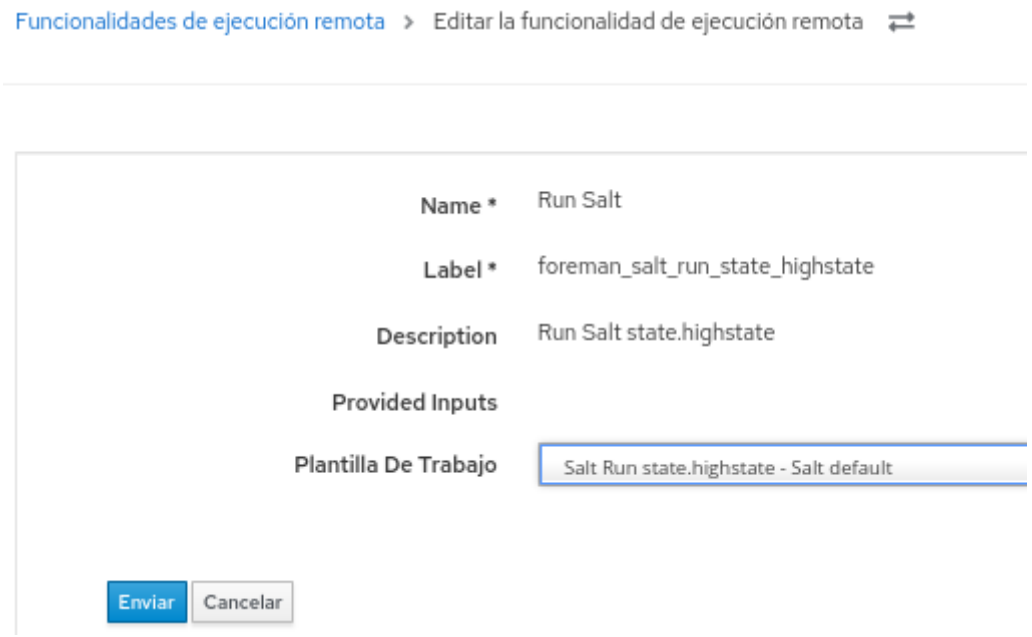



Continuando con las opciones que hay disponibles desde la interfaz web de la vista de equipo, en la parte superior derecha se tienen botones para realizar algunas acciones, las más útiles son:

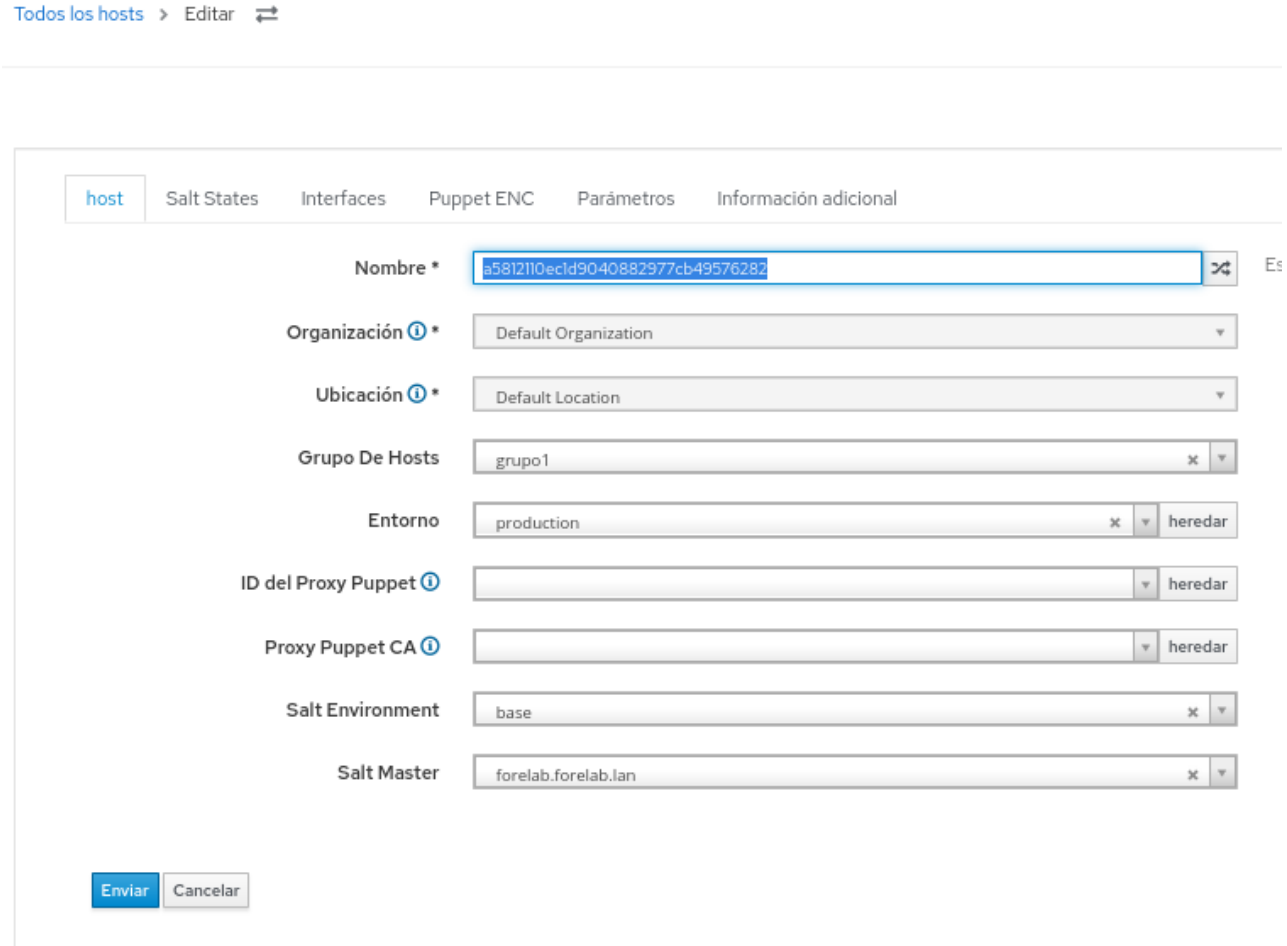
- "Programar un trabajo remoto":



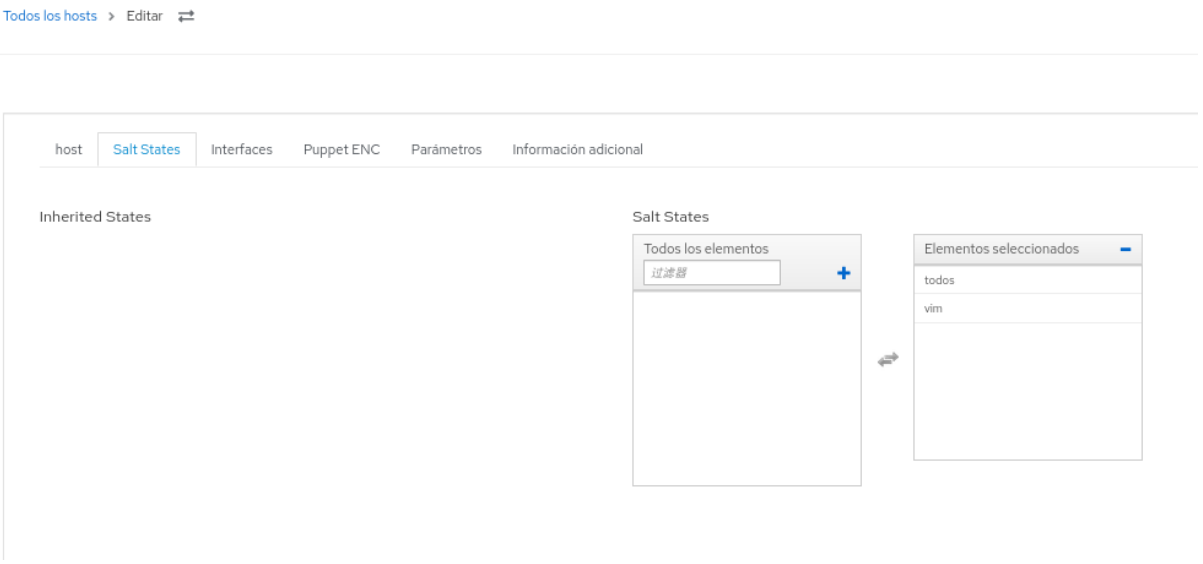

- "Ejecutar Salt", Nota importante: debe configurarse que acción será ejecutada a través la la opcion "Run Salt" desde la configuración de ejecución remota (menú Administrar→Remote Execution Features), no debe ejecutarse a través de SSH, debe de utilizarse solo acciones a través de canales de comunicación Salt.




- Editar: Edita los parámetros del equipo como pueden ser el grupo al que pertenece, entornos, selección del servidor maestro Salt…

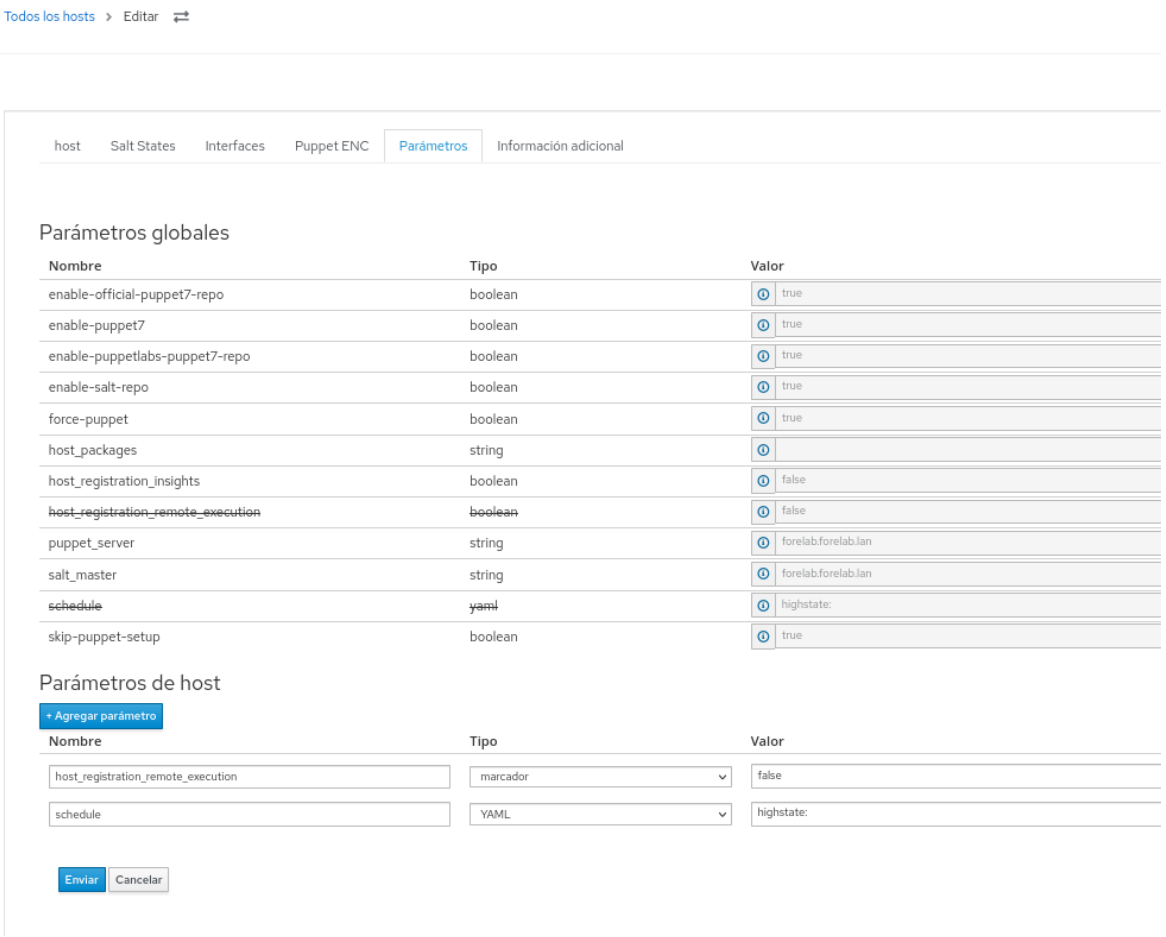


… estados Salt aplicados a este equipo …




Nótese que los elementos "estados Salt" seleccionados son los que aparecen al principio de la información que proporciona el "clasificador externo de nodos" y la información de "pillar" obtenida a través de la utilidad `foreman-node`.

…también se ajustan parámetros del host, por defecto se heredan los globales del sistema o objetos de orden superior, no obstante, pueden ser sobreescritos con mayor especificidad desde esta pestaña de la edición de host…



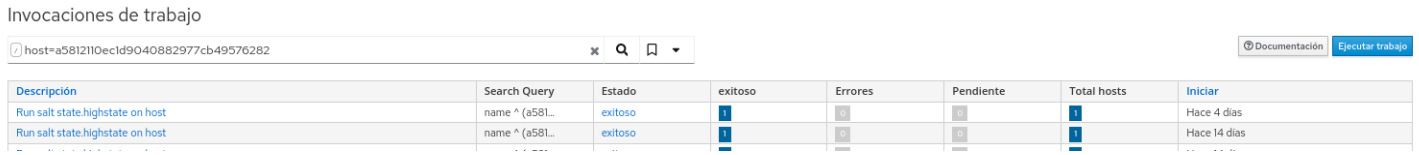
- Borrar: Permite eliminar el borrado de este equipo del sistema Foreman, ya no será administrable a través del servicio.

## Obtención de información

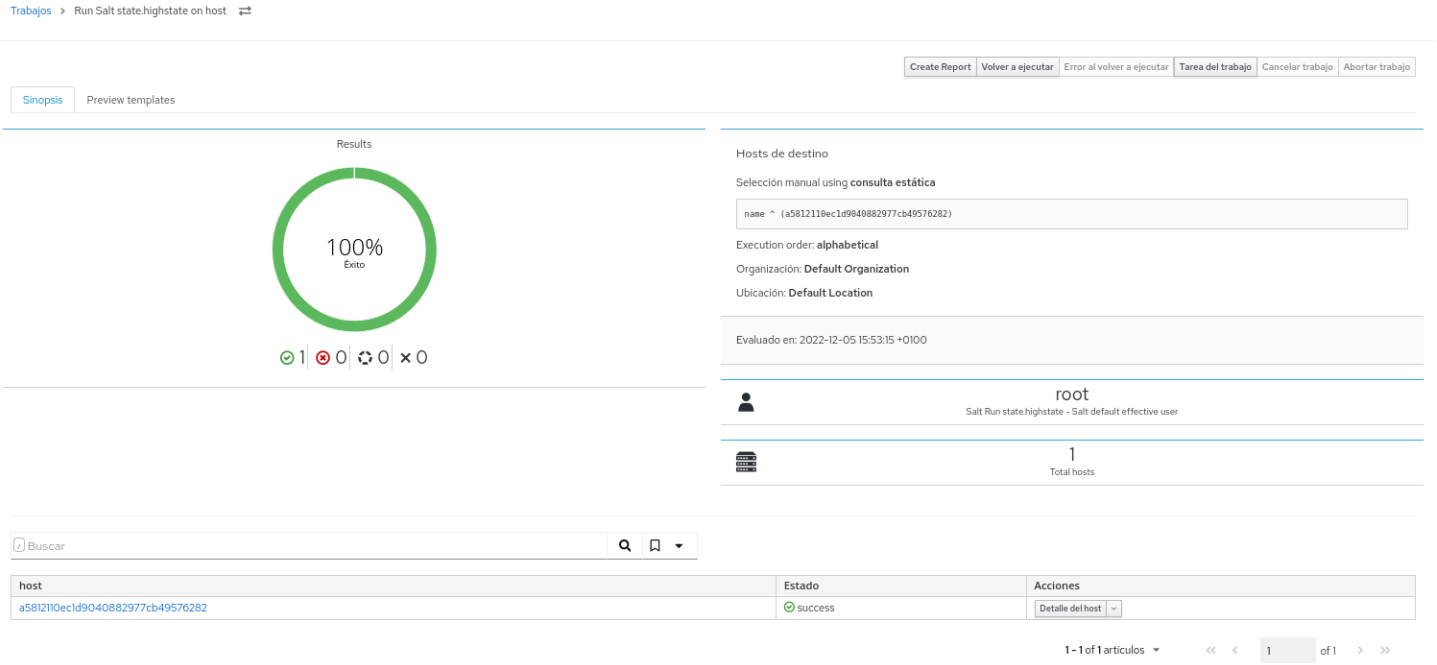
Para la obtención de información del equipo administrado generalmente debe ejecutarse algún tipo de comando que debe ser recopilada por el servidor centralizado.

Para ejecutar este comando, debe ser realizado a través de la programación que ofrece Foreman. Utilizando la programación desde Foreman quedará una entrada en la lista de trabajos de este host, la cual contiene una traza del trabajo con la salida emitida.

Lista de trabajos (a través del botón trabajos del host, o buscando de forma general el host a través del menú Monitor→Jobs) :

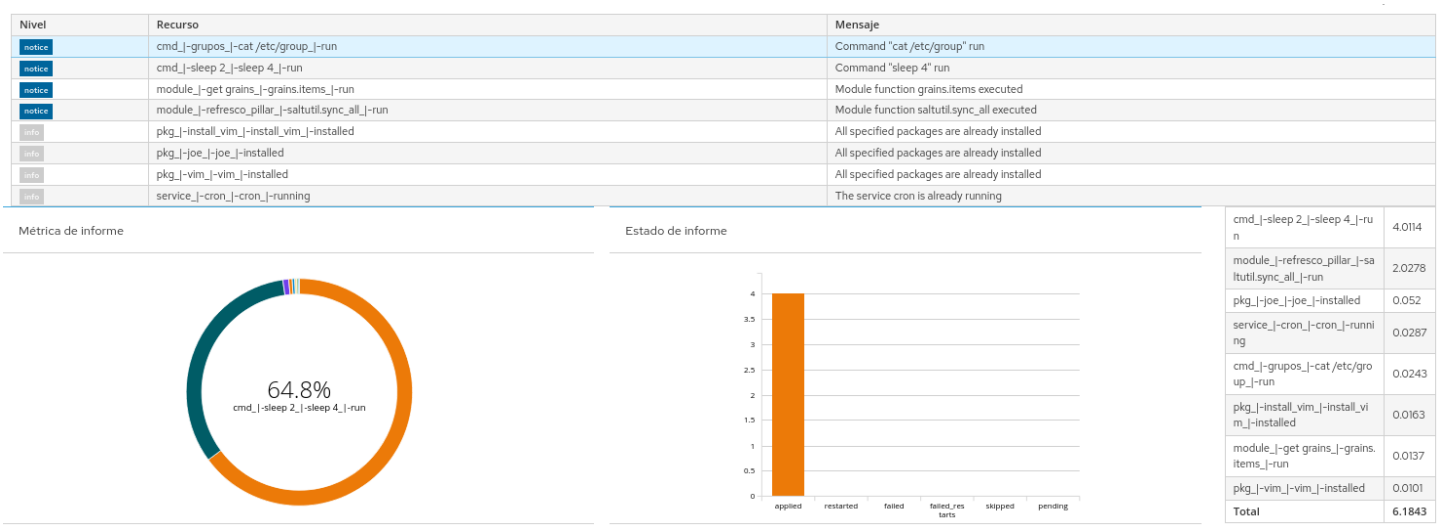



Detalle del trabajo (el registro de la salida en la parte inferior):




Vista de la salida del registro:

Si el comando es ejecutado a través de la programación Salt, las únicas trazas que serán almacenadas en Foreman, serán a través de la opción de informes remitidos hacia Foreman, las cuales contienen métricas de ejecución de los estados alcanzados y si han terminado con éxito o no.



| Nivel | Recurso | Mensaje |
|---|---|---|
| notice | cmd_\|-grupos_\|-cat /etc/group_\|-run | Command "cat /etc/group" run |
| notice | cmd_\|-sleep 2_\|-sleep 4_\|-run | Command "sleep 4" run |
| notice | module_\|-get grains_\|-grains.items_\|-run | Module function grains.items executed |
| notice | module_\|-refresco_pillar_\|-saltutil.sync_all_\|-run | Module function saltutil.sync_all executed |
| info | pkg_\|-install_vim_\|-install_vim_\|-installed | All specified packages are already installed |
| info | pkg_\|-joe_\|-joe_\|-installed | All specified packages are already installed |
| info | pkg_\|-vim_\|-vim_\|-installed | All specified packages are already installed |
| info | service_\|-cron_\|-cron_\|-running | The service cron is already running |



| | |
|---|---|
| cmd_\|-sleep 2_\|-sleep 4_\|-run | 4.0114 |
| module_\|-refresco_pillar_\|-saltutil.sync_all_\|-run | 2.0278 |
| pkg_\|-joe_\|-joe_\|-installed | 0.052 |
| service_\|-cron_\|-cron_\|-running | 0.0287 |
| cmd_\|-grupos_\|-cat /etc/group_\|-run | 0.0243 |
| pkg_\|-install_vim_\|-install_vim_\|-installed | 0.0163 |
| module_\|-get grains_\|-grains.items_\|-run | 0.0137 |
| pkg_\|-vim_\|-vim_\|-installed | 0.0101 |
| **Total** | **6.1843** |

El método óptimo para recopilar información (generalmente estática o poco cambiante) del equipo administrado es a través de los llamados "grain", que son recopilados por el equipo y remitidos al servidor Foreman, estos son almacenados en la base de datos y pueden ser visualizados a través del botón eventos del host.

| | |
|---|---|
| autosign_key | e531a9fbc07f5904c9e4 |
| biosreleasedate | 12/01/2006 |
| biosversion | VirtualBox |
| cpuarch | x86_64 |
| ⌄ cpu_flags | |
| cpu_model | Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz |
| cwd | / |
| ⌄ disks | |
| ⌄ dns | |
| efi | false |

Esta misma información puede obtenerse en línea de comandos a través de los siguientes comandos.

```
salt '<identificador>' grains.items

root@forelab:~# salt 'a58*' grains.items|head
a5812110ec1d9040882977cb49576282:
    ----------
    autosign_key:
        e531a9fbc07f5904c9e4
    biosreleasedate:
        12/01/2006
    biosversion:
        VirtualBox
    cpu_flags:
        - fpu
```

Existen más formas de realizar la llamada anterior, no obstante está es la forma adecuada para ser realizada desde una línea de comandos desde el servidor central, desde el propio equipo local, esta llamada sería realizada mediante:

```
salt-call --grains
```

La información contenida en el conjunto de información que por defecto es remitida es bastante generalista y puede no contener toda la información que de una forma útil es deseada recibir, por ello es muy posible que deba ser ampliada para incluir un conjunto más grande de información, esto es realizado a través de la inclusión de "custom grains" que pueden contener información estática (un valor fijo) o bien dinámica (ejecución de un comando).

### Obtención de información adicional (custom grain)

Para la obtención de información adicional (poco cambiante) del equipo administrado puede añadirse para que sea añadida información estática o dinámica a la generación del listado de "grains".

Si la información es estática, no cambiante, se puede fijar el valor en la configuración de la configuración del agente a través de algún fichero (*.conf) en la carpeta de configuraciones `/etc/salt/minion.d`

```
grains:
      identificador: equipo_77
      asignado_a:
         -  Pedro Perez
         -  Juan Gomez
```

No obstante el método más útil para generar información en un "grain" de forma generalista y más versátil es ejecutando un comando que puede ser programado, permitiendo información estática o dinámica a través de scripts Python y tan sólo introduciendo (o retirando) del servidor central el script.

Será necesario crear la carpeta de grains ("`_grains`") dentro de la carpeta servida por el servidor Salt, por defecto `/srv/salt` y dejar los script Python ahí.

Estos script Python tan solo deben devolver un diccionario con los valores a ser añadidos.

Por ejemplo, un script para aportar "uptime" a los "grain" (por defecto la tecnología Puppet lo provee, no obstante la tecnología Salt no lo incluye, es utilizado por Foreman por defecto).

```
#!/usr/bin/env python

import subprocess

def main():
    return {'uptime':subprocess.check_output('uptime')}
```

## Configurar la vista de equipo

A través de la página de la vista de host (o lista general de todos los host) puede ser visualizada determinada información del host directamente organizada como propiedades del host.

| Propiedades | | |
|---|---|---|
| Estado | Aceptar | |
| Construir | Instalado | borrar |
| Configuración | Activo | borrar |
| Ejecución | La última ejecución se realizó correctamente. | borrar |
| Duración de la compilación | menos de 1 minuto | |
| Token | N/C | |
| Uptime | 10:16:46 up 1:00, 1 user, load average: 0,07, 0,04, 0,00 | |
| UUID | a5812110-ec1d-9040-8829-77cb49576282 | |
| Dirección IP | 10.6.0.15 | |
| Dirección MAC | 08:00:27:ed:ab:aa | |
| Entorno de Puppet | production | |
| Arquitectura | x86_64 | |
| Sistema operativo | Debian 11 | |
| Cargador de PXE | | |
| Grupo de hosts | grupo1 | |
| Tiempo de arranque | No informado | |
| Ubicación | Default Location | |
| Organización | Default Organization | |
| Propietario | Admin User | |

No obstante en la vista mostrada, no toda la información está incluida por defecto, para el caso, los campos "Uptime" y "UUID" no están incluidos por defecto en la instalación de Foreman, está información es obtenida a través de los valores de los "grain" del host.

| Host group | Last report | Comment | Architecture | Uptime | Acciones |
|---|---|---|---|---|---|
| grupo1 | Hace 4 minutos | | x86_64 | 10:26:46 up 1:10,... | Editar |
| grupo1/grupo1_1 | Hace 28 dias | | x86_64 | 1:50 hours | Editar |

Para ser mostrada en la vista de propiedades (o el listado general de hosts) en Foreman, es utilizado el plugin "foreman_column_view".

Para utilizarlo es necesario ajustar el contenido a mostrar editando el fichero de configuración del plugin:

```
/etc/foreman/plugins/foreman_column_view.yaml
```

Por defecto se configuran las columnas de la vista de todos los hosts, pero también se pueden configurar las propiedades que aparecen en la columna de los detalles dentro de la vista de un host.

La línea “:view: :hosts_properties” especifica que se está definiendo el elemento dentro de la columna de los detalles en la vista de host individual.

```
:column_view:
  :architecture:
    :title: Architecture
    :after: last_report
    :content: facts_hash['cpuarch']
  :uptime:
    :title: Uptime
    :after: architecture
    :content: facts_hash['uptime']
  :uptime_host:
    :title: Uptime
    :after: 6
    :content: facts_hash['uptime']
    :view: :hosts_properties
  :uuid_host:
    :title: UUID
    :after: 7
    :content: facts_hash['uuid']
    :view: :hosts_properties
```

Nota: En el sistema Puppet los valores aparecen con otra nomenclatura o en otro nivel de jerarquía, a modo de ejemplo válido para Salt si el elemento a mostrar no aparece en el primer nivel de la lista de "grains" el formato a utilizar sería:

| | |
|---|---|
| Puppet: `facts_hash['architecture']` | Salt: `facts_hash['cpuarch']` |
| Puppet: `facts_hash['dmi::product::uuid']` | Salt: `facts_hash['uuid']` |

Una vez modificado el fichero de configuración del plugin, será necesario reiniciar el servicio Foreman para que los cambios sean aplicados.

```
systemctl restart foreman
```

### Identificación del sistema operativo utilizado

De forma general a través de las funciones básicas que provee Salt ya está incluida la información del sistema operativo, no obstante para este proyecto y en concreto dado que será utilizada la distribución Lliurex como sistema operativo de los equipos administrados, puede ser necesario personalizar esta detección de sistema operativo debido que funciones estándar a través del comando `lsb_release` pueden no reflejar información correcta (dado que Lliurex es una distribución derivada de Ubuntu que no necesariamente desea o puede modificar esta identificación).

La información de los "grain" obtenidos por el "core" de Salt, pueden ser sobreescritos por valores calculados a través de "custom grains".

Para este proyecto sobre Lliurex por ejemplo puede ser necesario utilizar `lliurex-version` para aportar información acerca del sistema operativo.

## Mostrar paquetería instalada y versiones

Cuando es necesario almacenar la información acerca del software instalado en un equipo administrado y que sea fácilmente mostrado/organizado/buscado/almacenado también puede ser útil almacenarlo en forma de "custom grains".

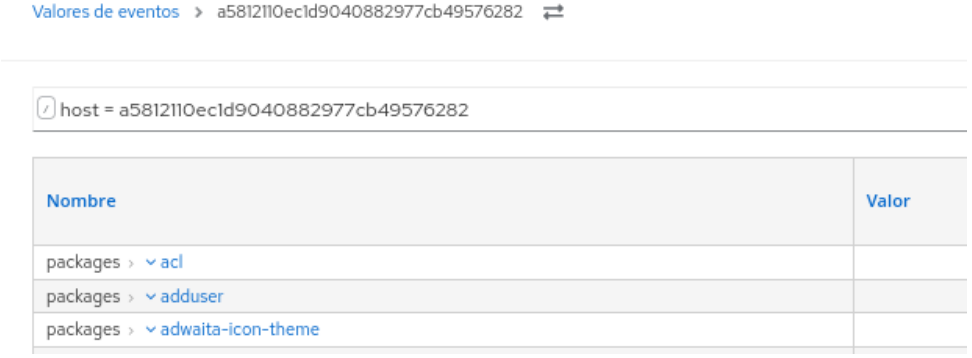
Puede ser utilizado un script Python que realice la llamada al comando de sistema operativo y organice la información correctamente para ser mostrada adecuadamente desde la interfaz Foreman.

Por ejemplo:

```
#!/usr/bin/env python

import subprocess

def main():
    pkgs = subprocess.check_output(['/usr/bin/dpkg-query','-W','-f',"${Package}
${Version}\n"]).decode().split('\n')
    packages = {}
    for pkg in pkgs:
        try:
            name, version, *other = pkg.split(' ')
        except:
            pass
        packages.setdefault(name, [version])
    return { 'packages' : packages }
```

Utilizando este script Python la información de paquetería será mostrada en niveles con el siguiente aspecto:

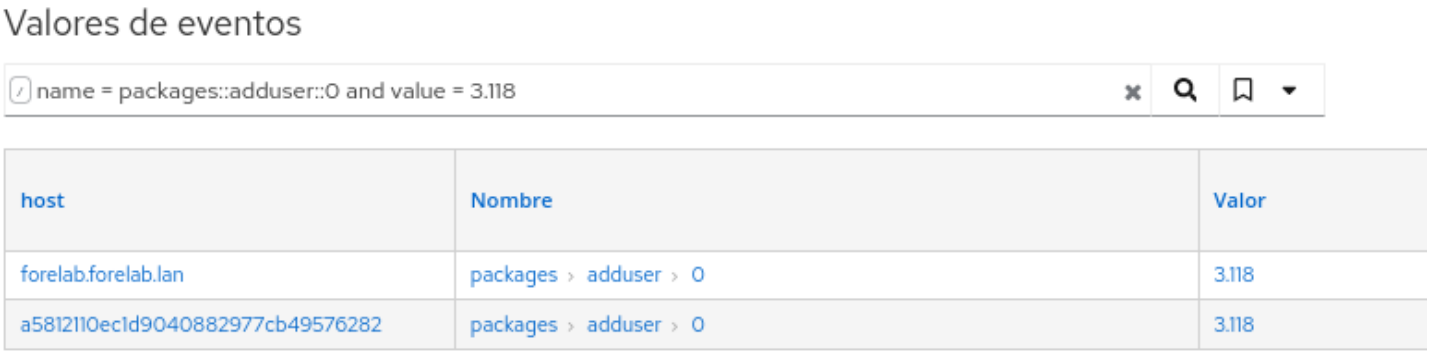

Navegando a través de la clave "packages":



Navegando a través del paquete deseado:


### Localizar equipos que cumplan un determinado criterio de paquetería

Si en algún momento es necesario realizar una búsqueda inversa a través de todos los equipos que cumplan un determinado criterio en función de la paquetería instalada, puede utilizarse el filtro de búsqueda de eventos del menú (Monitor→Eventos) para realizar esta localización de equipos utilizando las claves "name" y "value".




Cuando se están realizando pruebas para la generación de nuevos grain sobre un equipo local es posible que surjan problemas y pueden ser necesarias las siguientes llamadas Salt.

```
salt-call --grains (obtiene grains localmente)
salt-call saltutil.sync_grains refresh=true -l debug (sincroniza grains desde el servidor)
salt 'forelab*' saltutil.clear_cache (limpia la caché)
```

Otro problema que puede surgir es que hayan sido almacenados determinados grain en el servidor y deseen ser eliminados durante el proceso de desarrollo, esto no es una tarea de mantenimiento prevista en Foreman y puede resultar muy tedioso eliminar esa información una vez almacenada. A continuación se muestra el método más rápido y efectivo para realizar esta tarea, dado que de otra forma habría que realizar manipulación manual de los datos almacenados en la base de datos Postgres, un proceso delicado dado que puede romper la coherencia de los datos y relaciones almacenados.

### Eliminar "grains" ya almacenados (método "build_mode")

Consiste en poner el host en el estado "en construcción" donde al cambiarse como "ya construido" Foreman inicializará la base de datos de "grains" o eventos de ese equipo, dejando un listado vacío.

Colateralmente el equipo puede ser eliminado del sistema de autenticación Salt, no obstante puede volver a validar la identificación de ese host manualmente y en siguientes contactos por parte del cliente (se puede forzar un reinicio del cliente o esperar a que sea reiniciado el equipo).

Posteriormente se establecerá comunicación con el servidor centralizado mandando de nuevo un listado de "grain" que será almacenado poblando nuevamente la lista de eventos del host.

```
hammer host update --name 'forelab.forelab.lan' --build true
hammer host update --name 'forelab.forelab.lan' --build false
systemctl restart salt-minion
salt-key -A -y
systemctl restart salt-minion
```

## Administración de software

Para realizar una administración de software en los equipos administrados, lo que representa el caso de uso más importante o más utilizado de este sistema de administración será necesario introducir los conceptos y forma de funcionamiento que presenta la tecnología Salt utilizada para la gestión de la configuración en este proyecto.

### Tecnología Salt

Salt representa mediante "estados" (states) algunas condiciones que deben ser cumplidas por el equipo en el que están siendo aplicados.

Estos estados hacen uso de "módulos de ejecución" para expresar el estado, estos "módulos de ejecución" consisten en funciones Python parametrizadas definidas y organizadas en ficheros que realizan una variedad de acciones posibles, no obstante cabe aclarar que no es lo mismo un módulo de estado que un módulo de ejecución.

Los estados definidos a través de la tecnología Salt son organizados a través de ficheros plantilla nombrados con extensión SLS y dado que son distribuidos a los agentes cliente para su aplicación están ubicados en la carpeta pública del servidor Salt `/srv/salt`.

Estas plantillas pueden ser escritas de forma estática o bien utilizando la sintaxis del motor de plantillas Python Jinja, lo cual puede ser útil para realizar un código más compacto que permite asemejarse a un lenguaje de programación con bucles y condicionales.

Los ficheros SLS son ficheros en formato YAML estructurados y son organizados a partir de un fichero inicial que define el llamado "highstate", este fichero debe ser creado para definir unos entornos y la continuación de la jerarquía (árbol de definiciones de estados a través de más ficheros enlazados).

El fichero "highstate" es el fichero /srv/salt/top.sls que presenta el siguiente contenido por ejemplo:

```
base:                       # define el entorno
      '*':                  # define a qué equipos es aplicado
         -  todos           # define una lista de ficheros enlazados
```

Puede haber ficheros no enlazados a través de esta jerarquía, que definen estados y no serán aplicados automáticamente pero Foreman es consciente (a través del proceso de importación) de la existencia de estos, pudiéndose asignar a unos equipos y no a otros desde la propia interfaz Foreman.

En el caso anterior, el fichero top.sls que es aplicado al entorno "base" y todos los equipos en él (*) aplicará el contenido del fichero de estados `/srv/salt/todos.sls` el cual puede presentar un contenido parecido al siguiente (se añaden los números de línea para los comentarios posteriores del código):

```
(01)refresco_pillar:
(02)    module.run:
(03)        - name: saltutil.sync_all
(04)        - refresh: True
(05)#        - clear_cache: True

(06)cron:
(07)    service.running

(08)sleep 2:
(09)    cmd.run:
(10)        - name: sleep 4

(11)grupos:
(12)    cmd.run:
(13)        - name: cat /etc/group

(14)get grains:
(15)    module.run:
(16)        - name: grains.items

(17)joe:
(18)    pkg.installed
```

Nota previa a la explicación, en un sistema Debian los ficheros de código que son utilizados por el servicio Salt está ubicado en la siguiente ruta:

`/usr/lib/python3/dist-packages/salt` (en la explicación será referenciado como <salt-pylibs>)

El contenido muestra 6 estados, presentando la siguiente sintaxis Salt para definir los estados y utilidad:

- Estado 1:
  - Está definido como un estado que hace uso de un módulo de ejecución
  - (01) Utiliza un identificador libre nombrado como "refresco_pillar"
  - (02) Hace uso de la llamada a un módulo ejecutable
  - Utilizará los siguientes parámetros en la llamada al módulo ejecutable:
    - (03) Parámetro "name", nombre del módulo ejecutable a ejecutar que estará ubicado en la carpeta de módulos ejecutables en el servidor <salt-pylibs>/modules/saltutil.py
    - (04) Parámetro "refresh" con el valor "True"
    - (05) Parámetro comentado "clear_cache" con valor "True"
  - Realiza la tarea de forzar una sincronización de todo el contenido que provee el servidor a los clientes, con posibilidad de forzar un borrado de la caché en el agente cliente.

- Estado 2:
  - Está definido como un estado que hace uso de un módulo de estado
  - (06) Utiliza un identificador de estado nombrado como "cron"
  - (06) Es mapeado como el primer parámetro "name" del módulo estado a menos que esté definido un parámetro "name" definido
  - (07) Utilizará el módulo estado que será mapeado como <fichero.función> del código que está en la carpeta de modulos de estado <salt-pylibs>/states/service.py
  - Realiza la tarea de activar el servicio en la máquina y asegurar que está en funcionamiento.
- Estado 3:
  - Está definido como un estado que hace uso de un módulo de estado
  - (08) Utiliza un identificador de estado nombrado como "sleep 2"
  - (09) Utilizará el módulo estado que será mapeado como <fichero.función> del código que está en la carpeta de modulos de estado <salt-pylibs>/states/cmd.py
  - (10) Se fija el primer parámetro "name" del módulo estado con lo cual será realizada una llamada al comando "sleep" con valor 4 en vez de 2.
  - Realiza una tarea de ejemplo que realiza una pausa de 4 segundos
- Estado 4:
  - Está definido como un estado que hace uso de un módulo de estado
  - (11) Utiliza un identificador de estado nombrado como "grupos"
  - (12) Utilizará el módulo estado que será mapeado como <fichero.función> del código que está en la carpeta de modulos de estado <salt-pylibs>/states/cmd.py
  - (13) Se fija el primer parámetro "name" del módulo estado con lo cual será realizada una llamada al comando "cat" y el parámetro "/etc/group", sinó estuviera definido este parámetro el estado daría error dado que no podría ser ejecutado el comando "grupos".
  - Realiza una tarea para listar el contenido del fichero /etc/group
- Estado 5:
  - Está definido como un estado que hace uso de un módulo de ejecución
  - (14) Utiliza un identificador libre nombrado como "get grains"
  - (15) Hace uso de la llamada a un módulo ejecutable
  - Utilizará los siguientes parámetros en la llamada al módulo ejecutable:
    - (16) Parámetro "name", nombre del módulo ejecutable a ejecutar que estará ubicado en la carpeta de módulos ejecutables en el servidor <salt-pylibs>/modules/grains.py
  - Realiza la tarea de incluir un listado de los grains del equipo en la salida del estado.
- Estado 6:
  - Está definido como un estado que hace uso de un módulo de estado
  - (17) Utiliza un identificador de estado nombrado como "joe"
  - (18) Utilizará el módulo estado que será mapeado como <fichero.función> del código que está en la carpeta de modulos de estado <salt-pylibs>/states/pkg.py
  - Realiza una tarea para tener instalado el paquete "joe"

Una vez definidos estados como parte del fichero top.sls que serán aplicados a todos los equipos, también pueden definirse múltiples ficheros SLS de estados adicionales ("huérfanos"), estos simplemente estarán disponibles en el servidor para ser aplicados manualmente, ya que no forman parte de la jerarquía incluida como parte del "highstate".

Por ejemplo puede ser creado un fichero `/srv/salt/install_htop.sls` con el siguiente contenido:

```
install_htop:
    module.run:
        - name: pkg.install
        - m_name: htop
```

Aquí se aprecia cómo se está definiendo una instalación de paquete, pero está vez en vez de como un estado, realizando la llamada como un módulo ejecutable a través del estado "module.run", por ello, los parámetros a utilizar son el nombre del módulo ejecutable "pkg.install" y el parámetro a utilizar en esa llamada "name", el cual para diferenciarlo de los posibles parametros que dispone el estado "module.run" debe ser concatenado con "m_", en este caso "m_name".

Para lanzar este fichero estado "huérfano" puede aplicarse sobre un equipo con la siguiente llamada:

```
salt <selector> state.apply <nombre_estado>

salt 'a58*' state.apply install_htop      (desde el servidor maestro)
salt-call state.apply install_htop        (desde el equipo administrado)
```

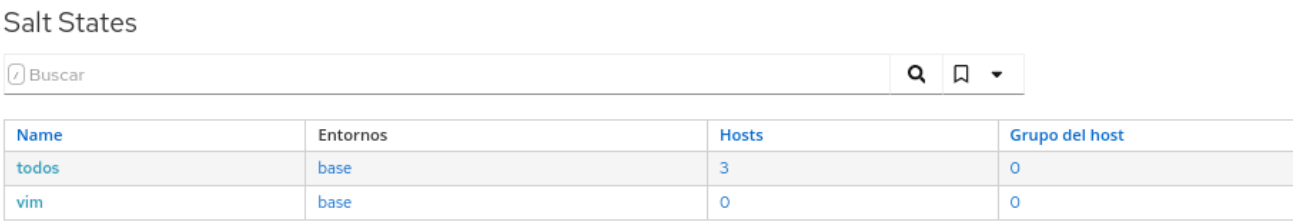
### Pillar

Complementando la definición de estados Salt utiliza otra información representada en formato YAML que es asignada de forma individualizada para cada cliente o agente, está información es provista por el servidor y puede ser consultada tanto desde el propio equipo (solo su información) como desde el servidor (la de cualquier equipo), esta información es nombrada como "pillar".
Como se ha nombrado anteriormente está información cuando se está utilizando un sistema Salt conjuntamente con Foreman es provista por el comando "foreman-node" el cual puede ser lanzado desde el servidor para consultar los datos que utilizará la tecnología Salt.

Esta información contiene datos particulares que pueden ser utilizados de múltiples formas, se puede hacer referencia a estos datos desde ficheros SLS de estados para parametrizar los estados y las acciones que estos realizan, pueden cambiar el comportamiento del cliente como por ejemplo estableciendo programación de acciones, pueden proveer información confidencial acerca de usuarios o contraseñas necesarias para el proceso de gestión de la configuración de una máquina.

### Salt desde Foreman

A través de Foreman pueden ser importados todos los ficheros SLS que existen en el servidor, para ello se accede a través de la página Configurar→Salt (states), donde es presentado un listado con los diferentes ficheros definidos, su entorno y los host o grupos en los que está siendo aplicado dicho fichero de plantilla.



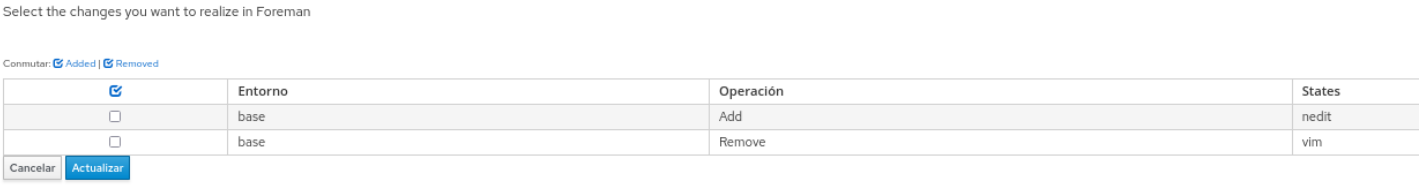
En la esquina superior derecha de la página están disponibles los botones para poder realizar las acciones que añaden estados al sistema Foreman.


- Importar desde el contenido en disco `/srv/salt` del servidor Foreman
- Crear un estado desde Foreman

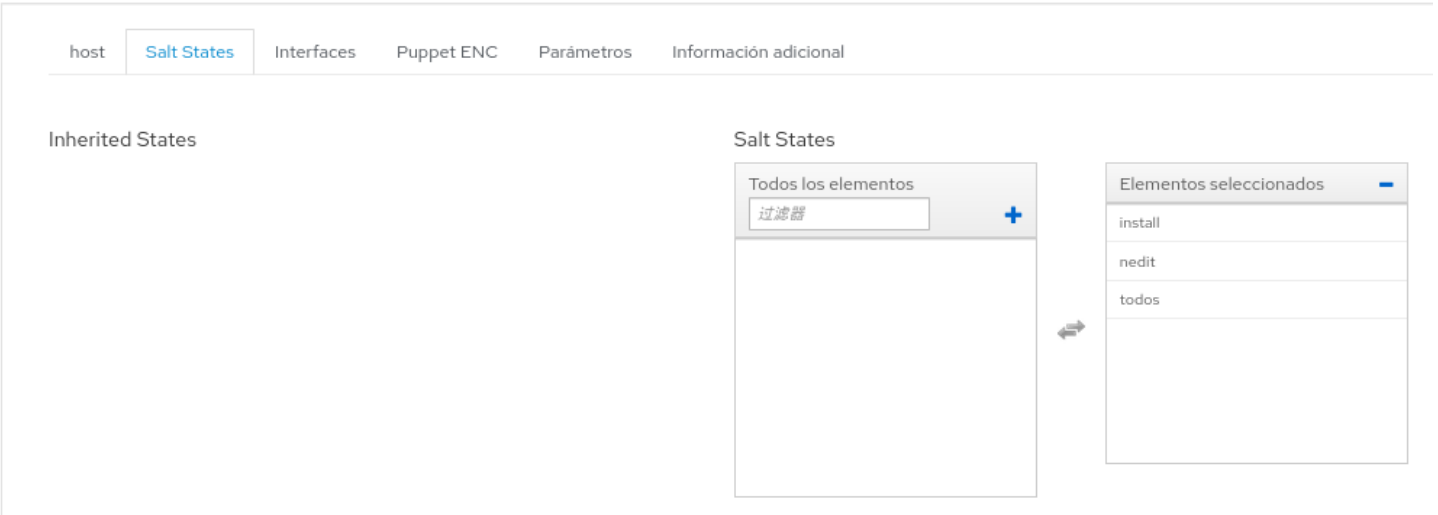
Cuando se presiona el botón importar de servidor, se muestra una página resumen con el contenido que ha cambiado, pueden ser nuevos ficheros o ficheros que ya no están disponibles a través de la creación/copia/borrado de los ficheros en la carpeta pública.

Este proceso es especialmente importante para poder importar todos aquellos ficheros que no están enlazados de alguna forma a través del "highstate", dado que esos ficheros "huérfanos" son los que podrán ser asignados y desasignados desde la interfaz Foreman.

Marcando el checkbox y el botón actualizar, se aplicarán los cambios necesarios para que dicha información esté incluida en Foreman para su uso.

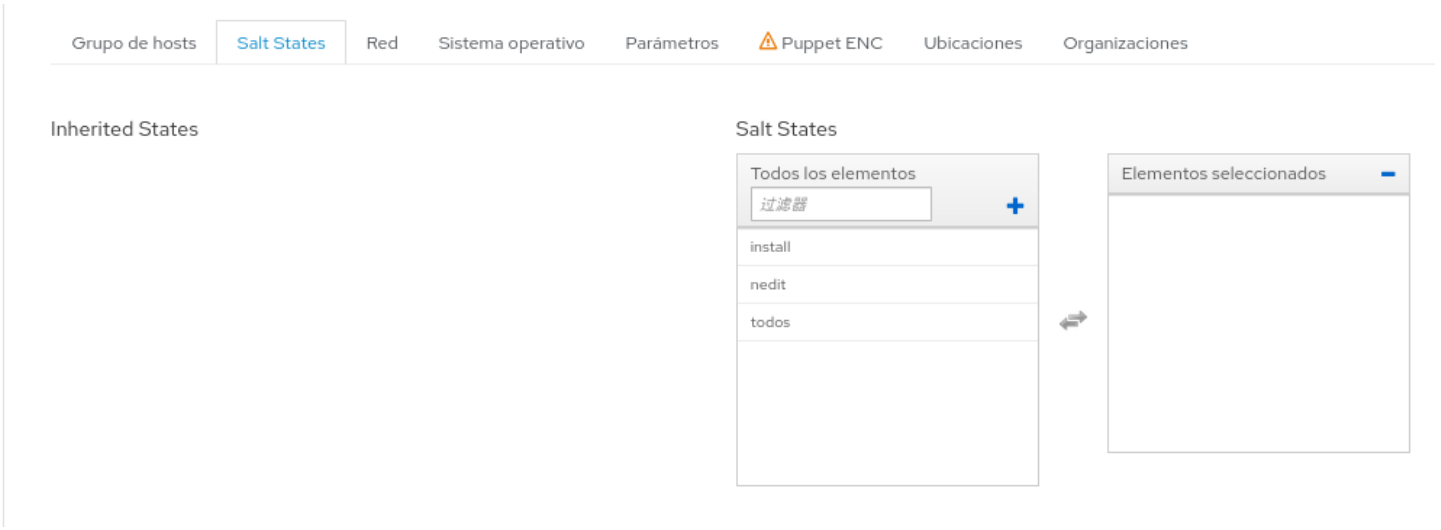



Una vez importados los estados ya aparecen en el listado de estados disponibles e igualmente a través de las propiedades de un host en la pestaña "Salt states" listos para ser aplicados a un host.




Igualmente y de forma similar también pueden ser aplicados a un grupo de hosts creado previamente, de esta forma el estado se aplica de una forma un poco menos específica pero más cómoda para un conjunto de

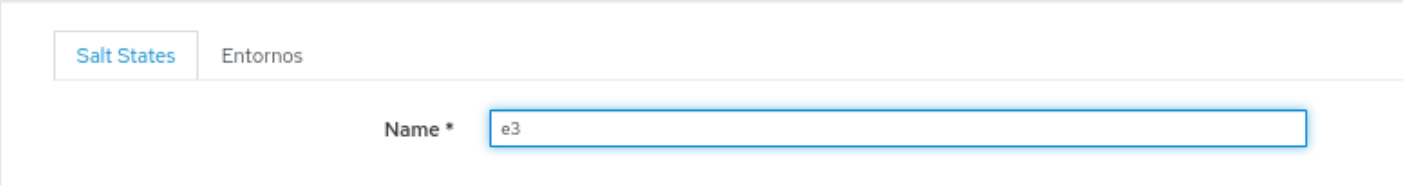
equipos, dado que las propiedades de un host son más específicas que un grupo, tienen prioridad para ser aplicadas o eliminadas de su aplicación.



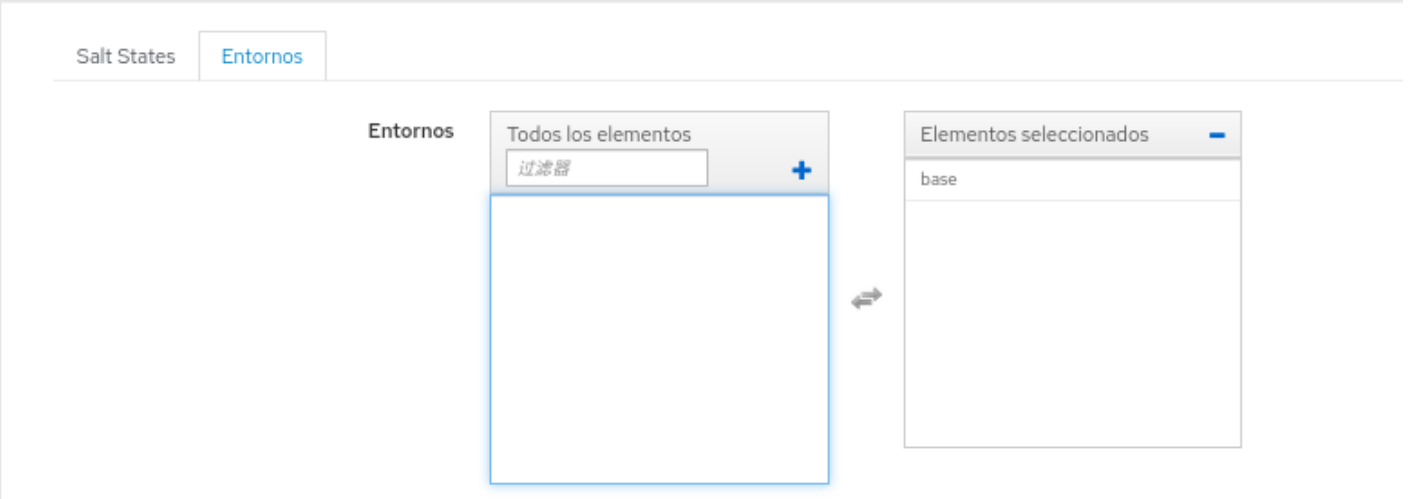

A través de Foreman y como se ha visto anteriormente es posible la creación de un nuevo estado, no obstante en ningún caso Foreman creará ficheros en disco en el servidor, la opción para la creación de estados a través de Foreman simplemente permite crear un objeto "estado" en la interfaz de foreman manualmente, de forma similar a como funciona una importación que detecte el contenido en disco, pero manual. Esta opción es útil si no se dispone del servicio "salt-api" gracias al cual es posible realizar una importación de los estados existentes en disco.

Así se realiza la creación de un estado a través de Foreman:

Se asigna un nombre:



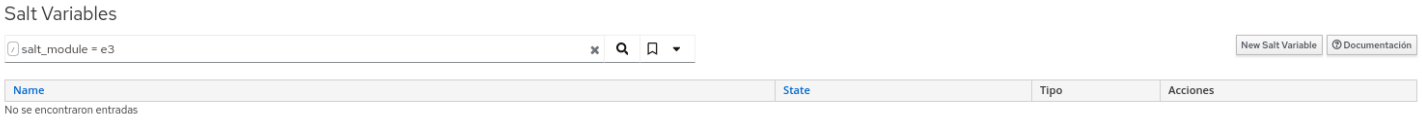

Y un entorno:




A través del menú Configuración→Salt también se dispone de la página de "Variables", el significado que presentan las "variables" es proveer de un lugar adicional donde definir más información (genérica) que será

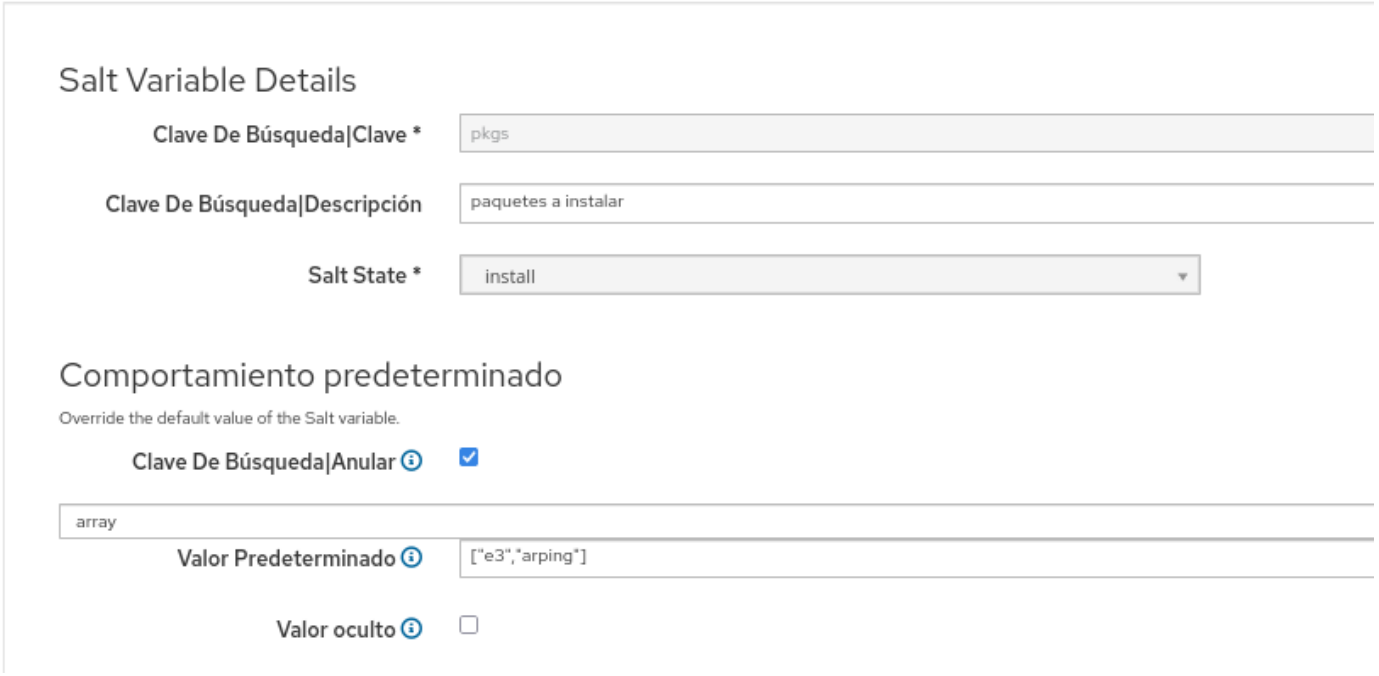
incluida en el conjunto de datos que conforma el "pillar" de un determinado equipo, no obstante, la información definida a través de variables es más específica para su uso con tecnología Salt.

Inicialmente no serán mostradas variables y la lista puede aparecer vacía:




De forma similar a la página de estados, se presentan las acciones disponibles en la esquina superior derecha en forma de botón.


Únicamente se dispone la acción de crear nuevas variables, cuando es seleccionado se presenta un formulario donde rellenar el contenido de la variables.

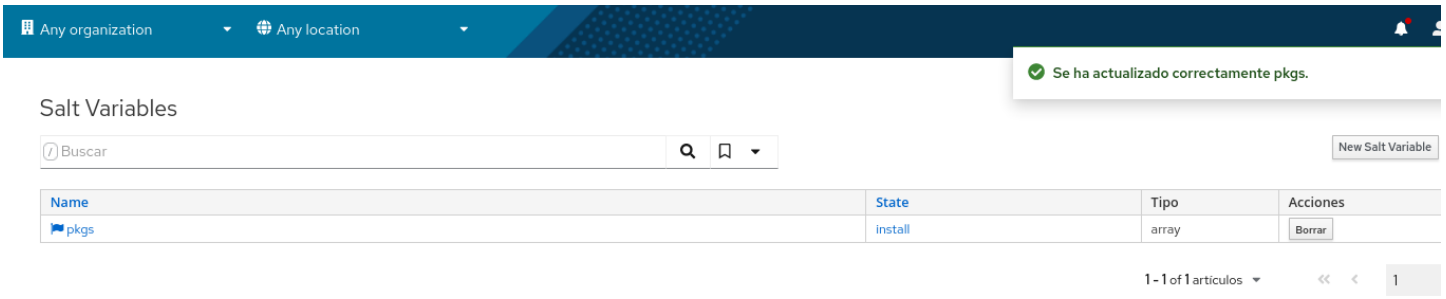



Rellenando los datos como la captura mostrada, es creada una variable nombrada como "pkgs" que representará los "paquetes a instalar" como figura en la descripción introducida, también debe ser seleccionado un estado existente en Foreman, en este caso se ha seleccionado "install".

Junto con el nombre de la clave y el estado, es importante configurar el valor que va a presentar, para ello es necesario seleccionar el checkbox que permite la edición del cuadro de texto valor y especificar el tipo de dato que va a contener el cuadro de texto valor.

En el caso de la captura ha sido seleccionado el tipo "array" con el valor ["e3","arping"]

Posteriormente a completar todos los datos básicos (permite introducir más) es posible pulsar el botón enviar para que sean guardados en el sistema, mostrando el listado con la nueva variable introducida.




La variable introducida estará asociada al estado seleccionado y cuando un host tiene ese estado seleccionado dicha variable estará introducida con su valor entre los datos del "pillar". Puede comprobarse utilizando los comandos:

- `foreman-node <fqdn>` (desde el servidor maestro)
- `salt-call pillar.items` (desde el equipo administrado)
- `salt 'a58*' pillar.items` (desde el servidor maestro)

En cualquier caso serán mostradas las variables en el contenido YAML de información de "pillar". El contenido relevante que debe aparecer es:

```
classes:
  - install
  - nedit
  - todos
```

…demás contenido omitido…

```
pkgs:
  - e3
  - arping
```

Estas "variables" son especialmente útiles para ser utilizadas en plantillas como se ha comentado anteriormente, dado que las plantillas pueden utilizar el motor Jinja para reutilizar código y generar soluciones más limpias y sencillas.
Se muestra como tiene asignado a través del "pillar" la clase "install" (la que está incluida con variables) y la variable "pkgs" con su contenido, una lista con los elementos "e3" y "arping".

Si es utilizado por ejemplo un contenido para el fichero en disco "install.sls" como el siguiente:

```
{% for package in pillar.get('pkgs') %}
{{package}}:
    pkg.installed
{% endfor %}
```

Se estará utilizando el motor de plantillas para definir estados que fijen como instalados la lista de programas definidos en la variable, posteriormente puede editarse el contenido de la variable desde Foreman para incluir o excluir paquetes instalables.

## Salt desde línea de comandos

Aquí se introducirá cual es el funcionamiento de la línea de comandos Salt para realizar las acciones más básicas.

La línea de órdenes Salt tiene diferentes comandos algunos solo están disponibles en el servidor como parte del paquete salt-master y otros están disponibles tanto en el equipo administrado como en el servidor como parte del paquete salt-common.

Algunos comandos disponibles y más utilizados son:

| | |
|---|---|
| `salt-call <modulo.funcion> <parametros>` | Ejecuta funciones de módulos localmente, está pensada para ser utilizada en los agentes minion de los equipos administrados. |
| `salt <selector> <modulo.funcion> <parametros>` | Ejecuta comandos en paralelo en los host seleccionados por el selector, pensada para ser utilizada desde el servidor maestro. |
| `salt-cp <selector> <origen> <destino>` | Copia pequeños ficheros de texto entre el servidor maestro y un conjunto de equipos administrados. |
| `salt-key <opciones>` | Administra las claves que son utilizadas para realizar la conexión segura entre los equipos administrados y el servidor |
| `salt-run <runner>` | Ejecuta módulos "runner" en el servidor maestro. |

En cualquier comando de los descritos existen opciones muy útiles como son las siguientes:

| | |
|---|---|
| `-l <nivel de debug>` | Permite aumentar la verbosidad del comando pudiendo ver trazas o errores más fácilmente. |
| `--out <tipo>` | Permite cambiar el "outputter" generando una salida en diferentes formatos como json, txt, yaml y otros |

Cuando se utilizan selectores para filtrar los equipos en los que va a ser aplicada una acción por defecto se utiliza un campo tratado como un "shell glob" (parecido a las expresiones regulares pero más simple) aplicado al FQDN, no obstante pueden ser utilizados otros criterios a través del uso de opciones como las siguientes:

| | |
|---|---|
| `-E '<pcre_regexp>'` | Permite utilizar expresiones regulares compatibles Perl |
| `-L '<fqdn1,fqdn2,fqdn3>'` | Utiliza una lista separada por comas para definir el destino |
| `-F '<grain:expresion_glob>'` | Utiliza "shell globs" sobre la información de los "grain" calculada por los equipos |

| | |
|---|---|
| `--grain-pcre '<grain:expresion_pcre_regexp>'` | Como la anterior pero utilizando expresiones PCRE |
| `-N '<grupo>'` | Utilizando un grupo definido a través de los ficheros de configuración en el maestro. |
| `-I '<key_pillar:valor>'` | Utiliza la información "pillar" del equipo |
| `-S '<subred>'` | Utiliza la información de red del equipo. |

Ejemplo: `salt -G 'os:Debian' test.version`

A través de estos comandos puede realizarse todo tipo de operativas de gestión de la configuración en los equipos administrados, en los anexos se incluyen algunas de las llamadas que pueden resultar más útiles en el día a día del administrador a modo de referencia.

### Instalación de aplicaciones

Como se ha visto en el apartado administración software, para realizar la instalación de una aplicación en un equipo administrado puede realizarse de múltiples formas, aquí se enumeran las más convenientes.

#### Instalación de aplicación a través de un estado de Salt.

Para realizar una instalación de software a través de un estado Salt deberá ser creado un fichero estado (SLS) que sea aplicado en el equipo administrado, bien porque está enlazado a través del "highstate" o bien porque está en un fichero SLS "huerfano" y ha sido asignado a través de la interfaz Foreman, bien al host o bien al grupo que pertenezca el host.

Por ejemplo para instalar el paquete "vim" se crearía el fichero /stv/salt/vim.sls con el contenido:

```
install_vim:
      pkg.installed:
          - pkgs:
              - vim
```

o con la nomenclatura más breve:

```
vim:
      pkg.installed
```

Si el fichero de estado es aplicado a través del "highstate" o está incluido en los estados Salt asignados por algún método, en la próxima invocación del "highstate" será instalado automáticamente.

#### Instalación de una aplicación a través de un módulo salt de forma interactiva

Puede ser instalada una aplicación en un determinado equipo sin necesidad de confeccionar un fichero de estado, útil si es una acción aislada puntualmente que no debe replicarse a otros ordenadores de la infraestructura.

Puede realizarse una llamada desde el servidor tal cual la siguiente:

```
salt '<selector>' pkg.install <nombre_paquete>
```

Obteniéndose una salida con el éxito o fallo de la operación.

Aunque menos útil, también puede ser realizada la siguiente llamada equivalente desde el propio ordenador administrado remotamente (no obstante requiere permisos de administración y con estas condiciones posiblemente sea más sencillo utilizar las herramientas propias del sistema operativo)

```
salt-call pkg.install <nombre_paquete>
```

#### Instalación de aplicaciones a través de la ejecución de un comando.

De forma generalista siempre pueden encontrarse varias formas de realizar la misma tarea.

En el caso de Salt y la ejecución de módulos, si existe una herramienta en la plataforma que está siendo administrada remotamente y se tiene la certeza que todos los equipos que el selector cumplen los requisitos necesarios para ejecutar un comando de sistema operativo, puede realizarse una llamada equivalente, bien a través de un estado como a través de la línea de comandos.

A través de un estado lanzando el módulo ejecutable `cmd.run`:

```
install_vim:
      cmd.run:
         -  name: apt-get install -y vim
```

O equivalentemente de forma interactiva:

```
salt '<selector>' cmd.run 'apt-get install -y vim'
```

### Eliminación de aplicaciones

La eliminación del software instalado en una máquina es muy similar al proceso de instalación, de forma análoga se utilizará el método "remove" o "removed" en vez de "install" o "installed".

Para la eliminación de aplicaciones quizás es más conveniente realizar llamadas interactivas que tener una definición en ficheros de estado.

#### Eliminación de aplicaciones a través de un estado de Salt

Para eliminar el paquete "vim" se crearía el fichero /stv/salt/remove_vim.sls con el contenido (haciendo uso de módulo estado y la definición compacta, también es válida la nomenclatura similar a el caso de instalación):

```
vim:
      pkg.removed
```

#### Eliminación de aplicaciones a través de un módulo salt de forma interactiva

Puede realizarse una llamada desde el servidor tal cual la siguiente:

```
salt '<selector>' pkg.remove <nombre_paquete>
```

Obteniéndose una salida con el éxito o fallo de la operación.

### Eliminación de aplicaciones a través de la ejecución de un comando.

A través de un estado lanzando el módulo ejecutable `cmd.run`:

```
remove_vim:
      cmd.run:
         -  name: apt-get remove -y vim
```

O equivalentemente de forma interactiva:

```
salt '<selector>' cmd.run 'apt-get remove -y vim'
```

### Actualizar una aplicación a una versión en concreto

Actualizar una aplicación a la última versión consiste simplemente es replicar el proceso de instalación de la aplicación, no obstante cuando se desea que una aplicación esté instalada o se instale utilizando una versión en concreto de las disponibles, debe hacerse uso del parámetro que especifica una versión, de no utilizar el parámetro siempre trata de realizar la instalación más moderna disponible.

A través de un estado:

```
clamav-docs:
      pkg.installed:
         -  version: 0.103.6*
```

De forma interactiva:

(se puede apreciar como también de forma interactiva se puede utilizar el paso de parámetros nombrando los parámetros)

```
salt 'a58' pkg.install name='clamav-docs' version='0.103.6*'
```

Con un comando de sistema operativo, como se puede apreciar en este caso no puede ser aplicado un "shell glob" siendo más tedioso de utilizar:

```
salt 'a58*' cmd.run 'apt-get install -y clamav-docs=0.103.7+dfsg-0+deb11u1'
```

### Actualizar una todas las aplicaciones del sistema

Cabe hacer notar que actualizar todas las aplicaciones del sistema y en menor medida una aplicación por separado, puede requerir introducir información de forma interactiva (p.ej: preguntas debconf obligatorias), estos requerimientos deben ser tenidos en cuenta (p.ej: respuestas preseed) porque pueden dejar el proceso de actualización a medias y terminar dando error.

Para actualizar todas las aplicaciones del sistema debe utilizarse la función "upgrade" del módulo "pkg" (aptpkg.py).

A través de un estado:

```
pkg.upgrade:
    module.run
```

De forma interactiva:

```
salt 'a58*' pkg.upgrade
```

Con comando de sistema operativo

```
salt 'a58*' cmd.run 'apt-get -y upgrade'
```

## Administración del sistema operativo

### Realizar una actualización de todo el sistema operativo a otra release

Para realizar una actualización a otra versión del sistema operativo deberían darse las siguientes condiciones:

- El sistema operativo debe permitir la actualización a través de sus propias herramientas, es un proceso delicado que no debería realizarse de forma desatendida porque pueden surgir múltiples problemas o particularidades.
- No debe requerir intervención humana
- No debe requerir reinicios durante el proceso

Para realizar una actualización del sistema operativo, en el caso que fuera utilizada la tecnología Salt, se debería utilizar las llamadas a comandos de sistema operativo "cmd.run" que ofrece Salt para realizar el proceso con el comando de sistema operativo adecuado.

### Modificación de ficheros

Para la explicación del uso de operaciones de lectura/escritura se evitará realizar las operaciones con comandos de sistema operativo que aunque serían posibles deberían evitarse en la medida de lo posible.

#### Lectura

Para realizar la lectura de ficheros generalmente se utilizará una llamada interactiva a petición de un administrador de la red administrada.

Para realizar la lectura de un fichero en un equipo administrado se utilizará la siguiente llamada interactiva:

```
salt <selector> file.read /camino/al/fichero
```

#### Búsqueda

Si es necesario listar el contenido de un directorio, será necesario realizar la llamada a través del módulo "cmd".

```
salt <selector> cmd.run 'ls -l /camino/al/directorio'
```

#### Escritura

Desde la definición de un estado:

```
file.write:
    module.run:
        - path: /tmp/coco
        - args:
            - uno
            - dos
```

Un uso más avanzado realizando una copia desde el servidor y utilizando el motor de plantillas Jinja podría ser:

```
/etc/apt/sources.list:
      file.managed:
         -  source: salt://configuraciones/etc/apt/sources.list
         -  template: jinja
```

Siendo el fichero de plantilla Jinja:

```
{%- set mirror = salt['pillar.get']('repositorios:llx_mirror', 'llxmirror.org') %}
{%- set distro = grains.lsb_distrib_codename %}
deb http://{{ mirror }}/llx {{ distro }} main
```

Desde llamadas interactivas en línea de comandos:

```
salt <selector> file.write /tmp/ejemplo_una_linea “JAJA”
salt <selector> file.write path="/tmp/ejemplo_multilinea"
      args="['uno','dos','tres']" (múltiples líneas)
```

### Copia

Una opción más interesante que la escritura "manual" de ficheros puede ser la copia desde el servidor manejando siempre ficheros sometidos a un control de versiones.

Para realizar la copia de ficheros desde el servidor puede realizarse de la siguiente forma:

De forma interactiva:

```
salt-cp <selector> /path/fichero/origen /path/fichero/en/destino
```

Copia de ficheros a través de un estado:

```
copia_ficheros:
    file.recurse:
        - name: "/tmp/configuraciones"
        - source: salt://configuraciones
        - makedirs: True
        - replace: True
        - clean: True

copia_fichero:
    file.managed:
        - name: /tmp/config.txt
        - source: salt://config.txt
        - makedirs: True
```

## Ejecución de comandos

Para la ejecución de comandos de sistema operativo son utilizadas las funciones del módulo de ejecución "cmd" como se ha visto anteriormente para otros usos, aquí se presenta un uso un poco más avanzado que utiliza otras funcionalidades que ofrece Salt a modo de introducción a requerimientos, condiciones o dependencias que puede presentar un estado.

```
mi_script:
      cmd.run:
            - name: /ruta/a/mi_script
            - cwd: /
            - stateful: True

Ejecutado cuando cambia algo mi_script:
      cmd.wait:
            - name: echo Algo ha cambiado
            - cwd: /
            - watch:
                  - cmd: mi_script
```

En este ejemplo puede verse cómo se utilizan las funciones "run" y "wait", las cuales presentan las siguientes características:

- "run" se ejecuta siempre que es aplicado el fichero de estado
- "wait" sólo es ejecutado cuando otro estado ha cambiado, está pensado para vigilar los cambios de otros estados.

Otro ejemplo que muestra estas características es:

```
/usr/local/bin/postinstall.sh:
      cmd:
            - wait
            - watch:
                  - pkg: mycustompkg
      file:
            - managed
            - source: salt://utils/scripts/postinstall.sh

mycustompkg:
      pkg:
            - installed
            - require:
                  - file: /usr/local/bin/postinstall.sh
```

Aquí se muestra como se fija un estado para instalar el paquete "mycustompkg" pero se establece una dependencia ("require") para que exista el fichero "/usr/local/bin/postinstall.sh" el cual es obtenido desde el repositorio público del servicio salt en el servidor maestro y una vez realizada la instalación ("watch") será disparada la ejecución del script "postinstall.sh".

## Dependencias o relaciones entre estados Salt

Información extendida con más ejemplos: https://docs.saltproject.io/en/latest/ref/states/requisites.html

Cuando son configurados servicios a menudo es necesario realizar diversas acciones como puede ser completar o modificar ficheros de configuración, realizar instalación de paquetes, modificar el estado por defecto del servicio y cambiar el modo de ejecución del servicio.

Estas acciones son realizadas a través de estados Salt en ficheros de plantillas SLS, no obstante es necesario que sigan un orden para una configuración funcional, por ejemplo: no tiene sentido tratar de arrancar un servicio que no está instalado o no está configurado previamente.

Una configuración sencilla que crea una dependencia ("require") de un paquete instalado antes de tratar de arrancar el servicio puede ser:

```
nginx:
      pkg.installed:
            - name: nginx-light
      service.running:
            - enable: True
            - require:
                  - pkg: nginx
```

Esta dependencia puede presentar una forma alternativa que invierte el sentido de la dependencia (util en bucles) concatenando "_in" a la palabra clave que está definiendo el requisito, ejemplo "require_in" o "watch_in".

Cuando existen ficheros de configuración la anterior definición puede ampliarse así:

```
nginx:
      pkg.installed: []
      file.managed:
            - name: /etc/nginx/nginx.conf
      service.running:
            - enable: True
            - require:
                  - nginx
                  - file: /etc/nginx/nginx.conf
```

Ahora hay dos dependencias tanto del paquete instalado como de la existencia del fichero de configuración, también se muestra como la definición del módulo de estado ("pkg") en los requisitos es opcional.

Si el fichero de configuración es modificado, el servicio debería ser ejecutado de nuevo para que la configuración surta efecto, por ello puede utilizarse una relación para vigilar los cambios ("watch") en el fichero (o conjunto de ficheros, ya que soporta "shell globs")

```
apache2:
      service.running:
            - watch:
                  - file: /etc/apache2/*
```

Para explicar y mejorar el entendimiento de cómo funciona la tecnología Salt y el compilador de estados se puede partir del siguiente ejemplo donde un fichero obtenido por red es descomprimido:

```
Baja fichero al servidor:
      file.managed:
            - name: /configuraciones/confs.tgz
            - source: salt://configs/confs.tgz
            - makedirs: True
            - replace: True

Descomprime fichero configs:
      archive.extracted:
            - name: /tmp/descomprimido
            - source: /configuraciones/confs.tgz
            - archive_format: tar
            - clean: True
            - enforce_toplevel: False
            - onchanges:
                  - file: Baja fichero al servidor
```

Esto puede escribirse de una forma más compacta:

```
Baja fichero al servidor:
      file:
            - managed
            - name: /configuraciones/confs.tgz
            - source: salt://configs/confs.tgz
            - makedirs: True
            - replace: True

Descomprime fichero configs:
      archive:
            - extracted
            - name: /tmp/descomprimido
            - source: /configuraciones/confs.tgz
            - archive_format: tar
            - clean: True
            - enforce_toplevel: False
            - onchanges:
                  - file: Baja fichero al servidor
```

Las dependencias también pueden realizar el match de la especificación tanto con el identificador del estado como con el parámetro "name" consiguiendo una sintaxis más clara.

```
Baja fichero al servidor:
      file:
            - managed
            - name: /configuraciones/confs.tgz
            - source: salt://configs/confs.tgz
            - makedirs: True
            - replace: True

Descomprime fichero configs:
      archive:
            - extracted
            - name: /tmp/descomprimido
            - source: /configuraciones/confs.tgz
            - archive_format: tar
            - clean: True
            - enforce_toplevel: False
            - onchanges:
                  - file: /configuraciones/confs.tgz
```

Salt presenta los siguientes tipos de relaciones:

| | |
|---|---|
| "require" | Previene la ejecución del estado hasta que todos los estados dependientes se ejecutan satisfactoriamente. |
| "onchanges" | Ocasiona que el estado solo sea aplicado si el estado relacionado presenta cambios y si el resultado del estado "result" tiene el valor de "True" |
| "watch" | Utilizado para aportar un comportamiento adicional monitoreando cambios en otros estados |
| "listen" | Utilizado de forma similar a "watch" pero ejecutándose al final de la ejecución del estado relacionado. |
| "prereq" | Utilizado de forma similar a "onchanges" pero utilizando resultado "test=True" del estado relacionado que observa. |
| "onfail" | Permite reacciones a un estado con ejecución fallida |
| "use" | Utilizado para heredar los argumentos provistos en otra declaración de estado |

Algunos como "require", "watch", "onchanges" y "onfail" presentan modificadores como "require_any", "watch_any", "onchanges_any" y "onfail_any" para establecer múltiples relaciones de forma condicional, también se presenta "onfail_all" para establecer una condición más amplia.

Para una explicación más detallada y ejemplos de todos los tipos de relaciones consultar documentación oficial.

Los estados pueden ser alterados para asegurar que los estados son evaluados exactamente como el administrador espera o asegurar las condiciones que deben cumplirse para que sean ejecutados.

Las alteraciones del comportamiento son conseguidas a través de las siguientes claves:

| | |
|---|---|
| reload | Compuesto por "reload_modules", "reload_pillar", "reload_grains" fuerza la recarga de módulos después que el estado ha sido ejecutado. |
| "unless" | Especifica que el estado solo debe ser ejecutado en caso que un comando de sistema operativo especificado devuelva False (0) (ejecución correcta), a través del parámetro "fun" puede especificar un módulo de ejecución junto con más parámetros que serán pasados al módulo de ejecución. |
| "onlyif" | Especifica requisitos para ejecutar comandos en los que todos deben devolver "True" para que el estado sea ejecutado (0) (ejecución correcta) |
| "creates" | Especifica un estado que solo debe ejecutarse cuando cualquiera de los ficheros no exista. |
| "runas" | Específica globalmente como fijar el usuario que ejecuta un comando a través de "cmd.run". |
| "runas_password" | Específica globalmente la contraseña del usuario a utilizar cuando se utiliza la alteración "runas". |
| "check_cmd" | Es utilizado un comando para determinar si el estado ha terminado (o no) con éxito. |

Los estados también pueden realizar notificaciones de eventos cuando son completados a través de la opción "fire_event".

También puede ser establecida una política de reintentos cuando un estado es ejecutado y no siempre puede completarse exitosamente, para utilizar la política de reintentos es utilizada la clave "retry" modificada por los siguientes argumentos:

| | |
|---|---|
| attempts | Fija el número de reintentos |
| until | Fija el resultado que evitará que sean realizados nuevos intentos |
| interval | Cantidad de tiempo en segundos a esperar entre reintentos |
| splay | Fija una aleatorización del intervalo añadiendo un valor aleatorio fijado |

Los valores que produce un estado durante la realización de reintentos y que son devueltos son:

| | |
|---|---|
| result | Representa el resultado de la última ejecución |
| started | Representa el resultado de la primera ejecución |
| duration | Representa el tiempo de ejecución de todos los reintentos junto con los intervalos de espera |
| comment | Incluye el resultado "result" y comentarios de todos los intentos |

### Control sobre el origen de las fuentes del software instalado

Para la gestión de los repositorios sobre las máquinas administradas pueden ser utilizados los estados que provee el módulo de estado "pkgrepo" o bien realizar la escritura o borrado del fichero a través de las funciones de otros módulos.

La invocación de estos módulos no es posible conseguirla fácilmente desde línea de comandos, por lo cual es recomendable ser utilizada desde ficheros de estado, además es una información muy estática y fijada por la organización que permite ser utilizada óptimamente de forma no interactiva.

Un fichero de estados para el control de repositorios podría tener este aspecto:

```
apt-transport-https:
      pkg.installed

apt-get-update:
      cmd.wait:
            - name: apt-get update

/etc/apt/sources.list:
      file.managed:
            - source: salt://system/files/etc/apt/sources.list
            - template: jinja
            - watch_in:
                  - cmd: apt-get-update

docker-repo:
      pkgrepo.managed:
            - humanname: Official Docker Repository
            - name: deb [arch=amd64] https://download.docker.com/linux/debian {{
grains.lsb_distrib_codename }} stable
            - file: /etc/apt/sources.list.d/docker.list
            - key_url: https://download.docker.com/linux/debian/gpg
            - refresh: False
            - watch_in:
                  - cmd: apt-get-update
```

Mediante un fichero de estado similar se puede especificar los ficheros de fuentes de repositorio que deben estar en la máquina manteniendo actualizado el catálogo software disponible.

Cuando se realiza la instalación de paquetes nuevos a través del módulo pkg.install existe el parámetro "fromrepo" que permite especificar qué repositorio de software debe ser utilizado para realizar la instalación.

```
dotdeb.repo:
      pkgrepo.managed:
            - name: deb http://packages.dotdeb.org wheezy-php55 all
            - dist: wheezy-php55
            - file: /etc/apt/sources.list.d/dotdeb.list
            - keyid: 89DF5277
            - keyserver: keys.gnupg.net
            - refresh_db: true

php.packages:
      pkg.installed:
            - fromrepo: wheezy-php55
            - pkgs:
                  - php5-fpm
                  - php5-cli
                  - php5-curl
```

## Agrupación de los equipos administrados en grupos

Cuando deben ser realizadas acciones sobre multitud de equipos casi siempre existen conjuntos de equipos que comparten características y configuración, cuando son aplicadas acciones deben ser aplicadas por igual a todos los miembros de este conjunto. Foreman permite realizar una agrupación de equipos para poder ser referenciado el nombre del grupo como si de cualquier otro parámetro se tratara para poder realizar acciones sobre ellos.

En combinación de la tecnología Salt, esta forma de realizar acciones sobre el conjunto de equipos que forma el grupo puede aplicarse de dos formas:

- Asignando al grupo determinados estados que serán aplicados conforme vayan ejecutando la sincronización con el servidor.
- Forzando un trabajo desde Foreman y aplicándolo sobre los equipos que cumplen el criterio del nombre de grupo que es introducido durante la creación del trabajo.

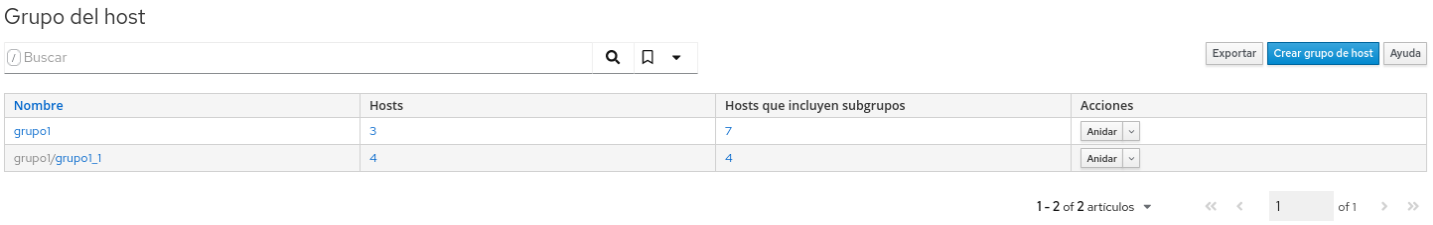
Es importante reseñar que Foreman permite que un equipo sea asignado a un y solo un grupo, un equipo no puede estar asignado en dos grupos simultáneamente.

Foreman permite también el concepto de subgrupo, es decir un equipo puede estar asignado a un y solo un grupo, en el caso que sea un subgrupo, seguirá siendo la única asignación de grupo que podrá disponer un equipo, no obstante existirá cierta herencia de determinados atributos que son aplicados a todos los equipos del subgrupo heredadas desde los atributos asignados desde el grupo padre de orden superior.

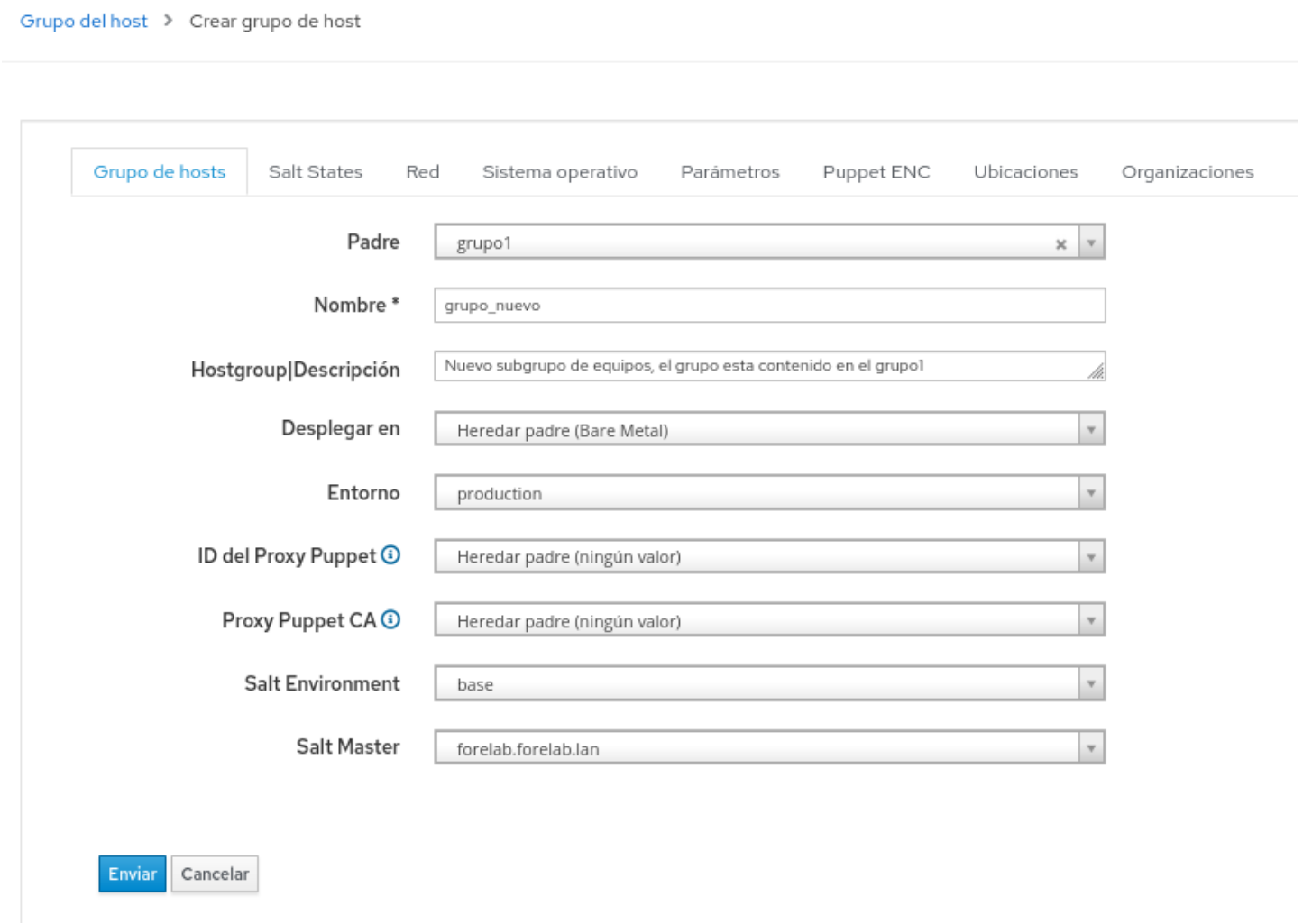
Los subgrupos son expresados en la interfaz de Foreman como "nombre_grupo/nombre_subgrupo…" esto corresponde desde el punto de vista de un objeto equipo asignado con subgrupo con atributos internos como "hostgroup" = "nombre_subgrupo" y "hostgroup_fullname" = "nombre_grupo/nombre_subgrupo".

## Crear un grupo o subgrupo en Foreman

Para crear un nuevo grupo en Foreman es necesario acceder a través de la página "Configurar → Grupo de hosts" donde se muestra un listado de los grupos/subgrupos existentes y a través de los botones en la parte superior derecha utilizar "Crear grupo de host".

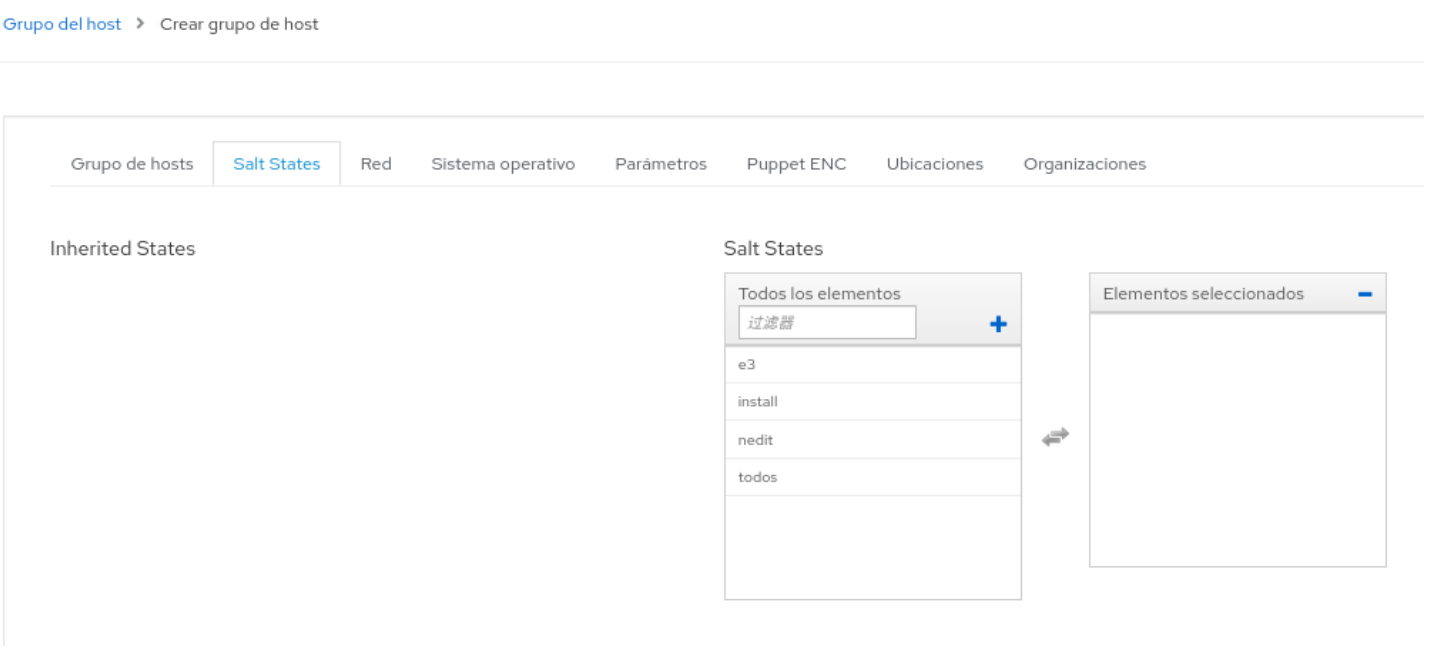



Cuando se está creando un grupo nuevo se ofrece la acción de establecer un grupo padre y los datos particulares que se establecerán para dicho grupo nuevo, es importante seleccionar correctamente los valores específicos relacionados con la tecnología Salt.

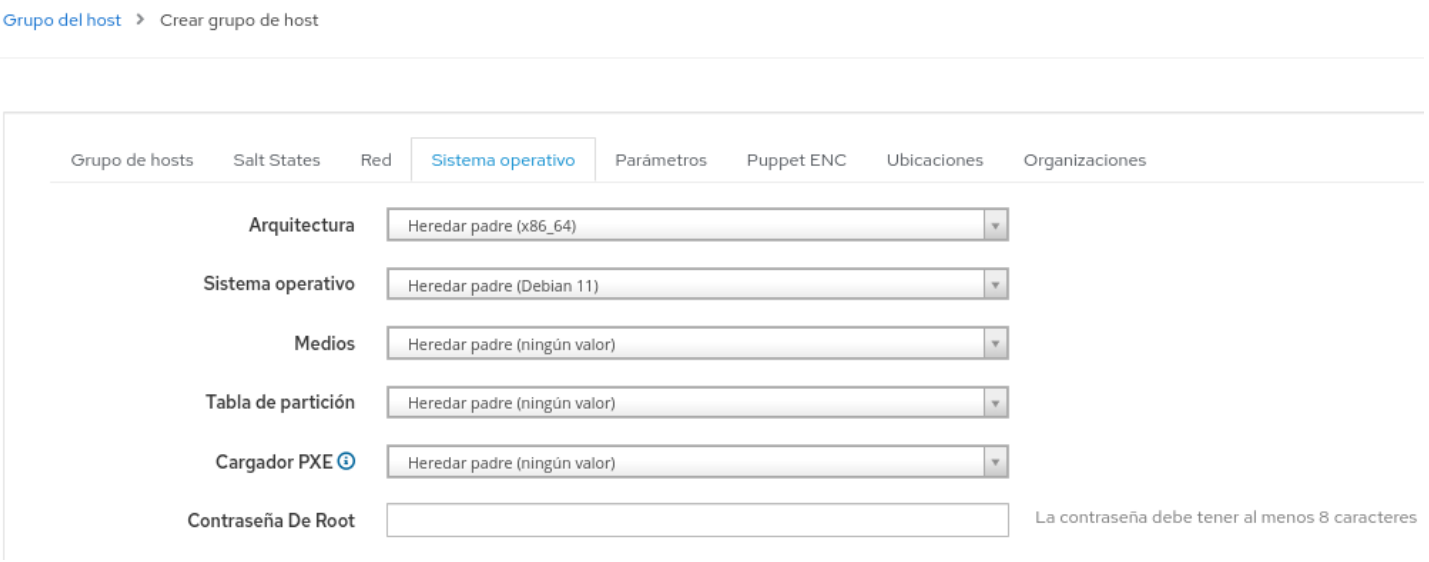



A través de las pestañas que ofrece la página son configurados los estados Salt que tendrá asignados el grupo, si los estados son creados posteriormente o son reliazadas modificaciones mientras el grupo ya esta creado y asignado deberá editarse posteriormente el grupo para actualizar dichos cambios en estados Salt.

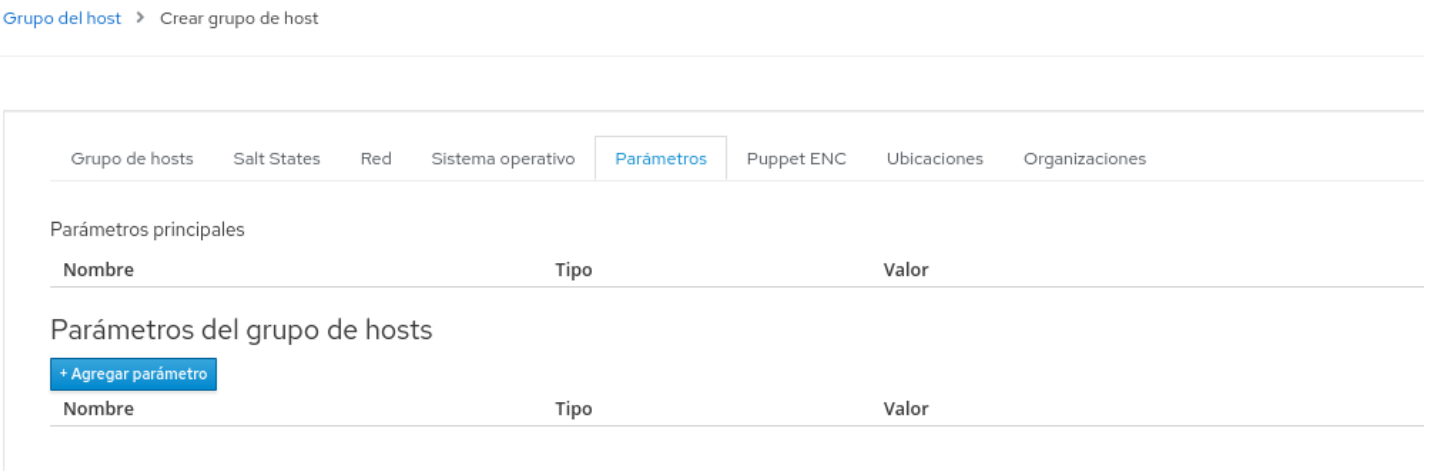
Si un grupo hereda de otro (subgrupo) cuando están siendo seleccionados los estados Salt, aparece en la parte izquierda de la página los estados que están asignados al grupo de orden superior y que serán aplicados, no permitiendo la asignación de estos estados solo los restantes.



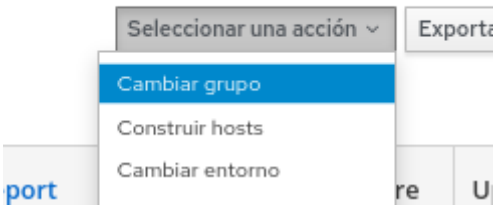

Se seleccionan y asignan los valores de sistema operativo, aunque no todos los ajustes son relevantes debido a que no son equipos que serán desplegados por la herramienta desde "bare metal".



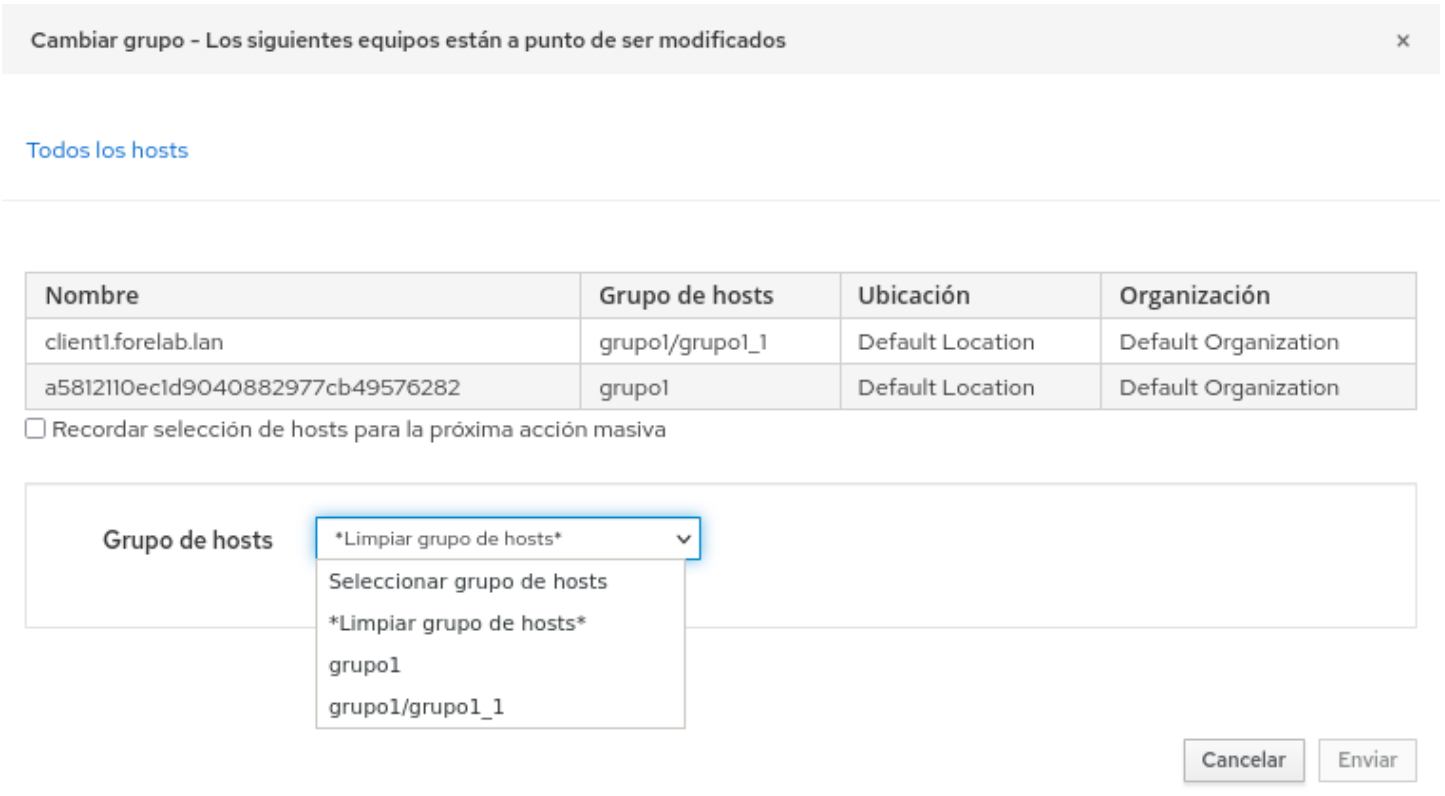

También es posible establecer parámetros Foreman (se recuerda que conformarán información de "pillar") para los equipos del grupo.




Posteriormente deberán de editarse los "host" para fijar la pertenencia del equipo a un grupo en concreto uno a uno, o en caso de ser muchos puede utilizarse un criterio de busqueda y a través de las casillas marcarlos y aplicar una acción múltiple desde el selector de acciones




Para eliminar del grupo asignado a un "host" o múltiples pueden realizarse las mismas acciones pero vaciando el campo de grupo



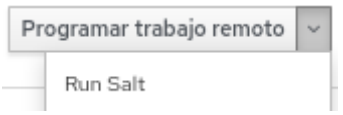

Si se desea eliminar el grupo deberá procederse desde la página "Configurar → Grupos de hosts".

## Ejecución de acciones sobre el grupo

Como se ha nombrado anteriormente existen dos formas de realizar acciones sobre los equipos que conforman un grupo/subgrupo, a través de la asignación de estados Salt o bien invocar un trabajo remoto que es aplicado con determinados criterios.

A través de la edición de las propiedades de un grupo/subgrupo o un host pueden ser especificados los estados que son asignados a los equipos, estos estados se aplicarán en la siguiente sincronización que realiza el equipo cliente, es la forma más sencilla de realizar acciones utilizando la tecnología Salt.

Además, por defecto a través de la página que muestra la información de un "host" existe la posibilidad de ejecutar una definición de un trabajo para ese equipo en concreto ("Programar trabajo remoto"), junto con la posibilidad de ejecutar el "highstate" de Salt (se recuerda que hay que especificar cual es la plantilla asociada a este botón desde las propiedades de "Administrar → Remote Execution Features").

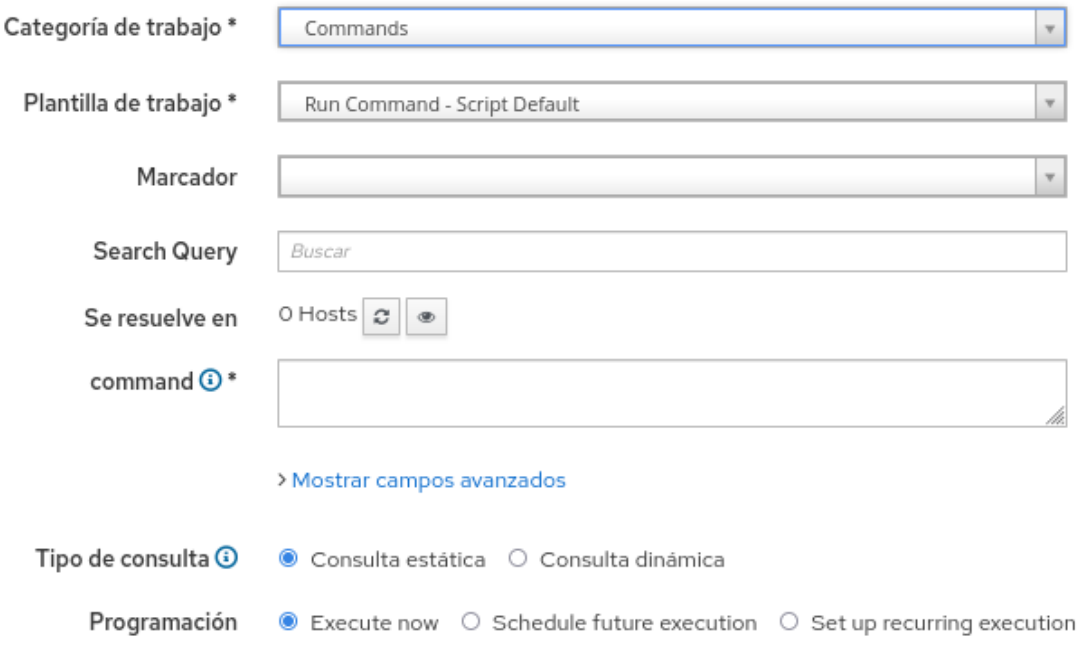



De forma análoga a través de la página "Monitor → Jobs" se pueden ver las ejecuciones de trabajos permitiendo definir un trabajo y que sea aplicado a un conjunto de "hosts" aplicando ciertos criterios.

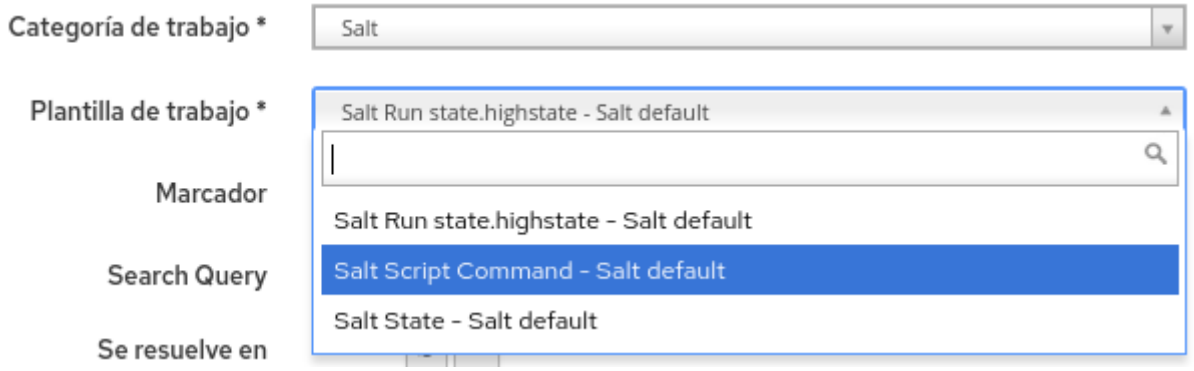

En resumen las definiciones para lanzar trabajos pueden configurarse desde:

- Monitor → Jobs → "Ejecutar trabajo"
- Hosts → Todos los host → (marcar casillas) → "Programar trabajo remoto"
- Hosts → Todos los host → (aplicar un filtrado p.ej: "hostgroup") → (marcar todas las casillas) → "Programar trabajo remoto"
- Desde un host → "Programar trabajo remoto"

Cuando se está configurando una tarea se presenta el siguiente formulario:



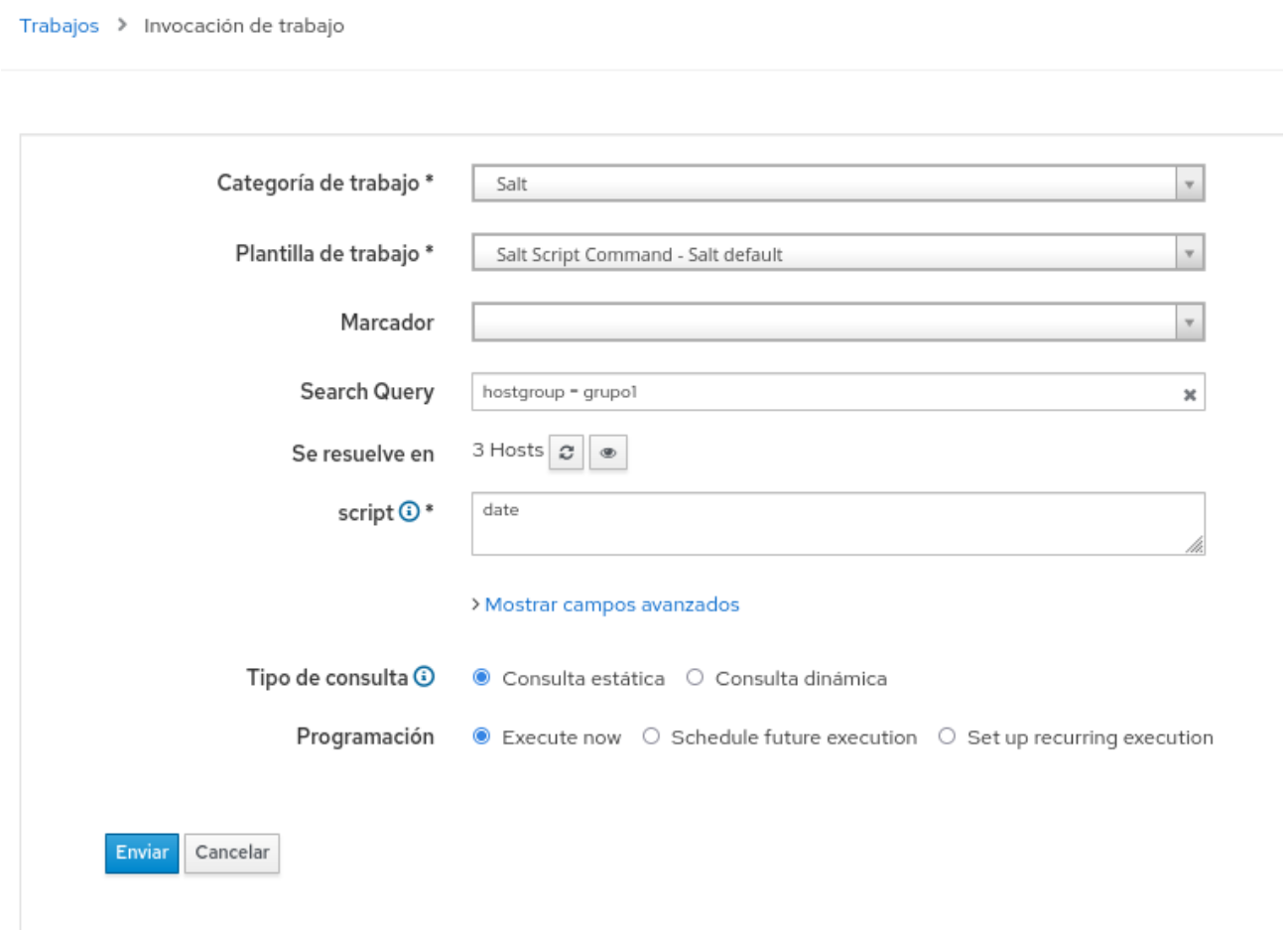
Los campos a completar dependen de la "categoría del trabajo" para este proyecto y teniendo en cuenta que no es posible utilizar capacidades de ejecución remota a través de SSH, será necesario seleccionar la categoría Salt donde por defecto se ofrecen las siguientes plantillas para lanzar trabajos:



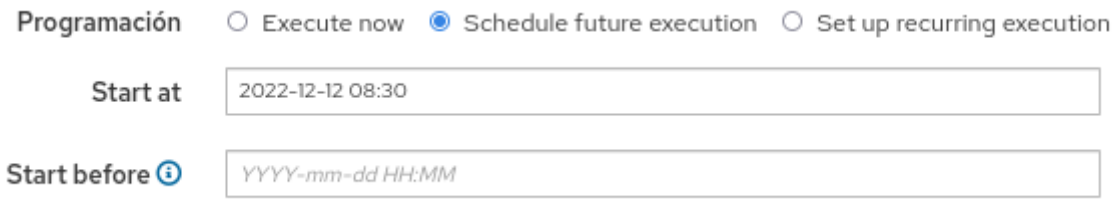

- La primera opción mostrada se corresponde en un lanzamiento para aplicar el estado "highstate".
- La segunda opción mostrada permite ejecutar a través del módulo Salt "cmd.run" un comando de sistema operativo pasado como parámetro.
- La tercera opción mostrada permite definir un estado completo Salt a ejecutar pasado como parámetro, utilizando el módulo "state.template_str".

### Definiendo un trabajo con plantilla para ejecutar un comando de sistema operativo

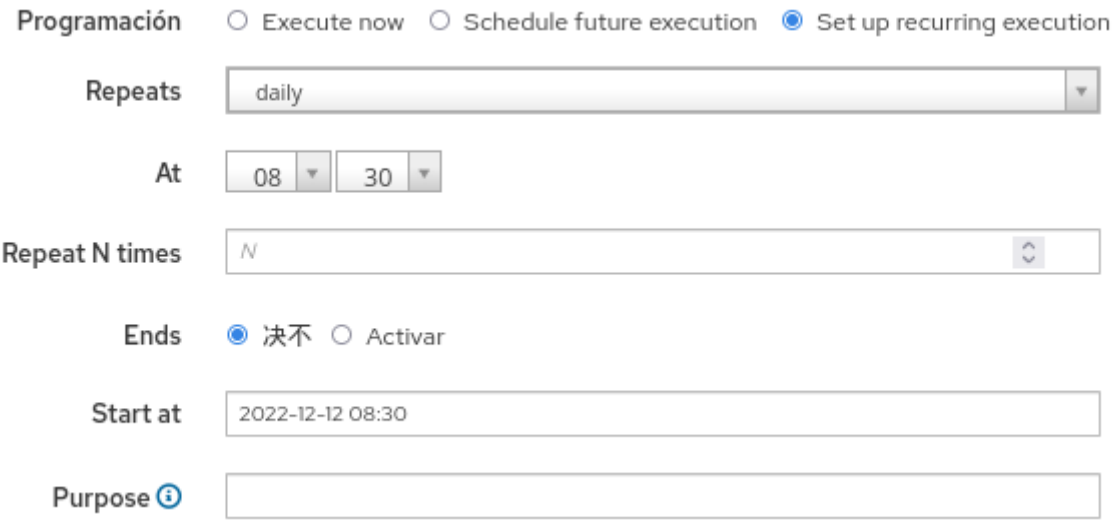
Utilizando la opción para ejecutar sobre un grupo utilizando la plantilla "Salt Script Command" quedará algo como lo siguiente:



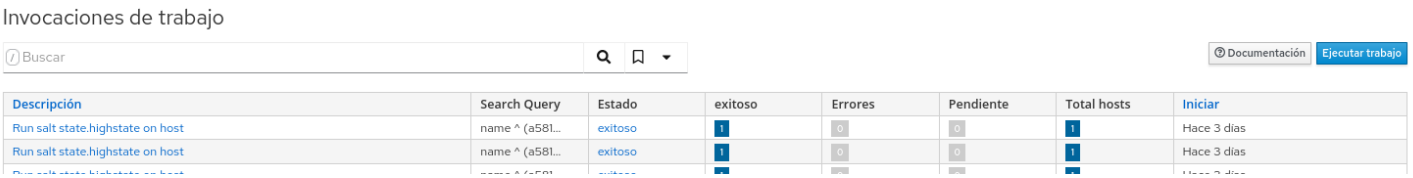
Permitiendo opciones comunes para realizar una programación y que no sea ejecutada inmediatamente:



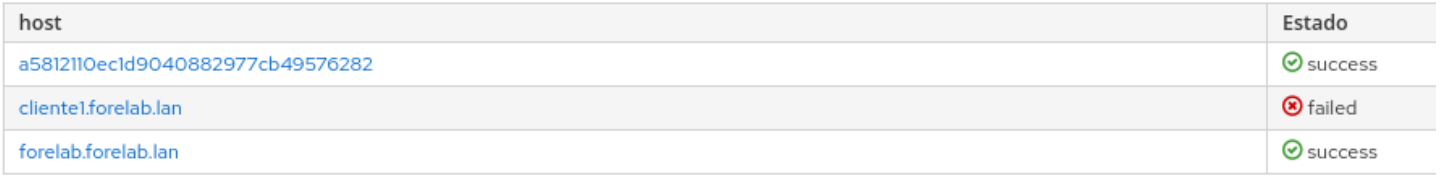

o de forma reiterativa:




Cuando las acciones son ejecutadas, puede observarse el resultado a través de la página de trabajos y seleccionando el trabajo




En cualquier tipo de trabajo cuando es ejecutado por el método que sea puede accederse al detalle en la parte inferior de la página de trabajo mostrando el éxito o fracaso de la operación individualizado para cada "host" en el que ha sido aplicada la tarea.

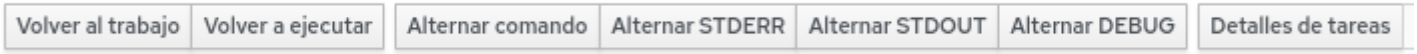


Seleccionando el "host" se muestra la salida de la ejecución realizada:

Destino: a5812110ec1d9040882977cb49576282 using Smart Proxy forelab.forelab.lan

```
jid: 20221212073947809855
a5812110ec1d9040882977cb49576282:
----------
          ID: execute script
    Function: cmd.run
        Name: date
      Result: True
     Comment: Command "date" run
     Started: 08:39:50.742094
    Duration: 8.742 ms
     Changes:
              ----------
              pid:
                  1829
              retcode:
                  0
              stderr:
              stdout:
                  Mon Dec 12 08:39:50 CET 2022

Summary for a5812110ec1d9040882977cb49576282
------------
Succeeded: 1 (changed=1)
Failed:    0
------------
Total states run:     1
Total run time:   8.742 ms
Estado de salida: 0
```

Presentando las opciones relacionadas con el trabajo:



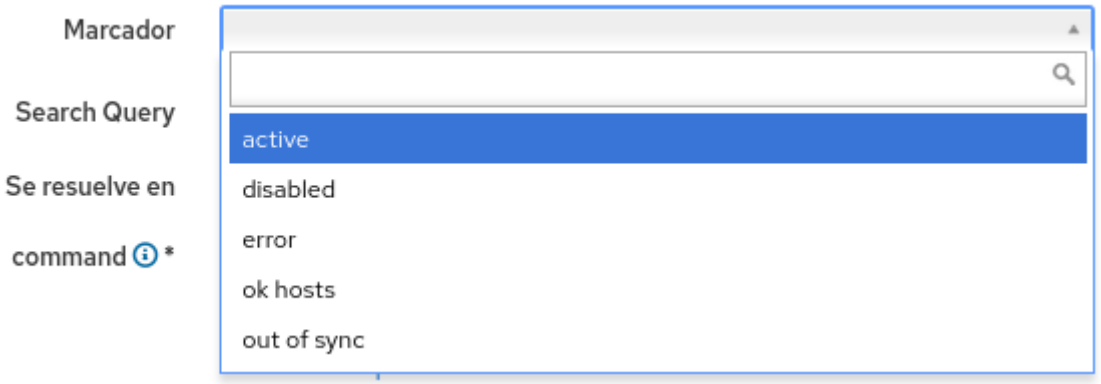

### Particularidades definiendo acciones para grupos/subgrupos

Aplicar acciones sobre grupos tiene alguna peculiaridad cuando son utilizados subgrupos junto con la tecnología Salt para aplicar acciones sobre equipos de un grupo/subgrupo:

- Cuando son aplicados estados: Los equipos heredan los estados que hayan sido aplicados al grupo padre y solo ofrecen para asignar los estados restantes menos generalistas.
- Cuando se aplica un trabajo aplicando un criterio de grupo ("hostgroup") cobra especial importancia el operador que sea utilizado.
    - Si es utilizado el operador "=" con el valor del nombre de subgrupo puede que solo sean seleccionados los miembros del subgrupo dejando sin aplicar la acción a los restantes miembros del grupo superior que no están incluidos en el subgrupo.
    - Si es utilizado el operador "=" con el valor del nombre de grupo puede que solo sean seleccionados los miembros del grupo dejando sin aplicar la acción a los miembros del subgrupo.
    - Pueden ser añadidos conjuntos utilizando expresiones lógicas como "and" para realizar una agrupación en la selección de objetivos del trabajo.
- Cuando se aplica un trabajo aplicando un criterio de grupo ("hostgroup_fullname"):
    - Si es utilizado el operador "~" con el valor del grupo de orden superior pueden ser seleccionados todos los elementos que son contenidos tanto en el grupo como en el subgrupo.

En cualquier caso es de especial importancia mientras se está definiendo un trabajo a realizar en un conjunto de equipos observar a cuantos equipos será aplicado el trabajo a través de los botones que aparecen en la página de invocaciones de trabajo.


El primer botón  actualiza el contador de elementos que es mostrado indicando el número de equipos objetivo mientras que el segundo botón  mostrará en una ventana el FQDN de los equipos objetivo.
Para facilitar la tarea, puede ser utilizada la opción de "Marcador" para seleccionar un criterio y no tener que definir un criterio de búsqueda complejo, permitiendo elegir entre los criterios almacenados, por ejemplo "activo".




Los criterios mostrados pueden ser visualizados a través de la página "Administrar → Marcadores" donde aparece un listado con todos los criterios de búsqueda existentes (tanto para realizar definiciones de trabajo como para búsquedas filtradas sobre la lista de equipos) siendo posible:

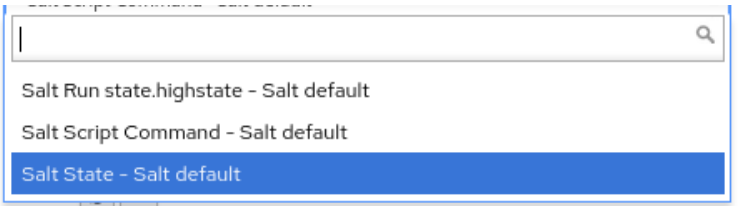
- crear nuevos marcadores que se ajusten a las necesidades del proyecto.
- ver el detalle de los existentes

Marcadores

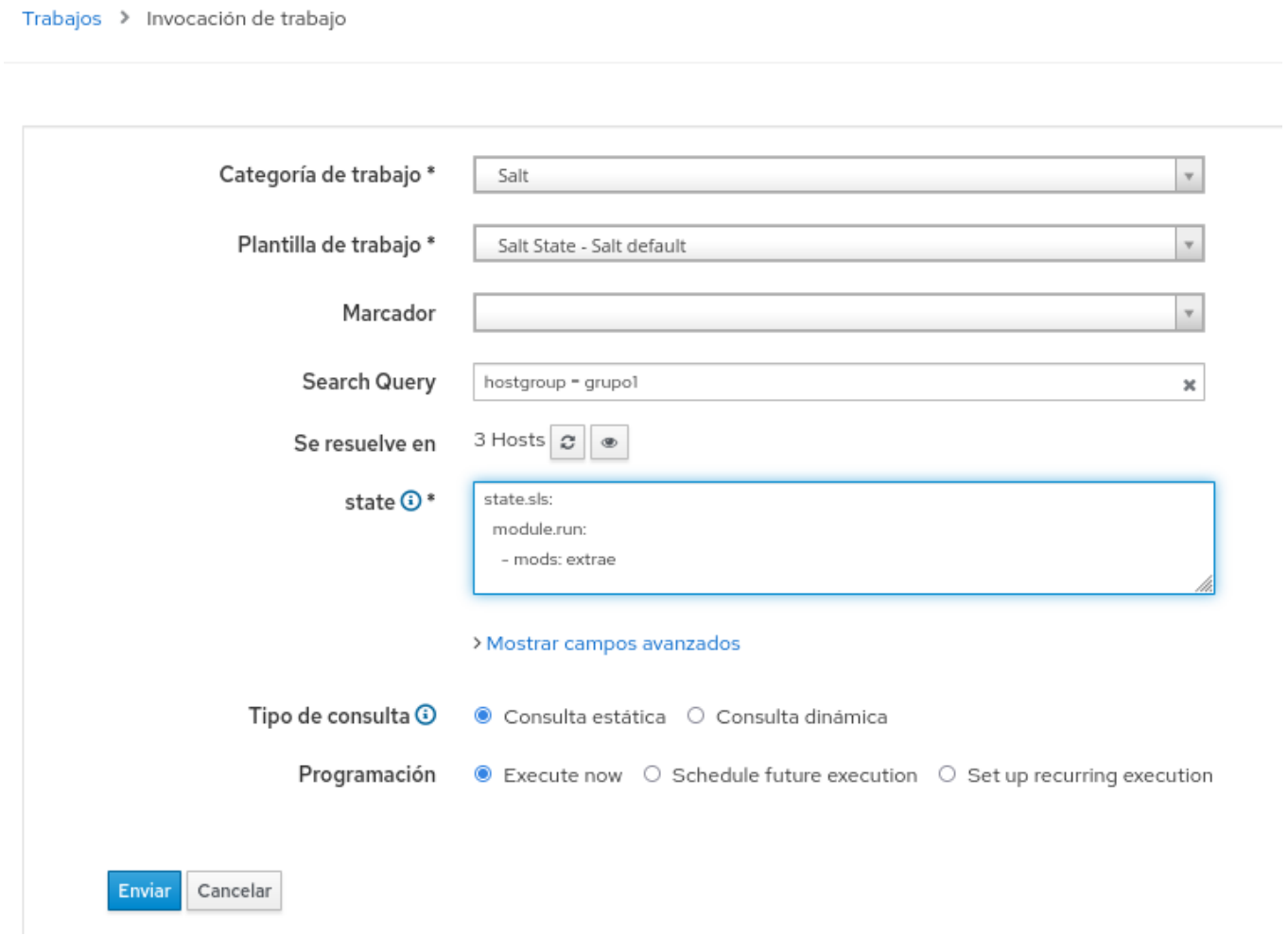
Buscar

| Nombre | Consulta | Controlador | Público | Acciones |
|---|---|---|---|---|
| active | last_report > "35 minutes ago" and (status.applied > 0 or status.restarted > 0) | hosts | true | Borrar |
| disabled | status.enabled = false | hosts | true | Borrar |
| error | last_report > "35 minutes ago" and (status.failed > 0 or status.failed_restarts > 0 or status.skipped > 0) | hosts | true | Borrar |
| eventful | eventful = true | config_reports | true | Borrar |
| failed | state = paused or result = error or result = warning | foreman_tasks_tasks | true | Borrar |
| failed | status = failed | job_invocations | true | Borrar |
| ok hosts | last_report > "35 minutes ago" and status.enabled = true and status.applied = 0 and status.failed = 0 and status.pending = 0 | hosts | true | Borrar |
| out of sync | last_report < "30 minutes ago" and status.enabled = true | hosts | true | Borrar |
| pending | status = queued or status = running | job_invocations | true | Borrar |
| recent | started_at > "24 hours ago" | job_invocations | true | Borrar |
| recurring | recurring = true | job_invocations | true | Borrar |
| running | status = running | job_invocations | true | Borrar |
| running | state = running | foreman_tasks_tasks | true | Borrar |
| succeeded | status = succeeded | job_invocations | true | Borrar |
| uptime client1 | host = client1.forelab.lan and fact_short_name = uptime | fact_values | true | Borrar |

Definiendo un trabajo con plantilla para ejecutar un estado




Utilizando la opción para ejecutar sobre un grupo utilizando la plantilla "Salt State" quedará algo como lo siguiente:




Se puede ver que resulta tedioso utilizar esta forma para invocar estados sobre equipos administrados, siendo más práctico asociarlos al "grupo/subgrupo" o "host", a través del módulo state.state_str se ha tenido que definir un estado de la siguiente forma:

```
state.sls:
      module.run:
            - mods: extrae
```

Donde se está ejecutando el estado definido en un fichero extrae.sls, aunque según la tarea puede ser más simple:

```
vim:
      pkg.installed
```

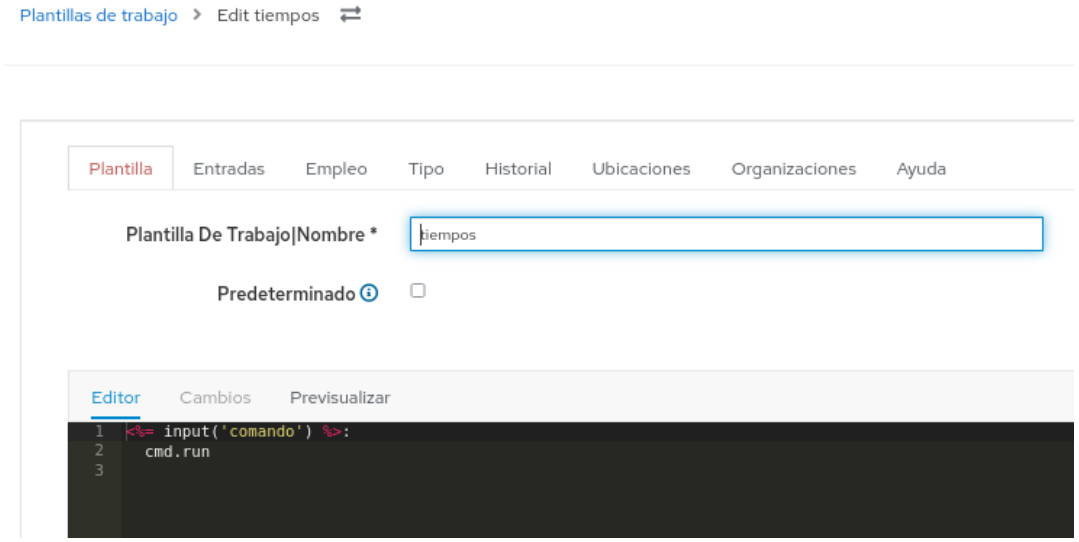
## Creación de plantillas de trabajo personalizadas

Foreman permite una personalización del entorno para realizar acciones bastante versátil permitiendo definir nuevas plantillas para lanzamiento de trabajos que se adapten a la infraestructura y uso en la organización, estas nuevas plantillas aparecerán en el selector de "Categorías de trabajo" y "Plantillas de trabajo" cuando se están definiendo nuevas tareas.

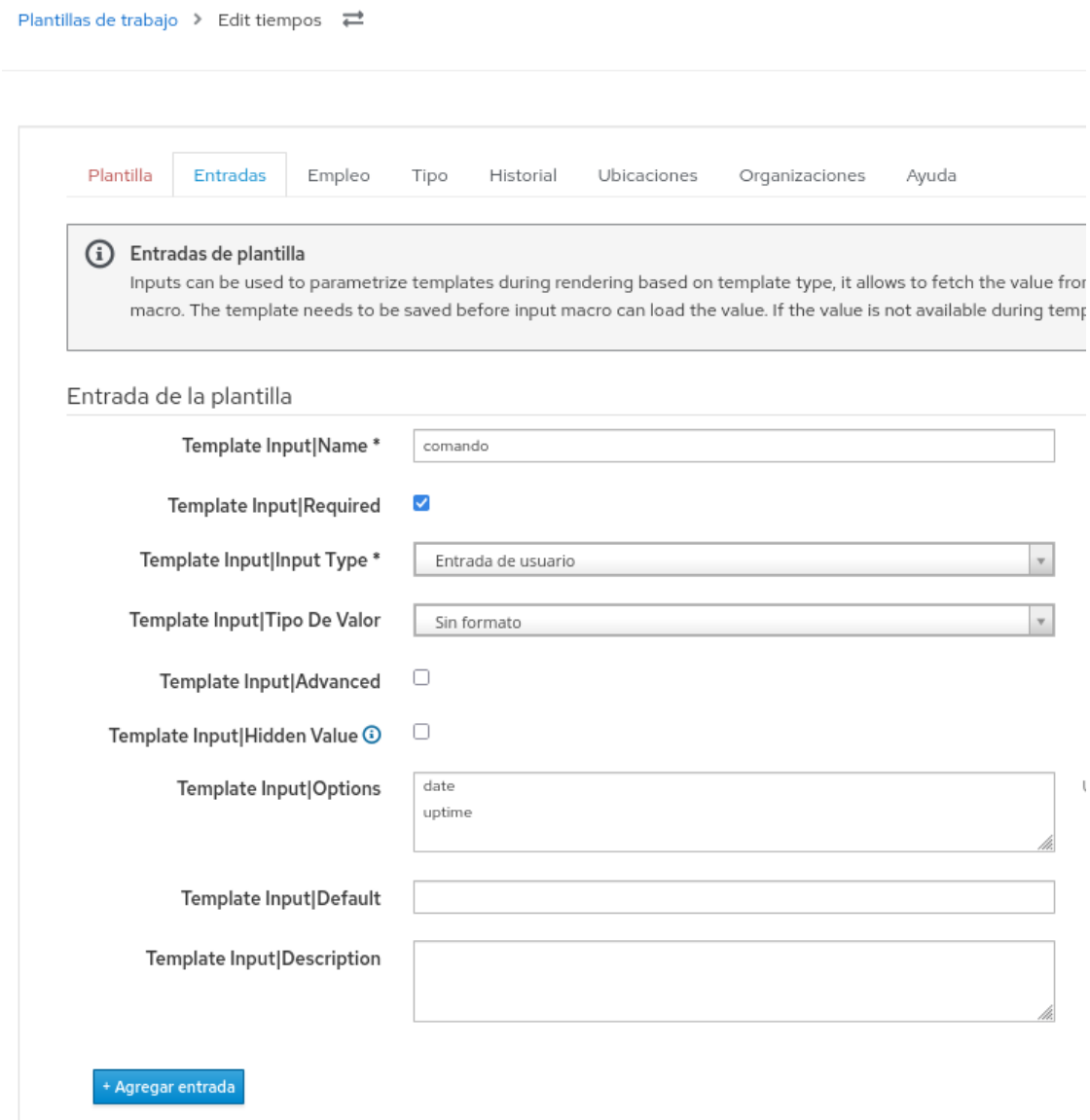
Para mostrar o crear nuevas plantillas de trabajo se accede a través de la página del menú "Hosts → Job templates", inicialmente siendo mostradas las configuradas por defecto en el sistema


Con el botón de la esquina superior derecha "Nueva plantilla de trabajo" se configurará una nueva plantilla:

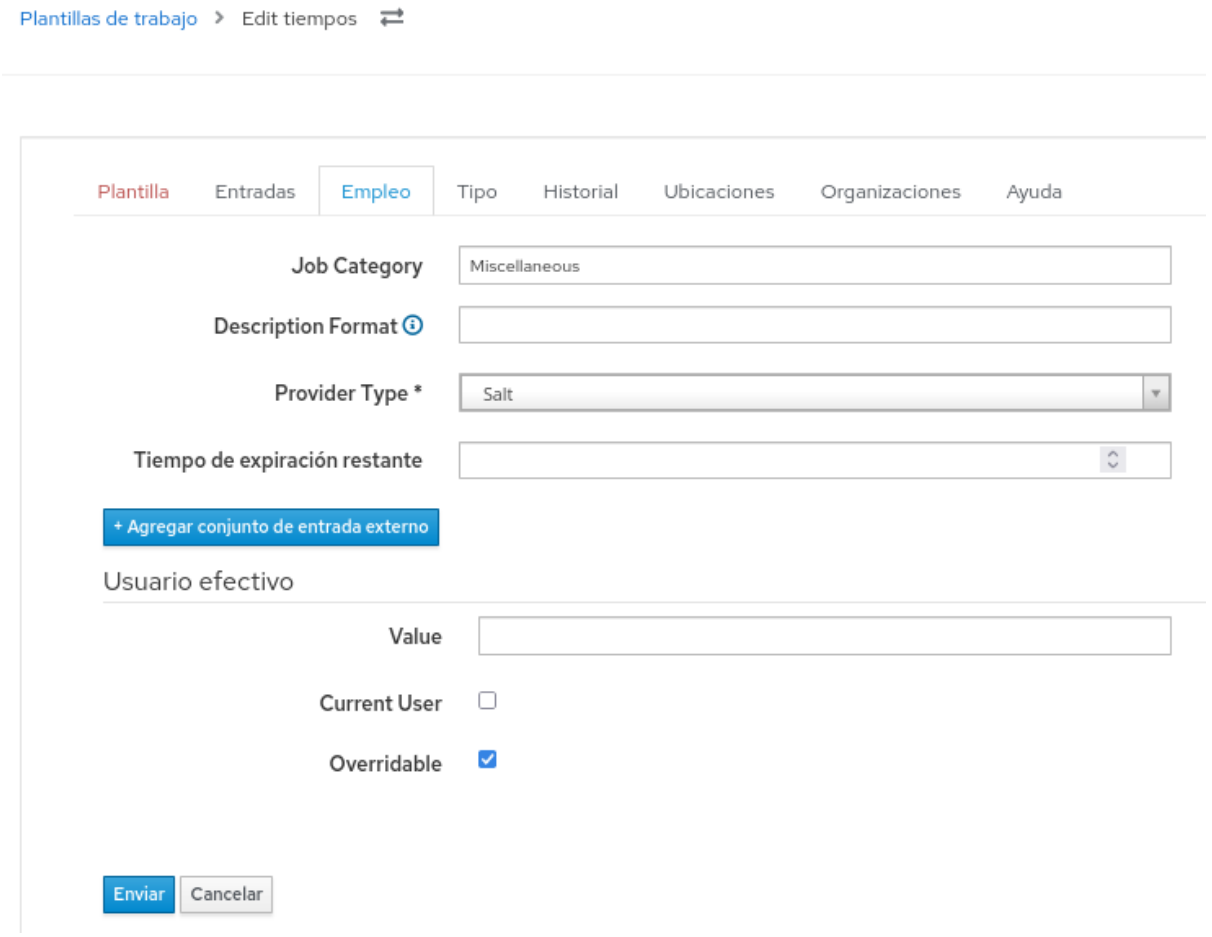

La edición de una nueva plantilla comienza estableciendo un nombre ("tiempos") y un contenido que será renderizado utilizando el motor de plantillas Jinja para formar la sintaxis Salt que se está intentando definir con las acciones. Nota: los parámetros Jinja tienen el formato `<%= input('nombre parámetro') %>`



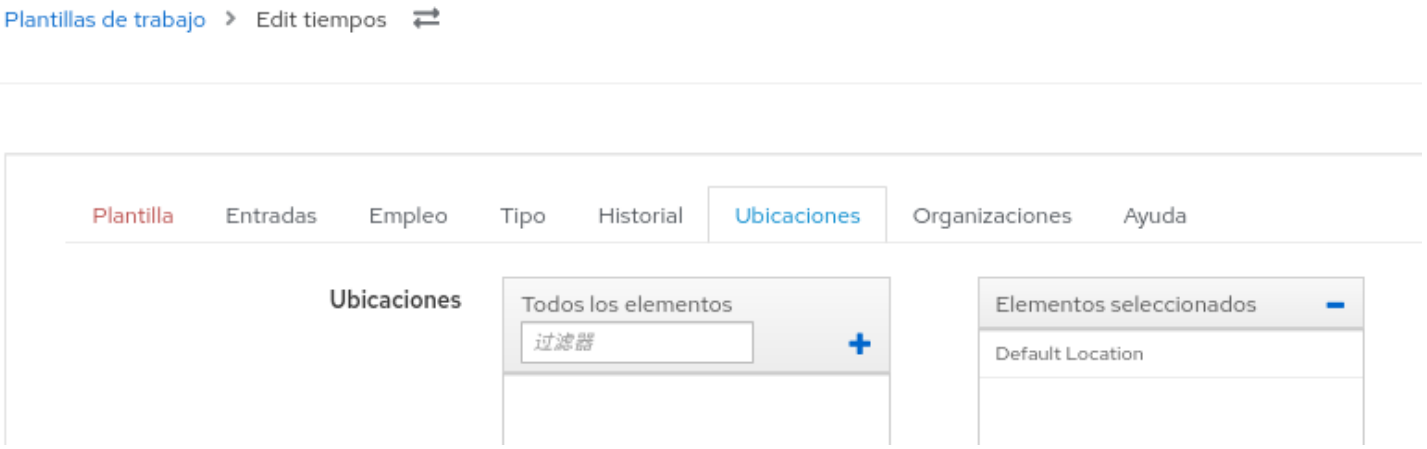

Estas plantillas pueden ser parametrizables para realizarlas de una forma general y que en el momento de la definición de una nueva tarea se aporten parámetros que completen la definición:



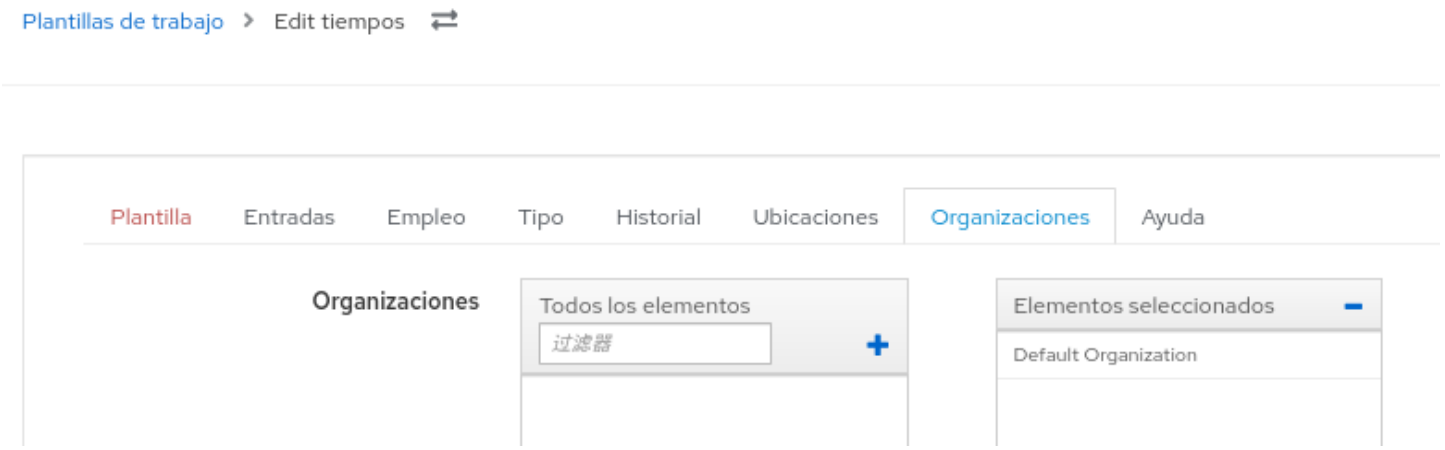
Se asigna una categoría y un proveedor, en este caso "Salt":



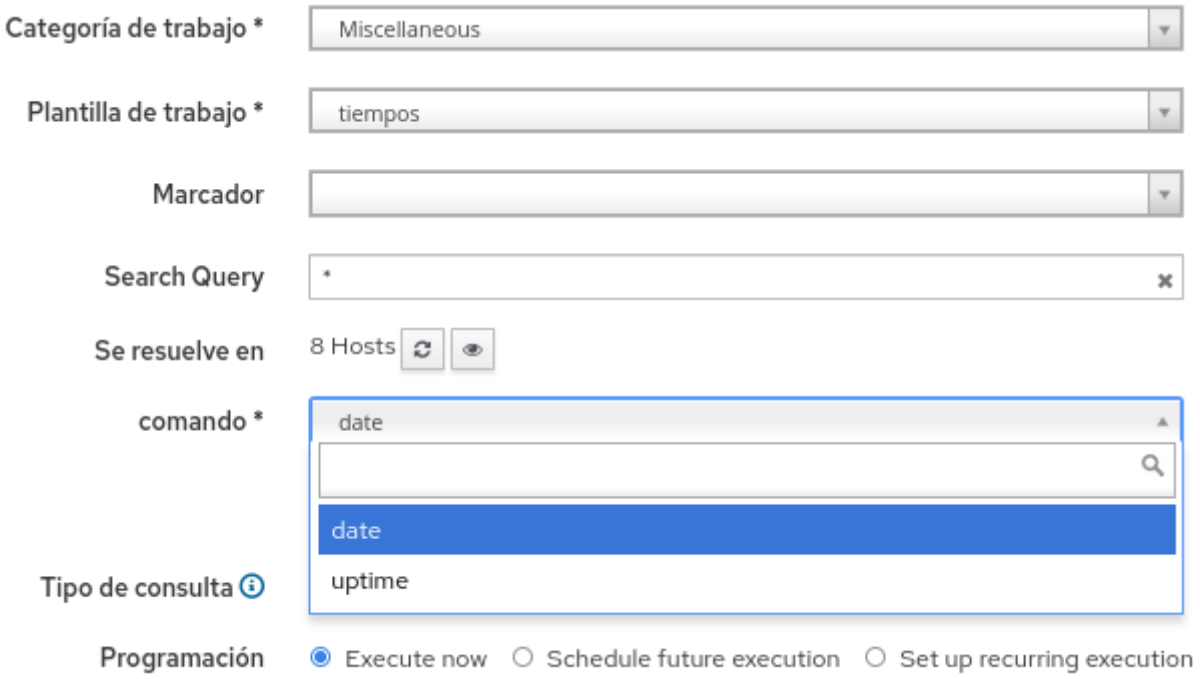

Se asigna una "Ubicación":



Y una "Organización":




Finalmente ya está disponible la plantilla parametrizada para ser utilizada en un conjunto de equipos



## Obtención de trazas sobre la ejecución de las acciones realizadas

Como se ha comentado las trazas de ejecución son mostradas a través del listado de trabajos en Foreman siendo el método recomendado.

Es recomendado porque solo queda una buena traza, si la programación se realiza desde foreman, si se realiza desde Salt, Foreman no almacena el resultado para mostrarlo y es más tedioso obtener pistas sobre qué o cómo han ocurrido las acciones.

En caso de requerir utilizar la tecnología Salt para obtener trazas desde línea de comandos puede procederse utilizando las siguientes órdenes desde el servidor maestro.

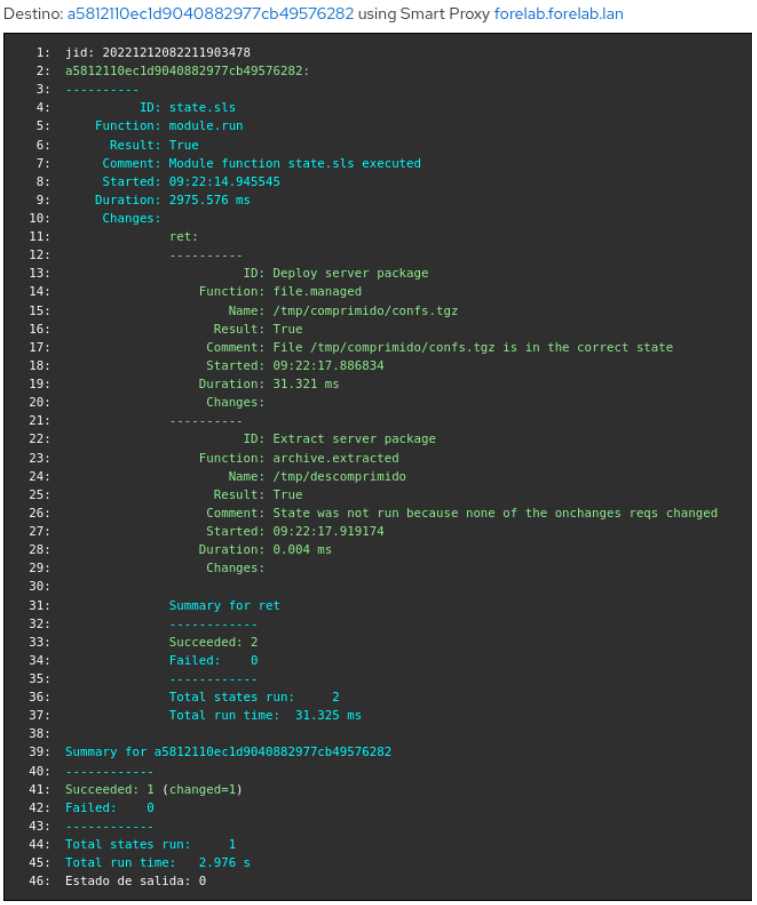
Consulta de caché de trabajos para mostrar un listado de trabajos y seleccionar uno en concreto:

```
salt-run jobs.list_jobs search_target='a58*'
salt-call saltutil.runner jobs.list_jobs search_target='a58*'
```

Consulta de caché de trabajos para mostrar las trazas de un trabajos determinado dado su identificador 'jid':

```
salt-run jobs.print_job 20221212082211903478
salt-call saltutil.runner jobs.print_job arg="['20221212082211903478']"
```

Mediante la llamada a "`salt-run jobs.print_job <jid>`" es mostrado el mismo contenido que a través de la interfaz Foreman.



## Programación de acciones o trabajos

A través de la tecnología Salt se pueden establecer procesos periódicos que realicen algunas acciones, esta programación puede ser realizada desde diferentes lugares:

- Ficheros de configuración del agente
- Información contenida en el "pillar"

No obstante la forma aconsejada es con información desde "pillar" debido a que puede ser cambiada fácilmente desde el servidor maestro ajustándose según requerimientos, si existe una programación fijada mediante ficheros de configuración en el agente está tendrá preferencia sobre los datos en "pillar".

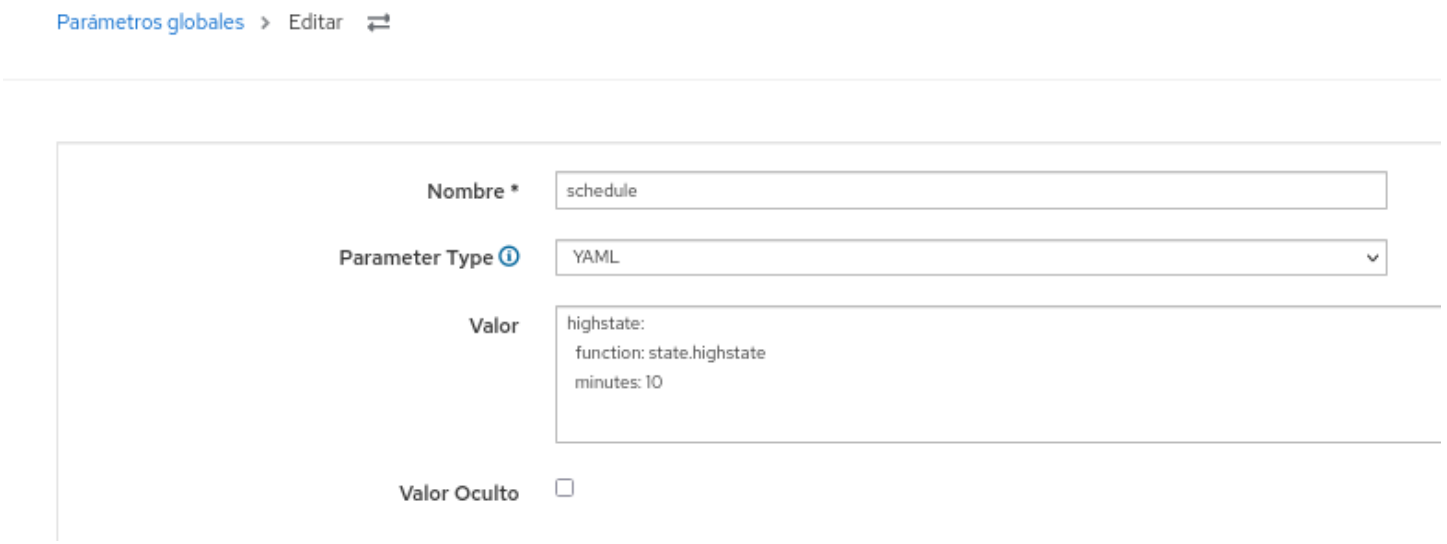
Para realizar una programación de acciones debe utilizarse el estado "schedule", el cual debe ser aportado por cualquier método que permite introducir valores como datos "pillar".

Se recuerda que los métodos para introducir datos "pillar" que son suministrados al "host" a través de Foreman que es el que controla la información "pillar" generada a través de "foreman-node" son:

- Mediante variables globales en Foreman
- Mediante variables de grupo en Foreman
- Mediante variables de host en Foreman
- Mediante variables Salt en Foreman

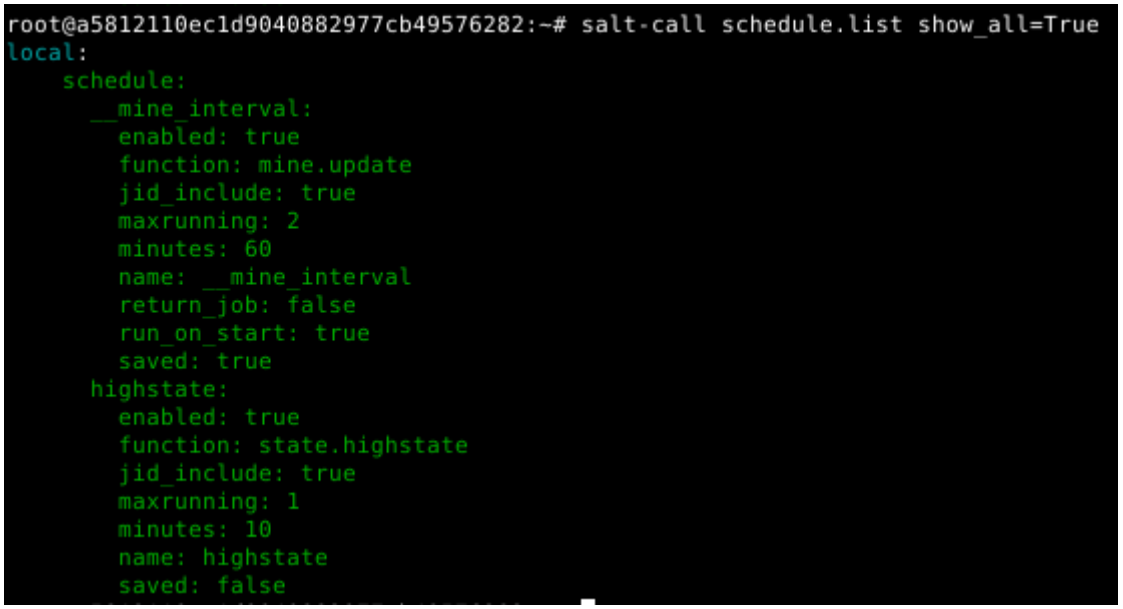
Cada lugar de definir una variable tiene una prioridad siendo donde más específicamente sea definido el que tendrá más prioridad, p.ej: es más prioritario algo fijado a través de las propiedades del "host" que a través de variables globales, es más prioritario un fichero en el "host" que cualquier información en "pillar".

Para realizar una programación que por ejemplo ejecute una llamada al "highstate" cada 10 minutos, puede definirse desde el servidor maestro una variable nombrada como "schedule" con el siguiente contenido en formato YAML.



Desde un cliente se puede consultar la lista de programaciones que actualmente tiene configuradas ejecutando la siguiente llamada:

```
salt-call schedule.list show_all=True
```




## Mantenimiento de Foreman

En ocasiones puede ser necesario realizar determinadas acciones sobre la base de datos utilizada por Foreman durante tareas de mantenimiento, Foreman provee la herramienta "foreman-rake" para realizar determinados mantenimientos sin necesidad de manipular manualmente la instancia PostgreSQL directamente y de una forma mucho más segura e íntegra. p. ej: eliminar datos de ejecución de tareas

### Foreman-rake

Muestra ayuda del comando

```
foreman-rake --help
```

Muestra la lista de las acciones disponibles

```
foreman-rake --tasks
```

Realiza un cambio de la contraseña del usuario administrador que realizó la instalación de Foreman

```
foreman-rake permissions:reset username=myadmin password='Thepassw0rd&’
```

# Requisitos hardware

https://docs.saltproject.io/en/latest/topics/tutorials/intro_scale.html

Los requisitos de hardware para implementar el sistema de gestión remota presenta una difícil forma de estimación debido a que existen múltiples factores que pueden hacer variar dichos requisitos, tanto de proceso de cálculo, como memoria, como de espacio de almacenamiento.

Algunos factores que pueden modificar los requerimientos del servicio pueden ser ocasionados por el número de nodos a administrar, las configuraciones que presente el sistema o la distribución de los componentes que conforman el servicio entre varias máquinas, lo cual es normal en un servicio, no obstante también existen factores que dependen del uso o de la forma de uso que está realizando en cada momento.

| Factor limitante que afecta | Cpu | Memoria | Disco | Red |
|---|---|---|---|---|
| Número de nodos que utilizan el sistema de forma simultánea | ✔ | ✔ | | ✔ |
| Tasa de tareas por hora que son aplicadas en los nodos que devuelven datos al servidor maestro. | ✔ | ✔ | ✔ | ✔ |
| Volumen de datos que emiten las tareas realizadas en los nodos | | | ✔ | ✔ |
| Número de agentes que realizan la tarea al mismo tiempo, esto incluye auth/re-auth/reconnect/envio | ✔ | ✔ | | ✔ |
| Tamaño del "pillar" que ha de renderizar | ✔ | ✔ | | ✔ |
| Tipo de tareas configuradas que pueden requerir más o menos recursos desde el master | | | ✔ | ✔ |
| Número de eventos "beacon" y "reactor" que son utilizados | ✔ | ✔ | | ✔ |
| Operaciones en el lado del maestro: runners, pillar, reactor, job-cache | ✔ | ✔ | | |
| Retención de datos en el servidor | | | ✔ | |

Aunque el sistema se puede dimensionar para un escalado horizontal para su funcionamiento con múltiples servidores maestros y/o proxies inteligentes para dar servicio de una forma particionada al conjunto de nodos cliente, no se tomará en cuenta para realizar esta evaluación de requisitos dado que es posible extrapolar la estimación de un caso simple de un servidor maestro contra cada uno de los servidores que realizan la tarea para todos los nodos o una partición del conjunto total de nodos.

Ante un escenario de una primera implantación sin experiencias previas en el entorno y la falta de una documentación consistente que proporcione recomendaciones de hardware siempre es aconsejable optar por un sistema que permita un escalado vertical hasta poder adaptar correctamente las características del sistema que cumpla con los requerimientos para el funcionamiento que finalmente será utilizado, de esta forma se puede optimizar dinámicamente el hardware utilizando menos recursos durante la implantación inicial y escalar conforme vaya siendo requerido, para esto puede ser muy interesante que el sistema esté funcionando de forma virtualizada para escalar fácilmente los requisitos asignados o realizar un cambio de máquina fácilmente sin tener que reinstalar todos los componentes del sistema.

Teniendo en cuenta una estimación inicial para el uso del servicio:

- Se pueden estimar aproximadamente 10k clientes, inicialmente menos y posteriormente en función de los resultados obtenidos puede crecer.
- Los clientes serán del tipo "escritorio":
  - No todos están encendidos permanentemente
  - Presentarán conexiones bastante aleatorias, principalmente en horario diurno
- No se requiere una monitorización muy detallada o muy activa, principalmente estado de paquetería y versiones utilizadas.
- El número de acciones a realizar en los nodos no será muy elevado, dado que el uso principal del servicio será instalar/actualizar/desinstalar/configurar aplicaciones.
- No se requiere orquestación entre los diferentes nodos cliente, así el número de eventos y reactores se prevé bajo.
- El uso del servicio será ofrecido mientras los clientes estén conectados a la red local con lo que no se contempla su uso a través de redes privadas alejado del entorno corporativo.

Para realizar la estimación de requisitos hardware, dado que no existe mucha documentación y es difícil llegar a replicar el número de nodos o trabajos a realizar en condiciones similares a las planificadas, se han tomado como referencia experiencias publicadas en internet de sistemas en producción y recomendaciones de fabricantes de tecnologías similares como una primera aproximación que debería ser ajustada con un factor multiplicador en función de la estimación inicial para el uso del servicio aplicado al caso de este proyecto con sus particularidades.

- Recomendaciones obtenidas desde el fabricante Vmware: https://docs.vmware.com/en/VMware-vRealize-Automation-SaltStack-Config/8.10/install-configure-saltstack-config/GUID-7C841425-1D80-4D51-9AE2-73F21D73D20C.html
  - (1) → 1 master por cada 5k nodos, con 16 cores, 16Gb RAM
- Reportes publicados por usuarios sobre sistemas Saltstack en producción en otras empresas:
  - (2) → 1 master para 15k nodos, con 12 cores, 40Gb RAM, (reporta sistema infrautilizado)
  - (3) → 1 master para 1k nodos, con 16 cores, 16Gb RAM

En estas recomendaciones o reportes obtenidos, se habla de "cores" sin tener en cuenta el trabajo que puede ofrecer según el tipo/modelo/generación, no obstante independientemente del trabajo que pueda realizar, si que es importante este valor porque puede ofrecer una estimación de los trabajos que simultáneamente se están ejecutando en el procesador de la máquina que ofrece el servicio con un uso "normal".

Teniendo en cuenta las recomendaciones, se puede extrapolar para el caso de 10k clientes:

- (1) → 32 cores, 32Gb RAM
- (2) → 8 cores, 26Gb RAM
- (3) → 160 cores, 160Gb RAM → valores "absurdos" probablemente es un sistema infrautilizado

Si nuestro sistema en su estimación inicial presenta las características descritas, bien podría ser equivalente a otro sistema con menos clientes, dado que no estarán todos conectados simultáneamente y con un uso del servicio bajo comparado con otros sistemas, parece que si se toman los límites inferiores puede ser la elección más acertada.

Los parámetros más influyentes en el funcionamiento del servicio serán siempre el número y potencia de los núcleos disponibles y la cantidad de memoria disponible en el sistema.
Teniendo en cuenta (1) y (2), parece prudente aconsejar entre 8 y 16 cores que puedan realizar 8-16 tareas simultáneamente, aunque no sean de última generación, si se puede instalar el servicio desde una máquina

virtual, una postura conservadora sería aconsejable dimensionarla con 8 cores con posibilidad de ampliarla posteriormente.

De forma semejante, respecto a la cantidad de memoria sería aconsejable al menos 16Gb de memoria si se es conservador y 32Gb si se quiere asegurar mejor rendimiento y hay posibilidad. Igualmente en caso de sistemas virtualizados, es realmente sencillo poder partir de posturas conservadoras y posteriormente dimensionarla acorde a la carga real que presente el servicio.

Poco se ha hablado del almacenamiento en disco que presenta el servicio, pero dado que el almacenamiento en disco resulta económico y fácilmente ampliable, se puede recomendar cualquier valor partiendo de al menos 120 Gb, dado que puede descomponerse como 20Gb para el sistema operativo y 100Gb para posible almacenamiento (10Mb por cliente).

A nivel de red es difícil estimar los requerimientos o uso previsto, dependerá de la programación y tipo de tareas a realizar en el sistema en producción, no obstante se puede tomar como referencia:

- 2 conexiones permanentes TCP por cliente conectado
- 250~500 KB de media por acción y cliente
- Con una conexión de 100Mb suponiendo pérdidas por el protocolo del 20% (se maximiza este factor) y dados los datos anteriores se puede ofrecer el servicio para atender 160 clientes de forma simultánea en un mismo momento.
- Dado que la conexión de red hoy en día está popularizada a velocidades de Gb, y el servicio funcionará en redes LAN corporativas, no parece que deba ser un problema limitante el uso de red.

## Requisitos software

Los requisitos que serán necesarios para implementar el sistema serán básicamente los diferentes componentes que conforman el sistema propuesto, los de desarrollo propio y los descargados de la red.

Los componentes necesarios que debe presentar el sistema son:

- Sistema basado en RPM
- Instalación de Foreman 3.4 o superior
- Configurar (al menos) los siguientes componentes/plugins en foreman:
    - `foreman`
    - `foreman_cli`
    - `foreman_cli_puppet`
    - `foreman_cli_remote_execution`
    - `foreman_cli_webhooks`
    - `foreman_proxy`
    - `puppet`
    - `foreman_plugin_column_view`
    - `foreman_plugin_puppet`
    - `foreman_plugin_remote_execution`
    - `foreman_plugin_salt`
    - `foreman_plugin_webhooks`
    - `foreman_proxy_plugin_remote_execution_script`
    - `foreman_proxy_plugin_salt`
    - `foreman_proxy_plugin_shellhooks`

  (La instalación de foreman_plugin_salt y foreman_proxy_plugin_salt implican la instalación del servicio maestro Salt)
- Interprete Python 3 disponible

# Entorno, pruebas y restriciones

## Entorno

Para el desarrollo y pruebas de este proyecto ha sido utilizado el sistema de virtualización VirtualBox donde se ha instalado la distribución Linux Debian version 11 (bullseye), mediante el sistema de virtualización es posible realizar los cambios en la infraestructura de red y generación de clientes adecuada y fácilmente para ejecutar las pruebas necesarias.

Se ha utilizado un nodo a modo de servidor central con las siguientes características:

- 6 GB de memoria RAM
- 20 GB disco duro
- 2 redes:
    - 1 tipo NAT para proveer salida a internet, dado que actúa de gateway para los clientes
    - 1 tipo interna para conexión con los clientes

Los nodos cliente utilizados son configurados conforme a las siguientes características:

- 1 GB de memoria RAM
- 12 GB disco duro
- 1 red tipo interna conectada al servidor

En el nodo servidor al ser realizadas tareas de gateway ha sido configurado un NAT para dar salida a internet a los equipos cliente, mediante:

```
# iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Dado que es necesario proveer direccionamiento IP a los clientes que autoconfiguran su dirección por DHCP se ha creado en el servidor central un rango para la red interna (enp0s8) 10.6.0.1/24 a través del software dnsmasq con las siguientes opciones:

```
port=53
server=1.1.1.1
interface=enp0s8
dhcp-range=10.6.0.10,10.6.0.30,255.255.255.0,12h
dhcp-option=option:router,10.6.0.1
dhcp-option=option:dns-server,10.6.0.1
```

## Simulación de equipos conectados a través de un NAT

Para simular que los clientes están en una red donde la conexión al servidor se realiza a través de una conexión NAT no siendo posible una conexión directa del servidor hasta los clientes han sido utilizadas las siguientes reglas de firewall que impiden dicha comunicación:

```
# iptables -I OUTPUT -d 10.6.0.15 -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -I OUTPUT 1 -p udp --match multiport --sports 67,68 -j ACCEPT
# iptables -A OUTPUT -d 10.6.0.15 -j DROP
```

## Cambio de dirección IP de los clientes

Para forzar un cambio de direccionamiento IP en clientes ya configurados, se procede desde VirtualBox a realizar un cambio de dirección MAC en la tarjeta de red (red interna) que provee la conexión con el servidor central. Mediante este cambio el servidor central a través de DHCP asigna una dirección nueva al equipo en el siguiente reinicio.

## Conectividad limitada hasta completar el inicio de sesión

Para forzar una conectividad limitada puede realizarse a través de desactivar la conectividad de la tarjeta de red desde VirtualBox no obstante es difícil controlar el momento donde se obtiene conectividad de red por DHCP y el servicio del agente ha contactado con el servidor central.

Para simular esta conectividad limitada, se han utilizado las siguientes reglas de firewall en el servidor central que impiden comunicación IP salvo la necesaria para obtener la autoconfiguración de red.

```
# iptables -I INPUT -d 10.6.0.1 -p tcp ! --dport 67:68 -j DROP
```

Para comprobar que el cliente no ha podido conectar los servicios de administración remota puede ser utilizado el log que el cliente provee al arrancar el servicio del agente

```
# journalctl -fu salt-minion
```

Cuando sea necesario habilitar la conectividad de red puede eliminarse la regla de bloqueo mediante:

```
# iptables -F INPUT
```

# Implantación

https://docs.saltproject.io/en/latest/topics/tutorials/intro_scale.html

Durante el proceso de implantación del sistema de gestión remota, deben o es recomendable ser configurados determinados parámetros de funcionamiento Salt en el servidor que permitan un funcionamiento óptimo y no sea saturado el servidor, complementado con un uso responsable de la programación o tipo de acciones que deben ser realizadas por los clientes y la información que envían al servidor, todo esto es importante dado el número de clientes que maneja la infraestructura.

- Parámetro para evitar saturación si muchos clientes se inician/reinician simultáneamente (acción que no es aconsejable realizar remotamente):
  - "acceptance_wait_time" (valor sumador de espera si timeout hasta llegar al máximo)
  - "acceptance_wait_time_max" (tope máximo de espera)
- Parámetro para evitar eventos de reautenticación cuando algunas operaciones como reinicio o borrado de claves ocurren en el servidor maestro:
  - "random_reauth_delay: 60" (en la configuración del agente).
- Parámetros para reconexión a la cola de comunicación (ZMq) en agentes:
  - "recon_default: 1000" (1 seg)
  - "recon_max: 5000" (delay antes de intentar reconectar, 59000 (1 min))
  - "recon_randomize: True" (aleatoriedad entre los dos parámetros anteriores)

- Evitar acciones a múltiples clientes con un selector '*' desde línea de comandos!
    - Utilizar mejor el modo por lotes "-b 50" (de 50 en 50)
- Parámetro para fijar un tamaño de clave SSL para minions no demasiado grande que simplifique la carga en encriptación (afecta a la carga de CPU) del servidor maestro:
    - "keysize: 2048"
- Parámetros para reducir el tiempo de renderizado del "pillar" en el maestro (afecta a la carga de CPU)
    - Comienzan con "pillar_cache" en la configuración del servidor maestro Salt
- Parámetros del servidor maestro para limitar la retención de los ficheros de caché de los minion
    - "cachedir: /var/cache/salt" (ajusta donde se crea la caché de trabajos Salt)
    - "keep_jobs: 24" (en horas) (afecta al disco)
    - "job_cache: False" (deshabilita totalmente la caché de trabajos)

# Trabajo futuro

A través de este documento se ha comentado la operativa básica y los procedimientos a desarrollar que puedan cumplir los requerimientos del proyecto saltando la documentación acerca de la instalación y configuración de los servicios Salt y Foreman debido a que ha sido planificada la provisión y configuración inicial por el servicio de sistemas de la organización.

En función de la evaluación del sistema ya implantado puede evaluarse en un futuro aumentar las capacidades y características que ofrece el servicio con las siguientes acciones.

Aumento de la capacidad del sistema a través de nuevos "smart-proxy" de Foreman que permitan repartir la carga que soporta el servidor, dado que durante en el desarrollo del proyecto no ha podido ser evaluada en condiciones reales de carga por número de clientes o condiciones reales de trabajo podría ser interesante desarrollar.

Otra opción para aumentar la capacidad o resiliencia del servicio puede ser a través del uso de múltiples maestros Salt, bien introduciendolos como servidores de respaldo donde un cliente si no puede contactar con uno pueda conectar con otro o bien organizando una red multimaestro a través de las capacidades que provee el servicio Salt (salt-syndic).

Aunque los requerimientos del proyecto restringen las capacidades de conexión que ofrecen los clientes, es posible que en un futuro surja la necesidad u oportunidad de permitir conexiones entrantes en los clientes, esto abre la puerta a configurar servicios en los ordenadores administrados que permita la visualización y monitorización del estado, así como la modificación de configuraciones manualmente o obtención de una shell interactiva, esto podría ser conseguido configurando servicios como "Cockpit" en los equipos administrados, donde a través de un servicio web se obtienen todas estas características y puede estar integrado junto a Foreman en la administración de la infraestructura.

También es posible la opción de desarrollo propio de módulos Salt personalizados y particularizados para su uso en el sistema operativo Lliurex, la infraestructura utilizada o la realización de tareas de una forma más óptima o sencilla, no olvidando la opción de desarrollar paquetes específicos para ser instalados en los equipos administrados que también ejecuten comandos de sistema operativo adaptados a la operativa de gestión remota de equipos con tecnología Salt.

También es posible la confección de servicios web propios adaptados al uso en esta infraestructura desarrollados para ser utilizados con tecnología Salt y las necesidades del proyecto que puedan sustituir o complementar la operativa ofrecida por Foreman y el plugin de manejo de Salt.

Puede ser útil en un futuro configurar bases de datos que puedan recoger eventos generados por los clientes y almacenar información en bases de datos independientes para su procesado posterior a través de las capacidades que ofrece Salt mediante los "returner".

Otro software interesante encontrado y disponible en internet que puede ser útil en este proyecto:

Salt minion inventory: https://github.com/neilmunday/Salt-Minion-Inventory

# Anexos

## Plantillas utilizadas

Las modificaciones realizadas están marcadas en negrita, las partes no útiles están tachadas, las cuales pueden ser eliminadas o bien evitar su ejecución fijando los valores de la variable de la condición.

### Plantilla inicialización de registro

- Modificada desde la plantilla "Global registration"
- Especificada a través del apartado "Administrar→Configuraciones" para establecerla por defecto
- Modificada para:
    - Fijar el hostname del equipo con el valor del UUID de la máquina
    - Ejecutar el código que sigue con más verbosidad
- Tareas que realiza:
    - Fija variables para utilizar cabeceras validación y transporte seguro SSL
    - Calcula el package manager que va a utilizar
    - Fija un posible repositorio de paquetes en la máquina
    - Pide a Foreman el script para auto-registro aportando determinadas variables calculadas (hostname) o especificadas al seleccionar opciones de registro ("Organización", "Ubicación", "Grupo") y ejecuta el código de auto-configuración que recibe.
    - (En caso de sistemas RedHat y el plugin "Katello" realiza cálculo de variables para inicializar el sistema y después de configurarlo pide a Foreman el código de auto-configuración)

```
<%#
kind: registration
name: Global Registration clone
model: ProvisioningTemplate
description: |
The registration template used to render OS agnostic script that any host can use to register to this
Foreman instance. It is rendered as a response in the registration API endpoint. The resulting script
contains instructions to prepare the machine for registration, to create a new Host record in Foreman, and
to fetch and run the host specific initial configuration script. The initial script is   rendered based on
the template of host_init_config kind.
-%>
#!/bin/sh

# Make sure, all command output can be parsed (e.g. from subscription-manager)
export LC_ALL=C LANG=C
<%
    headers = ["-H 'Authorization: Bearer #{@auth_token}'"]
    activation_keys = [(@hostgroup.params['kt_activation_keys'] if @hostgroup),
@activation_keys].compact.join(',')
-%>

# Rendered with following template parameters:
<%= "# User: [#{@user.login}]" if @user -%>
<%= "\n# Organization: [#{@organization.name}]" if @organization -%>
<%= "\n# Location: [#{@location.name}]" if @location -%>
<%= "\n# Host group: [#{@hostgroup.title}]" if @hostgroup -%>
<%= "\n# Operating system: [#{@operatingsystem}]" if @operatingsystem -%>
<%= "\n# Setup Insights: [#{@setup_insights}]" unless @setup_insights.nil? -%>
<%= "\n# Setup remote execution: [#{@setup_remote_execution}]" unless @setup_remote_execution.nil? -%>
<%= "\n# Setup remote execution pull: [#{@setup_remote_execution_pull}]" unless
@setup_remote_execution_pull.nil? -%>
<%= "\n# Remote execution interface: [#{@remote_execution_interface}]" if
@remote_execution_interface.present? -%>
<%= "\n# Packages: [#{@packages}]" if @packages.present? -%>
<%= "\n# Update packages: [#{@update_packages}]" unless @update_packages.nil? -%>
<%= "\n# Repository: [#{@repo}]" if @repo.present? -%>
```

```
<%= "\n# Repository GPG key URL: [#{@repo_gpg_key_url}]" if @repo_gpg_key_url.present? -%>
<%= "\n# Force: [#{@force}]" unless @force.nil? -%>
<%= "\n# Ignore subman errors: [#{@ignore_subman_errors}]" unless @ignore_subman_errors.nil? -%>
<%= "\n# Lifecycle environment id: [#{@lifecycle_environment_id}]" if @lifecycle_environment_id.present? -%>
<%= "\n# Activation keys: [#{activation_keys}]" if activation_keys.present? -%>


if ! [ $(id -u) = 0 ]; then
  echo "Please run as root"
  exit 1
fi

<%= snippet 'pkg_manager' %>

SSL_CA_CERT=$(mktemp)
cat << EOF > $SSL_CA_CERT
<%= foreman_server_ca_cert %>
EOF

cleanup_and_exit() {
  rm -f $SSL_CA_CERT
  exit $1
}
<% unless @repo.blank? -%>
echo '#'
echo '# Adding repository'
echo '#'


if [ x$PKG_MANAGER = xdnf -o x$PKG_MANAGER = xyum -o x$PKG_MANAGER = xzypper ]; then
  cat << EOF > /tmp/foreman_registration.repo
[foreman_register]
name=foreman_register
baseurl=<%= shell_escape @repo %>
enabled=1
type=rpm-md
EOF
<% if @repo_gpg_key_url.present? -%>
  echo gpgcheck=1 >> /tmp/foreman_registration.repo
  echo gpgkey=<%= shell_escape @repo_gpg_key_url %> >> /tmp/foreman_registration.repo
<% else -%>
  echo gpgcheck=0 >> /tmp/foreman_registration.repo
<% end -%>
  if [ x$PKG_MANAGER = xdnf -o x$PKG_MANAGER = xyum ]; then
    mv -f /tmp/foreman_registration.repo /etc/yum.repos.d/foreman_registration.repo
    echo "Building yum metadata cache, this may take a few minutes"
    $PKG_MANAGER makecache
  else
    zypper --quiet --non-interactive addrepo /tmp/foreman_registration.repo
  fi
elif [ -f /etc/debian_version ]; then
  cat << EOF > /etc/apt/sources.list.d/foreman_registration.list
<%= shell_escape @repo %>
EOF
<% if @repo_gpg_key_url.present? -%>
  apt-get -y install ca-certificates gpg
  curl --silent --show-error <%= shell_escape @repo_gpg_key_url %> | apt-key add -
<% end -%>
  apt-get update

else
  echo "Unsupported operating system, can't add repository."
  cleanup_and_exit 1
fi
<% end -%>
```

```
register_host() {
    echo 127.0.0.1 > /etc/hosts
    hostnamectl set-hostname $(dmidecode -s system-uuid|sed 's/-//g')
  curl --silent --show-error --cacert $SSL_CA_CERT --request POST <%= @registration_url %> \
      <%= headers.join(' ') %> \
      --data "host[name]=$(dmidecode -s system-uuid|sed 's/-//g')" \
      --data "host[build]=false" \
      --data "host[managed]=false" \
<%= "       --data 'host[organization_id]=#{@organization.id}' \\\n" if @organization -%>
<%= "       --data 'host[location_id]=#{@location.id}' \\\n" if @location -%>
<%= "       --data 'host[hostgroup_id]=#{@hostgroup.id}' \\\n" if @hostgroup -%>
<%= "       --data 'host[operatingsystem_id]=#{@operatingsystem.id}' \\\n" if @operatingsystem -%>
<%= "       --data host[interfaces_attributes][0][identifier]=#{shell_escape(@remote_execution_interface)}
\\\n" if @remote_execution_interface.present? -%>
<%= "       --data 'setup_insights=#{@setup_insights}' \\\n" unless @setup_insights.nil? -%>
<%= "       --data 'setup_remote_execution=#{@setup_remote_execution}' \\\n" unless
@setup_remote_execution.nil? -%>
<%= "       --data remote_execution_interface=#{shell_escape(@remote_execution_interface)} \\\n" if
@remote_execution_interface.present? -%>
<%= "       --data packages=#{shell_escape(@packages)} \\\n" if @packages.present? -%>
<%= "       --data 'update_packages=#{@update_packages}' \\\n" unless @update_packages.nil? -%>

}

echo "#"
echo "# Running registration"
echo "#"

<% if plugin_present?('katello') -%>
if [ -f /etc/redhat-release ]; then
    register_katello_host(){
        UUID=$(subscription-manager identity | head -1 | awk '{print $3}')
        curl --silent --show-error --cacert $KATELLO_SERVER_CA_CERT --request POST "<%= @registration_url
%>" \
            --data "uuid=$UUID" \
            <%= headers.join(' ') %> \
<%= "           --data 'host[organization_id]=#{@organization.id}' \\\n" if @organization -%>
<%= "           --data 'host[location_id]=#{@location.id}' \\\n" if @location -%>
<%= "           --data 'host[hostgroup_id]=#{@hostgroup.id}' \\\n" if @hostgroup -%>
<%= "           --data 'host[lifecycle_environment_id]=#{@lifecycle_environment_id}' \\\n" if
@lifecycle_environment_id.present? -%>
<%= "           --data 'setup_insights=#{@setup_insights}' \\\n" unless @setup_insights.nil? -%>
<%= "           --data 'setup_remote_execution=#{@setup_remote_execution}' \\\n" unless
@setup_remote_execution.nil? -%>
<%= "           --data remote_execution_interface=#{shell_escape(@remote_execution_interface)} \\\n" if
@remote_execution_interface.present? -%>
<%= "           --data 'setup_remote_execution_pull=#{@setup_remote_execution_pull}' \\\n" unless
@setup_remote_execution_pull.nil? -%>
<%= "           --data packages=#{shell_escape(@packages)} \\\n" if @packages.present? -%>
<%= "           --data 'update_packages=#{@update_packages}' \\\n" unless @update_packages.nil? -%>

    }

    KATELLO_SERVER_CA_CERT=/etc/rhsm/ca/katello-server-ca.pem
    RHSM_CFG=/etc/rhsm/rhsm.conf

    # Backup rhsm.conf
    if [ -f $RHSM_CFG ] ; then
      test -f $RHSM_CFG.bak || cp $RHSM_CFG $RHSM_CFG.bak
    fi

    # rhn-client-tools conflicts with subscription-manager package
    # since rhn-tools replaces subscription-manager, we need to explicitly
    # install subscription-manager after the rhn-tools cleanup
    if [ x$ID = xol ]; then
      $PKG_MANAGER remove -y rhn-client-tools
      $PKG_MANAGER install -y --setopt=obsoletes=0 subscription-manager
    fi
```

```
    # Prepare SSL certificate
    mkdir -p /etc/rhsm/ca
    cp -f $SSL_CA_CERT $KATELLO_SERVER_CA_CERT
    chmod 644 $KATELLO_SERVER_CA_CERT

    # Prepare subscription manager
    <% if truthy?(@force) -%>
    if [ -x "$(command -v subscription-manager)" ] ; then
      subscription-manager unregister || true
      subscription-manager clean
    fi

    $PKG_MANAGER remove -y katello-ca-consumer\*
    <% end -%>

    if ! [ -x "$(command -v subscription-manager)" ] ; then
      $PKG_MANAGER install -y subscription-manager
    else
      $PKG_MANAGER upgrade -y subscription-manager
    fi

    if ! [ -f $RHSM_CFG ] ; then
      echo "'$RHSM_CFG' not found, cannot configure subscription-manager"
      cleanup_and_exit 1
    fi

    # Configure subscription-manager
    test -f $RHSM_CFG.bak || cp $RHSM_CFG $RHSM_CFG.bak
    subscription-manager config \
        server.hostname="<%= @rhsm_url.host if @rhsm_url %>" \
      --server.port="<%= @rhsm_url.port if @rhsm_url %>" \
        server.prefix="<%= @rhsm_url.path if @rhsm_url %>" \
      --rhsm.repo_ca_cert="$KATELLO_SERVER_CA_CERT" \
      --rhsm.baseurl="<%= @pulp_content_url %>"

    # Older versions of subscription-manager may not recognize
    # report_package_profile and package_profile_on_trans options.
    # So set them separately and redirect out & error to /dev/null
    # to fail silently.
    subscription-manager config --rhsm.package_profile_on_trans=1 > /dev/null 2>&1 || true
    subscription-manager config --rhsm.report_package_profile=1 > /dev/null 2>&1 || true

    # Configuration for EL6
    if grep --quiet full_refresh_on_yum $RHSM_CFG; then
      sed -i "s/full_refresh_on_yum\s*=.*$/full_refresh_on_yum = 1/g" $RHSM_CFG
    else
      full_refresh_config="#config for on-premise management\nfull_refresh_on_yum = 1"
      sed -i "/baseurl/a $full_refresh_config" $RHSM_CFG
    fi

    subscription-manager register <%= '--force' if truthy?(@force) %> \
        org='<%= @organization.label if @organization %>' \
      --activationkey='<%= activation_keys %>' || <%= truthy?(@ignore_subman_errors) ? 'true' :
'cleanup_and_exit 1' %>

    register_katello_host | bash
else
    register_host | bash
fi
<% else -%>
register_host | bash -x
<% end -%>

cleanup_and_exit
```

# Plantilla inicialización de configuración

- Modificada desde la plantilla "host_init_config"
- Especificada a través del apartado "Host→Sistemas operativos→Plantillas" para establecerla por defecto
- Modificada para:
  - Incrementar la verbosidad de su ejecución
  - Fijar el hostname de la máquina
  - Realizar la instalación de los componentes Salt
- Tareas que realiza:
  - Configuración para comunicación segura SSL
  - Fija el nombre de máquina
  - (No útil) Instalación de los componentes Puppet
  - Instalación de los componentes Salt
  - (No útil) Instala posibles paquetes adicionales
  - (No útil) Configuración de las claves SSH para permitir ejecución remota de código y shell
  - (No útil) Configuración del sistema "Katello"
  - (No útil) Actualiza paquetes si así es seleccionado
  - (No útil) Lanza otro script para acciones port-instalación

```
<%#
kind: host_init_config
name: Linux host_init_config default
model: ProvisioningTemplate
oses:
- AlmaLinux
- CentOS
- CentOS_Stream
- Fedora
- Rocky
- SLES
- Debian
- Ubuntu
description: |
       This template is used during the host registration to perform the initial host configuration. After
       the host is created by starting the registration, the registration script asks for the host init
       config script, that is rendered based on this template. It is rendered for the specific host
       therefore it contains instructions specific for the OS of the host. It's content can differ based on
       any parameters applicable for the host.

       It deploys the CA certificate so any later communication with the Foreman is TLS secured. Then it
       performs initial steps, such as puppet deployment, remote execution SSH key configuration etc. At the
       end it informs Foreman that provisioning has finished.
-%>
<% built_https = foreman_url('built').start_with?('https') -%>

#!/bin/bash
set -e
set -x
echo "# Running [<%= @host.name %>] host initial configuration"

<% if built_https -%>
SSL_CA_CERT=$(mktemp)
cat << EOF > $SSL_CA_CERT
<%= foreman_server_ca_cert %>
EOF
<% end -%>

foreman_curl() {
  curl --silent --show-error <%= '--cacert $SSL_CA_CERT' if built_https %> -o /dev/null --noproxy \* "$@"
}
```

```
exit_and_cancel_build() {
  echo 'Host [<%= @host.name %>] initial configuration failed'
  foreman_curl --request POST '<%= foreman_url('failed') %>' \
       --data 'Host initial configuration failed, please see the registration log for more details.'
  exit 1
}

set +e
trap 'exit_and_cancel_build' ERR

# Se fija el nombre de host (desde el parámetro de llamada)
echo <%= @host.name %> > /etc/hostname
echo 127.0.0.1 localhost > /etc/hosts

<% if !host_param_true?('skip-puppet-setup') && (host_puppet_server.present? ||
host_param_true?('force-puppet')) -%>
<%= snippet 'puppetlabs_repo' %>
<%= snippet 'puppet_setup_clone' %>
<% end -%>

<% if !host_param_true?('skip-salt-setup') || host_param_true?('force-salt') -%>
<%= snippet 'salt_repo_customized' %>
<%= snippet 'saltstack_setup clone' %>
<% end -%>

<% if host_param_true?('host_registration_remote_execution') -%>
<%= snippet 'remote_execution_ssh_keys' %>
<% end -%>

<% if plugin_present?('katello') && host_param_true?('host_registration_remote_execution_pull') -%>
<%= snippet 'remote_execution_pull_setup' %>
<% end -%>

<%= install_packages(host_param('host_packages')) -%>

<% if host_param_true?('host_registration_insights') -%>
<%= snippet 'insights' %>

<% end -%>

<% if plugin_present?('katello') -%>
if command -v subscription-manager &>/dev/null; then
  echo "Refreshing subscription data"
  subscription-manager refresh
fi

<% if host_param_true?('redhat_install_host_tools') -%>
<%= install_packages('katello-host-tools') -%>
<% end -%>

<% if host_param_true?('redhat_install_host_tracer_tools') -%>
<%= install_packages('katello-host-tools-tracer') -%>
<% end -%>

<% end -%>
<%= update_packages if host_param_true?('host_update_packages') -%>

<%= snippet_if_exists('host_init_config_post') -%>

# Call home to exit build mode
trap - ERR
foreman_curl '<%= foreman_url('built') %>'

if [[ $? == 0 ]] ; then
  echo "Host [<%= @host.name %>] successfully configured."
else
  echo "Host [<%= @host.name %>] successfully configured, but failed to set built status."
fi
```

```
<% if plugin_present?('katello') && @host.operatingsystem.family == 'Redhat' -%>
subscription-manager facts --update

<% end -%>

exit 0
```

# Plantilla ("snippet") repositorio Salt (salt_repo_customized)

- Modificada desde la plantilla "puppetlabs_repo"
- Tareas que realiza:
  - Fijar el repositorio para obtener los paquetes de componentes Salt

```
<%#
kind: snippet
name: salt_repo_customized
model: ProvisioningTemplate
snippet: true
description: |
  Fetches the package that deploys the Salt repository that can be
  used to install Salt from. It only performs the installation in case
  one of the enable-salt-repo parameter is set to true.
-%>
<%
http_proxy   = host_param('http-proxy') ? " --httpproxy #{host_param('http-proxy')}" : nil
http_port    = host_param('http-proxy-port') ? " --httpport #{host_param('http-proxy-port')}" : nil
proxy_uri    = host_param('http-proxy') ?
"http://#{host_param('http-proxy')}:#{host_param('http-proxy-port')}" : nil
proxy_string = proxy_uri ? " -e https_proxy=#{proxy_uri}/" : ''
proxy_string_bits = proxy_uri ? " -ProxyUsage Override -ProxyList #{proxy_uri}" : ''
os_family = @host.operatingsystem.family
os_major  = @host.operatingsystem.major.to_i
os_name   = @host.operatingsystem.name

if os_family == 'Redhat'
  # no configurado
elsif os_family == 'Suse'
  # no configurado
elsif os_family == 'Debian'
  repo_host = 'repo.saltproject.io'
  repo_os = @host.operatingsystem.release_name
elsif os_family == 'Windows'
  # no configurado
end

if host_param_true?('enable-salt-repo')
  repo_name = 'puppet-release'
  repo_subdir = ''
end
-%>
<% if repo_name -%>
<% if os_family == 'Debian' -%>
apt-get update
apt-get -y install ca-certificates curl
curl -fsSL -o /usr/share/keyrings/salt-archive-keyring.gpg https://<%= repo_host
%>/salt/py3/debian/11/amd64/latest/salt-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/salt-archive-keyring.gpg arch=amd64] https://"<%=repo_host
%>"/salt/py3/debian/11/amd64/latest bullseye main" | tee /etc/apt/sources.list.d/salt.list
apt-get update
<% end -%>
<% end -%>
```

# Plantilla ("snippet") instalación Salt (saltstack_setup clone)

- Modificada desde la plantilla "puppet_setup"
- Modificada para:
  - Realizar la instalación de los componentes Salt
- Tareas que realiza:
  - Instalar el componente "minion" (cliente) de Salt
  - Configurar el cliente con una configuración válida
  - Fijar el identificador del cliente Salt

```
<%#
kind: snippet
name: saltstack_setup clone
model: ProvisioningTemplate
snippet: true
description: this snippet will configure the Saltstack Minion
-%>
<%
etc_path = (@host.operatingsystem.family == 'Freebsd') ? '/usr/local/etc/salt' : '/etc/salt'
bin_path = (@host.operatingsystem.family == 'Freebsd') ? '/usr/local/bin' : '/usr/bin'
%>

<% if @host.operatingsystem.family == 'Debian' -%>
apt-get update
apt-get install -y salt-minion
<% elsif @host.operatingsystem.family == 'Freebsd' -%>
pkg install -y py27-salt
<% elsif @host.operatingsystem.family == 'Redhat' -%>
if [ -f /usr/bin/dnf ]; then
  dnf -y install salt-minion
else
  yum -t -y install salt-minion
fi
<% elsif @host.operatingsystem.family == 'Suse' -%>
  /usr/bin/zypper -n install salt-minion
<% end -%>

cat > <%= etc_path %>/minion.d/minion.conf << EOF
<%= snippet 'saltstack_minion' %>
EOF

#echo <%= @host.name %> > <%= etc_path %>/minion_id
UUID=$(dmidecode -s system-uuid|sed 's/-//g')
echo $UUID > <%= etc_path %>/minion_id

<% if @host.operatingsystem.family == 'Freebsd' -%>
echo 'salt_minion_enable="YES"' >>/etc/rc.conf
echo 'salt_minion_paths="/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin"' >>/etc/rc.conf
<% elsif @host.operatingsystem.family == 'Redhat' -%>
/sbin/chkconfig --level 345 salt-minion on
<% elsif @host.operatingsystem.family == 'Suse' -%>
/sbin/chkconfig salt-minion on -f
<% end -%>

<%= bin_path %>/salt-call --no-color --grains >/dev/null
```

# Plantilla ("snippet") configuración salt-minion (saltstack_minion)

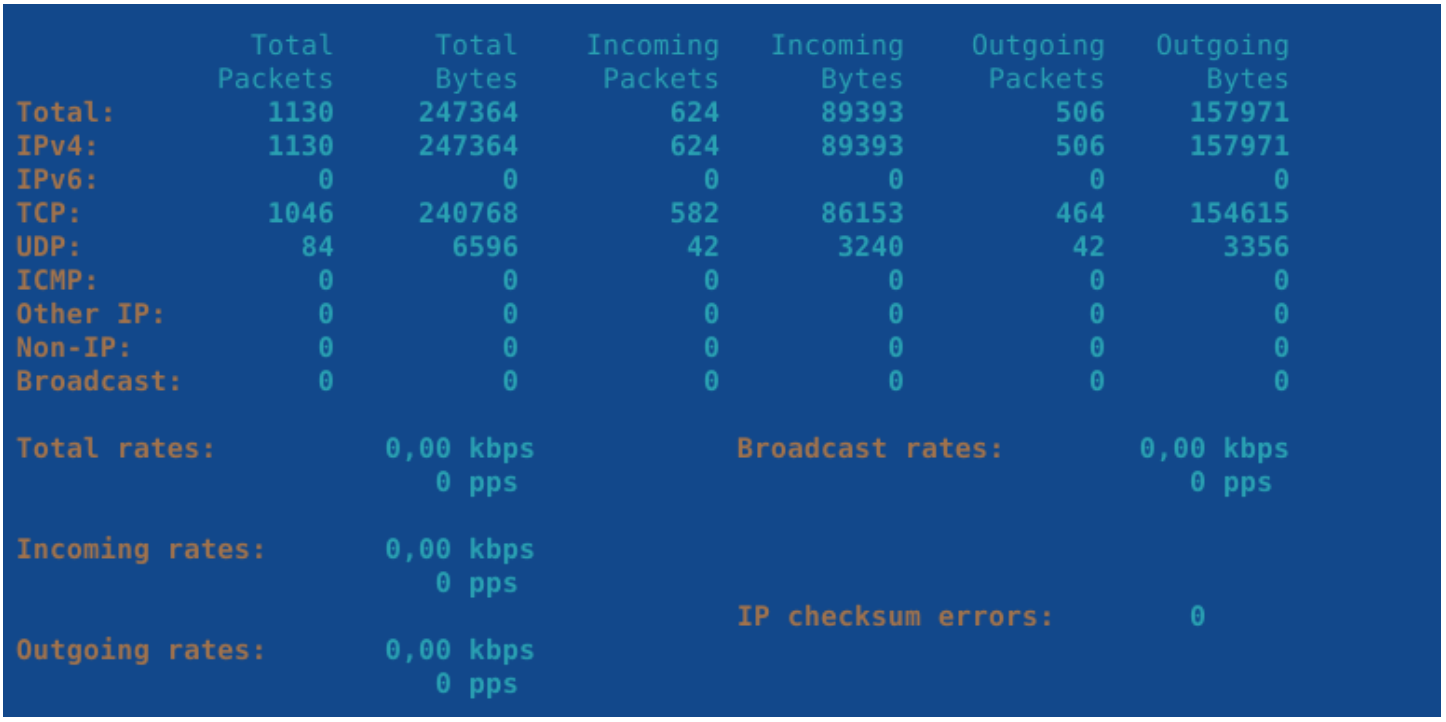
- No modificada
- Tareas que realiza:
  - Fija que servidor maestro utilizará
  - Fija el nivel de log
  - Fija el valor de "grains" para auto-registro de las claves en el sistema Salt

```
<%#
kind: snippet
name: saltstack_minion
model: ProvisioningTemplate
snippet: true
description: |
  Generates a Salt minion configuration file which is required for the Salt bootstraping.
  The Salt master is configured based on the host parameter called "salt_master".
  It can also statically assign grains based on the "salt_grains" host parameter.
-%>
master: <%= host_param('salt_master') %>
log_level: warning

autosign_grains:
  - autosign_key

<%#
# Grains (http://docs.saltstack.com/en/latest/topics/targeting/grains.html#grains-in-the-minion-config)
#
# Possible parameters are:
#
# * {'cluster': 'alpha'}
# * {'roles': ['webserver', 'frontend']}
#
-%>
<% if plugin_present?('foreman_salt') -%>
grains: <%= to_json(@host.derive_salt_grains(use_autosign: true)) %>
<% else -%>
grains: <%= host_param('salt_grains') ? to_json(host_param('salt_grains')) : '{}' %>
<% end -%>
```

# Trazas de rendimiento

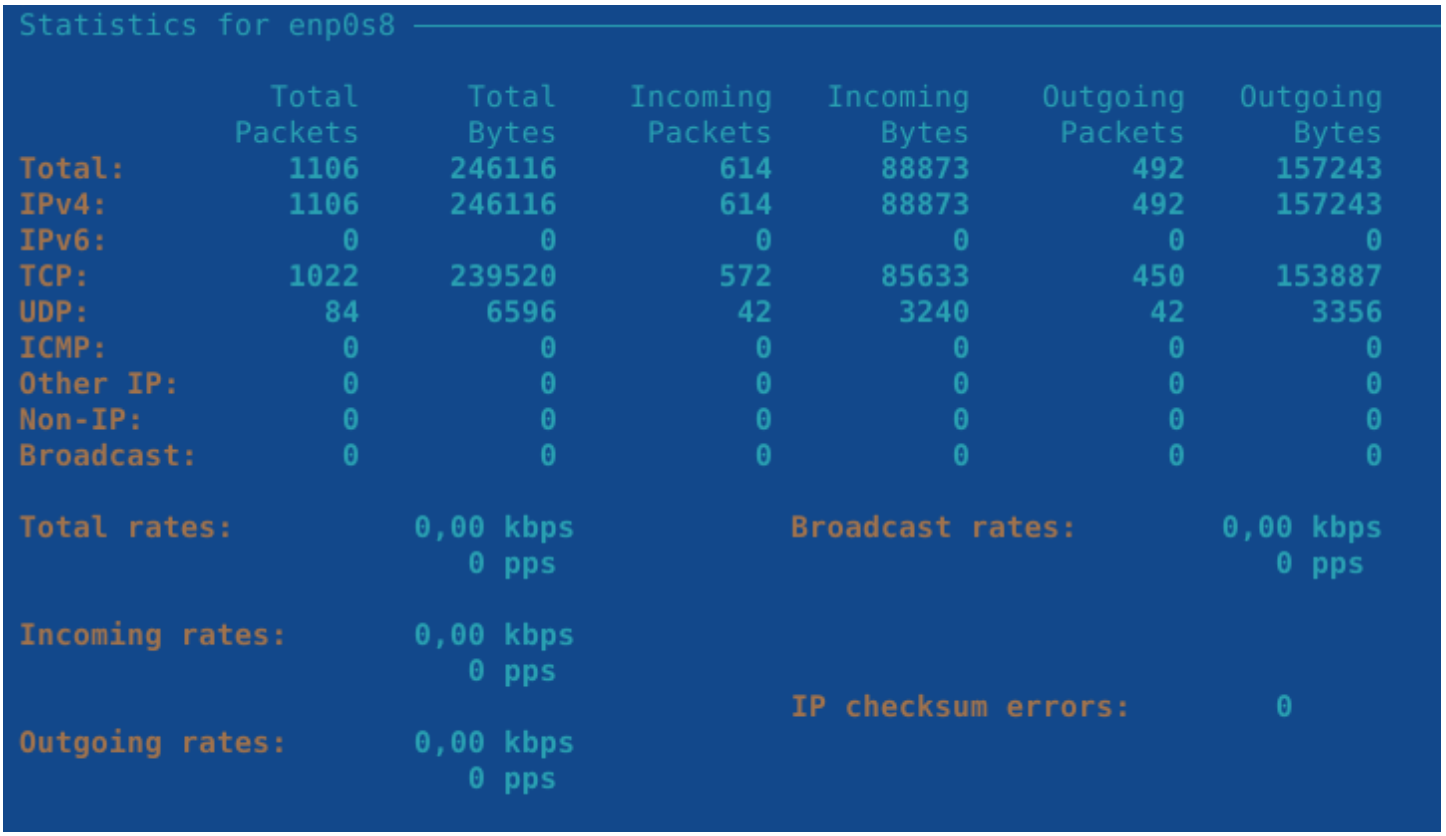
## Red

Ejecución de una conexión de cliente al servidor Salt maestro donde se aprecia el volumen de datos utilizado



| | Total Packets | Total Bytes | Incoming Packets | Incoming Bytes | Outgoing Packets | Outgoing Bytes |
|---|---|---|---|---|---|---|
| Total: | 1130 | 247364 | 624 | 89393 | 506 | 157971 |
| IPv4: | 1130 | 247364 | 624 | 89393 | 506 | 157971 |
| IPv6: | 0 | 0 | 0 | 0 | 0 | 0 |
| TCP: | 1046 | 240768 | 582 | 86153 | 464 | 154615 |
| UDP: | 84 | 6596 | 42 | 3240 | 42 | 3356 |
| ICMP: | 0 | 0 | 0 | 0 | 0 | 0 |
| Other IP: | 0 | 0 | 0 | 0 | 0 | 0 |
| Non-IP: | 0 | 0 | 0 | 0 | 0 | 0 |
| Broadcast: | 0 | 0 | 0 | 0 | 0 | 0 |

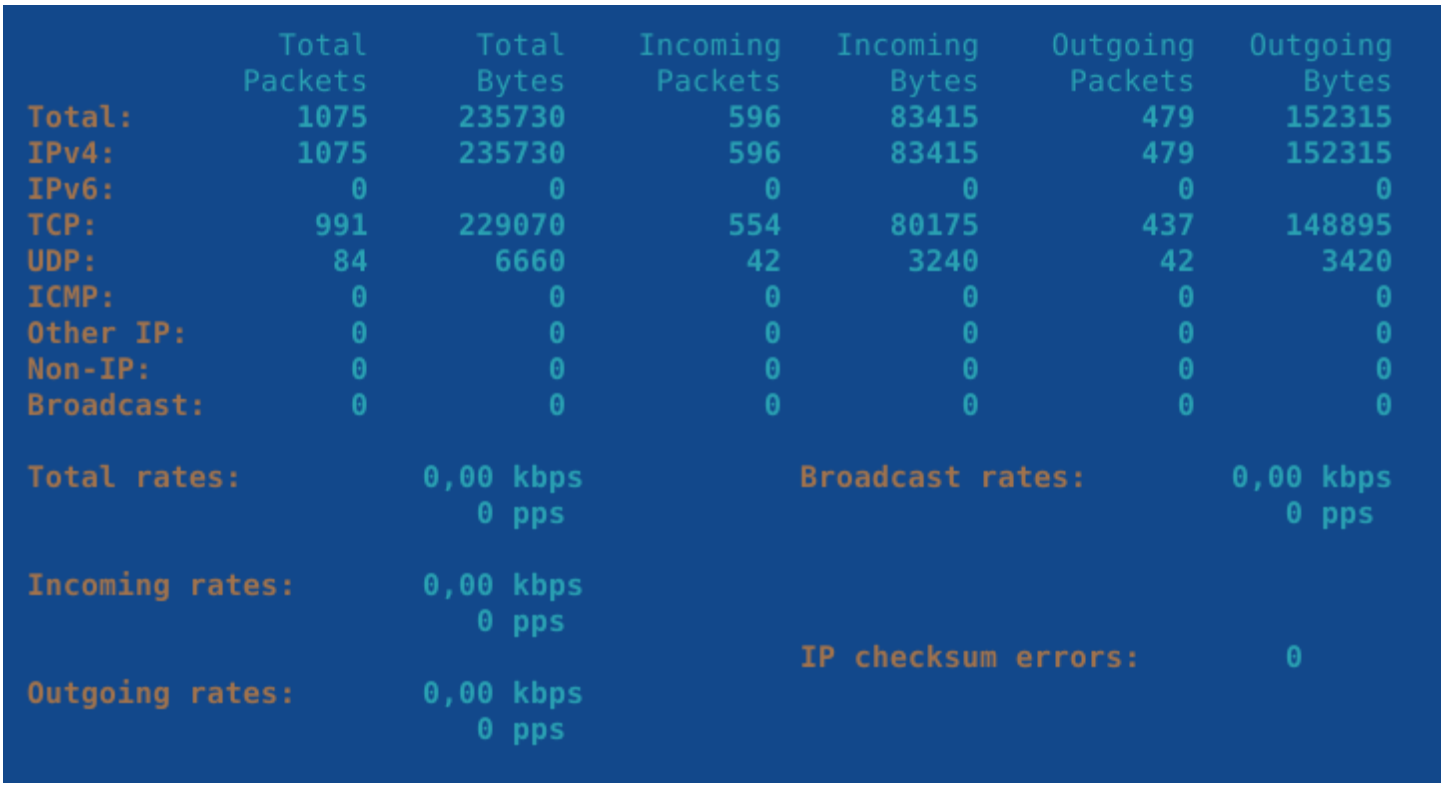
Total rates: 0,00 kbps / 0 pps
Broadcast rates: 0,00 kbps / 0 pps
Incoming rates: 0,00 kbps / 0 pps
IP checksum errors: 0
Outgoing rates: 0,00 kbps / 0 pps



Statistics for enp0s8

| | Total Packets | Total Bytes | Incoming Packets | Incoming Bytes | Outgoing Packets | Outgoing Bytes |
|---|---|---|---|---|---|---|
| Total: | 1106 | 246116 | 614 | 88873 | 492 | 157243 |
| IPv4: | 1106 | 246116 | 614 | 88873 | 492 | 157243 |
| IPv6: | 0 | 0 | 0 | 0 | 0 | 0 |
| TCP: | 1022 | 239520 | 572 | 85633 | 450 | 153887 |
| UDP: | 84 | 6596 | 42 | 3240 | 42 | 3356 |
| ICMP: | 0 | 0 | 0 | 0 | 0 | 0 |
| Other IP: | 0 | 0 | 0 | 0 | 0 | 0 |
| Non-IP: | 0 | 0 | 0 | 0 | 0 | 0 |
| Broadcast: | 0 | 0 | 0 | 0 | 0 | 0 |

Total rates: 0,00 kbps / 0 pps
Broadcast rates: 0,00 kbps / 0 pps
Incoming rates: 0,00 kbps / 0 pps
IP checksum errors: 0
Outgoing rates: 0,00 kbps / 0 pps



| | Total Packets | Total Bytes | Incoming Packets | Incoming Bytes | Outgoing Packets | Outgoing Bytes |
|---|---|---|---|---|---|---|
| Total: | 1075 | 235730 | 596 | 83415 | 479 | 152315 |
| IPv4: | 1075 | 235730 | 596 | 83415 | 479 | 152315 |
| IPv6: | 0 | 0 | 0 | 0 | 0 | 0 |
| TCP: | 991 | 229070 | 554 | 80175 | 437 | 148895 |
| UDP: | 84 | 6660 | 42 | 3240 | 42 | 3420 |
| ICMP: | 0 | 0 | 0 | 0 | 0 | 0 |
| Other IP: | 0 | 0 | 0 | 0 | 0 | 0 |
| Non-IP: | 0 | 0 | 0 | 0 | 0 | 0 |
| Broadcast: | 0 | 0 | 0 | 0 | 0 | 0 |

Total rates: 0,00 kbps
0 pps

Broadcast rates: 0,00 kbps
0 pps

Incoming rates: 0,00 kbps
0 pps

IP checksum errors: 0

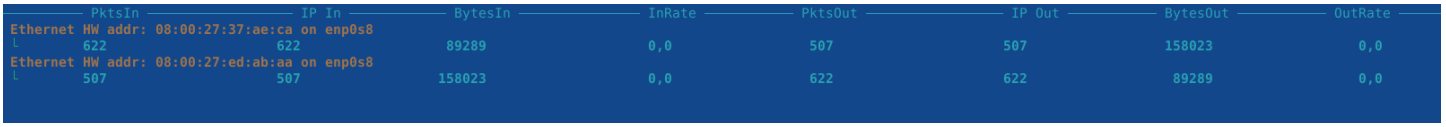
Outgoing rates: 0,00 kbps
0 pps

Se puede ver el funcionamiento del cliente Salt donde al inicio de su ejecucióin se establecen conexiones en diferentes puertos al servicio:

Puerto 4505: Publicador de trabajos

Puerto 4506: Recepción de trabajos

| TCP Connections (Source Host:Port) | | Packets | Bytes | Flag | Iface |
|---|---|---|---|---|---|
| 10.6.0.1:4505 | > | 2 | 1162 | -PA- | enp0s8 |
| 10.6.0.15:54206 | > | 2 | 104 | --A- | enp0s8 |
| 10.6.0.1:4506 | = | 8 | 634 | -PA- | enp0s8 |
| 10.6.0.15:47834 | = | 9 | 5849 | --A- | enp0s8 |
| 10.6.0.1:4506 | = | 8 | 634 | CLOS | enp0s8 |
| 10.6.0.15:47382 | = | 10 | 1149 | CLOS | enp0s8 |
| 10.6.0.15:47590 | = | 10 | 806 | CLOS | enp0s8 |
| 10.6.0.1:4506 | = | 8 | 730 | CLOS | enp0s8 |
| 10.6.0.15:47420 | > | 1 | 52 | --A- | enp0s8 |
| 10.6.0.1:4506 | = | 0 | 0 | ---- | enp0s8 |
| 10.6.0.15:47592 | > | 1 | 52 | --A- | enp0s8 |
| 10.6.0.1:4506 | = | 0 | 0 | ---- | enp0s8 |
| 10.6.0.15:47608 | > | 1 | 52 | --A- | enp0s8 |
| 10.6.0.1:4506 | = | 0 | 0 | ---- | enp0s8 |
| 10.6.0.15:47620 | > | 1 | 52 | --A- | enp0s8 |
| 10.6.0.1:4506 | = | 0 | 0 | ---- | enp0s8 |
| 10.6.0.1:4506 | = | 0 | 0 | ---- | enp0s8 |
| 10.6.0.15:51152 | > | 1 | 52 | --A- | enp0s8 |



| | PktsIn | IP In | BytesIn | InRate | PktsOut | IP Out | BytesOut | OutRate |
|---|---|---|---|---|---|---|---|---|
| Ethernet HW addr: 08:00:27:37:ae:ca on enp0s8 | | | | | | | | |
| └ | 622 | 622 | 89289 | 0,0 | 507 | 507 | 158023 | 0,0 |
| Ethernet HW addr: 08:00:27:ed:ab:aa on enp0s8 | | | | | | | | |
| └ | 507 | 507 | 158023 | 0,0 | 622 | 622 | 89289 | 0,0 |

## Memoria

Memoria utilizada por el servidor Salt con tres clientes conectados

| PID | USER | PRI | NI | VIRT | RES▽ | SHR | S | CPU% | MEM% | TIME+ | Command |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1129 | root | 20 | 0 | 2199M | 101M | 10444 | S | 1.0 | 1.7 | 2:38.07 | /usr/bin/python3 /usr/bin/salt-master |
| 1155 | root | 20 | 0 | 840M | 68736 | 11760 | S | 0.0 | 1.1 | 0:08.12 | /usr/bin/python3 /usr/bin/salt-master |
| 1167 | root | 20 | 0 | 841M | 68692 | 11760 | S | 0.0 | 1.1 | 0:08.18 | /usr/bin/python3 /usr/bin/salt-master |
| 1196 | root | 20 | 0 | 978M | 67784 | 15064 | S | 1.0 | 1.1 | 1:19.51 | /usr/bin/python3 /usr/bin/salt-minion |
| 1140 | root | 20 | 0 | 840M | 63956 | 11684 | S | 0.0 | 1.0 | 0:07.28 | /usr/bin/python3 /usr/bin/salt-master |
| 1161 | root | 20 | 0 | 839M | 63324 | 11684 | S | 0.0 | 1.0 | 0:07.65 | /usr/bin/python3 /usr/bin/salt-master |
| 1159 | root | 20 | 0 | 840M | 62484 | 11684 | S | 0.0 | 1.0 | 0:06.85 | /usr/bin/python3 /usr/bin/salt-master |
| 1090 | root | 20 | 0 | 119M | 61284 | 5452 | S | 0.0 | 1.0 | 0:01.52 | /usr/bin/python3 /usr/bin/salt-master |
| 1131 | root | 20 | 0 | 828M | 61140 | 8968 | S | 0.0 | 1.0 | 2:18.93 | /usr/bin/python3 /usr/bin/salt-master |
| 1053 | root | 20 | 0 | 2885M | 56640 | 8460 | S | 0.0 | 0.9 | 1:26.33 | /usr/bin/python3 /usr/bin/salt-api |
| 627 | root | 20 | 0 | 83160 | 49752 | 17392 | S | 0.0 | 0.8 | 0:11.71 | /usr/bin/python3 /usr/bin/salt-master |
| 626 | root | 20 | 0 | 81148 | 40272 | 16908 | S | 0.0 | 0.7 | 0:03.13 | /usr/bin/python3 /usr/bin/salt-api |
| 111980 | root | 20 | 0 | 153M | 37284 | 4640 | S | 0.0 | 0.6 | 0:00.14 | /usr/bin/python3 /usr/bin/salt-master |
| 1085 | root | 20 | 0 | 156M | 23812 | 8940 | S | 0.0 | 0.4 | 0:00.14 | /usr/bin/python3 /usr/bin/salt-master |
| 1136 | root | 20 | 0 | 443M | 20256 | 7480 | S | 0.0 | 0.3 | 0:04.40 | /usr/bin/python3 /usr/bin/salt-master |
| 1132 | root | 20 | 0 | 77752 | 17604 | 4348 | S | 0.0 | 0.3 | 0:02.44 | /usr/bin/python3 /usr/bin/salt-master |
| 1058 | root | 20 | 0 | 48884 | 17096 | 12760 | S | 0.0 | 0.3 | 0:00.61 | /usr/bin/python3 /usr/bin/salt-minion |
| 1271 | root | 20 | 0 | 122M | 8108 | 3372 | S | 0.0 | 0.1 | 0:00.00 | /usr/bin/python3 /usr/bin/salt-minion |
| 955 | root | 20 | 0 | 48420 | 6352 | 3208 | S | 0.0 | 0.1 | 0:00.00 | /usr/bin/python3 /usr/bin/salt-master |

# Llamadas y módulos Salt utilizados en este documento

| Módulo ejecutable | Estado | Utilidad |
|---|---|---|
| · test.version | | Obtiene la versión Salt instalada |
| · test.ping | | Realiza una prueba de conectividad |
| · system.poweroff | | Apaga un equipo |
| · sys.list_modules | | Lista todos los módulos ejecutables disponibles |
| · grains.items | | Lista los grain de un equipo |
| · pillar.items | | Muestra la información pillar del equipo |
| · pillar.data | | Muestra la información pillar del equipo |
| · pkg.install | · pkg.installed | Instala un paquete |
| · pkg.remove | · pkg.removed | Elimina un paquete |
| · pkg.list_repos | | Lista los repositorios configurados |
| · pkg.list_upgrades | | Informa de las actualizaciones de sistema operativo pendientes |
| · lowpkg.list_pkgs | | Lista paquetes y versiones instaladas |
| · pkg.upgrade | | Actualiza las versiones de todos los paquetes en el sistema |
| · state.apply | | Aplica un estado |
| · state.show_sls '*'<br>· state.show_highstate | | Muestra todos los estados configurados ("highstate)" |
| · state.show_sls 'install_htop' | | Muestra el fichero de estado |
| · state.template_str <definición> | | Permite inyectar un estado desde línea de comandos |
| · cmd.run | | Ejecuta un comando de sistema operativo |
| · file.read | | Lee el contenido de un fichero |
| · file.write | | Escribe contenido en un fichero |
| · service.get_all | | Lista todos los servicios en la máquina administrada |
| · service.status | | Comprueba el estado del servicio pasado como parámetro |
| | · service.running | Define que el servicio esté en funcionamiento |
| · archive.tar | · archive.extracted | Extrae ficheros empaquetados |

## Runners

| | |
|---|---|
| · salt-run manage.alived | Permite comprobar el listado de equipos operativos consultando solo la caché del servidor maestro. |
| · salt-run manage.up | Permite comprobar el listado de equipos operativos consultando a todos los clientes. |

| | |
|---|---|
| · `salt-run jobs.list_jobs` | Permite listar los trabajos que han sido realizados consultando la caché de maestro |
| · `salt-run jobs.print_job` | Permite obtener la traza de un trabajo realizado |

Más información en el listado completo oficial:

Módulos ejecutables: https://docs.saltproject.io/en/latest/ref/modules/all/index.html
Módulos de estado: https://docs.saltproject.io/en/latest/ref/states/all/index.html
Módulos runner: https://docs.saltproject.io/en/latest/ref/runners/all/index.html