Guia de buenas prácticas para administración de equipos a través de Foreman & Salt

Índice

Introducción

En este documento se pretende documentar algunos de los casos de uso a modo de ejemplos que pueden presentarse cuando se realiza una administración remota de equipos Lliurex a través de los servicios de Foreman y Salt que están funcionando.

Es conveniente recordar y hacer notar antes de abordar cualquier caso de uso que existen múltiples formas o métodos de obtener el mismo resultado utilizando este sistema de administración y que dependerá del desarrollador u operador encontrar la forma más ventajosa para realizar las acciones teniendo en cuenta factores como:

- Dificultad de desarrollo
- Portabilidad
- Seguridad en la realización de acciones
- Flujo de ejecución que ofrecen los servicios y los equipos administrados (portátiles)
- Tipo de ejecución (puntual o periódica)
- Limitaciones impuestas por la red

En cualquier caso, una recomendación básica es tratar de utilizar en la medida de lo posible el propio gestor de paquetería presente en la distribución o las herramientas incluidas en la distribución presentes o futuras.

Casos de uso propuestos como ejemplos

- Programación del agente
- Búsqueda de información de un equipo
- Asociación de estados a grupos
- Instalación/eliminación de aplicaciones
- Ejecución arbitraria de comandos de sistema
- Mantener el sistema actualizado
- Instalación de ficheros EPI (Zero-Center)
- Modificación de ficheros
- Creación de lanzadores en los escritorios de los usuarios
- Acciones sobre los usuarios

Programación del agente

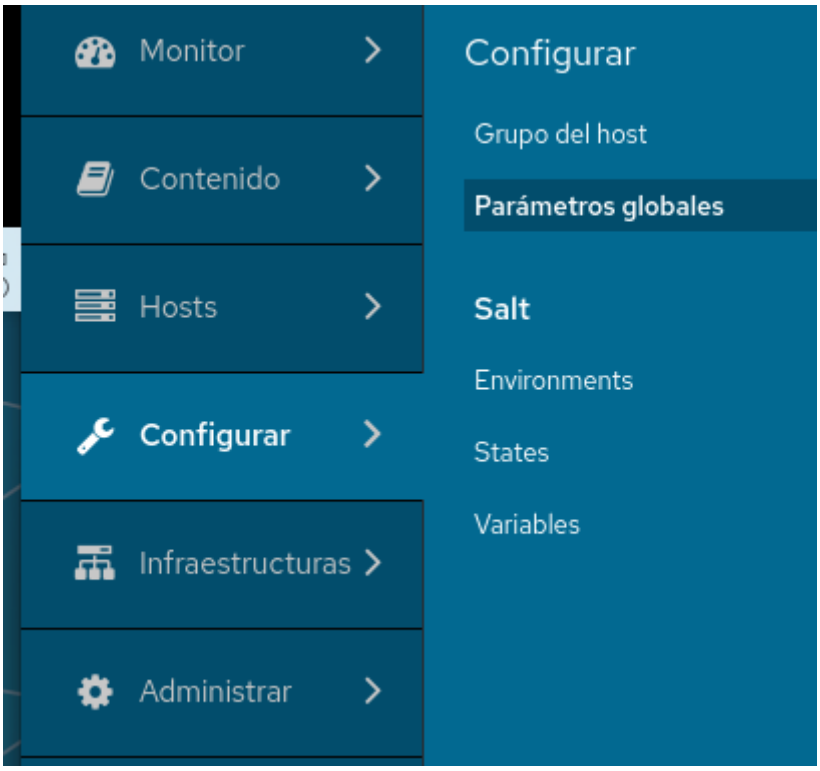
El agente Salt que está funcionando en los clientes realiza tareas periódicas contra el servidor como sincronización bidireccional de datos.

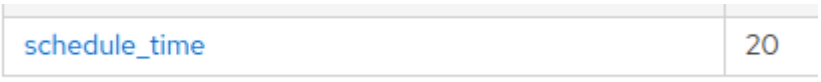
Esta sincronización permite que en el servidor puedan ser recopilados datos de funcionamiento y estado de los equipos administrados a través de "*facts*" que pueden ser consultados para obtener la información de un equipo o bien para localizarlo filtrando entre los diferentes equipos que son mostrados en la interfaz de Foreman y ejecutar acciones basadas en estos datos.

En el proceso de sincronización también son actualizados los datos en disco necesarios en los clientes que son provistos por el servidor para poder realizar la correcta disponibilidad y ejecución de los procesos programados.

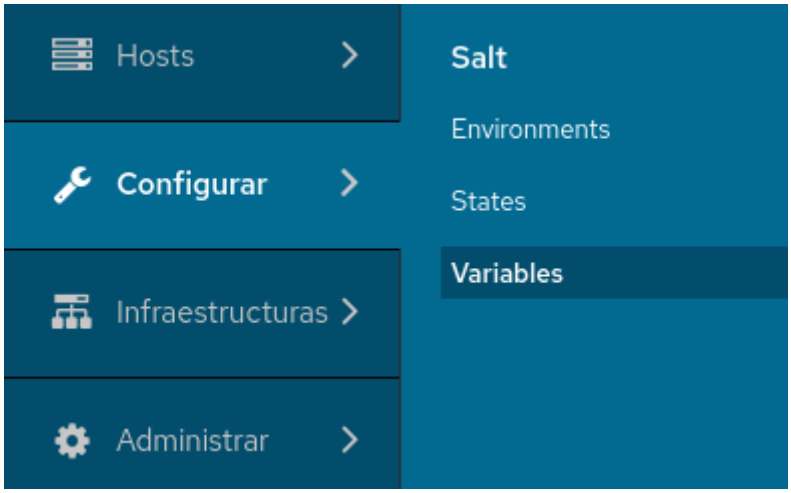
La sincronización se produce de forma periódica teniendo en cuenta valores fijados a través de la interfaz de Foreman desde varios lugares, hay que tener presente el ámbito de las variables de Foreman para fijar de forma global, al grupo, al ordenador el valor de tiempo adecuado para la sincronización.

La sincronización puede ser ajustada de forma global a través de las opciones de parámetros globales, *Configurar>Parámetros globales*, nombradas como "schedule_time".

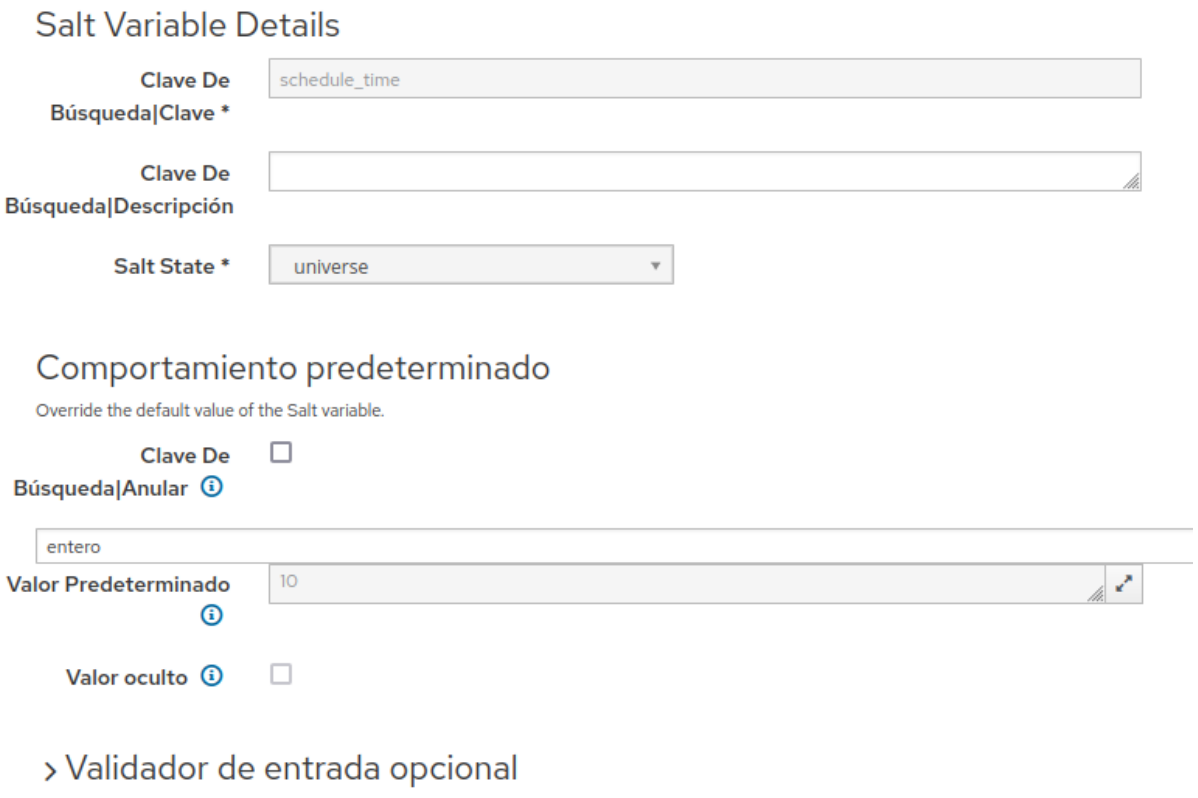




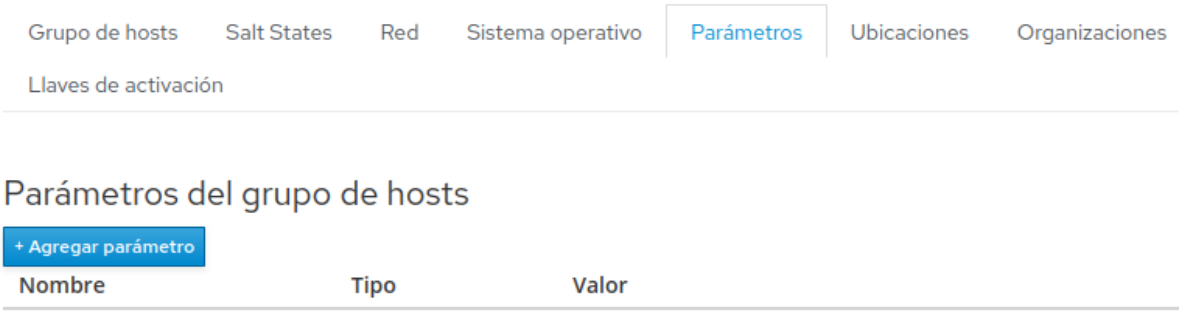
También puede ser fijada de forma global a través de las opciones que nos da Foreman para fijar variables en las plantillas de Salt asociadas, *Configurar>Salt Variables*.



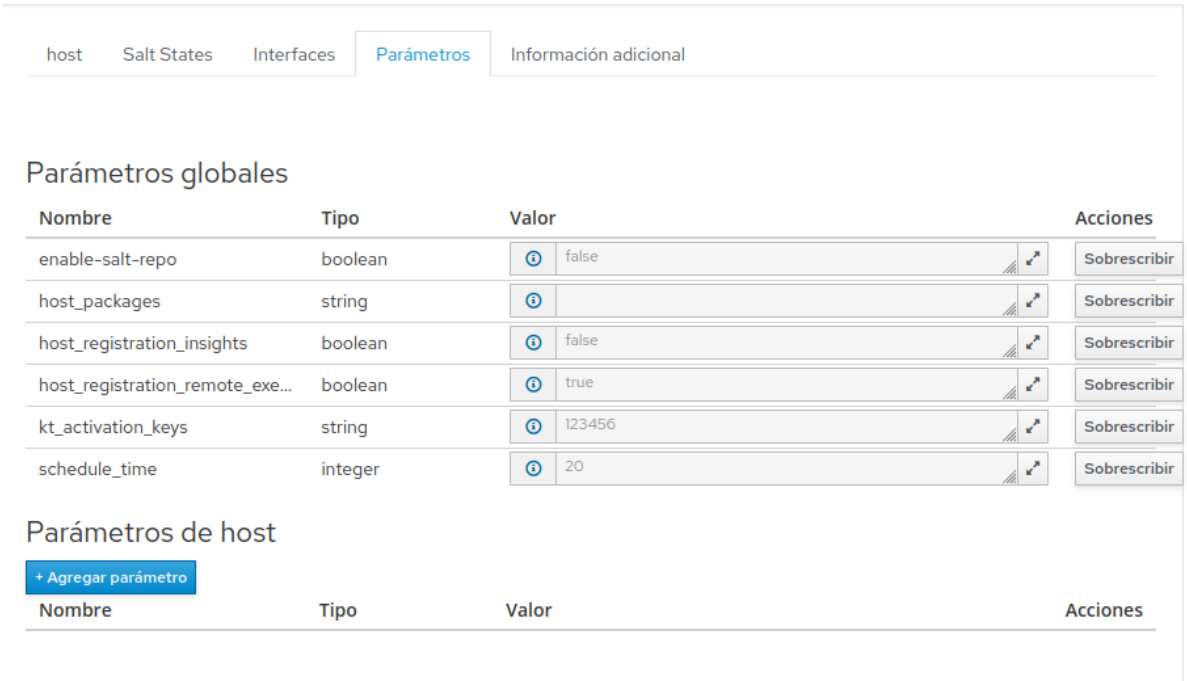
schedule_time	universe	integer



Particularizando más este valor puede ajustarse de forma similar a través de las opciones que ofrece fijar variables en grupos editándolos, *Configura>Grupo del host* seleccionar grupo:

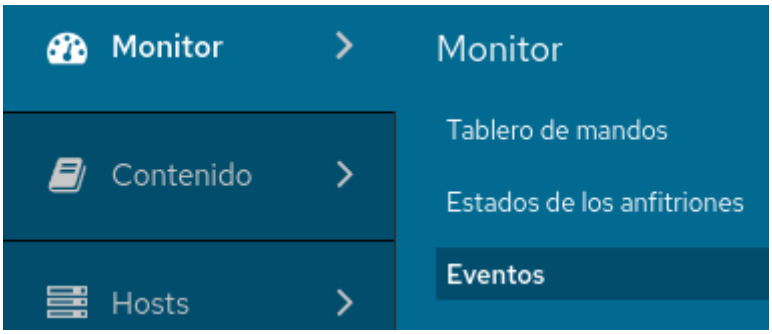


o bién a equipos en concreto editando los parámetros del equipo seleccionado desde el listado de equipos y utilizando la opción *Editar*:



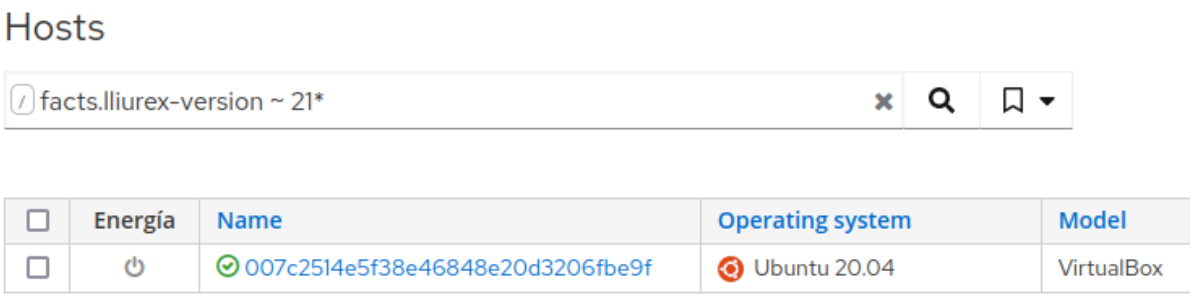
Búsqueda de información de un equipo

Para mostrar la información que provee el equipo tal como listado de paquetería, versión instalada, etiquetas aplicadas pueden utilizarse los accesos a través de *Monitor>Eventos:*



007c2514e5f38e4...	lliurex-version	21.230412
007c2514e5f38e4...	cwd	/
007c2514e5f38e4...	shell	/bin/sh
007c2514e5f38e4...	uptime	17:54:20

Es importante remarcar que esta información es útil porque puede ser utilizada aplicando filtros, los cuales pueden ser guardados para facilitar el uso de la herramienta y generar listados en base a características de los equipos:



Con listados de esta información filtrada es bastante sencillo mover/incluir equipos de unos grupos a otros o aplicar acciones de forma parcial.

Alguna de esta información no es genérica y es dependiente del sistema Linux o Lliurex, no obstante se puede calcular cualquier información deseada para que los equipos la remitan al servidor y posteriormente ser utilizada, para demostrar esa funcionalidad son utilizados los "*grain*" personalizados a través de la

ejecución de código Python que puede ser distribuida incluyendo el contenido en la carpeta "*_grains*" del repositorio Git.

Ejemplos de mandar información personalizada son:

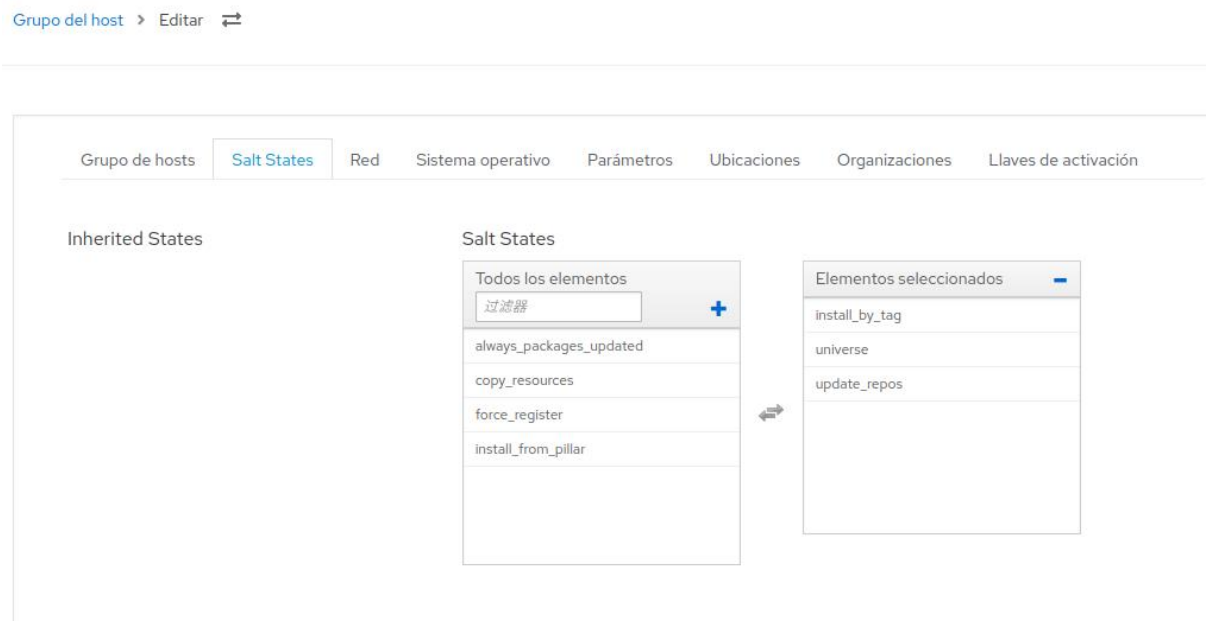
- Versión de Lliurex (*_grains/llxversion.py*): Muestra el contenido con el "*grain*" "llx-version".
- Listado de paquetes instalados (*_grains/pkg_list.py*): Muestra el contenido con el "*grain*" "pkgs".
- Listado de etiquetas en equipo (*_grains/tags_list.py*): Muestra el conjunto de etiquetas con el "*grain*" "tags".
- Resultado de un comando de sistema (*_grains/uptime.py*): Muestra el tiempo encendido un equipo con el "*grain*" "uptime".

Asociación de estados a grupos

Para conseguir realizar acciones sobre los equipos es necesario generalmente asociar estados a equipos, estos estados provienen desde el repositorio Git que utiliza el sistema Foreman/Salt.

Cabe hacer notar que aunque es posible la ejecución a través de la programación de tareas que ofrece Foreman debido a la naturaleza del conjunto de equipos administrados que no siempre estarán conectados o disponibles para tareas en el momento de la ejecución es recomendable la administración a través de los estados que son aplicados.

Es posible asociar los estados a través de la edición de un grupo:

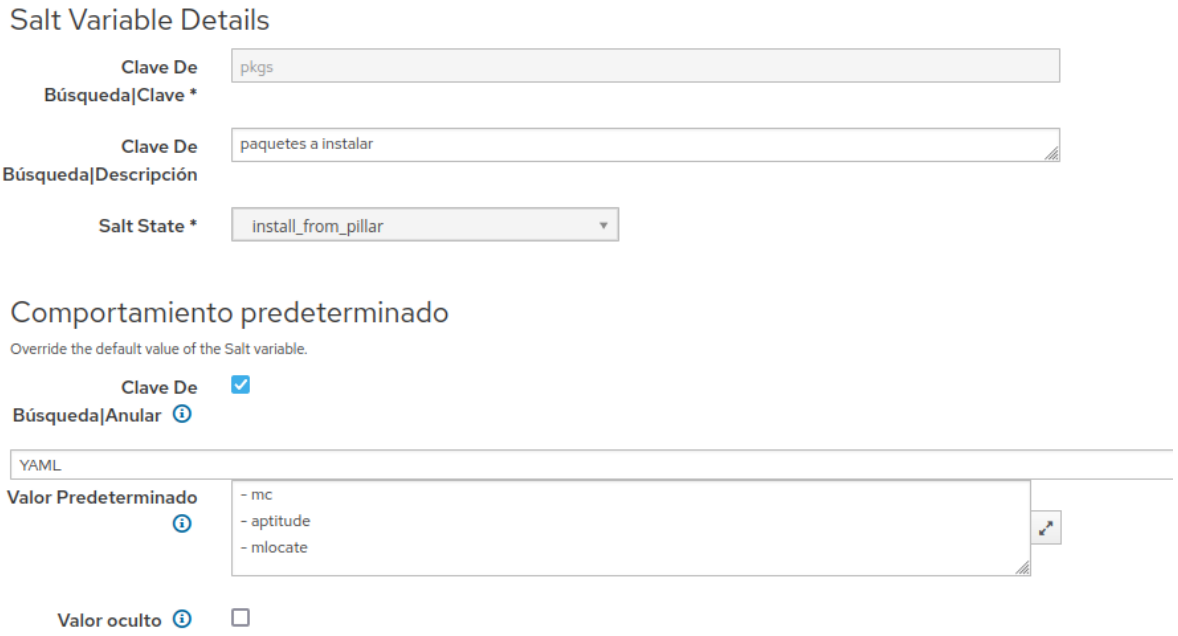


Instalación/eliminación de aplicaciones

Para la instalación de aplicaciones es posible utilizar varios métodos según los estados que se proveen.

- Es posible instalar aplicaciones utilizando el parámetro global "*pkgs*" utilizado en el estado *install_from_pillar.sls*
- De forma similar, es posible instalar aplicaciones utilizando el parámetro fijado "*pkgs*" como variable del estado "*install_from_pillar*" en la variable de los estados Salt.

pkgs	install_from_pillar	yaml	Borrar



- De forma similar, se puede particularizar la instalación a los equipos de un grupo utilizando el parámetro "*pkgs_<nombre_de_tag>*", mediante este método el listado de paquetes será instalados en los equipos que posean dicho tag.

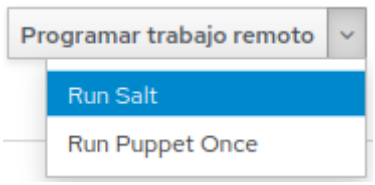
Cabe hacer notar que dichas variables son provistas a los equipos a través de los datos "*pillar*" de Salt y que pueden fijarse o particularizar desde los diferentes lugares anteriormente nombrados.

Aunque no está provisto un estado específico para realizar desinstalaciones de paquetería, de una forma similar y muy sencilla pueden generarse dichos estados viendo el código de los ficheros sls.

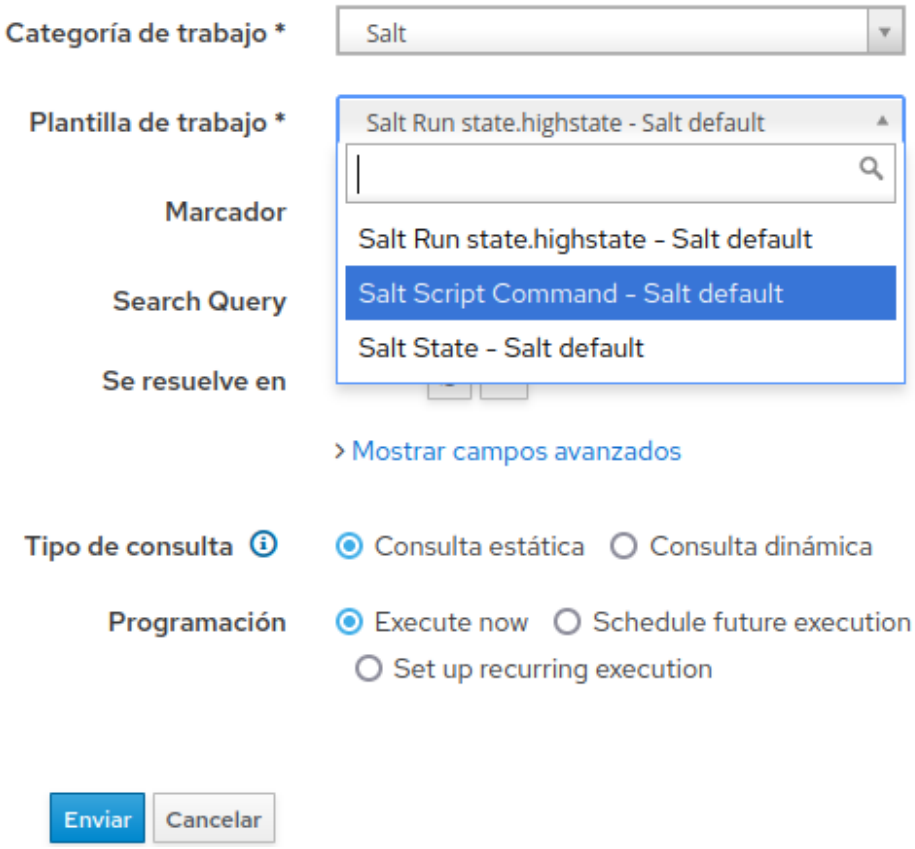
Ejecución arbitraria de comandos de sistema

A través de la interfaz de Foreman es posible realizar la aplicación de plantillas que a modo "interactivo" utilizan Salt como backend para su ejecución.

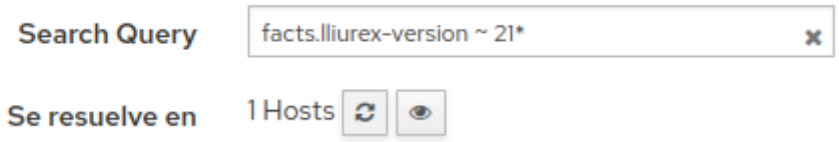
La tarea más básica que permite ser lanzada a través de Foreman es la ejecución del "*highstate*" disponible desde la vista de un equipo, utilizando la opción que se despliega desde *Programar trabajo remoto*.



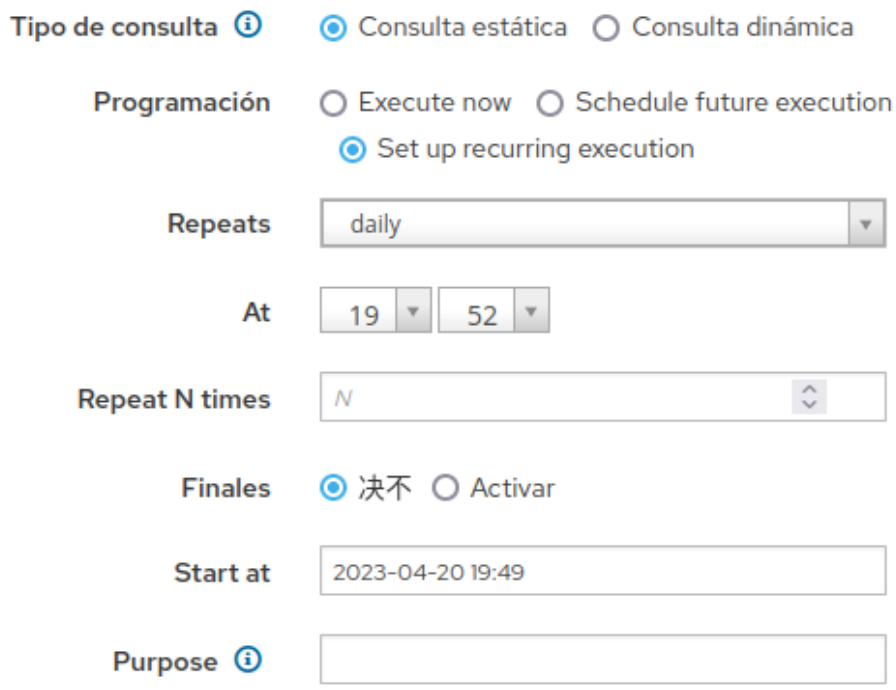
Si se desea especificar el tipo de "trabajo Foreman" que se va a ejecutar se debe seleccionar la opción de *Programar trabajo remoto*



Es interesante el uso del filtro y mostrar los equipos en los que se resuelve para limitar o seleccionar los equipos en los que debe ser aplicada este trabajo.



También es posible utilizar las opciones para programar trabajos recurrentes o que son ejecutados de una forma programada, aunque siempre es preferible asociar a los equipos un estado Salt.


Tipo de consulta
Consulta estática
Consulta dinámica
Programación
Execute now
Schedule future execution
Set up recurring execution
Repeats
daily
At
19
52
Repeat N times
N
Finales
决不
Activar
Start at
2023-04-20 19:49
Purpose

Las diferentes plantillas que ofrece el plugin Salt por defecto en Foreman son:

- Salt Script Command - Salt default

  Utilizando este tipo de trabajo se pasará lo que se escriba en la caja de texto "*script*" como un parámetro al estado Salt "*cmd.run*", es una forma rápida de ejecutar un comando sobre un conjunto de equipos de forma puntual, aunque los equipos deberían estar encendidos y con el canal de comunicación del agente funcionando.

- Salt State - Salt default

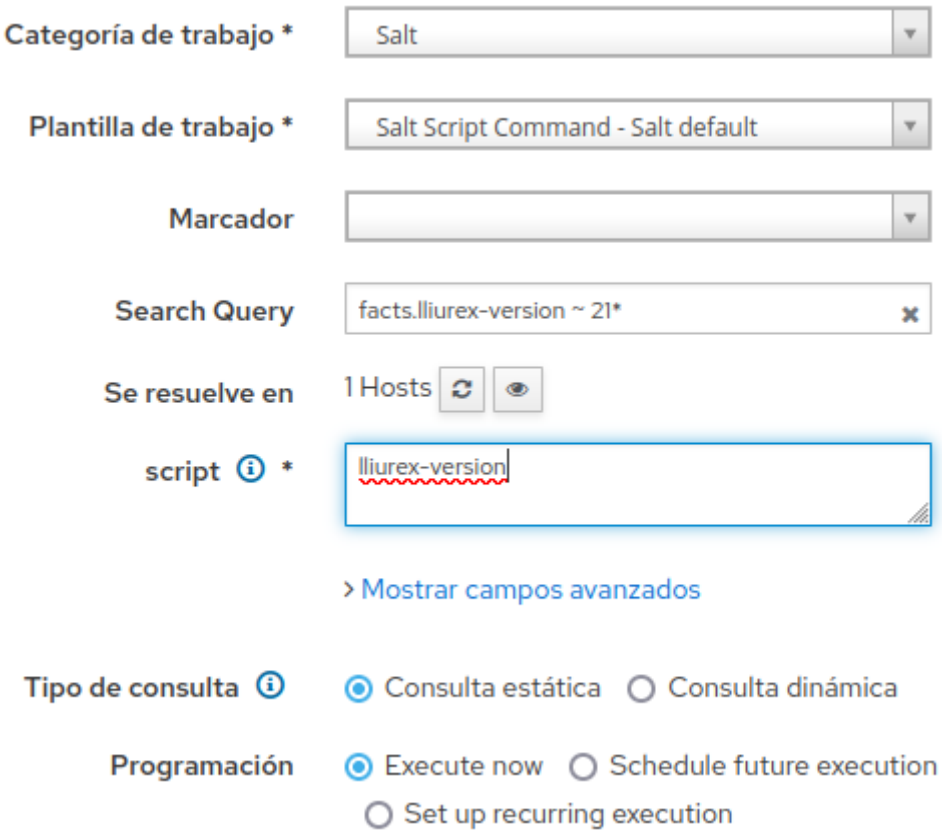
  Utilizando este tipo de trabajo debe ser escrito en la caja de texto la definición de una plantilla de estado completamente, puede no parecer tan versátil, pero ofrece muchas más posibilidades al no estar limitado al uso del estado "*cmd.run*", también es necesario que el equipo esté encendido y el agente funcionando correctamente.

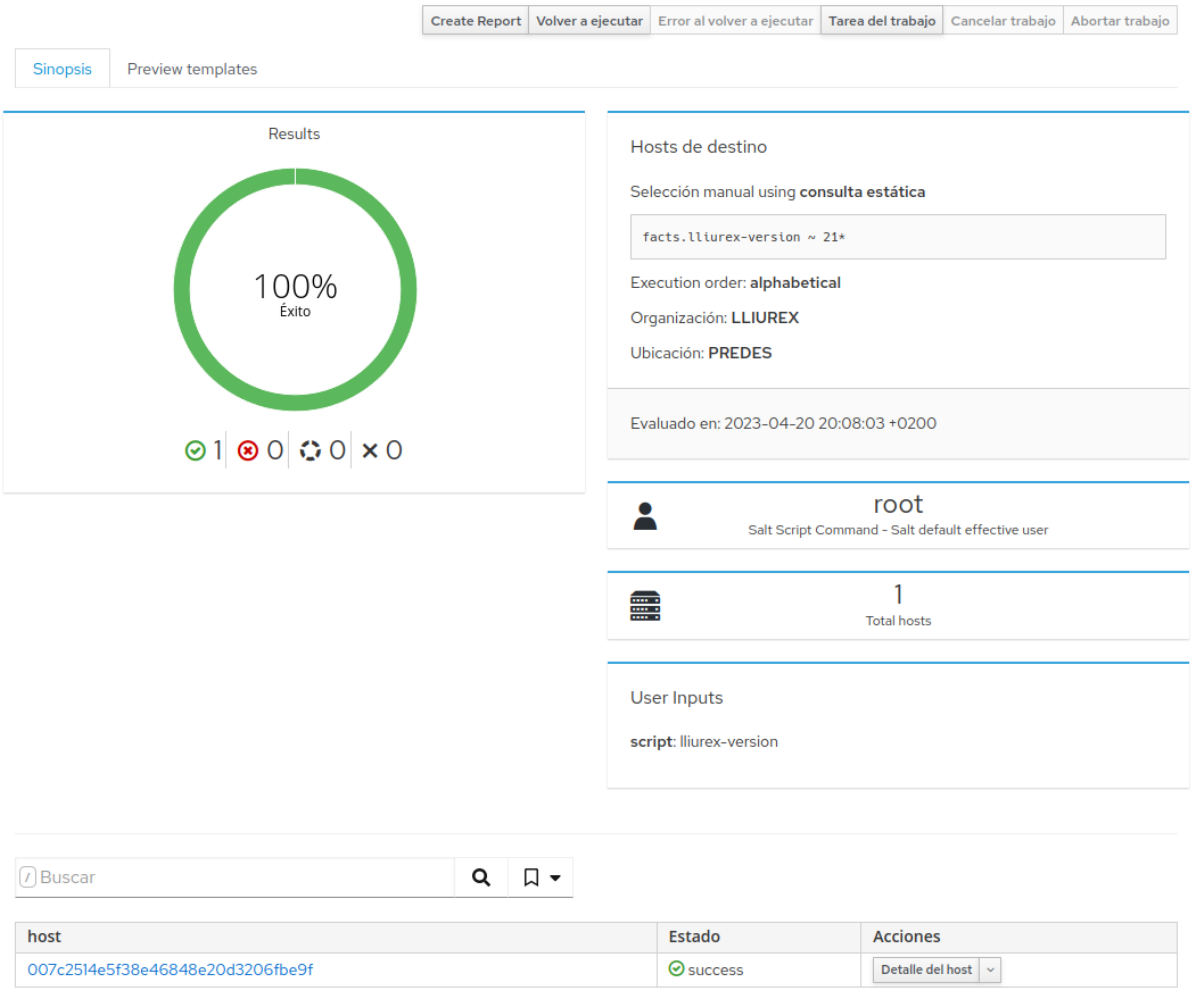
En los ejemplos propuestos se trata de utilizar y mostrar este método pues lo escrito en la caja de texto puede ser transcrito directamente en un fichero sls y ser utilizado asociándolo a algún elemento.

De forma inversa, también puede utilizarse este método para depurar estados que deben ser escritos como fichero sls.

Ejemplos sencillos de estos tipos de ejecuciones son:

- Ejecución directamente de un comando de sistema





Es importante darse cuenta que para obtener los detalles de la tarea Foreman y obtener la salida que esta produce se debe utilizar el enlace con el nombre del equipo que es mostrado al fondo de la página con éxito/fracaso.


Volver al trabajo
Volver a ejecutar
Alternar comando
Alternar STDERR
Alternar STDOUT
Alternar DEBUG
Detalles de tareas
Cancelar trabajo
Abortar trabajo
Destino: 007c2514e5f38e46848e20d3206fbe9f using Smart Proxy foreman.forelab.lan
Desplazarse hasta el final
jid: 20230420180804632533
007c2514e5f38e46848e20d3206fbe9f:

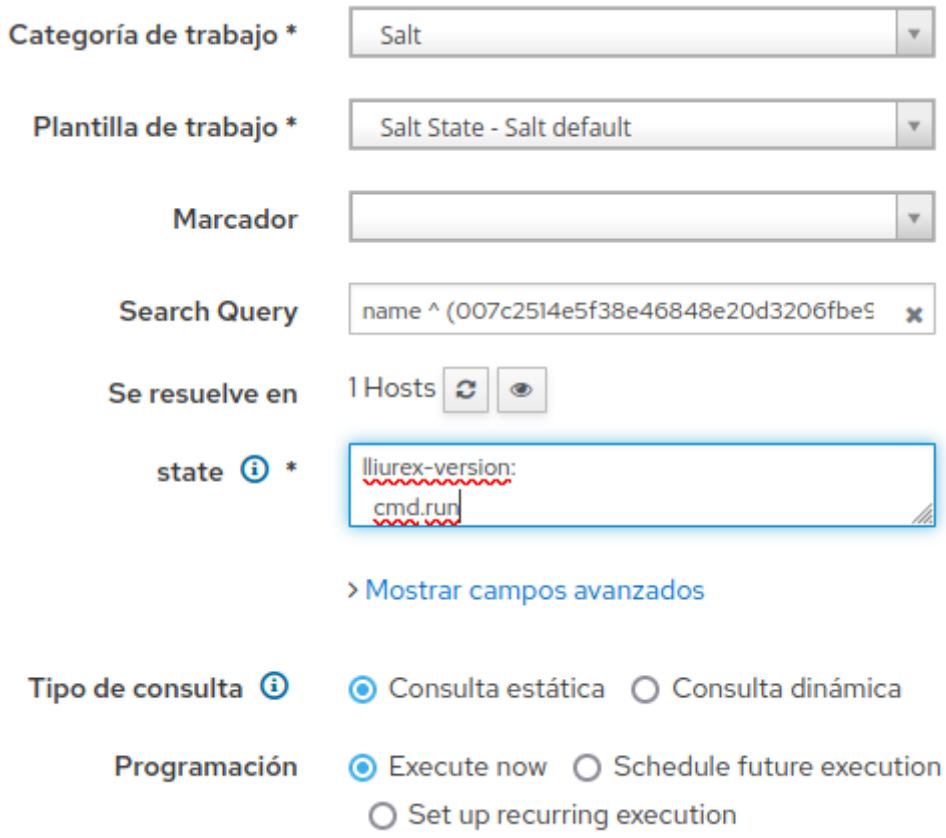
ID: execute script
Function: cmd.run
Name: lliurex-version
Result: True
Comment: Command "lliurex-version" run
Started: 20:08:09.594066
Duration: 112.394 ms
Changes:
----
pid:
216822
retcode:
0
stderr:
stdout:
desktop, edu, 21.230412
Summary for 007c2514e5f38e46848e20d3206fbe9f

Succeeded: 1 (changed=1)
Failed: 0

Total states run: 1
Total run time: 112.394 ms
Estado de salida: 0
Desplazarse hasta la parte superior

- Ejecutando directamente la definición de un estado

  En el ejemplo se muestra la definición de la plantilla equivalente al ejemplo anterior (con el mismo resultado) para un mayor entendimiento progresivo de la definición de estados, a través de plantillas similares es posible ejecutar cualquier comando de sistema a través del estado "*cmd.run*" u otras acciones diferentes como se muestra en los ejemplos posteriores.



Mantener el sistema actualizado

Para mantener el sistema operativo de los equipos administrados siempre actualizado con todas las actualizaciones publicadas en el repositorio configurado en el equipo deben ser realizadas dos acciones.

- Asegurarse que se actualiza la caché de paquetes desde internet
- Aplicar el estado correspondiente para lanzar la actualización de sistema

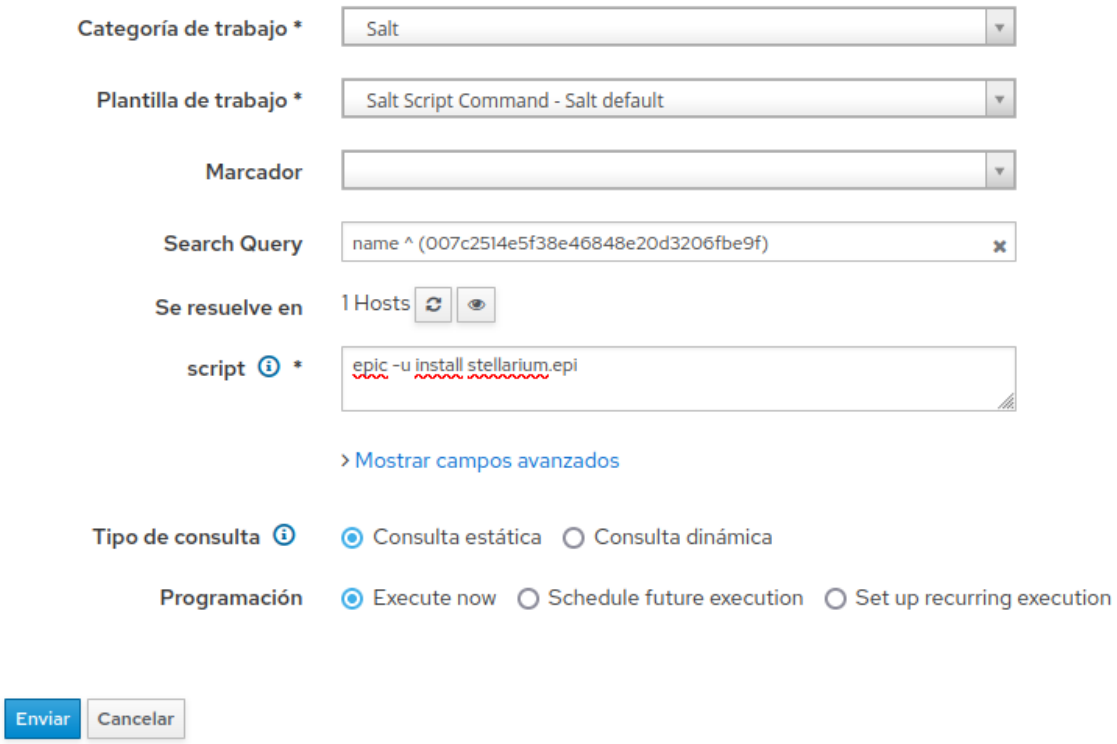
Estas dos acciones se proveen de forma independiente a través de los ficheros sls ubicados en el repositorio nombrados como "*update_repos.sls*" y "*always_packages_updated.sls*"

Estas plantillas hacen uso del módulo y estado "*pkg*" con las funciones "*pkg.refresh_db*" y "*pkg.uptodate*".

Cuando estas plantillas son asociadas a un conjunto de equipos, estos se mantienen actualizados.

Instalación de ficheros EPI (Zero-Center)

Para instalar elementos incluidos en Zero-Center se debe ejecutar directamente el comando de sistema operativo "*epic*" con los parámetros adecuados y el fichero EPI que debe estar en el equipo (de no ser así también será necesario instalar el paquete instalador que provee el fichero EPI, opciones interesantes serían el modo desatendido y la especificación de los elementos.Un ejemplo utilizando una tarea Foreman que ejecuta a través de "*cmd.run*" el comando "*epic -u install stellarium.epi*":



Nota: Dado que el instalador "*epic*" realiza una instalación incondicional del paquete, esté o no instalado, es conveniente introducir lógica para que la ejecución del *script* sea realizada únicamente cuando la comprobación del estado del paquete sea que no está instalado.

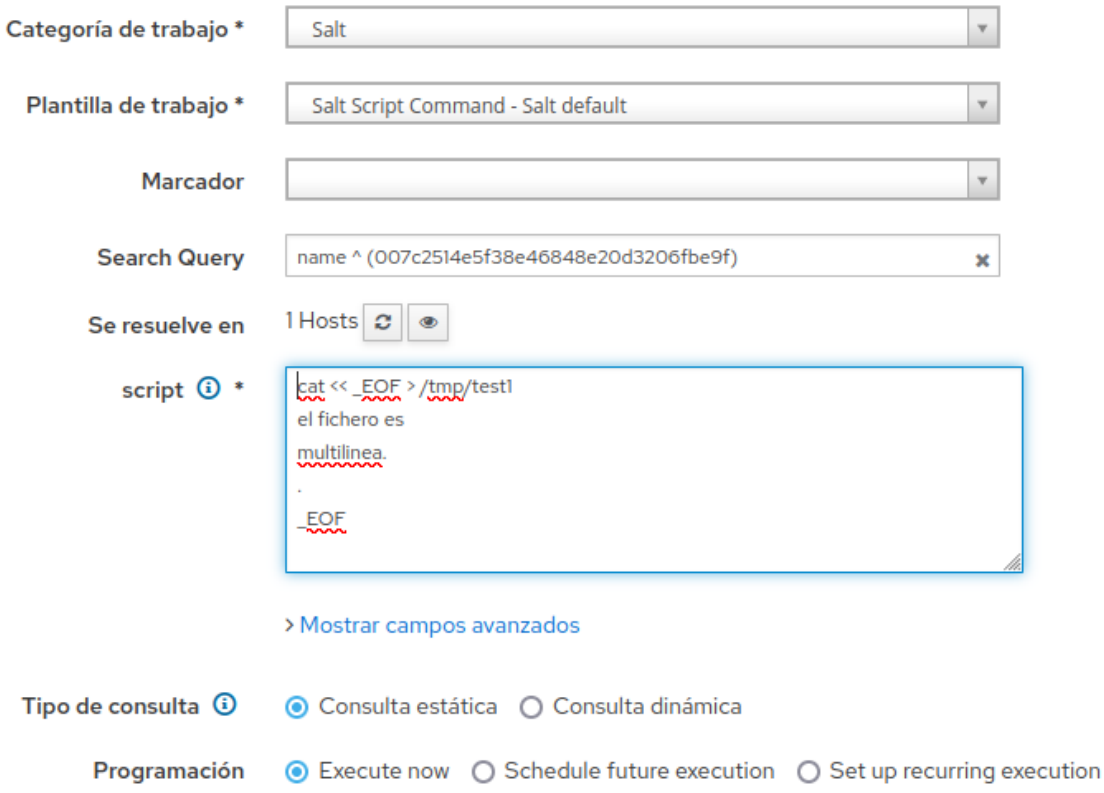
```
                        - Application: stellarium
                        - Status: availabled
                        - Uninstall process availabled: Yes
                      [EPIC]: Packages selected to install: stellarium
                      [EPIC]: Checking internet connection. Wait a moment...
                    User Groups: ['root', '*']
                    Reading package lists...
                    Building dependency tree...
                    Reading state information...
                    The following NEW packages will be installed:
                      stellarium
                    0 upgraded, 1 newly installed, 0 to remove and 99 not upgraded.
                    Need to get 0 B/7102 kB of archives.
                    After this operation, 18.4 MB of additional disk space will be used.
                    Selecting previously unselected package stellarium.
                    (Reading database ...
                    (Reading database ... 5%
                    (Reading database ... 10%
                    (Reading database ... 15%
                    (Reading database ... 20%
                    (Reading database ... 25%
                    (Reading database ... 30%
                    (Reading database ... 35%
                    (Reading database ... 40%
                    (Reading database ... 45%
                    (Reading database ... 50%
                    (Reading database ... 55%
                    (Reading database ... 60%
                    (Reading database ... 65%
                    (Reading database ... 70%
                    (Reading database ... 75%
                    (Reading database ... 80%
                    (Reading database ... 85%
                    (Reading database ... 90%
                    (Reading database ... 95%
                    (Reading database ... 100%
                    (Reading database ... 347568 files and directories currently installed.)
                    Preparing to unpack .../stellarium_0.19.3-1build1_amd64.deb ...
                    Unpacking stellarium (0.19.3-1build1) ...
                    Setting up stellarium (0.19.3-1build1) ...
                    Processing triggers for mime-support (3.64ubuntu1) ...[0
;0m
                    Processing triggers for hicolor-icon-theme (0.17-2) ...
                    Processing triggers for man-db (2.9.1-1) ...
                    Processing triggers for shared-mime-info (1.15-1) ...
                    Processing triggers for desktop-file-utils (0.24-1ubuntu3) ...
                    User Groups: ['root', '*']
                    User Groups: ['root', '*']
                    ****************************************************************
                    ******************* INSTALLING APPLICATION *******************
                    ****************************************************************
                      [EPIC]: Gathering information....
                      [EPIC]: Preparing installation...
                      [EPIC]: Installing application...
                      [EPIC]: Ending installation...
                      [EPIC]: Installation completed successfully

Summary for 007c2514e5f38e46848e20d3206fbe9f
------------
Succeeded: 1 (changed=1)
Failed:    0
------------
Total states run:     1
Total run time:  11.800 s
Estado de salida: 0
```

Para eliminar el paquete simplemente será utilizada la opción correspondiente para eliminar, "*epic –u uninstall stellarium.epi*"

Algunos de los ficheros EPI pueden agrupar múltiples aplicaciones si se desea instalar parcialmente se procedería mediante la ejecución de: "*epic -u install zero-installer-util-media.epi kde-service-menu-reimage conversor-kde-servicemenu*" o bien su correspondiente eliminación con la operación "*uninstall*" en el comando.

Modificación de ficheros

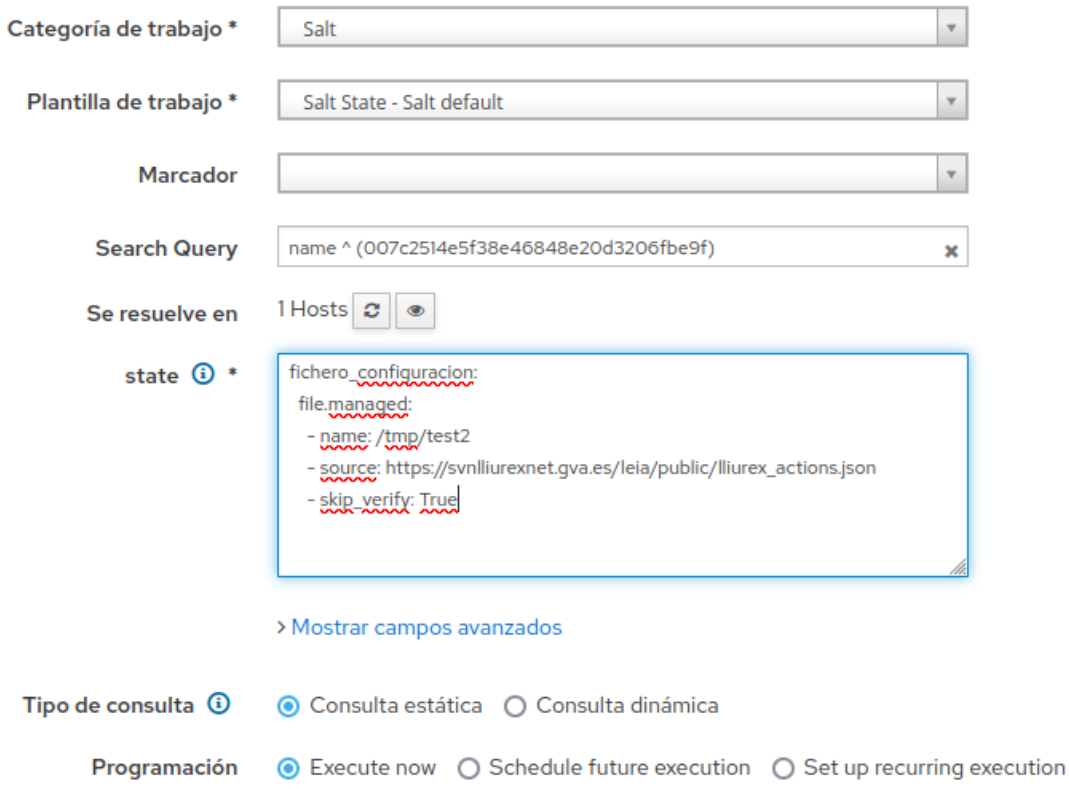
La forma más simple de modificar puntualmente un fichero puede ser realizada a través de la ejecución de un comando de sistema operativo con órdenes del intérprete de comandos.



```
jid: 20230418154219486785
007c2514e5f38e46848e20d3206fbe9f:
----------
          ID: execute script
    Function: cmd.run
        Name: cat << _EOF > /tmp/test1
el fichero es
multilinea.
.
_EOF
      Result: True
     Comment: Command "cat << _EOF > /tmp/test1
              el fichero es
              multilinea.
              .
              _EOF" run
     Started: 17:42:15.368083
    Duration: 13.095 ms
     Changes:
              ----------
              pid:
                  927782
              retcode:
                  0
              stderr:
              stdout:

Summary for 007c2514e5f38e46848e20d3206fbe9f
------------
Succeeded: 1 (changed=1)
Failed:    0
------------
Total states run:     1
Total run time:  13.095 ms
Estado de salida: 0
```

Para ejecutar una plantilla de estado que realice una modificación de un fichero puede ser utilizado el estado "*file-managed*" y las opciones que este presenta (permisos, usuarios…), no obstante, también puede ser interesante proveer el fichero desde un repositorio o bien una URL desde internet, el fichero puede ser obtenido remotamente y no es necesario teclear el contenido resultando una operación más limpia y sometida al posible control de versiones.

Un ejemplo que muestra cómo se crea un fichero con una fuente a partir de contenido en internet:




```
jid: 20230418154811832739
007c2514e5f38e46848e20d3206fbe9f:
----------
          ID: fichero_configuracion
    Function: file.managed
        Name: /tmp/test2
      Result: True
     Comment: File /tmp/test2 updated
     Started: 17:48:11.470838
    Duration: 1417.637 ms
     Changes:
              ----------
              diff:
                  New file
              mode:
                  0644

Summary for 007c2514e5f38e46848e20d3206fbe9f
------------
Succeeded: 1 (changed=1)
Failed:    0
------------
Total states run:     1
Total run time:   1.418 s
Estado de salida: 0
```

Otra opción interesante provista a través del fichero "*copy_resources.sls*" (aunque no aconsejable para contenido muy pesado) puede ser proveer ficheros a través del repositorio Git que el propio Salt utiliza, es

un caso similar al de obtener un fichero desde una URL, aunque en este caso para mostrar la funcionalidad se obtiene un directorio completo desde el servidor Foreman que provee el contenido en Git y es copiado al equipo al directorio */tmp*.

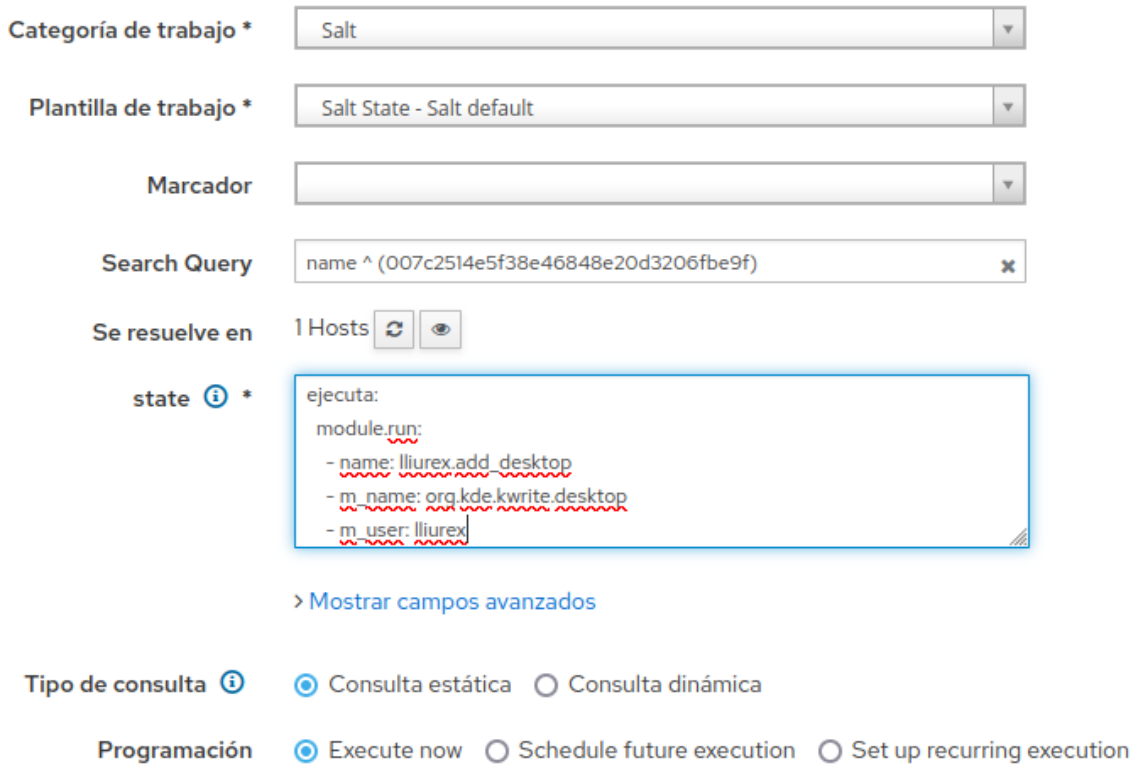
Creación de lanzadores en los escritorios de los usuarios

Para tareas más complejas como creación de lanzadores en los "homes" de los usuarios pueden escribirse estados o módulos personalizados que realicen las acciones correspondientes a través de la programación con Python e introducirlos en el repositorio de código Git, para ello son utilizados las carpetas "_states" y "_modules".

A través de los ejemplos propuestos se puede mostrar cómo crear plantillas que hagan uso de dichos estados o módulos, de forma similar pueden ser utilizados los otros estados o módulos incluidos.

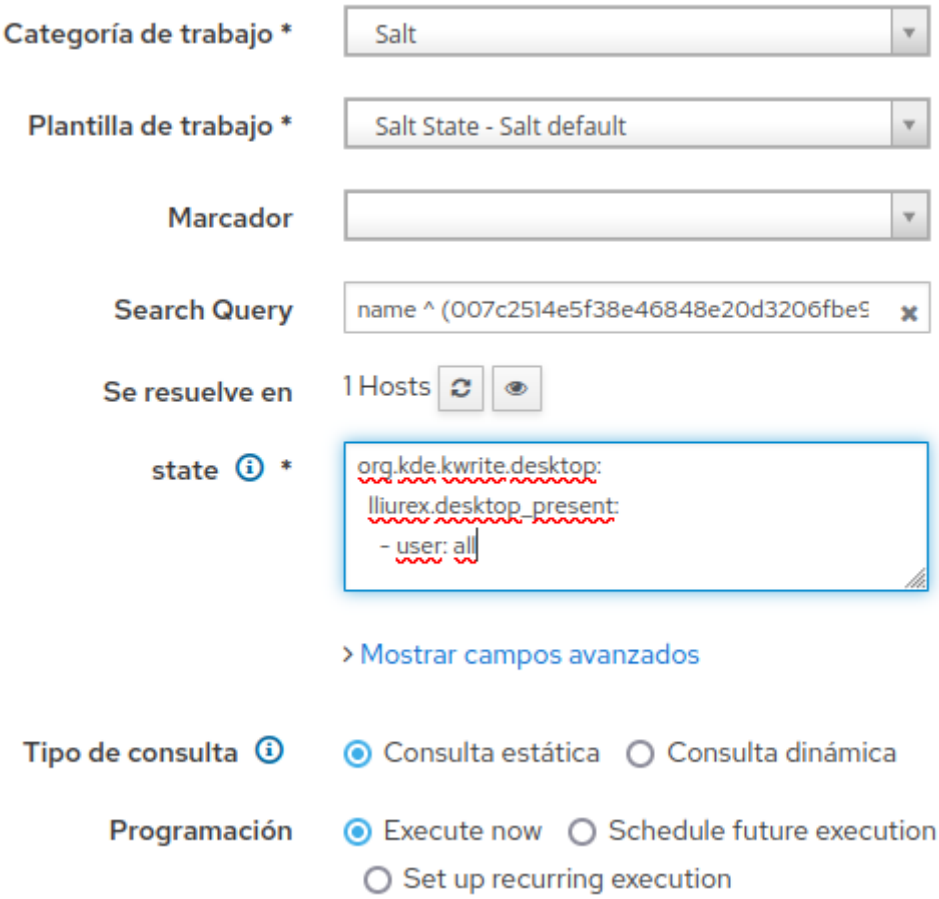
Para ser utilizadas estas plantillas y obtener su funcionalidad se pueden programar estados similares a estos:

- Ejecución de un módulo



```
jid: 20230420152723611016
007c2514e5f38e46848e20d3206fbe9f:
----------
          ID: ejecuta
    Function: module.run
        Name: lliurex.add_desktop
      Result: True
     Comment: Module function lliurex.add_desktop executed
     Started: 17:27:29.043962
    Duration: 18687.029 ms
     Changes:
              ----------
              ret:
                  True

Summary for 007c2514e5f38e46848e20d3206fbe9f
------------
Succeeded: 1 (changed=1)
Failed:    0
------------
Total states run:     1
Total run time:  18.687 s
Estado de salida: 0
```

- Ejecución del estado




```
jid: 20230420191515996945
007c2514e5f38e46848e20d3206fbe9f:
----------
          ID: org.kde.kwrite.desktop
    Function: lliurex.desktop_present
      Result: True
     Comment: Created for user/s: lliurex,lliurex2
     Started: 21:15:21.373848
    Duration: 12203.457 ms
     Changes:
              ----------
              result:
                  Created for user/s: lliurex,lliurex2

Summary for 007c2514e5f38e46848e20d3206fbe9f
------------
Succeeded: 1 (changed=1)
Failed:    0
------------
Total states run:     1
Total run time:  12.203 s
Estado de salida: 0
```

Acciones sobre los usuarios

Para ejecutar acciones sobre usuarios (y se recuerda que el sistema está planteado principalmente para usuarios SOLO a través de LDAP (en principio no es necesario crear usuarios)) tales como eliminación de directorios de usuario o limpieza de carpetas de caché puede realizarse a través de la ejecución de scripts que luego son ejecutados a través de Salt mediante la ejecución de comandos o directamente los comandos adecuados, no obstante se recomienda utilizar herramientas desarrolladas para tal efecto que puedan tener un mayor control de las tareas a realizar y simplemente ejecutar dichas herramientas periódicamente desde la programación Salt.

Anexos

top.sls

```
base:
 '*':
   - universe

main:
 '*':
   - universe
```

universe.sls

```
refresco_pillar:
 module.run:
   - name: saltutil.sync_all
   - refresh: True

highstate:
 schedule.present:
   - function: state.highstate
   - minutes: {{ pillar.get('schedule_time',60) }}
   - splay: 60
```

install_from_pillar.sls

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

install_by_tag.sls

```
{% for gra in grains['tags'] %}
{% for pkg in pillar.get('pkgs_'+gra,{}) %}
{{ pkg }}:
 pkg.installed
{% endfor %}
{% endfor %}
```

copy_resources.sls

```
/tmp/resources:
 file.recurse:
   - source: salt://resources
   - include_empty: True
```

update_repos.sls

```
update_packages:
 module.run:
   - name: pkg.refresh_db
```

always_packages_updated.sls

```
update_all_packages:
 pkg.uptodate:
   - name: '*'
   - refresh: True
   - require:
     - module: update_packages
```

_grains/llxversion.py

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

import subprocess
import os.path

def main():
   if os.path.exists('/usr/bin/dpkg-query'):
       version = subprocess.check_output([
             '/usr/bin/dpkg-query',
             '-W',
             '-f',
             "${Version}",
             'lliurex-version-timestamp']).decode()
       if isinstance(version,(list,dict)):
           version = str(version)
       return {'lliurex-version':str(version)}
```

_grains/pkg_list.py

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

import subprocess
import os

def main():
   if os.path.exists('/usr/bin/dpkg-query'):
       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 }
```

_grains/tags_list.py

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

import os

def main():
    if os.path.exists('/etc/lliurex-auto-upgrade/tags'):
        try:
            return {'tags':os.listdir('/etc/lliurex-auto-upgrade/tags')}
        except Exception as e:
            return {'tags':[str(e)]}
```

_grains/uptime.py

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

import subprocess

def main():
    try:
        return {'uptime':subprocess.check_output('uptime').decode().split()[0]}
    except Exception as e:
        return {'uptime':str(e)}
```

_modules/lliurex.py

```
#
# Example proof-of-concept minimal module for copy desktop app to any home
#
import subprocess
import os

def add_desktop(name, user='all'):
    created = []
    try:
        filename='/usr/share/applications/'+name
        if not os.path.isfile(filename):
            raise Exception('File {} not available'.format(filename))
        if user != 'all':
            dirname=subprocess.check_output(
                     "su {} - -c 'xdg-user-dir DESKTOP'".format(user),
                     shell=True,
                     stderr=subprocess.STDOUT)
            try:
                dirname=dirname.decode()
            except:
                pass
            dirname=dirname.strip()
            if not dirname or not os.path.isdir(dirname):
                raise Exception(
'usererror: user {} with desktop {} not available'.format(user,dirname))
            outmsg = subprocess.check_output(
                     "cp {} {}".format(filename,dirname),
                     shell=True,
                     stderr=subprocess.STDOUT)
            created.append(user)
        else:
            out = []
            for us in os.listdir('/home'):
                dirname=subprocess.check_output(
                             "su {} - -c 'xdg-user-dir DESKTOP' -s /bin/sh".format(us),
                             shell=True,
                             stderr=subprocess.STDOUT)
                try:
                    dirname=dirname.decode()
                except:
                    pass
                dirname=dirname.strip()
                if not dirname or not os.path.isdir(dirname):
                    continue
                outmsg = subprocess.check_output(
                             "cp {} {}".format(filename,dirname),
                             shell=True,
                             stderr=subprocess.STDOUT)
                created.append(us)
                if outmsg:
                    out.append(outmsg)
            if out:
                out = ','.join(out)
        return True
    except Exception as e:
        return False
```

_states/lliurex.py

```
import subprocess
import os.path
import os


def desktop_present(name, user='all'):
    ret = {"name": name, "result": False, "comment": "", "changes": {}}
    created = []
    try:
        filename='/usr/share/applications/'+name
        if not os.path.isfile(filename):
            raise Exception('File {} not available'.format(filename))
        if user != 'all':
            dirname=subprocess.check_output(
                     "su {} - -c 'xdg-user-dir DESKTOP'".format(user),
                     shell=True,
                     stderr=subprocess.STDOUT)
            try:
                dirname=dirname.decode()
            except:
                pass
            dirname=dirname.strip()
            if not dirname or not os.path.isdir(dirname):
                raise Exception(
'usererror: user {} with desktop {} not available'.format(user,dirname))
            outmsg = subprocess.check_output(
                     "cp {} {}".format(filename,dirname),
                     shell=True,
                     stderr=subprocess.STDOUT)
            created.append(user)
        else:
            out = []
            for us in os.listdir('/home'):
                dirname=subprocess.check_output(
                             "su {} - -c 'xdg-user-dir DESKTOP' -s /bin/sh".format(us),
                             shell=True,
                             stderr=subprocess.STDOUT)
                try:
                    dirname=dirname.decode()
                except:
                    pass
                dirname=dirname.strip()
                if not dirname or not os.path.isdir(dirname):
                    continue
                outmsg = subprocess.check_output(
                             "cp {} {}".format(filename,dirname),
                             shell=True,
                             stderr=subprocess.STDOUT)
                created.append(us)
                if outmsg:
                    out.append(outmsg)
            if out:
                out = ','.join(out)
        ret['result'] = True
    except Exception as e:
        out = str(e)
        ret['result'] = False
    if ret['result'] == True:
        ret['comment'] = 'Created for user/s: {}'.format(','.join(created))
    else:
        ret['comment'] =
              'Created for user/s: {}\nNote: {}'.format(','.join(created),out)
    ret['changes'] = {"result": ret['comment']}
    return ret
```

install_vscode.sls

```
instala_vscode:
 cmd.run:
   - name: dpkg-query -W -f '${Status}' code 2>/dev/null | egrep -q '^install ok' || epic -u
            install zero-fp-informatica.epi code
```

always_packages_updated2.sls

update_all_packages:

```
 module.run:
   - name: pkg.upgrade
   - dist_upgrade: True
   - refresh: True
```

official_sources_list.sls

```
/etc/apt/sources.list:
 file.managed:
   - source: salt://configs/sources.list
   - skip_verify: True
```