



# The BackStage

RH Forum 2018

By Moisés Rivera

Team Lead for Cloud, Automation and Infrastructure

# The BackStage

## ¿QUE ES THE BACKSTAGE?

The Backstage es el laboratorio de Ansible preparado para su realización en el Red Hat Forum 2018.

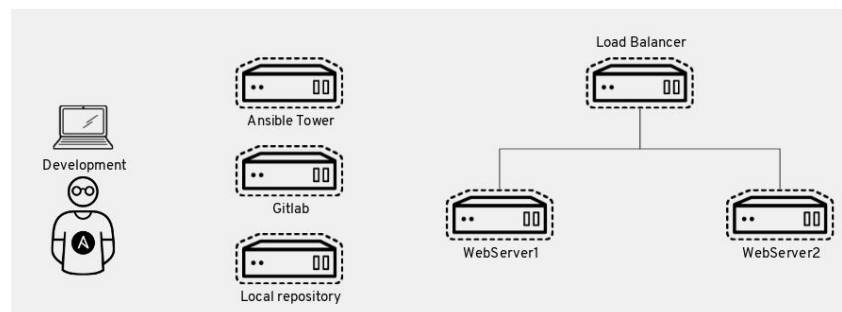
Durante las dos horas siguientes vamos a realizar todas las operaciones necesarias para crear el entorno de *demo*, utilizado por los Solution Architect de Red Hat, y que posiblemente hayas visto en alguna de sus presentaciones sobre Ansible.

Prepárate, pues vamos a trabajar principalmente con Ansible Tower, pero también vamos a realizar operaciones con Gitlab, crearemos workflows, surveys, credenciales... y algún que otro playbook :)

## ENTORNO DE TRABAJO

Para comenzar a trabajar, vamos a disponer de un laptop, donde estás leyendo este documento :) con una serie de máquinas virtuales ya creadas para poder ganar tiempo y así poder realizar todas las operaciones que tenemos por delante.

El siguiente esquema muestra todos los elementos que vamos a utilizar.



La siguiente tabla muestra los datos de cada uno de los elementos:

FQDN	IP	RoI
localrepo.rhforum.com	192.168.110.160	Repositorio local con todo el software necesario para realizar el laboratorio.
gitlab.rhform.com	192.168.110.161	Instancia local de Gitlab que utilizaremos como SCM del laboratorio.
tower.rhforum.com	192.168.110.162	Instancia de Ansible Tower.
dev.rhforum.com	192.168.110.1	Máquina de desarrollo y host de tiene todas las VMs.
lb.rhforum.com	192.168.110.163	Load balancer y punto de entrada a los Web Servers.
web1.rhforum.com	192.168.110.164	Web Server #1
web2.rhforum.com	192.168.110.165	Web Server #2

Además de la información anterior, vamos a necesitar las credenciales que vamos a utilizar durante la realización del laboratorio. La siguiente tabla muestra la información de usuarios y password necesarios.

Usuario	Password	RoI
rhforum	rhforum	Usuario de trabajo estándar.
rhforum	rhforum123.	Usuario para utilizar en GitLab
admin	rhforum	Usuario administrador de Ansible Tower.

Todas las máquinas tienen desactivado el acceso directo como root vía SSH, pero en caso de necesitarlo, comentarlo para verificar si es estrictamente necesario.

A priori, estos son los datos que necesitamos para ponernos a trabajar; por lo que vamos adelante, no? :)

## METODOLOGÍA

La base metodológica en la que se basan los laboratorios es IaC (aka *Infrastructure as Code*).

From Wikipedia. [https://en.wikipedia.org/wiki/Infrastructure\\_as\\_Code](https://en.wikipedia.org/wiki/Infrastructure_as_Code)

**Infrastructure as code** (IaC) is the process of managing and provisioning computer [data centers](#) through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.<sup>[1]</sup> The [IT infrastructure](#) managed by this comprises both physical equipment such as [bare-metal servers](#) as well as [virtual machines](#) and associated configuration resources. The definitions may be in a [version control system](#). It can use either scripts or declarative definitions, rather than manual processes, but the term is more often used to promote declarative approaches.

IaC approaches are promoted for [cloud computing](#), which is sometimes marketed as [infrastructure as a service](#) (IaaS). IaC supports IaaS, but should not be confused with it.<sup>[1]</sup>

Si resumimos esta aproximación, básicamente tenemos:

- Definición de toda nuestra infraestructura, incluidos servicios, en ficheros de configuración de texto plano.
- Utilización de una herramienta - Ansible - para el despliegue automatizado y orquestado de los servicios definidos.
- Utilización de una única *fuentes de verdad* - repositorio de código - como origen y depositario del *know-how* de nuestro proyecto.

Ciertamente, realizaremos operaciones manuales, ya que las herramientas necesitan de una configuración mínima para poder comenzar a trabajar... qué fue antes, la gallina o el huevo? :)

## LABORATORIOS

Durante las dos horas restantes, vamos a intentar realizar las siguientes acciones:

- Despliegue del entorno de producción (aka, loadbalancers y webservers). Tareas de día 1.
- Modificación de configuraciones en el entorno de producción y subida de contenido. Tareas de día 2.
- BONUS TRACK: Modificación del acceso SSH a las máquinas de producción.

Para poder realizar todas las operaciones partimos de la infraestructura creada, el código Ansible ya escrito, y la gran mayoría de elementos configurados. Si el lector tiene interés en cómo se ha realizado, por favor, no deje de preguntar ;)

Los laboratorios están pensados para realizarse en un orden secuencial. Una vez realizados, podemos interaccionar con ellos de la manera de deseemos.

## LAB #0. CLONADO DEL REPOSITORIO DE CÓDIGO.

Este laboratorio no es absolutamente necesario, pero sí que muy recomendado para poder realizar la inspección del código.

Como el lector conoce, existen muchos y diversos repositorios de código, y no es una herramienta *nueva*, este tipo de herramientas llevan siendo utilizadas por los desarrolladores, desde tiempos inmemoriales :) pero ahora nos toca a *los de sistemas* también utilizar este tipo de herramientas, así que sin miedo alguno, allá vamos :)

Podemos decir que - actualmente - el repositorio *estandar* es GitHub (<https://github.com/>), pero nada más lejos de la realidad, ya que lo más importante, es que escojamos el repositorio que escojamos, es que sean *compatibles* entre diferentes *marcas*; y ahí es donde entra Git.

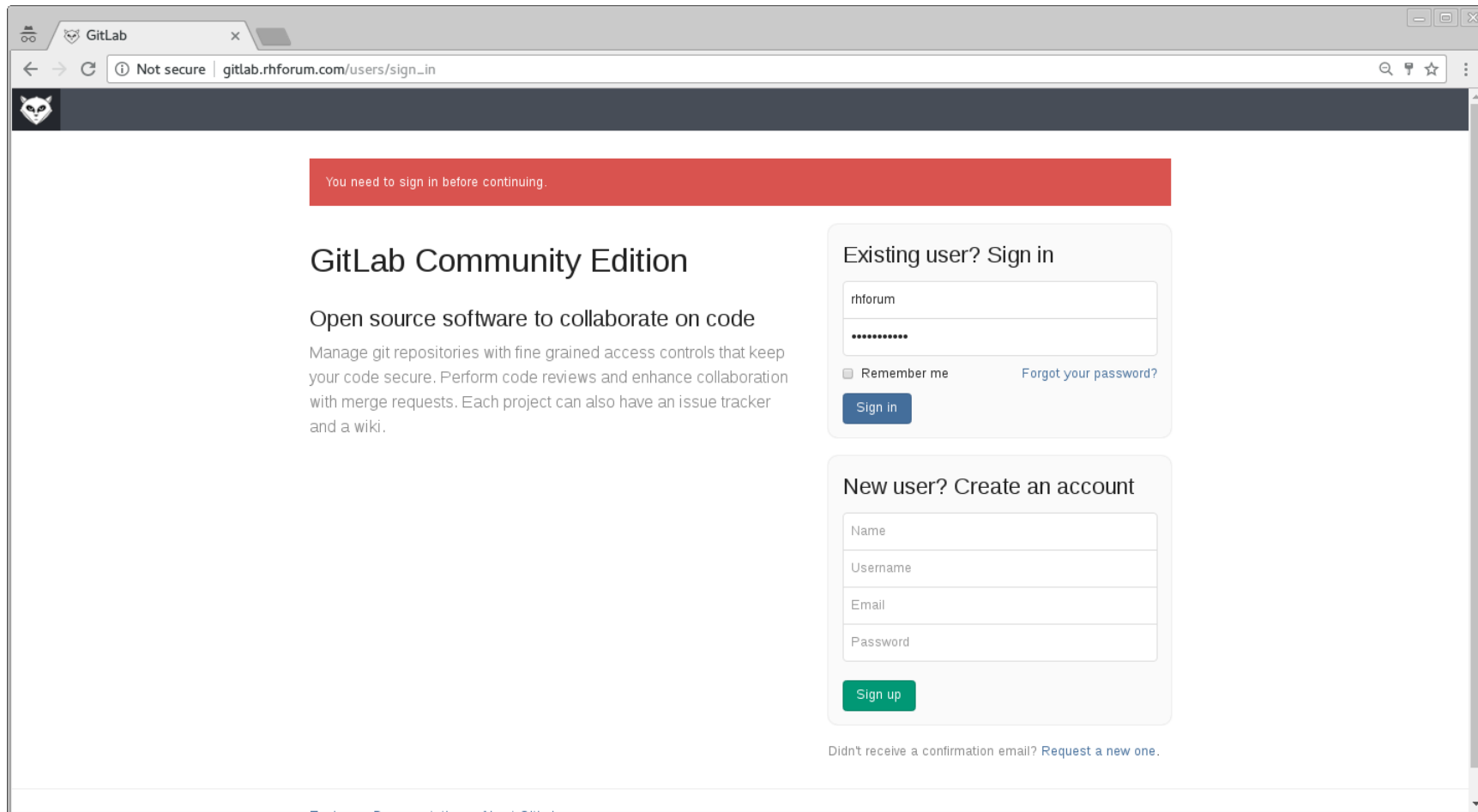
Nosotros para este laboratorio hemos escogido la versión de comunidad de GitLab <https://about.gitlab.com/>. Hemos descargado esta versión, y la hemos instalado en una de nuestras VMs.

Para poder entrar en la herramienta, abriremos un navegador, y tecleamos la dirección del mismo, que en nuestro caso es:

```
http://gitlab.rhforum.com
```

y nos aparecerá una pantalla como la que se muestra en la página siguiente. Ahora utilizaremos las credenciales que se han comentado al comienzo del documento para poder hacer login con nuestro usuario del laboratorio.

```
User: rhforum  
Pass: rhforum123.
```



The screenshot shows a web browser window with the address bar displaying "gitlab.rhforum.com/users/sign\_in". The page features a red banner at the top stating "You need to sign in before continuing.". Below this, the heading "GitLab Community Edition" is followed by the tagline "Open source software to collaborate on code" and a descriptive paragraph. On the right, there are two main sections: "Existing user? Sign in" and "New user? Create an account". The sign-in section includes a username field with "rhforum", a password field with masked characters, a "Remember me" checkbox, a "Forgot your password?" link, and a "Sign in" button. The account creation section includes fields for Name, Username, Email, and Password, along with a "Sign up" button. At the bottom, a link "Request a new one." is provided for users who didn't receive a confirmation email.

You need to sign in before continuing.

## GitLab Community Edition

Open source software to collaborate on code

Manage git repositories with fine grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

### Existing user? Sign in

rhforum

.....

☐ Remember me [Forgot your password?](#)

Sign in

### New user? Create an account

Name

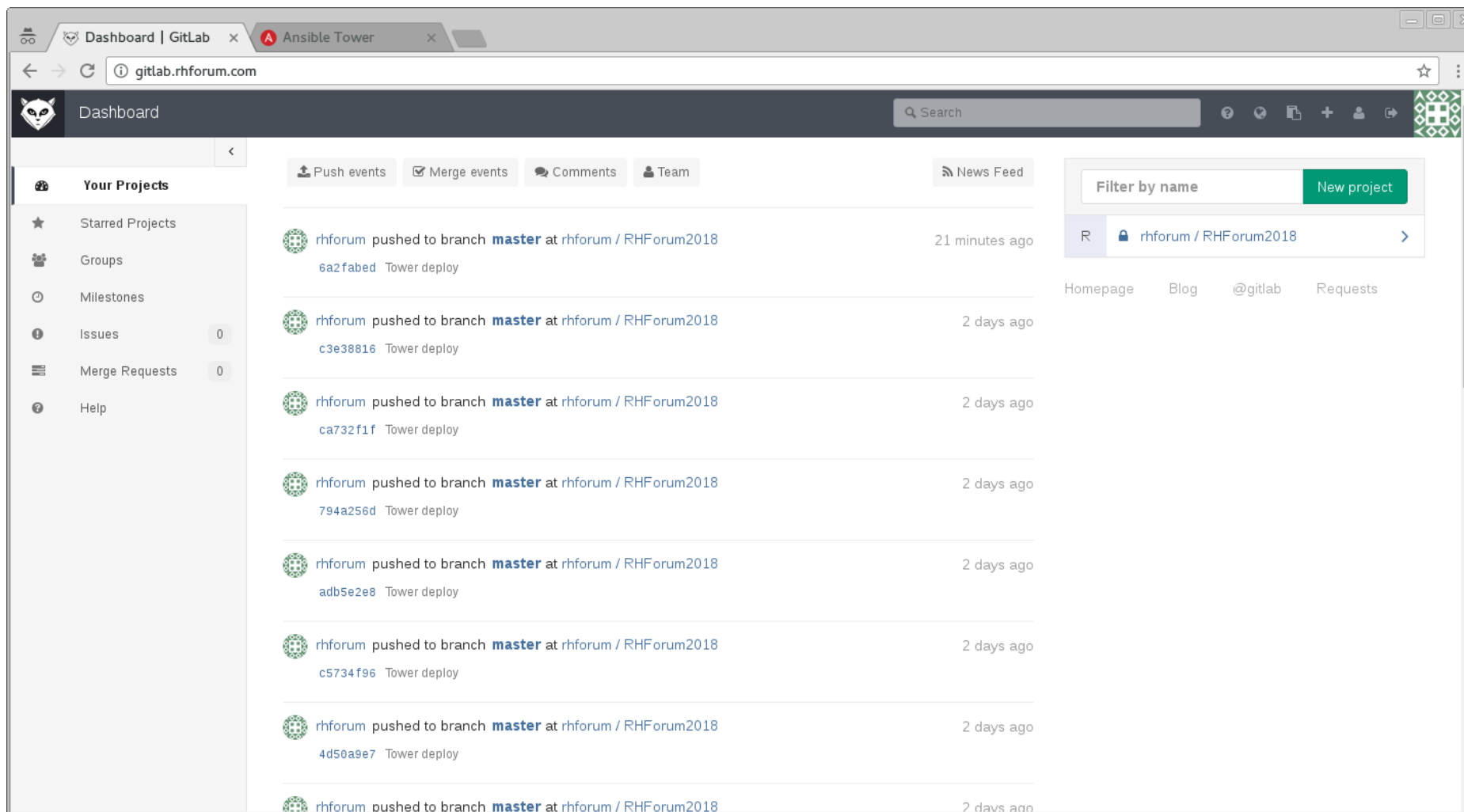
Username

Email

Password

Sign up

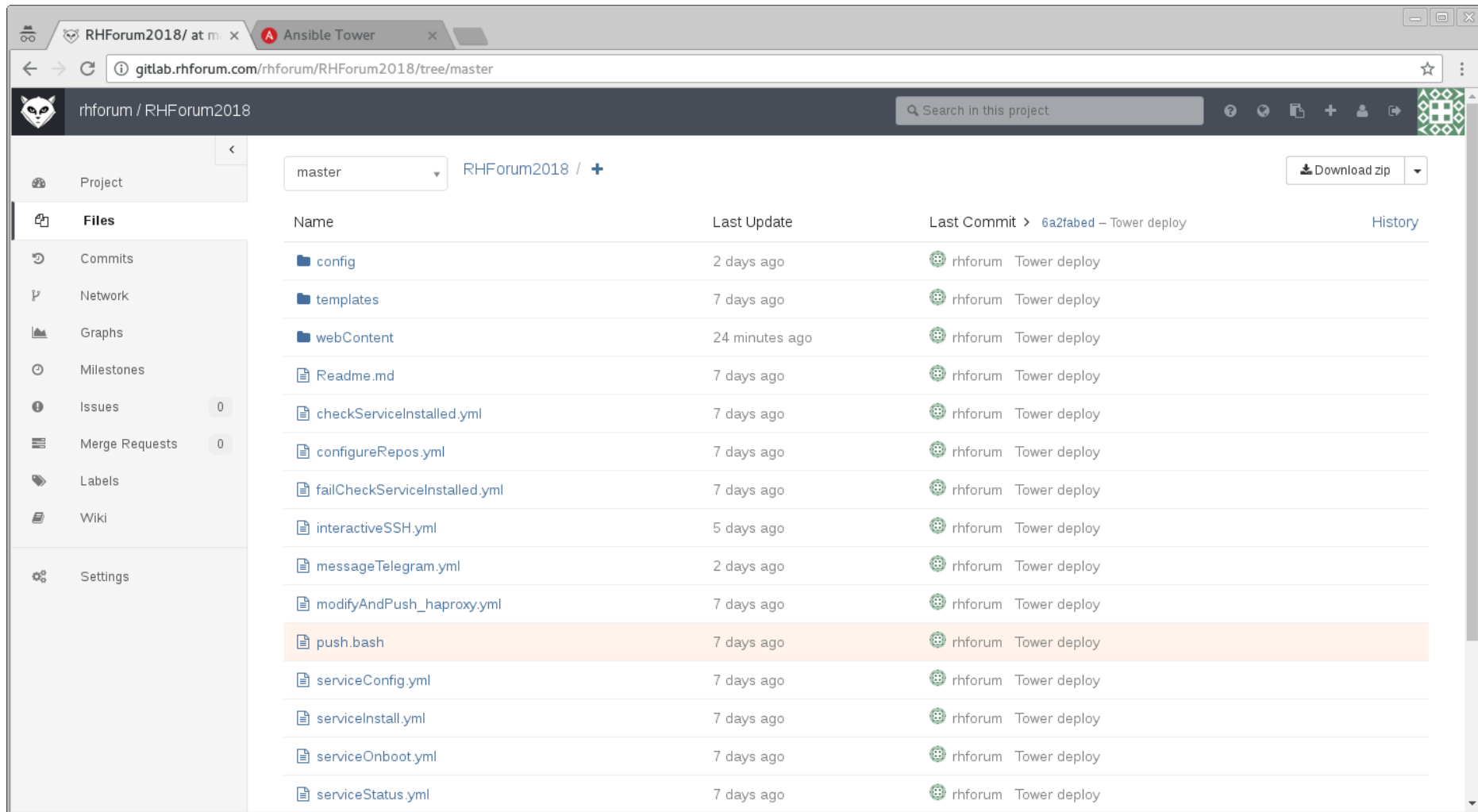
Didn't receive a confirmation email? [Request a new one.](#)



The screenshot shows the GitLab dashboard for the user 'gitlab.rhforum.com'. The left sidebar contains navigation links: 'Your Projects', 'Starred Projects', 'Groups', 'Milestones', 'Issues' (0), 'Merge Requests' (0), and 'Help'. The main content area displays a list of project events for the 'rhforum / RHForum2018' project. The events are filtered by 'Push events' and show multiple 'pushed to branch master' actions, each followed by a 'Tower deploy' status. The events are timestamped as '21 minutes ago' and '2 days ago'. On the right, there is a search bar, a 'Filter by name' dropdown, and a 'New project' button. Below these, there are links for 'Homepage', 'Blog', '@gitlab', and 'Requests'.

Event	Time
rhforum pushed to branch master at rhforum / RHForum2018 6a2fabed Tower deploy	21 minutes ago
rhforum pushed to branch master at rhforum / RHForum2018 c3e38816 Tower deploy	2 days ago
rhforum pushed to branch master at rhforum / RHForum2018 ca732f1f Tower deploy	2 days ago
rhforum pushed to branch master at rhforum / RHForum2018 794a256d Tower deploy	2 days ago
rhforum pushed to branch master at rhforum / RHForum2018 adb5e2e8 Tower deploy	2 days ago
rhforum pushed to branch master at rhforum / RHForum2018 c5734f96 Tower deploy	2 days ago
rhforum pushed to branch master at rhforum / RHForum2018 4d50a9e7 Tower deploy	2 days ago
rhforum pushed to branch master at rhforum / RHForum2018	2 days ago

Como podemos observar, ya tenemos creado un proyecto, *RHForum2018*. Si hacemos click en el link que aparece en la parte derecha, entramos en el proyecto, y nos aparece un nuevo menú a la derecha. Haciendo click en **Files**, podemos observar los diferentes ficheros que lo conforman.



Name	Last Update	Last Commit	History
config	2 days ago	rhforum Tower deploy	
templates	7 days ago	rhforum Tower deploy	
webContent	24 minutes ago	rhforum Tower deploy	
Readme.md	7 days ago	rhforum Tower deploy	
checkServiceInstalled.yml	7 days ago	rhforum Tower deploy	
configureRepos.yml	7 days ago	rhforum Tower deploy	
failCheckServiceInstalled.yml	7 days ago	rhforum Tower deploy	
interactiveSSH.yml	5 days ago	rhforum Tower deploy	
messageTelegram.yml	2 days ago	rhforum Tower deploy	
modifyAndPush_haproxy.yml	7 days ago	rhforum Tower deploy	
push.bash	7 days ago	rhforum Tower deploy	
serviceConfig.yml	7 days ago	rhforum Tower deploy	
serviceInstall.yml	7 days ago	rhforum Tower deploy	
serviceOnboot.yml	7 days ago	rhforum Tower deploy	
serviceStatus.yml	7 days ago	rhforum Tower deploy	

La primera acción que vamos a realizar es el clonado del mismo en nuestro *host* para poder inspeccionar su contenido, aunque evidentemente, tambien podemos hacerlo directamente desde GitLab.



La primera operación a realizar es la de crear un llave SSH y meterla dentro de nuestro perfil de usuario en GitLab; de esta manera, no nos estará pidiendo passwords si realizamos algún *push*.

Para generar la llave SSH, debemos proceder de la siguiente manera:

```
[rhforum@th88 ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/rhforum/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/rhforum/.ssh/id_rsa.
Your public key has been saved in /home/rhforum/.ssh/id_rsa.pub.
The key fingerprint is:
77:a0:41:d9:0b:51:e2:75:f4:7c:e8:d8:9e:e0:27:a2 rhforum@th88
The key's randomart image is:
+--[ RSA 2048 ]-----+
|           +=o.o      |
|          ooo.. o .   |
|           o... + .   |
|           o.. + .   |
|          S . + o     |
|           . o o .    |
|           . o +     |
|           . . o     |
|          E           |
+-----+

```

Una vez generada, hacemos un *cat* de la llave pública, la copiamos y la metemos en GitLab. Para realizar esta última operación, una vez estemos dentro de GitLab, en la esquina superior derecha, aparece un icono similar a una persona, donde podemos editar nuestro perfil. Si hacemos click, nos aparece en la parte derecha un nuevo menú, donde están las *SSH Keys*. Haciendo click sobre esta opción, podemos añadir nuestra nueva llave SSH.

Para realizar el clonado, abriremos un terminal desde nuestra máquina, y ejecutaremos los siguientes comandos:

```
[rhforum@dev ~]$ git clone http://gitlab.rhforum.com/rhforum/RHForum2018.git
Cloning into 'RHForum2018'...
Username for 'http://gitlab.rhforum.com': rhforum
Password for 'http://rhforum@gitlab.rhforum.com': rhforum123.
```

```
remote: Counting objects: 580, done.
remote: Compressing objects: 100% (330/330), done.
remote: Total 580 (delta 353), reused 388 (delta 239)
Receiving objects: 100% (580/580), 119.02 KiB | 0 bytes/s, done.
Resolving deltas: 100% (353/353), done.
Checking connectivity... done.
[rhforum@dev ~]$ cd RHForum2018/
[rhforum@dev RHForum2018]$ git config user.name rhforum
[rhforum@dev RHForum2018]$ git config user.email rhforum@domain.com
[rhforum@dev RHForum2018]$ git remote set-url --add origin git@gitlab.rhforum.com:rhforum/RHForum2018.git
[rhforum@dev RHForum2018]$ git remote set-url --delete origin http://gitlab.rhforum.com/rhforum/RHForum2018.git
[rhforum@dev RHForum2018]$ cat .git/config
[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
[remote "origin"]
    fetch = +refs/heads/*:refs/remotes/origin/*
    url = git@gitlab.rhforum.com:rhforum/RHForum2018.git
[branch "master"]
    remote = origin
    merge = refs/heads/master
[user]
    name = rhforum
    email = rhforum@domain.com
```

Es cierto que el lector puede considerar realizar estas operaciones de otra manera, pero estas - a priori - son válidas para nuestro laboratorio.

Desde este momento, ya podemos realizar la inspección del código que vamos a utilizar en el laboratorio.

## LAB #1. DEPLOYMENT. TAREAS DE DÍA 1.

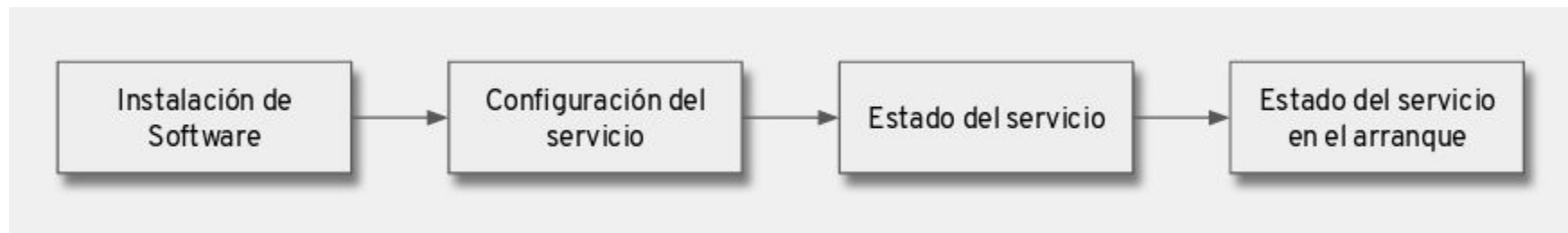
Tal y como hemos comentado al comienzo del lab, nuestra orientación metodológica será *laC*, lo que significa, entre otras muchas cosas, que debemos plantearnos cómo vamos a parametrizar/modelar nuestros servicios y operaciones; así las premisas sobre las que se ha construido este laboratorio son:

- Cualquier servicio debe poder ser descrito mediante un fichero de configuración.
- Descomposición del servicio en tareas reutilizables.
- Creación de tareas más *sofisticadas* en base a las tareas atómicas.

A priori, la instalación de un servicio - entendido como un servicio unitario - lo podríamos descomponer en los siguientes pasos:

- Instalación del software.
  - Configuración de los repositorios para acceso al software.
  - Instalación propiamente dicha del software.
- Configuración del servicio.
- Estado del servicio una vez realizado el deployment.
- Estado del servicio en el arranque de la máquina.

Evidentemente, todos estos pasos deben ser orquestados para tener la certeza que el paso  $n+1$  se realiza, si y sólo si el paso  $n$  ha terminado correctamente; algo parecido a lo siguiente:



Bien, una vez tenemos definidos los pasos que vamos a realizar, lo primero, es *definir* el servicio en un fichero de configuración; a continuación se muestra el fichero de uno de los servicios que vamos a desplegar en el laboratorio:

```
---
# Software repo config
service_software_repo_name: localrepo
service_software_repo_url: http://localrepo.rhforum.com/

# Service name string and service OS name
service_name: NGINX
service_os_name: rh-nginx18-nginx

# Software needed to install the service
service_pkgs: ['rh-nginx18']

# Config file vars. We need the template and the destination file
service_config_main_file: /etc/opt/rh/rh-nginx18/nginx/nginx.conf

service_config_files:
  - { template: ./templates/nginx_template.conf.j2, config: /etc/opt/rh/rh-nginx18/nginx/nginx.conf }

# Vars to configure the service
service_config_var_listen_port: 80
service_config_var_document_root: /opt/rh/rh-nginx18/root/usr/share/nginx/html

# Status on boot
service_onboot: on

# Service status after the deploy
service_status: start
...
```

Cada uno de los apartados/variables que aparecen tienen un significado; y aunque aparece una descripción al respecto, a continuación se explica por separado cada uno de ellos:

- Configuración de repositorios. Es necesario poder acceder al software que necesita nuestro servicio; por ello, lo primero que definimos es el repositorio donde se encuentra dicho software.

**service\_software\_repo\_name.** Indica el nombre que le vamos a dar a este repositorio.

**service\_software\_repo\_url.** Indica la URL donde vamos a poder acceder al software.

- Nombres de los servicios. Hay veces que el servicio a nivel de sistema operativo es el mismo que queremos darle en nuestro “login”, pero dado que se puede dar la otra casuística, se han incluido dos variables para ello.

**service\_name.** Variable que se utiliza para mostrar el nombre del servicio, no tiene mayor implicación de los mensajes por pantalla.

**service\_os\_name.** Contiene el nombre del servicio a nivel de sistema operativo. Lo utilizamos para la configuración del arranque/parada del mismo.

- Software necesario para instalar el servicio. Especifica la lista de software necesario para instalar el servicio. Debemos intentar que el empaquetado de este software sea lo más *estandar* posible: .rpm, .dev, .msi... aunque evidentemente también podría ser un paquete .tar, .tgz, .zip... el laboratorio está pensado para .rpm, pero seguro que despues de hacerlo, se te ocurre como hacer los despliegues de los paquetes .tar, .zip...

**service\_pkgs.** Lista con los paquetes necesarios para instalar el servicio. Se pueden explicitar todos, o dejar al gestor de software que resuelva por si solo las dependencias.

- Ficheros y variables de configuración. En este apartado, vamos a definir las variables y ficheros de configuración del que dispone el servicio. Nuestra aproximación es la de ficheros de configuración con sintaxis *Jinja2*.

**service\_config\_main\_file.** Fichero que indica el path del fichero de configuración principal del servicio.

**service\_config\_files.** Lista, donde configuramos el fichero plantilla, y el path del fichero de configuración en la(s) máquina(s) destino.

**service\_config\_var\_\***. Las variables con este nombre las utilizamos para especificar los campos a sustituir en nuestros ficheros de configuración con sintaxis *Jinja2*. Pueden utilizarse todas las necesarias.

- Estado del servicio. Indicamos el estado del servicio una vez se ha realizado el deployment y en el arranque de la máquina.

**service\_status.** Puede tomar los valores de *start*, *stop*, *restart* o *reload*. Realiza la operación indicada una vez se ha realizado el deploy del servicio.

**service\_onboot.** Puede tomar los valores True or False. Según la operación indicada, así deja configurado el servicio en el arranque de la máquina.

Realizada la introducción y planteamiento del laboratorio, lo primero que vamos a realizar es la conexión a Ansible Tower, ya que es la herramienta *foco* del laboratorio.

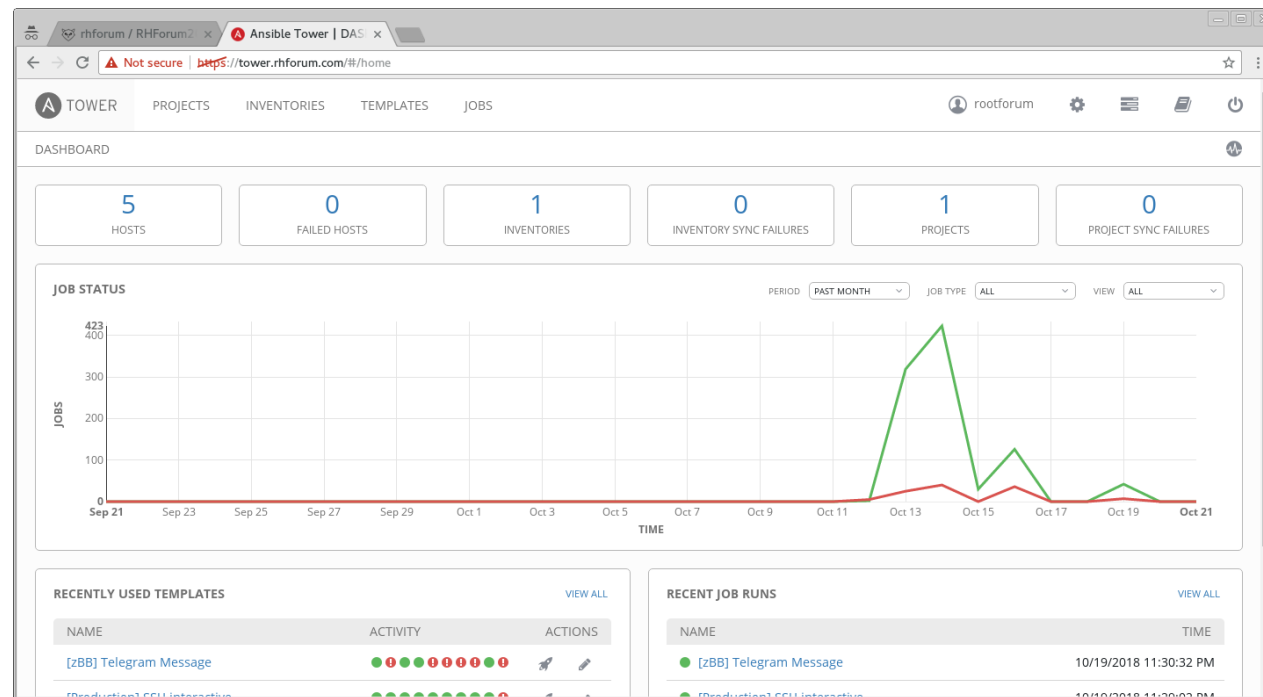
Para ello, y en un navegador, tecleamos la siguiente url:

```
http://tower.rhforum.com
```

Ahora utilizaremos las credenciales que se han comentado al comienzo del documento para poder hacer login con nuestro usuario del laboratorio.

```
User: rootforum  
Pass: rhforum123.
```

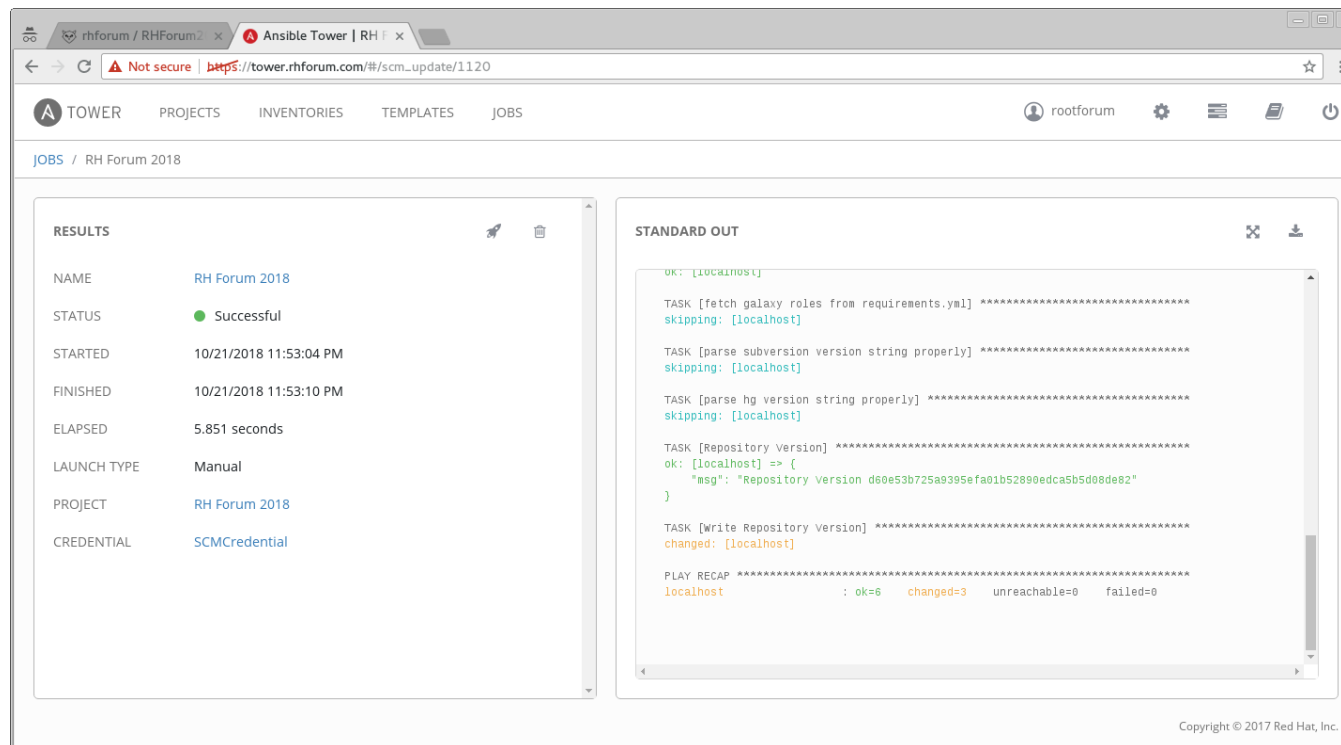
y obtenemos una pantalla similar a la siguiente:



Podemos dar un vistazo rápido por la herramienta, verificando el inventario creado, las credenciales que vamos a utilizar, la definición del proyecto... se recomienda encarecidamente, realizar este recorrido :)

La primera acción que vamos a ejecutar es una sincronización del proyecto, de esta manera verificamos la correcta conexión con nuestro repositorio de código, y además nos cercioramos que la última versión del código la tenemos en Ansible Tower.

Para ello, hacemos click en **PROJECTS**, y cuando nos aparece nuestro proyecto: *RH Forum 2018*, a la altura del nombre, a la derecha, hacemos click en el icono que representa una *nube*. En ese momento, al lado del nombre del proyecto, comenzará una *bola verde* a parpadear, indicando que se está realizando dicha operación de sincronización. Una vez terminado, la *bola* dejará de parpadear, y si su color es verde, indicará que la sincronización se ha realizado correctamente. Haya terminado bien o no, si hacemos click en la misma, podemos ver las operaciones que conllevan esta sincronización.



The screenshot displays the Ansible Tower web interface. The top navigation bar includes 'TOWER', 'PROJECTS', 'INVENTORIES', 'TEMPLATES', and 'JOBS'. The user 'rootforum' is logged in. The main content area shows the 'JOBS / RH Forum 2018' page. On the left, the 'RESULTS' section lists job details: NAME (RH Forum 2018), STATUS (Successful), STARTED (10/21/2018 11:53:04 PM), FINISHED (10/21/2018 11:53:10 PM), ELAPSED (5.851 seconds), LAUNCH TYPE (Manual), PROJECT (RH Forum 2018), and CREDENTIAL (SCMCredential). On the right, the 'STANDARD OUT' section shows the execution log, including tasks like 'fetch galaxy roles from requirements.yml', 'parse subversion version string properly', 'hg version string properly', 'Repository Version', and 'Write Repository Version'. The log concludes with a 'PLAY RECAP' showing 6 OK, 3 changed, 0 unreachable, and 0 failed on the localhost.

```
OK: [localhost]

TASK [fetch galaxy roles from requirements.yml] *****
skipping: [localhost]

TASK [parse subversion version string properly] *****
skipping: [localhost]

TASK [parse hg version string properly] *****
skipping: [localhost]

TASK [Repository Version] *****
ok: [localhost] => {
  "msg": "Repository Version d60e53b725a9395efa01b52890edca5b5d08de82"
}

TASK [Write Repository Version] *****
changed: [localhost]

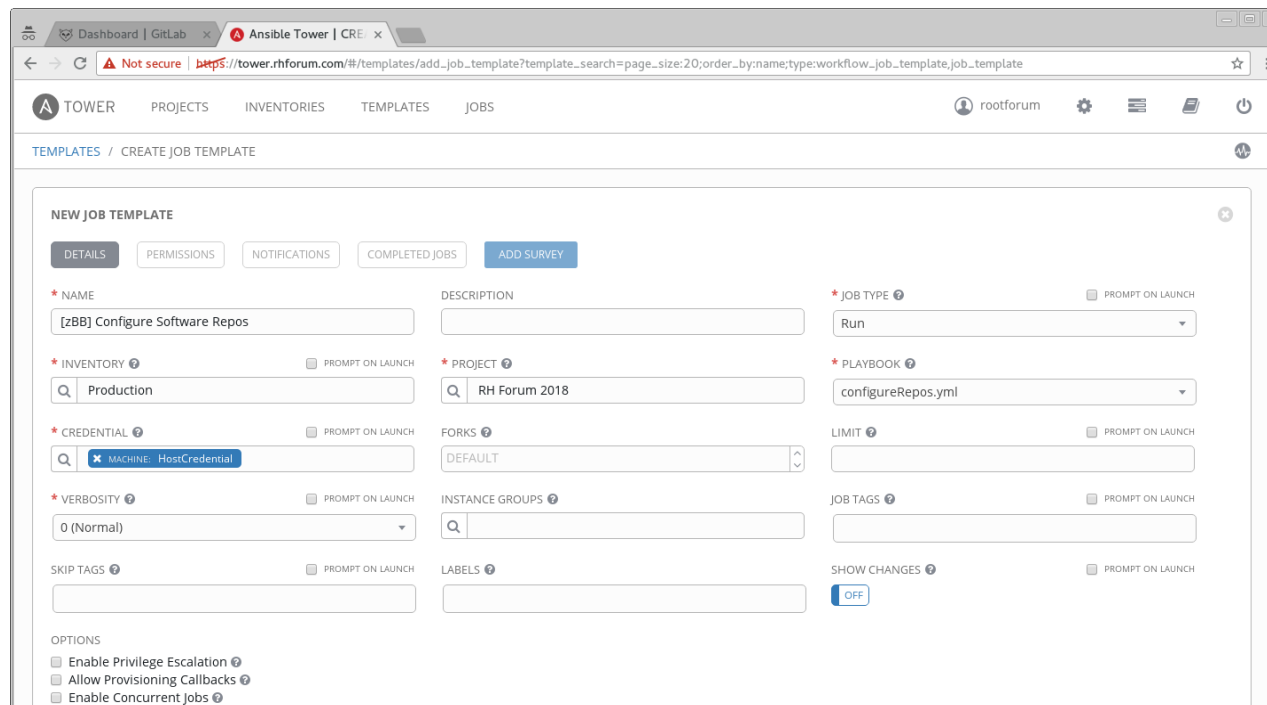
PLAY RECAP *****
localhost          : ok=6   changed=3   unreachable=0   failed=0
```

Copyright © 2017 Red Hat, Inc.

Como hemos comentado, la gran mayoría del trabajo de creación de los *Jobs Templates* está hecho, para poder ser más ágiles en el laboratorio. No obstante, vamos a realizar la creación de un *Job Template* para que el lector verifique cual es la manera de hacerlo. Este Job Template será el utilizado para realizar la configuración de los repositorios de software.

Para ello, iremos al apartado **TEMPLATES**, y allí hacemos click en el botón de la derecha **+ADD**, y elegimos la opción de **Job Template**. Nos aparece una pantalla, donde vamos a utilizar los siguiente datos para rellenar los campos necesarios y suficientes a la hora de crear una Job Template.

```
NAME: [zBB] Configure Software Repos
JOB TYPE: Run
INVENTORY: Production
PROJECT: RH Forum 2018
PLAYBOOK: configureRepos.yml
CREDENTIAL: HostCredential
```





Terminado de meter los datos, hacemos click en el botón **Save**, y ya tendríamos nuestro primer Job Templates hecho :)

Pero la idea, es realizar un despliegue orquestado del servicio, por lo que tenemos que realizar un *Workflow*, donde utilizaremos diferentes Jobs templates.

La siguiente tabla muestra los pasos definidos para realizar el despliegue del servicio, y su mapeo con los Jobs Templates que ya tenemos definidos en Ansible Tower:

Configuración de los repositorios de software	[zBB] Configure Software Repos
Instalación del software del servicio	[zBB] Service Install
Configuración del servicio	[zBB] Service Configure
Estado del servicio después del deploy	[zBB] Service Status
Configuración del servicio en el arranque de la máquina	[zBB] Service Status On Boot

Así, el siguiente paso es la creación del Workflow, por lo que desde la misma pantalla donde hemos creado el jobs template anterior, hacemos click en el botón **+ADD**, y ahora en vez de Job Template, escogemos Workflow Template. Una vez escogido, utilizamos los siguientes datos para crearlo:

```
NAME: [Production] Service Deploy  
ORGANIZATION: RHForum2018
```

Una vez introducidos los datos, pulsamos el botón **Save**, y vemos como los botones de **Survey** y **Workflow Editor** se activan; hacemos click en este último, y nos aparece el editor de Workflows, donde vamos a crear nuestra orquestación.

Iremos creando cada uno de los pasos del workflow - representados por una caja - y le asignaremos los jobs templates especificados en la tabla anterior. En este caso, solo vamos a ejecutar el paso  $n+1$  cuando el  $n$  haya sido correcto, como pista para crear el workflow ;)

La siguiente figura muestra el workflow creado.



Nótese la línea azul del **START** al primero de los pasos, esta línea significa que lo que hay detrás de ella se ejecutará **siempre**, mientras que las líneas verdes significan que el siguiente paso se ejecutará **si el anterior ha terminado correctamente**, y las rojas - aunque en este workflow no aparezcan - significan que el siguiente paso se ejecutará **si el anterior ha terminado de manera incorrecta**.

Por último, para terminar nuestro Workflow, vamos a dotarlo de un Survey. Un Survey es una *data entry* que aparece al comienzo de la ejecución del Workflow, y que nos sirve para que el usuario interaccione con el Workflow, escogiendo las diferentes opciones que se piden/muestran en el mismo.

Para ello, hacemos click en el botón **Add Survey**, y creamos nuestro survey con los siguientes datos:

```
PROMPT: TARGET
DESCRIPTION: Type host or hosts groups where deploy the service
ANSWER VARIABLE NAME: target
ANSWER TYPE: Text
REQUIRED: Check
```

```
PROMPT: SERVICE
DESCRIPTION: Choose the service to deploy
ANSWER VARIABLE NAME: survey_service
ANSWER TYPE: Multiple Choice (single select)
MULTIPLE CHOICE OPTIONS:
    apache
    nginx
    haproxy
REQUIRED: Check
```

Terminado de crear el Survey, hacemos click en **Save**, y ya tendríamos nuestro survey creado. La siguiente figura muestra cómo quedaría el mismo.

[Production] Service Deploy | SURVEY ON

#### ADD SURVEY PROMPT

\* PROMPT

DESCRIPTION

\* ANSWER VARIABLE NAME 

\* ANSWER TYPE 

Choose an answer type 

☒ REQUIRED

CLEAR

+ ADD

#### PREVIEW

\* TARGET


Type host or hosts groups where deploy the service



\* SERVICE

Choose the service to deploy







DELETE SURVEY

CANCEL

SAVE

Una vez hemos salvado el Survey, procedemos a salvar el Workflow, haciendo click en el boton **Save**.

Llegados a este punto, podemos proceder a realizar el deploy de nuestros servicios. Para ello, localizamos nuestro Workflow, y hacemos click en el icono del cohete, para lanzarlo.

Lanzaremos el Workflow tres veces, cada uno con los siguientes datos:

```
# Primera ejecución.  
TARGET: web1.rhforum.com  
SERVICE: apache
```

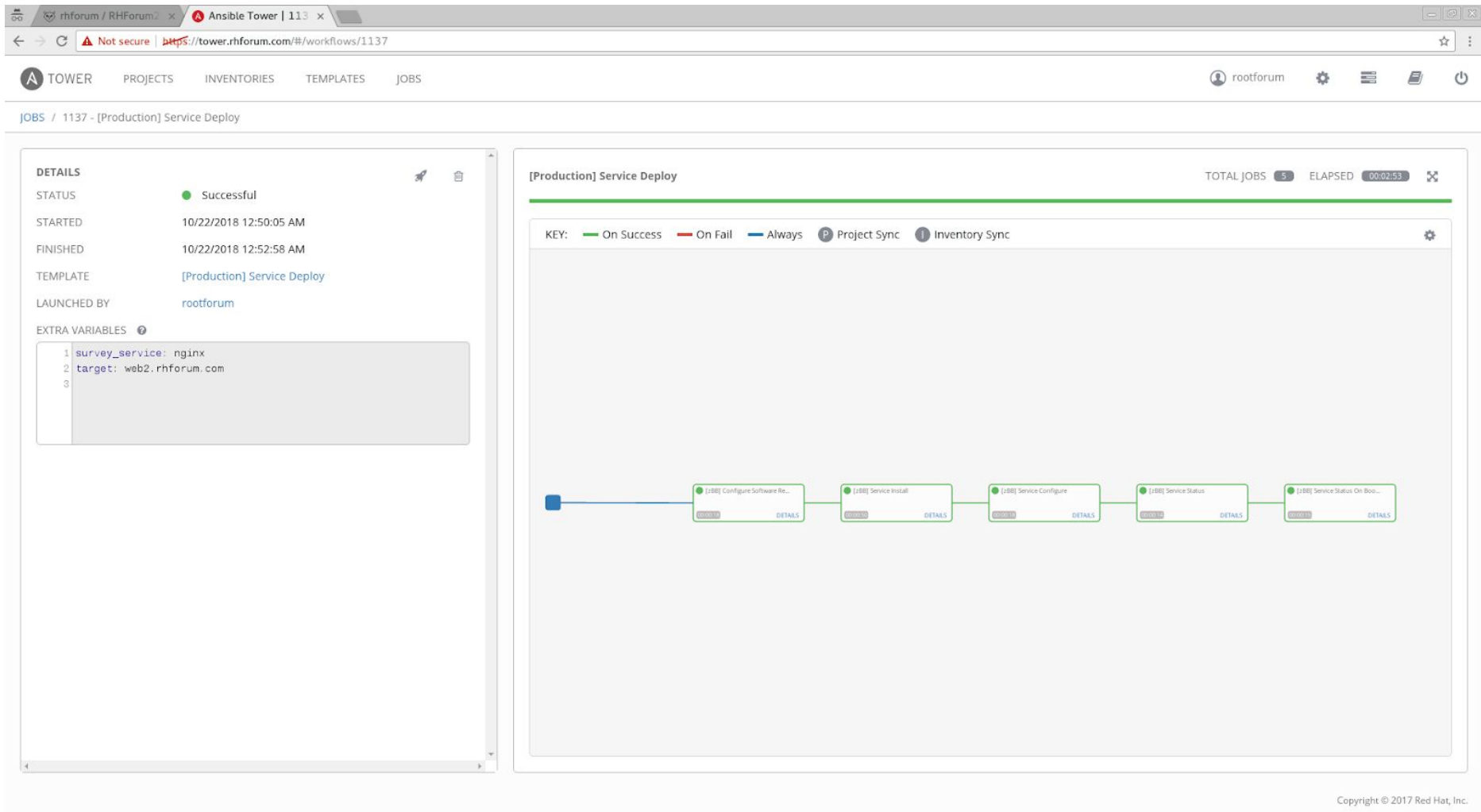
```
# Segunda ejecución.  
TARGET: web2.rhforum.com  
SERVICE: nginx
```

```
# Tercera ejecución.  
TARGET: lb.rhforum.com  
SERVICE: haproxy
```

Podemos verificar que operaciones se están ejecutando en un momento dado dentro del Workflow, ya que aparece una *bolita* parpadeando en el paso actual. Si termina, y su color es verde, es que ese paso a terminado correctamente; si es rojo, es que ha terminado incorrectamente.

Durante la ejecución, podemos hacer click en **DETAILS**, dentro de cada uno de los pasos, para verificar las tareas concretas de esa ejecución.

La figura de la página siguiente muestra el Workflow *terminado* para la segunda ejecución, donde en la máquina web2.rhforum.com, instalamos un servidor web nginx.



The screenshot displays the Ansible Tower web interface. The top navigation bar includes 'TOWER', 'PROJECTS', 'INVENTORIES', 'TEMPLATES', and 'JOBS'. The user 'rootforum' is logged in. The main content area shows the details of a job run for '[Production] Service Deploy'.

**DETAILS**

- STATUS: Successful
- STARTED: 10/22/2018 12:50:05 AM
- FINISHED: 10/22/2018 12:52:58 AM
- TEMPLATE: [Production] Service Deploy
- LAUNCHED BY: rootforum
- EXTRA VARIABLES:

```
1 survey_service: nginx
2 target: web2.rhforum.com
3
```

**[Production] Service Deploy**

TOTAL JOBS: 5 ELAPSED: 00:02:53

KEY: On Success (green), On Fail (red), Always (blue), Project Sync (P), Inventory Sync (I)

The job flow consists of five steps, all marked as successful (green):

- Configure Software Re...
- Service Install
- Service Configure
- Service Status
- Service Status On Boo...

Copyright © 2017 Red Hat, Inc.

Una vez terminados los deploys de los tres servicios, debemos proceder a la verificación de los mismos; para ello, desde un navegador, abrimos tres pestañas, y tecleamos las siguientes urls, cada una en una de las pestañas:

```
http://web1.rhforum.com
```

Se retorna la página por defecto de Apache.

```
http://web2.rhforum.com
```

Se retorna la página por defecto de Nginx.

```
http://lb.rhforum.com
```

Se retorna la página de Apache y Nginx, según se recargue la página, ya que el HAproxy está configurado con una política de *Round Robin*.

Si el lector desea inspeccionar la ejecución completa de los Workflows, podemos acudir al menú **JOBS** dentro de Ansible Tower, y verificar la ejecución de cada uno de los Jobs Templates.

## LAB #2. MODIFICACIÓN DE CONFIGURACIÓN Y SUBIDA DE CONTENIDO. TAREAS DE DÍA 2.

En muchas ocasiones, las herramientas de automatización y orquestación se suelen identificar para realizar tareas de despliegue, y nada más lejos de la realidad, ya que muchas de las operaciones que se realizan sobre los entornos, son *tareas de día 2* como:

- Parcheado.
- Modificación de configuraciones.
- Verificación de compliance...

Estas *tareas de día 2*, se contradicen con la evolución del *IaC*, que es la *Immutable Infrastructure*, donde su paradigma dice que no se puede realizar ninguna operación de modificación en la infraestructura en ejecución, y por ende, hay que realizar un nuevo deployment si es requerida realizar alguna modificación... le suena de algo al lector? Cúal es la tecnología que *de base* cumple con ello?

Pero mientras se realiza esta evolución de nuestros entornos, podemos y **debemos** utilizar herramientas como Ansible y Ansible Tower para estandarizar cualquier operación que se realice sobre la plataforma.

Así, como parte del laboratorio, en el caso de *tareas de día 2*, hemos incluido dos tareas:

- Modificación de la configuración del balanceador. Adición o sustracción de servidores al pool de balanceo.
- Subida de contenido a los servidores webs.

Para la primera de las tareas, se ha creado un Workflow para dicho menester, donde la principal característica del mismo es que las modificaciones en la configuración, no se realizan directamente sobre el balanceador, sino que se realizan sobre el fichero de definición del servicio del haproxy, se ejecuta un *commit* al repositorio, y posteriormente se aplica la configuración.

Para ello, desde el apartado de **TEMPLATES** de Ansible Tower, localizar el Workflow con nombre: **[Production] PROXY Config Modification**, y ejecutarlo como se ha indicado anteriormente, haciendo click en el icono del cohete.

Nada más lanzar el Workflow, nos aparece un Survey, donde utilizaremos la siguiente información para su ejecución:

```
TARGET SERVICE: haproxy
TARGET HOST: web1.rhforum.com
```

```
HTTP SERVICE: apache  
OPERATION: DELETE
```

En este caso, para verificar que el host web1.rhforum.com se ha eliminado de la configuración del HAproxy, podemos hacerlo de varias maneras:

- Recargando la conexión desde el browser a la url: **lb.rhforum.com**.
- Verificando el fichero **config/haproxy\_config.yml** desde una conexión a GitLab.
- Verificando el contenido del fichero de configuración del HAproxy **/etc/haproxy/haproxy.conf**.

Se recomienda al lector que realice más de una ejecución de este Workflow, con las siguientes entradas de datos.

```
# Entrada de datos  
TARGET SERVICE: haproxy  
TARGET HOST: web1.rhforum.com  
HTTP SERVICE: apache  
OPERATION: ADD  
  
# Resultado esperado  
El servidor web1.rhforum.com aparece de nuevo en la configuración del HAproxy.
```

```
# Entrada de datos  
TARGET SERVICE: haproxy  
TARGET HOST: web2.rhforum.com  
HTTP SERVICE: apache  
OPERATION: DELETE  
  
# Resultado esperado  
El workflow termina mandando un mensaje de error diciendo que el web2.rhforum.com no tiene un servidor Apache instalado.
```

Seguro que el lector se ha percatado que este Workflow puede mejorarse, tomando la premisa que, al menos siempre debe haber un host dentro del pool de balanceo, ya que si primero se elimina uno de los hosts, y luego el otro, en la segunda ejecución, el Workflow falla... se le invita al lector en verificar una posible solución al respecto.



La segunda operación *de día 2* que vamos a realizar, es la actualización del contenido estático de nuestro servidor web.

A priori, esta tarea, básicamente es sencilla: copiar el contenido que estará en un directorio origen, a la máquina(s) que le especifiquemos. El asunto aquí es que debemos tener en cuenta que tenemos diferentes tipos de servidores webs, y por tanto, se nos puede dar la casuística - entre otras - que la ruta donde se debe poner el contenido, no sea la misma para cada uno de ellos.

Tanto es así, que si inspeccionamos los ficheros de configuración de los servicios apache y nginx, podemos verificar:

```
[rhforum@dev RHForum2018]$ grep -i document_root config/apache_config.yml
service_config_var_document_root: /var/www/html
[rhforum@dev RHForum2018]$ grep -i document_root config/nginx_config.yml
service_config_var_document_root: /opt/rh/rh-nginx18/root/usr/share/nginx/html
```

Por tanto se nos abren dos posibilidades:

- Pedir al usuario que nos introduzca el tipo de servicio donde vamos a realizar la actualización del contenido.
- Que la propia automatización sea capaz de discernir sobre qué servicio está actuando, y actúe en consecuencia.

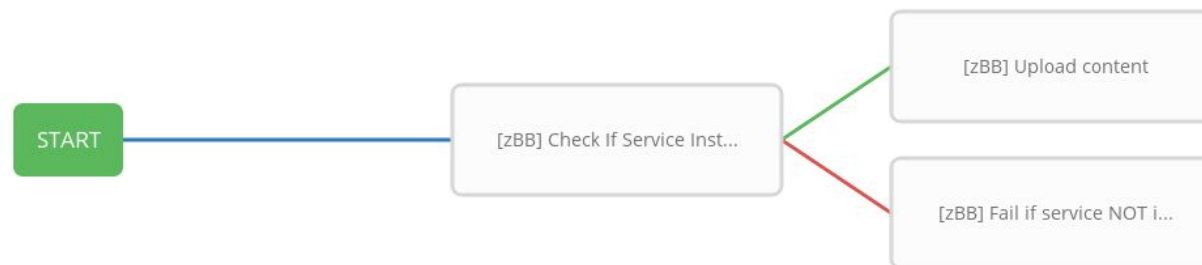
Es evidente que la mejor de las opciones es la segunda, pero en el laboratorio, se ha optado por la primera de ellas, dejando la segunda como ejercicio para el lector.

Fijándonos en la primera de ellas, tenemos que verificar si la máquina destino tiene el servicio instalado que nos introducen, y dado que en el Workflow de modificación de la configuración del proxy ya realizamos esta operativa, vamos a reutilizar los Jobs Templates que realizamos, y crearemos un Workflow para realizar la gestión del contenido estático.

Así, desde el apartado de **TEMPLATES** de Ansible Tower, vamos a realizar un Workflow con los siguientes datos:

```
NAME: [Production] Upload Web Content
ORGANIZATION: RHForum2018
```

Una vez terminado de crear el Workflow, quedará de la siguiente manera:



Por último, y dado que necesitamos pedir datos al usuario, vamos a crear un Survey con los siguientes datos:

```
PROMPT: TARGET
DESCRIPTION: Type the host to upload the content
ANSWER VARIABLE NAME: target_http
ANSWER TYPE: Text
REQUIRED: Check
```

```
PROMPT: HTTP SERVICE
DESCRIPTION: The host is Apache or Nginx?
ANSWER VARIABLE NAME: survey_service_http
ANSWER TYPE: Multiple Choice (single select)
MULTIPLE CHOICE OPTIONS:
    apache
    nginx
REQUIRED: Check
```

En la página siguiente se muestra el Survey una vez terminado. Salvamos todos nuestros cambios en el Workflow que terminamos de crear y lo ejecutamos con los siguientes datos:

```
TARGET: web1.rhforum.com
HTTP SERVICE: apache
```

[Production] Upload Web Content | SURVEY ON

#### ADD SURVEY PROMPT

\* PROMPT

DESCRIPTION

\* ANSWER VARIABLE NAME ?

\* ANSWER TYPE ?

Choose an answer type ▼

☒ REQUIRED



CLEAR

+ ADD

#### PREVIEW



\* TARGET

Type the host to upload the content

⋮   

\* HTTP SERVICE

The host is Apache or Nginx?

⋮  ▼  

DELETE SURVEY

CANCEL

SAVE

Para verificar que el contenido ha cambiado, y que actualmente ya no se muestra la página por defecto de Apache, se le deja al lector que pruebe a conectarse al servidor web en concreto o al balanceador.

Por último, podemos hacer otro tipo de control - siguiendo el mismo ejemplo - que sería utilizando código. Para verificarlo, habría que crear un Job Template con los siguientes datos:

```
NAME: [Production] Upload Web Content - Playbook
JOB TYPE: Run
INVENTORY: Production
PROJECT: RH Forum 2018
PLAYBOOK: uploadWebContent_witchCheck.yml
CREDENTIAL: HostCredential
```

Una vez creado el Job Template, hay que añadirle el mismo Survey que hemos añadido en este apartado al Workflow anterior, al de subir el contenido a los web servers.

## LAB #3. BONUS TRACK: MODIFICACIÓN ACCESO SSH MÁQUINAS DE PRODUCCIÓN

Por último, y no menos importante, se ha preparado este último lab, donde vamos a poder activar o desactivar el *acceso interactivo SSH* por consola a las máquinas de producción.

El acceso interactivo vía SSH a las máquinas de producción está desactivado. Vamos a probarlo:

```
[rhforum@dev RHForum2018]$ ssh rhforum@web1.rhforum.com
rhforum@web1.rhforum.com's password:
PTY allocation request failed on channel 0
Connection to web1.rhforum.com closed.
[rhforum@dev RHForum2018]$ ssh rhforum@web2.rhforum.com
rhforum@web2.rhforum.com's password:
PTY allocation request failed on channel 0
Connection to web2.rhforum.com closed.
[rhforum@dev RHForum2018]$ ssh rhforum@lb.rhforum.com
rhforum@lb.rhforum.com's password:
PTY allocation request failed on channel 0
Connection to lb.rhforum.com closed.
```

Para activar o desactivar el SSH interactivo, se ha preparado el WorkFlow: **[Production] SSH interactive**.

Prueba a ejecutarlo, rellenando los datos que pide el Survey, y veras que si lo pones a ON, podrás posteriormente acceder vía SSH, eso si, con el usuario rhforum, ya que root, está desactivado :)

```
[rhforum@dev RHForum2018]$ ssh rhforum@web1.rhforum.com
rhforum@web1.rhforum.com's password:
Last login: Fri Oct 19 23:00:02 2018 from 192.168.110.162
[rhforum@web1 ~]$ hostname
web1.rhforum.com
[rhforum@web1 ~]$ logout
Connection to web1.rhforum.com closed.
[rhforum@dev RHForum2018]$ ssh root@web1.rhforum.com
root@web1.rhforum.com's password:
Permission denied, please try again.
root@web1.rhforum.com's password:
```

Pero dado que esta operación, es una operación crítica, podemos modificar el Workflow, y añadir el envío de un mensaje a Telegram.

Para ello, vamos a modificar el Workflow indicado, y añadimos un paso más al Workflow, quedando de la siguiente manera:



Además, tenemos que añadir al Survey que tiene creado, un campo más, con los siguientes datos:

```
PROMPT: MESSAGE  
ANSWER VARIABLE NAME: survey_message  
ANSWER TYPE: Text  
DEFAULT ANSWER: RH Forum 2018  
REQUIRED: Check
```

Una vez salvado el Survey, prueba de nuevo a ejecutarlo, y en el mensaje, utiliza el que está por defecto, o pon tú uno para saber que tu mensaje llega correctamente... aunque aquí dependemos de la conexión a internet :)