TAREA 2

Autor: Diego Marco Beisty, 755232
Fecha: 14/04/2020
Asignatura: Administración de Sistemas 2

FUNCIONALIDAD

Tras la realización de la parte 2 del trabajo, el script u soporta las siguientes opciones:

```
Uso: u [grupo o maquina] p | s "comando" | c manifiesto
p                   ping al puerto 22
s "command"         ejecución comando remoto
c manifiesto        ejecución manifiesto remoto
```

ASPECTOS DE DISEÑO

Lectura fichero .u/hosts

Se va a añadir la posibilidad de definir grupos de máquinas tanto por su ip como por su dominio. Para ello delante del nombre del grupo tiene que estar el carácter “-”.
Para que el script incluya el conjunto de máquinas que se especifican debajo del nombre de grupo hay que añadir el mismo nombre de grupo precedido por el carácter ‘+’ .
Se ha planteado el siguiente diseño para implementar esta característica.

1) Detección de argumentos:
   Se va a detectar si el usuario quiere incluir un grupo específico o todos los grupos incluidos en el script. Para ello se va a comprobar el primer argumento del script.
   Si es un comando del script (p, s o c) implica que quiere añadir todos los grupos incluidos en hosts y por lo tanto se invoca a la rutina de tratamiento del fichero sin indicar un grupo específico. En caso de que el primer argumento sea una dirección IPv4 o IPv6 no invoca a la rutina de tratamiento del fichero.
   Finalmente si detecta que es un nombre de grupo, se lo indica a la función que lee el fichero hosts.

2) Función obtener:
   La función que trata el fichero, si se le ha especificado un grupo concreto desde los argumentos del programa, va a buscar que ese grupo esté incluido en el fichero precedido por un signo ‘+’. Si no lo encuentra devolverá un mensaje de error y abortará la ejecución, en caso contrario almacenará el grupo.
   Si no se especifica ningún grupo el comportamiento será guardar todos los grupos que se incluyan con un signo ‘+’. Si no se ha guardado ninguno, es decir no se había incluido ninguno en el fichero, se abortará la ejecución.
   Una vez almacenados todos los grupos incluidos en el fichero, se buscarán todos los grupos definidos con un signo ‘-’ y si coinciden con alguno de los grupos incluidos, se guardarán todos los hosts que contenga. Finalmente se eliminarán todos aquellos hosts que estén repetidos.

Comando c

Permite la ejecución remota de manifiestos Puppet. Para ello primero comprueba que para cada fichero <manifiesto> especificado en los argumentos de programa existe la ruta ~/.u/manifiestos/<manifiesto>. Si existe, para cada host destino recopilado en la lectura del

fichero hosts, establece una conexión ssh. Primero guardará el manifiesto en la ruta /tmp/.puppet del host destino, después ejecutará el manifiesto con el comando puppet:
sudo puppet apply o doas puppet apply según si el sistema operativo destino es ubuntu u openBSD respectivamente.
Finalmente borrará el directorio temporal y devolverá la salida obtenida del comando.

Manifiesto dns_cliente.pp

Este manifiesto se va a asegurar que los clientes del servicio dns, concretamente del servicio unbound corriendo en la máquina 2001:470:736b:1ff::2, tienen el fichero de configuración resolv.conf configurado correctamente para poder resolver nombres de dominio.
Puesto que la dirección y nombre de este fichero varía según el sistema operativo, primero se mira la variable $operatingsystem y según sea openbsd o ubuntu, se le asigna a otra variable el fichero de configuración correcto.
Después se añade una declaración del fichero que debe existir en la máquina con la ruta adaptada anteriormente a cada sistema operativo, con permisos 0644 y cuyo contenido es:
"nameserver 2001:470:736b:1ff::2", indicando el servidor recursivo que va a atender la petición dns.

Manifiesto ntp_cliente.pp

Este manifiesto se asegura que los clientes tengan el servicio ntp activo y el fichero ntp.conf correctamente configurado.
Para ello primero se va a identificar cual es el sistema operativo del host y en función del mismo se guardará el nombre del servicio y la dirección del fichero de configuración específica.
El nombre del servicio para openbsd es ntpd y el fichero de configuración es /etc/ntpd.conf.
El nombre del servicio para ubuntu es ntp y el fichero de configuración es /etc/ntp.conf.

VERIFICACIÓN

Se ha comprobado la lectura de los grupos del fichero hosts mediante instrucciones puts que permitían imprimir por stdout los hosts incluidos por el script.
Se ha verificado el comando c mediante la ejecución de un manifiesto llamado prueba.pp que creaba un fichero en /home/a755232 del grupo clientes_ntp con el contenido “Esto es una prueba”.
Para la verificación de los manifiestos dns_cliente.pp y ntp_cliente.pp se han vaciado los ficheros ntp.conf y tail en la máquina u1ff5 y los ficheros ntpd.conf y resolv.conf en la máquina o1ff3 y después se ha ejecutado el programa u con la opción c.
Se ha obtenido el resultado esperado.

PROBLEMAS ENCONTRADOS

He tenido problemas de red IPv6 al ejecutar el script desde mi portátil pese a tener instalado el paquete miredo. Este problema se solucionó dejando pasar un día de tiempo, probablemente fueron problemas de sobrecarga de la red.
También he tenido problemas de sintaxis con los manifiestos puppet que he solucionado comprobando el código en la página https://validate.puppet.com/

ANEXO: CÓDIGO FUENTE

Fichero dns_cliente.pp

```
case $operatingsystem {
      openbsd: { $file_name = '/etc/resolv.conf' }
      ubuntu:  { $file_name = '/etc/resolvconf/resolv.conf.d/tail' }
}

file { 'resolv.conf':
      ensure => file,
      path => $file_name,
      mode => '0644',
      content => "nameserver 2001:470:736b:1ff::2",
}
```

Fichero ntp_cliente.pp

```
case $operatingsystem {
      openbsd: { $service_name = 'ntpd'
                 $file_name = '/etc/ntpd.conf' }
      ubuntu:  { $service_name = 'ntp'
                 $file_name = '/etc/ntp.conf' }
}

service { 'ntp':
      ensure => running,
      name => $service_name,
      enable => true,
      subscribe => File['ntp.conf'],
}

file { 'ntp.conf':
      ensure => file,
      path => $file_name,
      mode => '0644',
      content => "#Sincronizacion NTP con:
server ntp1.1.ff.es.eu.org
",
}
```

Fichero u.rb

```ruby
#!/usr/bin/ruby -w
################################################################################
# File:   u.rb
# Author: Diego Marco Beisty 755232
# Date:   15/03/2020
# Coms:   Herramienta de ejecución remota, despliegue y configuracion
#       automática
################################################################################
require 'rubygems'
require 'net/ping/tcp'
require 'net/ssh'
require 'resolv'
require 'net/scp'

# <maquinas> := {"<domain>"|"<ip>"}
# Ejecuta ping TCP al puerto 22 con timeout de 0.3s a las máquinas
especificadas
# en el array <maquinas>
def ping_tcp(maquinas)
    num_host = 1
    maquinas.each do |host|
       t = Net::Ping::TCP.new(host, 22, 0.3)
       if t.ping?
             puts "máquina_" + num_host.to_s + ": FUNCIONA"
       else
             puts "máquina_" + num_host.to_s + ": falla"
       end
       num_host = num_host + 1
    end
end

# Ejecuta comando remoto <remote_comand> mediante ssh a todas las
máquinas
# del array <maquinas>
def ssh_command(maquinas, remote_command)
    num_host = 1
    maquinas.each do |host|
      begin
       ssh = Net::SSH.start(host, "a755232",:timeout=> 7)
       res = ssh.exec!(remote_command)
       puts "máquina" + num_host.to_s + ": exito"
       puts res + "\n\n"
      rescue
```

```
       puts "máquina" + num_host.to_s + ": UNREACHABLE\n\n"
      end
      num_host = num_host + 1
    end
end


# Aplica un manifiesto puppet del directorio <f_manifiestos> en
<maquinas>
def manif_puppet(maquinas, d_manifiestos, lista)
       f = File.expand_path(d_manifiestos) # Expandir ~
    lista.each do |manifiesto|
            if File.file?(f + "/" +  manifiesto)
            begin
             maquinas.each do |host|
                  Net::SSH.start(host, "a755232",:timeout=> 7) do |ssh|
                        ssh.exec!("mkdir /tmp/.puppet")
                        scp = Net::SCP.start(host, "a755232")
                        scp.upload! f + "/" + manifiesto, "/tmp/.puppet"
                        os = ssh.exec!("uname")
                        if os.strip == "OpenBSD" #host tiene sistema
                       operativo openBSD
                              res = ssh.exec!("doas puppet apply
/tmp/.puppet/" + manifiesto)
                              else #host tiene sistema operativo ubuntu
                        res = ssh.exec!("sudo puppet apply /tmp/.puppet/"
+ manifiesto)
                        end
                        ssh.exec!("rm -rf /tmp/.puppet")
                        puts "máquina " + host + ": exito"
                        puts res
                  end
             end
                  rescue
             puts "máquina : UNREACHABLE\n\n"
                  end
       else
            abort "Fichero " + manifiesto + " no existe"
       end
    end
end

# Comprueba que el grupo especificado en <grupo> se incluye en f_config
# con formato +grupo. Por defecto identifica todos los grupos incluidos.
# Por cada grupo incluido añade en <maquinas> los hosts asociados a la
```

```
# definicion del grupo con formato -grupo
def parse(maquinas, f_config, grupo="")
    gruposInc=[] #Guardo los grupos con formato +grupo de f_config
    File.foreach(f_config) do |line|
       line = line.strip
       if line[0] == "+"
            line[0]=''
            #Por defecto añado el grupo. Si <grupo> no es nulo, solo
añado
            #si coincide con <grupo>
            (grupo == "") ? gruposInc << line : if line == grupo then
gruposInc << line end
       end
      end

    if grupo == ""
       abort "Ningun grupo incluido en " + f_config \
                 unless !gruposInc.empty?
      else
       abort "El grupo " + grupo + " no esta incluido en " + f_config \
                 unless !gruposInc.empty?
    end

    incluir = false
    File.foreach(f_config) do |line|
       line = line.strip
       if incluir and !line.empty? and line[0] != "-" and line[0] != "+"
                  maquinas << line
       end
       if line[0] == "-"
            line[0]=''
            (gruposInc.include? line) ? incluir = true : incluir = false
       elsif line[0] == "+"
            incluir = false
       end
    end
    maquinas = maquinas.uniq # Filtro los hosts que aparecen más de una
vez.
end

# Lee y guarda @ips y dominios perteneciente al grupo <grupo> en
<maquinas>
# Por defecto lee todos los grupos incluidos en <f_config>
def obtener(maquinas, f_config, grupo="")
       f = File.expand_path(f_config) # Expandir ~
```

```
       if File.file?(f)
      parse(maquinas, f, grupo)
      else
            abort "Fichero " + f_config + " no existe"
      end
end




options = ["p", "s", "c"]     #Comandos disponibles
first = ARGV[0]          #Primer argumento introducido
second = ARGV[1]         #Segundo argumento introducido (puede ser vacío)
maquinas=[]                    #Guarda ips y dominos host
f_config = "~/.u/hosts"        #Fichero máquinas hosts
d_manifiestos = "~/.u/manifiestos"    #Directorio manifiestos puppet


#Error checking
abort "Uso: u [grupo o maquina] [p] [s \"comando\"] [c manifiesto]\n" +
      "  p          ping al puerto 22\n" +
      "  s \"command\"      ejecucion comando remoto\n"+
      "  c \"manifiesto\" ejecucion manifiesto remoto"\
      unless options.include?(first) or options.include?(second)


#Guardar los hosts en <maquinas>
if options.include?(first)
    obtener(maquinas, f_config) #Caso: u p
elsif (first =~ Regexp.union([Resolv::IPv4::Regex, Resolv::IPv6::Regex]))
    maquinas << first #Caso: u ip p
else
    obtener(maquinas, f_config, first) #caso u grupo p
end

#Ejecución subcomando
if !(options.include?(first)) then ARGV.shift end
command = ARGV[0]
if command == "p"
      ping_tcp(maquinas)
elsif command == "s"
      ssh_command(maquinas, ARGV[1])
elsif command == "c"
    manif_puppet(maquinas, d_manifiestos, ARGV[1].split(","))
end
```