Universidad De Zaragoza

Trabajo De Fin De Grado

Despliegue de una federación cloud: instanciación, presentación de recursos, gestión de la pertenencia y monitorización y recuperación ante desastres

Deployment of a cloud federation: instantiation, resource submission, membership management and monitoring and disaster recovery.

Lucas Cauhé Viñao

Director: Eduardo Tomás Fiat Gracia

Ponente: Unai Arronategui Arribalzaga

Grado en Ingeniería Informática
Computación

2025

Resumen

Glosario

Bridge huérfano: Interfaz *bridge* sin interfaz física como *master*.

Trunk: En el contexto de interfaces de red que dejan pasar tráfico de VLAN, capacidad de la interfaz para transmitir más de una VLAN.

pool Ceph: Agrupación lógica de un conjunto de objetos RADOS sobre la que se aplican un conjunto de reglas de replicación y mantienen los datos distribuidos con el uso de *placement groups*.

Conceptos

HA: Alta Disponibilidad. Característica de un sistema que asegura un cierto rendimiento, normalmente de "Tiempo en línea", por un periodo de tiempo más extenso al normal.
Confianza: Interfaz *bridge* sin interfaz física como *master*.

Índice

1. Introducción

1.1. Contexto

El SICUZ es el departamento de Servicios de Informática y Comunicaciones de la Universidad de Zaragoza. Encargado de mantener la operativa informática y la infraestrucutra que le da soporte. En este contexto, la gestión de un cloud privado permite controlar la información y servicios ofrecidos desde la Universidad.

En este departamento se está implementando un proyecto de colaboración interuniversitario, donde infraestructura en distintas universidades forman un clúster que da soporte a una instancia cloud de *OpenNebula*. Esto supone homogeneizar la infraestructura (almacenamiento, red y cómputo) para que cada universidad ofrezca y reciba rendimientos similares.

A nivel de aplicación y gestión de usuarios, la confianza entre los miembros del clúster se establece de palabra y siguiendo una serie de buenas prácticas establecidas de forma general.

Así, se ha contemplado la idea de federar cada uno de los cloud privados de cada universidad, permitiendo una gestión abstracta y establecida mediante políticas sobre el uso y recursos presentados.

1.2. Motivación y alcance

TODO

El uso de una federación cloud permite tener una gestión local de la infraestructura y compartida de la información de usuarios. Así, el nivel de confianza entre las entidades federadas rige el control de acceso a los recursos presentados.

1.3. Estudio del arte

TODO

1.4. Objetivos

TODO

El objetivo principal del proyecto es establecer una federación entre 2 instancias de OpenNebula usando Ceph como sistema de almacenamiento distribuido. Se dará una solución de recuperación ante desastres para los recursos de la federación, siendo estos: sistemas de ficheros (de máquinas virtuales, de configuración y estado de la federación), estado de máquinas virtuales y catálogo de servicios de la federación (imágenes persistentes del Marketplace privado de OpenNebula). Por último, se especificarán las políticas de gobernanza básicas que afectaran al uso y los recursos de la federación y un sistema de monitorización y control de accesos que valide las políticas definidas.

Para ello, primero se desarrollará la infraestructura que dé soporte a las instancias cloud. El diseño de la infraestructura deberá considerar la tolerancia a fallos y alta disponibilidad. El despliegue de la infraestructura (red y almacenamiento local de los servidores implicados, 2 instancias de OpenNebula, 2 instancias de Ceph) y federación de las entidades será automático.

Las políticas definidas deberán cubrir el uso y control de acceso a los servicios de la federación por parte de los usuarios de las entidades federadas. Habrá un sistema de monitorización y otro de validación de las políticas, que dado un servicio y su uso deseado, dictará si se permite o no el uso del servicio.

La recuperación ante desastres deberá cumplir las políticas establecidas, ya que será un servicio más de la federación. El despliegue de los recursos cloud de OpenNebula empleados para la recuperación ante desastres será automático mediante alguna herramienta de automatización que integre OpenNebula. El tipo de datos almacenados en este proceso será de objetos ya que responden a la naturaleza del problema: múltiples lecturas, única escritura y recuperación del objeto del medio rápida. La informa-

ción almacenada deberá estar disponible para todas las entidades en cualquier momento ya que la recuperación de los datos debe ser independiente del estado del servicio de backup en cada entidad.

1.5. Metodología

TODO

1.6. Organización de la memoria

TODO
Freestyle

2. Diseño

2.1. Diseño de la federación

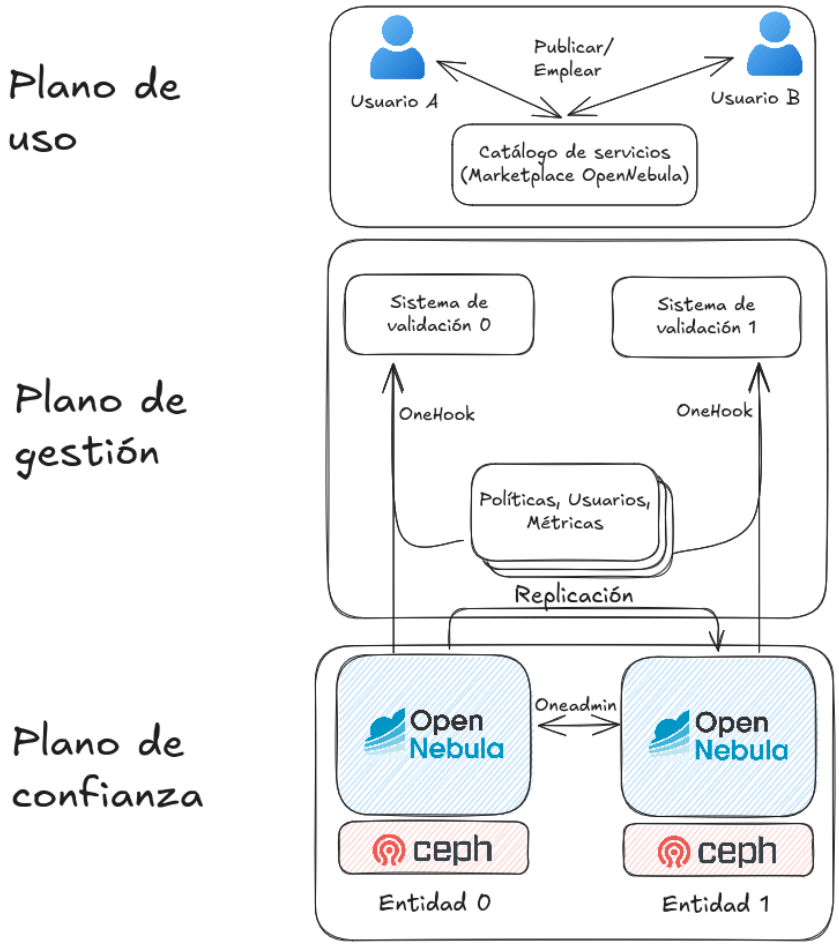

Figura 1: Arquitectura del sistema presentado de acuerdo con el modelo del NIST

La Figura 1 introduce los componentes principales principales del sistema. El modelo planteado por el NIST establece tres planos de abstracción: confianza, gestión y uso. La confianza entre entidades se establecerá mediante los protocolos internos de compartición de credenciales de administración de OpenNebula. En el plano de gestión se hará cumplir con las políticas de gobernanza definidas mediante un sistema de validación. El plano de uso describe la interacción que habrá entre los usuarios de las distintas entidades con los servicios ofrecidos por la federación.

2.1.1. Plano de confianza

El plano de confianza lo forman las instancias de OpenNebula, desplegadas en alta disponibilidad cada una de ellas. La confianza entre ambas entidades se consigue compartiendo las credenciales del usuario administrador *ondeadmin* de la entidad *maestra* a la *esclava*. En la Figura 1 la «Entidad 0» es la *maestra* y la «Entidad 1» la *esclava*.

Aunque Ceph no es estrictamente necesario, su uso es recomendado por ser un estándar en la industria.

2.1.2. Plano de gestión

La gestión de la federación estará a cargo de una serie de políticas de gobernanza que controlarán accesos de usuarios a servicios y el comportamiento y rendimiento de los servicios (backup, sistema de validación y máquinas virtuales de usuarios) en ejecución en la federación.

Habrá un sistema de validación en cada entidad federada dotando de alta disponibilidad a este servicio. Se plantean este servicio en un modo pasivo, es decir, que este sistema dictará si un recurso presentado cumple o con las políticas establecidas y dejará o no desplegarlo consecuentemente. En el caso de que se plantease activo, el sistema debería ser capaz de aplicar los cambios necesarios al recurso para que cumpla con las políticas. Se ha escogido el modelo pasivo por claridad y simpleza en el diseño y viabilidad técnica en tiempo y forma en su implementación.

Sistema de monitorización

2.1.3. Plano de uso

El plano de uso viene definido por dos componentes: usuarios y catálogo de la federación. Los usuarios...

2.2. Diseño de la Infraestructura

Infraestructura real a desplegar.

2.2.1. Red

2.2.2. Almacenamiento

2.3. Diseño de la Recuperación ante Desastres

El eje central de la recuperación ante desastres será el sistema de backup. Se hablará del backup de la federación más que backup de los componentes que la componen por separado. Así se van a tener una serie de políticas que afecten al backup, como a cualquier otro servicio, y la infraestructura que le dé soporte. Los recursos sobre los que se actuará serán sistemas de ficheros (de máquinas virtuales, configuraciones y estado de la federación y políticas) y estado de las máquinas virtuales de la federación. Al ser el despliegue de OpenNebula autocontenido, los componentes de este sistema se autogestionarán, haciendo que con la misma configuración se hagan backups de servicios de la federación y de los componentes de OpenNebula.

El servicio de backup y de validación de políticas no almacenan estado por lo que su recuperación ante desastres consistirá en almacenar la configuración de OpenNebula que permita desplegar automáticamente estos servicios cuando se detecte que falta alguno.

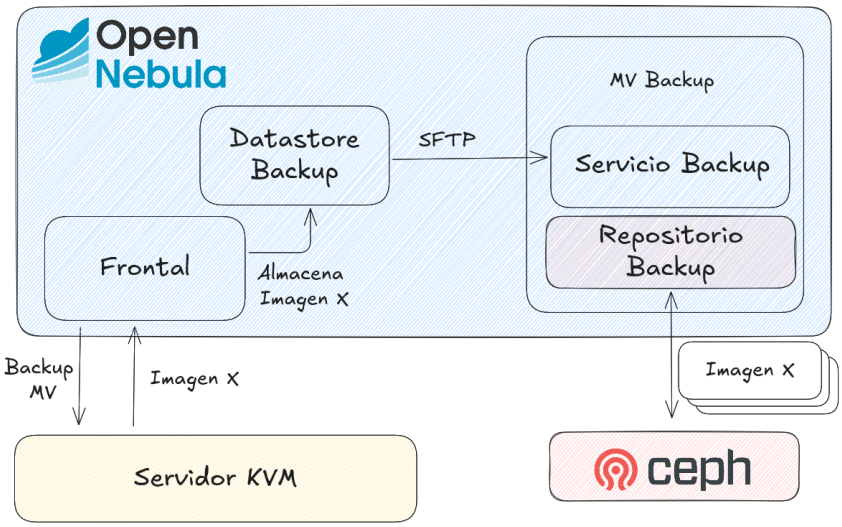

Figura 2: Arquitectura del servicio de backup para cada entidad

La Figura 2 destaca el uso de un repositorio backup que contacte con Ceph para el almacenamiento de objetos deseado y el protocolo de comunicación SFTP entre el frontal de OpenNebula y el servicio

de backup escogido. Este útlimo es una restricción de OpenNebula para conseguir backups nativos, otra opción hubiese sido desaprovechar la capacidad de definición de tareas de backup nativa de OpenNebula para emplear un protocolo de almacenamiento de objetos como Ceph S3. Con este diseño, se configurará el servicio de backup para que almacene localmente las imágenes de backup de OpenNebula y el sistema de ficheros empleado tendrá contacto con Ceph.

La infraestructura de red que dé soporte al backup en cada entidad es libre pero estará sujeta a los SLA acordados para backup. Para ello se propone una red basada en la tecnología fibre channel, con posibilidad de utilizar FCoE si se cuenta con mecanismos de DCB en la infraestructura interna de la entidad.

3. Implementación

3.1. Implementación de los servicios de la federación

3.1.1. Políticas implementadas

3.1.1.1. Política de nomenclatura

Todas los nombres tendrán como prefijo común el nombre de la entidad seguido por un guión. El Listado 1 muestra un ejemplo de una política de nombrado implementada usando el lenguaje *Rego*.

1 **Nombrado de etiquetas:** <nombre-etiqueta> (ej. one-10-prod)
2 **Nombrado de máquinas virtuales:** <nombre-usuario>-<nombre-maquina> (ej. one-10-user10--ntp)
3 **Nombrado de imágenes:** <nombre-usuario>-<nombre-imagen> (ej. one-10-user10-Ubuntu)
4 **Nombrado de tareas de backup:** <prioridad> (ej. one-10-90)
5 **Nombrado de *hooks*:** hook-<nombre-hook> (ej. one-10-hook-update-backupjob)

3.1.1.2. Políticas de servicio mínimo

1 **Sistema de monitorización:** Cada entidad contará con un sistema que monitorice la actividad local de OpenNebula.
   i **Punto de enlace**: Definir una IP virtual donde establecer la comunicación de la federación.
   ii **Exposición de métricas:** Las métricas se expondrán en formato *JSON* accesibles desde el punto de enlace definido en la ruta */metrics* y su esquema en */metrics/schema* que deberá coincidir con el esquema esperado por la federación.
   iii **Periodo de monitorización:** Se establecerá una frecuencia de monitorización de 30 segundos.
   iv **Control de acceso:** Habrá un grupo de usuarios con permisos, exclusivamente, de consulta de métricas llamado "fed-exporter".
2 ***Uptime* (tiempo en línea) de servicios:** 90% de tiempo en línea requerido para cada uno de los servicios etiquetados como producción.
3 **Definición de downtime o caída:** Un servicio se considera caído si no se exportan sus métricas durante un periodo entero y el estado de la máquina virtual en la que se ejecuta en OpenNebula es *ERROR*, *POWEROFF* o no existe.
4 **Sobreaprovisionamiento:**
5 **Sobrecarga:**
   i Una entidad no podrá desplegar más máquinas virtuales si la utilización de *vCPU* está por encima del 80%.
   ii Una entidad no podrá desplegar más máquinas virtuales cuyas imágenes sean persistentes si el almacenamiento supera el 90%.

3.1.1.3. Políticas de backup

1 **Tipo de backup:** Se realizarán backups completos cada 4 incrementales.
2 **Modo de backup:** Si el almacenamiento de un servicio se basa en bloque (bases de datos), se escogerá el modo *CBT* de OpenNebula para seguir cambios a nivel de bloque. Si un servicio solo interactúa con el sistema de ficheros, se escogerá el modo *snapshot* con diferenciales de *Copy On Write*.
3 **Proceso de backup:** Si un servicio cuenta con solución de backup integrada, se deberá activar dicha solución antes de empezar el backup desde OpenNebula.

3.1.1.4. Políticas de almacenamiento

1 **Uso de imágenes:** El uso de imágenes persistentes estará restringido a máquinas que solo tengan desplegada una instancia y no haya una imagen persistente ya creada en al que basarse. Se deberá hacer uso de imágenes no persitentes en cualquier otro caso.

3.1.1.5. Políticas de aplicación

1 **Gateways de acceso a Internet:** Toda aplicación debe tener tener conexión a Internet a través de alguna de las puertas de enlace definidas para ese propósito dentro de la federación.
2 **Etiquetado:** Toda aplicación debe estar correctamente etiquetada. Deberá distinguirse si corresponde a un servicio en producción, en desarrollo o pruebas. Para ello emplear los nombres "prod", "dev" y "test".
3 **Replicación mínima:** Las aplicaciones etiquetadas como producción, deberán tener, al menos, 2 instancias en ejecución en entidades federadas distintas.

3.1.1.6. Compatibilidad técnica

Cada entidad debe presentar una instancia de OpenNebula como software de gestión de máquinas virtuales y, opcionalmente, Ceph como sistema de almacenamiento, en su defecto deberá tener un diseño del almacenamiento similar al planteado en la fase de diseño. Se deberá emplear *KVM* como hipervisor y *Restic* como driver del datastore de backup. El resto de servicios pueden tener una implementación diferente a la presentada en los siguientes apartados.

3.1.2. Sistema de control de accesos y validación de políticas

```
package naming

default compliant := false

compliant if {
    resource_name := split(input.action.resource_name, "-")
    resource_name[0] == input.context.federation_entity
    resource_name[1] == input.context.username
    regex.match(`[A-Za-z]+`, resource_name[2])
}
```

Listado 1: Política de nombrado de máquinas virtuales empleando lenguaje Rego.

3.1.3. Servicio de monitorización

3.1.4. Catálogo de la federación

3.2. Implementación de la Infraestructura

Dado que no se contaba con la infraestructura necesaria para levantar todos los componentes de la infraestructura en distintas máquinas, se ha simulado el entorno real mencionado en el diseño.

El despliegue está basado en contenedores *Podman*. Sabiendo que el backup y el sistema de control de accesos se desplegarían en máquinas virtuales, se ha hecho la infraestructura lo más ligera posible. Cada contenedor se ejecutará en modo no privilegiado aunque se les asignarán las *capabilities* *NET_RAW* y *NET_ADMIN* que permitan modificar la red. Cualquier acceso a dispositivos físicos, en el caso de OSDs o asegurar la persistencia de las bases de datos, se realizado a través de volúmenes con los dispositivos previamente configurados y montados en una ruta específica en el host.

Cada instancia de OpenNebula tendrá como único host de virtualización el mismo servidor donde se despliega. Por ello ha habido que cambiar la política de nombrado por defecto de los dominios de *libvirt* para que no haya solapes entre las máquinas de las distintas instancias.

3.2.1. Red

La Figura 3 pretende simular la red descrita en el diseño, teniendo únicamente un servidor.

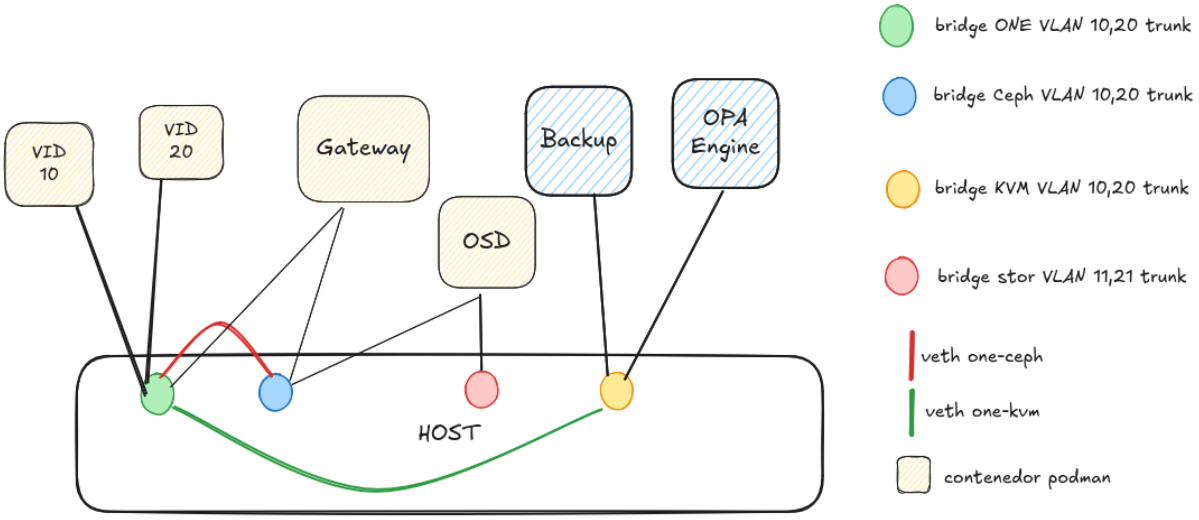

Figura 3: Infraestructura de red simulada

La distinción geográfica entre sedes se consigue mediante *VLANs*, haciendo que los componentes de cada entidad tengan que pasar por el router más cercano para poder comunicarse. Así, las interfaces de cada contenedor han sido configuradas correspondientemente con la *VLAN* a la que pertenece su servicio.

Para simular la comunicación interna de cada entidad, se han empleado interfaces tipo *bridge huérfano*[1] que permiten simular el enrutado de capa 2 de un switch. Hay 4 de estas interfaces: *br_ceph*, para la red pública Ceph (monitores, *managers*, OSDs y CephFS); *br_stor*, para la privada de Ceph (red de replicación de OSDs); *br_one*, la pública de OpenNebula (frontales, base de datos y acceso a la federación); y *br_kvm*, a la que se acoplan las máquinas virtuales. La interconexión entre los bridge se ha conseguido mediante interfaces de pares VETH. Así, la figura del router queda para relegada para el acceso a internet o comunicación entre VLAN diferentes. Cabe destacar que ambas interfaces mencionadas cuentan con la capacidad de *trunk*[2] para poder transmitir paquetes de varias VLAN.

Se ha definido una red frontal de todo el sistema que controle el acceso a Internet. Así, todo el tráfico hacia el exterior es procesado por una serie de reglas de red. En este caso, se han establecido reglas de retransmisión de paquetes entre VLAN como puede verse en la Listado 2. Este router es un contenedor *Alpine Linux* con las reglas establecidas ya que no hay más necesidades de enrutamiento que las mencionadas. Este contenedor se acopla a los bridge que simulan las redes públicas internas de cada entidad: *br_one* y *br_ceph*. Esto supone crear bucles de red entre el router, los *bridge* y la interfaz VETH que une los *bridge* por lo que se ha activado el protocolo STP en los bridge, otorgando máxima prioridad a los bridge para que la interfaz de VETH no quede en desuso.

```
iptables -t nat -A POSTROUTING -o %network[:podman][:interface]% -j MASQUERADE
iptables -t nat -A POSTROUTING -o $virt::containers::iface_one_10 -j MASQUERADE
iptables -t nat -A POSTROUTING -o $virt::containers::iface_one_20 -j MASQUERADE
iptables      -A      FORWARD      -i      $virt::containers::iface_one_10      -o
$virt::containers::iface_one_20 -j ACCEPT
iptables      -A      FORWARD      -i      $virt::containers::iface_one_20      -o
$virt::containers::iface_one_10 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Listado 2: Reglas de IPTables para retransmisión de VLAN.

La red de contenedores se implementado mediante *macvlan*, redes virtuales que se acoplan a una interfaz física, sobre las interfaces VLAN en cada bridge.

3.2.2. Almacenamiento

El almacenamiento local del entorno de simulación consta de 3 discos mecánicos de 1TB dispuestos en RAID5. Para ofrecer el almacenamiento al resto de la infraestructura, se ha definido un grupo volumen con volumen físico los discos mencionados y tantos volúmenes lógicos como necesite cada componente. Así, para los OSDs, que en total son 6, se han instanciado 6 volúmenes lógicos y sobre ellos se ha instalado un sistema de ficheros XFS.

En Ceph se almacenan las bases de datos de los frontales de OpenNebula, las imágenes de máquinas virtuales, su estado y los backup. Para ello se han definido los siguientes *pool ceph*[3], donde *eid* es el identificador de cada entidad (10 o 20): *one-db-eid*, que contiene 3 imágenes, una por cada base de datos del frontal en *ha*[1] *one-eid*, que da soporte a los datastores de OpenNebula y los pooles de backup *backup_data-ceph-eid* y *backup_meta-ceph-eid*, definidos en la siguiente sección. Cada uno de los pooles mencionados anteriormente, a excepción de los de backup, están configurados en modo réplica 3, 3 copias idéndticas en 3 OSD diferentes, y primario-copia, el dato se da por asentado cuando una mayoría simple de OSDs se lo comunican la principal.

3.2.3. Despliegue

El despliegue de la infraestructura se ha realizado mediante la herramienta de automatización Puppet. Se han definido los módulos *net*, *storage* y *virt* para el despliegue de cada parte de la infraestructura por separado. En el módulo *net* están definidos los recursos de red definidos en la Sección 3.2.1 con dependencias de despliegue internas: primero se despliegan los *bridge* y después, sobre estas, las interfaces VETH. El módulo *storage* contiene tanto el almacenamiento local, con la definición de la estructura de LVM, como los recursos Ceph. El almacenamiento local no tiene dependencias internas ni externas, pero Ceph depende de los recursos de red y del almacenamiento local. Internamente, primero se despliegan los monitores, después los *managers* y posteriormente los OSD y MDS. El módulo *virt* contiene todo lo relacionado con la infraestructura de virtualización. Esta consiste en el *gateway* y los componentes de OpenNebula. El contenedor de la puerta de enlace es el primero en ser desplegado, seguido de las bases de datos y los frontales de OpenNebula. Posteriormente se establece la comunicación entre cada frontal, formando un único clúster en *ha*[1]. En la Sección B de los anexos se encuentra el grafo de dependencias entre recursos entero.

Entre los varios recursos personalizados que se han definido en Puppet, cabe destacar el de aprovisionamiento de contenedores. Pese a sonar contraintuitivo (los contenedores vienen aprovisionados ya antes de ser desplegados), el procedimiento por el que se establece *ha*[1] entre los tres frontales de OpenNebula requiere acceso a los contenedores en tiempo de ejecución, siendo capaz de modificar ficheros de configuración y ejecutar comandos, esperando su respuesta y sincronizando el estado de otros contenedores. El recurso definido permite esta operativa y ofrece una interfaz por la que acceder a recursos del contenedor a través de variables personalizadas.

En concreto, primero se despliegan 3 frontales en *ha*[1] que forman la zona maestra de la federación, seguido viene el frontal maestro de la segunda zona con el que se establece la federación. Una vez establecida, se incorporan dos frontales más a la zona esclava.

3.3. Implementación de la Recuperación ante Desastres

La implementación se enfoca en las dos necesidades planteadas en el diseño: servicio de backup y repositorio que se comunique con Ceph. La solución natural para el repositorio es CephFS, el cual se montará en el directorio */var/lib/one/datastores* de la máquina virtual donde se ejecute el servicio de backup. Para este servicio se ha escogido *Restic*, el cual cuenta con integración nativa en OpenNebula e implementa las necesidades de backup básicas (deduplicación, versionado y encriptado).

En Ceph se han definido dos pooles que dan soporte al sistesma de ficheros *backup*. El de metadatos está en modo réplica 3 y el de datos está distribuido mediante erasure code 3-1, para soportar el fallo de hasta 1 OSD (o una entidad si se expande el clúster), corregirlo y restaurar la E/S después de su corrección. Al tratarse de almacenamiento archivado (datos fríos), 1 bloque de paridad se considera aceptable. Los pooles y el sistema de ficheros en base a estos se ha definido en el manifiesto *virt/manifests/services/opennebula/ceph/backup.pp* Puppet.

Para desplegar Restic en OpenNebula, se ha definido la máquina virtual con usuario *oneadmin*, montado el sistema de ficheros Ceph *backup*, e instalado las dependencias *rsync* y *qemu-img*. Se ha creado un usuario llamado *backup* con permisos de explotación de la máquina virtual a nivel de zona. El nombre de la máquina virtual es *one-10-backup-backup* siguiendo con la política de nomenclatura y se ha etiquetado como *one-10-prod*. También se ha declarado un *datastore* cuyo driver es *restic* y no emplea puentes como intermediarios para almacenar las imágenes temporalmente. La máquina virtual contiene las claves *ssh* de los 3 frontales para su acceso remoto sin contraseña, requerido por la integración de *Restic*. Esta configuración es necesaria por cada entidad federada. La definición de la máquina virtual, usuario y datastore han sido definidos en el manifiesto *backup_datastore.tofu* OpenTofu.

Las tareas definidas, tienen en cuenta las políticas de backup. Para los sistemas de ficheros, se detendrá la E/S de dos maneras diferentes: si la máquina se ha creado desde OpenNebula, mediante un agente nativo (especificado como *FS_FREEZE="AGENT"*) o parando la máquina momentáneamente (*FS_FREEZE="SUSPEND"*). Para este recurso se realizarán backups incrementales usando como base el *snapshot* cada 4 repeticiones. Así, se guardará también el estado de la máquina de forma consistente en caso de fallo. En el Listado 3 se ve la aplicación de una de estas tareas para los servicios básicos de OpenNebula (3 frontales en HA con sus bases de datos).

Para la incorporación de nuevos servicios al sistema de backup, se ha definido un *hook*. Este leerá la etiqueta de la máquina virtual para determinar su prioridad y asignará su identificador a la tarea que le corresponda según la prioridad. Habrá igualmente otro *hook* definido para el borrado de máquinas donde se eliminará su identificador de la tarea.

```
NAME = "Servicios básicos"

BACKUP_VMS   = "0, 1, 2"
DATASTORE_ID = 102

FS_FREEZE = "SUSPEND"
KEEP_LAST = "4"
MODE      = "INCREMENT"

PRIORITY  = 90 # prioridad alta
EXECUTION = "SEQUENTIAL" # requerido por Restic

SCHED_ACTION = [
    REPEAT="0", # repetir cada semana
    DAYS="1,5", # lunes, miércoles y viernes
    END_TYPE="0", # tarea permanente
    TIME="18000" ] # 5 AM
```

Listado 3: Definición de la tarea de backup one-10-90 en OpenNebula

4. Trabajo futuro

- Establecer relación de ejecución entre hooks. Si un recurso falla en desplegarse por incumplir una política concreta, no ejecutar el hook de incorporación a las tareas de backup.

5. Anexos

A Hooks OpenNebula

Todos los *hooks* definidos en OpenNebula empiezan con unas líneas que definen la ubicación de las librerías de ruby que usa OpenNebula.

```
ONE_LOCATION = ENV["ONE_LOCATION"]

if !ONE_LOCATION
  RUBY_LIB_LOCATION = "/usr/lib/one/ruby"
  GEMS_LOCATION = "/usr/share/one/gems"
else
  RUBY_LIB_LOCATION = ONE_LOCATION + "/lib/ruby"
  GEMS_LOCATION = ONE_LOCATION + "/share/gems"
end

if File.directory?(GEMS_LOCATION)
  Gem.use_paths(GEMS_LOCATION)
end

$LOAD_PATH << RUBY_LIB_LOCATION
# carga librerías personalizadas
$LOAD_PATH << "/usr/share/one/hooks-lib"
```

A.1 Hooks BackupJobs

Inclusión del ID de un máquina virtual a una tarea de backup

```
#!/usr/bin/env ruby

require "base64"
require "nokogiri"

TAGS_MAP = {
  :prod => "80",
  :dev => "50",
  :test => "10",
}

#
# Al crear una MV, incluir el id en la tarea correspondiente
#

api_info = Nokogiri::XML(Base64::decode64(ARGV[0]))

success = api_info.xpath("/CALL_INFO/RESULT").text.to_i == 1

if !success
  puts "Resource wasn't created, hook not executed"
  exit 0
end

#
# Obtener tag con el que se ha creado
```

```
#

def parse_template(template_content)
  template_content.split(/\n/).inject({}) do |prev, kv|
    if kv != "]"
      key, value = kv.strip.split("=")
      value = value.strip.gsub('"', "")
      if value != "["
        prev[key] = value
      end
    end
    prev
  end
end

backupjob_file = "one-10-"
current_vm_id = ""
api_info.xpath("/CALL_INFO/PARAMETERS/PARAMETER").each do |param|
  if param.xpath("POSITION").text.to_i == 2 and param.xpath("TYPE").text == "IN"
    resource_template = parse_template(param.xpath("VALUE").text)
    label = resource_template["LABELS"][..-2]
    puts label
    backupjob_file += TAGS_MAP[label.to_sym]
    current_vm_id = `onevm list -f NAME=#{resource_template["NAME"]} --no-header | cut
-d ' ' -f4`
  end
end
puts "BACKUPJOB_FILE = #{backupjob_file}"
puts "CURRENT_VM_ID = #{current_vm_id}"

#
# Obtener máquinas actuales
#
backupjob_id = `onebackupjob list -f NAME=#{backupjob_file} --no-header | cut -d ' '
-f4`
current_vms = `onebackupjob show #{backupjob_id} | grep BACKUP_VMS`
puts "BACKUPJOB_ID = #{backupjob_id}"
puts "CURRENT_VMS = #{current_vms}"

modified_vms = current_vms[..-2] + current_vm_id + "\""
#
# Modificar backupjob
#
temp_file = "/tmp/update_backupjob_#{backupjob_file}.tmpl"
File.write(temp_file, modified_vms)
`onebackupjob update #{backupjob_id} #{temp_file}`
File.delete(temp_file)

exit 0
,
```

Eliminación del ID de un máquina virtual a una tarea de backup

```ruby
#!/usr/bin/env ruby

require "base64"
require "nokogiri"

TAGS_MAP = {
  :prod => "80",
  :dev => "50",
  :test => "10",
}

#
# Al crear una MV, incluir el id en la tarea correspondiente
#

api_info = Nokogiri::XML(Base64::decode64(ARGV[0]))

success = api_info.xpath("/CALL_INFO/RESULT").text.to_i == 1

if !success
  puts "Resource wasn't created, hook not executed"
  exit 0
end

#
# Obtener tag con el que se ha creado
#

def parse_template(template_content)
  template_content.split(/\n/).inject({}) do |prev, kv|
    if kv != "]"
      key, value = kv.strip.split("=")
      value = value.strip.gsub('"', "")
      if value != "["
        prev[key] = value
      end
    end
    prev
  end
end

current_vm_id = ""
api_info.xpath("/CALL_INFO/PARAMETERS/PARAMETER").each do |param|
  if param.xpath("POSITION").text.to_i == 2 and param.xpath("TYPE").text == "IN"
    action = param.xpath("VALUE").text
    if action != "terminate" and action != "terminate-hard"
      exit 0
    end
  end
```

```
  if param.xpath("POSITION").text.to_i == 2 and param.xpath("TYPE").text == "OUT"
    current_vm_id = param.xpath("VALUE").text
  end
end
label = `onevm show #{current_vm_id} | grep LABELS | cut -d '"' -f2`
backupjob_file = "one-10-" + TAGS_MAP[label[..-2].to_sym]
puts "BACKUPJOB_FILE = #{backupjob_file}"
puts "CURRENT_VM_ID = #{current_vm_id}"

#
# Obtener máquinas actuales
#
backupjob_id = `onebackupjob list -f NAME=#{backupjob_file} --no-header | cut -d ' '
-f4`
current_vms = `onebackupjob show #{backupjob_id} | grep BACKUP_VMS`
puts "BACKUPJOB_ID = #{backupjob_id}"
puts "CURRENT_VMS = #{current_vms}"

modified_vms = current_vms.gsub(/5/, "")
modified_vms = modified_vms.gsub(/,,/, ",")
#
# Modificar backupjob
#
temp_file = "/tmp/update_backupjob_#{backupjob_file}.tmpl"
File.write(temp_file, modified_vms)
`onebackupjob update #{backupjob_id} #{temp_file}`
File.delete(temp_file)

exit 0
,
```

B Manifiestos Puppet