UNIVERSIDAD DE LA REPÚBLICA

FACULTAD DE INGENIERÍA

Instituto de Computación - InCo

PROYECTO DE GRADO

Bas7ion

Despliegue y configuración de infraestructura en la nube

Autores:

Martín Alayón

Laureano Klüver

Diego Méndez

Cliente:

Rafael Álvarez

Tutores:

Guzmán Llambías

Javier Barreiro

Montevideo, Uruguay

2018

RESUMEN

Actualmente una gran cantidad de soluciones empresariales tienen alojada su infraestructura en los proveedores de la nube (AWS, Azure, etc.), lo cual requiere una gestión de la infraestructura generada. Esto implica que los administradores de infraestructura deban acceder a los portales web de los proveedores o utilizar otros mecanismos como pueden ser la API o el cliente de consola.

En muchos casos los diseños de las infraestructuras son similares y por lo tanto a la hora de desplegar recursos existen tareas que se repiten, las cuales se pueden realizar de forma manual y/o automatizada. Para automatizar el despliegue de los recursos existe el concepto de infraestructura como código, el cual permite especificar, a través de código, la infraestructura a desplegar. Existen herramientas que utilizan el concepto anterior, sin embargo manejan conceptos de infraestructura específicos para cada proveedor. Esto implica que los administradores de infraestructura deban contar con conocimientos específicos de cada uno de ellos.

Este proyecto tiene como objetivo principal el desarrollo de una herramienta que, a través de infraestructura como código, permita generalizar los conceptos de infraestructura independientemente del proveedor donde se despliegan los recursos.

Inicialmente se analizó la realidad planteada para comprender la problemática y determinar los requerimientos. A partir de estos se investigaron las tecnologías existentes (Terraform, AWS Cloud Formation, Fog, Apache Libcloud), y se desarrolló una prueba de concepto para validar y mitigar los riesgos técnicos. Posteriormente se realizó un diseño extensible que permite integrar de forma sencilla nuevos proveedores. A partir de la implementación se obtuvo una herramienta, denominada Bas7ion (Bastion 7), desarrollada en Python la cual utiliza las tecnologías Vault, Apache Libcloud y Ansible.

Bas7ion es una herramienta que le da un valor agregado al área de infraestructura dado que sus usuarios no requieren contar con conocimientos específicos de los diferentes proveedores al realizar el despliegue de los recursos.

Palabras clave: Computación en la nube, Proveedores de infraestructura en la nube, DevOps, Infraestructura como Código.

Índice General

1. Introducción

Actualmente una gran cantidad de soluciones empresariales tienen alojada su infraestructura en los proveedores de la nube (AWS, Azure, etc.), lo cual requiere una gestión de la infraestructura generada. Esto implica que los administradores de infraestructura deban acceder a los portales web de los proveedores o utilizar otros mecanismos como pueden ser la API o el cliente de consola.

A la hora de generar la misma infraestructura en distintos proveedores, independientemente del mecanismo utilizado, se llevan a cabo tareas similares en cada uno de estos. Existen herramientas (Terraform, Fog, Apache Libcloud) que permiten automatizar dichas tareas, pero carecen de una visión normalizada de los recursos a la hora de desplegar la infraestructura en distintos proveedores. Por lo tanto, existe la complejidad de contar con conocimientos específicos para cada uno de los proveedores.

En el marco de este proyecto, el departamento de infraestructura de la empresa Pyxis [1] detectó ciertas mejoras a implementar en su operativa diaria, dentro de las que se incluye eliminar la complejidad mencionada anteriormente. Esto motivó a la presentación de este proyecto de grado por parte de dicha empresa (de ahora en más cliente) en el cual propone contar con una herramienta que normalice los conceptos de infraestructura para automatizar el despliegue de los recursos en los diferentes proveedores.

1.1. Objetivos

El objetivo general de este proyecto es desarrollar una herramienta que, a través de infraestructura como código, permita generalizar los conceptos de infraestructura independientemente del proveedor donde se despliegan los recursos.

Para lograrlo se definen los siguientes objetivos específicos:

- Analizar diferentes herramientas existentes en el mercado, para luego seleccionar aquellas que más se adecuan a los requerimientos del proyecto.
- Determinar el conjunto de operaciones y servicios que ofrecen en común los proveedores más relevantes.
- Diseñar una solución que permita agregar fácilmente nuevas operaciones y proveedores.
- Cumplir con los requerimientos acordados con el cliente.

1.2. Aportes

A continuación, se presentan los principales aportes luego de concluido el proyecto:

- Una herramienta que permite desplegar infraestructura en la nube independientemente del proveedor a utilizar.
- Diseño de una arquitectura de software y un modelo genérico de recursos y servicios de infraestructura en la nube, el cual es fácilmente extensible para la integración de nuevos proveedores de la nube.
- Conocimiento sobre las diferentes tecnologías existentes en el área de infraestructura en la nube.

1.3. Organización del documento

El contenido del documento está organizado en capítulos, los cuales se presentan a continuación.

A lo largo del capítulo 2 se desarrollan los conceptos y definiciones que son necesarios para comprender el resto del documento.

En el capítulo 3 se presenta la problemática planteada por el cliente y las motivaciones para la realización del proyecto.

A continuación, en el capítulo 4 se describen los requerimientos del proyecto y se define el alcance acordado.

A partir del análisis mencionado anteriormente, en el capítulo 5 se presenta el diseño de la solución, partiendo desde la arquitectura de software hasta aspectos específicos de diseño.

Luego de realizado el diseño, en el capítulo 6 se describe la implementación de los principales componentes de la solución, así como las tecnologías utilizadas. Además, se describen las funcionalidades más relevantes. Por último, se mencionan algunas de las problemáticas que se presentaron durante dicha etapa, finalizando con una descripción de las pruebas realizadas.

En el capítulo 7 se explica cómo fue llevada a cabo la gestión del proyecto.

El capítulo 8 contiene el trabajo a futuro, donde se mencionan posibles mejoras a tener en cuenta en la solución.

Por último, en el capítulo 9 se presentan las conclusiones del proyecto, resumiendo el trabajo realizado y describiendo sus contribuciones.

2. Marco Conceptual

En este capítulo se presentan los conceptos necesarios para comprender la solución del proyecto. En primer lugar, se da una descripción general de computación en la nube con sus características esenciales, además de la definición de proveedor de servicios en la nube y sus características. Posteriormente se presentan los modelos de despliegue y servicio existentes, como así también los servicios de los proveedores más relevantes de la industria (Amazon Web Service y Microsoft Azure), los cuales fueron los utilizados en el proyecto. Finalmente se presenta el concepto de DevOps y las prácticas fundamentales de dicha disciplina, haciendo énfasis en la práctica de Infraestructura como Código, la cual es esencial para el desarrollo del proyecto.

2.1. Computación en la Nube

Cuando una organización necesitaba desplegar una solución de software con características específicas en cuanto a los recursos físicos (servidores, routers, etc.) era común que dicha organización adquiriera infraestructura a proveedores de hardware, lo que se traducía en un costo de inversión considerable. Adicionalmente a esto, era responsabilidad de la organización, la configuración y administración de los componentes físicos adquiridos. Esta infraestructura física estaba localizada en data centers, ya sea on-premise o en proveedores locales.

En los últimos años, la generación y administración de la infraestructura ha venido evolucionando hacia un modelo de computación en la nube, el cual es una nueva forma de alojar las soluciones mencionadas anteriormente.

IBM define computación en la nube como la distribución de recursos informáticos a demanda (desde aplicaciones hasta centro de cómputos) a través de Internet, basado en un modelo de pago por uso. [2]

Según el NIST (National Institute of Standards and Technology), la computación en la nube es un modelo que permite el acceso a través de Internet a un conjunto de recursos (redes, servidores, almacenamiento, aplicaciones, servicios) que se pueden aprovisionar y liberar rápidamente. [3]

En el marco de este proyecto, se considera a la computación en la nube como un modelo que permite a los usuarios tener acceso a través de Internet a un conjunto de recursos, los cuales, a partir de un modelo de pago por uso pueden ser configurados y utilizados rápidamente.

Hoy en día las organizaciones tienen la posibilidad de elegir dónde alojar su infraestructura, ya sea on-premise, en la nube o mediante la combinación de éstas.

A continuación, se presentan ciertas características esenciales de la computación en la nube [3]:

- *Autoservicio bajo demanda*: los usuarios tienen el control sobre la creación, modificación y destrucción de los distintos recursos, en todo momento y sin necesidad de interacción con intermediarios.
- *Amplio acceso a la red*: los servicios están disponibles desde la red (internet) y son accedidas a través de mecanismos estandarizados (portal web, API, cliente de consola).
- *Agrupación de recursos*: los recursos físicos son agrupados para atender a varios usuarios a la vez. Por medio de la virtualización de los recursos físicos se logra abstraer al usuario de la locación física en donde se encuentra el recurso (CPU, Disco Duro, etc.).
- *Rápida elasticidad*: los recursos escalan rápidamente a medida que aumenta o disminuye la demanda en su utilización (más/menos CPU, RAM, Disco Duro).
- *Servicios medibles*: se utilizan métricas que ayudan a aproximar la utilización real de los recursos por parte de los usuarios (tiempo, cómputo, espacio, etc.), aspecto vital a la hora de cobrar por los servicios ofrecidos.

2.1.1. Proveedores de Servicios

Para hacer uso de los servicios de computación en la nube existen empresas, denominadas proveedores de servicios de la nube, que ofrecen estos servicios a los usuarios.

En la actualidad existen varios proveedores, ya que muchas de las grandes empresas han incursionado en este negocio. Algunos de los proveedores más importantes son:

- Amazon Web Services [5]
- Microsoft Azure [8]
- Google Cloud Platform [9]
- VMWare Cloud [10]
- IBM Cloud [11]

A la hora de elegir el o los proveedores de servicios de la nube existen ciertos indicadores, dentro de los cuales se encuentran:

- Costo por el uso de los recursos
- Localización de los recursos físicos

- Restricciones geo-políticas, jurídicas y normativas
- Confiabilidad y seguridad de los datos

Estos indicadores determinan que los proveedores de servicios de la nube sean una buena elección para las organizaciones que requieren servicios en internet, ya que les permite mejorar su eficiencia y beneficiarse de la economía de escala.

Adicionalmente a los indicadores anteriores se deben tener en cuenta los servicios que ofrecen los proveedores. Algunos de los servicios más importantes son:

- *Máquinas Virtuales:* Servicios de infraestructura que permiten utilizar servidores virtuales dentro de una gran variedad de configuraciones de hardware (CPU, RAM, etc.) y software (sistema operativo, aplicaciones, etc.).
- *Almacenamiento:* Servicios de infraestructura que ofrecen almacenamiento y acceso escalable para cualquier tipo y volumen de información.
- *Redes Virtuales:* Servicios de infraestructura que permiten diseñar y configurar redes virtuales dentro de la nube.
- *Redes de Contenido:* Servicios de CDN que permiten utilizar una red global y distribuida de servidores proxy para la entrega de contenido (vídeos, imágenes, etc.).
- *Resolución de Nombres:* Servicios de DNS escalables y de alta disponibilidad. Permite la resolución de nombres en la web, como también la resolución de nombres sobre una red privada.
- *Balanceamiento de Carga:* Servicios que distribuyen el tráfico a través de múltiples destinos (máquinas virtuales, direcciones IPs, etc.).
- *Base de Datos Relacionales:* Servicios que permiten iniciar, operar y escalar bases de datos relacionales en la nube.
- *Base de Datos no Relacionales:* Servicios que permiten iniciar, operar y escalar bases de datos no relacionales en la nube.

2.1.2. Modelos de Servicios

Partiendo de una arquitectura orientada a servicios, los proveedores ofrecen distintos grados de abstracción. Se puede pensar cada grado de abstracción como capas dentro de un stack. A partir de esto, se definen tres modelos de servicios: Infraestructura como Servicio (IaaS - Infrastructure as a Service), Plataforma como Servicio (PaaS - Platform as a Service) y Software como Servicio (SaaS - Software as a Service). [3]

Hay proveedores que cubren todos los modelos de servicio y otros que se especializan en alguno de ellos. En la figura 2.1 se pueden observar cada uno de los modelos con los respectivos componentes del stack que manejan.

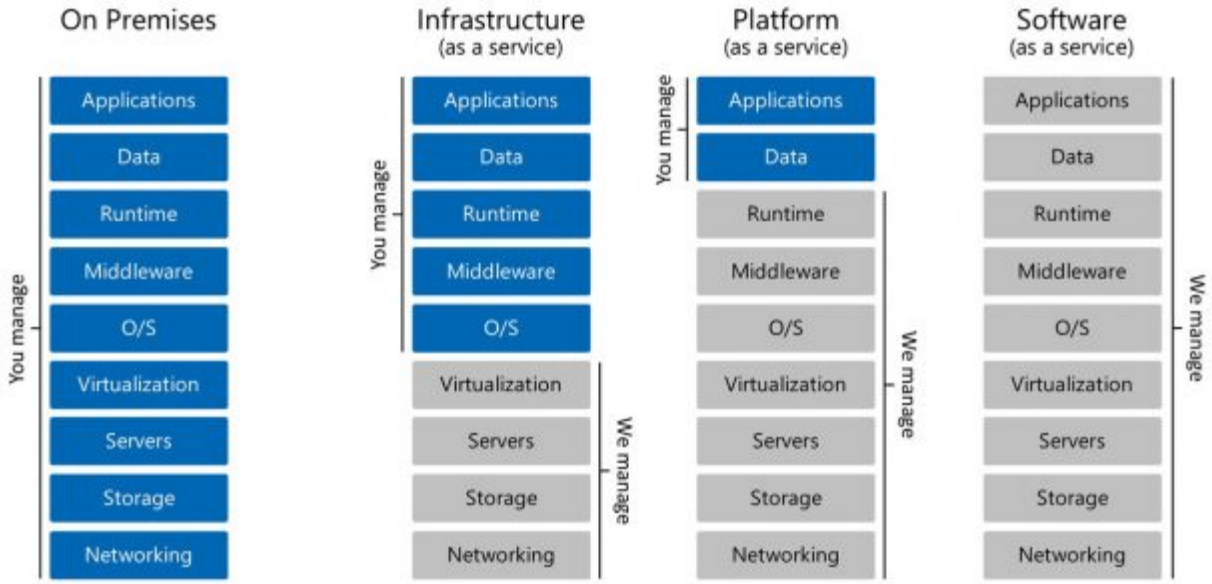


Figura 2.1: Separación de responsabilidades [12]

A continuación, se describen cada uno de los modelos de servicio mencionados anteriormente.

2.1.2.1. Infraestructura como Servicio

El proveedor ofrece al usuario la capacidad de definir de forma virtual la infraestructura que más se ajuste a sus necesidades. El usuario tiene el control de dicha infraestructura de forma indirecta a través del uso de los recursos virtuales, agregando una capa adicional de abstracción.

Los servicios que ofrecen se pueden agrupar en las siguientes categorías: capacidad de cómputo, redes y almacenamiento. Dentro de capacidad de cómputo se encuentran las máquinas virtuales y su configuración (imagen de sistema operativo, tamaño de almacenamiento, etc.). Con respecto a las redes ofrece la configuración de los elementos de red, como las redes virtuales, subredes, interfaz de red, firewalls, etc. Por

último, en almacenamiento se encuentran los elementos de almacenamiento como los discos virtuales, snapshots, etc.

Algunos de los principales proveedores de IaaS son:

- Amazon Web Services [5]
- Microsoft Windows Azure [8]
- Rackspace [13]
- VMware Cloud [10]

2.1.2.2. Plataforma como Servicio

El proveedor ofrece un conjunto de servicios y el uso de su infraestructura, donde se brinda un entorno de desarrollo e implementación en la nube. Dentro de los servicios se incluyen herramientas de desarrollo, sistemas de administración de base de datos, librerías, frameworks, etc.

Los principales proveedores de PaaS son:

- Google App Engine [14]
- OpenShift [15]
- Heroku [16]
- Cloud Foundry [17]

2.1.2.3. Software como Servicio

El proveedor ofrece el uso de aplicaciones que se están ejecutando dentro de su infraestructura. El usuario solo tiene control sobre la aplicación de la cual está haciendo uso. La infraestructura por detrás no puede ser modificada directamente, sí indirectamente por medio de la configuración de la aplicación.

Algunos ejemplos de SaaS son:

- Google Drive [18]
- Gmail [19]
- Dropbox [20]
- Office 365 [21]

2.1.3. Modelos de Despliegue

Dentro de los modelos de despliegue de la nube se encuentran: Nube Pública, Nube Privada y Nube Híbrida. En este documento, cuando se utilice el concepto de nube, se estará haciendo referencia a una nube pública. [22]

En una Nube Pública, todos los componentes (hardware y software) de la infraestructura subyacente son propiedad del proveedor, que también los administra. Se comparte el mismo hardware, almacenamiento y dispositivos de red con otras organizaciones.

Por otra parte, en un Nube Privada la infraestructura física está a disposición exclusiva de una organización y sus respectivas unidades y/o departamentos. La infraestructura puede estar alojada dentro de la organización (on-premise) o remota, gestionada por la propia organización y/o por terceros.

Las ventajas de las nubes públicas con respecto a las nubes privadas son:

- *Costos inferiores*: no es necesario adquirir hardware o software, solo se paga por el servicio que se utiliza.
- *Sin mantenimiento*: el proveedor se encarga de ello.
- *Escalabilidad*: existen recursos a petición para satisfacer las necesidades de las organizaciones.
- *Disponibilidad*: una amplia red de servidores garantiza la alta disponibilidad de los servicios.

Por otra parte, las ventajas de las nubes privadas con respecto a las nubes públicas son:

- *Flexibilidad*: las organizaciones pueden personalizar el entorno de la nube para satisfacer necesidades empresariales específicas.
- *Seguridad*: los recursos no se comparten con otros, por lo tanto, es posible contar con mayores niveles de control y seguridad.

Adicionalmente, existe el concepto de Nube Híbrida, una combinación de nube pública y privada, donde las organizaciones puedan beneficiarse de las ventajas de ambas. En una nube híbrida, los datos y las aplicaciones pueden estar alojadas en diferentes tipos de nubes, para de esta forma obtener más flexibilidad y opciones de implementación. Por ejemplo, una empresa puede utilizar la nube pública para satisfacer necesidades de gran volumen con menor seguridad, como su página web, y la nube privada para operaciones confidenciales esenciales, como sus informes financieros.

2.1.4. Proveedores más relevantes

Dentro de los proveedores de servicios en la nube más relevantes, tomando en cuenta la adopción del mercado, Amazon Web Service y Microsoft Azure abarcan la mayor parte [4].

Amazon Web Services (AWS) se define como una plataforma de servicios en la nube que ofrece un amplio conjunto de productos, incluidas aplicaciones, almacenamiento, bases de datos, análisis, redes, herramientas para desarrolladores, herramientas de administración, Internet de las cosas, seguridad, etc. [5]

Por otra parte, Microsoft Azure es una colección de servicios integrados en la nube, donde los desarrolladores y los profesionales de las tecnologías de la información pueden crear, implementar y administrar aplicaciones a través de la red global del centro de datos de Microsoft.

A continuación, se presentan los servicios más relevantes ofrecidos por dichos proveedores [6][7].

Servicio	Amazon Web Services	Microsoft Azure
Máquinas Virtuales	Amazon EC2	Azure Virtual Machines
Almacenamiento	Amazon S3	Azure Storage
Redes Virtuales	Amazon VPC	Azure Virtual Network
Redes de Contenido	Amazon CloudFront	Azure CDN
Resolución de Nombres	Amazon Route 53	Azure DNS
Balanceamiento de Carga	Amazon Elastic Load Balancing	Azure Load Balancer
Base de Datos Relacionales	Amazon RDS	Azure SQL Database

Base de Datos no Relacionales	Amazon DynamoDB	Azure Cosmos DB

2.2. DevOps

DevOps es una combinación de prácticas y herramientas que incrementan las capacidades de una organización para llevar a cabo proyectos de software. Dentro de las capacidades se encuentra la velocidad, desarrollando y mejorando el software con mayor rapidez que las organizaciones que utilizan procesos tradicionales de desarrollo. Por otro lado, permite adaptarse a los cambios y/o trabajo no planificado de forma más ágil. Estas capacidades contribuyen en la mejora de la calidad del software desarrollado. [23]

Bajo un modelo de DevOps los equipos de desarrollo, operaciones y QA se encuentran integrados, donde los integrantes de cada equipo trabajan en todo el ciclo de vida, como se muestra en la figura 2.2. Esto permite que adquieran una variedad de habilidades no limitadas a una única función. Por otra parte, los equipos de control de calidad y de seguridad también se integran con el desarrollo y las operaciones, e intervienen durante todo el proceso.

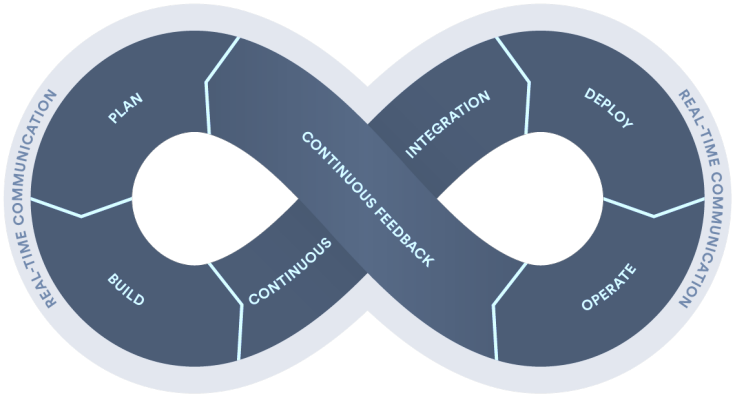


Figura 2.2: DevOps, Comunicación en tiempo real [24]

Los equipos de DevOps utilizan un stack de tecnologías y herramientas que les ayudan a desarrollar, mejorar y validar soluciones de forma rápida y fiable. Adicionalmente, algunas de estas herramientas permiten que miembros del equipo puedan realizar tareas de forma independiente, las cuales normalmente hubieran requerido la ayuda de otros equipos (por ejemplo, implementar código o aprovisionar infraestructura), incrementando así la velocidad.

Existen varias prácticas fundamentales o procesos que ayudan a las organizaciones a innovar con mayor rapidez mediante la automatización y la simplificación de los procesos de desarrollo de software, operaciones y administración de la infraestructura.

Una práctica fundamental consiste en realizar actualizaciones frecuentes, pero pequeñas. Para esto se utiliza integración continua, la cual es una práctica de desarrollo de software donde los desarrolladores integran los cambios en un repositorio central. Luego se generan versiones y finalmente se ejecutan pruebas automáticas. Tiene como objetivo principal identificar y corregir errores con mayor rapidez, mejorar la calidad del software y reducir el tiempo de validación y publicación de nuevas actualizaciones.

Otra de las prácticas principales relacionadas a la infraestructura es la denominada Infraestructura como Código. Debido a la importancia en el proyecto, se describe en la siguiente sección.

2.3. Infraestructura como Código

Infraestructura como Código es una práctica donde la infraestructura es aprovisionada y gestionada utilizando código, la cual especifica los recursos a desplegar mediante un archivo de configuración o un lenguaje de programación. A partir del código se pueden aplicar técnicas de desarrollo de software, como control de versión e integración continua. Esta práctica es posible debido a que los proveedores ofrecen una API, la cual es utilizada para ejecutar las acciones definidas en el código. [23]

Las ventajas con respecto al aprovisionamiento y gestión manual de los recursos son los siguientes:

- *Costos*: Existe una reducción en los costos, ya que esta práctica no requiere de recursos humanos para el despliegue de la infraestructura.
- *Riesgo*: Elimina el riesgo de errores humanos en la configuración de los recursos de infraestructura.
- *Velocidad*: Disminuye el tiempo de despliegue de los recursos debido a que se realiza automáticamente.

Debido a que la Infraestructura como Código aprovisiona y gestiona recursos de infraestructura, se presenta en la figura 2.3 el ciclo de vida de dichos recursos.

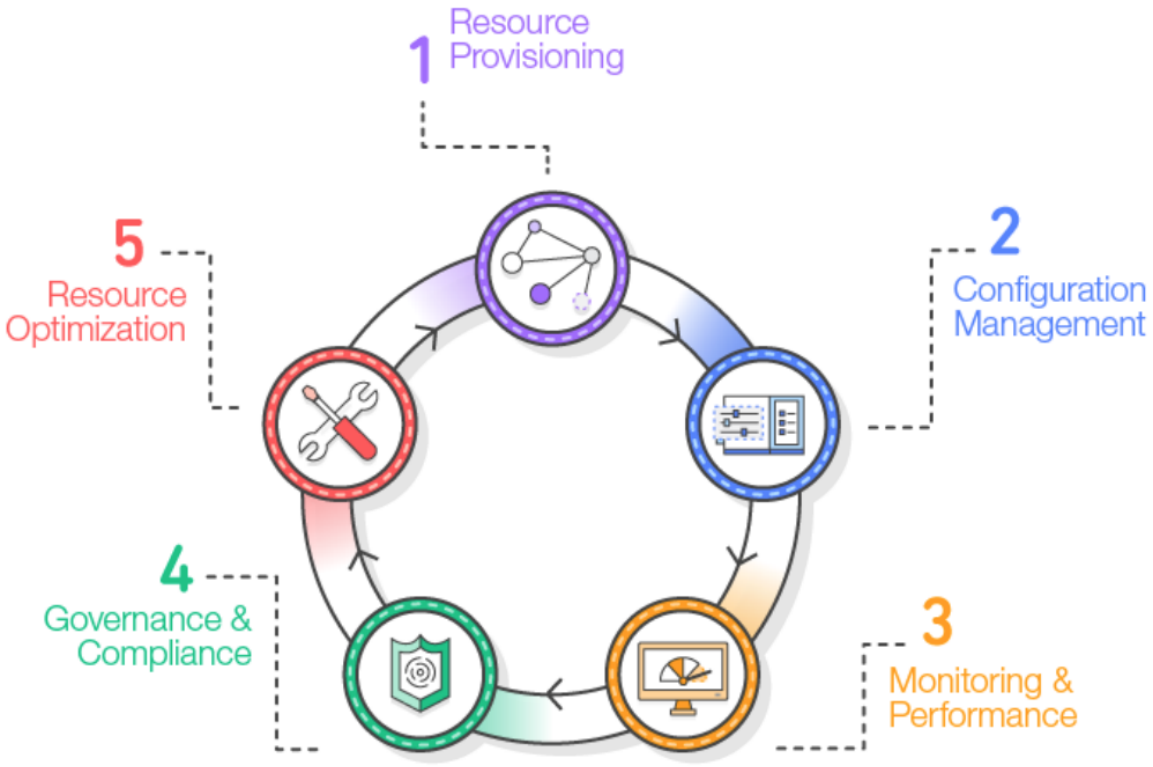


Figura 2.3: Ciclo de vida de los recursos de infraestructura [25]

1. *Resource Provisioning (Aprovisionamiento de Recursos)*: Los administradores de sistemas aprovisionan los recursos de infraestructura de acuerdo a una especificación de infraestructura definida.
2. *Configuration Management (Gestión de la Configuración)*: Los recursos de infraestructura son manejados como componentes de un sistema de gestión de configuración que soporta tareas de ajuste y corrección de errores.
3. *Monitoring and Performance (Monitoreo y Rendimiento)*: Herramientas de monitoreo y rendimiento validan el estado operacional de los recursos por el medio del análisis de métricas, archivos de log, etc.
4. *Compliance and Governance (Conformidad y Gobernanza)*: Marcos de conformidad y gobernanza agregan validación adicional para asegurar que se cumplan estándares corporativos y de la industria, así como también requerimientos regulatorios.
5. *Resource Optimization (Optimización de Recursos)*: Los administradores de sistemas revisan parámetros de rendimiento e identifican cambios necesarios para optimizar el ambiente según criterios de rendimiento y costos.

La infraestructura como código se utiliza principalmente en las etapas de Aprovisionamiento de Recursos y Gestión de la Configuración. También puede ser utilizada en algunas de las tareas de las restantes etapas.

3. Descripción de la realidad

En este capítulo inicialmente se describe la realidad actual, para luego presentar cuáles fueron las motivaciones del cliente para llevar a cabo el proyecto. Finalmente se presentan las herramientas relacionadas a la problemática de dicho proyecto.

3.1. Situación actual

Actualmente una gran cantidad de soluciones empresariales tienen alojada su infraestructura en los proveedores de la nube (AWS, Azure, etc.), lo cual requiere una gestión de la infraestructura generada. Esto implica que los administradores de infraestructura deban acceder a los portales web de los proveedores o utilizar otros mecanismos como pueden ser la API o el cliente de consola. Lo anteriormente mencionado se describe en la figura 3.1.

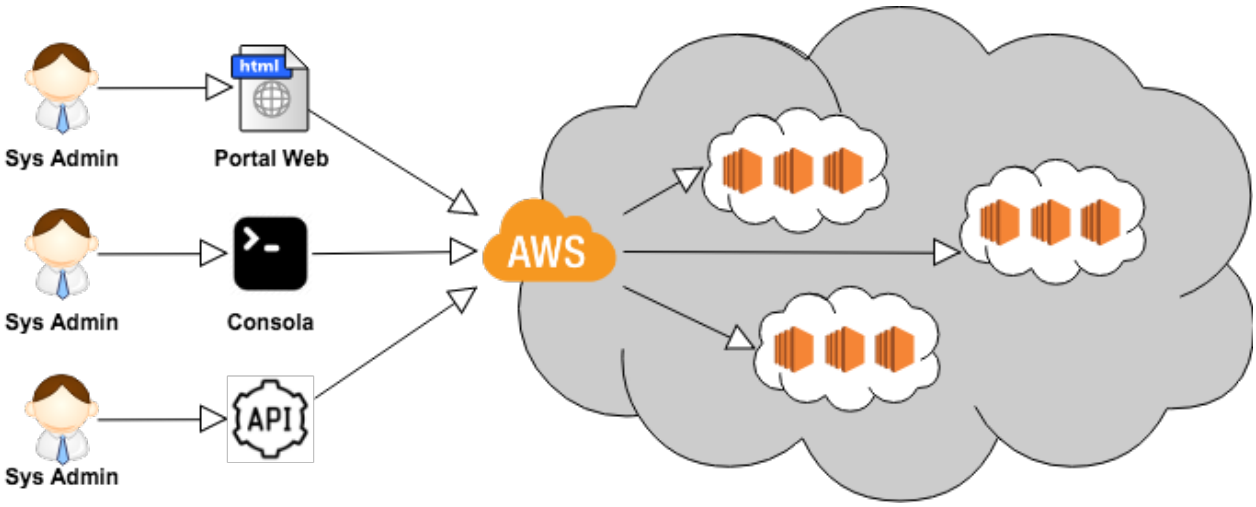


Figura 3.1: Accesos al proveedor

A la hora de generar la misma infraestructura en distintos proveedores, independientemente del mecanismo utilizado, se llevan a cabo tareas similares en cada uno de estos. A su vez, existe una complejidad en la ejecución de dichas tareas dado que es necesario contar con conocimientos específicos para cada uno de los proveedores.

3.2. Motivación

Teniendo en cuenta la situación actual, se puede observar que en la ejecución de tareas similares no se optimiza el tiempo en la generación de infraestructura. Adicionalmente, las tareas que se realizan en cada uno de los proveedores manejan conceptos en común. A partir de esto se pueden generalizar dichas tareas,

abstrayendo a los encargados de generar la infraestructura y de esta forma evitar la necesidad de contar con conocimientos específicos.

A partir de lo anterior el cliente identificó la necesidad de contar con una herramienta que permita a los usuarios, a partir de una especificación, generar infraestructura en la nube, logrando abstraer los proveedores de infraestructura a través de conceptos utilizados comúnmente en infraestructura. En la figura 3.2 se muestra un ejemplo de cómo sería la interacción entre un usuario y la herramienta.

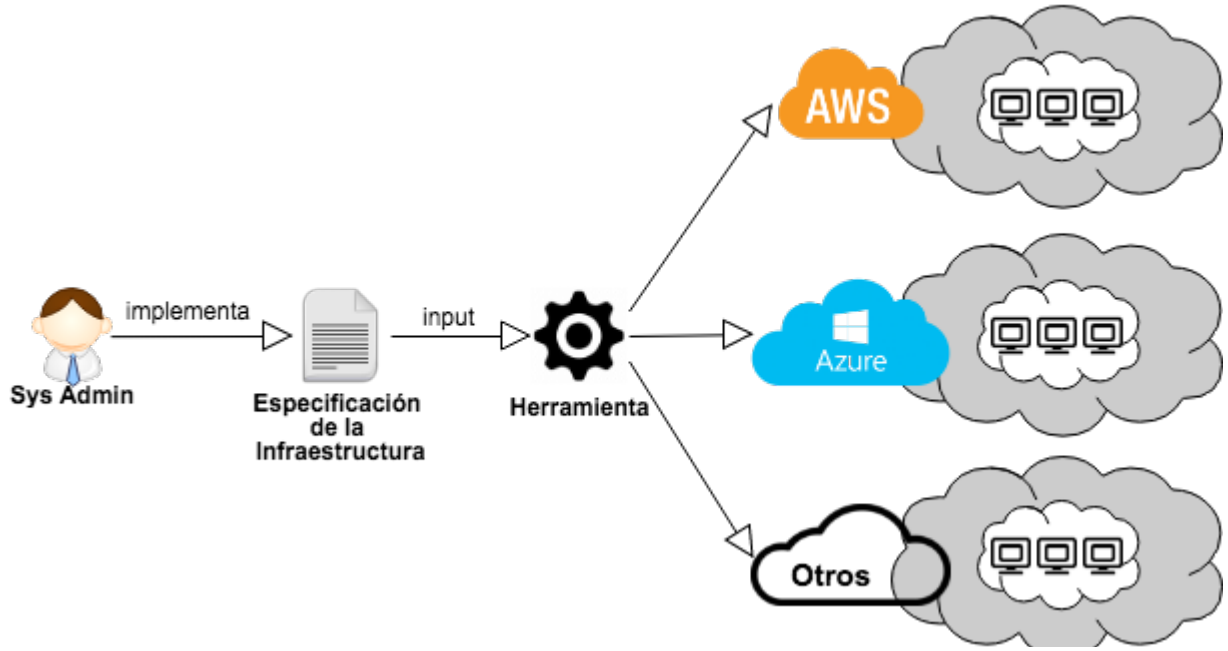


Figura 3.2: Interacción del usuario con la herramienta

3.3. Herramientas existentes

Existen herramientas que no logran solucionar en su totalidad la problemática planteada. A continuación, se presentan dichas herramientas haciendo foco en sus limitaciones.

3.3.1. Terraform

Es una herramienta para desplegar, modificar y versionar infraestructura en la nube a partir de archivos de configuración. Tiene soporte para los siguientes proveedores de infraestructura en la nube: Amazon Web Services, Microsoft Azure, Google Cloud Platform y OpenStack. [26] Esta herramienta no permite abstraer a los sysAdmin de los proveedores, debido a que es necesario definir un archivo de configuración, por proveedor, para especificar una misma infraestructura.

3.3.2. AWS Cloud Formation

Es un servicio que permite crear y gestionar recursos de infraestructura a partir de la de un archivo de configuración (plantilla). Tiene soporte únicamente para el proveedor AWS. [27]

3.3.3. Fog

Es una librería desarrollada en Ruby, que proporciona el acceso a la mayoría de los servicios ofrecidos por los proveedores a través de sus APIs correspondientes [28]. Sin embargo, no ofrece soporte para la generación y configuración de recursos de redes.

3.3.4. Apache Libcloud

Es una librería implementada en Python, que proporciona el acceso a la mayoría de los servicios ofrecidos por los proveedores a través de sus APIs correspondientes [29]. Permite abstraer la comunicación con las APIs y modelar las distintas entidades que se pueden identificar en cada proveedor [30]. Es importante resaltar que dicha librería no normaliza las diferencias entre los respectivos proveedores, sino que modela las entidades y operaciones de cada uno de ellos de manera independiente.

4. Requerimientos y Gestión de Alcance

El objetivo de este capítulo es presentar los requerimientos obtenidos en la etapa de análisis junto al alcance acordado con el cliente.

4.1. Requerimientos

A continuación, se presentan los requerimientos que debe satisfacer la solución. Los mismos se clasifican en funcionales y no funcionales. Adicionalmente los requerimientos funcionales se agrupan de acuerdo con el área de infraestructura a la que pertenecen.

4.1.1. Funcionales

4.1.1.1. Operaciones Básicas

La solución debe contar con un conjunto de operaciones básicas, las cuales se describen a continuación.

#	Requerimiento	Descripción
RF01	Crear Máquina Virtual	La solución debe poder crear una máquina virtual dentro de una subred existente en el proveedor. También se puede indicar el tamaño de la virtual (CPU, RAM) y la imagen del sistema operativo a utilizar.
RF02	Eliminar Máquina Virtual	La solución debe poder parar y dar de baja una virtual que fue previamente creada.
RF03	Iniciar Máquina Virtual	La solución debe poder levantar una máquina virtual que fue previamente detenida.
RF04	Detener Máquina Virtual	La solución debe poder pausar una máquina virtual de forma que no consuma horas de cómputo en el proveedor.
RF05	Reiniciar Máquina Virtual	La solución debe poder reiniciar una máquina virtual que fue previamente pausada.
RF06	Listar Imágenes	La solución debe poder listar imágenes de

		máquinas virtuales que están disponibles en el proveedor.
RF07	Listar Tamaños	La solución debe poder listar tamaños de máquinas virtuales que están disponibles en el proveedor.
RF08	Crear Red Virtual	La solución debe poder crear una red virtual con un nombre y un CIDR específico.
RF09	Listar Redes Virtuales	La solución debe poder listar las redes virtuales previamente creadas en el proveedor.
RF10	Eliminar Red Virtual	La solución debe poder identificar y dar de baja una Red Virtual.
RF11	Crear Subred	La solución debe poder crear una subred con un nombre y un CIDR dentro de una Red Virtual previamente creada.
RF12	Listar Subredes	La solución debe poder listar Subredes existentes dentro de una Red Virtual.
RF13	Eliminar Subred	La solución debe poder identificar y eliminar una Subred dentro de una Red Virtual.
RF14	Crear Interfaz de Red	La solución debe poder crear una Interfaz de Red dentro de una Subred.
RF15	Listar Interfaz de Red	La solución debe poder listar Interfaces de Red existentes dentro de una Subred.
RF16	Eliminar Interfaz de Red	La solución debe poder identificar y eliminar una Interfaz de Red.

4.1.1.2. Aprovisionamiento de Software

A partir del requerimiento RF01 - Crear Máquina Virtual, se obtienen máquinas virtuales exclusivamente con el sistema operativo. Posteriormente se define el siguiente requerimiento:

#	Requerimiento	Descripción
RF17	Aprovisionar Software en Máquina Virtual	La solución debe poder instalar y configurar software/aplicaciones en las máquinas virtuales levantadas previamente. A modo de ejemplo, realizar la instalación de una base de datos en una máquina virtual y un balanceador de carga en otra. Los recursos que sean necesarios como entrada para el aprovisionamiento no forman parte de este requerimiento. Por ejemplo, si se utiliza una tecnología que requiera de un archivo de configuración para realizar el aprovisionamiento de software, el usuario será el encargado de proveer dicha configuración y no la solución.

4.1.1.3. Operaciones de Almacenamiento

La solución debe contar con un conjunto de operaciones de almacenamiento, las cuales se describen a continuación.

#	Requerimiento	Descripción
RF18	Crear Volumen	La solución debe poder crear un volumen o disco de almacenamiento en el proveedor.
RF19	Listar Volúmenes	La solución debe poder listar los volúmenes o discos de almacenamiento en el proveedor.
RF20	Eliminar Volumen	La solución debe poder dar de baja un volumen o disco de almacenamiento previamente creado.
RF21	Crear Contenedor de Objetos	La solución debe poder crear un contenedor de objetos.

RF22	Listar Contenedores de Objetos	La solución debe poder listar los contenedores de objetos existentes en el proveedor.
RF23	Eliminar Contenedor de Objetos	La solución debe poder dar de baja un contenedor de objetos previamente creado, eliminando también los objetos dentro del mismo.

4.1.1.4. Comunicación entre redes

#	Requerimiento	Descripción
RF24	Conectar redes	La solución debe brindar una operación para poder comunicar 2 redes de una arquitectura dada. Existen 2 casos a tener en cuenta: ● *Redes de un mismo proveedor*: no es necesario proveer un mecanismo de seguridad confiable, dicha responsabilidad se le atribuye al proveedor. ● *Redes de distintos proveedores*: la solución debe proveer un mecanismo de seguridad confiable, por ejemplo: VPN SSL, IPsec VPN.

4.1.1.5. Resolución de nombres

#	Requerimiento	Descripción
RF25	Resolución de nombres	La solución debe brindar una operación en la cual se puedan resolver nombres de dominio privado (IP) asociados a las máquinas virtuales.

4.1.1.6. Autenticación

#	Requerimiento	Descripción
RF26	Autenticación	La solución debe proveer un único mecanismo de autenticación. El usuario puede tener acceso a más de un proveedor a partir de una única autenticación con la solución.

4.1.1.7. Proveedores de infraestructura en la nube Soportados

#	Requerimiento	Descripción
RF27	Soporte para AWS	La solución debe soportar todas las operaciones especificadas anteriormente en el proveedor AWS.
RF28	Soporte para Azure	La solución debe soportar todas las operaciones especificadas anteriormente en el proveedor Azure.

4.1.2. No Funcionales

4.1.2.1. RNF01 Generalización / Extensibilidad

La solución debe tener la capacidad de integrar nuevos proveedores de forma sencilla. También debe contemplar en su diseño la posibilidad de agregar nuevas operaciones básicas de forma que todos los proveedores las implementen.

4.1.2.2. RNF02 Manual Técnico

Se debe disponer de un manual técnico que muestre en detalle cómo agregar un nuevo proveedor y nuevas operaciones en la solución.

4.1.2.3. RNF03 Personalización / Servicios Transversales

La solución debe brindar los siguientes servicios adicionales, los cuales son considerados de valor agregado:

- *Orquestación de Software*: Centralización del aprovisionamiento de software por medio de un nodo (máquina virtual) que se encarga de aplicar las directivas de aprovisionamiento al resto de los componentes de infraestructura. Ejemplos: Chef [31], Puppet [32] o Salt [33]
- *Monitoreo*: Monitoreo del estado y salud de la infraestructura. Ejemplo: Newrelic [34]
- *Función Lambda* [35]: Despliegue y ejecución de código sin estado, sin la necesidad de la generación de recursos de infraestructura.

4.1.2.4. RNF04 Ejemplos de arquitecturas

La solución deberá contar con un conjunto de especificaciones que permitan generar las arquitecturas de infraestructura más comunes. También es necesario contar con la especificación del caso de éxito.

El fin de este requerimiento es poder lograr un ambiente colaborativo entre los distintos usuarios que utilizan la solución, para de esta forma poder reducir el tiempo de despliegue de distintas arquitecturas y a la vez compartir el conocimiento.

4.1.2.5. RNF05 Manejo de Errores

La solución debe mostrar los errores de una forma amigable hacia el usuario. Para esto se deben enumerar y brindar una breve descripción de los mismos y de esta forma poder identificar los problemas ocurridos.

4.1.2.6. RNF06 Glosario de Errores

Se debe contar con un glosario de errores, en el cual se identifican los errores por un código y su descripción. También se debe, en los casos que sea posible, indicar las posibles soluciones.

4.1.2.7. RNF07 Interfaz de Usuario

La herramienta debe brindar un entorno amigable para el usuario. Con esto se refiere a poder contar con una CLI (Command Line Interface) para poder interactuar por línea de comando. Dicha CLI debe contar con las operaciones de los requerimientos funcionales.

4.2. Alcance del proyecto

El alcance del proyecto se definió a partir de los objetivos y en base a los requerimientos establecidos en conjunto con el cliente.

Para determinar el alcance se definió un caso de éxito como se muestra en la figura 4.1.

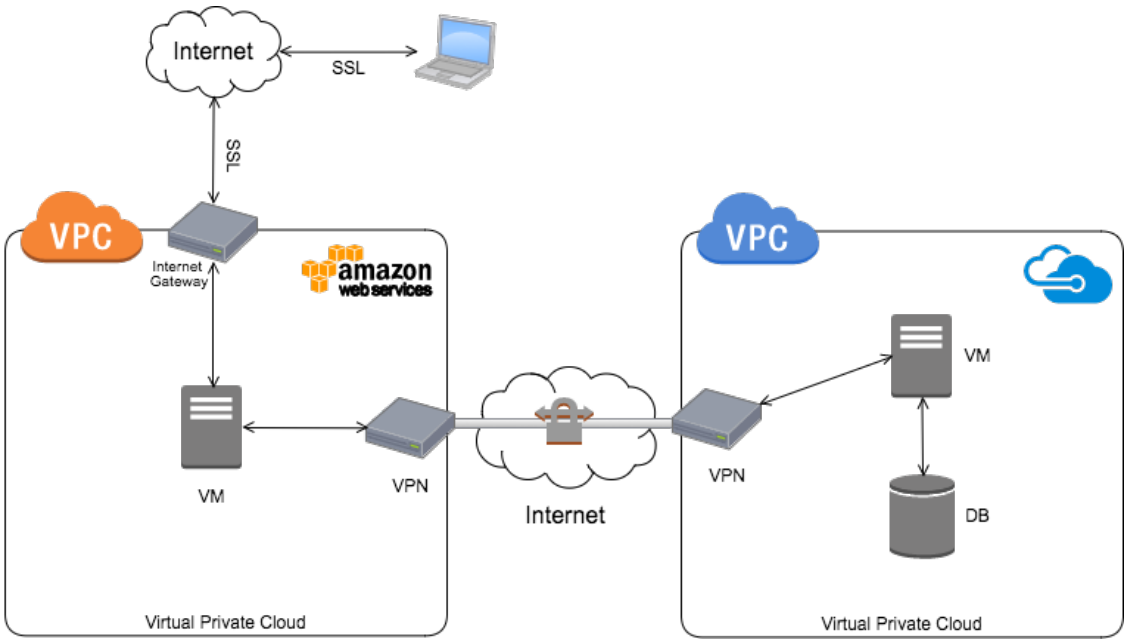


Figura 4.1: Caso de Éxito

En este escenario se muestra una arquitectura en la cual se encuentra una aplicación web que corre en una máquina virtual dentro de una red virtual de AWS. Adicionalmente, es necesario contar con un Gateway dentro de la red para garantizar el acceso público a dicha aplicación. Luego, la aplicación ejecuta consultas a una base de datos para obtener y mostrar determinada información. Debido a que esta base de datos se encuentra desplegada en una máquina virtual dentro de otra red en Azure, para realizar la comunicación con la aplicación es necesario establecer una comunicación VPN entre las 2 redes virtuales. Lo anterior se logra utilizando 2 terminales VPN para establecer dicha comunicación.

La solución deberá desplegar la siguiente infraestructura en los proveedores correspondientes:

- AWS
    - Crear una red (VPC) con una subred
    - Crear servicio de DNS privado para la red
    - Crear una máquina virtual (VM) dentro de la subred
    - Aprovisionar la VM con un servidor de aplicaciones junto con la aplicación web
    - Dar acceso público, a través de internet, a la aplicación por medio de un Gateway

- Azure
    - Crear una red (VPC) con una subred
    - Crear servicio de DNS privado para la red
    - Crear una VM dentro de la subred
    - Aprovisionar la VM con un servidor de Base de Datos
    - Crear una terminal VPN dentro de la red

En este caso de éxito se contemplan la mayoría de los requerimientos mencionados anteriormente.

A continuación, se muestra el alcance acordado del proyecto.

Dentro del Alcance			
#	Requerimiento	#	Requerimiento
RF01	Crear Máquina Virtual	RF14	Crear Interfaz de Red
RF02	Eliminar Máquina Virtual	RF15	Listar Interfaz de Red
RF03	Iniciar Máquina Virtual	RF16	Eliminar Interfaz de Red
RF04	Detener Máquina Virtual	RF17	Aprovisionar Software en Máquina Virtual
RF05	Reiniciar Máquina Virtual	RF24	Conectar redes
RF06	Listar Imágenes	RF25	Resolución de nombres
RF07	Listar Tamaños	RF26	Autenticación
RF08	Crear Red Virtual	RF27	Soporte para AWS
RF09	Listar Redes Virtuales	RF28	Soporte para Azure
RF10	Eliminar Red Virtual	RNF01	Generalización / Extensibilidad
RF11	Crear Subred	RNF02	Manual técnico

RF12	Listar Subredes	RNF04	Ejemplos de arquitecturas
RF13	Eliminar Subred		

Fuera del Alcance			
#	Requerimiento	#	Requerimiento
RF18	Crear Volumen	RF23	Eliminar Contenedor
RF19	Listar Volúmenes	RNF03	Personalización / Servicios transversales
RF20	Eliminar Volumen	RNF05	Manejo de errores
RF21	Crear Contenedor	RNF06	Glosario de errores
RF22	Listar Contenedores	RNF07	Interfaz de usuario

5. Arquitectura y Diseño de la solución

El objetivo de este capítulo es presentar el diseño de la solución propuesta al problema planteado, según el alcance acordado.

El capítulo se inicia dando una descripción general de los componentes que forman parte de la solución.

Luego se describe el diseño de la Librería Core, componente más importante de la solución. Dicho componente es el encargado de la integración de los restantes componentes.

Finalmente se presenta el despliegue de la solución con sus nodos correspondientes.

5.1. Arquitectura

La solución propuesta, denominada Bas7ion, consiste en un conjunto de componentes de software y librerías desarrolladas por terceros que en su conjunto (como se muestra en la figura 5.1), permiten cumplir con el alcance acordado.

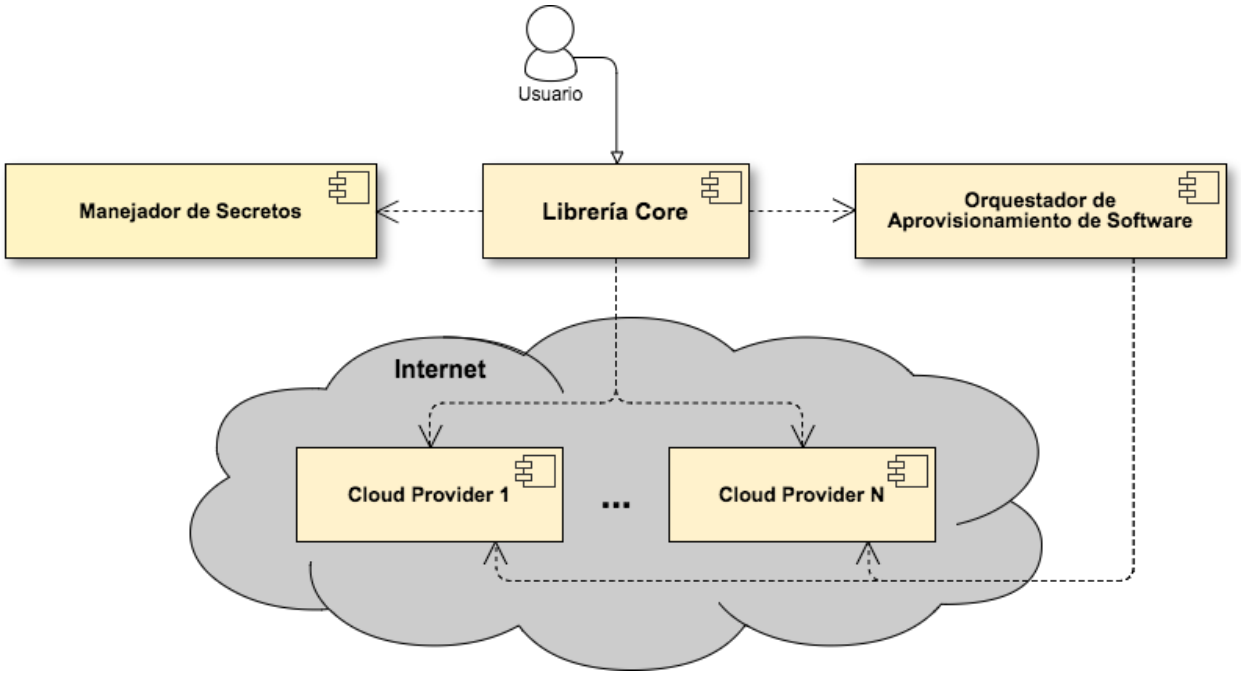


Figura 5.1: Diagrama de Componentes

5.1.1. Definición de componentes

A continuación, se describe en términos generales cada uno de los componentes.

5.1.1.1. Librería Core

Es el componente con el que interactúa el usuario de Bas7ion, en el cual se define una interfaz única de conexión con los proveedores de infraestructura en la nube por medio de la abstracción y normalización de los servicios que ofrecen. Presenta un modelado genérico y estandarizado de los recursos de infraestructura que resuelven los requerimientos funcionales.

5.1.1.2. Manejador de Secretos

Para poder cumplir con el requerimiento de autenticación (RF26), se define el componente Manejador de Secretos para el almacenamiento de las credenciales y configuraciones de los distintos proveedores de infraestructura en la nube. El mismo es responsable de almacenar dicha información de manera segura.

5.1.1.3. Orquestador de Aprovisionamiento de Software

Es el componente encargado de instalar y configurar software en máquinas virtuales, como por ejemplo frameworks, librerías, base de datos y servidor de aplicaciones. Para realizar dichas tareas se utiliza el protocolo ssh.

5.1.1.4. Cloud Provider

Este componente es el encargado de ofrecer los servicios de infraestructura para cada uno de los proveedores en la nube. Cada uno de éstos tiene su propio modelo de virtualización de infraestructura, como así también su propio mecanismo de autenticación. Dichos servicios son ofrecidos por cada uno de los proveedores a través de una API.

5.1.2. Interacción entre componentes

A continuación, se presentan las interacciones entre los componentes de la solución y la Librería Core:

- *Manejador de Secretos*: interactúa con dicho componente para obtener las credenciales y configuraciones de acceso a los respectivos proveedores.
- *Orquestador de Aprovisionamiento de Software*: se comunica con este componente para aprovisionar software en las máquinas virtuales de los proveedores.

- *Cloud Provider*: interactúa con este componente para realizar el despliegue y configuración de los recursos de infraestructura que ofrecen los proveedores.

En la figura 5.2 se muestra la creación de una máquina virtual, junto con la interacción de los componentes.

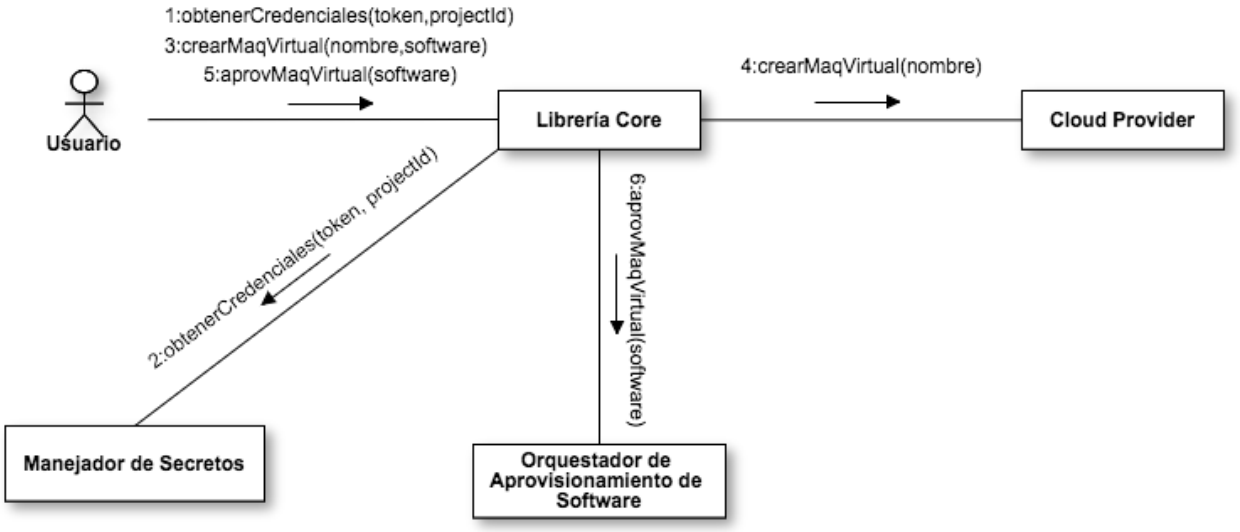


Figura 5.2: Diagrama de comunicación entre componentes - crear máquina virtual

Inicialmente el usuario obtiene las credenciales del proveedor. Esta tarea se lleva a cabo a partir de la interacción entre la Librería Core y el Manejador de Secretos. Luego el usuario crea una máquina virtual en dicho proveedor, lo cual es realizado por el componente Cloud Provider, a través de la Librería Core. Finalmente, dicha máquina virtual es aprovisionada por el Orquestador de Aprovisionamiento de Software. En este ejemplo se pueden apreciar todas las interacciones entre los componentes de la solución.

5.2. Diseño del componente Librería Core

Para entender la Librería Core es importante describir sus 2 controladores que la componen: Auth Driver y Cloud Provider Driver.

5.2.1. Auth Driver

Es el controlador encargado de realizar la autenticación y comunicación con el manejador de secretos, con el fin de obtener las credenciales y configuraciones de los proveedores en la nube.

Este controlador fue diseñado de forma tal que puede ser extendido a más de un tipo de manejador de secretos como se muestra en la figura 5.3. Se deben tener en cuenta las siguientes características que debe cumplir el manejador de secretos:

- *Autenticación*: debe proveer un mecanismo de autenticación (token).
- *Identificación de proyectos*: es necesario identificar cada proyecto (conjunto de credenciales y configuraciones) en el manejador de secretos.

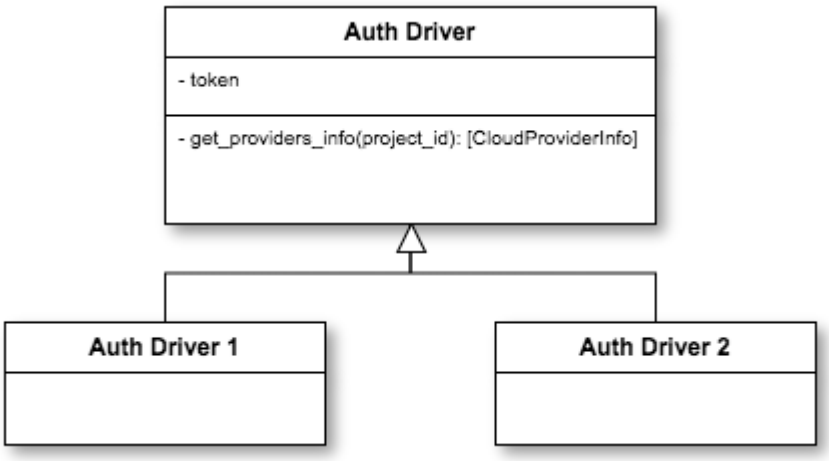


Figura 5.3: Diagrama de clases Auth Driver

Para obtener el conjunto de credenciales y configuraciones de cada proveedor en la nube, se deben extender las entidades correspondientes como se muestra en la figura 5.4.

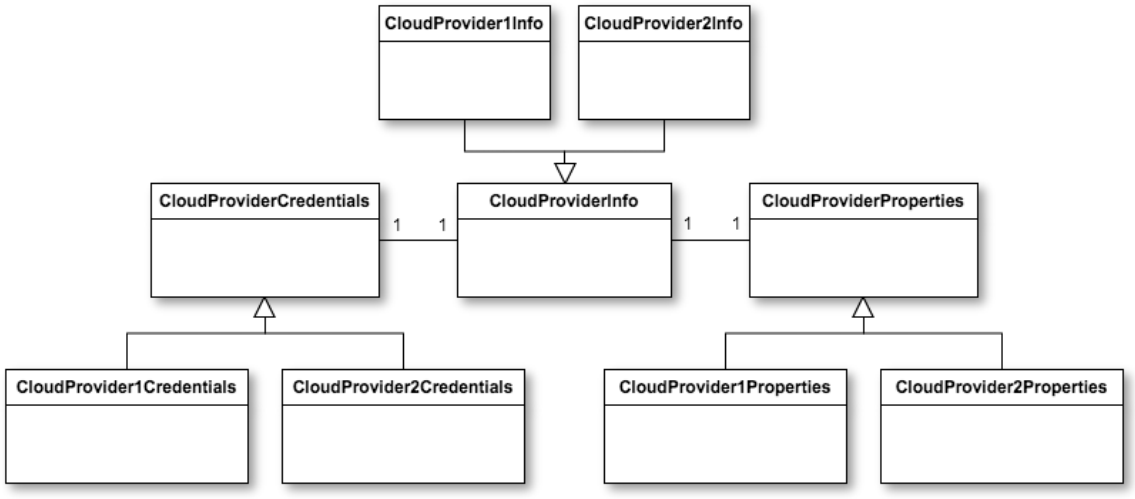


Figura 5.4: Diagrama de clases entidades Auth Driver

En CloudProviderInfo se encuentran las credenciales (CloudProviderCredentials) y configuraciones (CloudProviderProperties) del proveedor en la nube.

Dentro de las credenciales se deben guardar los atributos necesarios para realizar la autenticación con el proveedor en la nube. Por otra parte, en las propiedades se almacenan los datos necesarios para hacer uso de los servicios del proveedor.

5.2.2. Cloud Provider Driver

Este controlador abstrae al usuario de las implementaciones específicas para cada proveedor.

Con la implementación del patrón factory se instancian los controladores de los diferentes proveedores en la nube, tomando en cuenta las credenciales y configuraciones que se obtuvieron del AuthDriver. Este diseño se puede observar en la figura 5.5.

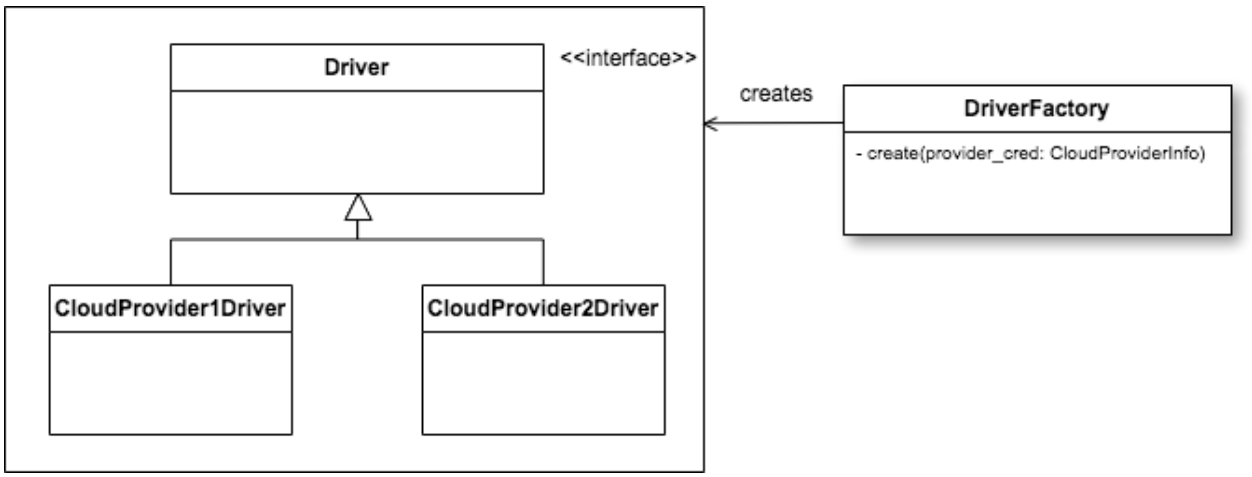


Figura 5.5: Diagrama de clases del patrón factory de Cloud Provider Driver

Al momento de obtener el controlador específico se obtienen las instancias de los servicios específicos del proveedor en la nube, denominados Cloud Services, como se muestra en la figura 5.6.

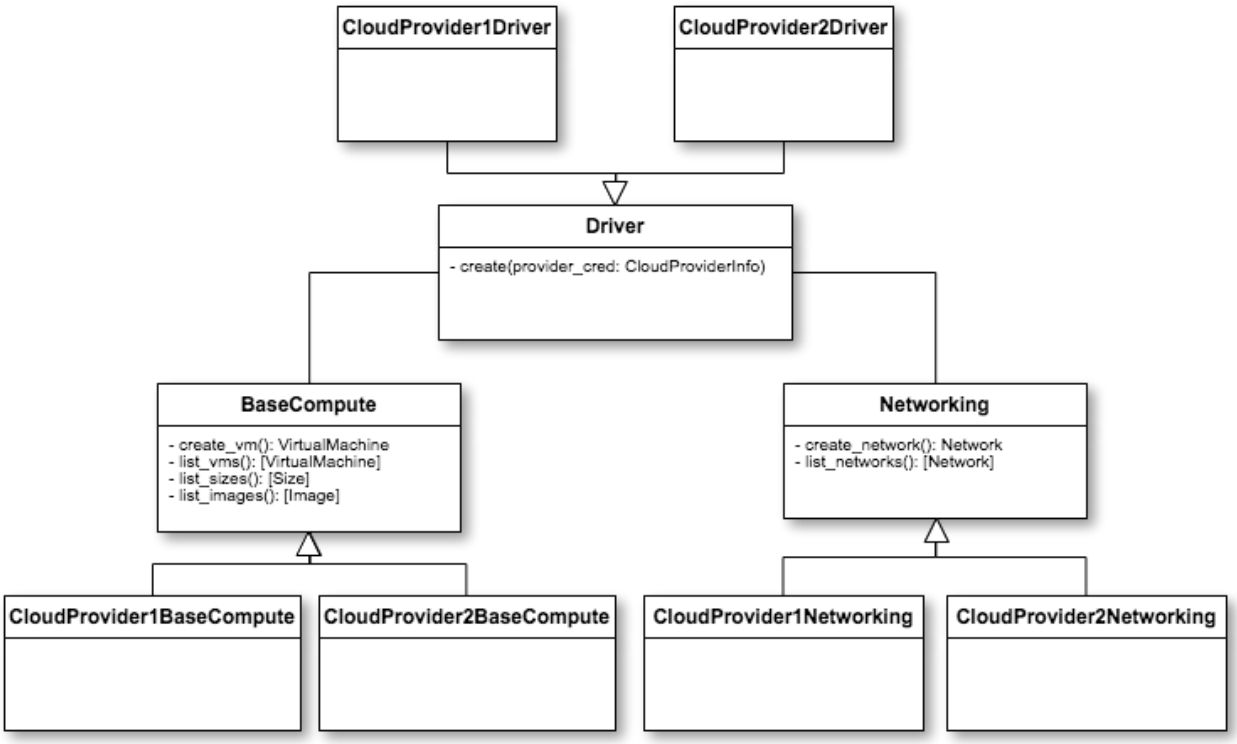


Figura 5.6: Diagrama de clases de los Cloud Services

Los Cloud Services contienen operaciones asociadas a un área específica de infraestructura, las cuales son: capacidad de cómputo (BaseCompute) y redes (Networking). Dichos servicios son los encargados de la creación de las entidades de infraestructura, denominadas Cloud Entities, como se muestran en la figura 5.7.

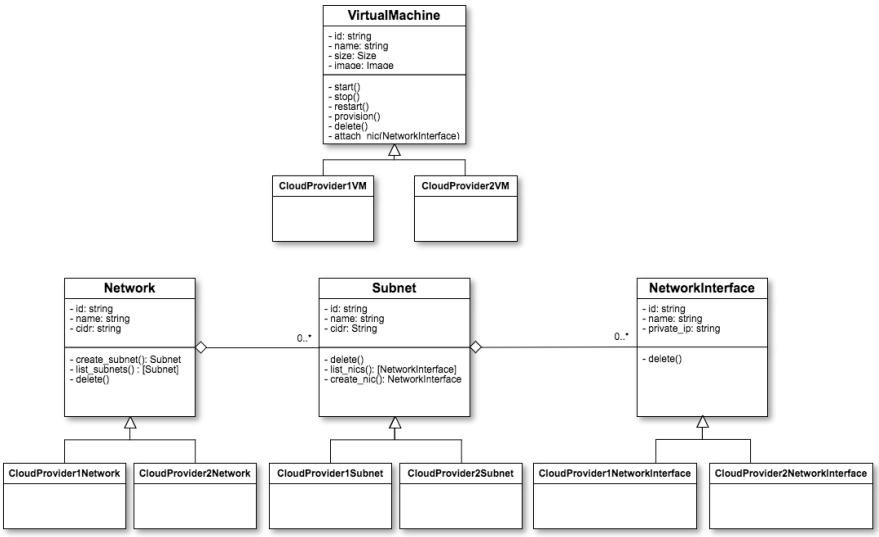


Figura 5.7: Diagrama de clases de los Cloud Entities

Estas entidades se definieron a partir de los requerimientos funcionales que se encuentran en el grupo Operaciones Básicas (RF01 a RF16).

5.3. Despliegue de la solución

En la figura 5.8 se muestra la distribución de los componentes en los distintos nodos (devices) que son necesarios para el despliegue de la solución.

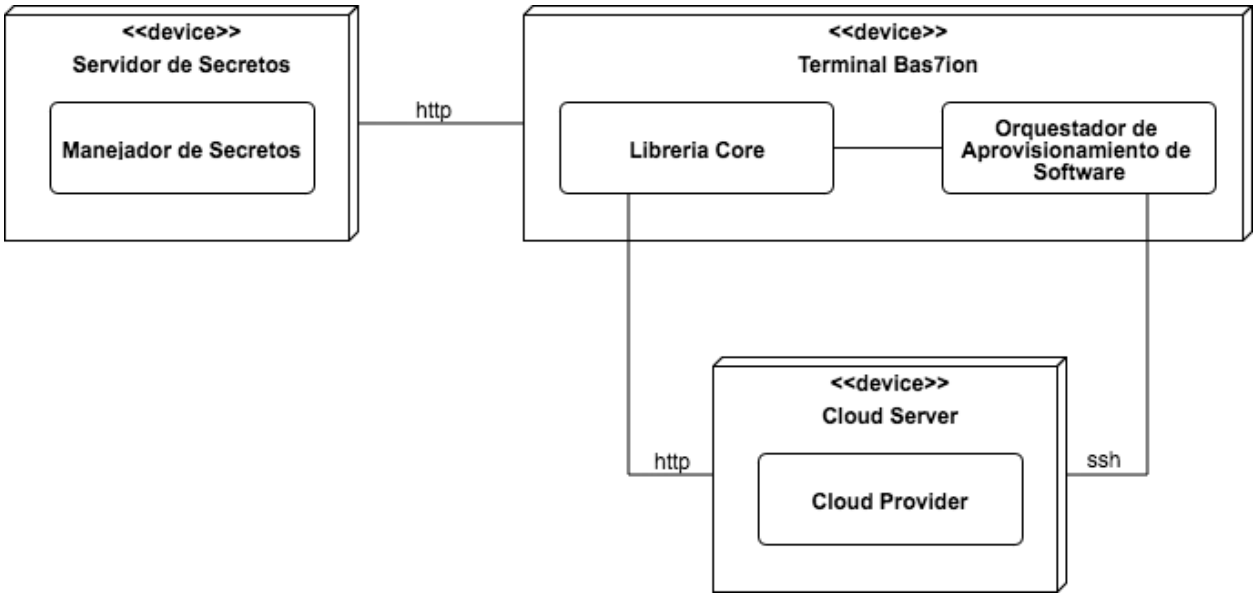


Figura 5.8: Diagrama de despliegue

En primer lugar, se definió el nodo Terminal Bas7ion, el cual incluye la Librería Core y el Orquestador de Aprovisionamiento de Software. La comunicación entre estos componentes se realiza a través de línea de comando.

Luego se encuentra el nodo Servidor de Secretos, el cual contiene al componente Manejador de Secretos. Este se comunica vía http con la Librería Core.

Por último, se dispone de un nodo Cloud Server por cada uno de los proveedores en la nube que componen la solución. La Librería Core se comunica vía http con la API del proveedor para generar los recursos de infraestructura. Por otra parte, el Orquestador de Aprovisionamiento de Software se comunica vía ssh con las máquinas virtuales generadas a través de la interacción mencionada anteriormente.

6. Implementación

En este capítulo se presentan las tecnologías utilizadas y su rol en la implementación de los componentes más relevantes de la solución. Luego se describen las principales decisiones técnicas tomadas a la hora de implementar las funcionalidades críticas de la Librería Core, las cuales cumplieron un rol importante en la definición de la arquitectura. Además, se comentan las dificultades y limitaciones encontradas a lo largo del proyecto y por último se presentan las pruebas realizadas en la etapa de testing.

6.1. Tecnologías utilizadas

En la figura 6.1 se muestra el diagrama de componentes presentado en el capítulo anterior, con las respectivas tecnologías utilizadas.

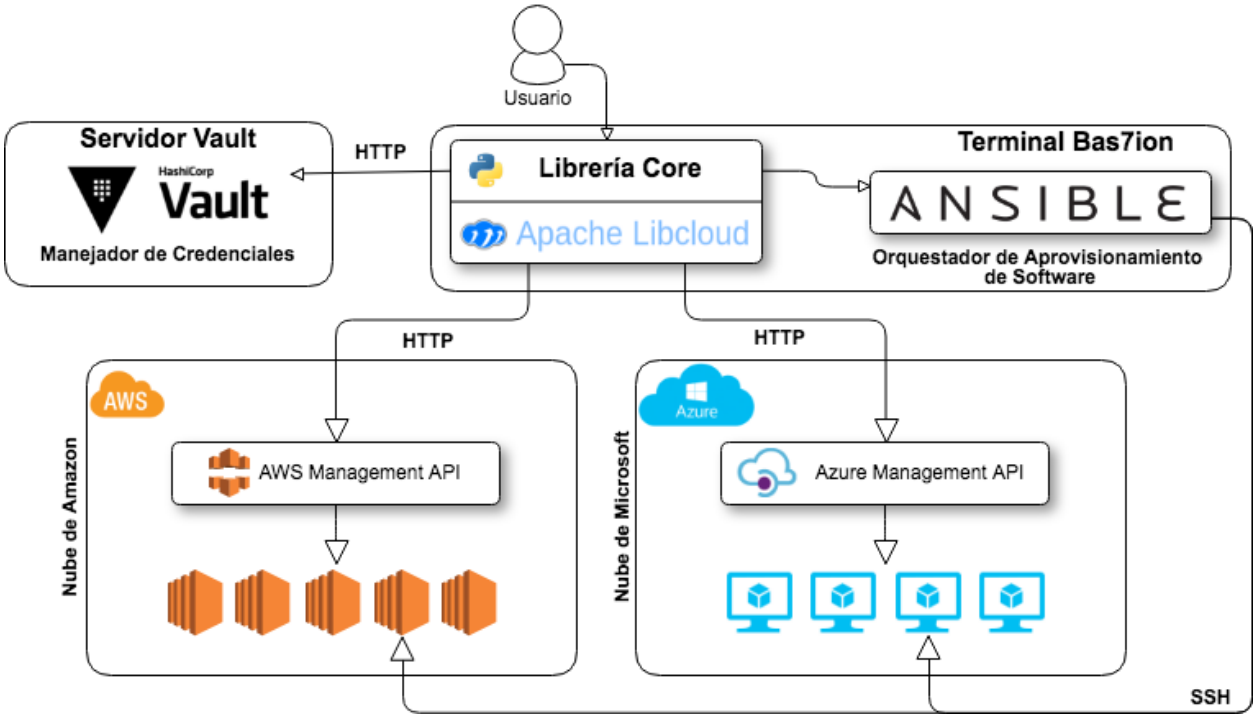


Figura 6.1: Diagrama de componentes con tecnologías utilizadas

6.1.1. Python

Python [36] es un lenguaje de programación interpretado, orientado a objetos, que cuenta con un amplio soporte de librerías y una gran comunidad de desarrolladores.

Teniendo en cuenta las características del lenguaje y que una de las librerías más importantes (Apache Libcloud) a integrar está implementada en Python, se optó por

dicho lenguaje de programación. Adicionalmente, la elección fue discutida y validada con el cliente previo al comienzo de la implementación.

6.1.2. Vault

Vault [37] es un manejador de secretos que permite a los usuarios almacenar de forma segura cualquier tipo de información sensible (secreto), por ejemplo: passwords, claves, tokens y/o certificados. Estos secretos se agrupan dentro de estructuras llamadas backends y los usuarios pueden consultarlos a través de una API autenticada, segura y unificada.

Las propiedades mencionadas anteriormente cumplen con las consideraciones de diseño del componente Manejador de Secretos. Adicionalmente, Vault fue sugerido por el cliente del proyecto dada su experiencia con el mismo. Por estas razones se decidió utilizar Vault en el componente Manejador de Secretos.

6.1.3. Apache Libcloud

Apache Libcloud [30] es una librería open-source desarrollada en Python que permite interactuar con algunos de los proveedores en la nube más populares a través de una API unificada. Dicha librería abstrae la comunicación directa con las API de cada uno de los proveedores en la nube, modelando para cada uno de ellos sus servicios y entidades de infraestructura.

Tomando en cuenta que Libcloud provee comunicación con proveedores en la nube, en particular AWS y Azure, se decidió utilizar dicha librería con el fin de abstraerse de la implementación de la comunicación con cada uno de los proveedores.

6.1.4. Ansible

Ansible [38] es un software que automatiza el aprovisionamiento de software, la gestión de configuraciones y el despliegue de aplicaciones. Se considera una herramienta de orquestación muy útil para los administradores de sistema. Es decir, permite gestionar servidores, configuraciones y aplicaciones de forma sencilla, robusta y paralela.

Ansible gestiona desde una terminal, a través de SSH, los diferentes nodos remotos, los cuales requieren Python para ser gestionados. Utiliza un archivo en formato yaml, denominado playbook o receta, para describir las tareas y configuraciones a realizar en los diferentes nodos.

Debido a la complejidad en el uso de los orquestadores existentes, se decidió utilizar Ansible dado que el cliente posee conocimientos sobre dicho orquestador, de esta forma minimizar la curva de aprendizaje.

6.2. Herramientas de desarrollo

A continuación, se describen las herramientas de desarrollo que se utilizaron a lo largo del proyecto.

Se utilizó Bitbucket [39] para alojar el código fuente de la solución en un repositorio Git privado. Adicionalmente, se siguió el flujo de Git Flow [40] para implementar las distintas funcionalidades.

Debido a que el lenguaje elegido fue Python, se optó por utilizar PyCharm [41] como entorno de desarrollo. Dicha herramienta brinda un entorno amigable para desarrollar en el lenguaje mencionado anteriormente.

6.3. Aspectos de implementación

A continuación, se describen las principales decisiones técnicas tomadas a la hora de implementar las funcionalidades críticas, las cuales tienen un rol importante en la definición de la arquitectura.

6.3.1. Manejador de Secretos

Como mecanismo de autenticación del Manejador de Secretos se utilizó la autenticación por token (vault_token) provista por Vault. Cada token identifica a un usuario de Bas7ion, el cual brinda acceso a un conjunto de proyectos.

Para obtener el conjunto de credenciales asociadas a un proyecto es necesario conocer el identificador de dicho proyecto (project_id). Este identificador está compuesto por una ruta (directorio). Luego, para almacenar la información de un acceso a un proveedor, inicialmente se debe crear un sub-directorio (provider_access_id) dentro del proyecto al cual pertenece y así guardar dicha información en el mismo.

Una vez configurados los proyectos, se obtienen las credenciales del proyecto a través del token del usuario y la ruta al directorio del proyecto. Esto se realiza listando los sub-directorios dentro del proyecto. Para cada sub-directorio se obtienen las credenciales y configuraciones, generando las entidades correspondientes a partir de los datos obtenidos en formato json. En la figura 6.2 se muestra la interacción entre la Librería Core y Vault para la obtención de las credenciales y configuraciones correspondientes a un sub-directorio, el cual representa un acceso a un proveedor.

Figura 6.2: Ejemplo de interacción Librería Core-Vault

6.3.2. Comunicación con los Proveedores en la Nube

Para establecer la comunicación con los respectivos proveedores de la nube, se agregó la librería Apache Libcloud como una dependencia del componente Librería Core. La misma es utilizada por las implementaciones de los controladores específicos de cada proveedor (Cloud Provider Driver).

Existen operaciones necesarias para la implementación de la solución, las cuales no se encuentran en Libcloud pero si están disponibles en las APIs de los proveedores. Por este motivo se tomó la decisión de implementar una extensión de dicha librería. Se optó por utilizar Libcloud como una caja negra y crear un módulo entre Cloud Provider Driver y Libcloud como se muestra en la figura 6.3.

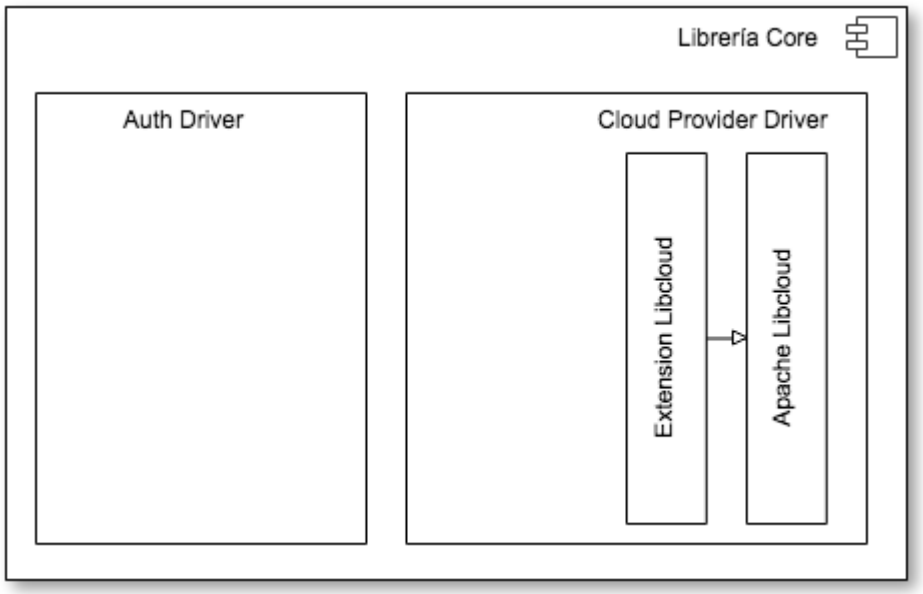


Figura 6.3: Extensión Libcloud

En la implementación de dicho módulo se extiende la interfaz provista por Libcloud, agregando las nuevas operaciones y sobrecargando otras que no contemplan ciertos escenarios. El detalle de dicha extensión se puede consultar en: “ANEXO A - Extensión de Libcloud”.

6.3.3. Aprovisionamiento de Software

El aprovisionamiento de software de las máquinas virtuales se realiza a través de Ansible. Se utilizan recetas de Ansible que son provistas y configuradas por el usuario de Bas7ion. Cada ejecución del aprovisionamiento de software aplica una receta en una única máquina virtual.

Para integrar Ansible con la Librería Core se ejecuta un proceso de línea de comando con la directiva ansible-playbook y los siguientes parámetros:

- IP pública de la máquina virtual.
- Ruta a la receta yaml.
- Clave privada asociada a la máquina virtual (conexión SSH).

En la Figura 6.4 se pueden observar los pasos en la ejecución de una receta de Ansible en Bas7ion:

1. El usuario de Bas7ion implementa la receta que configura la máquina virtual (VM) a ser levantada.
2. La Librería Core recibe como entrada la receta mencionada anteriormente.
3. Luego de haber dado de alta la máquina virtual, la Librería Core ejecuta un proceso de línea de comando.
4. El motor de Ansible se encarga de ejecutar las directivas de aprovisionamiento en la máquina virtual especificada.

Figura 6.4: Pasos en la ejecución de una receta

6.3.4. VPN

Actualmente no hay una forma genérica de interconectar redes en la nube, por este motivo el cliente considera importante que Bas7ion brinde dicha funcionalidad. Para ello fue necesario crear la capacidad de generar terminales VPN y así poder comunicar

redes de diferentes proveedores de infraestructura en la nube, sin importar la región en la que se encuentren.

Bas7ion ofrece la operación de interconectar 2 redes, independientemente de a que proveedores de infraestructura en la nube pertenecen dichas redes. Para ellos se realizó una implementación de VPN en cada uno de los proveedores de infraestructura en la nube disponibles (AWS y Azure).

Si bien ambos proveedores cuentan con la capacidad de crear conexiones VPN, en AWS solo existe la posibilidad de crear clientes VPN para ser conectados a una red de una infraestructura on-premise. En cambio, en Azure se puede conectar dos redes virtuales a través de un conector VPN a través de IPsec.

A continuación, se detalla la manera en que se implementó dicha funcionalidad en cada uno de los proveedores en cuestión.

6.3.4.1. AWS

Se creó una máquina virtual, la cual se aprovisionó utilizando Bas7ion, en la que se instaló StrongSwan [42]. Este software es el encargado de actuar como terminal VPN y que toda la comunicación desde la red a la cual pertenece dicha máquina virtual, que se dirige a la otra red, pase a través de ella.

6.3.4.2. Azure

En este proveedor existe el concepto de Virtual Network Gateway, que representa el terminal VPN. Para lograr dar soporte desde Bas7ion a esta capacidad se extendió Libcloud.

A continuación, se muestra la figura 6.5 en la cual se muestran los componentes de VPN mencionados anteriormente que utiliza Bas7ion.

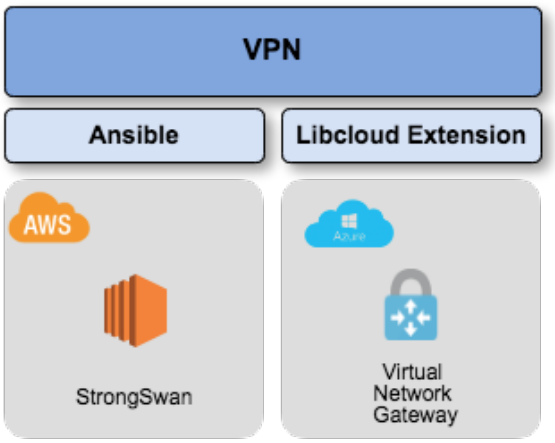


Figura 6.5: Componentes de VPN en Bas7ion

6.3.5. DNS

Otro de los aspectos de implementación críticos de Bas7ion es el servicio de DNS. Bas7ion ofrece las operaciones necesarias para la configuración de dicho servicio, como por ejemplo crear un registro a partir de un hostname, zona, tipo de registro e ip a vincular. Actualmente solo se ofrece la configuración del registro de tipo A, o sea nombrar ipv4 con hostname.

Para implementar las operaciones necesarias mencionadas anteriormente se realizó un estudio de la situación para identificar la forma de ofrecer dicho servicio por parte de cada proveedor.

6.3.5.1. AWS

Este proveedor ofrece un servicio al que denomina Route53, en el cual entre otras cosas ofrece servicios para DNS privado y público. Por esta razón Bas7ion se comunica con Libcloud para obtener las operaciones. Debido a que Libcloud únicamente ofrece las operaciones de DNS público, se extendió dicha librería para soportar las operaciones de DNS privado. Esto se realizó a partir de la documentación de la API ofrecida por Route53. De esta forma Bas7ion ofrece los servicios mencionados anteriormente.

6.3.5.2. Azure

La situación para este proveedor es diferente debido a que actualmente solo ofrece servicios para DNS público y no lo ofrece para DNS privado, a pesar de que en un futuro cercano probablemente sea implementado. Por este motivo, se requiere resolver la problemática de la mejor forma posible. De aquí surge la solución de utilizar los componentes existentes, en este caso la funcionalidad de aprovisionamiento de software. A partir de dicha funcionalidad que ofrece Bas7ion, se configura un servidor local de DNS, para la solución actual se utiliza un servidor BIND [43]. A través del aprovisionamiento se realiza la instalación y configuración dicho servidor de DNS. Cada vez que se agrega un nuevo registro de DNS, se vuelve a realizar el aprovisionamiento, pero solo se actualizan los hostnames e ips dentro del servidor.

A continuación, se muestra la figura 6.6 en la cual se muestran los componentes de DNS mencionados anteriormente que utiliza Bas7ion.

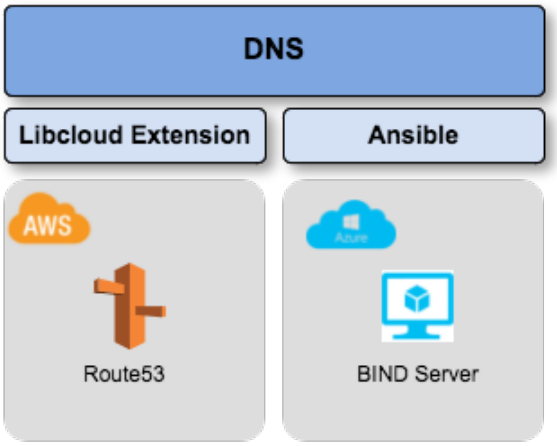


Figura 6.6: Componentes de DNS en Bas7ion

La implementación de DNS y VPN son claros ejemplos donde ambos proveedores no ofrecen dichos servicios, pero a través de Bas7ion es posible dar solución a estos problemas.

6.4. Testing

Para describir las pruebas realizadas se consideraron los diferentes tipos de pruebas existentes: pruebas de interfaz de usuario, pruebas de integración y pruebas unitarias [44].

Considerando las necesidades del cliente y el tiempo disponible para el desarrollo de las pruebas, se realizó la planificación de la etapa de testing. Se priorizó realizar únicamente pruebas de integración, ya que a partir de estas se pueden validar las funcionalidades en su conjunto.

Dado que el usuario de Bas7ion interactúa con una librería no se realizaron pruebas de interfaz de usuario. Por otra parte, tampoco se realizaron pruebas unitarias debido a que la validación de dichas pruebas requiere de gran esfuerzo para simular el comportamiento de los recursos generados por los proveedores. A modo de ejemplo, comprobar la conectividad entre las subredes de una red.

Con respecto a las pruebas de integración, se ejecutaron casos de prueba que cubren los requerimientos funcionales del alcance. La validación de las pruebas se llevó a cabo manualmente, corroborando en los portales de los proveedores que los recursos generados a partir de las pruebas se desplegaran satisfactoriamente. Dicha validación incluye, además de la generación de los recursos, la configuración y conectividad entre ellos.

A medida que finalizaba cada funcionalidad con su respectiva prueba de integración, se realizaron pruebas de regresión utilizando las pruebas de integración de las funcionalidades implementadas anteriormente.

Debido a las prioridades mencionadas anteriormente y el tiempo disponible para la realización de esta etapa, no se logró automatizar las pruebas ni obtener métricas de las mismas.

En “ANEXO D - Pruebas Realizadas” se encuentra el detalle de las pruebas realizadas.

6.5. Dificultades y limitaciones encontradas

En esta sección se describen las dificultades y limitaciones que se presentaron a lo largo de la implementación de la solución y realización de las pruebas.

6.5.1. Desconocimiento del lenguaje Python

Debido a la inexperiencia del equipo en el lenguaje utilizado, se presentó una limitación, la cual fue superada a partir de la curva de aprendizaje de dicho lenguaje.

6.5.2. Generación de credenciales de acceso a Azure

Se encontraron inconvenientes a la hora de generar las credenciales en Azure y los permisos necesarios para utilizar la API correspondiente. Esto se debió a la poca claridad de la documentación del proveedor, por lo que fue necesario recurrir a fuentes de información alternativas [45].

6.5.3. Tiempos excesivos en la ejecución de pruebas

Con respecto a la generación de los recursos se presentaron dificultades para poder realizar las pruebas, debido al tiempo de despliegue de dichos recursos. El caso más notorio se da en la generación de la comunicación entre redes (VPN), dónde ciertos elementos de infraestructura de Azure demoran aproximadamente 45 minutos en estar disponibles.

6.5.4. Diferencias en la generación de infraestructura entre la API y el portal web del proveedor

Para mitigar los riesgos en la implementación se realizaron pruebas de concepto desde los portales de los proveedores en la nube. A la hora de la implementación, la infraestructura generada a través de las APIs de los proveedores tenia diferencias, en

ciertos casos, con las pruebas de concepto realizadas. Estas diferencias se observaron en el comportamiento y configuración de los recursos generados.

Uno de los casos más relevantes en donde se presentó dicha dificultad fue en la implementación de la conexión VPN.

6.5.5. Errores en funcionalidades de Apache Libcloud

A la hora de realizar las pruebas, se encontraron errores en algunas operaciones ofrecidas por Libcloud. Esto requirió un análisis profundo para identificar el problema y posteriormente solucionarlo en la extensión de dicha librería.

7. Gestión del proyecto

En este capítulo se presentan los distintos aspectos del proceso utilizado para llevar adelante la gestión del proyecto. Inicialmente se describe la organización del proyecto, luego se presenta la planificación inicial y ejecución del mismo. Finalmente se presentan las desviaciones entre la planificación inicial y la ejecución.

7.1. Organización

A la hora de comenzar con el proyecto, se plantearon reuniones cada dos semanas con los tutores, y de esta forma poder definir el trabajo a realizar en periodos cortos. Inicialmente se dividían las tareas entre el equipo, y posteriormente integrar el trabajo realizado previo a cada reunión.

Por otro lado, se definieron reuniones con el cliente cada vez que existían conceptos o requerimientos que debían ser aclarados.

A partir de lo mencionado anteriormente, se logró hacer un seguimiento más detallado del avance del proyecto y manejar los cambios de los requerimientos de mejor manera.

Si bien la dinámica de trabajo fue la deseada a lo largo del proyecto, existieron momentos donde no se pudo mantener, debido a diferentes factores que afectaron a todos los integrantes del proyecto: licencias, aumento en la carga de trabajo por responsabilidades laborales, tiempo dedicado a otras materias.

7.2. Planificación inicial

Al inicio del proyecto se creó una hoja de ruta, con el fin de elaborar una planificación utilizando una metodología de cascada. En dicha hoja de ruta se agregaron las funcionalidades que se relevaron en la etapa de análisis, además de las tareas necesarias para llevar a cabo el proyecto. Las mismas fueron: investigación de herramientas y tecnologías, diseño de la solución, implementación de la herramienta, pruebas a realizar y documentación del proyecto.

A partir de esta hoja de ruta, se llegó a la planificación inicial que se muestra en la figura 7.1.

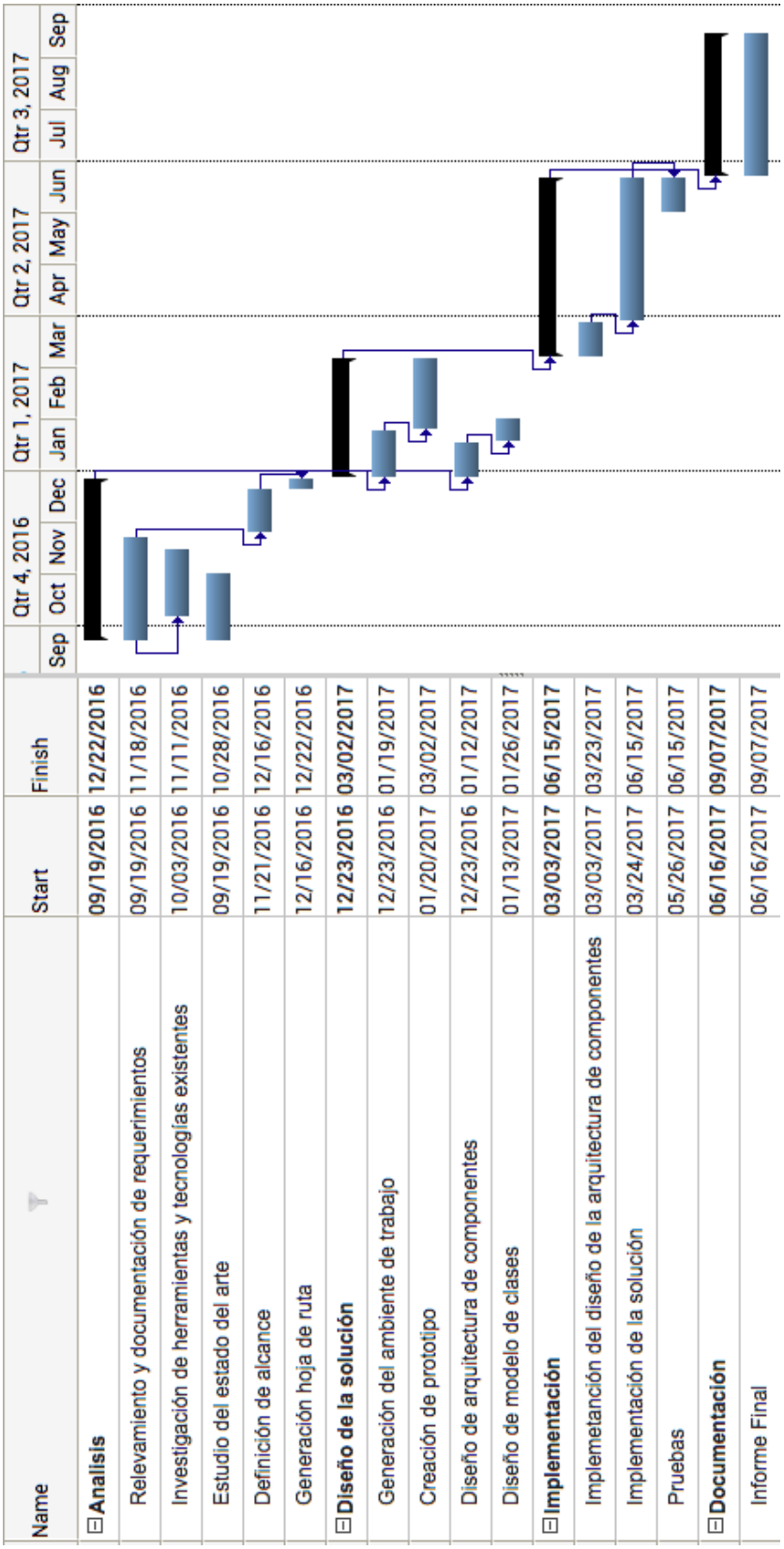


Figura 7.1: Planificación inicial del proyecto

7.3. Ejecución del plan del proyecto

Luego de determinado el plan inicial, se dio comienzo a la ejecución del mismo. En esta sección se describe cómo se llevaron a cabo cada una de las etapas del proyecto.

7.3.1. Análisis de requerimientos e investigación de tecnologías

Inicialmente se identificaron y estudiaron los conceptos más relevantes relacionados con la problemática a resolver. Por otro lado, se investigaron las herramientas existentes en el mercado vinculadas a la temática del proyecto.

En paralelo a la investigación, se realizaron reuniones con el cliente para relevar los requerimientos. A su vez, en dichas reuniones se validaron las especificaciones de los requerimientos obtenidos anteriormente. A partir de esto se creó un documento de requerimientos sobre el cual se realizaron iteraciones hasta llegar a especificar las funcionalidades con un nivel de detalle suficiente como para planificar la totalidad del proyecto.

7.3.2. Diseño de la Solución

Una vez finalizada la etapa anterior, se comenzó con el diseño de la solución.

Inicialmente se diseñó la arquitectura de componentes para poder cumplir de manera efectiva con la extensibilidad y generalización de la solución. Para lograr esto se hizo énfasis en definir el diagrama de clases, el cual fue la base para hacer extensible la solución y de esta forma poder soportar nuevos proveedores de infraestructura en la nube.

En paralelo al diseño del diagrama de clases se realizaron pruebas de concepto con el objetivo de validar las tecnologías investigadas, y de esta forma se mitigaron los riesgos técnicos más relevantes. Durante la implementación de dichas pruebas se identificaron dificultades técnicas que impactaron la planificación inicial del proyecto.

7.3.3. Implementación y pruebas del sistema

A partir de los requerimientos obtenidos y el diseño realizado se dio comienzo a la implementación de la solución.

En las reuniones con el cliente donde se presentaron los avances del desarrollo, se solicitaron cambios de prioridad en los requerimientos, además de cambios en los mismos. Debido a esto en conjunto con el cliente y los tutores se tomó la decisión de cambiar a una metodología más ágil, que le permitiera al cliente modificar y priorizar los requerimientos según sus necesidades.

Para ello se decidió tomar de referencia la metodología Scrum [46] y utilizar sus artefactos y eventos más relevantes (backlog, sprints, planificación y revisión de sprint). Como herramienta de gestión se utilizó Trello [47], en la cual se agregaron todas las funcionalidades a desarrollar en el backlog y en cada planificación de sprint se determinó en conjunto con el cliente que funcionalidades implementar. Como resultado se obtuvo una planificación de 8 sprint. El número de sprints fue determinado a partir de una negociación con el cliente y además tomando en cuenta la duración del proyecto. Los sprints incluyeron tareas de investigación, diseño e implementación, como se presenta a continuación en la figura 7.2.

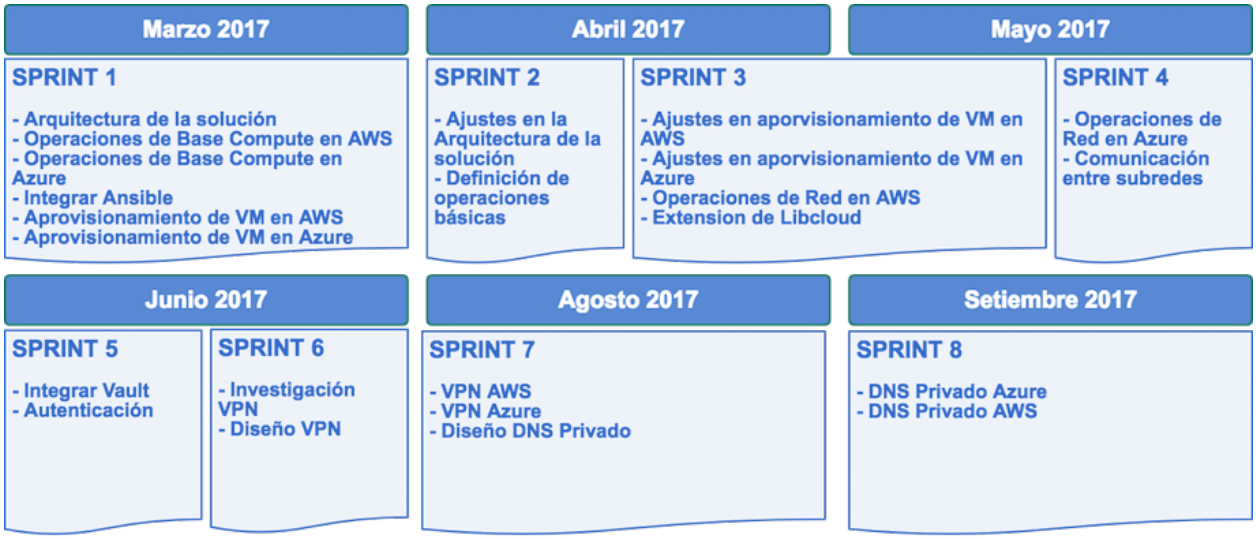


Figura 7.2: Sprints - Metodología Scrum

Como se observa en la imagen, el mes de Julio no forma parte de los sprint ya que en dicho mes no se realizó ninguna tarea del proyecto debido a exámenes.

Por otra parte, en los sprint 1 y 3 existieron desviaciones de 2 semanas debido a errores en la estimación de las tareas. Con respecto a los sprint 7 y 8 la desviación (1 semana) estuvo vinculada a la complejidad de las tareas.

7.3.4. Documentación

Esta etapa se realizó al finalizar todas las etapas anteriores, ya que en ese momento se contó con la información necesaria para redactar el informe final del proyecto. De todas formas, a lo largo del proyecto se generaron documentos que fueron utilizados para la generación del informe. Este fue creado en Google Docs, para luego ser compartido con los tutores y de esta forma obtener feedback del mismo.

7.4. Desviaciones

Luego de finalizado el proyecto se pueden observar ciertas desviaciones a lo largo de las distintas etapas. En la etapa de análisis la desviación ocurrió debido a los cambios en los requerimientos solicitados por el cliente. Con respecto a la implementación, los errores en la estimación y la complejidad de las tareas fueron los causantes de la desviación en dicha etapa. Por último, la desviación en la etapa de documentación está relacionada a la falta de documentación generada a lo largo del proyecto. Las desviaciones mencionadas anteriormente se pueden observar en la figura 7.3

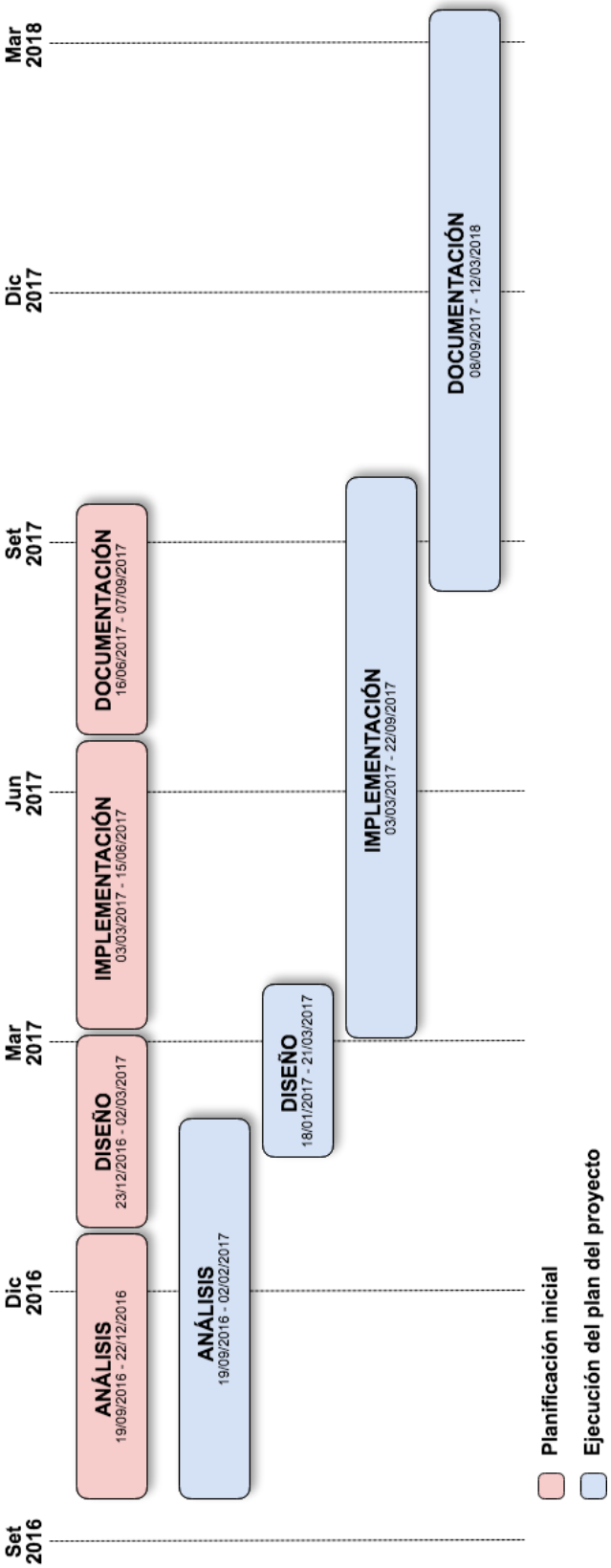


Figura 7.3: Desviaciones en las etapas del proyecto

8. Trabajo a futuro

En este capítulo se presentan los trabajos a futuro y las posibles mejoras a incluir en la herramienta. Algunas de estas tareas agregan mayor valor a la herramienta.

Requerimientos fuera del alcance

Un punto no menor a tener en cuenta es la necesidad del cliente de continuar el proyecto a partir de esta versión, agregando más servicios a la herramienta para lograr una solución más completa. Inicialmente se deberían agregar los requerimientos que quedaron fuera del alcance.

Agregar nuevos proveedores de infraestructura en la nube

Dado de que la herramienta soporta solamente AWS y Azure, y dicha solución tiene como objetivo principal abstraer al usuario de los proveedores de infraestructura en la nube, como trabajo a futuro se podría agregar el soporte para nuevos proveedores de infraestructura en la nube, por ejemplo, vCloud o Google Cloud Platform. Para realizar esta tarea ver “ANEXO C - Manual técnico”.

Actualización de tecnologías utilizadas

Con respecto a las tecnologías utilizadas, es conveniente actualizar a las últimas versiones estables de estas. Se utilizaron las últimas versiones estables al momento de comenzar con la implementación. Para el caso particular de nuevas versiones de la API de un proveedor, si estas ofrecen nuevas funcionalidades las cuales fueron implementadas en Bas7ion a través de aprovisionamiento de software, es conveniente actualizar Bas7ion para poder utilizar directamente la nueva API y de esta forma eliminar complejidad en la implementación. Por ejemplo, el servicio de DNS privado en Azure y VPN de AWS.

Autenticación temporal con proveedores de infraestructura en la nube

En cuanto a los aspectos de seguridad relacionados a la herramienta, es importante mencionar que Vault dispone de autenticación temporal para AWS. Este tipo de autenticación genera credenciales (Access Keys y Secret Keys) de conexión con AWS que son válidas por un tiempo o sesión determinado. Utilizar este mecanismo en Bas7ion agregaría una capa más de seguridad dado que el usuario Bas7ion no tendría nunca visibilidad sobre las credenciales estáticas (no temporales) de acceso a AWS.

La razón por la que dicho servicio no fue utilizado en la solución fue debido a que esta funcionalidad de Vault no está disponible para Azure. Aun así, sería conveniente

agregar este nivel de seguridad en la herramienta. Para llevar a cabo esta mejora se debe modificar el diseño del componente manejador de secretos.

Transaccionalidad

Actualmente la herramienta no permite realizar el despliegue de la infraestructura de forma transaccional, es decir en caso fallo de una operación, no se revierten los cambios aplicados anteriormente. Esto genera cierta complejidad para volver al estado de la infraestructura previo a la ejecución. Por este motivo sería conveniente que la herramienta sea transaccional.

Valores por defecto de las operaciones

Los valores por defecto que utilizan las operaciones en la herramienta no cuentan con una estrategia definida. Por ejemplo, a la hora de desplegar una máquina virtual, se podría utilizar un algoritmo que correlacione los tamaños y/o imágenes, entre los distintos proveedores de infraestructura en la nube.

9. Conclusiones

Como conclusión general del proyecto se puede destacar que el cliente quedó altamente satisfecho, ya que se cumplieron sus expectativas y se alcanzaron los objetivos propuestos. Se logró desarrollar la herramienta, la cual le da un valor agregado al área de infraestructura en la organización de la que forma parte. A su vez, a lo largo del proyecto se adquirió conocimiento y experiencia sobre el área. Esto fue posible, en parte, al conocimiento previo adquirido a lo largo de la carrera sobre las temáticas involucradas, por ejemplo: redes, sistema operativos y programación. Aun así, creemos que sería conveniente incluir dentro de la currícula conocimientos académicos relacionados al área de infraestructura en la nube.

Con respecto a las tecnologías utilizadas, en primer lugar, Apache Libcloud fue muy importante a la hora de desarrollar la herramienta ya que resuelve gran parte de la comunicación con los proveedores de infraestructura en la nube. Además, el hecho que la herramienta sea de código abierto y se tenga acceso a su implementación, contribuyó al momento de realizar parte del diseño de la solución. En contrapartida, a nuestro criterio, creemos que Libcloud no ha logrado un nivel de maduración y estabilidad suficiente. Esto se debe a que algunas funcionalidades no están actualizadas y/o poseen errores, y que además no cuenta con la totalidad de las operaciones ofrecidas por los proveedores. Debido a la investigación realizada sobre Ansible concluimos que es una herramienta con un gran potencial ya que provee muchas funcionalidades útiles y posee una comunidad activa de usuarios y colaboradores.

A pesar de que AWS y Azure son los proveedores más relevantes, se encontraron dificultades en ciertos servicios. Por ejemplo, la resolución de nombre de dominio privado en Azure y la comunicación a través de VPN en AWS, los cuales debieron ser implementados en la herramienta haciendo uso de componentes propios. De acuerdo con lo anterior opinamos que en la actualidad existen limitaciones importantes en los servicios que ofrecen los proveedores.

En cuanto a las distintas etapas del proyecto, entendemos que es fundamental seguir el orden establecido de las mismas, pero creemos que la etapa de documentación debe ser transversal a todas las etapas. De esta forma a la hora de realizar la documentación final, se dispone de una base de documentos.

Un hecho relevante fue contar con un cliente que tuviera conocimientos técnicos del área de infraestructura. Esto permitió poder asimilar de manera más simple los objetivos del proyecto. También, contribuyó para resolver ciertos problemas técnicos que fueron surgiendo a lo largo del proyecto.

Desde un inicio se dejó en claro que la herramienta sería el puntapié inicial para validar la idea del proyecto para luego, en un futuro cercano, seguir trabajando a partir de ésta agregando nuevas funcionalidades. De esta forma concluimos que la comunidad de desarrolladores debe continuar con su desarrollo para aumentar el potencial de la misma. Esto quiere decir que colaborativamente los integrantes de la comunidad puedan aportar sus conocimientos y nuevas ideas que surjan a futuro.

REFERENCIAS

[1] Pyxis [En línea]. Available: http://www.pyxisportal.com/ [Último acceso: 20 febrero 2018].
[2] What is cloud computing? [En línea]. Available: https://www.ibm.com/cloud-computing/learn-more/what-is-cloud-computing/ [Último acceso: 17 setiembre 2017].
[3] The NIST definition of Cloud Computing [En línea]. Available: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf [Último acceso: 22 setiembre 2017].
[4] Get the RightScale State of the Cloud Report [En línea]. Available: https://www.rightscale.com/lp/state-of-the-cloud [Último acceso: 7 marzo 2018].
[5] Cloud Computing with Amazon Web Services [En línea]. Available: https://aws.amazon.com/what-is-aws/ [Último acceso: 23 setiembre 2017].
[6] Cloud Products: https://aws.amazon.com/products [En línea]. Available: [Último acceso: 26 setiembre 2017].
[7] Azure products [En línea]. Available: https://azure.microsoft.com/en-us/services/ [Último acceso: 26 setiembre 2017].
[8] What is Azure? [En línea]. Available: https://azure.microsoft.com/en-us/overview/what-is-azure/ [Último acceso: 23 setiembre 2017].
[9] Google Cloud Computing, Hosting Services & APIs [En línea]. Available: https://cloud.google.com/ [Último acceso: 30 setiembre 2017].
[10] Cloud Services & Cloud Infrastructure [En línea]. Available: https://cloud.vmware.com/ [Último acceso: 29 setiembre 2017].
[11] IBM Cloud Computing for Builders and Innovators [En línea]. Available: https://www.ibm.com/cloud-computing/ [Último acceso: 30 setiembre 2017].
[12] Hosting Microsoft Dynamics in the Cloud – evaluating IaaS, PaaS and SaaS [En línea]. Available: https://community.dynamics.com/ax/b/erpsoftwareblog/archive/2016/09/22/hosting-microsoft-dynamics-in-the-cloud-evaluating-iaas-paas-and-saas [Último acceso: 29 setiembre 2017].
[13] Managed Dedicated & Cloud Computing Services [En línea]. Available: https://www.rackspace.com [Último acceso: 29 setiembre 2017].
[14] Build Scalable Web & Mobile Backends in Any Language [En línea]. Available: https://cloud.google.com/appengine/ [Último acceso: 30 setiembre 2017]
[15] Container Application Platform by Red Hat, Built on Docker and Kubernetes [En línea]. Available: https://www.openshift.com/ [Último acceso: 30 setiembre 2017]
[16] Cloud Application Platform [En línea]. Available: https://www.heroku.com/ [Último acceso: 30 setiembre 2017].
[17] Cloud Application Platform - Devops Platform [En línea]. Available: https://www.cloudfoundry.org/ [Último acceso: 30 setiembre 2017].

[18] Google Drive - Cloud Storage & File Backup for Photos, Docs & More [En línea]. Available: https://www.google.com/drive/ [Último acceso: 30 setiembre 2017].
[19] Free Storage and Email from Google [En línea]. Available: https://www.google.com/gmail/about/ [Último acceso: 30 setiembre 2017].
[20] Reinventing teamwork [En línea]. Available: https://www.dropbox.com/ [Último acceso: 30 setiembre 2017].
[21] Welcome to Office. Your place to create, communicate, collaborate, and get great work done [En línea]. Available: https://www.office.com/ [Último acceso: 30 setiembre 2017].
[22] What are public, private and hybrid clouds? [En línea]. Available: https://azure.microsoft.com/en-gb/overview/what-are-private-public-hybrid-clouds/ [Último acceso: 14 noviembre 2017].
[23] What is DevOps? [En línea]. Available: https://aws.amazon.com/devops/what-is-devops/. [Último acceso: 4 setiembre 2017].
[24] DevOps: Breaking the Development-Operations barrier [En línea]. Available: https://www.atlassian.com/devops. [Último acceso: 4 setiembre 2017].
[25] Infrastructure as Code [En línea]. Available: https://d0.awsstatic.com/whitepapers/DevOps/infrastructure-as-code.pdf [Último acceso: 7 noviembre 2017].
[26] Terraform [En línea]. Available: https://www.terraform.io/. [Último acceso: 23 octubre 2017].
[27] Cloud Formation [En línea]. Available: https://aws.amazon.com/es/cloudformation/. [Último acceso: 23 octubre 2017].
[28] Fog [En línea]. Available: http://fog.io/. [Último acceso: 23 octubre 2017].
[29] Supported Providers [En línea]. Available: https://libcloud.readthedocs.io/en/latest/supported_providers.html [Último acceso: 17 octubre 2017].
[30] One Interface to Rule Them All [En línea]. Available: https://libcloud.apache.org/ [Último acceso: 17 octubre 2017].
[31] Chef [En línea]. Available: https://www.chef.io/ [Último acceso: 10 febrero 2018].
[32] Puppet [En línea]. Available: https://puppet.com/ [Último acceso: 10 febrero 2018].
[33] Salt [En línea]. Available: https://saltstack.com/ [Último acceso: 10 febrero 2018].
[34] Newrelic [En línea]. Available: https://newrelic.com/ [Último acceso: 10 febrero 2018].
[35] Función Lambda [En línea]. Available: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html [Último acceso: 10 febrero 2018].
[36] Welcome to Python.org [En línea]. Available: https://www.python.org/ [Último acceso: 17 octubre 2017].

[37] A Tool for managing Secrets [En línea]. Available: https://www.vaultproject.io/ [Último acceso: 17 octubre 2017].
[38] Automation for Everyone [En línea]. Available: https://www.ansible.com/ [Último acceso: 17 octubre 2017].
[39] Bitbucket [En línea]. Available: https://bitbucket.org/ [Último acceso: 19 octubre 2017].
[40] Comparing Workflows [En línea]. Available: https://www.atlassian.com/git/tutorials/comparing-workflows#gitflow-workflow [Último acceso: 19 octubre 2017].
[41] PyCharm [En línea]. Available: https://www.jetbrains.com/pycharm/ [Último acceso: 19 octubre 2017].
[42] StrongSwan [En línea]. Available: https://strongswan.org/ [Último acceso: 20 octubre 2017].
[43] BIND [En línea]. Available: https://www.isc.org/downloads/bind/ [Último acceso: 19 octubre 2017].
[44] Test Pyramid [En línea]. Available: https://martinfowler.com/bliki/TestPyramid.html [Último acceso: 14 noviembre 2017].
[45] Connect your app to Microsoft Azure Active Directory [En línea]. https://auth0.com/docs/connections/enterprise/azure-active-directory/v2 [Último acceso: 31 enero 2018].
[46] What is scrum? [En línea]. Available: https://www.scrum.org/resources/what-is-scrum [Último acceso: 13 octubre 2017].
[47] Trello [En línea]. Available: https://trello.com/ [Último acceso: 23 octubre 2017].

GLOSARIO

API: interfaces de servicios ofrecidos a través de la Web, de manera pública y que sirven de puerta de entrada para el uso de los diferentes servicios que una organización posee.

Base de datos relacional: es una recopilación de elementos de datos con relaciones predefinidas entre ellos. Estos elementos se organizan como un conjunto de tablas con columnas y filas.

Base de datos no relacional: base de datos que permite almacenar información en aquellas situaciones en las que las bases de datos relacionales generan ciertos problemas debido principalmente a problemas de escalabilidad.

Capacidad de cómputo: es la cantidad de procesamiento que puede realizar un sistema de cómputo en una unidad de tiempo.

CDN: es un conjunto de ubicaciones en el mundo, que redistribuyen localmente el contenido de los servidores y guardan en caché los archivos que no necesitan actualización permanente, según unas reglas personalizables.

CIDR: es un estándar de red para la interpretación de direcciones IP.

CLI: es un método que permite a los usuarios dar instrucciones a algún programa informático por medio de una línea de texto simple.

Contenedor de objetos: es el espacio de almacenamiento donde se puede guardar y obtener objetos (imágenes, videos, documentos, etc.).

CPU: es el hardware dentro de una computadora, que interpreta las instrucciones de un programa informático mediante la realización de las operaciones básicas aritméticas, lógicas y de entrada/salida del sistema.

Datacenter: conjunto de equipos conectados en red que las organizaciones o empresas emplean para organizar, procesar, almacenar y difundir, grandes volúmenes de información.

DNS: es un sistema de nomenclatura jerárquico descentralizado para dispositivos conectados a redes IP como Internet o una red privada.

Gateway: es un dispositivo que permite interconectar redes con protocolos y arquitecturas diferentes a todos los niveles de comunicación. Su propósito es traducir la información del protocolo utilizado en una red al protocolo usado en la red de destino.

Integración continua: es un modelo informático que consiste en hacer integraciones automáticas de un proyecto lo más a menudo posible para así poder detectar fallos cuanto antes.

IPsec: es un conjunto de protocolos cuya función es asegurar las comunicaciones sobre el Protocolo de Internet (IP) autenticando y/o cifrando cada paquete IP en un flujo de datos.

JSON: es un formato de intercambio de datos rápido.

On-premise: software instalado y ejecutado en servidores locales de la organización que utiliza el software.

Portal web: sitio web que ofrece, de forma integrada, una amplia variedad de servicios y recursos al usuario.

RAM: memoria principal de la computadora, donde residen programas y datos, sobre la que se pueden efectuar operaciones de lectura y escritura.

Snapshot: es una función de algunos sistemas que realizan copias de seguridad de ficheros almacenándolos tal y como fueron capturados en el pasado.

SSL: protocolo criptográfico que proporcionan autenticación, privacidad e integridad de los datos transferidos entre aplicaciones que se comunican a través de la red.

SysAdmin: es la persona que tiene la responsabilidad de implementar, configurar, mantener, monitorear, documentar y asegurar el correcto funcionamiento de un sistema informático, o algún aspecto de éste.

VPN: es una tecnología de red de computadoras que permite una extensión segura de la red de área local (LAN) sobre una red pública o no controlada como Internet.

YAML: es un formato para el intercambio de información que tiene como objetivo facilitar el mapeo de estructuras de datos más complejas en un documento de texto plano.

APÉNDICE A - Extensión de Libcloud

En este anexo se explica cómo realizó la extensión de la librería Libcloud y como agregar nuevas operaciones en dicha extensión.

Para extender Libcloud lo primero que se realizó fue la extensión de la interfaz de acceso a libcloud (LibcloudDriver), junto a las interfaces de acceso a cada uno de los proveedores (AzureNodeDriver, EC2NodeDriver) para los cuales fue necesario implementar las nuevas operaciones. En la figura A.1 se puede observar lo anteriormente mencionado junto a la interacción entre el driver de Bas7ion y la extensión realizada.

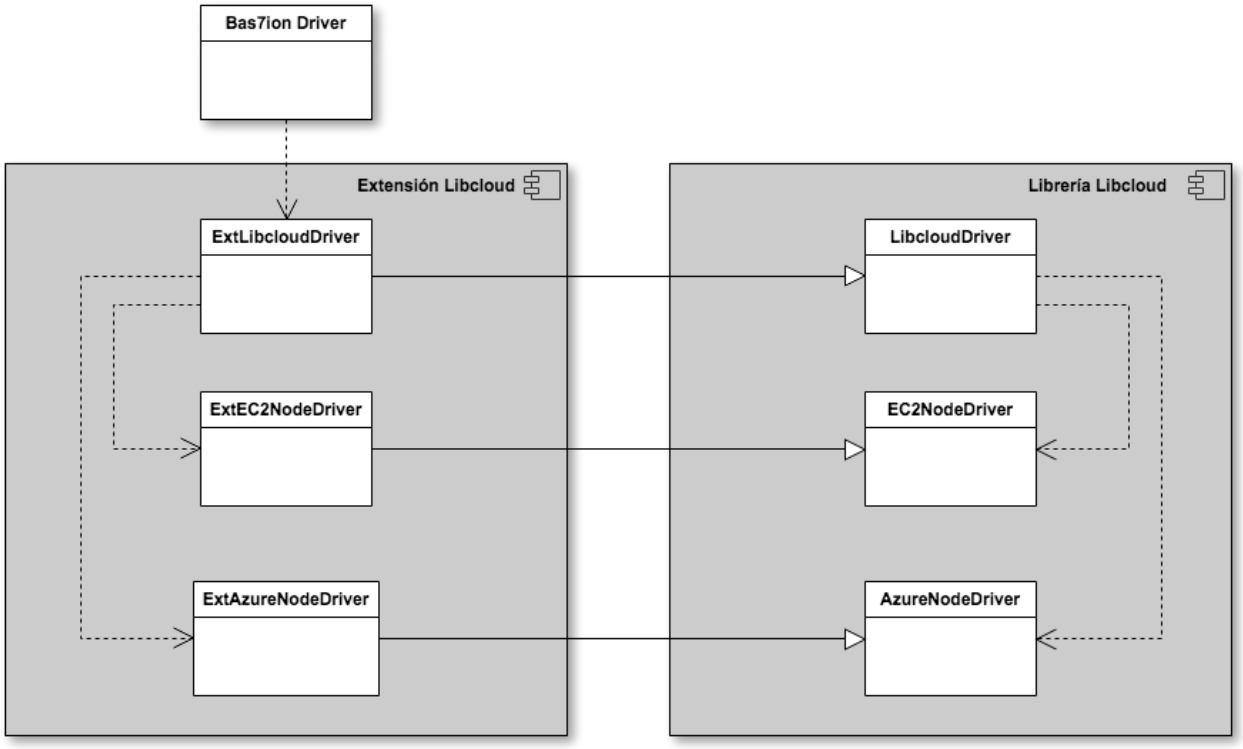


Figura A.1: Diseño de extensión de Libcloud

Para poder tener una referencia en ExtLibcloudDriver la a las extensiones de los drivers de los proveedores (ExtAzureNodeDriver y ExtEC2NodeDriver) se tuvo que sobrescribir la ruta de dichos drivers. Esto se puede observar en la figura A.2.

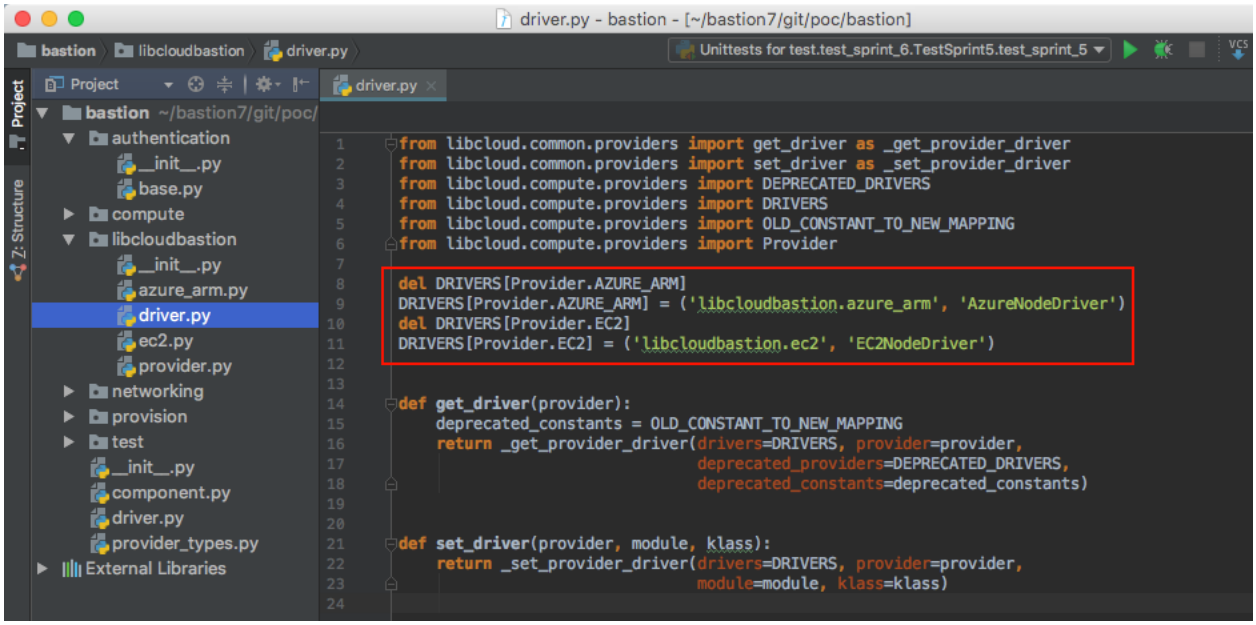


Figura A.2: Ruta a los nuevos drivers de los proveedores

A la hora de agregar una nueva operación en la extensión de Libcloud solo hace falta agregar dicha operación en la extensión del driver del proveedor en el cual se quiere agregar dicha operación. Por ejemplo, si se quiere agregar una nueva operación de AWS, solo hay que agregar la operación en ExtEC2NodeDriver

En caso de querer agregar una operación de un nuevo proveedor se tiene que extender el driver de dicho proveedor existente en Libcloud. Luego se debe sobrescribir la ruta al nuevo driver del proveedor en ExtLibcloudDriver, tal como se muestra en la figura anterior. Finalmente se agrega la operación en la extensión del driver del nuevo proveedor.

APÉNDICE B - Configuración de Vault

En este anexo se explica cómo realizar la configuración del manejador de secretos Vault.

Para los efectos de este proyecto, el servidor Vault se despliega en modo desarrollo desde la misma terminal donde se utiliza la Librería Core. Por lo que en este caso el servidor Vault y la terminal Bas7ion coinciden en el mismo nodo físico.

En comparación con el modo de producción, las características del modo desarrollo son las siguientes:

- No requiere ser configurado, ya que inicializa el servidor con la configuración por defecto.
- El cliente de consola ya se encuentra autenticado para ejecutar operaciones.
- La información es almacenada en memoria, por lo tanto no requiere permisos del sistema de archivos.
- La API de Vault es iniciada en la siguiente ruta: http://127.0.0.1:8200

En la figura B.1 se muestra la inicialización del servidor Vault en modo desarrollo.

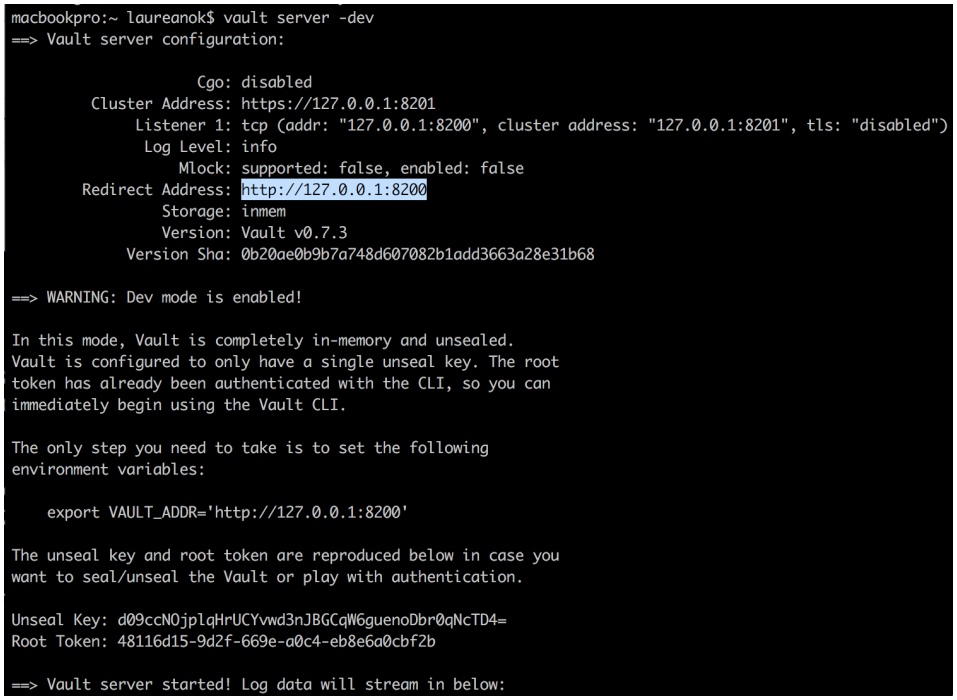


Figura B.1: Inicialización del servidor Vault en modo desarrollo

Una vez iniciado el servidor Vault y luego de configurada la variable de ambiente *"VAULT_ADDR"* (esta variable necesita configurarse en cada terminal que utilice Bas7ion), se cargan los secretos que utiliza Bas7ion.

El formato de la ruta de los secretos en Vault brinda flexibilidad al usuario a la hora de organizar su biblioteca de secretos. Lo única restricción en la ruta es que comience con el prefijo "secret**",** el cual es el identificador del tipo de backend.

En Bas7ion se define el siguiente formato de ruta para guardar las credenciales:

secret/{project_id}/{provider_access_id}

- *project_id*: es un valor alfanumérico que el usuario utiliza para identificar un proyecto de infraestructura en particular o el cliente asociado a dicho proyecto.
- *provider_access_id*: es un valor alfanumérico que el usuario utiliza para identificar las credenciales de un proveedor, como por ejemplo *"aws_1"*.

Para cargar las credenciales de AWS se requiere de un archivo JSON, como se muestra en la figura B.2:

```
{
  "cloud_provider" : "aws",
  "access_key" : "AKIAIINV67XXXXXXXXXX",
  "secret_key" : "8/W7A6opGyfW3Bi04zkBaNiCn4h1XXXXXXXXXXXX",
  "region" : "us-west-2"
}
```

Figura B.2: Ejemplo de archivo JSON para AWS

El valor de la clave *"cloud_provider"* se requiere para poder mapear la estructura del JSON a la clase CloudProviderInfo de la Librería Core.

Para cargar el secreto se ejecuta el siguiente comando *"vault write path_al_secreto archivo_json"*

```
macbookpro:vault_credentials laureanok$ vault write secret/project1/aws @aws_diego.json
Success! Data written to: secret/project1/aws
```

Para consultar el valor del secreto recientemente guardado se utiliza el comando "*vault read path_al_secreto*"

```
macbookpro:vault_credentials laureanok$ vault read secret/project1/aws
Key                 Value
---                 -----
refresh_interval    768h0m0s
access_key          AKIAIINV67XXXXXXXXXX
cloud_provider      aws
region              us-west-2
secret_key          8/W7A6opGyfW3Bi04zkBaNiCn4h1XXXXXXXXXXXX
```

Análogamente, se guardan las credenciales de Azure como se muestra en la figura B.3:

```
{
  "cloud_provider" : "azure",
  "tenant_id" : "36b707d7-9461-XXXX-XXXX-XXXXXXXXXXXX",
  "subscription_id" : "7bf43588-5056-XXXX-XXXX-XXXXXXXXXXXX",
  "application_id" : "06a9e2d0-2f35-XXXX-XXXX-XXXXXXXXXXXX",
  "password" : "C1mhB5mSVa/YxwXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
}
```

Figura B.3: Ejemplo de archivo JSON para Azure

Luego de repetir el proceso para Azure, se cuentan con las credenciales cargadas para ambos proveedores de infraestructura en la nube. Para acceder a las mismas, desde la Librería Core, solo resta indicar el token de acceso a Vault y la ruta de la credencial a utilizar.

```
auth_driver = VaultAuthDriver('48116d15-9d2f-669e-a0c4-eb8e6a0cbf2b')
provider_cred = auth_driver.get_provider_info('secret/project1/aws')
```

APÉNDICE C - Manual Técnico

Las características más importantes de Bas7ion son su extensibilidad y generalización, tanto a nivel de proveedores de infraestructura en la nube como de funcionalidades. Este manual se enfoca en detallar cómo se llevan a cabo dichas tareas.

Dicho manual se divide en 2 secciones:

- *Cómo agregar un nuevo proveedor*: implementar todas las funcionalidades ya existentes en Bas7ion para un nuevo proveedor.
- *Cómo agregar una nueva funcionalidad*: agregar una nueva funcionalidad en todos los proveedores soportados por Bas7ion.

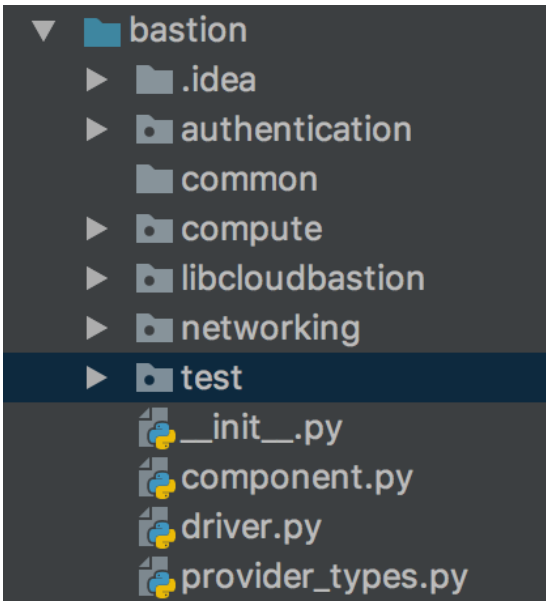
En la figura C.1 se representan gráficamente dichas secciones.

Figura C.1: Comparación Nueva Funcionalidad vs Nuevo Proveedor

C.1 Nuevo Proveedor

Para agregar un nuevo proveedor es necesario entender el diseño de la solución, sino también la estructura de la Librería Core.

La estructura de la librería consiste en las siguientes partes:



- Master Driver (driver.py)
- Authentication Module
- Compute Module
- Networking Module
- LibcloudBastion (Véase “ANEXO A - Extensión de Libcloud”)
- Tests

```
class Providers():
    AZURE = 'azure'
    AWS = 'aws'
    VCLOUD = 'vcloud'
```

provider_types: se debe agregar el identificador del nuevo proveedor en dicha clase, el cual contiene los identificadores de todos los proveedores de infraestructura en la nube soportados.

C.1.1 Authentication Module

Se debe implementar la clase ProviderInfo (en el archivo base.py), la responsabilidad de dicha clase es la siguiente:

- Determina el tipo a la clase ProviderInfo.
- Cargar las credenciales y las propiedades desde los JSONs que retorna Vault.

C.1.2 Master Driver

Se debe implementar la clase Driver (en el archivo driver.py), la responsabilidad de dicha clase es la siguiente:

- Asignar el Cloud API Client correspondiente al proveedor en cuestión. En este caso se utiliza Libcloud, pero podría haber otros adicionales en el futuro.
- Instanciar los módulos soportados por Bas7ion. Actualmente se utilizan Compute y Networking, más adelante se podrían agregar otros.

- Asignar las credenciales y propiedades obtenidas previamente a través módulo de Autenticación para ser utilizadas posteriormente.

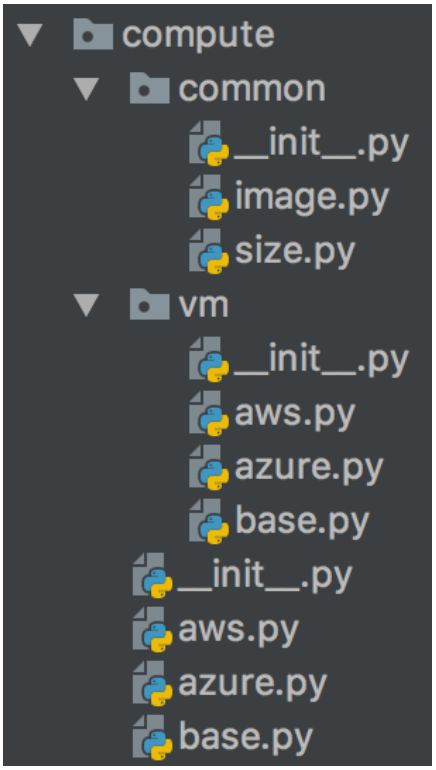
Una vez creada la nueva implementación de Driver para el nuevo proveedor, debe agregarse al método estático de creación de la clase Driver. Este método implementa el patrón factory que se describe en el capítulo de Diseño.

```
@staticmethod
def create(provider_info):

    if provider_info.type == Providers.AWS:
        driver = AWSDriver(provider_info)
    if provider_info.type == Providers.AZURE:
        driver = AzureDriver(provider_info)

    return driver
```

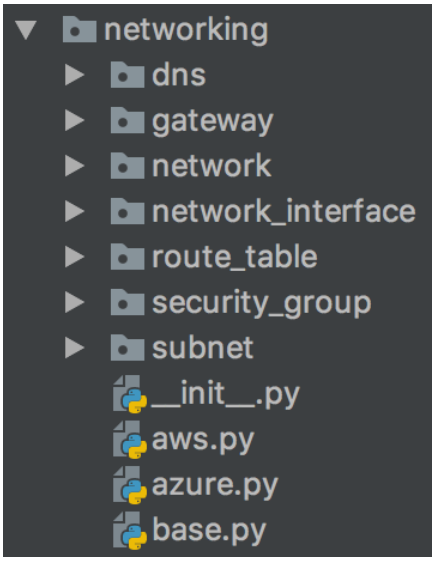
C.1.3 Compute Module



- Implementar la interfaz BaseCompute que se encuentra en compute/base.py, en un nuevo archivo .py dentro del paquete compute.
- Implementar la interfaz VirtualMachine que se encuentra en compute/vm/base.py, en un nuevo archivo .py dentro del paquete compute/vm.

C.1.4 Networking Module

Su extensión es análoga al módulo de compute.



- Implementar la interfaz Networking que se encuentra en networking/base.py en un nuevo archivo .py dentro del paquete networking
- Implementar la interfaz Network que se encuentra en networking/network/base.py en un nuevo archivo .py dentro del paquete networking/network
- Implementar la interfaz Subnet que se encuentra en networking/subnet/base.py en un nuevo archivo .py dentro del paquete networking/subnet
- Implementar la interfaz NetworkInterface que se encuentra en networking/network_interface/base.py en un nuevo archivo .py dentro del paquete networking/network_interface
- Implementar la interfaz SecurityGroup que se encuentra en networking/security_group/base.py en un nuevo archivo .py dentro del paquete networking/security_group
- Implementar la interfaz RouteTable que se encuentra en networking/route_table/base.py en un nuevo archivo .py dentro del paquete networking/route_table
- Implementar la interfaz Gateway que se encuentra en networking/gateway/base.py en un nuevo archivo .py dentro del paquete networking/gateway
- Implementar la interfaz Dns que se encuentra en networking/dns/base.py en un nuevo archivo .py dentro del paquete networking/dns

C.2 Nueva Funcionalidad

A la hora de agregar una nueva funcionalidad debemos distinguir los siguientes escenarios:

1. Agregar una nueva operación dentro de una entidad existente.
2. Agregar una nueva entidad dentro de un módulo existente (por ejemplo: VM en Compute, Network en Networking).
3. Agregar un nuevo módulo (por ejemplo: Compute, Networking).

El orden de los escenarios utiliza un enfoque bottom-up lo cual significa que un escenario de mayor orden incluye el de menor orden. Por ejemplo, dentro de las tareas a realizar para la creación de una nueva entidad se deben implementar las operaciones de la misma.

Para agregar una nueva operación en una entidad existente se requiere:

1. Definir el método en la interfaz (clase base.py de la entidad).
2. Implementar dicho método en cada una de las implementaciones de los proveedores de infraestructura en la nube soportados.

Para agregar una nueva entidad en un módulo existente se requiere:

1. Ubicar el módulo relacionado a dicha entidad.
2. Crear un paquete vacío dentro del módulo (directorio con el archivo vacío __init__.py).
3. Crear la interfaz de la nueva entidad con las operaciones a implementar (utilizar “base.py” como nombre de archivo para las interfaces).
4. Implementar dicha interfaz para cada proveedor (utilizar el nombre del proveedor como nombre de archivo de la implementación).

Para agregar un nuevo módulo:

1. Crear un paquete en la raíz del proyecto.
2. Crear la interfaz del nuevo módulo con las operaciones a implementar (utilizar “base.py” como nombre de archivo para las interfaces)
3. Dentro de las implementaciones de los controladores de los distintos proveedores de infraestructura en la nube (que se encuentran en el archivo driver.py de la raíz del proyecto), instanciar del nuevo módulo, como se puede ver en la figura C.2.

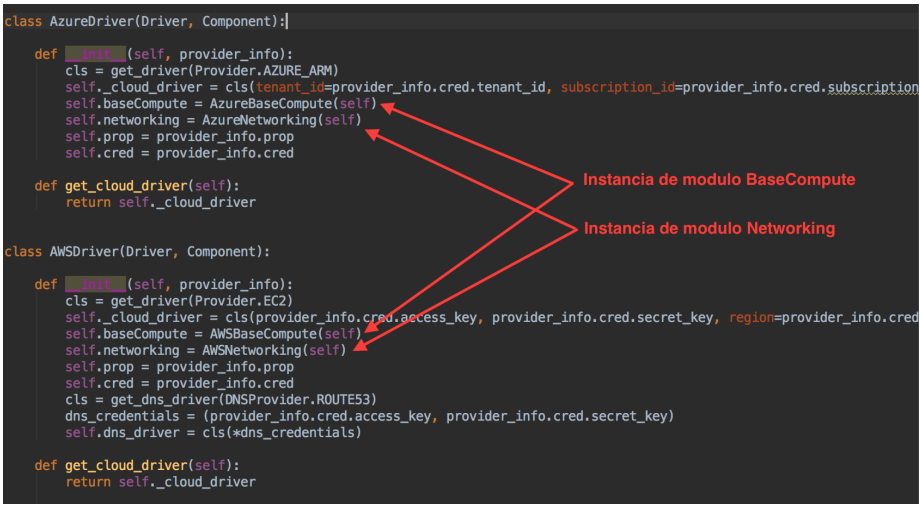


Figura C.2: Ejemplo de instancia de los módulos de Bas7ion

APÉNDICE D - Pruebas realizadas

A continuación de presentan las pruebas de los casos más relevantes y los resultados obtenidos. Las pruebas fueron realizadas en los proveedores que soporta la herramienta (Bas7ion): AWS y Microsoft Azure.

Funcionalidad	Caso de prueba	Resultado esperado	Resultado obtenido
Creación de máquinas virtuales	- Crear una máquina virtual.	La herramienta crea una máquina virtual	El esperado
Creación de redes	- Crear una red.	La herramienta crea una red	El esperado
Creación de subredes	- Crear una red. - Crear una subred en cada red.	La herramienta crea la subred dentro de una red	El esperado
Creación de interfaces de red	- Crear una red. - Crear una subred en cada red. - Crear una interfaz de red en cada subred.	La herramienta crea la interfaz de red dentro de una subred	El esperado
Creación de una máquina virtual con una interfaz de red	- Crear una red. - Crear una subred en cada red. - Crear una interfaz de red en cada subred. - Crear una máquina virtual. - Asociar la interfaz de red a la máquina virtual.	La herramienta crea una máquina virtual con su interfaz de red asociada	El esperado
Aprovisionar máquina virtual	- Crear una red. - Crear una subred en cada red. - Crear una interfaz de red en cada subred. - Crear una	La herramienta aprovisiona una máquina virtual con un software dado	El esperado

	máquina virtual. - Asociar la interfaz de red a la máquina virtual. - Aprovisionar la máquina virtual con un software dado.		
Comunicación de subredes dentro de una red	- Crear una red. - Crear 2 subredes dentro de la red. - Comunicar las 2 subredes	La herramienta comunica las 2 subredes dentro de la red	El esperado
Comunicación de redes	- Crear 2 redes. - Generar los recursos dentro de la red para establecer la comunicación. - Comunicar las 2 redes.	La herramienta comunica las 2 redes	El esperado
Resolución de nombres	- Crear una red. - Crear una subred en cada red. - Crear una interfaz de red en cada subred. - Crear una máquina virtual. - Asignar hostname para la ip local de la máquina virtual	La herramienta permite nombrar la ip local de la máquina virtual a partir de un hostname.	El esperado
Autenticación de proveedores de infraestructura en la nube	- Ingresar el token de autenticación. - Obtener las credenciales y configuraciones de los proveedores de infraestructura en la nube.	La herramienta obtiene las configuraciones y credenciales de los proveedores de infraestructura en la nube a partir de un token de autenticación.	El esperado