



ugr | Universidad
de **Granada**

TRABAJO FIN DE GRADO
INGENIERÍA EN INFORMÁTICA

Despliegue automático de infraestructura + CI/CD

Implementación de técnicas DevOps

Autor

Víctor Moreno Jiménez

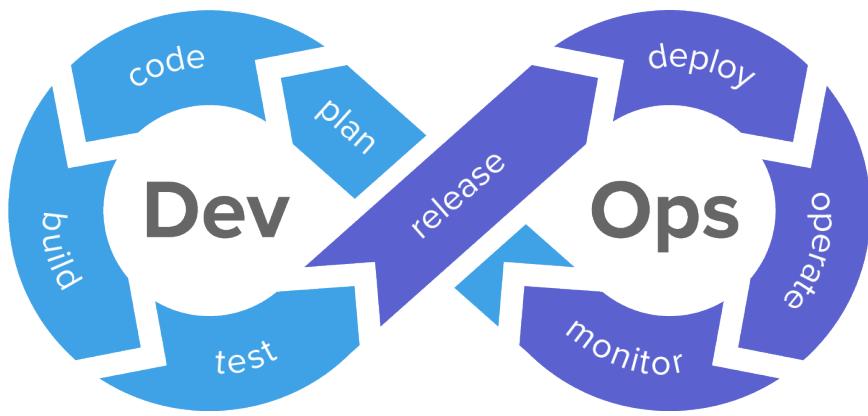
Directores

Juan Julián Merelo Guervós



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, Junio de 2020



Despliegue automático de infraestructura + CI/CD

Implementación de técnicas DevOps

Autor

Víctor Moreno Jiménez

Directores

Juan Julián Melero Guervós

Despliegue automático de infraestructura + CI/CD: Implementación de técnicas DevOps

Víctor Moreno Jiménez

Palabras clave: DevOps, IaC (Infrastructure as Code), CI (Continuous Integration), CD (Continuous Delivery)

Resumen

El objetivo de este TFG es implementar las técnicas actuales conocidas como DevOps en una empresa con limitados recursos.

En concreto se va a automatizar todo el proceso de despliegue de infraestructura (IaC) y se van a aplicar los principios de integración continua y entrega continua propios de la filosofía DevOps en todo el ciclo de desarrollo de los productos de la empresa.

La finalidad de esto es poder disponer de una infraestructura sólida y fácilmente recreable en caso de catástrofe o ampliación del cluster. Desde el punto de vista de los desarrolladores de la empresa, podrán tener a su disposición y bajo demanda, un entorno de desarrollo mediante ficheros de configuración sin depender directamente del departamento de sistemas.

Automatic deployment of infrastructure + CI / CD: Implementation of DevOps techniques

Víctor Moreno Jiménez

Keywords: DevOps, IaC (Infraestructure as Code), CI (Continous Integration), CD (Continous Delivery)

Abstract

The main goal of this project is to apply known DevOps techniques in a company with limited resources.

The entire company infraestructure will be automated using some orchestration tools via configuration files (Infrastructure as Code). DevOps philosophy will be applied throughout the development cycle of the company's products.

The purpose of this is to be able to have a solid and easily deployable infrastructure. From the point of view of the company's developers, they will have the hability to create development infrastructures on demand without depending directly on the systems department

Yo, **Víctor Moreno Jiménez**, alumno de la titulación Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 47252942V, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Víctor Moreno Jiménez

Granada a 31 de Julio de 2020.

D. **Juan Julián Merelo Guervós**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Automatic deployment of infrastructure + CI / CD: Implementation of DevOps***, ha sido realizado bajo su supervisión por **Víctor Moreno Jiménez**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 25 de Agosto de 2020.

Los directores:

Fdo: Juan Julián Merelo Guervós

Agradecimientos

Para toda la gente maravillosa que me he cruzado en estos 4 años de carrera. No habría sido lo mismo sin vosotros... Gracias

Índice general

1. Introducción	1
1.1. Introducción al problema	1
1.2. Motivación	2
1.3. Conceptos básicos	2
1.3.1. DevOps	2
1.3.2. Integración Continua (CI)	3
1.3.3. Despliegue continuo (CD)	4
1.3.4. Infraestructura como Código (IaaC)	4
1.4. Estado del arte	5
1.4.1. Introducción	5
1.4.2. Proveedores Cloud	5
1.5. Objetivos del TFG	8
1.5.1. Objetivos propuestos	8
1.5.2. Grado de cumplimiento de los objetivos propuestos . .	9
2. Análisis del sistema	11
2.1. Definición del problema	11
2.2. Que se pretende resolver	11
2.3. Casuísticas que se dan y como se resuelve cada una	12
2.4. Tipos de usuario	12
2.5. Especificación de requisitos	13
2.6. Diagramas	14
2.6.1. Diagramas casos de uso	14
2.6.2. Diagramas de secuencia	15
2.7. Arquitectura del Sistema	17
2.7.1. Servidores Físicos	17
2.7.2. Infraestructura objetivo	17
2.8. Metodología de desarrollo	19
2.8.1. Milestones	19
2.8.2. Issues	20

3. Planificación	23
3.1. Estimación recursos necesarios	23
3.1.1. Estimación temporal	23
3.2. Presupuesto	25
4. Diseño	27
4.1. Solución adoptada	27
4.1.1. Proveedor Infraestructura	27
4.1.2. Servicio Virtualización	29
4.1.3. Firewall	30
4.1.4. Contenedores	33
4.1.5. Orquestación contenedores	33
4.1.6. Almacenamiento estático	34
4.1.7. Servidor Web / Proxy	35
4.1.8. Control de versiones	36
4.1.9. Base de datos	37
4.2. Diseño cluster	38
4.2.1. Docker Swarm	38
4.2.2. Cluster	38
4.3. Posibles mejoras	40
5. Conclusiones	41
5.1. Futuro del proyecto	41
5.1.1. Adaptación proyecto distintas plataformas	41
5.1.2. Creación WUI para el despliegue de playbooks	41
6. Anexo I: Manual Usuario	43
6.1. Despliegue automático sobre bare metal	43
6.1.1. Requisitos previos	43
Bibliografía	47

Índice de figuras

1.1.	¿Qué es DevOps?	3
1.2.	Integración Continua	3
1.3.	Despliegue Continuo	4
1.4.	Infraestructura como Código	4
2.1.	Casos de uso Administrador	14
2.2.	Casos de uso Desarrollador	15
2.3.	Diagrama secuencia Desarrollador 1	15
2.4.	Diagrama secuencia Desarrollador 2	16
2.5.	Diagrama secuencia Desarrollador 3	16
2.6.	Diagrama secuencia Administrador 1	16
2.7.	Diagrama secuencia Administrador 2	17
2.8.	Diagrama secuencia Administrador 3	17
2.9.	Infraestructura Objetivo	18
2.10.	GitHub milestones	19
2.11.	GitHub panel	21
3.1.	Diagrama Gantt	24
4.1.	Hetzner Data Center	28
4.2.	Hetzner Building	28
4.3.	Proxmox WUI	30
4.4.	Ciberataques	31
4.5.	pfSense interfaz web	32
4.6.	CEPH interfaz web	35
4.7.	Estadísticas Servidores Web	36
4.8.	Diseño Docker Swarm	38
4.9.	Diseño Cluster	40

Índice de cuadros

3.1. Presupuesto	25
----------------------------	----

Capítulo 1

Introducción

1.1. Introducción al problema

T radicionalmente, en el mundo IT han existido dos departamentos claramente diferenciados entre sí, Desarrolladores y Operaciones. Éstos hacen referencia a dos tipos de informático. Por un lado el desarrollador de aplicaciones, que se encarga de escribir código para cumplir unos requisitos funcionales y darle forma al producto. Este primer grupo se caracteriza por únicamente preocuparse de que la aplicación funcione correctamente en su entorno de desarrollo y que cumpla las funcionalidades descritas por el cliente. Por otro lado nos encontramos la otra cara de la moneda, el perfil de operaciones o sistemas o, como lo llaman algunos, 'sysadmin'. Éstos no se encargan de desarrollar aplicaciones, se encargan de realizar todas las operaciones que hay alrededor de una aplicación a nivel de sistemas, a la hora de lanzarla en un entorno de producción. Un sysadmin se encargará de toda la parte de pruebas y despliegue así como del mantenimiento de los servidores una vez desplegada la aplicación.

Alguno ya sabrá y se habrá dado cuenta de que esta diferenciación y separación de un trabajo en dos bandos diferenciados va a causar problemas a largo plazo y no se equivoca. De hecho, si preguntas a cualquiera que haya trabajado con esta filosofía de trabajo te explicará rápidamente los problemas. Al tener los equipos separados, el desarrollador únicamente se preocupa de que su aplicación funcione en su entorno de desarrollo, lavándose las manos y dejando toda la responsabilidad en manos del equipo de sistemas. Del mismo modo, el equipo de sistemas ante cualquier fallo en la aplicación, echarán las culpas al equipo de desarrollo, ya que es su trabajo que la aplicación funcione correctamente y así sucesivamente... Son habituales los comentarios del tipo: 'Si a mí en local me funciona...' O ante cualquier fallo en producción: 'La aplicación funciona correctamente, seguro que son los sistemas...'. Como era de suponer, este enfoque no iba a durar mucho tiempo ya que los in-

convenientes de esta forma de trabajar (sobre todo en equipos grandes) son insostenibles. Desafortunadamente hoy en día sigue habiendo empresas con esta filosofía de trabajo, sobre todo empresas con una larga trayectoria que se niegan a cambiar algo que 'ya funciona'. Este trabajo pretende justamente facilitar la adopción de técnicas DevOps a una pequeña empresa, cambiando el enfoque y la forma de trabajar de los departamentos de desarrollo y sistemas.

1.2. Motivación

Allá por 2007, en la Agile Conference de Toronto, Andrew Shafer (fundador de Puppet) da una charla sobre aplicar la metodología Agile a la infraestructura. Su único oyente Patrick Debois interesado por el tema, mantiene una larga conversación sobre el tema y deciden fundar una lista de correo: Agile System Administration. Este momento ha sido crucial para la filosofía DevOps ya que es en esta lista donde empiezan a germinar algunos conceptos conocidos hoy día como integración continua, entrega continua, infraestructura como código, etc.

Es en 2009 cuando se produce el famoso congreso 'O'reilly Velocity Conference' donde se realiza una charla llamada '10 Deploys per day'. En esta charla se empieza a plantear seriamente la fusión de los departamentos de Desarrollo y Operaciones. A raíz de esta charla nacen los llamados DevOpsdays, desarrolladores y equipo de operaciones empiezan a trabajar conjuntamente dando ideas de como poder fusionar ambos departamentos. Los DevOpsdays, rápidamente se extienden por todo el mundo popularizándose el término DevOps como fusión de Desarrollo y Operaciones.

1.3. Conceptos básicos

A continuación se van a describir uno a uno, los conceptos básicos para entender este proyecto. Son conceptos claves sin los cuales un lector inexperto no comprenderá ni el problema ni la solución adoptada para el problema.

1.3.1. DevOps

El término DevOps es una fusión de las dos palabras Desarrollo y Operaciones. DevOps es una filosofía, una forma de abordar el desarrollo de software. El objetivo de DevOps es fusionar los departamentos Desarrollo y Operaciones de forma que sea más fácil y rápida la creación de software. La siguiente imagen define el término:

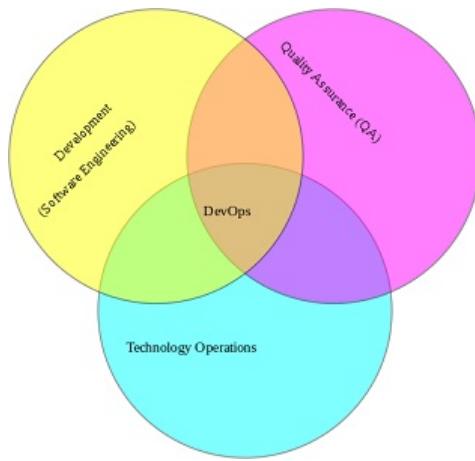


Figura 1.1: ¿Qué es DevOps? [1]

1.3.2. Integración Continua (CI)

La Integración Continua o CI para abreviar, es uno de los pilares de la filosofía DevOps. La integración continua se basa en hacer integraciones automáticas de un proyecto lo más a menudo posible para poder detectar fallos rápidamente. La Integración Continua consta de dos partes: compilación y ejecución de test de un proyecto.

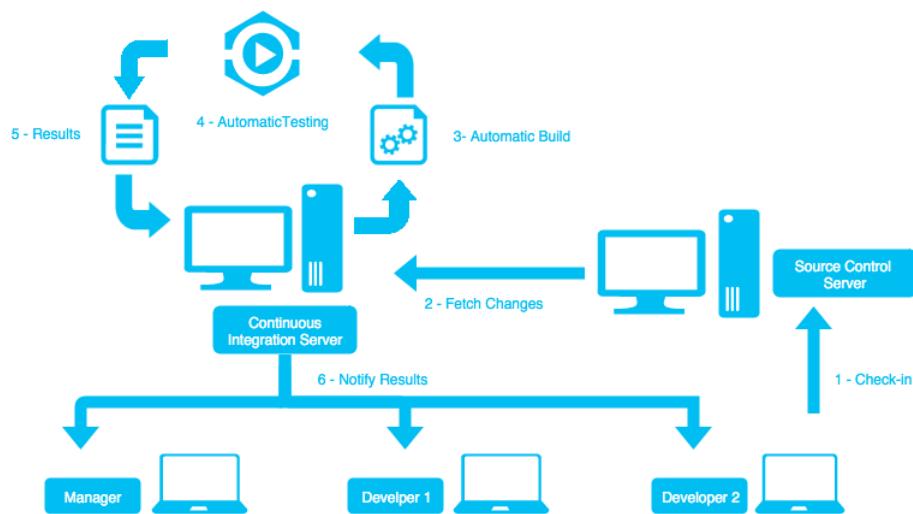


Figura 1.2: Integración Continua [2]

1.3.3. Despliegue continuo (CD)

En el Despliegue Continuo o CD por sus siglas en inglés 'Continous Delivery' complementa a la integración continua desplegando el proyecto software en los servidores de producción una vez ha pasado el proceso de la integración continua. Gracias a CD podemos garantizar entregas rápidas y seguras de software.

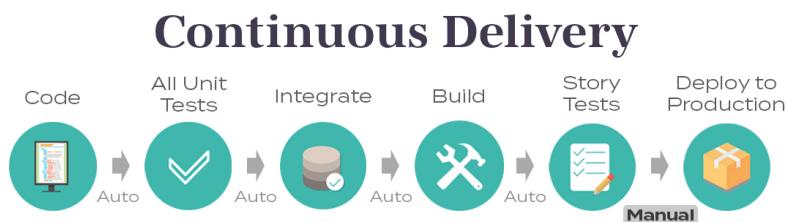


Figura 1.3: Despliegue Continuo [3]

1.3.4. Infraestructura como Código (IaaC)

La infraestructura como código pretende tratar los servidores y toda la infraestructura alrededor de una organización como un software de programación. De este modo, la infraestructura está escrita en ficheros de configuración y es fácilmente replicable y testeable. Este concepto al igual que los dos anteriores está íntimamente ligado con el término DevOps, ya que es uno de los primeros pasos a adoptar. IaaC pretende difuminar la línea entre el código que ejecutan las aplicaciones y el código que configura la infraestructura. IaaC acerca a los desarrolladores al equipo de operaciones o administradores de sistemas.

De esta manera no únicamente se testea el software antes de ser lanzado, si no también la infraestructura. A continuación se muestra como sería el cauce de trabajo siguiendo los principios de IaaC.

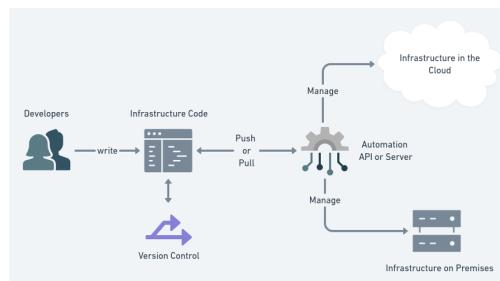


Figura 1.4: Infraestructura como Código [4]

1.4. Estado del arte

En esta sección se va a comentar el estado actual de proyectos parecidos al que se describe en este documento. Sin embargo, antes de abordar esta cuestión me gustaría hacer un pequeño análisis de cómo ha evolucionado el mercado laboral de este tipo de puestos de trabajo para hacer énfasis en la importancia de todo lo mencionado anteriormente.

1.4.1. Introducción

Es una realidad que cada día más las empresas optan por soluciones Cloud y es bastante sencillo para un programador medio crear cauces de desarrollo y despliegue para sus aplicaciones en éstos proveedores Cloud y convertirse en todo un auténtico DevOps. Se ha hablado mucho de la muerte de la figura del Administrador de Sistemas con el auge de este tipo de tecnologías, sin embargo, siempre es buena idea tener un perfil de Sistemas en los equipos DevOps ya que aportan una visión distinta y su experiencia configurando servidores, se convierte en un gran capital para un equipo DevOps.

Sin embargo, no todas las empresas pueden permitirse los costes de este tipo de proveedores Cloud, que si bien son muy cómodos y es una muy buena solución para la infraestructura, son bastante costosos de mantener.

Uno de los objetivos de este proyecto es aportar una solución integral tanto para la infraestructura donde se van a alojar los proyectos software, como la infraestructura que aloja dicha infraestructura, dando una solución directa a empresas con bajo presupuesto que no pueden permitirse soluciones Cloud.

1.4.2. Proveedores Cloud

A continuación se va a hacer un análisis de los principales proveedores Cloud, ya que no se ha encontrado nada similar a lo que pretende este proyecto. Sin embargo los proveedores Cloud sí ofrecen dichas soluciones, contratando sus servicios.

En estudio vamos a comparar los servicios Cloud y su coste con los servicios contratados para este proyecto (Servidores Bare metal)

Servidores Bare Metal

Se han contratado tres servidores en el proveedor Hetzner. Éstos son tres servidores físicos que cuentan con las siguientes características hardware:

- **tfg.intelligenia.com:** El servidor cuenta con las siguientes características hardware:
 - **CPU:** 8 cores.
 - **RAM:** 64GB
 - **Almacenamiento:** 2 NvMe 500GB Raid 1.
 - **Red:** 1000Mb/s
 - **Tráfico:** Ilimitado
 - **Coste mes:** 38,6555 euros
 - **Coste anual:** 502,521 euros
- **tfg2.intelligenia.com:** El servidor cuenta con las siguientes características hardware:
 - **CPU:** 8 cores.
 - **RAM:** 16GB
 - **Almacenamiento:** 2 SSD 2TB Raid 1.
 - **Red:** 1000Mb/s
 - **Tráfico:** Ilimitado
 - **Coste mes:** €22.6891
 - **Coste anual:** €272,9692
- **tfg3.intelligenia.com:** El servidor cuenta con las siguientes características hardware:
 - **CPU:** 8 cores.
 - **RAM:** 16GB
 - **Almacenamiento:** 2 SSD 2TB Raid 1.
 - **Red:** 1000Mb/s
 - **Tráfico:** Ilimitado
 - **Coste mes:** €22.6891
 - **Coste anual:** €272,9692

Sumando ambos costes, tendríamos un coste anual de infraestructura de **1047,75** euros. Destacar que este sería el coste únicamente de la infraestructura donde se va alojar todo el proyecto. Más adelante se detallará el coste en recursos humanos que ha supuesto configurar estos servidores para que ofrezcan servicios parecidos a los que ofrecen los grandes proveedores de Cloud.

AWS

Amazon Web Services es una de las plataformas de computación en la nube más usadas en el mundo. AWS ofrece una gran cantidad de servicios, recursos de cómputo en la nube, almacenamiento, bases de datos, herramientas de administración, seguridad, dispositivos móviles...

Actualmente Amazon ofrece múltiples soluciones DevOps para empresas, sin embargo no todas las compañías se pueden permitir los costes, que aún son elevados. AWS ofrece:

- IaaC
- AWS OpWorks
- AWS CodeDeploy
- AWS CodePipeline
- AWS CodeCommit
- AWS Elastic Beanstalk
- Security

Azure

Azure es la gran competidora de AWS, ofreciendo múltiples servicios en la nube. También disponen de servicios DevOps [5] para empresas, una solución integral y muy completa. Sin embargo deberíamos contratar el hosting para las aplicaciones a parte, ya que este servicio únicamente es para ejecutar cauces de CI/CD. Una vez más la única limitación es el poder adquisitivo de una empresa, que en caso de pequeñas compañías no lo pueden afrontar. Azure DevOps ofrece:

- Azure Boards
- Azure Pipelines
- Azure Repos
- Azure Artifacts (CI/CD)
- Azure test plans

Google Cloud

A igual que las dos anteriores, Google Cloud ofrece servicios DevOps para empresas a precios similares [6]. De igual manera el servicio de hosting para las aplicaciones va a parte y ofrece servicios como:

- Control de versiones
- CI
- Automatización de implementación
- Desarrollo basado en troncales
- Automatización de pruebas
- Arquitectura
- Seguridad

1.5. Objetivos del TFG

E sta sección se va a encargar de definir los objetivos propuestos para este proyecto, sirviendo como primera aproximación para los requisitos funcionales. A continuación se redactan los objetivos propuestos en un estado inicial del proyecto y qué nivel de cumplimiento de éstos se ha logrado.

1.5.1. Objetivos propuestos

E l objetivo principal de este proyecto es dar una solución integra y completa para gestionar la infraestructura de una pequeña empresa adoptando una filosofía DevOps.

- **Objetivo 1:** *Automatizar despliegue de sistema de virtualización en servidores bare metal.* Dados los servidores bare metal, poder desplegar algún sistema de virtualización para gestionar máquinas virtuales donde poder crear los servicios necesarios para la infraestructura.
- **Objetivo 2:** *Automatizar creación de esquema Firewall - Proxy - WebServer.* Dado el objetivo de ofrecer servicios a clientes, tenemos que securizar los servidores con Firewalls y necesitamos Proxys y Web-servers para servir las aplicaciones finales.
- **Objetivo 3:** *Automatizar creación de servicios necesarios para poder ofrecer servicios propios de hosting.* Crear cluster de almacenamiento distribuido, cluster donde alojar las aplicaciones...

- **Objetivo 4:** *Automatizar la creación de los servicios necesarios para poder adoptar un enfoque DevOps para el desarrollo de aplicaciones.* Uno de los objetivos principales de este proyecto es crear y configurar los cauces de integración continua y entrega continua y aplicarlo al desarrollo de aplicaciones para la compañía.

1.5.2. Grado de cumplimiento de los objetivos propuestos

Capítulo 2

Análisis del sistema

2.1. Definición del problema

Hoy en día, aún hay muchas compañías que siguen teniendo dos equipos diferenciados dentro el departamento de IT. Sistemas o Operaciones y Desarrolladores. Ya hemos visto en la introducción que esto a la larga causa problemas y no es la forma más rápida y eficiente de desarrollar software. Trabajo como Administrador de Sistemas en una empresa pequeña y continuamente aparecen problemas. Es por eso que se ha decidido adoptar un enfoque DevOps para el desarrollo del software. En este cambio están implicados ambos departamentos ya que es responsabilidad de ambos que los cauces de desarrollo y despliegue funcionen correctamente.

En mi compañía la Infraestructura donde se alojan los proyectos, se configura manualmente haciendo prácticamente imposible una replicación de ésta en caso de que sea necesario. También las subidas de proyectos y ejecución de tests no están automatizados y cada desarrollador se encarga de ejecutar los tests en local. Ésto implica discrepancias varias entre el equipo de operaciones y el de desarrolladores. Es por esto que se hace necesario la adopción de técnicas DevOps como IaaC, CI y CD en la empresa.

2.2. Que se pretende resolver

- Todos los problemas que involucra la configuración manual de infraestructura.
- Ralentización de desarrollo de software, al no tener los cauces automatizados.

2.3. Casuísticas que se dan y como se resuelve cada una

- **Problema 1:** *Recreación ante fallo total o réplica de infraestructura.* Ante un fallo total de la infraestructura o replicación para propósitos de testing, con los sistemas actuales sería imposible, ya que dada la configuración manual de cada uno de los servidores, sería imposible replicar al 100
- **Solución 1:** Al adoptar IaaC, la infraestructura estará codificada en ficheros de configuración bajo control de versiones. Pudiendo recrear la infraestructura bajo demanda en cualquier momento y asegurándonos de que sea la misma al 100
- **Problema 2:** *Diferencias entre servidor de desarrollo y producción.* Al desarrollar en local los desarrolladores, no tienen una réplica de los servidores de producción para poder probar sus cambios. Ésto hace que haya discrepancias entre Sistemas y desarrolladores puesto que puede funcionar en local con una configuración dada pero no en producción.
- **Solución 2:** Al estar la infraestructura de cada aplicación codificada y bajo control de versiones, será sencillo replicar el entorno de producción en un entorno test para el desarrollo de aplicaciones.
- **Problema 3:** *Petición de subidas a producción.* Al no existir cauces de integración o despliegue, no se tiene un conocimiento exacto de qué comportamiento va a tener la aplicación en producción. Ésto y la necesidad de hacer peticiones al equipo de sistemas para subir nuevas versiones a producción ralentizan el desarrollo del software.
- **Solución 3:** Al tener el desarrollador un servidor réplica de producción para desplegar las aplicaciones, sabe perfectamente el comportamiento que va a tener en producción, puesto que son entornos idénticos. Ésto y los cauces CI-CD solucionan el problema.

2.4. Tipos de usuario

A adoptando la filosofía DevOps, tanto desarrolladores como sysadmin debería adoptar el mismo rol. Sin embargo, en una empresa en la cual crean y configuran sus propios servidores sin contratar servicios externos (como Azure DevOps, AWS...) se crea la necesidad de crear y configurar la infraestructura que va a alojar toda la infraestructura necesaria para el desarrollo cada aplicación. Es por esto que se distinguen entre dos tipos de usuarios: **Desarrolladores y Sistemas + Desarrolladores**.

- **Desarrolladores.** Este tipo usuario tiene las siguientes necesidades:
 - Poder construir una infraestructura para el desarrollo de aplicaciones bajo demanda. [RF 1]
 - Poder integrar cambios a las aplicaciones. [RF 2]
 - Poder tener feedback de el estado de la aplicación tras el despliegue. [RF 3]
 - Conocer la configuración del servidor donde se aloja la aplicación web. [RF 4]
- **Desarrolladores + Sistemas**
 - Poder construir una infraestructura para el desarrollo de aplicaciones bajo demanda. [RF 1]
 - Poder integrar cambios a las aplicaciones. [RF 2]
 - Poder tener feedback de el estado de la aplicación tras el despliegue. [RF 3]
 - Conocer la configuración del servidor donde se aloja la aplicación web. [RF 4]
 - Poder desplegar la infraestructura completa que aloja la infraestructura para las aplicaciones. [RF 5]
 - Poder conocer la configuración de la infraestructura. [RF 6]

Estos historias de usuario representan la funcionalidad final del proyecto, con un nivel de abstracción muy alto. En la siguiente sección, se desgranarán estas historias para definir de forma precisa los requisitos funcionales. [7]

2.5. Especificación de requisitos

A continuación se detallan los requisitos funcionales de este proyecto.

- Construir y desplegar infraestructura para el desarrollo de aplicaciones bajo demanda. [RF 1]
- Poder integrar cambios en las aplicaciones de forma autónoma [RF 2]
- Conocer el estado de la aplicación una vez hecho el despliegue [RF 3]
- Conocer la configuración del servidor donde se aloja la aplicación web. [RF 4]

- Poder desplegar la infraestructura completa que aloja la infraestructura para las aplicaciones a través de ficheros de configuración. [RF 5]
- Conocer en todo momento la configuración del servidor [RF 6]
- Poder desplegar la infraestructura rápidamente ante un error catastrófico [RF 7]
- Realizar tareas de mantenimiento en los servidores con tareas automatizadas [RF 8]
- Crear máquinas virtuales bajo demanda con tareas automatizadas y ficheros de configuración [RF 9]
- Modificar la configuración de los firewall con tareas automatizadas [RF 10]

2.6. Diagramas

2.6.1. Diagramas casos de uso

A continuación se muestran los principales diagramas según los requisitos funcionales descritos en la sección Especificación de requisitos.

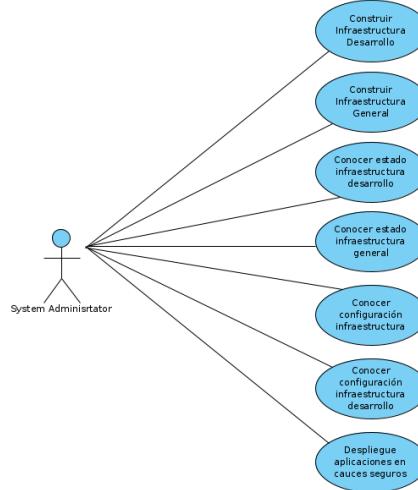


Figura 2.1: Casos de uso [8]

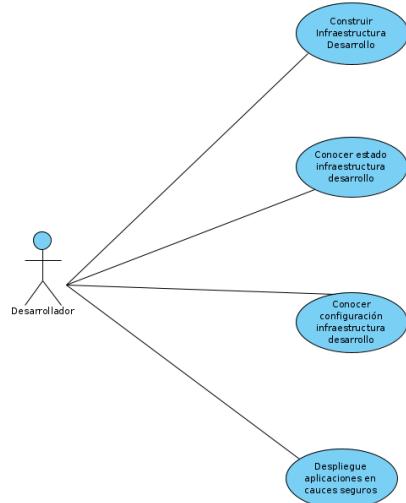


Figura 2.2: Casos de uso [8]

2.6.2. Diagramas de secuencia

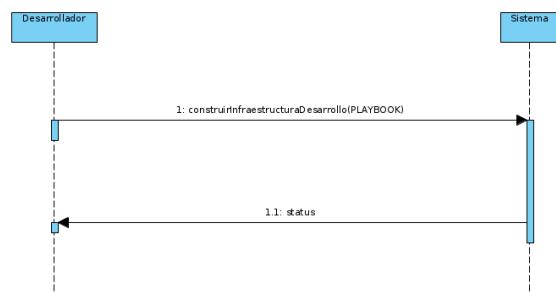


Figura 2.3: Diagrama secuencia Desarrollador 1 [9]

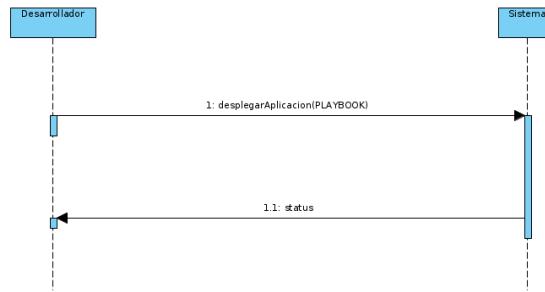


Figura 2.4: Diagrama secuencia Desarrollador 2 [9]

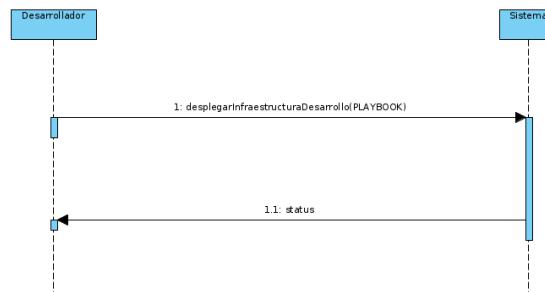


Figura 2.5: Diagrama secuencia Desarrollador 3 [9]

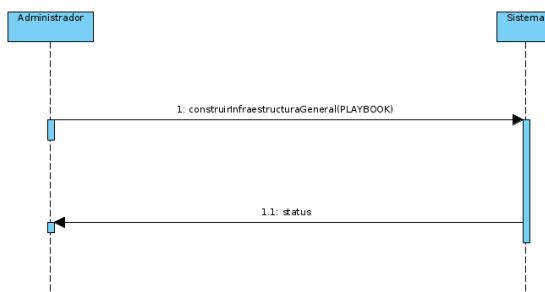


Figura 2.6: Diagrama secuencia Administrador 1 [9]

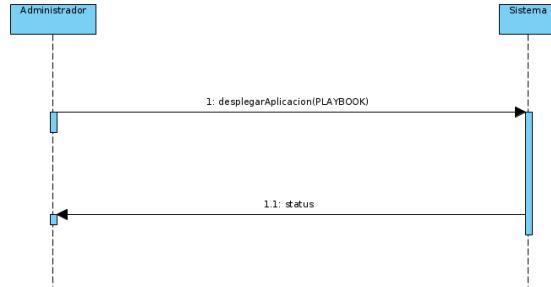


Figura 2.7: Diagrama secuencia Administrador 2 [9]

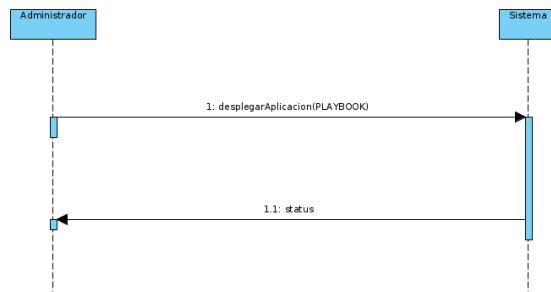


Figura 2.8: Diagrama secuencia Administrador 3 [9]

2.7. Arquitectura del Sistema

2.7.1. Servidores Físicos

A al trabajar en una empresa de hosting, he tenido la suerte de contar con 3 servidores bare metal para el desarrollo de este proyecto. Las características técnicas de los servidores se pueden consultar en [Servidores Bare Metal](#).

2.7.2. Infraestructura objetivo

Este proyecto pretende crear una infraestructura robusta para una pequeña empresa que se dedique al desarrollo del software. Ésta infraestructura debe ser robusta al igual que segura, con lo que ha de proporcionar firewalls redundantes y algún mecanismo para proporcionar alta disponibilidad en las aplicaciones web. A continuación se muestra la infraestructura objetivo.

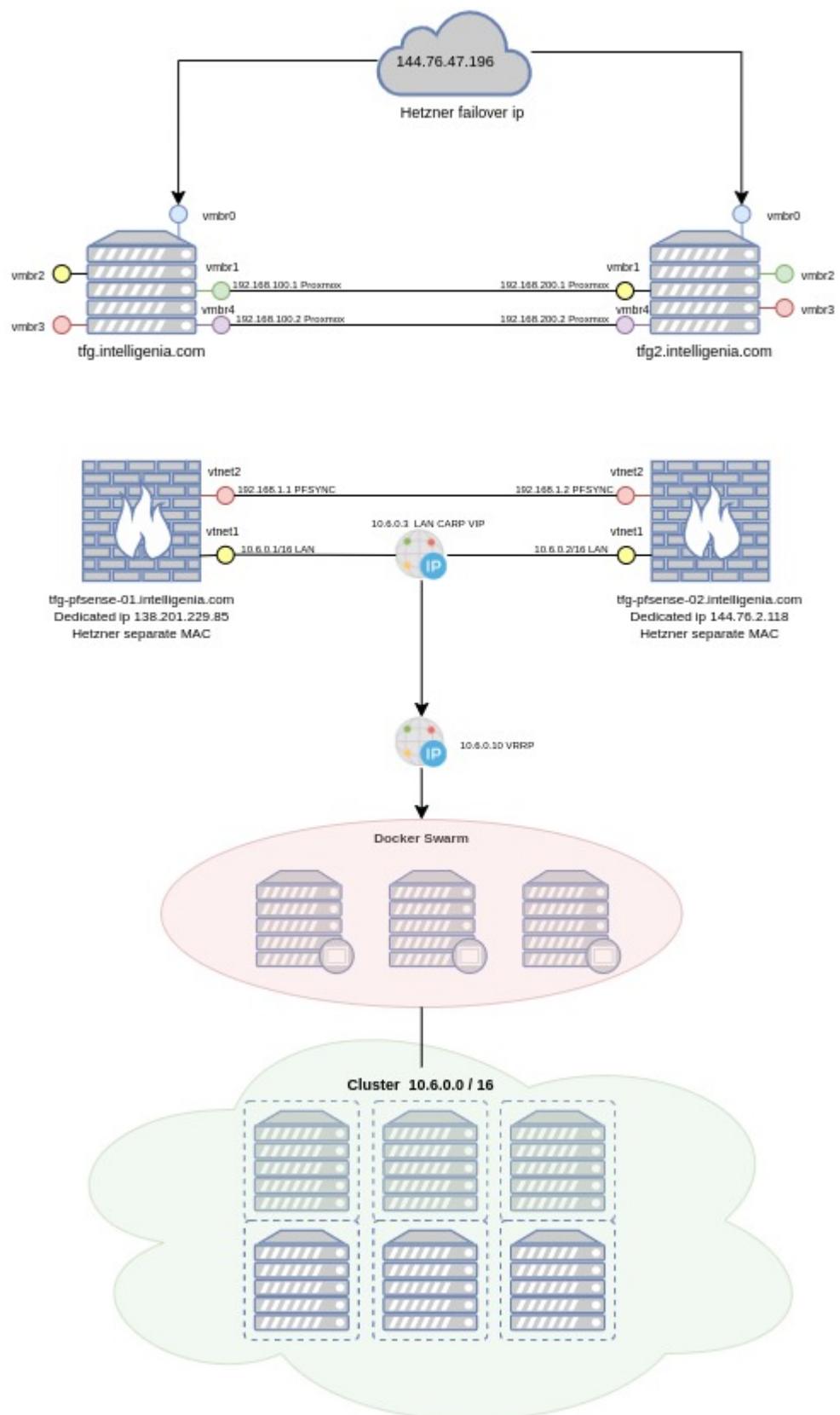


Figura 2.9: Infraestructura Objetivo

2.8. Metodología de desarrollo

Todo proyecto software debe tener una organización y unas etapas de desarrollo bien definidas. En esta sección se pretende explicar la metodología de desarrollo elegida para realizar este proyecto.

Se ha tratado este proyecto como cualquier otro proyecto software. Para la organización y el control de versiones se ha elegido Github, un software basado en git originalmente creado para el control de versiones. Actualmente, GitHub ofrece múltiples servicios, como almacenamiento, gestión de paneles de trabajo, registry, integración con múltiples tecnologías...

En cuanto a la metodología de desarrollo, se ha optado por un desarrollo basado en Milestones. Cada Milestone está compuesto por Issues y estos están etiquetados y asignados a personas. A continuación se explican con mayor detalle estos conceptos.

2.8.1. Milestones

Los Milestones o Hitos en castellano, corresponden con estados finales deseados de la aplicación. Sabiendo esto, podríamos crear un Milestone por ejemplo: "Servidores configurados a través de ficheros de configuración Ansible". Esto será un estado final deseado para nuestra aplicación o proyecto. Para que un hito quede totalmente realizado, deben estar completos todos los issues marcados como esenciales para el hito. Un hito está compuesto por issues. A continuación se muestran algunos milestones creados en este proyecto, en el panel de administración GitHub.

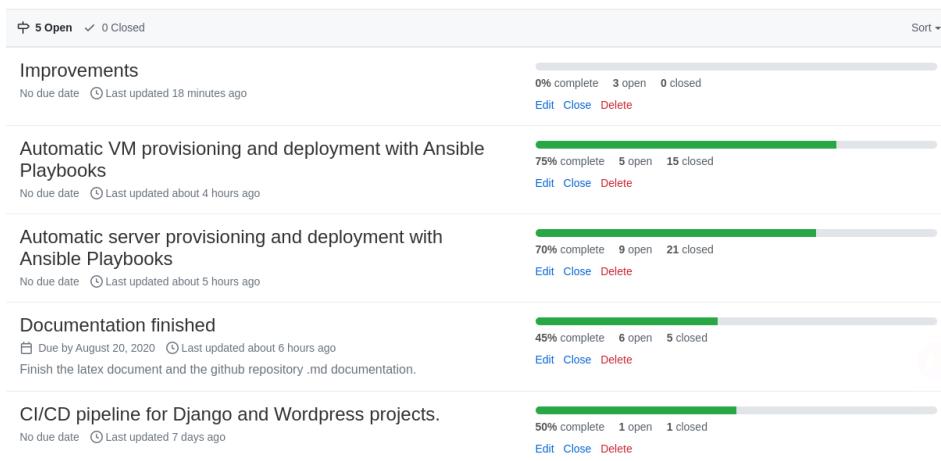


Figura 2.10: GitHub milestones

2.8.2. Issues

Como ya hemos visto, los issues forman parte de los hitos. Es una forma de desgranar el problema. Siguiendo el ejemplo anterior, si tenemos un hito: "Servidores configurados a través de ficheros de configuración Ansible", podemos desgranar el siguiente en distintos issues, que serían tareas más sencillas que hay que realizar para completar el hito. Por ejemplo, algunos issues serían:

- Crear estructura directorios Ansible.
- Instalar paquetes en servidor a través de ficheros de configuración Ansible. **Core**
- Configurar interfaces de red a través de ficheros de configuración Ansible. **Core**
- Instalar ISO en servidor a través de ficheros de configuración Ansible. **Core**
- Instalar certificados SSL a través de ficheros de configuración Ansible. **Mejora**

Y así seguiríamos creando issues según creamos que van a ser necesarios para completar el hito en cuestión.

En la sección Milestones hemos hablado que los issues tienen etiquetas. En la lista anterior por ejemplo, únicamente tenemos dos etiquetas que nos indican en este caso si son issues imprescindibles para el hito o simplemente mejoras. Gracias a estas etiquetas, podemos distinguir entre distintos tipos de issues y asignar mayor o menos prioridad por etiqueta. También gracias a Github, cada issue puede ser asignado a un desarrollador del proyecto. A continuación se muestra un ejemplo del panel de Github para mostrar los issues, etiquetas y hitos.

Como se puede comprobar, el sistema de etiquetas y asignación de issues a desarrolladores, es más que suficiente para manejar proyectos. Permite asignar prioridades, agrupar issues en hitos y escribir comentarios en cada issue / milestone.

VictorMorenoJimenez / tfg2020

Code Issues 24 Pull requests Actions Projects Wiki Security Insights Settings

Filters is:issue is:open Labels 13 Milestones 5 New issue

	Author	Label	Projects	Milestones	Assignee	Sort
24 Open	63 Closed					
Add tags to playbooks, in order to execute only one part. Improvement	#87 opened 22 minutes ago by VictorMorenoJimenez	Improvements				
Add Software Development process to docs. documentation	#83 opened 2 days ago by VictorMorenoJimenez	Documentation f...				
Check general documentation structure	#82 opened 2 days ago by VictorMorenoJimenez	Documentation f...				1
Review docs typos documentation	#81 opened 2 days ago by VictorMorenoJimenez	Documentation f...				
On 1.4.2 add cloud providers budget to compare documentation	#79 opened 2 days ago by VictorMorenoJimenez	Documentation f...				
On deploy_pfSense_firewall playbook, add option to add additional firewall rules with easyrule Improvement	#74 opened 5 days ago by VictorMorenoJimenez	Improvements				1
On task create_vm on role proxmox_kvm, check if VM is already started before start vm Improvement	#73 opened 5 days ago by VictorMorenoJimenez	Improvements				
Certbot fails if you request more than 5 renewals a week. external behaviour	#70 opened 6 days ago by VictorMorenoJimenez	Automatic serve...				
Clean ansible folder structure enhancement	#69 opened 6 days ago by VictorMorenoJimenez	Automatic serve...				
Create and configure galera cluster with Ansible playbooks core	#68 opened 6 days ago by VictorMorenoJimenez	Automatic VM p...				
Create docker swarm with Ansible playbooks core	#64 opened 7 days ago by VictorMorenoJimenez	Automatic VM p...				
Add broker vm to cluster. enhancement	#63 opened 7 days ago by VictorMorenoJimenez	Automatic VM p...				
Can't add hosts to ceph cluster. cephadm bug	#56 opened 8 days ago by VictorMorenoJimenez	Automatic VM p...				
End project objectives, 1.5 section core documentation	#52 opened 9 days ago by VictorMorenoJimenez	Documentation f...				
On proxmox role, configure_ssl_letsencrypt task, ansible copy module not working as expected. bug enhancement	#50 opened 9 days ago by VictorMorenoJimenez	Automatic serve...				
On create_vm_from_backup task on role proxmox, pfSense has 4 interfaces. enhancement	#49 opened 9 days ago by VictorMorenoJimenez	Automatic VM p...				
Add task on configure_bare_metal that deploys public_key to other hosts on cluster core	#47 opened 9 days ago by VictorMorenoJimenez	Automatic serve...				
Check if pdflatex exists before executing doc_gen Makefile's rule. enhancement	#41 opened on 29 Jun by VictorMorenoJimenez	Documentation f...				
Add node to Proxmox cluster via ssh not working. Proxmox issue. bug wontfix	#38 opened on 12 Jun by VictorMorenoJimenez	Automatic serve...				2
Change vars of manage_users task, improve list enhancement	#37 opened on 4 Jun by VictorMorenoJimenez	Automatic serve...				
Add option to join cluster instead of creating one to Proxmox role bug enhancement	#33 opened on 17 Apr by VictorMorenoJimenez	Automatic serve...				1
Add multiple fingerprints to Activate Hetzner role enhancement wontfix	#29 opened on 13 Apr by VictorMorenoJimenez	Automatic serve...				
Check inline vars like /etc/fool/{{ user }}" enhancement	#28 opened on 6 Apr by VictorMorenoJimenez	Automatic serve...				
Create test environment for a project with Ansible core	#27 opened on 3 Apr by VictorMorenoJimenez	CI/CD pipeline f...				

Figura 2.11: GitHub panel

Capítulo 3

Planificación

3.1. Estimación recursos necesarios

En esta sección se va a crear una estimación de los recursos, tanto humanos como económicos que se van a necesitar para llevar a cabo el proyecto. Cabe destacar que en la estimación temporal se incluye un lapso de tiempo para adaptarse a la tecnología a usar. Ésto no se ha indicado de forma explícita, pero va implícito en estimación temporal de ejecución de cada hito.

3.1.1. Estimación temporal

Para dar una estimación del tiempo requerido para el proyecto, vamos a utilizar un diagrama de Gantt, en el que incluiremos los principales hitos del proyecto y una estimación en semanas de la duración del mismo. Una estimación inicial ha sido de 24 semanas en total, a continuación se muestra el diagrama de Gantt.

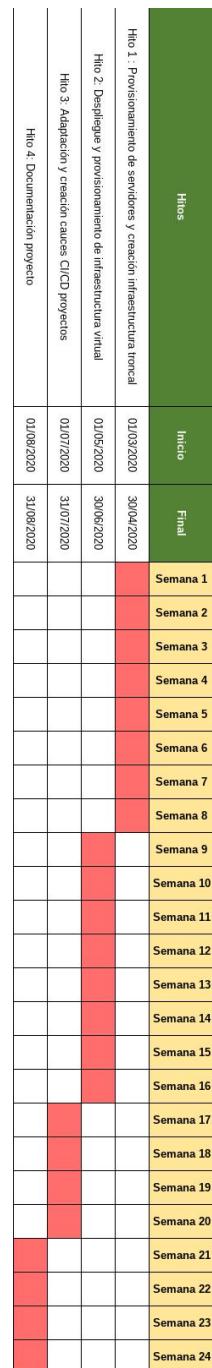
Diagrama Gantt

Figura 3.1: Diagrama Gantt [10]

3.2. Presupuesto

Para realizar este presupuesto, se ha partido del sueldo base de un graduado en Ingeniería Informática. Recalcar que en esta partida, se incluyen horas extra para formación en las tecnologías elegidas. Por otra parte se ha incluido en el presupuesto el coste de mantenimiento de la infraestructura.

Concepto	Precio / mes €	Importe Total €
Partida Personal		
Sueldo DevOps Junior	1.800	10.800
Partida Inventariable		
Coste infraestructura	61,34	368,04
Partida Fungible		
-	-	-
Partida servicios técnicos		
Mantenimiento	Incluido	0
Partida viajes y dietas		
-	-	-
Partida de otros		
Incluido	Incluido	0
Total	1861,34	11.168,04
<hr/> <hr/>		

Cuadro 3.1: Presupuesto [11]

Debe destacar que en esta partida únicamente se incluyen el coste de un empleado Junior y el coste de la infraestructura necesaria para desarrollar este proyecto. Nótese que en ningún caso, se están añadiendo costes de oficina ni de equipo para desarrollo. En una partida real, habría que incluir estos gastos ya que son imprescindibles para el proceso de desarrollo. En este presupuesto aparece como Otros.

Capítulo 4

Diseño

4.1. Solución adoptada

Con el fin de cumplir todos los objetivos definidos en la sección Objetivos propuestos se han elegido las siguientes tecnologías. Cabe destacar la influencia ejercida por la empresa para utilizar tecnologías ya existentes en la infraestructura actual como los firewall pfSense o el servicio de virtualización Proxmox. Sin embargo, son tecnologías con continuo desarrollo y muy aptas para desarrollar su papel dentro del cluster.

Todas las tecnologías que se van a mencionar a continuación son de código abierto y satisfacen las necesidades de este proyecto.

4.1.1. Proveedor Infraestructura

La infraestructura necesaria para formar el ecosistema que cumpla con los objetivos de este proyecto, se ha de alojar en servidores físicos. Para cumplir esta necesidad se ha elegido el proveedor Hetzner [12], uno de los proveedores de servidores bare metal más conocidos en Europa.

Hetzner

Hetzner posee tres grandes centros de datos, Suiza, Irlanda y Luxemburgo. Una de las principales ventajas que tiene Hetzner sobre otras compañías es el soporte 24/7 que ofrece en los servidores dedicados. Ante cualquier problema siempre tienes una línea abierta de soporte para solucionarlo. Ésto junto con la gran variedad de servidores de distintas gamas que ofrece, hace a Hetzner un candidato perfecto para contratar la infraestructura.



Figura 4.1: Hetzner Data Center [12]



Figura 4.2: Hetzner Building [12]

4.1.2. Servicio Virtualización

Para poder ofrecer todos los servicios descritos en Objetivos propuestos y cumplir con la arquitectura objetivo Infraestructura Objetivo, se hace necesario añadir una capa de virtualización a los servidores contratados. Estos 3 servidores físicos, servirán para desplegar los servicios descritos, sin embargo es necesario separar estos servicios en distintas máquinas. El servicio de virtualización nos permitirá crear tantas máquinas virtuales como servicios, virtualizando el hardware del servidor físico y aportando entornos cerrados, virtualizados y fácilmente recreables.

El servicio de virtualización elegido ha sido Proxmox VE [13], basado en KVM (Kernel-based Virtual Machine) y de código abierto. Se ha elegido este servicio de virtualización por necesidades de la empresa.

Proxmox

Proxmox VE es una solución completa para gestionar máquinas virtuales KVM. Entre sus servicios podemos destacar:

- **Hypervisor KVM.** Fácil acceso a las máquinas a través de una consola gestionada por el hypervisor KVM.
- **ISO.** Creación y gestión de máquinas virtuales tanto Windows como Linux.
- **Storage.** Software-defined storage. Podemos crear soluciones de almacenamiento con posibilidad de integrar tecnologías como CEPH o glusterfs.
- **Networking.** Solución a la gestión de redes desde un punto de vista software. Se pueden crear tantas interfaces virtuales como se necesiten.
- **Backups.** Gestión de backups completo de las máquinas.
- **User management.** Gestión de espacios de usuario así como restricción de permisos según tipo de usuario.
- **Firewall.** Firewall por máquina virtual y por cluster. El firewall se puede manejar por la interfaz web o por la API.
- **Proxmox VE API.** La interfaz web es una gran ayuda para el Administrador, sin embargo Proxmox también posee una extensa API [14] para poder realizar todo lo que se puede hacer desde la interfaz web, desde línea de comandos a través de su API con el comando **pvsh**

- **Proxmox Forums.** Al no ser la versión enterprise, Proxmox VE no dispone de soporte dedicado como tal, sin embargo posee una gran comunidad muy activa que responden a tus dudas en los foros [15].
- **Interfaz Web.** Gracias a su interfaz web, se hace muy sencilla la gestión de las distintas máquinas.

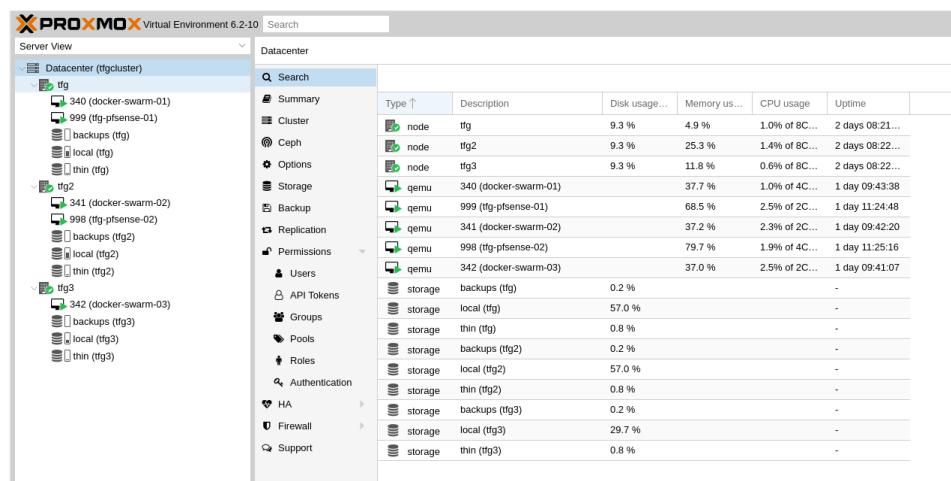


Figura 4.3: Proxmox WUI

4.1.3. Firewall

Hoy en día, con el auge de la informática y de las nuevas tecnologías, cada vez se están dando más y más ciberataques. Los denominados 'hackers' o delincuentes, aprovechan brechas de seguridad en los sistemas para poder hacerse con información sensible y después poder pedir rescate por dicha información. Estos ataques suelen aprovechar puertos abiertos, servicios obsoletos y vulnerabilidades conocidas para perpetrar sus ataques. Por ejemplo, esta web nos muestra en tiempo real la cantidad de ataques que sufre un país determinado [16]. Es por esto que se hace necesaria la protección de la infraestructura final, exponiendo únicamente los puertos necesarios para desviar el tráfico y trabajar con los servicios únicamente dentro de la red LAN segura.

Para este fin se ha elegido la tecnología pfSense [17]. PfSense dispone de su versión community gratuita y de código abierto. Esta tecnología se ha elegido ya que es la tecnología utilizada en el cluster de producción de la empresa.



Figura 4.4: Ciberataques [16]

pfSense

Basado en FreeBSD, este firewall se instala en una máquina virtual como un servicio más sirviendo como router/firewall según la configuración. Entre otros, los principales servicios que ofrecen pfSense son:

- **DHCP Server.** Un servidor DHCP totalmente configurable para asignar ya sea de forma dinámica o estática las IP a las máquinas del cluster.
- **DNS Resolver.** Posibilidad de configurar el servidor DNS. Forzar resolución de nombres, modificar servidores...
- **Firewall.** Por defecto deniega todo el tráfico menos el necesario para la interfaz web. Podemos definir reglas firewall dedicadas para cada interfaz. También podemos crear reglas NAT para redirigir el tráfico dentro de la red. Todo gestionado desde la interfaz web.
- **Virtual IP.** Al querer ofrecer alta disponibilidad, es necesario disponer de una ip virtual del tipo CARP (Common Address Redundancy Protocol). De esta forma podemos tener una IP virtual de acceso al cluster para los firewall. En caso de fallo del firewall principal, se modificará esta IP para que el pfSense secundario se haga cargo de dirigir el tráfico dentro del cluster a través de esta ip virtual.
- **Interfaces.** Creación y asignación de interfaces. A través de la interfaz de usuario o dentro de la propia consola de la máquina virtual, pode-

mos definir las distintas interfaces necesarias, asignarles ip y asociarlas con interfaces físicas del anfitrión.

- **Package Manager.** La interfaz web posee un gestor de paquetes muy útil que nos ofrece una gran variedad de paquetes que nos facilitarán el uso de pfSense. Paquetes como tinc, openvpn, HAproxy que no vienen integrados por defecto en pfSense pero se pueden añadir gracias a este gestor de paquetes.
- **VPN.** Gestión de redes VPN para realizar conexiones con equipos fuera del cluster. Por ejemplo esto ha resultado muy útil para poder conectar el equipo de desarrollo con las distintas máquinas virtuales del cluster a través de una VPN. Esto nos permite no exponer ningún puerto y poder tener acceso.
- **Services Status.** Dentro de la interfaz web, disponemos de un panel informativo con el estado de cada uno de los servicios gestionados por pfSense. A través de este pequeño panel de control, podemos comprobar el estado, reiniciar o parar un servicio.
- **Web Interface.** Esta es la parte más atractiva de pfSense, ya que podemos configurarlo de una forma intuitiva y a través de una interfaz web.

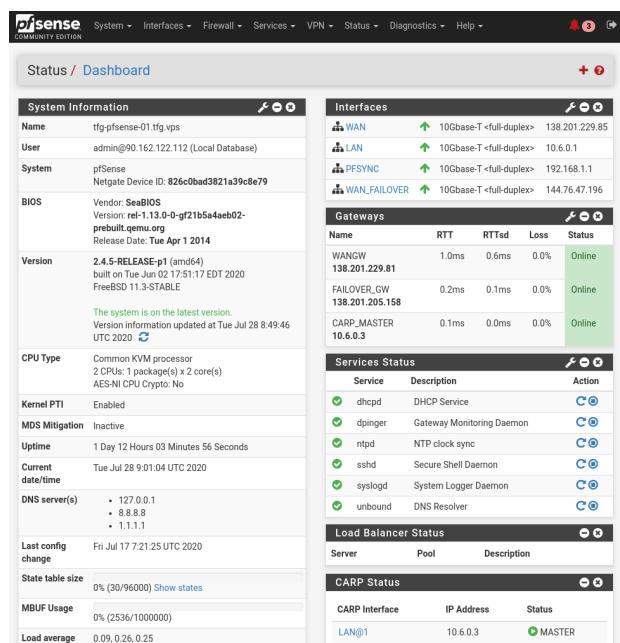


Figura 4.5: pfSense interfaz web

4.1.4. Contenedores

Como ya hemos visto, Proxmox nos proporciona una capa de virtualización del hardware de los servidores para poder crear máquinas virtuales donde alojar los distintos servicios.

Sin embargo, a la hora del desarrollo y despliegue de aplicaciones web en el cluster, se hace inviable crear una máquina virtual por cada una de las aplicaciones web creadas. Esto sería insostenible ya que las máquinas virtuales demandan recursos.

Para solventar este problema se van a crear aplicaciones en contenedores. Cada aplicación tendrá uno o varios contenedores dependiendo de los servicios, normalmente 1 servicio 1 contenedor. Para realizar este proceso se ha elegido la tecnología Docker. [18].

Docker

Docker es la tecnología más famosa para crear contenedores. Gracias a Docker podemos aislar las aplicaciones en contenedores seguros y fácilmente replicables. Se ha elegido Docker por su extensa comunidad y porque es la tecnología más utilizada para este fin. Sin embargo, con esto no es suficiente ya que, una vez tengamos la aplicación separada en contenedores, necesitamos un entorno donde lanzar estos contenedores y poder gestionarlos. Es aquí donde surgen los orquestadores de contenedores. A continuación se hace una pequeña a dos orquestadores: Docker Swarm [19] y Kubernetes [20].

4.1.5. Orquestación contenedores

Kubernetes

Kubernetes es un software de orquestación de contenedores que nos permite automatizar el proceso de despliegue, escalado y gestión de las aplicaciones en contenedores. Kubernetes se ha convertido en la solución principal para grandes empresas por su robustez. La principal ventaja de Kubernetes frente a sus competidores es la posibilidad de modificar las condiciones de escalado vertical o horizontal de los contenedores sin afectar a la disponibilidad de la aplicación. Su gran desventaja es la complejidad, ya que para poder manejar de forma eficiente un cluster de Kubernetes debemos pasar por un largo camino de aprendizaje.

Se ha elegido esta tecnología por su gran demanda en el sector y como proceso de aprendizaje. Esta tecnología no estaba implementada en la empresa, sin embargo, se va a introducir como orquestador adicional.

Docker Swarm

Docker Swarm es un software de orquestación de contenedores. Es la solución nativa de Docker para la gestión de los contenedores. Docker Swarm nos permite automatizar los procesos de despliegue escalado y gestión de contenedores. Sin embargo, cuando hablamos de escalado vertical o horizontal, Docker Swarm, al contrario que Kubernetes, no permite este escalado asegurando la disponibilidad. Debemos relanzar los contenedores con los nuevos parámetros para el escalado.

Se ha elegido esta tecnología por ser la usada en un principio en la empresa para la gestión de contenedores. Cabe destacar que, aunque no ofrece los mismos servicios que Kubernetes, Docker Swarm es una solución muy buena para la orquestación de contenedores y su principal ventaja frente a Kubernetes es la facilidad para desplegar y gestionar los contenedores.

4.1.6. Almacenamiento estático

Si bien tanto Docker Swarm como Kubernetes nos permiten gestionar los contenedores, estas tecnologías están pensadas para aplicaciones stateless o sin estado [21]. A grandes rasgos, esto quiere decir que tanto Kubernetes como Docker Swarm, funcionan bien con aplicaciones que no necesitan almacenar información para su correcto funcionamiento.

Para esto, Docker ofrece una solución en forma de volúmenes. Sin embargo, esta solución no se adapta a las necesidades de un entorno de alta disponibilidad y alta fiabilidad. Es por esto que surge la necesidad de crear alguna solución de almacenamiento distribuido que nos proporcione dichas características. La solución elegida ha sido Ceph [22].

Ceph

Ceph es un software de código abierto que proporciona almacenamiento distribuido. Ceph ofrece almacenamiento a nivel de bloque, objeto y a nivel de fichero.

En nuestro caso particular, nos vamos a beneficiar de CephFS o Ceph File System, que está implementado sobre el sistema de almacenamiento a nivel de bloque creado por Ceph. Las principales ventajas de Ceph son:

- Seguridad de la información
- Almacenamiento ilimitado por sistema de ficheros.
- Balanceo de carga.
- Alta disponibilidad y alta fiabilidad.

- Interfaz web. Ceph nos ofrece la opción de gestionar el cluster desde una interfaz web. Aunque las opciones son limitadas ya que está bajo desarrollo, facilita en gran medida la gestión del cluster.

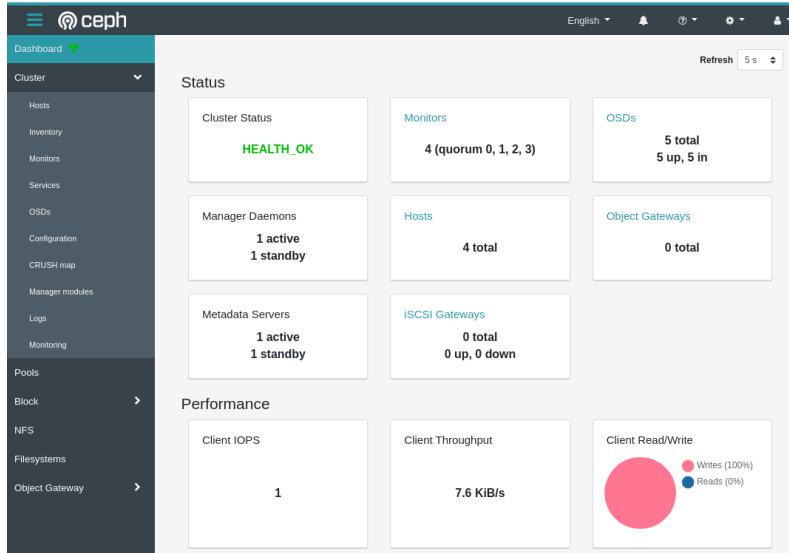


Figura 4.6: CEPH interfaz web

Se ha elegido esta tecnología por ser la más completa de las opciones de código abierto.

Aclaraciones Ceph

Ceph es una tecnología compleja, para abordar de forma correcta este documento se cree necesaria la aclaración de cómo funciona Ceph así como de los servicios que involucra:

4.1.7. Servidor Web / Proxy

Para el correcto acceso de los clientes y desarrolladores a las aplicaciones, se hace necesario un proxy que desvíe las peticiones a la aplicación correcta. Para este fin se ha elegido la tecnología Nginx [23], que sirve a la vez de servidor web y de proxy. Un proxy es esencial para asegurar la seguridad del cluster, ya que únicamente hay un punto de entrada protegido por un proxy que a su vez está protegido por el firewall. De esta forma se aumenta la seguridad y se dificultan los intentos de ataque masivos.

Nginx

Nginx, es uno de los servidores web más utilizados en el mundo por su facilidad en el uso así por su versatilidad, puede ser utilizado como servidor web, proxy o balanceador.

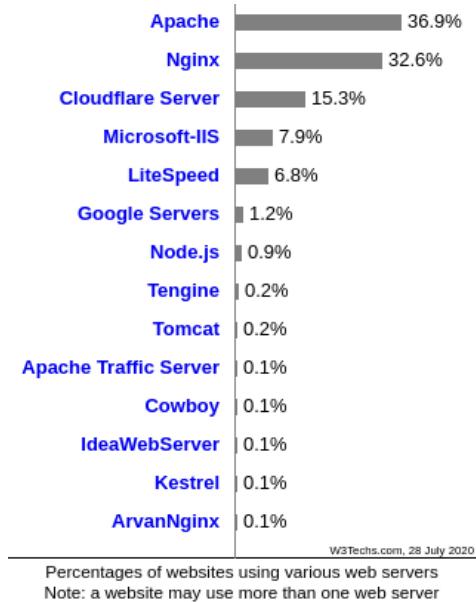


Figura 4.7: Estadísticas Servidores [24]

Se ha elegido Nginx por ser la solución adoptada por la empresa en primer lugar, no se ha propuesto utilizar otra tecnología ya que esta satisface las necesidades de este proyecto.

4.1.8. Control de versiones

Para el correcto desarrollo del software y en este caso de la infraestructura, es necesario disponer de un servicio de control de versiones donde subir el código y llevar un seguimiento del mismo. Esto es muy importante, ya que cualquier cambio en el código queda reflejado y hacer rollbacks a versiones anteriores es posible. Para este fin se ha elegido la herramienta basada en git GitLab [25].

GitLab

GitLab es un software basado en Git, que proporciona una interfaz web para la gestión de proyectos, usuarios y ofrece herramientas para todo el

ciclo DevOps. Integración continua, despliegue continuo, registry para contenedores... GitLab ofrece una versión Community gratuita que se puede instalar en servidores propios. GitLab ofrece múltiples servicios así como integración con servicios externos.

Se ha elegido GitLab por ser la solución adoptada en un momento inicial por la empresa así como por ofrecer la posibilidad de instalar GitLab de forma gratuita en los servidores y administrar el servicio por completo.

4.1.9. Base de datos

Las aplicaciones lanzadas tanto en Docker Swarm como en Kubernetes, necesitarán almacenar el estado de la aplicación así como datos necesarios para el correcto funcionamiento de esta. Estos datos deben ser persistentes y no borrarse ante el reinicio de los contenedores o re-despliegue. Para esto, es necesario tener una base de datos que nos proporcione almacenamiento para las aplicaciones. La tecnología elegida para esta causa ha sido MariaDB, ya que las aplicaciones necesitan bases de datos relacionales y MariaDB está licenciado por GNU General Public License, así que lo podemos usar sin problemas. También se ha elegido esta tecnología porque es con la que más familiarizados estamos todos en la empresa.

Para garantizar la alta disponibilidad y alto rendimiento a la hora de recuperar datos, se ha optado por la creación de un cluster de Galera [26] con 3 nodos, uno en cada nodo principal de Proxmox.

Galera Cluster

Galera cluster es una tecnología de replicación de bases de datos MariaDB para InnoDB. Galera nos ofrece una solución síncrona con replicación de datos y resincronización automática. Todos los nodos de galera, tienen la información actualizada en tiempo real con lo cual, en caso de caída de alguno de los nodos, la información estaría disponible en cualquiera los otros nodos. La facilidad para crear un cluster de galera es una gran ventaja frente a otras soluciones, ya que se puede crear un cluster con el paquete de mysql-server que viene por defecto en distribuciones debian.

Galera cluster nos permite el re-escalado del cluster, tanto aumentando la cantidad de nodos maestros o esclavos como reducir el número de nodos. Esto es genial ya que en caso de aumento de la demanda, o incremento en el número de aplicaciones se puede aumentar el cluster de galera sin problema de forma sencilla ya que viene con una herramienta integrada para ello. Cabe destacar que, no hay que hacer ninguna configuración especial para adaptar las aplicaciones, ya que, galera cluster es visto por las aplicaciones como un nodo normal de MariaDB.

4.2. Diseño cluster

En la sección Infraestructura objetivo se muestra un diagrama de la Infraestructura General diseñada para este proyecto. Un esquema Firewall - Proxy - Webserver. El objetivo de esta sección es reducir un nivel de abstracción y mostrar el diseño del Docker Swarm y del Cluster, rodeados por un círculo rojo en el caso de Docker Swarm y por una nube verde en el caso del cluster.

4.2.1. Docker Swarm

Como se puede observar en el siguiente diagrama, cada aplicación estará separada en distintos contenedores dentro del docker swarm. En este caso, para exemplificarlo, se ha elegido una aplicación que tiene 3 contenedores. Backend, frontend y un contenedor que se encargará de encolar tareas Celery. De esta forma, al lanzar la aplicación el cluster de docker swarm, docker swarm se encargará de replicar los contenedores y de asegurar la disponibilidad de los servicios.

Sin embargo, lo ideal sería poder alojar en el swarm, únicamente el proxy y que este se encargue de balancear el tráfico hacia otro cluster donde se alojan las aplicaciones, ya sea otro swarm o un cluster de Kubernetes.

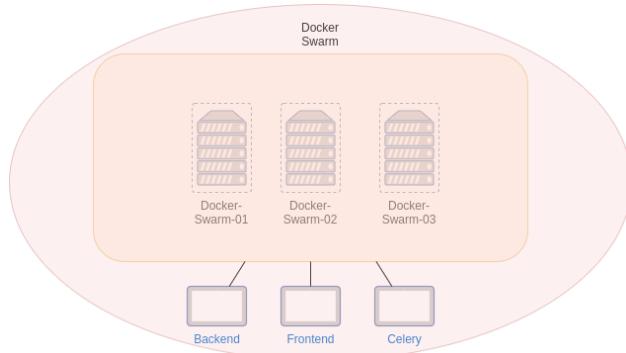


Figura 4.8: Diseño Docker Swarm

4.2.2. Cluster

A continuación, bajando el nivel de abstracción, se explican los principales servicios que componen el cluster. Esta parte corresponde con la nube verde llamada cluster en la sección Infraestructura objetivo.

- **K8 cluster.** Kubernetes cluster formado por 4 nodos. Un nodo administrador y 3 workers. Este cluster servirá para lanzar aplicaciones en

contenedores Docker. Es una mejora al cluster de docker swarm.

- **Ceph cluster.** Cluster de CEPH formado por 4 nodos. Un nodo administrador y otros 3 nodos. En cada nodo habrá los siguientes servicios de CEPH ¹

- **ceph-admin:** ceph_mgr, ceph_mon
- **ceph-node-01:** ceph_mds, ceph_mon, ceph_osd
- **ceph-node-02:** ceph_mds, ceph_mon, ceph_osd
- **ceph-node-03:** ceph_mgr, ceph_mon, ceph_osd

Este cluster nos proporcionará almacenamiento estático para las aplicaciones así como alta disponibilidad y tolerancia a fallos de hasta 2 nodos.

- **Virtualmin.** Virtualmin [27] es un servicio que proporciona hosting web a través de un panel de administración web. Este servicio lo utilizaremos para las aplicaciones Wordpress de la empresa.
Virtualmin nos ofrece todo lo necesario para gestionar servidores virtuales para cada una de las aplicaciones Wordpress.
- **Icinga.** Sistema de monitorización del cluster. Icinga2 es un software de monitorización basado en nagios de código abierto. Surgió de un fork de Nagios en 2009. Encaja en nuestra infraestructura para monitorizar todos los servicios.
- **Gitlab.** Control de versiones, véase sección Control de versiones.
- **Rabbit MQ.** Broker de mensajería. Software de código abierto que sirve como middleware de mensajería, implementa el estándar Advanced Message Queuing Protocol.
Será el encargado de tramitar los mensajes lanzados en el cluster por Celery.

¹Servicios de CEPH explicados en sección Aclaraciones Ceph

Diagrama Cluster

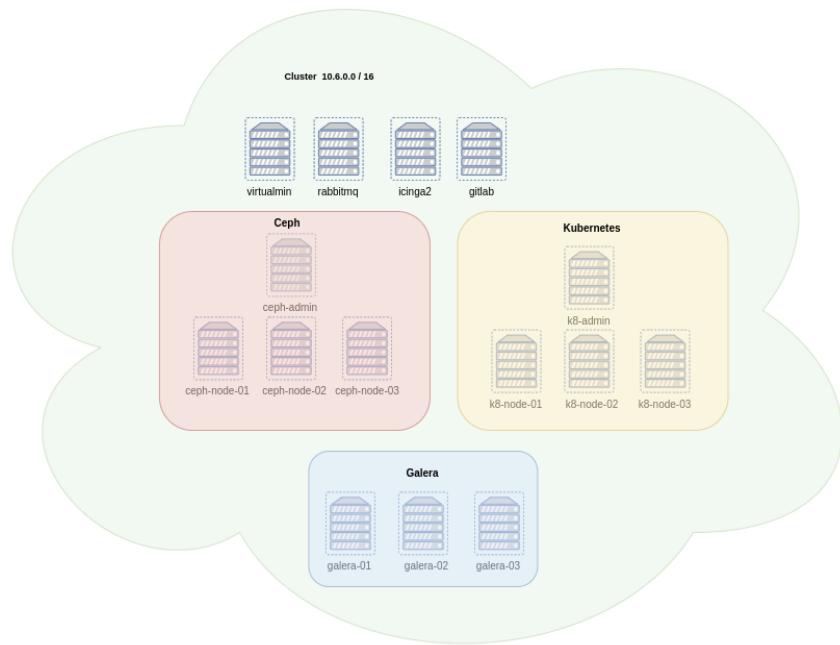


Figura 4.9: Diseño Cluster

4.3. Posibles mejoras

El proyecto que describe este documento, es una solución integral para resolver el problema de infraestructura en una pequeña empresa. Si hablamos de posibles mejoras, ya que siempre se puede mejorar, hablaríamos de hacer más eficientes los playbooks de Ansible o estructurarlos de forma que sean más comprensibles. Cabe destacar que en el diseño del cluster se incluye un cluster de Kubernetes, el cual está justamente para mejorar el cluster de docker-swarm.

Un punto importante de mejora sería implementar las técnicas de integración continua y entrega continua a las aplicaciones Wordpress de Virtualmin, ya que en este proyecto únicamente se abordan los proyectos Django + Angular de la empresa.

A grandes rasgos y exceptuando pequeñas mejoras en la calidad del código o la documentación, pienso que el proyecto es muy ambicioso y está bien diseñado a nivel de infraestructura, no necesitando grandes cambios en un futuro próximo.

Capítulo 5

Conclusiones

5.1. Futuro del proyecto

A continuación se hace un pequeño análisis de cómo puede evolucionar el proyecto en un futuro. De tener éxito y intentar aplicarlo a empresas, hay muchos aspectos del proyecto que se pueden mejorar o añadir.

5.1.1. Adaptación proyecto distintas plataformas

Una de las principales desventajas que tiene este proyecto, es que está limitado su instalación y uso al sistema operativo Debian 10 buster. Se podría ampliar a otros entornos basados en debian pero actualmente, únicamente está comprobado y probado que funciona en Debian 10 buster.

En caso de que el proyecto prospere, se debería ampliar la compatibilidad con distintas distribuciones de Linux como Ubuntu o CentOS que son de las más usadas. De esta forma se podría llegar a más público y hacer más atractivo el proyecto.

5.1.2. Creación WUI para el despliegue de playbooks

Dado el carácter técnico del proyecto, un usuario normal que utilizara este proyecto como servicio, tendría que aprender como funcionan los playbooks de Ansible en los que están basados este proyecto... Esto no es muy atractivo para posibles usuarios, ya que tendrían que tener conocimiento de cómo funcionan los playbooks, los roles... Para solventar el problema anterior, se sugiere la implementación de una interfaz de usuario, mediante la cual poder elegir que tareas ejecutar. Por ejemplo, se podrían crear formularios para insertar las variables necesarias y un selector donde elegir la tarea a desplegar. De esta forma el usuario no tendría que conocer como funcionan

los playbooks o los roles de Ansible, simplemente mediante una interfaz intuitiva sabría que hace cada uno pero no como lo hacen.

Esta parte del futuro del proyecto me parece especialmente interesante ya que una interfaz de usuario hace muy atractivo el producto. A continuación se muestra una posible interfaz

Capítulo 6

Anexo I: Manual Usuario

6.1. Despliegue automático sobre bare metal

6.1.1. Requisitos previos

Hetzner

Expliquemos que es Hetzner y mencionamos capítulos anteriores.

- IP adicional nodos maestros.
- MAC adicional para ip adicional nodos maestros.
- IP failover.
- Virtual Switch conexión nodos en cluster.
- Clave pública. Fingerprint

Dominio

Expliquemos la necesidad de tener comprado un dominio para todos los servicios. La necesidad de crear los siguientes CNAMES dentro de la zona DNS del dominio comprado:

- nodo1.dominio.com
- nodo2.dominio.com
- nodo3.dominio.com
- gitab.dominio.com

- pfsense-01.dominio.com
- pfsense-02.dominio.com
- failover.dominio.com
- ceph-admin.dominio.com
- rabbitmq-01.dominio.com

Ansible Hosts

Dependencias. Configuración host

Lista de dependencias que hay que instalar en el host donde se lanzan los playbooks. Comprobar todos los modulos de Ansible que utilizamos para ver sus dependencias y agregarlas aqui.

- Ansible 2.9.11
- Python 3.7
- Proxmoxer pip
- requests pip
- dependencias pip varias.

Clonar repositorio desde Github

6.1.2. Configuración variables Playbook

6.1.3. Ejecución Playbook

6.2. Despliegue automático y aprovisionamiento de máquinas virtuales

6.2.1. pfSense

6.2.2. GitLab

GitLab Runner

6.2.3. Webproxy

6.2.4. Docker Swarm

6.2.5. Kubernetes

6.2.6. Galera cluster

6.2.7. Ceph

6.2.8. RabbitMQ

6.2.9. Virtualmin

Bibliografía

- [1] “What is devops? — the agile admin.” <https://theagileadmin.com/what-is-devops/>. (Accessed on 07/17/2020).
- [2] “Alcanzando la agilidad con integración continua.” <https://xurxodev.com/alcanzando-la-agilidad-con-integracion-continua/>. (Accessed on 07/17/2020).
- [3] “continuous delivery deployment - agile for all.” <https://agileforall.com/frequently-asked-questions-about-agile-technical-skills/continuous-delivery-deployment/>. (Accessed on 07/17/2020).
- [4] “What is infrastructure as code? how infrastructure as code works.” <https://blog.stackpath.com/infrastructure-as-code-explainer/>. (Accessed on 07/17/2020).
- [5] “Azure devops services pricing — microsoft azure.” <https://azure.microsoft.com/en-us/pricing/details/devops/azure-devops-services/>. (Accessed on 07/18/2020).
- [6] “¿qué es devops? investigación y soluciones — google cloud.” <https://cloud.google.com/devops/?hl=es-419>. (Accessed on 07/18/2020).
- [7] “Requerimientos funcionales: Ejemplos - la oficina de proyectos de informática.” <http://www.pmoinformatica.com/2017/02/requerimientos-funcionales-ejemplos.html#:~:text=Los%20requerimientos%20funcionales%20de%20un,cuando%20se%20cumplen%20ciertas%20condiciones.> (Accessed on 07/21/2020).
- [8] “Uml: Casos de uso – ingeniería del software.” [https://ingsotoftwarekarlacevallos.wordpress.com/2015/06/04/uml-casos-de-uso/](https://ingsotftwarekarlacevallos.wordpress.com/2015/06/04/uml-casos-de-uso/). (Accessed on 07/27/2020).

- [9] “Uml: Diagrama de secuencia – ingeniería del software.” <https://ingsoftwarekarlacevallos.wordpress.com/2015/07/07/uml-diagrama-de-secuencia/>. (Accessed on 07/27/2020).
- [10] “Diagrama de gantt - wikipedia, la enciclopedia libre.” https://es.wikipedia.org/wiki/Diagrama_de_Gantt. (Accessed on 07/27/2020).
- [11] “Presupuesto técnico.” <https://fundacioncientifica.aecc.es/tramites/TR0000001012/documento/000904730125>. (Accessed on 07/27/2020).
- [12] “Dedicated root server hosting - hetzner online gmbh.” <https://www.hetzner.com/dedicated-rootserver>. (Accessed on 07/28/2020).
- [13] “Open-source virtualization management platform proxmox ve.” <https://www.proxmox.com/en/proxmox-ve>. (Accessed on 07/28/2020).
- [14] “Proxmox ve api documentation.” <https://pve.proxmox.com/pve-docs/api-viewer/>. (Accessed on 07/28/2020).
- [15] “Proxmox support forum.” <https://forum.proxmox.com/>. (Accessed on 07/28/2020).
- [16] “Mapa — mapa en tiempo real de amenazas ciberneticas kaspersky.” <https://cybermap.kaspersky.com/es>. (Accessed on 07/28/2020).
- [17] “pfSense® - world's most trusted open source firewall.” <https://www.pfsense.org/>. (Accessed on 07/28/2020).
- [18] “Empowering app development for developers — docker.” <https://www.docker.com/>. (Accessed on 07/28/2020).
- [19] “Swarm mode overview — docker documentation.” <https://docs.docker.com/engine/swarm/>. (Accessed on 07/28/2020).
- [20] “¿qué es kubernetes? — kubernetes.” <https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/>. (Accessed on 07/28/2020).
- [21] “Stateless over stateful applications — by rachna singhal — medium.” <https://medium.com/@rachna3singhal/stateless-over-stateful-applications-73cbe025f07#:~:text=A%20Stateless%20app%20is%20an,the%20client%20makes%20a%20request>. (Accessed on 07/28/2020).
- [22] “Ceph homepage - ceph.” <https://ceph.io/>. (Accessed on 07/28/2020).

- [23] “Nginx — high performance load balancer, web server, & reverse proxy.” <https://www.nginx.com/>. (Accessed on 07/28/2020).
- [24] “Usage statistics and market share of web servers, july 2020.” https://w3techs.com/technologies/overview/web_server. (Accessed on 07/28/2020).
- [25] “The first single application for the entire devops lifecycle - gitlab — gitlab.” <https://about.gitlab.com/>. (Accessed on 07/28/2020).
- [26] “¿qué es mariadb galera cluster? - mariadb knowledge base.” <https://mariadb.com/kb/es/what-is-mariadb-galera-cluster/>. (Accessed on 07/31/2020).
- [27] “Open source web hosting and cloud control panels — virtualmin.” <https://www.virtualmin.com/>. (Accessed on 07/28/2020).
- [28] “Roadway to it revolution: The history of devops.” <https://www.appknox.com/blog/history-of-devops#:~:text=The%20concept%20of%20DevOps%20emerged,it%20became%20quite%20a%20buzzword>. (Accessed on 07/17/2020).
- [29] “The 10 most in-demand tech jobs for 2020 — and how to hire for them — cio.” <https://www.cio.com/article/3235944/hiring-the-most-in-demand-tech-jobs-for-2018.html>. (Accessed on 07/17/2020).
- [30] “Aws marketplace: Devops as a service.” <https://aws.amazon.com/marketplace/pp/Kaiburr-DevOps-as-a-Service/B079Z5F722>. (Accessed on 07/18/2020).
- [31] “Google académico.” <https://scholar.google.es/>. (Accessed on 07/19/2020).
- [32] “Cómo crear un clúster de kubernetes usando kubeadm en ubuntu 16.04 — digitalocean.” <https://www.digitalocean.com/community/tutorials/how-to-create-a-kubernetes-cluster-using-kubeadm-on-ubuntu-16-04-es>. (Accessed on 07/22/2020).
- [33] “Messaging that just works — rabbitmq.” <https://www.rabbitmq.com/>. (Accessed on 07/28/2020).
- [34] “Icinga 2.” <https://icinga.com/docs/icinga2/latest/>. (Accessed on 07/28/2020).

