De casa a la nube: Cloud comunitaria y servicios distribuidos IPFS

Leopoldo Álvarez Huerta
Grado de Tecnologías de Telecomunicación
Área de Aplicaciones y Sistemas Distribuidos

Félix Freitag
Joan Manuel Marquès Puig
Enero 2019

A Aida, a mis padres y a mi hermano:
Por darme todo su apoyo.
Por soportarme, tanto en los buenos momentos como en los menos buenos.

A Juan Alonso y a Fernando:
Por prestarme su ayuda siempre que la he necesitado.
Por cederme una RPI remota.

A mis amigos y compañeros:
Por tener paciencia y aguantar mis locuras.

A Félix Freitag:
Por sus consejos, valoraciones, y haberme dado luz en momentos de oscuridad.

Porque sin todos vosotros, no hubiese sido posible.

FICHA DEL TRABAJO FINAL

Título del trabajo:	De casa a la nube: Cloud comunitaria y servicios distribuidos con IPFS
Nombre del autor:	Leopoldo Álvarez Huerta
Nombre del consultor:	Félix Freitag
Fecha de entrega (mm/aaaa):	01/2019
Área del Trabajo Final:	Aplicaciones y Sistemas Distribuidos
Titulación:	*Grado de Tecnologías de Telecomunicación*

Resumen del Trabajo (máximo 250 palabras):

En los últimos años, el número de dispositivos conectados a la red ha crecido notablemente, y ligado a ello, han evolucionado las infraestructuras y los servicios desplegados sobre las mismas. Los requisitos de accesibilidad, disponibilidad, adaptabilidad, y, en definitiva, la manera continua, y de forma remota, de explotar los servicios; desde visualización de contenidos, almacenamiento compartido, hasta soluciones distribuidas de ofimática o servicios de mensajería, han impactado directamente sobre los centros de datos.

La tendencia conocida como Edge Computing ha generado una descentralización de los *datacenter*, propiciando la creación de microcentros más cerca de los datos, los cuales utilizan servicios que recogen y procesan la información prácticamente en tiempo real y, cada vez más, apoyándose en una *cloud.* Este modelo favorece el envío de la información, a un único punto virtual, logrando que los datos estén distribuidos en varios centros, en una infraestructura distribuida y flexible, siendo objetivo de este trabajo extrapolar este modelo a entornos domésticos, de uso acotado y colaborativo.

Este trabajo está formado por una componente práctica; instalación, despliegue, y evaluación de una *microcloud* en un entorno de Smart Home, basada en productos abiertos como Cloudy, Docker, Orbit, IPFS-Cluster, Stack ELK, sobre

hardware ARM como son las Raspberry PI, y otra componente teórica, de investigación.

En el aspecto teórico, se realiza un acercamiento a IPFS y su campo de aplicación desde entornos computacionales reducidos, como el desplegado en el trabajo, hasta Internet 3.0.

Abstract (in English, 250 words or less):

Nowadays, the number of devices connected to the network has grown significantly, and linked to this, the infrastructures and services which have deployed on them have evolved. The requirements of accessibility, availability, adaptability, and the continuous and remotely way of exploiting the services; from content visualization, shared storage, to distributed office automation solutions or messaging services, have directly impacted on the datacenters.

The trend known as Edge computing has generated a decentralization of the datacenter, favoring the creation of microcenters closer to the data, which use services that collect and process information practically in real time and, increasingly, relying on a cloud. This model favors the sending of information to a single virtual point, achieving that the data is distributed in several centers, in a distributed and flexible infrastructure, being the objective of this work to extrapolate this model to domestic environments, of limited and collaborative use.

This work is formed by a practical component; installation, deployment, and evaluation of a microcloud in a Smart Home environment, based on open products such as Cloudy, Docker, Orbit, IPFS-Cluster, Stack ELK, on ARM hardware such as Raspberry PI, and another theoretical, research component.

The theoretical aspect is an approach to IPFS, and its field of application, from small computational environments, such as the one that has deployed at work, to Internet 3.0.

Palabras clave (entre 4 y 8):

Cloudy, microCloud, Docker, Raspberry, IPFS, Orbit, Edge Computing

Índice

Lista de figuras

1. Introducción

Este trabajo, denominado “Cloud comunitaria: De casa a la nube y servicios IPFS”, pretende realizar un recorrido por el camino de las nuevas tecnologías y protocolos emergentes, centrándose en el uso de componentes abiertos, y con la premisa de colaboración y mejora dentro de diferentes comunidades.

Por otro lado, se realizará una exposición de la situación actual en cuanto a *clouds* comunitarias y aplicaciones distribuidas, desplegando una microCloud, primero local, y posteriormente escalando a una microCloud regional, basada en hardware de uso personal y cotidiano, como puede ser un PC y las Raspberry PI.

Todos los componentes utilizados, como se indica en párrafos posteriores, serán abiertos o cumplirán el estándar para uso con tecnologías abiertas.

1.1 Contexto y justificación del Trabajo

Con el crecimiento, exponencial en los últimos años, del uso de nuevas tecnologías y el empleo de *smartphones* u otros *gadgets* nos hemos encontrado, casi sin darnos cuenta, conectados unos con otros.

Este conjunto de dispositivos, sumado a los conocidos como tradicionales; ordenadores de sobremesa, portátiles, etc., generan y consumen gran cantidad de datos, impactando directamente en los sistemas de información y en los servicios que se desplegaban en un pasado muy cercano.

En estos días en los que el IoT aún está despegando, es una realidad que el mundo está cada vez más conectado, y los seres humanos estamos inmersos dentro de esa realidad. Tanto a nivel profesional como personal, prácticamente no hay justificación para no estar disponible en todo momento, y tener a mano todas las posibilidades que ofrece Internet.

Se dispone de una serie de servicios, que, sin entrar en juicio de valor, pueden brindar un mayor confort en el día a día; como el acceso continuo a la información o a diversas opciones de entretenimiento. No obstante, para obtener ese tipo de acceso descentralizado, desde prácticamente cualquier lugar, y desde cualquier tipo de dispositivo; *tablet*, *smartphone*, portátil, PC, etc. se deben, entre otros aspectos, hacer uso de las aplicaciones y sistemas distribuidos.

Uno de los problemas comunes en los sistemas de información o en la ingeniería de sistemas es que el software es diseñado para que funcione con unas determinadas características o unas especificaciones que muchas veces están limitadas por los fabricantes o desarrolladores. Esto sucede, generalmente, con aplicaciones o sistemas operativos cerrados como es el caso de Windows en Microsoft o MacOS en Apple Inc, de ahí que, con el crecimiento del catálogo de servicios y las diversas aplicaciones, se requiera, aún más, el uso del Software libre como solución.

El usuario de hoy en día que necesita acceso a los servicios distribuidos, en cualquier momento, desde cualquier dispositivo y lugar, requiere también poder acceder de manera fiable y segura, además de disponer de cierta flexibilidad, no proporcionada a veces por las nubes y software propietario. Las características que proporcionan los servicios distribuidos, como concurrencia y tolerancia a fallos, unidas al uso de *OpenSource;* sistemas operativos, hipervisores, *containers*, etc. y al trabajo colaborativo, son claves para los objetivos de este proyecto.

Por lo tanto, el objetivo principal de este trabajo es el de implementar un sistema distribuido sobre diferentes componentes hardware repartidos geográficamente, de una forma accesible, práctica, y económica, haciendo uso de tecnologías *Open Source,* y de infraestructura ARM y X86, que hoy en día, se adapta en cualquiera de los considerados hogares inteligentes o *Smart Home*.

La primera aproximación del trabajo consiste en desplegar una microCloud local, es decir, un sistema distribuido basado en el siguiente hardware:

- RaspBerry PI 3 B
- RaspBerry PI 3 B+
- PC AMD Athlon XP 2400+

Acto seguido, se deberá demostrar la escalabilidad de esta microCloud extendiendo dichos servicios distribuidos fuera del “hogar”. De ahí el título del proyecto: “De casa a la nube”.

Por un lado, se tendrá un servicio descentralizado y distribuido geográficamente, además de escalable, cumpliendo una serie de requisitos que quedaran definidos a lo largo del trabajo.

Para realizar la comunicación entre el hardware se utilizará la red IP, no obstante, se empleará una VPN soportada por Cloudy para conectar los nodos remotos.

Por último, se realiza un acercamiento a IPFS (InterPlanetary File System), conocido comúnmente como la “Web distribuida”, detallando el protocolo, desplegando servicios basados en él, y estudiando la integración y las aportaciones dentro de un entorno de nube comunitaria.

1.2 Objetivos del Trabajo

Los objetivos de este trabajo se pueden dividir en dos líneas diferenciadas.

Por un lado, unos objetivos generales ligados a una línea teórica y de investigación:

- Investigar sobre los componentes, tanto *software* como *hardware*, además de los protocolos necesarios para el diseño de una microcloud local y regional.
- Realizar un acercamiento a la implementación de servicios distribuidos de alta disponibilidad en entornos no corporativos.
- Investigar el servicio Orbit y OrbitDB.
- Estudiar el impacto, integración y potencial de IPFS.

Por otro lado, unos objetivos específicos acordes con la línea práctica de desarrollo y evaluación:

- Establecer plazos de ejecución y despliegue; escalabilidad de la microcloud.
- Elaborar un presupuesto que se adapte a los recursos.
- Realizar la “cloudynitzación” en varios dispositivos hardware, entre ellos, Raspberry Pi 3.
- Realizar la interconexión entre todos los elementos del sistema formando una microcloud local en una fase inicial, y acto seguido, una cloud regional
- Ofrecer una serie de servicios distribuidos dentro de la microcloud.
- Implementar servicio bajo IPFS

1.3 Enfoque y método seguido

Debido a que es un proyecto con partes diferenciadas de aprendizaje e investigación, y otra, destinada a poner en práctica productos ya existentes, complementando y evolucionado los mismos dentro de una plataforma escalable, tanto a nivel de *software* como de *hardware*, se decide, desarrollar un primer despliegue de un producto de computación en una micro nube, para posteriormente añadir diversas funcionalidades a través de las características que nos ofrece un producto como Docker.

Por otro lado, se evaluará como complementar nuevos servicios a nuestra *cloud*, además de investigar sobre IPFS.

Por lo tanto, existe una primera fase de despliegue del entorno, y una segunda fase o componente de evaluación e investigación del potencial de este tipo de entornos y las nuevas tendencias tecnológicas.

1.4 Planificación del Trabajo

Para elaborar la planificación se diferencian varias fases que incluyen:

- Estudio e investigación
    - Estudio de las compatibilidades software y hardware.
    - Definición de arquitectura de red y diseño a implantar.
    - Estudio y PoC servicios distribuidos; Docker, Orbit, OrbitDB, ELK.
    - Foco en el producto Orbit y protocolo IPFS
    - Estudio tit-for-tat. Posibilidad de control de reputación e integración BlockChain
- Implantación y configuración
    - Instalación, configuración y despliegue de un nodo.
    - Replicación al resto de nodos.
    - Envío nodo remoto.
    - Instalación, configuración y despliegue nodo PC (AMD Athlon)
    - Integración microcloud local
    - Integración microcloud regional
    - Despliegue SSDD sobre microCloud
- Presupuesto
- Elaboración de documentación

1.4.1 Planificación del Trabajo. EDT

La distribución de las tareas globales del proyecto consistirá en los siguientes apartados cuyo desglose en la estructura de descomposición del trabajo es de 300 horas:

Nombre de tarea	Trabajo	Duración	Comienzo	Fin
TFG - Cloud Comunitaria	**300,22 horas**	**79 días**	**nié 19/09/18**	**om 06/01/19**
Planificación de proyecto	**38 horas**	**11 días**	**nié 19/09/18**	**nié 03/10/18**
Propuesta	6 horas	0,75 días	nié 19/09/18	nié 19/09/18
Descripción	8 horas	1 día	jue 20/09/18	sáb 22/09/18
Objetivos	12 horas	1,5 días	om 23/09/18	vie 28/09/18
Planificación de proyecto	8 horas	1 día	sáb 29/09/18	om 30/09/18
Entregable: PEC1	4 horas	0,5 días	om 30/09/18	nié 03/10/18
Investigación	**24,02 horas**	**7 días**	**lun 01/10/18**	**nié 10/10/18**
Estado actual	8 horas	1 día	lun 01/10/18	vie 05/10/18
Estado futuro y posibilidades	8,01 horas	1 día	vie 05/10/18	om 07/10/18
Entregable: PEC2	8,01 horas	1 día	om 07/10/18	nié 10/10/18
Diseño	**16,02 horas**	**2 días**	**jue 11/10/18**	**lun 15/10/18**
Diseño físico	8,01 horas	1 día	jue 11/10/18	sáb 13/10/18
Diseño lógico	8,01 horas	1 día	sáb 13/10/18	lun 15/10/18
Implantación y despliegue	**172,15 horas**	**57 días**	**lun 01/10/18**	**nar 18/12/18**
Instalación nodos principales	**124,1 horas**	**33 días**	**nar 16/10/18**	**sáb 01/12/18**
Instalación SW RPI (3nodos)	**60,04 horas**	**16 días**	**nar 16/10/18**	**nié 07/11/18**
Configuración	**64,05 horas**	**16 días**	**jue 08/11/18**	**sáb 01/12/18**
Despliegue cluster	**20,02 horas**	**50 días**	**lun 01/10/18**	**sáb 08/12/18**
Escalabilidad	**28,03 horas**	**57 días**	**lun 01/10/18**	**nar 18/12/18**
Servicios distribuidos (IPFS)	**10 horas**	**2 días**	**nié 19/12/18**	**sáb 22/12/18**
Orbit	10 horas	1,25 días	nié 19/12/18	sáb 22/12/18
Soporte digital	**32,04 horas**	**9,88 días**	**sáb 22/12/18**	**vie 04/01/19**
Contenido multimedia	8,01 horas	1 día	sáb 22/12/18	lun 24/12/18
Memoria	24,03 horas	3 días	nar 25/12/18	vie 04/01/19
Presupuesto	8 horas	1 día	vie 04/01/19	om 06/01/19
Entrega TFG	0 horas	0 días	om 06/01/19	om 06/01/19

Figura 1: Estructura de desglose de tareas

Cabe destacar que, de estas 300 horas, más de 24 son específicas de investigación y análisis, además de las empleadas indirectamente dentro de otros capítulos como el de Servicios distribuidos. La implantación de despliegue de los entornos supera las 170 horas, y la planificación, seguimiento y actualización del proyecto consume 38 horas.

1.4.2 Planificación del Trabajo. Gantt

Para el cronograma del proyecto se utiliza Microsoft Project Professional 2016. Se tiene en cuenta que para realizar una planificación lo más real posible, con un único recurso, las tareas se planifican de manera secuencial, ejecutando algunas en paralelo como puede ser la documentación, entrega de PECs, etc.

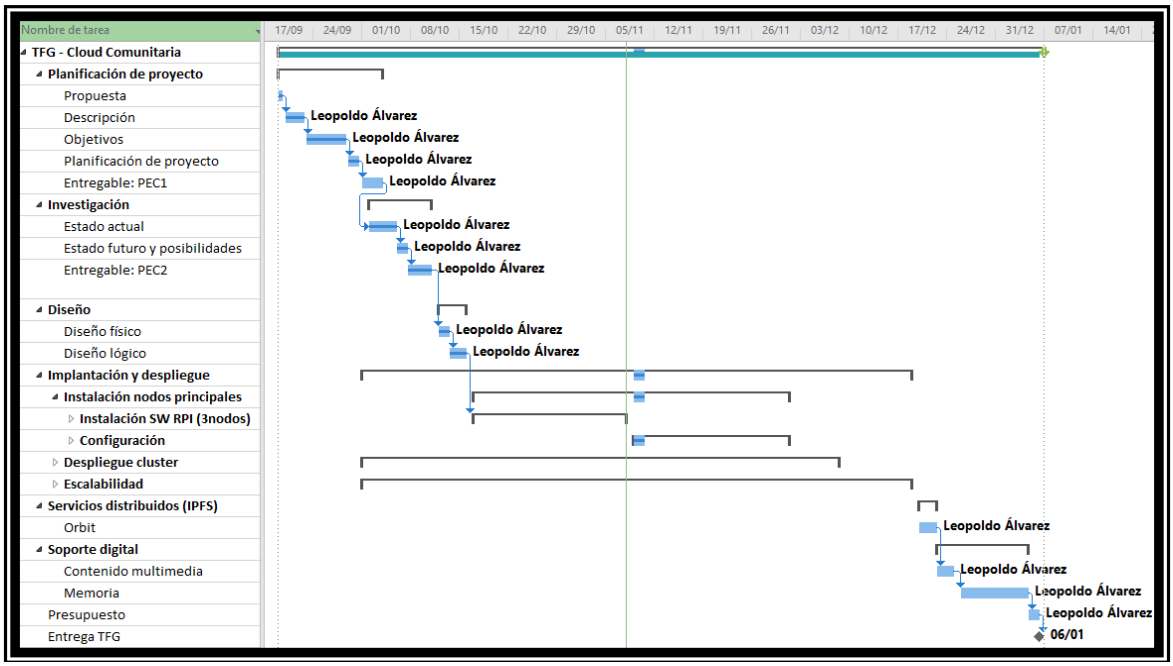


Figura 2: Diagrama de Gantt

Se adjunta en el apartado de anexos, los ficheros relacionados con la planificación.

1.5 Breve sumario de productos obtenidos

Los productos obtenidos, que se detallaran en apartados posteriores, son:

- MicroCloud local dentro del Smart Home
- MicroCloud regional o comunitaria
- MicroCloud valor añadido. Backup, escalabilidad, disponibilidad, seguridad, etc.
- Acercamiento a IPFS: presente y futuro.
- MicroCloud + IPFS. Servicios de almacenamiento distribuido, swarm, cluster privado, etc.

1.6 Breve descripción de los otros capítulos de la memoria

En los siguientes capítulos se desarrollan las diferentes fases por las que transcurre el trabajo, que finalizará, con las conclusiones obtenidas durante la ejecución de este.

- Capítulo 2. Estado actual. Donde se detalla la evolución de los sistemas distribuidos hasta la foto actual del modelo de servicios basados en la nube o en Edge Computing.
- Capítulo 3. Diseño del sistema. Se detalla el diseño del sistema, tanto a nivel de arquitectura física, como lógica.
- Capítulo 4. Servicios distribuidos e IPFS. Se realiza un acercamiento a los servicios distribuidos basados en IPFS.
- Capítulo 5. Implantación: Instalación y despliegue. En este capítulo se detalla la instalación de la microCloud y el despliegue de los nodos, así como la instalación de *software* adicional y útil en el desarrollo del proyecto.
- Capítulo 6. Evaluación. Se realiza una evaluación desde diferentes características de un sistema distribuido, como son la escalabilidad, disponibilidad, adaptabilidad o tolerancia a fallos, y seguridad.
- Capítulo 7. Valoración de la solución obtenida. Se aportan una serie de valoraciones en base a los capítulos anteriores y se indica el coste del despliegue del sistema empleado.
- Capítulo 8. Conclusiones. Se exponen las conclusiones obtenidas durante la ejecución del trabajo, los hitos conseguidos, y un posible enfoque sobre las opciones de siguientes trabajos.

2. Estado actual.

Tal y como se indica en el capítulo previo; “Contexto y justificación del trabajo”, en los últimos años se ha detectado un crecimiento exponencial del uso de dispositivos conectados a Internet. Dicho crecimiento está ligado al aumento de *Smartphones* y otros *gadgets,* como pueden ser *wearables* o cualquier otro dispositivo IoT (*Internet of Things*). Toda esta agrupación de dispositivos se conecta, mediante diversas aplicaciones, a los sistemas de información. Estos sistemas, son el resultado de aplicar diversas ciencias como la física y las matemáticas, en conjunto con la informática y la electrónica, con el fin de desarrollar, entre otras, tecnologías que aporten beneficio a la humanidad, y mejoras en la eficiencia del esfuerzo científico.

Los sistemas de información, y los servicios que ofrecen, están vinculados directamente a la empresa y a sus oficinas. Se encuentran acotados a la localización de las infraestructuras físicas de la empresa, limitados, como si de una frontera se tratase, no obstante, este escenario ha ido evolucionando con la llegada de nuevas tecnologías de red, protocolos de comunicación, etc. así como ha propiciado el desarrollo de nuevos servicios. Y es en la era de la información y la movilidad, cuando este escenario evoluciona aún más hacia una descentralización total de los sistemas, consolidándose el *Cloud,* y asistiendo al nacimiento del *Edge* y *Fog Computing.*

El *Cloud Computing*, definida por Cisco Systems Inc. como los “Recursos de TI que se abstraen de la infraestructura subyacente y se brindan bajo demanda, y a escala, en un entorno multiusuario.” [1], es un conjunto de tecnologías que se vale de Internet para ofrecer sus recursos, tanto *software* como *hardware*, a unos clientes que pueden ser desde todo tipo de empresas a organizaciones gubernamentales o usuarios finales.

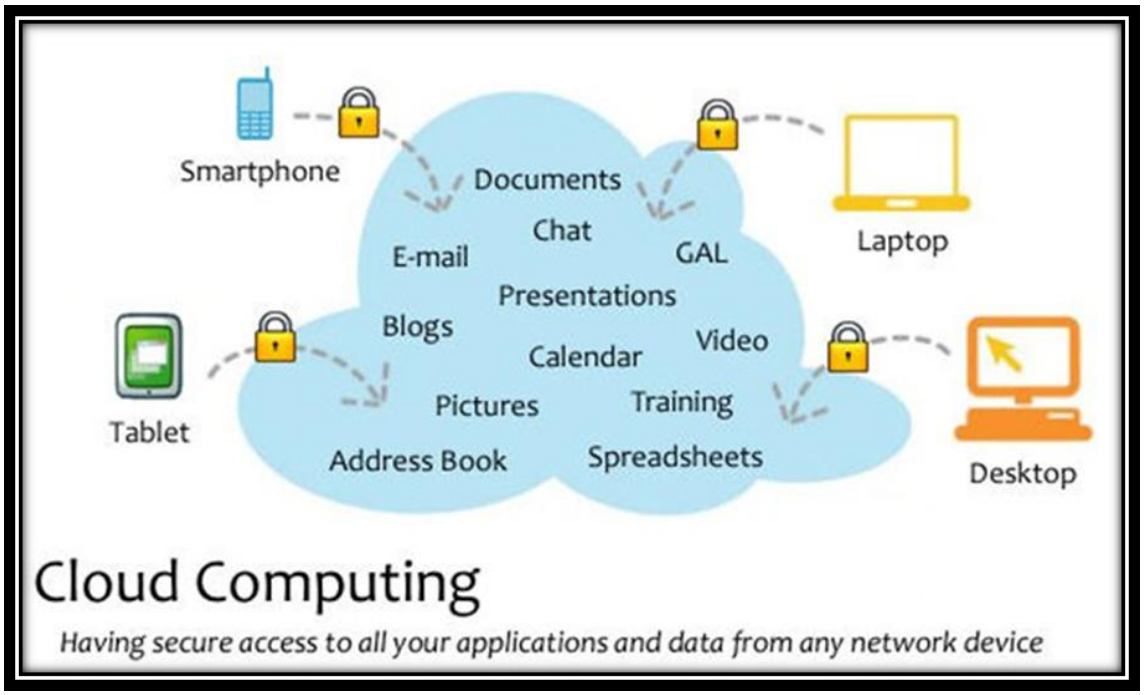


Figura 3: Características de acceso del Cloud Computing [2]

En este sentido, se desarrolla un nuevo modelo de gestión tecnológico que modifica el cómo, el cuándo, y el dónde, es decir, se modifica la forma contemporánea de provisión, despliegue y utilización de los recursos hacia un nuevo paradigma de servicios; los servicios *cloud*.

Podemos diferenciar este modelo en tres capas fundamentales: Software como Servicio (SaaS), Plataforma como servicio (PaaS) e Infraestructura como Servicio (IaaS), aunque este modelo es dinámico, y se tiende hacia un “Todo como Servicio” (Anything as a Service - XaaS).

Figura 4: Modelo de servicios Cloud [3]

Pese a la tendencia indicada anteriormente, el uso de este tipo de servicios no es obligatorio, no obstante, se cita alguno considerado de uso cotidiano o general, con el fin de comprobar su alta penetrabilidad en la sociedad hoy en día, por ejemplo:

- Google Docs: Herramienta de creación de documentos.
- Dropbox: Herramienta de almacenamiento compartido, utilizada para compartir archivos. Servicio distribuido de discos duros virtual.
- Slideshare: Herramienta destinada a compartir presentaciones en diferentes formatos.
- Wordpress: Herramienta para creación de páginas web.
- WikiSpaces: Herramienta de trabajo en red.

Estos servicios distribuidos basados en la nube tienen un propósito comercial, y requieren una serie de vinculaciones o aceptaciones para su uso. Vinculaciones que en el ámbito del uso colectivo o corporativo pueden no ser atractivas, de ahí que existan otras alternativas que se tratan en el marco de este trabajo.

Volviendo al concepto de sistemas de información, como cualquier otra tecnología, se debe conocer el funcionamiento, alcance, y evolución de esta, con el fin de poder maximizar su aplicación y ser más eficaces o eficientes en esta era de la información. Uno de los problemas comunes en los sistemas de información, sin dejar de lado los servicios basados en C*loud*, es que el software se diseña para que funcione con unas determinadas características, para cumplir unos estándares o unas especificaciones, que muchas veces están limitadas por los fabricantes o desarrolladores. Esto sucede, generalmente, con aplicaciones o sistemas operativos cerrados como es el caso de Windows y Azure en Microsoft, iCloud y MacOS en Apple Inc, de ahí que, con el crecimiento del catálogo de servicios y las diversas aplicaciones, se requiera, aún más, el uso del Software Libre como solución.

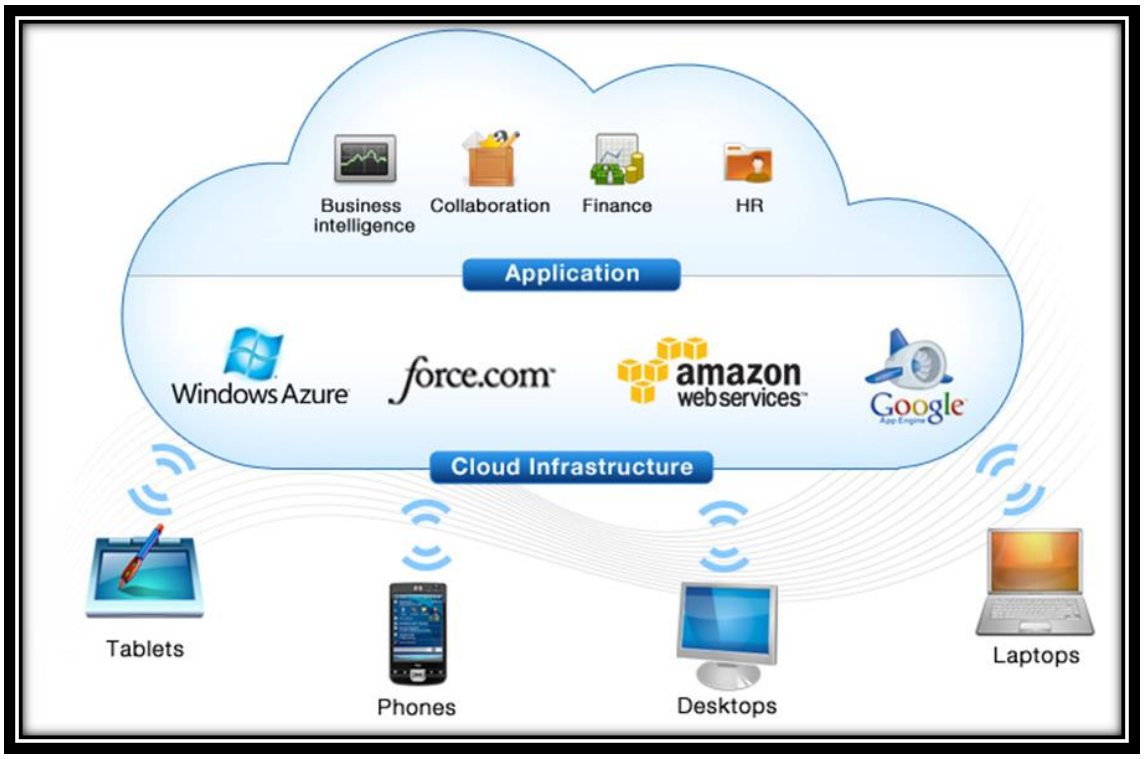


Figura 5: Proveedores de Cloud Computing [4]

Por otro lado, desde el punto de vista de los servicios distribuidos sobre C*loud*, es necesario definir, profundizar y utilizar unos estándares, debido a que, por normal general, cada fabricante de *software* está desarrollando sus propias nubes, y utilizando metodologías propietarias e incluso protocolos que puede inducir a problemas con integración de otras aplicaciones e interconexión con otras nubes.

Estos estándares se pueden dividir en dos clases; estándares prescriptivos y estándares evaluativos [5]. Los estándares prescriptivos se refieren a los estándares de comunicaciones como pueden ser los protocolos TCP/IP, mientras que los estándares evaluativos se refieren a estándares de calidad de estos servicios de la nube, responsables de definir, detallar y evaluar los procedimientos seguidos en los procesos, como es el caso de la familia ISO 9000 o específicos de seguridad de la información como la ISO 27000. Algunos de los principales avances de estandarización de Cloud Computing [6], son los realizados por la ITU-T, mediante un FG (Focus Group) específico para Cloud que decidió que el grupo de trabajo SG13 (Study Group 13) dirigiría la actividad de normalización de Cloud Computing, y el grupo SG17 cubriría la seguridad en la nube, y, por otro lado, el trabajo realizado por DMTF (Distributed Management Task) y el OOC (Open Cloud Consortium).

No obstante, en nubes propietarias se pierde la visión dentro de cada servicio del uso de determinados estándares o de protocolos, más allá de la mención en la documentación oficial del fabricante. Algunos aspectos para la auditoria de calidad de los proveedores de servicios de Cloud Computing incluyen KPIs como: tiempo de actividad o utilización, rendimiento, disponibilidad, seguridad, privacidad, servicio al cliente, etc. En este punto de cumplimiento de estándares y especificaciones, si el cliente demanda, por ejemplo; un nuevo KPI, una aplicación no presente en el catálogo o una modificación de un servicio existente, se puede encontrar con uno de los problemas indicados anteriormente sobre el software propietario.

Por otro lado, no solo se debe pensar en las empresas como clientes, con contratos específicos, también se debe tener en cuenta a otro tipo de usuarios finales como pueden ser universidades, asociaciones sin ánimo de lucro, comunidades e individuos particulares, en resumen, este tipo de tecnología debe ponerse en conocimiento general, y su espectro ser lo más amplio posible.

Por lo tanto, el usuario de hoy en día que necesita acceso a los servicios distribuidos, en cualquier momento, desde cualquier dispositivo y lugar, requiere también poder acceder de manera fiable y segura, además de disponer de cierta flexibilidad, no proporcionada a veces por las nubes y software propietario. El análisis de la viabilidad de cumplir estos requisitos se ve reflejada en este trabajo mediante la combinación del *OpenSource,* y el *Cloud Computing;* sistemas operativos, hipervisores, contenedores virtuales, servicios distribuidos, comunicaciones estándar y de nueva generación, etc. Recogidos en el marco de proyectos de gran envergadura, como pueden ser Raspbian [7], Openstack [8], Docker [9], OpenDaylight [10], y algunos otros, que se han nombrado en capítulos previos y se detallan en capítulos posteriores; IPFS [11], Serf [12].

Por último, sin dejar de lado el párrafo anterior, pero volviendo al punto inicial de este estado del arte, se debe tener en cuenta que la palabra “usuario” está siendo sustituida por “dispositivo”. Cualquier dispositivo existente en el entorno puede conectarse de forma segura a servicios distribuidos descentralizados, interactuando entre ellos de una manera escalable y eficiente, operando con información procesada de servidores diferentes. En este modelo, el procesamiento de datos y las aplicaciones requieren concentrarse en los dispositivos al borde de la red; *Edge Computing*, en lugar de completamente en

la nube. De esta manera el acceso a los datos es más ágil y pueden ser procesados localmente en lugar de ser enviados a la nube. Este modelo persigue especialmente una arquitectura IoT real, donde todos los dispositivos conectados que recogen datos interactúen entre sí, procesando información y generando otro tipo de datos o acciones. La tendencia a extender la nube hacia el borde de la red, más cerca del dispositivo o del usuario, es el paradigma conocido como *Fog Computing*.

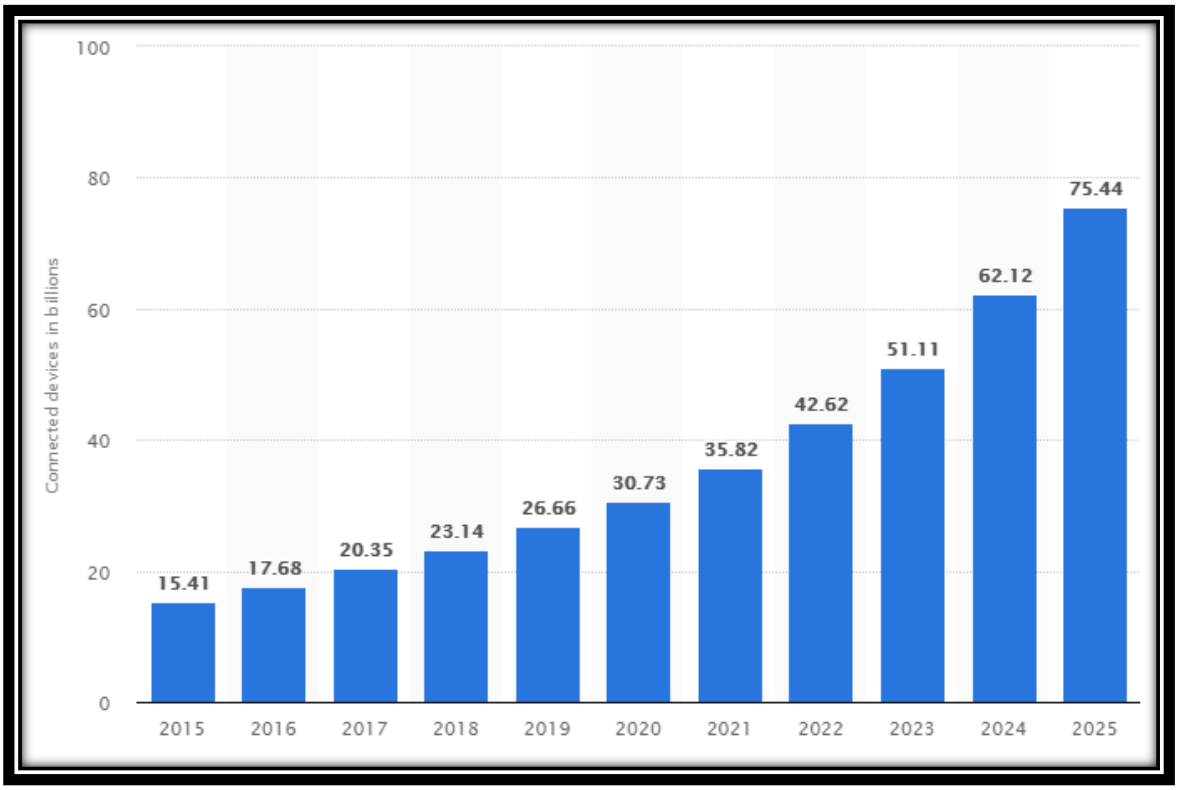


Figura 6: Número de dispositivos conectados (billones) [13]

Las características distintivas de *Fog* respecto al *Cloud Computing* son su proximidad a los usuarios (dispositivos) finales, su densa distribución geográfica y su soporte para la movilidad. Los servicios se alojan en el borde de la red o en los propios dispositivos finales, como decodificadores, sensores o puntos de acceso. Así se reduce la latencia del servicio y mejora la QoS, estimando un resultado de una QoE superior. *Fog Computing* es compatible con las aplicaciones emergentes de Internet de todo (IoE), que requieren comunicación en tiempo real con una latencia inexistente o predecible. Por otro lado, debido a la distribución geográfica, el paradigma de *Fog* está bien posicionado para *Big Data* en tiempo real, soportando una alta densidad de puntos de recolección de

datos distribuidos, por lo tanto, agrega un cuarto eje a las dimensiones de *Big Data*; volumen, variedad y velocidad, mencionadas a menudo.

Dentro del marco de este estado del arte, y en el ámbito del *Cloud, Edge* y *Fog Computing,* se estudian varios proyectos como Cloudy [14], EdgeX Foundry[15], FogFlow [15].

EdgeX Foundry es un *framework* de microservicios de código abierto que proporcionan una plataforma con una capacidad mínima de *Edge Computing.* El código fuente del microservicio de EdgeX Foundry se puede descargar y compilar en los dispositivos de implementación, no obstante, si no se tiene una necesidad específica de ejecutar EdgeX Foundry de forma nativa, existe una opción de ejecutarlo mediante contenedores virtuales y Docker. Los microservicios de EdgeX Foundry se crean y almacenan automáticamente a medida que el nuevo código se registra en el repositorio de origen. Por lo tanto, EdgeX Foundry "Dockerized" es más fácil de obtener e implementar.

FogFlow, por el contrario, es un entorno de trabajo orientado a gestionar de manera dinámica el flujo de procesamiento de datos en la nube y los bordes, permitiendo procesar la información de contexto bajo demanda. Por otro lado, nace con el objetivo de cumplir las siguientes inquietudes reflejadas en estos últimos párrafos:

- Coste de una solución dedicada en la nube es elevado para ejecutar un sistema IoT a gran escala con más de mil dispositivos distribuidos geográficamente
- Muchos servicios de IoT requieren un tiempo de respuesta rápido, inferior a diez milisegundos de latencia de extremo a extremo
- Complejidad y costes elevados para diseño e implementación ágil de servicios de IoT en un mercado de continuos cambios en las demandas.
- Falta de interoperabilidad y estandarización para compartir y reutilizar los datos.

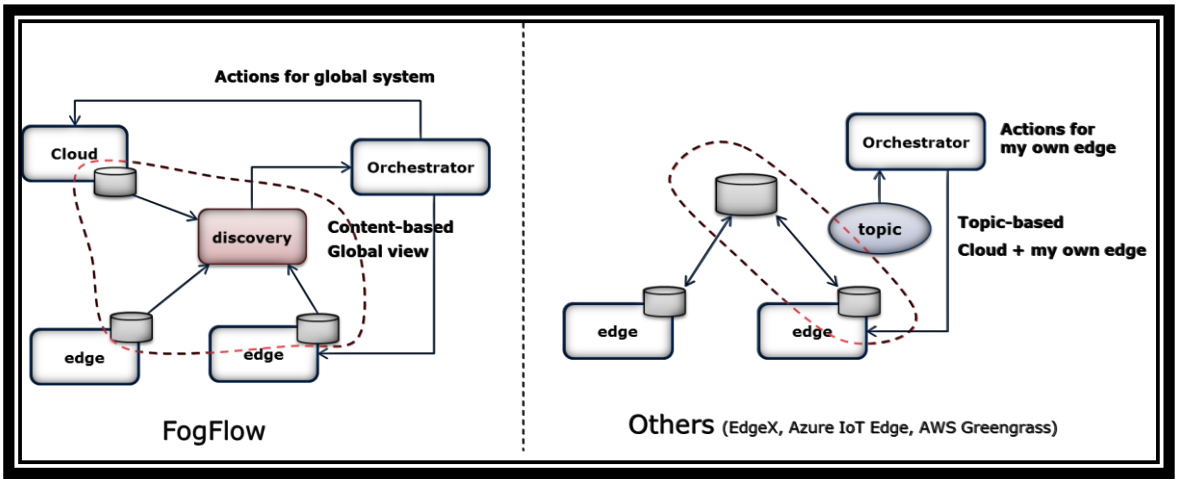


Figura 7: FogFlow vs Others [16]

Finalmente, y debido al alcance de este proyecto, se decide por la solución basada en Cloudy, descrita en el capítulo siguiente.

3. Diseño del sistema

El diseño del sistema se basa en una microCloud abierta, desvinculada de *software* propietario y *hardware* específico. Dicha C*loud* está compuesta por varios nodos bajo una configuración que dotará al sistema de escalabilidad y versatilidad, permitiendo desarrollar diferentes tipos de funciones o servicios distribuidos, como se puede ver a lo largo del desarrollo del trabajo.

Este capítulo se divide en arquitectura; diseño físico, elementos, y diseño lógico.

3.1 Diseño físico

El diseño o la arquitectura física del sistema está basada en la interconexión de los siguientes componentes hardware que detallaremos a continuación:

- Raspberry PI 3B (microcloud local – red LAN)
- Raspberry PI 3B+ (microcloud local – red LAN)
- AMD Athlon XP 2400 (microcloud local – red LAN)
- Raspberry PI 3B (recurso remoto microcloud regional – VPN)
- Máquinas virtuales x86/amd64 (recurso remoto microcloud regional – VPN)
- CableModem Router Cisco EPC 3925 (Router - NAT)
- Clientes; PC, Portatil, máquina virtual, smartphone, etc.

Los nodos sobre los que se encuentra soportada esta microcloud son de 3 tipos; Raspberry PI, PC doméstico y máquinas virtuales sobre servidor doméstico.

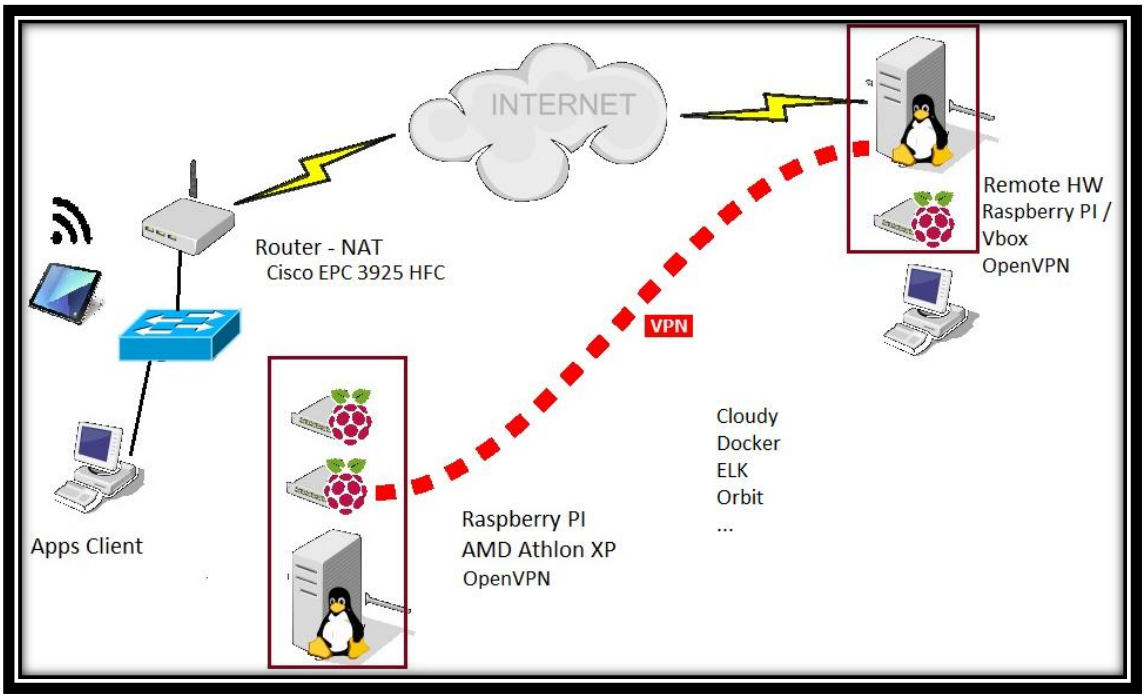


Figura 8: Esquema de diseño físico

3.1.1 Rasperry PI 3B

La Raspberry PI es un dispositivo denominado SBC (Single Board Computer), con arquitectura ARM (Advanced RISC Machine) de 64 bits, y en el caso de los modelos utilizados, las siguientes características:

- Procesador: Chip Broadcom BCM2387 ARMv8
    - RPI3B: 1,2 GHz 4 cores
- GPU: Dual Core VideoCore IV
- RAM: 1GB LPDDR2
- Ethernet:
    - RPI3B: Ethernet 10/100 Base T. Wi-fi 802.11 b / g / n
- USB 2.0 hub (4 usable)

Figura 9: Raspberry PI 3B

Los procesadores ARM, al disponer de un diseño basado en RISC, requieren una cantidad menor de transistores que los procesadores x86 CISC, utilizados en la mayoría de los ordenadores personales. Este diseño, por tanto, se traduce en una reducción de los costes, calor, y en definitiva, energía. Este aspecto ha sido clave para identificar estos SBC como el hardware idóneo para este proyecto.

Por otro lado, aunque no es el objetivo de este trabajo, existen diferentes soluciones para las RPI tipo carcasa apilable donde se puede realizar un *stack* de las mismas escalando hasta 3, 5 o 10 RPI según modelo.

3.1.2 Rasperry PI 3B+

La Raspberry PI 3B+ únicamente difiere de la anterior en algunas características hardware.

- Procesador: Chip Broadcom BCM2387 ARMv8
    - RPI3B+: 1,4 GHz 4 cores
- GPU: Dual Core VideoCore IV
- RAM: 1GB LPDDR2
- Ethernet:
    - RPI3B+: Ethernet 10/100/1000 Base T. Wi-fi 802.11 b / g / n / ac & 802.3az Energy Efficient Ethernet

- USB 2.0 hub (4 usable)

Nueva Raspberry Pi 3 Model B+

ESPECIFICACIONES	RASPBERRY PI 3 MODEL B+ (2018)	RASPBERRY PI 3 MODEL B (2016)
PROCESADOR	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz	Broadcom BCM2837, Cortex-A53 (ARMv8) 64-bit SoC @ 1.2GHz
RAM	1GB RAM	1GB RAM
CONECTIVIDAD	WiFi 802.11.b/g/n/ac de doble banda 2.4GHz y 5GHz Bluetooth 4.2 Puerto Ethernet de hasta 300Mbps	WiFi 802.11 b/g/n (2.4GHz) Bluetooth 4.1 Puerto Ethernet de hasta 100Mbps
PUERTOS	HDMI completo, 4 USB 2.0, Micro SD, CSI camera, DSI display	HDMI completo, 4 USB 2.0, Micro SD, CSI camera, DSI display

Figura 10: Raspberry PIB vs RPI 3B+ [17]

3.1.2 AMD Athlon 2400 XP

Este componente de la microcloud local es un ordenador personal de uso domestico. Se utiliza, únicamente, para demostrar que el proyecto se puede desarrollar sobre un escenario de hardware heterogeneo, que puede darse en cualquier *SmartHome* actual.

Las características hardware del PC utilizado son:

- Procesador: Arquitectura x86 AMD Athlon XP 2,7GHz
- RAM: 4GB RAM DDR3
- Ethernet: 10/100 Base T
- GPU: Nvidia Geforce TI 4200 512 MB DDR
- USB 1.0 (6 usable)
- Disk: 2 TB

3.1.3 Cisco EPC 3925 EuroDocsis

Se describe este dispositivo de electrónica de red debido a que es una parte fundamental en la microcloud desplegada en el proyecto. El Cisco EPC3925 es un dispositivo denominado CMR (Cable-Modem-Router) compatible con DOCSIS 3.0 cuyas características principales son las siguientes:

- Ethernet: 4 puertos RJ45 10/100/1000 y Wi-fi 802.11 b / g / n

- 8 canales de bajada hasta 440 Mbps
- 4 canales de subida hasta 120Mbps

Figura 11: Cisco EPC 3925 Docsis 3.0

3.2 Diseño lógico

El diseño lógico del sistema está formado por varias capas. Tal y como se ha visto en el diseño físico, se dispone una capa correspondiente al *hardware* donde se encuentra la infraestructura, tanto de red como de servidores, mientras que las capas superiores serían la de sistema operativo, contenedores virtuales, servicios y otras aplicaciones o *Front End*.

Sobre la capa de física se despliegan sistemas operativos basados en Debian como son Raspbian 9 o Ubuntu 16.04 LTS, formando la capa de sistema operativo, posteriormente se implementa Cloudy y Docker; proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software. La utilización de Docker es una de las claves para desplegar una solución basada en contenedores virtuales en GNU/Linux, dejando al margen el despliegue de un hipervisor. Este diseño con contenedores proporciona una primera capa de virtualización y segmentación, directamente sobre la capa del sistema operativo. Esta capa se encarga de la abstracción y automatización de

aplicaciones en diferentes sistemas. Por último, sobre el motor del gestor de contenedores virtuales se encuentra la capa de aplicación, desde donde se accede a los diferentes servicios distribuidos vía API, CLI o WebGUI. En esta capa se puede encontrar tanto la WebGui de Cloudy como la de otros servicios ofrecidos en la plataforma, además de la interactuación vía cliente o línea de comandos.

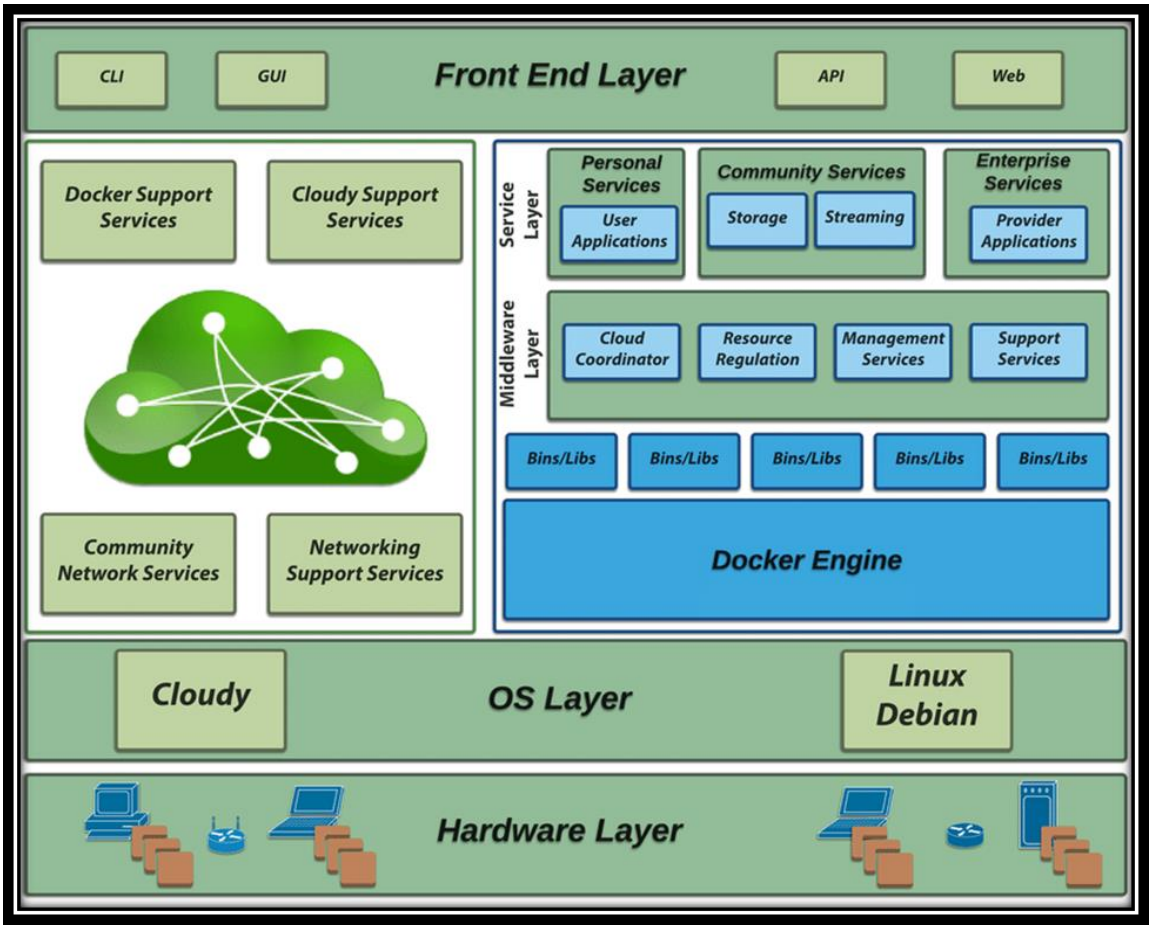


Figura 12: Diseño lógico. Arquitectura por capas: Cloudy y Docker [18]

3.2.1 Cloudy

Cloudy, tal y como se ha visto en el capítulo previo, se encuentra en la capa de sistema operativo y es la plataforma elegida para el desarrollo del proyecto. En este trabajo se ha optado por convertir varios sistemas operativos; Debian 9, Raspbian y Ubuntu 16.04 LTS en Cloudy (cloudynitzar), no obstante, se puede desplegar sobre un hipervisor (ESXi, Virtual Box, Proxmox) o instalar directamente sobre prácticamente cualquier *hardware*.

Cloudy nace como una solución bajo el proyecto CLOMMUNITY [19], con el objetivo de reducir la barrera tecnológica animando a los usuarios a compartir

recursos y servicios con la comunidad. Esta distribución, basada en GNU/Linux, elimina la necesidad del usuario de requerir conocimiento técnico para el despliegue y mantenimiento de un servidor con sus servicios asociados. Por lo tanto, este aporte hace que la adopción y transición a un entorno Cloud en redes comunitarias sea más accesible, además, posibilita trasladar la localización de los servicios más cerca del usuario; *Edge Computing*.

3.2.1.1 Motivaciones de la distribución Cloudy

Cloudy reúne una serie componentes, los cuales se pueden revisar en detalle en la siguiente referencia [20], destinados a cumplir las siguientes motivaciones:

- Descentralización: Se ha visto en el estado del arte como los usuarios se encuentran distribuidos geográficamente y topológicamente por la red. Se confirma que la concentración de servicios en un único lugar no es óptima, ya que crea un punto único de error y proporciona calidades de servicio desiguales dependiendo de la ubicación del usuario. Sin embargo, los servicios descentralizados pueden trasladarse más cerca de los usuarios, mejorando así su QoE en cualquier punto de la red, máxime en una red comunitaria. Cloudy implementa un orquestador de servicios que usa un protocolo de *gossip* que permite a los nodos comunicarse entre ellos directamente sin la necesidad de un servidor central. Este mecanismo de comunicación descentralizado está basado en Serf, y se verá en capítulos posteriores.

  Para cumplir esta motivación, Cloudy implementa un orquestador de servicios que usa un protocolo de *gossip* que permite a los nodos comunicarse entre ellos directamente sin la necesidad de un servidor central. Este mecanismo de comunicación descentralizado está basado en Serf, y se verá en capítulos posteriores.
- Distribución: Con el fin de consolidar y fomentar la distribución de los servicios en las redes comunitarias el usuario requiere una plataforma donde publicarlos y descubrirlos. Se debe reducir al mínimo las acciones manuales y específicas, el acceso al servicio no debe depender de configuraciones de red estáticas y está al tanto de los cambios dinámicos de estado de las CNs (Community Networks).

Con el objetivo de cumplir esta motivación Cloudy incorpora DADS (Descubrimiento y Anuncio de Servicios Distribuido), una herramienta que utiliza un mecanismo de comunicación descentralizado.

- Facilidad de uso y experiencia de usuario: La accesibilidad y usabilidad son claves para ayudar a Cloudy a romper esa barrera tecnológica de entrada para el usuario final. Una interfaz web simple y accesible es hoy en día casi un requisito.

  Cloudy incorpora una plataforma web con un *front-end* de administración, configuración, y monitorización de los servicios *cloud* activos en el nodo. La instalación y configuración de los servicios en la nube habilitados en la distribución Cloudy se realiza de una manera ágil y sencilla.
- Software libre y abierto: La distribución de Cloudy está basada en Debian GNU/Linux, una de les distribuciones más populares. Además de cumplir con los requisitos técnicos, ha sido elegida ya que se encuentra bajo el marco del Debian Social Contract, el cual garantiza que el software incluido sea siempre abierto y libre.

3.2.1.2 Arquitectura Cloudy

A continuación, se describirá la arquitectura interna de la distribución Cloudy. En la figura 13, se puede ver como en la parte superior se encuentra la capa L2 virtual sobre la capa de red L3, que provee un *overlay* para interconectar todos los servicios en un microCloud. Sobre esta red overlay se apoyan los procesos de anuncio y descubrimiento de servicios, que publican y reciben información.

Los demonios de descubrimiento y de anuncio sondean periódicamente y actualizan sus listas de servicios, además, esta lista puede ser consultada directamente por los servicios o por el usuario.

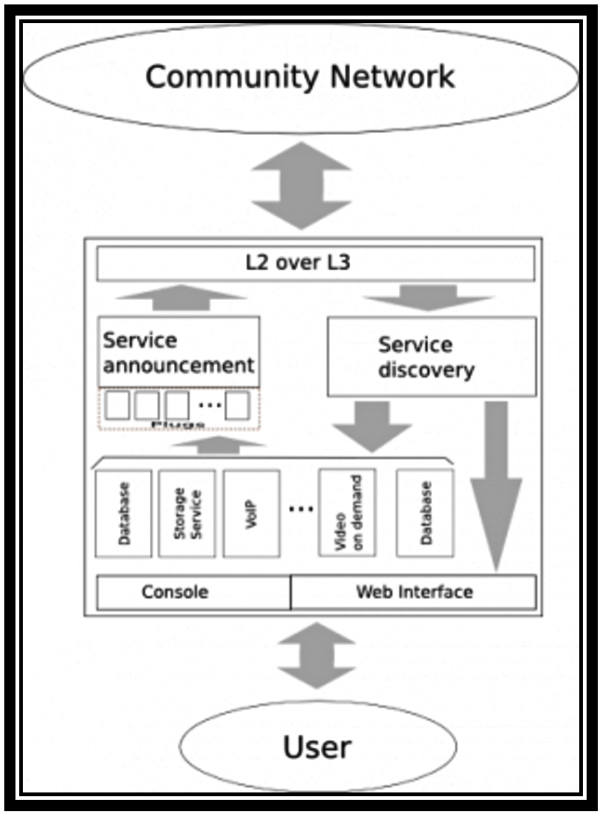


Figura 13: Arquitectura Cloudy [14]

4. Servicios distribuidos e IPFS

En este capítulo se presentan una serie de definiciones con el fin de profundizar en los servicios distribuidos, así como realizar un acercamiento a IPFS, explorar y desarrollar su aplicación en este ámbito.

Se puede describir, un servicio distribuido, como una aplicación con diversos componentes cuya ejecución se realiza en entornos separados, conectados entre sí, a través de una infraestructura de red.

La aparición y el éxito de los servicios distribuidos se ha debido a las ventajas que aportan sobre los servicios clásicos centralizados, y dentro de este ámbito conceptual, se trata IPFS.

IPFS (InterPlanetary File System) o Sistema de Archivos Interplanetario, es un protocolo Open Source diseñado para crear un método descentralizado y permanente destinado a almacenar y compartir archivos en una red. Tanto el protocolo como la red han sido desarrollados para crear un método *peer-to-peer* direccionable por contenido formando un DSN (Decentralized Storage Networks) [21] o red de almacenamiento descentralizado, es decir, se crea un método de

almacenamiento, y uso compartido de datos, en un sistema de archivos distribuido entre *peers*.

El desarrollo de IPFS se realiza bajo Protocol Labs [22] y la ayuda de la comunidad Open Source, aunque inicialmente fue diseñado por Juan Benet, quien es uno de los fundadores de Protocol Labs. IPFS combina iniciativas de Git, BitTorrent, Kademlia, SFS (Self-Certified Filesystems), y la Web. Es como un único *swarm* de Bittorrent, pero intercambiando objetos de tipo git. Por otro lado, IPFS proporciona una interfaz simple tipo web HTTP, pero con un sistema de permanencia implícito.

Figura 14: Mapeo lógico tipo pila OSI [23]

Para entrar y comprender IPFS en detalle sería recomendable la lectura del siguiente *white paper* [24], no obstante, este trabajo intenta realizar un acercamiento tanto a nivel teórico como práctico.1

Se podría decir, viendo la figura 14, que IPFS es similar en la capa de aplicación a la World Wide Web, pero IPFS proporciona un modelo de almacenamiento en bloque de alto rendimiento, y contenido direccionado con hipervínculos dirigidos a un contenido específico.

Para entender cómo funciona IPFS, y por qué se han referido previamente ciertas similitudes, se deben conocer las tecnologías subyacentes:

- DHT (Distributed Hash Tables): Una tabla DHT es un tipo de almacén o de tabla de hash que contiene pares (clave, valor), y permite consultar el valor

asociado a una clave, donde los datos se almacenan de forma distribuida en una serie de nodos. En el caso de IPFS, la clave es un hash sobre el contenido, por lo tanto, si se solicita a un nodo de IPFS el contenido mediante un hash valido, el nodo busca en la tabla DHT quienes tienen ese contenido solicitado con el hash. La manera en la que se encuentra un valor específico, de manera eficiente, y cómo se administran las DHT para que los cambios se detecten con el menor impacto posible, depende de las diferentes implementaciones de DHT que existen. Algunas de ellas son Chord, Tapestry, Kademila o Pastry, en las siguientes referencias [25], [26], se puede observar el funcionamiento del routing en Pastry.

- Block Exchanges - Bit Torrent: Aunque sea una red de intercambio P2P conocida, existen una serie de referencias que pueden ayudar notablemente a entender el funcionamiento en detalle de Bit Torrent, y por qué interesa para IPFS; redes peer-to-peer con Bit Torrent [27], y la especificación del protocolo Bit Torrent [28].

  El intercambio de datos por bloques en IPFS está inspirado en BitTorrent, pero no es exactamente igual. A continuación, se indican dos características de BitTorrent que utiliza IPFS:

  - Estrategia *tit-for-tat*. Basada en compartir para recibir o en que, si no se comparte, no se recibe.
  - Obtener los fragmentos de piezas raras primero. Este algoritmo hace que el cliente busque la pieza más rara disponible entre los *peers*, debido a que estima que las más comunes estarán disponibles al final.

  Una de las diferencias notable es que en BitTorrent cada archivo tiene un *swarm* de *peers* separados, formando una red P2P entre sí, mientras que en IPFS se da un gran *swarm* de *peers* para todos los datos.

  Como se ve en los capítulos de despliegue y de evaluación de IPFS, una vez se inicia el demonio, se conecta a unos *peers* denominados semilla y, al cabo de un breve periodo de tiempo, el número de *peers* crece notablemente.

- Version Control Systems - Git: Los sistemas de control de versiones aportan la facilidad de modelar archivos que son dinámicos, es decir, que se van modificando con el tiempo, además, se encargan de gestionar y distribuir versiones diferentes de una manera eficiente. El sistema de control de versiones Git proporciona un modelo de objetos Merkle DAG [29], que recoge

las modificaciones en un árbol del sistema de ficheros y es utilizado también en IPFS.

- SFS (Self-certified FileSystems): Se aplica con el fin de diseñar un sistema de archivos auto certificados, direccionando sistemas de archivos remotos mediante la utilización de un esquema formado por /sfs/<location>:<HostID>, donde el HostID es el hash (public_key || Location), lo que sirve para implementar el sistema de nombres IPNS para IPFS, permitiendo generar una dirección para un sistema de archivos remoto, donde el usuario puede verificar la validez de la dirección.

Por lo tanto, en la fecha de realización de este trabajo, se puede resumir que IPFS combina una tabla hash distribuida, un intercambio de bloques incentivado, y un espacio de nombres de auto certificación, eliminando puntos de fallo, y aportando confiabilidad entre los nodos. Por otro lado, como veremos en capítulos posteriores, se puede agregar un archivo local cualquiera al sistema de archivos IPFS, poniéndolo a disposición de la red global. También se puede acceder al sistema de archivos por diferentes vías, incluso a través de FUSE [30] y de HTTP. Los archivos agregados a la red IPFS se identifican por sus valores hash, lo que no facilita recordar su nombre, pero si facilita la tarea de almacenaje en cache, además, esta arquitectura optimiza el ancho de banda y previene ataques DDoS, un problema que HTTP intenta solucionar.

4.1 Diseño IPFS

Como se ha ido viendo, la contribución de IPFS consiste en simplificar, evolucionar y conectar técnicas probadas en un solo sistema cohesivo, cuyo beneficio sea mayor que la suma de las partes que lo componen. Como se ve a continuación, IPFS presenta una nueva plataforma para escribir y desplegar aplicaciones, y un nuevo sistema para distribuir y versionar grandes cantidades de datos.

El Protocolo IPFS, tal y como se observado en el capítulo previo, está dividido en una serie de capas o tecnologías subyacentes, a continuación, se explica esa pila de sub-protocolos responsables de diferentes funcionalidades:

- *Identities*: Se gestiona la generación de un nodo, su nombre e identidad.

- *Network*: Se gestionan las conexiones con otros *peers* de la red, es decir, se anuncian y se encuentran objetos. Se utilizan diversos protocolos de red como pueden ser uTP, SCTP, LEDBAT, etc.
- *Routing*: Mantiene información útil para la localización de objetos y otros nodos. Responde tanto a consultas locales como remotas.
- *Exchange:* Se utiliza el protocolo de intercambio de bloques BitSwap, que gestiona la distribución de bloques de manera eficiente. Se intercambia la información. Se modela para incentivar la replicación de datos.
- *Objects:* Se encarga de organizar la información mediante un DAG de Merkle de objetos inmutables direccionados por contenido que portan enlaces. Representa estructuras de datos arbitrarias como jerarquías de archivos.
- *Files:* Proporciona una jerarquía del sistema de archivos versionada inspirada por Git
- *Naming:* Se añade la mutabilidad, sistema de nombres mutable y auto certificado.

Estos subsistemas no son independientes, están integrados entre sí aprovechando las sinergias y propiedades comunes.

Esta pila se puede ver reflejada gráficamente n la figura 15.

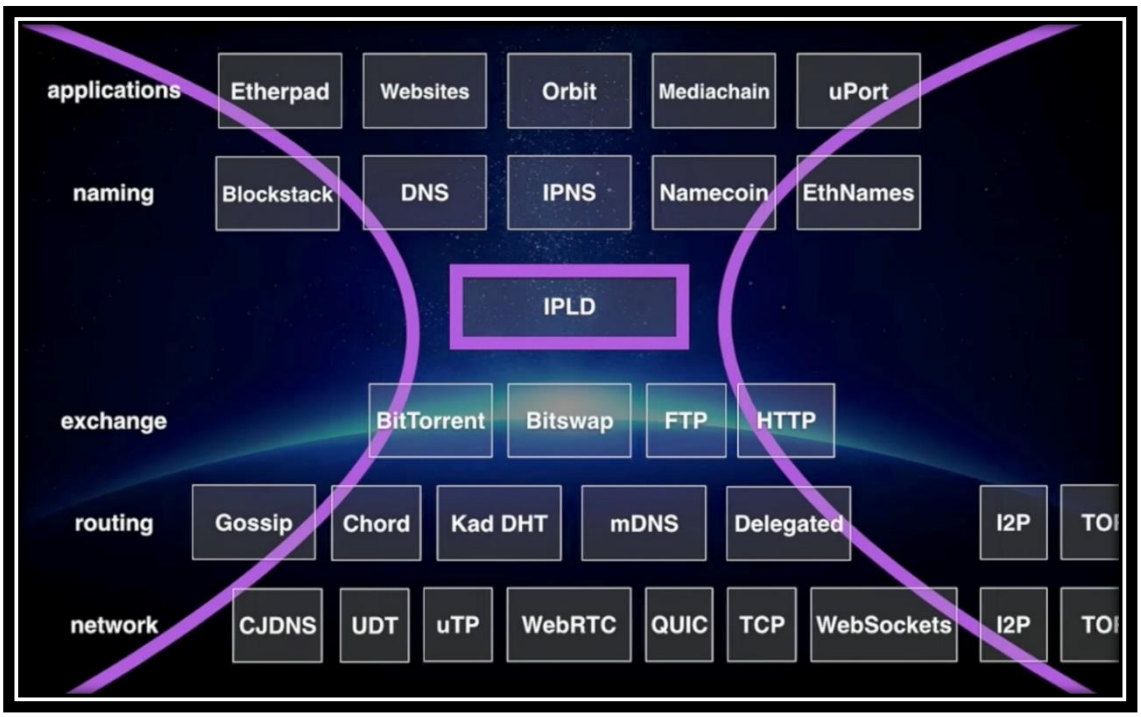


Figura 15: IPFS stack [31]

4.2 IPFS e Internet

En la actualidad, el protocolo para intercambio de información por excelencia es HTTP, este protocolo es, sin duda, una de las palabras que le viene a la gente cuando se habla de la Web o de Internet. En este capítulo se intenta describir la premisa que se encuentra en diversos artículos sobre IPFS, posicionándolo como la nueva web 3.0 o uno de los cambios importantes de Internet. Se conoce que la arquitectura actual de Internet no posee los niveles de descentralización con los que se concibió inicialmente, además, algunos de los protocolos utilizados son considerados obsoletos, no cumplen, de manera eficiente, la utilización que se le da hoy en día, y el acceso al contenido mediante su ubicación a través de una URL genera una serie de limitaciones. No obstante, aunque se indican algunas de estas limitaciones, el protocolo HTTP ha permitido el desarrollo de Internet hasta el día de hoy. Algunos de estos problemas se pueden resumir en:

- Contenido se ubicado en un único punto accesible mediante URL, los servidores deben estar capacitados para recibir altos niveles de tráfico, lo que en ocasiones puede generar problemas de latencia debido a sobrecarga en la conexión al servidor. Por un lado, se tiene un consumo de ancho de banda ineficiente, empleo innecesario de capacidad de almacenamiento, y por otro, un incremento de la escalabilidad vertical, en vez de la horizontal, aumentando la centralización de servidores y proveedores de servicios, lo cual enfoca Internet hacia un mayor control de la información desde un menor número de puntos.
- El acceso al contenido mediante una ubicación fija o establecida genera un único punto de fallo, por lo que, ante un fallo, la información dejaría de estar accesible.
- La volatilidad de la información en la red genera que la información se pueda perder para siempre, traduciéndose en inconsistencias en los direccionamientos por URL, *Links* rotos. El historial de la información almacenada es breve, teniendo en cuenta la vida de las páginas webs, de cien días de promedio.

- Conexiones intermitentes y lentas. La red IP funciona en un modo *best-effort.* Numerosos saltos de red, diferentes tasas de transferencia, protocolos de routing IGP y EGP, etc.

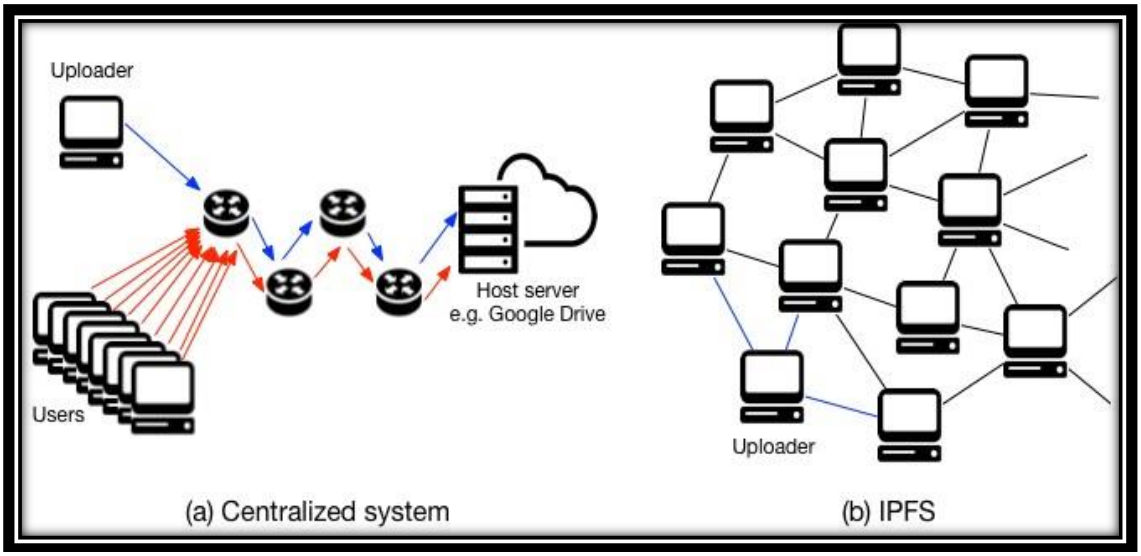


Figura 16: Flujo de datos en modelo cliente-servidor centralizado vs IPFS [32]

IPFS proporciona el almacenamiento distribuido, y el sistema de archivos que Internet necesita para alcanzar su verdadero potencial. Se calcula que, al transmitir vídeo en IPFS, se ahorraría el 60% del ancho de banda.

Como se ha visto hasta ahora, en lugar de descargar archivos de servidores individuales en HTTP, aún con balanceo de carga, en IPFS se solicitan esos ficheros a sus *peers* en la red, proporcionando una ruta de acceso a un archivo en vez de una única ubicación, lo que permite la distribución de datos a gran escala. Por otro lado, esto proporciona una mayor eficiencia, mantener un histórico de versiones, y redes más resistentes a la censura, aunque esté en una versión beta, y queden muchos aspectos de seguridad por trabajar, el contenido es asegurado y verificado a través de hashes criptográficos.

Por lo tanto, IPFS es una visión ambiciosa de la nueva infraestructura de una Internet descentralizada, sobre la cual se pueden diseñar y desarrollar nuevas aplicaciones o servicios, como los que veremos en los subcapítulos posteriores. Actualmente se puede utilizar, y así se verá en este trabajo, como mínimo, un sistema de archivos y espacio de nombres global, montado y versionado, también, como el sistema de intercambio de archivos de la próxima generación. En el mejor de los casos, tal y como indica Juan Benet en el capítulo de ideas de futuro de su tercer borrador: “podría llevar a la web a nuevos horizontes, donde

la publicación de información valiosa no impone el hospedaje en el editor, sino en aquellos interesados, donde los usuarios pueden confiar en el contenido que reciben sin confiar en los compañeros de quienes lo reciben, y en los sitios antiguos donde los archivos importantes no se pierden. IPFS espera poder llevarnos hacia la web permanente."

En definitiva, y como se ha indicado en diversos capítulos durante el desarrollo de este trabajo, se considera que cada día son más los dispositivos con acceso a Internet, por lo tanto, es necesario optimizar el uso de recursos en el b*ackbone* de Internet, de ahí que el uso de un sistema descentralizado de información parece una buena aproximación. En los capítulos posteriores de instalación, despliegue y evaluación, se ve como IPFS cuenta con una suite completa para ser utilizado, incluyendo un cliente en terminal vía CLI, y acceso a archivos mediante navegador web. No obstante, el principal *trade-off* de este protocolo es que no se puede borrar información de la red, por lo que, si algún usuario comparte información de manera accidental, no puede arrepentirse de dicha acción. De hecho, este es un mensaje que se encuentra anclado en los canales de chat de Protocol Labs a modo de aviso. Otra desventaja es el acceso a la información en capa de aplicación, pues al utilizar hashes largos no es útil o memorizable a la hora de compartir, lo que se conoce como *human-friendly naming*, no obstante, este aspecto aporta cierto nivel de seguridad y evita, en cierto modo, censura y control de tráfico.

4.3 Servicio IPFS Cluster

IPFS Cluster es un software, desarrollado por Protocol Labs, dedicado a orquestar demonios IPFS que se ejecutan en los diferentes hosts. Un grupo de IPFS Cluster está formado por varios *peers*, cada uno de ellos asociado a un demonio de IPFS, requisito para poder formar parte del grupo o *swarm.* Los *peers* comparten un *pinset*, también conocido como estado compartido, que enumera los CID que están anclados en cada clúster, y sus propiedades: asignaciones, factor de replicación, etc. Los pares de cada clúster se comunican utilizando libp2p (*cluster swarm*), de manera similar a IPFS, pero independientes de él. Por lo tanto, cada peer del clúster necesita su propia clave privada, diferente de la utilizada por el demonio IPFS, además, de su *ID peer* propio.

Todos los miembros del clúster comparten una clave secreta adicional que garantiza que solo puedan comunicarse con partes conocidas.

El protocolo IPFS es utilizado por Protocol Labs para mantener y replicar un conjunto de pines grandes, a través de integraciones como el Pinbot IPFS IRC [33].

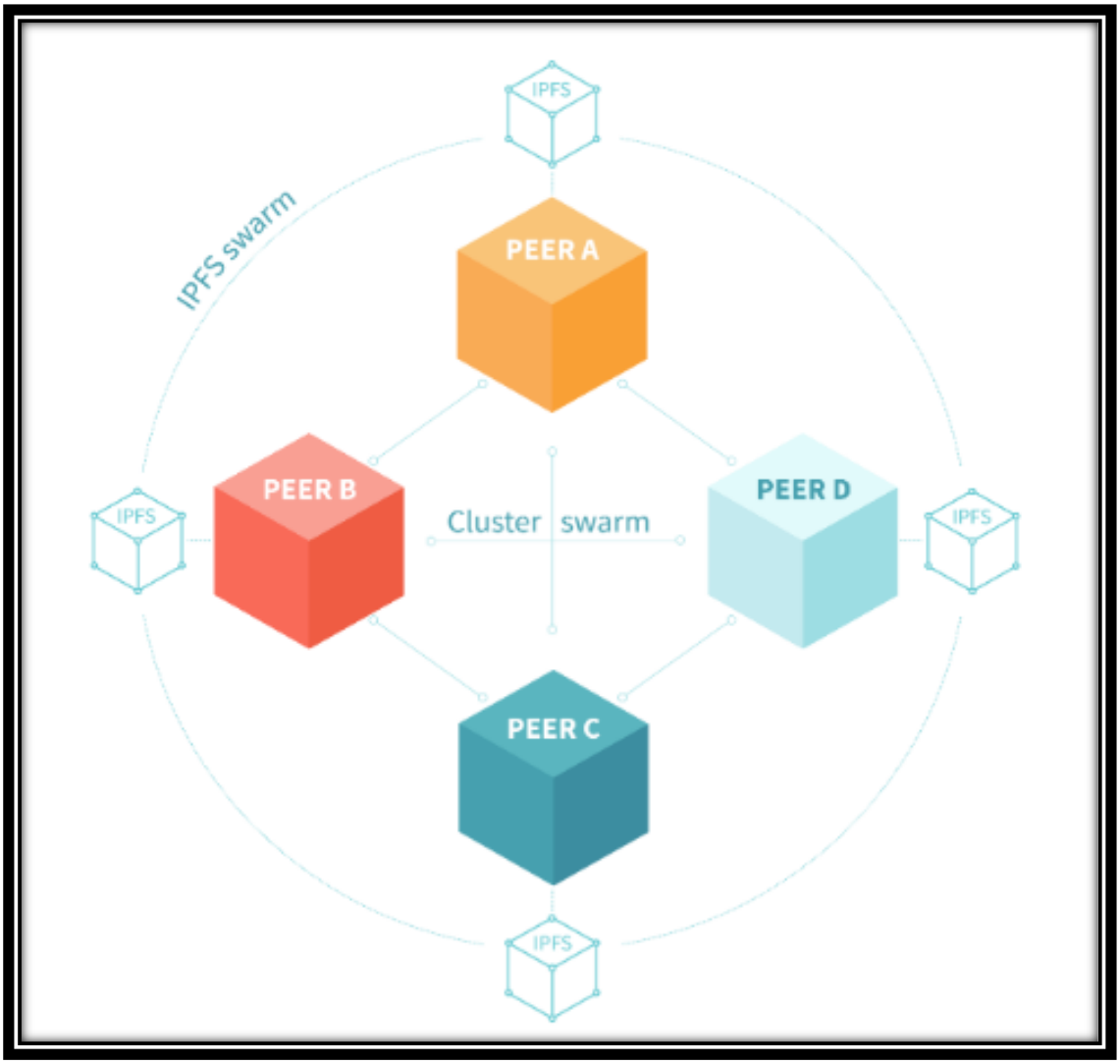


Figura 17: IPFS Cluster. Swarm IPFS [34]

Las características de la última versión estable de IPFS-Cluster, la v0.70 durante el desarrollo de este trabajo, se pueden resumir en:

- Se agrega, se replica y se fija, de forma directa, el contenido a múltiples *peers* de IPFS a la vez, a través de Clúster.
- Se obtiene el contenido de múltiples pares de IPFS a través del clúster.
- Existe una capa de consenso, basada en el algoritmo distribuido Raft [35], que aporta una protección ante segmentación de red y redireccionamiento

automático del líder: Cada *peer* del clúster de IPFS puede controlar el clúster, modificar el conjunto de pines y realizar cualquier operación.

- Pines distribuidos uniformemente de acuerdo con el espacio del repositorio de cada demonio IPFS.
- Opciones de configuración parametrizables y flexibles, se permiten conexiones de alta latencia.
- Proceso de migración viable entre versiones estables.
- *Pin-set* exportables e importables, útil para las migraciones de clúster.
- Los grupos pueden crecer (se pueden agregar nuevos compañeros) y disminuir (los compañeros se pueden eliminar) sin necesidad de tiempo de inactividad.
- Soporte DNS-multidireccional.
- HTTPs y autenticación básica soportada.
- Cliente Go API con soporte completo.
- Se ejecuta independientemente de IPFS, utilizando la API de go-ipfs (generalmente sobre tcp / 5001), para controlar el demonio de IPFS.
- Documentación actualizada, incluida la documentación centrada en implementaciones de producción de IPFS y IPFS Cluster.

Por otro lado, existen una serie de limitaciones y problemas que se indican a continuación:

- IPFS Cluster no admite el *pinning* colaborativo con individuos aleatorios que se suscriben a un conjunto de pines y, por lo tanto, contribuyen con su espacio en disco para almacenar datos interesantes.
- No están claros los límites de escalabilidad:
    - Probado con clúster de diez *peers* en una configuración global:
        - Tamaño del repositorio de alrededor de 70 GB / cada uno.
        - ~ 2000 pins / peer
    - Probado con clúster de 5 *peers* en una configuración regional:
        - Disco de 44 TB
        - ~ 5000 entradas en el conjunto de *pins*
- No hay *bootstrappers* públicos. Los nuevos *peers* deben arrancar de clústeres ya existentes.

En el capítulo de evaluación se realiza una exposición sobre los límites de capacidad, y el despliegue de un clúster regional formado por varios *peers,* posteriormente, se realiza un escalado a 6 *peer*s.

4.4 Servicio IPFS-Log

IPFS-Log es una estructura de datos replicada (CRDT) inmutable, basada en operaciones y libre de conflictos enfocada al uso en sistemas distribuidos. Es un registro de solo aplicación que se puede usar para modelar un estado mutable, y compartido entre pares en aplicaciones p2p. Cada entrada en el registro de logs queda registrada en IPFS, y cada una apunta a un hash de entradas anteriores que forman un gráfico, donde las ramas se pueden bifurcar y unir de nuevo. Este módulo funciona en Node.js.

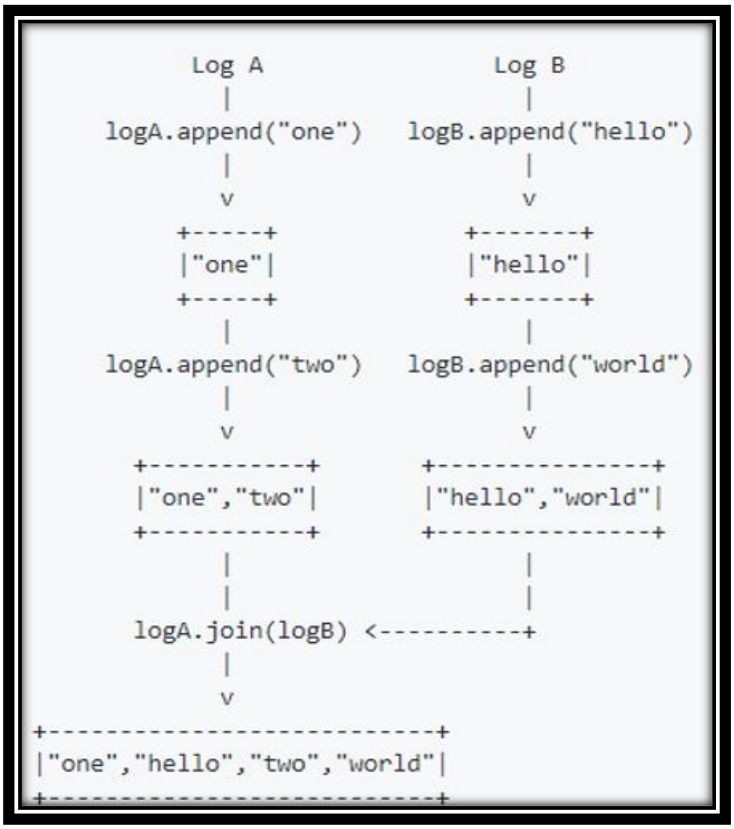


Figura 18: Orbit Log [36]

IPFS Log tiene algunos casos de uso definidos:

- CRDTs
- Registro de operaciones de la base de datos
- *Feed* de datos

- Rastrea las diferentes versiones de un archivo
- Mensajería

Originalmente fue creado para OrbitDB.

4.5 IPFS en Cloudy

IPFS está integrado en Cloudy [37], de forma limitada, como servicio de soporte a la distribución. Para publicar los servicios localmente utiliza Avahi, mientras que en otros nodos de la red comunitaria utiliza Serf. Este se encarga, con IPFS, de realizar el intercambio de hashes en la publicación de servicios. Esta integración, de cara a aprovechar el potencial de IPFS, pretende ser aún más escalable.

4.6 Servicio Orbit

Orbit [38], es una aplicación distribuida de chat *peer-to-peer* desarrollada sobre IPFS. Se debe tener en cuenta que, a fecha de elaboración de este trabajo, Orbit está en una fase experimental y dispone de 2 tipos de cliente:

- Web: Orbit-web
- Escritorio: Orbit-electron

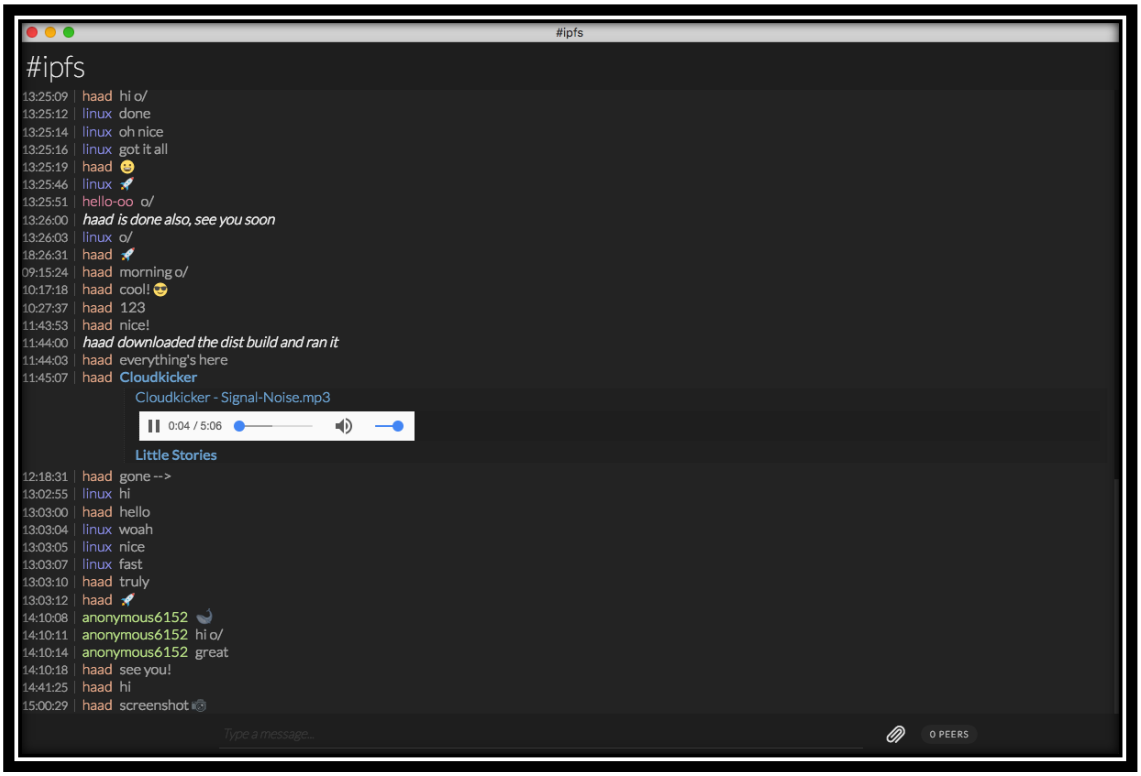


Figura 19: Orbit chat [38]

La versión de escritorio de Orbit puede funcionar tanto en OSX como en GNU/Linux, por otro lado, se está trabajando en el soporte para Windows.

Orbit tiene una serie de librerías necesarias para trabajar adecuadamente:

- Orbit-Core: Implementación en JavaScript destinada a ser embebida en diversas aplicaciones. Se puede utilizar en Node.js, Electron y otros sitios web.
- Orbit-db es una base de datos distribuida, punto a punto, que utiliza IPFS como su almacenamiento de datos, e IPFS Pubsub para sincronizar automáticamente las bases de datos con los *peers.*
- Js-ipfs: Nuevo protocolo de hipermedia p2p. Utiliza IPFS como almacenamiento de datos y libp2p para gestionar todas las redes p2p.

Orbit puede trabajar con CRDTs (Conflict-free Replicated Data Type), un tipo de datos replicado sin conflictos. Representa una estructura de datos que se puede replicar en múltiples nodos dentro de en una red, donde las réplicas se pueden actualizar de forma independiente, y concurrente, sin coordinación entre ellas, y donde siempre se pueden resolver las inconsistencias que se puedan generar.

El concepto CRDT es definido formalmente en 2011 por Marc Shapiro, Nuno Preguiça, Carlos Baquero y Marek Zawirski, motivado por la edición colaborativa

de texto y la computación móvil. Los CRDT también se han utilizado en sistemas de chat y juegos de azar *on-line*, en la plataforma de distribución de audio SoundCloud. etc. Por otro lado, otras bases de datos distribuidas NoSQL Redis y Riak tienen tipos de datos CRDT.

Referente a OrbitDB, se dedica un subcapítulo, a continuación, ya que es un proyecto independiente y se considera de notable importancia para este TFG.

4.7 Servicio OrbitDB

Existen una serie de confusiones con los nombres de Orbit; por un lado, se hace referencia al proyecto de desarrollo global, y por otro, se refiere al servicio de chat sobre IPFS que hemos visto en el subcapítulo anterior. Este último se refiere también en Internet como Orbit Chat u Orbit Web, para diferenciarlo del proyecto general.

En el caso de OrbitDB, como se ha indicado previamente, se puede definir como una base de datos distribuida, p2p, que utiliza IPFS como su almacenamiento de datos. Es una base de datos eventualmente consistente que utiliza CRDT para las combinaciones de bases de datos sin conflictos, lo que convierte a OrbitDB en una excelente opción para aplicaciones descentralizadas (*dApps*), diferentes aplicaciones de *blockchain*, y otras aplicaciones web de tipo *offline-first*.

OrbitDB proporciona varios tipos de bases de datos para diferentes modelos de datos y casos de uso:

- Log: un registro inmutable (solo de apéndice) con historial transversal. Utilizada, por ejemplo, por IPFS-Log y útil para los "últimos N” casos de uso o como cola de mensajes.
- Feed: un registro mutable con historial transversal. Las entradas se pueden agregar y eliminar, siendo útil para el tipo de casos de uso de "carro de la compra" o como fuente de publicaciones de blog o *tweets*.
- Keyvalue: una base de datos clave-valor.
- Docs: una base de datos de documentos en la cual se pueden almacenar e indexar documentos JSON mediante una clave específica. Es útil para crear índices de búsqueda o como control de versiones de documentos y datos.
- Counter: Base datos útil para contar eventos separados de los datos de tipo Log/Feed.

Todas las bases de datos se implementan sobre IPFS-log, y el estado del proyecto se encuentra en desarrollo activo en fase alfa. Esto quiere decir, aunque dentro del marco del TFG no es relevante, que OrbitDB no ha sido auditado por áreas de seguridad, y las API de programación y los formatos de datos aún pueden cambiar.

4.8 Servicio eChat

eChat [39], es un servicio de mensajería basado en una tecnología descentralizada. Se tiene en cuenta y realiza una investigación sobre eChat debido a que está basado en IPFS, no obstante, únicamente está disponible una versión beta a nivel de aplicación cliente para descargar y utilizar en el App Store y Google Play. Por lo tanto, se hace referencia en este capítulo ya que se considera significativo, debido a que utiliza una metodología independiente de servidores centralizados e incorpora protocolos específicos de seguridad y privacidad, pero se desestima de cara a entrar más en detalle en este trabajo.

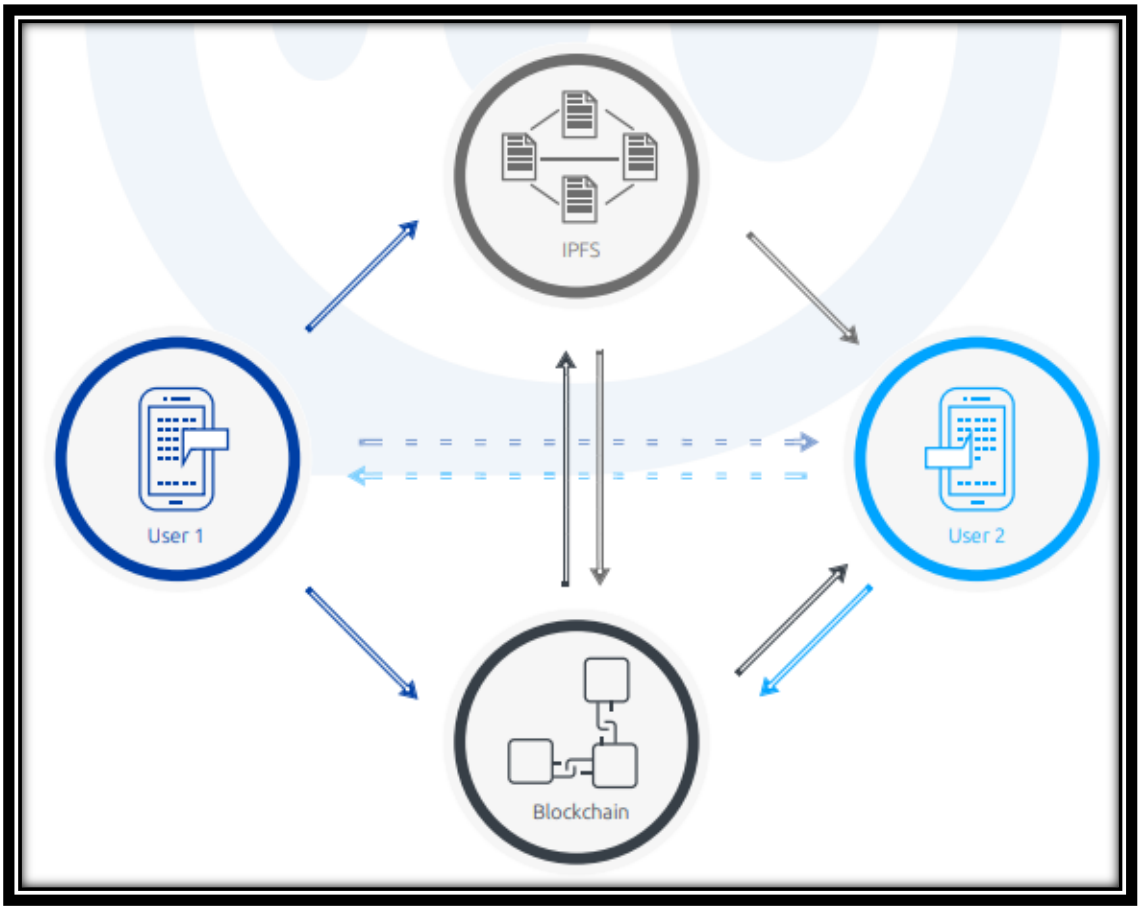


Figura 20: eChat IPFS y Blockchain [39]

4.9 Serf

Serf es una herramienta destinada a trabajar entre miembros de un clúster, proporcionando detección de fallos y una orquestación descentralizada. Es tolerante a fallos y diseñada para trabajar en entornos de alta disponibilidad. Se utiliza en las principales plataformas como GNU/Linux, Mac OS X, Windows, y los requisitos hardware que necesita son mínimos; utiliza de 5 a 10 MB de memoria y se comunica principalmente mediante tráfico UDP.

Serf utiliza un protocolo denominado Gossip protocol, con el fin resolver tres problemas principales:

- Miembros: Mantiene las listas de miembros del clúster y puede ejecutar scripts personalizados cuando el estado del clúster o el número de miembros cambie. Por ejemplo, Serf puede mantener una lista de servidores web para un balanceador de carga, y notificarle cuando un nodo está disponible o no para balancear.

- Detección y recuperación de fallos: Detecta de manera automática y proactiva los nodos en fallo, y acto seguido, notifica al resto del clúster y ejecuta las secuencias de comandos definidas para gestionar este tipo de eventos, por ejemplo, diagnóstico, reconexión, etc.
- Propagación de eventos personalizada: Puede transmitir eventos y consultas personalizados al clúster. Se pueden utilizar para automatizar despliegues, propagar configuraciones, etc.

Gossip protocol está basado en SWIM (Scalable Weakly-consistent Infection-style process group Membership protocol), y se utiliza para realizar broadcast de mensajes dentro del clúster.

4.10 Serf en Cloudy

Serf está integrado con Cloudy, tal y como se puede ver en el interfaz web, con el objetivo de transmitir en sus mensajes que servicios están disponibles en cada nodo. La solución implementada pasa por subir la información sobre los servicios en IPFS, y enviar el hash generado vía Serf.

El destinatario utiliza ese hash de IPFS recibido para acceder a la información sobre el servicio.

5. Implantación: Instalación, configuración y despliegue

En este capítulo se desarrollan las diferentes fases de instalación, despliegue y evaluación del entorno. El primer paso, después del conexionado físico de las RPI, es la instalación del sistema operativo y software necesario para iniciar el entorno del proyecto, acto seguido, se irá desplegando el software adicional para trabajar con contenedores en GNU/Linux y aportando capas de valor añadido al entorno.

5.1 Instalación del sistema operativo

El sistema operativo elegido será la distribución de GNU/Linux Raspbian, basada en Debian Stretch y parametrizada para una arquitectura ARM. En primer lugar, se realizará la implementación en una de las microSD destinadas para el

proyecto y dedicadas a las Raspberry, y posteriormente, se realizará la configuración de Raspbian.

La configuración de las Raspberry se puede realizar en local, conectados al propio dispositivo mediante periféricos de entrada y de salida; como un teclado, ratón y monitor o en remoto accediendo vía SSH.

Si se dispone de un *stack* considerable de RPI, la opción de configuración vía acceso remoto por SSH cobra peso. En la siguiente referencia web [40], de Hackernoon, se indican una serie de instrucciones para realizarlo.

Se configuran los dispositivos como “lCloudyX”, para los dispositivos locales, y “rCloudyX” en el caso de los dispositivos remotos. Por lo tanto, se realiza la siguiente configuración para el primer nodo, mCloud1, utilizando la aplicación “raspi-config”.

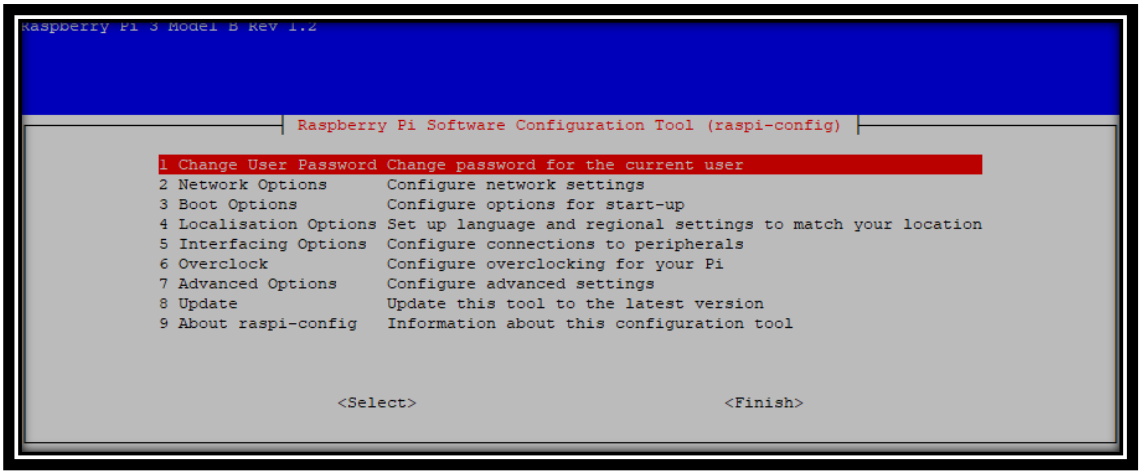


Figura 21: Nodo lCloudy1 sistema

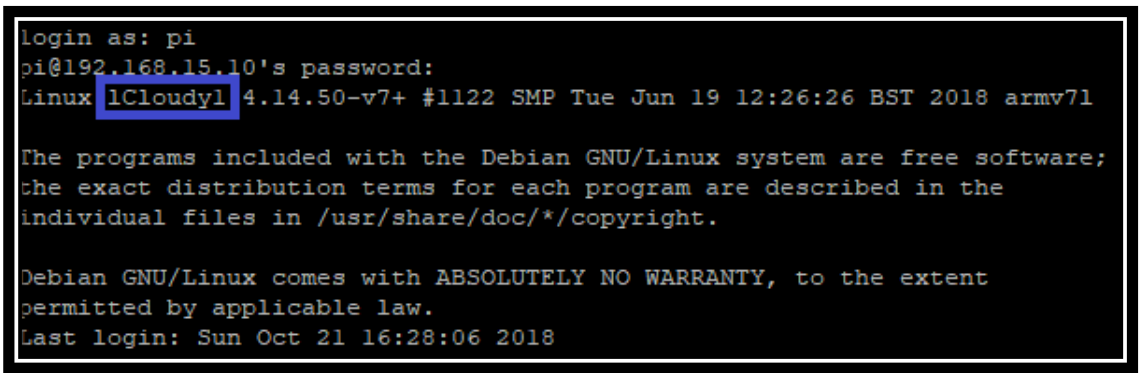


Figura 22: Nodo lCloudy1 sistema

A nivel de red, al estar configurado el DHCP server en el Cisco EPC, y el cliente DHCP en las Raspbian, se dispone de conectividad IP, no obstante, se fija esta

configuración de red para que no se padezcan cambios a nivel IP durante el proyecto.

```
# Example static IP configuration:
interface eth0
static ip_address=192.168.15.10/24
static ip6_address=fd51:42f8:caae:d92e::ff/64
static routers=192.168.15.1
static domain_name_servers=192.168.15.1 8.8.8.8 fd51:42f8:caae:d92e::1
```

Figura 23: Nodo lCloudy1 red

Esta configuración se realiza también el en el segundo nodo. En un primer momento, se puede ver el resumen con los nodos conectados a la red local previa configuración, y a continuación, desde el equipo de red; Cisco EPC3925, con los nodos de la microCloud ya configurados a nivel de parámetros de red.

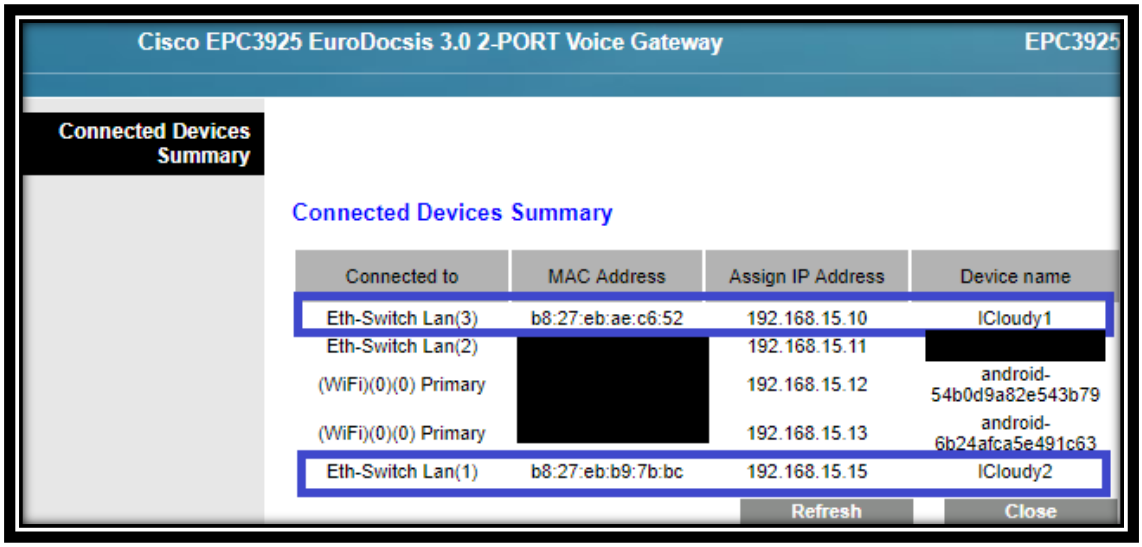


Figura 24: Web GUI Cisco EPC. Nodos lCloudyX

5.2 Instalación Cloudy. “Cloudynizar”

Una vez se ha instalado y configurado Raspbian, se realiza la acción denominada como Cloudynizar, convirtiendo la distribución en un dispositivo Cloudy, mediante la ejecución del siguiente *script* “Cloudynitzar.sh”:

```
pi@lCloudy1:~ $ sudo apt-get update; sudo apt-get install -y curl lsb-release
Hit:1 http://archive.raspberrypi.org/debian stretch InRelease
Hit:2 http://raspbian.raspberrypi.org/raspbian stretch InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
lsb-release is already the newest version (9.20161125+rpi1).
lsb-release set to manually installed.
The following packages will be upgraded:
  curl libcurl3
2 upgraded, 0 newly installed, 0 to remove and 112 not upgraded.
```

```
pi@lCloudy1:~ $ sudo curl -k https://raw.githubusercontent.com/Clommunity/cloudynitzar/master/cloudynitzar.sh | bash -
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  6530  100  6530    0     0  20713      0 --:--:-- --:--:-- --:--:-- 20796
```

Figura 25: Cloydynitzar

Este script está disponible en la referencia [41] de Github, además de otros *scripts* para comprobar el funcionamiento de Cloudy, no obstante, se puede realizar la prueba de que está funcionando correctamente realizando una conexión http al puerto 7000 mediante el navegador:

Desde la pantalla de login se accede mediante el usuario y contraseña que se haya creado en nuestro sistema operativo. En el caso de Raspbian el usuario "pi" está creado por defecto y el password se había cambiado en la parametrización del sistema.

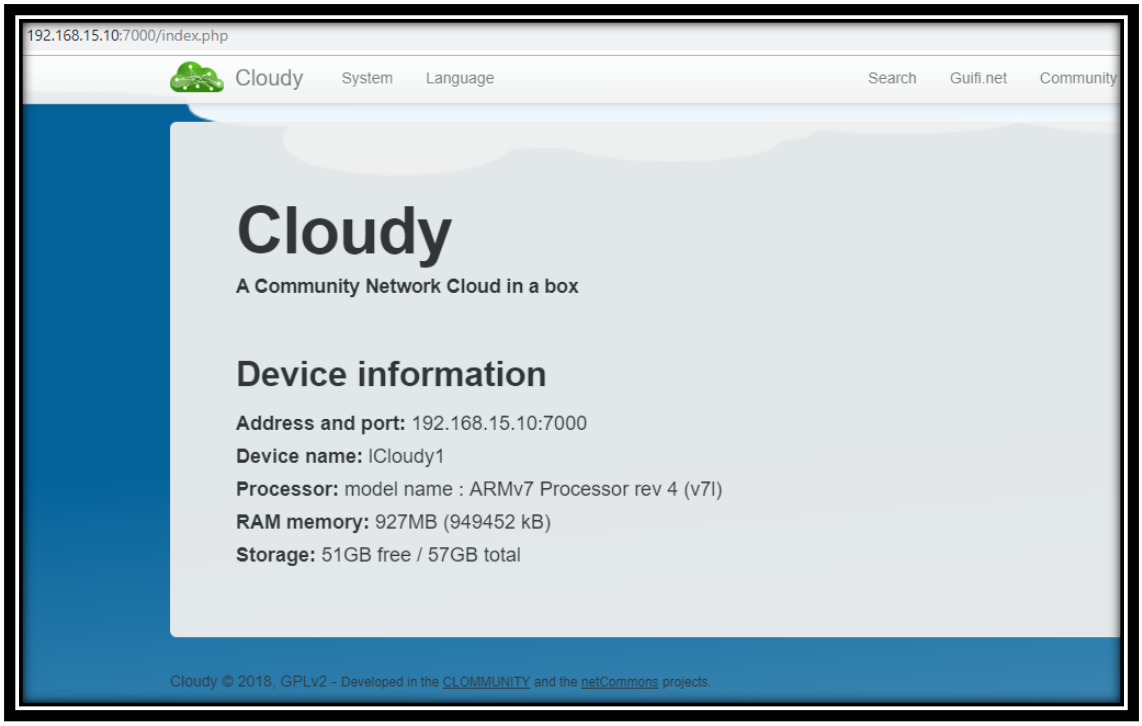


Figura 26: WebGUI Cloudy lCloudy1

5.3 Instalación Docker

La instalación de Docker se puede realizar desde el interfaz web de Cloudy, debido a que este proceso se encuentra integrado en las versiones utilizadas en este proyecto, es decir, se puede realizar la instalación y activación desde el portal web de Cloudy en el menú “Enterprise Cloud”.

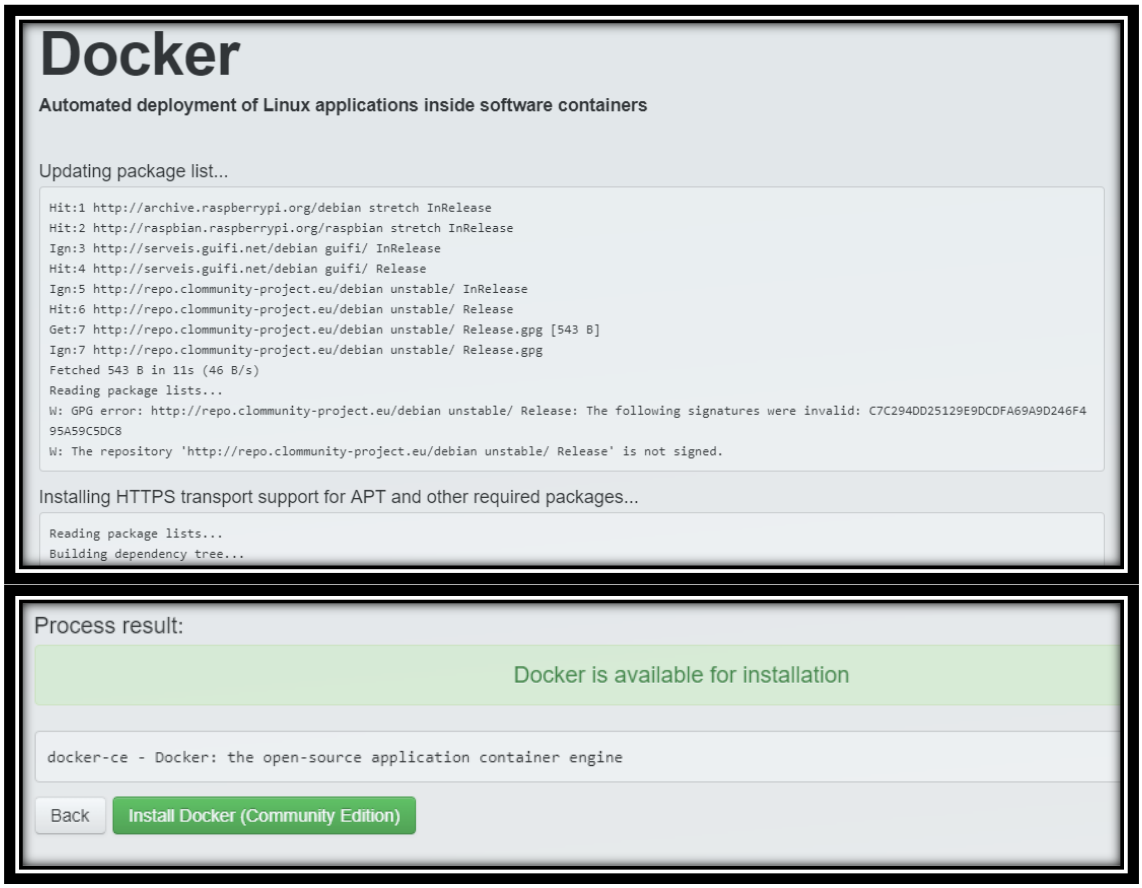


Figura 27: Instalación Docker desde Cloudy

Desde dicho menú se accede a una página específica donde se indica la carencia de fuentes de Docker dentro del gestor de paquetes. Una vez se haya ejecutado su instalación, Cloudy nos indicará que se puede instalar el software, y a continuación, se procede a la instalación del paquete Docker-ce.
Una vez se haya realizado la instalación de Docker Community Edition se comprueba el estado de este, tal y como se puede ver en la figura 16.

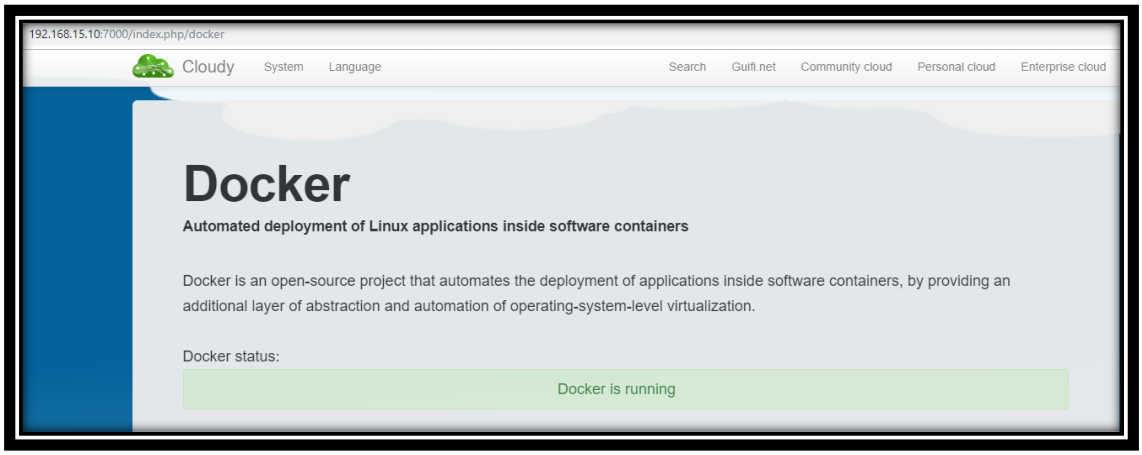


Figura 28: Estado Docker Cloudy

5.4 Instalación *software* adicional

En este capítulo se llevan a cabo una serie de instalaciones de *software* necesario para adecuar el entorno de trabajo.

5.4.1 OpenVPN

Se realiza la instalación de OpenVPN mediante la ejecución de secuencia de comandos apt-get:

- apt-get install openvpn

Posteriormente, se realiza la activación del servicio de OpenVPN:

- service openvpn start

```
leopoldo@ubuntucloudy16:/etc/openvpn$ service openvpn status
● openvpn.service - OpenVPN service
   Loaded: loaded (/lib/systemd/system/openvpn.service; enabled; vendor preset: enabled)
   Active: active (exited) since vie 2018-10-19 19:54:15 CEST; 16min ago
  Process: 762 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
 Main PID: 762 (code=exited, status=0/SUCCESS)
    Tasks: 0
   Memory: 0B
      CPU: 0
   CGroup: /system.slice/openvpn.service

oct 19 19:54:15 ubuntucloudy16 systemd[1]: Starting OpenVPN service...
oct 19 19:54:15 ubuntucloudy16 systemd[1]: Started OpenVPN service.
leopoldo@ubuntucloudy16:/etc/openvpn$
```

Figura 29: Estado OpenVPN

A continuación se realizan las parametrizaciones oportunes en el fichero de configuración:

- /etc/openvpn/client.conf

5.4.2 XRDP y TightVNC

Aunque esta instalación es opcional, facilita las tareas administrativas poder acceder por escritorio remoto a nuestras RPI, además de la conexión por consola vía SSH. La instalación para acceder por escritorio remoto se realiza de la siguiente manera:

- sudo apt-get update
- sudo apt-get upgrade
- sudo apt-get install xrdp tightvncserver

Una vez se instala el software, el servidor de acceso remoto está funcionando en las Raspberry del entorno.

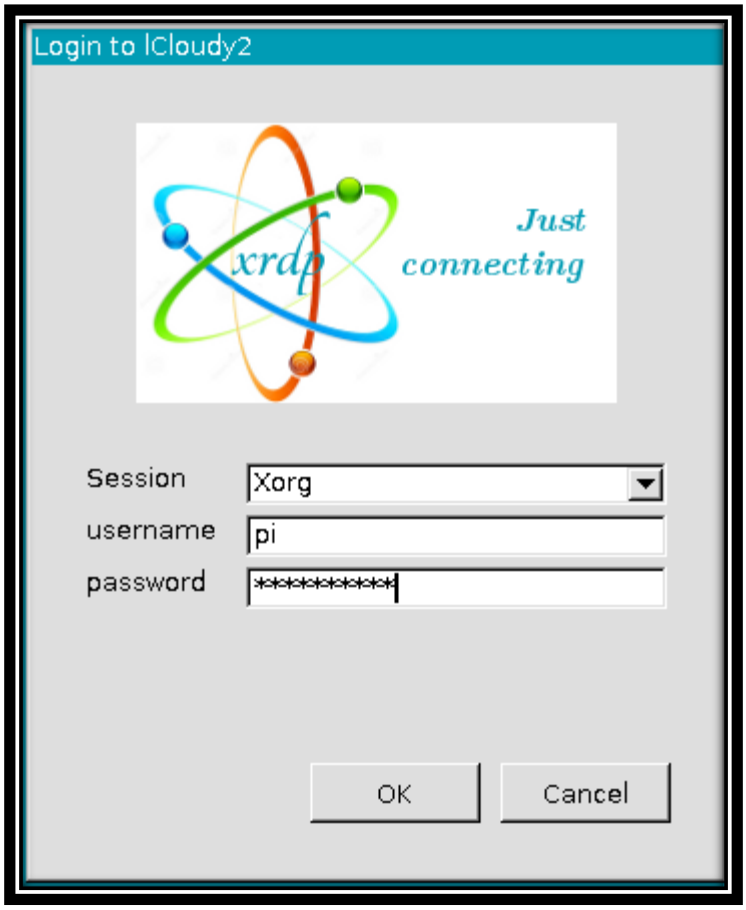


Figura 30: Graphical remote desktop

5.4.3 NPM

El NPM [42], facilita a los desarrolladores de JavaScript compartir y reutilizar el código, y facilita la actualización del código si este se está compartiendo. Gestiona y controla los paquetes dentro del entorno de ejecución JavaScript Node.js.

Desde la versión 0.6.3 de Node.js, NPM es instalado automáticamente con el entorno y se ejecuta desde la línea de comandos, gestionando las dependencias para una aplicación. Además, permite a los usuarios instalar aplicaciones Node.js que se encuentran en el repositorio. NPM está escrito enteramente en JavaScript y fue desarrollado por Isaac Z. Schlueter.

```
pi@lCloudy1:~/orbit-web $ node -v
v8.11.1
pi@lCloudy1:~/orbit-web $ npm -v
1.4.21
(node:28605) [DEP0022] DeprecationWarning: os.tmpDir() is deprecated. Use os.tmpdir() instead.
pi@lCloudy1:~/orbit-web $
```

```
pi@lCloudy1:~/orbit-web $ sudo npm install npm@latest -g
(node:29043) [DEP0022] DeprecationWarning: os.tmpDir() is deprecated. Use os.tmpdir() instead.
npm WARN package.json path-is-inside@1.0.2 No README data
npm WARN package.json sorted-object@2.0.1 No README data
npm WARN package.json config-chain@1.1.11 No license field.
npm WARN package.json cyclist@0.2.2 No license field.
npm WARN package.json json-schema@0.2.3 No license field.
npm WARN package.json punycode@1.4.1 punycode is also the name of a node core module.
npm WARN package.json qrcode-terminal@0.12.0 No license field.
npm WARN package.json string_decoder@1.1.1 string_decoder is also the name of a node core module.
/usr/local/bin/npm -> /usr/local/lib/node_modules/npm/bin/npm-cli.js
/usr/local/bin/npx -> /usr/local/lib/node_modules/npm/bin/npx-cli.js
npm@6.4.1 /usr/local/lib/node_modules/npm
pi@lCloudy1:~/orbit-web $
```

Figura 31: NPM & Node.js

5.5 Instalación IPFS

IPFS, cuyo detalle está incluido en el capítulo 4 de este proyecto, es el acrónimo de InterPlanetary File System. Si IPFS no se encuentra instalado en nuestro sistema se pueden realizar los siguientes pasos indicados en la guía de instalación [43] en el repositorio del proyecto IPFS [44].

Una vez instalado, se procede a la activación tal y como se indica en la figura 25.

```
pi@lCloudy1:~ $ ipfs init
initializing IPFS node at /home/pi/.ipfs
generating 2048-bit RSA keypair...done
peer identity: QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
to get started, enter:

        ipfs cat /ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv/readme

pi@lCloudy1:~ $
```

Figura 32: IPFS init

Una vez activado se puede comenzar a utilizar ejecutando la concatenación de comandos; ipfs cat y la secuencia indicada tras la realización del init.

```
pi@1Cloudy1:~ $ ipfs cat /ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv/readme
Hello and Welcome to IPFS!

If you're seeing this, you have successfully installed
IPFS and are now interfacing with the ipfs merkledag!

 -------------------------------------------------------
| Warning:                                              |
|   This is alpha software. Use at your own discretion! |
|   Much is missing or lacking polish. There are bugs.  |
|   Not yet secure. Read the security notes for more.   |
 -------------------------------------------------------

Check out some of the other files in this directory:

  ./about
  ./help
  ./quick-start     <-- usage examples
  ./readme          <-- this file
  ./security-notes
pi@1Cloudy1:~ $
```

Figura 33: IPFS cat <hash ID>. Comprobación instalación IPFS

Por otro lado, para inicializar el demonio y tener acceso, entre otros, a la interfaz de usuario vía web, se debería ejecutar “ipfs daemon”:

```
pi@1Cloudy1:~ $ ipfs daemon
Initializing daemon...
Successfully raised file descriptor limit to 2048.
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/172.17.0.1/tcp/4001
Swarm listening on /ip4/192.168.15.10/tcp/4001
Swarm listening on /ip6/::1/tcp/4001
Swarm listening on /ip6/fd51:42f8:caae:d92e::ff/tcp/4001
Swarm listening on /p2p-circuit/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/172.17.0.1/tcp/4001
Swarm announcing /ip4/192.168.15.10/tcp/4001
Swarm announcing /ip6/::1/tcp/4001
Swarm announcing /ip6/fd51:42f8:caae:d92e::ff/tcp/4001
API server listening on /ip4/127.0.0.1/tcp/5001
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```

Figura 34: IPFS Daemon. Arrancando demonio IPFS

5.6.1 Instalación IPFS-Cluster

Tal y como se define bajo el marco del proyecto, IPFS Cluster es una aplicación independiente y un cliente CLI que asigna, replica y rastrea a través de un clúster de demonios IPFS.

Esta aplicación proporciona, por un lado, un servicio de clúster: ipfs-cluster-service, que se ejecutará junto con Go-ipfs [45], y por otro lado, un servicio de cliente: ipfs-cluster-ctl, que permite interactuar fácilmente, vía CLI, con la API HTTP de los *peers.*

Este servicio nace con el objetivo de facilitar la adopción de IPFS mediante las siguientes características:

- Proporcionar soporte para implementaciones de producción de IPFS en el centro de datos
- Facilitar la conservación y replicación de datos (conjuntos de pines) en múltiples nodos
- Admite el manejo de grandes volúmenes, donde un DAG completo no cabe en un solo nodo IPFS
- Habilitar los esfuerzos de almacenamiento colaborativo para hacer copias de seguridad de los datos de interés sobre IPFS

En primer lugar, se debe tener en cuenta que el demonio de IPFS debe estar corriendo y, acto seguido, ejecutar el servicio de clúster en uno de los nodos.

```
pi@1Cloudy1:~/Downloads/ipfs-cluster-ctl $ ../ipfs-cluster-service/ipfs-cluster-service daemon
```

```
pi@1Cloudy1:~/Downloads/ipfs-cluster-ctl $ ../ipfs-cluster-service/ipfs-cluster-service daemon
16:31:54.578  INFO    service: Initializing. For verbose output run with "-l debug". Please wait... daemon.go:43
16:31:54.869  INFO    cluster: IPFS Cluster v0.7.0 listening on:
        /p2p-circuit/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
        /ip4/127.0.0.1/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
        /ip4/192.168.15.10/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
        /ip4/172.18.0.1/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
        /ip4/169.254.149.152/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
        /ip4/172.17.0.1/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
        /ip4/169.254.148.105/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M

 cluster.go:107
16:31:54.871  INFO   ipfshttp: IPFS Proxy: /ip4/127.0.0.1/tcp/9095 -> /ip4/127.0.0.1/tcp/5001 ipfshttp.go:221
16:31:54.871  INFO  consensus: initializing raft cluster with the following voters:
        QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M

 raft.go:230
16:31:54.871  INFO    restapi: REST API (HTTP): /ip4/127.0.0.1/tcp/9094 restapi.go:414
16:31:54.873  INFO    restapi: REST API (libp2p-http): ENABLED. Listening on:
        /ip4/127.0.0.1/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
        /ip4/192.168.15.10/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
        /ip4/172.18.0.1/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
        /ip4/169.254.149.152/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
        /ip4/172.17.0.1/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
        /ip4/169.254.148.105/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
        /p2p-circuit/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M

 restapi.go:431
16:31:56.890  INFO  consensus: Current Raft Leader: QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M raft.go:293
16:31:56.890  INFO    cluster: Cluster Peers (without including ourselves): cluster.go:403
16:31:56.891  INFO    cluster:     - No other peers cluster.go:405
16:31:56.891  INFO    cluster: ** IPFS Cluster is READY ** cluster.go:418
```

Figura 35: IPFS-Cluster service daemon

Previamente se debe haber inicializado el ipfs-service-cluster, mediante la sentencia “ipfs-cluster-service init”, la cual inicia el fichero y escribe la configuración inicial en el fichero “service.json”. Este apartado es importante de cara a desplegar un clúster privado, como veremos en el siguiente capítulo.

```
pi@1Cloudy1:~/Downloads/ipfs-cluster-ctl $ ../ipfs-cluster-service/ipfs-cluster-service init
The peer's state will be removed from the load path.  Existing pins may be lost.
Configuration(service.json) will be overwritten. Continue? [y/n]: y
16:31:47.816  INFO  consensus: cleaning empty Raft data folder (/home/pi/.ipfs-cluster/raft) raft.go:648
16:31:47.817 WARNI    service: the /home/pi/.ipfs-cluster/raft folder has been rotated.  Next start will use an empt
y state state.go:171
16:31:51.292  INFO     config: Saving configuration config.go:327
ipfs-cluster-service configuration written to /home/pi/.ipfs-cluster/service.json
```

Figura 36: Inicialización servicio IPFS-Cluster

Para la instalación se ha optado por los binarios proporcionados en la página de distribuciones de IPFS. Esto se ha debido a que los contenedores Docker existentes son para arquitecturas amd64 y x86, por lo tanto, no se podrían usar en el componente principal de nuestra microcloud local, basada en SBC y arquitectura ARM.

Otro de los binarios que se utiliza es el “ipfs-cluster-ctl”, interfaz CLI para gestionar el cluster y sus peers.

```
pi@1Cloudy1:~/Downloads/ipfs-cluster-ctl $ ./ipfs-cluster-ctl peers ls
QmaPYTu6r6ANjJLPvKebx4czRxRbLSt95hNCGqaurCStnV | 1Cloudy2 | Sees 1 other peers
  > Addresses:
    - /ip4/127.0.0.1/tcp/9096/ipfs/QmaPYTu6r6ANjJLPvKebx4czRxRbLSt95hNCGqaurCStnV
    - /ip4/172.17.0.1/tcp/9096/ipfs/QmaPYTu6r6ANjJLPvKebx4czRxRbLSt95hNCGqaurCStnV
    - /ip4/192.168.15.20/tcp/9096/ipfs/QmaPYTu6r6ANjJLPvKebx4czRxRbLSt95hNCGqaurCStnV
    - /p2p-circuit/ipfs/QmaPYTu6r6ANjJLPvKebx4czRxRbLSt95hNCGqaurCStnV
  > IPFS: QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip4/127.0.0.1/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip4/172.17.0.1/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip4/192.168.15.20/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip4/93.156.73.143/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip6/::1/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip6/fd51:42f8:caae:d92e::ff/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7 | 1Cloudy1 | Sees 1 other peers
  > Addresses:
    - /ip4/127.0.0.1/tcp/9096/ipfs/Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7
    - /ip4/169.254.148.105/tcp/9096/ipfs/Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7
    - /ip4/169.254.149.152/tcp/9096/ipfs/Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7
    - /ip4/172.17.0.1/tcp/9096/ipfs/Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7
    - /ip4/172.18.0.1/tcp/9096/ipfs/Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7
    - /ip4/192.168.15.10/tcp/9096/ipfs/Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7
    - /p2p-circuit/ipfs/Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7
  > IPFS: QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/127.0.0.1/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/169.254.148.105/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/169.254.149.152/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/172.17.0.1/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/172.18.0.1/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/192.168.15.10/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/93.156.73.143/tcp/61570/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip6/::1/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip6/fd51:42f8:caae:d92e::ff/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
```

Figura 37: Listado peers clúster IPFS

5.6.2 Configuración clúster privado con IPFS-Cluster

A continuación, se realiza la configuración de un clúste*r* privado IPFS. Por un lado, se dispone de un nodo principal únicamente como rol representativo, y por otro, los *peers* privados que se irán añadiendo a dicho clúste*r*. Los nodos que se añaden deben cumplir los requisitos del capítulo 5.6.1, además, disponer del mismo *SECRET* que el nodo principal.

En la siguiente referencia [46], se puede encontrar una guía de la configuración de dicho aspecto, generar la llave en el principal y exportar la misma a los diferentes nodos que forman parte del clúste*r* o incluirla en los ficheros de configuración “service.json” de los *peers*.

```
pi@1Cloudy2:~/Downloads/ipfs-cluster-service $ ./ipfs-cluster-service daemon --bootstrap /ip4/192.168.15.10/tcp/90
96/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
16:46:54.407  INFO    service: Initializing. For verbose output run with "-l debug". Please wait... daemon.go:43
16:46:54.411  INFO  consensus: cleaning empty Raft data folder (/home/pi/.ipfs-cluster/raft) raft.go:648
16:46:54.411 WARNI    service: the /home/pi/.ipfs-cluster/raft folder has been rotated.  Next start will use an em
pty state state.go:171
16:46:54.451  INFO    cluster: IPFS Cluster v0.7.0 listening on:
        /p2p-circuit/ipfs/QmSwkaqawN2jnhkYcSrfBPiEMraavwspwCSQLjiqwh8PU9
        /ip4/127.0.0.1/tcp/9096/ipfs/QmSwkaqawN2jnhkYcSrfBPiEMraavwspwCSQLjiqwh8PU9
        /ip4/192.168.15.20/tcp/9096/ipfs/QmSwkaqawN2jnhkYcSrfBPiEMraavwspwCSQLjiqwh8PU9
        /ip4/172.17.0.1/tcp/9096/ipfs/QmSwkaqawN2jnhkYcSrfBPiEMraavwspwCSQLjiqwh8PU9

 cluster.go:107
16:46:54.453  INFO    restapi: REST API (HTTP): /ip4/127.0.0.1/tcp/9094 restapi.go:414
16:46:54.453  INFO    service: Bootstrapping to /ip4/192.168.15.10/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2V
TTFSEAfrinv8M daemon.go:153
16:46:54.454  INFO   ipfshttp: IPFS Proxy: /ip4/127.0.0.1/tcp/9095 -> /ip4/127.0.0.1/tcp/5001 ipfshttp.go:221
16:46:54.454  INFO    restapi: REST API (libp2p-http): ENABLED. Listening on:
        /p2p-circuit/ipfs/QmSwkaqawN2jnhkYcSrfBPiEMraavwspwCSQLjiqwh8PU9
        /ip4/127.0.0.1/tcp/9096/ipfs/QmSwkaqawN2jnhkYcSrfBPiEMraavwspwCSQLjiqwh8PU9
        /ip4/192.168.15.20/tcp/9096/ipfs/QmSwkaqawN2jnhkYcSrfBPiEMraavwspwCSQLjiqwh8PU9
        /ip4/172.17.0.1/tcp/9096/ipfs/QmSwkaqawN2jnhkYcSrfBPiEMraavwspwCSQLjiqwh8PU9

 restapi.go:431
16:46:54.464  INFO  consensus: peer is ready to join a cluster raft.go:221
16:46:54.965  INFO  consensus: Current Raft Leader: QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M raft.go:293
16:46:54.965  INFO    cluster: Cluster Peers (without including ourselves): cluster.go:403
16:46:54.965  INFO    cluster:     - QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M cluster.go:410
16:46:54.965  INFO    cluster: ** IPFS Cluster is READY ** cluster.go:418
16:46:55.173  INFO  consensus: Current Raft Leader: QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M raft.go:293
16:46:55.174  INFO    cluster: QmSwkaqawN2jnhkYcSrfBPiEMraavwspwCSQLjiqwh8PU9: joined QmViC61LsYVNkZhtnCwQFn7LhxE5
4ea2VTTFSEAfrinv8M's cluster cluster.go:692
```

Figura 38: Añadiendo peers al clúster IPFS

Mediante la utilización del binario “ipfs-cluster-ctl” se pueden monitorizar, entre otros aspectos, los peers del cluster.

```
pi@1Cloudy1:~/.ipfs-cluster $ 08:36:02.365  INFO  consensus: peer added to Raft: QmPQ2L1zsFfCAykqZseY59S1MGNTTiAr1tdN
UxA63cZDfm consensus.go:355
08:36:02.399  INFO    cluster: Peer added  QmPQ2L1zsFfCAykqZseY59S1MGNTTiAr1tdNUxA63cZDfm cluster.go:602
08:37:27.955  INFO  consensus: peer added to Raft: QmRVMDMczHqpzW1u4neAqzCXQ5gP1kh15k24xvyNCG5TZ1 consensus.go:355
08:37:27.968  INFO    cluster: Peer added  QmRVMDMczHqpzW1u4neAqzCXQ5gP1kh15k24xvyNCG5TZ1 cluster.go:602
```

```
pi@1Cloudy2:~/Downloads/ipfs-cluster-service $ ../ipfs-cluster-ctl/ipfs-cluster-ctl peers ls
QmaPYTu6r6ANjJLPvKebx4czRxRbLSt95hNCGqaurCStnV | 1Cloudy2 | Sees 1 other peers
  > Addresses:
    - /ip4/127.0.0.1/tcp/9096/ipfs/QmaPYTu6r6ANjJLPvKebx4czRxRbLSt95hNCGqaurCStnV
    - /ip4/172.17.0.1/tcp/9096/ipfs/QmaPYTu6r6ANjJLPvKebx4czRxRbLSt95hNCGqaurCStnV
    - /ip4/192.168.15.20/tcp/9096/ipfs/QmaPYTu6r6ANjJLPvKebx4czRxRbLSt95hNCGqaurCStnV
    - /p2p-circuit/ipfs/QmaPYTu6r6ANjJLPvKebx4czRxRbLSt95hNCGqaurCStnV
  > IPFS: QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip4/127.0.0.1/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip4/172.17.0.1/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip4/192.168.15.20/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip4/93.156.73.143/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip6/::1/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip6/fd51:42f8:caae:d92e::ff/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7 | 1Cloudy1 | Sees 1 other peers
  > Addresses:
    - /ip4/127.0.0.1/tcp/9096/ipfs/Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7
    - /ip4/169.254.148.105/tcp/9096/ipfs/Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7
    - /ip4/169.254.149.152/tcp/9096/ipfs/Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7
    - /ip4/172.17.0.1/tcp/9096/ipfs/Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7
    - /ip4/172.18.0.1/tcp/9096/ipfs/Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7
    - /ip4/192.168.15.10/tcp/9096/ipfs/Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7
    - /p2p-circuit/ipfs/Qme59NpiC2E7js7xKduoDokyTW9yUTGeP7q9JMeWToFjx7
  > IPFS: QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/127.0.0.1/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/169.254.148.105/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/169.254.149.152/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/172.17.0.1/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/172.18.0.1/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/192.168.15.10/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/93.156.73.143/tcp/61570/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip6/::1/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip6/fd51:42f8:caae:d92e::ff/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
```

Figura 39: Logs de nuevo *peer* en clúster y listado peers

En la figura 33, se puede ver como se detecta la existencia de otro *peer*, en este caso, lCloudy2 (192.168.15.20) está viendo ya a lCloudy1 (192.168.15.10).

Se ha optado por la instalación manual directamente sobre el nodo debido a que, aunque IPFS-Cluster dispone de versiones oficiales bajo contenedores Docker [47], no están diseñadas para funcionar sobre arquitecturas ARM, de momento.

5.6.2 Instalación Go IPFS

Este capítulo hace referencia a la necesidad de instalar la implementación de IPFS en Go, no obstante, si el entorno dispone del demonio IPFS y de la de la herramienta para gestionar código fuente Go (Golang Go), no sería necesaria su instalación.

La herramienta Go incluye un comando, denominado go, que automatiza la descarga, creación e instalación de los paquetes y comandos de Go. Go es un lenguaje de programación Opensource concurrente y compilado inspirado en la sintaxis de C.

5.7 Despliegue servicios Docker. “Dockerización”

En este capítulo se detalla el despliegue de servicios mediante el uso de contenedores en Linux o servicios *dockerizados*. Se han seleccionado una serie de servicios para el proyecto los cuales que se detallaran en los siguientes subcapítulos.

5.7.1 Despliegue ELK (ElasticSearch)

ELK es el acrónimo de tres proyectos de código abierto:

- Elasticsearch
- Logstash
- Kibana

Elasticsearch es un motor de búsqueda y análisis, mientras que Logstash es un canal de procesamiento, en el servidor, de toda la información recibida de logs de múltiples fuentes simultáneamente, la cual enriquece y vuelve a enviar a Elasticsearch. Kibana, por otro lado, es una herramienta web encargada de

permitir que los usuarios puedan visualizar la información obtenida en Elasticsearch de manera gráfica.

En la referencia [48] se puede obtener una imagen de Docker específica para nuestro entorno sobre arquitectura ARM.

El objetivo de desplegar ELK en Cloudy es concentrar y analizar los logs producidos en el entorno, por otro lado, se aprovecha para publicarlo como servicio dockerizado en Cloudy y así comprobar cómo se publica tanto localmente como remotamente.

```
pi@lCloudy1:~ $ sudo docker pull ind3x/rpi-elasticsearch
Using default tag: latest
latest: Pulling from ind3x/rpi-elasticsearch
234876e69431: Pull complete
5a98a2f1f5d9: Pull complete
e84549343770: Pull complete
a3ed95caeb02: Pull complete
f09aa00a3317: Pull complete
2a75c5995e9a: Pull complete
c0d85e4c6bd9: Pull complete
2590683e04c4: Pull complete
ea771facb0ba: Pull complete
495c0ebe382d: Pull complete
15358af1a605: Pull complete
98c0d2835ad2: Pull complete
67b85fc38473: Pull complete
839ec05d7a1c: Pull complete
Digest: sha256:3a404f69939dc3d12fc53e3c5c1082e1f7d925d597fcccd697c5b459c977a494
Status: Downloaded newer image for ind3x/rpi-elasticsearch:latest
pi@lCloudy1:~ $ sudo docker pull ind3x/rpi-logstash
Using default tag: latest
```

Figura 40: Despliegue imágenes ELK en lCloudy1

Por otro lado, se aprovecha para utilizar Docker Compose, herramienta integrada en Cloudy destinada a definir y ejecutar aplicaciones multicontenedores, como es este caso, donde se dispone de tres contenedores.

Se genera el fichero con extensión “yml” que se puede observar en la figura 41.

```
pi@1Cloudy2:~/docker-rpi-elk $ more docker-compose.yml
elasticsearch:
  image: ind3x/rpi-elasticsearch:latest
  command: elasticsearch -Des.network.host=0.0.0.0
  ports:
    - "9200:9200"
    - "9300:9300"

logstash:
  build: logstash/
  command: logstash -f /etc/logstash/conf.d/logstash.conf
  volumes:
    - ./logstash/config:/etc/logstash/conf.d
  ports:
    - "5000:5000"
  links:
    - elasticsearch

kibana:
  build: kibana/
  volumes:
    - ./kibana/config/:/opt/kibana/config/
  ports:
    - "5601:5601"
  links:
    - elasticsearch
pi@1Cloudy2:~/docker-rpi-elk $
```

Figura 41: ELK en Cloudy. Proyecto compose

Finalmente, se puede observar a través de Cloudy los contenedores ejecutandose y recibiendo logs.

Running Docker containers:

ContainerId	Image	Command	Created	Status	Ports	Names	Actions	
b5df4bbc57fa	dockerrpielk_logstash	"/docker-entrypoint. ..."	16 minutes ago	Up 16 minutes	0.0.0.0:5000->5000/tcp	dockerrpielk_logstash_1	Publish	Stop
0834b543c6b3	dockerrpielk_kibana	"/docker-entrypoint. ..."	16 minutes ago	Up 16 minutes	0.0.0.0:5601->5601/tcp	dockerrpielk_kibana_1	Publish	Stop
a97869d533d5	ind3x/rpi-elasticsearch:latest	"/docker-entrypoint. ..."	16 minutes ago	Up 16 minutes	0.0.0.0:9200->9200/tcp, 0.0.0.0:9300->9300/tcp	dockerrpielk_elasticsearch_1	Publish	Stop

```
pi@1Cloudy2:~/docker-rpi-elk $ nc localhost 5000 < /var/log/messages
elasticsearch_1  | [2018-12-08 10:42:01,383][INFO ][cluster.metadata         ] [Rune] [logstash-2018.12.08] creating index, cause
gstash], shards [5]/[1], mappings [_default_, logs]
elasticsearch_1  | [2018-12-08 10:42:01,789][INFO ][cluster.routing.allocation] [Rune] Cluster health status changed from [RED] to
ted [[logstash-2018.12.08][4]] ...]).
elasticsearch_1  | [2018-12-08 10:42:01,844][INFO ][cluster.metadata         ] [Rune] [logstash-2018.12.08] update_mapping [logs]
```

Figura 42: ELK en Cloudy. Servicio dockerizado

Es importante de cara a que Cloudy tiene un log propio de instalación y despliegue, pero luego utiliza el '/var/log/messages' para el resto de los logs generales, y con este stack ELK se pueden buscar y analizar logs concretos.

5.7.2 Despliegue OrbitDB

Orbit-db, tal y como se ha visto en el capítulo cuatro, es una base de datos distribuida punto a punto basada en IPFS.

En este punto, se descarga el código del repositorio [45] mediante "git clone", y acto seguido, desde el directorio de Orbit-Db, se despliega una imagen a medida para nuestro entorno:

- docker build -t orbit-db -f docker/Dockerfile .

Acto seguido, se puede lanzar el contenedor virtual asociado a dicha imagen desde Cloudy o bien mediante la ejecución de

- docker run -ti --rm orbit-db

De cara a realizar una iteración contra la bbdd, a modo de ejemplo, se puede realizar la siguiente ejecución del Node.js *examples*.

- docker run -ti --rm orbit-db nmp run examples:node

```
pi@lCloudy1:~/orbit-db $ sudo docker build -t orbit-db -f docker/Dockerfile .
Sending build context to Docker daemon  24.08MBB
Step 1/9 : FROM node:carbon
carbon: Pulling from library/node
16b1b4bf6981: Pull complete
002878cce7e1: Pull complete
d1a262dbf1b2: Pull complete
c1af7e0e98d6: Pull complete
4d57e0554c51: Pull complete
6b0da14153da: Pull complete
7602976cb05c: Pull complete
7c8f72353fa4: Pull complete
Digest: sha256:6b2c3d78f4e77fb1efle3058affdd1d3ba0b319b5eaf479167672914e555e346
Status: Downloaded newer image for node:carbon
 ---> 1e66bc1fab80
Step 2/9 : ENV NPM_CONFIG_PREFIX=/home/node/.npm-global
 ---> Running in 85a3df3b3ae0
Removing intermediate container 85a3df3b3ae0
 ---> b35634b50734
```

```
pi@lCloudy2:~/orbit-db $ sudo docker images
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
orbit-db                  latest              2374cd118819        About a minute ago  1.26GB
```

Figura 43: Despliegue imagen OrbitDB en lCloudy1

```
pi@lCloudy1:~/orbit-db $ sudo docker run -ti --rm orbit-db npm run examples:node


> orbit-db@0.19.9 examples:node /home/node
> node examples/eventlog.js

Starting...
Swarm listening on /ip4/127.0.0.1/tcp/4003/ws/ipfs/QmQGsJfp2wPLH3fwcdLTkKYUL8efpt6Bttuuqg3qXUTxyB
Swarm listening on /ip4/127.0.0.1/tcp/4002/ipfs/QmQGsJfp2wPLH3fwcdLTkKYUL8efpt6Bttuuqg3qXUTxyB
Swarm listening on /ip4/172.17.0.2/tcp/4002/ipfs/QmQGsJfp2wPLH3fwcdLTkKYUL8efpt6Bttuuqg3qXUTxyB
[Latest Visitors]
--------------------
ID  | Visitor
--------------------
696 | 🐴)

[Latest Visitors]
--------------------
ID  | Visitor
--------------------
114 | 🐇)
696 | 🐴)

[Latest Visitors]
--------------------
ID  | Visitor
--------------------
549 | 🐴)
753 | 🐸)
114 | 🐇)
696 | 🐴)
```

Figura 44: Ejemplos node.js contenedor OrbitDB en lCloudy1

A continuación, se indica cómo se puede realizar el despliegue directamente desde el código del repositorio Git o utilizando la herramienta npm:

```
Desde NPM
npm install orbit-db ipfs

Desde Git
git clone https://github.com/orbitdb/orbit-db.git
cd orbit-db
npm install
```

Es importante instalar dependencias como Babel y Webpack:

```
npm install --global babel-cli
npm install --global webpack
```

Una vez esté desplegado, es interesante Orbit-db-CLI, una CLI que será de utilidad para trabajar con las bases de datos Orbit.

```
Desde NPM
npm install -g orbit-db-cli

Desde Git
git clone https://github.com/orbitdb/orbit-db-cli.git
cd orbit-db-cli/
npm install
```

Ejecutando la ayuda del CLI se pueden visualizar una serie de comandos para trabajar con OrbitDB.

```
pi@1Cloudy1:~/orbit-db-cli $  node src/bin.js orbitdb help

                 _     _ _         _ _
                | |   (_) |       | | |
      ___  _ __| |__   _| |_    __| | |__
     / _ \| '__| '_ \| | __|  / _` | '_ \
    | (_) | |  | |_) | | |_  | (_| | |_) |
     \___/|_|  |_.__/|_|\__|  \__,_|_.__/

            Peer-to-Peer Database
      https://github.com/orbitdb/orbit-db


Usage: bin.js <command> <database>

Commands:
  bin.js add <database> [<data>]            Add an entry to an eventlog or feed
                                            database. Can be only used on:
                                            eventlog|feed
  bin.js create <database> <type>           Create a new database. Type can be
                                            one of:
                                            eventlog|feed|docstore|keyvalue|coun
                                            ter                   [aliases: new]
  bin.js del <database> <key>               Delete an entry from a database.
                                            Only valid for data types of:
                                            docstore|keyvalue|feed
                                                       [aliases: delete, remove]
  bin.js demo <name>                        Runs a sequence of commands as an
                                            example              [aliases: tour]
  bin.js drop <database> yes                Remove a database locally. This
                                            doesn't remove data on other nodes
                                            that have the removed database
                                            replicated.       [aliases: destroy]
  bin.js get <database> [<search>]          Query the database.
                                                         [aliases: query, search]
  bin.js id                                 Show information about current
                                            orbit-db id
  bin.js import <file> <database> <schema>  Import a CSV file to a document
                                            database              [aliases: csv]
  bin.js inc <database> [<increment>]       Increase the value of a counter
```

Figura 45: OrbitDB help

Se ejecuta una sentencia de comprobación de funcionamiento, donde se crea una base de datos de tipo documental, se realiza el hash de un documento, se sube un documento, etc., y a continuación, se elimina la base de datos.

```
root@1Cloudy2:/home/pi/orbit-db-cli# node ./src/bin.js demo hello

                 _     _ _          _ _
                | |   (_) |        | | |
      ___  _ __| |__  _| |_   __| | |__
     / _ \| '__| '_ \| | __|  / _` | '_ \
    | (_) | |  | |_) | | |_  | (_| | |_) |
     \___/|_|  |_.__/|_|\__|  \__,_|_.__/

            Peer-to-Peer Database
      https://github.com/orbitdb/orbit-db

> node "src/bin.js" create /orbitdb/demo docstore
/orbitdb/QmXNmdcExbRFWQ1tDAuoKf2FiBSaNkf8e9oXYeVmbuoSJv/orbitdb/demo

> node "src/bin.js" put /orbitdb/QmXNmdcExbRFWQ1tDAuoKf2FiBSaNkf8e9oXYeVmbuoSJv/orbitdb/demo "{\"_id\":1,\"name\":
\"hello\"}" --indexBy name
Added document 'hello'

> node "src/bin.js" get /orbitdb/QmXNmdcExbRFWQ1tDAuoKf2FiBSaNkf8e9oXYeVmbuoSJv/orbitdb/demo "hello" --progress
Loading '/orbitdb/QmXNmdcExbRFWQ1tDAuoKf2FiBSaNkf8e9oXYeVmbuoSJv/orbitdb/demo'
Loading '/orbitdb/QmXNmdcExbRFWQ1tDAuoKf2FiBSaNkf8e9oXYeVmbuoSJv/orbitdb/demo'
  1/1 | 100.0% | 00:00:00
Searching for 'hello' from '/orbitdb/QmXNmdcExbRFWQ1tDAuoKf2FiBSaNkf8e9oXYeVmbuoSJv/orbitdb/demo'

| name | _id |
|---|---|
| hello | 1 |

Found 1 matches (1 ms)

> node "src/bin.js" drop /orbitdb/QmXNmdcExbRFWQ1tDAuoKf2FiBSaNkf8e9oXYeVmbuoSJv/orbitdb/demo yes
Dropped database '/orbitdb/QmXNmdcExbRFWQ1tDAuoKf2FiBSaNkf8e9oXYeVmbuoSJv/orbitdb/demo'

Demo finished!
```

Figura 46: OrbitDB CLI ejecución demo “hello”

5.7.3 Despliegue Orbit

Orbit, tal y como se ha visto en el capítulo cuatro, es una aplicación de chat *peer-to-peer* basada en IPFS. En este punto, como en el anterior, nos bajaremos el código del repositorio, dado que no hay una imagen Docker válida para ARM. Se puede ejecutar directamente o crear un ejecutable, por otro lado, también se podría generar una imagen Docker.

```
pi@1Cloudy1:~ $ git clone https://github.com/orbitdb/orbit-web.git
Cloning into 'orbit-web'...
remote: Enumerating objects: 362, done.
remote: Counting objects: 100% (362/362), done.
remote: Compressing objects: 100% (170/170), done.
remote: Total 924 (delta 204), reused 299 (delta 186), pack-reused 562
Receiving objects: 100% (924/924), 7.26 MiB | 1.25 MiB/s, done.
Resolving deltas: 100% (520/520), done.
```

Figura 47: Despliegue Orbit via GitHub

En un primer momento se opta por el despliegue manual, descarga del código desde GitHub y ejecución.

- npm run build

Lo cual generar una carpeta dist/ con la release compilada.

- Nnm start

```
pi@1Cloudy1:~/orbit-web $ sudo npm start

> orbit-web@0.2.0-dev start /home/pi/orbit-web
> cd dist && ../node_modules/.bin/http-server -c-1 -p 8081 & open http://localhost:8081/index.html & wait

Starting up http-server, serving ./
Available on:
  http://127.0.0.1:8081
  http://192.168.15.10:8081
  http://172.18.0.1:8081
  http://169.254.156.33:8081
  http://172.17.0.1:8081
  http://169.254.123.226:8081
  http://169.254.4.62:8081
  http://169.254.29.167:8081
Hit CTRL-C to stop the server
```

Figura 48: Inicialización OrbitWeb (Orbit chat)

Y se arranca, pudiendo probar el acceso mediante:

- http://localhost:8081/index.html

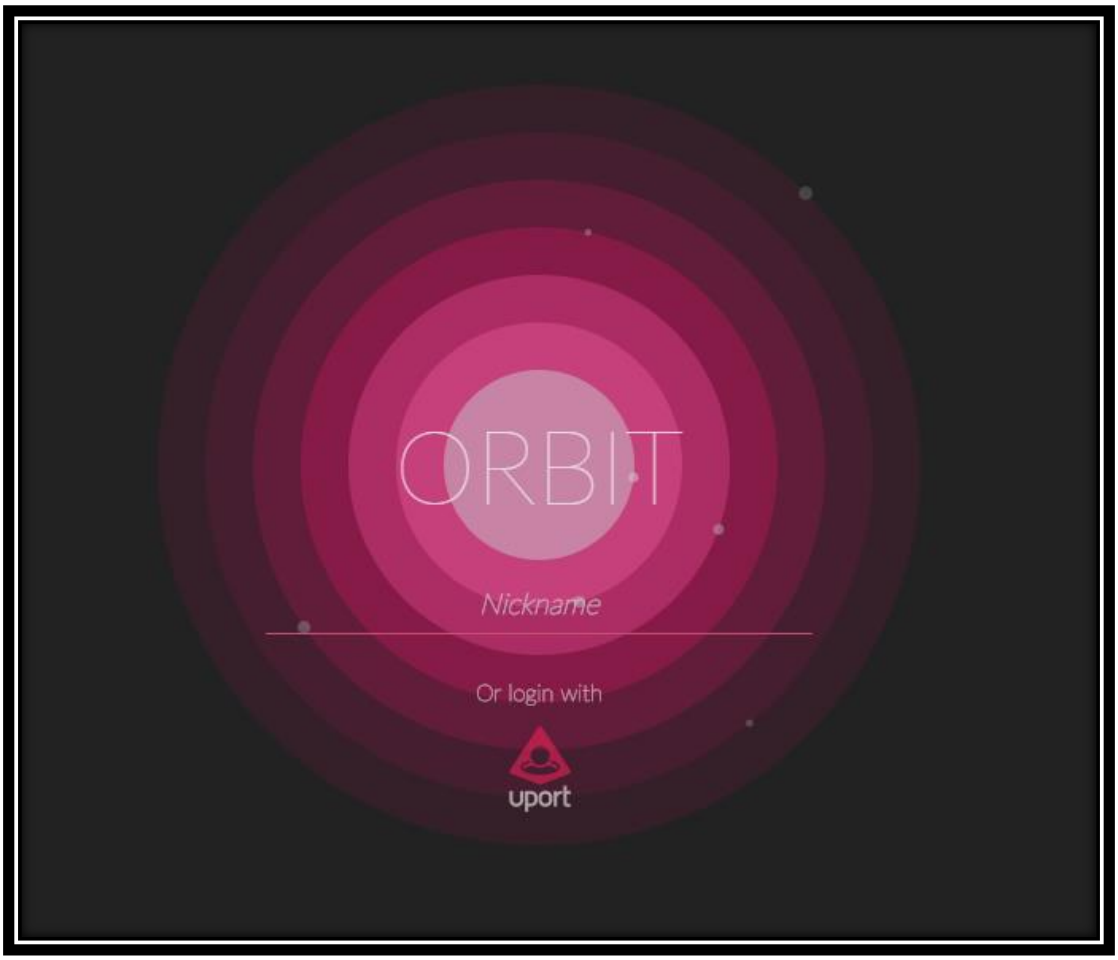


Figura 49: Ventana OrbitWeb. Chat Login

5.7.4 Dockerización IPFS-Cluster y Go-IPFS. Arquitectura ARM

Este subcapítulo muestra el despliegue de una imagen de Docker de IPFS-Cluster, no obstante, las únicas imágenes oficiales encontradas han sido para arquitecturas x86 y amd64.

```
pi@lCloudy1:~ $ sudo docker pull ipfs/ipfs-cluster:\v0.7.0
v0.7.0: Pulling from ipfs/ipfs-cluster
0de338cf4258: Pull complete
d0a0d42d89a8: Pull complete
8cb6e623e221: Pull complete
c295c5fddb9f: Pull complete
f4d90485a8d0: Pull complete
e66b803824e4: Pull complete
6814lb87cc5e: Pull complete
Digest: sha256:85eld8la553b7a3fdlb9b22375a55e96e36d508b393a83758f99ec948478915d
Status: Downloaded newer image for ipfs/ipfs-cluster:v0.7.0
```

Figura 50: Despliegue IPFS Cluster vía Docker arquitectura no ARM

Por lo tanto, si se quiere correr mediante Docker se deben construir imágenes a través del repositorio, como se han hecho en otros capítulos, por ejemplo, con OrbitDB.

Existen otras vías interesantes para desplegar, por ejemplo, Go-IPFS. Se puede utilizar ipfs y comprobar las versiones disponibles y descargarlas de dicha red.

```
root@lCloudy3:~/orbit-db# ipfs ls /ipns/dist.ipfs.io/go-ipfs/v0.4.8
QmanR5LmJraBwryvcK2YniB5xpfrMRwYwYzZDFd5P2AwDd 226     build-info
QmZftypYVNJ4zpiDJya3xLH77AkK1VmuhZfQPXS185sAkN 255     build-log-freebsd-386
QmZftypYVNJ4zpiDJya3xLH77AkK1VmuhZfQPXS185sAkN 255     build-log-freebsd-arm
QmTGyoWXiTamz6tkMeqbRmcH2ekXLaHvPXcoJ8JGwY5gcF 1426    dist.json
QmNY9Y5KZ1LUWb6WpBVcUgrCrLzzgN2BAfHXbDQwoQCyFN 8094753 go-ipfs_v0.4.8_darwin-386.tar.gz
QmfJ613Wmqa7Dv5Z2XaJvJrpSSariyewguurYwujAMUqfS 8511799 go-ipfs_v0.4.8_darwin-amd64.tar.gz
QmSmoi6SzhaikLL6J5jCwammwdXxJKxigGo6C3UkRWPz4T 8524320 go-ipfs_v0.4.8_freebsd-amd64.tar.gz
Qmdbjre2bXDLdTezPWQMwFcAW3Q1FPAse9mog8s2C1qYkF 8113455 go-ipfs_v0.4.8_linux-386.tar.gz
QmQApH3rKYanqy8zTJ7jiRBQ7aZTW6afeupe9iTZWa6oV9 8558220 go-ipfs_v0.4.8_linux-amd64.tar.gz
QmS1Z3nQDPZH9SQsVxLdzE5dCNq5WRreceXKdV8nKRjqUH 8093662 go-ipfs_v0.4.8_linux-arm.tar.gz
QmeNZzHSpwm3cnN8maogpvfHyJQkoHCchWbhZyxoJaq8xx 7865770 go-ipfs_v0.4.8_windows-386.zip
QmYXrQGCfnsZTmeeVsLZpiAtNqV1L51Xi7DVLZMY1m8w3i 8279998 go-ipfs_v0.4.8_windows-amd64.zip
QmejiRdKDajwKJvRJDpBYwKVEeAMhrdRSUMtNGrjZUQhzm 395     results
root@lCloudy3:~/orbit-db# ipfs get /ipns/dist.ipfs.io/go-ipfs/v0.4.18/go-ipfs_v0.4.18_linux-386.tar.gz
Saving file(s) to go-ipfs_v0.4.18_linux-386.tar.gz
 2.00 MB / 10.62 MB [=====================>-------------------------------------------------------
```

Figura 51: Despliegue Go IPFS utilizando IPFS. Listado versiones y descarga e instalación

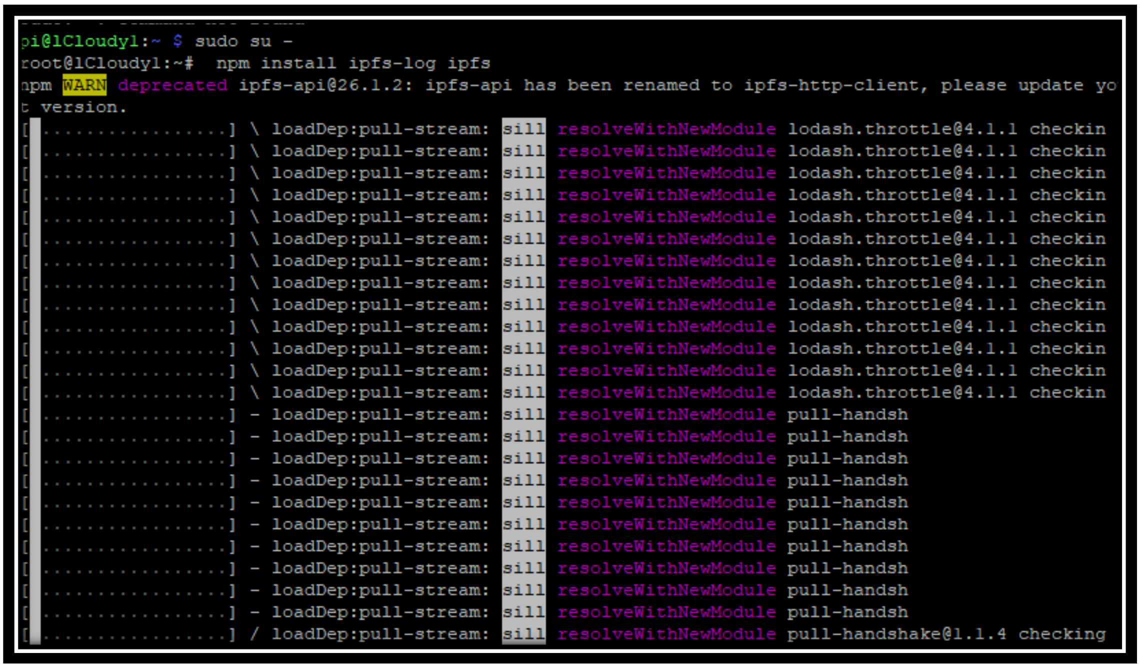
5.8 Instalación IPFS-Log

Tal y como se explica en el subcapítulo 4.4 dedicado a IPFS-Log, es una estructura de datos replicada (CRDT) inmutable enfocada al uso en sistemas distribuidos. Se requiere Node.js v8.0.0 o superior, en el caso del entorno se dispone de la versión v8.11.1.

```
pi@lCloudy1:~ $ node -v
v8.11.1
pi@lCloudy2:~ $ node -v
v8.11.1
leopoldo@lCloudy3:~$ node -v
v4.2.6 → Se debe actualizar
root@lCloudy3:~# node -v
v11.4.0
```

A continuación, se ejecuta:

- npm install ipfs-log



```
pi@lCloudy1:~ $ sudo su -
root@lCloudy1:~#  npm install ipfs-log ipfs
npm WARN deprecated ipfs-api@26.1.2: ipfs-api has been renamed to ipfs-http-client, please update yo
t version.
[█.................] \ loadDep:pull-stream: sill resolveWithNewModule lodash.throttle@4.1.1 checkin
[█.................] \ loadDep:pull-stream: sill resolveWithNewModule lodash.throttle@4.1.1 checkin
[█.................] \ loadDep:pull-stream: sill resolveWithNewModule lodash.throttle@4.1.1 checkin
[█.................] \ loadDep:pull-stream: sill resolveWithNewModule lodash.throttle@4.1.1 checkin
[█.................] \ loadDep:pull-stream: sill resolveWithNewModule lodash.throttle@4.1.1 checkin
[█.................] \ loadDep:pull-stream: sill resolveWithNewModule lodash.throttle@4.1.1 checkin
[█.................] \ loadDep:pull-stream: sill resolveWithNewModule lodash.throttle@4.1.1 checkin
[█.................] \ loadDep:pull-stream: sill resolveWithNewModule lodash.throttle@4.1.1 checkin
[█.................] \ loadDep:pull-stream: sill resolveWithNewModule lodash.throttle@4.1.1 checkin
[█.................] \ loadDep:pull-stream: sill resolveWithNewModule lodash.throttle@4.1.1 checkin
[█.................] \ loadDep:pull-stream: sill resolveWithNewModule lodash.throttle@4.1.1 checkin
[█.................] \ loadDep:pull-stream: sill resolveWithNewModule lodash.throttle@4.1.1 checkin
[█.................] \ loadDep:pull-stream: sill resolveWithNewModule lodash.throttle@4.1.1 checkin
[█.................] - loadDep:pull-stream: sill resolveWithNewModule pull-handsh
[█.................] - loadDep:pull-stream: sill resolveWithNewModule pull-handsh
[█.................] - loadDep:pull-stream: sill resolveWithNewModule pull-handsh
[█.................] - loadDep:pull-stream: sill resolveWithNewModule pull-handsh
[█.................] - loadDep:pull-stream: sill resolveWithNewModule pull-handsh
[█.................] - loadDep:pull-stream: sill resolveWithNewModule pull-handsh
[█.................] - loadDep:pull-stream: sill resolveWithNewModule pull-handsh
[█.................] - loadDep:pull-stream: sill resolveWithNewModule pull-handsh
[█.................] - loadDep:pull-stream: sill resolveWithNewModule pull-handsh
[█.................] - loadDep:pull-stream: sill resolveWithNewModule pull-handsh
[█.................] / loadDep:pull-stream: sill resolveWithNewModule pull-handshake@1.1.4 checking
```

Figura 52: Instalación IPFS-Log

5.9 Instalación Serf

Tal y como se ha visto, Cloudy utiliza Avahi para publicar los servicios en local, mientras que para publicar los servicios en otros nodos de la red comunitaria utiliza Serf. Este software habilita un puerto TCP en el nodo Cloudy, que escucha peticiones de otros nodos y así obtiene el poder comunicarse.

Serf es instalable desde Cloudy, por otro lado, se ha intentado configurar este software también por la interfaz web, no obstante, se utiliza la siguiente guía [50] de guifi.net vía CLI.

Esto no quiere decir que el interfaz web no funcione correctamente, pero al utilizar la guía se conoce el detalle de los ficheros de configuración que utiliza el demonio de Serf, y como impacta en la publicación de servicios.

Se detectan errores como es la ocupación del puerto 5000 por Kibana o el funcionamiento anómalo con IPFS en lo que a la publicación de servicios se refiere, es decir, cuando los dos servicios están levantados, se detecta que la utilidad de búsqueda de servicios publicados por IPFS prevalece sobre Serf.

Respecto al primer punto, hay que tener en cuenta que ningún otro servicio puede ocupar el puerto por defecto o el que asignemos para realizar el *bind*, ya que esto hará que el servicio no arranque.

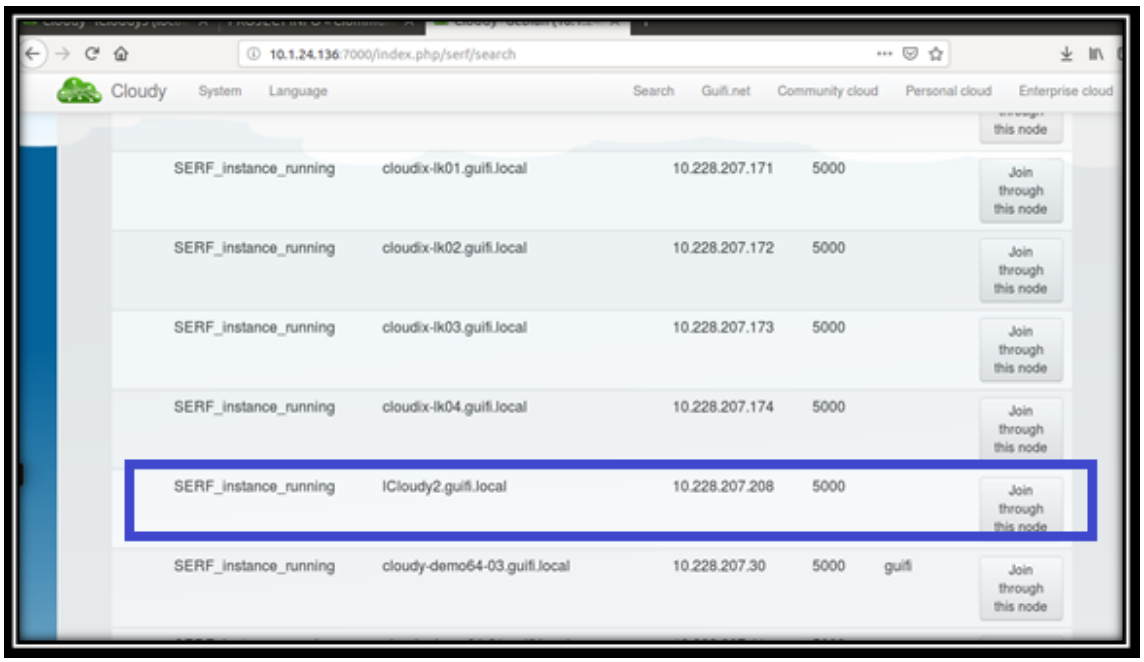


Figura 53: Cloudy y Serf

Se pueden comprobar, a continuación, los eventos de tipo INFO procedentes de Serf, y como se incluye el nuevo nodo con su instancia de Serf, mostrado en la figura anterior.

```
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: ffefe68654c68da79281e319e1ae79c8-984569744-debian-10.139.40.53:5000 10.139.40.53
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: 359908d726aaa80f9fd0526dbaaaa817-728791394-cloudix-testbed-01-10.1.24.71:5000 10.1.24.71
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: 1204f8d5fc2e3f0fab293235c8cb478f-213626083-AD1012712d-10.1.27.73:5000 10.1.27.73
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: e21d54f7c23a1b0688892ca58bcd9b5f-904392047-debian-10.1.24.158:5000 10.1.24.158
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: aafe36e1be29ba43b92f77bf16c9bde6-472123490-cloudix-lk01-10.228.207.171:5000 10.228.207.171
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: 0d38349ab912214c4ba558e326cb6d21-014524977-cloudygarlanet7-10.1.27.57:5000 10.1.27.57
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: 49859092d334d0b6a507eceb3da97a29-346832542-cloudygarlanet4-10.1.27.54:5000 10.1.27.54
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: 6f74bd449caa7f995b856a87a67cbc6d-753201290-AD1012711b-10.1.27.62:5000 10.1.27.62
2018/12/09 12:57:26 [INFO] agent: joined: 1 nodes
2018/12/09 12:57:27 [INFO] agent: Received event: member-join
```

6. Evaluación

En este capítulo se realiza una evaluación, extremo a extremo, del entorno desplegado. Se ejecutan pruebas principalmente sobre el *software* del entorno; sistema, servicios, y en especial, IPFS. Por otro lado, estas pruebas tienen impacto sobre el *hardware* utilizado; máquinas virtuales, Raspberry PI, etc.

La evaluación se distribuye en el siguiente esquema:

- Objetivo
- Fuentes de variación (si aplica)
- Desarrollo
- Resultado

Además, se presentan subcapítulos en base a algunas de las características valorables o cuantificables de los sistemas distribuidos; disponibilidad, escalabilidad, y seguridad.

Se dan por ejecutadas previamente pruebas de operación tipo acceso a Cloudy por interfaz web, instalación y prueba de software de terceros tipo OpenVPN, NPM, etc.

6.1 MicroCloud

Una vez desplegado el entorno de microCloud basado en diferentes nodos Cloudy, locales y remotos, se realizan una serie de pruebas como pueden ser:

- Arranque y parada de contenedores Docker
- Publicación de servicios vía contenedor Docker y Cloudy
- Visualización de servicios publicados vía Serf en diferentes nodos Cloudy
- Visualización de servicios publicados vía IPFS en diferentes nodos Cloudy

Este subcapítulo contiene diferentes *tests* correspondientes a la utilización de Cloudy, activación de servicios integrados en la plataforma, y despliegue de los mismos interactuando entre nodos.

6.1.1 Cloudy y Docker

Test 1

Objetivo: Activar contenedores Docker desde Cloudy.

Fuentes de variación: Acceso Cloudy, permisos Docker.

Desarrollo: Se accede al interfaz de Docker desde el portal de Cloudy, dentro del menú de Enterprise Cloud. En este interfaz del portal se debe acceder a revisar para revisar el estado de las imágenes de Docker implementadas o desplegadas anteriormente, además de poder arrancar y parar las mismas como contenedores virtuales.

Resultado: Tras haber realizado en el despliegue la publicación de varias imágenes, se accede y se arrancan varios contenedores virtuales. En el caso concreto de este test se arranca el stack de ELK en lCloudy2 y OrbitDB en lCloudy1.

Docker status:

Docker is running

Running Docker containers:

ContainerId	Image	Command	Created	Status	Ports	Names	Actions	
281783fc8e75	dockerrpielk_logstash	"/docker-entrypoint. ..."	5 days ago	Up 8 minutes	0.0.0.0:5500->5500/tcp	dockerrpielk_logstash_1	Publish	Stop
7750f4c38087	ind3x/rpi-elasticsearch:latest	"/docker-entrypoint. ..."	5 days ago	Up 8 minutes	0.0.0.0:9200->9200/tcp, 0.0.0.0:9300->9300/tcp	dockerrpielk_elasticsearch_1	Publish	Stop
05bceeab8332	dockerrpielk_kibana:latest	"/docker-entrypoint. ..."	5 days ago	Up 8 minutes	0.0.0.0:32768->5601/tcp	kibana	Unpublish	Stop

Docker status:

Docker is running

Running Docker containers:

ContainerId	Image	Command	Created	Status	Ports	Names	Actions	
a70a5bda16ae	orbit-db	"node"	28 hours ago	Up 28 hours		eager_elion	Publish	Stop

Figura 54: ICloudy2 y ICloudy1. Activación servicios vía contenedores virtuales

Se puede comprobar como las imágenes se arrancan y se puede acceder al servicio, por ejemplo, vía *webGUI*. Por otro lado, se observa el estado del sistema desde uno de los servicios activados como es Kibana, se gestionan logs, etc.

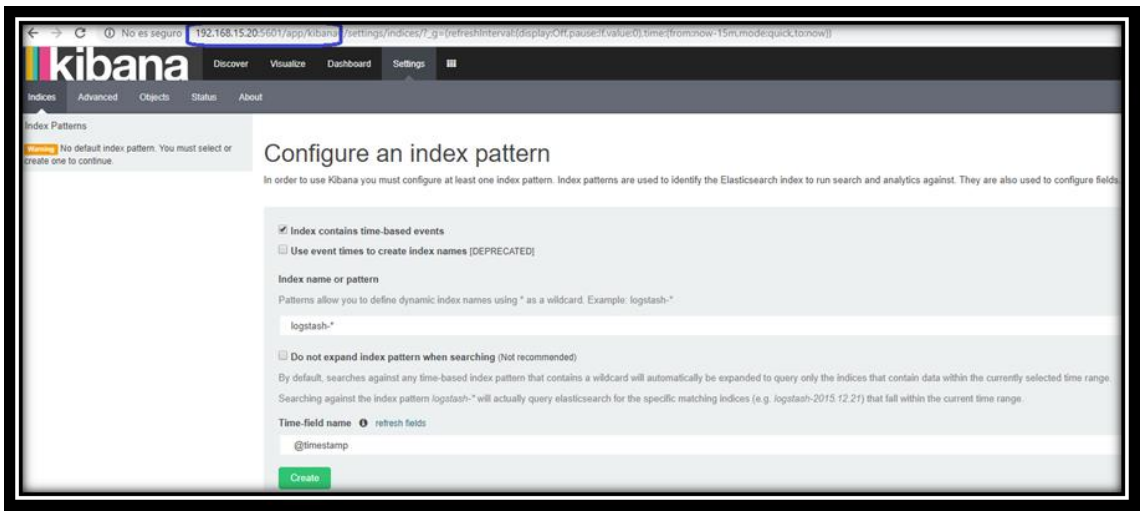


Figura 55: Kibana en ICloudy2 desde cliente remoto

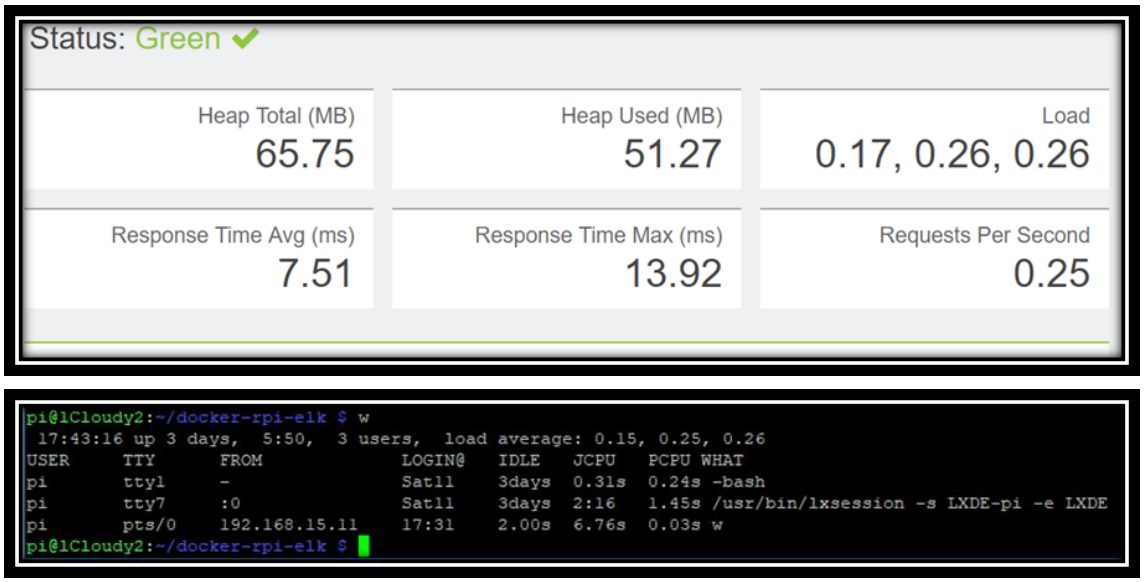


Figura 56: Kibana en lCloudy2 desde cliente remoto. Estado servidor

6.1.2 Activación Serf / IPFS

Pendiente probar la publicación de servicios con ambos activados, etc. Si en otros nodos están ambos activados, uno, etc.

Test 2

Objetivo: Activación Serf en Cloudy.

Fuentes de variación: Permisos usuario Cloudy.

Desarrollo: Se realiza la activación de Serf desde el portal de Cloudy. El proceso de arranque y publicación de servicios vía Serf debe realizarse de manera satisfactoria independientemente del usuario utilizado.

Resultado: Se activa correctamente el servicio de Serf desde Cloudy. Existen unos parámetros como puerto de escucha e IP utilizada como *bootstrap* que son claves en la publicación de servicios.

A continuación, se anexa figura con el detalle de la activación desde el portal. Por otro lado, se realiza la prueba de arrancar el demonio de Serf desde CLI, Cloudy muestra correctamente desde el portal el demonio corriendo.

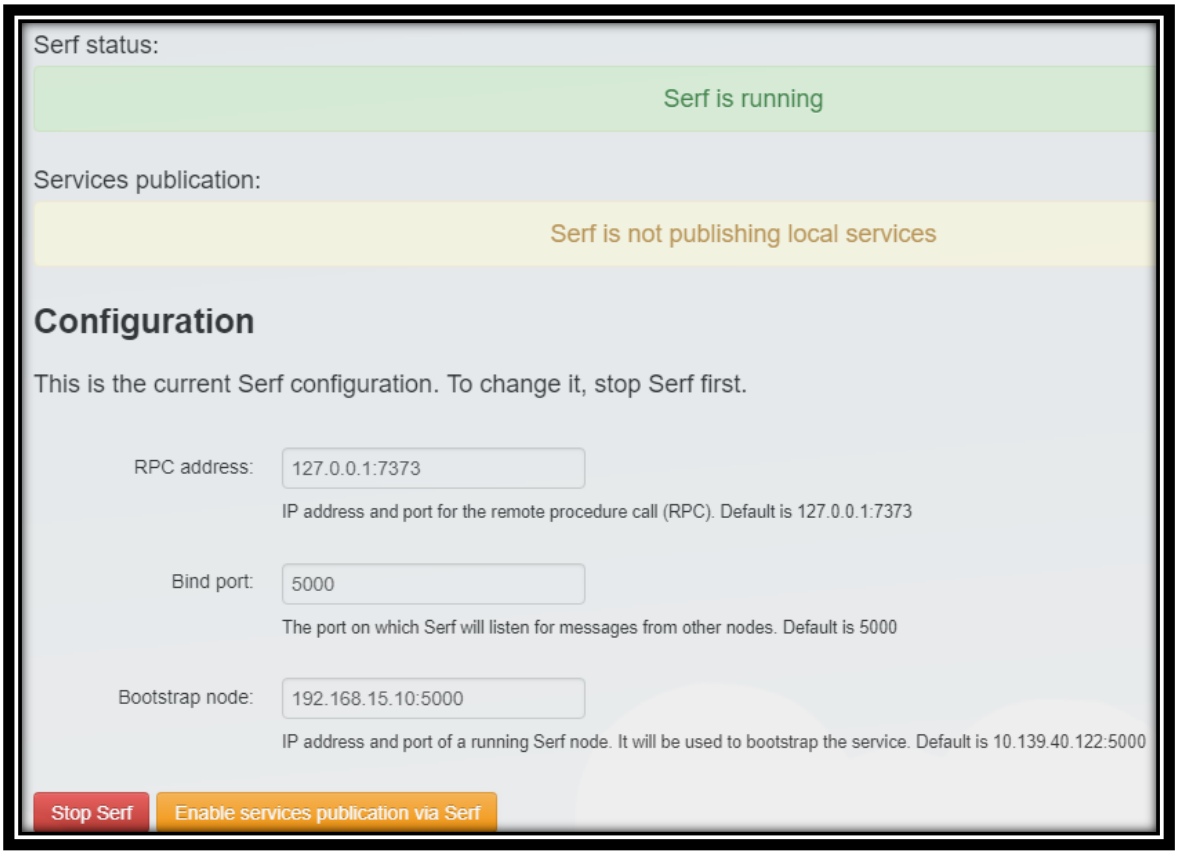


Figura 57: Serf is running

Se puede observar cómo se activa el demonio y se ven las diferentes instancias Serf de los nodos de la microCloud regional conectada. En este ejemplo se ha utilizado un *bootstrap* conocido perteneciente a la comunidad Guifi.net

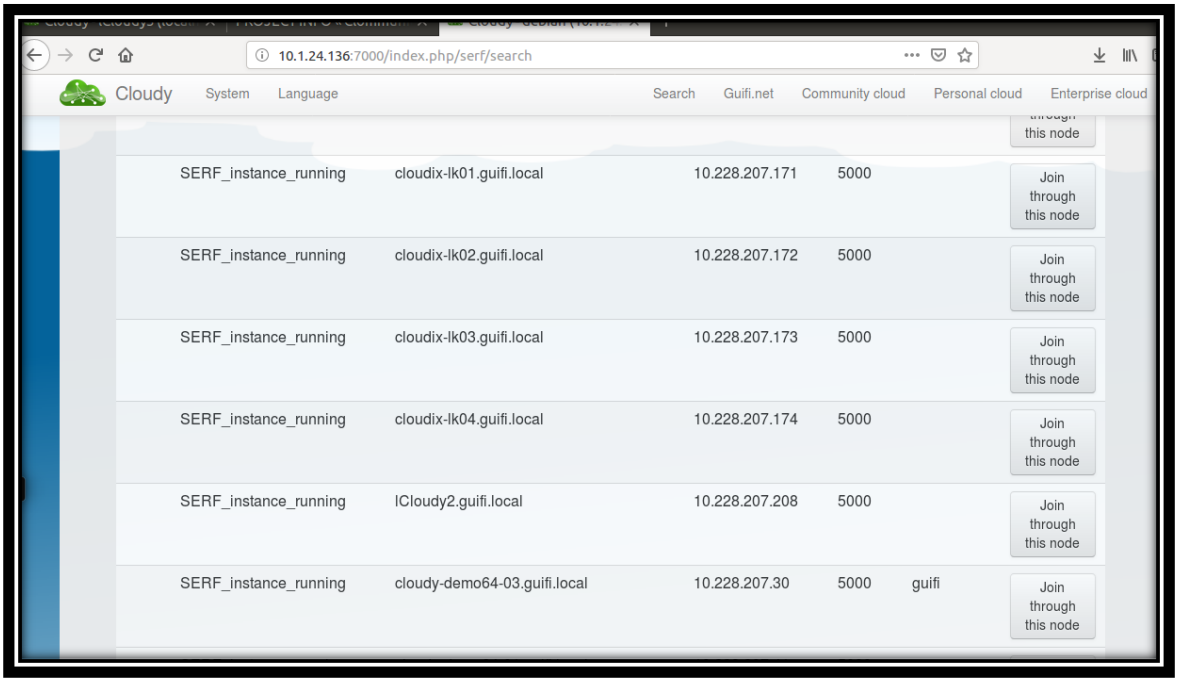


Figura 58: Instancia Serf running, Entre otras lCloudy2

En línea de comandos se puede revisar el siguiente log:

```
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: ffefe68654c68da79281e319e1ae79c8-984569744-debian-10.139.40.53:5000 10.139.40.53
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: 359908d726aaa80f9fd0526dbaaaa817-728791394-cloudix-testbed-01-10.1.24.71:5000 10.1.24.71
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: 1204f8d5fc2e3f0fab293235c8cb478f-213626083-AD1012712d-10.1.27.73:5000 10.1.27.73
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: e21d54f7c23a1b0688892ca58bcd9b5f-904392047-debian-10.1.24.158:5000 10.1.24.158
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: aafe36e1be29ba43b92f77bf16c9bde6-472123490-cloudix-lk01-10.228.207.171:5000 10.228.207.171
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: 0d38349ab912214c4ba558e326cb6d21-014524977-cloudygarlanet7-10.1.27.57:5000 10.1.27.57
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: 49859092d334d0b6a507eceb3da97a29-346832542-cloudygarlanet4-10.1.27.54:5000 10.1.27.54
2018/12/09 12:57:26 [INFO] serf: EventMemberJoin: 6f74bd449caa7f995b856a87a67cbc6d-753201290-AD1012711b-10.1.27.62:5000 10.1.27.62
2018/12/09 12:57:26 [INFO] agent: joined: 1 nodes
2018/12/09 12:57:27 [INFO] agent: Received event: member-join
```

Test 3

Objetivo: Activación IPFS en Cloudy.

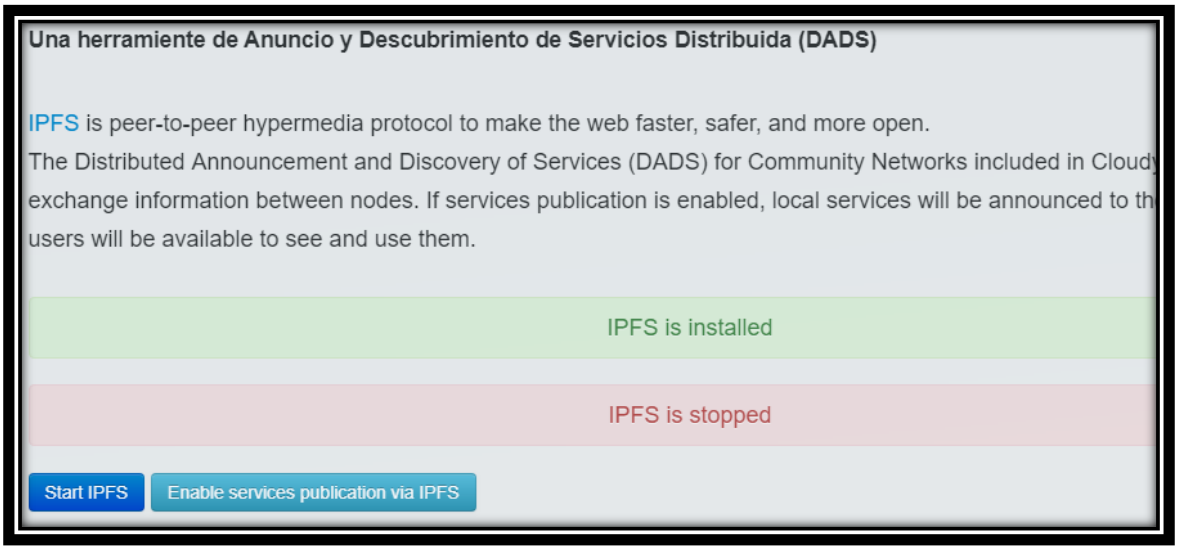
Fuentes de variación: Permisos usuario Cloudy.

Desarrollo: Se realiza la activación de IPFS desde el portal de Cloudy. El proceso de arranque y publicación de servicios vía IPFS se debe realizar de manera satisfactoria e independiente del usuario utilizado. Si es un usuario válido de sistemas y, por tanto, tiene acceso a Cloudy, debe poder activar estos servicios integrados.

Resultado: Se activa correctamente el servicio de IPFS desde Cloudy. Este servicio puede ser también arrancado desde línea de comandos, no obstante, se ha detectado en algún nodo que sí existe previamente una instancia corriendo de IPFS, Cloudy muestra IPFS instalado, pero no corriendo.

En las figuras siguientes se muestra el demonio corriendo el lCloudy1, y como se refleja detenido desde Cloudy. El botón de arranque no muestra ningún efecto.

No obstante, si se para el servicio desde línea de comandos se arranca desde Cloudy sin problema.



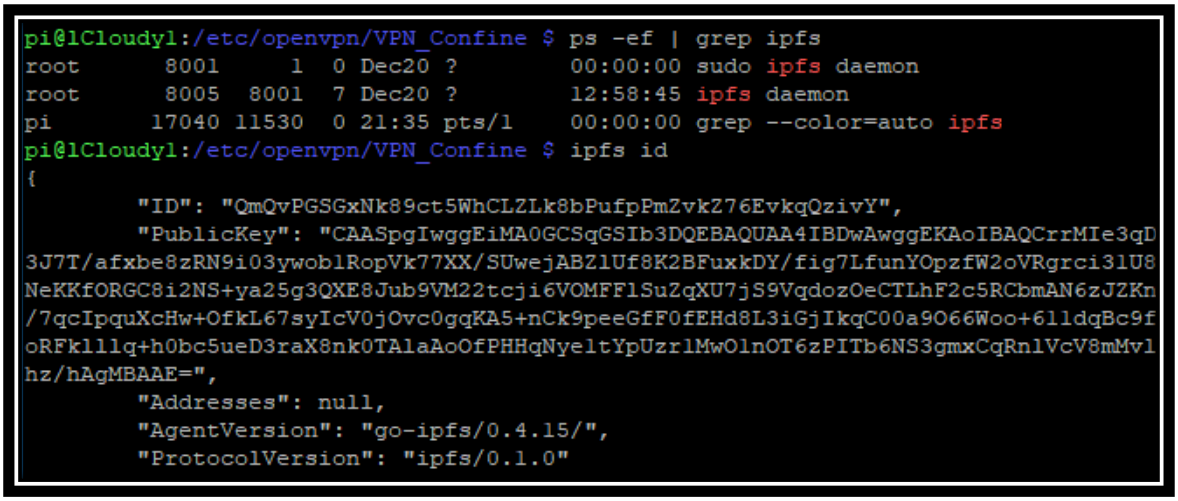


Figura 59: IPFS installed and running in lCloudy1

6.1.3 Publicación y visualización de servicios

En este subcapítulo se realiza la publicación de varios servicios *dockerizados* dentro de Cloudy, en diferentes nodos de la microCloud. El objetivo, después de familiarizarse con el entorno en la instalación y despliegue, es el de validar el funcionamiento de las publicaciones, visualización y el protocolo *gossip* de Serf para descubrimiento de otras instancias.

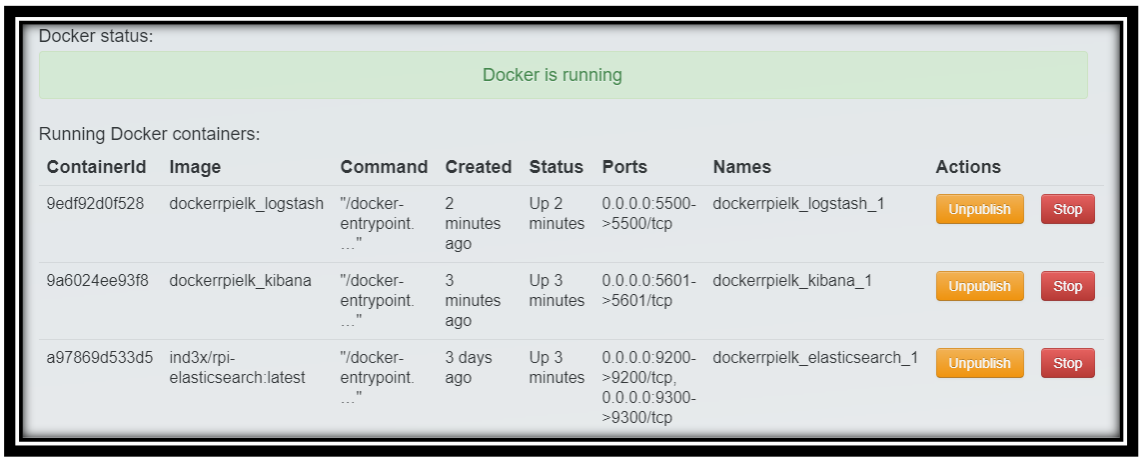
Test 4

Objetivo: Publicar servicio ELK en Docker vía Cloudy. lCloudy2.

Fuentes de variación: No aplica.

Desarrollo: Se realiza la publicación del servicio en Cloudy, y posteriormente, su visualización en otros nodos de la microCloud.

Resultado: Una vez se han publicado los servicios, tras su activación desde el área de Enterprise Cloud, se revisa que estos se ven desde el propio nodo y otros nodos de la microCloud. En la figura siguiente se puede ver el stack de ELK ya publicado. El botón a la izquierda del *stop* es *publish/unpublish.*


Docker status:

Docker is running

Running Docker containers:

ContainerId	Image	Command	Created	Status	Ports	Names	Actions	
9edf92d0f528	dockerrpielk_logstash	"/docker-entrypoint. …"	2 minutes ago	Up 2 minutes	0.0.0.0:5500->5500/tcp	dockerrpielk_logstash_1	Unpublish	Stop
9a6024ee93f8	dockerrpielk_kibana	"/docker-entrypoint. …"	3 minutes ago	Up 3 minutes	0.0.0.0:5601->5601/tcp	dockerrpielk_kibana_1	Unpublish	Stop
a97869d533d5	ind3x/rpi-elasticsearch:latest	"/docker-entrypoint. …"	3 days ago	Up 3 minutes	0.0.0.0:9200->9200/tcp, 0.0.0.0:9300->9300/tcp	dockerrpielk_elasticsearch_1	Unpublish	Stop

Figura 60: lCloudy2. Publicación servicio. Acción *publish*

Test 5

Objetivo: Publicar servicio MongoDB en Docker vía Cloudy: lCloudy2.

Fuentes de variación: No aplica.

Desarrollo: Se realiza la publicación del servicio en Cloudy.

Resultado: Se publican varios servicios, en este Test MongoDB en lCloudy1.Una vez se han publicado los servicios, tras su activación desde el área de Enterprise Cloud, se revisa que estos se ven desde el propio nodo y otros nodos de la microCloud. En la figura siguiente se puede ver el componente de ELK; elasticsearch y el servicio de referencia del test; MongoDB ya publicados. El botón a la izquierda del *stop* es *publish/unpublish.*

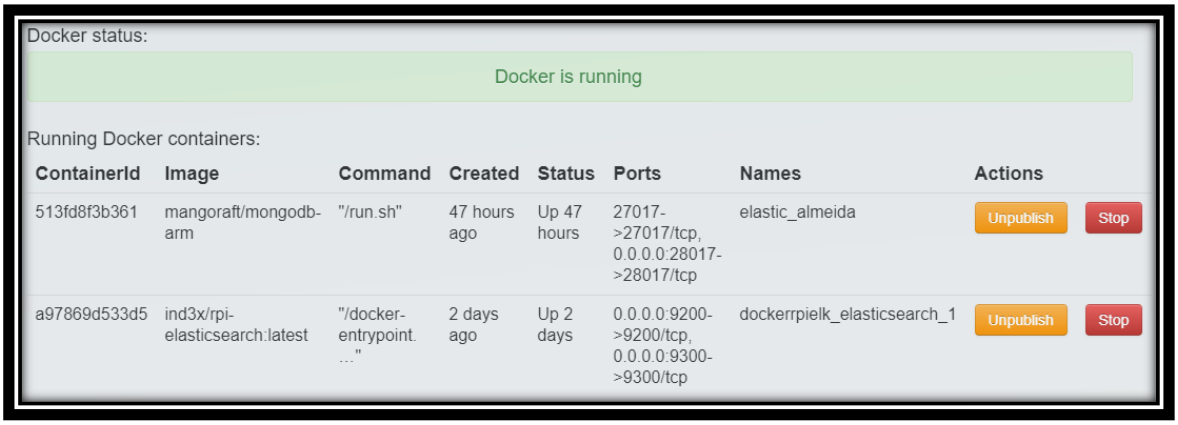


Figura 61: lCloudy2. Publicación servicio MongoDB. Acción *publish*

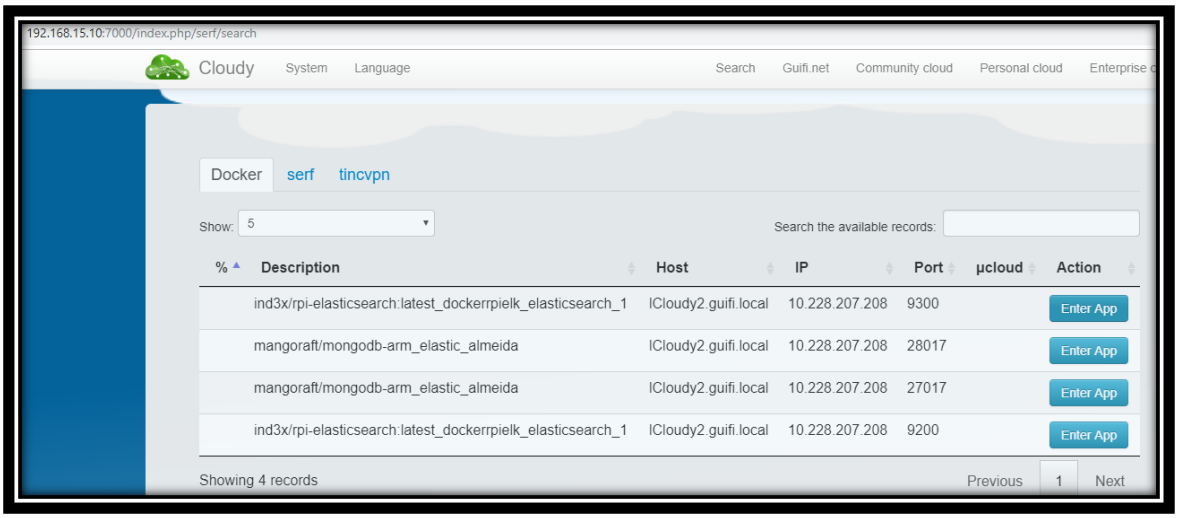


Figura 62: Visualización de servicios de Cloudy2 desde lCloudy1

Test 6

Objetivo: Publicar servicio OrbitDB en Docker vía Cloudy: lCloudy1.

Fuentes de variación: No aplica.

Desarrollo: Se realiza la publicación del servicio en Cloudy.

Resultado: La activación de otro servicio dockerizado, en este caso OrbitDB, se realiza con éxito, no obstante, se obtiene el siguiente resultado que imposibilita su publicación desde lCloudy1.

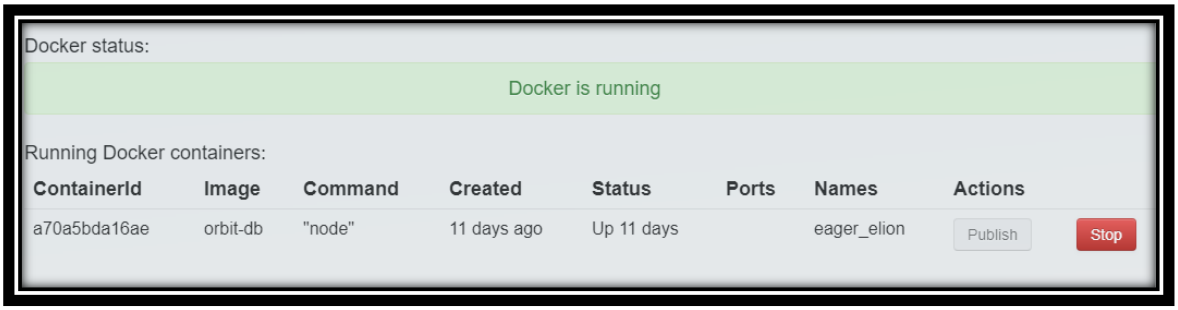


Figura 63: ICloudy1. Activación OrbitDB. Botón “publish” desactivado

El botón de publicación está inhabilitado y no se encuentra información relevante en los logs que permita encontrar el posible fallo o explicación.

Test 7

Objetivo: Publicar servicio OwnCloud en Docker vía Cloudy: rCloudy3.

Fuentes de variación: No aplica.

Desarrollo: Se realiza la publicación del servicio en Cloudy.

Resultado: La activación de otro servicio *dockerizado*, en este caso OwnCloud, se realiza con éxito, no obstante, se obtiene el resultado que se muestra en la figura 65, mostrando el siguiente error al intentar publicar en Cloudy:

/tmp/cDistrod is not file type FIFO, maybe daemon is not running

Este resultado imposibilita su publicación desde rCloudy3. Se realiza la misma prueba desde rCloudy2, debido al mismo sistema operativo y versionado obteniendo el mismo resultado.

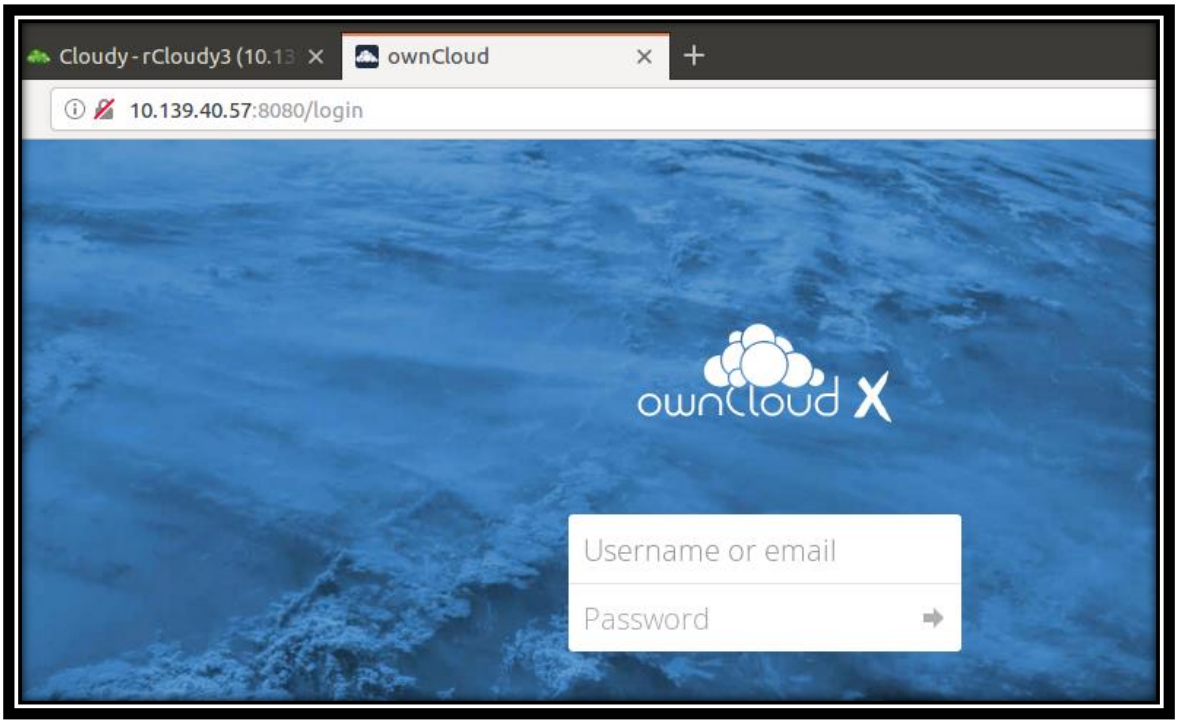


Figura 64: rCloudy3. Activación servicio OwnCloud. Acción *publish*

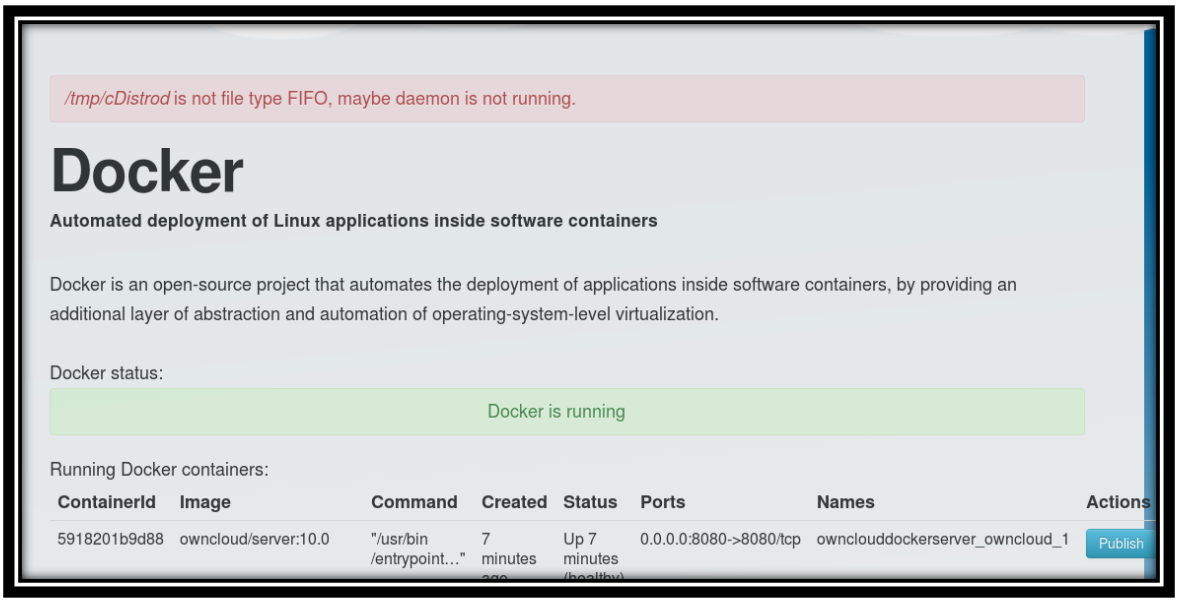


Figura 65: rCloudy3. Publicación servicio. Acción *publish*. Error

Test 8

Objetivo: Publicación vía Serf/IPFS. Instancias nodos microCloud.

Fuentes de variación: No aplica.

Desarrollo: Se realizan las activaciones de Serf en varios nodos de la microCloud regional (nodos locales más nodos remotos). Se debe revisar la publicación y comunicación entre los diferentes nodos.

Resultado: Tras la activación de diferentes demonios Serf en los nodos Cloudy, y modificar las configuraciones, se consigue que se vean nodos locales y remotos. Por un lado, se ven las instancias Serf, y por otro, los servicios publicados en cada nodo.

SERF_instance_running	debian.guifi.local		5000	guifi	Join through this node
SERF_instance_running	debian40199.guifi.local	10.139.40.55	5000	guifi	Join through this node
SERF_instance_running	lCloudy2.guifi.local	10.228.207.208	5000		Join through this node
SERF_instance_running	rCloudy3.guifi.local	10.139.40.57	5000		Join through this node
SERF_instance_running	SmartHome1.guifi.local	10.1.24.154	5000		Join through this node
SERF_instance_running	SmartHome2.guifi.local	10.1.24.145	5000		Join through this node
SERF_instance_running	SmartHome3.guifi.local	10.1.24.150	5000		Join through this node

Figura 66: rCloudy3. Visualización instancias Serf

No obstante, se encuentran, tal y como se ha indicado en *test* previos, problemas con la publicación de servicios en diferentes nodos; botón *publish*.

```
function execute_program_detached($c)
{
    $fdpipe="/tmp/cDistrod";

    $s = stat($fdpipe);
    $mode = $s['mode'];
    if (S_IFIFO != ($mode & S_IFMT)) {
        setFlash('<i>'.$fdpipe.'</i> '.t('is not file type FIFO, maybe daemon is not running.'), "error");
    } else {
        $fh = fopen($fdpipe, "a");
        if ($fh==false) {
            setFlash(t("Some problemes in open file!"));
        }

        fprintf($fh, "%s\n", $c);
        fclose($fh);
    }
}
```

Figura 67: fragmento código cDistro de la salida mostrada al ejecutar “publish”

Por lo tanto, se prosiguen las pruebas viendo los servicios publicados en lCloudy2, desde rCloudy3, rCloudy2, lCloudy1, y la propia lCloudy2

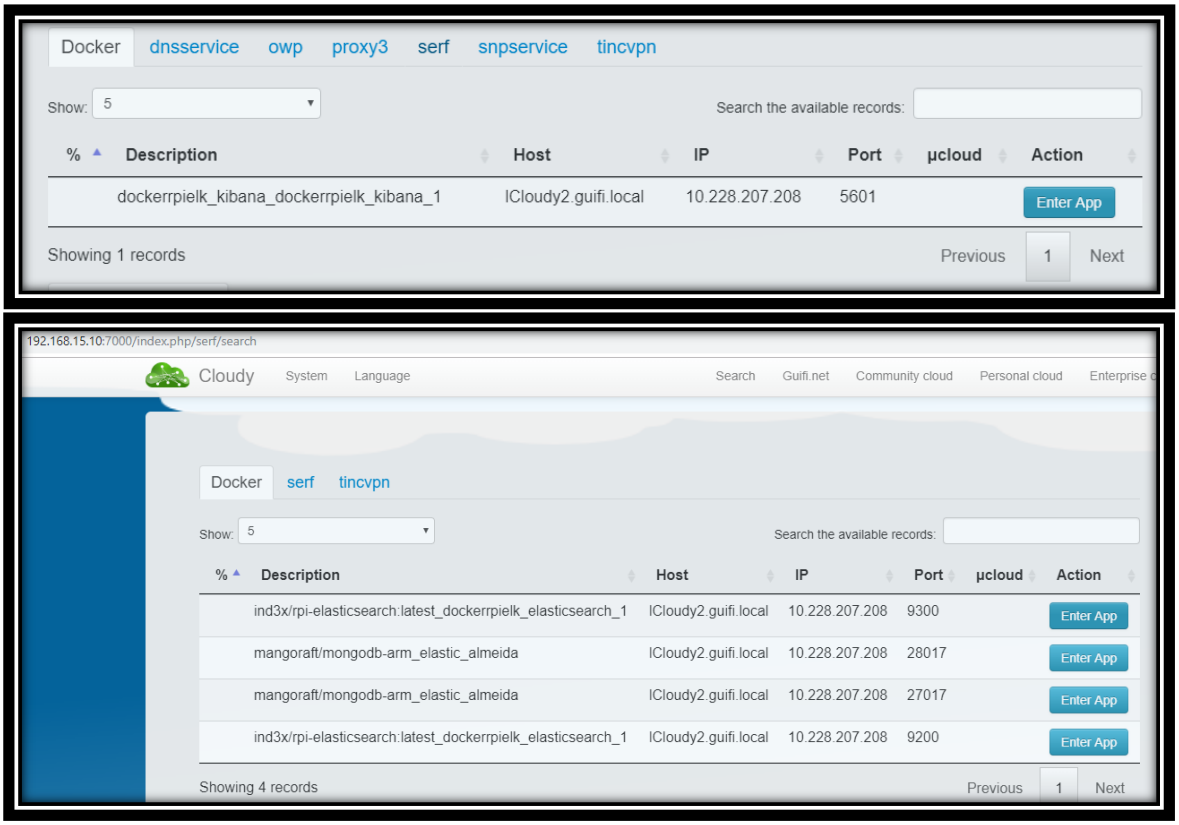


Figura 68: ICloudy2. Publicación servicio. Acción *publish*

Test 9

Objetivo: Publicación vía Serf/IPFS. Instancias nodos microCloud.

Fuentes de variación: No aplica.

Desarrollo: Se realiza la publicación de algún servicio en algún nodo específico de Cloudy. Se activa tanto Serf como IPFS, y posteriormente, se visualiza el resultado en los nodos de la microCloud.

Resultado: Al activar la publicación de servicios vía IPFS y mantener la activa la publicación en SERF, se obtiene el resultado de que los servicios aparecen publicados en IPFS pero no en Serf.

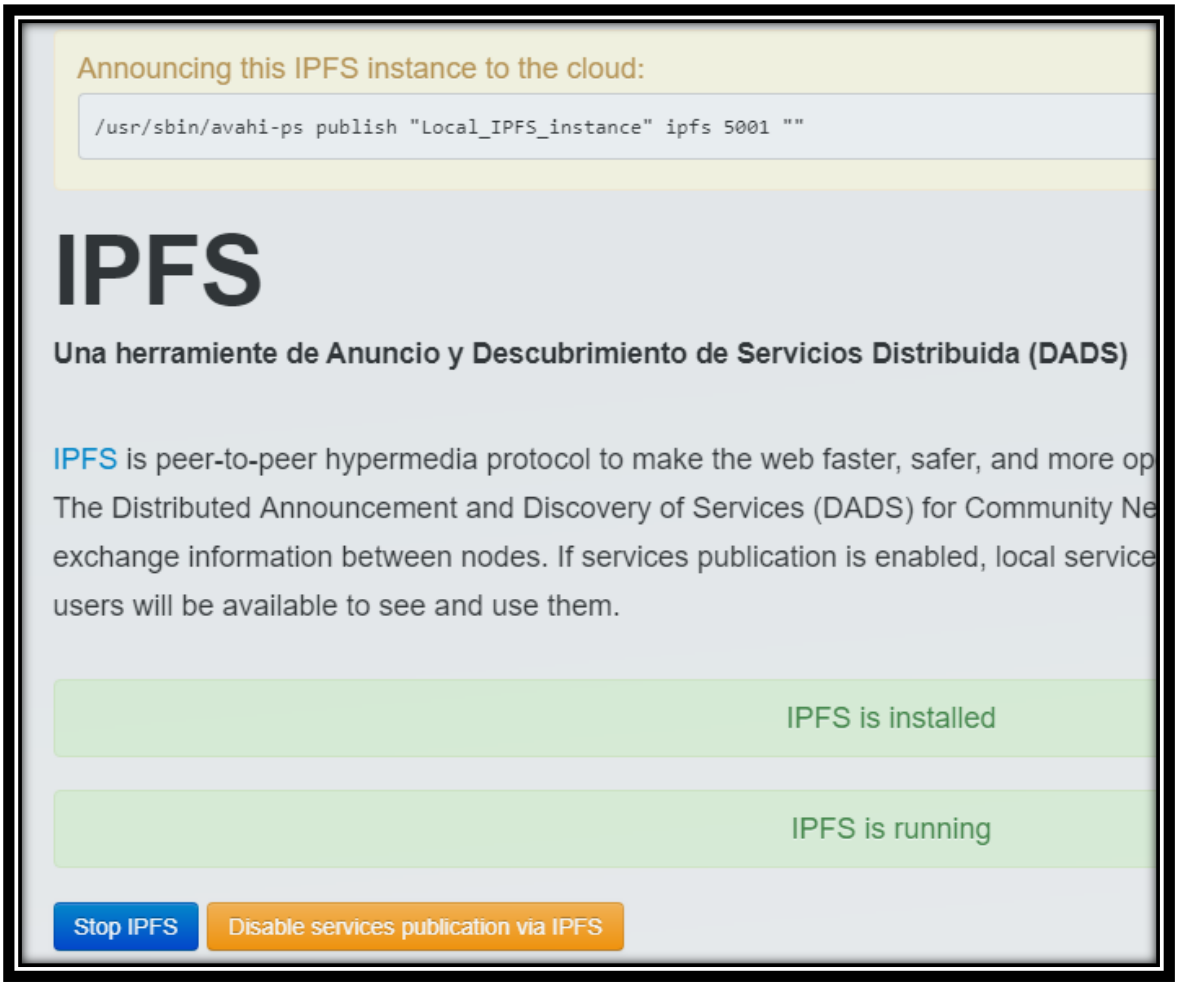


Figura 69: ICloudy2. Publicación IPFS

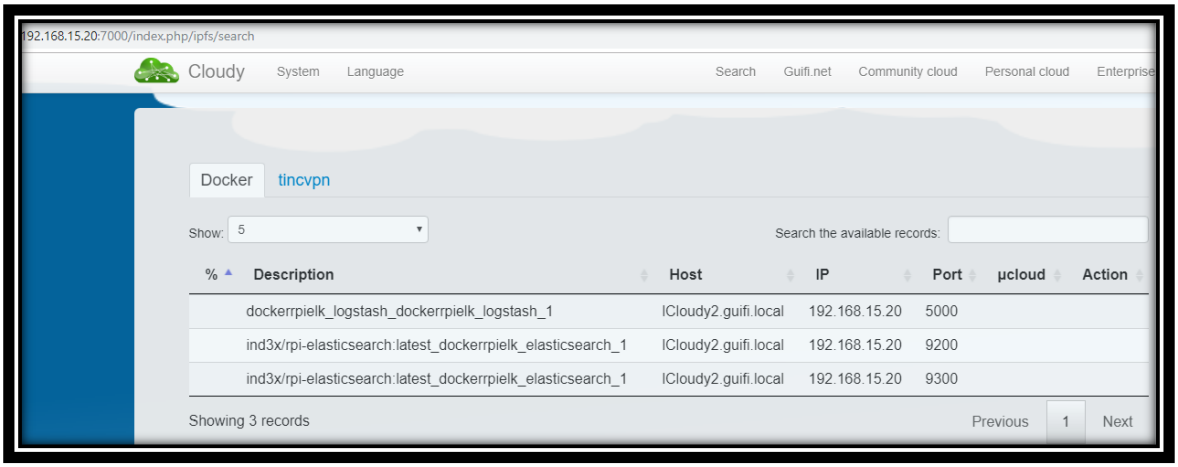


Figura 70: ICloudy2. IPFS search

6.2 IPFS

En este subcapítulo se desarrollan diferentes pruebas con IPFS. Algunas de estas pruebas, como operar con IPFS, se consideran de evaluación debido al grado de madurez en el que se encuentra el protocolo. Por otro lado, se tiene el

objetivo de aportar un hilo conductor del capítulo de instalación y despliegue de IPFS e IPFS-Cluster.

6.2.1 Visualización y adición de peers a red IPFS

Test 10

Objetivo: Visualizar nodos (*peers*) dentro de la red IPFS.

Fuentes de variación: Se piensa en un primer momento que podría afectar la conectividad ipv4/ipv6, direccionamiento, etc. pero finalmente no aplica por el modo de funcionamiento de IPFS.

Desarrollo: Se dispone de varios nodos Cloudy con IPFS inicializado. Por lo tanto, una vez se haya inicializado IPFS y arrancado el demonio se pueden ver los *peers* conectados al entorno con el siguiente comando:

- ipfs swarm peers

Resultado: Se observa en la siguiente figura como tras arrancar el demonio los peers empiezan a aparecer, después de unos minutos, la lista va aumentando.

```
pi@lCloudy2:~ $ ipfs swarm peers
/ip4/104.131.131.82/tcp/4001/ipfs/QmaCpDMGvV2BGHeYERUEnRQAwe3N8SzbUtfsmvsqQLuvuJ
/ip4/128.199.219.111/tcp/4001/ipfs/QmSoLSafTMBsPKadTEgaXctDQVcqN88CNLHXMkTNwMKPnu
pi@lCloudy2:~ $
```

Figura 71: lCloudy2. IPFS swarm peers

Esto es debido a una unos ficheros de semilla, parametrizables, que incorporan diferentes *peers* predefinidos.

Por otro lado, IPFS dispone de algunos servicios predeterminados que se ejecutan por defecto como pueden ser; dht, bitswap, y el servicio de diagnóstico. Cada uno de estos servicios simplemente registra un controlador en el IPFS *PeerHost*, y escucha en busca de nuevas conexiones. Esta visualización de nodos de la red IPFS se puede realizar también vía web.

IPFS dispone de una consola web donde se puede comprobar, de una manera sencilla y visual, el estado del nodo; PeerID, versión, etc., y los *peers*:

- `http://localhost:5001/webui`

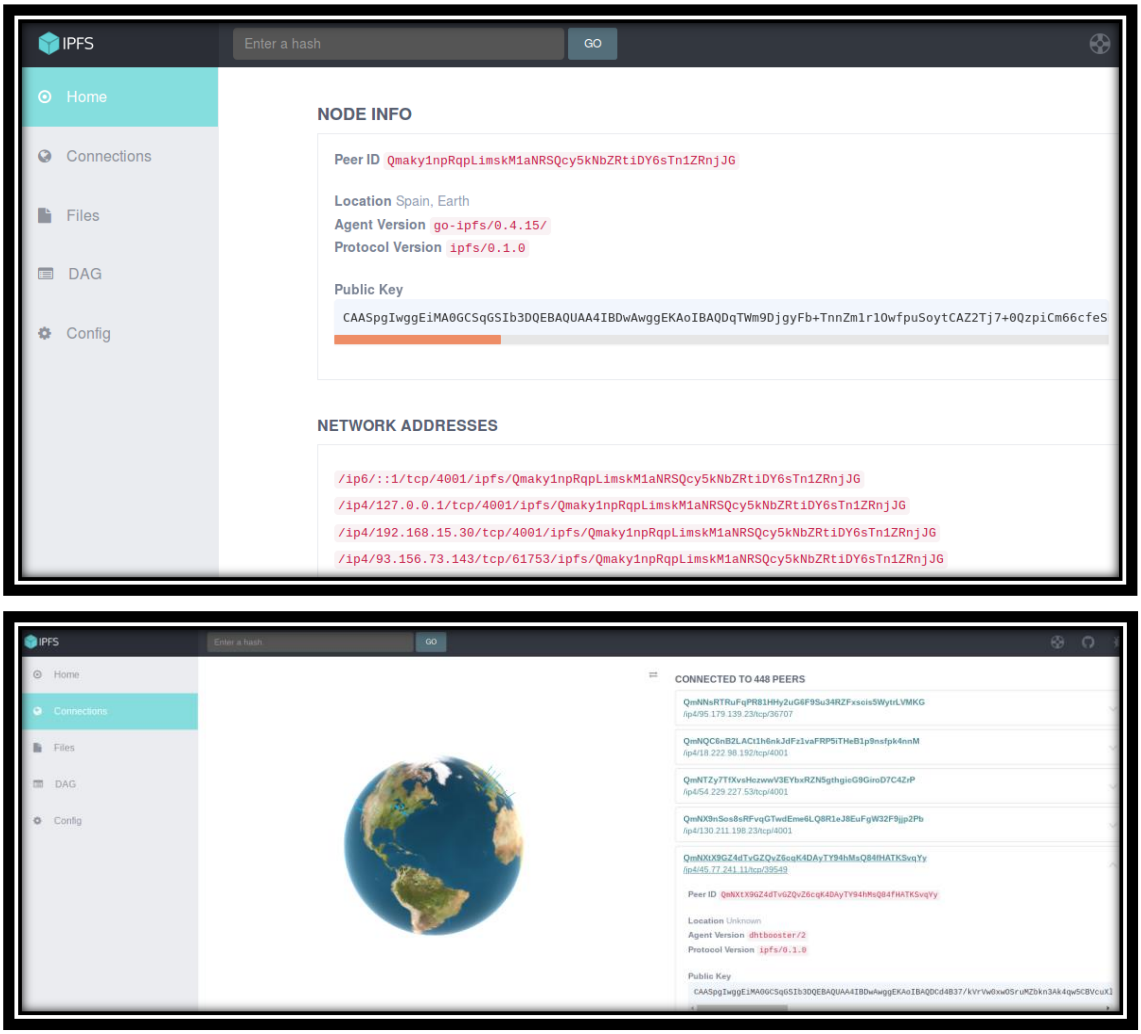


Figura 72: Acceso IPFS WebUI

Test 11

Objetivo: Conectar nodos manualmente dentro de la red IPFS.

Fuentes de variación: PeerID valido. Puerto TCP no bloqueado o en uso.

Desarrollo: Una vez se conecte la red IPFS se pueden ver los *peers* pero quizás no aparezca el nodo como miembro de la red IPFS o sea interesante añadir manualmente algún otro nodo. Se podría conectar manualmente a otro *peer* ejecutando la siguiente sentencia:

- ipfs swarm connect /ip4/<IP_del_nodo>/tcp/4001/ipfs/<Peer id>

Resultado: El resultado es positivo, se puede observar cono la lista de peers ha crecido significativamente y los hosts añadidos manualmente aparecen en la red IPFS. Es decir, el entorno de Cloudy local figura ya en la red IPFS global o semipública.

Esta sentencia se puede ver en la siguiente figura, además de la visualización de los peers lCloudy1 y lCloudy3, vistos desde lCloudy2.

```
pi@lCloudy2:~/Downloads/ipfs-cluster-service $ ipfs swarm connect /ip4/192.168.15.10/tcp/4001/ipfs/QmQvPGSGxNk89ct
5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
connect QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY success
pi@lCloudy2:~/Downloads/ipfs-cluster-service $  ipfs swarm peers | grep 192.168.15
/ip4/192.168.15.10/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
/ip4/192.168.15.30/tcp/4001/ipfs/QmakylnpRqpLimskMlaNRSQcy5kNbZRtiDY6sTnlZRnjJG
```

```
leopoldo@lCloudy3:~/Descargas/ipfs-cluster-service$ ipfs swarm peers
/ip4/101.251.230.213/tcp/4001/ipfs/QmZEvpGZfGqNFtAVGgF2ntJnsLiWhzB2C9Kiccr5Wzb2hk
/ip4/104.131.122.90/tcp/4001/ipfs/QmRzieLjw9zMhdqzQLkVoVSK9T53Uouyyb3ZiSg3RE6Efq
/ip4/104.248.31.20/tcp/4001/ipfs/QmTUufmYD5yzqHC9cLFMYEfymAeYbzw175zPEKSP9PMnLp
/ip4/107.22.151.244/tcp/4001/ipfs/QmXLe5krX55mk9pj26wbSvXoNuCUqeGUeCC4agi5tSiv2M
/ip4/108.241.29.84/tcp/4001/ipfs/QmdLM1bYZP5AhaLeNCpycEt9qGqf5a6o3t1Z9NwYVwhT1f
/ip4/108.61.169.87/tcp/4001/ipfs/QmXNFp6YrGvpBkwF8DxSw9BiiuZ29HUaXNvcsGhQvH9daF
/ip4/108.61.198.208/tcp/4001/ipfs/Qmc717H1hpCRu3dg7WZBqrY8cTWPGiU5xoRVrtPyWhBVre
/ip4/109.147.155.221/tcp/43543/ipfs/Qmawq7YwJQEc7viD2j5PcVJ33vnPC4aryCuqynZDu5dM5r
/ip4/112.10.205.19/tcp/48234/ipfs/QmNxfV73YefQHePPbDv7rJzze6ng5Pi3jPprh6BbpunYPT
/ip4/114.45.206.83/tcp/40127/ipfs/QmUE1vbvjU7DbQN254hbaxqPTBP1YYukxbRXGEpviNw7if
/ip4/115.78.231.221/tcp/4002/ipfs/QmadJHeUbYv6dJhobhiWYXwNwFW5GYF4VFmYFnkLV66Fqo
/ip4/122.146.85.154/tcp/4001/ipfs/QmPpJi4V7KfWb7YgLHePeA8XzCp9YzaQn3amVrWUTsKedV
/ip4/129.213.113.152/tcp/4001/ipfs/QmUAV2Eb6o4g4qL5fpCyz28vATVHcJRJQo1kv65TXgb1jY
/ip4/13.251.104.165/tcp/4001/ipfs/QmZ2FsnaxEzSHaN18yL3zwtjMTTgffkfLmHfXZKtsq6Mqw
/ip4/130.211.198.23/tcp/4001/ipfs/QmNX9nSos8sRFvqGTwdEme6LQ8R1eJ8EuFgW32F9jjp2Pb
/ip4/130.74.96.25/tcp/4001/ipfs/QmSeFpBkYmPajLapFv2Bob6e56KJb7rBpWP31PphMdcEhm
/ip4/136.25.18.241/tcp/63742/ipfs/QmXj5vKEKSU4kkjGnWq9ZJeG4xrkacDugme4iXz6qtvPgN
/ip4/138.197.165.79/tcp/4001/ipfs/Qmcmm2Ua9RsfXjKFb4rv7gUa4TuHxP6mMZpFE1TDsvMi64
/ip4/138.197.68.168/tcp/4001/ipfs/QmR4TkYZKeConfKh2o5HTwsyQ7gCLSaExjroiEaoit5iMk
/ip4/138.197.70.140/tcp/4001/ipfs/QmQDj7tke8fBby556Bdsbr6hMa4xAgqov3EoiGYSScGtRh
/ip4/138.201.67.218/tcp/4001/ipfs/QmbVWZQhCGrS7DhgLqWbgvdmKN7JueKCREVanfnVpgyq8x
/ip4/138.68.6.138/tcp/4001/ipfs/QmVXEtCV8s9R6aMf6HbY9C92R2nMNzwxPjj2NimCtK3WLw
/ip4/142.93.188.196/tcp/4001/ipfs/QmZBo6gmPEhnMxwRkozEf2hiyekifiJptWnmM7jGuSJnkE
/ip4/142.93.45.202/tcp/4001/ipfs/QmVxzMnDqXBExQtSuW4VVu6WTaxZ4eHY6TDjKYSoFB79Z1
/ip4/144.202.48.171/tcp/33597/ipfs/QmRHxhqP8e8F11qHb46PbXvEiJut2qttvAUU4hQ8s3tCR8
```

Figura 73: IPFS swarm connect. Añadiendo *peer* red IPFS

6.2.2 Carga y descarga de ficheros de la red IPFS

A continuación, se realizan pruebas de almacenamiento distribuido en la red IPFS. Estas pruebas se dividen en acceso a ficheros dentro de la red desde diferentes nodos, creación, publicación y descarga de ficheros dentro del ámbito de este proyecto.

Test 12

Objetivo: Generar un fichero de datos cualquiera y subirlo a la red IPFS

Fuentes de variación: Formar parte red IPFS.

Desarrollo: Se crea la carpeta “files” en lCloudy1 y la carpeta “files3” en lCloudy3, y se suben unos ficheros básicos de tipo HTML para esta y pruebas posteriores. En el caso de esta prueba, se crea un fichero con un pequeño

fragmento de código, y a continuación, con el hash o cid generado se sube a la red mediante:

- ipfs add -w CloudyTest.html.

Resultado: Se sube el fichero desde lCloudy1 a la red IPFS y se observa cómo se puede acceder a él desde otros nodos, como puede ser lCloudy3.

Por un lado, se confirma que cuando se conecta un nodo a una red IPFS global, se pueden añadir ficheros a dicha red, y por otro, que una vez esté disponible el fichero se puede descargar o visualizar desde cualquier otro nodo.

```
pi@lCloudy1:~/.ipfs/files $ ipfs add -w CloudyTest.html
added QmWVZ6fUCizR7wW1A3uD8ACFPH6YrTsUo2Puojgi46NXVc CloudyTest.html
added QmcMhzmRdB2Fsot94gyDfigtvdxVx9sYmgEssDhLzebVrm
```

```
leopoldo@lCloudy3:~/Descargas/ipfs-cluster-ctl$ ipfs cat QmWVZ6fUCizR7wW1A3uD8ACFPH6YrTsUo2Puojgi46NXVc
<html>
<body>
<h1> TFG Clouds comunitarias e IPFS </h1>
<h2> TESTING IPFS: The distributed Web</h2>
<h3> Adding HTML file to the IPFS network </h3>
</body>
</html>

leopoldo@lCloudy3:~/Descargas/ipfs-cluster-ctl$
```

Figura 74: Añadiendo ficheros a la red IPFS

Test 13

Objetivo: Descargar un fichero de datos de IPFS

Fuentes de variación: No aplica.

Desarrollo: Existen unos ficheros, dentro de la red IPFS global, que suelen tomarse para pruebas, debido a que aparecen en todas las guías de iniciación a IPFS o en diferentes conferencias. Estos ficheros, entre otros, son una imagen de un gato, ficheros de audio/video, etc. Se comprueba que mediante el *hash* disponible se accede sin problema desde cualquiera de los nodos donde corre IPFS.

Resultado: El resultado es satisfactorio debido a que mediante la secuencia vista previamente “ipfs cat”, podemos descargar el contenido del fichero a nuestro nodo. El resultado se puede ver en las figuras siguientes donde se descarga el fichero a lCloudy3 redireccionando el contenido del hash al fichero cat.jpg y, acto seguido, se abre el mismo en el nodo. Se realiza la misma operación con un fichero de texto que contiene la sentencia “Hello world”.

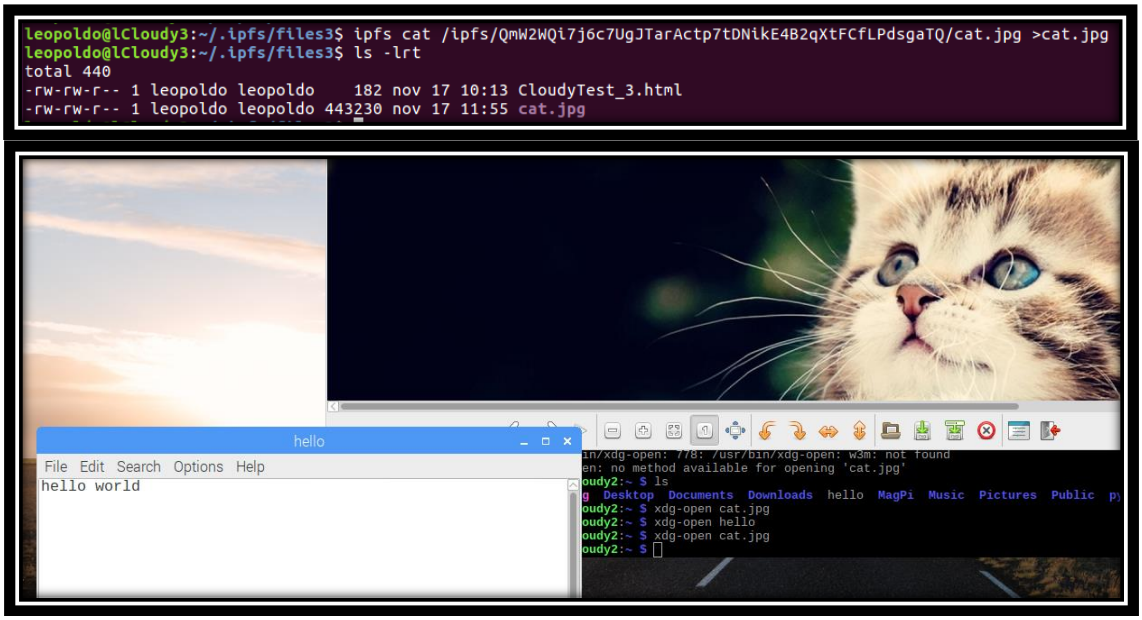


Figura 75: Descargando ficheros de la red IPFS

Test 14

Objetivo: Descargar un fichero de datos de IPFS subido desde un nodo que está caído o no se encuentra activo en la red IPFS.

Fuentes de variación: No aplica.

Desarrollo: Se descarga un fichero de audio mp3 bajo licencia FREE Creative Commons de la web Bensound [51], debido a que vamos a subirlo a la red IPFS y hacer uso de este. Una vez descargado, se añade a la red IPFS desde el nodo lCloudy3 del mismo modo que se ha realizado en pruebas previas.

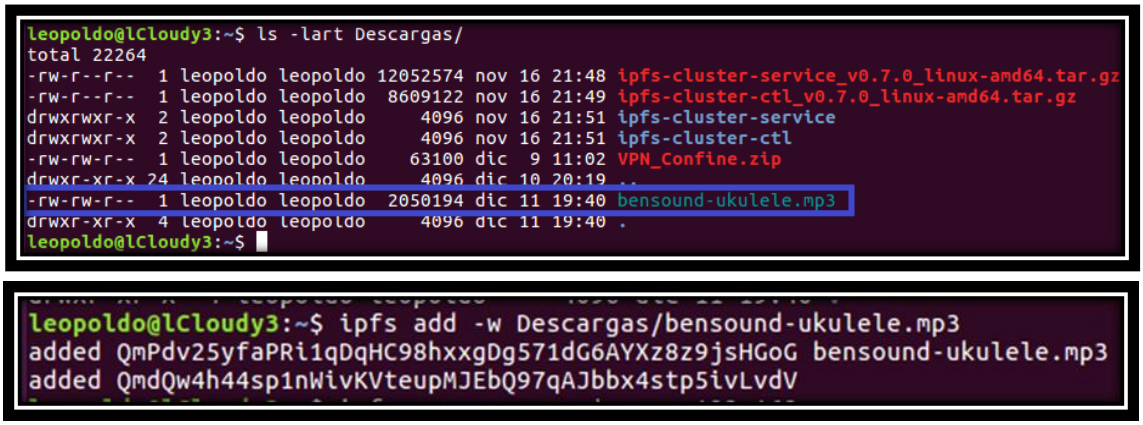


Figura 76: Descarga mp3 de la web y carga mp3 red IPFS

Resultado: El resultado es satisfactorio. Se realiza la carga del fichero bensound-ukulele.mp3 a la red IPFS proporcionando un hash en lCloudy3.

Acto seguido es viable la descarga dicho fichero desde lCloudy2.

```
pi@lCloudy2:~/.ipfs $ w
 18:49:42 up 3 days,  6:56,  3 users,  load average: 0.42, 0.33, 0.22
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
pi       tty1     -                Sat11    3days  0.31s  0.24s -bash
pi       tty7     :0               Sat11    3days  2:18   1.47s /usr/bin/lxsession -s LXDE-pi -e LXDE
pi       pts/0    192.168.15.11    17:31    3.00s  2:58   0.02s w
pi@lCloudy2:~/.ipfs $ ipfs cat QmPdv25yfaPRilqDqHC98hxxgDg571dG6AYXz8z9jsHGoG > audio.mp3
pi@lCloudy2:~/.ipfs $ w
 18:49:46 up 3 days,  6:56,  3 users,  load average: 0.42, 0.33, 0.22
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
pi       tty1     -                Sat11    3days  0.31s  0.24s -bash
pi       tty7     :0               Sat11    3days  2:18   1.47s /usr/bin/lxsession -s LXDE-pi -e LXDE
pi       pts/0    192.168.15.11    17:31    2.00s  2:59   0.03s w
pi@lCloudy2:~/.ipfs $
```

```
pi@lCloudy2:~/.ipfs $ ls -lart
total 2052
-rw-r--r--   1 pi pi       2 Oct 23 18:45 version
-rw-------   1 pi pi     190 Oct 23 18:45 datastore_spec
-rw-rw----   1 pi pi    5123 Oct 23 18:45 config
drwx------   2 pi pi    4096 Oct 23 18:45 keystore
drwxr-xr-x  28 pi pi    4096 Dec 11 17:31 ..
-rw-r--r--   1 pi pi       0 Dec 11 18:37 repo.lock
drwxr-xr-x   2 pi pi    4096 Dec 11 18:37 datastore
-rw-r--r--   1 pi pi      23 Dec 11 18:37 api
drwxr-xr-x 411 pi pi   12288 Dec 11 18:49 blocks
drwxr-xr-x   5 pi pi    4096 Dec 11 18:49 .
-rw-r--r--   1 pi pi 2050194 Dec 11 18:49 audio.mp3
pi@lCloudy2:~/.ipfs $
```

Figura 77: Descarga mp3 desde red IPFS

A continuación, se realiza la parada del demonio IPFS en lCloudy3. Se elimina como *peer* de la red de cara a realizar la comprobación real de la prueba, realizar la descarga de un fichero con el nodo desde donde se ha realizado la carga en estado off-*line* o indisponible.

```
root@lCloudy3:~/.ipfs# ipfs swarm peers | grep 192.168.15
/ip4/192.168.15.20/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
root@lCloudy3:~/.ipfs#
root@lCloudy3:~/.ipfs# jobs
[1]+  Ejecutando              ipfs daemon &
root@lCloudy3:~/.ipfs# fg 1
ipfs daemon
^C
Received interrupt signal, shutting down...
(Hit ctrl-c again to force-shutdown the daemon.
root@lCloudy3:~/.ipfs# exit
logout
leopoldo@lCloudy3:~$
```

```
pi@lCloudy2:~/.ipfs $ ipfs swarm peers | grep 192.168
pi@lCloudy2:~/.ipfs $
```

Figura 78: Parada demonio IPFS *peer* lCloudy3

Como se puede ver en las figuras previas, se realiza la parada en lCloudy3 y no se detecta como *peer* valido desde lCloudy2. Sin embargo, es posible realizar la descarga, se realiza la misma en pocos segundos.

```
pi@lCloudy2:~/.ipfs $ ipfs cat QmPdv25yfaPRilqDqHC98hxxgDg571dG6AYXz8z9jsHGoG > audio.mp3
pi@lCloudy2:~/.ipfs $ ls -lrt
total 2044
-rw-r--r--   1 pi pi       2 Oct 23 18:45 version
-rw-------   1 pi pi     190 Oct 23 18:45 datastore_spec
-rw-rw----   1 pi pi    5123 Oct 23 18:45 config
drwx------   2 pi pi    4096 Oct 23 18:45 keystore
-rw-r--r--   1 pi pi       0 Dec 11 18:37 repo.lock
drwxr-xr-x   2 pi pi    4096 Dec 11 18:37 datastore
-rw-r--r--   1 pi pi      23 Dec 11 18:37 api
drwxr-xr-x 411 pi pi   12288 Dec 11 18:49 blocks
-rw-r--r--   1 pi pi 2050194 Dec 11 18:52 audio.mp3
pi@lCloudy2:~/.ipfs $ w
 18:52:07 up 3 days,  6:59,  3 users,  load average: 0.35, 0.39, 0.26
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
pi       tty1     -                Sat11    3days  0.31s  0.24s -bash
pi       tty7     :0               Sat11    3days  2:18   1.47s /usr/bin/lxsession -s LXDE-pi -e LXDE
pi       pts/0    192.168.15.11    17:31    7.00s  3:32   0.03s w
pi@lCloudy2:~/.ipfs $
```

Figura 79: Comprobación parada *peer* lCloudy3. Descarga fichero desde IPFS

6.2.3 IPFS Cluster. Red IPFS privada

Se ha visto en el capítulo de despliegue como se realiza la configuración de un clúster privado de IPFS. Este capítulo se considera de especial utilidad debido a los dos siguientes motivos:

- Toda la información que se suba a la red IPFS global queda registrada. Por lo tanto, debido a que se está realizando una batería de pruebas vinculadas a un TFG, no se pretende subir más contenido que el meramente necesario.
- Como se ha indicado anteriormente, se considera IPFS Cluster útil para integrar en Cloudy, y generar un *swarm* acotado a entornos comunitarios de carácter privado.

Se realiza la siguiente batería de pruebas:

Test 15

Objetivo: Subida de un fichero al *swarm* privado o clúster IPFS.

Fuentes de variación: Conectividad TCP/IP.

Desarrollo: Se aprovecha la estructura creada en la prueba de subida de ficheros a la red IPFS. Existe una carpeta “files” en lCloudy1 y una carpeta “files3” en lCloudy3, donde se han generado unos ficheros básicos de tipo HTML

para esta y otras pruebas. En el caso de esta prueba, se crea un fichero con un pequeño fragmento de código, y a continuación, con el *hash* o *cid* generado se sube al clúster mediante la siguiente ejecución:

- ipfs-cluster-ctl add <nombre_fichero> (carga de fichero al *swarm* privado)

Si se remonta a la prueba donde el fichero se subía a la red IPFS, cabe señalar nuevamente el comando para revisar la diferencia:

- ipfs add -w CloudyTest.html (subida de fichero a la red IPFS)

Resultado: Se realiza la subida del fichero al *swarm* o clúster IPFS privado. Se puede observar, en la figura siguiente, como se genera el hash y aparecen los logs del fichero añadido al clúster realizando el *pinning* del fichero.

```
leopoldo@lCloudy3:~/.ipfs/files3$ ipfs-cluster-ctl add CloudyTest_3.html
added QmTtEUeXCJvkHu5JerA2TyKLSJcMN1fA5Hof276ACwsRNk CloudyTest_3.html
10:18:41.761  INFO    cluster: IPFS cluster pinning QmTtEUeXCJvkHu5JerA2TyKLSJcMN1fA5Hof276ACwsRNk everywhere:
 cluster.go:1010

10:18:44.219  INFO      adder: QmTtEUeXCJvkHu5JerA2TyKLSJcMN1fA5Hof276ACwsRNk successfully added to cluster ad
der.go:162
leopoldo@lCloudy3:~/.ipfs/files3$
leopoldo@lCloudy3:~/.ipfs/files3$ 10:18:44.274  INFO   ipfshttp: IPFS Pin request succeeded:  QmTtEUeXCJvkHu5J
erA2TyKLSJcMN1fA5Hof276ACwsRNk ipfshttp.go:593

leopoldo@lCloudy3:~/.ipfs/files3$ ipfs-cluster-ctl pin ls
QmTtEUeXCJvkHu5JerA2TyKLSJcMN1fA5Hof276ACwsRNk |  | PIN | Repl. Factor: -1 | Allocations: [everywhere] | Recur
sive
```

Figura 80: Añadiendo fichero red IPFS y haciendo *pinning*

Automáticamente, se detecta el *pinning* en los otros nodos del cluster:

```
10:18:46.043 INFO   ipfshttp: IPFS Pin request succeeded:
QmTtEUeXCJvkHu5JerA2TyKLSJcMN1fA5Hof276ACwsRNk ipfshttp.go:593
TtEUeXCJvkHu5JerA2TyKLSJcMN1fA5Hof276ACwsRNk ipfshttp.go:593QmT
```

Estas operaciones, como se ha indicado, generan el hash que identifica el fichero en la red, y se debe conocer para acceder al contenido, lo cual es una medida implícita de seguridad, ya que sin conocer ese *hash* no se puede acceder al fichero.

Al tratarse de ficheros HTML se accede vía IPFS desde un navegador como si de http se tratase.

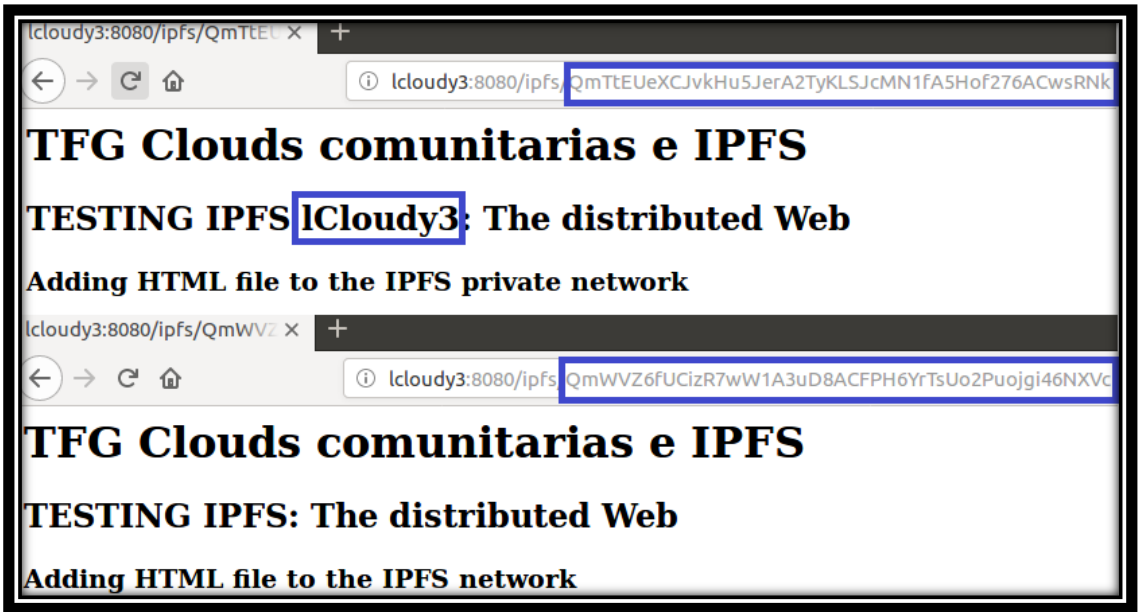


Figura 81: Acceso vía web a diferentes ficheros HTML desde lCloudy3

En la figura anterior se puede ver como desde el nodo lCloudy3 se accede al fichero subido desde el nodo lCloudy1, y al fichero subido desde el propio lCloudy3. Esta acción se podría realizar desde cualquier miembro del clúster con el *hash* correspondiente.

6.2.4 Gateway HTTP IPFS y Windows

Se desarrolla este subapartado dentro del capítulo de evaluación debido a que se considera una vía más de evaluar IPFS, sobre todo si se pretende un primer contacto o acercamiento con el protocolo.

Test 16

Objetivo: Visualización ficheros IPFS desde nodos no IPFS.

Fuentes de variación: Navegadores soportados. *Proxies* válidos.

Desarrollo: Se puede utilizar IPFS o acceder a ficheros de su red sin disponer de una instalación previa, es decir, existe una característica que hace posible acceder a ficheros en la red IPFS mediante la utilización de una puerta de acceso o *gateway* IPFS. No obstante, únicamente se puede realizar este tipo de visualización. La puerta de enlace asume la responsabilidad de llevar el contenido IPFS al “mundo” HTTP. Cada nodo dispone de un servidor HTTP, no obstante, los *gateways* más conocidas son:

- IPFS.io: https://ipfs.io/ipfs/<hash>
- CloudFare: https://cloudflare-ipfs.com/ipfs/ <hash>

Resultado: Mediante esta utilidad, empleando las pasarelas indicadas, se puede acceder a la información, como se puede ver en la figura siguiente, desde cualquier navegador como Chrome, Firefox, etc. Por otro lado, resulta útil para integrar IPFS en aplicaciones o entornos que no dispongan de una librería específica.

Figura 82: Acceso vía web desde terminal Windows y navegador Firefox a CloudyTest-html

Test 17

Objetivo: Visualización ficheros IPFS desde nodos Windows.

Fuentes de variación: Versiones S.O.

Desarrollo: Se puede utilizar IPFS o acceder a la red IPFS desde entornos Windows mediante software del proyecto Epona. Se instala el software Epona y únicamente se revisa su viabilidad, ya que está fuera del alcance del proyecto.

Resultado: Epona es una herramienta en fase alpha para administrar IPFS en Windows. El objetivo es hacer que compartir archivos en IPFS sea más simple y accesible. Se instala el ejecutable y se arranca el demonio IPFS en Windows automáticamente. Se crean las estructuras que cuelgan de "./ipfs" igual que en GNU/Linux.

6.3 OrbitDB, Orbit e IPFSLog

Test 18

Objetivo: Creación base de datos OrbitDB en nodo aislado.

Fuentes de variación: No aplica

Desarrollo: En este primer test se pretende realizar un acercamiento práctico con OrbitDB. Creación de una base de datos de tipo documental, se generan dos documentos y se suben al repositorio de la base de datos con el objetivo de comprobar el funcionamiento, tanto de línea de comandos, generación del hash de los documentos, etc.

Resultado: Se ejecuta mediante OrbitDB CLI la creación de una base de datos denominada “mydocstore”, de tipo docstore. A continuación, como se puede ver en la figura siguiente, se suben dos documentos; “lCloudy1.txt” y “lCloudy1.readme”

```
root@lCloudy1:/home/pi/orbit-db-cli# node src/bin.js create /orbitdb/mydocstore docstore
/orbitdb/QmYXp8Jh3kDlgBE5MJsCGfiJ6BM2gB39hEoRXxCKjjBs59/orbitdb/mydocstore
root@lCloudy1:/home/pi/orbit-db-cli# node src/bin.js put /orbitdb/QmapaR6rmS3JgTQBKSpSZUdqgf2t8V
id": 1}' --indexBy name
Error: Key not found in database [/IX/CIQLS43UZ7Q4Y7O347NICDYLV6XC7VRJ7TNAMNKDQKQDUEUUZXJXIXA]
root@lCloudy1:/home/pi/orbit-db-cli# node src/bin.js put /orbitdb/QmYXp8Jh3kDlgBE5MJsCGfiJ6BM2gB
id": 1}' --indexBy name
Added document 'lCloudy1.txt'
root@lCloudy1:/home/pi/orbit-db-cli# node src/bin.js put /orbitdb/QmYXp8Jh3kDlgBE5MJsCGfiJ6BM2gB
 "_id": 1}' --indexBy name
^C
root@lCloudy1:/home/pi/orbit-db-cli# node src/bin.js put /orbitdb/QmYXp8Jh3kDlgBE5MJsCGfiJ6BM2gB
" id": 2}' --indexBy name
Added document 'lCloudy1.readme'
```

Figura 83: Creación BBDD OrbitDB tipo doc

Se añaden correctamente a la base de datos. Por otro lado, la segunda entrada de la figura indica un fallo de clave no encontrada en la base de datos. Esto se debe al intentar añadir un documento con una clave diferente a la generada en la primera sentencia.

Test 19

Objetivo: Creación base de datos OrbitDB de tipo Log y Feed.

Fuentes de variación: No aplica.

Desarrollo: Se crea una base de datos de tipo feed o evenlog, y se realizan operaciones para evaluar el funcionamiento. Añadir entradas, eliminar entradas, solicitar información de la base de datos, etc.

Resultado: Se crea una base de datos de tipo “eventlog” y se añaden varias entradas simulando un log de Cloudy de manera satisfactoria:

```
pi@lCloudy2:~/orbit-db-cli $ sudo node ./src/bin.js create CloudyLogs eventlog
/orbitdb/QmZVb9vJk9xbek3ZfRCYapvEnvmeYsiPRgseNn1PPu3qid/CloudyLogs
```

```
pi@lCloudy2:~/orbit-db-cli$ sudo node ./src/bin.js add
/orbitdb/QmZVb9vJk9xbek3ZfRCYapvEnvmeYsiPRgseNn1PPu3qid/CloudyLogs "ERR: lCloudy1 peer
connect"
Added Qme1BPeVkTYpt4bEtvQiefMHi4TKeA6PdMtdBhEH4mHM6F
pi@lCloudy2:~/orbit-db-cli$sudo node ./src/bin.js add
/orbitdb/QmZVb9vJk9xbek3ZfRCYapvEnvmeYsiPRgseNn1PPu3qid/CloudyLogs "ERR: lCloudy2 peer
connect"
Added QmSUNTDZ4qTRRPi2WS7UMQnefEqZd2Jec26dRY1ZdgdBTi
pi@lCloudy2:~/orbit-db-cli $ sudo node ./src/bin.js add
/orbitdb/QmZVb9vJk9xbek3ZfRCYapvEnvmeYsiPRgseNn1PPu3qid/CloudyLogs "WARN: lCloudy2 service
Kibana is down"
Added QmaoFw6mtMsCcb17ApdJNVa1uF7NVqY97tdLXunMtFu2xj
```

A continuación, se solicita información de la base de datos creada:

```
pi@lCloudy2:~/orbit-db-cli $ sudo node ./src/bin.js get
/orbitdb/QmZVb9vJk9xbek3ZfRCYapvEnvmeYsiPRgseNn1PPu3qid/CloudyLogs
"ERR: lCloudy1 peer connect"
"ERR: lCloudy2 peer connect"
"WARN: lCloudy2 service Kibana is down"
pi@lCloudy2:~/orbit-db-cli $ sudo node ./src/bin.js info
/orbitdb/QmZVb9vJk9xbek3ZfRCYapvEnvmeYsiPRgseNn1PPu3qid/CloudyLogs
/orbitdb/QmZVb9vJk9xbek3ZfRCYapvEnvmeYsiPRgseNn1PPu3qid/CloudyLogs
> Type: eventlog
> Owner: /orbitdb/QmZVb9vJk9xbek3ZfRCYapvEnvmeYsiPRgseNn1PPu3qid/CloudyLogs
> Data file: ./orbitdb/QmZVb9vJk9xbek3ZfRCYapvEnvmeYsiPRgseNn1PPu3qid/CloudyLogs
> Entries: 3
> Oplog length: 3 / 3
> Write-access:
```

Se adjunta figura simuilar de base de datos de tipo “feed”, donde se pueden eliminar entradas quedando registro de la acción.

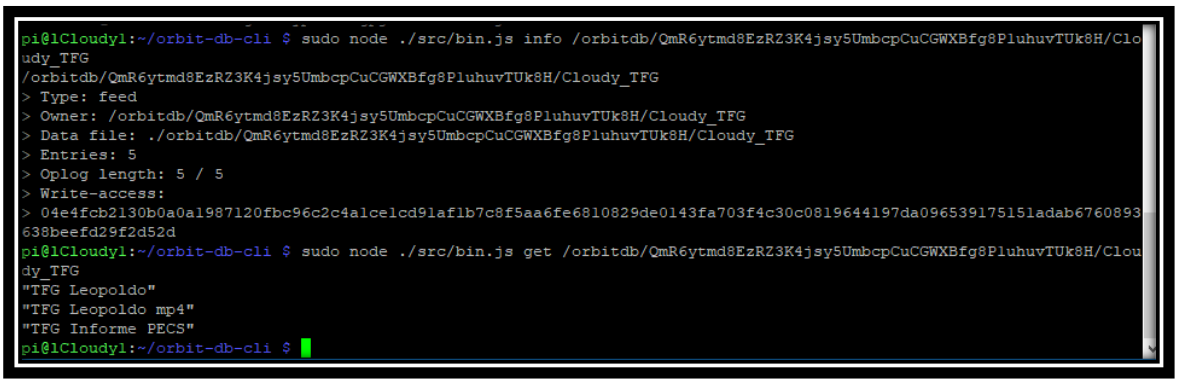


Figura 84: Consulta BBDD OrbitDB tipo feed

Test 20

Objetivo: Creación base de datos OrbitDB y replica entre nodos IPFS.

Fuentes de variación: Estado de los demonios IPFS.

Desarrollo: Se crea una BBDD de tipo “feed” o “eventlog” y se replica a cualquiera de los nodos que forman la microCloud. La replicación no está soportada en bases de datos de tipo “docstore” a fecha de este trabajo. El objetivo de la prueba es evaluar la distribución del servicio de base de datos entre nodos. En concreto, se crea una base de datos de tipo “eventlog”, se añaden entradas y se replica. Posteriormente, se añaden nuevas entradas y se eliminan, revisando la actualización en el nodo donde se ha replicado.

Resultado: Se crea una base de datos de tipo “eventlog” y se añaden varias entradas, al igual que en la prueba anterior.

```
pi@lCloudy1:~/orbit-db-cli $ sudo node ./src/bin.js create CloudyLogs eventlog

/orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs

pi@lCloudy1:~/orbit-db-cli $ sudo node ./src/bin.js add
/orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs "ERR: lCloudy1 peer connect"

Added QmSQphU611pxB2gjGfth4RUyaPJ1xTxbyVcmcod4nvAe8K

pi@lCloudy1:~/orbit-db-cli $ sudo node ./src/bin.js add
/orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs "INFO: lCloudy2 peer is up"

Added QmcMFhjiiQQoqBZDGx3iCBGUQZ7jHbUAFz4xRGenysKDYi

pi@lCloudy1:~/orbit-db-cli $ sudo node ./src/bin.js add
/orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs "INFO: rCloudy3 peer is up"

Added QmdLzXrWdYnETqeqXRDSEKrdShJxwNSkmqvzTg7JmY1HL5

pi@lCloudy1:~/orbit-db-cli $ sudo node ./src/bin.js add
/orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs "WARN: lCloudy3 disconnect"

Added Qmahe1VLV7eMbVfHzma7gzxoRaQEiTV1UaThrVUYYhFXNR

pi@lCloudy1:~/orbit-db-cli $ sudo node ./src/bin.js add
/orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs "WARN: lCloudy2 service Kibana
is down"

Added QmUGdvgdudfunHArbmMzhyd2PLe7JvCvdsQgiALnVmHuKe

pi@lCloudy1:~/orbit-db-cli $ sudo node ./src/bin.js get
/orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs --dashboard

"ERR: lCloudy1 peer connect"

"INFO: lCloudy2 peer is up"

"INFO: rCloudy3 peer is up"

"WARN: lCloudy3 disconnect"

"WARN: lCloudy2 service Kibana is down"
```

Acto seguido, desde cualquier nodo de nuestra red IPFS privada o global, activamos la replicación:

```
pi@lCloudy2:~/orbit-db-cli $ sudo node ./src/bin.js replicate
/orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs
Swarm listening on /ip4/127.0.0.1/tcp/36987/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn51z6uqUD8wmGQJ4NeSd
Swarm listening on
/ip4/192.168.15.20/tcp/36987/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn51z6uqUD8wmGQJ4NeSd
Swarm listening on /ip4/172.17.0.1/tcp/36987/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn51z6uqUD8wmGQJ4NeSd
Swarm listening on
/ip4/169.254.241.43/tcp/36987/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn51z6uqUD8wmGQJ4NeSd
Swarm listening on
/ip4/10.228.207.206/tcp/36987/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn51z6uqUD8wmGQJ4NeSd
Swarm listening on
/ip4/10.228.207.39/tcp/36987/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn51z6uqUD8wmGQJ4NeSd
Swarm listening on
/ip4/169.254.107.240/tcp/36987/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn51z6uqUD8wmGQJ4NeSd
Swarm listening on
/ip4/169.254.223.146/tcp/36987/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn51z6uqUD8wmGQJ4NeSd
Swarm listening on
/ip4/169.254.187.233/tcp/36987/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn51z6uqUD8wmGQJ4NeSd
Swarm listening on
/ip4/10.228.207.210/tcp/36987/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn51z6uqUD8wmGQJ4NeSd
Swarm listening on
/ip4/10.228.207.40/tcp/36987/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn51z6uqUD8wmGQJ4NeSd
Replicating '/orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs'^CShutting down...
Saving database... Saved!
pi@lCloudy2:~/orbit-db-cli $  sudo node ./src/bin.js get
/orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs
"ERR: lCloudy1 peer connect"
"INFO: lCloudy2 peer is up"
"INFO: rCloudy3 peer is up"
"WARN: lCloudy3 disconnect"
"WARN: lCloudy2 service Kibana is down"
"ERR: rCloudy1 is down"
```

Desde el nodo donde se desplego la base de datos, lCloudy1 en este caso, se ejecuta la sincronización:

```
pi@lCloudy1:~/orbit-db-cli $ sudo node ./src/bin.js add
/orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs "INFO: lCloudy2 service Kibana
UP" --interactive --sync
Swarm listening on /ip4/127.0.0.1/tcp/41997/ipfs/QmZLYRj5uaPsuj7fudjSNGAzxMkEUTm7iZHx7Vnu56yKG5
Swarm listening on /ip4/192.168.15.10/tcp/41997/ipfs/QmZLYRj5uaPsuj7fudjSNGAzxMkEUTm7iZHx7Vnu56yKG5
Swarm listening on /ip4/172.18.0.1/tcp/41997/ipfs/QmZLYRj5uaPsuj7fudjSNGAzxMkEUTm7iZHx7Vnu56yKG5
Swarm listening on /ip4/169.254.164.151/tcp/41997/ipfs/QmZLYRj5uaPsuj7fudjSNGAzxMkEUTm7iZHx7Vnu56yKG5
Swarm listening on /ip4/169.254.10.200/tcp/41997/ipfs/QmZLYRj5uaPsuj7fudjSNGAzxMkEUTm7iZHx7Vnu56yKG5
Searching for peers for '/orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs'
Connected to peers:
1. QmWfBK9EeC7YyeH7CmN9QBXTjZn51z6uqUD8wmGQJ4NeSd
Press CTRL+C twice to exit the program
> ERR: rCloudy1 is down
Added QmPzWaUH3E4jCjBPX84WDeszpnkvproVNqwc9rgemfGGBr
> ^C
Shutting down...
Saving database... Saved!
```

El resultado es completamente satisfactorio. Por un lado, se puede ver como la base de datos “CloudyLogs” y las entradas creadas en lCloudy1 se replican a lCloudy2, y por otro, como al añadir una nueva entrada “ERR: rCloudy1 is down” (en azul) se replica inmediatamente en lCloudy2.

Una vez se lanza la replicación se puede actualizar de manera automática o manual, ejecutando una sesión interactiva.

```
pi@lCloudy2:~/orbit-db-cli $ sudo node ./src/bin.js replicate /orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMte
R3rqB/CloudyLogs --progress --dashboard
Swarm listening on /ip4/127.0.0.1/tcp/42415/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn5lz6uqUD8wmGQJ4NeSd
Swarm listening on /ip4/192.168.15.20/tcp/42415/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn5lz6uqUD8wmGQJ4NeSd
Swarm listening on /ip4/172.17.0.1/tcp/42415/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn5lz6uqUD8wmGQJ4NeSd
Swarm listening on /ip4/169.254.241.43/tcp/42415/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn5lz6uqUD8wmGQJ4NeSd
Swarm listening on /ip4/10.228.207.206/tcp/42415/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn5lz6uqUD8wmGQJ4NeSd
Swarm listening on /ip4/10.228.207.39/tcp/42415/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn5lz6uqUD8wmGQJ4NeSd
Swarm listening on /ip4/169.254.107.240/tcp/42415/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn5lz6uqUD8wmGQJ4NeSd
Swarm listening on /ip4/169.254.223.146/tcp/42415/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn5lz6uqUD8wmGQJ4NeSd
Swarm listening on /ip4/169.254.187.233/tcp/42415/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn5lz6uqUD8wmGQJ4NeSd
Swarm listening on /ip4/10.228.207.210/tcp/42415/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn5lz6uqUD8wmGQJ4NeSd
Swarm listening on /ip4/10.228.207.40/tcp/42415/ipfs/QmWfBK9EeC7YyeH7CmN9QBXTjZn5lz6uqUD8wmGQJ4NeSd
Loading '/orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs' (eventlog)
Loading '/orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs' 0/1 |   0.0% | 00:00:00
Loading '/orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs' 1/6 |  16.7% | 00:00:00
Loading '/orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs' 2/6 |  33.3% | 00:00:00
Loading '/orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs' 3/6 |  50.0% | 00:00:00
Loading '/orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs' 4/6 |  66.7% | 00:00:00
Loading '/orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs' 5/6 |  83.3% | 00:00:00
                                                                              6/6 | 100.0% | 00:00:00
Replicating '/orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs'
                                                                              6/6 | 100.0% | 00:00:00
```

Figura 85: Replicación BBDD OrbitDB tipo eventlog

```
pi@lCloudy1:~/orbit-db-cli $ sudo node ./src/bin.js add  /orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPay
rf5wyMteR3rqB/CloudyLogs "ERR: Docker is not running"  --sync
Swarm listening on /ip4/127.0.0.1/tcp/46289/ipfs/QmZLYRj5uaPsuj7fudjSNGAzxMkEUTm7iZHx7Vnu56yKG5
Swarm listening on /ip4/192.168.15.10/tcp/46289/ipfs/QmZLYRj5uaPsuj7fudjSNGAzxMkEUTm7iZHx7Vnu56yK
Swarm listening on /ip4/172.18.0.1/tcp/46289/ipfs/QmZLYRj5uaPsuj7fudjSNGAzxMkEUTm7iZHx7Vnu56yKG5
Swarm listening on /ip4/169.254.164.151/tcp/46289/ipfs/QmZLYRj5uaPsuj7fudjSNGAzxMkEUTm7iZHx7Vnu56
G5
Swarm listening on /ip4/169.254.10.200/tcp/46289/ipfs/QmZLYRj5uaPsuj7fudjSNGAzxMkEUTm7iZHx7Vnu56y
5
Searching for peers for '/orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs'
Connected to peers:
1. QmWfBK9EeC7YyeH7CmN9QBXTjZn5lz6uqUD8wmGQJ4NeSd
Added QmeGapghioimZfU3Ph3WCA2HarJ7rr2vo63G5WtmkkNmyo
pi@lCloudy1:~/orbit-db-cli $
```

```
Loading '/orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs' 5/6 |  83.3% | 00:00:00
                                                                              6/6 | 100.0% | 00:00:00
Replicating '/orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs'
Replicating '/orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs' 6/6 | 100.0% | 00:00:00
Replicating '/orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs' 6/7 |  85.7% | 00:01:18
                                                                              7/7 | 100.0% | 00:01:19
```

Figura 86: Actualización y sincronización con la replica

Test 21

Objetivo: Acceso y operaciones BBDD OrbitDB entre nodos IPFS.

Fuentes de variación: Permisos sistema operativo, roles IPFS.

Desarrollo: Una vez creada la base de datos, y realizada la replicación entre nodos, se deben poder realizar acciones desde cualquier otro nodo en local y que los cambios se sincronicen.

Resultado: Tras replicar la base de datos y comprobar que cualquier cambio es sincronizado en los otros nodos, se evalúa si al modificar la información en los nodos replicados se sincronizaría de igual manera, es decir, una base de datos descentralizada y distribuida completamente.

Se ve en la siguiente figura como se obtiene un error de permisos:

```
pi@lCloudy2:~/orbit-db-cli $ sudo node ./src/bin.js get /orbitdb/QmRZC47Tpl(
CloudyLogs --dashboard
"ERR: lCloudy1 peer connect"
"INFO: lCloudy2 peer is up"
"INFO: rCloudy3 peer is up"
"WARN: lCloudy3 disconnect"
"WARN: lCloudy2 service Kibana is down"
"\"ERR: rCloudy1 is down\""
"ERR: Docker is not running"
pi@lCloudy2:~/orbit-db-cli $ sudo node ./src/bin.js add /orbitdb/QmRZC47Tpl(
CloudyLogs "ERR: Serf is not installed"
Error: Not allowed to write
pi@lCloudy2:~/orbit-db-cli $
```

Figura 87: Get CloudyLogs en lCloudy2 e intento de añadir entrada

Se realiza una pequeña investigación, debido al poco margen disponible, y se llega al siguiente resultado.

```
pi@lCloudy2:~/orbit-db-cli $ sudo node ./src/bin.js info /orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB
/CloudyLogs --dashboard


/orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs
> Type: eventlog
> Owner: /orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs
> Data file: ./orbitdb/QmRZC47TplCAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs
> Entries: 7
> Oplog length: 7 / 7
> Write-access:
  04e4fcb2130b0a0a1987120fbc96c2c4a1ce1cd91af1b7c8f5aa6fe6810829de0143fa703f4c30c0819644197da0965391751 51adab67
  893638beefd29f2d52d

pi@lCloudy1:~/orbit-db-cli $ sudo node ./src/bin.js key
04e4fcb2130b0a0a1987120fbc96c2c4a1ce1cd91af1b7c8f5aa6fe6810829de0143fa703f4c30c0819644197da0965391
151adab6760893638beefd29f2d52d

pi@lCloudy2:~/orbit-db-cli $ sudo node ./src/bin.js key
482a0bccfdcaba1d039a3fcb1c99fdfa1a27a4e610705ae047ff5008164264a68d2af053c3165af9b633949b7f5015bdb7756050cd67628
aa5a04677ed7e9909
pi@lCloudy2:~/orbit-db-cli $
```

Figura 88: Key asociada a la base de datos; key en lCloudy2 (rojo) y lCloudy1 (azul)

Se obtiene la llave de la base de datos desplegada en lCloudy1 y replicada en lCloudy2, y revisando foros se identifica el problema. Se debe añadir la clave del *peer* para que pueda escribir en la base de datos ya que por defecto solo se dispone de acceso de lectura.

```
pi@lCloudy2:~/orbit-db-cli $ sudo node ./src/bin.js info
/orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs --dashboard
/orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs
> Type: eventlog
> Owner: /orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs
> Data file: ./orbitdb/QmRZC47Tp1CAJGzL5t4XPUJg3QHjPayS7rf5wyMteR3rqB/CloudyLogs
> Entries: 7
> Oplog length: 7 / 7
> Write-access:
>
04e4fcb2130b0a0a1987120fbc96c2c4a1ce1cd91af1b7c8f5aa6fe6810829de0143fa703f4c30c0819644197da0965
39175151adab6760893638beefd29f2d52d
pi@lCloudy2:~/orbit-db-cli $ sudo node ./src/bin.js key
0482a0bccfdcaba1d039a3fcb1c99fdfa1a27a4e610705ae047ff5008164264a68d2af053c3165af9b633949b7f5015
bdb7756050cd676285aa5a04677ed7e9909
```

Por lo tanto, se debería añadir la *key* en azúl (lCloudy2), a la base de datos CloudyLogs para que se pudiera escribir en la misma.

6.3 Disponibilidad

En este capítulo se evalúa la característica o el grado de disponibilidad en el entorno desplegado, tanto en los sistemas como en los servicios desplegados.

6.3.1 Cloudy

Como plataforma, Cloudy aporta diferentes servicios que junto con otras tecnologías, servicios y dispositivos permite ofrecer alta disponibilidad. Los test realizados a nivel de microCLoud se consideran suficientes, tanto en el entorno de microCloud local como regional, para obtener una serie de valoraciones al respecto.

Por otro lado, no se ha perseguido una alta disponibilidad real implementado algún orquestador de servicios tipo Docker Swarm, línea y puntos de acceso redundados, ect. Posibilidades que están presentes.

6.3.2 IPFS

Por defecto, los ficheros que se añaden con “ipfs add” son almacenados en la red IPFS, esta característica no aporta una ventaja frente al sistema cliente-servidor y HTTP, no obstante, cuando se carga un fichero e IPFS guarda en caché una copia ofreciendo dicha información a la red se obtiene el valor añadido

del sistema distribuido, ya que se consigue que esos datos sean publicados y accesibles desde cualquier nodo IPFS, al menos, durante veinticuatro horas en la red IPFS. Este límite temporal mínimo es debido a la interactividad con la información. Si no se interactúa con los datos publicados, existe una herramienta específica con el rol de recolector de archivos no utilizados encargada de eliminar estos ficheros automáticamente.

Test 22

Objetivo: Descarga y acceso a ficheros nodos *off-line.*

Fuentes de variación: No aplica.

Desarrollo: Tal y como se indica en la introducción de este subcapítulo, al subir un fichero a la red IPFS desde un nodo cualquiera, este fichero esta accesible, al menos, durante veinticuatro horas en la red. Indistintamente de si este nodo deja de pertenecer a la red IPFS.

Resultado: Un resultado similar a esta prueba ha sido mostrado con anterioridad en el test 15, no obstante, ha sido probado únicamente en los nodos de entorno local, a continuación, se muestra más detalle con resultados de capturas en nodo remoto y nodos locales.

```
poldo@rCloudy2:~$ Initializing daemon...
Successfully raised file descriptor limit to 2048.

poldo@rCloudy2:~$ Swarm listening on /ip4/10.1.24.136/tcp/4001
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/172.17.0.1/tcp/4001
Swarm listening on /ip6/::1/tcp/4001
Swarm listening on /ip6/fd00:1714:1714:9001:d0fd:2bff:fe0d:49e3/tcp/4001
Swarm listening on /p2p-circuit/ipfs/Qme9vooYqZfzbwvqStPSKYgmPQfywKDnkMUCfQTmzTWiDN
Swarm announcing /ip4/10.1.24.136/tcp/4001
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/172.17.0.1/tcp/4001
Swarm announcing /ip6/::1/tcp/4001
Swarm announcing /ip6/fd00:1714:1714:9001:d0fd:2bff:fe0d:49e3/tcp/4001
API server listening on /ip4/127.0.0.1/tcp/5001
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```

```
poldo@rCloudy2:~$ sudo ifconfig -a
[sudo] password for poldo:
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:f7:40:e4:4f  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens18: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.24.136  netmask 255.255.255.224  broadcast 10.1.24.159
        inet6 fd00:1714:1714:9001:d0fd:2bff:fe0d:49e3  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::d0fd:2bff:fe0d:49e3  prefixlen 64  scopeid 0x20<link>
        ether d2:fd:2b:0d:49:e3  txqueuelen 1000  (Ethernet)
        RX packets 474  bytes 61111 (59.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 402  bytes 56305 (54.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
poldo@rCloudy2:~/pruebas$ vi rCloudyTest2.html
poldo@rCloudy2:~/pruebas$ ipfs add -w rCloudyTest2.html
added QmVZZxq7Rcux4nb5eZQCD1njR6qDMPXZRH2bjFxYm1qFXa rCloudyTest2.html
added QmchDzDbr339PxAkW1es3kEQJV3q6WWVKyfiKNot9BiyRK
poldo@rCloudy2:~/pruebas$ more rCloudyTest2.html
<html>
<body>
<h1> TFG Clouds comunitarias e IPFS </h1>
<h2> TESTING IPFS: The distributed Web </h2>
<h3> Adding HTML file to the IPFS network from rCloudy2 </h3>

<h4> Accesing via lCloudy2, lCloudy3 </h4>
</body>
</html>
poldo@rCloudy2:~/pruebas$
```

Figura 89: Inicialización demonio y carga de fichero desde rCloudy 2.

En las figuras previas se puede observar cómo se inicializa el demonio IPFS en rCloudy2, se genera el fichero de prueba “rCloudyTest2.html”, y se sube a la red IPFS. Se realiza una visualización del contenido del fichero para comparar de manera gráfica.

En la siguiente captura se puede ver como lCloudy3, en un principio, no detecta a rCloudy2 debido a que no se ha añadido manualmente y/o no han interactuado aún. Se procede a realizar el acceso al fichero HTML y, a continuación, se puede ver como ya se detecta como *peer*.

```
leopoldo@lCloudy3:~$ ipfs swarm peers | grep 10.1.24
leopoldo@lCloudy3:~$ ipfs cat QmVZZxq7Rcux4nb5eZQCD1njR6qDMPXZRH2bjFxYm1qFXa



<html>
<body>
<h1> TFG Clouds comunitarias e IPFS </h1>
<h2> TESTING IPFS: The distributed Web </h2>
<h3> Adding HTML file to the IPFS network from rCloudy2 </h3>

<h4> Accesing via lCloudy2, lCloudy3 </h4>
</body>
</html>
leopoldo@lCloudy3:~$
leopoldo@lCloudy3:~$
leopoldo@lCloudy3:~$
leopoldo@lCloudy3:~$ ipfs swarm peers | grep 10.1.24
/ip4/10.1.24.136/tcp/4001/ipfs/Qme9vooYqZfzbwvqStPSKYgmPQfywKDnkMUCfQTmzTWiDN
```

Figura 90: Consulta peers y acceso a fichero desde lCloudy3

Únicamente se ha accedido al fichero y no se ha descargado. No obstante, se procede a parar el demonio en rCloudy2, nodo remoto, y observar desde lCloudy3 como desaparece de la lista de *peers*.

```
poldo@rCloudy2:~/pruebas$ jobs
[1]+  Running                 ipfs daemon &  (wd: ~)
poldo@rCloudy2:~/pruebas$ fg 1
ipfs daemon     (wd: ~)
^C
Received interrupt signal, shutting down...
(Hit ctrl-c again to force-shutdown the daemon.)
22:46:12.069 ERROR    connmgr: received disconnected not
r.ID cbJ2dM> swarm.go:386
22:46:12.069 ERROR     swarm2: failed to reset connectio
4001: use of closed network connection swarm_dial.go:273
```

```
</body>
</html>
leopoldo@lCloudy3:~$
leopoldo@lCloudy3:~$
leopoldo@lCloudy3:~$
leopoldo@lCloudy3:~$ ipfs swarm peers | grep 10.1.24
/ip4/10.1.24.136/tcp/4001/ipfs/Qme9vooYqZfzbwvqStPSKYgmPQfywKDnkMUCfQTmzTWiDN
leopoldo@lCloudy3:~$ ipfs swarm peers | grep 10.1.24
leopoldo@lCloudy3:~$
leopoldo@lCloudy3:~$
leopoldo@lCloudy3:~$
```

Figura 91: Consulta peers y parada de demonio IPFS rCloudy2

En las siguientes figuras se puede ver el acceso vía web desde lCloudy3, y el acceso y la descarga desde lCloudy2 una vez el nodo remoto ha sido desconectado de la red IPFS.

Figura 92: Acceso a fichero CLI y web desde lCloudy3

El nodo ya no está disponible como *peer* valido pero el fichero permanece en la red IPFS. Es accesible y se puede descargar desde otros nodos.

```
pi@lCloudy2:~ $ ipfs swarm peers | grep 10.1.24
pi@lCloudy2:~ $ ipfs cat QmVZZxq7Rcux4nb5eZQCDlnjR6qDMPXZRH2bjFxYmlqFXa
<html>
<body>
<h1> TFG Clouds comunitarias e IPFS </h1>
<h2> TESTING IPFS: The distributed Web </h2>
<h3> Adding HTML file to the IPFS network from rCloudy2 </h3>

<h4> Accesing via lCloudy2, lCloudy3 </h4>
</body>
</html>
pi@lCloudy2:~ $
```

```
pi@lCloudy2:~ $ ipfs cat QmVZZxq7Rcux4nb5eZQCDlnjR6qDMPXZRH2bjFxYmlqFXa
<html>
<body>
<h1> TFG Clouds comunitarias e IPFS </h1>
<h2> TESTING IPFS: The distributed Web </h2>
<h3> Adding HTML file to the IPFS network from rCloudy2 </h3>

<h4> Accesing via lCloudy2, lCloudy3 </h4>
</body>
</html>
pi@lCloudy2:~ $ ipfs cat QmVZZxq7Rcux4nb5eZQCDlnjR6qDMPXZRH2bjFxYmlqFXa > Test20.html
pi@lCloudy2:~ $ more Test20.html
<html>
<body>
<h1> TFG Clouds comunitarias e IPFS </h1>
<h2> TESTING IPFS: The distributed Web </h2>
<h3> Adding HTML file to the IPFS network from rCloudy2 </h3>

<h4> Accesing via lCloudy2, lCloudy3 </h4>
</body>
</html>
pi@lCloudy2:~ $
```

Figura 93: Acceso a fichero CLI y web, y descarga, desde lCloudy2

Test 23

Objetivo: Hacer permanentes los datos en la red IPFS.

Fuentes de variación: No aplica.

Desarrollo: Como se ha indicado previamente, después de veinticuatro horas en la red los ficheros pueden ser eliminados. Si no se tienen garantías de la interactuación con los ficheros, y se pretende evitar la eliminación, se debe utilizar el *pinning.* La solución para mantener los datos en la red es fijar la información, indicando al nodo y el colector que no se recoja dicha información. Para activar o eliminar esta permanencia, sería suficiente con ejecutar:

- ipfs pin add <hash>
- ipfs pin rm -r <hash>

Resultado: Se obtiene un resultado deseado, pudiendo acceder al fichero días más tarde en el entorno local y en la red global.

```
pi@1Cloudy2:~ $ ipfs pin add QmVZZxq7Rcux4nb5eZQCDlnjR6qDMPXZRH2bjFxYmlqFXa
pinned QmVZZxq7Rcux4nb5eZQCDlnjR6qDMPXZRH2bjFxYmlqFXa recursively
pi@1Cloudy2:~ $ ipfs pin rm -r QmVZZxq7Rcux4nb5eZQCDlnjR6qDMPXZRH2bjFxYmlqFXa
unpinned QmVZZxq7Rcux4nb5eZQCDlnjR6qDMPXZRH2bjFxYmlqFXa
pi@1Cloudy2:~ $
```

Figura 94: Activando y eliminando *pinning* de ficheros

6.4 Escalabilidad

En este capítulo se evalúa la característica o el grado de escalabilidad de los sistemas y/o servicios desplegados.

6.4.1 Cloudy y servicios

Test 24

Objetivo: Escalabilidad horizontal. Se añaden nuevos nodos Cloudy.

Fuentes de variación: Recursos.

Desarrollo: Se pretende evaluar la agilidad y sencillez con la que se puede escalar una microCloud basada en Cloudy, tanto a nivel de nodos como de recursos. Uno de los objetivos de Cloudy es eliminar la responsabilidad del usuario final de desplegar un entorno complejo con una parametrización específica de sistema. Tal y como se ha visto durante los capítulos de instalación y despliegue, se pueden añadir nodos mediante despliegue de la distribución Cloudy o utilizando una distribución soportada; Debian 9 (Stretch), Ubuntu 16.04 LTS, Raspbian, etc. y ejecutando el script de cloudynitzación.

Resultado: De las opciones disponibles; incorporar una nueva RPI, un pc doméstico o una máquina virtual, se ha optado por la última opción. Esta decisión se toma para evaluar y demostrar que realizando un clonado de una distribución ya existente se escala horizontalmente en menos de veinte minutos. Aproximadamente cinco minutos de clonado, según dimensiones de la máquina, y el resto de modificación de los parámetros del nodo anterior.

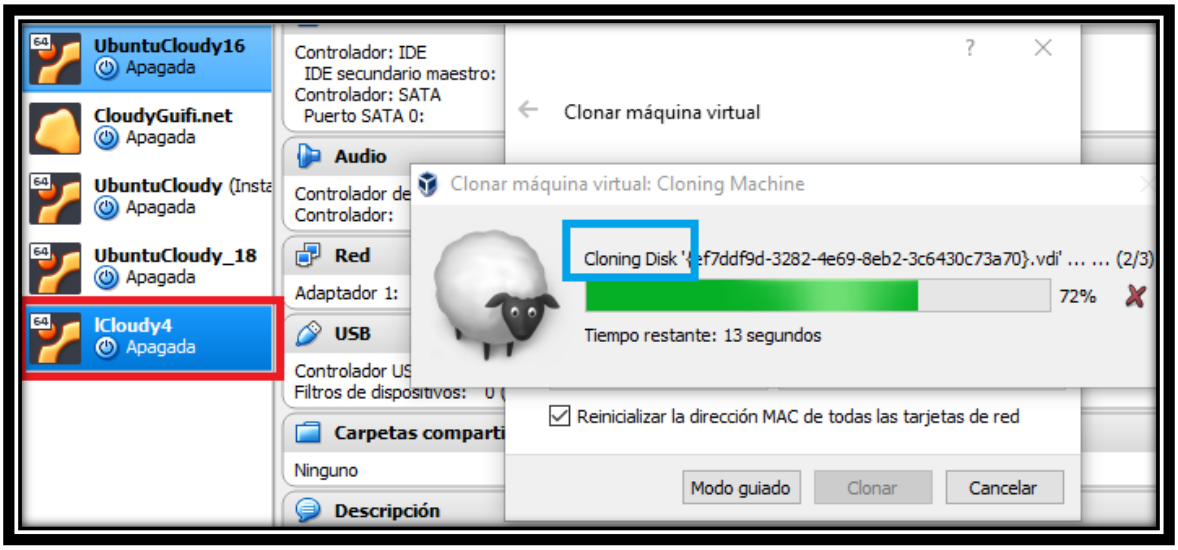


Figura 95: Clonando nodo virtual. Escalabilidad horizontal

Test 25

Objetivo: Escalabilidad vertical. Se añaden recursos a nodos existentes.

Fuentes de variación: Disponibilidad de recursos.

Desarrollo: Se pretende evaluar la agilidad y sencillez con la que se puede escalar una microCloud basada en Cloudy, tanto a nivel de nodos como de recursos. La escalabilidad vertical no siempre es posible. Se realiza una evaluación de escalabilidad vertical, conociendo que una de las ventajas de este entorno, y de la plataforma Cloudy, es que permite la escalabilidad horizontal.

Se debe tener en cuenta que los recursos pueden tener un límite físico o de soporte *hardware,* y quizás no se pueda escalar verticalmente. En un entorno heterogéneo, como el empleado para el proyecto, se dispone de máquinas virtuales, Raspberry Pi y PC domésticos. Se evalúa la escalabilidad en alguno de los componentes.

Resultado: A priori, con la virtualización y la abstracción de recursos, se podría escalar alguno de los nodos existentes otorgando más recursos a la máquina virtual, siempre que el servidor físico lo soporte.

Por otro lado, las RPI tienen una serie de recursos limitados, como pueden ser la memoria y la CPU, no obstante, y relacionado con las bases de datos e IPFS, se podría añadir almacenamiento, como se muestra en la siguiente figura.

```
pi@lCloudy2:~/orbit-db-cli $ df -kh
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        28G   12G   16G  43% /
devtmpfs        460M     0  460M   0% /dev
tmpfs           464M     0  464M   0% /dev/shm
tmpfs           464M  6.6M  458M   2% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           464M     0  464M   0% /sys/fs/cgroup
/dev/mmcblk0p1   43M   22M   21M  51% /boot
tmpfs            93M     0   93M   0% /run/user/1000
pi@lCloudy2:~/orbit-db-cli $
pi@lCloudy2:~/orbit-db-cli $
pi@lCloudy2:~/orbit-db-cli $ df -kh
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        28G   12G   16G  43% /
devtmpfs        460M     0  460M   0% /dev
tmpfs           464M     0  464M   0% /dev/shm
tmpfs           464M  6.6M  458M   2% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           464M     0  464M   0% /sys/fs/cgroup
/dev/mmcblk0p1   43M   22M   21M  51% /boot
/dev/sda1       299G   88G  211G  30% /media/pi/SAMSUNG
```

Figura 96: Agregando almacenamiento externo. Escalabilidad vertical

Se añade capacidad de procesamiento a lCloudy4.

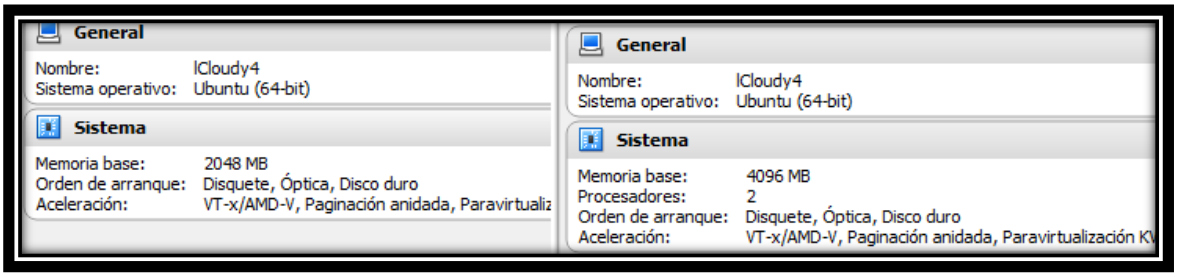


Figura 97: Agregando CPU y RAM lCloudy4

Test 26

Objetivo: Escalabilidad horizontal. Despliegue servicios.

Fuentes de variación: No aplica.

Desarrollo: Se pretende evaluar la agilidad y sencillez con la que se pueden desplegar servicios dentro de Cloudy, y por lo tanto escalar. Se realiza la activación de varios servicios activado únicamente en alguno de los nodos como pueden ser OwnCloud, Etherpad, etc. en diferentes nodos.

Resultado: Se replican los mismos servicios en varios nodos. Se comprueba que realizando el clonado de nodos virtuales con las imágenes de Docker

preinstaladas, la escalabilidad es prácticamente igual a la del test 24, en coste económico y temporal.

6.4.2 IPFS y Cluster IPFS

Test 27

Objetivo: Escalabilidad IPFS. Ampliación nodos en el clúster.

Fuentes de variación: Recursos hardware. Nodos disponibles.

Desarrollo: El clúster de IPFS admite añadir nodos o *peers* manualmente. Este test trata de evaluar el proceso de ampliación de un clúster IPFS. Para incluir un nuevo nodo, se debe tener en cuenta

- Inicializar y arrancar demonio IPFS
- Exportación de la llave del nodo con rol principal al nuevo *peer*
- Instalación del servicio de clúster IPFS
- Ejecución del servicio de clúster IPFS indicando la ruta de inicio el nodo principal

Resultado: El resultado es satisfactorio. Una vez se han realizado las acciones básicas indicadas en el desarrollo del test, el clúster IPFS se escala de manera ágil y sencilla.

Se añade un nuevo peer lCloudy3 desplegado, esta vez, sobre Ubuntu 16.04 LTS.

```
leopoldo@lCloudy3:~$ ipfs init
initializing IPFS node at /home/leopoldo/.ipfs
generating 2048-bit RSA keypair...done
peer identity: Qmaky1npRqpLimskM1aNRSQcy5kNbZRtiDY6sTn1ZRnjJG
to get started, enter:

        ipfs cat /ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv/readme

leopoldo@lCloudy3:~$
```

```
pi@lCloudy1:~/Downloads/ipfs-cluster-ctl $ ./ipfs-cluster-ctl peers ls
QmSTdySCHm7vCQEj1D1KSEHkLhRSQWyJTQJJdmxpRNS4Qv | lCloudy3 | Sees 2 other peers
  > Addresses:
    - /ip4/127.0.0.1/tcp/9096/ipfs/QmSTdySCHm7vCQEj1D1KSEHkLhRSQWyJTQJJdmxpRNS4Qv
    - /ip4/192.168.15.30/tcp/9096/ipfs/QmSTdySCHm7vCQEj1D1KSEHkLhRSQWyJTQJJdmxpRNS4Qv
    - /p2p-circuit/ipfs/QmSTdySCHm7vCQEj1D1KSEHkLhRSQWyJTQJJdmxpRNS4Qv
  > IPFS: QmakylnpRqpLimskM1aNRSQcy5kNbZRtiDY6sTn1ZRnjJG
    - /ip4/127.0.0.1/tcp/4001/ipfs/QmakylnpRqpLimskM1aNRSQcy5kNbZRtiDY6sTn1ZRnjJG
    - /ip4/192.168.15.30/tcp/4001/ipfs/QmakylnpRqpLimskM1aNRSQcy5kNbZRtiDY6sTn1ZRnjJG
    - /ip6/::1/tcp/4001/ipfs/QmakylnpRqpLimskM1aNRSQcy5kNbZRtiDY6sTn1ZRnjJG
```

Figura 98: Añadiendo y listando nuevos *peers*.

En la siguiente figura se pueden ver los 3 peers del clúster privado.

```
pi@1Cloudy1:~/Downloads/ipfs-cluster-ctl $ ./ipfs-cluster-ctl peers ls
QmSTdySCHm7vCQEj1D1KSEHkLhRSQWyJTQJJdmxpRNS4Qv | 1Cloudy3 | Sees 2 other peers
  > Addresses:
    - /ip4/127.0.0.1/tcp/9096/ipfs/QmSTdySCHm7vCQEj1D1KSEHkLhRSQWyJTQJJdmxpRNS4Qv
    - /ip4/192.168.15.30/tcp/9096/ipfs/QmSTdySCHm7vCQEj1D1KSEHkLhRSQWyJTQJJdmxpRNS4Qv
    - /p2p-circuit/ipfs/QmSTdySCHm7vCQEj1D1KSEHkLhRSQWyJTQJJdmxpRNS4Qv
  > IPFS: QmakylnpRqpLimskM1aNRSQcy5kNbZRtiDY6sTn1ZRnjJG
    - /ip4/127.0.0.1/tcp/4001/ipfs/QmakylnpRqpLimskM1aNRSQcy5kNbZRtiDY6sTn1ZRnjJG
    - /ip4/192.168.15.30/tcp/4001/ipfs/QmakylnpRqpLimskM1aNRSQcy5kNbZRtiDY6sTn1ZRnjJG
    - /ip6/::1/tcp/4001/ipfs/QmakylnpRqpLimskM1aNRSQcy5kNbZRtiDY6sTn1ZRnjJG
QmSwkaqawN2jnhkYcSrfBPiEMraavwspwCSQLjiqwh8PU9 | 1Cloudy2 | Sees 2 other peers
  > Addresses:
    - /ip4/127.0.0.1/tcp/9096/ipfs/QmSwkaqawN2jnhkYcSrfBPiEMraavwspwCSQLjiqwh8PU9
    - /ip4/172.17.0.1/tcp/9096/ipfs/QmSwkaqawN2jnhkYcSrfBPiEMraavwspwCSQLjiqwh8PU9
    - /ip4/192.168.15.20/tcp/9096/ipfs/QmSwkaqawN2jnhkYcSrfBPiEMraavwspwCSQLjiqwh8PU9
    - /p2p-circuit/ipfs/QmSwkaqawN2jnhkYcSrfBPiEMraavwspwCSQLjiqwh8PU9
  > IPFS: QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip4/127.0.0.1/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip4/172.17.0.1/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip4/192.168.15.20/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip4/93.156.73.143/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip6/::1/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
    - /ip6/fd51:42f8:caae:d92e::ff/tcp/4001/ipfs/QmWhTKWtCdFFvnVcdPvLBkjxPyeH7yJGdSoxeb4pEv7t94
QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M | 1Cloudy1 | Sees 2 other peers
  > Addresses:
    - /ip4/127.0.0.1/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
    - /ip4/169.254.148.105/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
    - /ip4/169.254.149.152/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
    - /ip4/172.17.0.1/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
    - /ip4/172.18.0.1/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
    - /ip4/192.168.15.10/tcp/9096/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
    - /p2p-circuit/ipfs/QmViC61LsYVNkZhtnCwQFn7LhxE54ea2VTTFSEAfrinv8M
  > IPFS: QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/127.0.0.1/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/169.254.148.105/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/169.254.149.152/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/172.17.0.1/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/172.18.0.1/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/192.168.15.10/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip4/93.156.73.143/tcp/61643/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip6/::1/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
    - /ip6/fd51:42f8:caae:d92e::ff/tcp/4001/ipfs/QmQvPGSGxNk89ct5WhCLZLk8bPufpPmZvkZ76EvkqQzivY
```

Figura 99: Resultado 3 *peers* dentro del clúster IPFS.

Test 28

Objetivo: Escalabilidad IPFS. Eliminación nodo en el clúster privado.

Fuentes de variación: No aplica.

Desarrollo: Por diferentes motivos se puede querer eliminar un nodo de un *swarm* privado. En caso de querer eliminar un nodo del clúster de IPFS, es suficiente con una única acción manual. Se debe ejecutar la siguiente sentencia:

- Ipfs-cluster-ctl peers rm <peer-id>

Resultado: Se ejecuta la sentencia indicada y se obtiene el resultado perseguido. Automáticamente el *peer* deja de formar parte del clúster.

```
pi@lCloudy1:~/Downloads/ipfs-cluster-service $ ../ipfs-cluster-ctl/ipfs-cluster-ctl peers ls | grep -i
Cloudy
QmNaqYLDC5qmWmKEx1zVV92hVCEz6rVHgYnhUsf7o7i6Tv | lCloudy2 | Sees 2 other peers
QmRVMDMczHqpzW1u4neAqzCXQ5gP1kh15k24xvyNCG5TZ1 | lCloudy3 | Sees 2 other peers
QmSVJZQbQwgqUtPqGxMyB1DWF8ypcNteUUuaKm3a9ef96h | lCloudy1 | Sees 2 other peers
pi@lCloudy1:~/Downloads/ipfs-cluster-service $ w
 11:43:00 up 16:25,  3 users,  load average: 0.49, 0.49, 0.37
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
pi       tty1     -                Thu19   16:25m  0.43s  0.34s -bash
pi       tty7     :0               Thu19   16:25m 28.61s  0.46s /usr/bin/lxsession -s LXDE-pi -e LXDE
pi       pts/0    192.168.15.11    10:26    4.00s  5:09   0.03s w
pi@lCloudy1:~/Downloads/ipfs-cluster-service $ 11:43:03.266 WARNI    cluster: Peer <peer.ID Qm*9ef96h>
received alert for ping in <peer.ID Qm*CG5TZ1> cluster.go:281
11:43:03.266 WARNI    cluster: no state has been agreed upon yet cluster.go:339
11:43:03.267 WARNI    cluster: Peer <peer.ID Qm*9ef96h> received alert for freespace in <peer.ID Qm*CG5
TZ1> cluster.go:281
11:43:09.083  INFO  consensus: peer added to Raft: QmVjWK6QHJDv1REygU6zB8KKDVTXxN3gpnjePN35yvQhSN cons
nsus.go:355
11:43:13.280  INFO    cluster: Peer added  QmVjWK6QHJDv1REygU6zB8KKDVTXxN3gpnjePN35yvQhSN cluster.go:6
2
11:43:18.266 WARNI    cluster: Peer <peer.ID Qm*9ef96h> received alert for freespace in <peer.ID Qm*CG5
TZ1> cluster.go:281
11:43:18.266 WARNI    cluster: Peer <peer.ID Qm*9ef96h> received alert for ping in <peer.ID Qm*CG5TZ1>
cluster.go:281
11:43:18.266 WARNI    cluster: no state has been agreed upon yet cluster.go:339

pi@lCloudy1:~/Downloads/ipfs-cluster-service $ ../ipfs-cluster-ctl/ipfs-cluster-ctl peers ls | grep -i
Cloudy
QmNaqYLDC5qmWmKEx1zVV92hVCEz6rVHgYnhUsf7o7i6Tv | lCloudy2 | Sees 3 other peers
QmRVMDMczHqpzW1u4neAqzCXQ5gP1kh15k24xvyNCG5TZ1 | lCloudy3 | Sees 3 other peers
QmSVJZQbQwgqUtPqGxMyB1DWF8ypcNteUUuaKm3a9ef96h | lCloudy1 | Sees 3 other peers
QmVjWK6QHJDv1REygU6zB8KKDVTXxN3gpnjePN35yvQhSN | lCloudy4 | Sees 3 other peers
```

Figura 100: Resultado 4 *peers* dentro del clúster IPFS.

Se elimina uno de los peers, lCloudy3.

```
pi@lCloudy1:~/Downloads/ipfs-cluster-service $ 11:45:43.986  INFO  consensus: peer removed from Raft: Q
mRVMDMczHqpzW1u4neAqzCXQ5gP1kh15k24xvyNCG5TZ1 consensus.go:382

pi@lCloudy1:~/Downloads/ipfs-cluster-service $ ../ipfs-cluster-ctl/ipfs-cluster-ctl peers ls | grep -i
Cloudy
QmNaqYLDC5qmWmKEx1zVV92hVCEz6rVHgYnhUsf7o7i6Tv | lCloudy2 | Sees 2 other peers
QmSVJZQbQwgqUtPqGxMyB1DWF8ypcNteUUuaKm3a9ef96h | lCloudy1 | Sees 2 other peers
QmVjWK6QHJDv1REygU6zB8KKDVTXxN3gpnjePN35yvQhSN | lCloudy4 | Sees 2 other peers
pi@lCloudy1:~/Downloads/ipfs-cluster-service $ w
 11:45:52 up 16:28,  3 users,  load average: 0.23, 0.46, 0.39
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
pi       tty1     -                Thu19   16:28m  0.43s  0.34s -bash
pi       tty7     :0               Thu19   16:28m 28.69s  0.46s /usr/bin/lxsession -s LXDE-pi -e LXDE
pi       pts/0    192.168.15.11    10:26    0.00s  5:47   0.02s w
```

Figura 101: Resultado 3 *peers* dentro del clúster IPFS.

Si no se elimina de esta forma el peer seguirá cacheado en el clúster y si no está disponible o no tiene conectividad, genera errores.

Tests 29

Objetivo: Limites escalabilidad IPFS Cluster.

Fuentes de variación: No aplica.

Desarrollo: Existen unas limitaciones de escalabilidad indicadas en el capítulo correspondiente de este proyecto. Se ha probado con un número máximo de 5 *peers* por clúster y con un almacenamiento de 300GB aproximados. Tal y como se ha visto en los tests de escalabilidad, se podría aumentar este almacenamiento de manera sencilla.

Resultado: Se realiza la escalabilidad a 5 nodos locales formando un clúster privado de IPFS. Tal y como se puede ver en la figura se ejecuta el comando para visualización de peers en el clúster.

```
127.0.0.1       lcloudy4
127.0.0.1       localhost
127.0.1.1       lcloudy4
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
lCLoydy1 192.168.15.10
lCloudy2 192.168.15.20
lCloudy3 192.168.15.30
#lCloudy4 192.168.15.40
lCloudy5 192.168.15.50
rCloudy1 192.168.1.200
rCloudy2 10.1.24.136
rCloudy3 10.139.40.57
```

```
pi@lCloudy1:~/Downloads/ipfs-cluster-ctl $ ./ipfs-cluster-ctl peers ls | grep -i cloudy
QmNaqYLDC5qmWmKExlzVV92hVCEz6rVHgYnhUsf7o7i6Tv | lCloudy2 | Sees 4 other peers
QmRVMDMczHqpzWlu4neAqzCXQ5gPlkhl5k24xvyNCG5TZl | lCloudy3 | Sees 4 other peers
QmSVJZQbQwgqUtPqGxMyBlDWF8ypcNteUUuaKm3a9ef96h | lCloudy1 | Sees 4 other peers
QmUvRkCVSMPo4lLD7XuG5zhYgUrcwsvH6Woim7GjCbp3je | lCloudy5 | Sees 4 other peers
QmVjWK6QHJDvlREygU6zB8KKDVTXxN3gpnjePN35yvQhSN | lCloudy4 | Sees 4 other peers
```

Figura 102: Limites escalabilidad IPFS Cluster

Test 30

Objetivo: Escalabilidad IPFS. Nodos rCloudyX y lCloudyX

Fuentes de variación: No aplica.

Desarrollo: Existen unas limitaciones de escalabilidad, como se ha visto en el test anterior. Con el fin de superar dichas limitaciones se intenta formar un clúster

con todos los nodos disponibles; lCLoudy1, lCloudy2, lCloudy3, lCloudy4, lCloudy5, rCloudy1, rCloudy2, rCloudy3, un total de ocho nodos. Tal y como se ha visto en los tests de escalabilidad vertical/horizontal, se podría aumentar este almacenamiento de manera sencilla.

Resultado: Se realiza el test, pero se encuentran una serie de errores los cuales imposibilitan realizar el escalado completo .Se llega a realizar un primer análisis y se documenta a continuación.

Se detecta que mediante la VPN basada en OpenVPN parece fallar el tráfico de vuelta, es decir, los nodos conectados remotamente por VPN no alcanzan el clúster IPFS correctamente. Parece ser un problema de NAT más que de routing o de tráfico de vuelta en la VPN, no obstante, en el canal de IPFS Cluster de Protocol Labs nos indican que puede ser por el tipo de VPN y el túnel establecido:

postablesel viernes pasado a las 20:01

leolvarezh: so just to confirm your nodes on the same LAN can connect to each other? How are you configuring your connection to the offsite node? I'm personally using an IPSec+IKEv2 tunnel

---

29 de diciembre de 2018

leoalvarezhel sábado pasado a las 11:04

Hi!! the VPN is from Openvpn. I think SSL VPN. I'm going to check... Did you check with any LAN-to-LAN software like Hamachi but Opensource?(editado)

---

postablesel sábado pasado a las 11:38

Ah, that could be part of the issue. Honestly, I would just use IPSec+IKEv2, it's designed for these kinds of solutions and is like the go-to for site-to-site VPNs

---

postablesel sábado pasado a las 11:51

SSL VPNs and OpenVPN are great say, if you're trying to route your traffic for anonymity purposes or you're trying to remotely access some computers, but for this kind of stuff you REALLY want to use IPSec, it's incredibly secure, very very customizable

as for checking, im not quite sure I understand? I did basic tests like running telnet, trying to ssh between my nodes, pinning data in on-location to see if it successfully was picked up by my nodes in another location

another issue to with using OpenVPN/SSL VPNs can be ports not being allowed through properly. You may also have a sort of "one way" VPN in which in-bound access is allowed, but outbound access initiated from your nodes is not

Oh right one thing I haven't mentioend. the consensus.raft configuration section of service.json will probably need to be modified depending on network latency and stuff. settings to look at: wait_for_leader_timeout network_timeout heartbeat_timeout election_timeout Another thing to ensure is that you properly bootstrapped the node

---

31 de diciembre de 2018

leoalvarezhel lunes pasado a las 14:32

Sorry for delay! I'm reading now

---

postablesel lunes pasado a las 17:24

Np!

---

1 de enero de 2019

leoalvarezhel martes pasado a las 10:26

Hi @postables !Yes, regarding that, I think the issue to with using OpenVPN/SSL VPNs can be ports not being allowed through properly. It's like one way communication but I though that was a routing or NAT problem

BTW, happy new year mates

---

2 de enero de 2019

leoalvarezhayer a las 12:57

I've installed OpenSwan to check over IPSec

I'm triying to configure

---

postablesayer a las 20:06

Happy new year as well! For your IPSec VPN make sure you have matching tunnel settings on both ends

If you don't have matching configs your tunnel may work but it'll be very unstable

Se modifican los parámetros indicados de consensus.raft dentro de service.json:

```
"consensus": {
    "raft": {
      "init_peerset": [],
      "wait_for_leader_timeout": "25s",
      "network_timeout": "15s",
      "commit_retries": 1,
      "commit_retry_delay": "200ms",
```

```
    "backups_rotate": 6,
    "heartbeat_timeout": "3s",
    "election_timeout": "3s",
    "commit_timeout": "50ms",
    "max_append_entries": 64,
    "trailing_logs": 10240,
    "snapshot_interval": "2m0s",
    "snapshot_threshold": 8192,
    "leader_lease_timeout": "500ms"
  }
```

Debido al fallo de comunicación bidireccional del túnel parece apuntar a la VPN y el tráfico de vuelta, el error sigue dando de timeout pero con errores previos de dial backoff del clúster.

```
11:40:59.328 ERROR  p2p-gorpc: dial backoff call.go:63
11:40:59.328 ERROR    cluster: dial backoff cluster.go:656
11:40:59.328 ERROR    service: bootstrap to
/ip4/192.168.15.10/tcp/9096/ipfs/QmQmsuHpzzrZwFkDRFFrf921aGYdb6FBNHDmMQqYrmo
KSk failed: dial backoff daemon.go:156
```

Se modifica el Bootstrap apuntando a la IP origen que se levanta con la conexión VPN en el nodo principal del clúster, y se consigue realizar, por fin, el objetivo.

Se construye un clúster privado de 7 nodos, siendo 2 de ellos remotos. rCloudy2 y rCloudy3.

Por lo tanto, se consigue superar el límite de 5 nodos en un clúster en un entorno heterogéneo de VPN y LAN. Cabe reseñar que se han de modificar ligeramente los parámetros anteriormente indicados en el “consensus” dentro del service.json en los nodos conectados por VPN.

```
pi@lCloudy1:~/Downloads/ipfs-cluster-ctl $ ./ipfs-cluster-ctl peers ls | grep -i Cloudy
QmNaqYLDC5qmWmKExlzVV92hVCEz6rVHgYnhUsf7o7i6Tv | lCloudy2 | Sees 6 other peers
QmQmsuHpzzrZwFkDRFFrf92laGYdb6FBNHDmMQqYrmoKSk | lCloudy1 | Sees 6 other peers
QmRVMDMczHqpzWlu4neAqzCXQ5gPlkhl5k24xvyNCG5TZl | lCloudy3 | Sees 6 other peers
QmUerFBt2cTR8X2s7BNzSXAnYiY9rNmuEnPJuA4GFEtzYr | rCloudy3 | Sees -1 other peers
QmUvRkCVSMPo4lLD7XuG5zhYgUrcwsvH6Woim7GjCbp3je | lCloudy5 | Sees 6 other peers
QmVF73UyEpfChglcESC4wRoLSxKCQ7NqoRCYwQU34VAYPz | rCloudy2 | Sees 6 other peers
QmVjWK6QHJDvlREygU6zB8KKDVTXxN3gpnjePN35yvQhSN | lCloudy4 | Sees 6 other peers
pi@lCloudy1:~/Downloads/ipfs-cluster-ctl $ 15:33:08.404  INFO    cluster: Peer added  QmUerFBt2c
zSXAnYiY9rNmuEnPJuA4GFEtzYr cluster.go:602

pi@lCloudy1:~/Downloads/ipfs-cluster-ctl $ ./ipfs-cluster-ctl peers ls | grep -i Cloudy
QmNaqYLDC5qmWmKExlzVV92hVCEz6rVHgYnhUsf7o7i6Tv | lCloudy2 | Sees 6 other peers
QmQmsuHpzzrZwFkDRFFrf92laGYdb6FBNHDmMQqYrmoKSk | lCloudy1 | Sees 6 other peers
QmRVMDMczHqpzWlu4neAqzCXQ5gPlkhl5k24xvyNCG5TZl | lCloudy3 | Sees 6 other peers
QmUerFBt2cTR8X2s7BNzSXAnYiY9rNmuEnPJuA4GFEtzYr | rCloudy3 | Sees 6 other peers
QmUvRkCVSMPo4lLD7XuG5zhYgUrcwsvH6Woim7GjCbp3je | lCloudy5 | Sees 6 other peers
QmVF73UyEpfChglcESC4wRoLSxKCQ7NqoRCYwQU34VAYPz | rCloudy2 | Sees 6 other peers
QmVjWK6QHJDvlREygU6zB8KKDVTXxN3gpnjePN35yvQhSN | lCloudy4 | Sees 6 other peers
pi@lCloudy1:~/Downloads/ipfs-cluster-ctl $ ./ipfs-cluster-ctl peers ls | grep -i Cloudy | wc -l
7
```

Figura 103: Clúster de 7 nodos (remotos y locales).

Se adjunta figura con el resultado, y aunque se instala Openswan (también se ha revisado la opción de StrongSwan), no ha dado tiempo a configurar ese tipo de VPN IPSEC por realizar una comparativa.

Test pendiente 1

Objetivo: Escalabilidad IPFS. Arquitectura clúster de clústeres.

Fuentes de variación: No aplica.

Desarrollo: Se puede realizar una red IPFS privada formada por diferentes clústeres. Es decir, se puede realizar un *swarm* privado de *peers* que a su vez tengan el rol principal dentro de clústeres privados.

Resultado: No se llega a realizar dicho test por falta de tiempo, pero se adjunta detalle del proyecto IPFS-Cluster con el esquema perseguido en la prueba.

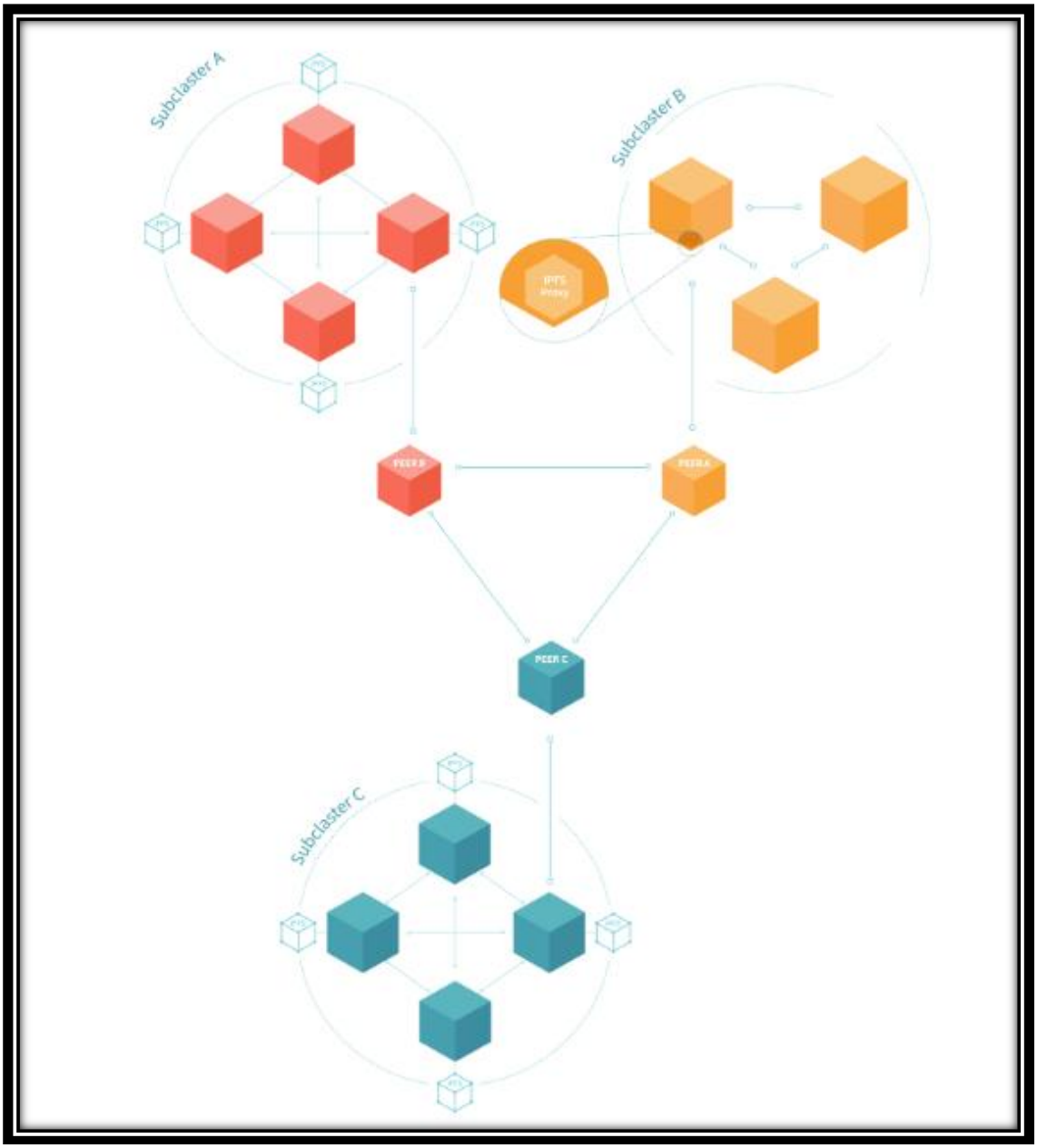


Figura 104: Clúster de clústeres y *peers*.

6.5 Seguridad

Este capítulo está dedicado a la seguridad dentro del entorno, haciendo foco en mayor medida dentro del marco de IPFS.

Por un lado, se han identificado una serie de aspectos a controlar desde Cloudy, y por otro, puntos que comentar sobre IPFS. No se han realizado tests específicos, pero se deja constancia de diferentes situaciones analizadas.

6.6.1 Cloudy

Cloudy dispone de la opción de acceso y funcionamiento sobre SSL. Se puede activar únicamente https en el siguiente menú del portal.

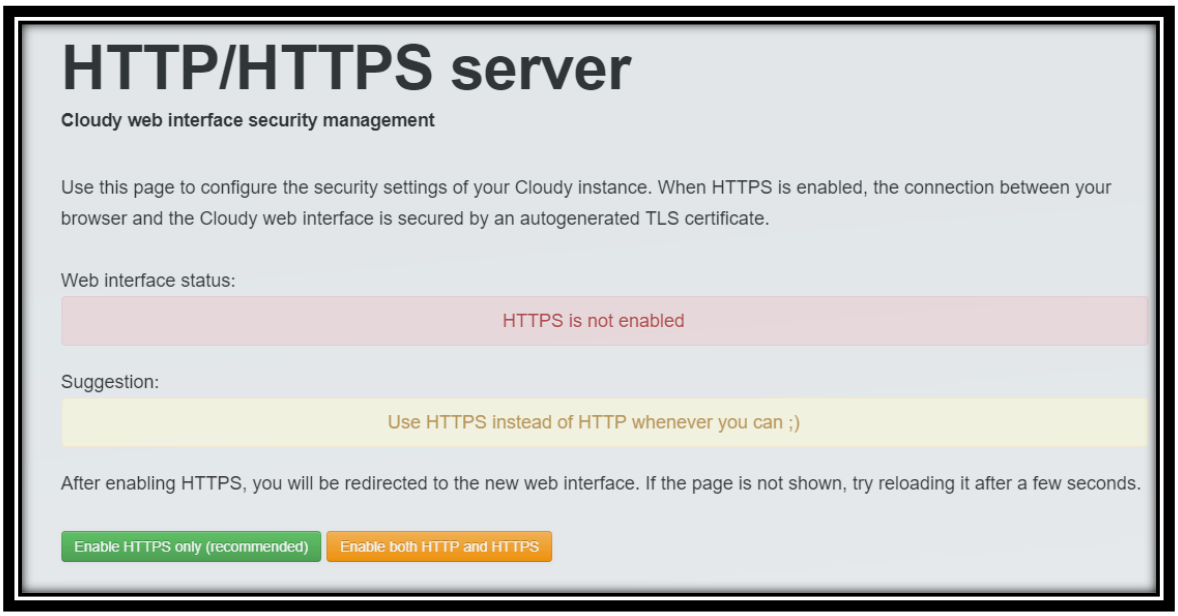


Figura 105: Activación HTTPS

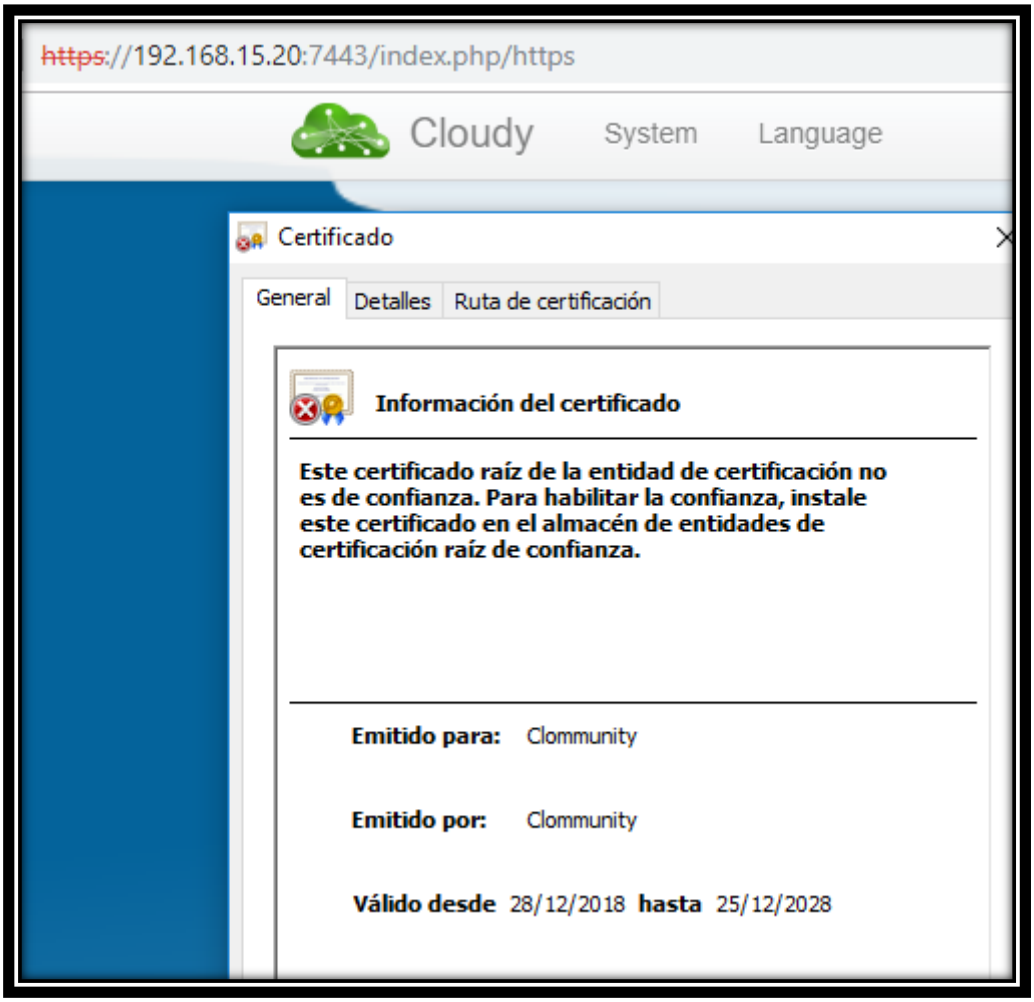


Figura 106: Certificado interno Clommunity. Acceso https puerto 7443

Al tratarse de entornos colaborativos o de más de un nodo es recomendable el acceso por https.

Por otro lado, se ha detectado que cualquier usuario físico del sistema puede acceder a Cloudy:

User Poldo

auth.log:Dec 28 15:12:06 lCloudy2 sshd[14145]: Accepted password for poldo from 127.0.0.1 port 56804 ssh2

auth.log:Dec 28 15:12:06 lCloudy2 sshd[14145]: pam_unix(sshd:session): session opened for user poldo by (uid=0)

auth.log:Dec 28 15:12:07 lCloudy2 systemd-logind[293]: New session c5 of user poldo.

auth.log:Dec 28 15:12:07 lCloudy2 systemd: pam_unix(systemd-user:session): session opened for user poldo by (uid=0)

auth.log:Dec 28 15:12:08 lCloudy2 sshd[14145]: pam_unix(sshd:session): session closed for user poldo

auth.log:Dec 28 15:12:08 lCloudy2 systemd: pam_unix(systemd-user:session): session closed for user poldo

User Pi

```
Dec 28 15:14:45 lCloudy2 sshd[14804]: Accepted password for pi from 127.0.0.1 port 57360 ssh2

Dec 28 15:14:45 lCloudy2 sshd[14804]: pam_unix(sshd:session): session opened for user pi by (uid=0)

Dec 28 15:14:45 lCloudy2 systemd-logind[293]: New session c7 of user pi.

Dec 28 15:14:45 lCloudy2 sshd[14817]: Received disconnect from 127.0.0.1 port 57360:11: PECL/ssh2
(http://pecl.php.net/packages/ssh2)

Dec 28 15:14:45 lCloudy2 sshd[14817]: Disconnected from 127.0.0.1 port 57360

Dec 28 15:14:45 lCloudy2 sshd[14804]: pam_unix(sshd:session): session closed for user pi
```

Y una vez dentro de Cloudy publicar servicios, detener servicios, etc. aunque no hayan sido gestionados inicialmente por dicho usuario.

6.5.2 IPFS e IPFS Cluster

En este subcapítulo se indican algunas consideraciones de seguridad al ejecutar el Clúster de IPFS.

Hay que tener en cuenta que al exponer un nodo tanto a un clúster como a la red IPFS, si no está protegido y alguien toma el control, puede obtener el “poder” sobre clúster o realizar acciones ilegales en red IPFS.

Los tipos de nodos en un clúster IPFS se dividen en cuatro:

Cluster swarm endpoint: Los *peers* del clúster de IPFS se comunican entre sí utilizando flujos cifrados con libp2p. Estas transmisiones están protegidas de forma predeterminada por la *secret* compartida del clúster secreto de clúster compartido. El *endpoint* está controlado por la clave de configuración cluster.listen_multiaddress, por defecto es /ip4/0.0.0.0/tcp/9096, y representa la dirección de escucha para establecer la comunicación con otros nodos a través de llamadas remotas RPC y protocolo de consenso (consensus).

Únicamente los nodos con el *secret* compartido podrán comunicarse, el resto serán bloqueados.

APIs

- **HTTP API:** Puede estar expuesto. Es posible habilitar SSL y configurar la autenticación básica.
- Se puede configurar IPFS-cluster para una autenticación básica mediante la siguiente configuración:

```
ipfs-cluster-ctl –basic-auth <username:password>
```

- Si se activa esta característica debemos tener en cuenta que solo están soportadas solicitudes con “–https”.
- **API libp2p-HTTP:** Cuando se usa un host alternativo libp2p, para la api, la libp2p_listen_multiaddress puede estar expuesta, es necesario habilitar la autenticación básica.
- **API de IPFS:** tcp: 5001 es la API del demonio de IPFS y no debe exponerse a otro que no sea localhost.

IPFS endpoint: tcp: 9095 no debe exponerse sin un mecanismo de autenticación. De manera predeterminada, no proporciona autenticación ni cifrado (similar a tcp de IPFS: 5001).

Los miembros del Clúster de IPFS se comunican con el demonio de a través de HTTP simple. Se utiliza la API HTTP de IPFS de forma predeterminada en /ip4/127.0.0.1/tcp/9095. Por otro lado, los nodos también proporcionan un punto final de HTTP IPFS Proxy no autenticado, controlado por la opción ipfshttp.proxy_listen_multiaddress que por defecto es /ip4/127.0.0.1/tcp/9095.

El acceso a cualquiera de estos dos puntos finales puede implicar el control del demonio clúster de IPFS y del demonio en cierta medida.

Resumiendo, la exposición de un punto final desprotegido puede dar a cualquier usuario el control del clúster.

7. Valoración de la solución obtenida

Una vez finalizadas las instalaciones, despliegues y evaluación del entorno se emiten una serie de valoraciones.

Por un lado, se subdivide este capítulo en la valoración económica, y por otro, la valoración técnica con una serie de comentarios y recomendaciones acerca de los resultados obtenidos.

7.1 Valoración económica

A continuación, se incluye una estimación económica del entorno desplegado, cabe destacar que no se ha tenido en cuenta las aportaciones de máquinas virtuales.

Los precios de venta al público se han obtenido de Amazon.com Inc.

Dispositivo	Nombre	P.V.P.
Raspberry PI 3B	lCloudy1	37,99 €
Raspberry PI 3B+	lCloudy2 / rCloudy1	38,99 €
Cisco EPC CMR	Router	350 €
Cajas RPI 3B	N.A.	12,99 €

Si se tiene en cuenta que en los *Smart Home* prácticamente todo el mundo dispone de electrónica de red, se podría eliminar la entrada correspondiente al *router*, resultando un entorno muy económico.

Por lo tanto, una de las ventajas de diseñar y desplegar este tipo de entorno para microCloud es el coste mínimo que supone.

7.2 Resultados y recomendaciones

Los resultados obtenidos se consideran, en líneas generales, satisfactorios.

Desde el capítulo de estado del arte hasta el final del capítulo de evaluación se han ido generando una serie evidencias, tanto de la componente teórica como de la práctica, que se detallan a continuación.

Respecto a la componente teórica del proyecto, se ha realizado un acercamiento al *software* y *hardware* existente para el diseño de una microCloud. Se ha optado finalmente por Cloudy, aunque se han investigado otras opciones incluidas en la bibliografía como EdgeX Foundry y FogFlow.

Se ha realizado un acercamiento a la implementación de servicios distribuidos en alta disponibilidad, estudiando diferentes herramientas y protocolos.

Se ha investigado acerca del proyecto Orbit, y las bases de datos distribuidas y descentralizadas; OrbitDB.

Además, se ha estudiado IPFS, su integración con los sistemas actuales y comentado diferentes casos de uso. Por otro lado, se han identificado diferentes servicios sobre IPFS como IPFS-Log e IPFS-Cluster.

Respecto a la componente práctica del proyecto, se ha desplegado un nodo basado en Cloudy sobre Raspberry PI, y se ha ido evolucionando hacia un entorno de microCloud. El entorno se ha diseñado con nodos *hardware* diversos, no obstante, la flexibilidad que nos dan tanto Cloudy como Docker, y en general el OpenSource, ha facilitado un entorno heterogéneo, funcional, y por un coste estimado de 120 euros.

Finalmente, se ha desplegado un entorno de microCloud de 4 máquinas virtuales; rCloudy2, rCloudy3, lCloudy4, lCloudy5 y 4 máquinas físicas: rCloudy1, lCloudy1, lCloudy2 y lCloudy3. Con este despliegue se ha evaluado la escalabilidad y la flexibilidad de la microCloud, desplegando Cloudy sobre diferentes distribuciones GNU/Linux basadas en Debian. Sin contar las diferentes arquitecturas hardware utilizadas; x86, amd64 y ARM. Cabe destacar el soporte de diversas aplicaciones, entre ellas Cloudy, Docker, IPFS, Orbit, etc.

El uso de contenedores virtuales con Docker genera una capa de abstracción y virtualización que, aunque puede parecer más complejo en un principio, permite un uso más “limpio” del entorno y parametrizaciones más agiles de los servicios.

Por último, se ha ido un paso más allá y se ha desplegado en el entorno IPFS, realizando diferentes pruebas de operación y servicios asociados a IPFS:

- OrbitDB
- IPFS-Cluster

Se ha trabajado con la red IPFS y se han desplegado tanto un clúster privado IPFS como diferentes bases de datos Orbit. Se considera interesante el uso de una base de datos descentralizada como es OrbitDB para Cloudy. Además, se puede utilizar un almacenamiento distribuido privado basado en IPFS Cluster.

En el despliegue hacia la microCloud regional se han encontrado una serie de dificultades, por un lado, problemas en la publicación de los servicios entre los diferentes nodos remotos, y por otro, los problemas de comunicación de IPFS-Cluster a través de la VPN.

Los problemas en la publicación vía Serf se han encontrado en la última fase del proyecto, al igual que en el caso de IPFS Cluster. Se ha detectado como algún servicio dejaba de estar publicado y no permitía publicarse de nuevo o instancias Serf que se detectaban desde unos nodos sí, y otros no, al activar IPFS. En caso

de disponer de más tiempo se buscaría una solución tipo *LAN-to-LAN* similar al *software* Hamachi (FreeWare), con el fin de evitar NAT y posibles tráficos de vuelta bloqueados. Se han analizado soluciones OpenSource de VPN IPSec tipo Strongswan u OpenSwan pero no se ha tenido tiempo para produnfizar.

Respecto a la seguridad, se han identificado una serie de recomendaciones como pueden ser la securización del acceso a Cloudy:

- Integración de certificado válido en https y no autofirmado.
- Parametrización del acceso a Cloudy. Actualmente se autentica mediante el módulo PAM, y cualquier usuario del sistema puede logarse y ejecutar acciones; para contenedores de otro usuario, quitar la publicación, detener servicios tipo IPFS o Serf, etc.
- Centralización de logs específicos fuera del auth.log, daemon.log y messages.

En IPFS Cluster, se ha visto que con disponer de la llave incluida en el fichero “services.json” se puede pasar a formar parte del clúster privado, por lo tanto, se recomienda la parametrización de https y autenticación básica.

8. Conclusiones

Por concluir, se considera que en las diferentes fases del proyecto se han obtenido una serie de resultados positivos logrando los objetivos de este Trabajo Fin de Grado. En una primera fase, se ha confirmado la viabilidad de despliegue de un sistema completamente abierto, tanto a nivel físico como lógico. Demostrando la adaptabilidad del software utilizado sobre un entorno accesible, a prácticamente cualquier hogar, basado en diferentes arquitecturas.

En fases posteriores se ha desplegado un entorno con mayor número de nodos, mayor poder computacional y aumento del almacenamiento. Confirmando así la escalabilidad del sistema, y como las Raspberry PI lo soportan y cumplen las expectativas a nivel de carga.

En el aspecto de la escalabilidad se ha visto mayor potencial en el escalado horizontal, creciendo en número de nodos multiplataforma; amd64, x86, ARM, etc. proporcionando agilidad y facilidad de despliegue de servicios distribuidos o microservicios en este tipo de entornos.

Otro de los aspectos evaluados ha sido la disponibilidad. En este punto se ha demostrado que el aumento de la disponibilidad de los servicios mediante este tipo de soluciones es viable, no obstante, para conseguir una alta disponibilidad real se deben implementar otros componentes como pueden ser; balanceadores de carga, *routers*, líneas de acceso de *backup*, y aplicaciones de orquestación de servicios tipo Docker Swarm. De todas maneras, se ha conseguido desplegar servicios en un total de 5 nodos locales y 3 remotos, con la restricción de que el usuario, al no tener otros mecanismos implementados, debe dirigirse a los servicios desde Cloudy o directamente desde alguno de los servidores donde estén implementados.

Como comentario general, se ha de decir que el hecho de implementar microservicios, y soluciones basadas en protocolos en fase alfa o con poca madurez ha sido un reto. Diferentes componentes de IPFS, Orbit, OrbitDB u otros servicios como IPFS Cluster han sido parte fundamental del desarrollo de este proyecto.

Se considera que existe potencial en IPFS, y la integración en Cloudy de servicios como IPFS Cluster o IPFS Log, sin olvidar la aportación de OrbitDB como base de datos distribuida y CRDT. No obstante, con el objetivo de disponer de estos servicios IPFS operativos se ha dejado en un segundo plano la seguridad, únicamente dedicando un subcapítulo a alto nivel. Referente a la seguridad y modificaciones, se debe tener en cuenta que OrbitDB es un software en fase alfa en desarrollo activo.

Respecto a la planificación, reseñar que se ha seguido sin grandes desviaciones. Se ha echado en falta un mejor dimensionado de la tarea de elaboración de la memoria que, sumado a los fallos encontrados en red con Orbit y Serf, han propiciado ese pequeño desvío. No obstante, la agilidad para desplegar los nodos ha permitido realizar la entrega final acorde en fecha.

8.1 Siguientes pasos

Se han identificado una serie de aspectos que podrían tenerse en cuenta de cara acciones futuras.

Por un lado, se prevé que IPFS sea el reemplazo de HTTP, teniendo en cuenta que el primero, al ser una tecnología distribuida, no tiene un único punto de fallo.

Al ser un sistema de archivos distribuido de igual a igual (P2P) permitiría la descentralización de Internet, logrando la distribución de la información en un número mayor de actores. Los ataques DDoS podrían ser irrelevantes ya que estos ocurren en sistema centralizados, no obstante, uno de los siguientes pasos sería profundizar en la seguridad de IPFS. Se han visto mecanismos de autenticación básica, nombres de ficheros codificados mediante hash, pero no cifrados, etc.

Por otro lado, se ha evaluado subjetivamente como IPFS permite mayores velocidades de conexión, ya que los usuarios cercanos comparten información en lugar de tener que solicitarla individualmente a servidores centralizados remotos. No se ha evaluado objetivamente con herramientas de rendimiento tipo IPERF o TCPKali [52].

Sin perder de vista IPFS, se ha visto que está diseñado para utilizarse de numerosas maneras y en diferentes áreas. Algunos de los casos de uso que se han evaluado y se podría desarrollar e integrar en Cloudy son:

- Sistema de archivos global, usando /ipfs y /ipns.
- Sistema de actualización automática de versiones, publicaciones y copias de seguridad, tanto privado como público.
- Sistema de *file-sharing* codificado con posibilidad de cifrado.
- Administrator de versiones de paquete para cualquier tipo de software.
- Como un CDN cifrado.

Por falta de tiempo no sé ha podido evaluar más a fondo OrbitDB e IPFS Log, no obstante, se considera interesante para el uso con Cloudy. Tampoco se ha evaluado, y quedaría pendiente, la utilización de PubSub y FUSE dentro de IPFS.

Por último, siguiendo el hilo de seguridad, se han identificado una serie de aspectos en la autenticación y autorización en Cloudy que podrían ser estudiados. Quizás la autenticación podría seguir realizándose con el módulo PAM y la autorización; rol y privilegios del usuario, delegarse desde una base de datos integrada en Cloudy, etc.

9. Glosario

ARM: ARM es el acrónimo de Advanced RISC Machine. Es una arquitectura RISC (Reduced Instruction Set Computer u ordenador de conjunto reducido de instrucciones) de 32 y 64bits (v8) desarrollada por ARM Holdings.

API: Acrónimo de Aplicación Programming Interface. Es un conjunto de rutinas que se proporcionan para proveer acceso a funciones determinadas de una plataforma o *software*.

Avahi: Protocolo libre que permite a las aplicaciones publicar y descubrir nodos y servicios asociados dentro de una red local.

CID: Acrónimo de Content Identifier. CID es un formato para hacer referencia al contenido en sistemas de información distribuidos, como IPFS.

CLI: Es la interfaz de línea de comandos o Command Line. Método que permite dar instrucciones a algún sistema operativo o aplicación por medio de expresiones de texto.

Clúster: Conjunto de dos o más nodos que pueden compartir recursos de computación, almacenamiento o servicios. Los nodos de un clúster se monitorización entre sí mediante un ***heartbeat***.

CN: Community Network. Son entornos que promueven la participación ciudadana en asuntos comunes o con un objetivo e interés común.

CRDT: Es el acrónimo de Conflict-free Replicated Data Type. Es una estructura de datos que se puede replicar en múltiples nodos en una red, donde las réplicas se pueden actualizar de forma independiente y concurrente, sin coordinación entre las réplicas, y donde siempre es matemáticamente posible resolver las inconsistencias que pueden resultar.

Docker: Proyecto de código abierto que se basa en contenedores virtuales de GNU/Linux automatizando el despliegue de aplicaciones bajo esos contenedores.

Docker HUB: Servicio de Docker donde se registran imágenes predefinidas de contenedores en la nube. Permite compartir con el resto de la comunidad imágenes según tipo de aplicación, arquitectura del nodo, etc.

Docker Compose: una herramienta para crear y administrar aplicaciones de múltiples contenedores donde todos los contenedores están definidos y vinculados en un solo archivo, girando todos los contenedores en un solo comando.

Docker Swarm: una herramienta de agrupación en clúster nativa que agrupa los hosts de Docker y programa contenedores utilizando estrategias de programación, también convierte un grupo de máquinas host en un único host virtual.

EDT: Acrónimo de Estructura de Descomposición del Trabajo (EDT). Consiste en la descomposición jerárquica y detallada del trabajo que va a ser realizado dentro de un proyecto.

Firmware: Software mínimo que necesita un dispositivo para interactuar a nivel lógico con él. Establece la lógica de bajo nivel que controla la electrónica de cualquier dispositivo.

FW: Acrónimo de Firewall. Hardware o software utilizado como elemento de seguridad activa dentro de una red o sistema.

HASH: Es un texto resultado de una función hash. Las funciones hash criptográficas son aquellas que cifran una entrada y actúan de forma parecida a las funciones hash, ya que comprimen la entrada a una salida de menor longitud y son fáciles de calcular.

Heartbeat: Servicio que proporciona funcionalidad de infraestructura de clúster; comunicación y pertenencia, a sus nodos.

HTTP: Hypertext Transfer Protocol. Protocolo de la capa de aplicación de la pila OSI. Famoso a nivel TCP/IP por ser un protocolo esencial en el desarrollo de Internet.

IoT: Internet of Things. Internet de las cosas es un término que hace referencia a la interconexión de objetos cotidianos con los sistemas de información o Internet.

IoE: Internet of Everything. El Internet de todo (IoE) es un concepto que extiende el énfasis de la internet de las cosas (IoT), en las comunicaciones de máquina a máquina (M2M). Tiene el objetivo de describir un sistema más complejo que también abarca personas y procesos además de los dispositivos.

IPNS: El Sistema de nombres interplanetarios (IPNS) es un sistema para crear y actualizar enlaces mutables al contenido de IPFS. Dado que los objetos en IPFS están direccionados por contenido, su dirección cambia cada vez que lo hace su contenido.

IPLD: Es el modelo de datos direccionable por contenido. Permite tratar todas las estructuras de datos vinculadas a hash como subconjuntos de un espacio de información unificado.

LAN: Acrónimo de Local Area Network. Termino muy utilizado en *networking* para referirse a la red de área local. Red de ordenadores de extensión local o reducida; una casa, oficina, instituto, etc.

M2M: Machine-to-Machine. Termino que hace referencia al intercambio de información o la comunicación entre dos dispositivos remotos.

microCloud: Infraestructura tipo Cloud mucho más ligera y reducida donde se ofrecen servicios más acotados a un grupo reducido de usuarios.

NAT: Network Address Translation. Técnica aplicada en routers y firewalls para intercambiar paquetes entre dos redes que asignan mutuamente direcciones solapadas o incompatibles entre sí. Consiste en convertir, en tiempo real, las direcciones utilizadas en los paquetes enrutados.

Node.js: Entorno de código abierto y multiplataforma basado en el lenguaje de programación ECMAScript.

PubSub: Sistema de búsqueda que notifica la noticias e información nueva que concuerda con la búsqueda.

TCP: Transmission Control Protocol. Protocolo orientado a conexión perteneciente al nivel cuatro de la capa OSI. Es uno de los protocolos fundamentales de Internet.

UDP: Junto con TCP otro protocolo del nivel de transporte. UDP o User Datagram Protocol es un protocolo no orientado a conexión. Permite el intercambio de información en la red mediante el envío de datagramas sin que se haya establecido una conexión previa.

URL: Uniform Resource Location. Es el término que describe la dirección a unos recursos que pueden ser variables en el tiempo. Esta dirección es la manera de localizar dichos recursos.

10. Bibliografía

- Gustavo A.A. Santana. CCNA Cloud CLDFND 210-451 Official Cert Guide. Cisco Press, Indianapolis, 2016

- Chad Hintz, Cesar Obediente, Ozden Karakok. CCNA Data center DCICN 200-150 Official Cert Guide. Cisco Press, Indianapolis, 2017.

- Harry Newton. Newtons Telecom dictionary. 22nd edition. Group West, Berkeley 2006.

- [1] Cisco Systems Inc, San Jose, CA. (s.f.). Cloud Computing - Data Center Strategy, Architecture, and Solutions.
  Recuperado 2 de octubre de 2018, de
  https://www.cisco.com/c/en/us/solutions/data-center-virtualization/index.html

- [2] Western Sydney University (s.f.). An overview of Cloud Computing.
  Recuperado 8 de octubre de 2018, de
  https://www.westernsydney.edu.au/tld/home/how_to/how-to_resources/internet_resources/the_cloud_-_overview

- [3] IBM. (s.f.). IaaS, PaaS, SaaS, Modelos de servicio cloud.
  Recuperado 8 de octubre de 2018, de
  https://www.ibm.com/cloud-computing/es-es/learn-more/iaas-paas-saas/

- [4] Technologypcthree (6 de julio de 2018). What is cloud application
  Extraído 9 de octubre de 2018, de
  http://technologypcthree.com/2018/07/06/what-is-cloud-application/

- [5] OnCloud (s.f.) Estándares de proveedores en la nube
  Recuperado 9 de octubre de 2018, de
  https://on-cloud.mx/blog/innovacion-5/post/estandares-de-proveedores-en-la-nube-64

- [6] Jhon Jairo Padilla, Javier Pinzón Castellanos. (6 de abril de 2015). Estándares para Cloud Computing: Estado del arte y análisis de protocolos para varias nubes.
  Recuperado 29 de diciembre de 2018, de
  https://revistas.upb.edu.co/index.php/puente/article/viewFile/7107/6498

- [7] Welcome to Raspbian (s.f.). Raspbian.
  Recuperado 29 de diciembre de 2018, de
  https://www.raspbian.org/

- [8] Openstack (s.f.). Build the future of Open infrastructure.
  Recuperado 29 de diciembre de 2018, de
  https://www.openstack.org/

- [9] Docker (s.f.). Docker Opensource.
  Recuperado 29 de diciembre de 2018, de
  https://www.docker.com/community/open-source

- [10] OpendayLight (s.f.). OpenDayLight opensource project.
  Recuperado 29 de diciembre de 2018, de
  https://www.opendaylight.org/

- [11] IPFS (s.f.) IPFS is the Distributed Web.
  Recuperado 9 de octubre de 2018, de
  https://ipfs.io/

- [12] HashiCorp Serf (s.f.). Decentralized Cluster Membership, Failure and Orchestration.
  Recuperado 9 de octubre de 2018, de
  https://www.serf.io/

- [13] Internet of Things (2018). Connected devices installed base worldwide from 2015 to 2025.
  Recuperado 9 de octubre de 2018, de
  https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

- [14] Cloudy (s.f.). Cloudy GNU/LINUX distribution.
  Recuperado 29 de diciembre de 2018, de
  https://cloudy.community/

- [15] Edge X Foundry (s.f.). The Open Platform for the IoT Edge
  Recuperado 29 de diciembre de 2018, de
  https://www.edgexfoundry.org/

- [16] Fog Flow (s.f.). IoT Edge computing framework
  Recuperado 29 de diciembre de 2018, de
  https://fogflow.readthedocs.io/en/latest/

- [17] Raspberry PI. (14 de marzo de 2018).
  Recuperado 11 de octubre de 2018, de
  https://www.raspberrypi.org/blog/raspberry-pi-3-model-bplus-sale-now-35/

- [18] Amin M.Khan, Félix Freitag. (diciembre 2017). Research gate. Layered Architecture for Cloudy platform.
  Recuperado 11 de octubre de 2018, de
  https://www.researchgate.net/figure/Layered-architecture-for-Cloudy-platform-with-Docker_fig1_322111645

- [19] Clommunity (2014). A community networking Cloud in a box.
  Recuperado 11 de octubre de 2018, de
  http://clommunity-project.eu/

- [20] Cloudy (s.f.). What is Cloudy.
  Recuperado 29 de diciembre de 2018, de
  https://cloudy.community/es/what-is-cloudy/

- [21] Sudhir Khatwani. Coinsutra. (Septiembre 2018). A look at the top decentralized storage.
  Recuperado 2 de diciembre de 2018, de
  https://coinsutra.com/decentralized-storage-network-dsn/

- [22] Protocol Labs (s.f.).
  Recuperado 2 de diciembre de 2018, de
  https://protocol.ai/

- [23] Catching the Blockchain train journal. (10 de agosto de 2017). Understanding the IPFS White Paper part1.
  Recuperado 2 de diciembre de 2018, de
  https://decentralized.blog/understanding-the-ipfs-white-paper-part-1.html

- [24] Juan Benet. (s.f.). IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3)
  Recuperado 2 de diciembre de 2018, de
  https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf

- [25] Anne-Marie Kermarrec. (11 de marzo de 2014). Routing in DHT.
  Recuperado 2 de diciembre de 2018, de
  https://www.youtube.com/watch?v=WqQRQz_XYg4

- [26] Anne-Marie Kermarrec. (11 de marzo de 2014). Dynamic Distributed Hash tables.
  Recuperado 2 de diciembre de 2018, de
  https://www.youtube.com/watch?v=p8iugvHeGcg

- [27] Alejandro Pardo Tapia, Diego Martinez Campos (s.f.). Protocolo BitTorrent.
  Recuperado 2 de diciembre de 2018, de
  http://profesores.elo.utfsm.cl/~agv/elo322/1s09/project/reports/proyecto_bittorrent_elo322.pdf

- [28] Jonh Hoffman (febrero 2008). Multitracker Metadata Extension
  Recuperado 2 de diciembre de 2018, de
  http://www.bittornado.com/docs/multitracker-spec.txt
  http://bittorrent.org/beps/bep_0012.html

- [29] Gitbooks. (s.f.). Tutorial: Merkle Trees and the IPFS DAG
  Recuperado 4 de diciembre de 2018, de
  https://flyingzumwalt.gitbooks.io/decentralized-web-primer/content/ipfs-dag/

- [30] libfuse.(s.f.).FUSE. Filesystem in Userspace interface
  Recuperado 4 de diciembre de 2018, de
  https://github.com/libfuse/libfuse

- [31] Juan Benet, extracted by Mark Pors (septiembre 2017). The IPFS white Paper: IPFS Design
  Recuperado 4 de diciembre de 2018, de
  https://hackernoon.com/understanding-the-ipfs-white-paper-part-2-df40511addbd

- [32] zk Capital. (septiembre 2018). IPFS: A complete analysis of The Distributed Web
  Recuperado 2 de diciembre de 2018, de
  https://medium.com/zkcapital/ipfs-the-distributed-web-e21a5496d32d

- [33] Protocol Labs. IPFS.Pinbot-irc (s.f.). Bot for the IPFS IRC channel.
  Recuperado 6 de diciembre de 2018, de
  https://github.com/ipfs/pinbot-irc

- [34] Protocol Labs. IPFS Cluster (s.f.). IPFS Cluster Overview.
  Recuperado 6 de diciembre de 2018, de
  https://cluster.ipfs.io/documentation/overview/

- [35] Shubheksha. (octubre 2017). Understanting the Raft consensus algorithm.
  Recuperado 6 de diciembre de 2018, de
  https://medium.freecodecamp.org/in-search-of-an-understandable-consensus-algorithm-a-summary-4bc294c97e0d

- [36] Append-only log CRDT on IPFS (s.f.). IPFS Log
  Recuperado 6 de diciembre de 2018, de
  https://github.com/orbitdb/ipfs-log

- [37] Cloudy (s.f.). IPFS como servicio en Cloudy.
  Recuperado 6 de diciembre de 2018, de
  https://cloudy.community/es/ipfs-as-support-service-in-cloudy/

- [38] Proyecto Orbit (s.f.). Orbit
  Recuperado 26 de diciembre de 2018, de
  https://github.com/orbitdb/orbit

- [39] Cryptocurrency Service Center Company Limited (octubre 2016). eChat.
  Recuperado 28 de diciembre de 2018, de
  https://investors.echat.io/static/doc-pdf/es/e-Chat_Whitepaper.pdf

- [40] Mackenzie, James. (enero 2017). Headless Raspberry Pi setup
  Recuperado 6 de diciembre de 2018, de
  https://hackernoon.com/raspberry-pi-headless-install-462ccabd75d0

- [41] Clommunity GitHub (s.f.) Cloudynitzar
  Recuperado 6 de diciembre de 2018, de
  https://github.com/Clommunity/cloudynitzar/blob/master/cloudynitzar.sh

- [42] NPM.js (s.f.).
  Recuperado 6 de diciembre de 2018, de
  https://www.npmjs.com/get-npm

- [43] ipfs.io (s.f.). Install IPFS
  Recuperado 6 de diciembre de 2018, de
  https://docs.ipfs.io/introduction/install/

- [44] IPFS (s.f.) Documentación IPFS
  Recuperado 26 de diciembre de 2018, de
  https://docs.ipfs.io/

- [45] Go IPFS (s.f.). Proyecto GoIPFS.
  https://github.com/ipfs/go-ipfs

- [46] IPFS Cluster (s.f.). Guia de inicio IPFS Cluster.
  Recuperado 26 de diciembre de 2018, de
  https://cluster.ipfs.io/guides/quickstart/],

- [47] Docker HUB (s.f.). Listados contenedores IPFS Cluster.
  Recuperado 6 de diciembre de 2018, de
  https://hub.docker.com/r/ipfs/ipfs-cluster/

- [48] Docker HUB (s.f.). Listados contenedores ELK.
  Recuperado 6 de diciembre de 2018, de
  https://hub.docker.com/r/ind3x/rpi-elasticsearch/

- [49] OrbitDB (s.f.). Código proyecto OrbitDB
  Recuperado 6 de diciembre de 2018, de
  https://github.com/orbitdb/orbit-db

- [50] Guifi.net (s.f.). Instalación Cloudy
  Recuperado 6 de diciembre de 2018, de
  http://es.wiki.guifi.net/wiki/Cloudy_en_nodo_cliente

- [51] Bensound. (s.f.) Ukulele Royalty free music.
  Recuperado 11 de diciembre de 2018.
  https://www.bensound.com

- [52] TCPKali (s.f) Fast multi-core TCP WebSockets load generator.
  Recuperado 5 de diciembre de 2018, de
  https://github.com/satori-com/tcpkali

- [53] Cisco (2014). Fog Computing and the IoT. Extend Cloud to Where the things are.
  Recuperado 5 de diciembre de 2018, de
  http://cdn.iotwf.com/resources/72/IoT_Reference_Model_04_June_2014.pdf

- [54] Git. Installing SW using Git.
  Recuperado 12 de octubre de 2018, de
  https://www.lifewire.com/installing-software-using-git-3993572

- [55] DiskImage32 (s.f). Utilización DiskImage32
  Recuperado 12 de octubre de 2018, de
  https://sourceforge.net/projects/win32diskimager/files/

- [56] Moya, P. (febrero de 2016). Raspberry PI 3.
  Recuperado 12 de octubre de 2018, de
  https://omicrono.elespanol.com/2016/02/raspberry-pi-3-model-b/

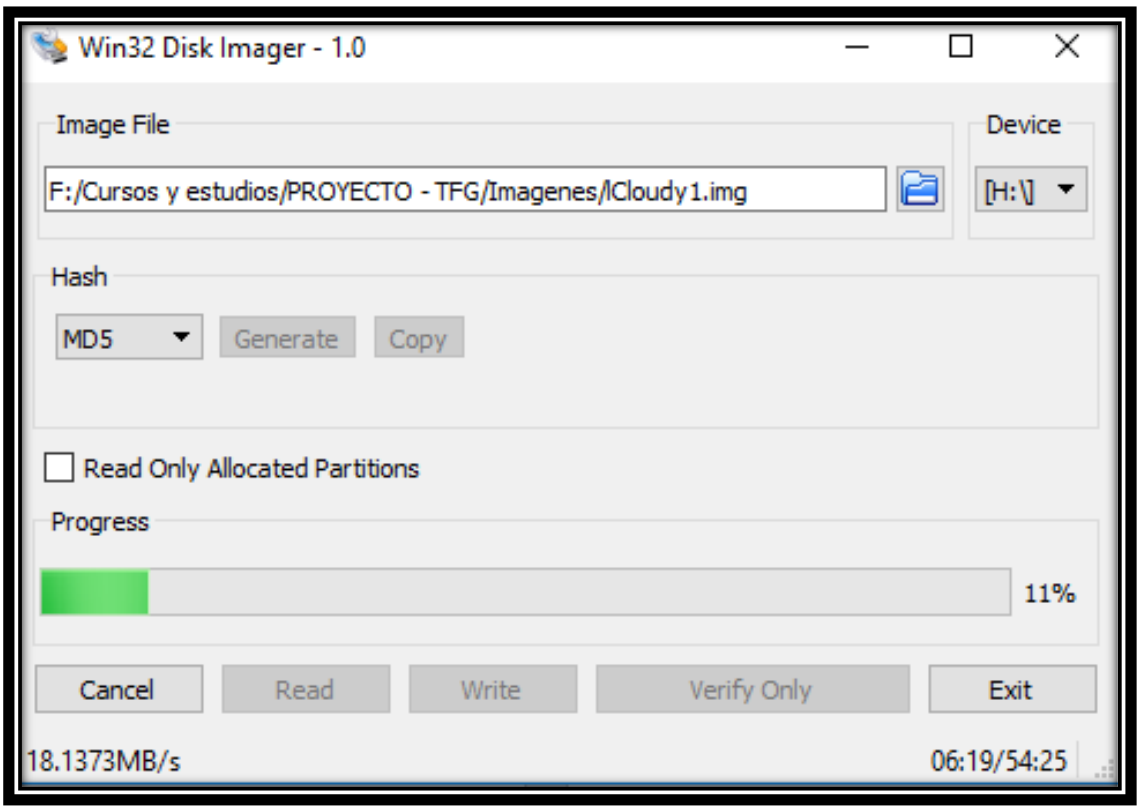
- [57] Shivam Raj. Beebom (octubre 2016). How to clone RPI SD Card. Recuperado 12 de octubre de 2018, de https://omicrono.elespanol.com/2016/02/raspberry-pi-3-model-b/

11. Anexos

A continuación, se incluyen una serie de anexos que se consideran relevantes para el despliegue del entorno y ejecución de proyecto.

11.1 Backup del entorno

Una vez se haya llegado a este punto, donde se dispone en nuestras RPI de Raspbian, Cloudy, OpenVPN y Docker, además de la configuración local, sería recomendable realizar una copia o clon de nuestros entornos. Esto se puede realizar desde nuestro cliente Windows 10 con el software Win32 Disk Image, gratuito y licenciado bajo GPL-2 o bien con una serie de comandos desde entorno GNU/Linux, como puede ser “dd”. La siguiente información se encuentra detallada en Beebom [57].



Si se utiliza un entorno virtual, se podrían realizar clones o también *snapshots* como se puede apreciar en la siguiente figura:

11.2 Configuración servicio IPFS automático en el arranque del SO

Si se quiere dejar el servicio de IPFS funcionando constantemente en el nodo se debe configurar el demonio para arranque automático dentro del systemd. Se crea el fichero ipfs.service.

```
[Unit]
Description= IPFS Daemon

[Service]
ExecStart= /usr/local/bin/ipfs daemon
Restart= on-failure

[Install]
WantedBy= default.target
```

Se añade al systemd:

```
sudo cp ipfs.service /et
```

Y se configura como servicio bajo el systemctl:

```
sudo systemctl –user start ipfs.service
sudo systemctl –user enable ipfs.service
```

11.3 Guía comandos útiles Docker y arranque automático contenedores

Listado de contenedores ejecutando algún proceso:

```
sudo docker ps
```

```
leopoldo@lCloudy3:~/Descargas/ipfs-cluster-service$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
 STATUS              PORTS                NAMES
fc7bd5613032        mysql:latest        "docker-entrypoint.s…"   2 minutes ago
 Up 2 minutes        3306/tcp, 33060/tcp   CloudyTFG
```

Listado de mapeo de puertos de un contenedor

sudo docker port <ContainerID>

Listado de procesos ejecutándose en un contenedor

sudo docker top <ContainerID>

Información detallada de un contenedor en ejecución

sudo docker inspect <ContainerID>

```
leopoldo@lCloudy3:~/Descargas/ipfs-cluster-service$ sudo docker top fc7bd5613032
UID                 PID                 PPID                C                   STIME
    TIME                CMD
vboxadd             12995               12978               0                   15:16
    00:00:01            mysqld
leopoldo@lCloudy3:~/Descargas/ipfs-cluster-service$ sudo docker inspect fc7bd5613032
[
    {
        "Id": "fc7bd5613032efed6093ae80bb50e2c6c595f3ab18d59f5854ac604098032ffe",
        "Created": "2018-12-29T14:16:44.900510908Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "mysqld"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
```

Parar y arrancar un contenedor

sudo docker stop <ContainerID> sudo docker start <ContainerID>

Borrar un contenedor

sudo docker rm <ContainerID>

Listado de imágenes descargadas en local

sudo docker images

Descarga de una imagen

sudo docker pull elasticsearch:2.1.0 (ejemplo de imagen definida)

Borrado de imagen

sudo docker rmi <NombreImagen y versión> sudo docker rmi ubuntu14.04/utils:0.0.1 (ejemplo)