DIRECCIÓN DE ÁREA TERRITORIAL MADRID–SUR
CONSEJERÍA DE EDUCACIÓN, UNIVERSIDADES,
CIENCIA Y PORTAVOCÍA
Comunidad de Madrid

INFRAESTRUCTURA CLOUD ESCALABLE PARA LA AUTOMATIZACIÓN DE SERVICIOS EN CONTENEDORES

Realizado por:
Asier García Cerdeiras
Andrés Sierra Sánchez

Tutor:
Miguel Ángel González Martín

INSTUTUTO POLITÉCNICO CALDERÓN DE LA BARCA
PINTO. MADRID

Ciclo Formativo de Grado Superior
Técnico Superior en Administración de Sistemas Informáticos en Red

Proyecto de Administración de Sistemas Informáticos en Red

Marzo, 2025

Contenido

. Título y Autor

1.1. Título y Autor

- **Título del Proyecto:** Infraestructura Cloud Escalable para la Automatización de Servicios en Contenedores
- **Autores:**
    - Asier García Cerdeiras
    - Andrés Sierra Sánchez
- **Ciclo Formativo:** Grado Superior de Administración de Sistemas en Red (ASIR)
- **Centro Educativo:** IES Calderón de la Barca, Pinto (Madrid)

1.2. Resumen

El proyecto consiste en el diseño e implementación de una infraestructura cloud escalable y segura, orientada a la automatización de servicios en contenedores para pequeñas empresas. La idea principal es ofrecer una plataforma en la nube que permita a los clientes contratar y gestionar servicios como correo privado, almacenamiento en la nube (Drive), suite ofimática, facturación y CRM, utilizando tecnologías OpenSource.

La infraestructura se basa en servidores cloud, donde se despliega una aplicación web desarrollada con tecnologías como HTML5, CSS, PHP y SQL. Esta aplicación web actúa como interfaz principal para que los clientes puedan contratar y gestionar los servicios de manera sencilla e intuitiva. A través de la web, los usuarios pueden registrarse, iniciar sesión y seleccionar los servicios que desean contratar. Una vez que el cliente elige un servicio (por ejemplo, Nextcloud o FacturaScript), la aplicación web procesa la solicitud y, mediante scripts automatizados y Ansible, se despliega el servicio en un clúster de Kubernetes.

El clúster de Kubernetes está compuesto por un nodo maestro y varios nodos worker, donde se gestionan los contenedores de las aplicaciones. Cada servicio contratado por el cliente se despliega en un contenedor independiente, garantizando la escalabilidad y el aislamiento de los recursos. Además, se ha implementado un sistema de autenticación seguro que utiliza hashes para proteger las credenciales de los usuarios, evitando la fuga de contraseñas.

Aunque en esta fase inicial no se ha implementado un firewall ni se utiliza Terraform para la creación de máquinas remotas, se ha dejado la puerta abierta para su futura integración. El firewall se añadirá en una

fase posterior para mejorar la seguridad de la infraestructura, mientras que Terraform se utilizará para automatizar la creación y configuración de las máquinas remotas en la nube.

En resumen, el proyecto ofrece una solución integral y automatizada para pequeñas empresas que desean migrar a la nube, proporcionando una plataforma web intuitiva y un sistema de despliegue de servicios escalable y seguro.

1.3. Abstract

This project focuses on the design and implementation of a scalable and secure cloud infrastructure aimed at automating containerized services for small businesses. The main goal is to provide a cloud platform that allows customers to subscribe to and manage services such as private email, cloud storage (Drive), office suites, billing, and CRM, using OpenSource technologies.

The infrastructure is based on cloud servers, where a web application developed with technologies like HTML5, CSS, PHP, and SQL is deployed. This web application serves as the primary interface, enabling customers to easily and intuitively subscribe to and manage services. Through the web platform, users can register, log in, and select the services they wish to subscribe to. Once a customer chooses a service (e.g., Nextcloud or FacturaScript), the web application processes the request and, through automated scripts and Ansible, deploys the service on a Kubernetes cluster.

The Kubernetes cluster consists of a master node and several worker nodes, where application containers are managed. Each service subscribed by the customer is deployed in an independent container, ensuring scalability and resource isolation. Additionally, a secure authentication system using hashes has been implemented to protect user credentials and prevent password leaks.

Although a firewall and Terraform are not implemented in this initial phase, their future integration has been planned. The firewall will be added in a later phase to enhance infrastructure security, while Terraform will be used to automate the creation and configuration of remote cloud machines.

In summary, this project offers a comprehensive and automated solution for small businesses looking to migrate to the cloud, providing an intuitive web platform and a scalable, secure service deployment system.

1.4. Introducción

En la actualidad, las pequeñas empresas se enfrentan al desafío de adaptarse a un entorno tecnológico en constante evolución, donde la gestión eficiente de datos y procesos empresariales es fundamental para mantenerse competitivas. Sin embargo, muchas de estas empresas no cuentan con los recursos económicos ni técnicos necesarios para implementar infraestructuras complejas y costosas. Este proyecto surge como una respuesta a esta necesidad, ofreciendo una solución integral y automatizada que permite a las pequeñas empresas acceder a servicios en la nube de manera sencilla, segura y, sobre todo, económica.

El uso de tecnologías y software OpenSource es uno de los pilares fundamentales de este proyecto. Al basar la infraestructura en herramientas como Docker, Kubernetes, Ansible, y sistemas operativos como Ubuntu Server, se consigue reducir significativamente los costes asociados a licencias de software propietario. Además, el software OpenSource cuenta con el respaldo de una amplia comunidad de desarrolladores y usuarios que contribuyen constantemente a su mejora, lo que garantiza una mayor escalabilidad y flexibilidad en la implementación de los servicios. Esta comunidad no solo proporciona soporte y actualizaciones, sino que también permite adaptar las soluciones a las necesidades específicas de cada empresa.

1.5. Objetivos generales

1. Diseñar e implementar una infraestructura de red segura y escalable para ofrecer servicios en la nube.
2. Automatizar la creación y gestión de contenedores para el despliegue de aplicaciones.
3. Desarrollar una aplicación web que permita a los clientes contratar y gestionar servicios de nube privada, correo y facturación.

1.6. Definición general del Proyecto

El proyecto se centra en la creación de una plataforma en la nube que ofrece servicios como correo privado, almacenamiento en la nube (Drive), suite ofimática, facturación y CRM, todos ellos basados en tecnologías OpenSource. La infraestructura se ha diseñado para ser escalable y segura, permitiendo a los clientes contratar y gestionar estos servicios a través de una aplicación web intuitiva. Esta aplicación web, desarrollada con tecnologías como HTML5, CSS, PHP y SQL, actúa

como interfaz principal para que los usuarios puedan registrarse, iniciar sesión y seleccionar los servicios que desean contratar.

Uno de los aspectos más innovadores del proyecto es la automatización del despliegue de servicios mediante el uso de Kubernetes y Ansible. Cada vez que un cliente contrata un servicio, se despliega automáticamente en un contenedor dentro del clúster de Kubernetes, lo que garantiza un alto nivel de eficiencia y escalabilidad. Además, se ha implementado un sistema de autenticación seguro que utiliza hashes para proteger las credenciales de los usuarios, evitando así posibles fugas de contraseñas.

---

2. Memoria del Proyecto

2.1. Descripción Técnica

El proyecto se basa en una arquitectura cloud escalable y segura, diseñada para ofrecer servicios en la nube a pequeñas empresas. La infraestructura técnica se compone de los siguientes elementos clave:

1. Servidores Cloud

La infraestructura se aloja en servidores cloud contratados, lo que permite una mayor flexibilidad y escalabilidad. Estos servidores son el núcleo del proyecto, ya que en ellos se despliegan todos los componentes necesarios para el funcionamiento de la plataforma.

El actual proyecto se ha desarrollado sobre máquinas virtuales con el software VirtualBox para la realización de las pruebas. Sin embargo, está ideado para su implementación en cloud.

2. Aplicación Web

La interfaz principal del proyecto es una aplicación web desarrollada con tecnologías modernas como **HTML5**, **CSS**, **PHP** y **SQL**. Esta aplicación web permite a los clientes registrarse, iniciar sesión y contratar los servicios ofrecidos por la plataforma. Los servicios disponibles incluyen correo privado, almacenamiento en la nube (Drive), suite ofimática, facturación y CRM.

- **Funcionalidades de la aplicación web:**
    - Registro y autenticación de usuarios mediante un sistema seguro que utiliza hashes para proteger las credenciales.

- Interfaz intuitiva para la contratación de servicios.
- Panel de control para que los usuarios gestionen los servicios contratados.

- **Despliegue automatizado con Ansible y Kubernetes:**

La aplicación web se despliega de manera automatizada dentro del clúster de Kubernetes utilizando Ansible. Este proceso incluye la creación de contenedores basados en imágenes de Docker, que contienen todo el entorno necesario para ejecutar la aplicación:

- Imagen de Apache con PHP: Esta imagen se utiliza para desplegar el frontend y el backend de la aplicación web, asegurando que el servidor web esté correctamente configurado para ejecutar scripts PHP.

- Imagen de MariaDB: Se utiliza para desplegar la base de datos que almacena la información de los usuarios, los servicios contratados y otros datos relevantes. MariaDB es una alternativa OpenSource a MySQL, compatible y eficiente.

El despliegue automatizado garantiza que la aplicación web esté siempre disponible y escalable, adaptándose a las necesidades de los clientes. Además, el uso de contenedores en Kubernetes permite un aislamiento eficiente de los recursos y una gestión centralizada de la infraestructura.

3. Clúster de Kubernetes

El corazón de la infraestructura es un clúster de **Kubernetes**, que se encarga de gestionar los contenedores donde se ejecutan los servicios contratados por los clientes. El clúster está compuesto por:

- **Nodo maestro:** Encargado de gestionar y orquestar los contenedores desplegados en los nodos worker.
- **Nodos worker:** Donde se ejecutan los contenedores con los servicios contratados por los clientes (por ejemplo, Nextcloud o FacturaScript).

Cada servicio contratado se despliega en un contenedor independiente, lo que garantiza el aislamiento de recursos y la escalabilidad del sistema.

4. Automatización con Ansible

Para automatizar el despliegue y la configuración de los servicios, se utiliza **Ansible**. Este sistema de automatización permite:

- Desplegar los servicios en el clúster de Kubernetes de manera dinámica y eficiente.
- Configurar los entornos necesarios para cada servicio contratado.
- Gestionar la infraestructura de manera centralizada, reduciendo el tiempo y los errores asociados a la configuración manual.

5. Base de Datos

La aplicación web utiliza una base de datos **SQL** (mariaDB) para almacenar la información de los usuarios, los servicios contratados y otros datos relevantes. La base de datos se integra con la aplicación web a través de **PHP**, permitiendo una gestión eficiente de la información.

6. Futuras Implementaciones

Aunque en esta fase inicial no se ha implementado un **firewall** ni se utiliza **Terraform**, se ha planificado su futura integración para mejorar la seguridad y la automatización de la infraestructura:

- **Firewall:** En una fase posterior, se implementará un firewall (pfSense) para proteger el tráfico de red, garantizar un entorno más seguro y balancear la carga de trabajo entre los diferentes nodos.
- **Terraform:** Se utilizará para automatizar la creación y configuración de las máquinas remotas en la nube, lo que permitirá una gestión más eficiente y escalable de los recursos.

7. Tecnologías Utilizadas

- **Desarrollo Web:** HTML5, CSS6, PHP 8.3, mariaDB 11.3.2.
- **Infraestructura:** Ubuntu Server 24, Apache 2.4.
- **Contenedores y Orquestación:** Docker, Kubernetes 1.32.
- **Automatización:** Ansible 2.18.
- **Control de Versiones:** Git.
- **Virtualización:** VirtualBox 7.0.16

8. Flujo de Trabajo

1. El cliente accede a la aplicación web. Debe registrarse y seleccionar el servicio que desea contratar o logearse con su usuario para acceder al panel de control.
2. La aplicación web procesa la solicitud a través de PHP y envía los datos a Ansible.
3. Ansible despliega el servicio en un contenedor dentro del clúster de Kubernetes.
4. El servicio queda disponible para el cliente, quien puede acceder a él a través de la aplicación web.

2.2. Objetivos concretos y situación inicial

Objetivos concretos:
El proyecto tiene como objetivo principal ofrecer una solución tecnológica accesible y eficiente para pequeñas empresas que desean migrar a la nube. Para ello, se han definido los siguientes objetivos específicos:

- **Automatización del despliegue de servicios:** Utilizando herramientas como Kubernetes y Ansible, se busca automatizar la creación y gestión de contenedores, reduciendo el tiempo y los errores asociados a la configuración manual.
- **Interfaz web intuitiva:** Desarrollar una aplicación web que permita a los clientes contratar y gestionar servicios de manera sencilla, con un enfoque en la usabilidad y la experiencia del usuario.
- **Seguridad y escalabilidad:** Implementar un sistema de autenticación seguro y garantizar que la infraestructura sea escalable, permitiendo adaptarse a las necesidades crecientes de los clientes.
- **Reducción de costes:** Al basar la infraestructura en tecnologías OpenSource, se minimizan los costes asociados a licencias de software propietario, haciendo que la solución sea más accesible para pequeñas empresas.

Situación inicial:
El proyecto se enmarca en un mercado donde existen soluciones similares, pero con un enfoque diferenciador en la automatización, la escalabilidad y el uso de tecnologías OpenSource. La aplicación se

integra en un entorno donde las pequeñas empresas buscan soluciones tecnológicas económicas y fáciles de implementar, pero que a la vez sean seguras y escalables.

La infraestructura inicial se compone de servidores en máquinas virtuales, donde se ha desplegado un clúster de Kubernetes gestionado con Ansible para en un futuro realizar la implementación en servidores cloud. La aplicación web, desarrollada con tecnologías como HTML5, CSS, PHP y SQL, actúa como la interfaz principal para que los clientes puedan contratar y gestionar servicios. Aunque en esta fase no se ha implementado un firewall ni se utiliza Terraform, se ha planificado su futura integración para mejorar aún más la seguridad y la automatización de la infraestructura.

En resumen, el proyecto parte de una situación inicial donde se ha establecido una base sólida para ofrecer servicios en la nube de manera automatizada y escalable, con un enfoque claro en la reducción de costes y la seguridad.

2.3. Requisitos necesarios para la implantación

Para la implementación del proyecto, se han definido los siguientes requisitos técnicos y de infraestructura, teniendo en cuenta que la primera fase se lleva a cabo en un entorno de máquinas virtuales:

1. Requisitos de Hardware

- **Máquinas virtuales:**
    - Se utilizarán **3 máquinas virtuales** con **Ubuntu Server 24** como sistema operativo base.
    - Cada máquina virtual contará con un mínimo de **2 núcleos de CPU** y **4 GB de RAM**, lo que garantiza un rendimiento adecuado para la ejecución de los servicios y la aplicación web.
    - Una máquina actuará como **nodo de control** con Ansible instalado, otro nodo será el **nodo maestro** de Kubernetes, mientras que el otro u otros nodos funcionarán como **nodos worker** para la ejecución de los contenedores.

2. Requisitos de Software

- **Sistema operativo:** Ubuntu Server 24 en todas las máquinas virtuales.
- **Servidor web:** Apache para alojar la aplicación web en un servicio de Kubernetes.

- **Base de datos:** MariaDB para gestionar la información de usuarios y servicios.
- **Automatización:** Ansible para el nodo de control y Kubernetes en los nodos del clúster.
- **Desarrollo web:**
    - **HTML5**, **CSS**, **PHP** y **SQL** para el desarrollo de la aplicación web.
    - **PhpMyAdmin** para la gestión de la base de datos.

3. Requisitos de Red

- **Configuración de red:**
    - Las máquinas virtuales estarán conectadas en una red interna con direcciones IP fijas para garantizar la comunicación entre los nodos del clúster de Kubernetes.
    - Se utilizará un esquema de red privada con direcciones IP estáticas para evitar conflictos y facilitar la configuración.

4. Requisitos de Seguridad

- **Autenticación segura:**
    - Se implementará un sistema de autenticación basado en hashes para proteger las credenciales de los usuarios.
    - Las contraseñas no se almacenarán en texto plano, lo que reduce el riesgo de fugas de información.
- **Acceso SSH:**
    - Como requisito para el funcionamiento de Ansible todas las máquinas virtuales estarán configuradas para permitir el acceso remoto mediante SSH, utilizando claves públicas y privadas para garantizar la seguridad.
- **Usuario Administrador**
    - Se crea un usuario administrador en todos los nodos con permisos sudo y sin necesidad de introducir contraseñas para la gestión de los nodos a través de Ansible.

5. Requisitos de Automatización

Despliegue automatizado con Ansible:

Ansible se utiliza como la herramienta principal para la automatización de todo el proceso de configuración y despliegue. Esto incluye:

- **Instalación de Kubernetes:** Ansible gestiona la instalación y configuración de Kubernetes en todas las máquinas virtuales, tanto en el nodo maestro como en los nodos worker.

- **Creación del clúster:** Mediante playbooks de Ansible, se automatiza la creación del clúster de Kubernetes, configurando la comunicación entre el nodo maestro y los nodos worker, y asegurando que todos los componentes estén correctamente integrados.

- **Despliegue de servicios:** Ansible también se encarga de desplegar los servicios en el clúster de Kubernetes. Cada vez que un cliente contrata un servicio, Ansible ejecuta los playbooks necesarios para crear y configurar los contenedores correspondientes, garantizando que los servicios estén disponibles de manera rápida y eficiente.

- **Configuración de la red y seguridad:** Ansible automatiza la configuración de la red interna entre los nodos, así como la implementación de medidas de seguridad básicas, como la configuración de SSH con claves públicas y privadas.

6. Futuras Implementaciones

Aunque en esta fase inicial no se ha implementado un **firewall** ni se utiliza **Terraform**, se ha planificado su futura integración para mejorar la seguridad y la automatización de la infraestructura:

- **Firewall:** En una fase posterior, se implementará un firewall para proteger el tráfico de red y garantizar un entorno más seguro.
- **Terraform:** Se utilizará para automatizar la creación y configuración de las máquinas remotas en la nube, lo que permitirá una gestión más eficiente y escalable de los recursos.

2.4. Entregables y Acuerdos

Entregables del Proyecto

Como parte de la finalización del proyecto, se entregarán los siguientes elementos:

1. **Documento del Proyecto:**
    - Este documento incluirá toda la información técnica, la planificación, los requisitos, la descripción de la infraestructura y los manuales de uso. Será la memoria final del proyecto, detallando cada fase del desarrollo y la implementación.
2. **Video Demostrativo:**
    - Un video en el que se mostrará el funcionamiento de la aplicación web, el proceso de contratación de servicios, y el despliegue automatizado de los mismos en el clúster de Kubernetes. El video servirá como una demostración práctica de las funcionalidades del proyecto.
3. **Diagrama de la Topología de Red:**
    - Un diagrama detallado que representará la arquitectura de la red, incluyendo la disposición de los servidores, el clúster de Kubernetes, la comunicación entre los nodos y la integración de la aplicación web. Este diagrama ayudará a visualizar la estructura técnica del proyecto.
4. **Diagrama de Flujo del Proyecto:**
    - Un diagrama que describirá el flujo de trabajo del proyecto, desde la contratación de servicios por parte del cliente hasta el despliegue automatizado en Kubernetes. Este diagrama incluirá los pasos clave, las herramientas utilizadas y la interacción entre los diferentes componentes del sistema.

Acuerdos y Planificación

- **Reparto de Tareas:**
    - El trabajo se ha dividido entre los miembros del equipo, asignando responsabilidades específicas para el desarrollo de la aplicación web, la configuración de la infraestructura, la automatización con Ansible y la documentación del proyecto. Aunque realmente se ha realizado de manera conjunta en todas las fases del proyecto.
    - Cada miembro del equipo se ha comprometido a cumplir con los plazos establecidos para garantizar la entrega del proyecto en tiempo y forma.

2.5. Metodología o ciclo de vida

La Metodología del proyecto ha sido Hackathon. Dado el corto plazo de tiempo disponible para el desarrollo del proyecto, se ha seguido un enfoque práctico y continuo, sin un modelo de versiones formal, pero con una secuencia clara de tareas que permitieron avanzar de manera eficiente. A continuación, se describe el ciclo de vida seguido durante el desarrollo del proyecto:

1. Configuración de los Nodos

- El primer paso fue la configuración de las **máquinas virtuales** que actuarían como nodos para la infraestructura.
- Se instaló **Ubuntu Server 24** en cada una de las máquinas, asegurando que cumplieran con los requisitos mínimos de hardware (2 núcleos y 4 GB de RAM).
- Se configuró la red interna entre los nodos, asignando direcciones IP fijas para garantizar la comunicación entre ellos.

2. Instalación de Ansible y sus Dependencias

- Una vez configurados los nodos, se procedió a la instalación de **Ansible** en la máquina designada como nodo de control.
- Se configuraron los archivos de inventario y los playbooks básicos para gestionar la automatización de tareas en los nodos remotos.
- Se instalaron las dependencias necesarias para que Ansible pudiera interactuar con los nodos y ejecutar comandos de manera remota.

3. Configuración e Instalación de Kubernetes

- Utilizando Ansible, se automatizó la **instalación de Kubernetes** en todos los nodos.
- Se configuró el **nodo maestro** y los **nodos worker**, asegurando que estuvieran correctamente comunicados y listos para formar el clúster.
- Se verificó la correcta creación del **clúster de Kubernetes**, asegurando que los nodos estuvieran sincronizados y listos para ejecutar contenedores.

4. Diseño y Desarrollo de la Aplicación Web

- Paralelamente, se inició el **diseño y desarrollo de la aplicación web**, utilizando tecnologías como **HTML5**, **CSS**, **PHP** y **SQL**.

- Se crearon las interfaces de usuario para el registro, autenticación y contratación de servicios, así como el panel de control para la gestión de los mismos.
- Se integró la aplicación web con la base de datos **MariaDB** para almacenar la información de los usuarios y los servicios contratados.

5. Despliegue de la Aplicación Web y Creación del Servicio

- Una vez finalizado el desarrollo de la aplicación web, se procedió a su **despliegue en el clúster de Kubernetes**.
- Se creó un **servicio** en Kubernetes para exponer la aplicación web, asegurando que estuviera accesible desde el exterior.
- Se configuraron los servicios en modo **NodePort** para la exposición de estos fuera del clúster.
- En un futuro el despliegue de los servicios se realizará mediante **IngressController** para gestionar el tráfico entrante y redirigirlo al servicio de la aplicación web.

6. Creación de Despliegues y Servicios Adicionales

- Finalmente, se crearon los **despliegues y servicios adicionales** para los servicios ofrecidos por la plataforma (como Nextcloud y FacturaScript).
- Cada servicio se desplegó en un contenedor independiente dentro del clúster de Kubernetes, utilizando imágenes de **Docker** preconfiguradas.
- Se automatizó el proceso de despliegue mediante **playbooks de Ansible**, lo que permitió que los servicios estuvieran disponibles de manera rápida y eficiente.

7. Pruebas y Ajustes Finales

- A lo largo de todo el proceso, se realizaron **pruebas continuas** para verificar el correcto funcionamiento de cada componente.
- Se ajustaron los playbooks de Ansible, las configuraciones de Kubernetes y la aplicación web para corregir errores y optimizar el rendimiento.

3. Planificación y Costes

3.1. Planificación

La planificación del proyecto se ha estructurado en fases claras, teniendo en cuenta el corto plazo de tiempo disponible y la necesidad de avanzar de manera eficiente. A continuación, se detalla la temporalización de las actividades principales:

Fase 1: Configuración Inicial de los Nodos

- **Configuración de las máquinas virtuales:**
    - Instalación de Ubuntu Server 24 en las 3 máquinas virtuales (nodo maestro, nodos worker y nodo de control).
    - Configuración de la red interna con direcciones IP fijas para garantizar la comunicación entre los nodos.
- **Preparación del entorno:**
    - Instalación de dependencias básicas y configuración de acceso SSH entre los nodos.

Fase 2: Automatización con Ansible

- **Instalación de Ansible:**
    - Configuración de Ansible en el nodo de control.
    - Creación del archivo de inventario y playbooks básicos para la gestión de los nodos remotos.
- **Automatización de tareas iniciales:**
    - Uso de Ansible para instalar y configurar dependencias comunes en todos los nodos (Docker, Kubernetes, etc.).

Fase 3: Instalación y Configuración de Kubernetes

- **Instalación de Kubernetes:**
    - Automatización de la instalación de Kubernetes en el nodo maestro y los nodos worker mediante playbooks de Ansible.
- **Creación del clúster:**
    - Configuración del clúster de Kubernetes, asegurando la comunicación entre el nodo maestro y los nodos worker.
    - Verificación del correcto funcionamiento del clúster.

Fase 4: Desarrollo de la Aplicación Web

- **Diseño y desarrollo:**
    - Creación de la aplicación web utilizando HTML5, CSS, PHP y SQL.
    - Desarrollo de las interfaces de usuario para el registro, autenticación y contratación de servicios.
- **Integración con la base de datos:**
    - Configuración de MariaDB para almacenar la información de usuarios y servicios.
    - Integración de la aplicación web con la base de datos mediante PHP.

Fase 5: Despliegue de la Aplicación Web y Servicios

- **Despliegue en Kubernetes:**
    - Creación de un despliegue y servicio en Kubernetes para la aplicación web.
    - Configuración de un IngressController para gestionar el tráfico entrante.
- **Despliegue de servicios adicionales:**
    - Automatización del despliegue de servicios como Nextcloud y FacturaScript mediante playbooks de Ansible.
    - Creación de contenedores independientes para cada servicio dentro del clúster de Kubernetes.

Fase 6: Pruebas y Ajustes Finales

- **Pruebas de funcionamiento:**
    - Verificación del correcto funcionamiento de la aplicación web, los servicios desplegados y la automatización con Ansible.
- **Ajustes y optimización:**
    - Corrección de errores y optimización de los playbooks de Ansible, las configuraciones de Kubernetes y la aplicación web.

- **Preparación de entregables:**
    - Documentación del proyecto, creación del video demostrativo, diagrama de la topología de red y diagrama de flujo del proyecto.

Fase 7: Presentación y Entrega

- **Presentación final:**
    - Preparación de la presentación del proyecto, incluyendo una demostración en vivo del funcionamiento de la plataforma.
- **Entrega de documentación:**
    - Entrega del documento del proyecto, manuales de uso, video demostrativo y diagramas.

3.2. Manuales de usuario

Manual de Despliegue del Proyecto

Este manual está dirigido a los administradores o técnicos que deseen implementar la infraestructura del proyecto desde cero. Su objetivo es guiar paso a paso en la instalación, configuración y despliegue de todos los componentes necesarios para que la plataforma funcione correctamente.

1. Requisitos Previos

Antes de comenzar, es necesario contar con al menos tres máquinas virtuales con **Ubuntu Server 24** instalado. Cada máquina debe tener un mínimo de **2 núcleos de CPU** y **4 GB de RAM**. Además, se requiere acceso a una red interna con direcciones IP fijas para garantizar la comunicación entre los nodos. Es recomendable tener conocimientos básicos de Linux, Kubernetes y Ansible para seguir este manual.

2. Configuración de las Máquinas Virtuales

El primer paso es configurar las máquinas virtuales. En una de ellas, que actuará como **nodo de control**, se instalará Ansible. Las otras dos máquinas serán el **nodo maestro** de Kubernetes y un **nodo worker**. Es importante asegurarse de que todas las máquinas tengan acceso SSH entre sí, utilizando claves públicas y privadas para mayor seguridad. Para ello, se generan claves SSH en el nodo de control y se copian a los demás nodos.

3. Instalación de Ansible
En el nodo de control, se procede a instalar Ansible. Una vez instalado, se configura el archivo de inventario, donde se especifican las direcciones IP de los nodos maestro y worker. También se crean los playbooks iniciales para tareas básicas, como la instalación de dependencias comunes (Kubernetes, curl, http-transport, etc.) en todos los nodos. Ansible se convierte en la herramienta central para automatizar todas las configuraciones posteriores.

4. Automatización de la Instalación de Kubernetes
Utilizando Ansible, se automatiza la instalación de Kubernetes en el nodo maestro y el nodo worker. Esto incluye la configuración de los componentes necesarios, como **kubeadm**, **kubelet** y **kubectl**. Una vez instalado, se inicializa el clúster en el nodo maestro y se unen los nodos worker al clúster. Se verifica que todos los nodos estén correctamente sincronizados y listos para ejecutar contenedores. Este proceso se realiza mediante playbooks de Ansible, lo que garantiza que sea repetible y escalable.

5. Despliegue de la Aplicación Web
La aplicación web, desarrollada con **HTML5**, **CSS**, **PHP** y **SQL**, se despliega en el clúster de Kubernetes. Para ello, se genera un namespace específico por el cual se crea un despliegue (Deployment) en Kubernetes que define cómo se ejecutará la aplicación. Luego, se expone la aplicación mediante un servicio de tipo **NodePort**. Este tipo de servicio permite acceder a la aplicación web desde fuera del clúster utilizando la dirección IP de cualquiera de los nodos y un puerto específico (por ejemplo, http://<IP-del-nodo>:30000).

Su futura implementación será a través de IngressController.

6. Despliegue de Servicios Adicionales
Los servicios adicionales, como **Nextcloud** y **FacturaScript**, también se despliegan en el clúster de Kubernetes al ser contratados por el cliente. Cada servicio se configura mediante un despliegue (Deployment) y se expone mediante un servicio de tipo **NodePort**. Esto permite que los usuarios accedan a estos servicios a través de la dirección IP de los nodos y los puertos asignados. Por ejemplo, Nextcloud podría estar accesible en http://<IP-del-nodo>:30001 y FacturaScript en http://<IP-del-nodo>:30002.

7. Verificación y Pruebas
Una vez desplegados todos los servicios, se realizan pruebas para verificar que todo funciona correctamente. Esto incluye acceder a la aplicación web, contratar servicios y comprobar que los servicios

adicionales están disponibles y funcionan como se espera. También se verifica que los servicios sean escalables y que el clúster de Kubernetes gestione correctamente la carga.

Manual de Usuario de Acceso a la Web

Este manual está dirigido a los usuarios finales (clientes) que utilizarán la plataforma para contratar y gestionar servicios. Su objetivo es guiar a los usuarios en el uso de la aplicación web de manera sencilla e intuitiva.

1. Acceso a la Aplicación Web

Para acceder a la aplicación web, los usuarios deben ingresar a la URL proporcionada por la plataforma. Dado que el acceso se realiza a través de **NodePort**, la URL tendrá un formato similar a http://<IP-del-nodo>:30000. En la página de inicio, encontrarán dos opciones principales: **Registrarse** (para nuevos usuarios) o **Iniciar Sesión** (para usuarios ya registrados).

2. Registro de Nuevos Usuarios

Los nuevos usuarios deben hacer clic en el botón **Registrarse** y completar un formulario con sus datos personales, como nombre, correo electrónico y contraseña. La contraseña se almacena de manera segura utilizando un sistema de hashes, lo que garantiza la protección de las credenciales. Una vez completado el registro, el usuario recibirá un correo de confirmación y podrá iniciar sesión.

3. Inicio de Sesión

Los usuarios registrados pueden iniciar sesión introduciendo su correo electrónico y contraseña en la página de inicio. Si el usuario olvida su contraseña, puede utilizar la opción **Recuperar Contraseña**, que le permitirá restablecerla mediante un enlace enviado a su correo electrónico.

4. Contratación de Servicios

Una vez dentro de la aplicación web, los usuarios pueden navegar por los servicios disponibles, como **Nextcloud** (almacenamiento en la nube) o **FacturaScript** (gestión de facturas). Para contratar un servicio, el usuario debe hacer clic en el botón **Contratar** y seguir las instrucciones que aparecen en pantalla. El sistema procesará la solicitud y desplegará automáticamente el servicio en el clúster de Kubernetes. Una vez desplegado, el servicio estará accesible a través de su respectivo **NodePort** (por ejemplo, http://<IP-del-nodo>:30001 para Nextcloud).

5. Panel de Control
Después de contratar un servicio, el usuario accederá a un **panel de control** donde podrá gestionar sus servicios. Desde aquí, podrá ver el estado de los servicios contratados, modificar configuraciones, acceder a las herramientas proporcionadas por cada servicio (por ejemplo, subir archivos a Nextcloud o generar facturas en FacturaScript) y cancelar servicios si lo desea.

6. Soporte y Ayuda
En caso de problemas o dudas, los usuarios pueden acceder a la sección de **Ayuda** dentro de la aplicación web, donde encontrarán respuestas a preguntas frecuentes y un formulario para contactar con el soporte técnico. Además, se proporciona una guía básica sobre cómo acceder a los servicios.

3.3. Licencias

El proyecto **ASAN-TECH** se basa en el uso de tecnologías y software **OpenSource**, lo que permite reducir costes y aprovechar las ventajas de la comunidad de desarrollo. A continuación, se detallan las licencias principales que se aplican en el proyecto:

1. Licencia del Software Utilizado

- **Kubernetes:** Bajo la licencia **Apache 2.0**, que permite el uso, modificación y distribución del software, siempre que se incluya la nota de licencia original y se reconozca la autoría.
- **Docker:** Utiliza la licencia **Apache 2.0**, similar a Kubernetes, lo que permite su uso en proyectos comerciales y no comerciales sin restricciones significativas.
- **Ansible:** También bajo la licencia **GPLv3** (GNU General Public License), que garantiza la libertad de usar, modificar y distribuir el software, siempre que cualquier derivado también se distribuya bajo la misma licencia.
- **Ubuntu Server 24:** Distribuido bajo la licencia **GPL**, que permite su uso y modificación libremente, siempre que se respeten los términos de la licencia.
- **Nextcloud:** Bajo la licencia **AGPLv3** (Affero General Public License), que asegura que cualquier modificación o extensión del software también esté disponible bajo la misma licencia.

- **FacturaScript:** Utiliza la licencia **LGPL** (Lesser General Public License), que permite su uso en proyectos propietarios, siempre que se mantenga la licencia original para el software.

2. Licencia del Proyecto ASAN-TECH

El proyecto **ASAN-TECH** se distribuirá bajo la licencia **GPLv3** (GNU General Public License). Esta licencia fue elegida por las siguientes razones:

- **Libertad de uso:** Permite a cualquier persona utilizar el software sin restricciones, ya sea para fines comerciales o no comerciales.
- **Modificación y distribución:** Cualquier modificación o derivado del proyecto debe distribuirse bajo la misma licencia, lo que fomenta la colaboración y el crecimiento de la comunidad OpenSource.
- **Transparencia:** Garantiza que el código fuente esté disponible para todos, promoviendo la transparencia y la mejora continua del proyecto.

3. Justificación de la Elección de Licencias

La elección de licencias OpenSource para el proyecto se debe a varios factores clave:

- **Reducción de costes:** Al utilizar software OpenSource, se evitan los costes asociados a licencias de software propietario, lo que hace que la solución sea más accesible para pequeñas empresas.
- **Comunidad y soporte:** Las licencias OpenSource cuentan con el respaldo de una amplia comunidad de desarrolladores y usuarios, lo que garantiza un soporte continuo, actualizaciones y mejoras.
- **Flexibilidad y adaptabilidad:** Las licencias OpenSource permiten modificar y adaptar el software a las necesidades específicas del proyecto, lo que es fundamental para una infraestructura escalable y personalizable.

4. Consideraciones Futuras

Aunque en esta fase inicial no se ha implementado un **firewall** ni se utiliza **Terraform**, se ha planificado su futura integración. Ambos componentes también se basarán en software OpenSource, lo que garantiza la coherencia con la filosofía del proyecto y la continuidad en el uso de licencias libres.

3.4. Recursos

El proyecto **ASAN-TECH** requiere una combinación de recursos de **hardware** y **software** para su desarrollo, implementación y funcionamiento. A continuación, se detallan los recursos necesarios, explicando su papel en el proyecto y cómo contribuyen al éxito de la infraestructura.

1. Recursos de Hardware

Máquinas Virtuales

El núcleo del proyecto se basa en tres máquinas virtuales que actúan como los nodos principales de la infraestructura. Estas máquinas están configuradas con **Ubuntu Server 24**, un sistema operativo estable y ampliamente utilizado en entornos de servidores. Cada máquina cuenta con un mínimo de **2 núcleos de CPU** y **4 GB de RAM**, lo que garantiza un rendimiento adecuado para la ejecución de los servicios y la aplicación web. Además, se ha asignado un mínimo de **20 GB de almacenamiento** por máquina, preferiblemente en SSD para mejorar la velocidad de acceso a los datos.

Las máquinas virtuales están conectadas a internet a través de un adaptador NAT y a la **red interna** a través de un adaptador Puente con direcciones IP fijas, lo que facilita la comunicación entre los nodos y asegura que el clúster de Kubernetes funcione correctamente. Esta configuración es esencial para garantizar que los nodos maestro y worker estén sincronizados y listos para ejecutar contenedores.

Hardware Adicional

Además de las máquinas virtuales, se ha utilizado un **ordenador de desarrollo** con especificaciones más potentes (al menos 32 GB de RAM y un procesador de 16 núcleos y discos m2) para facilitar la creación y prueba de la aplicación web, así como la gestión de las máquinas virtuales. Este equipo también cuenta con una **conexión a Internet estable**, necesaria para la descarga de dependencias y la gestión remota de los servidores.

En una fase posterior, se planea migrar la infraestructura a **servidores cloud contratados**, lo que permitirá una mayor escalabilidad y disponibilidad. Esto será especialmente útil cuando el proyecto esté en producción y se requiera un mayor rendimiento y redundancia.

2. Recursos de Software

Sistemas Operativos

El proyecto se basa en **Ubuntu Server 24** como sistema operativo principal para todas las máquinas virtuales. Esta elección se debe a su estabilidad, soporte a largo plazo y compatibilidad con las herramientas utilizadas en el proyecto, como Docker, Kubernetes y Ansible. Ubuntu Server es una opción popular en entornos de servidores debido a su facilidad de uso y su amplia comunidad de soporte.

Herramientas de Contenerización y Orquestación

Para la gestión de los servicios, se utiliza **containerd** como herramienta de contenerización. Containerd es el runtime que permite empaquetar aplicaciones y sus dependencias en contenedores de imágenes descargadas de DockerHub, lo que facilita su despliegue y gestión en diferentes entornos. Estos contenedores se orquestan mediante **Kubernetes**, que gestiona el clúster de nodos y asegura que los servicios estén siempre disponibles y escalables.

La automatización de tareas se realiza mediante **Ansible**, una herramienta que permite configurar y desplegar los servicios de manera eficiente. Ansible se utiliza para instalar dependencias, configurar Kubernetes y desplegar la aplicación web y los servicios adicionales en el clúster.

Servidor Web y Base de Datos

La aplicación web se aloja en un servicio de Kubernetes donde se instala **Apache**, que proporciona un entorno estable y seguro para la ejecución de aplicaciones PHP. La base de datos se gestiona mediante otro servicio de Kubernetes con **MariaDB**, una alternativa OpenSource a MySQL que ofrece un alto rendimiento y compatibilidad con la aplicación web. Para facilitar la gestión de la base de datos, se utiliza una imagen de Apache con **PhpMyAdmin**, una herramienta gráfica que permite administrar la base de datos de manera intuitiva.

Desarrollo de la Aplicación Web

La aplicación web se ha desarrollado utilizando tecnologías modernas como **HTML5**, **CSS**, **PHP** y **SQL**. Estas tecnologías permiten crear una interfaz de usuario intuitiva y funcional, así como gestionar la lógica del negocio y la interacción con la base de datos. El desarrollo se ha realizado en **Visual Studio Code**, un editor de código ligero pero potente, que facilita la escritura y depuración del código.

Herramientas de Control de Versiones

Para gestionar el código fuente y colaborar en el desarrollo, se utiliza **Git**. Git permite mantener un historial de cambios, trabajar en

equipo de manera eficiente y revertir errores si es necesario. Además, se ha utilizado **VirtualBox** para crear y gestionar las máquinas virtuales durante la fase de desarrollo, lo que permite simular el entorno de producción de manera local.

Software Adicional
El proyecto incluye el despliegue de servicios adicionales como Nextcloud (para almacenamiento en la nube) y FacturaScript (para gestión de facturas). Estos servicios se despliegan en contenedores dentro del clúster de Kubernetes, lo que garantiza su aislamiento y escalabilidad.

3. Recursos Humanos

El proyecto ha sido desarrollado por un equipo de dos personas con conocimientos en administración de sistemas, redes, Kubernetes, Ansible y desarrollo web. La colaboración continua entre los miembros del equipo ha sido fundamental para la implementación, pruebas y documentación del proyecto. Además, se ha contado con el apoyo de la comunidad OpenSource, que ha proporcionado recursos, tutoriales y soluciones a problemas técnicos.

4. Futuras Necesidades de Recursos

Aunque en esta fase inicial no se ha implementado un **firewall** ni se utiliza **Terraform**, se ha planificado su futura integración. El firewall será necesario para proteger el tráfico de red y garantizar un entorno más seguro, mientras que Terraform se utilizará para automatizar la creación y configuración de máquinas remotas en la nube. Estas herramientas permitirán una gestión más eficiente y escalable de la infraestructura.

3.5. Diseño de la base de datos

La base de datos es un componente fundamental del proyecto **ASAN-TECH**, ya que almacena toda la información relacionada con los clientes y los servicios que contratan. A continuación, se describe el diseño de la base de datos, basado en el script SQL proporcionado, y cómo se integra con el resto de la infraestructura.

1. Estructura de la Base de Datos

La base de datos, llamada **asan_tech**, se compone de dos tablas principales: **clientes** y **servicios**. Estas tablas están diseñadas para gestionar la información de los usuarios y los servicios que contratan, respectivamente.

1.1. Tabla 'clientes'

Esta tabla almacena la información básica de los clientes que utilizan la plataforma. Los campos principales son:

- **id_cliente:** Un identificador único para cada cliente, generado automáticamente mediante AUTO_INCREMENT. Este campo es la clave primaria de la tabla.
- **nombre:** El nombre del cliente, que es un campo obligatorio (NOT NULL) y tiene un límite de 100 caracteres.
- **contrasena:** La contraseña del cliente, que se almacena de manera segura utilizando un hash. Este campo también es obligatorio y tiene un límite de 255 caracteres para garantizar que se puedan almacenar hashes largos.

1.2. Tabla 'servicios'

Esta tabla almacena la información de los servicios contratados por los clientes. Los campos principales son:

- **id_servicio:** Un identificador único para cada servicio, generado automáticamente mediante AUTO_INCREMENT. Este campo es la clave primaria de la tabla.
- **id_cliente:** Un campo que hace referencia al cliente que ha contratado el servicio. Este campo es una clave foránea (FOREIGN KEY) que referencia la tabla clientes y está configurado con ON DELETE CASCADE, lo que significa que, si un cliente es eliminado, todos sus servicios también se eliminarán automáticamente.
- **nombre_servicio:** El nombre del servicio contratado, que es un campo obligatorio y tiene un límite de 100 caracteres.
- **descripcion:** Una descripción detallada del servicio, que puede ser opcional y no tiene un límite de longitud definido.
- **fecha_inicio:** La fecha y hora en que el servicio fue contratado.
- **fecha_fin:** La fecha y hora en que el servicio finaliza (si es aplicable).
- **costo:** El costo del servicio, representado como un número decimal con hasta 10 dígitos en total y 2 decimales.

2. Relaciones entre Tablas

La relación entre las tablas **clientes** y **servicios** es de **uno a muchos**, lo que significa que un cliente puede contratar múltiples servicios, pero

cada servicio está asociado a un único cliente. Esta relación se establece mediante la clave foránea id_cliente en la tabla servicios, que referencia la clave primaria id_cliente en la tabla clientes.

La configuración ON DELETE CASCADE asegura que, si un cliente es eliminado de la base de datos, todos los servicios asociados a ese cliente también se eliminen automáticamente. Esto mantiene la integridad de la base de datos y evita la existencia de registros huérfanos.

3. Integración con la Aplicación Web

La base de datos se integra con la aplicación web a través de **PHP**, que se encarga de gestionar las consultas y operaciones sobre la base de datos. Cuando un cliente se registra en la plataforma, se inserta un nuevo registro en la tabla clientes. Cuando un cliente contrata un servicio, se inserta un nuevo registro en la tabla servicios, asociándolo al cliente correspondiente mediante el campo id_cliente.

Además, la aplicación web utiliza **PhpMyAdmin** como herramienta de gestión de la base de datos, lo que permite a los administradores realizar consultas, inserciones, actualizaciones y eliminaciones de manera gráfica y sencilla.

4. Futuras Mejoras

Aunque la base de datos actual cumple con los requisitos del proyecto, se han identificado algunas posibles mejoras para futuras versiones:

- **Tabla de roles:** Se podría añadir una tabla para gestionar roles de usuarios (por ejemplo, administrador, cliente, etc.), lo que permitiría implementar un sistema de permisos más avanzado.
- **Historial de cambios:** Se podría añadir una tabla para registrar cambios en los servicios (por ejemplo, modificaciones en el costo o la descripción), lo que permitiría mantener un historial de las operaciones realizadas.
- **Optimización de consultas:** A medida que la base de datos crezca, se podrían implementar índices adicionales para optimizar el rendimiento de las consultas.

4. Versiones del Software

4.1. Prototipos

Dado que el proyecto se ha desarrollado en un formato de **hackathon**, la versión actual representa el **primer prototipo funcional**. Este prototipo incluye la infraestructura básica, la aplicación web y los servicios principales, pero se ha planificado una serie de mejoras y prototipos futuros para ampliar y optimizar la plataforma. A continuación, se describen las versiones futuras del proyecto:

1. Primer Prototipo (Versión Actual)

La versión actual del proyecto incluye:

- Una **infraestructura básica** basada en máquinas virtuales con Ubuntu Server 24.
- Un **clúster de Kubernetes** gestionado con Ansible para la orquestación de contenedores.
- Una **aplicación web** desarrollada con HTML5, CSS, PHP y SQL, que permite a los clientes registrarse, iniciar sesión y contratar servicios.
- Servicios como **Nextcloud** y **FacturaScript** desplegados en contenedores dentro del clúster.
- Una **base de datos** (MariaDB) para almacenar la información de los clientes y los servicios contratados.

Este prototipo cumple con los requisitos mínimos para ofrecer una solución funcional, pero está diseñado para ser escalable y permitir mejoras futuras.

2. Segundo Prototipo: Integración de Terraform y Firewall

En esta versión, se planea mejorar la infraestructura mediante la integración de **Terraform** y un **firewall**:

- **Terraform:** Se utilizará para automatizar la creación y configuración de máquinas remotas en la nube. Esto permitirá una gestión más eficiente de los recursos y facilitará el despliegue de nuevos nodos en el clúster de Kubernetes.
- **Firewall:** Se implementará un firewall para proteger el tráfico de red y garantizar un entorno más seguro. El firewall se configurará para filtrar el tráfico entrante y saliente, protegiendo los servicios y la aplicación web de posibles ataques.

3. Tercer Prototipo: Subdominios y Acceso Seguro

En esta versión, se mejorará la forma en que los clientes acceden a sus servicios:

- **Subdominios personalizados:** Cada cliente tendrá un subdominio único (por ejemplo, cliente1.asan-tech.com) para acceder a sus servicios. Esto mejorará la experiencia del usuario y facilitará la gestión de los servicios.
- **Acceso más seguro:** Se implementarán medidas de seguridad adicionales, como certificados SSL/TLS para cada subdominio y autenticación de dos factores (2FA) para el acceso a los servicios.

4. Cuarto Prototipo: Copias de Seguridad

En esta versión, se añadirá un sistema de **copias de seguridad** para proteger los datos de la empresa y los clientes:

- **Copias de seguridad automáticas:** Se configurarán copias de seguridad periódicas de la base de datos y los servicios, que se almacenarán en un servidor seguro o en la nube.
- **Restauración de datos:** Se implementará una herramienta que permita restaurar los datos en caso de pérdida o corrupción, garantizando la continuidad del negocio.

5. Quinto Prototipo: Mejora de la Interfaz Web

En esta versión, se mejorará la **interfaz web** para ofrecer una experiencia más personalizada y funcional:

- **Panel de control avanzado:** Los clientes podrán realizar cambios y configuraciones en sus servicios directamente desde la aplicación web, sin necesidad de contactar con el soporte técnico.
- **Diseño responsive:** La interfaz web se optimizará para dispositivos móviles, mejorando la experiencia de los usuarios que acceden desde smartphones o tablets.
- **Notificaciones y alertas:** Se añadirá un sistema de notificaciones para informar a los clientes sobre el estado de sus servicios, actualizaciones o problemas técnicos.

6. Prototipos Futuros Adicionales

Además de las versiones mencionadas, se han identificado algunas ideas adicionales para futuros prototipos:

6.1. Implementación de un Sistema de Monitorización

- Se integrará una herramienta de monitorización (como Prometheus y Grafana) para supervisar el estado del clúster de Kubernetes, los servicios y la infraestructura en tiempo real.
- Esto permitirá detectar y resolver problemas de rendimiento o disponibilidad antes de que afecten a los clientes.

6.2. Integración con Herramientas de Colaboración

- Se añadirán herramientas de colaboración en la nube, como **OnlyOffice** o **Collabora**, para permitir a los clientes trabajar en documentos, hojas de cálculo y presentaciones directamente desde la plataforma.
- Esto ampliará la oferta de servicios y mejorará la productividad de los clientes.

6.3. Implementación de un Sistema de Facturación Automatizado

- Se desarrollará un sistema de facturación automatizado que genere facturas y recordatorios de pago de manera automática, integrado con **FacturaScript**.
- Esto reducirá la carga administrativa y mejorará la gestión financiera de los clientes.

5. Presentación del Software

5.1. Conclusiones

El proyecto **ASAN-TECH** representa una solución innovadora y accesible para pequeñas empresas y usuarios que desean migrar a la nube de manera sencilla y económica. La idea central del proyecto es **ofrecer servicios en la nube**, como correo privado, almacenamiento, suite ofimática y facturación, utilizando tecnologías **OpenSource** y una infraestructura altamente automatizada. Esto no solo reduce los costes asociados a licencias de software propietario, sino que también garantiza una implementación rápida y eficiente, adaptada a las necesidades de **empresas con recursos limitados**.

Accesibilidad y Ventaja Comercial

Una de las principales ventajas de **ASAN-TECH** es su **accesibilidad**. Las pequeñas empresas y usuarios finales pueden acceder a servicios en la nube de alta calidad sin necesidad de realizar grandes inversiones en infraestructura o personal técnico especializado. La plataforma está diseñada para ser intuitiva y fácil de usar, lo que permite a los clientes contratar y gestionar servicios en cuestión de minutos. Esta facilidad de uso, combinada con un coste reducido, representa una **gran ventaja comercial** en un mercado donde muchas empresas buscan soluciones tecnológicas económicas pero eficientes.

Escalabilidad y Automatización

El proyecto está pensado para ser **escalable** desde su concepción. Gracias al uso de tecnologías como **Kubernetes**, **Docker** y **Ansible**, la infraestructura puede crecer según las necesidades de los clientes, añadiendo nuevos nodos o servicios sin interrupciones. Además, la **automatización** de todos los procesos, desde el despliegue de servicios hasta la configuración de la red, garantiza que la plataforma sea eficiente y fácil de gestionar. Esto no solo reduce el tiempo de implementación, sino que también minimiza los errores humanos y los costes operativos.

Replicabilidad y Flexibilidad

Uno de los aspectos más destacados del proyecto es su **replicabilidad**. Las bases de **ASAN-TECH** están diseñadas para que la infraestructura pueda replicarse en cuestión de minutos, como si se tratara de un **backup de la infraestructura**. Esto significa que otras empresas o proyectos pueden utilizar esta misma base para desplegar sus propias infraestructuras de red de manera rápida y automatizada. Esta flexibilidad abre la puerta a nuevas ideas de negocio que requieran una infraestructura en la nube escalable y eficiente, sin necesidad de partir desde cero.

Futuro del Proyecto

El proyecto no solo cumple con los objetivos iniciales, sino que también sienta las bases para futuras mejoras y expansiones. La inclusión de tecnologías como **Terraform**, un **firewall** avanzado, subdominios personalizados y sistemas de copias de seguridad permitirá que **ASAN-TECH** siga creciendo y adaptándose a las necesidades del mercado.

Además, la filosofía **OpenSource** del proyecto asegura que pueda beneficiarse de las contribuciones de la comunidad, lo que garantiza su evolución continua.

Impacto en el Mercado

En un mundo donde la digitalización es cada vez más importante, **ASAN-TECH** se posiciona como una solución **ideal para pequeñas empresas** que buscan sumarse a las nuevas tecnologías sin incurrir en costes prohibitivos. La combinación de **accesibilidad, escalabilidad y automatización** hace que este proyecto no solo sea viable, sino también altamente competitivo en el mercado actual. Además, su capacidad para replicarse y adaptarse a otros negocios lo convierte en una herramienta versátil y valiosa para cualquier empresa que busque desplegar una infraestructura en la nube de manera rápida y eficiente.

5.2. Anexos

- **Anexo I:** Diagrama de la Topología de Red
- **Anexo II:** Diagrama de la infraestructura del proyecto. y la base de datos.
- **Anexo III:** Diagrama de Flujo de trabajo.
- **Anexo IV:** Código GitHub.

5.3. Webgrafía

- **Ansible.** (s.f.). *Documentación oficial de Ansible*. Recuperado de https://docs.ansible.com
- **Apache HTTP Server.** (s.f.). *Documentación oficial de Apache*. Recuperado de https://httpd.apache.org/docs/
- **Docker.** (s.f.). *Documentación oficial de Docker*. Recuperado de https://docs.docker.com
- **FacturaScripts.** (s.f.). *Documentación oficial de FacturaScripts*. Recuperado de https://facturascripts.com/documentacion
- **Git.** (s.f.). *Documentación oficial de Git*. Recuperado de https://git-scm.com/doc
- **Kubernetes.** (s.f.). *Documentación oficial de Kubernetes*. Recuperado de https://kubernetes.io/docs/home/
- **MariaDB.** (s.f.). *Documentación oficial de MariaDB*. Recuperado de https://mariadb.com/kb/en/documentation/
- **Nextcloud.** (s.f.). *Documentación oficial de Nextcloud*. Recuperado de https://docs.nextcloud.com

- **PHP.** (s.f.). *Documentación oficial de PHP*. Recuperado de https://www.php.net/docs.php
- **Ubuntu.** (s.f.). *Documentación oficial de Ubuntu Server*. Recuperado de https://ubuntu.com/server/docs
- **Visual Studio Code.** (s.f.). *Documentación oficial de Visual Studio Code*. Recuperado de https://code.visualstudio.com/docs