1. Introducción

Durante los años 2023 y 2024, he desarrollado 4 proyectos personales centrados en el despliegue de aplicaciones web en diferentes entornos tecnológicos. Estos proyectos tienen como objetivo principal adquirir y perfeccionar habilidades en el uso de diversas tecnologías relacionadas con la virtualización, contenedores, cloud y servicios serverless.

El primer proyecto, titulado **"Virtualización"**, consistía en desarrollar una aplicación web y desplegarla en un entorno de máquinas virtuales utilizando VirtualBox. El objetivo era simular un entorno productivo y comprender los fundamentos esenciales de la virtualización correctamente.

El segundo proyecto, **"Contenedores"**, se centraba en el despliegue de la misma aplicación web en un entorno de contenedores utilizando Docker Desktop. El objetivo principal era familiarizarse con la tecnología de contenedores, aprender a crear y gestionar imágenes Docker, y orquestar múltiples contenedores mediante Docker Compose.

El tercer proyecto, **"Cloud"**, abordaba el despliegue de la aplicación web en AWS (Amazon Web Services). Con tan solo un crédito de 100 dolares, crear una infraestructura compleja utilizando varios servicios de AWS como VPC, EC2, RDS y S3, con el objetivo de aprender a gestionar y escalar aplicaciones en un entorno cloud.

Finalmente, el cuarto proyecto, **"Serverless"**, implicaba el desarrollo de la aplicación web utilizando servicios serverless de AWS, tales como Lambda y API Gateway. El propósito de este proyecto era reducir la complejidad de la infraestructura y entender las ventajas de los servicios serverless para desarrollar aplicaciones escalables y eficientes.

2. Desarrollo y resultados

Proyecto 1: Virtualización

En el primer proyecto, se creó una infraestructura de máquinas virtuales utilizando VirtualBox. Se generaron tres máquinas virtuales (VM) con Ubuntu Server como sistema operativo. La primera VM, denominada "frontend", tenía Nginx instalado para desplegar la aplicación del frontend. La segunda VM, llamada "backend", fue configurada con **Spring Boot** para el desarrollo del backend, junto con Corretto 11 y Tomcat para su despliegue. Además, se instaló el cliente MySQL para permitir la conexión con la base de datos. La tercera VM, nombrada "bd", se dedicó al servidor MySQL, donde se creó la base de datos necesaria para la aplicación backend.

La configuración de las VM se realizó utilizando la funcionalidad de redireccionamiento de puertos para permitir el acceso vía SSH y web desde la máquina anfitriona. También se modificó el archivo application-VM.yml para configurar las propiedades de conexión a la base de datos MySQL.

Proyecto 2: Contenedores

En este segundo proyecto, se utilizó Docker Desktop para crear una infraestructura de contenedores para la misma aplicación web. Se crearon imágenes Docker tanto para el backend como para el frontend, basándose en imágenes de Tomcat y Nginx, respectivamente. El backend fue desarrollado y configurado con **Spring Boot**, Corretto 11 y Tomcat, mientras que el frontend se desplegó utilizando Nginx. Además, se creó un contenedor para MySQL, donde se alojaba la base de datos, y otro contenedor con Adminer para facilitar la gestión de la misma.

Se utilizó Docker Compose para orquestar todos los contenedores, asegurando que estuvieran en la misma red y pudieran comunicarse entre ellos. Los archivos Dockerfile y docker-compose.yml se revisaron minuciosamente para garantizar una configuración y despliegue correctos.

Proyecto 3: Cloud

El tercer proyecto se centró en el despliegue de la aplicación web en AWS (Amazon Web Services). Se creó una VPC (Virtual Private Cloud) con subnets públicas y un Internet Gateway para permitir el acceso a internet. Para controlar el tráfico de red, se configuró una Network ACL (Access Control List) y un Security Group.

Se creó una Amazon Machine Image (AMI) para una instancia EC2, en la cual se desplegó el backend desarrollado con **Spring Boot**, junto con Corretto 11 y Tomcat, dentro de la VPC. Además, se configuró una instancia RDS (Relational Database Service) con MySQL para la base de datos. Para asegurar la disponibilidad y escalabilidad de la aplicación, se creó un Auto Scaling Group y se utilizó un Load Balancer para distribuir el tráfico entre las instancias EC2. Por último, se creó un Bucket S3 para el frontend, permitiendo así el acceso público a la aplicación.

Proyecto 4: Serverless

En el cuarto y último proyecto, se utilizó AWS Lambda para gestionar el backend y API Gateway para definir la API. El backend no utilizó **Spring Boot** en este caso, ya que fue completamente reescrito para funcionar en un entorno serverless con Lambda. Se creó una base de datos en DynamoDB, adaptando la estructura de datos a un modelo no relacional. En este proyecto, se desarrollaron tres funciones Lambda: una para obtener todos los miembros, otra para obtener los gastos de un miembro específico, y una tercera para actualizar los gastos de los miembros.

El frontend se desplegó en un Bucket S3, similar a la tercera práctica, pero con las URL apuntando a la nueva API definida en API Gateway. Esta infraestructura serverless redujo significativamente la complejidad y los costos asociados al mantenimiento de la aplicación.

Tecnologías y Herramientas

Proyecto 1: Virtualización

- **Métodos:** Configuración de máquinas virtuales, red NAT.
- **Técnicas:** Redireccionamiento de puertos, configuración de servicios.
- **Tecnologías:** VirtualBox, Ubuntu Server, **Spring Boot**, Nginx, Tomcat, MySQL.
- **Herramientas:** VirtualBox, cliente SSH, cliente MySQL.

Proyecto 2: Contenedores

- **Métodos:** Creación de imágenes Docker, orquestación de contenedores.
- **Técnicas:** Dockerfile, Docker Compose.
- **Tecnologías:** Docker Desktop, **Spring Boot**, Tomcat, Corretto 11, Nginx, MySQL, Adminer.
- **Herramientas:** Docker, Docker Compose.

Proyecto 3: Cloud

- **Métodos:** Configuración de servicios en la nube, escalabilidad automática.
- **Técnicas:** Creación de VPC, subnets, Auto Scaling.
- **Tecnologías:** AWS (VPC, EC2, RDS, S3, Auto Scaling, Load Balancer), **Spring Boot**, Corretto 11, Tomcat, MySQL.
- **Herramientas:** Consola AWS, AWS CLI.

Proyecto 4: Serverless

- **Métodos:** Desarrollo de funciones Lambda, API Gateway.
- **Técnicas:** Gestión de bases de datos no relacionales, integración de servicios.
- **Tecnologías:** AWS (Lambda, API Gateway, DynamoDB, S3), Node.js.
- **Herramientas:** AWS Management Console, AWS CLI, entorno de desarrollo de Node.js.

Costos

Proyecto 1: Virtualización

- **Estudio:** 10 horas dedicadas a familiarizarse con tecnologías como VirtualBox, **Spring Boot** y Ubuntu Server.
- **Diseño:** 8 horas para planificar la infraestructura de las VMs y definir los servicios en cada VM.
- **Implementación:** 12 horas para la creación y configuración de las VMs, instalación de servicios (Nginx, **Spring Boot**, Tomcat, MySQL), configuración de la red NAT y redireccionamiento de puertos.
- **Experimentación:** 5 horas dedicadas a pruebas y ajustes para asegurar la correcta comunicación entre los componentes de la aplicación.

Proyecto 2: Contenedores

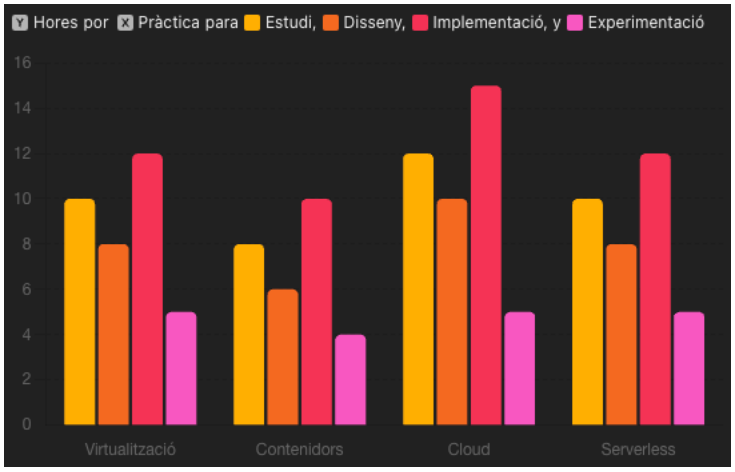
- **Estudio:** 8 horas para aprender Docker y Docker Compose, y entender la funcionalidad de cada herramienta.
- **Diseño:** 6 horas para planificar la infraestructura de contenedores, crear Dockerfiles para backend (**Spring Boot**) y frontend, y definir la orquestación con Docker Compose.
- **Implementación:** 10 horas para crear las imágenes Docker, configurar MySQL y Adminer en contenedores, y escribir el archivo docker-compose.yml.
- **Experimentación:** 4 horas de pruebas para asegurar el correcto funcionamiento de los contenedores y su comunicación.

Proyecto 3: Cloud

- **Estudio:** 12 horas para aprender sobre los servicios AWS (VPC, EC2, RDS, Auto Scaling, S3).
- **Diseño:** 10 horas para planificar la infraestructura en AWS, incluyendo la configuración de la red, subnets, y otros componentes necesarios.
- **Implementación:** 15 horas para crear y configurar los servicios en AWS, desplegar la instancia EC2 con el backend (**Spring Boot**), configurar RDS, Auto Scaling, Load Balancer y S3 para el frontend.
- **Experimentación:** 5 horas de pruebas para asegurar la eficiencia y seguridad de la infraestructura desplegada.

Proyecto 4: Serverless

- **Estudio:** 10 horas para aprender sobre los servicios serverless de AWS (Lambda, API Gateway, DynamoDB) y comprender su implementación.
- **Diseño:** 8 horas para planificar la arquitectura serverless, definir las funciones Lambda, la estructura de la base de datos DynamoDB y configurar API Gateway.
- **Implementación:** 12 horas para desarrollar las funciones Lambda, crear la tabla DynamoDB, configurar API Gateway y desplegar el frontend en S3.
- **Experimentación:** 5 horas de pruebas para asegurar el correcto funcionamiento de la solución serverless, ajustes y resolución de problemas.



3. Conclusiones

Estos proyectos me han proporcionado una experiencia completa en diferentes tecnologías y metodologías para el despliegue de aplicaciones web.

Virtualización:
El proyecto de virtualización fue esencial para comprender los fundamentos de la creación y configuración de infraestructuras básicas utilizando máquinas virtuales. Me permitió aprender a gestionar recursos limitados y a configurar entornos que simulan escenarios reales de producción. A través de este proyecto, adquirí una comprensión sólida sobre la importancia de la virtualización en el desarrollo y la implementación de soluciones tecnológicas.

Contenedores:
El proyecto con contenedores fue clave para entender las ventajas de la contenedorización en términos de portabilidad, escalabilidad y eficiencia. Utilizar Docker simplificó el proceso de despliegue, permitiendo encapsular las aplicaciones y sus dependencias en contenedores ligeros. Este proyecto destacó la importancia de los contenedores en el entorno de desarrollo moderno, donde la iteración rápida y el despliegue continuo son cruciales.

Cloud:
El despliegue de la aplicación en AWS me proporcionó una comprensión profunda de cómo gestionar y escalar aplicaciones en la nube. Aprendí a utilizar servicios de infraestructura como servicio (IaaS) para crear una infraestructura flexible y escalable. Este proyecto mostró las ventajas de la computación en la nube, incluyendo la facilidad para escalar recursos según las necesidades y la posibilidad de gestionar la infraestructura de manera eficiente.

Serverless:
Finalmente, el proyecto de Serverless ofreció una visión de las posibilidades de reducir la complejidad de la infraestructura mediante servicios gestionados que no requieren la gestión de servidores. El uso de AWS Lambda y API Gateway demostró cómo se pueden construir aplicaciones escalables y de bajo costo, centrándose únicamente en la lógica de negocio sin preocuparse por la gestión de la infraestructura subyacente. Este proyecto fue clave para entender los beneficios de los servicios serverless en términos de simplificación y eficiencia operativa.

A través de estos proyectos, adquirí una comprensión amplia y detallada de los diferentes entornos tecnológicos para el despliegue de aplicaciones web. Cada proyecto aportó conocimientos y habilidades específicas, desde la configuración de máquinas virtuales hasta la implementación de arquitecturas serverless. Esta experiencia práctica ha sido invaluable para desarrollar una base sólida en técnicas de aplicaciones de software y para prepararme para futuros desafíos en el campo de la tecnología de la información.