Módulo 8
Fundamentos de integración continua

Fundamentos de DevOps

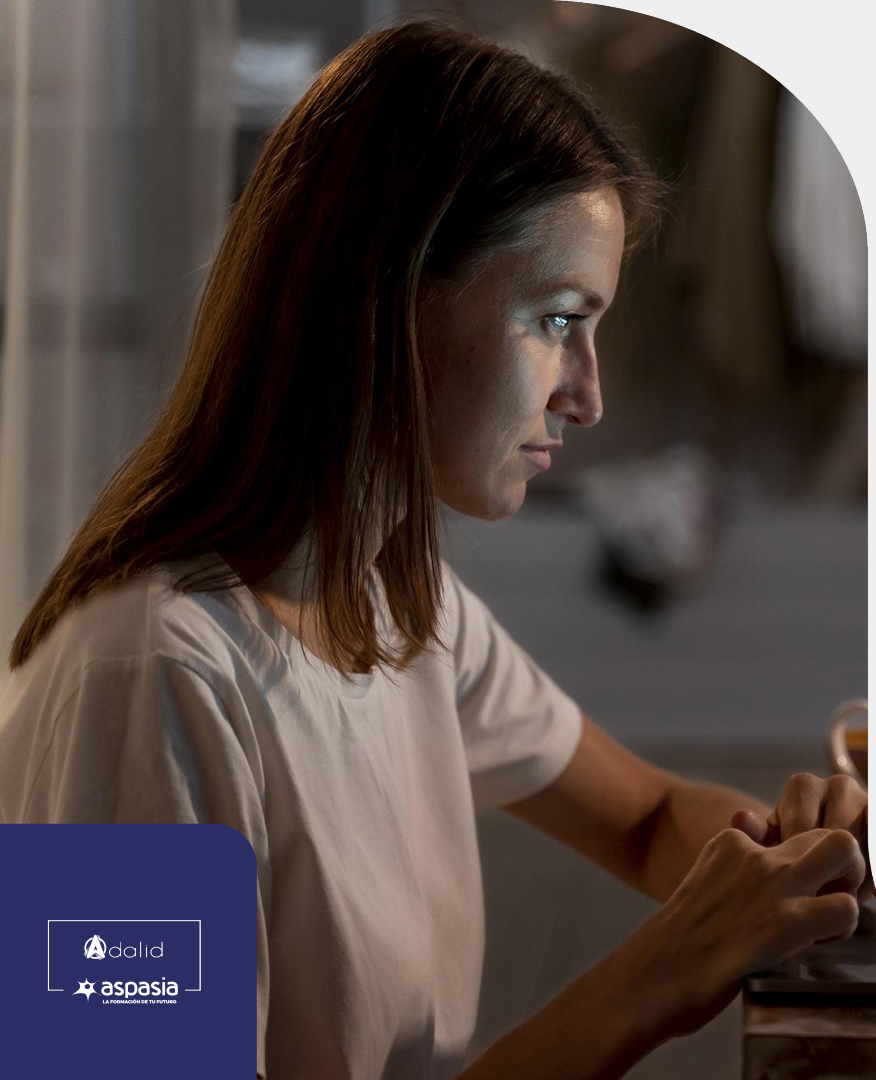
Módulo 8

AE 1.2

OBJETIVOS

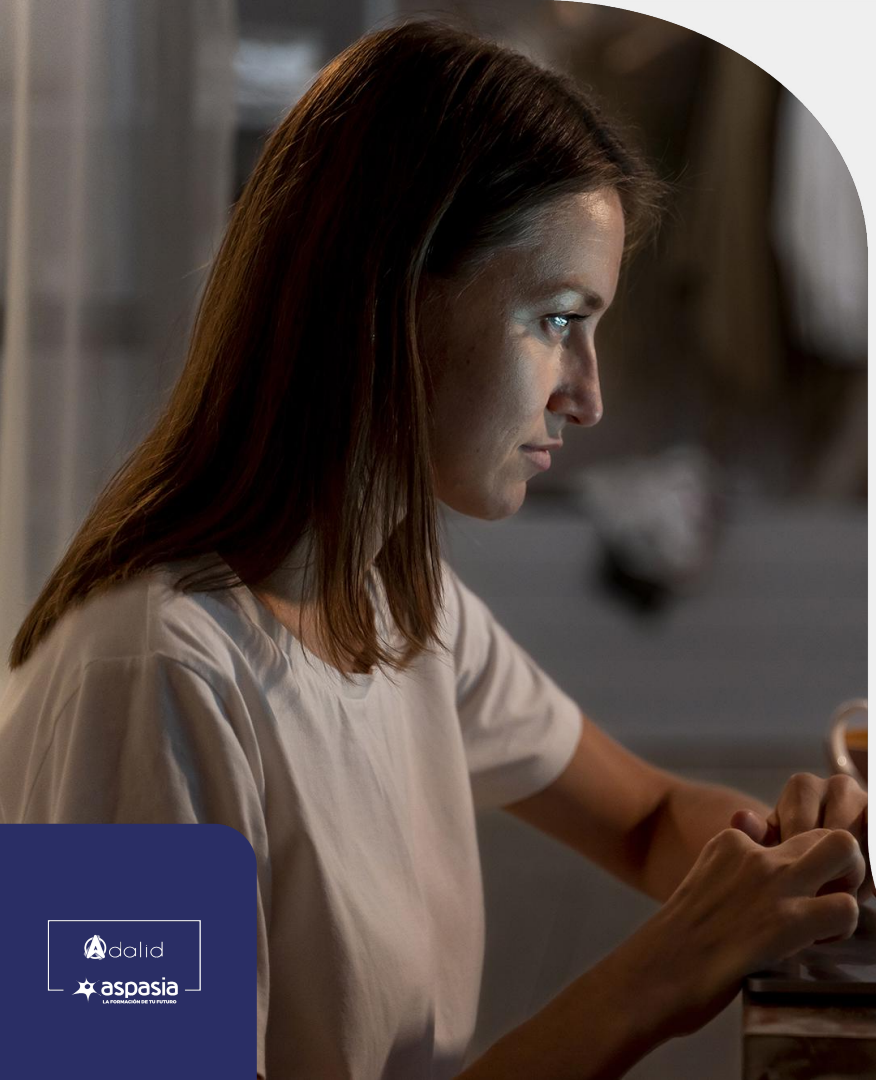
Entender qué es DevOps, su cultura, principios y beneficios en el desarrollo de software. Conocer la importancia de la Integración y Entrega Continua en la automatización del ciclo DevOps. Aprender el uso de contenedores Docker y su papel en la infraestructura ágil y escalable.

Chile
dalid
aspasia
LA FORMACIÓN DE TU FUTURO



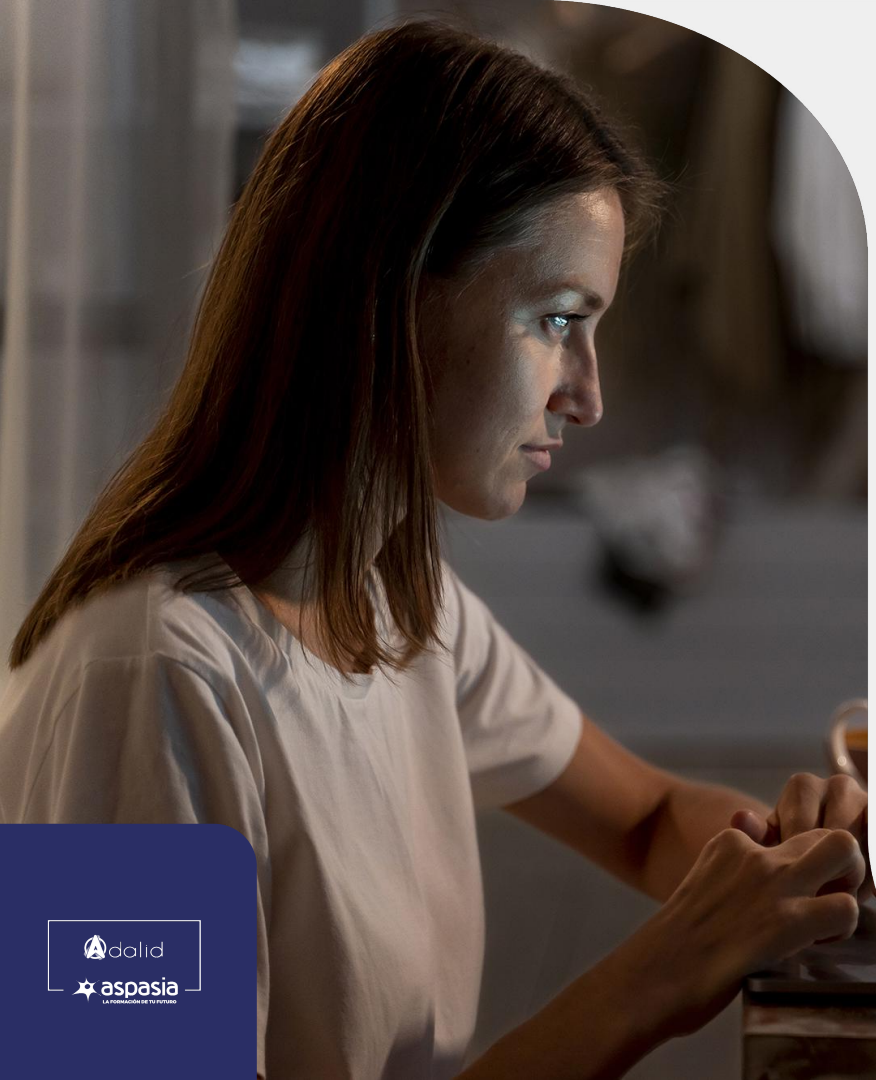
¿QUÉ VAMOS A VER?

- Fundamentos de DevOps y Cultura.
  - Qué es DevOps, su propósito, origen y evolución.
  - Cultura, principios y beneficios de DevOps.
  - Modelo CAMS (Culture, Automation, Measurement, Sharing).
  - DevOps vs. DevSecOps y ciclo de vida DevOps.



¿QUÉ VAMOS A VER?

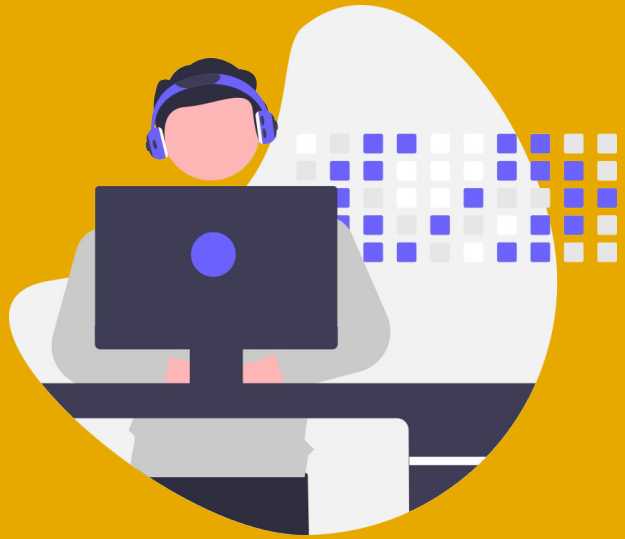
- Integración y Entrega Continua.
  - Qué es Integración Continua y diferencias con Entrega Continua.
  - Flujo del proceso de Integración Continua: control de versiones, compilación y testing.
  - Sistemas de integración continua: Jenkins, Circle CI, GitLab CI, Bamboo.



¿QUÉ VAMOS A VER?

- Contenedores de Aplicaciones y Docker.
  - Qué es un contenedor de aplicaciones y diferencias con una máquina virtual.
  - Conceptos básicos de Docker: imágenes, DockerFile, contenedores y volúmenes.
  - Beneficios y rol de Docker en el ciclo DevOps.
  - Implementación de Docker: instalación, configuración, comandos básicos y monitoreo de eventos/logs.

Pongamos a prueba lo aprendido 😊!!!



Adalid aspasia

Ejercicio Guiado: Implementando DevOps con Docker en un Proyecto Node.js y React

Crear paso a paso una aplicación con **Node.js (backend) y React (frontend)**, utilizando **Docker** para su ejecución, asegurando buenas prácticas de DevOps.

Aprenderemos a:

- Configurar Docker desde cero para un proyecto real.
- Crear contenedores separados para frontend (React) y backend (Node.js).
- Utilizar Docker Compose para gestionar múltiples servicios.
- Aplicar conceptos de Integración Continua y Entrega Continua (CI/CD).

Ejercicio Guiado: Implementando DevOps con Docker en un Proyecto Node.js y React

Paso 1: Instalación de Docker y Configuración Inicial

Instalar Docker

Antes de empezar, asegúrate de tener **Docker y Docker Compose** instalados:

- **Windows / Mac:** Descargar Docker Desktop
- **Linux:**

```
sudo apt update
sudo apt install docker.io docker-compose

docker --version # Verifica la instalación
docker-compose --version
```

Ejercicio Guiado: Implementando DevOps con Docker en un Proyecto Node.js y React

Paso 2: Configurar la Estructura del Proyecto

```
devops-project/
├── backend/ (Node.js API)
│   ├── server.js
│   ├── package.json
│   ├── Dockerfile
│   ├── .dockerignore
│   └── src/
├── frontend/ (React App)
│   ├── Dockerfile
│   ├── package.json
│   ├── .dockerignore
│   └── src/
├── docker-compose.yml
└── README.md
```

Ejercicio Guiado: Implementando DevOps con Docker en un Proyecto Node.js y React

Paso 3: Crear el Backend con Node.js

1. Inicializar un Proyecto **Node.js**

```
mkdir devops-project && cd devops-project
mkdir backend && cd backend
npm init -y
```

2. Instalar **Express.js**

```
npm install express
```

3. Crear el Archivo **backend/server.js**

```
const express = require("express");
const app = express();
const PORT = process.env.PORT || 5000;

app.get("/", (req, res) => {
  res.send("¡Hola Mundo desde Node.js en Docker! 🚀");
});

app.listen(PORT, () => {
  console.log(`Servidor corriendo en
http://localhost:${PORT}`);
});
```

Ejercicio Guiado: Implementando DevOps con Docker en un Proyecto Node.js y React

Paso 3: Crear el Backend con Node.js

4. Crear el Archivo **backend/Dockerfile**

```
# 1. Usar la imagen oficial de Node.js
FROM node:16

# 2. Crear y establecer el directorio de trabajo
WORKDIR /app

# 3. Copiar los archivos del backend
COPY package.json ./
COPY server.js ./

# 4. Instalar dependencias
RUN npm install

# 5. Exponer el puerto
EXPOSE 5000

# 6. Comando para ejecutar el servidor
CMD ["node", "server.js"]
```

5. Crear un archivo **backend/.dockerignore**

```
node_modules
npm-debug.log
```

6. **Construir** y **Ejecutar** el Contenedor del Backend

```
docker build -t my-backend .
docker run -p 5000:5000 my-backend
```

Ejercicio Guiado: Implementando DevOps con Docker en un Proyecto Node.js y React

Paso 4: Crear el Frontend con React

1. Inicializar el Proyecto React Desde la carpeta principal **(devops-project)**:

```
npm create vite@latest frontend --template react
cd frontend
npm install
```

2. Modificar **frontend/src/App.js** para Conectar con el Backend

```
import { useEffect, useState } from "react";

function App() {
  const [message, setMessage] = useState("");
  useEffect(() => {
    fetch("http://localhost:5000")
      .then((res) => res.text())
      .then((data) => setMessage(data));
  }, []);

  return (
    <div>
      <h1>React + Node.js con Docker 🚀</h1>
      <p>{message}</p>
    </div>
  );
}

export default App;
```

Ejercicio Guiado: Implementando DevOps con Docker en un Proyecto Node.js y React

Paso 4: Crear el Frontend con React

3. Crear **frontend/Dockerfile** para React

```
# 1. Usar imagen de Node.js
FROM node:16

# 2. Crear el directorio de trabajo
WORKDIR /app

# 3. Copiar archivos del frontend
COPY package.json package-lock.json ./

# 4. Instalar dependencias
RUN npm install
COPY . ./

# 5. Exponer el puerto
EXPOSE 5173

# 6. Comando para ejecutar React
CMD ["npm", "run", "dev", "--", "--host"]
```

4. Crear **frontend/.dockerignore** para React

```
node_modules
dist
npm-debug.log
.vite
```

5. **Construir** y **Ejecutar** el Contenedor del frontend

```
docker build -t my-frontend .
docker run -p 5173:5173 my-frontend
```

Ejercicio Guiado: Implementando DevOps con Docker en un Proyecto Node.js y React

Paso 4: Crear el Frontend con React

6. Orquestación con Docker Compose, crear el archivo **devops-project/docker-compose.yml**

```yaml
version: "3.8"
services:
  backend:
    build: ./backend
    ports:
      - "5000:5000"
    networks:
      - app-network
  frontend:
    build: ./frontend
    ports:
      - "5173:5173"
    depends_on:
      - backend
    networks:
      - app-network

networks:
  app-network:
```

Ejercicio Guiado: Implementando DevOps con Docker en un Proyecto Node.js y React

Paso 4: Crear el Frontend con React

6. Ejecutar **Docker Compose**:

```
docker-compose up --build
```

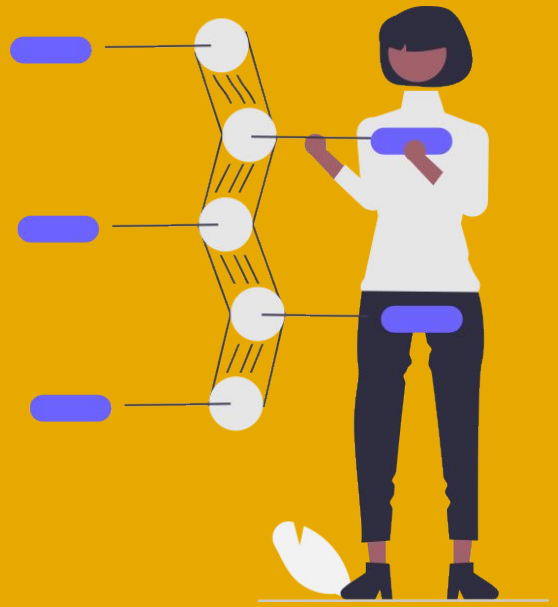
7. Detener los contenedores cuando termines:

```
docker-compose down
```

Resultado esperado:

- http://localhost:5173 → React + Vite en Docker mostrando el mensaje del backend.
- http://localhost:5000 → Node.js funcionando en Docker.

Resumen de lo aprendido 🧑‍🎓



Adalid aspasia

Resumen de lo aprendido

- DevOps fomenta la colaboración, automatización y monitoreo para mejorar el desarrollo y operaciones.
- La Integración y Entrega Continua optimizan despliegues frecuentes y confiables en software.
- Docker permite empaquetar y ejecutar aplicaciones de forma aislada, eficiente y portable.
- Contenedores facilitan la escalabilidad y consistencia en entornos DevOps modernos.

GRACIAS POR TU ATENCIÓN

Nos vemos en la próxima clase

Chile
Adalid
aspasia
LA FORMACIÓN DE TU FUTURO