EduStreaming

Guía de Instalación y Despliegue con Docker

Plataforma web para transmisión de clases en vivo y bajo demanda para entornos educativos. Esta guía explica cómo instalar Docker Desktop y desplegar la aplicación en desarrollo y producción usando Docker y Docker Compose.

Versión: 1.0 | **Fecha:** 2025-10-06

Proyecto: EduStreaming

Integrantes

Canchingre Tamayo Neil Aldhair
Castillo Gonzales Ximena Nohemí
Morales Torres Wilfrido Israel
Nieves Reinado Charli Steven
Valencia Bautista María Angélica
Arevalo Bernal Juan Diego

Guía Completa: Instalación de Docker y Despliegue de EduStreaming

Estudio de Caso: Implementación de Plataforma de Streaming Educativo

Contexto del Proyecto

Una universidad necesita implementar una plataforma de streaming para transmitir clases en vivo y bajo demanda a estudiantes remotos. La solución debe soportar hasta 500 conexiones simultáneas y ofrecer calidad adaptativa según el ancho de banda de cada usuario.

Información de Análisis del Proyecto de Aula

Objetivos del Proyecto:

- **Objetivo Principal**: Desarrollar una plataforma web de streaming educativo que permita la transmisión de clases en vivo y contenido bajo demanda
- **Objetivo Técnico**: Implementar una solución escalable usando tecnologías modernas (React, Docker, Nginx)
- **Objetivo Académico**: Demostrar competencias en desarrollo full-stack, containerización y despliegue de aplicaciones

Requerimientos Funcionales:

- **Sistema de Autenticación**: Login/registro de usuarios con roles (estudiante, profesor, admin)
- **Streaming en Vivo**: Transmisión de clases en tiempo real con chat interactivo
- **Contenido Bajo Demanda**: Biblioteca de clases grabadas con búsqueda avanzada
- **Sistema de Notificaciones**: Alertas para nuevas clases, tareas y recordatorios
- **Dashboard Administrativo**: Panel de control para profesores y administradores
- **Perfil de Usuario**: Gestión de información personal y progreso académico

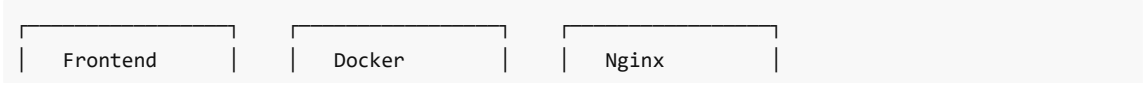
Requerimientos No Funcionales:

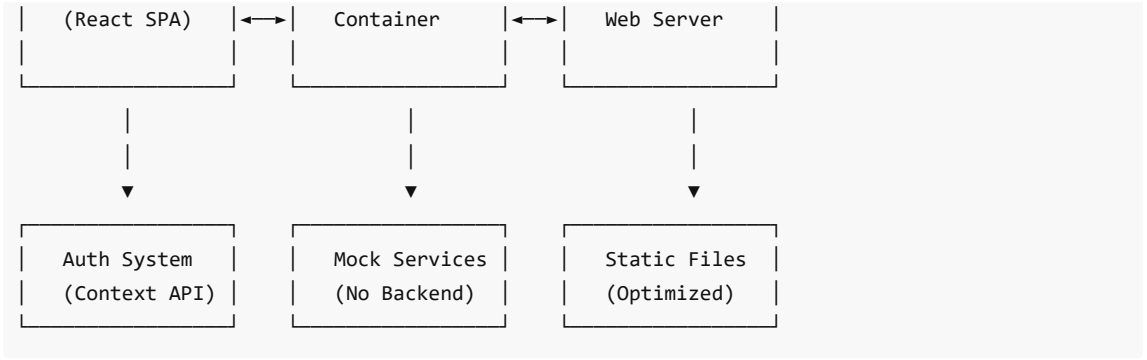
- **Escalabilidad**: Soporte para 500+ conexiones simultáneas
- **Rendimiento**: Tiempo de carga < 3 segundos
- **Disponibilidad**: 99.9% de uptime
- **Seguridad**: Autenticación segura y encriptación de datos
- **Usabilidad**: Interfaz intuitiva y responsive design
- **Compatibilidad**: Funcionamiento en múltiples navegadores y dispositivos

Tecnologías Implementadas:

- **Frontend**: React 18 + Vite + Material-UI + Styled Components
- **Containerización**: Docker + Docker Compose
- **Servidor Web**: Nginx con configuración optimizada
- **Estado Global**: Context API + React Query
- **Routing**: React Router DOM
- **Estilos**: Material-UI + Styled Components + CSS3
- **Despliegue/Hosting**: Vercel para un servicio de servidor (APIs/SSR) cuando aplica
- **Base de imagen**: Node.js 18 sobre Alpine Linux para imágenes ligeras

Arquitectura de la Solución:





Notas de arquitectura:

- Además de la containerización, se dispone de un servidor desplegado en Vercel para funcionalidades de servidor (como APIs ligeras o SSR) cuando el caso de uso lo requiere.
- Las imágenes de Docker usan una base de Alpine Linux con Node.js 18 durante la fase de build, priorizando tamaño reducido y tiempos de descarga rápidos.

Casos de Uso Principales:

1. **Estudiante Accede a Clase en Vivo**
    - Autenticación → Navegación → Selección de clase → Streaming → Chat
2. **Profesor Inicia Transmisión**
    - Login → Dashboard → Configuración → Inicio de stream → Monitoreo
3. **Administrador Gestiona Contenido**
    - Login → Panel admin → Gestión de usuarios → Configuración → Reportes
4. **Usuario Busca Contenido**
    - Búsqueda → Filtros → Resultados → Reproducción → Favoritos

Métricas de Rendimiento Objetivo:

- **Tiempo de Carga Inicial**: < 3 segundos
- **Tiempo de Respuesta API**: < 500ms
- **Calidad de Video**: 720p mínimo, 1080p recomendado
- **Latencia de Streaming**: < 5 segundos
- **Disponibilidad**: 99.9% uptime
- **Concurrencia**: 500+ usuarios simultáneos

Tabla de Contenidos

Prerrequisitos

Sistema Operativo

- **Windows 10/11** (versión 1903 o superior)
- **RAM**: Mínimo 4GB, recomendado 8GB
- **Espacio en disco**: 2GB libres
- **Procesador**: 64-bit con soporte para virtualización

Software Requerido

- **Docker Desktop** (se instalará en esta guía)
- **Git** (para clonar el repositorio)
- **Navegador web** (Chrome, Firefox, Edge)

Cuenta de GitHub

- Cuenta de GitHub activa para acceder al repositorio del proyecto
- Permisos de lectura en el repositorio EduStreaming

Instalación de Docker Desktop

Paso 1: Descargar Docker Desktop

1. **Abrir navegador web** y navegar a: https://www.docker.com/products/docker-desktop/
2. **Hacer clic** en "Download for Windows"
3. **Esperar** a que se complete la descarga (archivo .exe de aproximadamente 500MB)

Paso 2: Instalar Docker Desktop

1. **Ejecutar** el archivo descargado como administrador
2. **Aceptar** los términos de licencia
3. **Seleccionar** "Use WSL 2 instead of Hyper-V" (recomendado)
4. **Hacer clic** en "Install"
5. **Esperar** a que se complete la instalación (5-10 minutos)
6. **Reiniciar** el sistema cuando se solicite

Paso 3: Configurar Docker Desktop

1. **Abrir** Docker Desktop desde el menú de inicio
2. **Aceptar** el acuerdo de servicio
3. **Configurar** la cuenta Docker (opcional, se puede omitir)
4. **Esperar** a que Docker se inicie completamente (ícono verde en la bandeja del sistema)

Paso 4: Verificar la Instalación

1. **Abrir** PowerShell o Command Prompt
2. **Ejecutar** el siguiente comando:

```
docker --version
```

3. **Verificar** que se muestre la versión de Docker
4. **Ejecutar** el siguiente comando:

```
docker-compose --version
```

5. **Confirmar** que Docker Compose esté instalado

Despliegue en Vercel

¿Por qué Vercel?

Vercel es la plataforma elegida para el despliegue del frontend de EduStreaming por las siguientes razones:

Simplicidad y Facilidad de Uso:

- **Integración directa con GitHub**: Despliegue automático con cada push
- **Configuración mínima**: No requiere configuración compleja de servidores
- **Interfaz intuitiva**: Dashboard fácil de usar para monitoreo

Optimización para Frontend:

- **CDN global**: Entrega de contenido desde servidores cercanos al usuario
- **Optimización automática**: Compresión, minificación y caching automático
- **Edge Functions**: Ejecución de código en el edge para mejor rendimiento

Integración con GitHub:

- **Deploy automático**: Cada commit se despliega automáticamente
- **Preview deployments**: Versiones de prueba para cada pull request
- **Rollback fácil**: Reversión a versiones anteriores con un clic

Costo y Mantenimiento:

- **Plan gratuito generoso**: Suficiente para proyectos educativos
- **Sin mantenimiento de servidor**: Vercel maneja toda la infraestructura
- **Escalabilidad automática**: Se adapta al tráfico sin configuración adicional

Comparación con AWS:

- **AWS**: Requiere configuración de EC2, Load Balancers, CloudFront, etc.
- **Vercel**: Configuración en minutos vs horas/días en AWS
- **Costo**: Plan gratuito vs costos variables en AWS
- **Mantenimiento**: Cero vs constante en AWS

Desarrollo y Testing:

- **Preview URLs**: Cada branch tiene su propia URL de prueba
- **Variables de entorno**: Configuración fácil de diferentes entornos
- **Analytics integrado**: Métricas de rendimiento incluidas

Configuración de Vercel

Paso 1: Crear Cuenta en Vercel

1. **Navegar** a https://vercel.com
2. **Hacer clic** en "Sign Up"
3. **Seleccionar** "Continue with GitHub"
4. **Autorizar** Vercel para acceder a GitHub

Paso 2: Conectar Repositorio

1. **En el dashboard de Vercel**, hacer clic en "New Project"
2. **Seleccionar** el repositorio "EduStreaming" de GitHub

3. **Hacer clic** en "Import"

Paso 3: Configurar Proyecto

1. **Framework Preset**: Seleccionar "Vite"
2. **Root Directory**: Dejar por defecto (./)
3. **Build Command**: `npm run build`
4. **Output Directory**: `dist`
5. **Install Command**: `npm install`

Paso 4: Desplegar

1. **Hacer clic** en "Deploy"
2. **Esperar** a que se complete el build (2-5 minutos)
3. **Verificar** que el despliegue sea exitoso
4. **Acceder** a la URL proporcionada por Vercel

Monitoreo y Gestión

Dashboard de Vercel:

- **Analytics**: Métricas de rendimiento y uso
- **Functions**: Monitoreo de serverless functions
- **Domains**: Gestión de dominios personalizados
- **Environment Variables**: Configuración de variables de entorno

Despliegues Automáticos:

- **Cada push a main**: Despliegue automático a producción
- **Pull requests**: Preview deployments automáticos
- **Rollback**: Reversión a versiones anteriores desde el dashboard

---

Configuración del Proyecto

Paso 1: Clonar el Repositorio

1. **Abrir** PowerShell o Command Prompt
2. **Navegar** al directorio donde se desea clonar el proyecto
3. **Ejecutar** el siguiente comando:

```
git clone https://github.com/tu-usuario/EduStreaming.git
```

4. **Navegar** al directorio del proyecto:

```
cd EduStreaming
```

Paso 2: Verificar la Estructura del Proyecto

La estructura del proyecto debe ser similar a:

```
EduStreaming/
├── src/
│   ├── components/
│   ├── pages/
│   ├── services/
```

```
│   └── styles/
├── public/
├── docker-compose.yml
├── Dockerfile
├── package.json
└── README.md
```

Paso 3: Verificar Archivos Docker

Dockerfile (debe existir en la raíz del proyecto):

```
# Dockerfile para desarrollo
FROM node:18-alpine

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "run", "dev"]
```

docker-compose.yml (debe existir en la raíz del proyecto):

```
version: '3.8'

services:
  edustreaming:
    build: .
    ports:
      - "3000:3000"
    volumes:
      - .:/app
      - /app/node_modules
    environment:
      - NODE_ENV=development
```

Despliegue en Desarrollo

Paso 1: Construir la Imagen Docker

1. **Abrir** PowerShell en el directorio del proyecto
2. **Ejecutar** el siguiente comando:

   ```
   docker-compose build
   ```

3. **Esperar** a que se complete la construcción (2-5 minutos)

Paso 2: Iniciar el Contenedor

1. **Ejecutar** el siguiente comando:

```
docker-compose up
```

2. **Verificar** que no haya errores en la salida
3. **Esperar** a que aparezca el mensaje "Local: http://localhost:3000"

Paso 3: Acceder a la Aplicación

1. **Abrir** el navegador web
2. **Navegar** a: http://localhost:3000
3. **Verificar** que la aplicación cargue correctamente
4. **Probar** la funcionalidad básica (navegación, login, etc.)

Paso 4: Verificar Logs

1. **En la terminal** donde se ejecutó docker-compose up
2. **Observar** los logs de la aplicación
3. **Verificar** que no haya errores críticos
4. **Para detener** la aplicación: `Ctrl + C`

Despliegue en Producción

Paso 1: Preparar el Dockerfile de Producción

Dockerfile (versión optimizada para producción):

```
# Multi-stage build para producción
FROM node:18-alpine AS builder

WORKDIR /app

COPY package*.json ./
RUN npm ci --only=production

COPY . .
RUN npm run build

# Etapa de producción
FROM nginx:alpine

COPY --from=builder /app/dist /usr/share/nginx/html
COPY nginx.conf /etc/nginx/nginx.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

Nota sobre Alpine Linux: Se utiliza Alpine Linux como base de imagen porque es extremadamente ligero (solo ~~5MB~~) ~~comparado con imágenes estándar de Ubuntu (~~70MB). Esto resulta en:

- **Tiempos de descarga más rápidos**
- **Menor uso de espacio en disco**
- **Inicio más rápido de contenedores**
- **Menor superficie de ataque** (menos paquetes instalados)

Paso 2: Configurar Nginx

nginx.conf (archivo de configuración optimizado):

```
events {
    worker_connections 1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    sendfile        on;
    keepalive_timeout  65;

    server {
        listen       80;
        server_name  localhost;

        location / {
            root   /usr/share/nginx/html;
            index  index.html index.htm;
            try_files $uri $uri/ /index.html;
        }

        location /static/ {
            expires 1y;
            add_header Cache-Control "public, immutable";
        }
    }
}
```

Paso 3: Construir Imagen de Producción

1. **Ejecutar** el siguiente comando:

   ```
   docker build -t edustreaming:production .
   ```

2. **Esperar** a que se complete la construcción
3. **Verificar** que la imagen se haya creado:

   ```
   docker images
   ```

Paso 4: Ejecutar en Producción

1. **Ejecutar** el siguiente comando:

```
docker run -d -p 80:80 --name edustreaming-prod edustreaming:production
```

2. **Verificar** que el contenedor esté ejecutándose:

```
docker ps
```

3. **Acceder** a la aplicación en: http://localhost

Paso 5: Verificar el Despliegue

1. **Abrir** el navegador
2. **Navegar** a: http://localhost
3. **Verificar** que la aplicación cargue correctamente
4. **Probar** todas las funcionalidades principales
5. **Verificar** que los assets se carguen desde el cache

Comandos Útiles de Docker

Comandos Básicos

```
# Ver contenedores en ejecución
docker ps

# Ver todas las imágenes
docker images

# Ver logs de un contenedor
docker logs <container_name>

# Detener un contenedor
docker stop <container_name>

# Eliminar un contenedor
docker rm <container_name>

# Eliminar una imagen
docker rmi <image_name>
```

Comandos de Desarrollo

```
# Construir imagen
docker-compose build

# Iniciar servicios
docker-compose up

# Iniciar en segundo plano
docker-compose up -d
```

```
# Detener servicios
docker-compose down

# Ver logs
docker-compose logs

# Reconstruir y reiniciar
docker-compose up --build
```

Comandos de Limpieza

```
# Eliminar contenedores detenidos
docker container prune

# Eliminar imágenes no utilizadas
docker image prune

# Eliminar volúmenes no utilizados
docker volume prune

# Limpieza completa
docker system prune -a
```

Consideraciones para Producción

Seguridad

1. **Actualizar regularmente** las imágenes base
2. **Usar imágenes oficiales** de Docker Hub
3. **Configurar firewall** para limitar acceso
4. **Implementar HTTPS** con certificados SSL
5. **Usar secrets** para información sensible

Monitoreo

1. **Configurar logs** centralizados
2. **Implementar health checks** en Docker
3. **Monitorear recursos** (CPU, memoria, disco)
4. **Configurar alertas** para fallos
5. **Usar herramientas** como Prometheus + Grafana

Escalabilidad

1. **Usar Docker Swarm** o Kubernetes para orquestación
2. **Implementar load balancing** con múltiples instancias
3. **Configurar auto-scaling** basado en métricas
4. **Usar bases de datos** externas para persistencia
5. **Implementar CDN** para assets estáticos

Backup y Recuperación

1. **Backup regular** de volúmenes de datos

2. **Versionado** de imágenes Docker
3. **Documentación** de procedimientos de recuperación
4. **Testing** de procedimientos de backup
5. **Almacenamiento** de backups en ubicaciones seguras

Optimización de Rendimiento

1. **Usar multi-stage builds** para imágenes más pequeñas
2. **Optimizar layers** de Docker para mejor caching
3. **Configurar Nginx** para compresión y caching
4. **Usar imágenes Alpine** para menor tamaño
5. **Implementar lazy loading** en la aplicación

Conclusión

Esta guía proporciona una base sólida para implementar EduStreaming usando Docker y Vercel. La combinación de containerización con Docker y el despliegue simplificado en Vercel ofrece una solución robusta, escalable y fácil de mantener para plataformas de streaming educativo.

Próximos Pasos Recomendados:

1. **Implementar CI/CD** con GitHub Actions
2. **Agregar testing automatizado** en el pipeline
3. **Configurar monitoreo** con herramientas profesionales
4. **Implementar backup automatizado** de datos
5. **Documentar procedimientos** de operación

Recursos Adicionales:

- Documentación oficial de Docker
- Guía de Vercel
- Mejores prácticas de Docker
- Optimización de imágenes Docker

Nota: Esta guía está diseñada para entornos educativos y de desarrollo. Para implementaciones en producción a gran escala, se recomienda consultar con especialistas en DevOps y seguir las mejores prácticas de la industria.