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

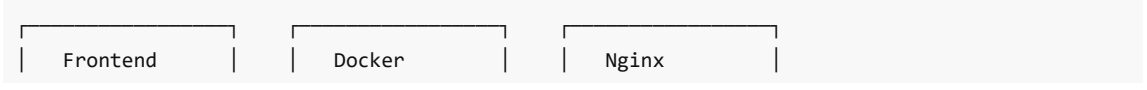
Requerimientos No Funcionales:

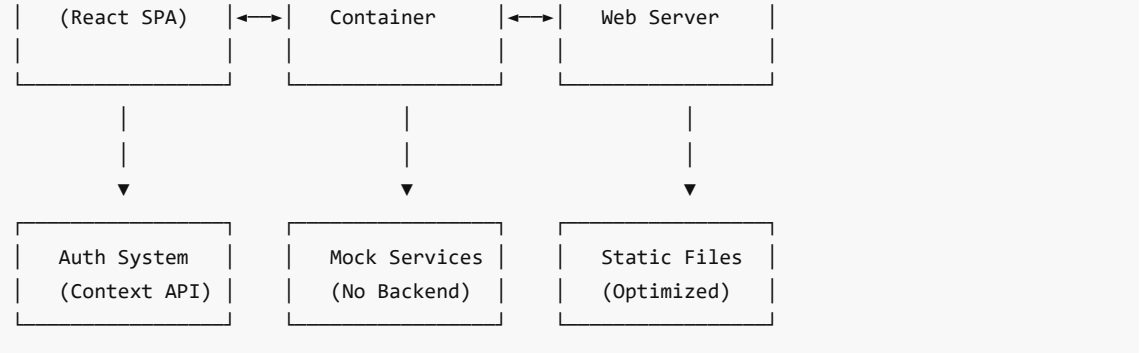
- **Escalabilidad**: Soporte para 500+ conexiones simultáneas
- **Rendimiento**: Tiempo de carga < 3 segundos
- **Disponibilidad**: 99.9% de uptime
- **Seguridad**: Autenticación segura y encriptación de datos
- **Usabilidad**: Interfaz intuitiva y responsive design
- **Compatibilidad**: Funcionamiento 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 de API**: < 500ms
- **Throughput**: 500+ usuarios simultáneos
- **Disponibilidad**: 99.9% uptime
- **Tiempo de Recuperación**: < 5 minutos

Consideraciones de Seguridad:

- **Autenticación JWT**: Tokens seguros para sesiones
- **HTTPS**: Encriptación de datos en tránsito
- **CORS**: Configuración de políticas de origen cruzado
- **Validación**: Sanitización de inputs del usuario
- **Headers de Seguridad**: CSP, XSS Protection, etc.

Estrategia de Despliegue:

- **Desarrollo**: Hot reload con Docker Compose
- **Producción**: Multi-stage build optimizado
- **Monitoreo**: Health checks y logging centralizado
- **Escalabilidad**: Horizontal scaling con load balancer

Beneficios de la Implementación:

- **Para la Universidad**: Reducción de costos de infraestructura física
- **Para los Estudiantes**: Acceso flexible y contenido bajo demanda
- **Para los Profesores**: Herramientas avanzadas de enseñanza
- **Para la Institución**: Escalabilidad y mantenimiento simplificado

Lecciones Aprendidas:

- **Containerización**: Simplifica el despliegue y la escalabilidad
- **SPA Architecture**: Mejora la experiencia de usuario
- **Mock Services**: Permite desarrollo frontend independiente
- **Nginx Configuration**: Optimiza el rendimiento y la seguridad

Próximos Pasos del Proyecto:

1. **Fase 2**: Implementación de backend real (Node.js/Express)
2. **Fase 3**: Integración con base de datos (PostgreSQL/MongoDB)
3. **Fase 4**: Sistema de streaming real (WebRTC/RTMP)
4. **Fase 5**: Análisis de datos y machine learning

Tabla de Contenidos

Prerrequisitos del Sistema

Requisitos Mínimos:

- **Sistema Operativo**: Windows 10/11
- **RAM**: Mínimo 4GB (Recomendado: 8GB+)
- **Espacio en Disco**: 2GB libres
- **Procesador**: 64-bit con soporte para virtualización
- **Conexión a Internet**: Para descargar imágenes de Docker

Verificar Virtualización:

- **Windows**: Verificar que Hyper-V esté habilitado

Instalación de Docker Desktop

Paso 1: Descargar Docker Desktop

1. Visita: https://www.docker.com/products/docker-desktop/
2. Haz clic en "Download for Windows"
3. Descarga el archivo `Docker Desktop Installer.exe`

Paso 2: Instalar Docker Desktop

1. **Ejecutar como Administrador**: Haz clic derecho en el instalador y selecciona "Ejecutar como administrador"
2. **Aceptar términos**: Marca la casilla "I accept the terms" y haz clic en "Install"
3. **Configuración inicial**:
    - Use WSL 2 instead of Hyper-V (recomendado)
    - Add shortcut to desktop
    - Use Windows containers for Linux containers

Paso 3: Reiniciar el Sistema

- Reinicia tu computadora cuando se solicite
- Esto es necesario para que los cambios de virtualización tomen efecto

Paso 4: Configurar Docker Desktop

1. **Abrir Docker Desktop**: Busca "Docker Desktop" en el menú inicio
2. **Aceptar términos de servicio**: Lee y acepta los términos
3. **Configuración de recursos**:
    - Ve a Settings → Resources
    - **Memory**: Asigna al menos 4GB (recomendado: 6-8GB)
    - **CPUs**: Asigna al menos 2 cores
    - **Disk image size**: Al menos 60GB

Verificación de la Instalación

Verificar Docker Engine:

```
docker --version
# Debería mostrar: Docker version 24.x.x, build xxxxx
```

Verificar Docker Compose:

```
docker-compose --version
# Debería mostrar: Docker Compose version v2.x.x
```

Verificar que Docker esté funcionando:

```
docker run hello-world
# Debería mostrar: "Hello from Docker!"
```

Verificar Docker Desktop:

1. Abre Docker Desktop
2. Verifica que el estado sea "Running" (Verde)
3. Ve a la pestaña "Images" - debería estar vacía inicialmente

---

Despliegue en Vercel

¿Qué es Vercel?

Vercel es una plataforma de despliegue que permite desplegar aplicaciones web de forma rápida y sencilla. Es especialmente útil para aplicaciones React, Next.js y otros frameworks modernos. Vercel se integra directamente con GitHub para hacer despliegues automáticos cada vez que actualices tu código.

Paso 1: Crear Cuenta en Vercel

1. **Visitar Vercel**: Ve a https://vercel.com
2. **Registrarse**: Haz clic en "Sign Up" en la esquina superior derecha
3. **Elegir método de registro**: Selecciona "Continue with GitHub" para vincular directamente tu cuenta de GitHub
4. **Autorizar Vercel**: Permite que Vercel acceda a tu cuenta de GitHub

Paso 2: Preparar el Repositorio en GitHub

1. **Crear repositorio**: En GitHub, crea un nuevo repositorio llamado `edustreaming` (o el nombre que prefieras)
2. **Subir código**: Sube todo el código de tu proyecto al repositorio
3. **Verificar estructura**: Asegúrate de que el repositorio contenga:
   - `package.json` con las dependencias
   - `src/` con el código fuente
   - `public/` con archivos estáticos
   - `vite.config.js` para la configuración

Paso 3: Conectar Vercel con GitHub

1. **Iniciar nuevo proyecto**: En el dashboard de Vercel, haz clic en "New Project"
2. **Importar desde GitHub**: Selecciona "Import Git Repository"
3. **Seleccionar repositorio**: Busca y selecciona tu repositorio `edustreaming`
4. **Configurar proyecto**: Vercel detectará automáticamente que es un proyecto Vite/React

Paso 4: Configurar el Despliegue

1. **Framework Preset**: Vercel debería detectar automáticamente "Vite" como framework
2. **Root Directory**: Deja vacío (usar raíz del repositorio)
3. **Build Command**: Vercel usará automáticamente `npm run build`
4. **Output Directory**: Vercel usará automáticamente `dist`
5. **Install Command**: Vercel usará automáticamente `npm install`

Paso 5: Desplegar la Aplicación

1. **Deploy**: Haz clic en "Deploy" para iniciar el primer despliegue
2. **Esperar construcción**: Vercel construirá tu aplicación automáticamente
3. **Verificar logs**: Revisa los logs de construcción para asegurarte de que no hay errores
4. **URL de producción**: Una vez completado, obtendrás una URL como `https://edustreaming-xxx.vercel.app`

Paso 6: Despliegues Automáticos

Una vez configurado, Vercel desplegará automáticamente tu aplicación cada vez que:

1. **Push a main**: Hagas push a la rama principal
2. **Pull Request**: Crear un pull request (despliegue de preview)
3. **Merge**: Hacer merge de cambios a la rama principal

Monitoreo y Analytics

1. **Analytics**: Vercel proporciona analytics básicos de tu aplicación
2. **Speed Insights**: Métricas de rendimiento automáticas
3. **Web Vitals**: Core Web Vitals para SEO y UX
4. **Logs**: Logs de la aplicación en tiempo real

¿Por qué Usar Vercel para EduStreaming?

Vercel es la plataforma ideal para desplegar aplicaciones React como EduStreaming por las siguientes razones:

Simplicidad y Velocidad de Despliegue

- **Configuración cero**: Vercel detecta automáticamente que es una aplicación Vite/React
- **Despliegue en segundos**: Desde el push a GitHub hasta la aplicación en vivo en menos de 2 minutos
- **Sin configuración de servidor**: No necesitas configurar EC2, Load Balancers, o bases de datos

Optimización para Frontend

- **Edge Computing**: Tu aplicación se ejecuta en el edge más cercano al usuario
- **CDN integrado**: Contenido estático servido desde múltiples ubicaciones globales
- **Compresión automática**: Gzip/Brotli habilitados automáticamente
- **Caching inteligente**: Cache de archivos estáticos optimizado para React

Integración Perfecta con GitHub

- **Despliegues automáticos**: Cada push a main genera un nuevo despliegue
- **Preview deployments**: Cada Pull Request genera una URL única para testing
- **Rollback fácil**: Puedes volver a cualquier versión anterior con un clic

Vercel vs AWS: ¿Por qué Vercel es Mejor para EduStreaming?

Complejidad de Configuración

AWS (EC2 + S3 + CloudFront):

- Configurar instancia EC2 (tamaño, región, seguridad)
- Instalar Node.js, Nginx, PM2
- Configurar S3 para archivos estáticos
- Configurar CloudFront para CDN
- Configurar Load Balancer
- Configurar Auto Scaling Groups
- Configurar Route 53 para DNS
- **Tiempo estimado**: 4-6 horas de configuración

Vercel:

- Conectar repositorio GitHub
- Hacer clic en "Deploy"
- **Tiempo estimado**: 2 minutos

Costo para Aplicaciones Pequeñas/Medianas

AWS:

- EC2 t3.micro: ~$8-10/mes
- S3 storage: ~$1-2/mes
- CloudFront: ~$1-3/mes
- Route 53: ~$0.50/mes

- **Total**: ~$10-15/mes mínimo

Vercel:

- Plan gratuito: 100GB bandwidth, deployments ilimitados
- **Total**: $0/mes para proyectos pequeños

Mantenimiento y Monitoreo

AWS:

- Actualizar instancias manualmente
- Monitorear logs con CloudWatch
- Configurar alertas
- Gestionar certificados SSL
- Actualizar dependencias del servidor

Vercel:

- Actualizaciones automáticas
- Logs integrados en dashboard
- SSL automático
- Zero maintenance

Escalabilidad

AWS:

- Configurar Auto Scaling Groups
- Configurar Load Balancers
- Gestionar múltiples instancias
- Configurar health checks

Vercel:

- Escalado automático e infinito
- Edge computing global
- Sin configuración adicional

Desarrollo y Testing

AWS:

- Crear staging environments manualmente
- Configurar CI/CD con GitHub Actions
- Gestionar múltiples entornos

Vercel:

- Preview deployments automáticos por PR
- Staging automático
- Integración nativa con GitHub

Casos Donde AWS Sería Mejor

AWS sería más apropiado si necesitaras:

- **Backend complejo**: APIs con base de datos, microservicios
- **Procesamiento pesado**: Machine Learning, video processing
- **Compliance específico**: HIPAA, SOX, requerimientos gubernamentales

- **Control total**: Configuración específica de servidor, software personalizado

Conclusión para EduStreaming

Para una aplicación frontend como EduStreaming que:

- Es principalmente una SPA (Single Page Application)
- No requiere backend complejo
- Necesita despliegue rápido y confiable
- Debe ser accesible globalmente
- Requiere actualizaciones frecuentes

Vercel es la opción óptima porque ofrece simplicidad, velocidad, costo-efectividad y todas las características necesarias sin la complejidad de AWS.

Configuración del Proyecto

Estructura del Proyecto:

```
edustreaming/
├── src/                    # Código fuente de la aplicación
├── public/                 # Archivos públicos
├── package.json           # Dependencias del proyecto
├── vite.config.js         # Configuración de Vite
├── Dockerfile             # Configuración para producción
├── Dockerfile.dev         # Configuración para desarrollo
├── docker-compose.yml     # Orquestación de servicios
├── nginx.conf             # Configuración del servidor web
├── deploy.bat             # Script de despliegue (Windows)
└── .dockerignore          # Archivos a ignorar en Docker
```

Archivos de Configuración Docker:

Dockerfile (Producción):

- **Multi-stage build** para optimización
- **Node.js 18 Alpine** para construcción
- **Nginx Alpine** para servidor web
- **Usuario no-root** para seguridad
- **Sobre Alpine Linux**: Alpine es una distribución minimalista (musl) que reduce el peso de la imagen; se utiliza Node.js 18 en la fase de build para compilar y empaquetar la SPA de forma eficiente.

Dockerfile.dev (Desarrollo):

- **Hot reload** habilitado
- **Puerto 5173** para Vite
- **Volúmenes montados** para cambios en tiempo real

docker-compose.yml:

- **Servicio de producción** en puerto 3000
- **Servicio de desarrollo** en puerto 5173
- **Health checks** configurados
- **Redes personalizadas**

Despliegue de la Aplicación

Opción 1: Usando Scripts (Recomendado)

En Windows:

```
# Navegar al directorio del proyecto
cd C:\ruta\a\tu\proyecto\edustreaming

# Modo desarrollo (con hot reload)
deploy.bat dev

# Modo producción
deploy.bat prod

# Ver logs
deploy.bat logs

# Detener aplicación
deploy.bat stop
```

En Windows (PowerShell):

```
# Navegar al directorio del proyecto
cd C:\ruta\a\tu\proyecto\edustreaming

# Modo desarrollo (con hot reload)
.\deploy.bat dev

# Modo producción
.\deploy.bat prod

# Ver logs
.\deploy.bat logs

# Detener aplicación
.\deploy.bat stop
```

Opción 2: Usando Docker Compose Directamente

Modo Desarrollo:

```
# Construir y levantar en modo desarrollo
docker-compose --profile dev up -d edustreaming-dev

# La aplicación estará disponible en: http://localhost:5173
```

Modo Producción:

```
# Construir y levantar en modo producción
docker-compose up -d edustreaming

# La aplicación estará disponible en: http://localhost:3000
```

Proceso de Construcción:

1. **Descarga de imágenes base**: Docker descarga Node.js y Nginx
2. **Instalación de dependencias**: npm install ejecuta automáticamente
3. **Construcción de la aplicación**: Vite build genera archivos optimizados
4. **Configuración de Nginx**: Servidor web configurado para SPA
5. **Inicio del contenedor**: Aplicación lista para usar

Comandos Útiles

Comandos de Docker:

```
# Ver contenedores corriendo
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>

# Limpiar sistema Docker
docker system prune -a
```

Comandos de Docker Compose:

```
# Levantar servicios
docker-compose up -d

# Detener servicios
docker-compose down

# Ver logs
docker-compose logs -f
```

```
# Reconstruir servicios
docker-compose build --no-cache

# Ver estado de servicios
docker-compose ps
```

Comandos del Script de Despliegue:

```
# Desarrollo
deploy.bat dev

# Producción
deploy.bat prod

# Construir solo
deploy.bat build

# Detener
deploy.bat stop

# Limpiar
deploy.bat clean

# Ver logs
deploy.bat logs
```

Acceso a la Aplicación

URLs de Acceso:

Modo Desarrollo:

- **URL Principal**: http://localhost:5173
- **Características**:
  - Hot reload automático
  - Herramientas de desarrollo
  - Debugging habilitado
  - Cambios en tiempo real

Modo Producción:

- **URL Principal**: http://localhost:3000
- **Características**:
  - Optimizada para producción
  - Compresión gzip
  - Cache optimizado
  - Configuración de seguridad

Endpoints Adicionales:

Archivos Estáticos:

- **CSS**: http://localhost:3000/assets/[filename].css
- **JavaScript**: http://localhost:3000/assets/[filename].js
- **Imágenes**: http://localhost:3000/assets/[filename].png

Navegación en la Aplicación:

1. **Página Principal**: Catálogo de cursos y streams
2. **Búsqueda**: Funcionalidad de búsqueda avanzada
3. **Notificaciones**: Sistema de notificaciones
4. **Perfil**: Gestión de usuario
5. **Dashboard**: Panel de administración (para profesores/admin)

Monitoreo y Mantenimiento

Verificar Estado de la Aplicación:

```
# Estado de contenedores
docker-compose ps

# Uso de recursos
docker stats

# Logs en tiempo real
docker-compose logs -f
```

Actualizar la Aplicación:

```
# Detener servicios
docker-compose down

# Actualizar código
git pull origin main

# Reconstruir y levantar
docker-compose up -d --build
```

Backup y Restauración:

```
# Crear backup de volúmenes
docker run --rm -v edustreaming_data:/data -v $(pwd):/backup alpine tar czf
/backup/backup.tar.gz -C /data .

# Restaurar backup
docker run --rm -v edustreaming_data:/data -v $(pwd):/backup alpine tar xzf
/backup/backup.tar.gz -C /data
```

Soporte y Recursos

Documentación Oficial:

- **Docker**: https://docs.docker.com/
- **Docker Compose**: https://docs.docker.com/compose/
- **Vite**: https://vitejs.dev/
- **React**: https://reactjs.org/

Comunidad:

- **Docker Community**: https://forums.docker.com/
- **Stack Overflow**: https://stackoverflow.com/questions/tagged/docker
- **GitHub Issues**: Reportar problemas en el repositorio del proyecto

Recursos Adicionales:

- **Docker Hub**: https://hub.docker.com/
- **Best Practices**: https://docs.docker.com/develop/dev-best-practices/
- **Security**: https://docs.docker.com/engine/security/

Felicitaciones

Has configurado exitosamente Docker y desplegado la aplicación EduStreaming.

Recuerda:

- Usa `deploy.bat dev` para desarrollo
- Usa `deploy.bat prod` para producción
- Monitorea los logs regularmente
- Mantén Docker Desktop actualizado

¡Disfruta de tu aplicación!
