

Dockerfile para Frontend Nginx

Configuración de Contenedor para Aplicación Web Estática

Propósito General

Este Dockerfile define la configuración para containerizar la aplicación frontend estática utilizando Nginx como servidor web. Incluye configuración optimizada para Single Page Application (SPA), proxies reversos para APIs y WebSockets, y mejores prácticas de seguridad y rendimiento.

Estructura del Dockerfile

Imagen Base

Utiliza nginx:1.25-alpine como imagen base, proporcionando:

Nginx 1.25: Versión estable con soporte HTTP/2 y características modernas

Distribución Alpine: Imagen minimalista para reducir tamaño y superficie de ataque

Optimización: Configurado específicamente para aplicaciones web estáticas

Metadatos del Contenedor

Incluye labels descriptivos para:

Mantenimiento: Información del equipo de desarrollo

Descripción: Especifica que es el frontend estático para aplicación IoT

Versión: Control de versiones del contenedor

Gestión de Usuarios y Seguridad

Usuario No-Root

dockerfile

```
RUN addgroup -g 1001 -S nginx && \
    adduser -S nginx -u 1001
```

Crea grupo y usuario dedicado con IDs específicos

Previene ejecución con privilegios de root

Sigue principio de mínimo privilegio

Configuración de Permisos

dockerfile

```
RUN chown -R nginx:nginx /usr/share/nginx/html && \  
    chown -R nginx:nginx /var/log/nginx && \  
    chown -R nginx:nginx /var/cache/nginx && \  
    chown -R nginx:nginx /etc/nginx/conf.d
```

Establece permisos consistentes en todos los directorios críticos

Asegura que el usuario nginx tenga acceso necesario

Configuración de Nginx Personalizada

Configuración del Servidor

nginx

```
server {  
    listen 80;  
    server_name localhost;
```

Escucha en puerto 80 para tráfico HTTP

Configuración básica del servidor

Proxy Reverso para API

nginx

```
location /api/ {  
    proxy_pass http://iot-backend:3000/;  
    proxy_http_version 1.1;  
    proxy_set_header Upgrade $http_upgrade;  
    proxy_set_header Connection 'upgrade';  
    # ... headers adicionales  
}
```

Destino: Servicio backend en iot-backend:3000

Headers: Configuración para WebSockets y información de cliente

Función: Redirección transparente de llamadas API

Proxy para WebSockets

nginx

```
location /ws/ {  
    proxy_pass http://iot-backend:3000/;  
    proxy_http_version 1.1;  
    proxy_set_header Upgrade $http_upgrade;  
    proxy_set_header Connection "Upgrade";
```

```
# ... headers adicionales  
}
```

Protocolo: Soporte para upgrade a WebSocket

Headers: Específicos para conexiones persistentes

Función: Comunicación en tiempo real con backend

Headers de Seguridad

Protecciones Implementadas

nginx

```
add_header X-Frame-Options "SAMEORIGIN" always;  
add_header X-Content-Type-Options "nosniff" always;  
add_header X-XSS-Protection "1; mode=block" always;  
add_header Referrer-Policy "strict-origin-when-cross-origin" always;
```

X-Frame-Options: Previene clickjacking

X-Content-Type-Options: Evita MIME sniffing

X-XSS-Protection: Protección contra cross-site scripting

Referrer-Policy: Control de información de referencia

Optimización de Rendimiento

Compresión GZIP

nginx

```
gzip on;  
gzip_vary on;  
gzip_min_length 1024;  
gzip_types  
    text/plain  
    text/css  
    text/xml  
    text/javascript  
    application/javascript  
    application/xml+rss  
    application/json;
```

Activación: Compresión habilitada para tipos de archivo comunes

Umbral: Solo comprime archivos mayores a 1KB

Tipos: Incluye todos los formatos textuales comunes

Estrategia de Caching

nginx

```
location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg|woff|woff2|ttf|eot)$ {  
    expires 1y;  
    add_header Cache-Control "public, immutable";  
    try_files $uri =404;  
}
```

Recursos estáticos: Cache por 1 año con inmutabilidad

Hash único: Asume que archivos tienen hash en nombre para cache busting

nginx

```
location / {  
    try_files $uri $uri/ /index.html;  
    add_header Cache-Control "no-cache, no-store, must-revalidate";  
}
```

SPA Fallback: Redirige todas las rutas a index.html

Sin cache: Para HTML principal de la aplicación

Configuración Específica por Tipo de Archivo

Archivos de Configuración

nginx

```
location /config.json {  
    add_header Cache-Control "no-cache, no-store, must-revalidate";  
    try_files $uri =404;  
}
```

Sin cache: Configuración puede cambiar frecuentemente

Service Worker y Manifest

nginx

```
location /sw.js {  
    add_header Cache-Control "no-cache, no-store, must-revalidate";  
    try_files $uri =404;  
}
```

```
location /manifest.webmanifest {
```

```
    add_header Cache-Control "no-cache, no-store, must-revalidate";
    try_files $uri =404;
}
```

Actualizaciones inmediatas: Para funcionalidades PWA

Health Check

nginx

```
location /health {
    access_log off;
    return 200 "healthy\n";
    add_header Content-Type text/plain;
}
```

Endpoint simple: Respuesta inmediata sin procesamiento

Sin logging: Evita llenar logs con checks de salud

Configuración del Contenedor

Directorios de Cache

dockerfile

```
RUN mkdir -p /var/cache/nginx/client_temp && \
    mkdir -p /var/cache/nginx/proxy_temp && \
    # ... más directorios
    chown -R nginx:nginx /var/cache/nginx
```

Pre-creación: Garantiza que existan todos los directorios necesarios

Permisos: Configuración correcta de propiedad

Health Check del Contenedor

dockerfile

```
HEALTHCHECK --interval=30s --timeout=3s --start-period=5s --retries=3 \
    CMD wget --no-verbose --tries=1 --spider http://localhost/health || exit 1
```

Intervalo: Verificación cada 30 segundos

Comando: Usa wget para verificar endpoint de salud

Tolerancia a fallos: 3 reintentos antes de marcar como unhealthy

Comando de Inicio

dockerfile

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

Primer plano: Mantiene Nginx ejecutando en primer plano

Formato array: Evita problemas de shell interpretation

Características de Seguridad

Best Practices Implementadas

Usuario no-root: Ejecución con privilegios mínimos

Headers de seguridad: Protecciones contra vulnerabilidades web

Configuración segura: Sin información sensible en imagen

Permisos restringidos: Acceso limitado a archivos del sistema

Consideraciones de Producción

SSL/TLS: Debe manejarse en proxy manager externo

Rate limiting: Podría añadirse para protección DDoS

Logging: Configuración para agregación centralizada

Monitoring: Integración con sistemas de monitoreo

Integración con el Ecosistema

Conectividad con Backend

DNS interno: Usa nombre de servicio iot-backend

Puerto interno: Conexión al puerto 3000 del backend

Red docker: Comunicación a través de la red bridge

Compatibilidad con SPA

Routing client-side: Todas las rutas redirigen a index.html

API transparente: Proxy reverso para llamadas a backend

WebSockets: Soporte para comunicación en tiempo real

Esta configuración proporciona un servidor web optimizado, seguro y listo para producción que sirve eficientemente la aplicación frontend mientras maneja proxying inteligente hacia los servicios backend.

