Pràctiques (reptes) i projecte de Docker i Apache HTTP Server

ÍNDEX

---

1. INTRODUCCIÓ I OBJECTIUS

Què aprendràs en aquesta pràctica?

En aquesta pràctica, aprendràs a utilitzar Docker com a plataforma de contenidorització per desplegar i gestionar servidors web Apache. Docker és una tecnologia fonamental en el món DevOps actual que permet empaquetar aplicacions i les seves dependències en contenidors portables i reproducibles.

Objectius d'aprenentatge

Al finalitzar aquesta pràctica, hauràs adquirit coneixements sobre:

- **Docker**: Gestió de contenidors, imatges, volums i xarxes
- **Dockerfile**: Creació d'imatges personalitzades amb instruccions declaratives
- **Docker Compose**: Orquestració de múltiples contenidors amb fitxers YAML
- **Apache HTTP Server 2.4**: Configuració de servidors web professionals
- Mòduls d'Apache (SSL, rewrite, proxy, headers)
- Virtual Hosts (múltiples llocs web en un mateix servidor)
- HTTPS amb certificats SSL/TLS
- URL Rewriting amb mod_rewrite

- Logging personalitzat i redirecció de logs

Prerequisits

- Docker Desktop instal·lat i funcionant al vostre ordinador
- Editor de text (VS Code, Sublime Text, Notepad++, etc.)
- Coneixements bàsics de línia de comandes (terminal/cmd)
- Coneixements bàsics d'HTTP i servidors web

Verificació de l'entorn

Abans de començar, verifica que Docker està correctament instal·lat executant aquestes comandes al terminal:

```
# Verifica la versió de Docker
docker --version

# Verifica que el daemon de Docker està funcionant
docker ps

# Comprova la versió de Docker Compose
docker compose version
```

Resultat esperat:

- Docker version 24.0.0 o superior
- Docker Compose version 2.20.0 o superior
- La comanda `docker ps` hauria de mostrar una taula buida (sense errors)

---

2. PART 1: FONAMENTS DE DOCKER (45 minuts)

2.1. Conceptes bàsics de Docker

Docker utilitza el concepte de **contenidors**, que són entorns aïllats que comparteixen el kernel del sistema operatiu però tenen els seus propis processos, sistema de fitxers i xarxa. Això és diferent de les màquines virtuals, que emulen un sistema complet amb el seu propi sistema operatiu.

Conceptes clau:

- **Imatge**: Plantilla de només lectura que conté el sistema de fitxers i la configuració necessària per executar una aplicació. És com una "fotografia" d'un sistema en un moment determinat.

- **Contenidor**: Instància en execució d'una imatge. És com un "procés" que s'executa de forma aïllada.
- **Dockerfile**: Fitxer de text amb instruccions per construir una imatge personalitzada.
- **Docker Hub**: Registre públic d'imatges Docker (com GitHub però per a imatges).
- **Volum**: Mecanisme per persistir dades més enllà del cicle de vida d'un contenidor.
- **Xarxa**: Permet la comunicació entre contenidors i amb l'exterior.

2.2. Primer contenidor: Apache bàsic

Anem a descarregar i executar el nostre primer contenidor amb Apache HTTP Server.

```
# Descarrega la imatge oficial d'Apache (httpd) versió 2.4.65
docker pull httpd:2.4.65

# Verifica que la imatge s'ha descarregat correctament
docker images

# Executa un contenidor amb Apache
docker run -d -p 8080:80 --name primer-apache-nomcognom httpd:2.4.65
```

Explicació de les opcions:

- `-d`: Executa el contenidor en segon pla (detached mode)
- `-p 8080:80`: Mapeja el port 8080 de l'amfitrió al port 80 del contenidor
- `--name primer-apache-nomcognom`: Assigna un nom al contenidor (més fàcil que usar l'ID)
- `httpd:2.4.65`: La imatge a utilitzar amb la seva etiqueta (tag) específica

Ara obre el navegador i accedeix a: `http://localhost:8080`

Hauries de veure la pàgina per defecte d'Apache: "It works!"

Fes una captura conforme funciona i on es vegi com has llançat la comanda docker

2.3. Comandes bàsiques per gestionar contenidors

```
# Llista els contenidors en execució
docker ps

# Llista tots els contenidors (inclosos els aturats)
docker ps -a

# Atura el contenidor
docker stop primer-apache-nomcognom

# Inicia el contenidor aturat
docker start primer-apache-nomcognom
```

```
# Reinicia el contenidor
docker restart primer-apache-nomcognom

# Veure els logs del contenidor
docker logs primer-apache-nomcognom

# Seguir els logs en temps real (com tail -f)
docker logs -f primer-apache-nomcognom

# Executar una comanda dins del contenidor
docker exec primer-apache-nomcognom ls /usr/local/apache2/htdocs

# Accedir a una shell interactiva dins del contenidor
docker exec -it primer-apache-nomcognom /bin/bash
```

Exercici pràctic 1.1:

Dins del contenidor, explora l'estructura de directoris d'Apache:

```
# Executa aquesta comanda per entrar al contenidor
docker exec -it primer-apache-nomcognom /bin/bash

# Dins del contenidor, explora aquests directoris
ls -la /usr/local/apache2/
ls -la /usr/local/apache2/conf/
cat /usr/local/apache2/conf/httpd.conf | grep DocumentRoot
```

2.4. Personalitzant el contingut web

Anem a modificar la pàgina per defecte utilitzant un **volum** per muntar contingut local dins del contenidor.

```
# Atura i elimina el contenidor anterior
docker stop primer-apache-nomcognom
docker rm primer-apache-nomcognom

# Crea un directori per al contingut web
mkdir ~/docker-apache-lab
cd ~/docker-apache-lab
mkdir html

# Crea una pàgina HTML personalitzada
Crea una carpeta que es digui html i dintre un fitxer index.html amb el
següent contingut:
'''html
<!DOCTYPE html>
<html lang="ca">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>El Meu Primer Apache amb Docker</title>
    <style>
```

```
        body {
            font-family: Arial, sans-serif;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            color: white;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }
        .container {
            text-align: center;
            background: rgba(255, 255, 255, 0.1);
            padding: 40px;
            border-radius: 15px;
            backdrop-filter: blur(10px);
        }
        h1 { font-size: 3em; margin: 0; }
        p { font-size: 1.2em; }
    </style>
</head>
<body>
    <div class="container">
        <h1> Apache + Docker</h1>
        <p>Pràctica de laboratori ASIX 2n</p>
        <p>Servidor Apache funcionant en un contenidor Docker!</p>
    </div>
</body>
</html>
'''

# Executa el contenidor amb un volum muntat
docker run -d -p 8080:80 --name apache-custom-nomcognom \
  -v "./html:/usr/local/apache2/htdocs" \
  httpd:2.4.65
```

Concepte important - Volums:

Quan utilitzem `-v` estem creant un **bind mount** que connecta un directori del nostre sistema amb un directori del contenidor. Qualsevol canvi que facis al directori local es reflectirà immediatament dins del contenidor, i viceversa.

Refresca el navegador a `http://localhost:8080` i veuràs la teva pàgina personalitzada!

2.5. Inspecció de contenidors

Docker proporciona informació detallada sobre els contenidors en execució:

```
# Informació detallada del contenidor (JSON)
docker inspect apache-custom-nomcognom

# Estadístiques d'ús de recursos en temps real
docker stats apache-custom-nomcognom
```

```
# Processos en execució dins del contenidor
docker top apache-custom-nomcognom
```

Repte 1

1. Crea una segona pàgina HTML anomenada `about.html` dins del directori `html` amb informació sobre tu (nom, curs, data).
2. Hauries de poder accedir-hi a `http://localhost:8080/about.html` sense reiniciar el contenidor.
3. Adjunta captures de pantalla de com ho has fet.

---

3. PART 2: DOCKERFILE - CREANT LES NOSTRES PRÒPIES IMATGES

3.1. Introducció al Dockerfile

Un **Dockerfile** és un fitxer de text que conté una sèrie d'instruccions per construir una imatge Docker. És com una recepta que especifica exactament com ha de ser el teu contenidor.

Instruccions principals d'un Dockerfile:

- `FROM` : Especifica la imatge base (punt de partida)
- `RUN` : Executa comandes durant la construcció de la imatge
- `COPY` / `ADD` : Copia fitxers del sistema host a la imatge
- `WORKDIR` : Estableix el directori de treball dins del contenidor
- `ENV` : Defineix variables d'entorn
- `EXPOSE` : Documenta els ports que el contenidor utilitza
- `CMD` : Comanda per defecte que s'executa quan s'inicia el contenidor
- `ENTRYPOINT` : Punt d'entrada fix per al contenidor
- `LABEL` : Afegeix metadades a la imatge

3.2. El nostre primer Dockerfile

Crea un directori nou per a aquesta part:

```
cd ~/docker-apache-lab
mkdir dockerfile-basic
cd dockerfile-basic
```

Crea un fitxer anomenat `Dockerfile` (sense extensió) amb aquest contingut:

```
# Utilitzem Apache 2.4.65 com a imatge base (Alpine és una versió lightweight)
FROM httpd:2.4.65-alpine

# Informació sobre qui manté aquesta imatge
LABEL maintainer="elteuemail@sapalomera.cat"
LABEL description="Apache HTTP Server personalitzat per a pràctiques ASIX"
LABEL version="1.0"

# Copia el contingut web personalitzat
COPY ./web-content/ /usr/local/apache2/htdocs/

# Exposa el port 80 (documentació, no fa res funcionalment)
EXPOSE 80

# La imatge base ja té definit el CMD per iniciar Apache
# CMD ["httpd-foreground"]
```

Crea el subdirectori web-content amb el contingut web i crea un fitxer index.html amb el següent contingut

```
<!DOCTYPE html>
<html lang="ca">
<head>
    <meta charset="UTF-8">
    <title>Imatge Docker Personalitzada</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background: #2c3e50;
            color: #ecf0f1;
            padding: 50px;
            text-align: center;
        }
        .badge {
            background: #e74c3c;
            padding: 10px 20px;
            border-radius: 20px;
            display: inline-block;
            margin: 20px 0;
        }
    </style>
</head>
<body>
    <h1>🚀 Imatge Docker Personalitzada</h1>
    <div class="badge">Construïda amb Dockerfile</div>
    <p>Aquesta pàgina s'ha copiat automàticament durant la construcció de la
imatge!</p>
</body>
</html>
EOF
```

3.3. Construir la imatge

Ara construïm la nostra pròpia imatge Docker:

```
# Construcció de la imatge amb un tag (etiqueta)
docker build -t apache-custom:v1.0 .

# Llista les imatges per veure la nova imatge
docker images

# Executa un contenidor amb la nostra imatge
docker run -d -p 8081:80 --name apache-dockerfile apache-custom:v1.0
```

Explicació del procés de construcció:

Quan executes `docker build`, Docker llegeix el Dockerfile línia per línia i crea **capes** (layers) per a cada instrucció. Cada capa és immutable i es pot reutilitzar (cache), la qual cosa fa que les construccions posteriors siguin molt més ràpides.

Accedeix a `http://localhost:8081` i veuràs la pàgina que es va copiar durant la construcció.

3.4. Multi-stage builds (construccions en múltiples etapes)

Les **multi-stage builds** són una tècnica avançada que permet crear imatges més petites i segures. S'utilitzen quan necessites eines de construcció que no vols en la imatge final.

Crea un nou directori:

```
cd ~/docker-apache-lab
mkdir dockerfile-multistage
cd dockerfile-multistage
```

Crea aquest Dockerfile:

```
# ETAPA 1: Construcció del contingut
FROM node:18-alpine AS builder

# Instal·la dependències per "construir" el lloc web
WORKDIR /build

# En una aplicació real, aquí compilaries TypeScript, minificaries CSS, etc.
# Per simplicitat, només crearem fitxers HTML
RUN mkdir -p dist && \
    echo '<!DOCTYPE html>' > dist/index.html && \
    echo '<html><head><title>Multi-stage Build</title></head>' >>
dist/index.html && \
    echo '<body style="font-family: Arial; background: #34495e; color: white;
padding: 50px; text-align: center;">' >> dist/index.html && \
    echo '<h1> Multi-Stage Build</h1>' >> dist/index.html && \
    echo '<p>Aquesta imatge s'\''ha creat utilitzant construcció en múltiples
etapes</p>' >> dist/index.html && \
```

```
    echo '<p style="background: #e67e22; padding: 10px; border-radius:
5px;">Imatge final: només Apache + fitxers estàtics (sense Node.js)</p>' >>
dist/index.html && \
    echo '</body></html>' >> dist/index.html

# ETAPA 2: Imatge final (només runtime)
FROM httpd:2.4.65-alpine

# Metadades
LABEL maintainer="elteuemail@sapalomera.cat"
LABEL description="Apache amb multi-stage build"

# Copia NOMÉS els fitxers construïts de l'etapa anterior
# No inclou Node.js ni les eines de construcció
COPY --from=builder /build/dist/ /usr/local/apache2/htdocs/

EXPOSE 80
```

Construeix i executa:

```
docker build -t apache-multistage:v1.0 .
docker run -d -p 8082:80 --name apache-multistage apache-multistage:v1.0
```

Avantatge clau: La imatge final és molt més petita perquè no inclou Node.js ni les eines de construcció, només Apache i els fitxers HTML resultants.

Comprova les mides:

```
docker images | grep apache
```

3.5. Best practices per a Dockerfile (2025)

Segons les recomanacions actuals de Docker (referències: Docker Documentation i community best practices 2025):

1. Utilitza imatges base oficials i específiques:

```
# MALAMENT: tag "latest" pot canviar amb el temps
FROM httpd:latest

# CORRECTE: versió específica i variant lightweight
FROM httpd:2.4.65-alpine
```

2. Minimitza el nombre de capes:

```
# MALAMENT: Cada RUN crea una nova capa
RUN apt-get update
RUN apt-get install -y curl
RUN apt-get install -y vim
```

```
# CORRECTE: Combina comandes relacionades
RUN apt-get update && apt-get install -y \
    curl \
    vim \
    && rm -rf /var/lib/apt/lists/*
```

3. Utilitza .dockerignore:

Crea un fitxer `.dockerignore` per excloure fitxers innecessaris:

```
cat > .dockerignore << 'EOF'
.git
.gitignore
README.md
node_modules
*.log
.env
.DS_Store
EOF
```

4. Ordena les instruccions per aprofitar la cache:

Les instruccions que canvien més freqüentment han d'anar al final del Dockerfile.

5. No executis contenidors com a root (seguretat):

```
FROM httpd:2.4.65-alpine

# Crea un usuari no privilegiat
RUN addgroup -g 1001 apache-user && \
    adduser -D -u 1001 -G apache-user apache-user

# Canvia el propietari dels fitxers necessaris
RUN chown -R apache-user:apache-user /usr/local/apache2/

# Canvia a l'usuari no privilegiat
USER apache-user

EXPOSE 80
CMD ["httpd-foreground"]
```

Repte 2

1. Crea un Dockerfile des de zero
2. Utilitza la imatge base `httpd:2.4.65-alpine`
3. Copia un directori `static-site` amb almenys 3 pàgines HTML (index.html, services.html, contact.html)
4. Inclou metadades apropiades (LABEL)

5. Segueix les best practices mencionades
6. Construeix la imatge amb el tag `apache-exercise-nomcognom:v1.0` i executa-la al port 8083.
7. Adjunta les captures de pantalla on es vegi clarament el funcionament.

---

4. PART 3: APACHE HTTP SERVER - CONFIGURACIÓ BÀSICA (45 minuts)

4.1. Estructura de configuració d'Apache

Apache HTTP Server 2.4.65 organitza la seva configuració en diversos fitxers i directoris. El fitxer principal és `httpd.conf`, però també es poden utilitzar fitxers `.htaccess` per a configuracions específiques de directori.

Directoris importants dins del contenidor oficial:

```
/usr/local/apache2/
├── conf/                # Fitxers de configuració
│   ├── httpd.conf       # Configuració principal
│   ├── extra/           # Configuracions addicionals
│   └── mime.types       # Tipus MIME
├── htdocs/              # Document root (contingut web)
├── logs/                # Logs d'accés i errors
└── modules/             # Mòduls d'Apache (.so)
```

4.2. Mòduls d'Apache

Apache utilitza una arquitectura modular. Els mòduls més importants que treballarem són:

- **mod_rewrite**: Reescriptura d'URLs
- **mod_ssl**: Suport HTTPS/SSL/TLS
- **mod_headers**: Manipulació de capçaleres HTTP
- **mod_proxy**: Proxy i proxy invers
- **mod_log_config**: Configuració de logging personalitzat

Verifica els mòduls carregats:

```
docker exec apache-custom httpd -M
```

4.3. Configuració personalitzada d'Apache

Crea un directori per treballar amb configuracions:

```
cd ~/docker-apache-lab
mkdir apache-config
cd apache-config
mkdir conf html logs
```

Crea una configuració personalitzada (`conf/httpd-custom.conf`):

```
# Carrega els mòduls necessaris
LoadModule authz_core_module modules/mod_authz_core.so
LoadModule dir_module modules/mod_dir.so
LoadModule mime_module modules/mod_mime.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule unixd_module modules/mod_unixd.so
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule headers_module modules/mod_headers.so

# Configuració del servidor
ServerName localhost
Listen 80

# Usuari i grup (per defecte en la imatge oficial)
User daemon
Group daemon

# Document Root
DocumentRoot "/usr/local/apache2/htdocs"

<Directory "/usr/local/apache2/htdocs">
    # Permet seguir enllaços simbòlics
    Options Indexes FollowSymLinks

    # Permet l'ús de .htaccess
    AllowOverride All

    # Permet accés a tothom
    Require all granted

    # Activa el mod_rewrite per aquest directori
    RewriteEngine On
</Directory>

# Index per defecte
DirectoryIndex index.html index.htm

# Tipus MIME
TypesConfig conf/mime.types

# Configuració de logging
ErrorLog "logs/error.log"
LogLevel warn

# Format de log personalitzat (Combined Log Format + temps de resposta)
```

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %D"
combined_with_time
CustomLog "logs/access.log" combined_with_time

# Capçalera de seguretat (exemple)
Header always set X-Content-Type-Options "nosniff"
Header always set X-Frame-Options "SAMEORIGIN"
```

Crea un fitxer `Dockerfile`:

```
FROM httpd:2.4.65-alpine

LABEL maintainer="elseu-email@exemple.cat"
LABEL description="Apache amb configuració personalitzada"

# Copia la configuració personalitzada
COPY conf/httpd-custom.conf /usr/local/apache2/conf/httpd.conf

# Copia el contingut web
COPY html/ /usr/local/apache2/htdocs/

EXPOSE 80

CMD ["httpd-foreground"]
```

Crea contingut HTML de prova (`html/index.html`):

```
<!DOCTYPE html>
<html lang="ca">
<head>
    <meta charset="UTF-8">
    <title>Apache Configurat</title>
    <style>
        body {
            font-family: 'Courier New', monospace;
            background: #1a1a2e;
            color: #eee;
            padding: 40px;
        }
        .config-box {
            background: #16213e;
            border-left: 5px solid #0f3460;
            padding: 20px;
            margin: 20px 0;
            border-radius: 5px;
        }
        code {
            background: #0f3460;
            padding: 2px 8px;
            border-radius: 3px;
            color: #feca57;
        }
    </style>
</head>
<body>
```

```
    <h1>Apache amb Configuració Personalitzada</h1>

    <div class="config-box">
        <h3>Funcionalitats actives:</h3>
        <ul>
            <li>Mòdul Rewrite activat</li>
            <li>Capçaleres de seguretat configurades</li>
            <li>Logging personalitzat amb temps de resposta</li>
            <li>.htaccess permès (AllowOverride All)</li>
        </ul>
    </div>

    <p>Comprova els logs amb: <code>docker logs apache-configured</code></p>
</body>
</html>
```

Construeix i executa:

```
docker build -t apache-configured:v1.0 .
docker run -d -p 8084:80 --name apache-configured \
  -v "$(pwd)/logs:/usr/local/apache2/logs" \
  apache-configured:v1.0
```

Nota important sobre volums: Hem muntat el directori `logs` per poder veure els logs d'Apache fora del contenidor.

Accedeix a `http://localhost:8084` i després comprova els logs:

```
# Veure els logs d'accés
cat logs/access.log

# Veure els logs d'error
cat logs/error.log
```

4.4. Fitxers .htaccess

Els fitxers `.htaccess` permeten configuracions específiques per directori sense modificar `httpd.conf`.

Crea `html/.htaccess`:

```
# Activar el motor de rewrite
RewriteEngine On

# Exemple: Redirigir .htm a .html
RewriteCond %{REQUEST_FILENAME}.html -f
RewriteRule ^(.*)$ $1.html [L]

# Protegir fitxers de configuració
<FilesMatch "^\.">
    Require all denied
</FilesMatch>
```

```
# Capçalera personalitzada per indicar que s'està utilitzant .htaccess
Header set X-Powered-By "Apache-ASIX-Lab"

# Desactivar listatge de directoris (si no hi ha index.html)
Options -Indexes
```

Reconstrueix la imatge per incloure el .htaccess:

```
docker stop apache-configured
docker rm apache-configured
docker build -t apache-configured:v1.1 .
docker run -d -p 8084:80 --name apache-configured \
  -v "$(pwd)/logs:/usr/local/apache2/logs" \
  apache-configured:v1.1
```

Repte 3

1. Modifica el fitxer `.htaccess` per:
   - Crea una regla que redirigeixi qualsevol petició a `/old-page` cap a `/new-page.html`
   - Afegeix una capçalera personalitzada `X-Lab-Version: 1.0` a totes les respostes
2. Reconstrueix la imatge i prova que funciona accedint a `http://localhost:8084/old-page`.
3. Adjunta captures de pantalla de tot el procés

---

5. PART 4: CONFIGURACIÓ AVANÇADA D'APACHE (60 minuts)

5.1. Virtual Hosts - Múltiples llocs web

Els **Virtual Hosts** permeten allotjar múltiples llocs web en un mateix servidor Apache. Hi ha dos tipus:

- **Name-based**: Múltiples dominis en una mateixa IP (més comú)
- **IP-based**: Cada domini té la seva pròpia IP

Crea un nou directori:

```
cd ~/docker-apache-lab
mkdir apache-vhosts
cd apache-vhosts
mkdir -p conf sites/site1 sites/site2 logs
```

Crea el contingut per al lloc 1 (`sites/site1/index.html`):

```
<!DOCTYPE html>
<html lang="ca">
<head>
    <meta charset="UTF-8">
    <title>Lloc 1 - Empresa Tech</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            color: white;
            padding: 50px;
            text-align: center;
        }
        .logo { font-size: 4em; }
    </style>
</head>
<body>
    <div class="logo">🚀</div>
    <h1>Empresa Tech Solutions</h1>
    <p>Virtual Host 1: site1.local</p>
    <p>Desenvolupament de software i consultoria IT</p>
</body>
</html>
```

Crea el contingut per al lloc 2 (`sites/site2/index.html`):

```
<!DOCTYPE html>
<html lang="ca">
<head>
    <meta charset="UTF-8">
    <title>Lloc 2 - Disseny Web</title>
    <style>
        body {
            font-family: 'Georgia', serif;
            background: linear-gradient(135deg, #f093fb 0%, #f5576c 100%);
            color: white;
            padding: 50px;
            text-align: center;
        }
        .logo { font-size: 4em; }
    </style>
</head>
<body>
    <div class="logo">🎨</div>
    <h1>Creative Design Studio</h1>
    <p>Virtual Host 2: site2.local</p>
    <p>Disseny gràfic i experiència d'usuari</p>
</body>
</html>
```

Crea la configuració d'Apache amb Virtual Hosts (`conf/httpd-vhosts.conf`):

```
# Carrega mòduls necessaris
LoadModule mpm_event_module modules/mod_mpm_event.so
LoadModule authz_core_module modules/mod_authz_core.so
```

```
LoadModule dir_module modules/mod_dir.so
LoadModule mime_module modules/mod_mime.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule unixd_module modules/mod_unixd.so
LoadModule headers_module modules/mod_headers.so
LoadModule vhost_alias_module modules/mod_vhost_alias.so

# Configuració bàsica del servidor
ServerName localhost
Listen 80

User daemon
Group daemon

TypesConfig conf/mime.types
DirectoryIndex index.html

# Format de log
LogFormat "%v %h %l %u %t \"%r\" %>s %b" vhost_combined
ErrorLog "logs/error.log"

# Virtual Host per defecte (quan no coincideix cap altre)
<VirtualHost *:80>
    ServerName localhost
    DocumentRoot "/usr/local/apache2/htdocs"
    ErrorLog "logs/default-error.log"
    CustomLog "logs/default-access.log" vhost_combined

    <Directory "/usr/local/apache2/htdocs">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>

# Virtual Host 1: site1.local
<VirtualHost *:80>
    ServerName site1.local
    ServerAlias www.site1.local
    DocumentRoot "/usr/local/apache2/sites/site1"

    ErrorLog "logs/site1-error.log"
    CustomLog "logs/site1-access.log" vhost_combined

    <Directory "/usr/local/apache2/sites/site1">
        Options -Indexes +FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    # Capçalera personalitzada per identificar el virtual host
    Header always set X-VHost "Site1"
</VirtualHost>

# Virtual Host 2: site2.local
<VirtualHost *:80>
    ServerName site2.local
```

```
    ServerAlias www.site2.local
    DocumentRoot "/usr/local/apache2/sites/site2"

    ErrorLog "logs/site2-error.log"
    CustomLog "logs/site2-access.log" vhost_combined

    <Directory "/usr/local/apache2/sites/site2">
        Options -Indexes +FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    Header always set X-VHost "Site2"
</VirtualHost>
```

Crea el Dockerfile:

```
FROM httpd:2.4.65-alpine

LABEL maintainer="elteuemail@sapalomera.cat"
LABEL description="Apache amb Virtual Hosts"

# Copia la configuració
COPY conf/httpd-vhosts.conf /usr/local/apache2/conf/httpd.conf

# Copia els llocs web
COPY sites/ /usr/local/apache2/sites/

EXPOSE 80

CMD ["httpd-foreground"]
```

Construeix i executa:

```
docker build -t apache-vhosts:v1.0 .
docker run -d -p 8085:80 --name apache-vhosts \
  -v "$(pwd)/logs:/usr/local/apache2/logs" \
  apache-vhosts:v1.0
```

Configuració del sistema per provar els Virtual Hosts:

Per accedir als virtual hosts utilitzant noms de domini locals, has d'editar el fitxer `/etc/hosts` (Linux/Mac) o `C:\Windows\System32\drivers\etc\hosts` (Windows):

```
# Afegeix aquestes línies al fitxer hosts
127.0.0.1 site1.local
127.0.0.1 site2.local
```

Ara pots accedir a: - `http://site1.local:8085` - `http://site2.local:8085`

Comprova els logs separats:

```
tail -f logs/site1-access.log
tail -f logs/site2-access.log
```

5.2. HTTPS amb SSL/TLS

Ara configurarem HTTPS utilitzant certificats auto-signats (per entorns de desenvolupament).

Crea un nou directori:

```
cd ~/docker-apache-lab
mkdir apache-ssl
cd apache-ssl
mkdir -p conf certs html logs
```

Generació de certificats SSL auto-signats:

Crearem un Dockerfile que generi els certificats durant la construcció:

```
FROM httpd:2.4.65-alpine

LABEL maintainer="elseu-email@exemple.cat"
LABEL description="Apache amb HTTPS/SSL"

# Instal·la OpenSSL (ja ve a Alpine, però assegurem dependències)
RUN apk add --no-cache openssl

# Crea el directori per als certificats
RUN mkdir -p /usr/local/apache2/certs

# Genera un certificat auto-signat
# Atenció: Això és NOMÉS per a desenvolupament! En producció usa Let's Encrypt
o un CA oficial
RUN openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
    -keyout /usr/local/apache2/certs/server.key \
    -out /usr/local/apache2/certs/server.crt \
    -subj "/C=ES/ST=Catalunya/L=Barcelona/O=ASIX Lab/OU=IT/CN=localhost"

# Ajusta permisos dels certificats
RUN chmod 600 /usr/local/apache2/certs/server.key && \
    chmod 644 /usr/local/apache2/certs/server.crt

# Copia la configuració d'Apache amb SSL
COPY conf/httpd-ssl.conf /usr/local/apache2/conf/httpd.conf

# Copia el contingut web
COPY html/ /usr/local/apache2/htdocs/

# Exposa els ports HTTP i HTTPS
EXPOSE 80 443

CMD ["httpd-foreground"]
```

Crea la configuració SSL (`conf/httpd-ssl.conf`):

```
# Mòduls necessaris
LoadModule mpm_event_module modules/mod_mpm_event.so
LoadModule authz_core_module modules/mod_authz_core.so
LoadModule dir_module modules/mod_dir.so
LoadModule mime_module modules/mod_mime.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule unixd_module modules/mod_unixd.so
LoadModule headers_module modules/mod_headers.so
LoadModule ssl_module modules/mod_ssl.so
LoadModule socache_shmcb_module modules/mod_socache_shmcb.so
LoadModule rewrite_module modules/mod_rewrite.so

ServerName localhost
User daemon
Group daemon

TypesConfig conf/mime.types
DirectoryIndex index.html

# Configuració de logging
ErrorLog "logs/error.log"
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
combined
CustomLog "logs/access.log" combined

# Cache SSL per millorar el rendiment
SSLSessionCache "shmcb:/usr/local/apache2/logs/ssl_scache(512000)"
SSLSessionCacheTimeout 300

# Virtual Host HTTP (port 80) - Redirigeix a HTTPS
<VirtualHost *:80>
    ServerName localhost
    DocumentRoot "/usr/local/apache2/htdocs"

    # Redirigeix tot el tràfic HTTP a HTTPS
    RewriteEngine On
    RewriteCond %{HTTPS} off
    RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [R=301,L]

    ErrorLog "logs/http-error.log"
    CustomLog "logs/http-access.log" combined
</VirtualHost>

# Virtual Host HTTPS (port 443)
<VirtualHost *:443>
    ServerName localhost
    DocumentRoot "/usr/local/apache2/htdocs"

    # Activa SSL
    SSLEngine on

    # Rutes als certificats
    SSLCertificateFile "/usr/local/apache2/certs/server.crt"
    SSLCertificateKeyFile "/usr/local/apache2/certs/server.key"

    # Protocols SSL/TLS moderns (desactiva versions antigues insegures)
```

```
    # Seguint les recomanacions de seguretat 2025
    SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1

    # Cipher suites segures (prioritza algoritmes moderns)
    SSLCipherSuite HIGH:!aNULL:!MD5:!RC4
    SSLHonorCipherOrder on

    # Capçaleres de seguretat HTTP
    Header always set Strict-Transport-Security "max-age=31536000;
includeSubDomains"
    Header always set X-Frame-Options "SAMEORIGIN"
    Header always set X-Content-Type-Options "nosniff"
    Header always set X-XSS-Protection "1; mode=block"

    <Directory "/usr/local/apache2/htdocs">
        Options -Indexes +FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog "logs/https-error.log"
    CustomLog "logs/https-access.log" combined
</VirtualHost>
```

Crea el contingut web (`html/index.html`):

```
<!DOCTYPE html>
<html lang="ca">
<head>
    <meta charset="UTF-8">
    <title>Connexió Segura HTTPS</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background: linear-gradient(135deg, #11998e 0%, #38ef7d 100%);
            color: white;
            padding: 50px;
            text-align: center;
        }
        .secure-badge {
            background: rgba(255, 255, 255, 0.2);
            backdrop-filter: blur(10px);
            padding: 30px;
            border-radius: 20px;
            display: inline-block;
            margin: 30px 0;
        }
        .lock { font-size: 5em; margin: 20px 0; }
        .info {
            background: rgba(0, 0, 0, 0.2);
            padding: 20px;
            border-radius: 10px;
            margin: 20px auto;
            max-width: 600px;
            text-align: left;
        }
```

```
    </style>
</head>
<body>
    <div class="secure-badge">
        <div class="lock"></div>
        <h1>Connexió Segura HTTPS</h1>
        <p>Apache 2.4.65 amb SSL/TLS activat</p>
    </div>

    <div class="info">
        <h3>Informació SSL/TLS:</h3>
        <ul>
            <li>Protocol: TLS 1.2 o superior</li>
            <li>Certificat: Auto-signat (només desenvolupament)</li>
            <li>HSTS: Activat (31536000 segons)</li>
            <li>Redirecció HTTP → HTTPS: Activa</li>
        </ul>
    </div>

    <p><small>El navegador mostrarà un avís perquè el certificat és auto-
signat.<br>
    En producció, utilitza certificats d'una CA reconeguda (Let's Encrypt,
etc.)</small></p>

    <script>
        // Mostra informació sobre la connexió (només funciona en HTTPS)
        if (location.protocol === 'https:') {
            console.log('Connexió segura HTTPS establerta');
        } else {
            console.log('Hauries de ser redirigit a HTTPS...');
        }
    </script>
</body>
</html>
```

Construeix i executa:

```
docker build -t apache-ssl:v1.0 .
docker run -d -p 8086:80 -p 8443:443 --name apache-ssl \
  -v "$(pwd)/logs:/usr/local/apache2/logs" \
  apache-ssl:v1.0
```

Proves:

1. Accedeix a `http://localhost:8086` → Hauries de ser redirigit a `https://localhost:8443`
2. El navegador mostrarà un avís de certificat no segur (normal amb certificats auto-signats)
3. Accepta l'avís i veuràs la pàgina amb HTTPS

Inspecciona els certificats:

```
# Veure informació del certificat
```

```
docker exec apache-ssl openssl x509 -in /usr/local/apache2/certs/server.crt -
text -noout
```

5.3. Mod_rewrite avançat - Reescriptura d'URLs

El **mod_rewrite** és un dels mòduls més potents d'Apache per manipular URLs. Utilitzarem regular expressions i condicions.

Crea `html/.htaccess` amb regles avançades:

```
RewriteEngine On

# 1. Redirecció de www a no-www (o viceversa)
RewriteCond %{HTTP_HOST} ^www\.(.*)$ [NC]
RewriteRule ^(.*)$ https://%1/$1 [R=301,L]

# 2. URLs amigables: /product/123 → /product.php?id=123
RewriteRule ^product/([0-9]+)$ product.php?id=$1 [L,QSA]

# 3. Eliminar l'extensió .php de les URLs
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME}.php -f
RewriteRule ^(.+)$ $1.php [L]

# 4. Redirecció d'URLs antigues a noves
RewriteRule ^old-blog/(.*)$ /new-blog/$1 [R=301,L]

# 5. Forçar HTTPS per a certes pàgines
RewriteCond %{HTTPS} off
RewriteCond %{REQUEST_URI} ^/admin [OR]
RewriteCond %{REQUEST_URI} ^/login
RewriteRule ^(.*)$ https://%{HTTP_HOST}/$1 [R=301,L]

# 6. Bloquejar accés basat en User-Agent (anti-bots)
RewriteCond %{HTTP_USER_AGENT} ^.*(bot|spider|crawler).* [NC]
RewriteRule ^api/ - [F,L]

# 7. Redirecció basada en la IP del client (geolocalització bàsica)
# Exemple: Si la IP comença per 192.168, redirigeix a versió local
RewriteCond %{REMOTE_ADDR} ^192\.168\.
RewriteRule ^$ /local-version.html [L]
```

5.4. Configuració de logging personalitzat

Apache permet configurar formats de log molt detallats. Modifica `conf/httpd-ssl.conf`:

```
# Format de log estàndard
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
combined

# Format de log amb temps de resposta en microsegons
LogFormat "%h %l %u %t \"%r\" %>s %b %D" with_response_time
```

```
# Format de log amb informació SSL
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{SSL_PROTOCOL}x\" \"%{SSL_CIPHER}x\""
ssl_combined

# Format JSON per a processament automàtic
LogFormat "{ \"timestamp\": \"%{%Y-%m-%dT%H:%M:%S}t\", \"client\": \"%h\",
\"request\": \"%r\", \"status\": %>s, \"bytes\": %b, \"referer\": \"%
{Referer}i\", \"user_agent\": \"%{User-Agent}i\", \"response_time_us\": %D }"
json_log

# Logging condicional: només registra errors 4xx i 5xx
CustomLog "logs/errors-only.log" combined expr=%{REQUEST_STATUS} >= 400

# Logging per tipus de petició
SetEnvIf Request_URI "\.jpg$" image-request
SetEnvIf Request_URI "\.png$" image-request
SetEnvIf Request_URI "\.gif$" image-request
CustomLog "logs/images.log" combined env=image-request

# Excloure certes peticions del log (com healthchecks)
SetEnvIf Request_URI "^/health$" dontlog
CustomLog "logs/access.log" combined env=!dontlog
```

Redirecció de logs a stdout/stderr (best practice per a contenidors):

En entorns Docker, és recomanable enviar els logs a stdout/stderr perquè Docker els pugui capturar:

```
# Envia els logs a la sortida estàndard del contenidor
ErrorLog /proc/self/fd/2
CustomLog /proc/self/fd/1 combined
```

Repte 4

1. Crea un Virtual Host amb les següents característiques:
   - Nom de domini: `secure-site-nomcognoms.local`
   - HTTPS obligatori (redirigeix HTTP a HTTPS)
   - Un fitxer `.htaccess` que:
     - Bloqueja l'accés al directori `/private`
     - Reescriu les URLs per eliminar `.html`
     - Afegeix una capçalera `X-Secure-Site: true`
2. Logs separats per a aquest virtual host 3, Adjunta les captures de pantalla on es vegi clarament la configuració.

---

6. PART 5: DOCKER COMPOSE - ORQUESTRACIÓ DE CONTENIDORS (45 minuts)

6.1. Introducció a Docker Compose

Docker Compose és una eina per definir i executar aplicacions Docker multi-contenidor. Utilitza fitxers YAML per configurar els serveis, xarxes i volums de l'aplicació.

Avantatges de Docker Compose:

- Definició declarativa de tota l'arquitectura
- Un sol fitxer `docker-compose.yml` per gestionar múltiples contenidors
- Facilita el desplegament, l'escalat i la gestió d'aplicacions complexes
- Xarxes automàtiques entre contenidors
- Gestió de dependències entre serveis

6.2. Estructura bàsica d'un docker-compose.yml

```yaml
services:
  # Cada servei és un contenidor
  nom-servei:
    image: imatge:tag          # Imatge a utilitzar
    # o
    build: ./directori         # Construir des d'un Dockerfile

    ports:
      - "host:container"       # Mapatge de ports

    volumes:
      - ./local:/container     # Volums

    environment:
      - VAR=valor              # Variables d'entorn

    depends_on:
      - altre-servei           # Dependències

    networks:
      - nom-xarxa              # Xarxes

networks:
  nom-xarxa:                   # Definició de xarxes personalitzades

volumes:
  nom-volum:                   # Definició de volums amb nom
```

6.3. Exemple pràctic: Apache + MySQL + phpMyAdmin

Crearem una aplicació completa amb 3 contenidors que es comuniquen entre ells.

Crea un nou directori:

```
cd ~/docker-apache-lab
mkdir docker-compose-stack
cd docker-compose-stack
mkdir -p apache/html apache/conf logs mysql-data
```

Estructura del projecte:

```
docker-compose-stack/
├── docker-compose.yml
├── apache/
│   ├── Dockerfile
│   ├── conf/
│   │   └── httpd.conf
│   └── html/
│       └── index.php
├── logs/
└── mysql-data/
```

Crea el Dockerfile per a Apache amb PHP (`apache/Dockerfile`):

```
FROM php:8.2-apache

LABEL maintainer="elteuemail@sapalomera.cat"
LABEL description="Apache amb PHP i extensió MySQL"

# Instal·la l'extensió mysqli per connectar a MySQL
RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli

# Activa mod_rewrite
RUN a2enmod rewrite

# Copia el contingut web
COPY html/ /var/www/html/

# Exposa el port 80
EXPOSE 80
```

Crea una pàgina PHP que es connecta a MySQL (`apache/html/index.php`):

```
<!DOCTYPE html>
<html lang="ca">
<head>
    <meta charset="UTF-8">
    <title>Stack LAMP amb Docker Compose</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            color: white;
```

```
            padding: 40px;
            margin: 0;
        }
        .container {
            max-width: 800px;
            margin: 0 auto;
            background: rgba(255, 255, 255, 0.1);
            backdrop-filter: blur(10px);
            padding: 40px;
            border-radius: 20px;
        }
        .status {
            padding: 20px;
            margin: 20px 0;
            border-radius: 10px;
            background: rgba(255, 255, 255, 0.2);
        }
        .success { border-left: 5px solid #2ecc71; }
        .error { border-left: 5px solid #e74c3c; }
        code {
            background: rgba(0, 0, 0, 0.3);
            padding: 2px 8px;
            border-radius: 3px;
        }
        table {
            width: 100%;
            margin: 20px 0;
            border-collapse: collapse;
        }
        th, td {
            padding: 12px;
            text-align: left;
            border-bottom: 1px solid rgba(255, 255, 255, 0.2);
        }
        th { background: rgba(0, 0, 0, 0.3); }
    </style>
</head>
<body>
    <div class="container">
        <h1>🐬 Docker Compose: Apache + MySQL + phpMyAdmin</h1>

        <?php
        // Configuració de la connexió a MySQL
        $host = 'mysql';  // Nom del servei en docker-compose.yml
        $user = 'root';
        $pass = 'rootpassword';
        $db = 'testdb';

        // Intenta connectar a MySQL
        $conn = @new mysqli($host, $user, $pass, $db);

        if ($conn->connect_error) {
            echo '<div class="status error">';
            echo '<h3>❌ Error de connexió a MySQL</h3>';
            echo '<p>' . $conn->connect_error . '</p>';
            echo '</div>';
        } else {
```

```
            echo '<div class="status success">';
            echo '<h3>✅ Connexió a MySQL establerta correctament</h3>';
            echo '<p><strong>Servidor:</strong> ' . $host . '</p>';
            echo '<p><strong>Base de dades:</strong> ' . $db . '</p>';

            // Mostra informació del servidor MySQL
            $version = $conn->query("SELECT VERSION() as version")-
>fetch_assoc();
            echo '<p><strong>Versió MySQL:</strong> ' . $version['version'] .
'</p>';

            // Crea una taula de prova i insereix dades
            $conn->query("CREATE TABLE IF NOT EXISTS visits (
                id INT AUTO_INCREMENT PRIMARY KEY,
                timestamp DATETIME,
                ip_address VARCHAR(50)
            )");

            $ip = $_SERVER['REMOTE_ADDR'];
            $conn->query("INSERT INTO visits (timestamp, ip_address) VALUES
(NOW(), '$ip')");

            // Mostra les últimes visites
            echo '<h3>📊 Últimes 5 visites registrades:</h3>';
            $result = $conn->query("SELECT * FROM visits ORDER BY timestamp
DESC LIMIT 5");

            if ($result->num_rows > 0) {
                echo '<table>';
                echo '<tr><th>ID</th><th>Data i hora</th><th>IP</th></tr>';
                while($row = $result->fetch_assoc()) {
                    echo '<tr>';
                    echo '<td>' . $row['id'] . '</td>';
                    echo '<td>' . $row['timestamp'] . '</td>';
                    echo '<td>' . $row['ip_address'] . '</td>';
                    echo '</tr>';
                }
                echo '</table>';
            }

            echo '</div>';
            $conn->close();
        }
        ?>

        <div class="status">
            <h3>🔗 Enllaços útils:</h3>
            <p><a href="http://localhost:8081" style="color: #fff;">📊
phpMyAdmin</a> (usuari: root, contrasenya: rootpassword)</p>
            <p><strong>Info del sistema:</strong></p>
            <ul>
                <li>PHP versió: <?php echo phpversion(); ?></li>
                <li>Apache versió: <?php echo apache_get_version(); ?></li>
                <li>Contenidor: <?php echo gethostname(); ?></li>
            </ul>
        </div>
```

```
        <div class="status">
            <h3>🧩 Conceptes demostrats:</h3>
            <ul>
                <li>✅ Multi-contenidor amb Docker Compose</li>
                <li>✅ Comunicació entre contenidors via xarxa Docker</li>
                <li>✅ Volums persistents per a MySQL</li>
                <li>✅ Variables d'entorn per a configuració</li>
                <li>✅ Dependències entre serveis (depends_on)</li>
            </ul>
        </div>
    </div>
</body>
</html>
```

Crea el fitxer `docker-compose.yml`:

```yaml
# Versió de Docker Compose (opcional en versions recents)
version: '3.9'

# Definició dels serveis (contenidors)
services:

  # Servei Apache amb PHP
  apache:
    # Construeix la imatge des del Dockerfile
    build:
      context: ./apache
      dockerfile: Dockerfile

    # Nom del contenidor
    container_name: lamp-apache

    # Reinicia automàticament si el contenidor falla
    restart: unless-stopped

    # Mapatge de ports: host:container
    ports:
      - "8090:80"

    # Volums: persistència i desenvolupament
    volumes:
      - ./apache/html:/var/www/html  # Codi PHP (bind mount)
      - ./logs:/var/log/apache2      # Logs d'Apache

    # Variables d'entorn
    environment:
      - TZ=Europe/Madrid

    # Depèn del servei MySQL (s'inicia després)
    depends_on:
      - mysql

    # Xarxa per comunicar-se amb altres serveis
    networks:
      - lamp-network
```

```
    # Límits de recursos (opcional però recomanat)
    deploy:
      resources:
        limits:
          cpus: '1.0'
          memory: 512M

  # Servei MySQL
  mysql:
    # Imatge oficial de MySQL (versió específica, no "latest")
    image: mysql:8.0.35

    container_name: lamp-mysql
    restart: unless-stopped

    # Variables d'entorn per configurar MySQL
    environment:
      - MYSQL_ROOT_PASSWORD=rootpassword
      - MYSQL_DATABASE=testdb
      - MYSQL_USER=appuser
      - MYSQL_PASSWORD=apppassword
      - TZ=Europe/Madrid

    # Volum amb nom per persistir dades de MySQL
    volumes:
      - mysql-data:/var/lib/mysql

    # Exposa el port MySQL (opcional, només si vols accedir des de l'host)
    ports:
      - "3306:3306"

    networks:
      - lamp-network

    # Healthcheck per verificar que MySQL està llest
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost", "-uroot", "-
prootpassword"]
      interval: 10s
      timeout: 5s
      retries: 3
      start_period: 30s

  # Servei phpMyAdmin (interfície web per gestionar MySQL)
  phpmyadmin:
    image: phpmyadmin:5.2-apache
    container_name: lamp-phpmyadmin
    restart: unless-stopped

    ports:
      - "8081:80"

    environment:
      - PMA_HOST=mysql              # Nom del servei MySQL
      - PMA_PORT=3306
      - PMA_USER=root
      - PMA_PASSWORD=rootpassword
```

```
      - UPLOAD_LIMIT=50M

    depends_on:
      - mysql

    networks:
      - lamp-network

# Definició de xarxes
networks:
  lamp-network:
    driver: bridge  # Xarxa bridge per defecte (permet comunicació entre
contenidors)

# Definició de volums amb nom (persistents)
volumes:
  mysql-data:
    driver: local  # Volum local gestionat per Docker
```

6.4. Executar l'stack amb Docker Compose

```
# Construeix i inicia tots els serveis en segon pla
docker compose up -d

# Veure els logs de tots els serveis
docker compose logs

# Seguir els logs en temps real
docker compose logs -f

# Veure només els logs d'un servei
docker compose logs -f apache

# Llista els contenidors de l'stack
docker compose ps

# Veure l'estat dels serveis
docker compose ps --format json

# Aturar tots els serveis (sense eliminar contenidors)
docker compose stop

# Iniciar els serveis aturats
docker compose start

# Reiniciar un servei específic
docker compose restart apache

# Aturar i eliminar tots els contenidors, xarxes (però manté els volums)
docker compose down

# Eliminar també els volums (⚠️ elimina les dades de MySQL!)
docker compose down -v

# Reconstruir les imatges i iniciar
```

```
docker compose up -d --build

# Executar una comanda en un servei
docker compose exec apache bash

# Escalar un servei (crear múltiples instàncies)
docker compose up -d --scale apache=3
```

Prova l'aplicació:

1. Accedeix a `http://localhost:8090` → Veuràs la pàgina PHP connectada a MySQL
2. Accedeix a `http://localhost:8081` → phpMyAdmin per gestionar la base de dades
3. Refresca diverses vegades `http://localhost:8090` per veure com es registren les visites

6.5. Xarxes i comunicació entre contenidors

Dins de Docker Compose, els contenidors poden comunicar-se entre ells utilitzant el **nom del servei** com a hostname.

```
# Accedeix al contenidor d'Apache
docker compose exec apache bash

# Des de dins d'Apache, pots fer ping a MySQL
ping mysql

# Pots connectar-te a MySQL des del contenidor d'Apache
mysql -h mysql -u root -prootpassword -e "SHOW DATABASES;"
```

Concepte important: Docker Compose crea automàticament una xarxa bridge privada on tots els serveis es poden comunicar. Els noms dels serveis es resolen automàticament com a hostnames via DNS intern.

6.6. Volums persistents vs Bind mounts

Bind mount (`./apache/html:/var/www/html`):

- Connecta un directori de l'host amb el contenidor
- Útil per a desenvolupament (canvis en temps real)
- El directori existeix al sistema de fitxers de l'host

Volum amb nom (`mysql-data:/var/lib/mysql`):

- Gestionat completament per Docker
- Millor rendiment (especialment en Windows/Mac)
- Persistent però independent del sistema de fitxers de l'host

- Ideal per a dades de producció

Gestió de volums:

```
# Llista tots els volums
docker volume ls

# Inspecciona un volum específic
docker volume inspect docker-compose-stack_mysql-data

# Elimina volums no utilitzats
docker volume prune

# Copia dades d'un volum (backup)
docker run --rm -v docker-compose-stack_mysql-data:/data -v $(pwd):/backup
alpine tar czf /backup/mysql-backup.tar.gz -C /data .
```

6.7. Variables d'entorn i fitxers .env

Docker Compose suporta fitxers `.env` per centralitzar la configuració:

Crea un fitxer `.env`:

```
# Configuració de MySQL
MYSQL_ROOT_PASSWORD=supersecret2025
MYSQL_DATABASE=production_db
MYSQL_USER=webapp
MYSQL_PASSWORD=webapp_secret_2025

# Ports
APACHE_PORT=8090
PHPMYADMIN_PORT=8081
MYSQL_PORT=3306

# Timezone
TZ=Europe/Madrid
```

Modifica `docker-compose.yml` per utilitzar variables:

```
version: '3.9'

services:
  apache:
    # ...
    ports:
      - "${APACHE_PORT}:80"
    environment:
      - TZ=${TZ}
    # ...

  mysql:
    # ...
    environment:
```

```
      - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
      - MYSQL_DATABASE=${MYSQL_DATABASE}
      - MYSQL_USER=${MYSQL_USER}
      - MYSQL_PASSWORD=${MYSQL_PASSWORD}
      - TZ=${TZ}
    ports:
      - "${MYSQL_PORT}:3306"
    # ...

  phpmyadmin:
    # ...
    ports:
      - "${PHPMYADMIN_PORT}:80"
    # ...
```

Important: Afegeix `.env` al `.gitignore` per no pujar credencials al repositori!

Repte 5

1. Modifica el `docker-compose.yml` per afegir un quart servei:
   - **Redis** (base de dades en memòria per a cache):
   - Imatge: `redis:7-alpine`
   - Port: 6379
   - Volum persistent per a les dades
   - Modifica `index.php` per utilitzar Redis com a comptador de visites. Ha de sortir el teu nom i cognom + nombre visites
   - Utilitza la llibreria PHP Redis: `pecl install redis`
2. Adjunta captures de pantalla on es vegi clarament la configuració i el resultat.

---

7. PART 6: PROJECTE FINAL

PROJECTE FINAL D'INTEGRACIÓ

Ara que has après tots els conceptes, és hora de demostrar els teus coneixements creant una aplicació completa des de zero.

Objectiu: Crear un stack Docker Compose amb els següents requisits:

Especificacions tècniques:

1. Servei Apache (Frontend)

- Utilitza un Dockerfile personalitzat basat en `httpd:2.4.65-alpine`
- Configura 2 Virtual Hosts:
- `frontend.local` → Lloc web principal amb HTML/CSS/JS
- `api.local` → API REST (pot ser un mock amb fitxers JSON)
- Activa i configura **mod_rewrite** per:
- URLs amigables (eliminar extensions .html)
- Redirigir HTTP a HTTPS
- Configura **HTTPS** amb certificats auto-signats utilitzant let's encrypt
- Implementa logging personalitzat amb format JSON
- Capçaleres de seguretat (HSTS, X-Frame-Options, CSP)

2. Servei MySQL (Backend Database)

- Versió específica: `mysql:8.0.35`
- Crea una base de dades amb almenys 2 taules:
- `users` (id, username, email, created_at)
- `articles` (id, user_id, title, content, published_at)
- Volum persistent per a les dades
- Healthcheck configurat
- Script d'inicialització amb dades de prova (utilitza volum amb fitxer `.sql`)

3. Servei Redis (Cache)

- Imatge: `redis:7-alpine`
- Volum persistent
- Configuració de límits de memòria

4. Servei phpMyAdmin (Eina d'administració)

- Interfície web per gestionar MySQL
- Port personalitzat

Requisits addicionals:

Seguretat:

- Utilitza variables d'entorn per a credencials (fitxer `.env`)
- No hardcodegis contrasenyes al codi
- Implementa restriccions d'accés via .htaccess per a certes rutes

Xarxes:

- Crea dues xarxes separades:
- `frontend-network` : només Apache
- `backend-network` : Apache, MySQL, Redis
- Això simula una arquitectura amb capes de seguretat

Volums:

- Utilitza volums amb nom per a MySQL i Redis
- Utilitza bind mounts per al codi i configuracions d'Apache
- Munta els logs d'Apache fora del contenidor

Documentació:

- Crea un fitxer `README.md` amb:
- Arquitectura del sistema (diagrama de text o descripció)
- Instruccions per desplegar l'aplicació
- URLs d'accés i credencials
- Explicació de les funcionalitats implementades

Funcionalitat aplicació web:

- Pàgina principal (`frontend.local`):
- Mostra estadístiques (nombre de visites utilitzant Redis)
- Llista els últims 5 articles de la base de dades MySQL
- Formulari per crear nous articles
- API REST (`api.local`):
- Endpoint GET `/api/articles` → Retorna JSON amb tots els articles
- Endpoint POST `/api/articles` → Crea un nou article
- Endpoint GET `/api/stats` → Retorna estadístiques (visites, usuaris, articles)

Estructura esperada del projecte

```
projecte-final/
├── docker-compose.yml
├── .env
├── .gitignore
├── README.md
├── apache/
│   ├── Dockerfile
│   ├── conf/
```

```
│   │   ├── httpd.conf
│   │   └── vhosts/
│   │       ├── frontend.conf
│   │       └── api.conf
│   ├── certs/
│   │   ├── (generats durant build)
│   ├── sites/
│   │   ├── frontend/
│   │   │   ├── index.php
│   │   │   ├── .htaccess
│   │   │   └── assets/ (css, js, images)
│   │   └── api/
│   │       ├── index.php
│   │       └── .htaccess
├── mysql/
│   └── init/
│       └── 01-schema.sql
├── logs/
└── (altres fitxers si cal)
```

Criteris d'avaluació

Has de realitzar totes aquestes accions més dos del bonus per considerar el projecte superat.

- L'aplicació s'inicia correctament amb `docker compose up -d` (1p)
- Es pot accedir als 2 Virtual Hosts configurant `/etc/hosts` (1p)
- HTTPS funciona correctament (amb avís de certificat auto-signat) (1p)
- Les peticions HTTP es redirigeixen a HTTPS (1p)
- La base de dades MySQL conté les taules i dades inicials (1p)
- Redis està funcionant i registra visites (1p)
- phpMyAdmin permet administrar la base de dades (1p)
- L'API retorna JSON vàlid (1p)
- Els logs es poden consultar des de l'host (0,5p)
- El README documenta tot el projecte clarament (0,5p)

Bonus (opcional)

- Implementa un healthcheck personalitzat per a cada servei (1p)
- Afegeix Nginx com a reverse proxy davant d'Apache (1p)
- Implementa rate limiting amb mod_evasive (1p)
- Crea un script de backup automàtic per a MySQL (1p)
- Afegeix un servei de monitoring (Prometheus/Grafana o similar) (1p)

LLIURAMENT DEL PROJECTE

Què has de lliurar:

1. **Directori complet del projecte final** (comprimit en .zip o .tar.gz)
2. **Captures de pantalla** que demostrin:
3. `docker compose ps` mostrant tots els serveis en execució
4. Navegador accedint a `https://frontend.local` amb el certificat SSL
5. phpMyAdmin mostrant les taules de la base de dades
6. Output de l'API en format JSON
7. Logs d'Apache mostrant peticions registrades
8. **README.md** amb tota la documentació

Format del nom del fitxer:

`COGNOM_NOM_ASIX2_Docker_Apache.zip`

ANNEX: COMANDES DE REFERÈNCIA RÀPIDA

Docker

```
# Imatges
docker images                          # Llista imatges
docker pull <imatge>:<tag>             # Descarrega imatge
docker rmi <imatge>                    # Elimina imatge
docker build -t <nom>:<tag> .          # Construeix imatge
docker tag <imatge>:<tag> <nou-nom>    # Etiqueta imatge

# Contenidors
docker ps                              # Llista contenidors actius
docker ps -a                           # Llista tots els contenidors
docker run [opcions] <imatge>          # Executa contenidor
docker start <contenidor>              # Inicia contenidor
docker stop <contenidor>               # Atura contenidor
docker restart <contenidor>            # Reinicia contenidor
docker rm <contenidor>                 # Elimina contenidor
docker exec -it <contenidor> <cmd>     # Executa comanda en contenidor
docker logs <contenidor>               # Veure logs
docker logs -f <contenidor>            # Seguir logs
docker inspect <contenidor>            # Informació detallada
docker stats                           # Estadístiques d'ús

# Volums
docker volume ls                       # Llista volums
docker volume create <nom>             # Crea volum
```

```
docker volume inspect <volum>          # Informació del volum
docker volume rm <volum>               # Elimina volum
docker volume prune                    # Elimina volums no utilitzats

# Xarxes
docker network ls                      # Llista xarxes
docker network create <nom>            # Crea xarxa
docker network inspect <xarxa>         # Informació de xarxa
docker network rm <xarxa>              # Elimina xarxa

# Neteja
docker system prune                    # Neteja contenidors, imatges i xarxes
no utilitzats
docker system prune -a                 # Neteja + imatges sense contenidors
docker system prune -a --volumes       # Neteja tot incloent volums
```

Docker Compose

```
# Gestió de serveis
docker compose up                      # Inicia serveis (primer pla)
docker compose up -d                   # Inicia serveis (segon pla)
docker compose up -d --build           # Reconstrueix i inicia
docker compose down                    # Atura i elimina contenidors
docker compose down -v                 # Atura, elimina contenidors i volums
docker compose start                   # Inicia serveis aturats
docker compose stop                    # Atura serveis
docker compose restart                 # Reinicia serveis
docker compose pause                   # Pausa serveis
docker compose unpause                 # Reprèn serveis

# Logs i informació
docker compose ps                      # Llista serveis
docker compose logs                    # Veure logs de tots els serveis
docker compose logs -f                 # Seguir logs
docker compose logs -f <servei>        # Logs d'un servei específic
docker compose top                     # Processos en execució

# Execució de comandes
docker compose exec <servei> <cmd>     # Executa comanda en servei
docker compose run <servei> <cmd>      # Executa comanda en nou contenidor

# Escalat
docker compose up -d --scale <servei>=<n>  # Escala servei a N instàncies

# Validació
docker compose config                  # Valida i mostra la configuració
docker compose config --services       # Llista serveis
docker compose version                 # Versió de Docker Compose
```

Apache (dins del contenidor)

```
# Configuració
httpd -V                               # Informació de compilació
```

```
httpd -M                                # Mòduls carregats
httpd -t                                # Testa la configuració
httpd -S                                # Mostra virtual hosts configurats
apachectl configtest                    # Testa configuració (igual que httpd -
t)
apachectl -k graceful                   # Reinicia gracefully (no talla
connexions)

# Logs
tail -f /usr/local/apache2/logs/access.log
tail -f /usr/local/apache2/logs/error.log
```

RECURSOS I REFERÈNCIES

Documentació oficial:

1. **Docker Documentation** (2025)
   https://docs.docker.com/
   Documentació oficial de Docker amb best practices actualitzades
2. **Docker Compose Documentation**
   https://docs.docker.com/compose/
   Guia completa de Docker Compose
3. **Apache HTTP Server 2.4 Documentation**
   https://httpd.apache.org/docs/2.4/
   Documentació oficial d'Apache 2.4.65
4. **Docker Hub - Official Images**
   https://hub.docker.com/_/httpd
   Imatge oficial d'Apache httpd

Versions utilitzades en aquesta pràctica:

- **Docker Desktop**: 4.47.0 o superior
- **Docker Engine**: 27.3.1 o superior
- **Docker Compose**: v2.20.0 o superior
- **Apache HTTP Server**: 2.4.65 (juliol 2025)
- **MySQL**: 8.0.35
- **PHP**: 8.2
- **Redis**: 7-alpine
- **phpMyAdmin**: 5.2

Conceptes clau que has après:

- Contenidorització amb Docker
- Creació d'imatges personalitzades amb Dockerfile
- Multi-stage builds per optimitzar imatges
- Configuració completa d'Apache HTTP Server
- Virtual Hosts per múltiples llocs web
- HTTPS/SSL/TLS
- Mod_rewrite per URLs amigables
- Logging personalitzat
- Docker Compose per orquestrar múltiples contenidors
- Xarxes i volums en Docker
- Best practices de seguretat i optimització (2025)

Pròxims passos que et recomano:

- Explorar Kubernetes per orquestració a escala
- Aprendre CI/CD amb Docker (GitLab CI, GitHub Actions, Jenkins)
- Estudiar Docker Swarm per clustering
- Implementar monitoring i logging (Prometheus, Grafana, ELK Stack)
- Aprofundir en seguretat de contenidors (Docker Scout, Trivy)

Bones pràctiques a recordar:

Seguretat: Mai utilitzis `:latest`, executa com a usuari no-root, escaneja vulnerabilitats
Imatges: Utilitza imatges Alpine quan sigui possible, minimitza capes
Logging: Envia logs a stdout/stderr per facilitar la gestió
Configuració: Utilitza variables d'entorn i fitxers .env
Documentació: Sempre documenta la teva infraestructura

Data de creació d'aquesta pràctica: Novembre 2025
Versió: 1.0
Feedback: [fbarragan@sapalomera.cat]