1. LAMP Stack

Folder Structure

```
/project-root
|-- /src
|   |-- /public
|   |   |-- index.php
|   |-- /includes
|   |   |-- header.php
|   |   |-- footer.php
|   |-- /config
|       |-- config.php
|-- /docker
|   |-- Dockerfile
|   |-- docker-compose.yml
|-- .env
|-- composer.json
|-- README.md
```

Key Files

- **Dockerfile**

```
FROM php:7.4-apache
COPY src/ /var/www/html/
RUN docker-php-ext-install mysqli pdo pdo_mysql
```

- **docker-compose.yml**

```
version: '3.1'
services:
  web:
    build: ./docker
    ports:
      - "80:80"
    volumes:
      - ./src:/var/www/html
  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: example
      MYSQL_DATABASE: exampledb
      MYSQL_USER: user
      MYSQL_PASSWORD: password
    volumes:
      - db_data:/var/lib/mysql
```

```
  volumes:
    db_data:
```

- **.env**

```
DB_HOST=db
DB_USER=user
DB_PASS=password
DB_NAME=exampledb
```

- **composer.json**

```
{
  "require": {
    "php": "^7.4",
    "monolog/monolog": "^2.0"
  }
}
```

2. LEMP Stack

Folder Structure

```
/project-root
|-- /src
|   |-- /public
|   |   |-- index.php
|   |-- /includes
|   |   |-- header.php
|   |   |-- footer.php
|   |-- /config
|       |-- config.php
|-- /docker
|   |-- Dockerfile
|   |-- docker-compose.yml
|-- .env
|-- composer.json
|-- README.md
```

Key Files

- **Dockerfile**

```

```

```
FROM php:7.4-fpm
RUN apt-get update && apt-get install -y libpng-dev
RUN docker-php-ext-install pdo_mysql
COPY src/ /var/www/html/
```

- **docker-compose.yml**

```
version: '3.1'
services:
  nginx:
    image: nginx:alpine
    ports:
      - "80:80"
    volumes:
      - ./src:/var/www/html
      - ./docker/nginx/default.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - php
  php:
    build: ./docker
    volumes:
      - ./src:/var/www/html
  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: example
      MYSQL_DATABASE: exampledb
      MYSQL_USER: user
      MYSQL_PASSWORD: password
    volumes:
      - db_data:/var/lib/mysql
volumes:
  db_data:
```

- **.env**

```
DB_HOST=db
DB_USER=user
DB_PASS=password
DB_NAME=exampledb
```

- **composer.json**

```
{
  "require": {
    "php": "^7.4",
```

```
    "monolog/monolog": "^2.0"
  }
}
```

3. WAMP Stack

Folder Structure

```
/project-root
|-- /src
|   |-- /public
|   |   |-- index.php
|   |-- /includes
|   |   |-- header.php
|   |   |-- footer.php
|   |-- /config
|       |-- config.php
|-- /docker
|   |-- Dockerfile
|   |-- docker-compose.yml
|-- .env
|-- composer.json
|-- README.md
```

Key Files

- **Dockerfile**

```
FROM php:7.4-apache
COPY src/ /var/www/html/
RUN docker-php-ext-install mysqli pdo pdo_mysql
```

- **docker-compose.yml**

```
version: '3.1'
services:
  web:
    build: ./docker
    ports:
      - "80:80"
    volumes:
      - ./src:/var/www/html
  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: example
```

```
      MYSQL_DATABASE: exampledb
      MYSQL_USER: user
      MYSQL_PASSWORD: password
    volumes:
      - db_data:/var/lib/mysql
volumes:
  db_data:
```

- **.env**

```
DB_HOST=db
DB_USER=user
DB_PASS=password
DB_NAME=exampledb
```

- **composer.json**

```
{
  "require": {
    "php": "^7.4",
    "monolog/monolog": "^2.0"
  }
}
```

4. MAMP Stack

Folder Structure

```
/project-root
|-- /src
|   |-- /public
|   |   |-- index.php
|   |-- /includes
|   |   |-- header.php
|   |   |-- footer.php
|   |-- /config
|       |-- config.php
|-- /docker
|   |-- Dockerfile
|   |-- docker-compose.yml
|-- .env
|-- composer.json
|-- README.md
```

Key Files

- **Dockerfile**

```
FROM php:7.4-apache
COPY src/ /var/www/html/
RUN docker-php-ext-install mysqli pdo pdo_mysql
```

- **docker-compose.yml**

```
version: '3.1'
services:
  web:
    build: ./docker
    ports:
      - "80:80"
    volumes:
      - ./src:/var/www/html
  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: example
      MYSQL_DATABASE: exampledb
      MYSQL_USER: user
      MYSQL_PASSWORD: password
    volumes:
      - db_data:/var/lib/mysql
volumes:
  db_data:
```

- **.env**

```
DB_HOST=db
DB_USER=user
DB_PASS=password
DB_NAME=exampledb
```

- **composer.json**

```
{
  "require": {
    "php": "^7.4",
    "monolog/monolog": "^2.0"
  }
}
```

5. MEAN Stack

Folder Structure

```
/project-root
|-- /server
|   |-- /config
|   |   |-- db.js
|   |-- /models
|   |   |-- user.js
|   |-- /routes
|   |   |-- userRoutes.js
|   |-- server.js
|-- /client
|   |-- /src
|       |-- /app
|           |-- app.module.ts
|           |-- app.component.ts
|-- /docker
|   |-- Dockerfile
|   |-- docker-compose.yml
|-- .env
|-- package.json
|-- README.md
```

Key Files

- **Dockerfile (Server)**

```
FROM node:14
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "server.js"]
```

- **Dockerfile (Client)**

```
FROM node:14
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 4200
CMD ["npm", "start"]
```

- **docker-compose.yml**

```
version: '3.8'
services:
  server:
    build: ./server
    ports:
      - "3000:3000"
    volumes:
      - ./server:/usr/src/app
  client:
    build: ./client
    ports:
      - "4200:4200"
    volumes:
      - ./client:/usr/src/app
  mongo:
    image: mongo
    ports:
      - "27017:27017"
```

- **.env**

```
MONGO_URI=mongodb://mongo:27017/yourdbname
```

- **package.json**

```
{
  "name": "mean-app",
  "version": "1.0.0",
  "description": "MEAN stack application",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "^4.17.1",
    "mongoose": "^5.9.10",
    "body-parser": "^1.19.0",
    "cors": "^2.8.5"


  }
}
```

6. MERN Stack

Folder Structure

```
/project-root
|-- /server
|   |-- /config
|   |   |-- db.js
|   |-- /models
|   |   |-- user.js
|   |-- /routes
|   |   |-- userRoutes.js
|   |-- server.js
|-- /client
|   |-- /src
|       |-- /components
|           |-- App.js
|-- /docker
|   |-- Dockerfile
|   |-- docker-compose.yml
|-- .env
|-- package.json
|-- README.md
```

Key Files

- **Dockerfile (Server)**

```
FROM node:14
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 5000
CMD ["node", "server.js"]
```

- **Dockerfile (Client)**

```
FROM node:14
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
```

- **docker-compose.yml**

```
version: '3.8'
services:
```

```
  server:
    build: ./server
    ports:
      - "5000:5000"
    volumes:
      - ./server:/usr/src/app
  client:
    build: ./client
    ports:
      - "3000:3000"
    volumes:
      - ./client:/usr/src/app
  mongo:
    image: mongo
    ports:
      - "27017:27017"
```

- **.env**

```
MONGO_URI=mongodb://mongo:27017/yourdbname
```

- **package.json**

```
{
  "name": "mern-app",
  "version": "1.0.0",
  "description": "MERN stack application",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "^4.17.1",
    "mongoose": "^5.9.10",
    "body-parser": "^1.19.0",
    "cors": "^2.8.5"
  }
}
```

7. MEVN Stack

Folder Structure

```
/project-root
|-- /server
|   |-- /config
|   |   |-- db.js
```

```
|   |-- /models
|   |   |-- user.js
|   |-- /routes
|   |   |-- userRoutes.js
|   |-- server.js
|-- /client
|   |-- /src
|       |-- /components
|           |-- HelloWorld.vue
|-- /docker
|   |-- Dockerfile
|   |-- docker-compose.yml
|-- .env
|-- package.json
|-- README.md
```

Key Files

- **Dockerfile (Server)**

```
FROM node:14
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 5000
CMD ["node", "server.js"]
```

- **Dockerfile (Client)**

```
FROM node:14
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 8080
CMD ["npm", "run", "serve"]
```

- **docker-compose.yml**

```
version: '3.8'
services:
  server:
    build: ./server
    ports:
      - "5000:5000"
    volumes:
```

```
      - ./server:/usr/src/app
  client:
    build: ./client
    ports:
      - "8080:8080"
    volumes:
      - ./client:/usr/src/app
  mongo:
    image: mongo
    ports:
      - "27017:27017"
```

- **.env**

```
MONGO_URI=mongodb://mongo:27017/yourdbname
```

- **package.json**

```
{
  "name": "mevn-app",
  "version": "1.0.0",
  "description": "MEVN stack application",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "^4.17.1",
    "mongoose": "^5.9.10",
    "body-parser": "^1.19.0",
    "cors": "^2.8.5"
  }
}
```

8. Django Stack

Folder Structure

```
/project-root
|-- /app
|   |-- /app
|   |   |-- __init__.py
|   |   |-- settings.py
|   |   |-- urls.py
|   |   |-- wsgi.py
|   |-- /migrations
|   |   |-- __init__.py
```

```
|   |-- /static
|   |-- /templates
|   |-- __init__.py
|   |-- admin.py
|   |-- apps.py
|   |-- models.py
|   |-- tests.py
|   |-- views.py
|-- /docker
|   |-- Dockerfile
|   |-- docker-compose.yml
|-- .env
|-- requirements.txt
|-- manage.py
|-- README.md
```

Key Files

- **Dockerfile**

```
FROM python:3.8
ENV PYTHONUNBUFFERED 1
WORKDIR /usr/src/app
COPY requirements.txt ./
RUN pip install -r requirements.txt
COPY . .
EXPOSE 8000
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

- **docker-compose.yml**

```
version: '3.8'
services:
  web:
    build: ./docker
    ports:
      - "8000:8000"
    volumes:
      - ./app:/usr/src/app
  db:
    image: postgres
    environment:
      POSTGRES_DB: mydatabase
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: mypassword
    volumes:
      - db_data:/var/lib/postgresql/data
volumes:
  db_data:
```

- **.env**

```
POSTGRES_DB=mydatabase
POSTGRES_USER=myuser
POSTGRES_PASSWORD=mypassword
```

- **requirements.txt**

```
Django>=3.0,<4.0
psycopg2-binary
```

9. Flask Stack

Folder Structure

```
/project-root
|-- /app
|   |-- /static
|   |-- /templates
|   |-- __init__.py
|   |-- routes.py
|   |-- models.py
|-- /docker
|   |-- Dockerfile
|   |-- docker-compose.yml
|-- .env
|-- requirements.txt
|-- wsgi.py
|-- README.md
```

Key Files

- **Dockerfile**

```
FROM python:3.8
ENV PYTHONUNBUFFERED 1
WORKDIR /usr/src/app
COPY requirements.txt ./
RUN pip install -r requirements.txt
COPY . .
EXPOSE 5000
CMD ["gunicorn", "--bind", "0.0.0.0:5000", "wsgi:app"]
```

- **docker-compose.yml**

```
version: '3.8'
services:
  web:
    build: ./docker
    ports:
      - "5000:5000"
    volumes:
      - ./app:/usr/src/app
  db:
    image: postgres
    environment:
      POSTGRES_DB: mydatabase
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: mypassword
    volumes:
      - db_data:/var/lib/postgresql/data
volumes:
  db_data:
```

- **.env**

```
POSTGRES_DB=mydatabase
POSTGRES_USER=myuser
POSTGRES_PASSWORD=mypassword
```

- **requirements.txt**

```
Flask
psycopg2-binary
gunicorn
```

10. Ruby on Rails Stack

Folder Structure

```
/project-root
|-- /app
|   |-- /controllers
|   |-- /models
|   |-- /views
|   |-- /assets
|-- /config
|-- /db
|-- /docker
```

```
|   |-- Dockerfile
|   |-- docker-compose.yml
|-- .env
|-- Gemfile
|-- README.md
```

Key Files

- **Dockerfile**

```
FROM ruby:2.7
RUN apt-get update -qq && apt-get install -y nodejs postgresql-client
WORKDIR /usr/src/app
COPY Gemfile* ./
RUN bundle install
COPY .
```

. EXPOSE 3000 CMD ["rails", "server", "-b", "0.0.0.0"]

```
- **docker-compose.yml**
```yaml
version: '3.8'
services:
  web:
    build: ./docker
    ports:
      - "3000:3000"
    volumes:
      - .:/usr/src/app
    depends_on:
      - db
  db:
    image: postgres
    environment:
      POSTGRES_DB: mydatabase
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: mypassword
    volumes:
      - db_data:/var/lib/postgresql/data
volumes:
  db_data:
```

- **.env**
```

```
POSTGRES_DB=mydatabase
POSTGRES_USER=myuser
POSTGRES_PASSWORD=mypassword
```

- **Gemfile**

```
source 'https://rubygems.org'
gem 'rails', '~> 6.0.3'
gem 'pg', '~> 1.1'
gem 'puma', '~> 4.1'
```

11. Spring Boot Stack

Folder Structure

```
/project-root
|-- /src
|   |-- /main
|       |-- /java
|           |-- /com
|               |-- /example
|                   |-- /demo
|                       |-- DemoApplication.java
|       |-- /resources
|           |-- application.properties
|-- /docker
|   |-- Dockerfile
|   |-- docker-compose.yml
|-- .env
|-- pom.xml
|-- README.md
```

Key Files

- **Dockerfile**

```
FROM openjdk:11
WORKDIR /usr/src/app
COPY target/demo-0.0.1-SNAPSHOT.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "app.jar"]
```

- **docker-compose.yml**

```
version: '3.8'
services:
  web:
    build: ./docker
    ports:
      - "8080:8080"
    volumes:
      - ./target:/usr/src/app
  db:
    image: postgres
    environment:
      POSTGRES_DB: mydatabase
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: mypassword
    volumes:
      - db_data:/var/lib/postgresql/data
volumes:
  db_data:
```

- **.env**

```
POSTGRES_DB=mydatabase
POSTGRES_USER=myuser
POSTGRES_PASSWORD=mypassword
```

- **pom.xml**

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
        </dependency>
```

```
    </dependencies>
</project>
```

12. .NET Stack

Folder Structure

```
/project-root
|-- /src
|   |-- /Controllers
|   |   |-- WeatherForecastController.cs
|   |-- /Models
|   |-- /Views
|   |-- /wwwroot
|   |-- appsettings.json
|   |-- Program.cs
|   |-- Startup.cs
|-- /docker
|   |-- Dockerfile
|   |-- docker-compose.yml
|-- .env
|-- project-name.csproj
|-- README.md
```

Key Files

- **Dockerfile**

```
FROM mcr.microsoft.com/dotnet/aspnet:5.0 AS base
WORKDIR /app
EXPOSE 80

FROM mcr.microsoft.com/dotnet/sdk:5.0 AS build
WORKDIR /src
COPY ["project-name.csproj", "./"]
RUN dotnet restore "project-name.csproj"
COPY . .
WORKDIR "/src/project-name"
RUN dotnet build "project-name.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "project-name.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "project-name.dll"]
```

- **docker-compose.yml**

```
version: '3.8'
services:
  web:
    build: ./docker
    ports:
      - "80:80"
    volumes:
      - ./src:/app
  db:
    image: mcr.microsoft.com/mssql/server
    environment:
      SA_PASSWORD: Your_password123
      ACCEPT_EULA: "Y"
    ports:
      - "1433:1433"
```

- **.env**

```
ConnectionStrings__DefaultConnection=Server=db;Database=YourDb;User
Id=sa;Password=Your_password123;
```

- **appsettings.json**

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=db;Database=YourDb;User
Id=sa;Password=Your_password123;"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

13. JAMstack

Folder Structure

```
/project-root
|-- /src
```

```
|   |-- /pages
|   |   |-- index.js
|   |-- /public
|-- /docker
|   |-- Dockerfile
|   |-- docker-compose.yml
|-- .env
|-- package.json
|-- README.md
```

Key Files

- **Dockerfile**

```
FROM node:14
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "run", "dev"]
```

- **docker-compose.yml**

```
version: '3.8'
services:
  web:
    build: ./docker
    ports:
      - "3000:3000"
    volumes:
      - ./src:/usr/src/app
```

- **.env**

```
NEXT_PUBLIC_API_URL=http://localhost:3000/api
```

- **package.json**

```
{
  "name": "jamstack-app",
  "version": "1.0.0",
  "description": "JAMstack application",
  "main": "index.js",
  "scripts": {
```

```
    "dev": "next dev",
    "build": "next build",
    "start": "next start"
  },
  "dependencies": {
    "next": "^10.0.0",
    "react": "^17.0.0",
    "react-dom": "^17.0.0"
  }
}
```

Common DevSecOps Considerations

1. **Environment Variables**: Use `.env` files to manage sensitive information and configuration.
2. **Dependency Management**: Ensure all dependencies are defined in `composer.json`, `package.json`, `requirements.txt`, `Gemfile`, etc.
3. **Container Security**: Use official base images and regularly update them. Scan images for vulnerabilities.
4. **Network Security**: Isolate services using Docker networks. Expose only necessary ports.
5. **Data Security**: Use encrypted storage for sensitive data and secure database connections.
6. **Access Control**: Implement RBAC (Role-Based Access Control) and least privilege principles.
7. **Monitoring and Logging**: Implement logging and monitoring for all services.
8. **CI/CD Pipelines**: Integrate security checks in CI/CD pipelines to ensure code quality and security.