🚀 Complete Guide to Full-Stack Application Cloud Deployment (GCP + Docker Dual Container Architecture)

Why Choose Dual Container Dual Port Architecture

In full-stack development, there are three main deployment modes:

A. Dual Container Dual Port Mode (Our Chosen Solution)

Frontend container (9191 port) ←——> Backend container (9090 port)

Advantages:

- Complete decoupling of frontend and backend services
- Independent scaling and maintenance
- Complies with microservice architecture principles
- Facilitates team collaboration

Disadvantages:

- Requires managing multiple containers
- Relatively complex configuration
- Requires multiple firewall rules

B. Single Container + Reverse Proxy Mode

Nginx container (80 port)
↓
Frontend static files + Backend API

Advantages:

- Simple configuration
- Single entry point
- Low maintenance cost

Disadvantages:

- Frontend and backend coupling
- Limited scalability
- Not suitable for large team collaboration

C. Single Container Frontend Proxy Mode

Frontend container (single port)
↓
API request proxy

Why We Choose Dual Container Dual Port

- Better scalability
- Aligns with modern microservice architecture
- Facilitates future feature expansion
- Supports independent deployment and updates

Key Concept Analysis

📦 1. Containerization

- Package programs and their runtime environment into independent executable units
- Analogy: You've made a "code delivery box" that runs anywhere

🌐 2. Port Mapping

- Frontend 9191 handles page display and static resources
- Backend 9090 processes API and database requests

🔐 3. Firewall Rules

- Controls external VM port access
- Follows the "**principle of least privilege**", only opening necessary ports

I. Prerequisites: Creating Google Cloud VM Instance (Compute Engine)

✅ Step 1: Open GCP Console

- Visit: https://console.cloud.google.com
- Login to your Google account
- Create new project

✅ Step 2: Enable Compute Engine API

- Navigate to "Compute Engine" → Enable API (First activation requires 1 minute wait)

✅ Step 3: Create VM Instance

- Click "Create Instance"

Recommended configuration:

- Name: demo-instance

- Region: us-central1
- Machine type: e2-micro (includes free tier)
- System image: Debian 11/12
- Check: Allow HTTP / HTTPS

Record the instance's external IP after creation

II. Connect to Server and Install Environment

✅ SSH Connection

Click "SSH" in browser, or use command line:

```
ssh -i ~/.ssh/google_compute_engine -o StrictHostKeyChecking=no your-userna
```

✅ Install Docker

```
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrir
sudo chmod a+r /etc/apt/keyrings/docker.asc
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/dock
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
```

✅ Install Git

```
sudo apt install git-all
```

III. Pull GitHub Project Code

✅ Clone Repository Using HTTPS

```
git clone https://github.com/your-username/logoinResgister.git
cd logoinResgister
```

IV. Project Structure

```
logoinResgister/
├── frontend/
│   ├── dockerfile
│   ├── package.json
│   └── index.js
├── backend/
│   ├── dockerfile
│   └── index.js
└── docker-compose.yml
```

V. Configure Docker Compose

docker-compose.yml Example

```
services:
  frontend:
    build:
      context: ./frontend
    ports:
      - "9191:9191"
    environment:
      - VITE_BACKEND_PATH=http://34.129.231.96:9090
  backend:
```

```
  build:
    context: ./backend
  ports:
    - "9090:9090"
```

VI. Configure GCP Firewall Rules

✅ Allow Access to Port 9191 (Frontend)

- Name: allow-9191
- Protocol: TCP
- Port: 9191
- Source: 0.0.0.0/0

✅ Allow Access to Port 9090 (Backend)

- Name: allow-9090
- Protocol: TCP
- Port: 9090
- Source: 0.0.0.0/0

VII. Start Services and Verify

✅ Start Containers

```
sudo docker compose up --build -d
```

✅ Check Running Status

```
sudo docker ps
```

✅ View Logs

```
sudo docker logs logoinresgister-frontend-1
sudo docker logs logoinresgister-backend-1
```

VIII. System Architecture Diagram (Text Representation)

```
User Browser  ↓
 Internet  ↓
GCP Firewall
┌──────────────────────────────────────┐
│      GCP Instance      │
│ ┌──────────────────────────────────┐ │
│ │ Frontend Container:9191 │ │
│ └──────────┬──────────────────────┘ │
│      ↓            │
│ ┌──────────────────────────────────┐ │
│ │ Backend Container:9090 │ │
│ └──────────────────────────────────┘ │
└──────────────────────────────────────┘
```

IX. Access Services

- Frontend access: http://34.129.231.96:9191
- Backend API: http://34.129.231.96:9090

X. Best Practice Recommendations

✅ Security

- Enable HTTPS

- Use JWT/Auth mechanism
- Limit backend API exposure

✅ Maintainability

- Log recording and backup
- Docker auto-restart policy
- Container monitoring (e.g., Docker stats)

✅ Scalability

- Use environment variable configuration
- Support load balancers
- Gradual migration to Kubernetes (GKE)

This deployment solution demonstrates the modularity, automation, and cloud-native characteristics of modern web applications, suitable for personal projects, small teams, and learning demonstrations.