📊 Báo Cáo Triển Khai Blog Microservices

📋 Thông Tin Dự Án

Thông tin	Chi tiết
Tên dự án	Blog Microservices
Kiến trúc	Microservices
Ngôn ngữ chính	TypeScript, JavaScript
Ngày báo cáo	24/6/2025
Người thực hiện	Nguyễn Quốc Khánh - 22127188

🎯 Mục Tiêu Triển Khai

Triển khai thành công hệ thống Blog Microservices bằng 3 phương pháp khác nhau:

1. **PM2** - Process Manager cho Node.js
2. **Docker Compose** - Containerization
3. **Railway** - Cloud Platform

🏗️ Kiến Trúc Hệ Thống

Các Microservices

- **API Gateway** (Port 8080): Điểm vào duy nhất cho client
- **User Service** (Port 3001): Quản lý người dùng và xác thực
- **Post Service** (Port 3002): Quản lý bài viết
- **Feed Service** (Port 3003): Tạo feed cá nhân hóa
- **Frontend** (Port 3000): Giao diện người dùng

Infrastructure Components

- **PostgreSQL**: Database cho User và Post services
- **Redis**: Cache cho Feed service
- **Kafka**: Message queue cho event-driven communication
- **Consul**: Service discovery và health checking

🚀 Phương Pháp 1: Triển Khai với PM2

📝 Mô Tả

PM2 là process manager cho Node.js applications, cho phép quản lý và monitor các processes một cách hiệu quả.

🔧 Cấu Hình

File cấu hình chính

```
script/ecosystem.dev.config.js
```

Các bước cấu hình:

Bước 1: Cài đặt dependencies

```
npm install
cd api-gateway && npm install
cd ../user-service && npm install
cd ../post-service && npm install
cd ../feed-service && npm install
cd ../Frontend && npm install
```

Bước 2: Khởi động infrastructure Tôi dùng các dịch vụ infrastructure cloud sau: redis: cloud.redis.io Apache Kafka: Confluent Cloud Postgres: Neon

Bước 3: Build TypeScript services

```
cd user-service && npm run build
cd ../post-service && npm run build
cd ../feed-service && npm run build
```

Bước 4: Khởi động với PM2

```
pm2 start script/ecosystem.dev.config.js
```

```
PS E:\Software_architecture\Lab1_3\blog-microservices> pm2 start script/ecosystem.dev.config.js
[PM2][WARN] Applications consul, api-gateway, user-service, post-service, feed-service, frontend-dev not running, starting...
[PM2] App [consul] launched (1 instances)
[PM2] App [api-gateway] launched (1 instances)
[PM2] App [user-service] launched (1 instances)
[PM2] App [post-service] launched (1 instances)
[PM2] App [frontend-dev] launched (1 instances)
[PM2] App [feed-service] launched (1 instances)
```

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
1	api-gateway	default	1.0.0	cluster	2236	0s	0	online	0%	54.5mb	quock	disabled
0	consul	default	N/A	fork	15684	0s	0	online	0%	65.8mb	quock	disabled
4	feed-service	default	1.0.0	cluster	4584	0s	0	online	0%	51.2mb	quock	disabled
5	frontend-dev	default	N/A	fork	16840	0s	0	online	0%	6.5mb	quock	disabled
3	post-service	default	1.0.0	cluster	17896	0s	0	online	0%	51.6mb	quock	disabled
2	user-service	default	1.0.0	cluster	13764	0s	0	online	0%	55.1mb	quock	disabled

```
next-server (v14.2.16)

> my-v0-project@0.1.0 dev
> next dev -p 3000

  ▲ Next.js 14.2.16
  - Local:        http://localhost:3000
  - Environments: .env

 ✓ Starting ...
 ✓ Ready in 1838ms
```

Kết Quả

PM2 Status

```
pm2 status
```

[

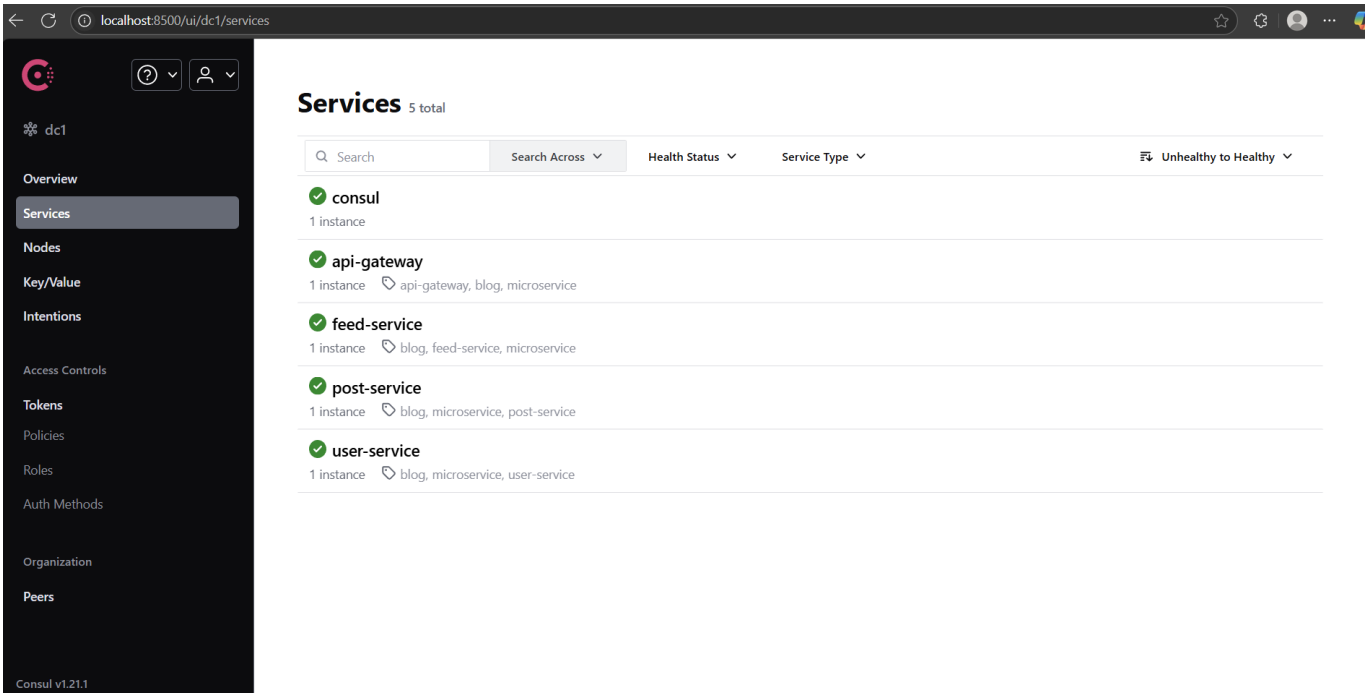
```
PS E:\Software_architecture\Lab1_3\blog-microservices> pm2 status
```

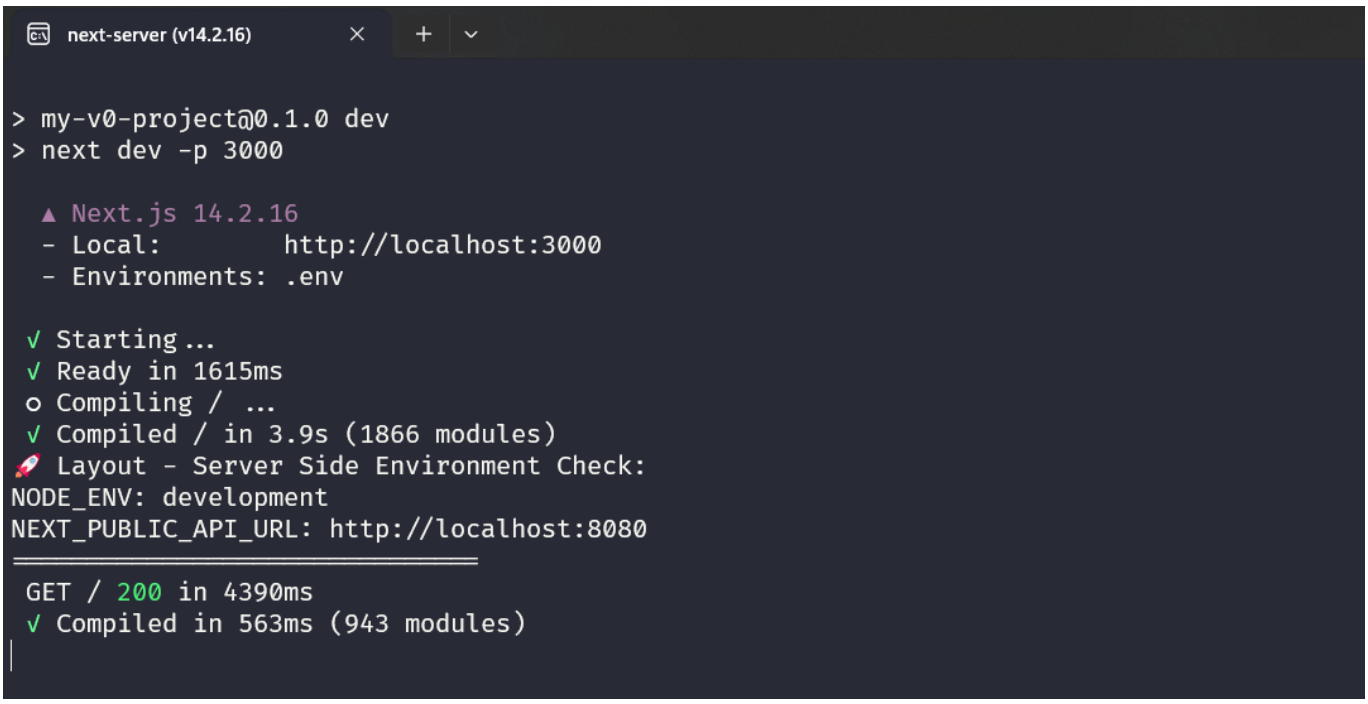
id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
1	api-gateway	default	1.0.0	cluster	2236	3m	0	online	0%	58.7mb	quock	disabled
0	consul	default	N/A	fork	15684	3m	0	online	0%	74.8mb	quock	disabled
4	feed-service	default	1.0.0	cluster	4584	3m	0	online	0%	52.1mb	quock	disabled
5	frontend-dev	default	N/A	fork	16840	3m	0	online	0%	6.6mb	quock	disabled
3	post-service	default	1.0.0	cluster	17896	3m	0	online	0%	68.3mb	quock	disabled
2	user-service	default	1.0.0	cluster	13764	3m	0	online	0%	66.1mb	quock	disabled

]

PM2 Monitoring

```
pm2 monit
```





Application Access

- **Frontend**: http://localhost:3000

- **API Gateway**: http://localhost:8080
- **Consul UI**: http://localhost:8500

Hình ảnh: [Đính kèm screenshots của từng service đang chạy]

☑ Ưu Điểm

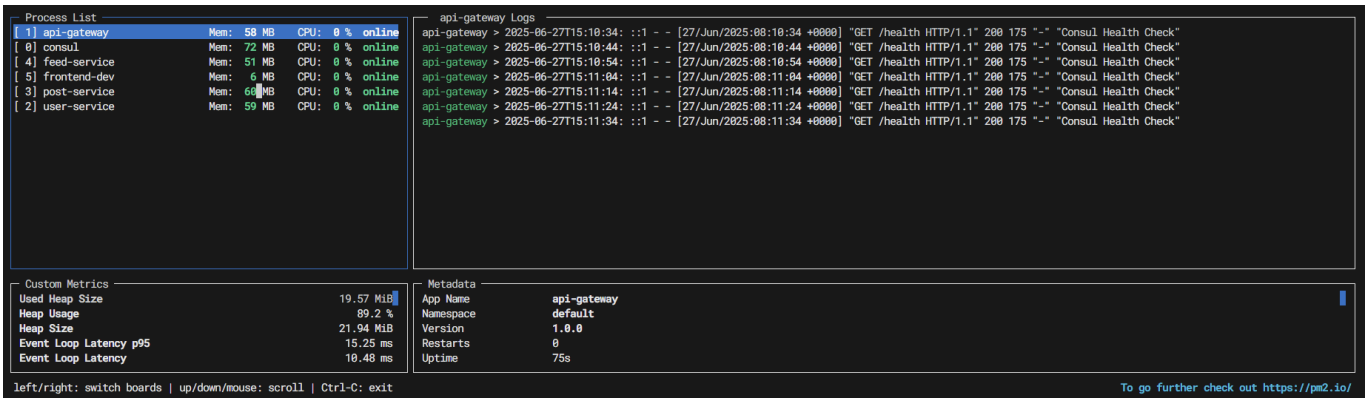
- Quản lý process dễ dàng
- Auto-restart khi crash
- Built-in monitoring
- Log management tự động
- Cluster mode support

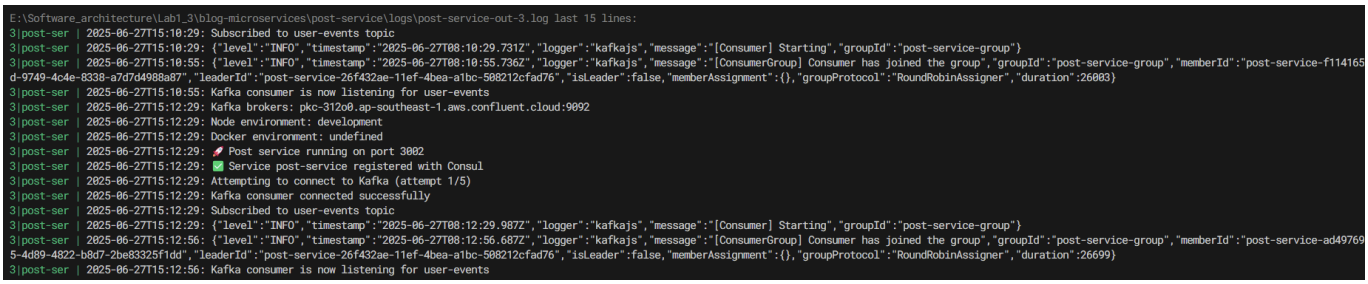
❌ Nhược Điểm

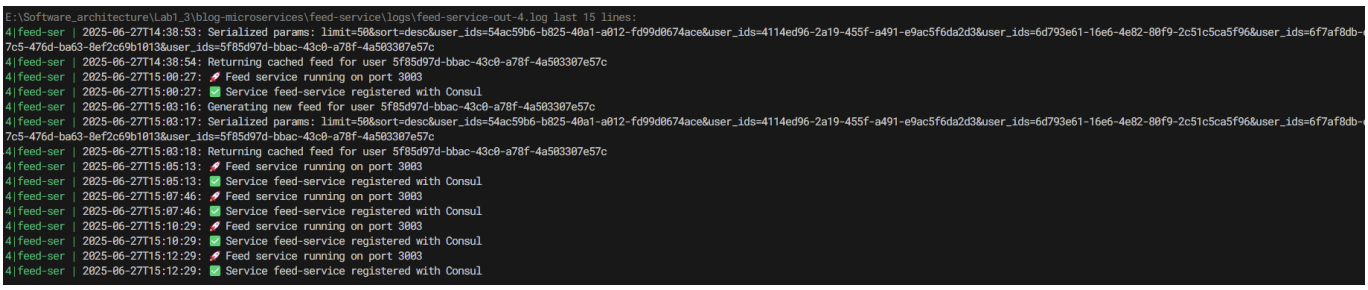
- Phụ thuộc vào môi trường local
- Không có isolation hoàn toàn
- Phức tạp khi scale

📈 Metrics và Logs

```
pm2 logs api-gateway
pm2 logs user-service
pm2 logs post-service
pm2 logs feed-service
```







Phương Pháp 2: Triển Khai với Docker Compose

Mô Tả

Docker Compose cho phép định nghĩa và chạy multi-container Docker applications với isolation hoàn toàn.

Cấu Hình

File cấu hình chính

```
docker-compose.yml
```

Các Dockerfile cho từng service:

- `api-gateway/Dockerfile`
- `user-service/Dockerfile`
- `post-service/Dockerfile`
- `feed-service/Dockerfile`
- `Frontend/Dockerfile`

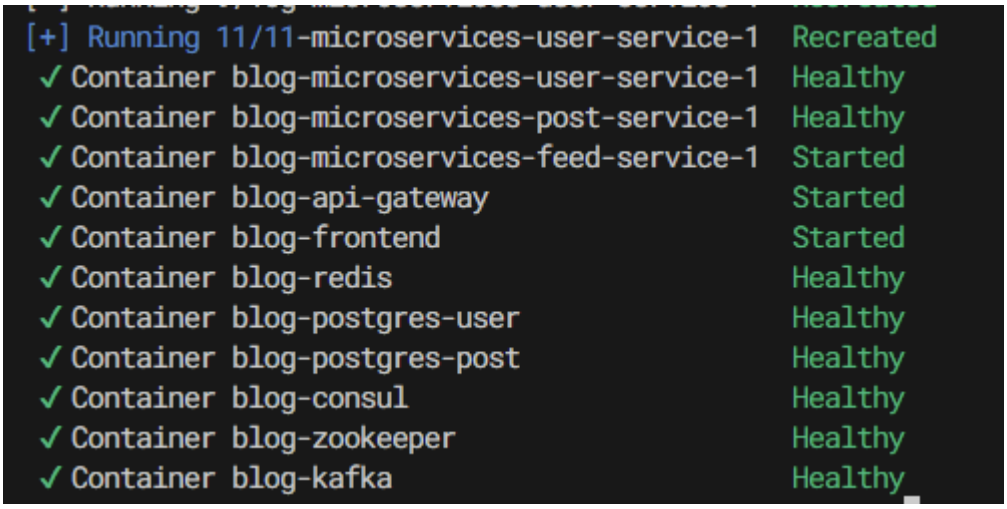
Các bước cấu hình:

Bước 1: Build containers

```
docker-compose build
```

Bước 2: Khởi động infrastructure và services

```
docker-compose up -d
```



Bước 3: Verify deployment

```
docker-compose ps
```

Kết Quả

Container Status

```
docker-compose ps
```

```
PS E:\Software_architecture\Lab1_3\blog-microservices> docker-compose ps
time="2025-06-27T15:26:27+07:00" level=warning msg="E:\\Software_architecture\\Lab1_3\\blog-microservices\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
"
NAME                                IMAGE                                COMMAND                  SERVICE         CREATED              STATUS                            PORTS
blog-api-gateway                    blog-microservices-api-gateway       "./entrypoint.sh npm…"   api-gateway     About a minute ago   Up About a minute (healthy)       0.0.0.0:8080-8081->8080-8081/tcp, [::]:8080-8081->8080-808
1/tcp, 0.0.0.0:9876->9876/tcp, [::]:9876->9876/tcp
blog-consul                         hashicorp/consul:latest              "docker-entrypoint.s…"   consul          13 hours ago         Up About a minute (healthy)       0.0.0.0:8500->8500/tcp, [::]:8500->8500/tcp, 0.0.0.0:8600-
>8600/tcp, [::]:8600->8600/tcp, 0.0.0.0:8600->8600/udp, [::]:8600->8600/udp
blog-frontend                       blog-microservices-frontend          "docker-entrypoint.s…"   frontend        About a minute ago   Up About a minute (health: starting)   0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp
blog-kafka                          confluentinc/cp-kafka:7.5.0          "/etc/confluent/dock…"   kafka           3 days ago           Up About a minute (healthy)       0.0.0.0:9092->9092/tcp, [::]:9092->9092/tcp, 0.0.0.0:29092
->29092/tcp, [::]:29092->29092/tcp
blog-microservices-feed-service-1   blog-microservices-feed-service      "docker-entrypoint.s…"   feed-service    About a minute ago   Up About a minute (healthy)       0.0.0.0:3003->3003/tcp, [::]:3003->3003/tcp
blog-microservices-post-service-1   blog-microservices-post-service      "/bin/sh /app/entryp…"   post-service    About a minute ago   Up About a minute (healthy)       0.0.0.0:3002->3002/tcp, [::]:3002->3002/tcp
blog-microservices-user-service-1   blog-microservices-user-service      "/app/entrypoint.sh …"   user-service    About a minute ago   Up About a minute (healthy)       0.0.0.0:3001->3001/tcp, [::]:3001->3001/tcp
blog-postgres-post                  postgres:15                          "docker-entrypoint.s…"   postgres-post   3 days ago           Up About a minute (healthy)       0.0.0.0:5433->5432/tcp, [::]:5433->5432/tcp
blog-postgres-user                  postgres:15                          "docker-entrypoint.s…"   postgres-user   3 days ago           Up About a minute (healthy)       0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp
blog-redis                          redis:7                              "docker-entrypoint.s…"   redis           3 days ago           Up About a minute (healthy)       0.0.0.0:6379->6379/tcp, [::]:6379->6379/tcp
blog-zookeeper                      confluentinc/cp-zookeeper:7.5.0      "/etc/confluent/dock…"   zookeeper       3 days ago           Up About a minute (healthy)       0.0.0.0:2181->2181/tcp, [::]:2181->2181/tcp
```

Container Logs

```
docker-compose logs api-gateway
docker-compose logs user-service
```

```
blog-api-gateway  | 🔧 CORS Configuration:
blog-api-gateway  |    Origins: [ 'http://localhost:3000', 'http://localhost:6060' ]
blog-api-gateway  |    Credentials: true
blog-api-gateway  |    Headers: [ 'Content-Type', 'Authorization', 'x-user-id' ]
blog-api-gateway  |    Methods: [ 'GET', 'POST', 'PUT', 'DELETE', 'OPTIONS' ]
blog-api-gateway  | [HPM] Proxy created: /  -> http://placeholder
blog-api-gateway  | [HPM] Proxy rewrite rule created: "^/users" ~> ""
blog-api-gateway  | [HPM] Proxy created: /  -> http://placeholder
blog-api-gateway  | [HPM] Proxy rewrite rule created: "^/posts" ~> ""
blog-api-gateway  | [HPM] Proxy created: /  -> http://placeholder
blog-api-gateway  | [HPM] Proxy rewrite rule created: "^/feed" ~> ""
blog-api-gateway  | Health check server started on port 8081
blog-api-gateway  | 🚀 API Gateway server started on port 8080
blog-api-gateway  | ✅ Service api-gateway registered with Consul
blog-api-gateway  | ✅ API Gateway registered with Consul
blog-api-gateway  | 👀 Watching for service changes...
blog-api-gateway  | ::1 - - [27/Jun/2025:08:25:37 +0000] "GET /health HTTP/1.1" 200 175 "-" "curl/8.14.1"
blog-api-gateway  | ::ffff:172.18.0.6 - - [27/Jun/2025:08:25:38 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"
blog-api-gateway  | ::ffff:172.18.0.6 - - [27/Jun/2025:08:25:48 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"
blog-api-gateway  | ::ffff:172.18.0.6 - - [27/Jun/2025:08:25:58 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"
blog-api-gateway  | ::1 - - [27/Jun/2025:08:26:07 +0000] "GET /health HTTP/1.1" 200 175 "-" "curl/8.14.1"
blog-api-gateway  | ::ffff:172.18.0.6 - - [27/Jun/2025:08:26:08 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"
blog-api-gateway  | ::ffff:172.18.0.6 - - [27/Jun/2025:08:26:18 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"
blog-api-gateway  | ::ffff:172.18.0.6 - - [27/Jun/2025:08:26:28 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"
blog-api-gateway  | ::1 - - [27/Jun/2025:08:26:37 +0000] "GET /health HTTP/1.1" 200 175 "-" "curl/8.14.1"
blog-api-gateway  | ::ffff:172.18.0.6 - - [27/Jun/2025:08:26:38 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"
blog-api-gateway  | ::ffff:172.18.0.6 - - [27/Jun/2025:08:26:48 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"
blog-api-gateway  | ::ffff:172.18.0.6 - - [27/Jun/2025:08:26:58 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"
blog-api-gateway  | ::1 - - [27/Jun/2025:08:27:07 +0000] "GET /health HTTP/1.1" 200 175 "-" "curl/8.14.1"
blog-api-gateway  | ::ffff:172.18.0.6 - - [27/Jun/2025:08:27:08 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"
blog-api-gateway  | ::ffff:172.18.0.6 - - [27/Jun/2025:08:27:18 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"
blog-api-gateway  | ::ffff:172.18.0.6 - - [27/Jun/2025:08:27:28 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"
```

```
user-service-1  |
user-service-1  | > user-service@1.0.0 init-db
user-service-1  | > node scripts/init-db.js
user-service-1  |
user-service-1  | Starting database initialization...
user-service-1  | Admin user created with ID: admin-fixed-id-123456789
user-service-1  | Test user created: user1 with ID: 5f85d97d-bbac-43c0-a78f-4a503307e57c
user-service-1  | Test user created: user2 with ID: 54ac59b6-b825-40a1-a012-fd99d0674ace
user-service-1  | Test user created: user3 with ID: 4114ed96-2a19-455f-a491-e9ac5f6da2d3
user-service-1  | Test user created: user4 with ID: 6d793e61-16e6-4e82-80f9-2c51c5ca5f96
user-service-1  | Test user created: user5 with ID: 6f7af8db-e7c5-476d-ba63-8ef2c69b1013
user-service-1  | Database initialization completed successfully!
user-service-1  | Starting the application...
user-service-1  |
user-service-1  | > user-service@1.0.0 start
user-service-1  | > node dist/index.js
user-service-1  |
user-service-1  | Kafka brokers: [ 'kafka:9092' ]
user-service-1  | Node environment: development
user-service-1  | Docker environment: true
user-service-1  | {"level":"WARN","timestamp":"2025-06-27T08:25:10.032Z","logger":"kafkajs","message":"KafkaJS v2.0.0 switched default partitioner. To retain the same partitioning behavior as in previous versi
ucer with the option \"createPartitioner: Partitioners.LegacyPartitioner\". See the migration guide at https://kafka.js.org/docs/migration-guide-v2.0.0#producer-new-default-partitioner for details. Silence thi
 the environment variable \"KAFKAJS_NO_PARTITIONER_WARNING=1\""}
user-service-1  | Attempting to connect to Kafka (attempt 1/5)
user-service-1  | 🚀 User service running on port 3001
user-service-1  | ✅ Service user-service registered with Consul
user-service-1  | Kafka producer connected successfully
```

Resource Usage

```
docker stats
```

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
945ffaa4dbf7	blog-frontend	0.00%	166.7MiB / 7.459GiB	2.18%	1.21kB / 126B	439MB / 274kB	49
f81206675961	blog-api-gateway	0.00%	55.38MiB / 7.459GiB	0.73%	111kB / 64.6kB	14.2MB / 451kB	33
33a58eaa3b3e	blog-microservices-feed-service-1	0.00%	171.8MiB / 7.459GiB	2.25%	20.2kB / 15.4kB	17.9MB / 106kB	33
ff3c76370d0f	blog-microservices-post-service-1	2.98%	134.4MiB / 7.459GiB	1.76%	83.3kB / 92.9kB	121MB / 193kB	40
67d6a9b0c61d	blog-microservices-user-service-1	2.65%	190.1MiB / 7.459GiB	2.49%	65.3kB / 53.7kB	172MB / 14.9MB	39
20d795f100c1	blog-consul	0.26%	111.7MiB / 7.459GiB	1.46%	156kB / 5.52MB	108MB / 1.16MB	22
178cd71d9f54	blog-kafka	0.45%	387.6MiB / 7.459GiB	5.07%	47kB / 38.2kB	2.3MB / 1.52MB	89
35320d384b87	blog-postgres-user	0.00%	32.29MiB / 7.459GiB	0.42%	36.1kB / 35.6kB	20.2MB / 569kB	6
d0a1a1b8883d	blog-postgres-post	0.00%	36.56MiB / 7.459GiB	0.48%	28.5kB / 20.4kB	26.5MB / 635kB	6
c0d5aae41751	blog-redis	0.24%	10.36MiB / 7.459GiB	0.14%	4.48kB / 6.24kB	16.5MB / 0B	6
1268891cce03	blog-zookeeper	0.08%	166.5MiB / 7.459GiB	2.18%	30.9kB / 28.9kB	85.7MB / 508kB	81

Application Access

- **Frontend**: http://localhost:3000
- **API Gateway**: http://localhost:8080
- **Consul UI**: http://localhost:8500

Hình ảnh: [Đính kèm screenshots của application running trong containers]

☑ Ưu Điểm

- Isolation hoàn toàn
- Consistent environment
- Easy scaling
- Infrastructure as code
- Cross-platform compatibility

❌ Nhược Điểm

- Resource overhead
- Learning curve
- Network complexity
- Storage management

☁️ Phương Pháp 3: Triển Khai trên Railway

📝 Mô Tả

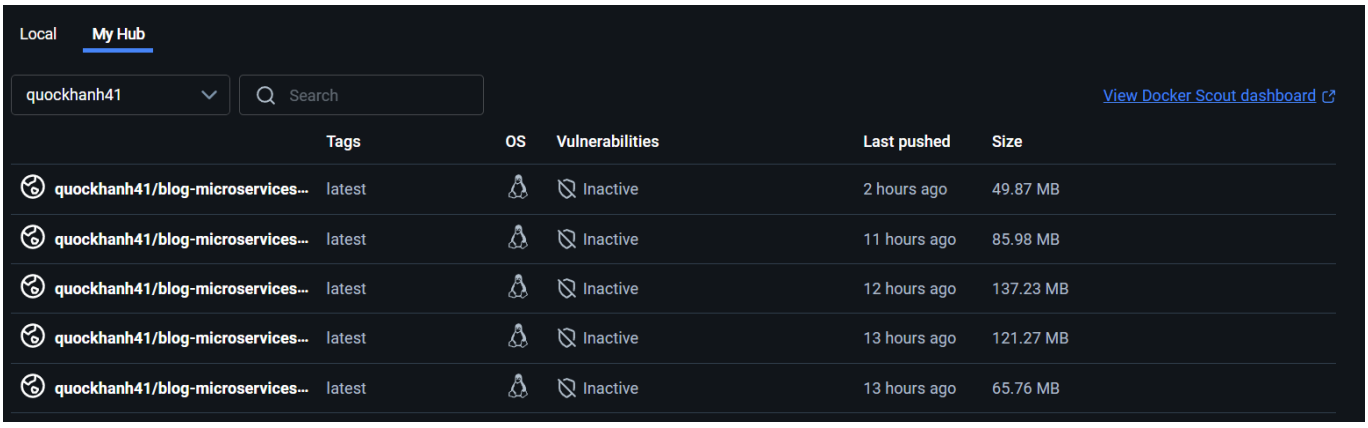
Railway là cloud platform cho phép deploy applications dễ dàng với automatic scaling và managed infrastructure.

🔧 Cấu Hình

Các bước cấu hình:

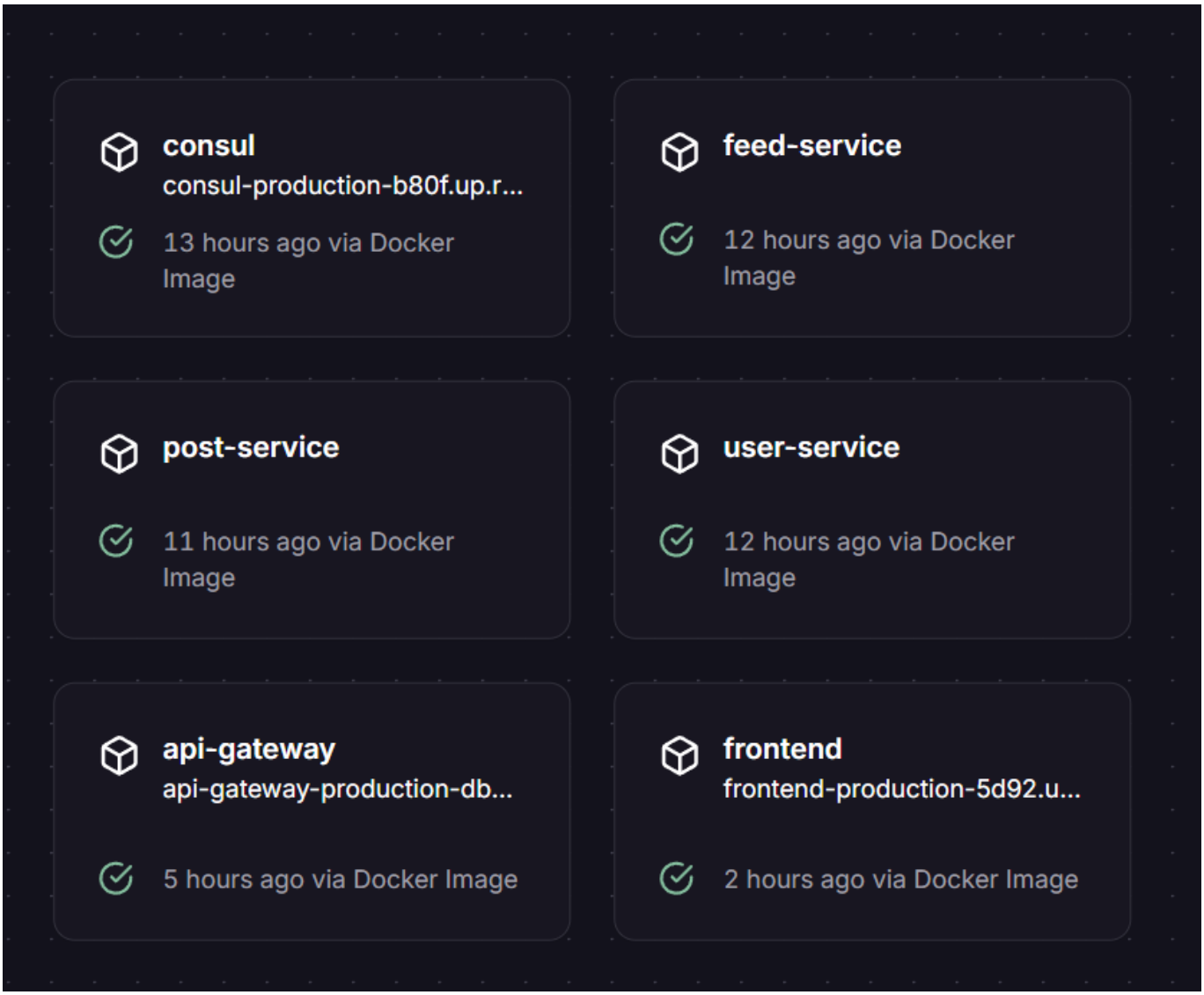
Bước 1: Build và push Docker images lên Docker Hub

```
# Build images từ docker-compose
docker-compose build

# Tag images cho Docker Hub
docker tag blog-microservices_user-service quockhanh41/blog-user-service:latest
docker tag blog-microservices_post-service quockhanh41/blog-post-service:latest
docker tag blog-microservices_feed-service quockhanh41/blog-feed-service:latest
docker tag blog-microservices_api-gateway quockhanh41/blog-api-gateway:latest
docker tag blog-microservices_frontend quockhanh41/blog-frontend:latest

# Push lên Docker Hub
docker push quockhanh41/blog-user-service:latest
docker push quockhanh41/blog-post-service:latest
docker push quockhanh41/blog-feed-service:latest
docker push quockhanh41/blog-api-gateway:latest
docker push quockhanh41/blog-frontend:latest
```



Bước 2: Tạo Railway project

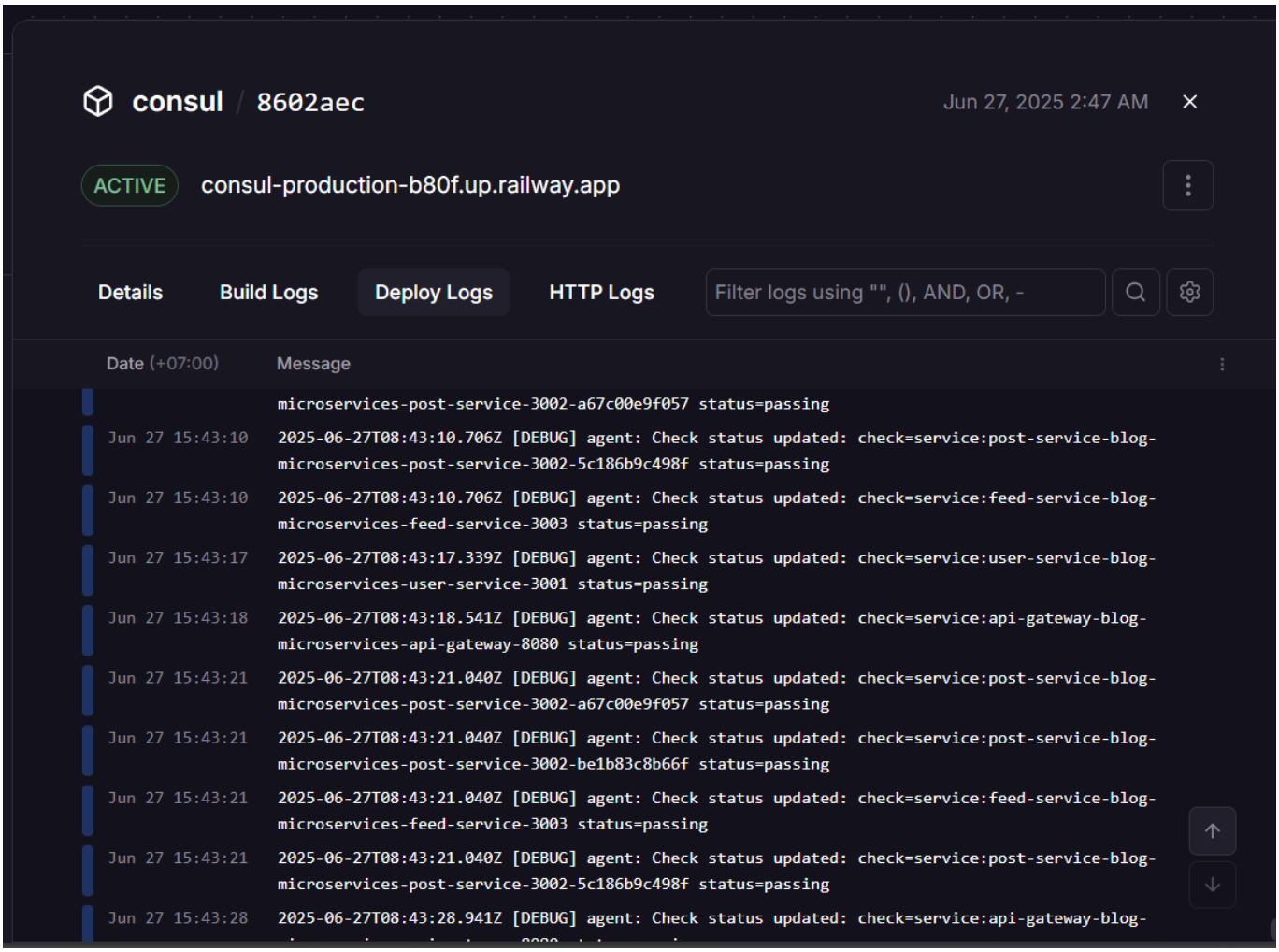
1. Truy cập Railway.app
2. Create new project (Empty Project)
3. Không cần connect GitHub repository



Bước 3: Deploy services theo thứ tự (quan trọng!)

1. Consul Service (Service Discovery)

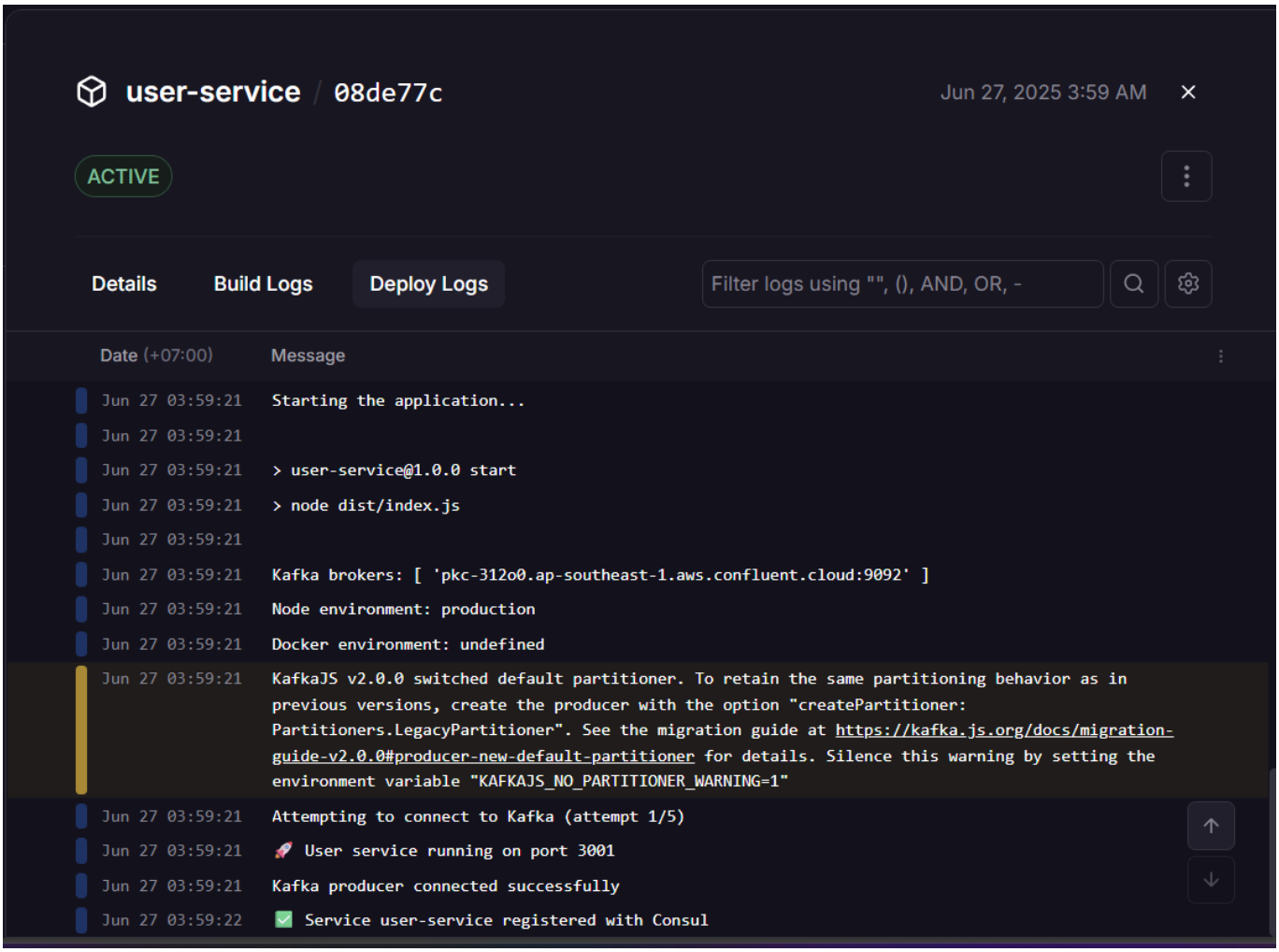
- **Deployment Method**: Docker Image
- **Docker Image**: `hashicorp/consul`
- **Port**: 8500
- **Environment Variables**: Không cần



2. User Service

- **Deployment Method**: Docker Image
- **Docker Image**: `quockhanh41/blog-microservices-user-service`
- **Port**: 3001
- **Environment Variables**:

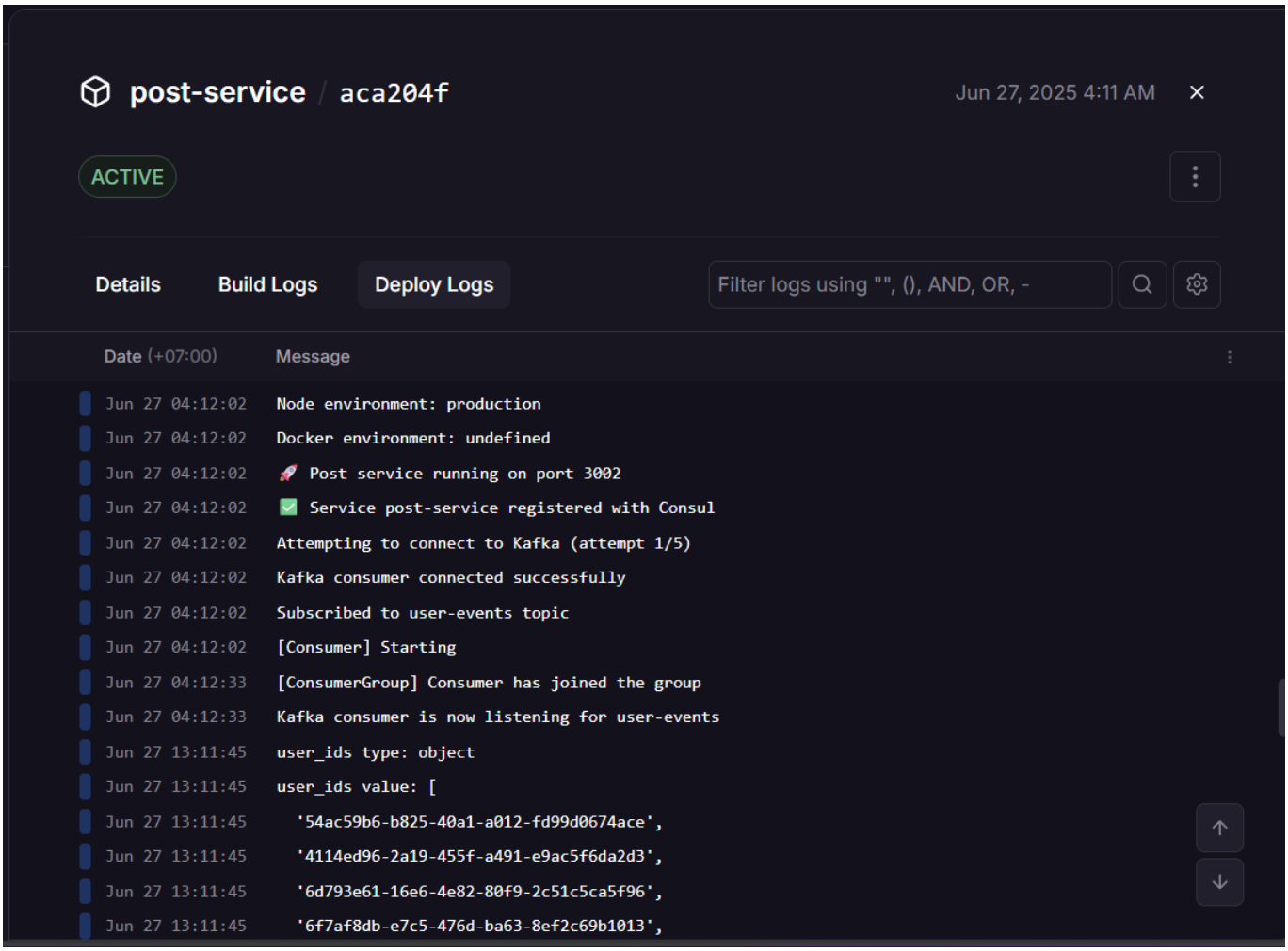
```
DATABASE_URL=postgresql://[railway-postgres-url]
JWT_SECRET=your-secret-key
CONSUL_HOST=[consul-service-internal-url]
CONSUL_PORT=8500
```



3. Post Service

- **Deployment Method**: Docker Image
- **Docker Image**: `quockhanh41/blog-microservices-post-service`
- **Port**: 3002
- **Environment Variables**:

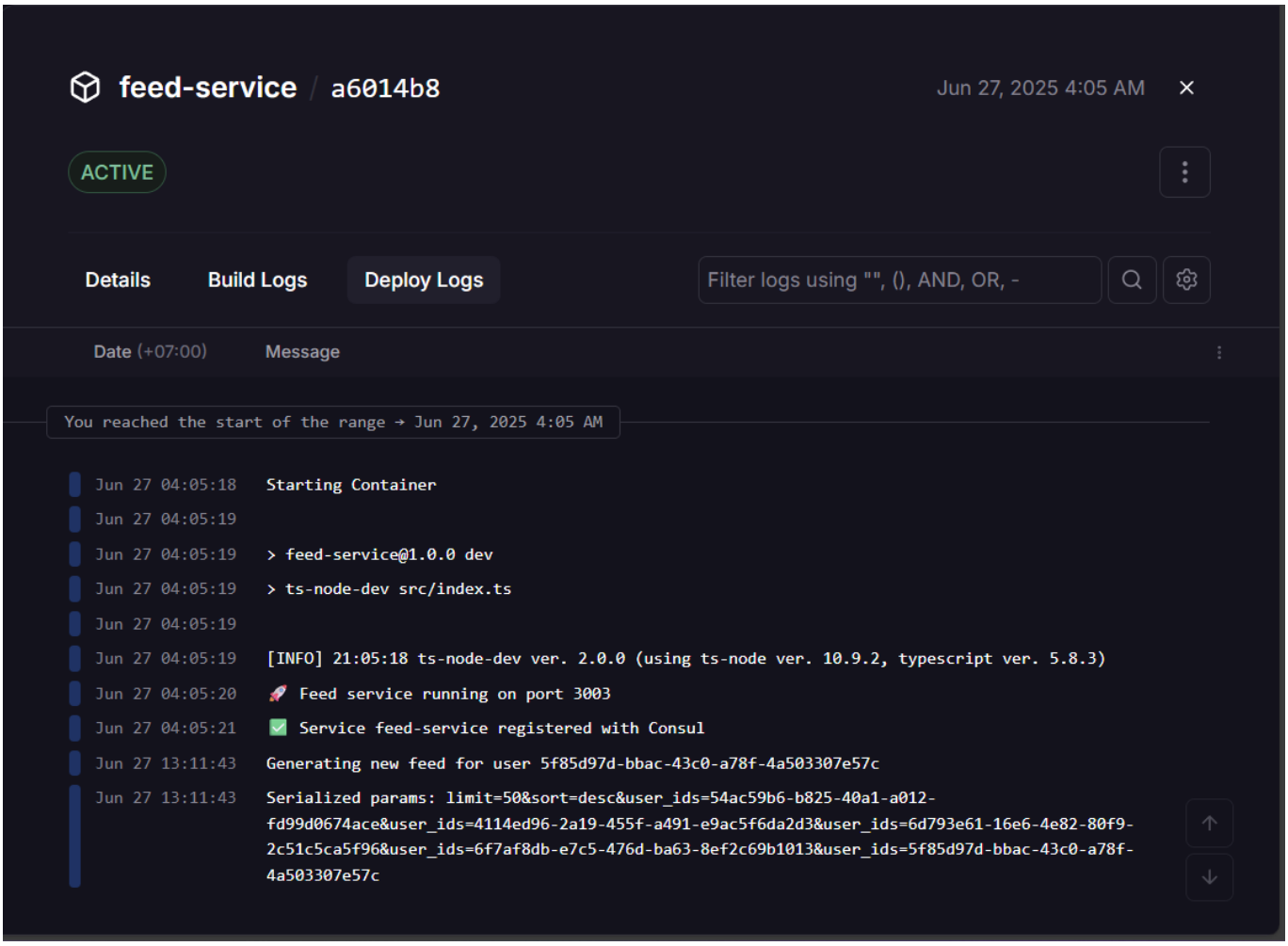
```
DATABASE_URL=postgresql://[railway-postgres-url]
JWT_SECRET=your-secret-key
CONSUL_HOST=[consul-service-internal-url]
CONSUL_PORT=8500
KAFKA_BROKERS=[kafka-broker-url]
```



4. Feed Service

- **Deployment Method**: Docker Image
- **Docker Image**: `quockhanh41/blog-microservices-feed-service`
- **Port**: 3003
- **Environment Variables**:

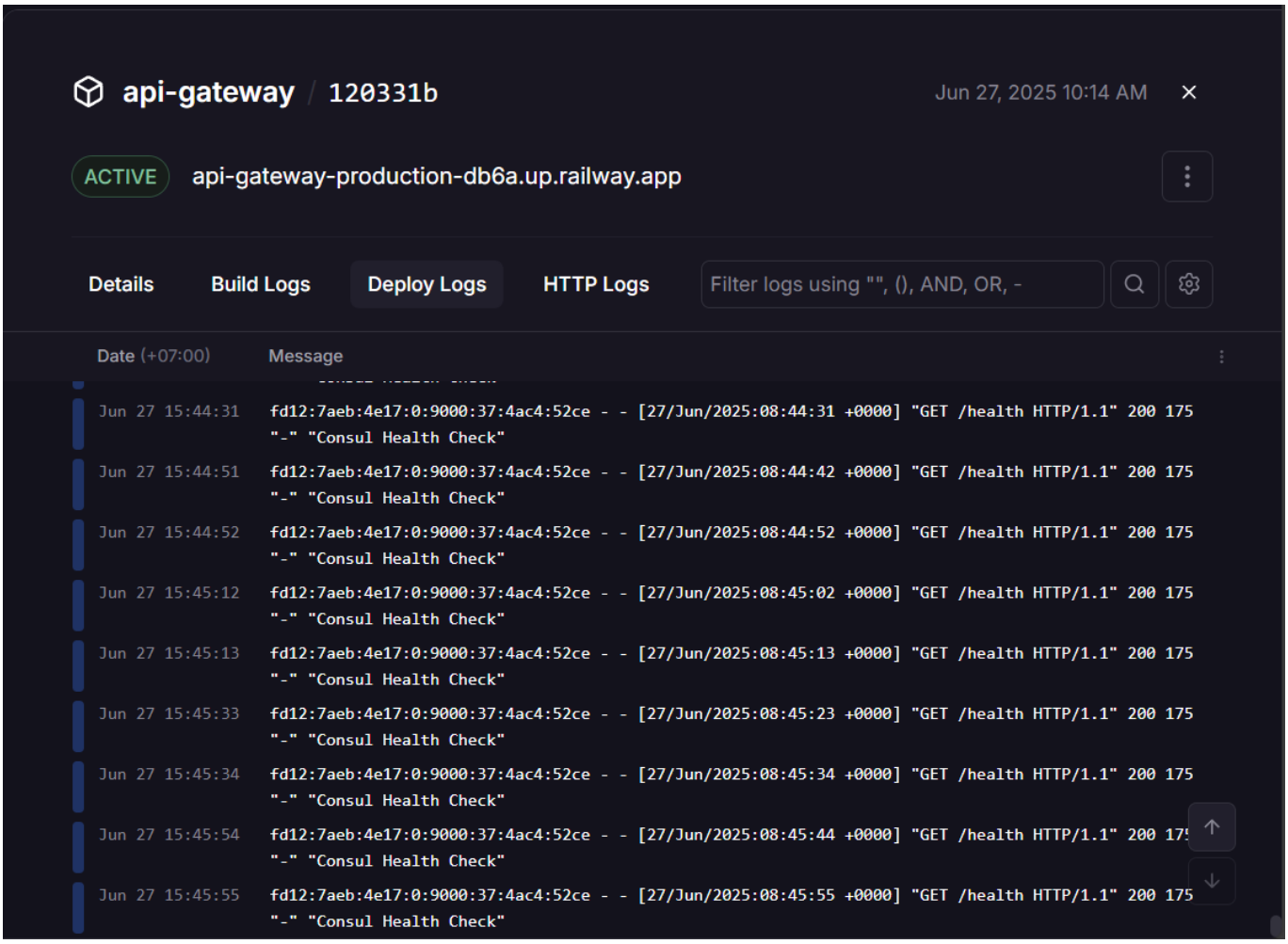
```
REDIS_URL=redis://[railway-redis-url]
CONSUL_HOST=[consul-service-internal-url]
CONSUL_PORT=8500
USER_SERVICE_URL=[user-service-internal-url]
POST_SERVICE_URL=[post-service-internal-url]
```



5. API Gateway

- **Deployment Method**: Docker Image
- **Docker Image**: `quockhanh41/blog-microservices-api-gateway`
- **Port**: 8080
- **Environment Variables**:

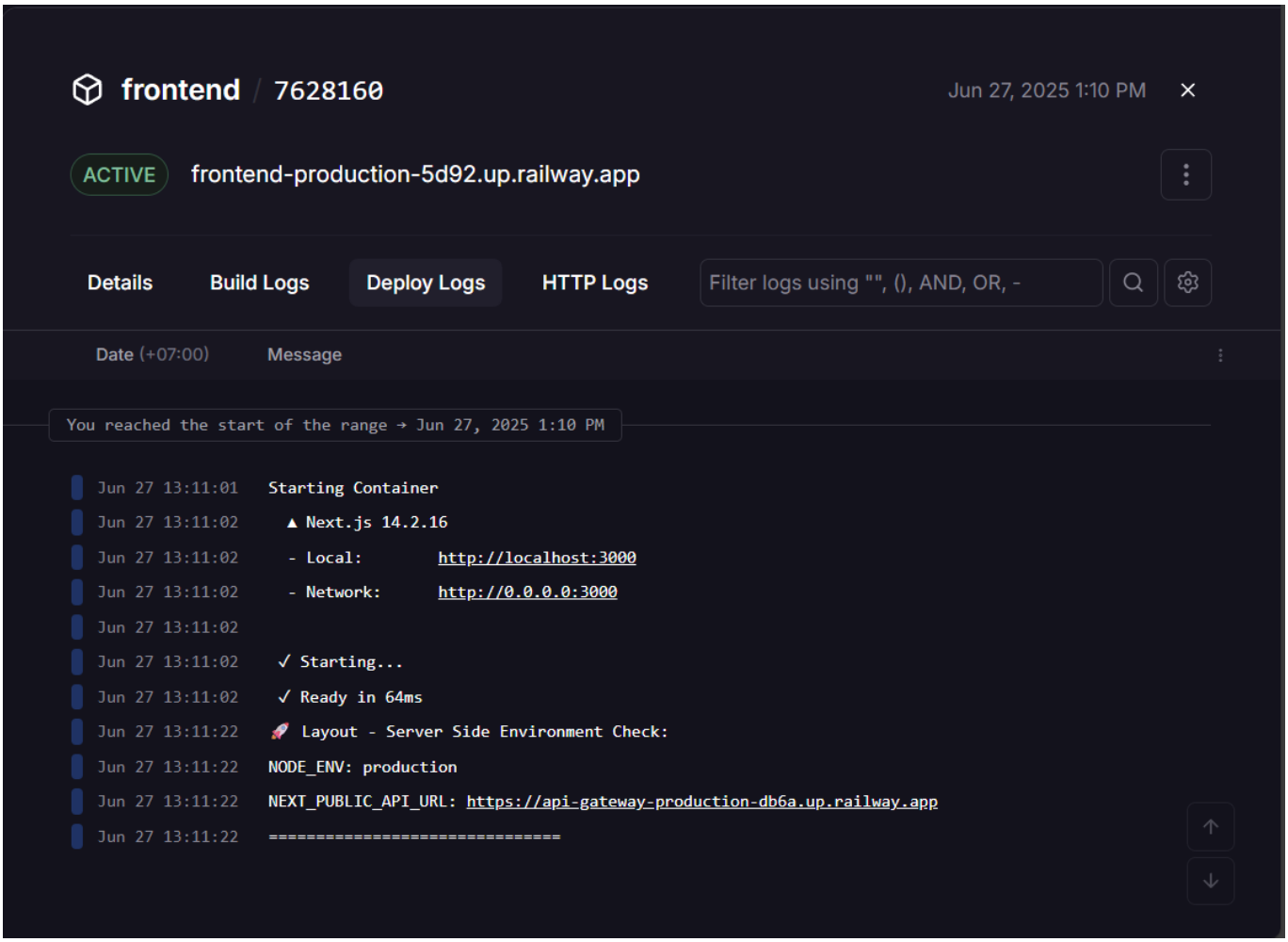
```
CONSUL_HOST=[consul-service-internal-url]
CONSUL_PORT=8500
USER_SERVICE_URL=[user-service-internal-url]
POST_SERVICE_URL=[post-service-internal-url]
FEED_SERVICE_URL=[feed-service-internal-url]
```



6. Frontend

- **Deployment Method**: Docker Image
- **Docker Image**: `quockhanh41/blog-microservices-frontend`
- **Port**: 3000
- **Environment Variables**:

```
NEXT_PUBLIC_API_URL=[api-gateway-public-url]
```



Bước 4: Configure managed databases

PostgreSQL Database

- Service: NEON PostgreSQL
- Connection URL: Sử dụng cho User Service và Post Service

Redis Database

- Service: Redis Cloud
- Connection URL: Sử dụng cho Feed Service

Bước 5: Cấu hình Service Dependencies và Networking

Sau khi mỗi service được deploy, sử dụng thông tin của service đó để cấu hình cho các service tiếp theo:

1. **Consul URL**: Lấy internal URL của Consul service
2. **Database URLs**: Từ Redis Cloud và NEON
3. **Service Internal URLs**: Sử dụng Railway private networking

Bước 6: Environment Variables tổng hợp

```
# Infrastructure Services
CONSUL_HOST=consul-production.railway.internal
```

```
CONSUL_PORT=8500
DATABASE_URL=
REDIS_URL=

# Service Internal URLs (Railway Private Network)
USER_SERVICE_URL=https://user-service-production.railway.internal:3001
POST_SERVICE_URL=https://post-service-production.railway.internal:3002
FEED_SERVICE_URL=https://feed-service-production.railway.internal:3003
API_GATEWAY_URL=https://api-gateway-production.railway.internal:8080

# Public URLs (for Frontend)
NEXT_PUBLIC_API_URL=https://api-gateway-production.up.railway.app

# Authentication
JWT_SECRET=your-secret-key

# External Services
KAFKA_BROKERS=your-kafka-broker-url
```

📊 Kết Quả

⚠ Lưu Ý Quan Trọng

Về Railway Deployment:

- Dự án này được deploy sử dụng gói **Railway Standard** để demo
- Gói này có **thời hạn 1 tháng** kể từ ngày triển khai (24/6/2025)
- **Nếu không thể truy cập các URLs bên dưới**, có nghĩa là gói đã hết hạn
- Đây là hạn chế về chi phí, không phải lỗi kỹ thuật trong quá trình deployment
- Mong thầy thông cảm cho việc này! 🙏

Deployment Status

Trạng thái: ☑ Thành công triển khai tất cả services **Ngày deploy**: 24/6/2025 **Thời hạn**: ~24/7/2025 (1 tháng)

Service URLs

- **Frontend**: https://frontend-production-5d92.up.railway.app/
- **API Gateway**: https://api-gateway-production-db6a.up.railway.app/
- **Consul UI**: https://consul-production-b80f.up.railway.app/ui/ **Internal Services** (accessible via Railway private network):
- **User Service**: https://user-service-production.railway.internal:3001
- **Post Service**: https://post-service-production.railway.internal:3002
- **Feed Service**: https://feed-service-production.railway.internal:3003

☑ Ưu Điểm

- Zero configuration deployment với pre-built Docker images
- Automatic scaling và managed infrastructure
- Built-in monitoring và health checks

- Easy collaboration với team
- Automatic HTTPS cho public endpoints
- Private networking giữa các services
- Managed databases (PostgreSQL, Redis)

❌ Nhược Điểm

- Cost cho production usage
- Cần quản lý Docker images trên Docker Hub
- Limited customization compared to self-hosted
- Vendor lock-in với Railway platform
- Phụ thuộc vào Docker Hub availability

🔍 Deployment Process Flow

1. **Infrastructure First**: Consul cho service discovery
2. **Core Services**: User và Post services với database connections
3. **Dependent Services**: Feed service phụ thuộc vào User/Post services
4. **Gateway Layer**: API Gateway routing tất cả requests
5. **Frontend**: Client application connect qua API Gateway

📊 Service Dependencies Chart

```
Consul (Service Discovery)
    ↓
User Service ← PostgreSQL
    ↓
Post Service ← PostgreSQL ← Kafka
    ↓
Feed Service ← Redis ← User/Post Services
    ↓
API Gateway ← All Services
    ↓
Frontend ← API Gateway
```

🔍 Health Monitoring

Hình ảnh: [Đính kèm screenshots Railway health monitoring]

📊 So Sánh Các Phương Pháp

Tiêu chí	PM2	Docker Compose	Railway
Độ phức tạp setup	★★★	★★★★	★★
Isolation	★★	★★★★★	★★★★★
Scalability	★★★	★★★★	★★★★★

Tiêu chí	PM2	Docker Compose	Railway
Cost	★★★★★ (Free)	★★★★★ (Free)	★★★ (Paid)
Monitoring	★★★★	★★★	★★★★★
Production Ready	★★★	★★★★	★★★★★

🎯 Kết Luận

Khuyến Nghị Sử Dụng

Development Environment

- **PM2**: Phù hợp cho development và testing local
- **Docker Compose**: Phù hợp cho development team và staging

Production Environment

- **Railway**: Phù hợp cho production với budget và cần managed infrastructure
- **Docker Compose**: Phù hợp cho production self-hosted với full control

Bài Học Kinh Nghiệm

1. **PM2**:
    - Cần cài đặt và cấu hình infrastructure dependencies manually
    - Tốt cho rapid development và debugging
    - Require system-level dependencies
2. **Docker Compose**:
    - Cần hiểu về Docker và containerization
    - Excellent cho reproducible environments
    - Resource overhead cần được consider
3. **Railway**:
    - Sử dụng pre-built Docker images từ Docker Hub
    - Deploy theo thứ tự dependencies (Consul → Services → Gateway → Frontend)
    - Automatic scaling và managed services
    - Cần quản lý thông tin service URLs cho inter-service communication

---

Ngày hoàn thành: [24/6/2025]
Tài liệu được tạo bởi: [Nguyễn Quốc Khánh - 22127188]
Version: 1.0