

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:

- PM2** - Process Manager cho Node.js
- Docker Compose** - Containerization
- 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) X + ▾

```
> 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



Services 5 total

Service	Instances	Type
consul	1 instance	
api-gateway	1 instance	api-gateway, blog, microservice
feed-service	1 instance	blog, feed-service, microservice
post-service	1 instance	blog, microservice, post-service
user-service	1 instance	blog, microservice, user-service

```

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 1615ms
○ Compiling /
✓ Compiled / in 3.9s (1866 modules)
🚀 Layout - Server Side Environment Check:
NODE_ENV: development
NEXT_PUBLIC_API_URL: http://localhost:8080
=====
GET / 200 in 4390ms
✓ Compiled in 563ms (943 modules)
|
```

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
```

Process List		api-gateway Logs									
[1]	api-gateway	Mem: 58 MB	CPU: 0 %	online	api-gateway > 2025-06-27T15:10:34: ::1 - - [27/Jun/2025:08:10:34 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"						
[0]	consul	Mem: 72 MB	CPU: 0 %	online	api-gateway > 2025-06-27T15:10:44: ::1 - - [27/Jun/2025:08:10:44 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"						
[4]	feed-service	Mem: 51 MB	CPU: 0 %	online	api-gateway > 2025-06-27T15:10:54: ::1 - - [27/Jun/2025:08:10:54 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"						
[5]	frontend-dev	Mem: 6 MB	CPU: 0 %	online	api-gateway > 2025-06-27T15:11:04: ::1 - - [27/Jun/2025:08:11:04 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"						
[3]	post-service	Mem: 65 MB	CPU: 0 %	online	api-gateway > 2025-06-27T15:11:14: ::1 - - [27/Jun/2025:08:11:14 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"						
[2]	user-service	Mem: 59 MB	CPU: 0 %	online	api-gateway > 2025-06-27T15:11:24: ::1 - - [27/Jun/2025:08:11:24 +0000] "GET /health HTTP/1.1" 200 175 "-" "Consul Health Check"						

Custom Metrics			Metadata		
Used Heap Size				19.57 MiB	
Used Heap Usage				99.2 %	
Heap Size				21.94 MiB	
Event Loop Latency p95				15.25 ms	
Event Loop Latency				10.48 ms	
left/right: switch boards up/down/mouse: scroll Ctrl-C: exit					
To go further check out https://pm2.io/					

```
E:\Software_architecture\Lab1_3\blog-microservices\user-service\logs\user-service-out-2.log last 15 lines:
2|user-ser | 2025-06-27T15:07:46: Kafka producer connected successfully
2|user-ser | 2025-06-27T15:10:29: Kafka brokers: [ 'pkc-312o0.ap-southeast-1.aws.confluent.cloud:9092' ]
2|user-ser | 2025-06-27T15:10:29: Node environment: development
2|user-ser | 2025-06-27T15:10:29: Docker environment: undefined
2|user-ser | 2025-06-27T15:10:29: Attempting to connect to Kafka (attempt 1/5)
2|user-ser | 2025-06-27T15:10:29: 🚀 User service running on port 3001
2|user-ser | 2025-06-27T15:10:29: ✅ Service user-service registered with Consul
2|user-ser | 2025-06-27T15:10:29: Kafka producer connected successfully
2|user-ser | 2025-06-27T15:12:29: Kafka brokers: [ 'pkc-312o0.ap-southeast-1.aws.confluent.cloud:9092' ]
2|user-ser | 2025-06-27T15:12:29: Node environment: development
2|user-ser | 2025-06-27T15:12:29: Docker environment: undefined
2|user-ser | 2025-06-27T15:12:29: Attempting to connect to Kafka (attempt 1/5)
2|user-ser | 2025-06-27T15:12:29: 🚀 User service running on port 3001
2|user-ser | 2025-06-27T15:12:29: ✅ Service user-service registered with Consul
2|user-ser | 2025-06-27T15:12:29: Kafka producer connected successfully
```

```
E:\Software_architecture\Lab1_3\blog-microservices\post-service\logs\post-service-out-3.log last 15 lines:
3|post-ser | 2025-06-27T15:10:29: Subscribed to user-events topic
3|post-ser | 2025-06-27T15:10:29: {"level": "INFO", "timestamp": "2025-06-27T08:10:29.731Z", "logger": "kafkajs", "message": "[Consumer] Starting", "groupId": "post-service-group"}
3|post-ser | 2025-06-27T15:10:55: {"level": "INFO", "timestamp": "2025-06-27T08:10:55.736Z", "logger": "kafkajs", "message": "[ConsumerGroup] Consumer has joined the group", "groupId": "post-service-group", "memberId": "post-service-f114165d-9749-4c4e-8338-a7d74998ba87", "leaderId": "post-service-25f432ae-11ef-4bea-abfc-598212cfad76", "isLeader": false, "memberAssignment": {}, "groupProtocol": "RoundRobinAssigner", "duration": 26003}
3|post-ser | 2025-06-27T15:10:55: Kafka consumer is now listening for user-events
3|post-ser | 2025-06-27T15:12:29: Kafka brokers: pkc-312o0.ap-southeast-1.aws.confluent.cloud:9092
3|post-ser | 2025-06-27T15:12:29: Node environment: development
3|post-ser | 2025-06-27T15:12:29: Docker environment: undefined
3|post-ser | 2025-06-27T15:12:29: 🚀 Post service running on port 3002
3|post-ser | 2025-06-27T15:12:29: ✅ Service post-service registered with Consul
3|post-ser | 2025-06-27T15:12:29: Attempting to connect to Kafka (attempt 1/5)
3|post-ser | 2025-06-27T15:12:29: Kafka consumer connected successfully
3|post-ser | 2025-06-27T15:12:29: Subscribed to user-events topic
3|post-ser | 2025-06-27T15:12:56: {"level": "INFO", "timestamp": "2025-06-27T08:12:29.987Z", "logger": "kafkajs", "message": "[Consumer] Starting", "groupId": "post-service-group"}
3|post-ser | 2025-06-27T15:12:56: {"level": "INFO", "timestamp": "2025-06-27T08:12:56.687Z", "logger": "kafkajs", "message": "[ConsumerGroup] Consumer has joined the group", "groupId": "post-service-group", "memberId": "post-service-ad497695-4d89-4822-b8d7-2be83325f1dd", "leaderId": "post-service-25f432ae-11ef-4bea-abfc-598212cfad76", "isLeader": false, "memberAssignment": {}, "groupProtocol": "RoundRobinAssigner", "duration": 26699}
3|post-ser | 2025-06-27T15:12:56: Kafka consumer is now listening for user-events
```

```
E:\Software_architecture\Lab1_3\blog-microservices\feed-service\logs\feed-service-out-4.log last 15 lines:
4|feed-ser | 2025-06-27T14:38:53: Serialized params: limit=5&sort=desc&user_ids=54ac59b6-b825-4b01-a012-fd99d0674ace&user_ids=4114ed96-2a19-455f-a491-e9ac5f6da2d3&user_ids=6d793e61-16e6-4e82-80f9-2c51c5ca5f96&user_ids=6f7af8db-7c5-476d-ba63-8ef2c69b1013&user_ids=f5f85d97d-bbac-43c0-a78f-4a593307e57c
4|feed-ser | 2025-06-27T14:38:54: Returning cached feed for user f5f85d97d-bbac-43c0-a78f-4a593307e57c
4|feed-ser | 2025-06-27T15:00:27: 🚀 Feed service running on port 3003
4|feed-ser | 2025-06-27T15:00:27: ✅ Service feed-service registered with Consul
4|feed-ser | 2025-06-27T15:00:27: Generating new feed for user f5f85d97d-bbac-43c0-a78f-4a593307e57c
4|feed-ser | 2025-06-27T15:03:17: Serialized params: limit=5&sort=desc&user_ids=54ac59b6-b825-4b01-a012-fd99d0674ace&user_ids=4114ed96-2a19-455f-a491-e9ac5f6da2d3&user_ids=6d793e61-16e6-4e82-80f9-2c51c5ca5f96&user_ids=6f7af8db-7c5-476d-ba63-8ef2c69b1013&user_ids=f5f85d97d-bbac-43c0-a78f-4a593307e57c
4|feed-ser | 2025-06-27T15:03:18: Returning cached feed for user f5f85d97d-bbac-43c0-a78f-4a593307e57c
4|feed-ser | 2025-06-27T15:05:13: 🚀 Feed service running on port 3003
4|feed-ser | 2025-06-27T15:05:13: ✅ Service feed-service registered with Consul
4|feed-ser | 2025-06-27T15:07:46: 🚀 Feed service running on port 3003
4|feed-ser | 2025-06-27T15:07:46: ✅ Service feed-service registered with Consul
4|feed-ser | 2025-06-27T15:10:29: 🚀 Feed service running on port 3003
4|feed-ser | 2025-06-27T15:10:29: ✅ Service feed-service registered with Consul
4|feed-ser | 2025-06-27T15:12:29: 🚀 Feed service running on port 3003
4|feed-ser | 2025-06-27T15:12:29: ✅ Service feed-service registered with Consul
```

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
```

```
[+] Running 11/11-microservices-user-service-1 Recreated
✓ Container blog-microservices-user-service-1 Healthy
✓ Container blog-microservices-post-service-1 Healthy
✓ Container blog-microservices-feed-service-1 Started
✓ Container blog-api-gateway Started
✓ Container blog-frontend Started
✓ Container blog-redis Healthy
✓ Container blog-postgres-user Healthy
✓ Container blog-postgres-post Healthy
✓ Container blog-consul Healthy
✓ Container blog-zookeeper Healthy
✓ Container blog-kafka Healthy
```

Bước 3: Verify deployment

```
docker-compose ps
```



Container Status

```
docker-compose ps
```

NAME	IMAGE	COMMAND	SERVICE	CREATED	STATUS	PORTS
<code>PS E:\Software_architecture\Lab1_3\blog-microservices> docker-compose ps</code>						
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->8080-8081
1/tcp	0.0.0.0:9876->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/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/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@1.0.0 init-db
user-service-1 | > node scripts/init-db.js
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-b225-40a1-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: 6d793e01-16e6-4e82-80f9-2c51c5ca5f96
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@1.0.0 start
user-service-1 | > node dist/index.js
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 versions, use the option `\\createPartitioner: Partitioners.LegacyPartitioner\\` or 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
2bd795f100c1	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

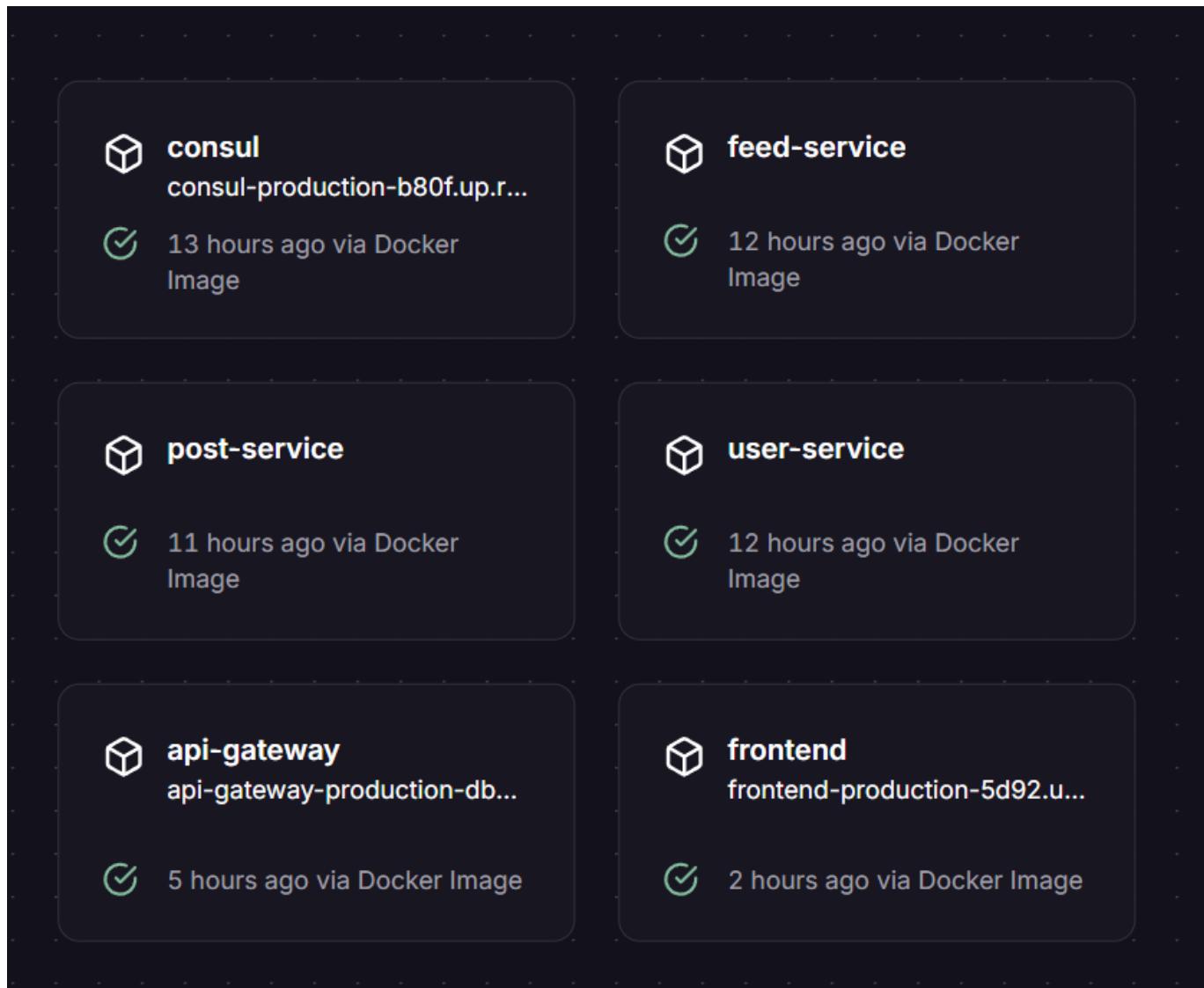
# Tag images cho Docker Hub
docker tag blog-microservices_user-service quockhanh41/blog-user-service:latest
docker tag blog-microservices_post-service quockhanh41/blog-post-service:latest
docker tag blog-microservices_feed-service quockhanh41/blog-feed-service:latest
docker tag blog-microservices_api-gateway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

Local		My Hub				
	Tags	OS	Vulnerabilities	Last pushed	Size	
⑥	quockhanh41/blog-microservices... latest	⚠️	⚡ Inactive	2 hours ago	49.87 MB	
⑥	quockhanh41/blog-microservices... latest	⚠️	⚡ Inactive	11 hours ago	85.98 MB	
⑥	quockhanh41/blog-microservices... latest	⚠️	⚡ Inactive	12 hours ago	137.23 MB	
⑥	quockhanh41/blog-microservices... latest	⚠️	⚡ Inactive	13 hours ago	121.27 MB	
⑥	quockhanh41/blog-microservices... latest	⚠️	⚡ Inactive	13 hours ago	65.76 MB	

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

The screenshot shows the Consul UI interface. At the top, it displays the service name "consul / 8602aec" and the date "Jun 27, 2025 2:47 AM". Below this, there's a status indicator "ACTIVE" and the service name "consul-production-b80f.up.railway.app". The main area is a log viewer with tabs for "Details", "Build Logs", "Deploy Logs" (which is selected), and "HTTP Logs". A search bar and filter options are also present. The log table has columns for "Date (+07:00)" and "Message". The messages show various service checks being updated as "passing".

Date (+07:00)	Message
Jun 27 15:43:10	microservices-post-service-3002-a67c00e9f057 status=passing 2025-06-27T08:43:10.706Z [DEBUG] agent: Check status updated: check=service:post-service-blog-microservices-post-service-3002-5c186b9c498f status=passing
Jun 27 15:43:10	2025-06-27T08:43:10.706Z [DEBUG] agent: Check status updated: check=service:feed-service-blog-microservices-feed-service-3003 status=passing
Jun 27 15:43:17	2025-06-27T08:43:17.339Z [DEBUG] agent: Check status updated: check=service:user-service-blog-microservices-user-service-3001 status=passing
Jun 27 15:43:18	2025-06-27T08:43:18.541Z [DEBUG] agent: Check status updated: check=service:api-gateway-blog-microservices-api-gateway-8080 status=passing
Jun 27 15:43:21	2025-06-27T08:43:21.040Z [DEBUG] agent: Check status updated: check=service:post-service-blog-microservices-post-service-3002-a67c00e9f057 status=passing
Jun 27 15:43:21	2025-06-27T08:43:21.040Z [DEBUG] agent: Check status updated: check=service:post-service-blog-microservices-post-service-3002-be1b83c8b66f status=passing
Jun 27 15:43:21	2025-06-27T08:43:21.040Z [DEBUG] agent: Check status updated: check=service:feed-service-blog-microservices-feed-service-3003 status=passing
Jun 27 15:43:21	2025-06-27T08:43:21.040Z [DEBUG] agent: Check status updated: check=service:post-service-blog-microservices-post-service-3002-5c186b9c498f status=passing
Jun 27 15:43:28	2025-06-27T08:43:28.941Z [DEBUG] agent: Check status updated: check=service:api-gateway-blog-microservices-api-gateway-8080 status=passing

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
```

Jun 27, 2025 3:59 AM X

ACTIVE

Details Build Logs Deploy Logs Filter logs using "", (), AND, OR, - Q ⚙️

Date (+07:00)	Message
Jun 27 03:59:21	Starting the application...
Jun 27 03:59:21	
Jun 27 03:59:21	> user-service@1.0.0 start
Jun 27 03:59:21	> node dist/index.js
Jun 27 03:59:21	
Jun 27 03:59:21	Kafka brokers: ['pkc-312o0.ap-southeast-1.aws.confluent.cloud:9092']
Jun 27 03:59:21	Node environment: production
Jun 27 03:59:21	Docker environment: undefined
Jun 27 03:59:21	KafkaJS v2.0.0 switched default partitioner. To retain the same partitioning behavior as in previous versions, create the producer 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 this warning by setting the environment variable "KAFKAJS_NO_PARTITIONER_WARNING=1"
Jun 27 03:59:21	Attempting to connect to Kafka (attempt 1/5)
Jun 27 03:59:21	🚀 User service running on port 3001
Jun 27 03:59:21	Kafka producer connected successfully
Jun 27 03:59:22	✓ Service user-service registered with Consul

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]
```

The screenshot shows a deployment log viewer interface. At the top, it displays the service name "post-service / aca204f" and the date "Jun 27, 2025 4:11 AM". There is also an "ACTIVE" status indicator and a three-dot menu icon.

The main area is a table with two columns: "Date (+07:00)" and "Message". The "Deploy Logs" tab is selected. A search bar at the top right allows filtering logs using various operators like "", (), AND, OR, -.

Date (+07:00)	Message
Jun 27 04:12:02	Node environment: production
Jun 27 04:12:02	Docker environment: undefined
Jun 27 04:12:02	🚀 Post service running on port 3002
Jun 27 04:12:02	✅ Service post-service registered with Consul
Jun 27 04:12:02	Attempting to connect to Kafka (attempt 1/5)
Jun 27 04:12:02	Kafka consumer connected successfully
Jun 27 04:12:02	Subscribed to user-events topic
Jun 27 04:12:02	[Consumer] Starting
Jun 27 04:12:33	[ConsumerGroup] Consumer has joined the group
Jun 27 04:12:33	Kafka consumer is now listening for user-events
Jun 27 13:11:45	user_ids type: object
Jun 27 13:11:45	user_ids value: [
Jun 27 13:11:45	'54ac59b6-b825-40a1-a012-fd99d0674ace',
Jun 27 13:11:45	'4114ed96-2a19-455f-a491-e9ac5f6da2d3',
Jun 27 13:11:45	'6d793e61-16e6-4e82-80f9-2c51c5ca5f96',
Jun 27 13:11:45	'6f7af8db-e7c5-476d-ba63-8ef2c69b1013',

At the bottom right of the log table, there are two small buttons: an upward arrow and a downward arrow.

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]
```

The screenshot shows a deployment log interface for a service named 'feed-service' with a hash 'a6014b8'. The status is 'ACTIVE'. The log tab is selected, showing deployment details and build logs. A search bar allows filtering logs using various operators like AND, OR, and NOT. The log output includes timestamped messages such as 'Starting Container', 'Feed service running on port 3003', and 'Service feed-service registered with Consul'. The interface also features a date range selector and a message input field.

```
Jun 27 04:05:18 Starting Container
Jun 27 04:05:19
Jun 27 04:05:19 > feed-service@1.0.0 dev
Jun 27 04:05:19 > ts-node-dev src/index.ts
Jun 27 04:05:19
Jun 27 04:05:19 [INFO] 21:05:18 ts-node-dev ver. 2.0.0 (using ts-node ver. 10.9.2, typescript ver. 5.8.3)
Jun 27 04:05:20 🚀 Feed service running on port 3003
Jun 27 04:05:21 ✅ Service feed-service registered with Consul
Jun 27 13:11:43 Generating new feed for user 5f85d97d-bbac-43c0-a78f-4a503307e57c
Jun 27 13:11:43 Serialized params: limit=50&sort=desc&user_ids=54ac59b6-b825-40a1-a012-fd99d0674ace&user_ids=4114ed96-2a19-455f-a491-e9ac5f6da2d3&user_ids=6d793e61-16e6-4e82-80f9-2c51c5ca5f96&user_ids=6f7af8db-e7c5-476d-ba63-8ef2c69b1013&user_ids=5f85d97d-bbac-43c0-a78f-4a503307e57c
```

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]
```

The screenshot shows a deployment log interface for a service named 'api-gateway / 120331b'. The 'Deploy Logs' tab is selected. The logs list several entries of 'Consul Health Check' requests, all returning a status code of 200 and a response time of 175ms. The log entries are timestamped from Jun 27 15:44:31 to Jun 27 15:45:55.

Date (+07:00)	Message
Jun 27 15:44:31	fd12:7aeb:4e17:0:9000:37:4ac4:52ce - - [27/Jun/2025:08:44:31 +0000] "GET /health HTTP/1.1" 200 175 "- " "Consul Health Check"
Jun 27 15:44:51	fd12:7aeb:4e17:0:9000:37:4ac4:52ce - - [27/Jun/2025:08:44:42 +0000] "GET /health HTTP/1.1" 200 175 "- " "Consul Health Check"
Jun 27 15:44:52	fd12:7aeb:4e17:0:9000:37:4ac4:52ce - - [27/Jun/2025:08:44:52 +0000] "GET /health HTTP/1.1" 200 175 "- " "Consul Health Check"
Jun 27 15:45:12	fd12:7aeb:4e17:0:9000:37:4ac4:52ce - - [27/Jun/2025:08:45:02 +0000] "GET /health HTTP/1.1" 200 175 "- " "Consul Health Check"
Jun 27 15:45:13	fd12:7aeb:4e17:0:9000:37:4ac4:52ce - - [27/Jun/2025:08:45:13 +0000] "GET /health HTTP/1.1" 200 175 "- " "Consul Health Check"
Jun 27 15:45:33	fd12:7aeb:4e17:0:9000:37:4ac4:52ce - - [27/Jun/2025:08:45:23 +0000] "GET /health HTTP/1.1" 200 175 "- " "Consul Health Check"
Jun 27 15:45:34	fd12:7aeb:4e17:0:9000:37:4ac4:52ce - - [27/Jun/2025:08:45:34 +0000] "GET /health HTTP/1.1" 200 175 "- " "Consul Health Check"
Jun 27 15:45:54	fd12:7aeb:4e17:0:9000:37:4ac4:52ce - - [27/Jun/2025:08:45:44 +0000] "GET /health HTTP/1.1" 200 175 "- " "Consul Health Check"
Jun 27 15:45:55	fd12:7aeb:4e17:0:9000:37:4ac4:52ce - - [27/Jun/2025:08:45:55 +0000] "GET /health HTTP/1.1" 200 175 "- " "Consul Health Check"

6. Frontend

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

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

The screenshot shows the Railway Deploy Logs interface for the 'frontend' service. The top bar displays the service name 'frontend / 7628160' and the date 'Jun 27, 2025 1:10 PM'. Below the header, there are tabs for 'Details', 'Build Logs', 'Deploy Logs' (which is active), and 'HTTP Logs'. A search bar and filter options are also present. The main area shows deployment logs starting with 'Starting Container' and ending with '====='. Log entries include environment variables like NODE_ENV and NEXT_PUBLIC_API_URL.

```
Jun 27 13:11:01 Starting Container
Jun 27 13:11:02 ▲ Next.js 14.2.16
Jun 27 13:11:02 - Local: http://localhost:3000
Jun 27 13:11:02 - Network: http://0.0.0.0:3000
Jun 27 13:11:02 ✓ Starting...
Jun 27 13:11:02 ✓ Ready in 64ms
Jun 27 13:11:22 🚀 Layout - Server Side Environment Check:
Jun 27 13:11:22 NODE_ENV: production
Jun 27 13:11:22 NEXT_PUBLIC_API_URL: https://api-gateway-production-db6a.up.railway.app
Jun 27 13:11:22 =====
```

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

```
graph TD; Consul[Consul (Service Discovery)] --> User[User Service]; User --> Post[Post Service]; Post --> Kafka[Kafka]; Kafka --> Feed[Feed Service]; Feed --> Redis[Redis]; Redis --> User; Redis --> Post; User --> API[API Gateway]; Post --> API; Feed --> API; API --> Frontend[Frontend]; Frontend --> API;
```

⌚ 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 Khanh - 22127188]

Version: 1.0