

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG



BÁO CÁO ĐỒ ÁN
Môn: CÔNG NGHỆ DEVOPS VÀ ỨNG DỤNG

Đề tài:

**DEPLOY REVIEW COMPANY MICROSERVICES SỬ
DỤNG DEVOPS TOOLS TÍCH HỢP CÁC CÔNG CỤ BẢO
MẬT VÀ GIÁM SÁT.**

Giảng viên hướng dẫn : ThS. LÊ ANH TUẤN
Lớp : NT548.P11.MMCL

SINH VIÊN THỰC HIỆN

Lâm Bảo Duy	MSSV: 20521231
Hồ Hải Dương	MSSV: 21520202

THỦ ĐỨC, TP. HCM – 2024

MỤC LỤC NỘI DUNG

A. LỜI MỞ ĐẦU	1
1. TỔNG QUAN VỀ ĐỀ TÀI	1
1.1. Tổng quan.....	1
1.2. Mục tiêu đồ án	1
2. BẢNG PHÂN CHIA CÔNG VIỆC.....	2
B. NỘI DUNG CHÍNH.....	3
1. CƠ SỞ LÝ THUYẾT	3
1.1. Tổng quan về project.....	3
1.2 Các công nghệ sử dụng cho project.....	3
1.1.1. CloudFormation	3
1.1.2. Github.....	4
1.1.3 Java Spring Boot	4
1.1.4 SonarQube.....	4
1.1.5 Docker.....	5
1.1.6 Trivy	6
1.1.7 Jenkins.....	6
1.1.8 Prometheus	7
1.1.9 Grafana.....	9
1.1.10. K3S.....	9
2. MÔ HÌNH HỆ THỐNG.....	11
3. QUY TRÌNH THỰC HIỆN	3
3.1 Xây dựng hạ tầng với CloudFormation.....	3
3.1.1 vpc.yaml	3
3.1.2 ec2.yaml	9
3.1.3 main.yaml.....	11
3.2 Triển Khai CloudFormation	12
3.2.1 Tạo S3 Bucket.....	12
3.2.2 Tạo Stack.....	12

3.2.3 Kiểm tra cấu hình trên AWS Console	14
3.3 Cấu hình SonarQube	17
3.3.1 Tạo token.....	17
3.3.2 Gán token cho các microservices.....	18
3.4 Cấu hình K3S	19
3.4.1 account.yaml	19
3.4.2 Kiểm tra cấu hình.....	21
3.5 Cấu hình Jenkins	22
3.5.1 Credential	22
3.5.2 Cấu hình K3s kết nối với Jenkins	23
3.5.3 Cấu hình Sonar Sever.....	24
3.5.4 Cài đặt các tools cần thiết trên Jenkins	24
3.6 Tạo Pipeline Job và Chạy Pipeline Jenkins.....	28
3.6.1 Tạo JenkinsFile cho các Microservices	28
3.6.2 Tạo JenkinsFile để Deploy lên K3S.....	35
4. CẤU HÌNH KUBERNETES	39
4.1 postgresql-deployment.yaml	39
4.2 companyms-deployment.yaml	42
4.3 jobms-deployment.yaml	44
4.4 reviewms-deployment.yaml	45
5. CẤU HÌNH PROMETHEUS & GRAFANA.....	46
5.1 Prometheus	46
5.2 Grafana	46
6. KẾT QUẢ THỰC HIỆN	47
6.1 Jenkins	47
6.1.1 Kết quả Run Pipeline	48
6.1.2 Pipeline Log	48
6.2 SonarQube	54
6.3 Docker Hub	56

6.4 K3S	57
6.5 Prometheus	58
6.6 Grafana	58
7. DEMO	58
7.1 Call API	58
7.2 Prometheus	62
7.3 Grafana	62
8. TỔNG KẾT.....	64
TÀI LIỆU THAM KHẢO.....	66

A. LỜI MỞ ĐẦU

1. TỔNG QUAN VỀ ĐỀ TÀI

1.1. Tổng quan

Trong bối cảnh các hệ thống phần mềm ngày càng trở nên phức tạp, microservices được xem như một kiến trúc hiệu quả giúp chia nhỏ các chức năng của ứng dụng thành các dịch vụ độc lập, dễ bảo trì và triển khai. Việc áp dụng DevOps và tích hợp các công cụ bảo mật, giám sát vào quy trình triển khai microservices là một bước quan trọng nhằm tối ưu hóa quy trình phát triển phần mềm, đảm bảo chất lượng, bảo mật, và khả năng vận hành của hệ thống.

Đề tài “**Triển khai và giám sát hệ thống microservices cho ứng dụng review công ty sử dụng công cụ DevOps tích hợp các công cụ bảo mật và giám sát**” nhằm mục tiêu xây dựng quy trình DevOps tự động, kết hợp bảo mật và giám sát toàn diện, tạo nền tảng để ứng dụng vận hành hiệu quả, bảo mật và ổn định.

1.2. Mục tiêu đồ án

1. Triển khai hệ thống microservices

- Xây dựng kiến trúc microservices gồm các dịch vụ độc lập như quản lý thông tin công ty, công việc, đánh giá.
- Sử dụng công cụ như Docker và Kubernetes để đóng gói và triển khai.

2. Áp dụng DevOps

- Thiết kế và triển khai pipeline CI/CD bằng Jenkins, tự động hóa quy trình build, test và deploy.
- Sử dụng Docker Hub để quản lý image.

3. Tích hợp công cụ bảo mật

- Áp dụng các công cụ bảo mật Trivy để kiểm tra lỗ hổng trong mã nguồn và image container.

4. Giám sát hệ thống

- Cài đặt các công cụ giám sát như Prometheus và Grafana để theo dõi hiệu năng, trạng thái dịch vụ và tài nguyên.

2. BẢNG PHÂN CHIA CÔNG VIỆC

STT	Tên	MSSV	Công việc	Hoàn thành
1	Lâm Bảo Duy	20521231	Triển khai tự động cơ sở hạ tầng, deploy pipeline Microservices	100%
2	Hồ Hải Dương	21520202	Theo dõi, đánh giá hiệu năng với Prometheus, Grafana.	100%

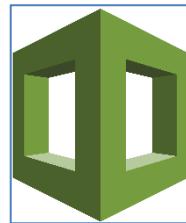
B. NỘI DUNG CHÍNH

1. CƠ SỞ LÝ THUYẾT

1.1. Tổng quan về project

1.2 Các công nghệ sử dụng cho project

1.1.1. CloudFormation



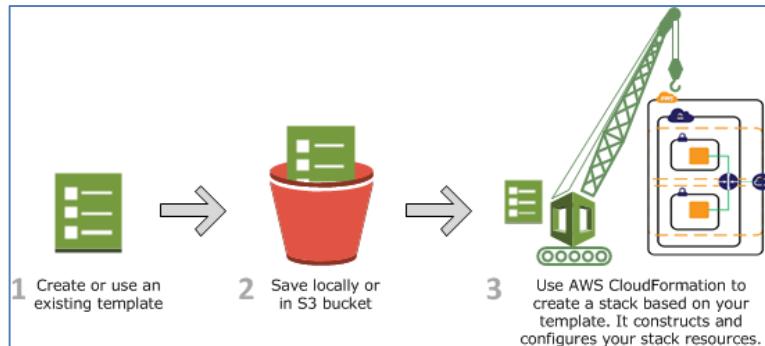
Hình 1 – Logo CloudFormation

AWS CloudFormation là một dịch vụ của Amazon Web Services (AWS) cho phép mô hình hóa và thiết lập các tài nguyên AWS một cách tự động và nhất quán.

Lợi ích khi sử dụng:

- CloudFormation cho phép quản lý một tập hợp các tài nguyên như một đơn vị duy nhất, giúp giảm thiểu công sức và thời gian trong việc tạo và cấu hình các tài nguyên AWS.
- có thể triển khai cùng một cấu trúc hạ tầng ở nhiều khu vực khác nhau một cách nhất quán và có thể lặp lại.
- Dễ dàng quay lại cấu hình trước đó nếu cần.

Cách hoạt động:



Hình 2 – Quy trình hoạt động của CloudFormation

1.1.2. Github



Hình 3 – Logo GitHub

GitHub là một nền tảng dựa trên web giúp các nhà phát triển phần mềm lưu trữ, quản lý và theo dõi các dự án mã nguồn một cách hiệu quả, cung cấp các công cụ mạnh mẽ để kiểm soát phiên bản, theo dõi vấn đề (issue tracking), và hỗ trợ quy trình xem xét mã (code review).

GitHub đóng vai trò quan trọng trong việc thúc đẩy các thực hành DevOps, giúp các nhóm phát triển phần mềm và vận hành hệ thống hợp tác hiệu quả hơn.

1.1.3 Java Spring Boot



Hình 4 – Logo Spring Boot

Spring Boot là một công cụ mã nguồn mở là một phần mở rộng của Spring Framework, cung cấp một bộ công cụ mạnh mẽ để xây dựng các ứng dụng độc lập. Phù hợp để triển khai các ứng dụng web và microservices.

Tích hợp dễ dàng với các công nghệ DevOps: Spring Boot dễ dàng tích hợp với các công cụ như Jenkins, Docker, Kubernetes và các nền tảng CI/CD khác, hỗ trợ tự động hóa quy trình phát triển và triển khai.

1.1.4 SonarQube



Hình 5 – Logo SonarQube

SonarQube là một nền tảng mã nguồn mở được phát triển bởi SonarSource, dùng để kiểm tra liên tục chất lượng mã nguồn thông qua việc thực hiện các đánh giá tự động

bằng phân tích tinh mã nguồn. Công cụ này giúp phát hiện các lỗi, mùi mã (code smells), và lỗ hổng bảo mật trong hơn 20 ngôn ngữ lập trình khác nhau.

Lợi ích:

- Cải thiện chất lượng mã nguồn.
- Giảm thiểu rủi ro bảo mật.
- Tăng hiệu quả phát triển,
- Tích hợp dễ dàng với các công cụ CI/CD như Jenkins, Azure DevOps, giúp tự động hóa quá trình kiểm tra chất lượng mã trong chu kỳ phát triển phần mềm.

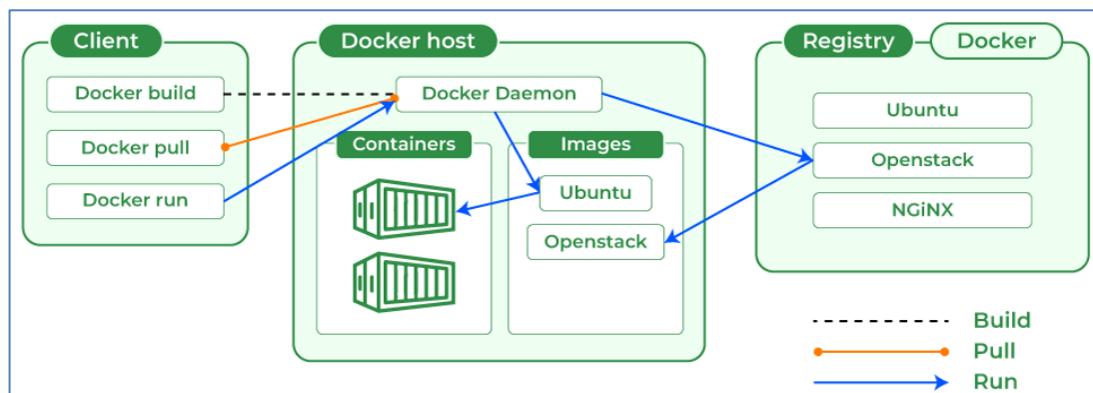
1.1.5 Docker



Hình 6 – Logo Docker

Docker là một nền tảng cung cấp dịch vụ Platform as a Service (PaaS), sử dụng công nghệ ảo hóa ở cấp độ hệ điều hành để đóng gói phần mềm trong các đơn vị gọi là container.

Kiến trúc của Docker:



Hình 7 – Kiến trúc của Docker

Ứng dụng của Docker:

- Docker cung cấp môi trường nhất quán cho việc phát triển, kiểm thử và triển khai ứng dụng, giúp giảm thiểu các vấn đề liên quan đến sự khác biệt giữa các môi trường.

- Docker hỗ trợ triển khai các kiến trúc microservices bằng cách đóng gói từng dịch vụ trong một container riêng biệt, giúp quản lý và mở rộng dễ dàng.
- Docker tích hợp tốt với các công cụ CI/CD, cho phép tự động hóa quy trình xây dựng, kiểm thử và triển khai ứng dụng.

1.1.6 Trivy



Hình 8 – Logo Trivy

Trivy là một công cụ mã nguồn mở được thiết kế để quét và phát hiện các lỗ hổng bảo mật trong hình ảnh container, mã nguồn, và các tệp cấu hình. Với khả năng tích hợp dễ dàng vào quy trình DevOps, Trivy giúp các nhóm phát triển và vận hành xác định và khắc phục sớm các lỗ hổng, đảm bảo an ninh cho ứng dụng trong suốt vòng đời phát triển.

Lợi ích khi sử dụng:

- Trivy không yêu cầu cấu hình phức tạp, cho phép người dùng bắt đầu quét lỗ hổng nhanh chóng mà không cần thiết lập nhiều.
- Trivy thực hiện quá trình quét nhanh chóng và cung cấp kết quả chi tiết, giúp tiết kiệm thời gian trong quá trình kiểm tra bảo mật.
- Ngoài việc quét hình ảnh container, Trivy còn hỗ trợ quét mã nguồn và tệp cấu hình, cung cấp cái nhìn toàn diện về tình trạng bảo mật của ứng dụng.

1.1.7 Jenkins

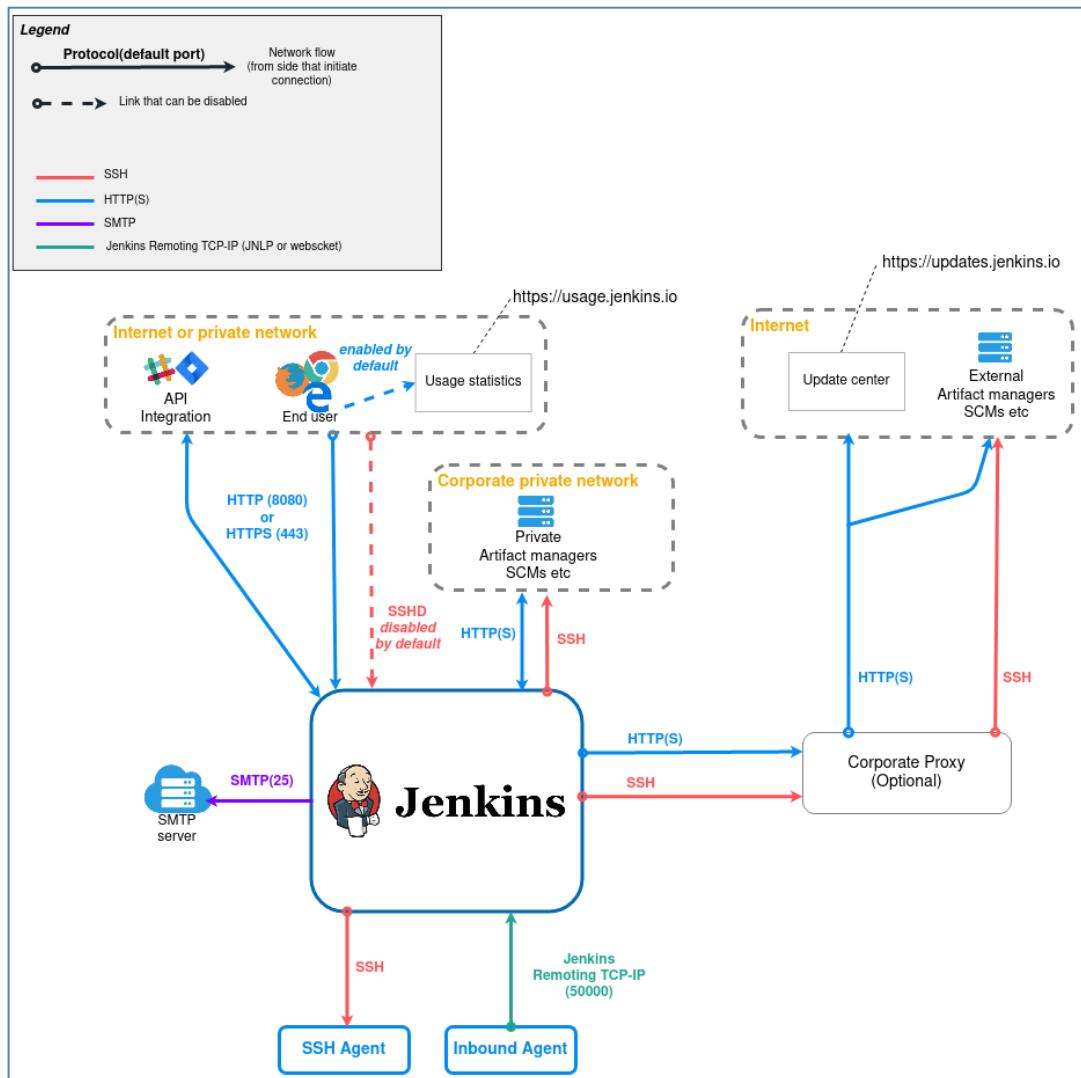


Hình 9 – Logo Jenkins

Tổng quan: Jenkins là một máy chủ tự động hóa mã nguồn mở được viết bằng Java, hỗ trợ các nhà phát triển trong việc xây dựng, kiểm thử và triển khai phần mềm một cách liên tục. Nó đóng vai trò quan trọng trong việc thực hiện các quy trình tích hợp

liên tục (CI) và triển khai liên tục (CD), giúp tăng tốc độ phát triển phần mềm và cải thiện chất lượng sản phẩm.

Kiến trúc của Jenkins:



Hình 10 – Kiến trúc của Jenkins

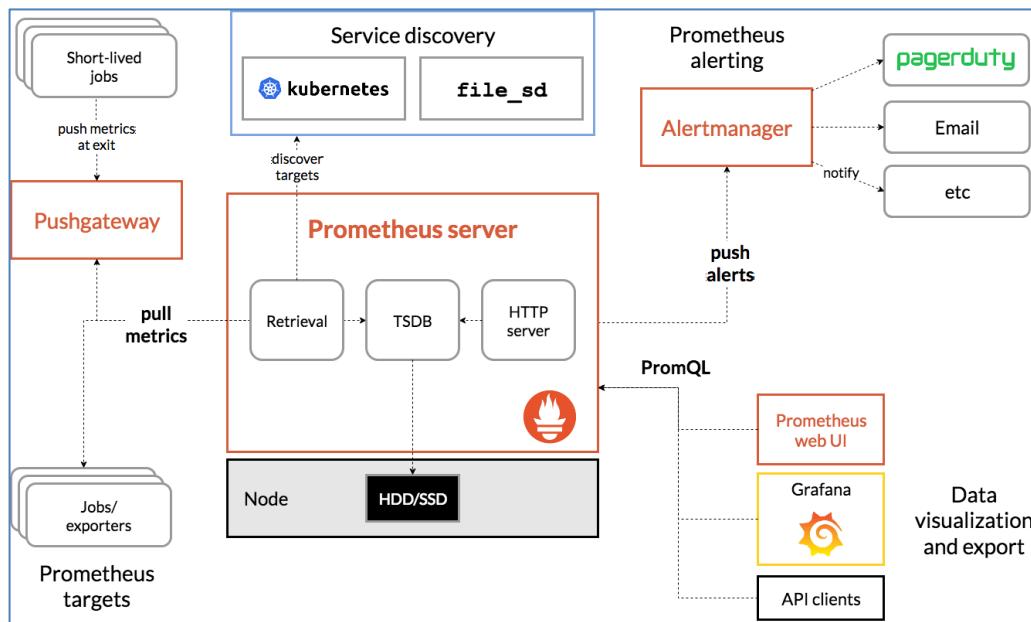
1.1.8 Prometheus



Hình 11 – Logo của Prometheus

Tổng quan: Prometheus là một hệ thống giám sát và cảnh báo mã nguồn mở được phát triển bởi SoundCloud. Công cụ này được thiết kế để thu thập, lưu trữ và truy vấn các chỉ số (metrics) từ các ứng dụng và hạ tầng, giúp phát hiện sớm các vấn đề và hỗ trợ tối ưu hóa hệ thống.

Kiến trúc của Prometheus:



Hình 12 – Kiến trúc của Prometheus

Lợi ích khi sử dụng:

- Cung cấp khả năng giám sát chi tiết và thời gian thực cho các ứng dụng và hạ tầng.
- Hỗ trợ giám sát trên quy mô lớn, đặc biệt trong các môi trường container hóa như Kubernetes.
- Cung cấp cảnh báo và các chỉ số quan trọng, giúp nhanh chóng phát hiện và khắc phục sự cố.
- Hỗ trợ tích hợp tốt với các công cụ DevOps như Grafana, Kubernetes, Docker và các exporters đa dạng.

1.1.9 Grafana



Hình 13 – Logo của Grafana

Tổng quan: Grafana là một nền tảng mã nguồn mở cho phép hiển thị và phân tích dữ liệu thời gian thực từ nhiều nguồn khác nhau. Cung cấp khả năng tạo các bảng điều khiển (dashboard) tùy chỉnh, giúp người dùng giám sát và hiểu rõ hơn về hiệu suất của hệ thống, ứng dụng và cơ sở hạ tầng.

Lợi ích khi sử dụng:

- Các dashboard trực quan giúp người dùng dễ dàng nhận diện xu hướng và bắt thường trong dữ liệu, hỗ trợ việc ra quyết định nhanh chóng.
- Khả năng cập nhật dữ liệu liên tục cho phép theo dõi hiệu suất hệ thống và ứng dụng một cách kịp thời, giúp phát hiện và xử lý sự cố nhanh chóng.
- Khả năng kết nối với nhiều nguồn dữ liệu khác nhau giúp Grafana trở thành công cụ mạnh mẽ trong việc tổng hợp và hiển thị thông tin từ nhiều hệ thống.
- Tích hợp dễ dàng với các công cụ CI/CD như Jenkins, Azure DevOps, giúp tự động hóa quá trình kiểm tra chất lượng mã trong chu kỳ phát triển phần mềm.

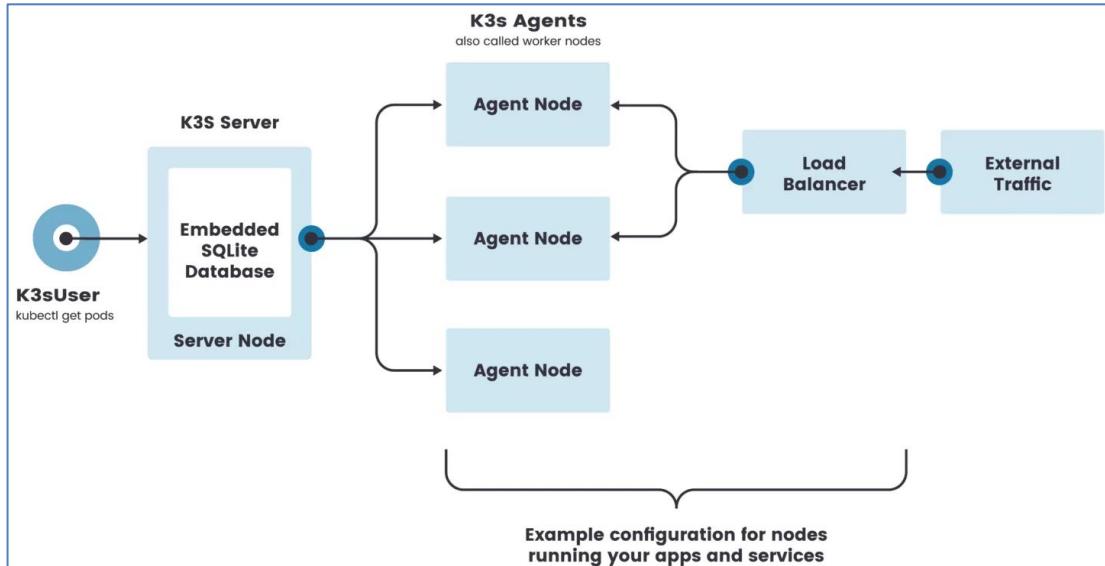
1.1.10. K3S



Hình 14 – Logo K3S

Tổng quan: K3s là một phiên bản nhẹ và dễ cài đặt của Kubernetes (K8s), được phát triển bởi Rancher Labs. Nó được thiết kế để hoạt động hiệu quả trên các thiết bị có tài nguyên hạn chế, như Raspberry Pi, và trong các môi trường biên (edge) hoặc IoT. K3s giữ nguyên các chức năng cốt lõi của Kubernetes nhưng loại bỏ những thành phần không cần thiết, giúp giảm dung lượng và tài nguyên sử dụng.

Kiến trúc của K3s:



Hình 15 – Kiến trúc của K3s

Kiến trúc của K3s có 2 thành phần chính:

K3s Server:

- **Embedded SQLite Database:** Thay vì sử dụng etcd như K8s, K3s sử dụng SQLite làm cơ sở dữ liệu mặc định, giúp giảm tải tài nguyên và đơn giản hóa cấu hình.
- **Server Node:** Chịu trách nhiệm điều phối các tác vụ trong cluster, quản lý trạng thái và phân phối công việc đến các Agent Node.
- Người dùng sử dụng công cụ `kubectl` để gửi yêu cầu và tương tác với K3s Server.

K3s Agents (Worker Nodes):

- **Agent Nodes:** Là nơi chạy các ứng dụng và dịch vụ trong cluster. Mỗi Agent Node nhận công việc từ K3s Server và thực thi các pod tương ứng.
- **Load Balancer:** Điều phối lưu lượng mạng đến các Agent Node, đảm bảo cân bằng tải và cung cấp khả năng mở rộng hiệu quả.
- **External Traffic:** Các dịch vụ được phai bày ra bên ngoài thông qua Load Balancer, đảm bảo kết nối ổn định và linh hoạt với các ứng dụng bên ngoài.

Lợi ích khi sử dụng:

- K3s có kích thước nhị phân dưới 100MB và yêu cầu dưới 512MB RAM, giúp triển khai nhanh chóng và tiết kiệm tài nguyên.
- Quá trình cài đặt đơn giản với một lệnh duy nhất, phù hợp cho người mới bắt đầu và các môi trường yêu cầu triển khai nhanh.
- Hỗ trợ các thiết bị sử dụng kiến trúc ARM, như Raspberry Pi, mở rộng khả năng ứng dụng trong các dự án IoT.

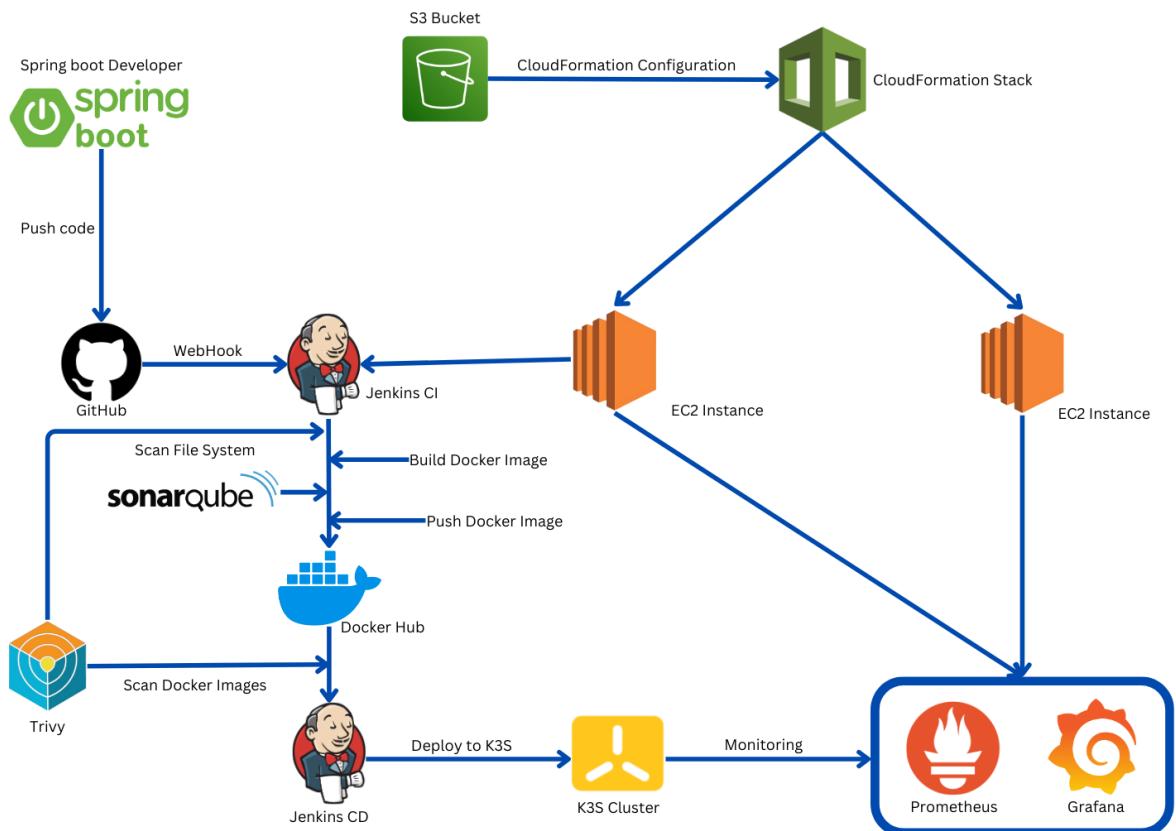
Nên sử dụng K3s khi:

- Môi trường tài nguyên hạn chế.
- Nơi cần thiết lập nhanh chóng và không yêu cầu toàn bộ chức năng của Kubernetes.

Nên sử dụng K8s khi:

- Quản lý hàng nghìn node và pod với yêu cầu phức tạp.
- Sử dụng các tính năng và tích hợp nâng cao mà K3s có thể không hỗ trợ đầy đủ.

2. MÔ HÌNH HỆ THỐNG



Hình 16 – Mô hình của hệ thống

CÁC THÀNH PHẦN CHÍNH

1. Spring Boot

- Nền tảng phát triển chính, được sử dụng để xây dựng ứng dụng microservices.
- Mã nguồn được lưu trữ và được push lên repository.

2. SonarQube

- Công cụ kiểm tra chất lượng mã nguồn.
- Phân tích mã nguồn từ ứng dụng Spring Boot, phát hiện lỗi tiềm ẩn, cải thiện chất lượng mã.

3. Jenkins

- Công cụ CI/CD đảm bảo tự động hóa quá trình build, kiểm tra, và triển khai.
- Jenkins thực hiện các bước sau:
 - Lấy mã nguồn từ kho lưu trữ và build ứng dụng microservices.
 - Tích hợp với SonarQube để phân tích mã.
 - Đóng gói ứng dụng Spring Boot thành image Docker.
 - Đẩy image lên container registry.

4. AWS CloudFormation

- Công cụ tự động hóa việc triển khai cơ sở hạ tầng trên AWS, bao gồm EC2, S3, và các tài nguyên khác.
- Lưu trữ file cấu hình hạ tầng trong S3 để đảm bảo khả năng tái sử dụng và quản lý tốt hơn.

5. Prometheus & Grafana

- Prometheus: Thu thập các số liệu từ các dịch vụ đang chạy.

- Grafana: Cung cấp giao diện trực quan để hiển thị và giám sát hiệu năng hệ thống.

3. QUY TRÌNH THỰC HIỆN

3.1 Xây dựng hạ tầng với CloudFormation

3.1.1 vpc.yaml

```

4  ↵ Parameters:
5   ↵   VpcCIDR:
6     |   Type: String
7   ↵   PublicSubnetCIDR:
8     |   Type: String
9   ↵   AvailabilityZone:
10    |   Type: String
11   ↵   PublicIP:
12     |   Type: String
13     |   Default: "0.0.0.0/0"
14   ↵   Environment:
15     |   Type: String
16     |   Default: "nhom14-project"
17
18   ↵ Resources:
19   ↵   MyVPC:
20     |   Type: AWS::EC2::VPC
21   ↵   Properties:
22     |   CidrBlock: !Ref VpcCIDR
23     |   EnableDnsSupport: true
24     |   EnableDnsHostnames: true
25   ↵   Tags:
26   ↵     - Key: Name
27     |     Value: !Sub "${Environment}-vpc"

```

Vpc

Tạo VPC với các cấu hình sau:

- Dải CIDR cho VPC và subnet công khai có thể được chỉ định khi triển khai.
- Bật hỗ trợ DNS và tên máy chủ cho VPC.
- Tạo Tags với tên được cấu hình dựa trên tham số Environment

```

PublicSubnet:
  Type: AWS::EC2::Subnet
Properties:
  VpcId: !Ref MyVPC
  CidrBlock: !Ref PublicSubnetCIDR
  AvailabilityZone: !Ref AvailabilityZone
  MapPublicIpOnLaunch: true
  Tags:
    - Key: Name
      Value: !Sub "${Environment}-public-subnet"

```

Public Subnet

Tạo **Public Subnet** trong VPC với cấu hình:

- CIDR block và VPC từ các tham số đã định nghĩa trong VPC.
- Availability Zone được chỉ định là us-east-1a.
- MapPublicOnLaunch: Các tài nguyên trong subnet sẽ nhận địa chỉ IP công cộng tự động khi được tạo.

```

41  InternetGateway:
42    Type: AWS::EC2::InternetGateway
43    Properties:
44      Tags:
45        - Key: Name
46          Value: !Sub "${Environment}-igw"

```

Internet Gateway

Tạo **Internet Gateway (IGW)** cho VPC, cho phép các tài nguyên trong VPC có thể giao tiếp với Internet.

```

48  AttachGateway:
49    Type: AWS::EC2::VPCGatewayAttachment
50    Properties:
51      VpcId: !Ref MyVPC
52      InternetGatewayId: !Ref InternetGateway

```

Attach VPC Internet Gateway

Tạo **VPCGatewayAttachment** để gắn **Internet Gateway** vào **VPC**. Điều này cho phép các tài nguyên trong VPC giao tiếp với Internet qua Internet Gateway đã được

tạo trước đó. Sau khi gắn kết, các tài nguyên trong VPC có thể truy cập Internet thông qua IGW.

```
54 |   PublicRouteTable:  
55 |     Type: AWS::EC2::RouteTable  
56 |     Properties:  
57 |       VpcId: !Ref MyVPC  
58 |       Tags:  
59 |         - Key: Name  
60 |           Value: !Sub "${Environment}-public-rtb"
```

Public Route Table

Tạo **Route Table** cho VPC, giúp định tuyến lưu lượng mạng trong VPC và kết nối các tài nguyên trong subnet công khai với Internet qua Internet Gateway (IGW).

```
61 |  
62 |   PublicRoute:  
63 |     Type: AWS::EC2::Route  
64 |     DependsOn: AttachGateway  
65 |     Properties:  
66 |       RouteTableId: !Ref PublicRouteTable  
67 |       DestinationCidrBlock: "0.0.0.0/0"  
68 |       GatewayId: !Ref InternetGateway
```

Public Route

Tạo route trong **Route Table** cho subnet công khai để định tuyến lưu lượng mạng ra ngoài Internet thông qua **Internet Gateway**.

- **Tạo Route:** Tạo một route trong PublicRouteTable, cho phép các tài nguyên trong subnet công khai có thể giao tiếp với Internet.
- **DestinationCidrBlock:** 0.0.0.0/0 nghĩa là tất cả lưu lượng mạng sẽ được định tuyến ra Internet.
- **GatewayId:** Route này trỏ đến **Internet Gateway**, cho phép lưu lượng đi ra ngoài Internet

```
70  PublicSubnetRouteTableAssociation:  
71    Type: AWS::EC2::SubnetRouteTableAssociation  
72    Properties:  
73      SubnetId: !Ref PublicSubnet  
74      RouteTableId: !Ref PublicRouteTable
```

Subnet Route Table Association

Tạo **SubnetRouteTableAssociation**, giúp **gắn Route Table vào Public Subnet**. Điều này là cần thiết để các tài nguyên trong subnet công khai có thể sử dụng các route đã định tuyến trong Route Table, đặc biệt là để truy cập Internet thông qua Internet Gateway.

- **Gắn Route Table vào Subnet:** Subnet công khai sẽ sử dụng Route Table đã tạo để định tuyến lưu lượng mạng.
- **Áp dụng đúng Route Table:** Việc gắn đúng Route Table cho subnet giúp các tài nguyên trong subnet có thể truy cập Internet và các tài nguyên khác trong VPC.

```

76    JenkinsK3SSecurityGroup:
77        Properties:
78            VpcId: !Ref MyVPC
79            SecurityGroupIngress:
80                - IpProtocol: tcp
81                    FromPort: 22
82                    ToPort: 22
83                    CidrIp: !Ref PublicIP
84                - IpProtocol: tcp
85                    FromPort: 6443
86                    ToPort: 6443
87                    CidrIp: !Ref PublicIP
88                - IpProtocol: tcp
89                    FromPort: 9000
90                    ToPort: 9000
91                    CidrIp: !Ref PublicIP
92                - IpProtocol: tcp
93                    FromPort: 8080
94                    ToPort: 8080
95                    CidrIp: !Ref PublicIP
96                - IpProtocol: tcp
97                    FromPort: 30000
98                    ToPort: 32767
99                    CidrIp: !Ref PublicIP
100
101        Tags:
102            - Key: Name
103            Value: !Sub "${Environment}-jenkins-k3s-sg"

```

Security Group Jenkins k3s instance

Tạo **Security Group** cho Jenkins và K3S, cho phép truy cập vào các cổng sau từ các địa chỉ IP công cộng:

- Port 22 (SSH) cho phép quản lý qua SSH.
- Port 6443 cho phép truy cập vào API của K3S.
- Port 9000 cho phép truy cập HTTP vào Sonarqube.
- Port 8080 cho dịch vụ HTTP của Jenkins.
- Dải Port từ 30000 đến 32767 cho **LoadBalancer** trong K3S. Cho phép các dịch vụ được truy cập từ bên ngoài.

```

106 | GrafanaPrometheusSecurityGroup:
107 |   Type: AWS::EC2::SecurityGroup
108 |   Properties:
109 |     GroupDescription: Allow access and port range for load balancer exposure
110 |     VpcId: !Ref MyVPC
111 |     SecurityGroupIngress:
112 |       - IpProtocol: tcp
113 |         FromPort: 22
114 |         ToPort: 22
115 |         CidrIp: !Ref PublicIP
116 |       - IpProtocol: tcp
117 |         FromPort: 9090
118 |         ToPort: 9090
119 |         CidrIp: !Ref PublicIP
120 |       - IpProtocol: tcp
121 |         FromPort: 9100
122 |         ToPort: 9100
123 |         CidrIp: !Ref PublicIP
124 |       - IpProtocol: tcp
125 |         FromPort: 3000
126 |         ToPort: 3000
127 |         CidrIp: !Ref PublicIP
128 |       - IpProtocol: tcp
129 |         FromPort: 30000
130 |         ToPort: 32767
131 |         CidrIp: !Ref PublicIP
132 |     Tags:
133 |       - Key: Name
134 |         Value: !Sub "${Environment}-grafana-prometheus-sg"

```

Security Group for Grafana Prometheus

Tạo **Security Group** cho các dịch vụ **Grafana** và **Prometheus**, cho phép truy cập vào các cổng cần thiết từ các địa chỉ IP công cộng:

- Port 22 cho SSH.
- Port 9090 cho giao diện người dùng Prometheus.
- Port 9100 cho Prometheus Node Exporter.
- Port 3000 cho giao diện người dùng Grafana.
- Dải Port 30000-32767 cho **LoadBalancer** trong K3S. Lấy metrics từ các services trong K3S.

3.1.2 ec2.yaml

```
21 Resources:
22   JenkinsK3sEC2:
23     Type: AWS::EC2::Instance
24     Properties:
25       InstanceType: !Ref INSTANCELARGE
26       ImageId: !Ref AMI
27       SubnetId: !Ref PublicSubnetId
28       SecurityGroupIds:
29         - !Ref JenkinsK3sSecurityGroup
30       BlockDeviceMappings:
31         - DeviceName: /dev/sda1
32           Ebs:
33             VolumeSize: 25
34             VolumeType: gp2
35             DeleteOnTermination: false
36       Tags:
37         - Key: Name
38           Value: !Sub "${Environment}-jenkins-k3s"
39
```

Jenkins K3S EC2

Tạo EC2 instance để triển khai **Jenkins** và **K3s**.

- **Instance Type:** t3.large
- **Ami:** Chỉ định hệ điều hành sử dụng là Ubuntu
- **Subnet:** Sử dụng PublicSubnetId tạo từ VPC stack.
- **Security Group:** Allow port 22 (SSH), 6443 (K3S), 8080 (Jenkins), 9000 (SonarQube), 30000-32767 (Đầu port LoadBalancer) đã cấu hình ở VPC Stack
- **EBS volume:** Dung lượng 25 GB được gắn vào instance, sử dụng loại gp2.

```

42 |     PrometheusGrafanaEC2:
43 |       Type: AWS::EC2::Instance
44 |       Properties:
45 |         InstanceType: !Ref INSTANCEMEDIUM
46 |         ImageId: !Ref AMI
47 |         SubnetId: !Ref PublicSubnetId
48 |         SecurityGroupIds:
49 |           - !Ref GrafanaPrometheusSecurityGroup
50 |         BlockDeviceMappings:
51 |           - DeviceName: /dev/sda1
52 |             Ebs:
53 |               VolumeSize: 15
54 |               VolumeType: gp2
55 |               DeleteOnTermination: false
56 |         Tags:
57 |           - Key: Name
58 |             Value: !Sub "${Environment}-grafana-prometheus"
59 |

```

Prometheus Grafana EC2

Tạo một **EC2 instance** cho **Prometheus** và **Grafana** với cấu hình:

- **Instance Type:** t3.medium
- **Ami:** Chỉ định hệ điều hành sử dụng là Ubuntu
- **Subnet:** Sử dụng PublicSubnetId tạo từ VPC stack.
- **Security Group:** Allow port 22 (SSH), 9090 (Prometheus), 9100 (Node Exporter), 3000 (Grafana), 30000-32767 (Dải port LoadBalancer) đã cấu hình ở VPC stack
- **EBS volume:** Dung lượng 15 GB được gắn vào instance, sử dụng loại gp2.

3.1.3 main.yaml

```
1 Parameters:
2   VpcCIDR:
3     Type: String
4     Default: "192.168.0.0/16"
5   PublicSubnetCIDR:
6     Type: String
7     Default: "192.168.1.0/24"
8   AvailabilityZone:
9     Type: String
10    Default: "us-east-1a"
11   BucketName:
12     Type: String
13     Default: "nhom14projects3bucket"      You, 2 days ago • Build infrastructure with CloudFormation
14
15 Resources:
16   VPCStack:
17     Type: AWS::CloudFormation::Stack
18     Properties:
19       TemplateURL: !Sub "https://${BucketName}.s3.us-east-1.amazonaws.com/vpc.yaml"
20       Parameters:
21         VpcCIDR: !Ref VpcCIDR
22         PublicSubnetCIDR: !Ref PublicSubnetCIDR
23         AvailabilityZone: !Ref AvailabilityZone
24
25   EC2Stack:
26     Type: AWS::CloudFormation::Stack
27     Properties:
28       TemplateURL: !Sub "https://${BucketName}.s3.us-east-1.amazonaws.com/ec2.yaml"
29       Parameters:
30         PublicSubnetId: !GetAtt VPCStack.Outputs.PublicSubnetId
31         JenkinsK3sSecurityGroup: !GetAtt VPCStack.Outputs.K3sJenkinsSgId
32         GrafanaPrometheusSecurityGroup: !GetAtt VPCStack.Outputs.GrafanaPrometheusSgId
33
```

Call vpc stack and ec2 stack

Định nghĩa hai stack con: VPCStack và EC2Stack. Các thông số và tài nguyên được mô tả như sau:

- **VPCStack** tạo VPC, subnet và các tài nguyên liên quan (như security group).
- **EC2Stack** tạo EC2 instances cho Jenkins, K3s, Grafana, và Prometheus, và cấu hình các security group phù hợp.

```
review-company
cloudformation
cloudformation
cloudformation
cd .\cloudformation\
cfn-lint.exe .\vpc.yaml
cfn-lint.exe .\ec2.yaml
cfn-lint.exe .\main.yaml
```

Scan code with cfn-lint

Sử dụng cfn-lint để quét mã nguồn cho các file vpc.yaml, ec2.yaml và main.yaml trước khi triển khai lên CloudFormation.

3.2 Triển Khai CloudFormation

3.2.1 Tạo S3 Bucket

The screenshot shows the AWS S3 console with the 'Objects' tab selected. The bucket name is 'nhom14projects3bucket'. There are three objects listed:

Name	Type	Last modified	Size	Storage class
ec2.yaml	yaml	November 30, 2024, 07:29:38 (UTC+07:00)	1.5 KB	Standard
main.yaml	yaml	November 30, 2024, 07:29:39 (UTC+07:00)	1.3 KB	Standard
vpc.yaml	yaml	November 30, 2024, 07:29:40 (UTC+07:00)	3.5 KB	Standard

Tạo S3 Bucket

Tạo S3 Bucket và đặt tên giống với Bucket đã khai báo trong main.yaml.

Upload các file cấu hình main.yaml vpc.yaml và ec2.yaml lên bucket

3.2.2 Tạo Stack

The screenshot shows the AWS CloudFormation console with the 'Stacks' tab selected. There are four stacks listed:

Stack name	Status	Created time	Description
nhom14-stack-EC2Stack-1K3AW2MFP5QMO NESTED	CREATE_COMPLETE	2024-11-30 07:31:09 UTC+0700	-
nhom14-stack-VPCStack-1XPUAN9XHOVW6 NESTED	CREATE_COMPLETE	2024-11-30 07:30:34 UTC+0700	VPC Module
nhom14-stack	CREATE_COMPLETE	2024-11-30 07:30:31 UTC+0700	-

Tạo Stack CloudFormation

Tạo Stack CloudFormation sử dụng Nested Stack.

Stacks (4)

Resources (2)

Logical ID	Physical ID	Type	Status
EC2Stack	arn:aws:cloudformation:us-east-1:637423577788:stack/nhom14-stack-EC2Stack-1K3AW2MFP5QMO/64f738f0-aeb2-11ef-97f3-0ef338eb6921	AWS::CloudFormation::Stack	CREATE_COMPLETE
VPCStack	arn:aws:cloudformation:us-east-1:637423577788:stack/nhom14-stack-VPCStack-1XPUAN9XHOVW6/503e8120-aeb2-11ef-8928-0affde745fe7	AWS::CloudFormation::Stack	CREATE_COMPLETE

Nested Stack

Truyền vào URL của file main.yaml nó sẽ tự động gọi đến vpc.yaml và ec2.yaml để tạo cơ sở hạ tầng.

Stacks (4)

Resources (9)

Logical ID	Physical ID	Type	Status
AttachGateway	IGWvpc-0c0b941d7984513cf	AWS::EC2::VP CGatewayAttachment	CREATE_COMPLETE
GrafanaPrometheusSecurityGroup	sg-06c778509773af165	AWS::EC2::SecurityGroup	CREATE_COMPLETE
InternetGateway	igw-09cecc0e01126948a	AWS::EC2::InternetGateway	CREATE_COMPLETE
JenkinsSSecurityGroup	sg-048a5781452fc76b3	AWS::EC2::SecurityGroup	CREATE_COMPLETE
MyVPC	vpc-0c0b941d7984513cf	AWS::EC2::VPC	CREATE_COMPLETE
PublicRoute	rtb-020b39189ceef3672 0.0.0/0	AWS::EC2::Route	CREATE_COMPLETE
PublicRouteTable	rtb-020b39189ceef3672	AWS::EC2::RouteTable	CREATE_COMPLETE
PublicSubnet	subnet-0842226e145ae8fa	AWS::EC2::Subnet	CREATE_COMPLETE
PublicSubnetRouteTableAssociation	rtbasoc-0b8726c6d733b499e	AWS::EC2::SubnetRouteTableAssociation	CREATE_COMPLETE

Resources của VPC

Các Resource được tạo trong VPC bao gồm:

- VPC:** Tạo mạng riêng trong AWS
- Public Subnet:** Dùng để truy cập trực tiếp Internet
- Internet Gateway:** Cho phép các tài nguyên trong Subnet có thể kết nối Internet

- **Route Table:** Định tuyến mạng giữa các subnet trong VPC, giao tiếp ra ngoài Internet thông qua Internet Gateway
- **Security Group:** Allow các port cho jenkins sonar, k3s, ...

The screenshot shows two AWS console pages side-by-side.

Left Panel (CloudFormation):

- Stacks (4)**: Shows a list of stacks, with one stack highlighted:
 - Logical ID:** nhom14-stack-EC2Stack-1K3AW2MFP5QMO
 - Created:** 2024-11-30 07:31:09 UTC+0700
 - Status:** CREATE_COMPLETE
- Filter status:** Active

Right Panel (AWS Lambda):

- nhom14-stack-EC2Stack-1K3AW2MFP5QMO**: Stack info page for the nested stack.
- Resources (2)**: List of resources created by the stack:

Logical ID	Physical ID	Type	Status
JenkinsK3sEC2	i-05d191140dfe1608a	AWS::EC2::Instance	CREATE_COMPLETE
PrometheusGrafanaEC2	i-0bca70df0938674c3	AWS::EC2::Instance	CREATE_COMPLETE

Resource EC2

Các tài nguyên được tạo bao gồm 2 máy EC2

- **JenkinsK3sEC2:** Cài đặt cấu hình Jenkins, K3s, SonarQube
- **PrometheusGrafanaEC2:** Cài đặt và cấu hình Grafana, Prometheus.

3.2.3 Kiểm tra cấu hình trên AWS Console

The screenshot shows the AWS Instances page displaying two running EC2 instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
nhom14-project-jenkins-k3s	i-05d191140dfe1608a	Running	t3.large	3/3 checks passed	View alarms +
nhom14-project-grafana-prometheus	i-0bca70df0938674c3	Running	t3.medium	3/3 checks passed	View alarms +

Ec2 Instance

JenkinsK3sEC2: t3.large tạo bởi CloudFormation

PrometheusGrafanaEC2: t3.medium tạo bởi CloudFormation

Security Groups (3) [Info](#)

[Actions](#) [Export security groups to CSV](#) [Create security group](#)

Find resources by attribute or tag

VPC ID = vpc-0c0b941d7984513cf [X](#) [Clear filters](#)

<input type="checkbox"/>	Name	Security group ID	Security group name	VPC ID	Desc
<input type="checkbox"/>	nhom14-project-jenkins-k3s-sg	sg-048a5781452fc76b3	nhom14-stack-VPCStack-1XP...	vpc-0c0b941d7984513cf	Allov
<input type="checkbox"/>	-	sg-00ed66cc59beb6d35	default	vpc-0c0b941d7984513cf	defa
<input type="checkbox"/>	nhom14-project-grafana-prometheus-sg	sg-06c778509773af165	nhom14-stack-VPCStack-1XP...	vpc-0c0b941d7984513cf	Allov

Security Group tạo bởi CloudFormation

sg-048a5781452fc76b3 - nhom14-stack-VPCStack-1XPUAN9XHOVW6-JenkinsK3SSecurityGroup-Jk7TQMIE1Apd

[Actions](#)

Details

Security group name nhom14-stack-VPCStack-1XPUAN9XHOVW6-JenkinsK3SSecurityGroup-Jk7TQMIE1Apd	Security group ID sg-048a5781452fc76b3	Description Allow access and port range for load balancer exposure	VPC ID vpc-0c0b941d7984513cf
Owner 637423577788	Inbound rules count 5 Permission entries	Outbound rules count 1 Permission entry	

[Inbound rules](#)

[Outbound rules](#)

[Sharing - new](#)

[VPC associations - new](#)

[Tags](#)

Inbound rules (5)

[Manage tags](#)

[Edit inbound rules](#)

Search

JenkinsK3sEC2 Security Group Rule

Allow các port gồm:

- 9000: SonarQube
- 8080: Jenkins
- 22: SSH
- 6443: K3S
- 30000-32767: LoadBalancer
- Ngoài ra còn có 9100 cho Node Exporter

sg-06c778509773af165 - nhom14-stack-VPCStack-1XPUAN9XHOVW6-GrafanaPrometheusSecurityGroup-vdOfJ184jkSC

Actions ▾

Details			
Security group name sg-06c778509773af165	Security group ID sg-06c778509773af165	Description Allow access and port range for load balancer exposure	VPC ID vpc-0c0b941d7984513cf
Owner 637423577788	Inbound rules count 5 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules Outbound rules Sharing - new VPC associations - new Tags

Inbound rules (5)						
<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range
<input type="checkbox"/>	-	sgr-0bb18ccc645c1d97f	IPv4	Custom TCP	TCP	9090
<input type="checkbox"/>	-	sgr-0c15521e33bc84e80	IPv4	SSH	TCP	22
<input type="checkbox"/>	-	sgr-0ca53d1ddcb3fd44a	IPv4	Custom TCP	TCP	30000 - 32767
<input type="checkbox"/>	-	sgr-0265136d9541c35f3	IPv4	Custom TCP	TCP	3000
<input type="checkbox"/>	-	sgr-028b6ac1a81916f35	IPv4	Custom TCP	TCP	9100

PrometheusGrafanaEC2 Security Group Rule

Allow các port gồm:

- 9090: Prometheus
- 3000: Grafana
- 22: SSH
- 30000-32767: LoadBalancer
- Ngoài ra còn có 9100 cho Node Exporter

3.3 Cấu hình SonarQube

3.3.1 Tạo token

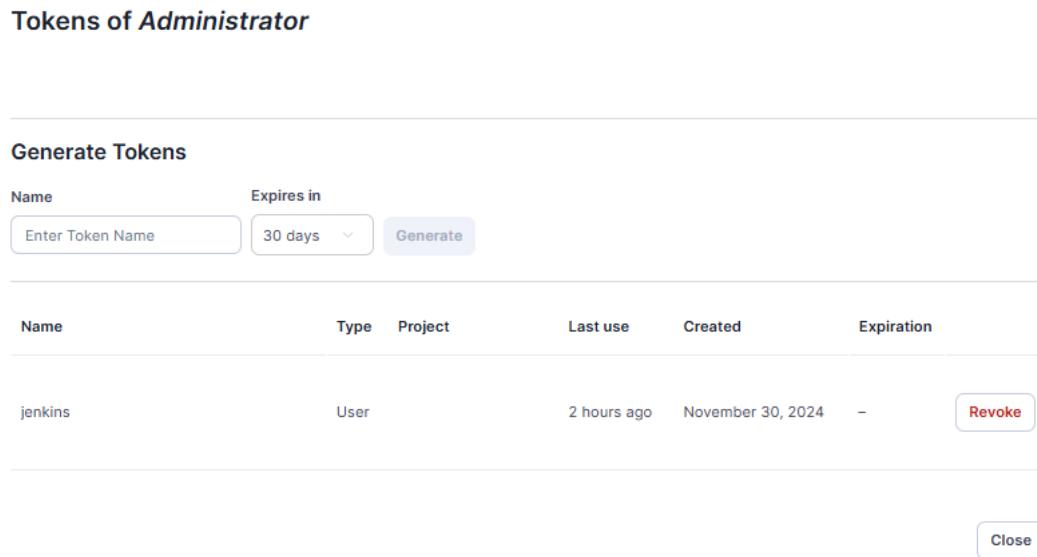
Tokens of Administrator

Generate Tokens

Name	Expires in
Enter Token Name	30 days

Name	Type	Project	Last use	Created	Expiration
jenkins	User		2 hours ago	November 30, 2024	-

[Close](#)



Tạo token cho Jenkins

Tạo Token Trong SonarQube

- Truy cập Public IP của EC2 với port 9000 sau đó đăng nhập để truy cập vào SonarQube.
- Click vào Phần Administrator → Users → Token → Generate Token để tạo token
- Nhập token_name là jenkins.
- Token này sẽ sử dụng để có thể tích hợp với Jenkins và dùng cho Sonnar Scanner quét các dự án.

3.3.2 Gán token cho các microservices

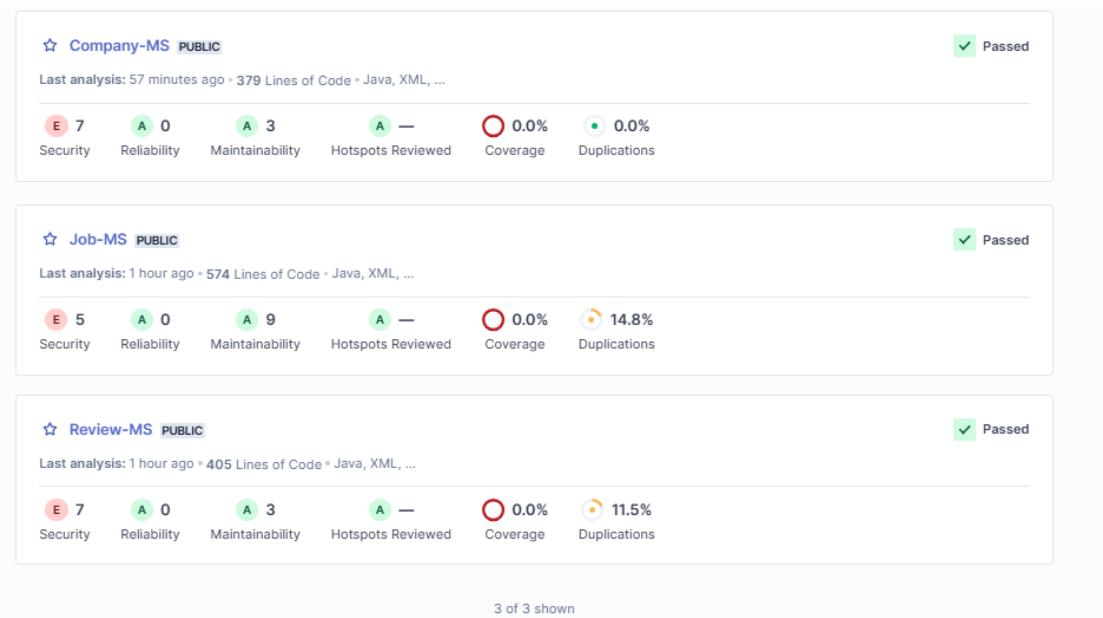
```
companyms > ⚒ sonar-project.properties
    You, 5 hours ago | 1 author (You)
1 sonar.projectKey=Company-MS
2 sonar.login=squ_9719a4fc5a539feea9bf0571736b67ce2ad84344

jobms > ⚒ sonar-project.properties
    You, 5 hours ago | 1 author (You)
1 sonar.projectKey=Job-MS
2 sonar.login=squ_9719a4fc5a539feea9bf0571736b67ce2ad84344

reviewms > ⚒ sonar-project.properties
    You, 5 hours ago | 1 author (You)
1 sonar.projectKey=Review-MS
2 sonar.login=squ_9719a4fc5a539feea9bf0571736b67ce2ad84344
```

Gán token cho các Microservices

Gán các token tạo trong Sonarqube cho file sonar-project.properties cho từng Microservices.



Tạo project trong SonarQube

Tạo các project trong SonarQube với tên của Project tương ứng với sonar.projectKey đã khai báo trong file sonar-project.properties ở các Microservices.

3.4 Cấu hình K3S

3.4.1 account.yaml

```
1  apiVersion: v1
2  kind: Namespace
3  metadata:
4    name: jenkins
```

Tạo namespace

Tạo một **namespace** tên là jenkins trong Kubernetes. Namespace này giúp phân tách và tổ chức các tài nguyên Kubernetes (như pod, service, deployment) liên quan đến Jenkins trong một không gian riêng biệt.

```
8  apiVersion: v1
9  kind: ServiceAccount
10 metadata:
11   name: jenkins
12   namespace: jenkins
```

Tạo Service Account

Tạo một **ServiceAccount** tên là jenkins trong namespace jenkins. ServiceAccount này sẽ được sử dụng để cấp quyền truy cập cho các tài nguyên Kubernetes mà Jenkins cần, chẳng hạn như truy cập vào các API của Kubernetes để thực hiện các tác vụ như deploy hoặc quản lý các pod.

```
16  apiVersion: v1
17  kind: Secret
18  metadata:
19    name: jenkins-token
20    namespace: jenkins
21  annotations:
22    kubernetes.io/service-account.name: jenkins
23  type: kubernetes.io/service-account-token
```

Tạo Token

Tạo một **Secret** có tên jenkins-token trong namespace jenkins. Secret này lưu trữ token của **ServiceAccount** jenkins, cho phép Jenkins (hoặc bất kỳ ứng dụng nào sử dụng ServiceAccount này) có thể truy cập vào các API của Kubernetes. Token này là

phương tiện để xác thực Jenkins với Kubernetes khi cần quyền truy cập hoặc tương tác với các tài nguyên trong cluster.

```
27  apiVersion: rbac.authorization.k8s.io/v1
28  kind: RoleBinding
29  metadata:
30    name: jenkins-admin-binding
31    namespace: jenkins
32  subjects:
33    - kind: ServiceAccount
34      name: jenkins
35      namespace: jenkins
36  roleRef:
37    kind: ClusterRole
38    name: admin
39    apiGroup: rbac.authorization.k8s.io
```

Tạo Role Binding

Tạo một **RoleBinding** có tên jenkins-admin-binding trong namespace jenkins. RoleBinding này gán quyền **admin** cho **ServiceAccount** jenkins trong namespace jenkins. Điều này cho phép Jenkins có quyền quản trị toàn bộ cluster Kubernetes thông qua quyền của ClusterRole admin.

```
43  apiVersion: rbac.authorization.k8s.io/v1
44  kind: ClusterRoleBinding
45  metadata:
46    name: jenkins-admin-binding
47  subjects:
48    - kind: ServiceAccount
49      name: jenkins
50      namespace: jenkins
51  roleRef:
52    kind: ClusterRole
53    name: cluster-admin
54    apiGroup: rbac.authorization.k8s.io
55
```

Tạo Cluster Role Binding

Tạo một **ClusterRoleBinding** có tên jenkins-admin-binding. ClusterRoleBinding này gán quyền **cluster-admin** cho **ServiceAccount** jenkins trong namespace jenkins. Quyền cluster-admin cho phép Jenkins có quyền quản trị hoàn toàn đối với toàn bộ Kubernetes cluster, bao gồm việc thao tác với các tài nguyên và cấu hình cluster.

3.4.2 Kiểm tra cấu hình

```
root@ip-192-168-1-183:/home/ubuntu/k3s# k3s kubectl config get-contexts
CURRENT  NAME      CLUSTER  AUTHINFO  NAMESPACE
*        default   default   default   jenkins
root@ip-192-168-1-183:/home/ubuntu/k3s# k3s kubectl describe sa
Name:           default
Namespace:      jenkins
Labels:          <none>
Annotations:    <none>
Image pull secrets: <none>
Mountable secrets: <none>
Tokens:          <none>
Events:          <none>

Name:           jenkins
Namespace:      jenkins
Labels:          <none>
Annotations:    <none>
Image pull secrets: <none>
Mountable secrets: <none>
Tokens:          jenkins-token
Events:          <none>
root@ip-192-168-1-183:/home/ubuntu/k3s# k3s kubectl config view
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data: DATA+OMITTED
  server: https://127.0.0.1:6443
  name: default
contexts:
- context:
  cluster: default
  namespace: jenkins
  user: default
  name: default
current-context: default
kind: Config
preferences: {}
users:
- name: default
  user:
    client-certificate-data: DATA+OMITTED
    client-key-data: DATA+OMITTED
root@ip-192-168-1-183:/home/ubuntu/k3s# |
```

Tạo token

Sau khi cấu hình trong file account.yaml ở bước trên thì triển khai bằng lệnh

k3s kubectl apply -f account.yaml

Gán namespace cho context bằng lệnh

k3s kubectl config set-context default –namespace=jenkins

Kiểm tra xem context hiện tại đã được gán đúng namespace chưa bằng lệnh

k3s kubectl config get-contexts

Kiểm tra xem service account đã được gán token chưa bằng lệnh

k3s kubectl describe sa

Kiểm tra cấu hình tổng thể của cluster bằng lệnh

k3s kubectl config view

Sau khi kiểm tra và chắc chắn cấu hình chính xác lấy token bằng lệnh:

k3s kubectl describe secret jenkins-token -n jenkins

3.5 Cấu hình Jenkins

3.5.1 Credential

Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	docker	lambaoduy1310/******** (docker auth)
		System	(global)	k3s	k3s token
		System	(global)	jenkins	sonarqube token

Tạo credential

Tạo các credential gồm:

- Docker: Nhập vào username và password để jenkin có thể login tới Docker Hub
- K3S: Nhập vào token đã tạo ở bước 3.4 để kết nối K3S với Jenkins
- Jenkins: Nhập vào token đã tạo ở bước 3.3 để SonarQube kết nối được với Jenkins

3.5.2 Cấu hình K3s kết nối với Jenkins

Cloud k3s Configuration

Name ?
k3s

Kubernetes URL ?
https://52.20.241.169:6443

Use Jenkins Proxy ?

Kubernetes server certificate key ?

Disable https certificate check ?

Kubernetes Namespace
jenkins

Agent Docker Registry ?

Inject restricted PSS security context in agent container definition ?

Credentials
k3s token

+ Add

Connected to Kubernetes v1.30.6+k3s1

Cấu hình Cloud K3s

Cấu hình Cloud K3S:

- Nhập vào Kubernetes URL là địa chỉ IP server trong config view thay bằng IP Public của EC2 với port 6443
- Nhập namespace là jenkins
- Nhập vào credential là k3s token đã tạo ở 3.5.1

Bấm vào Test Connection để kiểm tra kết nối K3S với Jenkins và nhận được kết quả là Connected to Kubernetes v1.30.6+k3s1

Kết nối thành công.

3.5.3 Cấu hình Sonar Sever

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables

Environment variables

SonarQube installations

List of SonarQube installations

Name

sonar-server

Server URL

Default is <http://localhost:9000>

http://52.20.241.169:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

- none -

+ Add

Advanced ▾

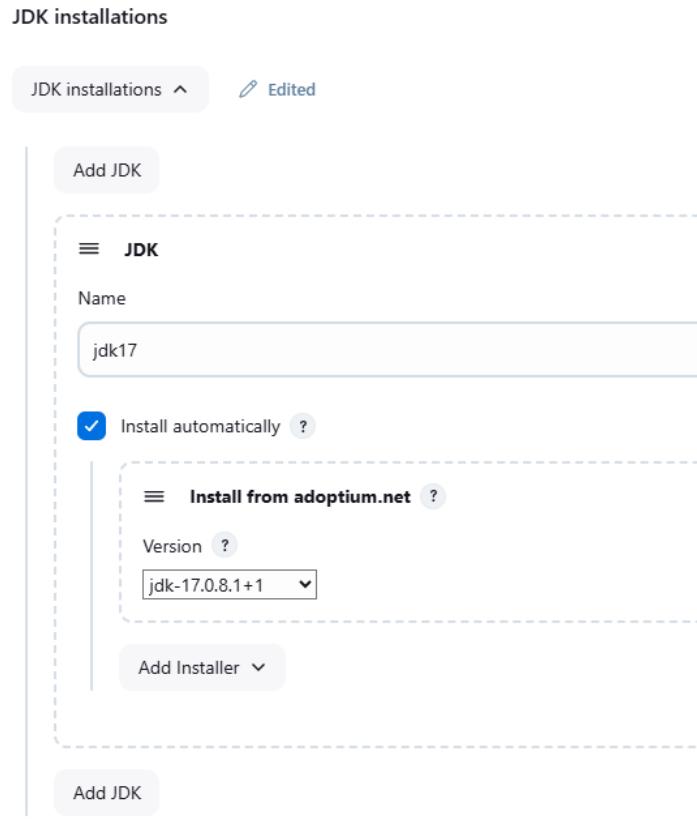
Add SonarQube

Cấu hình cho Sonar Server

Cấu hình cho Sonar server

- Cấu hình name là sonar-server
- Server URL là địa chỉ IP Public của EC2 với Port 9000, đây là nơi mà sonar scanner sẽ quét các dự án đã được cấu hình ở phía trên.

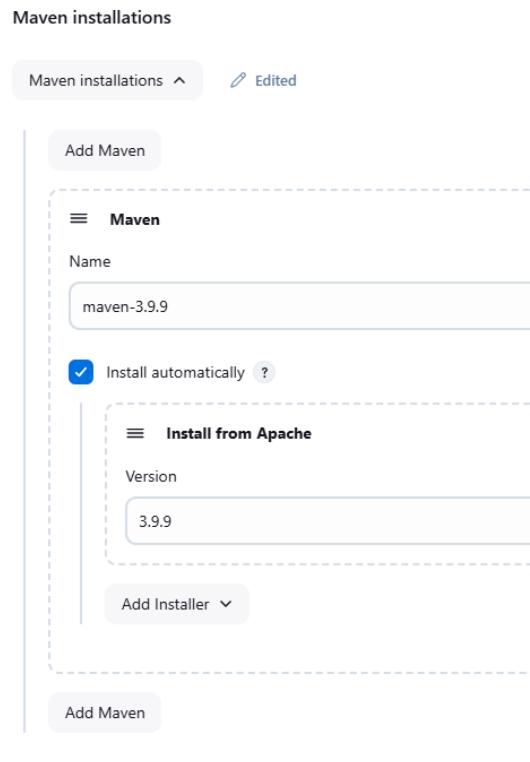
3.5.4 Cài đặt các tools cần thiết trên Jenkins



Cài đặt JDK

Cài đặt JDK cho dự án Microservices Java Spring:

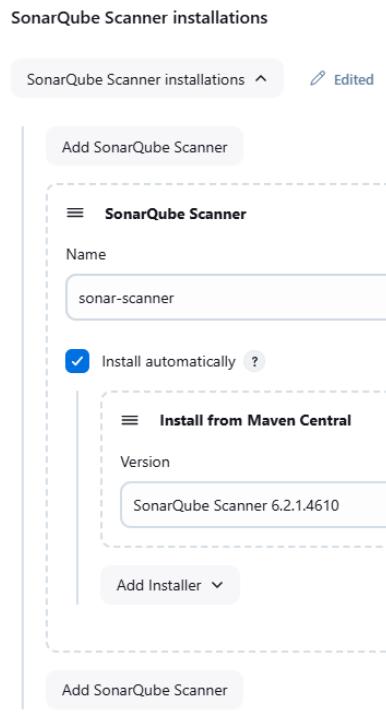
- Name = jdk17
- Chọn install from adoptium.net
- Chọn version =jdk17.0.8.1+1



Cài đặt Maven

Cài đặt Maven để build các dự án Java

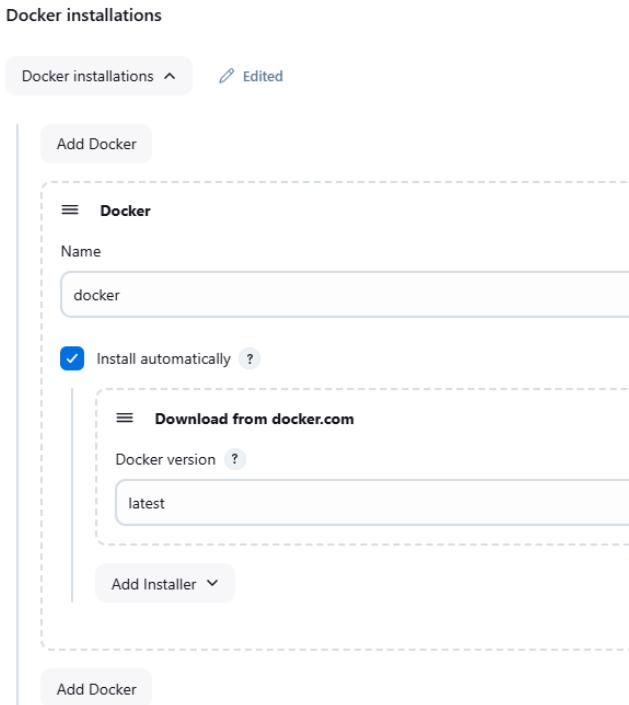
- Name = maven-3.9.9
- Version = 3.9.9



Cài đặt Sonnar Scanner

Cài đặt Sonnar Scanner để quét lỗ hổng cho các dự án

- Name = sonar-scanner
- Version = SonarQube Scanner 6.2.1.4610



Cài đặt Docker

Cài đặt Docker để build các project thành các image

- Name = docker
- Chọn download form docker.com
- Version = latest

3.6 Tạo Pipeline Job và Chạy Pipeline Jenkins

3.6.1 Tạo JenkinsFile cho các Microservices

```
pipeline {
    agent any

    tools {
        jdk 'jdk17'
        nodejs 'node18'
        maven 'maven-3.9.9'
        dockerTool 'docker'
    }
}
```

Khai báo các tools cần thiết

Khai báo các tools cần thiết để build dự án. Tên của các tool phải khớp với tên đã cấu hình các tool trong bước 3.5.4

```

11   environment {
12     SCANNER_HOME      = tool 'sonar-scanner'
13     SONAR_KEY_COMPANY = 'Company-MS'
14     DOCKER_REGISTRY   = 'lambaoduy1310'
15     IMAGE_TAG         = "v4.${BUILD_NUMBER}"
16   }
17
18   environment {
19     SCANNER_HOME      = tool 'sonar-scanner'
20     SONAR_KEY_JOB      = 'Job-MS'
21     DOCKER_REGISTRY   = 'lambaoduy1310'
22     IMAGE_TAG         = "v4.${BUILD_NUMBER}"
23   }
24
25   environment {
26     SCANNER_HOME      = tool 'sonar-scanner'
27     SONAR_KEY REVIEW    = 'Review-MS'
28     DOCKER_REGISTRY   = 'lambaoduy1310'
29     IMAGE_TAG         = "v4.${BUILD_NUMBER}"

```

Khai báo các biến môi trường cho các microservices

- Gán sonar-scanner đã cài ở 3.5.4 thiết lập cho biến môi trường SCANNER_HOME
- Gán username của Docker Hub cho biến môi trường DOCKER_REGISTRY
- Gán version cho biến môi trường IMAGE_TAG, version hiện tại là version 4 và các version cụ thể sẽ dự vào BUILD_NUMBER của Jenkins

```

18   stages {
19     stage('Clean workspace'){
20       steps{
21         cleanWs()
22       }
23     }
24
25     stage('Checkout SCM') {
26       steps {
27         checkout scm
28       }
29     }

```

Clean workspace và Checkout SCM stage

- Tiến hành Clean Workspace để dọn dẹp không gian làm việc trước khi bắt đầu các bước khác trong pipeline.
- cleanWs() sẽ xóa toàn bộ tệp và thư mục trong workspace hiện tại để đảm bảo môi trường sạch sẽ cho quá trình tiếp theo.
- Sử dụng lệnh checkout scm để Jenkins tự động lấy mã nguồn từ repository được chỉ định trong pipeline, bao gồm thông tin từ các file cấu hình Jenkins

```

31      stage('Build Company MS') {
32          steps {
33              dir('companyms') {
34                  script {
35                      sh './mvnw -N io.takari:maven:wrapper || true'
36                      sh 'chmod +x ./mvnw'
37
38                      sh ...
39                      './mvnw spring-boot:build-image -DskipTests \
40                          -Dspring-boot.build-image.imageName=${DOCKER_REGISTRY}/companyms:${IMAGE_TAG}'
41                      ...
42                  }
43              }
44          }
45      }
46
47
48      stage('Build Job MS') {
49          steps {
50              dir('jobms') {
51                  timeout(time: 10, unit: 'MINUTES') {
52                      script {
53                          sh './mvnw -N io.takari:maven:wrapper || true'
54                          sh 'chmod +x ./mvnw'
55
56                          sh ...
57                          './mvnw spring-boot:build-image -DskipTests \
58                              -Dspring-boot.build-image.imageName=${DOCKER_REGISTRY}/jobms:${IMAGE_TAG}'
59                          ...
60                      }
61                  }
62              }
63          }
64      }
65
66
67      stage('Build Review MS') {
68          steps {
69              dir('reviewms') {
70                  script {
71                      sh './mvnw -N io.takari:maven:wrapper || true'
72                      sh 'chmod +x ./mvnw'
73
74                      sh ...
75                      './mvnw spring-boot:build-image -DskipTests \
76                          -Dspring-boot.build-image.imageName=${DOCKER_REGISTRY}/reviewms:${IMAGE_TAG}'
77                      ...
78                  }
79              }
80          }
81      }
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2218

```

Các bước thực hiện gồm:

- Cài đặt Maven Wrapper nếu nó chưa có sẵn, đảm bảo môi trường có thể chạy Maven mà không cần cài đặt Maven toàn cầu.
- Cấp quyền thực thi cho file Maven Wrapper
- Xây dựng Docker image cho ứng dụng Spring Boot bằng Maven

```
45      stage('SonarQube Analys for Company MS') {
46          steps {
47              dir('companym') {
48                  script {
49                      withSonarQubeEnv('sonar-server') {
50                          sh '${SCANNER_HOME}/bin/sonar-scanner --version'
51                          sh """
52                              ${SCANNER_HOME}/bin/sonar-scanner \
53                                  -Dsonar.java.binaries=target/classes \
54                                  """
55                      }
56                  }
57              }
58          }
59      }
60
61
62
63
64      stage('SonarQube Analys for Job MS') {
65          steps {
66              dir('jobms') {
67                  script {
68                      withSonarQubeEnv('sonar-server') {
69                          sh '${SCANNER_HOME}/bin/sonar-scanner --version'
70                          sh """
71                              ${SCANNER_HOME}/bin/sonar-scanner \
72                                  -Dsonar.java.binaries=target/classes \
73                                  """
74                      }
75                  }
76              }
77          }
78      }
79
80
81
82
83      stage('SonarQube Analys for Review MS') {
84          steps {
85              dir('reviewms') {
86                  script {
87                      withSonarQubeEnv('sonar-server') {
88                          sh '${SCANNER_HOME}/bin/sonar-scanner --version'
89                          sh """
90                              ${SCANNER_HOME}/bin/sonar-scanner \
91                                  -Dsonar.java.binaries=target/classes \
92                                  """
93                      }
94                  }
95              }
96          }
97      }
```

Quét lỗi hỏng các Microservices bằng Sonar Scanner

Các bước thực hiện:

- Cấu hình môi trường SonarQube với withSonarQubeEnv('sonar-server') để kết nối với server SonarQube. Sonar-server được cấu hình ở 3.5.3

- Kiểm tra phiên bản SonarQube Scanner.
- Chạy SonarQube Scanner để phân tích mã nguồn của ứng dụng companyms, jobms, reviewwms và gửi kết quả phân tích lên SonarQube server.

```

61      stage('TRIVY FS SCAN') {
62          steps {
63              echo 'Scanning Files System...'
64              sh "trivy fs ."
65          }
66      }

```

Scan File System với Trivy cho toàn bộ code

Quét hệ thống tệp (file system) của ứng dụng hoặc dự án để phát hiện các lỗ hổng bảo mật

Lệnh trivy fs . sẽ kiểm tra các tệp trong thư mục hiện tại và tìm các lỗ hổng bảo mật, phần mềm bị lỗi thời, hoặc các cấu hình không an toàn.

```

68      stage('Push MS to Docker') {
69          steps {
70              script {
71                  withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {
72                      echo 'Pushing to Docker Hub...'
73                      sh "docker push ${DOCKER_REGISTRY}/companyms:${IMAGE_TAG}"
74                  }
75              }
76          }
77      }
78
79
80      stage('Push MS to Docker') {
81          steps {
82              script {
83                  withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {
84                      echo 'Pushing to Docker Hub...'
85                      sh "docker push ${DOCKER_REGISTRY}/jobms:${IMAGE_TAG}"
86                  }
87              }
88          }
89      }
90
91      stage('Push MS to Docker') {
92          steps {
93              script {
94                  withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {
95                      echo 'Pushing to Docker Hub...'
96                      sh "docker push ${DOCKER_REGISTRY}/reviewwms:${IMAGE_TAG}"
97                  }
98              }
99          }
100     }

```

Push các Image đã build cho từng microservices lên Docker Hub

Các bước thực hiện gồm:

- **credentialsId: 'docker'**: Sử dụng thông tin đăng nhập Docker đã được cấu hình trong Jenkins ở 3.5.1 để xác thực
- **toolName: 'docker'**: Chỉ định tool Docker đã được cài ở 3.5.4.
- Push image của các microservices lên Docker Hub dựa trên các biến môi trường đã khai báo trong pipeline.

```
79 |         stage('TRIVY DOCKER IMAGE SCAN') {
80 |             steps {
81 |                 echo 'Scanning docker image...'
82 |                 sh "trivy image ${DOCKER_REGISTRY}/companyms:${IMAGE_TAG}"
83 |             }
84 |         }
85 |
86 |     }
87 |
88 |     stage('TRIVY DOCKER IMAGE SCAN') {
89 |         steps {
90 |             echo 'Scanning docker image...'
91 |             sh "trivy image ${DOCKER_REGISTRY}/jobms:${IMAGE_TAG}"
92 |         }
93 |     }
94 |
95 |     stage('TRIVY DOCKER IMAGE SCAN') {
96 |         steps {
97 |             echo 'Scanning docker image...'
98 |             sh "trivy image ${DOCKER_REGISTRY}/reviewms:${IMAGE_TAG}"
99 |         }
100 |     }
101 }
```

Scan các Docker image của các microservices với Trivy

Quét bảo mật Docker image bằng Trivy: Lệnh trivy image sẽ kiểm tra Docker image của ứng dụng companyms, jobms, reviewms và phát hiện các lỗ hổng bảo mật có thể tồn tại trong các thư viện hoặc phần mềm được sử dụng trong image đó.

```

88 v      post {
89 v          always {
90 |              echo 'Cleaning up local Docker images...'
91 |              sh "docker rmi ${DOCKER_REGISTRY}/companyms:${IMAGE_TAG} || true"
92 }
93 v          success {
94 |              echo 'Pipeline completed successfully!'
95 }
96 v          failure {
97 |              echo 'Pipeline failed.'
98 }
99 }
100 }

89 post {
90     always {
91 |         echo 'Cleaning up local Docker images...'
92 |         sh "docker rmi ${DOCKER_REGISTRY}/jobms:${IMAGE_TAG} || true"
93 }
94     success {
95 |         echo 'Pipeline completed successfully!'
96 }
97     failure {
98 |         echo 'Pipeline failed.'
99 }
100 }

86 post {
87     always {
88 |         echo 'Cleaning up local Docker images...'
89 |         sh "docker rmi ${DOCKER_REGISTRY}/reviewms:${IMAGE_TAG} || true"
90 }
91     success {
92 |         echo 'Pipeline completed successfully!'
93 }
94     failure {
95 |         echo 'Pipeline failed.'
96 }
97 }
98 }

```

Xử lý sau khi hoàn thành pipeline cho các microservices

Sau khi pipeline hoàn tất, Docker image được tạo ra sẽ bị xóa khỏi máy cục bộ để tránh việc chiếm dụng tài nguyên lưu trữ không cần thiết.

Xuất ra thông tin về kết quả của giúp người dùng dễ dàng theo dõi và xử lý.

3.6.2 Tạo JenkinsFile để Deploy lên K3S

```
4     environment {
5         KUBE_SERVER = 'https://52.20.241.169:6443'
6         KUBE_NAMESPACE = 'jenkins'
7         KUBE_CONTEXT = 'default'
8         KUBE_CREDENTIALS = 'k3s'
9     }
```

Khai báo environment

Các biến môi trường được định nghĩa:

1. **KUBE_SERVER:**

- o Địa chỉ URL của Kubernetes API server: 'https://52.20.241.169:6443'.
- o Địa chỉ IP lấy từ IP Public của JenkinsK3sEC2 và port mặc định của K3S Cluster là 6443

2. **KUBE_NAMESPACE:**

- o Khai báo namespace là jenkins trong Kubernetes để triển khai hoặc tương tác với các tài nguyên Kubernetes. Namespace giúp tách biệt các tài nguyên trong cùng một cluster.

3. **KUBE_CONTEXT:**

- o Tên context của Kubernetes được sử dụng để xác định môi trường làm việc là 'default'.
- o Context này chứa các thông tin như cluster, namespace, và user để kết nối với Kubernetes.

4. **KUBE_CREDENTIALS:**

- o Cấu hình thông tin xác thực (credentials) dùng để kết nối với Kubernetes là 'k3s' đã cấu hình ở 3.5.1

```

11   stages [
12     stage('Clean workspace') {
13       steps{
14         |   cleanWs()
15       }
16     }
17
18     stage('Checkout SCM') {
19       steps {
20         |   checkout scm
21       }
22     }

```

Clean workspace, Checkout SCM

Clean workspace: Đảm bảo không có dữ liệu cũ hoặc không cần thiết làm ảnh hưởng đến các bước xây dựng tiếp theo.

Checkout SCM: Đảm bảo rằng pipeline làm việc với mã nguồn mới nhất từ hệ thống SCM, tránh sự phụ thuộc vào mã nguồn đã lưu trữ trong workspace từ các lần chạy trước.

```

24   stage('CHECK VERSION') {
25     steps {
26       withKubeConfig([
27         credentialsId: KUBE_CREDENTIALS,
28         serverUrl: KUBE_SERVER,
29         namespace: KUBE_NAMESPACE,
30         contextName: KUBE_CONTEXT,
31       ]) {
32         echo 'Checking version, namespace, context...'
33         sh 'k3s kubectl version'
34         sh 'k3s kubectl get ns'
35         sh 'k3s kubectl config get-contexts'
36       }
37     }
38   }

```

Kiểm tra version và thông tin trong K3s

Các bước thực hiện:

1. Kết nối đến Kubernetes cluster với cấu hình đã khai báo trong environment gồm credentialsId, serverUrl, namespace và contextName.
2. Kiểm tra version của kubectl
3. Liệt kê namespaces
4. Xác nhận context hiện tại để đảm bảo mọi thứ đã được thiết lập đúng đắn.

```

40    stage('DEPLOY DATABASE TO K3S') {
41        steps {
42            withKubeConfig([
43                credentialsId: KUBE_CREDENTIALS,
44                serverUrl: KUBE_SERVER,
45                namespace: KUBE_NAMESPACE,
46                contextName: KUBE_CONTEXT,
47            ]) {
48                sh 'k3s kubectl apply -f k8s/postgresql-deployment.yaml'
49            }
50        }
51    }

```

Deploy Postgresql Database

Triển khai PostgreSQL lên Kubernetes: Bằng cách áp dụng tệp YAML cấu hình, Jenkins sẽ tạo các tài nguyên cần thiết trong Kubernetes như Pod, Service, PersistentVolumeClaim, ConfigMap, StatefulSet để triển khai cơ sở dữ liệu PostgreSQL.

```

53    stage('CREATE DATABASE FOR MICROSERVICES') {
54        steps {
55            withKubeConfig([
56                credentialsId: KUBE_CREDENTIALS,
57                serverUrl: KUBE_SERVER,
58                namespace: KUBE_NAMESPACE,
59                contextName: KUBE_CONTEXT,
60            ]) {
61                echo 'Waiting postgres is running...'
62                sh 'k3s kubectl wait --for=condition=ready pod/postgres-0 -n jenkins --timeout=300s'
63
64                echo 'Creating database...'
65                sh '''
66                    k3s kubectl exec postgres-0 -n jenkins -- psql -U embarkx -c "SELECT 1 FROM pg_database WHERE datname = 'job';" | grep -v "no"
67                    k3s kubectl exec postgres-0 -n jenkins -- psql -U embarkx -c "SELECT 1 FROM pg_database WHERE datname = 'review';" | grep -v "no"
68                    k3s kubectl exec postgres-0 -n jenkins -- psql -U embarkx -c "SELECT 1 FROM pg_database WHERE datname = 'company';" | grep -v "no"
69
70            ...
71        }
72    }

```

Tạo Database

Các bước thực hiện:

1. Kết nối đến Kubernetes cluster với cấu hình đã khai báo trong environment gồm credentialsId, serverUrl, namespace và contextName.
2. Sử dụng lệnh kubectl exec để thực hiện các câu lệnh SQL trực tiếp trong PostgreSQL container sau đó tạo các cơ sở dữ liệu job, review, và company nếu chúng chưa tồn tại trong PostgreSQL.

```

74      stage('DEPLOY APPLICATIONS TO KUBERNETES') {
75          steps {
76              withKubeConfig([
77                  credentialsId: KUBE_CREDENTIALS,
78                  serverUrl: KUBE_SERVER,
79                  namespace: KUBE_NAMESPACE,
80                  contextName: KUBE_CONTEXT,
81              ]) {
82                  echo 'Deploy to k3s'
83                  sh 'k3s kubectl apply -f k8s/companyms-deployment.yaml'
84                  sh 'k3s kubectl apply -f k8s/jobms-deployment.yaml'
85                  sh 'k3s kubectl apply -f k8s/reviewms-deployment.yaml'
86              }
87          }
88      }
89  }

```

Deploy các microservices

Các bước thực hiện:

1. Kết nối đến Kubernetes cluster với cấu hình đã khai báo trong environment gồm credentialsId, serverUrl, namespace và contextName.
2. Áp dụng tệp YAML cho mỗi microservice (companyms, jobms, reviewms) để triển khai chúng lên cluster.

```

91    post {
92        success {
93            echo 'Pipeline completed successfully!'
94        }
95        failure {
96            echo 'Pipeline failed.'
97        }
98    }
99  }
100

```

Xử lý khi hoàn tất pipeline

Mục đích:

- Cung cấp thông báo rõ ràng về kết quả của pipeline sau khi quá trình thực thi hoàn tất.
- Giúp người dùng hoặc nhóm phát triển dễ dàng theo dõi tình trạng pipeline (thành công hay thất bại).

4. CẤU HÌNH KUBERNETES

4.1 postgresql-deployment.yaml

```
1  apiVersion: v1          You, 3 weeks ago
2  kind: ConfigMap
3  metadata:
4    name: postgres-config
5  data:
6    POSTGRES_DB: embarkx
7    POSTGRES_USER: embarkx
8    POSTGRES_PASSWORD: embarkx
9
```

Cấu hình ConfigMap cho Postgresql

Lưu trữ các biến môi trường cần thiết để cấu hình PostgreSQL, bao gồm tên cơ sở dữ liệu, tên người dùng và mật khẩu.

```
12 apiVersion: v1
13 kind: Service
14 metadata:
15   name: postgres
16 spec:
17 selector:
18   app: postgres
19 ports:
20   - port: 5432
21     targetPort: 5432
22   type: ClusterIP
```

Cấu hình Service cho Postgresql

Cấu hình Port là 5432, type là ClusterIP, cho phép các ứng dụng hoặc các Pod khác trong cluster Kubernetes dễ dàng kết nối đến PostgreSQL mà không cần phải biết địa chỉ IP cụ thể của các Pod PostgreSQL.

```

26  apiVersion: apps/v1
27  kind: StatefulSet
28  metadata:
29    name: postgres
30    labels:
31      app: postgres
32  spec:
33    serviceName: postgres
34    replicas: 1
35    selector:
36      matchLabels:
37        app: postgres
38    template:
39      metadata:
40        labels:
41          app: postgres
42    spec:     You, 3 weeks ago • Config deploy micro
43      containers:
44        - name: postgres
45          image: postgres
46          imagePullPolicy: IfNotPresent
47          envFrom:
48            - configMapRef:
49              name: postgres-config
50      resources:
51        requests:
52          cpu: 100m
53          memory: 256Mi
54        limits:
55          cpu: 500m
56          memory: 512Mi
57      volumeMounts:
58        - name: postgres-data
59          mountPath: "/var/lib/postgresql/data"
60    volumeClaimTemplates:
61      - metadata:
62          name: postgres-data
63        spec:
64          accessModes: [ "ReadWriteOnce" ]
65          resources:
66            requests:
67              storage: 5Gi

```

Cấu hình StatefulSet cho Postgres

Mục đích và Quy trình:

- StatefulSet PostgreSQL:** Triển khai một StatefulSet để chạy một Pod duy nhất (1 replica) PostgreSQL với tên postgres.
- Persistent Storage:** Dữ liệu của PostgreSQL sẽ được lưu trữ trong một Persistent Volume (PV) được quản lý thông qua PersistentVolumeClaim

(PVC). Volume này sẽ được gắn vào container tại /var/lib/postgresql/data, nơi dữ liệu của PostgreSQL được lưu trữ.

3. **ConfigMap**: Sử dụng ConfigMap postgres-config để cung cấp các thông tin cấu hình như POSTGRES_DB, POSTGRES_USER, và POSTGRES_PASSWORD cho PostgreSQL.
4. **Kích thước tài nguyên**: Container PostgreSQL sẽ có yêu cầu tài nguyên cụ thể (CPU và bộ nhớ), đảm bảo rằng tài nguyên hệ thống sẽ được phân bổ hợp lý.
5. **Quản lý trạng thái**: StatefulSet sẽ duy trì tên và lưu trữ của các Pod, giúp ứng dụng duy trì trạng thái và không bị mất dữ liệu khi các Pod được khởi động lại hoặc thay đổi.

```
71  apiVersion: v1
72  kind: PersistentVolumeClaim
73  metadata:
74    name: postgres-pc-volume-claim
75    labels:
76      app: postgres
77  spec:
78    storageClassName: manual
79    accessModes:
80      - ReadWriteOnce
81    resources:
82      requests:
83        storage: 5Gi
```

Cấu hình PVC cho Postgres

Mục đích:

- **PersistentVolumeClaim (PVC)** này yêu cầu một PersistentVolume có dung lượng 5Gi, có thể được sử dụng bởi PostgreSQL để lưu trữ dữ liệu bền vững.
- PVC sẽ chỉ được gắn vào một Pod tại một thời điểm và sẽ duy trì trạng thái của dữ liệu ngay cả khi Pod bị tái khởi động.

4.2 companyms-deployment.yaml

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: company
5    labels:
6      app: company
7  spec:
8    replicas: 1
9    template:
10      metadata:
11        name: company
12        labels:
13          app: company
14      spec:
15        containers:
16          - name: company
17            image: lambaoduy1310/companyms:v4.4
18            imagePullPolicy: Always
19            ports:
20              - containerPort: 8081
21            env:
22              - name: SPRING_PROFILES_ACTIVE
23                value: k8s
24            resources:
25              requests:
26                memory: "512Mi"
27                cpu: "250m"
28              limits:
29                memory: "1Gi"
30                cpu: "500m"
31            restartPolicy: Always
32          selector:
33            matchLabels:
34              app: company
```

Cấu hình Deployment Company Microservices

Deployment này tạo ra một Pod duy nhất với một container chạy ứng dụng companyms, sử dụng Docker image lambaoduy1310/companyms:v4.4. Docker images này được build từ pipeline và được push lên docker hub

Container port là 8081

Spring profile k8s sẽ sử dụng file application-k8s.properties trong ứng dụng để cấu hình.

Với replicaset là 1, deployment này sẽ tự động duy trì một Pod với container đã định nghĩa, khởi động lại nếu container gặp sự cố.

```
38  apiVersion: v1
39  kind: Service
40  metadata:
41    name: company
42  spec:
43    selector:
44      app: company
45    ports:
46      - port: 80
47        targetPort: 8081
48    type: LoadBalancer
```

Cấu hình Service cho company microservices

Tóm tắt về chức năng của Service này

- **Service** này cung cấp một điểm truy cập duy nhất cho ứng dụng company.
- **Selector** giúp Service này tiếp cận các Pods có nhãn app: company, tức là các Pods đã được triển khai qua Deployment của ứng dụng company.
- **Port mapping:** Service sẽ tiếp nhận yêu cầu trên cổng 80 và chuyển tiếp chúng đến cổng 8081 trong các Pods của ứng dụng company.
- **Type LoadBalancer:** Với cấu hình này, Kubernetes sẽ tạo ra một Load Balancer, cho phép truy cập từ bên ngoài vào ứng dụng thông qua một địa chỉ IP công cộng.

4.3 jobms-deployment.yaml

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: job
5   labels:
6     app: job
7 spec:
8   replicas: 1
9   template:
10    metadata:
11      name: job
12      labels:
13        app: job
14    spec:
15      containers:
16        - name: job
17          image: lambaoduy1310/jobms:v4.1
18          imagePullPolicy: Always
19          ports:
20            - containerPort: 8082
21          env:
22            - name: SPRING_PROFILES_ACTIVE
23              value: k8s
24          resources:
25            requests:
26              memory: "512Mi"
27              cpu: "250m"
28            limits:
29              memory: "1Gi"
30              cpu: "500m"
31          restartPolicy: Always
32      selector:
33        matchLabels:
34          app: job
```

Cấu hình Deployment cho job microservices.

```
38 v apiVersion: v1
39   kind: Service
40 v metadata:
41   | name: job
42 v spec:
43 v   selector:
44   | app: job
45 v   ports:
46 v     - port: 80
47       targetPort: 8082
48   type: LoadBalancer
49
```

Cấu hình Service cho company microservices

Cấu hình Service và Deployment cho review microservices tương tự với company microservices chỉ thay đổi metadata, selector và port để phù hợp cho từng microservices

4.4 reviewms-deployment.yaml

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: review
5    labels:
6      app: review
7  spec:
8    replicas: 1
9    template:
10      metadata:
11        name: review
12        labels:
13          app: review
14      spec:
15        containers:
16          - name: review
17            image: lambdaoudy1310/reviewms:v4.1
18            imagePullPolicy: Always
19            ports:
20              - containerPort: 8083
21            env:
22              - name: SPRING_PROFILES_ACTIVE
23                value: k8s
24            resources:
25              requests:
26                memory: "512Mi"
27                cpu: "250m"
28              limits:
29                memory: "1Gi"
30                cpu: "500m"
31            restartPolicy: Always
32        selector:
33          matchLabels:
34            app: review
```

Cáu hình Deployment cho review microservices

```
38  apiVersion: v1
39  kind: Service
40  metadata:
41    name: review
42  spec:
43    selector:
44      app: review
45    ports:
46      - protocol: TCP
47        port: 80
48        targetPort: 8083
49    type: LoadBalancer
50
```

Cáu hình Service cho review microservices

Cáu hình Service và Deployment cho review microservices tương tự với company microservices chỉ thay đổi metadata, selector và port để phù hợp cho từng microservices

5. CẤU HÌNH PROMETHEUS & GRAFANA

5.1 Prometheus

```
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"
    static_configs:
      - targets: ["localhost:9090"]

  - job_name: 'prometheus-grafana-node-exporter'
    static_configs:
      - targets: ['34.230.202.25:9100']

  - job_name: 'jenkins-k3s-node-exporter'
    static_configs:
      - targets: ['52.20.241.169:9100']

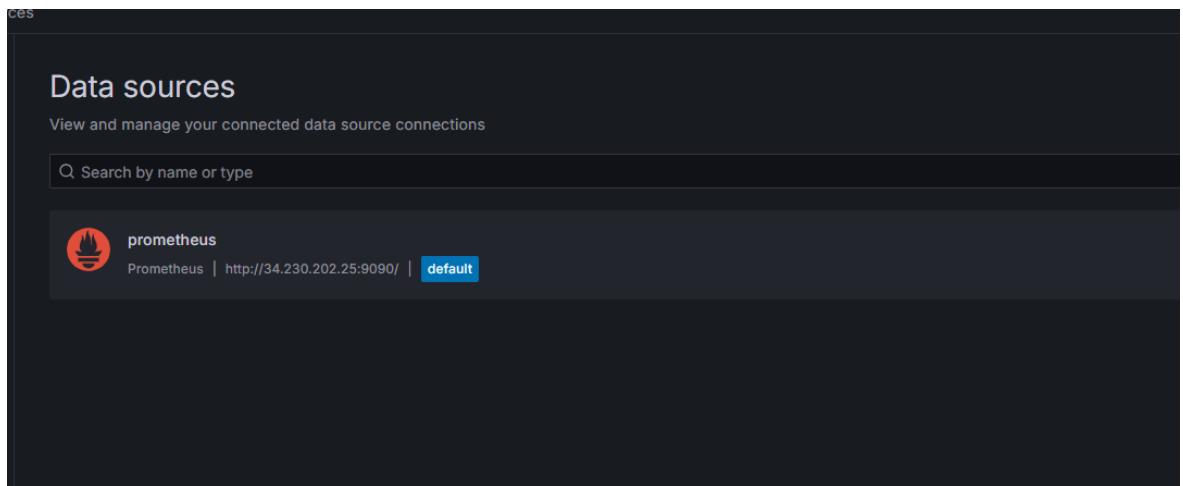
  - job_name: 'company-service'
    metrics_path: '/actuator/prometheus'
    static_configs:
      - targets: ['52.20.241.169:30938']

  - job_name: 'job-service'
    metrics_path: '/actuator/prometheus'
    static_configs:
      - targets: ['52.20.241.169:31432']

  - job_name: 'review-service'
    metrics_path: '/actuator/prometheus'
    static_configs:
      - targets: ['52.20.241.169:30321']
root@ip-192-168-1-101:/etc/prometheus# |
```

Cấu hình các job cho prometheus

5.2 Grafana



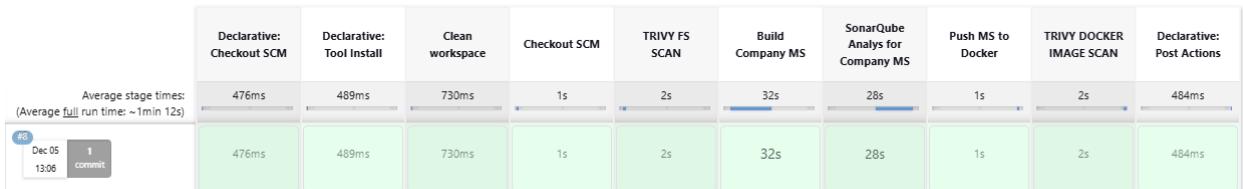
Cấu hình Datasource cho Grafana

6. KẾT QUẢ THỰC HIỆN

6.1 Jenkins

companyms

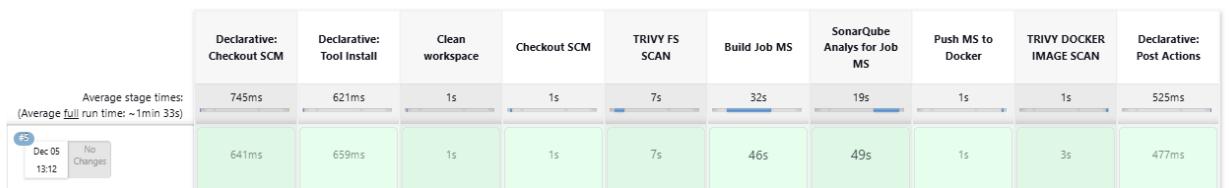
Stage View



Jenkins Pipeline Company MicroServices

jobms

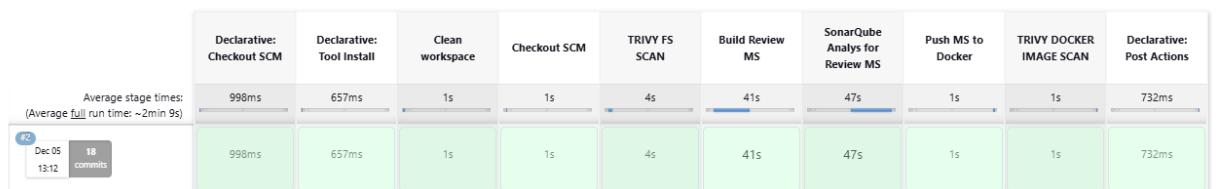
Stage View



Jenkins Pipeline Job MicroServices

reviewms

Stage View



Jenkins Pipeline Review MicroServices

Stage View

Stage	Clean workspace	Checkout SCM	CHECK VERSION	DEPLOY DATABASE TO K3S	CREATE DATABASE FOR MICROSERVICES	DEPLOY APPLICATIONS TO KUBERNETES	Declarative: Post Actions
Average stage times: (Average full run time: ~10s)	140ms	493ms	1s	1s	9s	1s	101ms
#2 Nov 30 08:56 No Changes	362ms	149ms	513ms	1s	1s	2s	2s
#1 Nov 30 08:55 No Changes	426ms	131ms	473ms	1s	990ms	17s failed	117ms failed

Permalinks

- Last build (#2), 1 day 10 hr ago
- Last stable build (#2), 1 day 10 hr ago
- Last successful build (#2), 1 day 10 hr ago
- Last failed build (#1), 1 day 10 hr ago
- Last unsuccessful build (#1), 1 day 10 hr ago
- Last completed build (#2), 1 day 10 hr ago

Jenkins Pipeline Deploy Microservices to K3S Cluster

6.1.1 Kết quả Run Pipeline

Name	Last Success	Last Failure
companyms	1 day 10 hr #4	1 day 10 hr #2
deploy to kubernetes	1 day 10 hr #2	1 day 10 hr #1
jobms	1 day 11 hr #1	N/A
reviewms	1 day 11 hr #1	N/A

Kết quả run Pipeline CI/CD

6.1.2 Pipeline Log

Pipeline cho các microservices:

```

69:8080/job/companyms4/consoleFull

[INFO] [creator] Launch Helper: contributing to layer
[INFO] [creator] Creating /layers/paketo-buildpacks_spring-boot/helper/exec.d/spring-cloud-bindings
[INFO] [creator] Spring Cloud Bindings 1.13.0: contributing to layer
[INFO] [creator] Downloading from https://repo1.maven.org/maven2/org/springframework/cloud/spring-cloud-bindings/1.13.0/spring-cloud-bindings-1.13.0.jar
[INFO] [creator] Verifying checksum
[INFO] [creator] Copying to /layers/paketo-buildpacks_spring-boot/spring-cloud-bindings
[INFO] [creator] Web Application Type: Contributing to layer
[INFO] [creator] Servlet web application detected
[INFO] [creator] Writing env.launchBPL_JVM_THREAD_COUNT.default
[INFO] [creator] 4 application slices
[INFO] [creator] Image labels:
[INFO] [creator] org.opencontainers.image.title
[INFO] [creator] org.opencontainers.image.version
[INFO] [creator] org.springframework.boot.version
[INFO] [creator] ===> EXPORTING
[INFO] [creator] Adding layer 'paketo-buildpacks/ca-certificates:helper'
[INFO] [creator] Adding layer 'paketo-buildpacks/bellsoft-liberica:helper'
[INFO] [creator] Adding layer 'paketo-buildpacks/bellsoft-liberica:java-security-properties'
[INFO] [creator] Adding layer 'paketo-buildpacks/bellsoft-liberica:jre'
[INFO] [creator] Adding layer 'paketo-buildpacks/executable-jar:classpath'
[INFO] [creator] Adding layer 'paketo-buildpacks/spring-boot:helper'
[INFO] [creator] Adding layer 'paketo-buildpacks/spring-boot:spring-cloud-bindings'
[INFO] [creator] Adding layer 'paketo-buildpacks/spring-boot:web-application-type'
[INFO] [creator] Adding layer 'buildpacksio/lifecycle:launch.sbm'
[INFO] [creator] Adding S/S app layer(s)
[INFO] [creator] Adding layer 'buildpacksio/lifecycle:launcher'
[INFO] [creator] Adding layer 'buildpacksio/lifecycle:config'
[INFO] [creator] Adding layer 'buildpacksio/lifecycle:process-types'
[INFO] [creator] Adding label 'io.buildpacks.lifecycle.metadata'
[INFO] [creator] Adding label 'buildpacks.build.metadata'
[INFO] [creator] Adding label 'buildpacks.project.metadata'
[INFO] [creator] Adding label 'org.opencontainers.image.title'
[INFO] [creator] Adding label 'org.opencontainers.image.version'
[INFO] [creator] Adding label 'org.springframework.boot.version'
[INFO] [creator] Setting default process type 'web'
[INFO] [creator] Saving docker.io/lmbaoduy1310/companyms:v4.4...
[INFO] [creator] *** Images (47f70ded1f2e):
[INFO] [creator] docker.io/lmbaoduy1310/companyms:v4.4
[INFO] [creator] Adding cache layer 'paketo-buildpacks/syft:syft'
[INFO] [creator] Adding cache layer 'buildpacksio/lifecycle:cache.sbm'
[INFO]
[INFO] Successfully built image 'docker.io/lmbaoduy1310/companyms:v4.4'
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] total time: 23.984 s
[INFO] Finished at: 2024-11-30T01:45:14Z
[INFO] -----
[Pipeline] )
[Pipeline] // script
[Pipeline] )

```

Build Microservices

41.169:8080/job/companyms/4/consoleFull

```
01:45:37.962 INFO Sensor IaC Kubernetes Sensor [iac] (done) | time=239ms
01:45:37.963 INFO Sensor Java Config Sensor [iac]
01:45:37.981 INFO 2 source files to be analyzed
01:45:38.195 INFO 2/2 source files have been analyzed
01:45:38.195 INFO Sensor Java Config Sensor [iac] (done) | time=241ms
01:45:38.195 INFO Sensor JavaScript inside YAML analysis [javascript]
01:45:38.221 INFO No input files found for analysis
01:45:38.221 INFO Hit the cache for 0 out of 0
01:45:38.221 INFO Miss the cache for 0 out of 0
01:45:38.222 INFO Sensor JavaScript inside YAML analysis [javascript] (done) | time=24ms
01:45:38.222 INFO Sensor CSS Rules [javascript]
01:45:38.222 INFO NO CSS, PHP, HTML or VueJS files are found in the project. CSS analysis is skipped.
01:45:38.224 INFO Sensor CSS Rules [javascript] (done) | time=0ms
01:45:38.225 INFO Sensor IaC Docker Sensor [iac]
01:45:38.377 INFO 0 source files to be analyzed
01:45:38.378 INFO 0/0 source files have been analyzed
01:45:38.379 INFO Sensor IaC Docker Sensor [iac] (done) | time=156ms
01:45:38.379 INFO Sensor TextAndSecretsSensor [text]
01:45:38.379 INFO Available processors: 2
01:45:38.382 INFO Using 2 threads for analysis.
01:45:39.298 INFO The property "sonar.tests" is not set. To improve the analysis accuracy, we categorize a file as a test file if any of the following is true:
 * The filename starts with "test"
 * The filename contains "test." or "tests."
 * Any directory in the file path is named: "doc", "docs", "test" or "tests"
 * Any directory in the file path has a name ending in "test" or "tests"

01:45:39.326 INFO Using git CLI to retrieve untracked files
01:45:39.443 INFO Analyzing language associated files and files included via "sonar.text.inclusions" that are tracked by git
01:45:39.471 INFO 15 source files to be analyzed
01:45:39.721 INFO 15/15 source files have been analyzed
01:45:39.724 INFO Sensor TextAndSecretsSensor [text] (done) | time=1347ms
01:45:39.729 INFO -----
01:45:40.032 INFO Sensor Zero Coverage Sensor
01:45:40.044 INFO Sensor Zero Coverage Sensor (done) | time=13ms
01:45:40.044 INFO Sensor Java CPD Block Indexer
01:45:40.092 INFO Sensor Java CPD Block Indexer (done) | time=45ms
01:45:40.101 INFO CPD Executor 5 files had no CPD blocks
01:45:40.101 INFO CPD Executor Calculating CPD for 5 files
01:45:40.123 INFO CPD Executor CPD calculation finished (done) | time=16ms
01:45:40.132 INFO SCM revision ID 'd98bed8902bb697459a187cd59efbf104d3deddfdc'
01:45:40.304 INFO Analysis report generated in 160ms, dir size=250.8 kB
01:45:40.381 INFO Analysis report compressed in 76ms, zip size=46.8 kB
01:45:40.415 INFO Analysis report uploaded in 31ms
01:45:40.415 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://52.20.241.169:9000/dashboard?id=Company-MS
01:45:40.415 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
01:45:40.416 INFO More about the report processing at http://52.20.241.169:9000/api/ce/task?id=26e05ff2-4ead-44b5-acdf-084c021b6e2d
01:45:40.439 INFO Analysis total time: 19.317 s
01:45:40.442 INFO SonarScanner Engine completed successfully
01:45:40.792 INFO EXECUTION SUCCESS
01:45:40.794 INFO Total time: 23.586s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
```

Sonar Scanner

```

Scanning Files System...
[Pipeline] sh
+ trivy fs .
2024-11-30T01:45:42Z INFO  [vuln] Vulnerability scanning is enabled
2024-11-30T01:45:42Z INFO  [secret] secret scanning is enabled
2024-11-30T01:45:42Z INFO  [secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2024-11-30T01:45:42Z INFO  [secret] Please see also https://aquasecurity.github.io/trivy/v0.57/docs/scanner/secret#recommendation for faster secret detection
2024-11-30T01:45:46Z WARN  [pom] Dependency version cannot be determined. Child dependencies will not be found. details="https://aquasecurity.github.io/trivy/v0.57/docs/coverage/language/java#empty-dependency-version"
2024-11-30T01:45:55Z INFO  [pom] Number of language-specific files num=4
2024-11-30T01:45:55Z INFO  [pom] Detecting vulnerabilities...

For OSS Maintainers: VEX Notice
-----
If you're an OSS maintainer and Trivy has detected vulnerabilities in your project that you believe are not actually exploitable, consider issuing a VEX (Vulnerability Exploitability Exchange) statement. VEX allows you to communicate the actual status of vulnerabilities in your project, improving security transparency and reducing false positives for your users.
Learn more and start using VEX: https://aquasecurity.github.io/trivy/v0.57/docs/supply-chain/vex/repo#publishing-vex-documents

To disable this notice, set the TRIVY_DISABLE_VEX_NOTICE environment variable.

companym/pom.xml (pom)
=====
Total: 40 (UNKNOWN: 0, LOW: 2, MEDIUM: 19, HIGH: 18, CRITICAL: 1)



| Library                              | Vulnerability  | Severity | Status | Installed Version | Fixed Version          | Title                                                                                                                                                                                                                 |
|--------------------------------------|----------------|----------|--------|-------------------|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ch.qos.logback:logback-classic       | CVE-2023-6378  | HIGH     | fixed  | 1.4.8             | 1.3.12, 1.4.12, 1.2.13 | logback: serialization vulnerability in logback receiver<br><a href="https://avd.aquasec.com/nvd/cve-2023-6378">https://avd.aquasec.com/nvd/cve-2023-6378</a>                                                         |
| ch.qos.logback:logback-core          |                |          |        |                   |                        |                                                                                                                                                                                                                       |
| com.fasterxml.woodstox:woodstox-core | CVE-2022-40152 | MEDIUM   |        | 6.2.1             | 6.4.0, 5.4.0           | woodstox-core: woodstox to serialise XML data was vulnerable to Denial of Service...<br><a href="https://avd.aquasec.com/nvd/cve-2022-40152">https://avd.aquasec.com/nvd/cve-2022-40152</a>                           |
| com.google.guava:guava               | CVE-2018-10237 |          |        | 19.0              | 24.1.1-android         | guava: Unbounded memory allocation in AtomicDoubleArray and CompoundOrdering classes allow remote attackers...<br><a href="https://avd.aquasec.com/nvd/cve-2018-10237">https://avd.aquasec.com/nvd/cve-2018-10237</a> |
|                                      | CVE-2023-2976  |          |        |                   | 32.0.0-android         | guava: insecure temporary directory creation<br><a href="https://avd.aquasec.com/nvd/cve-2023-2976">https://avd.aquasec.com/nvd/cve-2023-2976</a>                                                                     |
|                                      | CVE-2020-8908  | LOW      |        |                   |                        | guava: local information disclosure via temporary directory created with unsafe permissions<br><a href="https://avd.aquasec.com/nvd/cve-2020-8908">https://avd.aquasec.com/nvd/cve-2020-8908</a>                      |
| com.h2database:h2                    | CVE-2022-45868 | HIGH     |        | 2.1.214           | 2.2.220                | The web-based admin console in H2 Database Engine before                                                                                                                                                              |


```

Trivy quét file pom các microservices

```

169:8080/job/companym/4/consoleFull

[Pipeline] sh
+ docker push lambdauy1310/companyms:v4.4
The push refers to repository [docker.io/lambdauy1310/companyms]
1dc94a70dbaa: Preparing
ed2c7898e456: Preparing
8812c86dc600: Preparing
5f70bf18a086: Preparing
0c3cf1627429: Preparing
5f70bf18a086: Preparing
0345504805f0: Preparing
6fee5596fd1: Preparing
c88c71fb7de: Preparing
6763b99e3792: Preparing
974316173d3: Preparing
4d96bd69ecca: Preparing
7fbcc97c30fad: Preparing
f245ab22134e: Preparing
ec0381c8f321: Preparing
ec60a3e8086a6: Preparing
52efb1a98ceb: Preparing
1eb5b983d7301: Preparing
39d381810cef: Preparing
115fc79fb3d1: Preparing
fd93afbb1e1c: Preparing
f92983442b23: Preparing
4d274d05ee12: Preparing

```

Push image của các microservices lên Docker Hub.

```

Scanning docker image...
[Pipeline] sh
+ trivy image lambdaoduy1310/companyms:v4.4
2024-11-30T01:45:59Z    INFO    [vuln] Vulnerability scanning is enabled
2024-11-30T01:45:59Z    INFO    [secret] Secret scanning is enabled
2024-11-30T01:45:59Z    INFO    [secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2024-11-30T01:45:59Z    INFO    [secret] Please see also https://aquasecurity.github.io/trivy/v0.57/docs/scanner/secret#recommendation for faster secret detection
2024-11-30T01:45:59Z    INFO    Detected OS      Family="Ubuntu" version="18.04"
2024-11-30T01:45:59Z    INFO    Detecting vulnerabilities... os_version="18.04" pkg_num=97
2024-11-30T01:45:59Z    INFO    Number of language-specific files   num=6
2024-11-30T01:45:59Z    INFO    [gobinary] Detecting vulnerabilities...
2024-11-30T01:45:59Z    INFO    [jar] Detecting vulnerabilities...
2024-11-30T01:45:59Z    WARN    Using severities from other vendors for some vulnerabilities. Read https://aquasecurity.github.io/trivy/v0.57/docs/scanner/vulnerability#severity-selection for details.
2024-11-30T01:45:59Z    WARN    This OS version is no longer supported by the distribution      family="Ubuntu" version="18.04"
2024-11-30T01:45:59Z    WARN    The vulnerability detection may be insufficient because security updates are not provided

For OSS Maintainers: VEX Notice
-----
If you're an OSS maintainer and Trivy has detected vulnerabilities in your project that you believe are not actually exploitable, consider issuing a VEX (Vulnerability Exploitability Exchange) statement. VEX allows you to communicate the actual status of vulnerabilities in your project, improving security transparency and reducing false positives for your users. Learn more and start using VEX: https://aquasecurity.github.io/trivy/v0.57/docs/supply-chain/vex/repo#publishing-vex-documents

To disable this notice, set the TRIVY_DISABLE_VEX_NOTICE environment variable.

lambdaoduy1310/companyms:v4.4 (Ubuntu 18.04)
=====
Total: 0 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 0)

2024-11-30T01:45:59Z    INFO    Table result includes only package filenames. Use '--format json' option to get the full path to the package file.

Java (jar)
=====
Total: 40 (UNKNOWN: 0, LOW: 2, MEDIUM: 19, HIGH: 18, CRITICAL: 1)



| Library                                                        | Vulnerability  | Severity | Status | Installed Version | Fixed Version          | Title                                                                                                                                                                                                                    |
|----------------------------------------------------------------|----------------|----------|--------|-------------------|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ch.qos.logback:logback-classic (logback-classic-1.4.8.jar)     | CVE-2023-6378  | HIGH     | fixed  | 1.4.8             | 1.3.12, 1.4.12, 1.2.13 | logback: serialization vulnerability in logback receiver<br><a href="https://avd.aquasec.com/nvd/cve-2023-6378">https://avd.aquasec.com/nvd/cve-2023-6378</a>                                                            |
| ch.qos.logback:logback-core (logback-core-1.4.8.jar)           |                |          |        |                   |                        |                                                                                                                                                                                                                          |
| com.fasterxml.woodstox:woodstox-core (woodstox-core-6.2.1.jar) | CVE-2022-40152 | MEDIUM   |        | 6.2.1             | 6.4.0, 5.4.0           | woodstox-core: woodstox to serialise XML data was vulnerable<br>to Denial of Service...<br><a href="https://avd.aquasec.com/nvd/cve-2022-40152">https://avd.aquasec.com/nvd/cve-2022-40152</a>                           |
| com.google.guava:guava (guava-19.0.jar)                        | CVE-2018-10237 |          | 19.0   |                   | 24.1.1-android         | guava: Unbounded memory allocation in AtomicDoubleArray and<br>CompoundOrdering classes allow remote attackers...<br><a href="https://avd.aquasec.com/nvd/cve-2018-10237">https://avd.aquasec.com/nvd/cve-2018-10237</a> |


```

Trivy Scan Docker Images

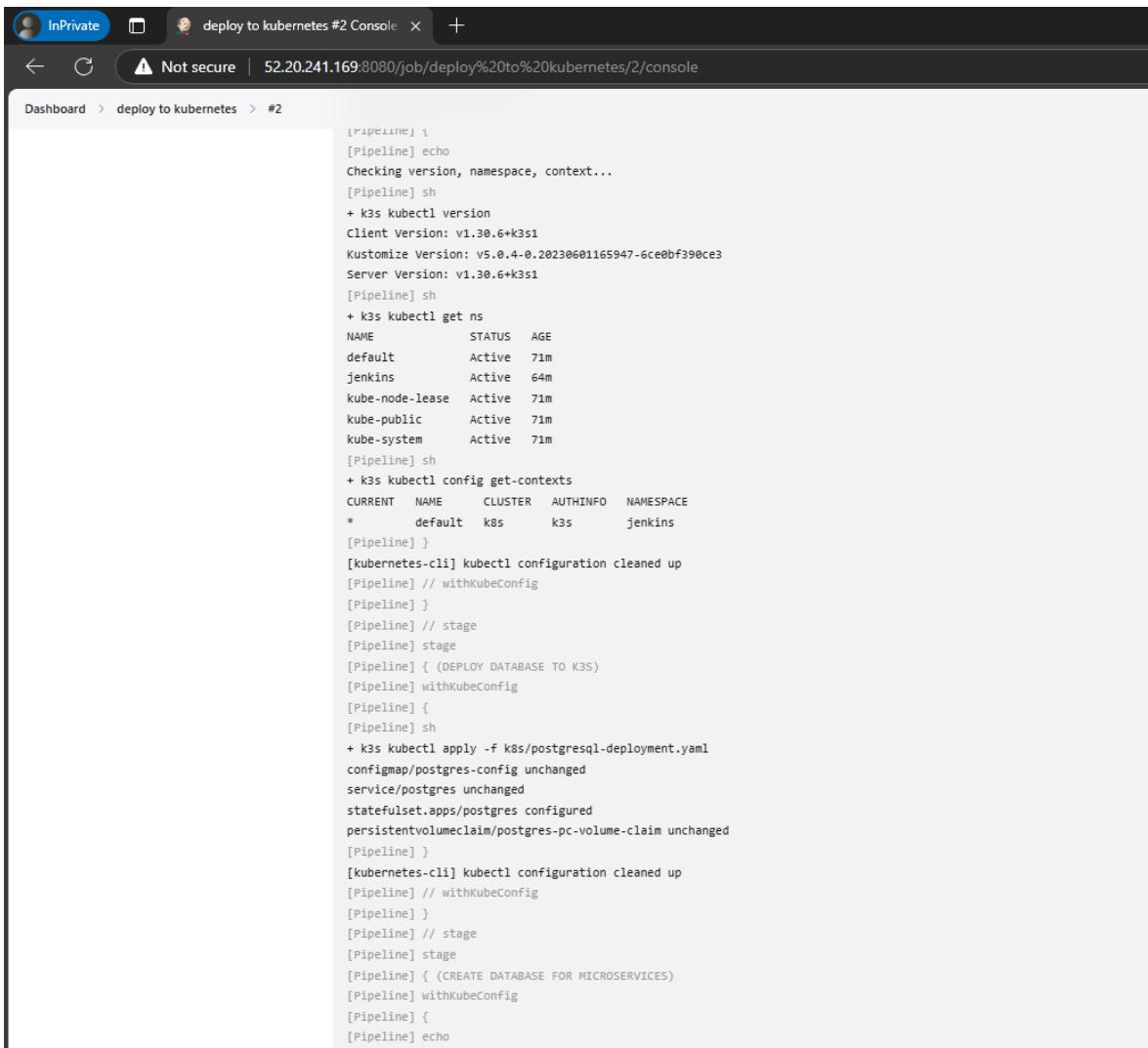
```

[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
Cleaning up local Docker images...
[Pipeline] sh
+ docker rmi lambdaoduy1310/companyms:v4.4
Untagged: lambdaoduy1310/companyms:v4.4
Untagged: lambdaoduy1310/companyms@sha256:7bcd345c1e65016f8379897b5ee690f74f7ea56a1d9dea4f6f2efc6bf04fc2b0
Deleted: sha256:a7f70de81f2e8779907f946b52fd0beb5549bedd5b57d676da0f2a29f2bdb48
Deleted: sha256:4fbde1d2f47e8c8452549d9101b492611af95b6eb9a7aae11a0f4b3a2zc0ef0
Deleted: sha256:027763b86db976840b0d848fde319f519e3f1ab49f0a2bc580a8a4ac6d11fa6d
Deleted: sha256:370d5758029d25c89d8e5284b92e0fc12ceaa00ca4a68a2cae97b31bbf3cb7
[Pipeline] echo
Pipeline completed successfully!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

Xóa images sau khi hoàn tất pipeline và thông báo kết quả

Log Pipeline Deploy lên K3s Cluster



The screenshot shows a browser window with the title "deploy to kubernetes #2 Console". The URL is "Not secure | 52.20.241.169:8080/job/deploy%20to%20kubernetes/2/console". The page content displays the log output of a Jenkins pipeline named "deploy to kubernetes" run #2. The log shows the following steps:

```
[Pipeline] echo
Checking version, namespace, context...
[Pipeline] sh
+ k3s kubectl version
Client Version: v1.30.6+k3s1
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: v1.30.6+k3s1
[Pipeline] sh
+ k3s kubectl get ns
NAME      STATUS   AGE
default   Active   71m
jenkins   Active   64m
kube-node-lease  Active   71m
kube-public  Active   71m
kube-system  Active   71m
[Pipeline] sh
+ k3s kubectl config get-contexts
CURRENT  NAME   CLUSTER   AUTHINFO   NAMESPACE
*        default  k3s       jenkins
[Pipeline]
[kubernetes-clij] kubectl configuration cleaned up
[Pipeline] // withKubeConfig
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] {
  (DEPLOY DATABASE TO K3S)
[Pipeline] withKubeConfig
[Pipeline] {
[Pipeline] sh
+ k3s kubectl apply -f k8s/postgresql-deployment.yaml
configmap/postgres-config unchanged
service/postgres unchanged
statefulset.apps/postgres configured
persistentvolumeclaim/postgres-pv-volume-claim unchanged
[Pipeline]
[kubernetes-clij] kubectl configuration cleaned up
[Pipeline] // withKubeConfig
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] {
  (CREATE DATABASE FOR MICROSERVICES)
[Pipeline] withKubeConfig
[Pipeline] {
[Pipeline] echo
```

Check version, namespace, context K3S và deploy database

```

[Pipeline] sh
+ k3s kubectl wait --for=condition=ready pod/postgres-0 -n jenkins --timeout=300s
pod/postgres-0 condition met
[Pipeline] echo
Creating database...
[Pipeline] sh
+ grep -q 1
+ k3s kubectl exec postgres-0 -n jenkins -- psql -U embarkx -c SELECT 1 FROM pg_database WHERE datname = 'job';
+ kubectl exec postgres-0 -n jenkins -- psql -U embarkx -c CREATE DATABASE job;
CREATE DATABASE
+ grep -q 1
+ k3s kubectl exec postgres-0 -n jenkins -- psql -U embarkx -c SELECT 1 FROM pg_database WHERE datname = 'review';
+ kubectl exec postgres-0 -n jenkins -- psql -U embarkx -c CREATE DATABASE review;
CREATE DATABASE
+ k3s kubectl exec postgres-0 -n jenkins -- psql -U embarkx -c SELECT 1 FROM pg_database WHERE datname = 'company';
+ grep -q 1
+ kubectl exec postgres-0 -n jenkins -- psql -U embarkx -c CREATE DATABASE company;
CREATE DATABASE
[Pipeline]
[kubernetes-cli] kubectl configuration cleaned up
[Pipeline] // withKubeConfig
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (DEPLOY APPLICATIONS TO KUBERNETES)
[Pipeline] withKubeConfig
[Pipeline] {
[Pipeline] echo
Deploy to k3s
[Pipeline] sh
+ k3s kubectl apply -f k8s/companyms-deployment.yaml
deployment.apps/company created
service/company created
[Pipeline] sh
+ k3s kubectl apply -f k8s/jobms-deployment.yaml
deployment.apps/job created
service/job created
[Pipeline] sh
+ k3s kubectl apply -f k8s/reviewms-deployment.yaml
deployment.apps/review created
service/review created
[Pipeline]
[kubernetes-cli] kubectl configuration cleaned up
[Pipeline] // withKubeConfig
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
Pipeline completed successfully!
[Pipeline]
[Pipeline] // stage

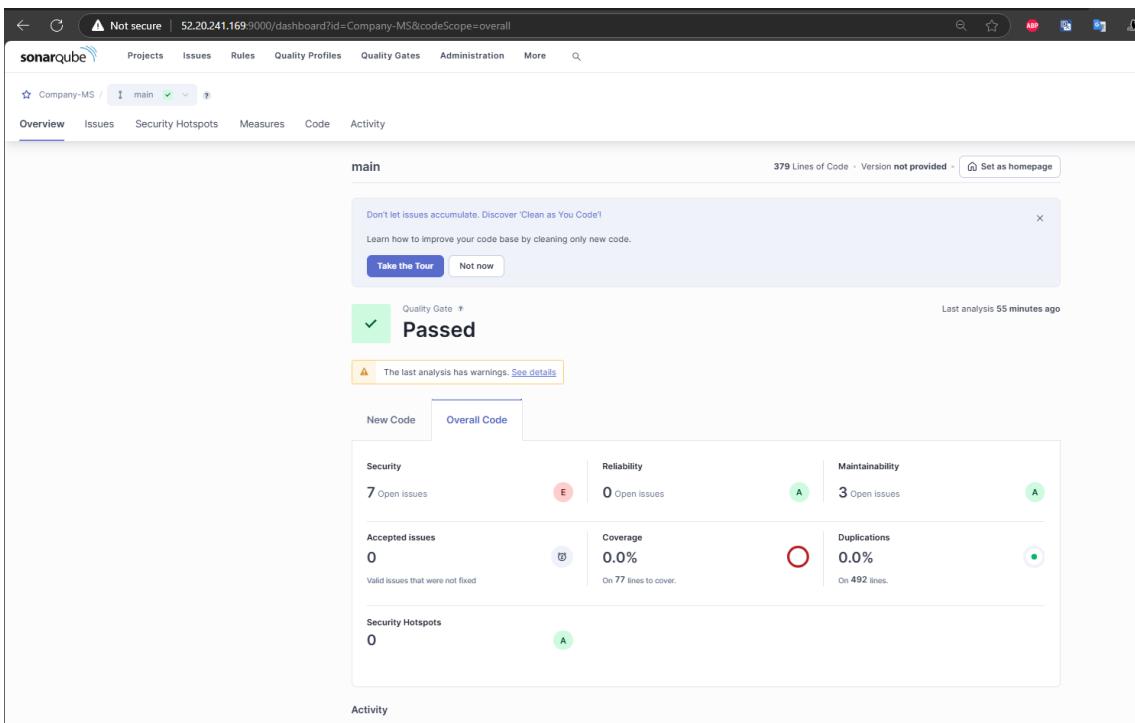
```

Tạo các Database cho Microservices và Deploy các Microservices

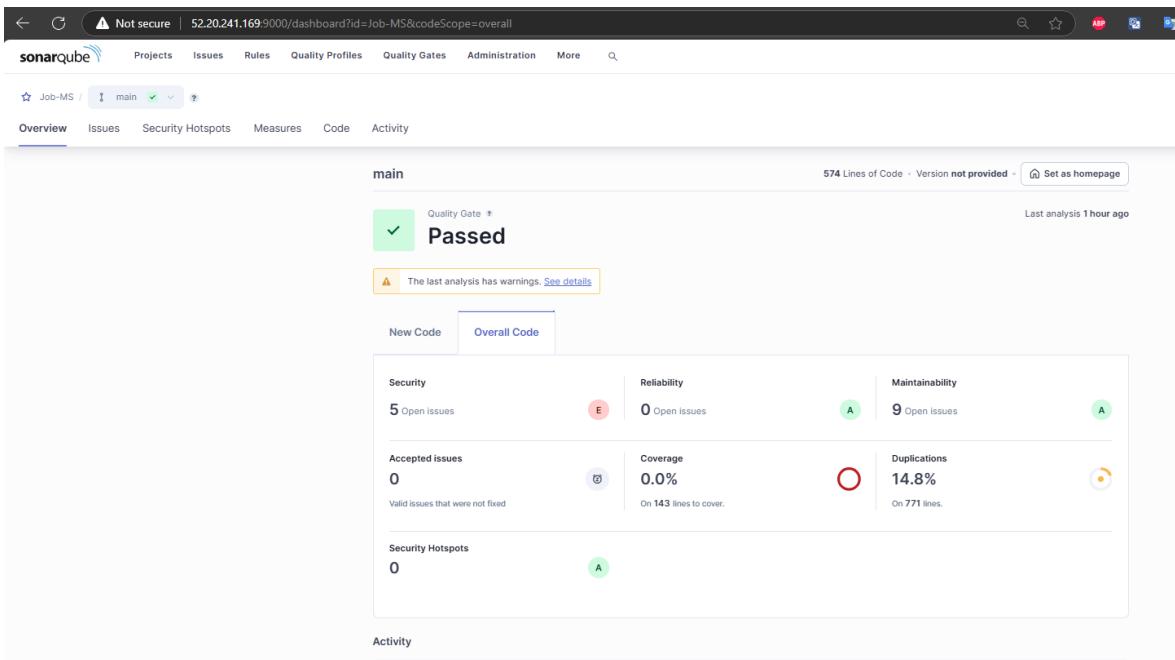
6.2 SonarQube

Các Microservices bao gồm Company Microservices, Job Microservices và Review Microservices được quét bằng Sonar Scanner và trả về kết quả Quality Gate là passed

Các thông số bao gồm Security, Reliability, Maintainability, Accepted Issues, Coverage, Duplications trên Dashboard của SonarQube.



Sonarqube Company Microservices



Sonarqube Job Microservices

The screenshot shows the SonarQube dashboard for the 'main' project. The overall status is 'Passed' (green). Key metrics shown include:

- Security:** 7 Open issues (E)
- Reliability:** 0 Open issues (A)
- Maintainability:** 3 Open issues (A)
- Accepted issues:** 0
- Coverage:** 0.0% (On 98 lines to cover)
- Duplications:** 11.5% (On 521 lines)
- Security Hotspots:** 0

A note at the bottom states: "The last analysis has warnings. See details".

SonarQube Review Microservices

6.3 Docker Hub

	Tags	OS	Vulnerabilities	Last pushed	Size
lambaoduy1310/companyms	v4.4 v4.1 v3.10	Ubuntu Ubuntu Ubuntu	0 0 0	1 day ago 1 day ago 5 days ago	171.32 MB 171.32 MB 171.32 MB
lambaoduy1310/reviewms	v4.1 v3.4 v1.5	Ubuntu Ubuntu Ubuntu	0 0 0	1 day ago 5 days ago 19 days ago	171.32 MB 171.32 MB 167.37 MB
lambaoduy1310/jobms	v4.1 v3.9 v3.8	Ubuntu Ubuntu Ubuntu	0 0 0	1 day ago 2 days ago 2 days ago	170.57 MB 166.48 MB 166.48 MB

Docker Hub

Các image được build pipeline Jenkins sẽ được tự động Push lên Docker Hub với version mới nhất của từng images là

- lambaoduy1310/companyms:v4.4
- lambaoduy1310/jobms:v4.1
- lambaoduy1310/reviewms:4.1

Các image này sẽ được cấu hình trong Deployments của các microservices để Jenkins Pipeline và deploy lên Kubernetes.

6.4 K3S

root@ip-192-168-1-183:/home/ubuntu# k3s kubectl get all -n jenkins					
NAME	READY	STATUS	RESTARTS	AGE	
pod/company-694dd4f96d-8chpd	1/1	Running	1 (30m ago)	22h	
pod/job-7bd7998f44-djdlj	1/1	Running	1 (30m ago)	22h	
pod/postgres-0	1/1	Running	1 (30m ago)	22h	
pod/review-856695746c-kr68n	1/1	Running	1 (30m ago)	22h	

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/company	LoadBalancer	10.43.20.46	<pending>	80:30938/TCP	22h
service/job	LoadBalancer	10.43.141.189	<pending>	80:31432/TCP	22h
service/postgres	ClusterIP	10.43.13.0	<none>	5432/TCP	22h
service/review	LoadBalancer	10.43.155.206	<pending>	80:30321/TCP	22h

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/company	1/1	1	1	22h
deployment.apps/job	1/1	1	1	22h
deployment.apps/review	1/1	1	1	22h

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/company-694dd4f96d	1	1	1	22h
replicaset.apps/job-7bd7998f44	1	1	1	22h
replicaset.apps/review-856695746c	1	1	1	22h

NAME	READY	AGE
statefulset.apps/postgres	1/1	22h

K3S

Các tài nguyên được deploy lên K3S bao gồm:

1. statefulset.apps:

- postgres

2. pod.apps:

- company, job, review, postgres-0

3. service:

- company, job, review với Type là LoadBalancer.
- postgres với Type là Cluster IP

4. deployment.apps:

- company, job, review

5. replicaset.apps:

- company, job, review với replicat set là 1 đảm bảo tính availability.

6.5 Prometheus

The screenshot shows the Prometheus Targets page with a list of monitored services:

- company-service (1/1 up)**:
 - Endpoint: http://52.20.241.169:30938/actuator/prometheus, State: UP, Labels: instance="52.20.241.169:30938", job="company-service". Last Scrape: 9.489s ago, Scrape Duration: 6.532ms.
- jenkins-k3s-node-exporter (1/1 up)**:
 - Endpoint: http://52.20.241.169:9100/metrics, State: UP, Labels: instance="52.20.241.169:9100", job="jenkins-k3s-node-exporter". Last Scrape: 12.504s ago, Scrape Duration: 26.794ms.
- job-service (1/1 up)**:
 - Endpoint: http://52.20.241.169:31432/actuator/prometheus, State: UP, Labels: instance="52.20.241.169:31432", job="job-service". Last Scrape: 11.567s ago, Scrape Duration: 22.974ms.
- prometheus (1/1 up)**:
 - Endpoint: http://localhost:9090/metrics, State: UP, Labels: instance="localhost:9090", job="prometheus". Last Scrape: 10.678s ago, Scrape Duration: 10.912ms.
- prometheus-grafana-node-exporter (1/1 up)**:
 - Endpoint: http://34.230.202.25:9100/metrics, State: UP, Labels: instance="34.230.202.25:9100", job="prometheus-grafana-node-exporter". Last Scrape: 11.965s ago, Scrape Duration: 46.106ms.
- review-service (1/1 up)**:
 - Endpoint: http://52.20.241.169:30321/actuator/prometheus, State: UP, Labels: instance="52.20.241.169:30321", job="review-service". Last Scrape: 13.976s ago, Scrape Duration: 8.002ms.

Prometheus Target

6.6 Grafana

The screenshot shows the Grafana Dashboards page with a list of available dashboards:

- Dashboards**: Create and manage dashboards to visualize your data.
- Search for dashboards and folders**: A search bar.
- Filter by tag**: A dropdown menu.
- Starred**: A checkbox.
- Name**: A column for dashboard names.
- Tags**: A column for tags, showing "linux" for one dashboard.
- Node Exporter Full**: A dashboard entry.
- Spring Boot Microservices**: A dashboard entry.

Grafana Dashboard

7. DEMO

7.1 Call API

<input checked="" type="checkbox"/> JOB_PORT	default	▼	8082	31432
<input checked="" type="checkbox"/> REVIEW_PORT	default	▼	8083	30321 ↗
<input checked="" type="checkbox"/> COMPANY_PORT	default	▼	8081	30938
<input checked="" type="checkbox"/> DEFAULT_URL	default	▼	http://127.0.0.1	http://52.20.241.169
Add new variable				

Tạo các biến môi trường dựa vào Port Service của K3S để call API

HTTP REVIEW COMPANY / companyms k8s / Post companies

POST [{{DEFAULT_URL}}: {{COMPANY_PORT}}/companies](#)

Params Authorization Headers (9) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL [JSON](#)

```
1 {
2   "name": "Google",
3   "description": "test"
4 }
```

Body Cookies Headers (5) Test Results [⌚](#) 201 Created

Pretty Raw Preview Visualize Text [🔗](#)

1 Company added successfully

Tạo thông tin công ty

HTTP REVIEW COMPANY / companyms k8s / Get companies

GET [{{DEFAULT_URL}}: {{COMPANY_PORT}}/companies](#)

Params Authorization Headers (7) Body Scripts Tests Settings

Query Params

	Key	Value
	Key	Value

Body Cookies Headers (5) Test Results [⌚](#) 200 OK

Pretty Raw Preview Visualize [JSON](#) [🔗](#)

```
1 [
2   {
3     "id": 1,
4     "name": "Google",
5     "description": "test"
6   }
7 ]
```

Lấy thông tin công ty vừa mới tạo

HTTP REVIEW COMPANY / reviewms k8s / Create Review

POST | {{DEFAULT_URL}}:{{REVIEW_PORT}}/reviews?companyId=1

Params • Authorization Headers (9) Body • Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON** ▾

```

1  {
2    "title": "Google Review",
3    "description": "Test Review",
4    "rating": 4.5,
5    "companyId": 1
6  }

```

Body Cookies Headers (5) Test Results | ⚙️ 200 OK

Pretty Raw Preview Visualize Text ▾ 1 Review Added Successfully

Tạo Review cho Công Ty

HTTP REVIEW COMPANY / reviewms k8s / Get Review

GET | {{DEFAULT_URL}}:{{REVIEW_PORT}}/reviews?companyId=1

Params • Authorization Headers (7) Body Scripts Tests Settings

Query Params

<input checked="" type="checkbox"/>	Key	Value
<input checked="" type="checkbox"/>	companyId	1
	Key	Value

Body Cookies Headers (5) Test Results | ⚙️

Pretty Raw Preview Visualize JSON ▾

```

1  [
2    {
3      "id": 1,
4      "title": "Google Review",
5      "description": "Test Review",
6      "rating": 4.5,
7      "companyId": 1
8    }
9  ]

```

Lấy Review của công ty

HTTP REVIEW COMPANY / jobms k8s / Create job

POST | {{DEFAULT_URL}}: {{JOB_PORT}} /jobs

Params Authorization Headers (9) **Body** • Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```

1  {
2    "title": "Software Engineer",
3    "description": "Test",
4    "minSalary": "10000",
5    "maxSalary": "20000",
6    "location": "HCM",
7    "companyId": 1
8  }

```

Body Cookies Headers (5) Test Results

201 Created

Pretty Raw Preview Visualize Text

1 Job added successfully

Tạo Job cho Công ty

HTTP REVIEW COMPANY / jobms k8s / Get Job

GET | {{DEFAULT_URL}}: {{JOB_PORT}} /jobs

Params Authorization Headers (7) **Body** Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```

1

```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1  [
2    {
3      "id": 1,
4      "title": "Software Engineer",
5      "description": "Test",
6      "minSalary": "10000",
7      "maxSalary": "20000",
8      "location": "HCM",
9      "company": {
10        "id": 1,
11        "name": "Google",
12        "description": "test"
13      },
14      "reviews": [
15        {
16          "id": 1,
17          "title": "Google Review",
18          "description": "Test Review",
19          "rating": 4.5
20        }
21      ]
22    }
23  ]

```

Lấy thông tin Job của công ty

JSON trả về của Review có thông tin của công ty và review về job của công đã tạo trước đó.

Các Microservices có thể call API tới nhau.

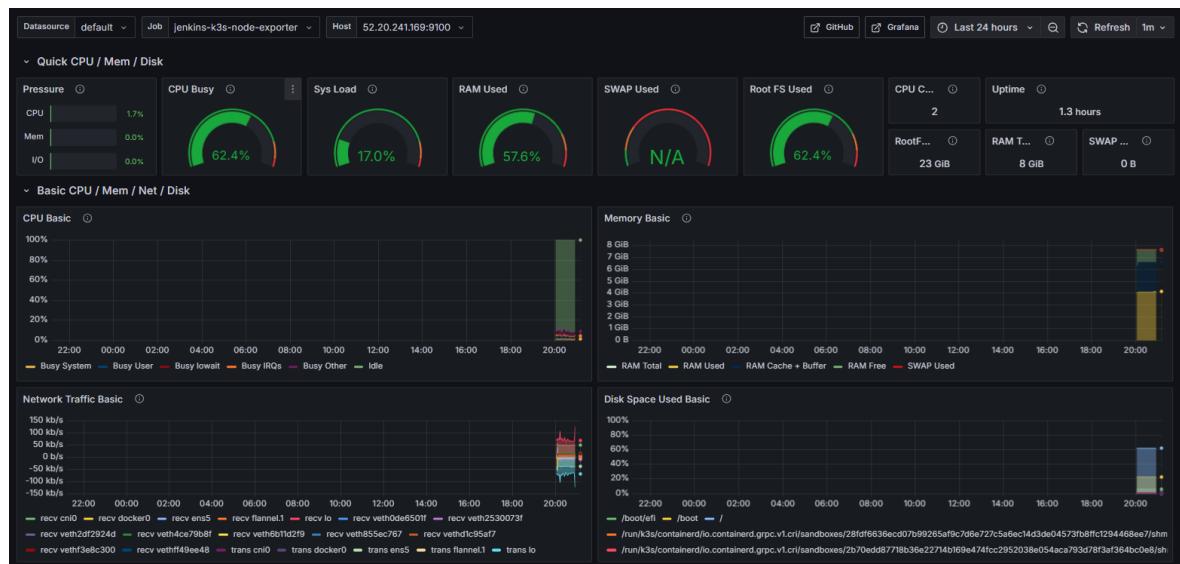
7.2 Prometheus

The screenshot shows the Prometheus Targets dashboard with five service instances listed:

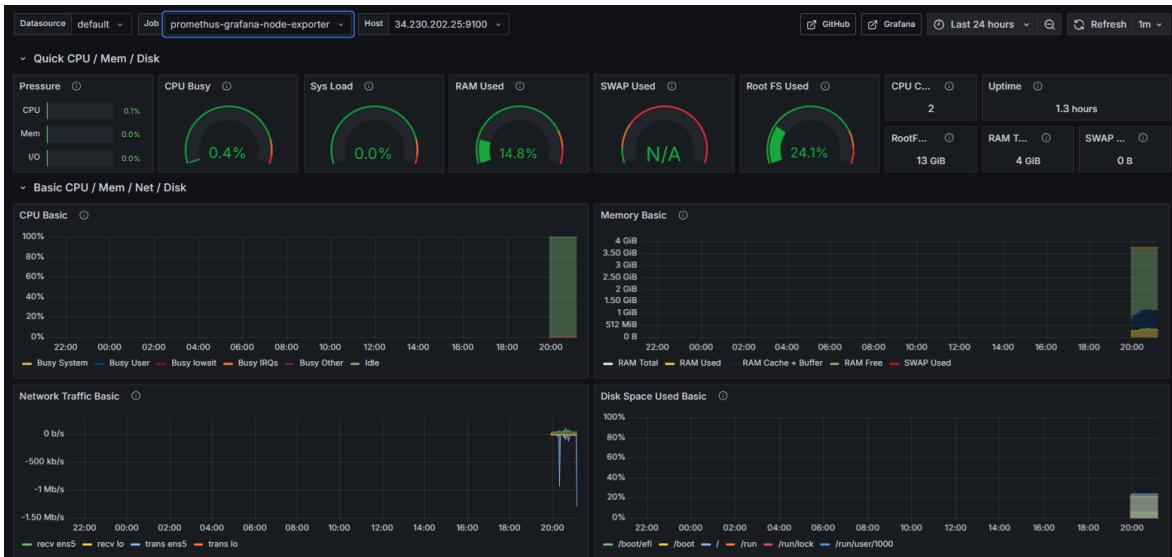
- company-service (1/1 up)**: Endpoint <http://52.20.241.169:30938/actuator/prometheus>, State UP, Labels instance="52.20.241.169:30938", job="company-service". Last Scrape: 9.489s ago, Scrape Duration: 6.532ms.
- jenkins-k3s-node-exporter (1/1 up)**: Endpoint <http://52.20.241.169:9100/metrics>, State UP, Labels instance="52.20.241.169:9100", job="jenkins-k3s-node-exporter". Last Scrape: 12.504s ago, Scrape Duration: 26.794ms.
- job-service (1/1 up)**: Endpoint <http://52.20.241.169:31432/actuator/prometheus>, State UP, Labels instance="52.20.241.169:31432", job="job-service". Last Scrape: 11.567s ago, Scrape Duration: 22.974ms.
- prometheus (1/1 up)**: Endpoint <http://localhost:9090/metrics>, State UP, Labels instance="localhost:9090", job="prometheus". Last Scrape: 10.678s ago, Scrape Duration: 10.912ms.
- prometheus-grafana-node-exporter (1/1 up)**: Endpoint <http://34.230.202.25:9100/metrics>, State UP, Labels instance="34.230.202.25:9100", job="prometheus-grafana-node-exporter". Last Scrape: 11.965s ago, Scrape Duration: 46.106ms.

Target Prometheus

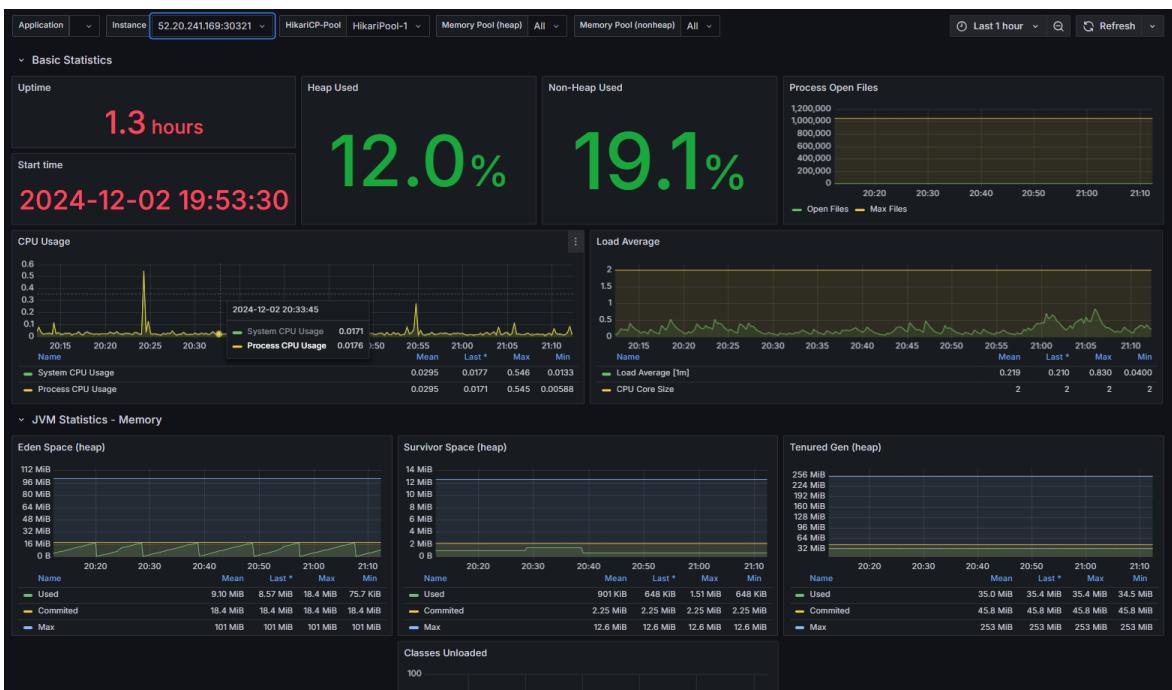
7.3 Grafana



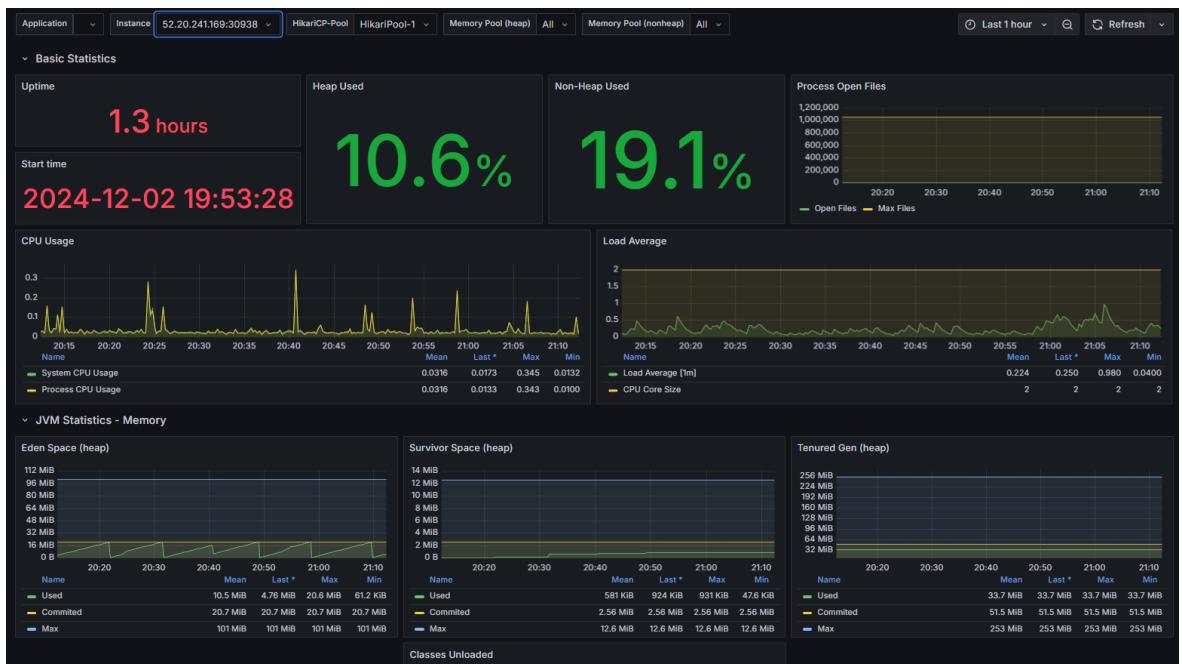
Node Exporter Grafana EC2 jenkins-k3s



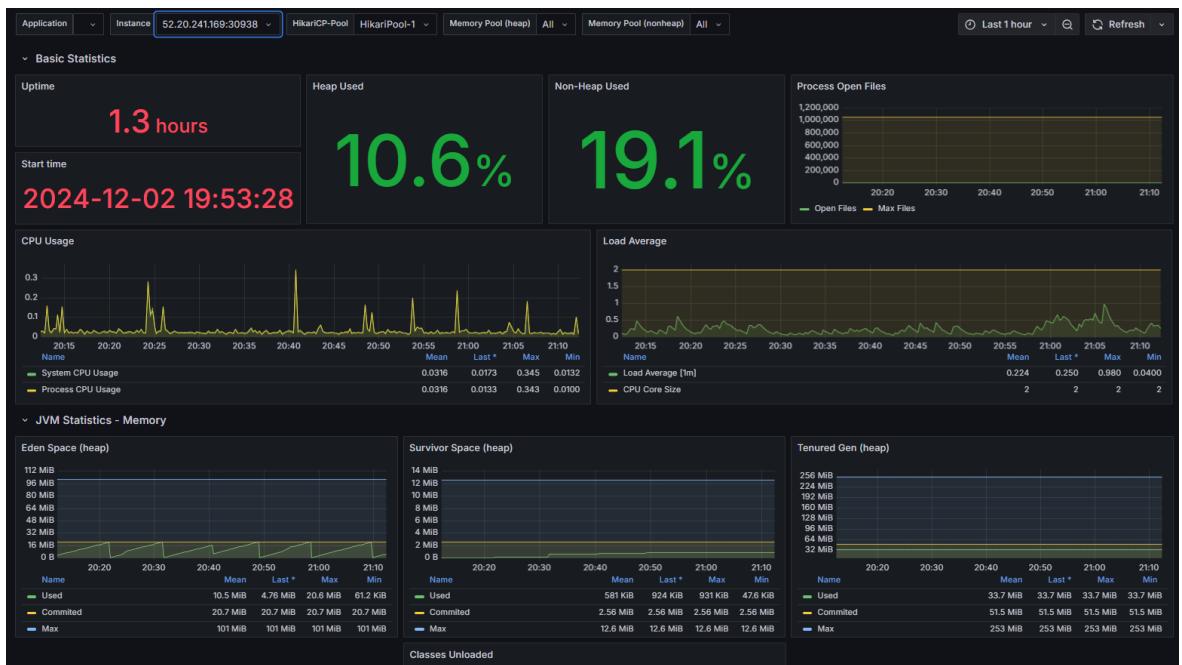
Node Exporter Grafana EC2 prometheus-grafana



Review microservices monitoring



Company microservices monitoring



Job microservices monitoring

8. TỔNG KẾT

- Link GitHub của dự án: <https://github.com/13octt/review-company>

Đề tài "Triển khai và giám sát hệ thống microservices sử dụng công cụ DevOps tích hợp công cụ bảo mật và giám sát" đã cung cấp một cái nhìn tổng

quan và chi tiết về việc ứng dụng các công cụ hiện đại để tối ưu hóa quy trình phát triển và vận hành phần mềm.

Mô hình tích hợp Spring Boot, Jenkins, Docker, và AWS CloudFormation không chỉ tự động hóa toàn bộ vòng đời CI/CD mà còn đảm bảo chất lượng mã nhờ SonarQube và duy trì an ninh hệ thống thông qua quy trình kiểm tra bảo mật. Đồng thời, với sự hỗ trợ của Prometheus và Grafana, hệ thống cung cấp khả năng giám sát hiệu quả, giúp phát hiện và xử lý sự cố nhanh chóng.

Ưu điểm chính của giải pháp này bao gồm:

- Tự động hóa giúp giảm thiểu lỗi thủ công và tăng hiệu suất triển khai.
- Kiểm tra chất lượng và bảo mật mã nguồn trong giai đoạn sớm.
- Giám sát toàn diện, đảm bảo tính sẵn sàng và ổn định của hệ thống.

Hạn chế và thách thức:

- Việc tích hợp nhiều công cụ phức tạp đòi hỏi kiến thức sâu rộng và thời gian thiết lập.
- Chi phí vận hành cao khi sử dụng cơ sở hạ tầng đám mây như AWS.

Định hướng phát triển:

- Tối ưu chi phí bằng cách sử dụng các giải pháp thay thế hoặc mã nguồn mở cho giám sát và triển khai.
- Mở rộng hệ thống để hỗ trợ thêm các dịch vụ hoặc người dùng lớn hơn.
- Tăng cường bảo mật bằng cách tích hợp thêm các công cụ quét lỗ hổng như Trivy hoặc các công cụ giám sát nâng cao.

Kết luận:

Đề tài đã chứng minh tính khả thi và hiệu quả của việc triển khai microservices kết hợp DevOps, bảo mật và giám sát. Đây là một giải pháp tiềm năng, mang lại giá trị lớn cho các doanh nghiệp đang tìm kiếm cách xây dựng và quản lý các hệ thống phần mềm hiện đại.

TÀI LIỆU THAM KHẢO

1. Amazon Web Services. (n.d.). *What is AWS CloudFormation?* Retrieved November 30, 2024, from <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide>Welcome.htm1>
2. K21 Academy. (n.d.). *AWS CloudFormation: Everything You Need to Know About AWS CloudFormation.* Retrieved November 30, 2024, from <https://k21academy.com/amazon-web-services/aws-devops/aws-cloudformation/>
3. Techopedia. (n.d.). *GitHub Definition.* Retrieved November 30, 2024, from <https://www.techopedia.com/definition/github>
4. GitHub. (n.d.). *What is DevOps?.* Retrieved December 1, 2024, from <https://github.com/resources/articles/devops/what-is-devops>
5. IBM. (n.d.). *What is Java Spring Boot?.* Retrieved December 1, 2024, from <https://www.ibm.com/topics/java-spring-boot>
6. DevOps School. (n.d.). *What is SonarQube and how it works: An overview and its use cases.* Retrieved December 1, 2024, from <https://www.devopsschool.com/blog/what-is-sonarqube-and-how-it-works-an-overview-and-its-use-cases/>
7. GeeksforGeeks. (n.d.). *Introduction to Docker.* Retrieved December 1, 2024, from <https://www.geeksforgeeks.org/introduction-to-docker/>
8. Aqua Security. (n.d.). *Trivy: Vulnerability scanner.* Retrieved December 1, 2024, from <https://github.com/aquasecurity/trivy>
9. GeeksforGeeks. (n.d.). *What is Jenkins?* Retrieved December 1, 2024, from <https://www.geeksforgeeks.org/what-is-jenkins/>
10. Codefresh. (n.d.). *What is Jenkins?* Retrieved December 1, 2024, from <https://codefresh.io/learn/jenkins/>
11. GeeksforGeeks. (n.d.). *Introduction to Kubernetes (K8s).* Retrieved December 1, 2024, from <https://www.geeksforgeeks.org/introduction-to-kubernetes-k8s/>
12. Prometheus. (n.d.). *Prometheus Documentation.* Retrieved December 1, 2024, from <https://prometheus.io/docs/introduction/overview/>

13. Walker, J. (2022, February 8). *What is Grafana and When Should You Use It?*. How-To Geek. Retrieved December 1, 2024, from <https://www.howtogeek.com/devops/what-is-grafana-and-when-should-you-use-it/>
14. Slingerland, C. (2023, September 12). *K3s Vs K8s: What's The Difference?*. CloudZero. Retrieved December 1, 2024, from <https://www.cloudzero.com/blog/k3s-vs-k8s/>
15. Aqua Security. (n.d.). *K3s: Lightweight Kubernetes*. Retrieved December 1, 2024, from <https://www.aquasec.com/cloud-native-academy/kubernetes-101/k3s/>