BÁO CÁO BÀI TẬP LỚN CUỐI KỲ CHƯƠNG TRÌNH VDT 2024

LĨNH VỰC CLOUD – GIAI ĐOẠN 1

**Sinh viên: Lê Minh Việt Anh – Đại học Bách khoa Hà Nội**

# 0 Môi trường thực hành:

Máy ảo Ubuntu desktop 24.04 được tạo trên Virtual Box (4 CPUs, 4 GB RAM, 80 GB ổ cứng) để triển khai Kubernetes.

Máy ảo Ubuntu server 22.04 được tạo trên Virtual Box (1 CPU, 2 GB RAM, 15 GB ổ cứng) để triển khai HAProxy.

# 1 Triển khai Kubernetes

|  |  |
| --- | --- |
| **Yêu cầu** | Triển khai được Kubernetes qua công cụ minikube trên 1 node, hoặc qua công cụ kubeadm hoặc kubespray lên 1 master node VM + 1 worker node VM |
| **Output** | Tài liệu cài đặt  Log của các lệnh kiểm tra hệ thống: **kubectl get nodes – o wide** |

## Tài liệu cài đặt Kubernetes qua công cụ minikube trên 1 node

Cần cài đặt: Docker, Kubernetes, minikube

### Cài đặt Docker

Bước 1: Cài đặt các package cần thiết

sudo apt update

sudo apt install apt-transport-https curl

Bước 2: Thêm khóa công khai chính thức của Docker lên máy Ubuntu

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o

/etc/apt/keyrings/docker.gpg

Bước 3: Cài đặt kho lưu trữ của Docker để cài đặt và cập nhật các phiên bản ổn định của Docker

echo "deb [arch=$(dpkg --print-architecture) signed - by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "$VERSION\_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

Bước 4: Làm mới kho lưu trữ

sudo apt update

Bước 5: Cài đặt Docker Engine

sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

Bước 6: Cho phép người dùng không có quyền root sử dụng Docker

sudo usermod -aG docker vietanh (có thể thay bằng username khác)

Bước 7: Gõ lệnh docker ps để kiểm tra

A screenshot of a computer

Description automatically generated

### Cài đặt kubecel

Bước 1: Cài đặt file nhị phân kubectl bằng curl

curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl

Bước 2: Tải file checksum kubectl

curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256

xác thực file nhị phân đã tải bằng file checksum, kết quả hiển thị OK

echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check

Bước 3: Cài đặt kubectl

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

Bước 4: Kiểm tra cài đặt kubectl thành công

A screenshot of a computer

Description automatically generated

### Cài đặt minikube

curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64

Chạy minikube

A screenshot of a computer

Description automatically generated

Log các lệnh kiểm tra hệ thống

A screenshot of a computer

Description automatically generated

Ta thấy minikube đã được triển khai trong máy ảo, với địa chỉ IP giao tiếp với máy ảo là 192.168.49.2/24

# K8S Helm Charts

|  |  |
| --- | --- |
| **Yêu cầu 1** | Cài đặt ArgoCD lên Kubernetes Cluster, expose được ArgoCD qua NodePort |
| **Output 1** | File manifests sử dụng để triển khai ArgoCD lên K8S Cluster  Ảnh chụp màn hình giao diện hệ thống ArgoCD khi truy cập qua trình duyệt |

## Cài đặt ArgoCD

Bước 1: Tạo namespace argocd

kubectl create namespace argocd

Bước 2: Tải argo trên namespace argocd

kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

Sử dụng file manifest có sẵn trên: <https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml>

## Expose ArgoCD qua NodePort

Bước 1: Đổi Service Type của argocd-server từ ClusterIP thành NodePort để có thể truy cập từ bên ngoài cluster

kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "NodePort"}}'

Bước 2:Lấy URL để truy cập giao diện ArgoCD từ trình duyệt

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Bước 3: Đăng nhập vào ArgoCD

Mặc định, username là admin, để lấy mật khẩu, ta gõ lệnh

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Giao diện hệ thống đã xuất hiện

|  |  |
| --- | --- |
| **Yêu cầu 2** | Viết 2 helm chart cho web deployment và api deployment, để vào 1 folder riêng trong repo web và repo api  Tạo 2 repo config cho web và api, trong các repo này chứa các file values.yaml là các cấu hình cần thiết để chạy web và api trên K8S bằng helm chart  Sử dụng tính năng multisource của ArgoCD để triển khai service we b và service api lên K8S Cluster, expose các service dưới dạng NodePort |
| **Output 2** | Các helm chart sử dụng để triển khai web và api lên k8s cluster  Các file values.yaml trong 2 config repo của web service và api service  maniefest của ArgoCD Application  Ảnh chụp giao diện hệ thống ArgoCD trên trình duyệt  Ảnh chụp giao diện WEB URL, API URL |

Helm chart:

Web: [vdt2024\_web/web-config at master · shukuchicoding/vdt2024\_web (github.com)](https://github.com/shukuchicoding/vdt2024_web/tree/master/web-config)

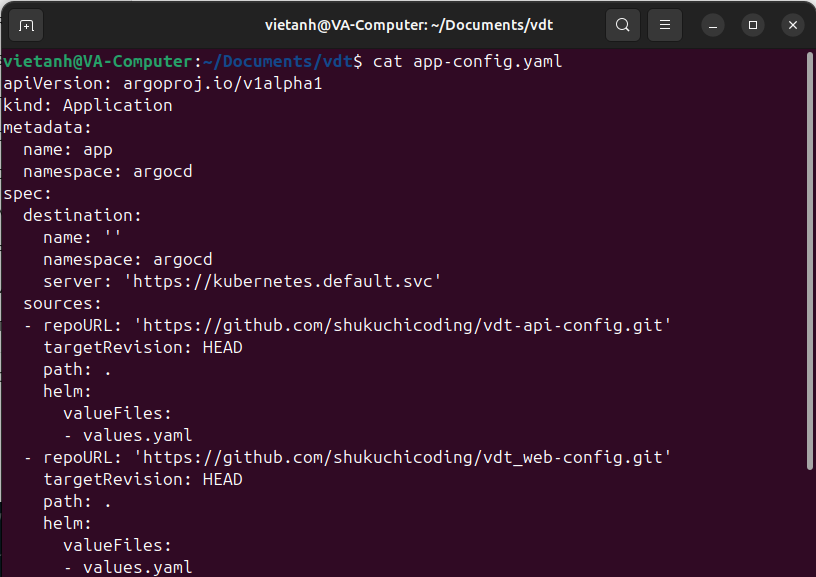
API: [vdt2024\_api/api-config at master · shukuchicoding/vdt2024\_api (github.com)](https://github.com/shukuchicoding/vdt2024_api/tree/master/api-config)

Config repo:

Web: [shukuchicoding/vdt\_web-config (github.com)](https://github.com/shukuchicoding/vdt_web-config)

API: [shukuchicoding/vdt-api-config (github.com)](https://github.com/shukuchicoding/vdt-api-config)

Manifest của Application:



A computer screen shot

Description automatically generated

Chạy lệnh sau

kubectl apply -f app-config.yaml -n argocd

sẽ hiển thị trên màn hình như sau

A screenshot of a chat

Description automatically generated

Truy cập WEB URL và API URL có kết quả như sau:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Continuous Delivery

File setup công cụ CI/CD

Api: [vdt2024\_api/.github/workflows/node.js.yml at master · shukuchicoding/vdt2024\_api](https://github.com/shukuchicoding/vdt2024_api/blob/master/.github/workflows/node.js.yml)

Web: [vdt2024\_web/.github/workflows/cicd.yml at master · shukuchicoding/vdt2024\_web](https://github.com/shukuchicoding/vdt2024_web/blob/master/.github/workflows/cicd.yml)

# Monitoring

# Logging

# Security

## Dựng HAProxy Loadbalancer

|  |  |
| --- | --- |
| **Yêu cầu 1** | Dựng HAProxy Loadbalancer trên 1 VM riêng với mode TCP, mở 2 port web\_port và api\_port trên LB trỏ đến 2 NodePort của Web Deployment và API Deployment của K8S cluster  Sử dụng 1 trong 2 giải pháp Ingress, hoặc HAProxy sidecar container cho các deployment, đảm bảo truy cập đến các port đều sử dụng https |
| **Output 1** | File cấu hình HAProxy Loadbalancer cho webport và apiport  File cấu hình ingress hoặc file cấu hình deployment sau khi thêm HAProxy Sidecar container vào Deployment  Kết quả truy cập vào web port và api port từ trình duyệt thông qua https hoặc dùng curl |

Ở phần này, ta sử dụng một máy ảo Ubuntu server 22.04

Thiết lập sơ đồ mạng như sau:

A black circle with white text

Description automatically generated

Thực hiện các bước sau trên HAProxyLoadbalancer:

Bước 1: Cài đặt haproxy

sudo apt install haproxy -y

Bước 2: Sử dụng nano chỉnh sửa file haproxy.cfg

sudo nano /etc/haproxy/haproxy.cfg

Bước 3: Chỉnh sửa file cấu hình

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

## Xác thực và phân quyền

|  |  |
| --- | --- |
| **Yêu cầu 2** | Đảm bảo một số URL của api service khi truy cập phải có xác thực thông qua 1 trong số các phương thức cookie, basic auth, token auth, nếu không sẽ trả về HTTP response code 403  Thực hiện phân quyền cho hai loại người dùng trên API:   * Nếu người dùng có role là user thì truy cập vào GET request trả về 200, còn truy cập vào POST/DELETE thì trả về 403 * Nếu người dùng có role là admin thì truy cập vào GET request trả về 200, còn truy cập vào POST/DELETE thì trả về 2xx |
| **Output 2** | File trình bày giải pháp sử dụng để authen/authorization cho các service  Kết quả HTTP response khi curl hoặc dùng postman gọi vào các URL khi truyền thêm thông tin xác thực và khi không truyền thông tin xác thực  Kết quả HTTP response khi curl hoặc dùng postman vào các URL với các method GET/POST/DELETE khi lần lượt dùng thông tin xác thực của các user có role là admin và user |

Giải pháp sử dụng: JWT và JSONWEBTOKEN

Bước 1: Cài đặt express-jwt và jsonwebtoken

npm install express-jwt jsonwebtoken

Bước 2: Viết thêm các hàm login, generateToken, authenticateToken, authorizeRole phục vụ cho xác thực danh tính và xác thực quyền

const generateToken = (user) => {

    return jwt.sign({ id: user.\_id, role: user.role }, "123456789", { expiresIn: '12h' });

};

const authenticateToken = (req, res, next) => {

    const token = req.cookies.token;

    console.log("Token: ",token);

    if (!token) {

        return res.status(403).json({ message: 'No token provided' });

    }

    jwt.verify(token, "123456789", (err, user) => {

        if (err) {

            return res.status(403).json({ message: 'Failed to authenticate token' });

        }

        req.user = user;

        next();

    });

};

const authorizeRoles = (...roles) => {

    return (req, res, next) => {

        console.log("Roles: ",req.user.role);

        if (!roles.includes(req.user.role)) {

            return res.status(403).json({ message: 'Access denied', role: req.user.role, requiredRoles: roles});

        }

        next();

    };

};

//dang nhap

app.post("/check", async (req, res) => {

    try {

        const { username, password } = req.body;

        const user = await student.findOne({ username });

        if (!user || user.username === 'none' || user.password === 'none')  {

            return res.status(401).json({ message: 'Invalid username' });

        }

        if (!compare(password, user.password)) {

            return res.status(401).json({ message: 'Invalid password' });

        }

        console.log("touch");

        const token = generateToken(user);

        res.cookie('token', token, { httpOnly: true });

        res.status(200).json({ message: `Logged in successfully as ${user.role}`, token, user });

    } catch (error) {

        logger.error({

            message: error.message,

            path: req.originalUrl,

            method: req.method,

            responseCode: 500

        });

        res.status(500).json({ message: error.message });

    }

});

Kết quả chạy postman:

## Rate limit

|  |  |
| --- | --- |
| **Yêu cầu 3** | Viết 2 helm chart cho web deployment và api deployment, để vào 1 folder riêng trong repo web và repo api  Tạo 2 repo config cho web và api, trong các repo này chứa các file values.yaml là các cấu hình cần thiết để chạy web và api trên K8S bằng helm chart  Sử dụng tính năng multisource của ArgoCD để triển khai service we b và service api lên K8S Cluster, expose các service dưới dạng NodePort |
| **Output 3** | Các helm chart sử dụng để triển khai web và api lên k8s cluster  Các file values.yaml trong 2 config repo của web service và api service  maniefest của ArgoCD Application  Ảnh chụp giao diện hệ thống ArgoCD trên trình duyệt  Ảnh chụp giao diện WEB URL, API URL |

Sử dụng express-rate-limit để giới hạn số lần truy cập endpoint của API, giải pháp này đơn giản, dễ sử dụng trong các dự án node js.

Bước 1: Cài đặt express-rate-limit

npm install express-rate-limit

Bước 2: Trong index.js của API, thêm các đoạn mã sau:

const { rateLimit } = require("express-rate-limit");

const apiLimiter = rateLimit({

    windowMs: 60 \* 1000, // Thời gian cửa sổ (trong mili giây)

    max: 10, // Số lượng yêu cầu tối đa trong cửa sổ thời gian

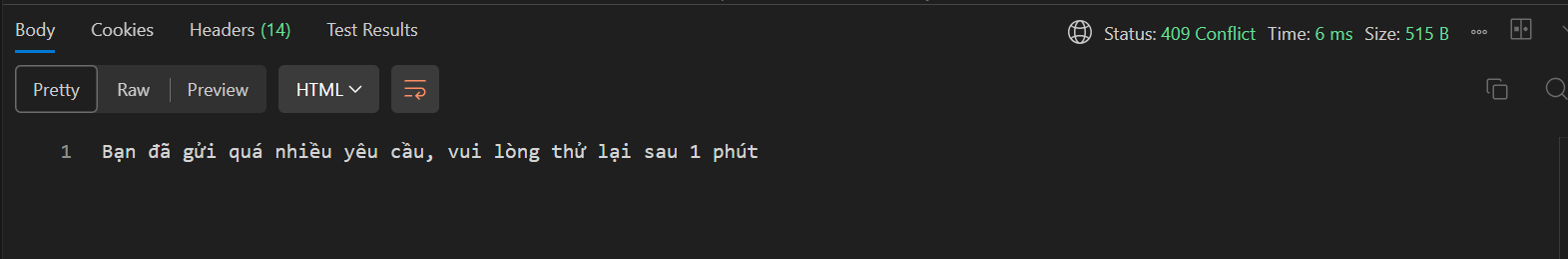
    message: "Bạn đã gửi quá nhiều yêu cầu, vui lòng thử lại sau 1 phút",

    statusCode: 409, // Mã HTTP Response trả về khi vượt quá giới hạn

});

app.use(apiLimiter);

Bước 3: Kiểm tra kết quả bằng Postman



A black screen with white lines

Description automatically generated