DISTRIBUTED SYSTEMS – TRAN HAI ANH

**BÀI THỰC HÀNH**

**HỌC PHẦN: CÁC HỆ THỐNG PHÂN TÁN VÀ Ứ.D CHƯƠNG 1: TỔNG QUAN VÀ KIẾN TRÚC HPT**

**1. Interface trong Các hệ thống phân tán**

**1.1. Nội dung**

Trong hệ thống phân tán, **Interface** (giao diện) là tập hợp các quy ước chuẩn mà cả bên cung cấp dịch vụ lẫn bên sử dụng dịch vụ đều phải tuân thủ. Interface quy định cách thức trao đổi thông điệp, cú pháp lệnh, ngữ nghĩa của phản hồi, và phiên bản của giao thức. Nhờ có Interface, các hệ thống khác nhau có thể tương tác với nhau một cách thống nhất, tránh tình trạng “mạnh ai nấy làm”. Bài thực hành này giúp sinh viên tiếp cận khái niệm Interface một cách cụ thể thông qua việc xây dựng một dịch vụ phân tán nhỏ dạng **Mini Key–Value Store Service (KVSS)**. Trong đó, sinh viên sẽ vừa đóng vai trò **server** (cung cấp dịch vụ lưu trữ và truy xuất dữ liệu key–value) vừa đóng vai trò **client** (gửi yêu cầu theo đúng chuẩn Interface). Qua quá trình cài đặt, kiểm thử và quan sát, sinh viên sẽ hiểu rõ hơn vai trò của Interface trong việc đảm bảo tính thống nhất, khả năng tương thích, và dễ dàng mở rộng của các dịch vụ trong hệ thống phân tán.

**1.2. Yêu cầu**

• Ngôn ngữ: tùy chọn (gợi ý **Python** cho nhanh; bonus **C**).

• Môi trường: Linux (Ubuntu) + công cụ: nc, telnet, curl, **Wireshark**. • Kiến trúc: TCP, mô hình request–response, **text-based line protocol** để nhìn gói tin dễ.

**1.3. Các bước thực hành**

Sinh viên sẽ xây dựng một Mini Key–Value Store Service (KVSS) gồm Server và Client theo Interface sau.

**Interface Specification (bắt buộc)**

• **Kết nối**: TCP host:port (mặc định 127.0.0.1:5050).

• **Đơn vị thông điệp**: dòng văn bản kết thúc bằng \n (LF).

• **Mã hóa**: UTF-8.

• **Phiên bản**: mọi request bắt đầu bằng tiền tố KV/1.0 (hoặc header riêng ở dạng lệnh).

• **Lệnh hợp lệ** (BNF giản lược):

Cú pháp: <version> " " <command> [ " " <args> ] "\n"

<version> ::= "KV/1.0"

<command> ::= "PUT" | "GET" | "DEL" | "STATS" | "QUIT"

<args> ::= <key> [ " " <value> ]

1

DISTRIBUTED SYSTEMS – TRAN HAI ANH

<key> ::= 1\*VCHAR (không chứa khoảng trắng)

<value> ::= 1\*VCHAR (tồn tại khi PUT)

Các phản hồi trả về:

• 200 OK [data]

• 201 CREATED (khi PUT tạo mới)

• 204 NO\_CONTENT (xóa thành công)

• 400 BAD\_REQUEST (sai cú pháp/thiếu tham số)

• 404 NOT\_FOUND

• 426 UPGRADE\_REQUIRED (nếu thiếu/khác phiên bản)

• 500 SERVER\_ERROR

| Câu hỏi 1: Interface trong hệ thống phân tán là gì? Tại sao cần phải có Interface khi triển khai các dịch vụ? |
| --- |

| Câu hỏi 2: Hãy giải thích ý nghĩa của mã trạng thái 201 CREATED, 204 NO\_CONTENT và 404 NOT\_FOUND trong giao thức KVSS. |
| --- |

Các ví dụ:

C: KV/1.0 PUT user42 Alice

S: 201 CREATED

C: KV/1.0 GET user42

S: 200 OK Alice

C: KV/1.0 DEL user42

S: 204 NO\_CONTENT

C: KV/1.0 GET user42

S: 404 NOT\_FOUND

2

DISTRIBUTED SYSTEMS – TRAN HAI ANH

C: KV/1.0 STATS

S: 200 OK keys=0 uptime=12s served=7

C: KV/1.0 QUIT

S: 200 OK bye

Hãy xây dựng 2 chương trình server và client như sau:

• Với **server**:

o Server TCP, **đơn luồng** (baseline) hoặc multiplexing (select()), xử lý **nhiều client tuần tự**.

o Lưu trữ KV trong bộ nhớ (dictionary/map).

o Ghi log mọi request/response theo timestamp.

o Đảm bảo **idempotent** cho GET, STATS, DEL (DEL nhiều lần với key không tồn tại phải trả 404 NOT\_FOUND).

• Với client:

o Client dòng lệnh, đọc stdin; gửi request theo chuẩn; in nguyên văn status + data.

Yêu cầu:

• Hãy viết 8–10 ca kiểm thử, bao gồm ca hợp lệ và ca lỗi (ví dụ thiếu version, thiếu value…).

• Hãy sử dụng nc/telnet để tương tác thủ công. Hãy cài đặt và sử dụng Wireshark để quan sát các phiên trao đổi.

| Câu hỏi 3: Trong bài lab KVSS, nếu client không tuân thủ quy ước Interface (ví dụ: thiếu version KV/1.0), server sẽ phản hồi thế nào? Tại sao phải quy định rõ ràng tình huống này? |
| --- |

| Câu hỏi 4: Quan sát một phiên làm việc qua Wireshark: hãy mô tả cách mà gói tin TCP được chia để truyền thông điệp theo “line-based protocol”. |
| --- |

| Câu hỏi 5: Giả sử có một client viết sai giao thức (gửi KV/1.0 POTT user42 Alice). Server sẽ xử lý như thế nào? Kết quả này thể hiện đặc điểm gì của Interface? |
| --- |

3

DISTRIBUTED SYSTEMS – TRAN HAI ANH

**2. Kiến trúc Microservices**

**2.1.Nội dung**

Ở bài thực hành này chúng ta sẽ xây dựng một kiến trúc microservices đơn giản. Cụ thể, chúng ta sẽ triển khai một ứng dụng thương mại điện tử dựa trên kiến trúc microservices sử dụng Kubernetes. Bài thực hành này sẽ cung cấp trải nghiệm thực tế về:

• Cài và sử dụng **kubectl**, **minikube** (hoặc Docker Desktop Kubernetes). • Triển khai các Deployment + Service cho 3 microservices

(users, catalog, orders).

• Cấu hình **Ingress** làm API Gateway (reverse proxy).

• Scale số replica cho 1 microservice (ví dụ catalog).

• Gọi API và quan sát kết quả.

**2.2.Yêu cầu**

**2.2.1. Lý thuyết**

• Microservices

• Kube fundamentals: Pods, Services, Deployments et al.

• Docker

**2.2.2. Phần cứng**

• Laptop/PC on Windows

**2.2.3. Phần mềm**

• VirtualBox

• Docker

• The kubernetes command line tool *kubectl*

• The minikube binary

• Git bash

**2.3.Các bước thực hành**

**Cài đặt môi trường**

- Cài đặt Docker:

Tải và cài đặt Docker từ: https://docs.docker.com/get-docker/

- Cài đặt công cụ Kubernetes (kubectl, Minikube):

Cài đặt kubectl và thiết lập Minikube:

- Cài đặt kubectl: tham khảo phần Phụ lục 2.4 phía dưới

- Cài đặt Minikube:

4

DISTRIBUTED SYSTEMS – TRAN HAI ANH

Làm theo hướng dẫn cài đặt Minikube tại

https://minikube.sigs.k8s.io/docs/start/

**Khởi động cụm Minikube**

minikube start

Xác nhận trạng thái:

kubectl get nodes

Sau đó bật add-on ingress:

minikube addons enable ingress

Hãy tạo cấu trúc thư mục project của chúng ta như sau:

k8s-microservices-lab/

├─ users-deploy.yaml

├─ catalog-deploy.yaml

├─ orders-deploy.yaml

├─ gateway-ingress.yaml

Tải các file cấu hình Kubernetes đặt vào thư mục như trên:

File users-deploy.yaml

https://www.dropbox.com/scl/fi/p9na40ayp0txu7oix4ig9/users deploy.yaml?rlkey=eluxlqzks7k3jni43umy9vcm8&dl=0

File catalog-deploy.yaml

https://www.dropbox.com/scl/fi/e9xyl2n50r9ldtmdre1n3/catalog deploy.yaml?rlkey=ljhwn5rrpf9c1qolxab1yr2u8&dl=0

File orders-deploy.yaml

https://www.dropbox.com/scl/fi/e10pdb3oqxhqrh7wnwb7v/orders deploy.yaml?rlkey=mkgip0j9tudiopynx6l0q4zfg&dl=0

File gateway-ingress.yaml

https://www.dropbox.com/scl/fi/qaenzymasdv687vgya9bx/gateway ingress.yaml?rlkey=oss096wdidbpk9h2z3plqv3r5&dl=0

**Triển khai hệ thống**

1. Tạo namespace (tùy chọn):

kubectl create namespace micro-lab

kubectl config set-context --current --namespace=micro-lab

5

DISTRIBUTED SYSTEMS – TRAN HAI ANH

2. Apply các file:

kubectl apply -f users-deploy.yaml

kubectl apply -f catalog-deploy.yaml

kubectl apply -f orders-deploy.yaml

kubectl apply -f gateway-ingress.yaml

3. Kiểm tra:

kubectl get pods

kubectl get svc

kubectl get ingress

4. Nếu dùng Minikube, bật ingress host:

minikube tunnel

hoặc lấy IP:

minikube ip

rồi thêm vào file /etc/hosts:

<MINIKUBE\_IP> micro.local

**Kiểm tra hoạt động**

• Truy cập:

o http://micro.local/users

o http://micro.local/catalog

o http://micro.local/orders

• Hoặc dùng curl:

o curl http://micro.local/users

o curl http://micro.local/catalog

o curl http://micro.local/orders

→ Kết quả: JSON echo chứa hostname Pod, xác nhận request đi đúng microservice.

**Scale dịch vụ**

Ví dụ scale catalog lên 3 replicas:

kubectl scale deploy catalog-deploy --replicas=3

kubectl get pods -l app=catalog

Gọi nhiều lần /catalog, bạn sẽ thấy hostname thay đổi → chứng **minh** khả năng cân bằng tải của Kubernetes Service.

6

DISTRIBUTED SYSTEMS – TRAN HAI ANH

**Kết thúc, hãy thực hiện dọn dẹp**

kubectl delete -f gateway-ingress.yaml

kubectl delete -f users-deploy.yaml

kubectl delete -f catalog-deploy.yaml

kubectl delete -f orders-deploy.yaml

kubectl delete namespace micro-lab

**Trả lời các câu hỏi sau:**

| Câu hỏi 6: Sau khi chạy kubectl apply -f users-deploy.yaml, dùng lệnh nào để kiểm tra Pod của service users đã chạy thành công? Hãy chụp màn hình kết quả. |
| --- |

| Câu hỏi 7: Trong file users-deploy.yaml, hãy chỉ ra:  **- Deployment** quản lý bao nhiêu replica ban đầu?  **- Service** thuộc loại nào (ClusterIP, NodePort, LoadBalancer)? |
| --- |

| Câu hỏi 8: Sau khi cài Ingress, em cần thêm dòng nào vào file /etc/hosts để truy cập bằng tên miền micro.local? |
| --- |

**2.4. Phụ lục**

**Hướng dẫn cài đặt kubectl**

Các bước cài đặt kubectl trên macOS, Linux và Windows.

*macOS*

Cách A — Homebrew (khuyên dùng)

1. Cài qua Homebrew:

brew install kubectl # hoặc: brew install kubernetes-cli 2. Kiểm tra:

kubectl version --client

Cách B — Tải file nhị phân (không cần Homebrew)

1. Tải bản mới nhất (tùy chip):

• Intel:

7

DISTRIBUTED SYSTEMS – TRAN HAI ANH

curl -LO "https://dl.k8s.io/release/$(curl -Ls

https://dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl" • Apple Silicon:

curl -LO "https://dl.k8s.io/release/$(curl -Ls

https://dl.k8s.io/release/stable.txt)/bin/darwin/arm64/kubectl" 2. (Tuỳ chọn) Kiểm tra checksum:

curl -LO "https://dl.k8s.io/release/$(curl -Ls

https://dl.k8s.io/release/stable.txt)/bin/darwin/$(uname -m | sed 's/x86\_64/amd64/')/kubectl.sha256"

echo "$(cat kubectl.sha256) kubectl" | shasum -a 256 --check 3. Cài vào PATH:

chmod +x kubectl

sudo mv ./kubectl /usr/local/bin/kubectl

kubectl version --client

Chú ý: cần chắc /usr/local/bin nằm trong PATH.

*Linux*

Cách A — Dùng trình quản lý gói (Debian/Ubuntu, RHEL/CentOS, SUSE) **Debian/Ubuntu (APT):**

sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl gnupg

sudo mkdir -p -m 755 /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.34/deb/Release.key \ | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.34/deb/ /' \

| sudo tee /etc/apt/sources.list.d/kubernetes.list sudo apt-get update

sudo apt-get install -y kubectl

kubectl version --client

**RHEL/CentOS (YUM):**

cat <<'EOF' | sudo tee /etc/yum.repos.d/kubernetes.repo [kubernetes]

name=Kubernetes

baseurl=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/ enabled=1

gpgcheck=1

gpgkey=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/repodata/repomd.xm l.key

EOF

sudo yum install -y kubectl

kubectl version --client

**SUSE (Zypper):**

8

DISTRIBUTED SYSTEMS – TRAN HAI ANH

cat <<'EOF' | sudo tee /etc/zypp/repos.d/kubernetes.repo [kubernetes]

name=Kubernetes

baseurl=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/ enabled=1

gpgcheck=1

gpgkey=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/repodata/repomd.xm l.key

EOF

sudo zypper update

sudo zypper install -y kubectl

kubectl version --client

Cách B — Tải file nhị phân (dùng cho mọi distro)

1. Tải bản mới nhất (chọn kiến trúc CPU):

# x86-64

curl -LO "https://dl.k8s.io/release/$(curl -Ls

https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl" # ARM64

curl -LO "https://dl.k8s.io/release/$(curl -Ls

https://dl.k8s.io/release/stable.txt)/bin/linux/arm64/kubectl" 2. (Tuỳ chọn) Kiểm tra checksum:

curl -LO "https://dl.k8s.io/release/$(curl -Ls

https://dl.k8s.io/release/stable.txt)/bin/linux/$(uname -m | sed 's/x86\_64/amd64/')/kubectl.sha256"

echo "$(cat kubectl.sha256) kubectl" | sha256sum --check 3. Cài vào PATH:

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

Nếu không có quyền root, bạn có thể cài vào ~/.local/bin và thêm vào PATH.

*Windows*

Cách A — Trình quản lý gói (dễ nhất)

Chạy **một** trong các lệnh sau trong PowerShell/CMD (chế độ Admin):

choco install kubernetes-cli

# hoặc

scoop install kubectl

# hoặc

winget install -e --id Kubernetes.kubectl

Kiểm tra:

kubectl version --client

Cách B — Tải file nhị phân

1. Tải:

9

DISTRIBUTED SYSTEMS – TRAN HAI ANH

curl.exe -LO

"https://dl.k8s.io/release/v1.34.0/bin/windows/amd64/kubectl.exe" # (kiểm tra https://dl.k8s.io/release/stable.txt để biết bản mới nhất)

2. (Tuỳ chọn) Kiểm tra checksum:

$(Get-FileHash -Algorithm SHA256 .\kubectl.exe).Hash -eq $(Invoke WebRequest -UseBasicParsing

https://dl.k8s.io/v1.34.0/bin/windows/amd64/kubectl.exe.sha256).Conte nt.Trim()

3. Chép kubectl.exe vào thư mục nằm trong PATH (ví

dụ: C:\Windows\System32 hoặc một thư mục tools).

4. Kiểm tra:

kubectl version --client

Sau khi cài đặt (mọi hệ điều hành)

• Đảm bảo phiên bản kubectl **khớp** với version của cluster (ví dụ: kubectl v1.34 sẽ tương thích với cluster v1.33–v1.35).

• Nếu dùng Minikube hoặc Docker Desktop, khi đã có cluster, kiểm tra kết nối: kubectl cluster-info

Nếu hiện URL của Kubernetes control plane → kubectl đã kết nối thành công.

10