

Bài tập cuối khoá lĩnh vực Cloud

Chi tiết đề bài

Triển khai Kubernetes (1 điểm)

Yêu cầu:

Y/c 1:

- Triển khai được Kubernetes thông qua công cụ minikube trên 1 node: 0.5 điểm

Hoặc

- Triển khai được Kubernetes thông qua công cụ kubeadm hoặc kubespray lên 1 master node VM + 1 worker node VM: 1 điểm

Output:

- Tài liệu cài đặt (công cụ gì, các file config, ...)
- Ảnh chụp Log của các lệnh kiểm tra hệ thống như: `kubectl get nodes -o wide`, `kubectl get pods -A -o wide`

Triển khai web application sử dụng các DevOps tools & practices

Sinh viên chọn 1 app bất kỳ theo cấu trúc microservices (ví dụ web app và api service).

K8S Helm Chart (1.5đ)

Yêu cầu 1:

- Cài đặt ArgoCD lên Kubernetes Cluster, expose được ArgoCD qua NodePort
- Cài đặt Jenkins lên Kubernetes Cluster, expose được Jenkins qua NodePort

Output 1:

- File manifests sử dụng để triển khai ArgoCD lên K8S Cluster
- Ảnh chụp giao diện màn hình hệ thống ArgoCD khi truy cập qua trình duyệt trình duyệt
- File manifests sử dụng để triển khai Jenkins lên K8S Cluster
- Ảnh chụp giao diện màn hình hệ thống Jenkins khi truy cập qua trình duyệt trình duyệt

Yêu cầu 2:

- Viết hoặc tìm mẫu Helm Chart cho app bất kỳ, để vào 1 folder riêng trong repo app
- Tạo Repo Config cho app trên, trong repo này chứa các file values.yaml với nội dung của cá file values.yaml là các config cần thiết để chạy ứng dụng trên k8s bằng Helm Chart

Output 2:

- Các Helm Chart sử dụng để triển khai app lên K8S Cluster
- Các file values.yaml trong config repo của app
- Manifest của ArgoCD Application
- Ảnh chụp giao diện màn hình hệ thống ArgoCD trên trình duyệt
- Ảnh chụp giao diện màn hình trình duyệt khi truy cập vào Web URL, API URL

CI/CD (1.5đ)

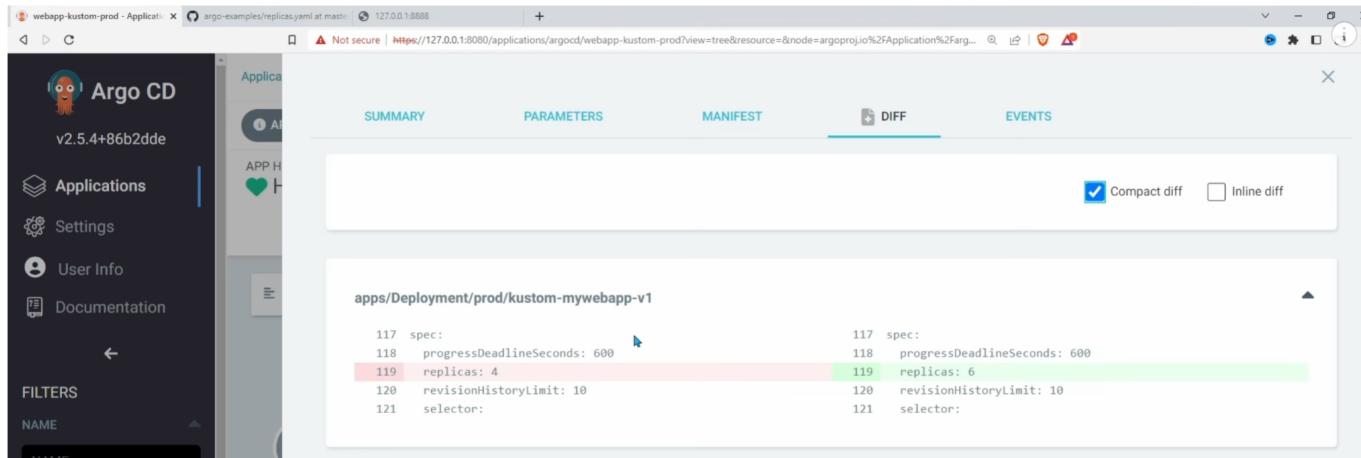
Yêu cầu:

- Viết 1 luồng CI/CD cho app, khi có thay đổi từ source code, 1 tag mới được tạo ra trên repo này thì luồng CI/CD tương ứng của repo đó thực hiện các công việc sau:
 - Sửa code trong source code
 - Thực hiện build source code trên jenkin bằng docker với image tag là tag name đã được tạo ra trên gitlab/github và push docker image sau khi build xong lên Docker Hub
 - Sửa giá trị Image version trong file values.yaml trong config repo và push thay đổi lên config repo.
 - Cấu hình ArgoCD tự động triển khai lại web Deployment và api Deployment khi có sự thay đổi trên config repo.

Output:

- Các file setup công cụ của luồng CI/CD
- Output log của luồng CI/CD khi tạo tag mới trên repo app

- Show log chứng minh jenkin đã chạy đúng
- Jenkin file cấu hình các luồng
- Ảnh luồng CI/CD chạy qua các stage trên giao diện Jenkins (sử dụng Plugin Pipeline Stage View)
- Hình ảnh app triển khai argoCD, hình ảnh diff khi argoCD phát hiện thay đổi ở config repo tương tự hình ảnh sau



- Hình ảnh app trước khi sửa code và sau khi sửa code.

Monitoring (1.5đ)

Yêu cầu:

- Expose metric của app ra 1 http path. Tham khảo:
 - <https://github.com/korfuri/django-prometheus>
- Sử dụng ansible playbooks để triển khai container Prometheus server. Sau đó cấu hình prometheus add target giám sát các metrics đã expose ở trên.

Output:

- Các file setup để triển khai Prometheus
- Hình ảnh khi truy cập vào Prometheus UI thông qua trình duyệt
- Hình ảnh danh sách target của App được giám sát bởi Prometheus, ví dụ:

yb-demo/cluster-1-yugabyte-yb-master/0 (3/3 up) show less						
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error	
http://10.36.2.3:7000/prometheus-metrics	UP	endpoint=ui, export_type=master_export, group=yb-master, instance=10.36.2.3:7000, job=cluster-1, namespace=yb-demo, node_prefix=cluster-1, pod=yb-master-2, service=yb-masters	18.936s ago	8.952ms		
http://10.36.3.3:7000/prometheus-metrics	UP	endpoint=ui, export_type=master_export, group=yb-master, instance=10.36.3.3:7000, job=cluster-1, namespace=yb-demo, node_prefix=cluster-1, pod=yb-master-1, service=yb-masters	26.362s ago	9.077ms		
http://10.36.5.3:7000/prometheus-metrics	UP	endpoint=ui, export_type=master_export, group=yb-master, instance=10.36.5.3:7000, job=cluster-1, namespace=yb-demo, node_prefix=cluster-1, pod=yb-master-0, service=yb-masters	20.58s ago	13.21ms		

Logging (1.5d)

Yêu cầu:

- Sử dụng ansible playbooks để triển khai stack EFK (elasticsearch, fluentd, kibana), sau đó cấu hình logging cho web service và api service, đảm bảo khi có http request gửi vào web service hoặc api service thì trong các log mà các service này sinh ra, có ít nhất 1 log có các thông tin:
 - Request Path(VD: /api1/1, /api2/3 ..)
 - HTTP Method VD: (GET PUT POST...)
 - Response Code: 302, 200, 202, 201...

Output:

- Hình ảnh chụp màn hình Kibana kết quả tìm kiếm log của các service theo **url path**

Security

Yêu cầu 1 (1đ):

- Dựng HAProxy Loadbalancer trên 1 VM riêng (trong trường hợp cụm lab riêng của sinh viên) với **mode TCP**, mở port trên LB trỏ đến NodePort của App trên K8S Cluster. (0.5)
- Sử dụng giải pháp Ingress cho các deployment, đảm bảo các truy cập đến các port App sử dụng https (0.5)
- Cho phép sinh viên sử dụng self-signed cert để làm bài

Output 1:

- File cấu hình của HAProxy Loadbalancer cho App
- File cấu hình ingress.
- Kết quả truy cập vào App từ trình duyệt thông qua giao thức https hoặc dùng curl. Ví dụ:



```
{
  "name" : "bastion",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "3C1SrJGZQwG09c0DqPjhKQ",
  "version" : {
    "number" : "8.13.4",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "da95df118650b55a500dcc181889ac35c6d8da7c",
    "build_date" : "2024-05-06T22:04:45.107454559Z",
    "build_snapshot" : false,
    "lucene_version" : "9.10.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Yêu cầu 2 (1d):

- Đảm bảo 1 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. (0.5)
- Thực hiện phân quyền cho 2 loại người dùng trên API:
 - Nếu người dùng có role là user thì truy cập vào GET request trả về code 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 code 200, còn truy cập vào POST/DELETE thì trả về 2xx

Output:

- 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 user và admin

Tham khảo cách sử dụng công cụ CURL với thông tin xác thực: <https://reqbin.com/req/c-haxm0xgr/curl-basic-auth-example>

Yêu cầu 3 (1d):

Sử dụng 1 trong số các giải pháp để ratelimit cho Endpoint của api Service, sao cho nếu có quá **10 request trong 1 phút** gửi đến Endpoint của api service thì các request sau đó bị trả về HTTP Response 409

Output:

- File tài liệu trình bày giải pháp
- File ghi lại kết quả thử nghiệm khi gọi quá 10 request trong 1 phút vào Endpoint của API Service.