

Monitoring & Logging with Kubernetes on BKE

Agenda

1. Cài đặt Loki và cấu hình thu thập log
2. Triển khai Monitoring sử dụng Prometheus

Log trong Kubernetes

Logging trên k8s thì có khác biệt so với các môi trường khác

Các vấn đề gặp phải

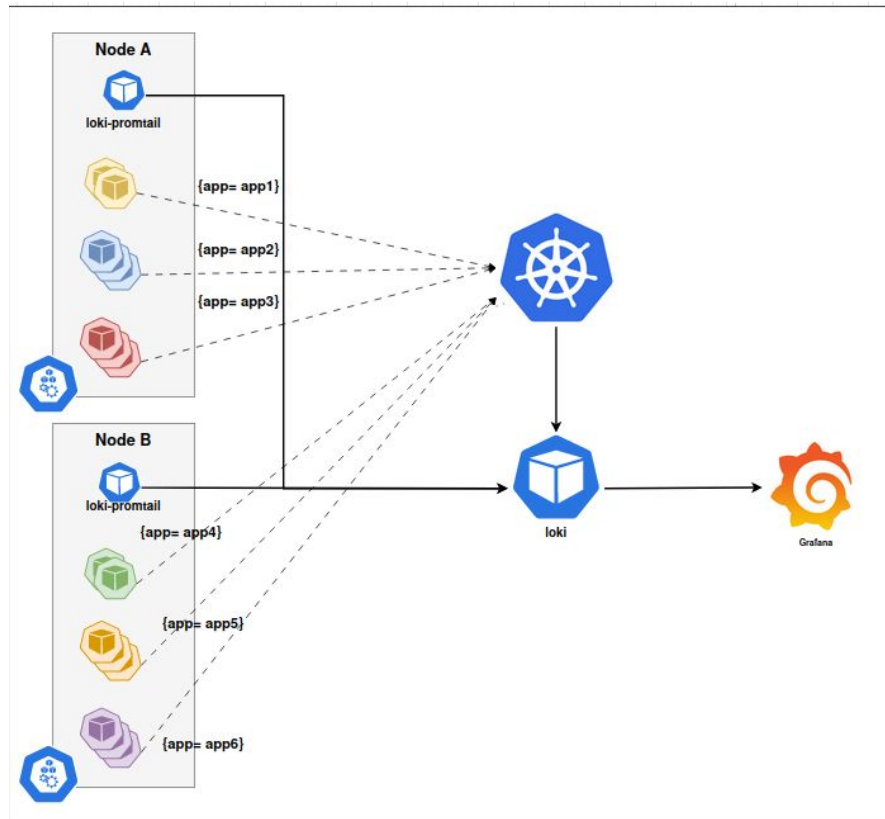
- Có nhiều replica/pod thì dùng lệnh "kubectl -n **<namespace>** logs -f **<pod-name>**" không hiệu quả
- Sau đó kết nối ssh vào node đó check log pod/container tại thư mục **/var/log/containers** rất mất công
- Pod của ứng dụng được tạo lại sẽ được tạo mới và lúc đó không thể xem lại log cũ bằng lệnh kubectl logs
- Giới hạn lưu trữ thấp, Kubernetes chỉ lưu trữ tối đa 10MB logs của container, khó để theo dõi log cũ kể cả khi không có pod tạo lại

Cài đặt Loki và cấu hình thu thập log

Loki là một hệ thống tổng hợp log được thiết kế cho Kubernetes. Có các lợi ích sau:

- **Hệ thống tổng hợp log tập trung:** Loki có thể thu thập log từ tất cả các pod, node và các thành phần khác trong cụm Kubernetes của bạn.
- **Hỗ trợ tìm kiếm mạnh mẽ:** Loki có một ngôn ngữ truy vấn mạnh mẽ có tên là LogQL. LogQL cho phép bạn lọc và truy vấn log dựa trên nhiều tiêu chí khác nhau.
- **Khả năng mở rộng và cao:** Loki được thiết kế để có thể mở rộng theo quy mô. Nó có thể xử lý khối lượng log lớn mà không bị giảm hiệu suất.
- **Tích hợp liền mạch với Grafana:** Loki tích hợp chặt chẽ với Grafana, một nền tảng giám sát phổ biến.

Cài đặt Loki và cấu hình thu thập log



Cài đặt Loki và cấu hình thu thập log

Promtail là agent, có trách nhiệm đi thu gom log và gửi về cho Loki.

Loki là thành phần chính, có trách nhiệm lưu trữ log và xử lý các câu query.

Grafana là công cụ query và hiển thị log

Cài đặt Loki và cấu hình thu thập log

Để triển khai **Loki** trên Kubernetes ta sẽ cài đặt thông quan **Helm**

Kubernetes Helm là một trình quản lý gói và công cụ quản lý ứng dụng cho Kubernetes. Nó giúp dễ dàng triển khai, cập nhật và quản lý các ứng dụng Kubernetes.

Helm hoạt động bằng cách đóng gói nhiều tài nguyên Kubernetes vào một đơn vị triển khai logic duy nhất được gọi là Chart.

Chart Helm bao gồm các tệp YAML định nghĩa các tài nguyên Kubernetes như Pod, Deployment, Service, Ingress, v.v.

Cài đặt Loki và cấu hình thu thập log

Cài đặt helm theo hướng dẫn: <https://helm.sh/docs/intro/install/>
hoặc

```
$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
```

```
$ chmod 700 get_helm.sh
```

```
$ ./get_helm.sh
```


Cài đặt Loki và cấu hình thu thập log

Cập nhật repo của loki và cài đặt loki, promtail và grafana

```
helm repo add grafana https://grafana.github.io/helm-charts
```

```
helm repo update
```

```
helm upgrade --install loki --create-namespace --namespace=loki-stack grafana/loki-stack --set grafana.enabled=true
```

Cài đặt Loki và cấu hình thu thập log

Kiểm tra việc cài đặt sử dụng lệnh: **kubectl get pod -n loki-stack**

```
quanlm@quanlm-desktop:~$ kubectl get pod -n loki-stack
```

NAME	READY	STATUS	RESTARTS	AGE
loki-0	1/1	Running	0	3m57s
loki-grafana-768c556fcb-5z7jq	2/2	Running	0	3m57s
loki-promtail-9fr9n	1/1	Running	0	3m57s
loki-promtail-bhp9z	1/1	Running	0	3m57s

Cài đặt Loki và cấu hình thu thập log

Kiểm tra các service sử dụng lệnh: **kubectl get service -n loki-stack**

```
quanlm@quanlm-desktop:~$ kubectl get service -n loki-stack
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
loki	ClusterIP	10.93.217.15	<none>	3100/TCP	6m53s
loki-grafana	ClusterIP	10.93.54.171	<none>	80/TCP	6m53s
loki-headless	ClusterIP	None	<none>	3100/TCP	6m53s
loki-memberlist	ClusterIP	None	<none>	7946/TCP	6m53s

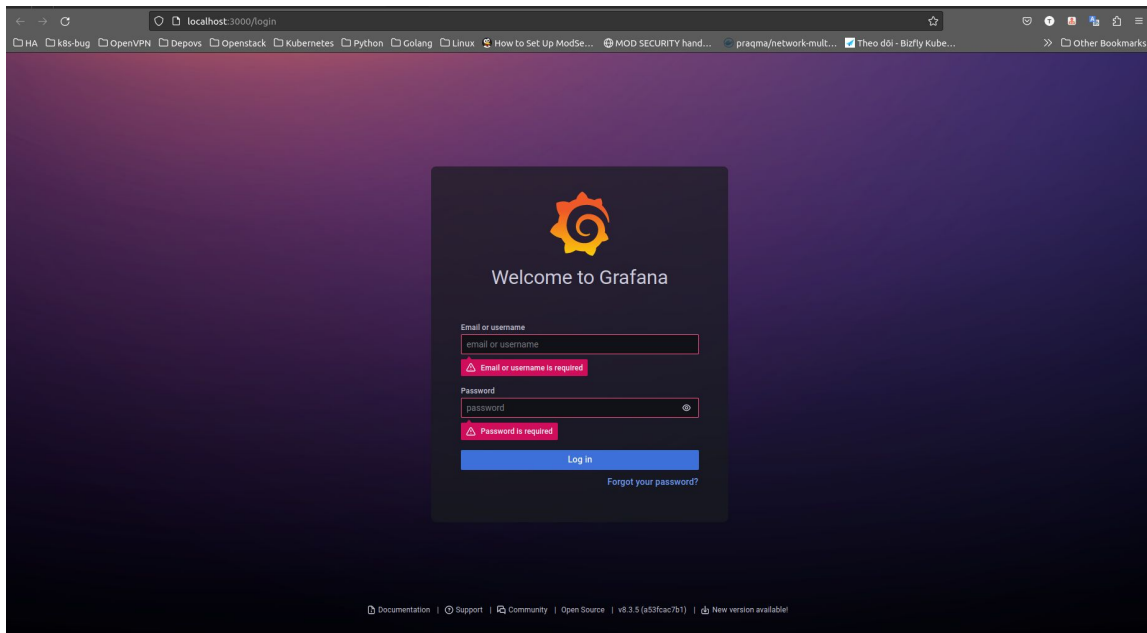
Cài đặt Loki và cấu hình thu thập log

Sử dụng port-forward để truy cập vào Grafana sử dụng lệnh:
**kubectl port-forward --namespace loki-stack
service/loki-grafana 3000:80**

```
quanlm@quanlm-desktop:~$ kubectl port-forward --namespace loki-stack service/loki-grafana 3000:80
Forwarding from 127.0.0.1:3000 -> 3000
Forwarding from [::1]:3000 -> 3000
```

Cài đặt Loki và cấu hình thu thập log

Truy cập **http://localhost:3000/login** để vào trang web grafana
và triển khai



Cài đặt Loki và cấu hình thu thập log

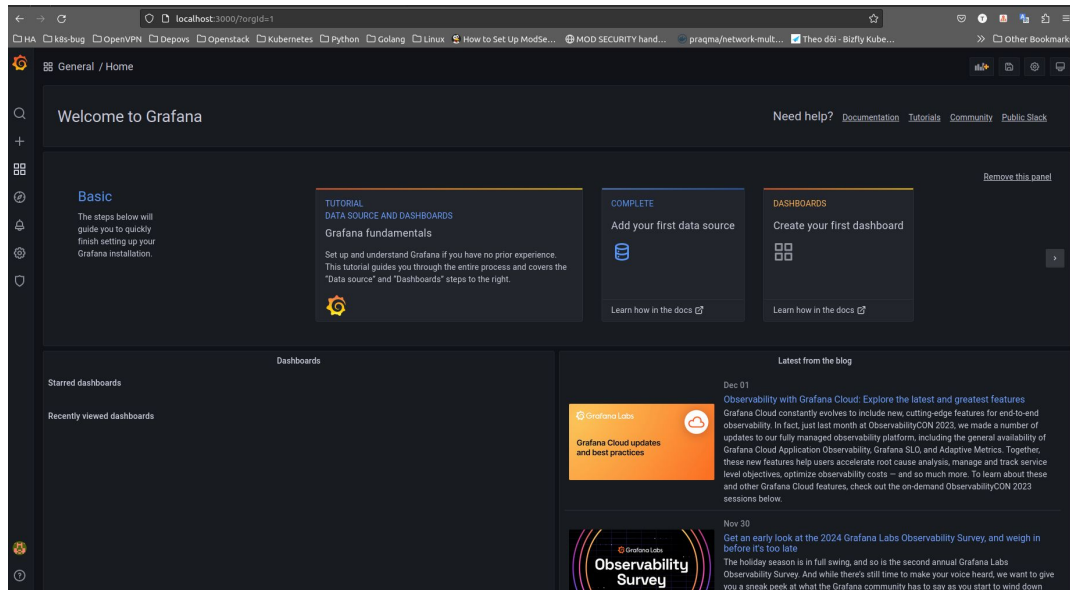
Yêu cầu tài khoản và mật khẩu, tài khoản là **admin**, còn mật khẩu sử dụng lệnh:

```
quanlm@quanlm-desktop:~$ kubectl get secret --namespace loki-stack loki-grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo  
AryRJQFat5k0ZkF16rwjAPH10afl9tnweRN3cSrD
```

để lấy, trong trường hợp này mật khẩu là:
AryRJQFat5k0ZkF16rwjAPH10afl9tnweRN3cSrD

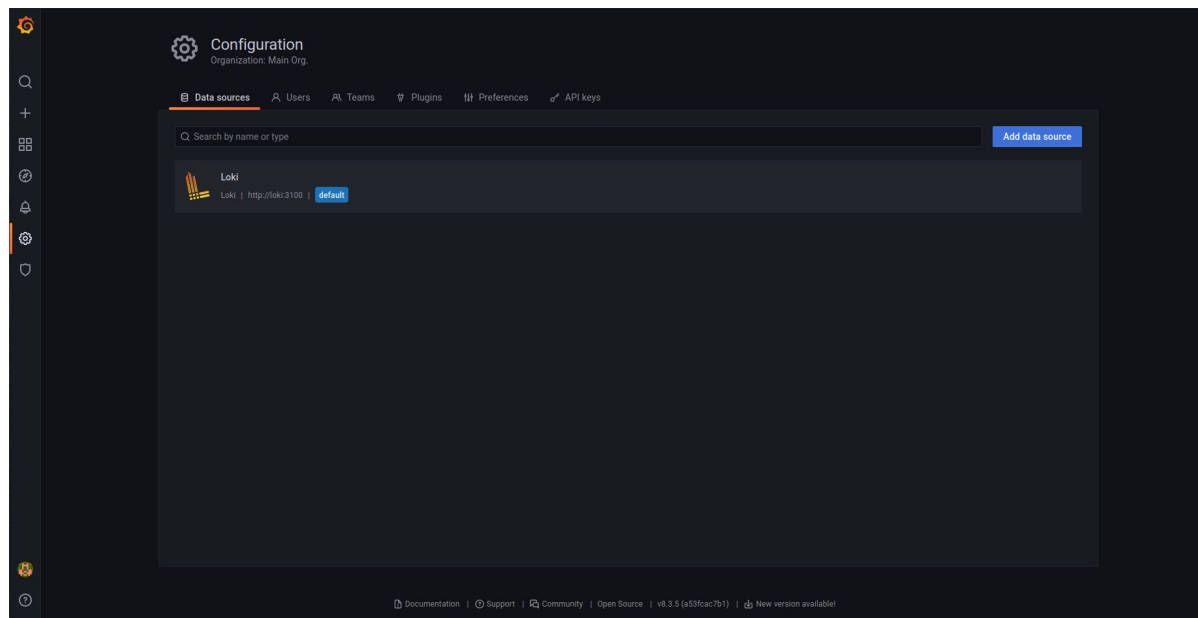
Cài đặt Loki và cấu hình thu thập log

Sau khi điền tài khoản và mật khẩu, ta vào được màn hình chính



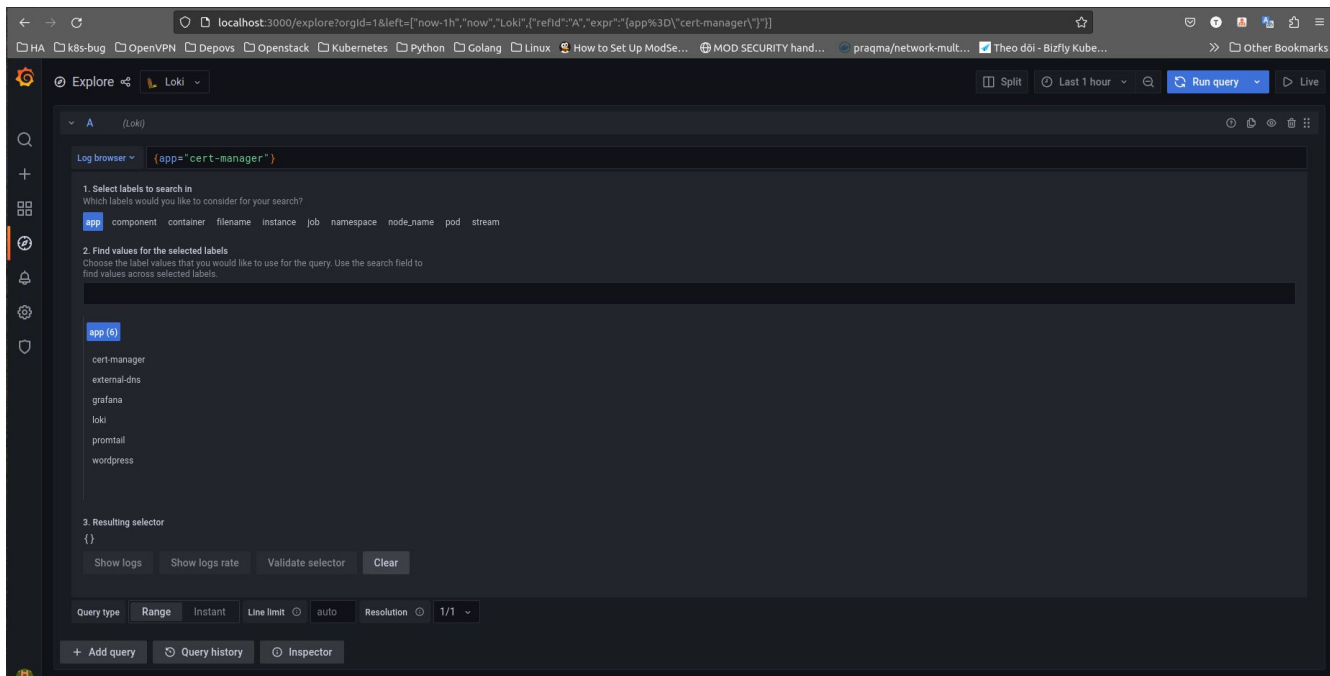
Cài đặt Loki và cấu hình thu thập log

Truy cập vào Configuration -> Datasource, ta có thể thấy Datasource Loki



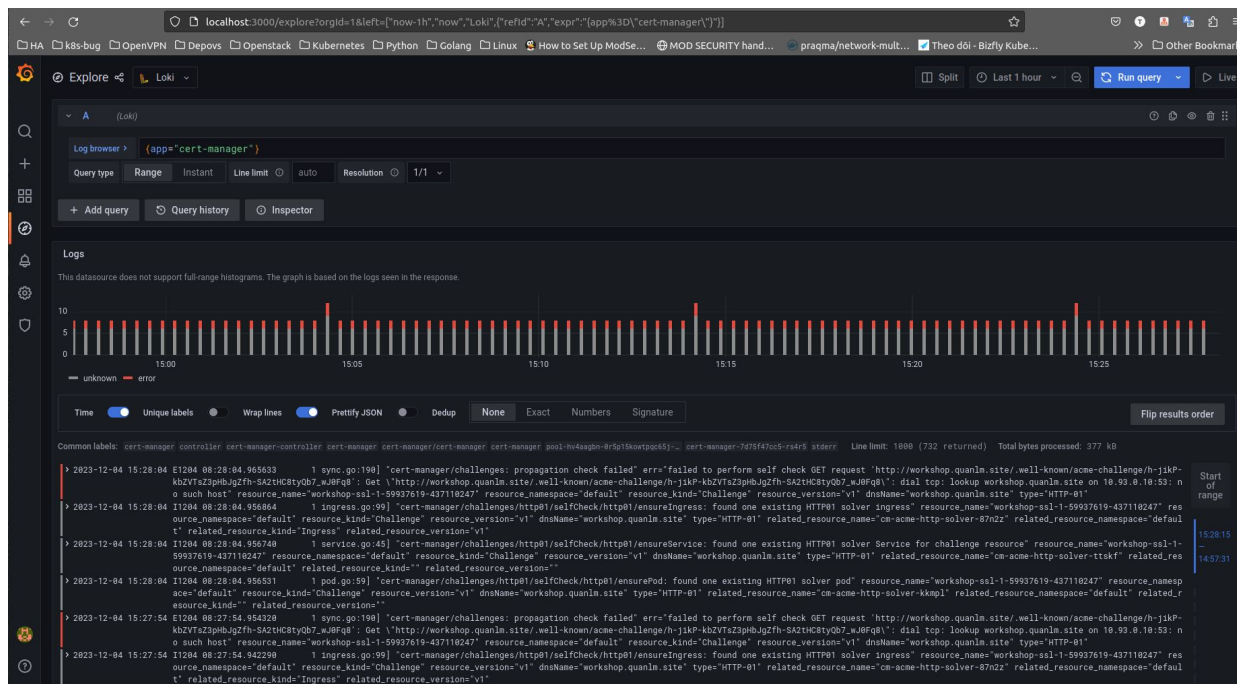
Cài đặt Loki và cấu hình thu thập log

Để xem log từ Loki, ta qua mục Explore, tạo query -> Run query



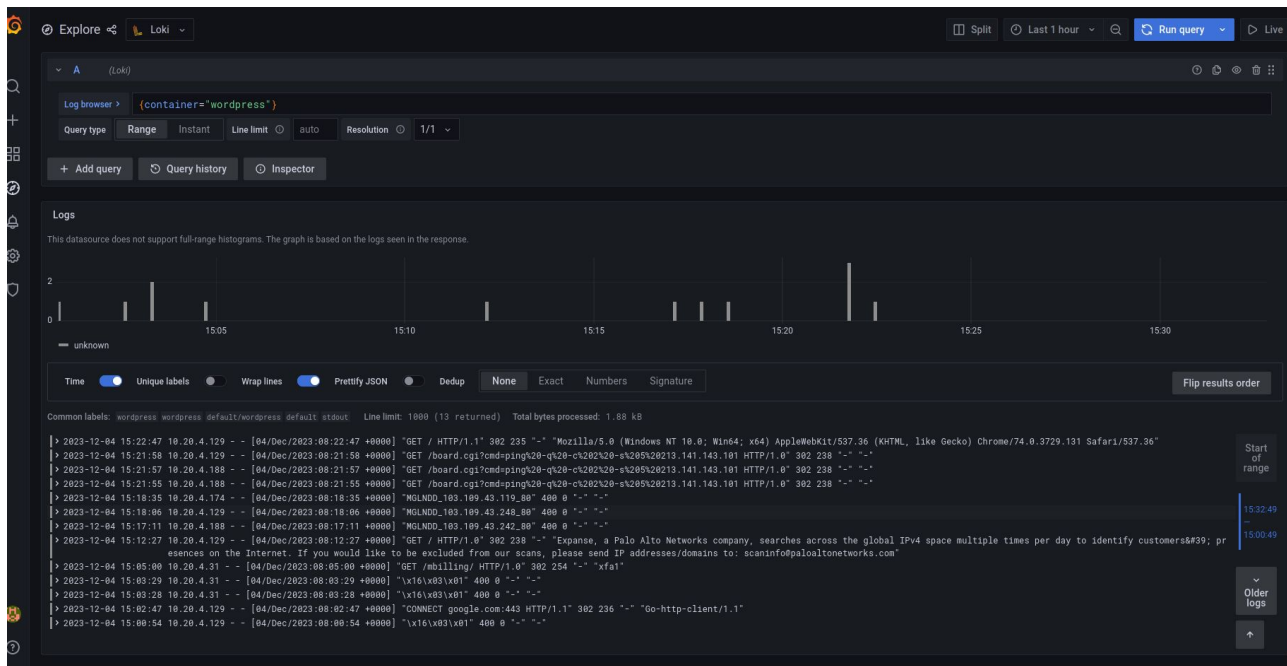
Cài đặt Loki và cấu hình thu thập log

Sau khi chạy Run query ta có thể thấy log



Cài đặt Loki và cấu hình thu thập log

tương tự ta thử với wordpress



Triển khai Monitoring

Monitoring Kubernetes là quá trình thu thập và phân tích các chỉ số từ các thành phần khác nhau của cụm Kubernetes, bao gồm:

- pod
- node
- Service
- Các cluster metrics khác ...

Triển khai Monitoring

Metrics Server: là một add-on đóng vai trò quan trọng trong việc **thu thập và cung cấp API** cho các chỉ số về tài nguyên **sử dụng** trong cụm Kubernetes.

Metrics Server để biết được các pod, container sử dụng bao nhiêu tài nguyên

Có thể sử dụng lệnh **kubectl top pod** để hiển thị tài nguyên các pod đang sử dụng (khi đã cài **metrics server**)

```
quanlm@quanlm-desktop:~$ kubectl top pod
error: Metrics API not available
quanlm@quanlm-desktop:~$ kubectl top pod
NAME                                CPU(cores)   MEMORY(bytes)
cm-acme-http-solver-kkmpl          1m           5Mi
external-dns-7964544c4d-kghjt      1m           15Mi
wordpress-6c8b9dfbdc-9f9j2         1m           89Mi
wordpress-6c8b9dfbdc-hgntk         1m           93Mi
wordpress-6c8b9dfbdc-phzzh         1m           93Mi
wordpress-mysql-0                  2m           461Mi
```

Triển khai Monitoring

Một số ví dụ cụ thể về cách sử dụng Metrics Server:

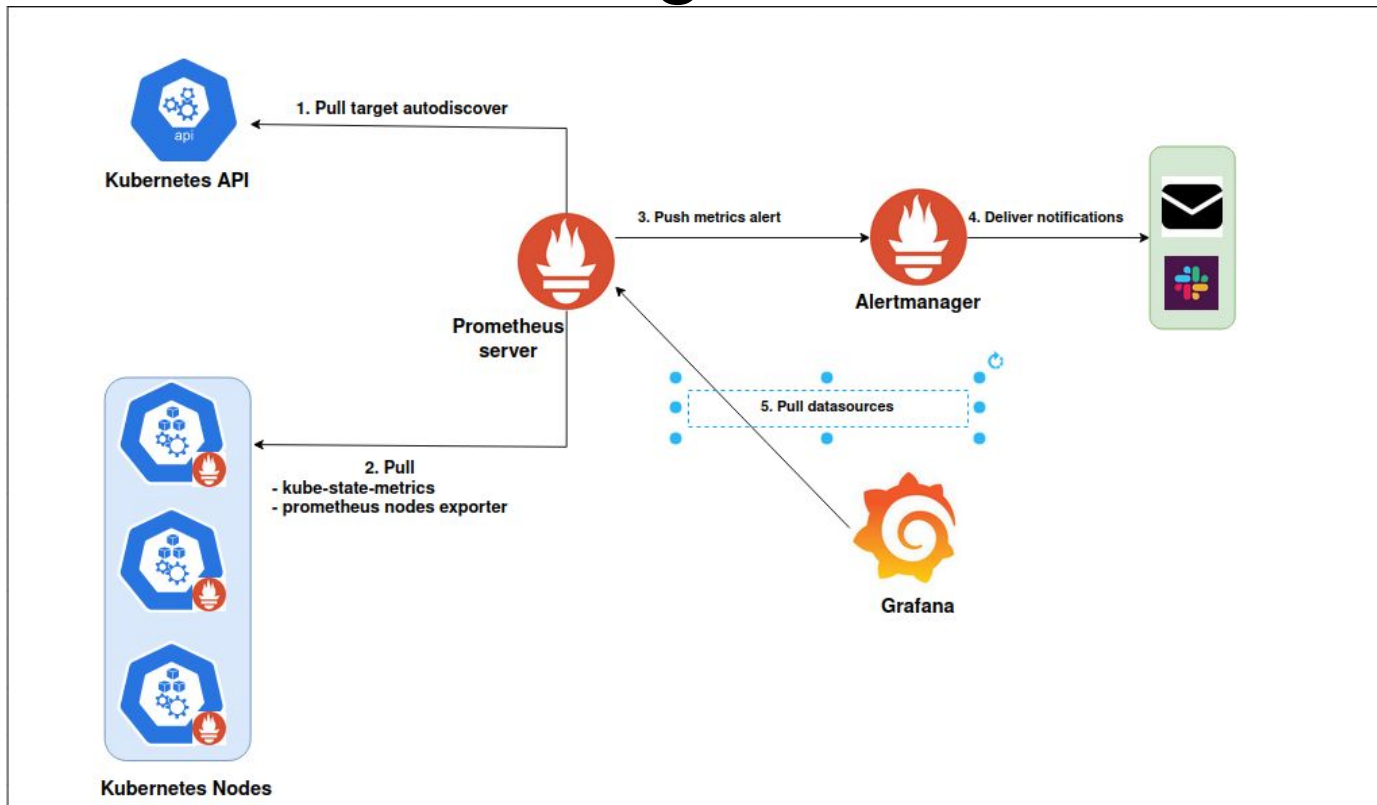
- Bạn có thể sử dụng Metrics Server để theo dõi việc sử dụng CPU và bộ nhớ của các node trong cụm.
- Thiết lập một Horizontal Pod Autoscaler (HPA) để tăng số lượng pod khi lưu lượng truy cập đến cụm tăng lên.
- Thiết lập một Vertical Pod Autoscaler (VPA) để tăng tài nguyên của pod khi số lượng truy vấn đến pod đó tăng lên.

Triển khai Monitoring

Nhược điểm

- Metrics server không lưu lại giữ liệu theo thời gian, cần công cụ bổ sung như Prometheus

Triển khai Monitoring



Triển khai Monitoring

Các thành phần của hệ thống Prometheus

- **Prometheus server:** Là thành phần chính của Prometheus, chịu trách nhiệm thu thập và lưu trữ các chỉ số.
- **Node exporter:** là một công cụ mã nguồn mở để thu thập và xuất các số liệu hệ thống từ các nút Kubernetes.
- **Alertmanager:** Là một dịch vụ chịu trách nhiệm gửi cảnh báo khi các chỉ số vượt quá ngưỡng đã đặt.
- **Grafana:** Là một công cụ trực quan hóa dữ liệu có thể được sử dụng để hiển thị các chỉ số của Prometheus.

Triển khai Monitoring

Cài đặt prometheus sử dụng helm

Add repo:

```
helm repo add prometheus-community
```

```
https://prometheus-community.github.io/helm-charts
```

```
helm repo update
```

Triển khai Monitoring

Cài đặt prometheus sử dụng helm

Tạo thư mục values.yaml với giá trị như hình để cấu hình khi sử dụng helm sẽ không tạo ra grafana

```
grafana:  
  enabled: false
```

Triển khai Monitoring

Cài đặt bằng helm sử dụng lệnh:

```
helm install prometheus-stack --namespace  
prometheus-stack --create-namespace  
prometheus-community/kube-prometheus-stack -f  
values.yaml
```

Triển khai Monitoring

Kiểm tra kết quả cài đặt sử dụng lệnh:

kubectl get pod -n prometheus-stack

kubectl get svc -n prometheus-stack

```
quanlm@quanlm-desktop:~$ kubectl get pod -n prometheus-stack
```

NAME	READY	STATUS	RESTARTS	AGE
alertmanager-prometheus-stack-kube-prom-alertmanager-0	2/2	Running	0	19m
prometheus-prometheus-stack-kube-prom-prometheus-0	2/2	Running	0	19m
prometheus-stack-kube-prom-operator-6fbdd5fbdd-2phx2	1/1	Running	0	19m
prometheus-stack-kube-state-metrics-56f9ddf4b7-mgffs	1/1	Running	0	19m
prometheus-stack-prometheus-node-exporter-bhz29	1/1	Running	0	19m
prometheus-stack-prometheus-node-exporter-dm4rh	1/1	Running	0	19m

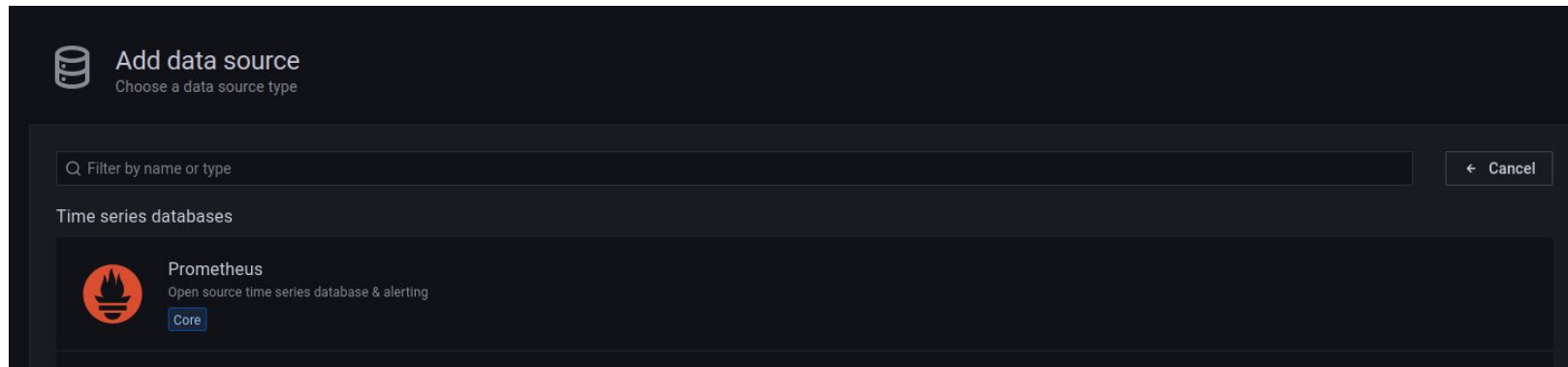
```
quanlm@quanlm-desktop:~$ kubectl get svc -n prometheus-stack
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
alertmanager-operated	ClusterIP	None	<none>	9093/TCP,9094/TCP,9094/UDP	19m
prometheus-operated	ClusterIP	None	<none>	9090/TCP	19m
prometheus-stack-kube-prom-alertmanager	ClusterIP	10.93.56.3	<none>	9093/TCP,8080/TCP	19m
prometheus-stack-kube-prom-operator	ClusterIP	10.93.110.200	<none>	443/TCP	19m
prometheus-stack-kube-prom-prometheus	ClusterIP	10.93.196.86	<none>	9090/TCP,8080/TCP	19m
prometheus-stack-kube-state-metrics	ClusterIP	10.93.188.73	<none>	8080/TCP	19m
prometheus-stack-prometheus-node-exporter	ClusterIP	10.93.124.59	<none>	9100/TCP	19m

Triển khai Monitoring

Cấu hình grafana để sử dụng pull metrics từ prometheus

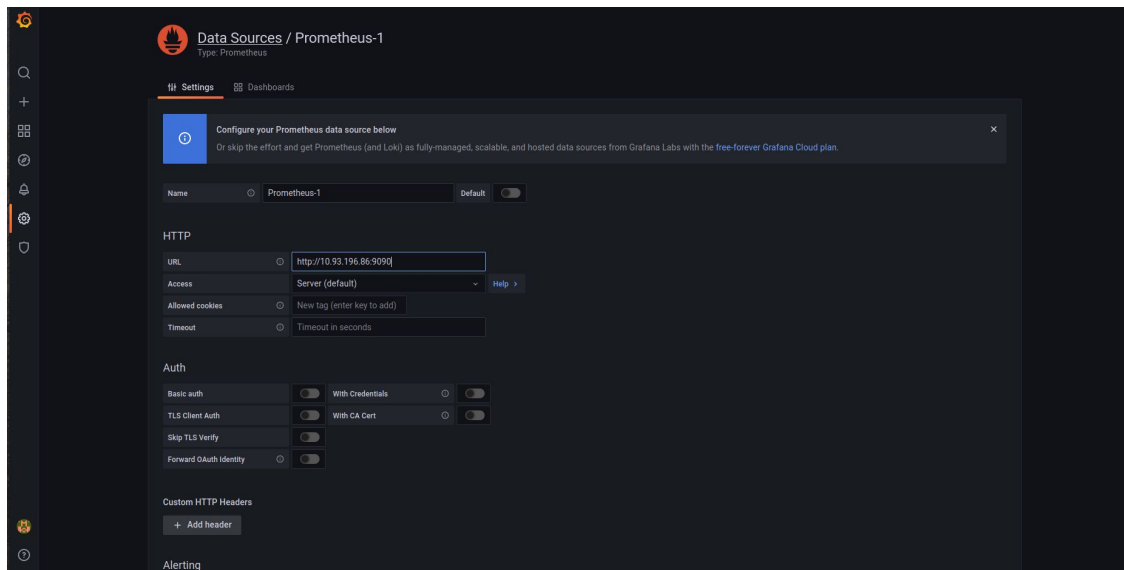
Tương tự như Loki ta sẽ vào datasources -> add datasource -> Prometheus



Triển khai Monitoring

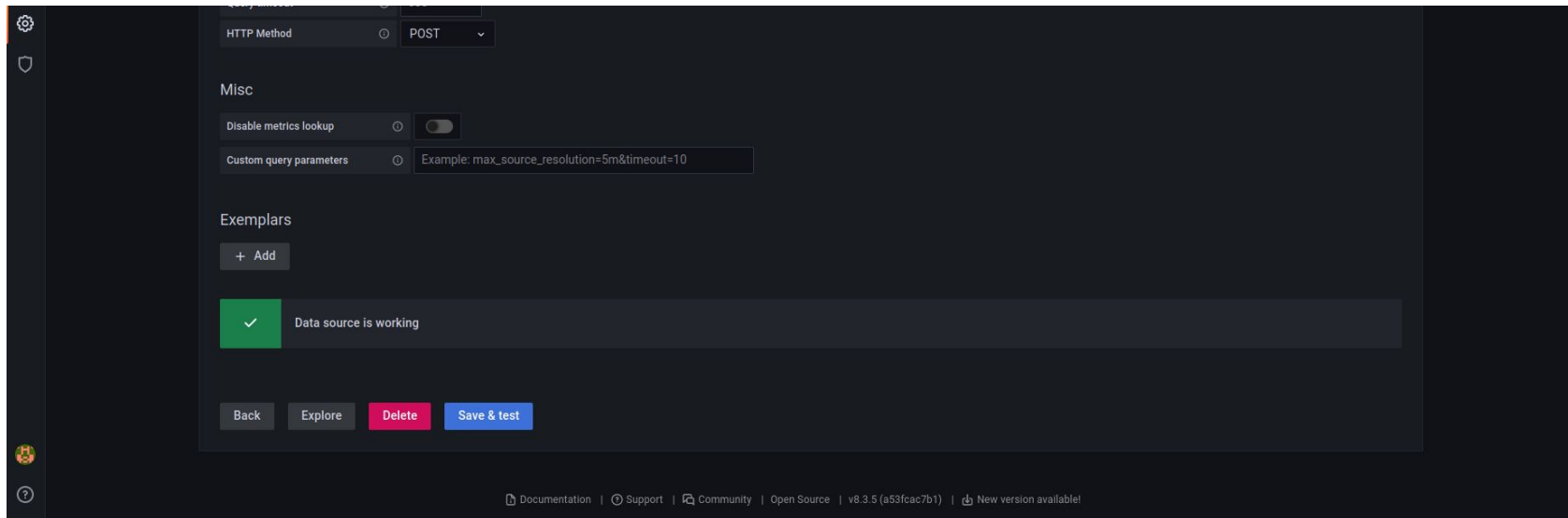
Ta sẽ lấy ClusterIP của service:

prometheus-stack-kube-prom-prometheus (trong lệnh **kubectl get svc** ở trên), ta sẽ điền url là: **http://10.93.196.86:9090**



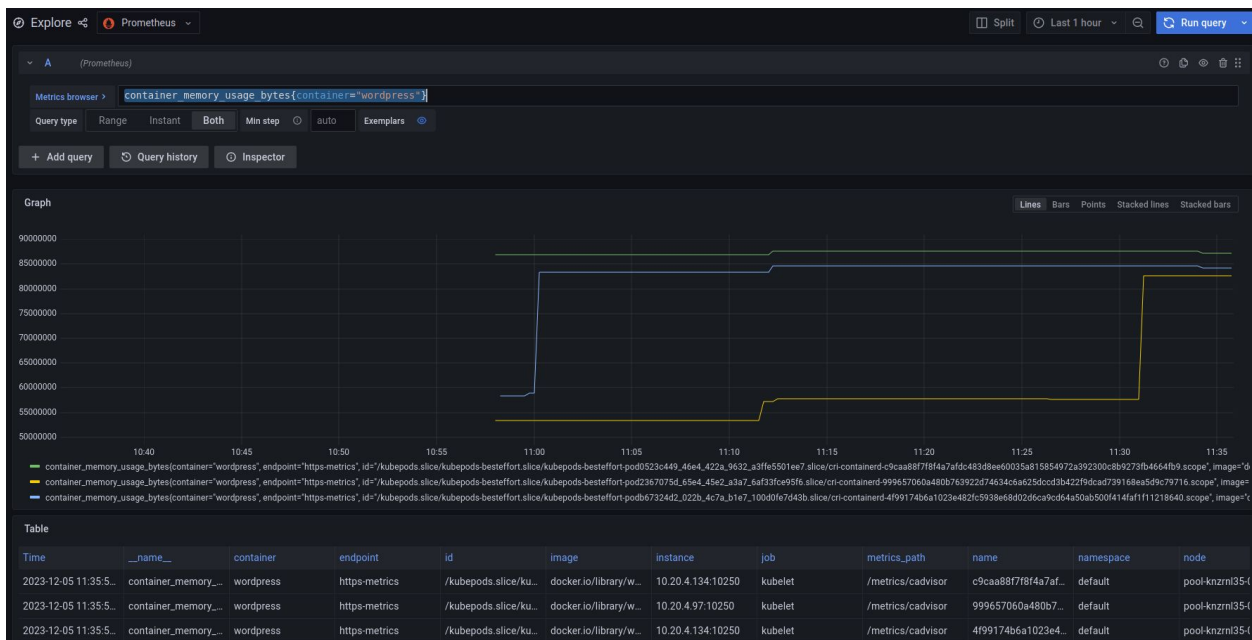
Triển khai Monitoring

Kéo xuống chọn save & test, sẽ hiện lên thông báo data sources đang hoạt động



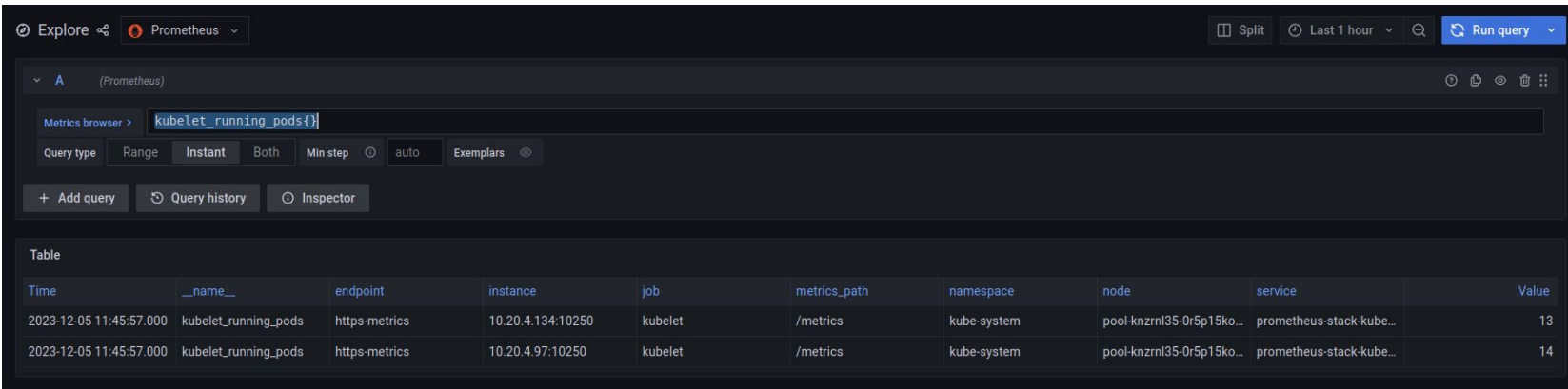
Triển khai Monitoring

Sử dụng Explore để query metrics, ta query lượng ram sử dụng của container wordpress: `container_memory_usage_bytes{container="wordpress"}`



Triển khai Monitoring

Sử dụng Explore để query metrics, ta query lượng pod trong các node: `kubelet_running_pods{}`



The screenshot shows the Prometheus Explore interface. At the top, there's a header with "Explore", "Prometheus", and a "Run query" button. Below the header, the query "kubelet_running_pods{}" is entered in the "Metrics browser" field. The "Query type" is set to "Range", and the "Instant" tab is selected. Below the query field, there are buttons for "Add query", "Query history", and "Inspector". The results are displayed in a table with the following columns: Time, __name__, endpoint, instance, job, metrics_path, namespace, node, service, and Value.

Time	__name__	endpoint	instance	job	metrics_path	namespace	node	service	Value
2023-12-05 11:45:57.000	kubelet_running_pods	https-metrics	10.20.4.134:10250	kubelet	/metrics	kube-system	pool-knznrl35-0r5p15ko...	prometheus-stack-kube...	13
2023-12-05 11:45:57.000	kubelet_running_pods	https-metrics	10.20.4.97:10250	kubelet	/metrics	kube-system	pool-knznrl35-0r5p15ko...	prometheus-stack-kube...	14

Thank you !

Q & A

More information

Bizfly Kubernetes Engine

<https://bizflycloud.vn/kubernetes-engine>

Container Registry

<https://bizflycloud.vn/container-registry>



Bizfly^{CLOUD} ❤️
kubernetes