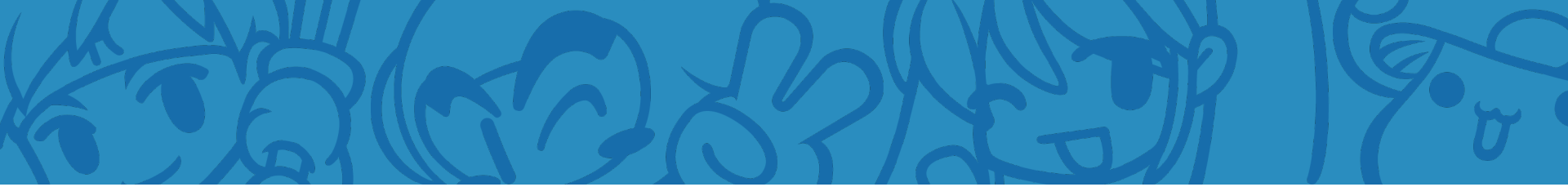
Rancher 도입기

김상혁 / 우병열 (Infrastructure Engineer)

2020. 11. 19

CONTENTS

1. 왜 Rancher인가요?
2. Rancher 도입 시 이슈들
3. Rancher를 도입을 하고 기대효과는?
4. 앞으로의 과제?



왜 Rancher인가요?

K8s를 어떻게 제공할 것인가?

- 퍼블릭 클라우드를 이용하여 제공하는 방법
  - ✓ 손쉬운 방법이지만 인프라 비용이 높음
  - ✓ Vendor Lock-in을 피하기 어려움
  - ✓ 해외 서비스가 용이함

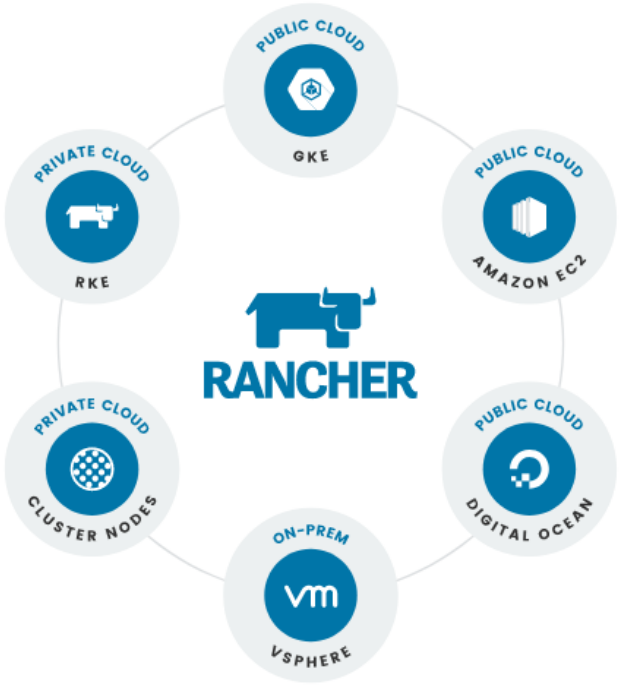
- On-Premise 인프라를 이용하여 제공하는 방법
  - ✓ 인프라 비용은 낮음
  - ✓ Vendor Lock-in을 피할 수 있지만 구성/유지가 어려움
  - ✓ 해외 서비스가 어려움

어떠한 형태로 제공할 것인가

- Vanilla kubernetes 구축하는 방법
  - ✓ 구축 및 운영이 어려움
  - ✓ 모니터링, 구성 등 통일된 관리가 어려움
  - ✓ 여러 상황에 유동적으로 대처 가능

- Kubernetes Management Platform을 이용하여 제공하는 방법
  - ✓ 구축 및 운영이 용이함
  - ✓ 별도 라이선스 비용이 필요
  - ✓ 다양한 기능으로 생산성 향상

왜 Rancher인가?



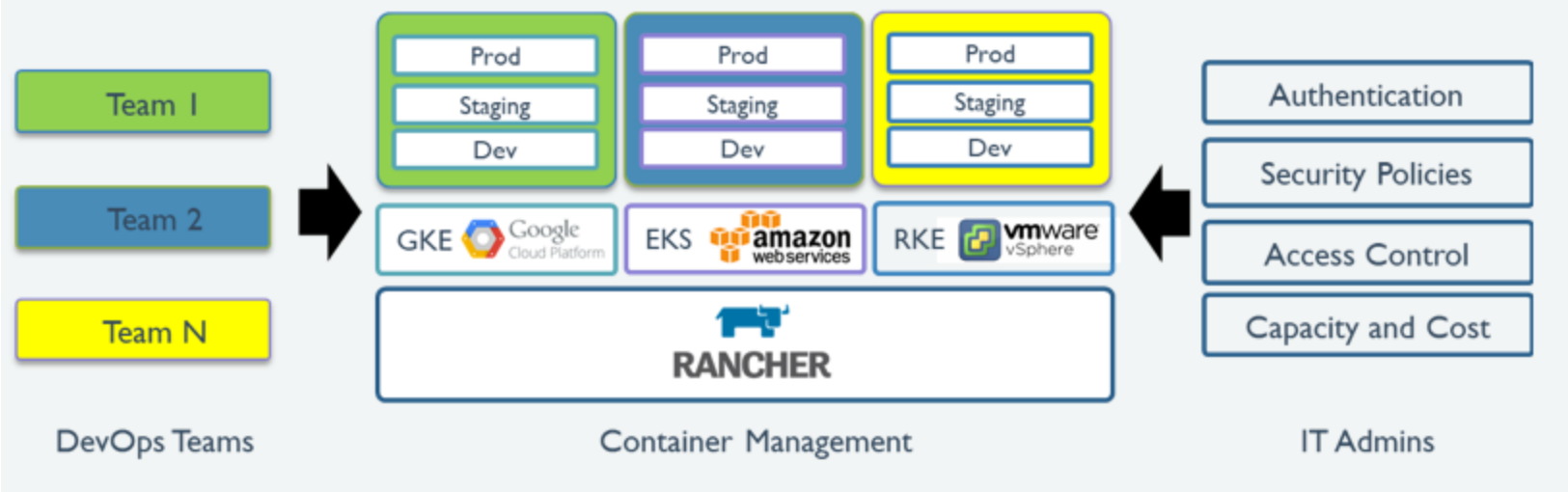
https://rancher.com/why-rancher/

- 다양한 On-Premise 지원
- 다양한 Public Cloud 지원
- 다양한 기능 및 통합 관리 제공
  - ✓ 통합 멀티 클러스터 관리
  - ✓ 앱 카탈로그
  - ✓ 훌륭한 UI

왜 Rancher인가?

- Open Source 입니다
  - ✓ Docker EE나 Openshift 처럼 사용에 제약이 없음
- 다양한 환경을 지원합니다
  - ✓ PaaS 기반 k8s, IaaS 기반 k8s, On-Premise BM/VM
  - ✓ 기존에 구성된 k8s 클러스터까지
- Vendor-Lockin 회피
  - ✓ 나갈 때도 마음대로지만 들어올 때도 마음대로입니다
- 가볍고 직관적인 UI
  - ✓ 진입 장벽을 낮춰줍니다

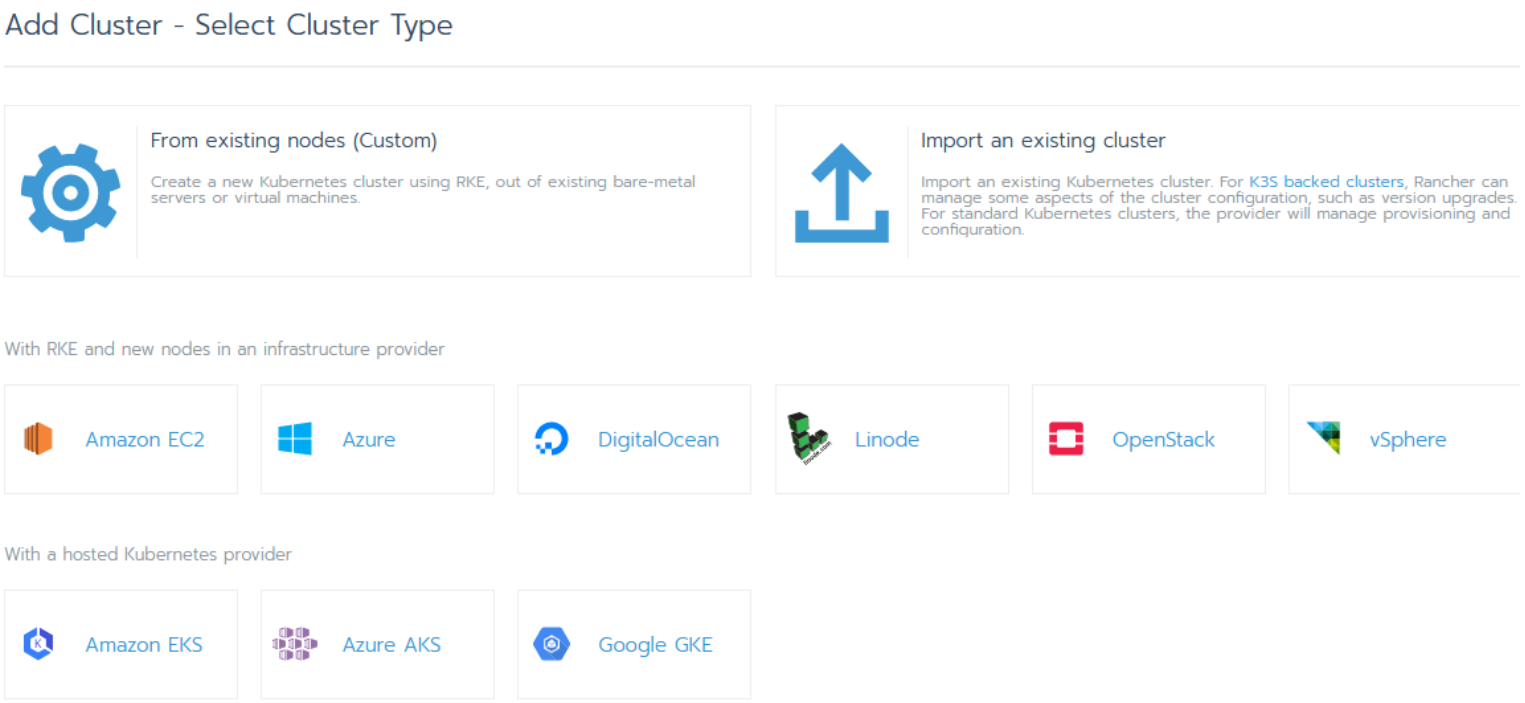
왜 Rancher인가?



https://rancher.com/docs/rancher/v2.x/en/overview/

- 역할별 관리
  - ✓ 개발자는 주어진 환경에서 개발에만 전념
  - ✓ 운영자는 컨테이너 관리에만 전념
  - ✓ 관리자는 인증, 접근제어, 인프라관리에만 전념

지원되는 인프라



- 대부분의 Major cloud provider를 지원
- On-premise Baremetal, VM 등 다양한 형태 지원
- Windows worker node 지원 (도커 EE 필요)

Rancher 클러스터 유형

ACTION	RANCHER LAUNCHED KUBERNETES CLUSTERS	HOSTED KUBERNETES CLUSTERS	IMPORTED CLUSTERS
Using kubectl and a kubeconfig file to Access a Cluster	✓	✓	✓
Managing Cluster Members	✓	✓	✓
Editing and Upgrading Clusters	✓	✓	*
Managing Nodes	✓	✓	✓
Managing Persistent Volumes and Storage Classes	✓	✓	✓
Managing Projects, Namespaces and Workloads	✓	✓	✓
Using App Catalogs	✓	✓	✓
Configuring Tools (Alerts, Notifiers, Logging, Monitoring, Istio)	✓	✓	✓
Cloning Clusters	✓	✓	
Ability to rotate certificates	✓		
Ability to back up your Kubernetes Clusters	✓		
Ability to recover and restore etcd	✓		
Cleaning Kubernetes components when clusters are no longer reachable from Rancher	✓		
Configuring Pod Security Policies	✓		
Running Security Scans	✓		

https://rancher.com/docs/rancher/v2.x/en/cluster-admin/#managing-clusters-in-rancher

- Rancher: RKE로 만든 k8s
- Hosted: EKS, AKS, GKE 등
- Imported: Vanilla k8s 등

Magnum을 고려해봤습니다만

- Magnum은 다음과 같은 단점이 있었습니다
  - ✓ 컨테이너 구동용 OS 이미지에 의존 (Fedora Atomic, CoreOS 등)
  - ✓ 오픈스택 환경에서만 구동 가능
  - ✓ 사용자 편의 기능이 많지 않음
  - ✓ 사용자가 직접 클러스터를 생성해야 함
  - ✓ 기존 k8s 관리 불가

Rancher와 Magnum 비교

항목	Rancher	Magnum
UI 편의성	• 별도 웹UI 제공 • API, CLI 제공 • kubectl 웹UI 제공	• Horizon 내에서 제공 • API, CLI 제공
K8s 업그레이드 지원	지원	미지원
라이선스	Apache 2.0	Apache 2.0
Cloud Provider	AWS, Azure, GCP, Openstack 등 지원	Openstack만 지원
모니터링 및 로깅 구성	모니터링/로깅 구성 지원	모니터링 구성 지원
App Catalog 지원	지원	미지원
제공 리소스	Cluster, Namespace, Project	Cluster
외부 k8s 관리	가능	불가능

미리 보기

전체 테이블

- Rancher 도입 시 이슈
- Rancher 를 도입 후 기대 효과는?
- 앞으로 할일?

도입 시 이슈들

- 어떤 성능 테스트 툴을 사용하지? : perf-tests 사용
- Grafana dashboard에서 중요한 지표가 표시가 안 된다 : Grafana Latency (-> Duration)
- 스토리지 배포 시 PV은 어떻게 제공을 하지? : CSI

Rancher를 도입 후 기대 효과?

- Enterprise-grade 서비스 제공(Logging, Monitoring, Audit)
- Public Cloud(EKS, GKE, AKS) 및 Private Cloud 둘 다 지원

앞으로 할일?

- Authentication Plugins 개발
- Windows node 테스트(for IIS)

대상 청자

- 2개 이상의 k8s cluster를 효율적으로 관리하고자 하는 분
- Enterprise-grade feature 서비스를 제공 하고자 하는 분

Rancher 도입 시 이슈들

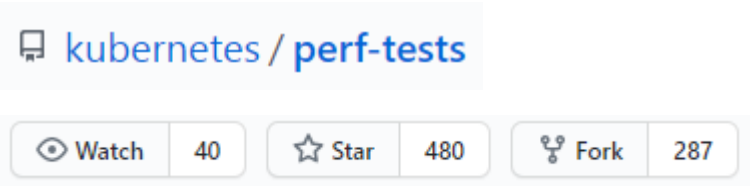
1. perf-tests

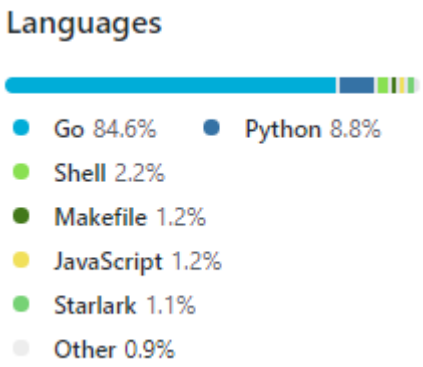
1. 성능 테스트는 왜 필요한가?

- 다양한 Cluster를 구성 후 각 클러스터마다 제공할 수 있는 SLOs정의할 수 있음
- 서비스 시나리오에 맞게 system design을 할 수 있음
- 위의 system design을 이용해서 high availability, scalability, stability를 제공

1. perf-tests

2. perf-tests 소개





- Kubernetes 클러스터의 성능테스트 툴을 모은 Github입니다. Start 480, Fork 287 으로 인기좋음
- 이 툴은 coredns, cluster load test, network bandwidth 등을 테스트를 할 수 있음
- 대부분의 툴은 golang으로 만들었음

1. perf-tests

3. DNS

개요

- DNS은 application 의 response time에 매우 중요한 요소

고려한 점

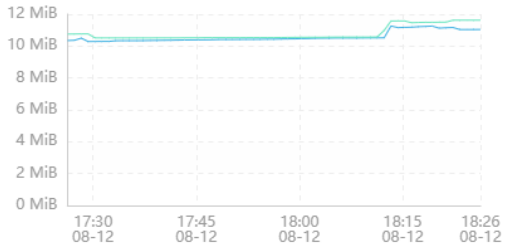
- 리소스 제한이 없는 경우 최대 QPS?
- 리소스 제한을 하는 경우 성능 측정
- SLO를 정의를 했는가?

1. perf-tests

4. DNS – Resource 사용량

CPU Utilization

Memory Utilization



Network Packets

Network I/O

- CPU는 vCPU 3개 사용
- Memory Utilization 10MiB 사용
- Network I/O 5Mbps 사용

1. perf-tests

5. DNS - 결과

run_id	queries_sent	queries_com	queries_lost	run_time	qps	avg_latency	min_latency	max_latency
1597223492	5387733	5381627	6106	600.008928	8969.244871	0.005378	0.000116	4.007899
1597223492	5278263	5272081	6182	600.001267	8786.783112	0.00541	0.00011	4.007694
1597223492	16236555	16235839	716	600.003265	27059.58442	0.003394	0.000103	4.016823
1597223492	5536620	5530436	6184	600.005081	9217.315278	0.005256	0.00011	4.007432

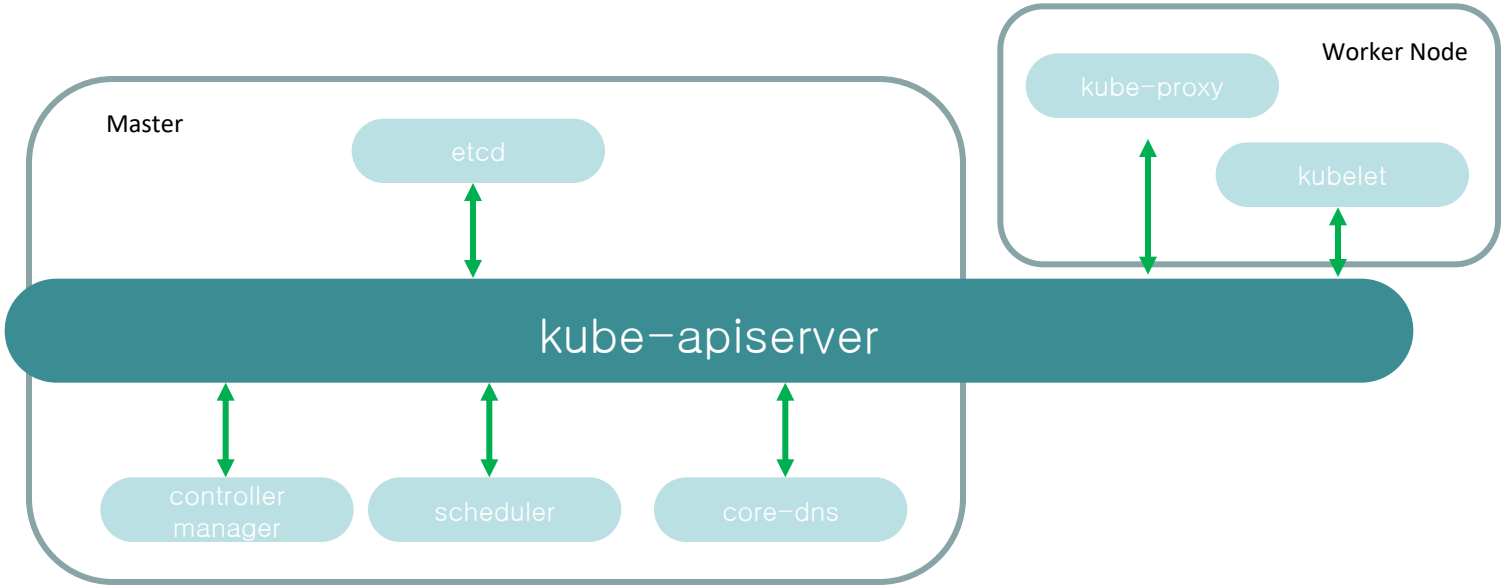
- 10분 동안 총 쿼리는 32,439,171
- 최소 응답 시간 0.1 ms, 최대 응답 시간 4s

도출된 결론

- 2만 이상 QPS를 넘어서면 SLO를 달성할 수 없음 (총쿼리 / (테스트시간(600) + instance(2)))
- 특정 리소스 사용량이 넘어서면 coredns autoscaler가 필요

1. perf-tests

6. API Server



- kube apiserver는 Cluster의 모든 Orchestration 작업에 입구
- 위의 이유로 성능에 매우 중요함

Source : https://478h5m1yrfsa3bbe262u7muv-wpengine.netdna-ssl.com/wp-content/uploads/2019/12/control_plane_api.png

1. perf-tests

7. API Server – 테스트 환경 및 방법

테스트 환경

- Control Plane과 Worker Plane을 구분해서 테스트 진행
- 위의 이유는 Application workload가 api server 성능 테스트에 영향을 주기 않기 위해

항목	설명	제한사항
Control Plane	NODE : 3 EA vCPU : 8 EA RAM : 16 GB	Network:750Mbps DISK IOPS : 500
Worker Plane	NODE : 5 EA vCPU : 8 EA RAM : 16 GB	Network:750Mbps DISK IOPS : 500

테스트 방법

- 이미 잘 정의되어 있는 테스트 시나리오 사용
- api server workload를 임의로 발생 시키기 쉽게 access token validation을 이용

항목	설명
테스트 툴	perf-tests clusterloader2
측정 방법	access token validation
Parameter	Namespace : 1 EA Service Account : 80 EA Token : 25 EA QPS per Worker : 6 Kubernetes Worker Node : 5 EA

1. perf-tests

8. API Server – SLO

Status	SLI	SLO
Official	Latency[1] of mutating[2] API calls for single objects for every (resource, verb) pair, measured as 99th percentile over last 5 minutes	In default Kubernetes installation, for every (resource, verb) pair, excluding virtual and aggregated resources and Custom Resource Definitions, 99th percentile per cluster-day <= 1s
Official	Latency[1] of non-streaming read-only[3] API calls for every (resource, scope[4]) pair, measured as 99th percentile over last 5 minutes	In default Kubernetes installation, for every (resource, scope) pair, excluding virtual and aggregated resources and Custom Resource Definitions, 99th percentile per cluster-day (a) <= 1s if `scope=resource` (b) <= 5s if `scope=namespace` (c) <= 30s if `scope=cluster`

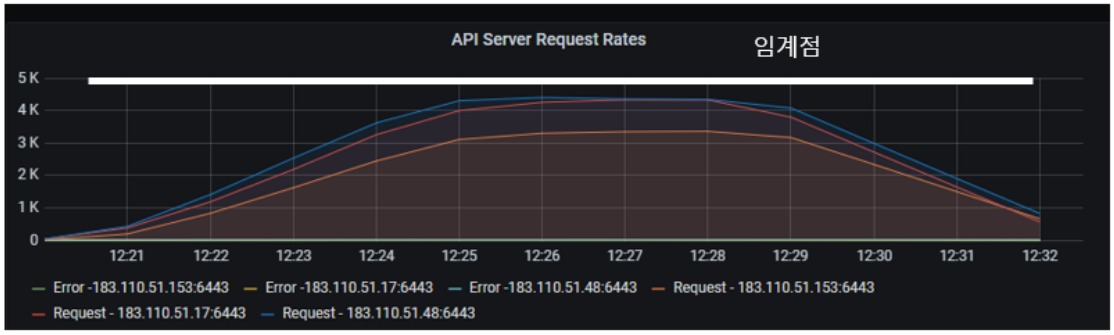
- resource <= 1s
- namespace <= 5s
- cluster <= 30s

source : https://github.com/kubernetes/community/blob/master/sig-scalability/slos/api_call_latency.md

1. perf-tests

9. API Server – 결과 1

그래프



테이블

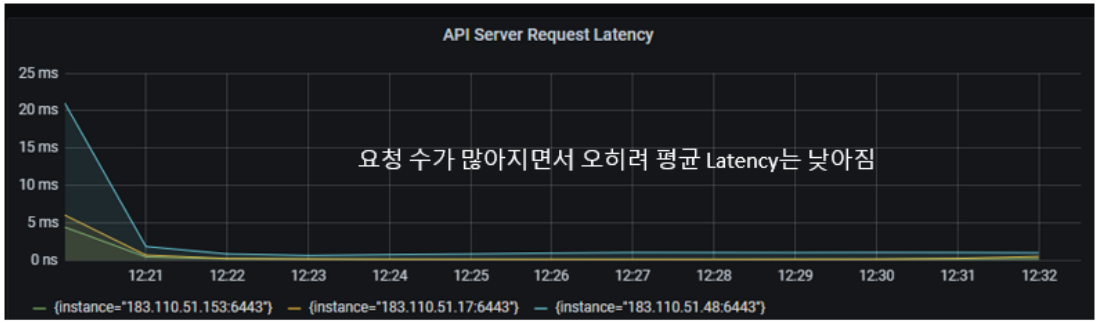
Time	ctrl-01	ctrl-02	ctrl-03
0	13	37	35
60	193	376	426
120	829	1184	1407
180	1617	2182	2531
240	2442	3253	3616
300	3105	3995	4301
360	3299	4255	4401
420	3347	4328	4356
480	3358	4328	4343
540	3167	3795	4077
600	2337	2719	2987
660	1499	1640	1901
720	659	560	824

- 최대 TPS 4k 입니다.
- 다른 테스트에서 TPS가 5k가 이상이면 api server 다운이 발생
- coredns 를 이용해서 round-robin으로 3개 노드가 고르게 workload를 분담

1. perf-tests

9. API Server – 결과 2

그래프



- 그래프를 보면 초반 60초는 test containers 요청
- 최대 처리 지연 시간 0.02(s) 단, Instance dimension
- 테스트 소요 시간 720(s)
- Rancher 기본 dashboard

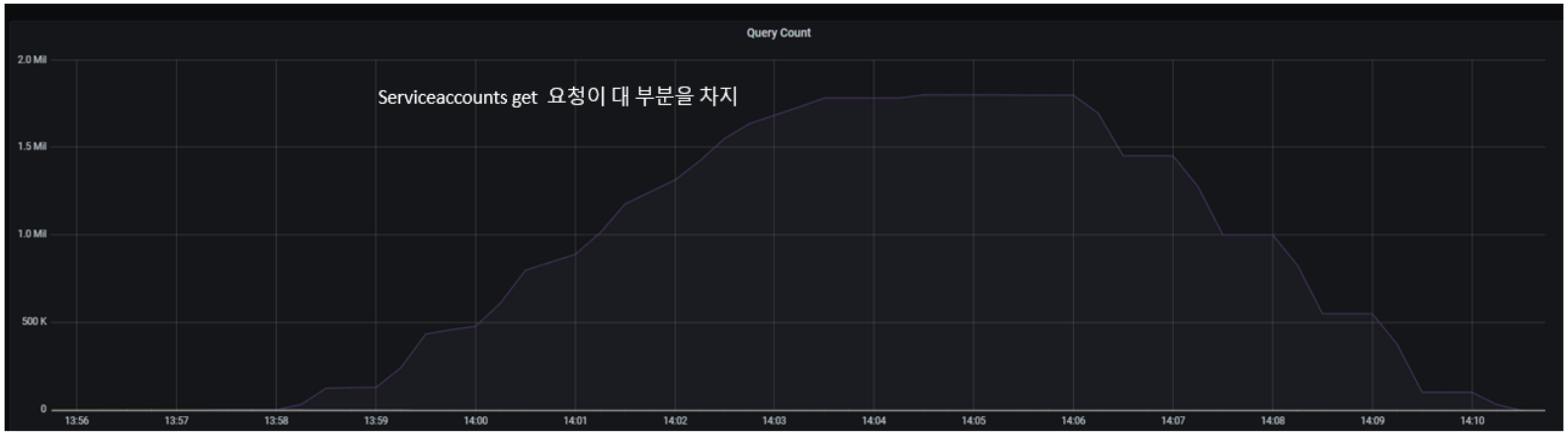
테이블

Time	ctrl-01	ctrl-02	ctrl-03
0	0.0044	0.0060	0.0210
60	0.0004	0.0006	0.0018
120	0.0002	0.0002	0.0008
180	0.0001	0.0001	0.0006
240	0.0001	0.0001	0.0007
300	0.0001	0.0001	0.0008
360	0.0001	0.0001	0.0009
420	0.0001	0.0001	0.0010
480	0.0001	0.0001	0.0010
540	0.0001	0.0001	0.0010
600	0.0001	0.0001	0.0010
660	0.0001	0.0002	0.0010
720	0.0002	0.0005	0.0010

1. perf-tests

9. API Server – 결과 3

그래프



- sliding windows 5m 기준 발생한 query 수에 피크는 2m

Formula

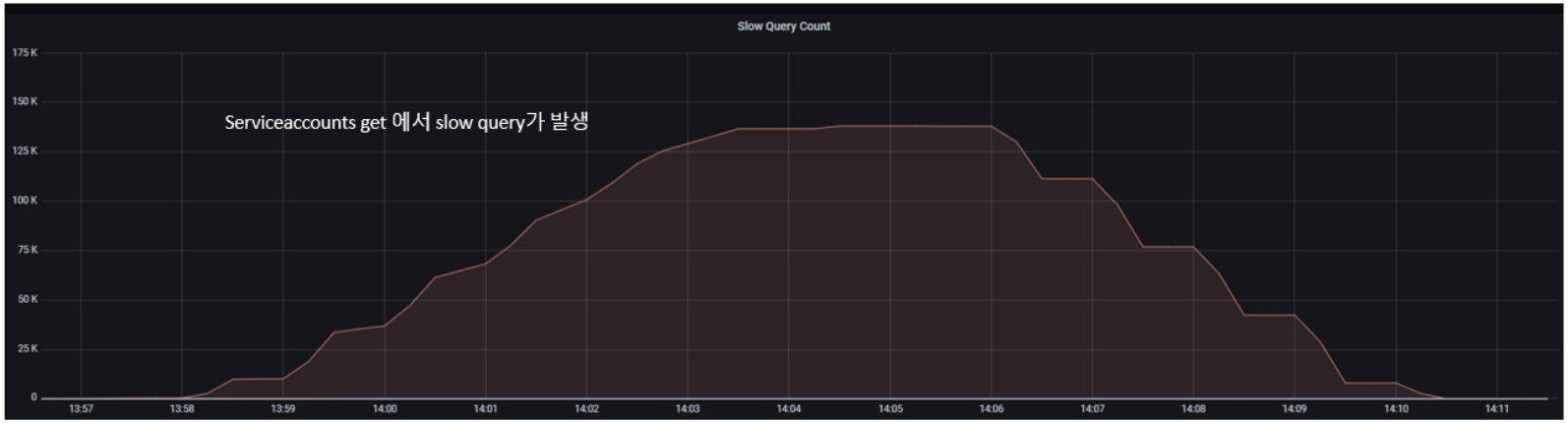
sum(increase(apiserver_request_duration_seconds_count{verb!~"WATCH|WATCHLIST|PROXY|CONNECT"}[5m])) by (resource, subresource, scope, verb)

- sum
- increase

1. perf-tests

9. API Server – 결과 4

그래프



- 슬로우 쿼리 전체(serviceaccount, namespace, get) 138k
- 요청 수(serviceaccount, namespace, get) 1.8m

138k / 1.8m = 0.076 -> 7% request는 1초 이상의 지연이 발생됨

Formula

sum(rate(apiserver_request_duration_seconds_bucket{verb!~"WATCH|WATCHLIST|PROXY|CONNECT", le!~"0.₩₩d+|1"}[5m])) by (resource, subresource, scope, verb)

- sum
- rate

2. Grafana API Server Latency

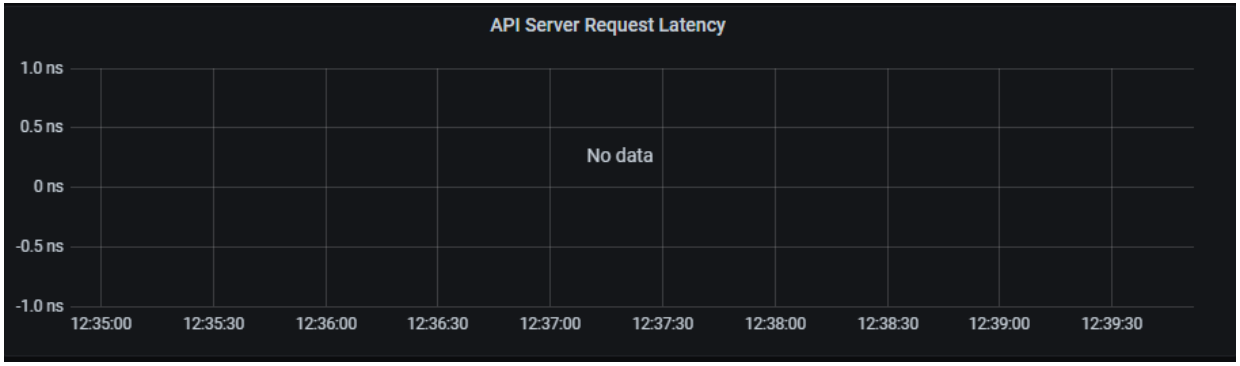
1. API Server Latency가 표시가 되지 않음

개요

- Grafana API Server Latency는 성능 지표에 매우 중요

Following metrics have been turned off: `apiserver_request_count` `apiserver_request_latencies` `apiserver_request_latencies_summary` `apiserver_dropped_requests` `etcd_request_latencies_summary` `apiserver_storage_transformation_latencies_microseconds` `apiserver_storage_data_key_generation_latencies_microseconds` `apiserver_storage_transformation_failures_total` (#83837, @RainbowMango)

- 이 지표가 Kubernetes의 1.17 버전 이상 부터 다른 지표이름이 사용이 되고 있음



2. Grafana API Server Latency

2. code 확인

1.15 version

```
requestLatencies = prometheus.NewHistogramVec(
    prometheus.HistogramOpts{
        Name: "apiserver_request_latencies",
        Help: "Response latency distribution in
        microseconds for each request handler and verb.",
        // Use buckets ranging from 125 ms to 8 seconds.
        Buckets: prometheus.ExponentialBuckets(125000, 2.
        0, 7),
    },
    []string{"handler", "verb"},
)
```

1.17 version

```
requestLatencies = compbasemetrics.NewHistogramVec(
    &compbasemetrics.HistogramOpts{
        Name: "apiserver_request_duration_seconds",
        Help: "Response latency distribution in seconds
        for each verb, dry run value, group, version,
        resource, subresource, scope and component.",
        // This metric is used for verifying api call
        latencies SLO,
        // as well as tracking regressions in this aspects.
        // Thus we customize buckets significantly, to
        empower both usecases.
        Buckets: []float64{0.05, 0.1, 0.15, 0.2, 0.25, 0.
        3, 0.35, 0.4, 0.45, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0,
            1.25, 1.5, 1.75, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5,
            5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 40, 50, 60},
        StabilityLevel: compbasemetrics.ALPHA,
    },
    []string{"verb", "dry_run", "group", "version",
    "resource", "subresource", "scope", "component"},
)
```

- apiserver_request_latencies는 apiserver.go 파일에 있음
- apiserver_request_latencies는 더 이상 지원이 되지 않음
- apiserver_request_duration_seconds는metric.go에 있음
- apiserver_request_duration_seconds가 신규 메트릭 지원 됨

2. Grafana API Server Latency

3. API Server Latency가 표시가 되지 않음

기존 지표 formula

avg(apiserver_request_latencies_sum / apiserver_request_latencies_count) by (instance)

- apiserver_request_latencies_sum, count 두 지표를 이용
- avg 함수 이용

신규 지표 formula

sum(rate(apiserver_request_duration_seconds_sum{verb!~"WATCH|WATCHLIST|PROXY|CONNECT"}[5m])) by (instance) /
sum(rate(apiserver_request_duration_seconds_count{verb!~"WATCH|WATCHLIST|PROXY|CONNECT"}[5m])) by (instance)

- apiserver_requests_duration_seconds_sum, count 두 지표 이용
- rate, sum 함수 이용

3. CSI for openstack

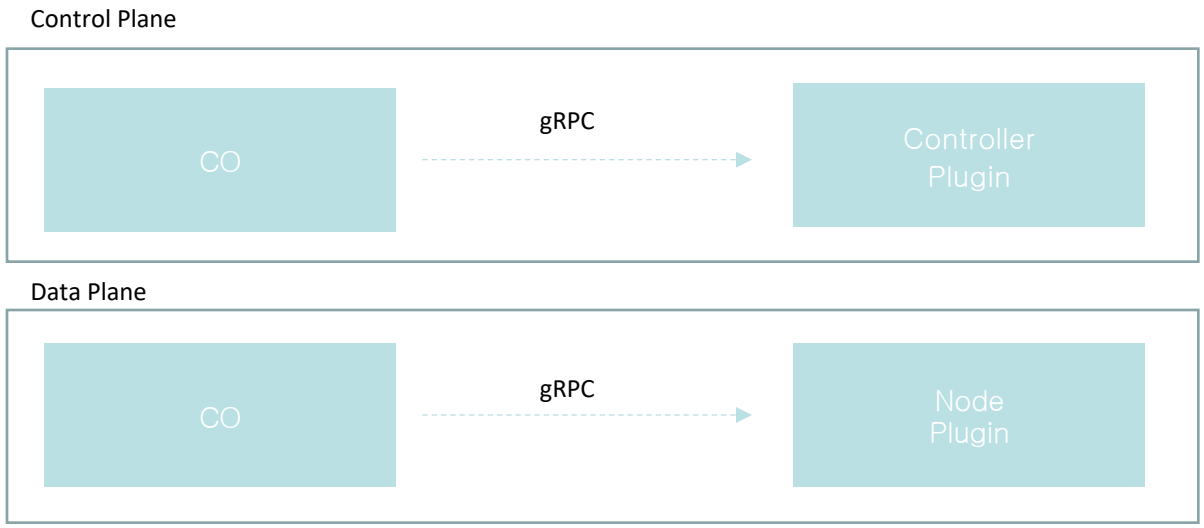
1. CSI (Container Storage Interface)

- Rancher CSI 통합 기능을 제공하지 않음
- 스토리지 솔루션 배포 시 필요
- CNCF에서 CSI 를 제시
- Storage Vendor는 CSI를 구현해서 서비스 제공

3. CSI for openstack

2. CSI 작동 방식

- Statefulset를 배포하기 위해서는 Persistent Storage가 필요
- Kubernetes는 Cloud Native 기능에서 더 이상 제공을 하지 않음
- Cloud provider openstack github에서 이 기능을 제공
- Cinder provisioner를 이용해서 PV를 제공
- Create Volume -> Controller Publish Volume -> Node Publish Volume

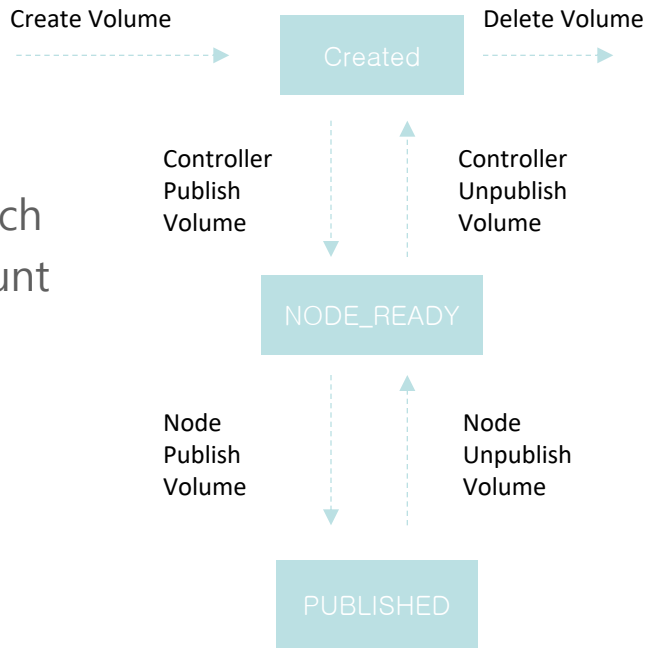


source : https://github.com/container-storage-interface/spec/blob/master/spec.md

3. CSI for openstack

3. Volume life cycle 소개

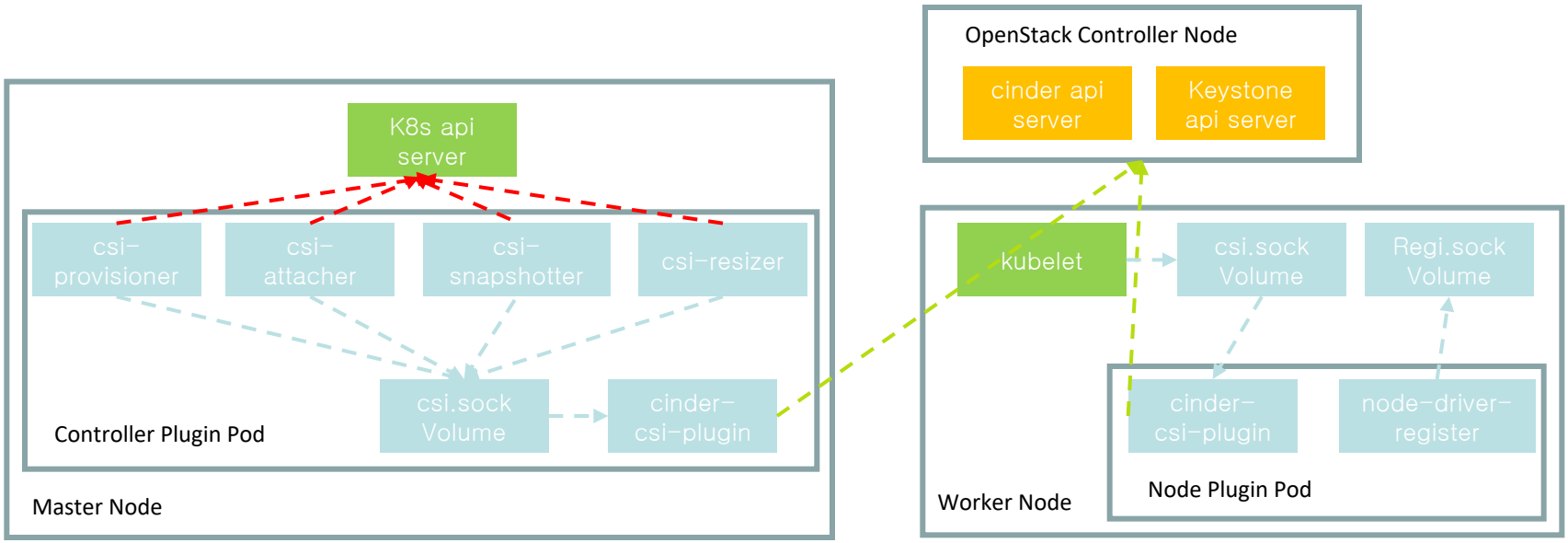
- Create Volume 시 provider를 통해 볼륨을 생성
- Controller Publish Volume은 node에 볼륨을 attach
- Node Publish Volume은 container에 볼륨을 Mount
- Node Unpublish Volume 은 unmount
- Controller Unpulish Volume은 node에 detach
- Delete Volume은 provider에서 볼륨을 제거



source : https://github.com/container-storage-interface/spec/blob/master/spec.md

3. CSI for openstack

4. Architecture



- Controller plugin pod –watch→ k8s api server
- Cinder-csi-plugin(controller plugin pod) cinder의 역할 수행
- Kubelet이 csi.sock에 요청 (NodePublishVolume/NodeUnpublishVolume)

Source : https://ssup2.github.io/theory_analysis/Kubernetes_OpenStack_Cinder_CSI_Plugin/

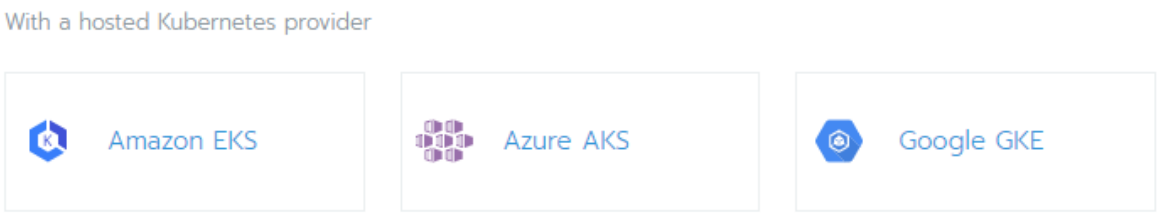
Rancher를 도입을 하고 기대효과는?

1. Public Cloud and Private Cloud 지원

1. Rancher Support Public Cloud and Private Cloud

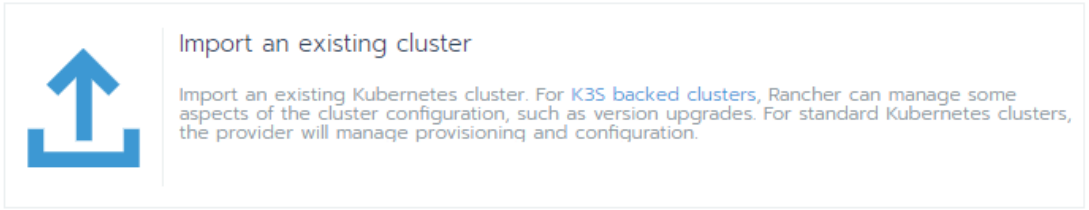
Public Cloud

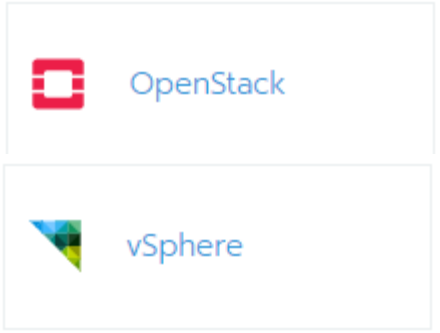
- Cloud(EKS, GKE, AKS) 지원



Private Cloud

- rke를 이용해서 bare-metal 에 구성
- 이미 IDC에 구성되어 있는 cluster 구성





1. Public Cloud and Private Cloud 지원

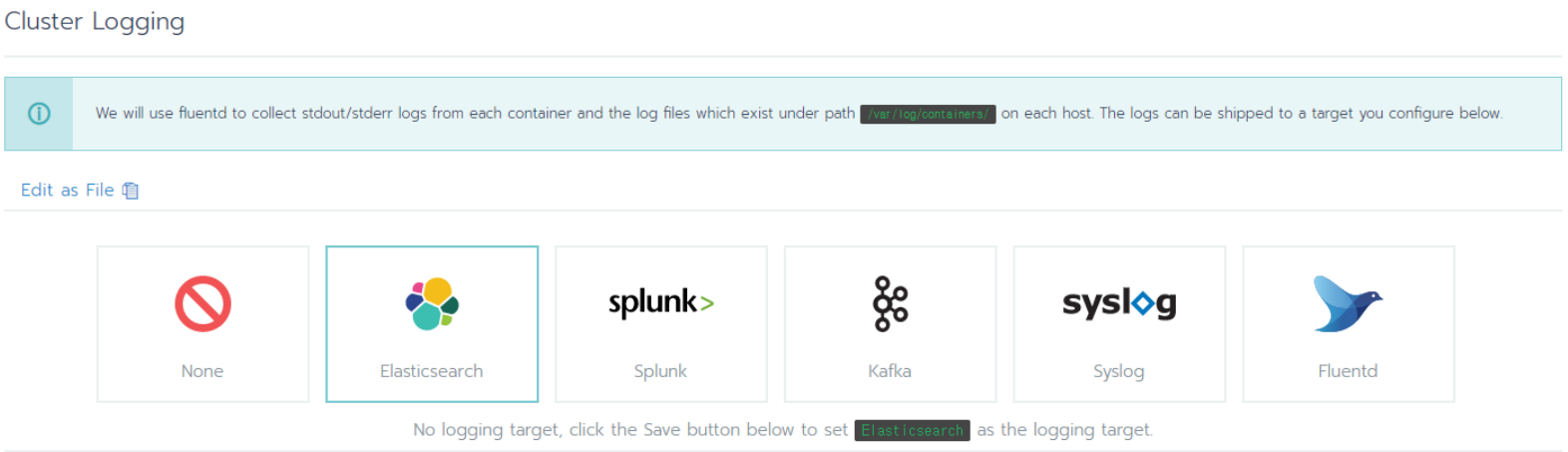
2. 장점

- 사용자에게 일관된 환경 제공
- EKS, GKE, AKS 등 다른 환경에 적응 X
- 필요에 따라 Private, Public Cloud 이동 가능

2. Enterprise-grade 서비스 제공

1. Logging

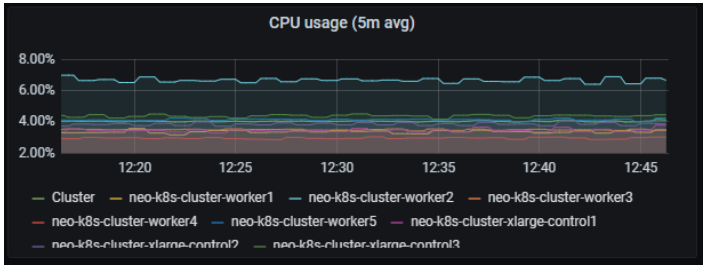
Rancher와 통합된 Agent를 이용해서 log를 전달할 수 있음(추가 작업 X)
다양한 스토리지에 Logging을 할 수 있음

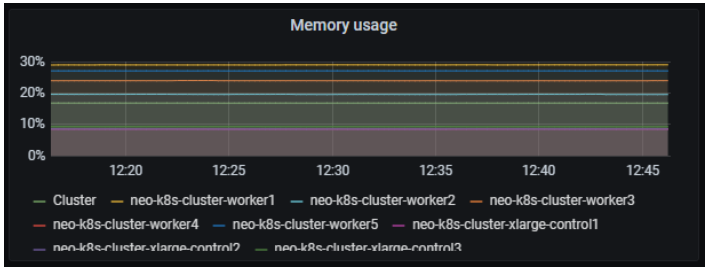


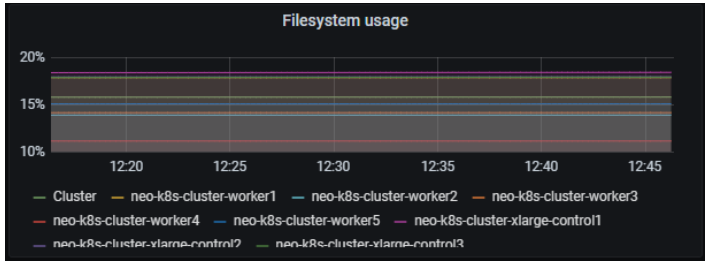
2. Enterprise-grade 서비스 제공

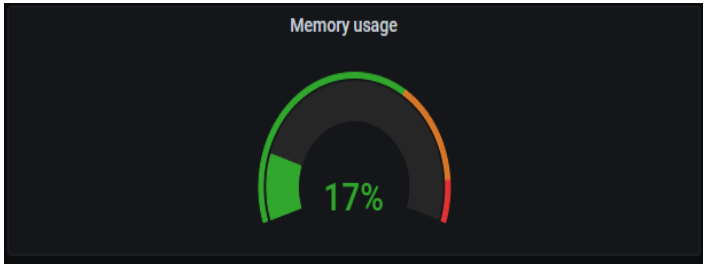
2. Monitoring

- Rancher와 통합된 Agent를 이용해서 metric을 전달할 수 있음(추가 작업 X)
- Prometheus, Grafana를 이용해서 제공
- 관리자뿐만 아니라, 사용자에게도 제공









2. Enterprise-grade 서비스 제공

3. Audit

- Rancher에서 Audit 기능을 제공
- 둘 이상의 Cluster에 대해서 Audit 기능 제공


```
{
  "auditID": "30022177-9e2e-43d1-b0d0-06ef9d3db183",
  "requestURI": "/v3/schemas",
  "sourceIPs": ["::1"],
  "user": {
    "name": "user-f4tt2",
    "group": ["system:authenticated"]
  },
  "verb": "GET",
  "stage": "RequestReceived",
  "stageTimestamp": "2018-07-20 10:22:43 +0800"
}
```

2. Enterprise-grade 서비스 제공

4. 장점

- 모든 클러스터의 사용자에게 Enterprise-grade 서비스 제공
- 위의 서비스를 제공을 하기 위해서 많은 노력 X
- 이미 잘 디자인되어진 agent를 이용해서 안정성 보장

앞으로의 과제?

1. Windows node 테스트 (for IIS)

필요한 이유

- IIS 서버 소요가 있음
- 만약, windows system을 지원화 하지 않으면 파편화가 됨 (관리 리소스 증가)

확인 할 사항

- Windows node 제한점 (CNI는 Flannel 지원 등)
- 지원하는 vendor 스토리지
- 지원하는 container runtime

2. Authentication plugins 개발

필요한 이유

- 각 사이트 마다 인증 방식이 다름
- open source 활용 시 plugin 기능을 작성해서 활용도를 높일 수 있음

코드 설명

- rancher project auth module을 보면 AuthProvider interface를 확인할 수 있음
- 이 interface를 구현해서 각 site 마다 맞는 provider에 연동할 수 있음

```
type AuthProvider interface {
    GetName() string
    AuthenticateUser(ctx context.Context, input interface{})
    (v3.Principal, []v3.Principal, string, error)
    SearchPrincipals(name, principalType string, myToken v3.
    Token) ([]v3.Principal, error)
    GetPrincipal(principalID string, token v3.Token) (v3.
    Principal, error)
    CustomizeSchema(schema *types.Schema)
    TransformToAuthProvider(authConfig map[string]interface{})
    (map[string]interface{}, error)
    RefetchGroupPrincipals(principalID string, secret string) (
    []v3.Principal, error)
    CanAccessWithGroupProviders(userPrincipalID string, groups
    []v3.Principal) (bool, error)
}
```

QnA

감사합니다.