
PORTFOLIO

이름	금기현
Email	kbzjung359@gamil.com
Github	https://github.com/KumKeeHyun/
Linkedin	https://www.linkedin.com/in/keehyunkum/
병역사항	대한민국 공군 병장 만기 전역

About Me

Golang 개발 경험

Facebook Golang Korea 커뮤니티 우수 참여자

90포인트 ①

in Golang Korea



우수 참여자 배지를 받으셨습니다!

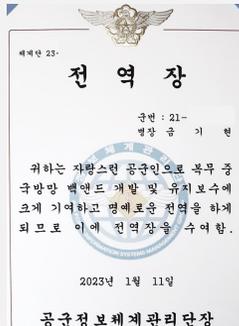
이제 회원님은 Golang Korea의 우수 참여자입니다! 계속 유지해보세요.

인트라넷 백엔드 실무 경험

신규 체계 설계 및 론칭

기존 체계 유지보수

Spring Boot, JPA, Oracle



Experience

- 2023.3 전공이 AI융합학부로 확대 개편
AI 관련 전공 강의는 수강하지 않음
- 2021.4 공군 SW개발병 복무
공군 인트라넷 백엔드 개발 및 유지보수 담당
- 2020.1 송실대학교 NCLab 학부연구생
리눅스 커널 및 RPIOS
- 2018.3 송실대학교 입학
스마트시스템소프트웨어 전공

Awards

- 2020.12 2020 공개SW개발자대회
학생부문 동상

Skills

Language	3 Golang	2 Java	
Backend	2 Gin	2 Gorm	1 Gomock
	2 Spring Boot	2 Spring MVC	2 Spring Security
	2 Spring Data JPA	1 Junit5	1 Mockito
Devops	2 MySQL	1 Oracle	
	2 Docker	2 Kubernetes	1 Helm
	1 Grafana	1 Github Actions	

- 3: 내부 동작 방식을 알고 있음, 트러블슈팅을 할 수 있음
- 2: 기본적인 동작 방식을 알고 있음, 기본적인 기능 구현 가능
- 1: 코드를 읽을 수 있음, 약간의 변경사항 추가 가능

Personal History

[1] <https://github.com/SSU-NC/toiot>

[2] <https://github.com/KumKeeHyun/raspi-cluster/tree/main/kubernetes/kubernetes-in-action>

[3] Diego Ongaro, John Ousterhout: In Search of an Understandable Consensus Algorithm

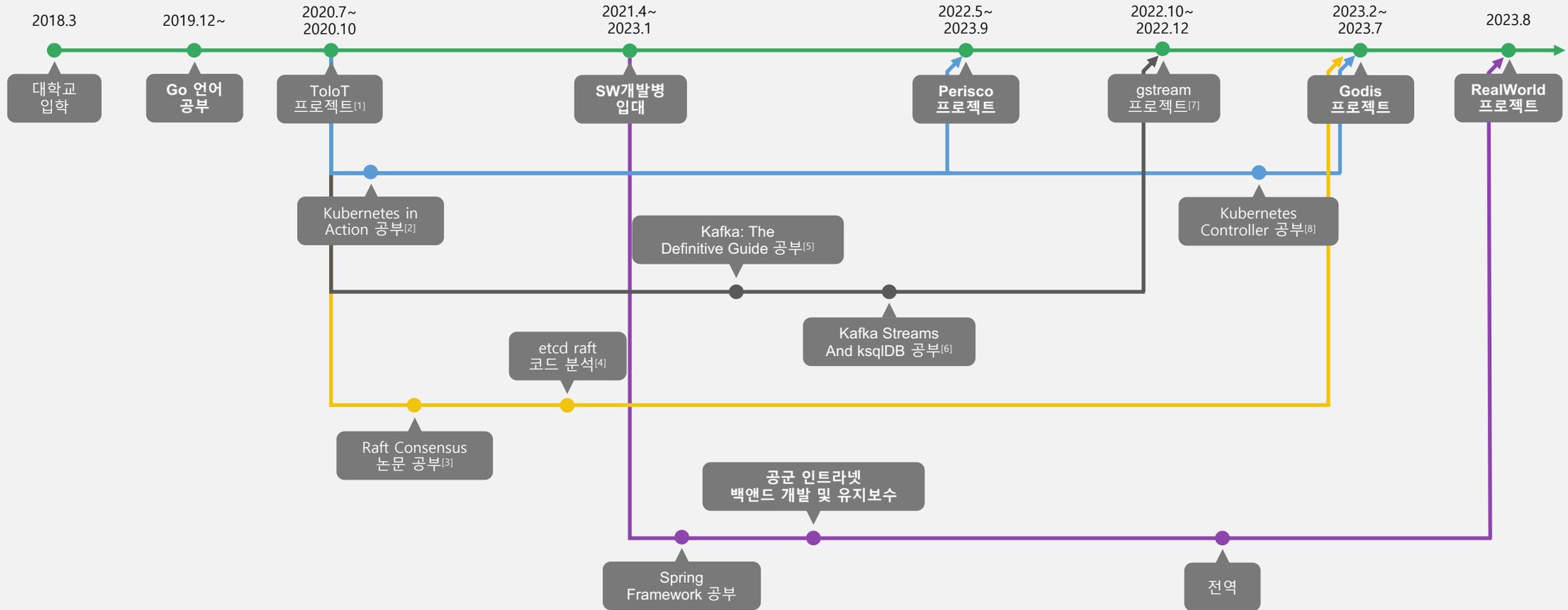
[4] <https://kumkeehyun.github.io/posts/etcd-raft-insides>

[5] <https://github.com/KumKeeHyun/raspi-cluster/tree/main/kafka/kafka-the-definitive-guide>

[6] <https://github.com/KumKeeHyun/raspi-cluster/tree/main/kafka/kafka-streams-and-ksqldb>

[7] <https://github.com/KumKeeHyun/gstream>

[8] <https://kumkeehyun.github.io/posts/kubernetes-custom-controller>



Project 01.

Perisco

About project

eBPF를 활용한 마이크로서비스 7계층 네트워크 모니터링 솔루션

개인 프로젝트

개발 기간: 2022.5 ~ 2023.9

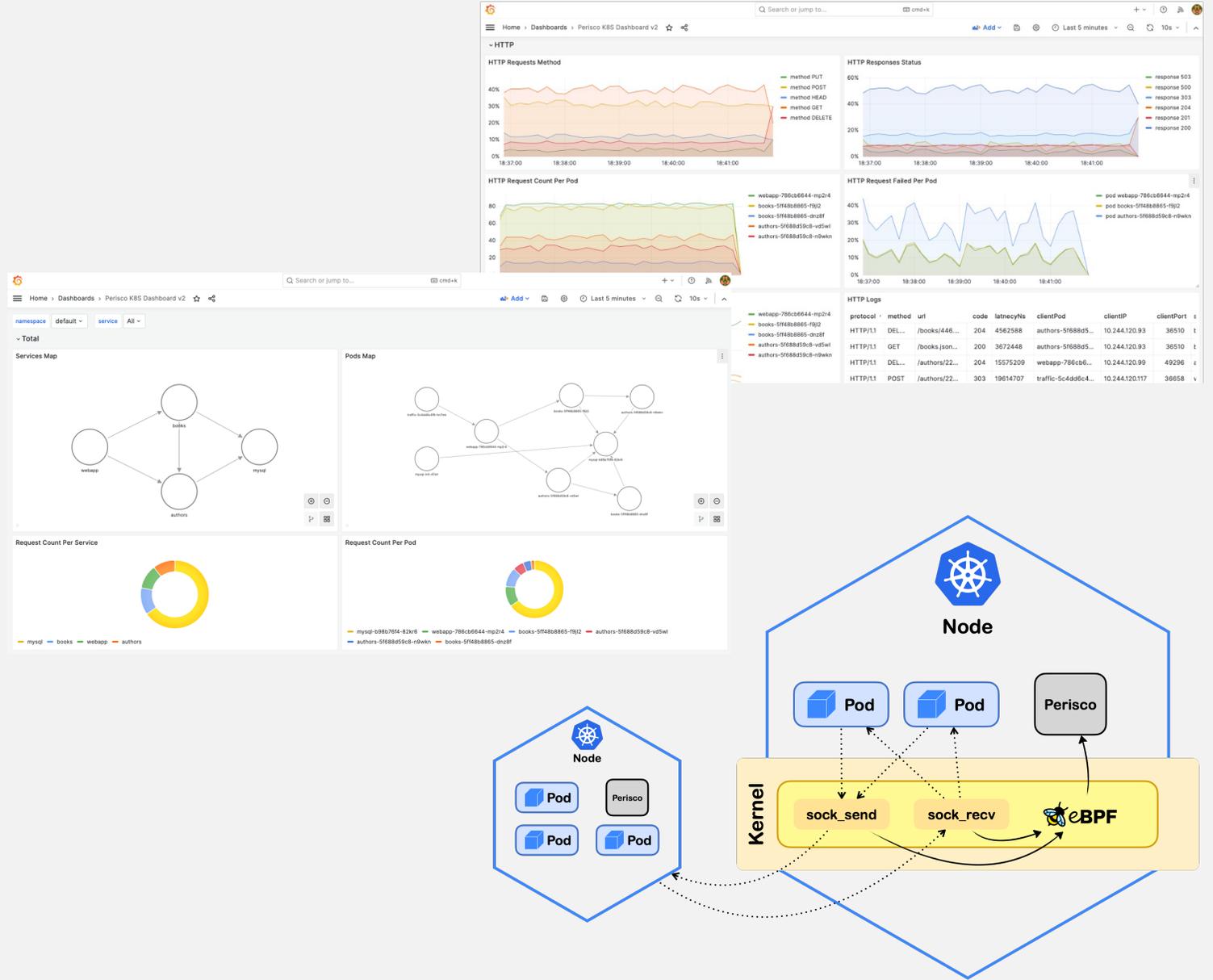
Github Repo: <https://github.com/KumKeeHyun/perisco>

• 개요

- Cilium CNI 및 Hubble의 네트워크 모니터링에서 영감을 받음
- 특정한 CNI에 종속되지 않는 독립적인 네트워크 모니터링 솔루션이 있으면 좋겠다고 생각
- 적은 오버헤드와 확장성을 고려한 설계에 중점들 두고 개발

• 작업 내용

- eBPF를 활용하여 자동으로 **소켓 송수신 페이로드**를 캡처
- Pod CIDR을 이용한 Pod 소켓 필터링
- 캡처한 페이로드를 기반으로 **7계층 네트워크 로그** 생성
- 다양한 7계층 프로토콜을 지원하기 위한 **추상화**
- HTTP/1.1, MySQL 프로토콜 지원
- 로그 장기 저장을 위한 Elasticsearch 저장소 지원
- Grafana 기본 대시보드 지원



기술 스택

• 네트워크 로그 생성 컴포넌트

- Go 1.18
- eBPF, cilium/ebpf (Go Library for eBPF)
- Ubuntu 20.04
- Linux Kernel 5.15.x
- Kubernetes 1.27.x
- Helm 3



• 모니터링 대시보드 구성

- Prometheus
- Elasticsearch 7.17.x
- Grafana



Prometheus



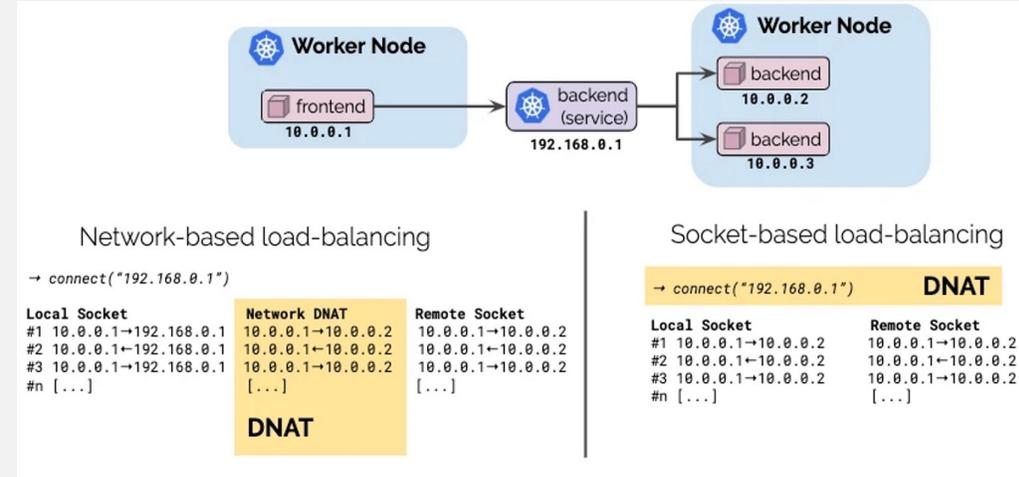
elasticsearch



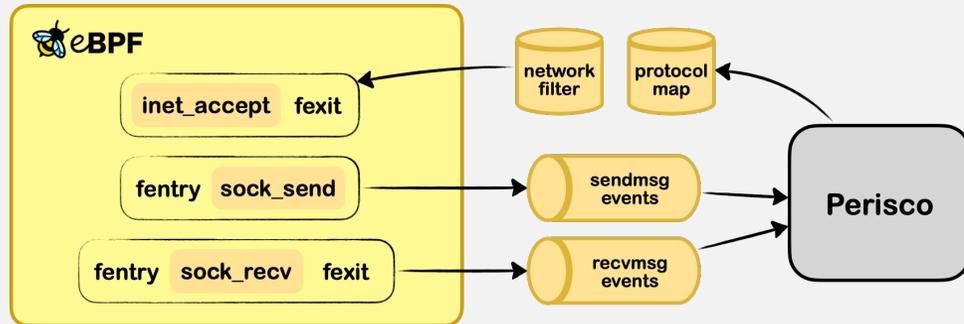
Grafana

① 소켓 송수신 페이로드를 캡처하는 eBPF 프로그램

- 모든 CNI 지원
 - Kubernetes의 NAT 환경을 고려
 - Remote Socket은 모든 CNI에서 동일한 정보를 갖고 있다는 것을 확인
 - 모든 CNI에서 일관된 동작을 보장하기 위해서 Remote Socket을 추적



<https://cilium.io/blog/2020/11/10/ebpf-future-of-networking/>

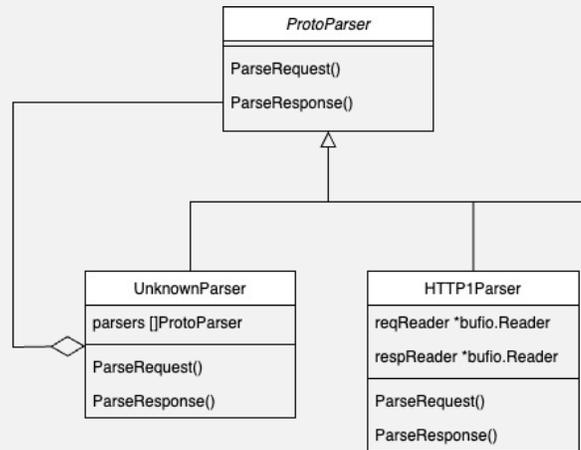


- 송수신 페이로드 캡처
 - Remote Socket을 추적하기 위해 inet_accept 이벤트 활용
 - 불필요한 Hook 실행을 피하기 위해 I/O 시스템 콜 추적을 피함
 - 페이로드 전달에 Ringbuf map을 사용하여 리소스 절약

② 7계층 네트워크 로그 생성 컴포넌트

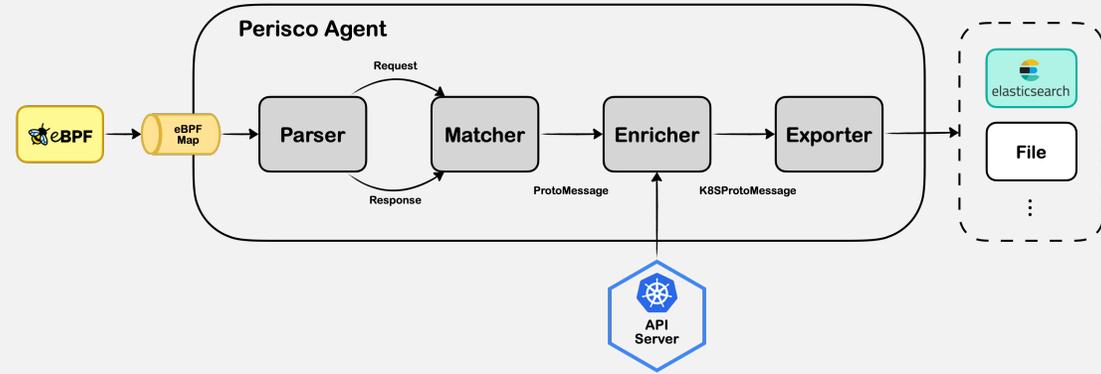
• 네트워크 로그 생성

- 네트워크 로그 생성 과정을 각 역할에 따라서 4단계로 추상화
- 각 단계를 개별적인 goroutine으로 실행시켜서 처리 성능 향상



• 페이로드를 전달받는 객체의 무분별한 생성 방지

- 크기가 약 4KB인 객체를 재활용하기 위해서 sync.Pool 활용
- 약 280,000개의 페이로드를 처리하는 동안 약 6,000개만 할당



• 다양한 7계층 프로토콜 지원을 위한 설계

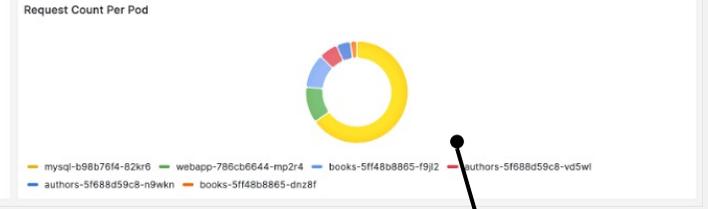
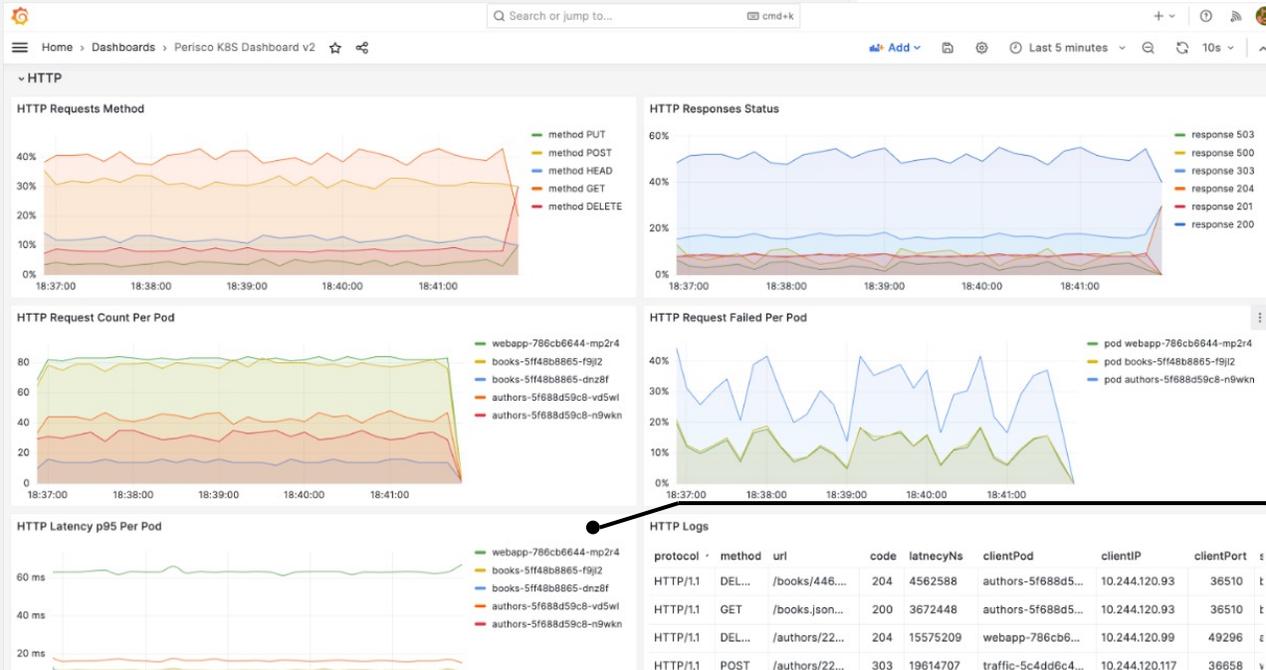
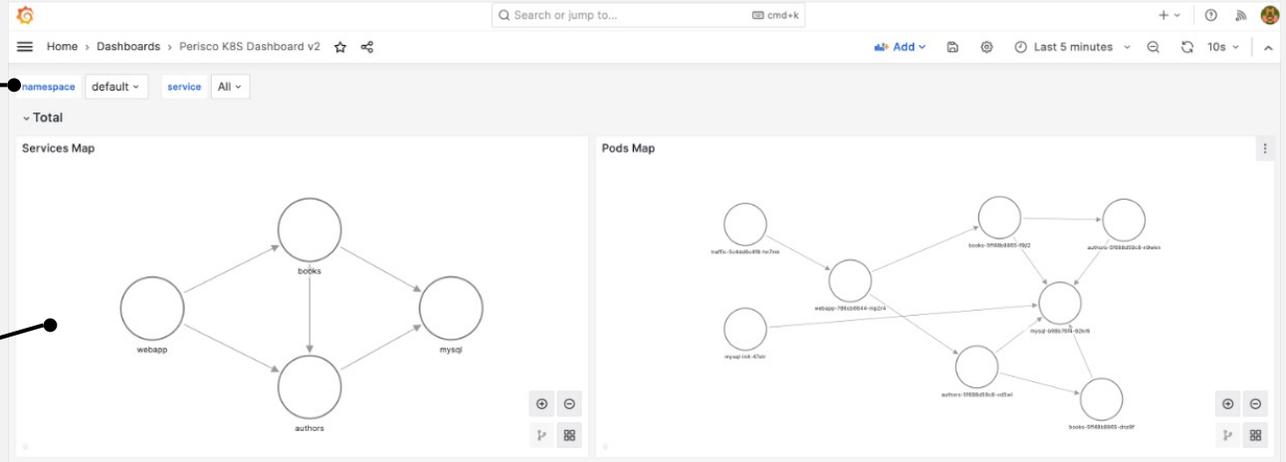
- 페이로드를 파싱하는 로직을 인터페이스로 분리
- Composite 패턴을 활용하여 프로토콜을 추정하는 기능 구현

```

# HELP perisco_debugging_bpf_msg_event_pool_new The new count in msg event pool.
# TYPE perisco_debugging_bpf_msg_event_pool_new counter
perisco_debugging_bpf_msg_event_pool_new 5735
# HELP perisco_debugging_bpf_msg_event_pool_put The put count in msg event pool.
# TYPE perisco_debugging_bpf_msg_event_pool_put counter
perisco_debugging_bpf_msg_event_pool_put 286173
# HELP perisco_debugging_bpf_recvmsg_event The count of recvmsg event.
# TYPE perisco_debugging_bpf_recvmsg_event counter
perisco_debugging_bpf_recvmsg_event 182144
# HELP perisco_debugging_bpf_sendmsg_event The count of sendmsg event.
  
```

③ Grafana 대시보드 구성

- Namespace, Service 필터링
- Grafana Variables 활용
- Service, Pod 간 의존성 확인
- Grafana Node Graph 활용



- Service, Pod 별 요청량 확인

- Pod 별 P95 응답 지연시간 확인

- HTTP 로그를 테이블 형식으로 확인

④ Minikube 및 GKE 환경에서 동작 확인

- Minikube에서 동작하는 모습

```
kumperisco5@perisco-1:~/GoProject/perisco/install/perisco-helm$ kubectl -n perisco-system get all
NAME                                     READY   STATUS    RESTARTS   AGE
pod/elasticsearch-master-0             1/1     Running  0           89s
pod/perisco-cbfcc                       1/1     Running  0           89s
pod/perisco-grafana-fc9c955dd-x7qwx     1/1     Running  0           89s

NAME                                     TYPE          CLUSTER-IP    EXTERNAL-IP  PORT(S)          AGE
service/elasticsearch-master            ClusterIP     10.100.155.34 <none>        9200/TCP,9300/TCP 89s
service/elasticsearch-master-headless   ClusterIP     None           <none>        9200/TCP,9300/TCP 89s
service/perisco-grafana                  NodePort      10.109.211.138 <none>        3000:31000/TCP   89s

NAME                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/perisco  1         1         1       1             1           <none>          89s

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/perisco-grafana  1/1     1             1           89s

NAME                DESIRED   CURRENT   READY   AGE
replicaset.apps/perisco-grafana-fc9c955dd  1         1         1       89s

NAME                READY   AGE
statefulset.apps/elasticsearch-master  1/1     89s
kumperisco5@perisco-1:~/GoProject/perisco/install/perisco-helm$ kubectl get no -o wide
NAME        STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE           KERNEL
minikube    Ready    control-plane  7d3h  v1.27.4   192.168.49.2 <none>        Ubuntu 22.04.2 LTS 5.15.0
```

```
kumperisco5@cloudshell:~/perisco/install/perisco-helm (perisco-project)$ kubectl get all -n perisco-system
NAME                                     READY   STATUS    RESTARTS   AGE
pod/elasticsearch-master-0             1/1     Running  0           14m
pod/perisco-b8bs8                       1/1     Running  0           14m
pod/perisco-czrh1                       1/1     Running  0           14m
pod/perisco-grafana-867fd99b69-4pkxz    1/1     Running  0           14m
pod/perisco-qcxxm                       1/1     Running  0           14m

NAME                                     TYPE          CLUSTER-IP    EXTERNAL-IP  PORT(S)          AGE
service/elasticsearch-master            ClusterIP     10.56.9.131   <none>        9200/TCP,9300/TCP 14m
service/elasticsearch-master-headless   ClusterIP     None           <none>        9200/TCP,9300/TCP 14m
service/perisco-grafana                  NodePort      10.56.5.151   <none>        3000:31000/TCP   14m

NAME                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/perisco  3         3         3       3             3           <none>          14m

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/perisco-grafana  1/1     1             1           14m

NAME                DESIRED   CURRENT   READY   AGE
replicaset.apps/perisco-grafana-867fd99b69  1         1         1       14m

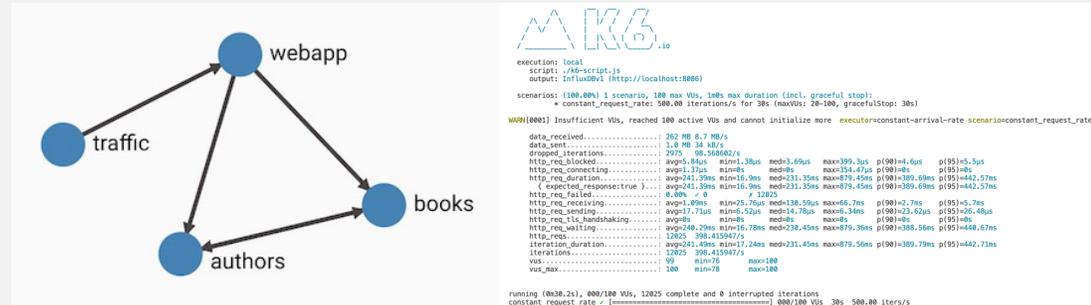
NAME                READY   AGE
statefulset.apps/elasticsearch-master  1/1     14m
kumperisco5@cloudshell:~/perisco/install/perisco-helm (perisco-project)$ kubectl get no
NAME        STATUS    ROLES    AGE   VERSION
gke-cluster-1-default-pool-cb664ed2-24vp  Ready    <none>   18m   v1.27.3-gke.100
gke-cluster-1-default-pool-cb664ed2-96dr  Ready    <none>   18m   v1.27.3-gke.100
gke-cluster-1-default-pool-cb664ed2-sxqx  Ready    <none>   18m   v1.27.3-gke.100
```

- Google Kubernetes Engine에서 동작하는 모습

- Standard 클러스터 유형 필요
- ubuntu-containerd 노드 이미지 유형 필요

⑤ Perisco 오버헤드 측정 및 타 프로젝트와 비교

- Perisco 응답 속도 오버헤드 측정을 위한 부하테스트
 - 8vcpu, 16GB VM
 - Minikube(1node, Calico CNI, containerd)
 - Booksapp 샘플 마이크로서비스 사용
 - k6(Request Index Page, 30s, max Vus 20~100)



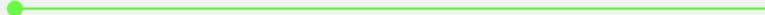
(ms)	Only Booksapp	With Perisco	With Pixie	With Linkerd
avg	241.39	267.5	260.02	295.23
min	16.9	17.98	18.17	62.12
max	879.45	954.12	822.27	817.07
p90	389.89	404.78	412.35	409.17

- Index Page 응답속도 측정
 - 평균 응답 속도에서 Pixie와 비슷한 오버헤드
 - 최소 응답 속도에서 Linkerd보다 확연히 빠른 속도

Project 02.

RealWorld Backend API

About project



Go, Gin, Gorm 기술 스택으로 구현한 RealWorld Backend API

개인 프로젝트

개발 기간: 2023.8

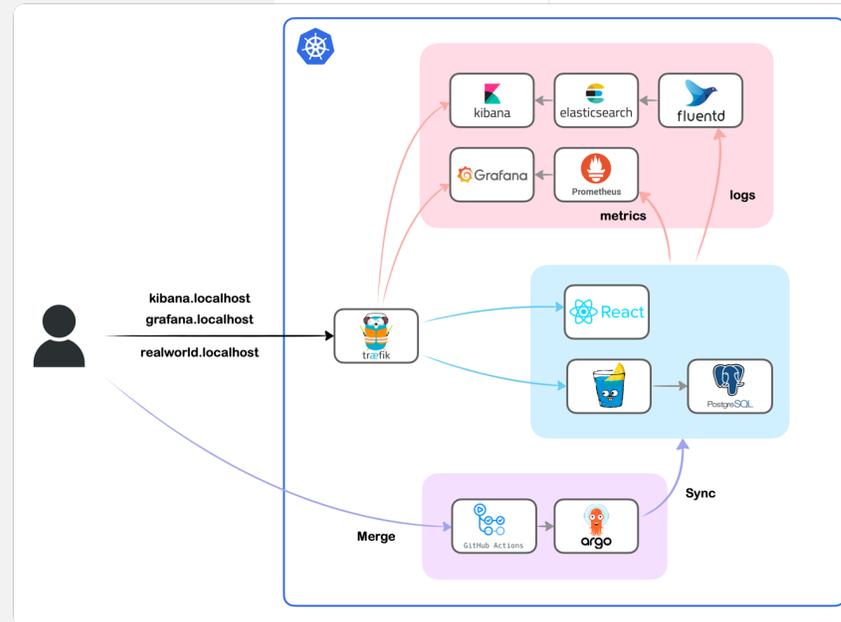
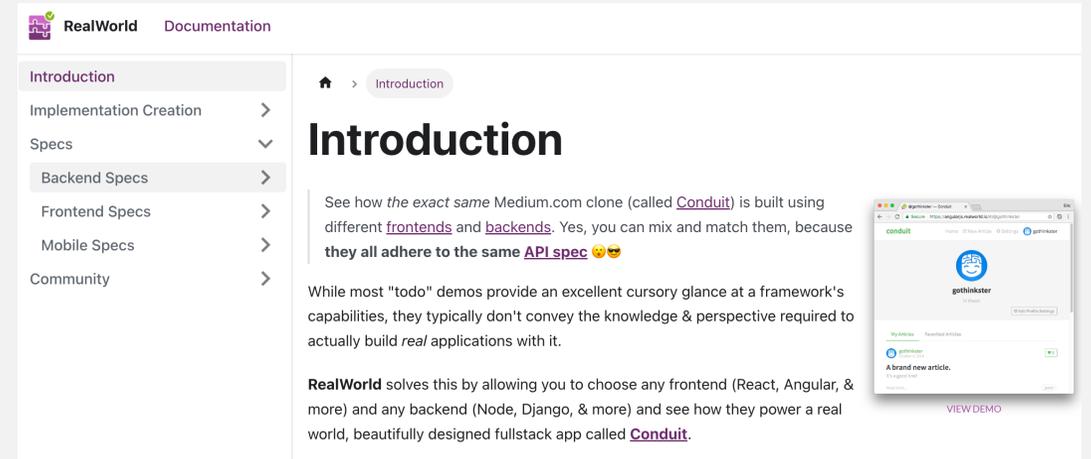
Github Repo: <https://github.com/KumKeeHyun/go-gin-realworld-example-app>

• 개요

- Spring 프레임워크 기반의 백엔드 개발 경험은 있지만 Go 언어로 본격적인 백엔드 개발 경험이 없었음
- Go 언어 기반의 백엔드 개발 역량을 기르기 위해 Gin, Gorm 기술 스택을 사용하여 RealWorld Backend API 구현

• 작업 내용

- Gin, Gorm을 사용하여 Restful API 구현
- Hexagonal Architecture 참고하여 프로젝트 구조 설계
- JWT 인증 구현
- gomock을 활용한 유닛 테스트 작성
- wire을 활용한 의존성 주입
- Github Actions, ArgoCD를 활용하여 CI/CD 적용
- Fluentd, Prometheus를 활용하여 Observability 적용



[id on Medium.](#)

기술 스택

• RealWorld Backend API

- Go 1.19
- Gin, Gorm
- wire, gomock
- PostgreSQL, sqlite3

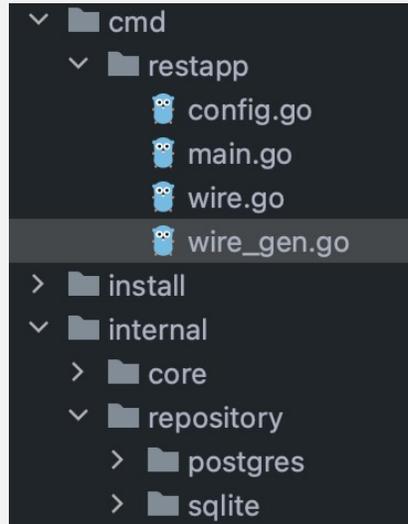
• CI/CD, Observability

- Github Actions, ArgoCD
- Kubernetes 1.27.x
- Helm 3
- Prometheus, Grafana
- Fluentd
- Elasticsearch, Kibana 7.17.x



① 백엔드 API 구현

- Hexagonal Architecture 참고하여 프로젝트 구조 설계
- ports-adapters 패턴 적용
- Repository-Service-Controller 계층화
- gomock 및 기본 testing 패키지를 활용하여 유닛 테스트 코드 작성



```

$ go generate go run github.com/google/wire/cmd/wire
$ go build -wiredinject
$ go build -wiredinject

package main

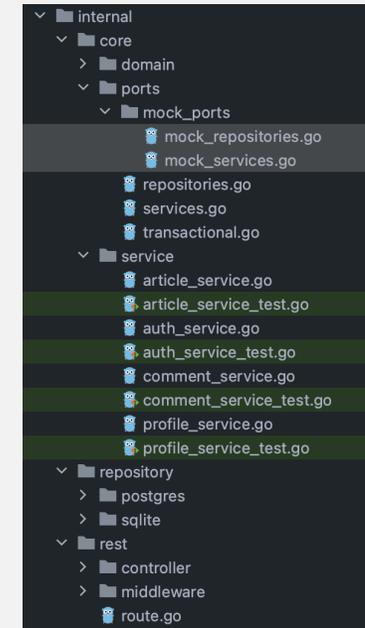
import ...

// Injectors from wire.go:

func InitRouterUsingSQLite(cfg *config, logger *zap.Logger) (*gin.Engine, error) {...}

func InitRouterUsingPostgres(cfg *config, logger *zap.Logger) (*gin.Engine, error) {
    jwtUtil := InitJwtUtil(cfg)
    checkJwtMiddleware := middleware.NewCheckJwtMiddleware(jwtUtil, logger)
    ensureAuthMiddleware := middleware.NewEnsureAuthMiddleware(logger)
    ensureNotAuthMiddleware := middleware.NewEnsureNotAuthMiddleware(logger)
    db, err := InitDataSource(cfg, logger)
    if err != nil {
        err := err
    }
    transactionMiddleware := middleware.NewTransactionMiddleware(db, logger)
    errorsMiddleware := middleware.NewErrorsMiddleware(logger)
    metricMiddleware := middleware.NewMetricMiddleware()
    userRepository := postgres.NewUserRepository(db)
    articleRepository := postgres.NewArticleRepository(db)
    authService := service.NewAuthService(userRepository, articleRepository, jwtUtil, logger)
    authController := controller.NewAuthController(authService)
    profileService := service.NewProfileService(userRepository, logger)
    profileController := controller.NewProfileController(profileService)
    articleService := service.NewArticleService(articleRepository, userRepository, logger)
    articleController := controller.NewArticleController(articleService)
    commentRepository := postgres.NewCommentRepository(db)
    commentService := service.NewCommentService(commentRepository, articleRepository, userRepository, logger)
    commentController := controller.NewCommentController(commentService)
    engine := rest.NewRouter(logger, checkJwtMiddleware, ensureAuthMiddleware, ensureNotAuthMiddleware, transactionMiddleware, errorsMiddleware, metricMiddleware)
    return engine, nil
}

```



• 개발 환경과 배포 환경 분리

- 개발 환경(sqlite3), 배포 환경(postgresql)을 위한 Repository 구현
- wire를 활용하여 의존성 주입 처리

② CI/CD, Observability 적용

• Github Actions workflow 작성

- Pull Request 이벤트시 바이너리 빌드, 테스트, 이미지 생성 수행
- Push 이벤트시 빌드, 테스트, 이미지 생성, 이미지 저장 수행
- 개발 환경(arm64), 테스트 환경(amd64)을 위해 멀티 플랫폼 빌드 설정

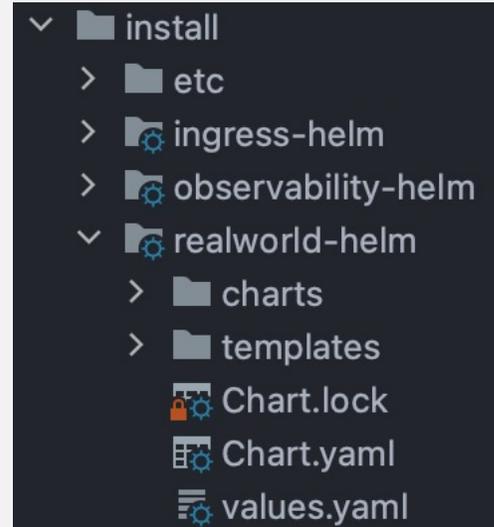
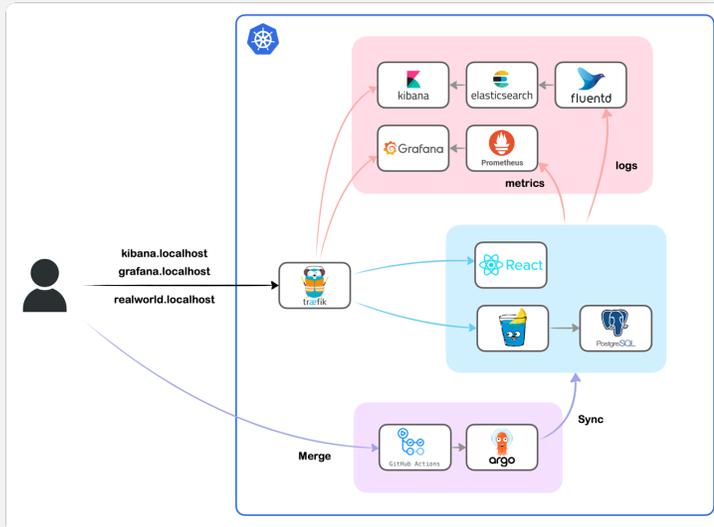
```

32 docker-image:
33   needs: [build-and-test]
34   if: ${{ vars.EXEC_WORKFLOW == 'true' }}
35   runs-on: ubuntu-latest
36   steps:
37     - uses: actions/checkout@v3
38
39     - name: Docker meta
40       id: meta
41       uses: docker/metadata-action@v4
42       with:
43         <2 keys>
44
45     - name: Set up QEMU
46       uses: docker/setup-qemu-action@v2
47
48     - name: Set up Docker Buildx
49       uses: docker/setup-buildx-action@v2
50       with:
51         <1 key>
52
53     - name: Login to Docker Hub
54       if: github.event_name != 'pull_request'
55       # You may pin to the exact commit or the version.
56       # uses: docker/login-action@465a7811f14bbb193fbed4728cca1ff8901fc
57       uses: docker/login-action@v2.2.0
58       with:
59         <2 keys>
60
61     - name: Build and push
62       # You may pin to the exact commit or the version.
63       # uses: docker/build-push-action@2eb1c1941a95fc15694676618e422e8ba1d63825
64       uses: docker/build-push-action@v4.1.1
65       with:
66         <6 keys>

```

• Helm Charts 작성

- Frontend, Backend, PostgreSQL을 배포하기 위한 Charts 작성
- Prometheus, Fluentd를 활용하여 Metrics, Logs 모니터링 구성
- Traefik을 활용하여 Ingress 구성



Project 03.

Godis

About project

Redis의 기능을 제한적으로 지원하는
간단한 분산 인메모리 데이터베이스

개인 프로젝트

개발 기간: 2023.2 ~ 2023.7

Github Repo: <https://github.com/KumKeeHyun/godis>

• 개요

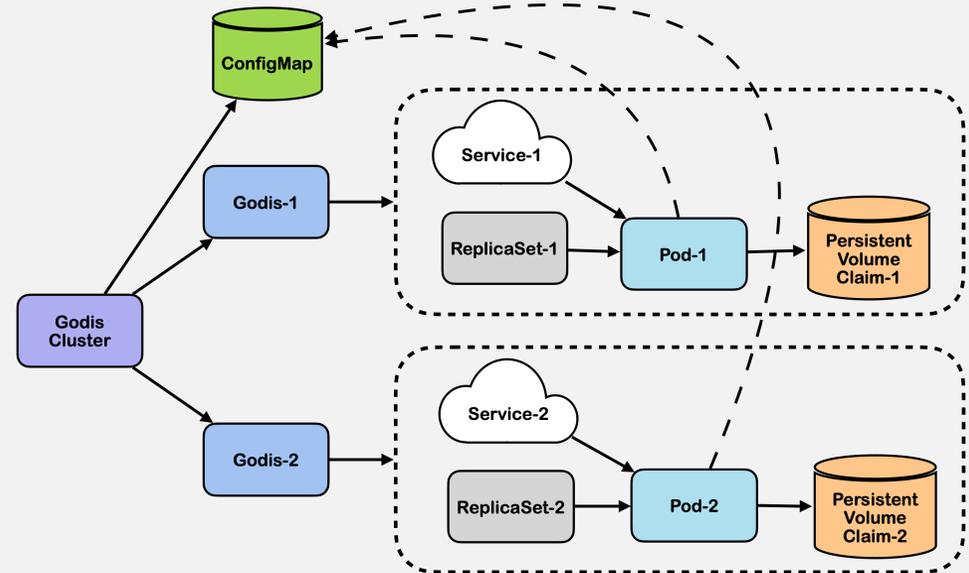
- Redis의 작동 방식을 이해하기 위해서 'Build Your Own Redis with C/C++' 책을 읽고 Go 언어로 따라서 구현
- 이전에 etcd raft 라이브러리를 분석했던 경험을 활용하고자 해당 라이브러리를 활용하여 복제 기능 구현
- Kubernetes 선언형 배포 개념을 공부하기 위해서 Godis를 위한 CRD 및 Custom Controller 구현

• 작업 내용

- Redis Serialization Protocol V2 구현
- In-Memory Key-Value, TTL 저장소 구현
- Redis GET, SET, MGET 커맨드 구현
- Raft Consensus 알고리즘 기반 Active-Standby 복제 기능 구현
- Snapshot 기반 복구 기능 구현
- Kubernetes Custom Controller를 활용한 선언형 배포 지원
- etcd raft 라이브러리의 example 코드에서 오류를 찾아 수정 후 Pull Request 요청
- 프로젝트를 진행하면서 얻은 지식을 블로그 글로 정리하여 커뮤니티에 공유

```
./godis client --port 6379
2023/03/09 19:04:02 start client
2023/03/09 19:04:02 send request to 127.0.0.1:6379
127.0.0.1:6379> set k v
"OK"
127.0.0.1:6379> get k
"v"
127.0.0.1:6379> get k
(error) redis: nil
127.0.0.1:6379> get k
"v"
127.0.0.1:6379> get k
"v"
127.0.0.1:6379> get k
(error) redis: nil
127.0.0.1:6379>

./godis client --port 6380
2023/03/09 19:04:04 start client
2023/03/09 19:04:04 send request to 127.0.0.1:6380
127.0.0.1:6380> get k
"v"
127.0.0.1:6380> set k v ex 5
"OK"
127.0.0.1:6380> get k
"v"
127.0.0.1:6380> set k v ex 5
"OK"
127.0.0.1:6380> 
```



기술 스택

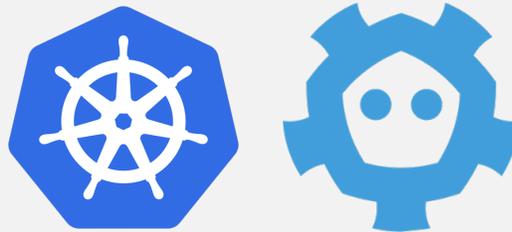
- **Godis 데이터베이스**

- Go 1.19
- etcd raft v3
- spf13/cobra, spf13/viper
- stretchr/testify



- **쿠버네티스 커스텀 컨트롤러**

- Go 1.19
- Kubernetes 1.25.x



① 간단한 분산 인메모리 데이터베이스

• Redis의 일부 String 명령어

- Redis Serialization Protocol V2를 구현하여 다른 Redis Client와 호환
- SET 명령어의 TTL 관련 옵션들도 충실하게 구현
- MGET 명령어의 반환 형식인 Array Reply도 충실하게 구현

```

2023/03/09 18:39:51 send request to 127.0.0.1:6379
127.0.0.1:6379> set k v
"OK"
127.0.0.1:6379> get k v
"v"
127.0.0.1:6379> set k vv get ex 5
"v"
127.0.0.1:6379> get k
"vv"
127.0.0.1:6379> get k
(error) redis: nil
127.0.0.1:6379> set k1 v1
"OK"
127.0.0.1:6379> set k2 v2
"OK"
127.0.0.1:6379> mget k k1 k2
1) (nil)
2) "v1"
3) "v2"

```

```

2023/03/09 19:04:04 send request to 127.0.0.1:6380
127.0.0.1:6380> get k
"v"
127.0.0.1:6380> set k v ex 5
"OK"
127.0.0.1:6380> get k
"v"
127.0.0.1:6380> set k v ex 5
"OK"
127.0.0.1:6380> 

```

→

```

2023/03/09 19:04:02 send request to 127.0.0.1:6379
127.0.0.1:6379> set k v
"OK"
127.0.0.1:6379> get k
"v"
127.0.0.1:6379> get k
(error) redis: nil
127.0.0.1:6379> get k
"v"
127.0.0.1:6379> get k
"v"
127.0.0.1:6379> get k
(error) redis: nil
127.0.0.1:6379> 

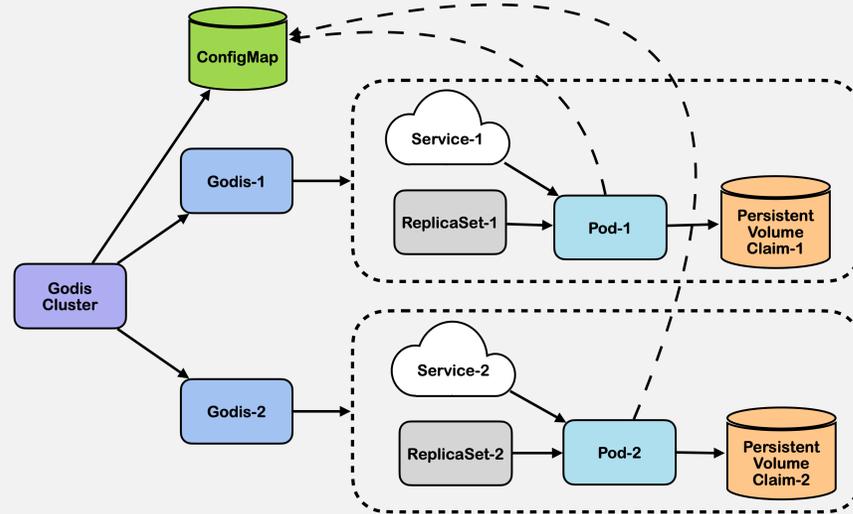
```

• Raft Consensus 알고리즘 기반 복제 기능

- Master-Read Replica 형식의 구성
- row-based replication 방식으로 TTL도 문제 없이 작동
- Quorum만큼 복제된 후에 쓰기 응답을 보내도록 구성하여 안정성 향상

② 선언형 배포를 위한 Kubernetes Custom Controller

- 선언적 배포를 위한 CRD 설계
 - WAL, Snapshot을 통한 복구를 이용하기 위해 ReplicaSet, PVC 사용
 - Node ID의 용이한 관리를 위해 CRD를 GodisCluster-Godis 2단계로 구성



- Replicas=3 으로 클러스터 생성

```

~$ cat example-godis.yml
apiVersion: kumkeehyun.github.com/v1
kind: GodisCluster
metadata:
  name: example-godis
spec:
  name: example-godis
  replicas: 3

~$ k apply -f example-godis.yml
godiscluster.kumkeehyun.github.com/example-godis created

~$ k get po -l "cluster-name=example-godis"
NAME                READY   STATUS              RESTARTS   AGE
example-godis-1-cqmg5 0/1     ContainerCreating   0           5s
example-godis-2-x2cj8 0/1     ContainerCreating   0           5s
example-godis-3-6vcn2 0/1     ContainerCreating   0           4s
  
```

- Godis 1, 2, 3 생성

```

~$ k get po -l "cluster-name=example-godis"
NAME                READY   STATUS              RESTARTS   AGE
example-godis-1-cqmg5 1/1     Running             0           3m53s
example-godis-2-x2cj8 1/1     Running             0           3m53s
example-godis-3-6vcn2 1/1     Running             0           3m52s

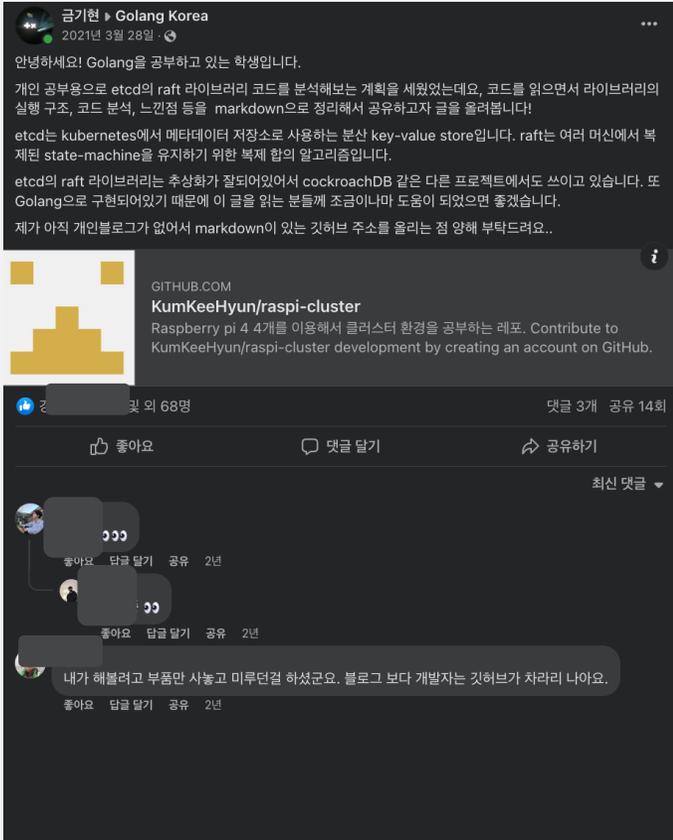
~$ k edit godiscluster
godiscluster.kumkeehyun.github.com/example-godis edited

~$ k get po -l "cluster-name=example-godis"
NAME                READY   STATUS              RESTARTS   AGE
example-godis-1-cqmg5 1/1     Running             0           4m28s
example-godis-2-x2cj8 1/1     Running             0           4m28s
example-godis-3-6vcn2 1/1     Running             0           4m27s
example-godis-4-8bmzs 0/1     ContainerCreating   0           4s
  
```

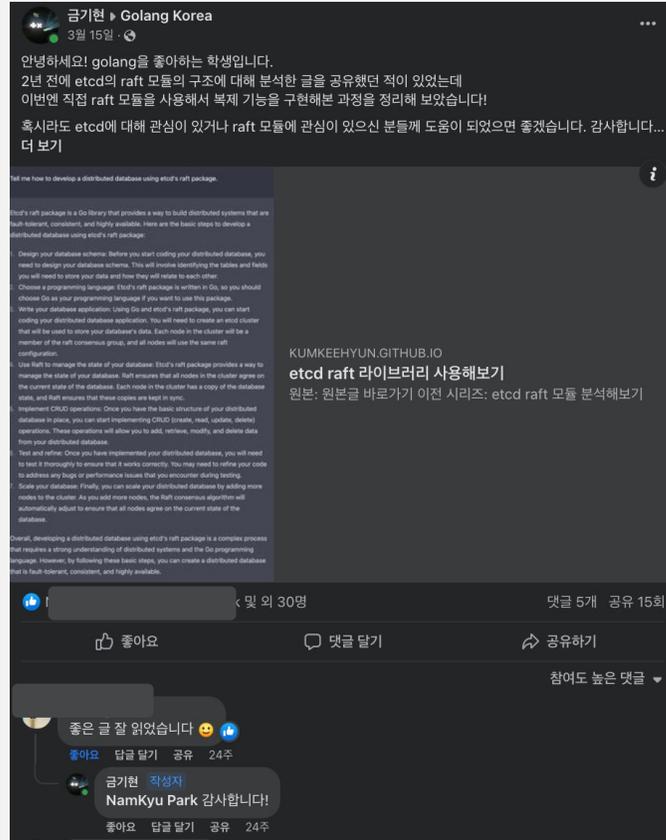
- Replicas=4 으로 수정

- Godis 4 생성

Main work ③ 프로젝트에서 공부한 내용을 블로그 글로 정리해서 Golang Korea 커뮤니티에 공유



<https://kumkeehyun.github.io/posts/etcd-raft-insides>



<https://kumkeehyun.github.io/posts/using-etcd-raft>



<https://kumkeehyun.github.io/posts/kubernetes-custom-controller>

Project 04.

ToloT

About project

대규모 센서 데이터를 쉽게 수집/가공/시각화 할 수 있는 IoT 플랫폼

🏆 2020 공개SW개발자대회 학생 부문 동상 수상

팀 프로젝트: BE 2명, FE 1명, 임베디드 1명 (BE 기여도 70%)

개발 기간: 2020.7 ~ 2020.10

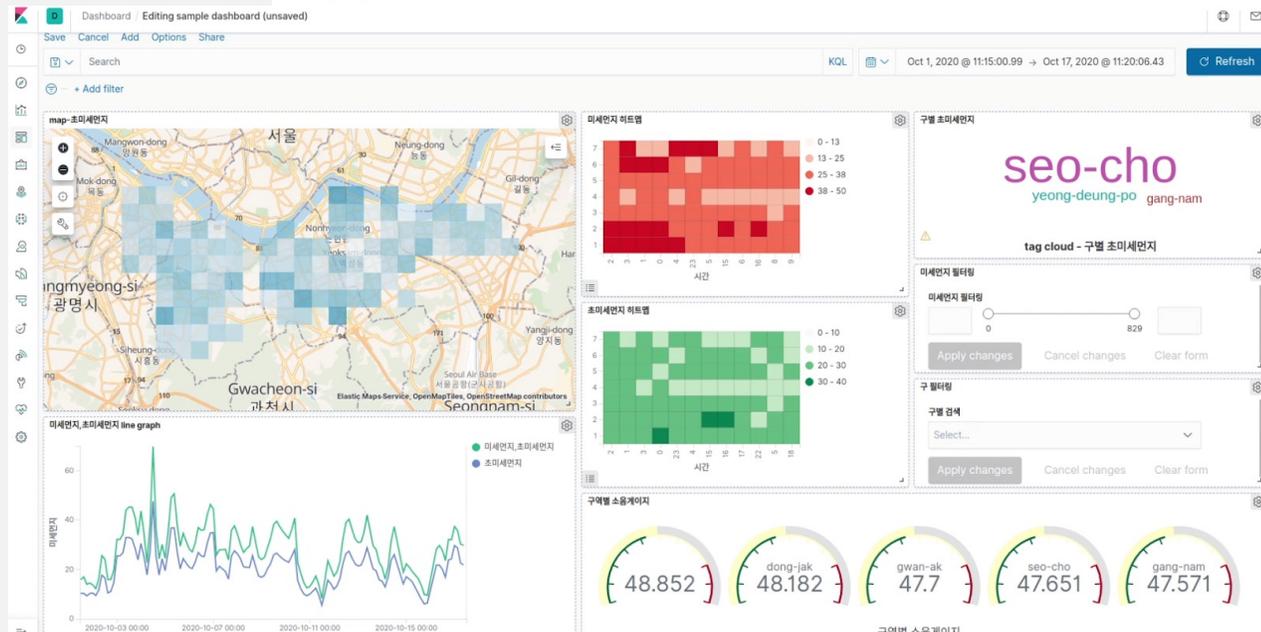
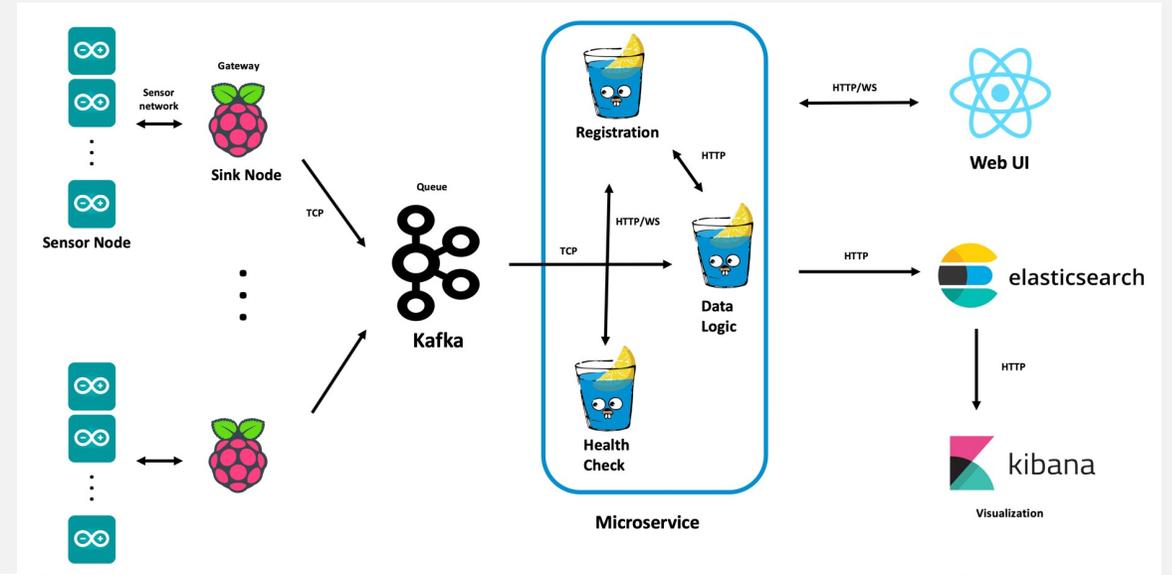
Github Repo: <https://github.com/SSU-NC/toiot>

• 개요

- 당시 사회적 이슈였던 미세먼지 및 공기질 문제에서 착안
- 공기질 측정 범위가 구/동 단위로 유의미한 정보를 전달하기엔 측정 밀도가 너무 낮다고 판단
- 도로명 단위 및 실내 단위 측정을 위해서 대규모 센서 데이터를 다룰 수 있는 IoT 플랫폼을 목표로 진행

• 맡은 역할

- IoT 장비 및 센서 관리를 위한 **API 서버** 개발
- Kafka에 적재된 센서데이터를 가공 및 분석하고 Elasticsearch에 저장하는 **ETL 파이프라인** 개발
- Kafka Consumer Group을 활용한 **센서 데이터 분산 처리**
- 센서 측정 데이터 실시간 분석 및 이메일 알림 기능 구현



기술 스택

- **API Server**

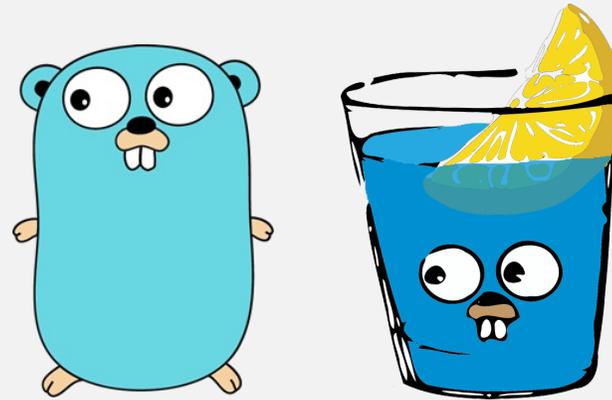
- Go 1.13
- Gin, Gorm
- MySQL

- **ETL Pipeline**

- Go 1.13
- IBM/sarama (Go Library for Kafka)
- Kafka 2.5.x
- Elasticsearch 7.6.x
- Kibana 7.6.x

- **ETC**

- Docker



① IoT 장비 및 센서 관리 화면

- 지도에 위치를 클릭하여 위도/경도 지정
- Kakao Map API 활용

HOME MANAGEMENT KAFKA SERVICE KIBANA

Node register node

Viewer type All Map

• : don't know • : stable • : unstable • : disconnect

#	name	id	sensors	health
0	node-gwan-ak-17	79	air-dust, air-noise, air-temp,	•
1	node-gwan-ak-21	83	air-dust, air-noise, air-temp,	•
2	node-gwan-ak-27	89	air-dust, air-noise, air-temp,	•

Register node

Node name
ji-node-1

location

 Click map!
 latitude : 37.4969548236336, longitude : 126.95742348753718

Select sensors
 sensor-1 x sensor-2 x sensor-3 x

Select sink
 soongsil

Submit Cancel

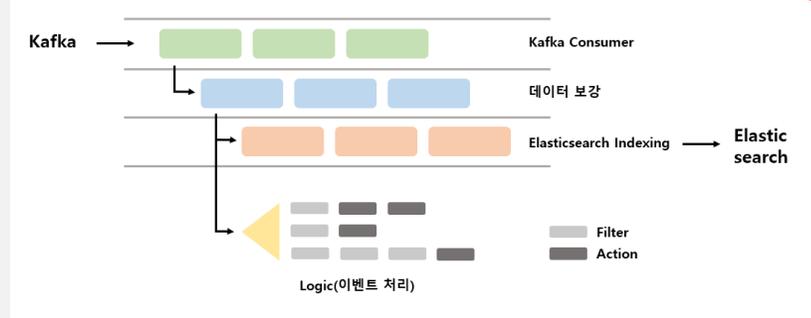
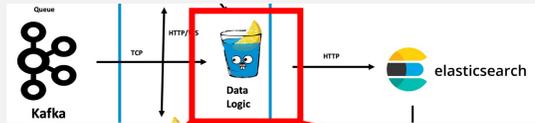
• 측정 센서 지정

- 등록된 IoT 장비 위치 확인
- 좌상단, 우하단 위도/경도로 사각형 검색 범위 생성

• 지도에 표시된 IoT 장비들 목록

② 사용자 정의 센서 데이터 분석 및 알림 기능

- 사용자 정의 로직 등록 화면
- 동적으로 데이터 분석 로직 등록 가능



Register Logic
A logic is registered at Logic Core

Logic name
dust-logic

Build Logic

sensor: dust

group: seo-cho

time: [Add scope]

value #0: fine-dust, 30 < fine-dust < 255 [Add scope]

action #0: email, Email address: balns@q.ssu.ac.kr [Add value] [Add action]

- 트리거 조건 지정
- 시간 필터링 기능
- 센서 측정값 필터링 기능
- 트리거 액션 지정
- 이메일 알림 기능

- 로직 등록시 실시간으로 ETL Pipeline에 로드

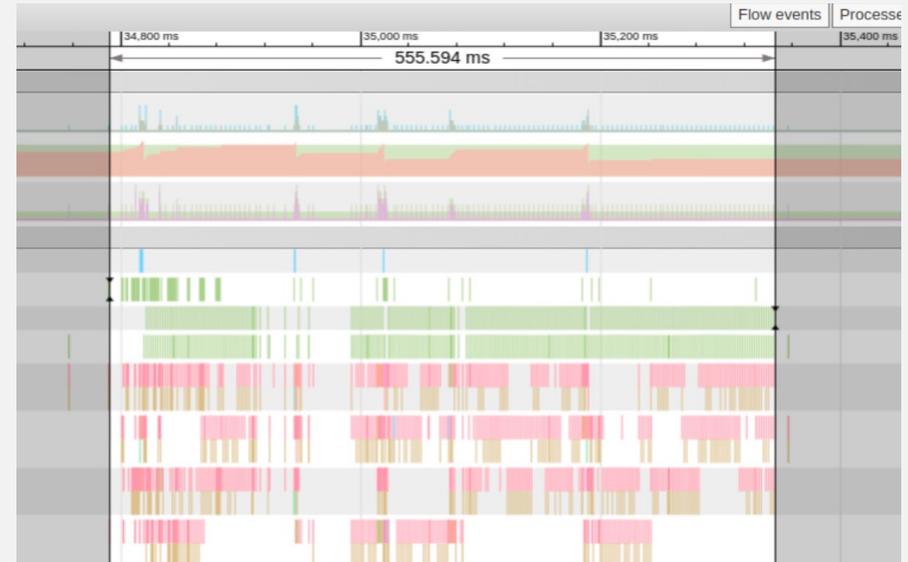


③ Kafka Consumer Group을 활용한 분산 처리

- ETL Pipeline 성능 확인을 위한 시뮬레이션
 - 서울시 공공데이터 미세먼지 측정 데이터 100,000개 사용
 - Kafka에 적재된 100,000개 데이터를 모두 처리하는데 걸리는 시간 측정

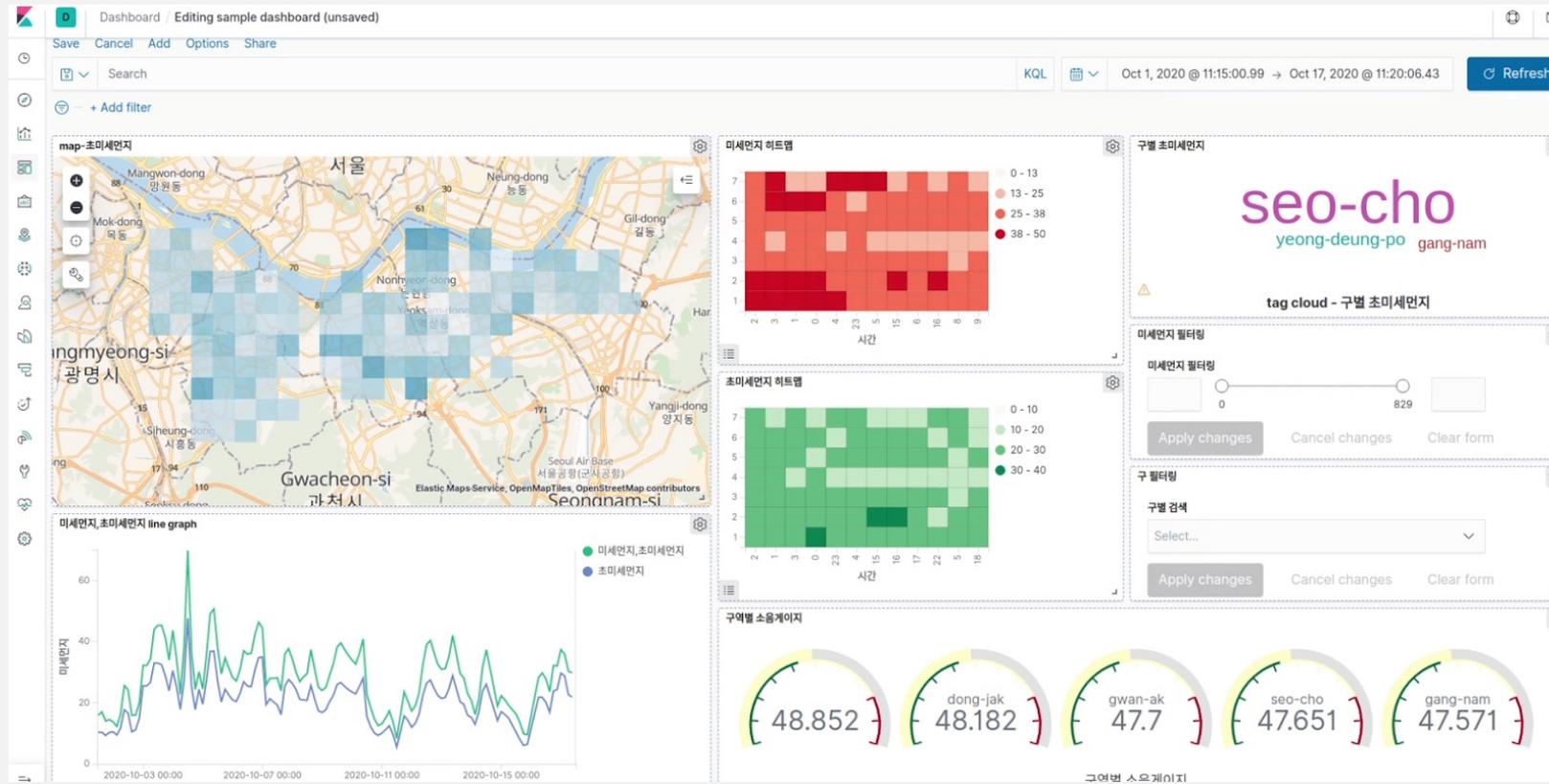


• 인스턴스 1개로 시뮬레이션 -> 100,000개/1초



• 인스턴스 2개로 시뮬레이션 -> 100,000개/0.55초

④ Kibana 대시보드 구성



* 서울시 공공데이터 미세먼지 측정 데이터 활용
위치는 시뮬레이션을 위해 임의 지정