DU-도전학기 결과보고서

과제명	세미나 기반의 포트폴리오 생성을 통한 역량 점검 및 개발		
참여자	성명	소속	학번
	이	컴퓨터 소프트웨어전공	
	곽	컴퓨터 소프트웨어전공	
	김	AI 소프트웨어 전공	
	김	컴퓨터 소프트웨어전공	
지도교수 의견	학부생 4명에서 직접적으로 클라우드 시스템을 개발하고 그 과정에서 현업 개발자에게 도움을 받는 등 상호 협력하여 우수한 성과를 보였습니다. 개발과정에서 개발성과와 전공분야 지식 습득에 서로 도움을 주는 등 매우 협력적인 모습을 보이는 등 우수한 학업 태도로 도전학기에 임하였습니다. 지도교수로서 학점을 부여할 수 있다면 A+을 부여하고 싶습니다. (소속) 컴퓨터 소프트웨어전공 (성명) (서명 또는 날인)		

1. 도전 과제 내용

이름	과제 내용	비고
이	팀원 지도 및 전공지식 공유 개발 기획 - 세미나 내용 피드백 - 소프트웨어 설계 및 멘토링 - 도전학기 행정업무	
김	운영체제 이론 학습 및 세미나 담당 서버개발 담당	
김	도커 컨테이너 기술에 대한 학습 및 세미나 담당 서버개발 담당	
곽	쿠버네티스 기술에 대한 학습 및 세미나 담당 Web Front-end React 라이브러리 개발 및 node.js 개발 go언어 세미나 진행	

2. 도전 과제 수행 결과 및 성과

1. 개발 소프트웨어 구성도

가. 소프트웨어 사양서

이름	버전
OS	Rocky Linux 8.5
Kernel	4.18.0-348.12.2.el8_5.x86_64
Node.js	v14.19.1
React	v17.0.2
Go	v1.19
Prometheus	v15.18.0
Grafana	v6.43.5
Helm	v3.7.0
Containerd	v1.6.8
Kubernetes	v.1.25.2
Docker	v20.10.18

표 1. 개발 소프트웨어 버전

- Production과 비슷한 환경을 유지하기 위해 redhat의 RHEL 풀카피 버전인 rocky linux 채용
- Frontend 및 middleware는 node.js로 구현이 되었으며, backend는 golang으로 작성됨
- Frontend는 react framework를 사용하여 개발되었으며, middleware는 express 프레임 워크를 사용함
- kubernetes 및 containerd, docker는 개발 당시 latest 버전을 사용하였음
- 이미지 및 패키지 배포를 위한 도구는 helm chart를 사용
- 메트릭 수집 및 데이터 가시화를 위한 prometheus, grafana는 helm chart로 배포함

Port 번호	Protocol	목적
22	TCP	SSH 접속
80	TCP	Http 웹 서버 접속
443	TCP	Https 웹 서버 접속
3000	TCP	Frontend 기본 port 번호
5000	TCP	Middleware 기본 port 번호
6443	TCP	API server 기본 port 번호
8080	TCP	Http 웹 서버 접속
9100	TCP	Node exporter 기본 port 번호
10250	TCP	Kubelet 기본 port 번호
30000-32767	TCP/UDP	NodePort 대역

표 2. 오픈된 port 리스트

나. Kubernetes 구성도

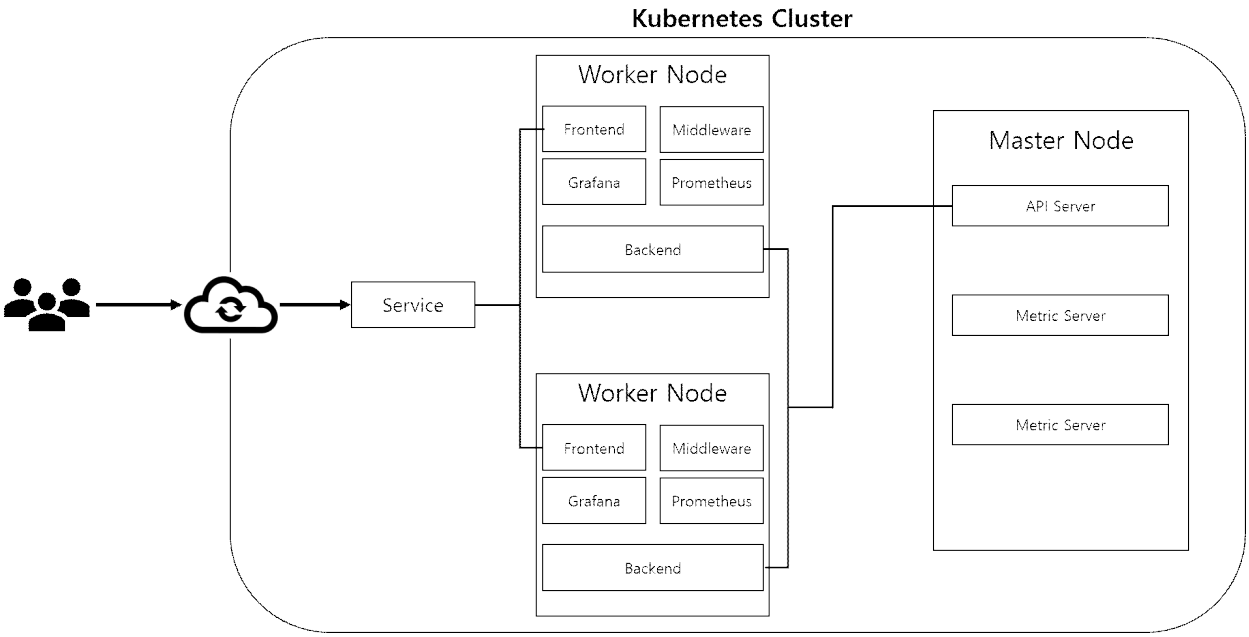

그림 1. 개발된 소프트웨어를 위한 Kubernetes 구성도

- 구현된 Kubernetes는 그림 1과 같이 1개의 Master node, 2개의 worker node로 구성되어 있음
- 개발한 소프트웨어는 모두 worker node에 배치가 됨
- Frontend 및 middleware는 사용자 접근을 위한 node port로 구성됨
- Backend는 clusterIP로 외부 접근은 불가능하고 클러스터 내부 접근만 가능함
- Backend는 middleware의 요청을 받고 master node의 API server에게 데이터 처리 요청 후 결과 값을 리턴받음
- Prometheus는 grafana 및 backend 메트릭 제공을 위해 사용되며, clusterIP로 구성됨
- Prometheus에서 node exporter 및 kube-metric 서버에서 메트릭을 수집
- Grafana는 frontend인 react에게 데이터 시각화를 제공함
- Node 간 pod 네트워크를 위한 서드파티 CNI는 weave network를 사용함(vxlan)

다. Frontend 구성도

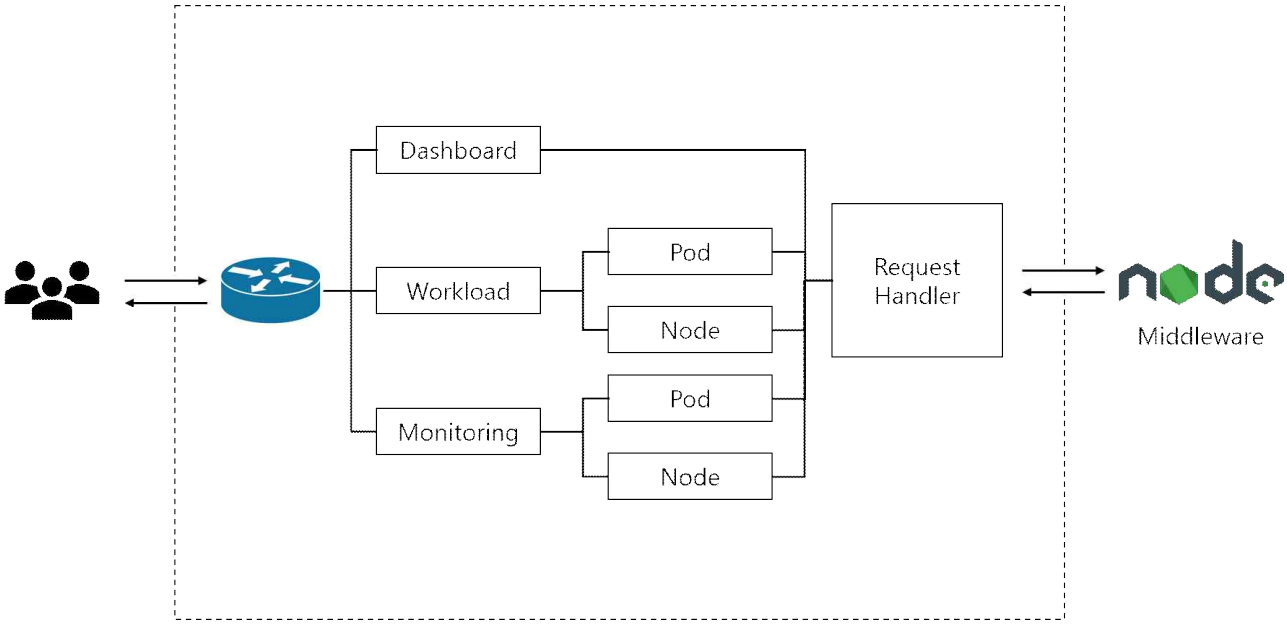

그림 2. Frontend 구성도

그림 3. Grafana 연동

- Frontend는 그림 2와 같이 dashboard, workload, monitoring 3개의 정보를 제공함
- Dashboard는 노드 개수, pod 개수, 프로세서 개수 및 상태 등 cluster 및 노드의 종합 정보를 제공함
- Workload는 pod 및 node의 리스트 혹은 제어 관련 부분을 제공함
- Monitoring은 pod, node의 리소스 사용량을 제공함
- Monitoring은 그림 3과 같이 grafana와 연동하여 제공됨
- Grafana는 iframe 태그를 사용하여 대시보드 전체 정보를 제공함
- Grafana는 기본적으로 iframe share 기능이 비활성화 되어있어 chart의 values.yaml에서 옵션을 수정해야함
- [grafana.ini].security.allow_embbeding: true 값을 추가하여 해결

라. Middleware 구성도

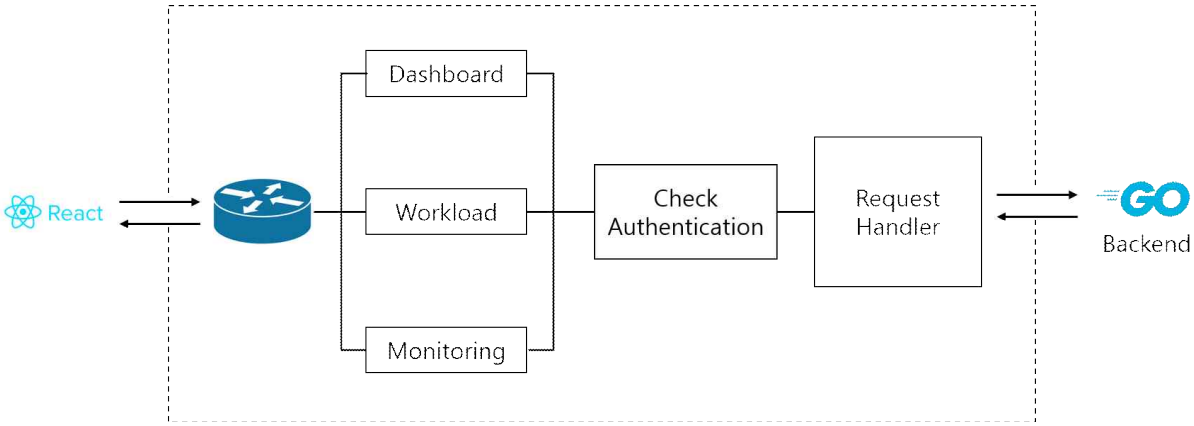

그림 4. Middleware 구성도

Key	Type	설명
node_id	string 40byte	kubernetes node UUID
node_name	string 255byte	kubernetes node의 hostname
node_address	string 255byte	kubernetes node ipv4 주소
node_label_key	string 255byte	kubernetes node label의 key
node_label_value	string 255byte	kubernetes node label의 value
node_role	string 255byte	kubernetes node의 role(master, worker)
Authorization	string 2000byte	사용자 token 값
pod_id	string 40byte	pod의 UUID
pod_name	string 255byte	kubernetes pod 이름
pod_namespace	string 255byte	pod가 속한 namespace 이름
pod_address	string 255byte	pod의 ipv4 주소
pod_nodeName	string 255byte	pod가 속한 node의 hostname
pod_service	string 255byte	pod service 이름

표 3. REST API 쿼리 목록

- Middleware는 frontend와 동일한 라우팅 구조를 가지며, backend에게 요청 전 사용자의 authentication 정보를 확인하기 위해 사용됨
- Frontend에서 전달받은 cookie 헤더를 확인 후 query 데이터를 파싱하여 backend에게 전달
- workload, monitornig에서 사용되는 query는 표 3과 같음

마. Backend 구성도

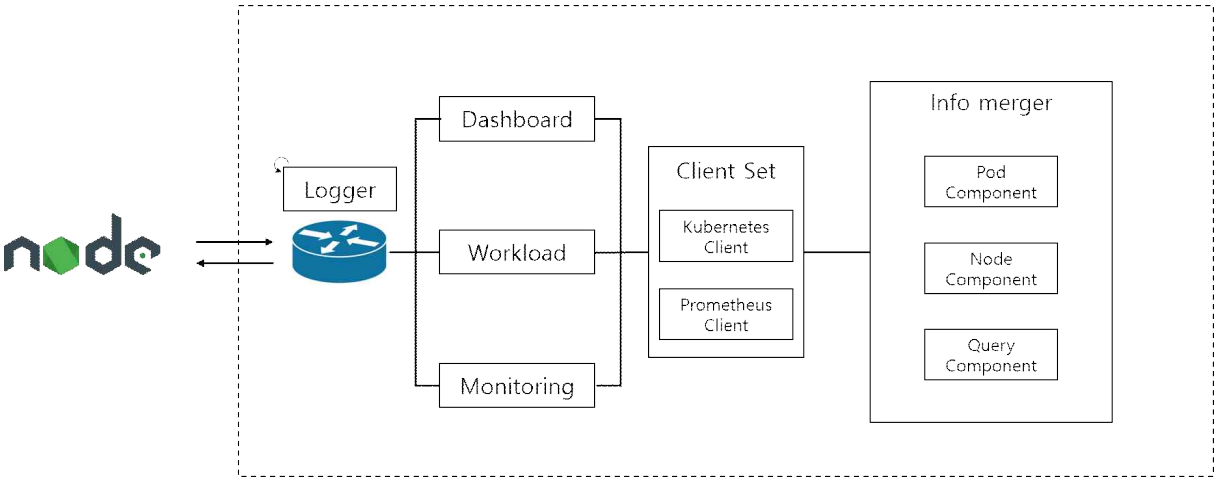

그림 5. Backend 구성도

Key	Type	설명
K8sConfigPath	string `json:"k8sConfigPath"`	kubernetes config file 위치
BrokerListenPort	int `json:"brokerListenPort"`	backend 서버 실행 시 사용될 port 번호
BrokerProtocol	string `json:"brokerProtocol"`	backend 서버에 사용될 프로토콜(http, https)
PromAddress	string `json:"prometheusAddress"`	prometheus ipv4 주소
PromPort	int `json:"prometheusPort"`	prometheus port 번호
PromProtocol	string `json:"prometheusProtocol"`	prometheus에서 사용되는 프로토콜(http, https)

표 4. Backend config 목록

- Backend는 사용자의 정의에 따라 별도의 재컴파일 없이 config file을 읽어 사용하도록 처리
- config의 key, value는 표 4와 같음
- Backend는 middleware로부터 전달받은 query를 읽어 처리
- 모든 request는 사전에 정의된 client set을 통해 필요한 각 컴포넌트를 통해 데이터가 취합됨
- client set은 초기화 과정에서 전역변수로 저장이 되며, 값이 변경되지 않음
- client set은 getter, setter 함수를 통하여 컴포넌트 패키지에 사용됨
  (deadlock 및 invalid memory access 방지를 통한 무결성 보장)

2. 서드파티 시스템 구성도

가. Prometheus

```
197   persistentVolume:
198     ## If true, alertmanager will create/use a Persistent Volume Claim
199     ## If false, use emptyDir
200     ##
201     enabled: false
```

그림 6. Prometheus PV 비활성화

```
prom-kube-state-metrics          ClusterIP   10.103.187.98   <none>   8080/TCP
prom-prometheus-alertmanager     ClusterIP   10.96.150.107   <none>   80/TCP
prom-prometheus-node-exporter    ClusterIP   10.108.59.20    <none>   9100/TCP
prom-prometheus-pushgateway      ClusterIP   10.103.63.121   <none>   9091/TCP
prom-prometheus-server           ClusterIP   10.101.59.45    <none>   80/TCP
```

그림 7. Prometheus Services

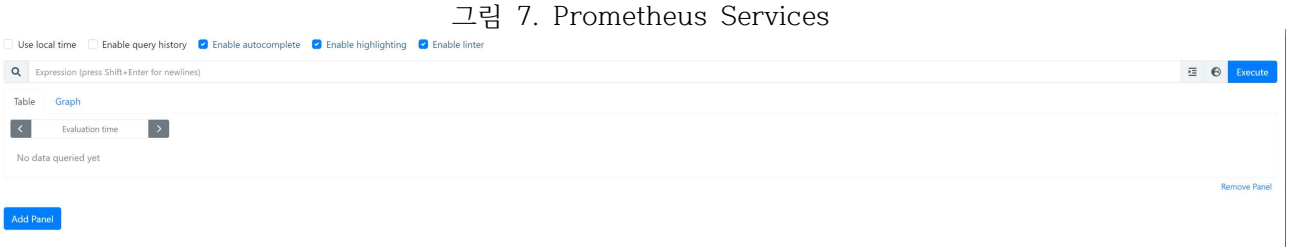

그림 8. Prometheus 초기 화면

- Prometheus github에 제공되는 helm chart를 clone하여 사용함
- Prometheus는 기본적으로 PV(Persistent volume)을 요구하며, 별도의 설정이 필요함
- 본 개발과정에서 장기간 데이터 축척은 불필요하여 그림 6과 같이 values.yaml의 persistentVolume.enabled를 비활성화
- Prometheus는 monitoring namespace에서 배포하며, 그림 7과 같이 ClusterIP 사용
- 배포 완료 후 prometheus-server IP로 접속하여 그림 8과 같이 초기 화면 접속 확인

나. Grafana

```
675 grafana.ini:
676   security:
677     allow_embedding: true
678   auth.anonymous:
679     enabled: true
```

그림 9 Grafana Iframe 태그 사용을 위한 옵션

```
grafana          NodePort    10.107.54.241   <none>        80:30400/TCP
```

그림 10 Grafana Service

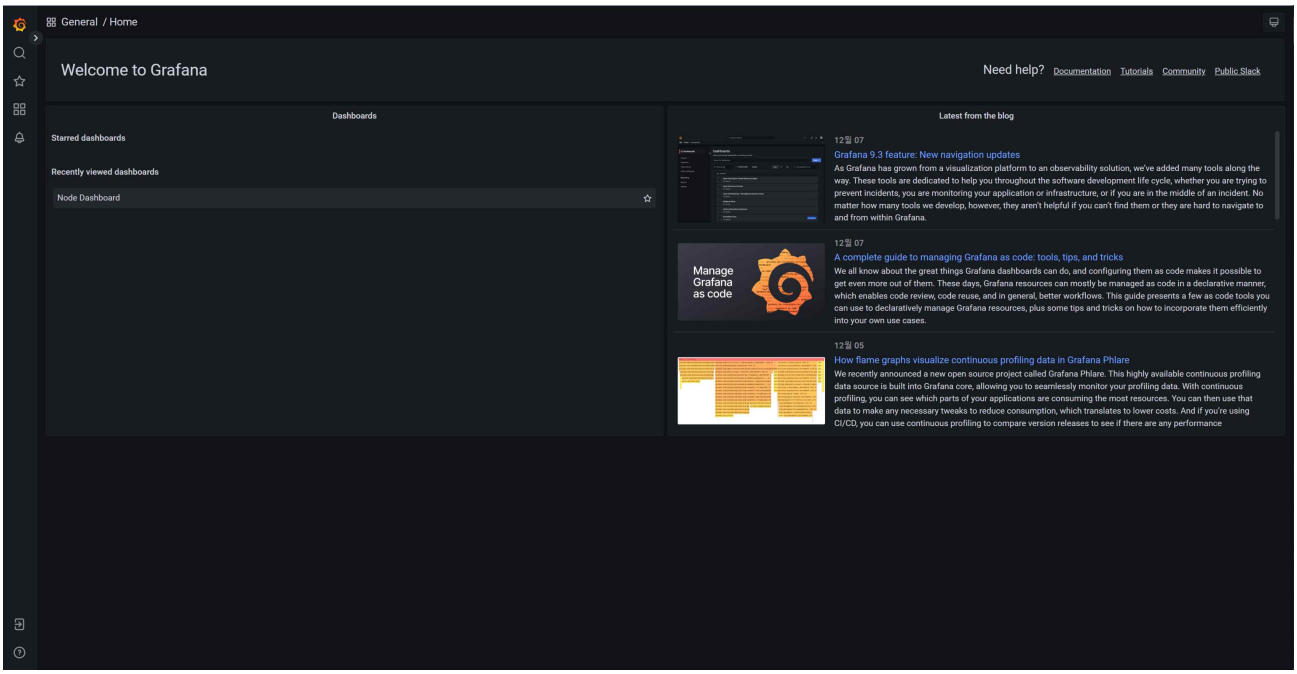

그림 11. Grafana 초기 화면

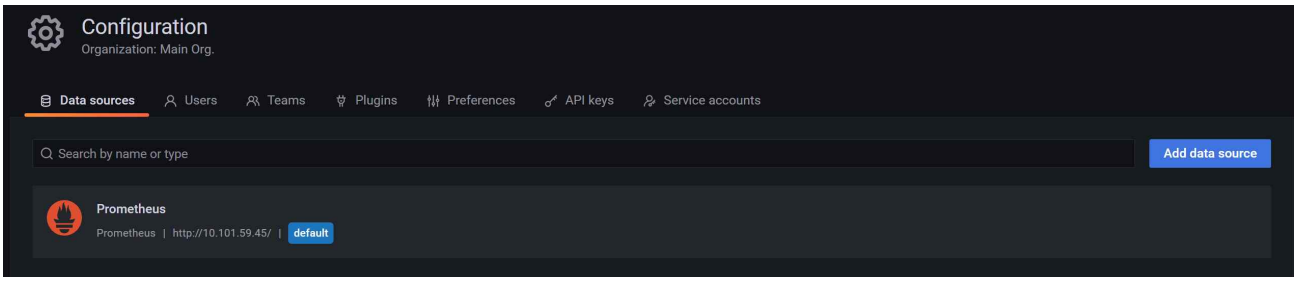

그림 12. Grafana와 prometheus 연동

- Grafana는 prometheus와 동일하게 helm chart를 사용하여 배포함
- Frontend 연동을 위해 values.yaml의 grafana.ini 수정
- 외부접속 및 react 연동을 위해 clusterIP에서 nodePort로 변경
- http://nodeIP:nodePort로 접속하여 초기 화면 확인
- prometheus 메트릭 활용을 위해 grafana prometheus plugin 추가