



챕터 6 쿠버네티스 리소스 로깅과 모니터링

Created

2021년 12월 22일 오전 2:02

Tags

비어 있음

클러스터 컴포넌트 모니터링

메트릭스 서버 깃헙



metrics-server
kubernetes-sigs

메트릭스 서버 설치

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/download/v0.5.2/components.yaml
```

메트릭스 서버 수정

```
kubectl edit deployments.apps -n kube-system metrics-server
```

args에 다음 내용 추가

```
--kubelet-insecure-tls
```

다음 명령으로 정상적으로 동작하는지 테스트

```
# kubectl top nodes NAME CPU(cores) CPU% MEMORY(bytes) MEMORY% gasbugs-01 269m 13% 1320Mi 34% gasbugs-02 384m 19% 2488Mi 64% gasbugs-03 208m 10% 2860Mi 74% gasbugs-04 186m 9% 2634Mi 68% # kubectl top pod NAME CPU(cores) MEMORY(bytes) hello-web 1m 1Mi http-go-v1 0m 6Mi http-go-v2 0m 5Mi http-go-v3 1m 4Mi py 0m 28Mi security-context-demo 0m 0Mi security-context-demo-4 0m 9Mi test-2 0m 0Mi
```

쿠버네티스 대시보드 설치와 활용

<https://kubernetes.io/ko/docs/tasks/access-application-cluster/web-ui-dashboard/>

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.4.0/aio/deploy/recommended.yaml
```

쿠버네티스 대시보드가 잘 배포되었는지 확인한다.

```
# kubectl get svc -n kubernetes-dashboard NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE dashboard-metrics-scraper ClusterIP 10.98.95.126 <none> 8000/TCP 33s kubernetes-dashboard ClusterIP 10.101.111.210 <none> 443/TCP 33s
```

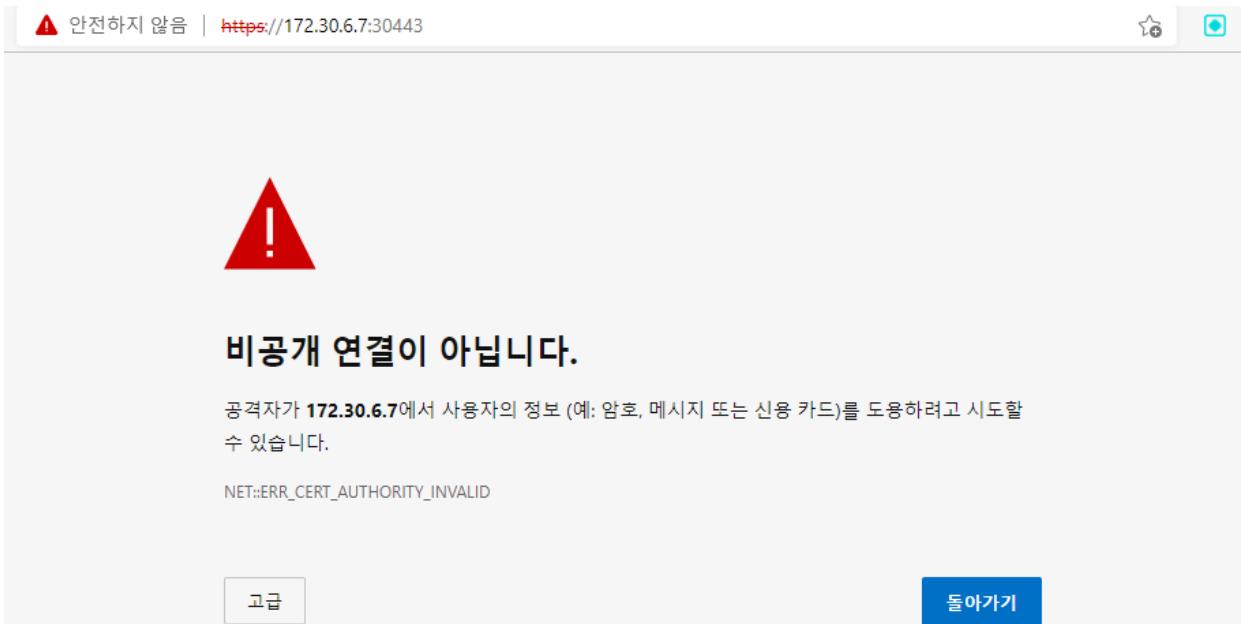
ClusterIP → NodePort로 변경해 외부에서 접근이 가능하도록 만들자. nodePort 옵션도 하나 추가하고 30443로 구성한다.

```
kubectl edit svc kubernetes-dashboard -n kubernetes-dashboard
```

마스터노드의 IP를 사용해 30443으로 접속해보자.

```
https://172.30.6.7:30443
```

고급 옵션으로 가서 (만약 고급 버튼이 없다면 페이지를 클릭하고 thisisunsafe라고 타이핑) 안전하지 않음을 클릭한다.



접속을 완전히 수행하면 대시보드활용되는 토큰을 발급 받으라고 나타난다.

쿠버네티스 대시보드

토큰
모든 서비스 어카운트는 시크릿을 가지고 있고, 시크릿에는 대시보드에 로그인할 때 사용할 수 있는 유효한 베어러(Bearer) 토큰이 있습니다. 베어러(Bearer) 토큰을 설정 및 사용하는 방법은 [인증](#) 섹션에서 확인할 수 있습니다.

Kubeconfig
클러스터에 접근을 설정하기 위해 생성한 kubeconfig 파일을 선택하세요. kubeconfig 파일을 설정 및 사용하기 위한 방법은 [멀티 클러스터에 접근 설정하기](#) 섹션에서 확인할 수 있습니다.

토큰 입력 *

로그인

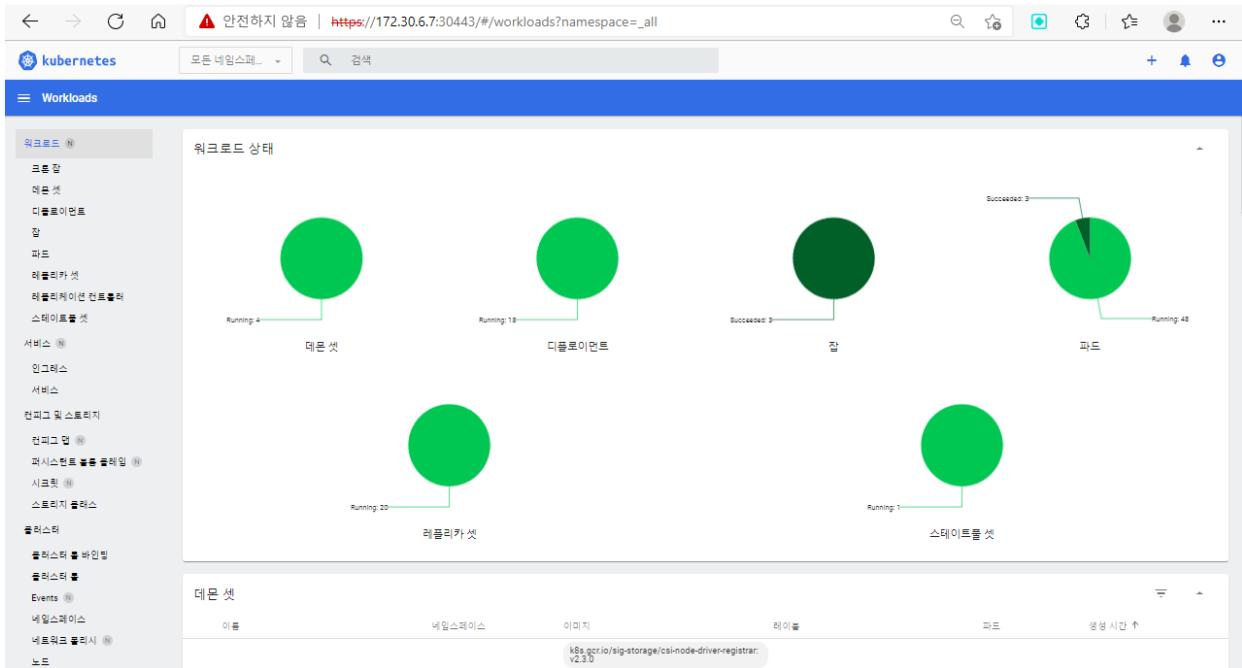
샘플 유저를 전체 클러스터 권한을 다 가지기에 주의를 요한다.

<https://github.com/kubernetes/dashboard/blob/master/docs/user/access-control/creating-sample-user.md>

```
cat <<EOF | kubectl apply -f - # 서비스어카운트 생성
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
--- # 클러스터를 바인딩 생성
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: ClusterRole
    name: cluster-admin
  subjects:
  - kind: ServiceAccount
    name: admin-user
  namespace: kubernetes-dashboard
EOF # 토큰 가져오기
kubectl -n kubernetes-dashboard get secret $(kubectl -n kubernetes-dashboard get sa/admin-user -o jsonpath='{.secrets[0].name}') -o go-template="{{.data.token | base64decode}}"
```

가져온 토큰을 사용해 로그인을 수행한다.

쿠버네티스 도큐먼트에서 지원하는 대시보드 설치가 끝났다.



프로메테우스 설치와 모니터링

프로메테우스와 그라파나를 위한 헬름 저장소를 추가

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

헬름 배포를 위해 그라파나와 프로메테우스의 values.yaml을 구성할 딕션리를 하나 구성

```
mkdir grafana_prometheus cd grafana_prometheus
```

다음 명령을 실행해 values-prometheus.yaml을 생성

```
cat <<EOF > values-prometheus.yaml
server:
  enabled: true
  persistentVolume:
    enabled: true
    accessModes:
      - ReadWriteOnce
    mountPath: /data
    size: 10Gi
    replicaCount: 1
    ## Prometheus data retention period (default if not specified is 15 days)
    ## retention: "15d"
EOF
```

다음 명령을 실행해 values-grafana.yaml을 생성

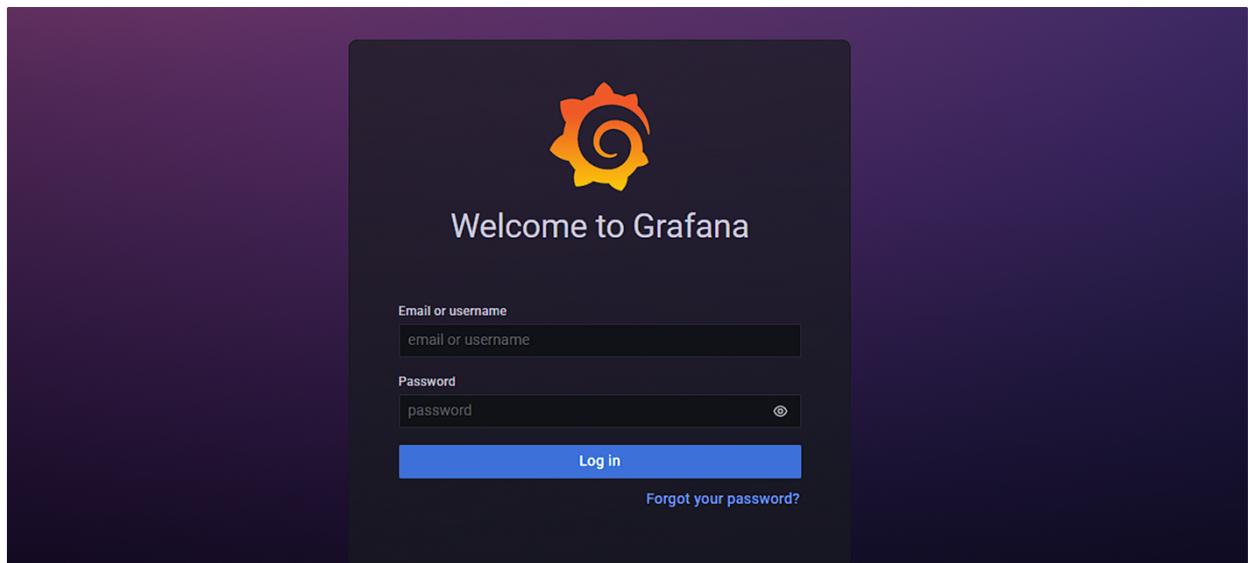
pvc를 구성하여 스토리지를 구성하여 설정정보를 유지할 수 있도록 구성

```
cat << EOF > values-grafana.yaml replicas: 1 service: type: NodePort persistence: type: pvc enabled: true # storageClassName: default accessModes: - ReadWriteOnce size: 10Gi # annotations: {} finalizers: - kubernetes.io/pvc-protection # Administrator credentials when not using an existing secret (see below) adminUser: admin adminPassword: test1234!234 EOF
```

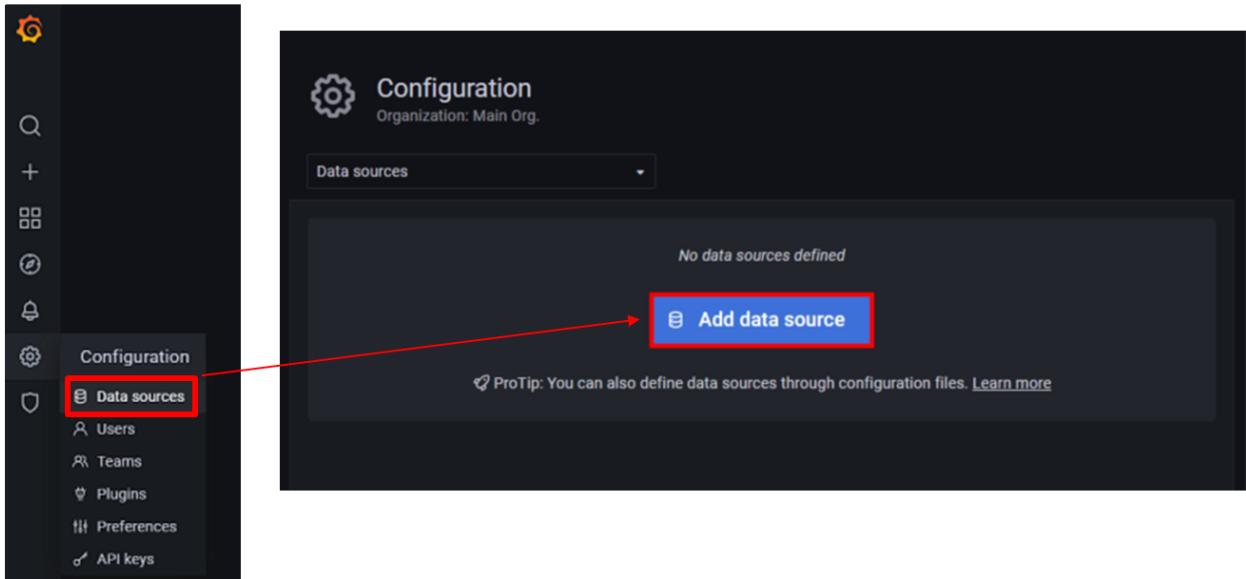
프로메테우스를 위한 네임스페이스를 구성하고 위 설정대로 yaml 파일을 배포

```
kubectl create ns prometheus helm install prometheus prometheus-community/prometheus -f values-prometheus.yaml -n prometheus helm install grafana grafana/grafana -f values-grafana.yaml -n prometheus
```

아이디와 패스워드는 admin//test1234!234



왼쪽 메뉴에서 Configuration – Data Sources 메뉴에서 데이터 소스 추가

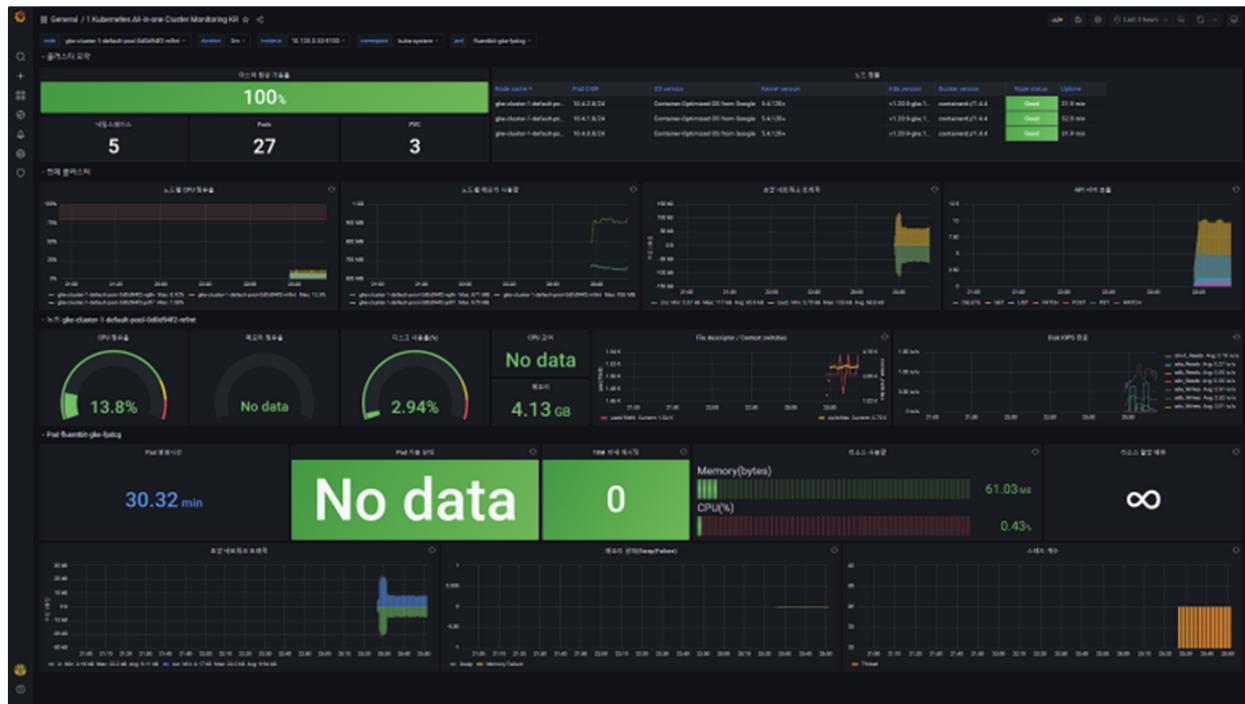


프로메테우스를 선택하고 URL에 앞서 자동 생성된 service의 이름을 입력

http://prometheus-server 을 입력하고 save & test 클릭

The screenshot shows the 'Add data source' configuration screen for Prometheus. The 'Prometheus' option is selected and highlighted with a red box. The 'Name' field is set to 'Prometheus' and has a 'Default' toggle switch turned on. The 'HTTP' section contains the 'URL' field with the value 'http://prometheus-server', which is also highlighted with a red box. Other fields in the 'HTTP' section include 'Access' (set to 'Server (default)'), 'Whitelisted Cookies' (with a placeholder 'New tag (enter key to add)'), and 'Timeout' (empty).

13770번 대시보드



Istio를 활용한 네트워크 모니터링

istioctl 설치

```
curl -L https://istio.io/downloadIstio | sh - cd istio-1.12.1 export PATH=$PWD/bin:$PATH # 실행 경로를 환경 변수에 추가 istioctl # kubectl 설정을 사용
```

istio를 쿠버네티스에 설치하고 디플트 네임스페이스에 적용

```
istioctl install --set profile=default --skip-confirmation kubectl label namespaces default istio-injection=enabled
```

북인포 프로젝트 배포

```
kubectl delete all --all # 잘못 설치한 경우 삭제 kubectl delete limitrange default-limit-range # 잘못 설치한 경우 삭제 kubectl delete -f samples/bookinfo/platform/kube/bookinfo.yaml # 잘못 설치한 경우 삭제 kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml
```

ingress gateway와 북인포 프로젝트 연결을 수행한다. 이 과정은 gateway를 만들어서 구성이 되는데 게이트웨이의 룰에 의해 어떻게 로드밸런싱 할지 결정된다.

```
kubectl apply -f samples/bookinfo/networking/bookinfo-gateway.yaml kubectl get svc -n istio-system -l istio=ingressgateway
```

대시보드와 데이터베이스를 설치하고 서비스를 오픈한다.

```
kubectl apply -f samples/addons/kiali.yaml kubectl apply -f samples/addons/prometheus.yaml istioctl dashboard kiali # localhost:20001 서비스를 오픈
```

EFK를 활용한 k8s 로그 모니터링

efk.zip 1.9KB

디렉토리 생성 및 파일 압축 해제

```
mkdir efk cd efk <file download> # wget https://blogattach.naver.com/54c148f8ebb7b06c40a2c7f4cc2e552b86db22c572/20211003_33_blogfile/isc0304_1633192930812_gYBc80_zip/efk.zip unzip efk.zip kubectl apply -f ns.yaml kubectl apply -f ./*
```

엘라스틱서치 서비스 확인

```
kubectl get svc -n elastic
```

오토스케일링 HPA 워크스루

<https://kubernetes.io/ko/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/>

Jaeger 트레이싱 튜토리얼

예거 설치하기

```
docker run -d --name jaeger -p 16686:16686 \ -p 6831:6831/udp \ jaegertracing/all-in-one:1.22 apt install python3-pip -y pip3 install jaeger-client pip3 install requests
```

정보를 받아오는 주소

<http://ip-api.com/json/naver.com>

python을 실행하고 다음 명령을 실행

python-jaeger-example.py

```
import logging from jaeger_client import Config import requests def init_tracer(service): logging.getLogger('').handlers = [] logging.basicConfig(format='%(message)s', level=logging.DEBUG) config = Config(config={ 'sampler': { 'type': 'const', 'param': 1, }, 'logging': True, }, service_name=service, ) # this call also sets opentracing.tracer return config.initialize_tracer() tracer = init_tracer('first-service') with tracer.start_span('get-ip-api-jobs') as span: try: res = requests.get('http://ip-api.com/json/naver.com') result = res.json() print('Getting status %s' % result['status']) span.set_tag('jobs-count', len(res.json())) for k in result.keys(): span.set_tag(k, result[k]) except: print('Unable to get site for') input()
```