DATADOG Webinar

Kubernetes Monitoring

About speaker

정영석 (Jacky Jung)

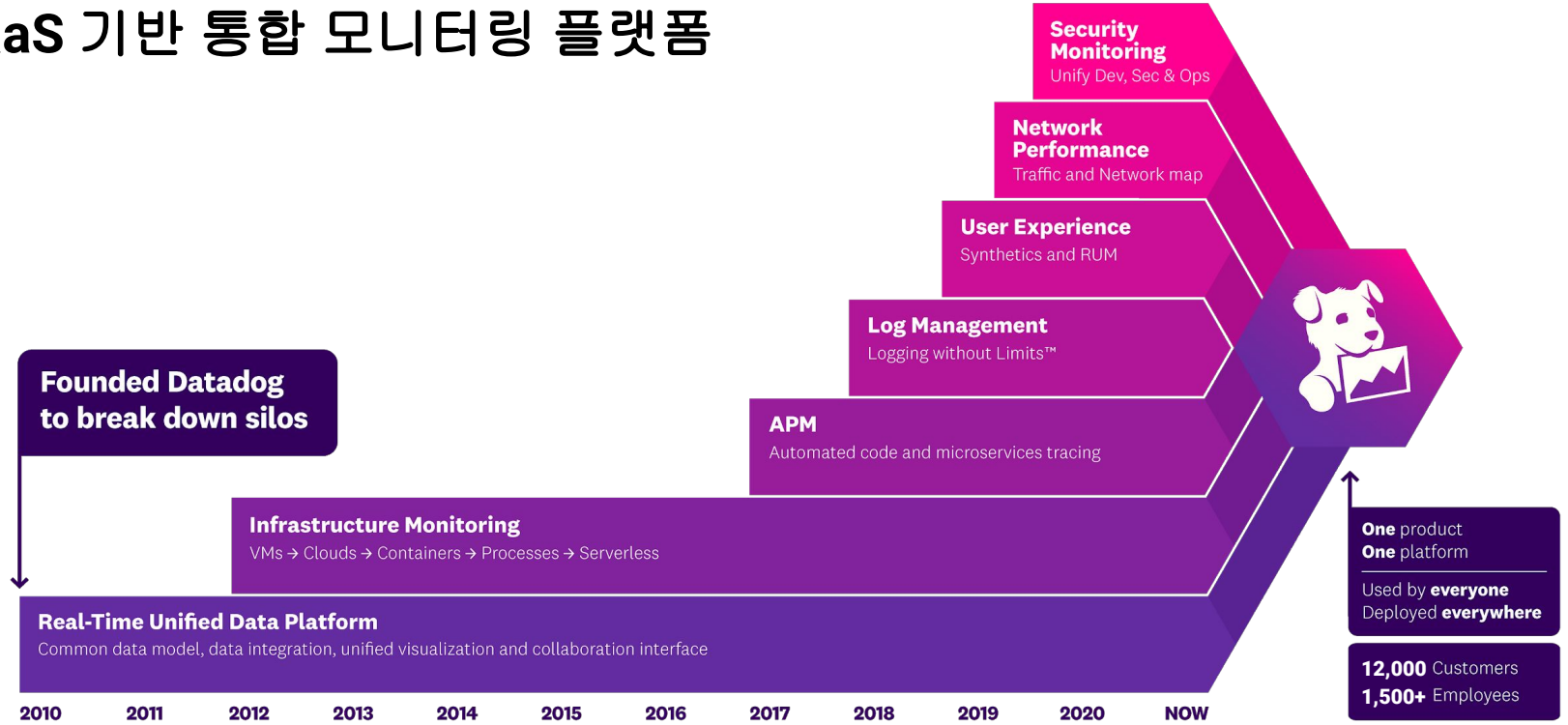
What do I do?

- **Pre-Sales 엔지니어 @Datadog (현재)**
- CDN 서비스 아키텍트
- Devops 엔지니어
- 개발자

DATADOG

Datadog 소개

SaaS 기반 통합 모니터링 플랫폼



DATADOG

오늘 살펴볼 내용

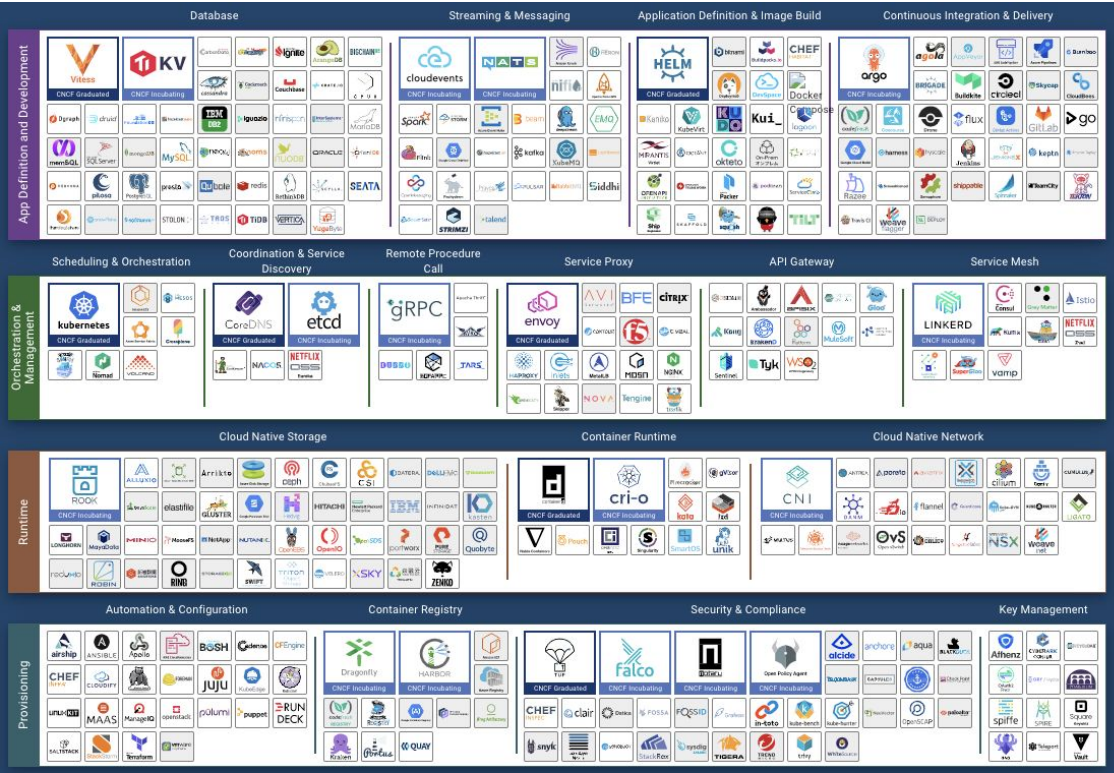
1. Kubernetes Monitoring Challenges

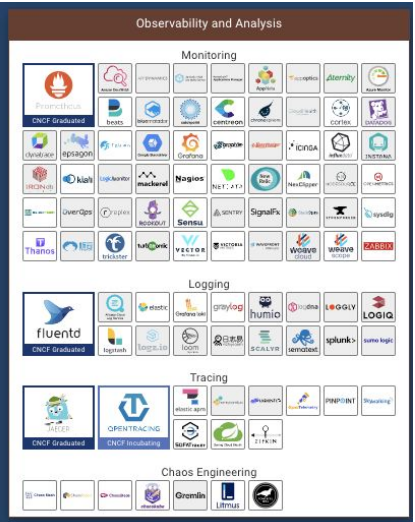
2. Kubernetes Monitoring Best Practice

3. Kubernetes Sample APP의 Full Stack 가시성 확보해보기 (Demo)

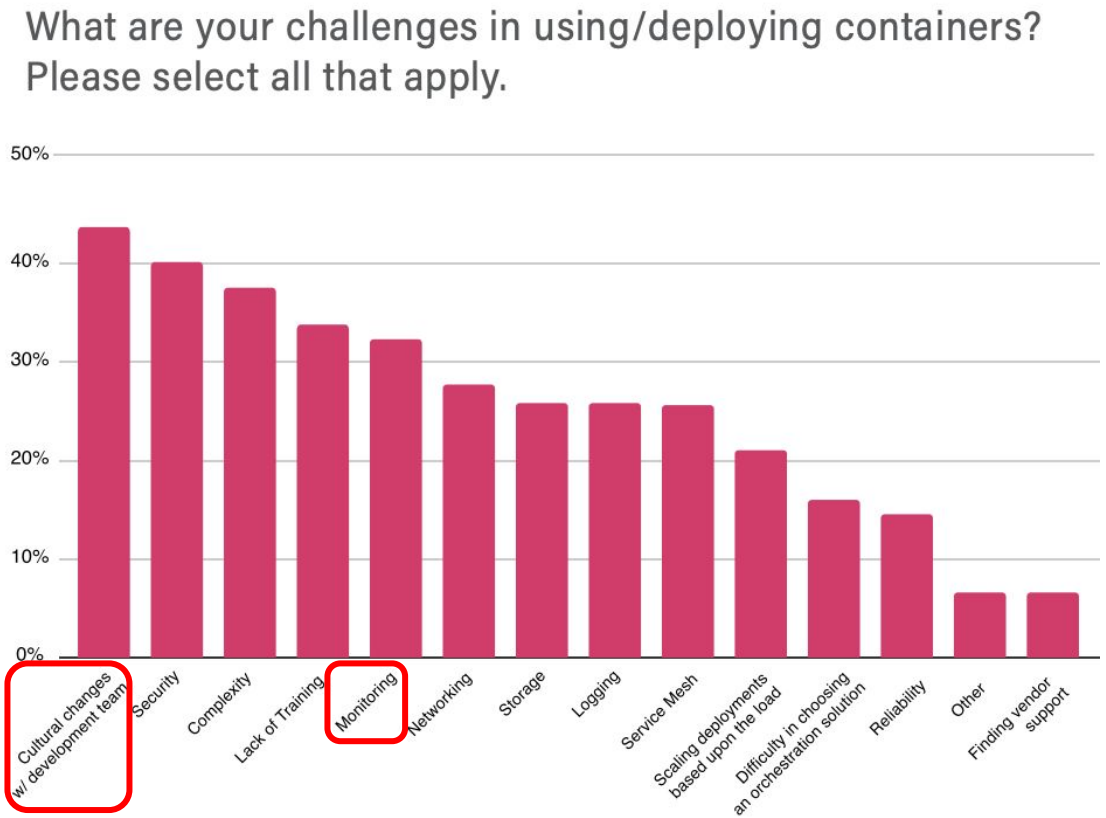
CNCF Landscape

1407 개의 CARD





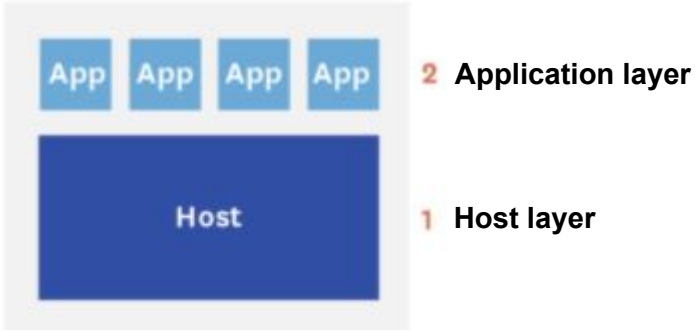
CNCF SURVEY 2019 9월 ~ 10월 조사



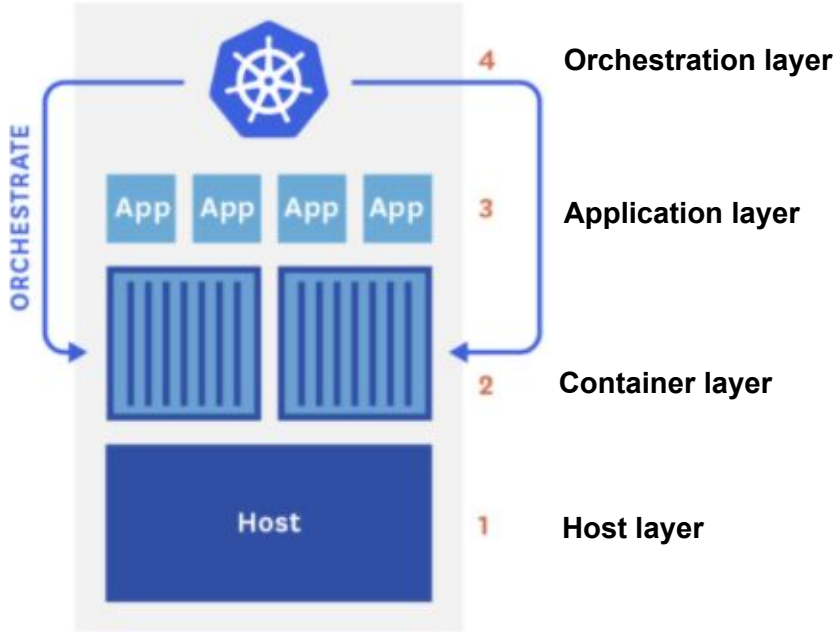
- **1337명 응답**
- **Monitoring이 Top5 Challenge**

Kubernetes Monitoring Challenges

1. 모니터링해야 할 컴포넌트가 많다

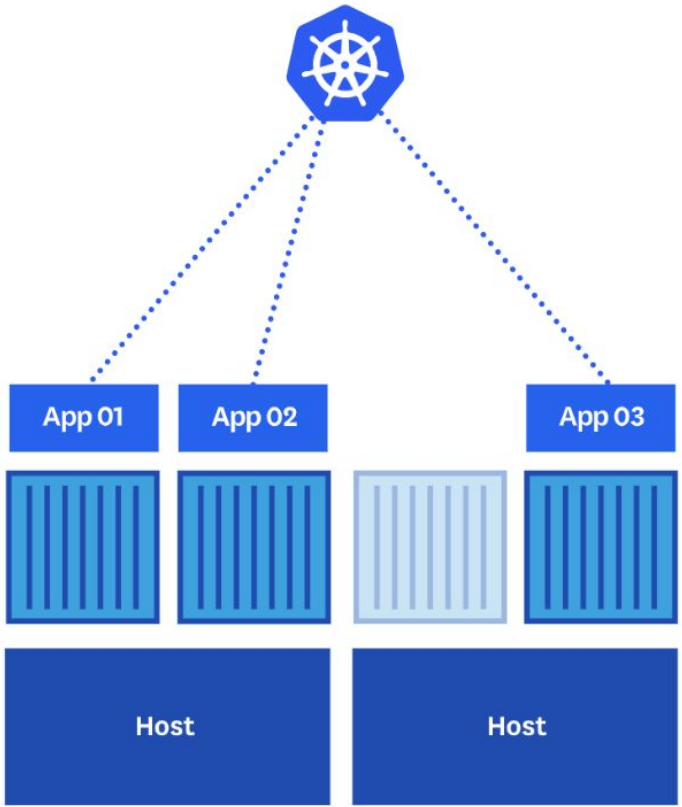


전통적인 운영환경



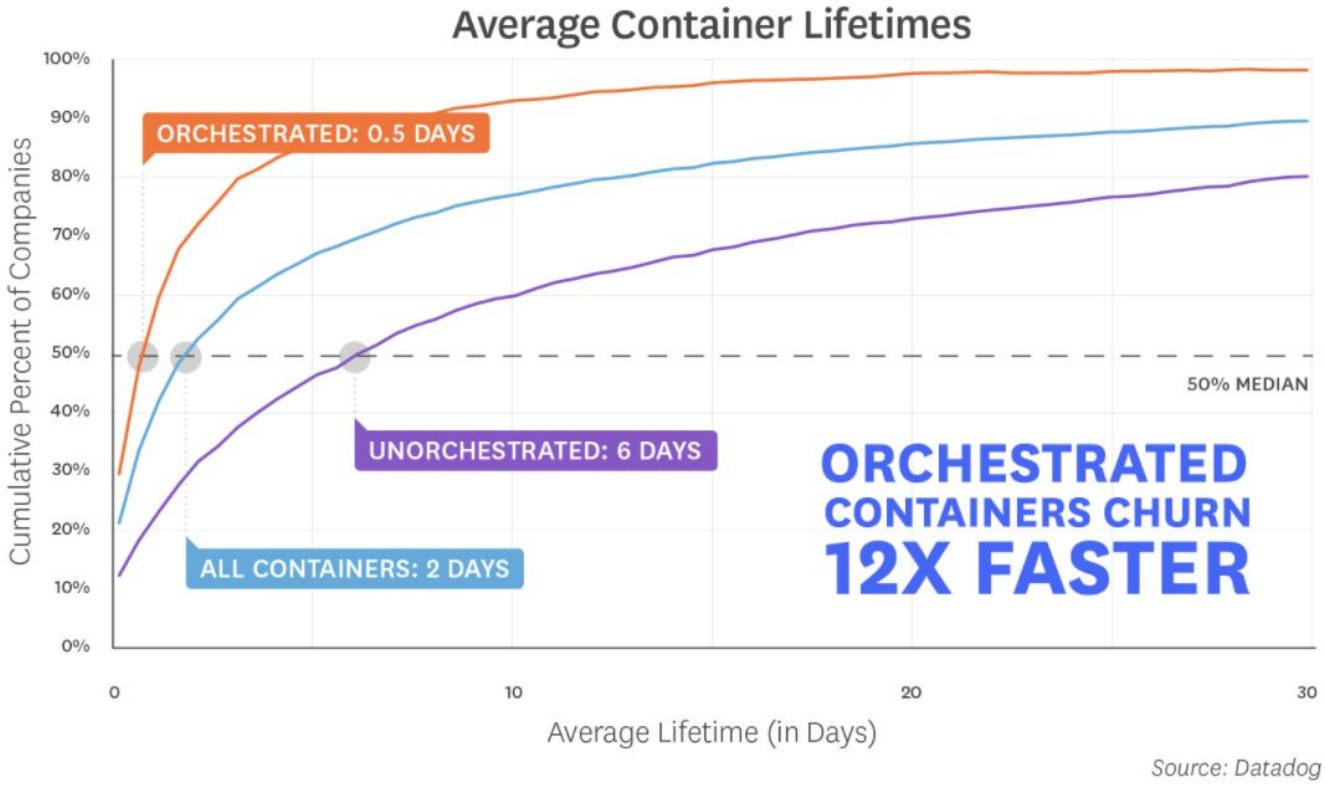
Kubernetes 운영환경

2. Life cycle이 짧은 컨테이너 모니터링



- Kubernetes 환경에서의 컨테이너는 매우 짧은 주기로 생성/삭제되어 모니터링을 더욱 어렵게 함
- Kubernetes 환경에서 컨테이너 당 평균 Lifetime은 약 0.5일로 리포트됨 (Datadog report)

2. Life cycle이 짧은 컨테이너 모니터링



3. 모니터링을 위해 다양한 솔루션 운영 필요

3. 모니터링을 위해 다양한 솔루션 운영 필요

Metric

Application

PINPOINT

Logging

Challenge

1. 모니터링 솔루션 및 Storage 운영 필요

2. Scale에 대한 고민 필요

3. 모니터링 환경이 파편화되어 연계 분석의 어려움

Kubernetes Monitoring Best Practice

Datadog ❤ Kubernetes

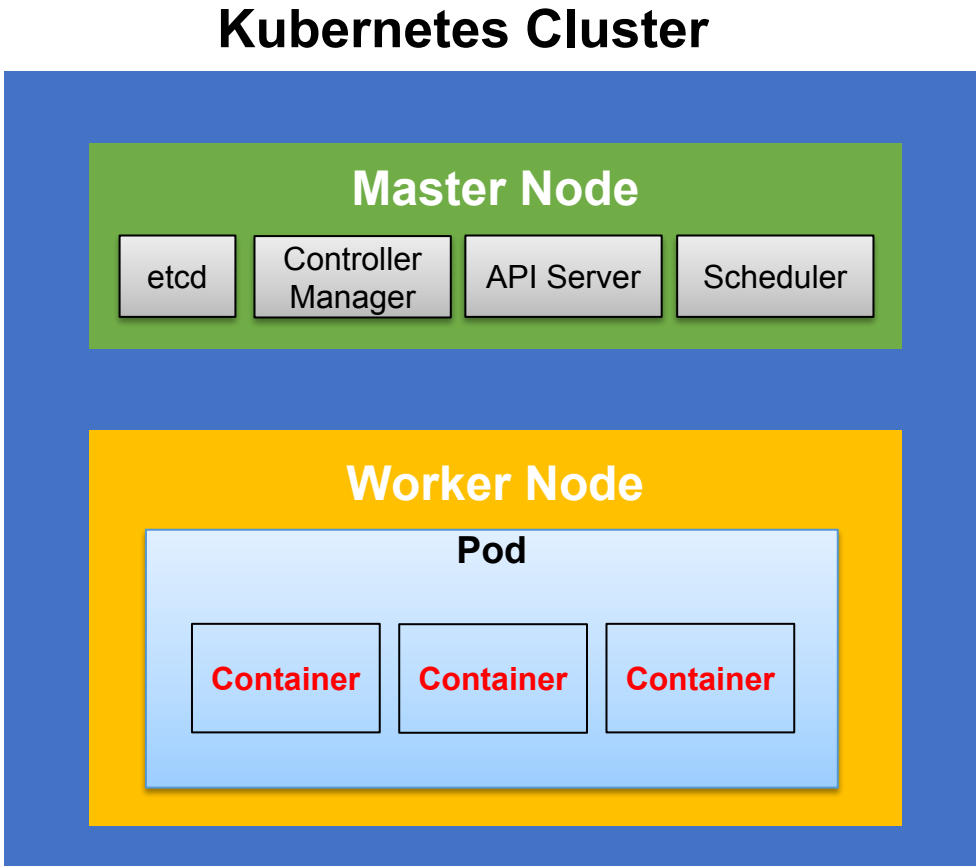
- 하루 수 조(兆)개의 Data point 생성
- Multi-Cloud 환경에서 수 십개의 Kubernetes Cluster 운영 중
- 수 만대의 노드에서 Kubernetes 운영 중

 # Kubernetes the hard way at datadog

무엇을 모니터링 해야 하나?

1. **Cluster Node**
2. **Control Plane**
3. **Pod/Container**
4. **Kubernetes Event**
5. **Application의 Trace 수집**
6. **Log**

1. Infrastructure (Nodes)



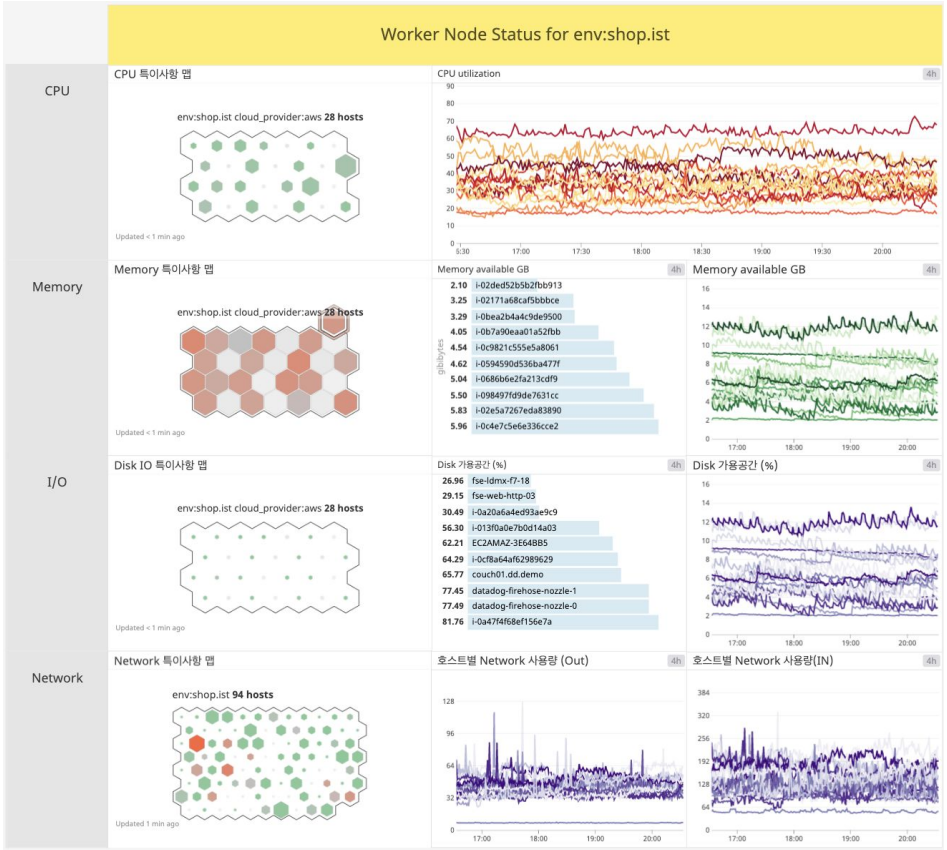
주요 모니터링 항목

1. Node별 CPU 사용량
2. Node별 Memory 사용량
3. Node별 Disk 사용량
4. Node별 Network 사용량
5. Node Status

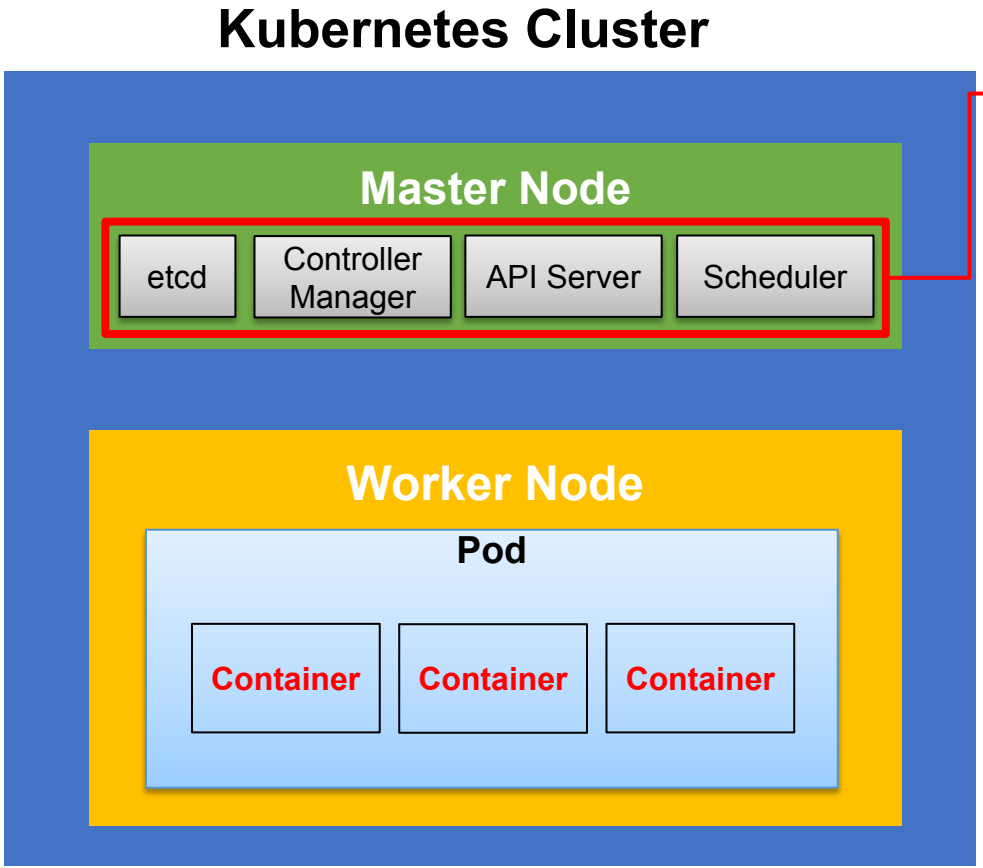
1. Infrastructure (Nodes) - Sample 대시보드

주요 모니터링 항목

1. Node별 CPU 사용량
2. Node별 Memory 사용량
3. Node별 Disk 사용량
4. Node별 Network 사용량



2. Control Plane 모니터링



주요 모니터링 항목

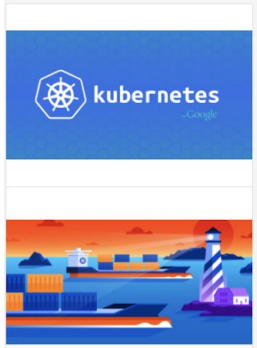
1. API Server
   - ex: drop된 request의 빈도 트래킹

2. etcd
   - ex: 저장된 object count, byte 전송량 트래킹

3. scheduler
   - ex: scheduler 요청 응답시간

4. Controller Manager
   - ex: queue depth

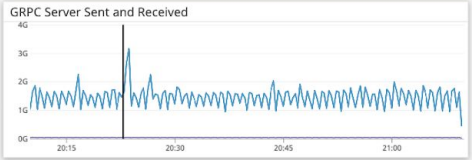
2. Control Plane 모니터링



ETCD

The key-value store for coordination and the state management of the cluster.

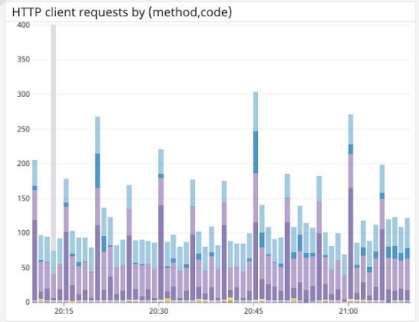
etcd has leader	Leader Changes
1	0
Failed Server P...	Pending Server...
0	727
Go Threads	Server Handle %
21	100%

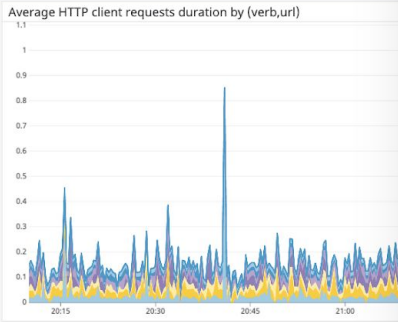


Scheduler

Schedules the deployment of pods by scouting for unscheduled pods and spreading them across available clusters.

Go Routines	Go Threads	HTTP Requests
132	18	80

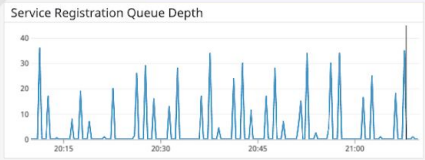


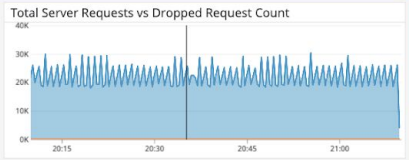


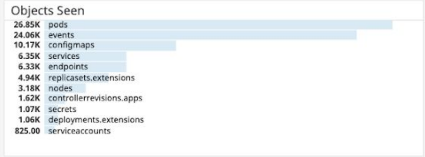
API Server

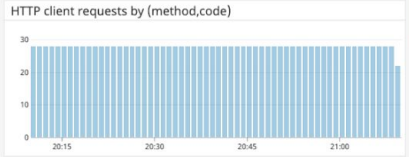
The communication hub of the Kubernetes cluster, listening and dispatching requests from the other components of the plane.

Go Threads	Go Routines	Client Requests
21	23.94K	243





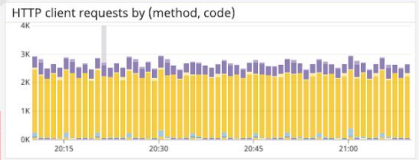


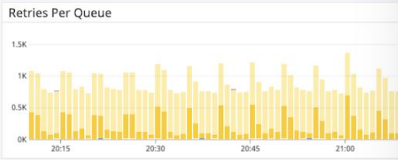


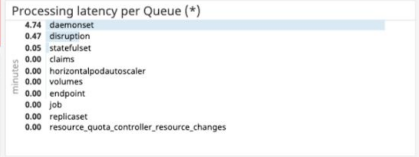
Controller Manager

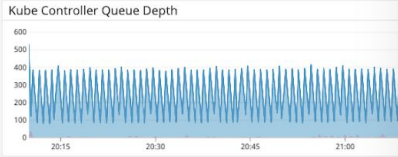
Runs control loops to monitor the current state of the cluster and takes action to move it towards the desired state.

Healthy Nodes	Unhealthy Nodes	Go Routines
1.04K	2	937

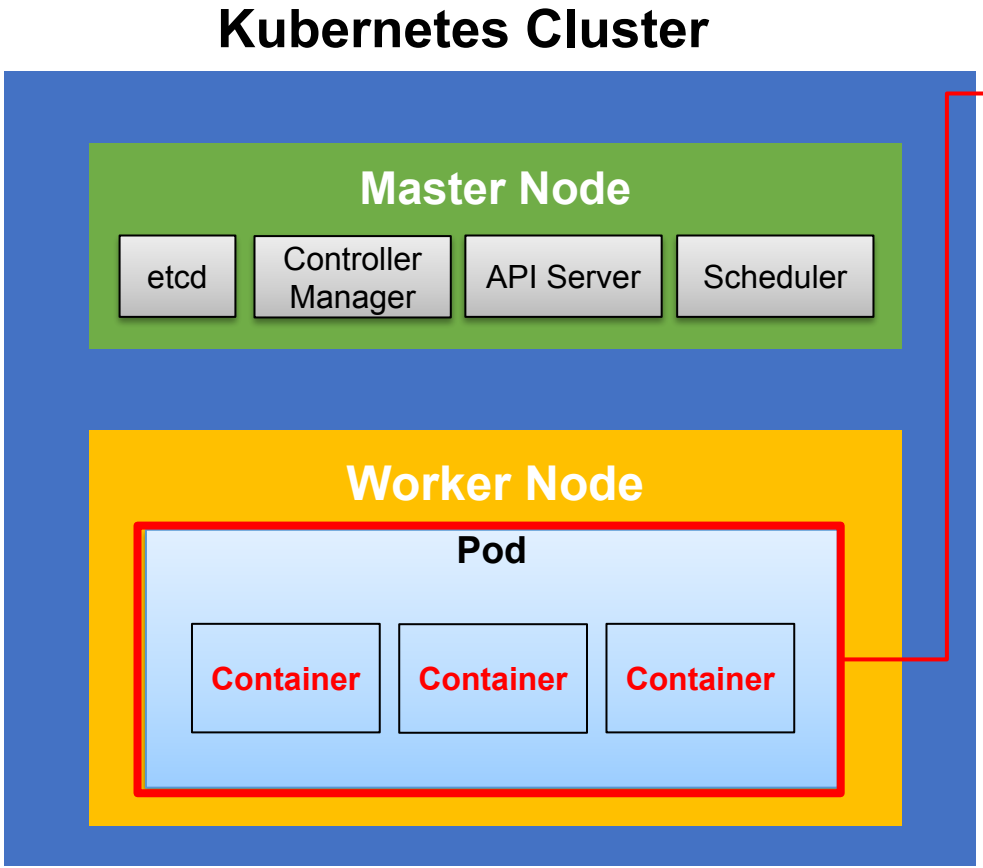








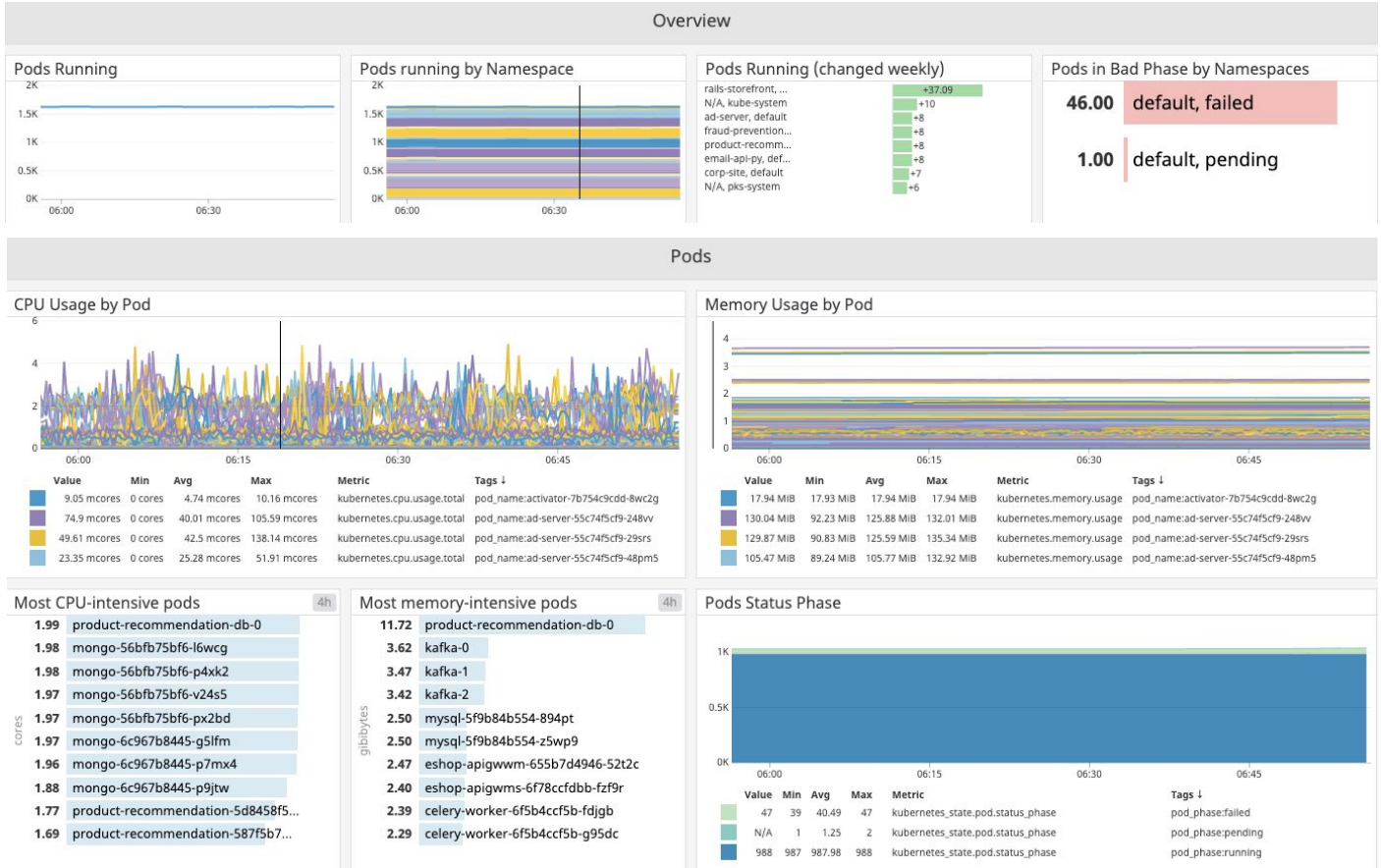
3. Pod 모니터링



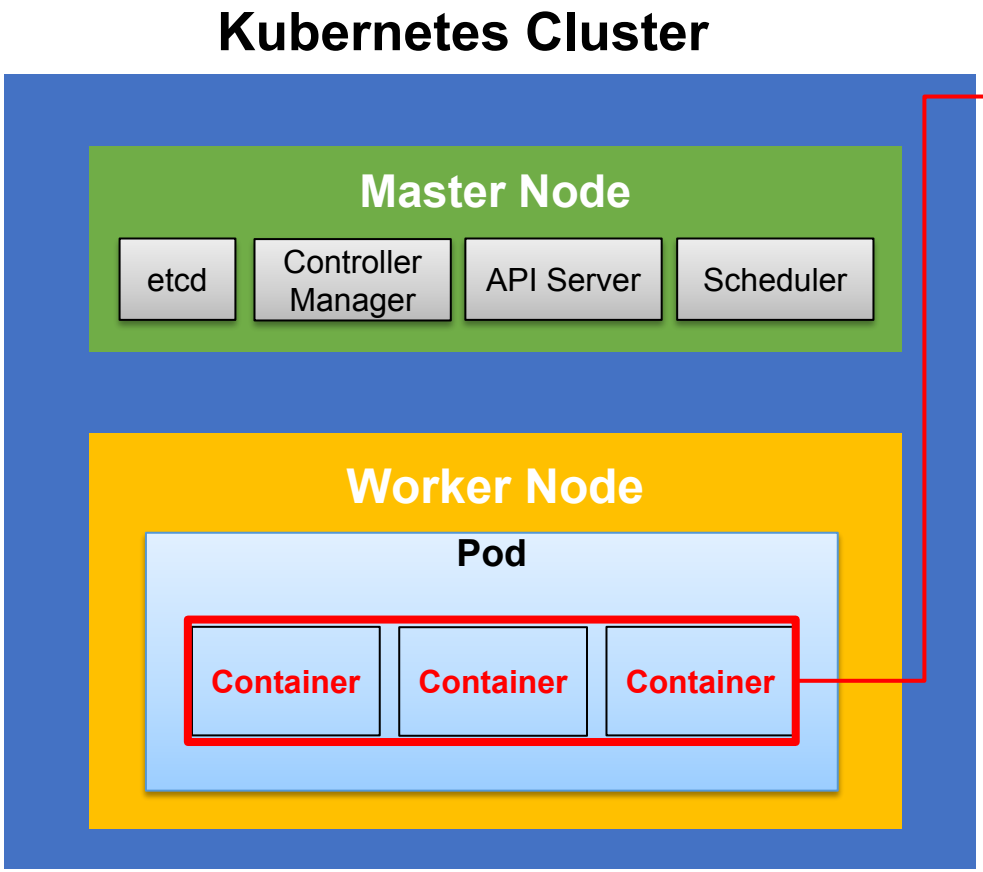
주요 모니터링 항목

1. CPU/Memory 사용량이 높은 Pod
2. Pod Phase (Pending/Running/Failed)
3. Pod eviction

3. Pod 모니터링


Overview
Pods Running
Pods running by Namespace
Pods Running (changed weekly)
rails-storefront, ... +37.09
N/A, kube-system +10
ad-server, default +8
fraud-prevention... +8
product-recomm... +8
email-api-py, def... +8
corp-site, default +7
N/A, pks-system +6
Pods in Bad Phase by Namespaces
46.00 default, failed
1.00 default, pending
Pods
CPU Usage by Pod
Value Min Avg Max Metric Tags
9.05 mcores 0 cores 4.74 mcores 10.16 mcores kubernetes.cpu.usage.total pod_name:activator-7b754c9cdd-8wc2g
74.9 mcores 0 cores 40.01 mcores 105.59 mcores kubernetes.cpu.usage.total pod_name:ad-server-55c74f5cf9-248vv
49.61 mcores 0 cores 42.5 mcores 138.14 mcores kubernetes.cpu.usage.total pod_name:ad-server-55c74f5cf9-29srs
23.35 mcores 0 cores 25.28 mcores 51.91 mcores kubernetes.cpu.usage.total pod_name:ad-server-55c74f5cf9-48pm5
Memory Usage by Pod
Value Min Avg Max Metric Tags
17.94 MiB 17.93 MiB 17.94 MiB 17.94 MiB kubernetes.memory.usage pod_name:activator-7b754c9cdd-8wc2g
130.04 MiB 92.23 MiB 125.88 MiB 132.01 MiB kubernetes.memory.usage pod_name:ad-server-55c74f5cf9-248vv
129.87 MiB 90.83 MiB 125.59 MiB 135.34 MiB kubernetes.memory.usage pod_name:ad-server-55c74f5cf9-29srs
105.47 MiB 89.24 MiB 105.77 MiB 132.92 MiB kubernetes.memory.usage pod_name:ad-server-55c74f5cf9-48pm5
Most CPU-intensive pods 4h
cores
1.99 product-recommendation-db-0
1.98 mongo-56bfb75bf6-l6wcg
1.98 mongo-56bfb75bf6-p4xk2
1.97 mongo-56bfb75bf6-v24s5
1.97 mongo-56bfb75bf6-px2bd
1.97 mongo-6c967b8445-g5lfm
1.96 mongo-6c967b8445-p7mx4
1.88 mongo-6c967b8445-p9jtw
1.77 product-recommendation-5d8458f5...
1.69 product-recommendation-587f5b7...
Most memory-intensive pods 4h
gibibytes
11.72 product-recommendation-db-0
3.62 kafka-0
3.47 kafka-1
3.42 kafka-2
2.50 mysql-5f9b84b554-894pt
2.50 mysql-5f9b84b554-z5wp9
2.47 eshop-apigwwm-655b7d4946-52t2c
2.40 eshop-apigwms-6f78ccfdbb-fzf9r
2.39 celery-worker-6f5b4ccf5b-fdjgb
2.29 celery-worker-6f5b4ccf5b-g95dc
Pods Status Phase
Value Min Avg Max Metric Tags
47 39 40.49 47 kubernetes_state.pod.status_phase pod_phase:failed
N/A 1 1.25 2 kubernetes_state.pod.status_phase pod_phase:pending
988 987 987.98 988 kubernetes_state.pod.status_phase pod_phase:running

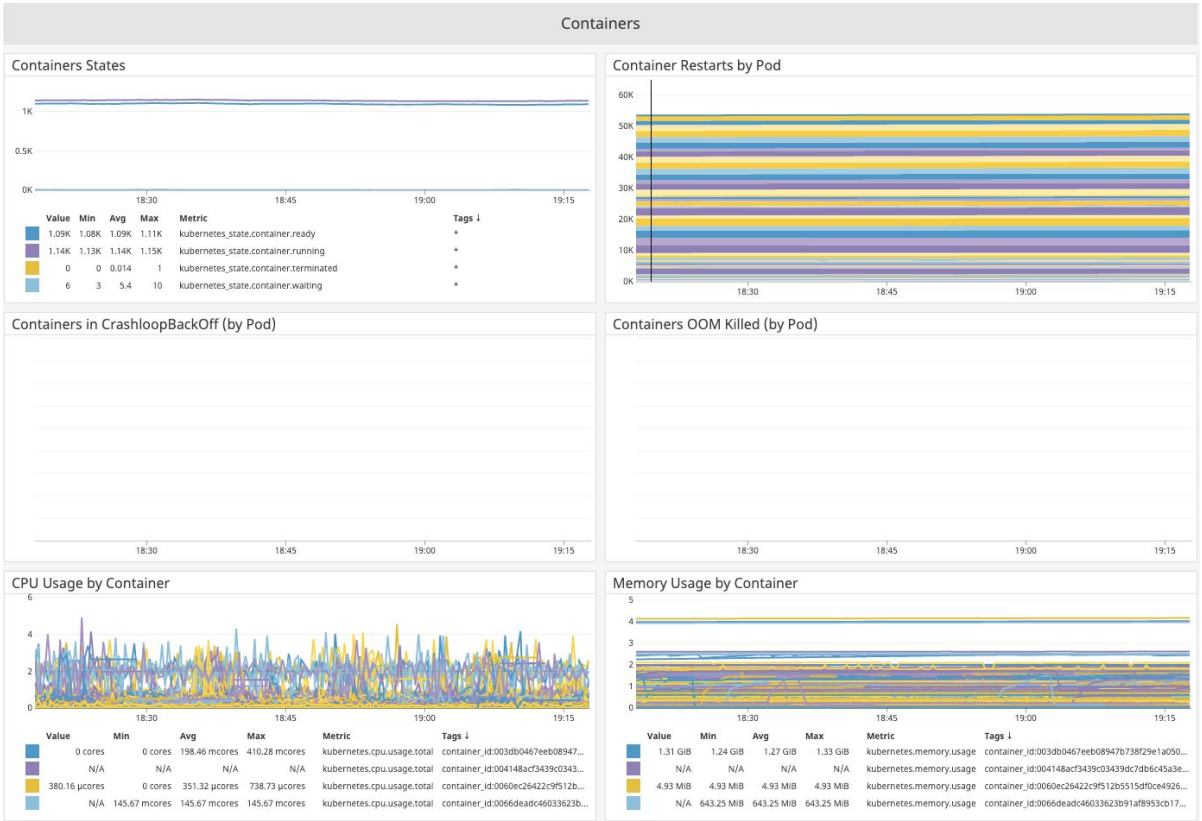
4. Container 모니터링



주요 모니터링 항목

1. Container 별 Resource 사용량
2. OOM Killed 빈도
3. CrashLoopBackOff 빈도
4. Restart 빈도

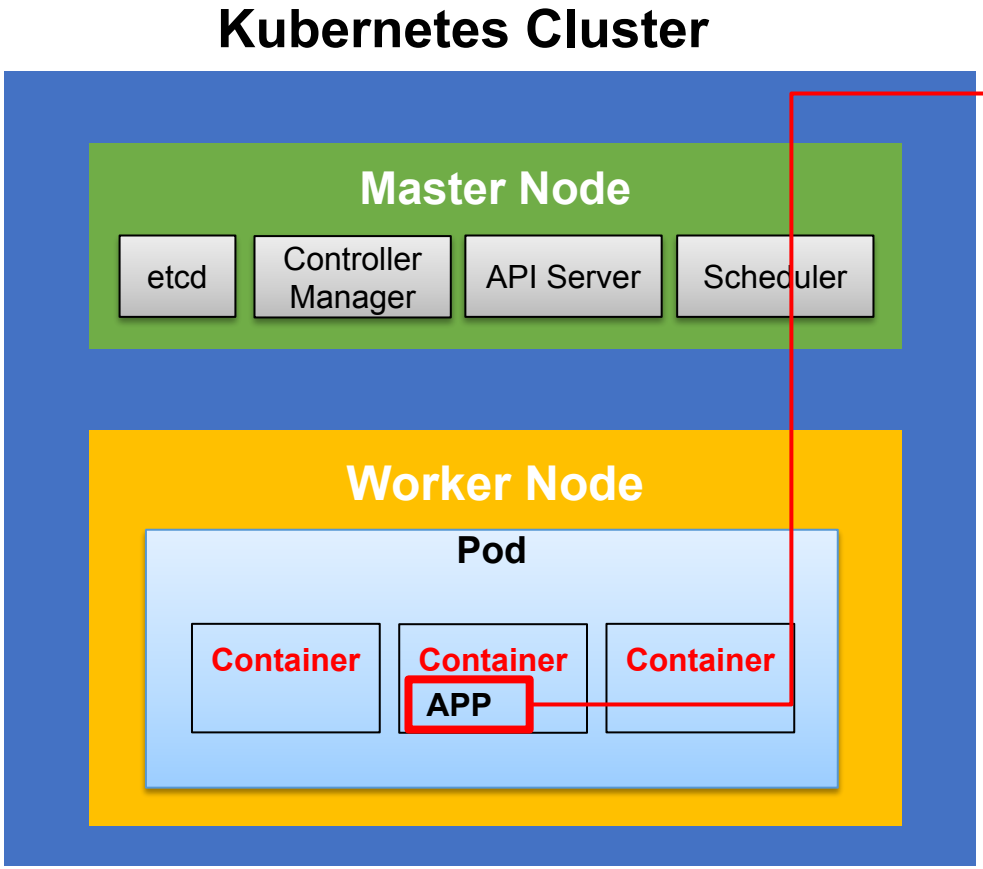
4. Container 모니터링



주요 모니터링 항목

1. Container 별 Resource 사용량
2. OOM Killed 빈도
3. CrashoopBackOff 빈도
4. Restart 빈도

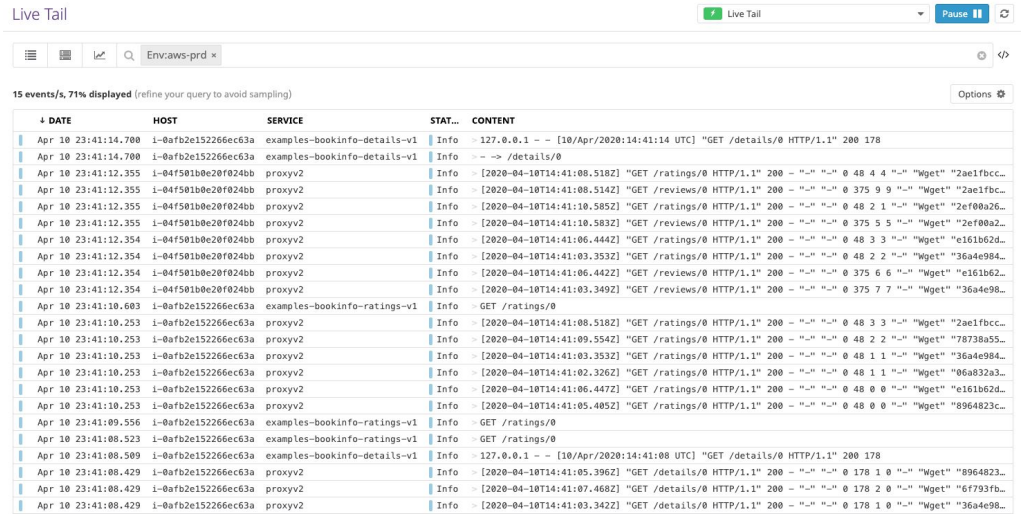
5. Application 모니터링



주요 모니터링 항목

1. Request 빈도
2. Request Latency (Slow API endpoint)
3. Request Error
4. Garbage collection
5. DB Slow query

6. Log 모니터링



주요 모니터링 항목

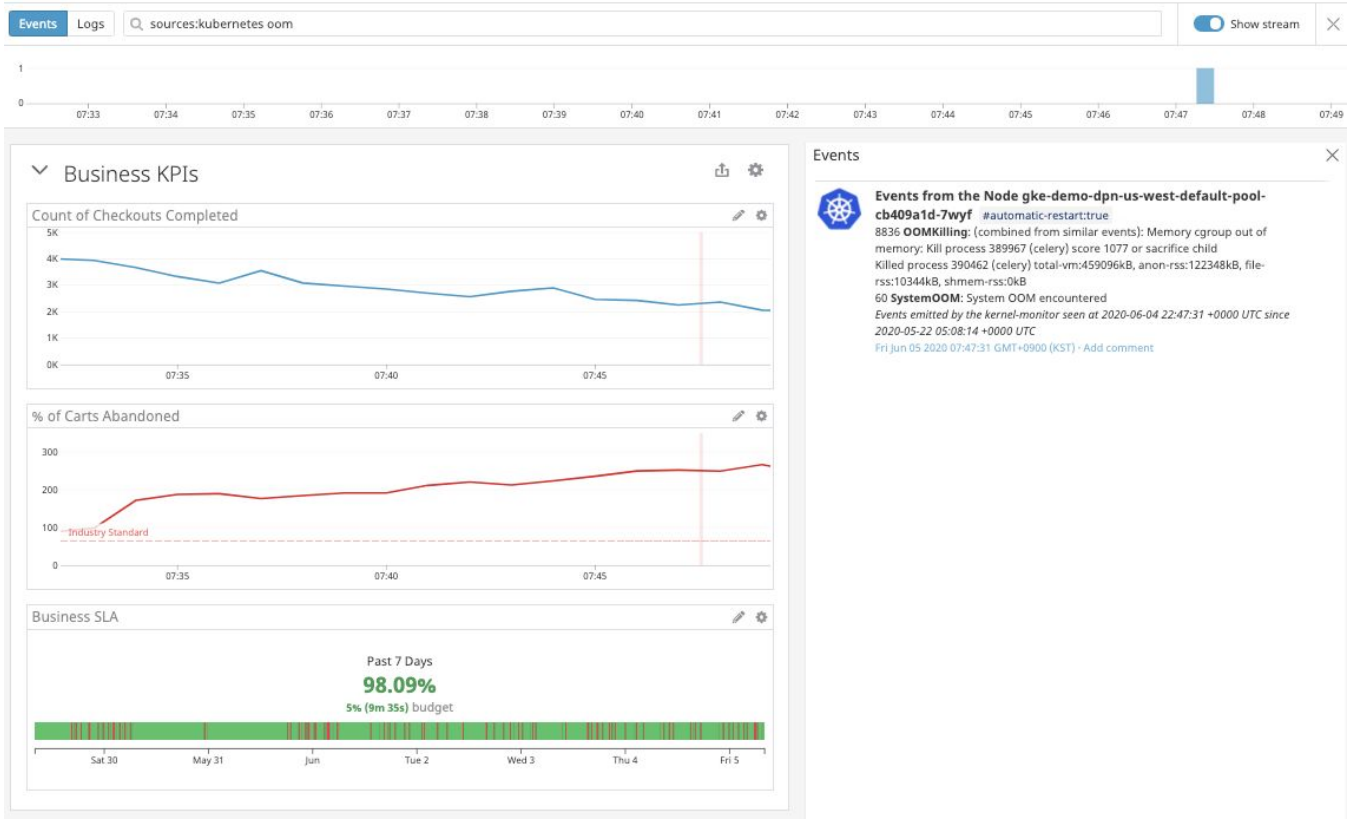
1. 어플리케이션 Access/Error 로그
2. 솔루션 로그 (ex: Redis, MySQL)
3. Ingress 로그
4. Cluster Audit 로그 (API Server log)

7. Event 모니터링

Event 주요 모니터링 항목

1. Resource 부족으로 인한 Pod 생성 실패
2. Unhealthy
3. Eviction
4. OOMKilling

7. Event 모니터링 (OOM Event와 연계 분석 예)



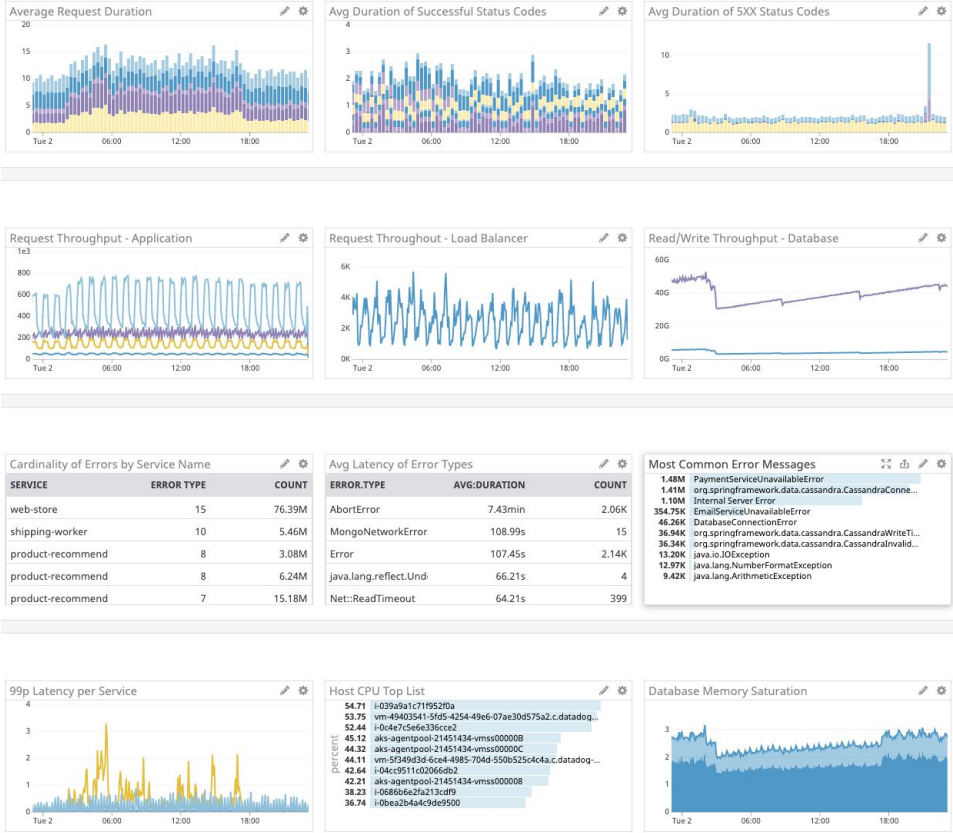
The Four Golden Signals

1. **Latency**
   - API 별 응답시간 트래킹

2. **Traffic**
   - 초당 처리 트랜잭션

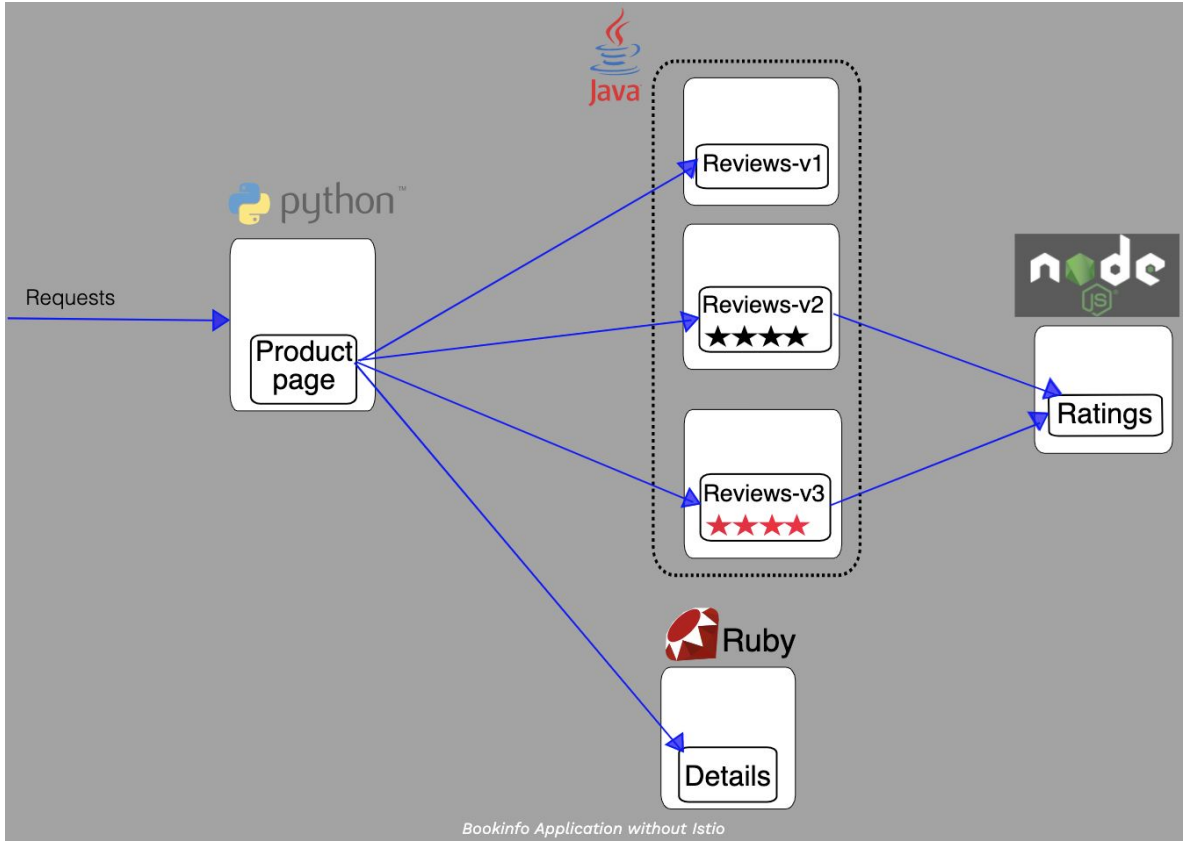
3. **Errors**
   - 에러 응답 비율

4. **Saturation**
   - 시스템 리소스(CPU/Memory/Disk) 현황



Kubernetes Sample APP의 Full Stack 가시성 확보해보기 (Demo)

Bookinfo App 에 Datadog Agent 연동



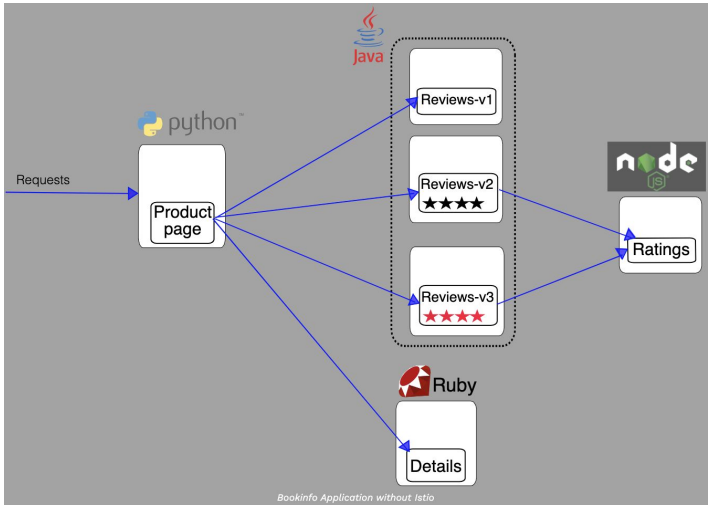
DATADOG

kubectl get pods -A 결과

```
jacky.jung@COMP11458:~/k8s_webinar|⇒  kubectl get pods -A
NAMESPACE              NAME                                        READY   STATUS    RESTARTS   AGE
default                details-v1-76ff5c68cb-gjbmj                 1/1     Running   0          30m
default                loadgenerator-v1.1-566bdd8c55-776t4         1/1     Running   0          21h
default                productpage-v1-6b9bd5d9bd-9sn2l             1/1     Running   0          3m13s
default                ratings-v1-5944768b44-t64hr                 1/1     Running   0          3m13s
default                reviews-v1-7944fd9fbb-g47nn                 1/1     Running   0          30m
default                reviews-v2-59b794bb4d-29thf                 1/1     Running   0          30m
default                reviews-v3-5cb7b59df8-k2mbf                 1/1     Running   0          30m
default                web-0                                       1/1     Running   0          44h
default                web-1                                       1/1     Running   0          44h
default                web-2                                       1/1     Running   0          44h
kube-system            aws-node-j2z6j                              1/1     Running   0          44h
kube-system            aws-node-vtrfs                              1/1     Running   0          44h
kube-system            coredns-7ff4cbb447-hjjgm                    1/1     Running   0          44h
kube-system            coredns-7ff4cbb447-hxl48                    1/1     Running   0          44h
kube-system            kube-proxy-j6shj                            1/1     Running   0          44h
kube-system            kube-proxy-rddvv                            1/1     Running   0          44h
kube-system            kube-state-metrics-96f95db6b-65bdf          2/2     Running   0          44h
kube-system            metrics-server-64c67868bd-vrs5c             1/1     Running   0          44h
kube-system            tiller-deploy-597765d7c-88mtf               1/1     Running   0          44h
kubernetes-dashboard   dashboard-metrics-scraper-69fcc6d9df-47hkc  1/1     Running   0          44h
kubernetes-dashboard   kubernetes-dashboard-6d75768647-v2ls4       1/1     Running   0          44h
```

목표 : Full Stack 가시성 확보

1. Cluster Node의 가시성 확보
2. Pod/Container 레벨의 가시성 확보
3. Kubernetes Event 수집
4. Application 로그 수집
5. Application의 Trace 수집
6. Cluster 외부 테스트 자동화를 통한 SLO 구성



어떤 작업이 필요한가요?

1. Cluster Node의 가시성 확보
2. Pod/Container 레벨의 가시성 확보
3. Kubernetes Event 수집
4. Application 로그 수집
5. Application의 Trace 수집

Datadog agent daemonset 배포

Option

1) Application에 Library 삽입
2) Istio를 통한 Trace 전송

6. Cluster 외부 자동화 테스트 자동화를 통한 SLO 구성

Synthetic을 이용한 Cluster 외부 Check

Step1: Datadog Agent 배포 (Demo)

```
kubectl create -f "https://raw.githubusercontent.com/DataDog/datadog-agent/master/Dockerfiles/manifests/rbac/clusterrole.yaml"
kubectl create -f "https://raw.githubusercontent.com/DataDog/datadog-agent/master/Dockerfiles/manifests/rbac/serviceaccount.yaml"
kubectl create -f "https://raw.githubusercontent.com/DataDog/datadog-agent/master/Dockerfiles/manifests/rbac/clusterrolebinding.yaml"
```

→ 1. RBAC 설정

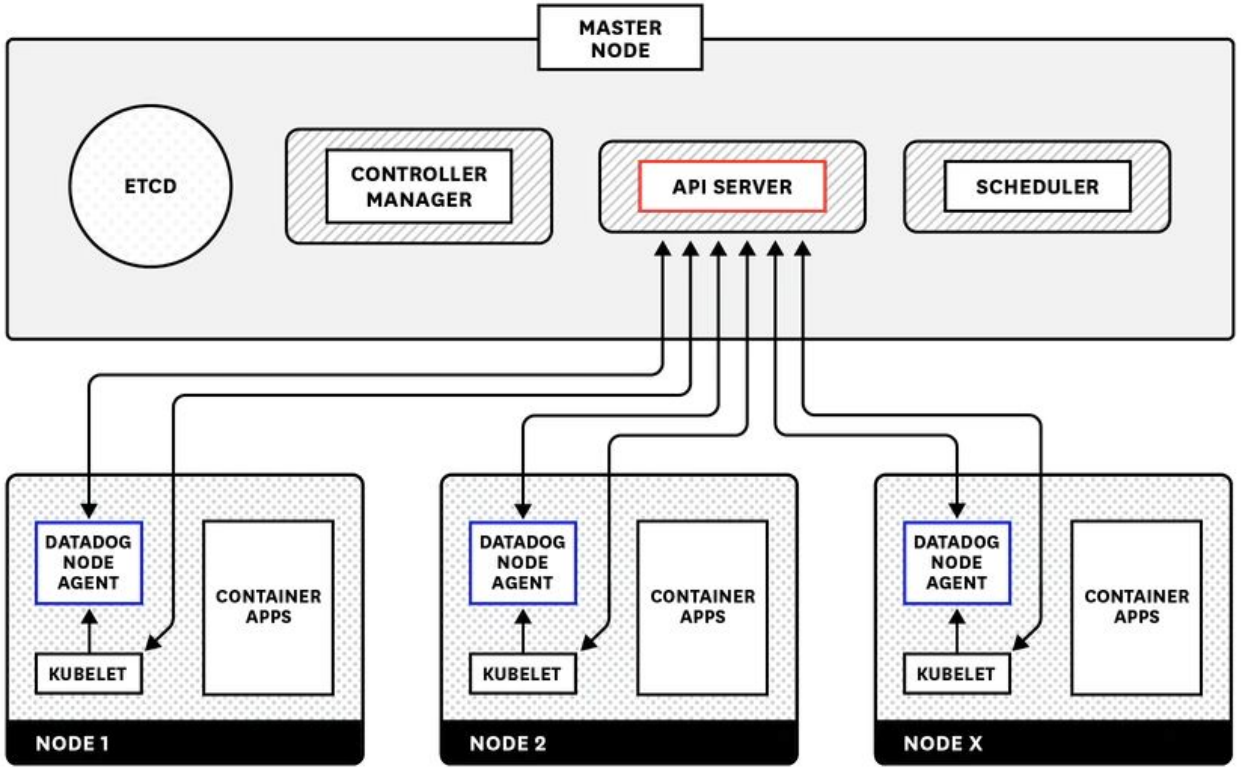
```
kubectl create secret generic datadog-secret --from-literal api-key="[redacted]"
```

→ 2. Secret 저장

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: datadog-agent
spec:
  selector:
    matchLabels:
      app: datadog-agent
  template:
    metadata:
      labels:
        app: datadog-agent
      name: datadog-agent
    spec:
      serviceAccountName: datadog-agent
      containers:
      - image: datadog/agent:7
        imagePullPolicy: Always
        name: datadog-agent
        ports:
          - containerPort: 8125
            # Custom metrics via DogStatsD - uncomment this section to enable custom metrics collection
            # hostPort: 8125
            name: dogstatsdport
            protocol: UDP
          - containerPort: 8126
            # Trace Collection (APM) - uncomment this section to enable APM
            # hostPort: 8126
            name: traceport
            protocol: TCP
```

→ 3. Agent Daemonset 배포

Datadog Agent 배포 후 아키텍쳐



Datadog Agent 주요 설정 확인 (1/5)

https://github.com/JungYoungseok/K8S_Webinar_202006/blob/master/datadog-agent-all-enabled.yaml

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: datadog-agent
  labels:
    app: datadog-agent
    service: datadog-agent
spec:
  revisionHistoryLimit: 10
  updateStrategy:
    type: RollingUpdate
  selector:
    matchLabels:
      app: datadog-agent
  template:
    metadata:
      labels:
        app: datadog-agent
        service: datadog-agent
      name: datadog-agent
      annotations:
        sidecar.istio.io/inject: "false"
        container.apparmor.security.beta.kubernetes.io/system-probe: unconfined
```

Agent는 Daemonset 으로 Deploy 됨

Datadog Agent 주요 설정 확인 (2/5)

```
24      spec:
25        serviceAccountName: datadog-agent
26        containers:
27          - name: datadog-agent
28            image: datadog/agent:7.19.2
29            imagePullPolicy: Always
30            ports:
31              - containerPort: 8125
32                hostPort: 8125
33                name: dogstatsdport
34                protocol: UDP
35              - containerPort: 8126
36                hostPort: 8126
37                name: traceport
38                protocol: TCP
39              - containerPort: 5555
40                hostPort: 5555
41                name: healthport
42                protocol: TCP
43            resources:
44              requests:
45                memory: "200Mi"
46                cpu: "0.25"
47              limits:
48                memory: "750Mi"
49                cpu: "1"
```

현재 Latest 버전으로 설정

Production에서는 특정 버전을 지정해서 사용 권장

Container의 resource request/limit 정의

노드당 Container 수가 많을 경우 모니터링을 통해 조절 필요

Datadog Agent 주요 설정 확인 (3/5)

```
50        env:
51          - name: DD_API_KEY
52            valueFrom:
53              secretKeyRef:
54                name: datadog-secret
55                key: api-key
56          - name: DD_AGENT_HOST
57            valueFrom:
58              fieldRef:
59                fieldPath: status.hostIP
60          - name: DD_SYSTEM_PROBE_ENABLED
61            value: "true"
62          - name: DD_SYSTEM_PROBE_EXTERNAL
63            value: "true"
64          - name: DD_SYSPROBE_SOCKET
65            value: "/var/run/sysprobe/sysprobe.sock"
66          - name: DD_COLLECT_KUBERNETES_EVENTS
67            value: "true"
68          - name: DD_LEADER_ELECTION
69            value: "true"
70          - name: KUBERNETES
71            value: "true"
72          - name: DD_KUBERNETES_KUBELET_HOST
73            valueFrom:
74              fieldRef:
75                fieldPath: status.hostIP
76          - name: DD_APM_ENABLED
77            value: "true"
78          - name: DD_AC_EXCLUDE
79            value: "name:datadog-agent name:agent name:system-probe"
```

Secret Object에 저장해놓은 API Key 사용

Kubernetes Event 수집 설정

APM Trace 수집 설정 활성화

로그 및 모니터링 데이터 수집 제외 컨테이너

Datadog Agent 주요 설정 확인 (4/5)

```
86      - name: DD_LOGS_CONFIG_CONTAINER_COLLECT_ALL
87        value: "true"
88      - name: DD_LOGS_CONFIG_USE_HTTP
89        value: "true"
90      - name: DD_LOGS_CONFIG_USE_COMPRESSION
91        value: "true"
92      - name: DD_LOGS_CONFIG_COMPRESSION_LEVEL
93        value: "6"
94      - name: DD_LOGS_ENABLED
95        value: "true"
96      - name: DD_TAGS
97        value: "env:aws-prd cluster-name:analysis-production"
98      - name: DD_TRACE_ANALYTICS_ENABLED
99        value: "true"
100     - name: DD_PROCESS_AGENT_ENABLED
101       value: "true"
102     - name: DD_KUBERNETES_POD_LABELS_AS_TAGS
103       value: '{"service":"service","team":"team"}'
104     - name: DD_HEALTH_PORT
105       value: "5555"
106     - name: DD_KUBELET_TLS_VERIFY
107       value: "false"
```

Container의 stdout/stderr 로그를 수집하도록 설정

Process 레벨 모니터링 설정 활성화

Datadog Agent 주요 설정 확인 (5/5)

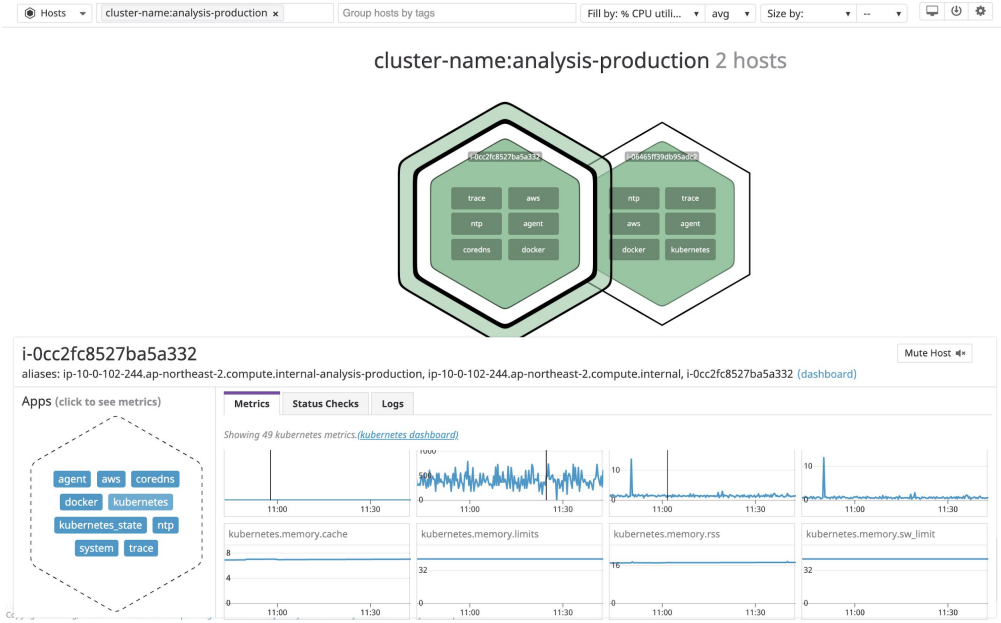
```
141       - name: system-probe
142         image: datadog/agent:7.19.2
143         imagePullPolicy: Always
144         command:
145           - /opt/datadog-agent/embedded/bin/system-probe
146         env:
147           - name: DD_SYSTEM_PROBE_ENABLED
148             value: "true"
149           - name: DD_SYSPROBE_SOCKET
150             value: "/var/run/sysprobe/sysprobe.sock"
151           - name: DD_DISABLE_DNS_INSPECTION
152             value: "false"
153           - name: DD_TAGS
154             value: "env:aws-prd cluster-name:analysis-production"
155           - name: DD_KUBERNETES_POD_LABELS_AS_TAGS
156             value: '{"service":"service","team":"team"}'
157           - name: DD_API_KEY
158             valueFrom:
159               secretKeyRef:
160                 name: datadog-secret
161                 key: api-key
162           - name: DD_AGENT_HOST
163             valueFrom:
164               fieldRef:
165                 fieldPath: status.hostIP
```

Network Performance Monitoring 용 컨테이너 활성화

- Network Flow 수집 (Network Map 구성 데이터)

Agent Deploy 후 활성화 된 기능 확인(Demo)

1. **Cluster Node의 가시성 확보**
2. **Pod/Container 레벨의 가시성 확보**
3. **Kubernetes Event 수집**
4. **Application 로그 수집**



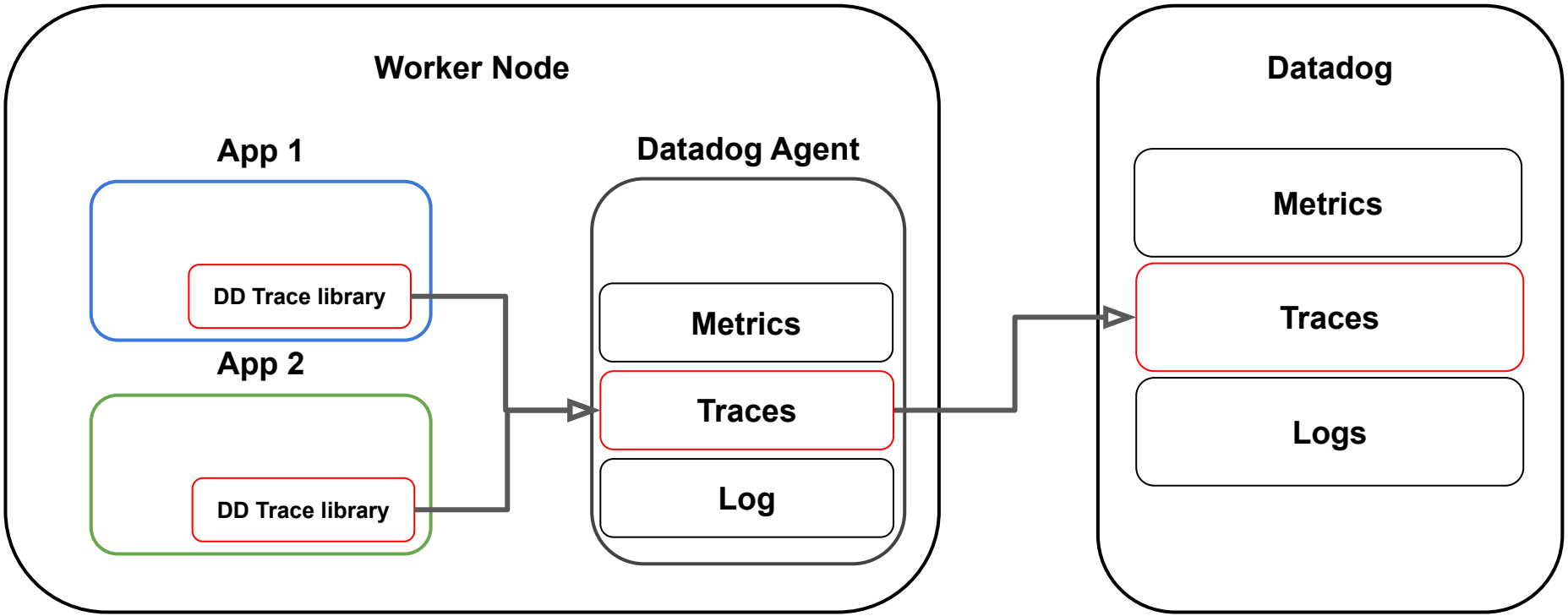
Step2: APM 설정 (Application Library 삽입)

1. **Cluster Node의 가시성 확보 ✓ Done**
2. **Pod/Container 레벨의 가시성 확보 ✓ Done**
3. **Kubernetes Event 수집 ✓ Done**
4. **Application 로그 수집 ✓ Done**
5. **Application의 Trace 수집**
6. **Cluster 외부 테스트 자동화를 통한 SLO 구성**

Option

1) Application에 Library 삽입
2) Istio를 통한 Trace 전송

Application Trace 수집 구조



APM 활성 설정 부분

1) Productpage 서비스(flask) 이미지의 Dockerfile

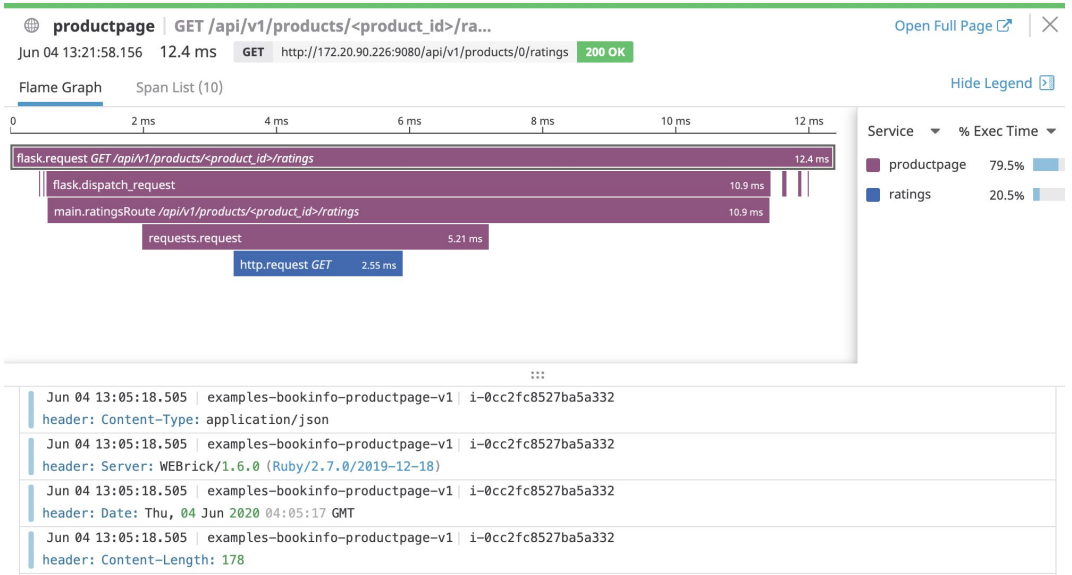
```
FROM python:3.7.4-slim

COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt

COPY test-requirements.txt ./
RUN pip install --no-cache-dir -r test-requirements.txt

COPY productpage.py /opt/microservices/
COPY tests/unit/* /opt/microservices/
COPY templates /opt/microservices/templates
COPY static /opt/microservices/static
COPY requirements.txt /opt/microservices/

ARG flood_factor
ENV FLOOD_FACTOR ${flood_factor:-0}

EXPOSE 9080
WORKDIR /opt/microservices
RUN python -m unittest discover
RUN pip install ddtrace
CMD ["ddtrace-run", "python", "productpage.py", "9080"]
```

2) Productpage 서비스 Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: productpage-v1
  labels:
    app: productpage
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: productpage
      version: v1
  template:
    metadata:
      labels:
        app: productpage
        version: v1
    spec:
      serviceAccountName: bookinfo-productpage
      containers:
      - name: productpage
        image: zsjung/examples-bookinfo-productpage-v1:1.0
        imagePullPolicy: Always
        env:
        - name: DD_AGENT_HOST
          valueFrom:
            fieldRef:
              fieldPath: status.hostIP
        - name: DD_TRACE_ANALYTICS_ENABLED
          value: "true"
        - name: DD_SERVICE
          value: "productpage"
        ports:
        - containerPort: 9080
```

APM 설정 후 활성화 된 기능 확인(Demo)

1. Cluster Node의 가시성 확보
2. Pod/Container 레벨의 가시성 확보
3. Kubernetes Event 수집
4. Application 로그 수집
5. Application의 Trace 수집

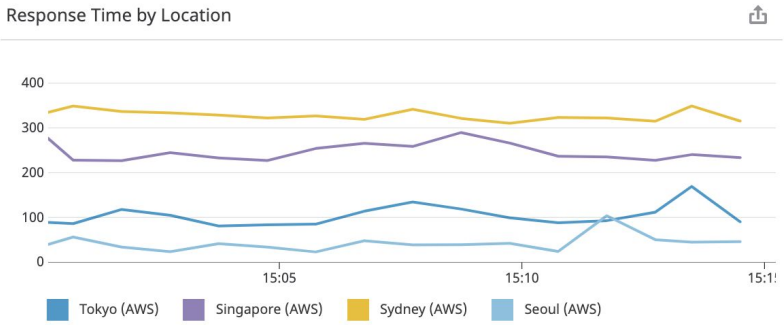


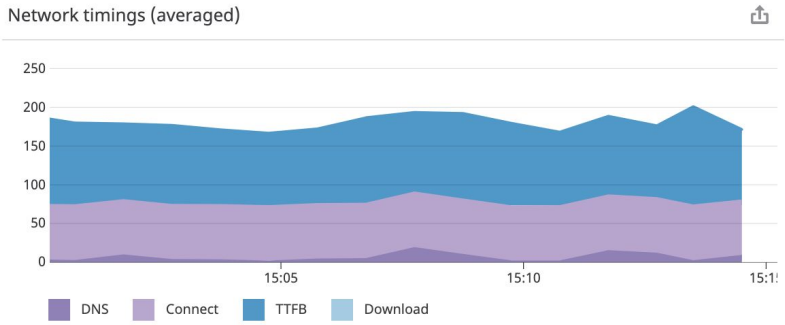
Step3: Synthetic 설정 (서비스 품질 측정 기능)

1. Cluster Node의 가시성 확보 ✓ Done
2. Pod/Container 레벨의 가시성 확보 ✓ Done
3. Kubernetes Event 수집 ✓ Done
4. Application 로그 수집 ✓ Done
5. Application의 Trace 수집 ✓ Done
6. Cluster 외부 테스트 자동화를 통한 SLO 구성

Synthetic을 이용하여 우리 서비스의 Availability/Quality 지표 측정

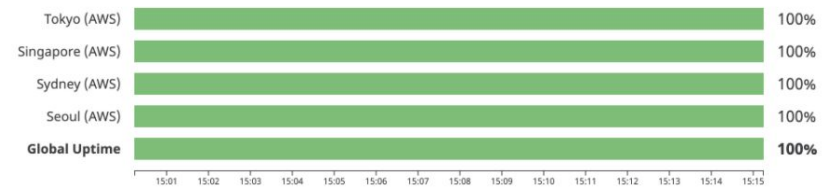
Step3: Synthetic 설정 (지역별 서비스 품질 측정)





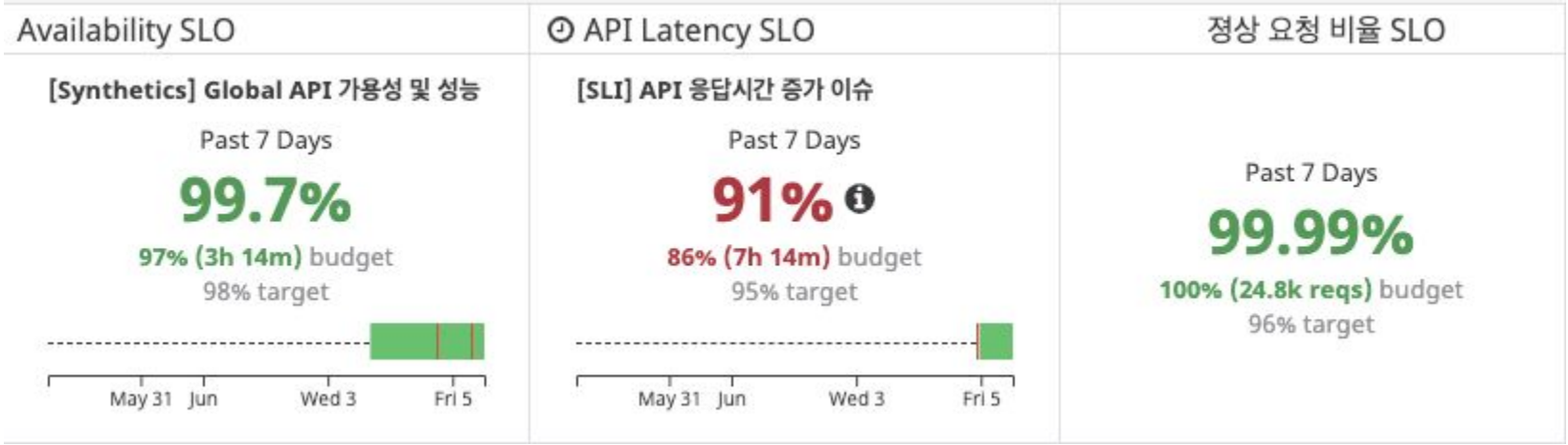
SLI(Service Level Indicator) 선정

우리 서비스 조직에서 가장 중요하게 판단하는 메트릭 선정 후 목표와 함께 관리

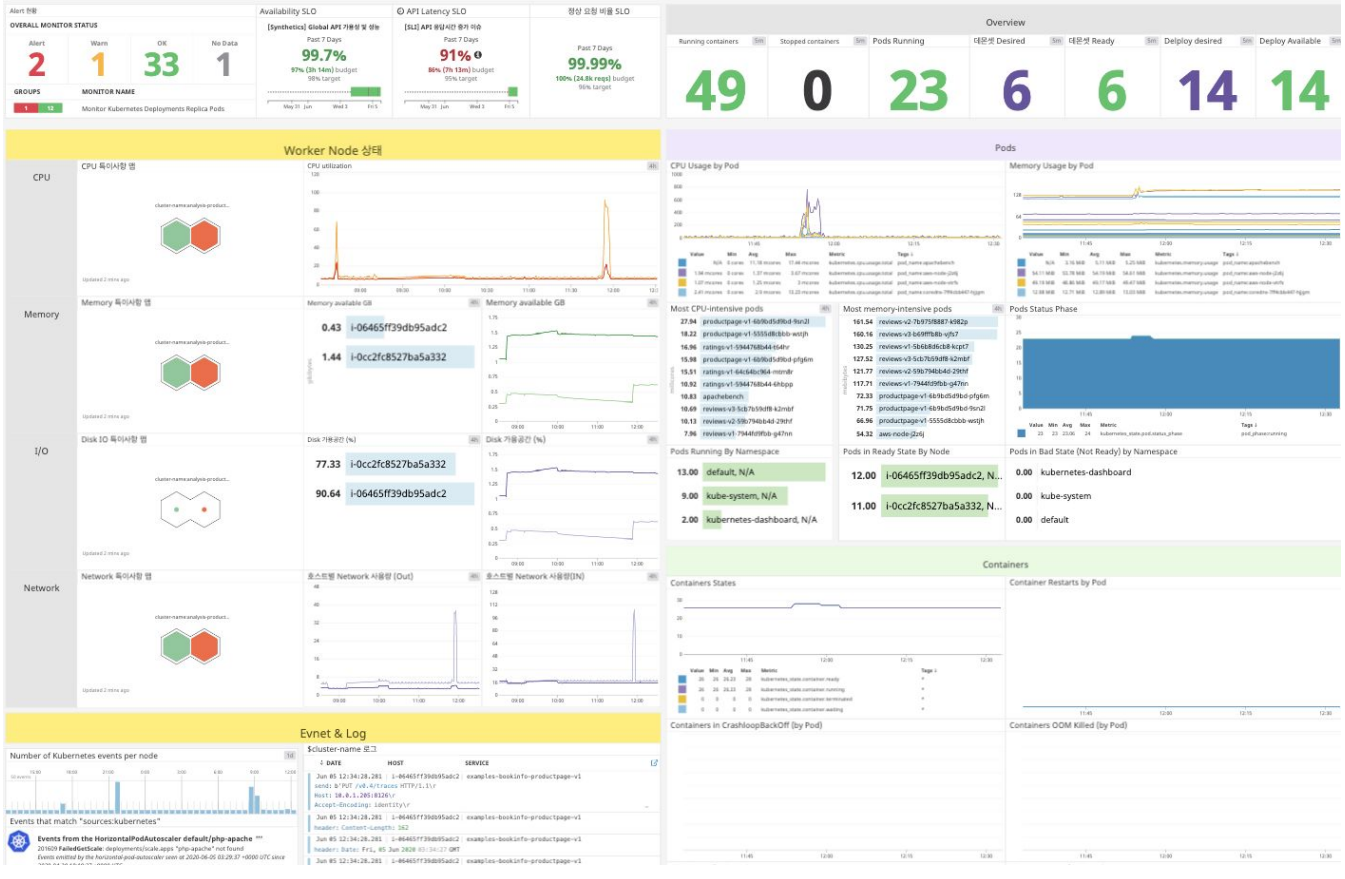
	메트릭 예	목표 예
서비스 가용성		Global Availability를 98% 이상 유지
성능 지표	RESOURCE / PC95:DURATION 34.26ms GET /productpage 50.86ms GET /api/v1/products/<product_id>/reviews 31.76ms GET /api/v1/products/<product_id>/ratings 26.65ms GET /api/v1/products/<product_id> 20.95ms GET /static/<path:filename> 8.66ms	주요 API의 p95 응답시간을 100ms 이하로 유지 (95% 이상의 요청을 0.5초 이내에 처리)
에러 비율	(정상 요청 수 / 모든 요청 수) *100	정상 처리 비율을 96% 이상 유지

SLO 관리 예 (Demo)

우리 서비스 조직에서 가장 중요하게 판단하는 메트릭의 목표를 적용하여 SLO 관리



K8S custom dashboard with SLO



Recap

1. Kubernetes Monitoring 의 어려운점

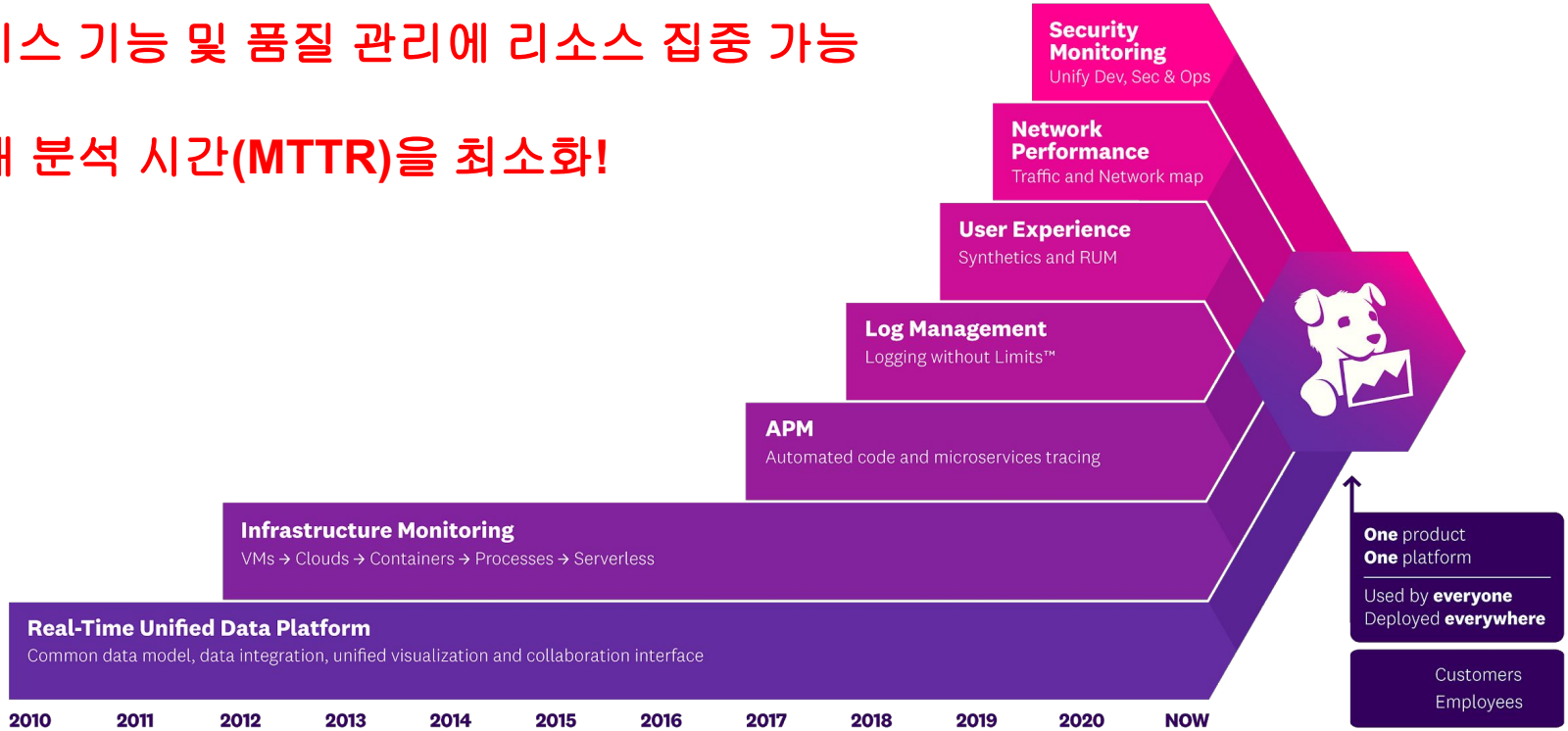
2. Kubernetes Monitoring Best Practice

3. Datadog Agent 배포 Demo를 통해 확보 가능한 Visibility 확인

Datadog Business Value

1. 서비스 기능 및 품질 관리에 리소스 집중 가능

2. 장애 분석 시간(MTTR)을 최소화!

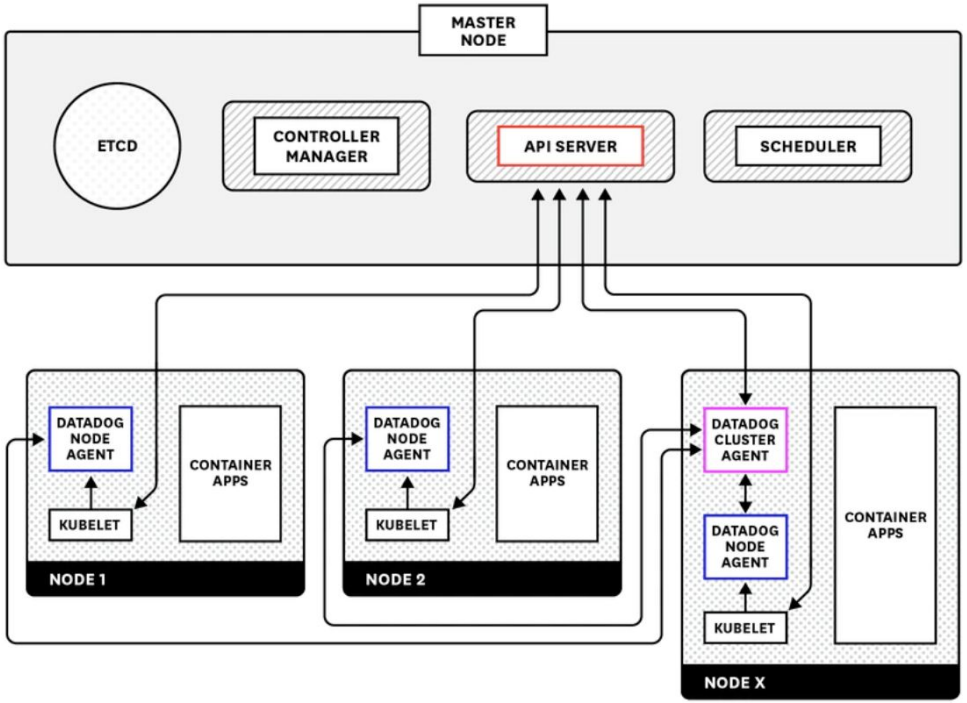


Q&A

감사합니다

영업 담당자와 미팅이 필요하시면
support@datadoghq.com로 메일 남겨주시면 됩니다

Datadog Cluster Agent



Datadog Cluster 사용 장점

1. API 서버의 부하 경감

2. External metric provider

3. K8S 환경 밖의 서비스 체크를 효과적으로 수행(ex:SaaS기반 솔루션, SNMP)

DATADOG