ISV2022SUMMIT

Datadog을 활용한 EKS FullStack 모니터링

정영석
TeamLead, Sales Engineering

14:45에 시작 예정입니다

About Speaker

정영석 (Jacky Jung)

What do I do?

- **TeamLead, Sales Engineering @Datadog (현재)**
- **CDN 서비스 아키텍트**
- **Devops 엔지니어**
- 개발자

오늘 다룰 주제는..

Datadog Introduction (25분)

EKS 환경의 샘플 e-commerce 서비스의 가시성 확보 과정 소개 (핸즈온 워크샵 90분)

Target Audiences

- Datadog이 국내 고객사에 어떤 Value를 전달하는지 궁금하신 분
- EKS FullStack 모니터링에 대해 고민하시는 분

Workshop 사전 준비 사항

1. **EKS 클러스터**
   - Kubernetes 1.7.6 이상 권장 (이번 워크샵 자료는 1.21 환경에서 진행)
   - 노드의 사양은 2 vCPU & 8 GB 이상 권장 (워크샵에서는 m5.large type으로 진행)
   - EKS 클러스터 생성 가이드(eksctl 사용 기준)
     i. eksctl 설치 (https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html)
     ii. # eksctl create cluster --name=ecommerce-isvworkshop (클러스터명은 자유롭게 설정 합니다)

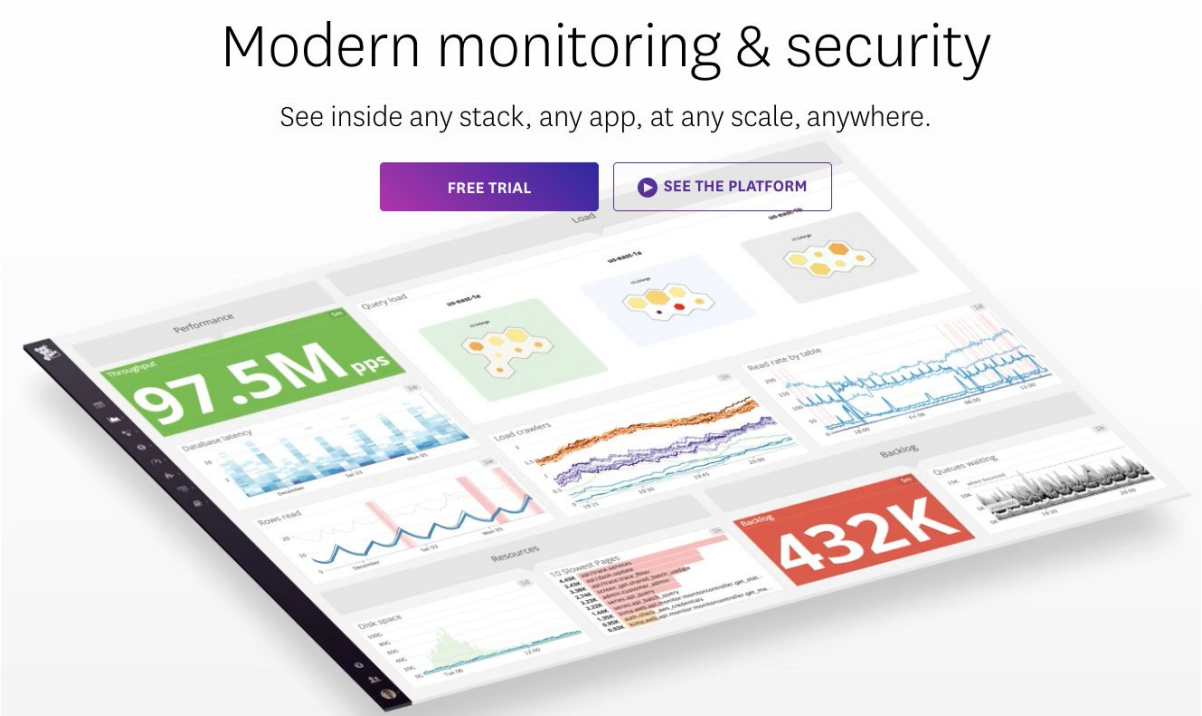
2. **kubectl**
   - 설치 가이드(https://kubernetes.io/docs/tasks/tools/)

3. **helm**
   - 설치 가이드(https://helm.sh/docs/intro/install/)
   - 워크샵은 Helm v3 기준으로 진행 예정

4. **git**

Datadog Introduction

Datadog?



빠르게 성장하는 **SaaS**기반 모니터링 플랫폼 회사

Datadog Korea

본사는 미국(NewYork), 직원수 3000+

저희는 선릉 Wework에 입주해 있습니다

Sales
Sales Engineer

6인실

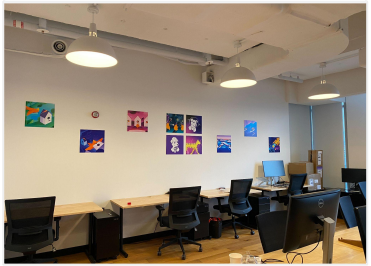
2019.04

Partner Sales
SDR
Sales
Sales Engineer

12인실

2020년

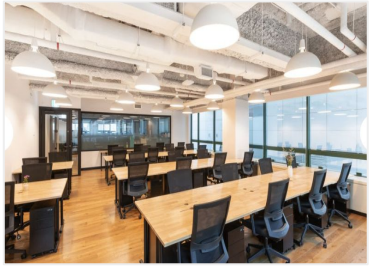
Solutions Engineer
Partner Sales
SDR
Sales
Sales Engineer



20인실

2021년

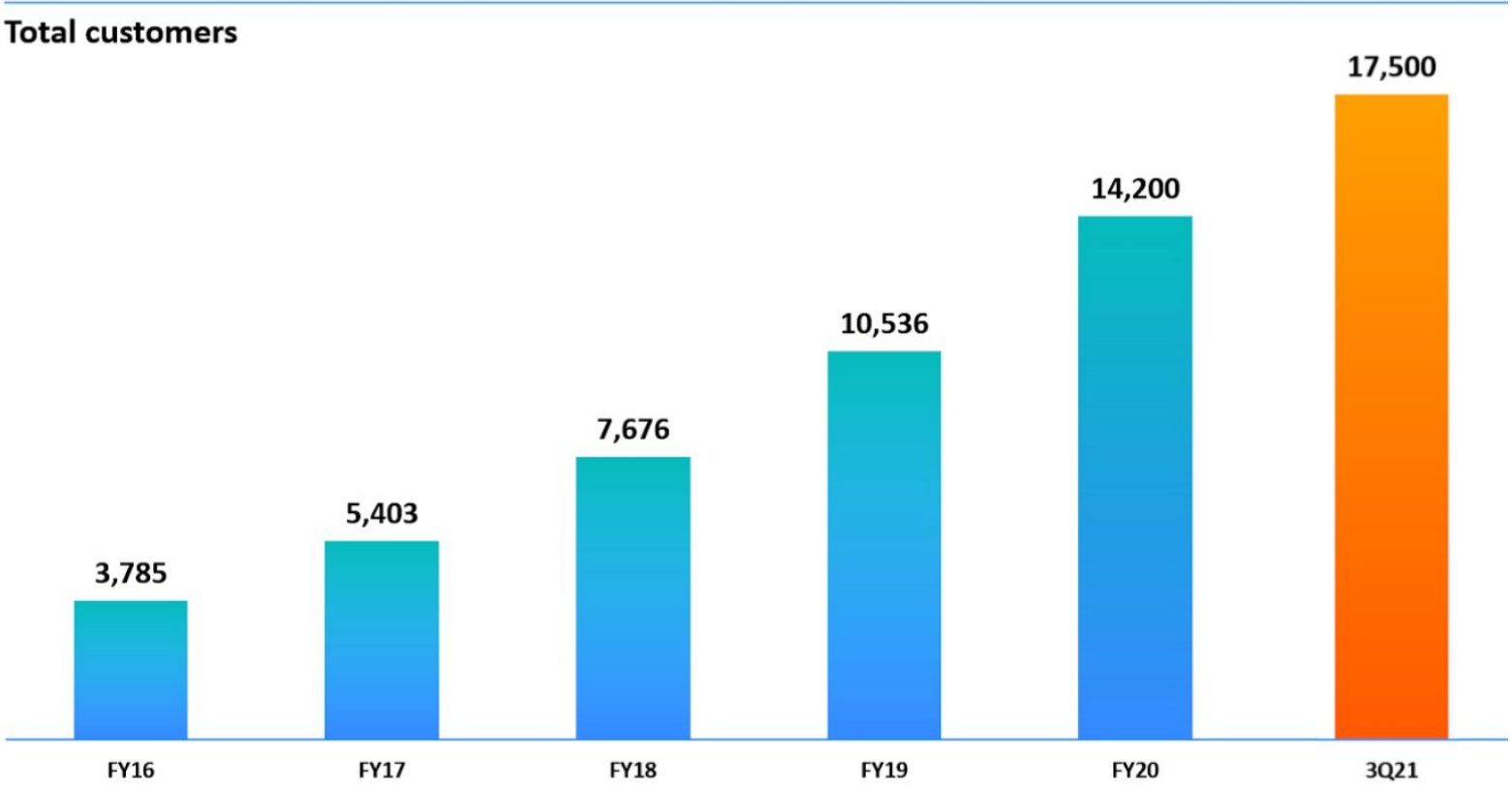
PSA
TAM/TEM
Customer Success
Solutions Engineer
Partner Sales
SDR
Sales
Sales Engineer



50인실(이사 예정)

2022년

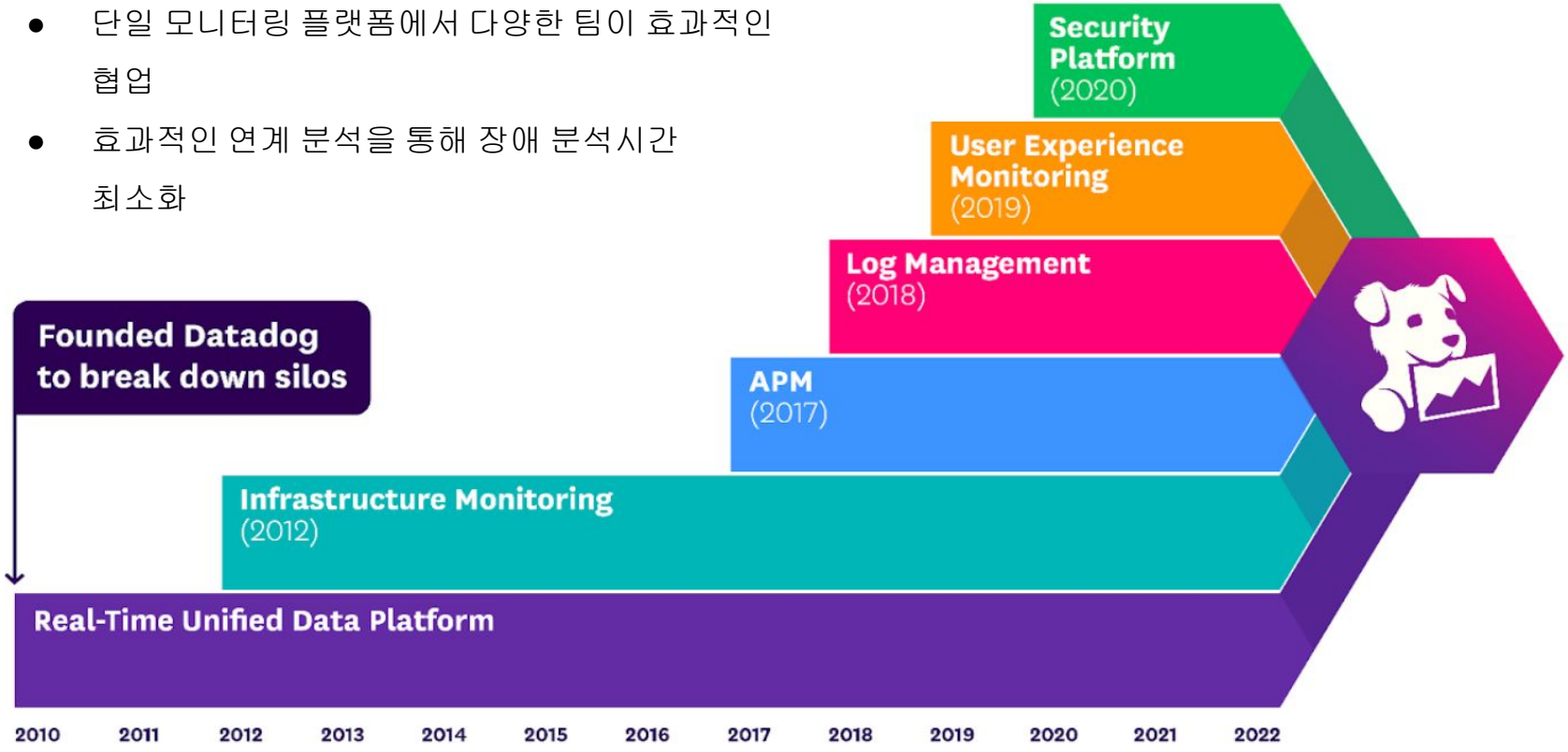
Datadog 고객 수


Total customers
3,785
5,403
7,676
10,536
14,200
17,500
FY16
FY17
FY18
FY19
FY20
3Q21

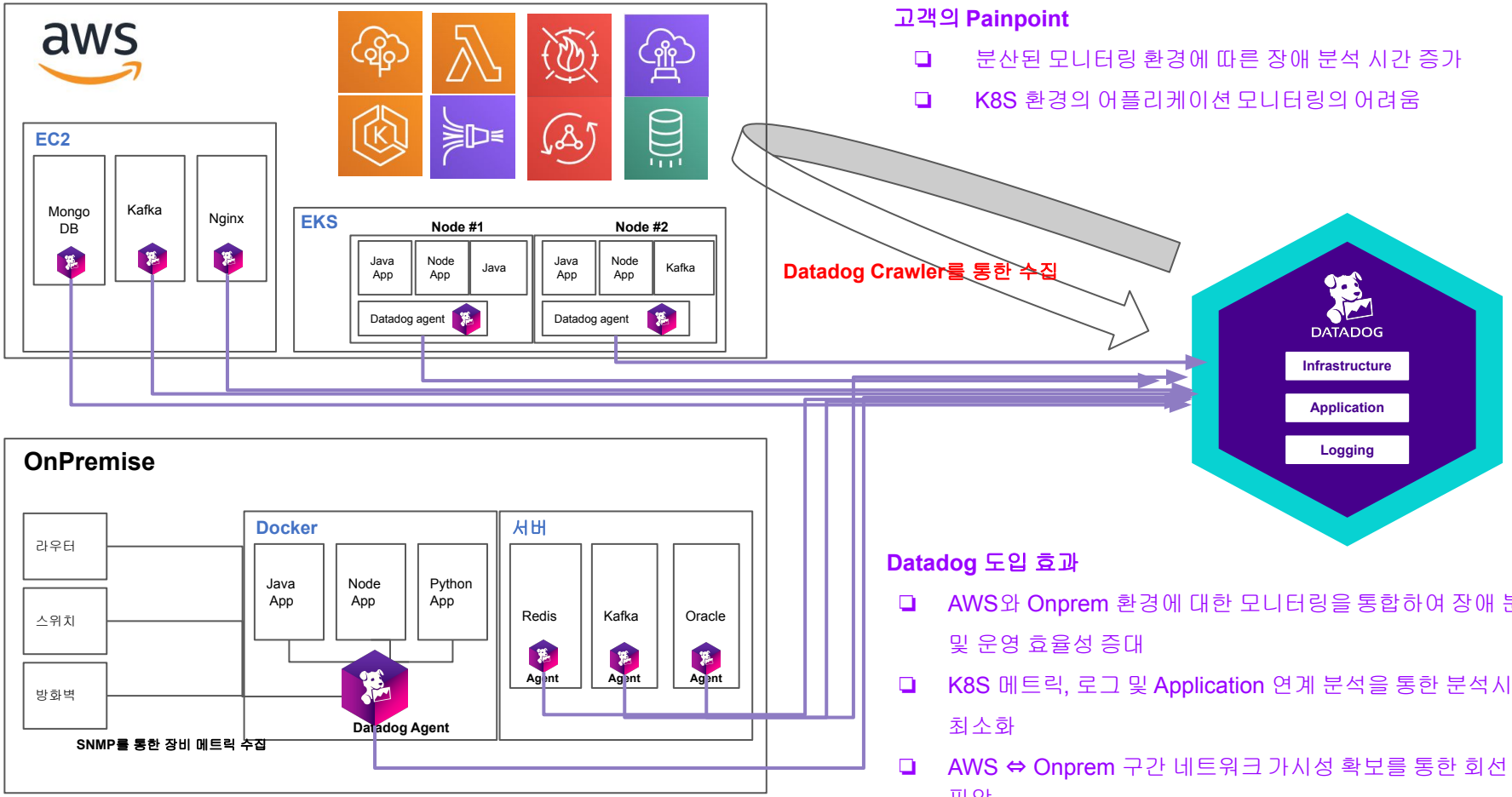
Datadog Products

핵심가치

- 단일 모니터링 플랫폼에서 다양한 팀이 효과적인 협업
- 효과적인 연계 분석을 통해 장애 분석시간 최소화



국내 고객 사례 #1 (Hybrid Datacenter)



고객의 Painpoint

- ❑ 분산된 모니터링 환경에 따른 장애 분석 시간 증가
- ❑ K8S 환경의 어플리케이션 모니터링의 어려움

Datadog 도입 효과

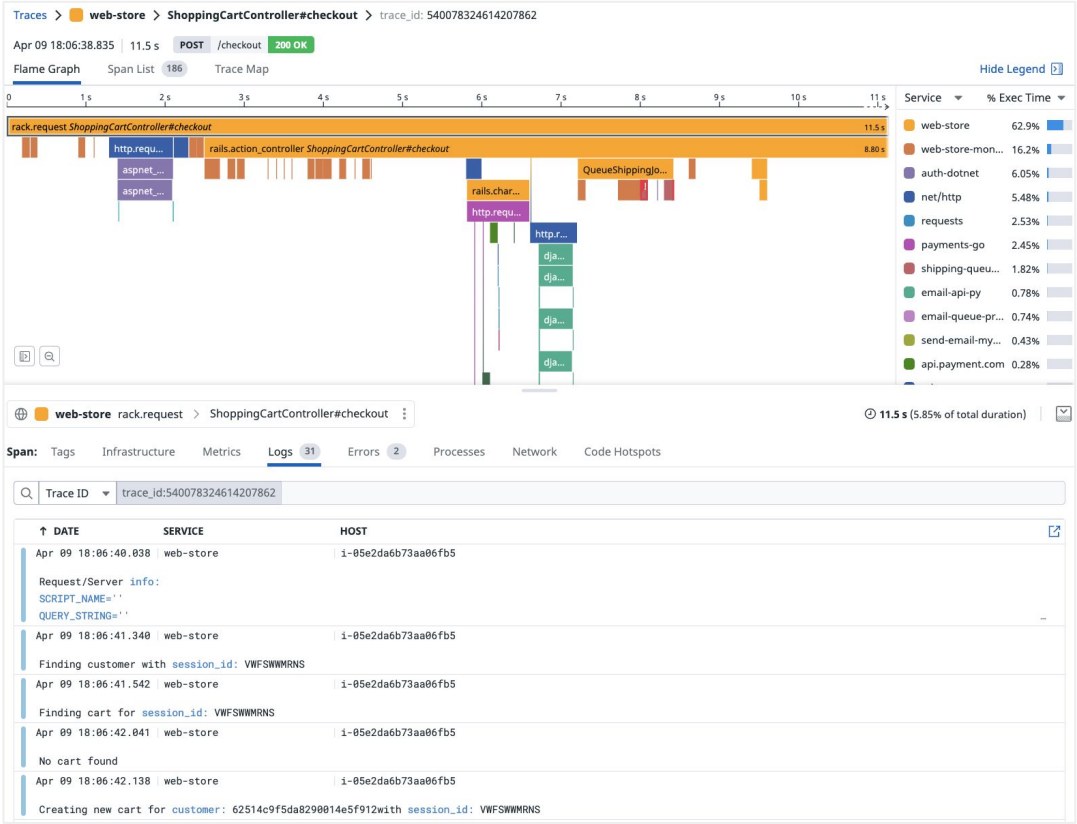
- ❑ AWS와 Onprem 환경에 대한 모니터링을 통합하여 장애 분석 및 운영 효율성 증대
- ❑ K8S 메트릭, 로그 및 Application 연계 분석을 통한 분석시간 최소화
- ❑ AWS ⇔ Onprem 구간 네트워크 가시성 확보를 통한 회선 이슈 파악

국내 고객 사례 #1 (도입효과)

- **AWS와 Onprem 환경에 대한 모니터링을 통합하여 장애 분석 및 운영 효율성 증대**

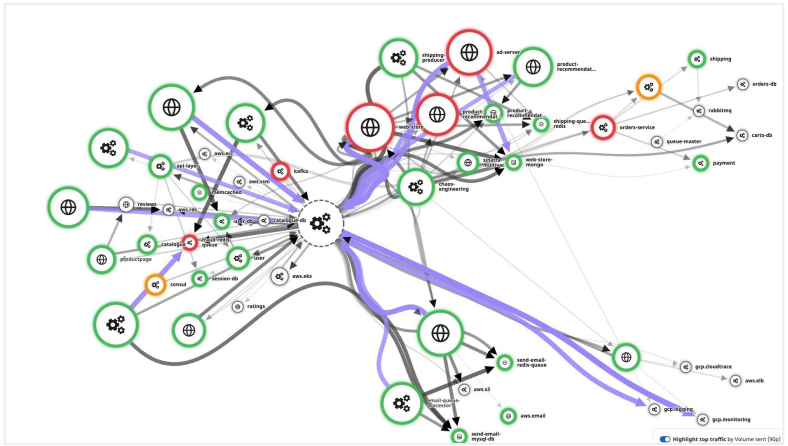
국내 고객 사례 #1 (도입효과)

- **K8S 메트릭, 로그 및 Application 연계 분석을 통한 분석시간 최소화**

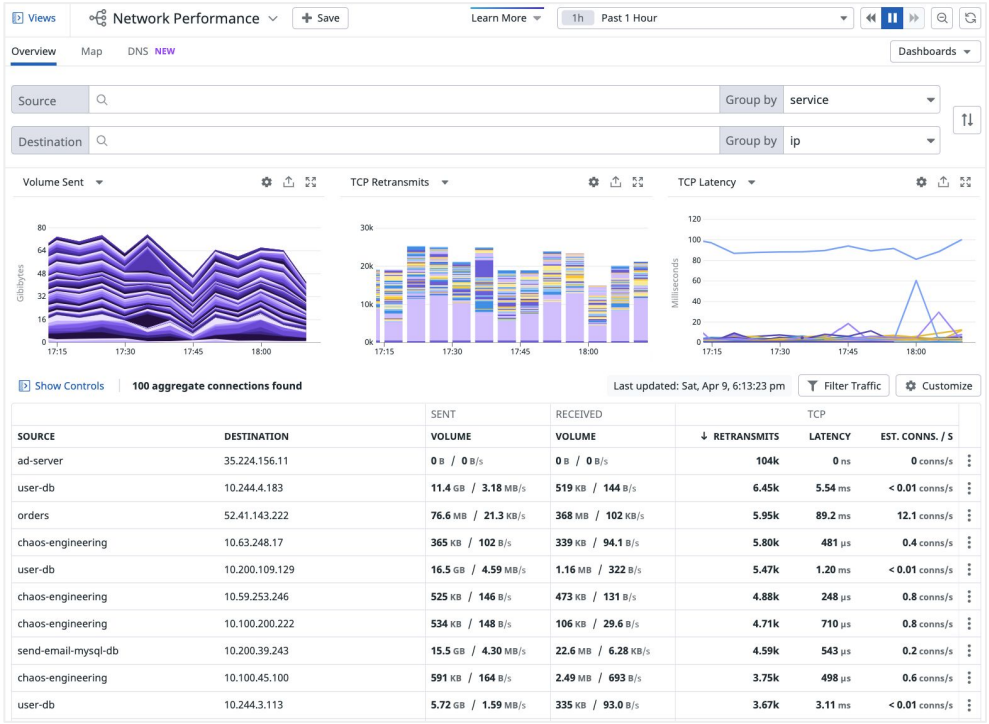


국내 고객 사례 #1 (도입효과)

- **AWS ⇔ Onprem 구간 네트워크 가시성 확보를 통한 IP별 특이사항 분석 가능**



container/pod 별 네트워크 흐름(Network Map)



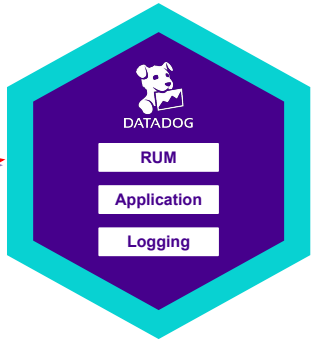
		SENT	RECEIVED	TCP		
SOURCE	DESTINATION	VOLUME	VOLUME	↓ RETRANSMITS	LATENCY	EST. CONNS. / S
ad-server	35.224.156.11	0 B / 0 B/s	0 B / 0 B/s	104k	0 ns	0 conns/s
user-db	10.244.4.183	11.4 GB / 3.18 MB/s	519 KB / 144 B/s	6.45k	5.54 ms	< 0.01 conns/s
orders	52.41.143.222	76.6 MB / 21.3 KB/s	368 MB / 102 KB/s	5.95k	89.2 ms	12.1 conns/s
chaos-engineering	10.63.248.17	365 KB / 102 B/s	339 KB / 94.1 B/s	5.80k	481 µs	0.4 conns/s
user-db	10.200.109.129	16.5 GB / 4.59 MB/s	1.16 MB / 322 B/s	5.47k	1.20 ms	< 0.01 conns/s
chaos-engineering	10.59.253.246	525 KB / 146 B/s	473 KB / 131 B/s	4.88k	248 µs	0.8 conns/s
chaos-engineering	10.100.200.222	534 KB / 148 B/s	106 KB / 29.6 B/s	4.71k	710 µs	0.8 conns/s
send-email-mysql-db	10.200.39.243	15.5 GB / 4.30 MB/s	22.6 MB / 6.28 KB/s	4.59k	543 µs	0.2 conns/s
chaos-engineering	10.100.45.100	591 KB / 164 B/s	2.49 MB / 693 B/s	3.75k	498 µs	0.6 conns/s
user-db	10.244.3.113	5.72 GB / 1.59 MB/s	335 KB / 93.0 B/s	3.67k	3.11 ms	< 0.01 conns/s

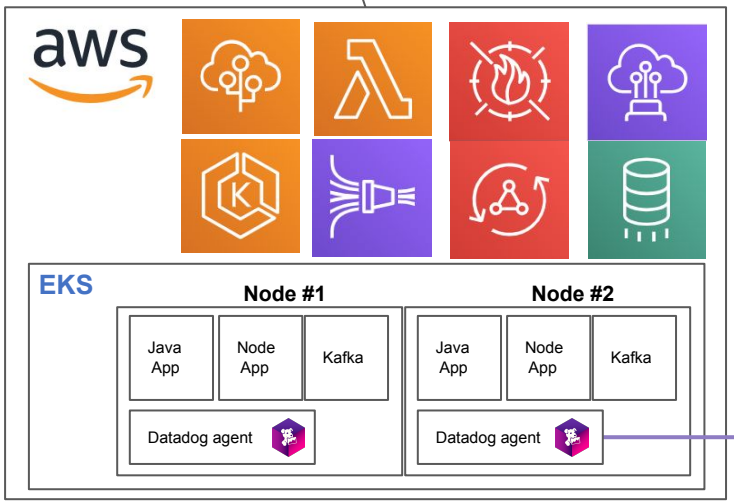
ip별 네트워크 현황 분석 (retransmits 기준 정렬)

국내 고객 사례 #2 (사용자 성능 모니터링)

사용자 관점의 Analytics 제공

- UX 여정 분석
- 성능 및 에러 분석 지원





고객의 Painpoint

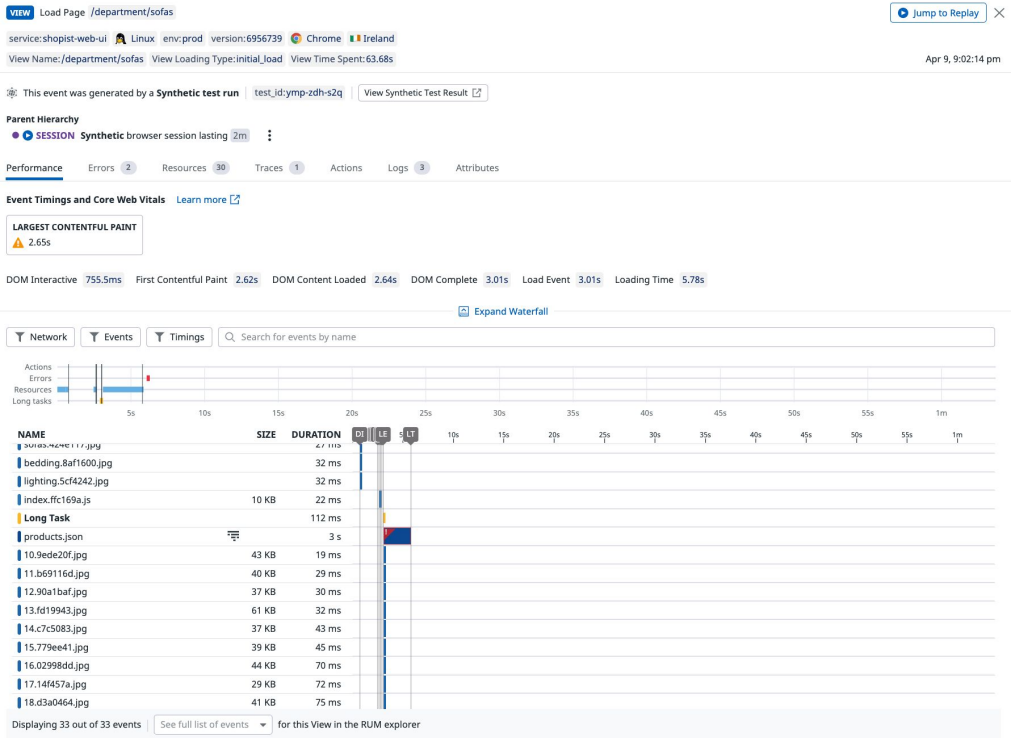
- ❑ 글로벌 서비스 사용자에 대한 체감 속도 파악의 어려움
- ❑ 사용자/인프라/Application/DB에 대한 통합 모니터링 구성 희망

Datadog 도입 효과

- ❑ 글로벌 사용자에 대한 체감 성능 파악
- ❑ 프론트부터 백엔드까지 연계 분석을 통한 신속한 사용자 이슈 분석
- ❑ Database 특이사항을 서비스 임팩트와 함께 연계 분석

국내 고객 사례 #2 (도입 효과)

- 글로벌 사용자에 대한 체감 성능 파악
- 프론트부터 백엔드까지 연계 분석을 통한 신속한 사용자 이슈 분석



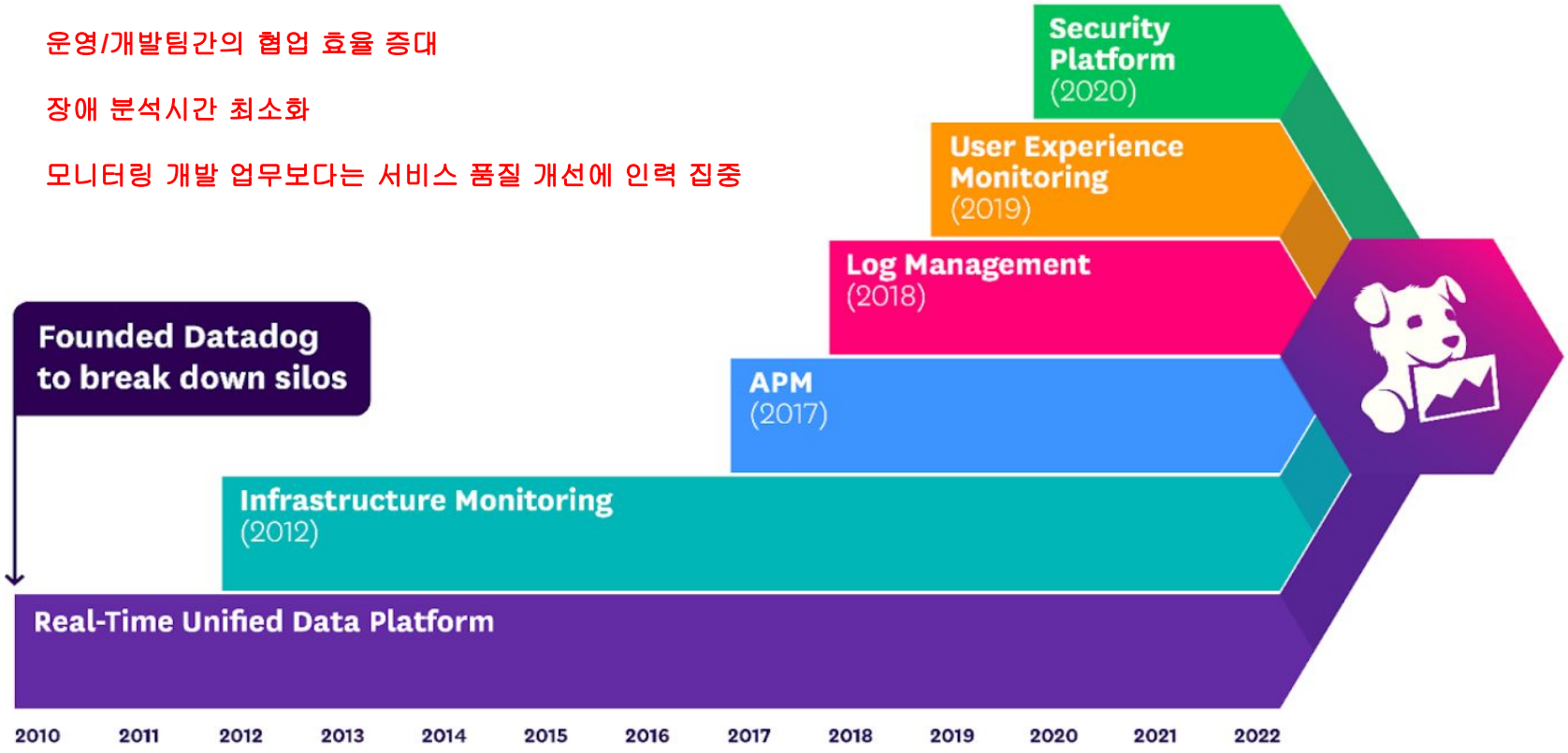
국내 고객 사례 #2 (도입 효과)

- **Database 특이사항을 서비스 임팩트와 함께 연계 분석**

Datadog 핵심 가치

IT 서비스 모니터링에 필요한 분석 데이터를 효과적으로 수집하여

- 운영/개발팀간의 협업 효율 증대
- 장애 분석시간 최소화
- 모니터링 개발 업무보다는 서비스 품질 개선에 인력 집중



Datadog 핸즈온 워크샵

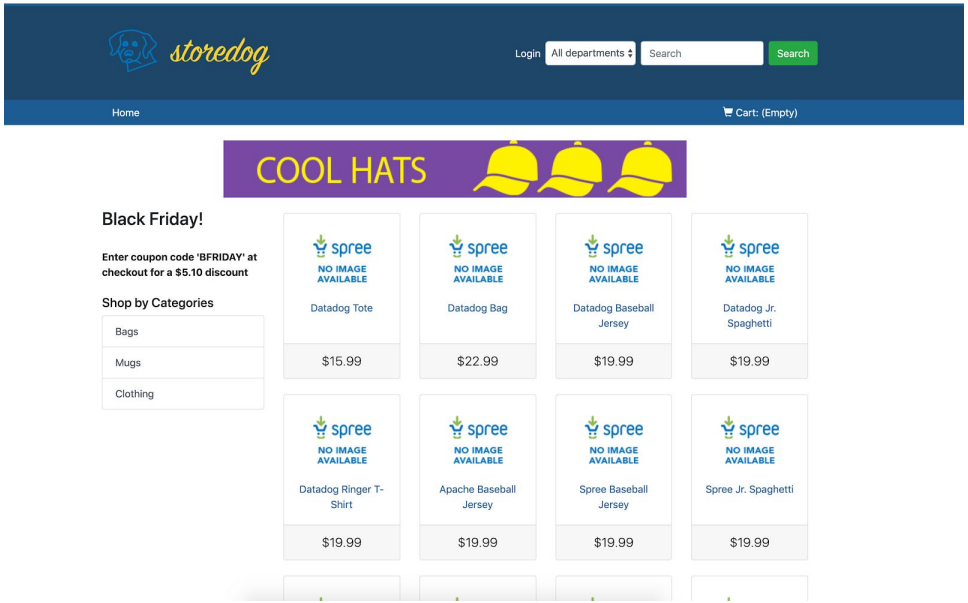
EKS 환경의 샘플 e-commerce 서비스의 가시성 확보 해보기 (90 min)

Workshop 내용

EKS 환경에 배포된 e-commerce 서비스의 가시성 확보해볼 예정입니다

서비스 구성

- Frontend (Javascript)
- Backends (Ruby + Python)
- Database (postgres + sqlite)



샘플 Ecommerce repository: https://bit.ly/3JrPsU5

Workshop 사전 준비 사항

1. **EKS 클러스터**
   - Kubernetes 1.7.6 이상 권장 (이번 워크샵 자료는 1.21 환경에서 진행)
   - 노드의 사양은 2 vCPU & 8 GB 이상 권장 (워크샵에서는 m5.large type으로 진행)
   - EKS 클러스터 생성 가이드(eksctl 사용 기준)
     i. eksctl 설치 (https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html)
     ii. # eksctl create cluster --name=ecommerce-isvworkshop (클러스터명은 자유롭게 설정 합니다)

2. **Kubectl**
   - 설치 가이드(https://kubernetes.io/docs/tasks/tools/)

3. **Helm**
   - 설치 가이드(https://helm.sh/docs/intro/install/)
   - 워크샵은 Helm v3 기준으로 진행 예정

4. **git**

다음의 명령어가 잘 실행되나요?

1. 현재 클러스터의 Pod 확인

```
[root@ip-172-31-31-95 ec2-user]# kubectl get pods -A
NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE
kube-system   aws-node-8swff                1/1     Running   0          5m10s
kube-system   aws-node-c4gj2                1/1     Running   0          5m14s
kube-system   coredns-6dbb778559-8ddsx      1/1     Running   0          13m
kube-system   coredns-6dbb778559-px5hk      1/1     Running   0          13m
kube-system   kube-proxy-bs2cg              1/1     Running   0          5m10s
kube-system   kube-proxy-qb76n              1/1     Running   0          5m14s
[root@ip-172-31-31-95 ec2-user]#
```

2. helm version 확인

```
[root@ip-172-31-31-95 ec2-user]# helm version
version.BuildInfo{Version:"v3.8.1", GitCommit:"5cb9af4b1b271d11d7a97a71df3ac337dd94ad37", GitTreeState:"clean", GoVersion:"go1.17.5"}
[root@ip-172-31-31-95 ec2-user]#
```

3. e-commerce 프로젝트 git clone

git clone https://github.com/JungYoungseok/ecommerce-workshop.git

```
[root@ip-172-31-31-95 project]# git clone https://github.com/JungYoungseok/ecommerce-workshop.git
Cloning into 'ecommerce-workshop'...
remote: Enumerating objects: 215131, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 215131 (delta 4), reused 4 (delta 4), pack-reused 215125
Receiving objects: 100% (215131/215131), 100.66 MiB | 9.40 MiB/s, done.
Resolving deltas: 100% (127137/127137), done.
[root@ip-172-31-31-95 project]#
```

E-commerce 서비스 배포하기

1. resource가 있는 디렉토리로 이동합니다

```
[root@ip-172-31-31-95 project]# ls
ecommerce-workshop
[root@ip-172-31-31-95 project]# cd ecommerce-workshop/deploy/generic-k8s/ecommerce-app/
[root@ip-172-31-31-95 ecommerce-app]# ls
advertisements.yaml  db.yaml  discounts.yaml  frontend.yaml
```

2. yaml 리소스를 배포합니다

```
[root@ip-172-31-31-95 ecommerce-app]# kubectl  apply -f .
deployment.apps/advertisements created
service/advertisements created
serviceaccount/postgres created
secret/db-password created
persistentvolume/task-pv-volume created
persistentvolumeclaim/task-pvc-volume created
deployment.apps/db created
service/db created
deployment.apps/discounts created
service/discounts created
deployment.apps/frontend created
service/frontend created
```

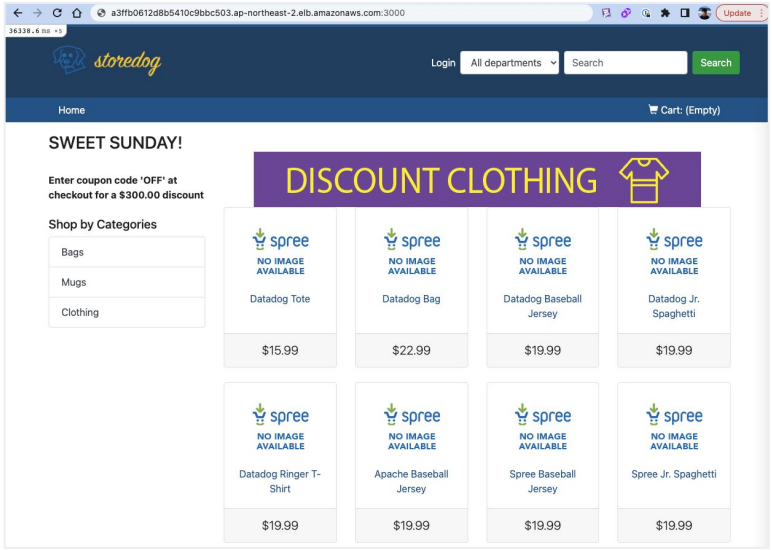
```
[root@ip-172-31-31-95 ecommerce-app]# kubectl get pods
NAME                             READY   STATUS    RESTARTS   AGE
advertisements-d8bdff946-j48dv   1/1     Running   0          20h
db-6cb7fb6db7-hw7kg              1/1     Running   0          22h
discounts-86d6b79d84-6s2ld       1/1     Running   0          20h
frontend-dbc8768fb-x7hr2         1/1     Running   0          4m3s
```

E-commerce 서비스 배포하기

3. 프론트 서비스 엔드포인트 확인

```
[root@ip-172-31-31-95 ecommerce-app]# kubectl  get service
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)
advertisements  ClusterIP      10.100.32.183   <none>                                                                   5002/TCP
db              ClusterIP      10.100.166.94   <none>                                                                   5432/TCP
discounts       ClusterIP      10.100.3.251    <none>                                                                   5001/TCP
frontend        LoadBalancer   10.100.6.152    a3ffb0612d8b5410c9bbc7-ap-northeast-2.elb.amazonaws.com   3000:32679/TCP
kubernetes      ClusterIP      10.100.0.1      <none>                                                                   443/TCP
```

4. 엔드포인트 접근시 다음의 페이지가 보이시나요?



혹시 접속이 안된다면?

- **Security Group**에 **3000 Port** 오픈이 되어 있나요?
- **Container**에 접속해서 서비스 접속이되는지 확인해볼까요?
  - kubectl -it exec [front POD] sh

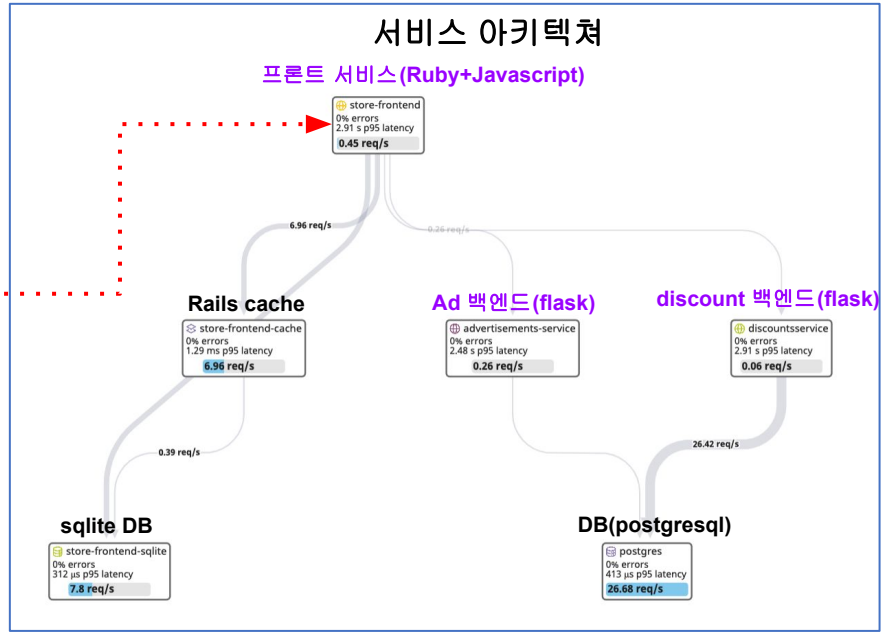
E-commerce 서비스 배포하기

5. 서비스에 부하를 주입해봅시다

```
[root@ip-172-31-31-95 ecommerce-app]# kubectl apply -f loadgenerator.yaml
deployment.apps/loadgenerator-v1.1 created
```

* 클러스터 내부에서 요청 생성하는 컨테이너

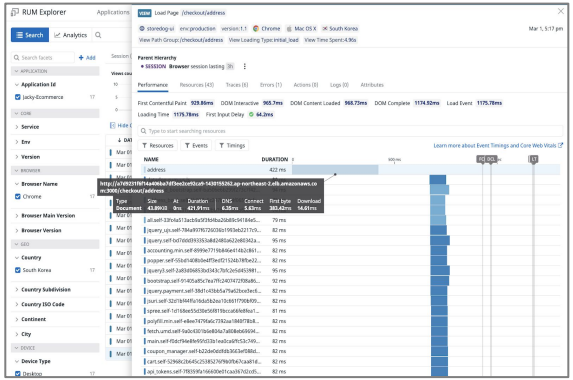
```
    spec:
      containers:
      - name: loadgenerator
        image: busybox
        command: ["/bin/sh"]
        args: ["-c", "TARGET=frontend.default.svc.cluster.local:3000;while
true; do wget -q -O- http://$TARGET/api_tokens>>dev/null; sleep 1;wget -q -
O- http://$TARGET/cart>>dev/null; sleep 1;wget -q -O- http://$TARGET/cart_l
ink>>dev/null; sleep 1;wget -q -O- http://$TARGET/cart?variant_id=11>>dev/n
ull; sleep 1;wget -q -O- http://$TARGET//products/datadog-ringer-t-shirt>>d
ev/null; sleep 1;wget -q -O- http://$TARGET//products/spree-bag>>dev/null;
sleep 1;wget -q -O- http://$TARGET/discounts/get>>dev/null; sleep 1; done"]
```



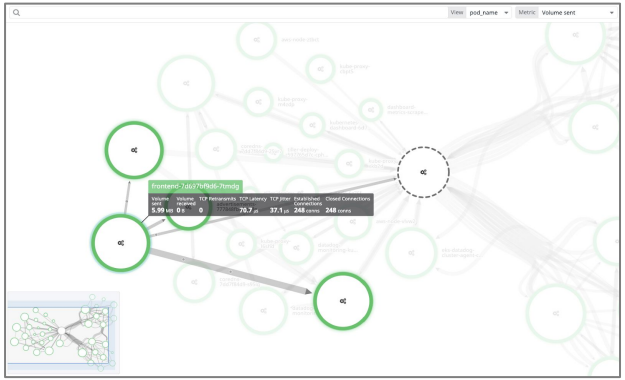
여기까지 하면 서비스 환경 구성 완료!

오늘 최종적으로 확인할 FullStack 가시성

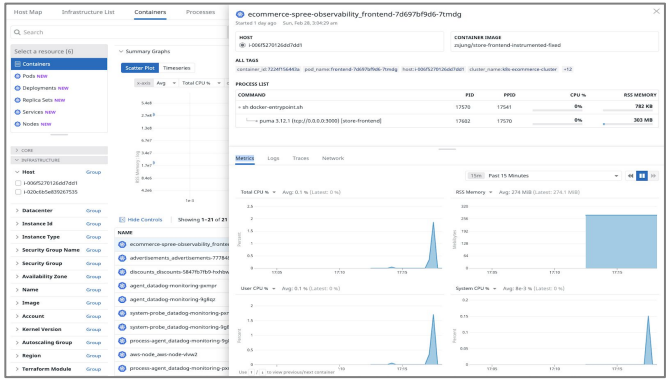
엔드 유저의 성능 분석 지표



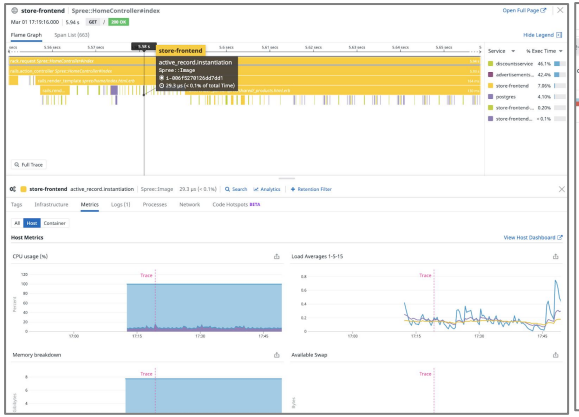
컨테이너간 네트워크 흐름 파악



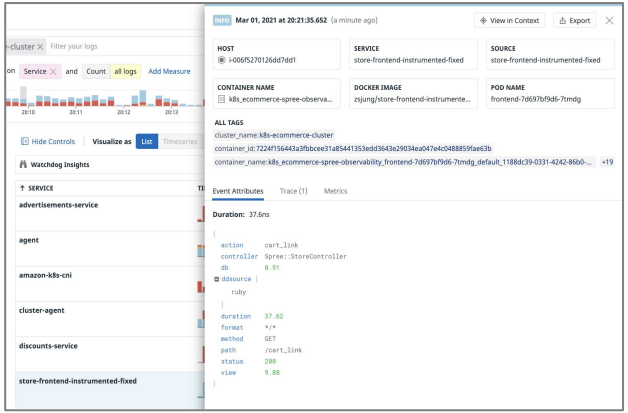
Live Container 분석



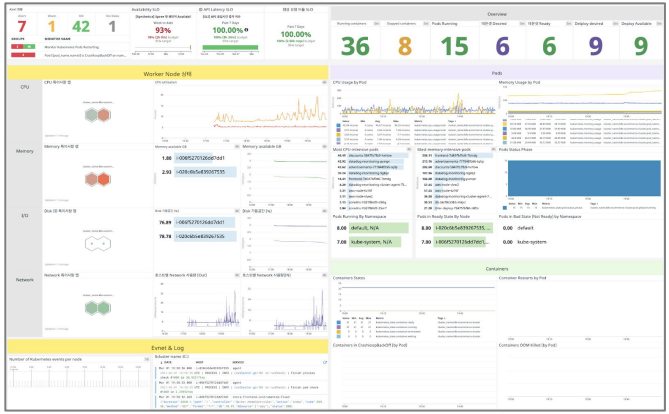
Application 분석(Trace)



Log 분석



통합 대시보드



1. AWS 서비스 가시성 확보

왜 필요한가요?

⇒ RDS, ALB, ElasticCache와 같은 AWS 서비스에 대한 메트릭 수집을 위함

⇒ IAM연동 후 Cloudwatch 통해 수집

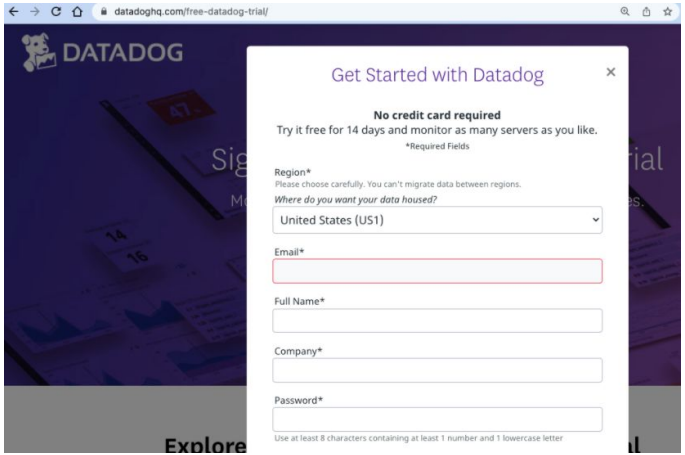
Datadog Free trial 계정 생성

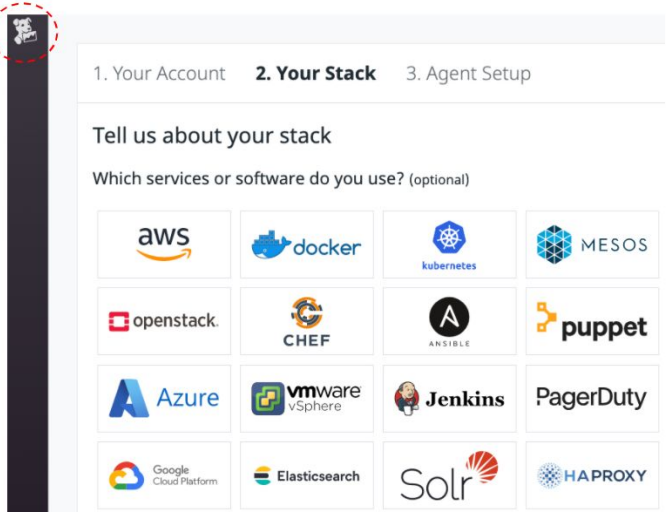
1. https://datadog.com/free-datadog-trial 에 접속

2. **Get Started** 클릭 후 보이는 팝업 화면에서 **Sign up** 진행(US1 선택)

3. 가입이 완료되면 https://app.datadoghq.com/ 에 로그인을 합니다

4. 왼쪽 상단의 **Bits** 아이콘을 클릭하여 메인페이지로 이동합니다



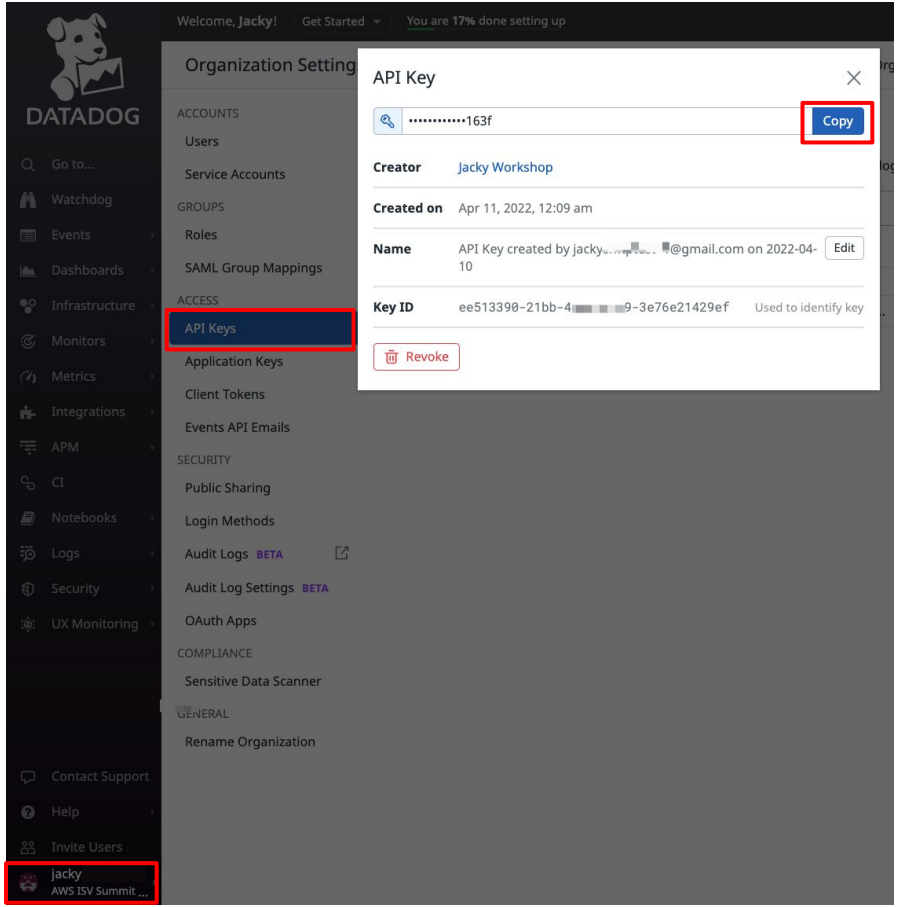


Datadog API Key 확인

1. 좌측 하단의 계정 ⇒ **Organization Setting ⇒ API Keys** 메뉴 진입

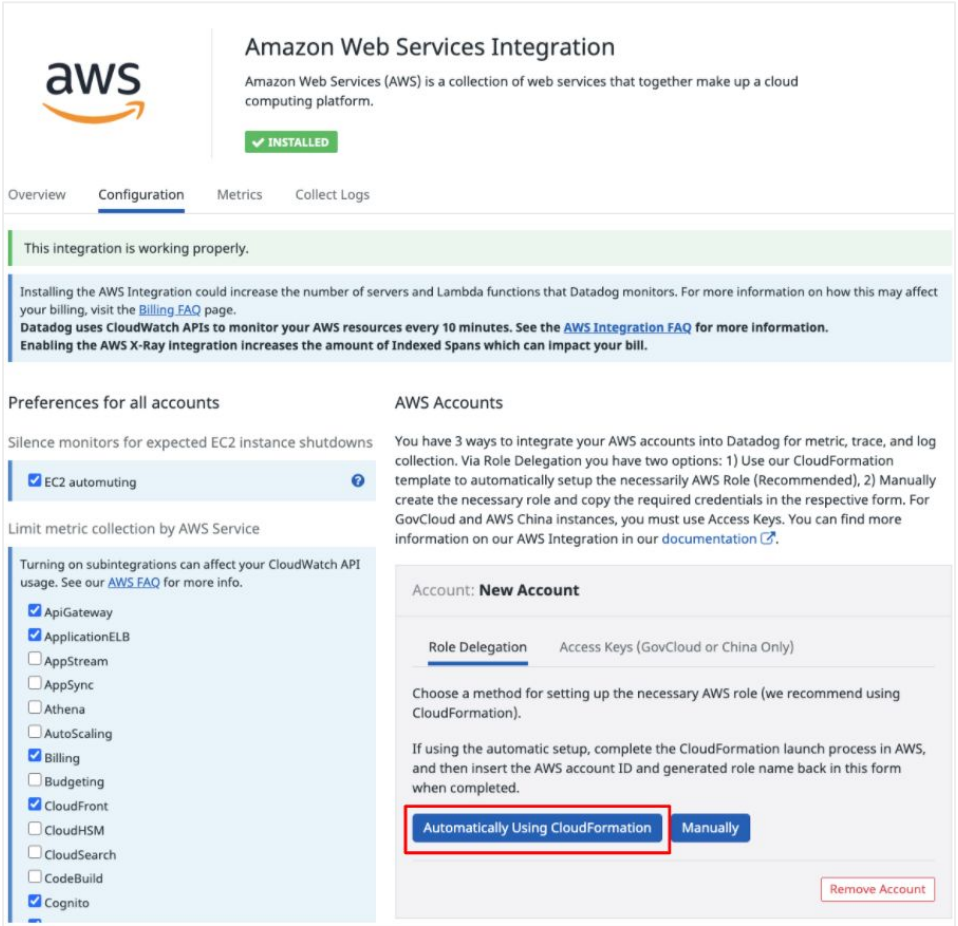
2. **Copy** 버튼을 통해 **API Key**를 메모장에 복사해 놓습니다

*** API_Key는 AWS 계정 및 EKS 연동시 사용됩니다**



AWS 계정 연동 (Step 1)

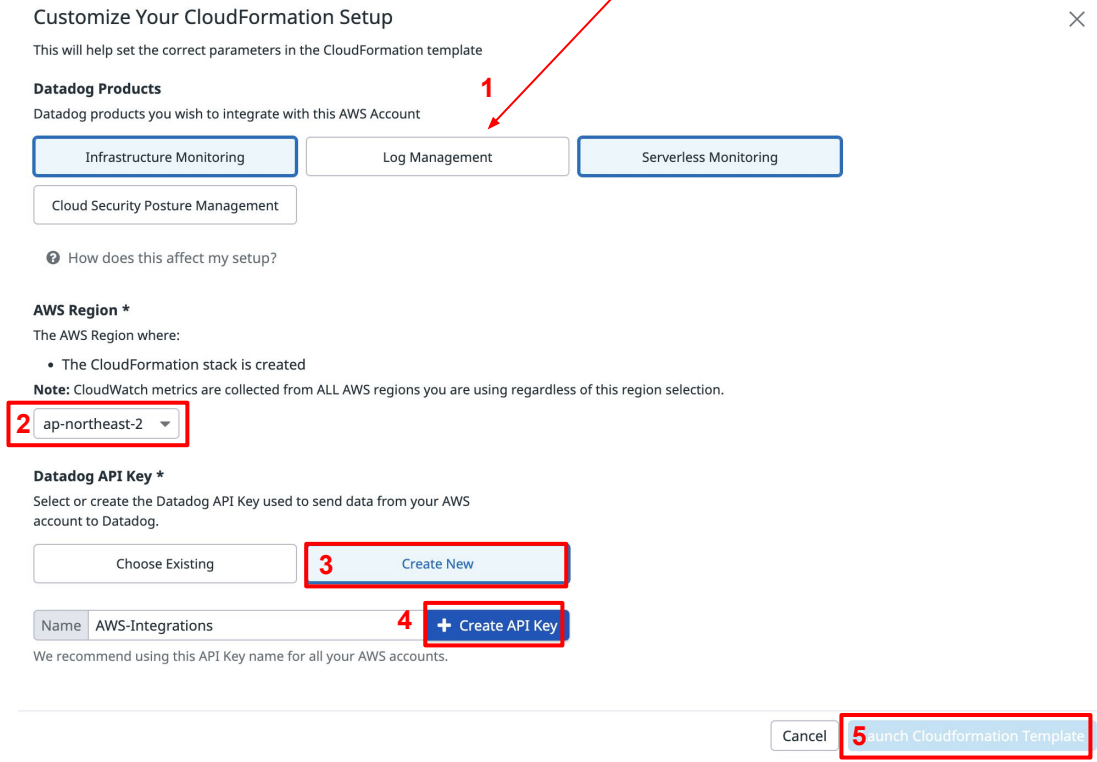
- ❑ Datadog Portal의 AWS Integration 페이지에서
- ❑ Automatically Using CloudFormation 선택



AWS 계정 연동 (Step 2)

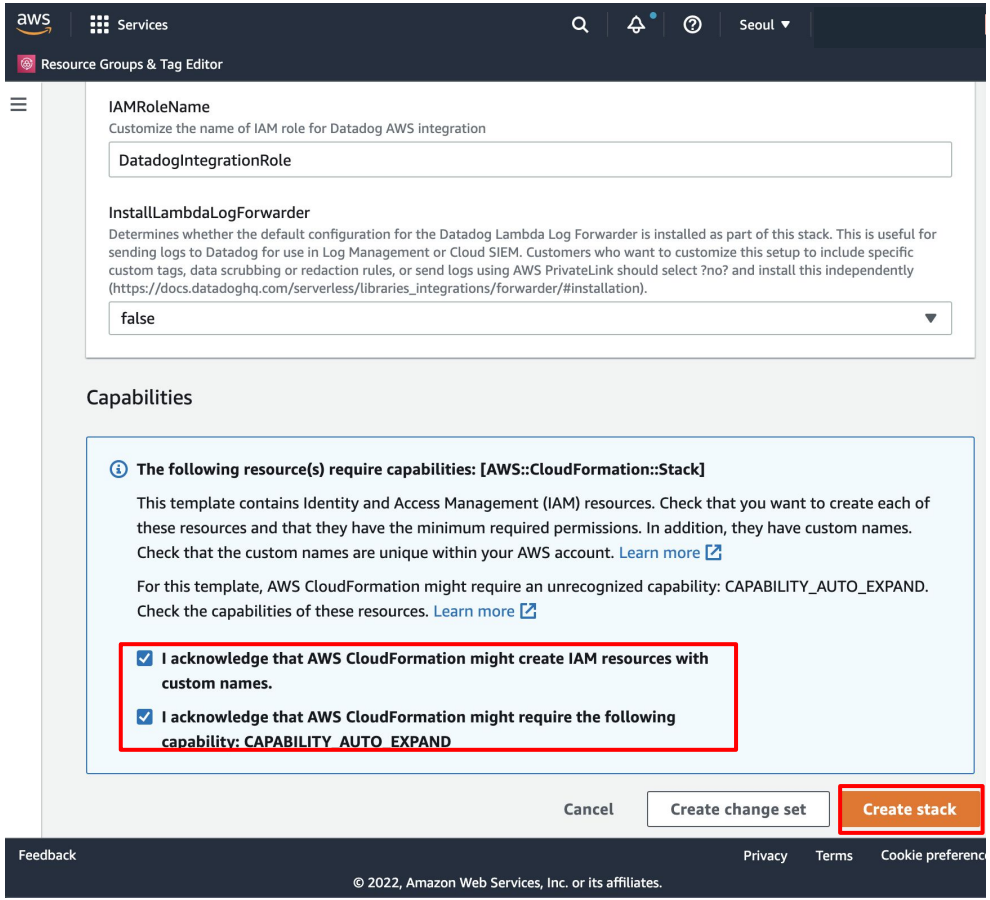
S3나 Cloudwatch의 로그를 수집할 수 있는
기능이지만 시간 관계상 이번 실습에서는 제외합니다
(Lambda Concurrency 100이상 확보 필요)

- ❑ Log Management 체크 해제
- ❑ CloudFormation stack이 생성될 위치 선택
- ❑ Datadog API Key 생성
- ❑ Launch Cloudformation Template 선택



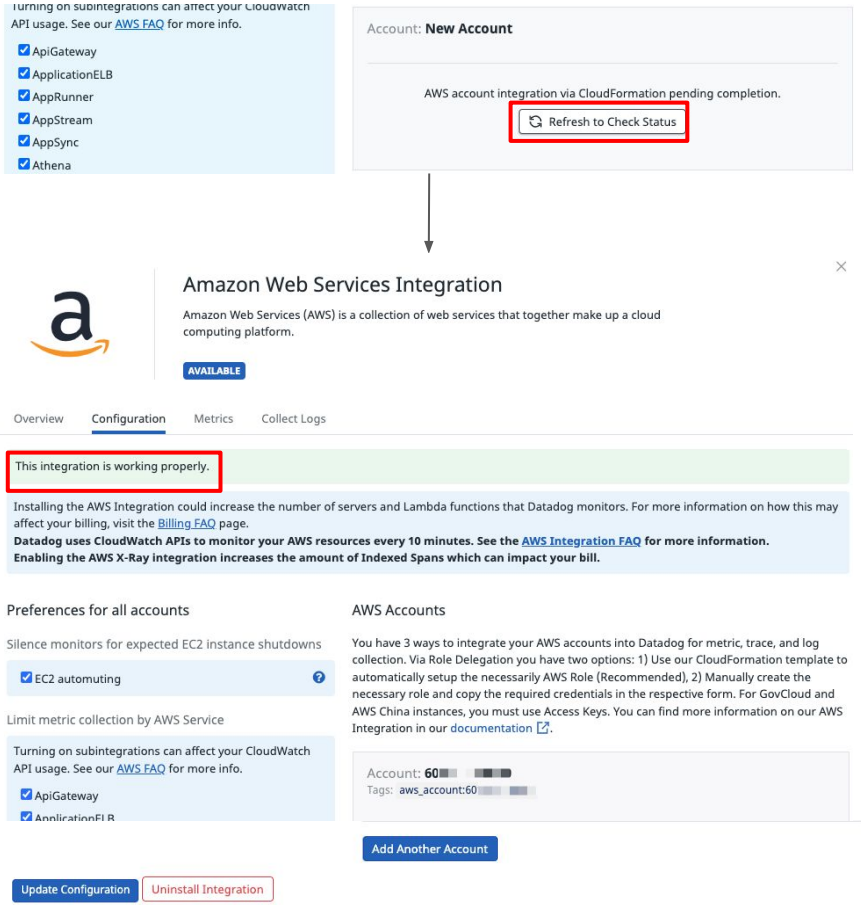
AWS 계정 연동 (Step 3)

- 하단의 체크 박스 선택 후 Create Stack 클릭
- Stack 생성 완료까지 5~10분 정도 소요
- 기대 결과
  - AWS ROLE 생성



AWS 계정 연동 (Step 4)

- Datadog Portal로 돌아와서 Refresh to Check Status 클릭
- This integration is working properly 확인
- 좌측 하단의 Update Configuration 선택
- 기대 결과
  - 계정 연동 완료
  - ELB, RDS 등 AWS Service 리소스 메트릭 자동 수집
  - 약 10분 후부터 AWS 대시보드 확인 가능



2. EKS FullStack 모니터링 연동

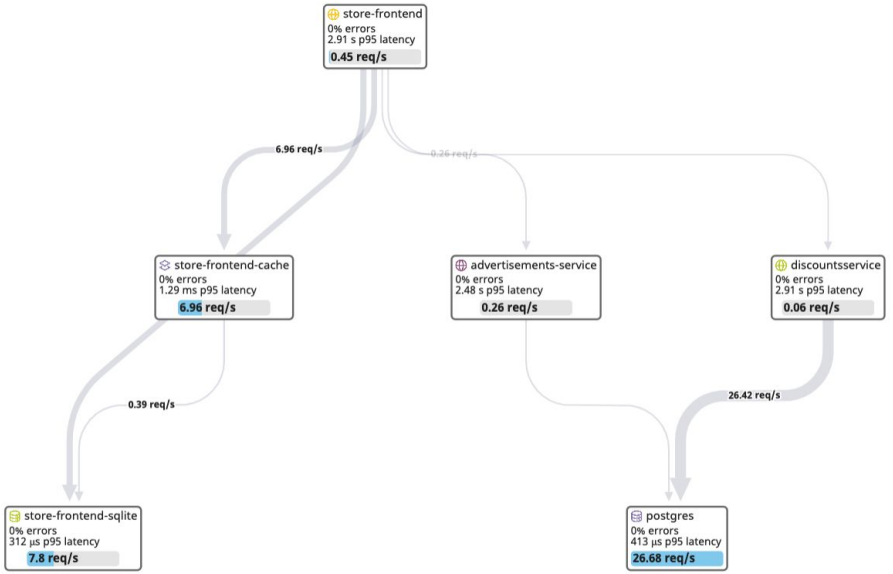
(Metric + Trace + Log + RUM)

왜 필요한가요?

⇒ Kubernetes의 상태 메트릭, Container 사용량, 호스트 사용량, Application 병목 및 로그 분석을 위함

목표: EKS 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. End User의 성능 지표 수집



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**
6. **End User의 성능 지표 수집**

Datadog agent daemonset 배포

1) Daemonset
2) Helm (Recommended)
3) Operator

APM 기능 활성화

1) Application에 Library 삽입
2) Istio를 통한 Trace 전송

프론트 코드에 sdk 초기화 로직 추가

Step1: Datadog Agent 배포하기 (helm)

1. Datadog helm chart 다운로드 (샘플)

wget https://raw.githubusercontent.com/JungYoungseok/ecommerce-workshop/master/deploy/generic-k8s/datadog-helm-chart/values.yaml

(혹은 git clone한 프로젝트의 ecommerce-workshop/master/deploy/generic-k8s/datadog-helm-chart/values.yaml의 파일을 사용합니다)

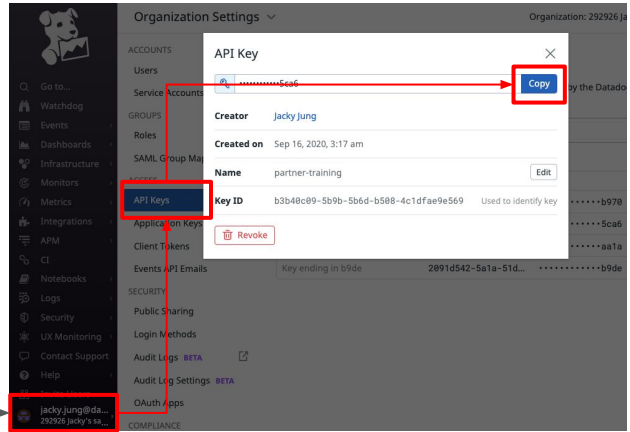
2. Helm Datadog repo 등록

```
helm repo add datadog https://helm.datadoghq.com
helm repo update
```

3. helm install (v3+)

```
helm install datadog-monitoring -f values.yaml  --set datadog.apiKey=<DATADOG_API_KEY> datadog/datadog --set targetSystem=linux
```

설정 ⇒ Organization Setting ⇒API Keys 메뉴에서 확인이 가능합니다



Helm v2는 이 링크를 참고합니다

Datadog Agent 배포하기 (helm)

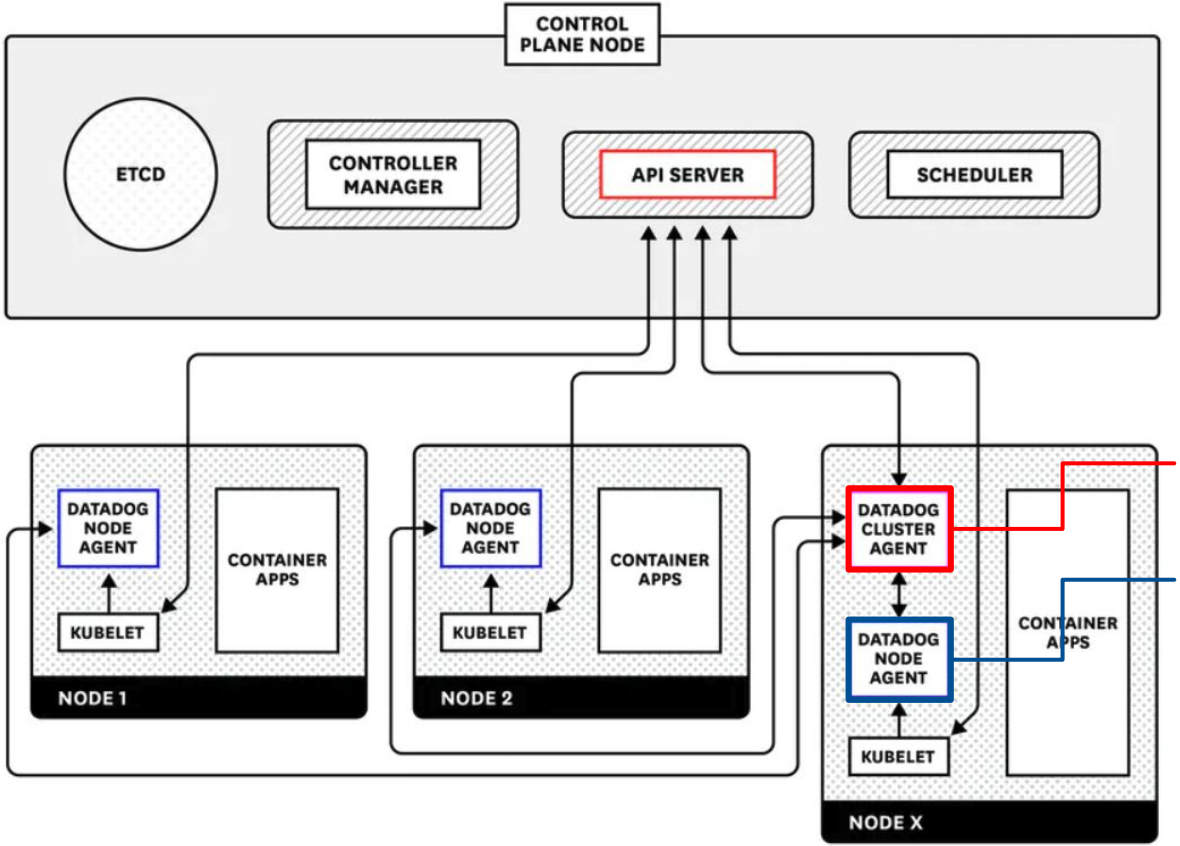
4. 실행 후 화면

```
[root@ip-172-31-31-95 datadog_helm_chart]# helm install datadog-monitoring -f values.yaml  --set datadog.apiKey=5f8e524f
c56e845e8599c785ca6 datadog/datadog --set targetSystem=linux
NAME: datadog-monitoring
LAST DEPLOYED: Sat Apr  9 16:57:45 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Datadog agents are spinning up on each node in your cluster. After a few
minutes, you should see your agents starting in your event stream:
    https://app.datadoghq.com/event/explorer

The Datadog Agent is listening on port 8126 for APM service.
[root@ip-172-31-31-95 datadog_helm_chart]#
```

```
[root@ip-172-31-31-95 datadog_helm_chart]# kubectl get pods
NAME                                                READY   STATUS    RESTARTS   AGE
advertisements-d8bdff946-j48dv                      1/1     Running   0          42h
datadog-monitoring-cluster-agent-7bdc9f944d-2tp89   1/1     Running   0          2m15s
datadog-monitoring-g7tgt                            4/4     Running   0          2m15s
datadog-monitoring-kube-state-metrics-7846f66f94-p6xjf   1/1     Running   0          2m15s
datadog-monitoring-vp7xz                            4/4     Running   0          2m15s
db-6cb7fb6db7-hw7kg                                 1/1     Running   0          44h
discounts-86d6b79d84-6s2ld                          1/1     Running   0          42h
frontend-dbc8768fb-x7hr2                            1/1     Running   0          22h
loadgenerator-v1.1-f4c8b7fc5-2ps6k                  1/1     Running   0          21h
```

Datadog HelmChart 배포 후 아키텍쳐



Deployment로 배포된 Pod로 API서버로의 통신을 대행

Daemonset으로 배포된 Node Agent로 Metric/Trace/log 수집을 담당

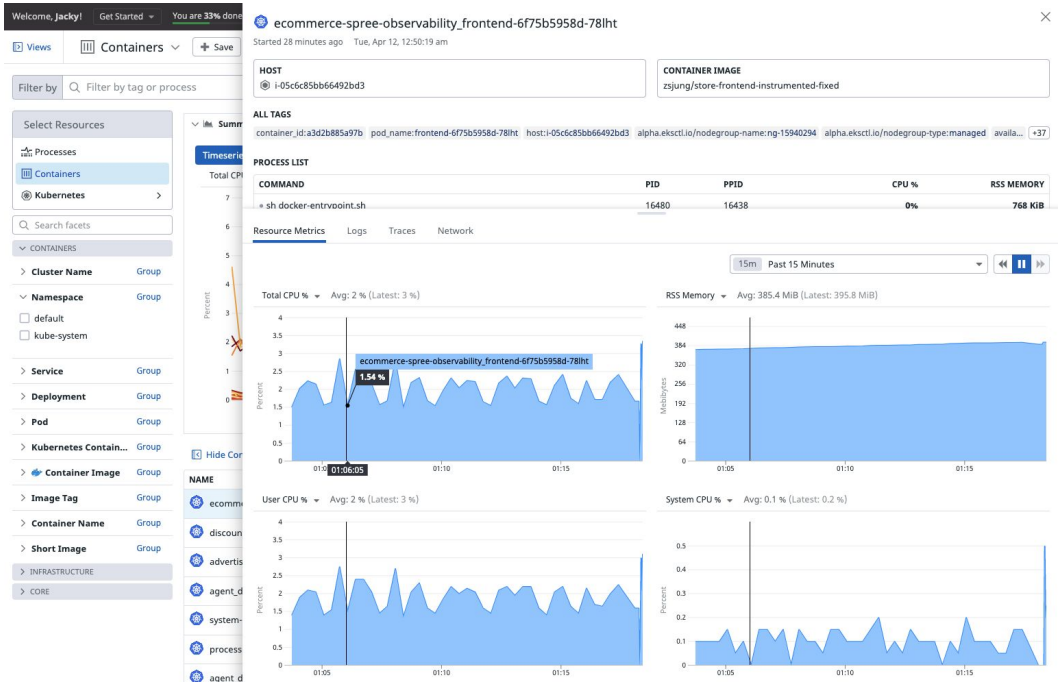
Agent Deploy 후 활성화 된 기능 확인

1. Cluster Node의 가시성 확보

2. Pod/Container 레벨의 가시성 확보

3. Kubernetes Event 수집

4. Application 로그 수집



기본 values.yaml에서 다음의 설정이 미리 적용됨

 # EKS Cluster 이름 설정

clusterName: **aws-isv-workshop**

 # Container로 부터 출력되는 로그를 수집되도록 설정

```
logs:
  enabled: true
  containerCollectAll: true
```

 # apm trace를 **tcp** 방식으로 수집되도록 설정

```
apm:
   socketEnabled: false
   portEnabled: true
```

 # Kubelet과 통신시 **TLS** 인증을 스킵

```
env:
  - name: DD_KUBELET_TLS_VERIFY
    value: "false"
```

 # Process 모니터링 활성화

```
processAgent:
  enabled: true
```

 # Network Monitoring 활성화

```
networkMonitoring:
  enabled: true
```

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. End User의 성능 지표 수집

APM 기능 활성화

1) Application에 Library 삽입

2) Istio를 통한 Trace 전송

APM 설정 부분 (advertisements 서비스)

```
⇒  cat requirements.txt
certifi==2020.11.8
chardet==3.0.4
click==7.1.2
ddtrace==0.46.0
Flask==1.1.2
Flask-SQLAlchemy==2.4.4
idna==2.10
intervaltree==3.1.0
itsdangerous==1.1.0
Jinja2==2.11.2
MarkupSafe==1.1.1
nose==1.3.7
protobuf==3.14.0
psycopg2==2.8.6
Random-Word==1.0.4
requests==2.25.1
six==1.15.0
sortedcontainers==2.3.0
SQLAlchemy==1.3.23
tenacity==6.2.0
urllib3==1.26.3
Werkzeug==1.0.1
```

pip install ddtrace 를 통해 trace 라이브러리 추가

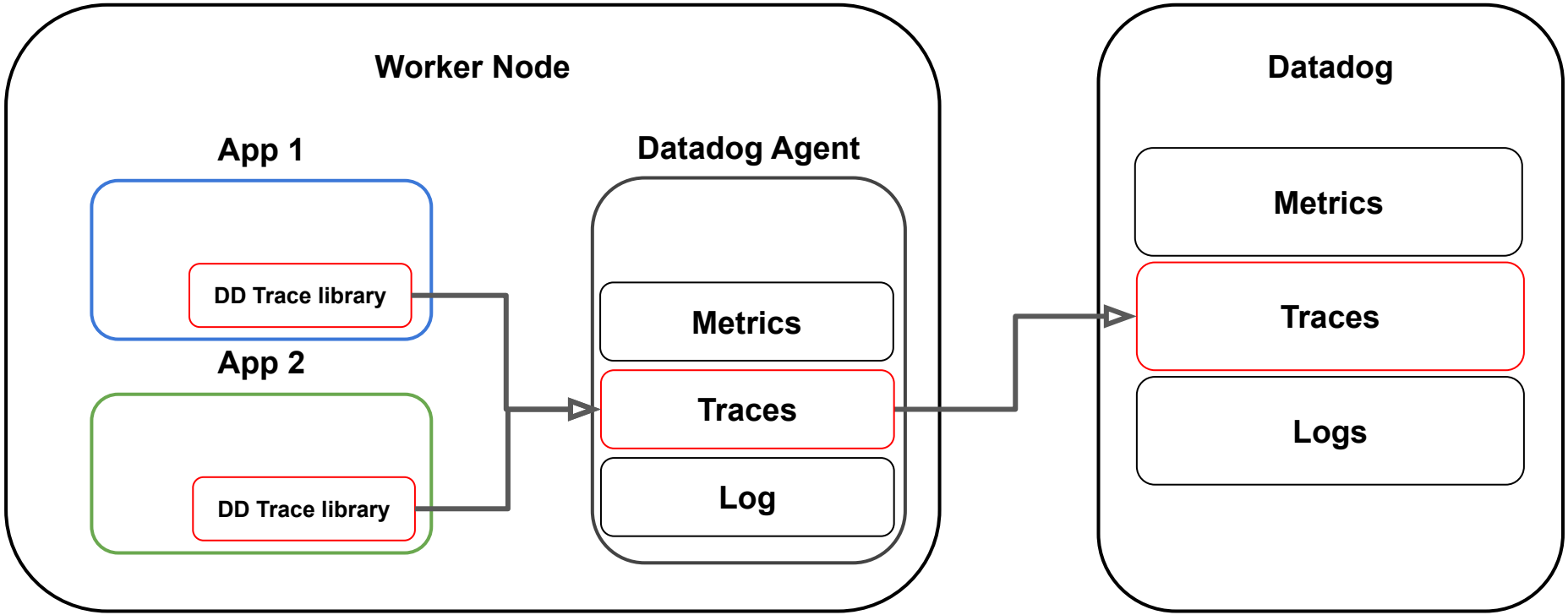
flask 실행시 ddtrace-run wrapper command를 이용해 코드 수정없이 APM 연동

환경 변수를 이용하여 필요한 기능 활성화

advertisements.yaml

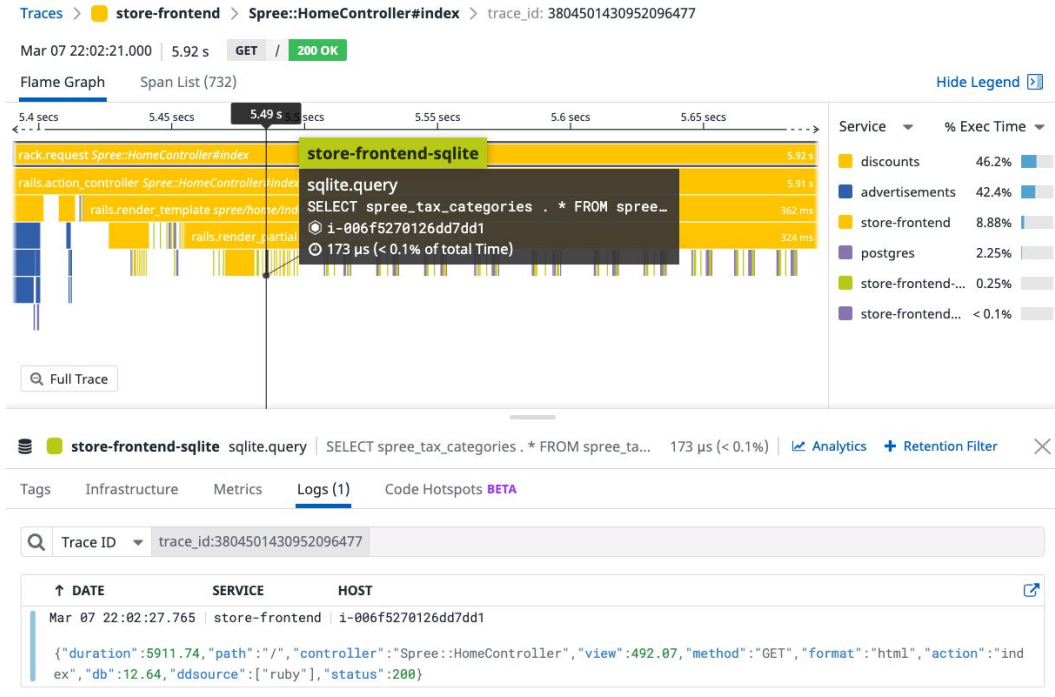
```
spec:
  containers:
  - image: ddtraining/advertisements-service:latest
    name: advertisements
    command: ["ddtrace-run"]
    args: ["flask", "run", "--port=5002", "--host=0.0.0.0"]
    env:
      - name: FLASK_APP
        value: "ads.py"
      - name: FLASK_DEBUG
        value: "1"
      - name: POSTGRES_PASSWORD
        valueFrom:
          secretKeyRef:
            key: pw
            name: db-password
      - name: POSTGRES_USER
        value: "user"
      - name: DD_SERVICE
        value: "advertisements"
      - name: DD_VERSION
        value: "2.0.16"
      - name: DD_AGENT_HOST
        valueFrom:
          fieldRef:
            fieldPath: status.hostIP
      - name: DD_LOGS_INJECTION
        value: "true"
      - name: DD_PROFILING_ENABLED
        value: "true"
    ports:
    - containerPort: 5002
    resources: {}
```

Application Trace 수집 구조



APM 설정 후 활성화 된 기능 확인

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

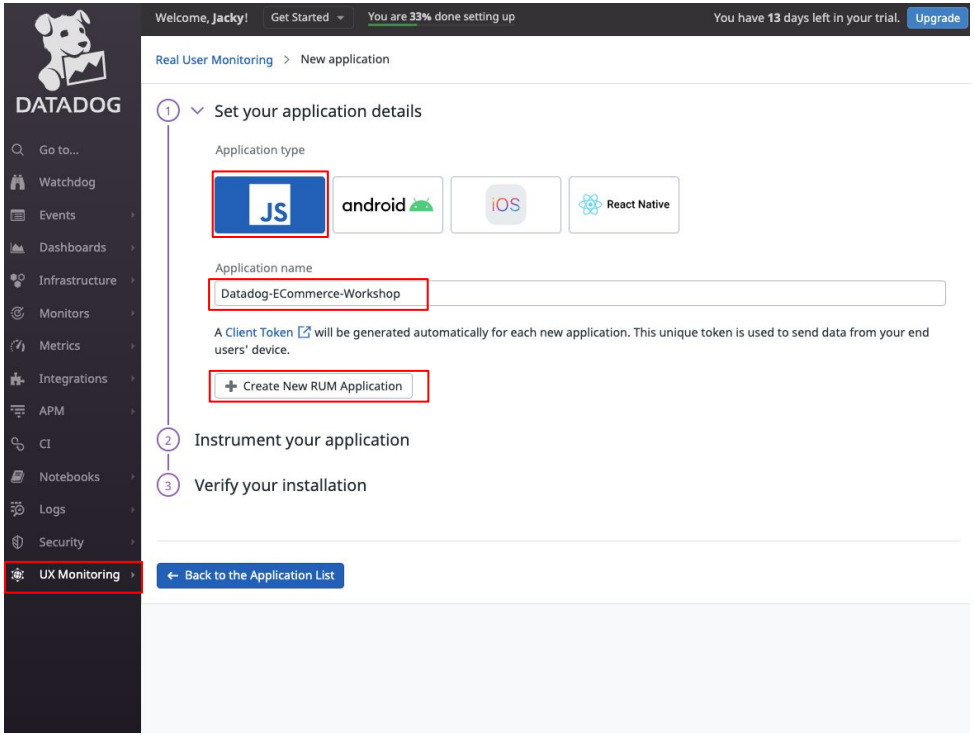


Step3: RUM 설정 (프론트 코드에 SDK 삽입)

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 (Done)
6. End User의 성능 지표 수집

프론트 코드에 sdk 초기화 로직 추가

RUM Application 생성



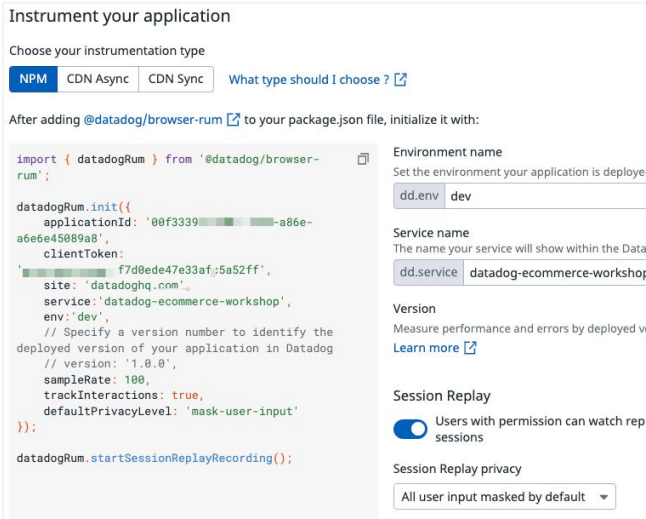
1. **UX Monitoring => RUM Application => New application 선택 (링크)**

2. **JS 선택**

3. **Application name 입력**

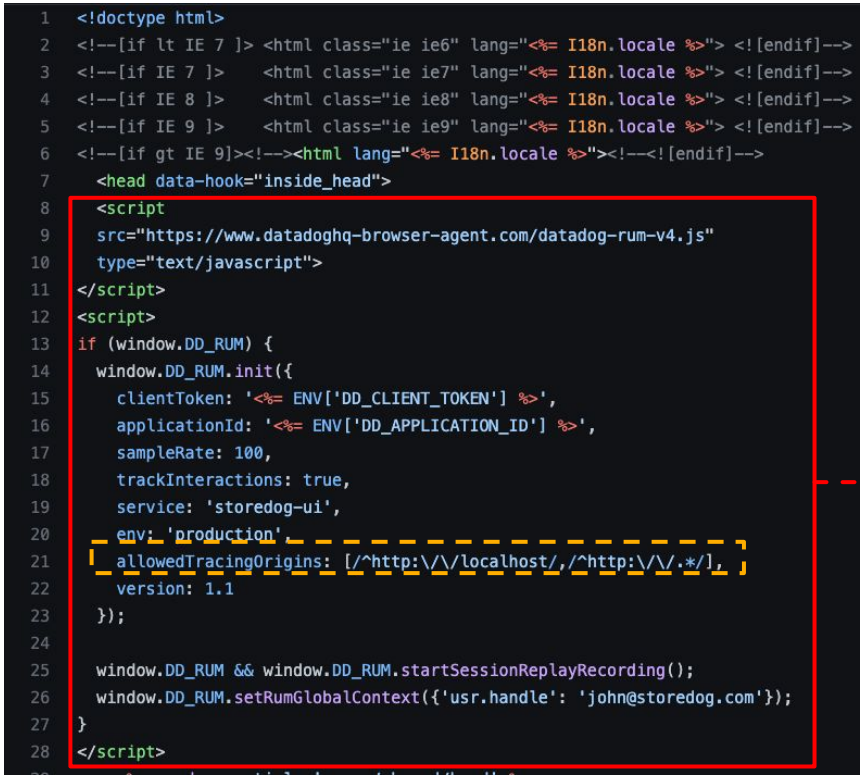
4. **Create New RUM Application 클릭**

5. **아래와 같은 Code가 생성되면 applicationId와 clientToken 을 메모합니다**



RUM (Real User Monitoring) 설정 부분

store-frontend-instrumented-fixed/frontend/app/views/spree/layouts/spree_application.html.erb



```
<!doctype html>
<!--[if lt IE 7 ]> <html class="ie ie6" lang="<%= I18n.locale %>"> <![endif]-->
<!--[if IE 7 ]>    <html class="ie ie7" lang="<%= I18n.locale %>"> <![endif]-->
<!--[if IE 8 ]>    <html class="ie ie8" lang="<%= I18n.locale %>"> <![endif]-->
<!--[if IE 9 ]>    <html class="ie ie9" lang="<%= I18n.locale %>"> <![endif]-->
<!--[if gt IE 9]><!--><html lang="<%= I18n.locale %>"><!--<![endif]-->
  <head data-hook="inside_head">
  <script
  src="https://www.datadoghq-browser-agent.com/datadog-rum-v4.js"
  type="text/javascript">
</script>
<script>
if (window.DD_RUM) {
  window.DD_RUM.init({
    clientToken: '<%= ENV['DD_CLIENT_TOKEN'] %>',
    applicationId: '<%= ENV['DD_APPLICATION_ID'] %>',
    sampleRate: 100,
    trackInteractions: true,
    service: 'storedog-ui',
    env: 'production',
    allowedTracingOrigins: [/^http:\/\/localhost/,/^http:\/\/.*/],
    version: 1.1
  });

  window.DD_RUM && window.DD_RUM.startSessionReplayRecording();
  window.DD_RUM.setRumGlobalContext({'usr.handle': 'john@storedog.com'});
}
</script>
```

- 사용자 성능 추적이 필요한 페이지에 Datadog RUM 초기화 코드 삽입 필요
- 공통 View layout에 추가하여 대부분의 페이지에 대해 트래킹이 되도록 처리
- allowedTracingOrigins 옵션에 허용 Origin을 추가하여 RUM⇔APM 연계 분석 구성

아래 수정 내용은 워크샵 시간 관계상 container 이미지에 미리 적용해 놓았습니다.
(image: zsjung/store-frontend-instrumented-fixed:v1.7)

RUM (Real User Monitoring) 설정 부분 - 변경 필요

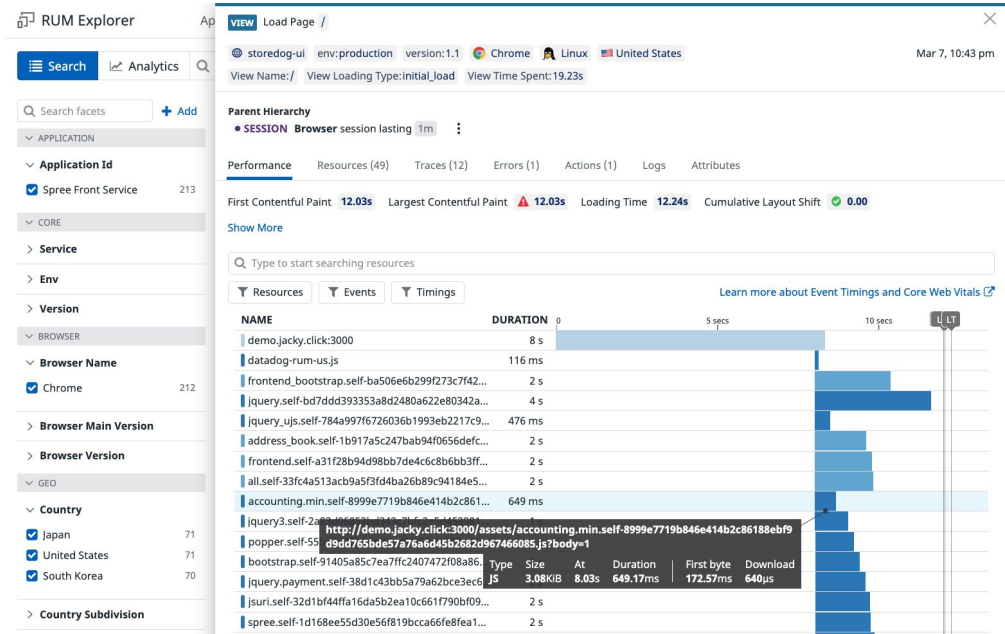
 # frontend.yaml

```
  - name: DD_AGENT_HOST
    valueFrom:
      fieldRef:
        fieldPath: status.hostIP
  - name: DD_LOGS_INJECTION
    value: "true"
  - name: DD_ENV
    valueFrom:
      fieldRef:
        fieldPath: metadata.labels['tags.datadoghq.com/env']
  - name: DD_SERVICE
    value: store-frontend
  - name: DD_VERSION
    value: "1.5.3"
  - name: DD_LOGS_INJECTION
    value: "true"
  - name: DD_CLIENT_TOKEN
    value: pubcbf8b3497047ad[redacted]
  - name: DD_APPLICATION_ID
    value: b0fd557b-57f7-4ed5[redacted]
  image: zsjung/store-frontend-instrumented-fixed:v1.7
  imagePullPolicy: Always
  name: ecommerce-spree-observability
```

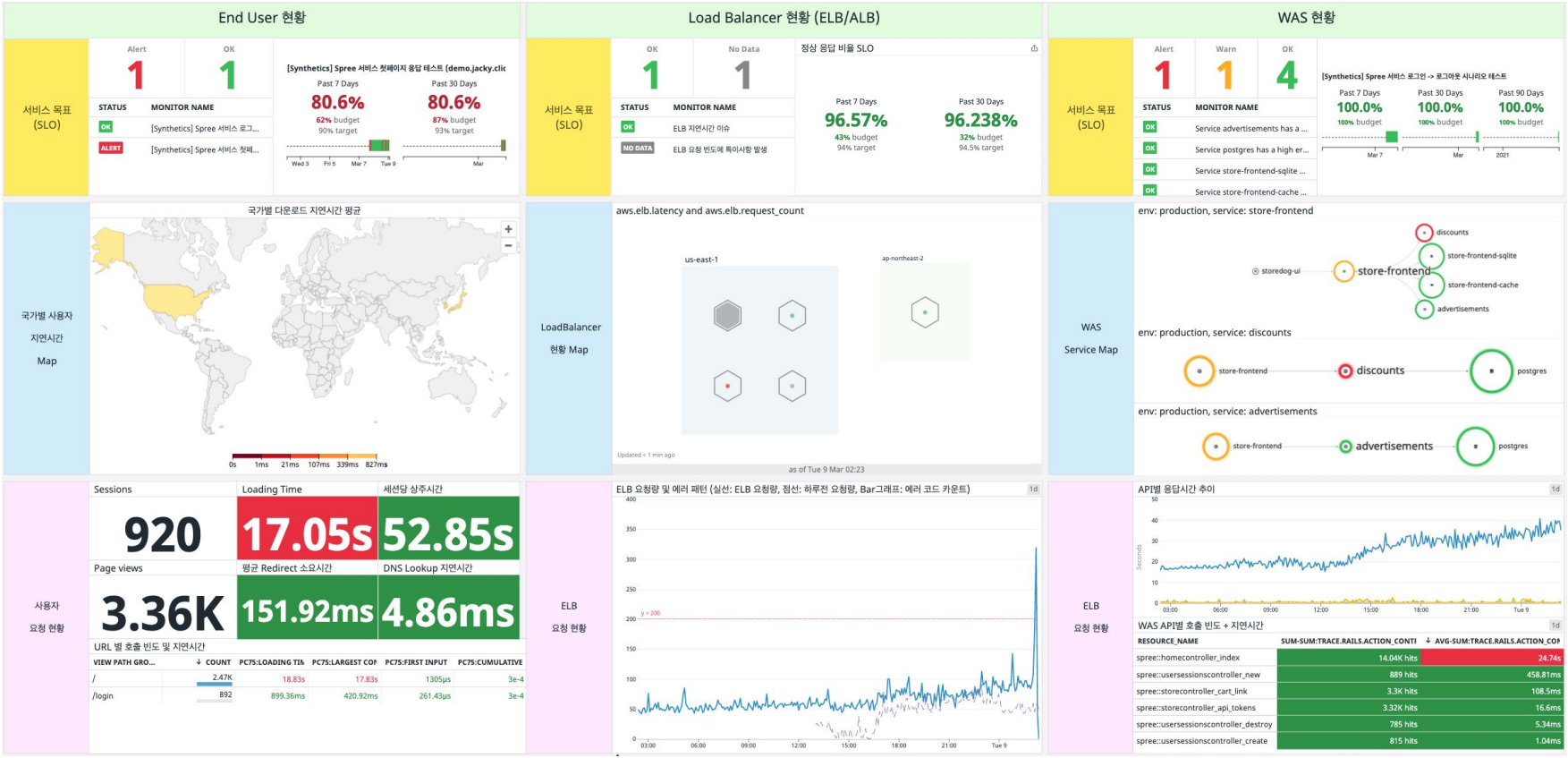
Datadog 포털의 RUM Application 생성 후 획득한 **ClientToken**과 **applicationId**를 frontend.yaml에 적용합니다

RUM 설정 후 활성화 된 기능 확인 (Demo)

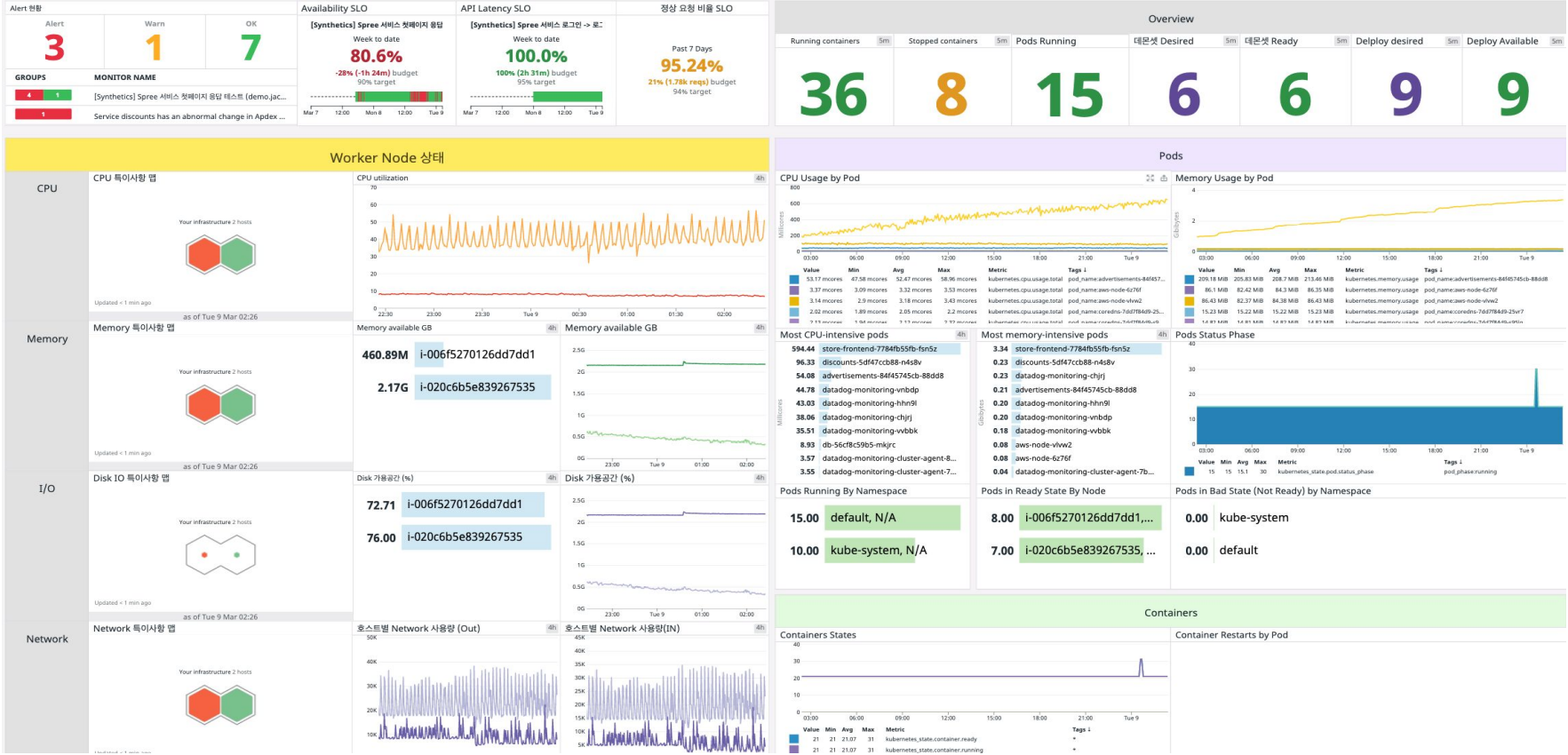
1. Cluster Node의 가시성 확보
2. Pod/Container 레벨의 가시성 확보
3. Kubernetes Event 수집
4. Application 로그 수집
5. Application의 Trace 수집
6. End User의 성능 지표 수집



서비스 관점의 FullStack Visibility Dashboard 샘플



Kubernetes 관점 FullStack Dashboard 샘플

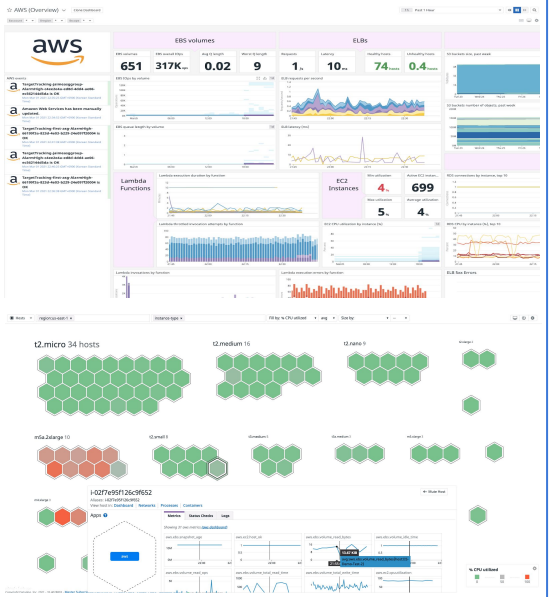


세션 요약

Datadog의 가치는

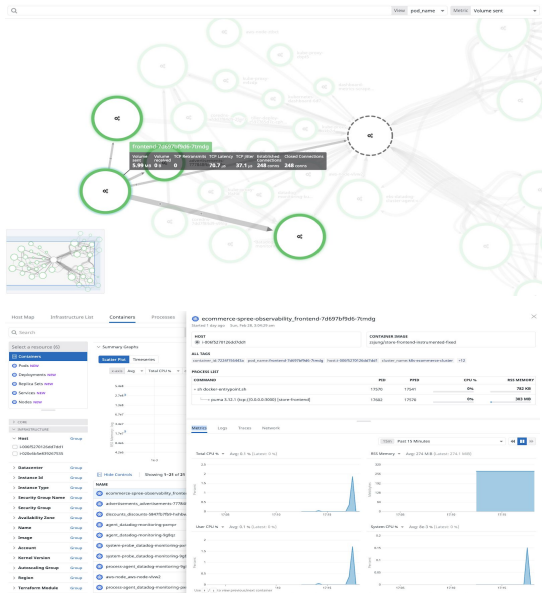
쉽게 FullStack Data를 연동하고 Powerful한 분석 환경을 통해 MTTR를 최소화 하는것

1. AWS 계정 연동



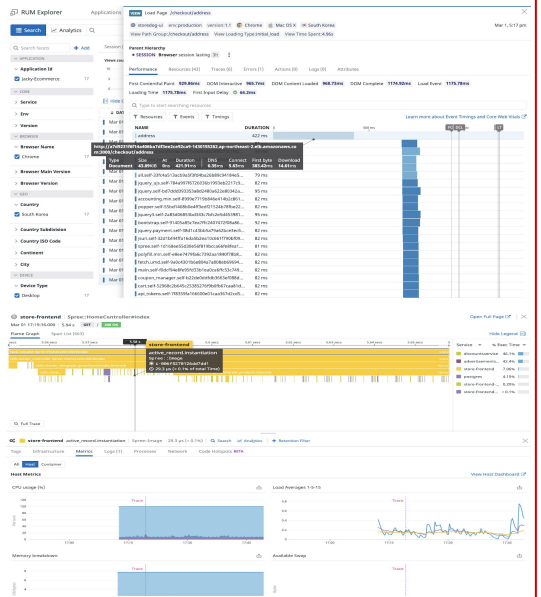
AWS 환경의 리소스를 효과적으로 시각화

2. Helm chart 배포



EKS의 Node/Pod/Container에 대한 가시성 확보

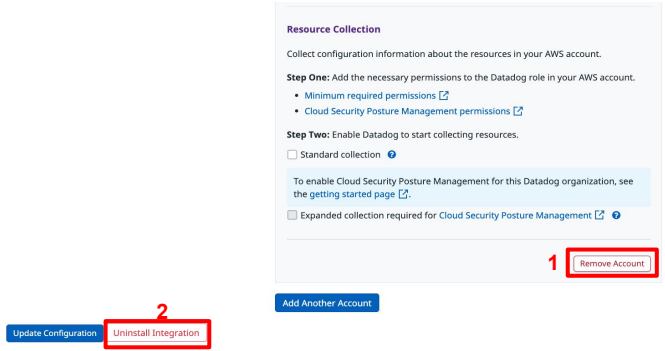
3. Application에 Datadog SDK 연동



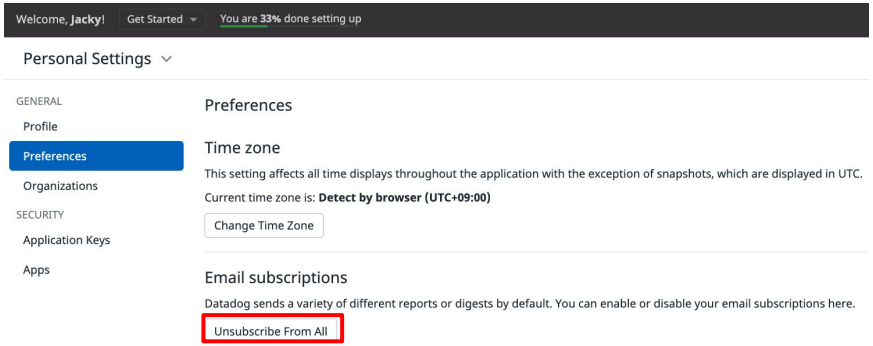
End User부터 Backend까지의 FullStack 가시성 확보

Resource Clean up

1. AWS 계정 연동 해제 (링크)



2. Datadog 계정에서 Daily/Weekly Report 해제



3. EKS 리소스 정리

```
[root@ip-172-31-31-95 ecommerce-app]# helm delete datadog-monitoring
```

```
[root@ip-172-31-31-95 ecommerce-app]# ls
advertisements.yaml  db.yaml  discounts.yaml  frontend.yaml
[root@ip-172-31-31-95 ecommerce-app]# kubectl delete -f .
```

4. EKS 클러스터 삭제

워크샵 용으로 생성하신 **EKS** 클러스터는 정리해주세요

```
eksctl delete cluster --name <클러스터명>
```

감사합니다

DATADOG

TECH ISV 2022 SUMMIT