



DATADOG 교육 세션

Kafka Monitoring

Agenda

Kafka Monitoring Overview

Kafka Host Monitoring + 로그 수집

Kafka Metric Integration

APM을 이용한 Kafka 트랜잭션 추적

참고 Blog

monitoring guide / infrastructure monitoring / apm / log management / apache / kafka / zookeeper / message queue



Part 1: Monitoring Kafka performance metrics

Part 2: Collecting Kafka performance metrics

Part 3: Monitoring Kafka with Datadog

Integrating Datadog, Kafka, and ZooKeeper

Monitoring your Kafka deployment in Datadog

Get started monitoring Kafka with Datadog

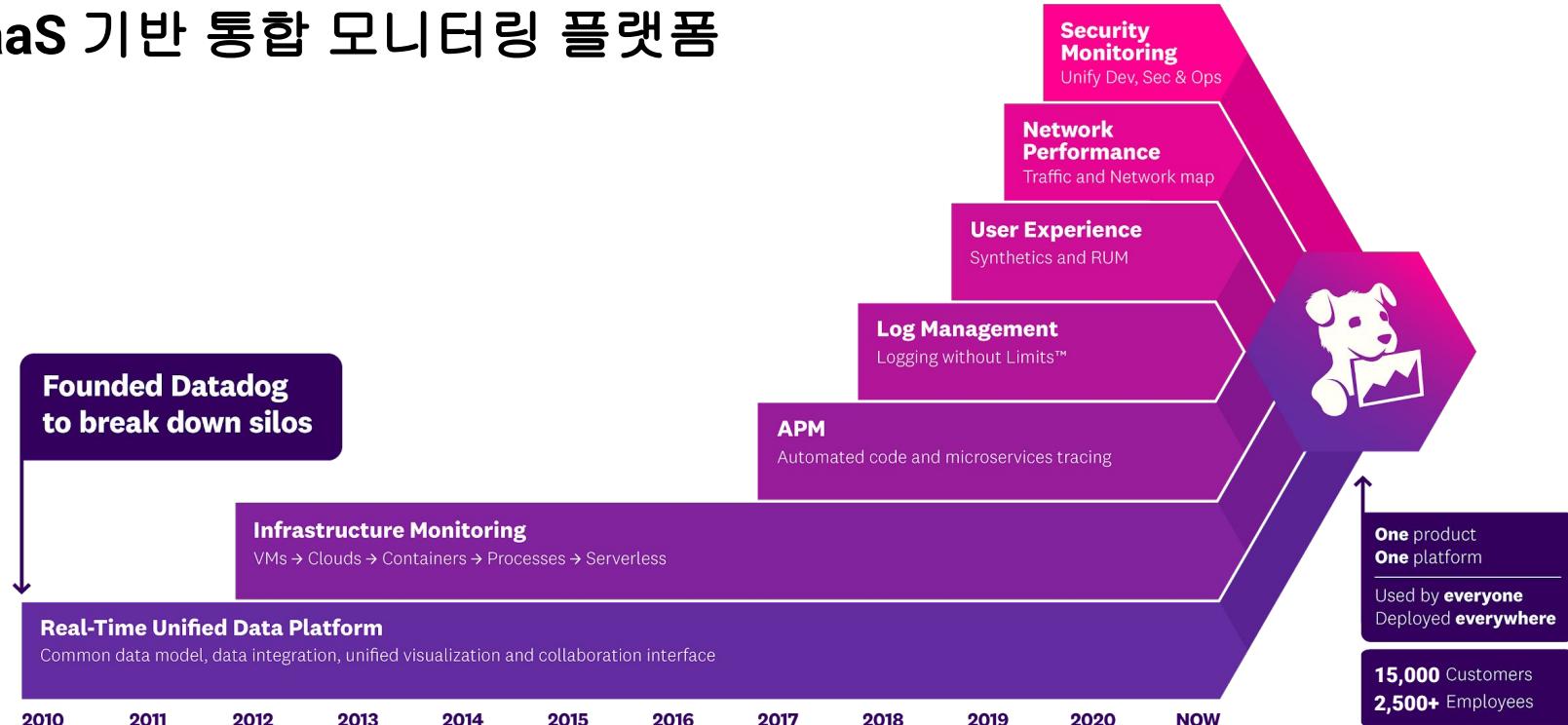
Kafka deployments often rely on additional software packages not included in the Kafka codebase itself—in particular, Apache ZooKeeper. A comprehensive monitoring implementation includes all the layers of your deployment so you have visibility into your Kafka cluster and your ZooKeeper ensemble, as well as your producer and consumer applications and the hosts that run them all. To implement ongoing, meaningful monitoring, you will need a platform where you can collect and analyze your Kafka metrics, logs, and distributed request traces alongside monitoring data from the rest of your infrastructure.

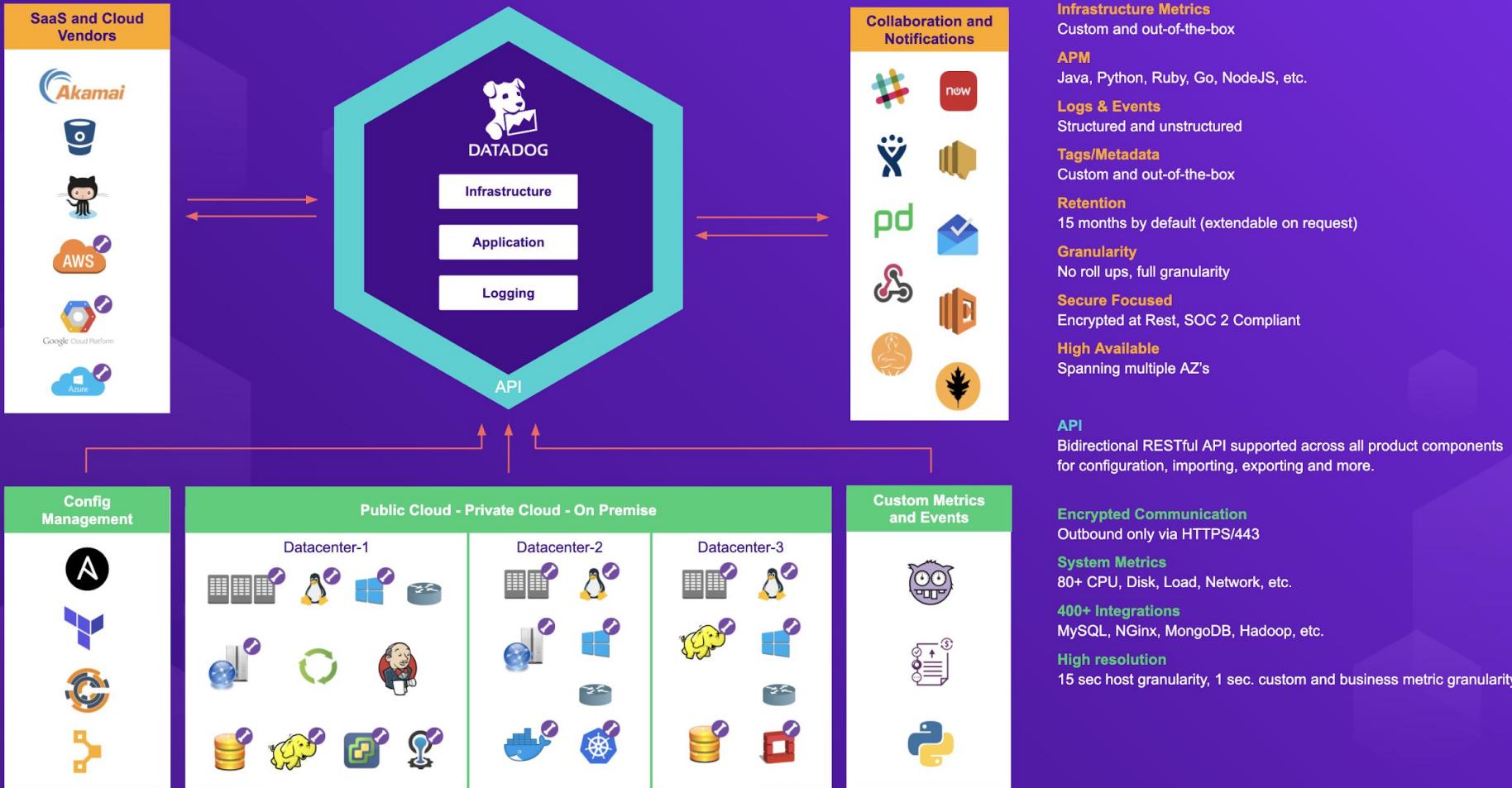


<https://www.datadoghq.com/blog/monitor-kafka-with-datadog/>

Datadog 소개

SaaS 기반 통합 모니터링 플랫폼





DATADOG

Kafka Monitoring Overview

공통 수집 사항

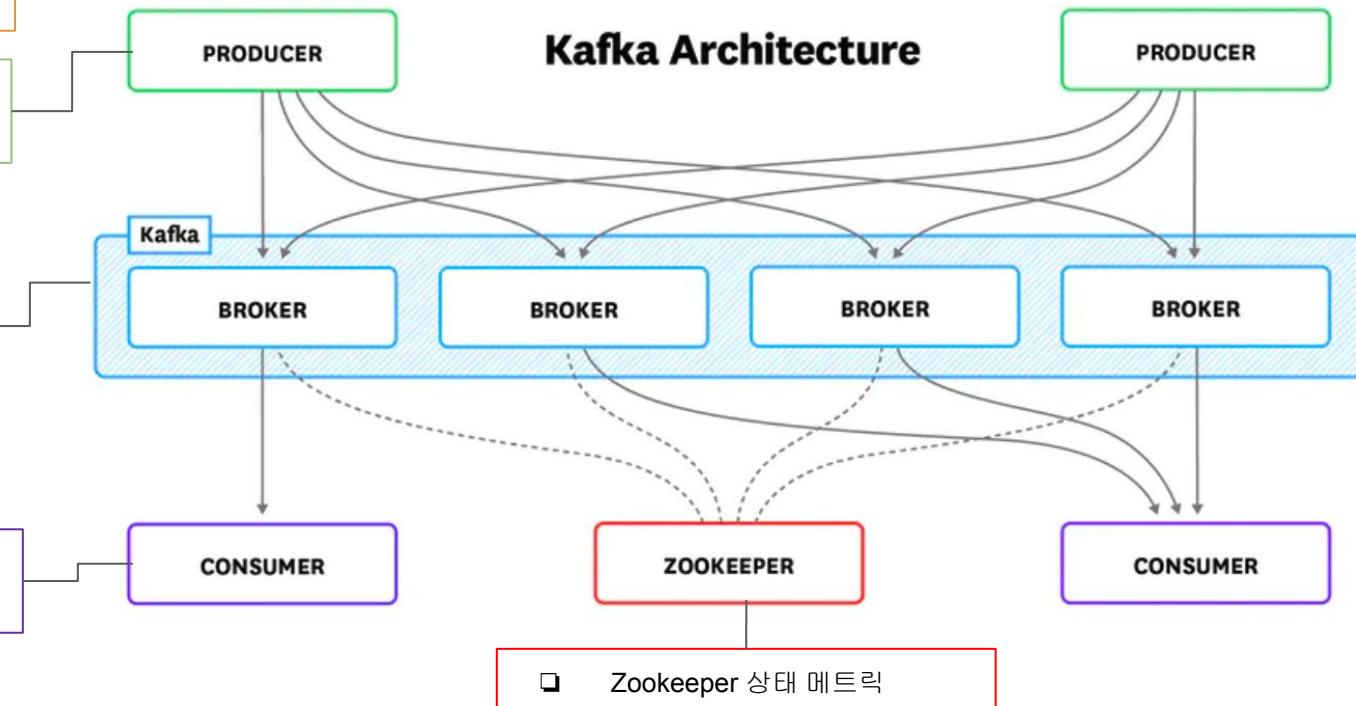
- Host 메트릭
- Log 수집
- Network Flow 수집

- Producer 상태 메트릭
- Producer 분산 트랜잭션 수집

- Kafka Broker 상태 메트릭

- Consumer 상태 메트릭
- Consumer 분산 트랜잭션 수집

Kafka Architecture





DATADOG

1. Host 모니터링 + 로그 수집

Host 메트릭 수집

API Key를 포함한 싱글 커맨드를 복사 후 Datadog agent 설치

Integrations Marketplace APIs Agent Agent Embeds

See instructions for [Agent 6](#) or [Agent 5](#) instead

Agent 7 Installation Instructions

 Overview
 Mac OS X
 Windows
 Debian
 Ubuntu
 Amazon Linux
 CentOS/Red Hat
 Fedora
 SUSE
 AIX
 CoreOS
 Docker
 Kubernetes
 OpenShift
 Chef
 Puppet
 Ansible
 SaltStack
 Cloud Foundry
 Chocolatey
 Heroku


Installing on Ubuntu

The Datadog Agent has x86_64 and arm64 (ARM v8) packages. For other architectures, use the [source install](#).

New installation

1 Use our easy one-step install.

```
DD_SITE="datadoghq.com" bash -c "$(curl -L https://s3.amazonaws.com/dd-agent/scripts/install_script.sh)"
```

This will install the APT packages for the Datadog Agent and will prompt you for your password.
If the Agent is not already installed on your machine and you don't want it to start automatically after the installation, just prepend `DD_INSTALL_ONLY=true` to the above script before running it.

Upgrade from Agent 6 or 5

Agent 7 only supports Python 3. Before upgrading, confirm that any custom checks you have are compatible with Python 3. See [this guide](#) for more information. If you're not using custom checks or have already confirmed their compatibility, you can upgrade now using one of the commands below.

1 If you're upgrading from Agent 6.

```
DD_AGENT_MAJOR_VERSION=7 DD_API_KEY= DD_SITE="datadoghq.com" bash -c "$(curl -L https://s3.amazonaws.com/dd-agent/scripts/install_script.sh)"
```

This will install the agent, similarly to what is described above. The existing Agent 6 configuration will be used by Agent 7.

2 If you're upgrading from Agent 5.17+.

```
DD_AGENT_MAJOR_VERSION=7 DD_UPGRADE=true DD_SITE="datadoghq.com" bash -c "$(curl -L https://s3.amazonaws.com/dd-agent/scripts/install_script.sh)"
```

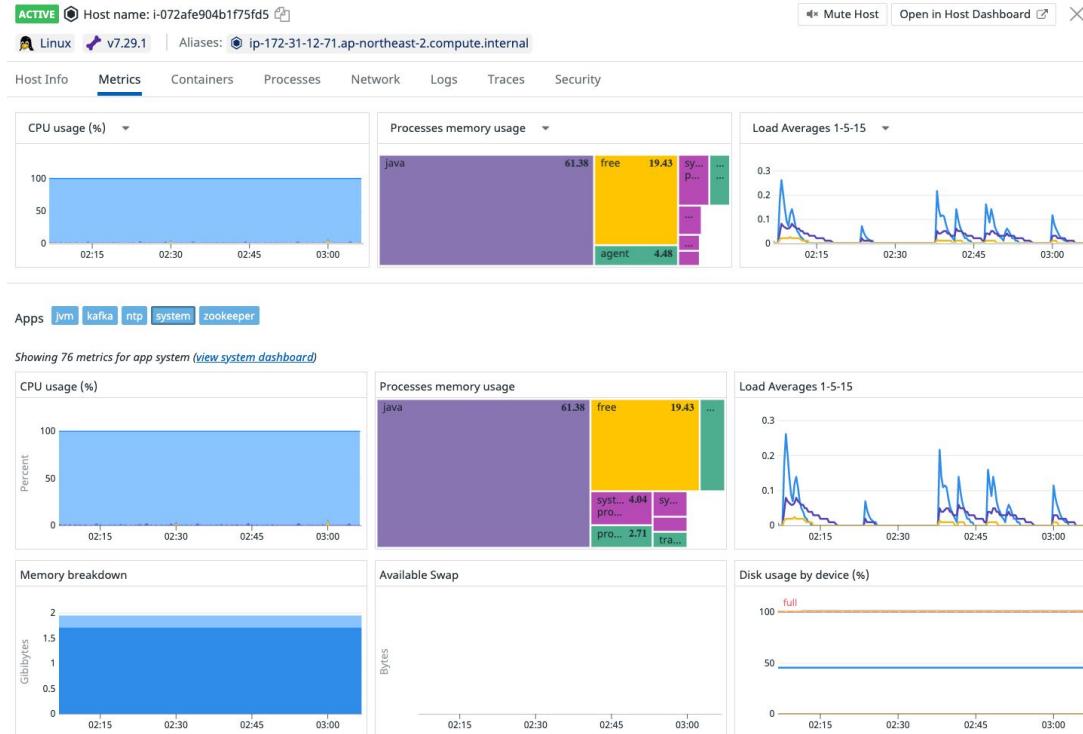
This will install the agent, similarly to what is described above, but we will also import your existing Agent 5 configuration so that you can get up and running immediately.

Note: the import process won't automatically move custom checks, this is by design since we cannot guarantee full backwards compatibility out of the box.

If you prefer to see the installation step-by-step, [click here](#).

Host 메트릭 수집

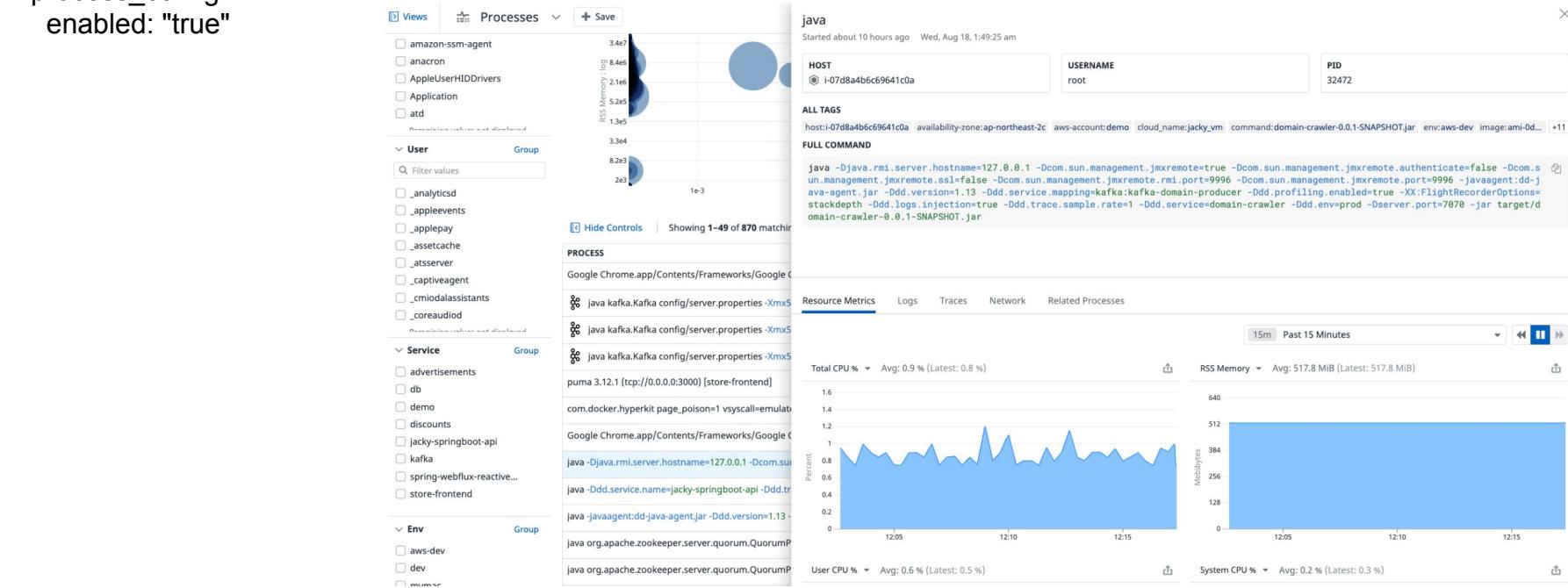
Agent 설치 후 CPU/Memory/Disk 등의 기본 시스템 메트릭 수집



Process 모니터링 활성화

1. /etc/datadog-agent/datadog.yaml에서 Process 기능 활성화 후 재시작

process_config:
enabled: "true"



참고 링크

Network 모니터링 활성화

1. /etc/datadog-agent/system-probe.yaml 에 network 기능 활성화

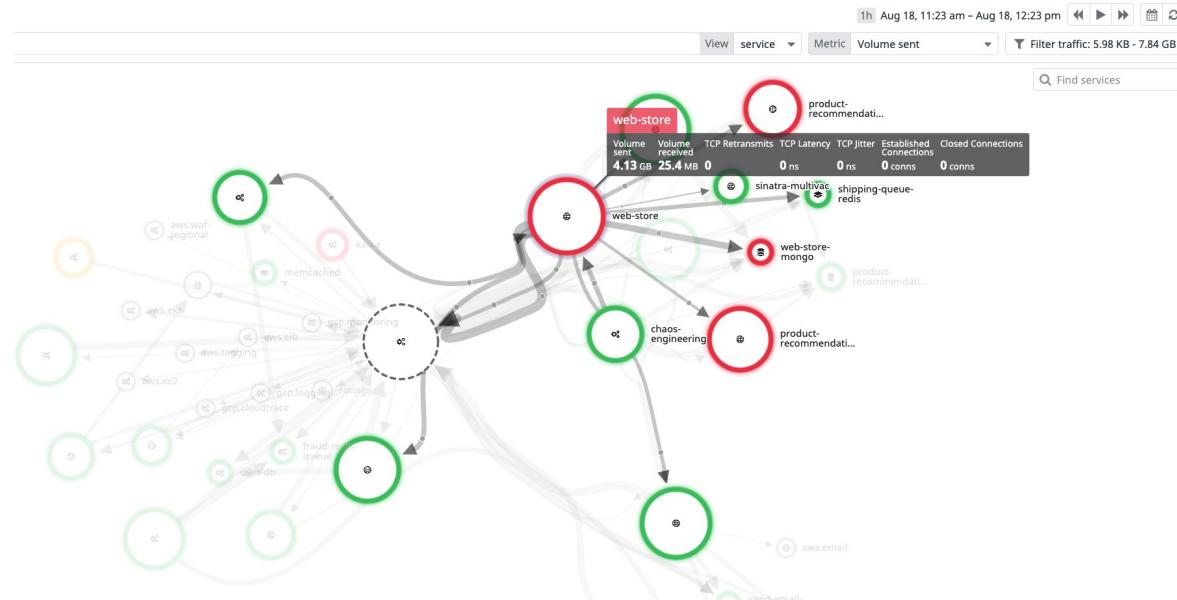
```
network_config:  
  enabled: true
```

2. system-probe 서비스 등록 및 시작

```
sudo systemctl start datadog-agent-sysprobe  
sudo systemctl enable datadog-agent-sysprobe
```

3. Datadog agent 재시작

```
sudo systemctl restart datadog-agent
```



로그 수집 설정

1. /etc/datadog-agent/datadog.yaml 에 로그 수집 기능 활성화

```
#####
## Log collection Configuration ##
#####

## @param logs_enabled - boolean - optional - default: false
## Enable Datadog Agent log collection by setting logs_enabled to true.
#
logs_enabled: true
```

2. 로그 수집을 희망하는 파일과 tag를 입력한 후 datadog agent 재시작하면 로그 수집 시작

```
logs:
  - type: file
    path: /home/ubuntu/kafka_2.12-2.8.0/logs/*.log
    source: kafka
    tags:
      - cluster:order-kafka
      - project:apple
      - kafka-index:3
      - broker:3
```

로그 설정 확인

3. datadog-agent status 명령어를 이용하여 로그 설정이 잘 되었는지 확인

```
root@ip-172-31-13-174:/home/ubuntu# datadog-agent status
Getting the status from the agent.

=====
Logs Agent
=====

    Sending compressed logs to HTTPS to agent-http-intake.logs.datadoghq.com on port 443
    BytesSent: 2.006076e+06
    EncodedBytesSent: 331420
    LogsProcessed: 5542
    LogsSent: 5542

kafka
-----
  - Type: file
    Path: /home/ubuntu/kafka_2.12-2.8.0/logs/*.log
    Status: OK
      8 files tailed out of 8 files matching
    Inputs:
      /home/ubuntu/kafka_2.12-2.8.0/logs/kafka-authorizer.log
      /home/ubuntu/kafka_2.12-2.8.0/logs/controller.log
      /home/ubuntu/kafka_2.12-2.8.0/logs/zookeeper-gc.log
      /home/ubuntu/kafka_2.12-2.8.0/logs/state-change.log
      /home/ubuntu/kafka_2.12-2.8.0/logs/server.log
      /home/ubuntu/kafka_2.12-2.8.0/logs/log-cleaner.log
      /home/ubuntu/kafka_2.12-2.8.0/logs/kafkaServer-gc.log
      /home/ubuntu/kafka_2.12-2.8.0/logs/kafka-request.log
    BytesRead: 562399
    Average Latency (ms): 1
    24h Average Latency (ms): 1
    Peak Latency (ms): 720
    24h Peak Latency (ms): 720
    MultiLine matches: 5542
```

로그 수집 확인

4. Tag 및 로그 메시지를 이용하여 검색 혹은 시각화 가능

The screenshot shows a log search interface with the following components:

- Views, Logs, Save**: Top navigation buttons.
- Search bar**: Contains the query `source:kafka`.
- Aggregate by**: A dropdown menu with options: Fields, Patterns, Transactions.
- Facets**:
 - Search facets: Showing 63 of 145 results.
 - Source facet: kafka (1.28k), vpc, java, advertisements-service, discounts-service, nginx, aws-for-fluent-bit, cluster-agent.
 - Host facet: host.
 - Service facet: service.
- Visualize as**: Buttons for Hide Controls, List, Timeseries, Top.
- Watchdog Insights BETA**: A section displaying log entries from a Kafka broker.
- Log Entry Details**:
 - INFO Aug 18, 2021 at 03:19:27.732 (a minute ago)**
 - HOST**: i-01ae038039f335fb1
 - SOURCE**: kafka
 - ALL TAGS**: broker:1, broker:broker-1, cluster:order-kafka, datadog:index:main, dirname:/home/ubuntu/kafka_2.12-2.8.0/logs, env:dev, filename:server.log, project:apple, service:kafka, source:kafka
 - Processing stat command from /127.0.0.1:52606**
 - Event Attributes**:
 - LOGGER NAME: org.apache.zookeeper.server.NIOServerCnxn
 - status: INFO
 - timestamp: 1629224367732

3. [로그 기반] 특정 에러 로그 발견시 트러블 슈팅

1) 알람 조건 설정

1 Define the search query

Env:prod X ERROR X Redis connection failed

Count * group by (everything) ▾

Simple alert, triggers a single alert when a threshold is reached. Group by a facet to enable separate alerts

2 Set alert conditions

Trigger when the metric is above the threshold during the last 5 minutes

Alert threshold: > 5

Warning threshold: > 1

Delay evaluation by 0 seconds

3 Say what's happening

Edit Preview

[Jacky] Redis connection error log 발생 중

Redis connection failed 로그가 빈번하게 발생되고 있습니다. 다음의 대시보드를 참고하여 확인 바랍니다.

[Redis 상태 확인 대시보드](<https://app.datadoghq.com/dashboard/8xd-gby-nfq/jacky-service>)

@slack-jacky-demo-alert

Select value renotify if the monitor has not been resolved

Tags: env:prod X status:error X

Priority: P2 (High)

2) 문제 로그 확인

ERROR Aug 06, 2021 at 10:17:34.710 (7 minutes ago) View in Context Export X

HOST	SERVICE	SOURCE
I-03325904a4bf8dd67	web-store	ruby

CONTAINER NAME DOCKER IMAGE POD NAME

k8s_rails-storefront_rails-s...	172597598159.dkr.ecr.us-east-...	rails-storefront-7fcddd5b4b-9...
---------------------------------	----------------------------------	----------------------------------

ALL TAGS

availability-zone:us-west-2b aws_account:172597598159
aws_autoscaling_groupname:eks-249cfc17-53f3-1eda-59b4-c3f7439a8d7d
aws_ec2_fleet-id:fleet-f643a4c9-5dd5-c813-0c12-83a83b324b7e aws_ec2launchtemplate_id:lt-0cf1c15... +43

Redis connection failed, requeueing shipping job.

Event Attributes Trace (1) Metrics Processes

LOGGER NAME ShoppingCartController

```
{
    cart_value      35
    ddtags        version:0.6.2
    level          error
    merch_detail {
        merchant_name Giovanni's Great Rooms
        merchant_poc Tom Haverford
    }
    shopist {
        webstore {
            cart {
                checked_out   true
                created_at   2021-08-06T01:17:33.636Z
                customer_id {
                    id 619c8d6dd69929000ca55c86
                }
                del         false
                id {
                    id 619c8dad1edb8f73fe18cf33
                }
            }
        }
    }
}
```

3) 문제 로그가 발생된 Application 위치 확인

ERROR Aug 06, 2021 at 10:17:34.710 (7 minutes ago) View in Context Export X

HOST	SERVICE	SOURCE
I-03325904a4bf8dd67	web-store	ruby

CONTAINER NAME DOCKER IMAGE POD NAME

k8s_rails-storefront_rails-s...	172597598159.dkr.ecr.us-east-...	rails-storefront-7fcddd5b4b-9...
---------------------------------	----------------------------------	----------------------------------

ALL TAGS

availability-zone:us-west-2b aws_account:172597598159
aws_autoscaling_groupname:eks-249cfc17-53f3-1eda-59b4-c3f7439a8d7d
aws_ec2_fleet-id:fleet-f643a4c9-5dd5-c813-0c12-83a83b324b7e aws_ec2launchtemplate_id:lt-0cf1c15... +43

Redis connection failed, requeueing shipping job.

Event Attributes Trace (1) Metrics Processes

web-store | ShoppingCartController#checkout View Trace Details

Aug 06 10:17:33.222 | 1.53 s POST /checkout 200 OK

Log

rack.request ShoppingCartController#checkout 1.53s
aspnet_core.request POST check-token 822 ms
rails.action_controller.ShoppingCartController#checkout 1.53s
django.request 1.53s
django.middlewares.common.BrowserLocaleMiddleware 1.53s
django.middlewares.common.CommonMiddleware 1.53s
django.middlewares.common.GZipMiddleware 1.53s
django.middlewares.common.SessionMiddleware 1.53s
django.middlewares.csrf.CsrfViewMiddleware 1.53s
django.middlewares.security.SecurityMiddleware 1.53s
django.view_apiv2.CheckTokenView 1.53s
django.view_apiv2.CheckTokenView 1.53s
django.view_apiv2.CheckTokenView 1.53s
django.view_apiv2.CheckTokenView 1.53s

Service % Exec Time

- auth-dotnet 53.0%
- email-api-py 22.6%
- web-store 11.0%
- api.payment.com 3.63%
- web-store-monitor 3.25%
- payments-go 1.56%
- auth-dotnet-post 0.70%
- payment-postg... 0.67%
- send-email-mail... 0.67%
- product-recom... 0.54%
- ad-server-mon... 0.50%
- shipping-queu... 0.46%
- ad-server 0.32%



2. Kafka Metric Integration

Kafka broker 메트릭 수집

공통 수집 사항

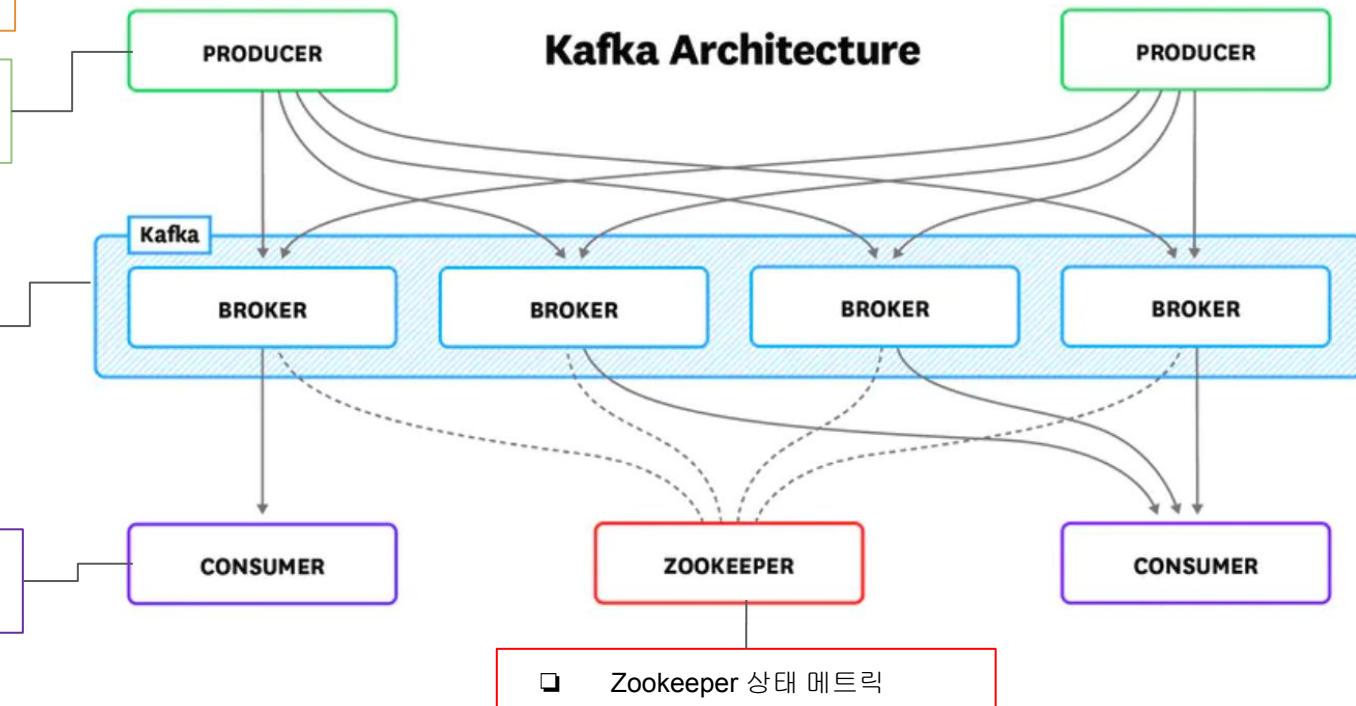
- ❑ Host 메트릭
- ❑ Log 수집
- ❑ Network Flow 수집

- ❑ Producer 상태 메트릭
- ❑ Producer 분산 트랜잭션 수집

- ❑ Kafka Broker 상태 메트릭

- ❑ Consumer 상태 메트릭
- ❑ Consumer 분산 트랜잭션 수집

Kafka Architecture



Kafka Integration (Broker)

1. Kafka broker에서 JMX remote port 활성화 (이미 활성화 되어 있다면 다음 단계로 진행)

```
export KAFKA_JMX_OPTS="-Djava.rmi.server.hostname=127.0.0.1 -Dcom.sun.management.jmxremote=true  
-Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false  
-Dcom.sun.management.jmxremote.rmi.port=9999 -Dcom.sun.management.jmxremote.port=9999"
```

2. /etc/datadog-agent/conf.d/kafka.d/conf.yaml 파일에 kafka jmx 주소와 포트 지정

- 샘플 파일 kafka.d/conf.yaml.example를 kafka.d/conf.yaml로 변경하면 기본 준비 완료
- default로 jmx port 9999로 되어 있는 부분 확인
- 설정 완료 후 Datadog Agent 재시작

```
init_config:  
  is_jmx: true  
  collect_default_metrics: true  
instances:  
  - host: localhost  
    port: 9999
```

Kafka broker metrics

All metrics reporting across your infrastructure in the past 1 hour -

Metric	<input type="text" value="kafka."/>	<input checked="" type="checkbox"/> Hide Controls	Showing 1-45 of 45 metrics
▼ Configuration			
<input type="checkbox"/> Has percentiles	0	METRIC NAME	
<input type="checkbox"/> Configured tags	0		kafka.consumer_lag
▼ Metric Type			
<input type="checkbox"/> Distributions	0		kafka.broker_offset
<input type="checkbox"/> Counts, Rates, Gauges	45		kafka.consumer_offset
			kafka.messages_in.rate
			kafka.net.bytes_in.rate
			kafka.net.bytes_out.rate
			kafka.log.flush_rate.rate
			kafka.replication.max.lag
			kafka.session.fetch.count
			kafka.session.fetch.eviction
			kafka.topic.messages_in.rate
			kafka.net.bytes_rejected.rate
			kafka.topic.net.bytes_in.rate
			check_run.kafka.can_connect.ok
			kafka.replication.leader_count
			kafka.request.produce.time.avg
			kafka.topic.net.bytes_out.rate
			kafka.request.fetch.failed.rate
			kafka.request.metadata.time.avg
			kafka.request.channel.queue.size
			kafka.replication.partition_count
			kafka.request.produce.failed.rate

kafka.request.metadata.time.avg
kafka.request.channel.queue.size
kafka.replication.partition_count
kafka.request.produce.failed.rate
kafka.session.zookeeper.sync.rate
kafka.replication.isr_expands.rate
kafka.replication.isr_shrinks.rate
kafka.session.zookeeper.expire.rate
kafka.net.processor.avg.idle.pct.rate
kafka.request.fetch_consumer.time.avg
kafka.request.fetch_follower.time.avg
kafka.session.zookeeper.readonly.rate
kafka.request.update_metadata.time.avg
kafka.replication.leader_elections.rate
kafka.request.handler.avg.idle.pct.rate
kafka.request.produce.time.99percentile
kafka.session.zookeeper.disconnect.rate
kafka.request.metadata.time.99percentile
kafka.replication.active_controller_count
kafka.replication.offline_partitions_count
kafka.request.fetch_request_purgatory.size
kafka.replication.under_replicated_partitions
kafka.request.producer_request_purgatory.size
kafka.request.fetch_consumer.time.99percentile
kafka.request.fetch_follower.time.99percentile
kafka.replication.unclean_leader_elections.rate
kafka.request.update_metadata.time.99percentile

METRIC NAME
jvm.heap_memory
jvm.gc.cms.count
jvm.gc.eden_size
jvm.thread_count
jvm.gc.parnew.time
jvm.loaded_classes
jvm.cpu_load.system
jvm.gc.old_gen_size
jvm.heap_memory_max
jvm.non_heap_memory
jvm.cpu_load.process
jvm.gc.survivor_size
jvm.heap_memory_init
jvm.gc.metaspacesize
jvm.non_heap_memory_max
jvm.non_heap_memory_init
jvm.heap_memory_committed
jvm.buffer_pool.direct.used
jvm.buffer_pool.mapped.used
jvm.buffer_pool.direct.count
jvm.buffer_pool.mapped.count
jvm.os.open_file_descriptors
jvm.non_heap_memory_committed
jvm.buffer_pool.direct.capacity
jvm.buffer_pool.mapped.capacity

Kafka Producer/Consumer 메트릭 수집

공통 수집 사항

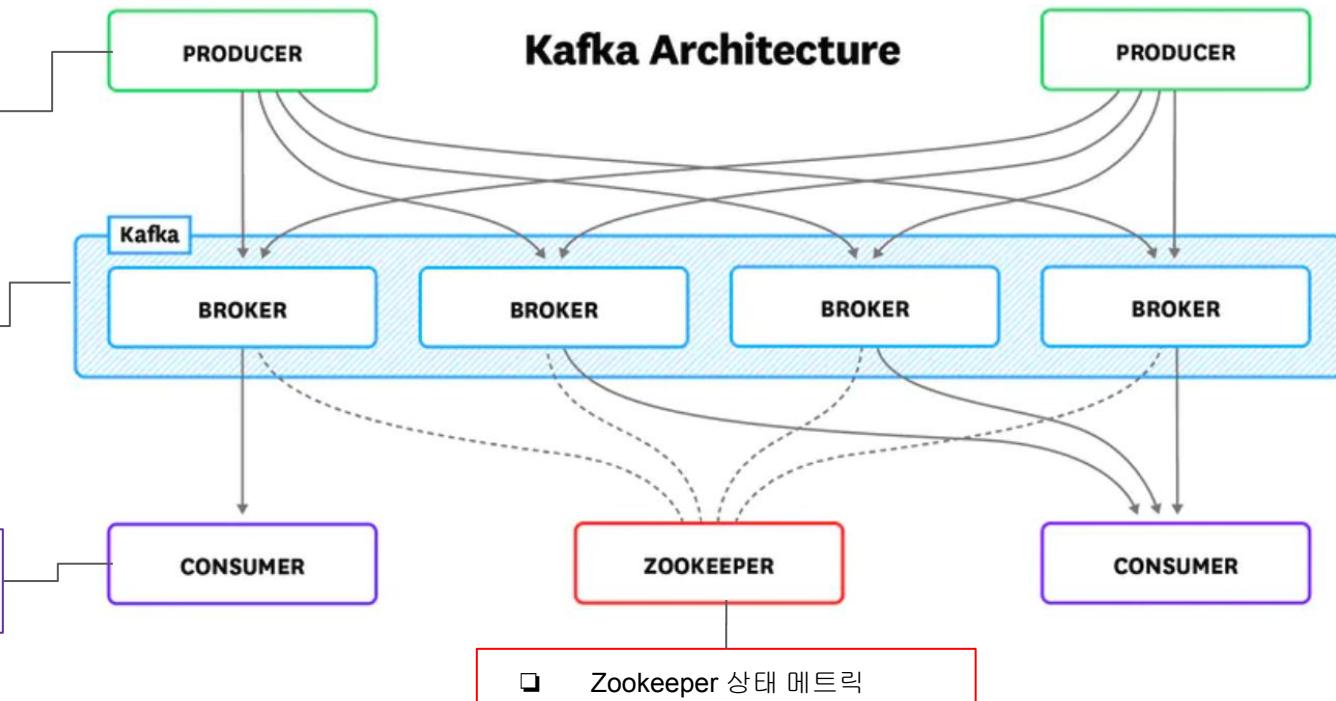
- ❑ Host 메트릭
- ❑ Log 수집
- ❑ Network Flow 수집

- ❑ Producer 상태 메트릭
- ❑ Producer 분산 트랜잭션 수집

- ❑ Kafka Broker 상태 메트릭

- ❑ Consumer 상태 메트릭
- ❑ Consumer 분산 트랜잭션 수집

Kafka Architecture



Kafka Integration (Producer / Consumer)

1. Producer/Consumer Application에서 JMX remote port 활성화

```
export KAFKA_JMX_OPTS="-Djava.rmi.server.hostname=127.0.0.1 -Dcom.sun.management.jmxremote=true  
-Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false  
-Dcom.sun.management.jmxremote.rmi.port=9999 -Dcom.sun.management.jmxremote.port=9999"
```

```
java $KAFKA_JMX_OPTS -jar myapp.jar
```

2. /etc/datadog-agent/conf.d/kafka.d/conf.yaml 파일에 kafka jmx 주소와 포트

- 샘플 파일 kafka.d/conf.yaml.example를 kafka.d/conf.yaml로 변경 후 사용
- default로 jmx port 9999로 되어 있는 부분 확인
- 설정 완료 후 Datadog Agent 재시작

```
init_config:  
  is_jmx: true  
  collect_default_metrics: true  
instances:  
  - host: localhost  
    port: 9999  
    tags:
```

Kafka Producer/Consumer metrics

All metrics reporting across your infrastructure in the past 1 hour ▾

Metric Hide Controls Showing 1-32 of 32 metrics

0	METRIC NAME
0	kafka.producer.io_wait
0	kafka.consumer.bytes_in
0	kafka.producer.bytes_out
32	kafka.consumer.messages_in
	kafka.producer.metadata_age
	kafka.producer.request_rate
	kafka.producer.response_rate
	kafka.consumer.fetch_size_avg
	kafka.consumer.fetch_size_max
	kafka.producer.batch_size_avg
	kafka.producer.batch_size_max
	check_run.kafka.can_connect.ok
	kafka.producer.record_size_avg
	kafka.producer.record_size_max
	kafka.producer.waiting_threads
	kafka.producer.compression_rate
	kafka.producer.record_send_rate
	kafka.producer.record_error_rate
	kafka.producer.record_retry_rate
	kafka.producer.records_send_rate
	kafka.producer.throttle_time_avg
	kafka.producer.throttle_time_max
	kafka.producer.buffer_bytes_total
	kafka.producer.requests_in_flight
	kafka.producer.records_per_request
	kafka.producer.request_latency_avg
	kafka.producer.request_latency_max
	kafka.producer.compression_rate_avg
	kafka.producer.record_queue_time_avg
	kafka.producer.record_queue_time_max
	kafka.producer.available_buffer_bytes
	kafka.consumer.records_per_request_avg

jvm.heap_memory
jvm.gc.cms.count
jvm.gc.eden_size
jvm.thread_count
jvm.gc.parnew.time
jvm.loaded_classes
jvm.cpu_load.system
jvm.gc.old_gen_size
jvm.heap_memory_max
jvm.non_heap_memory
jvm.cpu_load.process
jvm.gc.survivor_size
jvm.heap_memory_init
jvm.gc.metaspacesize
jvm.non_heap_memory_max
jvm.non_heap_memory_init
jvm.heap_memory_committed
jvm.buffer_pool.direct.used
jvm.buffer_pool.mapped.used
jvm.buffer_pool.direct.count
jvm.buffer_pool.mapped.count
jvm.os.open_file_descriptors
jvm.non_heap_memory_committed
jvm.buffer_pool.direct.capacity
jvm.buffer_pool.mapped.capacity

Zookeeper Integration

공통 수집 사항

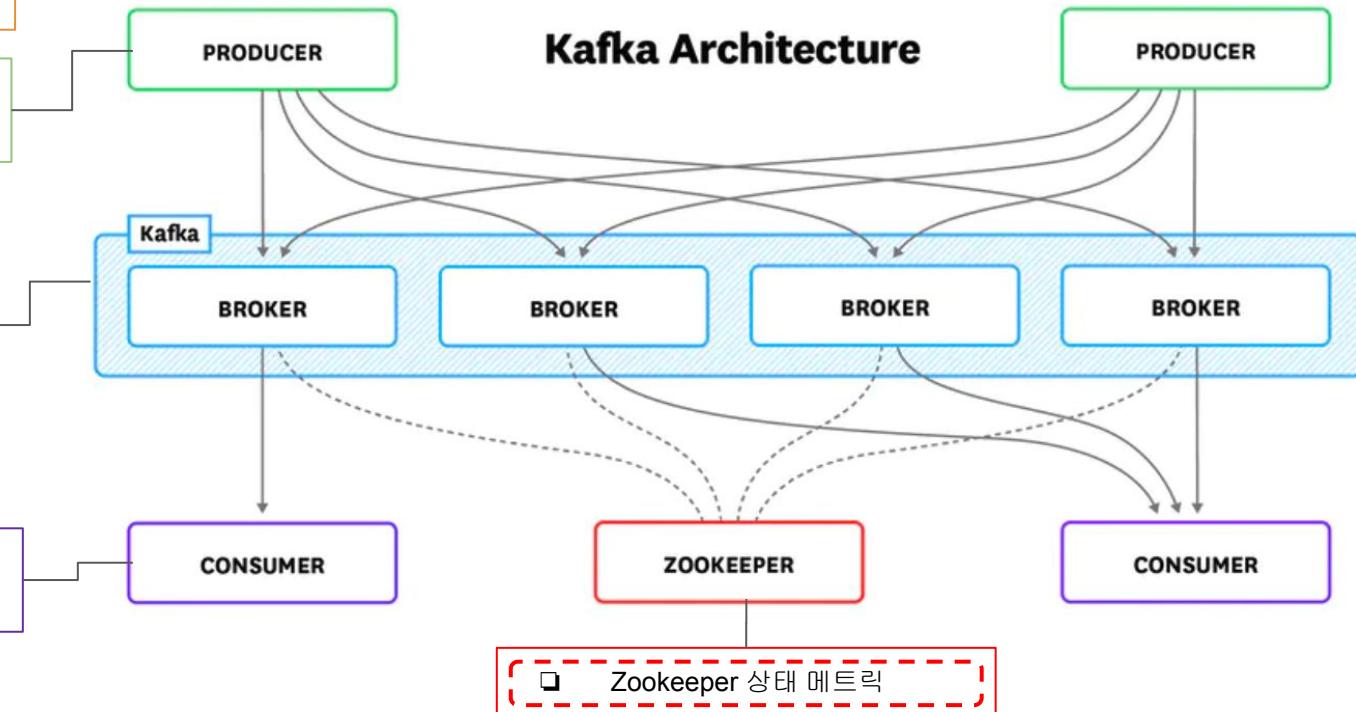
- ❑ Host 메트릭
- ❑ Log 수집
- ❑ Network Flow 수집

- ❑ Producer 상태 메트릭
- ❑ Producer 분산 트랜잭션 수집

- ❑ Kafka Broker 상태 메트릭

- ❑ Consumer 상태 메트릭
- ❑ Consumer 분산 트랜잭션 수집

Kafka Architecture



Zookeeper Integration

1. conf/zookeeper.properties 설정 파일에 모니터링 위한 command 추가

```
4lw.commands.whitelist=stat, ruok, conf, isro, mntr
```

2. /etc/datadog-agent/conf.d/zk.d/conf.yaml 파일에 zookeeper 주소와 포트 지정 후 Agent 재시작

- 샘플 파일 conf.d/zk.d/conf.yaml.example를 conf.d/zk.d/conf.yaml로 변경 후 사용
- 설정 완료 후 Datadog Agent 재시작

```
init_config:  
instances:  
  - host: localhost  
    port: 2181
```

Kafka Producer/Consumer metrics

Metric zookeeper

Configuration

- Has percentiles
- Configured tags

Metric Type

- Distributions
- Counts, Rates, Gauges

Hide Controls

Showing 1–38 of 38 metrics

METRIC NAME

0	zookeeper.nodes
0	zookeeper.followers
0	zookeeper.instances
38	zookeeper.bytes_sent
	zookeeper.zxid.count
	zookeeper.zxid.epoch
	zookeeper.avg_latency
	zookeeper.connections
	zookeeper.latency.avg
	zookeeper.latency.max
	zookeeper.latency.min
	zookeeper.max_latency
	zookeeper.min_latency
	zookeeper.watch_count
	zookeeper.znode_count
	zookeeper.packets.sent
	zookeeper.packets_sent
	zookeeper.pending_syncs
	zookeeper.bytes_received

zookeeper.ephemerals_count

zookeeper.instances.leader

zookeeper.packets.received

zookeeper.packets_received

zookeeper.synced_followers

check_run.zookeeper.ruok.ok

zookeeper.max_proposal_size

zookeeper.min_proposal_size

zookeeper.instances.follower

zookeeper.last_proposal_size

zookeeper.outstanding_requests

zookeeper.approximate_data_size

zookeeper.num_alive_connections

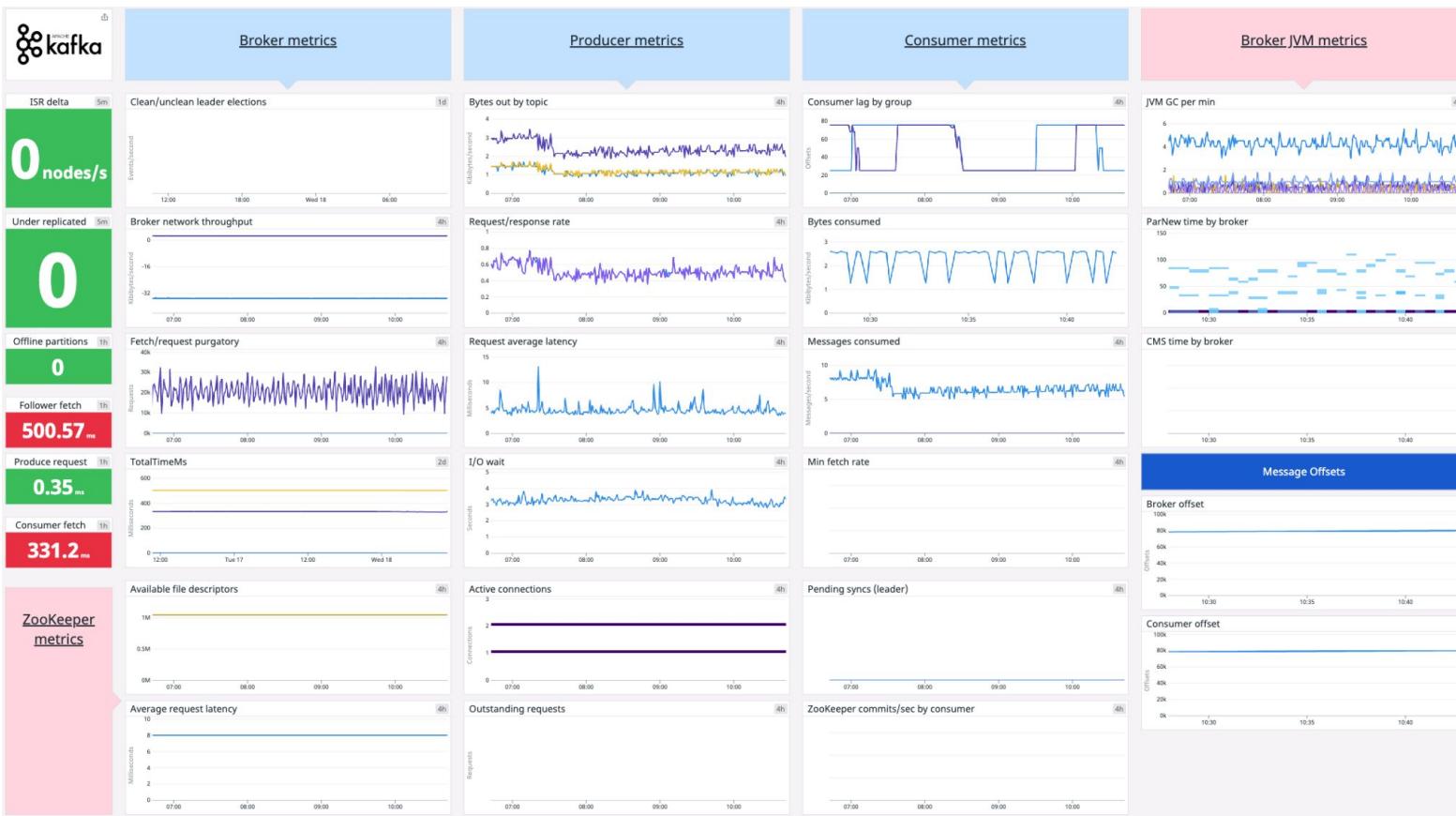
kafka.session.zookeeper.sync.rate

kafka.session.zookeeper.expire.rate

zookeeper.max_file_descriptor_count

zookeeper.open_file_descriptor_count

Kafka 기본 대시보드



Kafka Custom 대시보드

☆ Kafka Cluster Monitoring ▾ + Add Widgets

4h Past 4 Hours

Search... Add Template Variables ?

Overview
2 widgets

Kafka Host metric
6 widgets

Kafka Integration metric
12 widgets

Kafka GC metrics
3 widgets

Zookeeper Metric
15 widgets

Kafka를 사용하는 Application
6 widgets

Add widgets

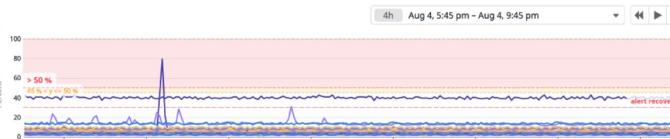
This image shows a screenshot of a custom Kafka cluster monitoring dashboard. The dashboard is titled "Kafka Cluster Monitoring" and includes a search bar and a time range selector for "Past 4 Hours". There are six main sections listed vertically, each represented by a colored bar indicating the number of widgets:

- Overview**: 2 widgets
- Kafka Host metric**: 6 widgets
- Kafka Integration metric**: 12 widgets
- Kafka GC metrics**: 3 widgets
- Zookeeper Metric**: 15 widgets
- Kafka를 사용하는 Application**: 6 widgets

At the bottom left, there is a "Add widgets" button with a plus sign icon.

메트릭 기반 알람 활용

1) 알람 조건 설정



4h Aug 4, 5:45 pm – Aug 4, 9:45 pm

Percent

100
80
60
40
20
0

25.50 %
25.50 %
25.50 %

18:00 18:30 19:00 19:30 20:00 20:30 21:00 21:30

alert recovery

1 Choose the detection method

Threshold Alert Change Alert Anomaly Detection Outliers Alert Forecast Alert

An alert is triggered whenever a metric crosses a threshold.

2 Define the metric

Metric aws.rds.cpuutilization from (everywhere) avg by dbinstanceidentifier + Advanced...

Trigger a separate alert for each dbinstanceidentifier reporting your metric

3 Set alert conditions

Trigger when the metric is above the threshold at least once during the last 5 minutes for any dbinstanceidentifier

Alert threshold: > 50 (50 %)

Warning threshold: > 45 (45 %)

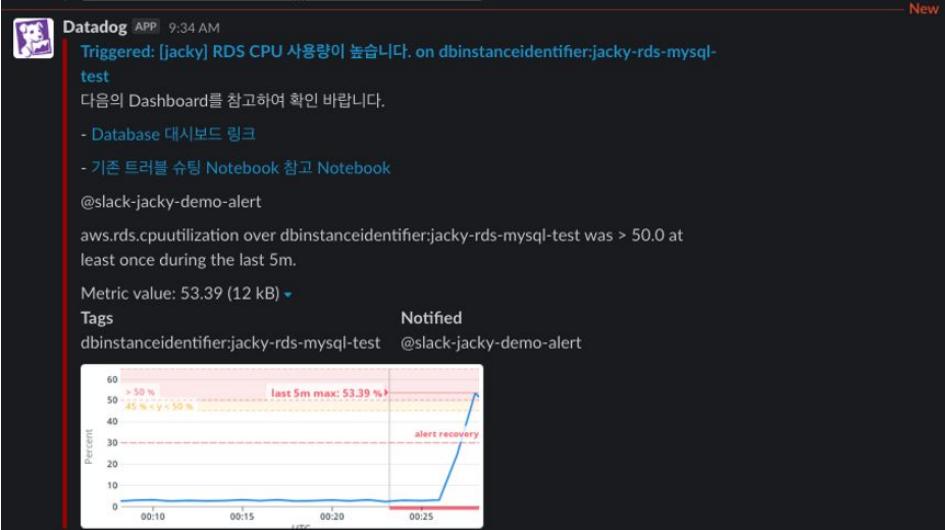
4 Say what's happening

Include triggering tags in notification title

Edit Preview Markdown Help Use Message Template Variables

[Jacky] RDS CPU 사용량이 높습니다.
다음의 Dashboard를 참고하여 확인 바랍니다.
[Database 대시보드 링크](https://app.datadoghq.com/dashboard/wmd-kgw-9ya/jacky-database-monitoring?from_ts=1628006252161&to_ts=1628092652161&live=true)
[기준 트리플 슈팅 Notebook 참고 Notebook](https://app.datadoghq.com/notebook/template/9/rfc-title)
@slack-jacky-demo-alert

2) 알람 메시지(Slack)를 통한 이슈 확인



Datadog APP 9:34 AM

Triggered: [Jacky] RDS CPU 사용량이 높습니다. on dbinstanceidentifier:jacky-rds-mysql-test

다음의 Dashboard를 참고하여 확인 바랍니다.

- Database 대시보드 링크

- 기준 트리플 슈팅 Notebook 참고 Notebook

@slack-jacky-demo-alert

aws.rds.cpuutilization over dbinstanceidentifier:jacky-rds-mysql-test was > 50.0 at least once during the last 5m.

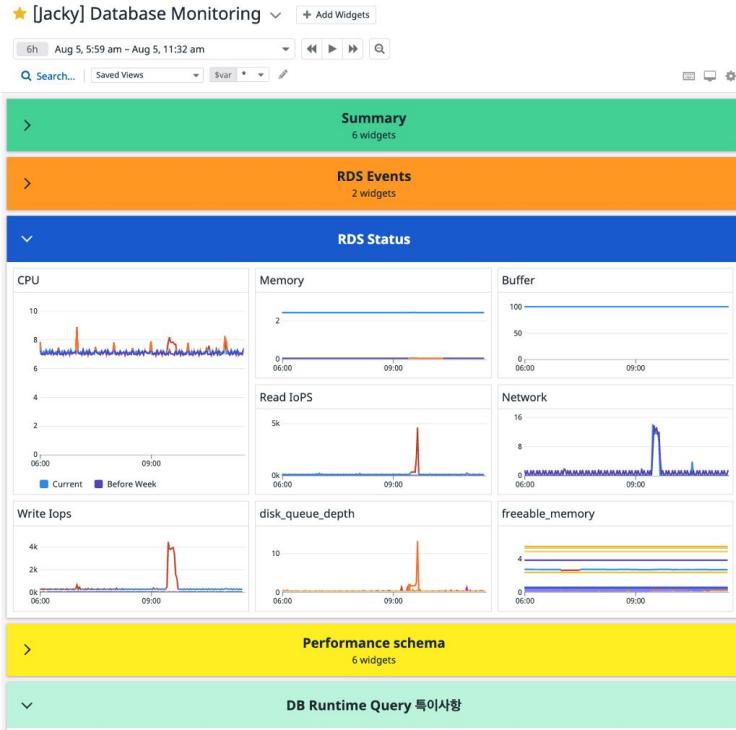
Metric value: 53.39 (12 kB)

Tags dbinstanceidentifier:jacky-rds-mysql-test Notified @slack-jacky-demo-alert



메트릭 기반 알람 활용

3) 알람 메시지를 참고하여 연계 대시보드로 넘어와 분석



4) 장애 분석 내용을 Notebook에 정리하여 팀원과 협업

[Jacky] RDS CPU 상승건에 대한 분석



Created by Datadog
Clone to save changes

PROPERTY	DETAILS
Author(s)	Jacky Jung (DevOps Engineer)
Status:	미분정
Date Created:	09:30 UTC+9 08/05/2021

Overview

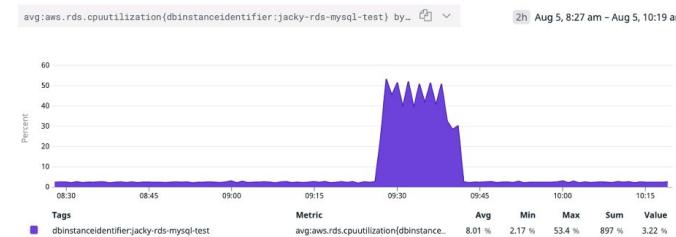
RDS CPU 상승으로 인해 사용자 로그인에 자연 발생 (Severity: 3등급)

Problem

RDS 사용량 증가로 인해 로그인 뿐만 아니라 주문 처리 등 주요 API에 자연 발생함

검토 사항

1. RDS Scale Up에 대한 고민 필요. (무증단 Scale Up 방안 검토 필요)
2. CPU 사용량을 증가하게 만든 최근 튜닝.
3. Query offloading 과 sharding에 대해 고민 필요





3. APM을 이용한 Kafka 트랜잭션 추적

Producer 및 Consumer 트랜잭션 추적

공통 수집 사항

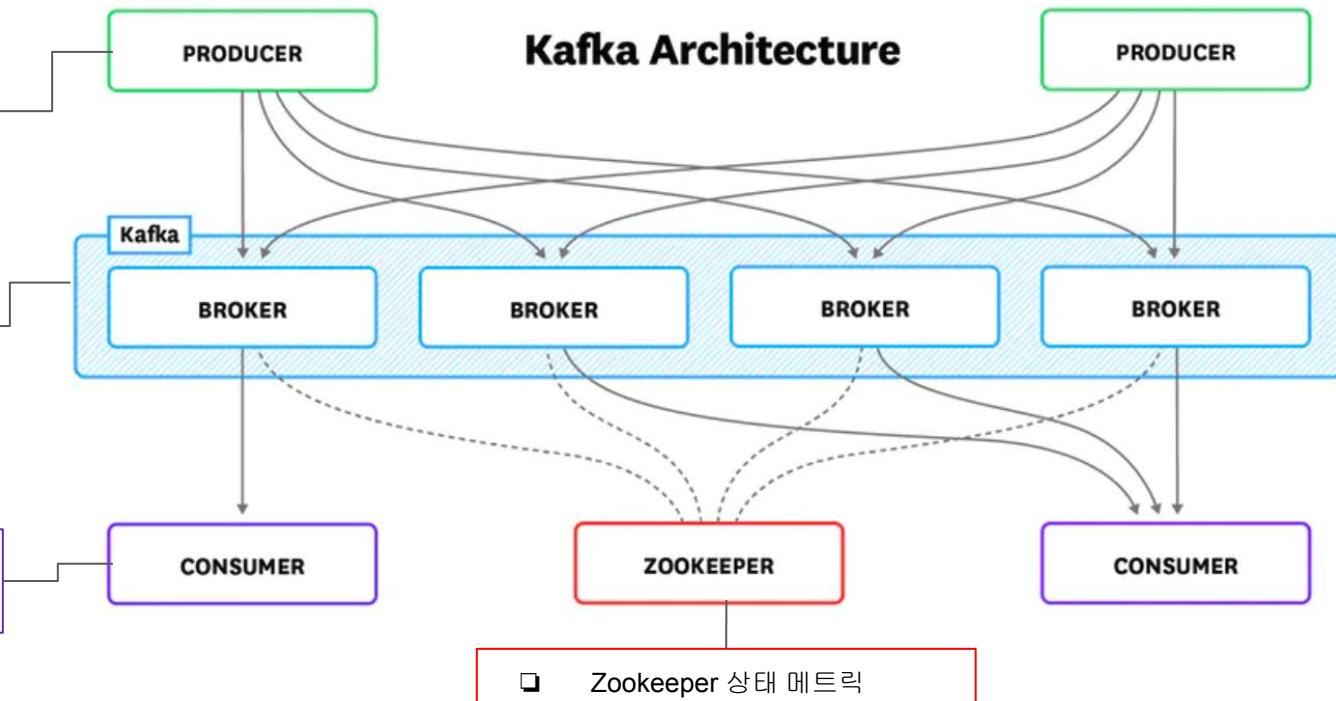
- ❑ Host 메트릭
- ❑ Log 수집
- ❑ Network Flow 수집

- ❑ Producer 상태 메트릭
- ❑ Producer 분산 트랜잭션 수집

- ❑ Kafka Broker 상태 메트릭

- ❑ Consumer 상태 메트릭
- ❑ Consumer 분산 트랜잭션 수집

Kafka Architecture



APM 연동 방법 (Producer/Consumer)

1. Datadog Trace 라이브러리 다운로드 (Java application 기준 설명)

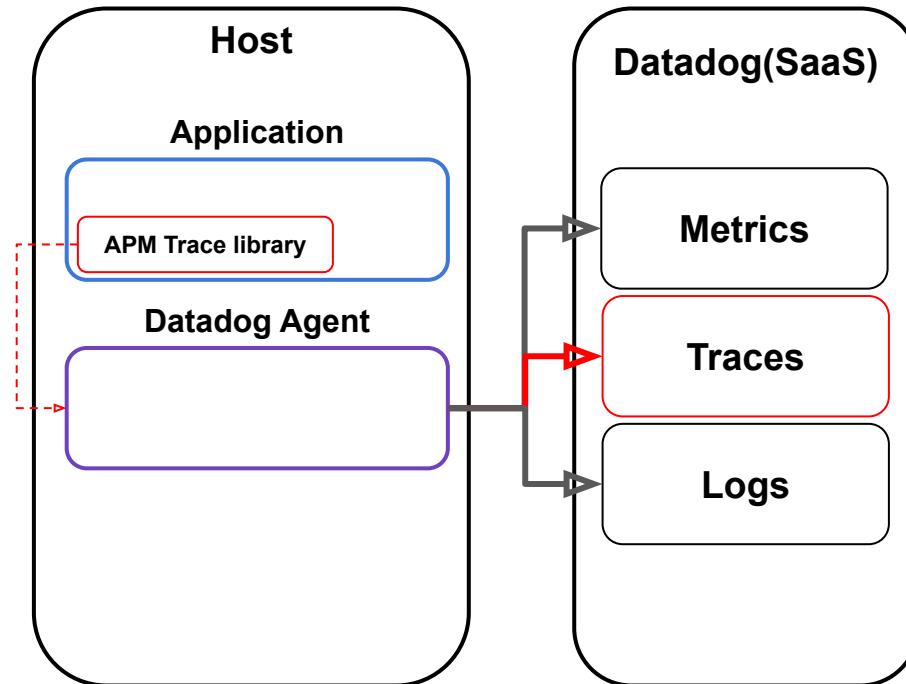
```
wget -O dd-java-agent.jar 'https://dtdg.co/latest-java-tracer'
```

2. Java application 실행 옵션에 -javaagent 옵션을 사용하여 trace 라이브러리 삽입 후 app 시작

```
export KAFKA_JMX_OPTS="-Djava.rmi.server.hostname=127.0.0.1 -Dcom.sun.management.jmxremote=true  
-Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false  
-Dcom.sun.management.jmxremote.rmi.port=9999 -Dcom.sun.management.jmxremote.port=9999"
```

```
java $KAFKA_JMX_OPTS -javaagent:dd-java-agent.jar -Ddd.version=1.13 -Ddd.logs.injection=true -Ddd.trace.sample.rate=1  
-Ddd.service=domain-crawler -Ddd.env=prod -Dserver.port=7070 -jar myapp.jar
```

APM 연동 아키텍쳐



APM Compatibility

1. 다음의 언어에 대해 APM 기능 지원



2. Java의 경우 Java7 이상부터 APM 지원

3. 언어별 탐지되는 라이브러리 및 버전 확인 필요

- 언어별 Compatibility 페이지 참고
- Java의 경우 Kafka-Clients 0.11+, Kafka-Streams 0.11+ 에 대해 가시성 지원

FRAMEWORK	VERSIONS	SUPPORT TYPE	INSTRUMENTATION NAMES (USED FOR CONFIGURATION)
Apache HTTP Client	4.0+	Fully Supported	httpclient, apache-httpclient, apache-http-client
Apache HTTP Async Client	4.0+	Fully Supported	httpsyncclient, apache-httpsyncclient
AWS Java SDK	1.11+, 2.2+	Fully Supported	aws-sdk
Commons HTTP Client	2.0+	Fully Supported	commons-http-client
Google HTTP Client	1.19.0+	Fully Supported	google-http-client
Grizzly HTTP Client	1.9+	Beta	grizzly-client
gRPC	1.5+	Fully Supported	grpc, grpc-client, grpc-server
HttpURLConnection	all	Fully Supported	httpurlconnection, urlconnection
Kafka-Clients	0.11+	Fully Supported	kafka
Kafka-Streams	0.11+	Fully Supported	kafka, kafka-streams

참고 링크

APM 연동 후 Kafka 분산 트랙잭션 트래킹

APM Services Traces Profiles Open Full Page

Search Save As

Flame Graph Span List (255) Hide Legend

Service % Exec Time domain-crawler 80.3% kafka 19.7% kafka-domain-consumer < 0.1%

Requests 14 total (0.2 req/s)

08:36 8 6 4 2 0 Requests

Facets Saved Views Hide Controls Watchdog Insights

Showing 54 of 60 + Add DATE

CORE Duration Status Env Service

Duration: Min 0ns Max 13.9ms

Status: Ok 14 Error 0

Env: Kafka-domain-consumer 14

Service: kafka-domain-consumer

domain-crawler | GET /domain/lookup/{name}

Aug 18 08:36:50.629 | 1.33 ms GET http://localhost:7070/domain/lookup/yahoo.com 200 OK

Flame Graph Span List (255)

kafka.produce Produce Topic active.web-domains 4.92 ms

kafka.produce Produce Topic active.web-domains 102 ms

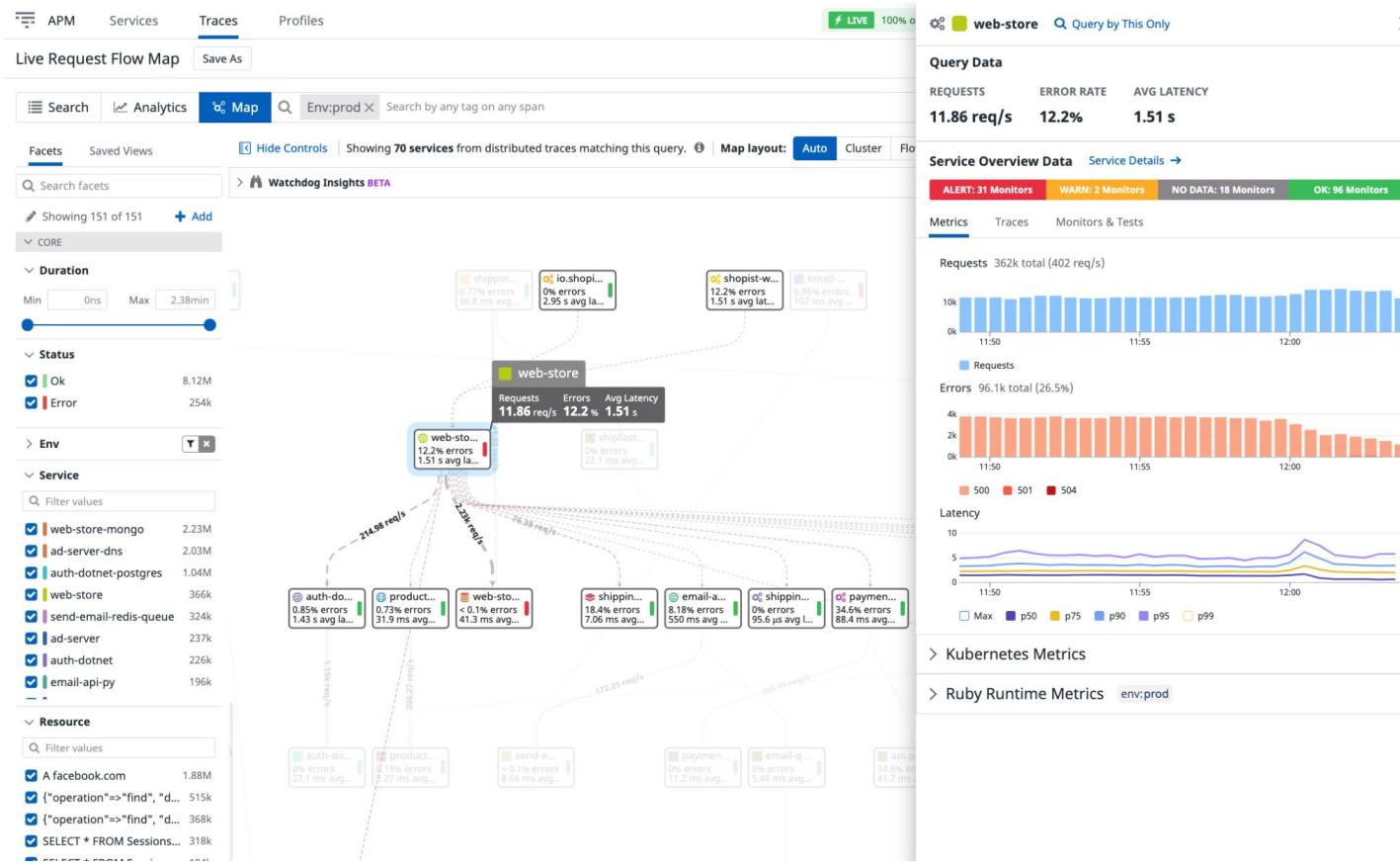
kafka-domain-consumer

kafka.consume Consume Topic active.web-domains ① 0-7d8a4b6c69641c0a ② 700 ns (< 0.1% of total Time)

Tags Infrastructure Metrics Logs 0 Processes Network Code Hotspots BETA

```
{ component: "java-kafka", env: "prod", language: "jvm", offset: "134232", partition: "0", record: { queue_time_ms: 9 }, runtime_id: "17aad866-61ce-45e9-afec-19690460e7a1", span: { } }
```

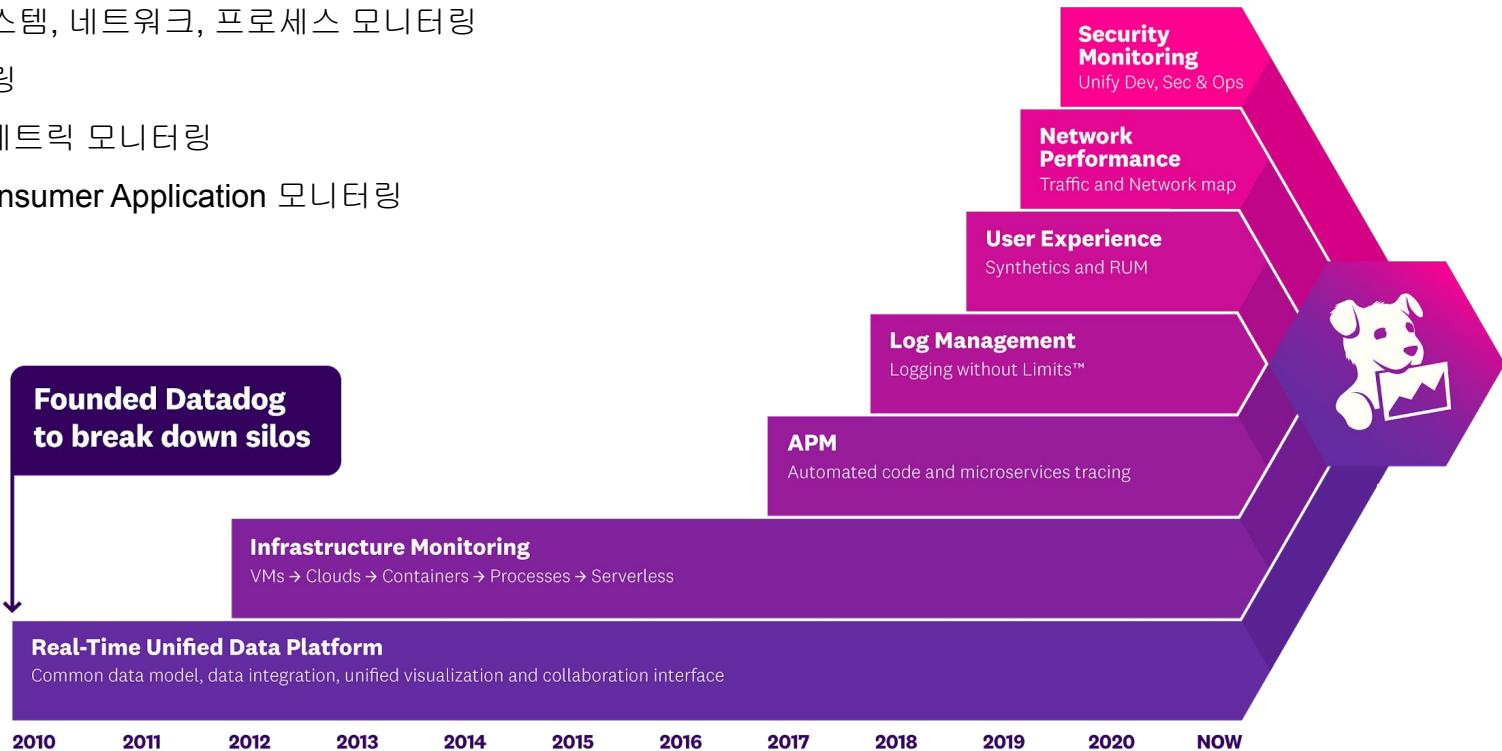
APM을 이용한 Microservice 분산 트레이싱



요약

다음의 모니터링 영역에 대해 살펴 보았습니다

1. 호스트의 시스템, 네트워크, 프로세스 모니터링
2. 로그 모니터링
3. Kafka 주요 메트릭 모니터링
4. Producer/Consumer Application 모니터링



Datadog 사용에 대한 기대 효과

Datadog 모니터링 플랫폼에 주요 모니터링 요소를 통합함으로써

- 1) 모니터링 연동을 위한 개발 및 관리 시간 최소화
- 2) 빠른 연계 분석을 통해 서비스 장애 분석 시간 최소화
- 3) 개발팀과 운영팀 협업 효율 증가



DATADOG

Q&A



감사합니다