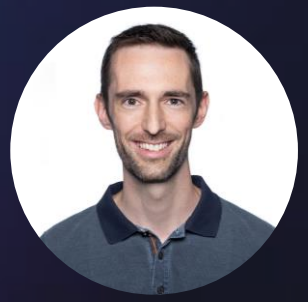
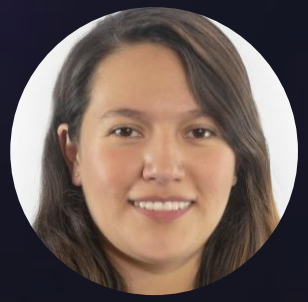
Ingest All the Metrics

Eric Eiswerth



Robert Schausberger



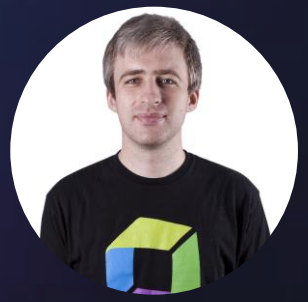
Lorena LaRosa

Mike Kopp

dynatrace
Perform

Ingest All the Metrics

Michal Nalezinski



Andrew Patterson

Herwig Moser

Stefan Moschinski

dynatrace
Perform

Michal Dec

Prerequisites

- Windows or Mac OS with outbound public internet access over ports, 22, 443, 9999
- Remote Machine Access via either:
  - Browser SSH shell, port 22
  - SSH Putty client port 22
- Updated Chrome, Safari, or Firefox browser with outbound network access over ports: 443, 22
- Notepad, or Notepad++

- Dynatrace University utilizes Zoom for delivering virtual online training. The technical requirements can be found in the Zoom Help Center.

Lab Environments

- Dynatrace SaaS Instance
- VM for OneAgent Deployment with
  - Telegraf
  - Statsd
  - Dynatrace_ingest env variable
  - Local metric ingest
- VM for ActiveGate, AWS Integration and remote telegraf (if time permits)
- VM for fluentd log ingestion
- Dynatrace University SSH access for all 3

Amazon CloudWatch

telegraf

STATSD

dynatrace

fluentd

Time Budgets = 3.5 Hours

- Generic metric ingest, statsd, and explaining the basics – 90 minutes
- Telegraf – 15 minutes(with optional no-oneagent config to add 15 minutes)
- Break – 15 minutes
- Log Ingest – at least 1 hour
- Break – 15 minutes
- Cloud – 30 minutes

HOT Session Ingest All the Metrics

dynatrace
Perform

Metric Ingest

Why

- Make it **easy** and **attractive** for customers to **push metrics** into Dynatrace
- **Seamless integration** of custom metrics into the Dynatrace software intelligence platform
  - Connect entities with business KPIs / metrics
  - Feed **Davis AI** and **Smartscape** with your own business-critical metrics
  - Grant Davis more visibility into your organizations KPIs
- Offer further **integrations** from a multitude of metric data sources

Metrics can easily be sent via different ingest channels

Data Source

temperature,room=5.30 22

http payload + API token

Public Rest ENDPOINT
https://<env-URL>/api/v2/metrics/ingest

OR

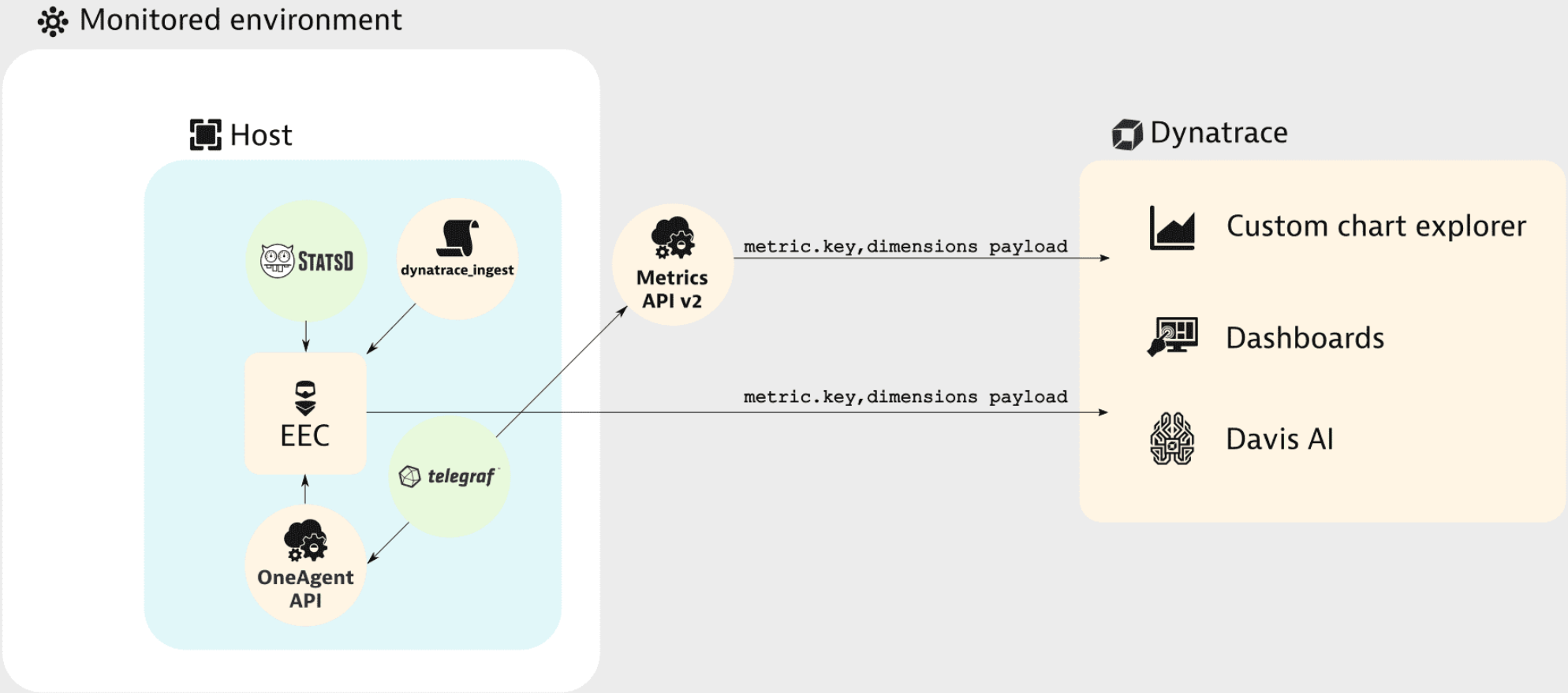
http payload

dynatrace_ingest (Pipe)

StatsD

From Host via OneAgent

Entity of OneAgent monitored Host is automatically enriched as dimension (dt.entity.host=<host-ID>)


Monitored environment
Host
STATSD
dynatrace_ingest
EEC
telegraf
OneAgent API
Metrics API v2
metric.key,dimensions payload
metric.key,dimensions payload
Dynatrace
Custom chart explorer
Dashboards
Davis AI

New metric ingest line protocol

What	Where	Measurement
mymetric,	team=teamA,businessapp=hr	10 1584619278002
Metric-ID	Metric dimension(s)	Data type & Timestamp

Supporting payload up to 1000 lines per Ingest API call

Examples:

```
humidity,room=5.30 78.9
humidity,office=Winterhafen,room="first floor" 100
temperature,room=5.30 gauge,min=17.1,max=17.3,sum=34.4,count=2
utilization,room=5.30 gauge,5 1588566479205
interface.packets_received.count,if=eth0 count,delta=10
```

Metric data type GAUGE

- Gauges are representing a single numeric value that can go up/down
  - CPU idle, CPU temperature, …
- Aggregation types: average, count, max, min, sum
- Examples:

```
temperature,city=linz 35 1598262190984

temperature,city=linz 32

temperature,city=gdansk gauge,22.1

temperature,city=linz gauge,min=17.1,max=17.3,sum=34.4,count=2
```

Metric data type COUNT

- COUNTERS are cumulative metrics
- Examples: number of requests, tasks completed, errors
- Only possible aggregation is COUNT
- Count metric keys are automatically suffixed by *.count if not already ending with .count*
- Client-side delta vs. Cluster-side delta mode
  - Delta (delta calculation on producer side)

    test.request.count count,**delta**=11
    test.request.count count,**delta**=33

    count=44 (for timeslot X)
  - Absolute (deltas calculated on server)

    test.request.count count,11
    test.request.count count,33

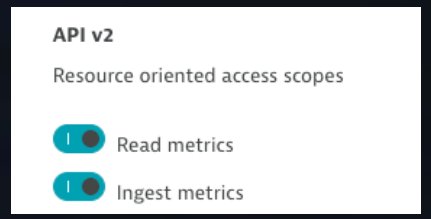
    count=22

Metrics & topology

- Topological information linked to your metric data
- Metric **without** topology
  - ‚Flat' metric without relation to the topology
  - Events and alerts are raised on environment level only
  - No Davis root cause analysis
  - `humidity,room=5.30 78.9`
- Metric **with** topology
  - Host entity is automatically added if ingest via OneAgent
  - Metric is mapped to topology entities such as a host, process, service, …
  - Davis AI can be triggered, root-cause analysis as usual
  - `cpu.temperature,dt.entity.host=HOST-123,cpu=cpu1 55`
  - `service.response,dt.entity.host=HOST-123,dt.entity.service=SERVICE-456 100`

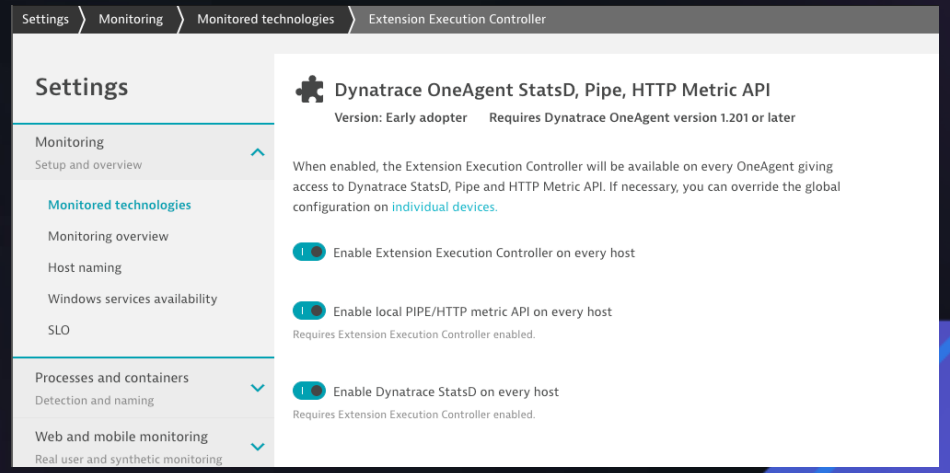
Ingest channels

- Public API
  - New Endpoint on Public Rest API: `/api/v2/metrics/ingest`
  - Generate API Token



OR

- OneAgent Channel
  - Enable the OneAgent local metric API
    `http://localhost:<port>/metrics/ingest`
  - Enable StatsD



New Public REST Endpoint for Ingest

New Endpoint on Public Rest API: `/api/v2/metrics/ingest`

POST /metrics/ingest — Early adopter

Ingest metric data points via line protocol.

Required Api-Token scope:
`metrics.ingest` (Ingest metrics using API V2)

E.g. via CURL Command:

curl -H "Authorization: Api-Token <TOKEN>" -X POST -H "Content-Type: text/plain; charset=utf-8" --data-ascii "cpu.temp,cpu=cpu1,cpu_type=INTEL 55" https://<env-URL>/api/v2/metrics/ingest

Body | Cookies | Headers (13) | Test Results

Pretty | Raw | Preview | Visualize | JSON

```
{
    "linesOk": 6,
    "linesInvalid": 0,
    "error": null
}
```

Responses

Code	Description	Links
202	The provided metric data points are valid and get accepted for background processing.	No links
400	A part or all of the provided data points are invalid. Valid data points get accepted for background processing.	No links

OneAgent Ingest Channel

Settings > Monitoring > Monitored technologies

Java/.NET/Node.js/Golang/PHP runtime metrics + WebServer metrics in Infrastructure Mode Requires Dynatrace OneAgent version 1.189 or later and includes JMX/PMI Extensions		Infrastructure insights
Dynatrace OneAgent StatsD, Pipe, HTTP Metric API Requires Dynatrace OneAgent version 1.201 or later	EARLY ADOPTER	Infrastructure insights
AIX kernel extensions		Infrastructure insights

Settings > Monitoring > Monitored technologies > Extension Execution Controller

Settings

Monitoring
Setup and overview

- Monitored technologies
- Monitoring overview
- Host naming
- Windows services availability
- SLO

Processes and containers
Detection and naming

Web and mobile monitoring
Real user and synthetic monitoring

Dynatrace OneAgent StatsD, Pipe, HTTP Metric API

Version: Early adopter **Requires Dynatrace OneAgent version 1.201 or later**

When enabled, the Extension Execution Controller will be available on every OneAgent giving access to Dynatrace StatsD, Pipe and HTTP Metric API. If necessary, you can override the global configuration on individual devices.

- [x] Enable Extension Execution Controller on every host
- [x] Enable local PIPE/HTTP metric API on every host
  Requires Extension Execution Controller enabled.
- [x] Enable Dynatrace StatsD on every host
  Requires Extension Execution Controller enabled.

Metric dimensions

- Dimension key / values can be used to further segment & structure data
  - cpu.temp[,dt.entity.host=HOST-D4EC0D8F510A6F60],cpu=cpu1,cpu_type=INTEL 55

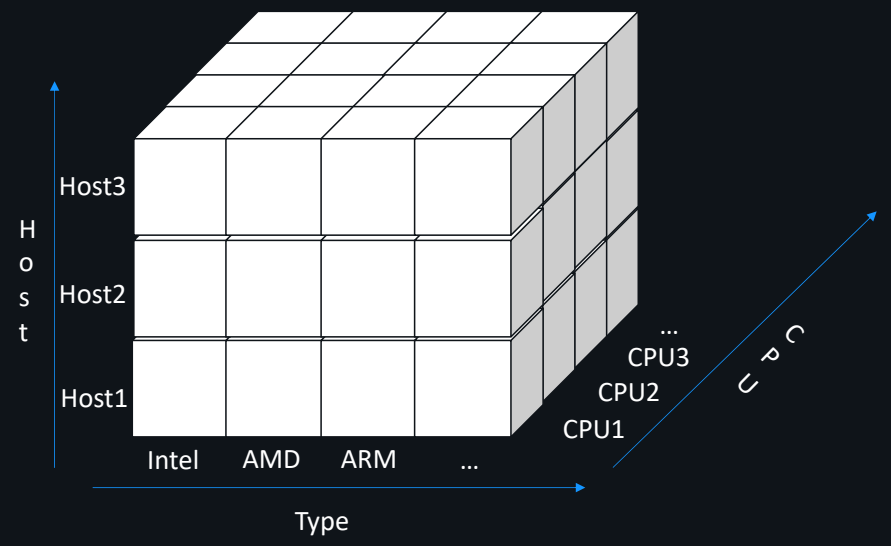
    dimension-key
    dimension-value

- Monitored entities can be ingested as dimensions
  - Reserved dimension keys (starting with *dt.entity.*)
  - Multiple monitored entities can be passed

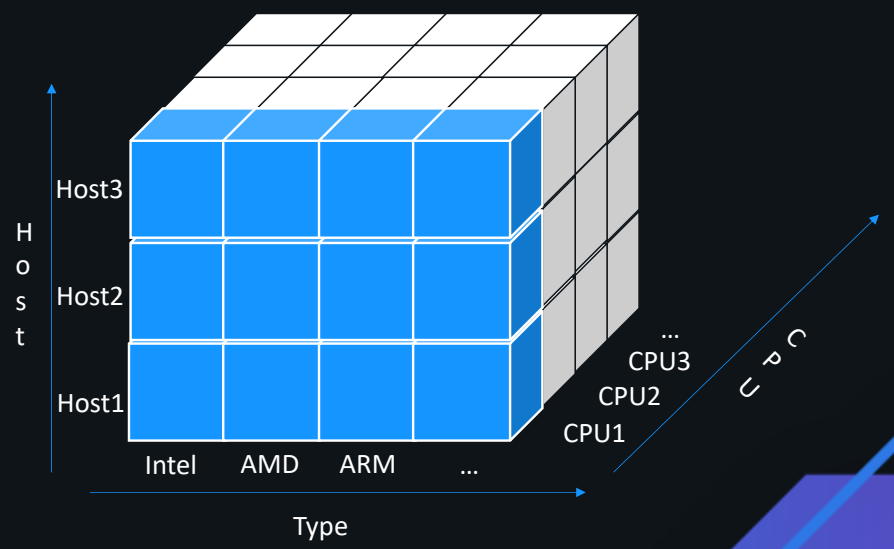
- Use dimensions for
  - Querying data via Metrics API
  - Charting with Explorer
  - Alerting with Davis AI

Metric dimensions

cpu.temp,dt.entity.host=HOST-D4EC0D8F510A6F60,cpu=cpu1,cpu_type=INTEL 55



Get temperature for CPU1



Metric Query with API v2:
metricSelector=cpu.temp:splitBy(cpu):filter(eq(cpu,"cpu1"))

Metric/dimension syntax

- **Metric keys** accepts [a-z,A-Z,0-9,_,-,.]
  - Must start with a character
  - Case sensitive
  - Max. length: 250 characters
- **Time** information
  - metric gauge,1 1595345973712
    → timestamp in millis
  - Optional / cluster time is used as default
  - 1 hour to the past (might change) /
    10 mins to the future

- **Dimension key**: only [a-z,0-9,_,-,.] characters
  - [a-z] is Latin lower case alphabet
  - Must start with character
  - Max. length: 100 characters
- **Dimension value**:
  - Allowed: All non-control characters
  - Not allowed: Any ASCII and Unicode control characters
  - Max. length: 250 characters

Ingest line examples (1/2)

- Metrics without relation to an entity

```
cpu.temperature,cpu=cpu1,cpu_type=INTEL 50
cpu.temperature,cpu=cpu2,cpu_type=INTEL 55
```

- Metrics with host as related entity

```
cpu.temperature,dt.entity.host=HOST-06F288EE2A930951,cpu=1 55
cpu.temperature,dt.entity.host=HOST-06F288EE2A930951,cpu=2 45
```

- Send multiple, consolidated measurements

```
cpu.temperature,cpu=1 gauge,min=17.1,max=17.3,sum=34.4,count=2
```

Ingest line examples (2/2)

- Monotonic count metric (delta already calculated by the client)

```
new_user_count,region=EAST count,delta=50

new_user_count,region=WEST count,delta=150
```

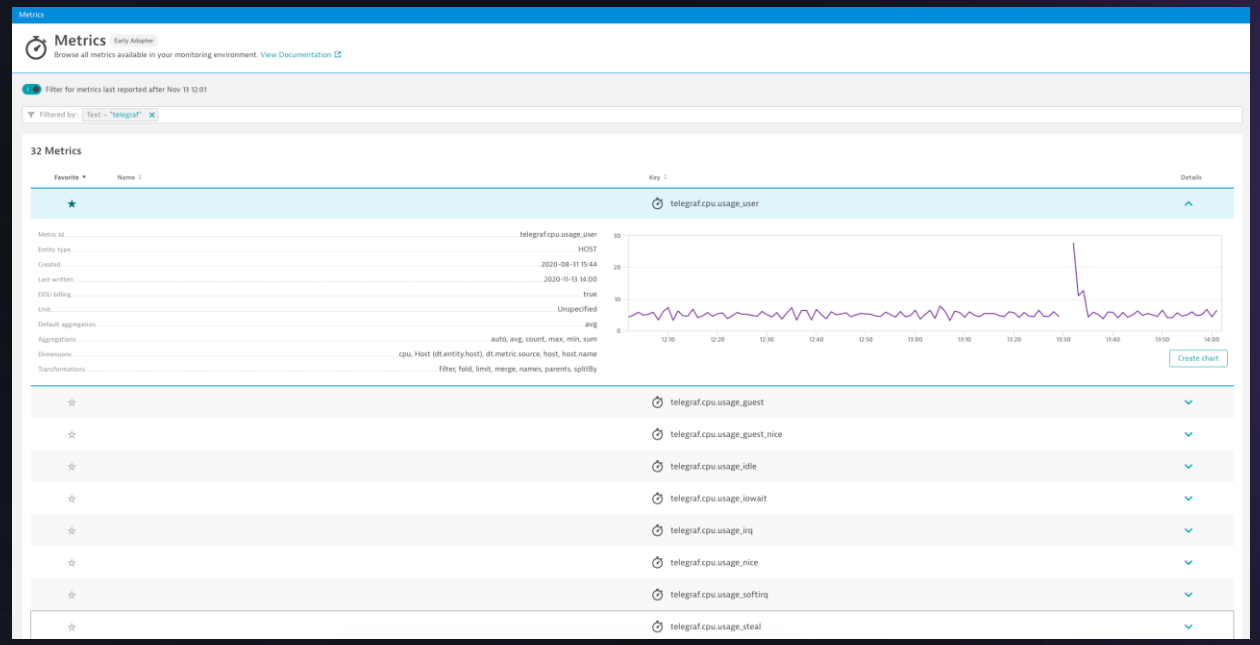
- Monotonic count metric (send total value, delta calculation by DT cluster)

```
new_user_count,region=EAST count,1000
new_user_count,region=EAST count,1500
```

Delta=500

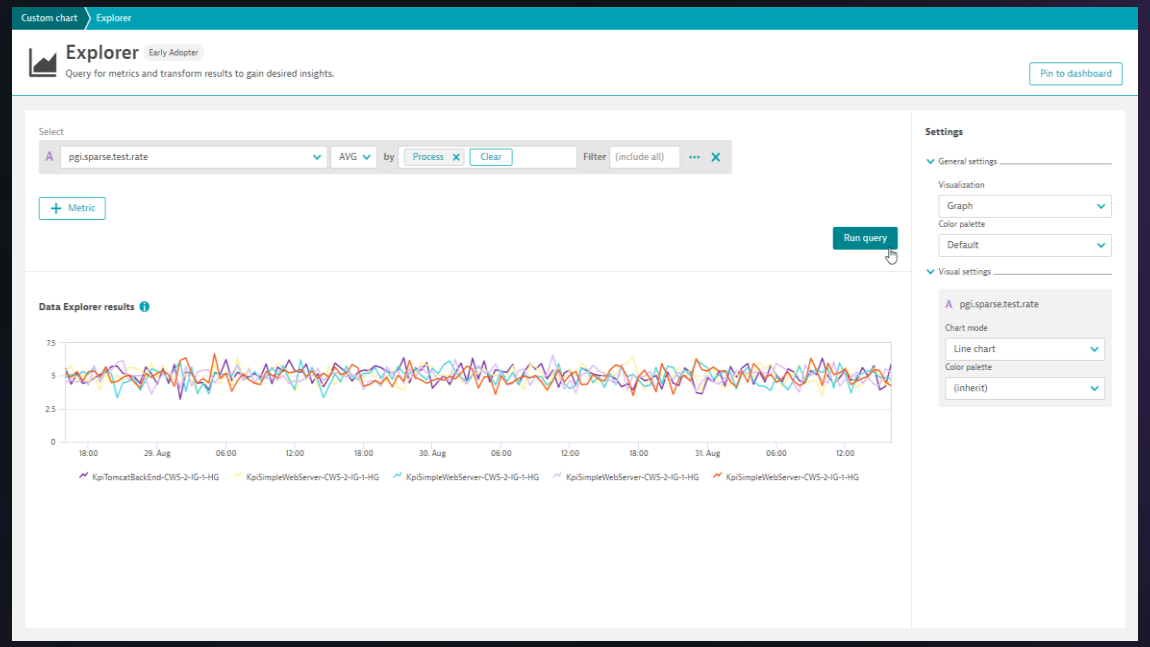
Metric Browser

- New “Home” for all metrics
- Search metrics by name
- Filter for active / recently reported metrics
- Drill-down to metrics information
- Chart preview
- Available with Dynatrace version 1.207



Metric Explorer

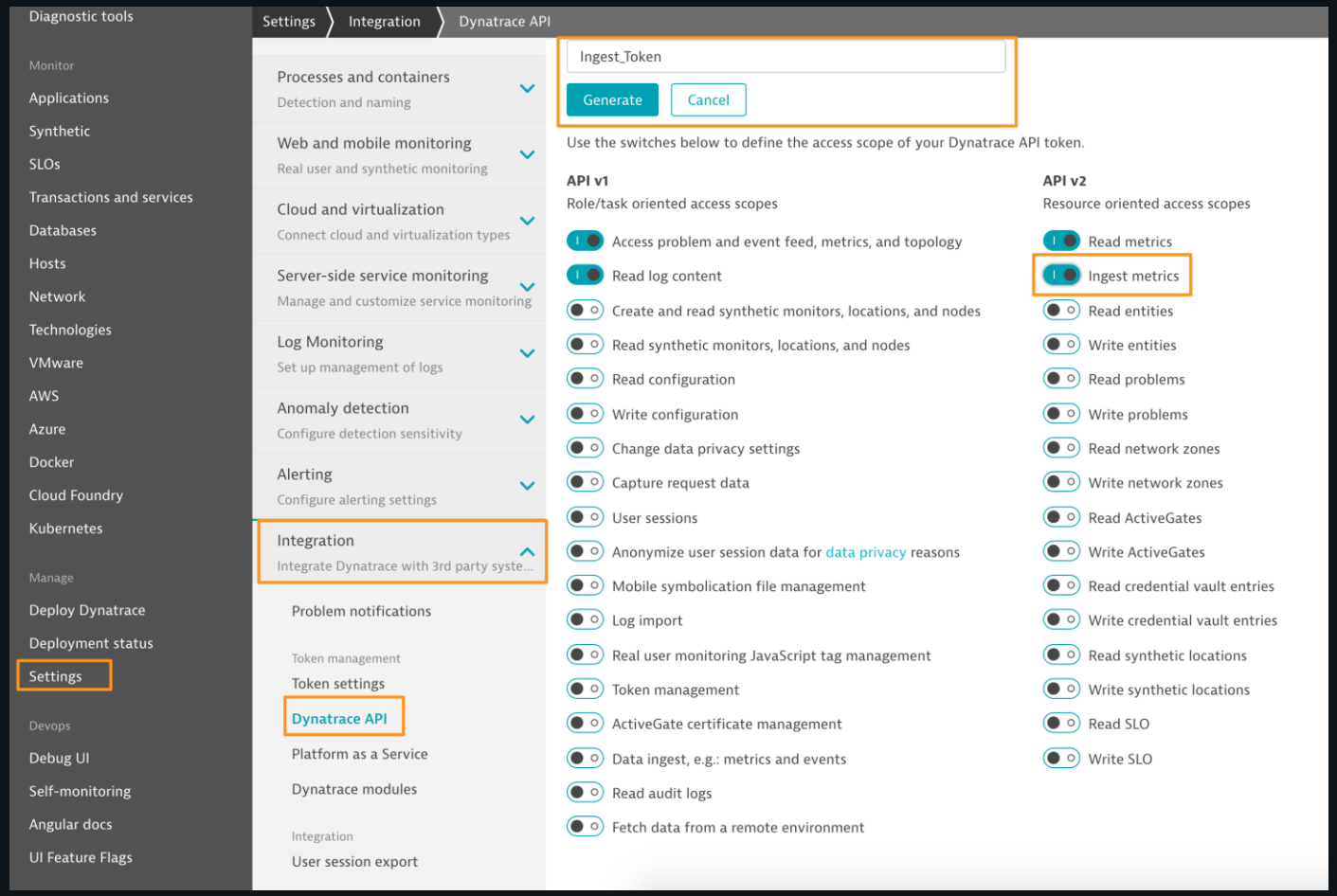
- Support for all metrics
- Allows to query and visualize
- Support 5 basic visualizations
- Pin to Dashboard
- Early Adopter since Version 1.202 - Accessible via "Custom Chart"



Ingest exercises

Setup

- Create an API Token with access scope “Ingest metrics”



Ingest Exercise – Ingest via Public Rest-API

- Use CURL to ingest a sample metric to your environment via public REST API

```
curl -H "Authorization: Api-Token <TOKEN>" -X POST -H "Content-Type: text/plain; charset=utf-8"
--data-ascii "mymetric.rest 100" https://<env-URL>/api/v2/metrics/ingest
```

- https://github.com/Dynatrace/perform-2021-hotday/blob/main/Ingest%20all%20the%20metrics/ingest_curl.sh

Ingest Exercise – Ingest via One Agent channel

- Use CURL to ingest a sample metric to your environment via One Agent

```
curl -X POST -H "Content-Type: text/plain; charset=utf-8" --data-ascii "mymetric.oa 200"
http://localhost:14499/metrics/ingest
```

- https://github.com/Dynatrace/perform-2021-hotday/blob/main/Ingest%20all%20the%20metrics/ingest_curl.sh

Ingest Exercise – Scripting integration

- Use `dynatrace_ingest` utility instead of CURL

```
mpstat -P ALL | tail -n +5 | awk {'print "mymetric.host.cpu.usr,core="$2"
"$3"\nmymetric.host.cpu.idle,core="$2" "$12;'} | dynatrace_ingest -v

ps -eo pcpu,pid,user,args --no-headers | sort -k 1 -r | head -10 | awk {'print
"mymetric.cpu_pid,pid="$2" " $1;'} | dynatrace_ingest -v
```

- https://github.com/Dynatrace/perform-2021-hotday/blob/main/Ingest%20all%20the%20metrics/dynatrace_ingest_samples.sh

Ingest Exercise – Metric browser “mymetric.rest”

Metrics

Metrics Early Adopter

Browse all metrics available in your monitoring environment. View Documentation

Filter for metrics last reported after Jan 05 11:28

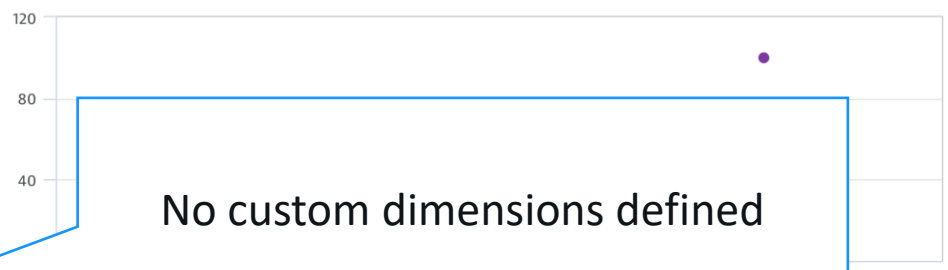
Filtered by: Text: mymetric.rest Clear all

Favorites
- Any
- Yes
- No

1 Metric

Favorite	Name	Key	Details
☆	mymetric.rest	mymetric.rest	^

Metric key	mymetric.rest
Created	2021-01-05 10:52
Last written	2021-01-05 11:57
DDU billing	true
Unit	Unspecified
Default aggregation	avg
Aggregations	auto, avg, count, max, min, sum
Dimensions	
Transformations	filter, fold, limit, merge, names, parents, splitBy



No custom dimensions defined

Create chart

1

Ingest Exercise – Metric browser “mymetric.oa”

Metrics

Metrics Early Adopter

Browse all metrics available in your monitoring environment. View Documentation

Filter for metrics last reported after Jan 05 00:00

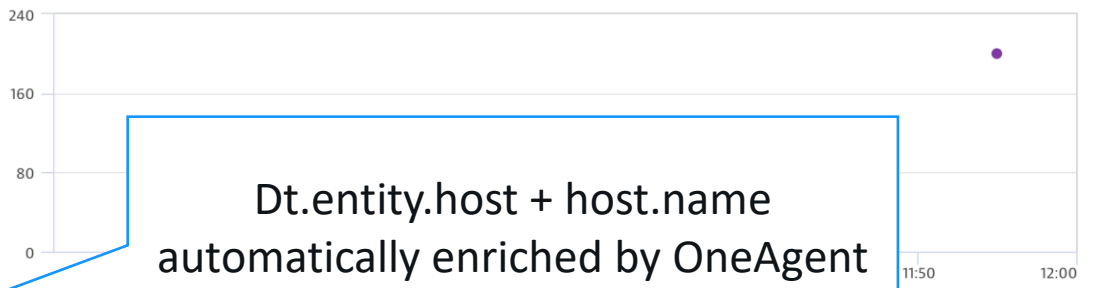
Filtered by: Text ~ "mymetric.oa"

Favorites

1 Metric

Favorite	Name	Key	Details
☆		mymetric.oa	

Metric key	mymetric.oa
Entity type	HOST
Created	2021-01-05 10:54
Last written	2021-01-05 11:55
DDU billing	true
Unit	Unspecified
Default aggregation	avg
Aggregations	auto, avg, count, max, min, sum
Dimensions	Host (dt.entity.host), host.name
Transformations	filter, fold, limit, merge, names, parents, splitBy



Create chart

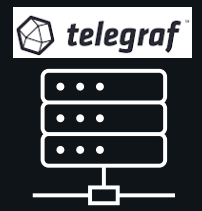
Dt.entity.host + host.name automatically enriched by OneAgent

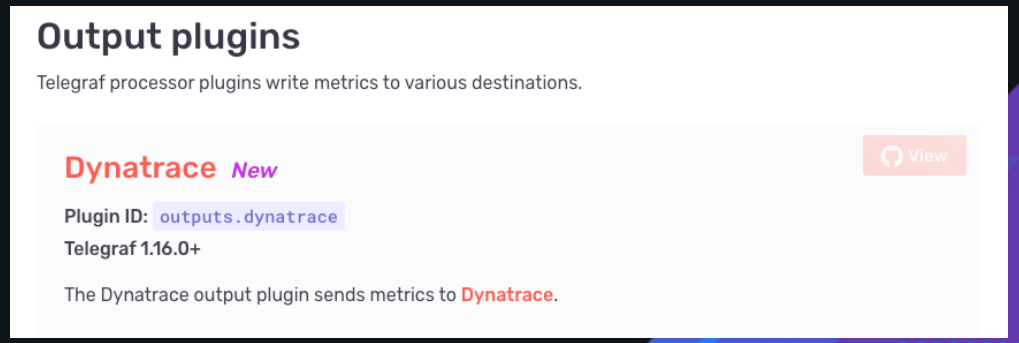
1

Telegraf Data Ingest

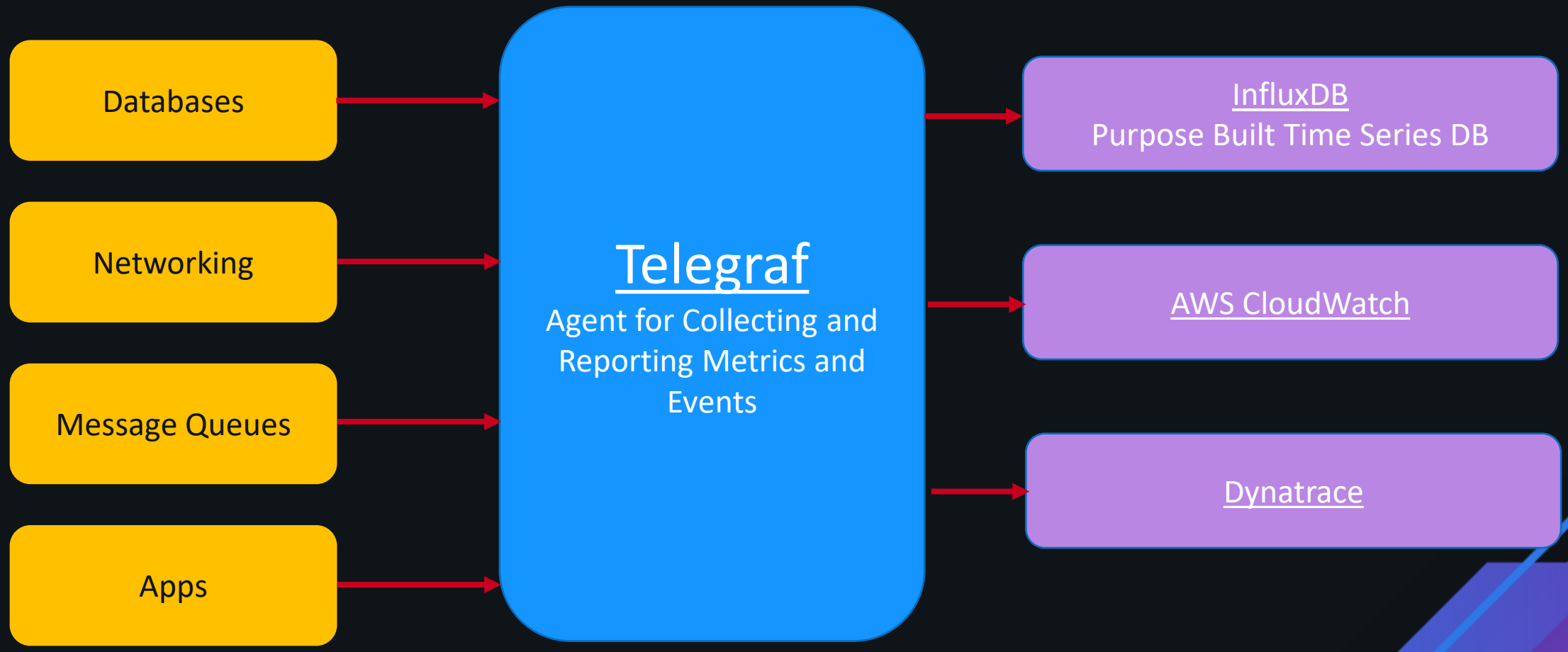
Example - Simple & fast integration via Telegraf

- **Telegraf** is an Open Source Agent to collect metrics
- Dynatrace integration via output plugin
- ~200 input plugins offer access to a variety of metric sources
- Easy & fast setup for Dynatrace
  - Alongside with OneAgent
  - Via Public Rest-API





Telegraf Overview



Telegraf With Dynatrace OneAgent

- Configure Telegraf Dynatrace output With OneAgent
- Configure Telegraf Dynatrace output Without OneAgent

Telegraf with OneAgent

Telegraf Output With Dynatrace OneAgent

- Navigate to the telegraf conf file on VM with OneAgent
  - sudo su
  - cd /etc/telegraf
- Edit the master telegraf.conf file. This is where all input and output plugins are configured
  - vi telegraf.conf
- Search for the Dynatrace output plugin with the keyword *dynatrace*
  - / Dynatrace
- Press “i” to edit the file to uncomment the lines:
  - [[outputs.dynatrace]]
  - prefix = “telegraf.”
- Press escape

Telegraf With Dynatrace OneAgent

```
[[outputs.dynatrace]]
  ## Leave empty or use the local ingest endpoint of your OneAgent monitored host (e.g.: http://127.0.0.1:14499/metrics/ingest).
  ## Set Dynatrace environment URL (e.g.: https://YOUR_DOMAIN/api/v2/metrics/ingest) if you do not use a OneAgent

  #api_token = "vC-Hzx73RK6xy4dvk7yLL"

  ## Optional prefix for metric names (e.g.: "telegraf.")
  prefix = "telegraf."
  ## Flag for skipping the tls certificate check, just for testing purposes, should be false by default
```

Telegraf Input Plugins with OneAgent

Telegraf Input Plugins With Dynatrace OneAgent

- Search for the input plugins fields
  - / INPUT PLUGINS

```
###############################################################################
#                                INPUT PLUGINS                                #
###############################################################################


# Read metrics about cpu usage
[[inputs.cpu]]
  ## Whether to report per-cpu stats or not
  percpu = true
  ## Whether to report total system cpu stats or not
  totalcpu = true
  ## If true, collect raw CPU time metrics.
  collect_cpu_time = false
  ## If true, compute and report the sum of all non-idle CPU states.
  report_active = false
```

- Press “i” to edit the file to uncomment the lines:
  - [[inputs.cpu]]
  - percpu = true
  - totalcpu = true
  - collect_cpu_time = false
  - report_active = false

Telegraf Input Plugins With Dynatrace OneAgent

- Press “i” to edit the file and scroll dow
  plugins:
  - [[inputs.kernel]]
  - [[inputs.mem]]
  - [[inputs.processes]]
  - [[inputs.swap]]
  - [[inputs.system]]

```
# Get kernel statistics from /proc/stat
[[inputs.kernel]]
  # no configuration


# Read metrics about memory usage
[[inputs.mem]]
  # no configuration


# Get the number of processes and group them by status
[[inputs.processes]]
  # no configuration


# Read metrics about swap memory usage
[[inputs.swap]]
  # no configuration


# Read metrics about system load & uptime
[[inputs.system]]
  ## Uncomment to remove deprecated metrics.
  # fielddrop = ["uptime_format"]
```

Telegraf Input Plugins With Dynatrace OneAgent

- Save the file
  - Press “Escape” key, then type “wq!” . Press Enter
- Restart Telegraf
  - service telegraf status
  - service telegraf stop
  - service telegraf status
  - service telegraf start

Telegraf without OneAgent

Telegraf Without Dynatrace OneAgent

- What you'll need:
  - API Token
  - Dynatrace SaaS URL endpoint

Dynatrace API Token

Settings > Integration > Dynatrace API

Preferences
Environment settings

Accounting
Verify consumption

Telegraf

Generated token

We're improving security of your tokens. Revealing the token will no longer be possible in future versions of Dynatrace. You will only see a token once – upon its creation.

Reveal token

Use the switches below to define the access scope of your Dynatrace API token.

API v1

Role/task oriented access scopes

- [x] Access problem and event feed, metrics, and topology
- [x] Read log content
- [ ] Deprecated: Configure maintenance windows
- [ ] Create and read synthetic monitors, locations, and nodes
- [ ] Read synthetic monitors, locations, and nodes
- [ ] Read configuration
- [ ] Write configuration
- [ ] Change data privacy settings
- [ ] Capture request data
- [ ] User sessions

API v2

Resource oriented access scopes

- [x] Read metrics
- [x] Ingest metrics
- [ ] Read entities
- [ ] Write entities
- [ ] Read network zones
- [ ] Write network zones
- [ ] Read ActiveGates
- [ ] Write ActiveGates
- [ ] Read credential vault entries
- [ ] Write credential vault entries

Telegraf Without Dynatrace OneAgent

- Navigate to the telegraf conf file on VM with OneAgent
  - sudo su
  - cd /etc/telegraf
- Edit the master telegraf.conf file. This is where all input and output plugins are configured
  - vi telegraf.conf
- Search for the Dynatrace output plugin with the keyword *dynatrace*
  - / dynatrace
- Press “i” to edit the file to un-comment the lines and add in the values:
  - [[outputs.dynatrace]]
  - url = “https://{your-environment-id}.live.dynatrace.com/api/v2/metrics/ingest
  - api_token = “{your-api-token}”
  - prefix = “telegraf.no-agent.”
  - Insecure_skip_verify = false

Telegraf Without Dynatrace OneAgent

```
[[outputs.dynatrace]]
  ## Leave empty or use the local ingest endpoint of your OneAgent monitored host (e.g.: http://127.0.0.1:14499/metrics/ingest).
  ## Set Dynatrace environment URL (e.g.: https://YOUR_DOMAIN/api/v2/metrics/ingest) if you do not use a OneAgent
  url = "https://[redacted].live.dynatrace.com/api/v2/metrics/ingest"
  api_token = "[redacted]"
  ## Optional prefix for metric names (e.g.: "telegraf.")
  prefix = "telegraf-noagent."
  ## Flag for skipping the tls certificate check, just for testing purposes, should be false by default
  insecure_skip_verify = false
```

Telegraf Input Plugins With Dynatrace OneAgent

- Search for the input plugins fields
  - / INPUT PLUGINS

```
###############################################################################
#                                INPUT PLUGINS                                #
###############################################################################


# Read metrics about cpu usage
[[inputs.cpu]]
  ## Whether to report per-cpu stats or not
  percpu = true
  ## Whether to report total system cpu stats or not
  totalcpu = true
  ## If true, collect raw CPU time metrics.
  collect_cpu_time = false
  ## If true, compute and report the sum of all non-idle CPU states.
  report_active = false
```

- Press “i” to edit the file to uncomment the lines:
  - [[inputs.cpu]]
  - percpu = true
  - totalcpu = true
  - collect_cpu_time = false
  - report_active = false

Telegraf Input Plugins With Dynatrace OneAgent

- Press “i” to edit the file and scroll down a few lines in the file to uncomment the lines for the following plugins:
  - [[inputs.kernel]]
  - [[inputs.mem]]
  - [[inputs.processes]]
  - [[inputs.swap]]
  - [[inputs.system]]

```
# Get kernel statistics from /proc/stat
[[inputs.kernel]]
  # no configuration


# Read metrics about memory usage
[[inputs.mem]]
  # no configuration


# Get the number of processes and group them by status
[[inputs.processes]]
  # no configuration


# Read metrics about swap memory usage
[[inputs.swap]]
  # no configuration


# Read metrics about system load & uptime
[[inputs.system]]
  ## Uncomment to remove deprecated metrics.
  # fielddrop = ["uptime_format"]
```

Break 15 Mins

Licensing

Licensing

- Pricing for custom metric ingest via Davis data units
  - Host-unit included metrics can be used (requires relation to host-entity)
- Each reported datapoint will consume DDU
  - Metric dimensions & splittings might produce multiple datapoints
    - E.g.

      CPU metric per host (1 datapoint per host)

      vs.

      CPU metric per host/CPU (1-n datapoints per host)
- Consumption overviews via UI and API

Pricing

- Host-included metrics can be used (via host dimension in data ingest)
- Pricing for metric ingest via Davis data units
- Each reported datapoint will consume DDU (with weight 0.001)

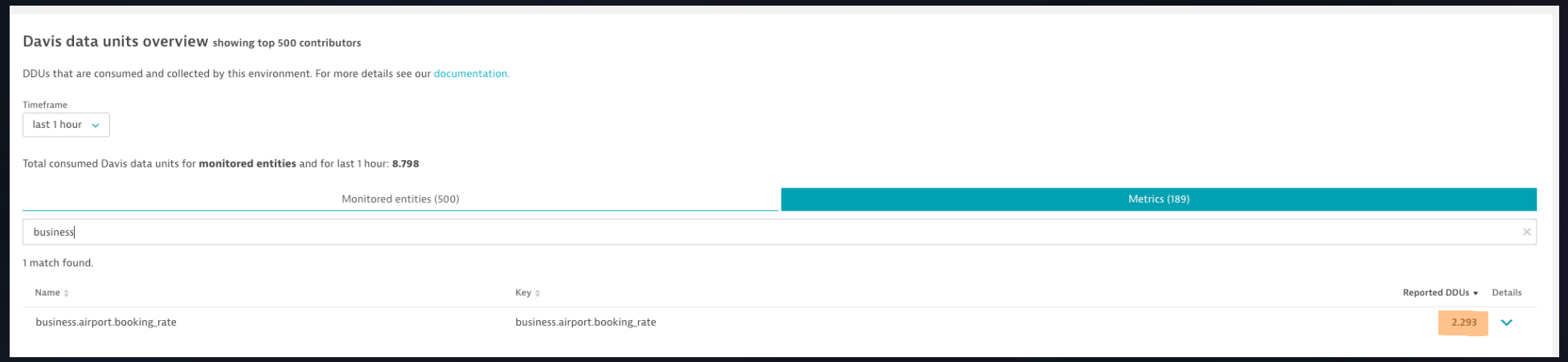
  Minute-by-minute reported metric will consume 526 DDUs per dimension per year

```
cpu.temp,cpu=cpu1,cpu_type=INTEL 50
cpu.temp,cpu=cpu2,cpu_type=INTEL 55
```

  → 2 dimensions will consume 526 * 2 = 1.052 DDUs per year

- Consumption overviews via UI and own DDU billing metrics

DDU consumption overview screens




```
{
    "totalCount": 1,
    "nextPageKey": null,
    "result": [
        {
            "metricId": "builtin:billing.ddu.metrics.byMetric:filter(prefix(Metric~ Key,business~.airport~.booking_rate))",
            "data": [
                {
                    "dimensions": [
                        "business.airport.booking_rate"
                    ],
                    "dimensionMap": {
                        "Metric Key": "business.airport.booking_rate"
                    },
                    "timestamps": [
                        1598279460000
                    ],
                    "values": [
                        2.2930009365081787
                    ]
                }
            ]
        }
    ]
}
```


DDU consumption can also be queried via API

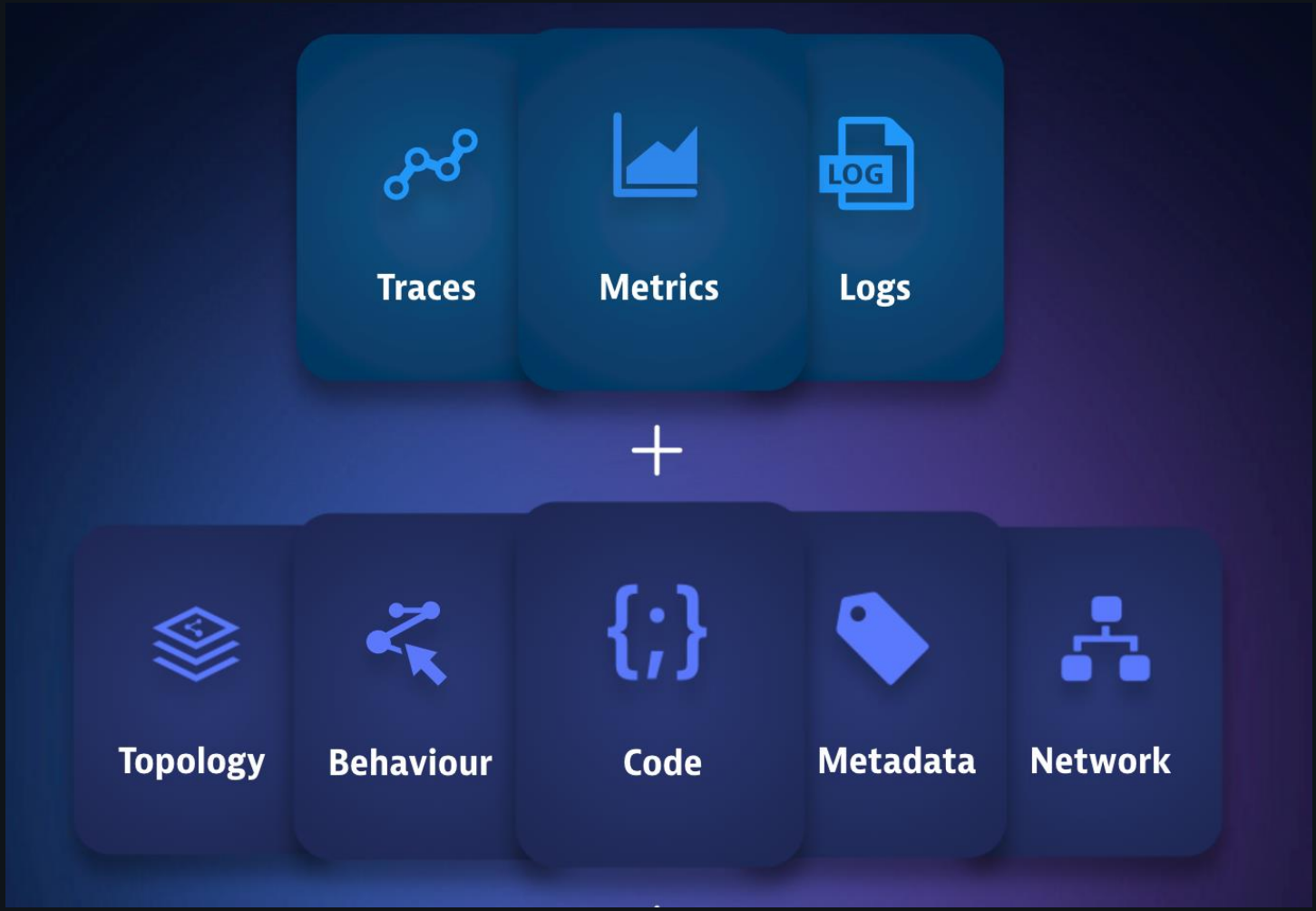
`builtin:billing.ddu.metrics.byMetric`

→ Shows **reported** DDUs
→ **Consumed** DDUs are depending on host-included metrics
→ Consumption shows up in the Monitored entites tab

Additional outputs

Ingest All the Logs

Key pillars for intelligent observability



Logs & events

JAVA application log

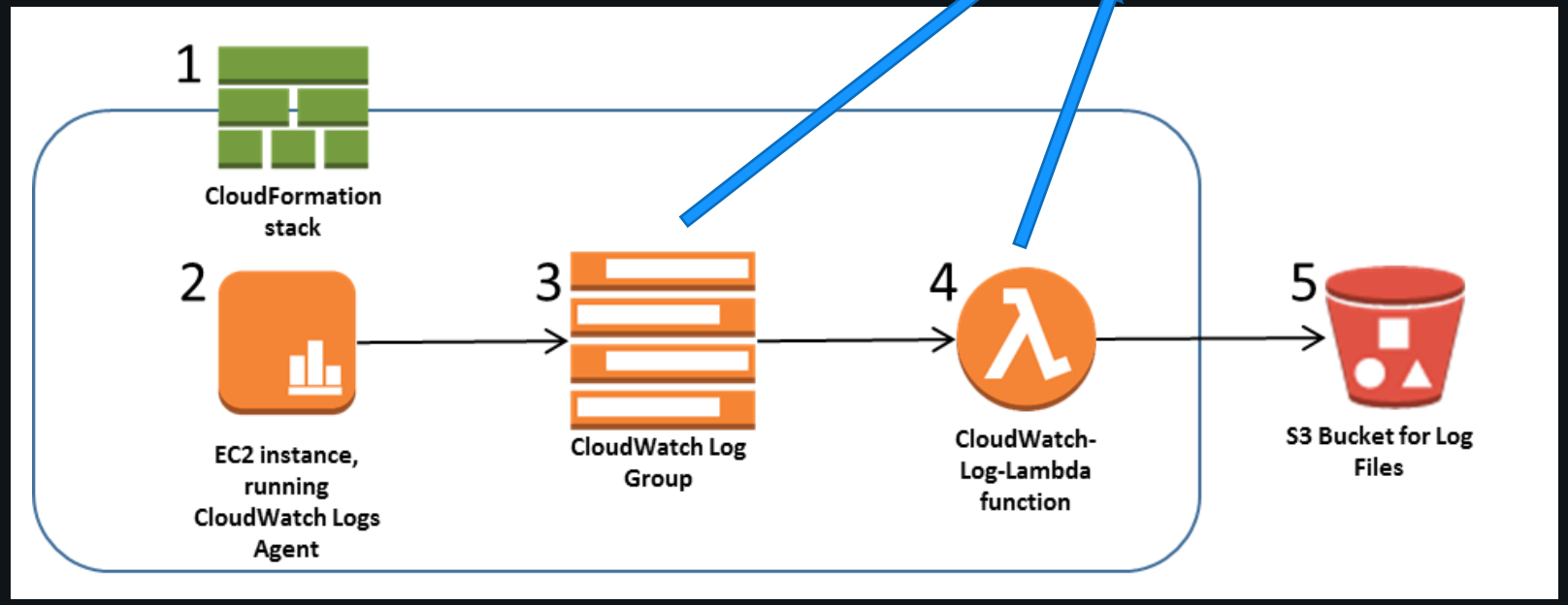
```
2020-12-17 09:19:09 INFO MacBook-Pro.local --- [Thread-8]: Shutting down ExecutorService 'taskScheduler'
```

Kubernetes Event


```
{
  "timestamp":"2020-12-17 09:20:17",
  "type":"Normal",
  "reason":"Pulling",
  "object":"pod/hello-minikube-5cd8589d76-cs578",
  "message":"Pulling image k8s.gcr.io/echoserver:1.4"
}
```

Cloud Service – logs & events

Dynatrace Log & Events Ingest



- CloudTrail events (audit)
- Cloud health events
- Serverless logs
- Cloud services logs (CloudWatch log group)

Source

Plug & Play architecture for streaming logs & events

- Dynatrace Log Ingest as a stream receiver

Cloud Service – logs & events

- CloudTrail event


```
"eventVersion": "1.08",
"userIdentity": {
  "type": "AWSService",
  "invokedBy": "cloudtrail.amazonaws.com"
},
"eventTime": "2021-01-04T15:44:15Z",
"eventSource": "s3.amazonaws.com",
"eventName": "PutObject",
"awsRegion": "us-east-1",
"sourceIPAddress": "cloudtrail.amazonaws.com",
"userAgent": "cloudtrail.amazonaws.com",
"requestParameters": {
  "bucketName": "dynatracecloudtrail",
```


- AWS Lambda execution log

```
▶   2020-12-03T21:30:38.556+01:00   REPORT RequestId: ab89ec8a-e4d0-47e7-9635-bf6beec47a9e Duration: 316.81 ms Billed Duration: 317 ms Memory Size:
                                    128 MB Max Memory Used: 68 MB Init Duration: 154.21 ms
```

Ingest

OneAgent, New log & events **ingest**

Explore

New Log & Events explorer

Detect

Log streams based alerting, RCA

New log & events ingest

New logs & events ingest

- Make it **easy** and **attractive** for customers to **push logs & events** into Dynatrace
- **Seamless integration** of custom log streams into the Dynatrace software intelligence platform
  - Connect entities with business KPIs extracted from log & events
  - Feed **Davis AI** and **Smartscape** with your own business-critical log
- Offer further **integrations** from a multitude of log data sources
  - Cloud Services logs
  - Kubernetes logs & events

New logs & events ingest

http payload + API token

JSON or plain text over HTTP
(TCP/UDP coming soon)

Rest ENDPOINT
https://<env-URL>/api/v2/logs/ingest

Log file - tail

Autodiscovery

Logs from Host via OneAgent

New logs & events ingest

Rest ENDPOINT
https://<env-URL>/api/v2/logs/ingest
https://ip_of_activegate:port/e/env_id/api/v2/logs/ingest

- Private Active Gate supported
- POST http method
- HTTP auth header: Authorization: Api-Token abcdefgh1234
  - Log Import API token
- Plain text format
  - Content-Type: text/plain; charset=utf-8
- JSON format
  - Content-Type: application/json; charset=utf-8

New logs & events ingest

Rest ENDPOINT
https://<env-URL>/api/v2/logs/ingest
https://ip_of_activegate:port/e/env_id/api/v2/logs/ingest

- Dynatrace API Explorer (*Generic Log Ingest is enabled on each public ActiveGate of the Cluster*)

POST `/logs/ingest` Process and store logs — In development

Ingested logs are stored to indexed log storage.

Required Api-Token scope:
`LogImport` (Log import)

Parameters — Try it out

No parameters

Request body — application/json; charset=utf-8

The log message in JSON or plain text format, depending on request Content-Type specification. In case of plain text format, message represents a single log event. In case of JSON format, message can be a single JSON object representing a single event or an array of JSON objects representing a set of events.

Examples: JSON format

Example Value | Schema

```
[
  {
    "content": "example log content 1",
    "dt.logpath": "/var/log/syslog"
  },
  {
    "content": "example log content 2"
  }
]
```

Example Description
JSON array representing two events.

New logs & events ingest

Rest ENDPOINT
https://<env-URL>/api/v2/logs/ingest
https://ip_of_activegate:port/e/env_id/api/v2/logs/ingest

Plain text

- message represents a single log event
- time of log event is ingestion time

```
curl -X POST --data "Log content" https://<env-URL>/api/v2/logs/ingest --
header "Content-Type:text/plain; charset=utf-8" --header
"Authorization: Api-Token abcdefgh1234"
```

New logs & events ingest

Rest ENDPOINT
https://<env-URL>/api/v2/logs/ingest
https://ip_of_activegate:port/e/env_id/api/v2/logs/ingest

JSON format

- The message can be a single JSON object representing a single event or an array of JSON objects representing a set of events
- **Content key**
  - List of supported keys: body, content, message, payload
  - Full tokenization of content
  - If not present – whole message treated as content
- Time of log event
  - List of supported keys: @timestamp, Timestamp, _timestamp, date, eventTime, published_date, syslog.timestamp, timestamp
  - Supported date time formats: RFC3339, RFC3164, UNIX
  - If now time key present in JSON: current time

New logs & events ingest

Rest ENDPOINT
https://<env-URL>/api/v2/logs/ingest
https://ip_of_activegate:port/e/env_id/api/v2/logs/ingest

JSON format – semantic attributes:

- List of reserved attributes that are indexed during log event ingest
- Fast search & aggregations created for these attributes

<u>Open telemetry log model standarization:</u>

- Trace context fields – correlation with trace context
  - `Trace ID, span ID`
- Log event attributes – describing event itself
  - `status, http method, http status code, response size`
- Log event resources – describing log source (producer)
  - `cloud region, host name, process id`

Excersise 1

Using generic log ingest

Excersise 1 - Using generic log ingest

Rest ENDPOINT
https://<env-URL>/api/v2/logs/ingest
https://ip_of_activegate:port/e/env_id/api/v2/logs/ingest

1. **Create and save log ingest security token**

   Settings -> Integrations -> Dynatarce API -> Generate Token

2. **Obtain Rest Endpoint for ingesting log events**
3. **Use curl (or Postaman)**

   JSON event

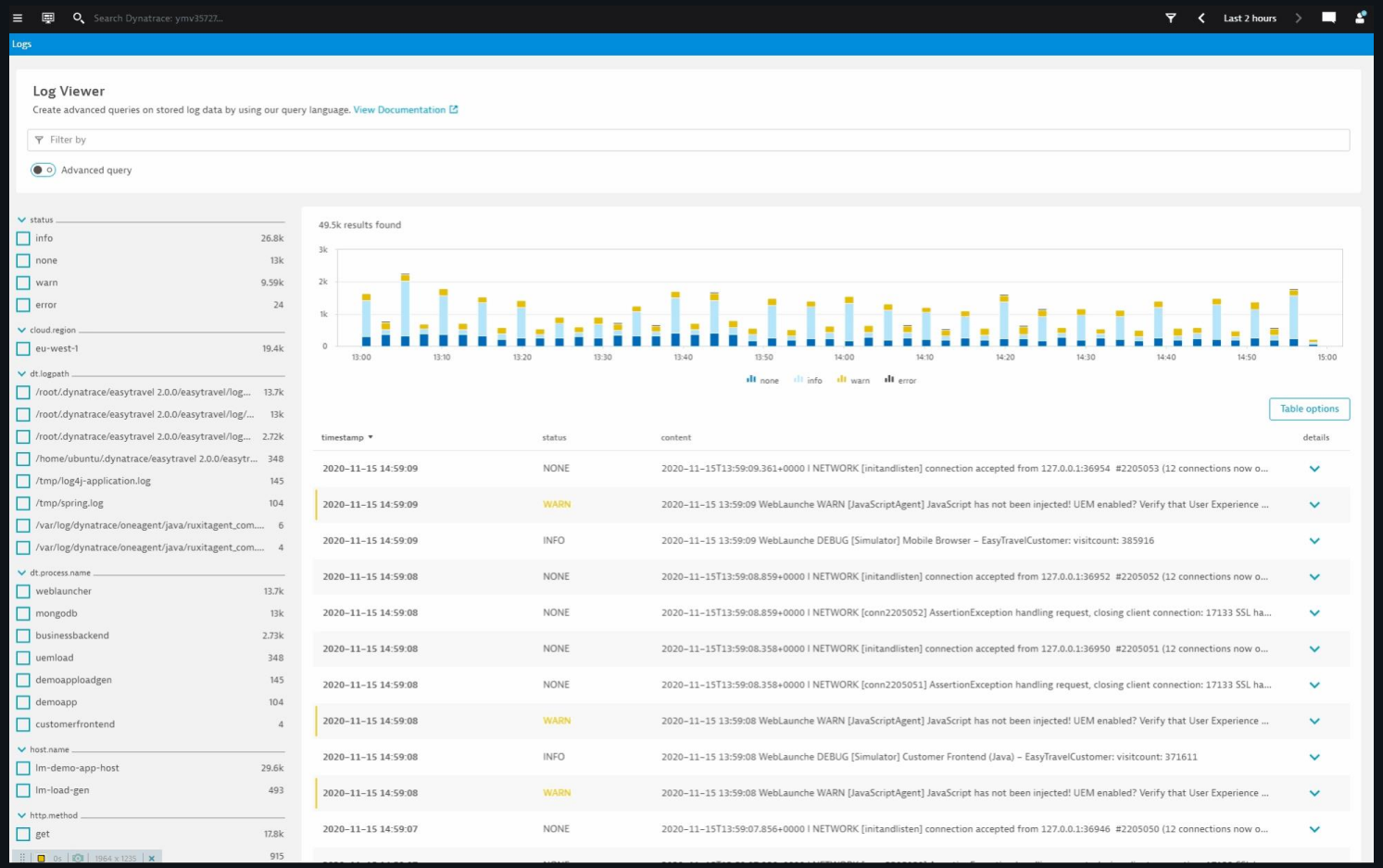
```
curl -X POST "https://<env-URL>/api/v2/logs/ingest" -H "accept: */*" -H
"Authorization: Api-Token ***" -H "Content-Type: application/json;
charset=utf-8" --data "{\"content\":\"JSON log content\"}"
```

   Plain text event

```
curl -X POST --data "Log content" https://<env-URL>/api/v2/logs/ingest -
-header "Content-Type:text/plain; charset=utf-8" --header
"Authorization:Api-Token ******"
```

New log & events Explorer

New log & events Explorer



- Quickly investigate and troubleshoot issues
  - Search, analyze and explore log and event data from any source
  - Get log data with context (Host, Cloud Region, Pod Name etc.)
- Analyze log data without the need of a query language (but query language is here for you)
- Powerful API

Excersise 2

Explore log viewer & API

Exercise 2 - Explore log viewer & API

1. Execute search by
    1. Content field
    2. Semantic attribute
2. Explore faceting component
3. Explore statefull behavior of Log viewer
4. Use API to:
    1. Query log data
    2. Get aggregation results

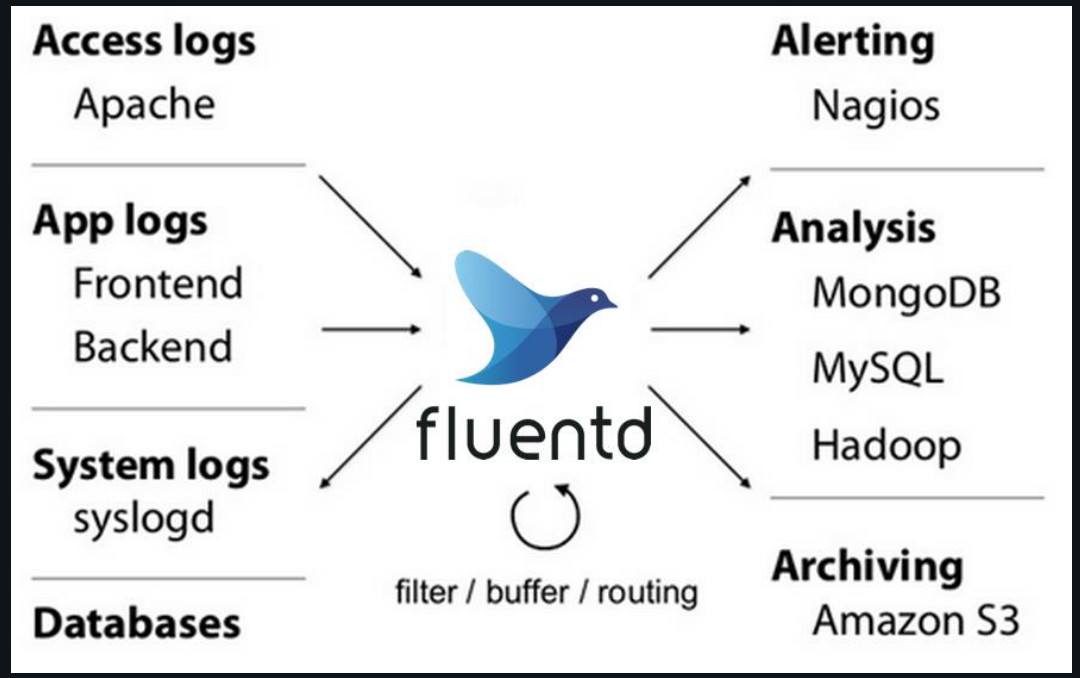
```
curl -X GET "https://<env-URL>/api/v2/logs/aggregate?query=content%3D%22fluentd%22&timeBuckets=1&maxGroupValues=10" -H "accept: application/json; charset=utf-8" -H "Authorization: Api-Token *"
```

FluentD log & events Ingest

FluentD - intro

FluentD is an open-source data collector, which lets you unify the data collection and consumption for a better use and understanding of data.

- Unified Logging with JSON
- Pluggable Architecture
- Built-in Reliability



Source

FluentD – pluggable architecture

Input Plugin	Parsing/Transform Plugin	Output plugin
• tail • TCP/UDP • HTTP • AWS S3 • AWS CloudWatch • GCP Pub/Sub • Azure Event Hub • RabbitMQ	• Regexp parser • Grok parser • Geo IP parser • UA parser • Tag rewrite	• File • Forward • HTTP • AWS CloudWatch • ElasticSearch • RabbitMQ

- Some plugins are part of fluentD core (no need to install)
- Additional plugins need to be installed (method depends on fluentD installation type)
  - `sudo fluent-gem install fluent-plugin-cloudwatch-logs`
  - `sudo /usr/sbin/td-agent-gem install fluent-plugin-cloudwatch-logs`
- Development frameworks for creating own plugins

FluentD – configuration

FluentD configuration file location:

- for installation with td-agent package: `/etc/td-agent/td-agent.conf`
- for GEM based installation: `sudo fluentd --setup /etc/fluent`

Configuration file – list of basic directives:

1. `source` - directives determine the input sources
2. `match` - directives determine the output destinations
3. `filter` - directives determine the event processing pipelines

FluentD process restart necessary after every configuration change!

FluentD – configuration, example

```
# http://this.host:9880/myapp.access?json={"event":"data"}
<source>
  @type http
  port 9880
</source>

<filter myapp.access>
  @type record_transformer
  <record>
    host_param "#{Socket.gethostname}"
  </record>
</filter>

<match myapp.access>
  @type file
  path /var/log/fluent/access
</match>
```

Source of log stream (HTTP listener)

Tag used as the directions for FluentD internal routing engine

Log event transformation (enrich with host name)

Log stream output destination (write to file)

FluentD – logging driver for Docker

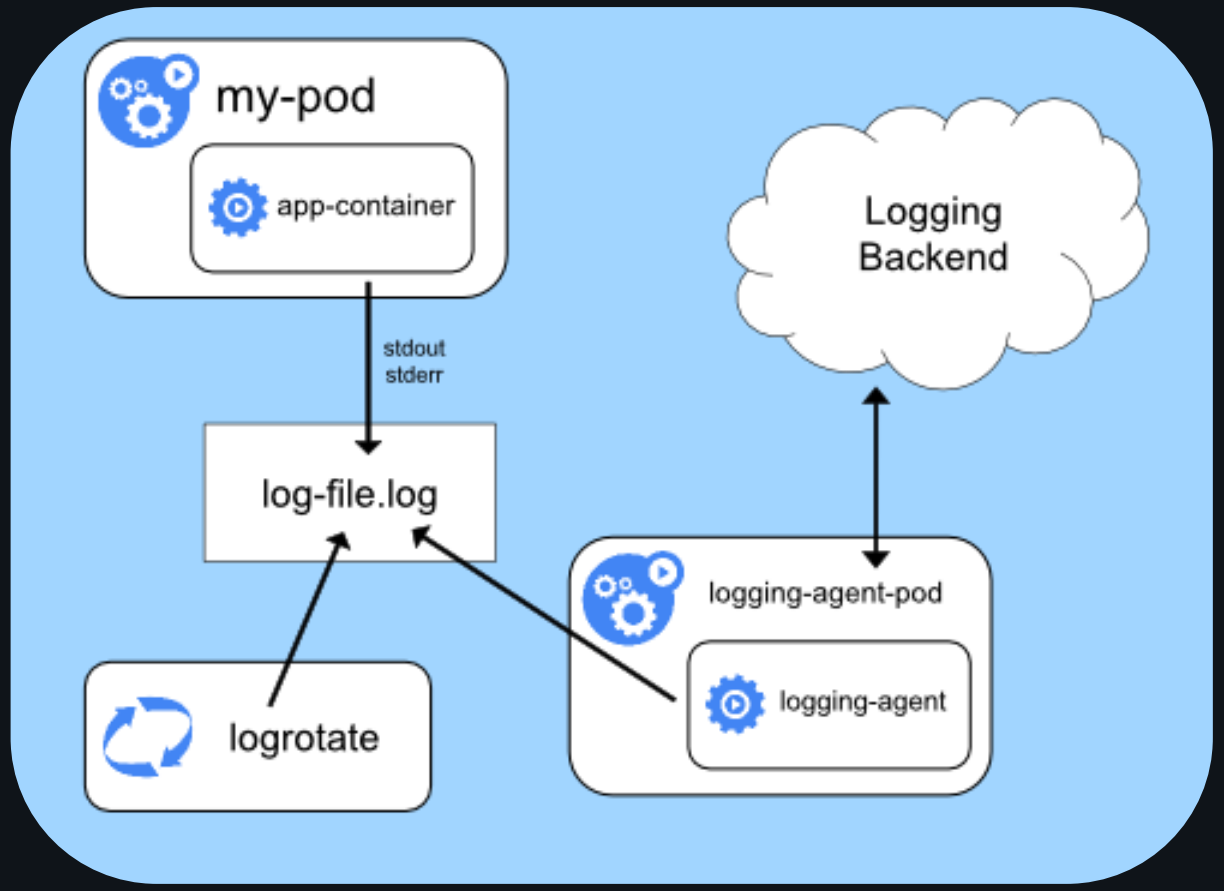
Fluentd logging driver for Docker:

- The fluentd logging driver sends container logs to the Fluentd collector as structured log data.

```
docker run -d --name config-server -e profile=dev
              --log-driver=fluentd
              --log-opt tag="docker.123"
              --log-opt fluentd-address=localhost:24224
```

Kubernetes - logging architecture, Cluster-level logging architectures

Logs should have a separate storage and lifecycle independent of nodes, pods, or containers



Source

Node logging agent

- **FluentD as a 'logging-agent'**

Dynatrace as a Logging Backend

FluentBit

- Lightweight version of FluentD
- Fully in C (FluentD – C & Ruby(
- No external dependencies for core part
- Much lower memory footprint
- Small number of plugins ~ 70 (FluentD ~ 1000)

Excersise 3

Use FluentD with Dynatrace

Configure FluentD

Exercise 3 - Use FluentD with Dynatrace

1. **Start fluentD and check status**

```
sudo systemctl start td-agent.service
```


```
sudo systemctl start td-agent.service
```


2. **Check fluentD logs**

```
sudo tail -fn 100 /var/log/td-agent/td-agent.log
```

3. **Review fluentD configuration file**

```
sudo vim /etc/td-agent/td-agent.conf
```

4. **Push log events over HTTP input plugin (default configuration)**

```
curl -X POST -d 'json={"json":"message"}'
http://localhost:8888/debug.test
```

Exercise 3 - Use FluentD with Dynatrace

1. **Configure fluentD to stream log events to Dynatrace via HTTP output plugin**

```
<match debug.http>
  @type http
  endpoint https://<env-URL>/api/v2/logs/ingest
  content_type "application/json; charset=utf-8"
  headers {"Authorization":"Api-Token ***"}
  <format>
    @type json
  </format>
  <buffer>
    flush_interval 2s
  </buffer>
</match>
```

```
curl -X POST -d 'json={"content":"Event from fluentD"}'
http://localhost:8888/debug.http
```

Exercise 3 - Use FluentD with Dynatrace

1. **Configure fluentD to stream log events from file to Dynatrace via HTTP output plugin**

```
<source>
  @type tail
  @id input_fail
  pos_file /var/log/td-agent/python.log.pos
  <parse>
    @type none
  </parse>
  path /var/log/python.log
  tag file.sample
</source>
```

1. **Start process writing to log file**

```
python logging_example.py &
```

Exercise 4 - EXTRA: send CloudTrail event to Dynatrace via fluentD

1. **Turn on logging CloudTrail events to CloudWatch log group**

```
CloudTrail -> Create new trail -> Configure CloudWatch log group
```

2. **Make sure that ec2 instance have proper IAM (access to CloudWatch log group)**
3. **Install CloudWatch logs input plugin for fluentD**

```
/usr/sbin/td-agent-gem install fluent-plugin-cloudwatch-logs
```

4. **Configure fluentD**

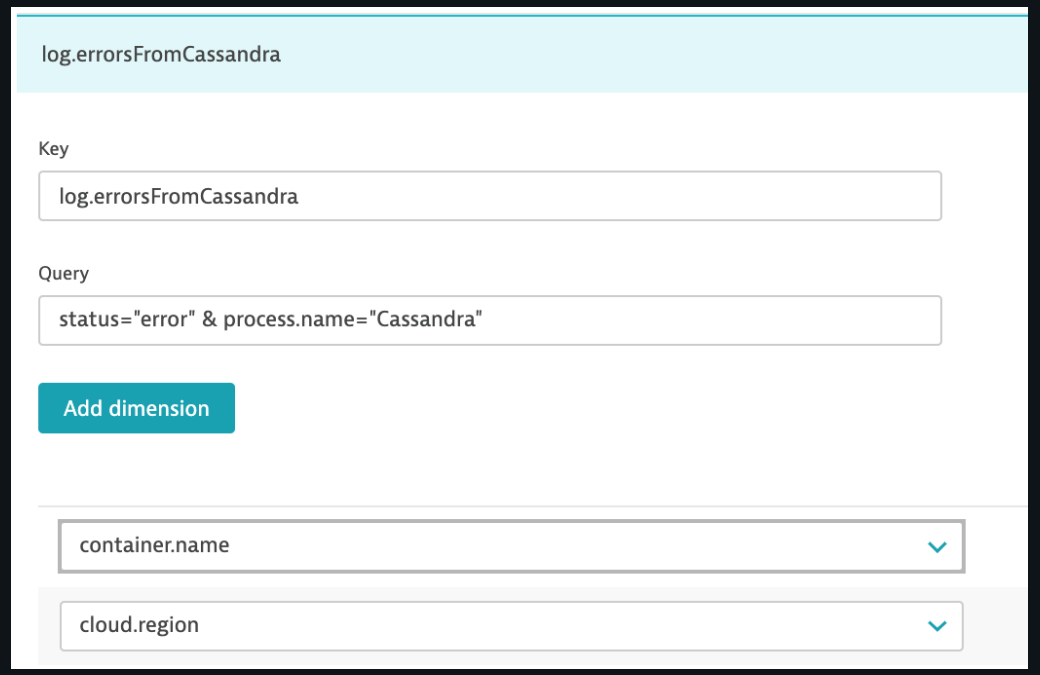
```
<source>
  @type cloudwatch_logs
  tag cloudwatch.cloudtrail
  log_group_name CloudTrailLogs
  log_stream_name 863243754016_CloudTrail_us-east-1
  region us-east-1
</source>
```

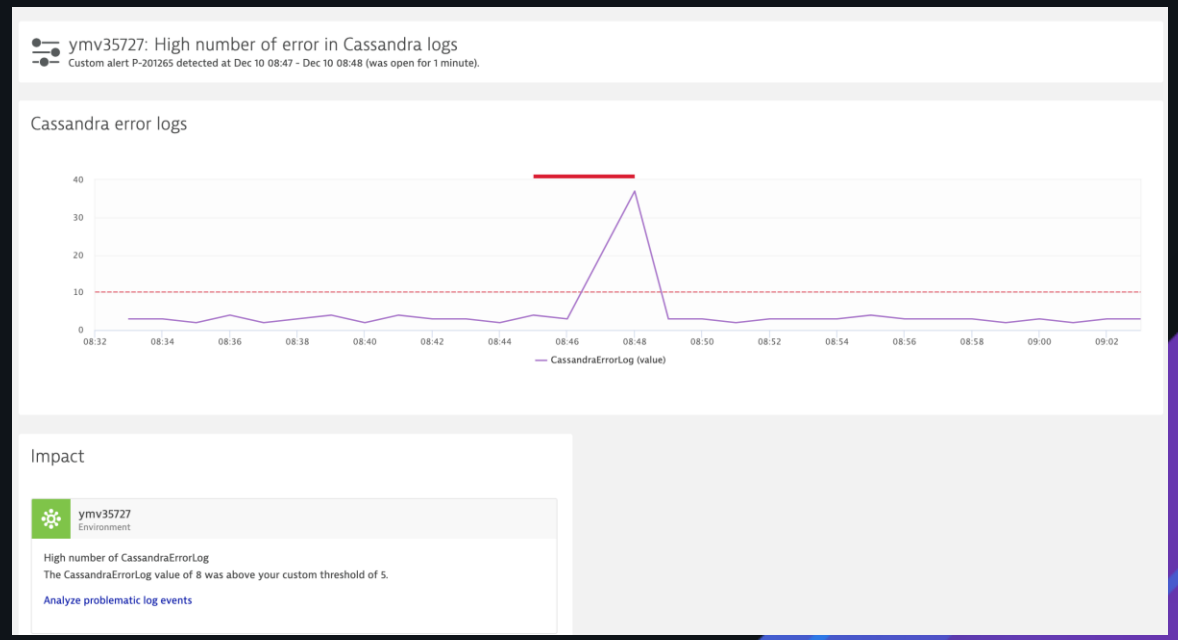
Logs & events-based alerting

Log Alerting - Review

Metrics subscription based on log & events streams.

- Empower DAVIS AI with insights from log streams, root cause analysis
- Customer alerting with dimensional log data





Break 15 Mins

Cloud Metric Ingest

Cloud Services

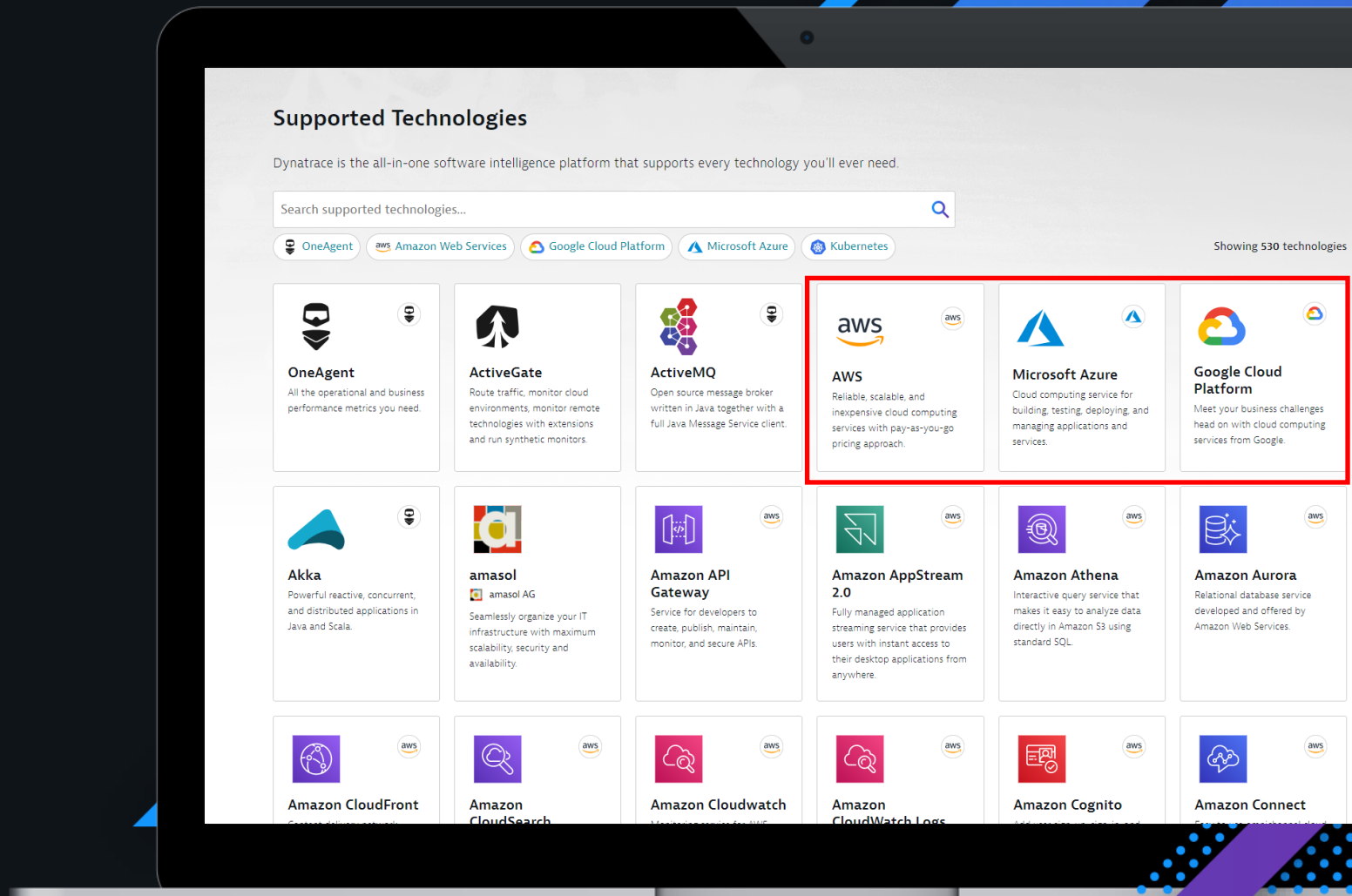
Cloud agnostic support

Support for all cloud services

93

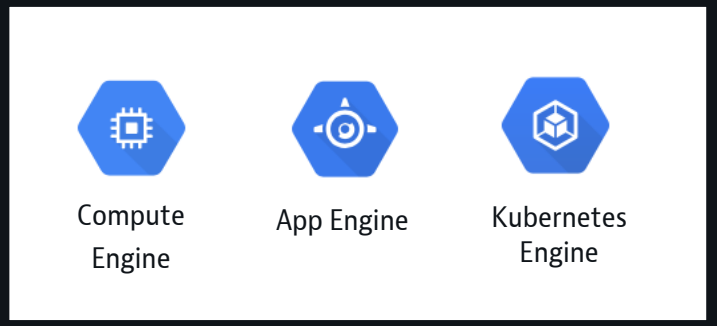
107

53



… and on full Google Cloud Platform support

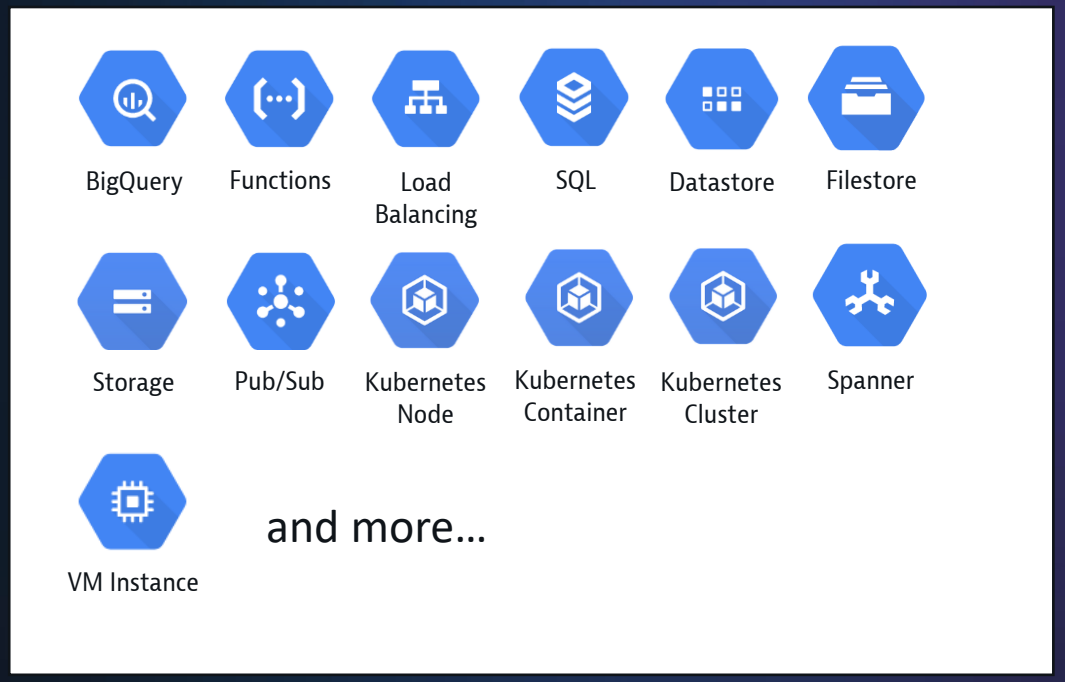
OneAgent already provides full stack monitoring for core compute services



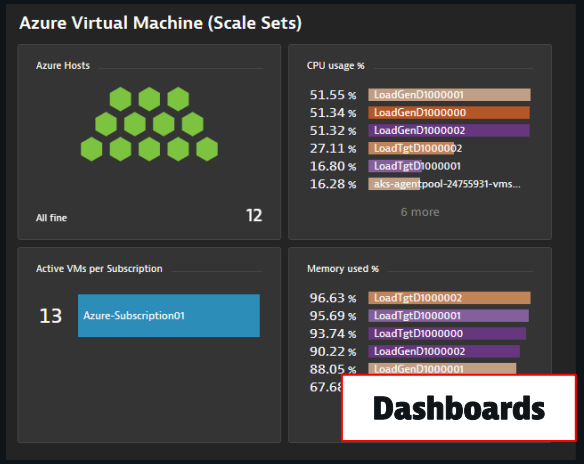
+

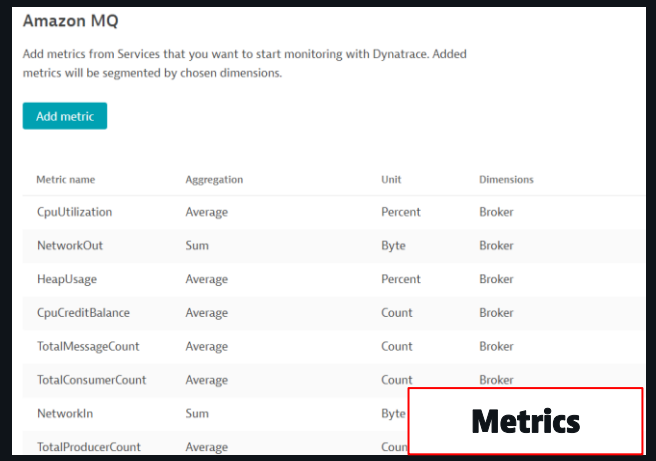
NEW: Fully integrated with Google Monitoring API

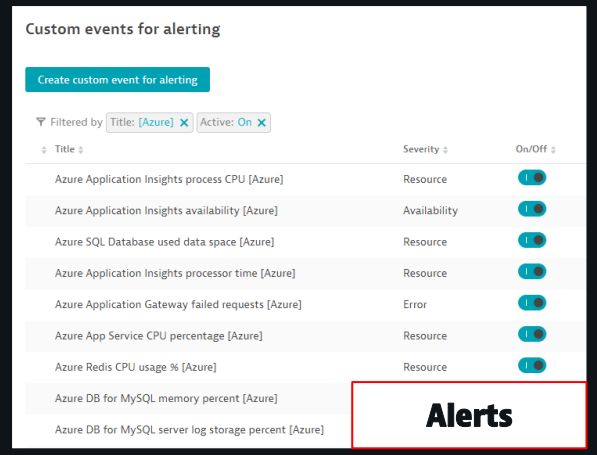
Now expanding automatic intelligent observability of infrastructure monitoring to all GCP services:

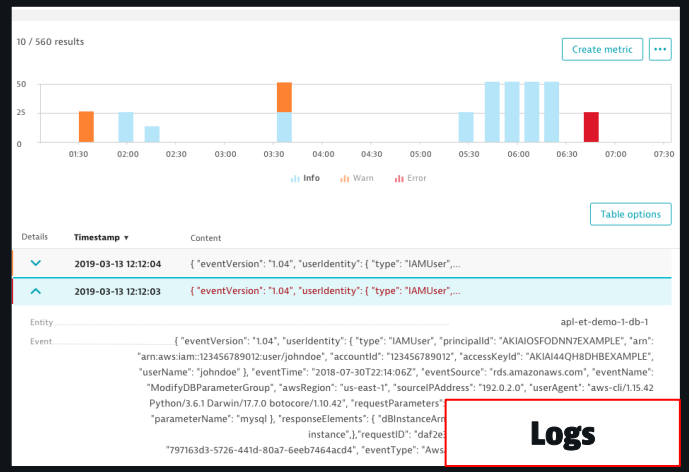


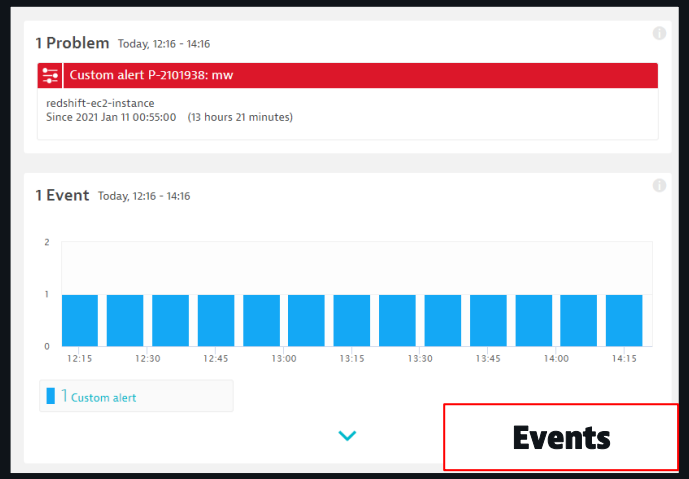
It’s not all about metrics











dynatrace
Perform

Configure ActiveGate for Cloud monitoring

dynatrace
Perform

Edit Custom.Properties File

- Navigate and edit the file: "/var/lib/Dynatrace/gateway/config/custom.properties"
  - vi /var/lib/dynatrace/gateway/config/custom.properties
- Press "i" to edit the file.
- Paste this into the file:

```
[aws_monitoring]
use_aws_proxy_role = false
aws_monitoring_enabled = true
```

- To save changes, press "Esc" button, and then ":wq!" then press "enter"
- Restart the ActiveGate service
  - service dynatracegateway stop
  - service dynatracegateway start

AWS

Exercise 1 - Monitor new AWS account

1. Prepare AWS account access (already done)
2. Configure access from Dynatrace to AWS account

Home > Technology support > Cloud platforms > Amazon Web Services > Set up Dynatrace SaaS for AWS monitoring

Search Dynatrace Help

- + Get started with Dynatrace
- + What's new
- - Technology support
  - - Cloud platforms
    - - Amazon Web Services
      - Set up Dynatrace SaaS for AWS monitoring
      - Set up Dynatrace Managed for AWS monitoring
      - + Integrations
      - - Configuration
        - CORS Amazon S3
        - Connect to Dynatrace using AWS PrivateLink
        - Define host naming rules based on AWS tags
        - Limit API calls to AWS using tags
      - + Supporting services
      - Introduction to AWS monitoring

Set up Dynatrace SaaS for AWS monitoring

You can integrate Dynatrace with Amazon Web Services (AWS) for intelligent monitoring of services running in the Amazon Cloud. AWS integration helps you stay on top of the dynamics of your data center in the cloud.

Important

Dynatrace can be deployed with or without an **Environment ActiveGate**. Configuring role-based access differs for Dynatrace deployments that use an **Environment ActiveGate**.

Overview

Follow these basic steps to integrate Dynatrace SaaS with Amazon Web Services (AWS):

1. Create AWS monitoring policy.
2. Choose an access method:
   - Key-based access.
   - Role-based access.
     If you choose role-based access, use the appropriate procedure for your deployment scenario:

AWS

Use this page to connect your AWS account to Dynatrace for monitoring. Go to your AWS Management Console and prepare access for Dynatrace using either key- or role-based authentication. For details, see Set up AWS monitoring.

Amazon will charge about $0.01 per 1,000 requests to CloudWatch API and include the cost in the bill for the AWS account you use with Dynatrace.

> Metric events for alerting

Name this connection

e.g. Dynatrace integration

Authentication method

Key based authentication

Access Key ID

Access key

Secret access key

Secret key

AWS partition

Default

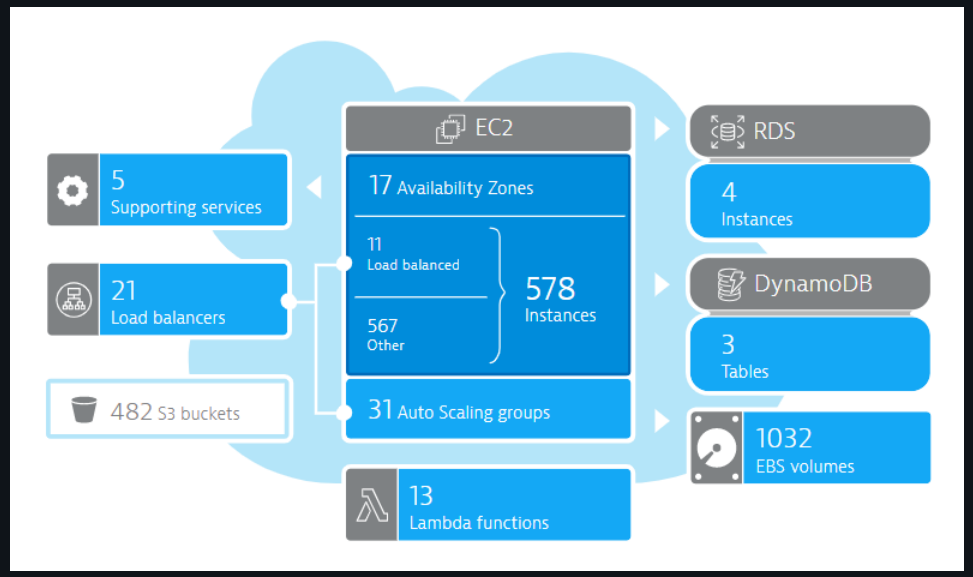
Resources monitoring method

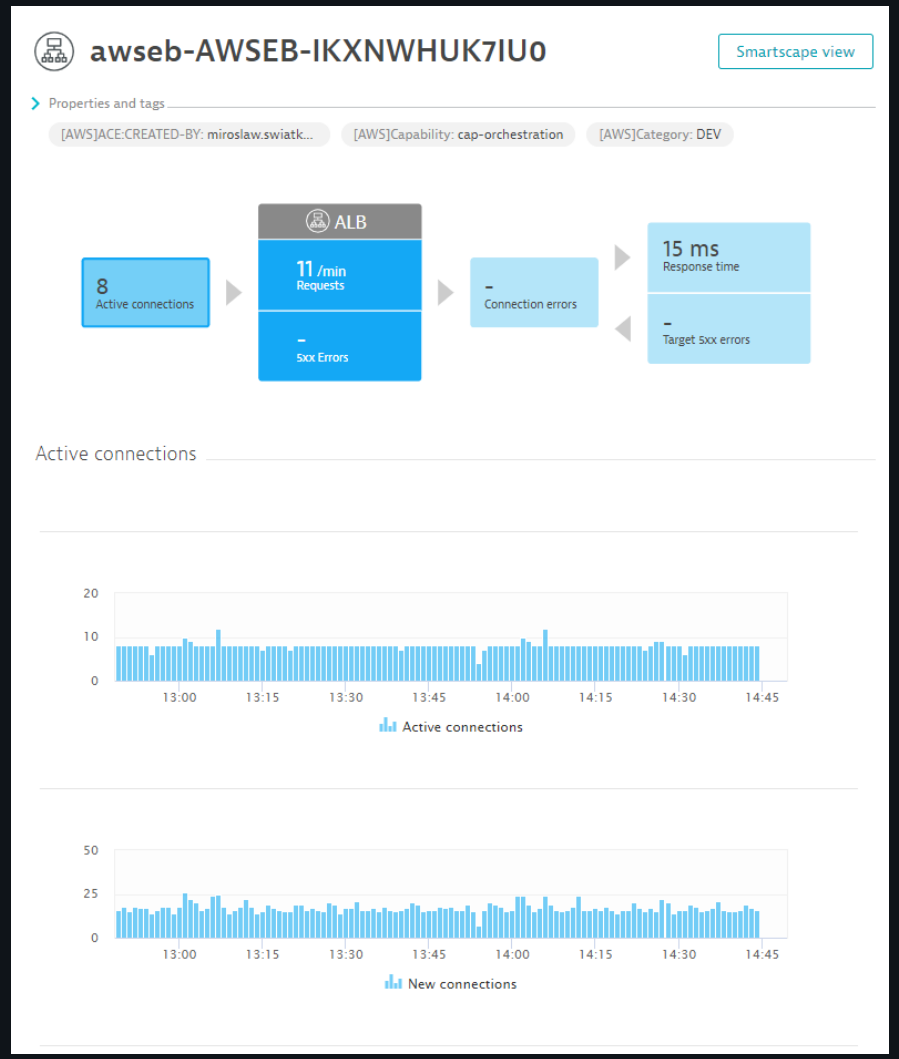
Monitor all resources

Connect | Cancel

Exercise 1 - Monitor new AWS account

3. Observe core services being monitored

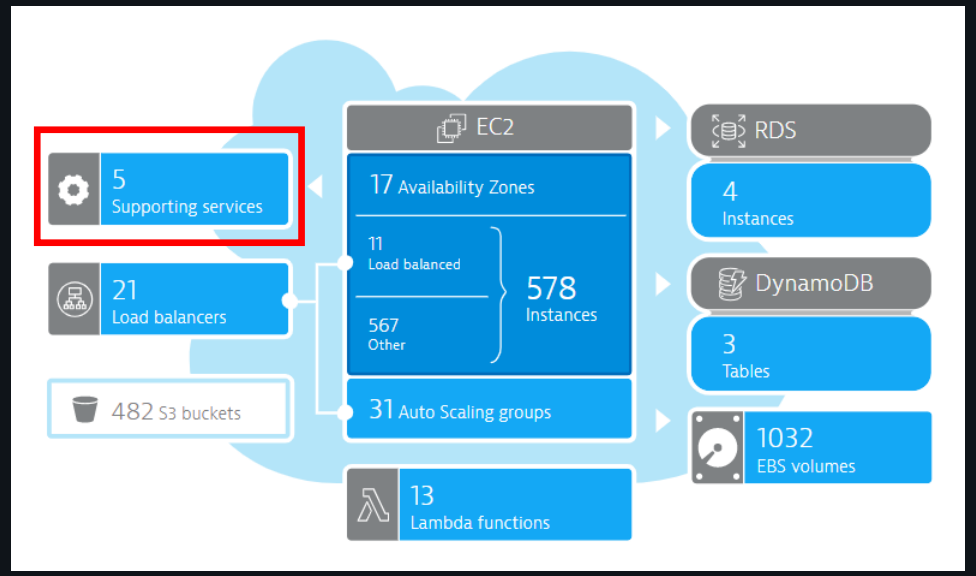


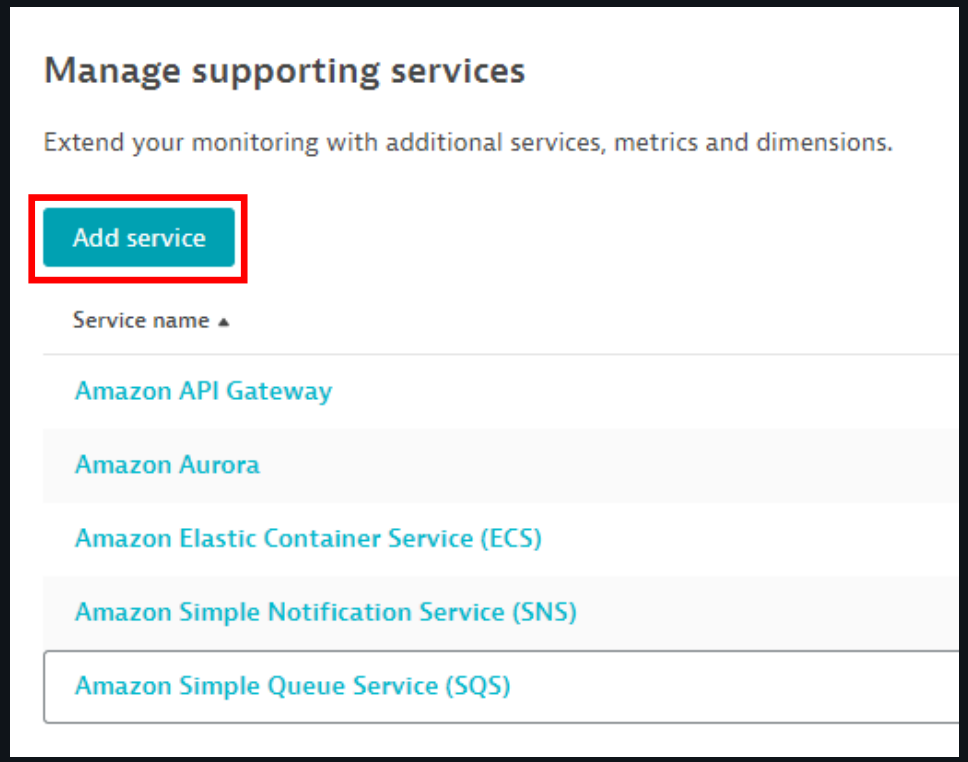


dynatrace
Perform

Exercise 2 - Expand monitoring onto new AWS service

1. From AWS infographic go to supporting services
2. In Configure services choose <service> to be monitored





Exercise 2 - Expand monitoring onto new AWS service

3. Review preselected metrics (edit/add if needed)
4. Wait couple of minutes till the data arrives

Amazon API Gateway

Add metrics from Services that you want to start monitoring with Dynatrace. Added metrics will be segmented by chosen dimensions.

Add metric

Metric name	Aggregation	Unit	Dimensions
4XXError	Sum	Count	ApiName
5XXError	Sum	Count	ApiName
Count	Sum	Count	ApiName, Region, ApiName and Stage
IntegrationLatency	Average + Minimum + Maximum	MilliSecond	ApiName
Latency	Average + Minimum + Maximum	MilliSecond	ApiName, ApiName and Stage

acceptanceWeatherBackend

Smartscape view

Properties and tags

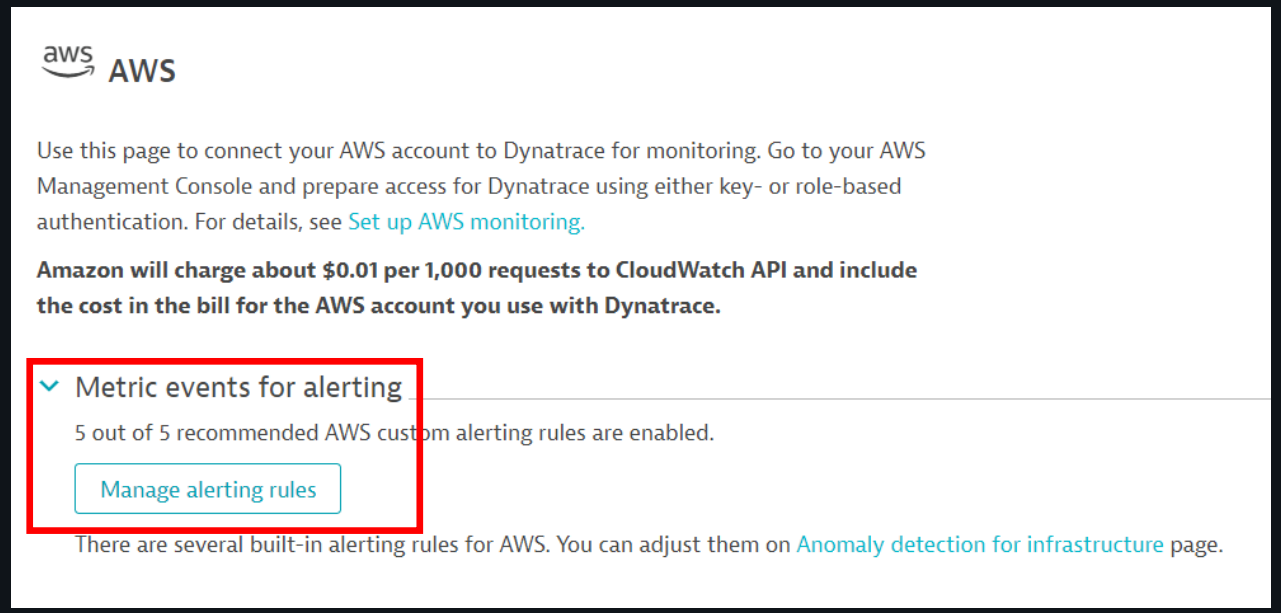
[AWS]Category: DEV [AWS]dynatrace-monitored: true

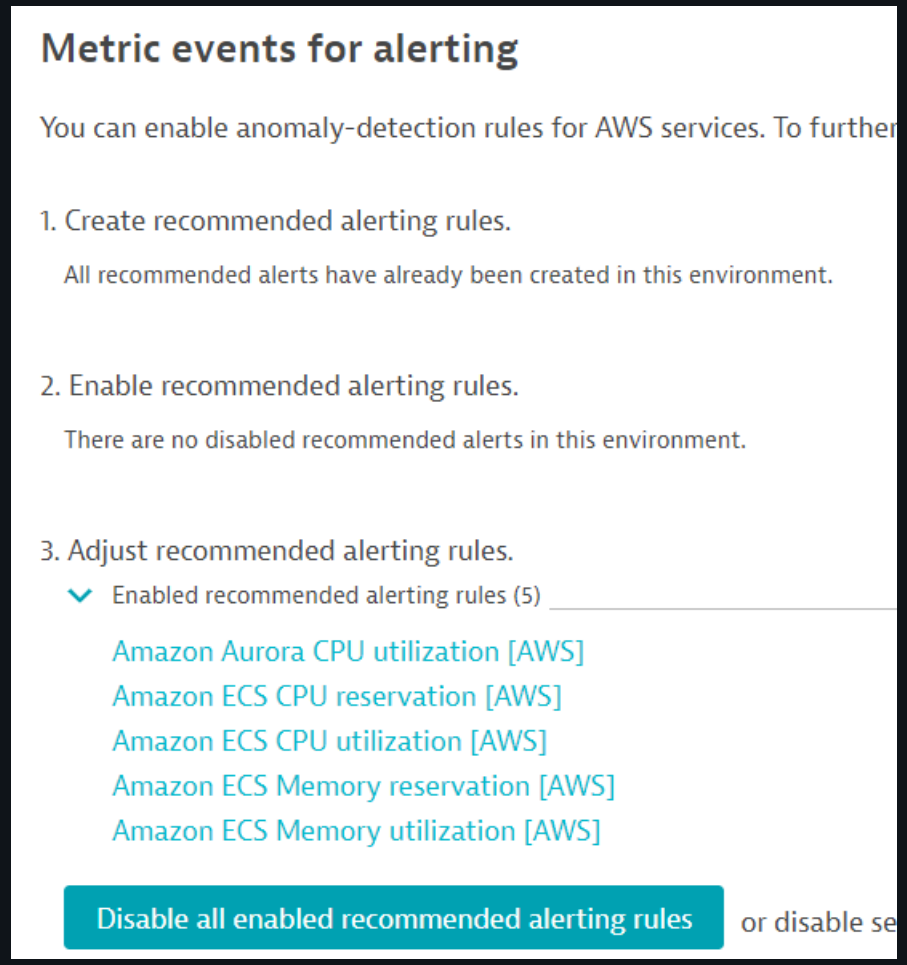
ARN	arn:aws:apigateway:us-east-1::/restapis/4onxlbmoij
ApiId	4onxlbmoij
Region	us-east-1

0 4XXError Sum	0 5XXError Sum
60 Count Sum	179 ms Latency

Exercise 3 - Configure alerts for AWS service

1. In AWS credential settings go to Alert management
2. Review proposed alerting and enable the alerts





Questions?

Key Links & Resources

- Ingestion Protocol
  - https://www.dynatrace.com/support/help/how-to-use-dynatrace/metrics/metric-ingestion/metric-ingestion-protocol/
- DDU Licensing – Metric Consumption
  - https://www.dynatrace.com/support/help/reference/monitoring-consumption-calculation/metric-cost-calculation/
- Metric Browser
  - https://www.dynatrace.com/news/blog/find-and-analyze-important-metrics-faster-with-the-new-metric-browser/
- Metric Explorer
  - https://www.dynatrace.com/support/help/shortlink/explorer
- Telegraf
  - https://www.dynatrace.com/support/help/how-to-use-dynatrace/metrics/metric-ingestion/ingestion-methods/telegraf/
- DynatraceStatsD
  - https://www.dynatrace.com/support/help/how-to-use-dynatrace/metrics/metric-ingestion/ingestion-methods/statsd/
- Dynatrace_ingest scripting integration
  - https://www.dynatrace.com/support/help/how-to-use-dynatrace/metrics/metric-ingestion/ingestion-methods/pipe/
- OneAgent metric API
  - https://www.dynatrace.com/support/help/how-to-use-dynatrace/metrics/metric-ingestion/ingestion-methods/local-api/

Thank you!