Compute the QoS of your infrastructure with DepC

FOSDEM 2019

OVH

Compute the QoS of your infrastructure with DepC

Nicolas CROCFER

@ncrocfer

/nicolascrocfer

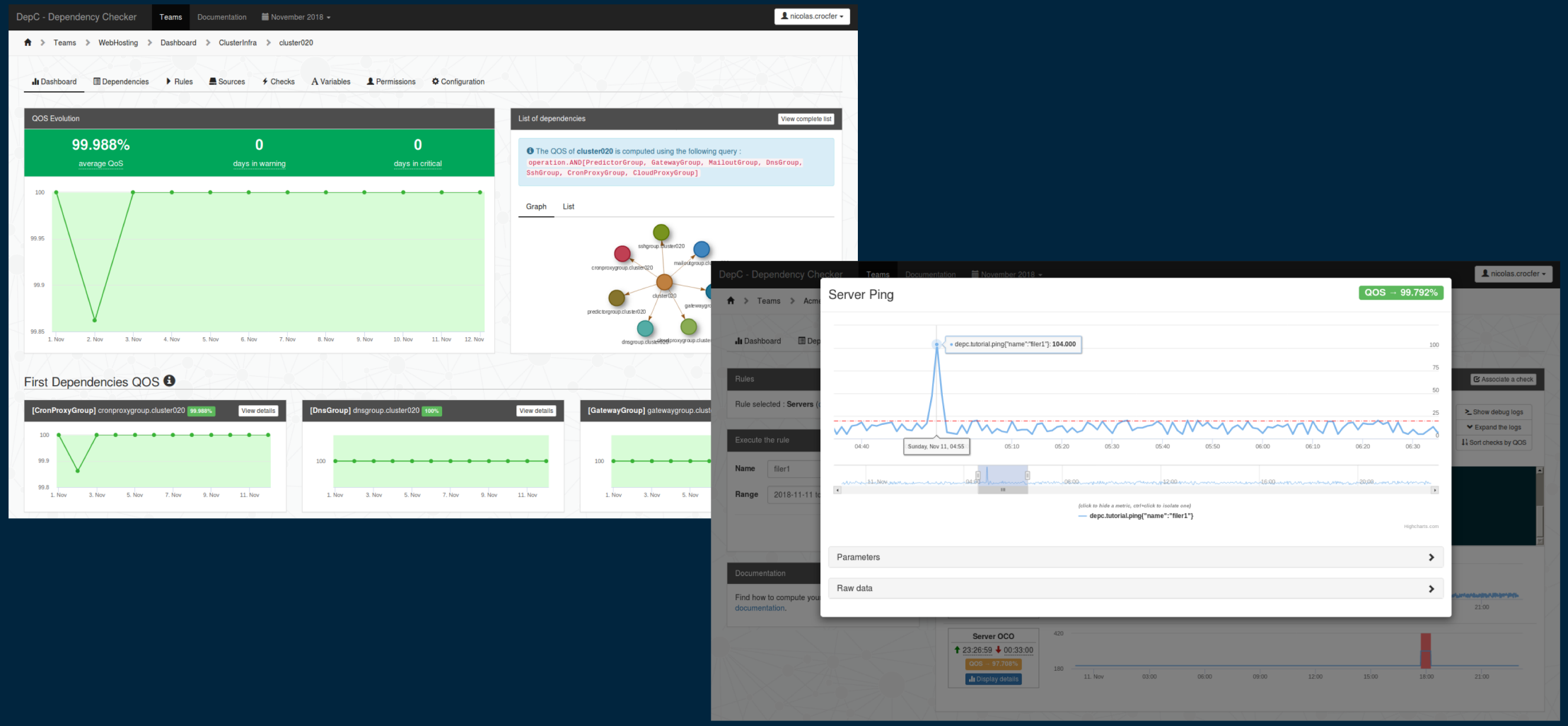
Anthony OLEA

@OleaAnthony

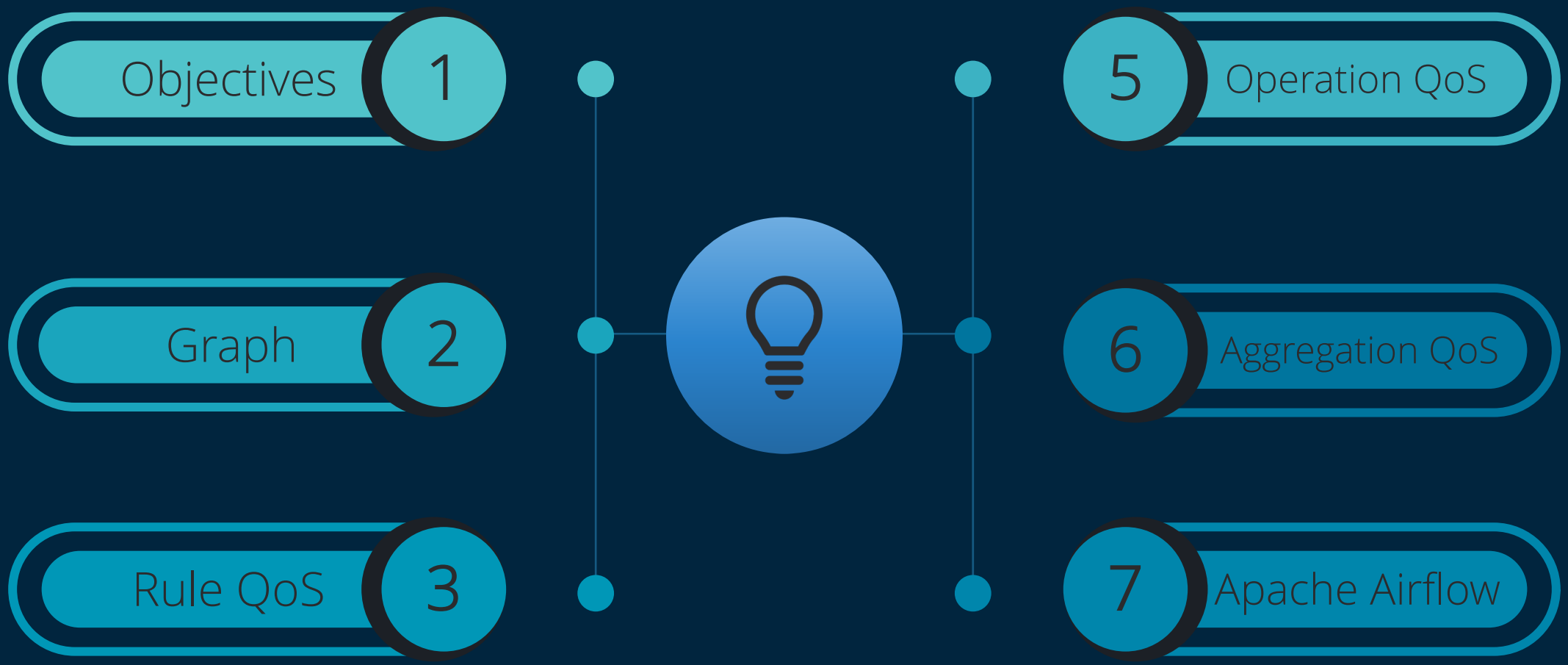
/anthonyolea

OVH

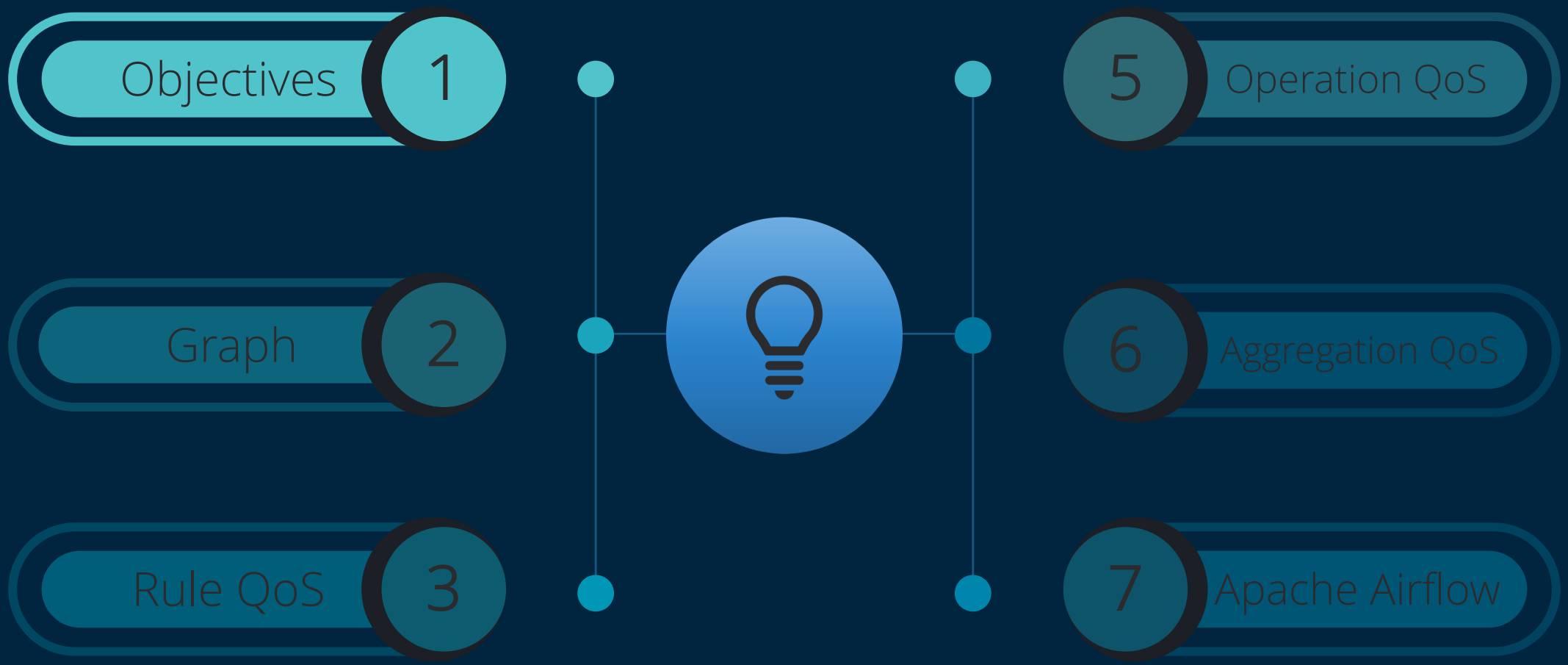
github.com/ovh/depc



Compute the QoS of your infrastructure with DepC



Compute the QoS of your infrastructure with DepC



OBJECTIVES

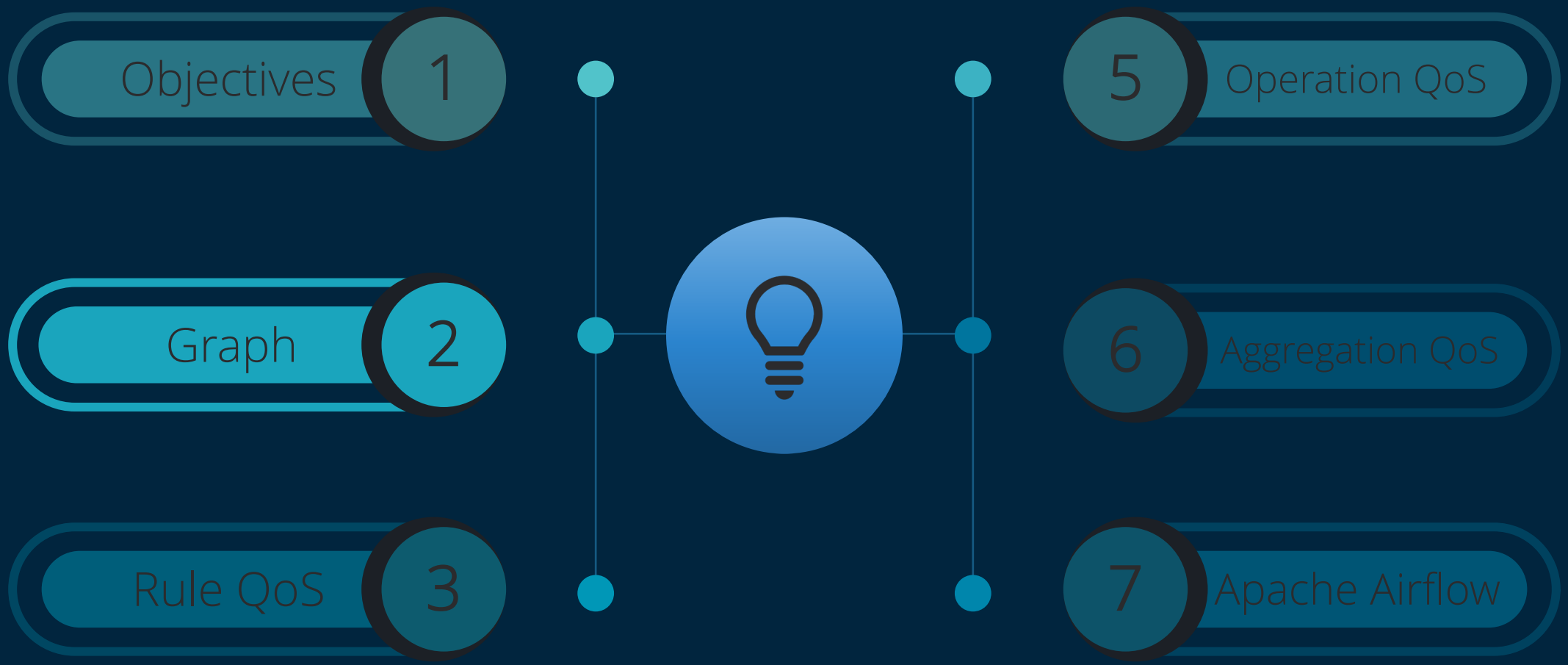
- **5M** websites distributed between **14 000** servers
- ... sometimes a server crashes, impacting some users ☹

Compute the QoS of **every** websites

 *In case of bad QoS, find the **root causes** !* 

OVH

Compute the QoS of your infrastructure with DepC



GRAPH

```
$ curl -s -o /dev/null -I -w "%{http_code}" -LI example.org
200
```

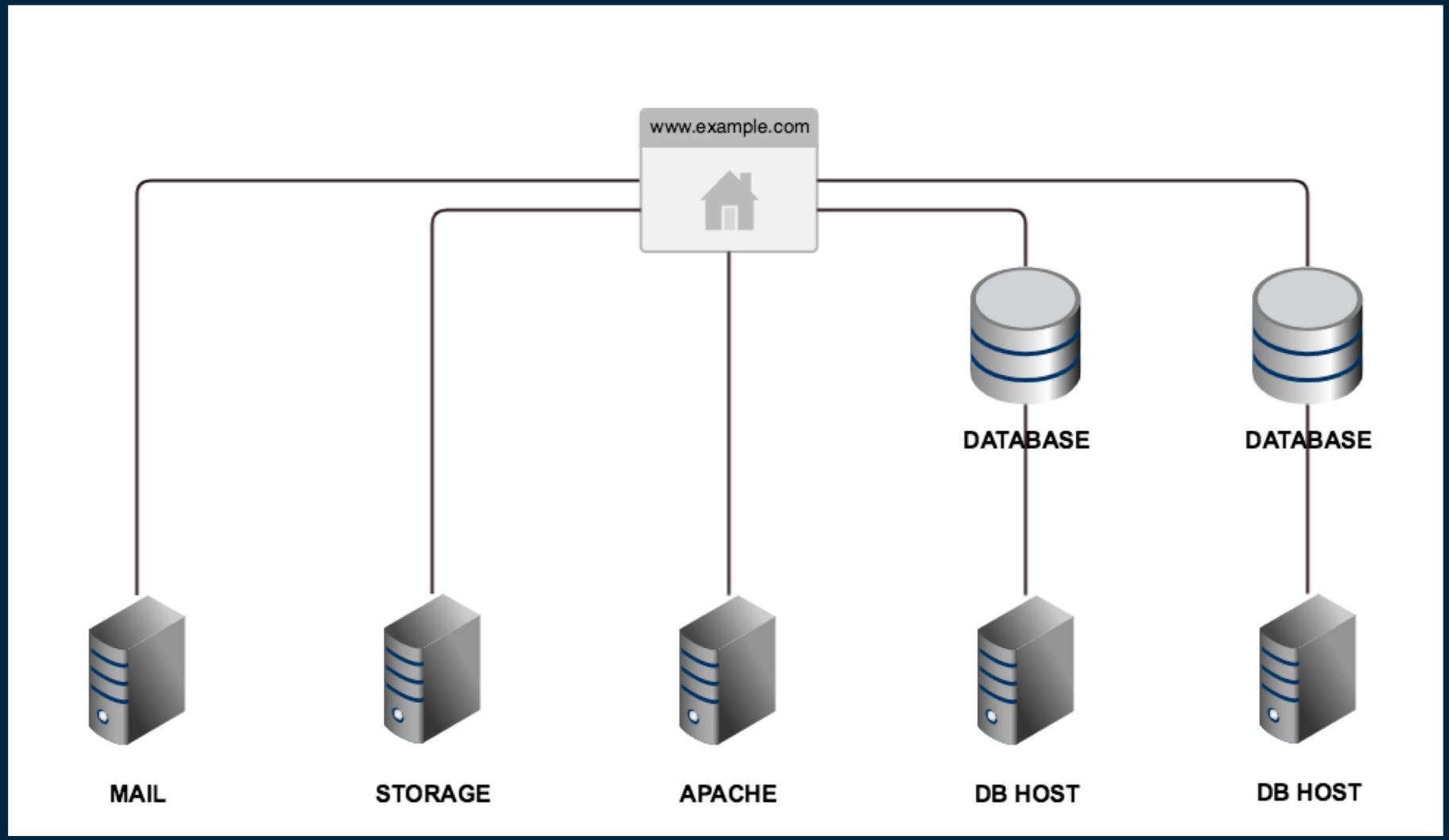
- Scaling complicated
- HTTP status code is not guaranted
- Not possible to find the root causes

GRAPH

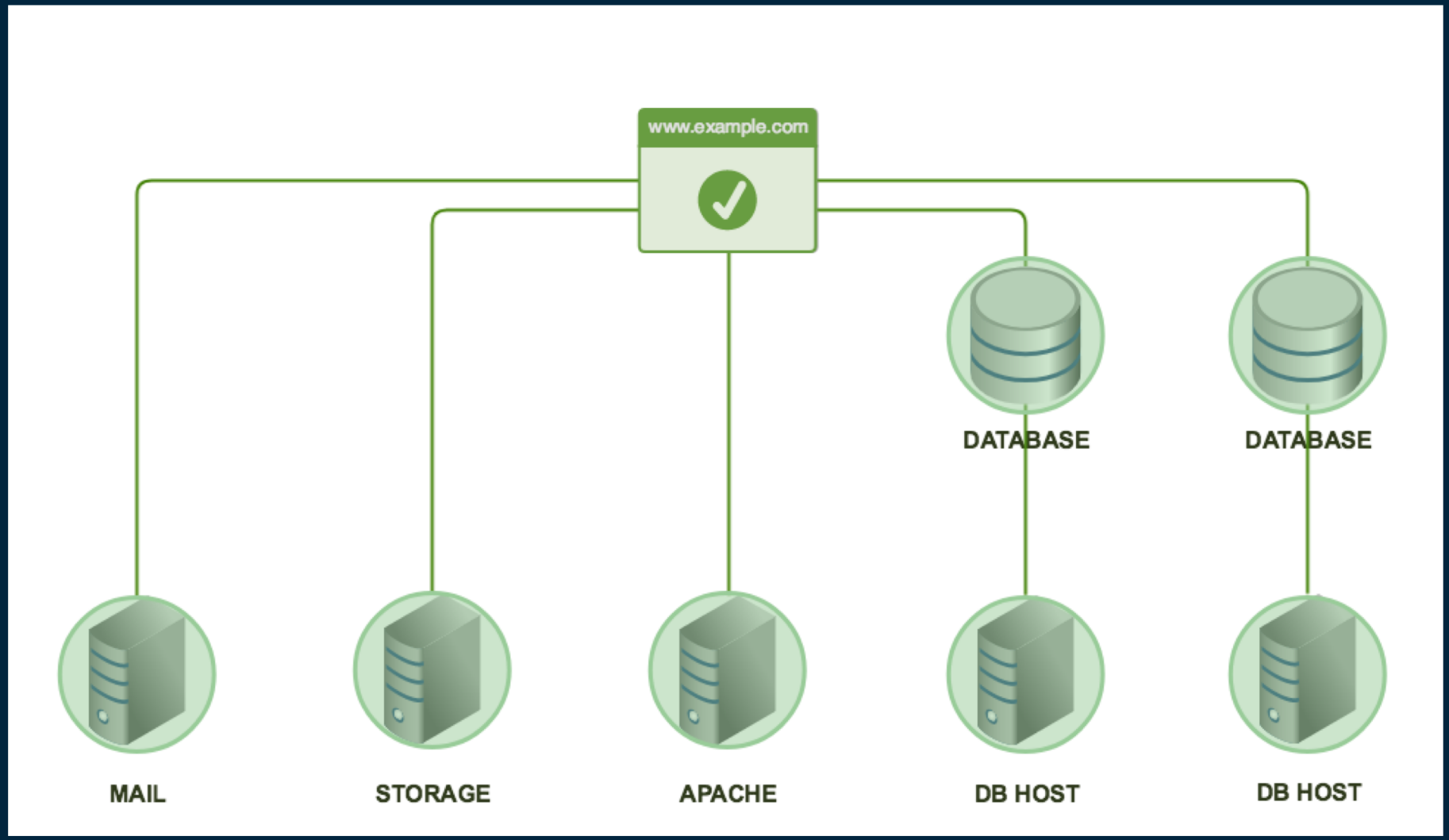
- So how to find the root causes ?

Using a Dependency Graph 

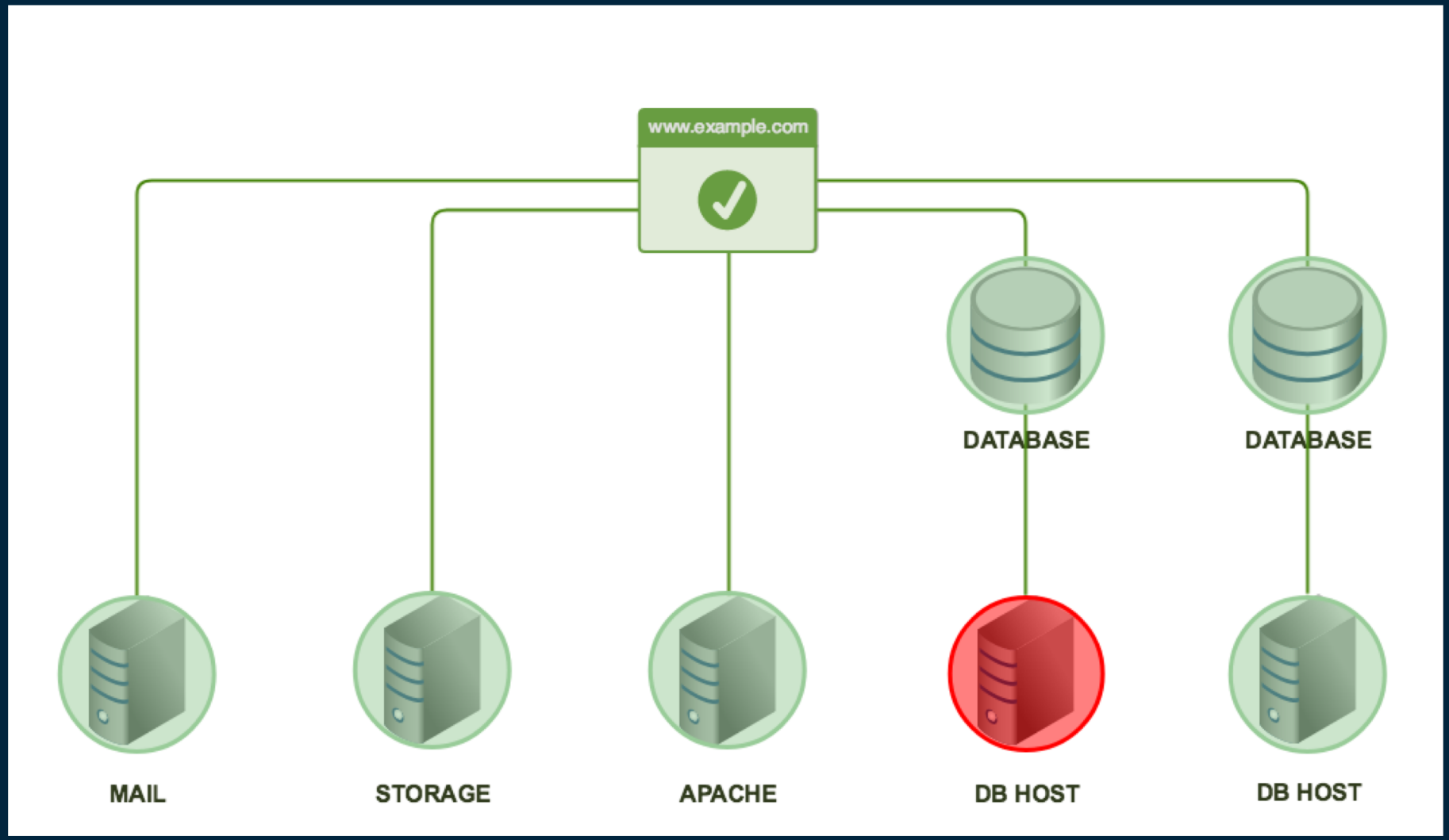
GRAPH


www.example.com
DATABASE
DATABASE
MAIL
STORAGE
APACHE
DB HOST
DB HOST

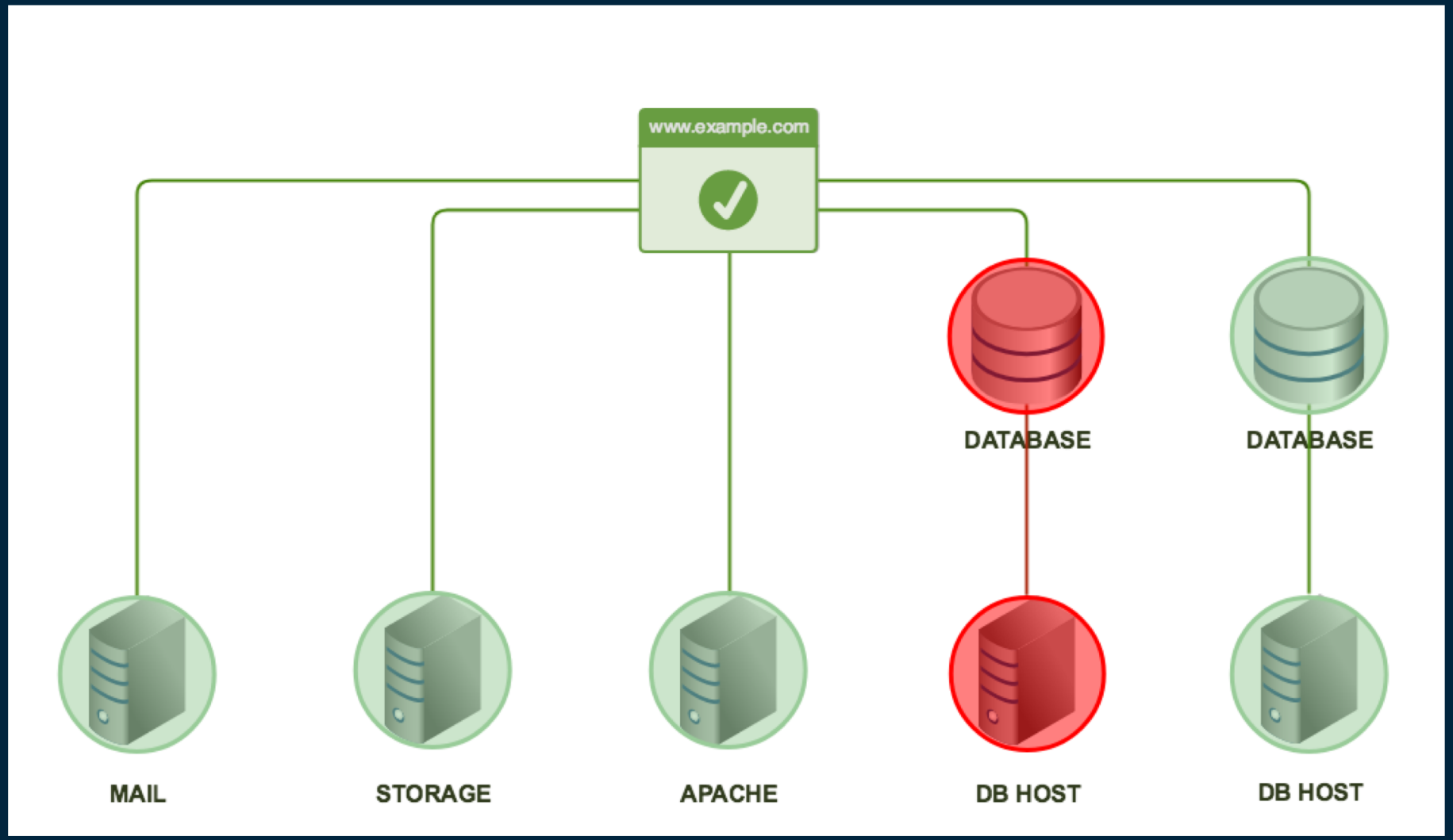
GRAPH


www.example.com
DATABASE
DATABASE
MAIL
STORAGE
APACHE
DB HOST
DB HOST

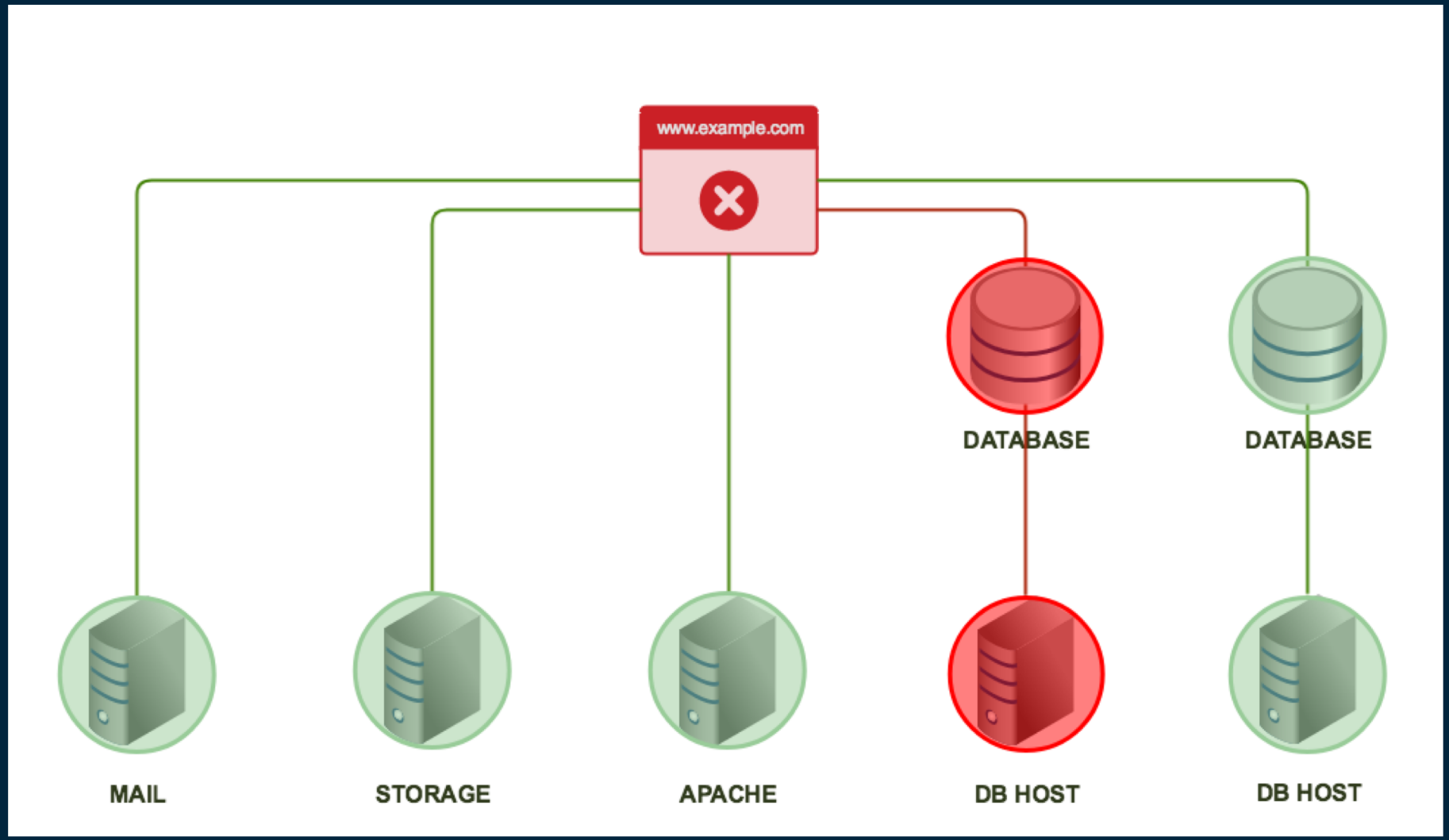
GRAPH


www.example.com
DATABASE
DATABASE
MAIL
STORAGE
APACHE
DB HOST
DB HOST

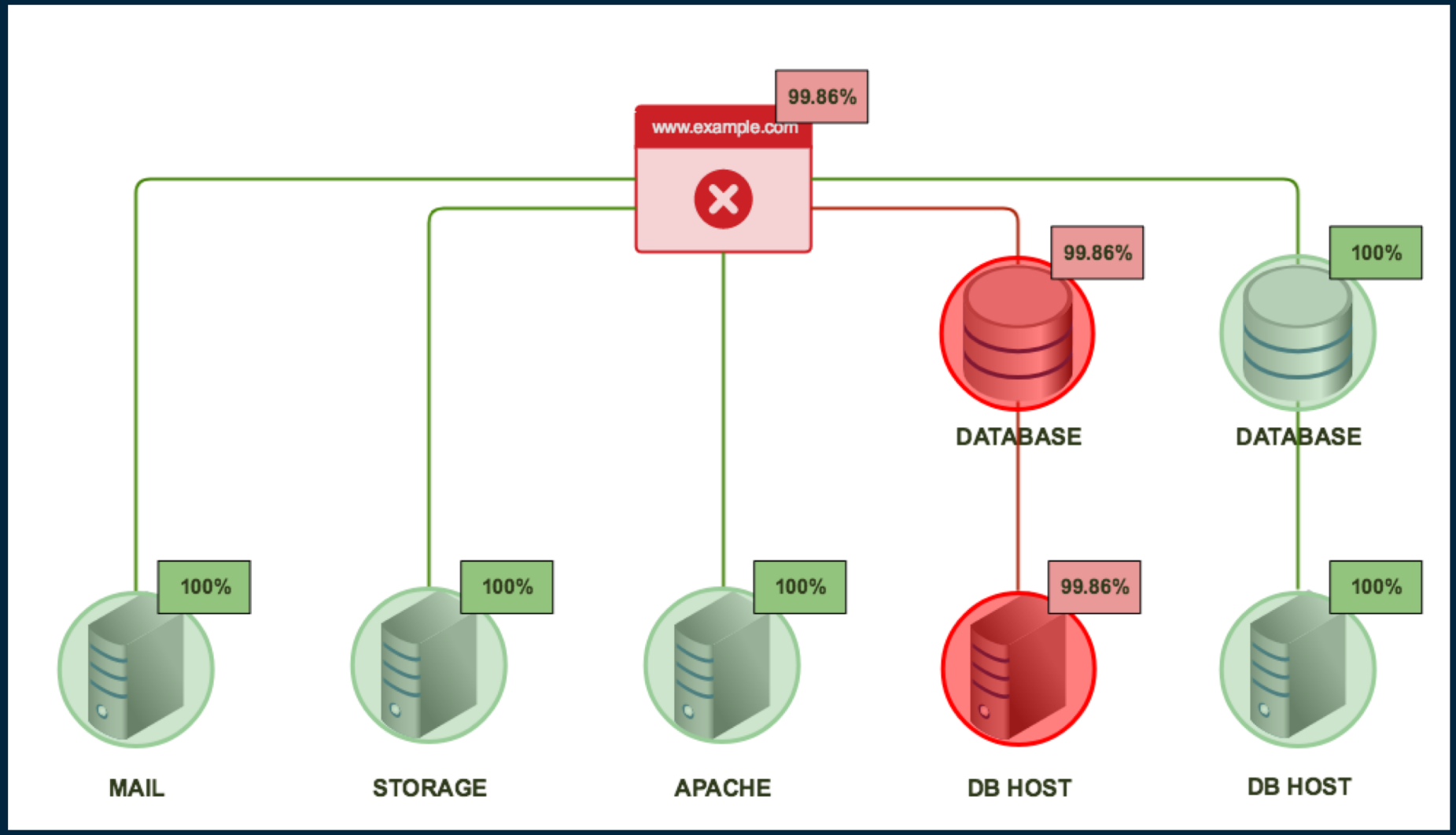
GRAPH


www.example.com
DATABASE
DATABASE
MAIL
STORAGE
APACHE
DB HOST
DB HOST

GRAPH


www.example.com
DATABASE
DATABASE
MAIL
STORAGE
APACHE
DB HOST
DB HOST

GRAPH


99.86%
www.example.com
99.86%
DATABASE
100%
DATABASE
100%
MAIL
100%
STORAGE
100%
APACHE
99.86%
DB HOST
100%
DB HOST

GRAPH

kafka

Validate

Reformat

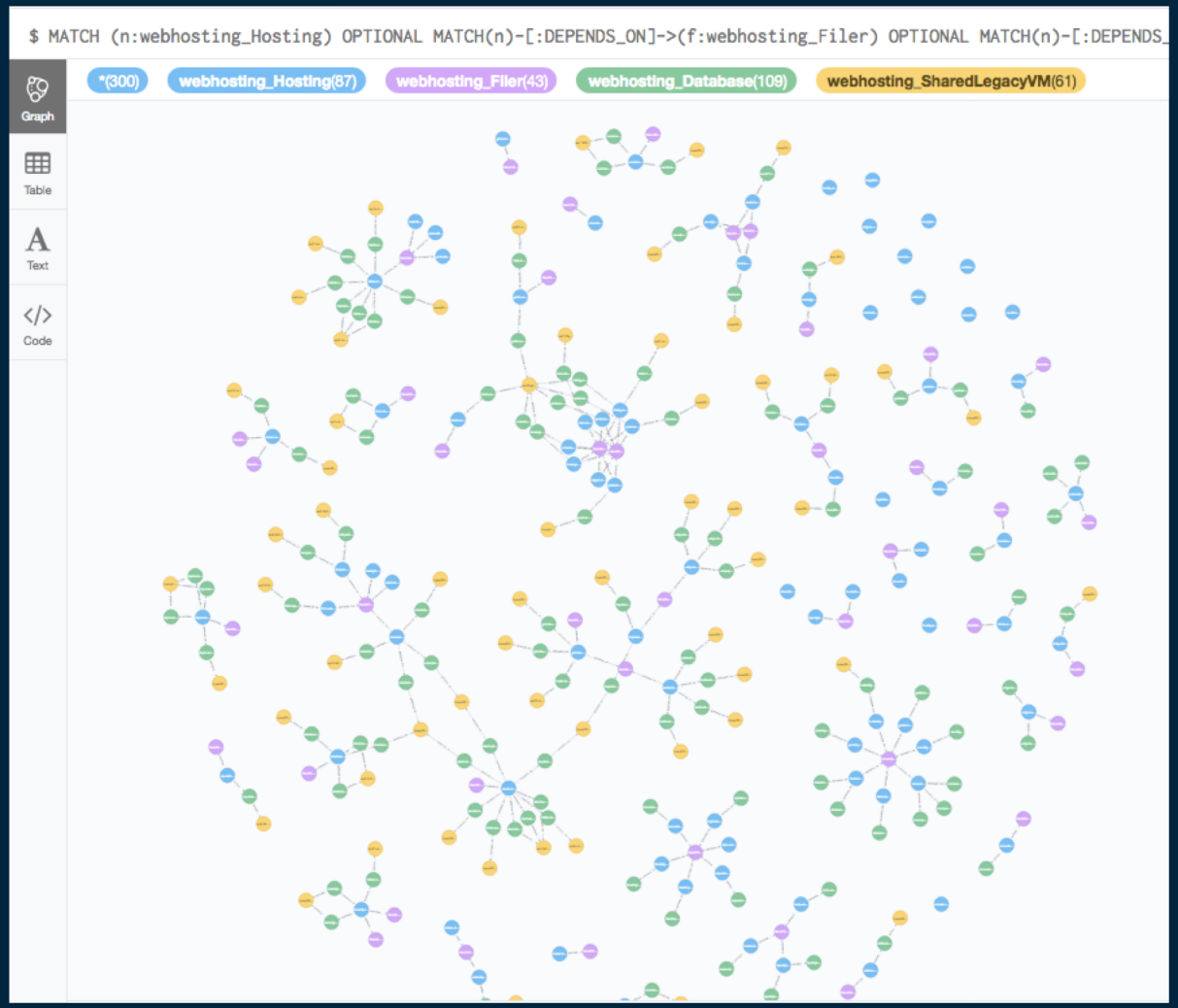
Server

Cluster

neo4j

OVH

GRAPH

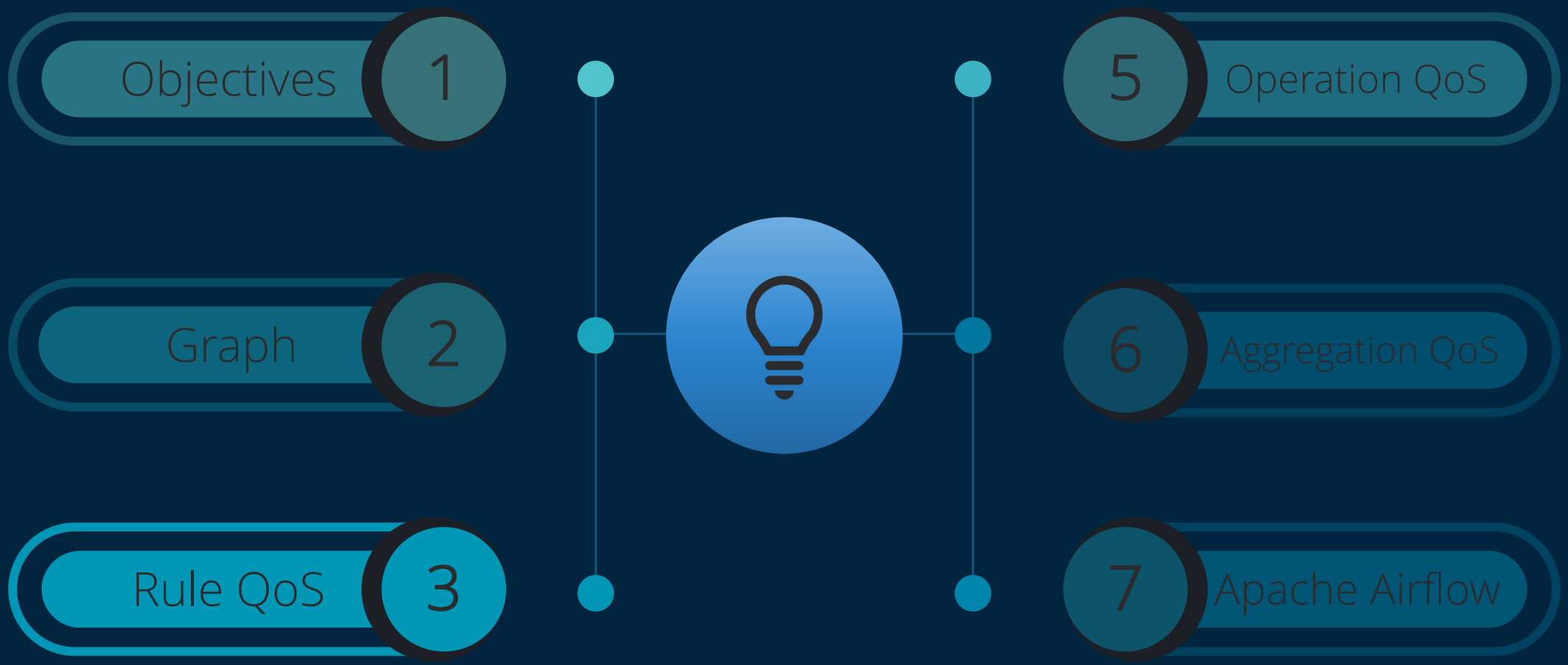


Internal DepC :

- 9M nodes
- 15M relationships
- 24 labels

  (categories of node)

Compute the QoS of your infrastructure with DepC



So we have nodes in a graph,

but how to compute their QoS ?

QOS

Nodes can be monitored : we analyse **TimeSeries**

→ *Rule QoS*

OPENTSDB

Nodes can't be monitored : we use its parent's QoS

→ *Operation QoS*

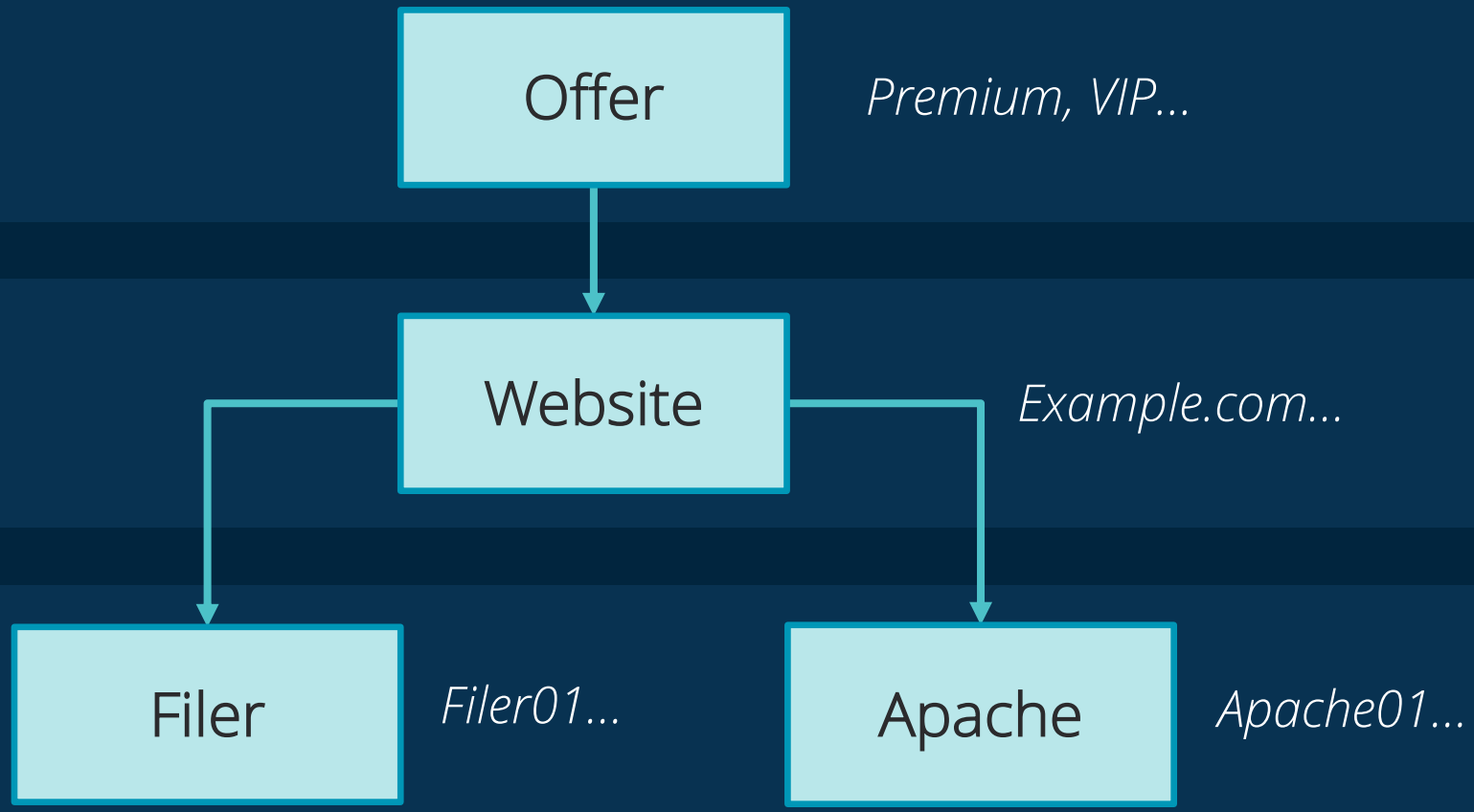
→ *Aggregation QoS*

QOS

Aggregation QoS

Operation QoS

Rule QoS

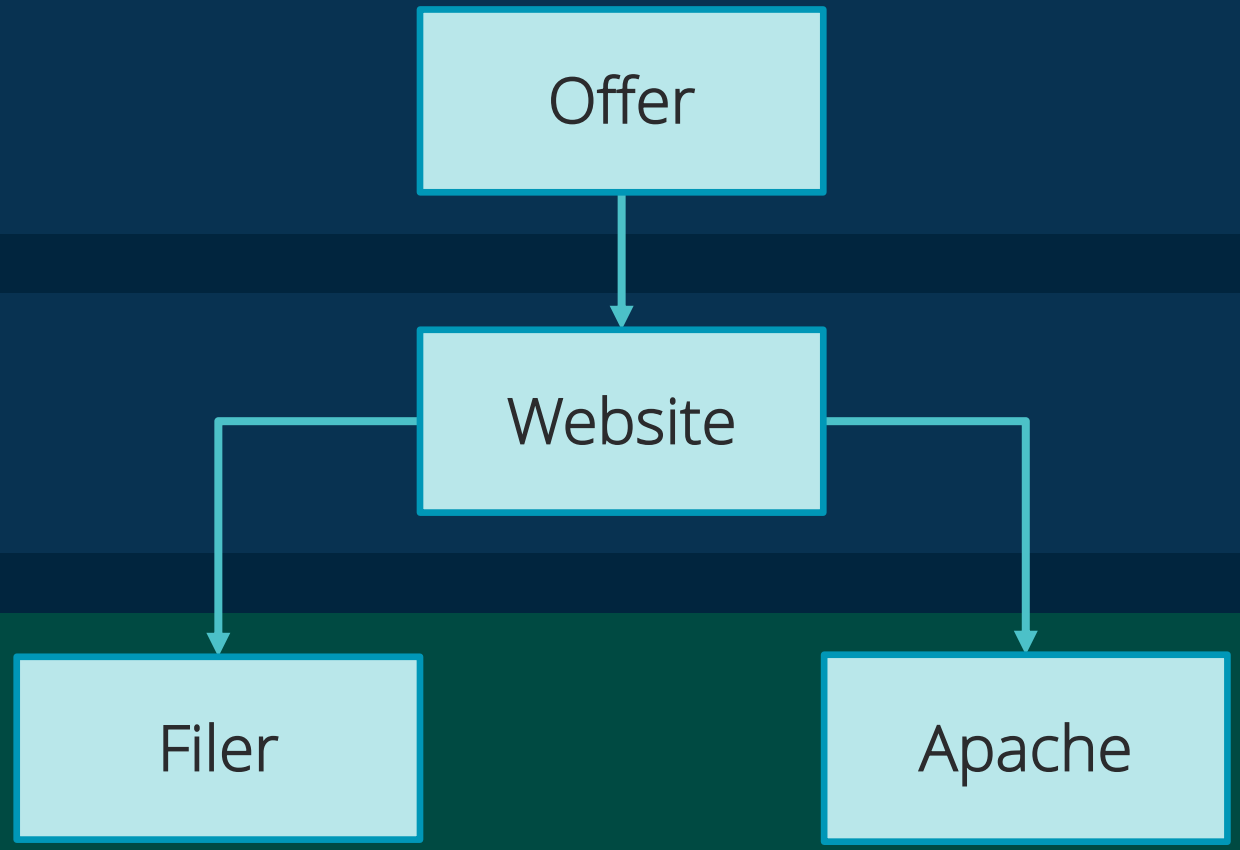


QOS

Aggregation QoS

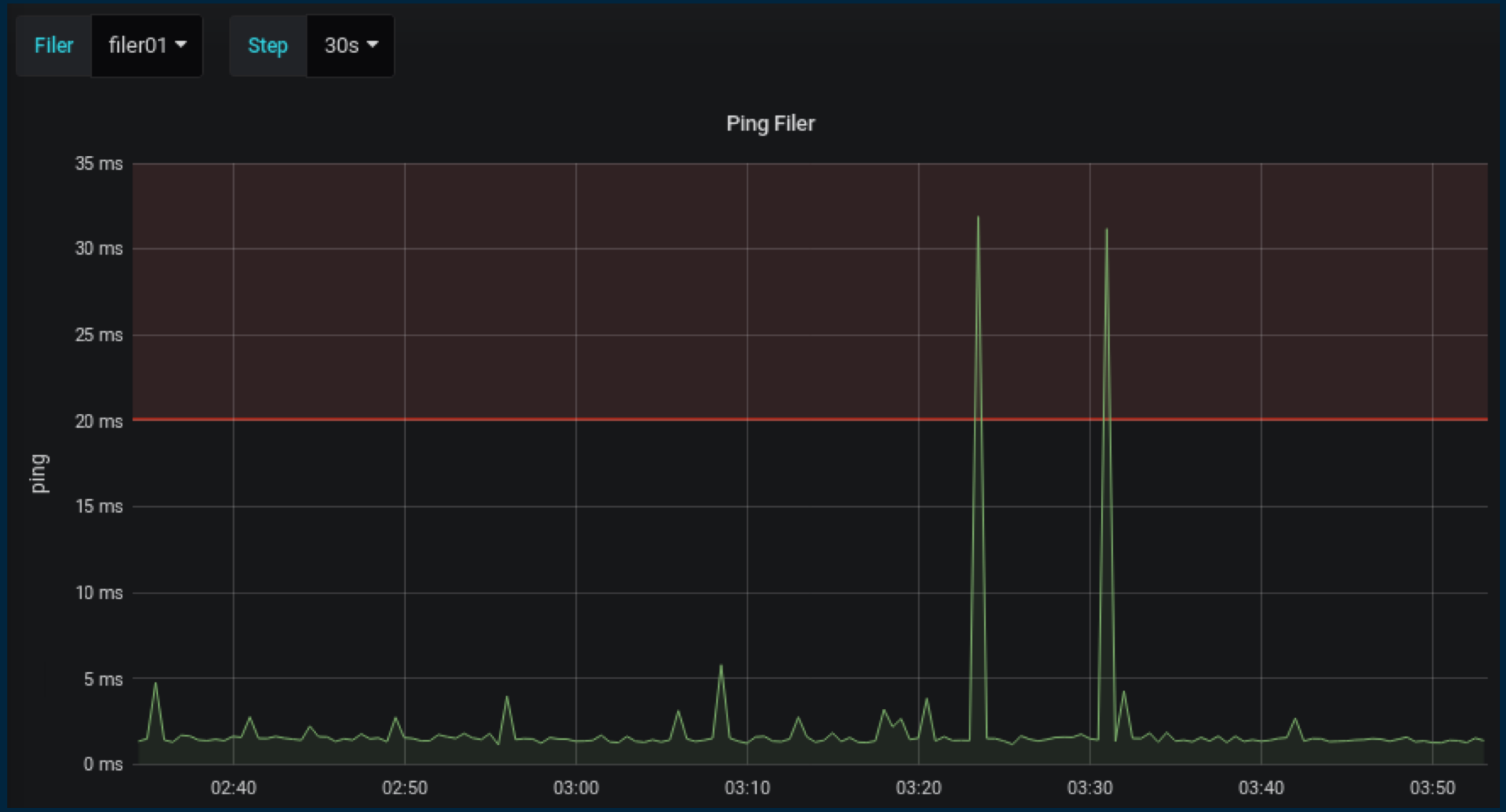
Operation QoS

Rule QoS



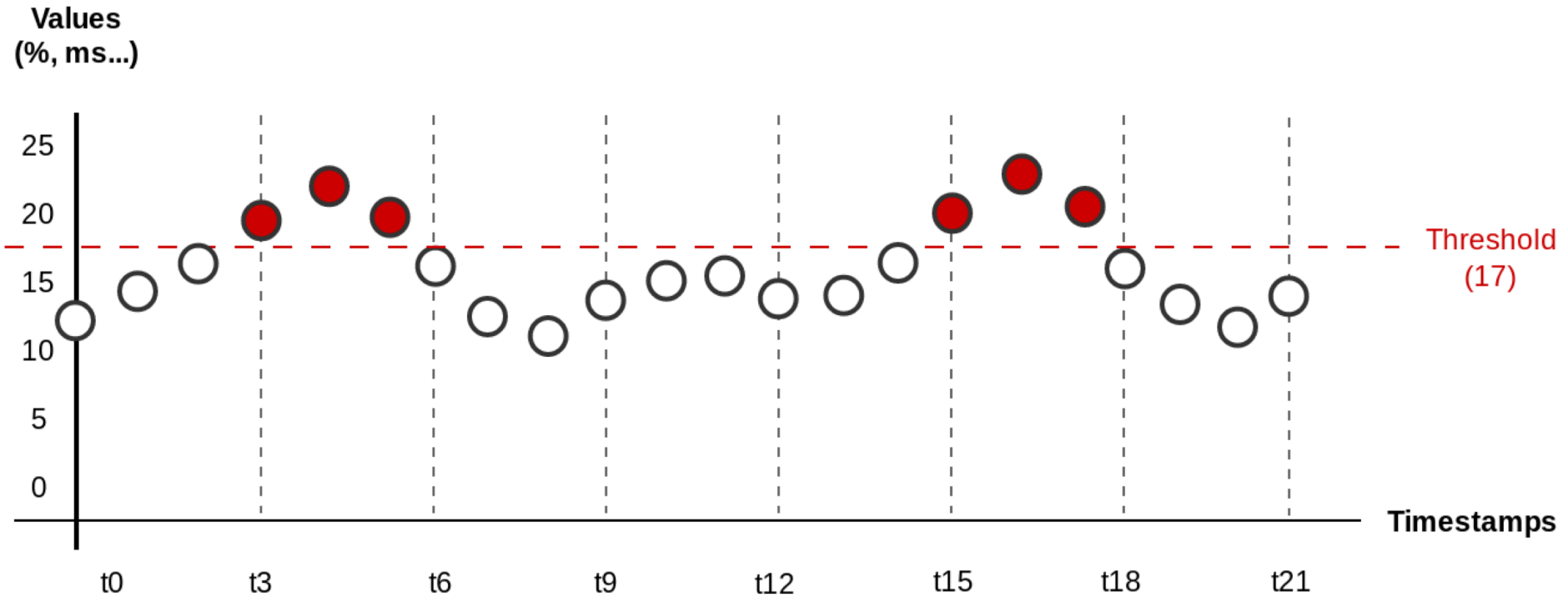
OVH

Source : https://play.grafana.org

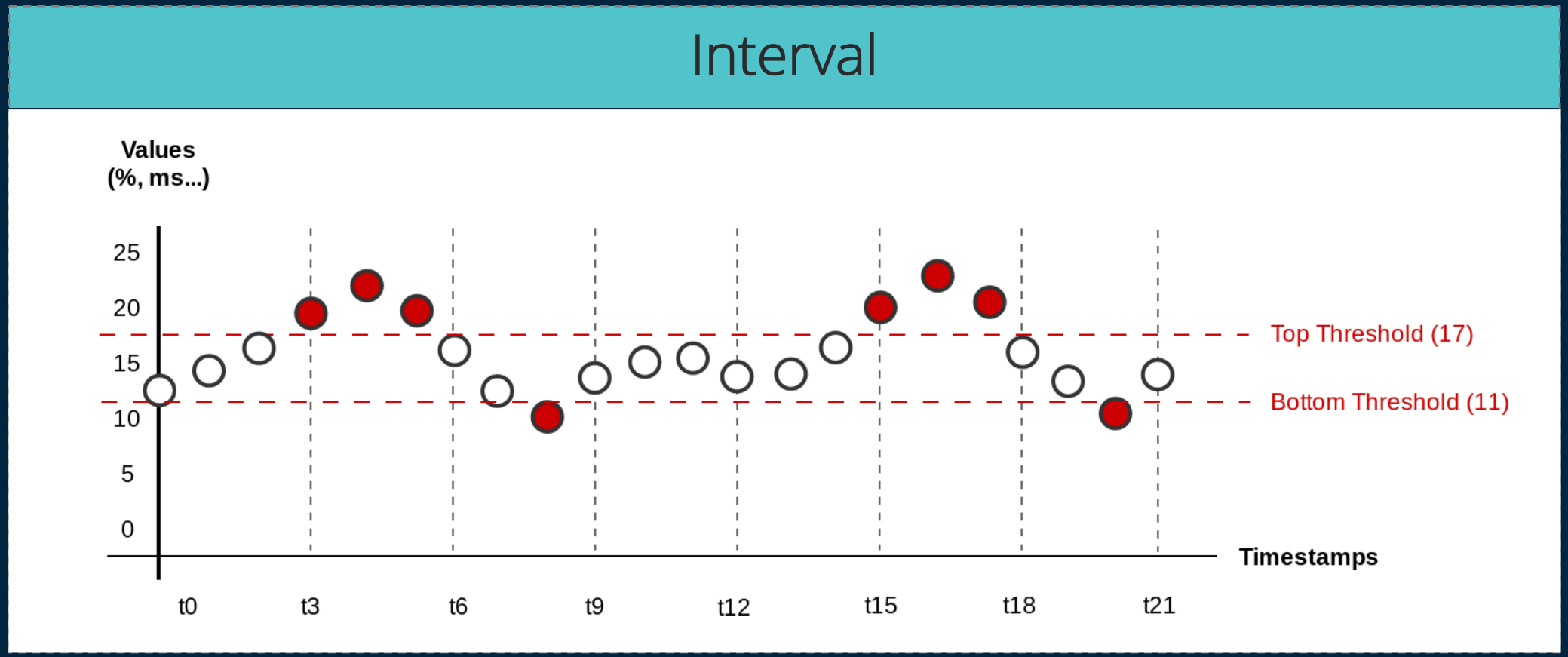

Filer
filer01
Step
30s
Ping Filer
35 ms
30 ms
25 ms
20 ms
15 ms
10 ms
5 ms
0 ms
ping
02:40
02:50
03:00
03:10
03:20
03:30
03:40
03:50

RULE QOS

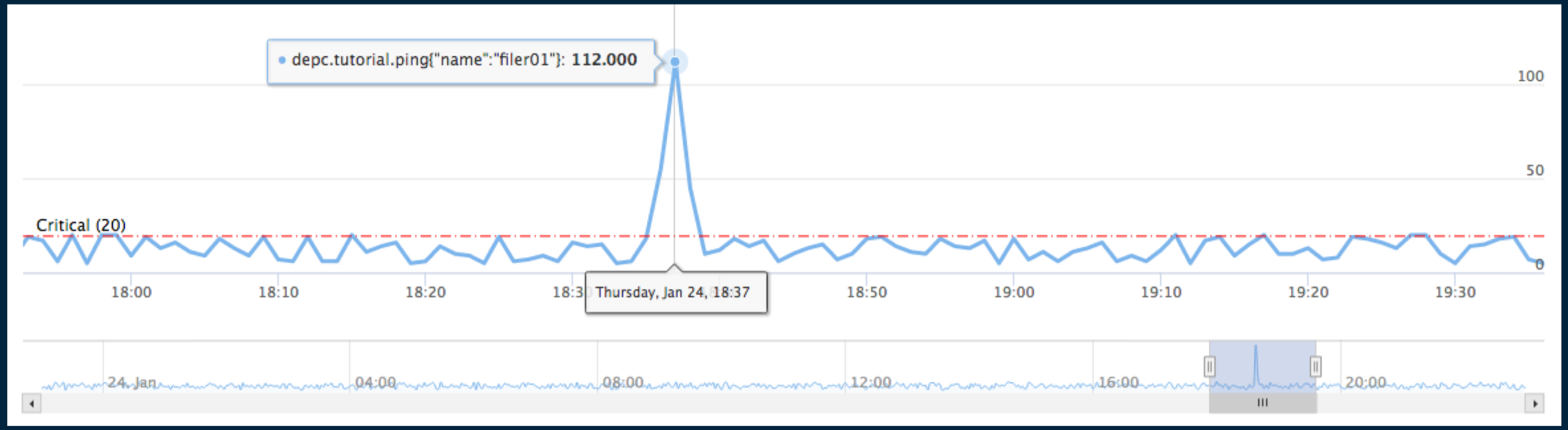
Threshold



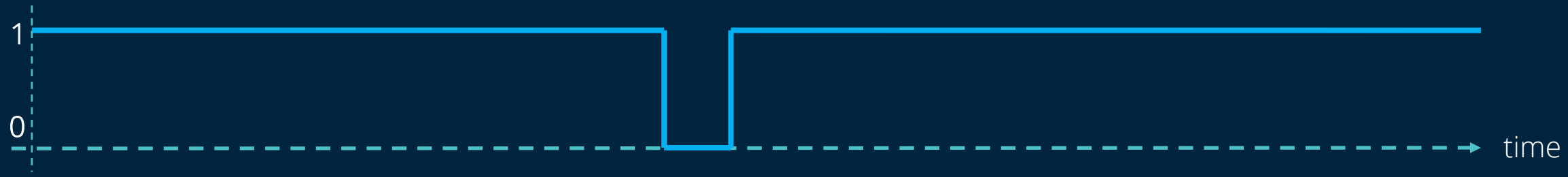
RULE QOS


Interval
Values
(%, ms...)
25
20
15
10
5
0
Top Threshold (17)
Bottom Threshold (11)
Timestamps
t0
t3
t6
t9
t12
t15
t18
t21

RULE QOS



Converting values into booleans



OVH

RULE QOS

Timestamp	Ping result
Ts_0	18
Ts_1	15
Ts_2	16
Ts_3	19
Ts_4	98
Ts_5	112
Ts_6	19
Ts-7	18
…	…
T_x+n	16

Timestamp	Ping result
Ts_0	True
Ts_1	True
Ts_2	True
Ts_3	True
Ts_4	False
Ts_5	False
Ts_6	True
Ts-7	True
…	…
T_x+n	True

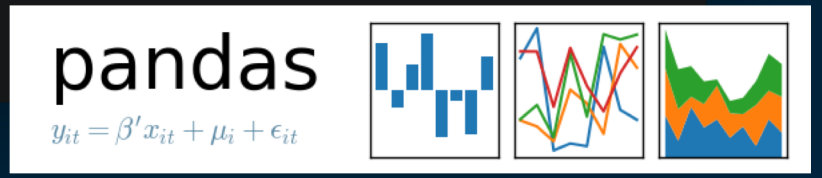
RULE QOS

```
In [1]: import pandas as pd

In [2]: dps = {
   ...:     "1548284400": 18,
   ...:     "1548284460": 15,
   ...:     "1548284520": 16,
   ...:     "1548284580": 19,
   ...:     "1548284640": 98,
   ...:     "1548284700": 112,
   ...:     "1548284760": 19,
   ...:     "1548284820": 18,
   ...:     #"...": "...",
   ...:     "1548370740": 16
   ...: }
```

```
In [3]: serie = pd.Series(dps).apply(lambda x: x <= 20)

In [4]: serie
Out[4]:
1548284400     True
1548284460     True
1548284520     True
1548284580     True
1548284640    False
1548284700    False
1548284760     True
1548284820     True
1548370740     True
dtype: bool
```



OVH

Rule Servers (1 checks)

Associate a check

99.792%

Show debug logs

Expand the logs

Sort checks by QOS

```
                                    {'Server Ping': {'metric': 'depc.tutorial.ping', 'threshold': 20}}}...
1/24/19 4:13:26 PM          INFO    [Servers] 1 checks to execute
1/24/19 4:13:26 PM          INFO    [Server Ping] Executing check (86ee5bdb-7088-400c-8654-a162f18b0710)...
1/24/19 4:13:26 PM          INFO    [Server Ping] Check returned 99.792%
1/24/19 4:13:26 PM          INFO    [Servers] Rule done
1/24/19 4:13:26 PM          INFO    [Servers] Rule QOS is 99.79200%
```

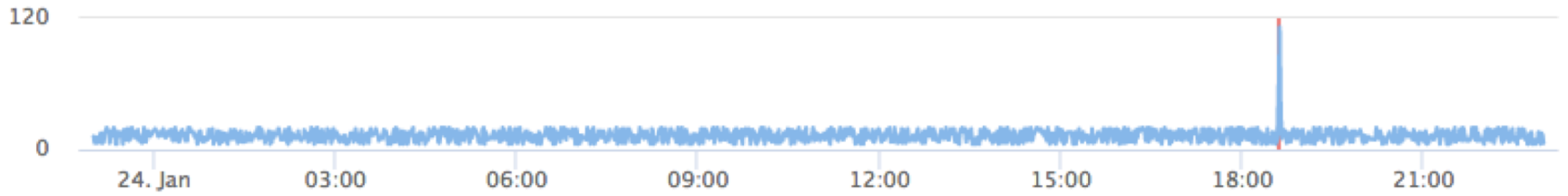
Server Ping

23:56:59

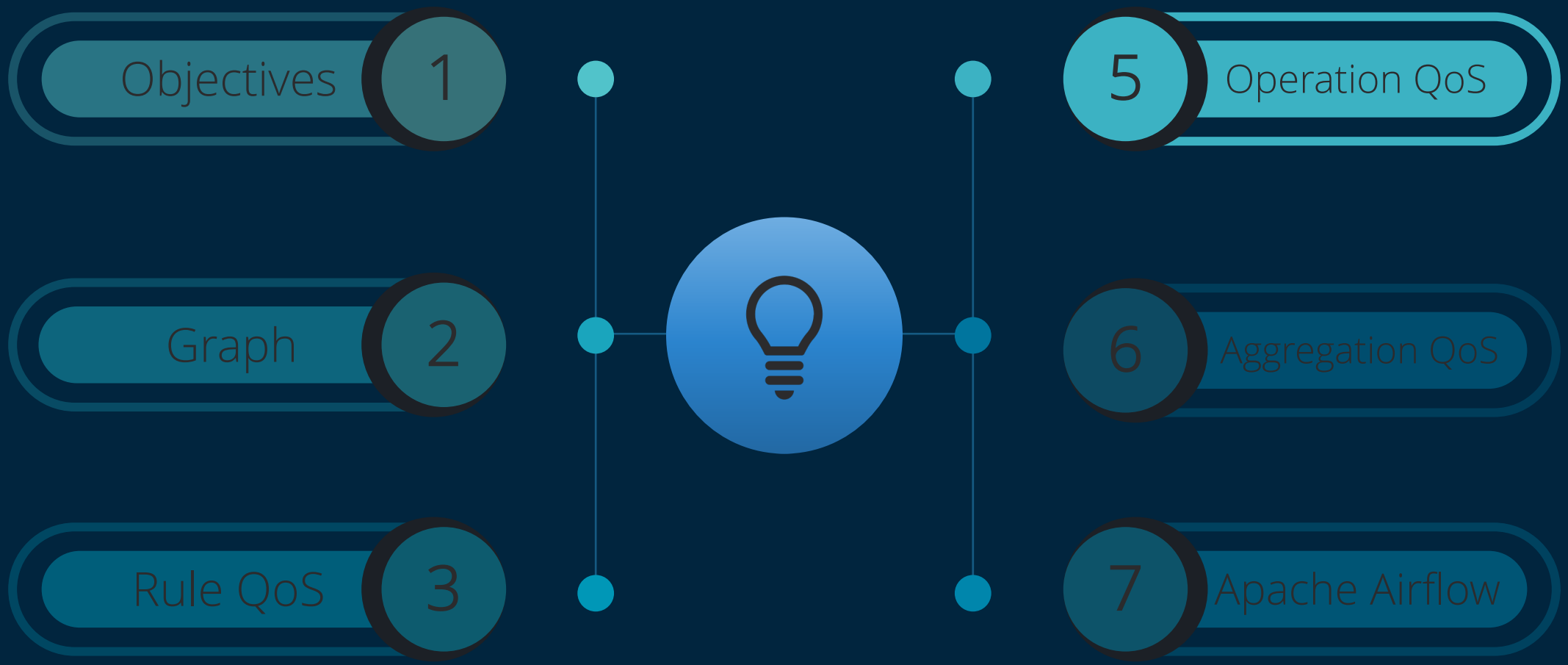
00:03:00

QOS → 99.792%

Display details



Compute the QoS of your infrastructure with DepC

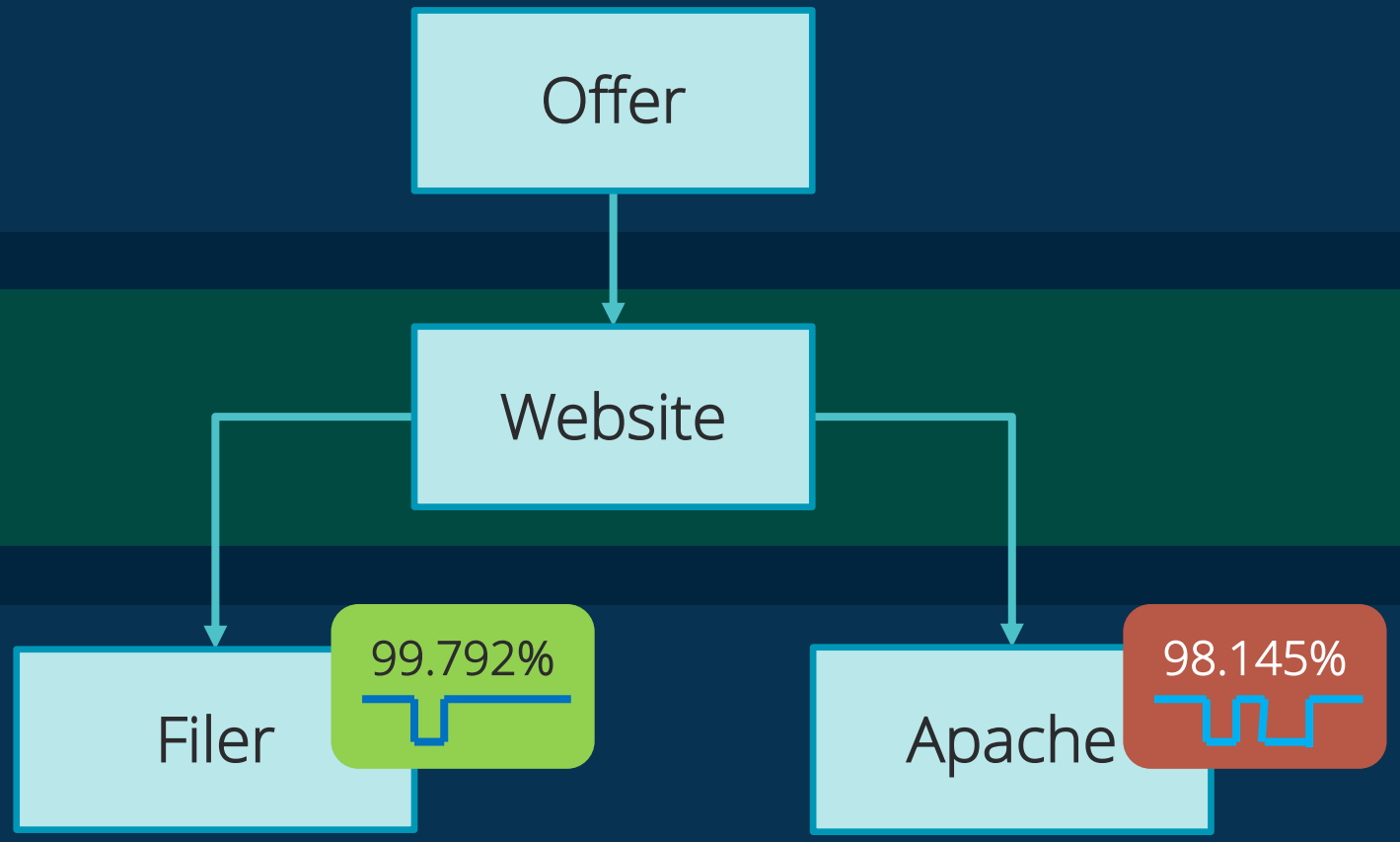


QOS

Aggregation QoS

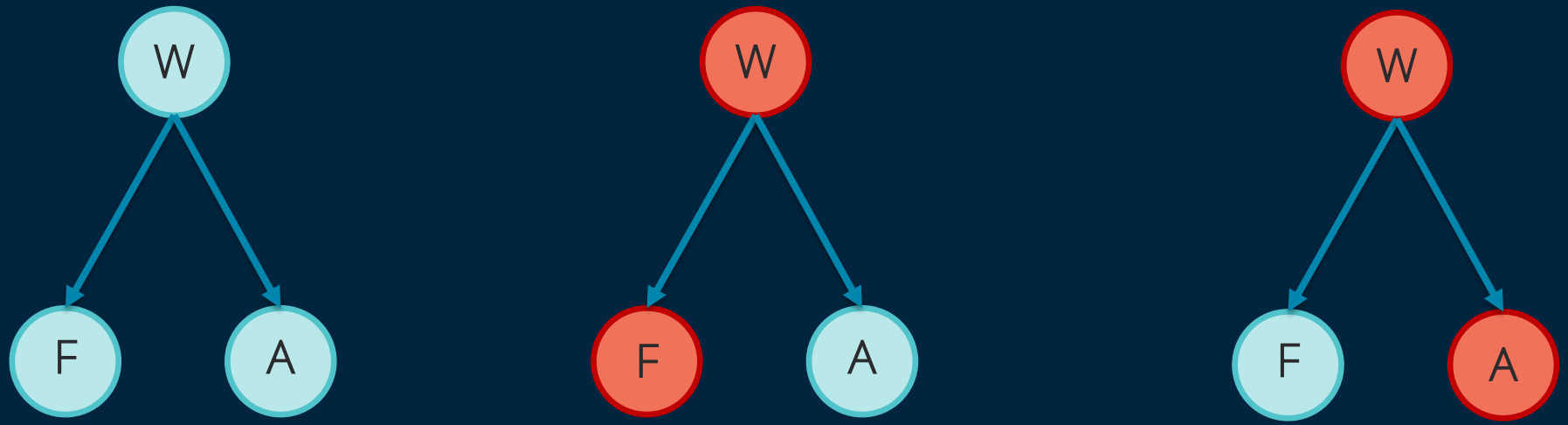
Operation QoS

Rule QoS



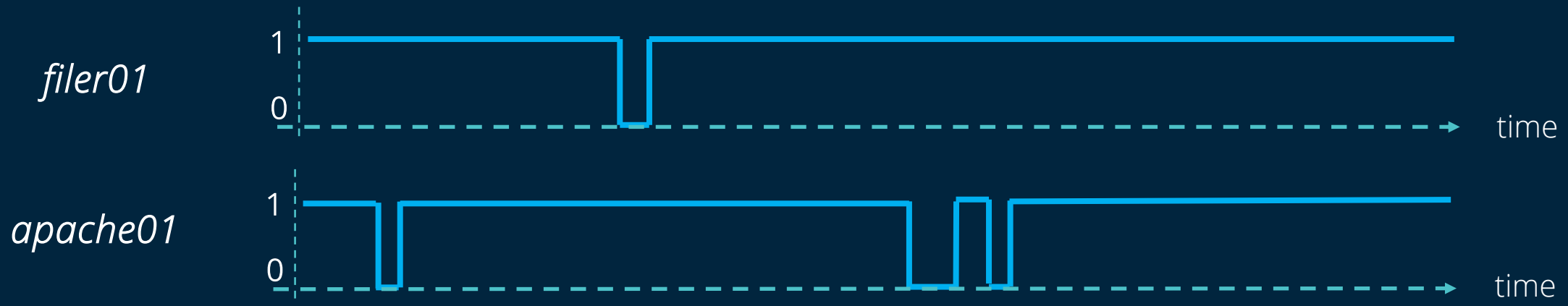
OPERATION QOS

- Website is OK when Filer **AND** Apache are OK



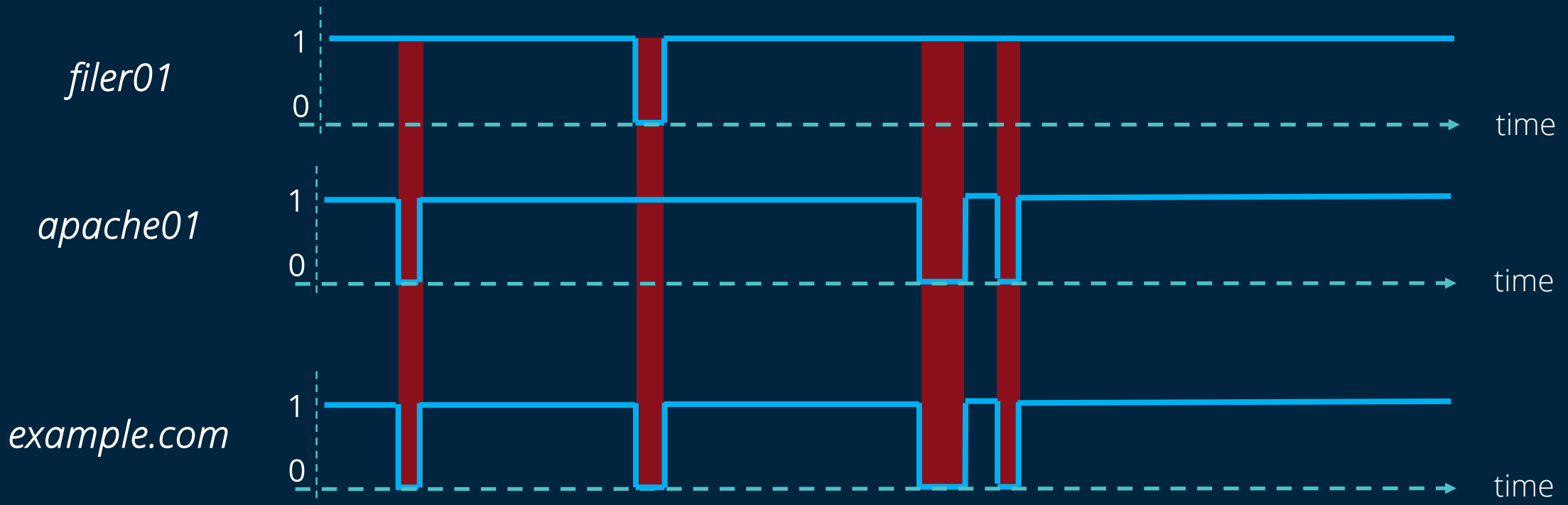
OPERATION QOS

- Apply a **AND** operation using the « boolean datapoints »

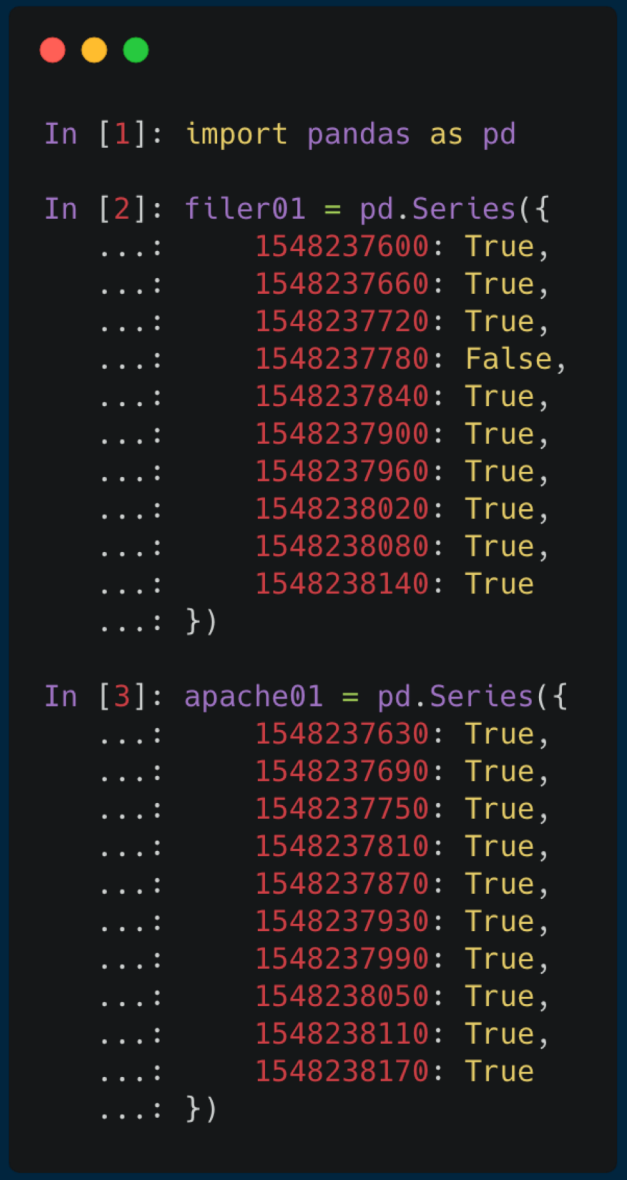


OPERATION QOS

- Apply a **AND** operation using the « boolean datapoints »



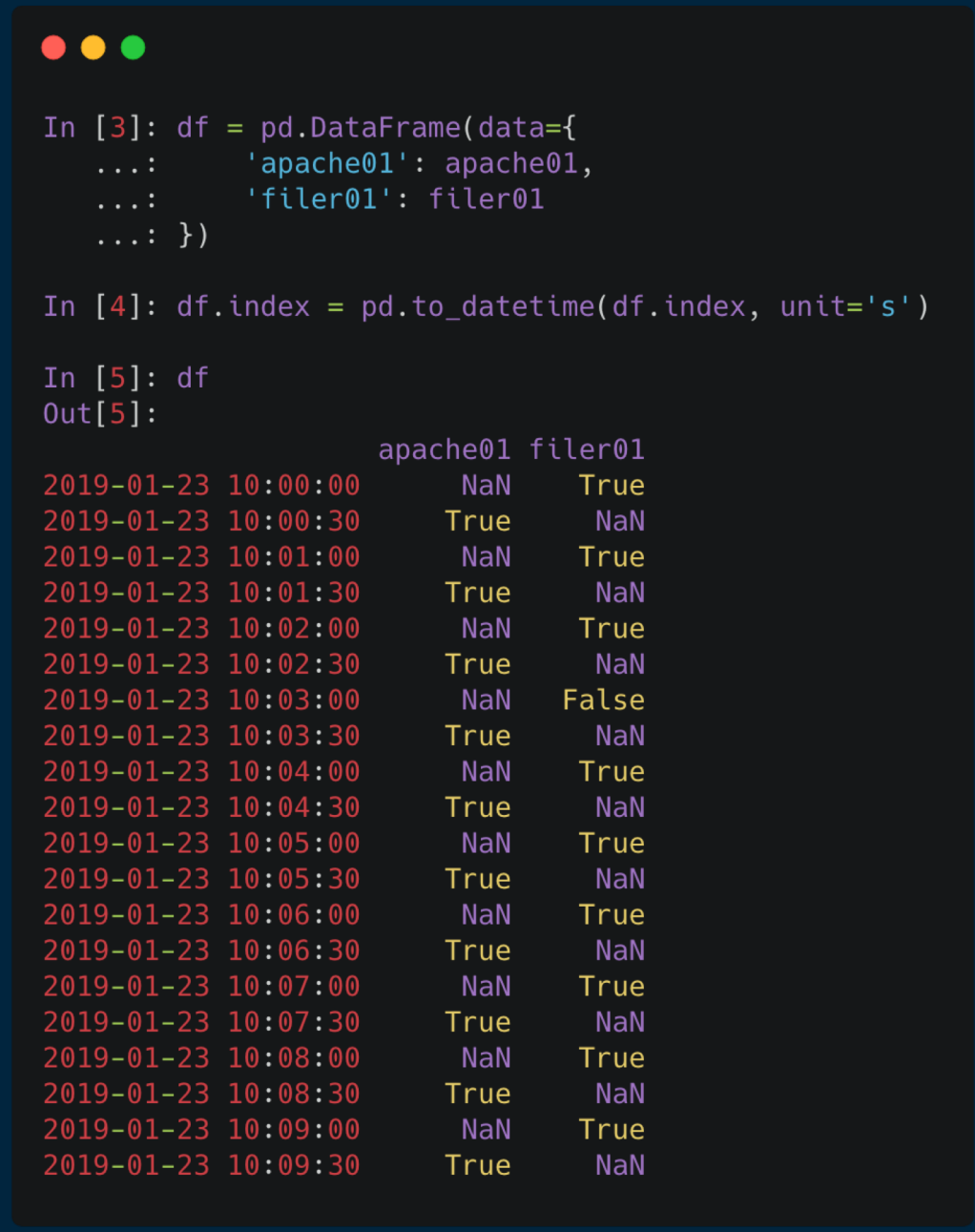
OPERATION QOS



```
In [1]: import pandas as pd

In [2]: filer01 = pd.Series({
   ...:     1548237600: True,
   ...:     1548237660: True,
   ...:     1548237720: True,
   ...:     1548237780: False,
   ...:     1548237840: True,
   ...:     1548237900: True,
   ...:     1548237960: True,
   ...:     1548238020: True,
   ...:     1548238080: True,
   ...:     1548238140: True
   ...: })

In [3]: apache01 = pd.Series({
   ...:     1548237630: True,
   ...:     1548237690: True,
   ...:     1548237750: True,
   ...:     1548237810: True,
   ...:     1548237870: True,
   ...:     1548237930: True,
   ...:     1548237990: True,
   ...:     1548238050: True,
   ...:     1548238110: True,
   ...:     1548238170: True
   ...: })
```

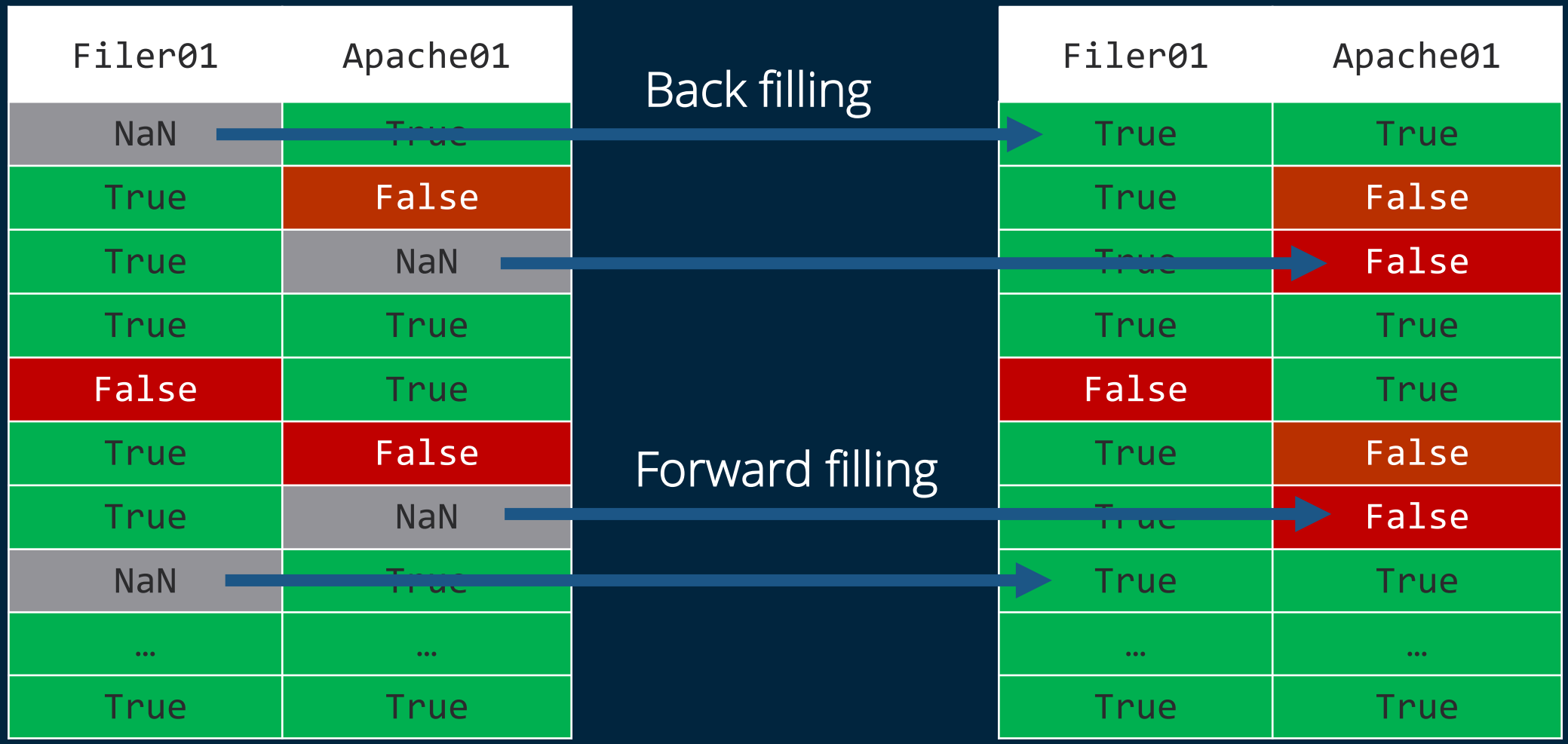


```
In [3]: df = pd.DataFrame(data={
   ...:     'apache01': apache01,
   ...:     'filer01': filer01
   ...: })

In [4]: df.index = pd.to_datetime(df.index, unit='s')

In [5]: df
Out[5]:
                     apache01 filer01
2019-01-23 10:00:00       NaN    True
2019-01-23 10:00:30      True     NaN
2019-01-23 10:01:00       NaN    True
2019-01-23 10:01:30      True     NaN
2019-01-23 10:02:00       NaN    True
2019-01-23 10:02:30      True     NaN
2019-01-23 10:03:00       NaN   False
2019-01-23 10:03:30      True     NaN
2019-01-23 10:04:00       NaN    True
2019-01-23 10:04:30      True     NaN
2019-01-23 10:05:00       NaN    True
2019-01-23 10:05:30      True     NaN
2019-01-23 10:06:00       NaN    True
2019-01-23 10:06:30      True     NaN
2019-01-23 10:07:00       NaN    True
2019-01-23 10:07:30      True     NaN
2019-01-23 10:08:00       NaN    True
2019-01-23 10:08:30      True     NaN
2019-01-23 10:09:00       NaN    True
2019-01-23 10:09:30      True     NaN
```

OPERATION QOS



```
In [3]: df = pd.DataFrame(data={
   ...:     'apache01': apache01,
   ...:     'filer01': filer01
   ...: })

In [4]: df.index = pd.to_datetime(df.index, unit='s')

In [5]: df
Out[5]:
                     apache01 filer01
2019-01-23 10:00:00       NaN    True
2019-01-23 10:00:30      True     NaN
2019-01-23 10:01:00       NaN    True
2019-01-23 10:01:30      True     NaN
2019-01-23 10:02:00       NaN    True
2019-01-23 10:02:30      True     NaN
2019-01-23 10:03:00       NaN   False
2019-01-23 10:03:30      True     NaN
2019-01-23 10:04:00       NaN    True
2019-01-23 10:04:30      True     NaN
2019-01-23 10:05:00       NaN    True
2019-01-23 10:05:30      True     NaN
2019-01-23 10:06:00       NaN    True
2019-01-23 10:06:30      True     NaN
2019-01-23 10:07:00       NaN    True
2019-01-23 10:07:30      True     NaN
2019-01-23 10:08:00       NaN    True
2019-01-23 10:08:30      True     NaN
2019-01-23 10:09:00       NaN    True
2019-01-23 10:09:30      True     NaN
```

```
In [6]: merged_df = df.fillna(method='ffill') \
   ...:                 .fillna(method='bfill')

In [7]: merged_df
Out[7]:
            apache01  filer01
1548237600      True     True
1548237630      True     True
1548237660      True     True
1548237690      True     True
1548237720      True     True
1548237750      True     True
1548237780      True    False
1548237810      True    False
1548237840      True     True
1548237870      True     True
1548237900      True     True
1548237930      True     True
1548237960      True     True
1548237990      True     True
1548238020      True     True
1548238050      True     True
1548238080      True     True
1548238110      True     True
1548238140      True     True
1548238170      True     True
```

OPERATION QOS

	Filer01	Apache01
Ts_0	True	True
Ts_1	True	False
Ts_2	True	True
Ts_3	True	True
Ts_4	False	True
Ts_5	True	False
Ts_6	True	True
Ts_7	True	True
…	…	…
Ts_n	True	True

Applying AND operation between series

Example.com
True
False
True
True
False
False
True
True
…
True

OVH

```
In [6]: merged_df = df.fillna(method='ffill') \
   ...:                .fillna(method='bfill')

In [7]: merged_df
Out[7]:
            apache01  filer01
1548237600      True     True
1548237630      True     True
1548237660      True     True
1548237690      True     True
1548237720      True     True
1548237750      True     True
1548237780      True    False
1548237810      True    False
1548237840      True     True
1548237870      True     True
1548237900      True     True
1548237930      True     True
1548237960      True     True
1548237990      True     True
1548238020      True     True
1548238050      True     True
1548238080      True     True
1548238110      True     True
1548238140      True     True
1548238170      True     True
```

```
In [8]: merged_df['results'] = merged_df.all(axis=1)

In [9]: merged_df
Out[9]:
            apache01  filer01  results
1548237600      True     True     True
1548237630      True     True     True
1548237660      True     True     True
1548237690      True     True     True
1548237720      True     True     True
1548237750      True     True     True
1548237780      True    False    False
1548237810      True    False    False
1548237840      True     True     True
1548237870      True     True     True
1548237900      True     True     True
1548237930      True     True     True
1548237960      True     True     True
1548237990      True     True     True
1548238020      True     True     True
1548238050      True     True     True
1548238080      True     True     True
1548238110      True     True     True
1548238140      True     True     True
1548238170      True     True     True
```

OPERATION QOS

- Reduce the amount of data
- Keep only changing states for next computation

```
In [10]: normalized_results = merged_df.results[
    ...:     merged_df.results.shift(1) != merged_df.results
    ...: ].dropna()
    ...: normalized_results = normalized_results.append(merged_df.results[-1:])

In [11]: normalized_results
Out[11]:
1548237600     True
1548237780    False
1548237840     True
1548238170     True
Name: results, dtype: bool
```

OVH

OPERATION QOS

- Compute the duration for each states

	Duration
False	00:01:00
True	00:08:30

```
In [12]: from datetime import datetime

In [13]: dates_results = pd.DataFrame({
    ...:     'dates': [ datetime.utcfromtimestamp(ts)
    ...:                for ts in normalized_results.index.values ],
    ...:     'results': normalized_results
    ...: })
    ...:
    ...: dates_results['periods'] = \
    ...:     dates_results.dates - dates_results.shift(1).dates
    ...: dates_results.periods = dates_results.periods.shift(-1)
    ...:
    ...: grouped_states = dates_results.groupby(['results']).sum()

In [14]: grouped_states
Out[14]:
         periods
results
False   00:01:00
True    00:08:30
```

OPERATION QOS

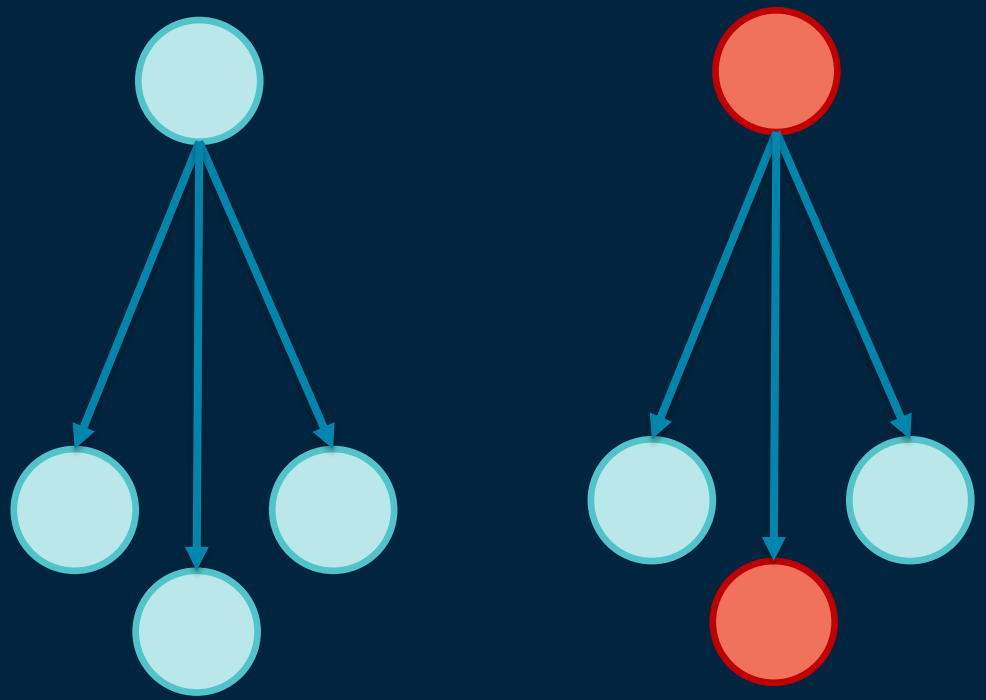
- Compute the QoS

```
In [17]: qos = grouped_states.loc[True].periods / grouped_states.periods.sum()
    ...: round(qos * 100, 3)
Out[17]: 89.474
```

QOS

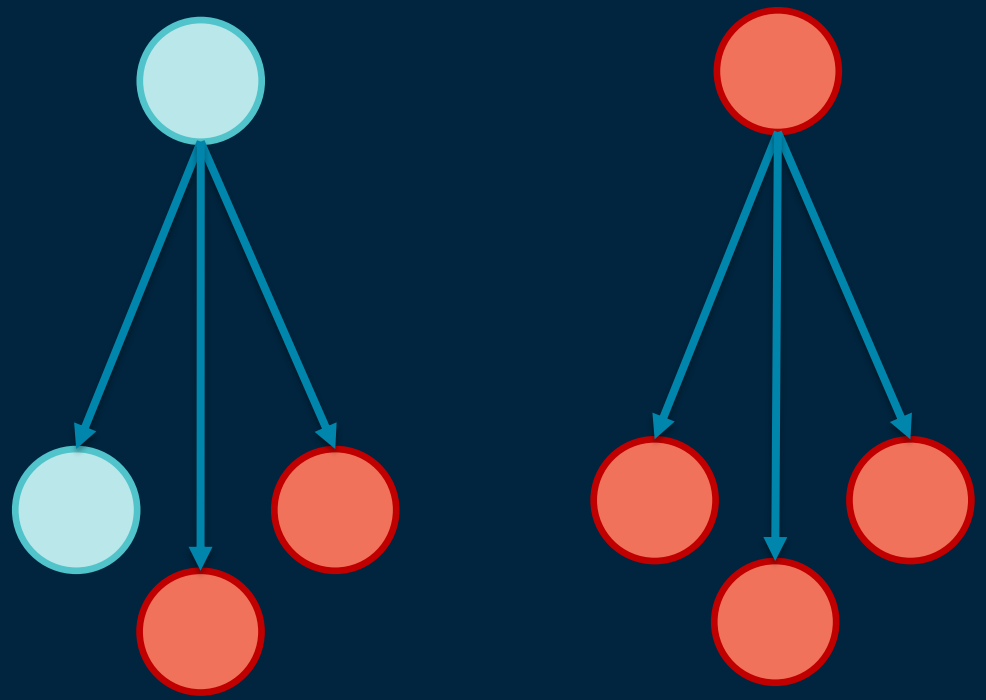
AND

Child is OK when all parents are OK



OR

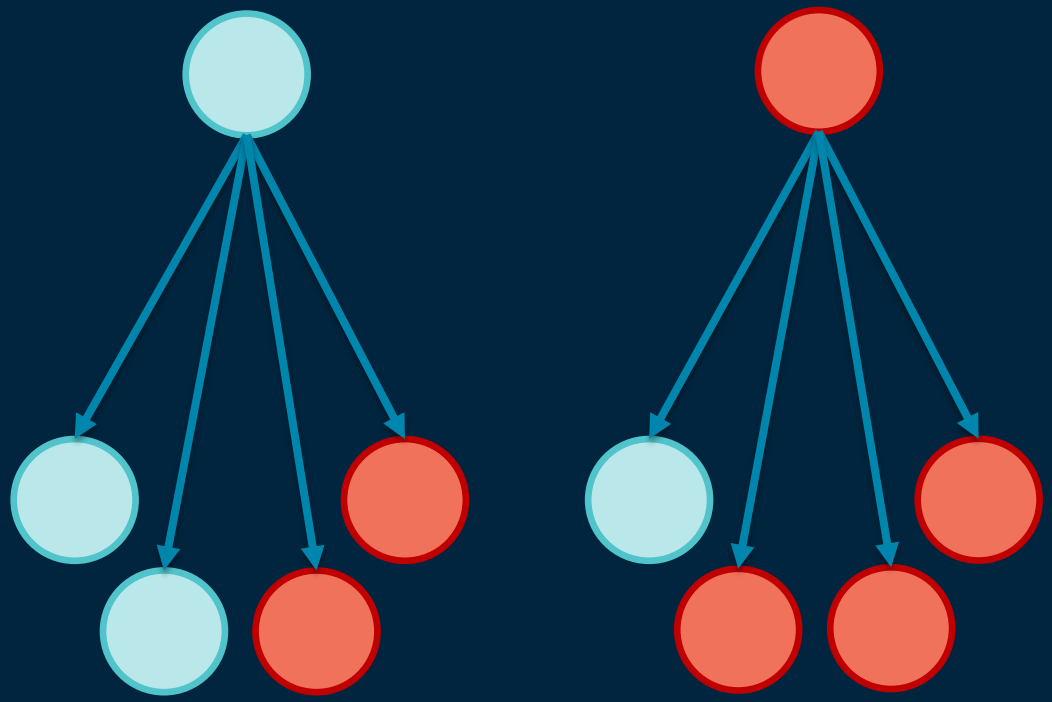
Child is OK when 1 parent is OK



QOS

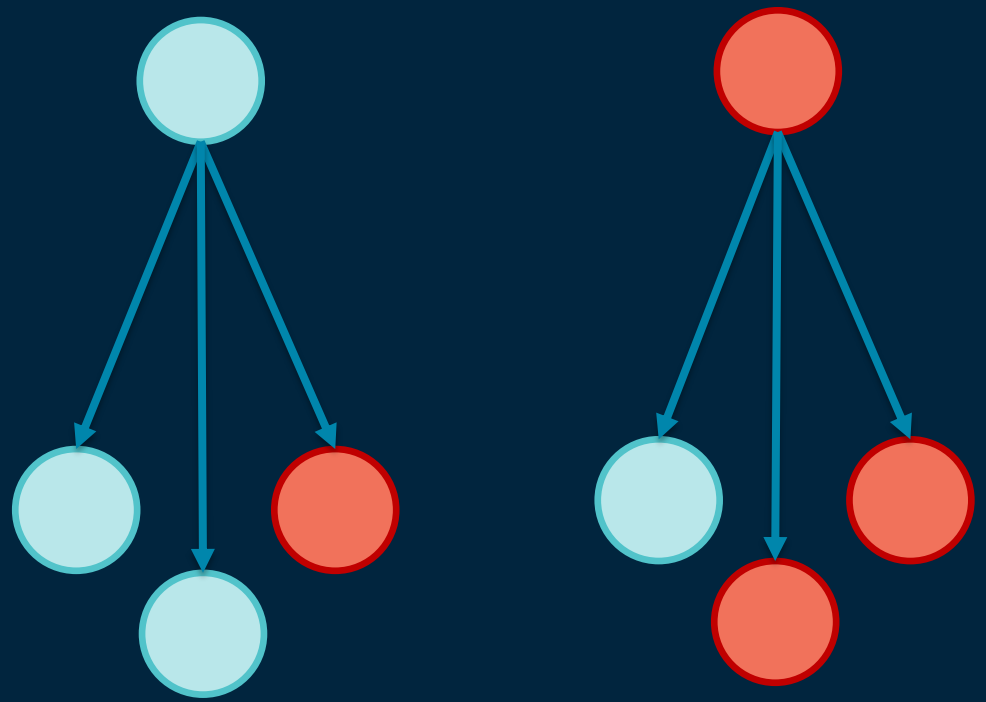
RATIO(50)

Child is OK when 50% of parent are OK

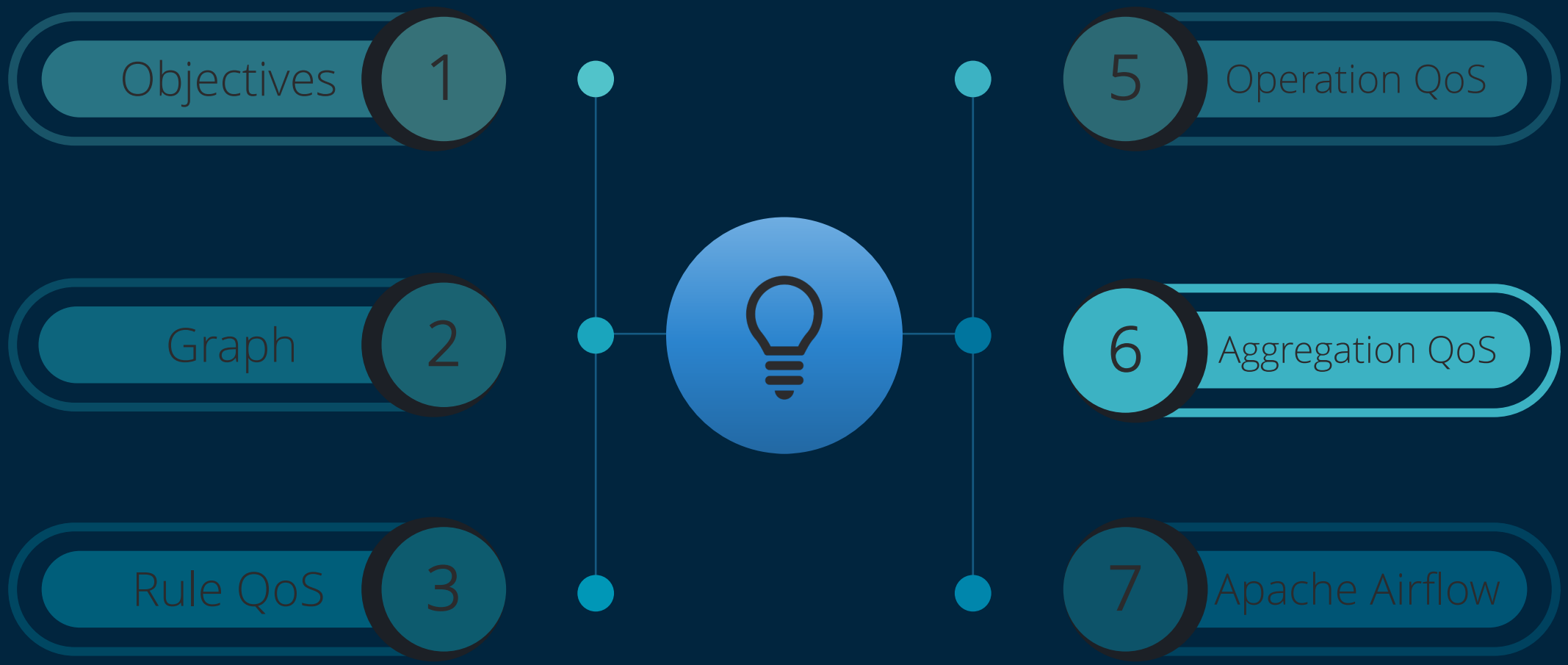


ATLEAST(2)

Child is OK when 2 parents are OK



Compute the QoS of your infrastructure with DepC

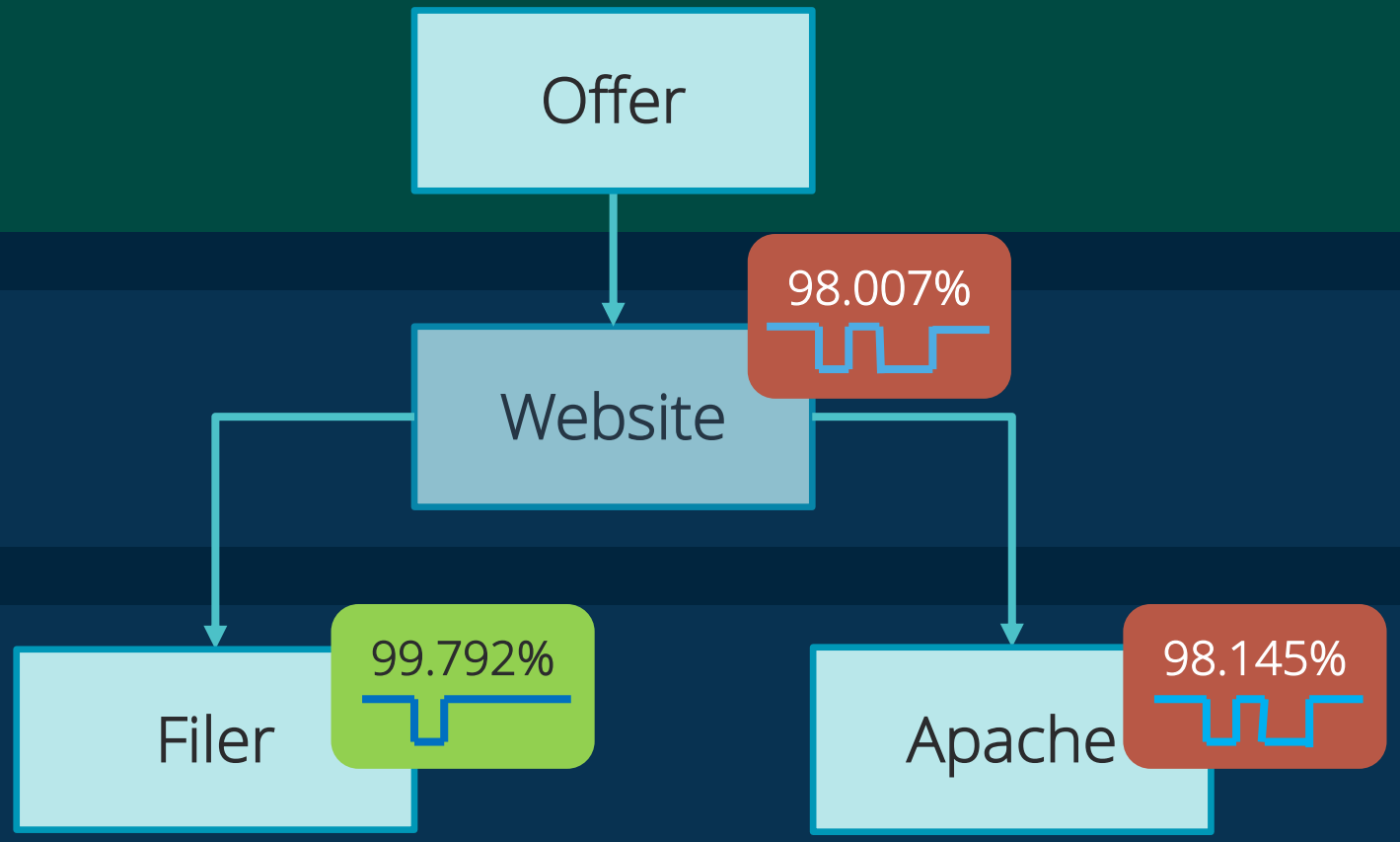


QOS

Aggregation QoS

Operation QoS

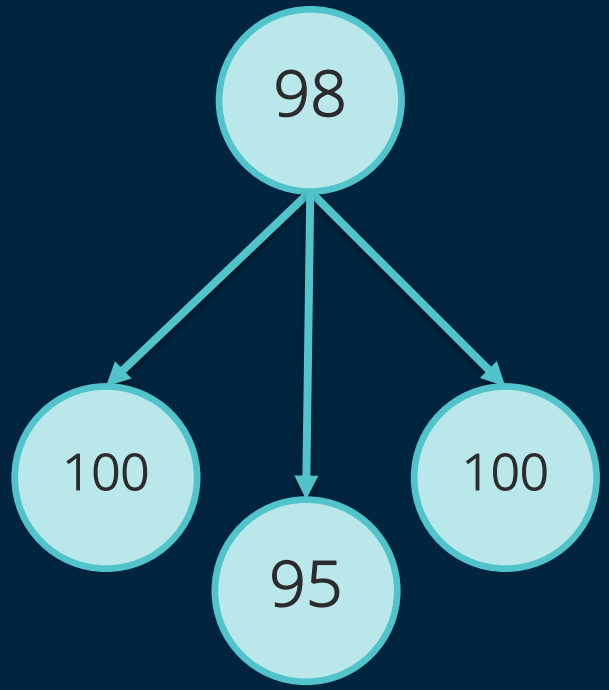
Rule QoS



QOS

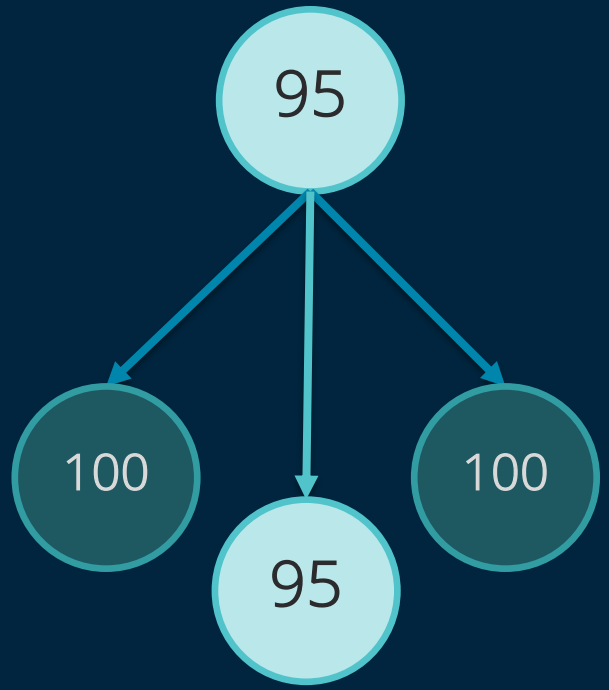
AVERAGE

Child QoS is the average of parents QoS



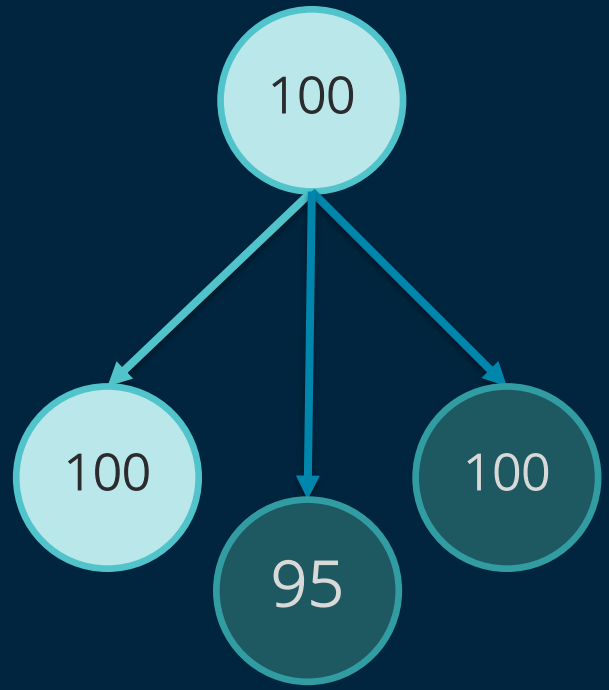
MIN

Child QoS is the minimum QoS of parents

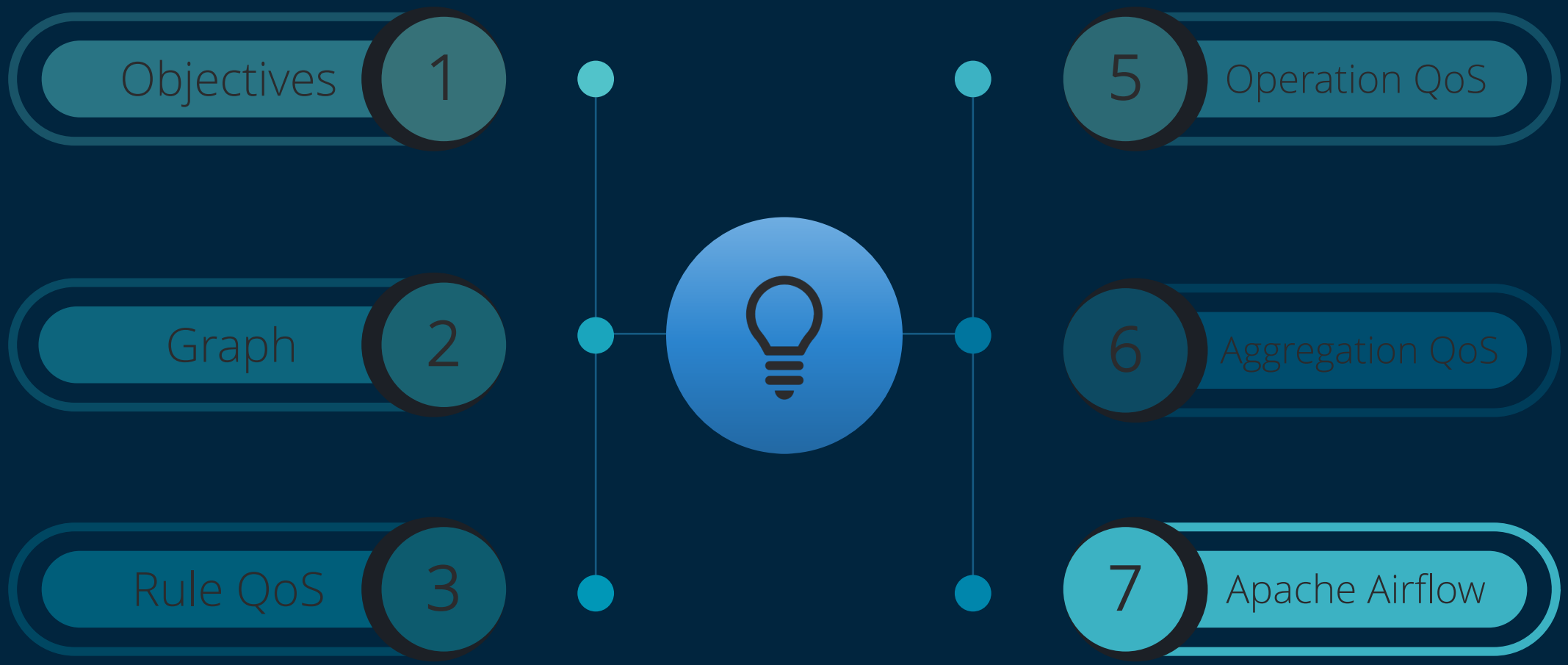


MAX

Child QoS is the maximum QoS of parents

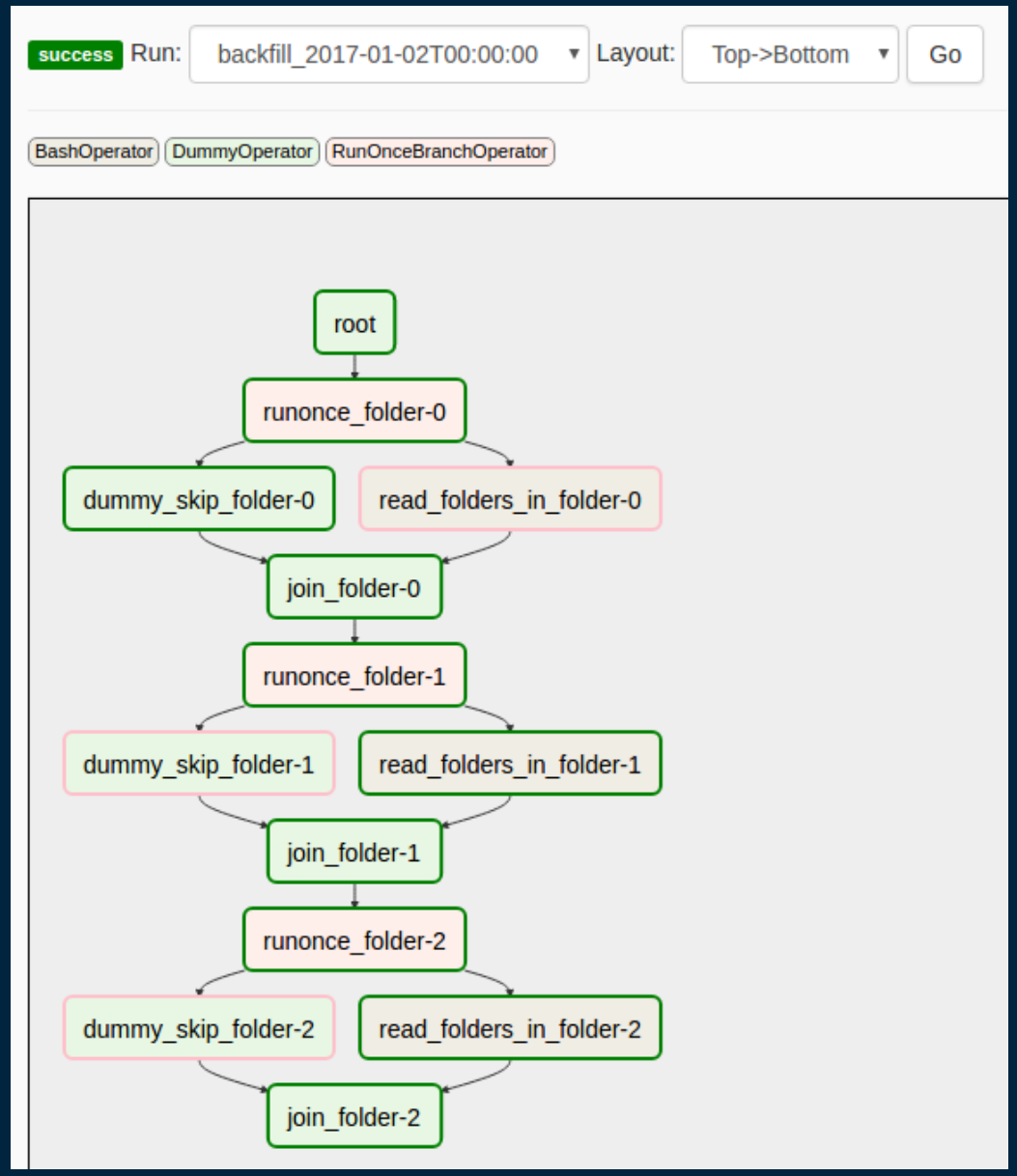


Compute the QoS of your infrastructure with DepC



APACHE AIRFLOW

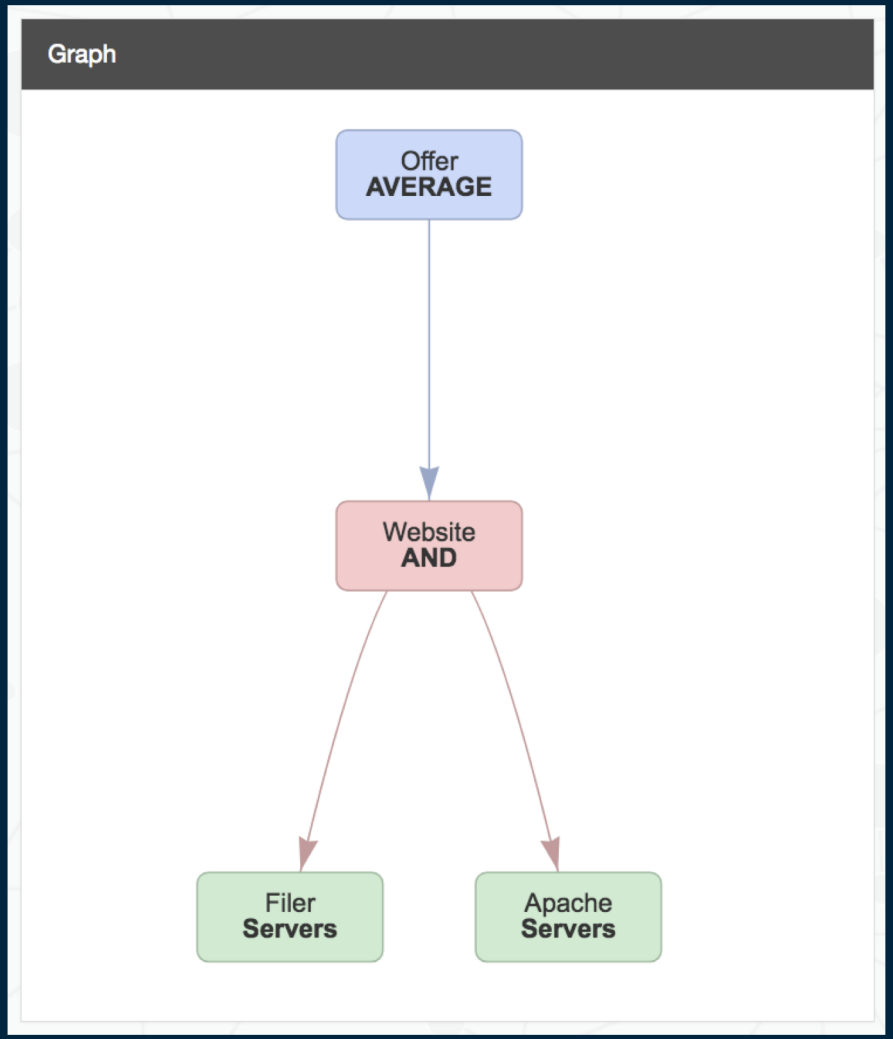
- Task scheduler in Python (cron++)
  - Worflow / Pipeline / Task dependencies
  - Uses DAG : Directed Acyclic Graph
- Features :
  - Retry
  - Logs
  - Web UI



https://medium.com/@plieningerweb/use-apache-airflow-to-run-task-exactly-once-6fb70ca5e7ec

OVH

APACHE AIRFLOW



APACHE AIRFLOW

```python
import datetime

import numpy as np
from airflow.models import DAG
from airflow.operators.python_operator \
    import PythonOperator

args = {
    "owner": "depc",
    "start_date": datetime.datetime(2019, 1, 1)
}

dag = DAG(
    dag_id="compute_acme_qos",
    default_args=args,
    schedule_interval="15 1 * * *"
)
```

```python
def compute_rule(ds, **kwargs):
    task_id = kwargs["task"].task_id
    return f"Compute rule for {task_id}"


def compute_operation(ds, **kwargs):
    task_id = kwargs["task"].task_id
    return f"Compute operation for {task_id}"


def compute_aggregation(ds, **kwargs):
    task_id = kwargs["task"].task_id
    return f"Compute aggregation for {task_id}"
```

OVH

APACHE AIRFLOW

```python
def create_task(task_id, python_callable,
                op_kwargs=None):
    return PythonOperator(
        task_id=task_id,
        provide_context=True,
        python_callable=python_callable,
        op_kwargs=op_kwargs if op_kwargs else {},
        dag=dag,
    )


apache_task = create_task(
    task_id="Apache",
    python_callable=compute_rule
)
filer_task = create_task(
    task_id="Filer",
    python_callable=compute_rule
)
```

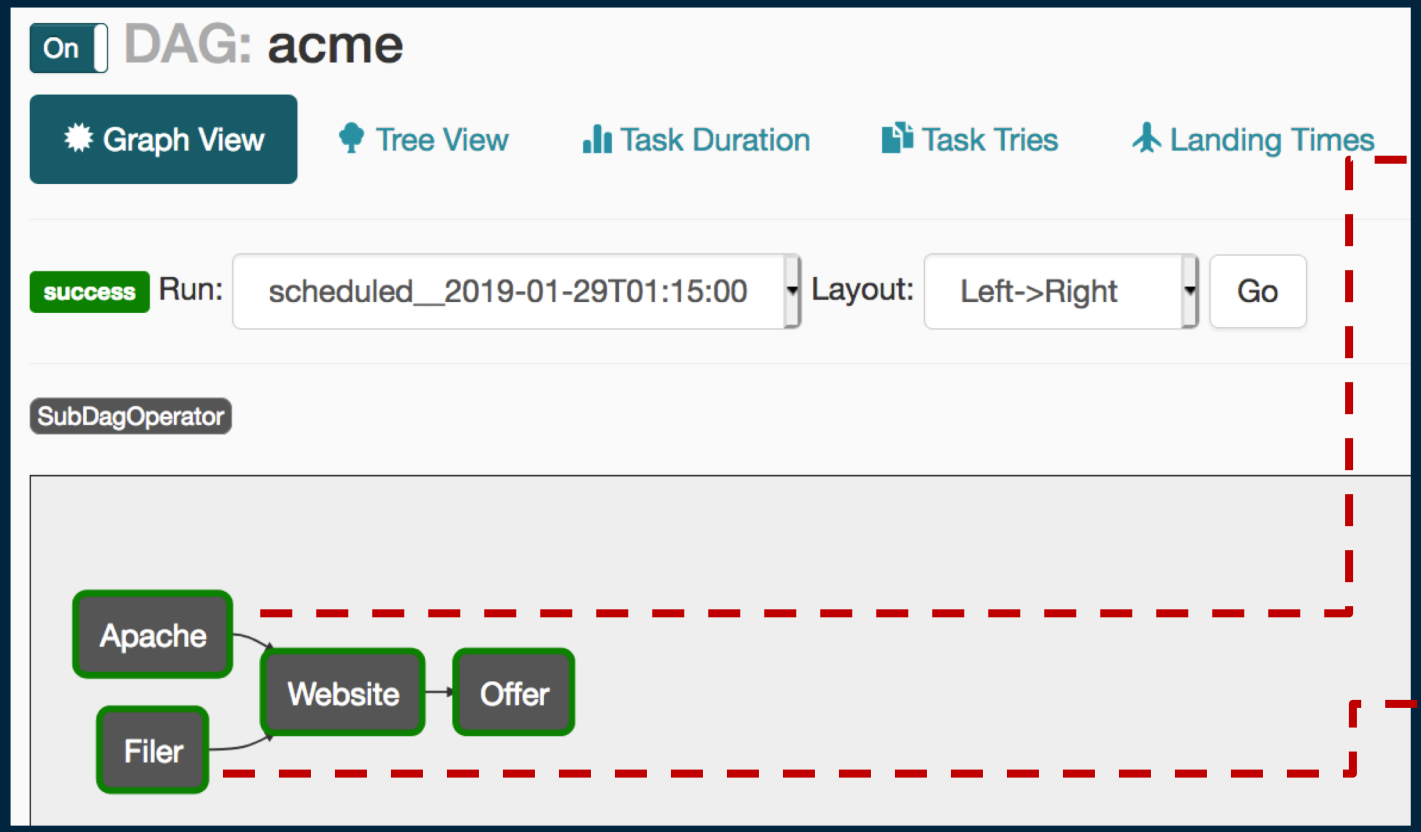
```python
website_task = create_task(
    task_id="Website",
    python_callable=compute_operation,
    op_kwargs={"operation": np.all},
)
offer_task = create_task(
    task_id="Offer",
    python_callable=compute_aggregation,
    op_kwargs={"aggregation": np.average},
)

[
  apache_task,
  filer_task
] >> website_task >> offer_task
```

Scheduling with bitwise operators

OVH

APACHE AIRFLOW



OVH

Q&A

@ovh

linkedin.com/company/ovh/

youtube.com/ovh

facebook.com/ovhcom/

OVH