Wizzie Data Platform: PROCESSING VARIED DATA STREAMS AT CONFIGURABLE WAY

Andrés Gómez Ferrer


Industry 4.0
Arduino
Temperature
MQTT
IoT
SNMP
Security
Sensors
ulnerabilities
Smartphones
Location
Coordinates
Proximity
Venues
WLC

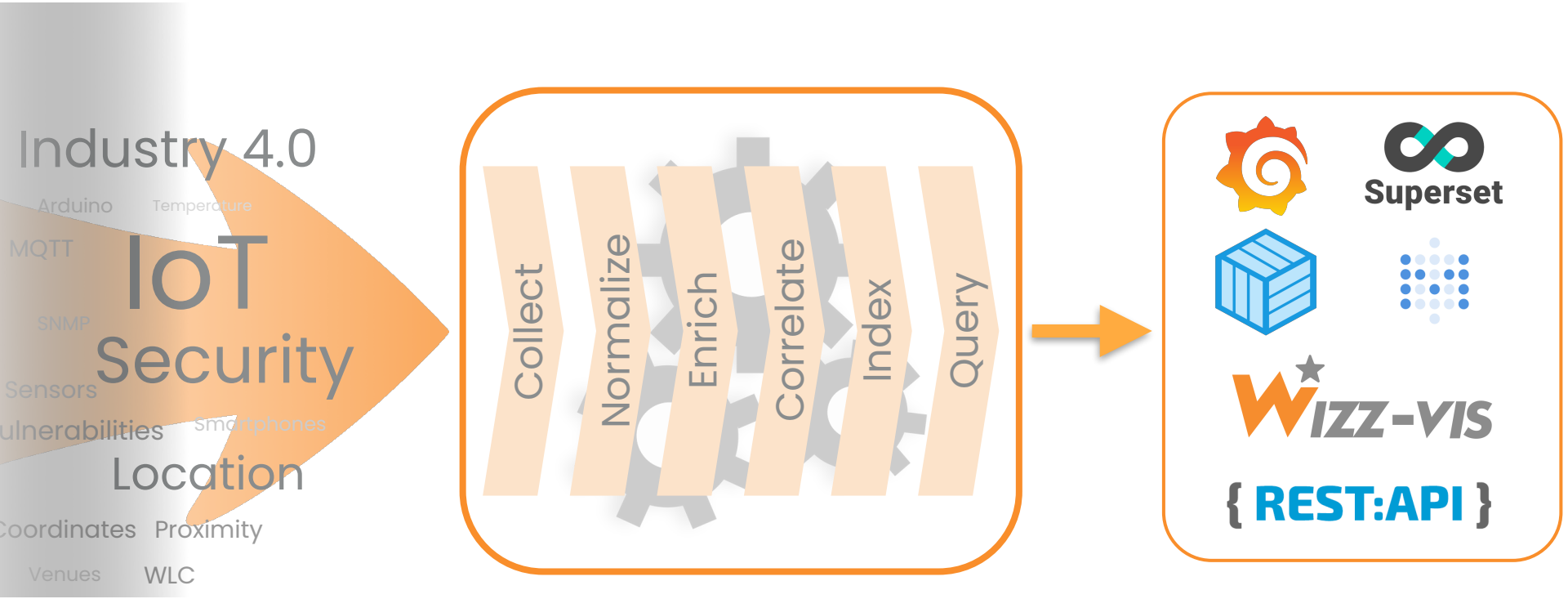

Industry 4.0
Arduino
Temperature
MQTT
IoT
SNMP
Security
Sensors
ulnerabilities
Smartphones
Location
Coordinates
Proximity
Venues
WLC
Collect


Industry 4.0
Arduino
Temperature
MQTT
IoT
SNMP
Security
Sensors
ulnerabilities
Smartphones
Location
Coordinates
Proximity
Venues
WLC
Collect
Normalize


Industry 4.0
Arduino
Temperature
MQTT
IoT
SNMP
Security
Sensors
ulnerabilities
Smartphones
Location
Coordinates
Proximity
Venues
WLC
Collect
Normalize
Enrich


Industry 4.0
Arduino
Temperature
MQTT
IoT
SNMP
Security
Sensors
ulnerabilities
Smartphones
Location
Coordinates
Proximity
Venues
WLC
Collect
Normalize
Enrich
Correlate


Industry 4.0
Arduino
Temperature
MQTT
IoT
SNMP
Security
Sensors
ulnerabilities
Smartphones
Location
Coordinates
Proximity
Venues
WLC
Collect
Normalize
Enrich
Correlate
Index


Industry 4.0
Arduino
Temperature
MQTT
IoT
SNMP
Security
Sensors
ulnerabilities
Smartphones
Location
Coordinates
Proximity
Venues
WLC
Collect
Normalize
Enrich
Correlate
Index
Query
Superset
WIZZ-VIS
{ REST:API }

wizzie

Requirements:

- Horizontal scalability, elastic, fault-tolerance, resilience…

Requirements:

- Horizontal scalability, elastic, fault-tolerance, resilience…
- Data with **flexible schema** or schema-less.

Requirements:

- Horizontal scalability, elastic, fault-tolerance, resilience…
- Data with **flexible schema** or schema-less.
- Quickly & easy configuration changes -> **without coding**.

Requirements:

- Horizontal scalability, elastic, fault-tolerance, resilience…
- Data with **flexible schema** or schema-less.
- Quickly & easy configuration changes -> **without coding**.

Requirements:

- Horizontal scalability, elastic, fault-tolerance, resilience…
- Data with **flexible schema** or schema-less.
- Quickly & easy configuration changes -> **without coding**.
- Easy to integrate new data processing components.

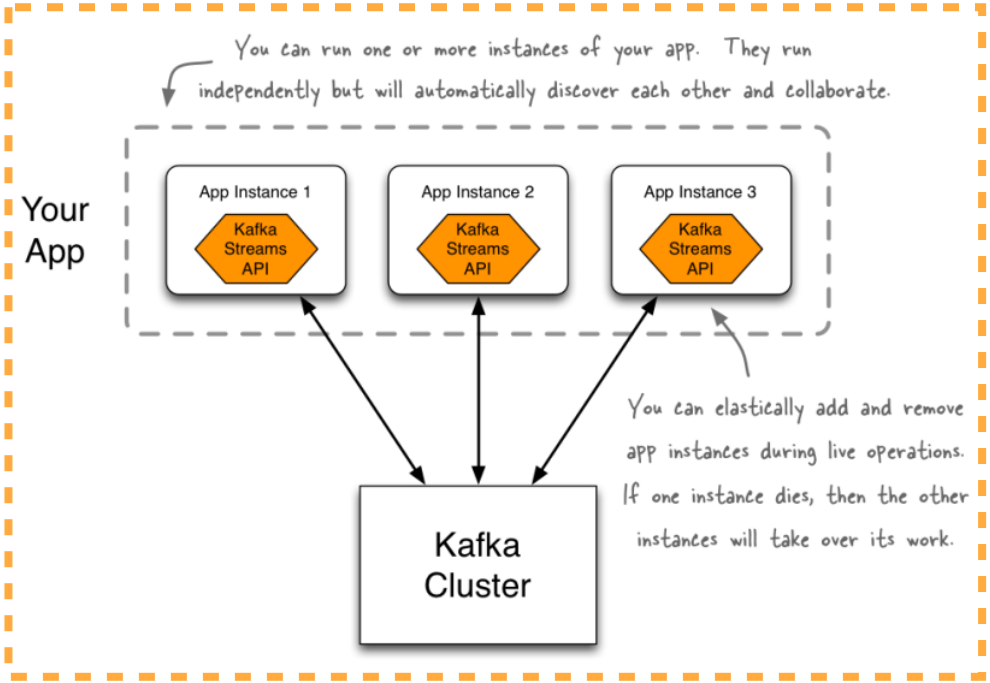
https://kafka.apache.org

Kafka Broker

- Distributed message queue system.
- Provides:
  - Highly scalable
  - Fault tolerance
  - Back pressure

https://kafka.apache.org

kafka

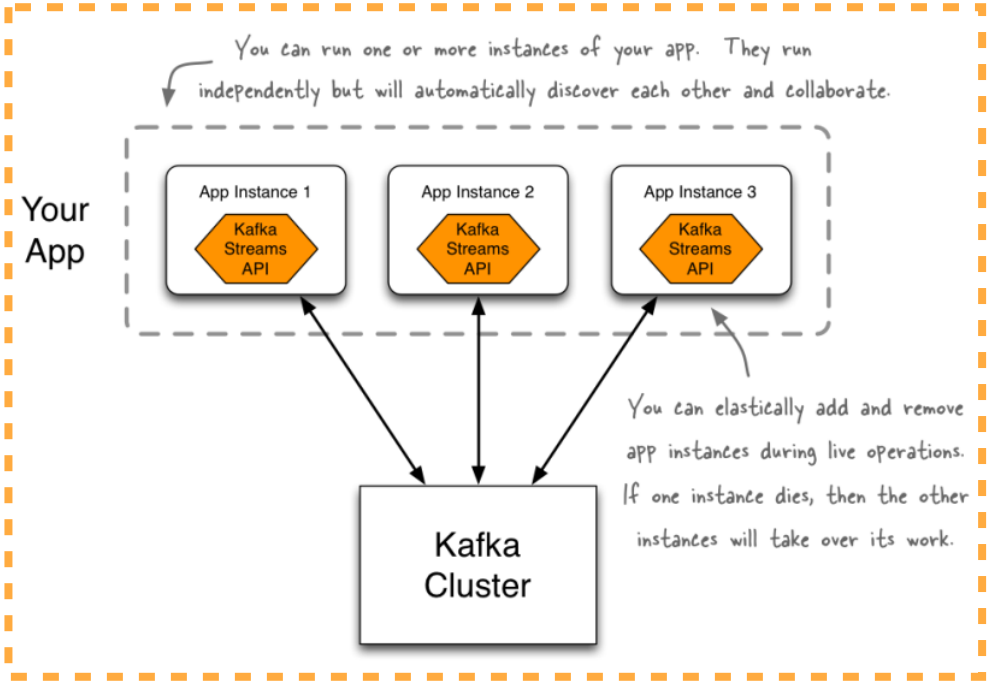


wizzie

kafka

Kafka Streams

- Client library for building applications and microservices to work with streaming data.
- **Provides**:
  - Highly scalable
  - Fault tolerance
  - Stateless & Stateful applications
  - Exactly-once semantics



wizzie

https://wizzie-io.github.io/normalizer/

NORMALIZER

Engine

- Streaming processing engine based on Kafka Streams.
- Configuration via JSON files.
- Allows to apply maps, flatmaps and filters functions:

https://wizzie-io.github.io/normalizer/

NORMALIZER

Engine

- Streaming processing engine based on Kafka Streams.
- Configuration via JSON files.
- Allows to apply maps, flatmaps and filters functions:
  - Projection fields
  - Array flatten
  - Arithmetic support
  - Work with times
  - Strings operations: replace, joins, upper/lower case, regex
  - Jq language support

https://wizzie-io.github.io/normalizer/

NORMALIZER

Engine

- Streaming processing engine based on Kafka Streams.
- Configuration via JSON files.
- Allows to apply maps, flatmaps and filters functions:
  - Projection fields
  - Array flatten
  - Arithmetic support
  - Work with times
  - Strings operations: replace, joins, upper/lower case, regex
  - Jq language support

https://wizzie-io.github.io/normalizer/

Deployment Mode

Distribution package

kubernetes


```
{
    "inputs":{
      "kafka_topic_in":["my_stream_example"]
    },
    "streams":{
      "my_stream_example":{
          "funcs":[
                {
                  "name":"selection_fields",
                  "className":"io.wizzie.normalizer.funcs.impl.SimpleMapper",
                  "properties": {
                    "maps": [
                      {"dimPath":["body", "messages"]},
                      {"dimPath":["header", "timestamp"], "as": "time"},
                      {"dimPath":["header", "mac"], "as": "id"}
                    ]
                  }
                },
                {
                  "name":"flatten_mnessages",
                  "className":"io.wizzie.normalizer.funcs.impl.ArrayFlattenMapper",
                  "properties": {
                    "flat_dimension": "messages"
                  }
                },
```

NORMALIZER

```
{
    "inputs":{
      "kafka_topic_in":["my_stream_example"]
    },
    "streams":{
      "my_stream_example":{
          "funcs":[
                {
                  "name":"selection_fields",
                  "className":"io.wizzie.normalizer.funcs.impl.SimpleMapper",
                  "properties": {
                    "maps": [
                      {"dimPath":["body", "messages"]},
                      {"dimPath":["header", "timestamp"], "as": "time"},
                      {"dimPath":["header", "mac"], "as": "id"}
                    ]
                  }
                },
                {
                  "name":"flatten_mnessages",
                  "className":"io.wizzie.normalizer.funcs.impl.ArrayFlattenMapper",
                  "properties": {
                    "flat_dimension": "messages"
                  }
                },
```

Inputs

- Define the relations between streams and Kafka topic.

NORMALIZER

```
{
    "inputs":{
      "kafka_topic_in":["my_stream_example"]
    },
    "streams":{
      "my_stream_example":{
          "funcs":[
                {
                  "name":"selection_fields",
                  "className":"io.wizzie.normalizer.funcs.impl.SimpleMapper",
                  "properties": {
                    "maps": [
                      {"dimPath":["body", "messages"]},
                      {"dimPath":["header", "timestamp"], "as": "time"},
                      {"dimPath":["header", "mac"], "as": "id"}
                    ]
                  }
                },
                {
                  "name":"flatten_mnessages",
                  "className":"io.wizzie.normalizer.funcs.impl.ArrayFlattenMapper",
                  "properties": {
                    "flat_dimension": "messages"
                  }
                },
```

Streams

- Stream definition, each stream has *functions* & *sinks.*

NORMALIZER

```
{
    "inputs":{
      "kafka_topic_in":["my_stream_example"]
    },
    "streams":{
      "my_stream_example":{
          "funcs":[
                {
                  "name":"selection_fields",
                  "className":"io.wizzie.normalizer.funcs.impl.SimpleMapper",
                  "properties": {
                    "maps": [
                      {"dimPath":["body", "messages"]},
                      {"dimPath":["header", "timestamp"], "as": "time"},
                      {"dimPath":["header", "mac"], "as": "id"}
                    ]
                  }
                },
                {
                  "name":"flatten_mnessages",
                  "className":"io.wizzie.normalizer.funcs.impl.ArrayFlattenMapper",
                  "properties": {
                    "flat_dimension": "messages"
                  }
                },
```

Functions

- List of functions where define the stream operations: transformations and filters

NORMALIZER

```
{
    "inputs":{
      "kafka_topic_in":["my_stream_example"]
    },
    "streams":{
      "my_stream_example":{
          "funcs":[
                {
                  "name":"selection_fields",
                  "className":"io.wizzie.normalizer.funcs.impl.SimpleMapper",
                  "properties": {
                    "maps": [
                      {"dimPath":["body", "messages"]},
                      {"dimPath":["header", "timestamp"], "as": "time"},
                      {"dimPath":["header", "mac"], "as": "id"}
                    ]
                  }
                },
                {
                  "name":"flatten_mnessages",
                  "className":"io.wizzie.normalizer.funcs.impl.ArrayFlattenMapper",
                  "properties": {
                    "flat_dimension": "messages"
                  }
                },
```

```
{
  "header": {
    "mac": "00:00:00:00:00:00",
    "version": 2,
    "timestamp": 1542882400123
  },
  "body": {
    "messages": [
      {
        "address": "1.1.1.1:4312",
        "status": "on"
      },
      {
        "address": "1.1.2.1:4312",
        "status": "off"
      }
    ]
  }
}
```

NORMALIZER

```
{
    "inputs":{
      "kafka_topic_in":["my_stream_example"]
    },
    "streams":{
      "my_stream_example":{
          "funcs":[
                {
                  "name":"selection_fields",
                  "className":"io.wizzie.normalizer.funcs.impl.SimpleMapper",
                  "properties": {
                    "maps": [
                      {"dimPath":["body", "messages"]},
                      {"dimPath":["header", "timestamp"], "as": "time"},
                      {"dimPath":["header", "mac"], "as": "id"}
                    ]
                  }
                },
                {
                  "name":"flatten_mnessages",
                  "className":"io.wizzie.normalizer.funcs.impl.ArrayFlattenMapper",
                  "properties": {
                    "flat_dimension": "messages"
                  }
                },
```

```
{
  "header": {
    "mac": "00:00:00:00:00:00",
    "version": 2,
    "timestamp": 1542882400123
  },
  "body": {
    "messages": [
      {
        "address": "1.1.1.1:4312",
        "status": "on"
      },
      {
        "address": "1.1.2.1:4312",
        "status": "off"
      }
    ]
  }
}
```

```
{
  "id": "00:00:00:00:00:00",
  "time": 1542882400123,
  "messages": [
    {
      "address": "1.1.1.1:4312",
      "status": "on"
    },
    {
      "address": "1.1.2.1:4312",
      "status": "off"
    }
  ]
}
```

NORMALIZER

```
{
    "inputs":{
      "kafka_topic_in":["my_stream_example"]
    },
    "streams":{
      "my_stream_example":{
          "funcs":[
                {
                  "name":"selection_fields",
                  "className":"io.wizzie.normalizer.funcs.impl.SimpleMapper",
                  "properties": {
                    "maps": [
                      {"dimPath":["body", "messages"]},
                      {"dimPath":["header", "timestamp"], "as": "time"},
                      {"dimPath":["header", "mac"], "as": "id"}
                    ]
                  }
                },
                {
                  "name":"flatten_mnessages",
                  "className":"io.wizzie.normalizer.funcs.impl.ArrayFlattenMapper",
                  "properties": {
                    "flat_dimension": "messages"
                  }
                },
```

```
{
  "header": {
    "mac": "00:00:00:00:00",
    "version": 2,
    "timestamp": 1542882400123
  },
  "body": {
    "messages": [
      {
        "address": "1.1.1.1:4312",
        "status": "on"
      },
      {
        "address": "1.1.2.1:4312",
        "status": "off"
      }
    ]
  }
}
```

```
{
  "id": "00:00:00:00:00",
  "time": 1542882400123,
  "messages": [
    {
      "address": "1.1.1.1:4312",
      "status": "on"
    },
    {
      "address": "1.1.2.1:4312",
      "status": "off"
    }
  ]
}
```

```
{
  "id": "00:00:00:00:00",
  "time": 1542882400123,
  "address": "1.1.1.1:4312",
  "status": "on"
}

{
  "id": "00:00:00:00:00",
  "time": 1542882400123,
  "address": "1.1.2.1:4312",
  "status": "off"
}
```


```
                {
                  "name": "time_to_sec",
                  "className": "io.wizzie.normalizer.funcs.impl.TimeMapper",
                  "properties": {
                    "dimension":"time",
                    "fromFormat":"millis",
                    "toFormat":"secs"
                  }
                },
                {
                   "name":"split_ip_port",
                   "className":"io.wizzie.normalizer.funcs.impl.StringSplitterMapper",
                   "properties": {
                     "dimension": "address",
                     "delimiter": ":",
                     "fields": ["ip", "port"]
                   }
                 }
            ],
            "sinks":[
                {"topic":"kafka_topic_out", "type":"kafka", "partitionBy":"id"}
            ]
        }
    }
}
```

NORMALIZER

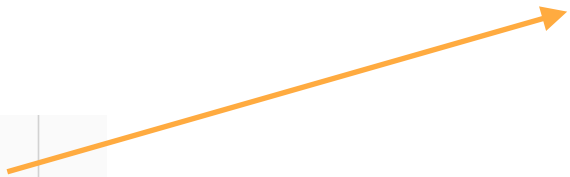
```
                {
                  "name": "time_to_sec",
                  "className": "io.wizzie.normalizer.funcs.impl.TimeMapper",
                  "properties": {
                    "dimension":"time",
                    "fromFormat":"millis",
                    "toFormat":"secs"
                  }
                },
                {
                   "name":"split_ip_port",
                   "className":"io.wizzie.normalizer.funcs.impl.StringSplitterMapper",
                   "properties": {
                     "dimension": "address",
                     "delimiter": ":",
                     "fields": ["ip", "port"]
                   }
                 }
            ],
            "sinks":[
                {"topic":"kafka_topic_out", "type":"kafka", "partitionBy":"id"}
            ]
        }
    }
}
```

Sinks

- Sinks are the outputs.
- Sinks allow to repartition the stream using a new key.
- There are two types of sinks:
  - Internal Sink: stream
  - External Sink: kafka

NORMALIZER

```
                {
                  "name": "time_to_sec",
                  "className": "io.wizzie.normalizer.funcs.impl.TimeMapper",
                  "properties": {
                    "dimension":"time",
                    "fromFormat":"millis",
                    "toFormat":"secs"
                  }
                },
                {
                   "name":"split_ip_port",
                   "className":"io.wizzie.normalizer.funcs.impl.StringSplitterMapper",
                   "properties": {
                     "dimension": "address",
                     "delimiter": ":",
                     "fields": ["ip", "port"]
                   }
                 }
            ],
            "sinks":[
                {"topic":"kafka_topic_out", "type":"kafka", "partitionBy":"id"}
            ]
        }
    }
}
```



```
{
  "id": "00:00:00:00:00",
  "time": 1542882400,
  "address": "1.1.1.1:4312",
  "status": "on"
}

{
  "id": "00:00:00:00:00",
  "time": 1542882400,
  "address": "1.1.2.1:4312",
  "status": "off"
}
```

NORMALIZER

```
                {
                  "name": "time_to_sec",
                  "className": "io.wizzie.normalizer.funcs.impl.TimeMapper",
                  "properties": {
                    "dimension":"time",
                    "fromFormat":"millis",
                    "toFormat":"secs"
                  }
                },
                {
                   "name":"split_ip_port",
                   "className":"io.wizzie.normalizer.funcs.impl.StringSplitterMapper",
                   "properties": {
                     "dimension": "address",
                     "delimiter": ":",
                     "fields": ["ip", "port"]
                   }
                 }
            ],
            "sinks":[
                {"topic":"kafka_topic_out", "type":"kafka", "partitionBy":"id"}
            ]
        }
      }
  }
```

```
{
  "id": "00:00:00:00:00",
  "time": 1542882400,
  "ip": "1.1.1.1",
  "port": "4312",
  "status": "on"
}

{
  "id": "00:00:00:00:00",
  "time": 1542882400,
  "ip": "1.1.2.1",
  "port":"4312",
  "status": "off"
}
```

```
{
  "id": "00:00:00:00:00",
  "time": 1542882400,
  "address": "1.1.1.1:4312",
  "status": "on"
}

{
  "id": "00:00:00:00:00",
  "time": 1542882400,
  "address": "1.1.2.1:4312",
  "status": "off"
}
```

NORMALIZER

```
                {
                  "name": "time_to_sec",
                  "className": "io.wizzie.normalizer.funcs.impl.TimeMapper",
                  "properties": {
                    "dimension":"time",
                    "fromFormat":"millis",
                    "toFormat":"secs"
                  }
                },
                {
                   "name":"split_ip_port",
                   "className":"io.wizzie.normalizer.funcs.impl.StringSplitterMapper",
                   "properties": {
                     "dimension": "address",
                     "delimiter": ":",
                     "fields": ["ip", "port"]
                   }
                 }
            ],
            "sinks":[
                {"topic":"kafka_topic_out", "type":"kafka", "partitionBy":"id"}
            ]
        }
    }
}
```

```
{
  "id": "00:00:00:00:00",
  "time": 1542882400,
  "address": "1.1.1.1:4312",
  "status": "on"
}

{
  "id": "00:00:00:00:00",
  "time": 1542882400,
  "address": "1.1.2.1:4312",
  "status": "off"
}
```

```
{
  "id": "00:00:00:00:00",
  "time": 1542882400,
  "ip": "1.1.1.1",
  "port": "4312",
  "status": "on"
}

{
  "id": "00:00:00:00:00",
  "time": 1542882400,
  "ip": "1.1.2.1",
  "port":"4312",
  "status": "off"
}
```

```
"00:00:00:00:00" ->
  {"id":"00:00:00:00:00","time":1542882400,"ip":"1.1.1.1","port":"4312","status":"on"}

"00:00:00:00:00" ->
  {"id":"00:00:00:00:00","time":1542882400,"ip":"1.1.2.1","port":"4312","status":"off"}
```

https://wizzie-io.github.io/enricher/

Engine

- Streaming processing engine based on Kafka Streams.
- Configuration via JSON files and **SQL friendly**.
- Allows to build tables from streams and join them with other streams.
- Enrich messages with external data using specific enrichers. Example: GeoIP.

https://wizzie-io.github.io/enricher/

ENRICHER

Engine

- Streaming processing engine based on Kafka Streams.
- Configuration via JSON files and **SQL friendly**.
- Allows to build tables from streams and join them with other streams.
- Enrich messages with external data using specific enrichers. Example: GeoIP.

Deployment Mode

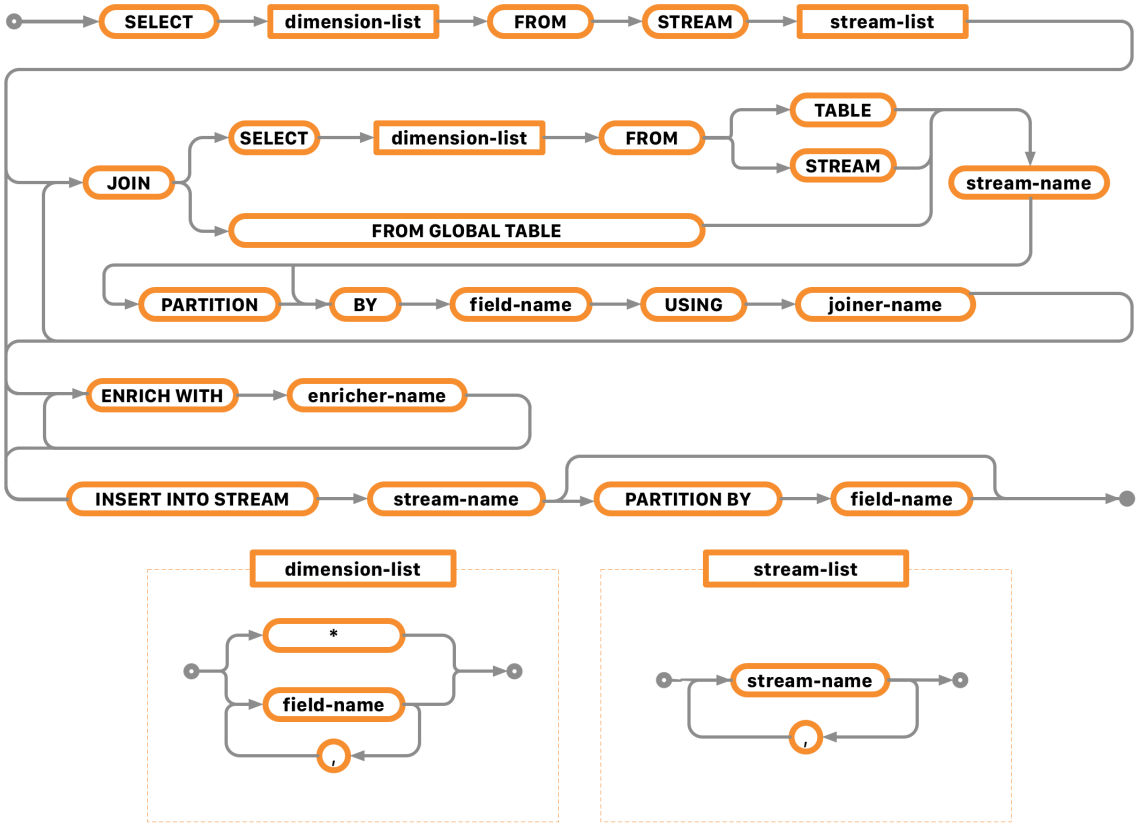
Distribution package

docker

kubernetes

https://wizzie-io.github.io/enricher/

Enrichment Query Language




```
{
  "joiners":[
    {
      "name":"simpleStreamPreferredJoiner",
      "className":"io.wizzie.enricher.enrichment.join.impl.StreamPreferredJoiner"
    }
  ],
  "enrichers": [
    {
      "name": "reputationStreamEnrich",
      "className": "example.ReputationEnrich",
      "properties": {
        "uri":"http://myreputation-service.com",
        "ip_dimension": "src"
      }
    }
  ],
  "queries": {
    "location_reputation": "
      SELECT src, protocol FROM STREAM flow
      JOIN
        SELECT * FROM STREAM location USING simpleStreamPreferredJoiner
      ENRICH WITH reputationStreamEnrich
      INSERT INTO STREAM enrichflow
    "
  }
}
```

Joiners

- Joiners definition. Here you define the relations between joiner name and joiner class.

```
{
  "joiners":[
    {
      "name":"simpleStreamPreferredJoiner",
      "className":"io.wizzie.enricher.enrichment.join.impl.StreamPreferredJoiner"
    }
  ],
  "enrichers": [
    {
      "name": "reputationStreamEnrich",
      "className": "example.ReputationEnrich",
      "properties": {
        "uri":"http://myreputation-service.com",
        "ip_dimension": "src"
      }
    }
  ],
  "queries": {
    "location_reputation": "
      SELECT src, protocol FROM STREAM flow
      JOIN
        SELECT * FROM STREAM location USING simpleStreamPreferredJoiner
      ENRICH WITH reputationStreamEnrich
      INSERT INTO STREAM enrichflow
    "
  }
}
```

Enrichers

- On this section, you define the enrichers with their properties.

```
{
  "joiners":[
    {
      "name":"simpleStreamPreferredJoiner",
      "className":"io.wizzie.enricher.enrichment.join.impl.StreamPreferredJoiner"
    }
  ],
  "enrichers": [
    {
      "name": "reputationStreamEnrich",
      "className": "example.ReputationEnrich",
      "properties": {
        "uri":"http://myreputation-service.com",
        "ip_dimension": "src"
      }
    }
  ],
  "queries": {
    "location_reputation": "
      SELECT src, protocol FROM STREAM flow
      JOIN
        SELECT * FROM STREAM location USING simpleStreamPreferredJoiner
      ENRICH WITH reputationStreamEnrich
      INSERT INTO STREAM enrichflow
    "
  }
}
```

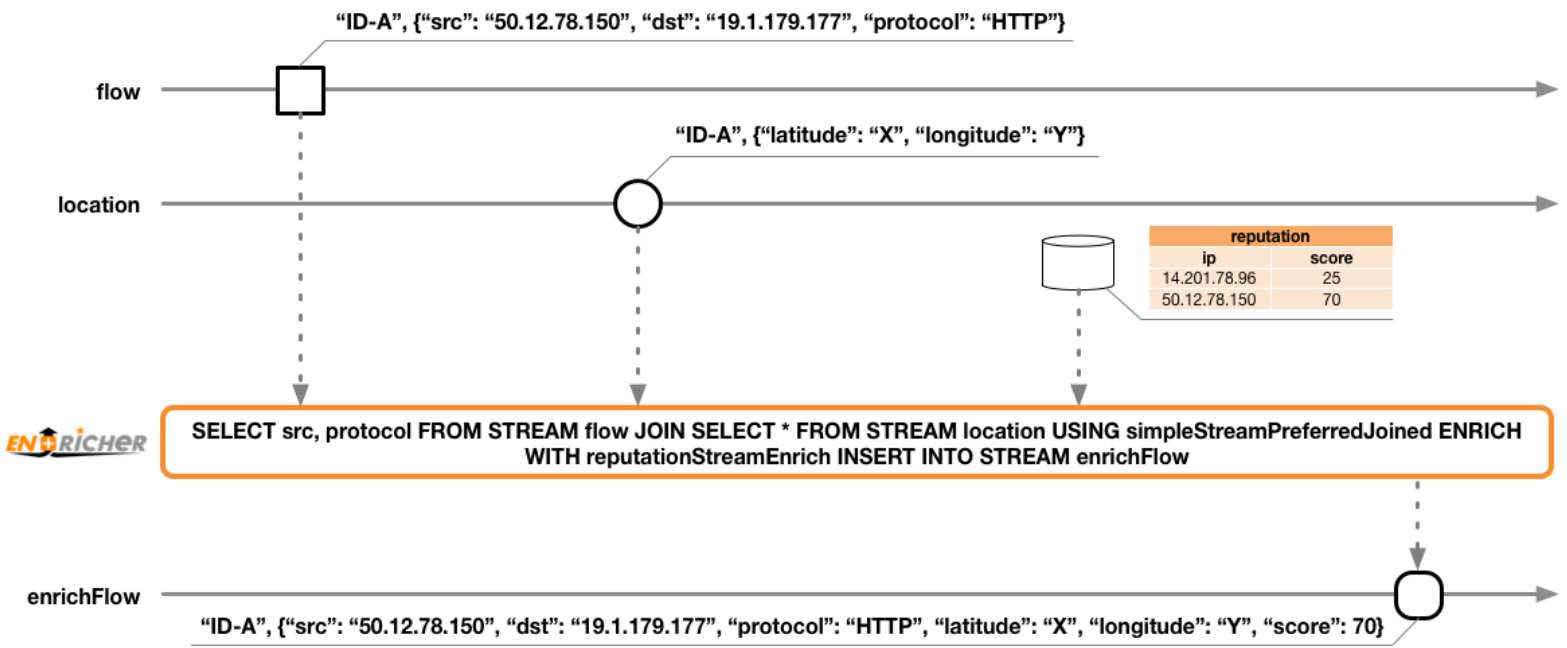
Queries

- At the queries section, you can define multiples queries using EQL.

```
{
  "joiners":[
    {
      "name":"simpleStreamPreferredJoiner",
      "className":"io.wizzie.enricher.enrichment.join.impl.StreamPreferredJoiner"
    }
  ],
  "enrichers": [
    {
      "name": "reputationStreamEnrich",
      "className": "example.ReputationEnrich",
      "properties": {
        "uri":"http://myreputation-service.com",
        "ip_dimension": "src"
      }
    }
  ],
  "queries": {
    "location_reputation": "
      SELECT src, protocol FROM STREAM flow
      JOIN
        SELECT * FROM STREAM location USING simpleStreamPreferredJoiner
      ENRICH WITH reputationStreamEnrich
      INSERT INTO STREAM enrichflow
    "
  }
}
```

ENRICHER

Example



https://wizzie-io.github.io/zz-cep/

Engine

- Streaming processing engine based on Kafka Clients and Siddhi Engine.
- Configuration via JSON files and Siddhi SQL.
- Allows to work with:

https://wizzie-io.github.io/zz-cep/

Engine

- Streaming processing engine based on Kafka Clients and Siddhi Engine.
- Configuration via JSON files and Siddhi SQL.
- Allows to work with:
  - Projection & Filters
  - Windowing
  - Sequences / Patterns
  - Grouping
  - Aggregations

https://wizzie-io.github.io/zz-cep/

Engine

- Streaming processing engine based on Kafka Clients and Siddhi Engine.
- Configuration via JSON files and Siddhi SQL.
- Allows to work with:
  - Projection & Filters
  - Grouping
  - Windowing
  - Aggregations
  - Sequences / Patterns

https://wizzie-io.github.io/zz-cep/

Deployment Mode

Distribution package

kubernetes


```
{
  "streams": [
    {
      "streamName": "atmStatsStream",
      "attributes": [
        { "name": "cardNo", "type": "string"},
        { "name": "cardName", "type": "string"},
        { "name": "cardMobile", "type": "string"},
        { "name": "amountWithdrawed", "type": "long"},
        { "name": "location", "type": "string"}
      ]
    }
  ],
  "rules": [
    {
      "id": "1",
      "version": "v1",
      "streams": {
        "in": [
          {
            "streamName": "atmStatsStream",
            "kafkaTopic": "kafka_topic_in"
          }
        ],
        "out": [
          {
            "streamName": "possibleFraudStream",
            "kafkaTopic": "kafka_topic_out"
          }
        ]
      },
      "executionPlan":
      "
      from every a1 = atmStatsStream[amountWithdrawed < 100]
          -> b1 = atmStatsStream[amountWithdrawed > 10000 and a1.cardNo == b1.cardNo]
          within 1 day
      select
        a1.cardNo as cardNo, a1.cardName as cardName,
        b1.amountWithdrawed as amountWithdrawed, b1.location as location,
        b1.cardMobile as cardMobile
      insert into possibleFraudStream;
      "
    }
  ]
}
```

Streams

- On this section, you can define the deferents streams with their attributes.

```
{
  "streams": [
    {
      "streamName": "atmStatsStream",
      "attributes": [
        { "name": "cardNo", "type": "string"},
        { "name": "cardName", "type": "string"},
        { "name": "cardMobile", "type": "string"},
        { "name": "amountWithdrawed", "type": "long"},
        { "name": "location", "type": "string"}
      ]
    }
  ]
  "rules": [
    {
      "id": "1",
      "version": "v1",
      "streams": {
        "in": [
          {
            "streamName": "atmStatsStream",
            "kafkaTopic": "kafka_topic_in"
          }
        ],
        "out": [
          {
            "streamName": "possibleFraudStream",
            "kafkaTopic": "kafka_topic_out"
          }
        ]
      },
      "executionPlan":
      "
      from every a1 = atmStatsStream[amountWithdrawed < 100]
          -> b1 = atmStatsStream[amountWithdrawed > 10000 and a1.cardNo == b1.cardNo]
          within 1 day
      select
        a1.cardNo as cardNo, a1.cardName as cardName,
        b1.amountWithdrawed as amountWithdrawed, b1.location as location,
        b1.cardMobile as cardMobile
      insert into possibleFraudStream;
      "
    }
  ]
}
```

Rules Mapping

- This section configures the ruleID, the ruleVersion and the relations between kafka and siddhiStreams.

```
{
  "streams": [
    {
      "streamName": "atmStatsStream",
      "attributes": [
        { "name": "cardNo", "type": "string"},
        { "name": "cardName", "type": "string"},
        { "name": "cardMobile", "type": "string"},
        { "name": "amountWithdrawed", "type": "long"},
        { "name": "location", "type": "string"}
      ]
    }
  ],
  "rules": [
    {
      "id": "1",
      "version": "v1",
      "streams": {
        "in": [
          {
            "streamName": "atmStatsStream",
            "kafkaTopic": "kafka_topic_in"
          }
        ],
        "out": [
          {
            "streamName": "possibleFraudStream",
            "kafkaTopic": "kafka_topic_out"
          }
        ]
      },
      "executionPlan":
      "
      from every a1 = atmStatsStream[amountWithdrawed < 100]
          -> b1 = atmStatsStream[amountWithdrawed > 10000 and a1.cardNo == b1.cardNo]
          within 1 day
      select
        a1.cardNo as cardNo, a1.cardName as cardName,
        b1.amountWithdrawed as amountWithdrawed, b1.location as location,
        b1.cardMobile as cardMobile
      insert into possibleFraudStream;
      "
    }
  ]
}
```

Execution Plan

- The execution plan is the business logic defined by Siddhi Streaming SQL.

```
{
  "streams": [
    {
      "streamName": "atmStatsStream",
      "attributes": [
        { "name": "cardNo", "type": "string"},
        { "name": "cardName", "type": "string"},
        { "name": "cardMobile", "type": "string"},
        { "name": "amountWithdrawed", "type": "long"},
        { "name": "location", "type": "string"}
      ]
    }
  ],
  "rules": [
    {
      "id": "1",
      "version": "v1",
      "streams": {
        "in": [
          {
            "streamName": "atmStatsStream",
            "kafkaTopic": "kafka_topic_in"
          }
        ],
        "out": [
          {
            "streamName": "possibleFraudStream",
            "kafkaTopic": "kafka_topic_out"
          }
        ]
      },
      "executionPlan":
      "
      from every a1 = atmStatsStream[amountWithdrawed < 100]
          -> b1 = atmStatsStream[amountWithdrawed > 10000 and a1.cardNo == b1.cardNo]
          within 1 day
      select
        a1.cardNo as cardNo, a1.cardName as cardName,
        b1.amountWithdrawed as amountWithdrawed, b1.location as location,
        b1.cardMobile as cardMobile
      insert into possibleFraudStream;
      "
    }
  ]
}
```

ZZ CEP

Example: ATM Transaction frauds

1. Processes the events received through the *atmStatsStream*.
2. Looks for an event a1 where the *amountWithdrawn* is less than 100.
3. Looks for another event b1 where the card number is the same as previous and *amountWithdrawn* is greater than 10000. The within keyword ensures that this condition should be satisfied within a day.
4. When the pattern condition is satisfied, selects the attribute and renames them.
5. Inserts the output events to the *possibleFraudStream*.

```
{
  "streams": [
    {
      "streamName": "atmStatsStream",
      "attributes": [
        { "name": "cardNo", "type": "string"},
        { "name": "cardName", "type": "string"},
        { "name": "cardMobile", "type": "string"},
        { "name": "amountWithdrawed", "type": "long"},
        { "name": "location", "type": "string"}
      ]
    }
  ],
  "rules": [
    {
      "id": "1",
      "version": "v1",
      "streams": {
        "in": [
          {
            "streamName": "atmStatsStream",
            "kafkaTopic": "kafka_topic_in"
          }
        ],
        "out": [
          {
            "streamName": "possibleFraudStream",
            "kafkaTopic": "kafka_topic_out"
          }
```

```
"
from every a1 = atmStatsStream[amountWithdrawed < 100]
    -> b1 = atmStatsStream[amountWithdrawed > 10000 and a1.cardNo == b1.cardNo]
    within 1 day
select
  a1.cardNo as cardNo, a1.cardrName as cardName,
  b1.amountWithdrawed as amountWithdrawed, b1.location as location,
  b1.cardMobile as cardMobile
insert into possibleFraudStream;
"
```

```
}
```

Druid

http://druid.io

Druid

Druid

- OLAP & Timeseries database
- Column-oriented storage
- Streaming and batch ingestion
- Flexible schemas
- Horizontally scalable
- REST JSON API & SQL support

http://druid.io

WIZZ-VIS

https://wizzie-io.github.io/wizz-vis/

WIZZ-VIS

- Analytics platform for time series metrics using Druid.
- Configuration via JSON REST API.

- **Features**
  - TopN, timeseries, groupBy, filters, aggregations
  - Time selector
  - Compare mode
  - Drill downs
  - Thresholds
- **Widgets**
  - Series, MultiSeries
  - Bars, Pies
  - Values , Gauges, Tables
  - Heatmaps, Planes, Routes
  - Chord, Sankey
  - Text, Images

https://wizzie-io.github.io/wizz-vis/

WIZZ-VIS

- Analytics platform for time series metrics using Druid.
- Configuration via JSON REST API.

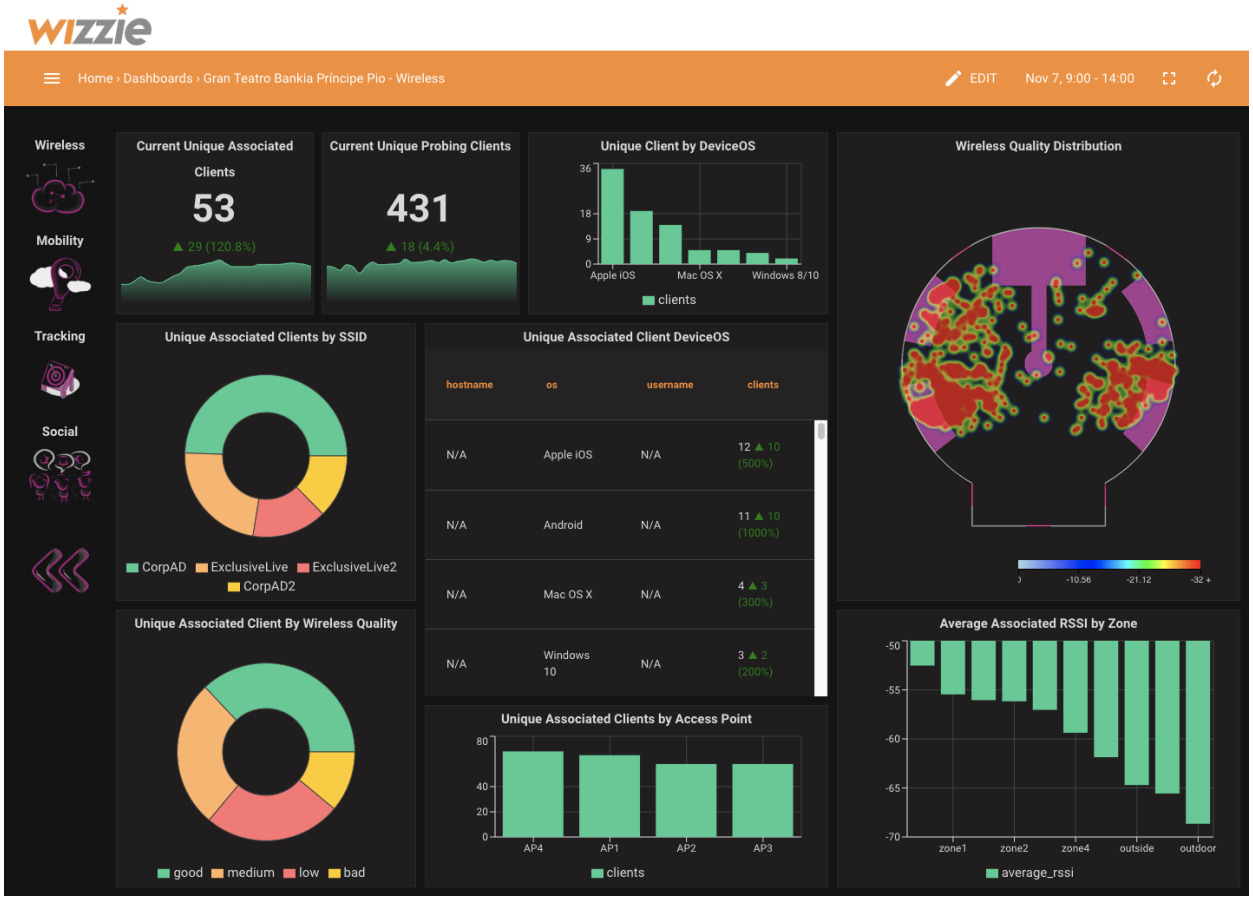
- **Features**
  - TopN, timeseries, groupBy, filters, aggregations
  - Time selector
  - Compare mode
  - Drill downs
  - Thresholds
- **Widgets**
  - Series, MultiSeries
  - Bars, Pies
  - Values , Gauges, Tables
  - Heatmaps, Planes, Routes
  - Chord, Sankey
  - Text, Images

https://wizzie-io.github.io/wizz-vis/

Deployment Mode

WIZZ-VIS

Dashboard



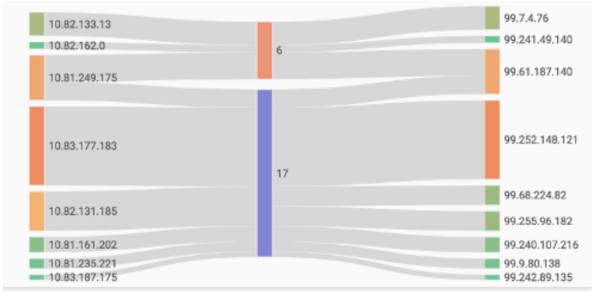
wizzie

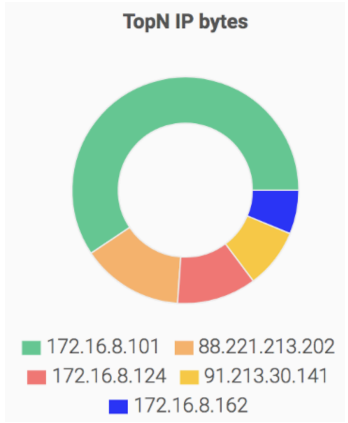
WIZZ-VIS

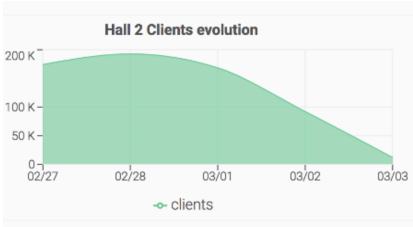
Widgets

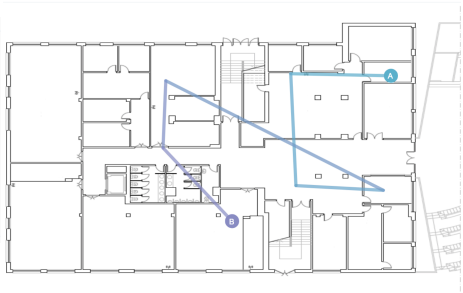
Widgets

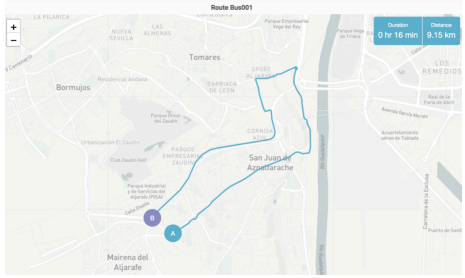

Avg Clients Last 7 days
75.80
▲ 3.18 (4.2%)

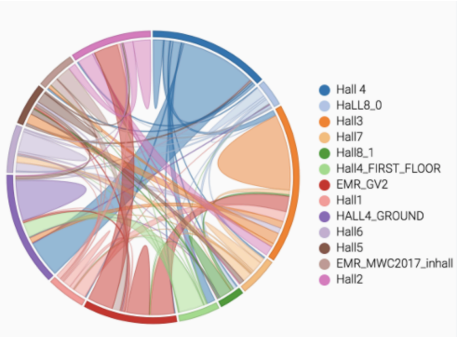

10.82.133.13
10.82.162.0
10.81.249.175
10.83.177.183
10.82.131.185
10.81.161.202
10.81.235.221
10.83.187.175
6
17
99.7.4.76
99.241.49.140
99.61.187.140
99.252.148.121
99.68.224.82
99.255.96.182
99.240.107.216
99.9.80.138
99.242.89.135

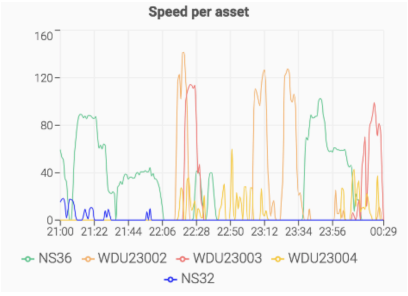

TopN IP bytes
172.16.8.101
88.221.213.202
172.16.8.124
91.213.30.141
172.16.8.162

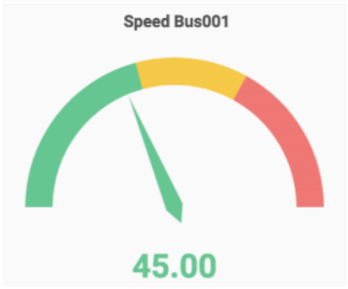

Hall 2 Clients evolution
200 K
100 K
50 K
0
02/27
02/28
03/01
03/02
03/03
clients



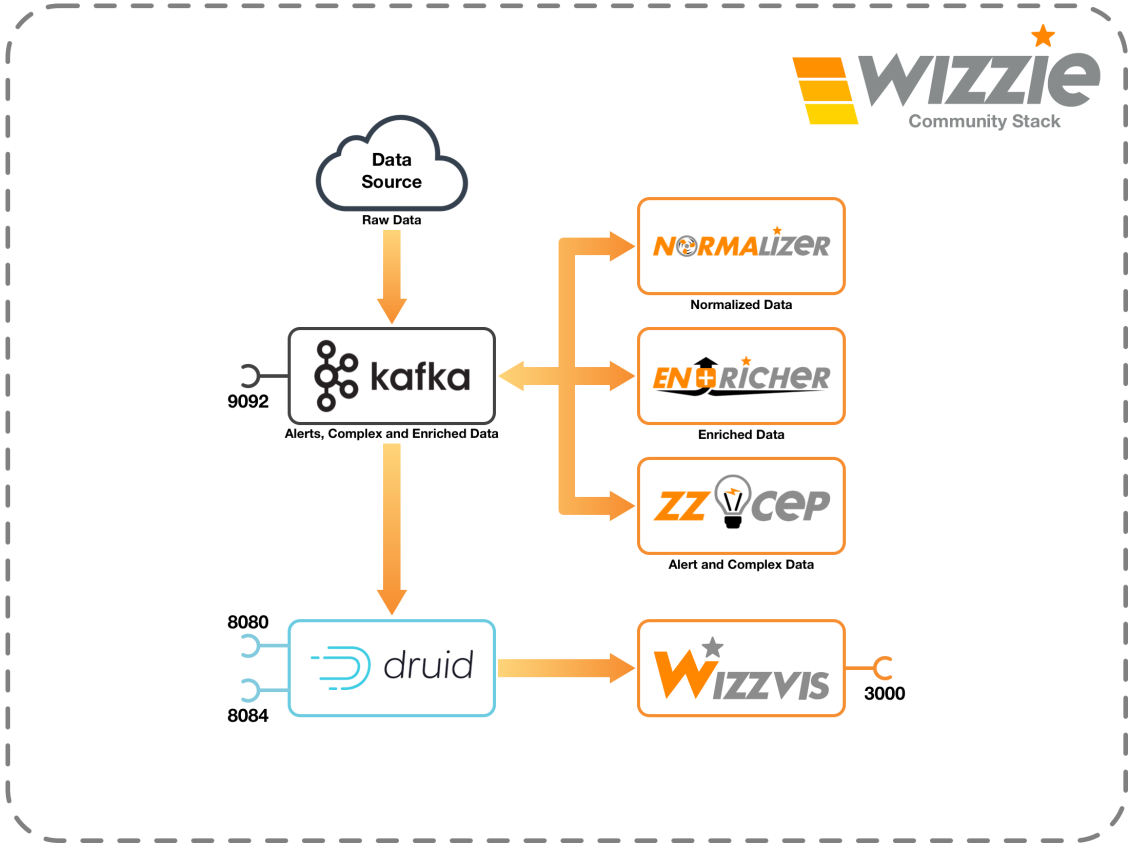

Route Bus001
Duration
0 hr 16 min
Distance
9.15 km
Tomares
Bormujos
San Juan de Aznalfarache
Mairena del Aljarafe


Hall 4
HaLL8_0
Hall3
Hall7
Hall8_1
Hall4_FIRST_FLOOR
EMR_GV2
Hall1
HALL4_GROUND
Hall6
Hall5
EMR_MWC2017_inhall
Hall2

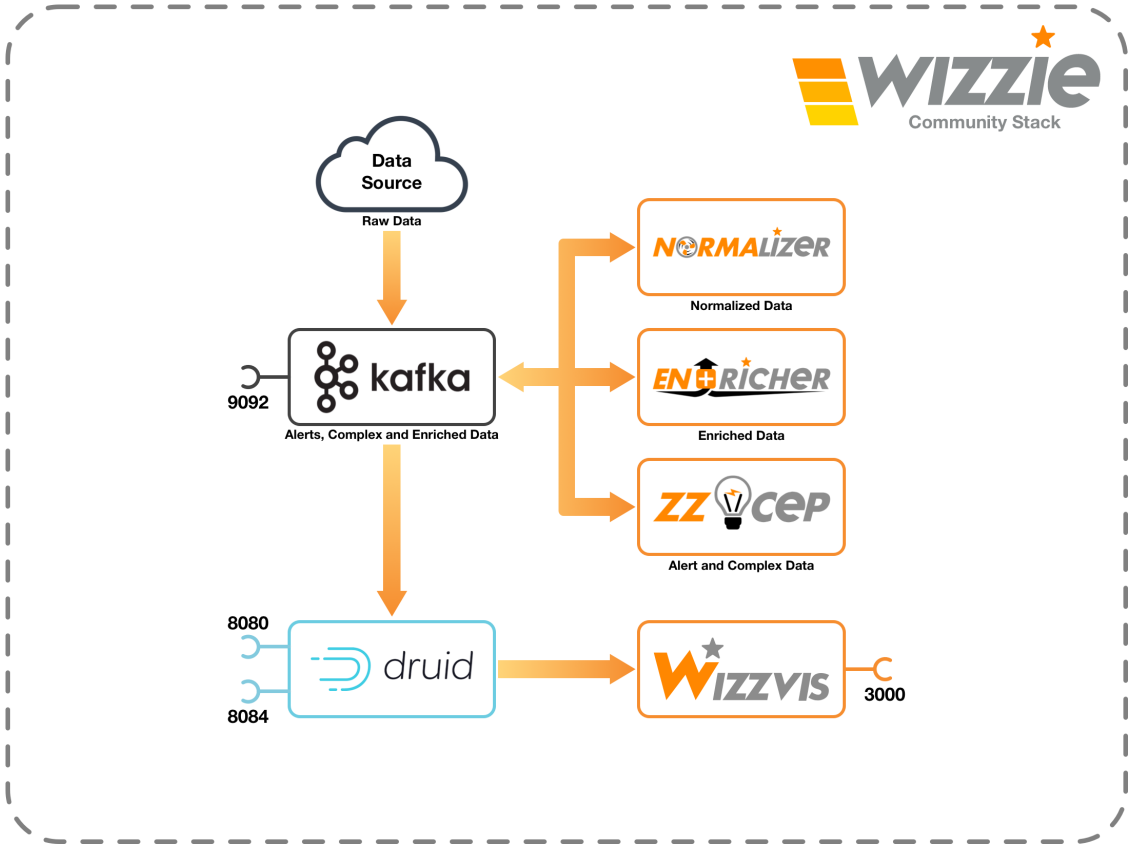

Speed per asset
160
120
80
40
0
21:00
21:22
21:44
22:06
22:28
22:50
23:12
23:34
23:56
00:29
NS36
WDU23002
WDU23003
WDU23004
NS32


Speed Bus001
45.00


Clicks por anunciante
1.40 K
700
350
0
Plus500
Hyundai
Wanda
Mahou
clicks



https://github.com/wizzie-io/community-stack



Try it, now!

Docker Compose

```
sudo bash -c "$(curl -L --header 'Accept: application/vnd.github.v3.raw' 'https://api.github.com/repos/wizzie-io/community-stack/contents/setups/linux_setup.sh?ref=1.0.0')"
```

https://github.com/wizzie-io/community-stack

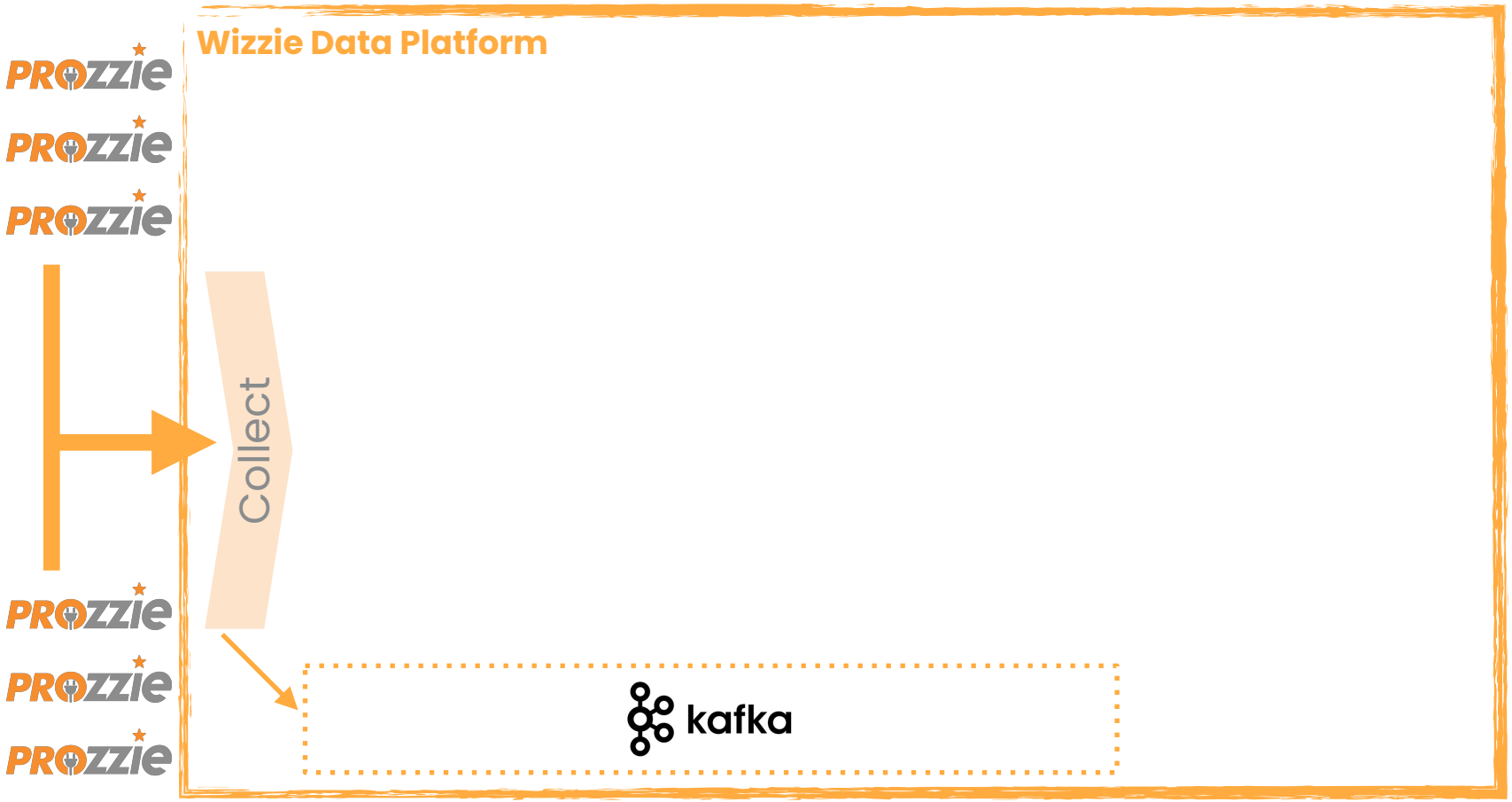

aws

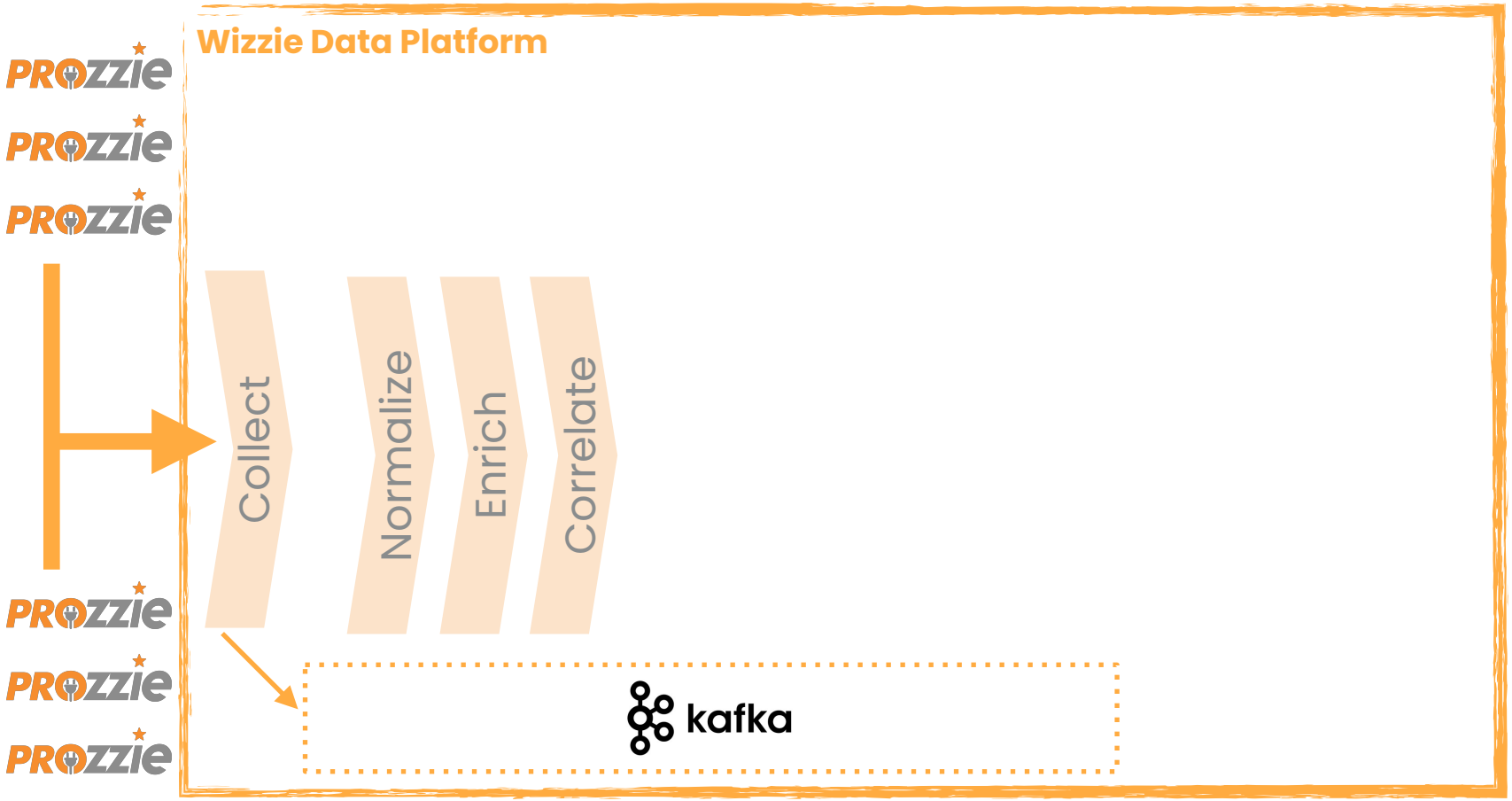

vmware

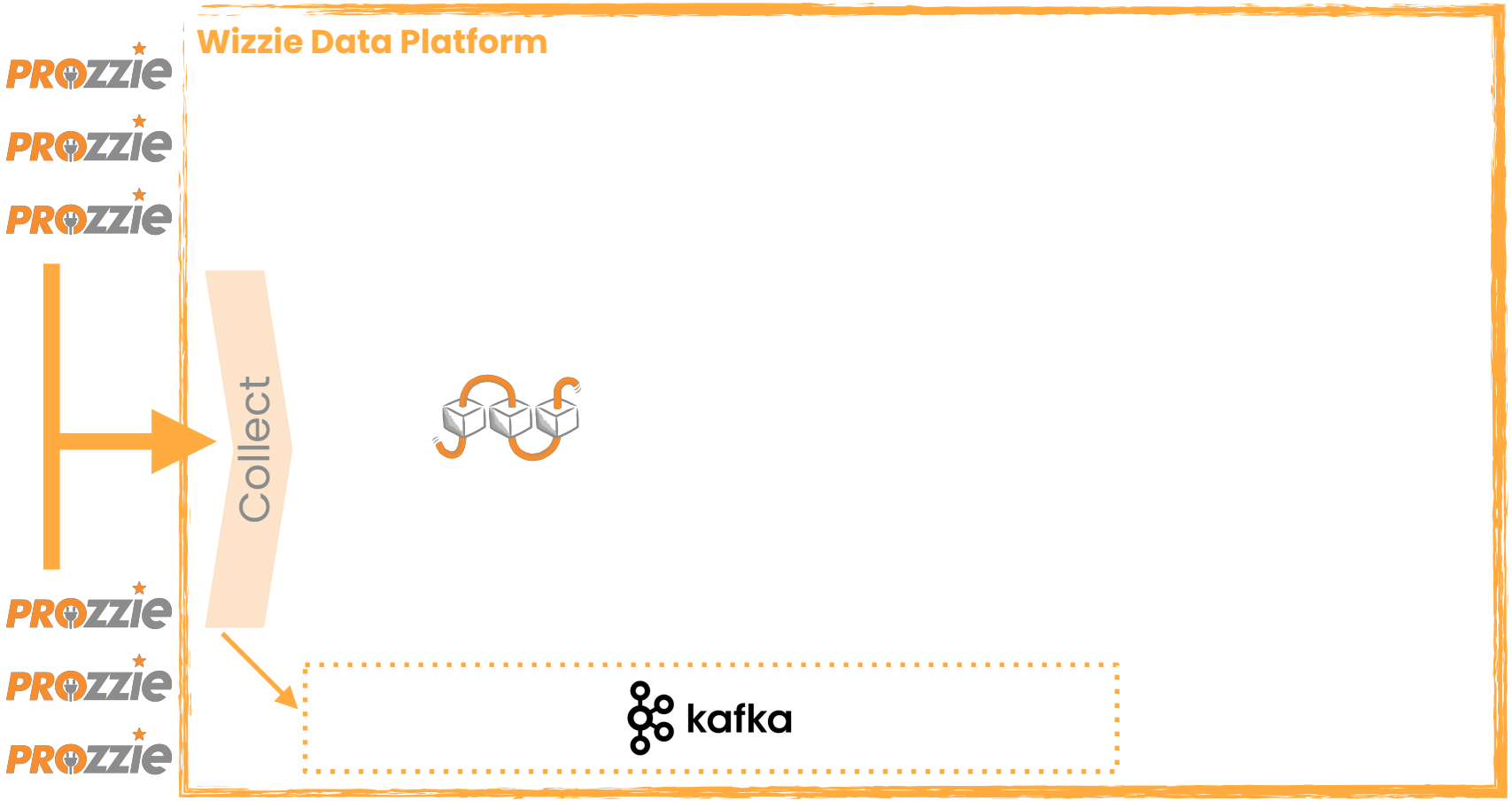

aws

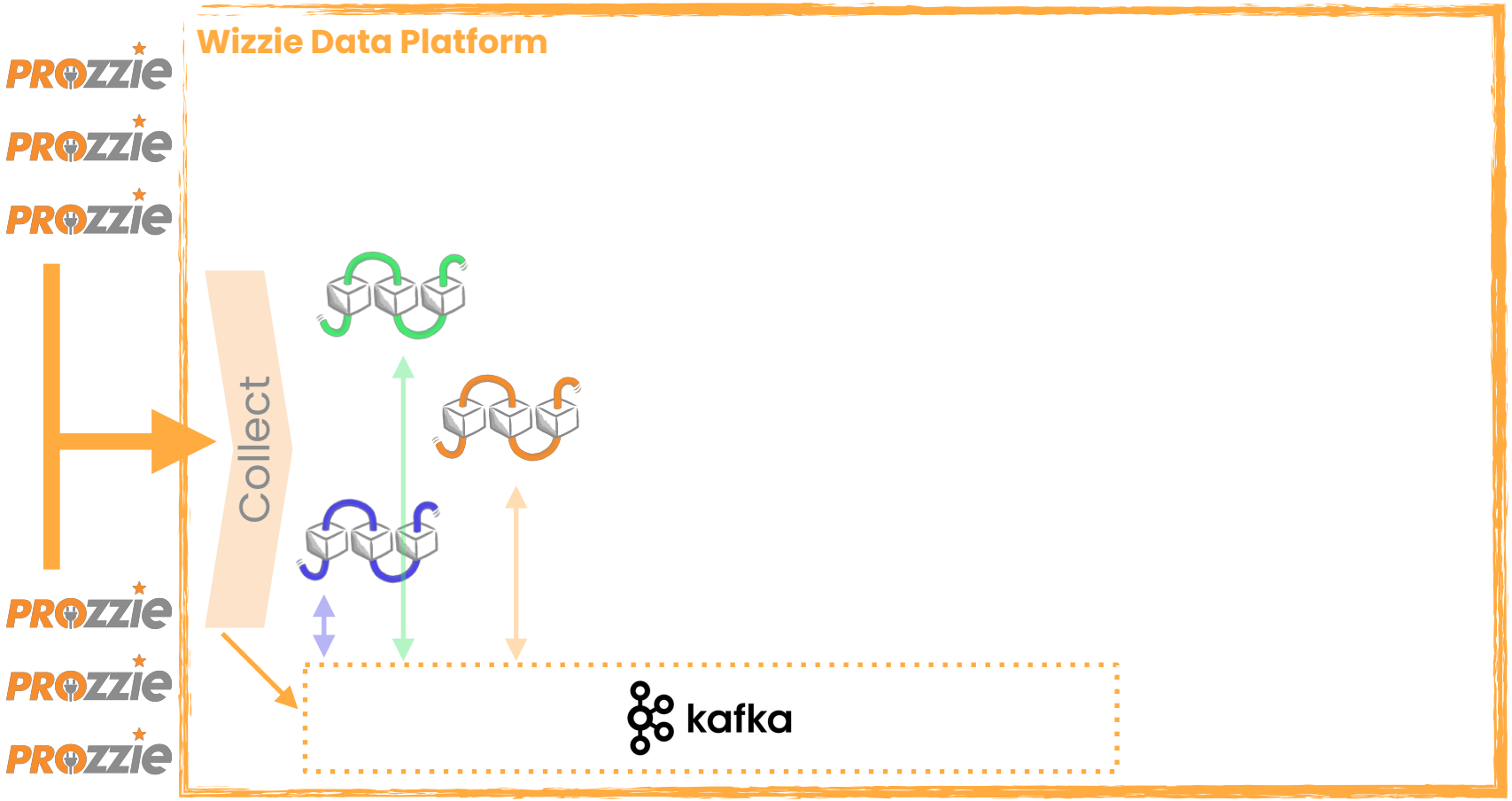

vmware

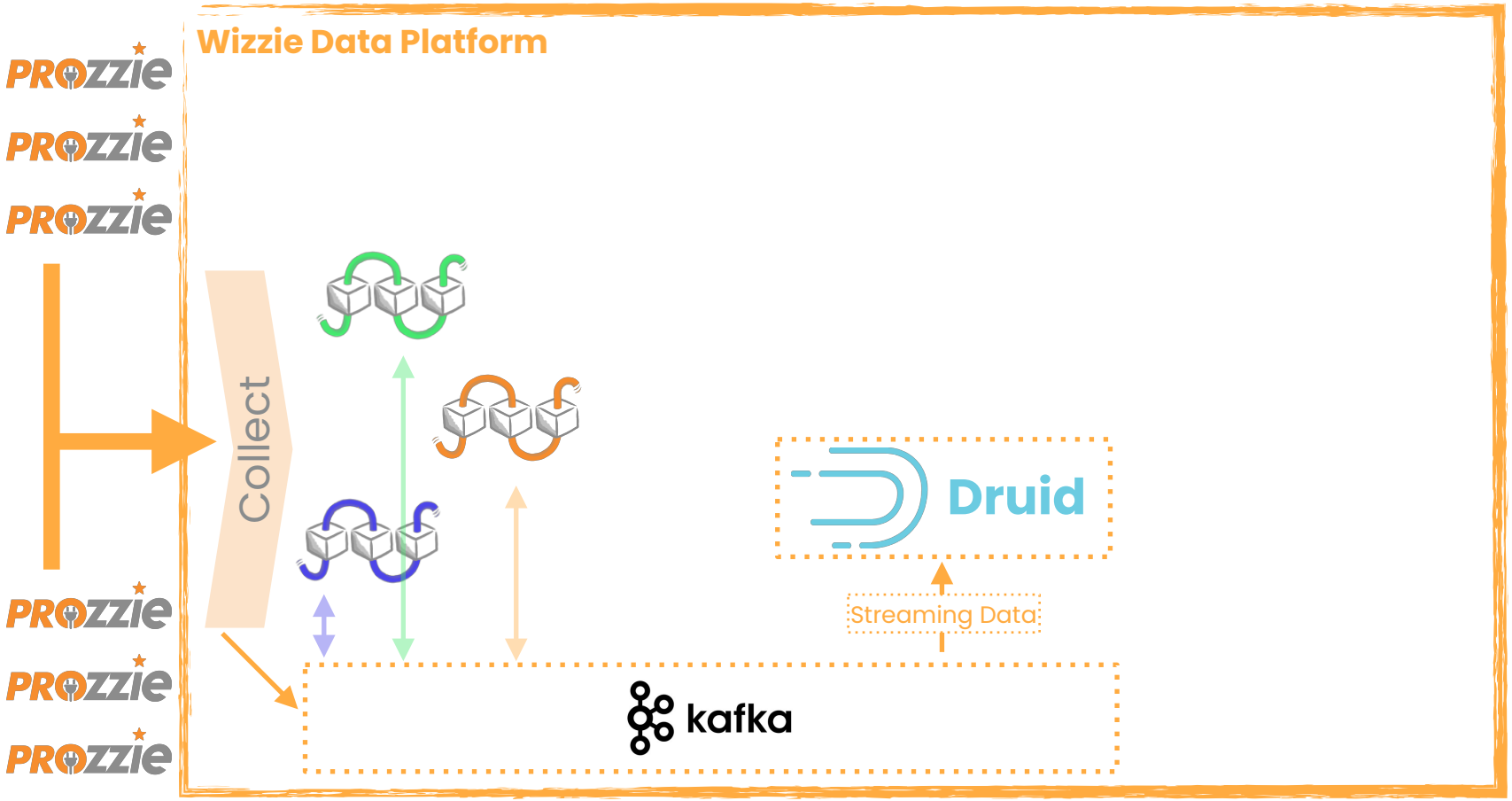
Wizzie Data Platform

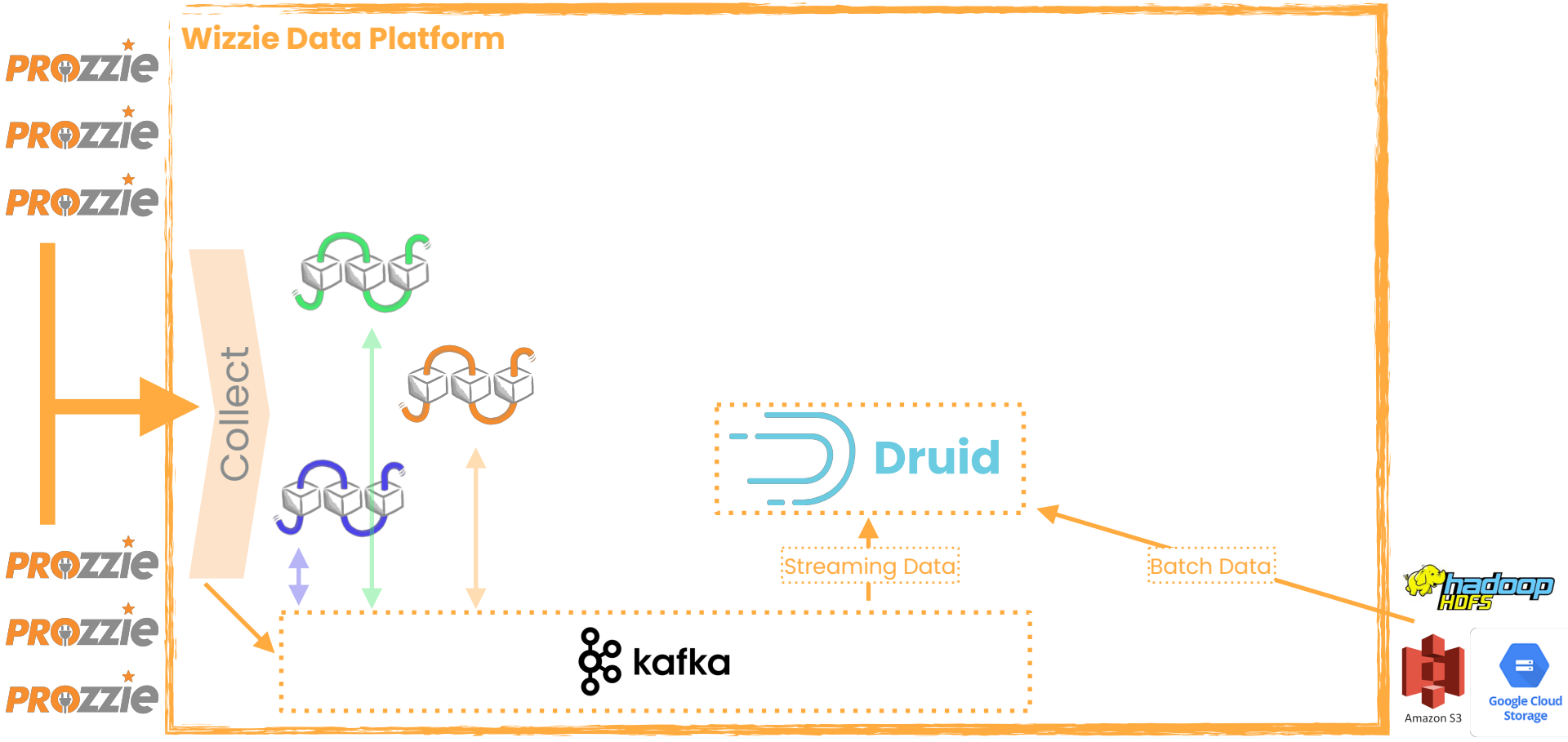

Wizzie Data Platform
PROZZIE
PROZZIE
PROZZIE
Collect
PROZZIE
PROZZIE
PROZZIE
kafka

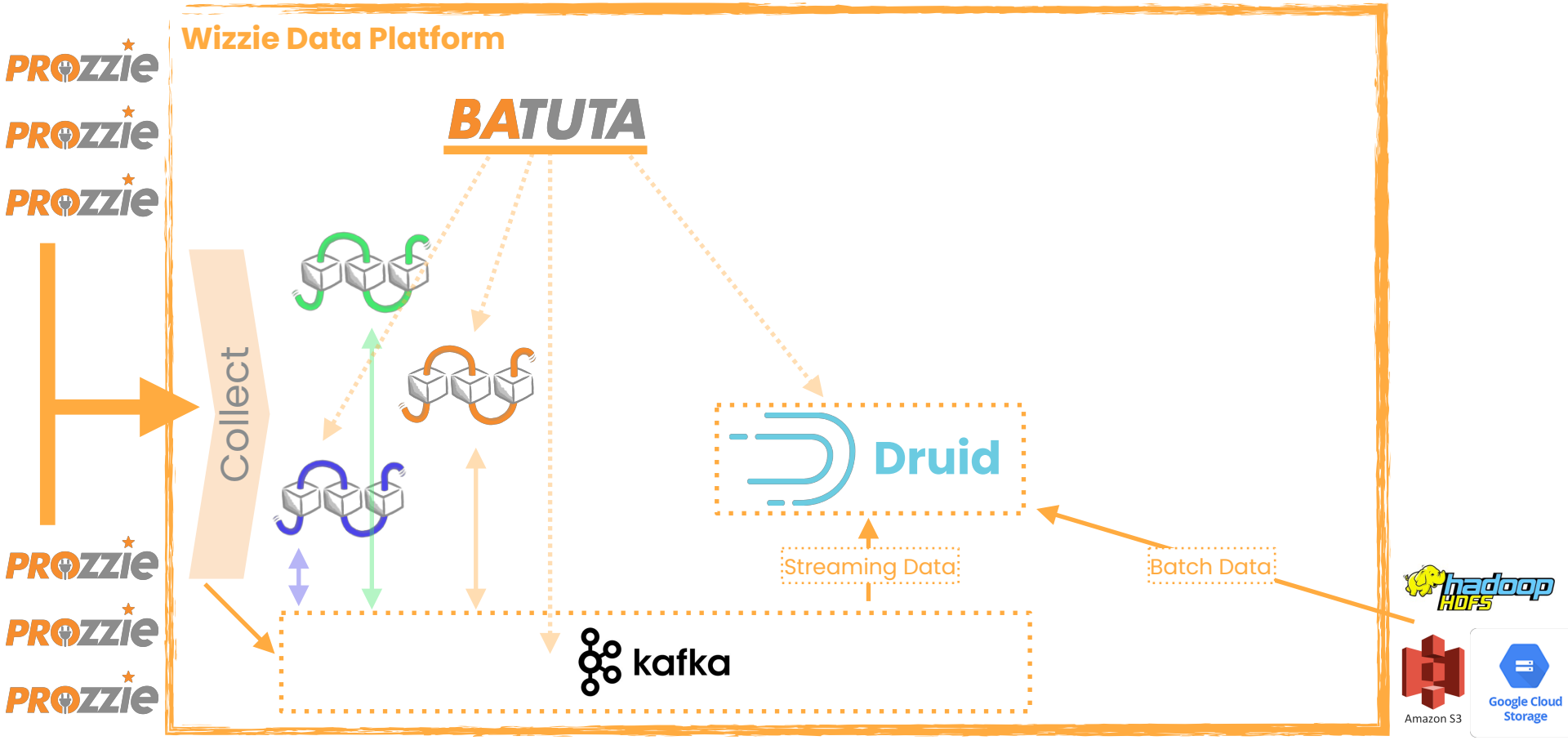

Wizzie Data Platform
PROZZIE
PROZZIE
PROZZIE
Collect
Normalize
Enrich
Correlate
PROZZIE
PROZZIE
PROZZIE
kafka


Wizzie Data Platform
PROZZIE
PROZZIE
PROZZIE
Collect
PROZZIE
PROZZIE
PROZZIE
kafka

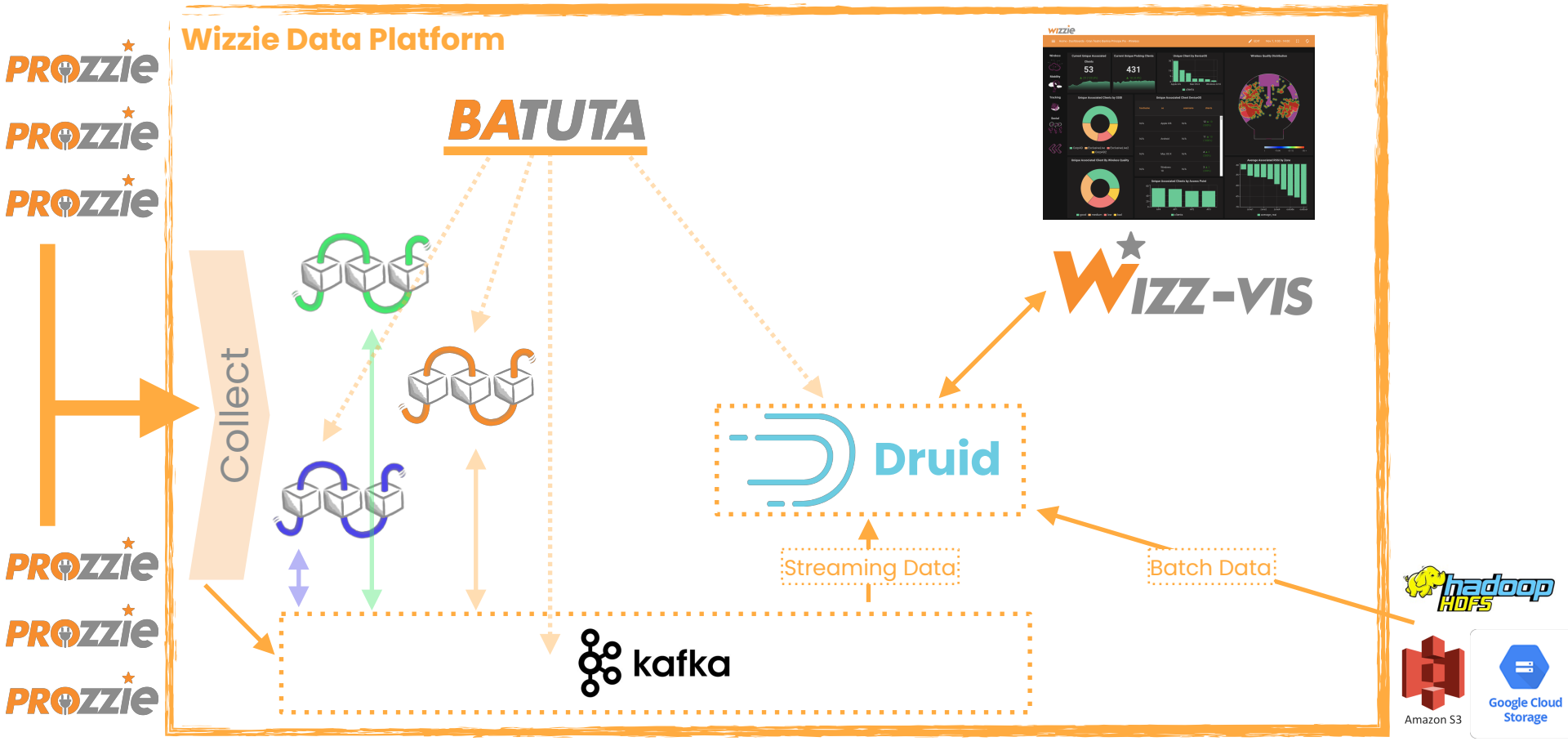

Wizzie Data Platform
PROZZIE
PROZZIE
PROZZIE
Collect
PROZZIE
PROZZIE
PROZZIE
kafka

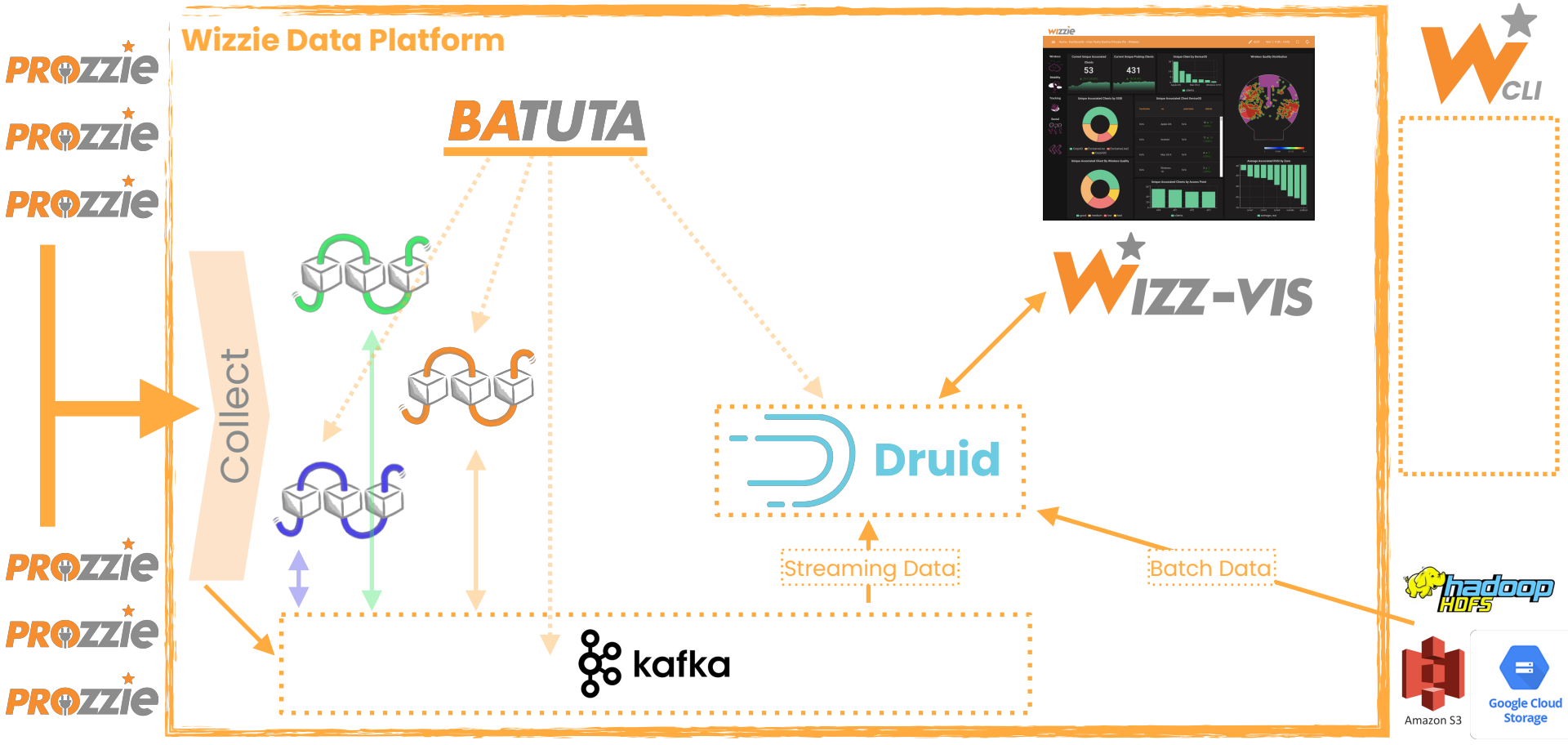

Wizzie Data Platform
PROZZIE
PROZZIE
PROZZIE
Collect
Druid
Streaming Data
kafka
PROZZIE
PROZZIE
PROZZIE

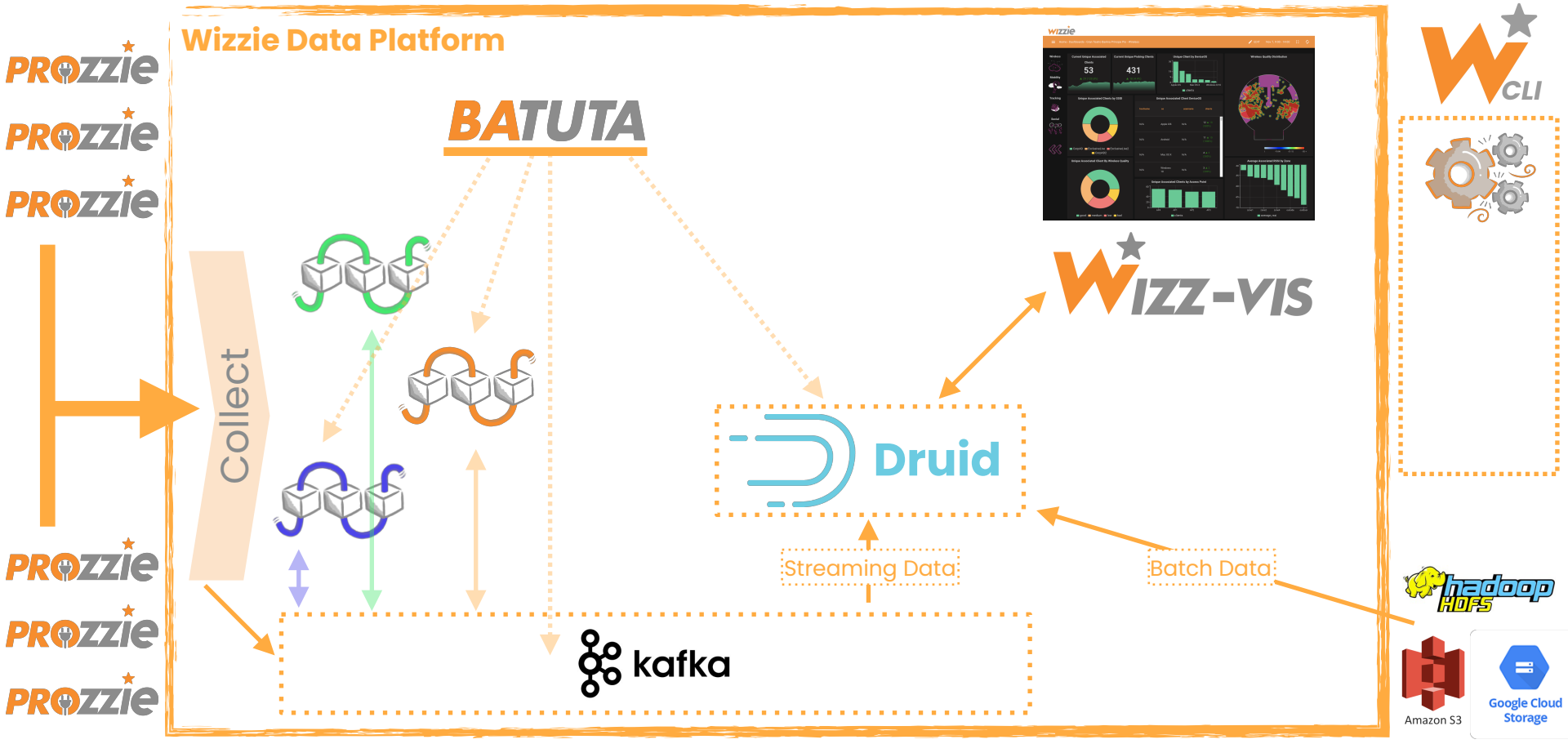

Wizzie Data Platform
PROZZIE
PROZZIE
PROZZIE
Collect
Druid
Streaming Data
Batch Data
kafka
PROZZIE
PROZZIE
PROZZIE
hadoop HDFS
Amazon S3
Google Cloud Storage

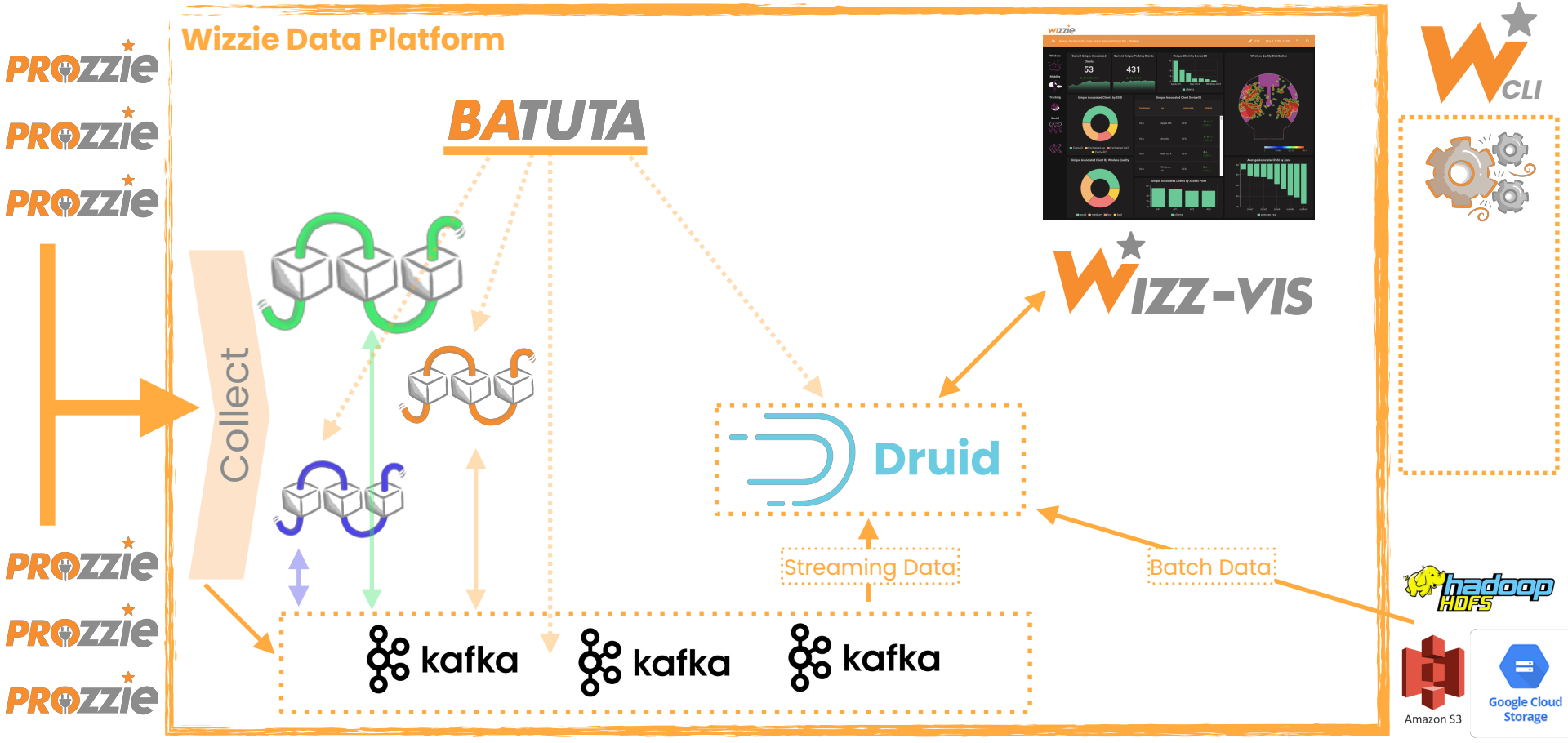

Wizzie Data Platform
PROZZIE
PROZZIE
PROZZIE
BATUTA
Collect
Druid
Streaming Data
Batch Data
kafka
PROZZIE
PROZZIE
PROZZIE
hadoop HDFS
Amazon S3
Google Cloud Storage

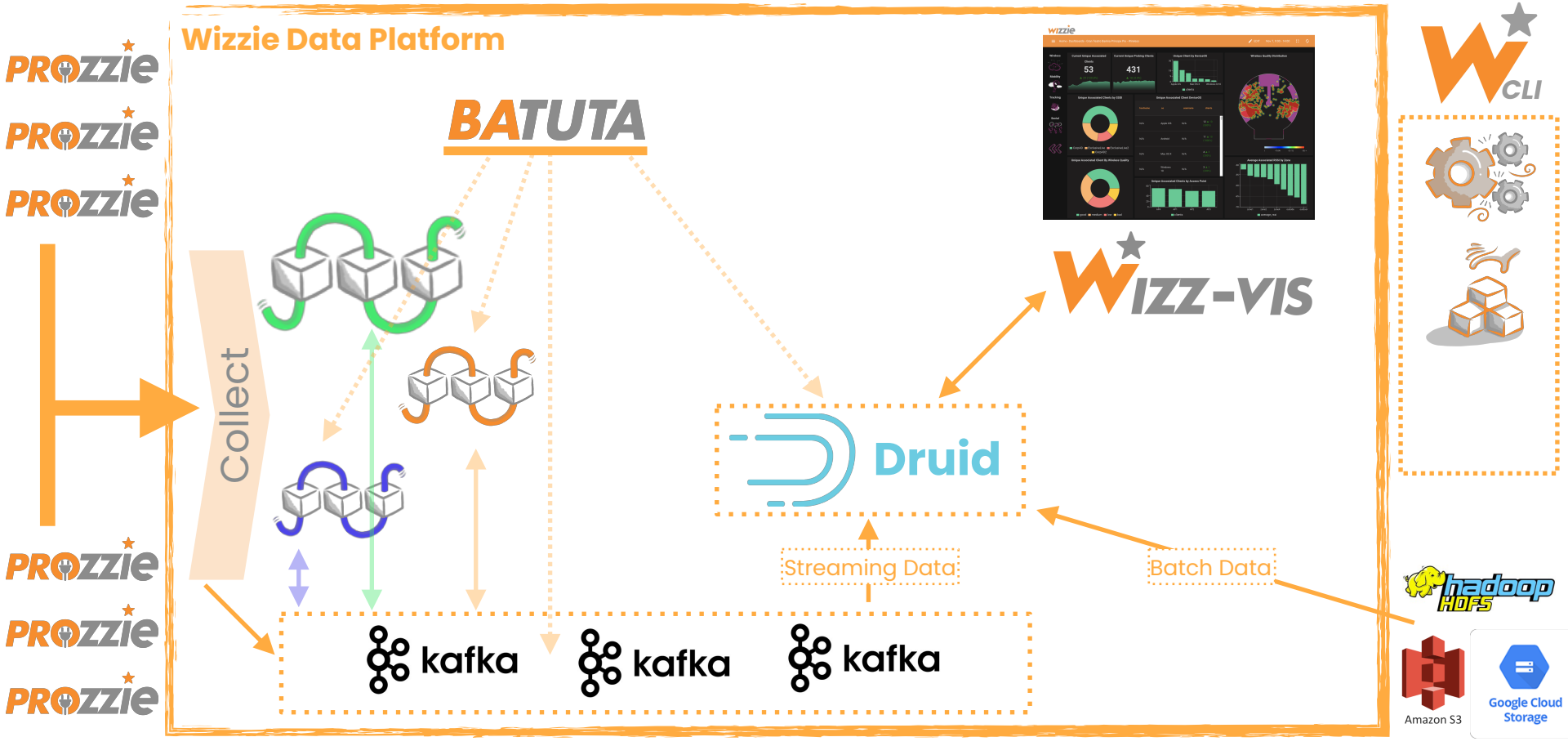
wizzie

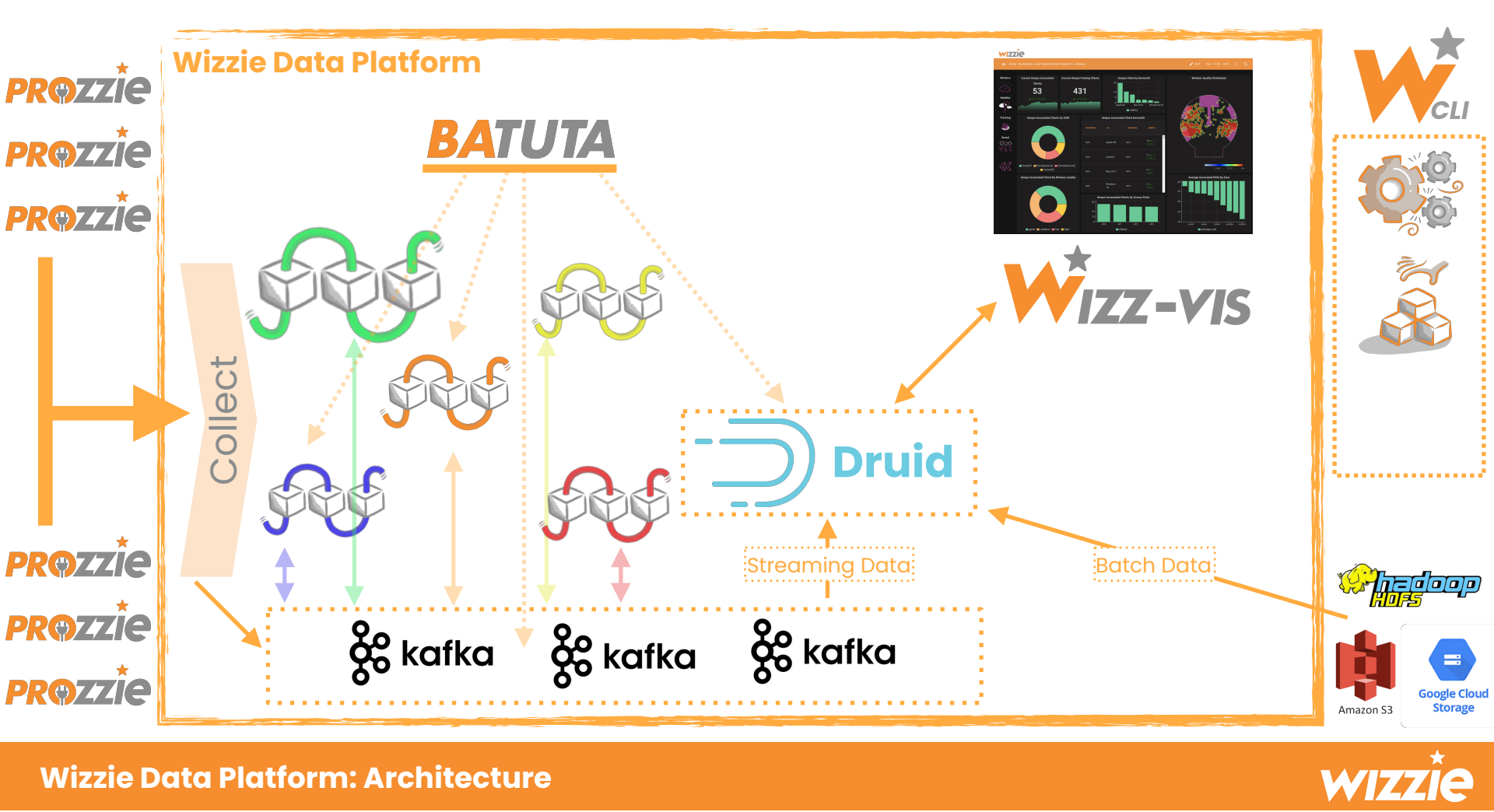

Wizzie Data Platform
PROZZIE
PROZZIE
PROZZIE
PROZZIE
PROZZIE
PROZZIE
BATUTA
Collect
WIZZ-VIS
Druid
Streaming Data
Batch Data
kafka
hadoop HDFS
Amazon S3
Google Cloud Storage

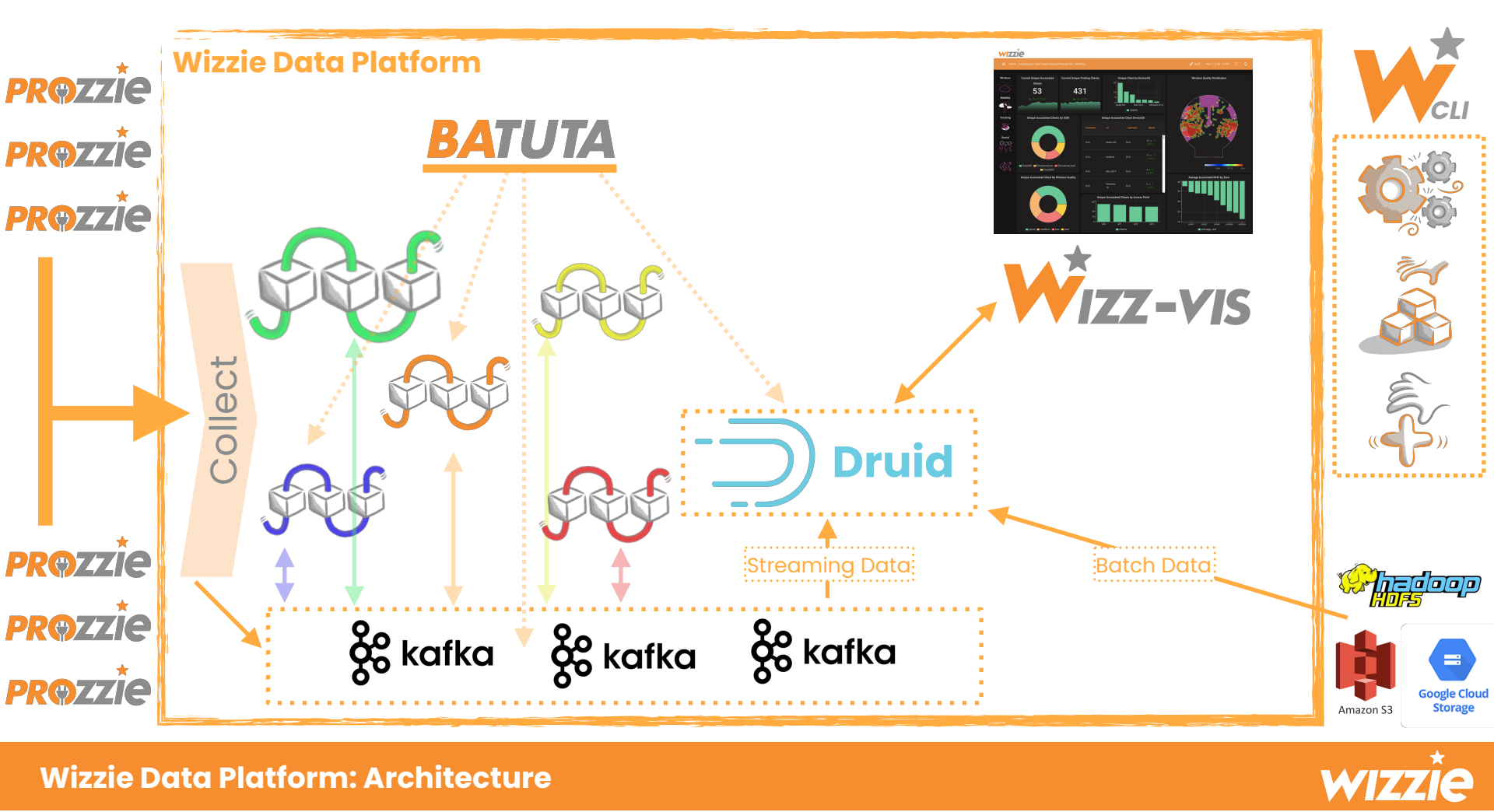

Wizzie Data Platform
PROZZIE
PROZZIE
PROZZIE
PROZZIE
PROZZIE
PROZZIE
BATUTA
Collect
WIZZ-VIS
WCLI
Druid
Streaming Data
Batch Data
kafka
hadoop HDFS
Amazon S3
Google Cloud Storage

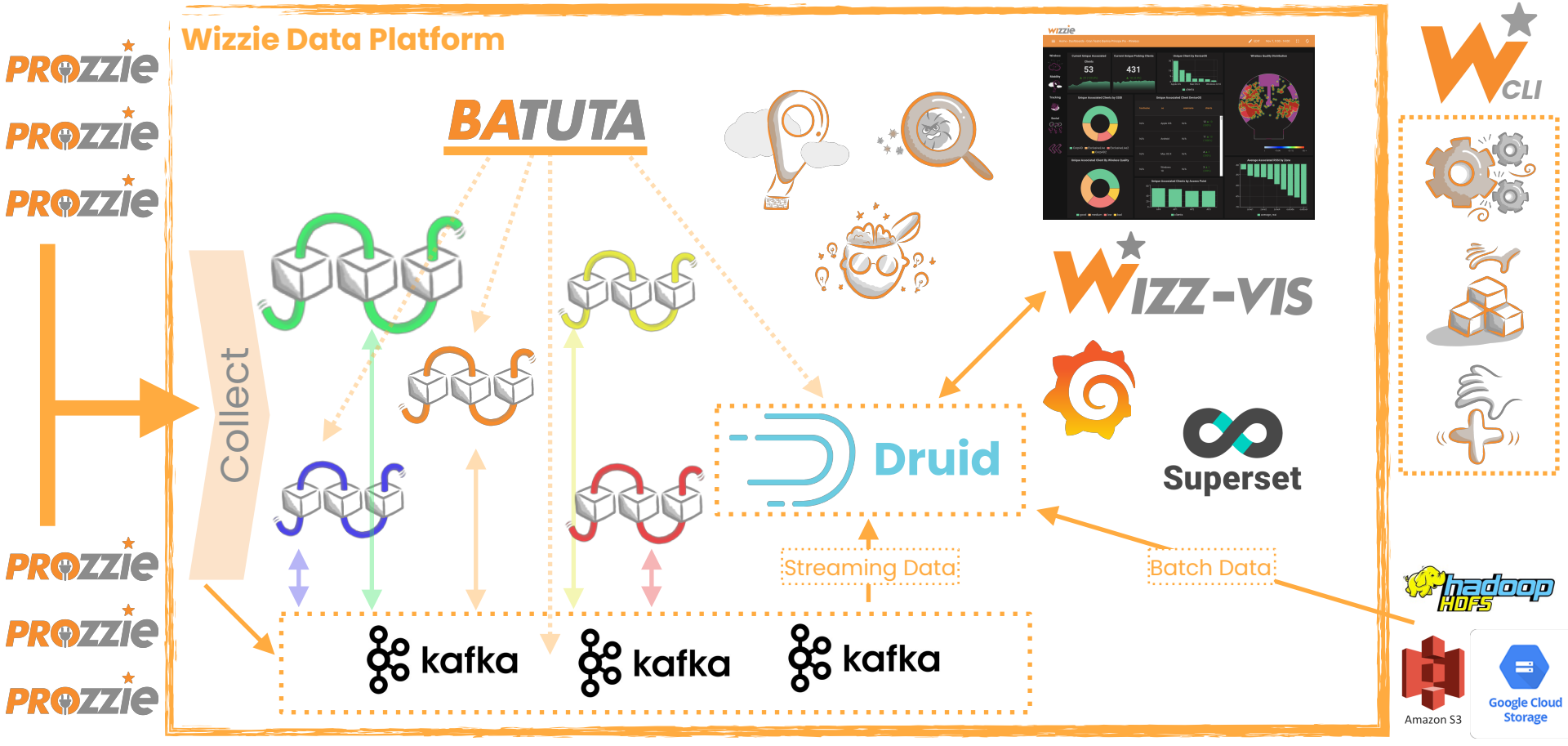

Wizzie Data Platform
PROZZIE
PROZZIE
PROZZIE
BATUTA
Collect
WIZZ-VIS
Druid
Streaming Data
Batch Data
kafka
WCLI
hadoop HDFS
Amazon S3
Google Cloud Storage
PROZZIE
PROZZIE
PROZZIE


Wizzie Data Platform
PROZZIE
PROZZIE
PROZZIE
PROZZIE
PROZZIE
PROZZIE
BATUTA
Collect
WIZZ-VIS
WCLI
Druid
Streaming Data
Batch Data
kafka
kafka
kafka
hadoop HDFS
Amazon S3
Google Cloud Storage


Wizzie Data Platform
PROZZIE
PROZZIE
PROZZIE
BATUTA
Collect
WIZZ-VIS
Druid
Streaming Data
Batch Data
kafka
kafka
kafka
WCLI
hadoop HDFS
Amazon S3
Google Cloud Storage
PROZZIE
PROZZIE
PROZZIE


Wizzie Data Platform
PROZZIE
PROZZIE
PROZZIE
BATUTA
Collect
Druid
WIZZ-VIS
WCLI
Streaming Data
Batch Data
kafka
kafka
kafka
PROZZIE
PROZZIE
PROZZIE
hadoop HDFS
Amazon S3
Google Cloud Storage


Wizzie Data Platform
PROZZIE
PROZZIE
PROZZIE
BATUTA
Collect
WIZZ-VIS
Druid
Streaming Data
Batch Data
kafka
kafka
kafka
WCLI
hadoop HDFS
Amazon S3
Google Cloud Storage
PROZZIE
PROZZIE
PROZZIE


Wizzie Data Platform
PROZZIE
PROZZIE
PROZZIE
BATUTA
Collect
WIZZ-VIS
Druid
Superset
Streaming Data
Batch Data
kafka
kafka
kafka
WCLI
hadoop HDFS
Amazon S3
Google Cloud Storage
PROZZIE
PROZZIE
PROZZIE

Wizzie Data Platform: Requirements

Requirements:

▶ Horizontal scalability, elastic, fault-tolerance, resilience…	
▶ Data with **flexible schema** or schema-less.	
▶ Quickly & easy configuration changes -> **without coding**.	
▶ Easy to integrate new data components.	

Requirements:

▶ Horizontal scalability, elastic, fault-tolerance, resilience…	     
▶ Data with **flexible schema** or schema-less.	
▶ Quickly & easy configuration changes -> **without coding**.	
▶ Easy to integrate new data components.	

Requirements:

▶ Horizontal scalability, elastic, fault-tolerance, resilience…	     
▶ Data with **flexible schema** or schema-less.	  
▶ Quickly & easy configuration changes -> **without coding**.	
▶ Easy to integrate new data components.	

Requirements:

▶ Horizontal scalability, elastic, fault-tolerance, resilience…	     
▶ Data with **flexible schema** or schema-less.	  
▶ Quickly & easy configuration changes -> **without coding**.	   
▶ Easy to integrate new data components.	

Requirements:

▶ Horizontal scalability, elastic, fault-tolerance, resilience…	
▶ Data with **flexible schema** or schema-less.	
▶ Quickly & easy configuration changes -> **without coding**.	
▶ Easy to integrate new data components.	

Andrés Gómez Ferrer
CTO at Wizzie.io

@andresgomezfrr

WIZZIE Community Stack	https://github.com/wizzie-io/community-stack
NORMALIZER	https://wizzie-io.github.io/normalizer/
ENRICHER	https://wizzie-io.github.io/enricher/
ZZ CEP	https://wizzie-io.github.io/zz-cep/
WIZZ-VIS	https://wizzie-io.github.io/wizz-vis/
PROZZIE	https://wizzie-io.github.io/prozzie/