Pragma Analytics Software Suite “PASS”

Time series data solution

matthieu@pragma-innovation.fr

pragmainnovation

Pragma Innovation

- Company Positioning: IT services focused on a « practice »
  - Valuate timeseries oriented data,
  - Analyze, report, decision support tool,
  - Market: ISP, Hosting providers, Telco
- Model: integration of open source avec customization
  - Follows big data reference model,
  - Evaluate, choose, design ideal open source software,
  - Consultancy,
- 20 years of expertise and track of records
  - Complex solution design,
  - Consulting during key transition phases

pragmainnovation

IP networking big data use cases

Traffic engineering

- Peering analysis,
- Backbone engineering,
- Capacity planning,
- Latency checking,
- Network Troubleshooting and monitoring,
- Network automation,
- …

Security

- Forensic and post mortem,
- Traffic anomaly,
  - DoS and DDoS,
- Remediation strategy
- Security Automation,
- …

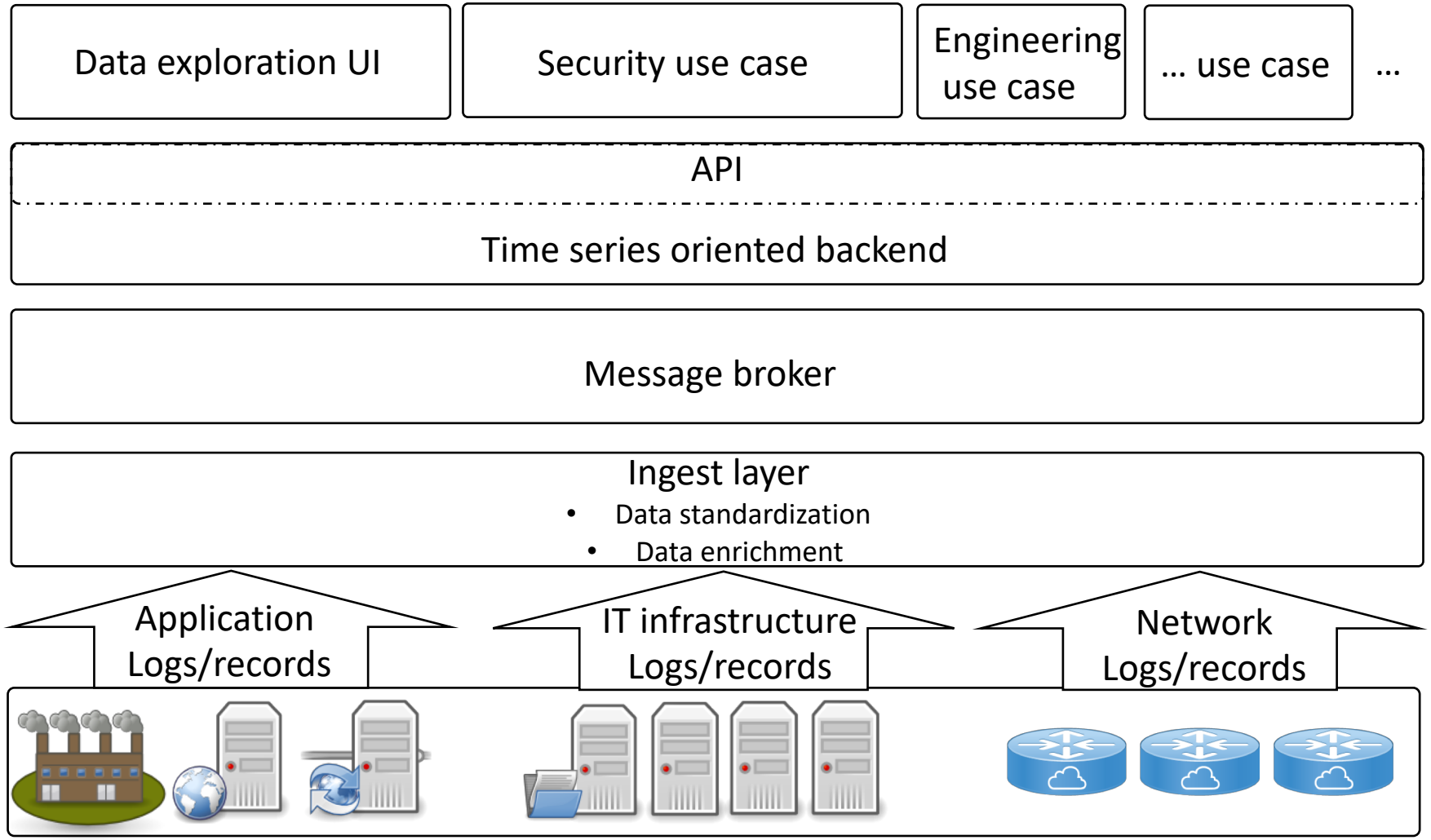
Business Intelligence

- Geo-marketing,
- Cost modeling,
- Profiling,
- Data mining,
- Service usage,
- OTT usage
- …

Compelling events for a game change

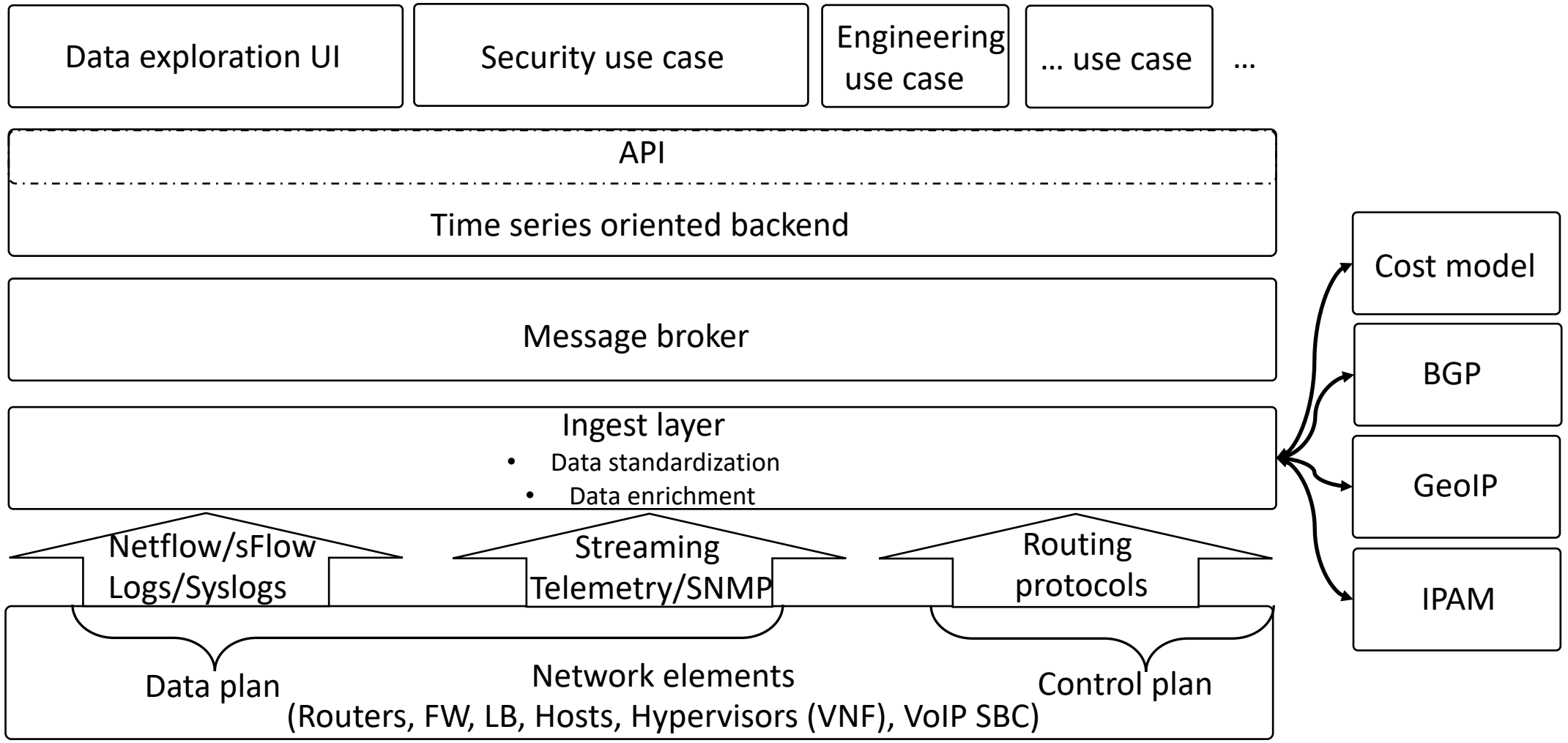
- Availability of open source time series backend solution
- Ingest tools of network data (open source)
- Community around open source UI analytics
- Appetite for valuable and customer specific big data solution
  - Companies data is part of their assets
- Need around customer specific use cases like
  - Cost optimization and network design as close as possible to enterprise specifics,
  - Application oriented SLA’s and metrics,
  - Network process automation,

Data pipe line: reference model



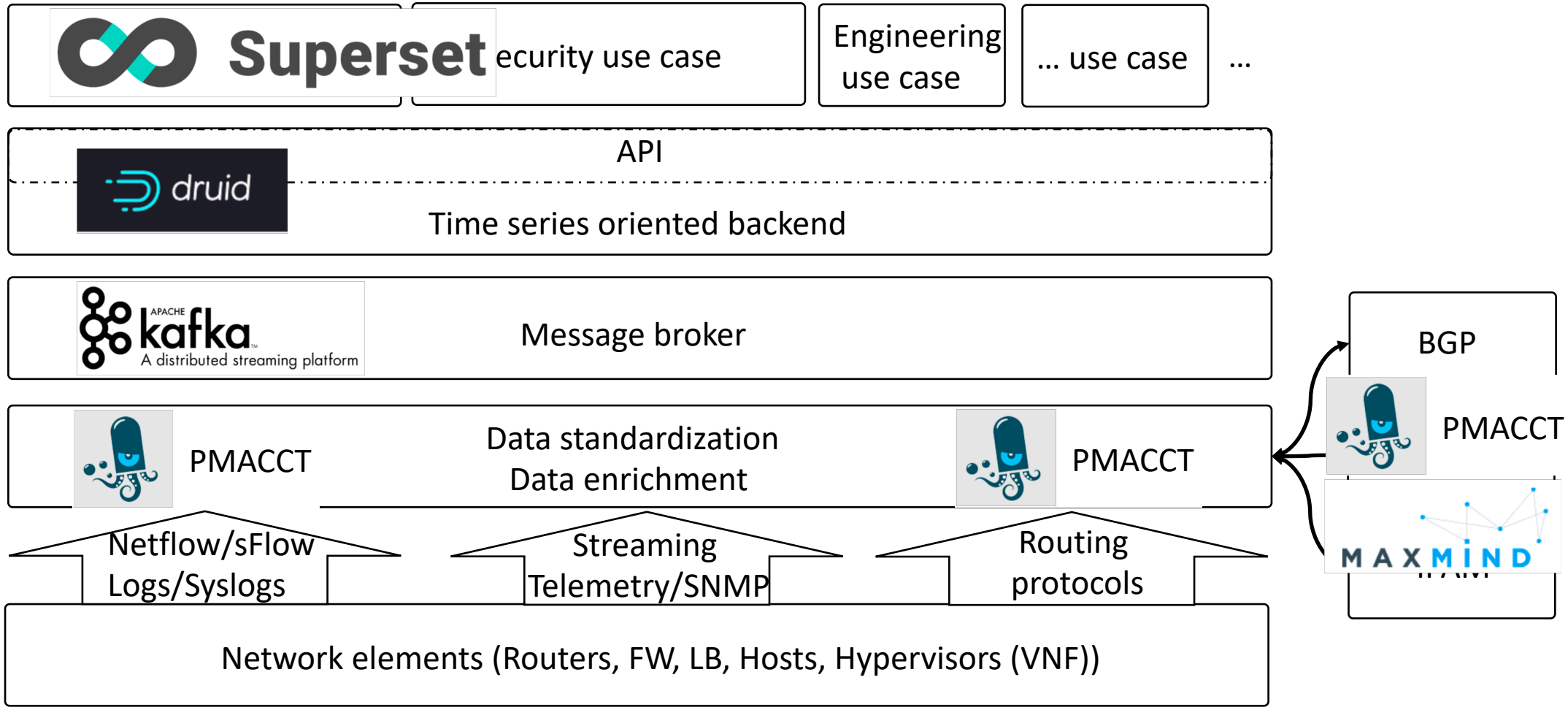
pragmainnovation

Pipe line overview: Network telemetry



pragmainnovation

Pipe line overview: IP telemetry R1.0



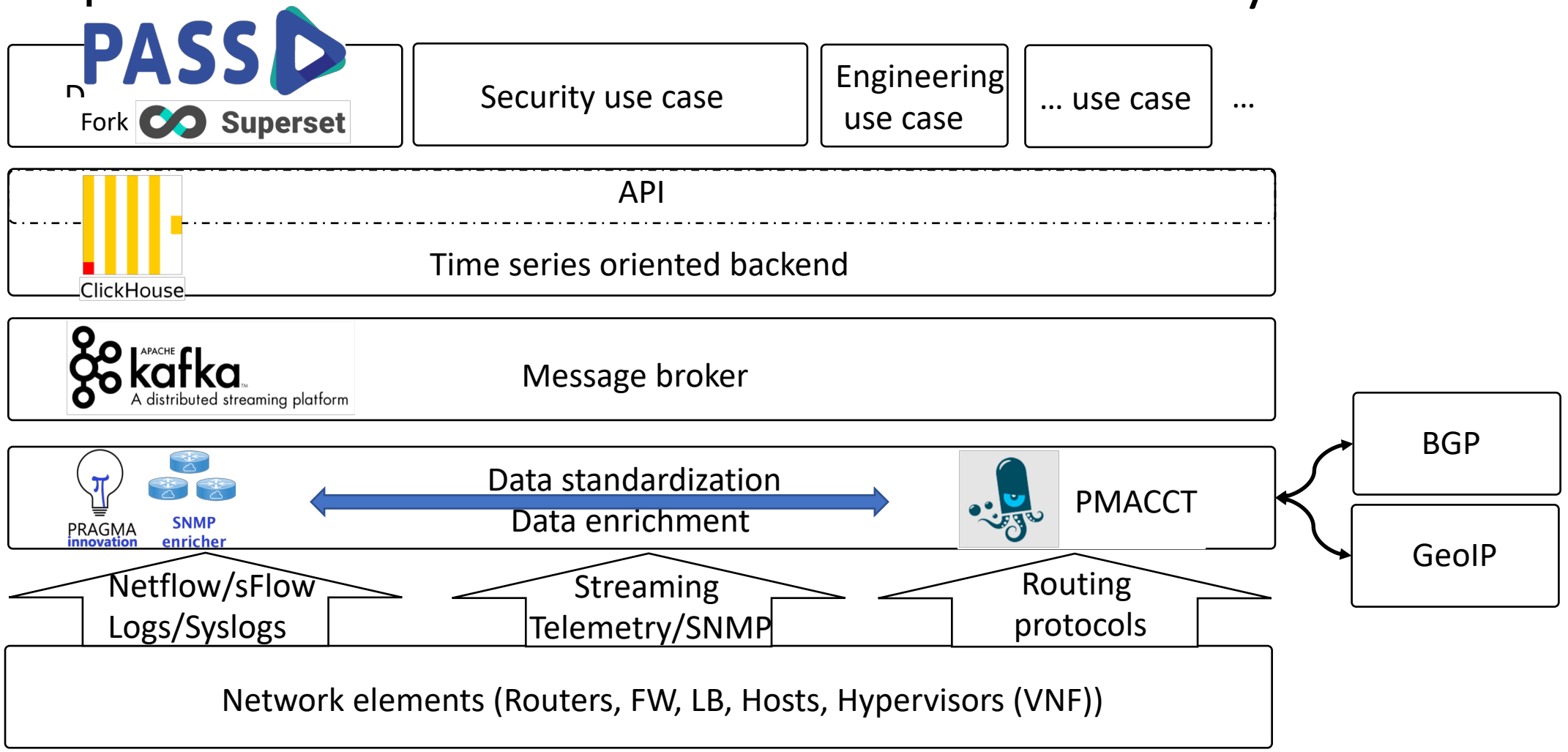
UI to be consolidated based on evaluation

pragmainnovation

R1.0 Main Issues highlighted with R 1.0

- Time granularity and data resampling:
  - We have to manage network bin cycle and user variable time granularity
  - Network bin cycle: SUM of flows that match the query
  - If User bin cycle > network bin cycle: User bin cycle: MAX/PERCENTILE of SUM(flows),
    - Not doable within Druid backend
- SNMP data enrichment : two options (not exclusive)
  - Create SNMP kafka streams,
  - Enrich Netflow/sFlow data,
- IPv4/IPv6 fields and CIDR block queries
  - Today IP address stored as string
  - Druid doesn’t support of binary storage of IPv4 and IPv6
  - Druid doesn’t support of CIDR block indexing
  - Blocking issue to enable CIDR filters or CIDR flow grouping
- Tech/Eco issue:
  - Druid cluster requires many nodes: zookeeper, meta data, druid data nodes, deep storage (HDFS), and an army of JVM’s.
- Build a release 2.0 that address those issues

Pipe line overview: IP flow telemetry R2.0

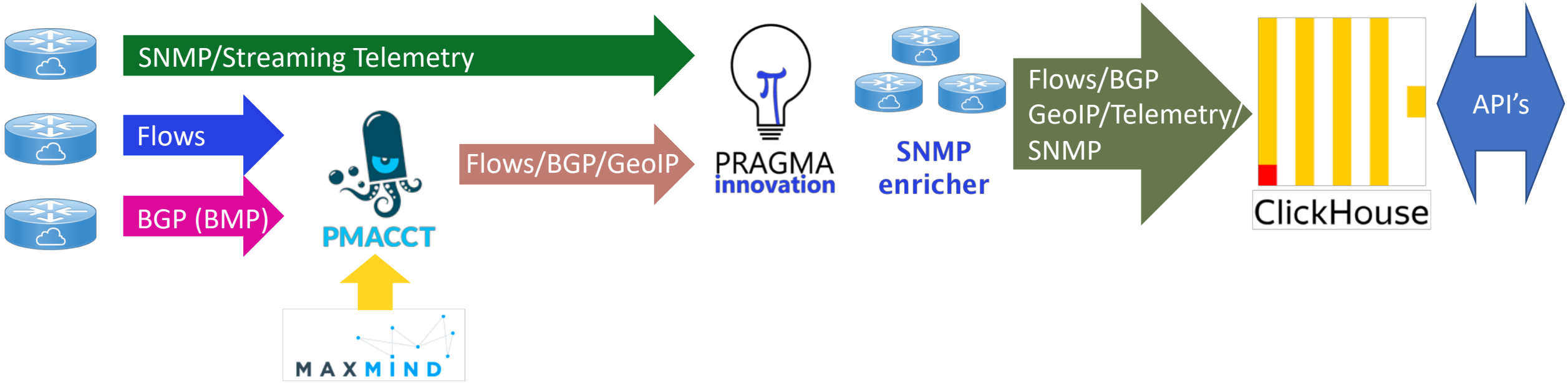


pragmainnovation

Issues of 1.0 addressed by Clickhouse

- No need of complex node hierarchy and deep storage like Hadoop HDFS
  - Clickhouse manage data and data resiliency by itself no need to add other level of complexity
- Clickhouse support of IPv4 and IPv6 fields as well as operators of those fields,
- Data sampling and re-sampling : clickhouse is not ”touching” or aggregating native metrics coming from network equipment and that’s much much better that way !
  - Even better, it can do it if you want using Aggregating Merge tree engine

IP telemetry: Network flows pipeline



Message broker

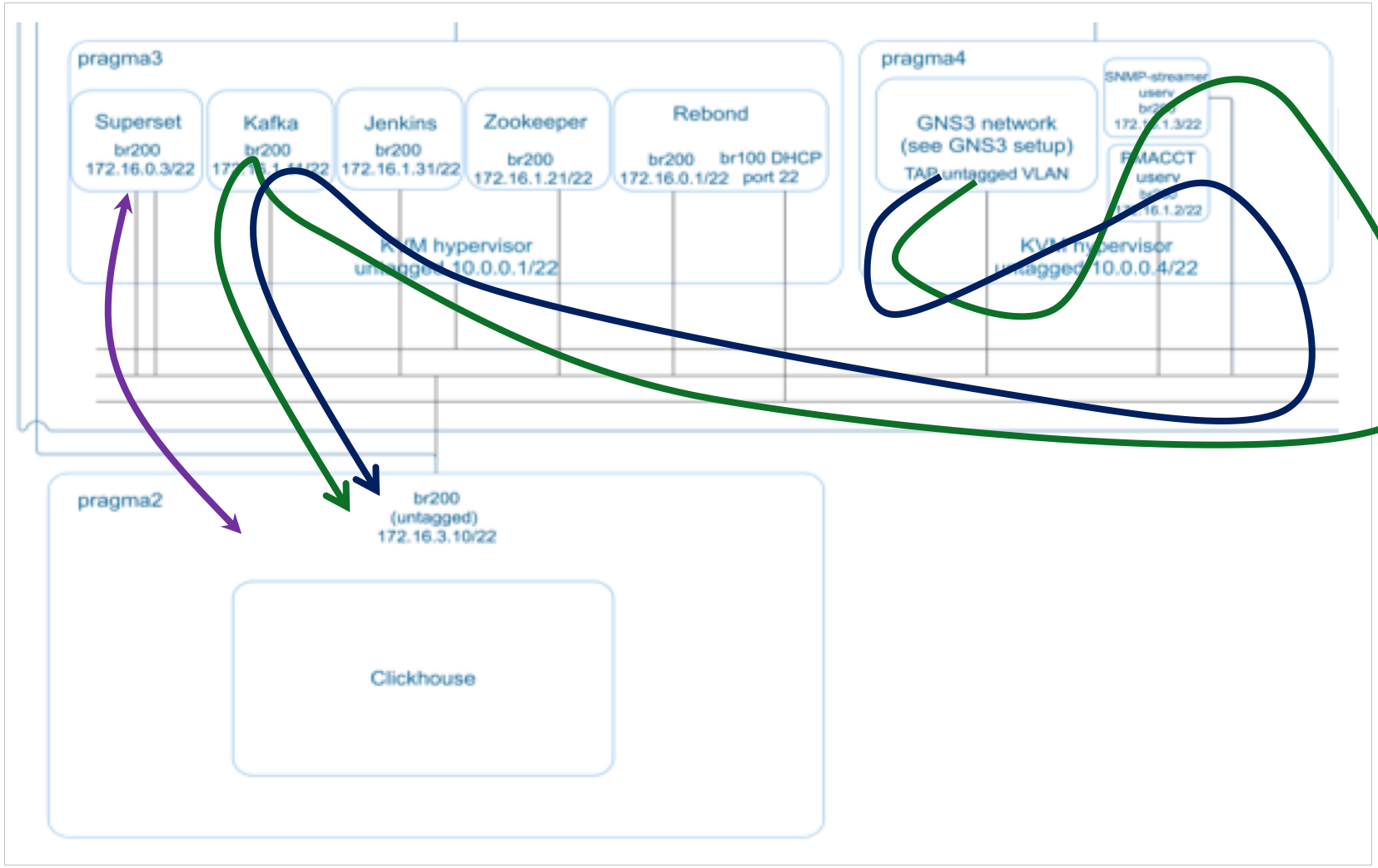
- Kafka
  - Kafka support on PMACCT, SNMP streamer, SNMP enricher
  - Performances:
    - Kafka: 100K's events per second,
    - RabbitMQ: 20K+ events per second,
  - RabbitMQ more routing capabilities between large amount of process
  - Kafka more oriented for large amount of messages between simple entities,

- Value for Netflow/SNMP telemetry use case:
  - Netflow could be very bursty depending on implementation (due to active timeout, hardware implementation), Kafka will be very useful to absorb traffic spikes,
  - Kafka could help during backend upgrades (storing data during the upgrade operations)

Backend

- Clickhouse
  - Performance at ingest, performance to deal with queries,
  - Manage storage and resiliency without additional software (no HDFS),
- Value for the IP telemetry use case:
  - Clickhouse propose valuable features for Kafka ingest
    - Queuing, Views and calculation between Kafka JSON and table model,
  - IP telemetry metrics are about speed (PPS and BPS)
    - Clickhouse do support all mathematical calculation (MAX or SUM, Quantile, …)
    - Clickhouse doesn't apply arbitrary metrics aggregation and keep it raw
- Open source (Yandex)
  - https://github.com/yandex/ClickHouse
    - CPP code with a decent amount of comments and specs
  - Good documentation quality: https://clickhouse.yandex/docs/en/
  - Company to provide advanced services Altinity

Example of design (Lab)



pragmainnovation

ClickHouse

Access data and display dashboards

- Choice we made about frontend and UI is to use Superset with addition of custom modules
  - https://github.com/apache/incubator-superset
- Clickhouse data are reachable via SQLAlchemy and makes clickhouse working with all frameworks using SQLAlchemy
  - https://github.com/xzkostyan/clickhouse-sqlalchemy
  - Allows both access via HTTP and native interface
- Clickhouse and SQLalchemy allow the use of PANDAS (lib python for “big data”)

pragmainnovation

Ingest data

- Native support of Kafka consumer
- Flexible implementation with some form of “SPARK like” features
  - Create a queue that map your data model (like JSON key/value)
  - Create your destination table which could differ from the queue table
  - Create a MATERIALIZED view between the two: you have all SQL tools to play with fields between the input queue table and the destination table
- Example with SNMP data:
  - Queue:

```
CREATE TABLE default.snmp_queue ( rtr_name String,  if_name String,  if_description String,  if_speed UInt32,  if_index UInt16,  observation_time UInt32,  date_calculation UInt64,  in_octets_qty UInt64,  in_unicast_pkts_qty UInt64,  in_multicast_pkts_qty UInt64,  in_broadcast_pkts_qty UInt64,  out_octets_qty UInt64,  out_unicast_pkts_qty UInt64,  out_multicast_pkts_qty UInt64,  out_broadcast_pkts_qty UInt64) ENGINE = Kafka('172.16.1.11:9092', 'snmp-streamer', 'clickhouse', 'JSONEachRow')
```

pragmainnovation

ClickHouse

Ingest data

- Example with SNMP data:
  - Destination table:

```
CREATE TABLE default.snmp ( rtr_name String,  if_name String,  if_description String,
if_speed UInt32,  if_index UInt16,  observation_time UInt32,  date_calculation DateTime,
day Date,  in_octets_qty UInt64,  in_unicast_pkts_qty UInt64,  in_multicast_pkts_qty
UInt64,  in_broadcast_pkts_qty UInt64,  out_octets_qty UInt64,  out_unicast_pkts_qty
UInt64,  out_multicast_pkts_qty UInt64,  out_broadcast_pkts_qty UInt64) ENGINE =
MergeTree(day, (rtr_name, if_index, date_calculation), 8192)
```

ClickHouse

Ingest data

- Example with SNMP data:
  - Materialized view playing with UNIX timestamp:

```
CREATE MATERIALIZED VIEW default.snmp_consumer TO default.snmp ( day Date,
date_calculation DateTime,  rtr_name String,  if_name String,  if_description String,
if_speed UInt32,  if_index UInt16,  observation_time UInt32,  in_octets_qty UInt64,
in_unicast_pkts_qty UInt64,  in_multicast_pkts_qty UInt64,  in_broadcast_pkts_qty
UInt64,  out_octets_qty UInt64,  out_unicast_pkts_qty UInt64,  out_multicast_pkts_qty
UInt64,  out_broadcast_pkts_qty UInt64) AS SELECT toDate(date_calculation) AS day,
toDateTime(date_calculation) AS date_calculation, rtr_name, if_name, if_description,
if_speed, if_index, observation_time, in_octets_qty, in_unicast_pkts_qty,
in_multicast_pkts_qty, in_broadcast_pkts_qty, out_octets_qty, out_unicast_pkts_qty,
out_multicast_pkts_qty, out_broadcast_pkts_qty FROM default.snmp_queue
```

Thanks