

High Altitude, Low Risk

Measuring Reliability in the Cloud using OSS technology

akass@digitalocean.com



table.of.contents

1. alex@digitalocean:~\$ whoami
2. **OSS** building blocks at DigitalOcean
3. **Service Level Management**: productizing reliability data
4. What's Next/Wrap-Up
5. Questions (and Answers?)

alex@digitalocean:~\$ whoami



Global Cloud Hosting Provider

12 Data Centers, worldwide

DO builds **products** that help engineering teams build,
deploy and scale cloud applications

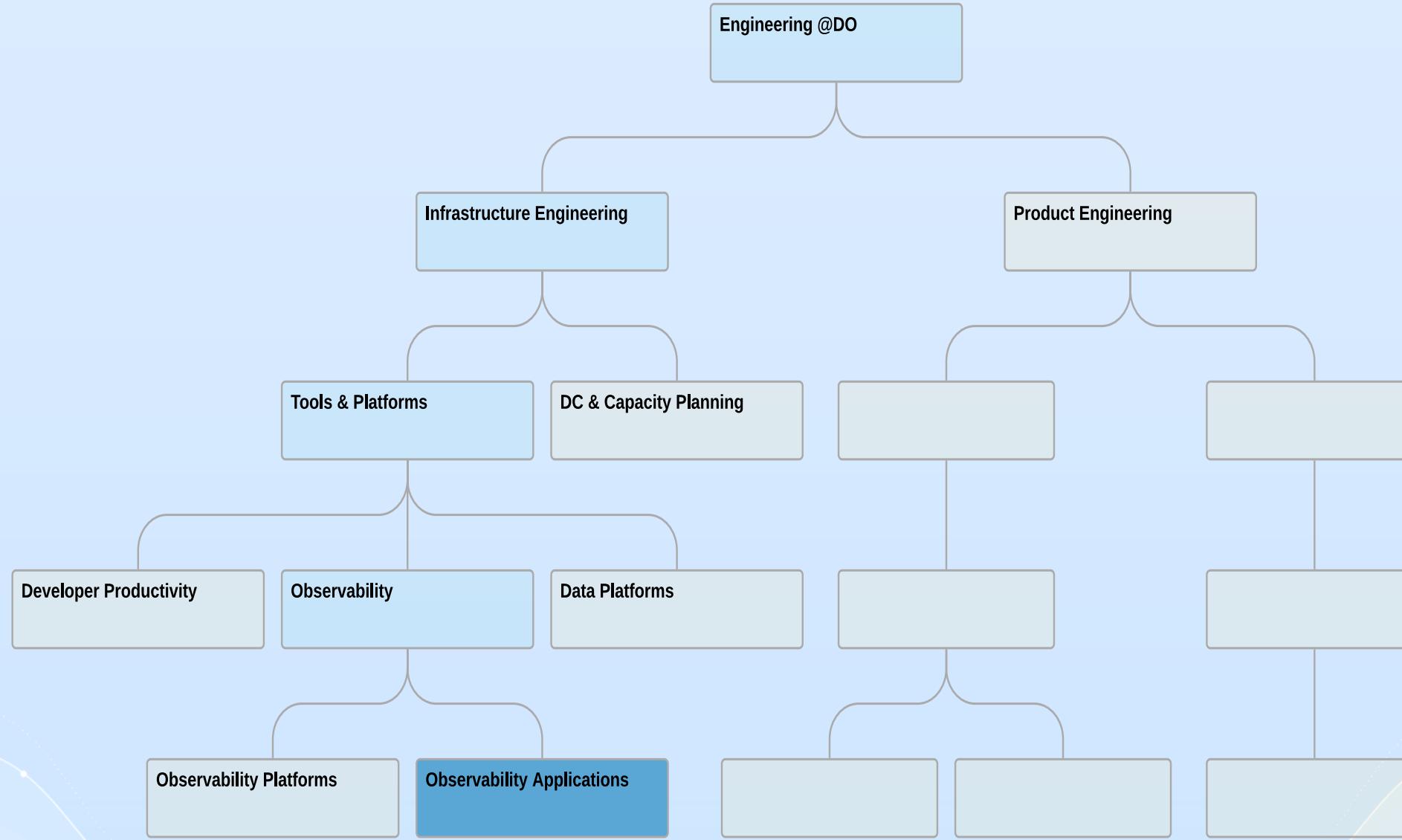


alex@digitalocean:~\$ whoami



my team

alex@digitalocean:~\$ whoami



```
alex@digitalocean:~$ whoami
```

Observability Applications - *what is our mission?*

1. to **simplify** and **optimize** consumption of data internally
2. to **reduce** incident **MTTD/MTTR** through custom applications
3. to help **define**, **maintain**, and **broadcast** source-of-truth performance and reliability data to the rest of the organization

```
alex@digitalocean:~$ whoami
```

How can we achieve these things?

1. to **simplify** and **optimize** consumption of data
2. to **reduce** incident **MTTD/MTTR** through custom applications
3. to help **define**, **maintain**, and **broadcast** source-of-truth performance and reliability data to the rest of the organization

```
alex@digitalocean:~$ whoami
```

How can we achieve these things?

- By building data products...
- ...for **specific stakeholders**...
- ...with widely available **open source tools**...
- ...to integrate with systems already in place!

(hint: **Service Level Management**)

Open Source Software Building Blocks



OSS tools in use at DigitalOcean

(a sampling)



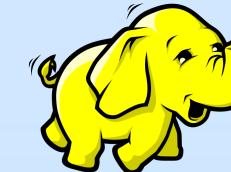
libvirt



ceph



Prometheus



OSS tools in use at DigitalOcean

<https://github.com/digitalocean>

Service Level Management (SLM)

Service Level Management (SLM)

SLM

SLAs

SLOs

Service Level Management (SLM)

SLA

an agreement with consequences.

Service Level Management (SLM)

SLO

an Objective, or goal, but not a commitment.

Service Level Management (SLM)

SLA = service consumption

SLO = service production

Service Level Management (SLM)

prioritization

1. SLOs
2. SLAs

Service Level Management (SLM)

ahem:

...to **standardize**, **centralize**, and **publish** objective data about the reliability of both customer- and internal-facing products and services.

aka: **a data product** to serve all of DO

ETL...E?

Service Level Management (SLM)

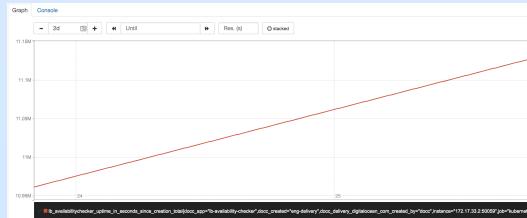
E

Extraction points:

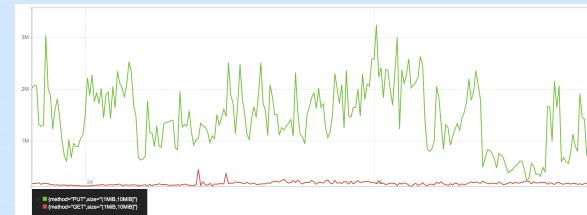
-  Prometheus
-  kafka
-  MySQL™
-  PostgreSQL



- Easy to implement and deploy at scale
- Flexible time-series metrics
 - *Counters: monotonically increasing*



- *Gauges: metrics for a current state of a service*





- Export to HTTP endpoint for consumption

```
func ServeSLAMetrics(e *Environment) error {
    reg := prometheus.NewRegistry()
    err := reg.Register(NewSLACollector(e.SLALastRunFile))
    if err != nil { return err }

    handler := promhttp.HandlerFor(reg, promhttp.HandlerOpts{})

    listener, err := net.Listen("tcp", e.SLAAddr)
    if err != nil { return err }

    go http.Serve(listener, handler)
    return nil
}
```

SLM: E



Globally-distributed servers



...

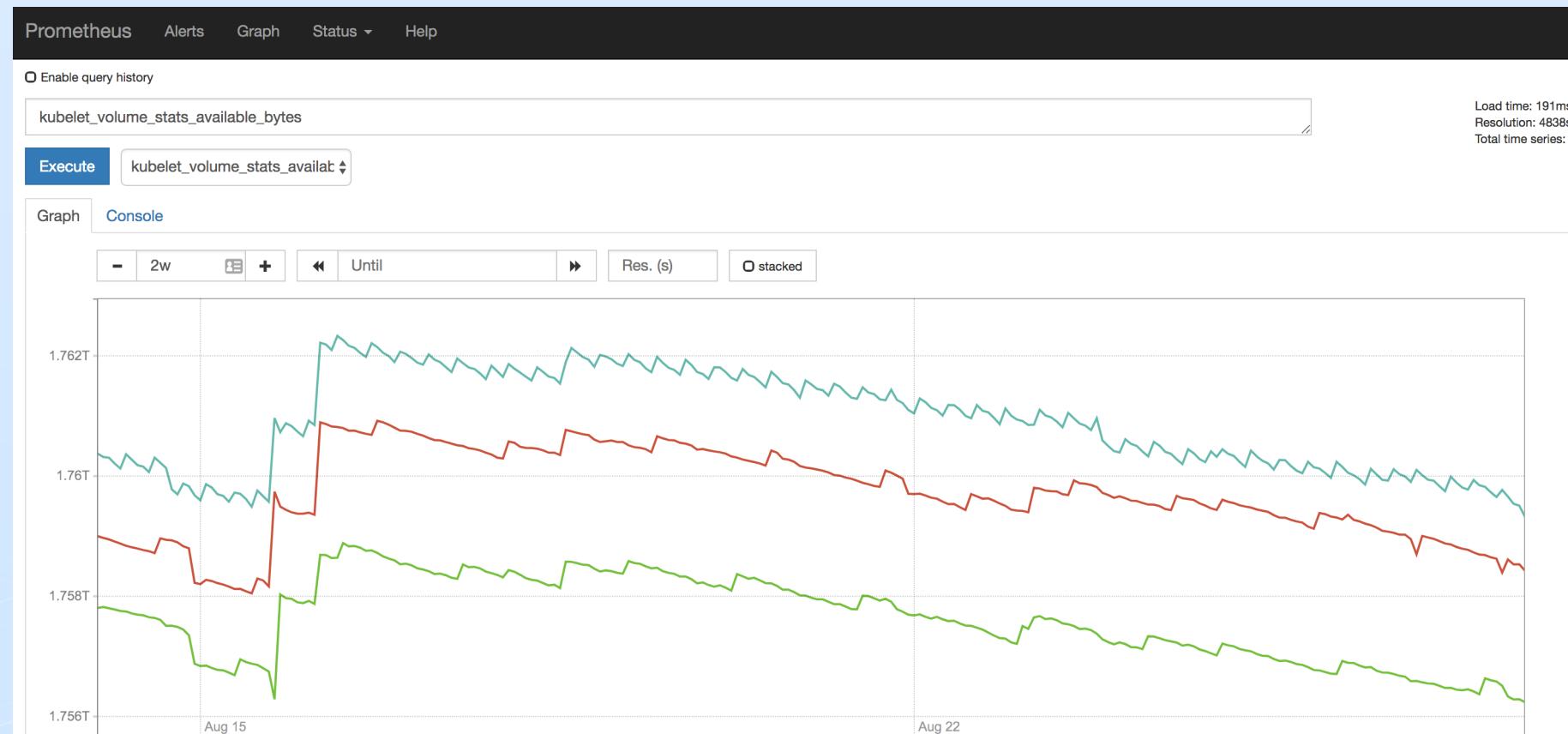


PANDORA

a bunch of Prometheus servers on management droplets
configured using a Git repository and some Go code

```
services:  
  # eng-platcore  
  - luca-kubernetes-alerts  
  - luca-kubernetes-apiservers  
  - luca-kubernetes-kubelet  
  - luca-kubernetes-nodes  
  - luca-kubernetes-pods  
  - luca-kubernetes-service-endpoints
```

SLM: E



SLM: E



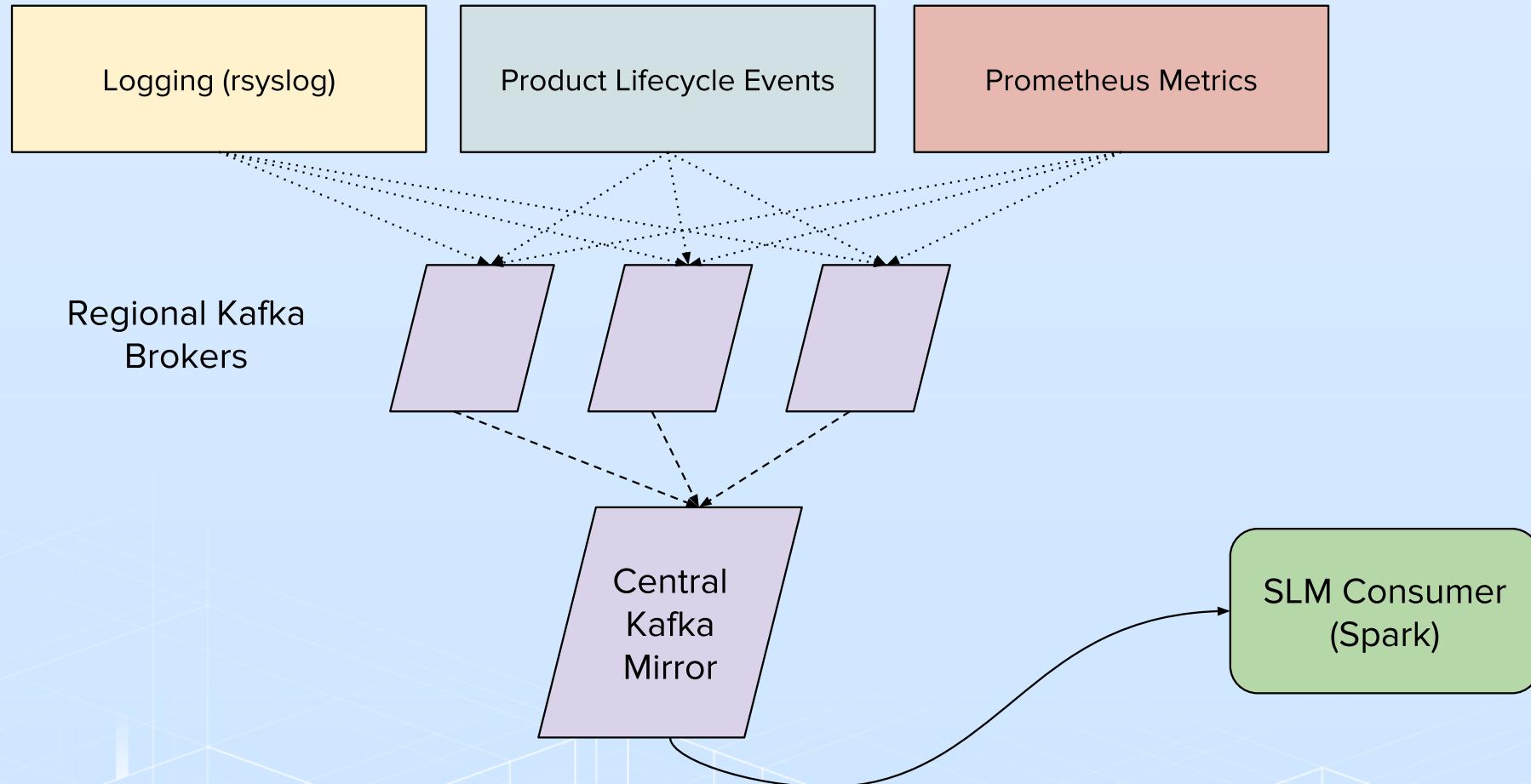
- Caveat: data retention buffers



Efficient distribution/consumption of global data, including:

- Centralized Logging ()
- Regional Product Lifecycle Events (protocol buffers)
- Global Metrics Data ()

SLM: E



SLM: E



- Caveat: data retention buffers

Service Level Management (SLM)

ET

SLM: ET



```
prom_data_payload = json.loads(  
    requests.get(full_prometheus_query_string, timeout=50)) \  
    .content.decode() ['data'] ['result']
```

SLM: ET



```
for timeseries in prom_data_payload:  
  
    metric = timeseries["metric"]  
    values = timeseries["values"]  
  
    rows.extend([  
        {  
            **metric,  
            "timestamp": datetime.fromtimestamp(v[0]),  
            "value": float(v[1])  
        }  
        for v in values])  
  
return pd.DataFrame(rows)
```

SLM: ET



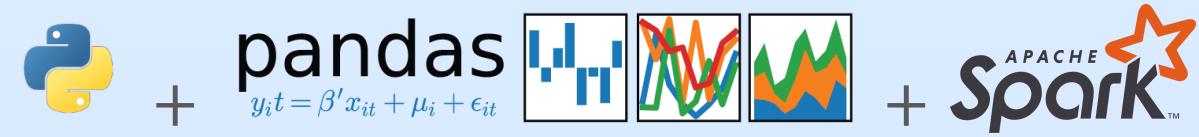
```
@F.pandas_udf(df_fixed.schema, F.PandasUDFType.GROUPED_MAP)
def resample_by_ts(x):
    return x.drop_duplicates('ts') \
        .sort_values('ts') \
        .set_index('ts') \
        .asfreq('300s', method='ffill') \
        .resample('300s') \
        .first().reset_index()

df_resampled = df_fixed.groupBy(
    "droplet_id", "node", "metric_name").apply(resample_by_ts)
```

Service Level Management (SLM)

ETL

SLM: ETL



```
spark_df \  
    .write \  
    .mode(saveMode="append") \  
    .partitionBy(  
        "metric_name",  
        "region",  
        "year",  
        "month",  
        "day",  
        "hour") \  
    .parquet(full_hdfs_target)
```

SLM: ETL



+

pandas
 $y_t = \beta' x_{it} + \mu_i + \epsilon_{it}$



```
import pyarrow as pa
import pyarrow.parquet as pq

def write(pandas_df, hdfs_url, *partition_cols):
    _check_url(hdfs_url)

    table = pa.Table.from_pandas(pandas_df, preserve_index=False)

    pq.write_to_dataset(
        table,
        root_path=hdfs_url,
        partition_cols=list(partition_cols),
        preserve_index=False,
        compression="snappy",
        flavor="spark")
```

Service Level Management (SLM)

ETLE

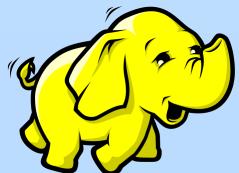
SLM: ETL



massively parallel, cross-catalog query engine

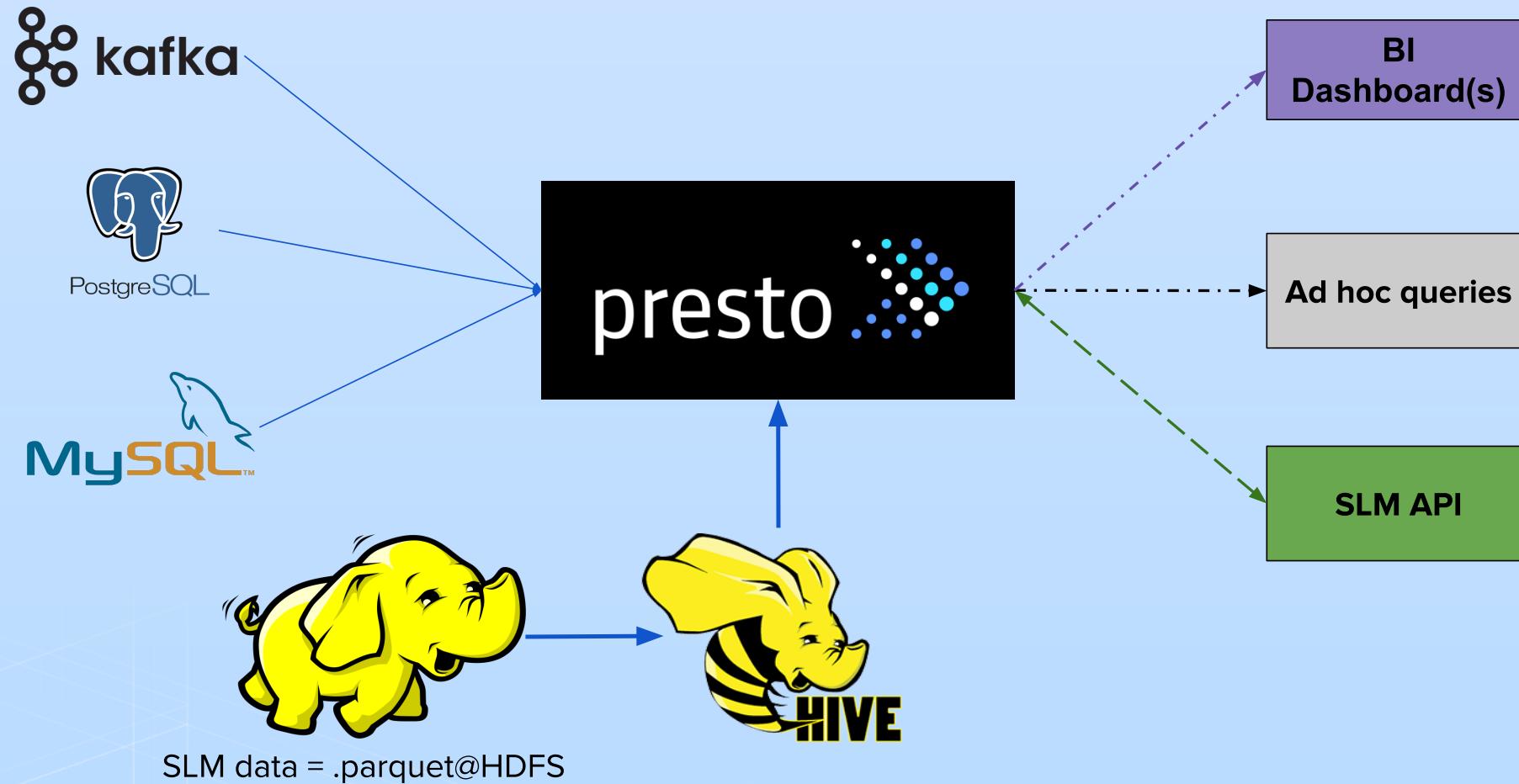


schema metadata



data warehouse

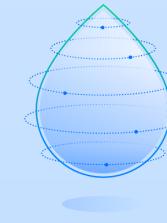
SLM: ETL



Service Level Management (SLM)

HV Droplet Availability:
an SLO Pipeline example

SLM: hvdroplet availability



A quick product history:

- Droplet product launched in 2011
- Pioneered selling VMs on SSDs
- over **100MM** droplets deployed

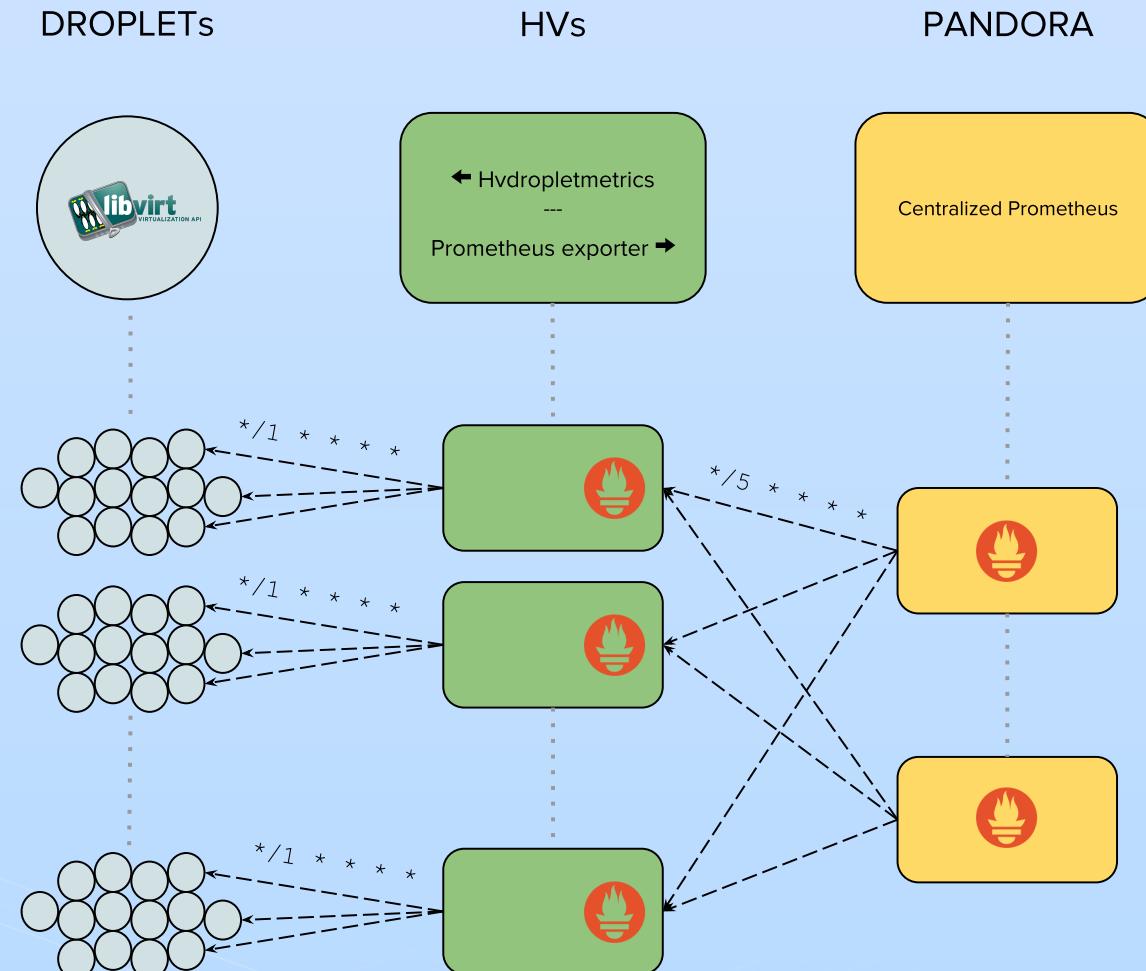
SLM: hypdroplet availability



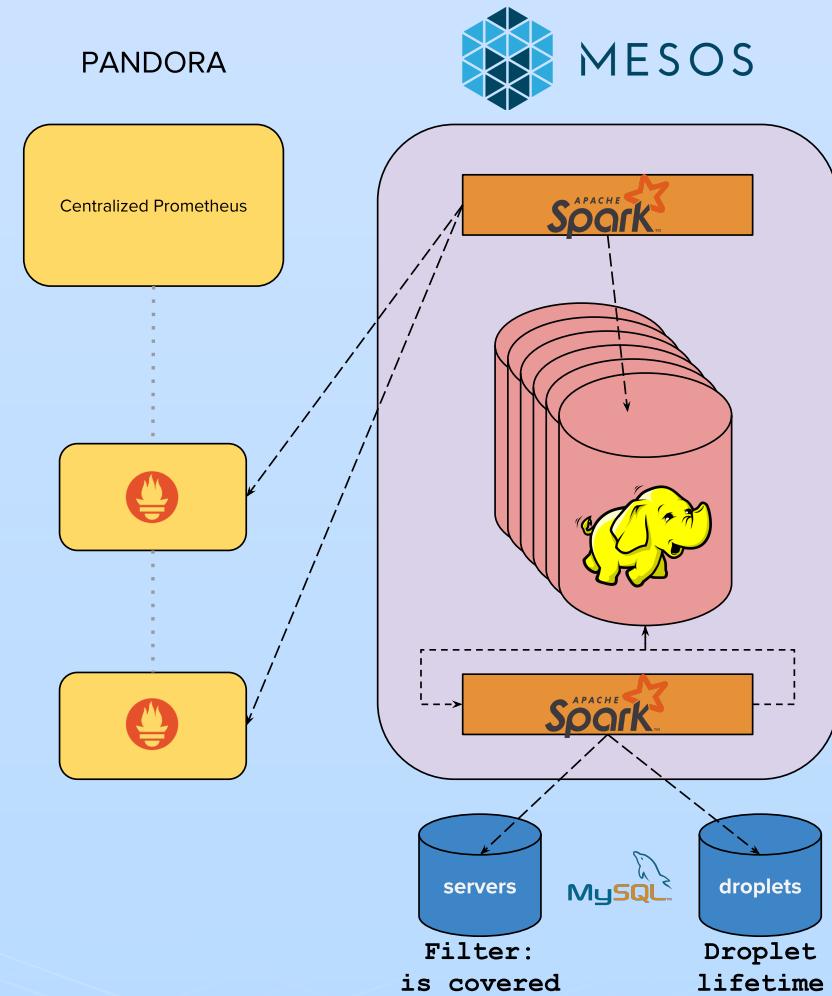
virtualization API

```
enum virDomainState {  
    VIR_DOMAIN_NOSTATE = 0  
    VIR_DOMAIN_RUNNING = 1  
    VIR_DOMAIN_BLOCKED = 2  
    VIR_DOMAIN_PAUSED = 3  
    VIR_DOMAIN_SHUTDOWN = 4  
    VIR_DOMAIN_SHUTOFF = 5  
    VIR_DOMAIN_CRASHED = 6  
    VIR_DOMAIN_PMSUSPENDED = 7  
}
```

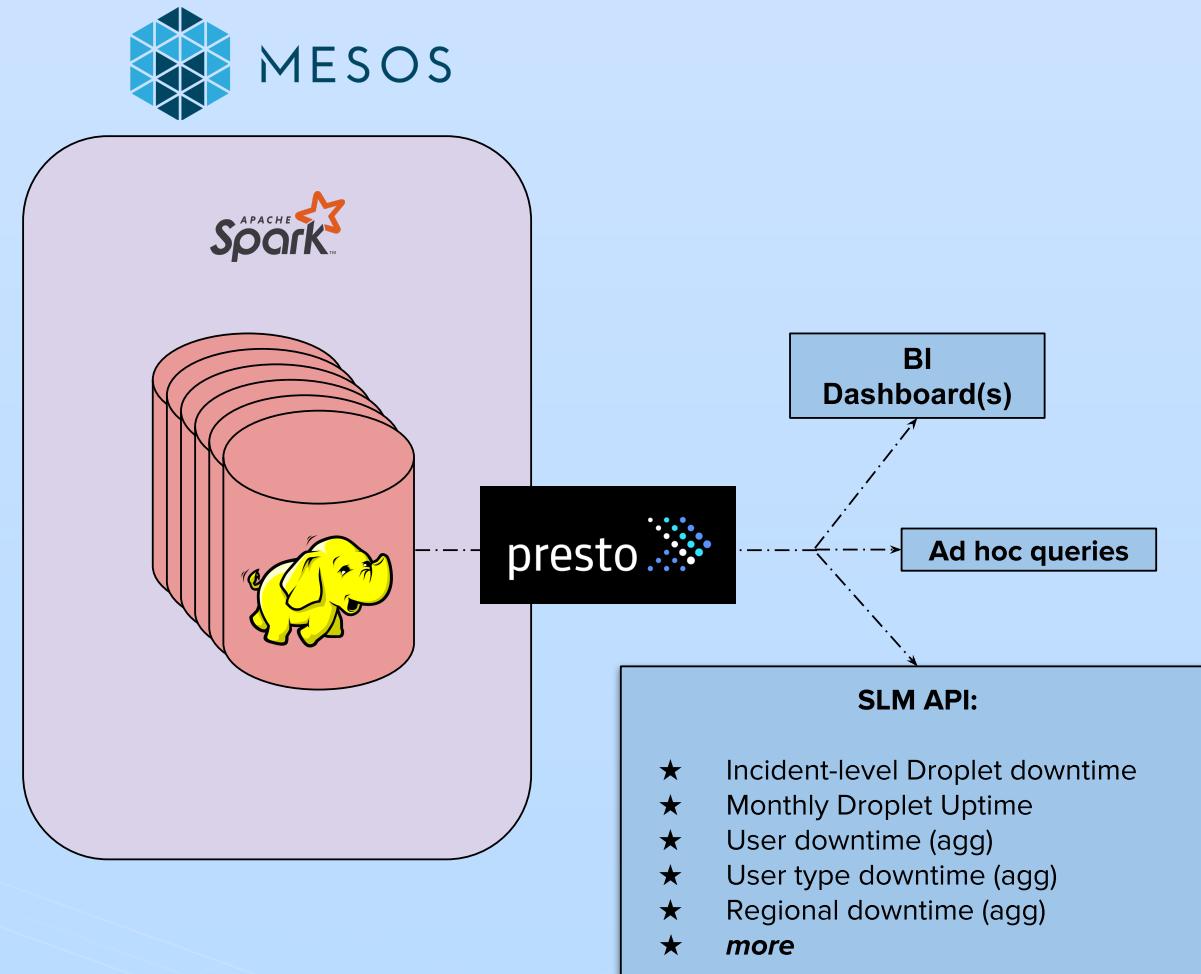
SLM: hvdroplet availability



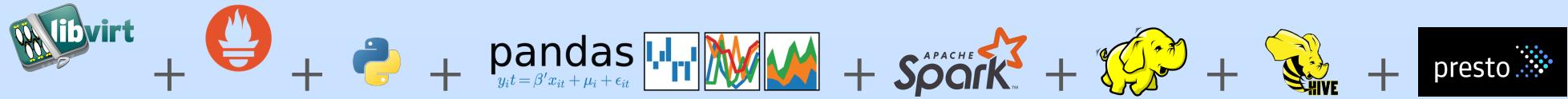
SLM: hvdroplet availability

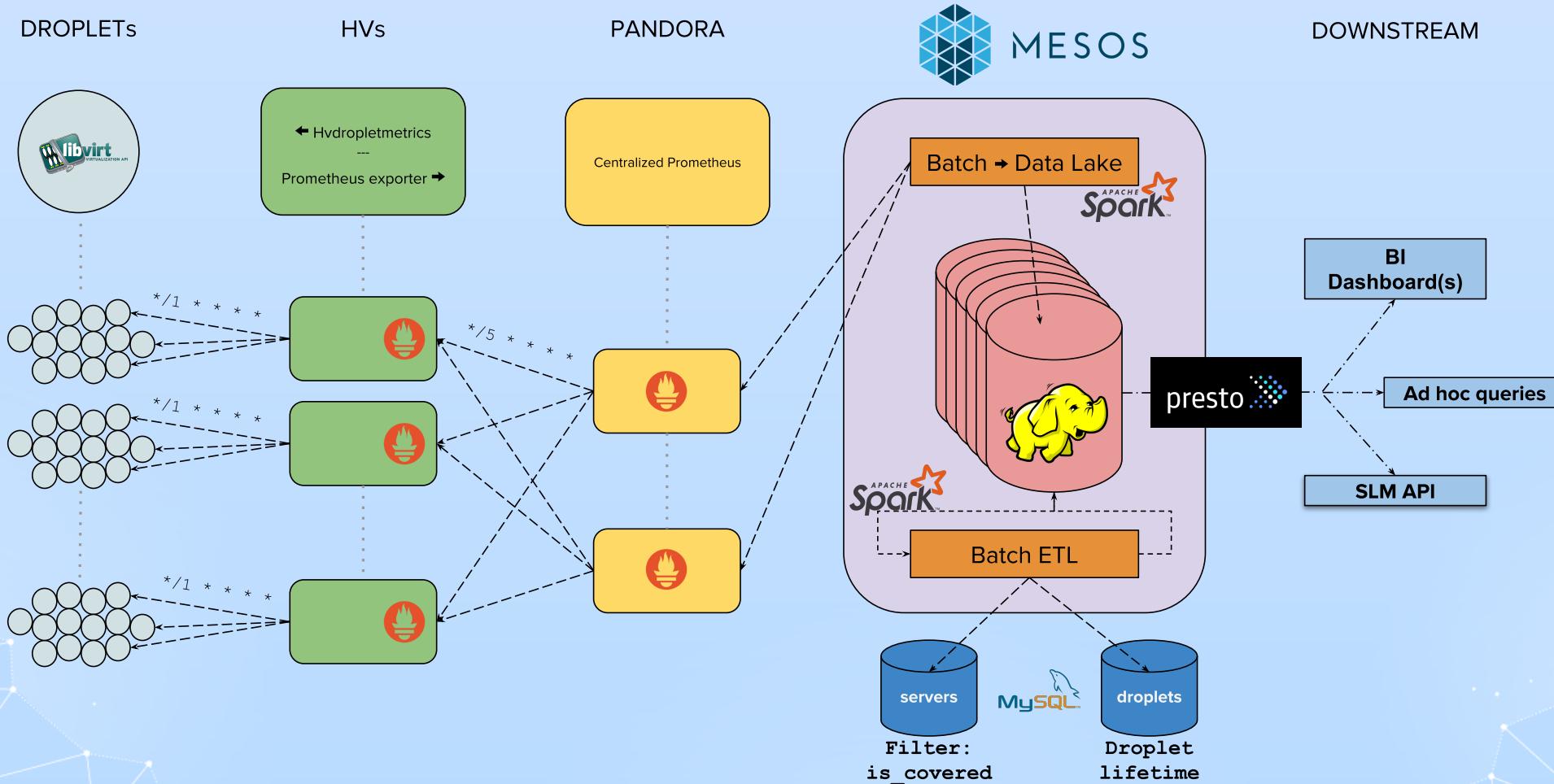


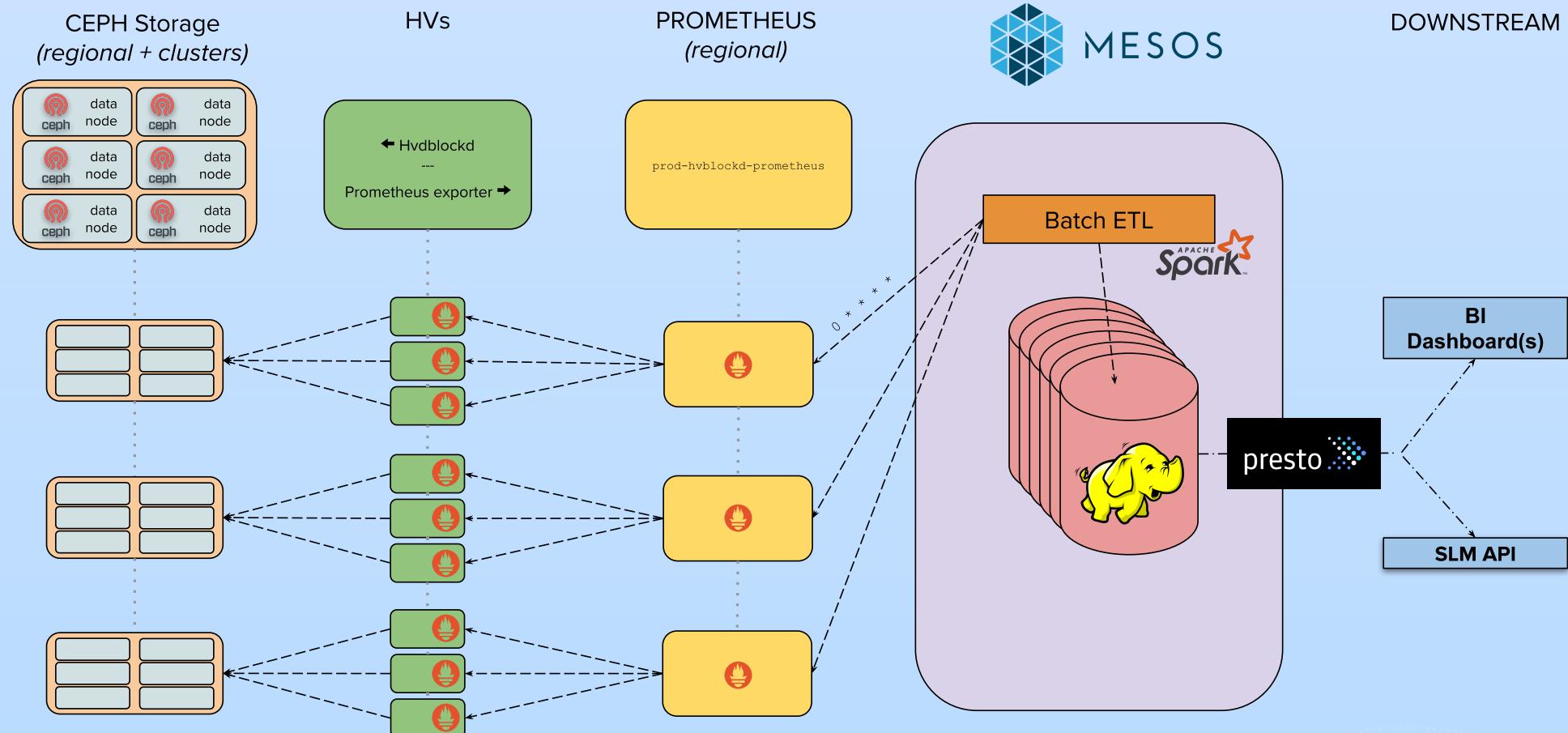
SLM: hvdroplet availability



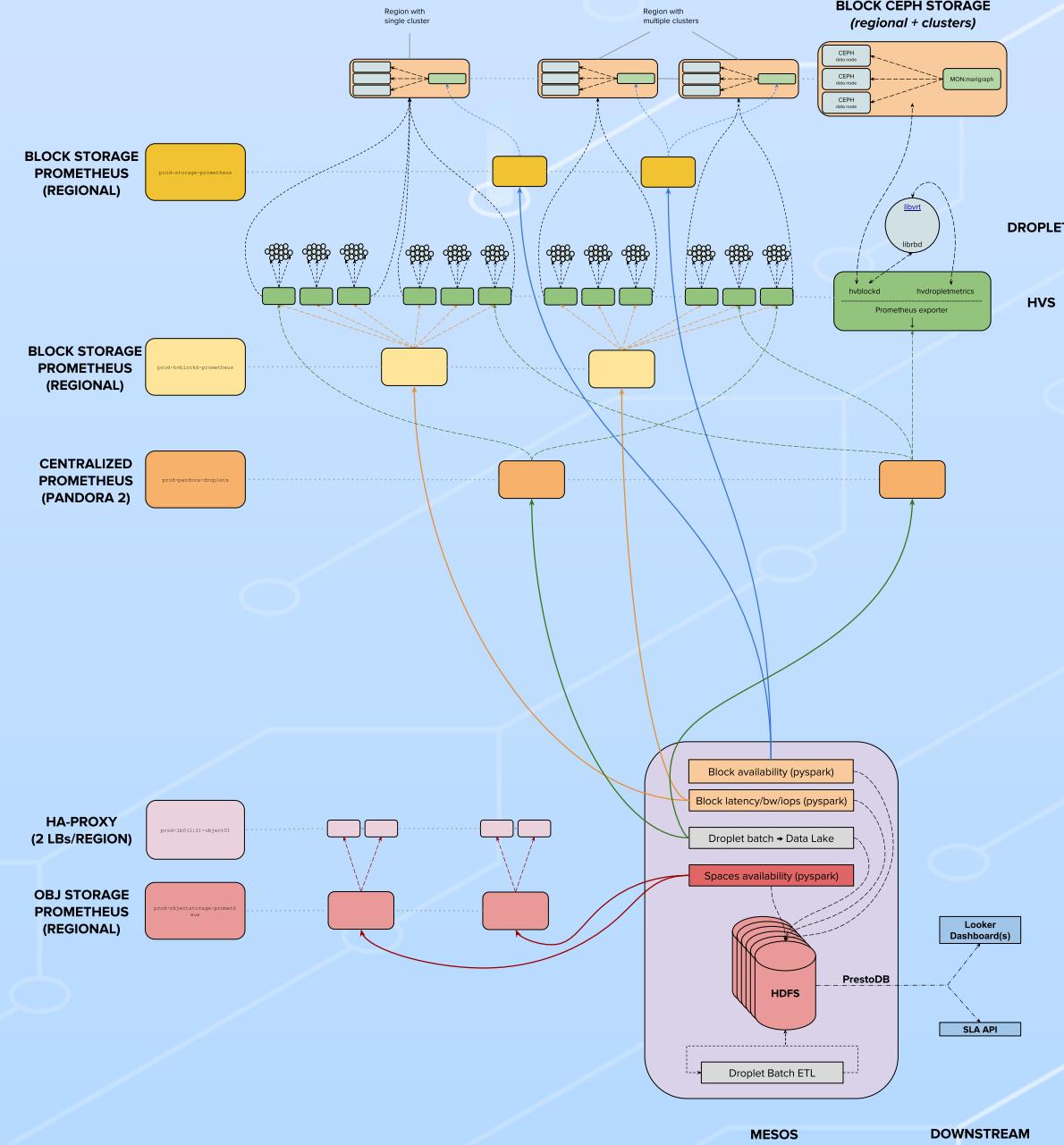
SLM: hvdroplet availability







SERVICE LEVEL MANAGEMENT (PRODUCT)



SLM: A Recap

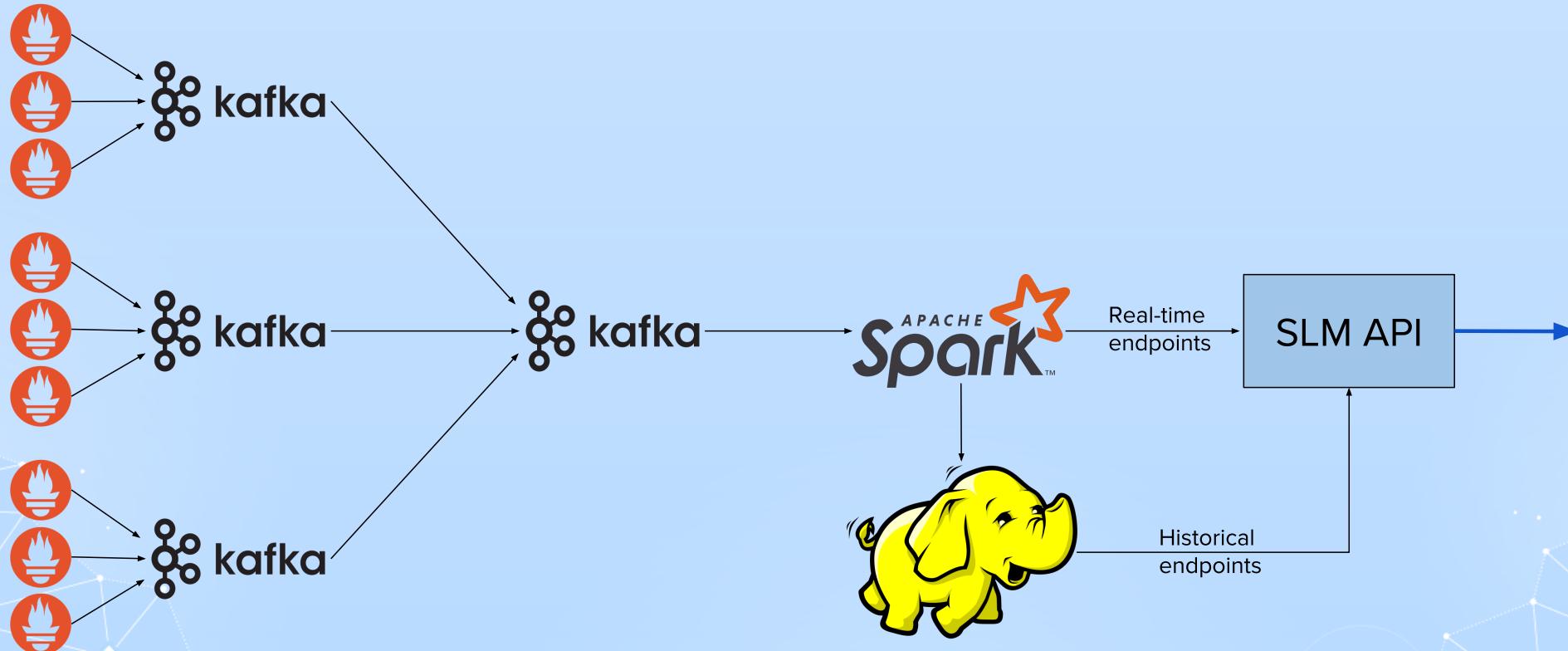
1. Portfolio
2. (nearly) unlimited data history
3. API

Next Steps in Observability Applications

Next Steps

SLM v2

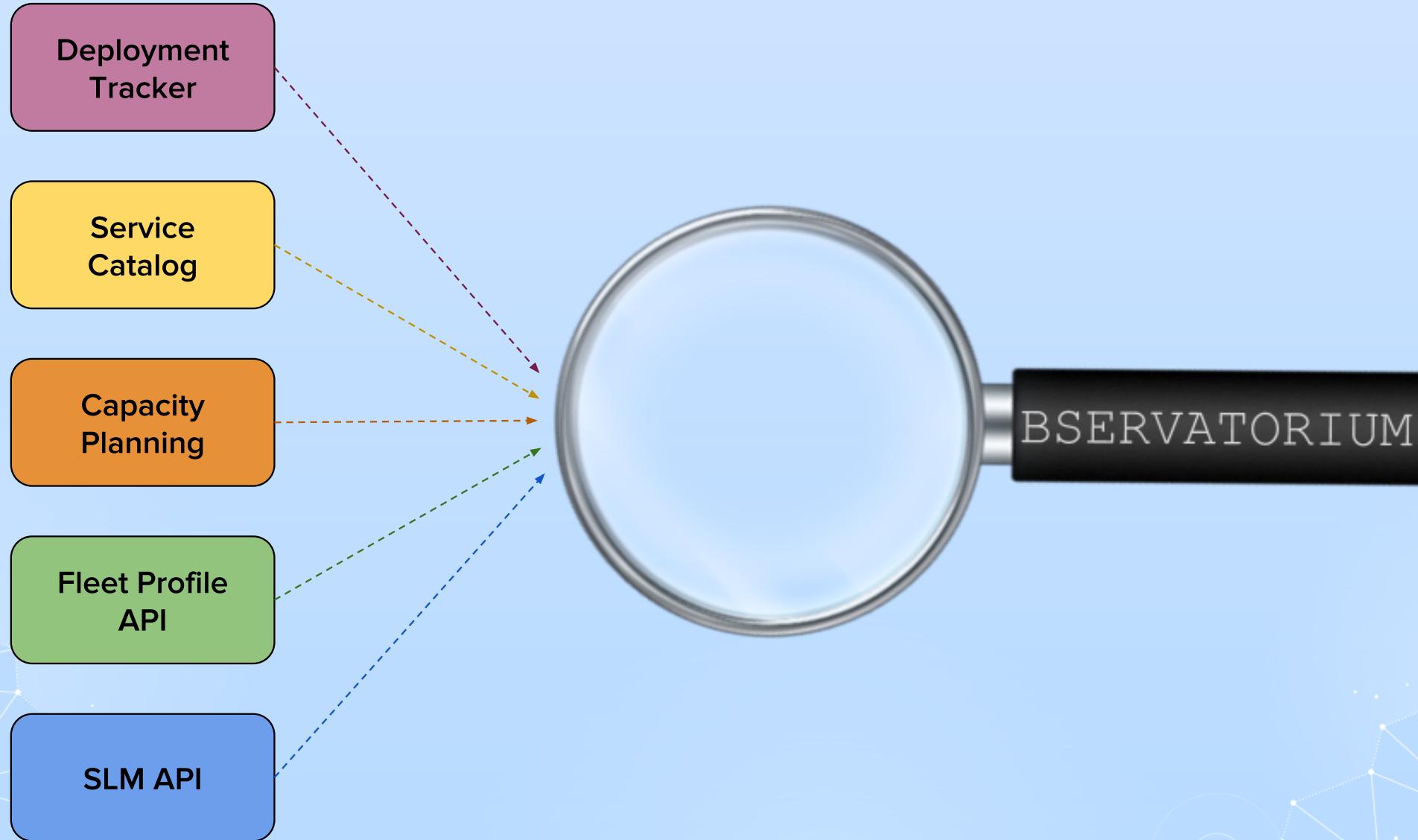
~~batch~~ streaming



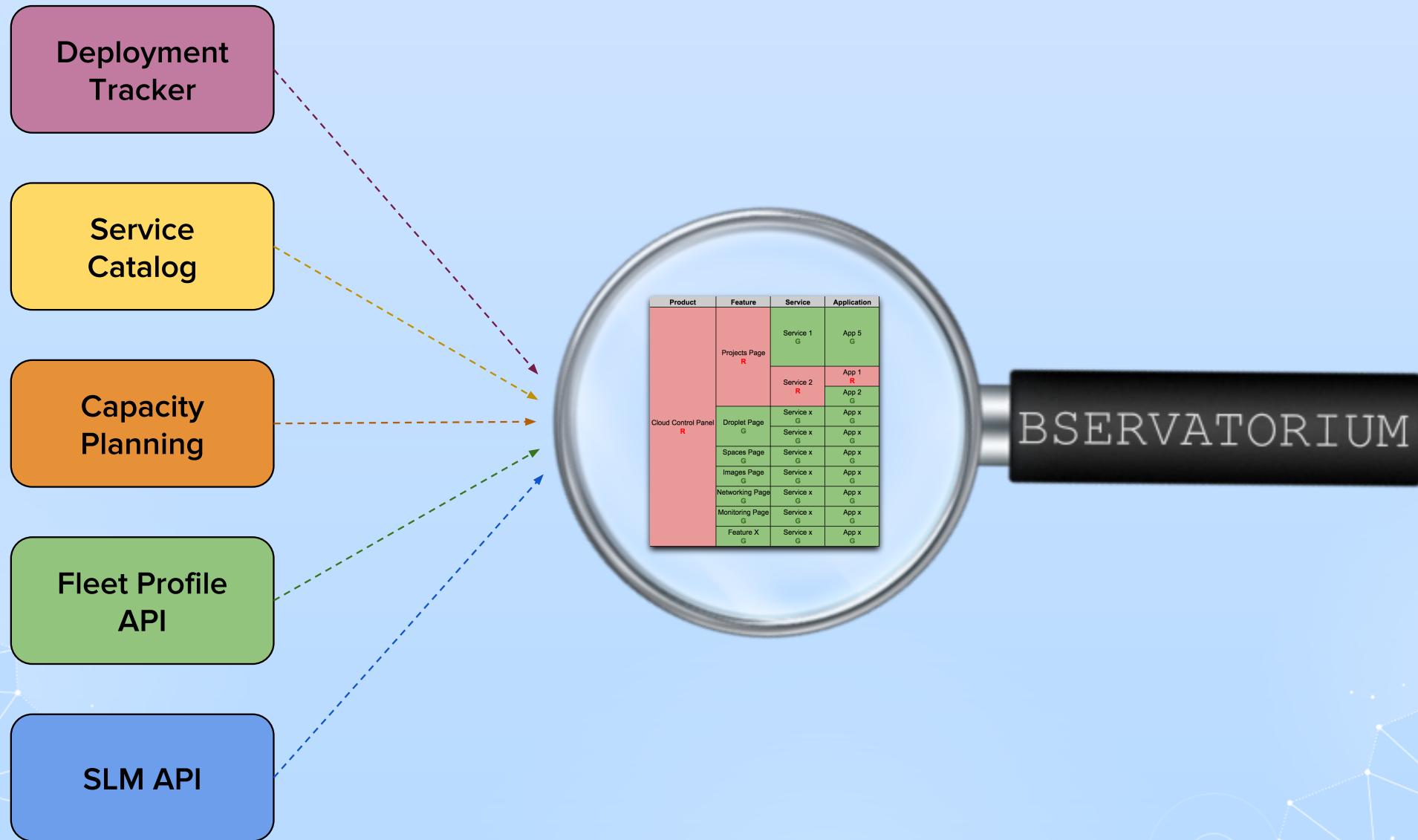
Next Steps



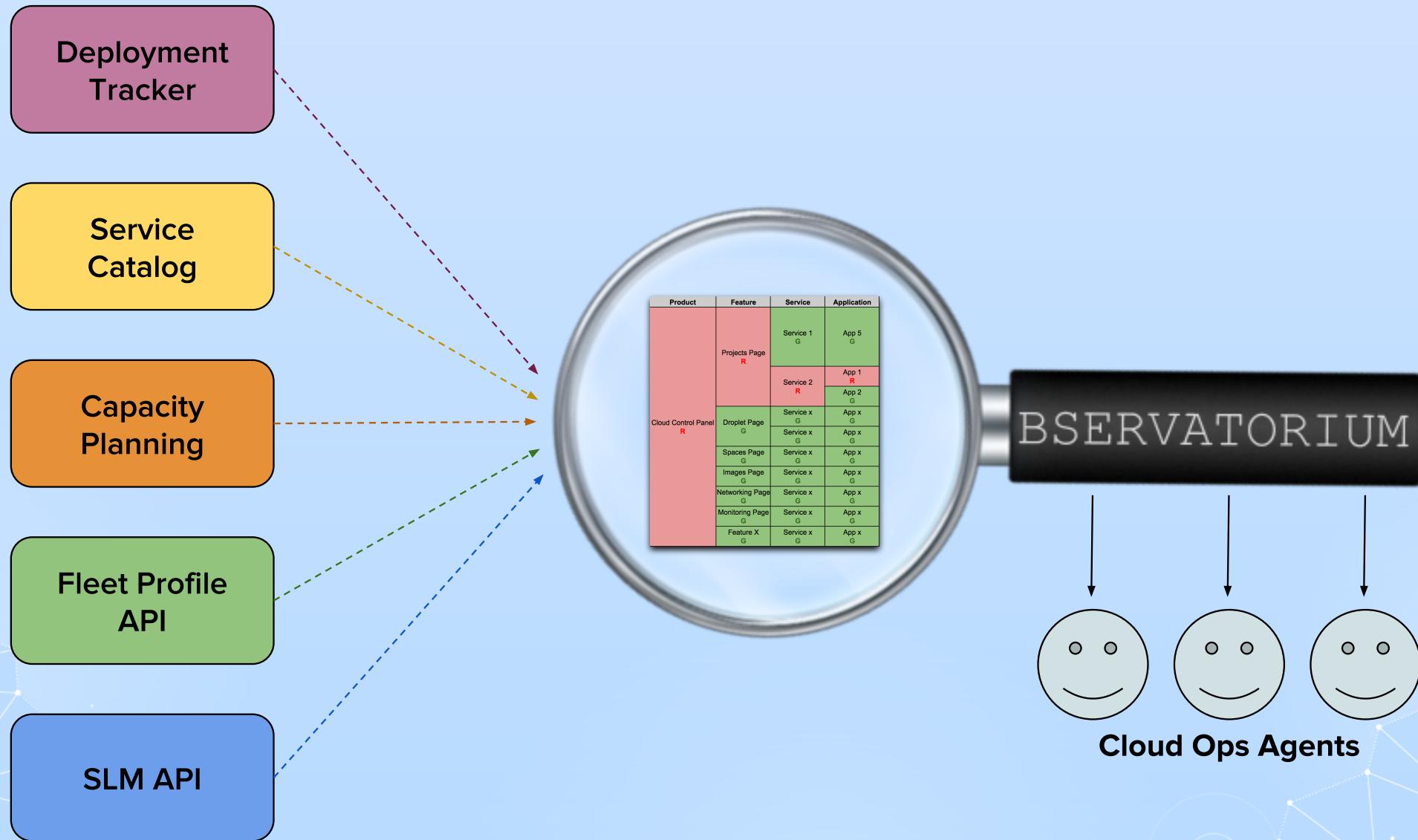
Next Steps



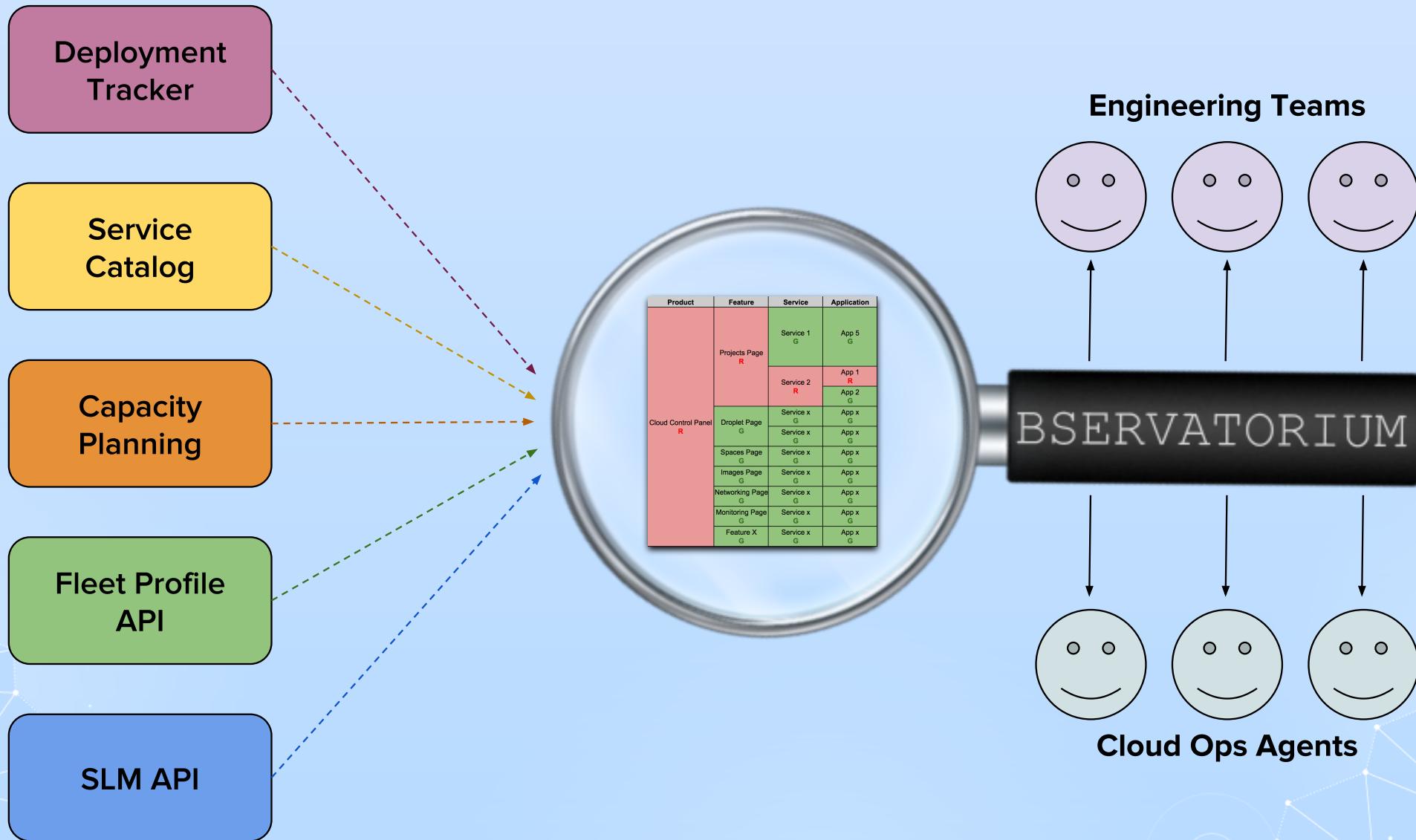
Next Steps



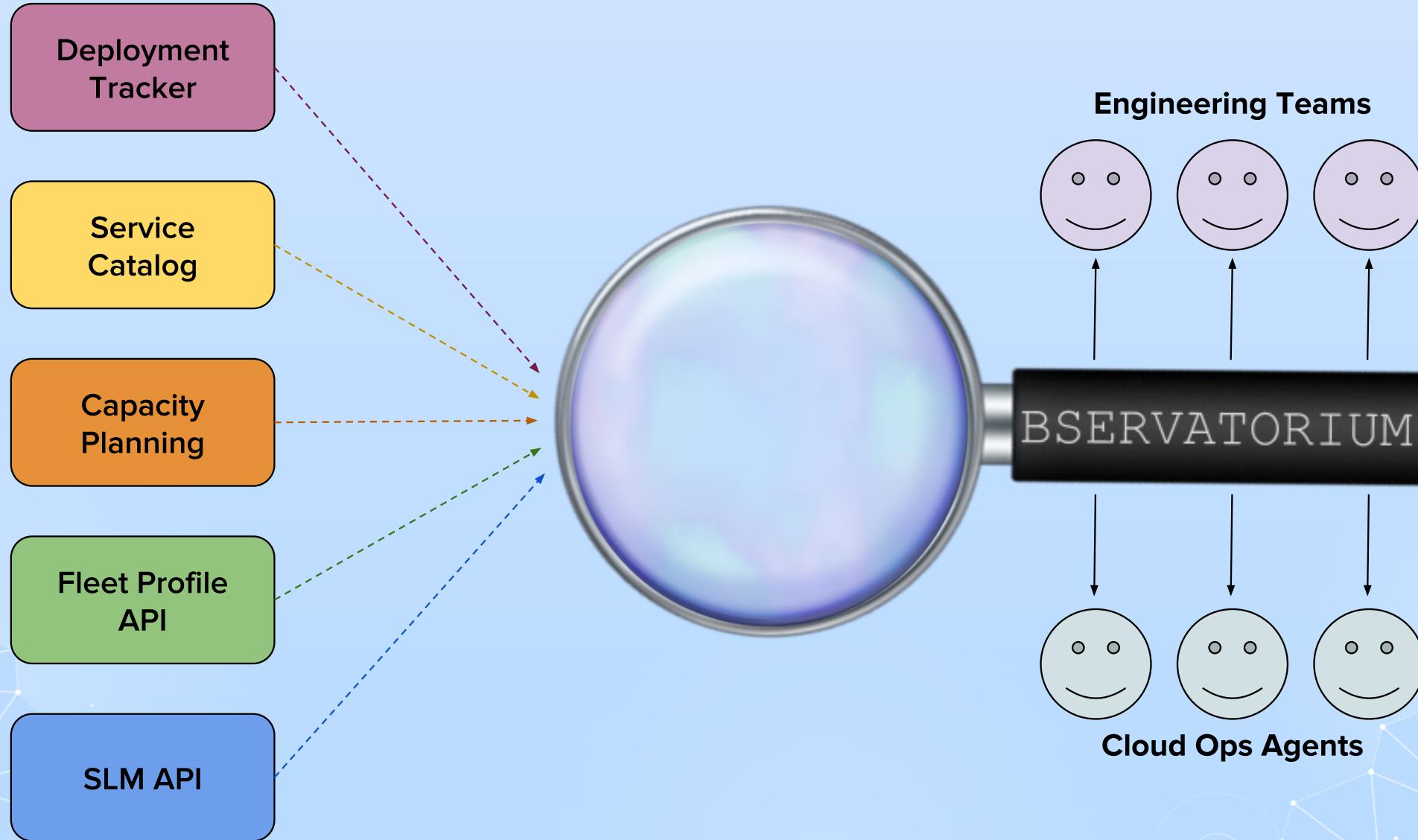
Next Steps



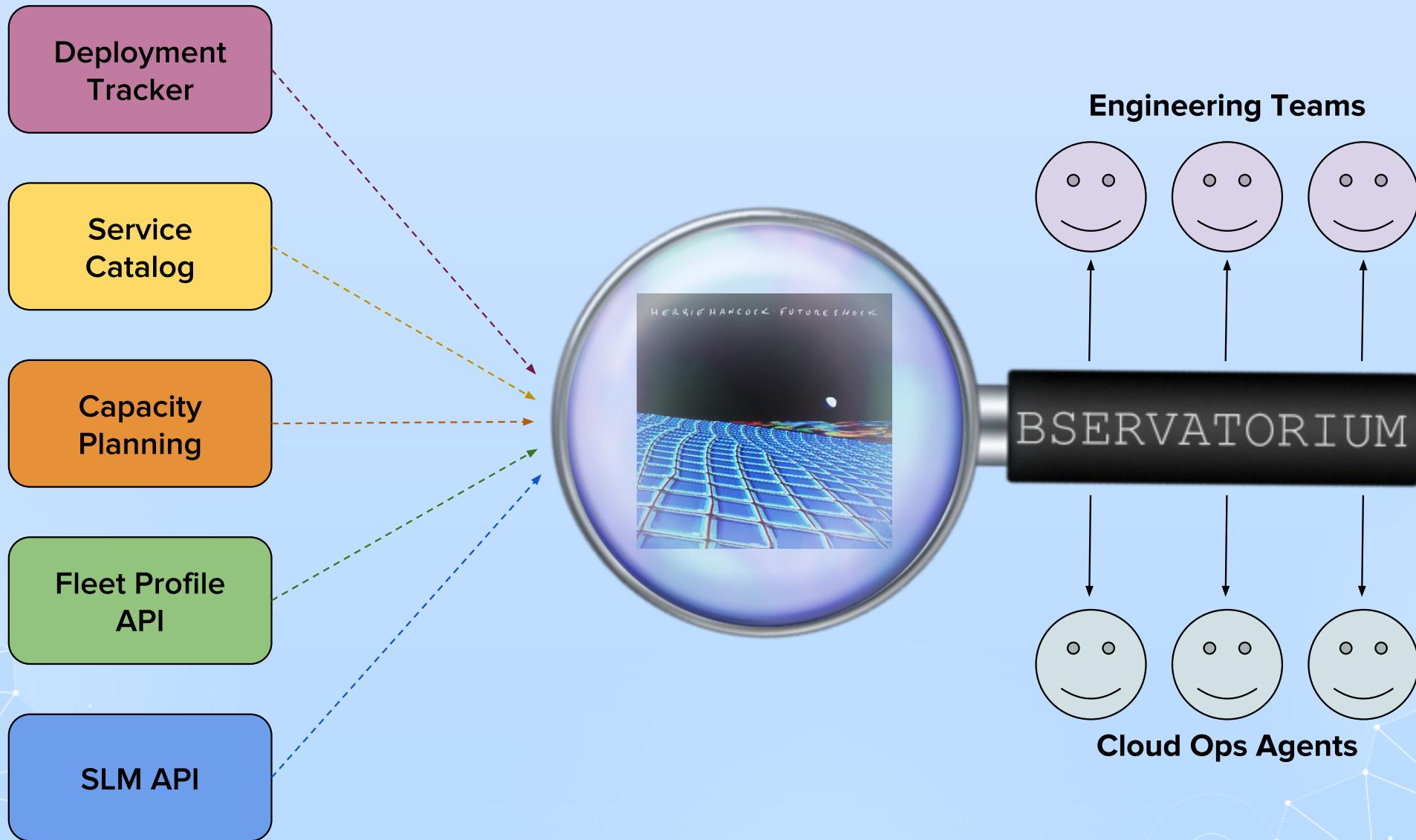
Next Steps



Next Steps



Next Steps



Some final thoughts

- Productize your reliability data
- Mix n' match your OSS tools
- Love 2 the 9s



Thank You!