Intelligent applications and machine learning on OpenShift with radanalytics.io

William Benton and Michael McCune
willb@redhat.com and msm@redhat.com

First things first

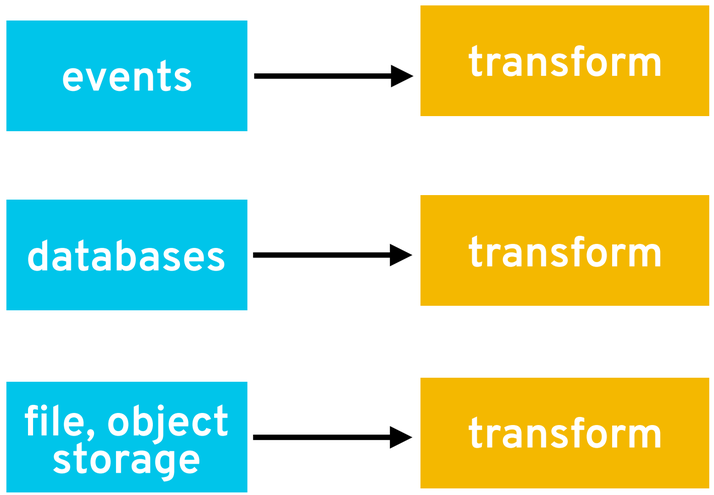
https://console.txl.radanalyticslabs.io:8443/

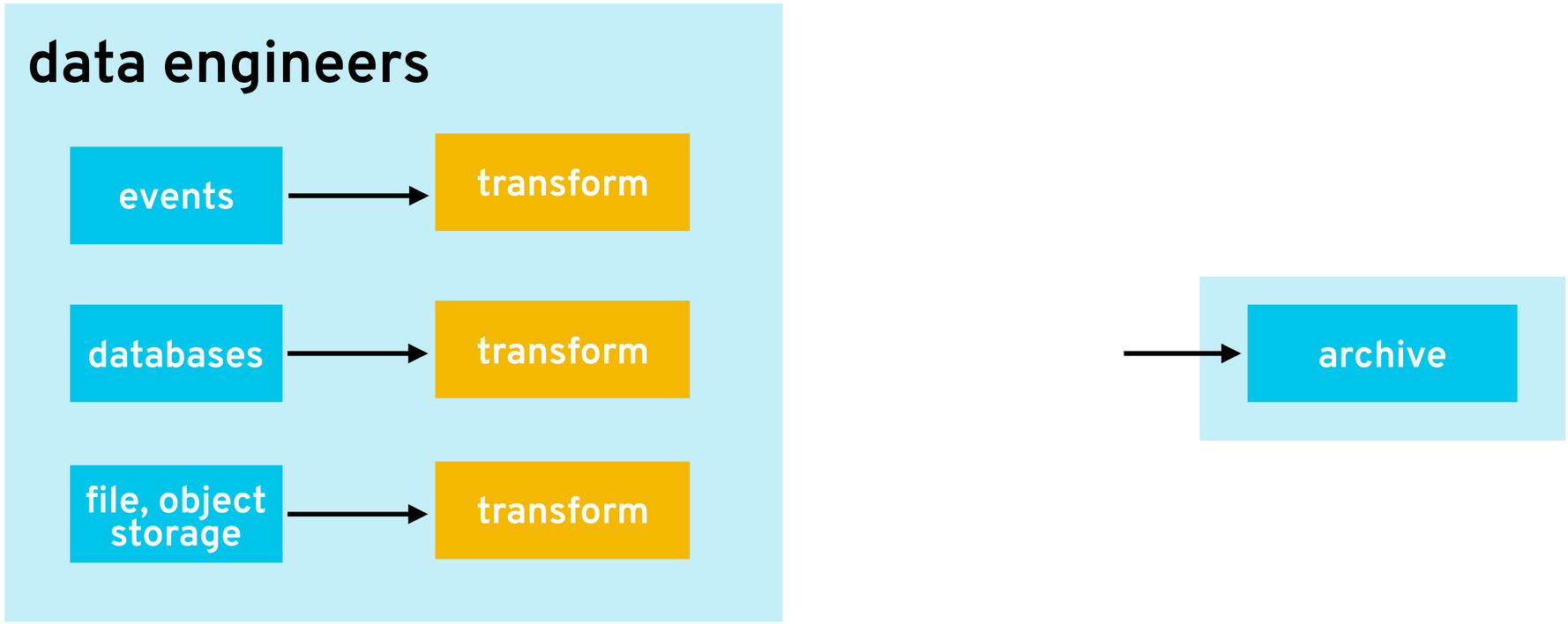
INTELLIGENT APPLICATIONS

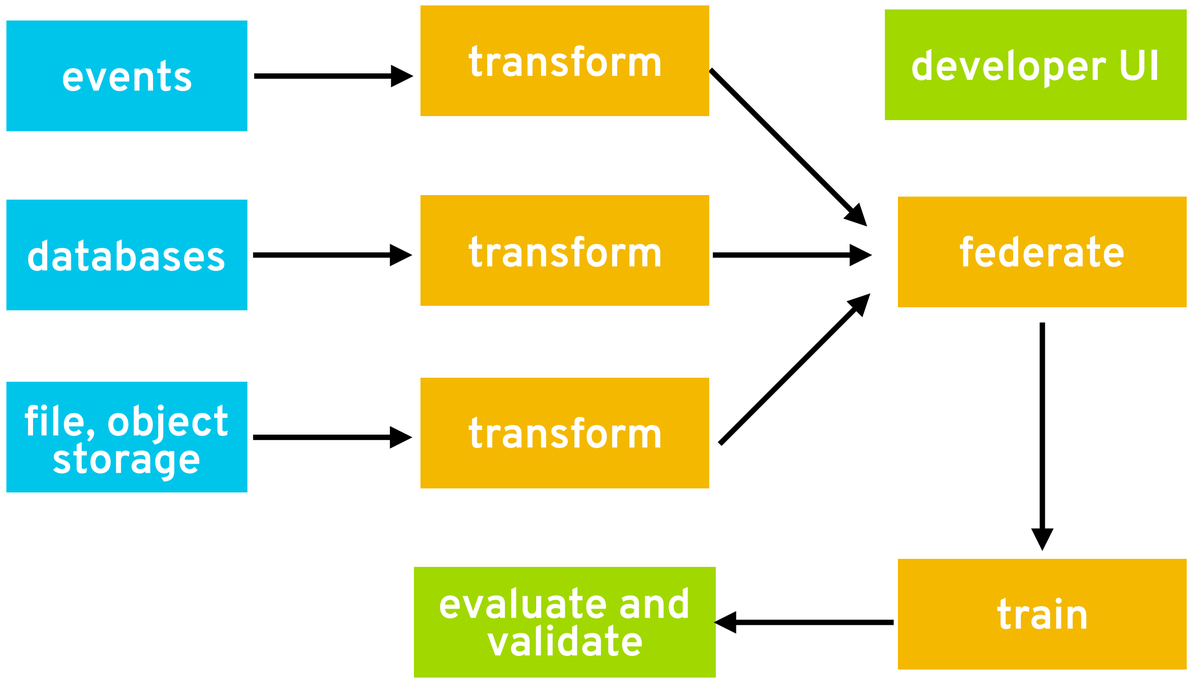
Intelligent applications **collect and learn from data** in order to provide **improved functionality** with **longevity** and **popularity**.

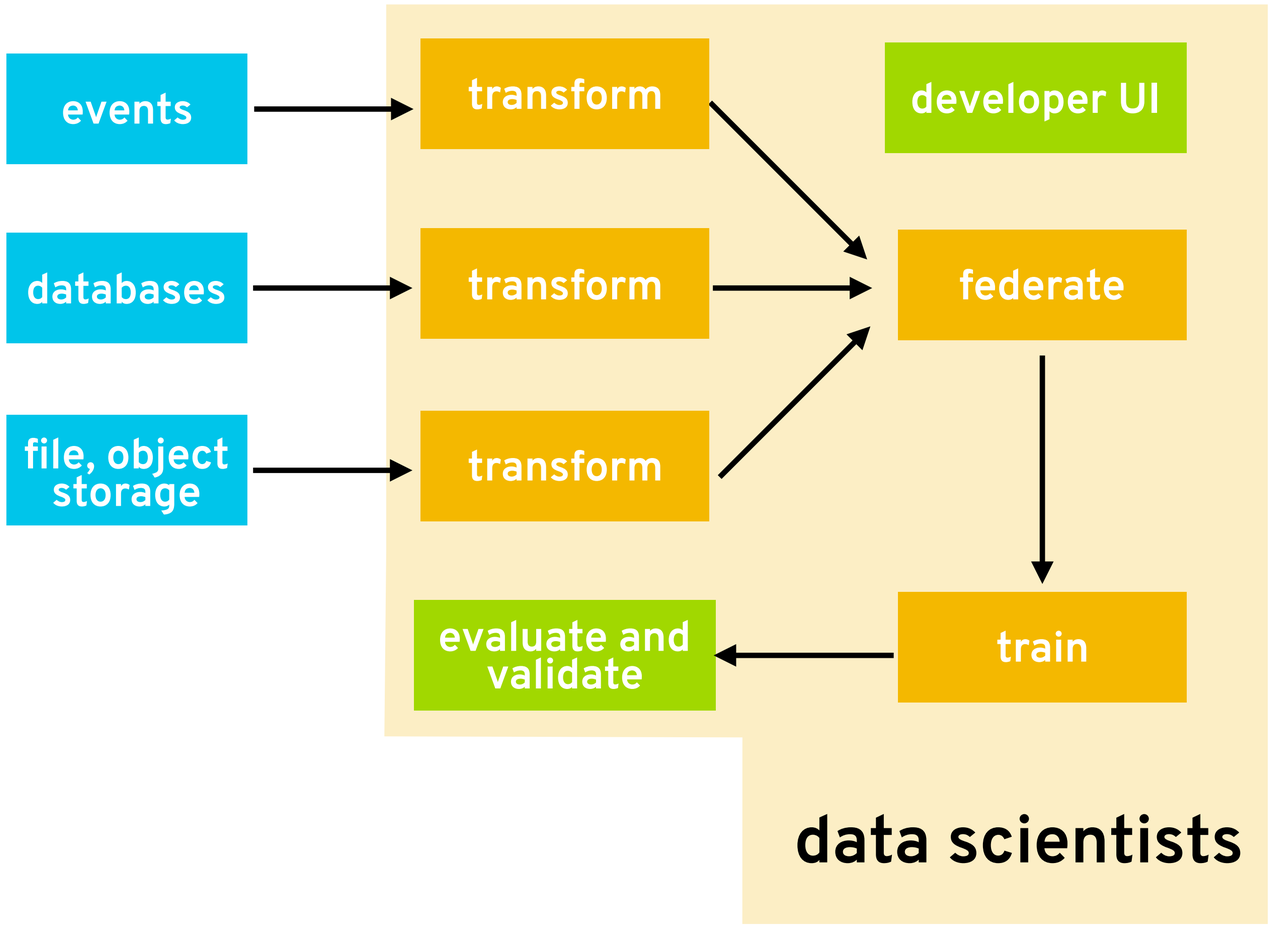
Intelligent applications are **how we put AI into production today**.

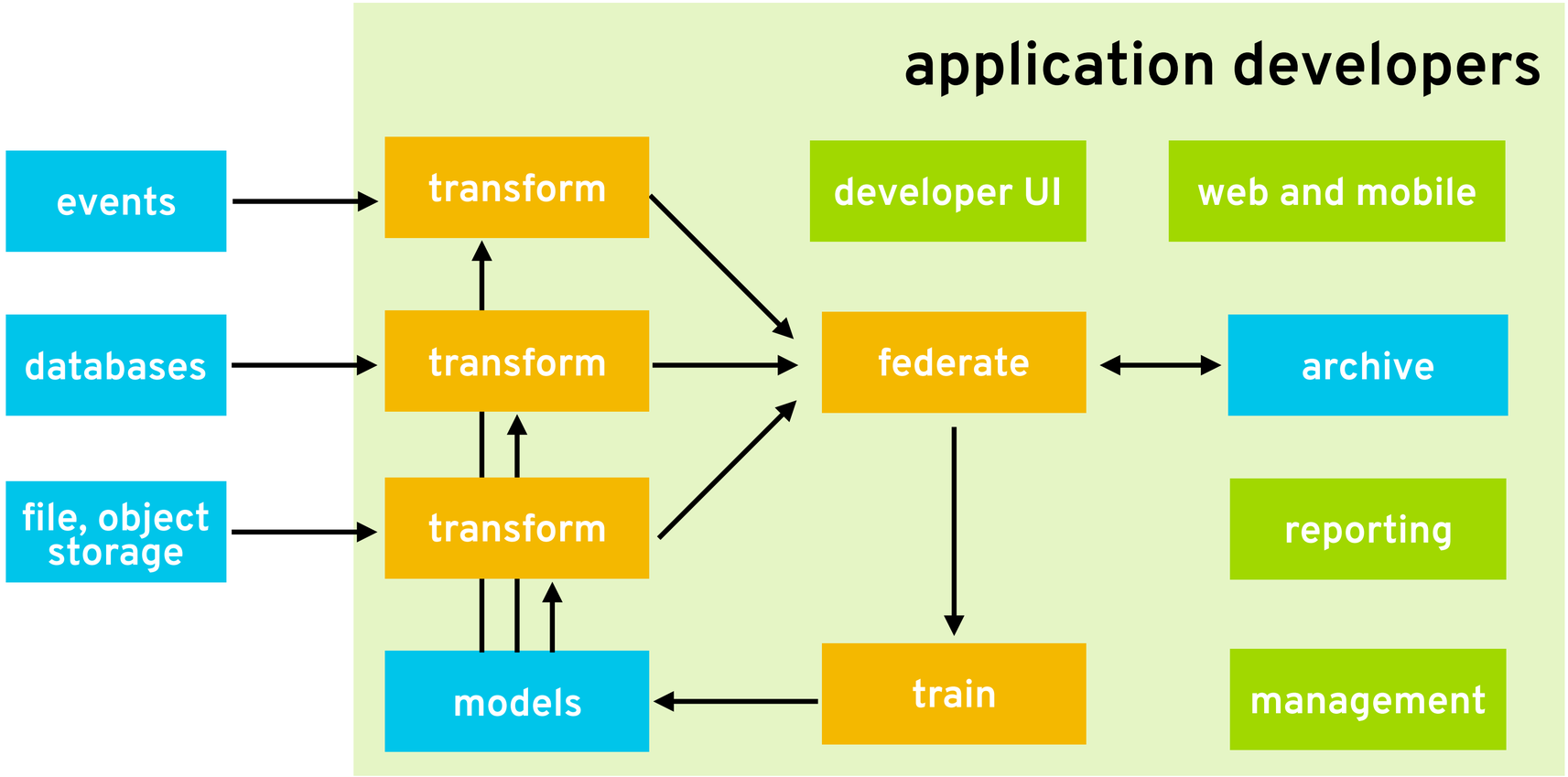
Intelligent applications have driven **advances in AI and machine learning** for the last two decades.

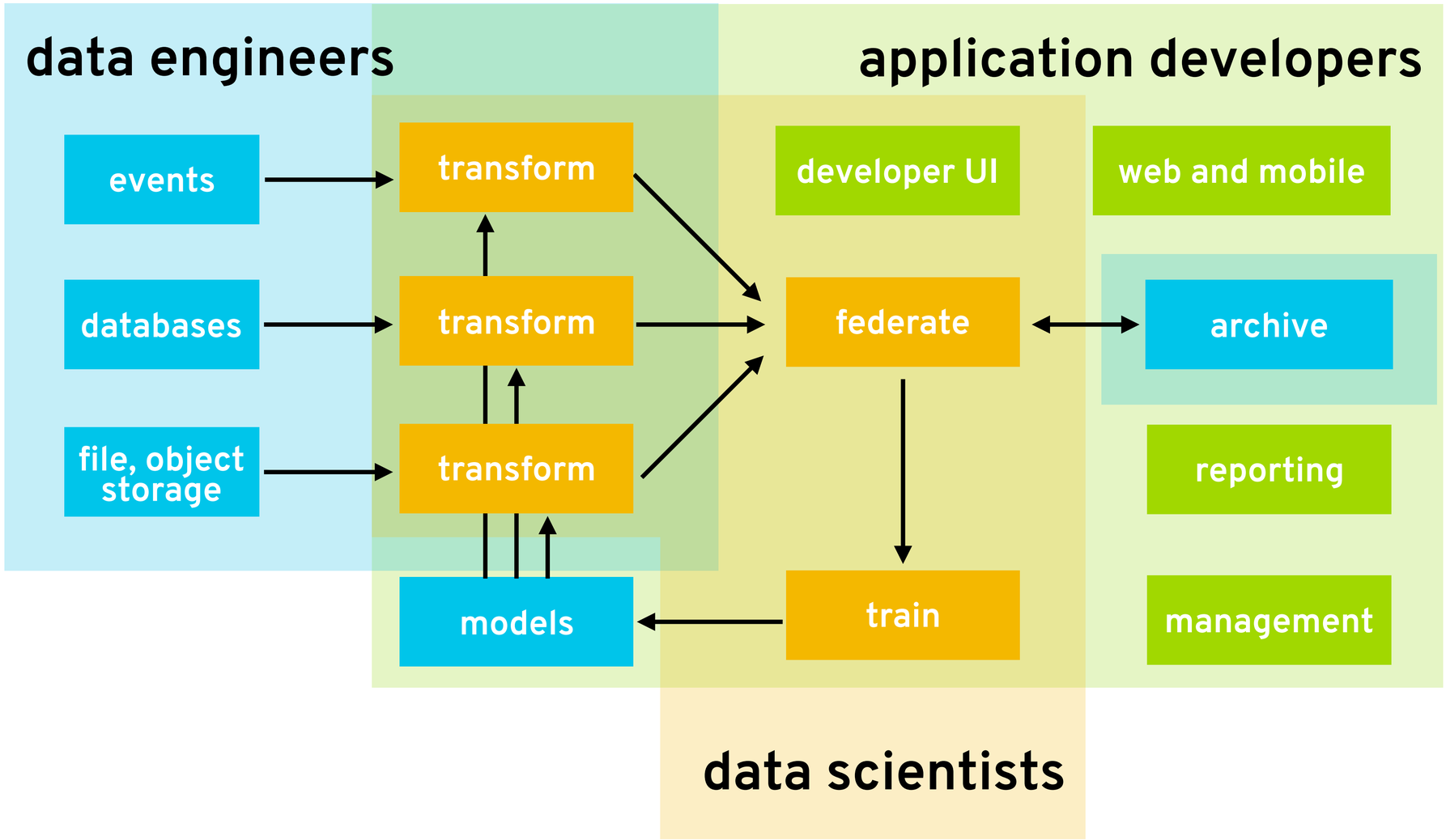

events
transform
databases
transform
file, object storage
transform

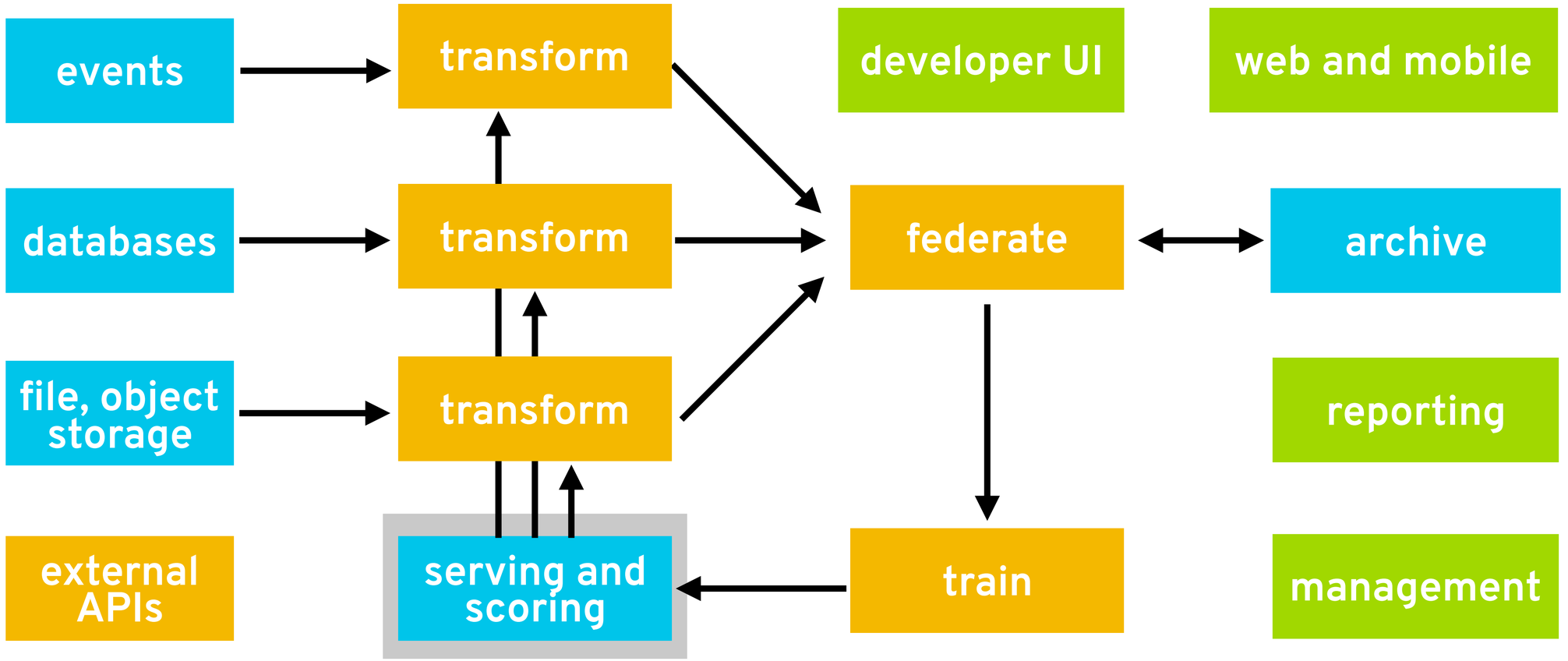

data engineers
events
transform
databases
transform
file, object storage
transform
archive

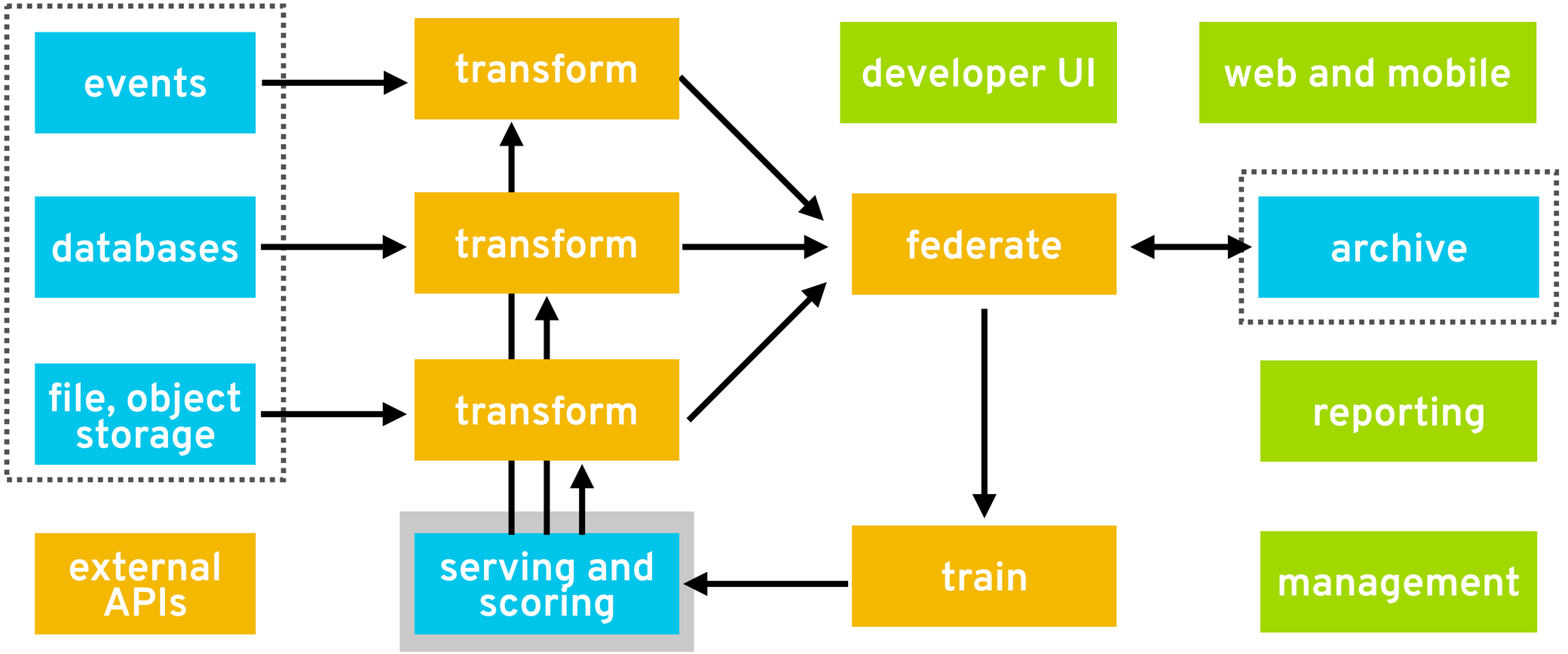

events
databases
file, object storage
transform
transform
transform
developer UI
federate
train
evaluate and validate

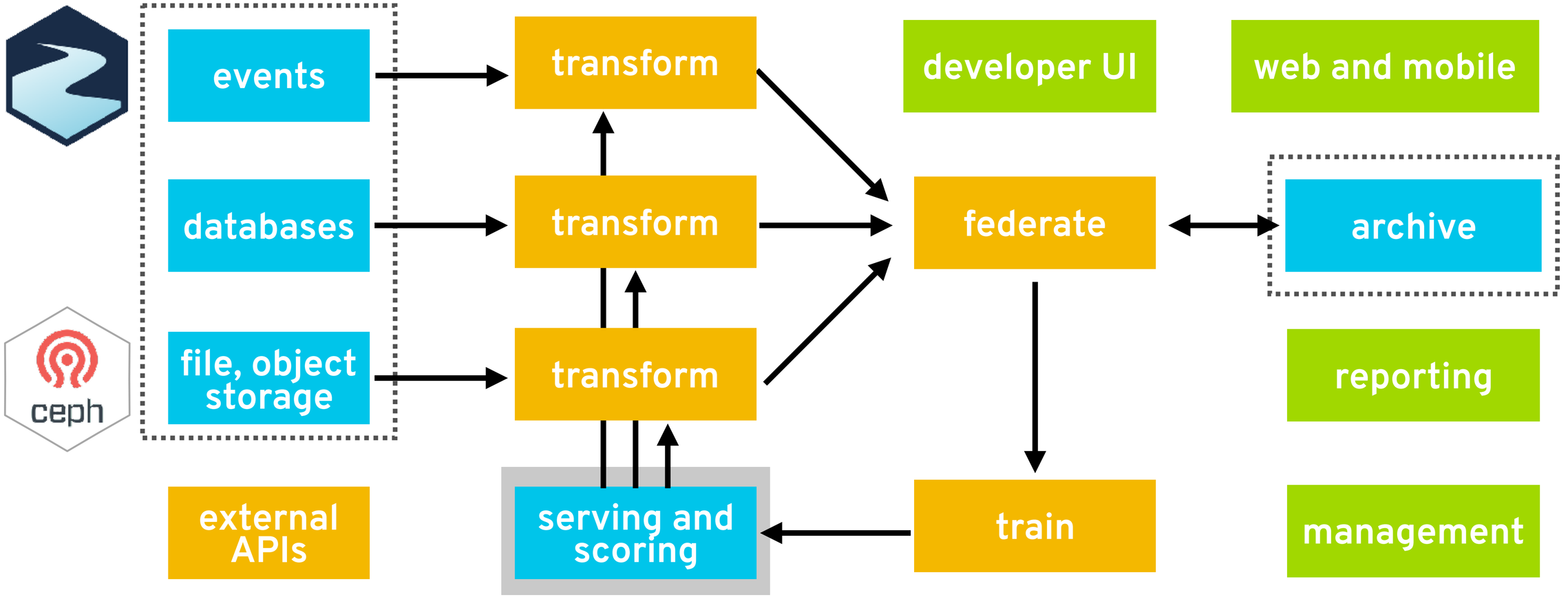

events
databases
file, object storage
transform
transform
transform
developer UI
federate
train
evaluate and validate
data scientists

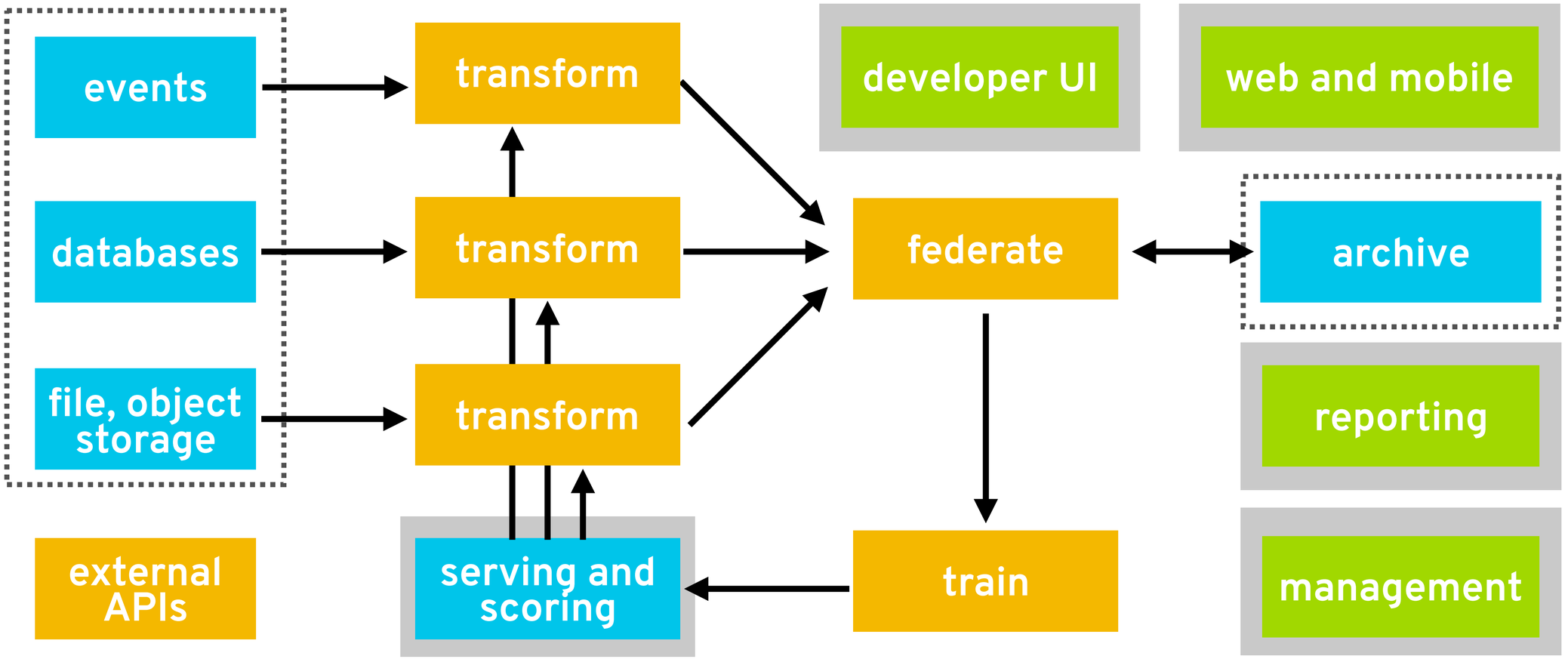

application developers
events
databases
file, object storage
transform
transform
transform
models
developer UI
web and mobile
federate
archive
reporting
train
management

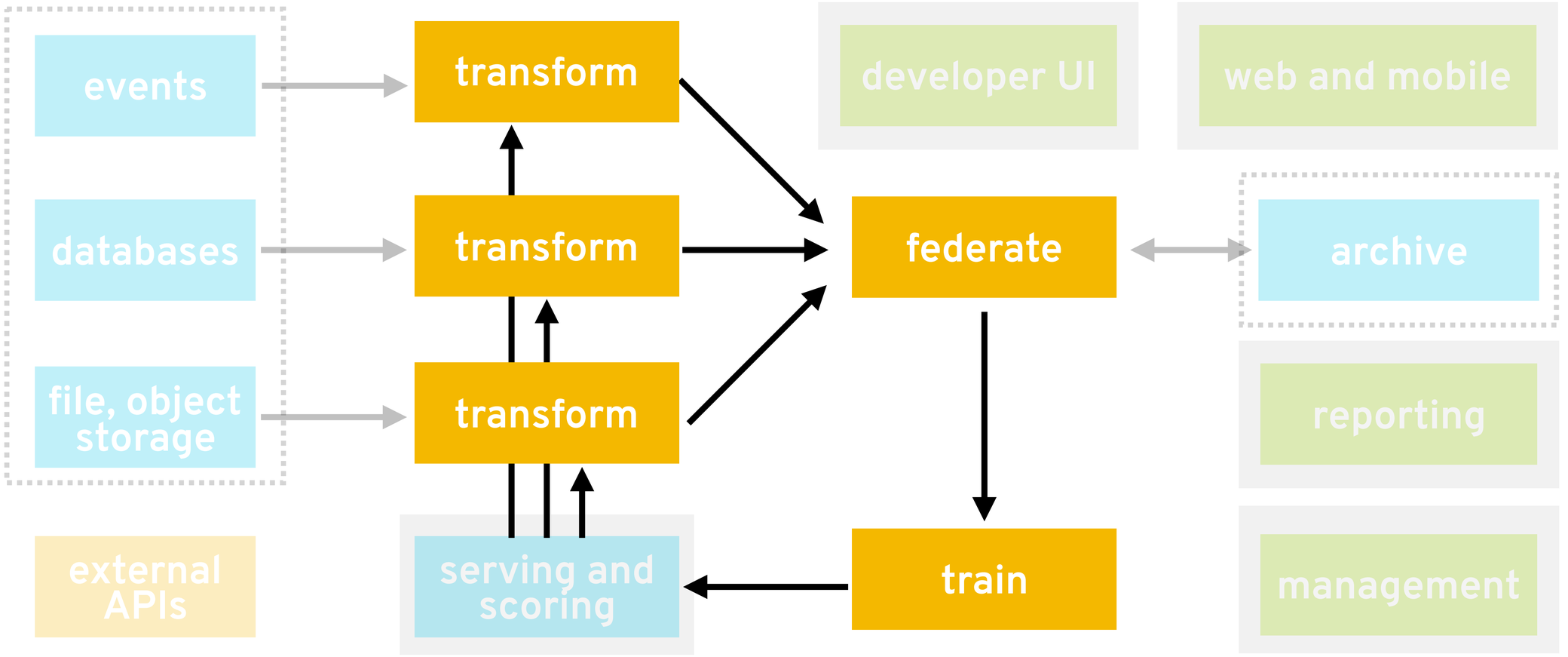

data engineers
application developers
events
databases
file, object storage
transform
transform
transform
models
federate
train
developer UI
web and mobile
archive
reporting
management
data scientists


events
databases
file, object storage
external APIs
transform
transform
transform
serving and scoring
developer UI
federate
train
web and mobile
archive
reporting
management

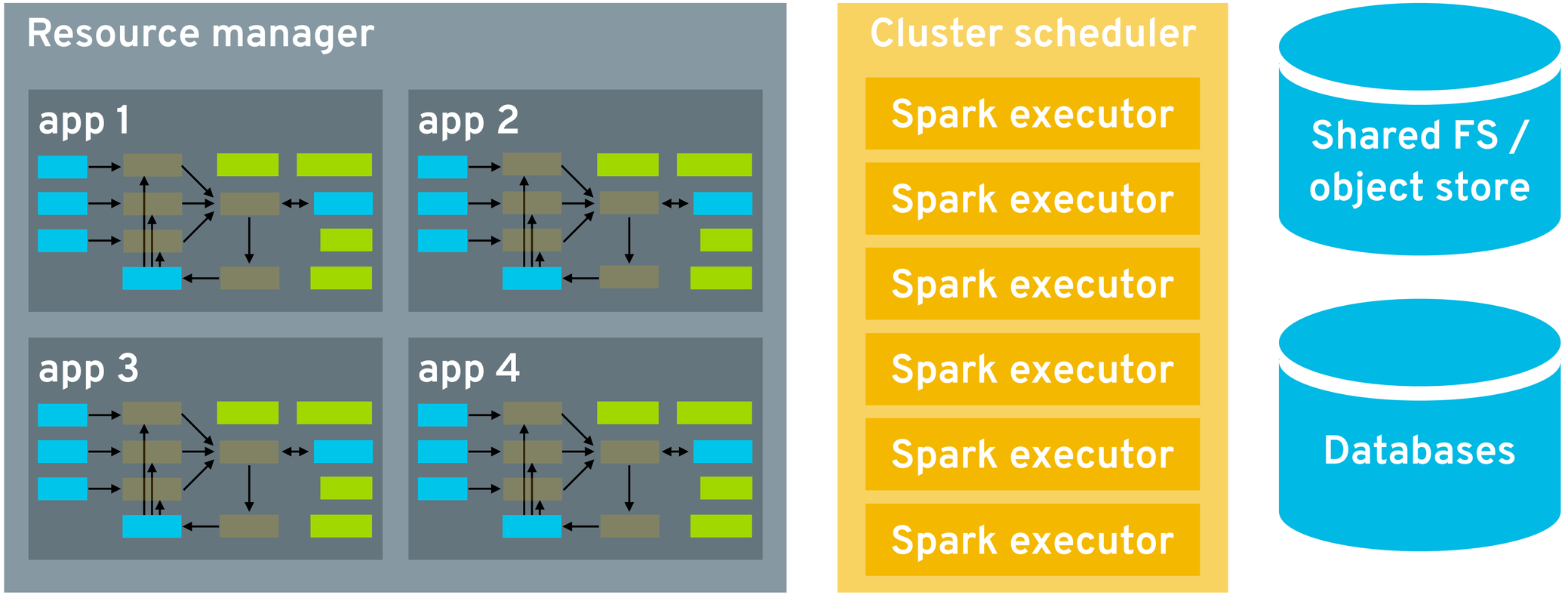

events
databases
file, object storage
external APIs
transform
transform
transform
serving and scoring
developer UI
federate
train
web and mobile
archive
reporting
management


events
databases
file, object storage
external APIs
ceph
transform
transform
transform
serving and scoring
developer UI
federate
train
web and mobile
archive
reporting
management

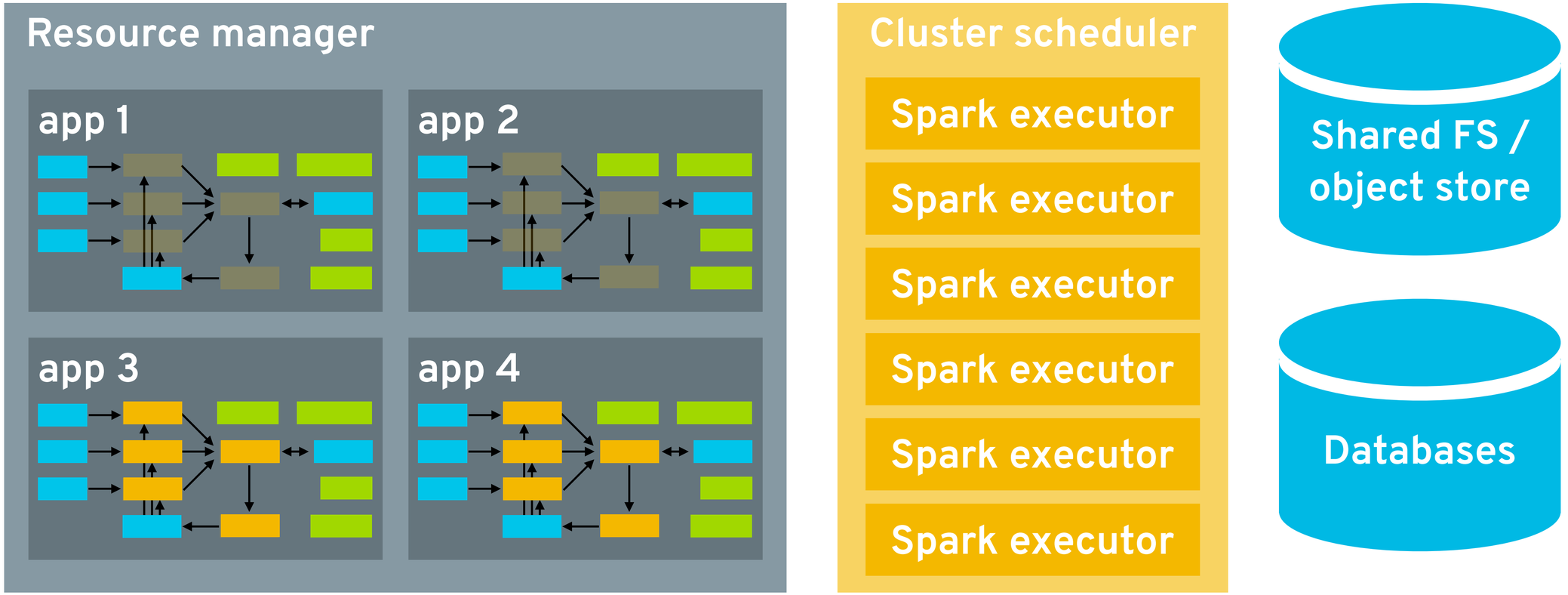

events
databases
file, object storage
external APIs
transform
transform
transform
serving and scoring
federate
train
developer UI
web and mobile
archive
reporting
management


events
databases
file, object storage
external APIs
transform
transform
transform
serving and scoring
federate
train
developer UI
web and mobile
archive
reporting
management

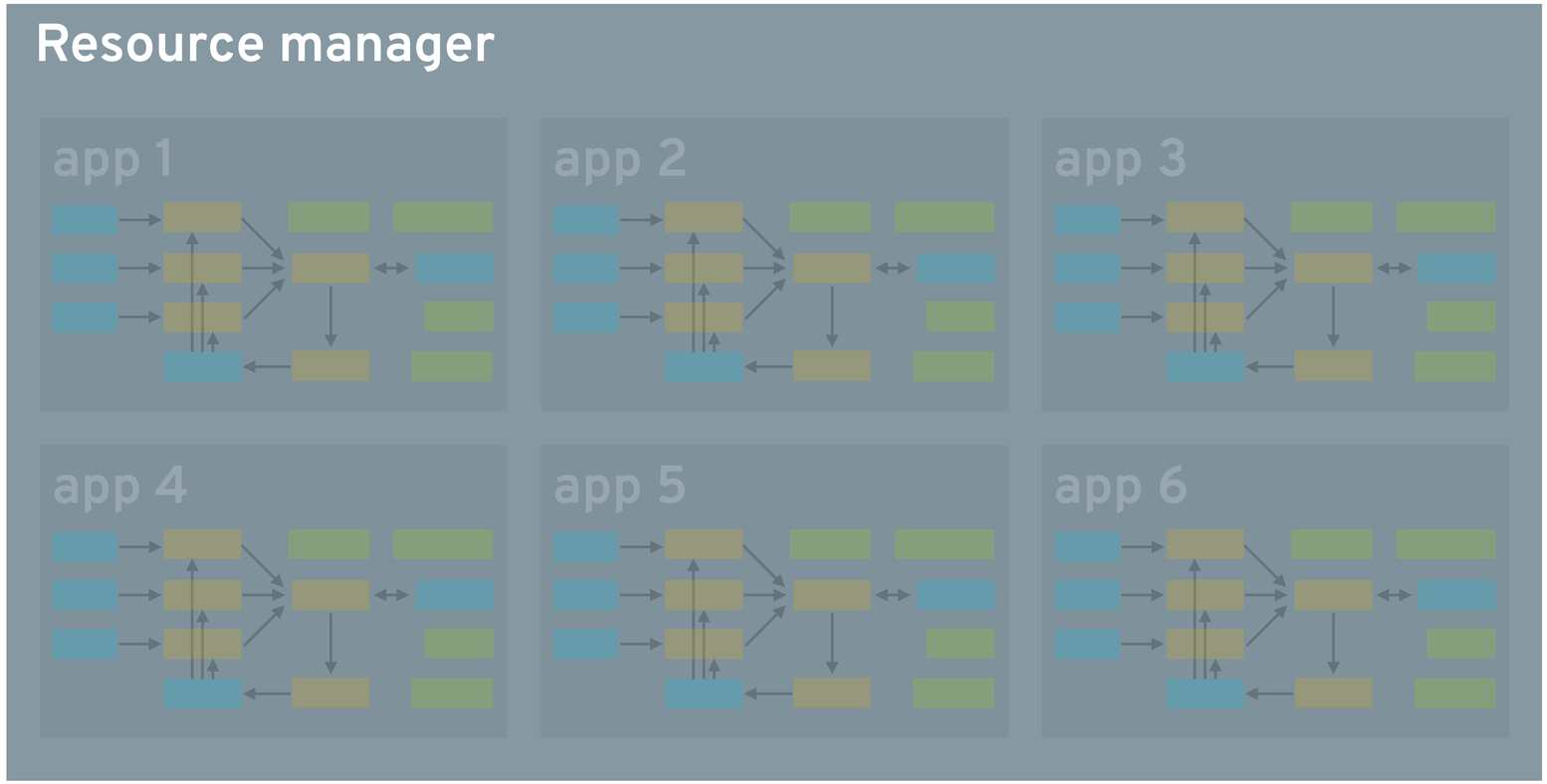
Multitenant compute clusters



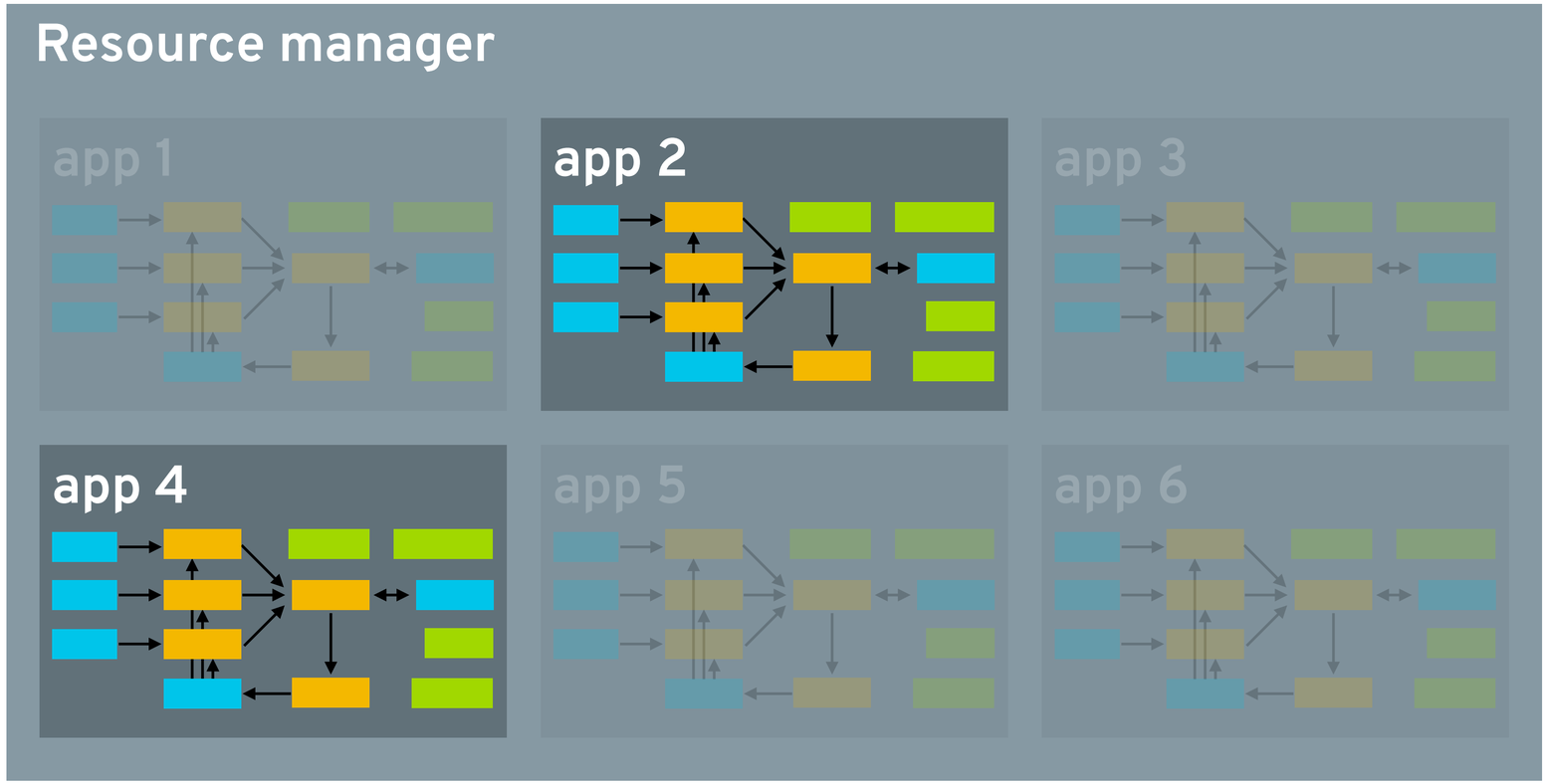
Multitenant compute clusters



One cluster per application



One cluster per application



A trivial Spark example

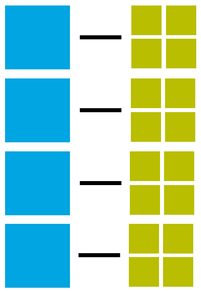
```
file = sc.textFile("file://...")

counts = file.flatMap(lambda l: l.split(" "))
              .map(lambda w: (w, 1))
              .reduceByKey(lambda x, y: x + y)

# computation actually occurs here
counts.saveAsTextFile("file://...")
```

A trivial Spark example

```
file = sc.textFile("file://...")

counts = file.flatMap(lambda l: l.split(" "))
              .map(lambda w: (w, 1))
              .reduceByKey(lambda x, y: x + y)

# computation actually occurs here
counts.saveAsTextFile("file://...")
```

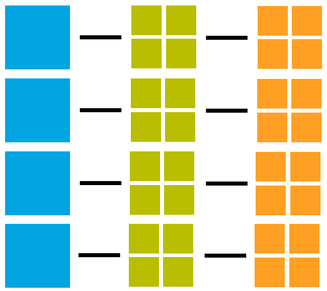
A trivial Spark example

```
file = sc.textFile("file://...")

counts = file.flatMap(lambda l: l.split(" "))
                  .map(lambda w: (w, 1))
                  .reduceByKey(lambda x, y: x + y)

# computation actually occurs here
counts.saveAsTextFile("file://...")
```



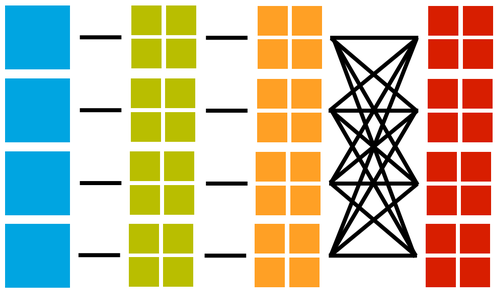
A trivial Spark example

```
file = sc.textFile("file://...")

counts = file.flatMap(lambda l: l.split(" "))
              .map(lambda w: (w, 1))
              .reduceByKey(lambda x, y: x + y)

# computation actually occurs here
counts.saveAsTextFile("file://...")
```



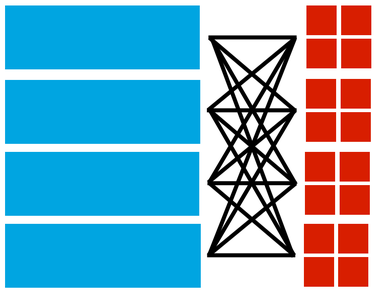
A trivial Spark example

```
file = sc.textFile("file://...")

counts = file.flatMap(lambda l: l.split(" "))
              .map(lambda w: (w, 1))
              .reduceByKey(lambda x, y: x + y)

# computation actually occurs here
counts.saveAsTextFile("file://...")
```

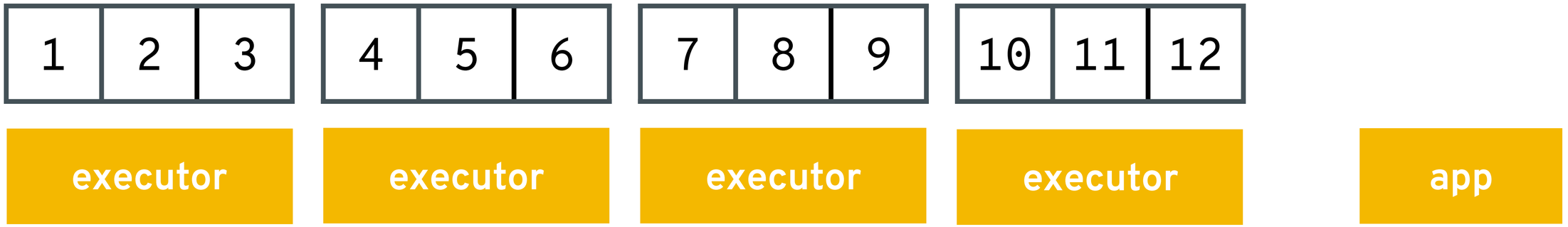


A trivial Spark example

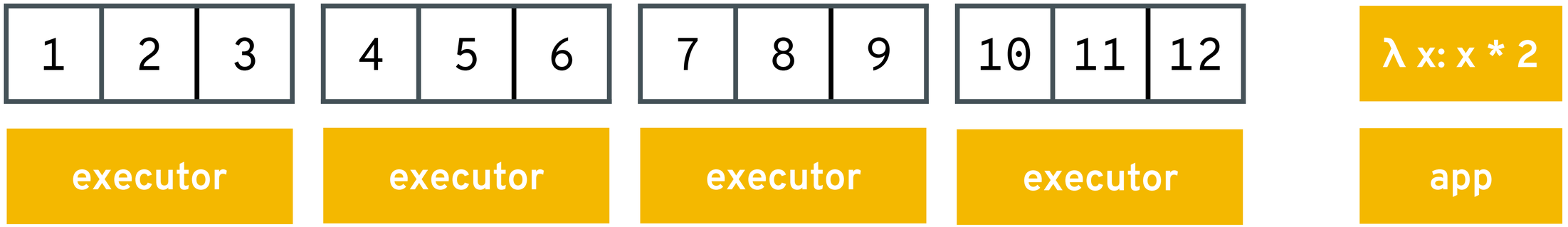
```
file = sc.textFile("file://...")

counts = file.flatMap(lambda l: l.split(" "))
              .map(lambda w: (w, 1))
              .reduceByKey(lambda x, y: x + y)

# computation actually occurs here
counts.saveAsTextFile("file://...")
```



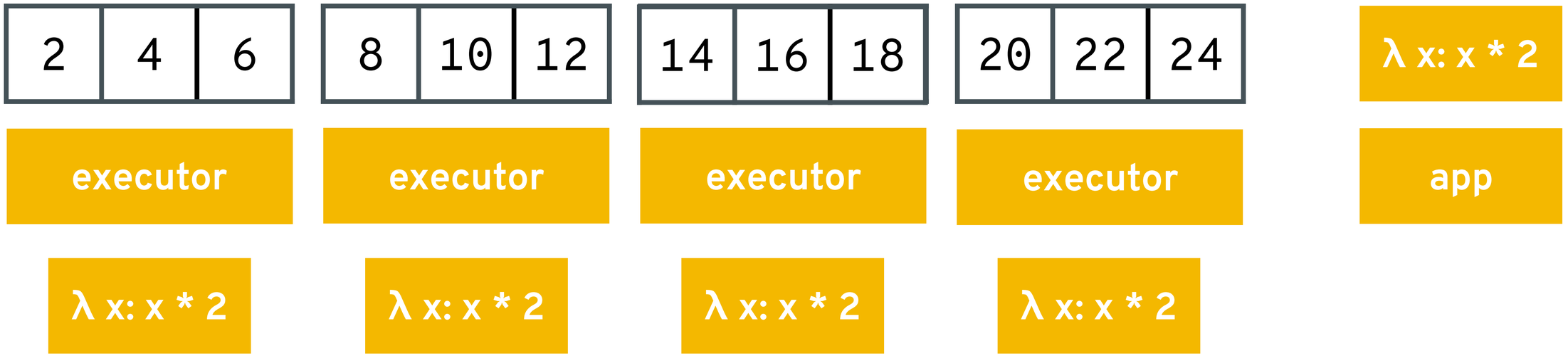
Data-processing microservices



Data-processing microservices

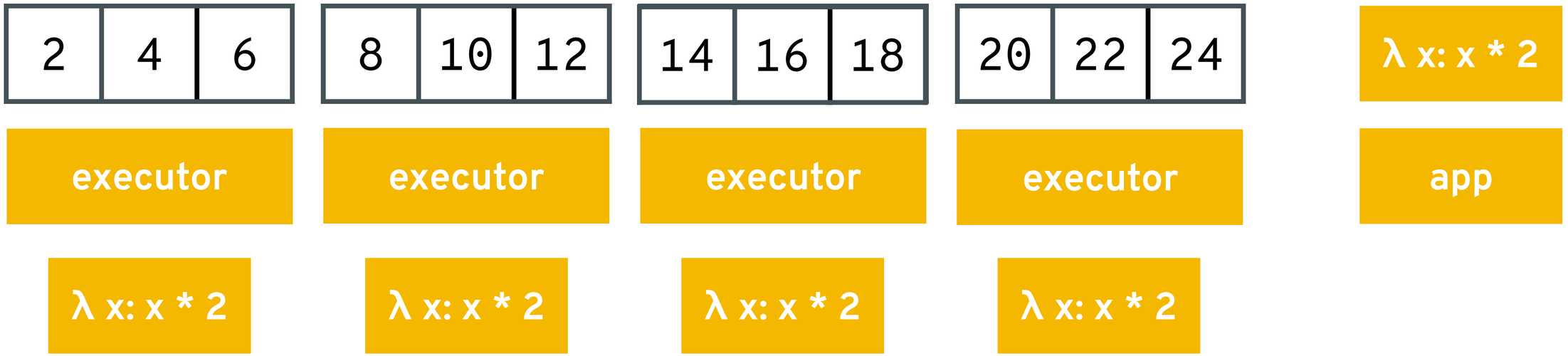


Data-processing microservices



Data-processing microservices

Data-processing microservices



Graph | SQL | ML | Streaming

Spark core (collections and scheduler)

ad hoc | Kubernetes | Mesos | YARN

RADANALYTICS.IO

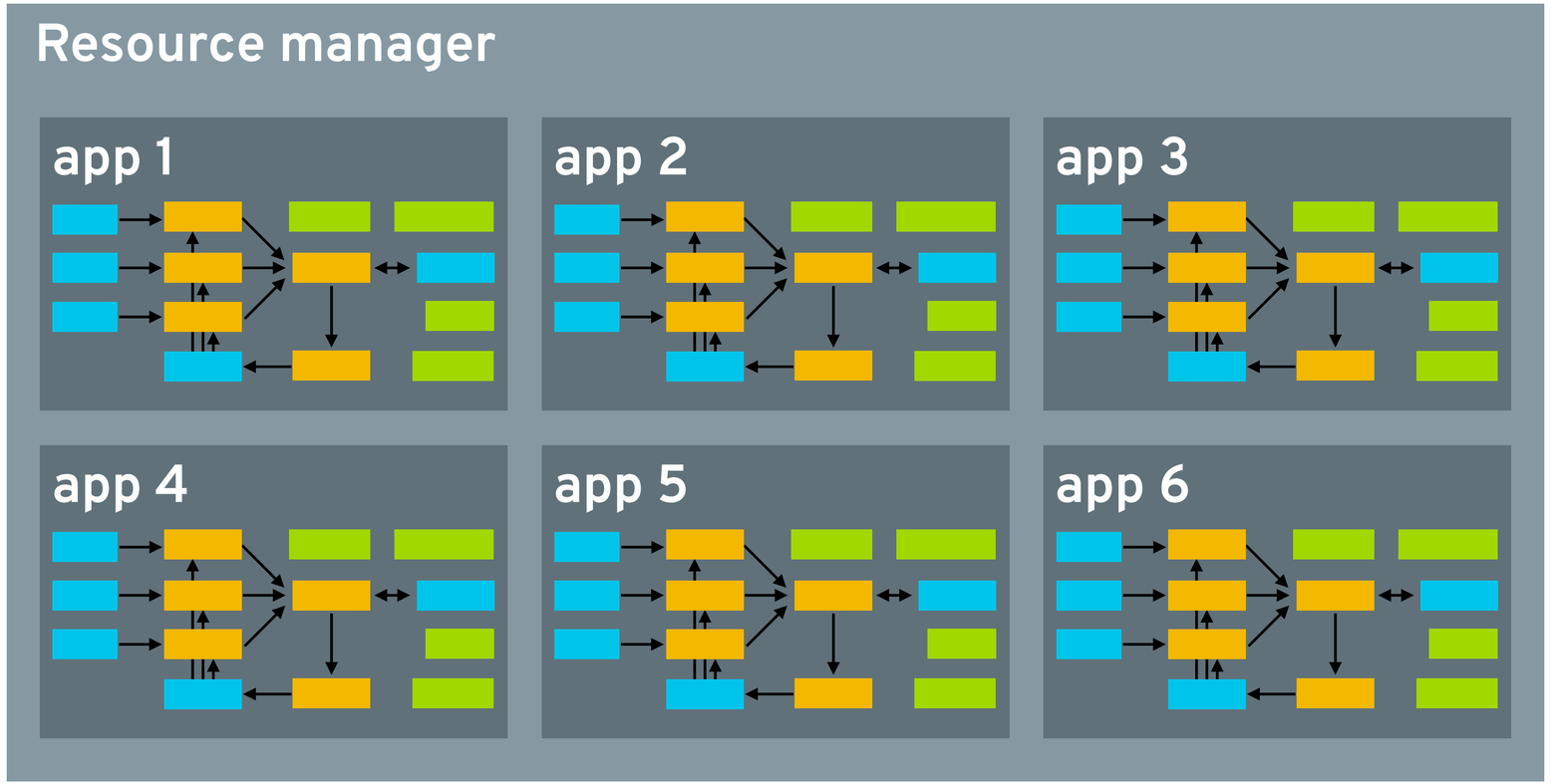
radanalytics.io

An **open-source community** enabling **intelligent applications** on **OpenShift**

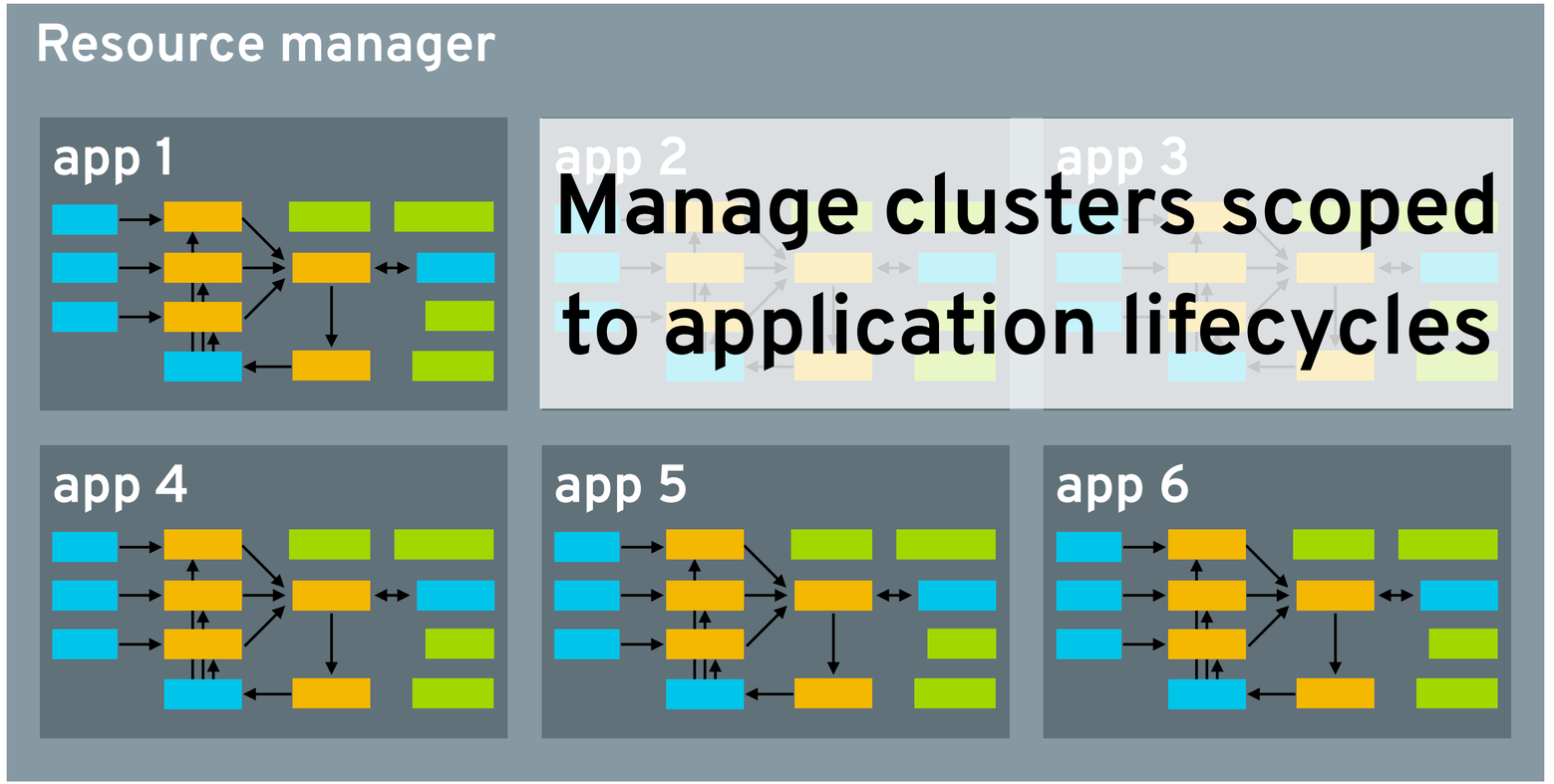
Tooling to manage **Apache Spark**, **Jupyter notebooks**, and **TensorFlow** **training** and **model serving**

Numerous example applications

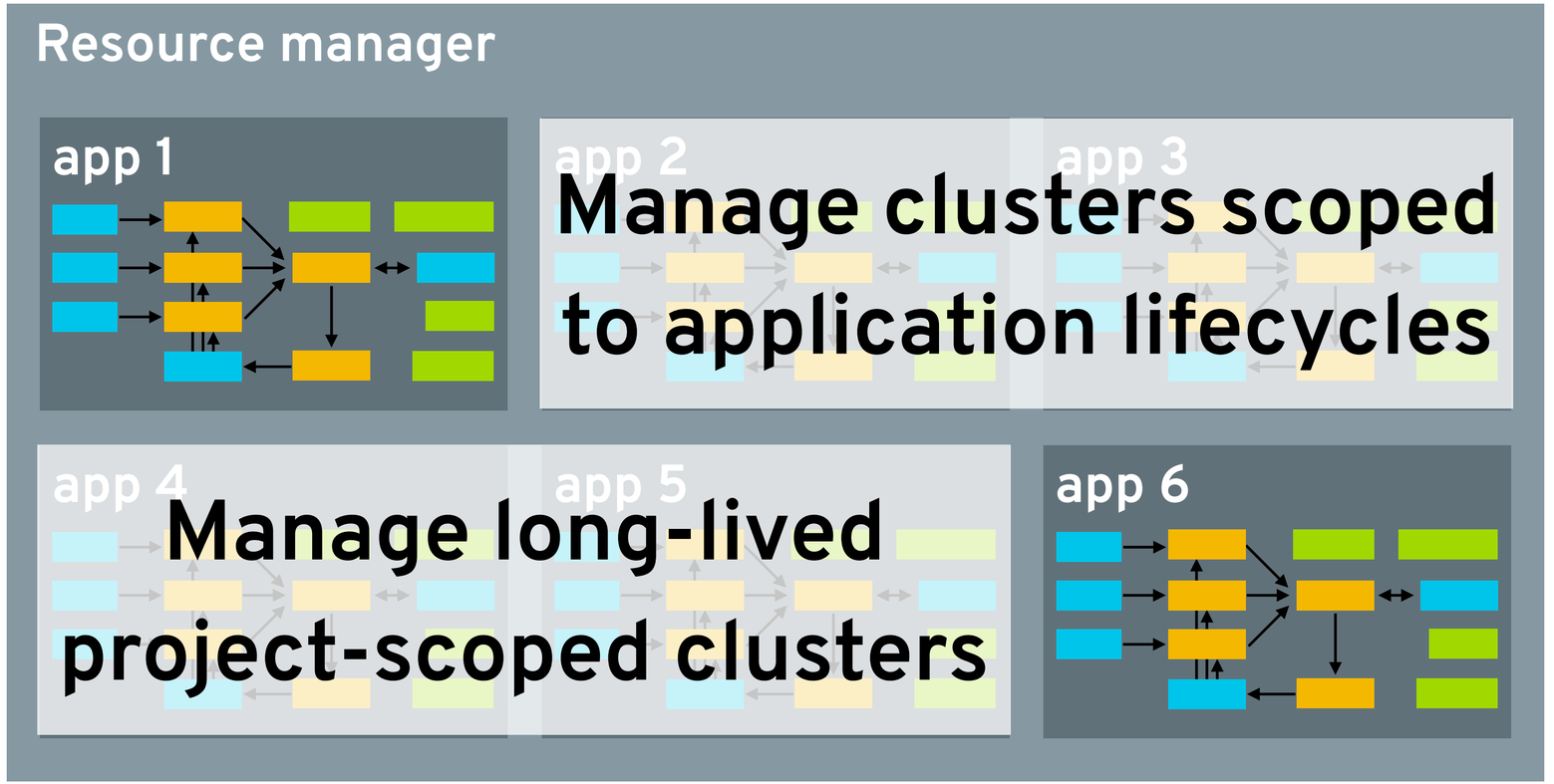
Spark management tooling



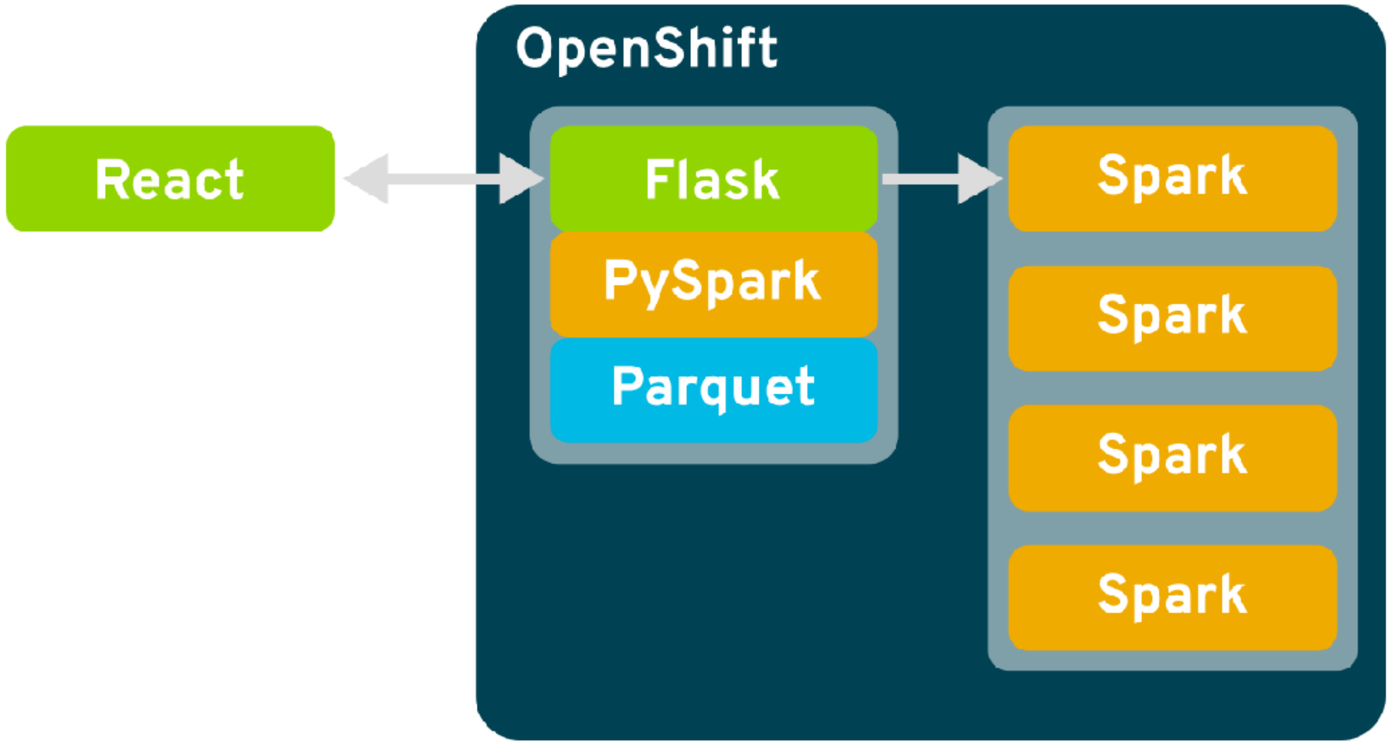
Spark management tooling



Spark management tooling

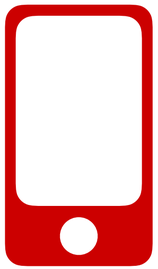


MIGRATING FROM NOTEBOOK TO APPLICATION



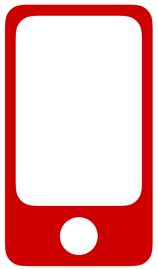
https://github.com/radanalyticsio/var-sandbox

```
def simstep(pf, params, prng):
  def daily_return(sym):
      mean, stddev = params[sym]
      change = (prng.normalvariate(mean, stddev) + 100) / 100.0
      return change
  return dict([(s, daily_return(s) * v) for s, v in pf.items()])
```



```
def simstep(pf, params, prng):
    """Simulate a single step of activity for a stock.

    This function takes a dictionary of stock value data, a dictionary
    containing stock prediction models, and a random.Random instance to
    generate new variances from. It will return a dictionary with the
    updated, randomly predicted, values for the stocks indexed by
    symbol.
    """
    def daily_return(sym):
        mean, stddev = params[sym]
        change = (prng.normalvariate(mean, stddev) + 100) / 100.0
        return change
    return {s: daily_return(s) * v for s, v in pf.items()}
```

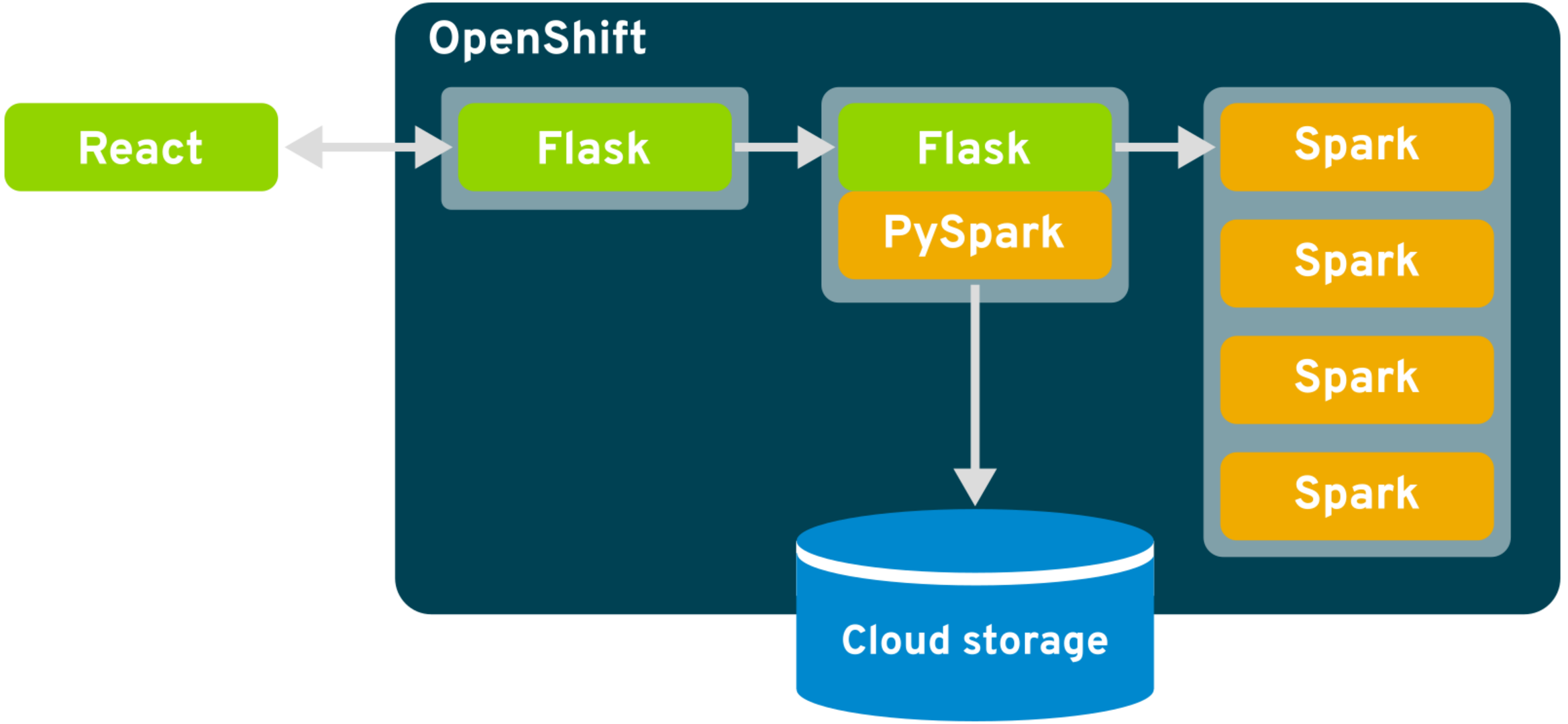
```
def simulate(seed, pf, params, days):
    from random import Random
    prng = Random()
    prng.seed(seed)
    pf = pf.copy()

    for day in range(days):
        pf = simstep(pf, params, prng)
    return pf
```

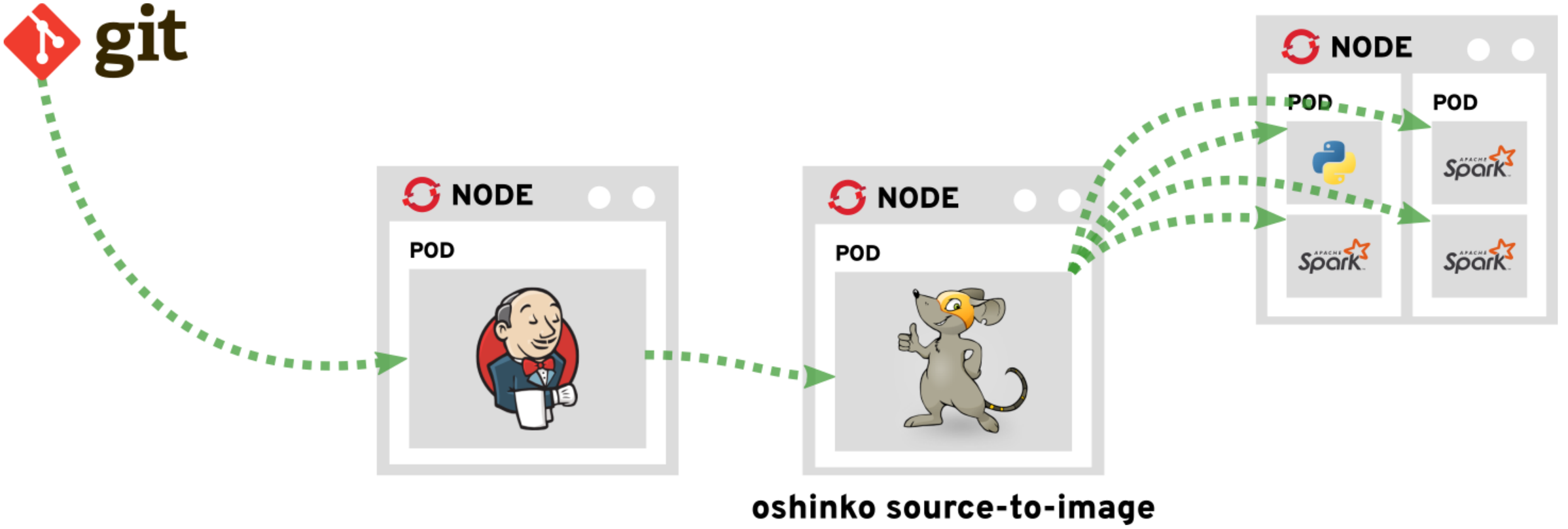


```
def simulate(seed, pf, params, days):
    """Simulate a number of days worth of stock changes.

    This function accepts a seed for the randomizer, a dictionary of
    stock value data, a dictionary of stock prediction models, and a
    number of days to simulate. It will return a dictionary with the
    updated stock value predictions indexed by symbol.
    """
    prng = random.Random()
    prng.seed(seed)
    pf = pf.copy()
    for day in range(days):
        pf = simstep(pf, params, prng)
    return pf
```

DEPLOYING THE APPLICATION

BUILDING TOWARDS PRODUCTION


OpenShift
React
Flask
Flask
PySpark
Spark
Spark
Spark
Spark
Cloud storage


git
NODE
POD
NODE
POD
NODE
POD
POD
Spark
Spark
Spark
oshinko source-to-image

THANKS!

https://radanalytics.io
https://github.com/radanalyticsio/workshop
https://github.com/radanalyticsio/var-sandbox

willb@redhat.com and msm@redhat.com
@willb and @FOSSJunkie