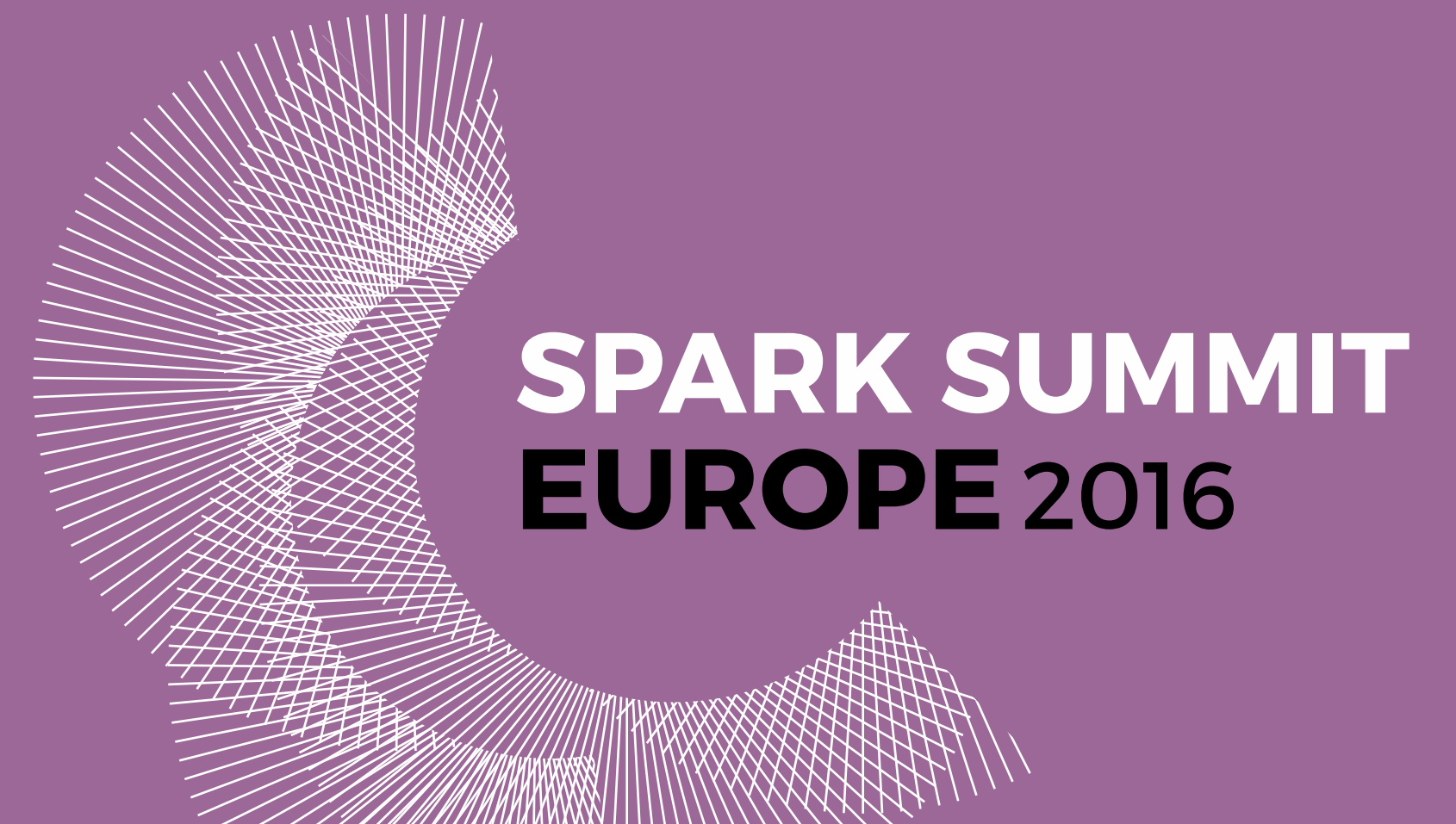


# CONTAINERIZED SPARK ON KUBERNETES

William Benton

Red Hat, Inc.

@willb • [willb@redhat.com](mailto:willb@redhat.com)

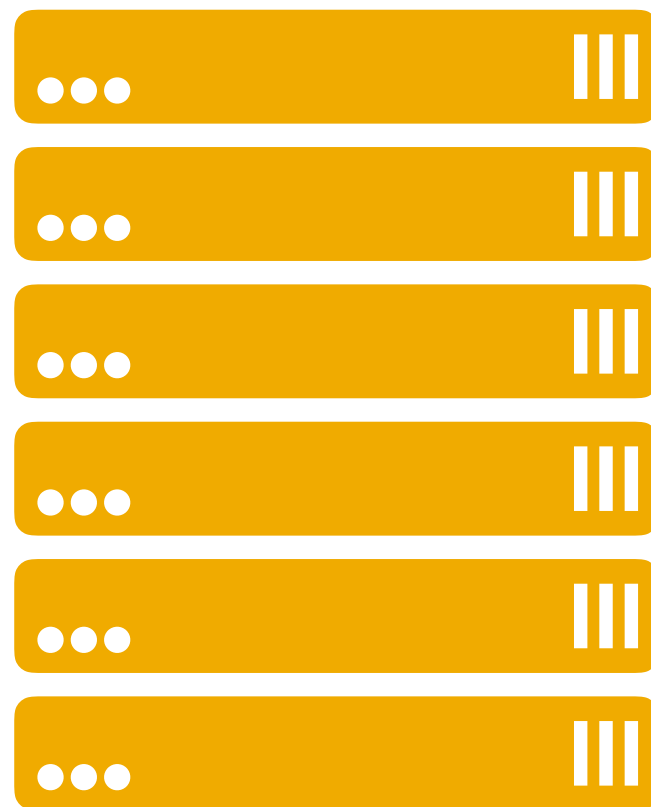


# BACKGROUND

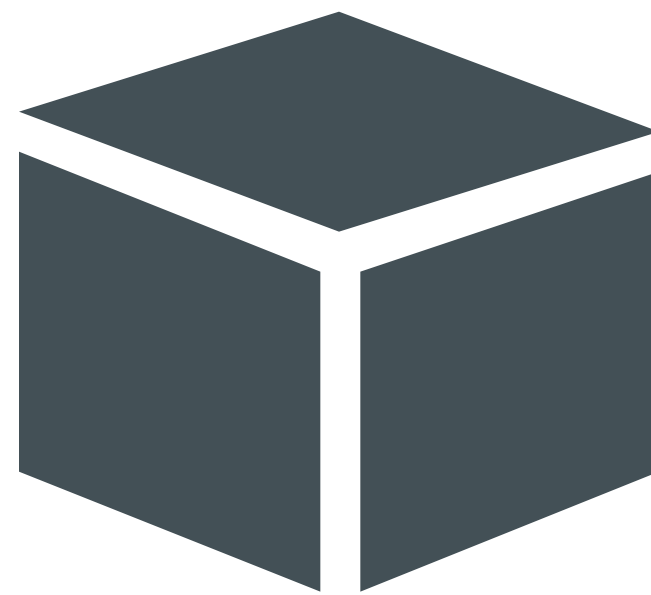
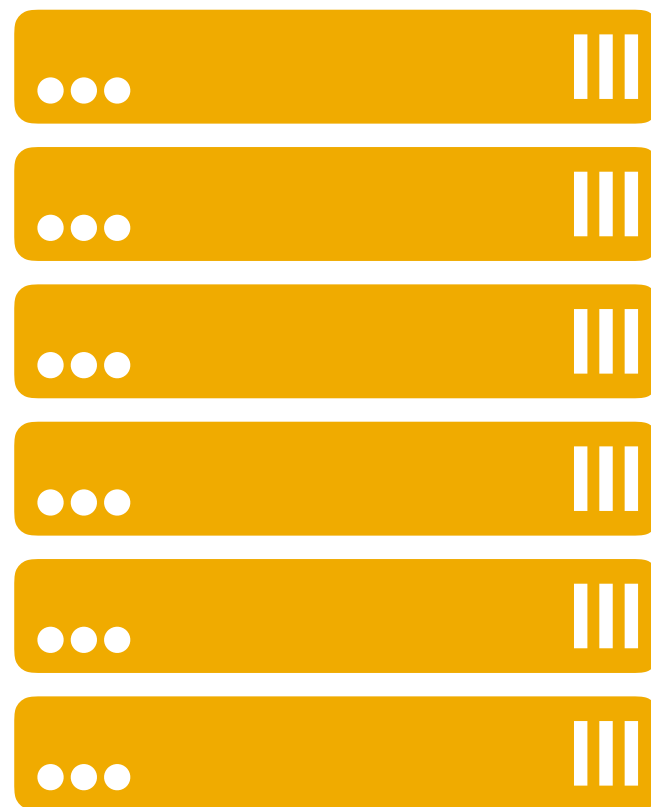
# BACKGROUND



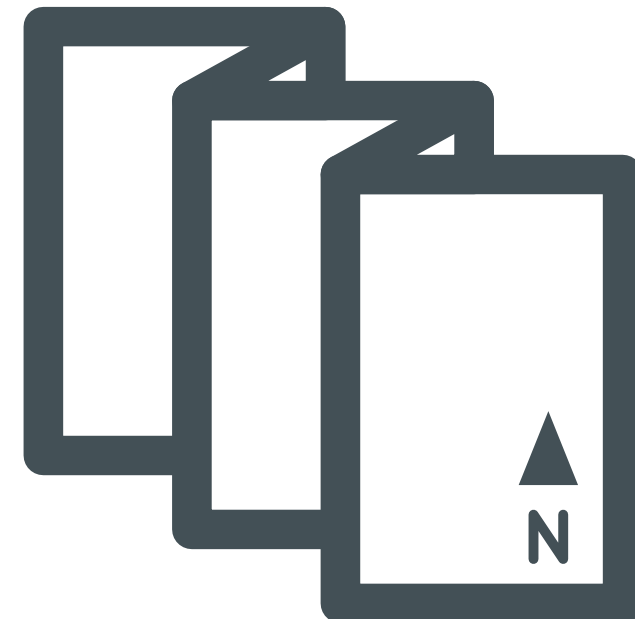
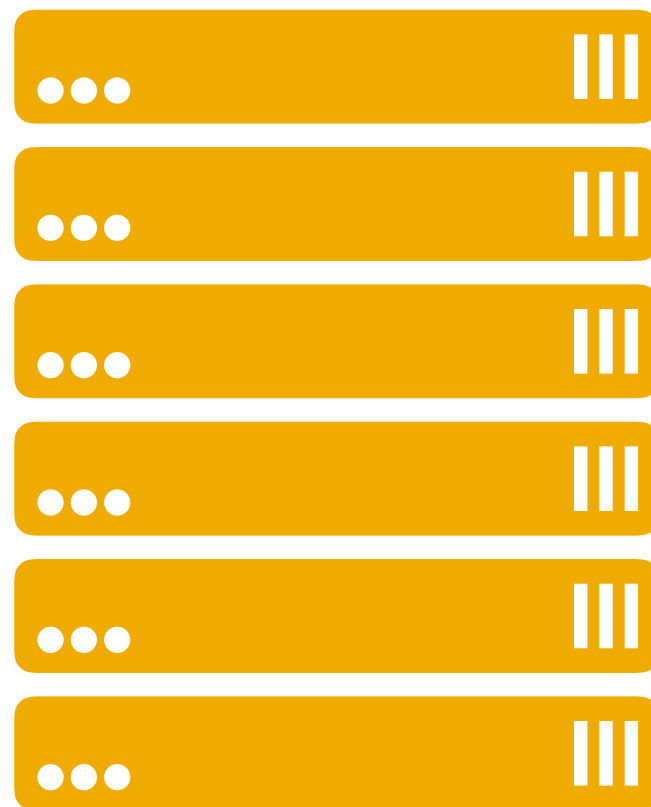
# BACKGROUND



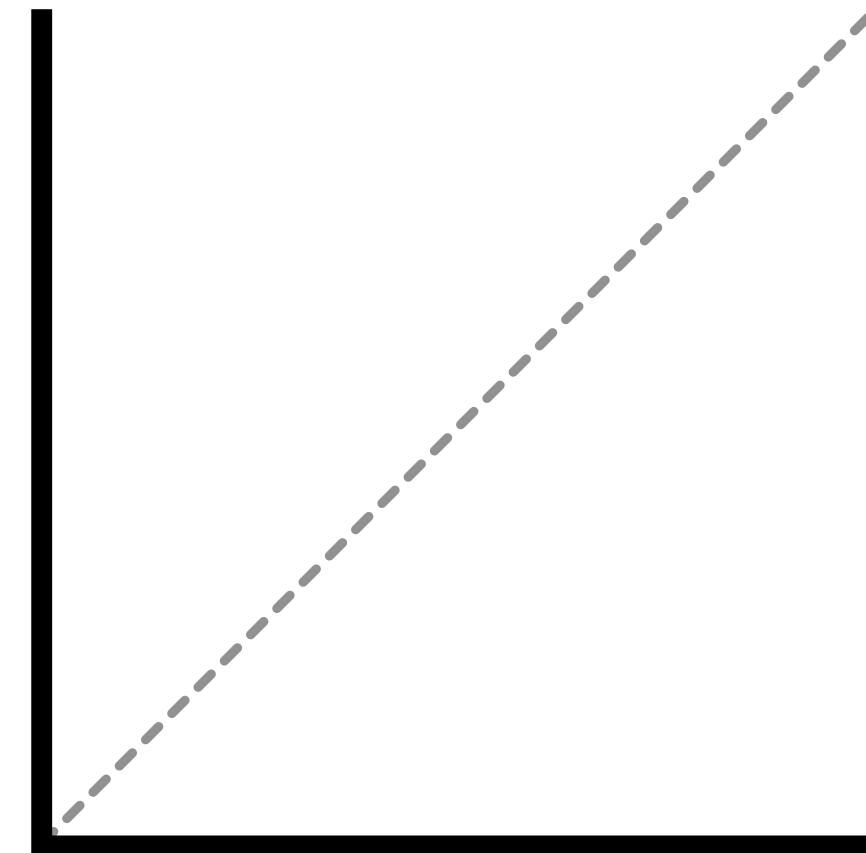
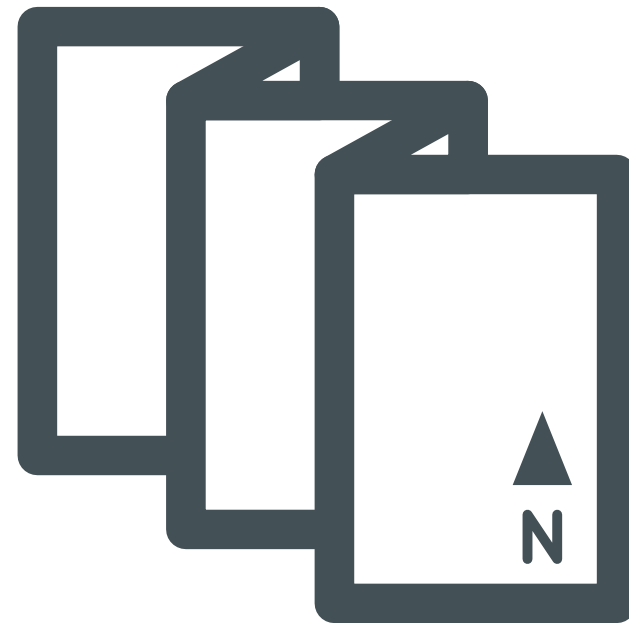
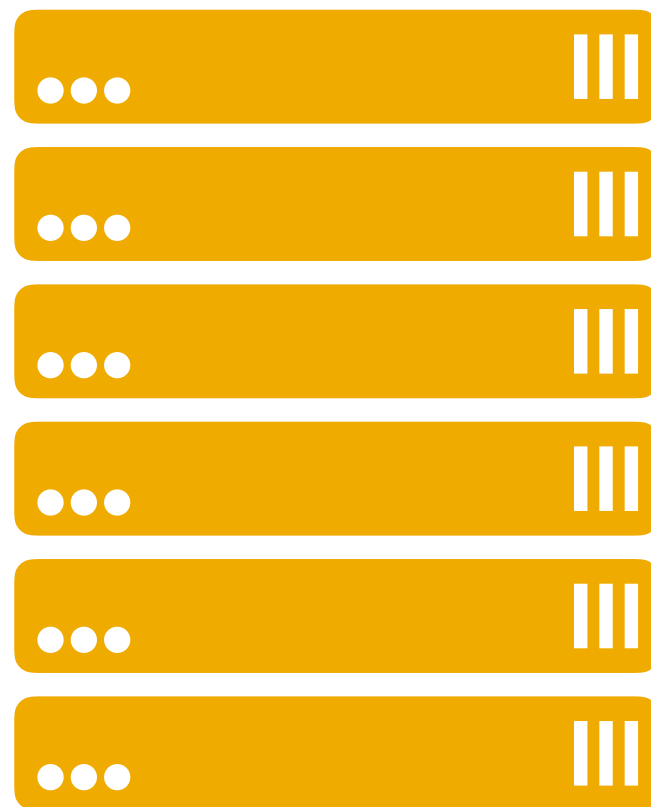
# BACKGROUND



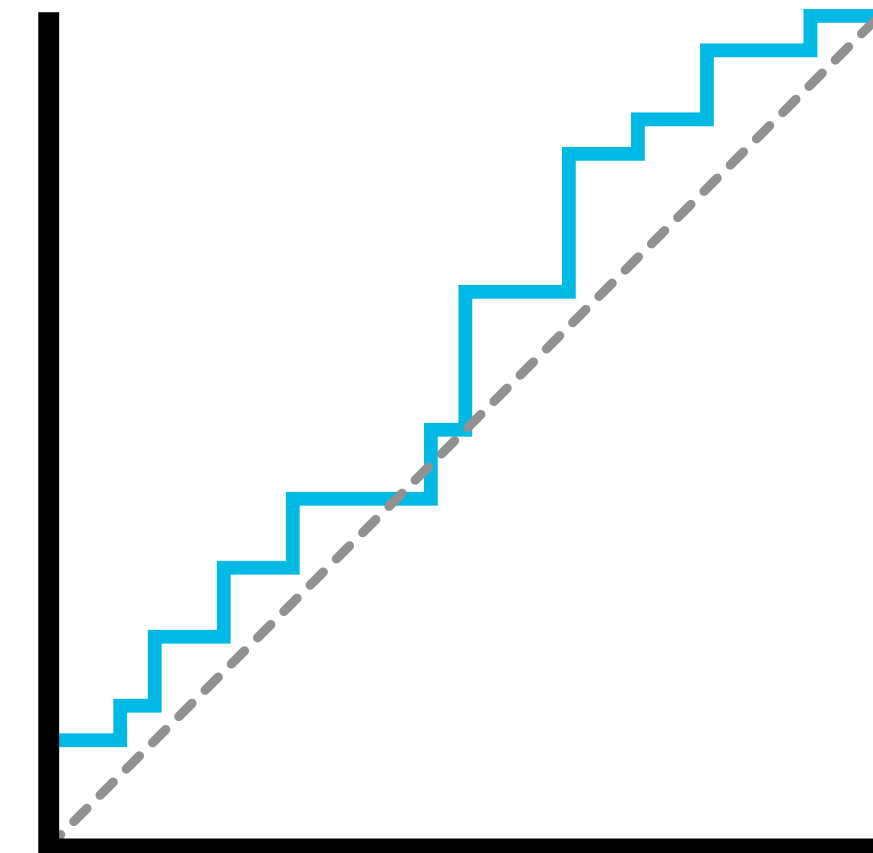
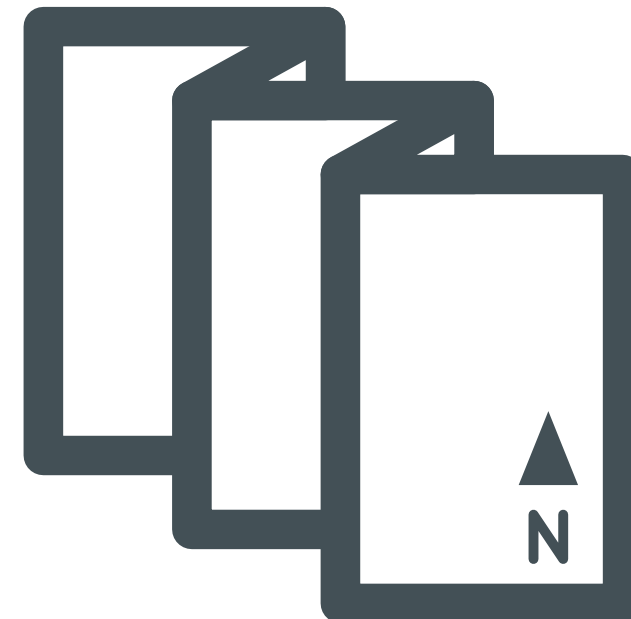
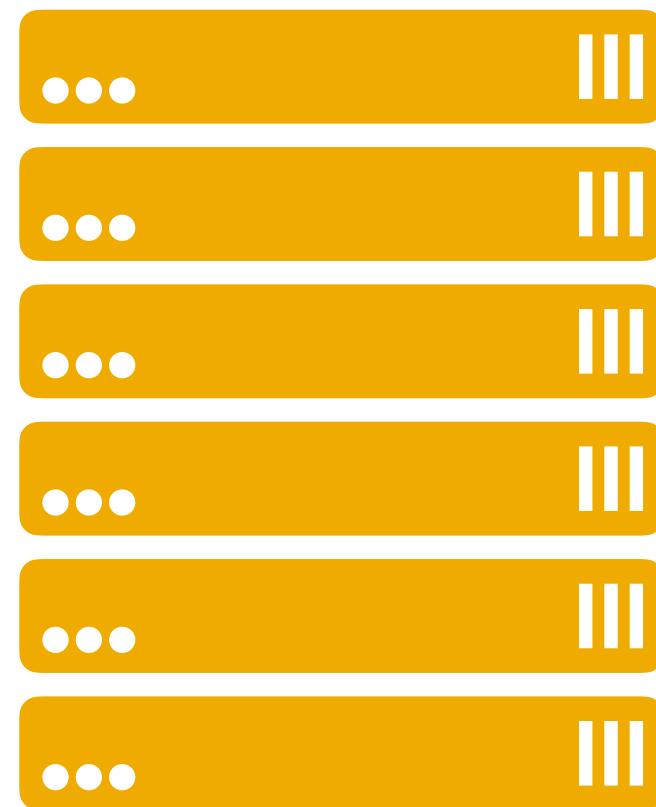
# BACKGROUND



# BACKGROUND

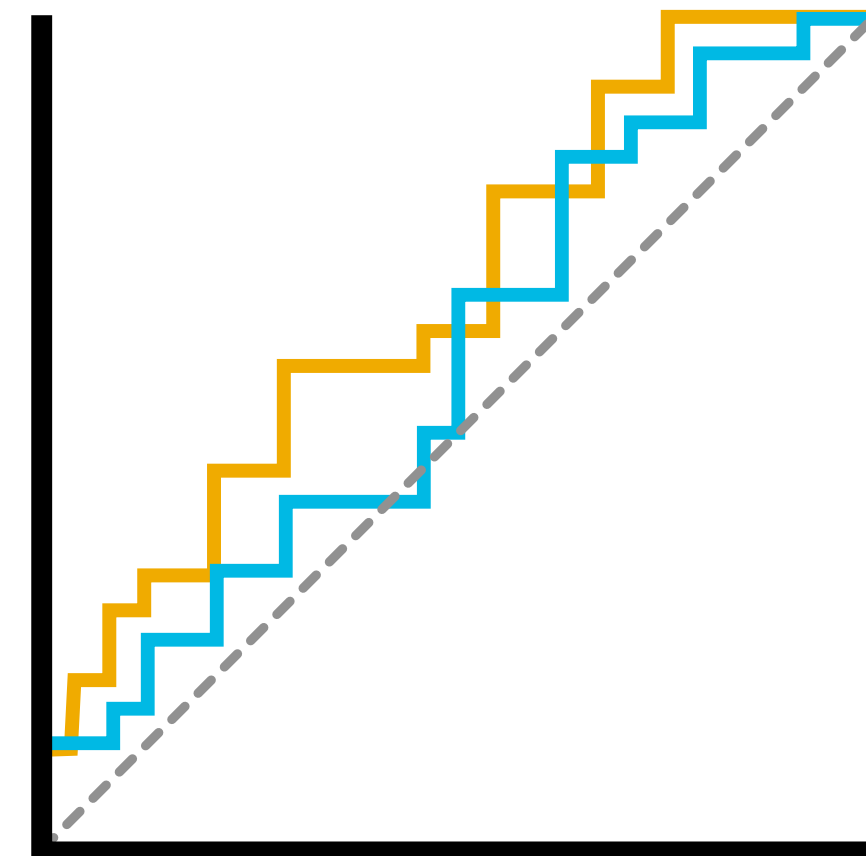
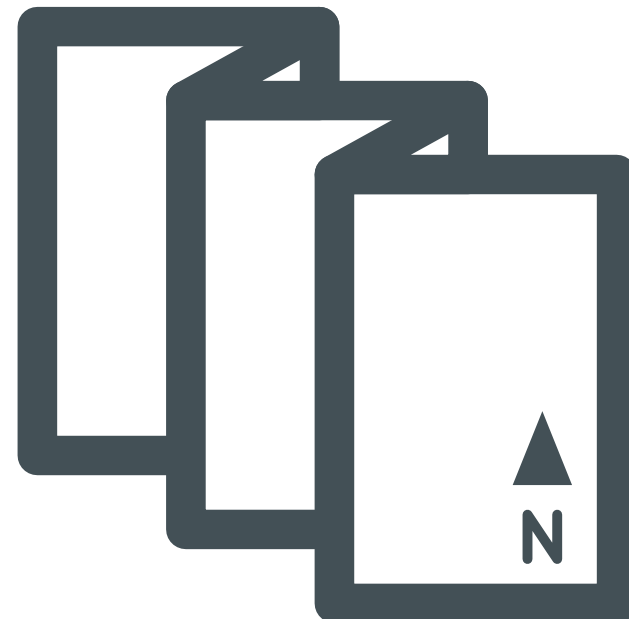
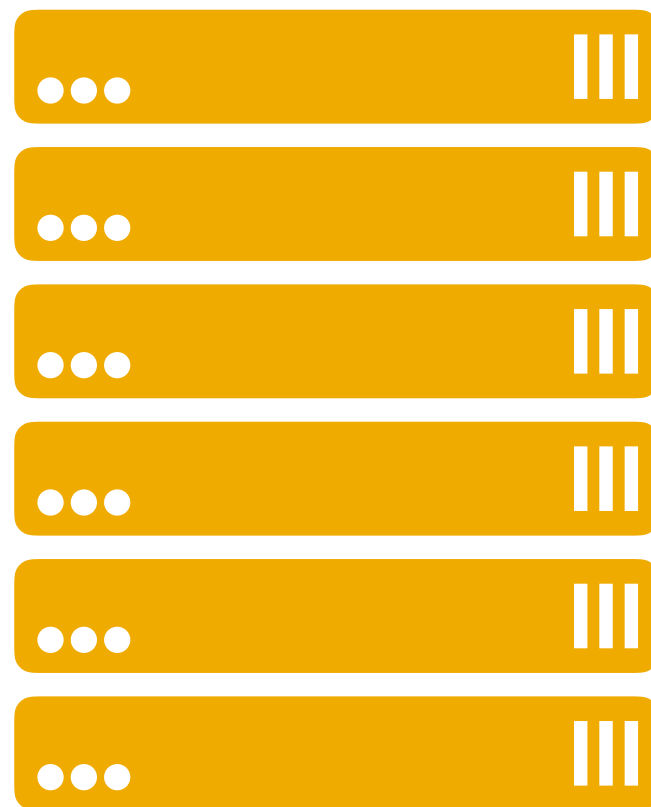


# BACKGROUND





# BACKGROUND



# WHAT OUR SPARK CLUSTER LOOKED LIKE IN 2014

# WHAT OUR SPARK CLUSTER LOOKED LIKE IN 2014

Spark executor

Spark executor

Spark executor

Spark executor

Spark executor

Spark executor

# WHAT OUR SPARK CLUSTER LOOKED LIKE IN 2014

Spark executor

Spark executor

Spark executor

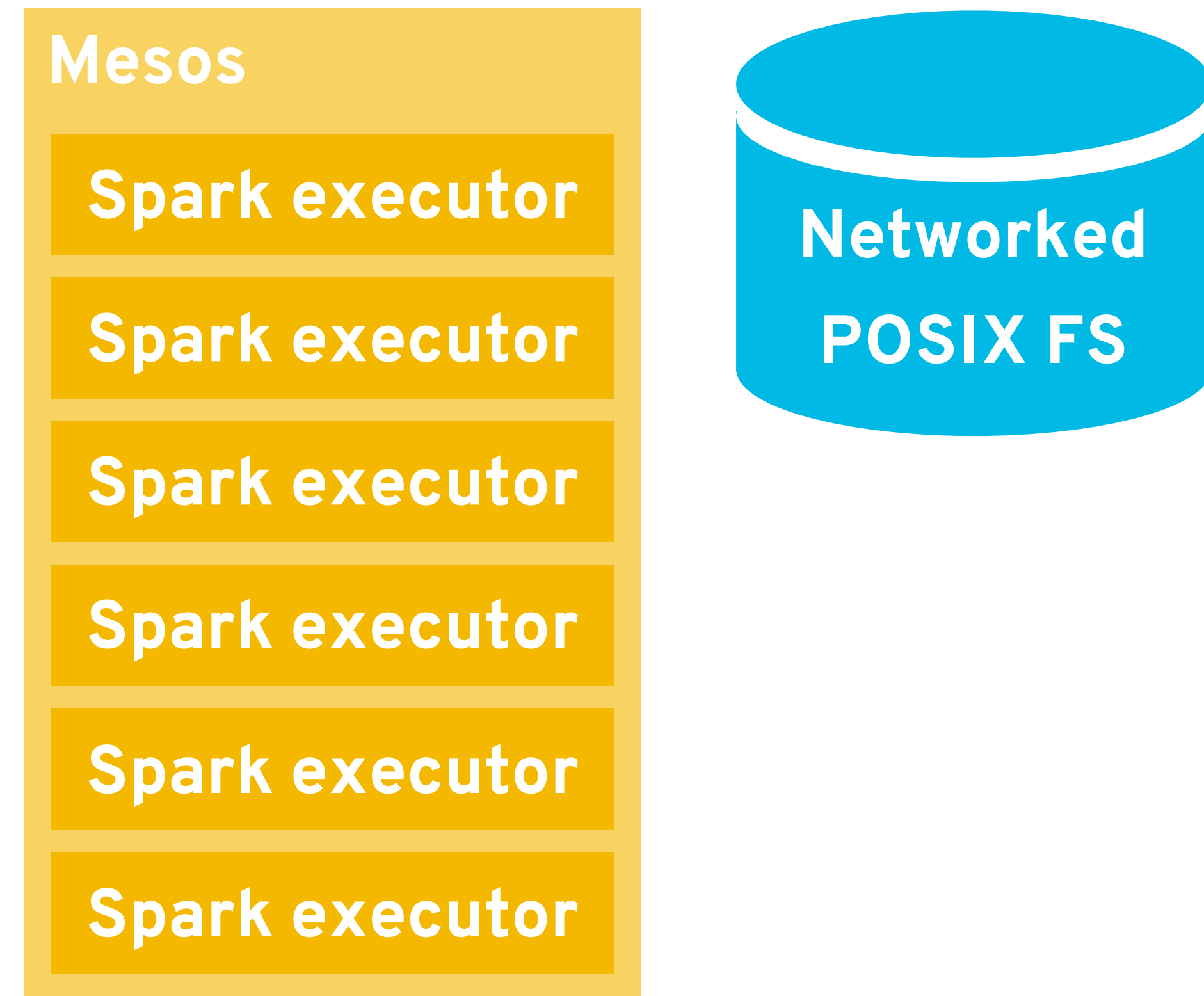
Spark executor

Spark executor

Spark executor

Networked  
POSIX FS

# WHAT OUR SPARK CLUSTER LOOKED LIKE IN 2014



# WHAT OUR SPARK CLUSTER LOOKED LIKE IN 2014



Mesos

Spark executor

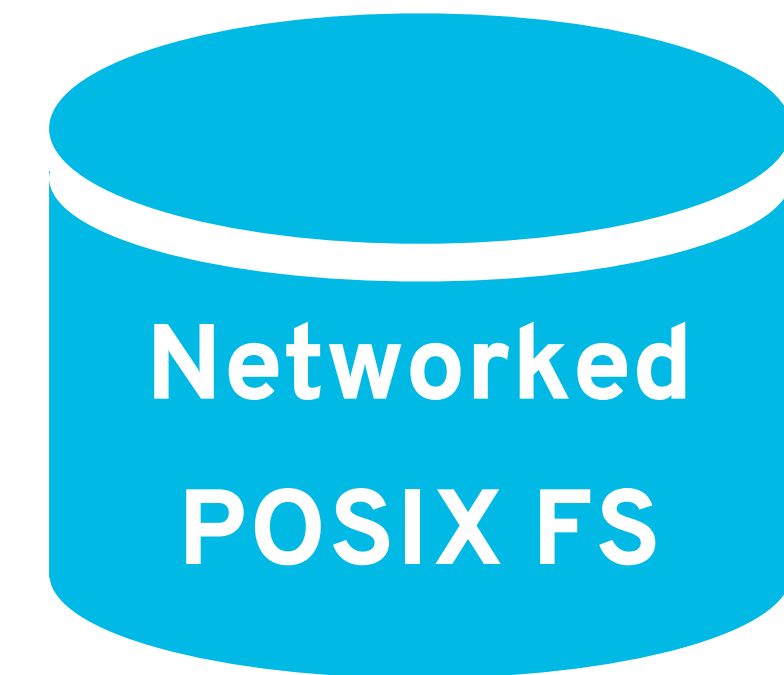
Spark executor

Spark executor

Spark executor

Spark executor

Spark executor



# WHAT OUR SPARK CLUSTER LOOKED LIKE IN 2014



SPARK SUMMIT  
EUROPE 2016

Mesos



Spark executor



Spark executor



Spark executor



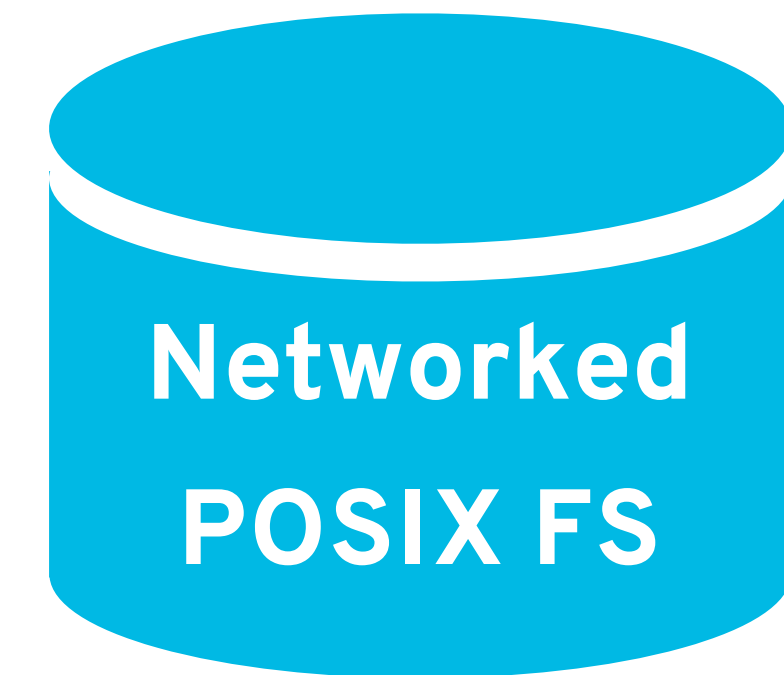
Spark executor



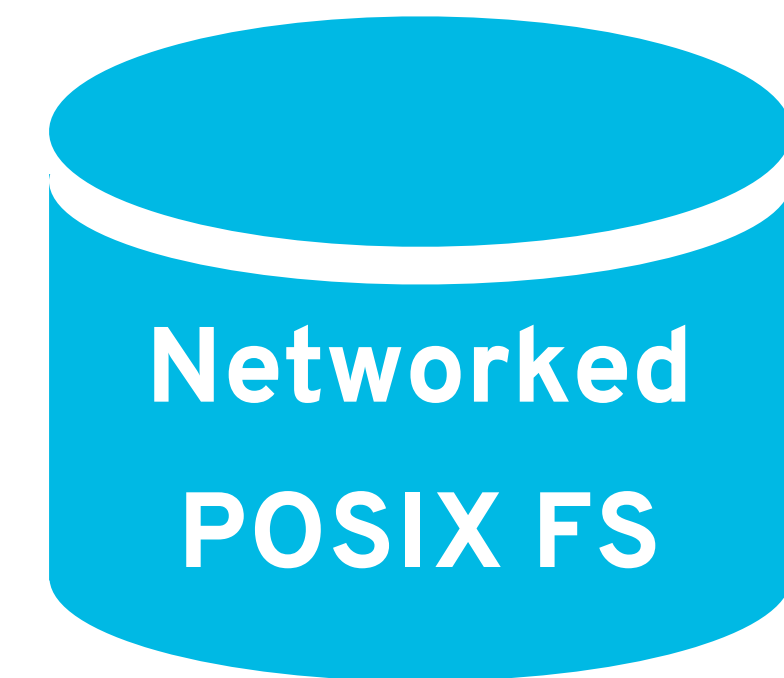
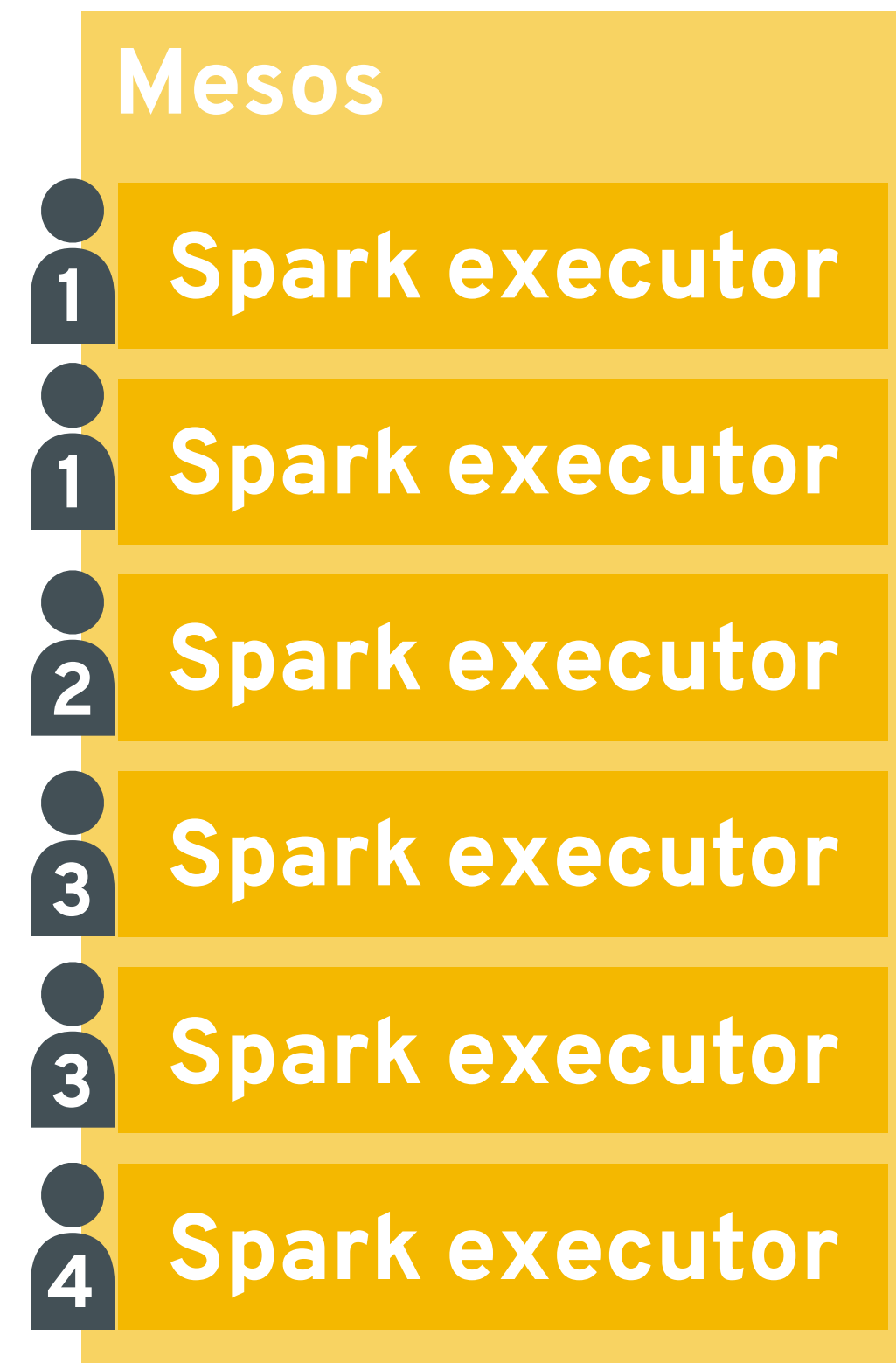
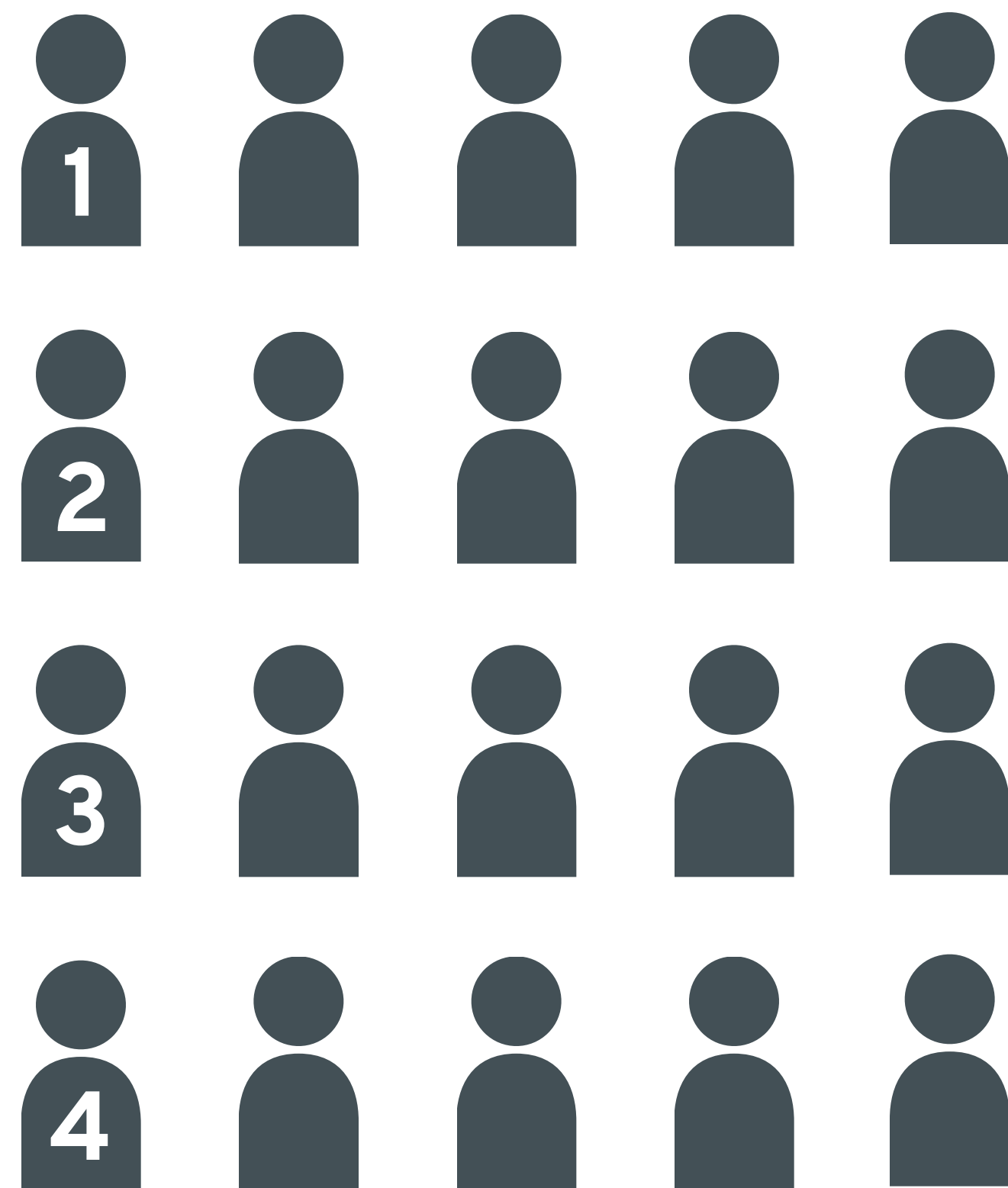
Spark executor



Spark executor



# WHAT OUR SPARK CLUSTER LOOKED LIKE IN 2014





**Analytics is no longer a  
separate workload.**

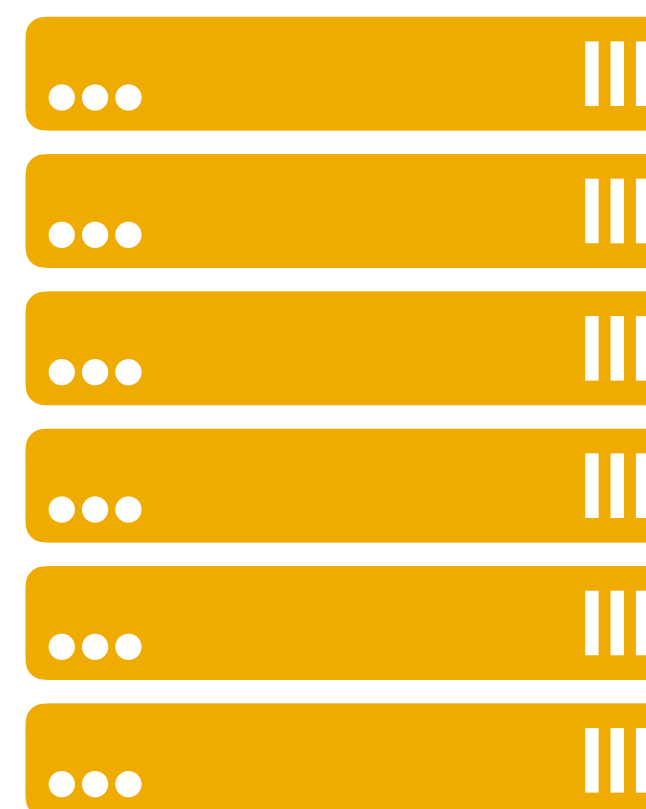
**Analytics is an essential  
component of modern data-  
driven applications.**

# OUR GOALS

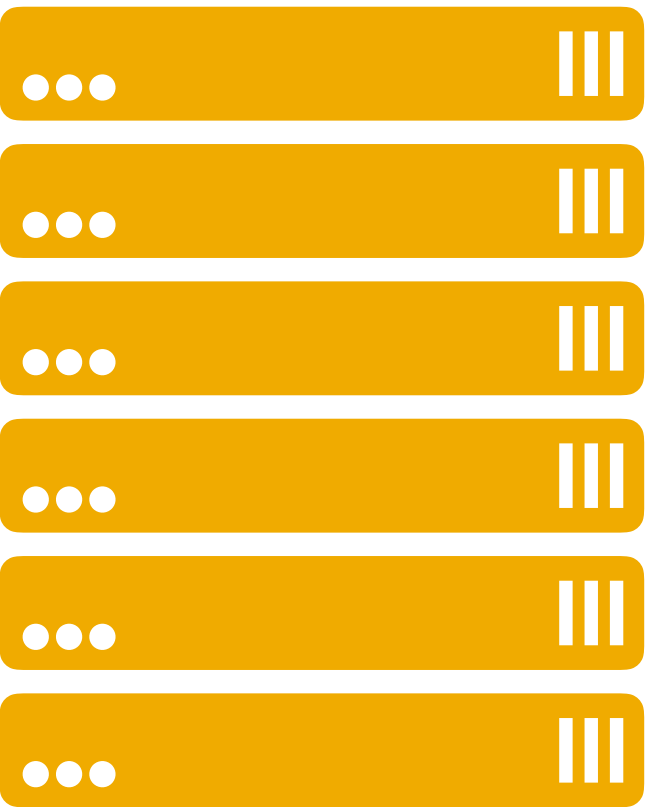
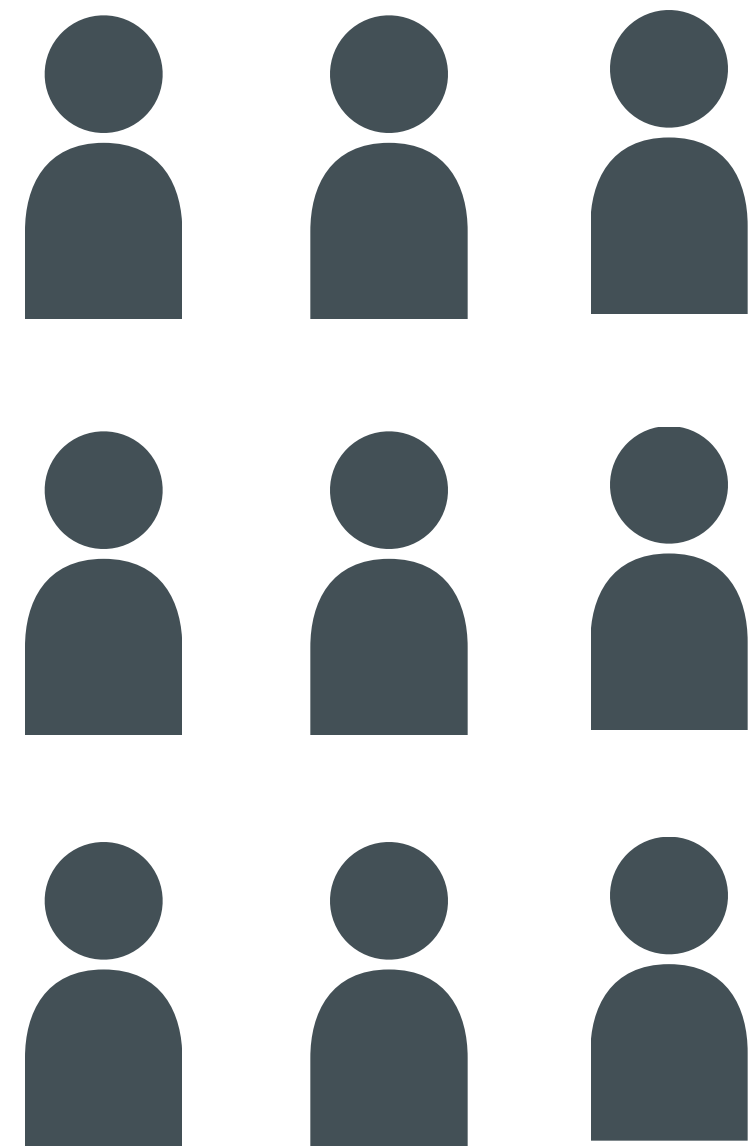
# OUR GOALS



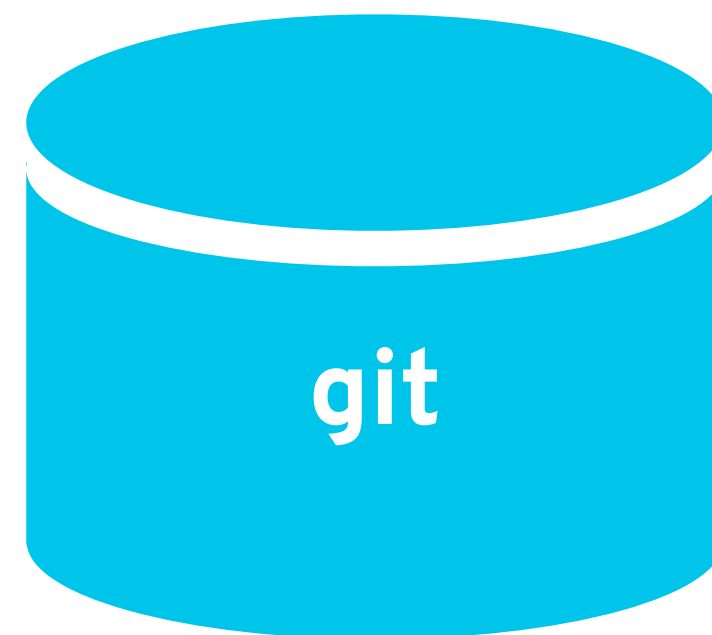
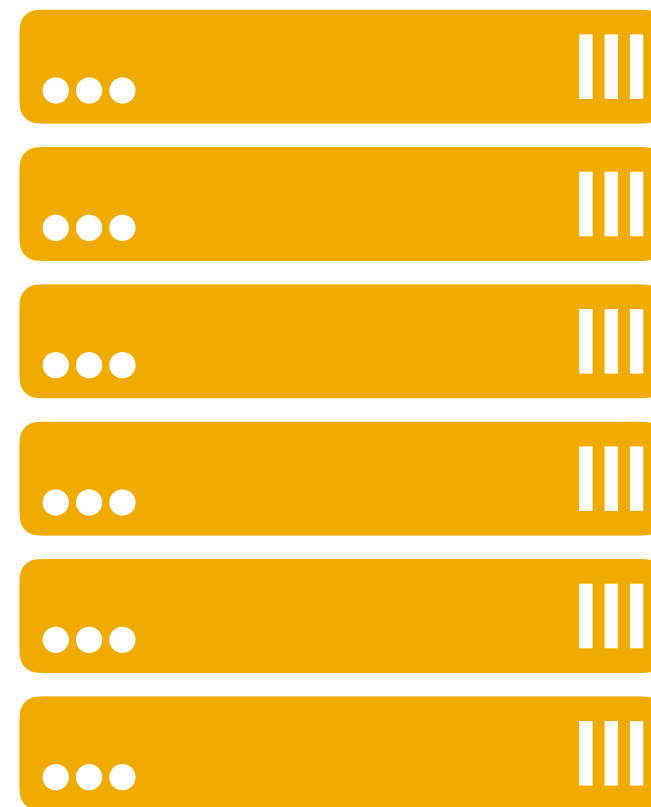
# OUR GOALS



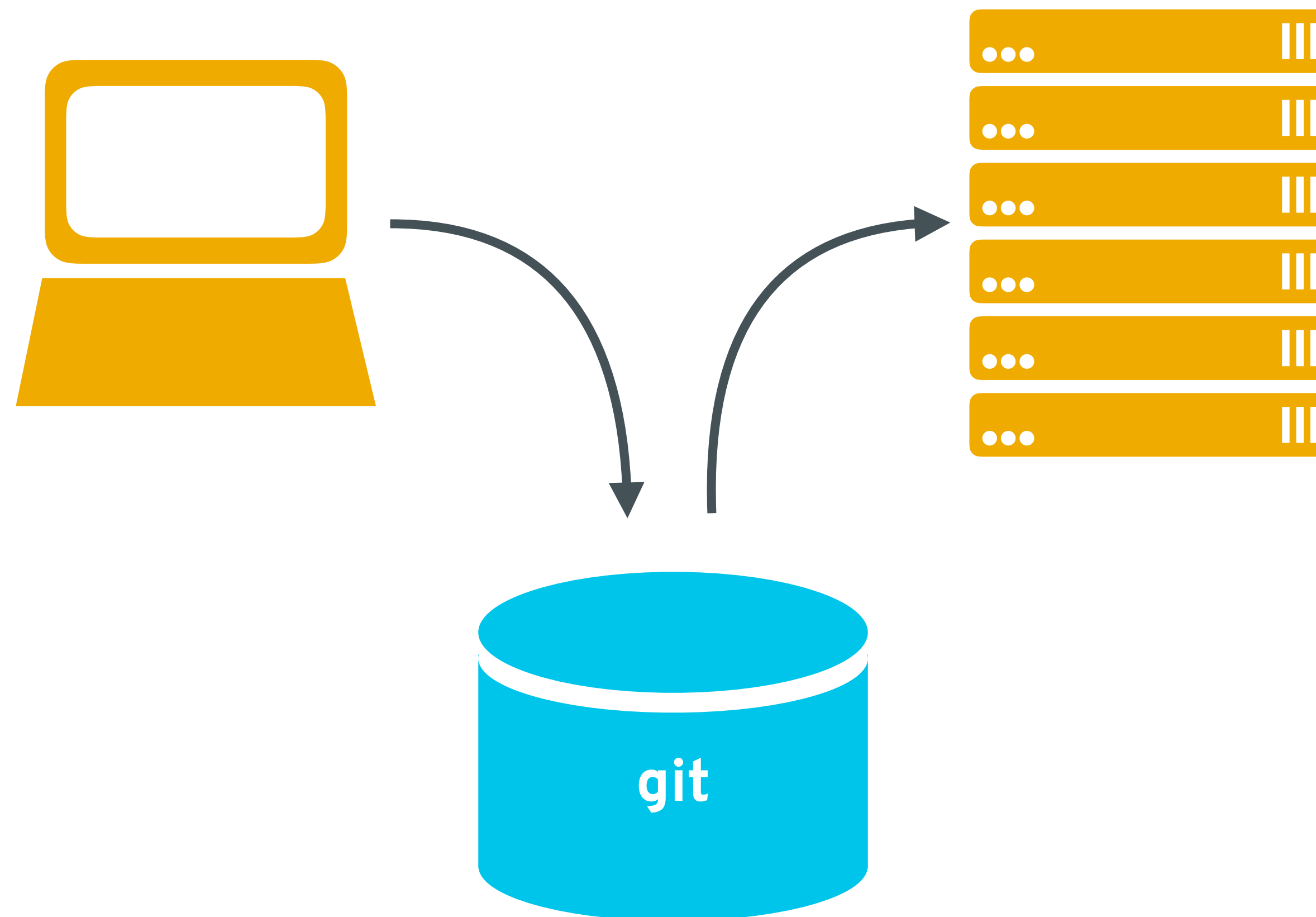
# OUR GOALS



# OUR GOALS

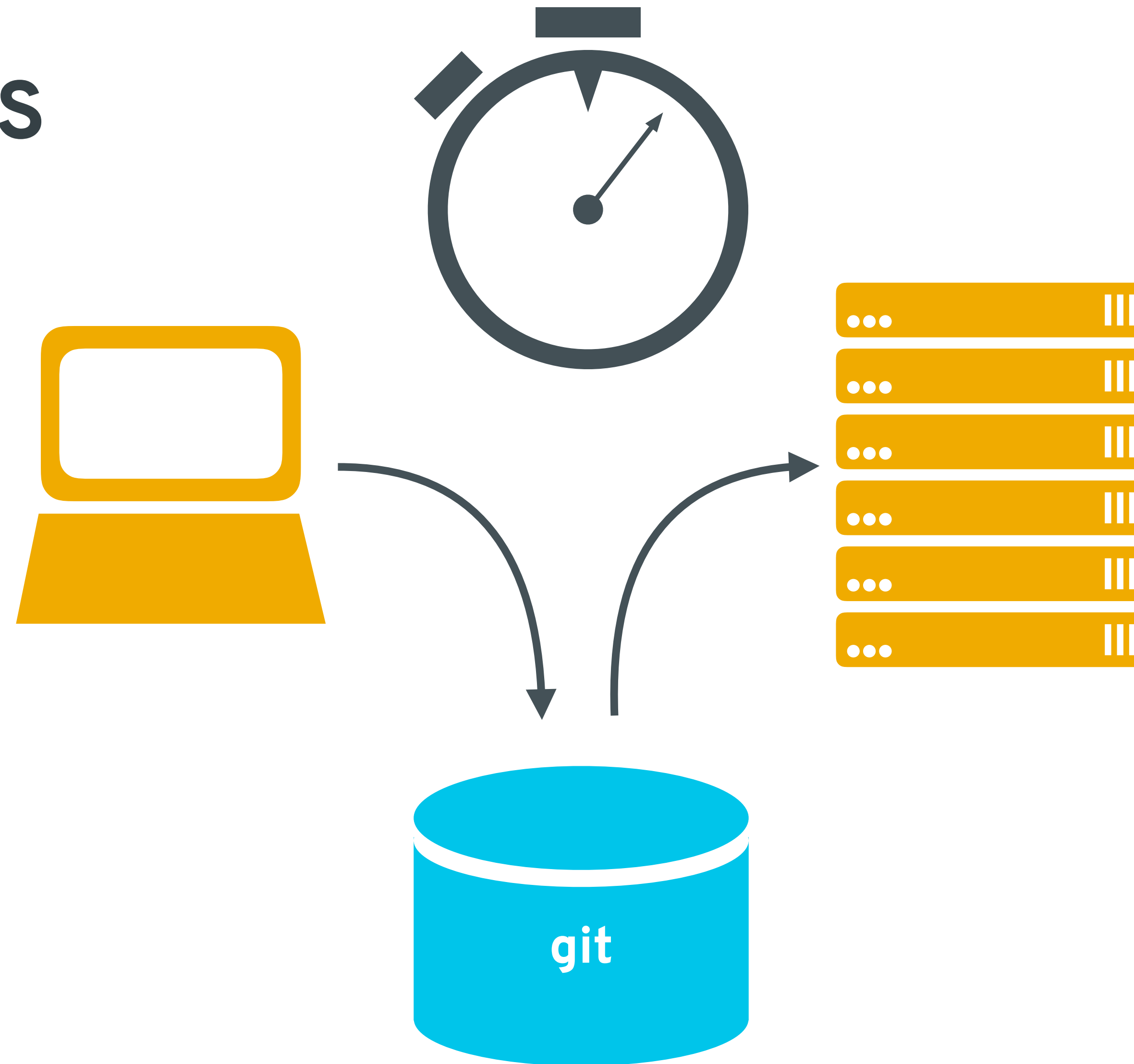


# OUR GOALS





# OUR GOALS



# FORECAST

Motivating containerized microservices

Architectures for analytics and applications

Spark clusters in containers: practicalities and pitfalls

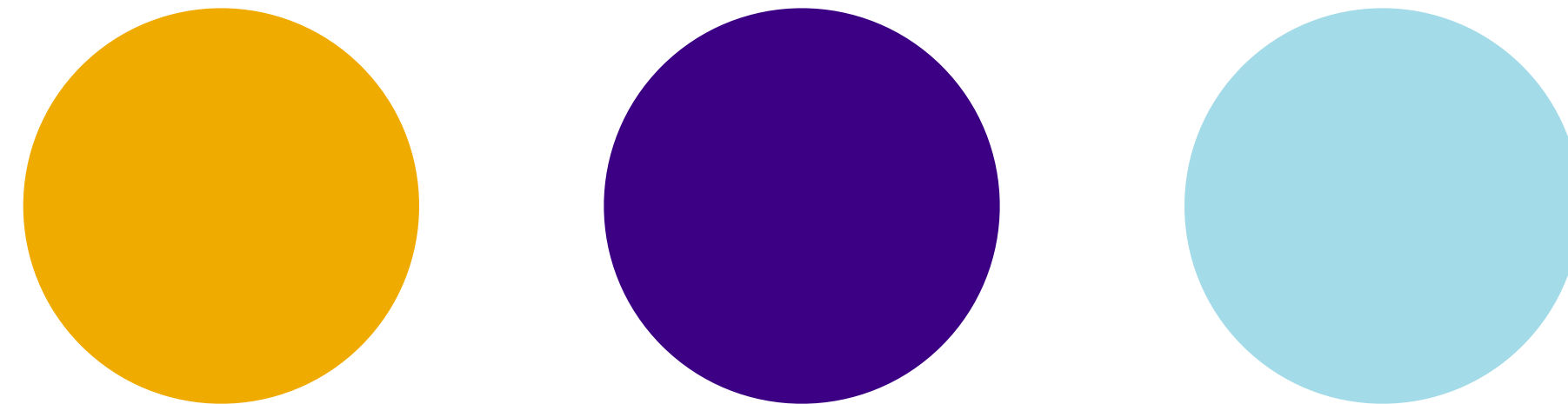
Play along at home

Future work

# MOTIVATING MICROSERVICES

**A microservice architecture  
employs lightweight, modular,  
and typically stateless  
components with well-defined  
interfaces and contracts.**

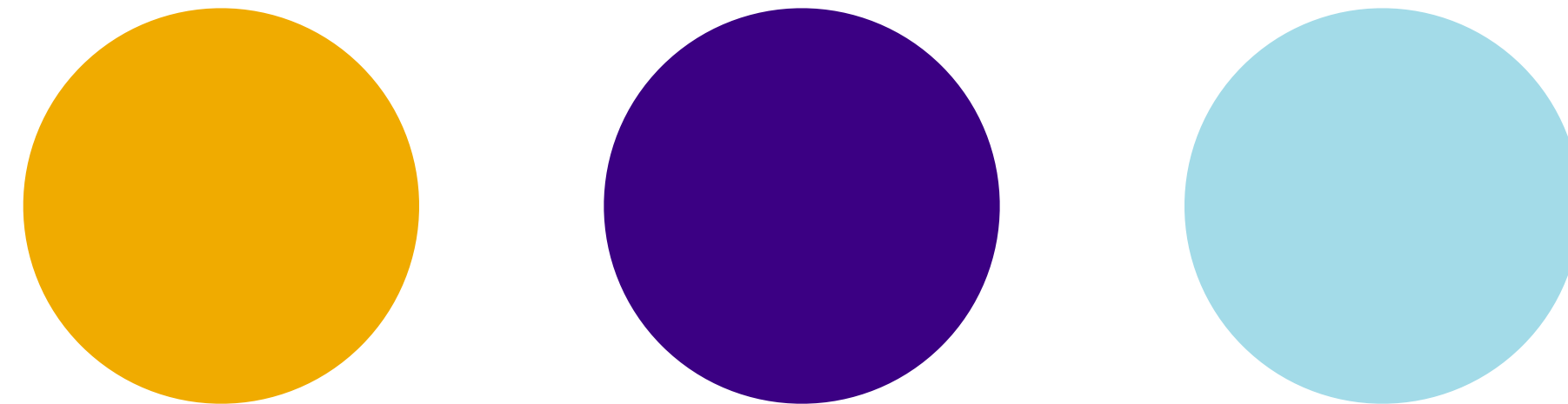
# BENEFITS OF MICROSERVICE ARCHITECTURES



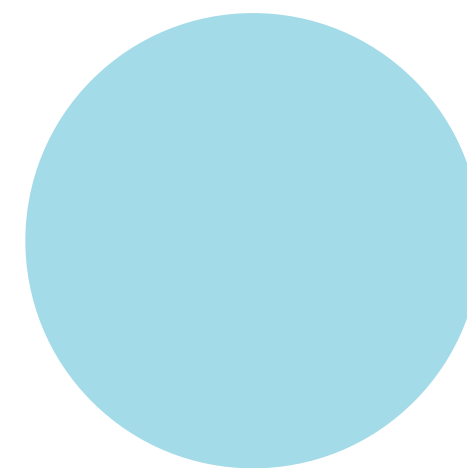
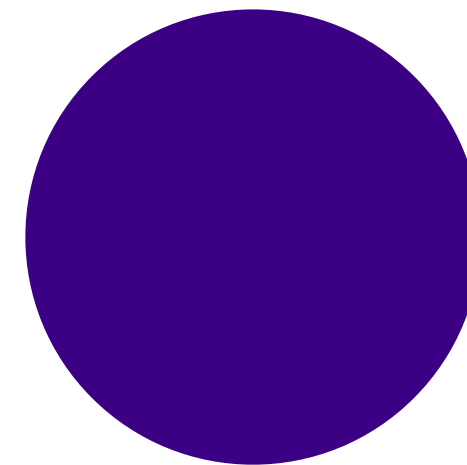
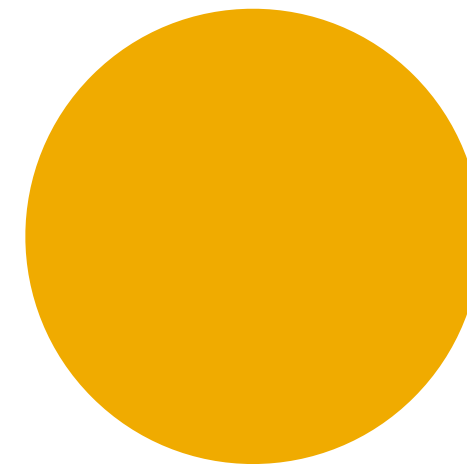
# BENEFITS OF MICROSERVICE ARCHITECTURES



# BENEFITS OF MICROSERVICE ARCHITECTURES

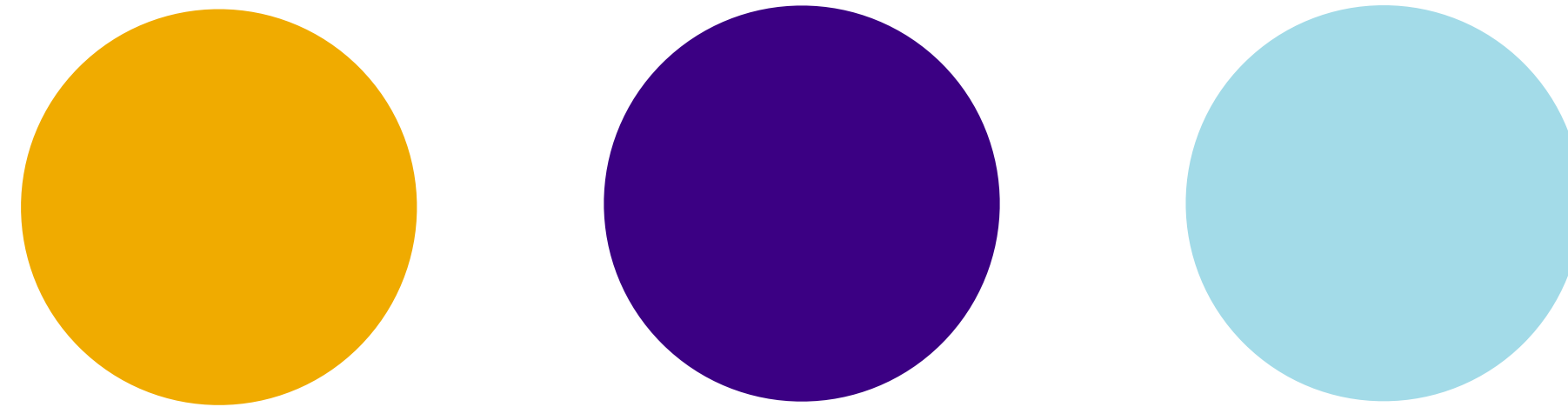


# BENEFITS OF MICROSERVICE ARCHITECTURES

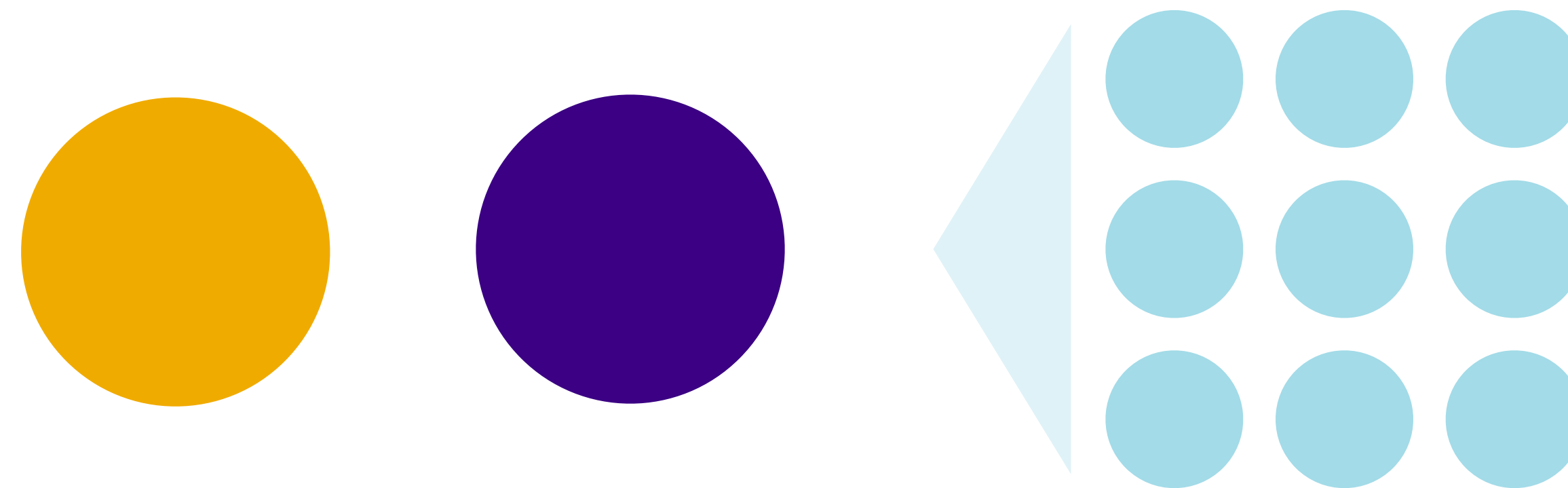




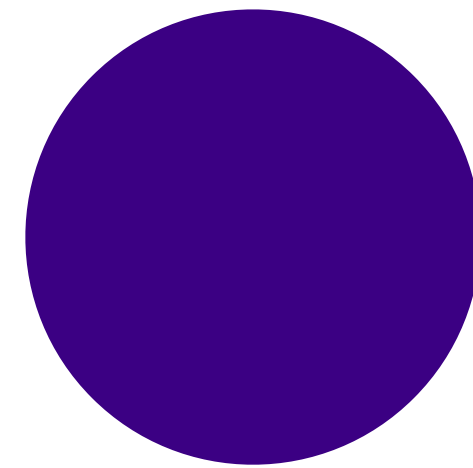
# BENEFITS OF MICROSERVICE ARCHITECTURES



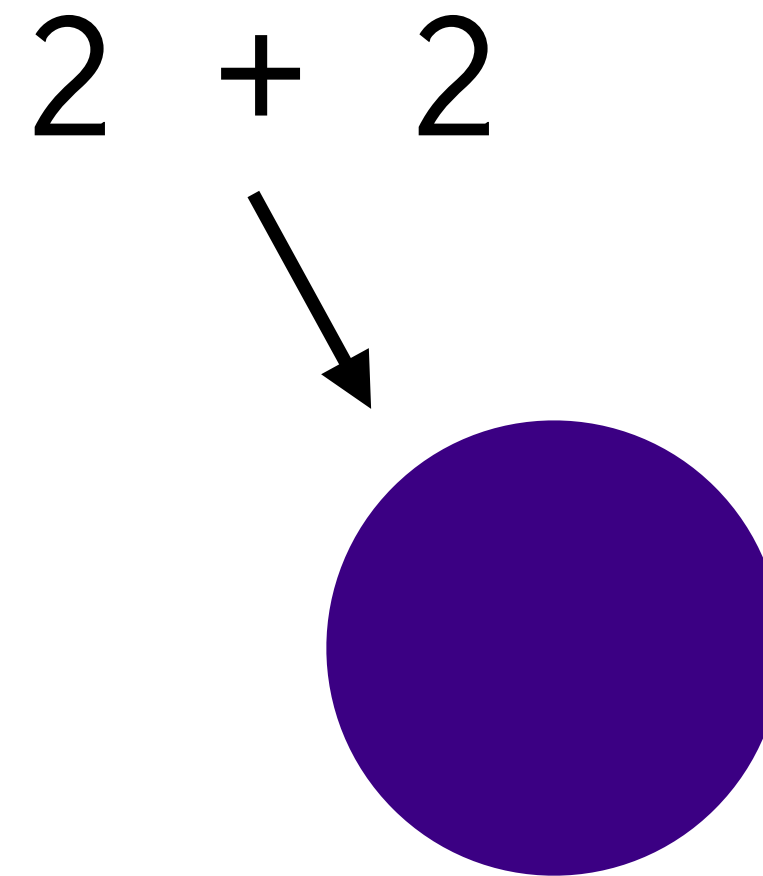
# BENEFITS OF MICROSERVICE ARCHITECTURES



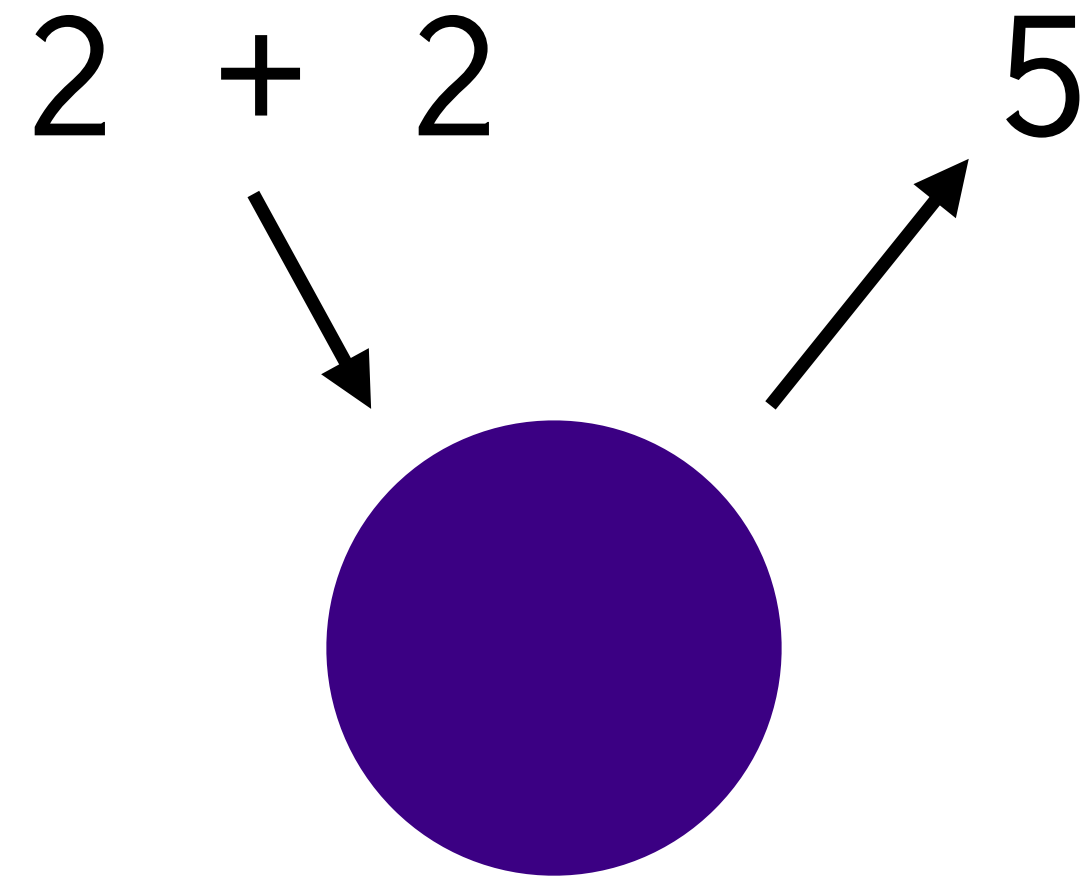
# BENEFITS OF MICROSERVICE ARCHITECTURES



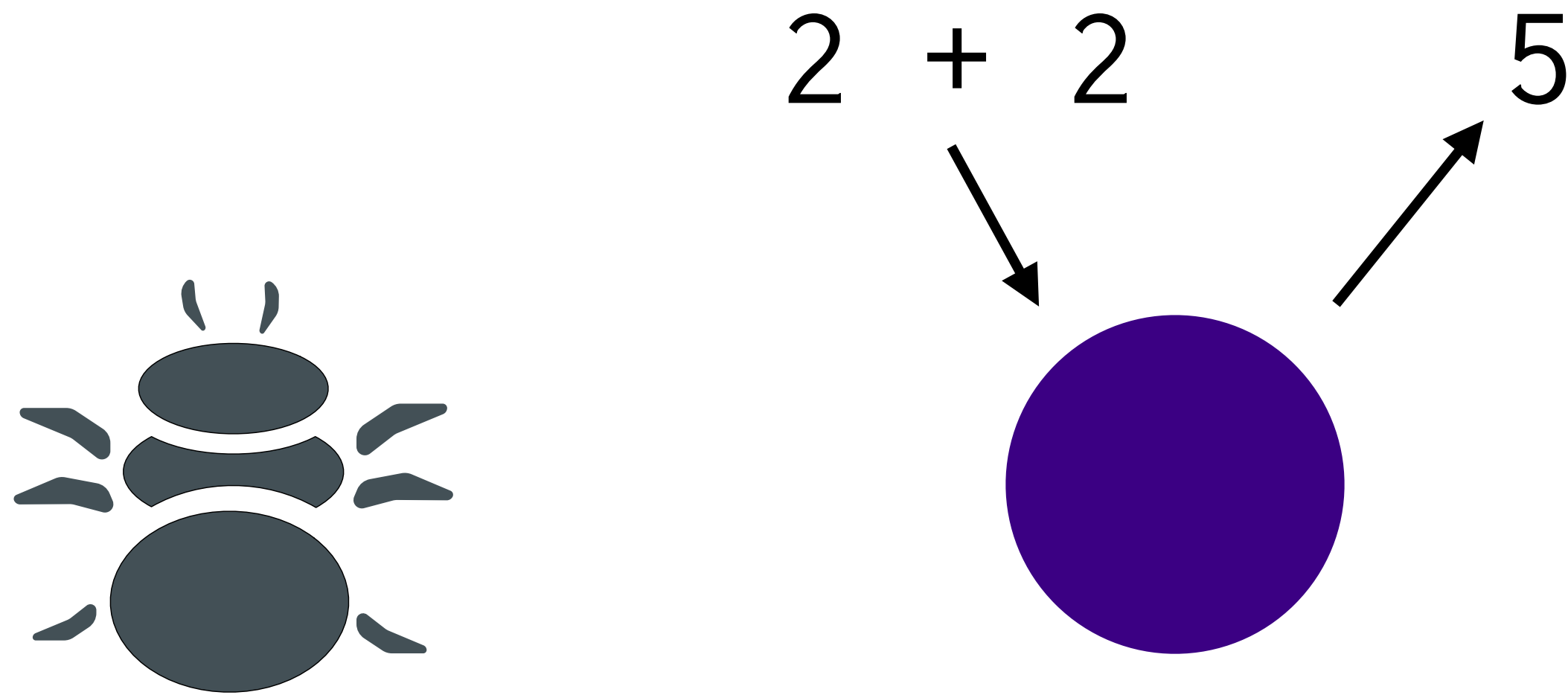
# BENEFITS OF MICROSERVICE ARCHITECTURES



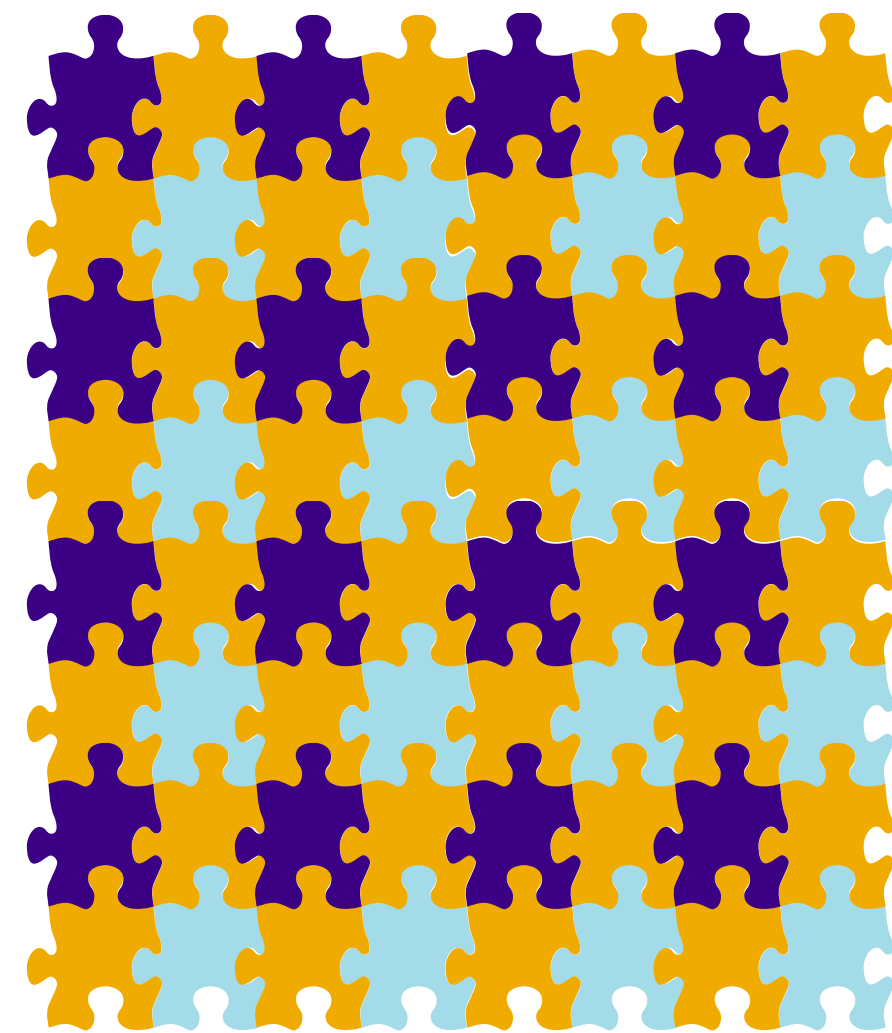
# BENEFITS OF MICROSERVICE ARCHITECTURES



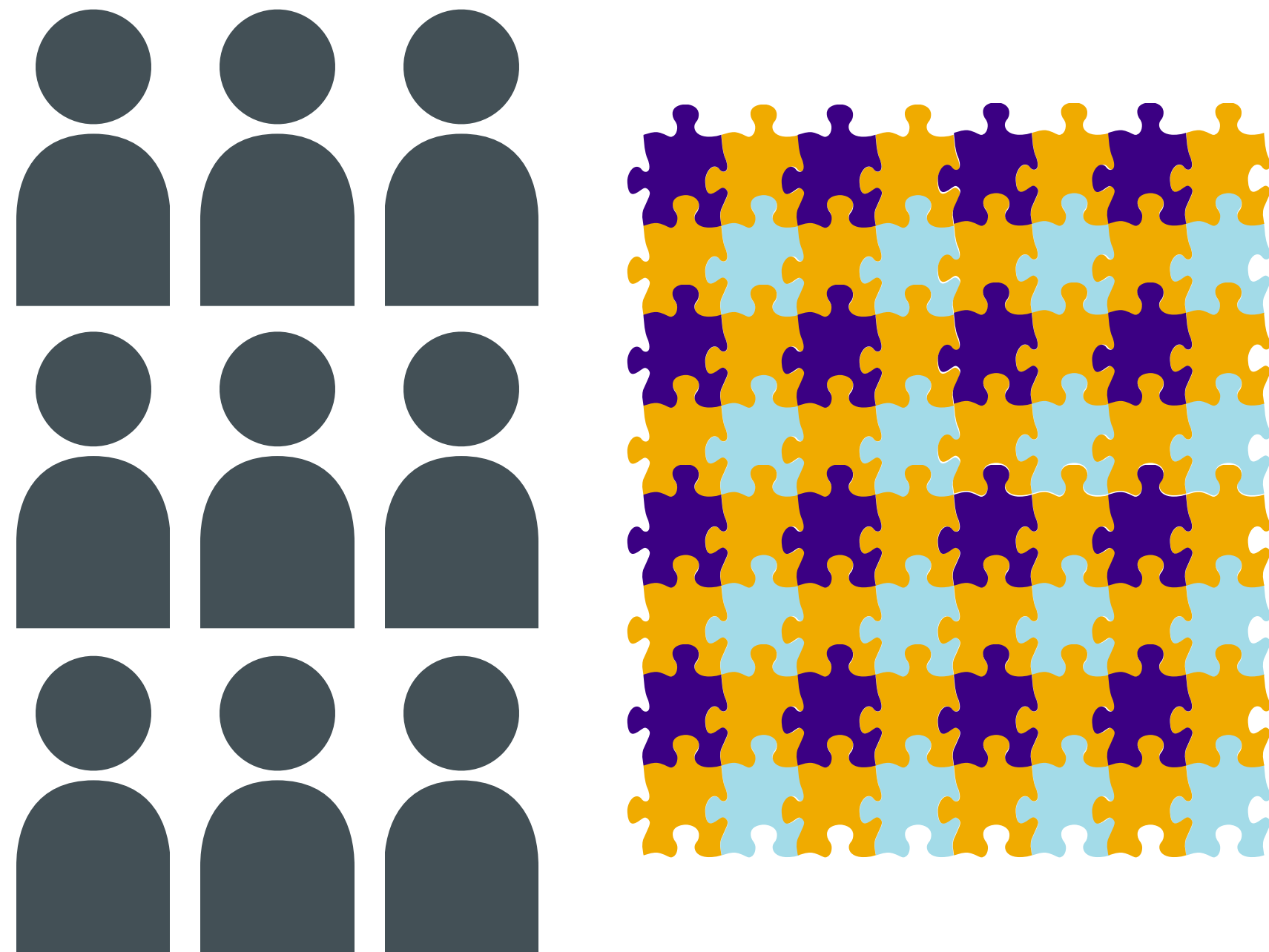
# BENEFITS OF MICROSERVICE ARCHITECTURES



# BENEFITS OF MICROSERVICE ARCHITECTURES

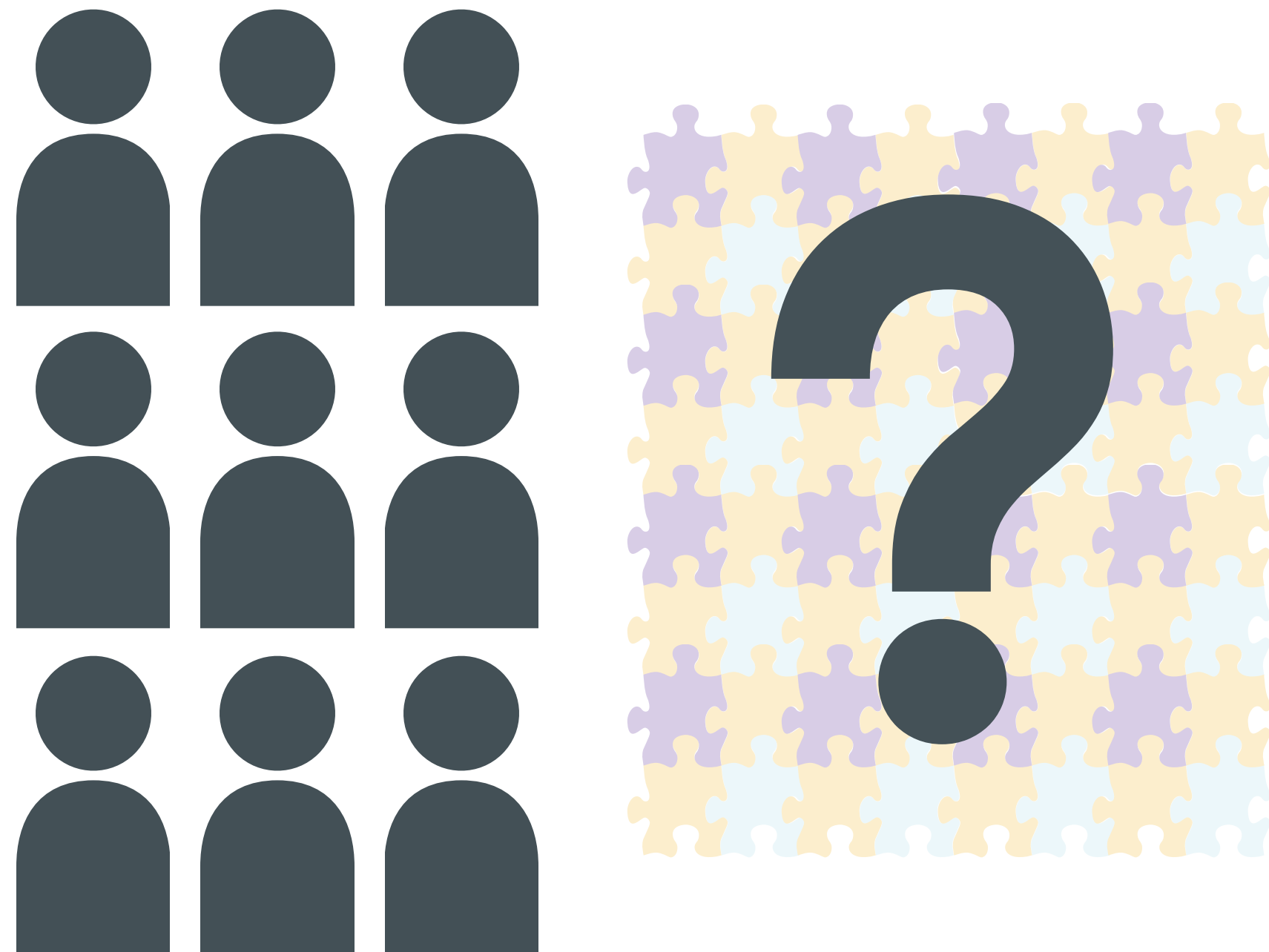


# BENEFITS OF MICROSERVICE ARCHITECTURES

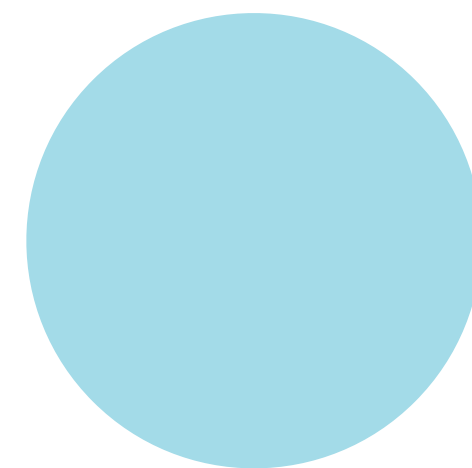
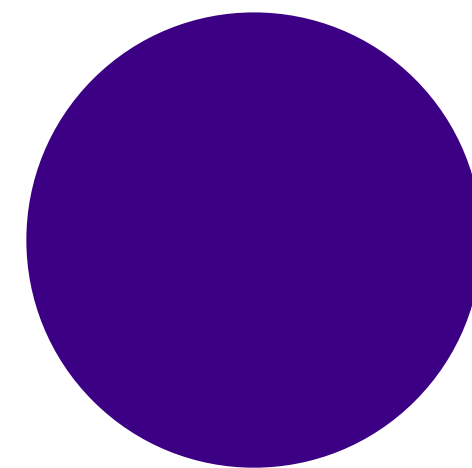
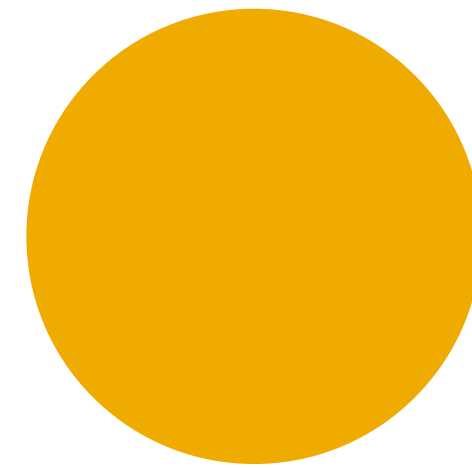
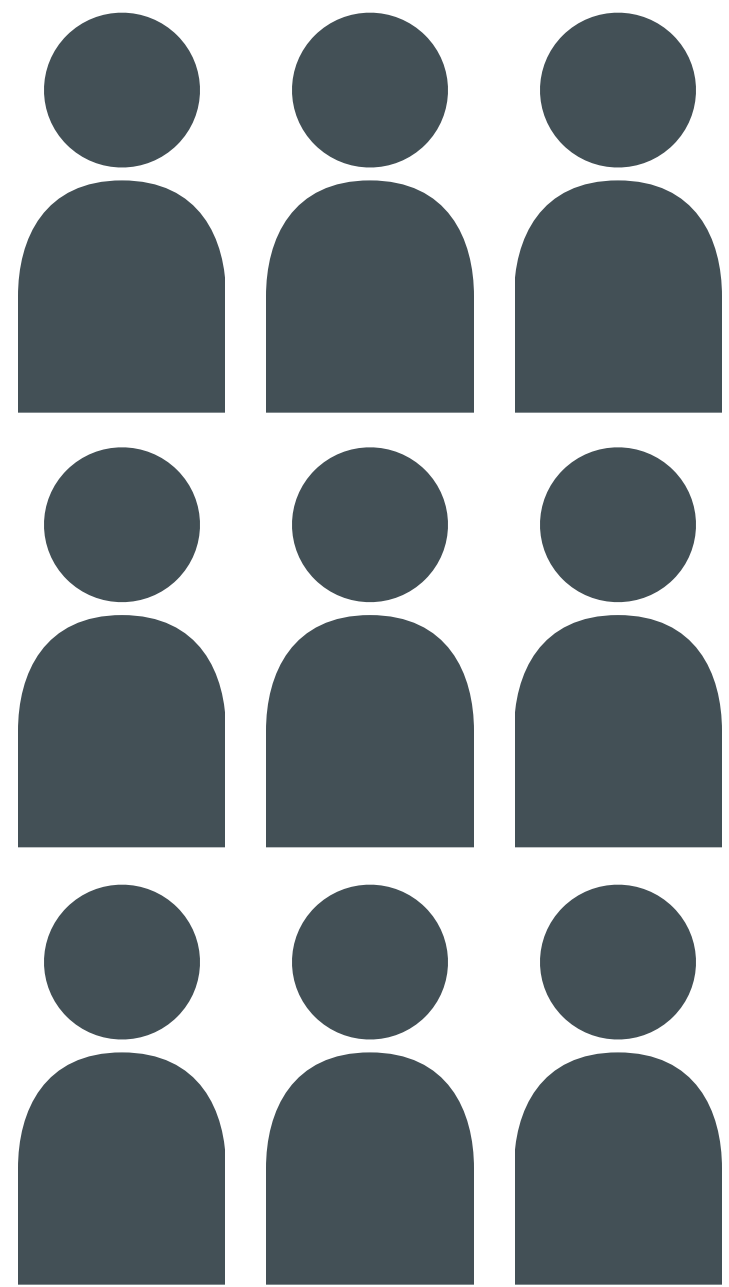




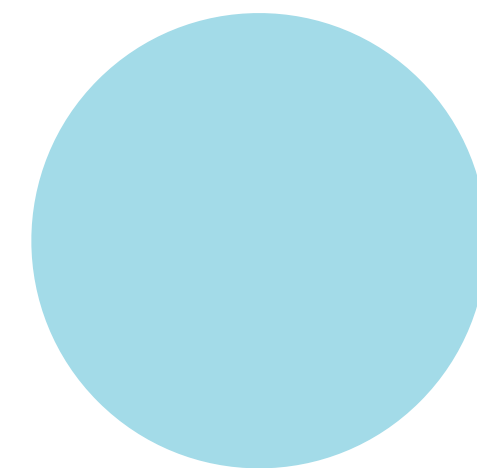
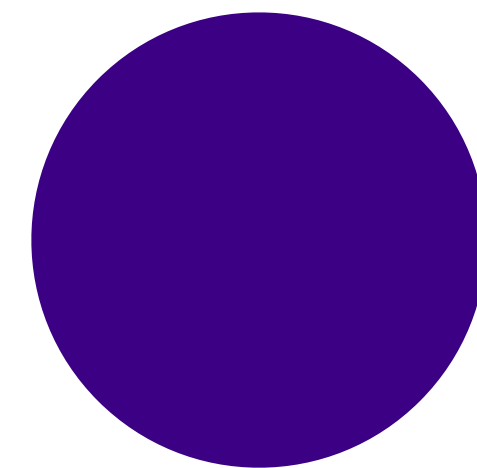
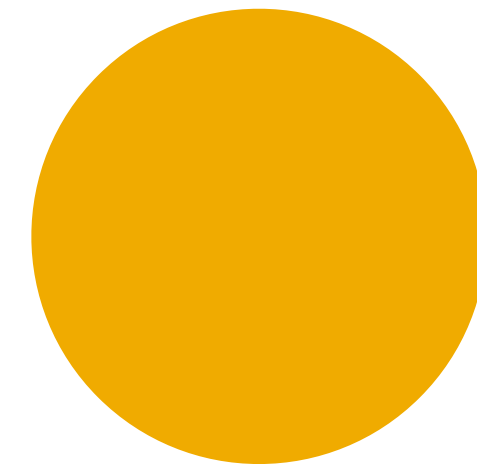
# BENEFITS OF MICROSERVICE ARCHITECTURES



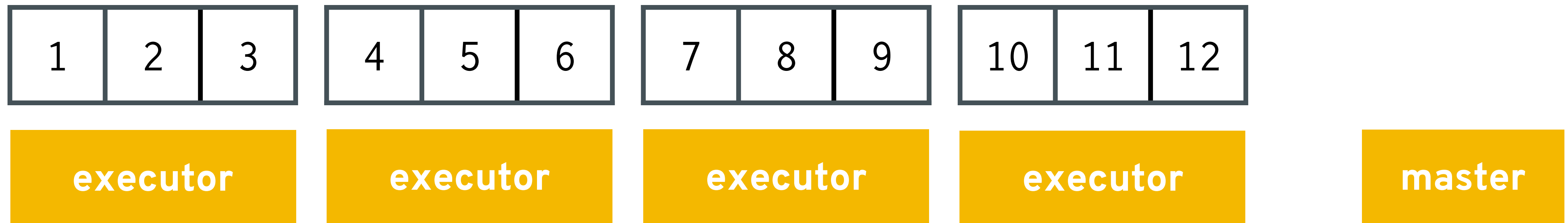
# BENEFITS OF MICROSERVICE ARCHITECTURES



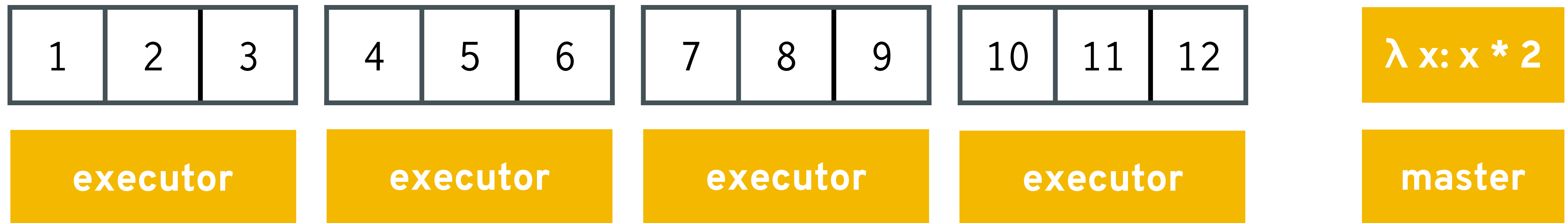
# BENEFITS OF MICROSERVICE ARCHITECTURES



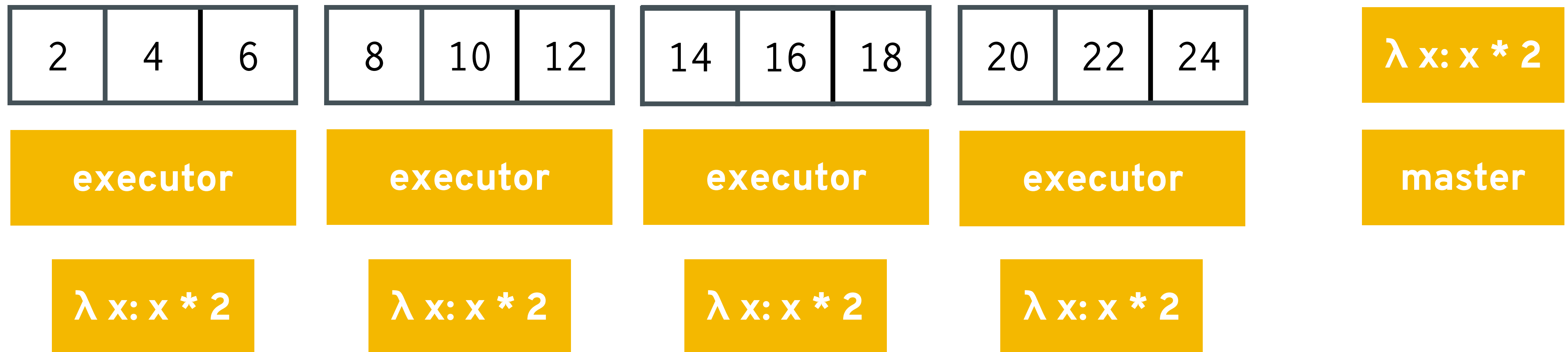
# MICROSERVICES AND SPARK



# MICROSERVICES AND SPARK



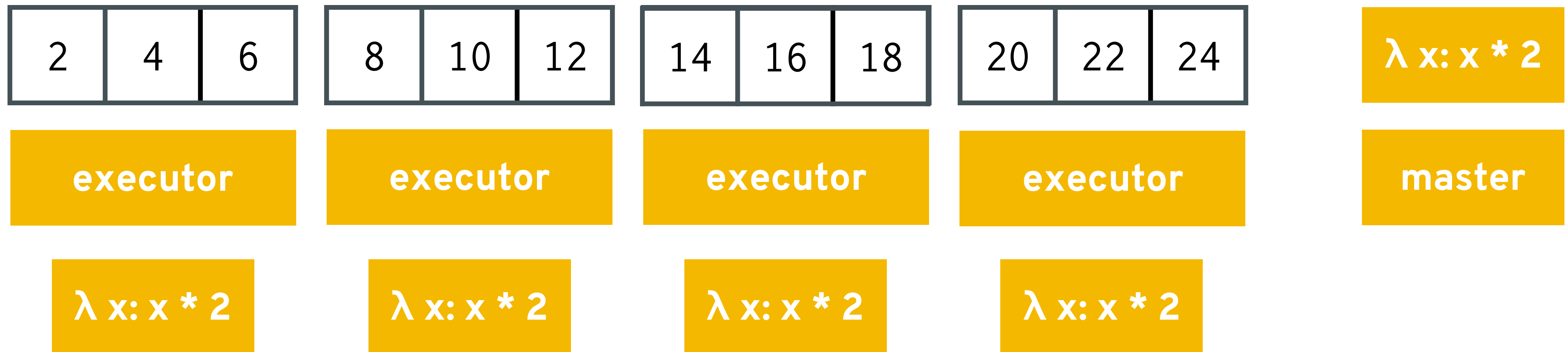
# MICROSERVICES AND SPARK



# MICROSERVICES AND SPARK



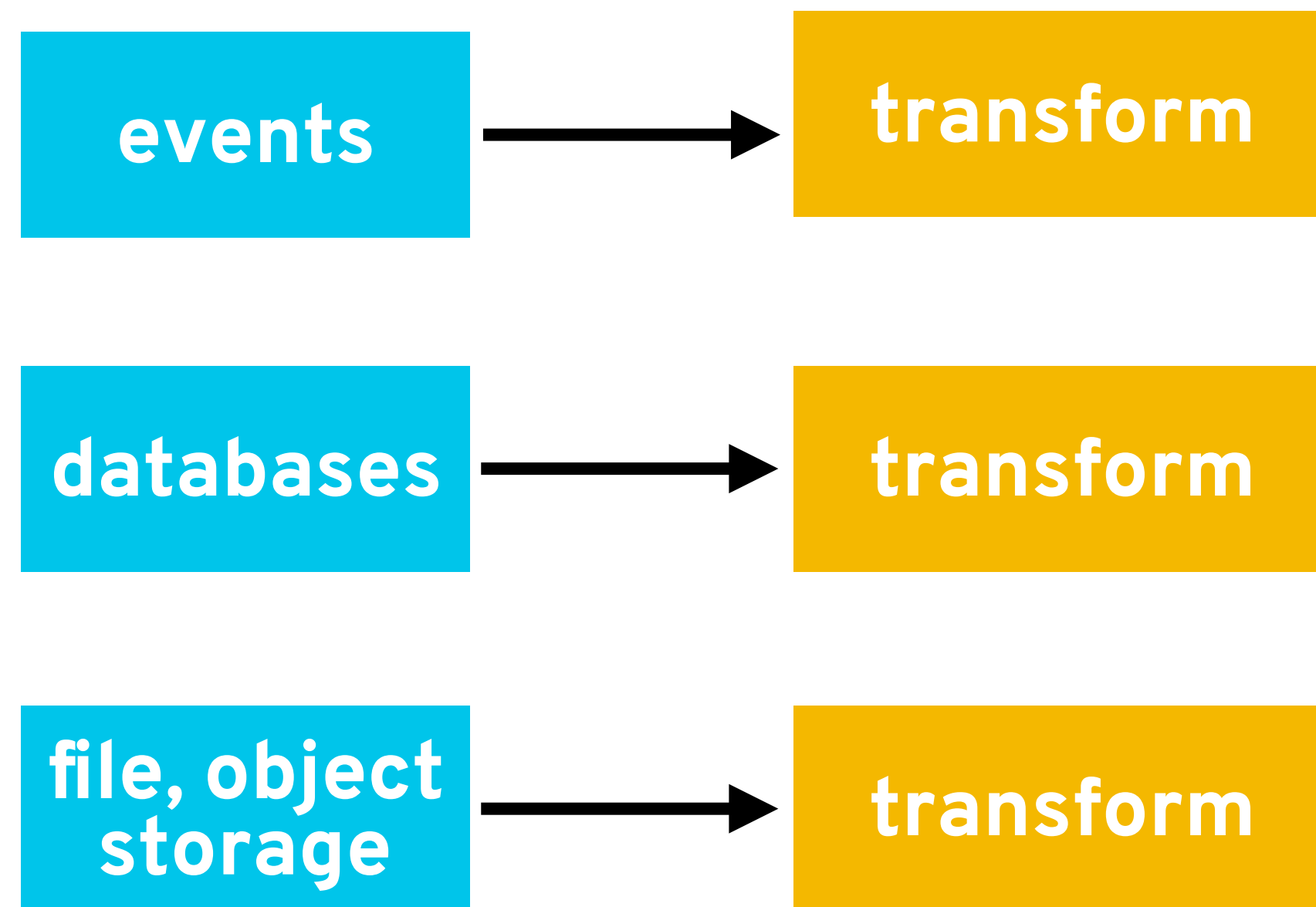
# MICROSERVICES AND SPARK



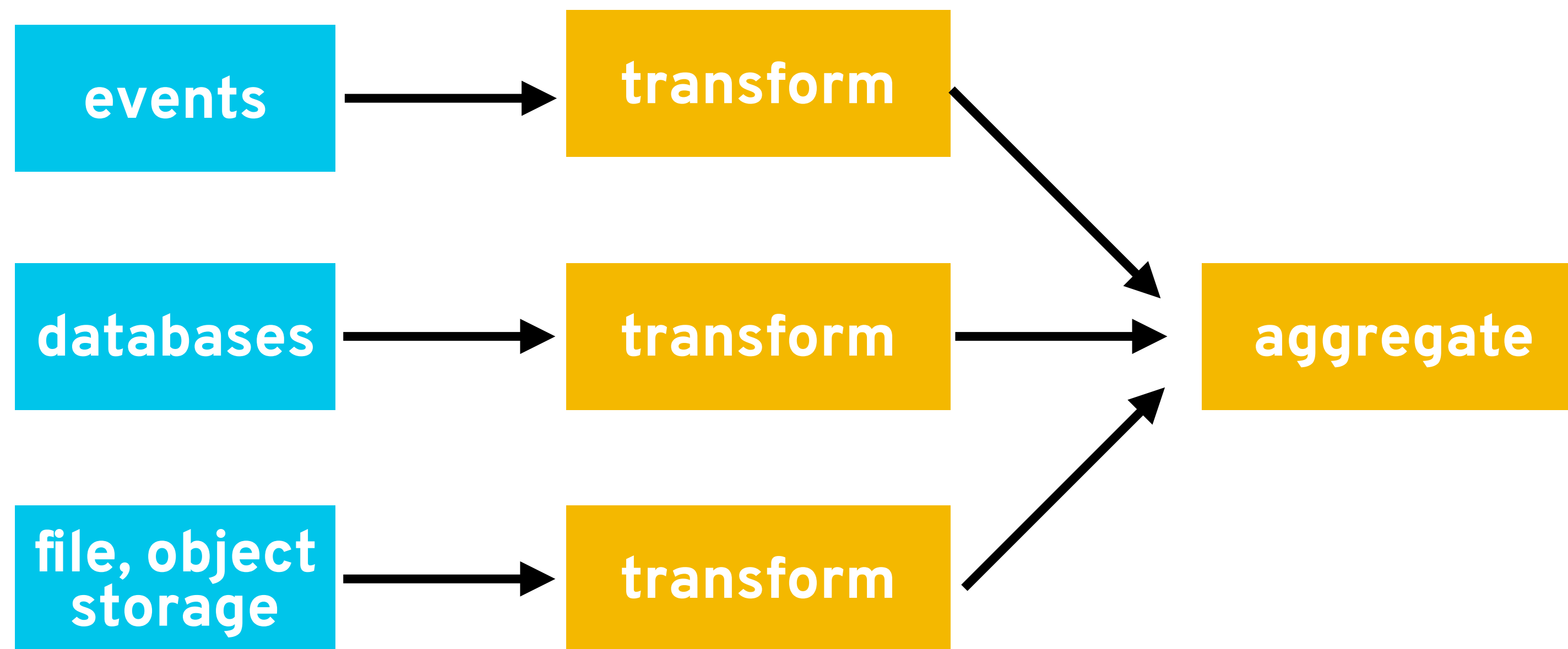


# ARCHITECTURES FOR ANALYTICS AND APPLICATIONS

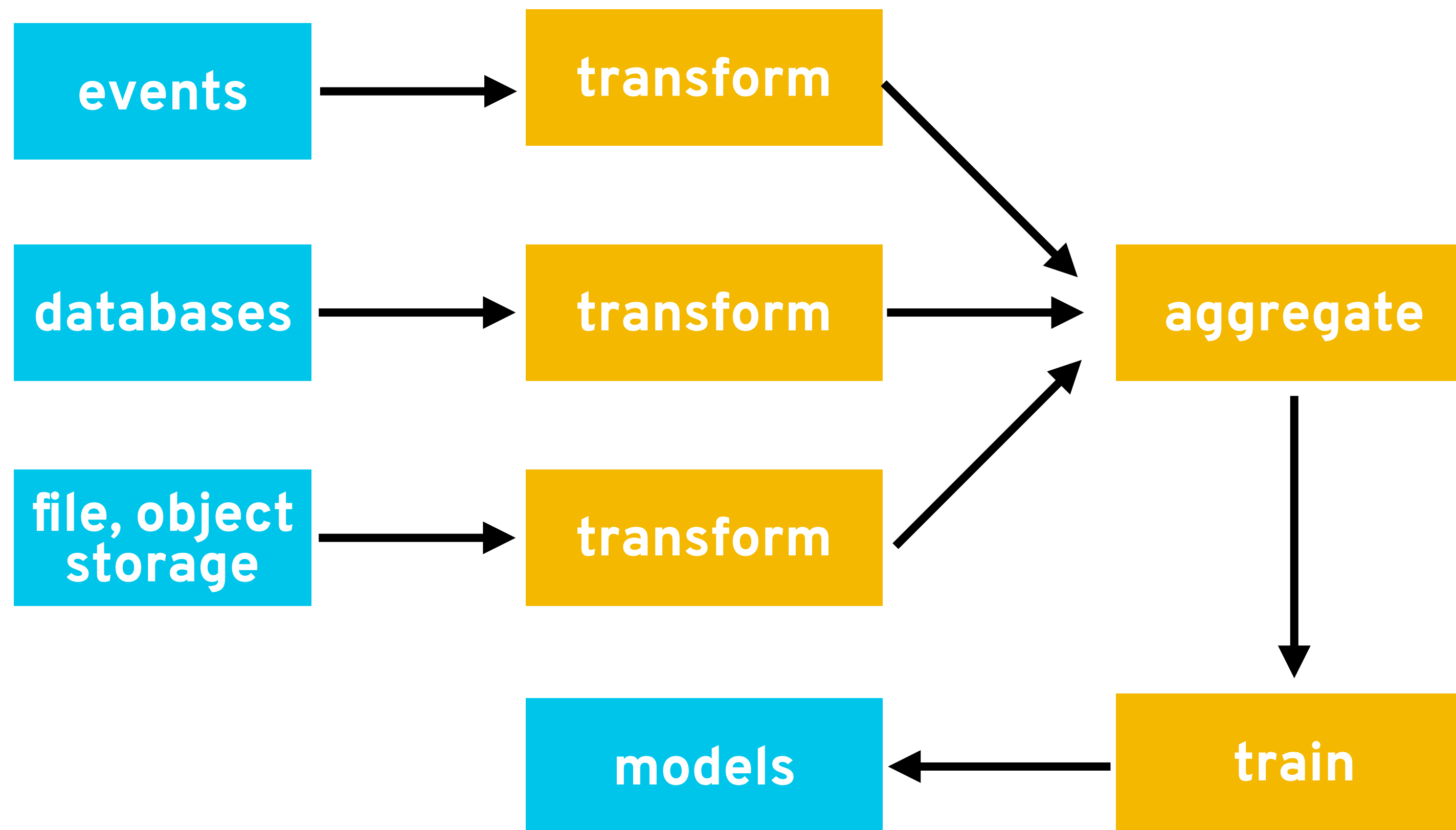
# APPLICATION RESPONSIBILITIES



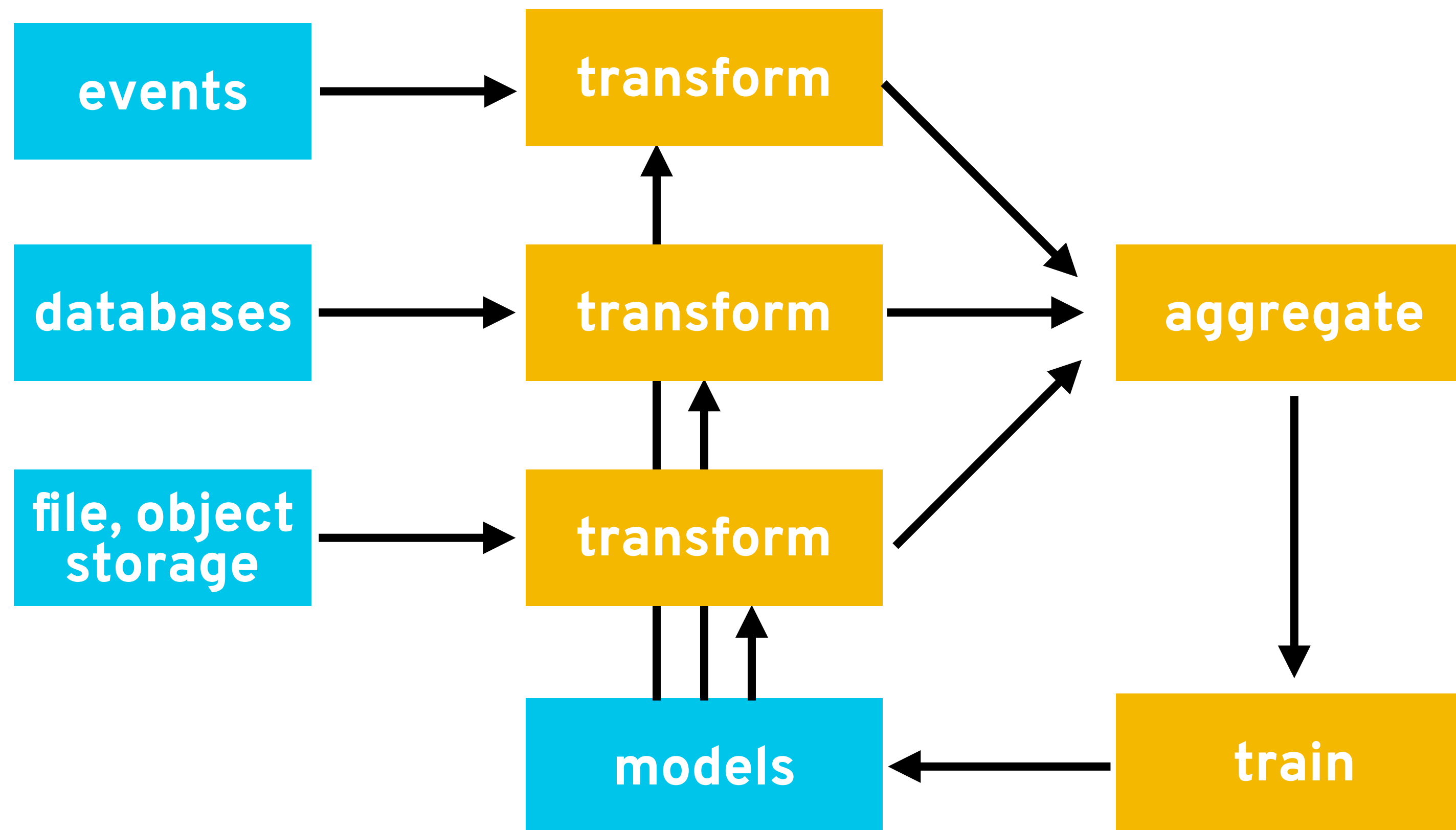
# APPLICATION RESPONSIBILITIES



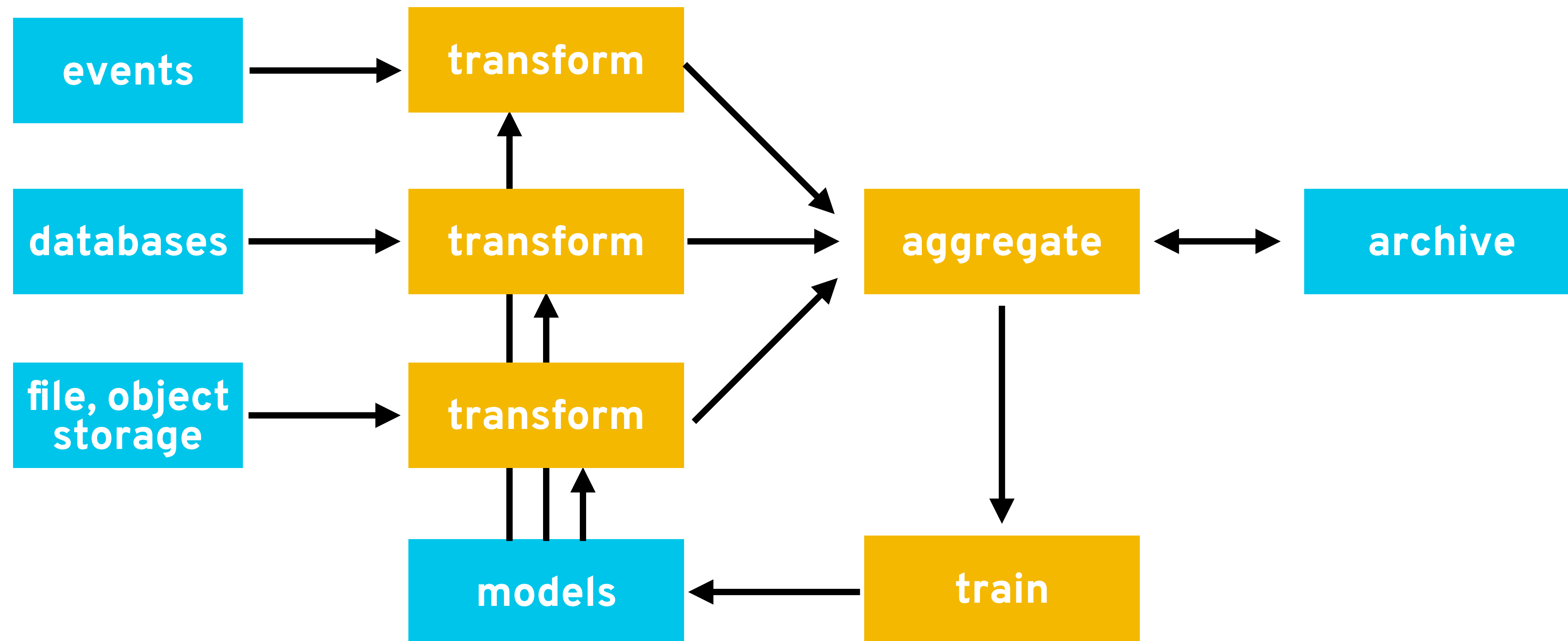
# APPLICATION RESPONSIBILITIES



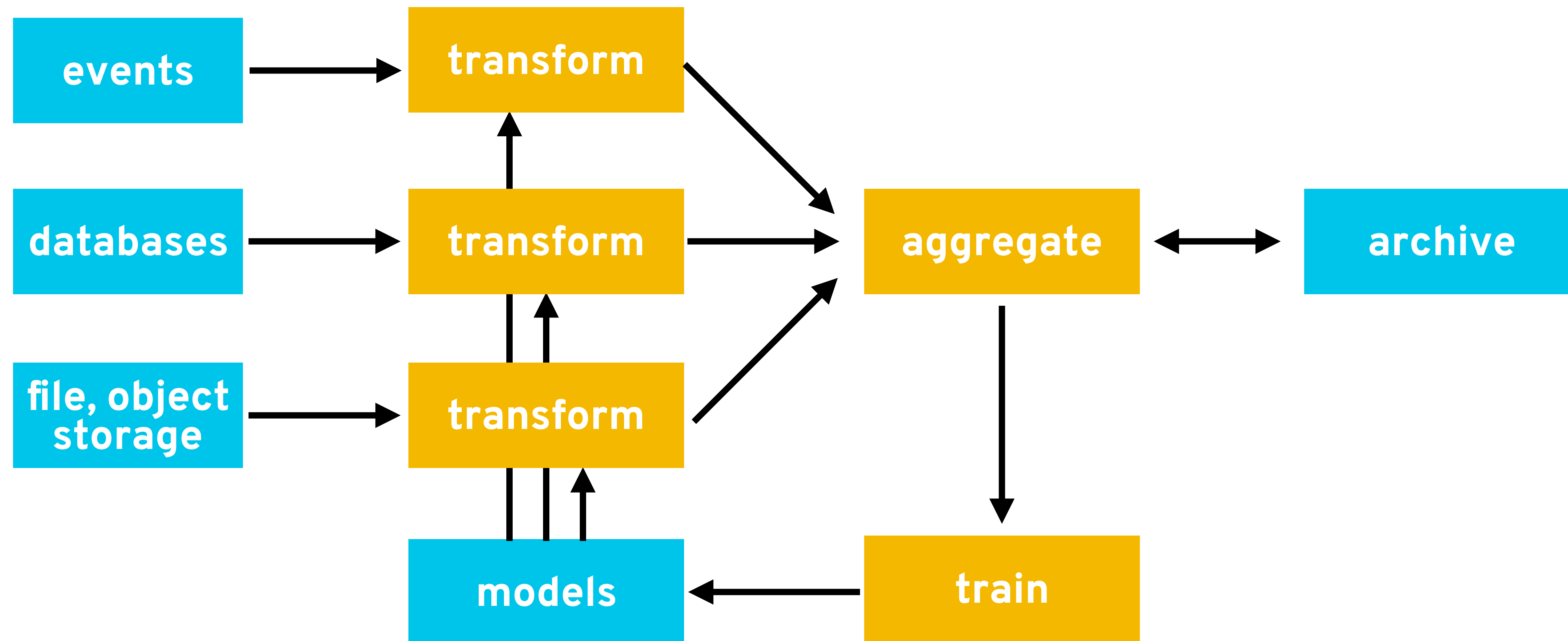
# APPLICATION RESPONSIBILITIES



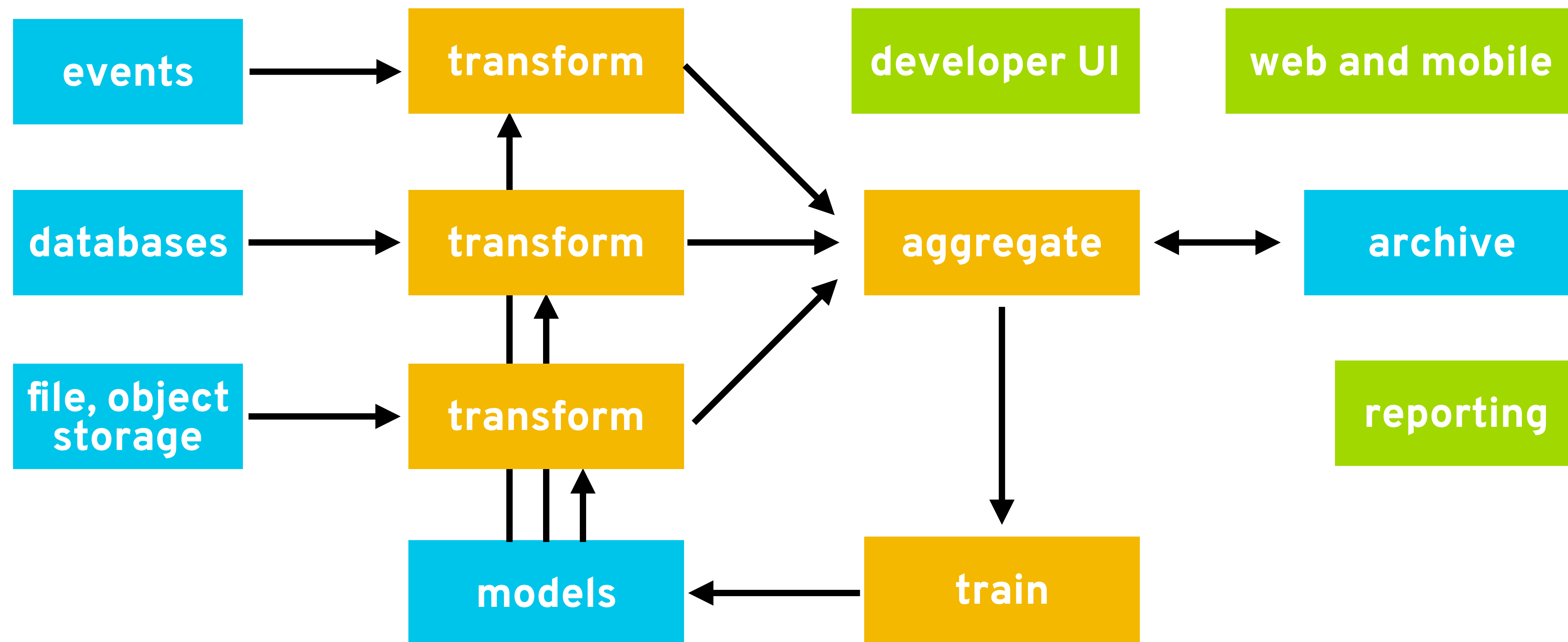
# APPLICATION RESPONSIBILITIES



# APPLICATION RESPONSIBILITIES

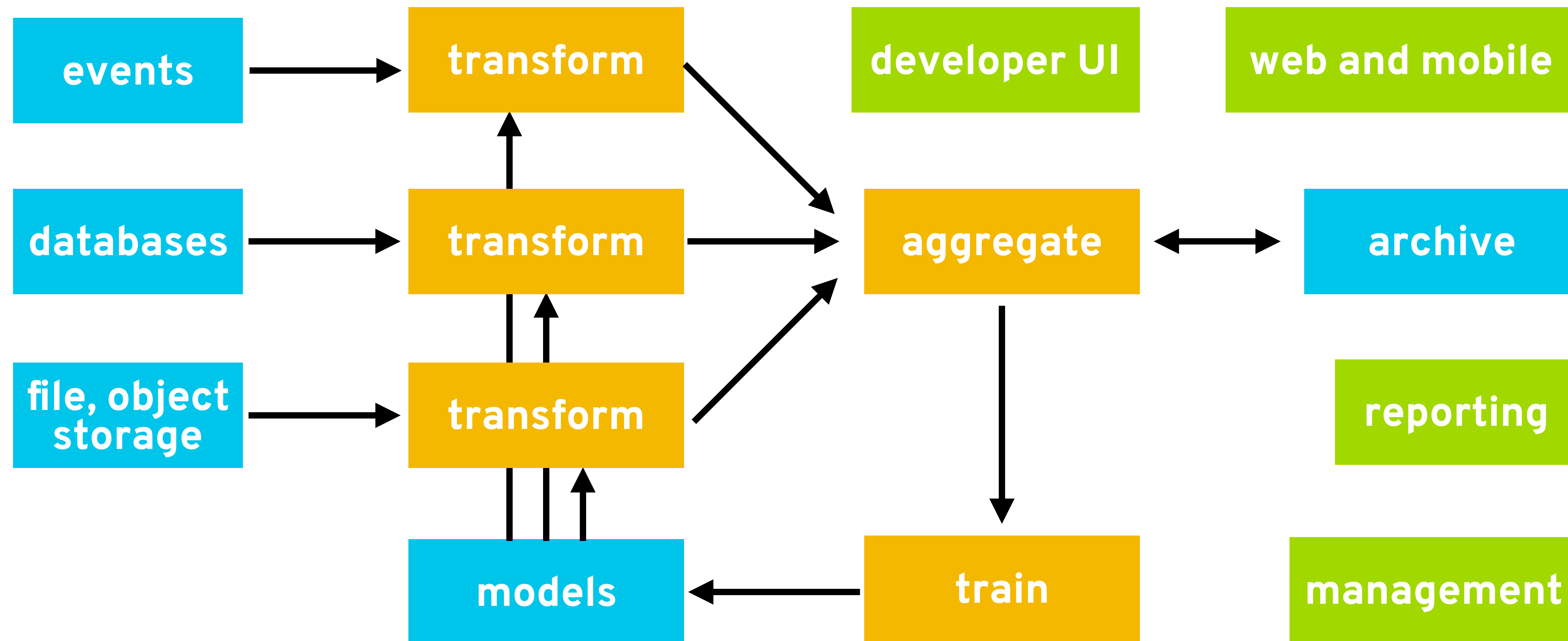


# APPLICATION RESPONSIBILITIES





# APPLICATION RESPONSIBILITIES

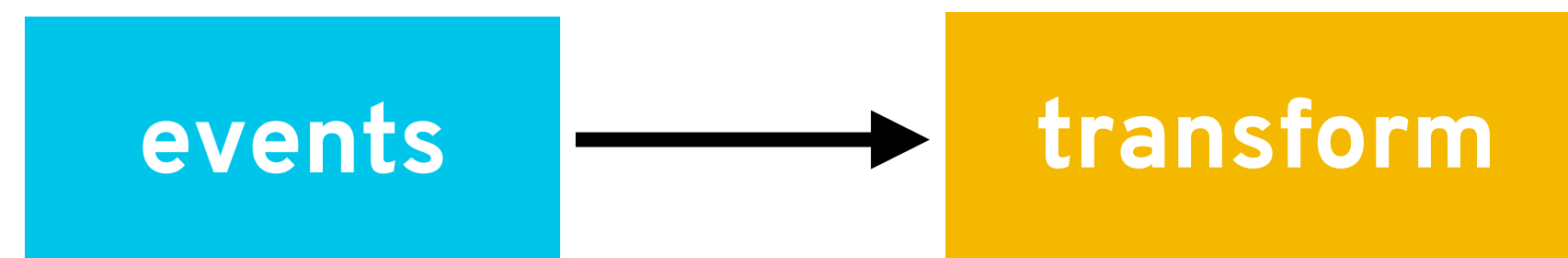


# LEGACY ARCHITECTURES

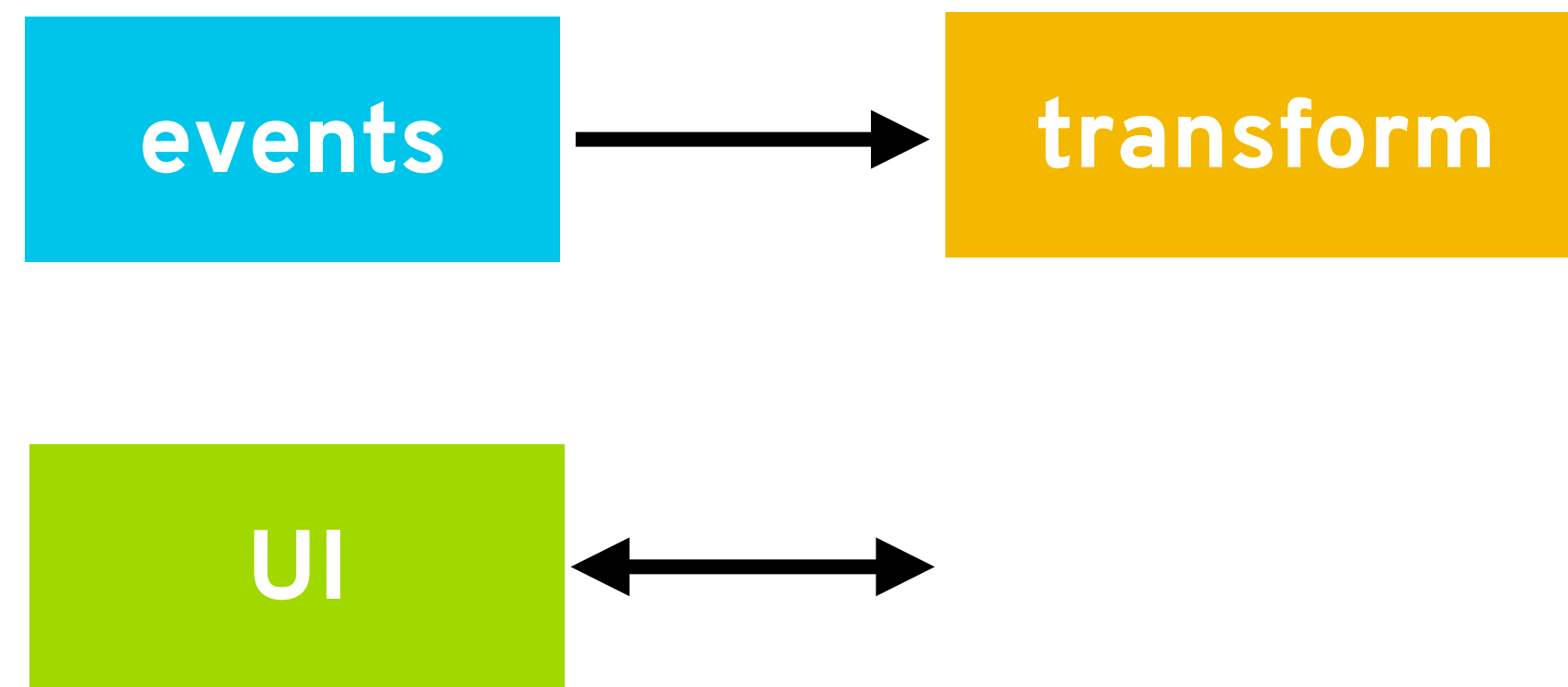
# CONVENTIONAL DATA WAREHOUSE

events

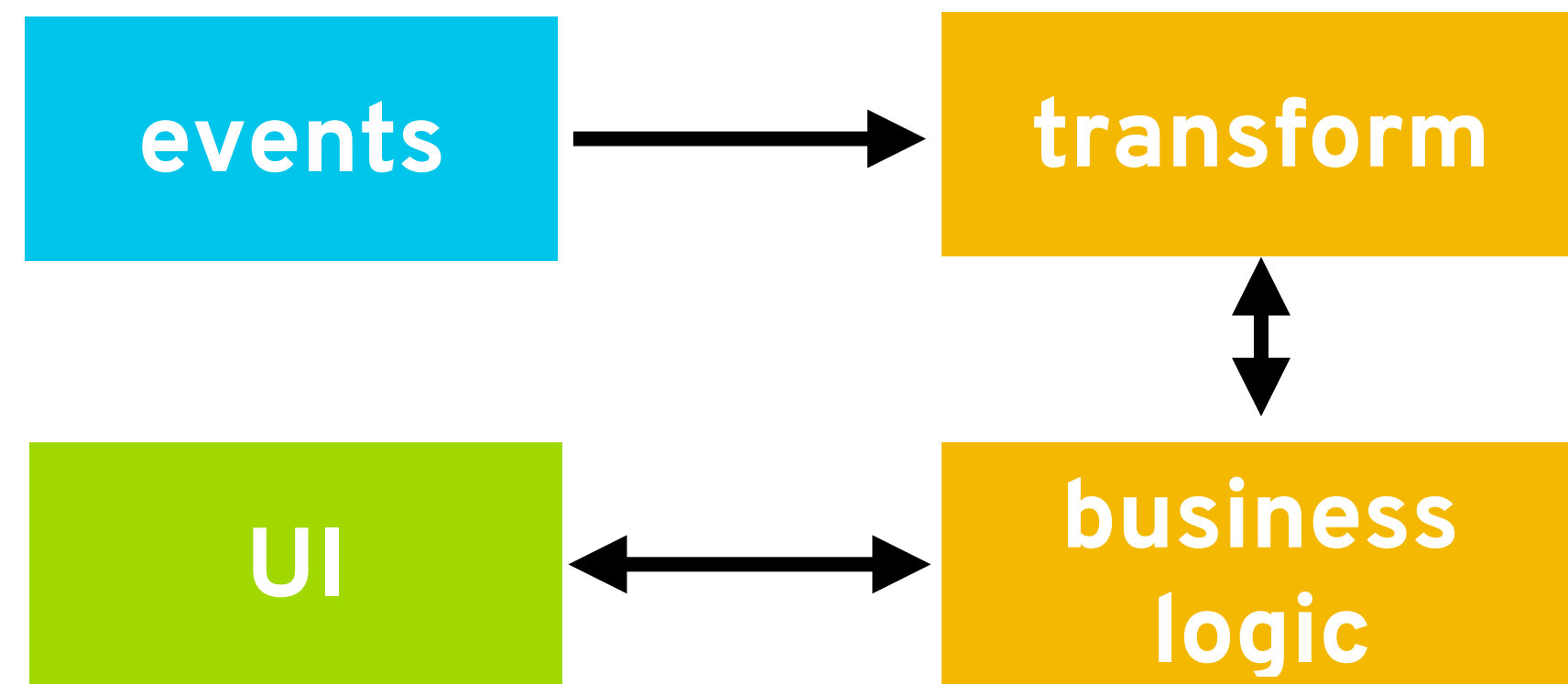
# CONVENTIONAL DATA WAREHOUSE



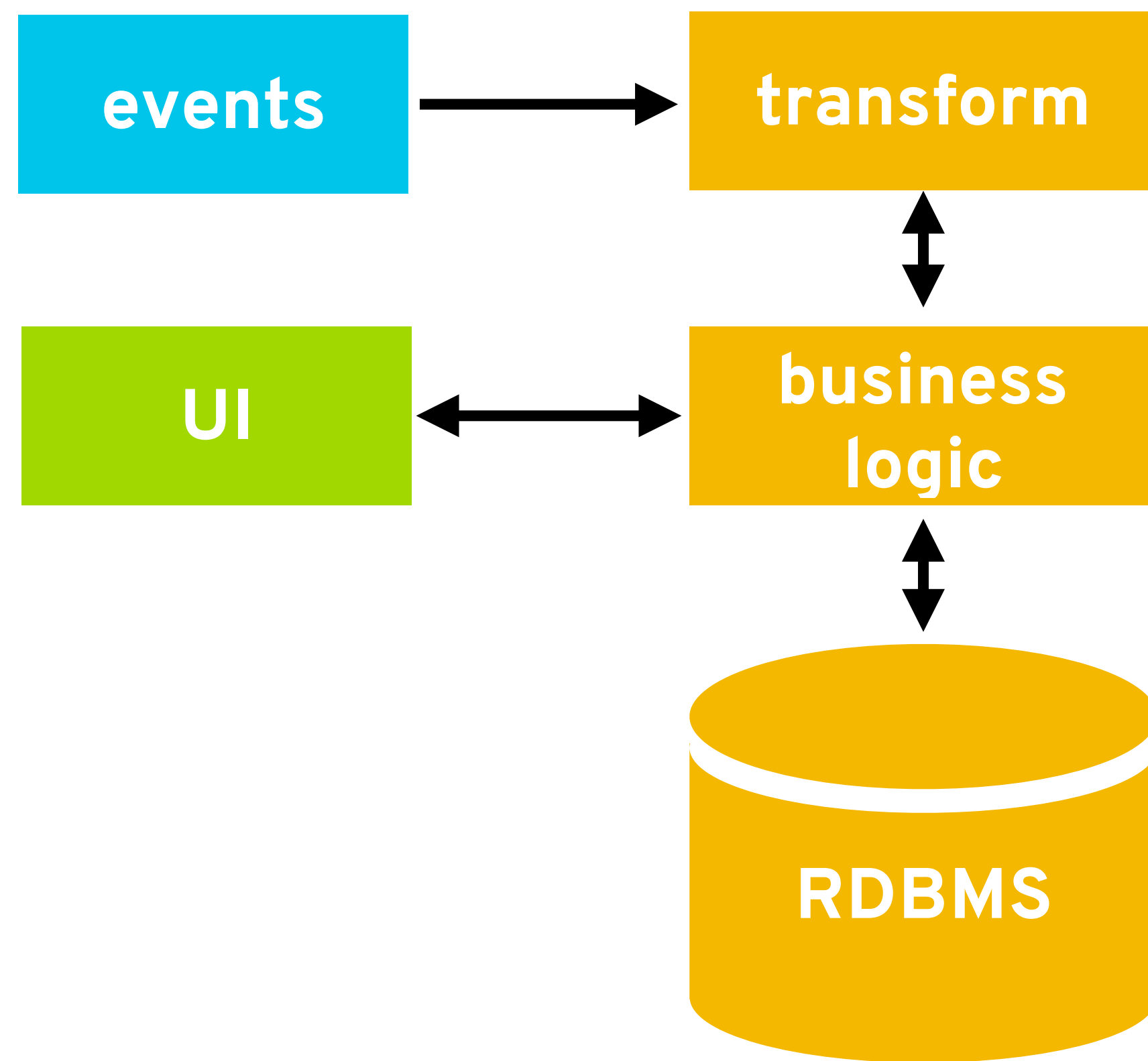
# CONVENTIONAL DATA WAREHOUSE



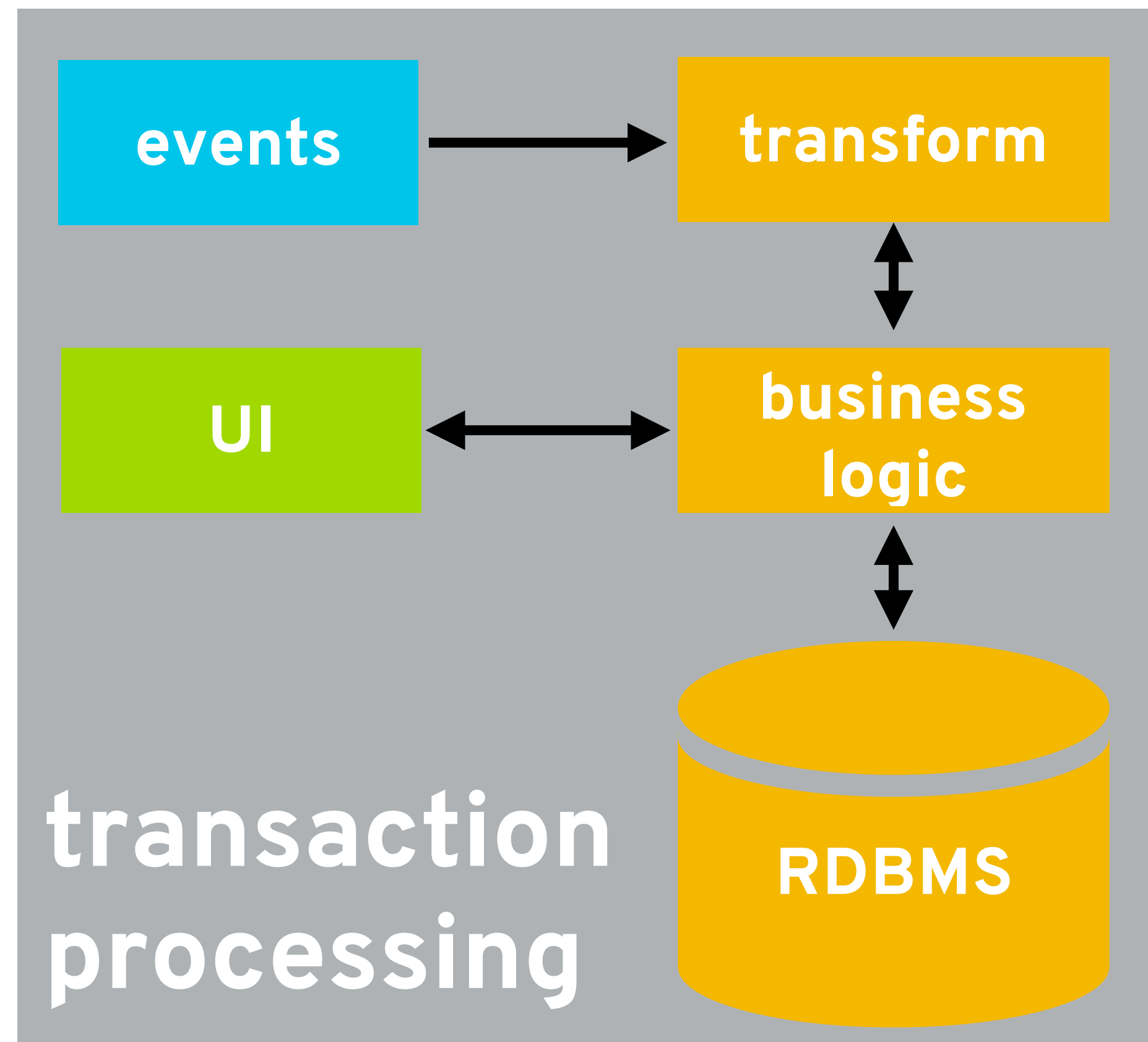
# CONVENTIONAL DATA WAREHOUSE



# CONVENTIONAL DATA WAREHOUSE

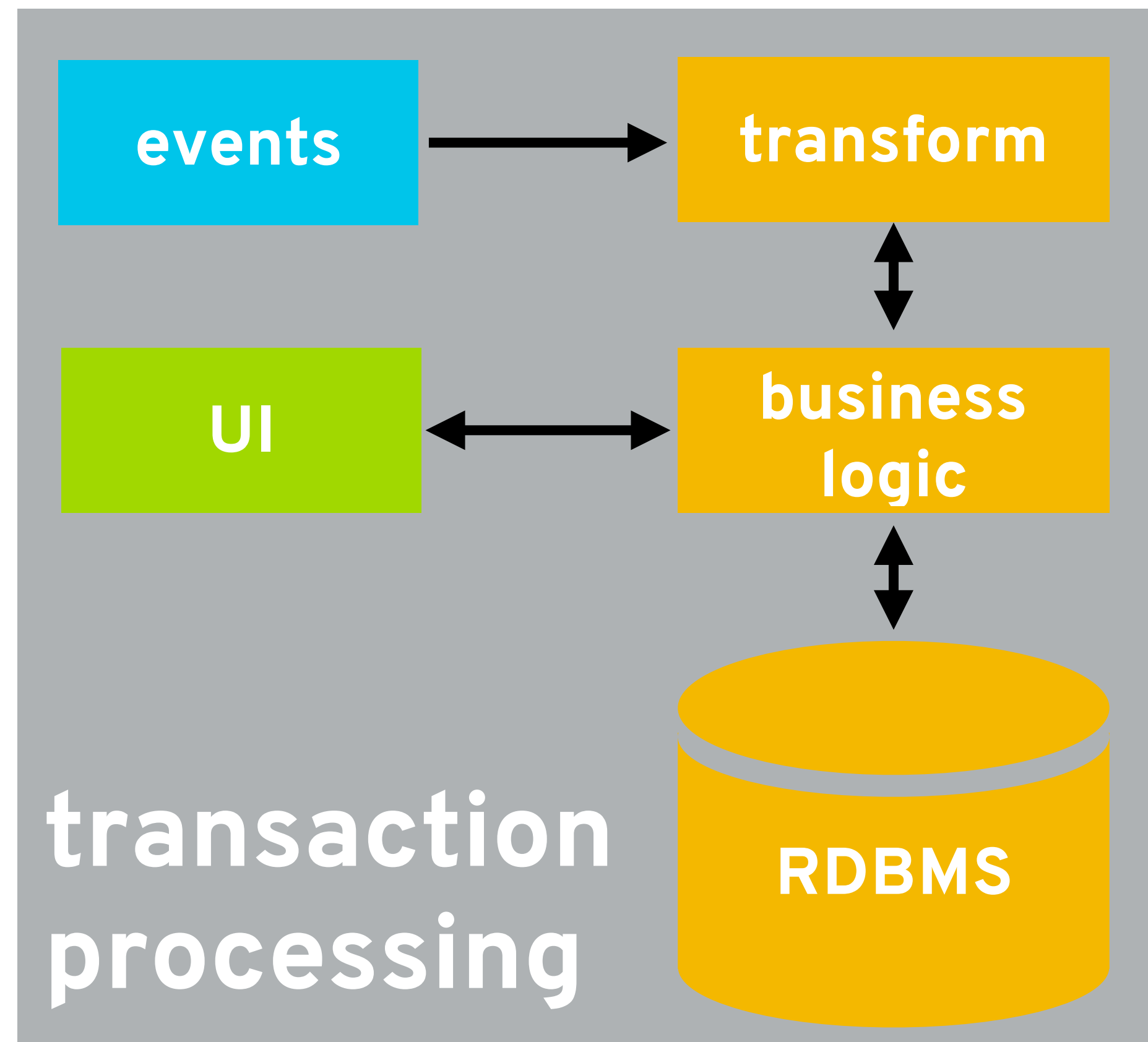


# CONVENTIONAL DATA WAREHOUSE

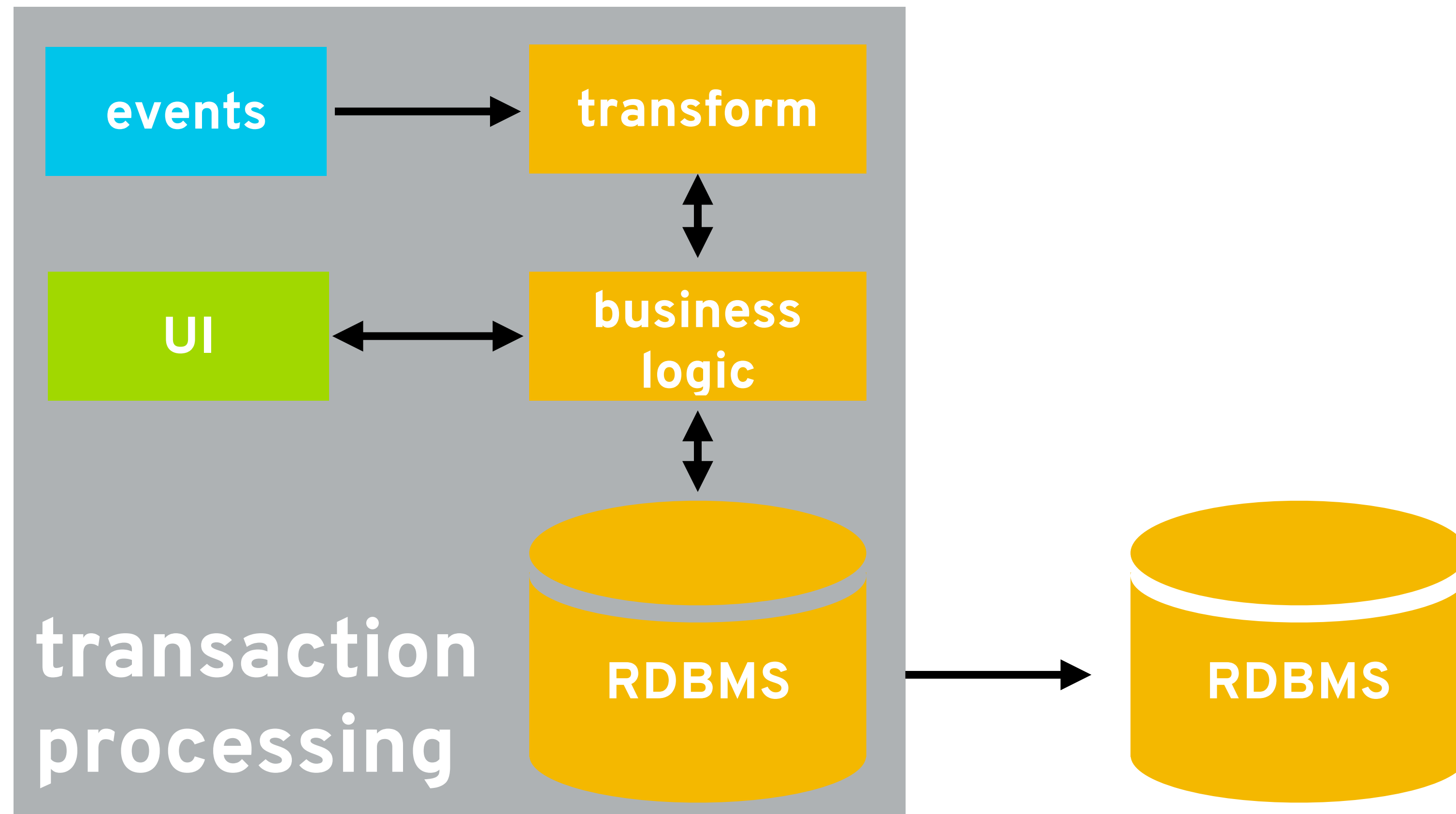




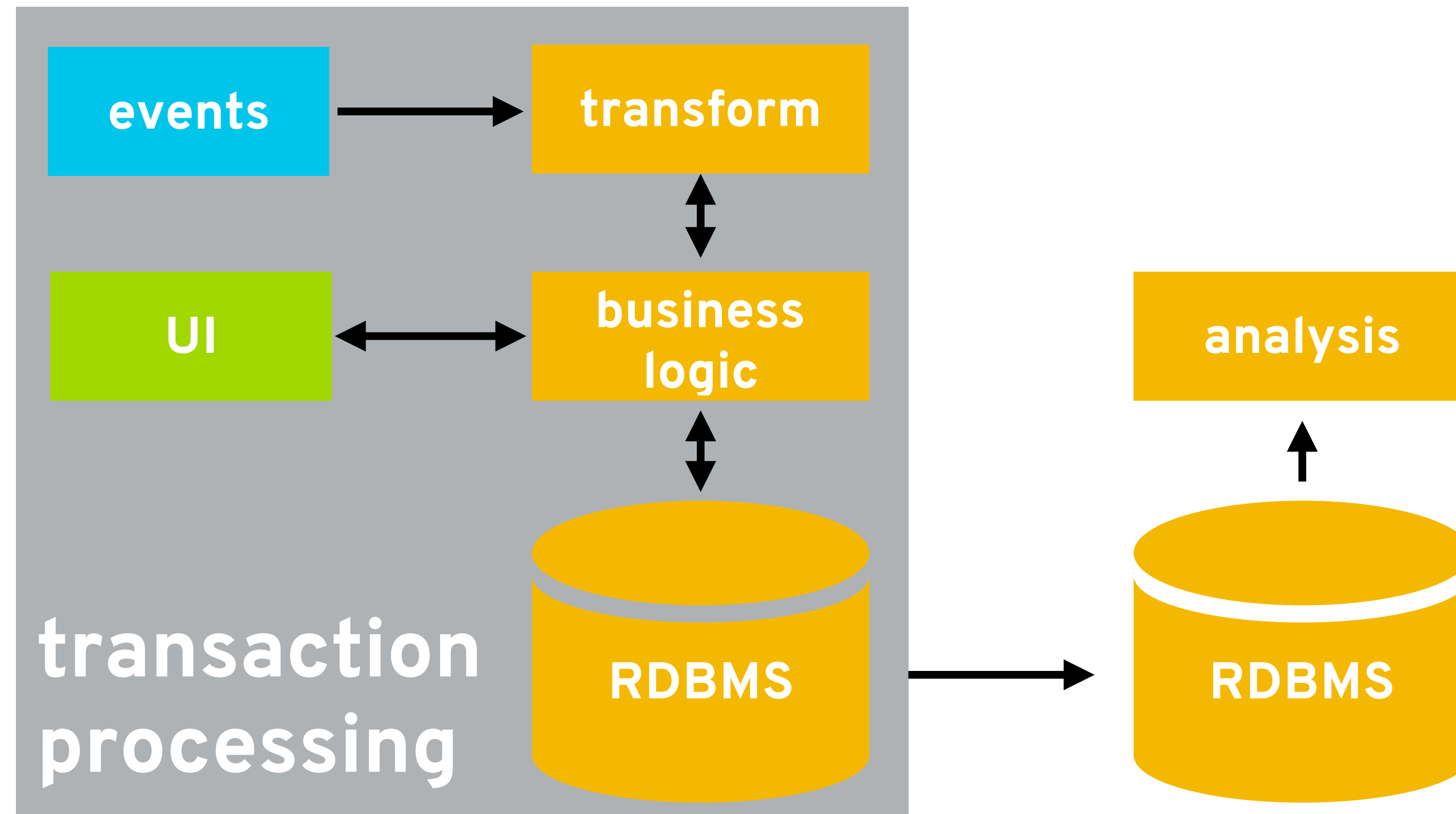
# CONVENTIONAL DATA WAREHOUSE



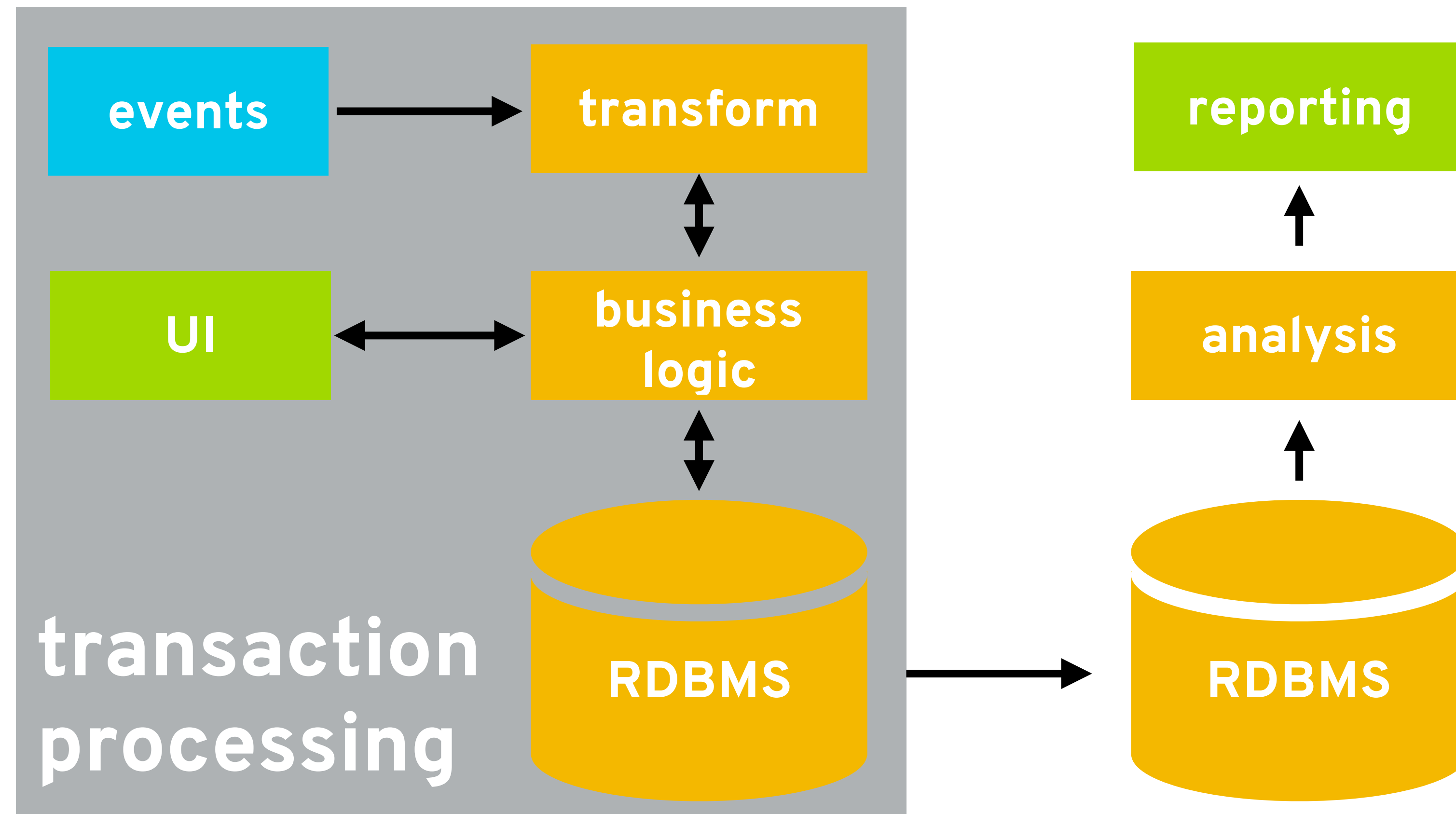
# CONVENTIONAL DATA WAREHOUSE



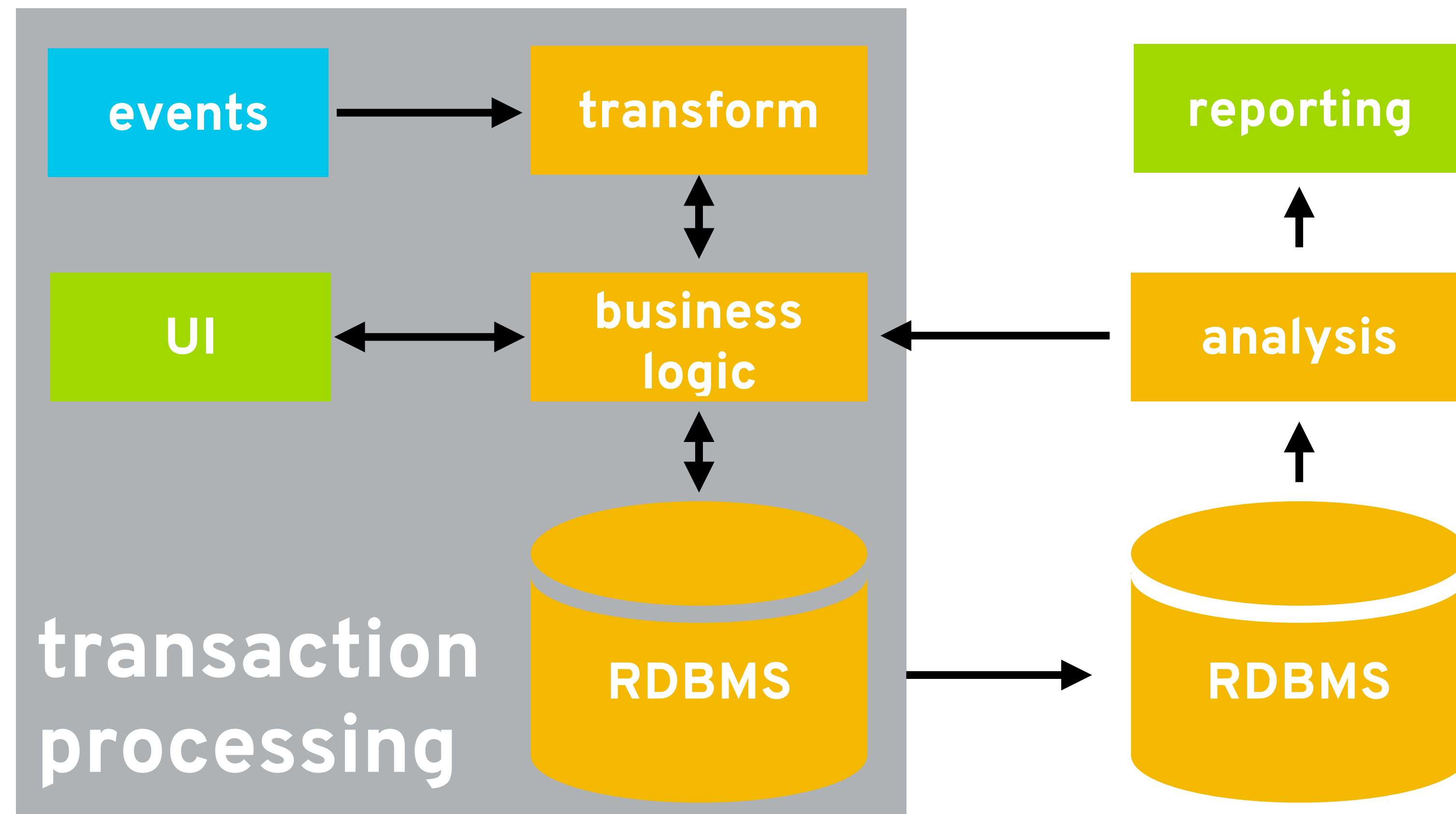
# CONVENTIONAL DATA WAREHOUSE



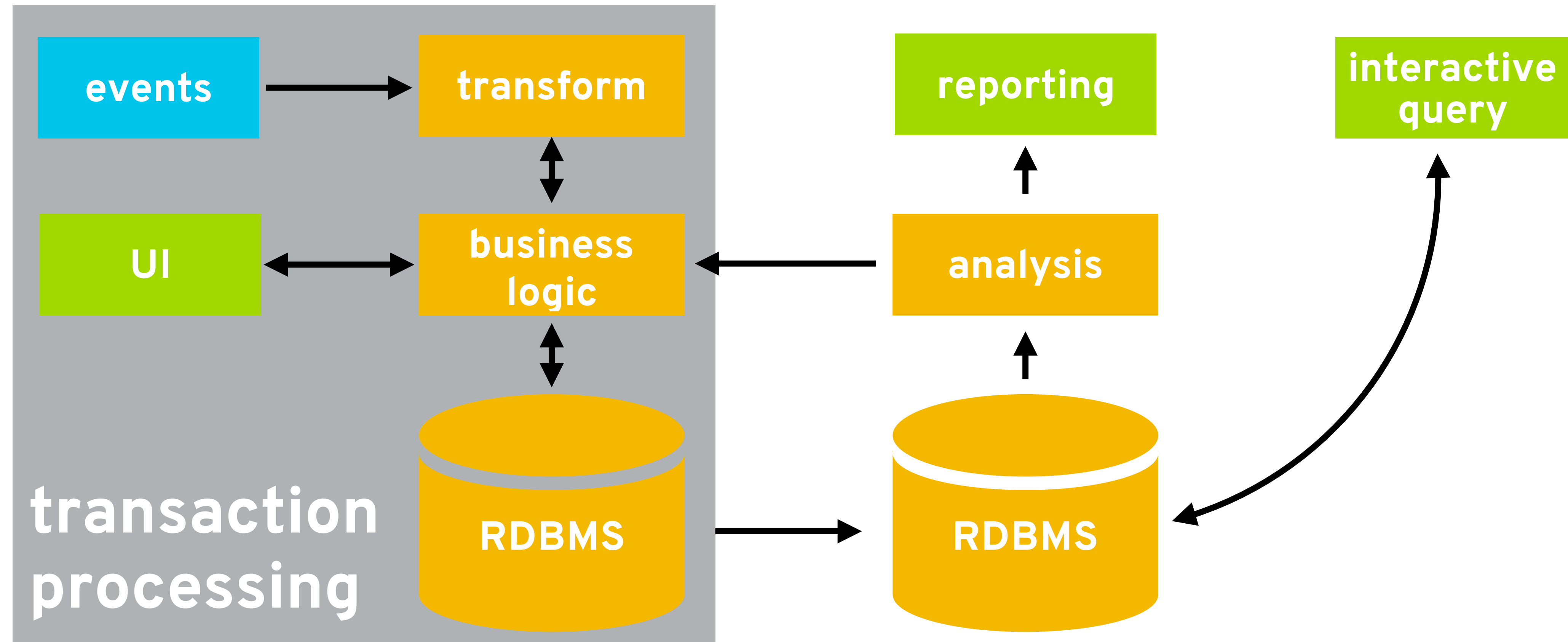
# CONVENTIONAL DATA WAREHOUSE



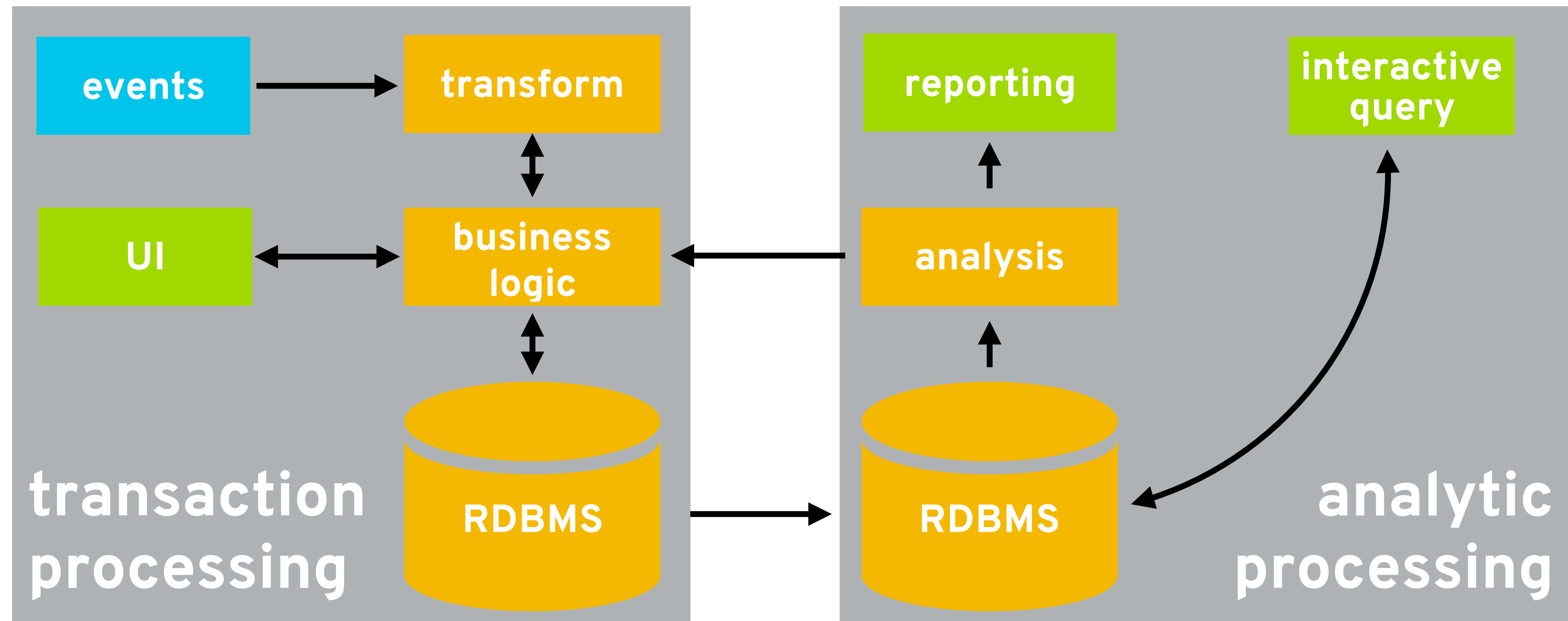
# CONVENTIONAL DATA WAREHOUSE



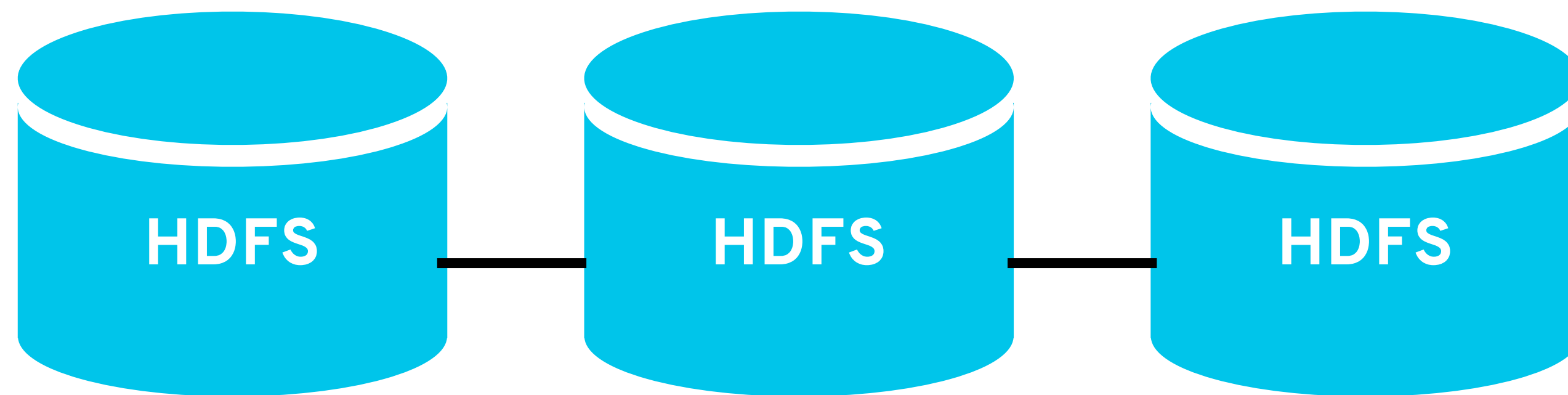
# CONVENTIONAL DATA WAREHOUSE



# CONVENTIONAL DATA WAREHOUSE

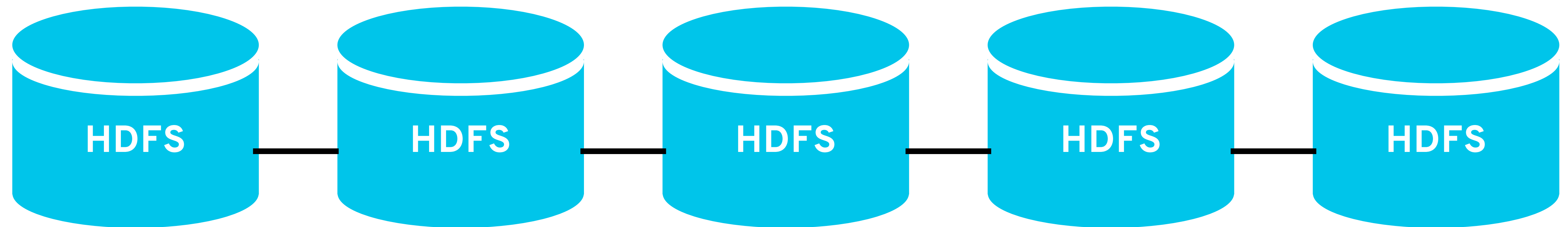


# HADOOP-STYLE “DATA LAKE”

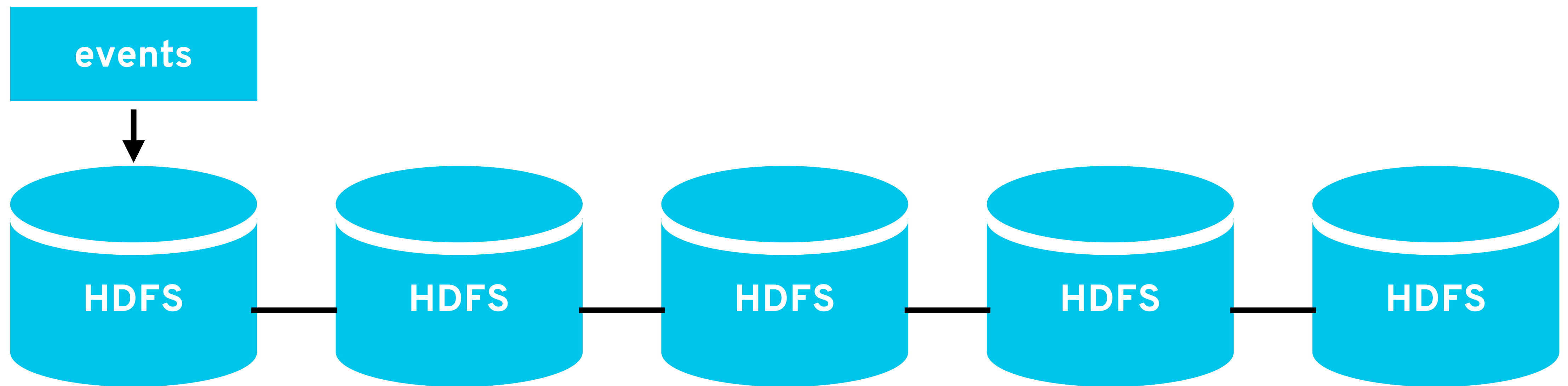




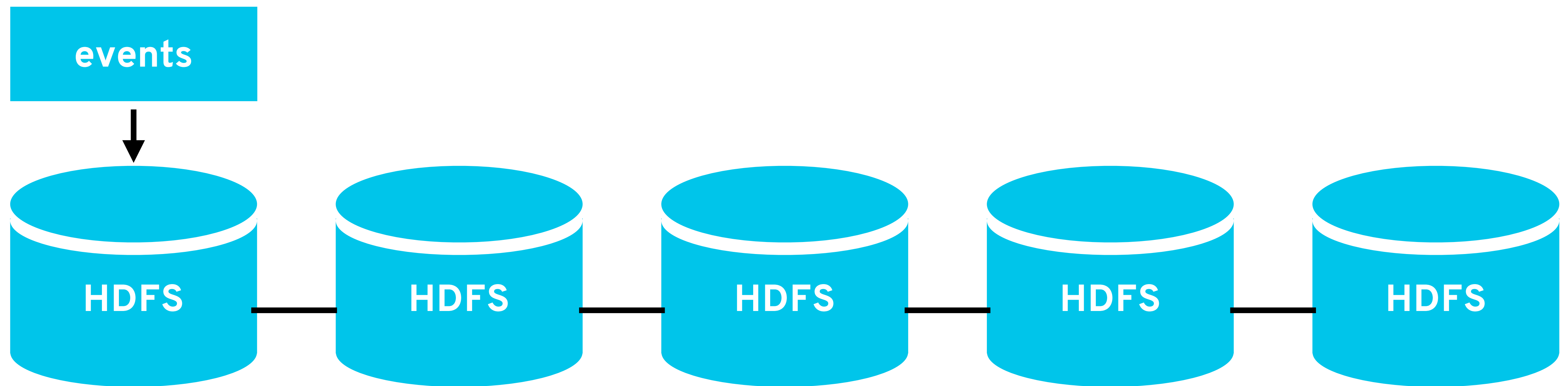
# HADOOP-STYLE “DATA LAKE”



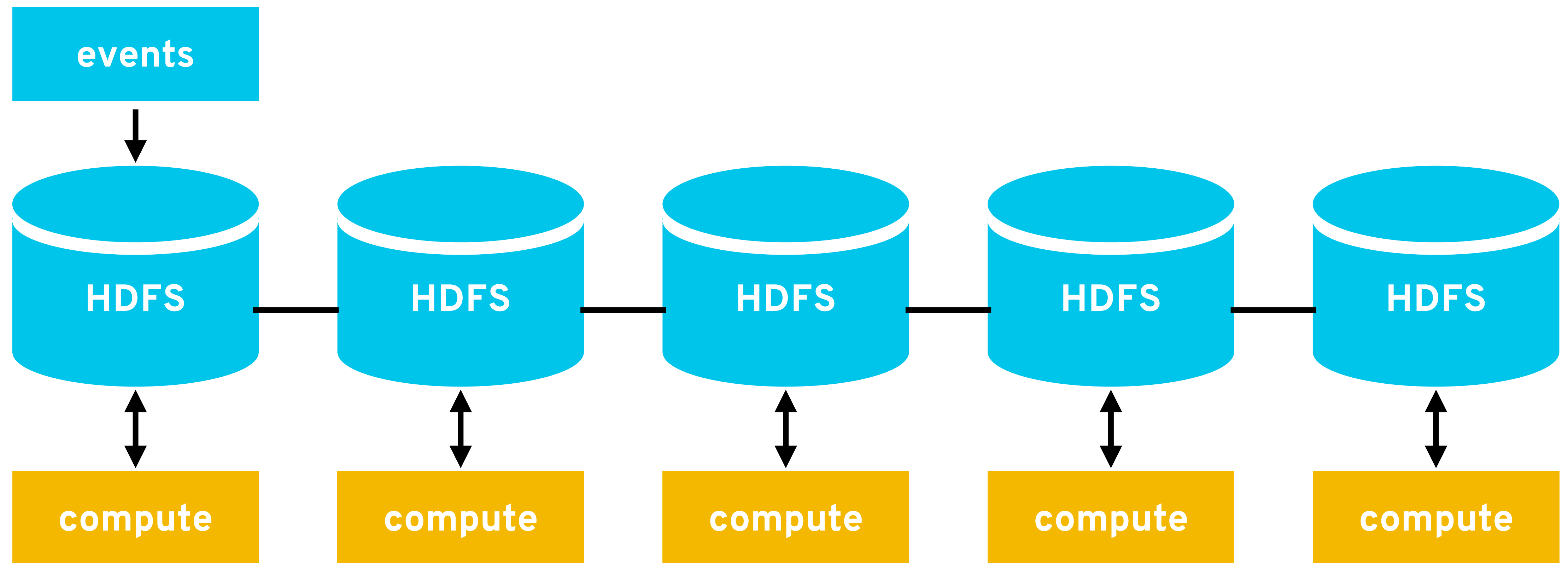
# HADOOP-STYLE “DATA LAKE”



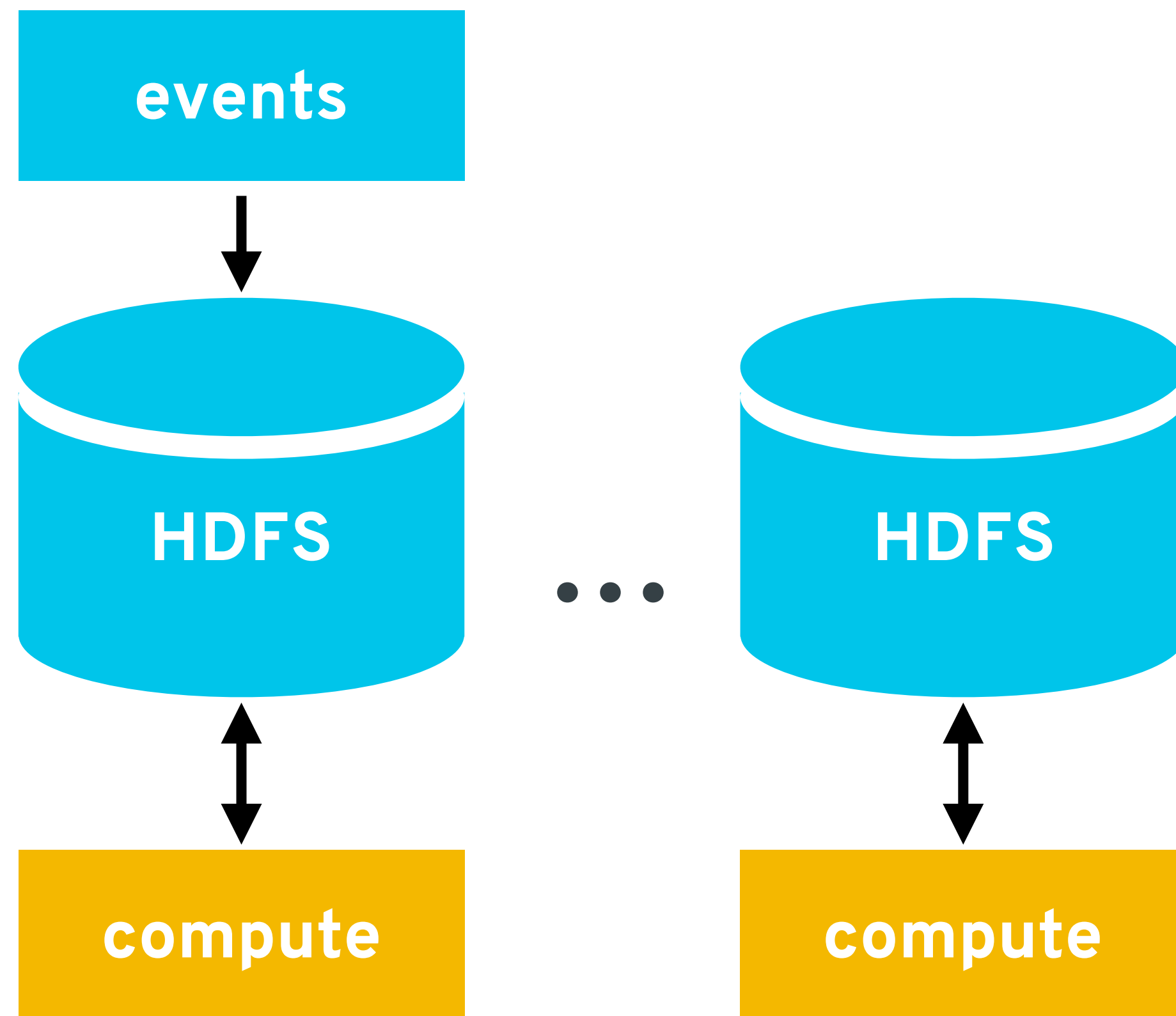
# HADOOP-STYLE “DATA LAKE”



# HADOOP-STYLE “DATA LAKE”



# HADOOP-STYLE “DATA LAKE”

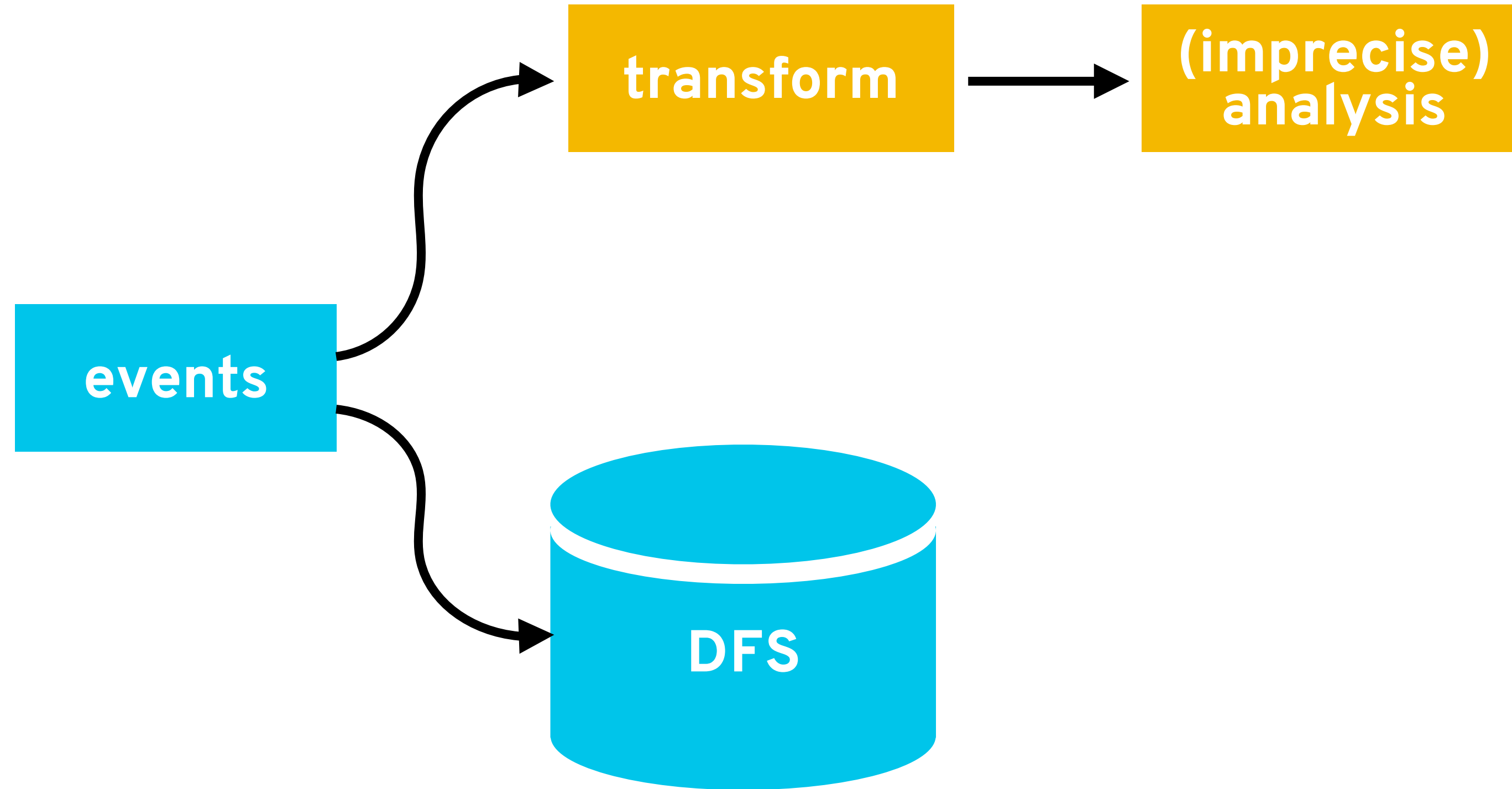


# MODERN ARCHITECTURES

# THE LAMBDA ARCHITECTURE

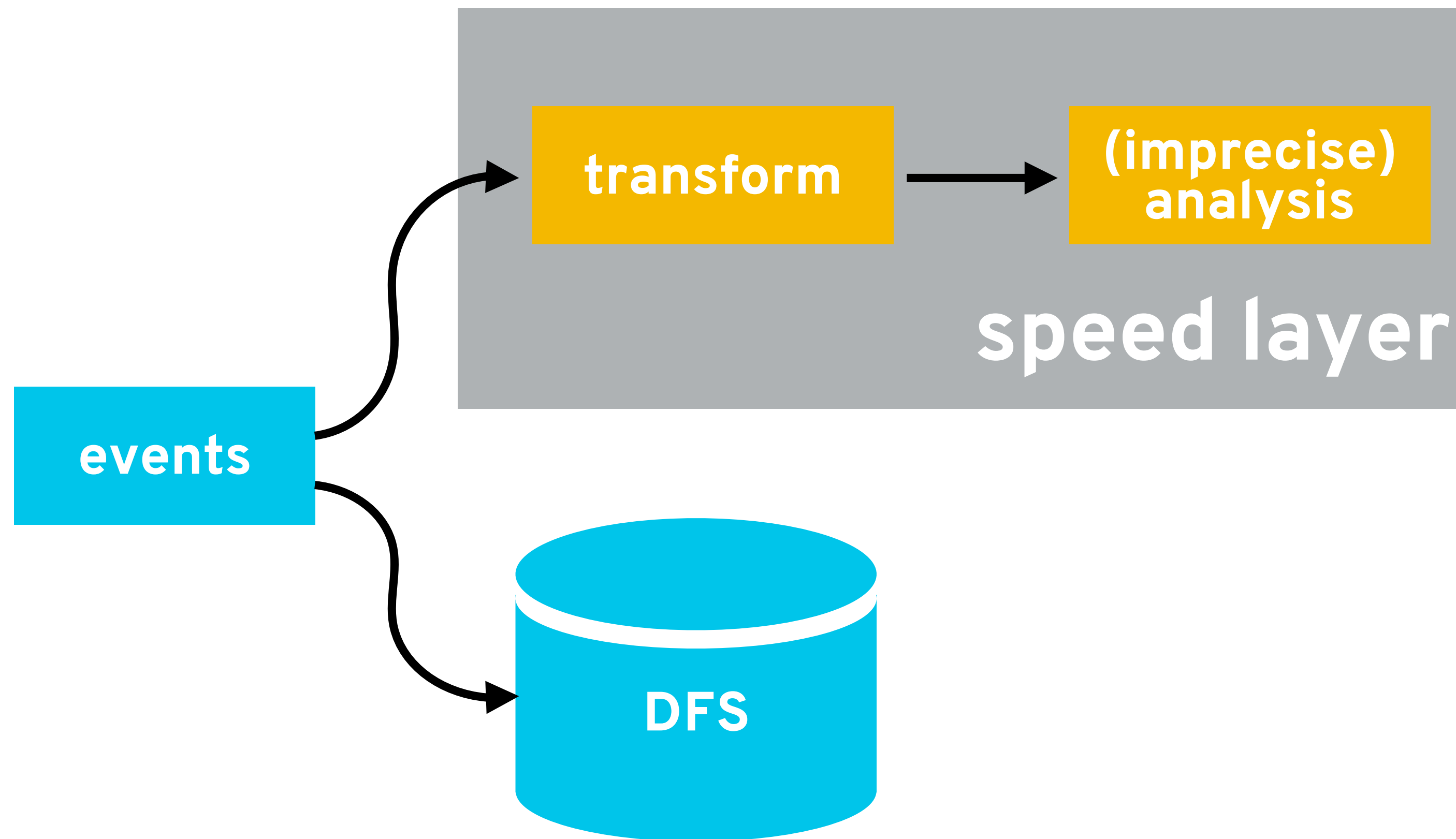


# THE LAMBDA ARCHITECTURE

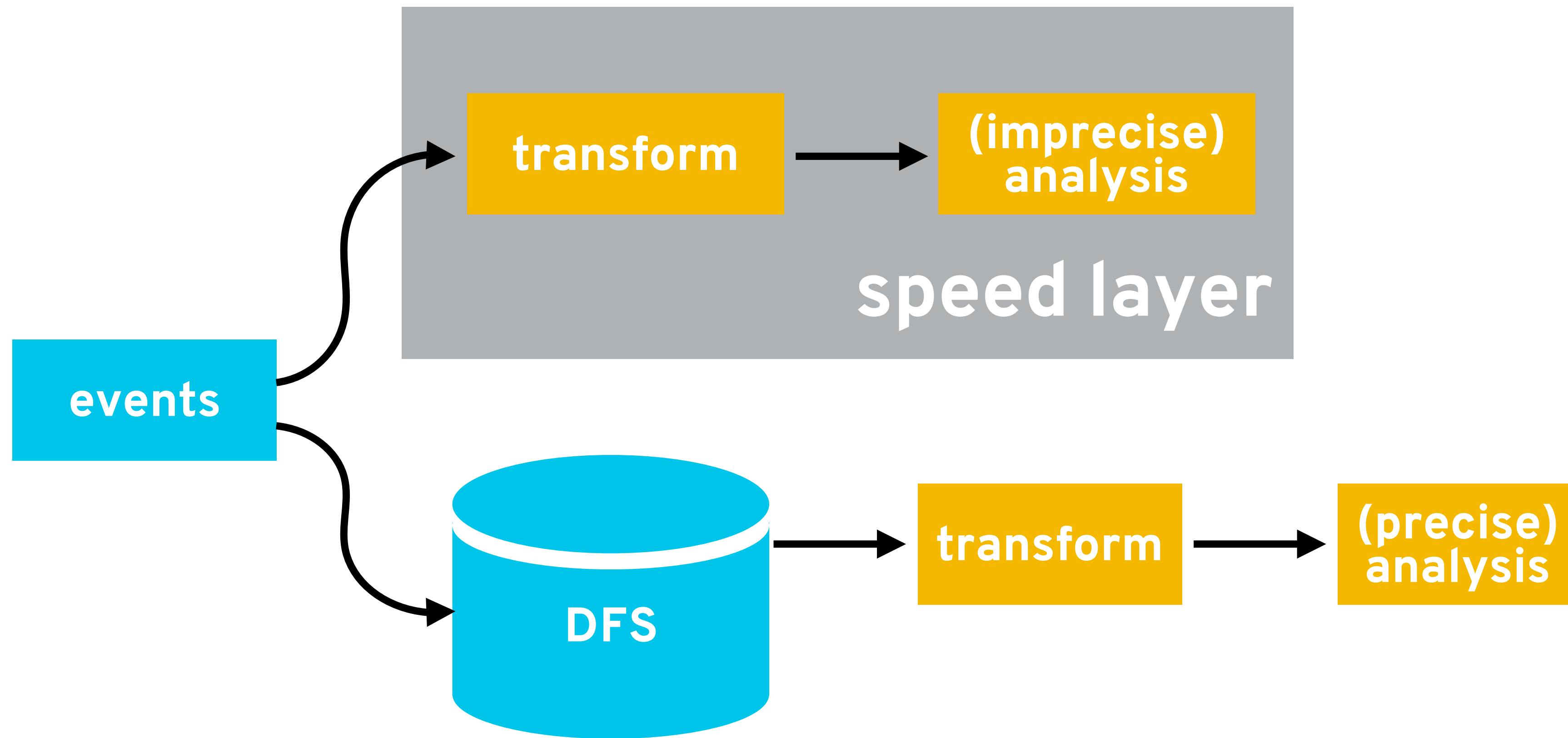




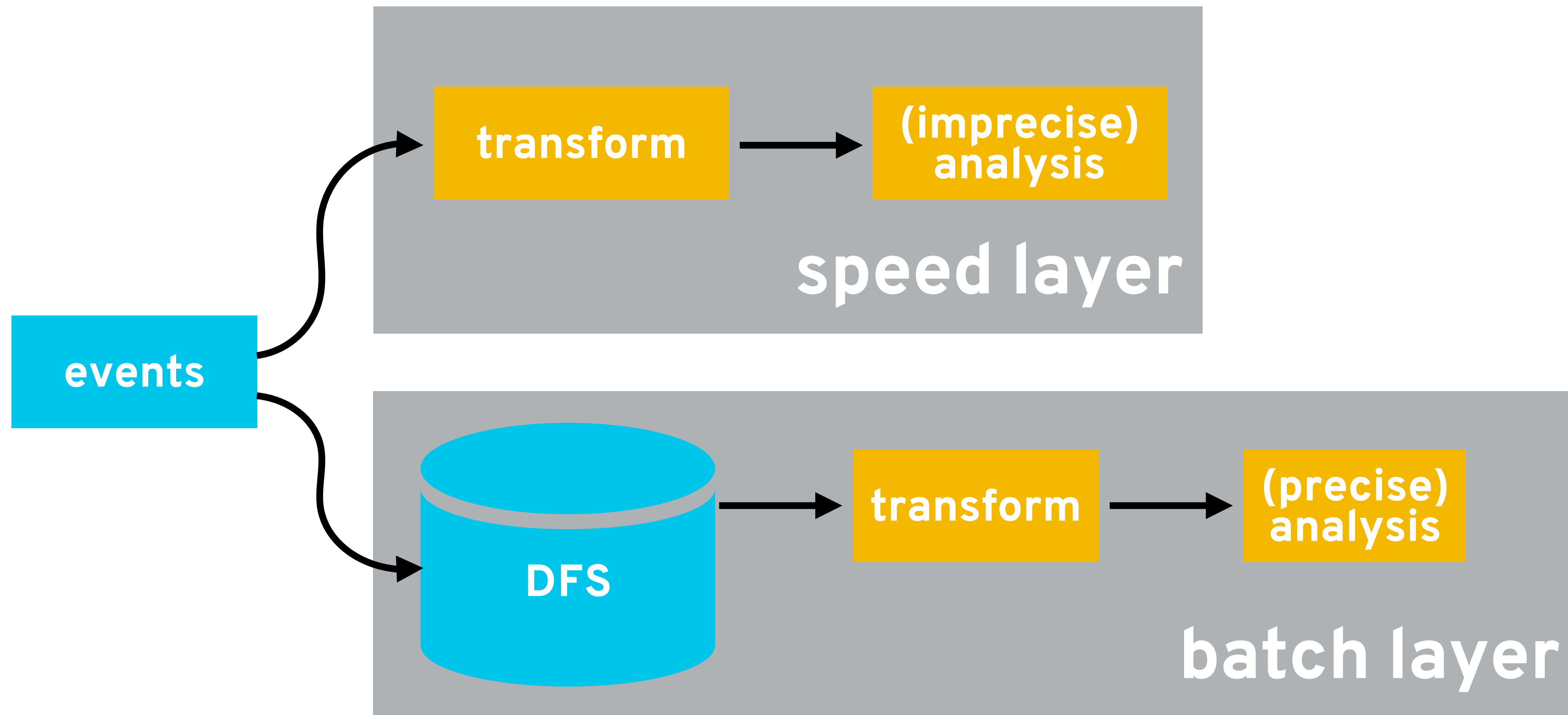
# THE LAMBDA ARCHITECTURE



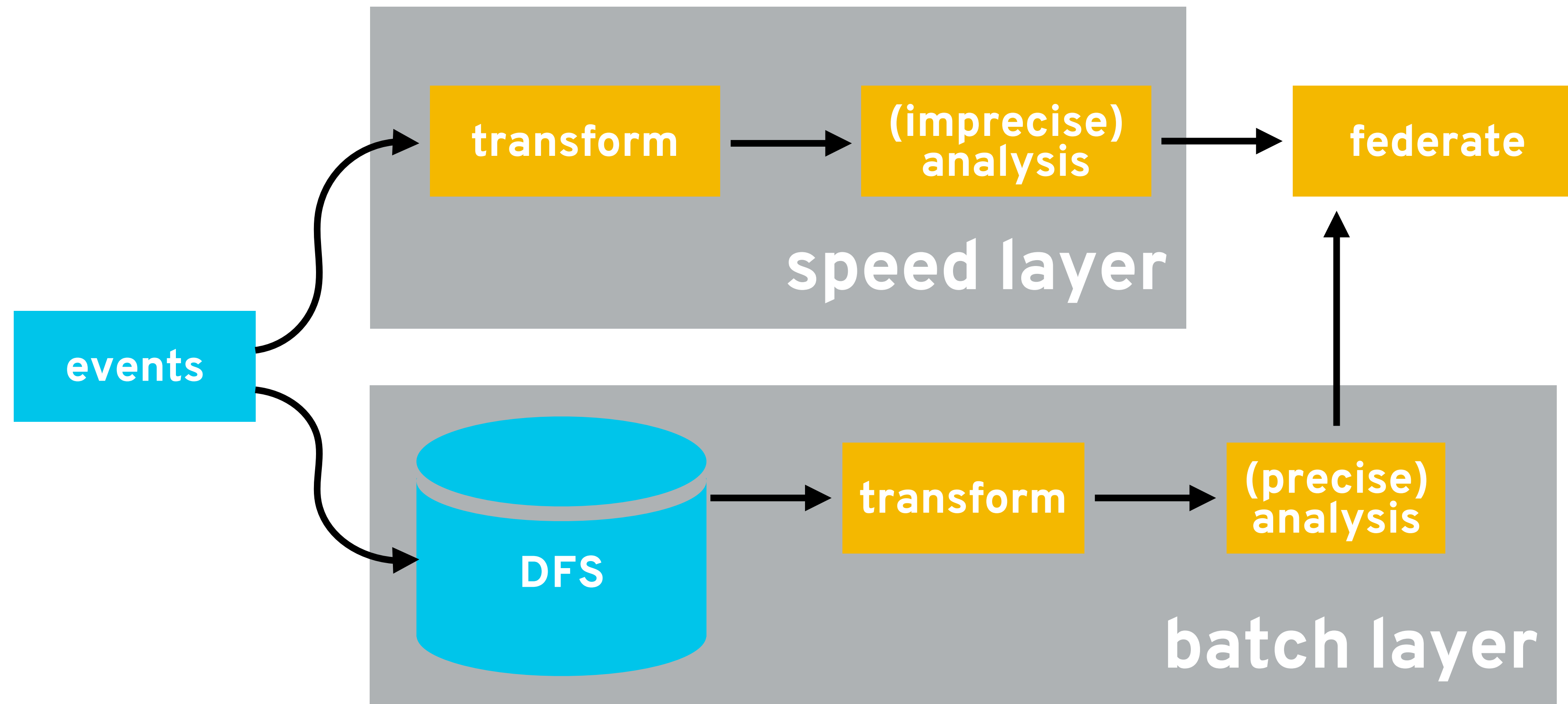
# THE LAMBDA ARCHITECTURE



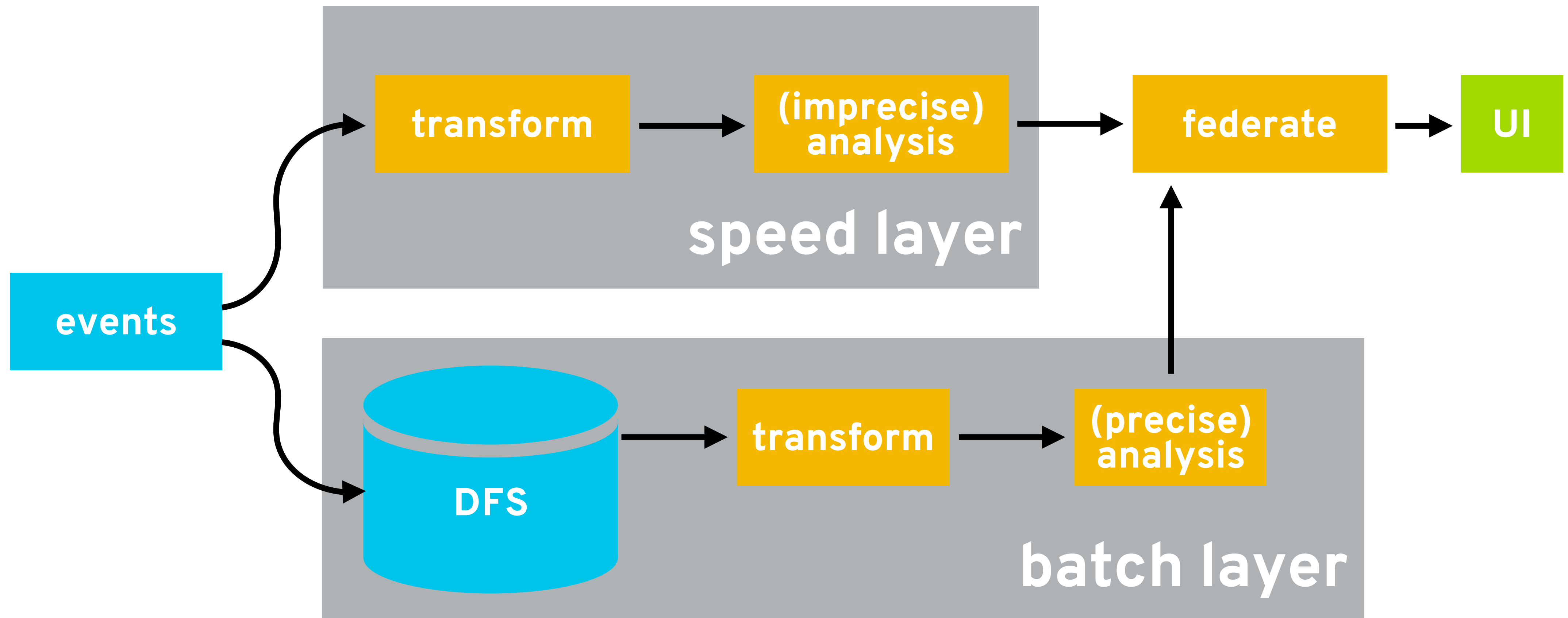
# THE LAMBDA ARCHITECTURE



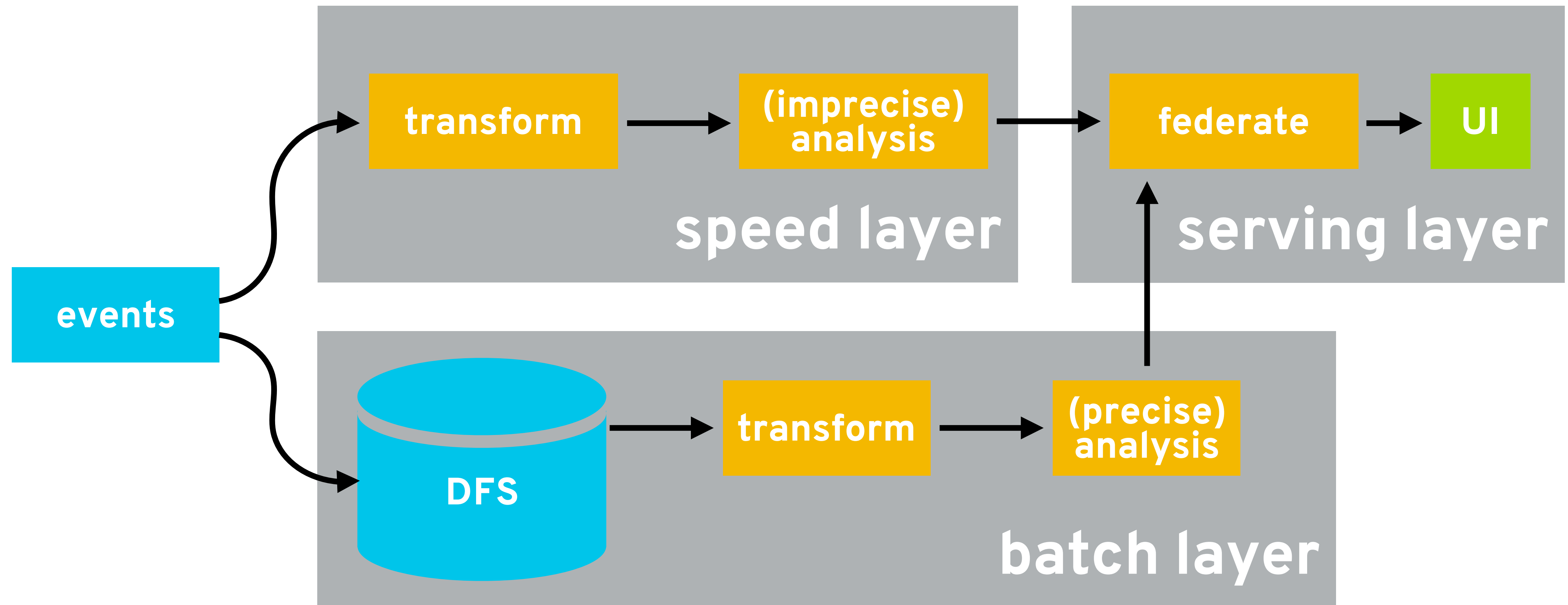
# THE LAMBDA ARCHITECTURE



# THE LAMBDA ARCHITECTURE



# THE LAMBDA ARCHITECTURE



# THE KAPPA ARCHITECTURE

events

# THE KAPPA ARCHITECTURE

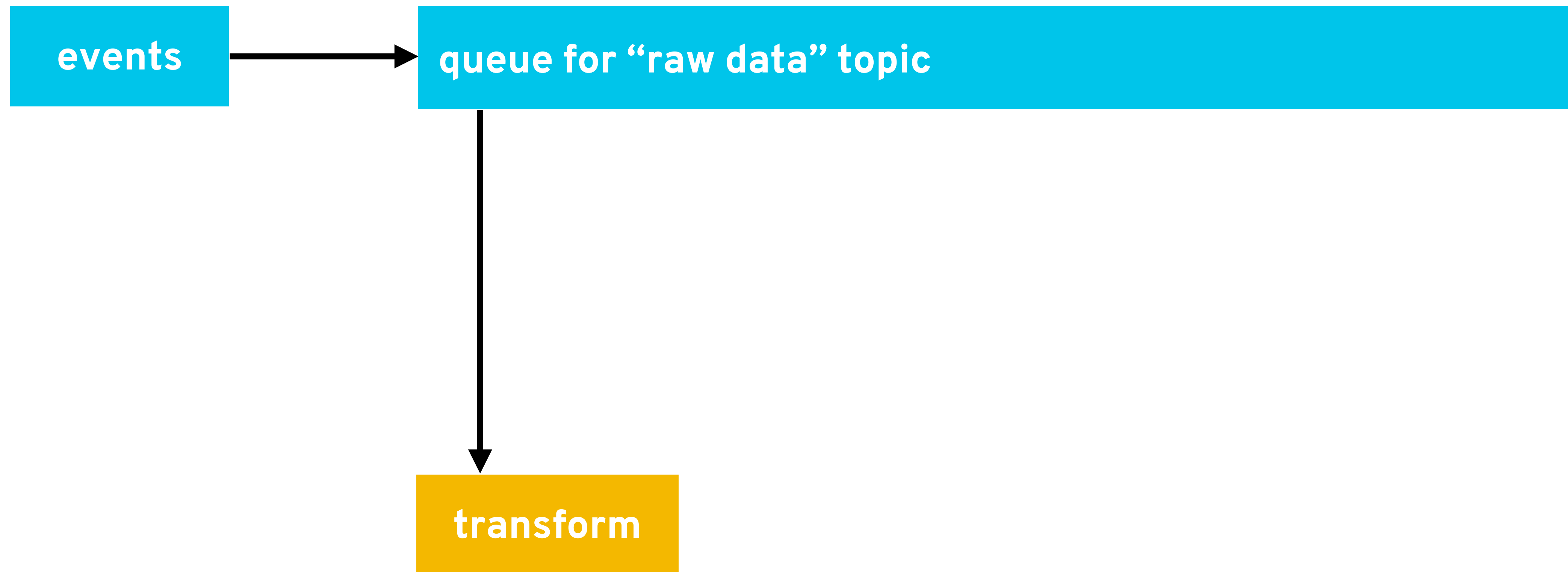




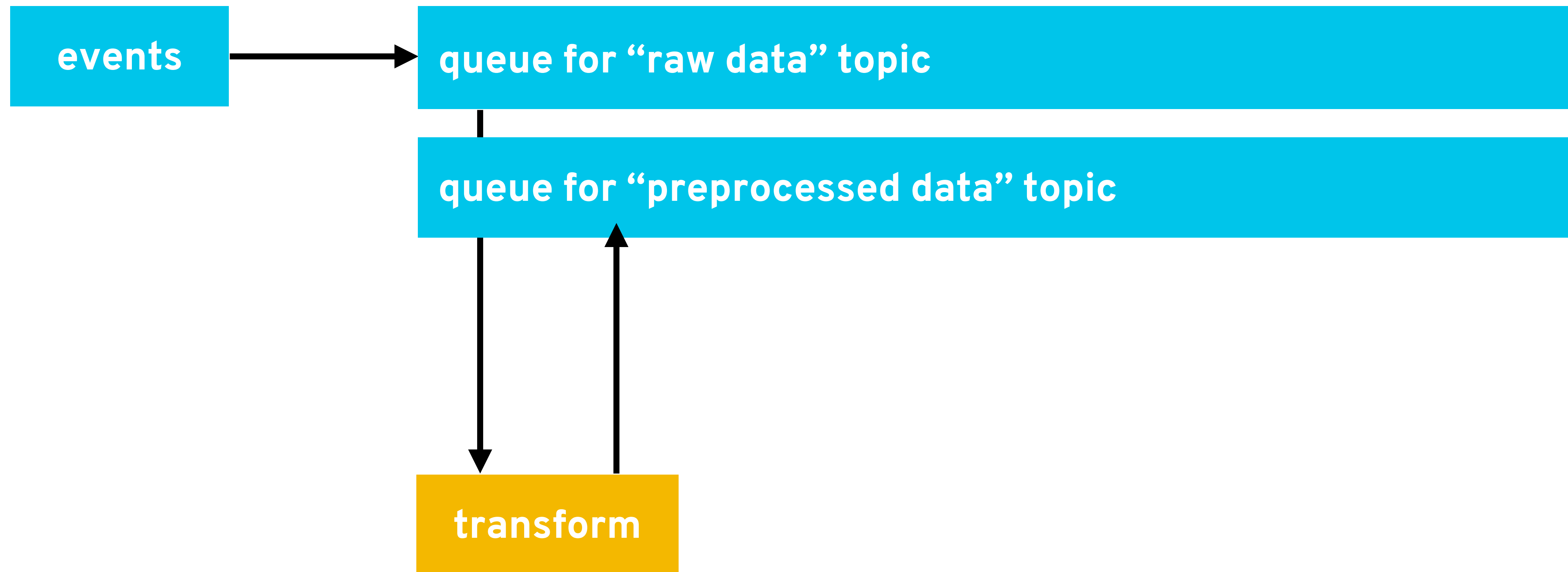
# THE KAPPA ARCHITECTURE



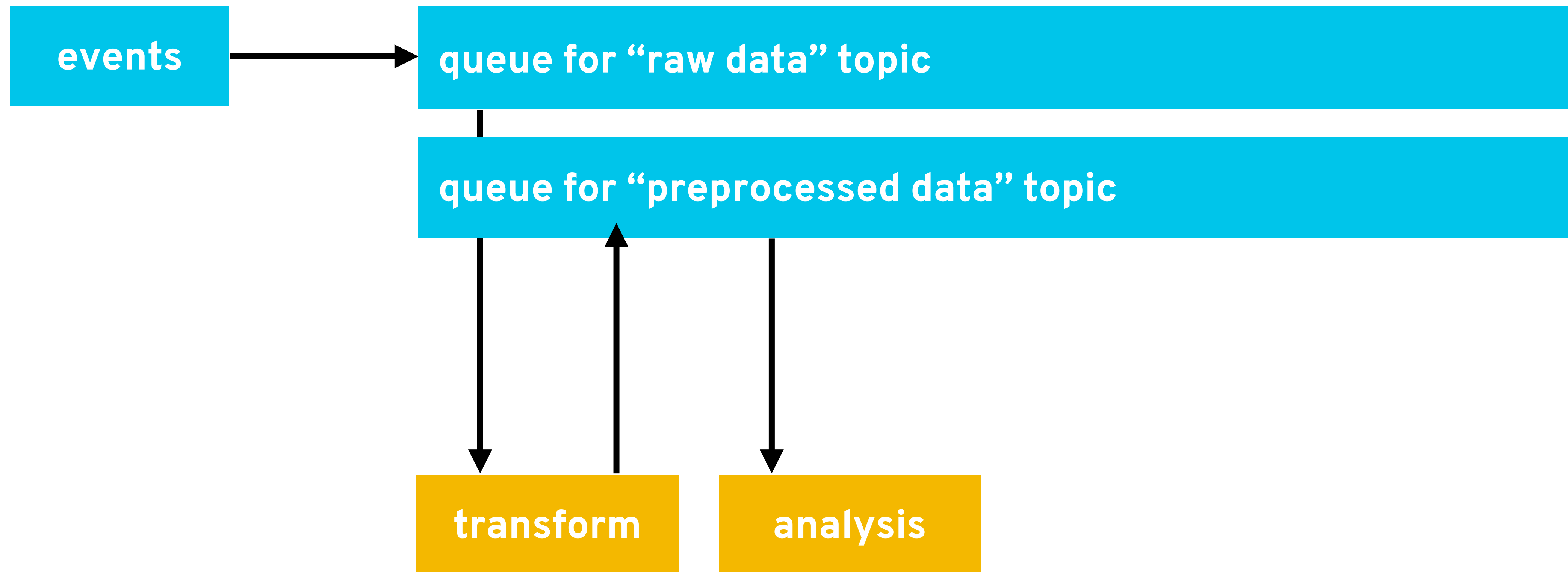
# THE KAPPA ARCHITECTURE



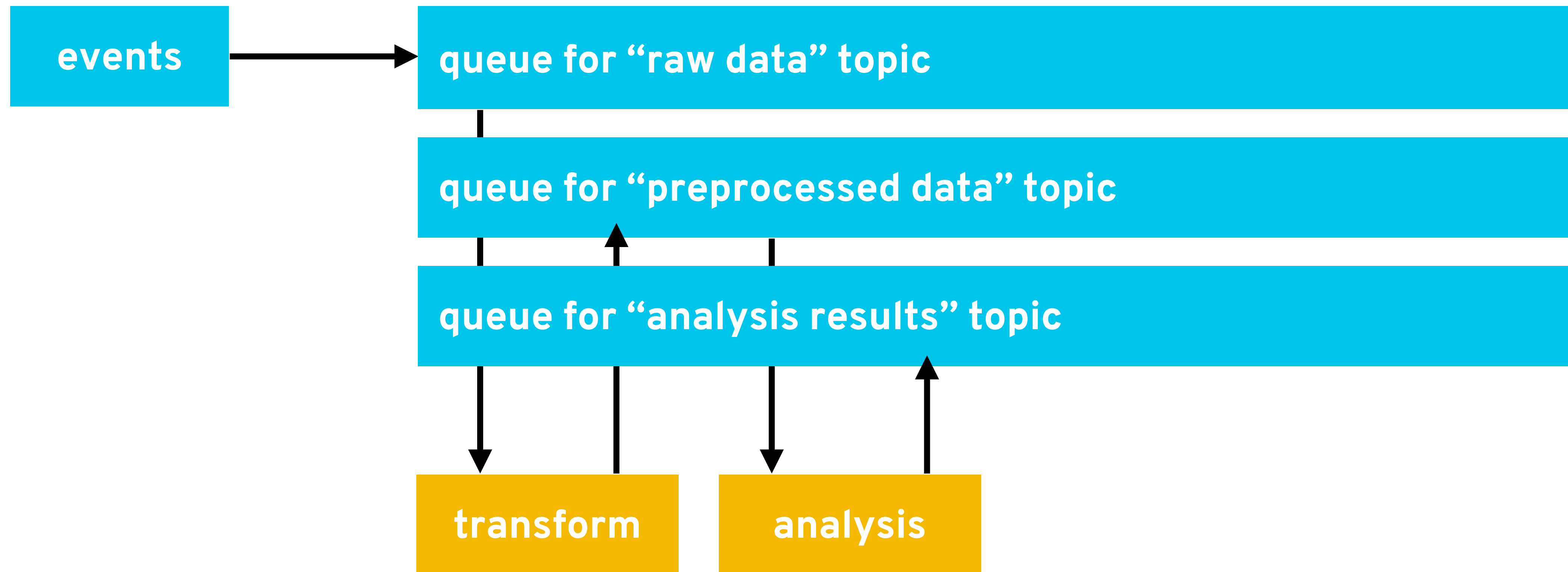
# THE KAPPA ARCHITECTURE



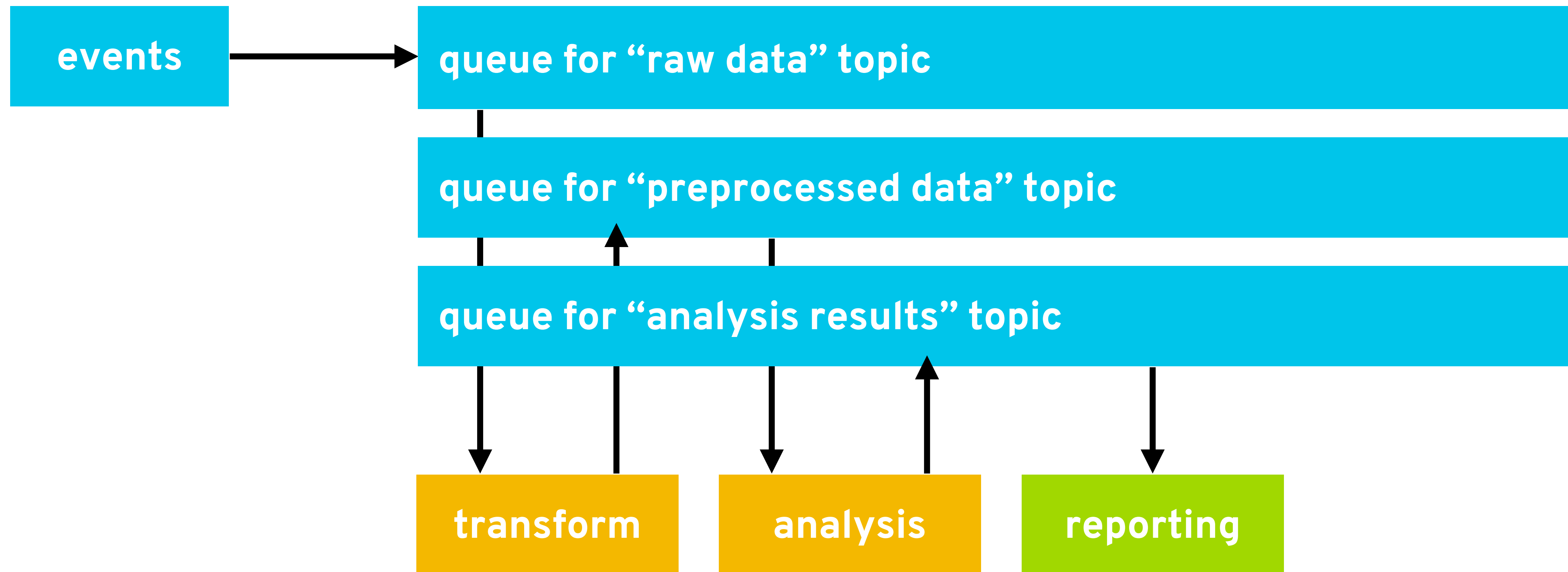
# THE KAPPA ARCHITECTURE



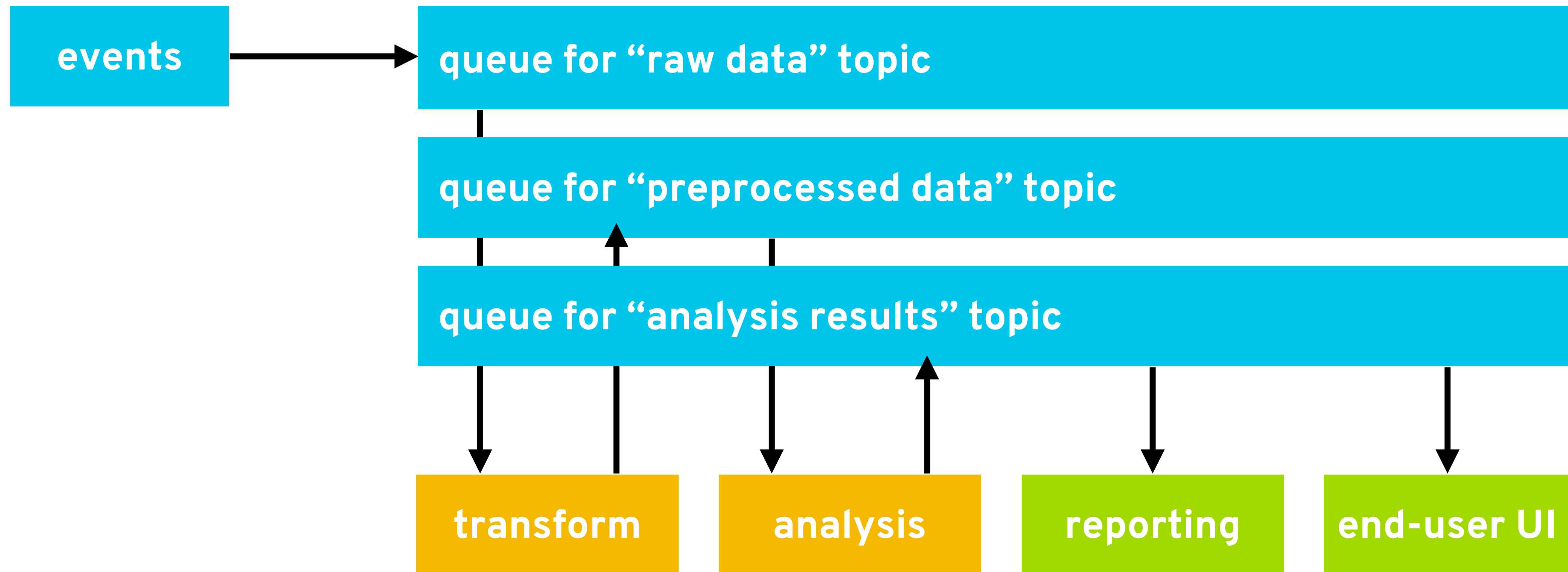
# THE KAPPA ARCHITECTURE



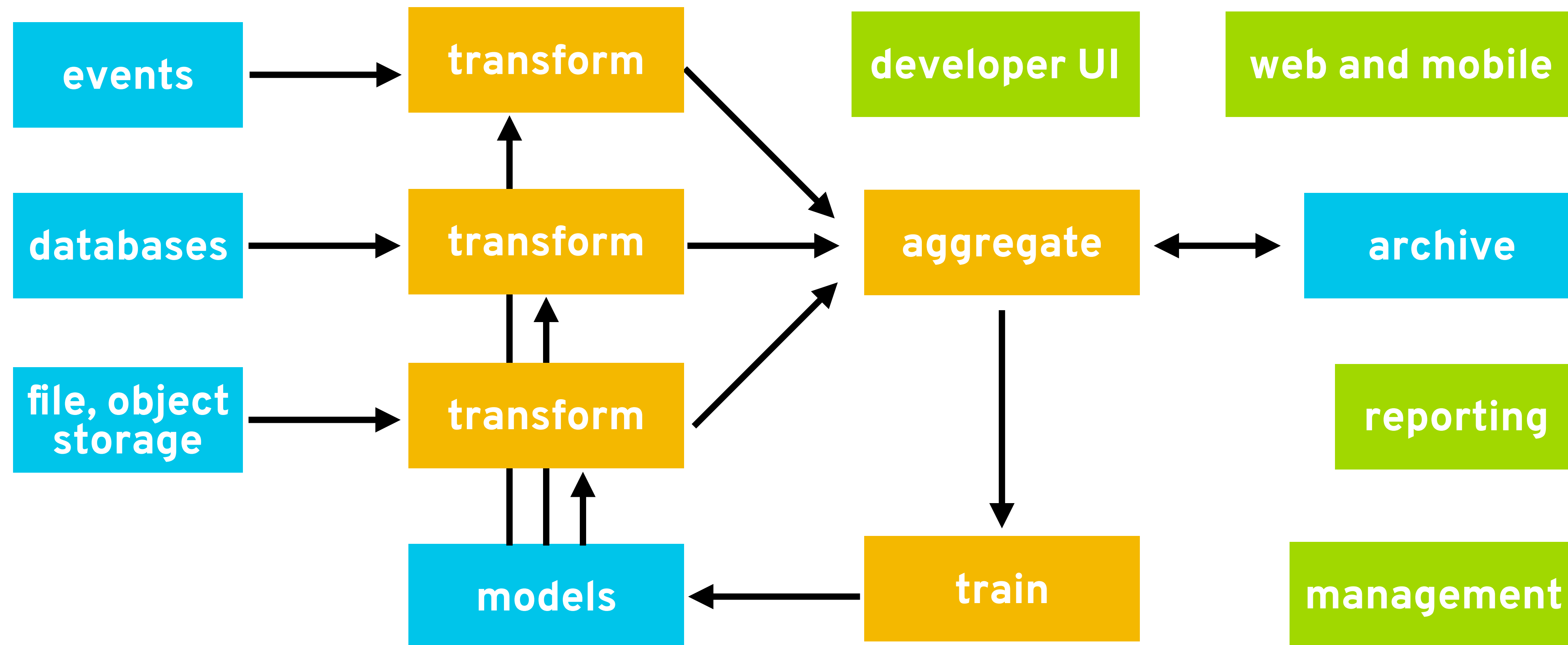
# THE KAPPA ARCHITECTURE



# THE KAPPA ARCHITECTURE

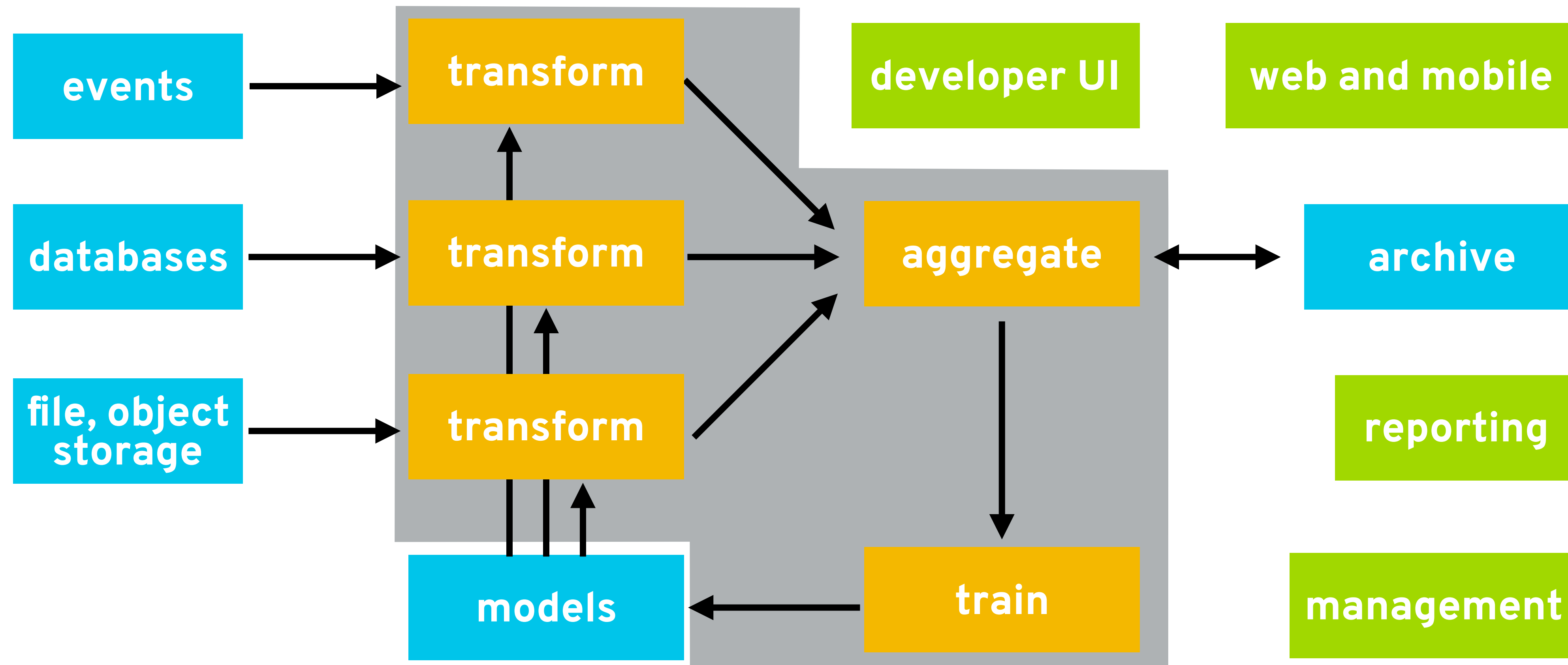


# DATA FEDERATION IN THE COMPUTE LAYER

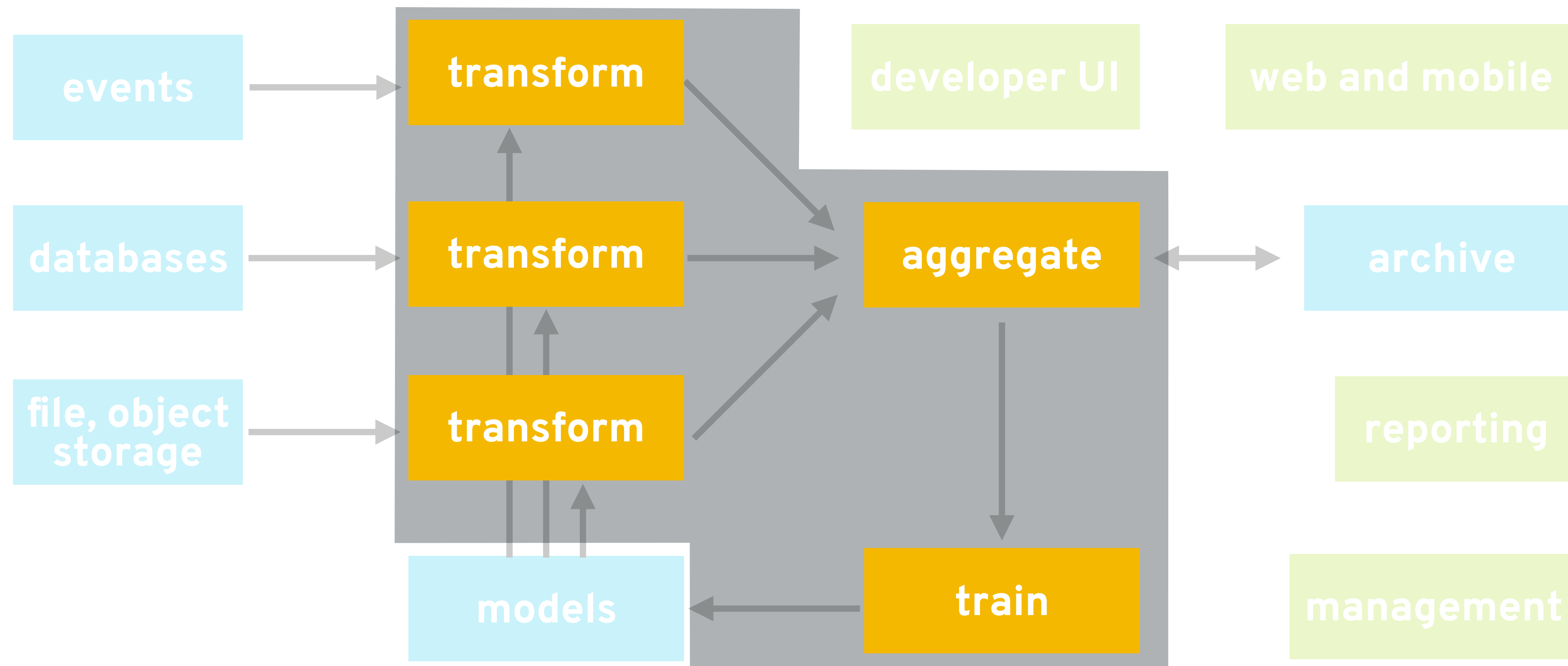




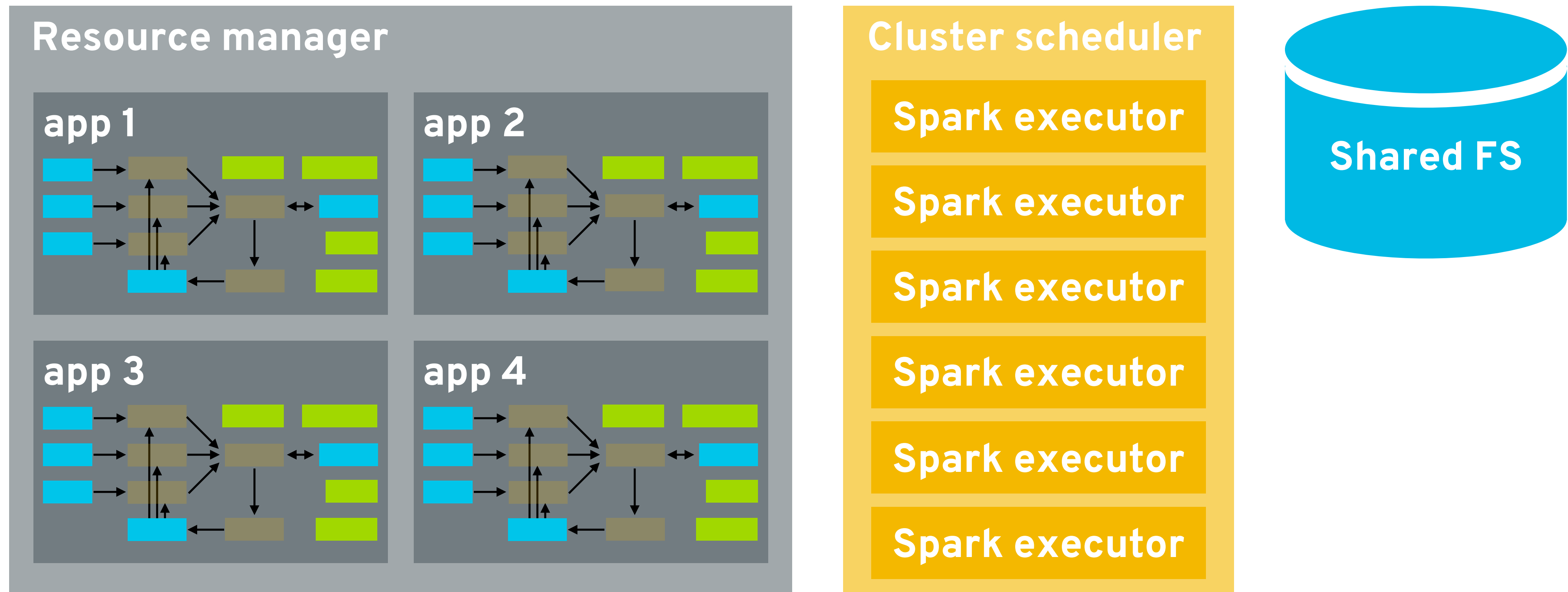
# DATA FEDERATION IN THE COMPUTE LAYER



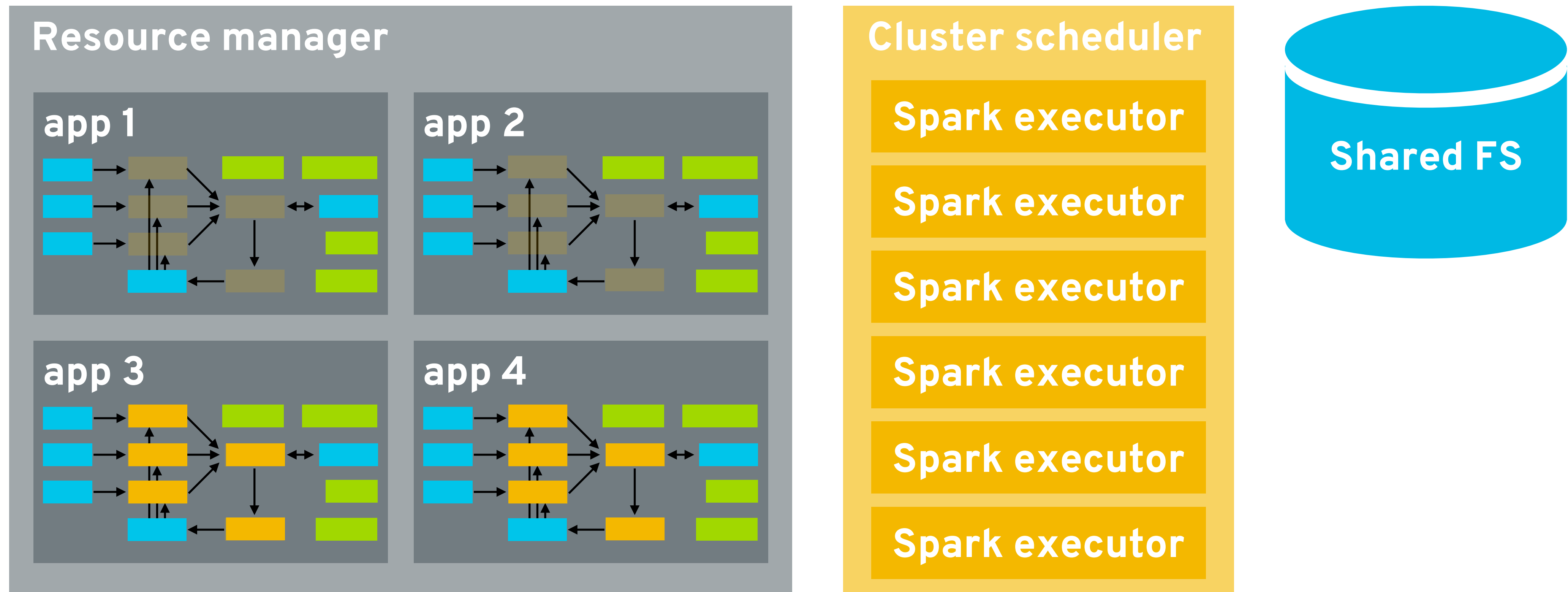
# DATA FEDERATION IN THE COMPUTE LAYER



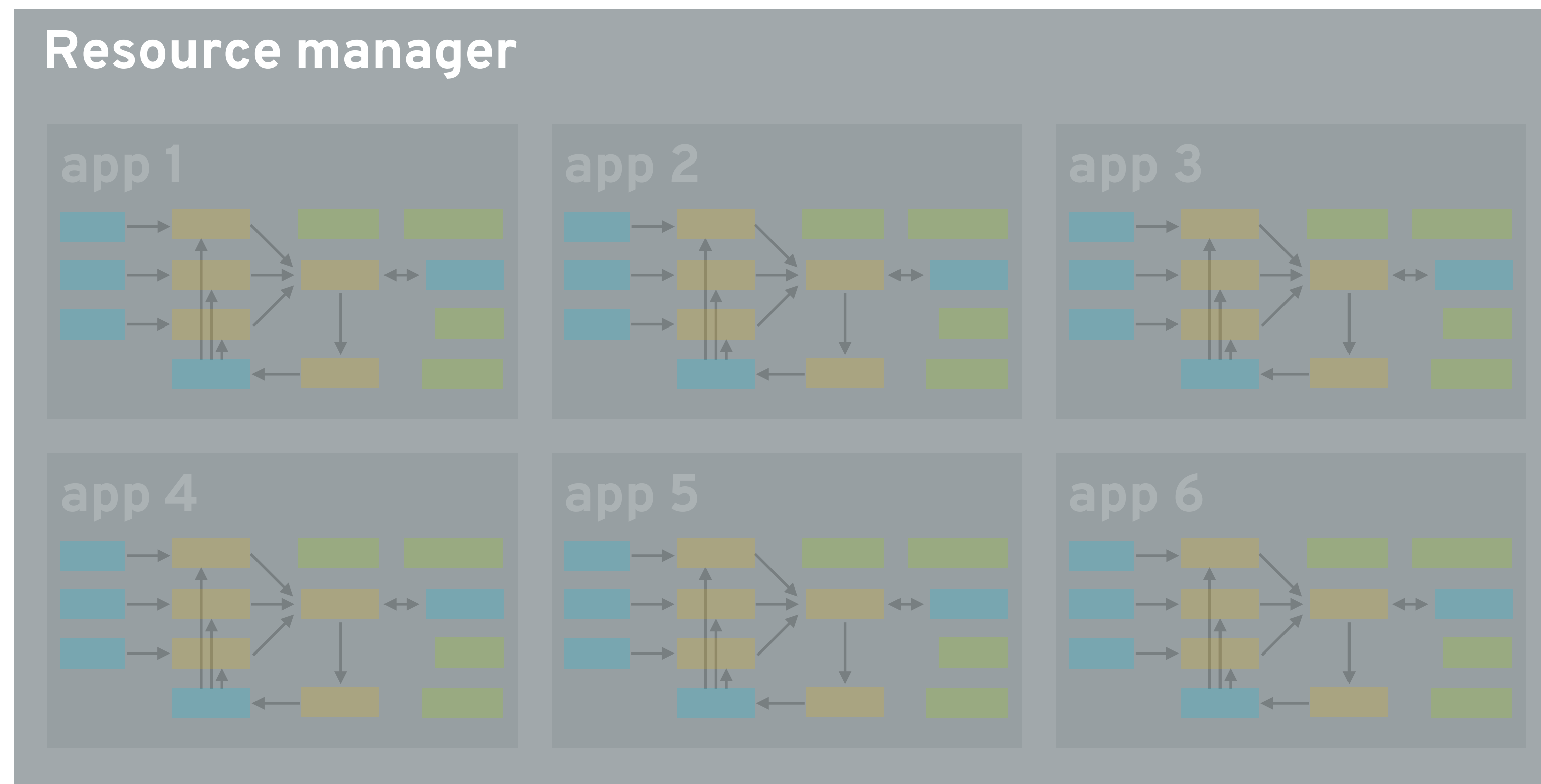
# SIDEBAR: THE MONOLITHIC SPARK ANTIPATTERN



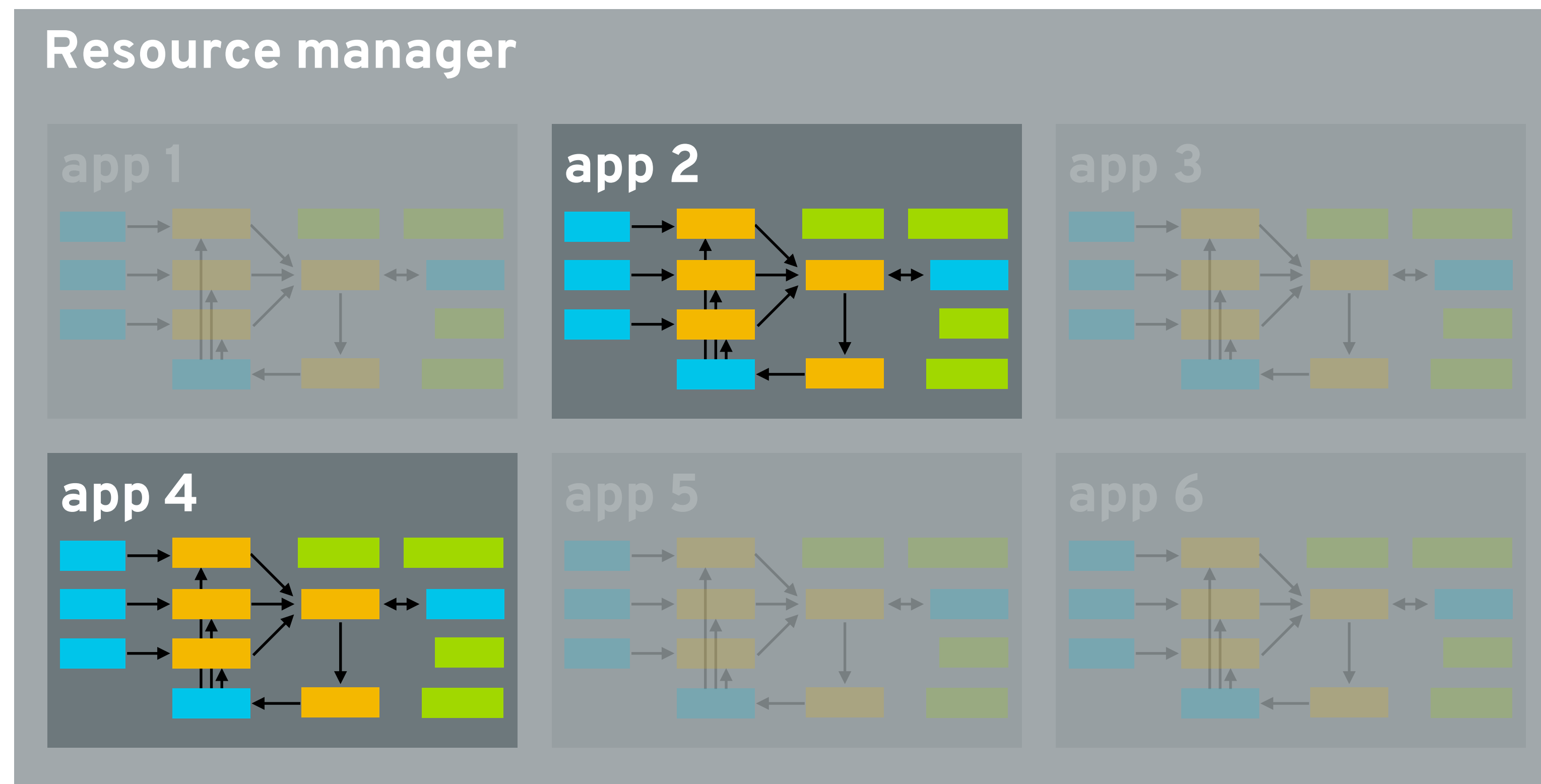
# SIDEBAR: THE MONOLITHIC SPARK ANTIPATTERN



# ONE CLUSTER PER APPLICATION

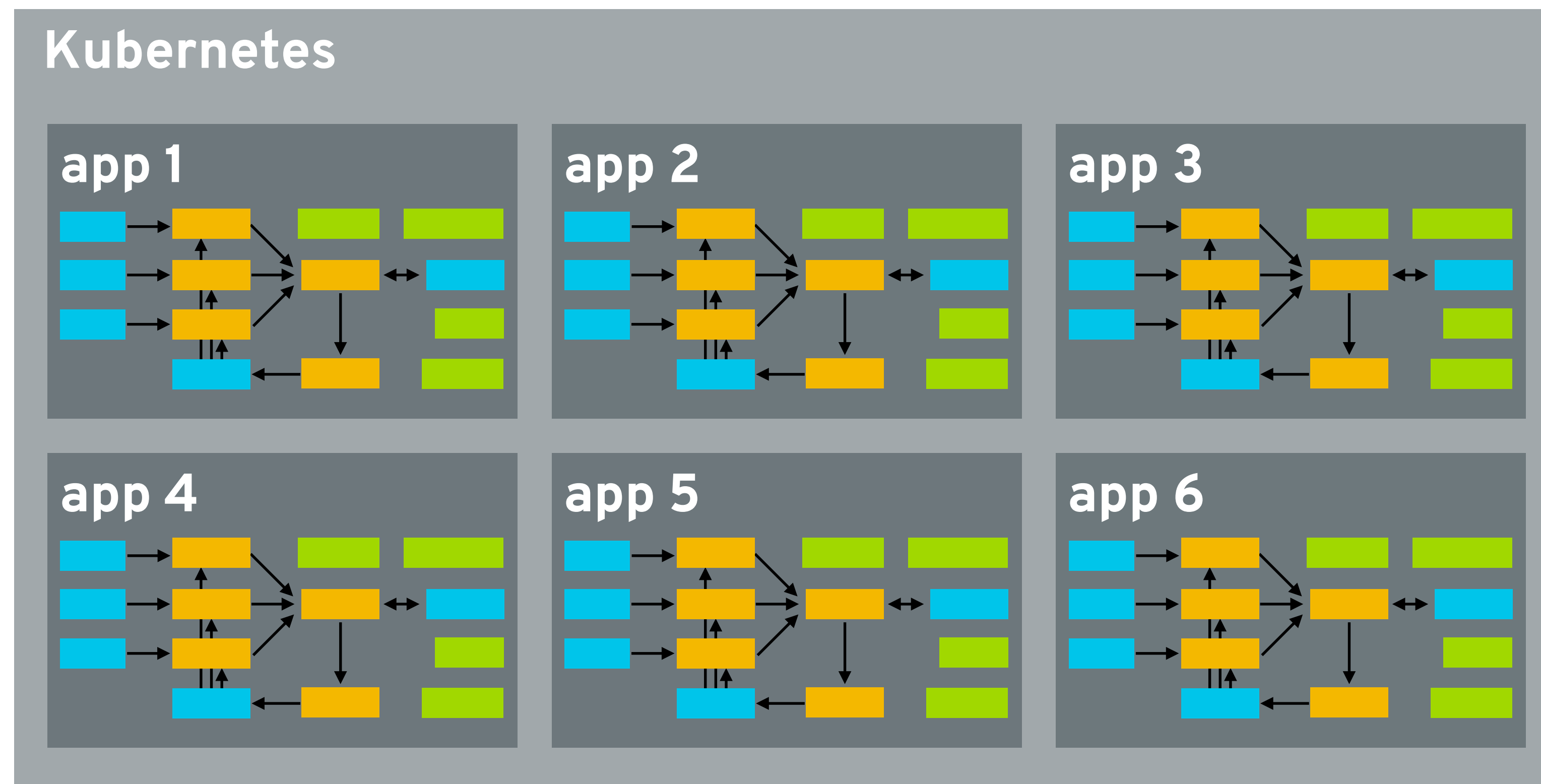


# ONE CLUSTER PER APPLICATION



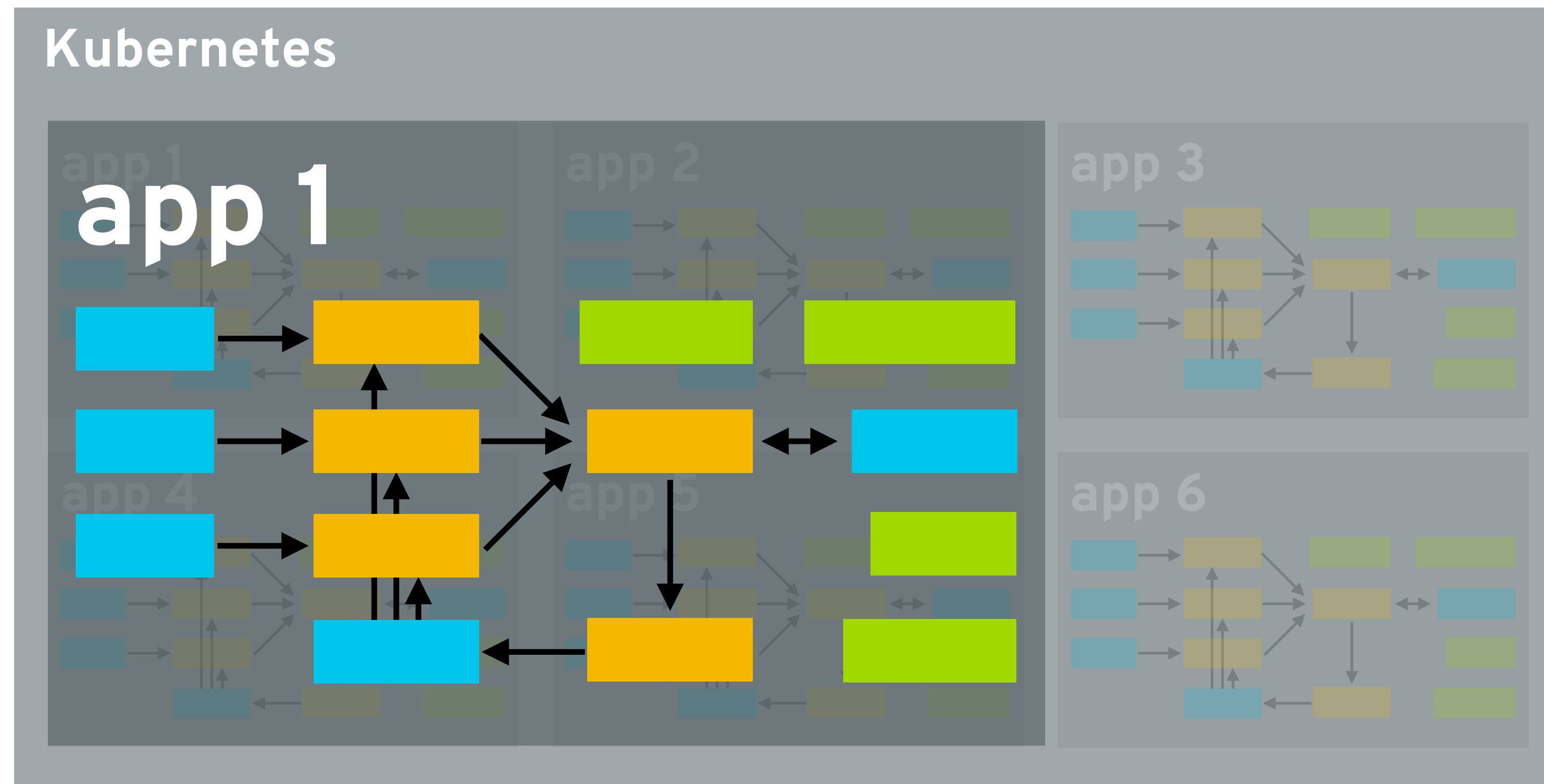
# PRACTICALITIES AND POTENTIAL PITFALLS

# SCHEDULING

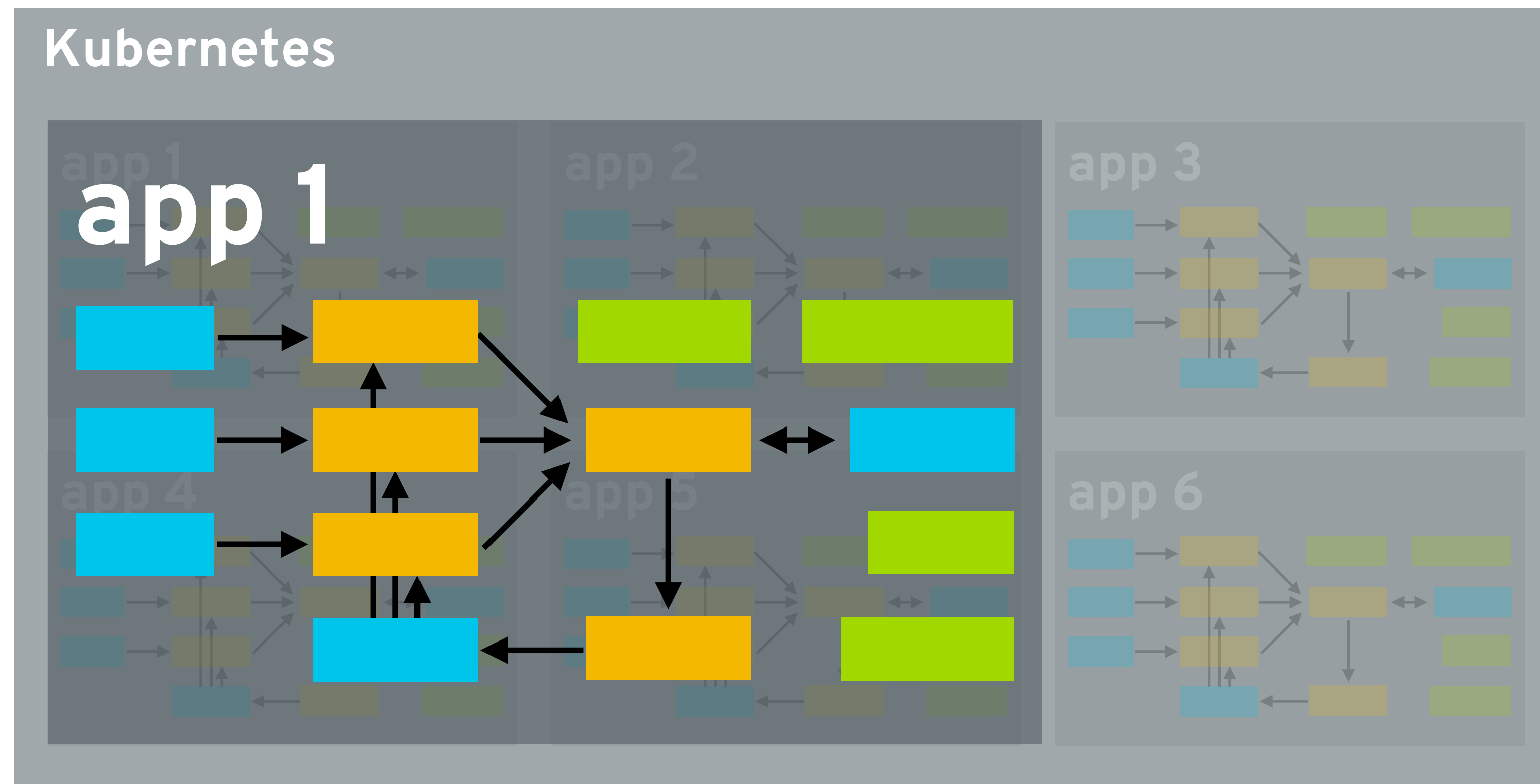




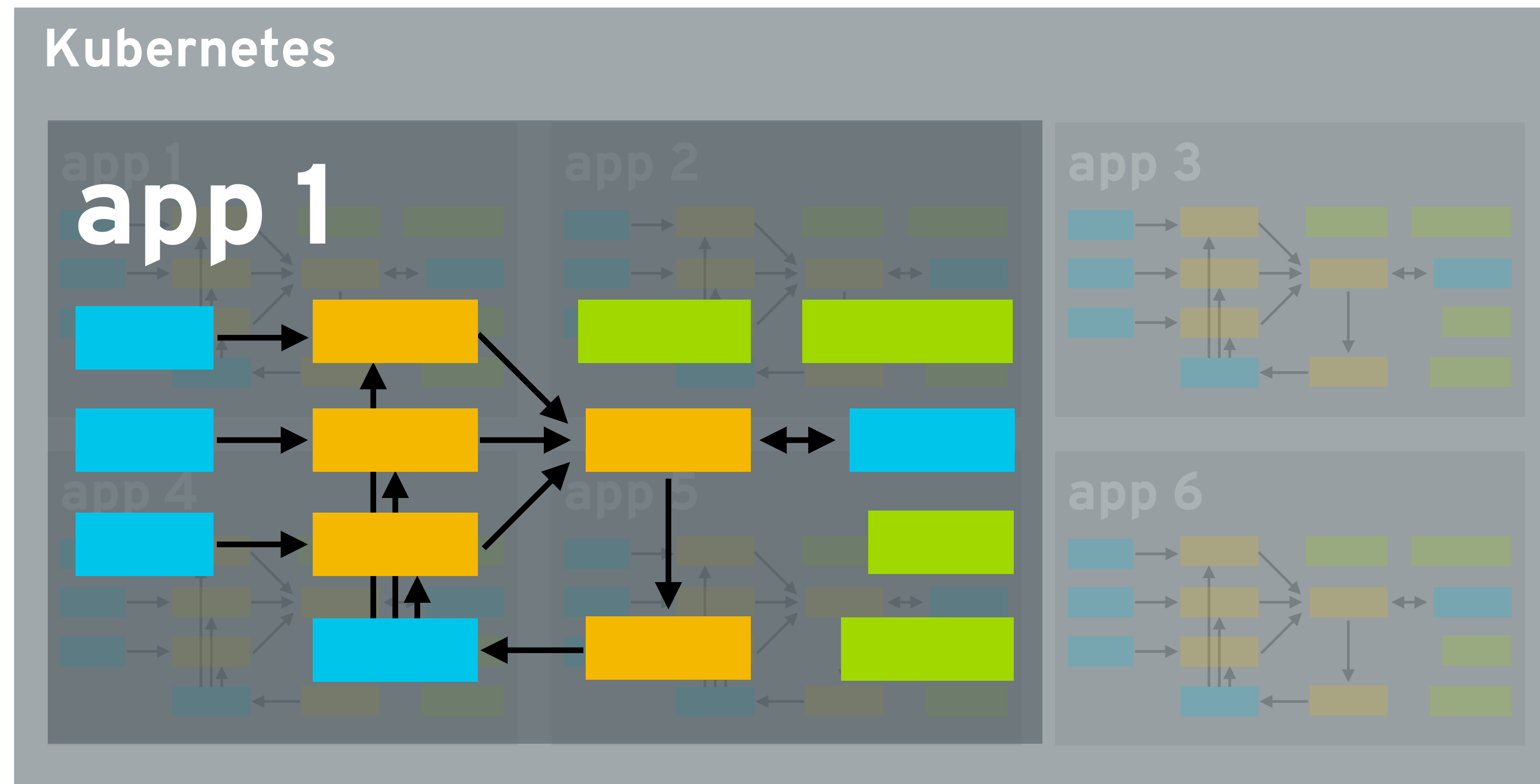
# SCHEDULING



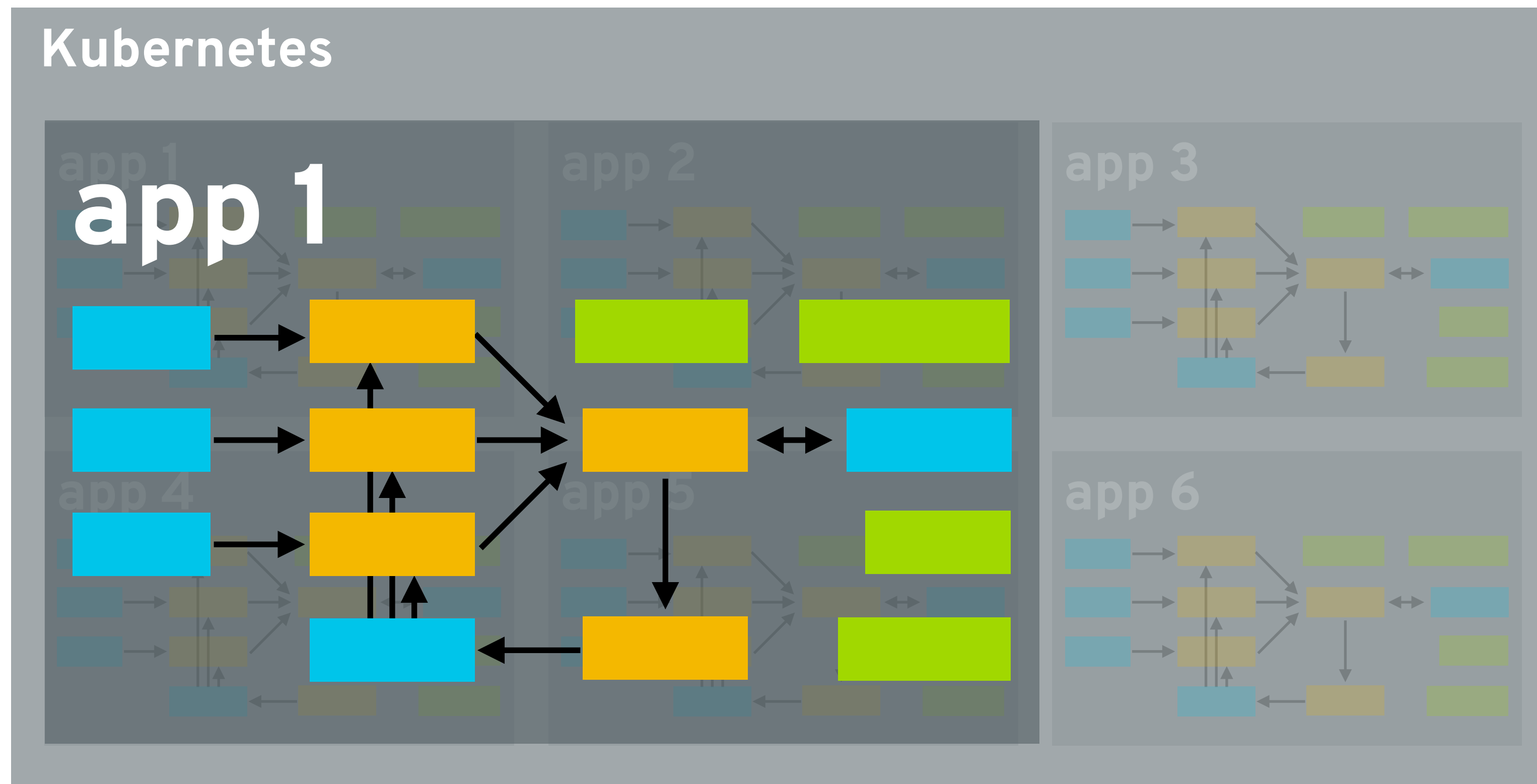
# SCHEDULING



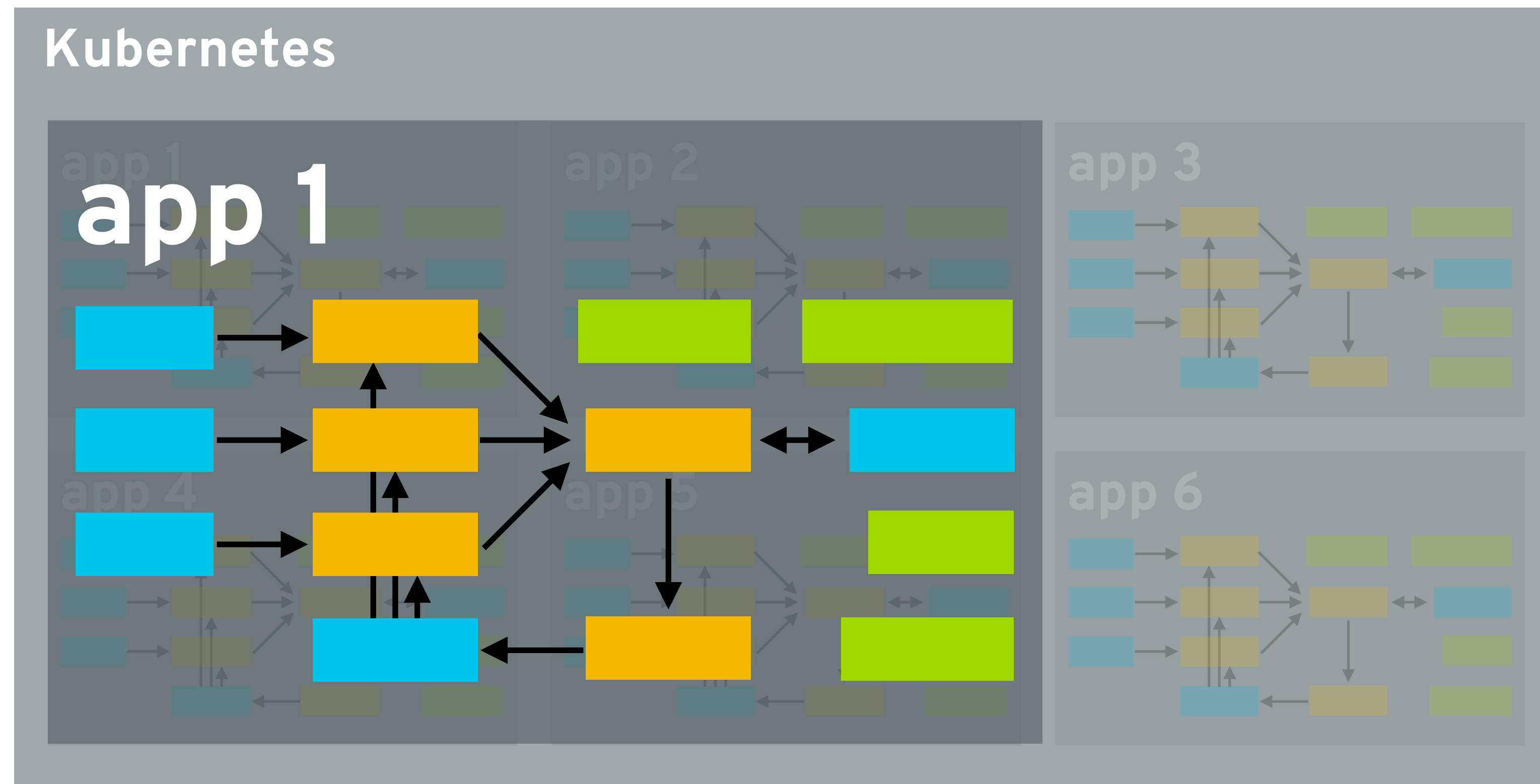
# SCHEDULING



# SCHEDULING



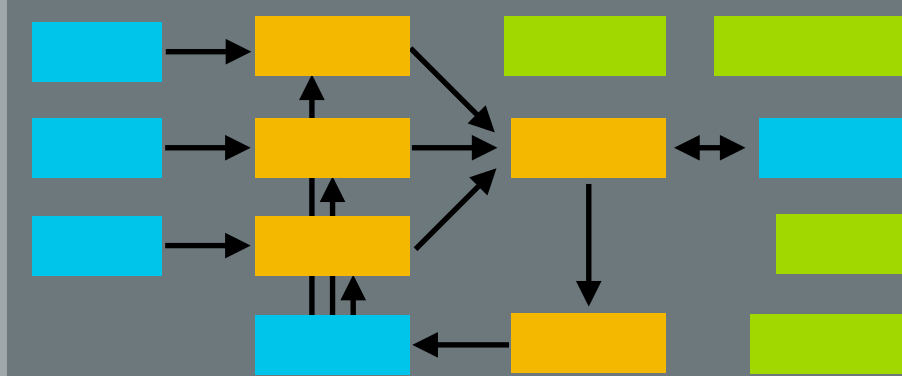
# SCHEDULING



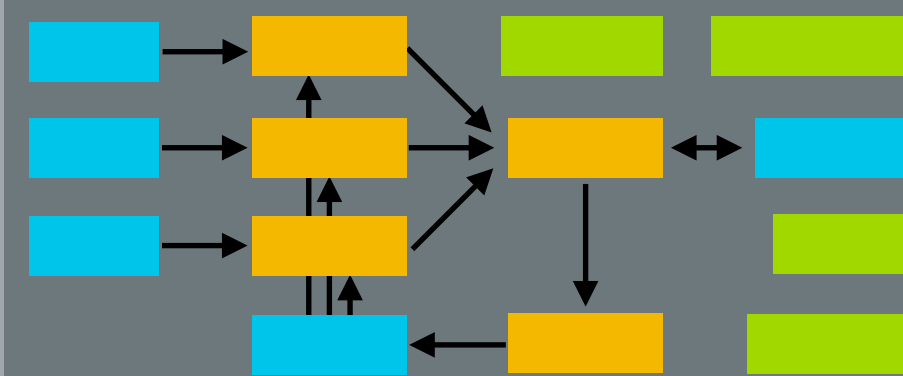
# STORAGE

## Kubernetes

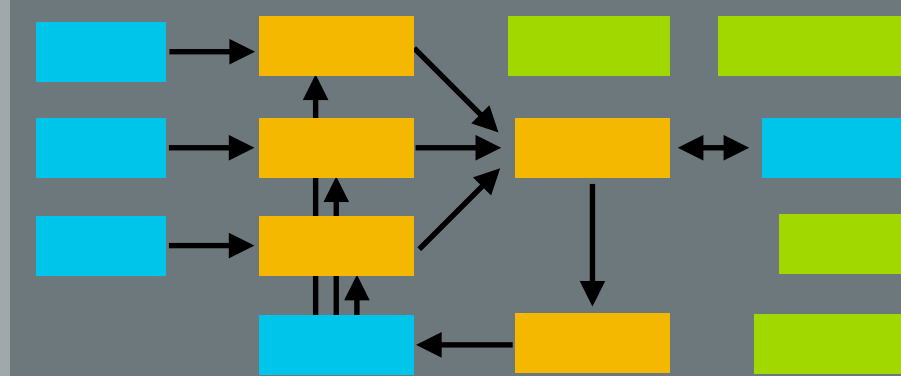
app 1



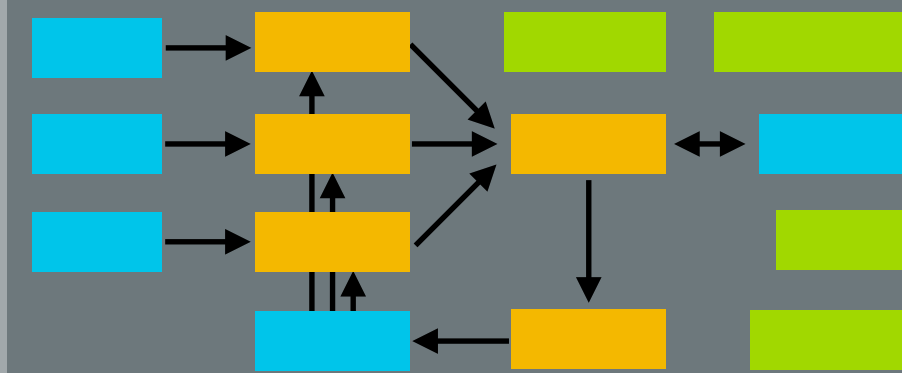
app 2



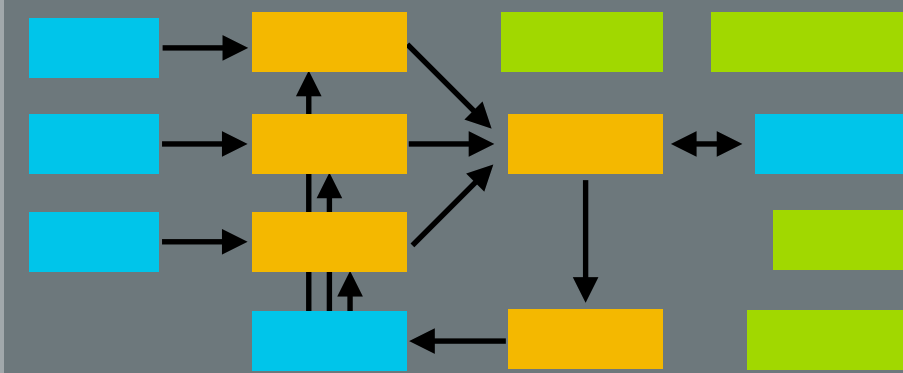
app 3



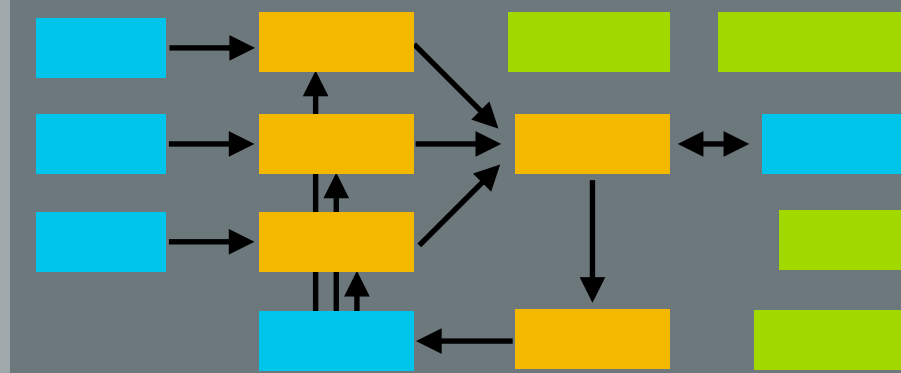
app 4



app 5



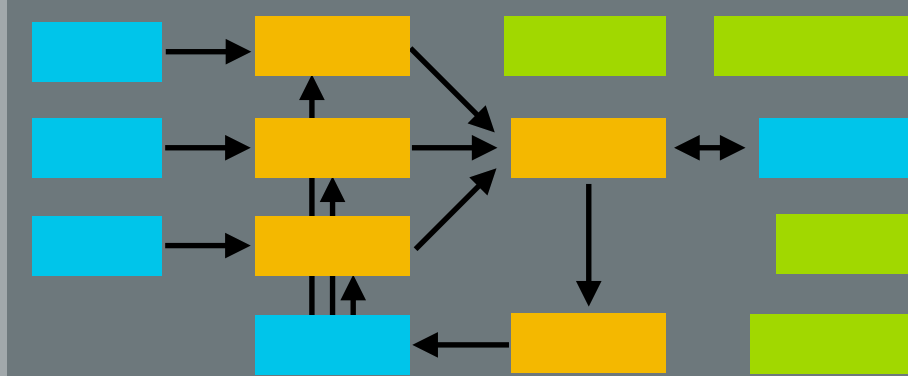
app 6



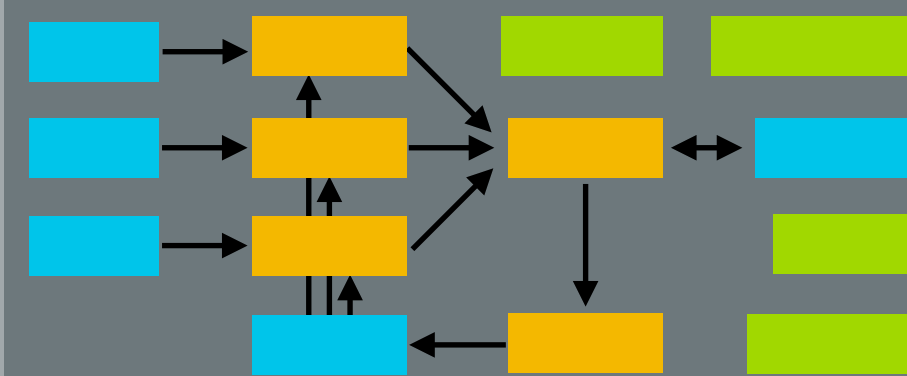
# STORAGE

## Kubernetes

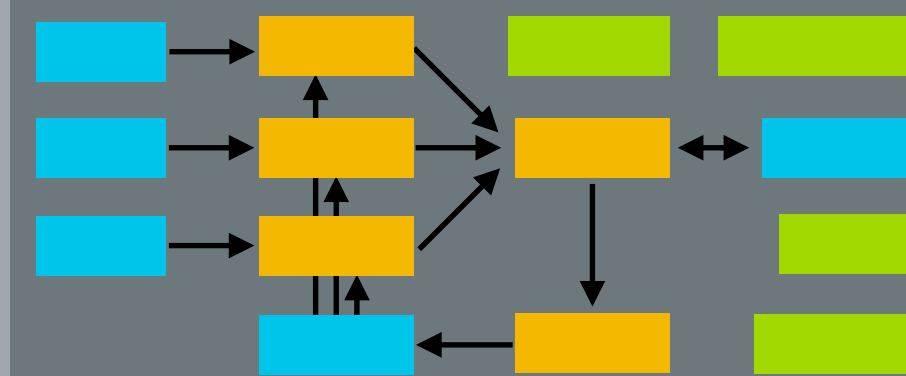
app 1



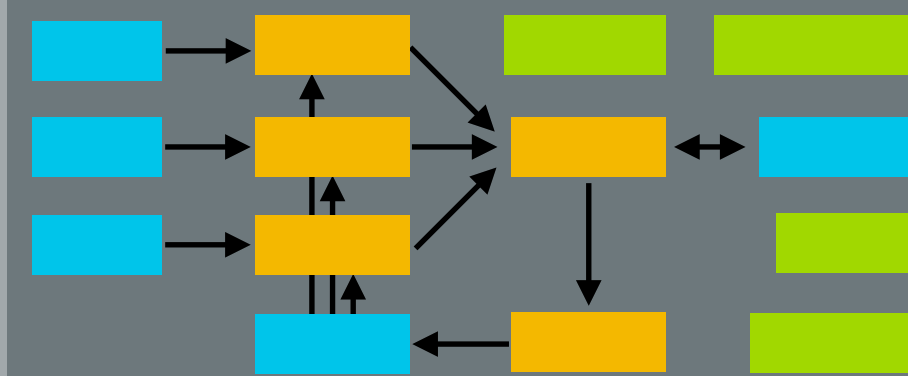
app 2



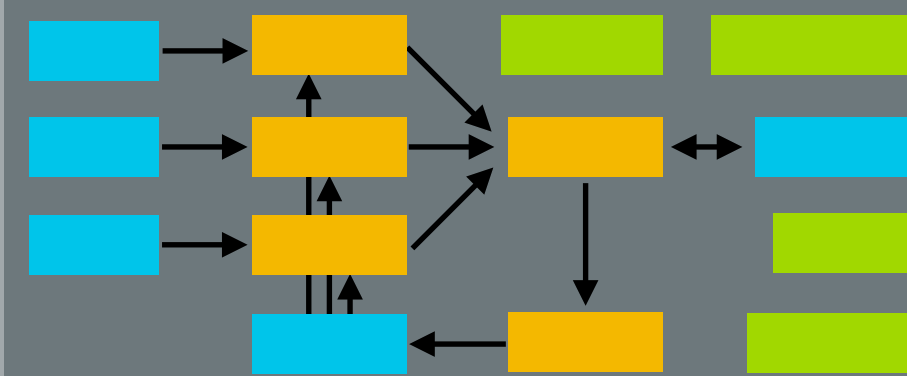
app 3



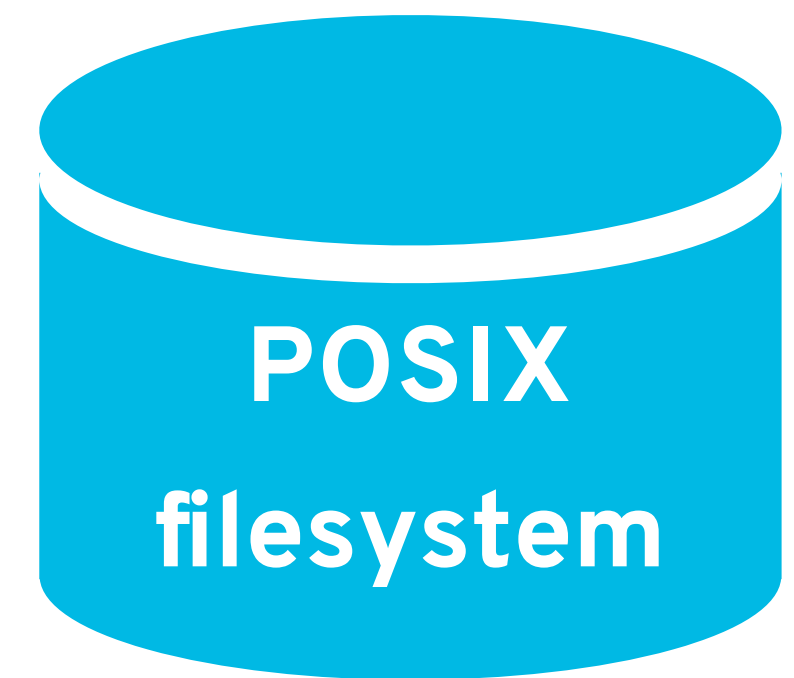
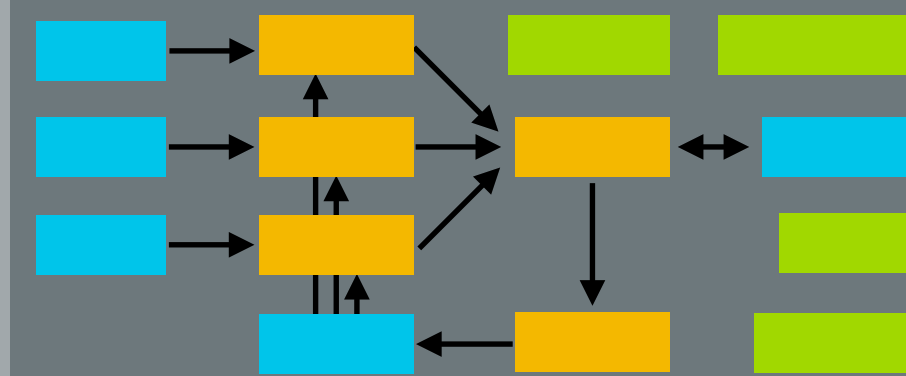
app 4



app 5



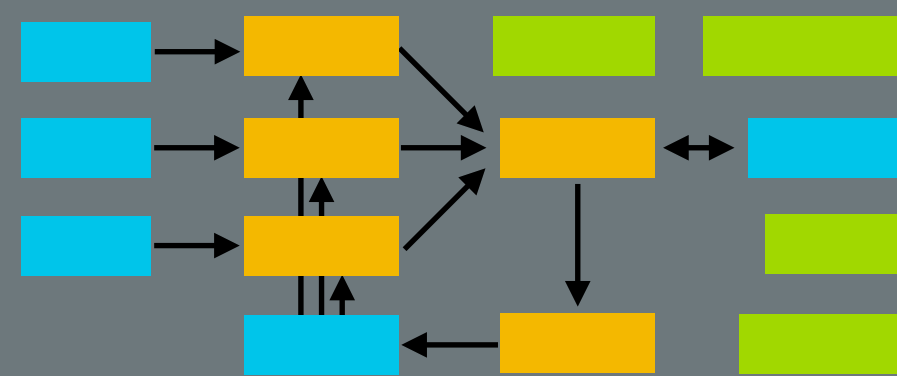
app 6



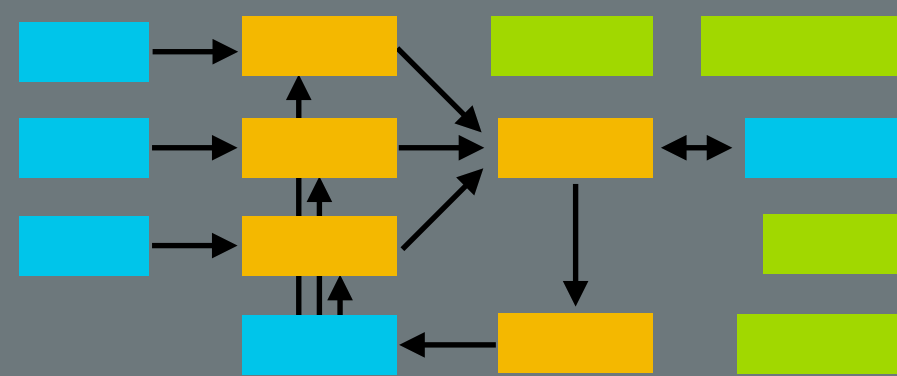
# STORAGE

## Kubernetes

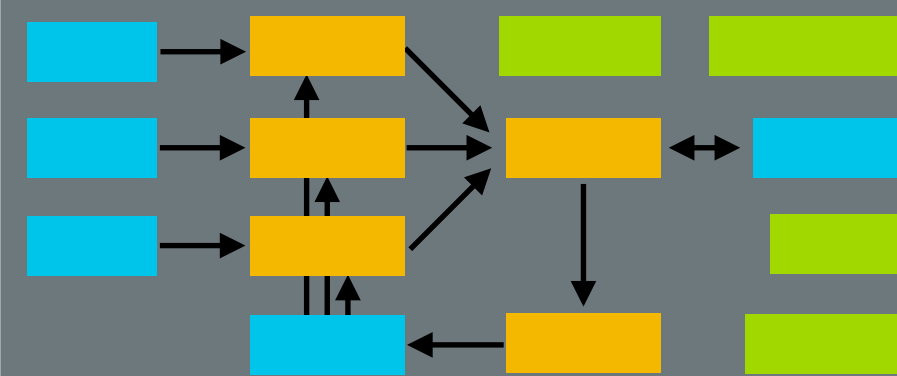
app 1



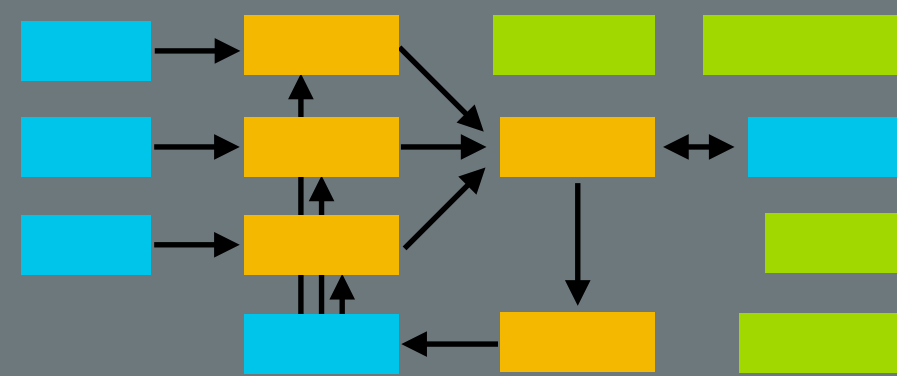
app 2



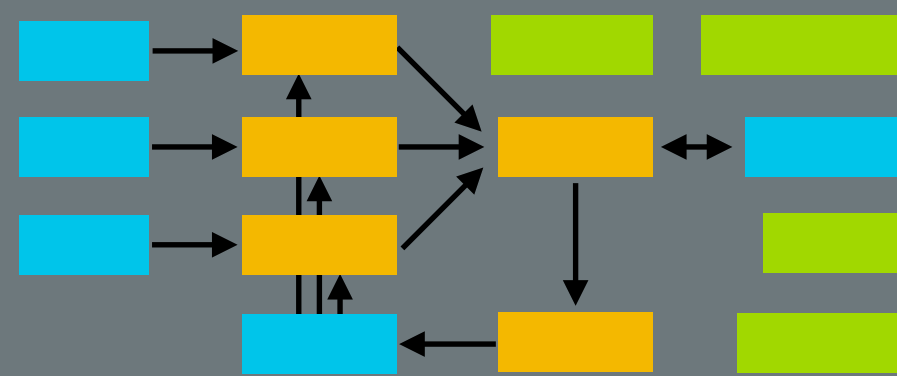
app 3



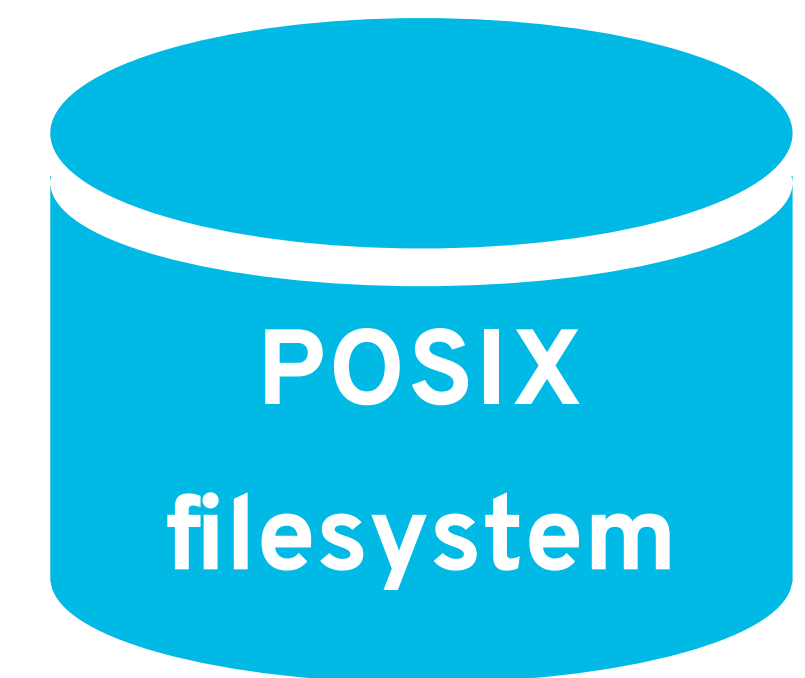
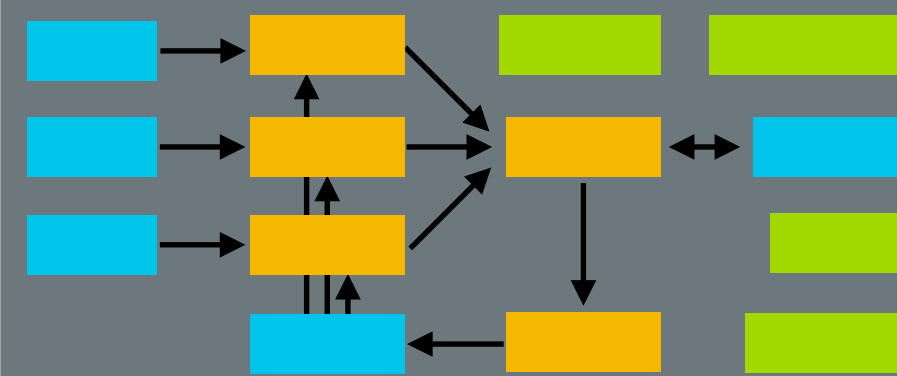
app 4



app 5



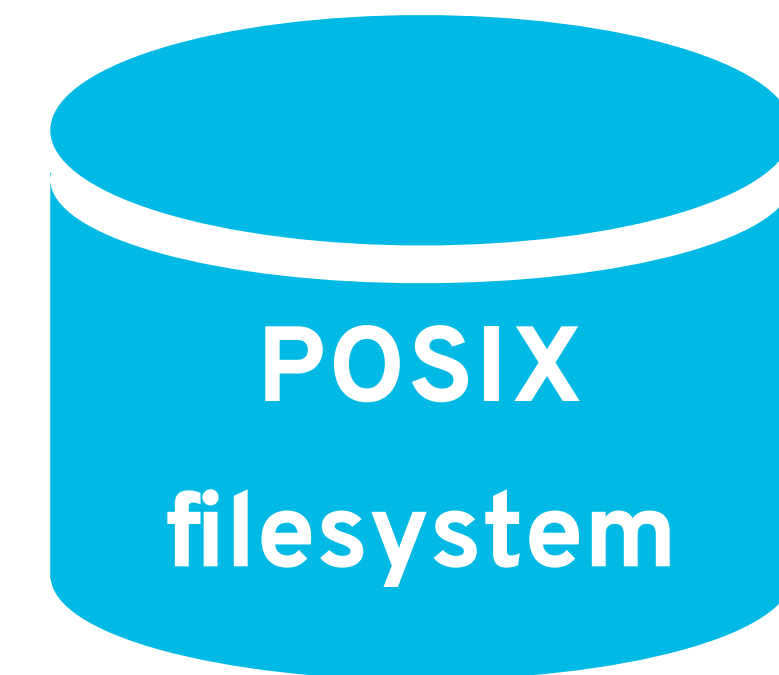
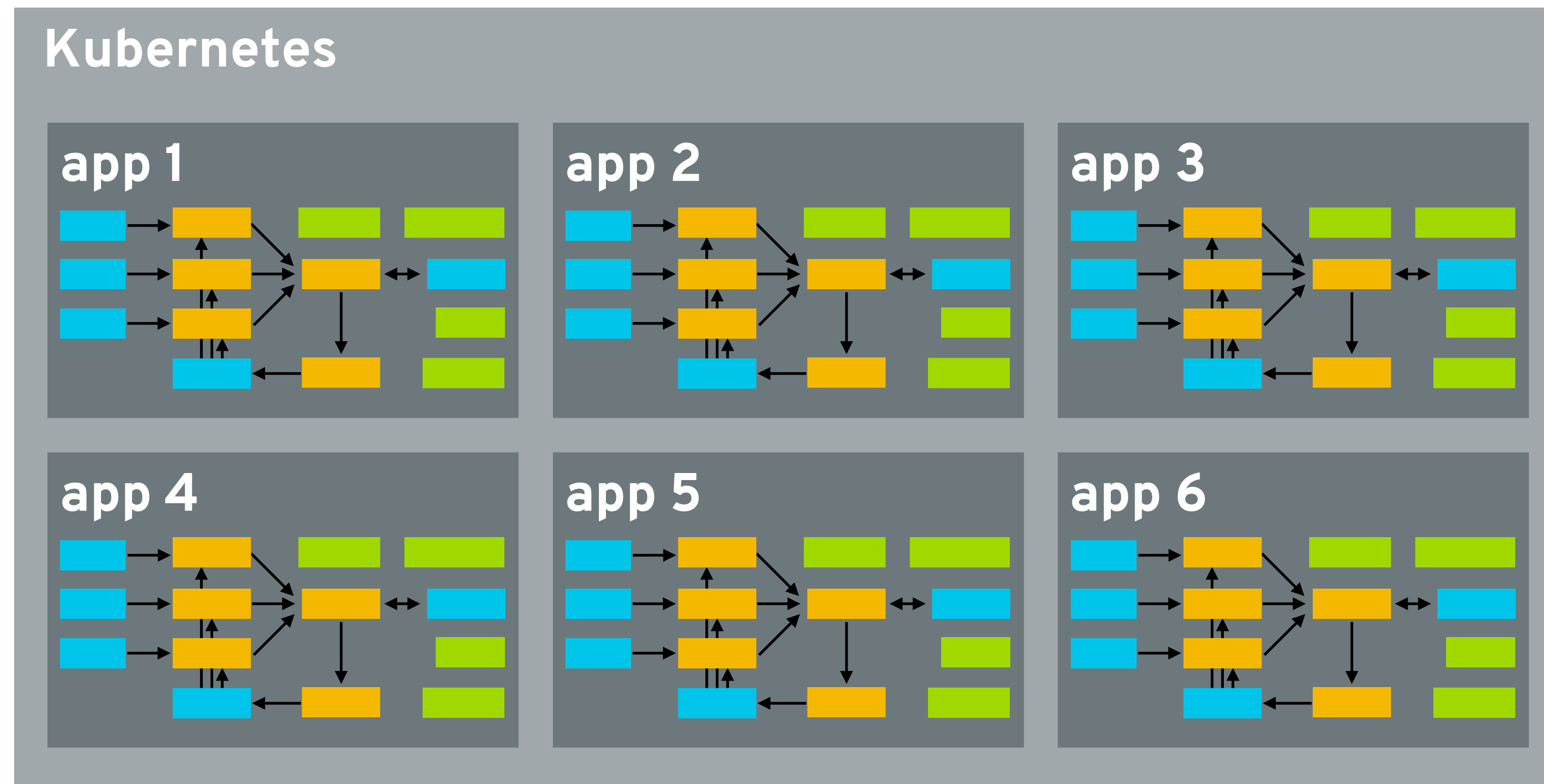
app 6



- ✓ familiar interface
- ✓ interoperability with other programs

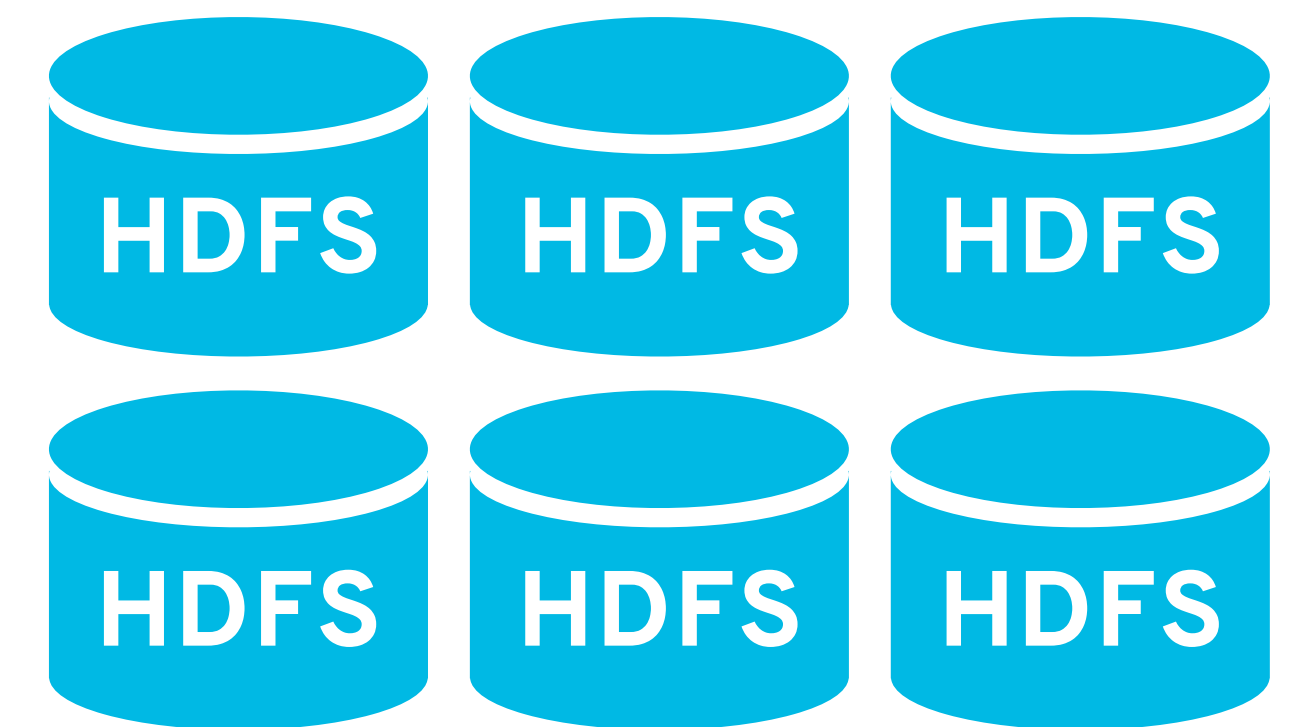
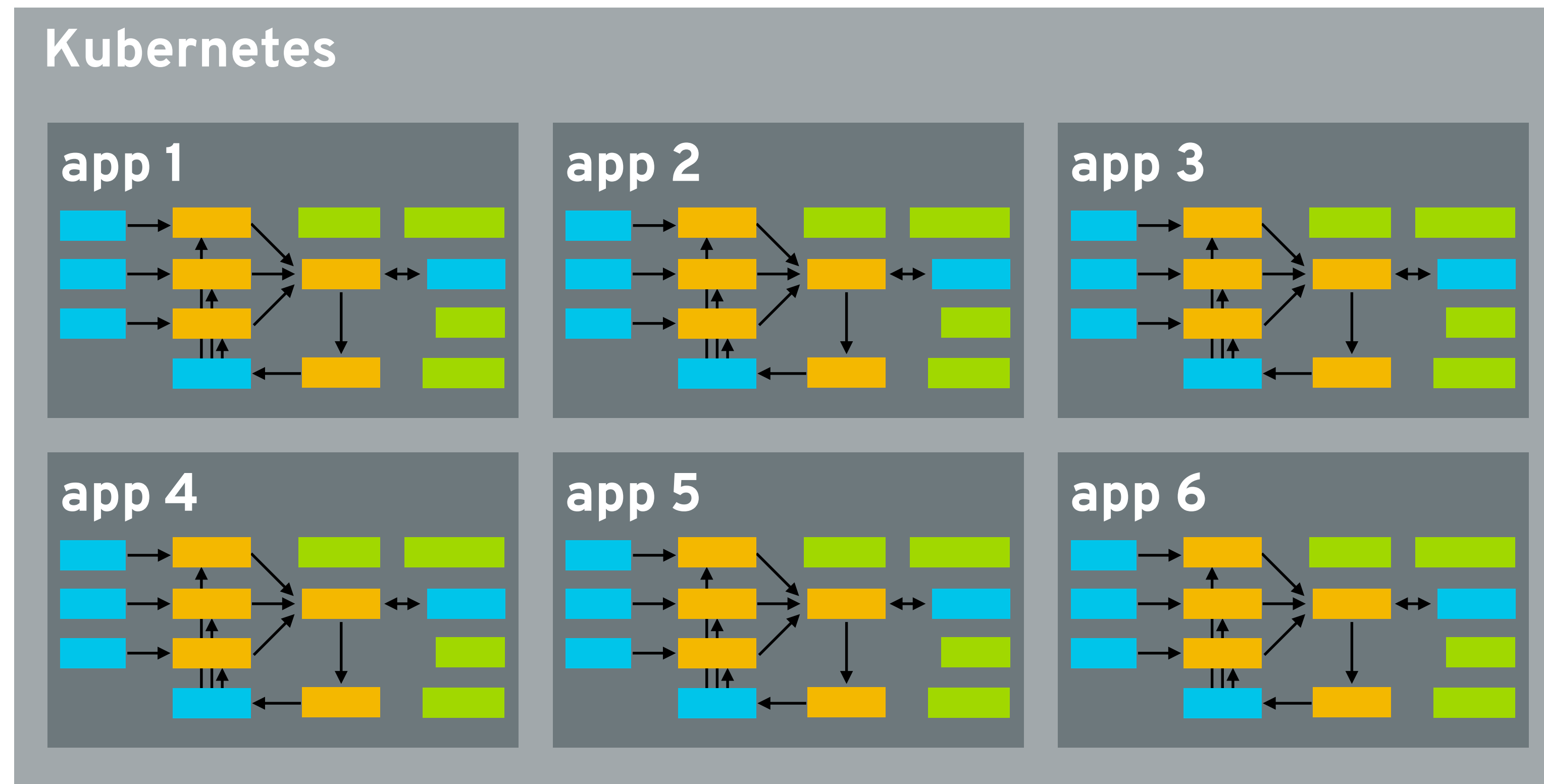


# STORAGE

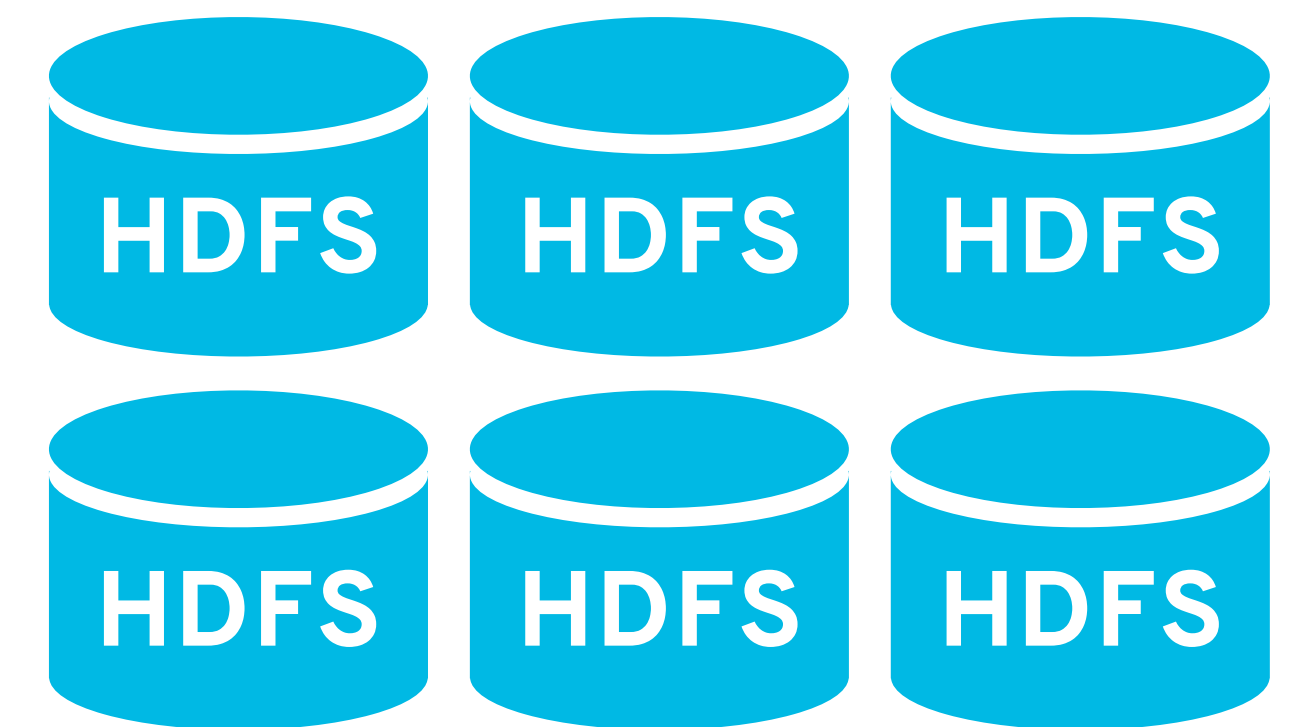
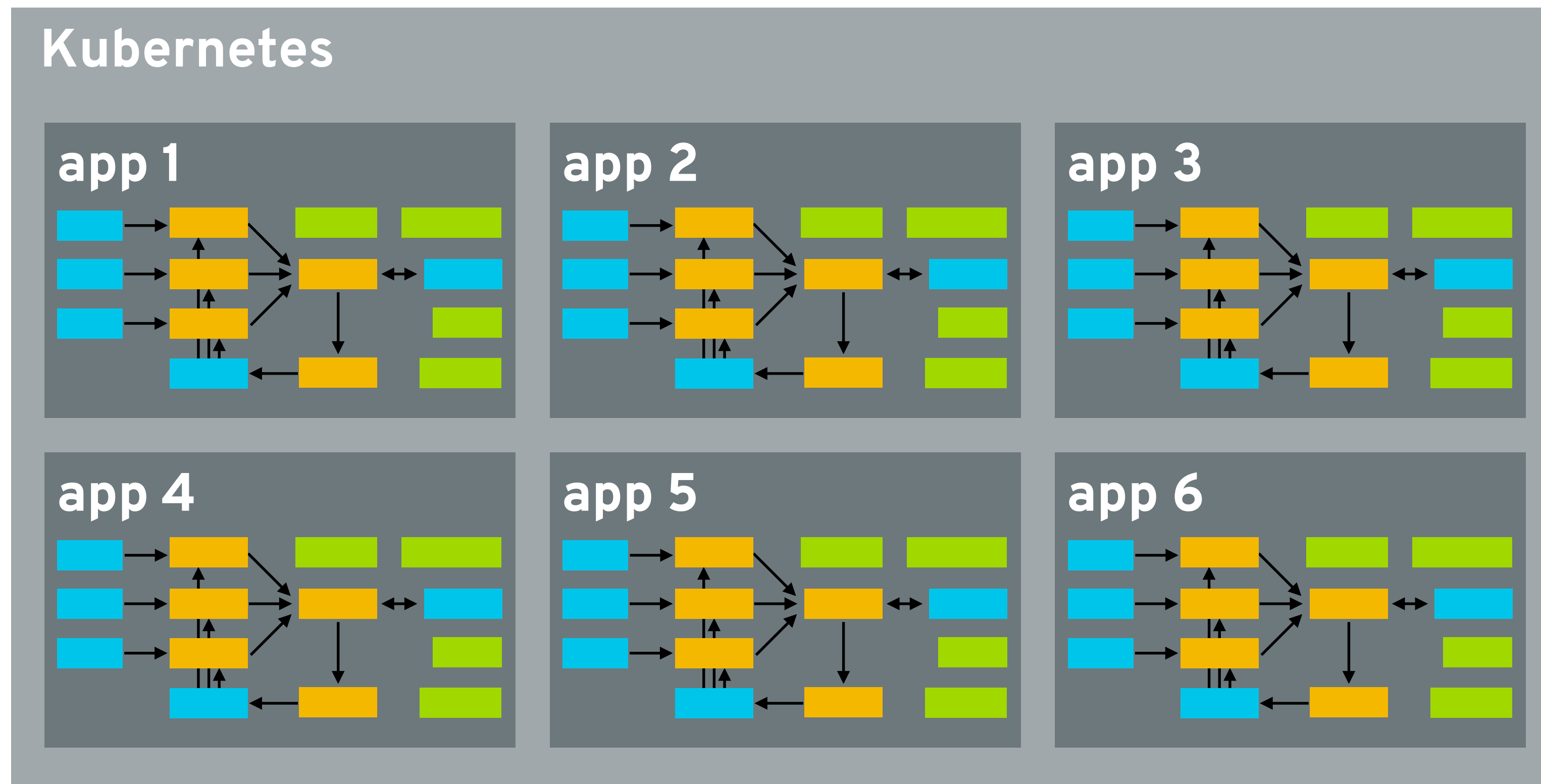


- ✓ familiar interface
- ✓ interoperability with other programs
- ✗ unnecessary semantic guarantees
- ✗ difficult to manage

# STORAGE

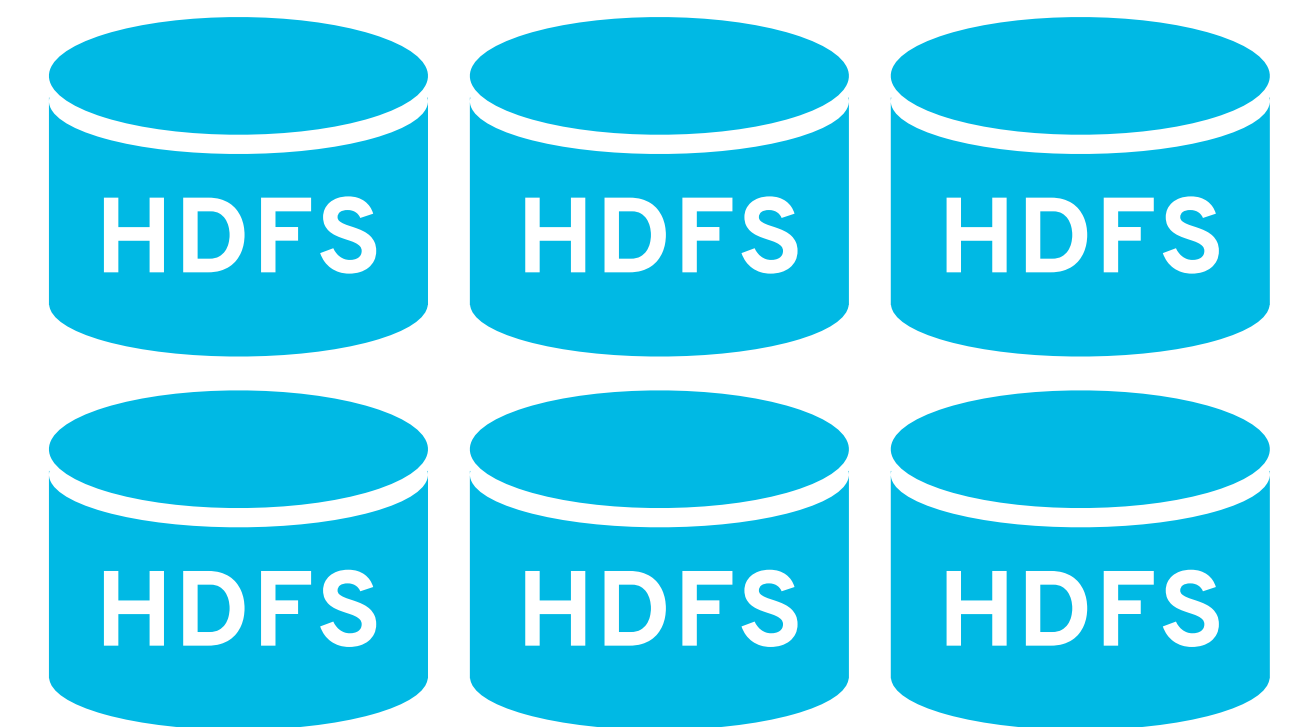
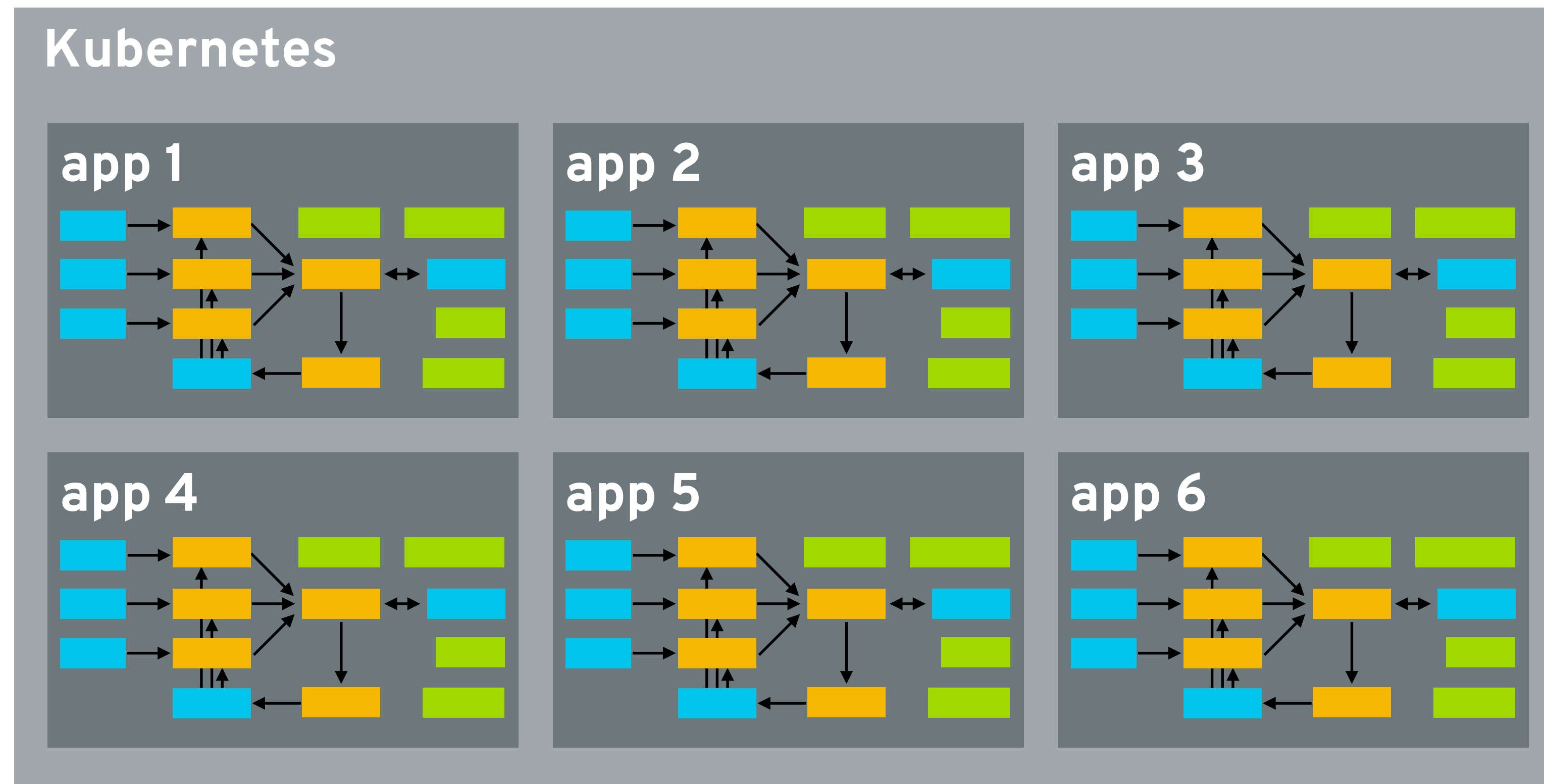


# STORAGE



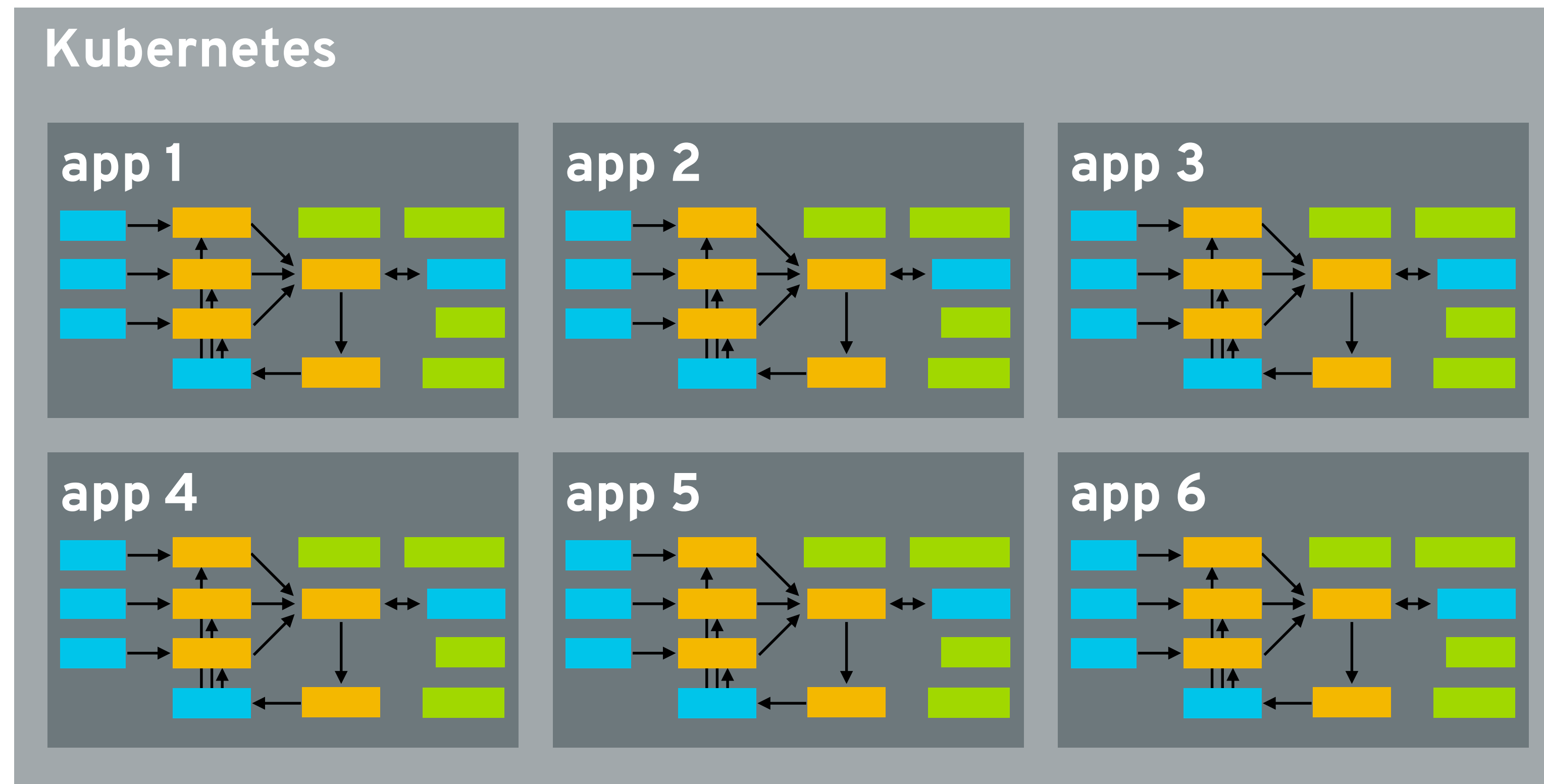
✓ support for legacy  
Hadoop installations

# STORAGE



- ✓ support for legacy Hadoop installations
- ✗ inelastic
- ✗ stateful
- ✗ can't colocate compute and data

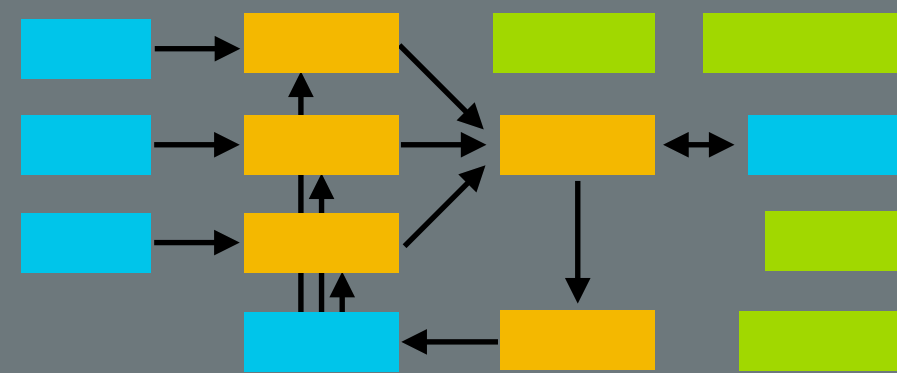
# STORAGE



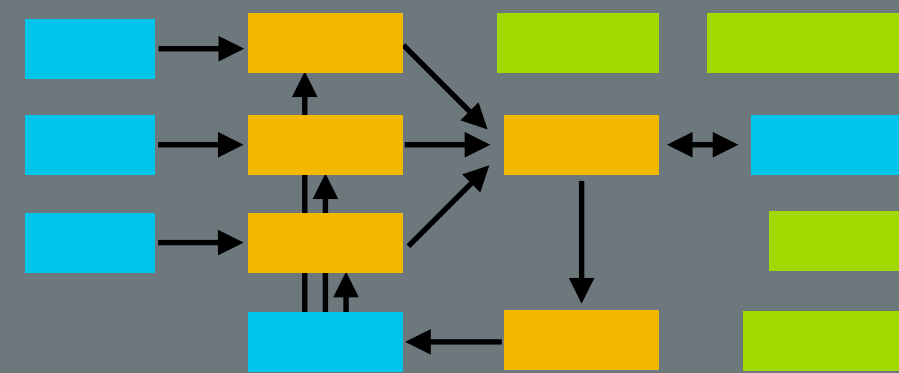
# STORAGE

## Kubernetes

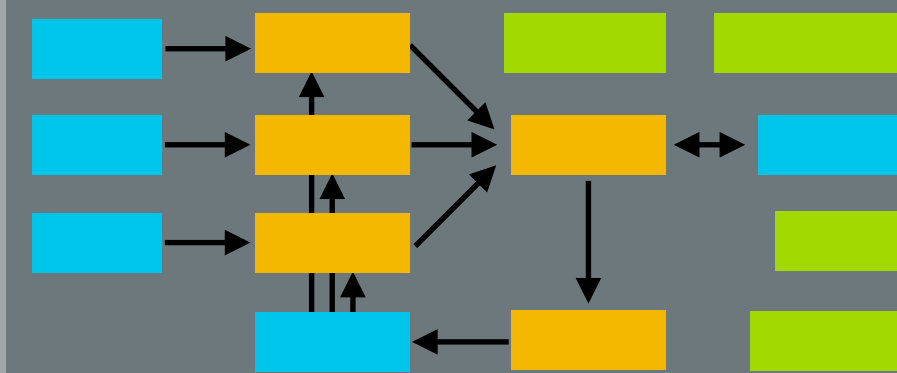
app 1



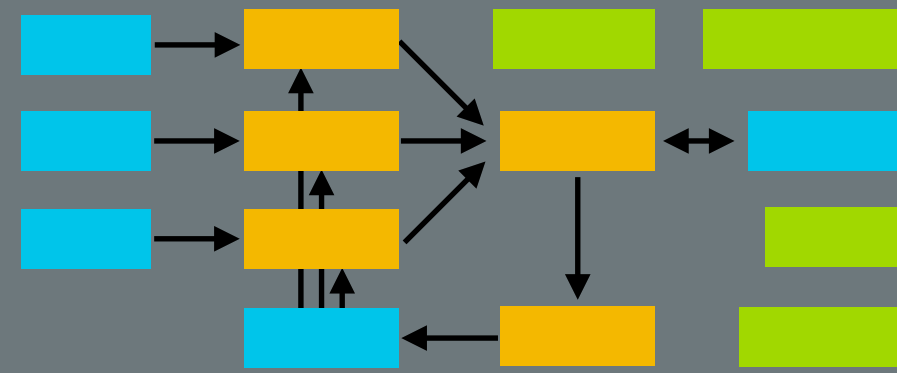
app 2



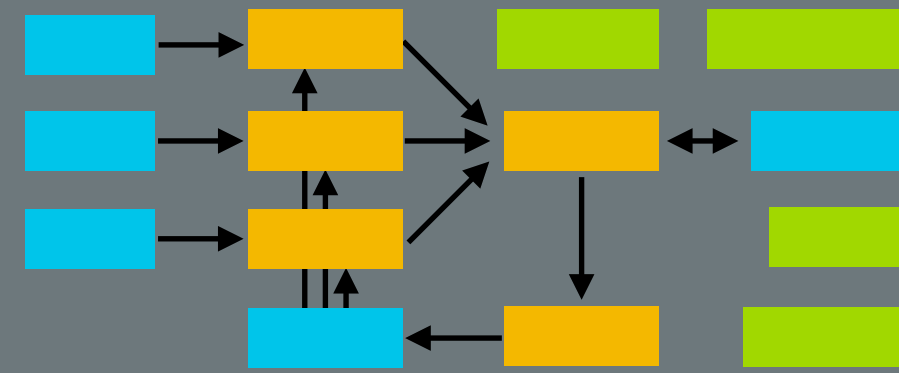
app 3



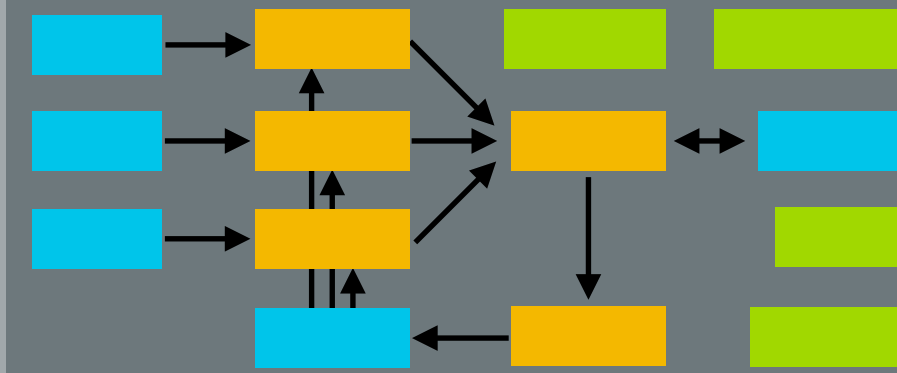
app 4



app 5



app 6

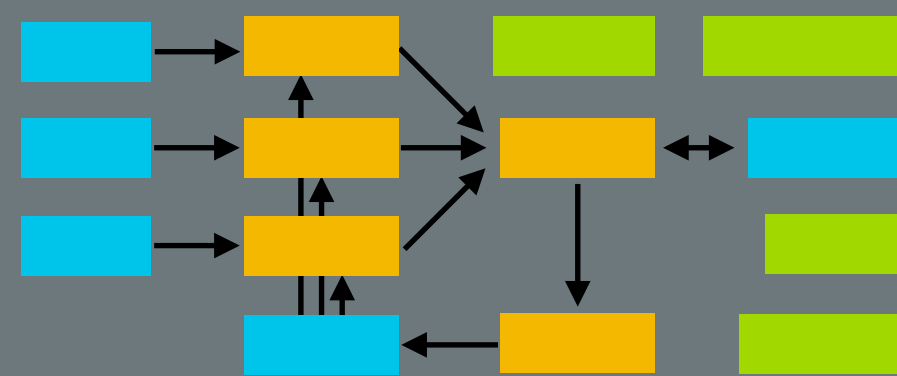


- ✓ interoperability
- ✓ fine-grained AC
- ✓ many implementations

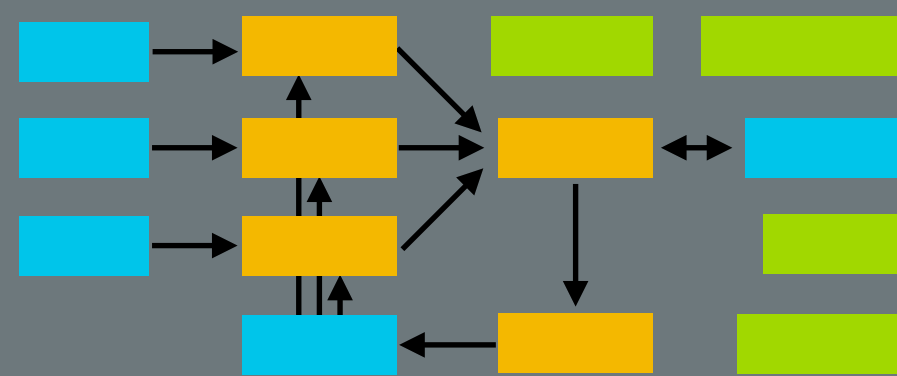
# STORAGE

## Kubernetes

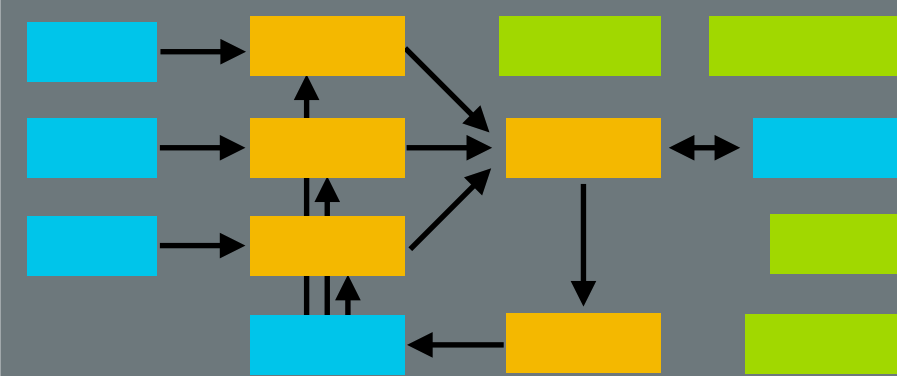
app 1



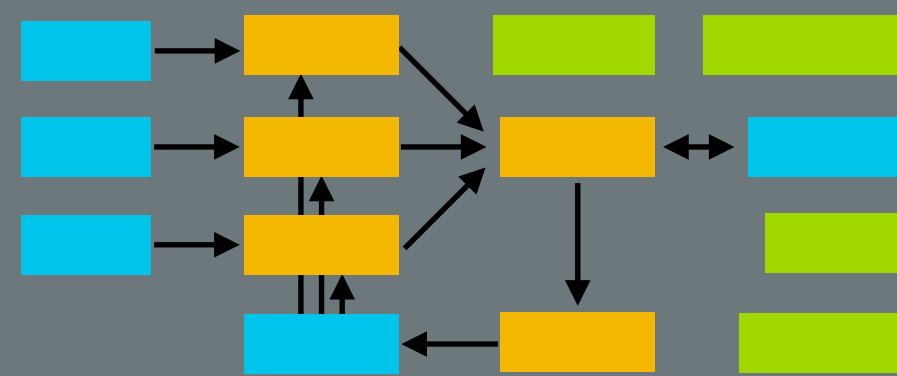
app 2



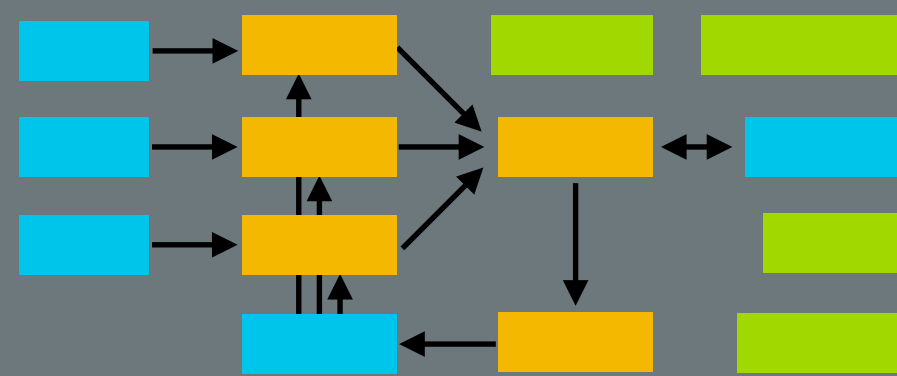
app 3



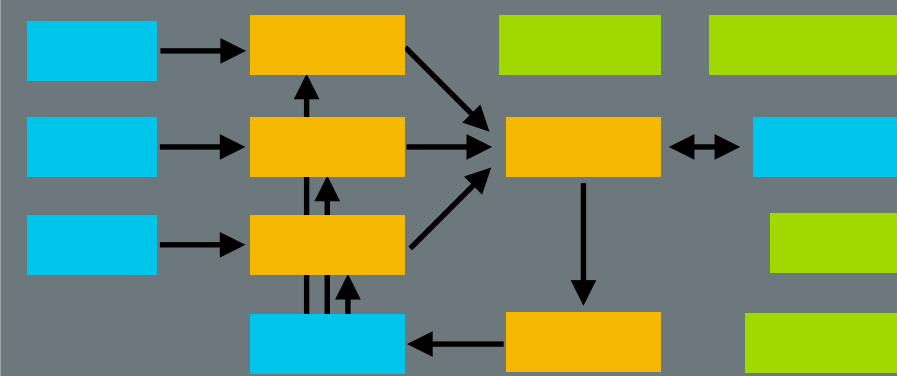
app 4



app 5



app 6

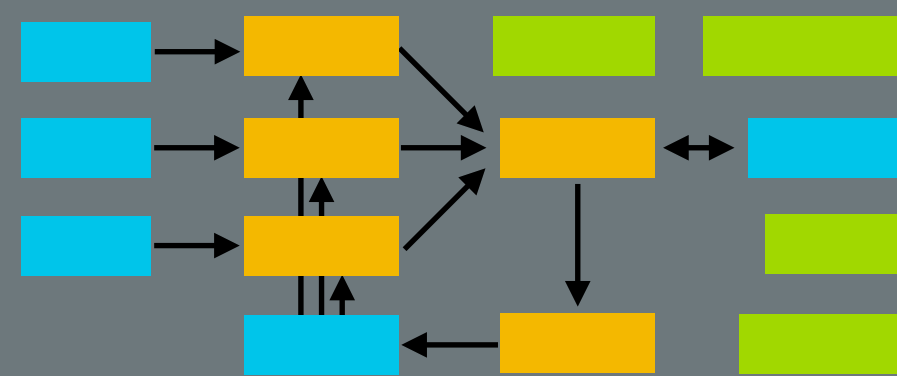


- ✓ interoperability
- ✓ fine-grained AC
- ✓ many implementations
- ✗ consistency model
- ✗ performance (?)

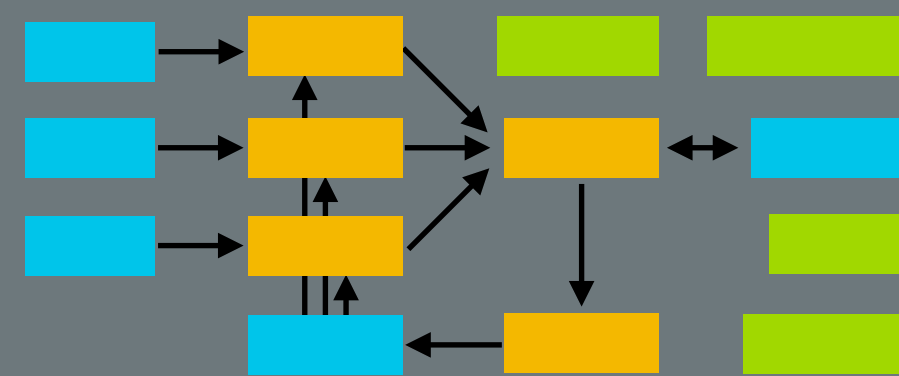
# STORAGE

## Kubernetes

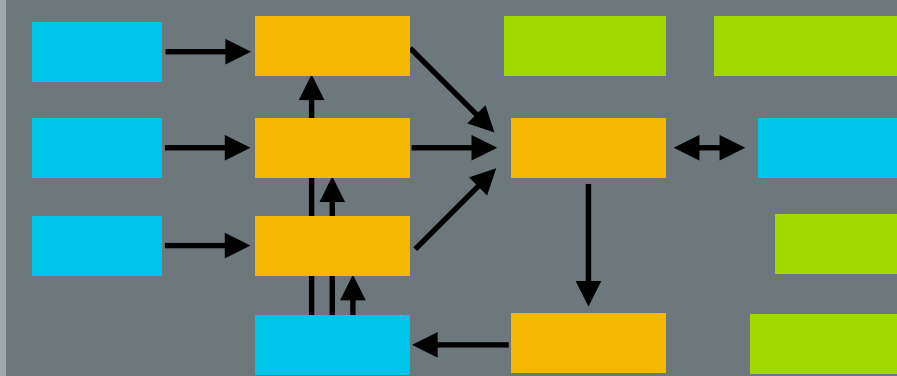
app 1



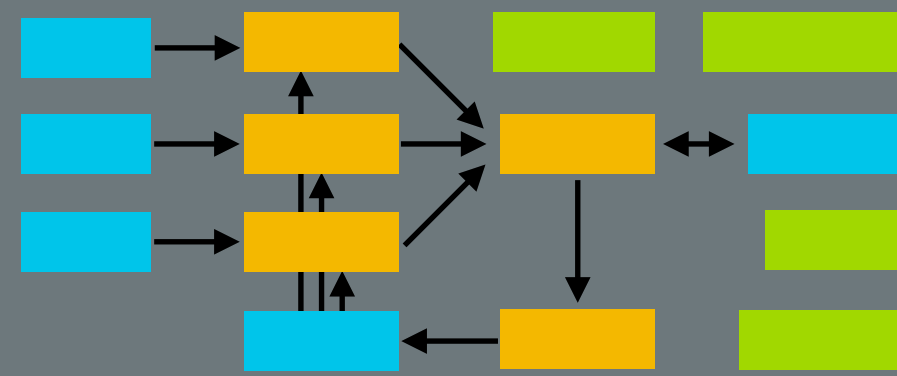
app 2



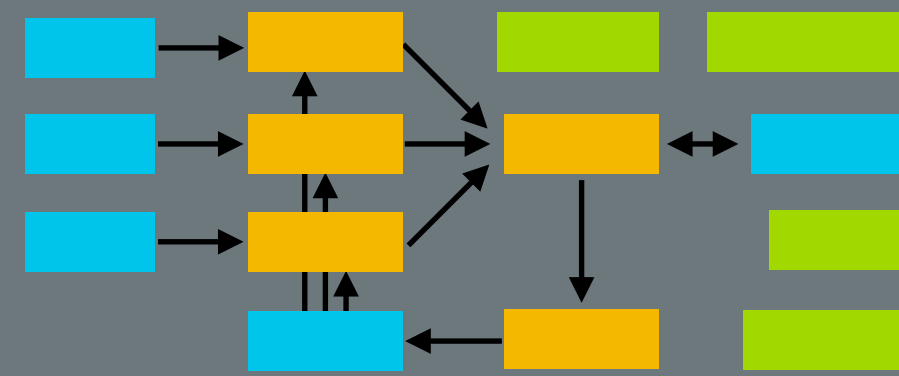
app 3



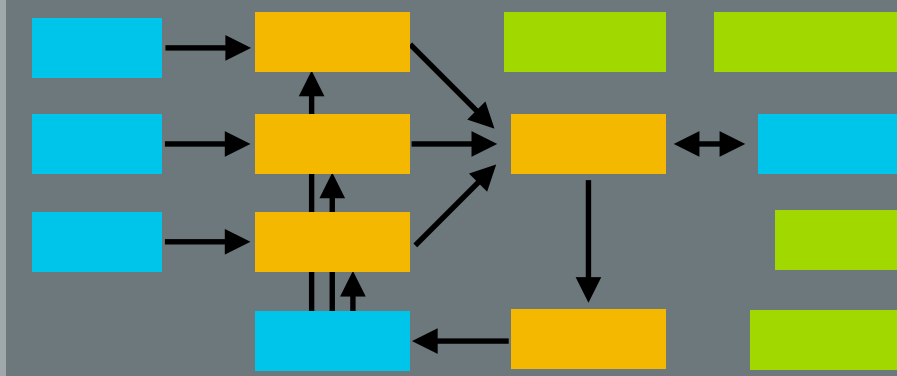
app 4



app 5



app 6



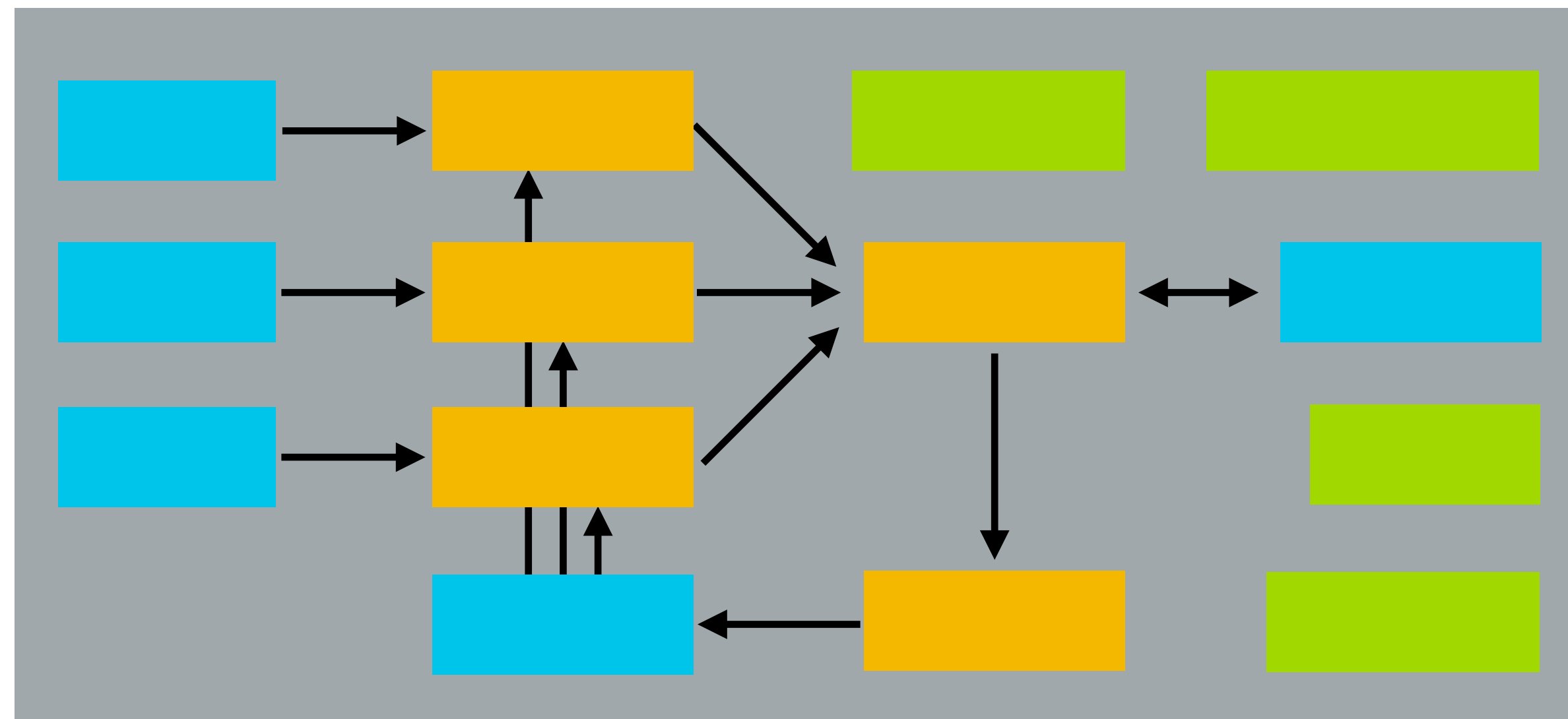
- ✓ interoperability
- ✓ fine-grained AC
- ✓ many implementations
- ✗ consistency model
- ✗ performance



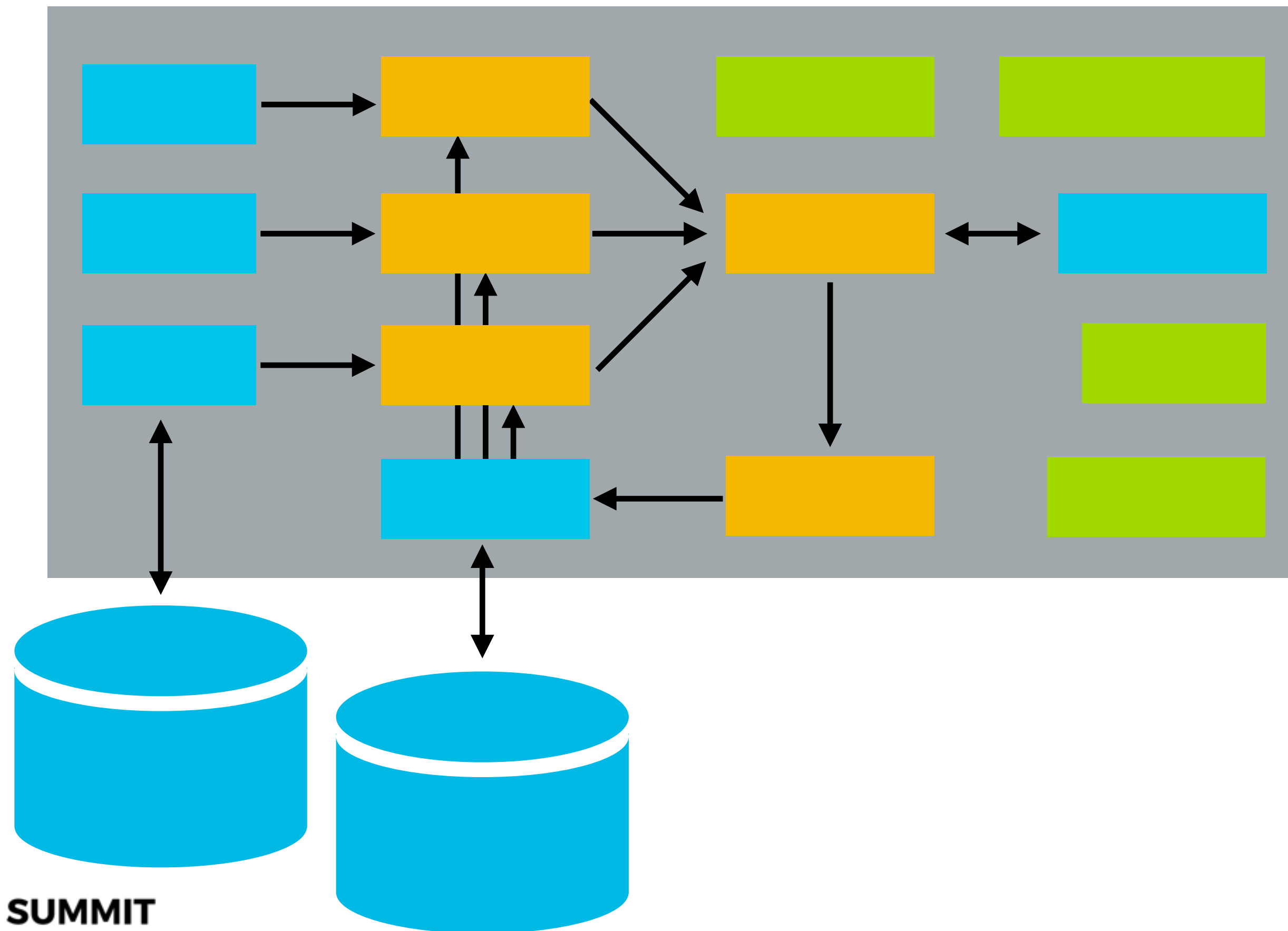
**“...in a cloud native architecture, the benefit of HDFS is actually very small and that is why many cloud-first organizations no longer run HDFS, or only run it as a caching layer for S3.”**

**—Reynold Xin on Quora (<http://qr.ae/TAF4cN>)**

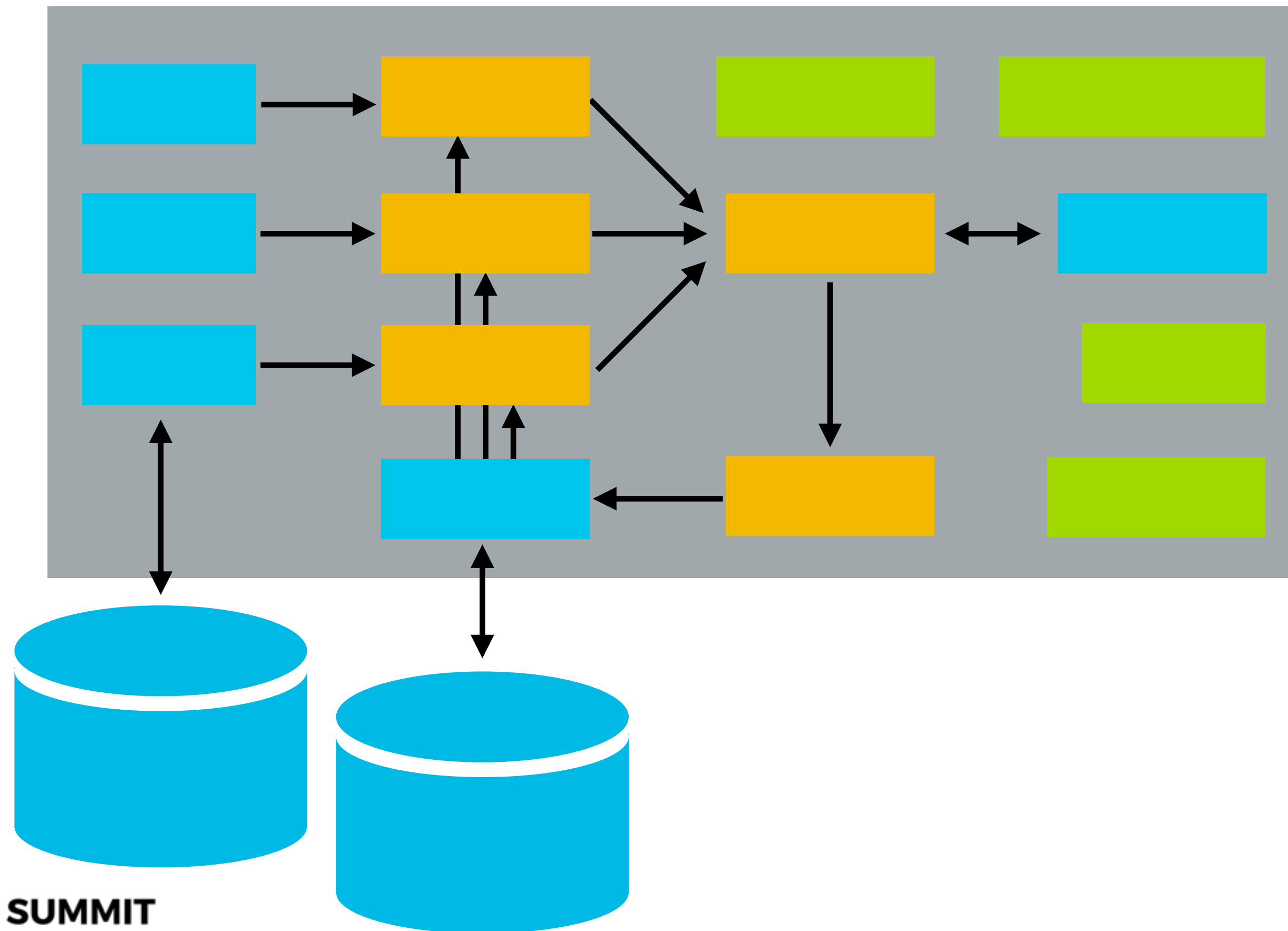
# NETWORKING



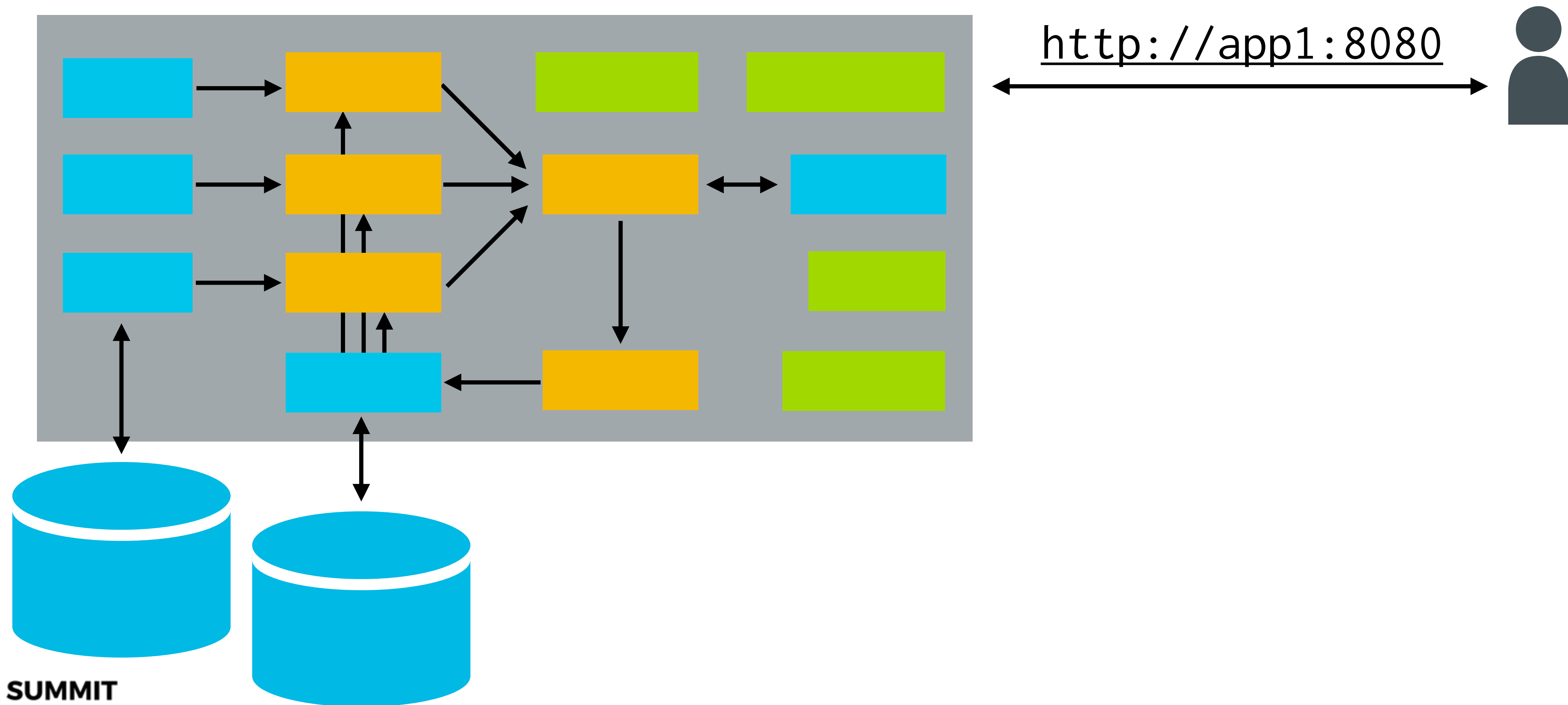
# NETWORKING



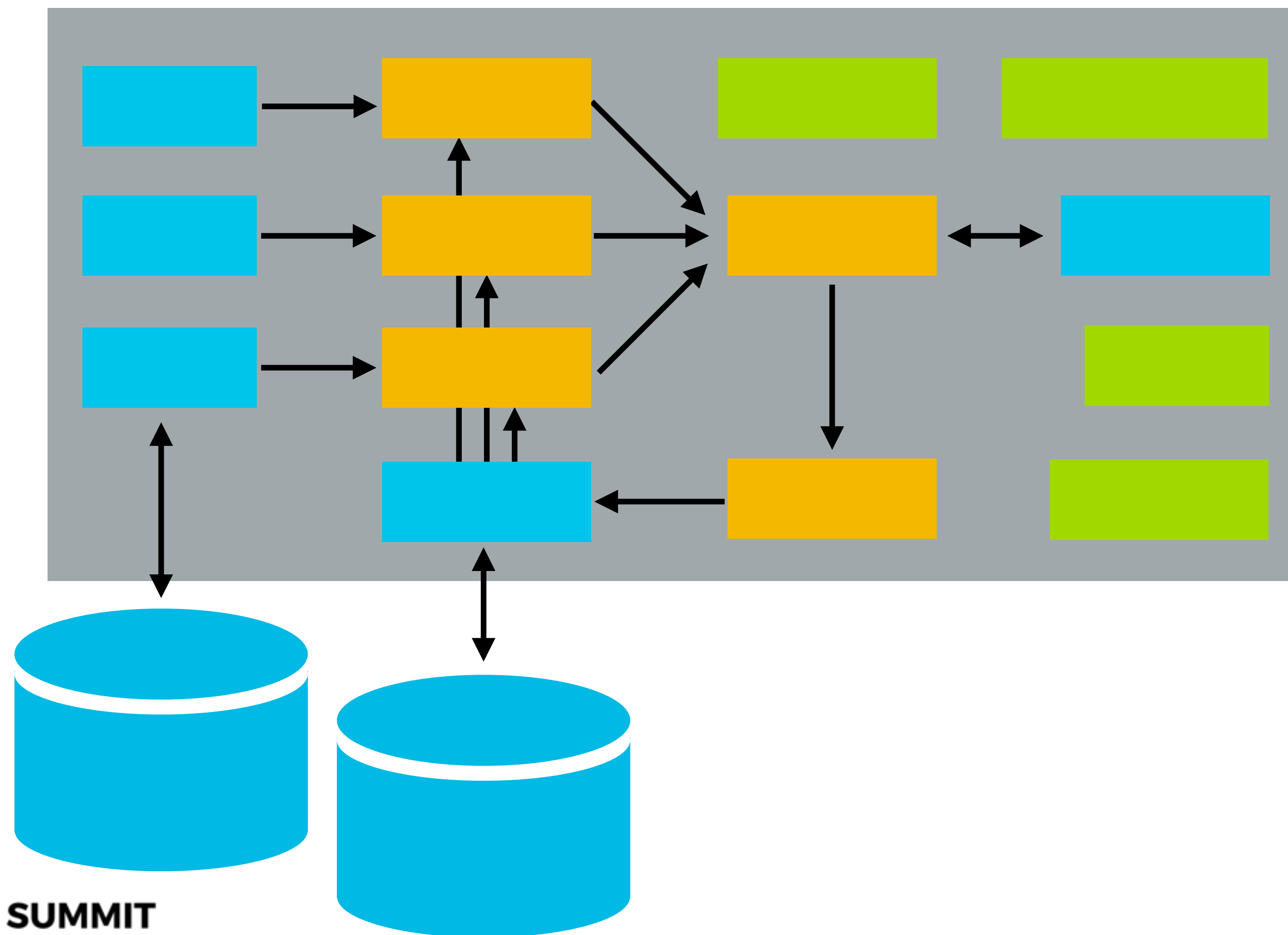
# NETWORKING

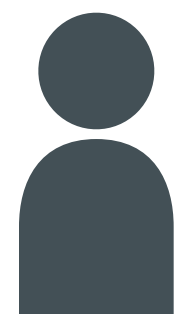


# NETWORKING



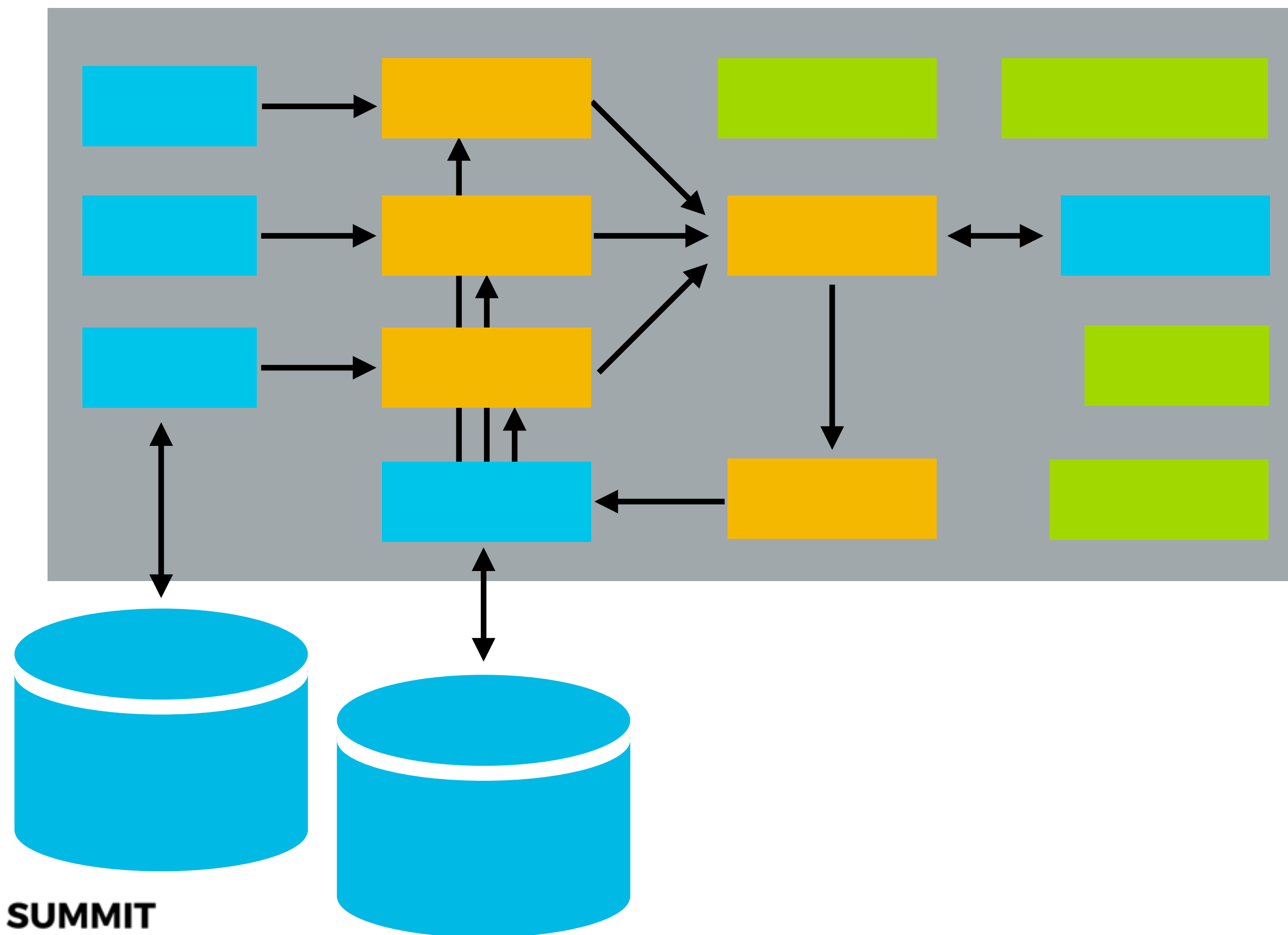
# NETWORKING

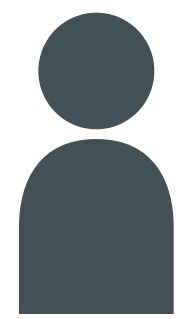


<http://app1:8080> 

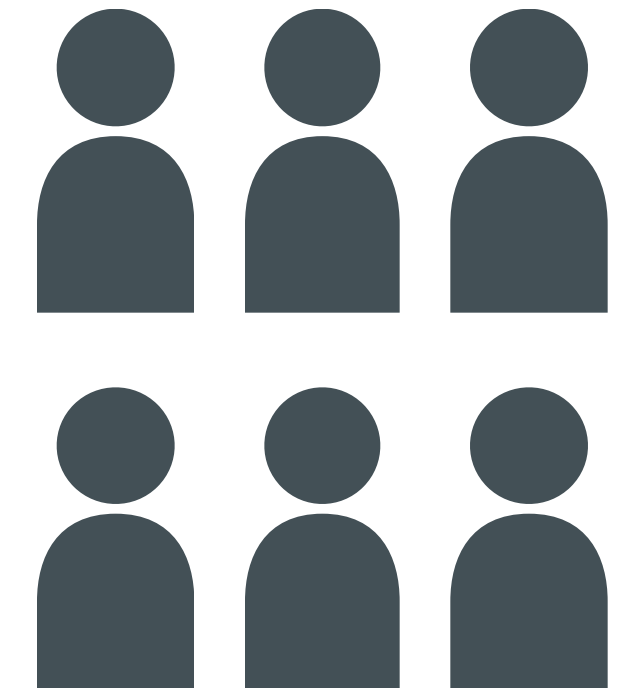
✗ can't access worker web UI  
(but wait for Spark 2.1!)

# NETWORKING

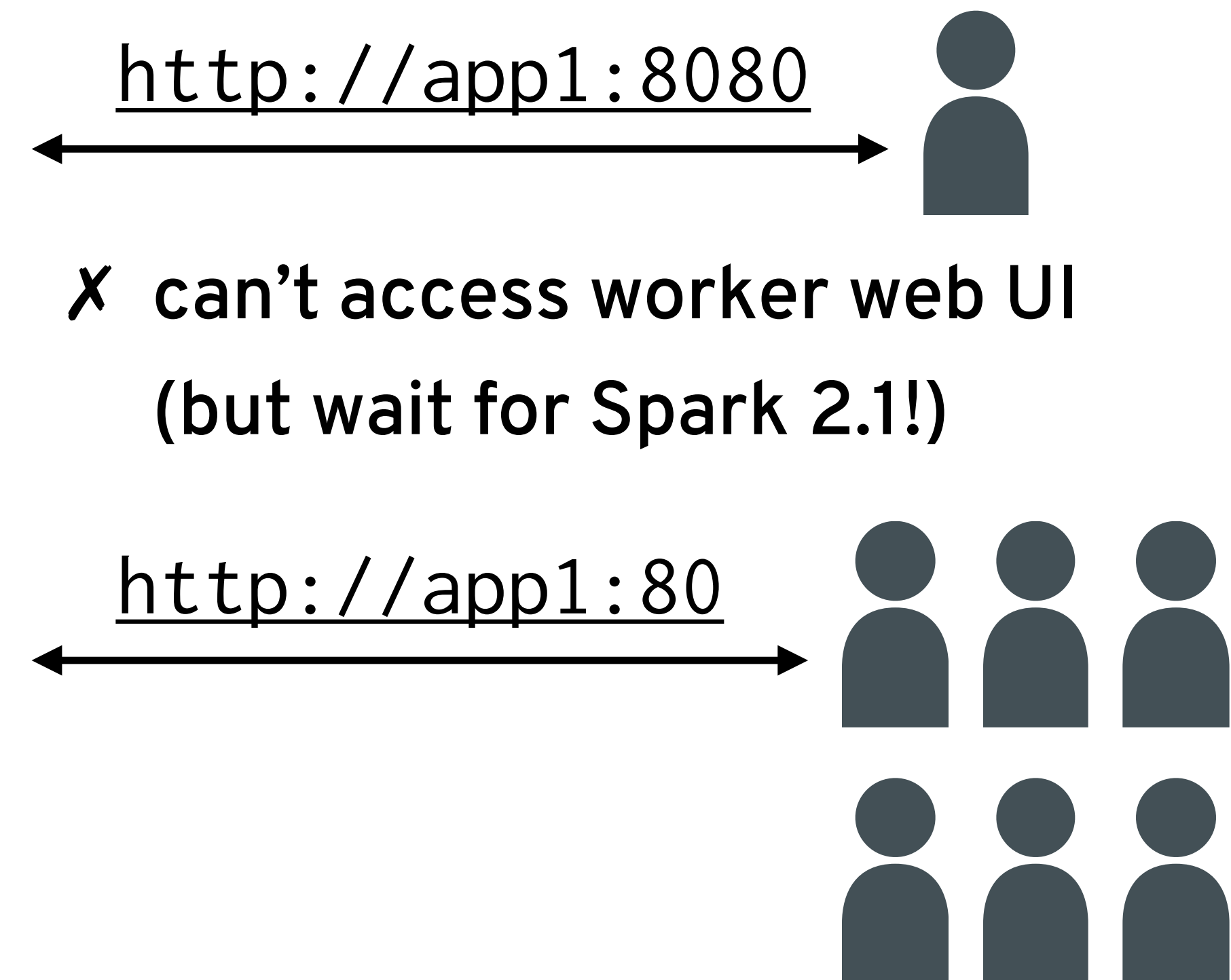
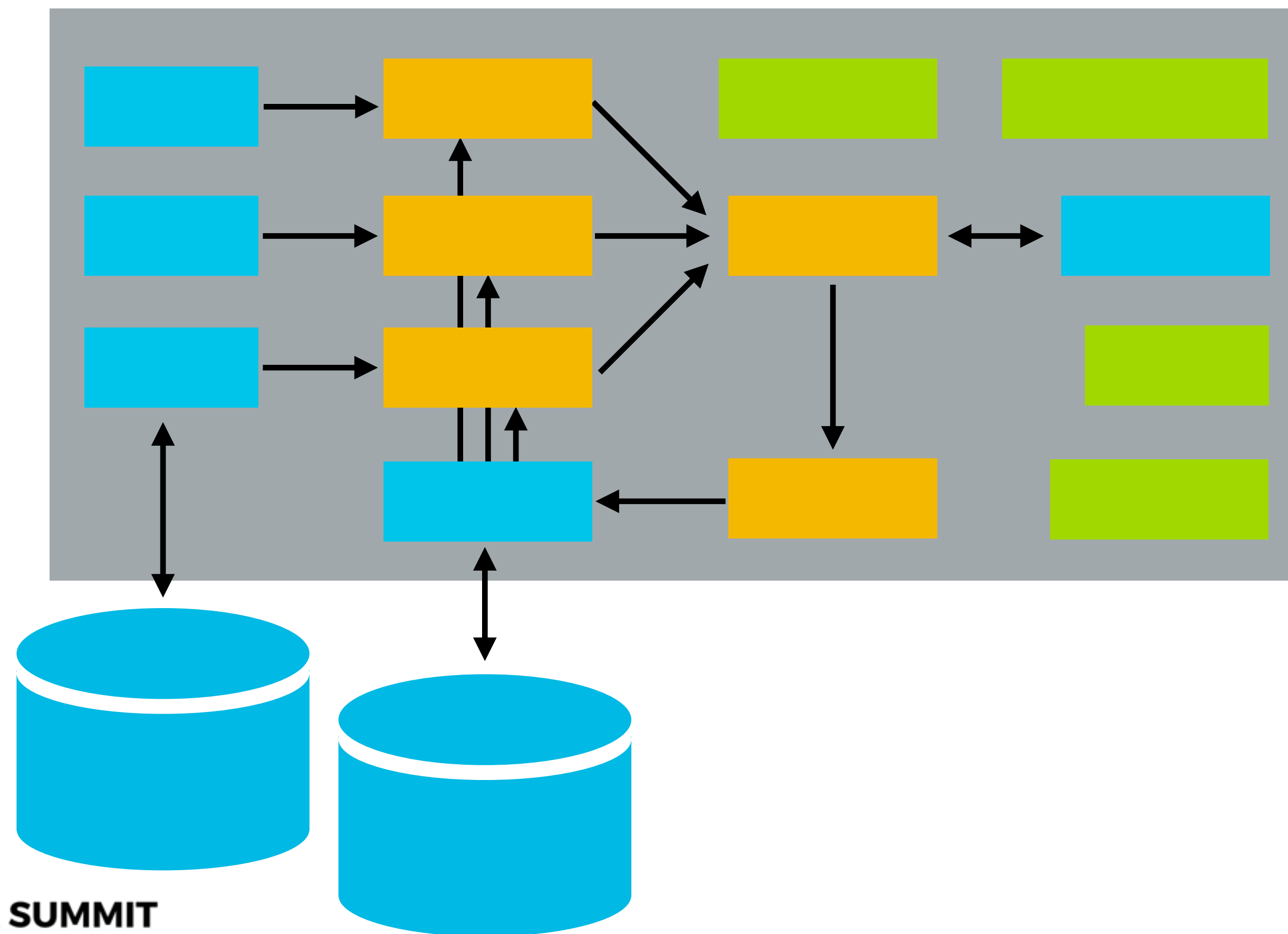


<http://app1:8080> 

✗ can't access worker web UI  
(but wait for Spark 2.1!)

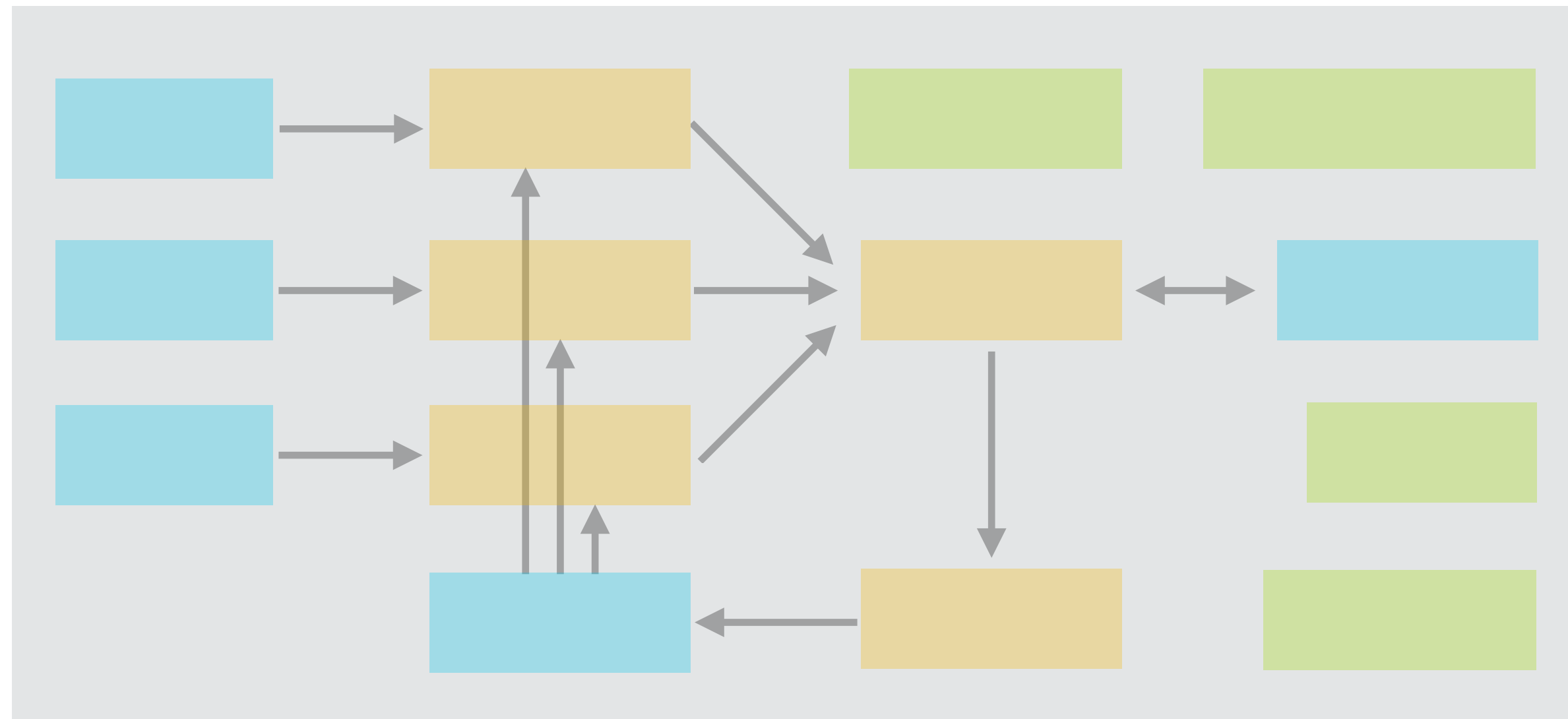


# NETWORKING

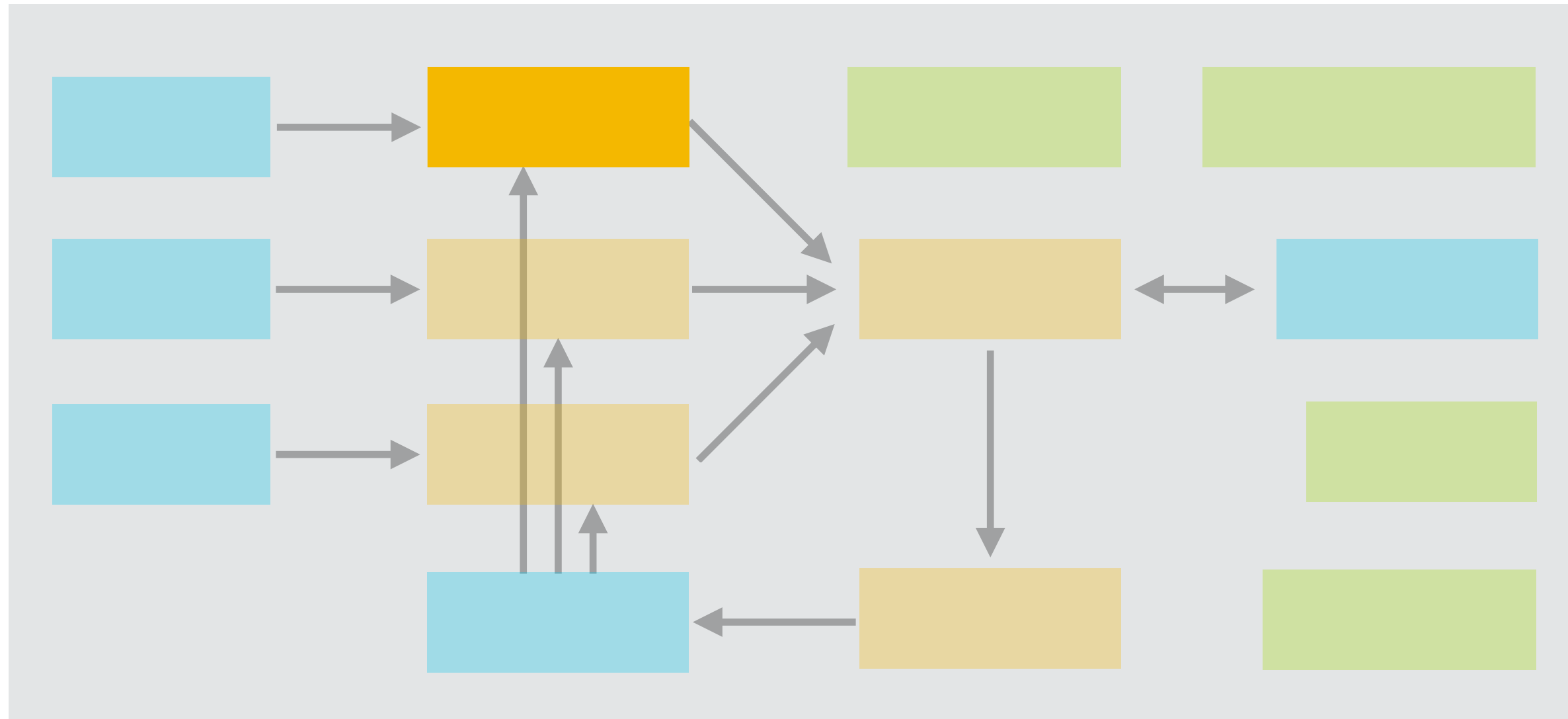




# SECURITY



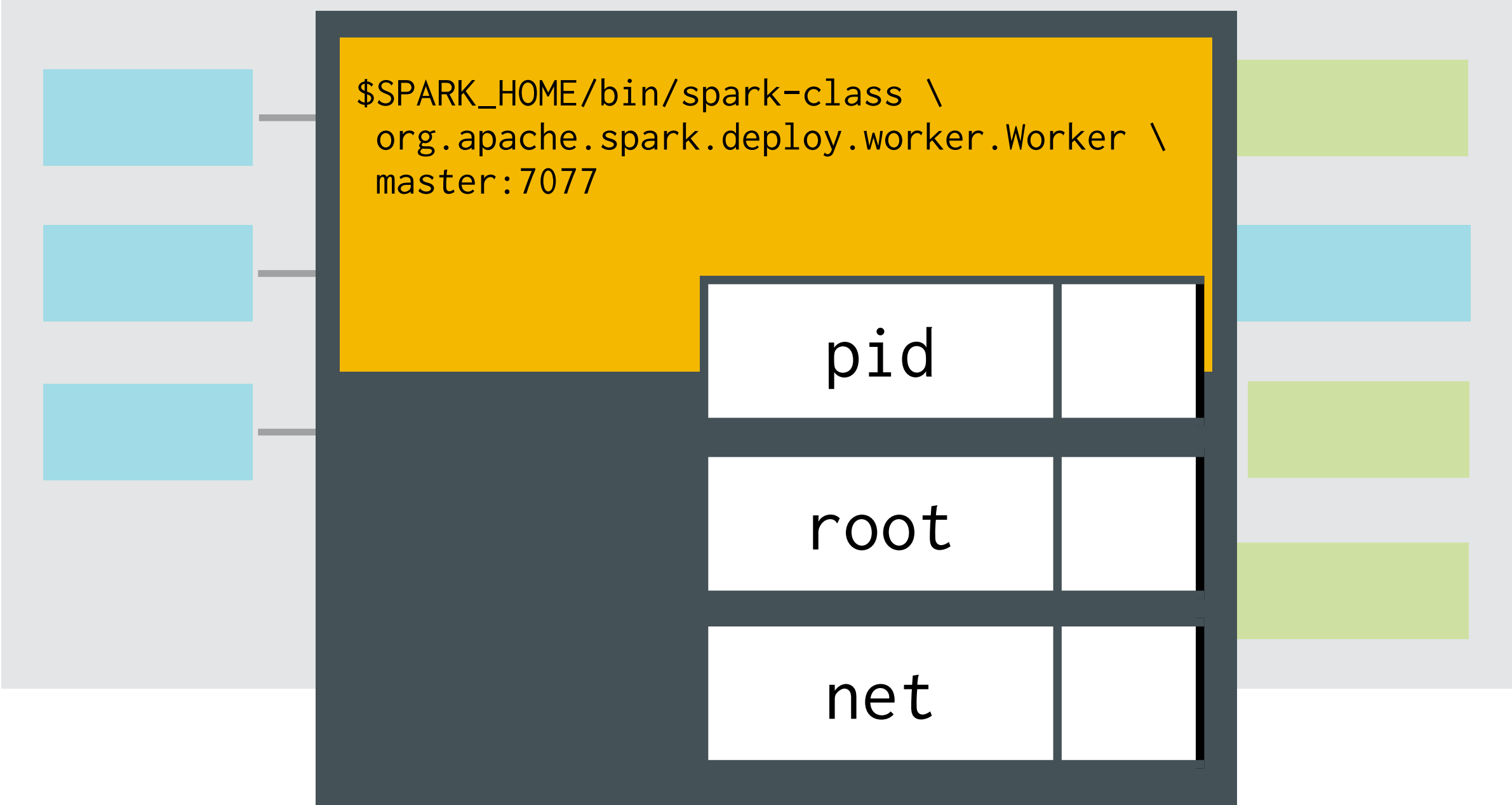
# SECURITY



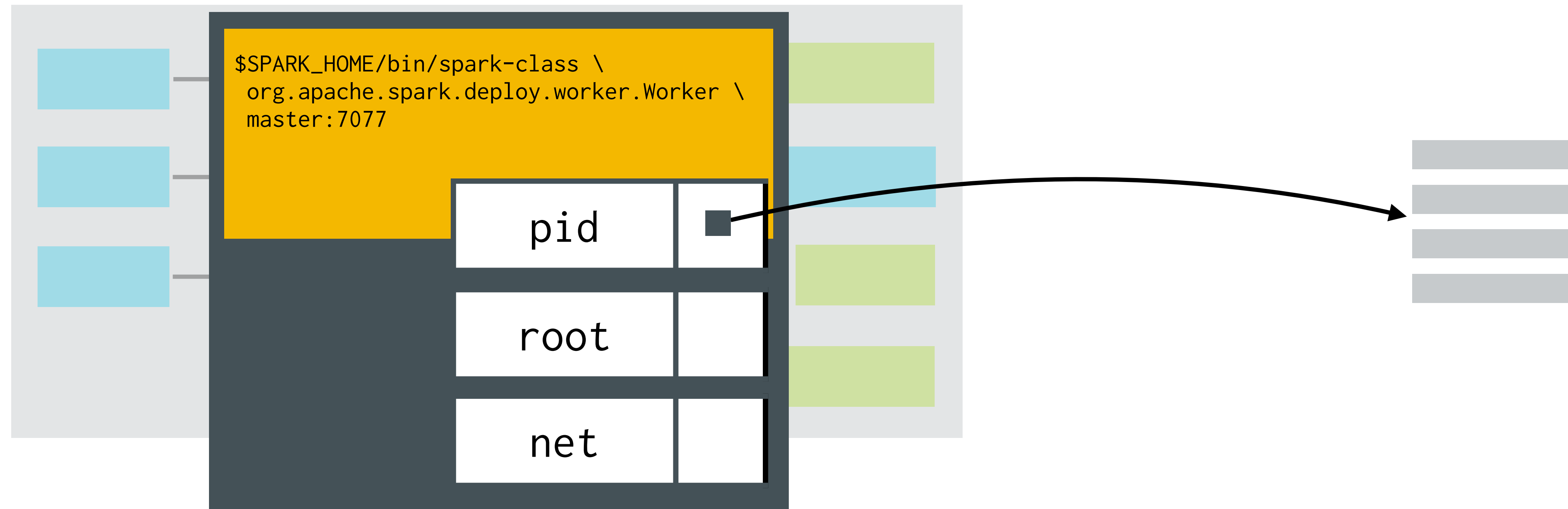
# SECURITY



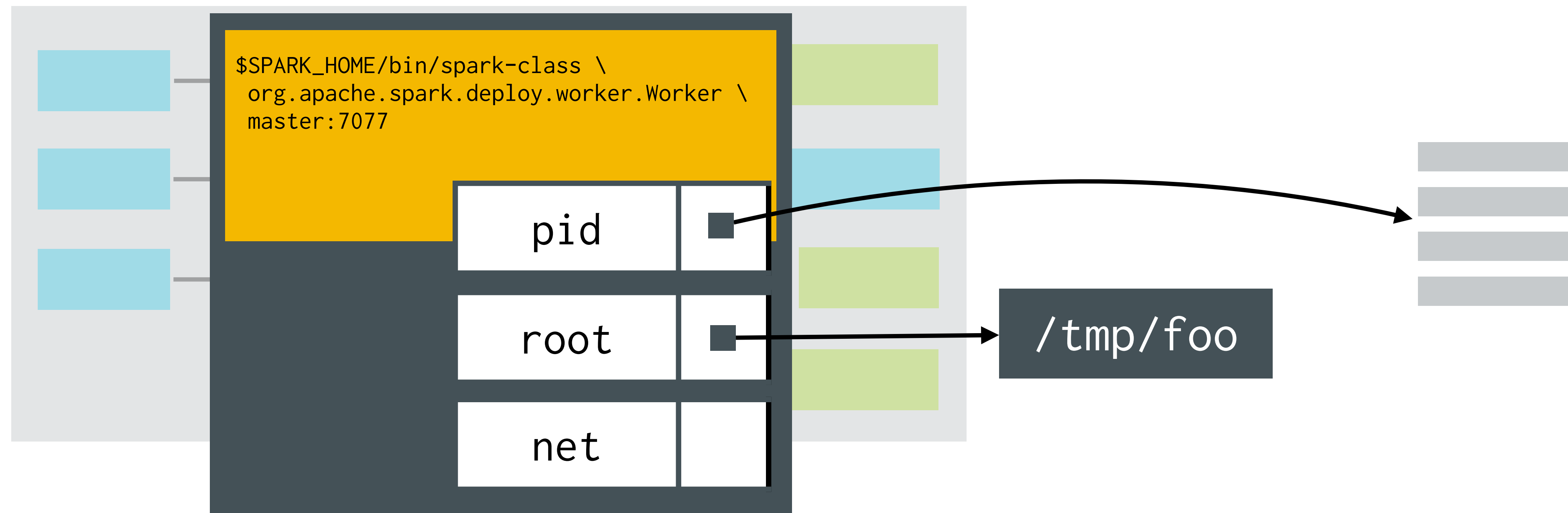
# SECURITY



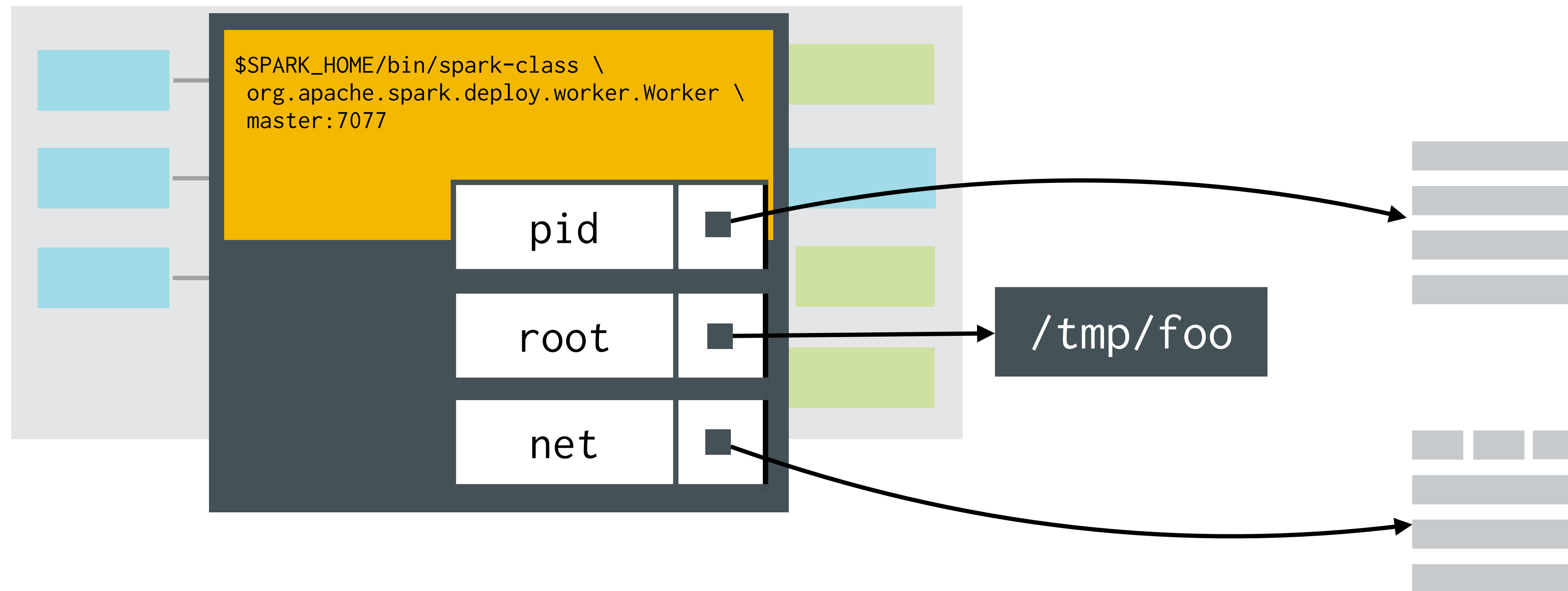
# SECURITY



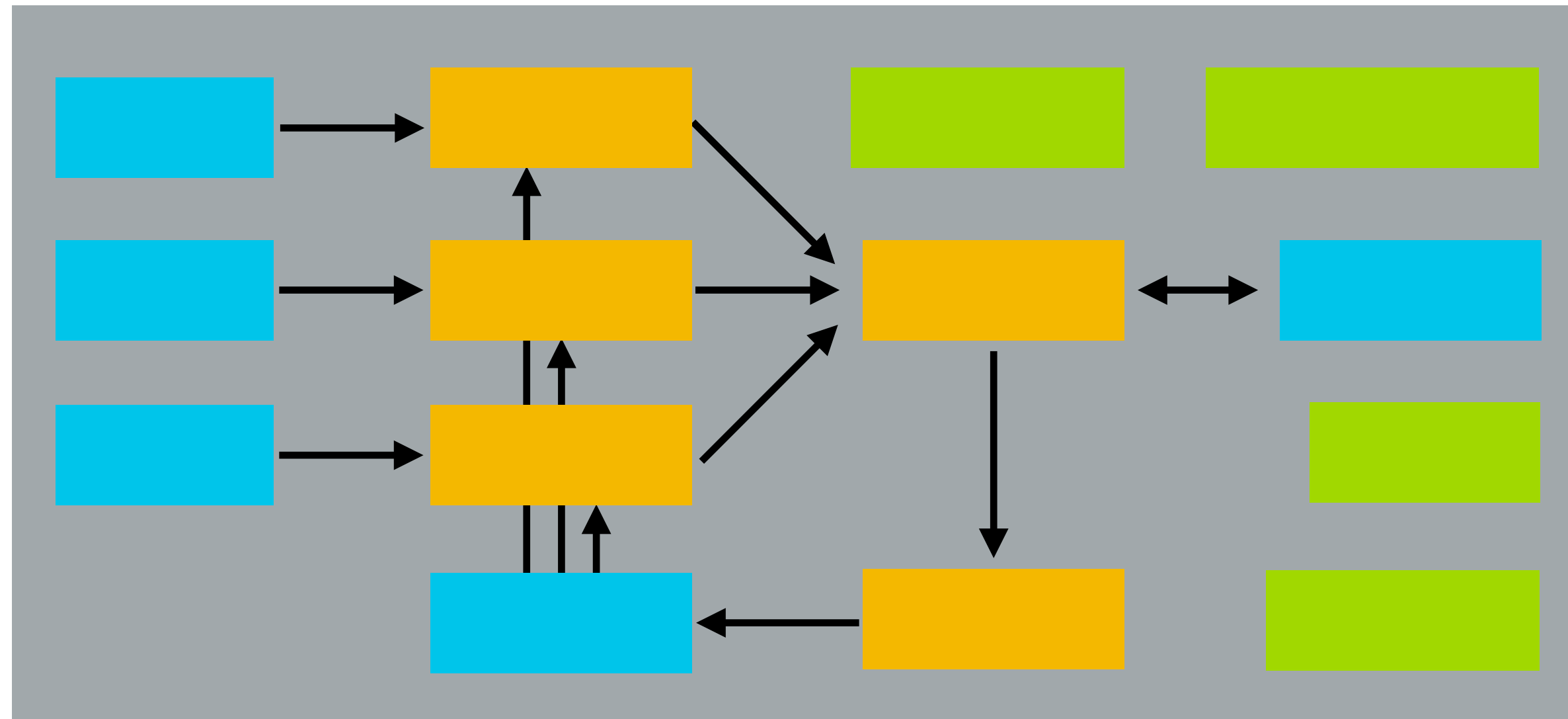
# SECURITY



# SECURITY

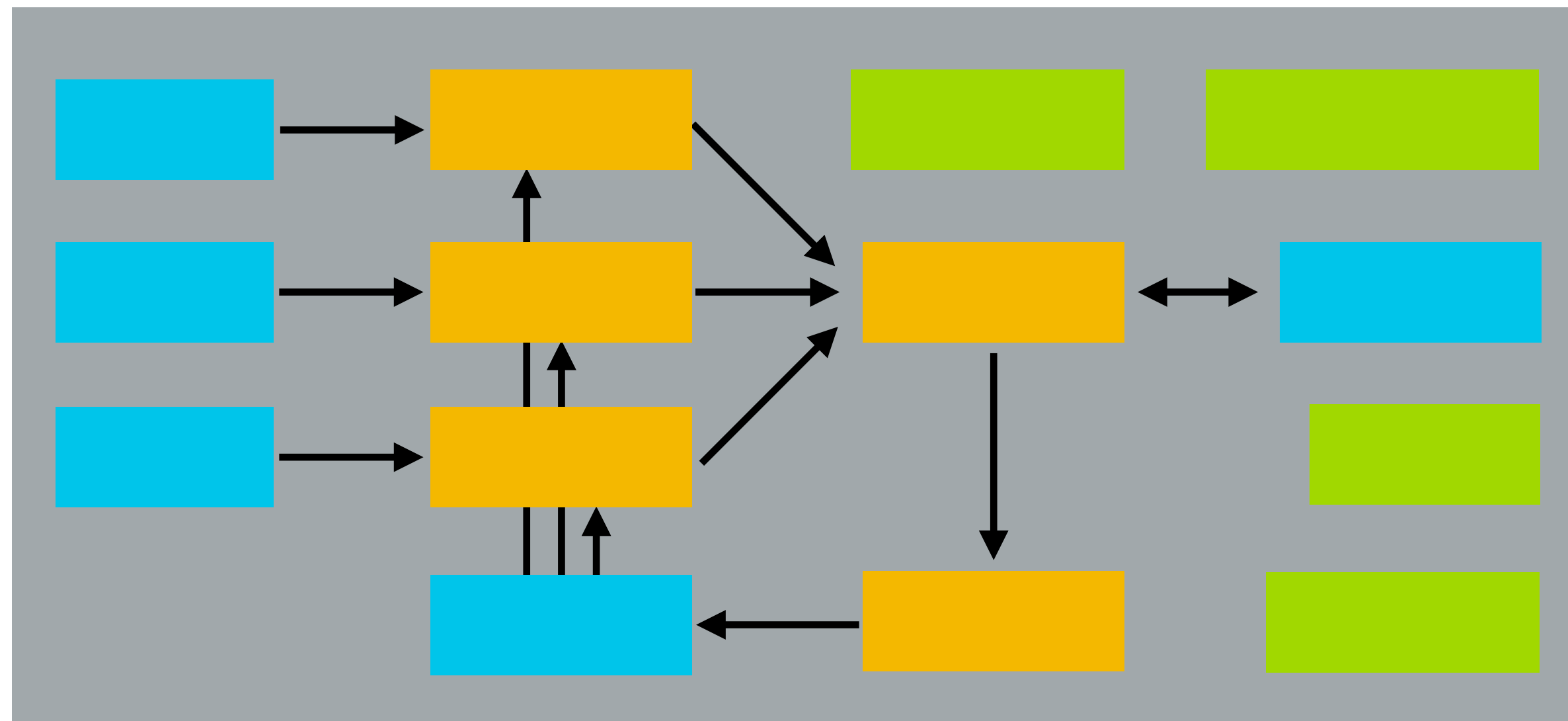


# SECURITY



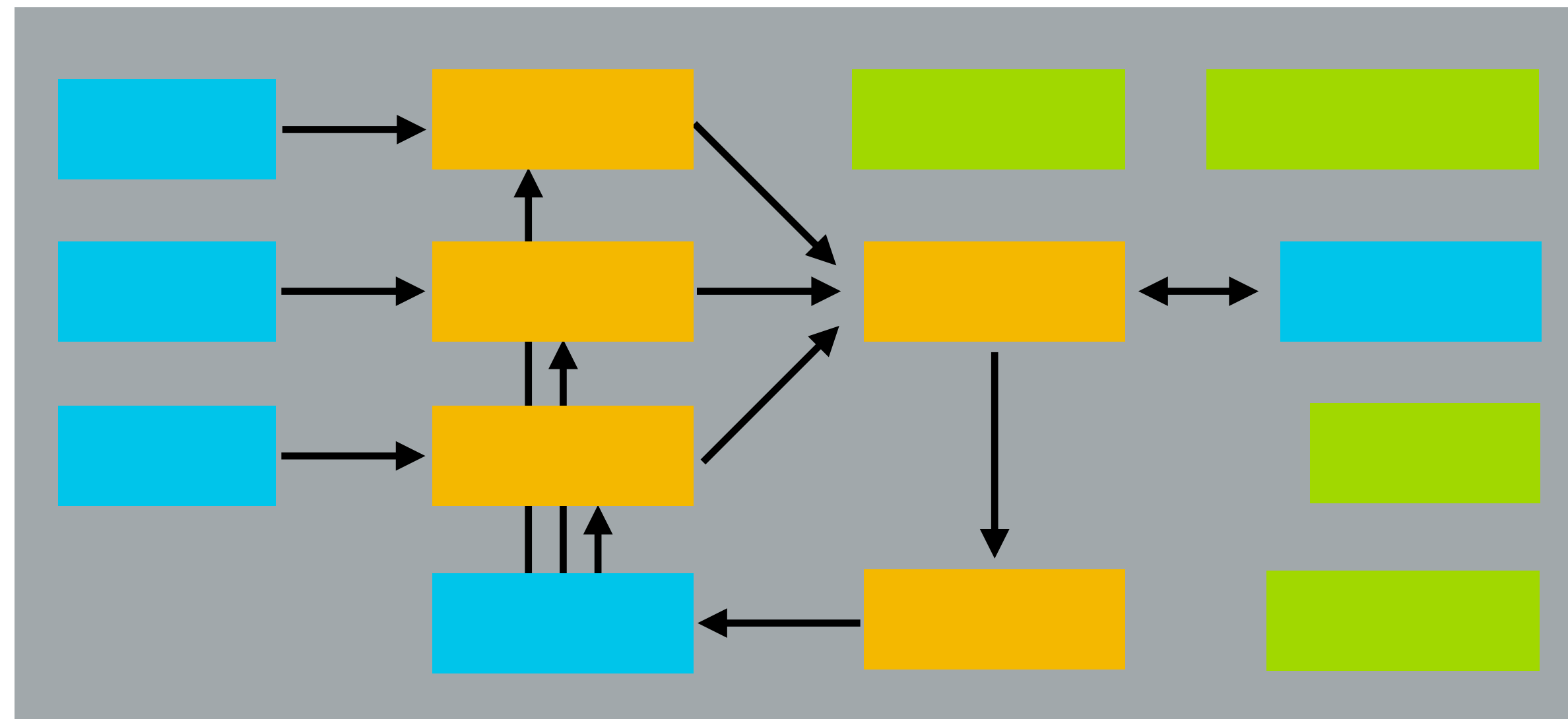


# SECURITY



k8s namespace

# SECURITY



# k8s namespace\*

**NEXT STEPS: FUTURE WORK &  
PLAYING ALONG AT HOME**

# NEXT STEPS

Further performance evaluation

Better developer experience

Improved scheduling of Spark tasks on Kubernetes

# TRY IT OUT YOURSELF

Kubernetes standalone Spark example:

<https://github.com/kubernetes/kubernetes/tree/master/examples/spark>

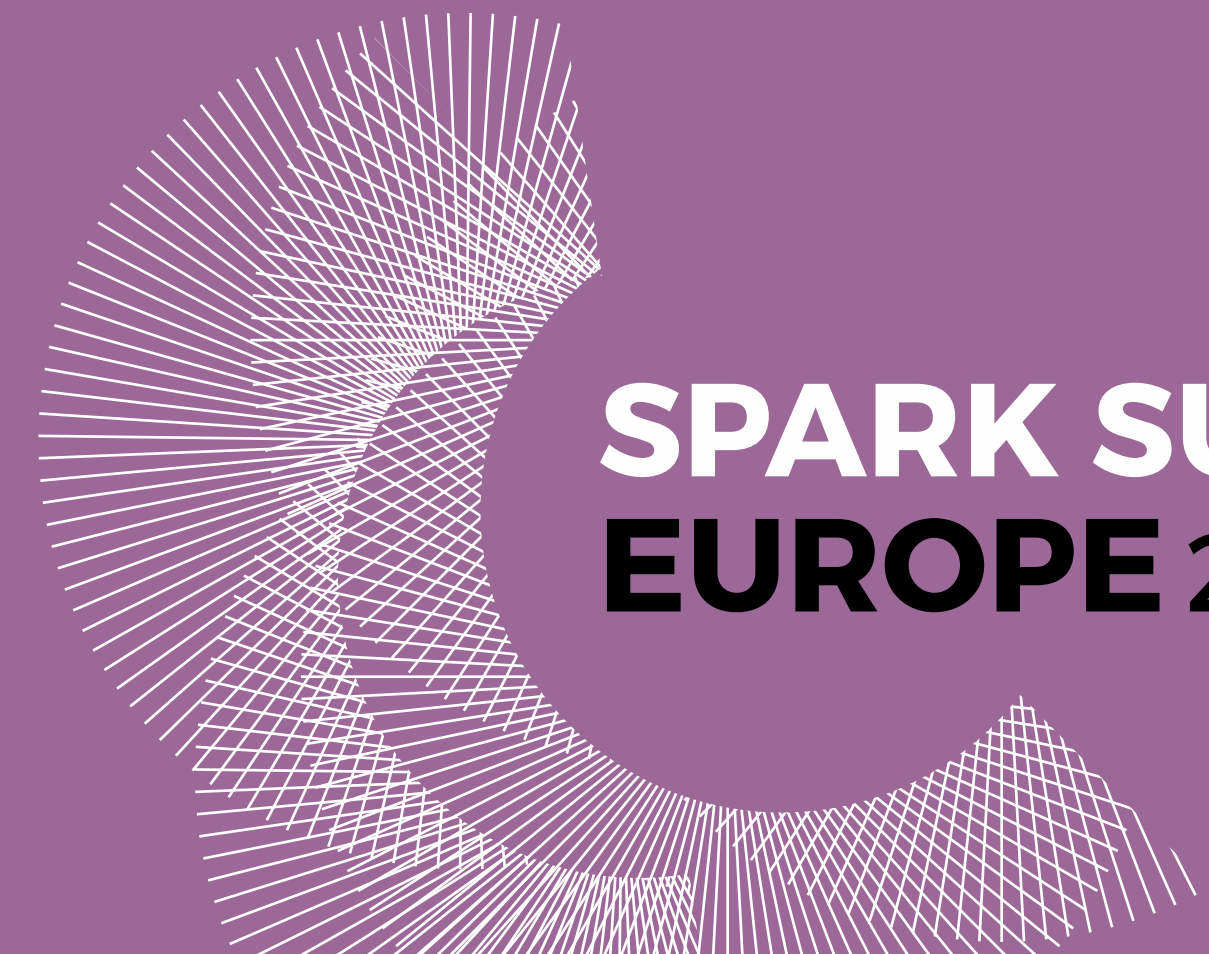
Enabling Spark on OpenShift: <https://github.com/radanalyticsio>

Native Spark on Kubernetes proposal:

<https://github.com/kubernetes/kubernetes/issues/34377>

# THANKS!

@willb • willb@redhat.com  
<https://chapeau.freevariable.com>



**SPARK SUMMIT**  
**EUROPE 2016**