Red Hat

Red Hat Dallas Emerging Tech Summit

December 5, 2019

AI and ML on Kubernetes

William Benton • Engineering Manager and Senior Principal Engineer

What is machine learning?

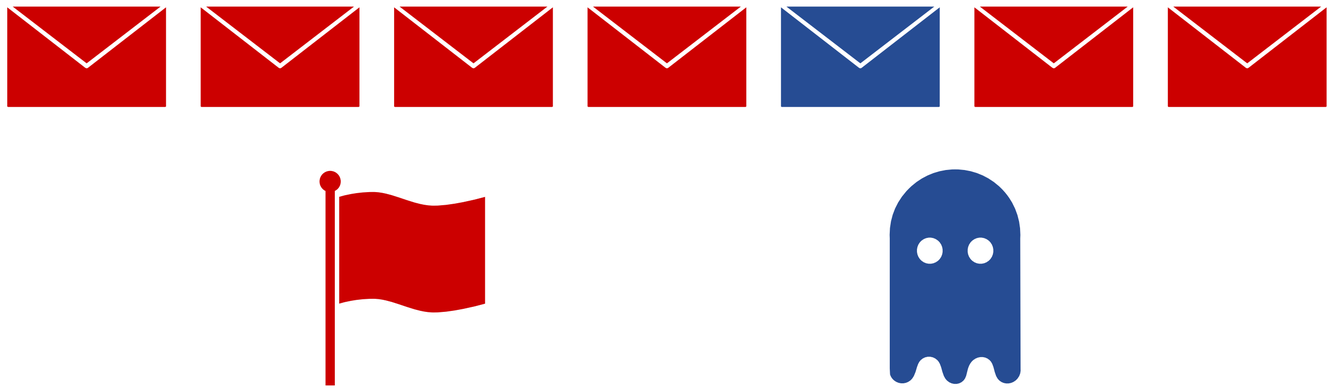
codifying problem and metrics

codifying problem and metrics ▶ data collection and cleaning ▶ feature engineering ▶ model training and tuning ▶ model validation ▶ model deployment ▶ monitoring, validation

codifying problem and metrics

codifying problem and metrics ▶ data collection and cleaning ▶ feature engineering ▶ model training and tuning ▶ model validation ▶ model deployment ▶ monitoring, validation

codifying problem and metrics



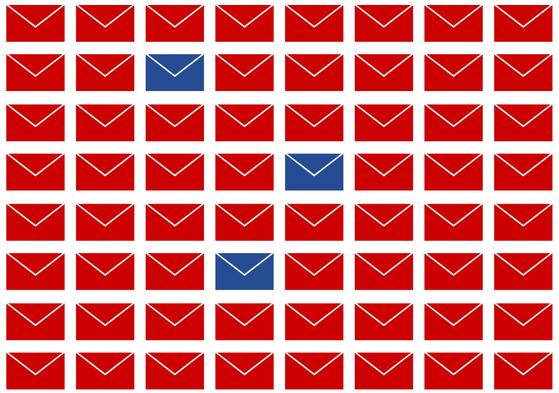
data collection and cleaning

codifying problem and metrics ▶ data collection and cleaning ▶ feature engineering ▶ model training and tuning ▶ model validation ▶ model deployment ▶ monitoring, validation

data collection and cleaning

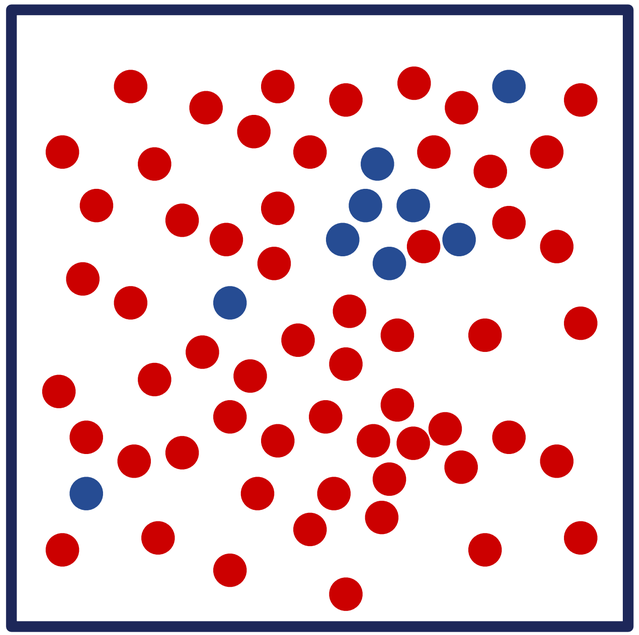
Impedit quo id dolorum debitis qui omnis. Et ipsa animi ab ipsa blanditiis consequatur. Est consequatur cumque minima nesciunt sint. Illum rerum minus odit qui.

In quia excepturi adipisci. Maxime aut est libero atque quod. Voluptatum quae quos occaecati expedita qui impedit sunt nisi. Qui quod eligendi provident.



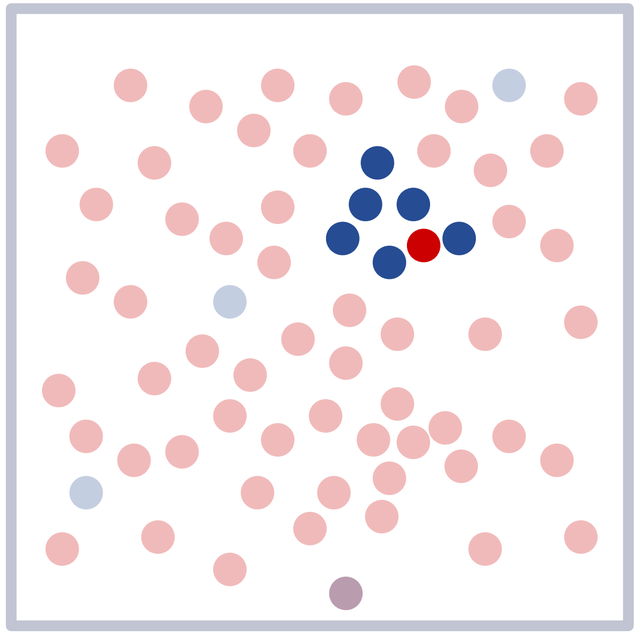
codifying problem and metrics ▶ data collection and cleaning ▶ feature engineering ▶ model training and tuning ▶ model validation ▶ model deployment ▶ monitoring, validation

feature engineering



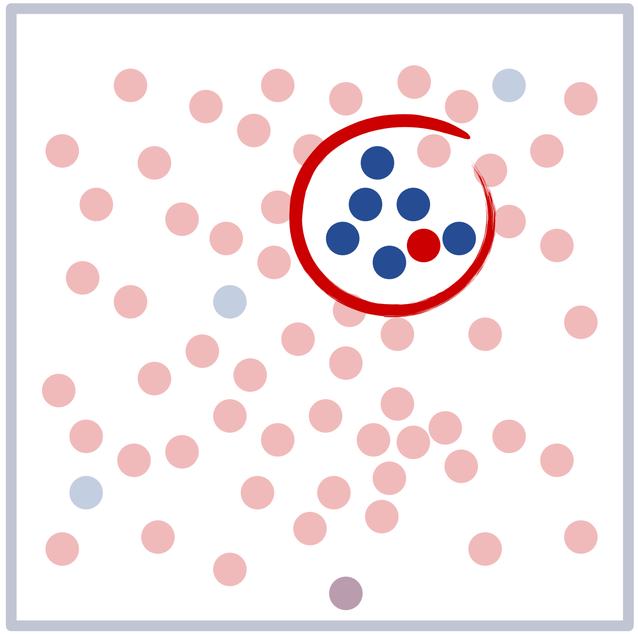
codifying problem and metrics ▶ data collection and cleaning ▶ feature engineering ▶ model training and tuning ▶ model validation ▶ model deployment ▶ monitoring, validation

feature engineering



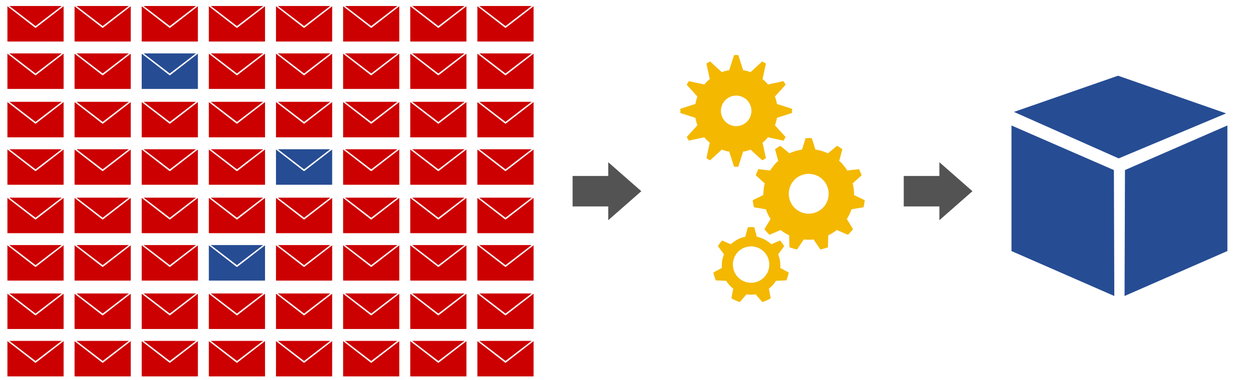
codifying problem and metrics ▶ data collection and cleaning ▶ feature engineering ▶ model training and tuning ▶ model validation ▶ model deployment ▶ monitoring, validation

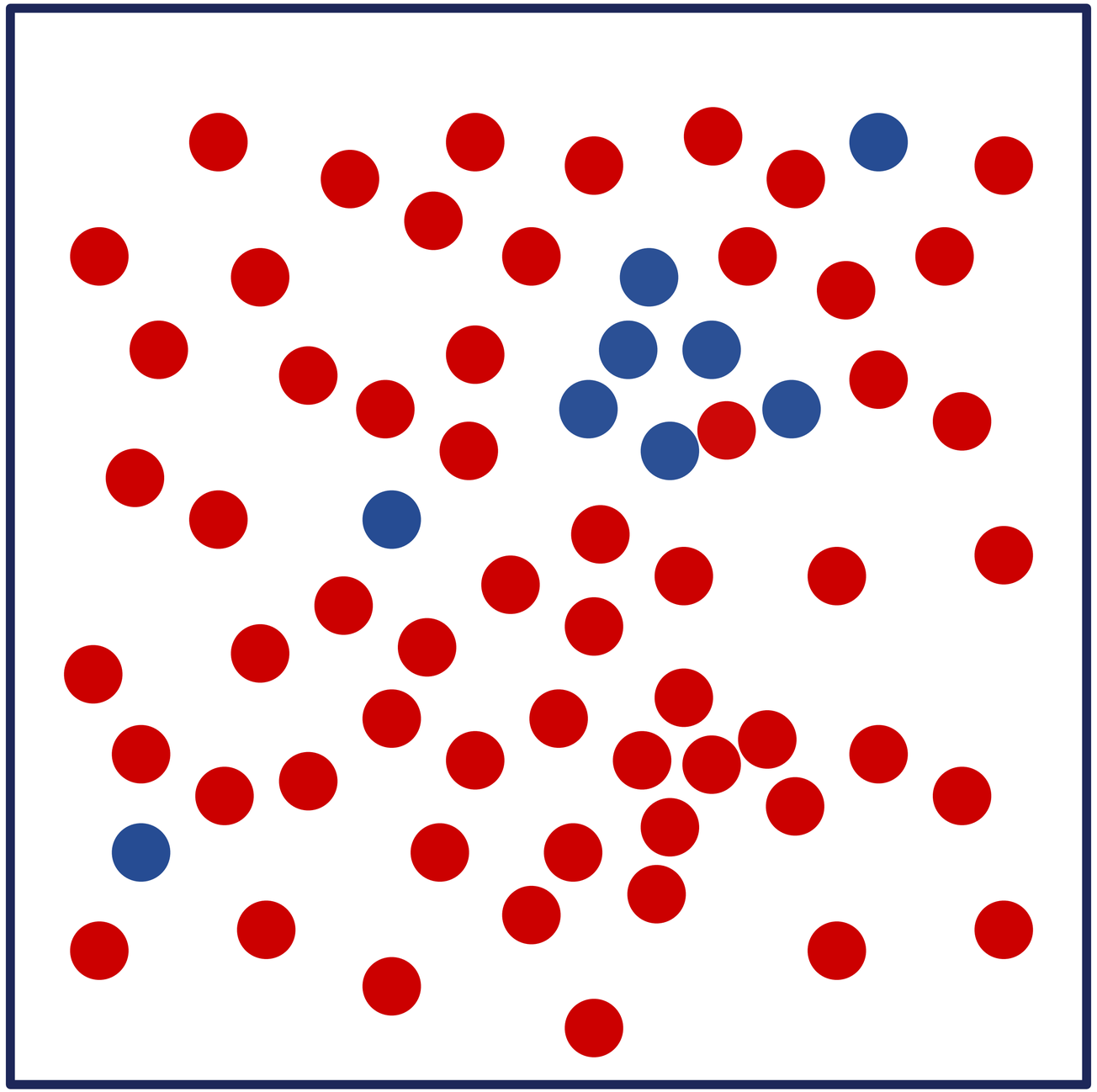
feature engineering



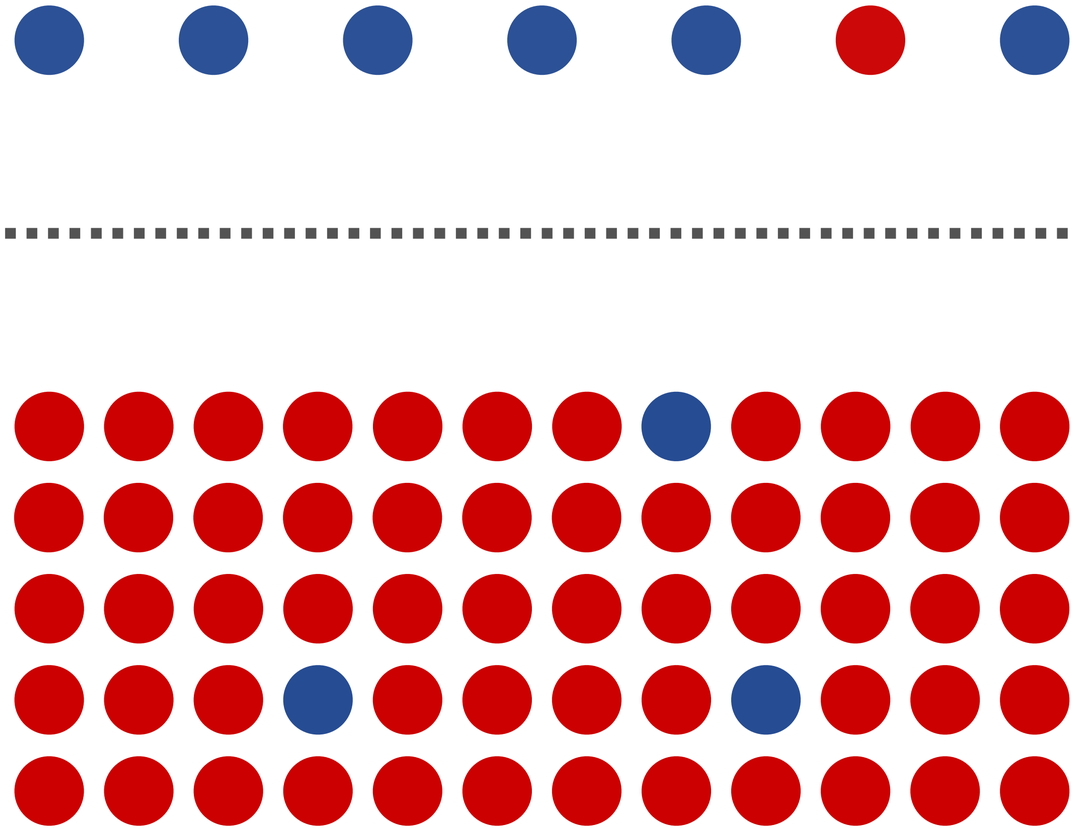
codifying problem and metrics ▶ data collection and cleaning ▶ feature engineering ▶ model training and tuning ▶ model validation ▶ model deployment ▶ monitoring, validation

model training and tuning



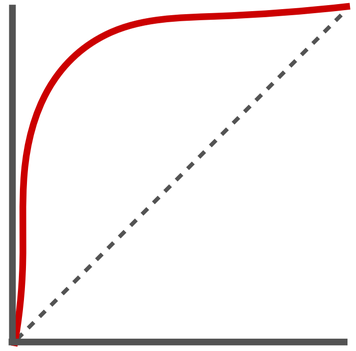


Red Hat



codifying problem and metrics ▶ data collection and cleaning ▶ feature engineering ▶ model training and tuning ▶ model validation ▶ model deployment ▶ monitoring, validation

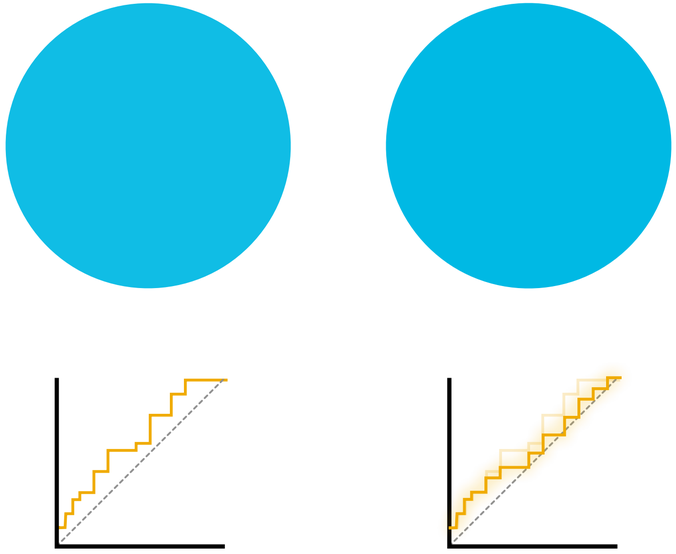
model validation



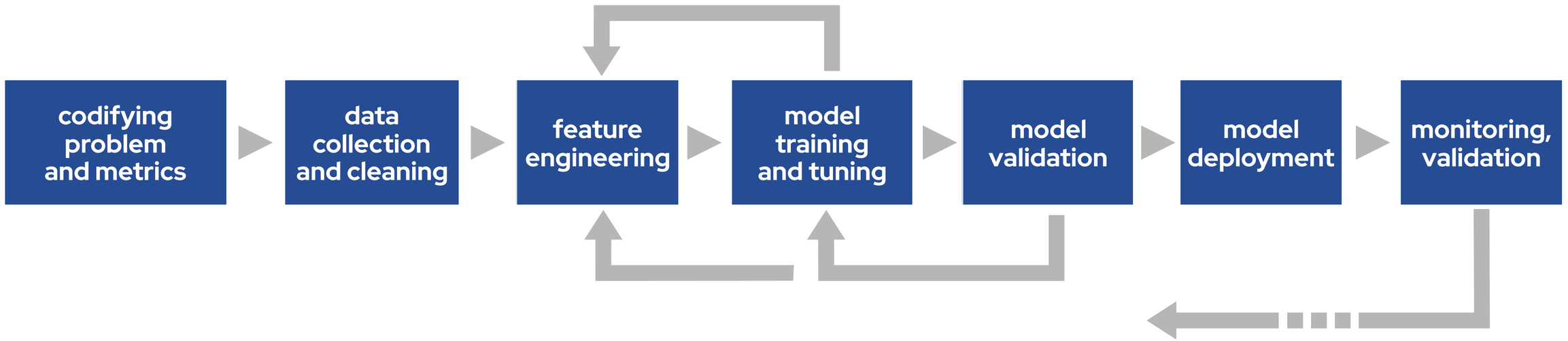
codifying problem and metrics ▶ data collection and cleaning ▶ feature engineering ▶ model training and tuning ▶ model validation ▶ model deployment ▶ monitoring, validation

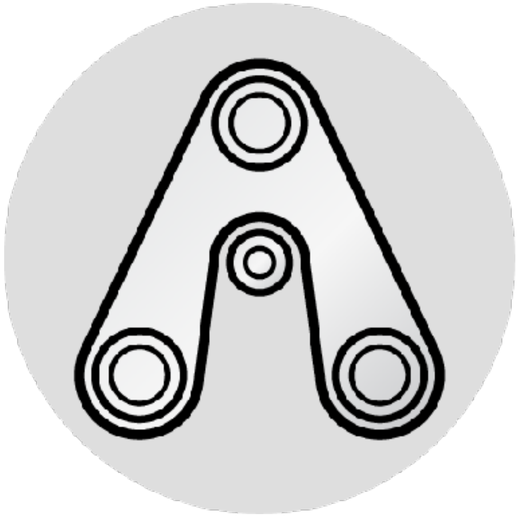
model deployment

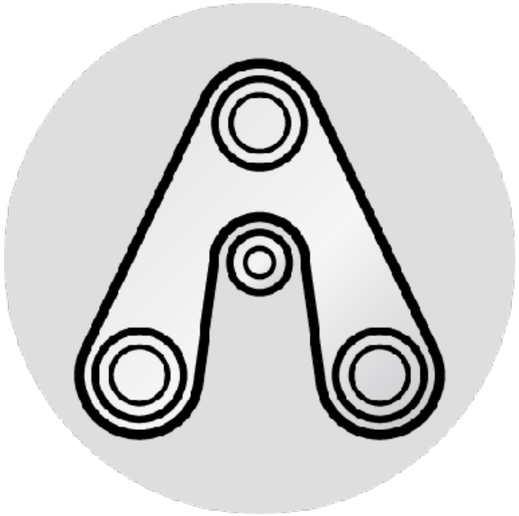
monitoring, validation

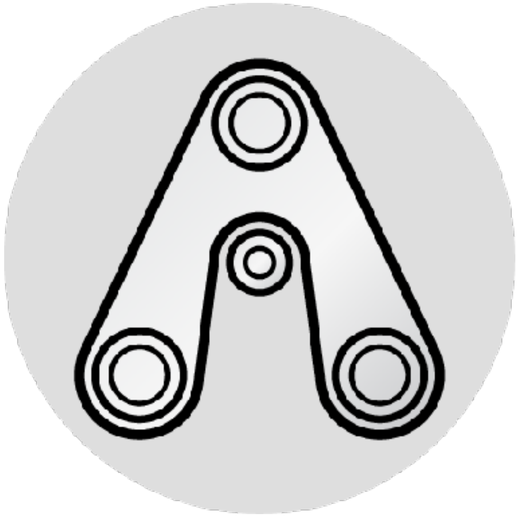


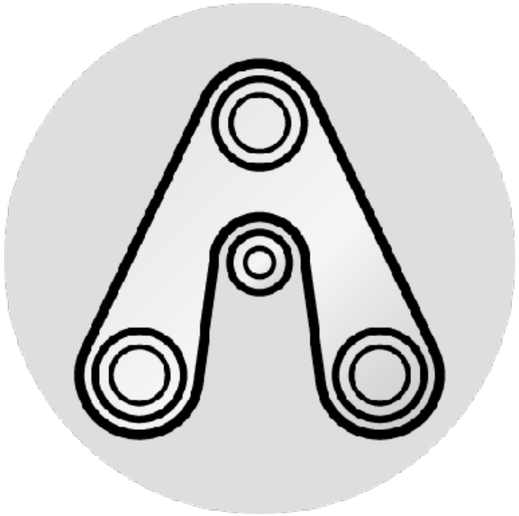

codifying problem and metrics
data collection and cleaning
feature engineering
model training and tuning
model validation
model deployment
monitoring, validation


codifying problem and metrics
data collection and cleaning
feature engineering
model training and tuning
model validation
model deployment
monitoring, validation



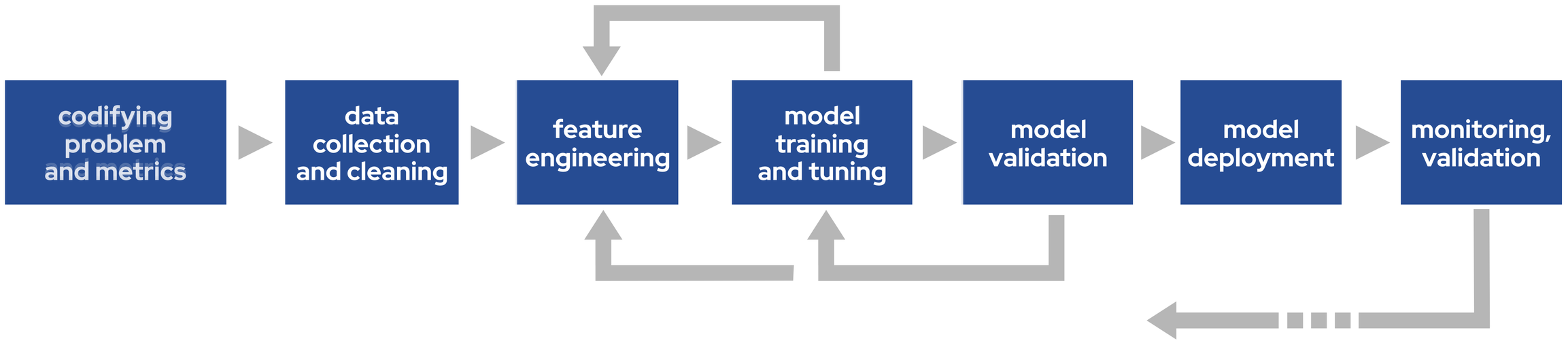




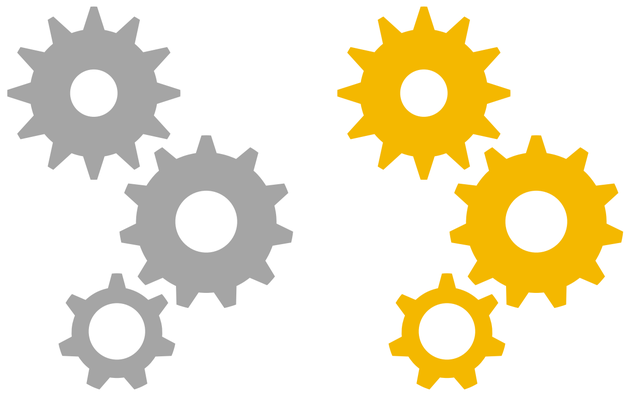



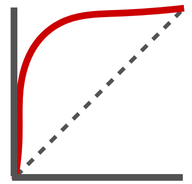
data as the foundation

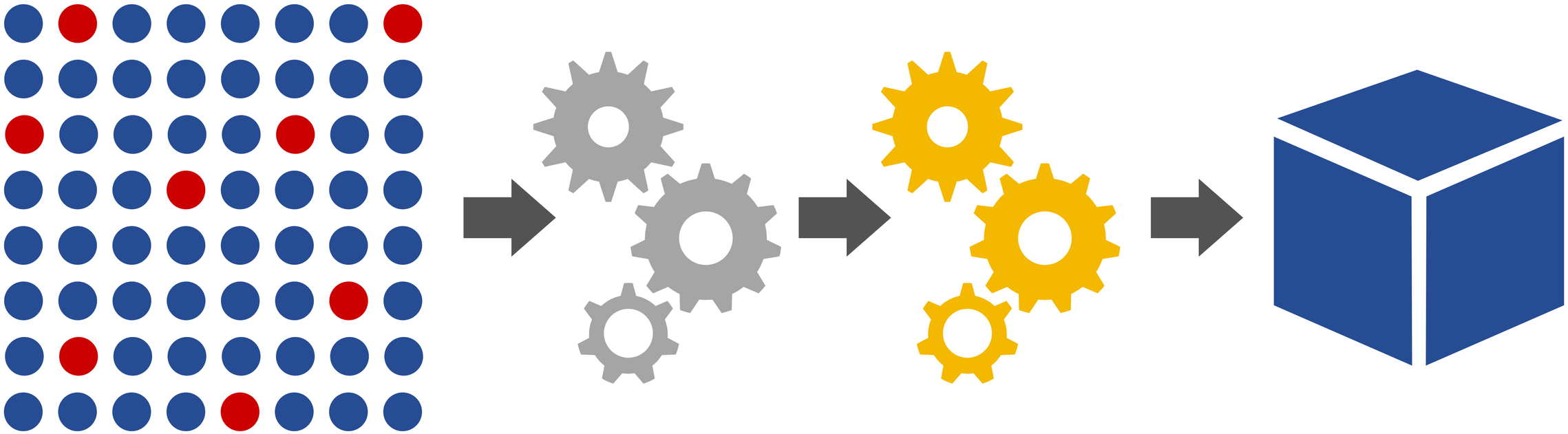
Why “on Kubernetes?”

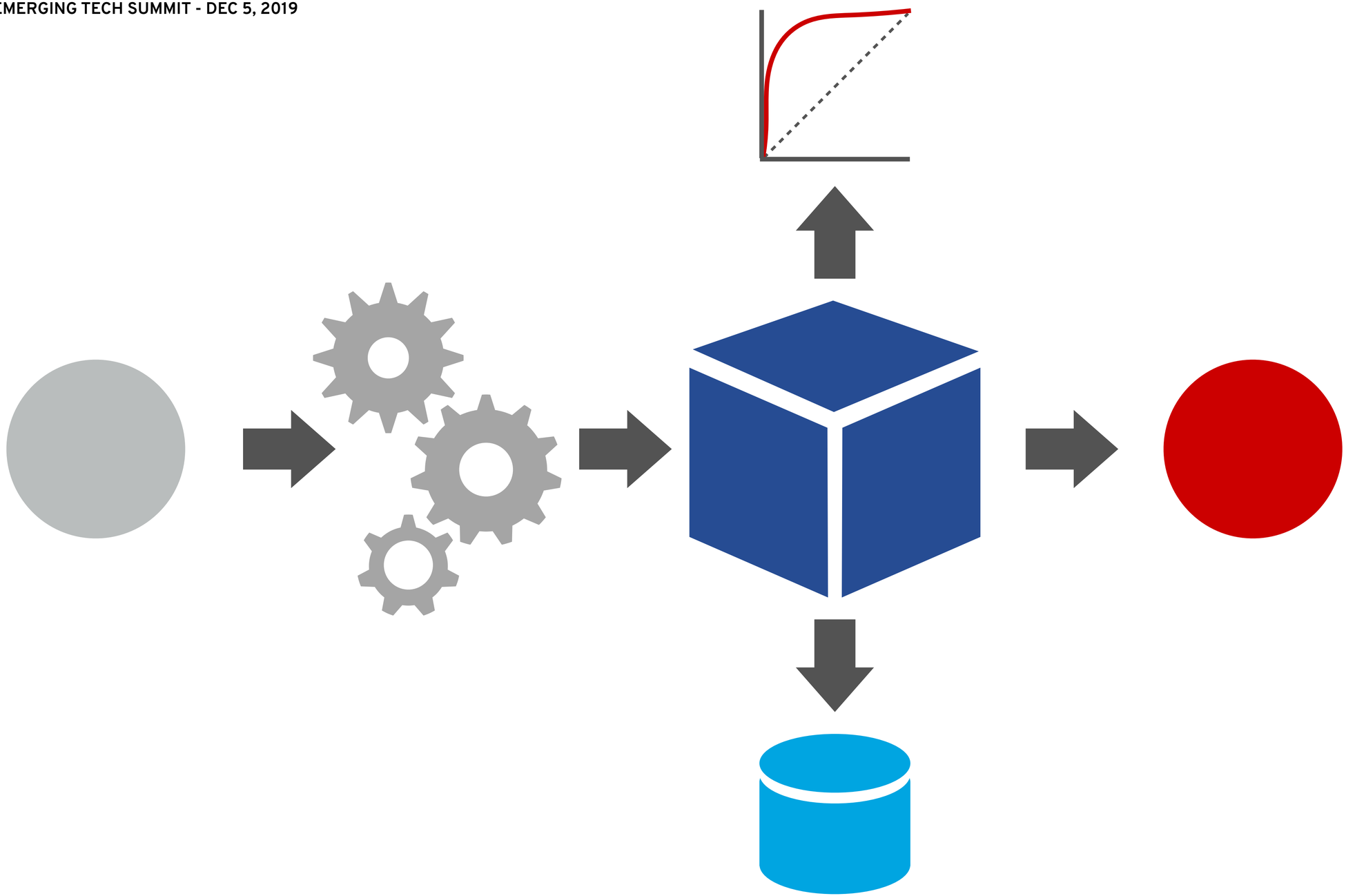

codifying problem and metrics
data collection and cleaning
feature engineering
model training and tuning
model validation
model deployment
monitoring, validation

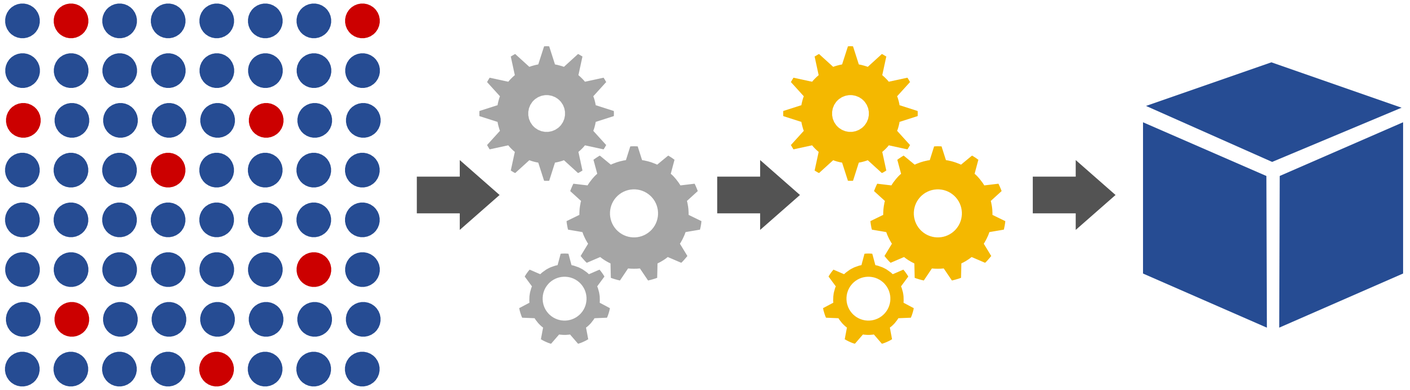
codifying problem and metrics ▶ data collection and cleaning ▶ **feature engineering** ▶ **model training and tuning** ▶ model validation ▶ **model deployment** ▶ **monitoring, validation**

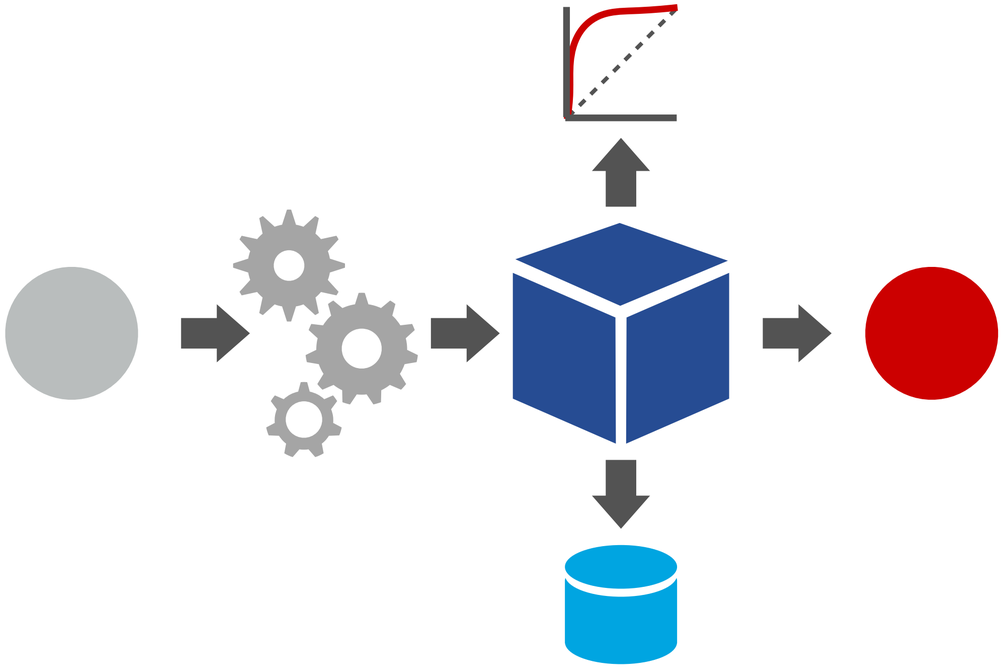


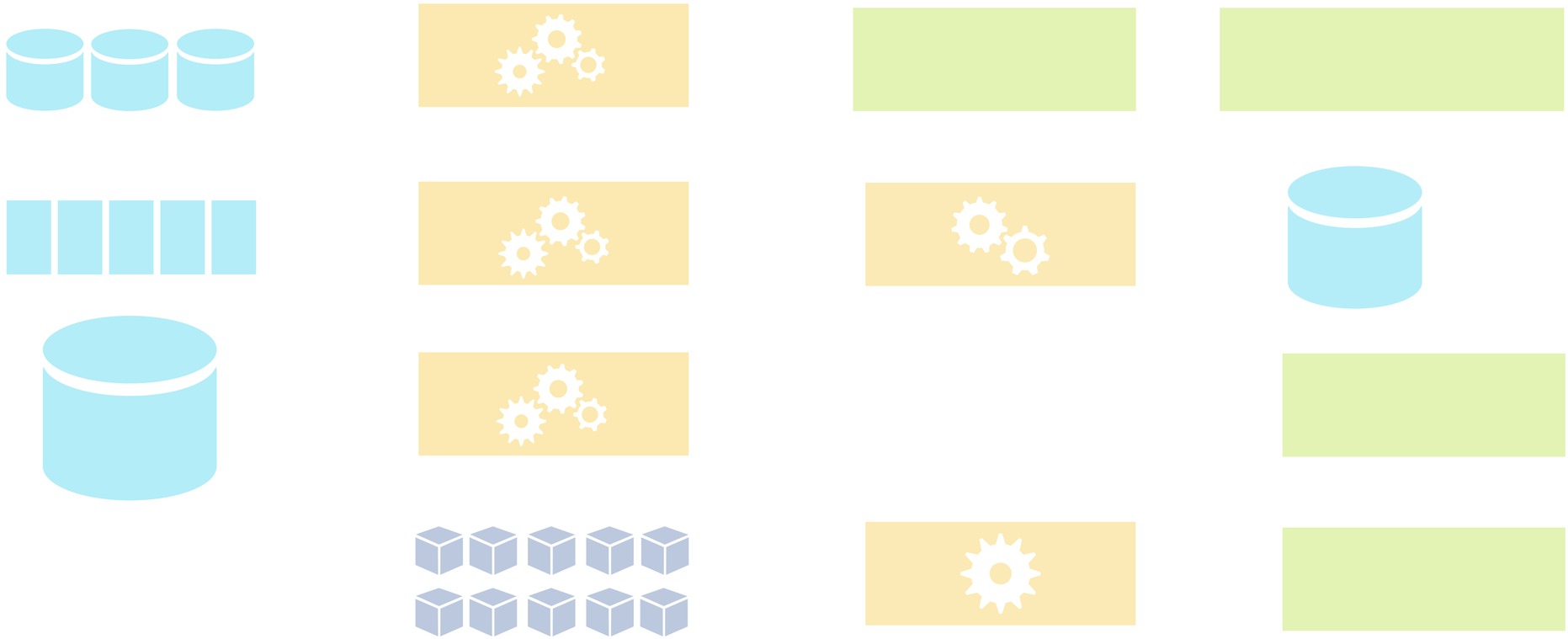


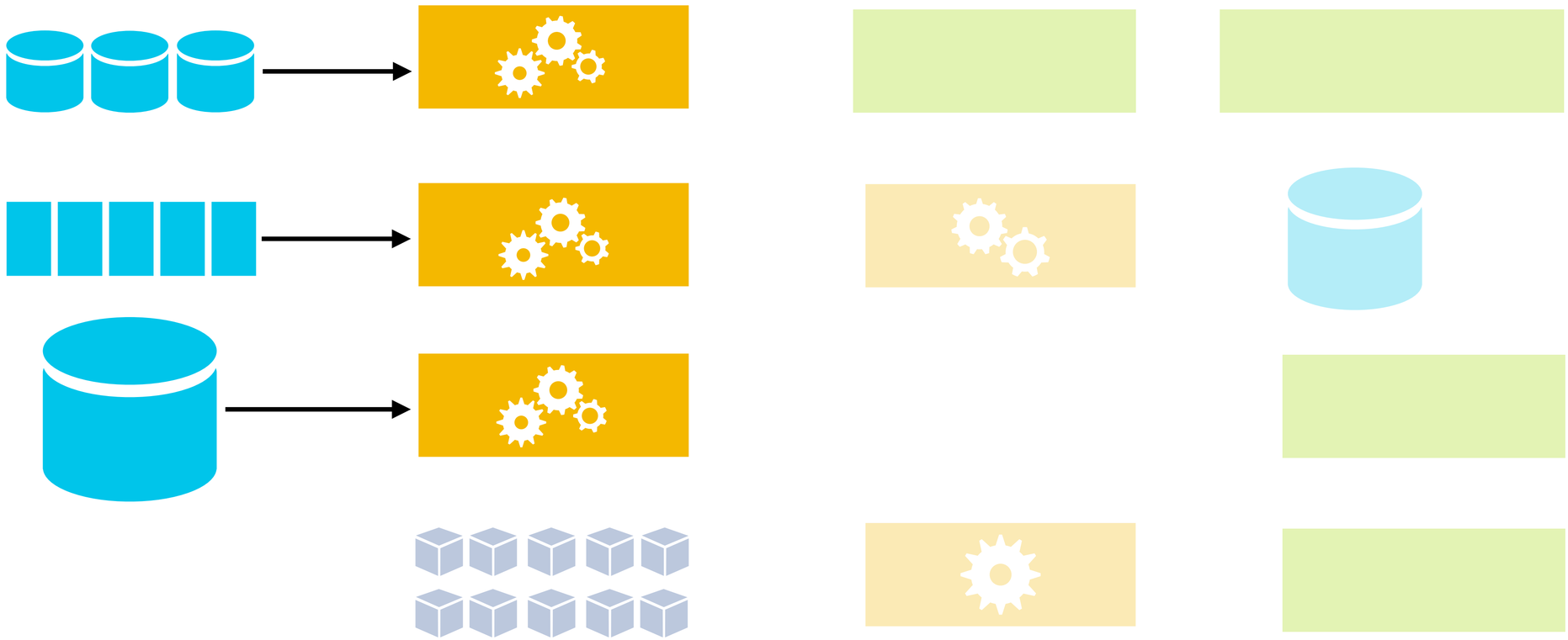


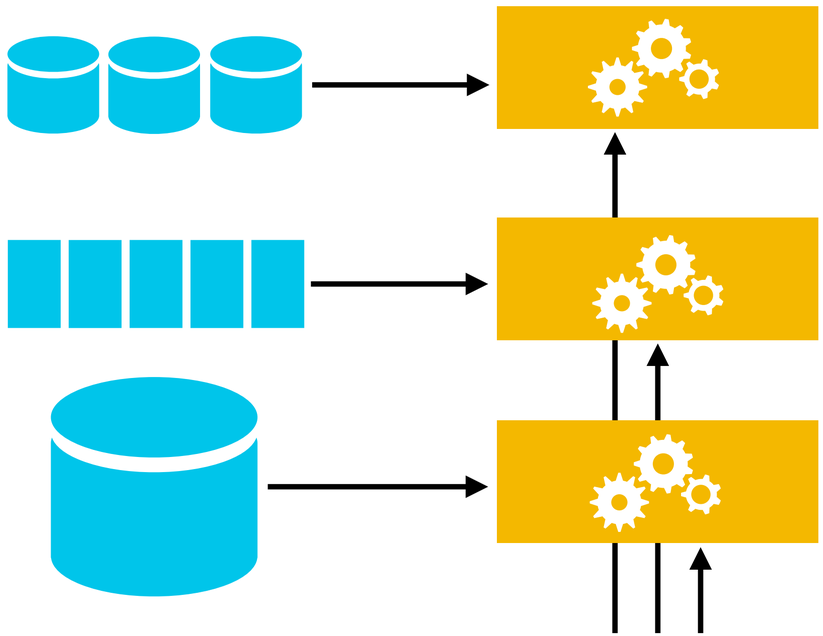


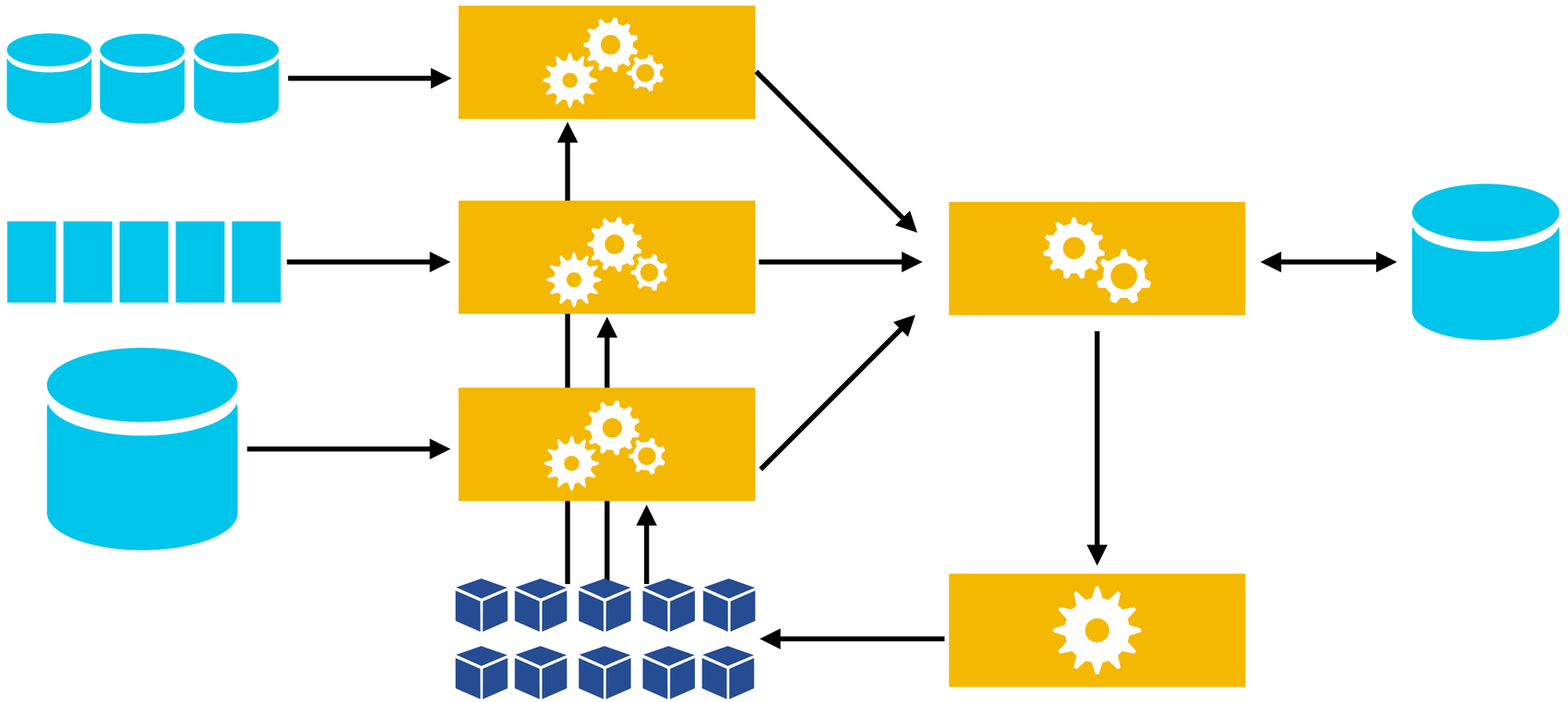


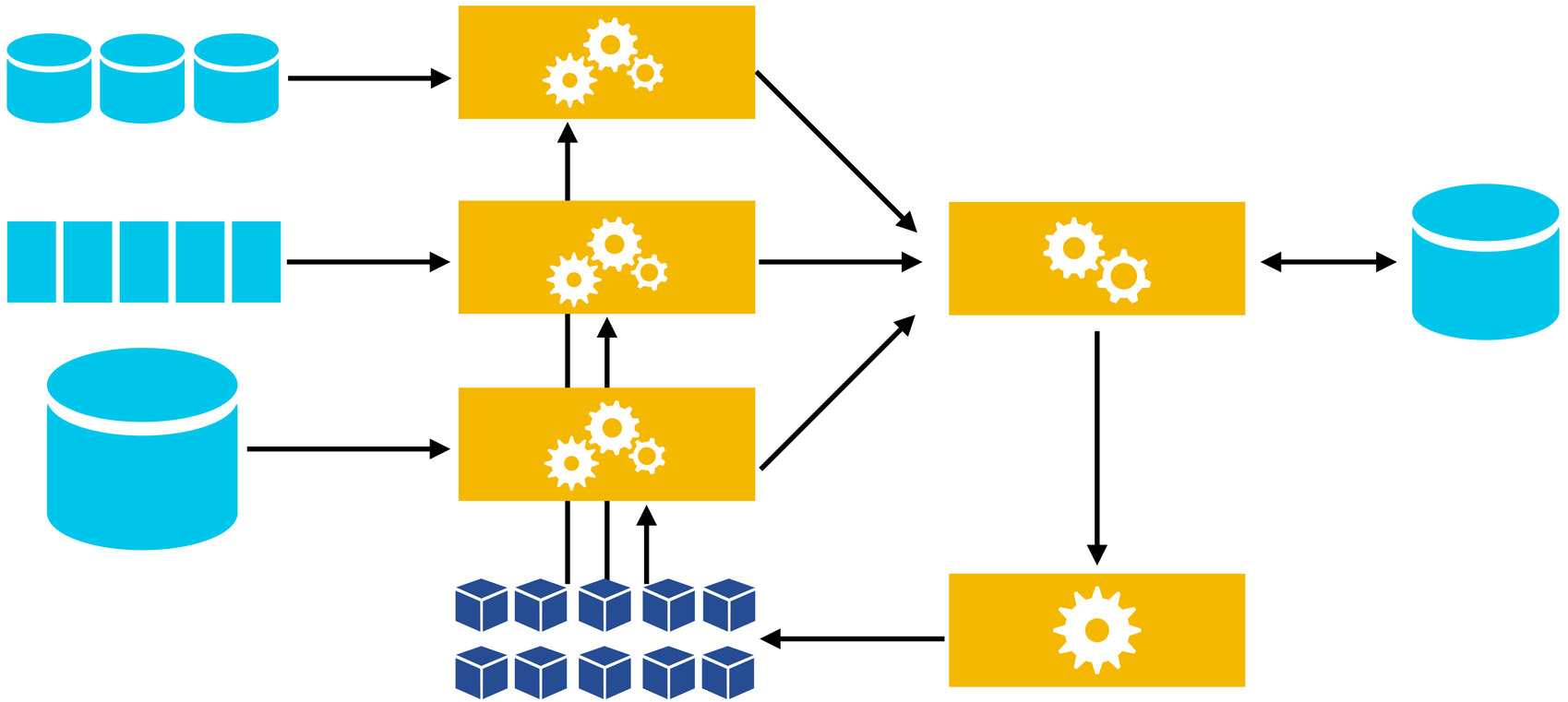


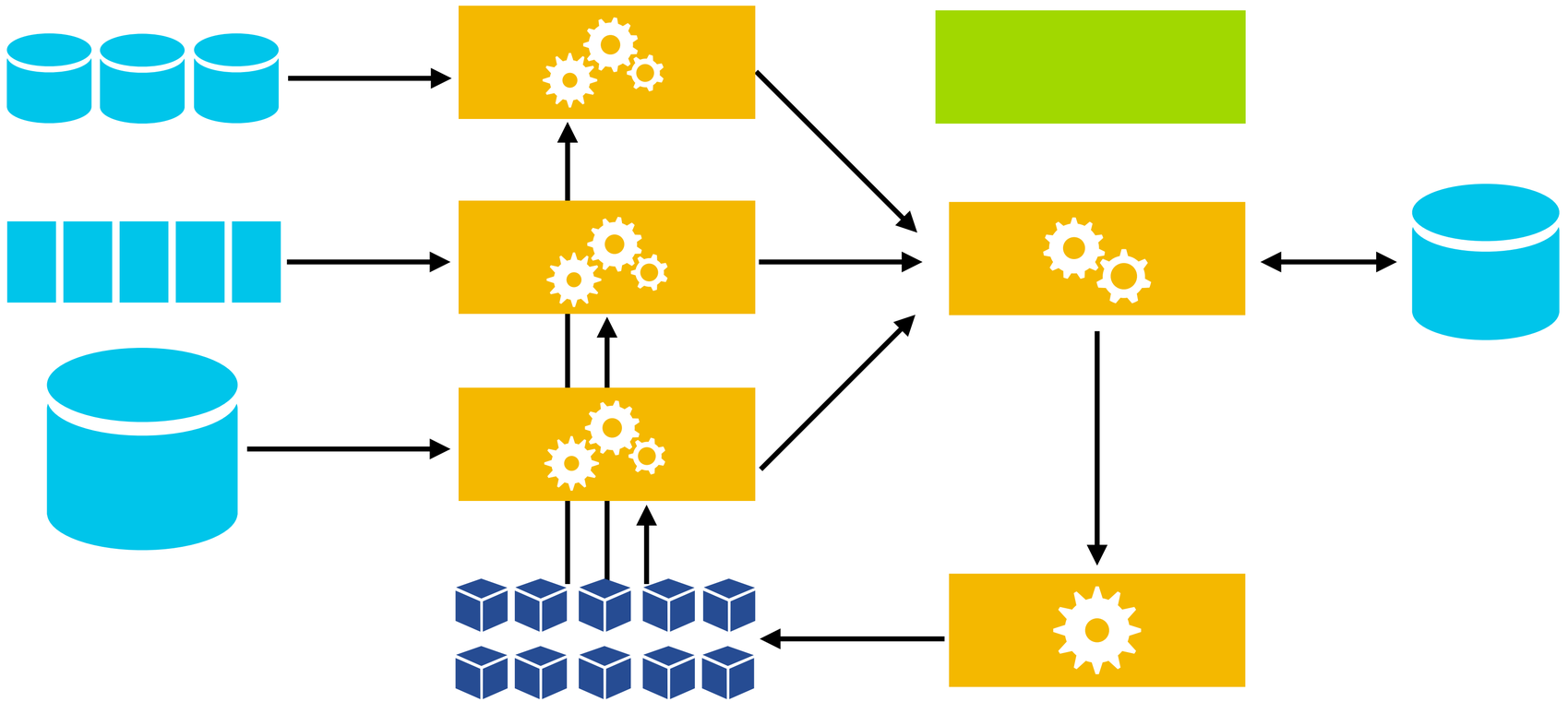


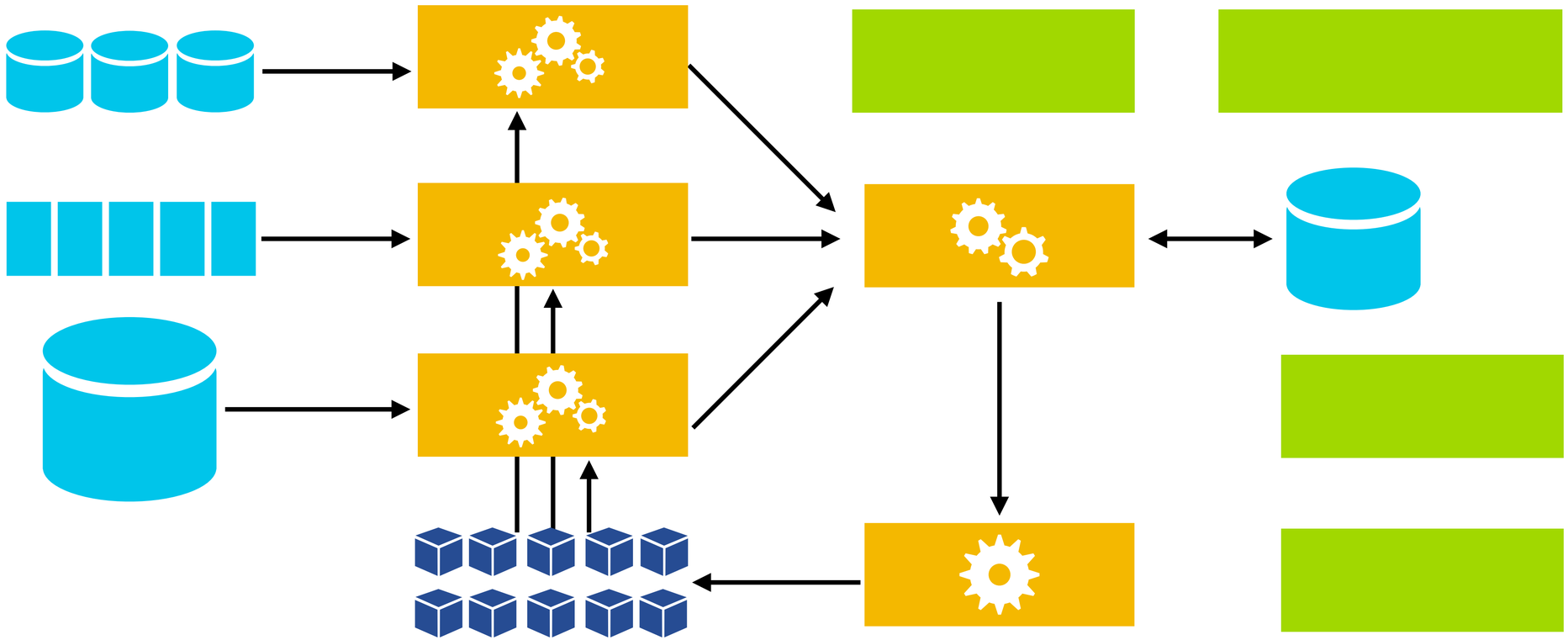


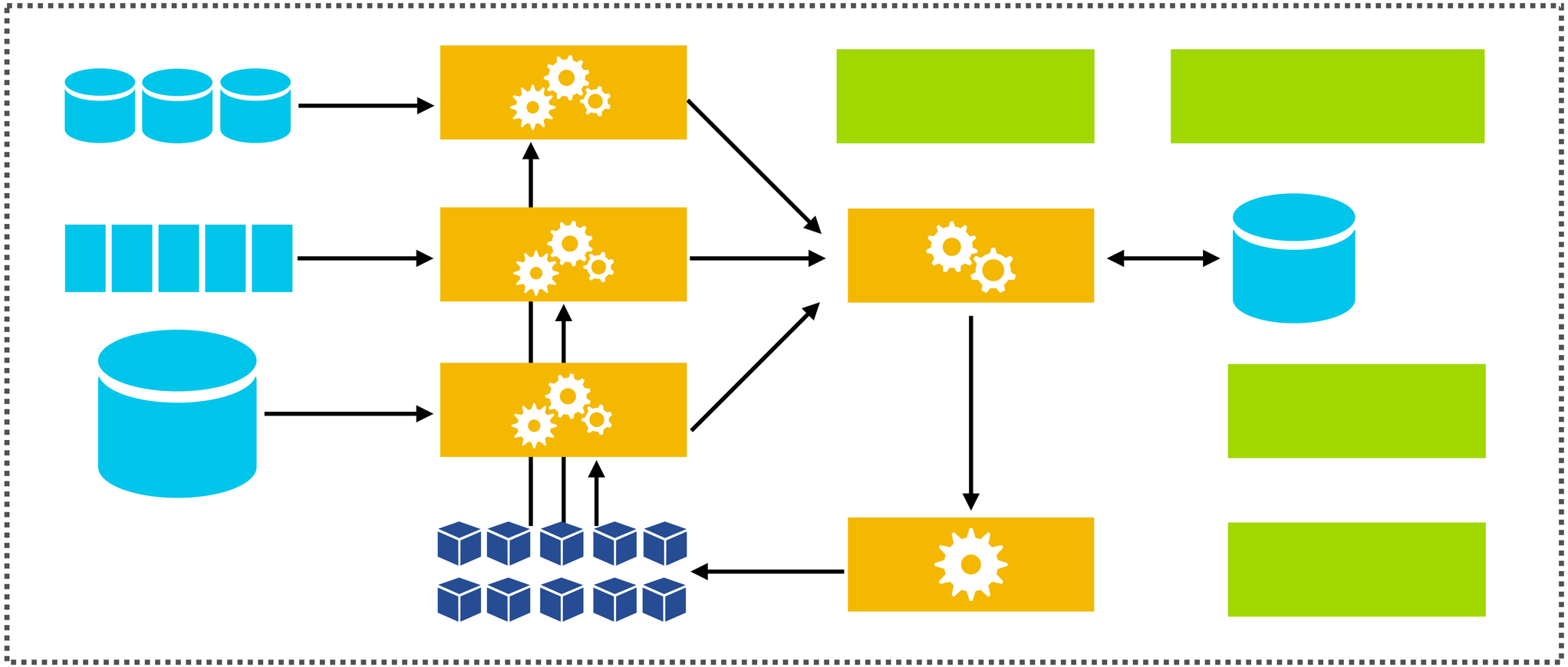












OpenShift is **enterprise Kubernetes** with a **great developer experience**.

declarative deployments;
resource management
for apps and compute

OpenShift is **enterprise Kubernetes**
with a **great developer experience**.

efficient isolation,
secure by default

declarative deployments;
resource management
for apps and compute

OpenShift is **enterprise Kubernetes**
with a **great developer experience**.

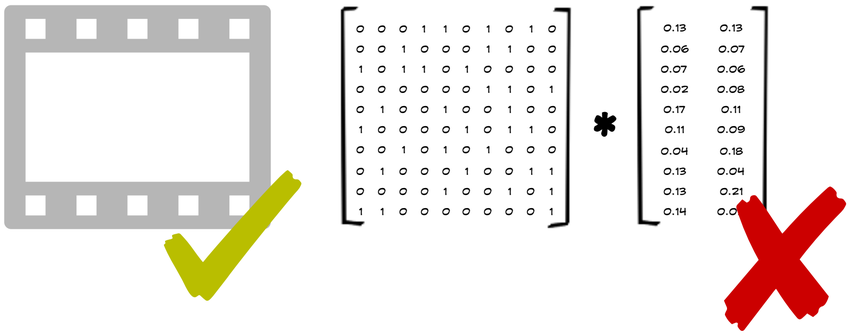
efficient isolation,
secure by default

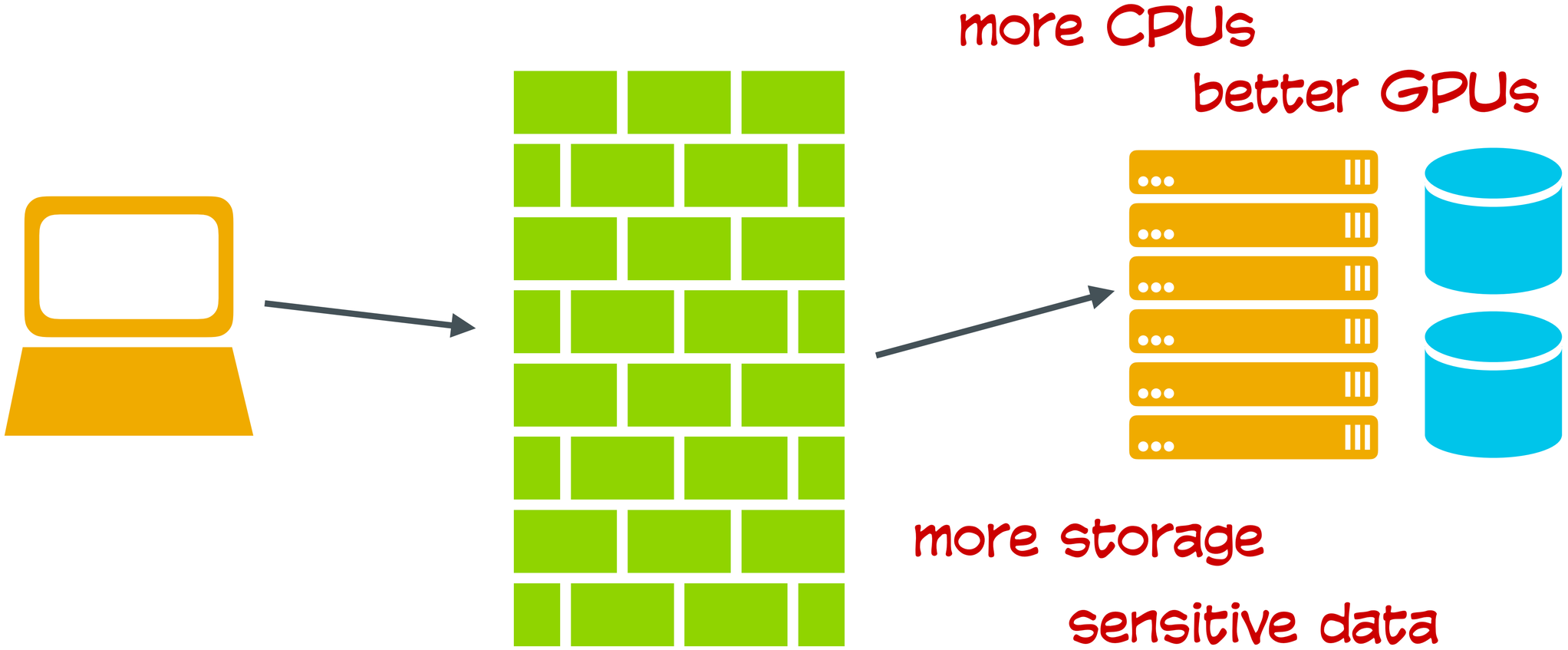
declarative deployments;
resource management
for apps and compute

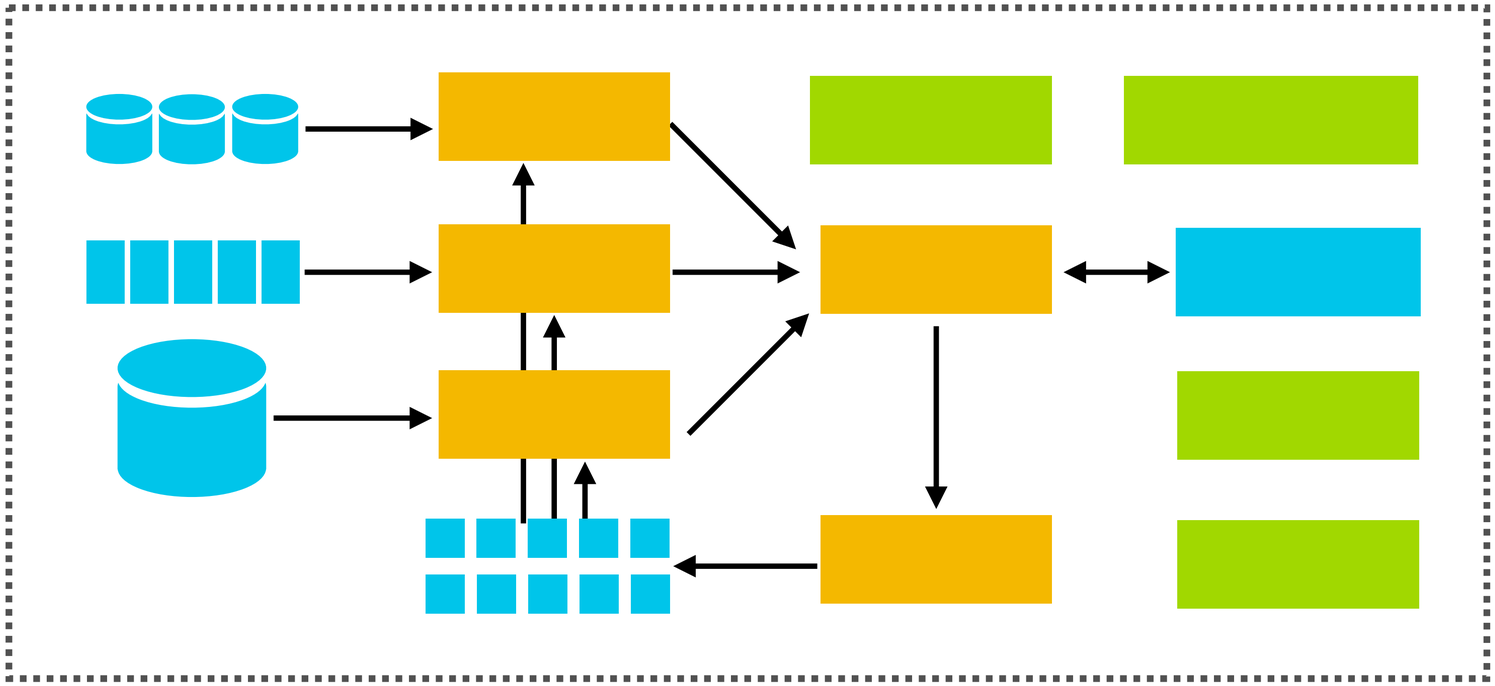
OpenShift is **enterprise Kubernetes**
with a **great developer experience**.

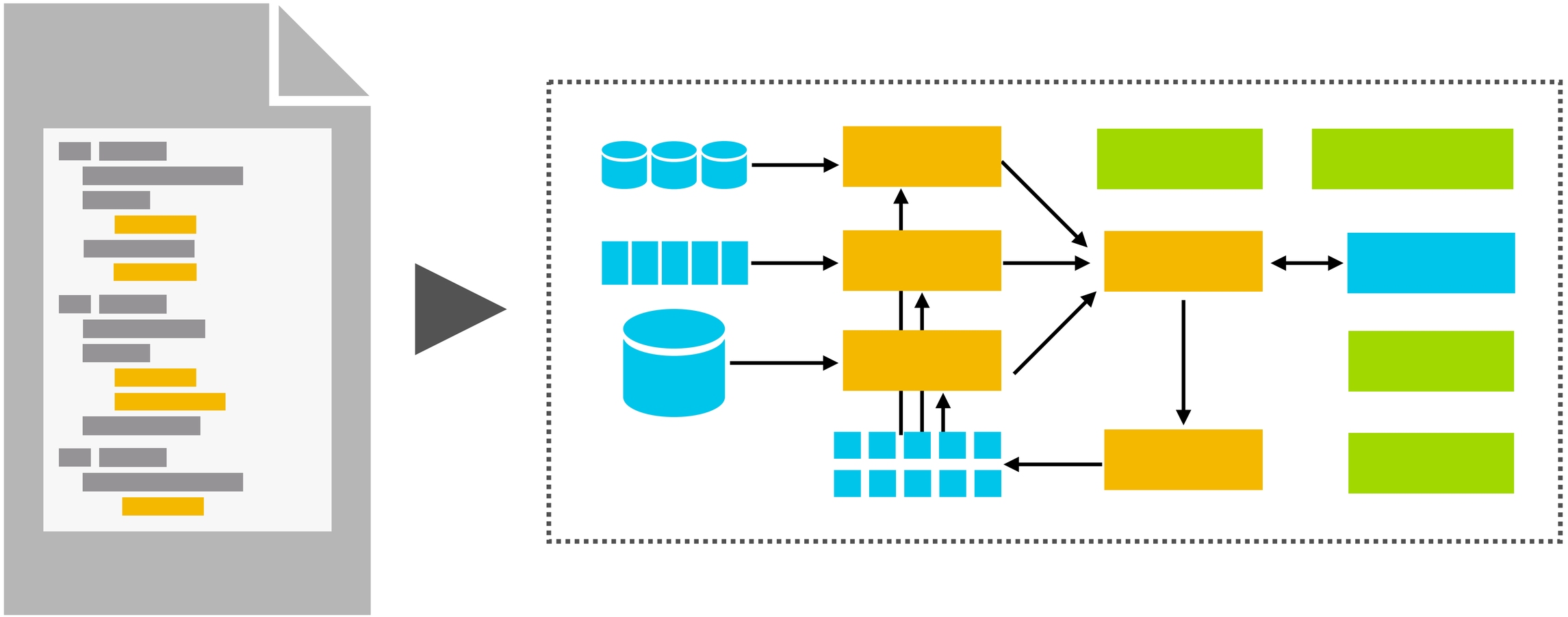
workflows to accelerate discovery

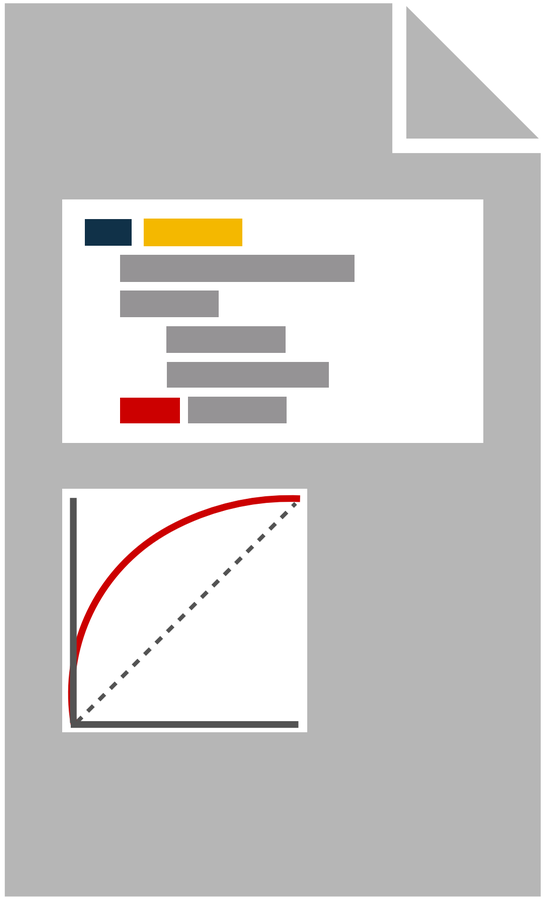
Some common concerns for AI/ML systems

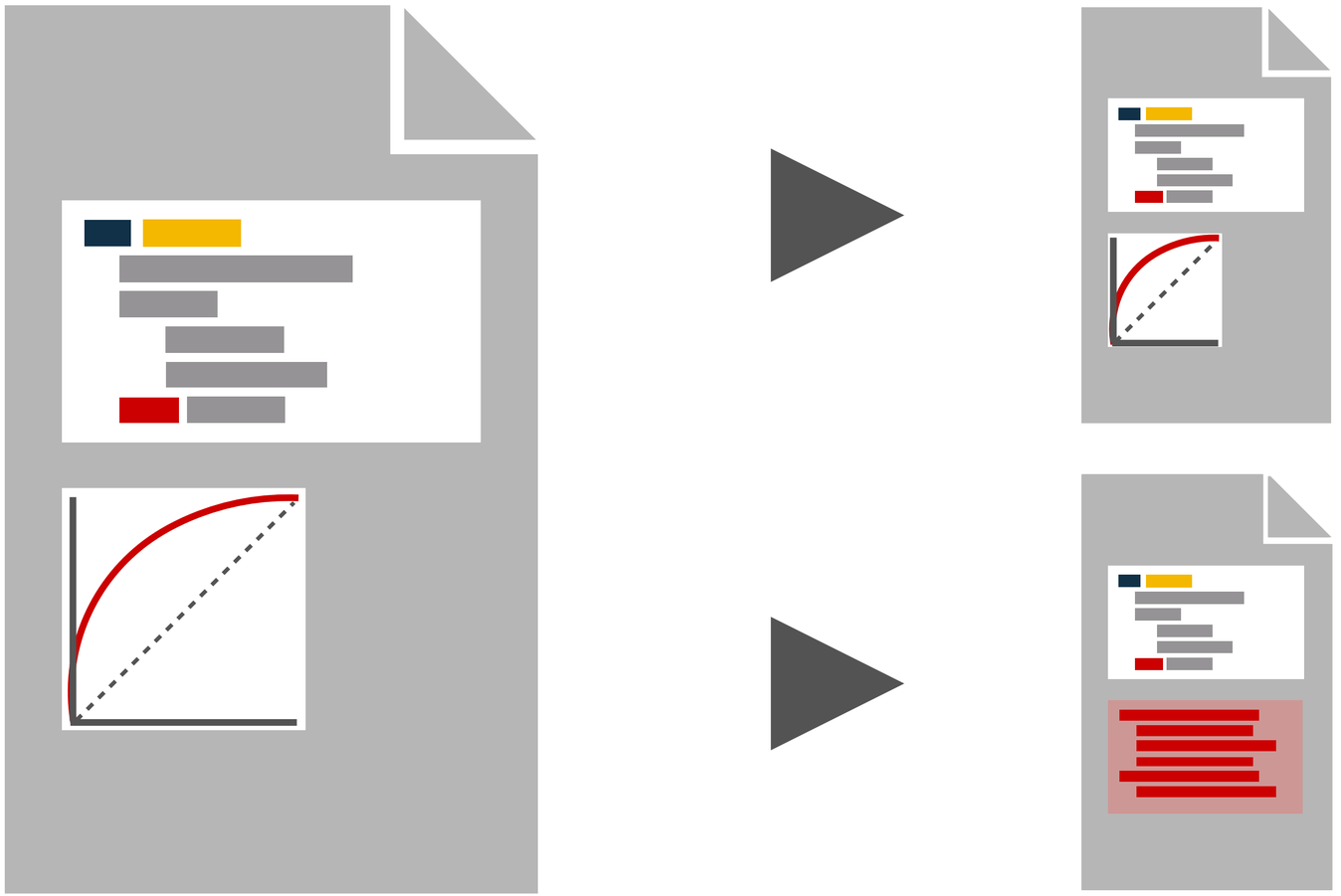

0 0 0 1 1 0 1 0 1 0
0 0 1 0 0 0 1 1 0 0
1 0 1 1 0 1 0 0 0 0
0 0 0 0 0 0 1 1 0 1
0 1 0 0 1 0 0 1 0 0
1 0 0 0 0 1 0 1 1 0
0 0 1 0 1 0 1 0 0 0
0 1 0 0 0 1 0 0 1 1
0 0 0 0 1 0 0 1 0 1
1 1 0 0 0 0 0 0 0 1
*
0.13 0.13
0.06 0.07
0.07 0.06
0.02 0.08
0.17 0.11
0.11 0.09
0.04 0.18
0.13 0.04
0.13 0.21
0.14 0.0

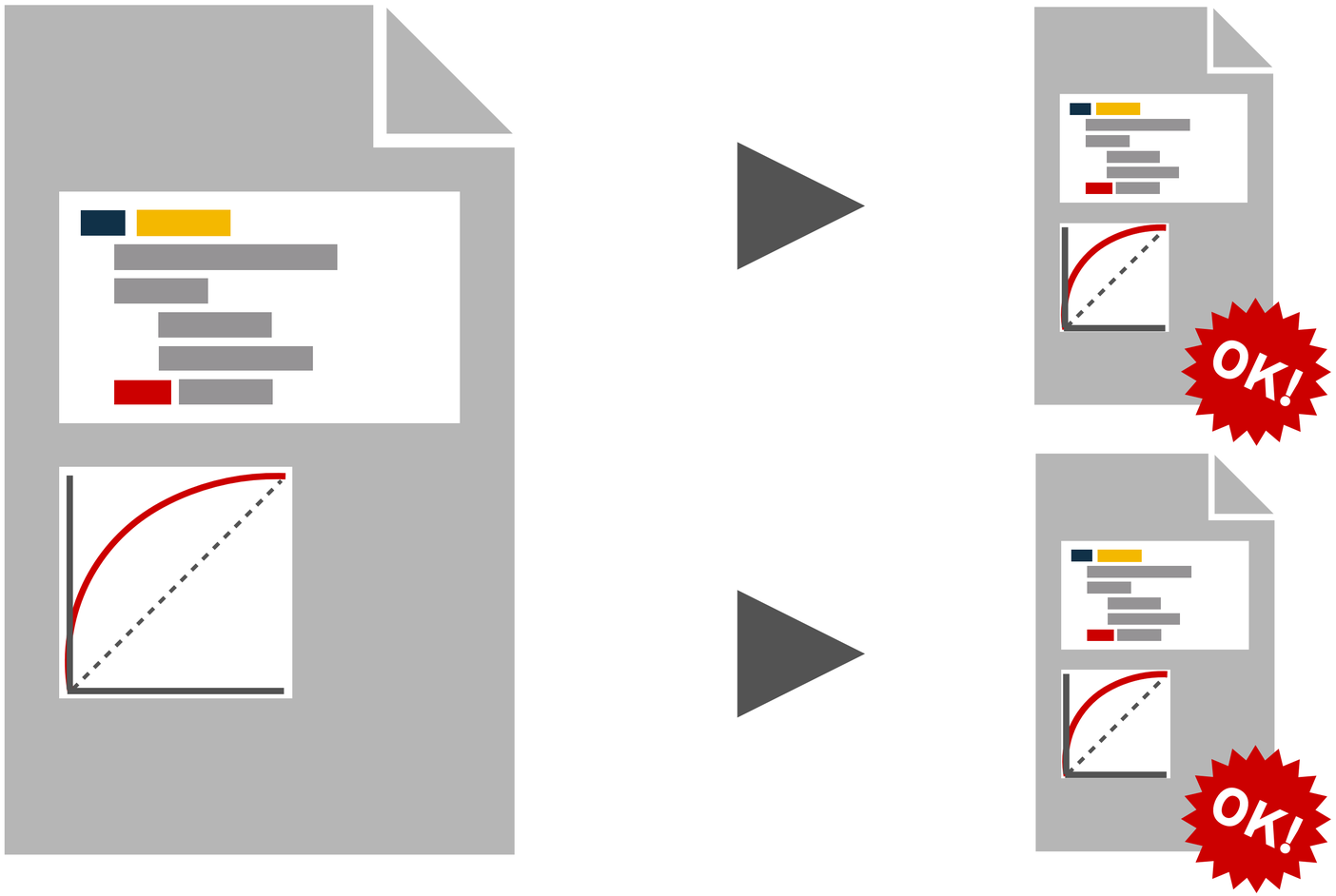

more CPUs
better GPUs
more storage
sensitive data

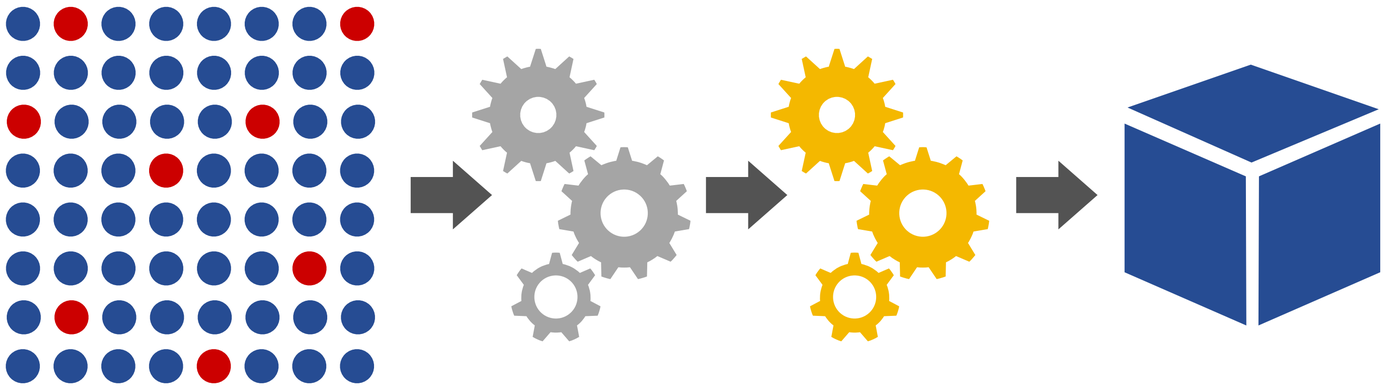


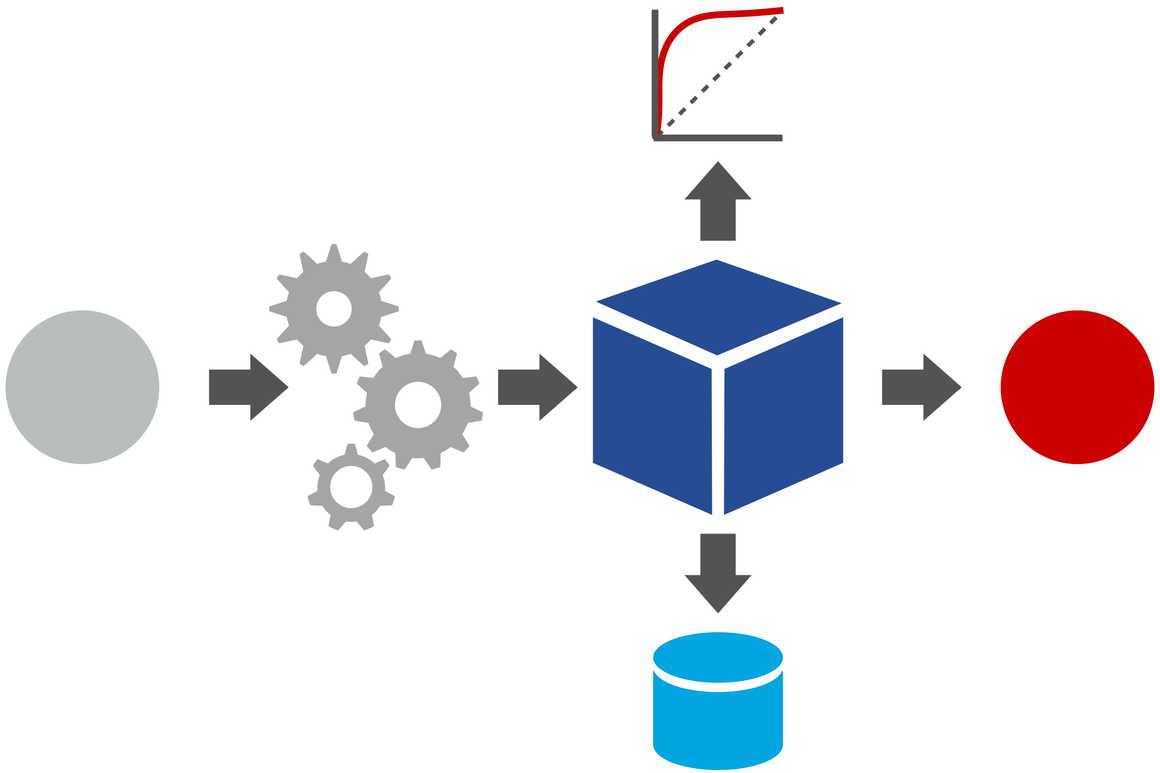


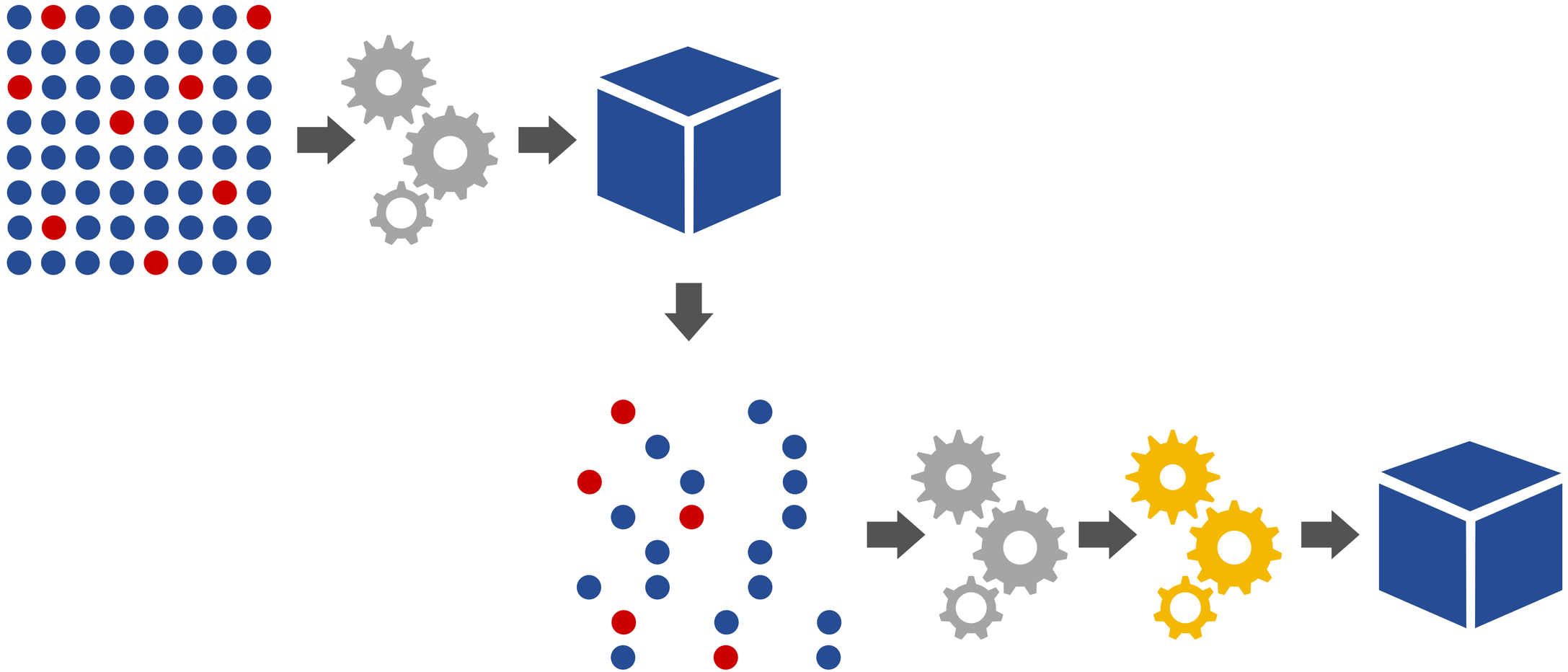


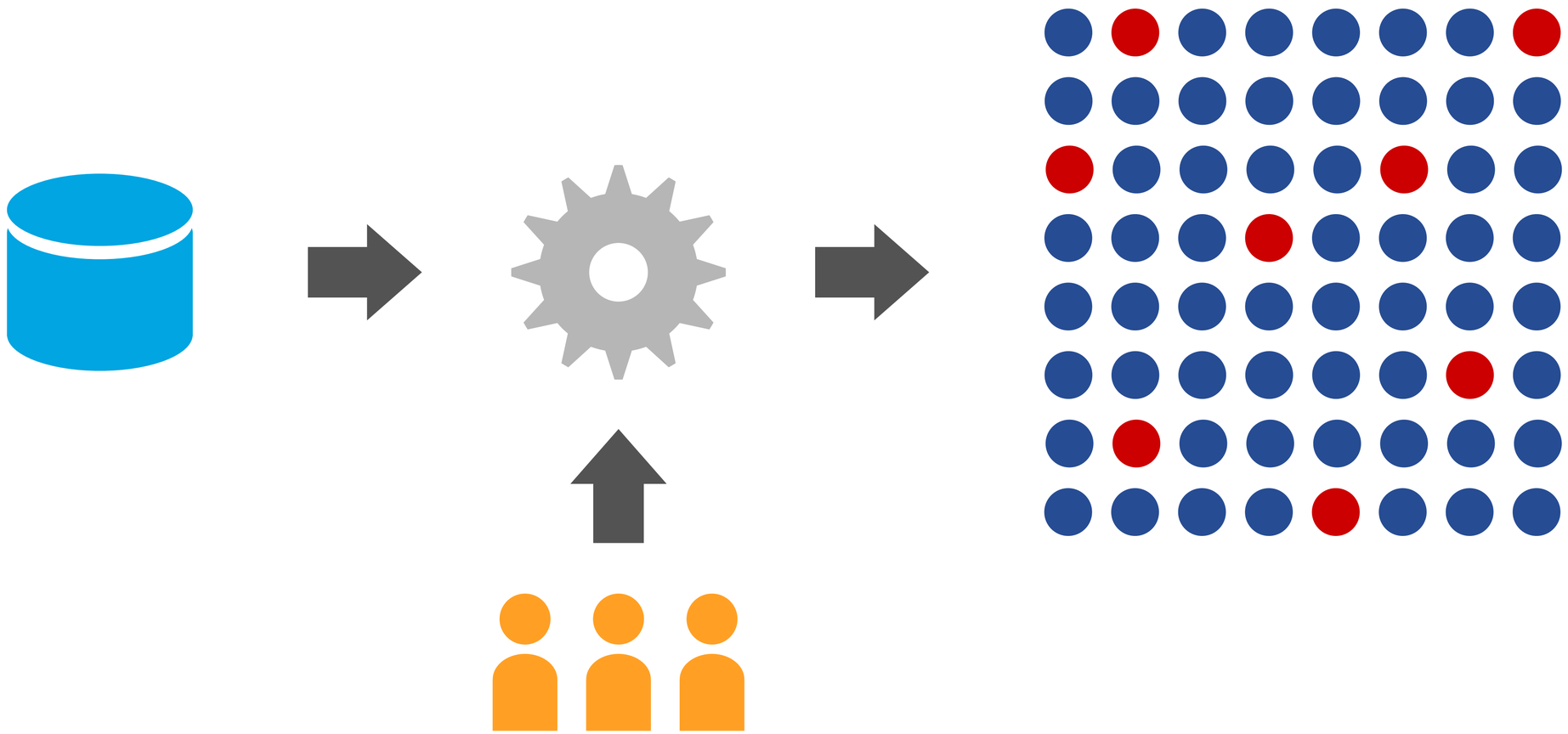


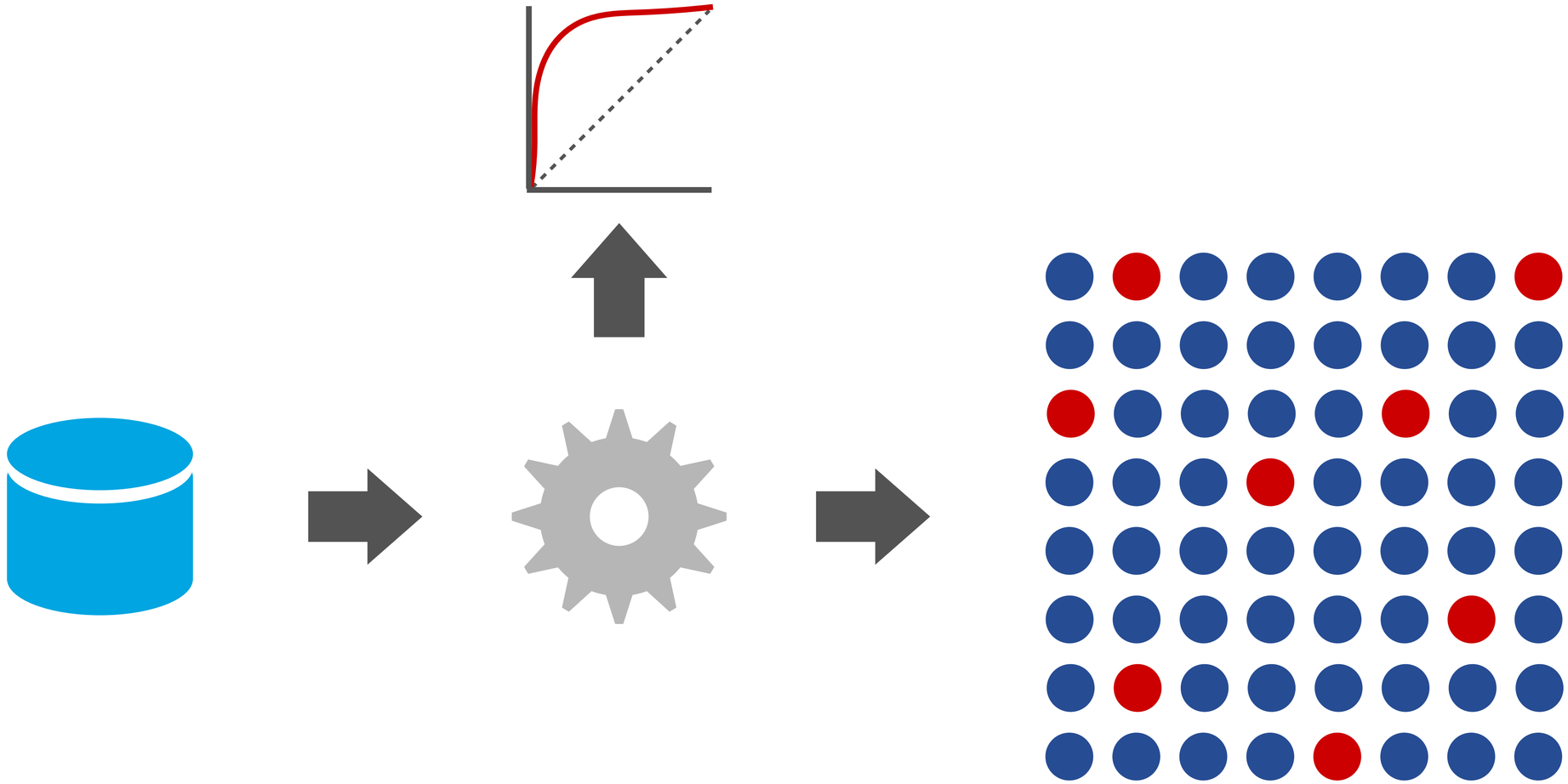

OK!
OK!

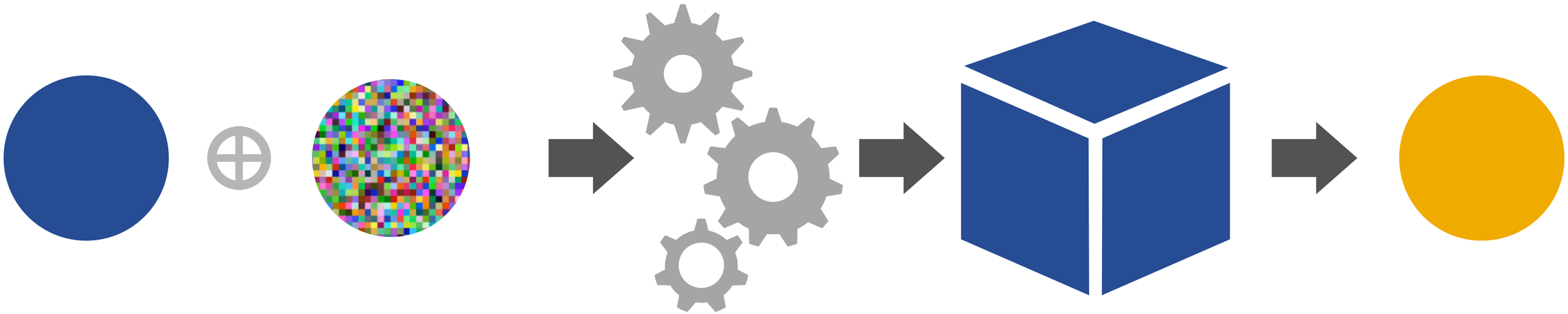


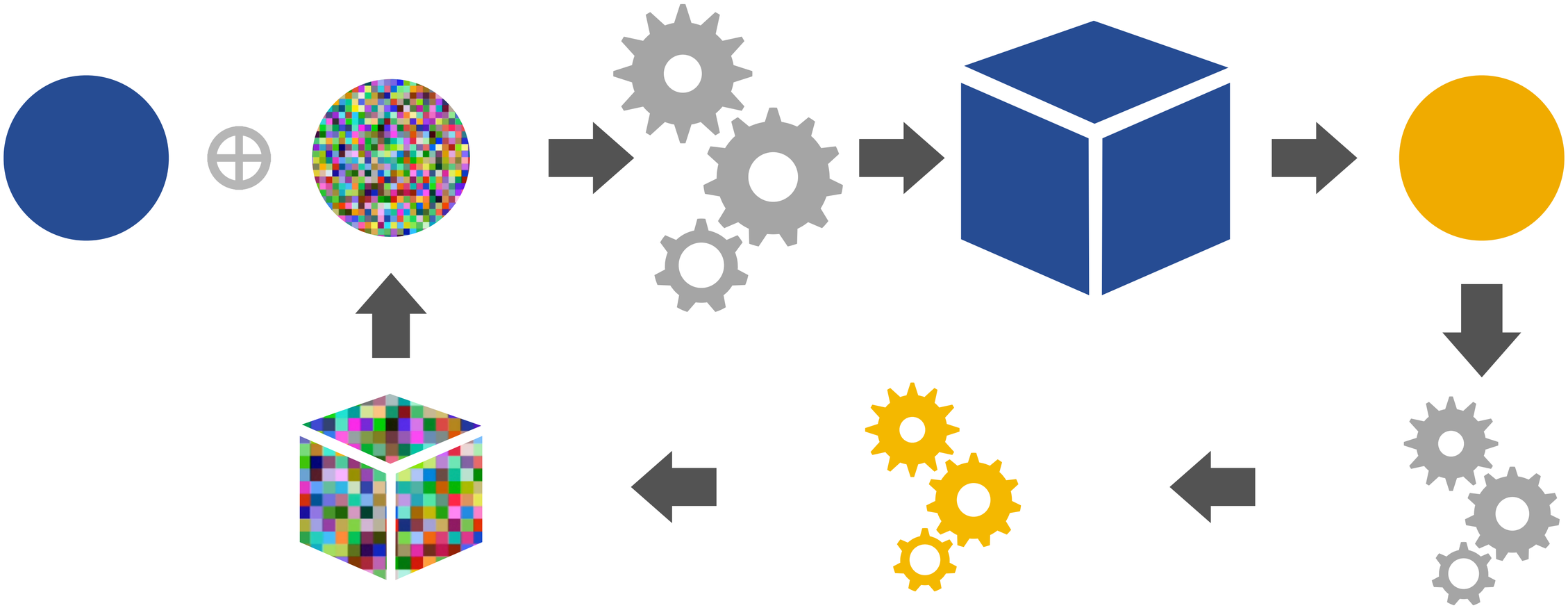


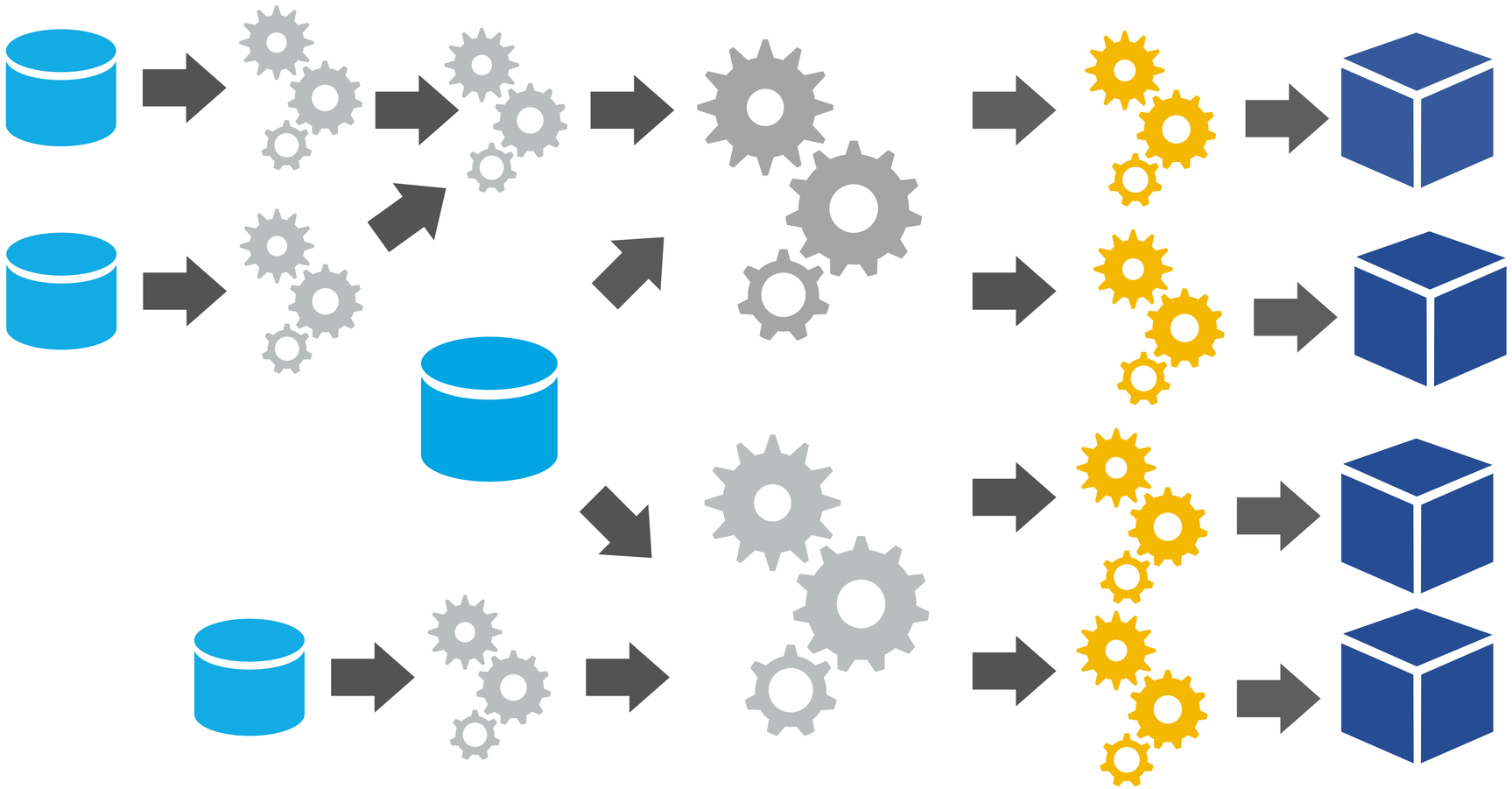


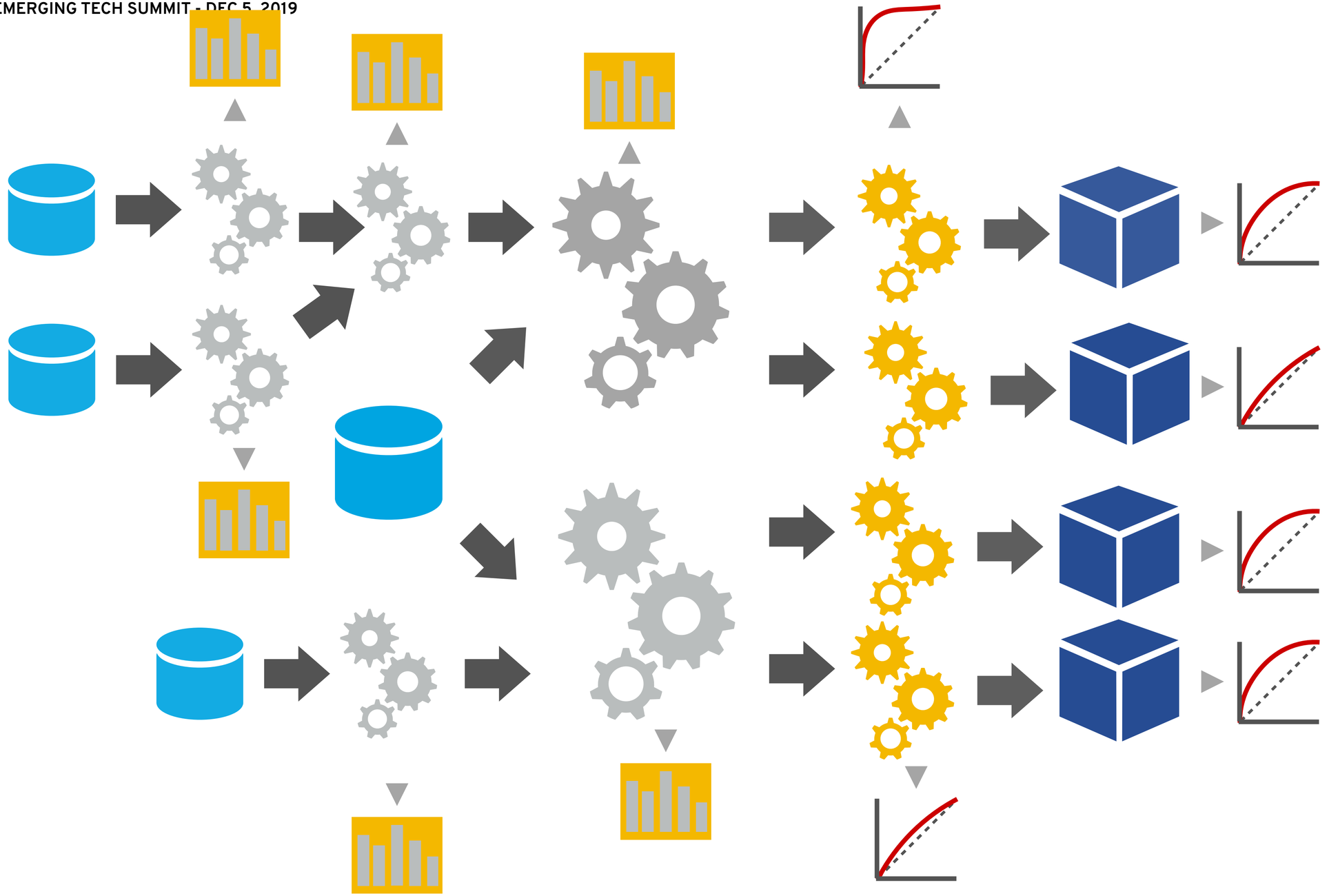


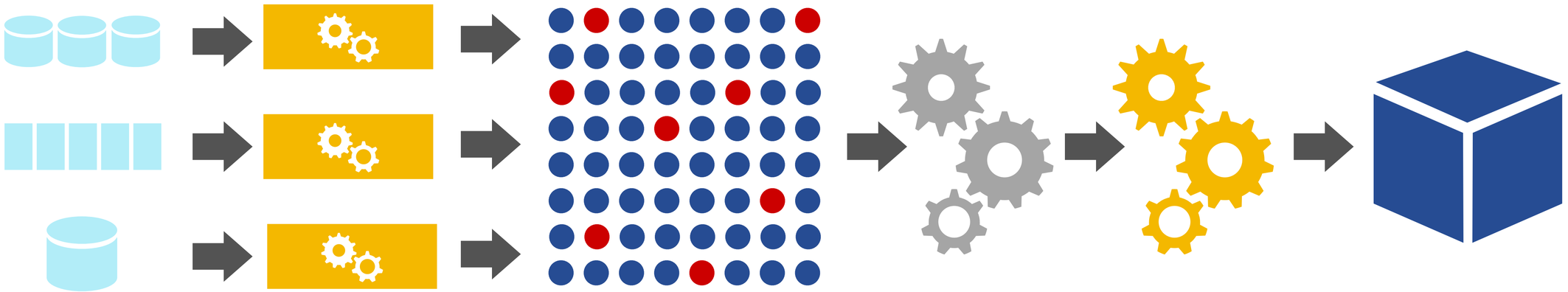




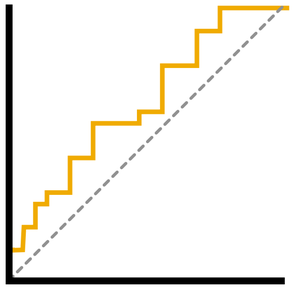




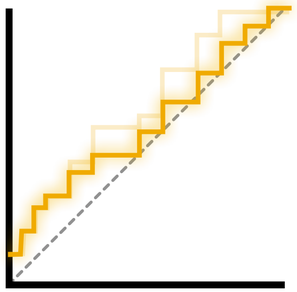





mA

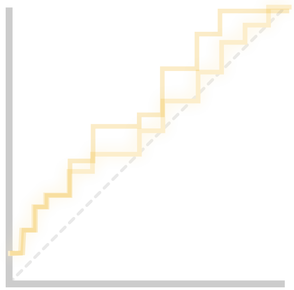



mA

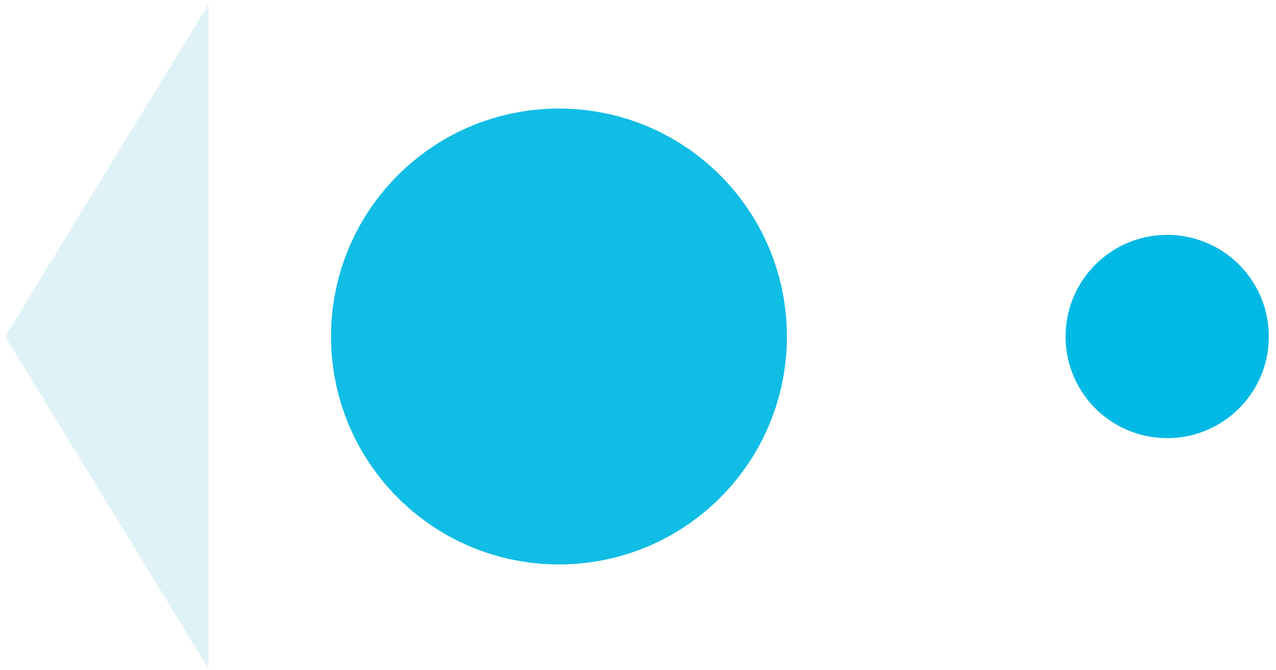


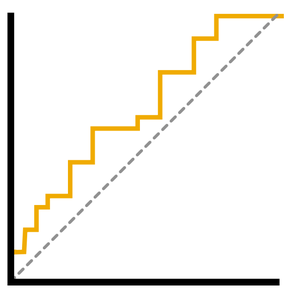
(joint) distribution of input data?

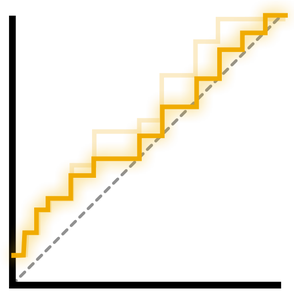
distribution of predictions?



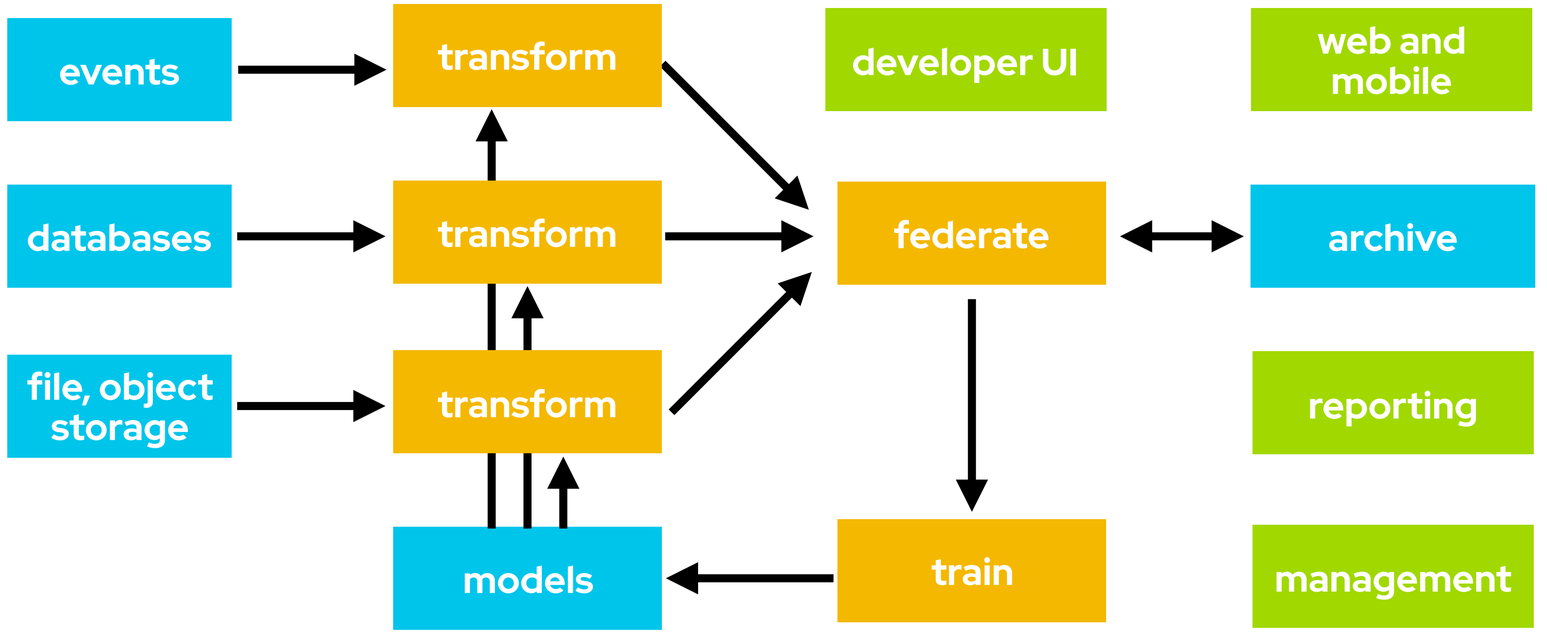
distribution of number of multiplications while scoring?

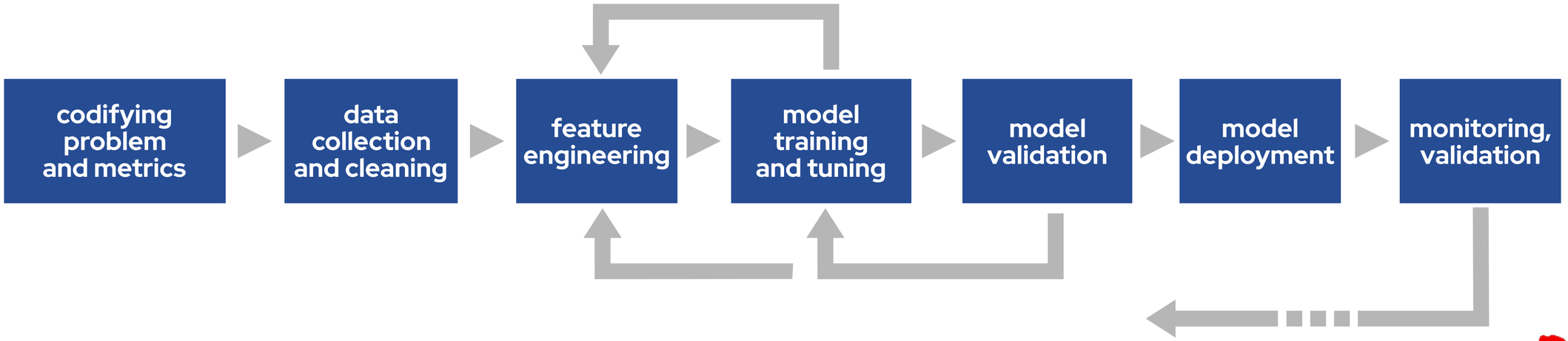




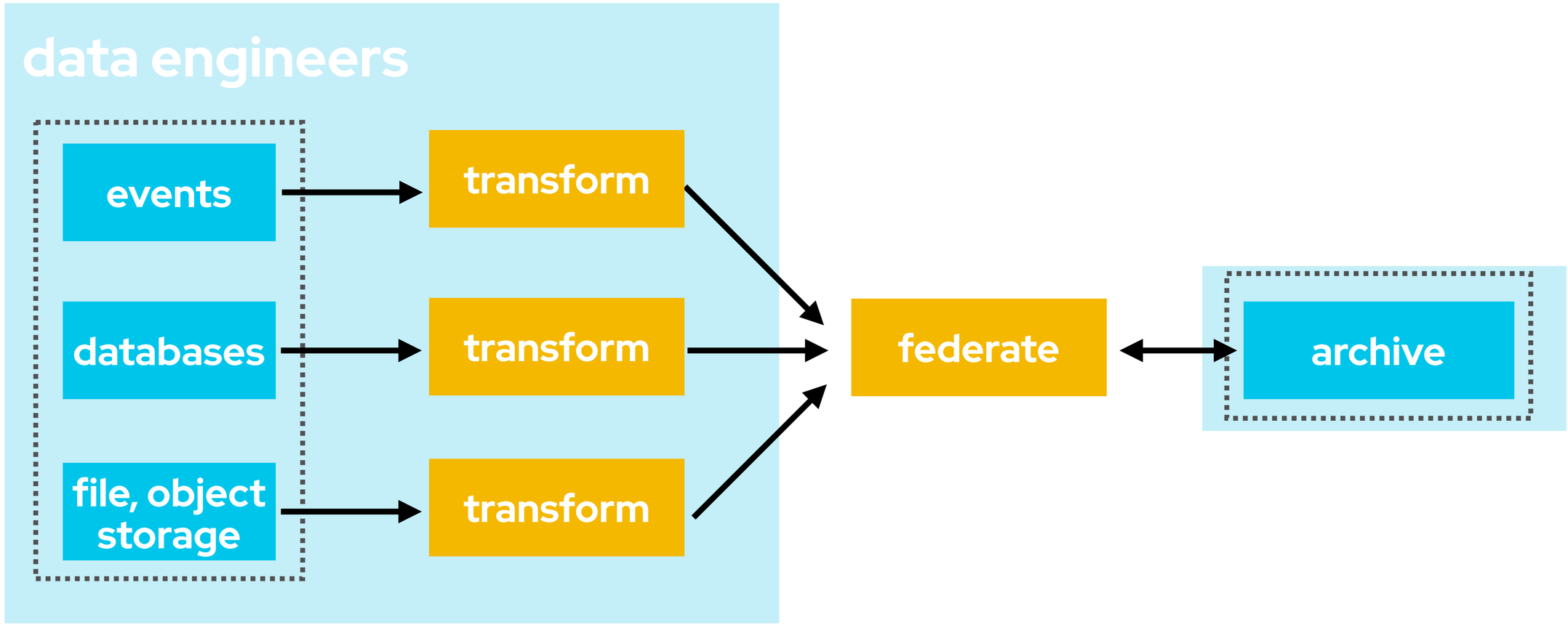


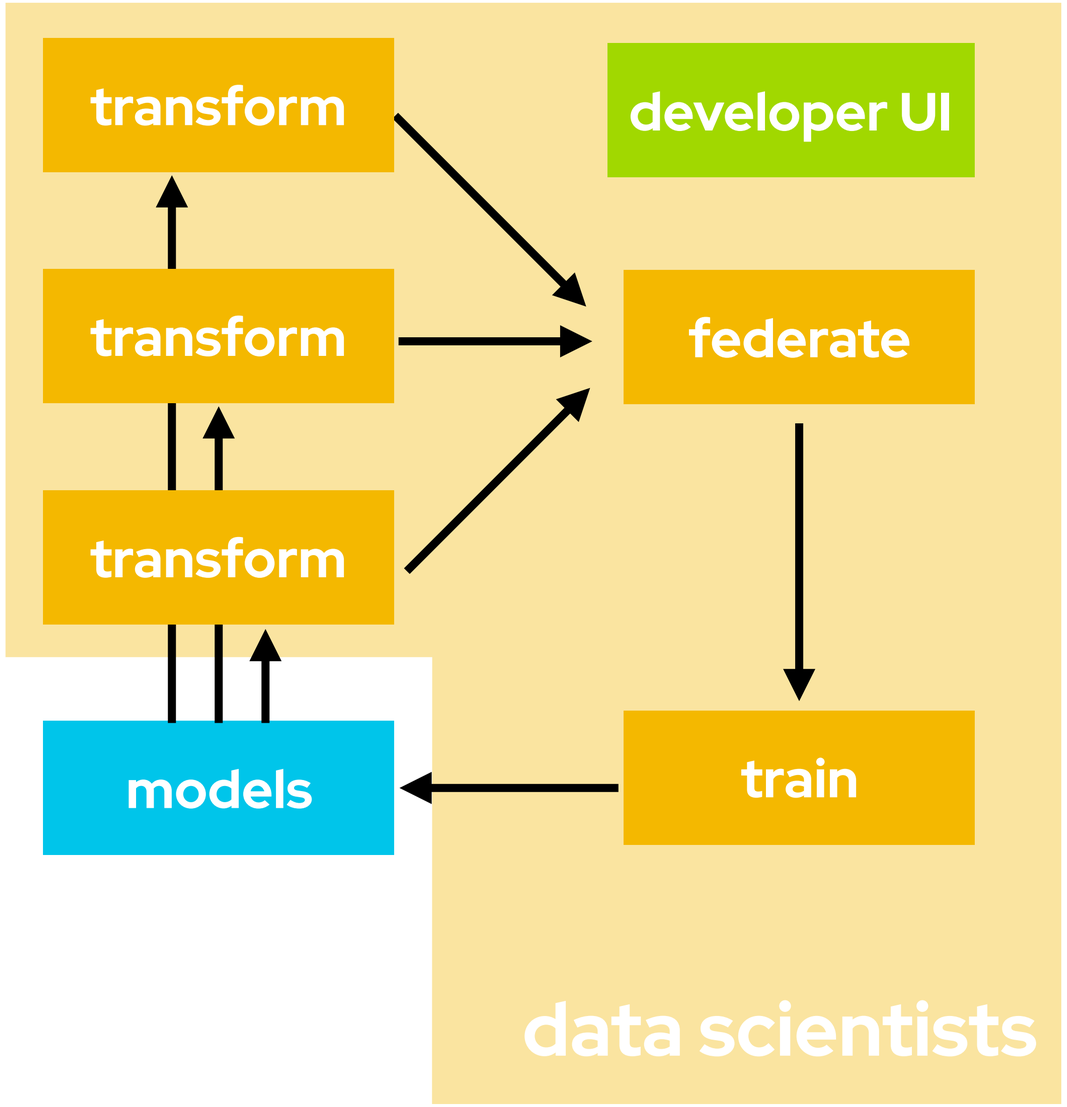
Intelligent applications are machine learning systems

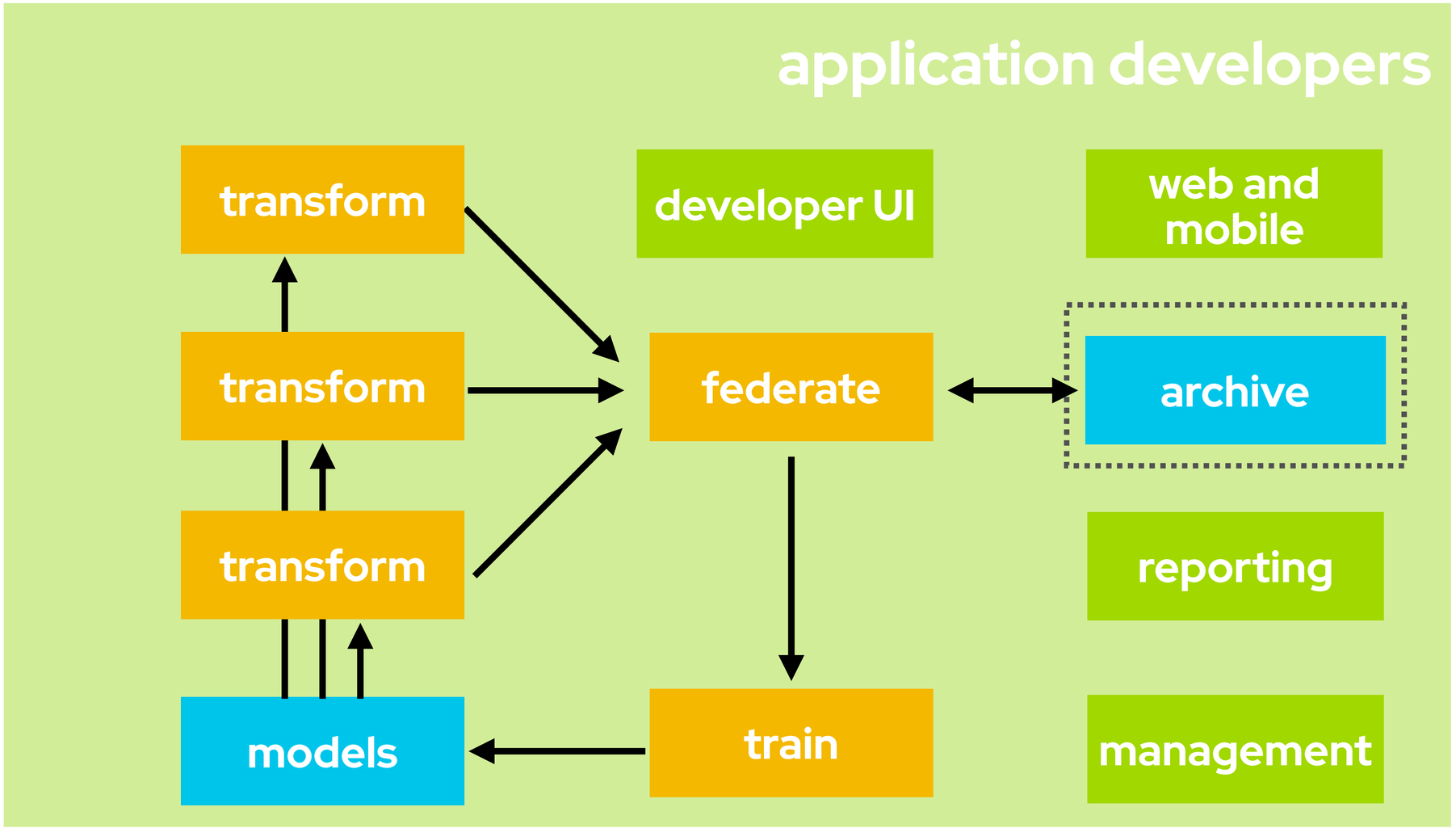

events
databases
file, object storage
transform
transform
transform
models
federate
train
developer UI
web and mobile
archive
reporting
management

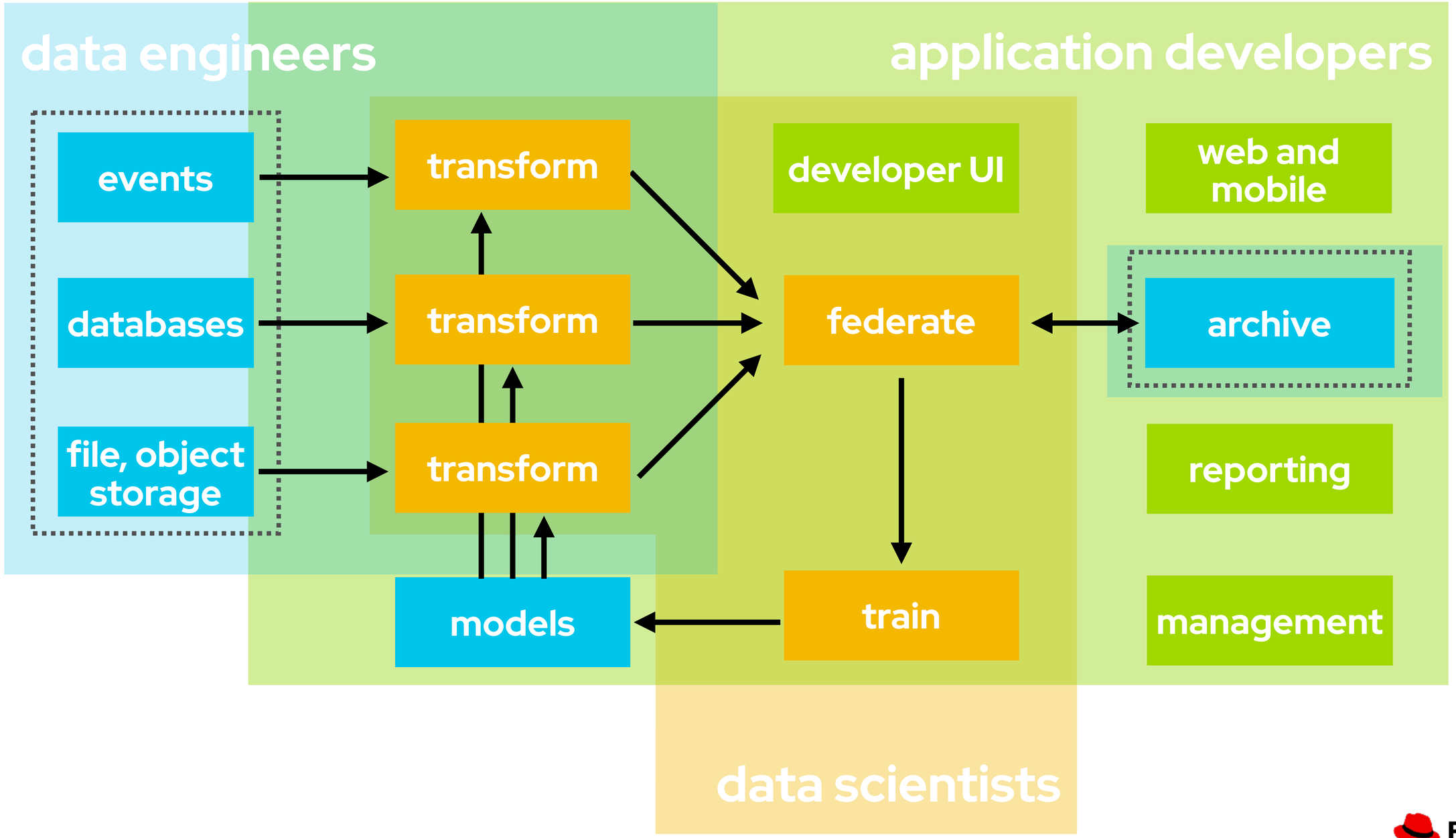

codifying problem and metrics
data collection and cleaning
feature engineering
model training and tuning
model validation
model deployment
monitoring, validation

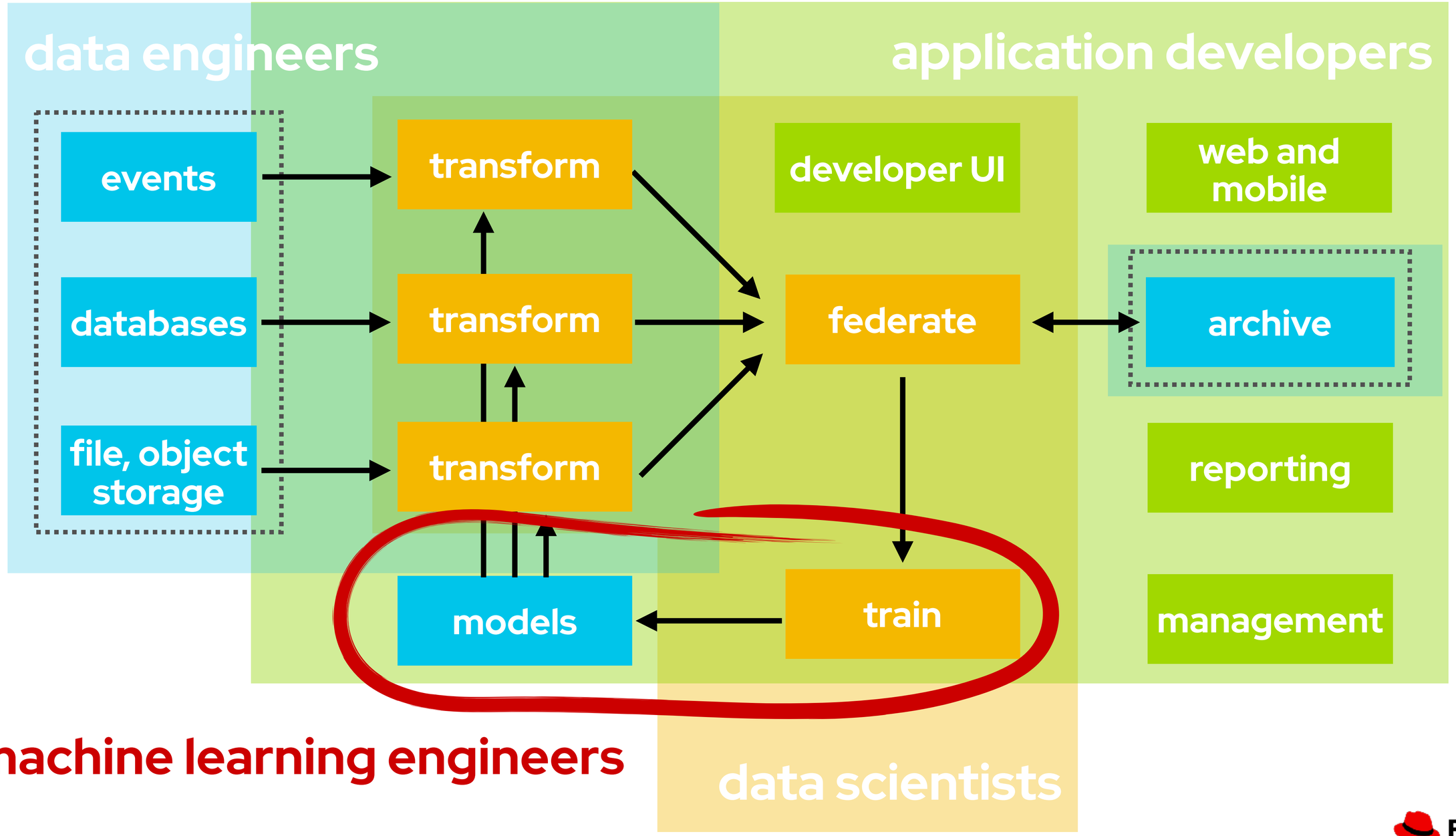
Red Hat


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

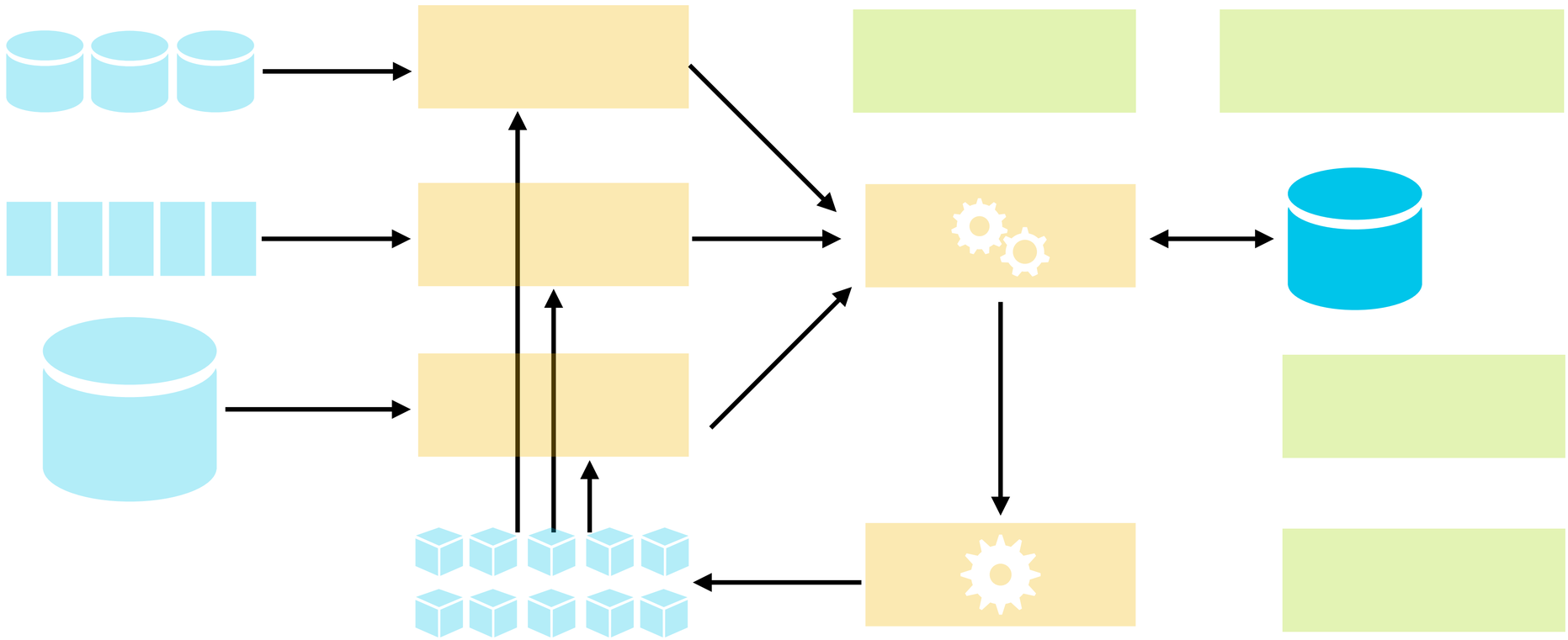

transform
developer UI
transform
federate
transform
models
train
data scientists

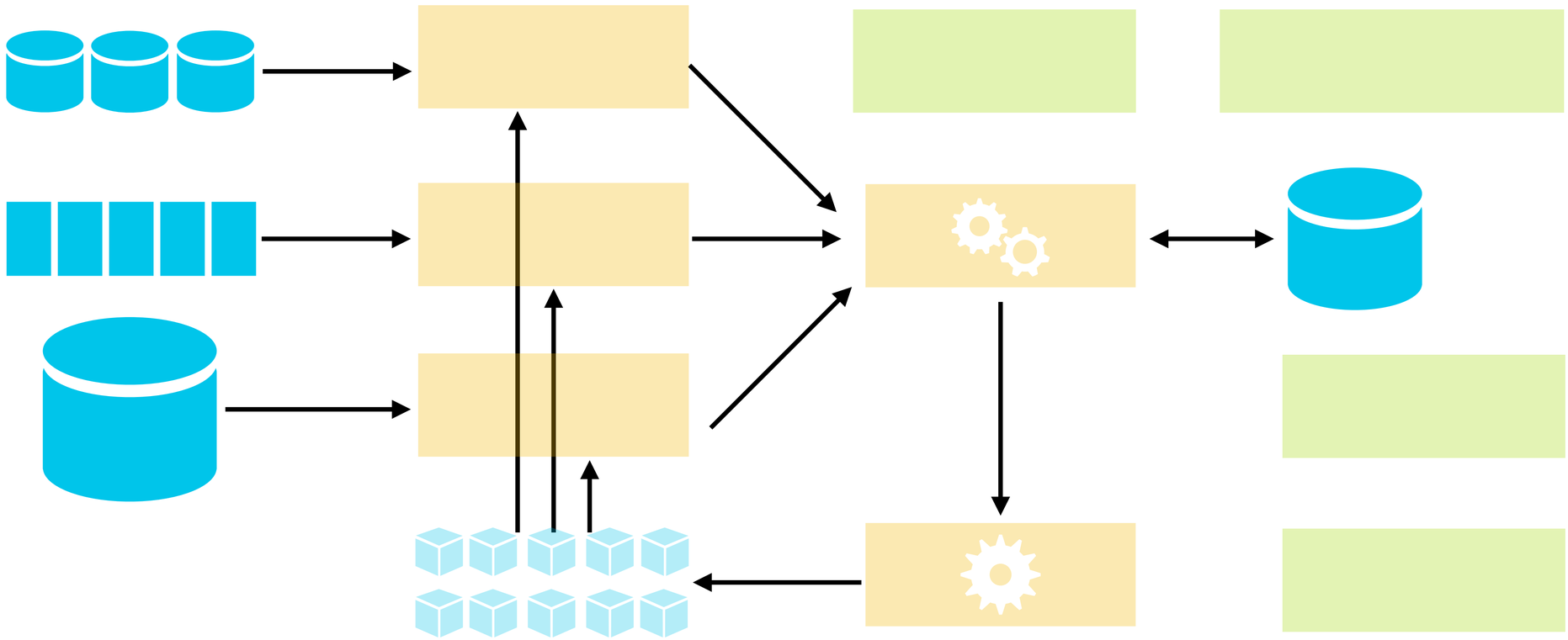

application developers
transform
transform
transform
models
developer UI
federate
train
web and mobile
archive
reporting
management

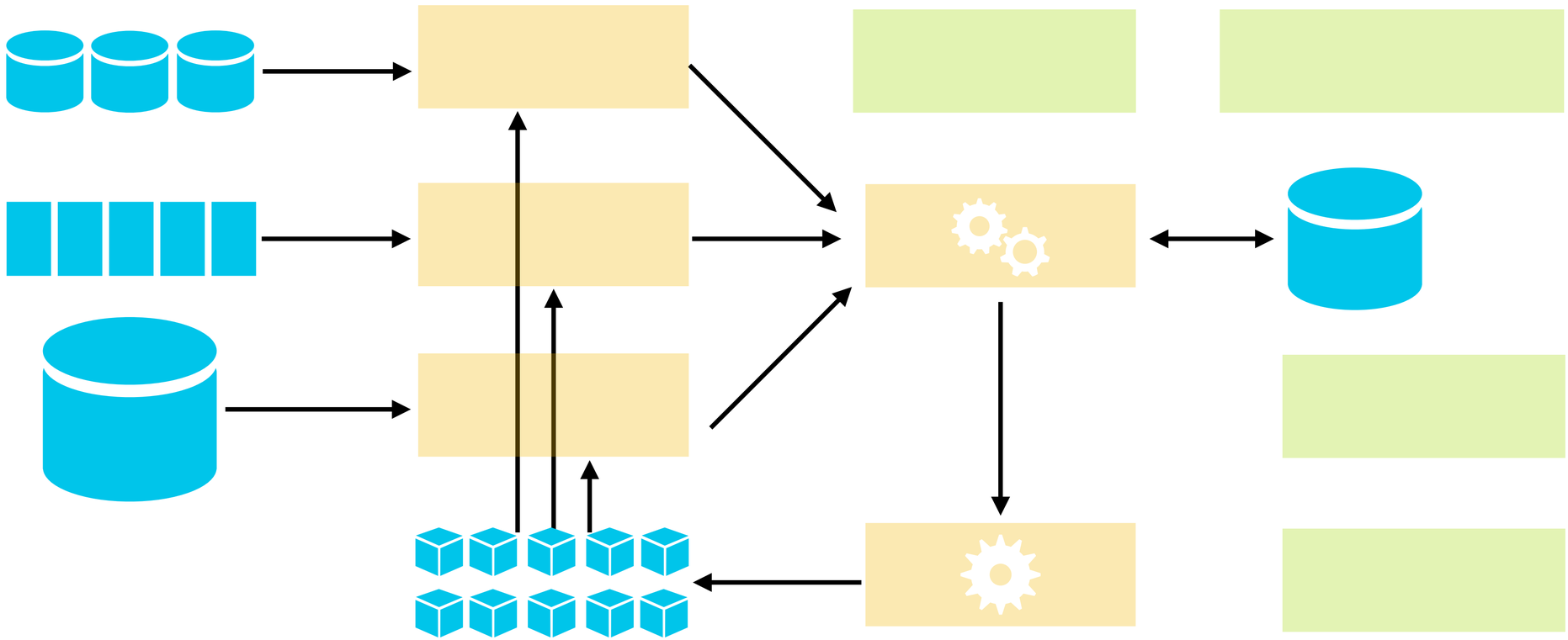

data engineers
application developers
events
databases
file, object storage
transform
transform
transform
models
developer UI
federate
train
web and mobile
archive
reporting
management
data scientists

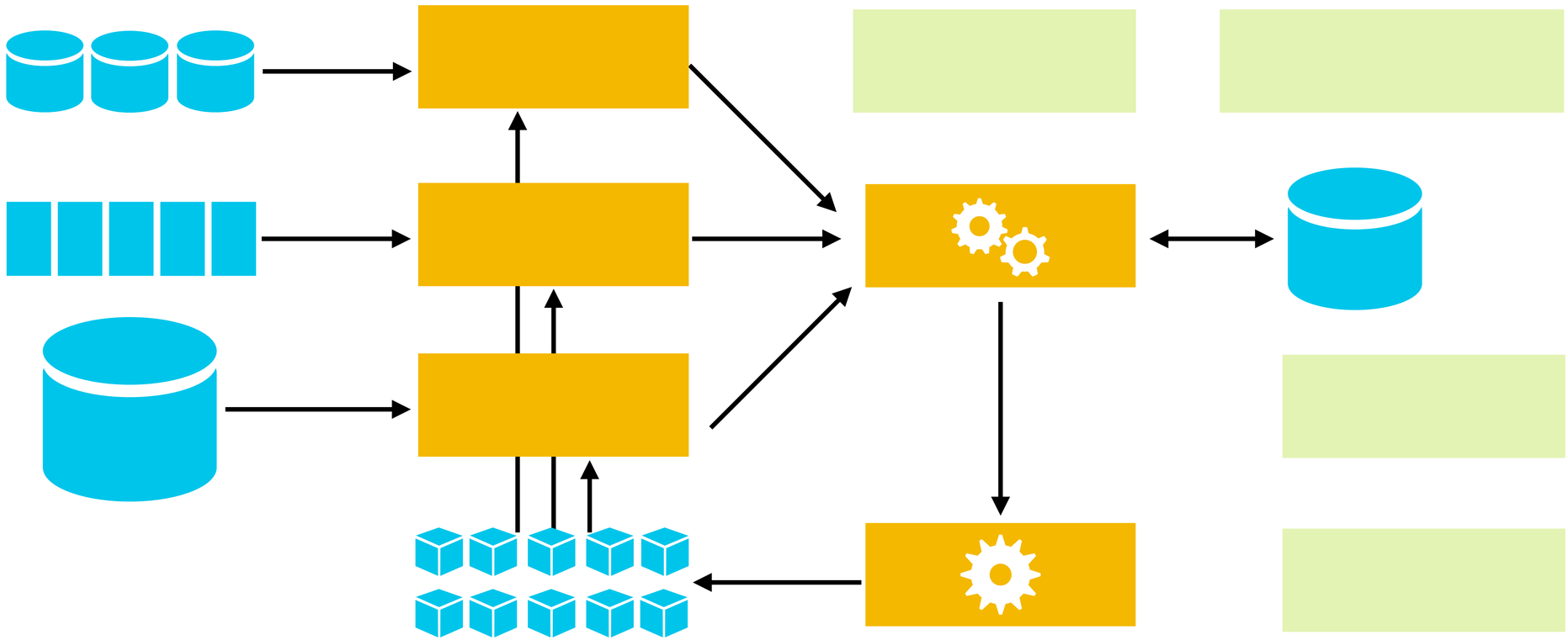

data engineers
application developers
events
databases
file, object storage
transform
transform
transform
developer UI
federate
web and mobile
archive
reporting
management
models
train
machine learning engineers
data scientists

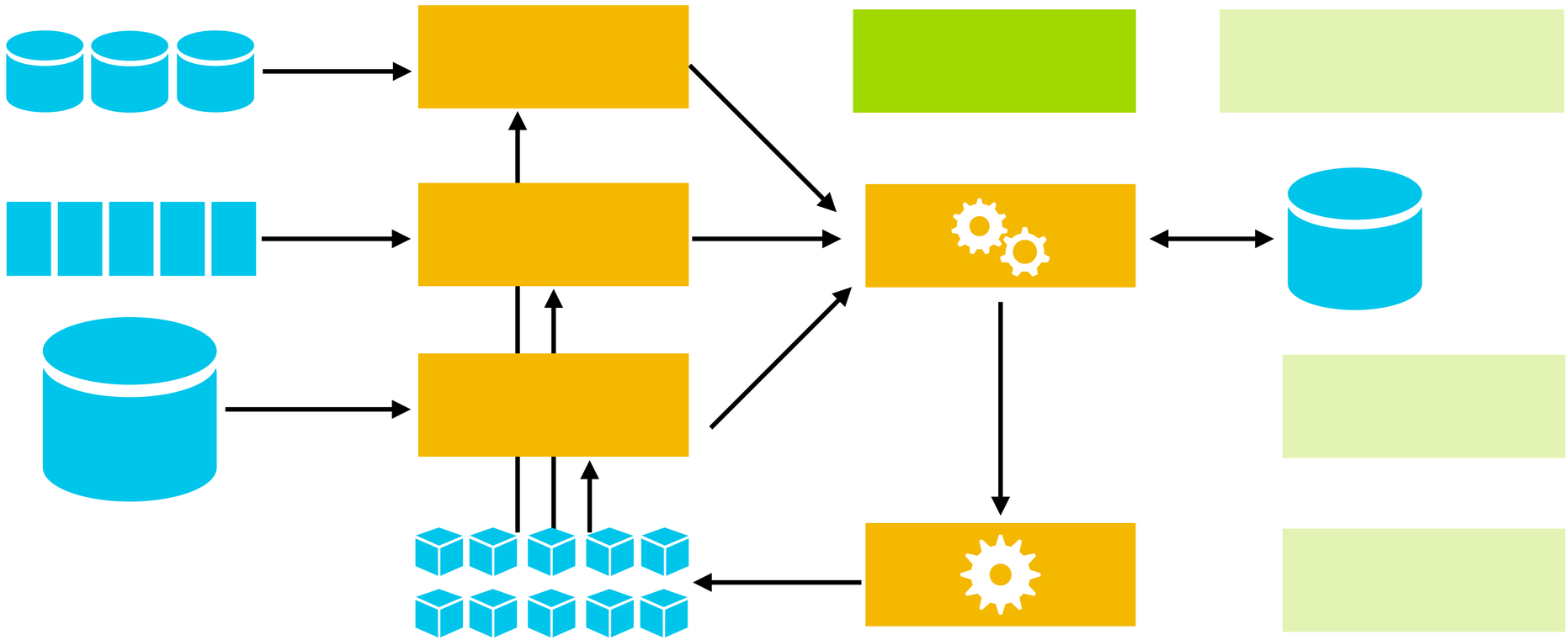
Managing compute and data in a shared discovery environment

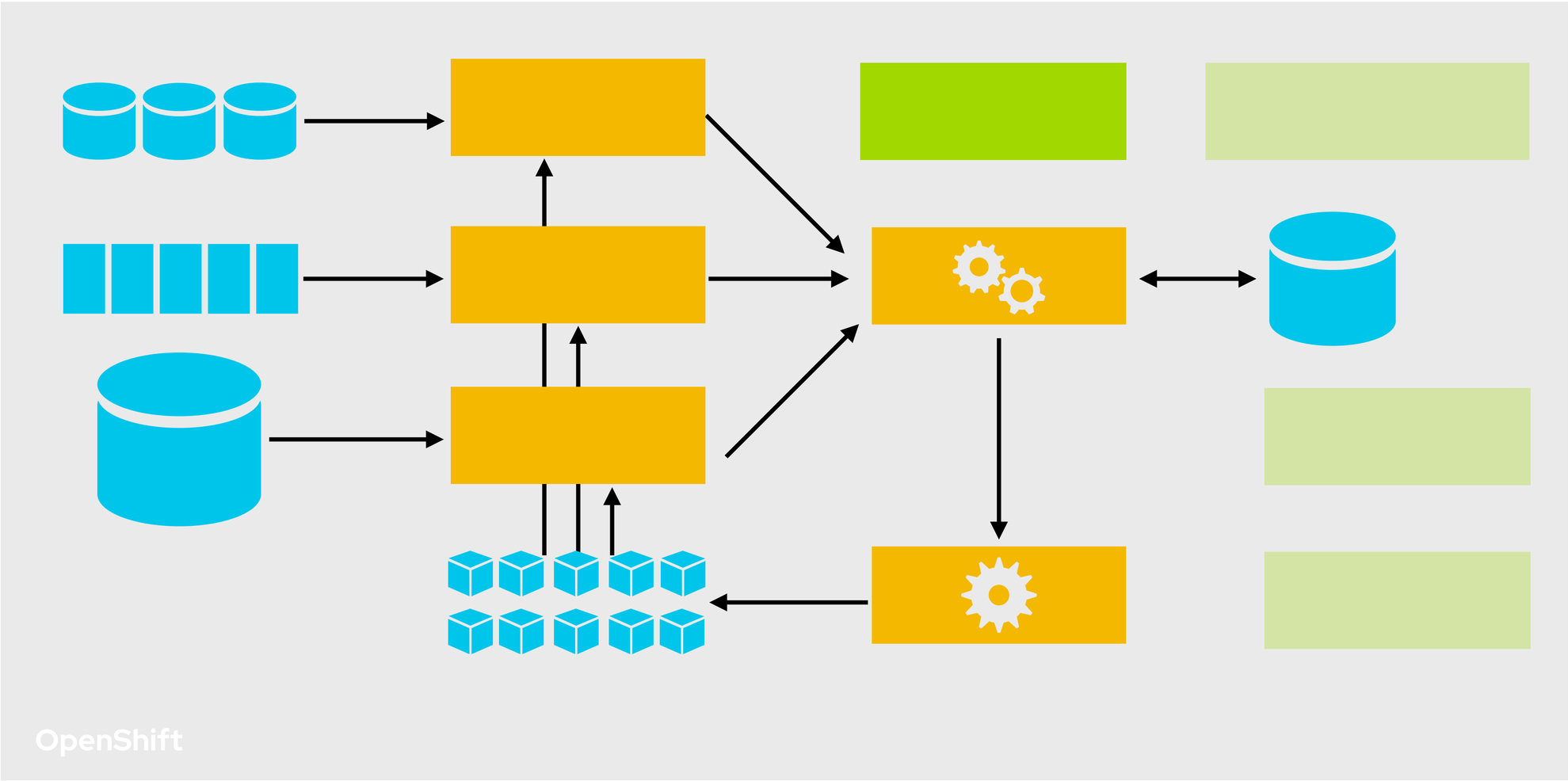








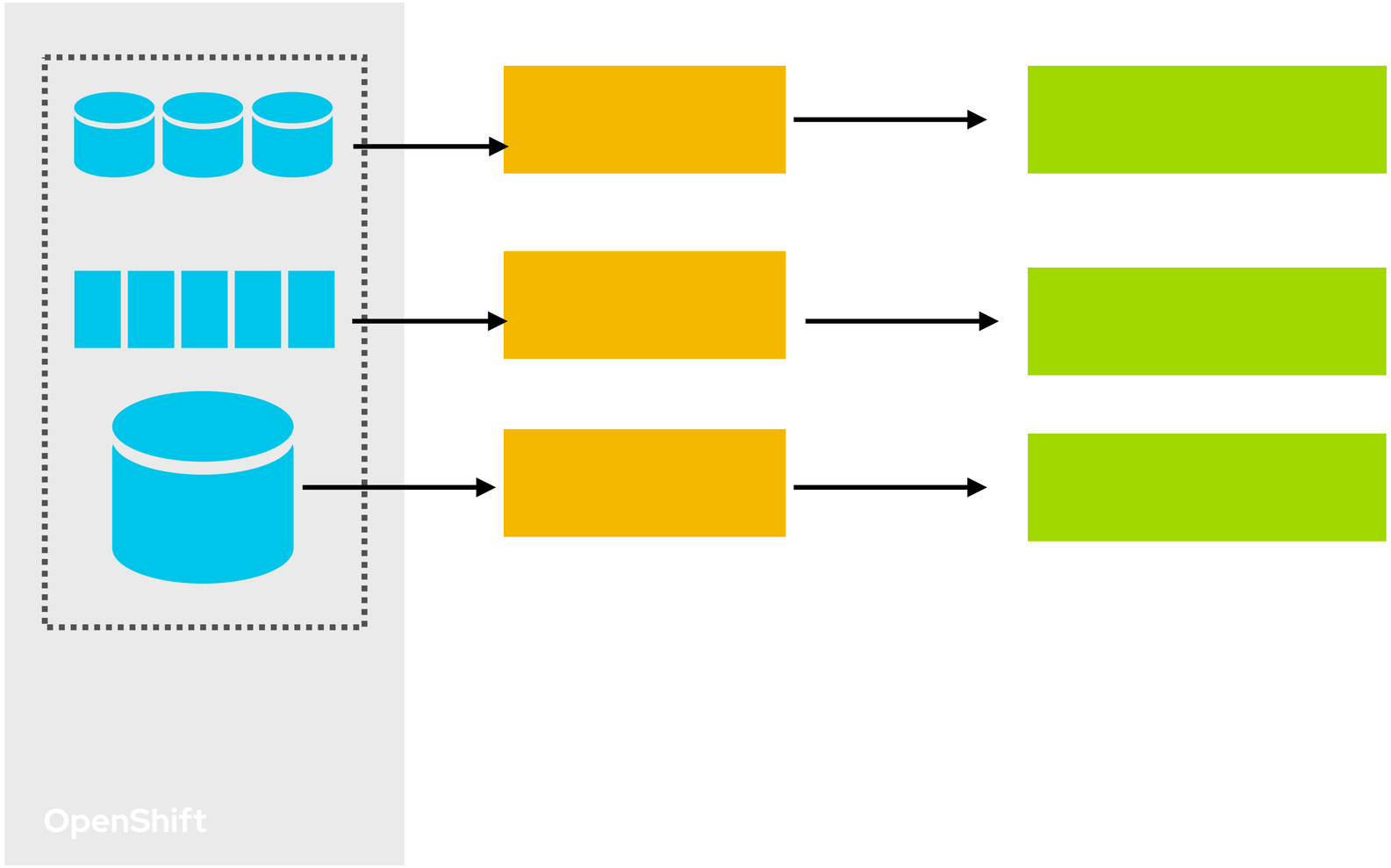


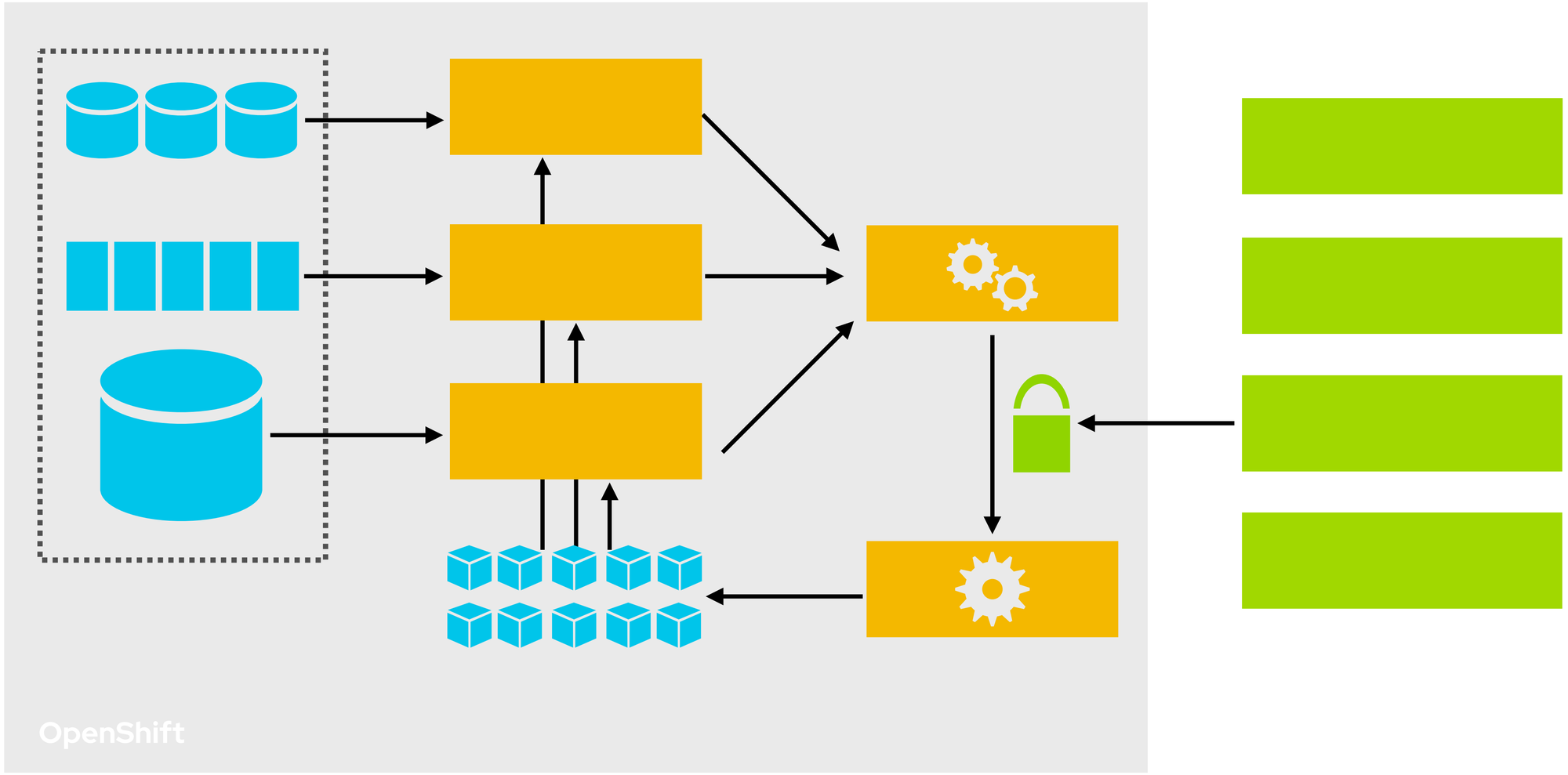

OpenShift

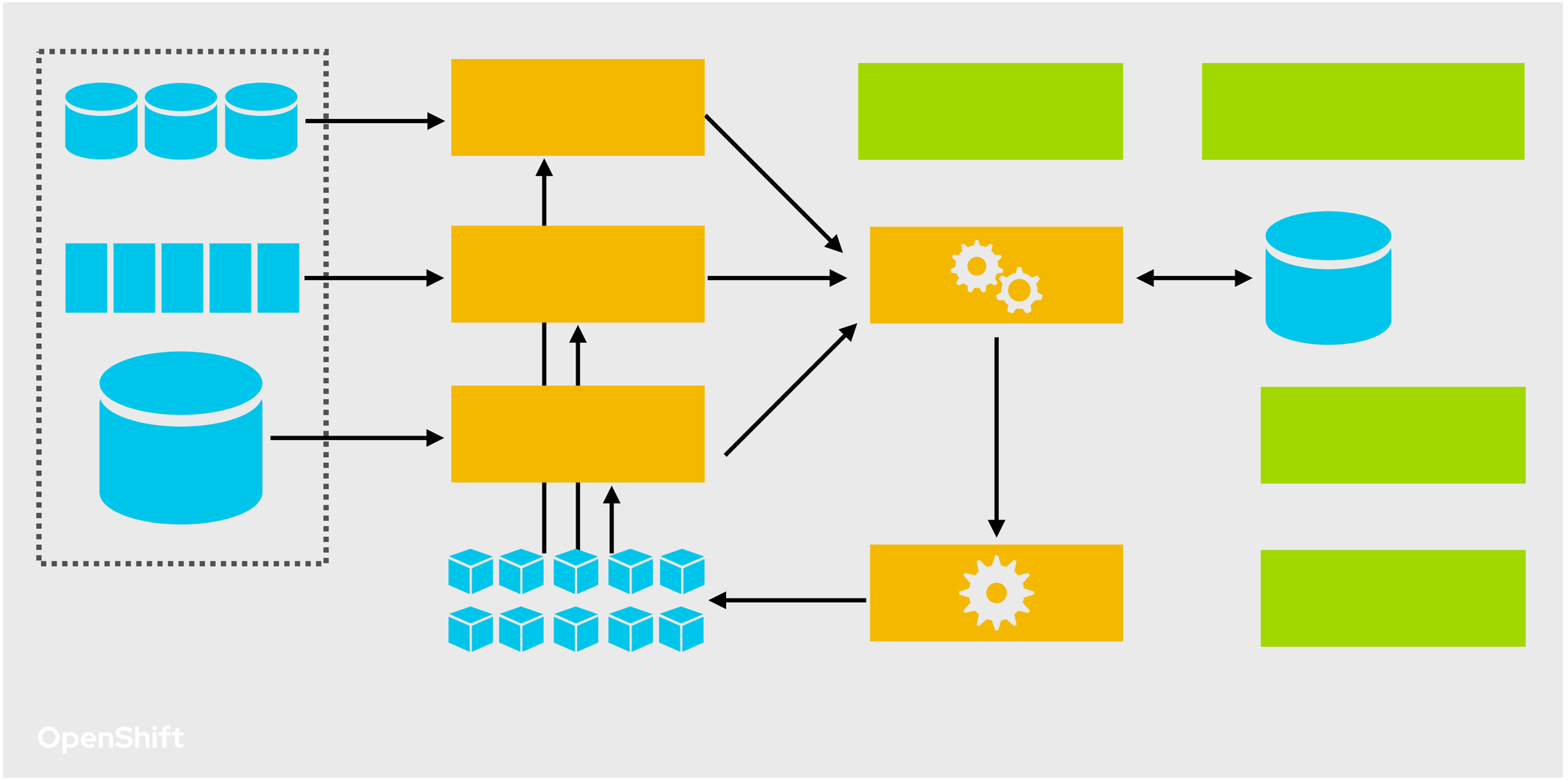

ceph

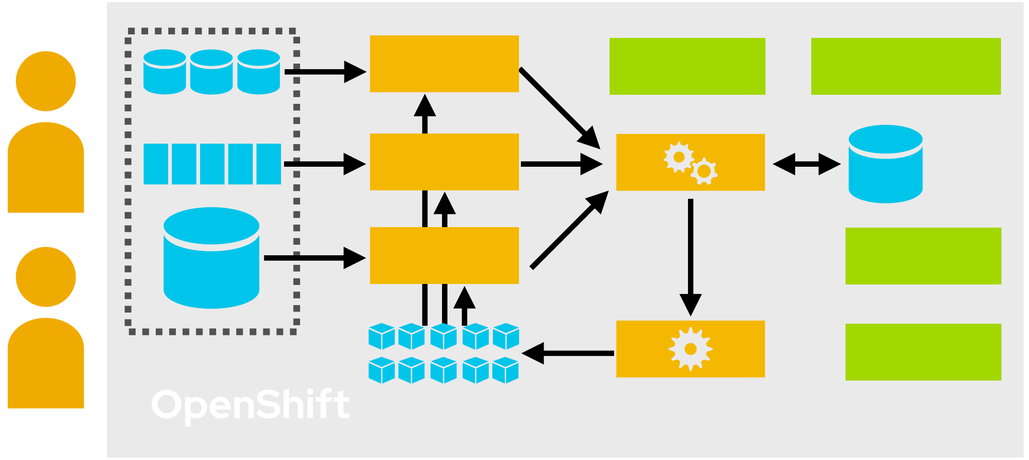

ceph
OPEN DATA HUB
A Platform powered by Open Source

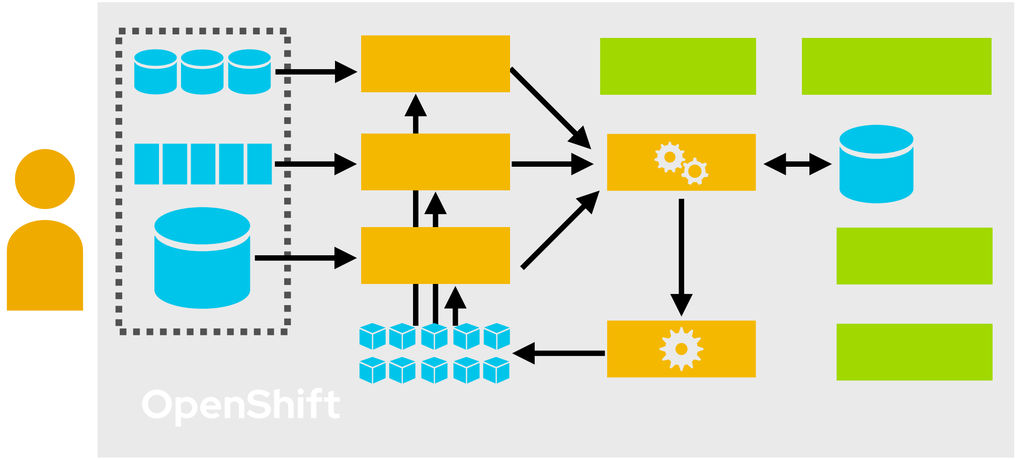
There's no one-size-fits-all architecture for AI/ML

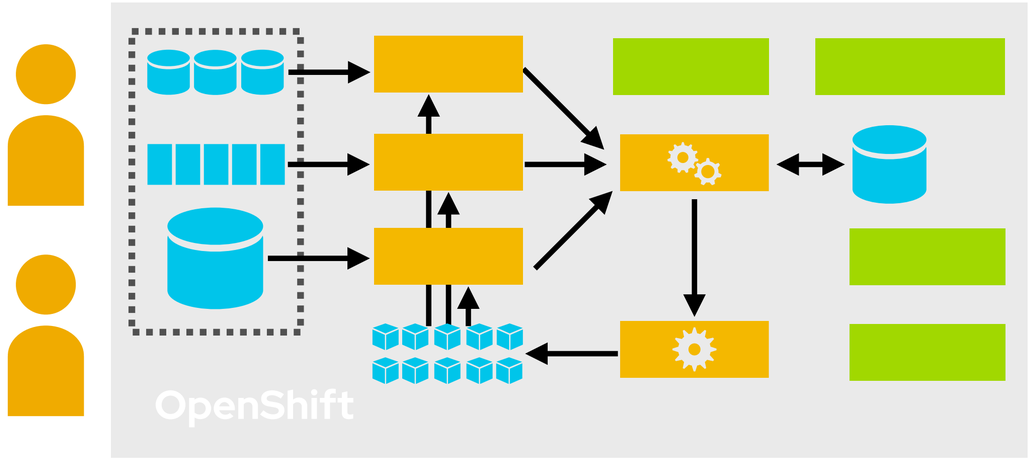

OpenShift

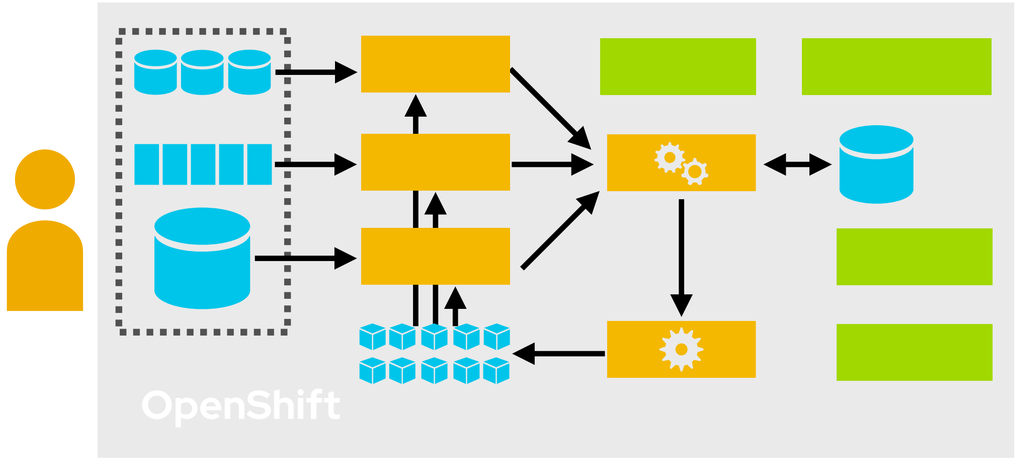

OpenShift

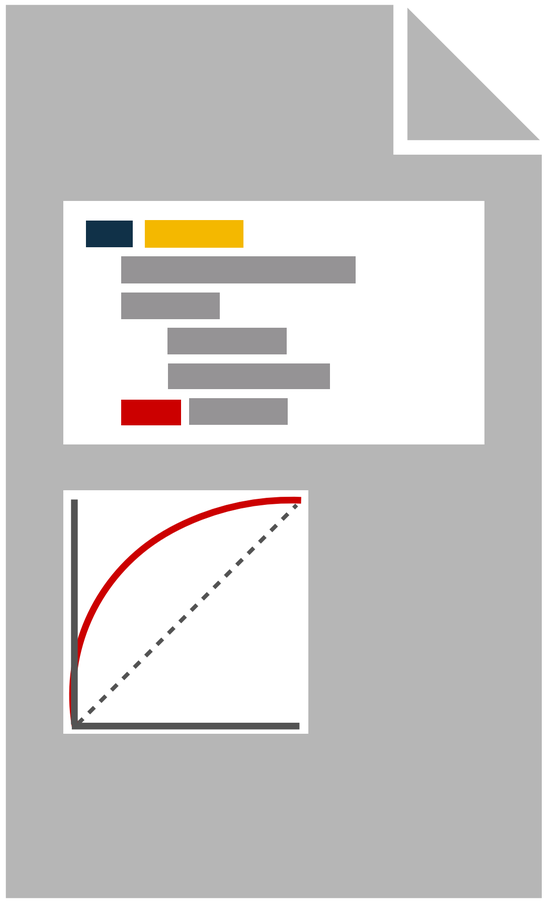

OpenShift

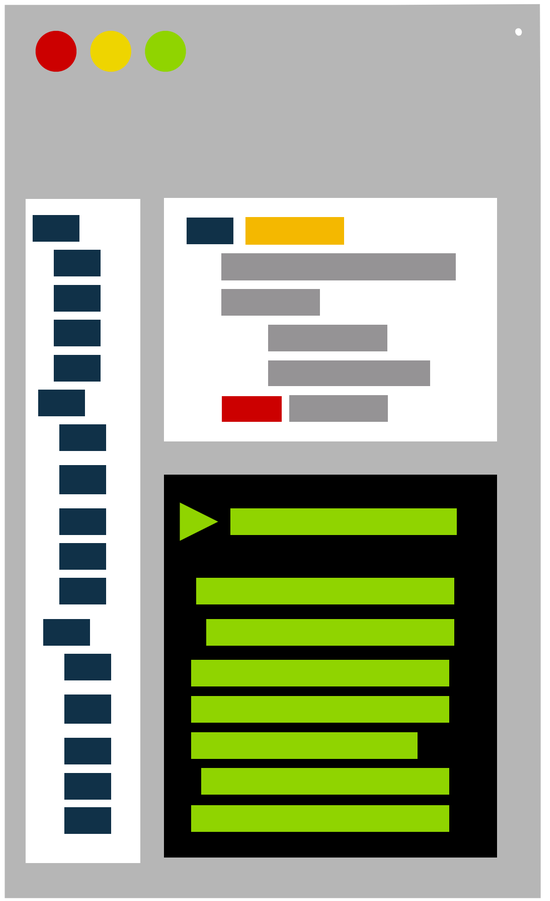

OpenShift


OpenShift


OpenShift

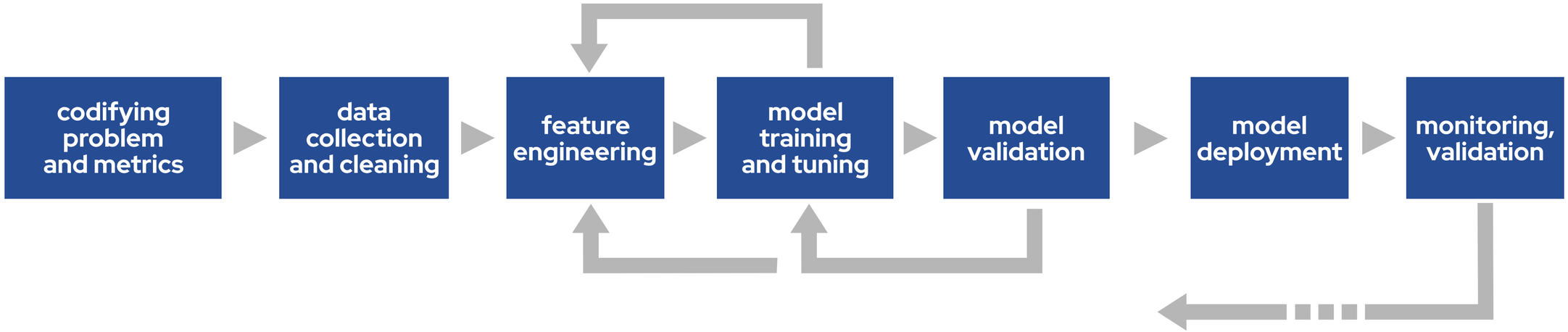

OpenShift

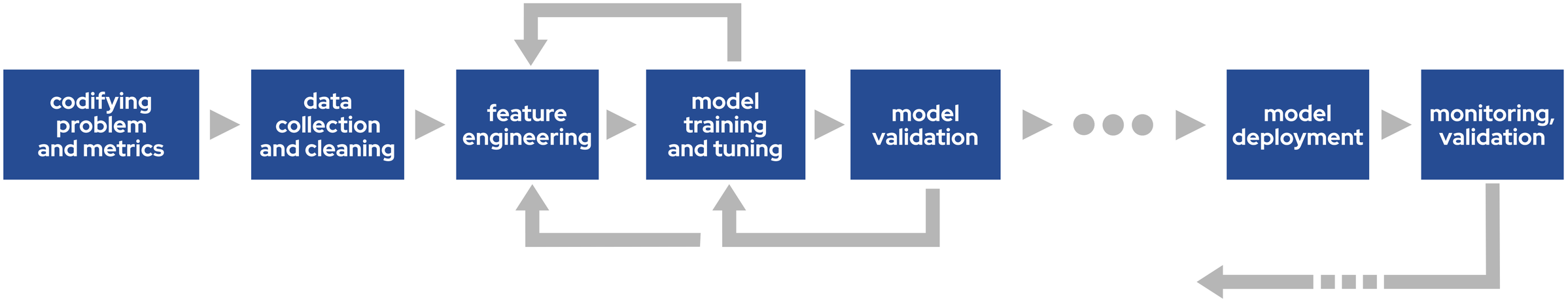


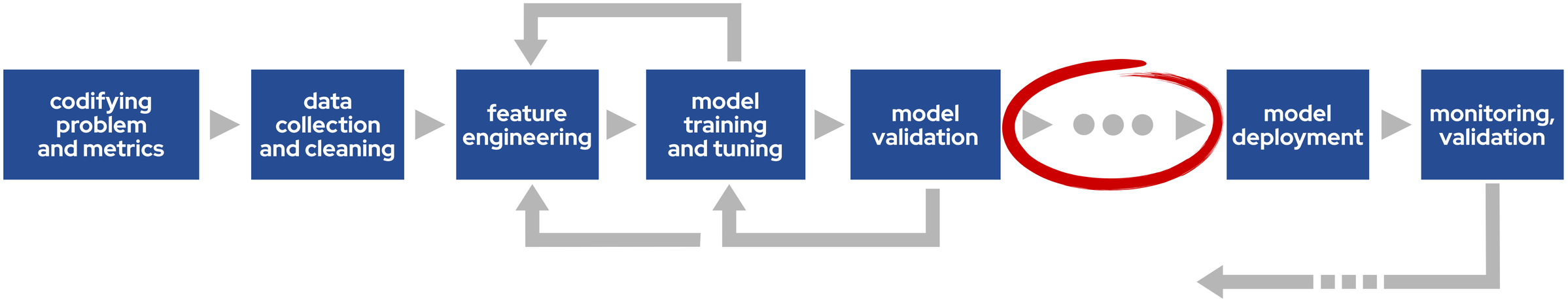


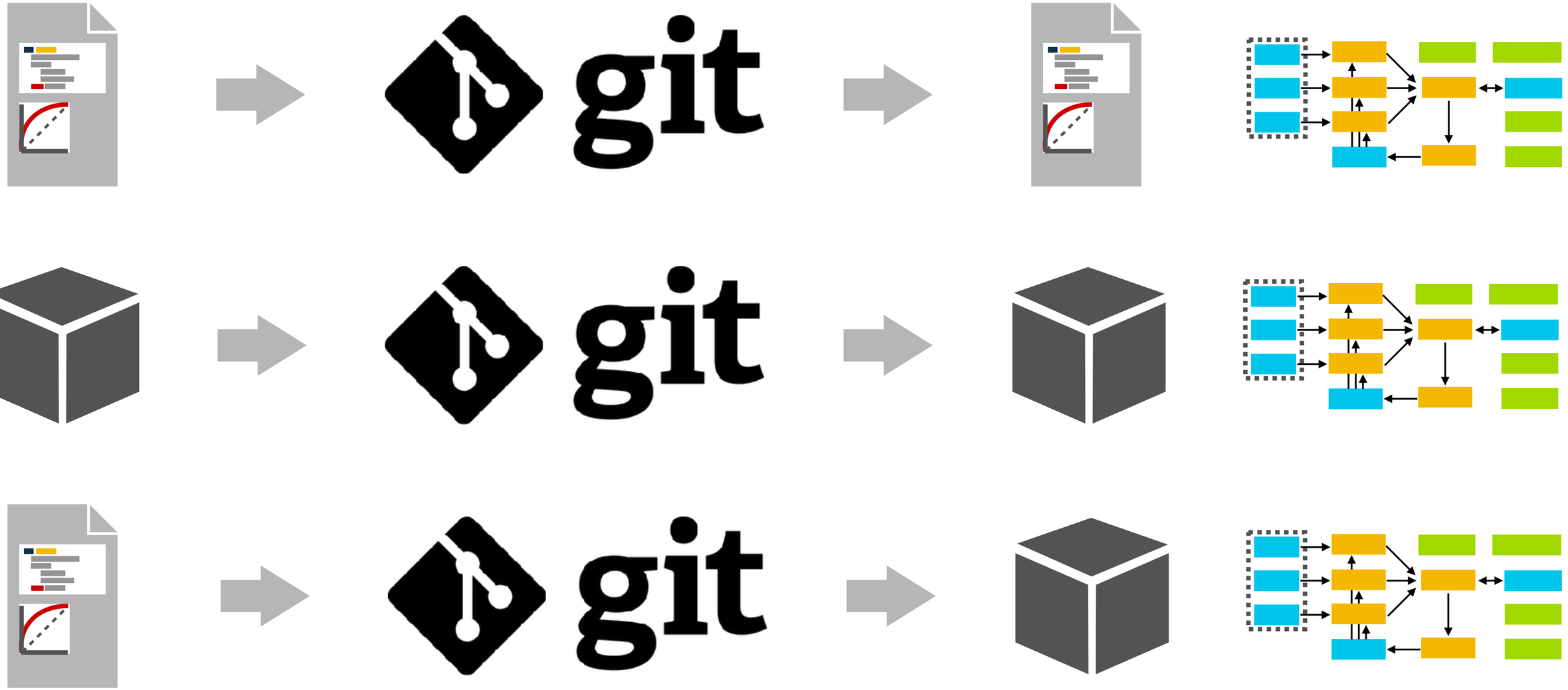
From developer experience to data scientist experience(s)


git
OK!


codifying problem and metrics
data collection and cleaning
feature engineering
model training and tuning
model validation
model deployment
monitoring, validation


codifying problem and metrics
data collection and cleaning
feature engineering
model training and tuning
model validation
model deployment
monitoring, validation


codifying problem and metrics
data collection and cleaning
feature engineering
model training and tuning
model validation
model deployment
monitoring, validation


git
git
git

In this notebook we will process the synthetic Austen/food reviews data and convert it into feature vectors. In later notebooks these feature vectors will be the inputs to models which we will train and eventually use to identify spam.

This notebook uses term frequency-inverse document frequency, or tf-idf, to generate feature vectors. Tf-idf is commonly used to summarise text data, and it aims to capture how important different words are within a set of documents. Tf-idf combines a normalized word count (or term frequency) with the inverse document frequency (or a measure of how common a word is across all documents) in order to identify words, or terms, which are 'interesting' or important within the document.

We begin by loading in the data:

In [1]:
```
import pandas as pd

df = pd.read_parquet("data/training.parquet")
```

To illustrate the computation of tf-idf vectors we will first implement the method on a sample of

In this notebook we will process the synthetic Austen/food reviews data and convert it into feature vectors. In later notebooks these feature vectors will be the inputs to models which we will train and eventually use to identify spam.

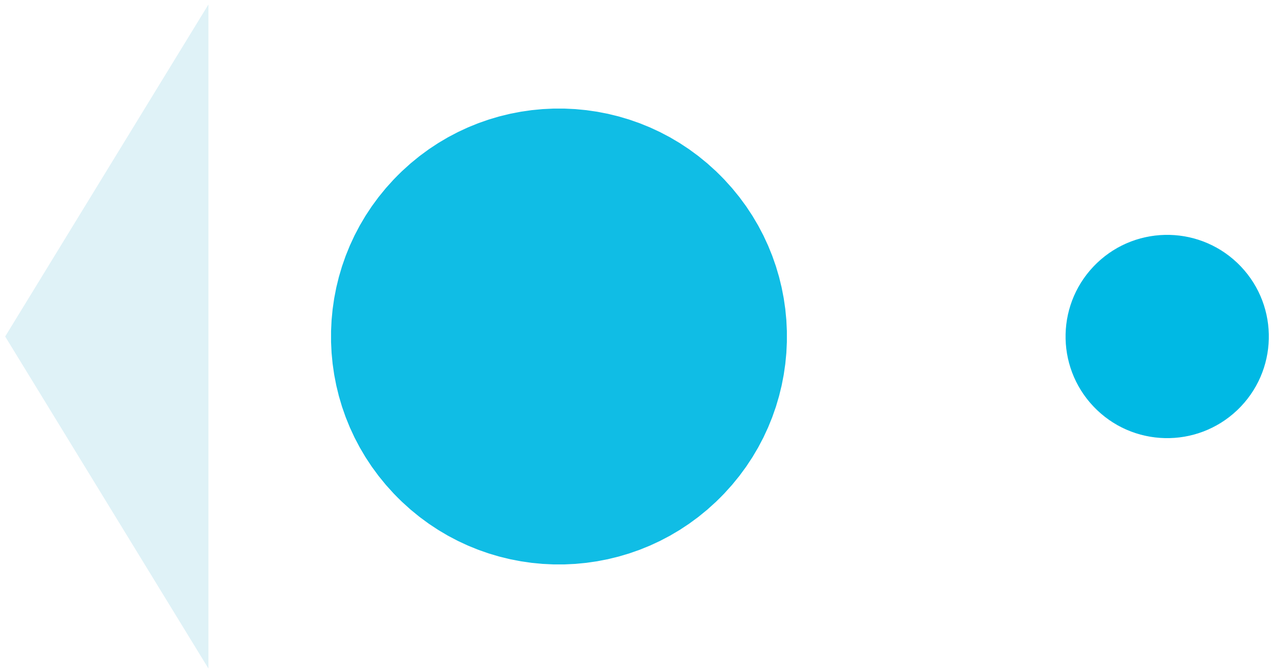
This notebook uses term frequency-inverse document frequency, or tf-idf, to generate feature vectors. Tf-idf is commonly used to summarise text data, and it aims to capture how important different words are within a set of documents. Tf-idf combines a normalized word count (or term frequency) with the inverse document frequency (or a measure of how common a word is across all documents) in order to identify words, or terms, which are 'interesting' or important within the document.

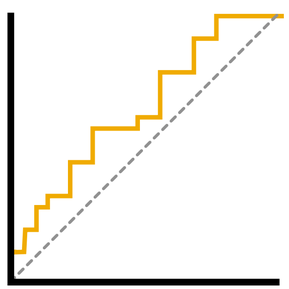
We begin by loading in the data:

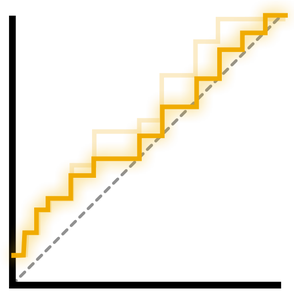
In [1]:
```
import pandas as pd

df = pd.read_parquet("data/training.parquet")
```

To illustrate the computation of tf-idf vectors we will first implement the method on a sample of







OpenShift Web Console | 07-services | Prometheus Time Series Colle

Not Secure | jupyter-d1pa-myproject.127.0.0.1.nip.io/notebooks/07-services.ipynb

jupyter 07-services Last Checkpoint: a minute ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run | Markdown

200 rows × 4 columns

Running our models as services gives us an interesting opportunity to detect data drift by publishing the distribution of our predictions as metrics. If the distribution of predictions shifts over time, we can use that as an indication that the distribution of the data we're evaluating has shifted as well, and that we should re-train our model.
In this example, our pipeline service publishes metrics related to the predictions made by the model (keys beginning with `pipeline_predictions_`) as well as metrics related to the computational performance of our pipeline service (keys beginning with `pipeline_processing_seconds_`).

In [ ]:
```
get_metrics()
```

Since our service publishes Prometheus metrics, we can define alerting rules or visualize how our metric values change over time.

In [ ]:
```
def experiment(data, size, **kwargs):
    for k, v in kwargs.items():
        sample = data[data.label == k].sample(int(size * v))
        score_text(sample["text"].values.tolist())
```

In [ ]:
```
experiment(data, 20000, legitimate=.05, spam=.95)
experiment(data, 20000, legitimate=.05, spam=.95)
experiment(data, 20000, legitimate=.05, spam=.95)
```

In [ ]:
```
experiment(data, 20000, legitimate=.25, spam=.75)
```

In [ ]:
```
experiment(data, 20000, legitimate=1)
```

Exercises

1. What would a REST API for serving multiple versions of multiple pipelines look like? (Hint: consider the verbs and nouns involved.)
2. While we don't have a monitoring stack enabled for this workshop, you can certainly install and configure one in your own OpenShift cluster. What sort of alerting rules might you install to identify data drift in this application?

OpenShift Web Console | 07-services | Prometheus Time Series Colle

Not Secure | jupyter-d1pa-myproject.127.0.0.1.nip.io/notebooks/07-services.ipynb

jupyter 07-services Last Checkpoint: a minute ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run | Markdown

200 rows × 4 columns

Running our models as services gives us an interesting opportunity to detect data drift by publishing the distribution of our predictions as metrics. If the distribution of predictions shifts over time, we can use that as an indication that the distribution of the data we're evaluating has shifted as well, and that we should re-train our model.
In this example, our pipeline service publishes metrics related to the predictions made by the model (keys beginning with `pipeline_predictions_`) as well as metrics related to the computational performance of our pipeline service (keys beginning with `pipeline_processing_seconds_`).

In [ ]:
```
get_metrics()
```

Since our service publishes Prometheus metrics, we can define alerting rules or visualize how our metric values change over time.

In [ ]:
```
def experiment(data, size, **kwargs):
    for k, v in kwargs.items():
        sample = data[data.label == k].sample(int(size * v))
        score_text(sample["text"].values.tolist())
```

In [ ]:
```
experiment(data, 20000, legitimate=.05, spam=.95)
experiment(data, 20000, legitimate=.05, spam=.95)
experiment(data, 20000, legitimate=.05, spam=.95)
```

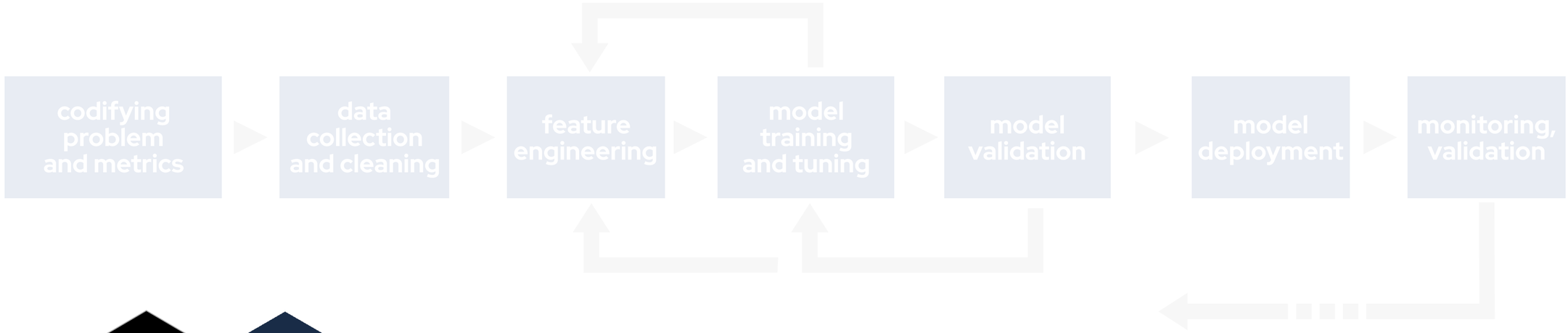
In [ ]:
```
experiment(data, 20000, legitimate=.25, spam=.75)
```

In [ ]:
```
experiment(data, 20000, legitimate=1)
```

Exercises

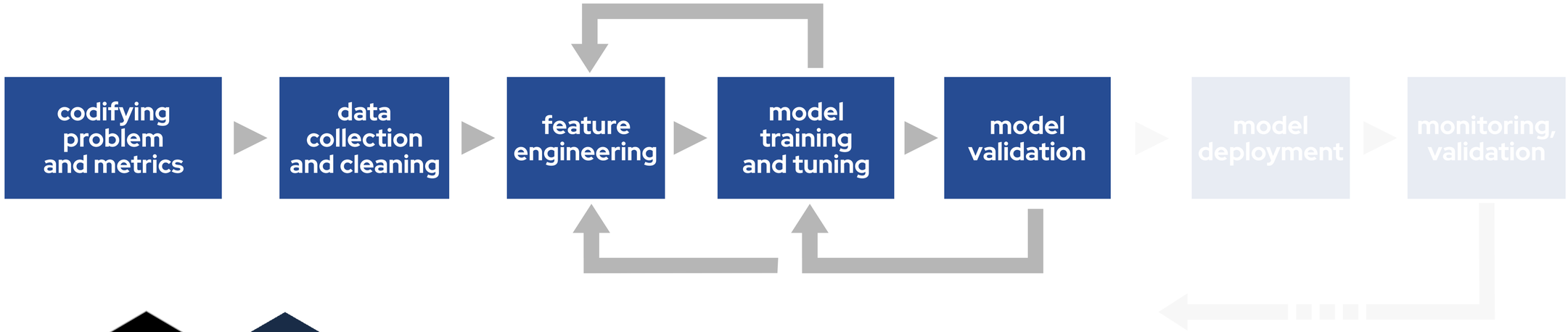
1. What would a REST API for serving multiple versions of multiple pipelines look like? (Hint: consider the verbs and nouns involved.)
2. While we don't have a monitoring stack enabled for this workshop, you can certainly install and configure one in your own OpenShift cluster. What sort of alerting rules might you install to identify data drift in this application?

Conclusions


codifying problem and metrics
data collection and cleaning
feature engineering
model training and tuning
model validation
model deployment
monitoring, validation

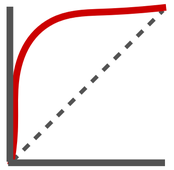

ceph
OPEN DATA HUB

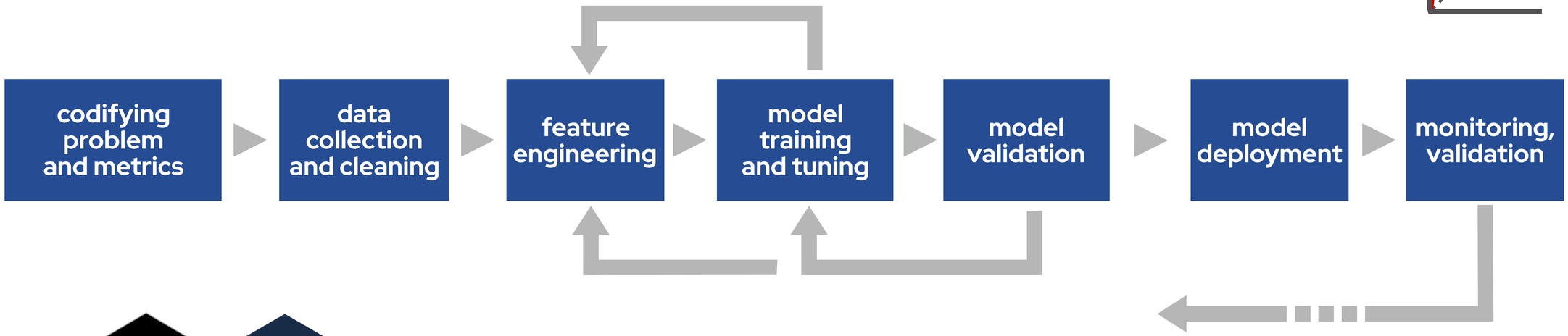
jupyter



Red Hat

jupyter





Red Hat