Course Curriculum - Part 1

Testing:

- Testing Theory, and the key challenges and strategies specific to machine learning systems
- ML Unit Test Exercises (hands-on notebooks)
- Unit Testing a Production ML System (example project)
- Docker Refresher
- ML Integration Testing (example project)
- Differential Testing (example project)

Course Curriculum - Part 2

Shadow Mode:

- Shadow mode theory
- Shadow mode implementation (example project)
- Shadow data interpretation and analysis (statistical techniques & hands-on notebook)

Course Curriculum - Part 3

Monitoring:

- Monitoring Theory, and how it applies to machine learning systems
- Implementing ML metrics monitoring with Prometheus and Grafana (example project)
- Implementing ML log monitoring with Kibana (example project)

Out of Scope

Not included:

- Deployment
- Algorithms/Feature Engineering/Data Science Fundamentals
- Model evaluation & tuning (i.e. in the research phase)
- Deep Learning/NLP example project

Course Requirements

Data Science Skills

This is an advanced course

Key Requirements:

Must have:

- Familiarity with Feature Engineering & Feature Selection
- Familiarity with Linear & Logistic Regression
- Familiarity with Gradient Boosting for Regression

Course Requirements

Software Engineering Skills

This is an advanced course

Key Requirements:

Must have:

- Decent knowledge of Python
- Knowledge of pandas/scikit-learn/numpy
- Knowledge of git

Nice to have:

- Basic knowledge of Docker (we will be doing a recap)
- Basic knowledge of Flask

Course Requirements

Your Computer

Key Requirements:

Must have:

- An internet connection
- Windows 7+ / MacOS / Linux Operating system (with admin permissions)
- A working Python 3.6 / 3.7 / 3.8 installation (we cover this)
- A working Docker installation (we cover this)
- **At least 2GB of free disk space**

How to Approach This Course

PLEASE DO NOT SKIP

Useful Information Inside

Understanding the Course Design

Approach

- Gradual setup (right before you need it).
- Gradual increase in complexity.
- Example project builds up over time.
- Be sure to follow along with the course lectures - the example project is designed as a continuous hands-on project for you to use.

Course Notes

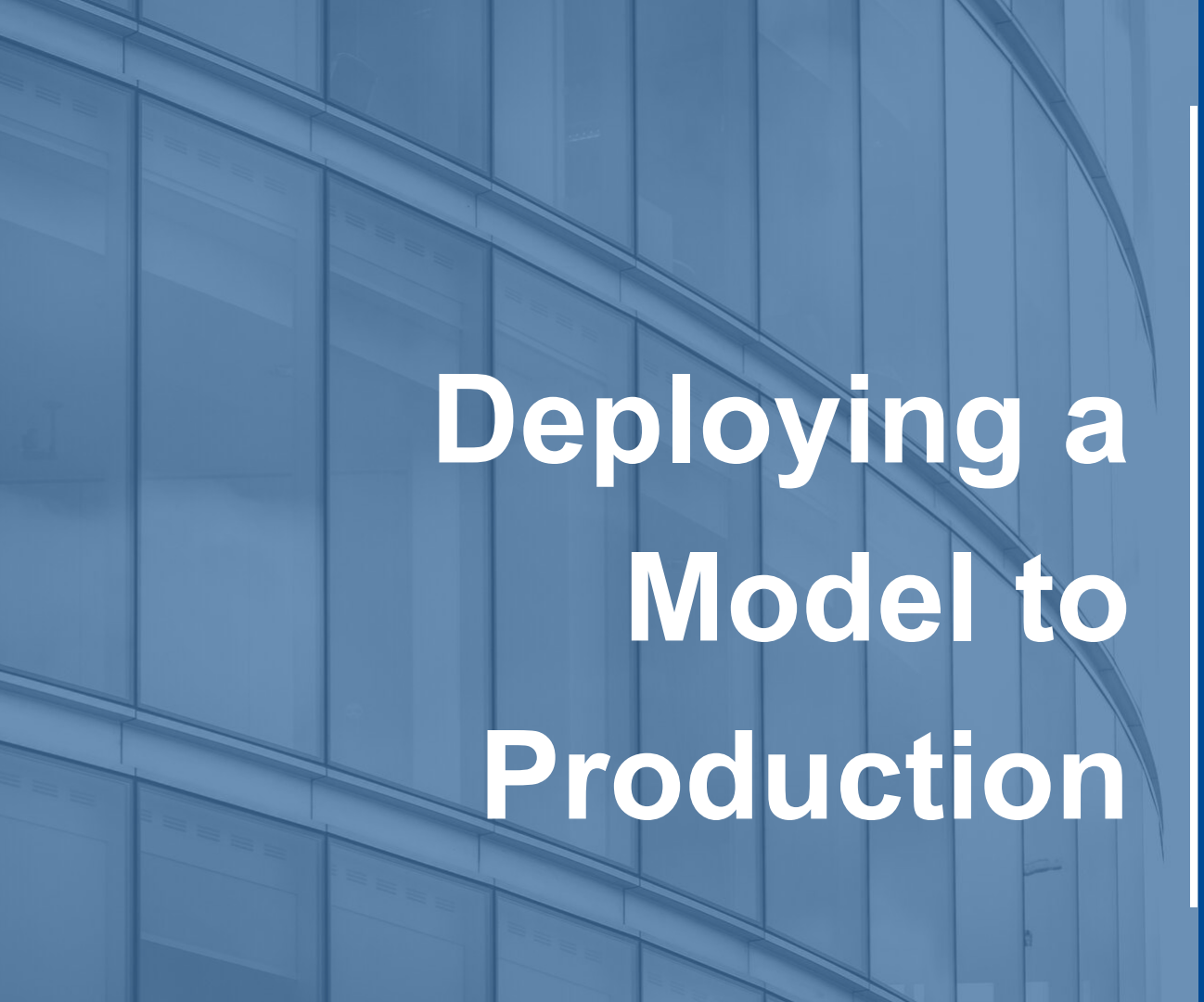
About the Notes

- Where applicable, lectures have the setup & commands included in the notes for easy copy and paste
- Additional reading suggestions
- Key concepts overview
- Much more

On Googling

A Humble Request

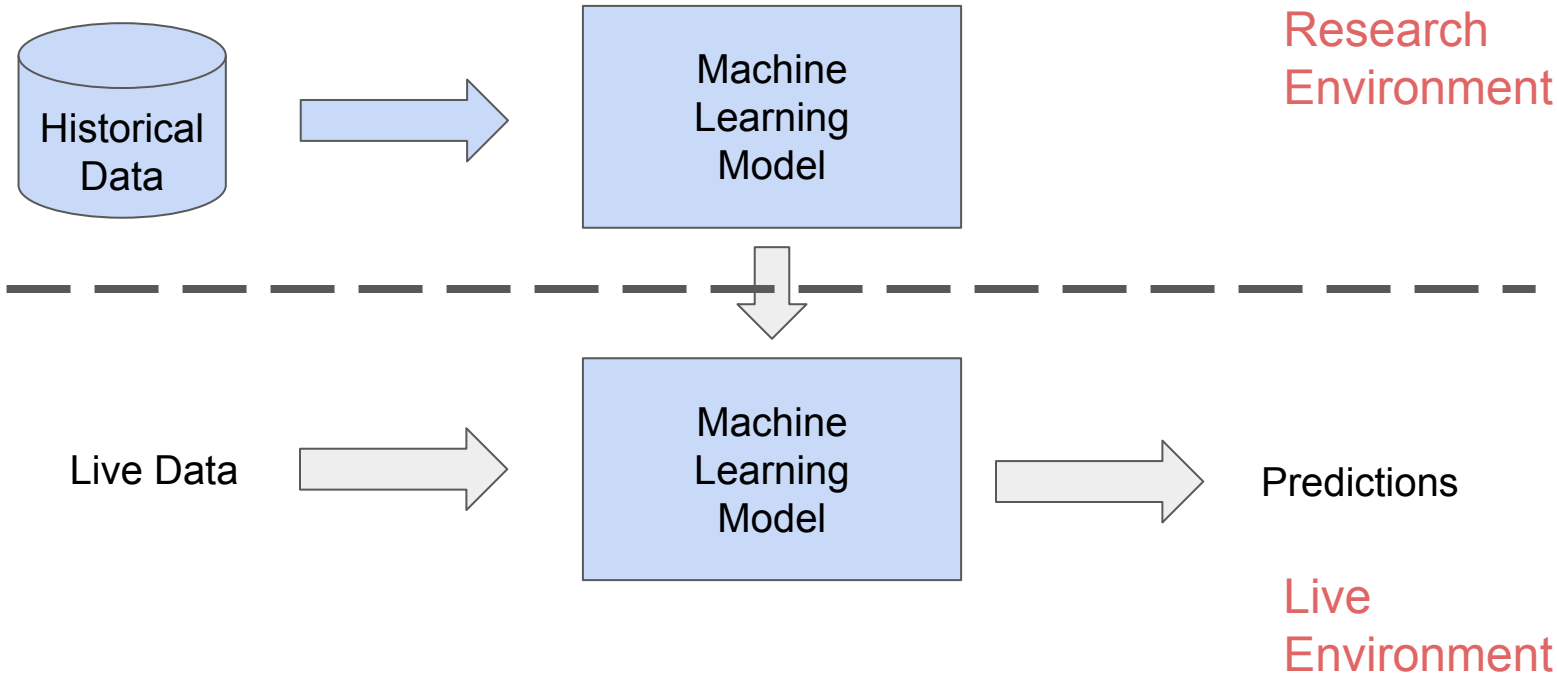
- We’re supporting multiple operating systems, 3 minor versions Python and multiple project dependencies
- We’ve done our best to include troubleshooting notes for any issues
- If you get stuck, please check the notes first, then try Googling (especially stackoverflow.com)
- If you’re still stuck, post in the Q&A and we’ll get back to you.



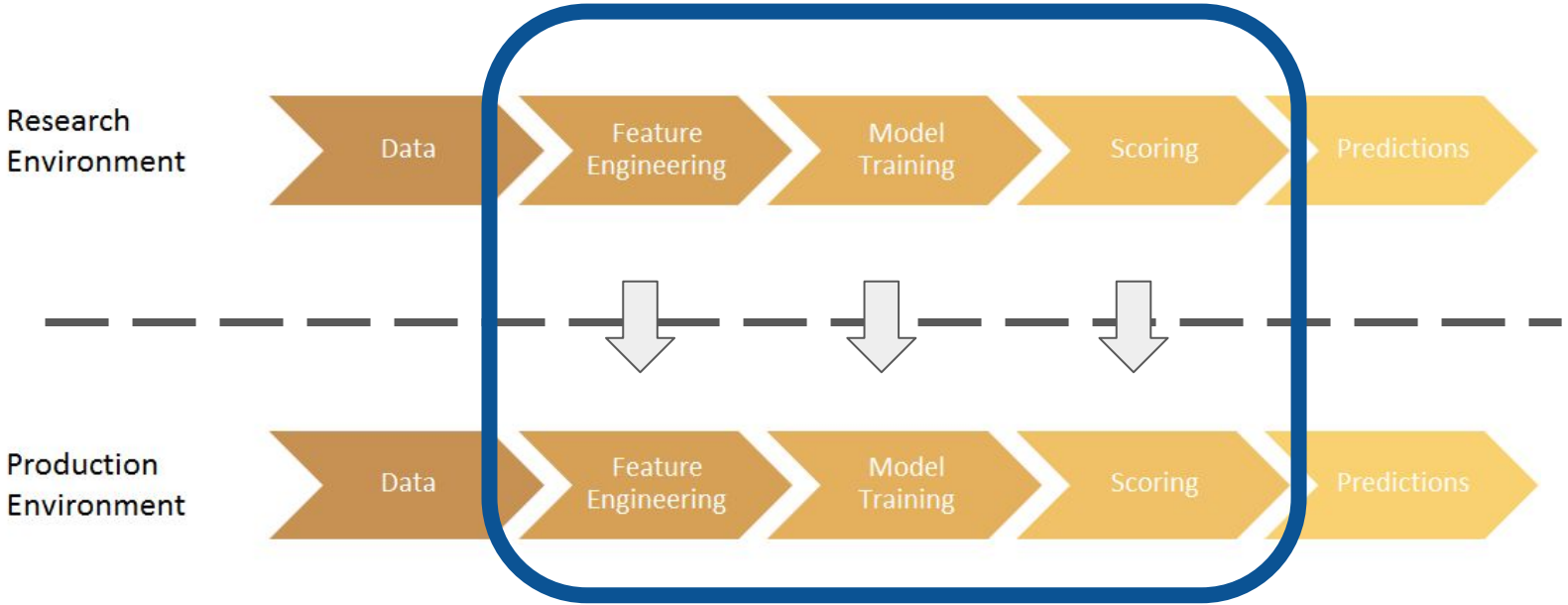
Deploying a Model to Production

Typical encountered scenarios

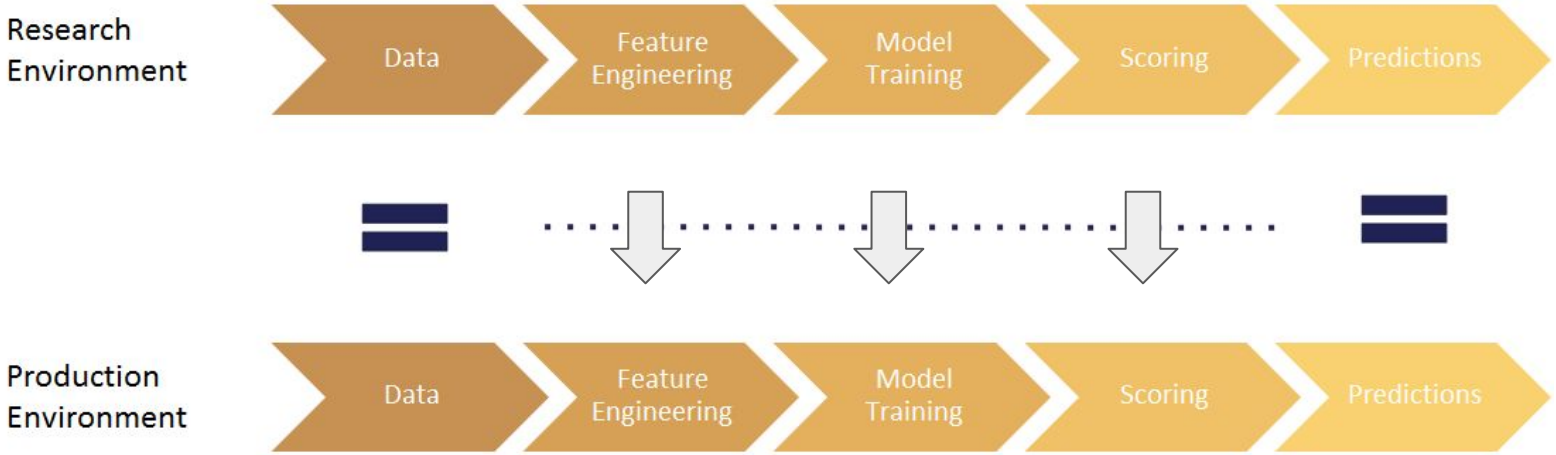
Deployment of ML models



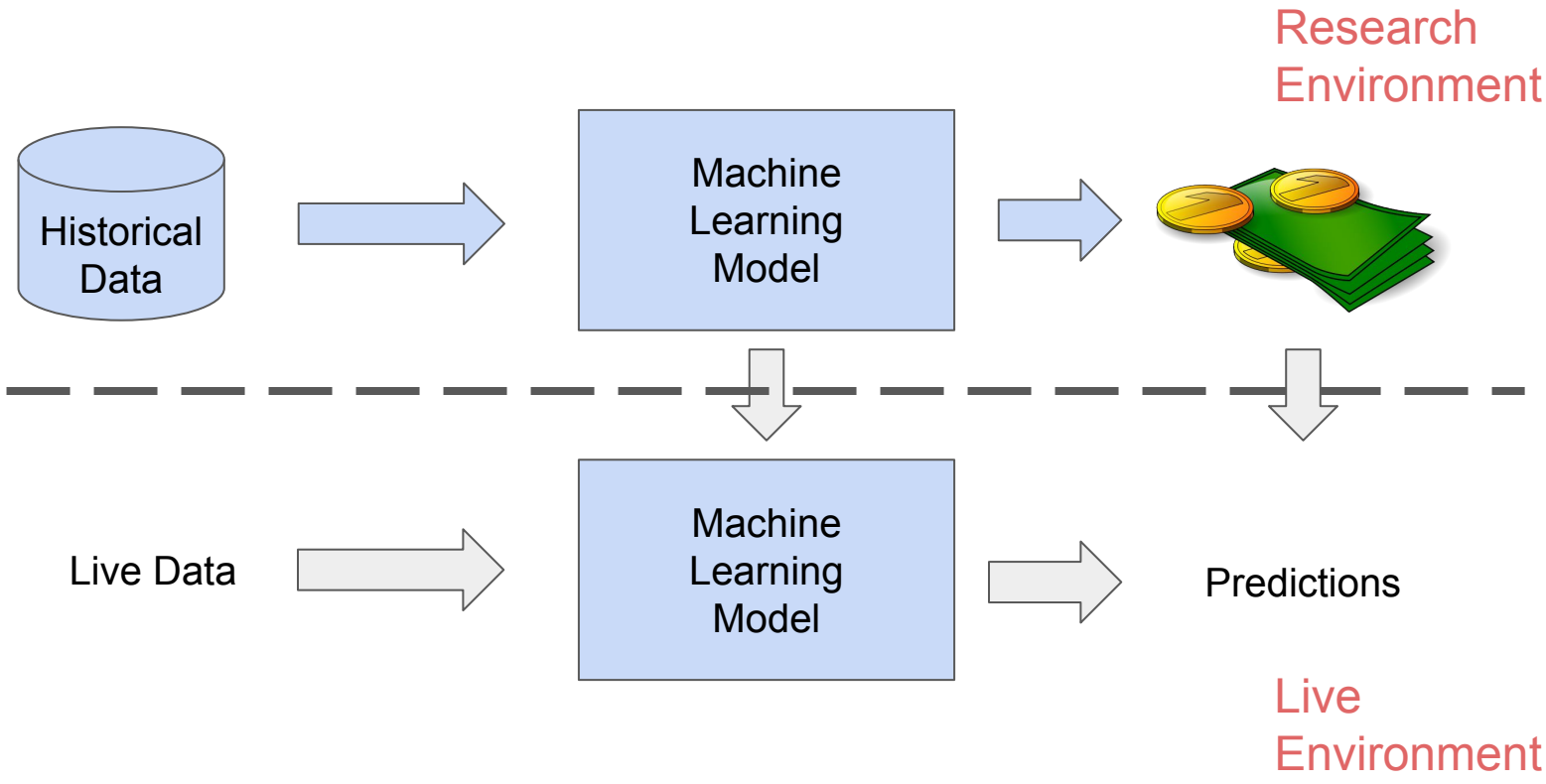
Deployment of ML pipelines



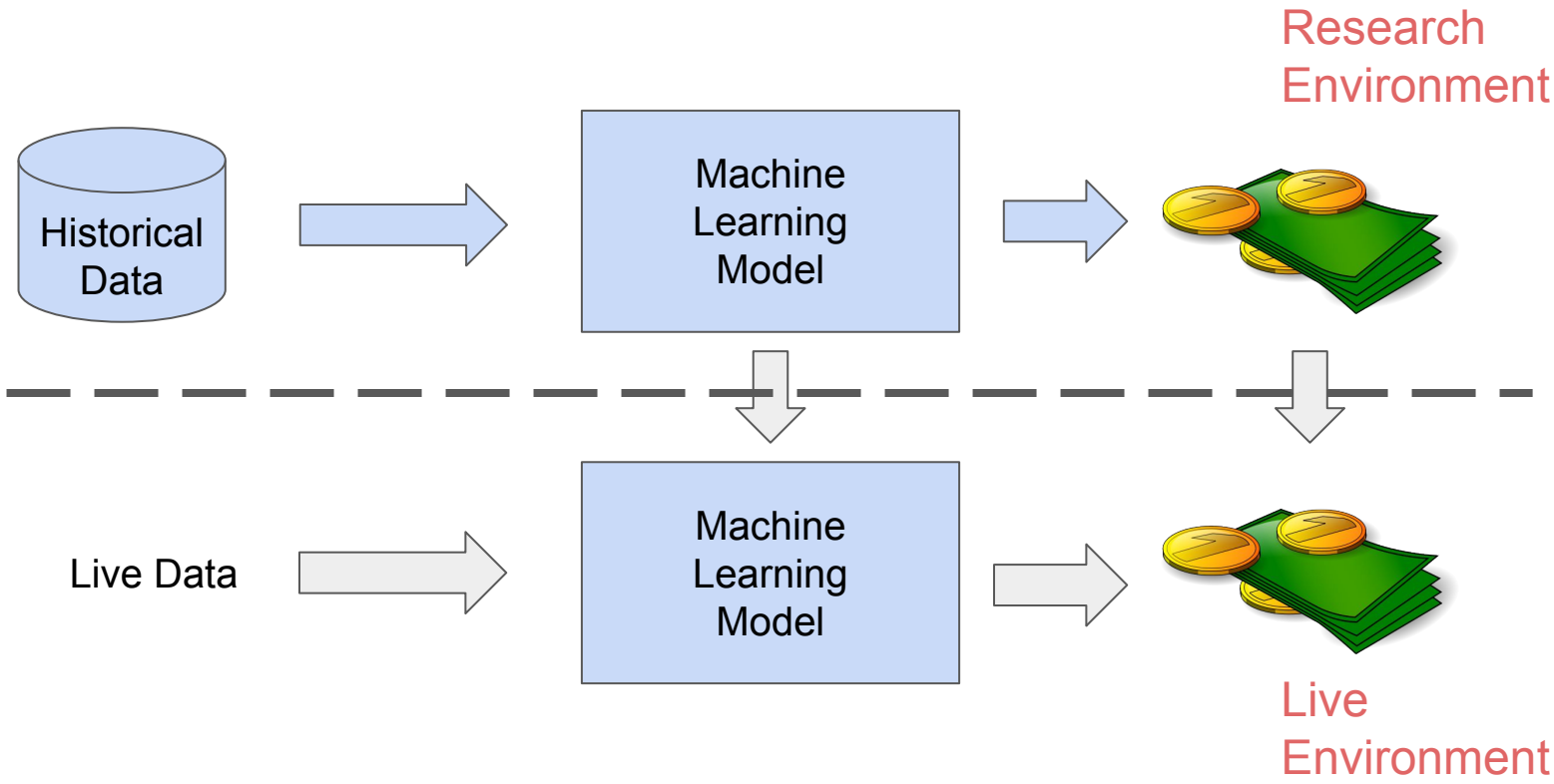
Reproducible ML pipelines



Deploying the value



Deploying the value



Scenarios we often encounter

Research
Environment

Deployment of
our first model

Live
Environment

Scenarios we often encounter

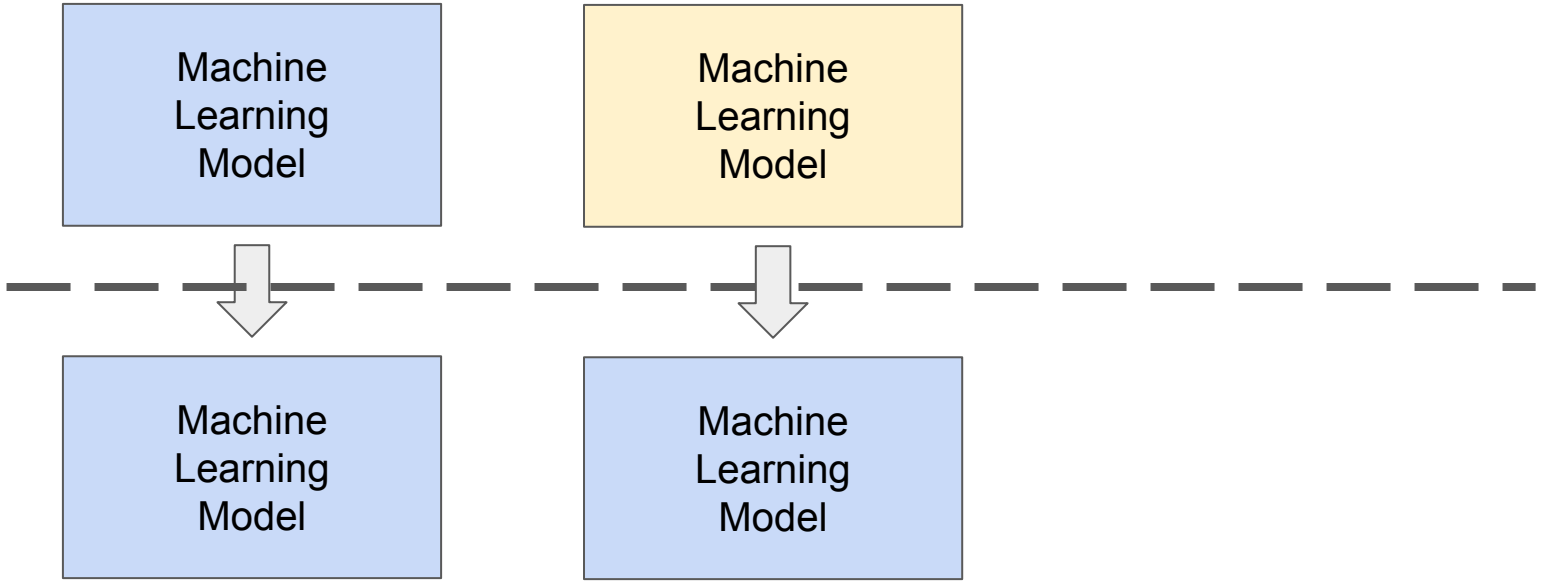
Research Environment

Deployment of our first model

Live Environment

Scenarios we often encounter

Research Environment



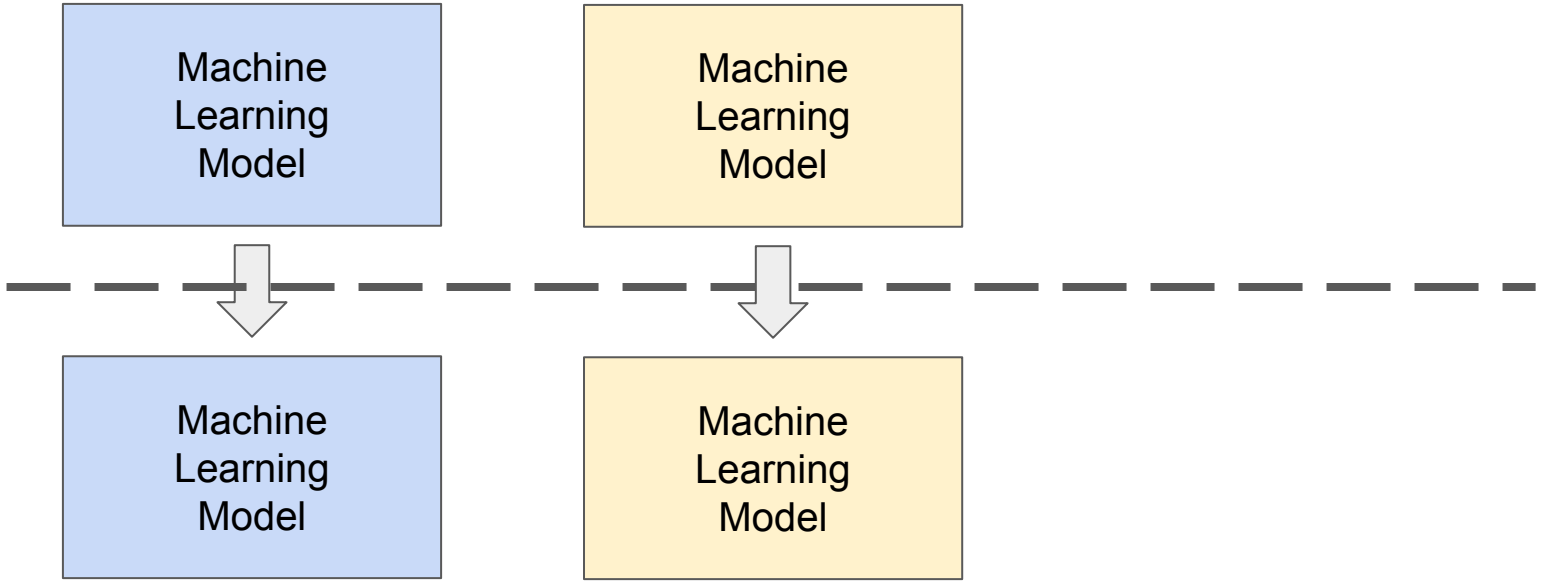
Deployment of our first model

Replacement of an existing model

Live Environment

Scenarios we often encounter

Research Environment

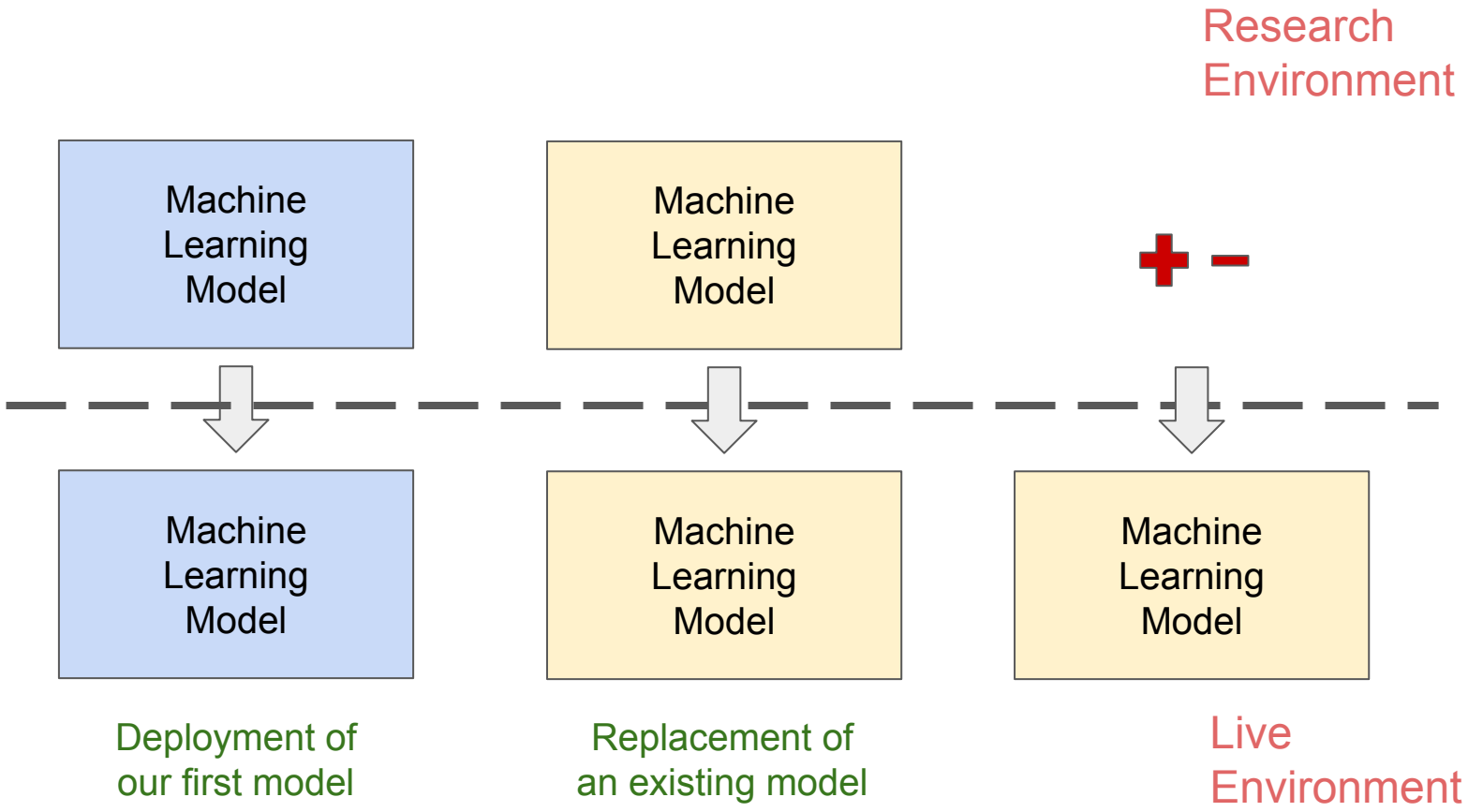


Deployment of our first model

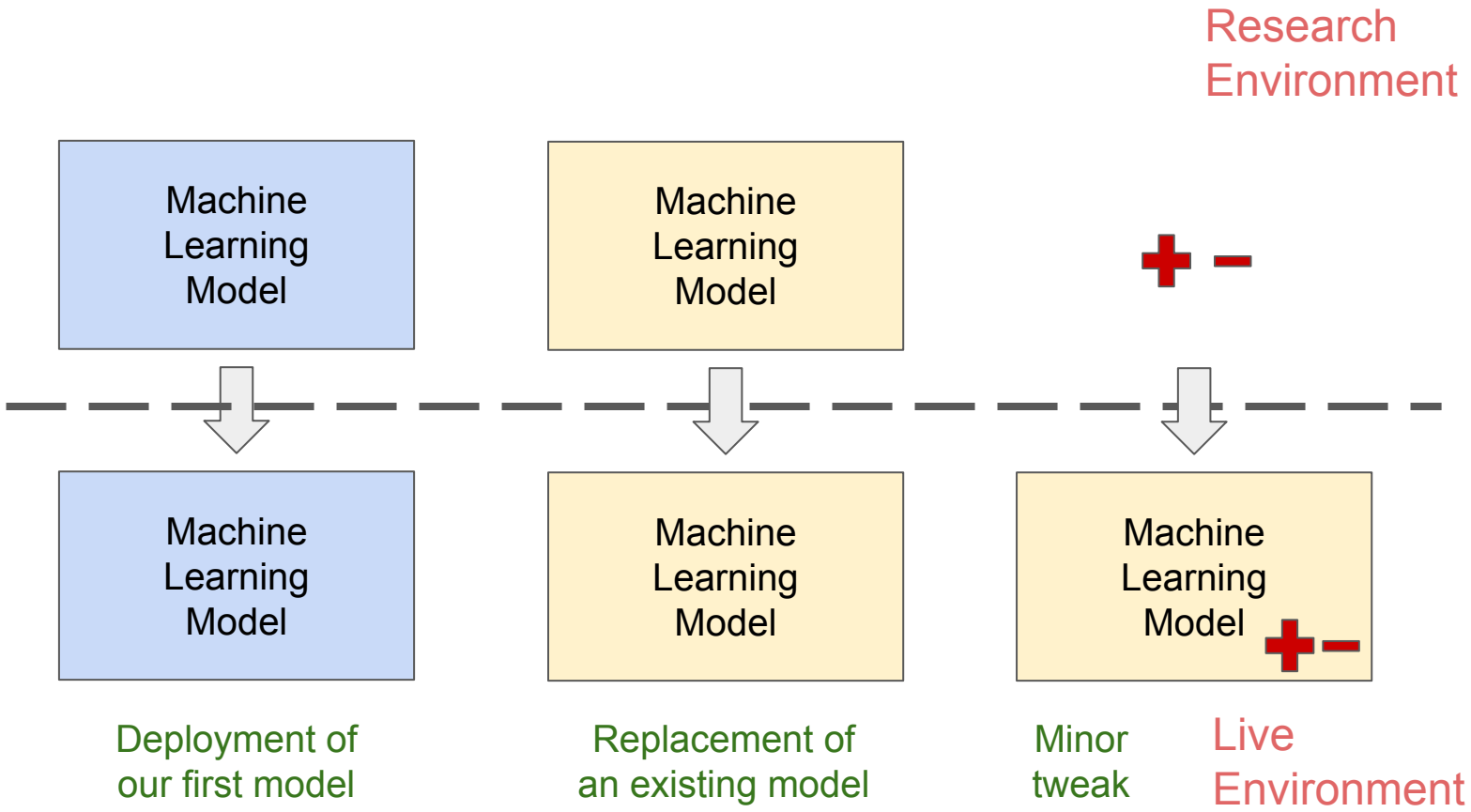
Replacement of an existing model

Live Environment

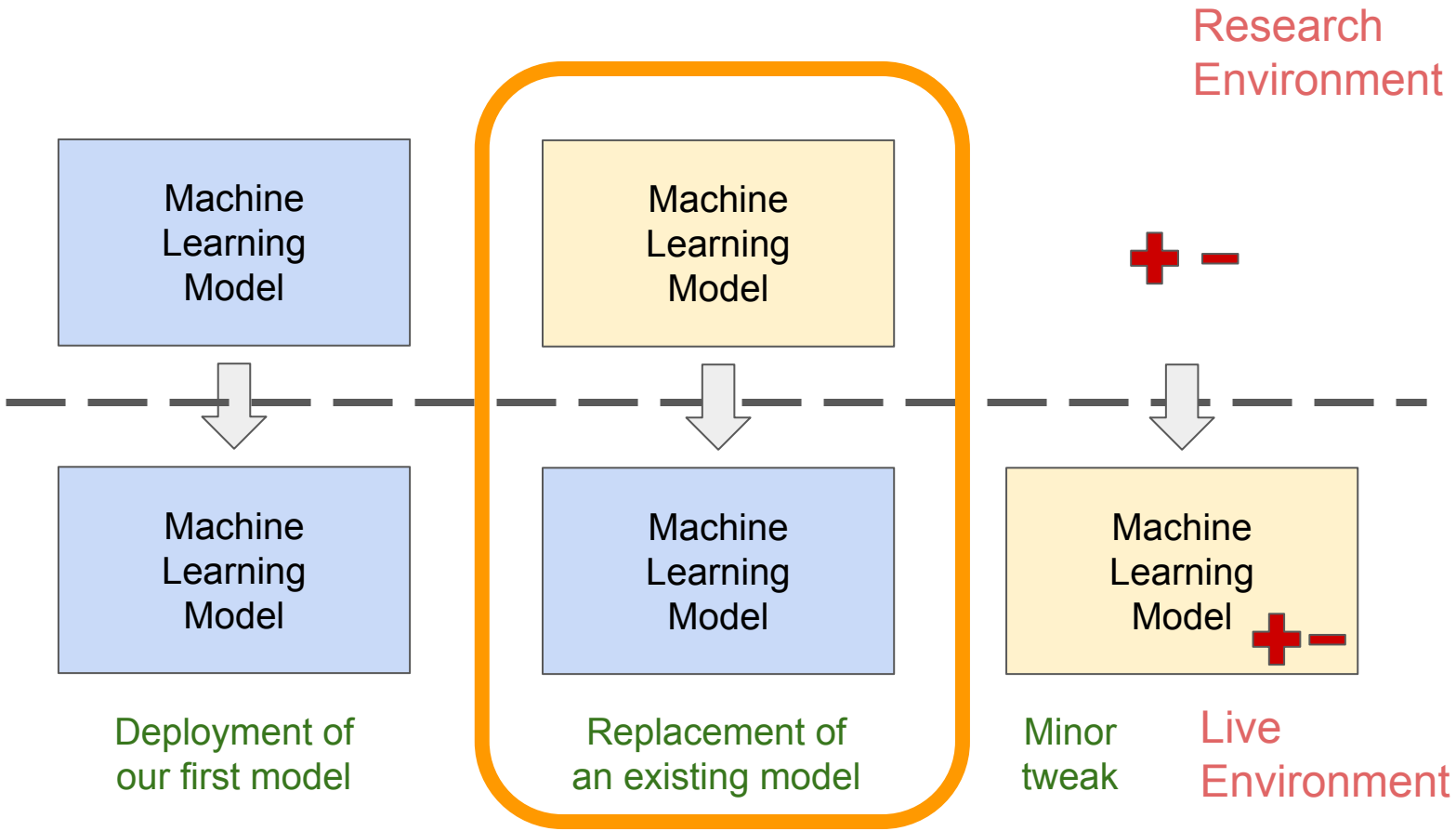
Scenarios we often encounter



Scenarios we often encounter



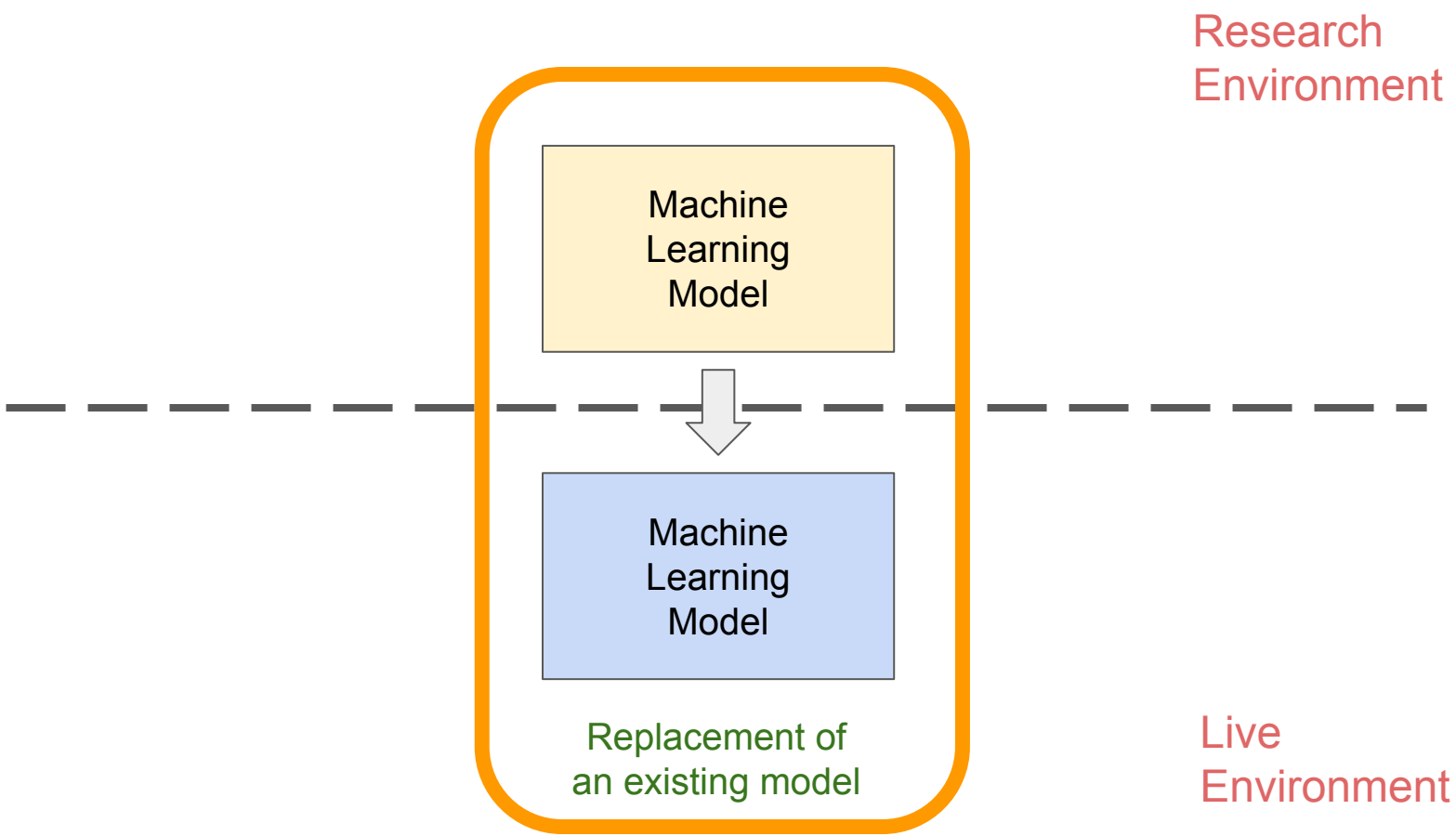
Scenarios we often encounter



Course Use Case - Our Scenario

Understanding what we want to achieve throughout the course

Replacing and existing live model



Estate Agents

- We put in touch house sellers with buyers
- We get a commission per house sale
  - % of house value

- Forecast our future revenue
- Predict House Sale Price

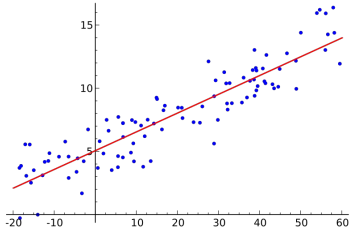
Our modelling data

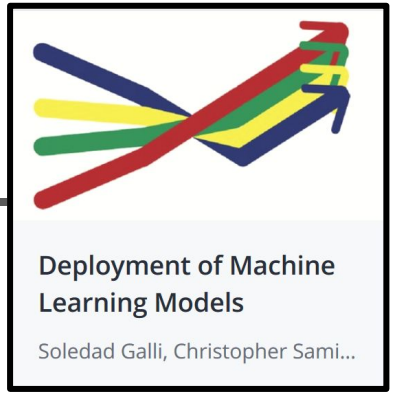
- We have historical data
- We know various characteristics of the houses we sold
  - Neighbourhood, when it was remodelled, the overall quality, etc
- We know the price for which the house was sold

Deploying the first ML model

Research Environment

Machine Learning Model





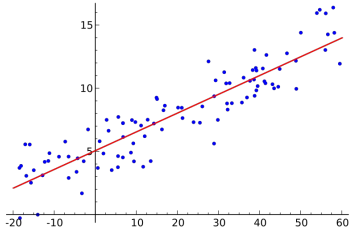
Deployment of our first model

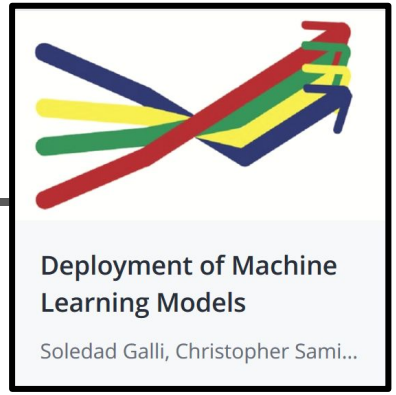
Live Environment

Deploying the first ML model

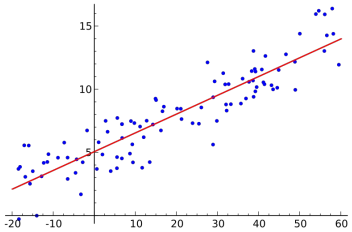
Research Environment

Machine Learning Model





Machine Learning Model



Deployment of our first model

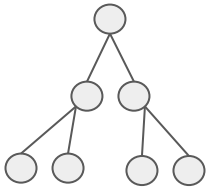
Live Environment

Our first model is in production

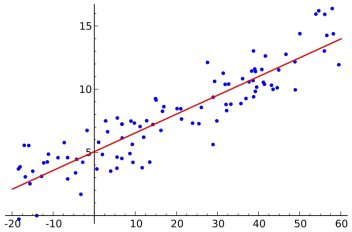
Replacing our current model by a better one

Research Environment

Machine Learning Model



Machine Learning Model

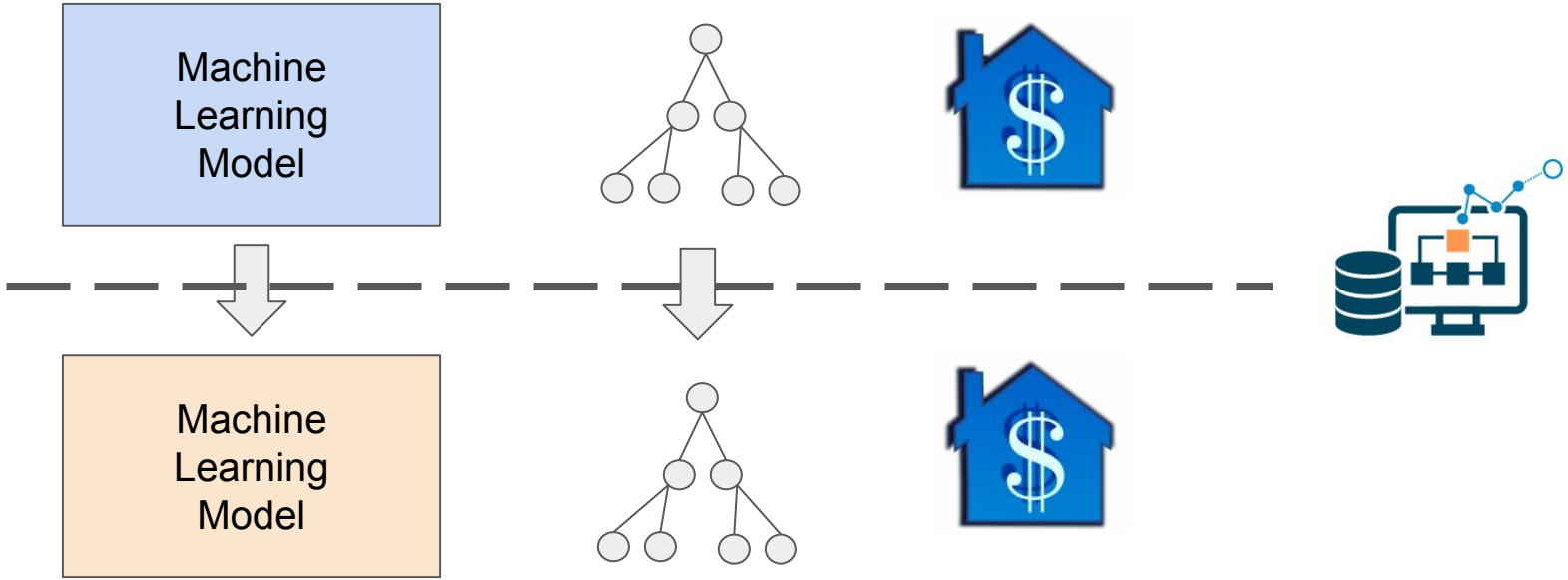


Replacement of existing model

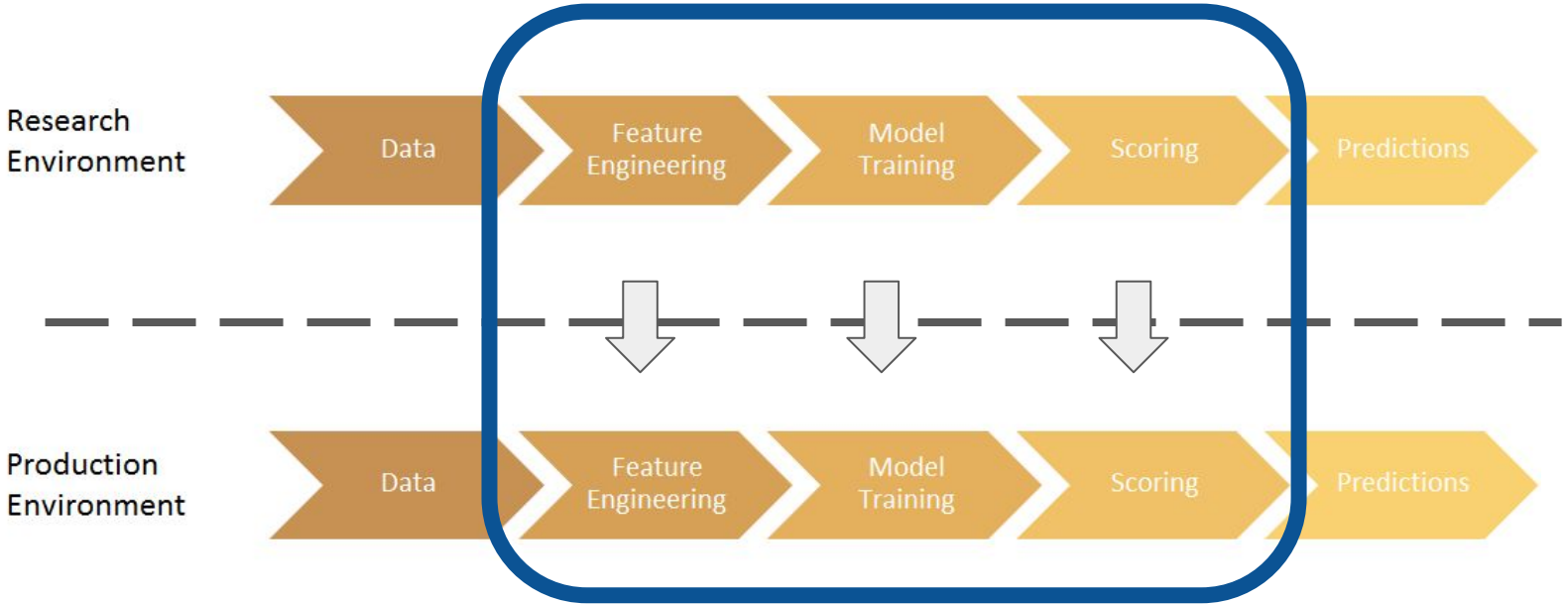
Live Environment

Replacing our current model by a better one

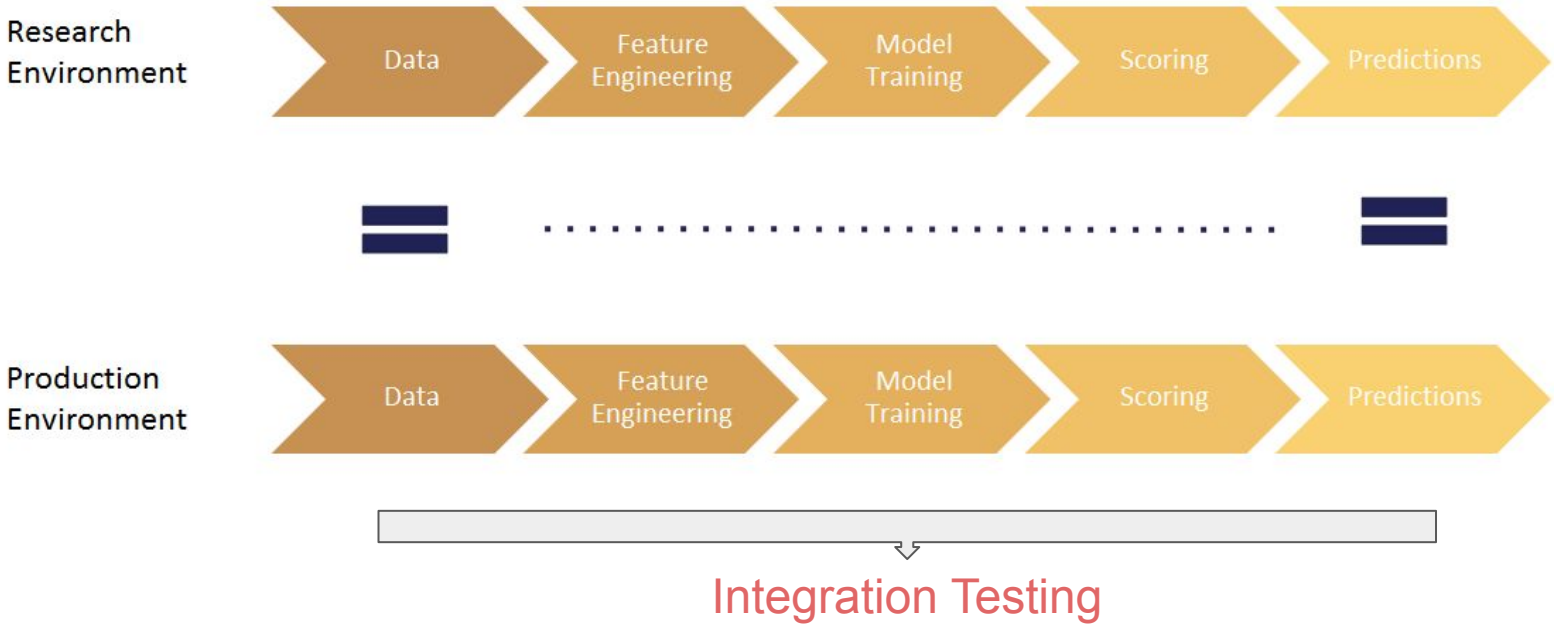
Research Environment



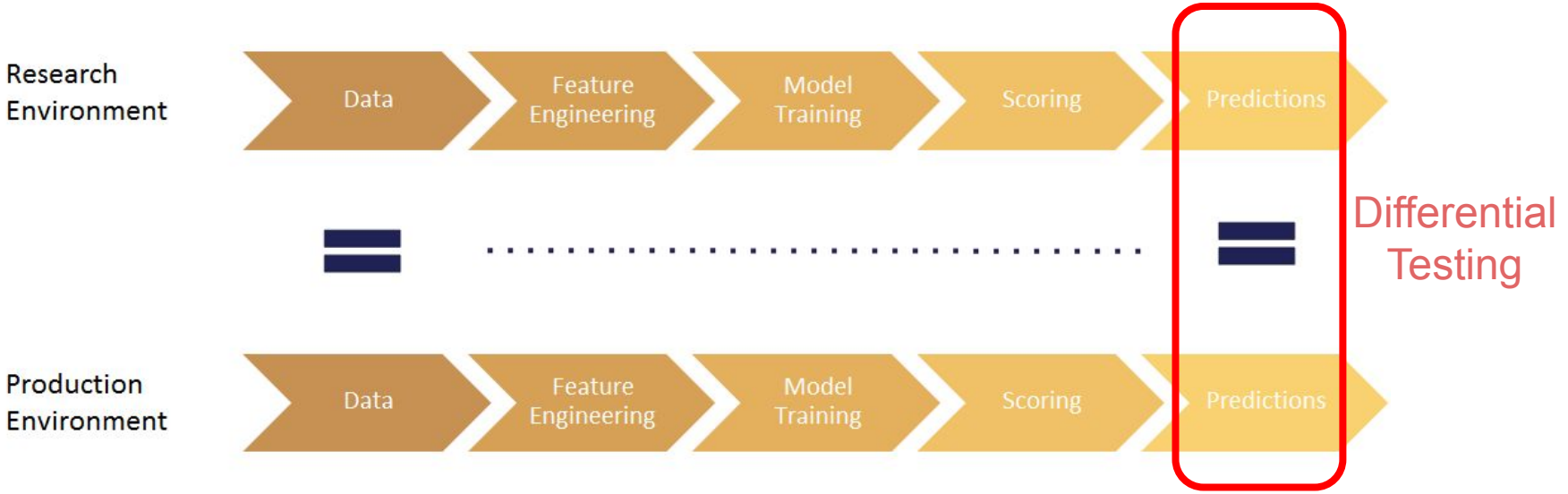
Maximising reproducibility



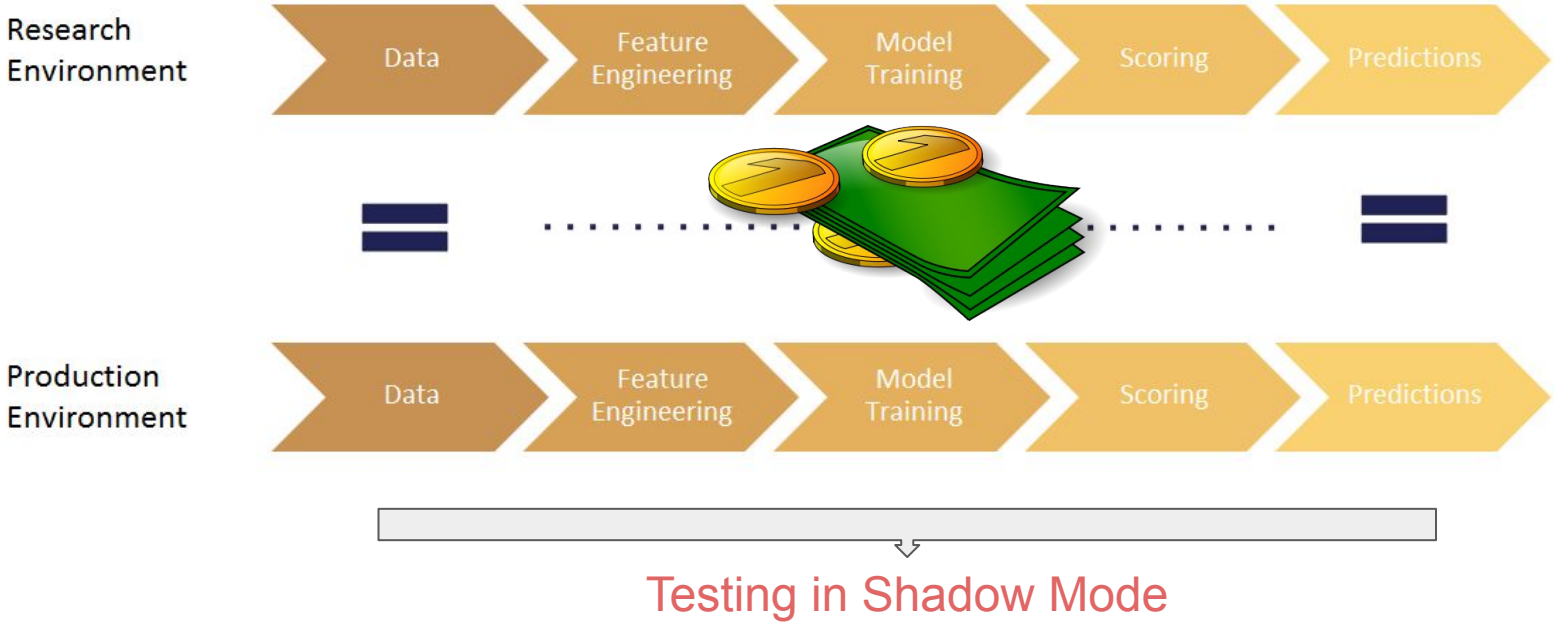
Maximising reproducibility



Maximising reproducibility



Maximising reproducibility

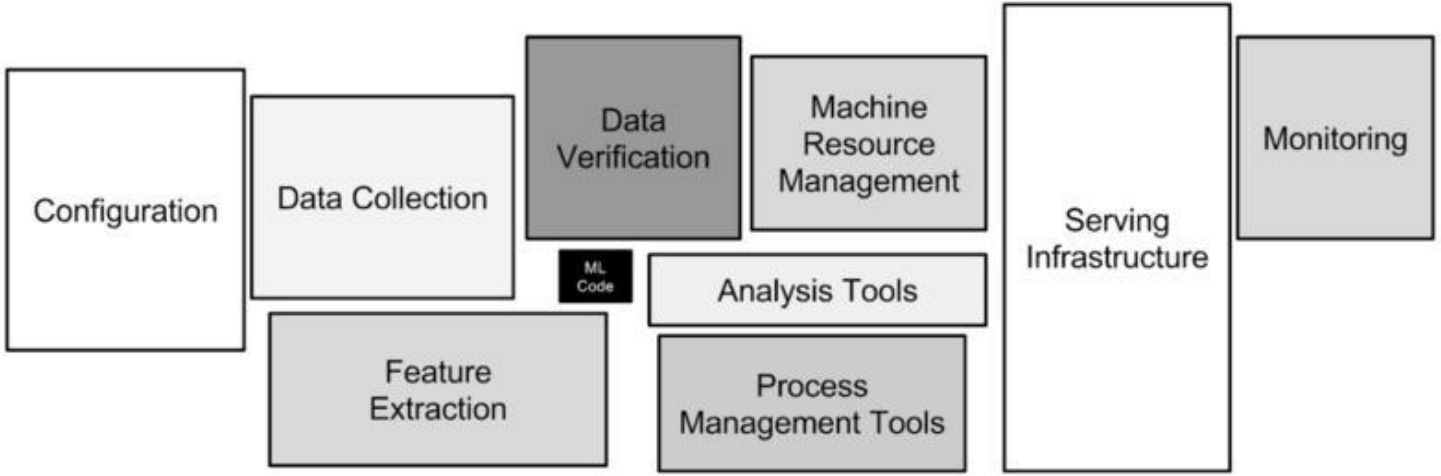


Continuous model performance monitoring

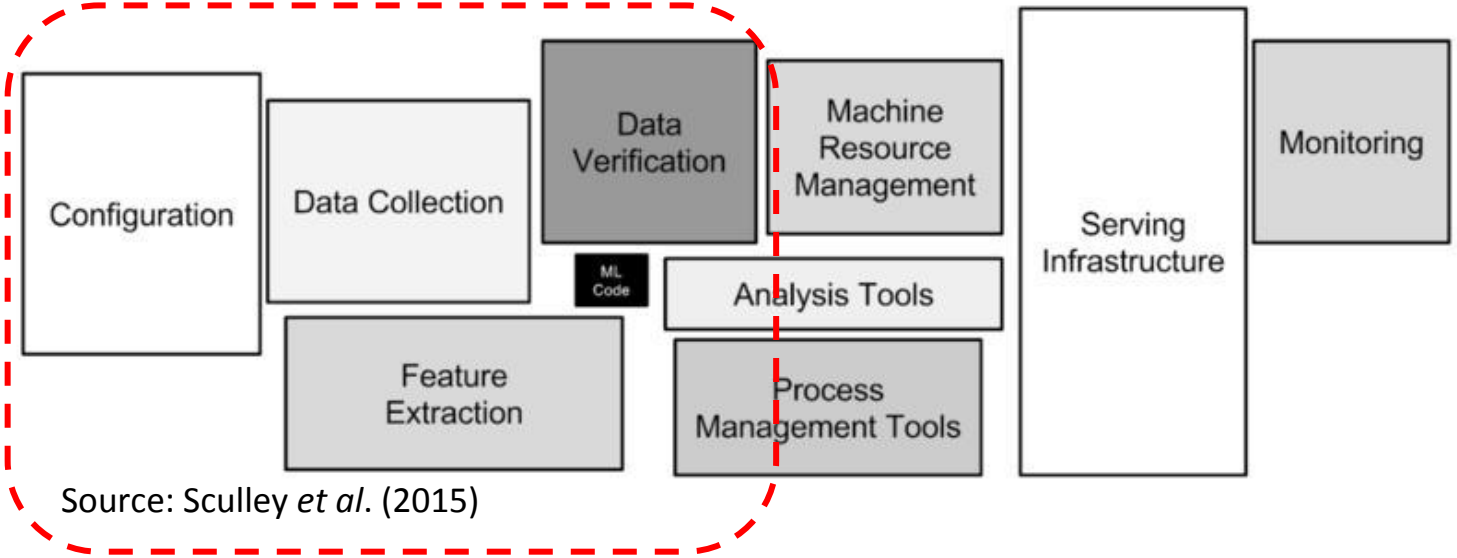
ML System Testing & Monitoring Lifecycle

Course Context

Effective ML Model Deployments are about Entire Systems
"Hidden Technical Debt in ML Systems"



Source: Sculley *et al*. (2015)



Source: Sculley *et al*. (2015)

Machine Learning System Testing and Monitoring

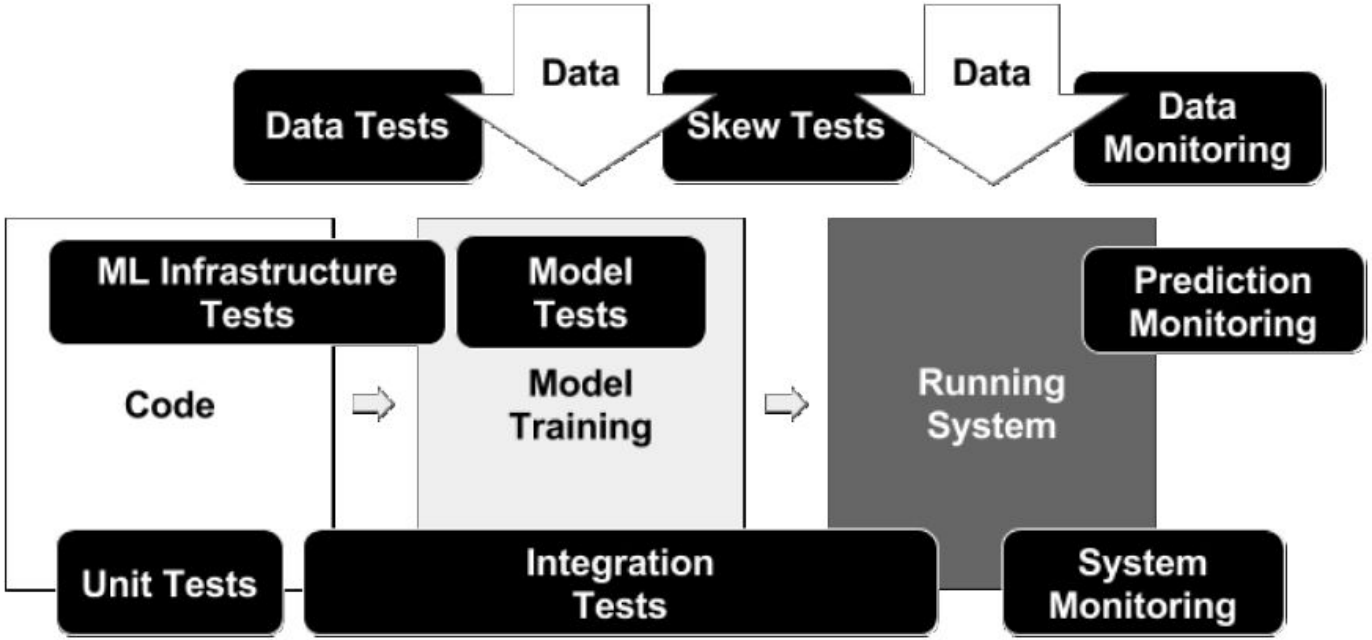


Image taken from: The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction' (2017) Breck *et al*. IEEE International Conference on Big Data

Machine Learning System Testing and Monitoring

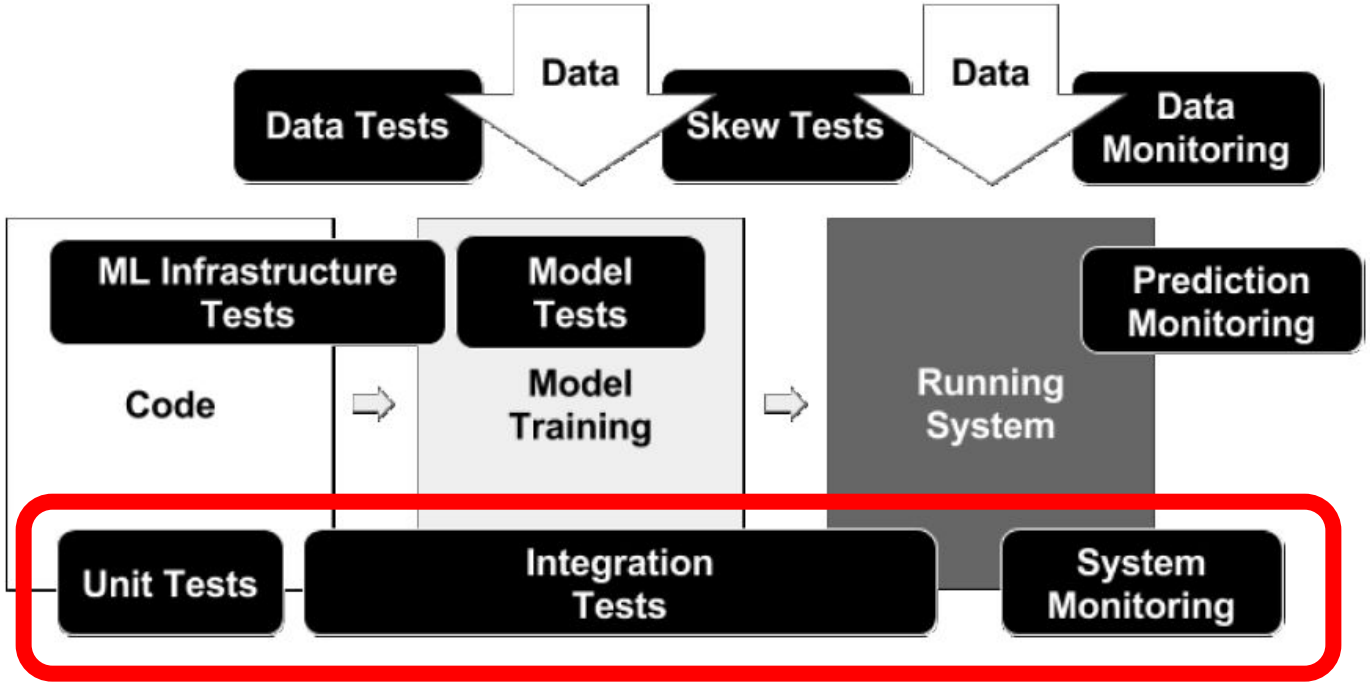


Image taken from: The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction' (2017) Breck *et al.* IEEE International Conference on Big Data

Machine Learning System Testing and Monitoring

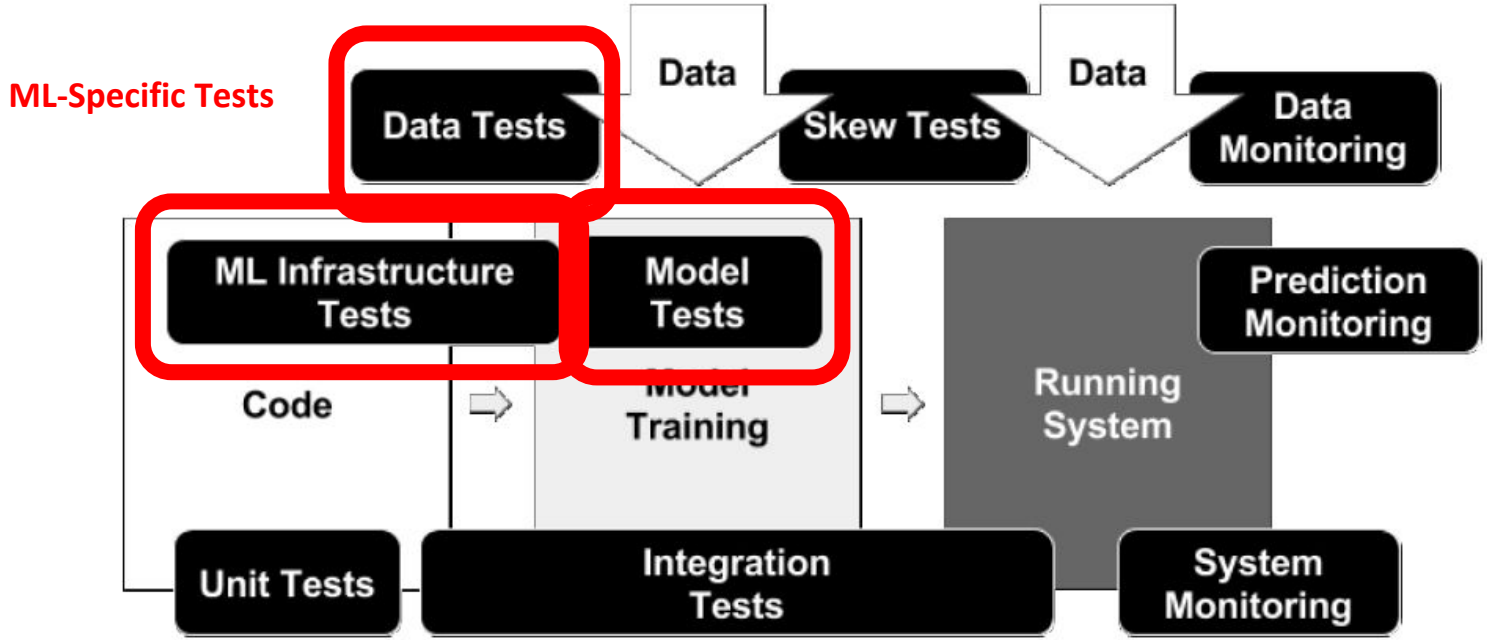
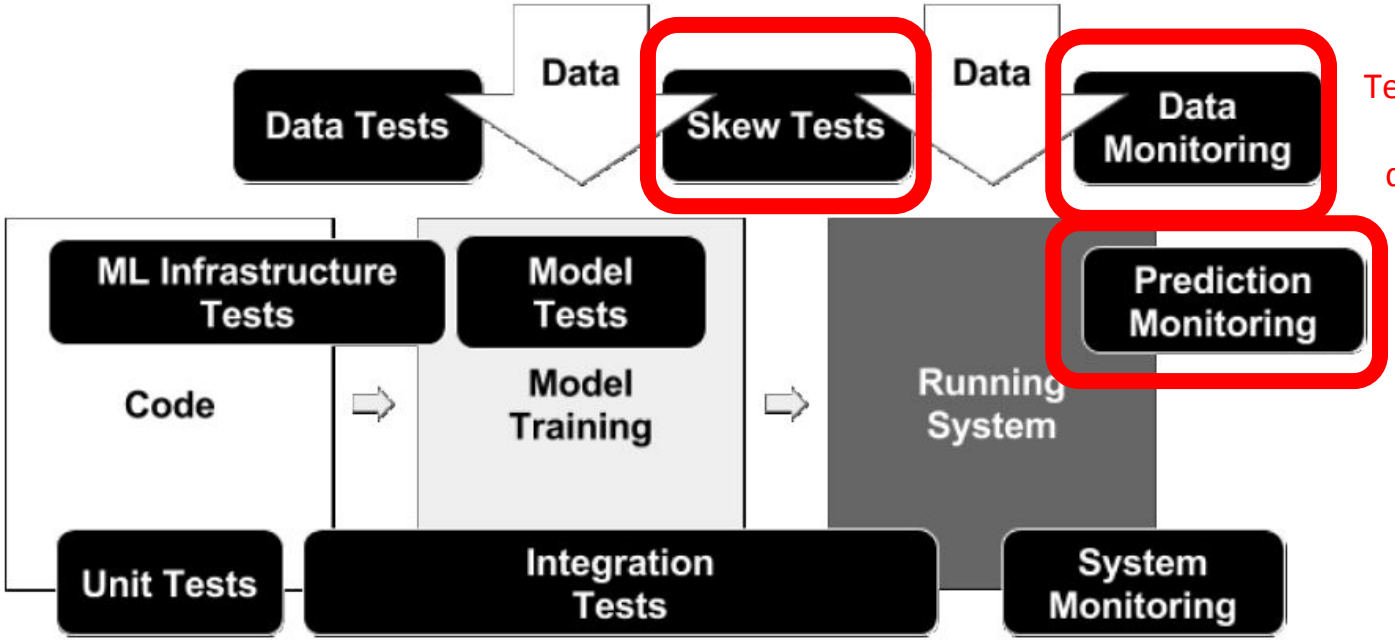


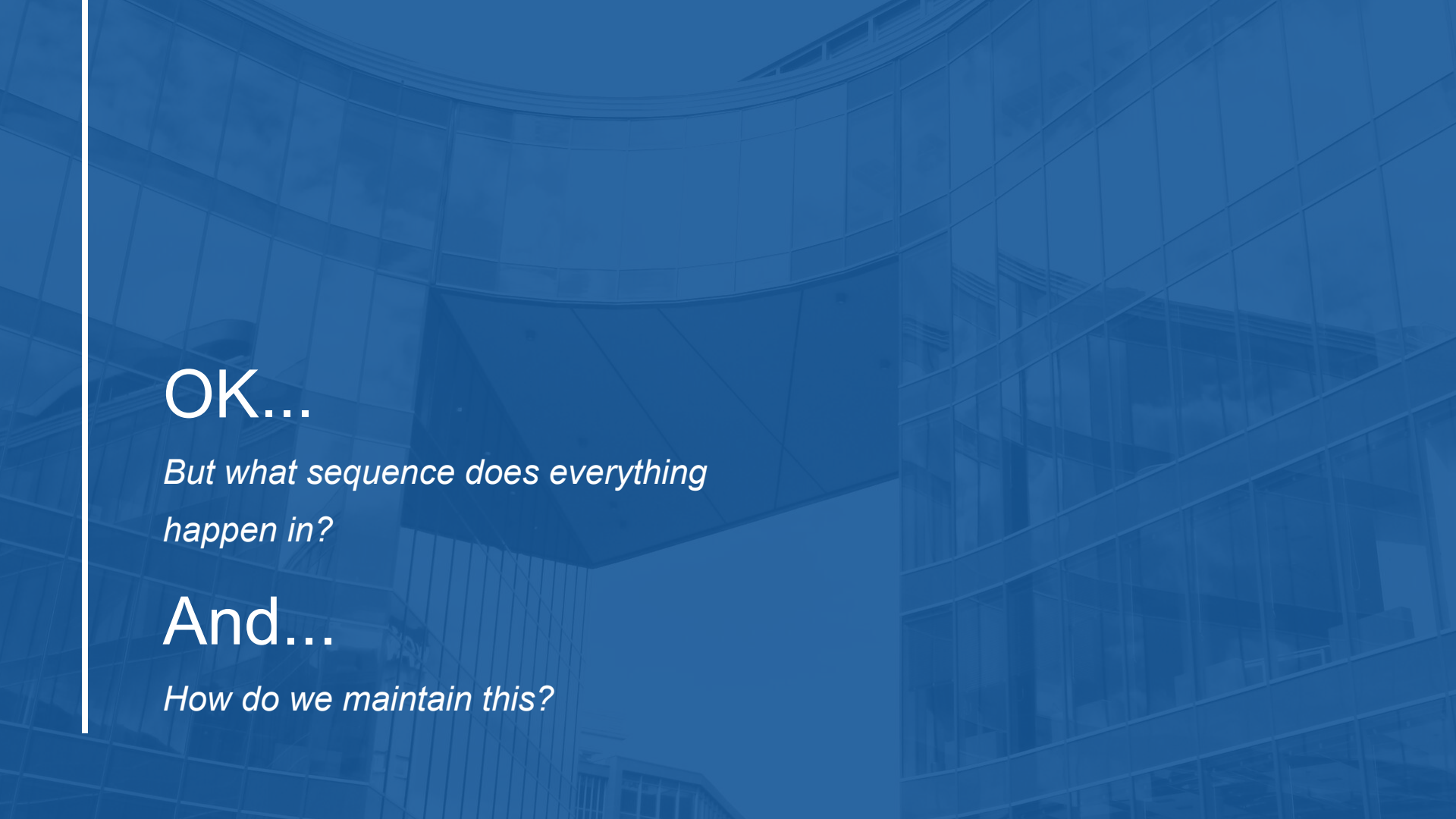
Image taken from: The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction' (2017) Breck *et al.* IEEE International Conference on Big Data

Machine Learning System Testing and Monitoring



Tests & Monitoring during Shadow deployment and **afterwards**

Image taken from: The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction' (2017) Breck *et al.* *IEEE International Conference on Big Data*



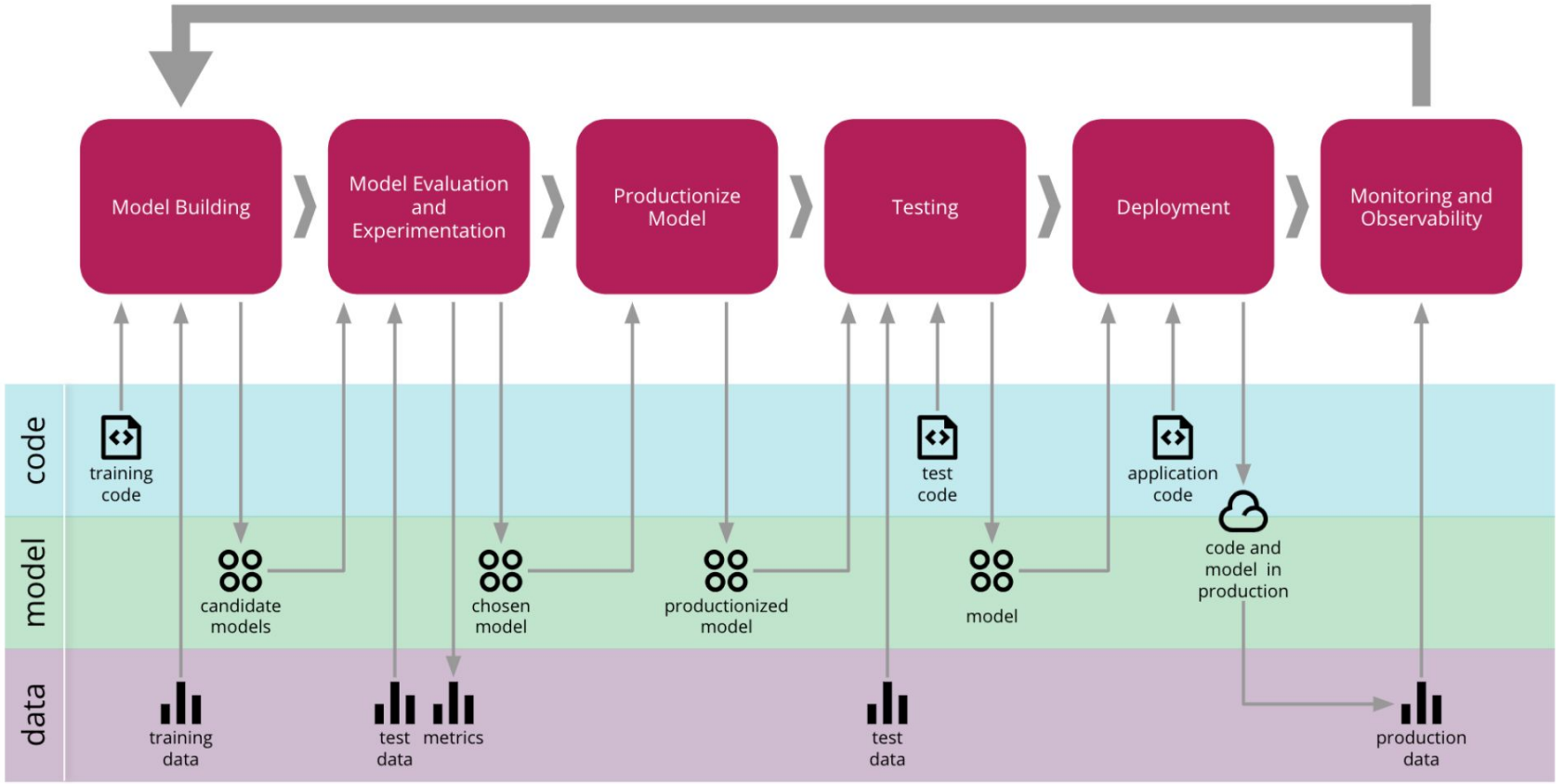
OK...

But what sequence does everything happen in?

And...

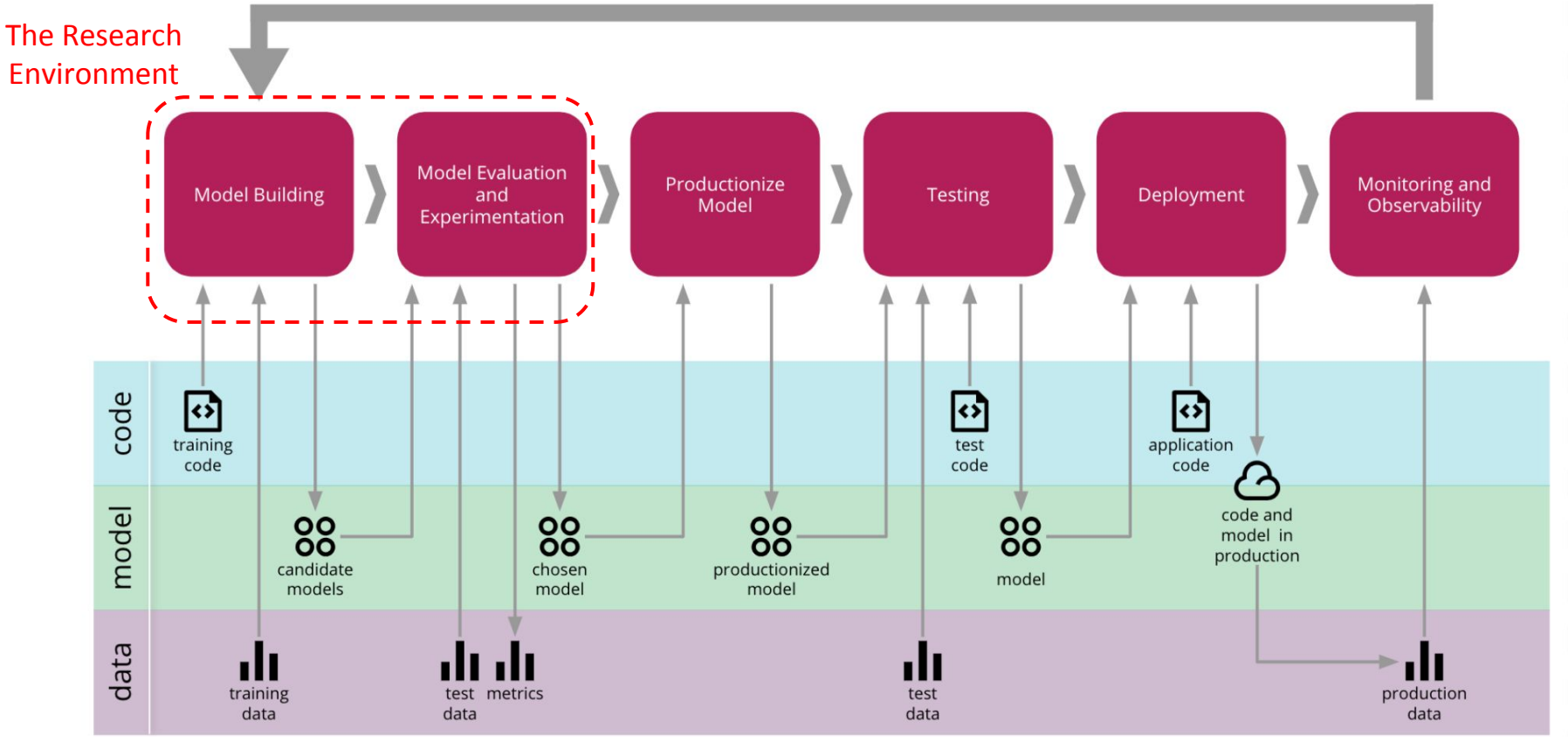
How do we maintain this?

ML System Lifecycle (CD4ML)



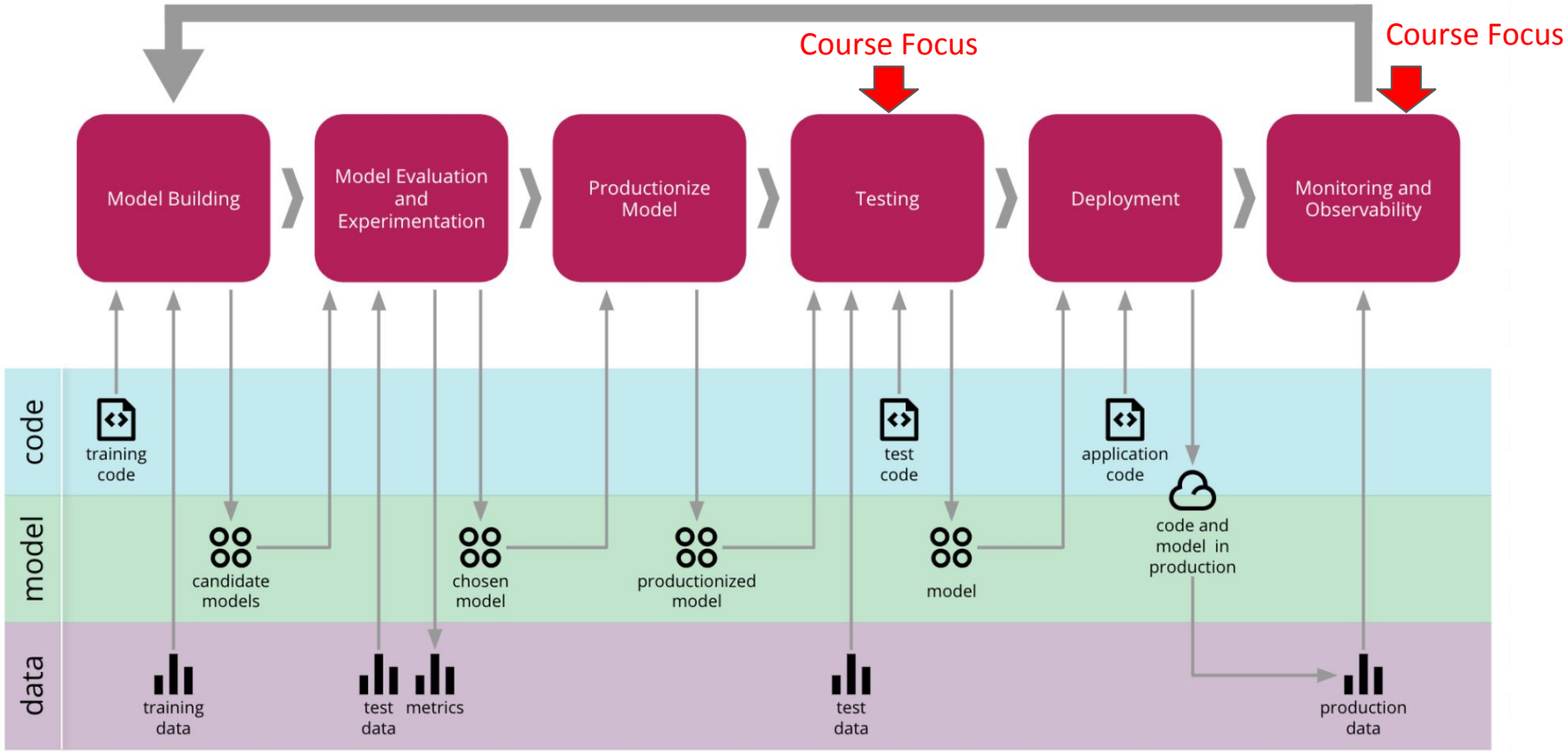
Source: https://martinfowler.com/articles/cd4ml.html

ML System Lifecycle (CD4ML)



Source: https://martinfowler.com/articles/cd4ml.html

ML System Lifecycle (CD4ML)



Source: https://martinfowler.com/articles/cd4ml.html

Section Overview

Testing Concepts

- Testing Scope In This Course
- Testing Theory
- ML System Testing Challenges
- Jupyter Installation and Setup
- Hands-on Notebooks

Testing Scope

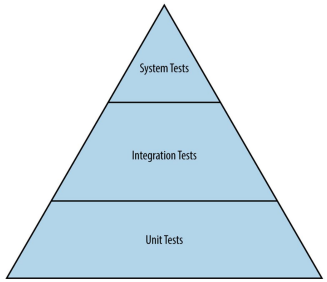
Situating Ourselves

Building an ML System: Key Phases

The Research Environment

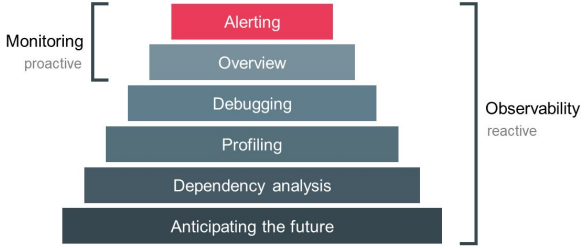
- Testing different algorithms
- Testing different configurations and hyperparameters
- Testing different data and features

Development Environment



- Unit, integration and acceptance tests
- Differential tests
- Benchmark tests
- Load tests

Production Environment

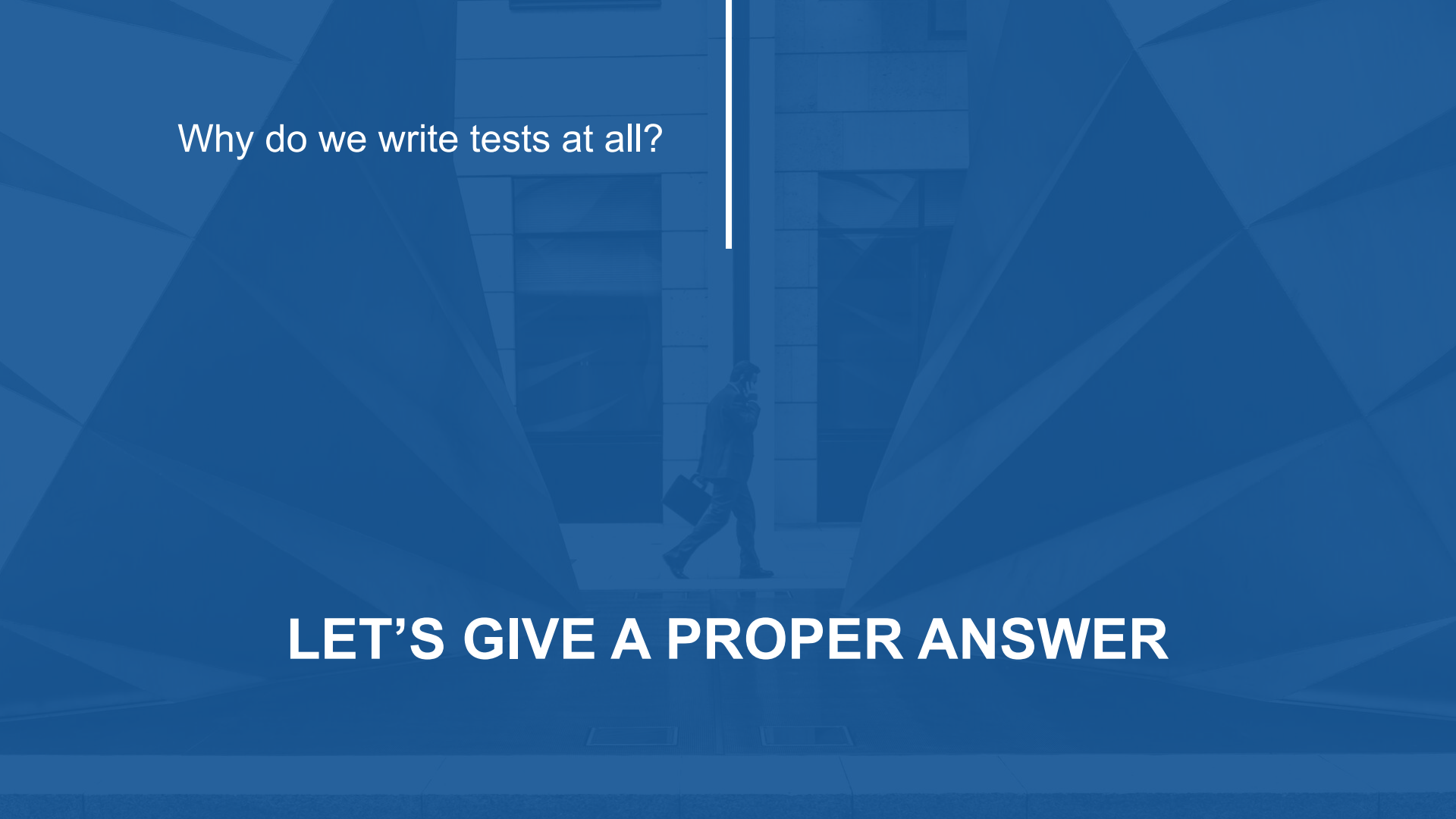


- Shadow mode testing
- Canary releases
- Observability & Monitoring
- Logging & Tracing
- Alerting

Why Test?

What’s all the fuss about?

Why do we write tests at all?



LET’S GIVE A PROPER ANSWER

Confidence

How to measure confidence

- Analyzing data provided by monitoring historic system behavior
- Making predictions from data about past system behavior

Have confidence in your system

Predicting

Your System's Future Reliability

- Most software systems are frequently changed
- The challenge is being able to confidently describe these changes **and their impact**

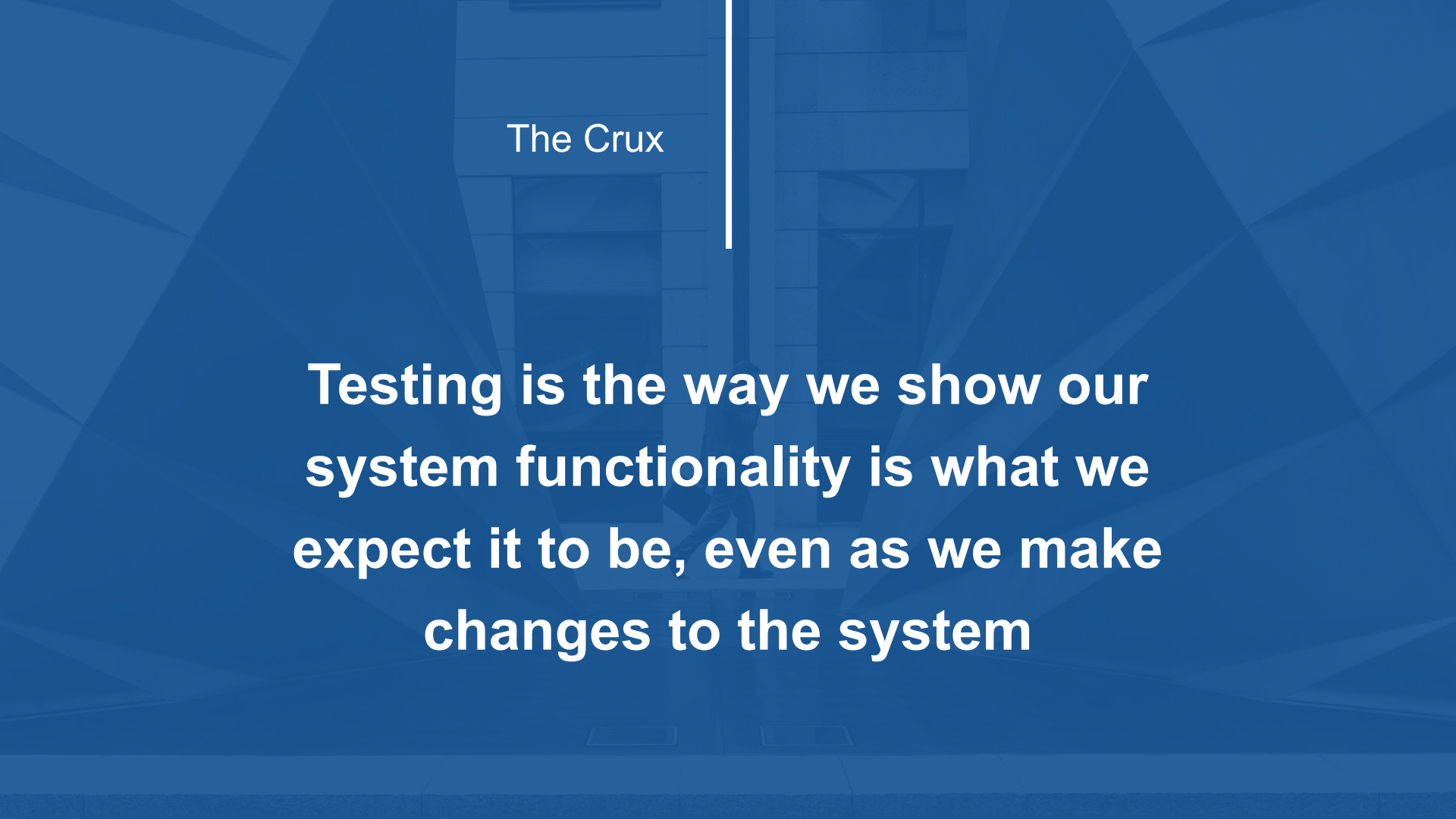
Analyse the uncertainty incurred by any system change

Functionality

What change are we tracking?

- It’s not just which code lines have changed (although version control is important).
- It’s about the entire system functionality, from user interface through the database.

Confidence that functionality remains unchanged.



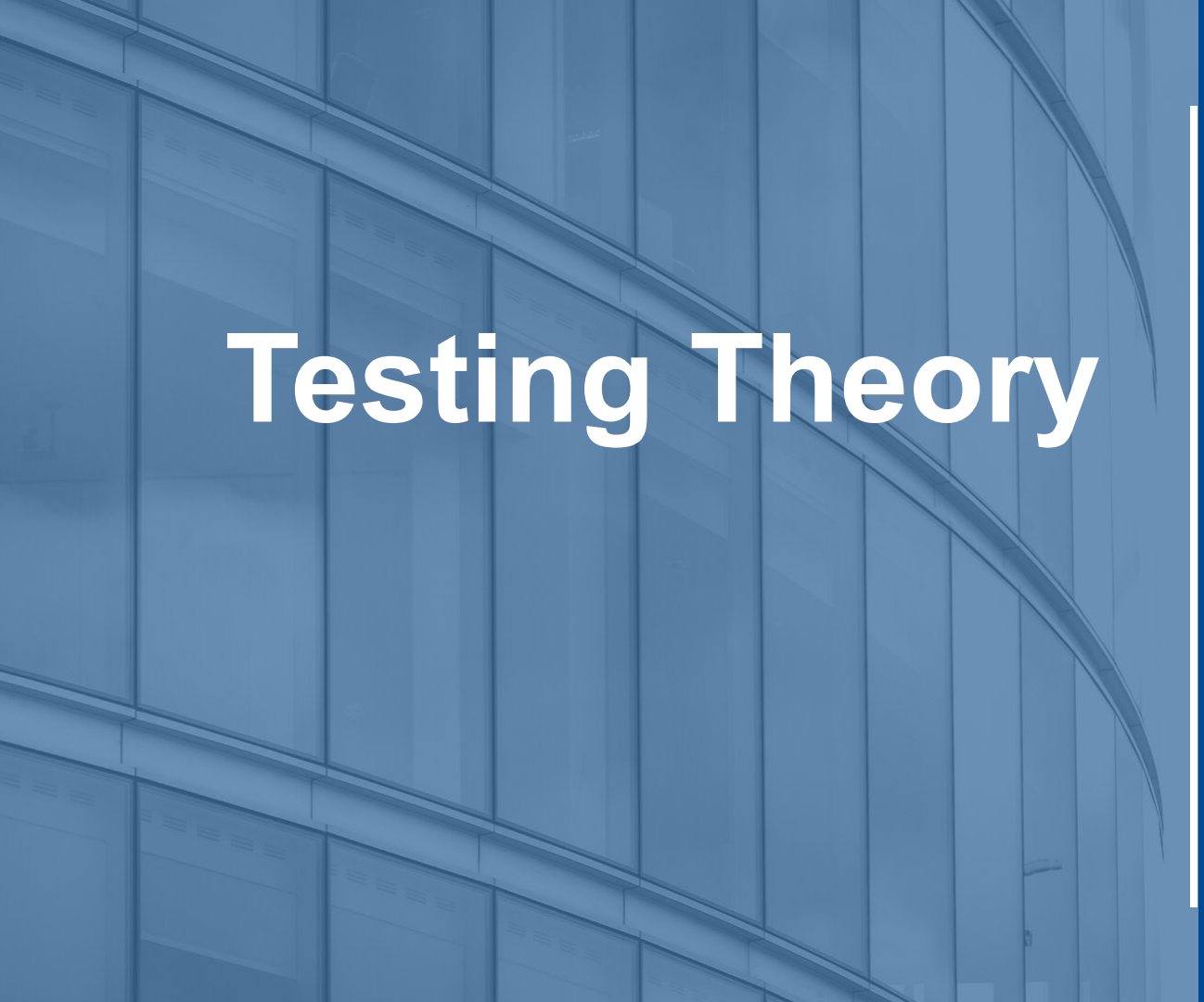
The Crux

Testing is the way we show our system functionality is what we expect it to be, even as we make changes to the system

The Credit Card Analogy

Summary

- The real value of testing occurs with system change
- Done correctly, each tests reduces the uncertainty when analysing a change to the system
- This is particularly true with ML systems...



Testing Theory


Unit Test
Fuzz Test
Functional Test
Canary Tests
Contract Test
End to End Test
Benchmark Test
Load Test
Soak Test
Coverage Test
Mutation Test
Acceptance Test
Component Test
TDD
Integration
Regression Test
Contract Test
Config Test
Stress Test
Smoke Test

The Testing Pyramid

Detailed in Google's Site Reliability Engineering (SRE) Handbook:

https://landing.google.com/sre/sre-book/chapters/testing-reliability/

How should we conceptualize our tests?

How Much Testing?

Finding a balance

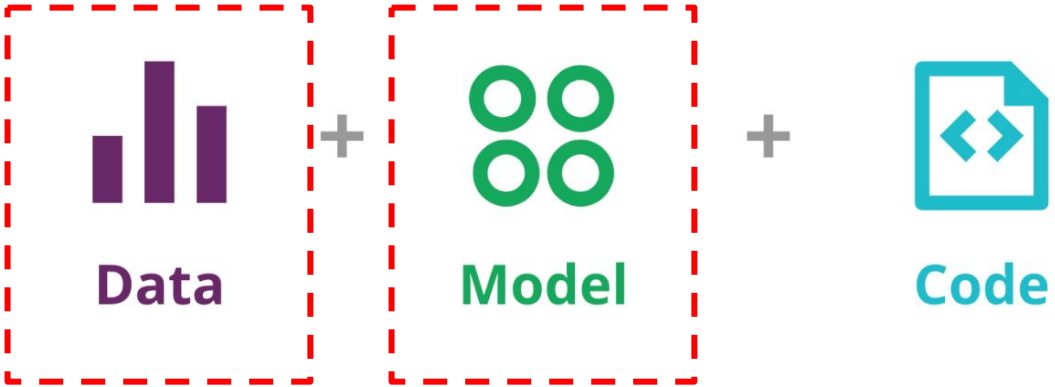
What to Test?

- *Can you prioritise the codebase?*
- *What is “mission critical”?*
- *Does the test reduce uncertainty about your system?*


Testing is Complex.
Now you know

Testing ML Systems

Key Adjustments



ML Systems

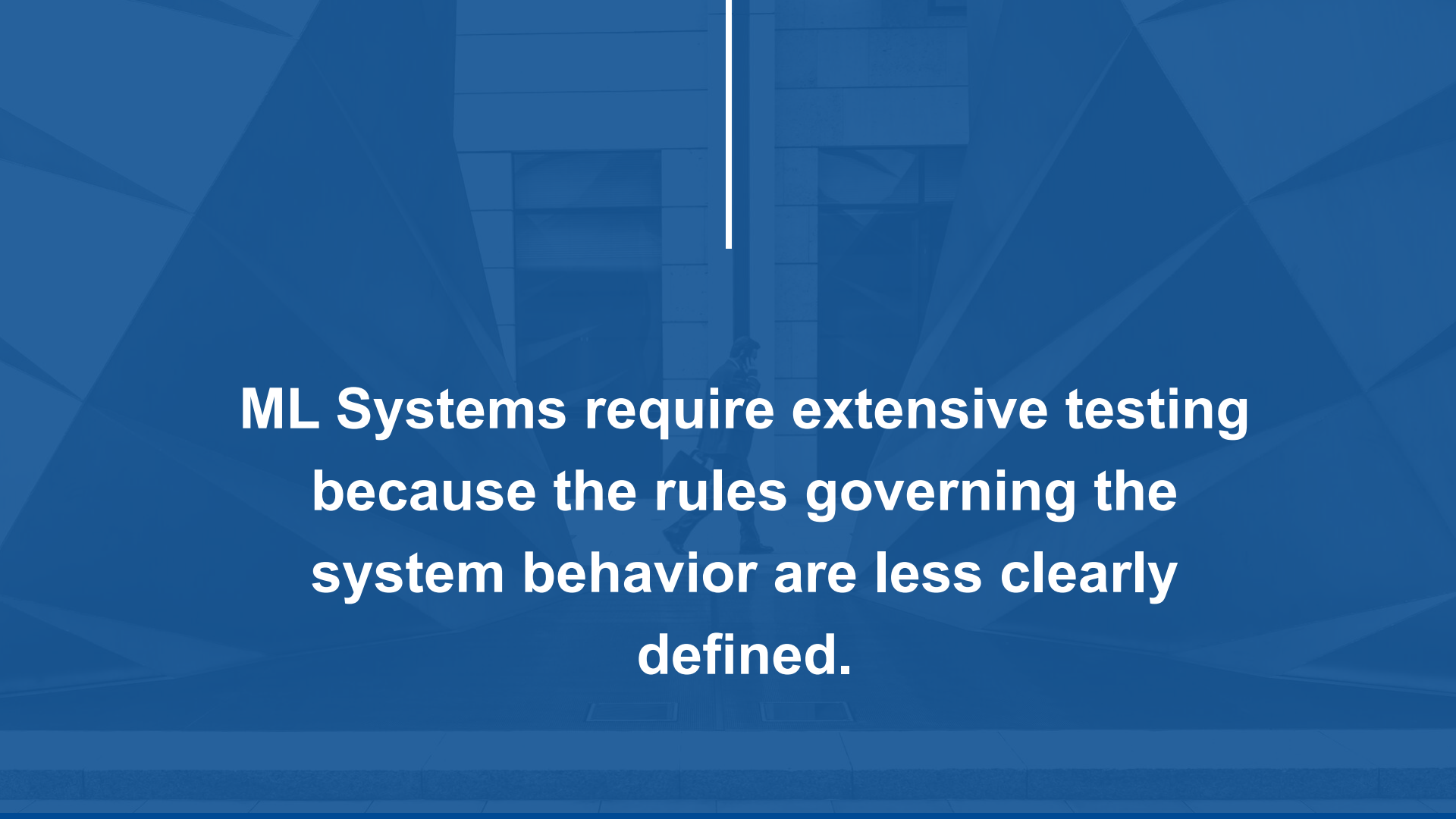
Two additional “axes” to consider for testing the behavior of the system

ML Systems

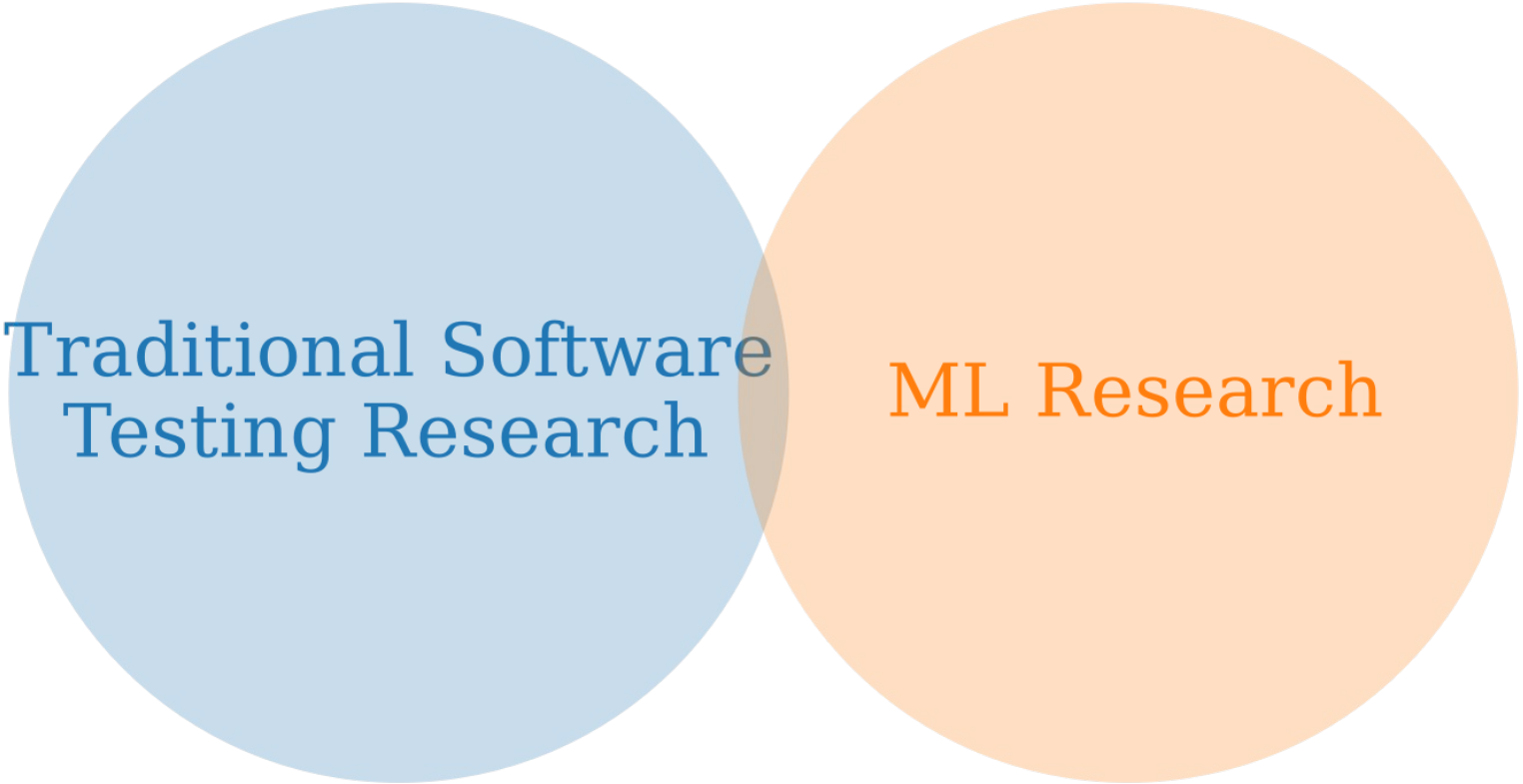
Constructed Inductively

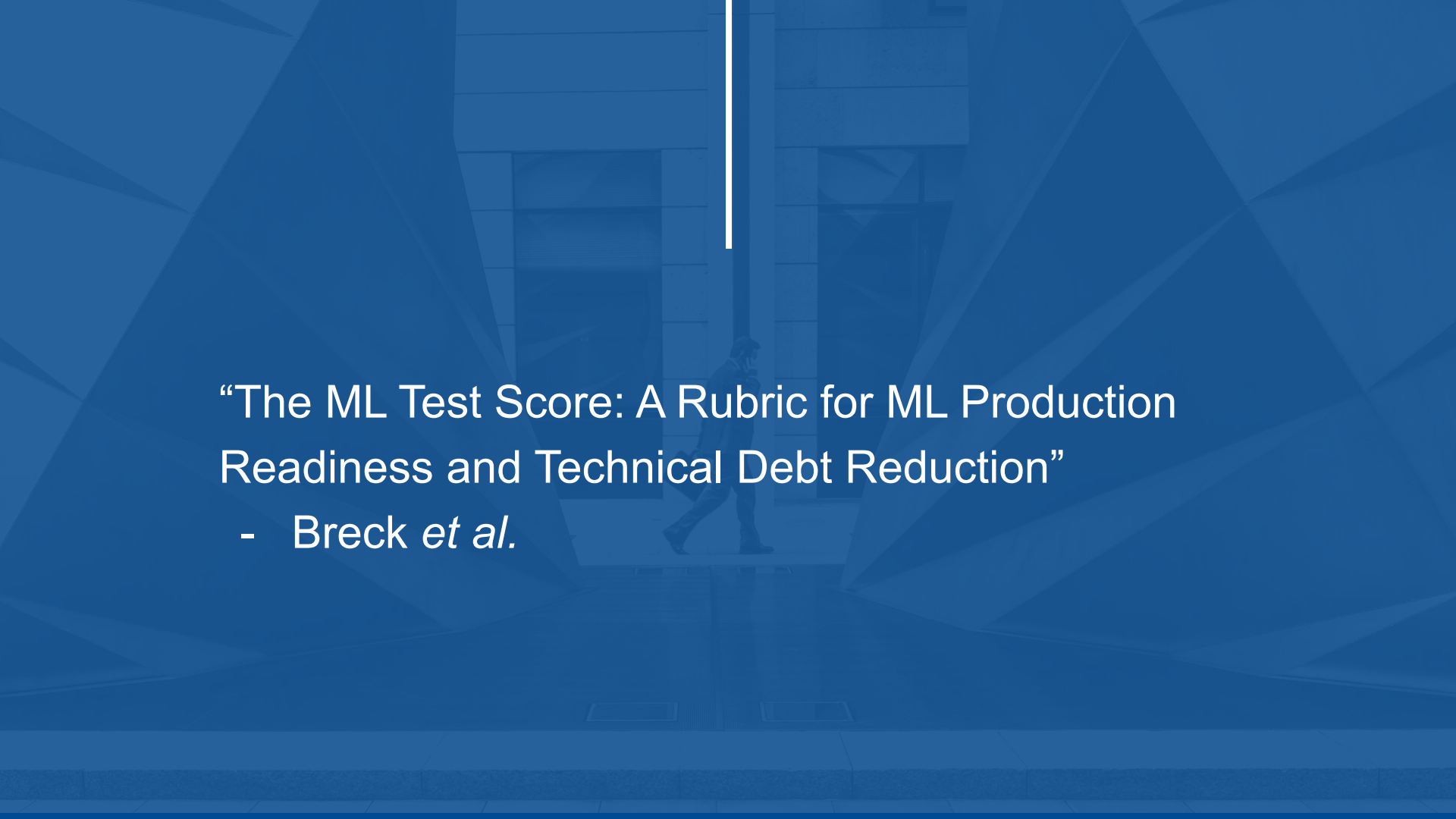
Traditional Systems

Constructed Deductively



ML Systems require extensive testing because the rules governing the system behavior are less clearly defined.


Traditional Software Testing Research
ML Research



“The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction”

- Breck *et al.*

Key Testing Principles for ML: Pre-Deployment

- **Use a Schema for Features**

  Schema should display feature expectations

- **Model Specification Tests**

  Model config (i.e. hyperparameters) needs unit testing

- **Validate Model Quality**

  Before attempting to serve the model

- **Test Input Feature Code**

  Bugs in features may be almost impossible to detect further in the pipeline

- **Training is Reproducible**

  Foundational step

- **Integration Test the Pipeline**

  A full test(s) that exercises the entire pipeline

Summary

Types of ML System Testing and their Timing				
Unit Testing	**Integration Testing**	**System Testing**	**Production Testing**	**Monitoring**
Inputs Schema	Reproducible Training	Reproducible Predictions	Shadow Deployments	Monitor System Serving Performance (latency etc.)
Model Specification	Pipeline Testing	Model-Infra Compatibility	Canary Releasing	Monitor Model Quality (accuracy, skew, staleness)
Input Feature Code		Validate Model Quality (differential tests)		
Model Quality (algorithmic)		Validate Computational Performance (benchmark tests)		
Model Quality (benchmarking)		Validate System Performance (load tests)		

Section Summary

Material Covered

- Testing Scope In This Course
- Testing Theory
- ML System Testing Challenges
- Jupyter Installation and Setup
- Hands-on Notebooks

Unit Testing a Production Model:

Section Overview

Upcoming Topics...

- Setup
- Pytest Fundamentals
- Code Base Overview
- 4 Types of Testing
- Alternating Theory/Hands-on

Python Code Conventions

Python Conventions 1: Style (PEP 8)

- The style guide for Python
- Makes working with your code much easier for other developers

Python Conventions 2: Type Hints (PEP 484)

```
1  import typing as t
2
3
4  def add_two_integers(first: int, second: t.Optional[int] = None) -> int:
5      """Sum two numbers."""
6
7      result = first
8      if second is not None:
9          result = first + second
10
11     return result
```

Python Conventions 3: Literal String Interpolation (PEP 498)

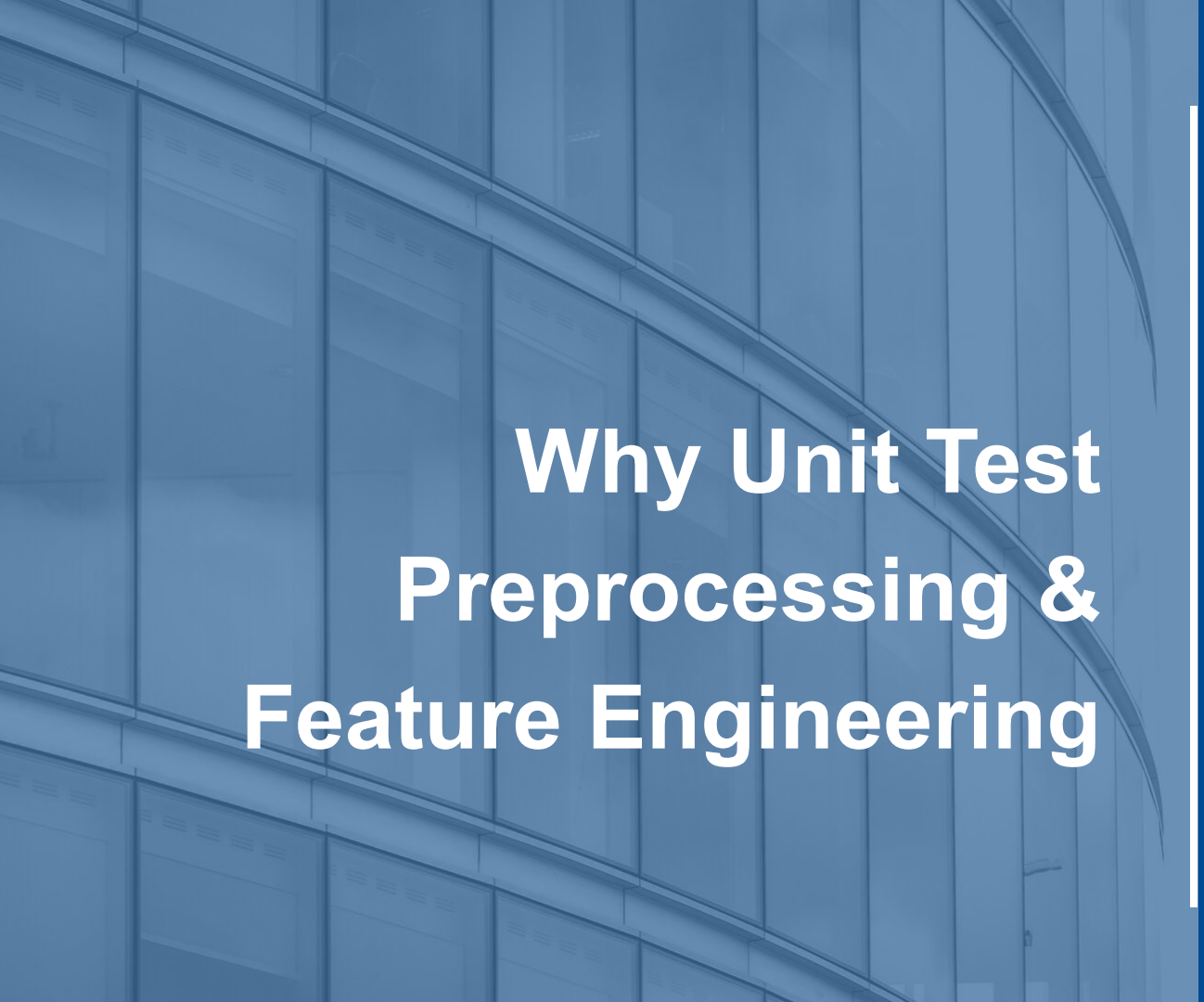

```
>>> import datetime
>>> name = 'Fred'
>>> age = 50
>>> anniversary = datetime.date(1991, 10, 12)
>>> f'My name is {name}, my age next year is {age+1}, my anniversary is {anniversary:%A, %B
%d, %Y}.'
'My name is Fred, my age next year is 51, my anniversary is Saturday, October 12, 1991.'
>>> f'He said his name is {name!r}.'
"He said his name is 'Fred'."
```

Python Conventions 4: Forcing Key Word Arguments (kwargs)

```
def my_function(*, foo):
    pass
```

- When we call this function, we have to name foo:

```
my_function(foo="bar")
```



Why Unit Test Preprocessing & Feature Engineering

“Data 7: All input feature code is tested: Feature creation code may appear simple enough to not need unit tests, but this code is crucial for correct behavior and so its continued quality is vital. Bugs in features may be almost impossible to detect once they have entered the data generation process, especially if they are represented in both training and test data.”

- Breck *et al.*, 2017, “The ML Test Score”

Test Engineered Data

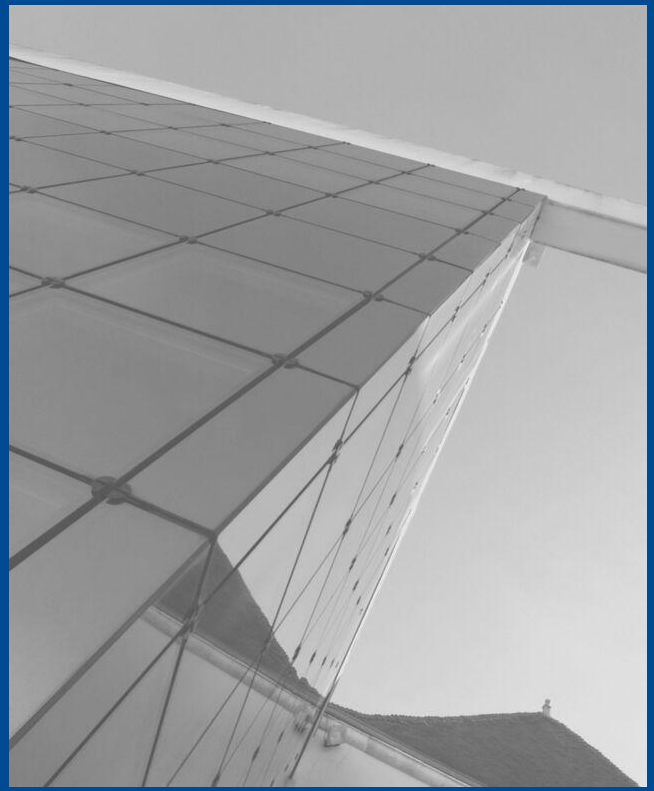
Examples

- All numeric features are scaled, for example, between 0 and 1
- Feature engineering steps involving calculations
- Missing data is replaced by mean or default values
- Data distributions after transformations conform to expectations
- Outliers are handled, such as by scaling or clipping

Why Unit Test Config

“Infra 2: Model specification code is unit tested. Although model specifications may seem like “configuration”, such files can have bugs and need to be tested.”

- Breck *et al.*, 2017, “The ML Test Score”



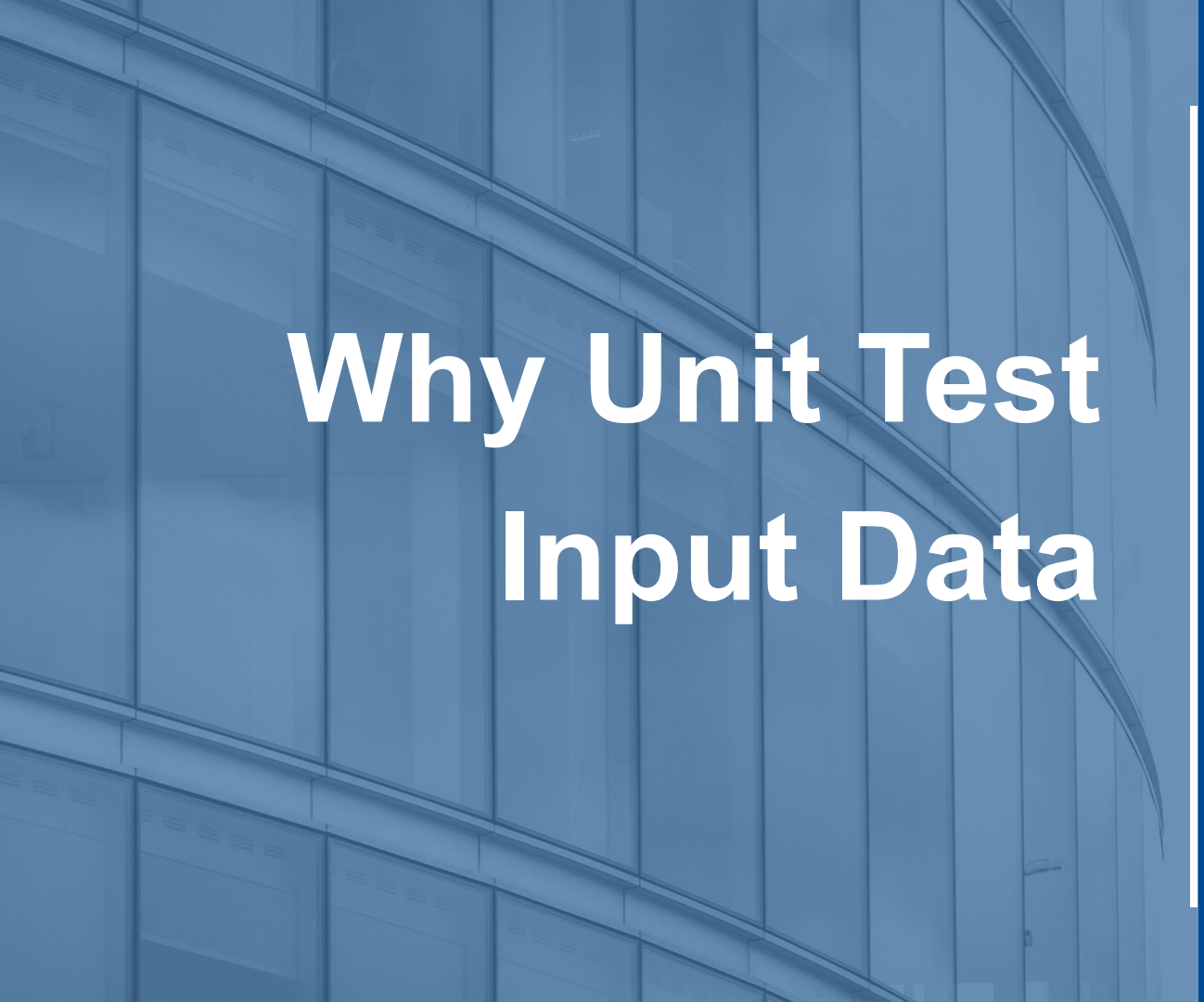
Understanding
Config Tests

Config Tests Matter

Especially in ML

Config Testing Complications

- Implicitly incorporates defaults that are built into the code (meaning that the tests are separately versioned as a result)
- Passes through a preprocessor such as bash into command-line flags (rendering the tests subject to expansion rules)



Why Unit Test Input Data

"Data 1: Feature expectations are captured in a schema."

- Breck *et al.*, 2017, "The ML Test Score"

Features

Understand the range and distribution. For categorical features, consider possible values.

Rule Creation

Encode your understanding into schema rules.

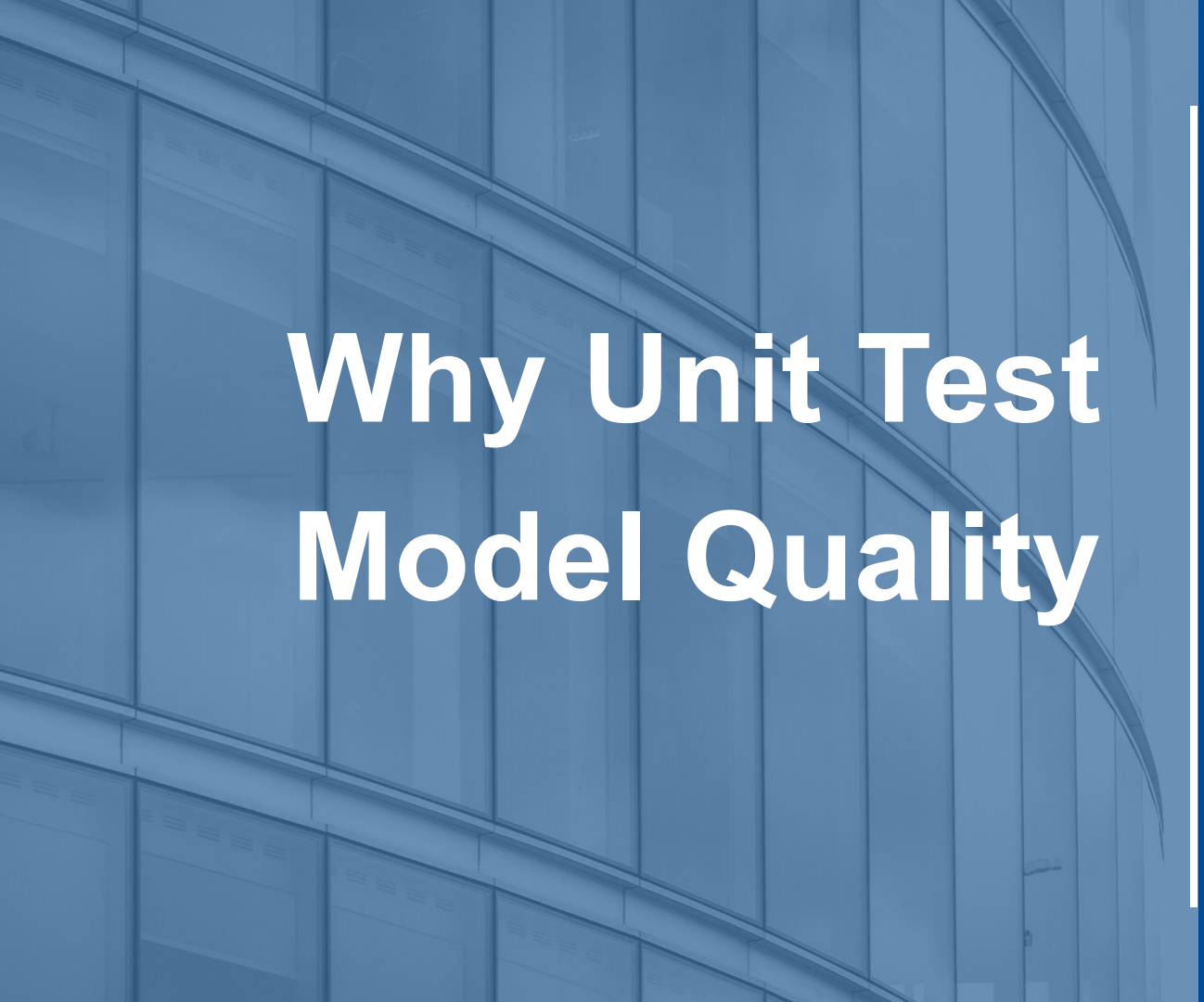
Testing

Test your data against the schema, consider when errors should be caught.

Validate Input Data Using a Schema

Implications

Of erroneous feedback loops



Why Unit Test Model Quality

“Infra 4: Model quality is validated before attempting to serve it: After a model is trained but before it actually affects real traffic, an automated system needs to inspect it and verify that its quality is sufficient; that system must either bless the model or veto it, terminating its entry to the production environment.”

- Breck *et al.*, 2017, “The ML Test Score”

Important Requirements

Keep the tests deterministic. If you need random numbers, be sure to set the random seed number so you can rerun the test.

Keep the tests short. Don’t have a unit test that trains to convergence and checks against a validation set.

Two Key Types

1. Sudden degradation

2. Gradual degradation

Model Quality Degradation

WRAP UP LECTURE

Unit Testing ML Systems

Key Concepts

Data Engineering Tests

Reduce risk of bugs in preprocessing/feature engineering code

Input Data Tests

Catch unexpected inputs, typically through a schema

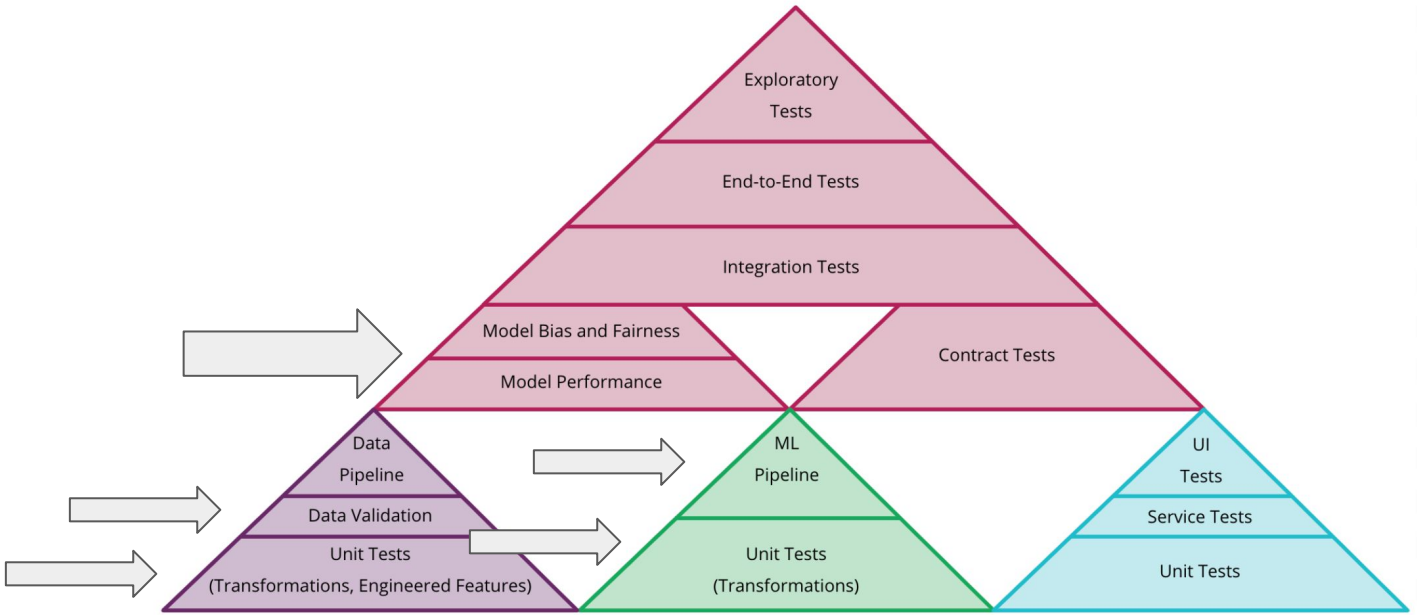
Config Tests

Reduce the risk of errors in our system configuration.

Model Quality Tests

Reduce the risk of sudden and gradual drops in model quality.

Situating Our Tests



Source: https://martinfowler.com/articles/cd4ml.html

Revisiting Containers

Section Overview

Docker

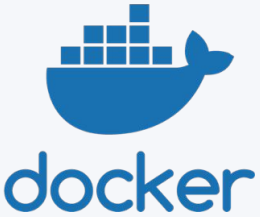
Docker Recap

Why Use Docker in ML Systems?

Docker Compose Intro

Installation and Setup

Hands on Exercise



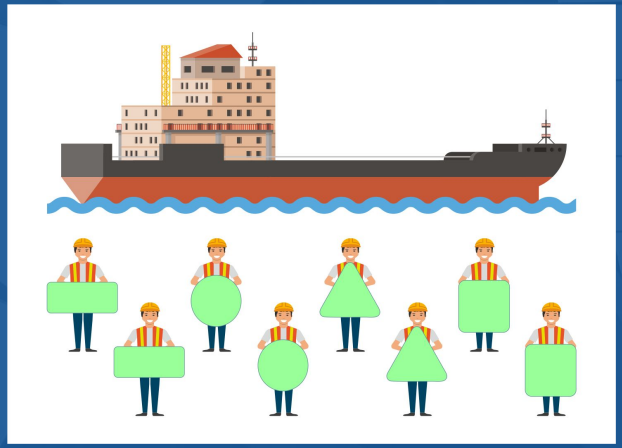
Docker

Brief Recap

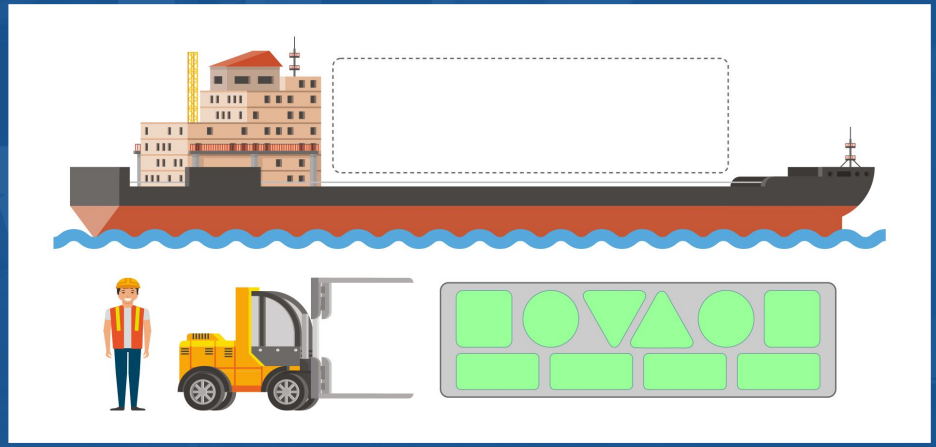
What is Docker

Docker is a software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code, and runtime.

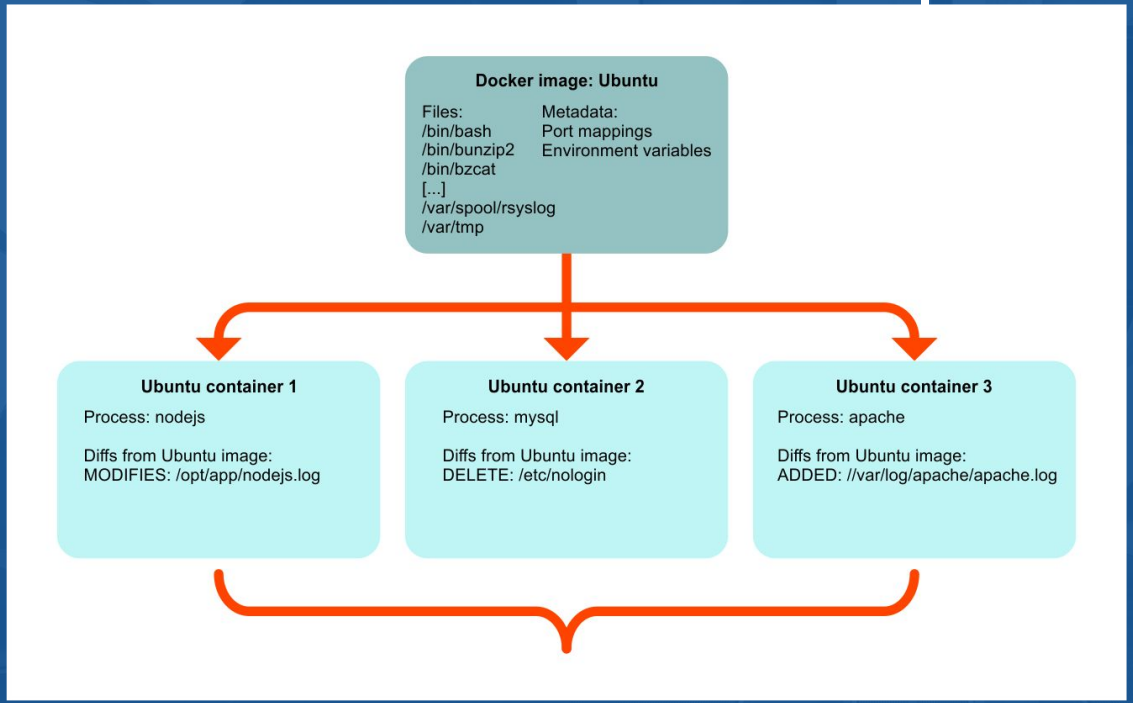
What is Docker



The shipping metaphor and the origin of the term “container”



What is Docker

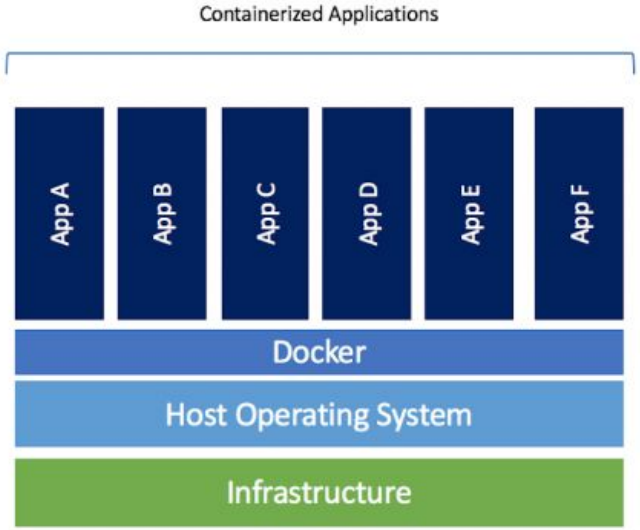


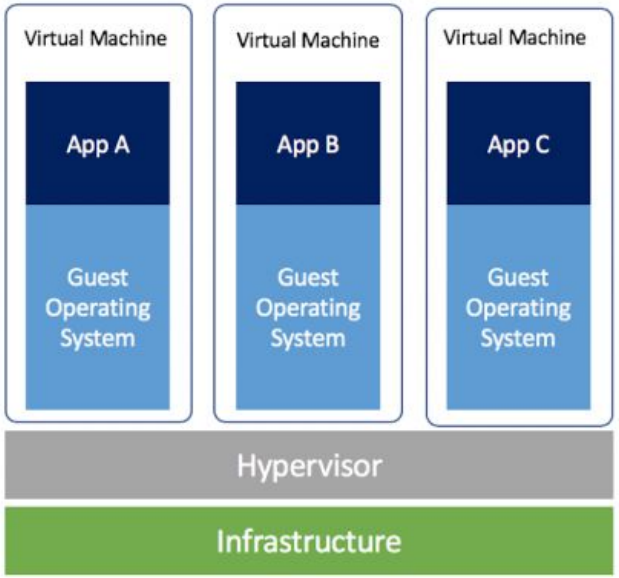
Images

- Contain files and metadata
- Are not running processes

Containers

- Are created from images
- Are running processes





Docker vs. Virtual Machines

Image source:
https://www.docker.com/blog/containers-replacing-virtual-machines/

What's the Difference?

```
FROM python:3.7-alpine
WORKDIR /code

# Set env vars required by Flask
ENV FLASK_APP app.py
ENV FLASK_RUN_HOST 0.0.0.0

# Install gcc so Python packages such as MarkupSafe
# and SQLAlchemy can compile speedups.
RUN apk add --no-cache gcc musl-dev linux-headers

# copy local requirements.txt into container
COPY requirements.txt requirements.txt

# install requirements inside the container
RUN pip install -r requirements.txt

# Copy the current directory . in the project
# to the workdir . in the image
COPY . .

# Set the default command for the container to flask run
CMD ["flask", "run"]
```

The Dockerfile

Key Commands

`docker build`

`docker run`

Service Provided by Docker

For sharing container images.

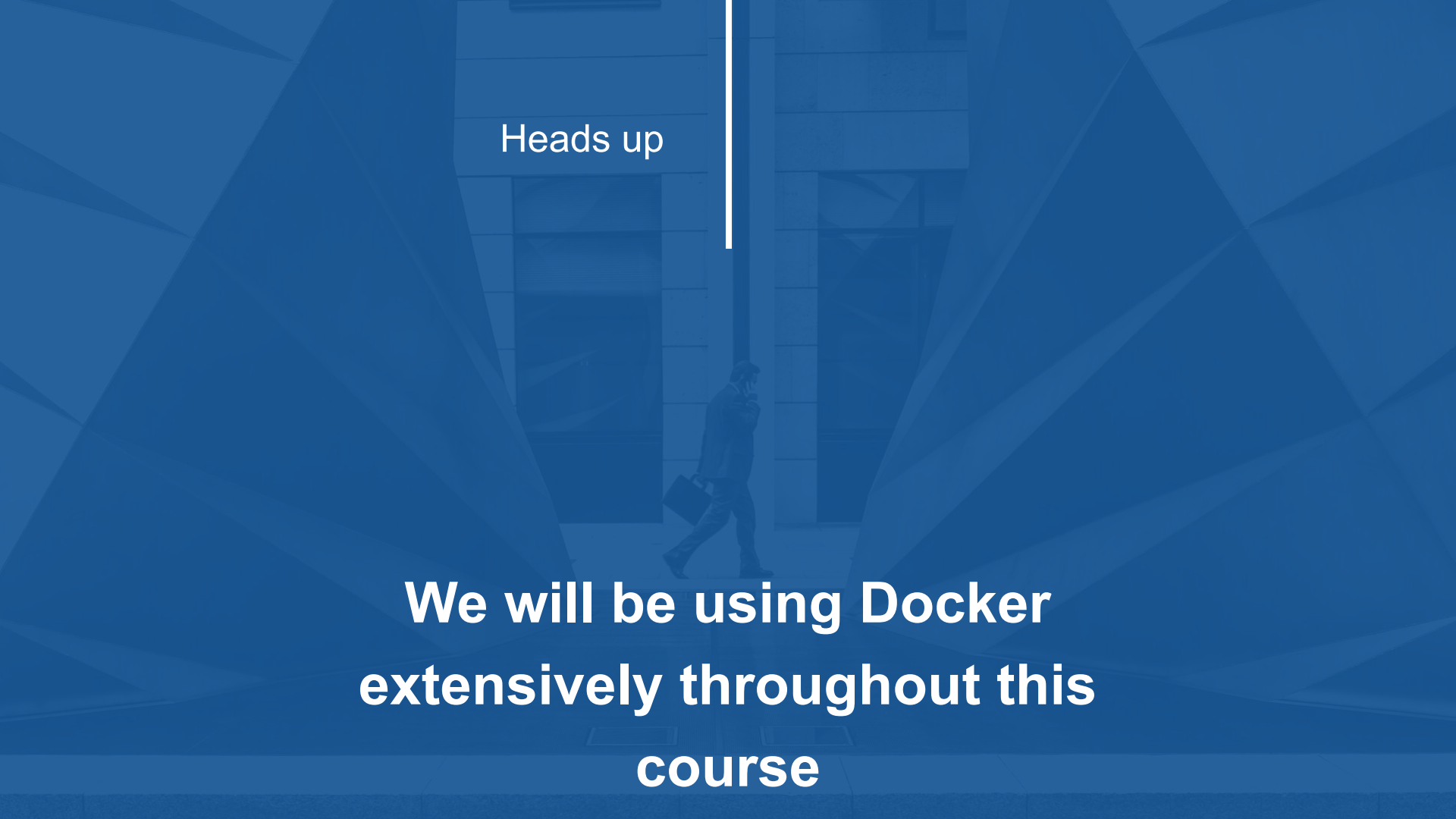
The Default Remote Repository

For Docker images.

We will make use of it in this course

Your first image is free

dockerhub



Heads up

We will be using Docker extensively throughout this course

Docker

Why should you care?

Why Use Docker

Software Systems

Standardization

CI Efficiency

Maintainability

Isolation

Security

Scalability

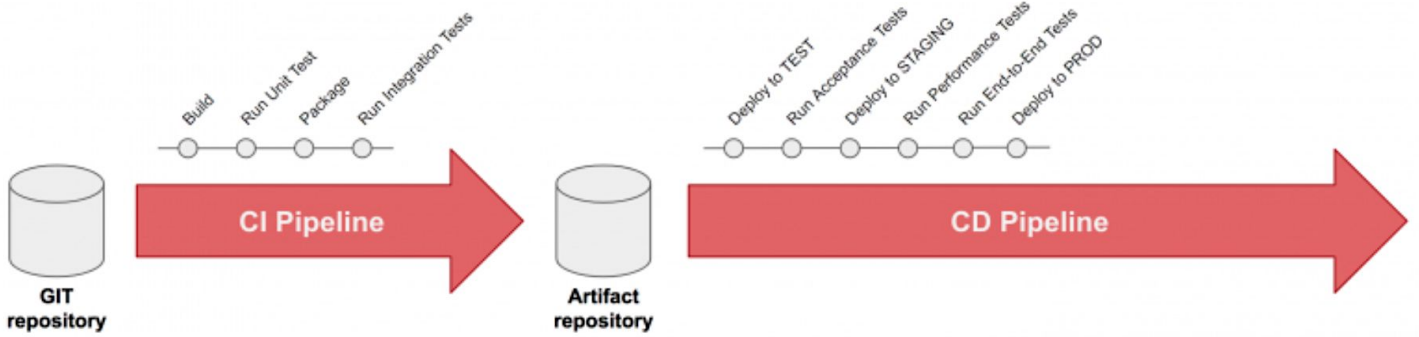
Business

Cost Reduction

Compatibility

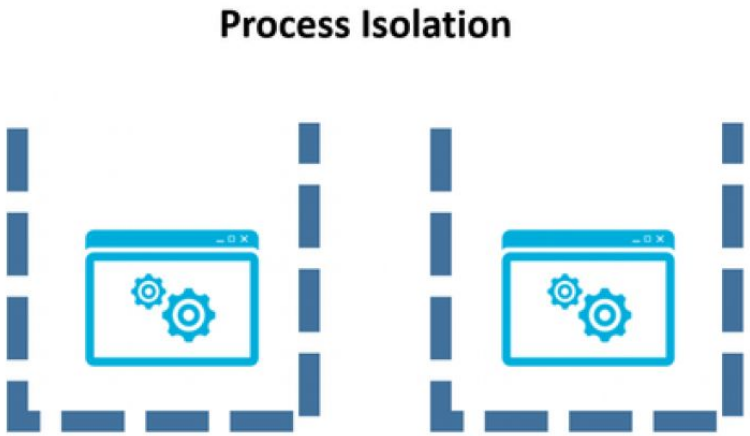
“Build once, run anywhere”

Standardization



CI Efficiency

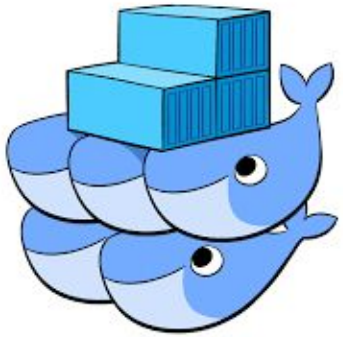
Maintainability



Isolation +

Security

Google Cloud Platform



Compatibility +

Scalability

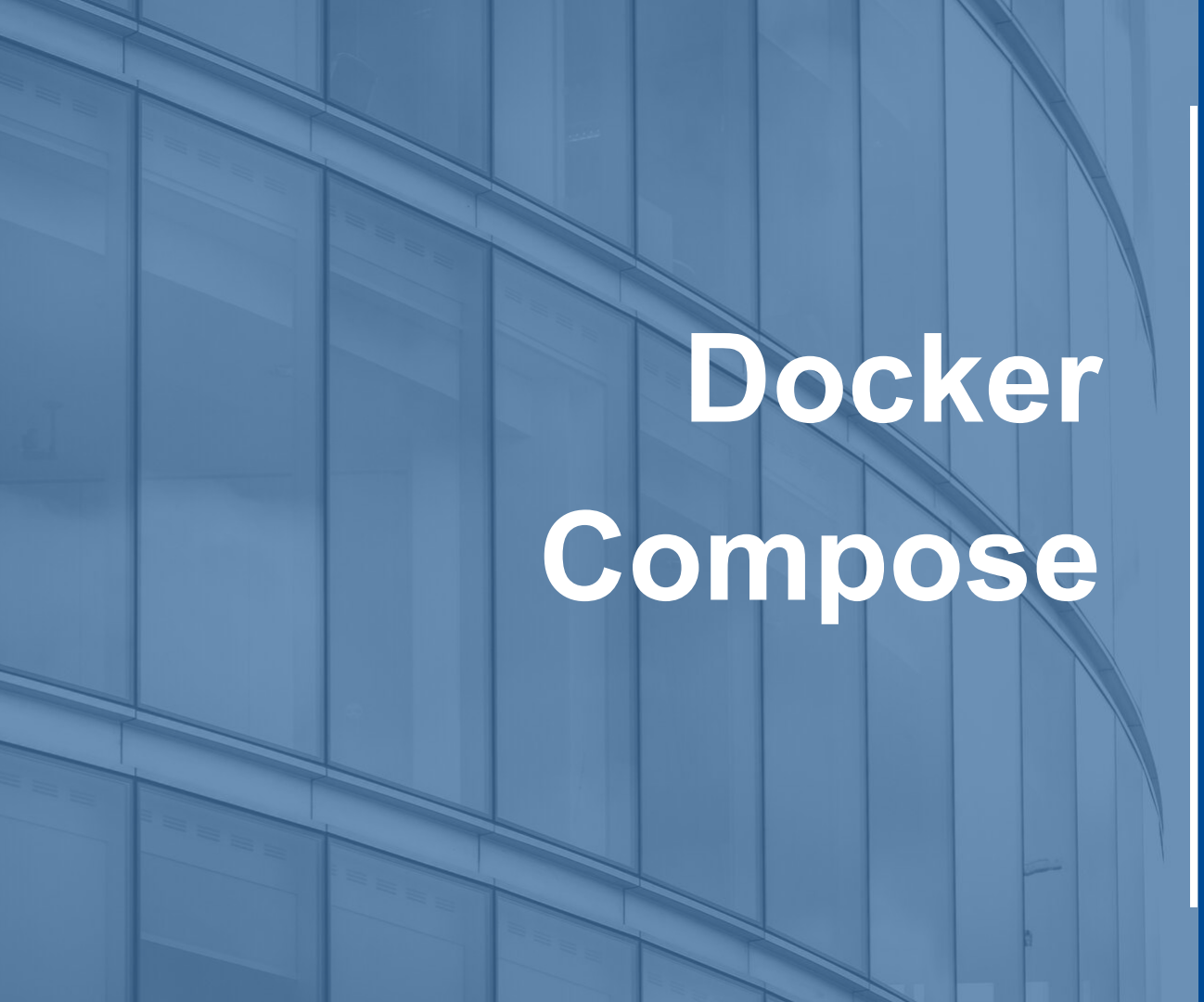
Business Cost Savings

Why choose Docker for ML Systems?

Reproducibility

Scalability

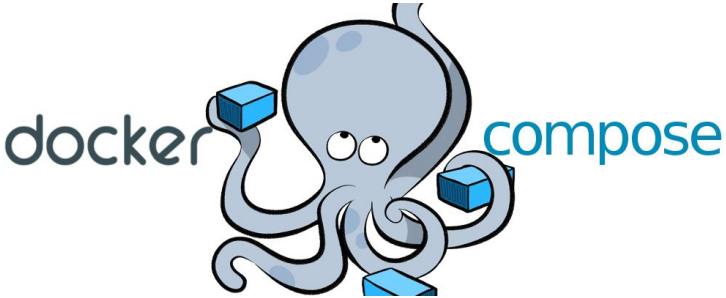
Compatibility



Docker Compose

An Introduction

What is Docker Compose?



Define and run multi-container Docker applications

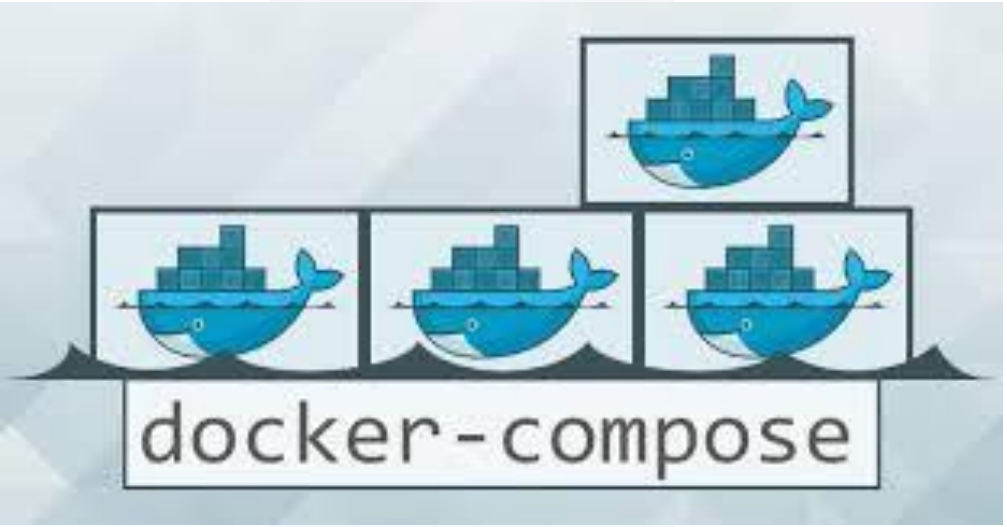
YAML file to configure your application’s services

Create and start all the services from your configuration

How Does Docker Compose Work

- Multiple isolated environments on a single host
- Volume data when containers are created
- Only recreate containers that have changed
- Variables and moving a composition between environments

Why Use Compose?



The docker-compose.yml file

Using Compose is basically a three-step process:

1. Define your app's environment with a `Dockerfile` so it can be reproduced anywhere.
2. Define the services that make up your app in `docker-compose.yml` so they can be run together in an isolated environment.
3. Run `docker-compose up` and Compose starts and runs your entire app.

```
version: '3'
services:
  web:
    build: .
    ports:
    - "5000:5000"
    volumes:
    - .:/code
    - logvolume01:/var/log
    links:
    - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

Example `docker-compose.yml` file

```
docker-compose build
docker-compose up
docker-compose down
```

Key Commands

Integration Testing a Production Model:

Section Overview

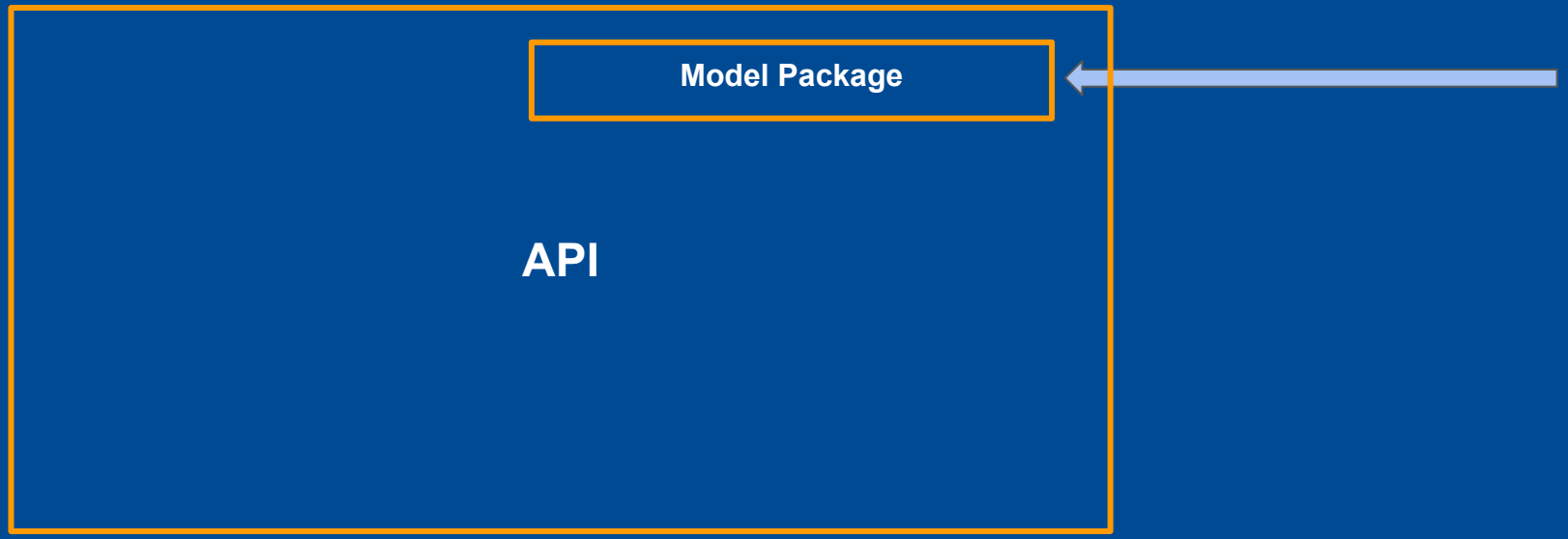
Upcoming Topics...

Code Base Overview (intro to Flask & Connexion)

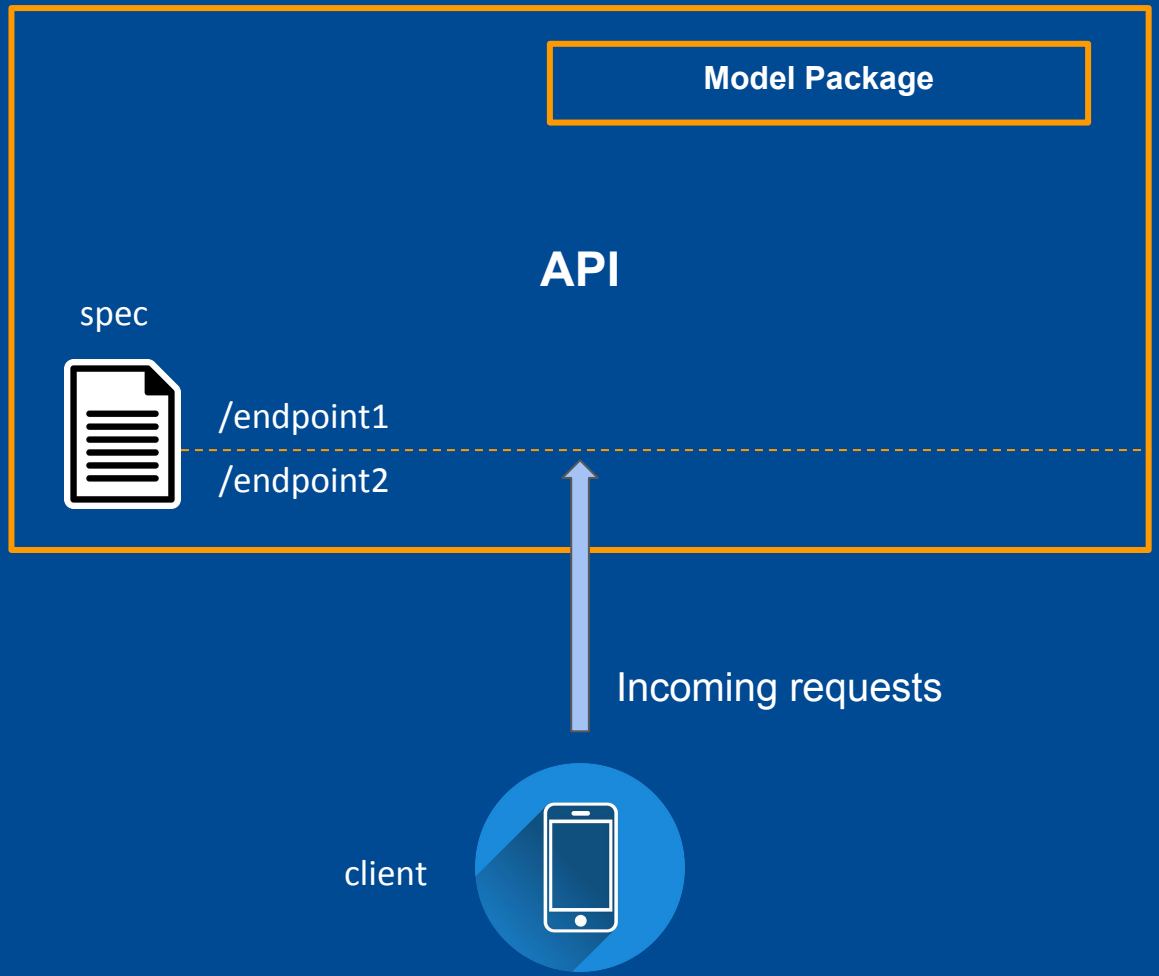
Running the Service via docker-compose

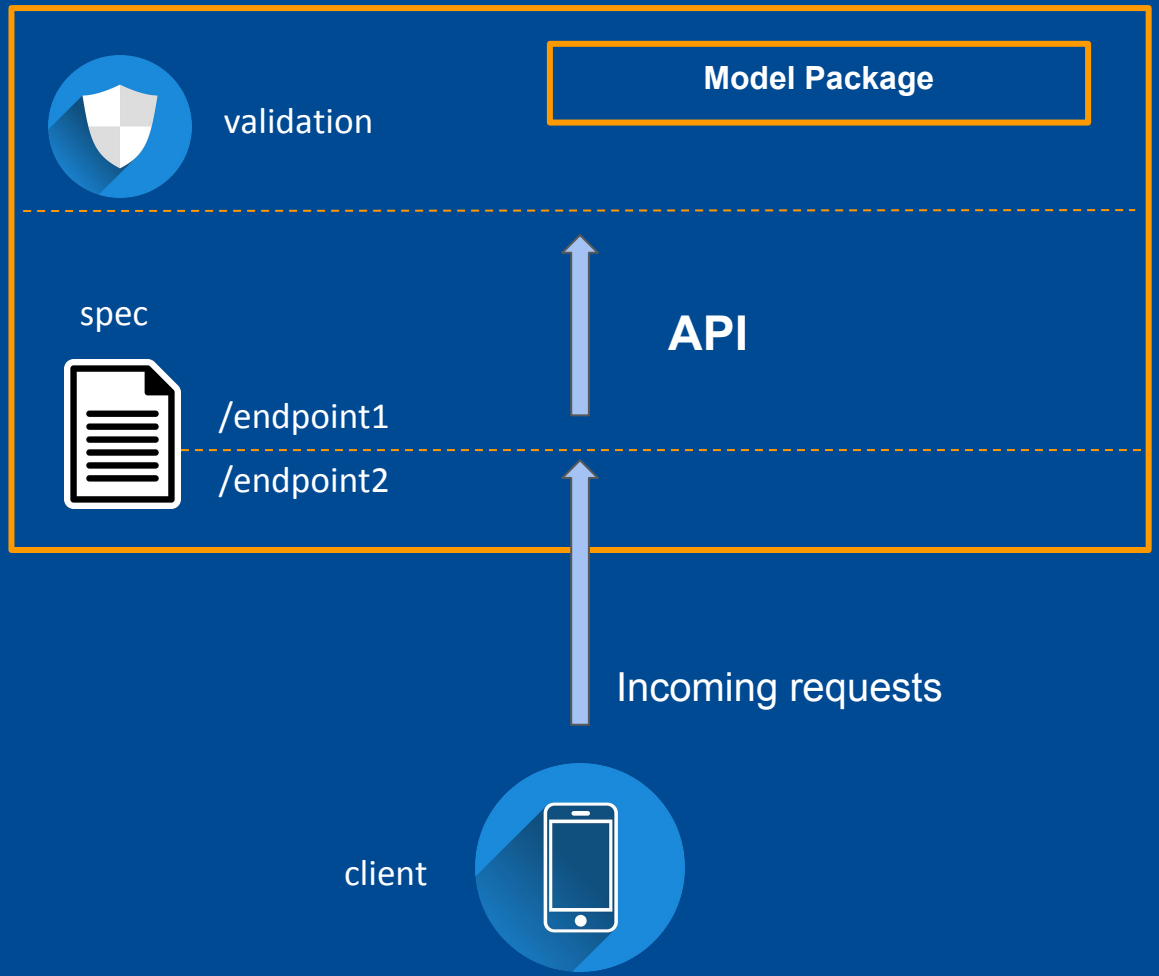
Alternating Theory/Hands-on

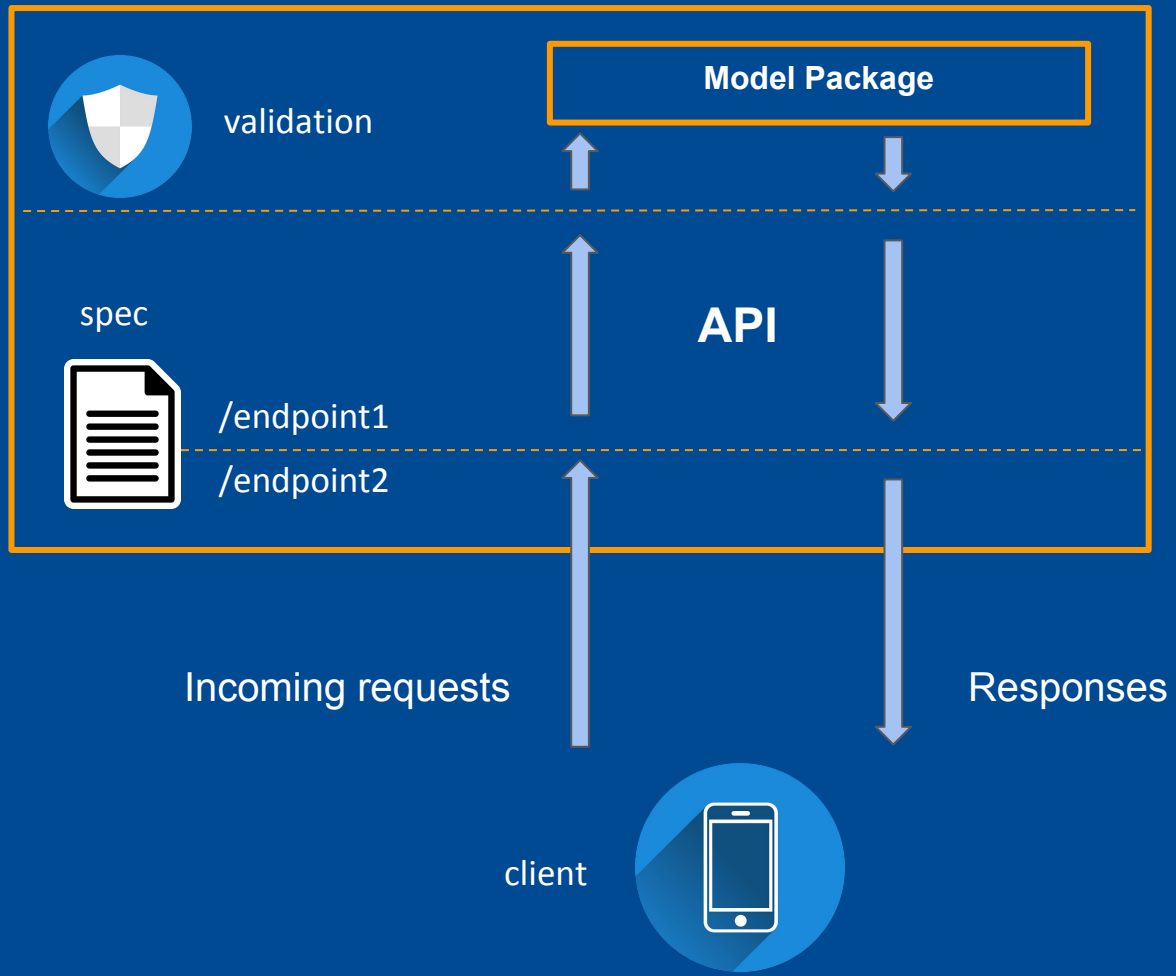
Conceptual Overview

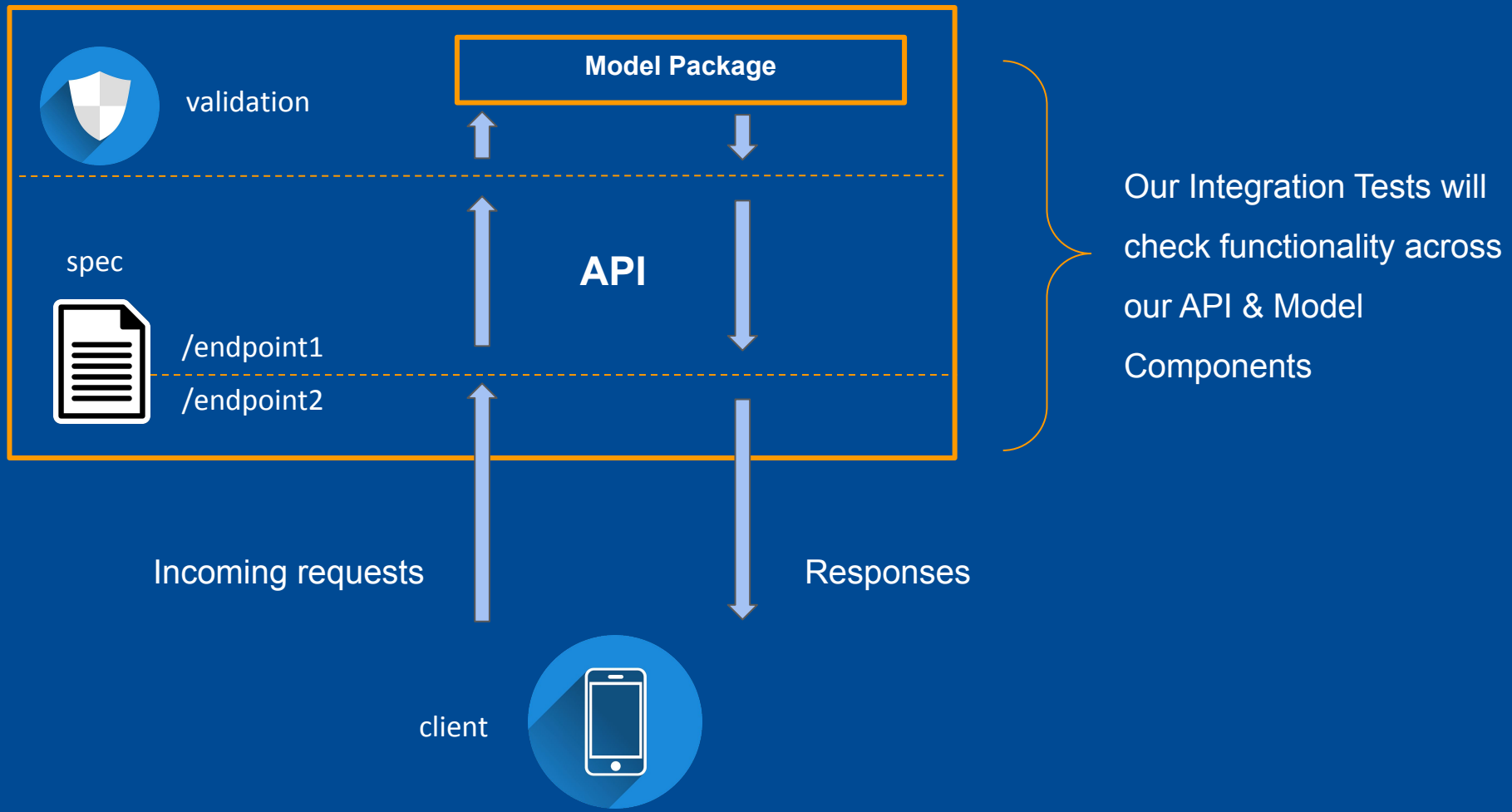


What new code are we introducing to our example project?


Model Package
API
spec
/endpoint1
/endpoint2
Incoming requests
client


Model Package
validation
spec
API
/endpoint1
/endpoint2
Incoming requests
client


validation
Model Package
spec
API
/endpoint1
/endpoint2
Incoming requests
Responses
client


validation
Model Package
spec
API
/endpoint1
/endpoint2
Incoming requests
Responses
client
Our Integration Tests will check functionality across our API & Model Components

Why Integration Test ML Systems

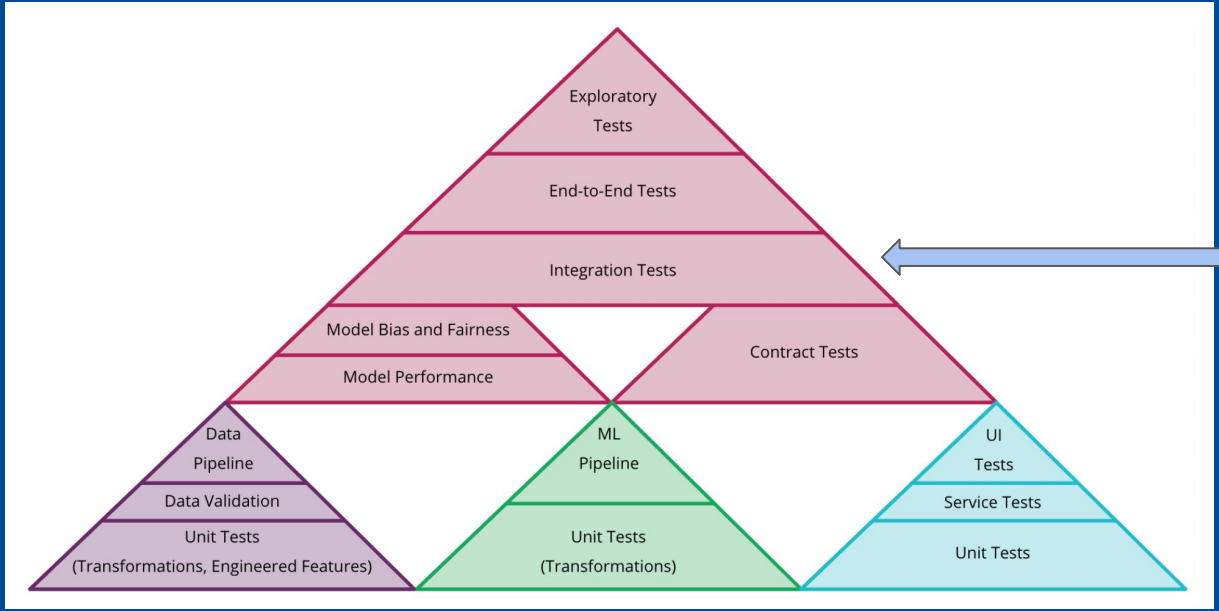
What is Integration Testing?

“The point of integration testing, as the name suggests, is to test whether many separately developed modules work together as expected.”

- Martin Fowler

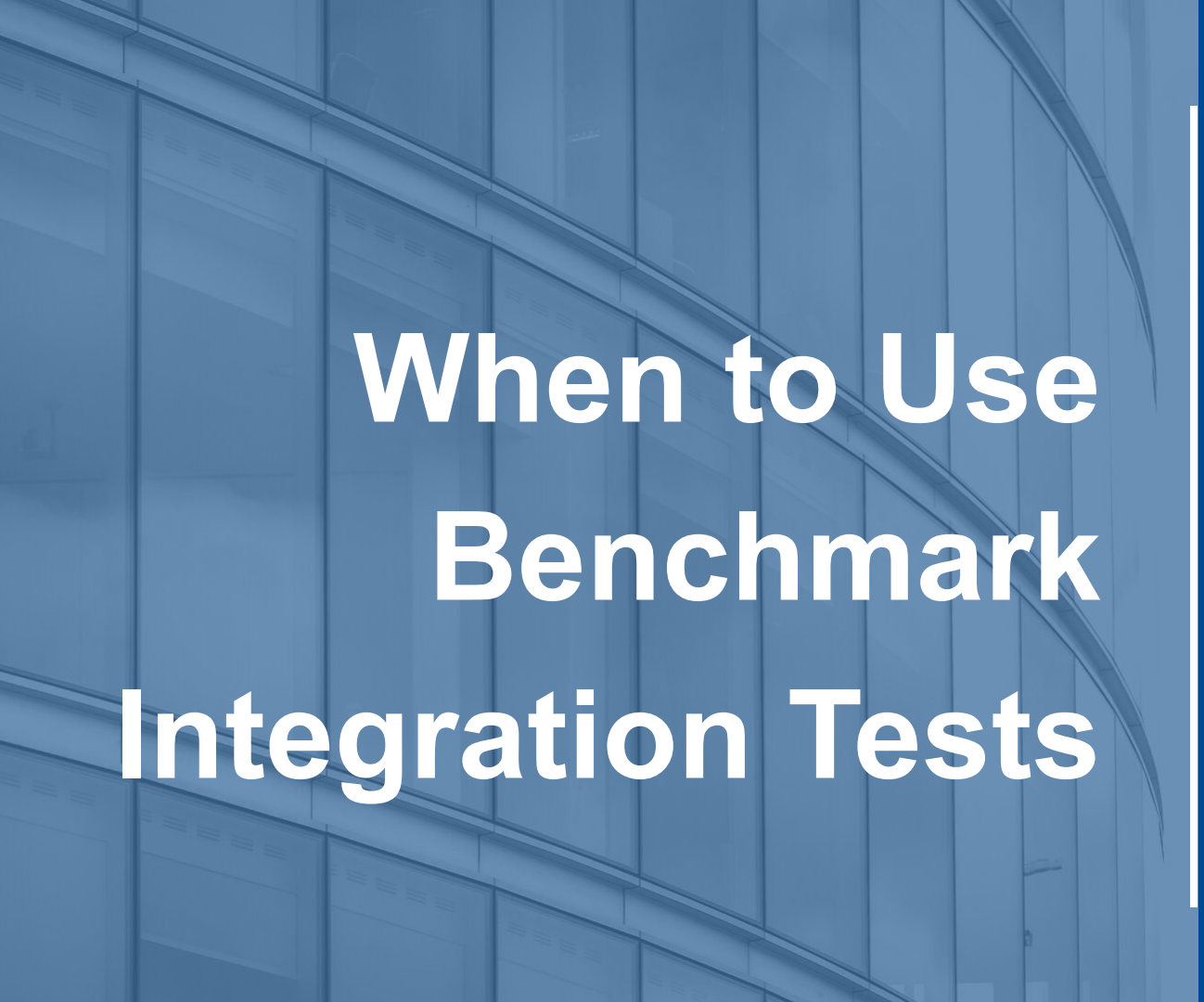
“Infra 3: The full ML pipeline is integration tested”

- Breck *et al.*, 2017, “The ML Test Score”



Source:
https://martinfowler.com/articles/cd4ml.html

Where do our Integration Tests Fit?



When to Use Benchmark Integration Tests

Be Aware

When models are distributed or exported to be used by a different application, you can also find issues where the engineered features are calculated differently between training and serving time.

Differential Testing a Production Model:

Section Overview

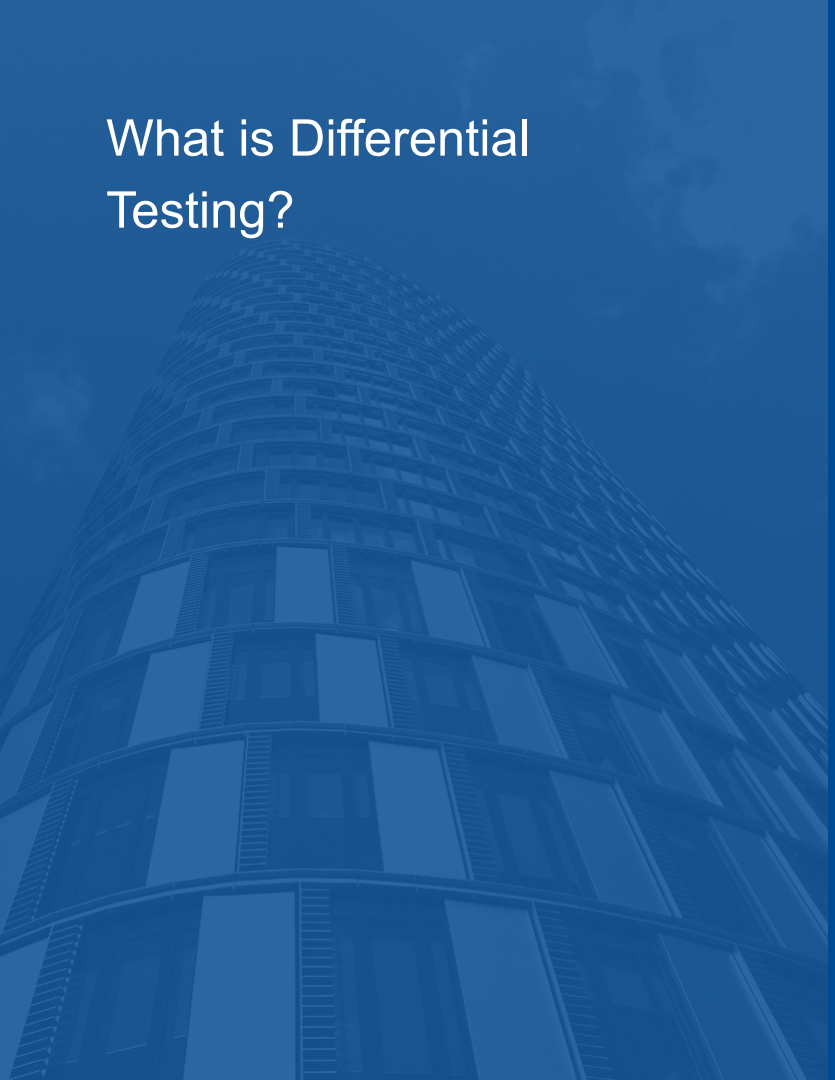
Upcoming Topics...

Differential Tests - What they are, and why they matter

Hands-on practice

Why Differential Test ML Systems

What is Differential Testing?



A type of test that compares the differences in execution from one system version to the next when the inputs are the same

Differential Tests

Also Known As...

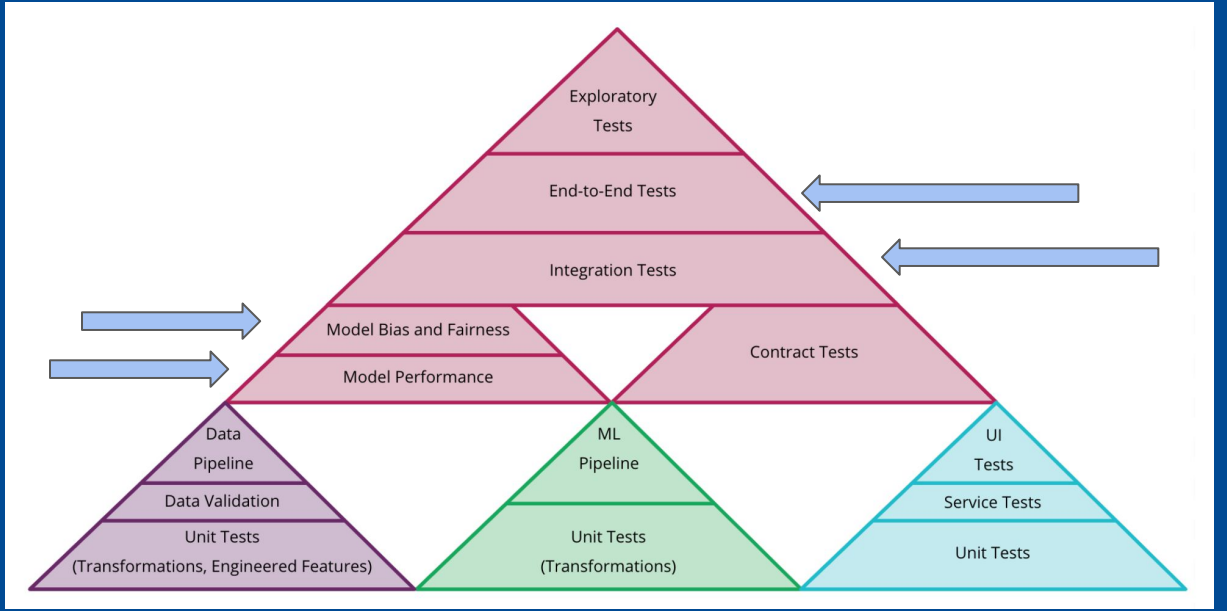
“Back to back” tests

Useful for...

Detecting ML errors that do not raise exceptions.

Tuning them...

Is a balancing act



Source:
https://martinfowler.com/articles/cd4ml.html

Where do our Differential Tests Fit?

Shadow Mode: Section Overview

Upcoming Topics...

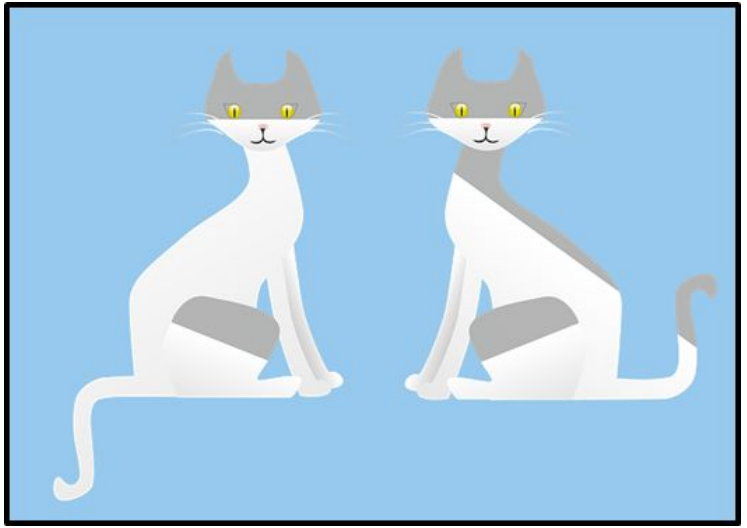
- Shadow Mode Theory from a Data Science Perspective (including specific statistical techniques)
- Shadow Mode from an Engineering Perspective (including implementation options)
- Add Shadow Mode Capabilities to Example Project
- Shadow Mode Assessment Simulation

Testing Models in Production

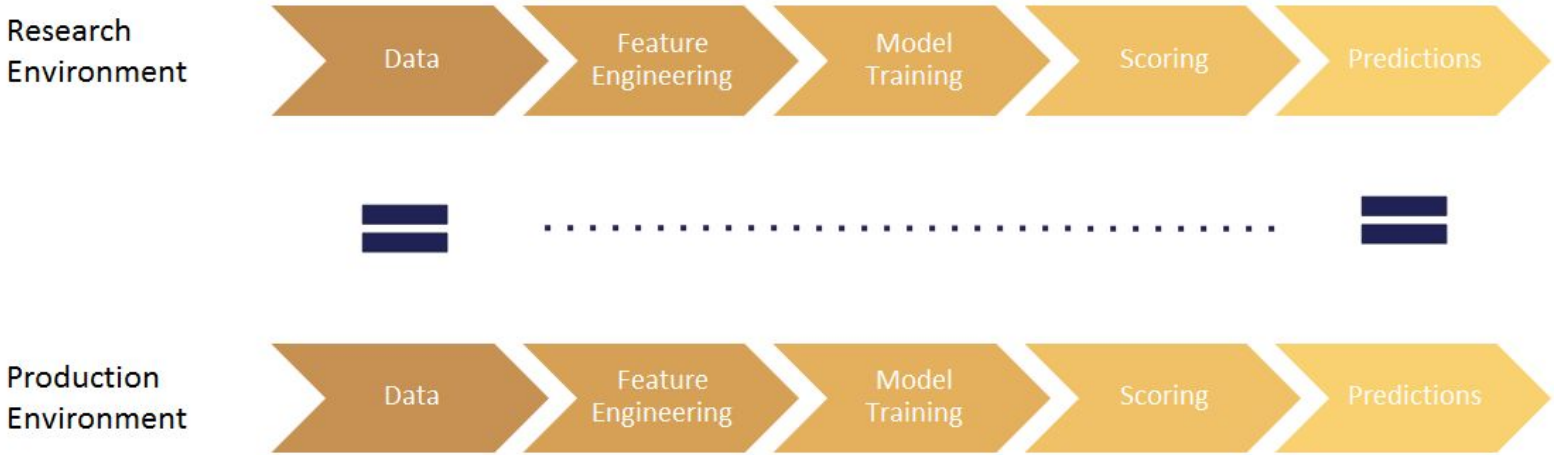
Making sure our live model mirrors our R&D model

Reproducibility in Machine Learning

Reproducibility is the ability to duplicate a machine learning model exactly, such that given the same raw data as input, both models return the same output.

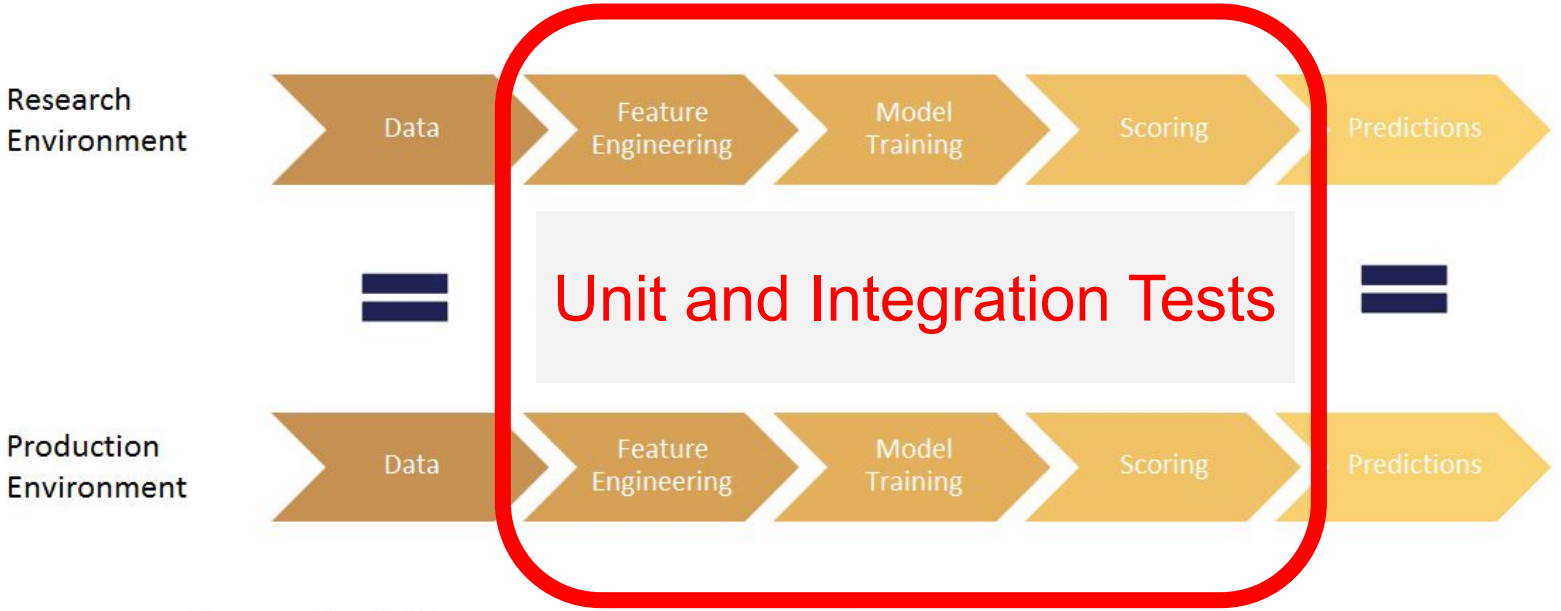


Reproducibility in Machine Learning



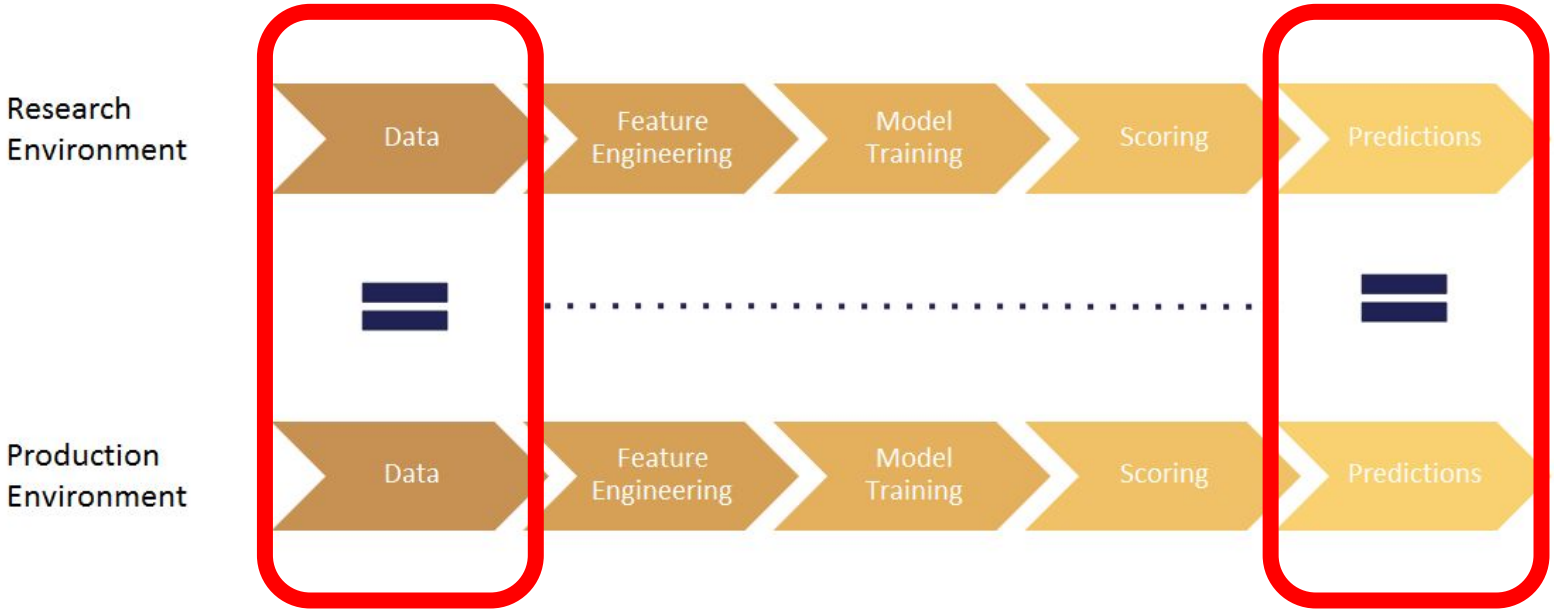
Reproducibility: same input comes in, same output comes out

Reproducibility in Machine Learning



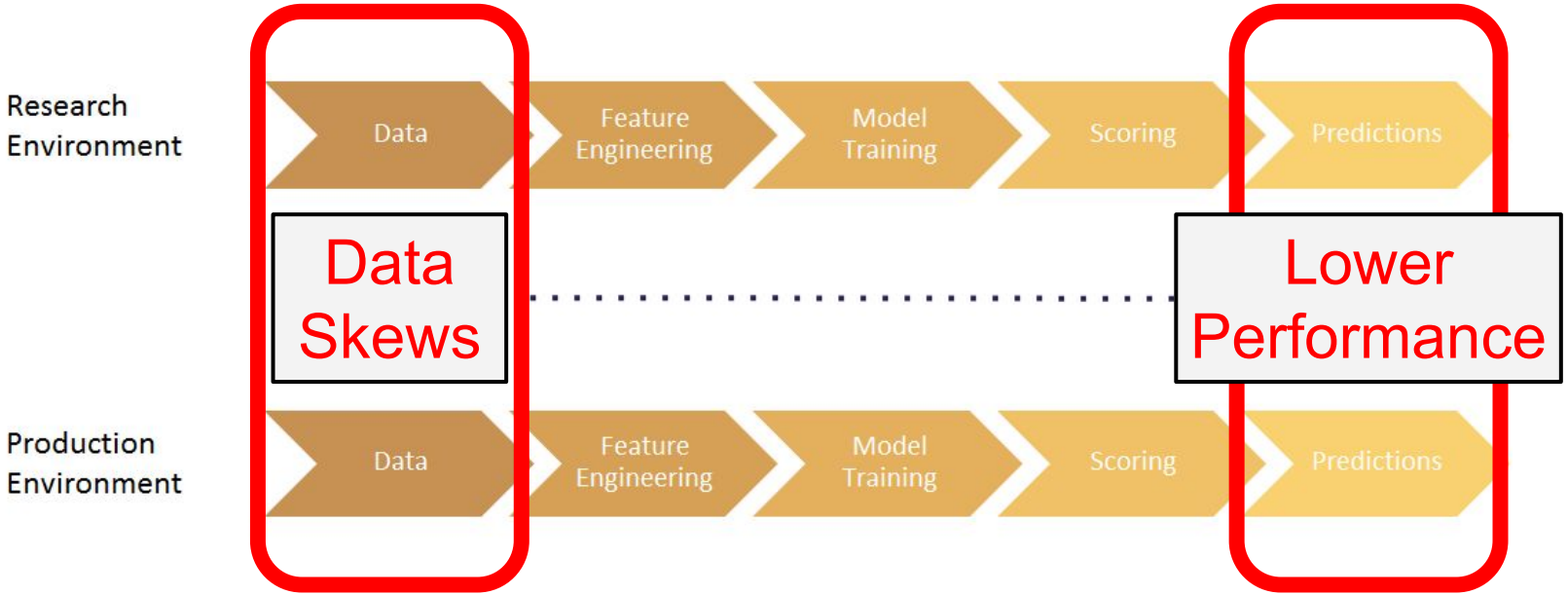
Reproducibility: same input comes in, same output comes out

Reproducibility in Machine Learning



Reproducibility: same input comes in, same output comes out

Challenges and Common Issues



Reproducibility: same input comes in, same output comes out

Data Skews

Training data is not representative of live data

- Error while designing the training data set
- A feature does not exist in production
- Data is gathered from different sources
- Changes in the population
- Feature dependencies changed
- Features change

Tests in Shadow Deployments

Model Input and Performance tests

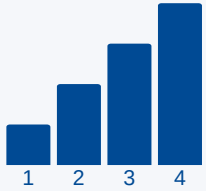
Data Skews

Training data is not representative of live data

- Error while designing the training data set
- A feature does not exist in production
- Data is gathered from different sources
- Changes in the population
- Feature dependencies changed
- Features change

Data Skews

Change in variable values



Change in distributions

Decrease in performance

What are the symptoms?

Values Checks - Allowed values

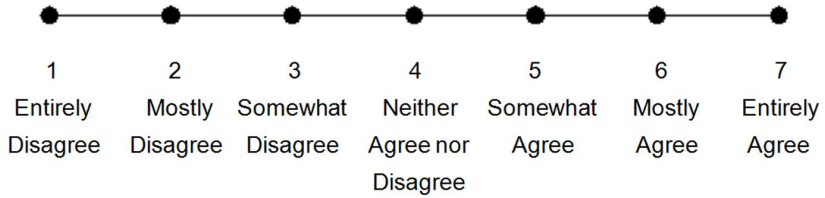
Feature takes allowed values (i.e., categorical)

- Gender: male of female
- Marital Status: single, married, divorced, widowed

Values Checks - Value range

Feature within a certain range (i.e., numerical)

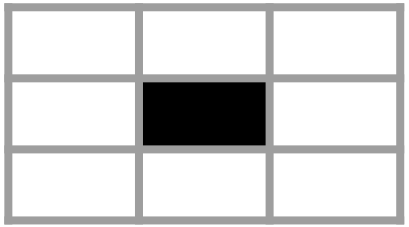
- Overall Quality: 1 - 10
- Pain Scale: 0 - 10
- Satisfaction Scale: 1 - 4



Values Checks - Missing values

Missing values

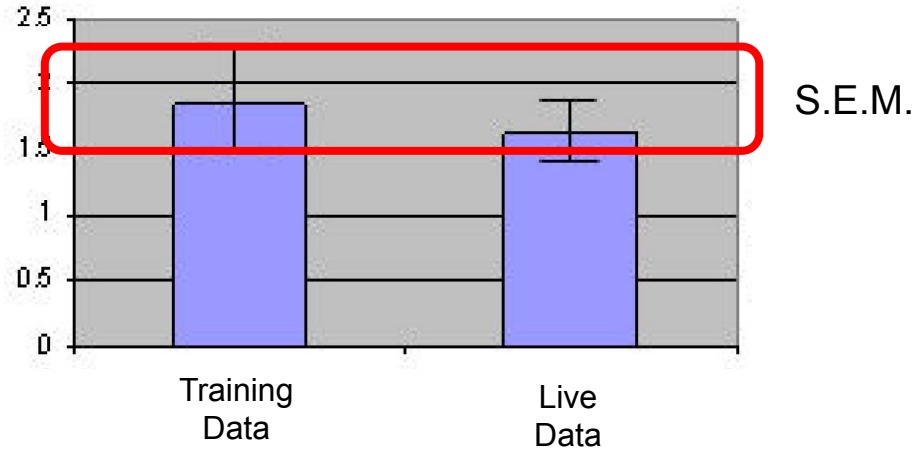
- Does the feature take missing values?
- Should the feature take missing values?
  - Age: can take missing values if respondent does not want to disclose
  - Name: can’t take missing values
  - Overall Quality: no missing values allowed



Distribution Checks - Basic statistics

Compare statistics between train and live data:

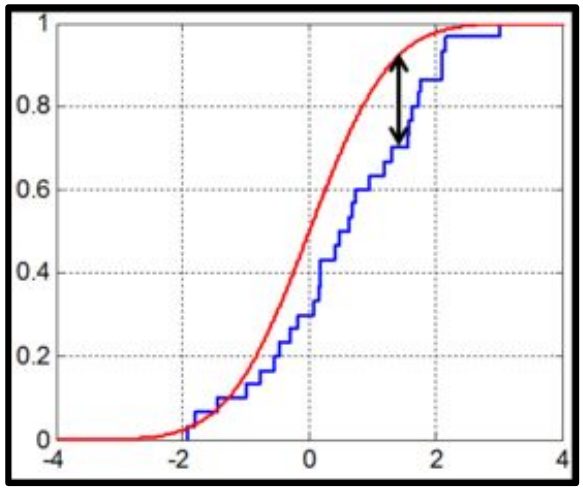
- Mean, median, standard deviation
- Maximum and minimum values



Distribution Checks - Statistical tests

Compare statistics between train and live data (numerical variables):

- Kolmogorov-Smirnov, Anova, Kruskal-Wallis, others



Distribution Checks - Statistical tests

Compare statistics between train and live data (categorical variables):

- Chi-square

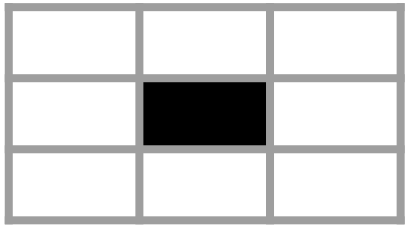
	Training Data	Live Data
Single	20	10
Married	60	32
Divorced	16	9
Widowed	4	2

Distribution Checks - Missing values

Missing values: % in training vs live data

- Chi-square

	Training Data	Live Data
Missing	20	9
Present	80	41



Distribution Checks - Sparse Variables

Sparse variables: Most values are zero

- Features derived from text, e.g.,Bag of Words

	dog	car	tree	fair	book	go	cook	nice	faint	sun	hello	yes
1	0	0	0	0	1	0	0	1	0	1	0	0
2	0	0	0	1	0	0	0	0	0	0	0	1
3	1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0	0	1	0

Non-Zero values: % in training vs live data

- Chi-square

	Training Data	Live Data
Non-Zero	10	2
Zero	990	98

Model Performance - Statistics

Testing the performance of live predictions

- Accuracy, Precision & recall, ROC-AUC
- MSE, RMSE, MAE, others

- Predictions distributions

Model Performance - Business metrics

Testing the performance of live predictions

- Revenue
- User experience, user satisfaction
- Conversion rates
- Time to settle, handling time

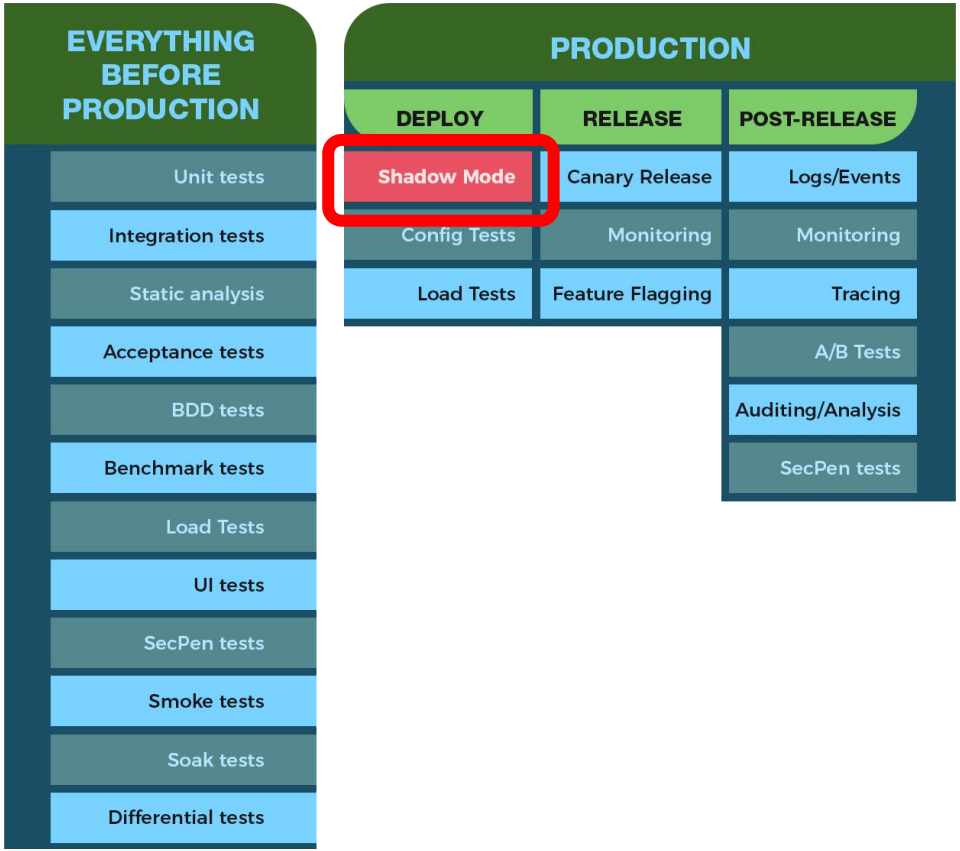
Model Performance - Bespoke populations

Testing the performance of live predictions

- Monitor data slices where performance is especially important or where model might perform poorly.
- Use understanding of the data to identify data slices of interest.
- Compare model metrics for data slices against the metrics for your entire data set.

Shadow Mode Theory

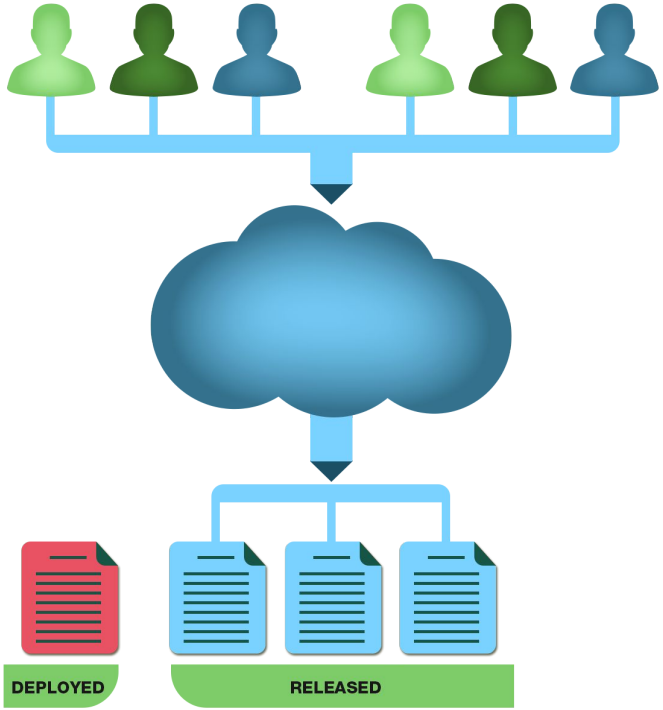
The Testing Spectrum




“Deployment need not expose customers to a new version of your service.”
– Art Gillespie

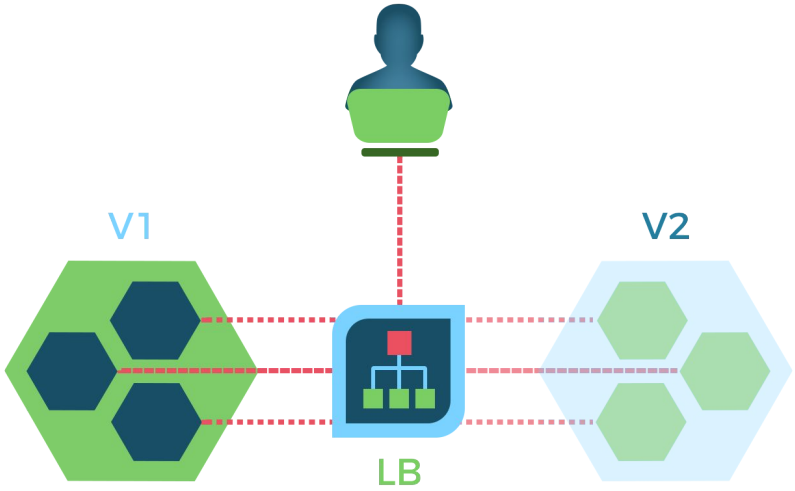
Shadow Deployments at the application level

- Incoming prediction requests are performed by the current model
- The same prediction request is *__also__* sent to the shadow model, where the prediction is **saved but not served** back to the customer
- The logic for dividing the predictions is done in code



Shadow Deployments at the Infrastructure Level

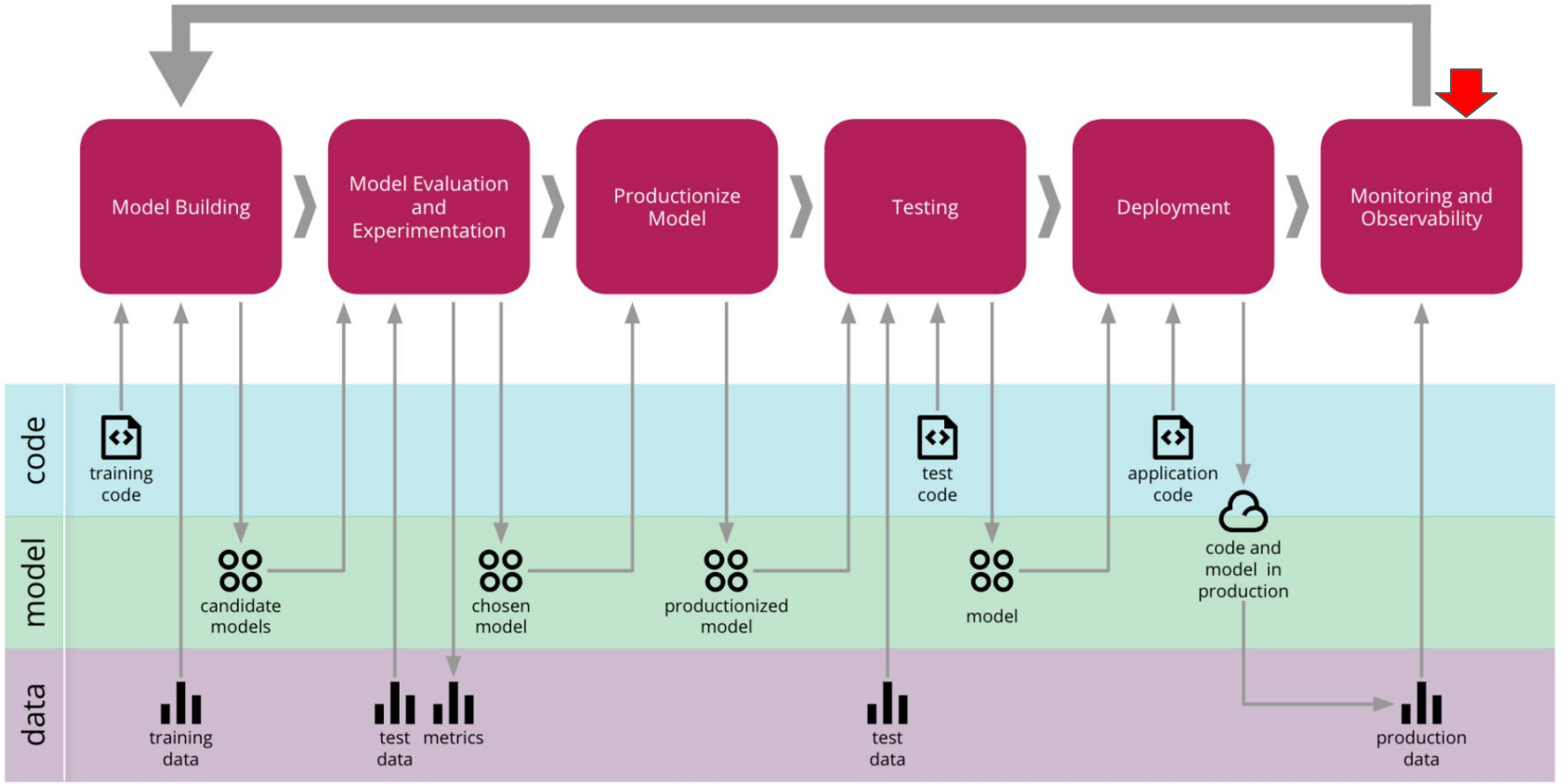
- Shadow prediction requests are forked at the infrastructure level (via a load balancer)
- Requires a mature DevOps team



Monitoring & Observability

Overview

ML System Lifecycle (CD4ML)



Source: https://martinfowler.com/articles/cd4ml.html

Monitoring Sections Overview (1)

Metrics

- Monitoring Theory [Important]
- Metrics for Machine Learning
- Introduction to Prometheus & Grafana
- Prometheus Hands-on lectures
- Applying Prometheus

Monitoring Sections Overview (2)

Logs

Logs for Machine Learning

Introduction to Kibana

Kibana Hands-on lectures

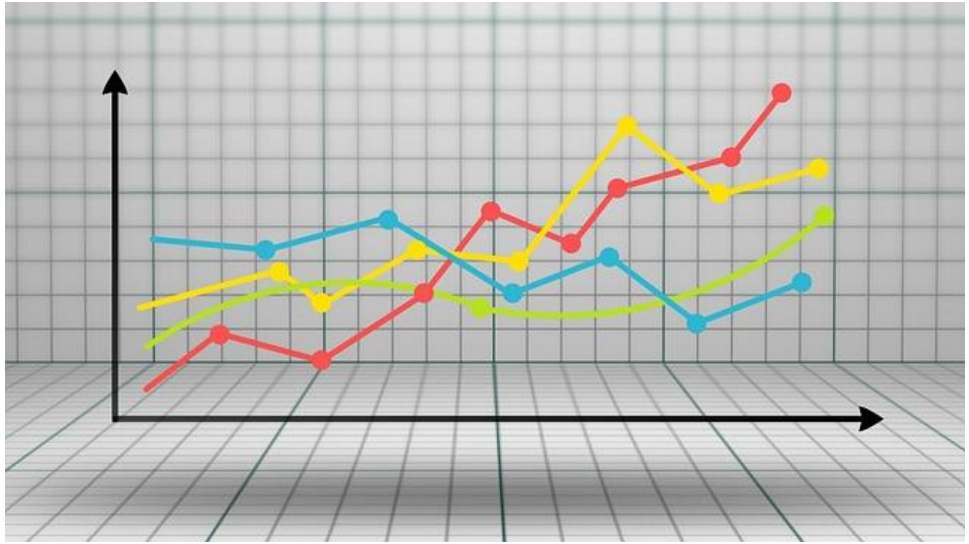
Applying Kibana

Monitoring is a huge topic.

Monitoring Model Performance throughout their Lifetime

Are our models still performing as expected?

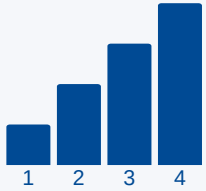
Data changes



- Changes in incoming population
- Feature definition change
- Features become unavailable

Continuous model monitoring after deployment

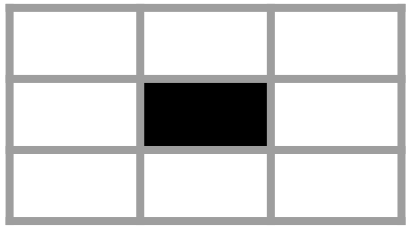
Change in variable values



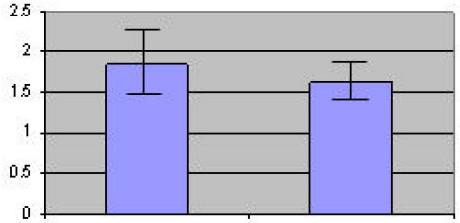
Change in distributions

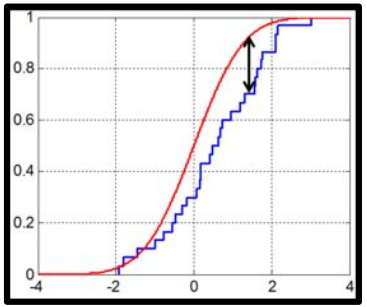
Decrease in performance

Model and input checks



Model input monitoring



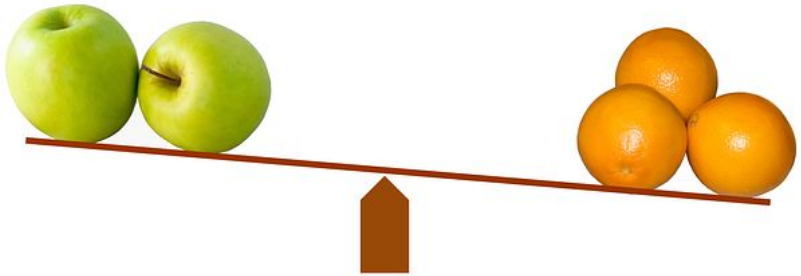


	Training Data	Live Data
Single	20	10
Married	60	32
Divorced	16	9
Widowed	4	2

Distribution monitoring

Model performance monitoring

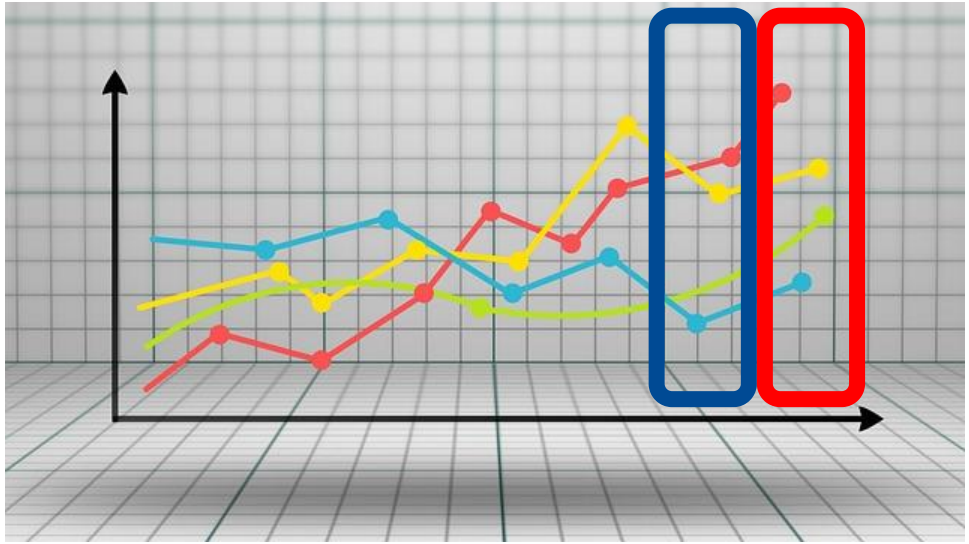
Data checks in shadow mode



Training data

Live data

Data changes



- Changes in incoming population
- Feature definition change
- Features become unavailable

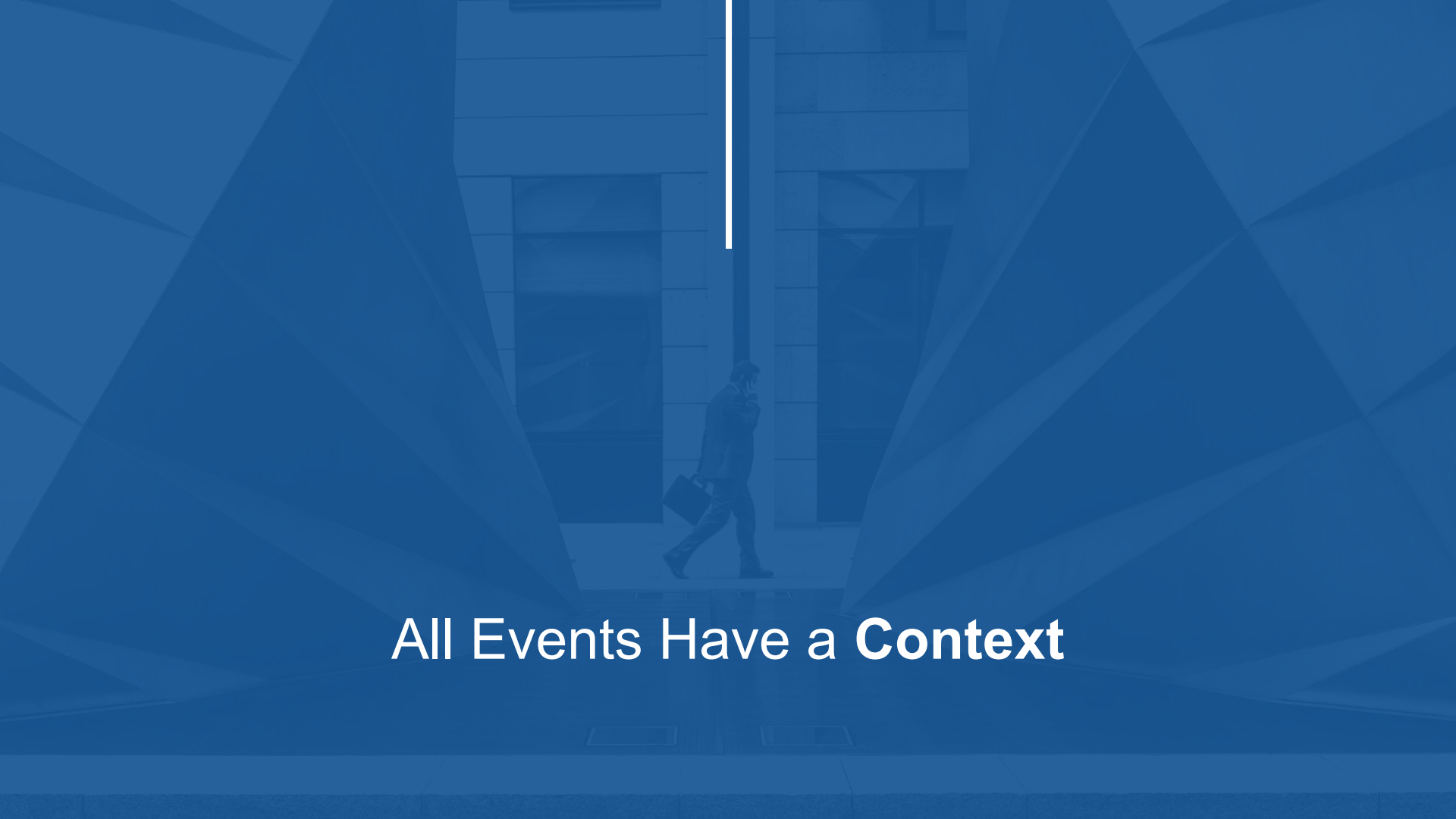
Monitoring & Observability

THEORY

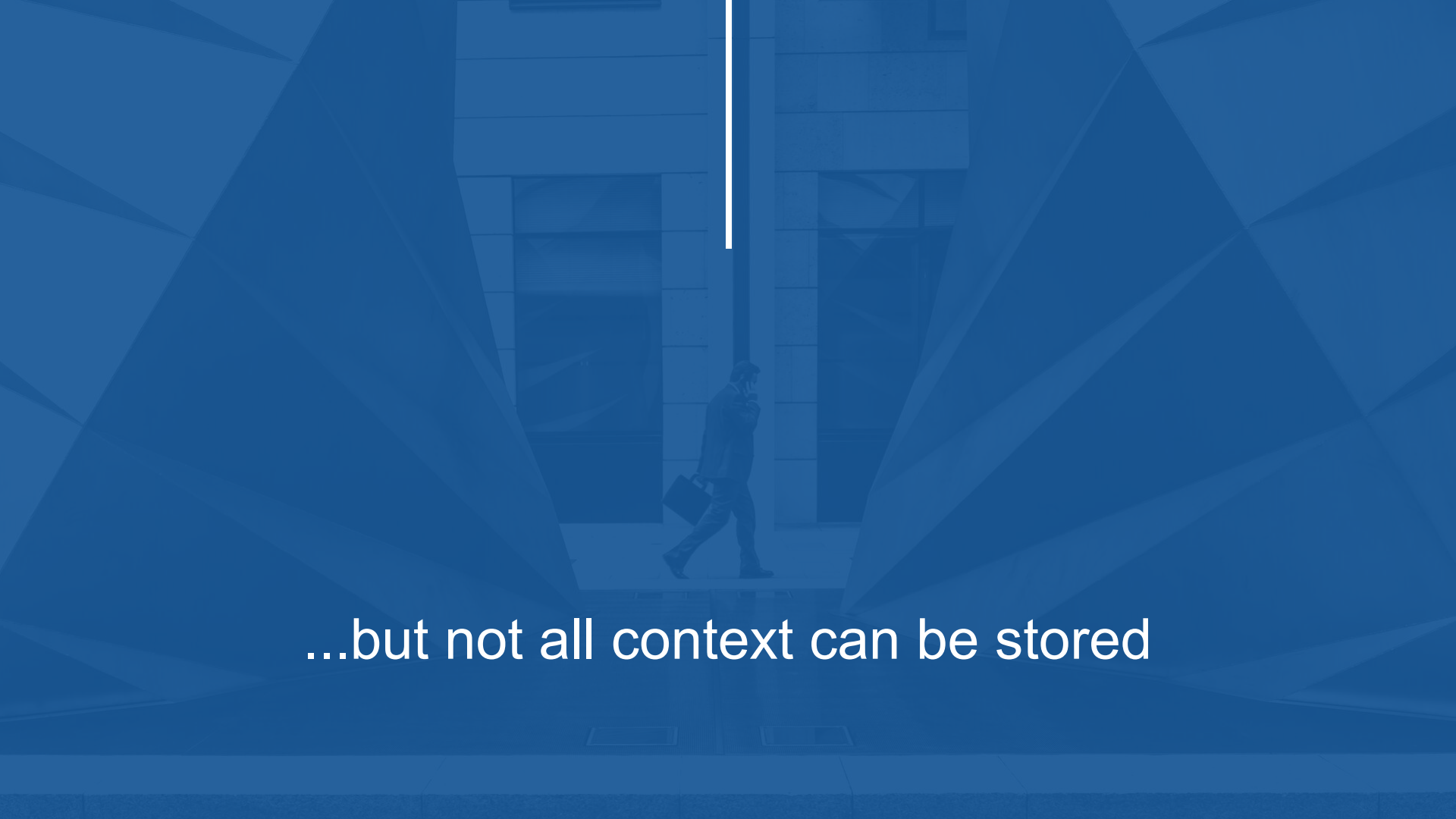
What Is Monitoring?

Monitoring = Events

- Receiving an HTTP request
- Sending an HTTP 500 response
- Entering a function
- Reaching the end of an if else statement
- Leaving a function
- A user logging in
- Writing data to disk
- And many more...

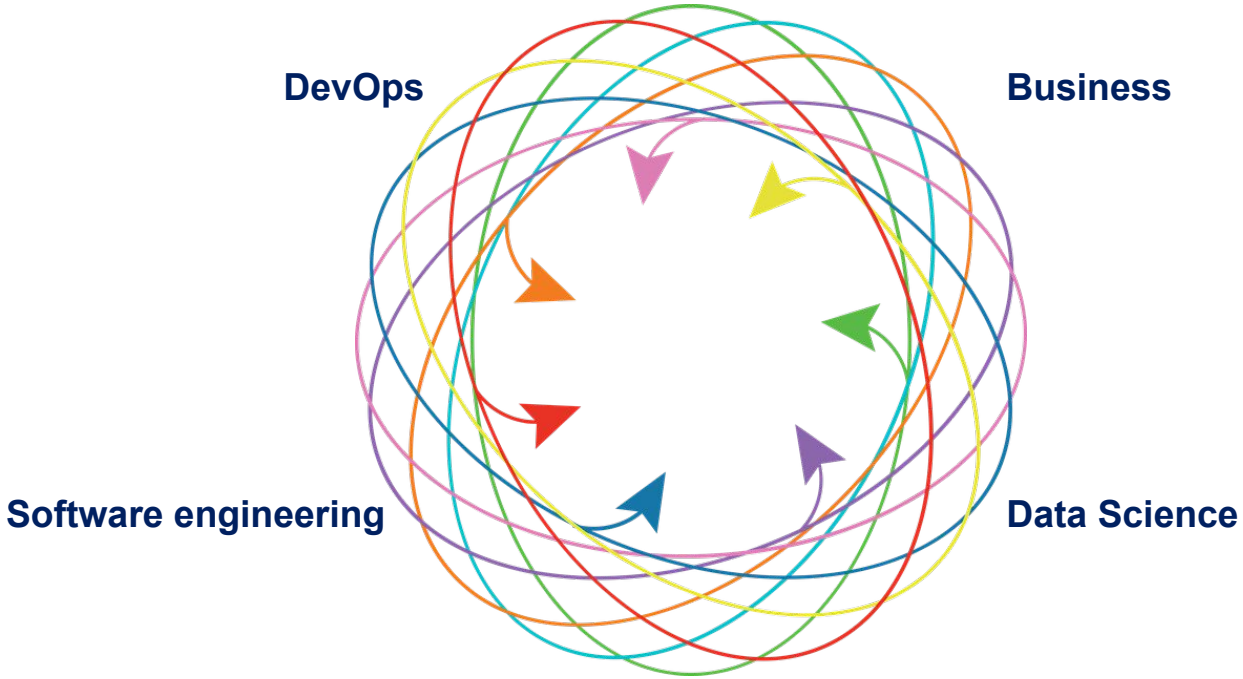


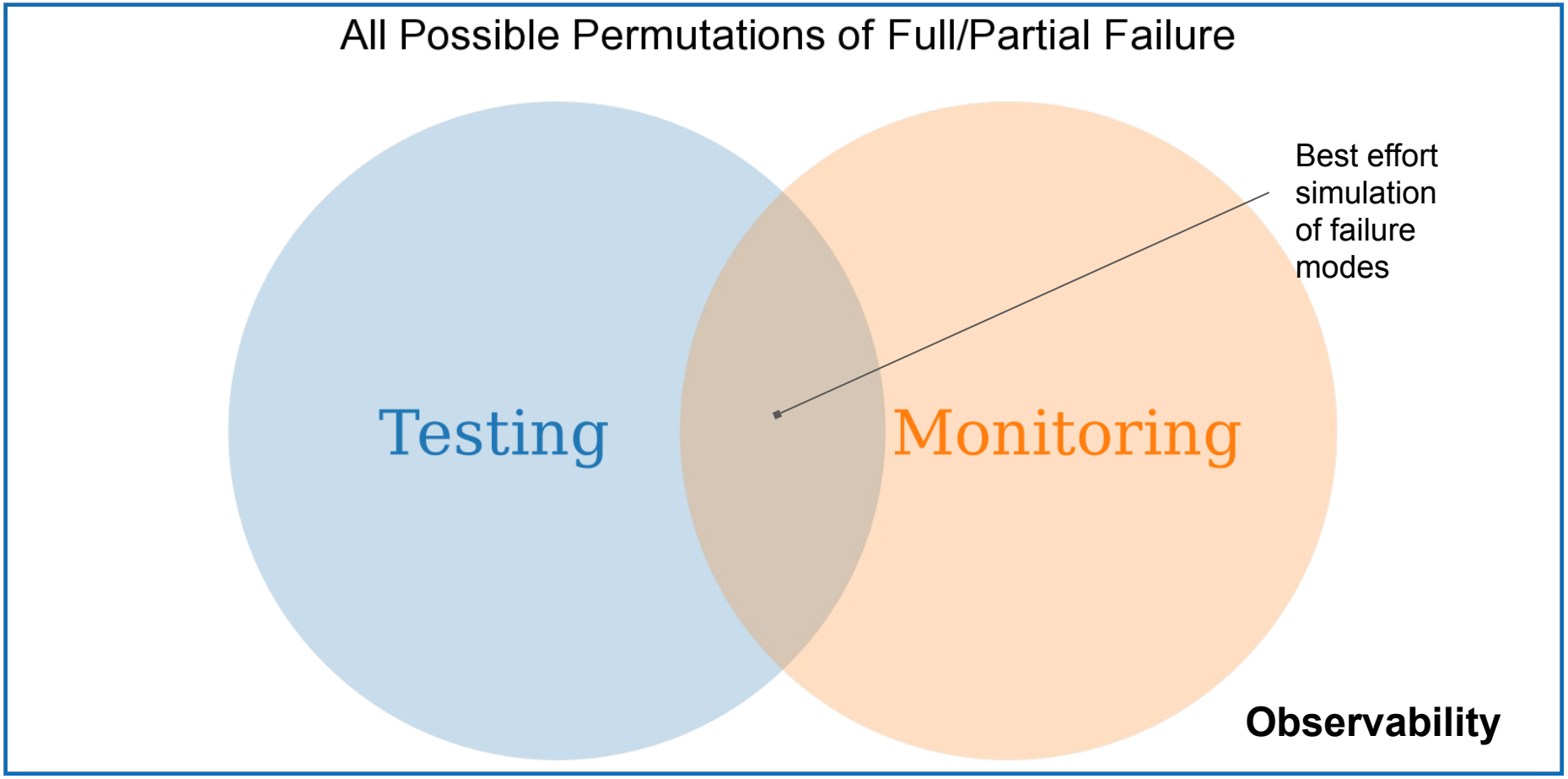
All Events Have a **Context**



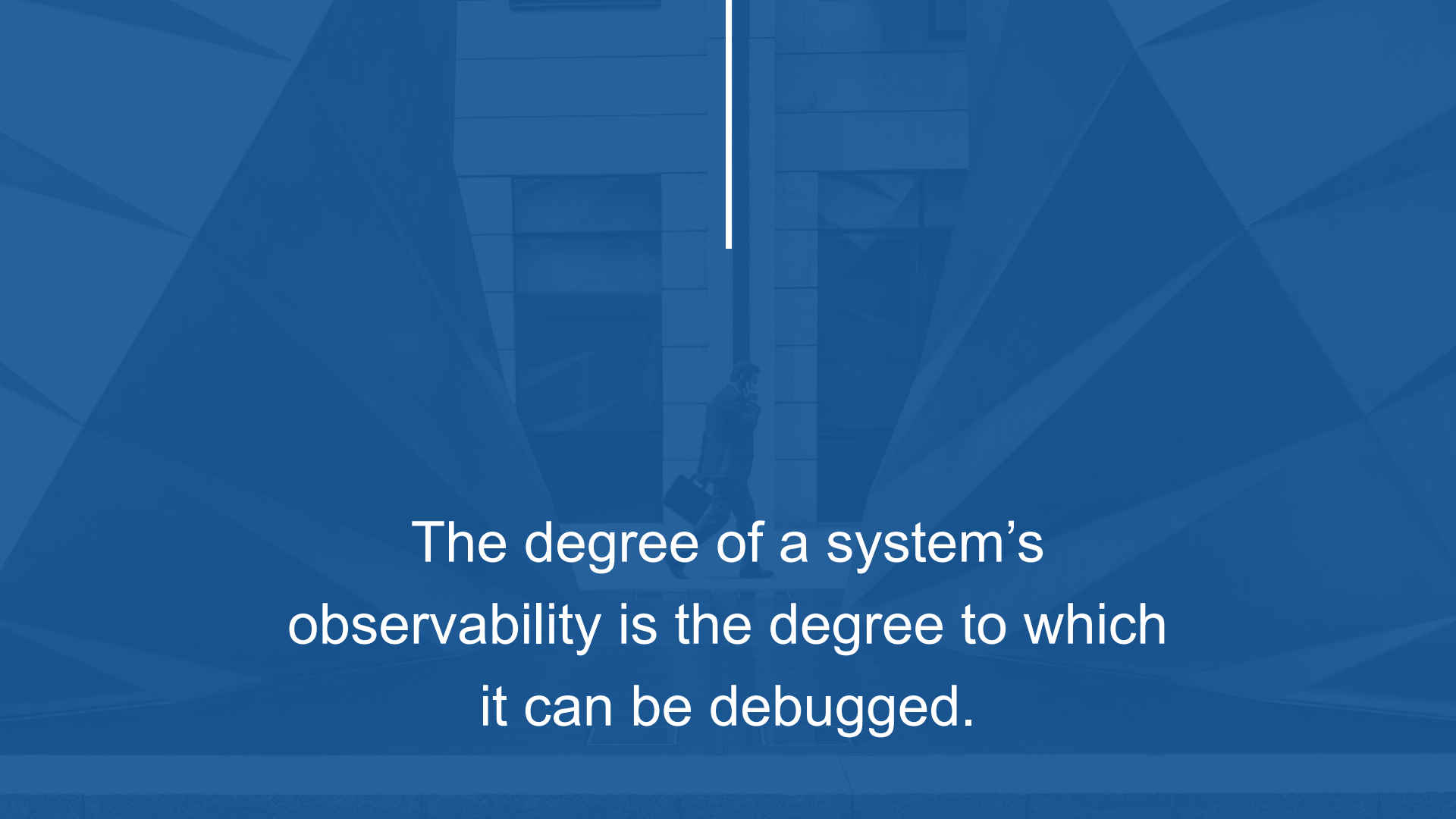
...but not all context can be stored

Monitoring Means Different Things to Different People





Adapted from: *Distributed Systems Observability*



The degree of a system’s observability is the degree to which it can be debugged.

The 3 Pillars of Observability

Logs

An immutable, timestamped record of events that happened over time.

Metrics

Are a numeric representation of data measured over intervals of time.

Distributed Tracing

A representation of a series of causally related distributed events that encode the end-to-end request flow through a distributed system

The 3 Pillars of Observability

Logs

An immutable, timestamped record of events that happened over time.

Metrics

Are a numeric representation of data measured over intervals of time.

Distributed Tracing

A representation of a series of causally related distributed events that encode the end-to-end request flow through a distributed system

The 3 Pillars of Observability

Logs

An immutable, timestamped record of events that happened over time.

Metrics

Are a numeric representation of data measured over intervals of time.

Distributed Tracing

A representation of a series of causally related distributed events that encode the end-to-end request flow through a distributed system

Alerting

Key Aspects of Alerting

Notify

Define situations that make sense to actively manage,

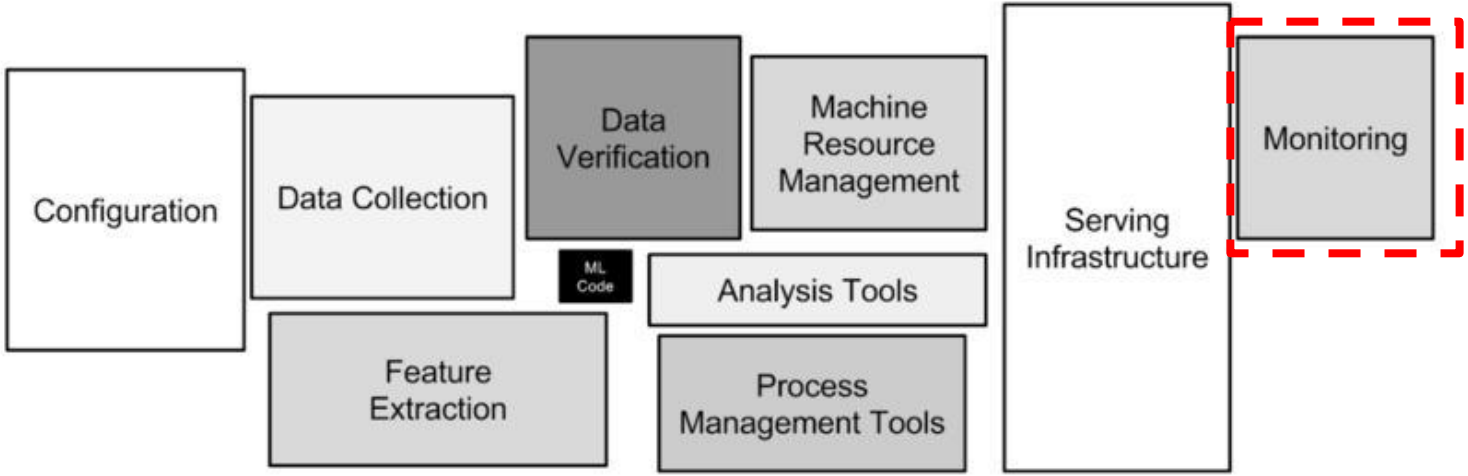
Automate

Programmatic responses can be triggered based on threshold violations as well.

Triage

Distinctions in alert severity depending on the scale of the problem.

Remember This Diagram?



Source: Sculley *et al*. (2015)

Monitoring Systems

Processing & Storage

Accept and store incoming and historical data

Visualization

Monitoring systems typically provide visualizations of data.

Alerting

Defining and activating alerts.

Key Monitoring Principles for ML: After Deployment

(Revisiting Breck et al. 2017)

- Monitor Model Predictions

  Check for skew, bias, staleness and other quality indicators

- Computational Performance

  Monitor system training speed, serving latency

THEORY

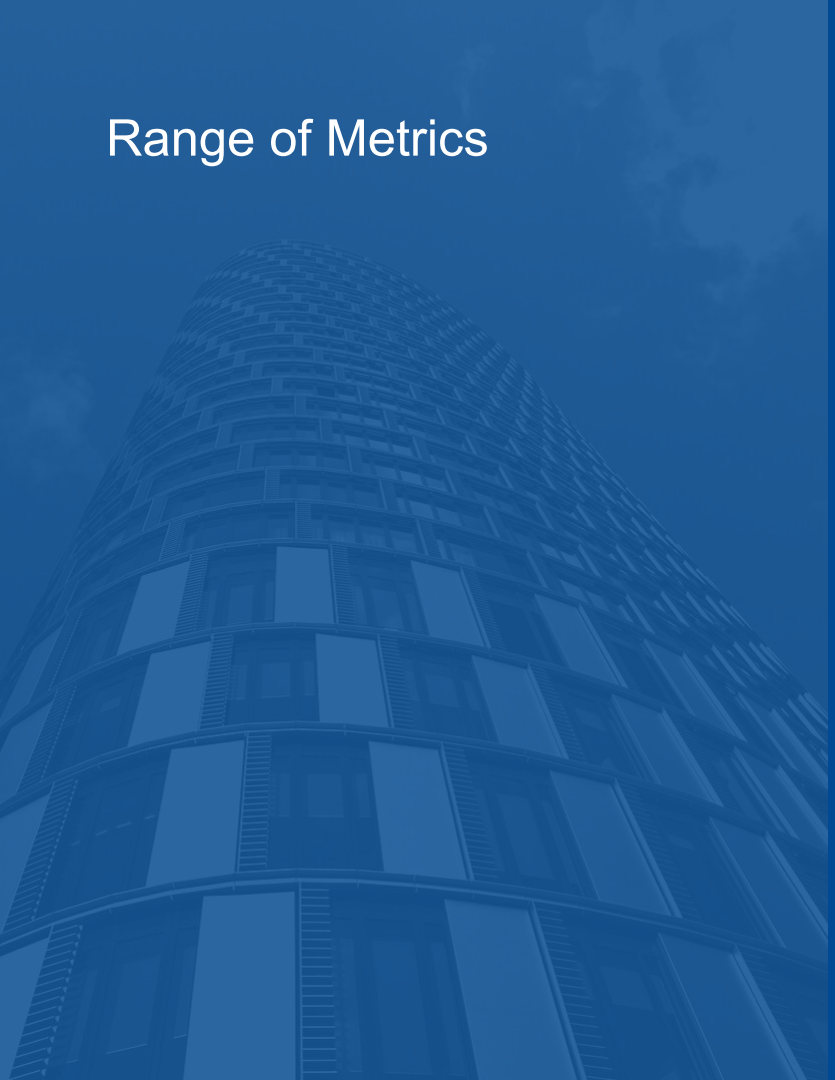
Monitoring Metrics for ML

Metrics

Are a numeric representation of data measured over intervals of time.

Real-Time Metrics

Range of Metrics



High Level & Low Level

Pros of Metrics

Constant overhead

Ideally suited to dashboards

Well-suited for alerting

Cons of Metrics

Not as information-rich as logs

Cardinality challenges

Scoped to a single system

Why Metrics for ML?

Key Part of Production Readiness

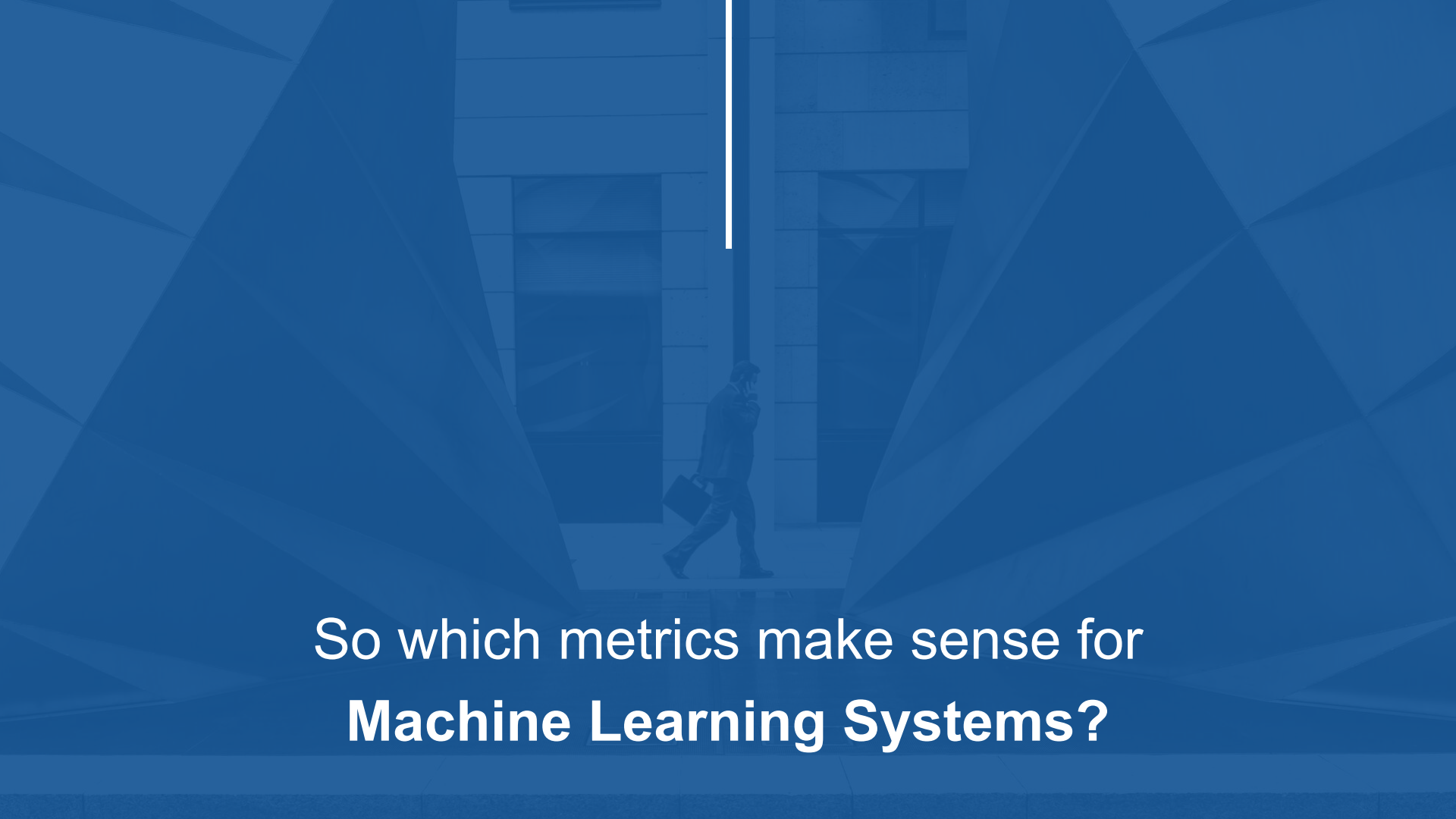
From both a DS & Operational perspective

Metrics Enable Alerting

Mitigate issues faster.

ML Systems Change Quickly

Due to concept drift.



So which metrics make sense for

Machine Learning Systems?

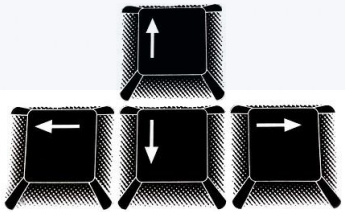
ML Metrics

Operational - Is it working?

- Latencies
- Memory size
- CPU usage

Are the predictions accurate?

- Model Outputs



Is the data what is expected?

- Model Inputs

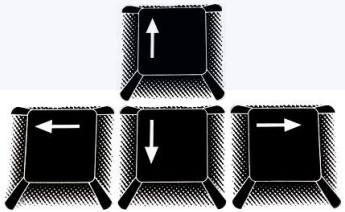
ML Metrics

Operational - Is it working?

- Latencies
- Memory size
- CPU usage

Are the predictions accurate?

- Model Outputs



Is the data what is expected?

- Model Inputs

ML Metrics

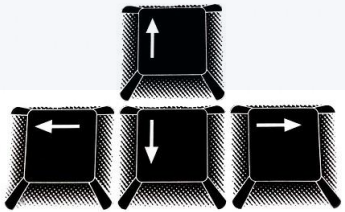
Data Skews

Operational - Is it working?

- Latencies
- Memory size
- CPU usage

Are the predictions accurate?

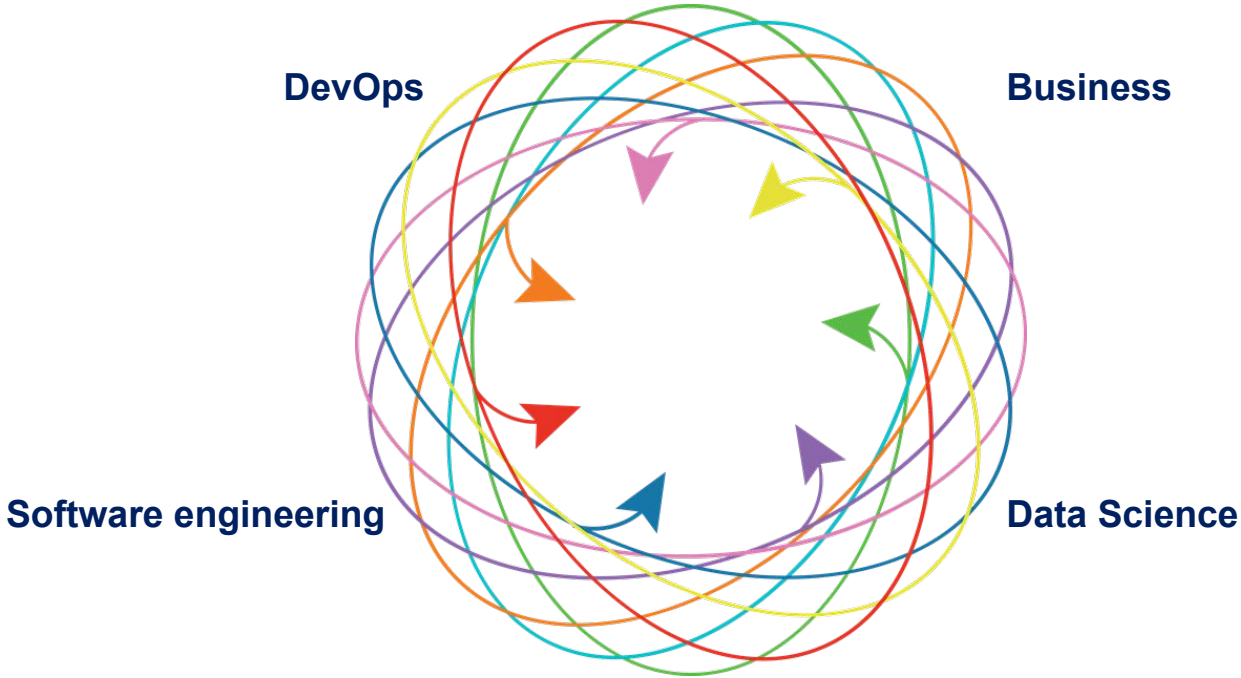
- Model Outputs



Is the data what is expected?

- Model Inputs

Metrics & Their Owners



THEORY

Understanding Prometheus & Grafana

Monitoring Systems

Processing & Storage

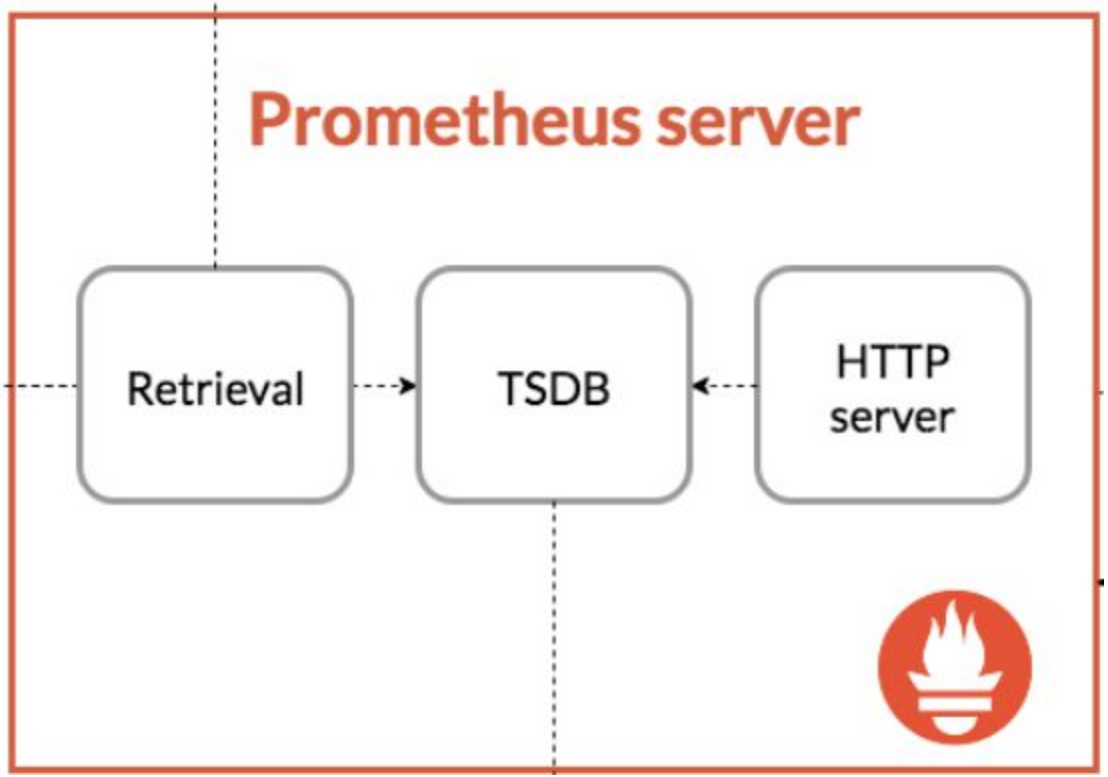
Accept and store incoming and historical data

Visualization

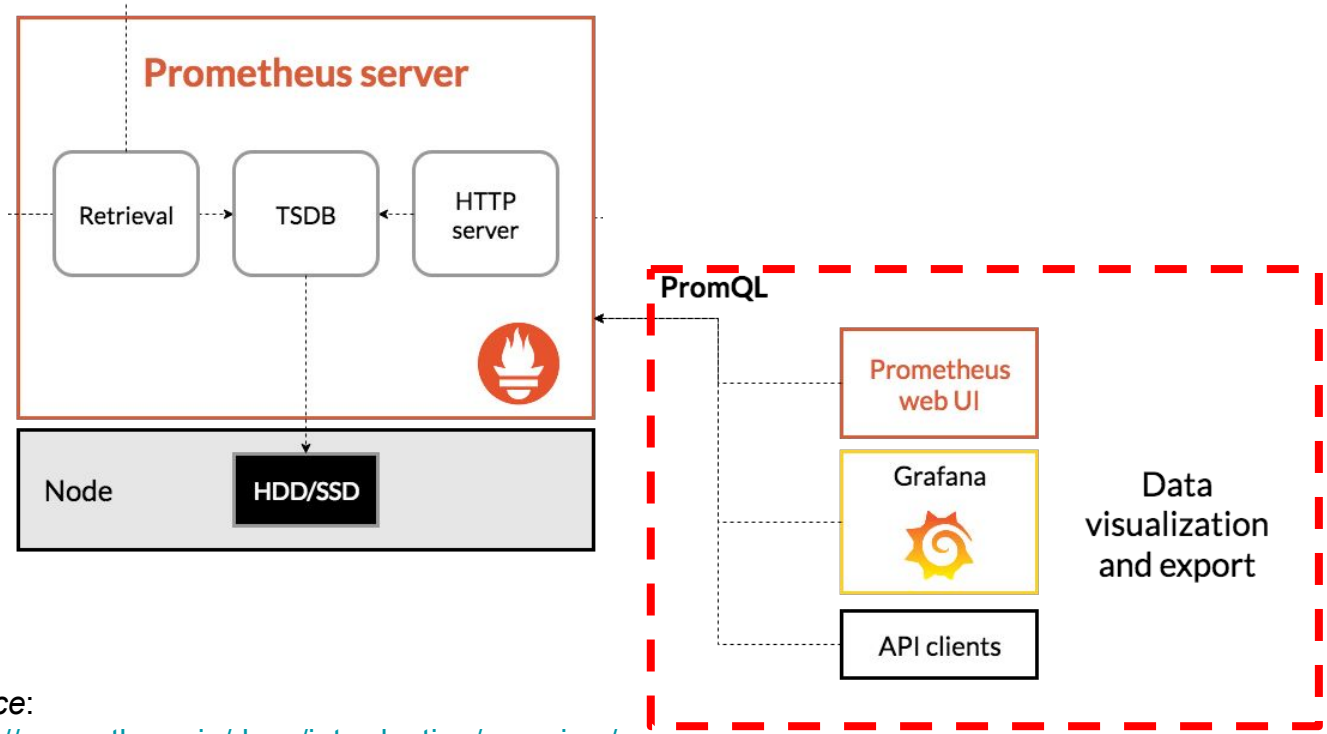
Monitoring systems typically provide visualizations of data.

Alerting

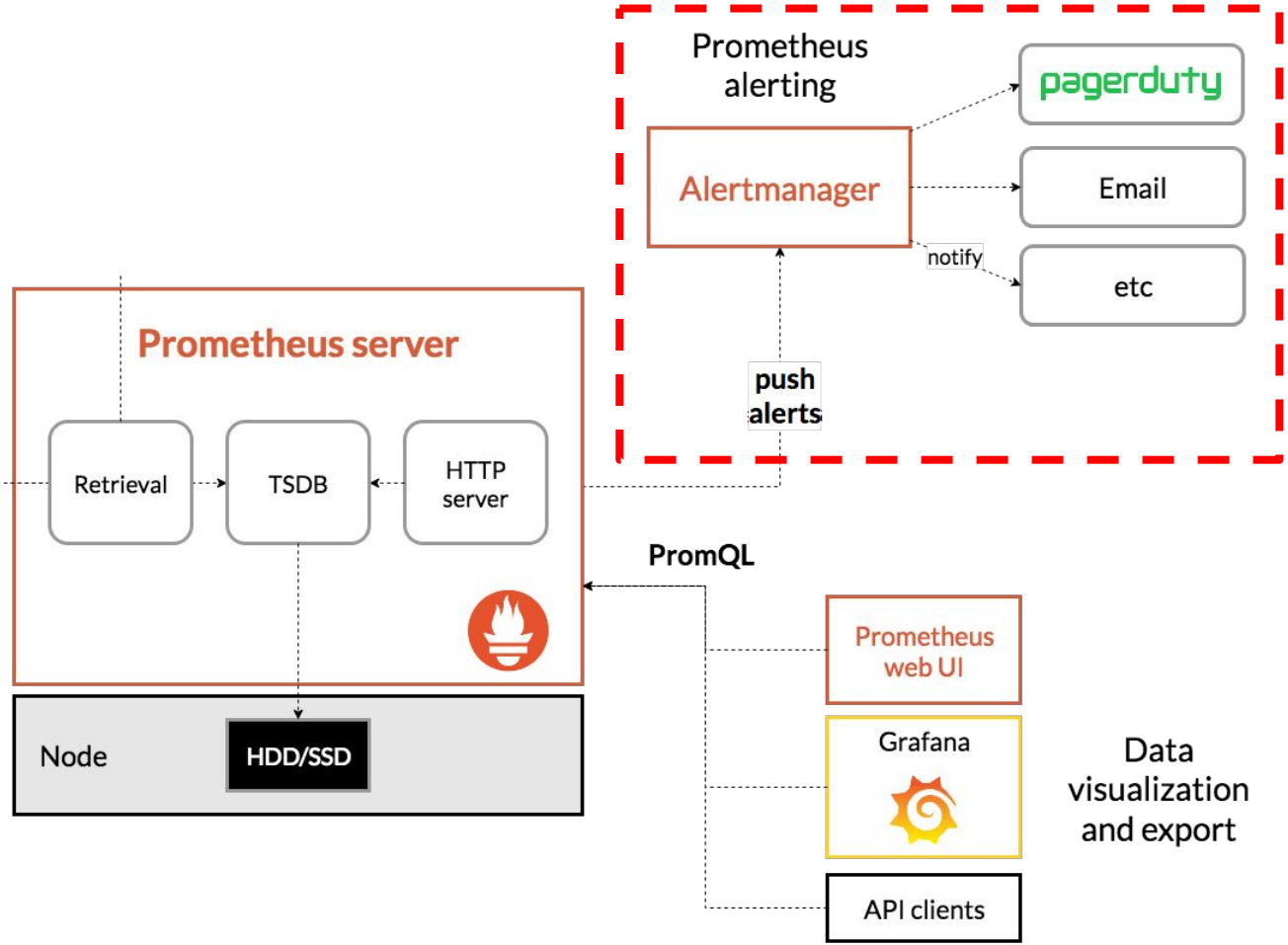
Defining and activating alerts.



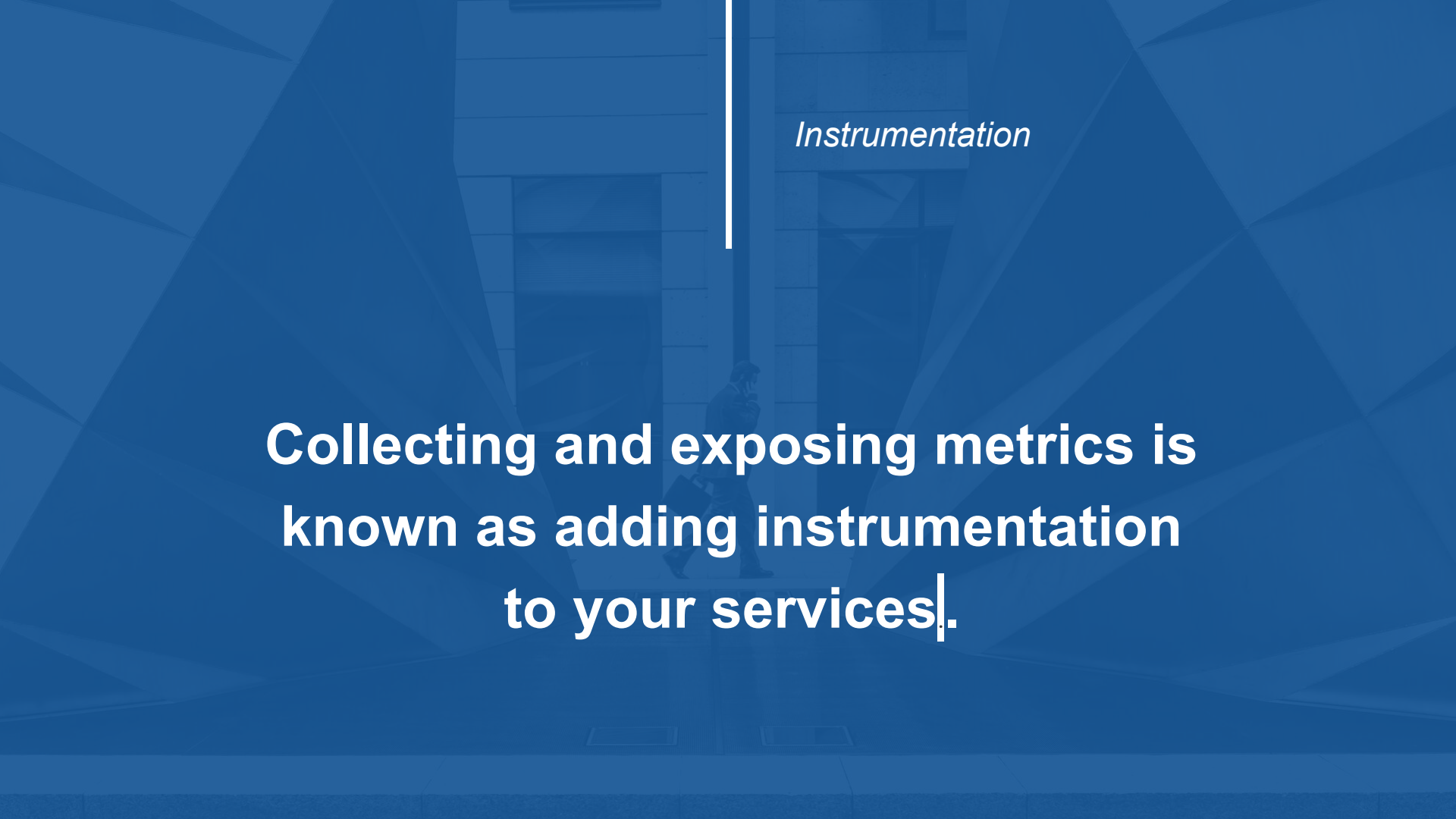
Source:
https://prometheus.io/docs/introduction/overview/



Source:
https://prometheus.io/docs/introduction/overview/

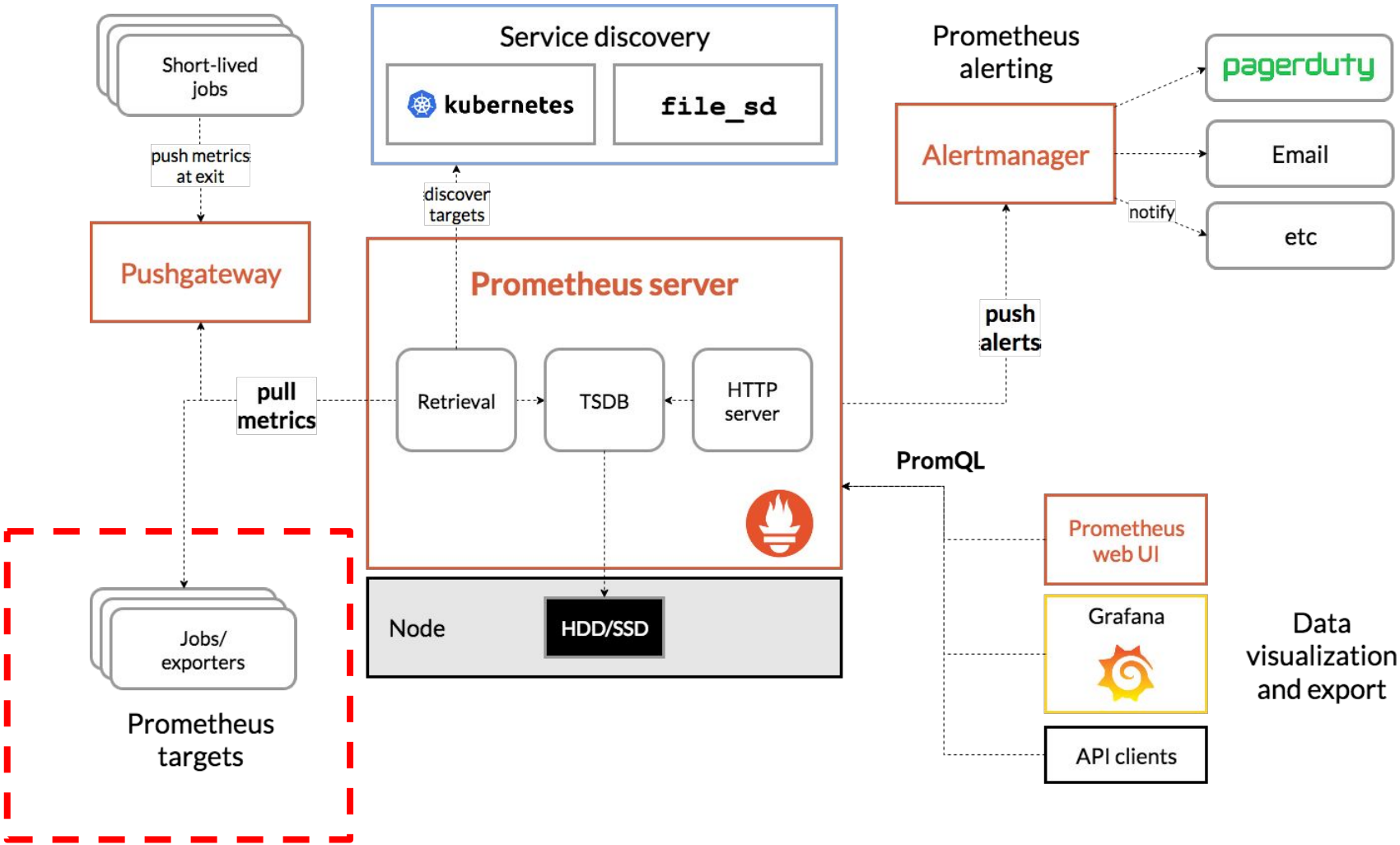


Source:
https://prometheus.io/docs/introduction/overview/

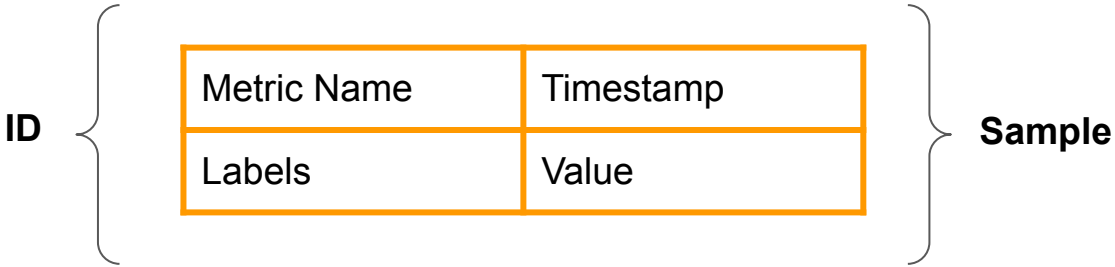


Instrumentation

Collecting and exposing metrics is known as adding instrumentation to your services.


Short-lived jobs
push metrics at exit
Pushgateway
Service discovery
kubernetes
file_sd
discover targets
Prometheus server
Retrieval
TSDB
HTTP server
pull metrics
Node
HDD/SSD
Jobs/ exporters
Prometheus targets
Prometheus alerting
Alertmanager
pagerduty
Email
etc
notify
push alerts
PromQL
Prometheus web UI
Grafana
API clients
Data visualization and export

A Metric In Prometheus



Two Components

Float64 Value

Timestamp

Revisiting Cardinality

How much to Instrument?

Metrics can add up fast.

Watch Labels

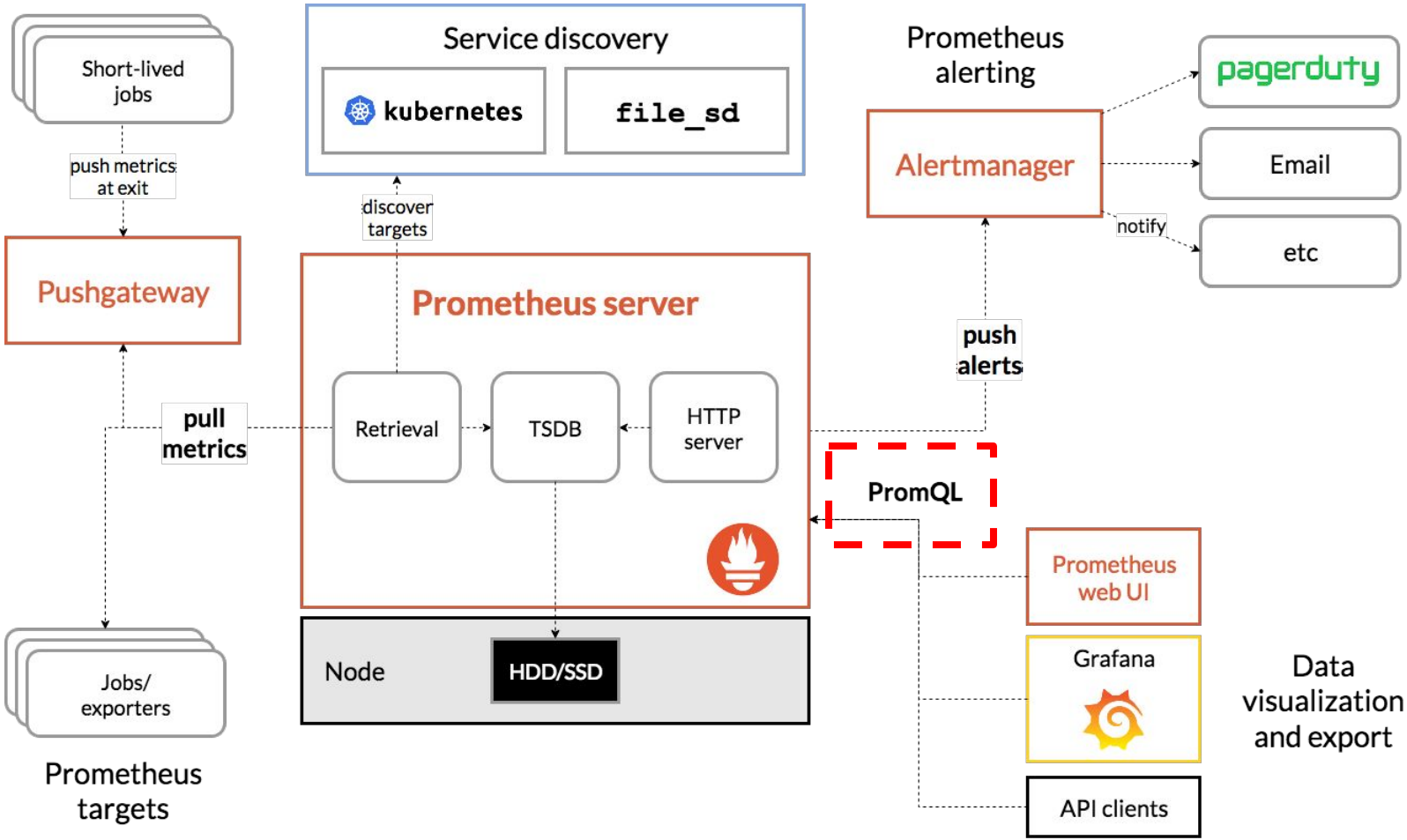
High label cardinality kills performance.

Common Problem

This is the *the* classic mistake with Prometheus.

Prometheus is not for Logs!

As a metrics-based system, Prometheus is not suitable for storing event logs or individual events. Nor is it the best choice for high cardinality data, such as email addresses or usernames.


Short-lived jobs
push metrics at exit
Pushgateway
Service discovery
kubernetes
file_sd
discover targets
Prometheus server
Retrieval
TSDB
HTTP server
pull metrics
Node
HDD/SSD
Jobs/ exporters
Prometheus targets
Prometheus alerting
Alertmanager
pagerduty
Email
etc
notify
push alerts
PromQL
Prometheus web UI
Grafana
API clients
Data visualization and export

PromQL Expression Evaluation Types

- **Instant vector**

  A set of time series containing a single sample for each time series, all sharing the same timestamp

- **Scalar**

  A simple numeric floating point value

- **Range Vector**

  A set of time series containing a range of data points over time for each time series

- **String**

  A simple string value

Instant Vector

```
http_requests_total{job="apiserver", handler="/api/comments"}
```

Range Vector

```
http_requests_total{job="apiserver", handler="/api/comments"}[5m]
```

Functions

```
rate(http_requests_total[5m])
```

THEORY

Monitoring Logs for ML

Logs

An immutable, timestamped record of events that happened over time.

Pros of Logs

Easy to generate

Logs contain more context

Highly effective within a single service

Cons of Logs

Performance implications

Less suitable for alerting

Effective processing at scale requires significant infrastructure

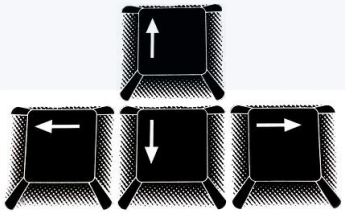
ML Monitoring

Operational - Is it working?

- Latencies
- Memory size
- CPU usage

Are the predictions accurate?

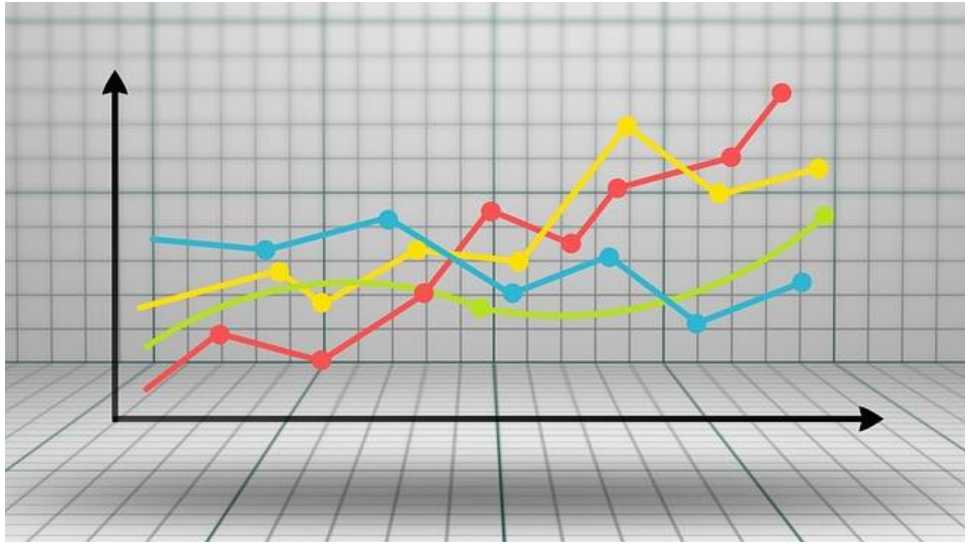
- Model Outputs



Is the data what is expected?

- Model Inputs

Alerting



- A feature becoming unavailable
- Shifts in distribution of key input values
- Patterns specific to your model

Introduction to Kibana & The Elastic Stack

THEORY

Monitoring Systems

Processing & Storage

Accept and store incoming and historical data

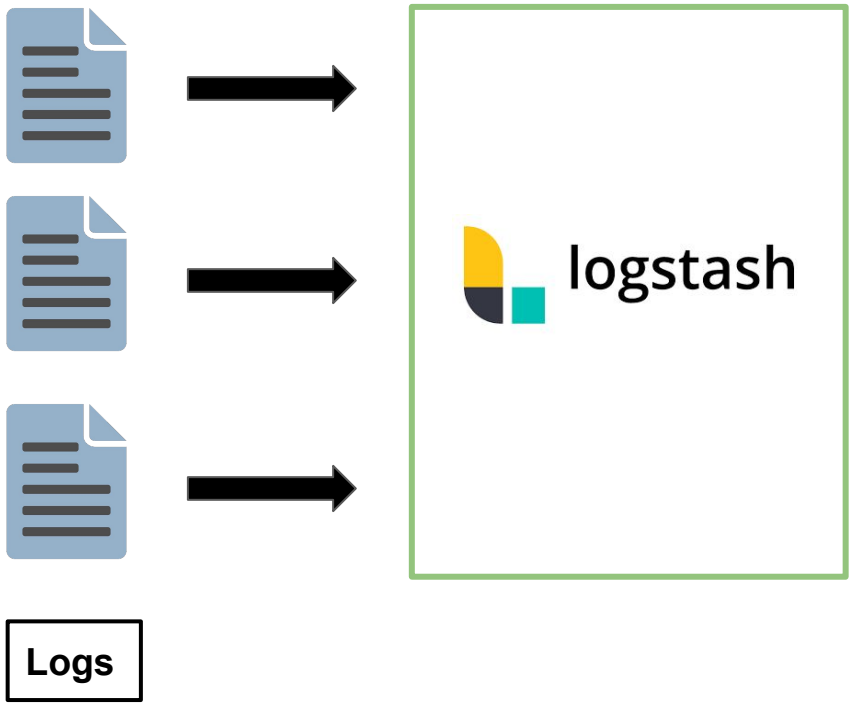
Visualization

Monitoring systems typically provide visualizations of data.

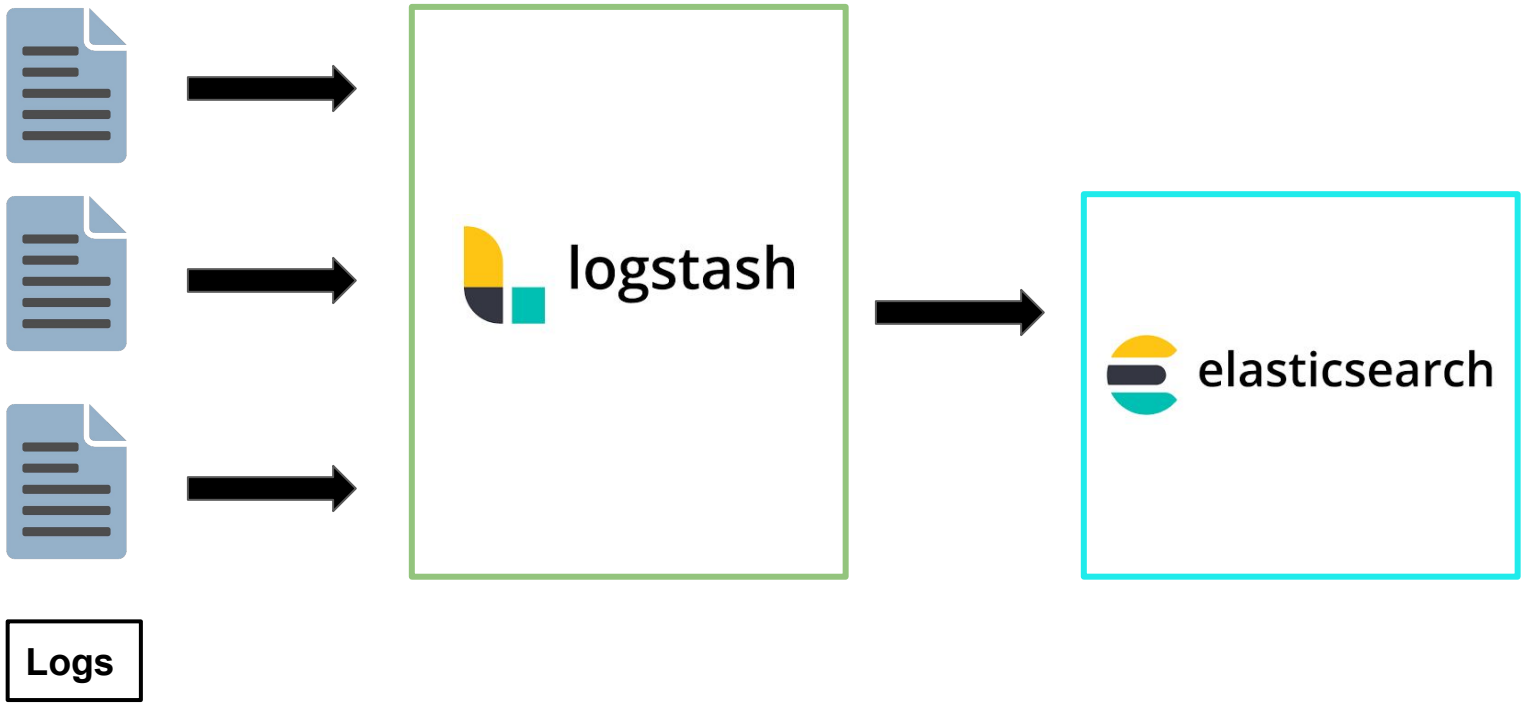
Alerting

Defining and activating alerts.

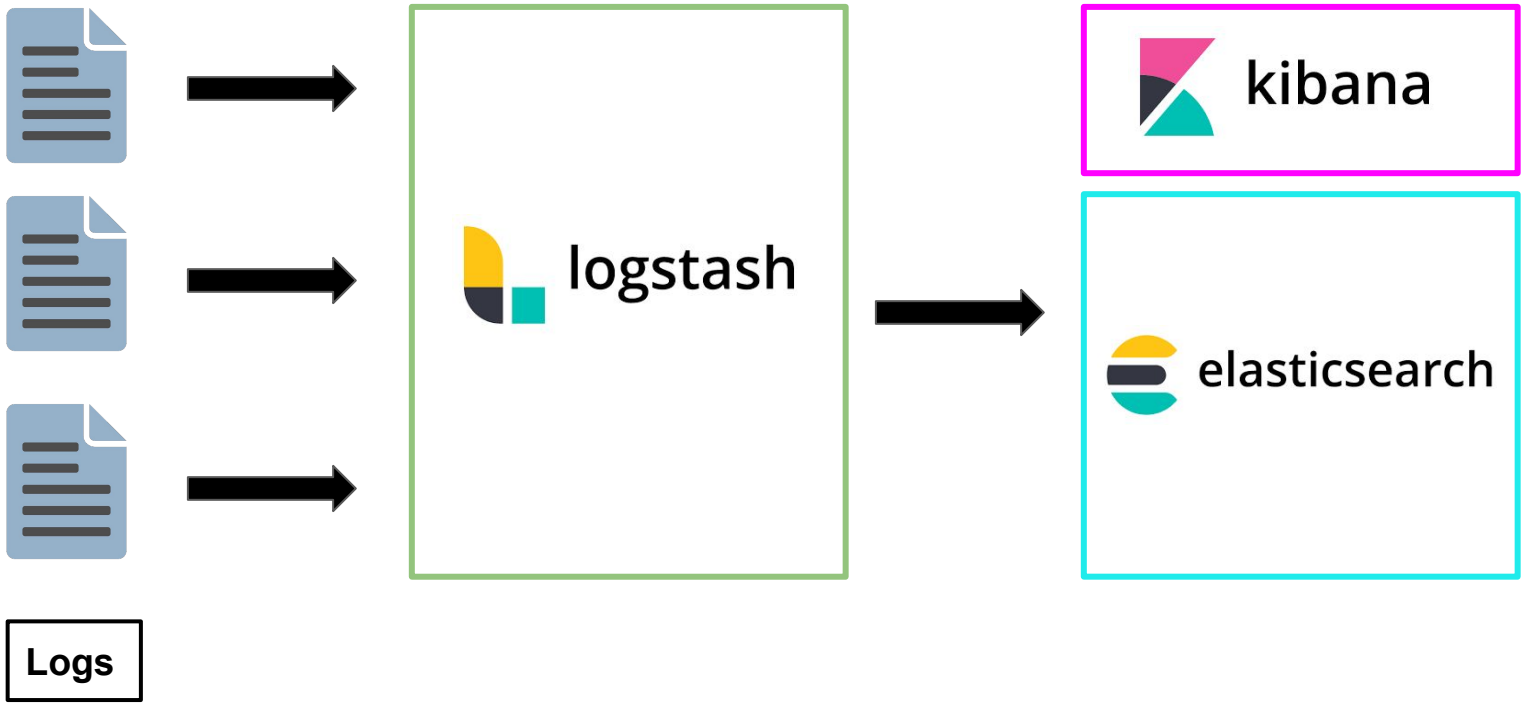
The Elastic Stack (ELK Stack) Components



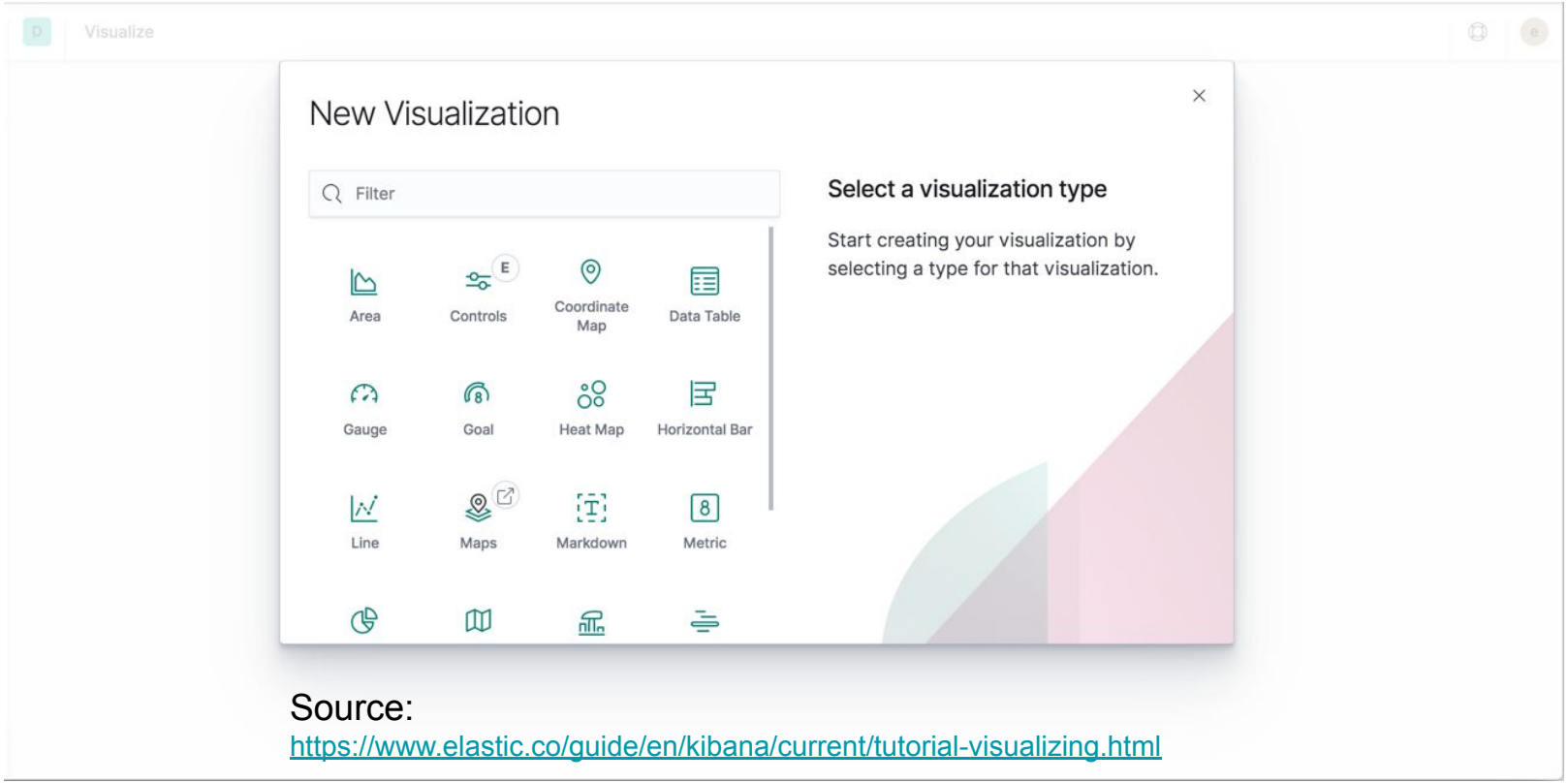
The Elastic Stack (ELK Stack) Components



The Elastic Stack (ELK Stack) Components



Kibana Visualization



Source:
https://www.elastic.co/guide/en/kibana/current/tutorial-visualizing.html

Alerting

Elasticsearch

Anything you can query for in Elasticsearch can be configured to be an alert.

Advantages

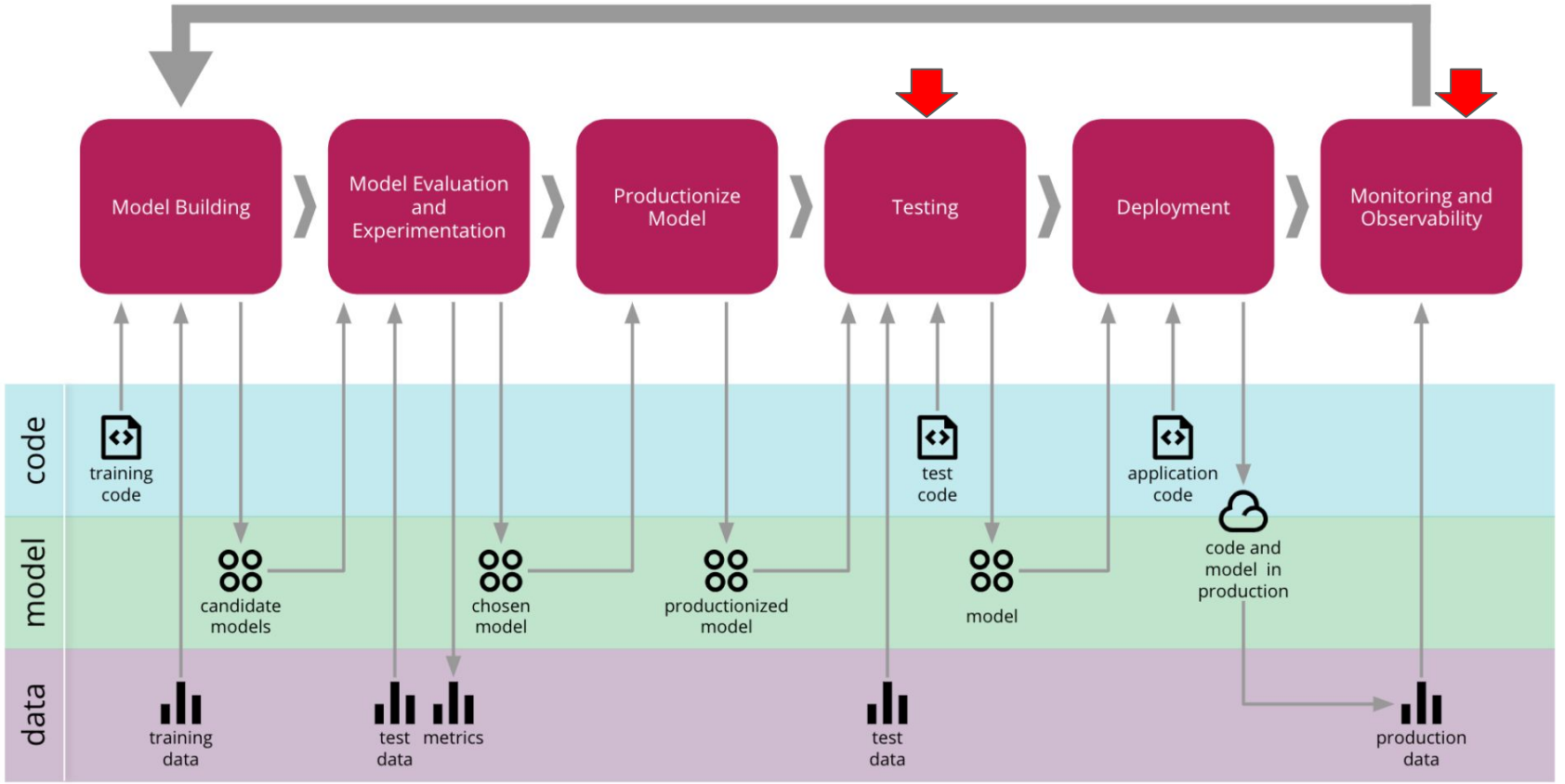
Text-parsing alerts can be particularly useful for detecting sensitive information.

Kibana UI

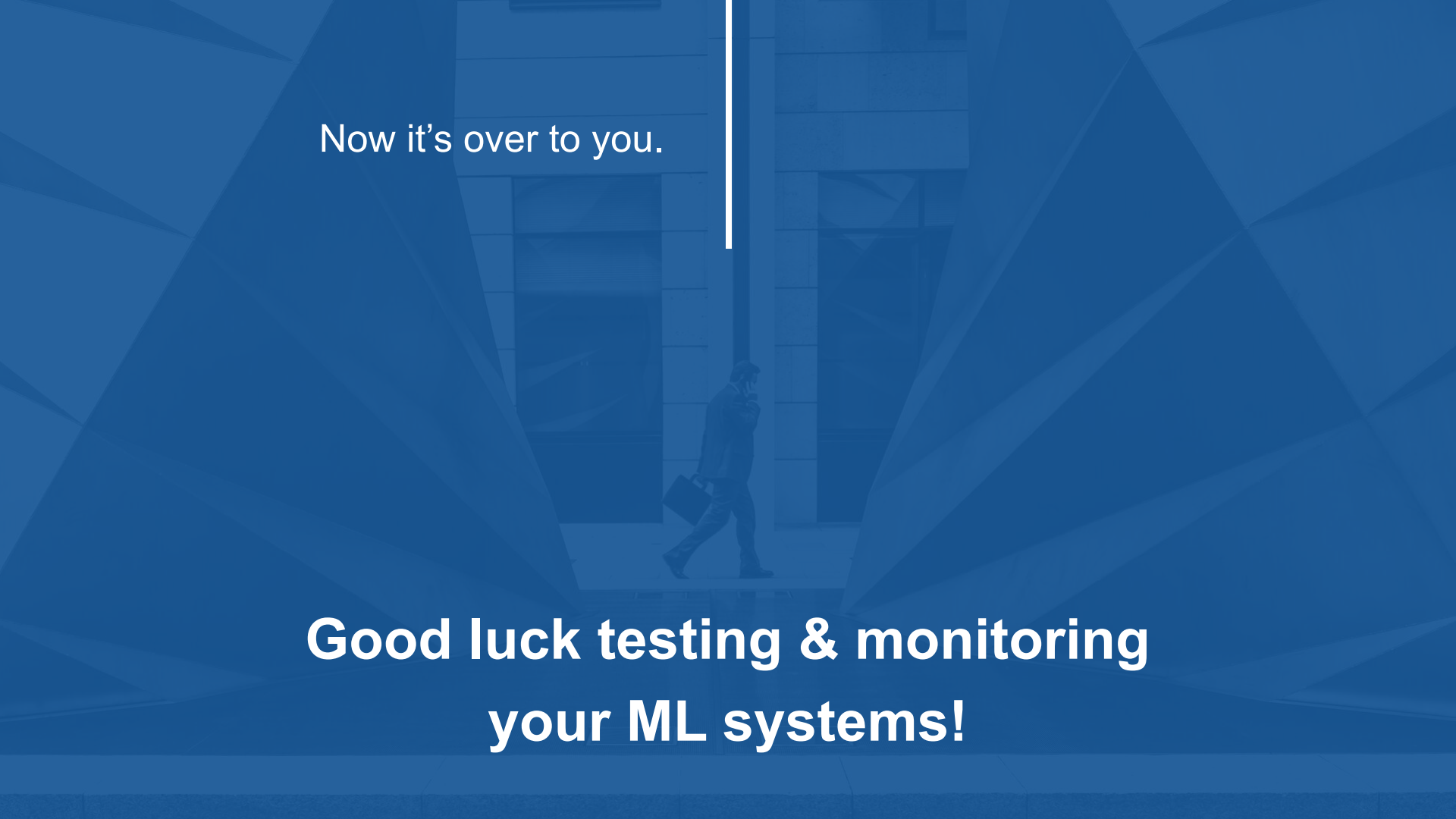
Alerts can be configured within Kibana, with 3rd party integrations

Conclusion

ML System Lifecycle (CD4ML)



Source: https://martinfowler.com/articles/cd4ml.html



Now it’s over to you.

Good luck testing & monitoring your ML systems!