

# Continuous Delivery for Machine Learning - ODSC East 2020

Christoph Windheuser - cwindheu@thoughtworks.com

David Johnson - dajohnst@thoughtworks.com

Eric Nagler - eric.nagler@thoughtworks.com

**ThoughtWorks®**

# Your Workshop Team



**Christoph Windheuser**  
**Global Head of Artificial  
Intelligence**

[cwindheu@thoughtworks.com](mailto:cwindheu@thoughtworks.com)



**David Johnston**  
**Principal Data Scientist**

[dajohnst@thoughtworks.com](mailto:dajohnst@thoughtworks.com)



**Eric Nagler**  
**Lead Data Engineer**

[eric.nagler@thoughtworks.com](mailto:eric.nagler@thoughtworks.com)

[www.thoughtworks.com](http://www.thoughtworks.com)



ThoughtWorks®

*7000 technologists with 40 offices in 14 countries*



*Partner for technology driven business transformation*



*Atlanta - Chicago - Dallas - Denver - New York - San Francisco - Toronto*

100+

books written

#1

in Agile and  
Continuous Delivery

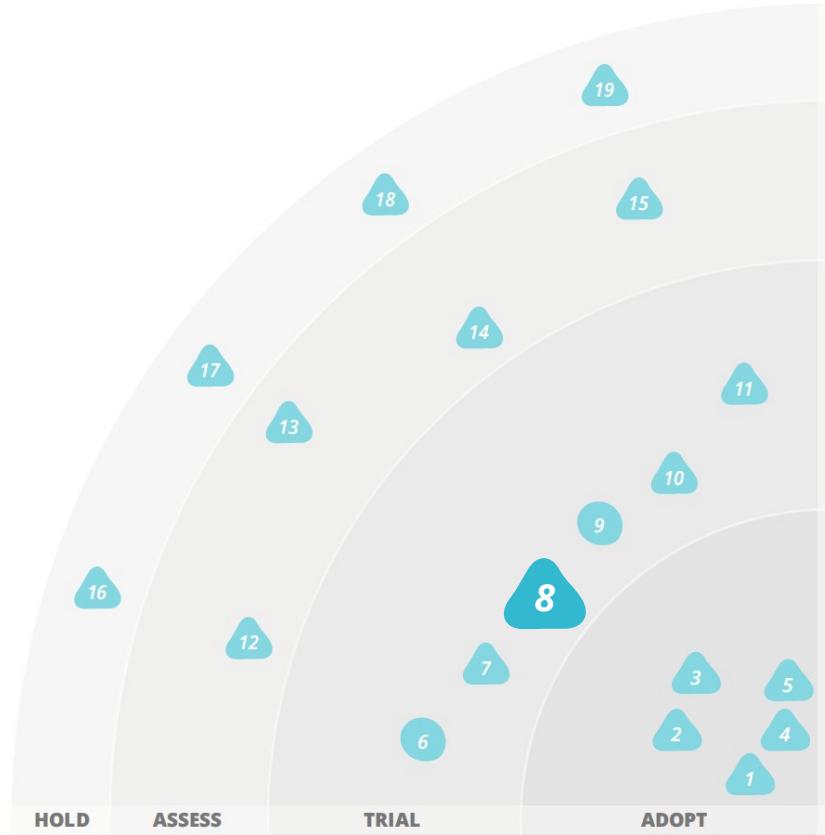




## TECHNIQUES

Continuous delivery #8  
for machine  
learning (CD4ML)  
models

TRIAL



# Agenda

1. Setting up the local environment on your laptop (20 min)
2. Continuous Delivery for Machine Learning (CD4ML) (40 min)
3. Let's start: Running through the different steps of the process (2h)

**Slack: #wed\_christoph-windheuser-david-johnston-eric-nagler\_data-science-best-practices**

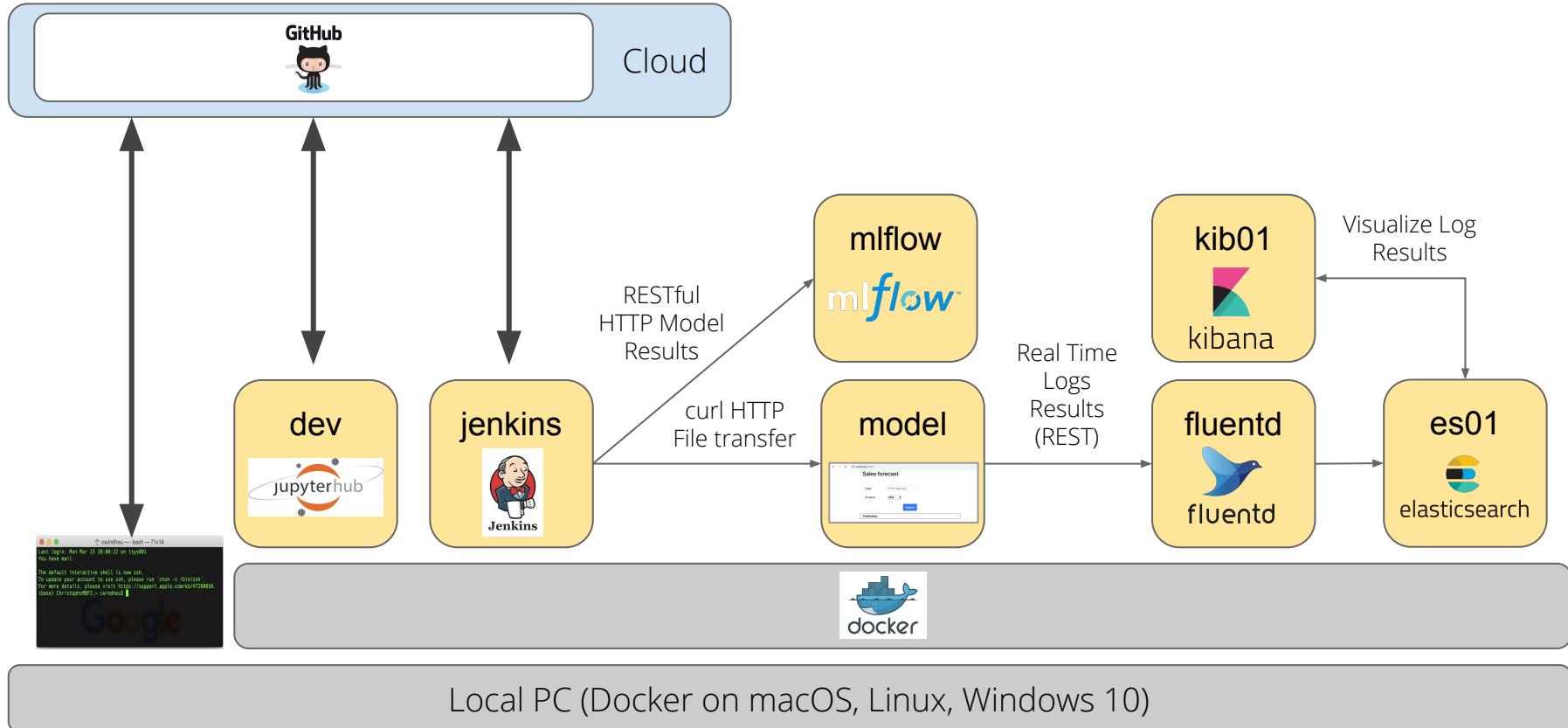
# Setting up your local environment

# Prerequisites

- PC with Mac OS, Windows 10 or Linux
- Docker installed on your PC
- Min 20 GB free space on your local hard drive
- Github account
- Git client installed on your machine
- Enable your slack account:

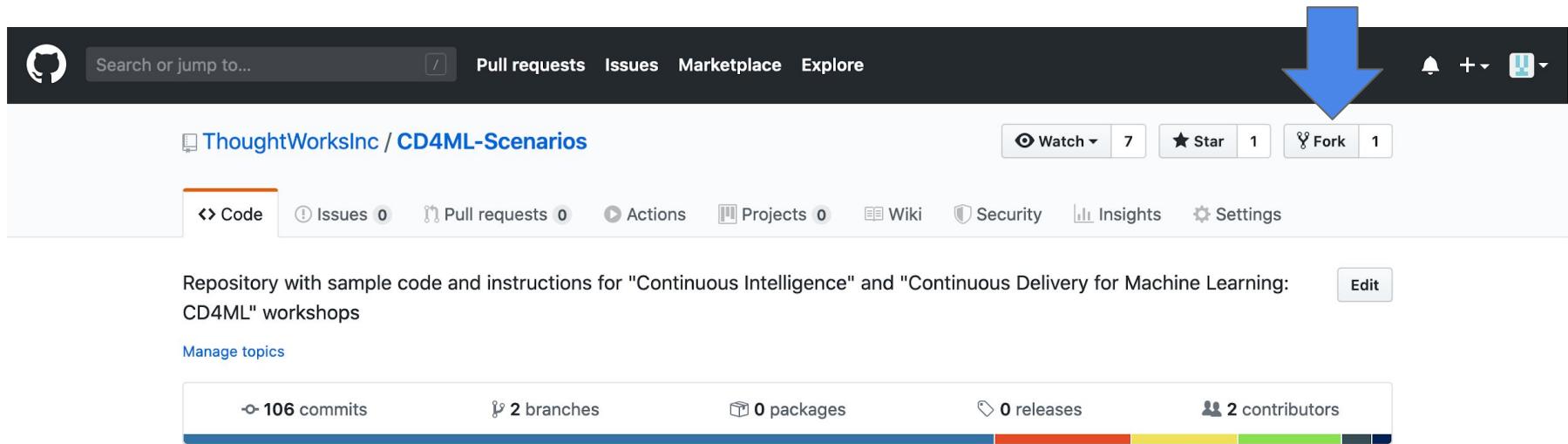
**#wed\_christoph-windheuser-david-johnston-eric-nagler\_data-science-best-practices**

# The Workshop Infrastructure



# Installation (1)

- Navigate to <https://github.com/ThoughtWorksInc/CD4ML-Scenarios>
- Fork the repo to your Personal GitHub Account



The screenshot shows a GitHub repository page for 'ThoughtWorksInc / CD4ML-Scenarios'. At the top, there is a navigation bar with links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation bar, the repository name 'ThoughtWorksInc / CD4ML-Scenarios' is displayed, along with buttons for 'Watch' (7), 'Star' (1), and 'Fork' (1). A large blue arrow points to the 'Fork' button. The main content area contains a brief description: 'Repository with sample code and instructions for "Continuous Intelligence" and "Continuous Delivery for Machine Learning: CD4ML" workshops'. There is a 'Manage topics' link below the description. At the bottom, there is a summary of repository statistics: 106 commits, 2 branches, 0 packages, 0 releases, and 2 contributors. A horizontal bar at the bottom is composed of colored segments corresponding to the repository's activity.

# Installation (2)

- Create a [github personal access token](#) with the following permissions to check out the code.
- When you click Generate Token a token will be displayed to you. Please securely save this token because it will only be displayed once

The screenshot shows the GitHub developer settings page. The navigation bar includes the GitHub logo, a search bar, and links for Pull requests, Issues, Marketplace, and Explore. The user has notifications and a profile icon. The main area shows the 'Developer settings' section with tabs for GitHub Apps, OAuth Apps, and Personal access tokens, which is currently selected. A note says: "Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#)." A note field contains "ODSC-Jenkins". A question "What's this token for?" is present. Under "Select scopes", there are two sections: "repo" and "user". The "repo" section has checkboxes for repo, repo:status, repo\_deployment, public\_repo, and repo:invite, all of which are checked. The "user" section has checkboxes for user, read:user, user:email, and user:follow, where only user:email is checked. Descriptions for each scope are provided.

Settings / Developer settings

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

ODSC-Jenkins

What's this token for?

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

Scope	Description
<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> user	Update all user data
<input type="checkbox"/> read:user	Read all user profile data
<input checked="" type="checkbox"/> user:email	Access user email addresses (read-only)
<input type="checkbox"/> user:follow	Follow and unfollow users

# Installation (3)

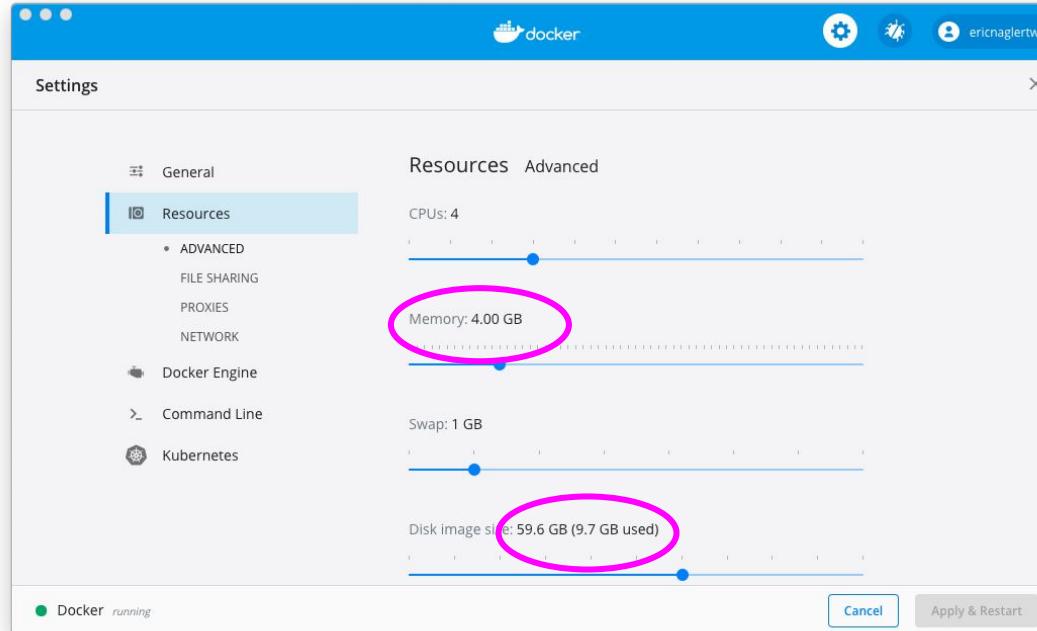
- Navigate to “<your github username>/CD4ML-Scenarios/instructions/1-SystemSetup.md” to begin setting up your development environment

The screenshot shows a GitHub repository page for 'ThoughtWorksInc / CD4ML-Scenarios'. The repository has 7 pull requests, 1 star, and 3 forks. The main navigation bar includes links for Pull requests, Issues, Marketplace, and Explore. Below the navigation bar, there are tabs for Code, Issues (0), Pull requests (0), Actions, Projects (0), Wiki, Security, Insights, and Settings. The current view is on the 'Code' tab, specifically the file '1-SystemSetup.md' under the 'master' branch. The file was last updated by 'ericnagler' 1 hour ago, with commit hash 0065f45. It has 3 contributors. The file content starts with a section titled 'Setting up your environment' and a 'Goals' section.

```
graph TD; A[ThoughtWorksInc / CD4ML-Scenarios] --> B[1-SystemSetup.md]; B --> C[Setting up your environment]; C --> D[Goals]; D --> E[• Setup a development environment for CD4ML including:]; E --> F[• Fork Git Code Repository into Personal GitHub]; F --> G[• Configuring a python code development environment]
```

# Installation (4)

- Increase the RAM allocated to docker to 4 Gigabytes (or more)
- Ensure you have about 20 GB Free on your Docker image



# Installation (5)

Configuring your jenkins password

- Create a file in jenkins/jenkins-admin-password.txt
- Open the file and put in a secure password on the first line. Save and close the file.

# Installation (6)

- Your choice of code development environment
  - JupyterHub Notebook
    - Ready to go Jupyter Environment with python and JupyterLab ready
    - Follow the "Setting up your local environment (using JupyterLab Development Environment)"
  - Local Machine virtualenv environment
    - Use your preferred tools for python development
    - Follow the "Setting up your local environment (using local machine environment)"
- Start the environment using docker-compose to download the images and software

```
docker-compose up -d --build
```

```
Creating jenkins ... done
Creating es01    ... done
Creating dev     ... done
Creating fluentd ... done
Creating kib01   ... done
Creating model   ... done
Creating mlflow   ... done
+ CD4ML-Scenarios git:(master) [
```



# QUESTIONS?



# Continuous Delivery for Machine Learning (CD4ML)

# PRODUCTIONIZING ML IS HARD

## ***HOW DO WE APPLY DECADES OF SOFTWARE DELIVERY EXPERIENCE TO INTELLIGENT SYSTEMS?***

Production systems should be:

- Reproducible
- Testable
- Auditable
- Continuously Improving

Machine Learning is:

- Non-deterministic (somewhat)
- Hard to test
- Hard to explain
- Hard to improve

CD4ML isn't a technology or a tool; it is a practice and a set of principles. Quality is built into software and improvement is always possible.

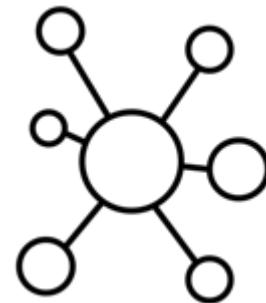
But machine learning systems have unique challenges; unlike deterministic software, it is difficult—or impossible—to understand the behavior of data-driven intelligent systems. This poses a huge challenge when it comes to deploying machine learning systems in accordance with CD principles.

# MANY SOURCES OF CHANGE

0101110  
0111010  
0101110



Data



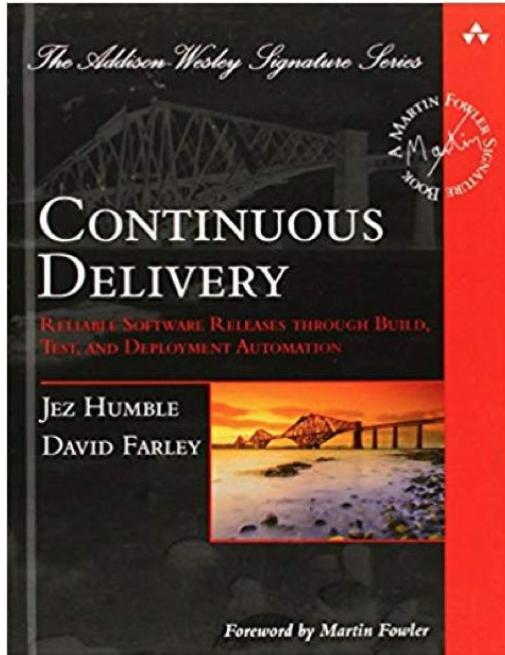
Model



Code

# PRINCIPLES OF CONTINUOUS DELIVERY

Published 2010



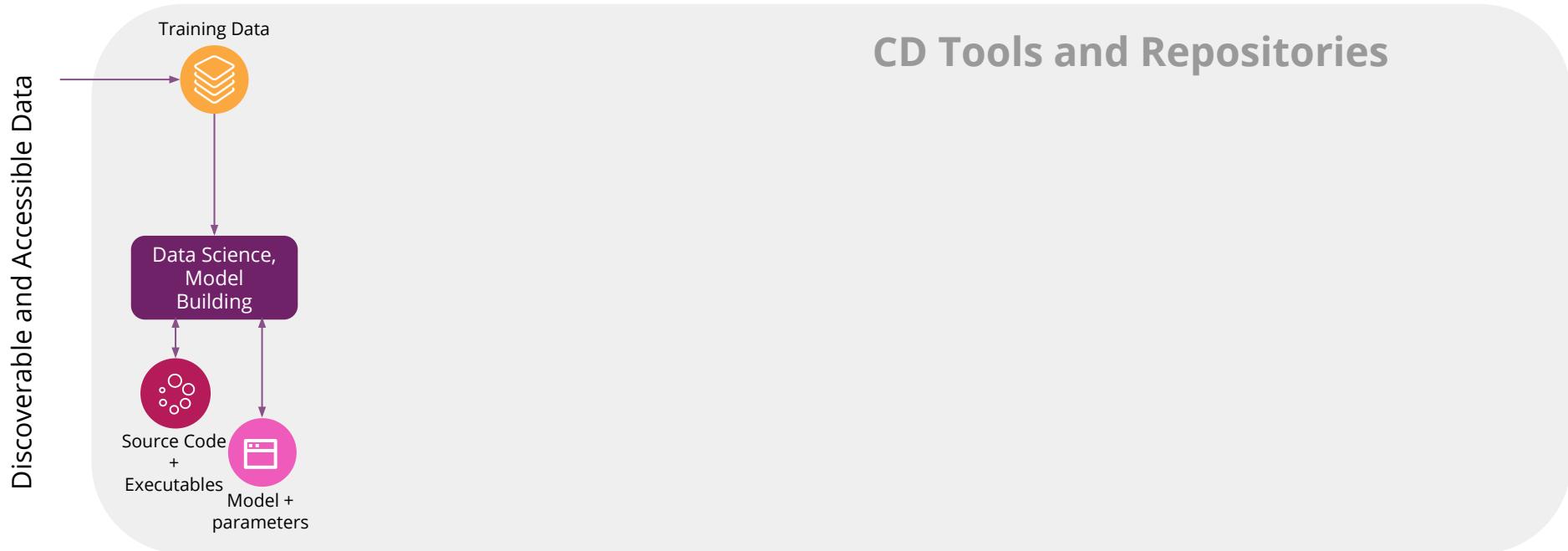
- Create a Repeatable, Reliable Process for Releasing Software
- Automate Almost Everything
- Build Quality In
- Work in Small Batches
- Keep Everything in Source Control
- Done Means "Released"
- Improve Continuously

*“Continuous Delivery is the ability to get **changes of all types** — including new features, configuration changes, bug fixes and experiments — into production, or into the hands of users, **safely and quickly in a sustainable way.**”*

- Jez Humble & Dave Farley

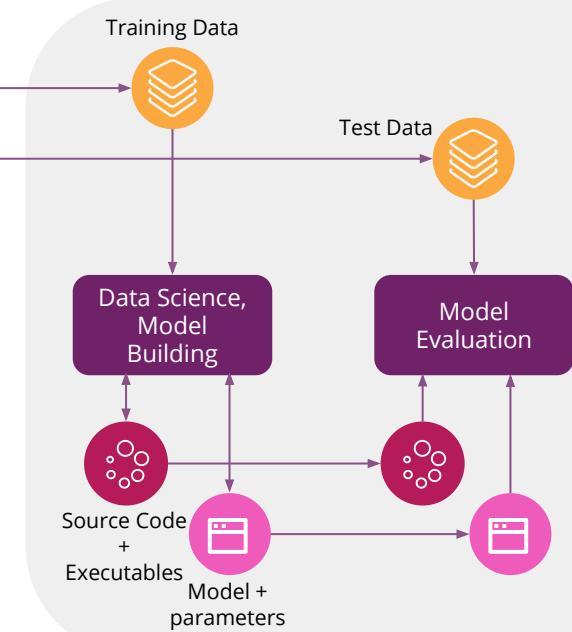
*Continuous Delivery for Machine Learning (CD4ML) is a software engineering approach in which a **cross-functional team** produces machine learning applications **based on code, data, and models** in **small and safe increments** that can be **reproduced** and **reliably released** at **any time**, in **short adaptation cycles**.*

# HOW DOES IT WORK



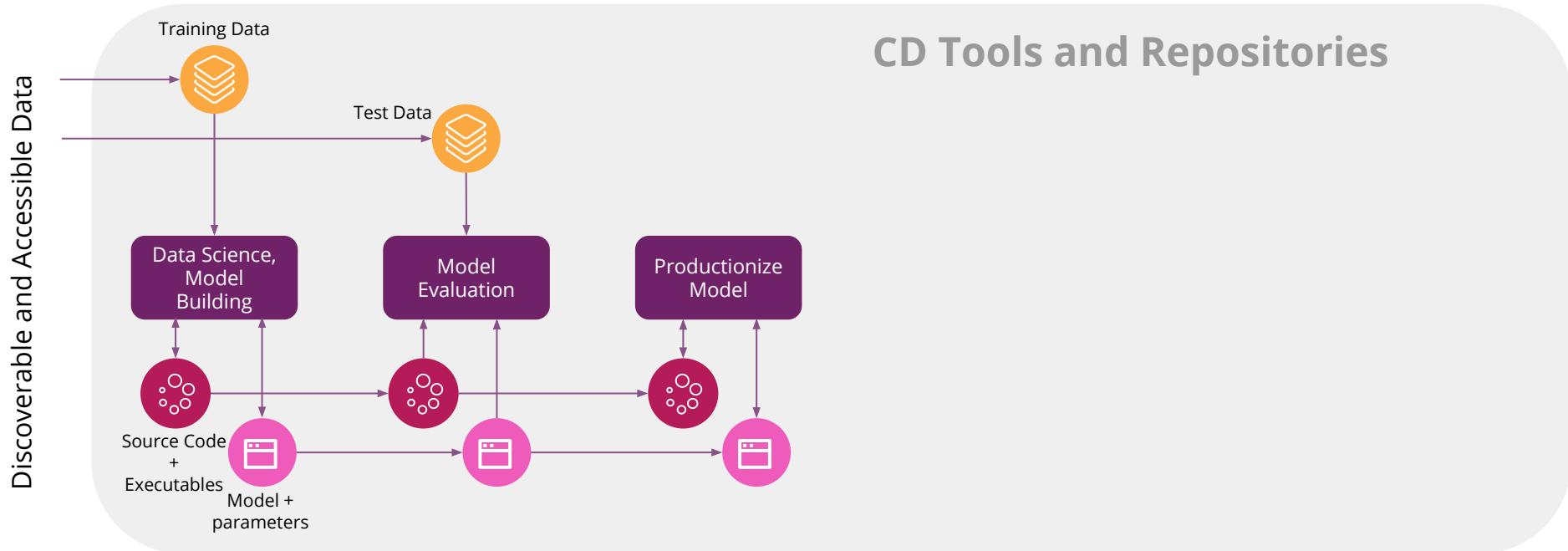
# HOW DOES IT WORK

Discoverable and Accessible Data

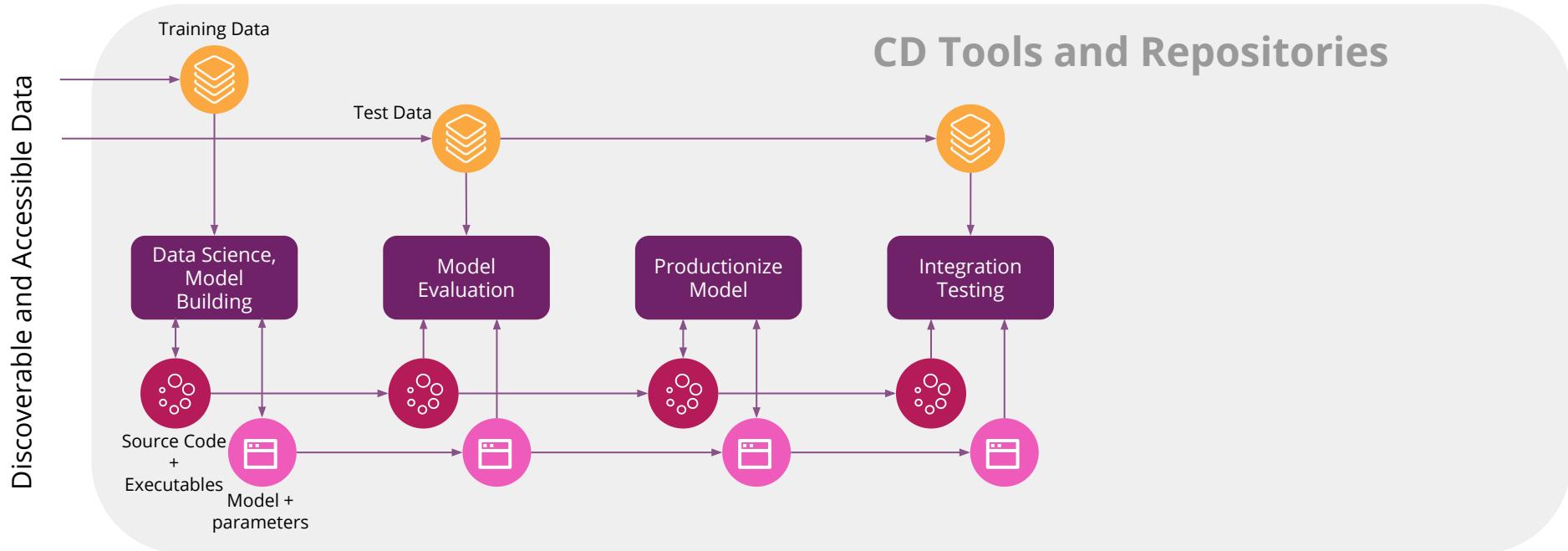


CD Tools and Repositories

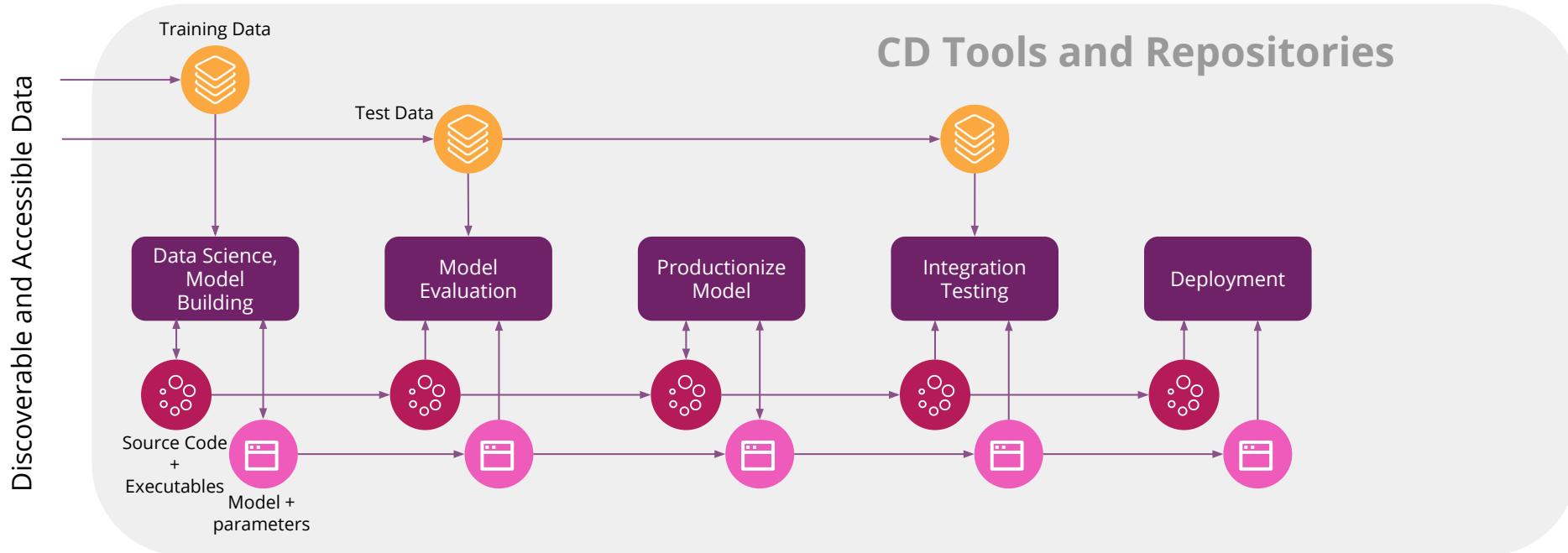
# HOW DOES IT WORK



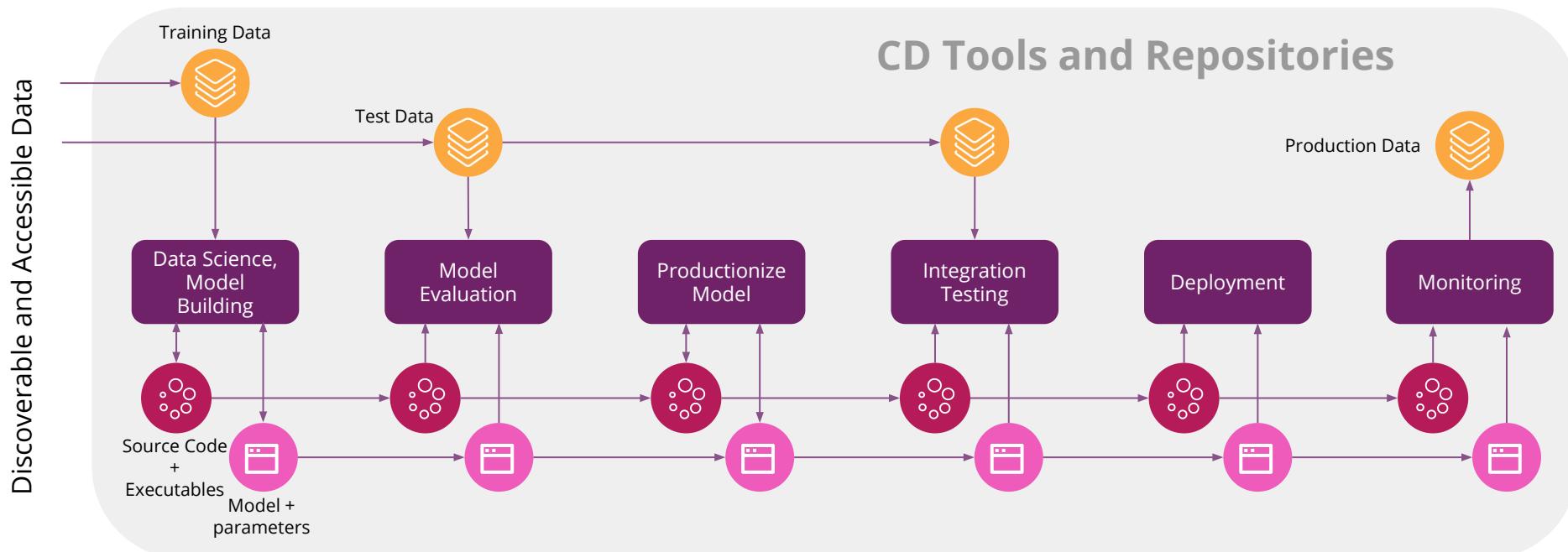
# HOW DOES IT WORK



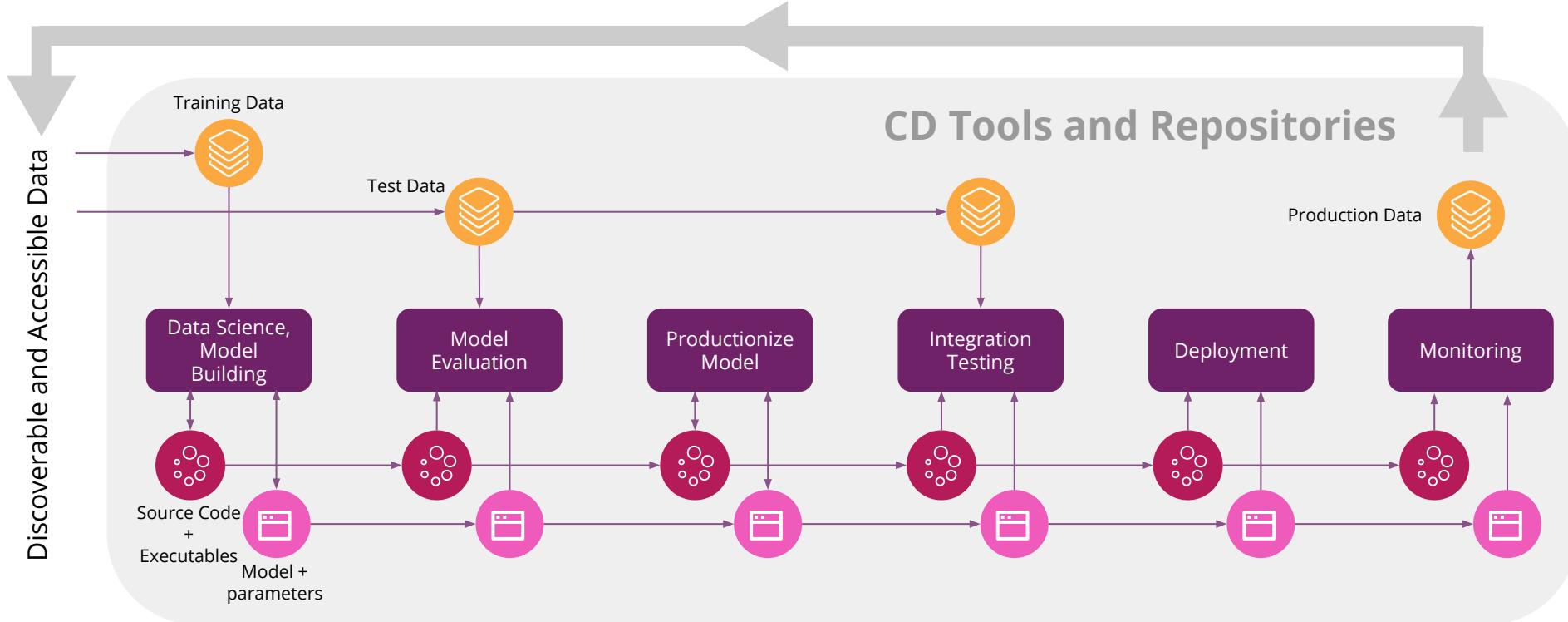
# HOW DOES IT WORK



# HOW DOES IT WORK



# HOW DOES IT WORK



# WHAT DO WE NEED IN OUR CD4ML STACK?



# THE BUSINESS PROBLEM: SALES FORECASTING FOR GROCERY RETAILER

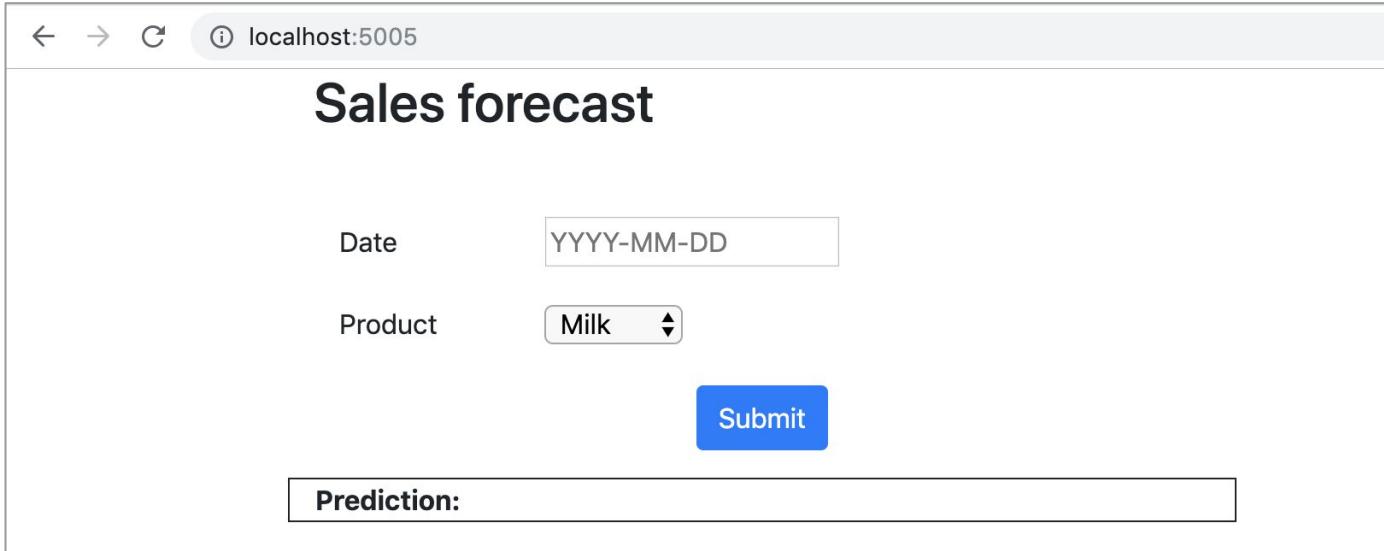
Make predictions based on data from:

- 4,000 items
- 50 stores
- 125,000,000 sales transactions
- 4.5 years of data

TASK:

Predict **how many** of each product will be purchased in **each store** on a **given date**

# WE WILL BUILD A SIMPLIFIED WEB APPLICATION



A screenshot of a web browser window displaying a simple sales forecast application. The URL bar shows 'localhost:5005'. The page title is 'Sales forecast'. The form has two input fields: 'Date' (with placeholder 'YYYY-MM-DD') and 'Product' (set to 'Milk', with a dropdown arrow). Below the form is a blue 'Submit' button. At the bottom is a long input field labeled 'Prediction:'.

← → ⌛ ⓘ localhost:5005

## Sales forecast

Date

Product  ▼

Submit

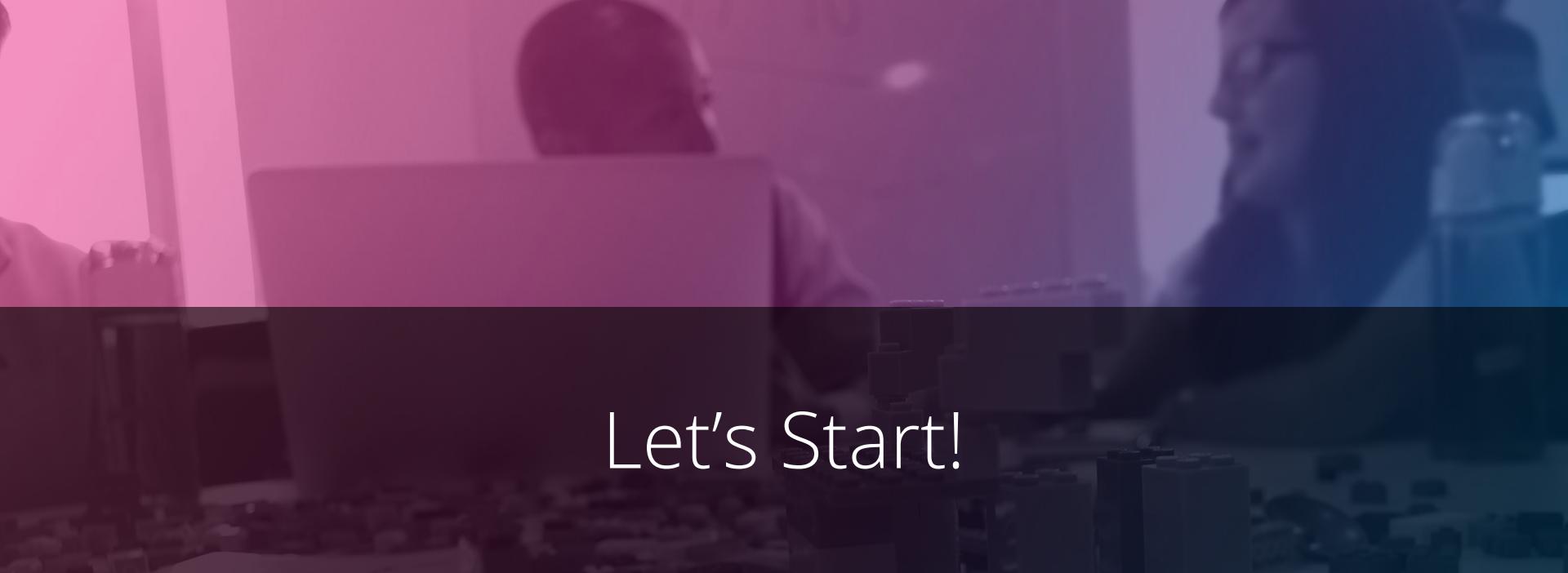
**Prediction:**

# STEP BY STEP

1. **Doing the plumbing:** Set up the pipeline and see if it is working
2. **Data Science:** Develop the model and test it (with Test-Driven Development - TDD)
3. **Machine Learning Engineering:** Improve the model in several steps and monitor the results
4. **Continuous Deployment:** Setup a performance test of the model, which only allows automatic deployment if the model passes the test
5. **Undo changes:** Roll back in time consistently with all artifacts
6. **Our app in the wild:** Monitor it in production with fluentd, elasticsearch and kibana



# Questions?



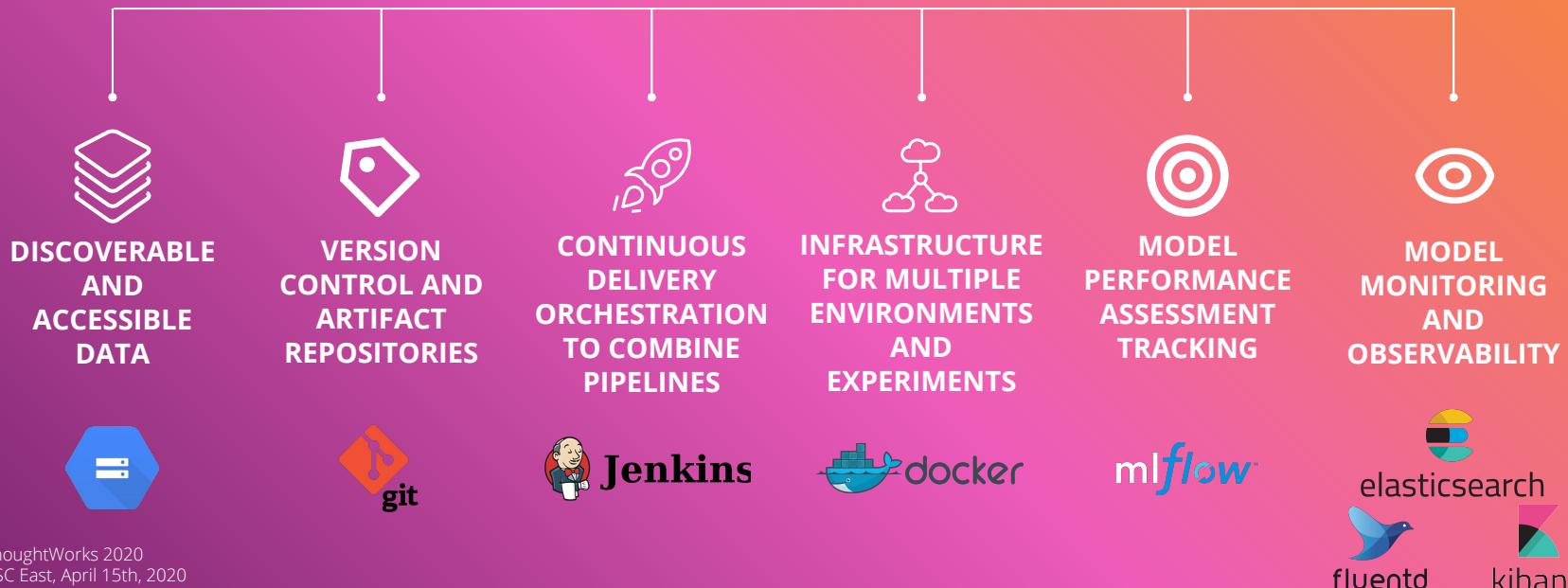
# Let's Start!

# STEP BY STEP

1. **Doing the plumbing:** Set up the pipeline and see if it is working (CW, EN)
2. **Data Science:** Develop the model and test it (with TDD) (DJ)
3. **Machine Learning Engineering:** Improve the model in several steps and monitor the results (DJ)
4. **Continuous Deployment:** Setup a performance test of the model, which only allows automatic deployment if the model passes the test (EN)
5. **Undo changes:** Roll back in time consistently with all artifacts (EN)
6. **Our app in the wild:** Monitor it in production with fluentd, elasticsearch and kibana (CW, EN)

# WHAT WE WILL USE IN THIS WORKSHOP

*There are many options for tools and technologies to implement CD4ML*



# LOG IN TO JENKINS



Welcome to Jenkins!

admin

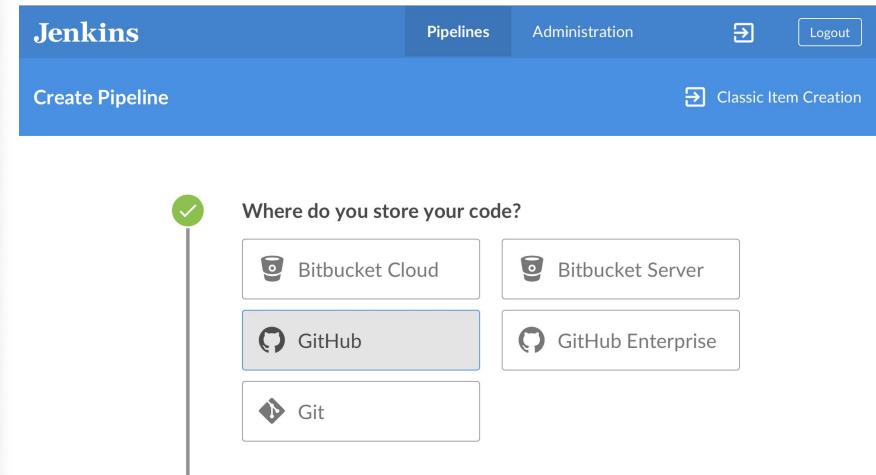
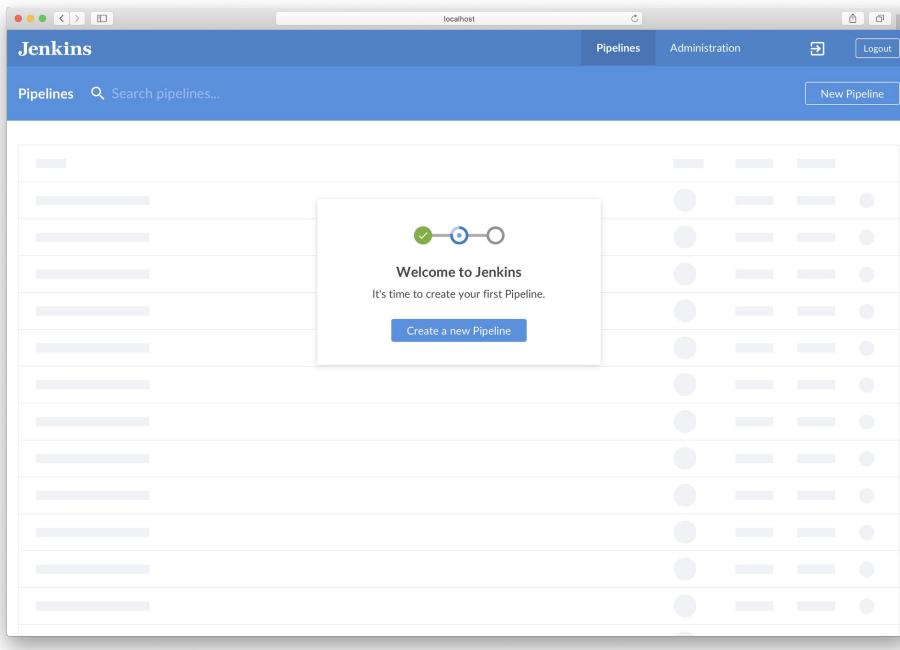
.....

Sign in

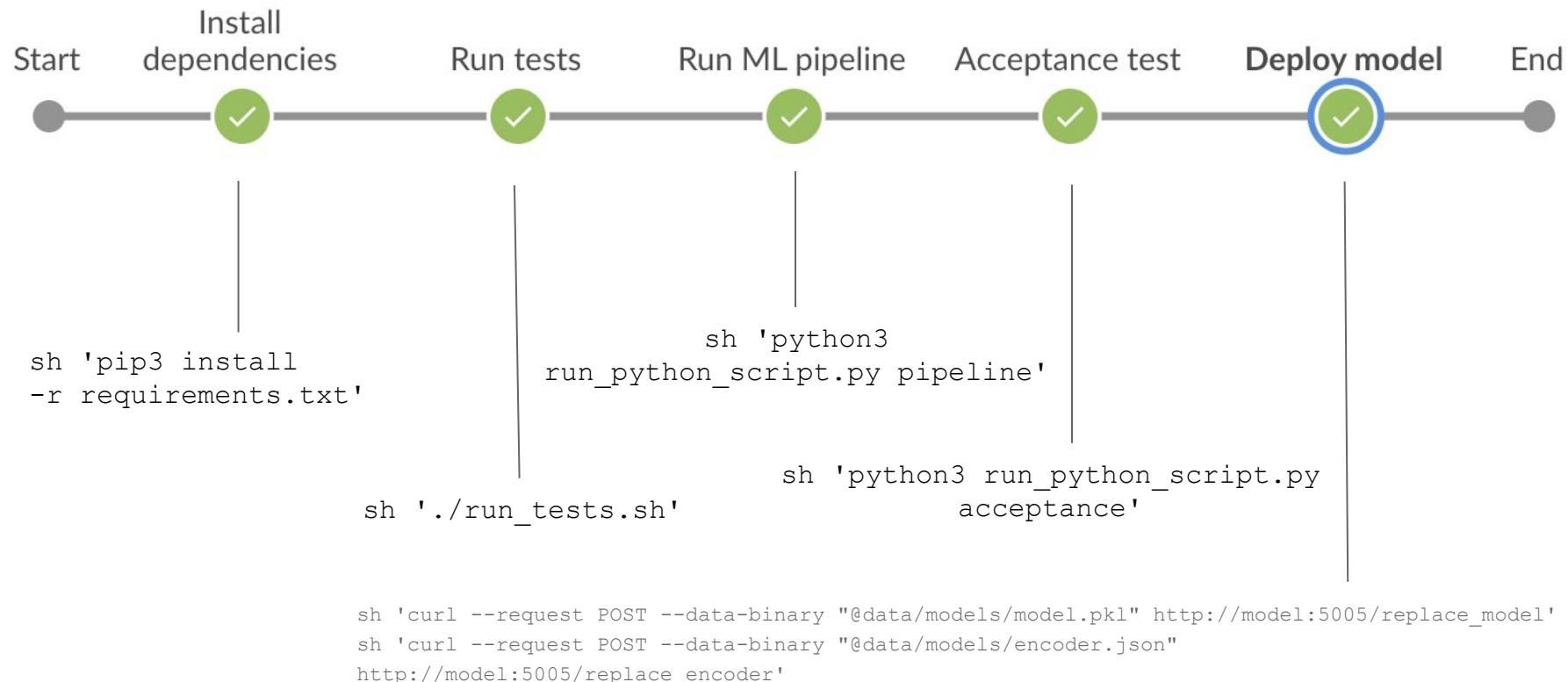
Keep me signed in

<http://localhost:10000/blue>

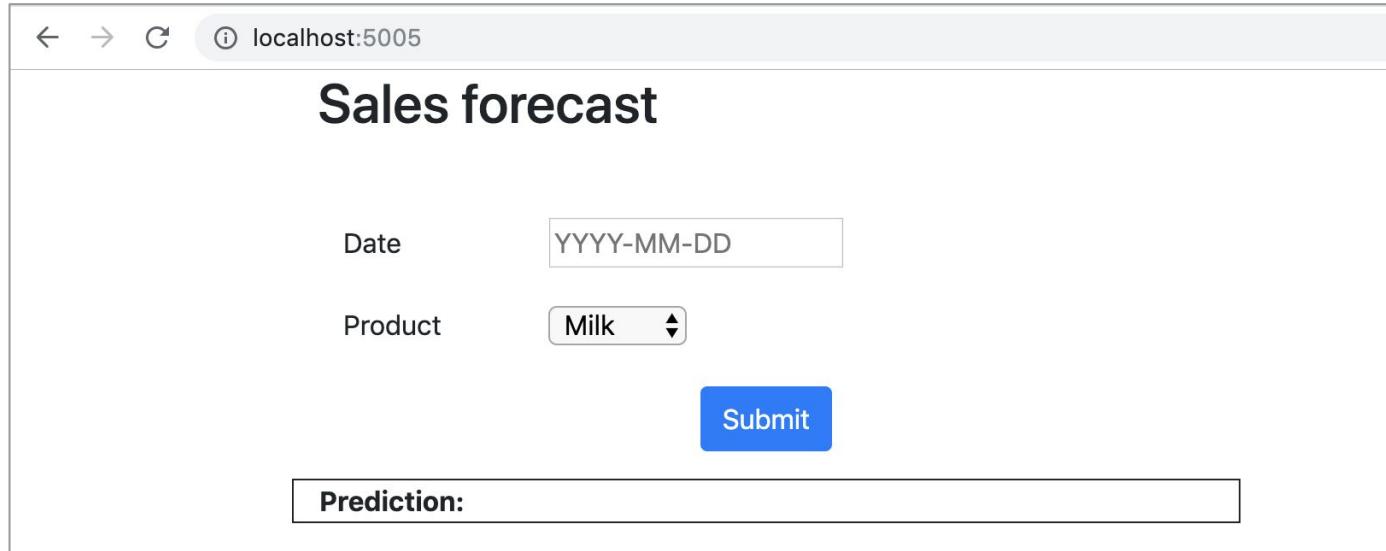
# SETUP THE PIPELINE



# OUR END GOAL JENKINS PIPELINE



# RUN THE WEB APPLICATION



A screenshot of a web browser window displaying a "Sales forecast" application. The URL bar shows "localhost:5005". The page title is "Sales forecast". There are two input fields: "Date" (with placeholder "YYYY-MM-DD") and "Product" (set to "Milk", with a dropdown arrow). A blue "Submit" button is below the inputs. At the bottom, there is a long, empty rectangular input field labeled "Prediction:".

← → C i localhost:5005

## Sales forecast

Date YYYY-MM-DD

Product Milk ▾

Submit

Prediction:

<http://localhost:11000/>

# Setup Jenkins

- Navigate to [instructions/2-SetupJenkins.md](#)

The screenshot shows a GitHub repository page for 'ThoughtWorksInc / CD4ML-Scenarios'. The repository has 7 watchers, 1 star, and 3 forks. The main content is the file '2-SetupJenkins.md', which contains the following text:

```
Setting up Jenkins

Goals



- Learn about Jenkins
- Setup and Configure a Deployment Pipeline to build and deploy your application to production
- Deploy to the Model server running in production

```

# STEP BY STEP

1. **Doing the plumbing:** Set up the pipeline and see if it is working (CW)
2. **Data Science:** Develop the model and test it (with TDD) (DJ)
3. **Machine Learning Engineering:** Improve the model in several steps and monitor the results (DJ)
4. **Continuous Deployment:** Setup a performance test of the model, which only allows automatic deployment if the model passes the test (EN)
5. **Undo changes:** Roll back in time consistently with all artefacts (EN)
6. **Our app in the wild:** Monitor it in production with fluentd, elasticsearch and kibana (CW)

# Guiding principles for ML pipelines

*Attention to good software development principles*

The key to quality, flexibility and maintainability is in controlling complexity

- Good design requires modularity and encapsulation
- Integrity of domain boundaries
- Maintainable software requires good code testing not just validation
- Automation, reproducibility including CI/CD

Using Popular ML Frameworks incorrectly can lead to hard-to-maintain software

Common major offenders

- Pandas and other data frame models. Even Spark.
- Notebooks used for pipeline coordination
- GGplot, other visualization libraries which include transformation and aggregations

# Example of abusing pandas

```
def prepare_sales_forecast(idx, sales_hist_dates, orders_all, sales_daily):
    sdate = sales_hist_dates.Date_min[idx]
    sales_default = pd.DataFrame(
        data={'Date': [sdate + timedelta(days=i) for i in range(0, int((last_date - sdate).days))]})
    sales_obs = sales_daily[(sales_daily.location == sales_hist_dates.location[idx]) & (
        sales_daily.factory_code == sales_hist_dates.factory_code[idx])]

    order_cust = orders_all[(orders_all.location == sales_hist_dates.location[idx]) & (
        orders_all.fact_code == sales_hist_dates.fact_code[idx]) & (orders_all.lodge_date < last_date)]
    order_endcust = order_cust[['Date', 'Completion', 'unitSales']]
    order_endcust = order_endcust[(order_endcust.VolUnitsSold > 0)]
    order_endcust_agg = order_endcust.groupby(['component'], as_index=False).agg(
        {'volume': ['mean', 'std', 'count']})
    order_endcust_agg.columns = ['component', 'mean', 'std', 'cnt']
    order_cust = order_endcust.merge(order_endcust_agg, how='left', on=['component'])

    return order_cust, sales_obs, sales_default
```

- Batch oriented
- Hard to make modular, hard to test
- Behavior of library functions hard to predict
- Hard to identify edge cases

# Example of clean modular code

```
def process(row_in):
    row = {'item_nbr': row_in['item_nbr'],
           'unit_sales': max(0.0, float(row_in['unit_sales'])),
           'date': row_in['date'],
           'year': row_in['year'],
           'month': row_in['month'],
           'day': row_in['day'],
           'class': row_in['class'],
           'family': row_in['family'],
           'perishable': int(row_in['perishable'] == '1'),
           'dayofweek': row_in['dayofweek'],
           'days_til_end_of_data': int(row_in['days_til_end_of_data']),
           'dayoff': int(row_in['dayoff'] == 'True')}

    return row
```

```
def test_process():
    row_in = {'id': '88219279',
              'date': '2016-08-16',
              'item_nbr': '103520',
              'unit_sales': '10.0',
              'family': 'GROCERY I',
              'class': '1028',
              'perishable': '0',
              'year': '2016',
              'month': '8',
              'day': '16',
              'dayofweek': '1',
              'days_til_end_of_data': '364',
              'dayoff': 'False'}

    row = process(row_in)
    expected = {'item_nbr': '103520',
                'unit_sales': '10.0',
                'date': '2016-08-16',
                'year': '2016',
                'month': '8',
                'day': '16',
                'class': '1028',
                'family': 'GROCERY I',
                'perishable': '0',
                'dayofweek': '1',
                'days_til_end_of_data': 364,
                'dayoff': 0}

    assert row == expected
```

# Example of a test

## Test Driven Development (TDD)

- Write the tests before the code
- Watch the test fail
- Write the code to make the test pass
- Write a new test
- Watch the test fail
- Write code to make all tests pass
- Repeat...

## Advantages

- All code functionality will be tested
- Debugging is easier
- Refactoring is easier

# Separate streaming versus batch

*With a preference for streaming*

Python makes streaming very easy with generators. E.g.

```
stream = csv.DictReader(open(filename, 'r'))  
print(next(stream))
```

```
{"id": "88221657", "date": "2016-08-16", "item_nbr": "1963838"}
```

Nice properties of streams

- They are generic. Encapsulates the ultimate source and its properties.
- Stream transformations are simple: stream\_2 = (transform(i) for i in stream)
- Encourages modular code and testing and separation of concerns
- Memory efficient
- Leads to higher productivity development / debugging. Depth-first.

# Depth first design

**Most common pattern is breadth-first design consisting of stages of batch processing**

Run Stage 1 on all rows

Run Stage 2 on all rows

....

Construct final outputs

Debugging is hard. Wait 10 minutes for it to crash in stage 3 and try to add print statements. Slow.

## Depth first design

Chain together transformation that can be run in streaming fashion. All stages first and then move to next row.

If it crashes on some line, just make that one line a test case and following TDD to fix it. Fast feedback

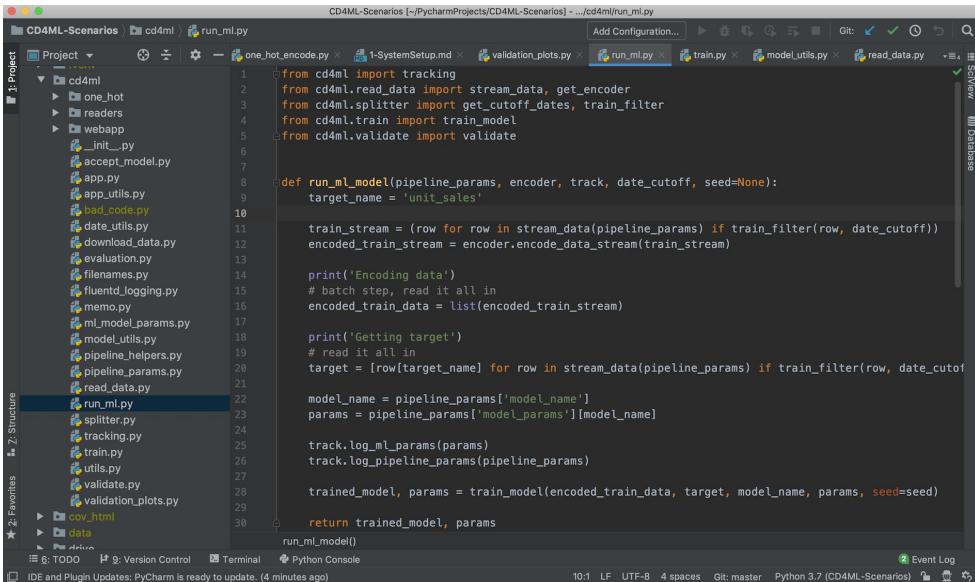
Don't  
mix

- Run all transformations as a single composite function to make new stream
- Apply filter function to create another stream
- Apply aggregation if needed to create another stream
- Collect the stream into a batch list only when needed. E.g. at Machine Learning Training
- More streams for example scoring and other processing

# STEP BY STEP

1. **Doing the plumbing:** Set up the pipeline and see if it is working (CW)
2. **Data Science:** Develop the model and test it (with TDD) (DJ)
3. **Machine Learning Engineering:** Improve the model in several steps and monitor the results (DJ)
4. **Continuous Deployment:** Setup a performance test of the model, which only allows automatic deployment if the model passes the test (EN)
5. **Undo changes:** Roll back in time consistently with all artefacts (EN)
6. **Our app in the wild:** Monitor it in production with fluentd, elasticsearch and kibana (CW)

# Overview of the Code Base and Scenarios



The screenshot shows the PyCharm IDE interface with the following details:

- Project:** CD4ML-Scenarios
- Run:** run\_ml.py
- Code Editor:** The current file is run\_ml.py, which contains Python code for a machine learning pipeline.
- Project Structure:** The project structure shows a directory named cd4ml containing various utility and model-related files like one\_hot\_encode.py, train.py, validate.py, etc.
- Toolbars and Status Bar:** The status bar at the bottom indicates the file is 10:1, LF, UTF-8, 4 spaces, Git: master, Python 3.7 (CD4ML-Scenarios), and shows the event log has 2 entries.

```
1 from cd4ml import tracking
2 from cd4ml.read_data import stream_data, get_encoder
3 from cd4ml.splitter import get_cutoff_dates, train_filter
4 from cd4ml.train import train_model
5 from cd4ml.validate import validate
6
7 def run_ml_model(pipeline_params, encoder, track, date_cutoff, seed=None):
8     target_name = 'unit_sales'
9
10    train_stream = (row for row in stream_data(pipeline_params) if train_filter(row, date_cutoff))
11    encoded_train_stream = encoder.encode_data_stream(train_stream)
12
13    print('Encoding data')
14    # batch step, read it all in
15    encoded_train_data = list(encoded_train_stream)
16
17    print('Getting target')
18    # read it all in
19    target = [row[target_name] for row in stream_data(pipeline_params) if train_filter(row, date_cutoff)]
20
21    model_name = pipeline_params['model_name']
22    params = pipeline_params['model_params'][model_name]
23
24    track.log_ml_params(params)
25    track.log_pipeline_params(pipeline_params)
26
27    trained_model, params = train_model(encoded_train_data, target, model_name, params, seed=seed)
28
29    return trained_model, params
30
run_ml_model()
```

- Structure of code base
- How to run the pipeline  
After making changes
- How to view results of  
models and keep track of  
changes

# STEP BY STEP

1. **Doing the plumbing:** Set up the pipeline and see if it is working (CW)
2. **Data Science:** Develop the model and test it (with TDD) (DJ)
3. **Machine Learning Engineering:** Improve the model in several steps and monitor the results (DJ)
4. **Continuous Deployment:** Setup a performance test of the model, which only allows automatic deployment if the model passes the test (EN)
5. **Undo changes:** Roll back in time consistently with all artefacts (EN)
6. **Our app in the wild:** Monitor it in production with fluentd, elasticsearch and kibana (CW)

# Continuous Deployment

- Navigate to [instructions/4-ContinuousDeployment.md](#)

The screenshot shows a GitHub repository page for **ThoughtWorksInc / CD4ML-Scenarios**. The repository has 7 pull requests, 1 star, and 3 forks. The main navigation bar includes links for Code, Issues (0), Pull requests (0), Actions, Projects (0), Wiki, Security, Insights, and Settings. The current file being viewed is **CD4ML-Scenarios / instructions / 4-CDCheck.md**. The commit history shows a single commit by **ericnagler** that cleaned up and simplified the instructions, pushed 1 hour ago. The file details show 22 lines (17 sloc) and 875 Bytes. The content of the file is as follows:

```
Continuous Deployment

Goals



- Learn about the principles of Continuous Deployment
- Learn about the steps in a Jenkinsfile
- Learn adding a new step into Jenkins

```

# Continuous Deployment

**“Continuous Deployment is the ability to get changes of all types — including new features, configuration changes, bug fixes and experiments — into production, or into the hands of users, safely and quickly in a sustainable way.”**

- In the earlier scenarios, we have iterated on our model multiple times. Let's add a step to our jenkins pipeline to add this continuous deployment check to ensure that our pipeline checks the model performance before deploying into our environments
- Uncomment the following block in your Jenkinsfile, commit and push your code

```
        }
    stage('Acceptance test') {
        steps {
            sh 'python3 run_python_script.py acceptance'
        }
    }
```

# Continuous Delivery

## How are we checking for performance (accept\_model.py)

```
def check_model_performance(metric_name, threshold_min, threshold_max):
    mlflow.set_tracking_uri(os.environ["MLFLOW_TRACKING_URL"])
    experiment = mlflow.get_experiment_by_name("local")
    runs = mlflow.search_runs(experiment_ids=experiment.experiment_id)
    last_run_record = get_latest_executed_run(runs) }
```

Connect to MLFlow and retrieve the last run of the model

```
metric_value = get_metric(metric_name, last_run_record)
run_name = last_run_record["tags.mlflow.runName"].head().values[0]
template = "Metric: {metric_name} for Run: {run_name} was not accepted, "
    "value: {metric_value}, "
    "threshold_min: {threshold_min}, threshold_max: {threshold_max}"
message = template.format(run_name=run_name,
                           metric_name=metric_name,
                           metric_value=metric_value,
                           threshold_min=threshold_min,
                           threshold_max=threshold_max)
```

```
assert threshold_min <= metric_value <= threshold_max, message }
```

Retrieve the r2 score of the model

```
assert threshold_min <= metric_value <= threshold_max, message }
```

Ensure the model score is acceptable for deployment

# STEP BY STEP

1. **Doing the plumbing:** Set up the pipeline and see if it is working (CW)
2. **Data Science:** Develop the model and test it (with TDD) (DJ)
3. **Machine Learning Engineering:** Improve the model in several steps and monitor the results (DJ)
4. **Continuous Deployment:** Setup a performance test of the model, which only allows automatic deployment if the model passes the test (EN)
5. **Undo changes:** Roll back code in time consistently with all artifacts (EN)
6. **Our app in the wild:** Monitor it in production with fluentd, elasticsearch and kibana (CW)

# Undo Changes for Performance Regressions

- Navigate to [instructions/5-UndoChanges.md](#)

The screenshot shows a GitHub repository page for 'ThoughtWorksInc / CD4ML-Scenarios'. The repository has 7 pull requests, 1 star, and 3 forks. The 'Code' tab is selected, showing the file '5-UndoChanges.md'. The file was last updated by 'ericnagler' 15 seconds ago, with a commit message 'Adjust file name'. It has 1 contributor. The file contains 30 lines (22 sloc) and 873 Bytes. The content of the file is displayed below:

```
Continuous Deployment - Undo Changes

Goals



- Observe the continuous deployment pipeline acceptance check
- Learn how to revert your code to a previous commit

```

# Undo Changes for Performance Regressions

- Verify that model adjustments that negatively impact model performance don't get released to production.
- We are going to purposely regress the model and watch our pipeline fail, then fix it!
- Change the model used to use less random forest trees. Change from 50 estimators to 5

X CD4ML-Scenarios < 13 >

Branch: master 59s 1 changes  
Commit: a24bb05 21 hours ago Started by an SCM change

Pipeline Changes Tests Artifacts ⚡ 🖊⚙️➡️ Logout X



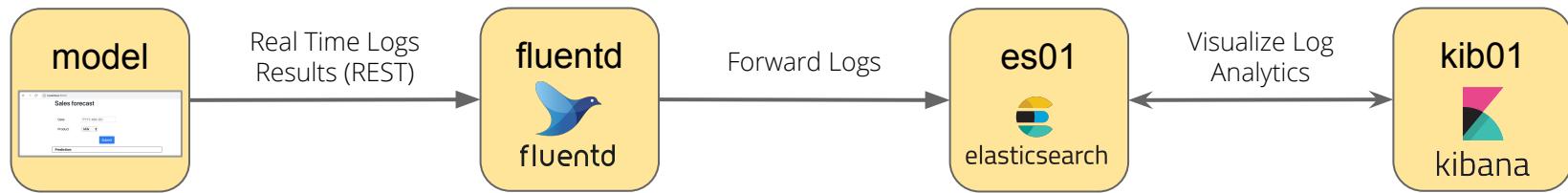
# Undo Changes for Performance Regressions

- But, because we performed our development in small incremental steps we can easily revert our changes back.
- `git revert HEAD`
- `git push`

# STEP BY STEP

1. **Doing the plumbing:** Set up the pipeline and see if it is working (CW)
2. **Data Science:** Develop the model and test it (with TDD) (DJ)
3. **Machine Learning Engineering:** Improve the model in several steps and monitor the results (DJ)
4. **Continuous Deployment:** Setup a performance test of the model, which only allows automatic deployment if the model passes the test (EN)
5. **Undo changes:** Roll back in time consistently with all artefacts (EN)
6. **Our app in the wild:** Monitor it in production with fluentd, elasticsearch and kibana (CW)

# Technical Details of the setup of fluentd, elasticsearch and kibana (EN)



- **FluentD** is a open source data collector that allows for unified logging. FluentD allows for collection and forwarding of logs from any application into whatever service you like. In this case, we collect the logs from the model and forward them to...
- **ElasticSearch** is a distributed JSON, search and analytics engine allowing rapid search and log ingest. This allows us to index our models for monitoring and analytics. These records can be visualized in...
- **Kibana** is a visualization tool for elasticsearch allowing for data discovery, exploration, visualization and dashboarding.
- **These tools enable you to understand:**
  - What your customers are searching for and asking your model for
  - What your model is inferencing so you can track it's performance

# SETUP fluentd, elasticsearch and kibana

- Navigate to [instructions/6-KibanaLogVisualization.md](#) page on github to setup fluentd, elasticsearch and kibana

The screenshot shows a GitHub repository page for 'ThoughtWorksInc / CD4ML-Scenarios'. The repository has 7 pull requests, 1 star, and 3 forks. The 'Code' tab is selected, showing the file '6-KibanaLogVisualization.md'. The file was last updated by 'ericnagler' 23 seconds ago, with a commit message 'Adjust exercise number'. It has 1 contributor. The file contains 45 lines (27 sloc) and is 2.34 KB. Below the code, there is a section titled 'Exercise 6: Model Monitoring and Observability' with a 'Goals' section containing the following bullet points:

- Learn about EFK Stack ([Elasticsearch](#), [FluentD](#), and [Kibana](#))
- Configure and deploy our application to log prediction events to Elastic Search
- Visualize events on Kibana dashboard
- Learn how to close the data feedback loop

# See prediction of your model in kibana



# Exercise

The screenshot shows the Kibana Discover interface with the following highlights:

- Search predictions**: Points to the search bar at the top.
- Filter predictions**: Points to the "model-\*" filter dropdown and its search input.
- Select fields**: Points to the "Available fields" section, specifically the "Popular" list which includes @timestamp, \_type, date\_string, item\_name, prediction, \_id, \_index, and score.
- Inspect table and json file**: Points to the table view showing two log entries with detailed JSON data.

**Discover View**

Search predictions

Filter predictions

Select fields

Inspect table and json file

12 hits

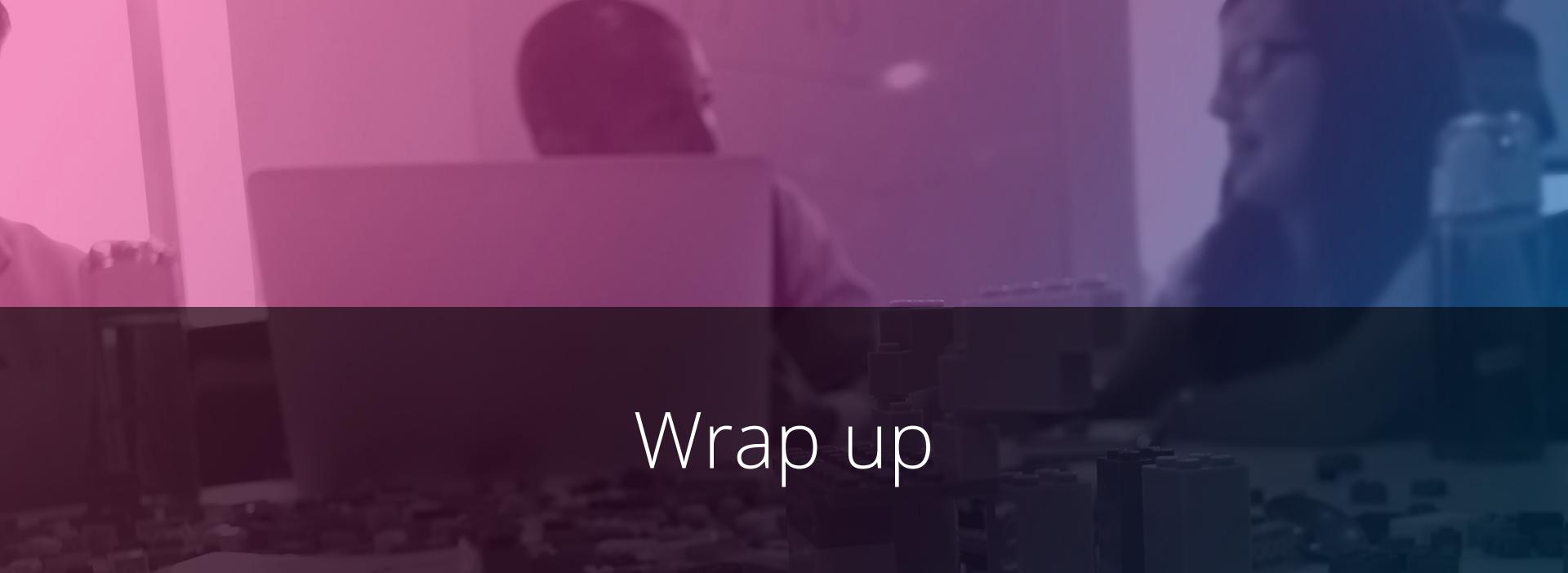
Apr 5, 2020 @ 00:00:00.000 - Apr 11, 2020 @ 23:59:59.999 — Auto

Count

Time

\_source

Time	_source
Apr 8, 2020 @ 12:34:49.000	<pre>prediction: 1.3 itemid: 99197 item_name: Milk date_string: Apr 10, 2020 @ 00:00:00.000 @timestamp: Apr 8, 2020 @ 12:34:49.000 _id: VE7JWXEBtRnhFvZrStR3 _type: _doc _index: model-2020.04.08 _score: -</pre>
Apr 8, 2020 @ 12:34:59.000	<pre>prediction: 2.06 itemid: 99197 item_name: Milk date_string: Apr 11, 2020 @ 00:00:00.000 @timestamp: Apr 8, 2020 @ 12:34:59.000 _id: VU7JWXEBtRnhFvZrbtQ8 _type: _doc _index: model-2020.04.08 _score: -</pre>



# Wrap up

# WHAT DID YOU LEARN?

- The Principles of CD4ML
- The components of a tech stack you need to run CD4ML
- Using Test Driven Development in a ML context
- Data Science best practice in the industry
- Run CD4ML on your laptop!

# Links

- **Github**

<https://github.com/ThoughtWorksInc/CD4ML-Scenarios>

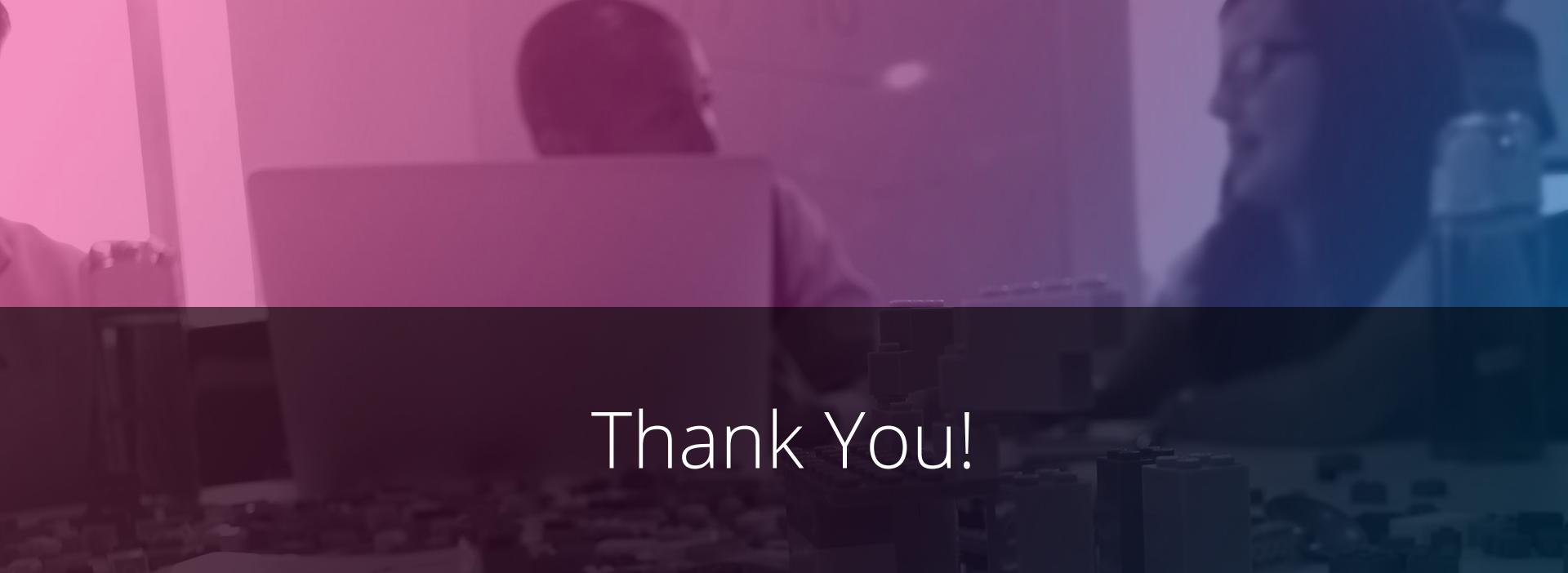
- **Articles**

<https://martinfowler.com/articles/cd4ml.html>

<https://www.thoughtworks.com/insights/articles/intelligent-enterprise-series-cd4ml>



# Questions?



# Thank You!