

Continuous Delivery for Machine Learning

Open Data Science Conference Europe
September 17, 2020

Christoph Windheuser - cwindheu@thoughtworks.com

David Johnston - dajohnst@thoughtworks.com

Eric Nagler - eric.nagler@thoughtworks.com

ThoughtWorks®

YOUR WORKSHOP TEAM



Christoph Windheuser

**Global Head of Artificial
Intelligence**

cwindheu@thoughtworks.com



David Johnston

Principal Data Scientist

dajohnst@thoughtworks.com



Eric Nagler

Lead Data Engineer

eric.nagler@thoughtworks.com

www.thoughtworks.com

AGENDA

Introduction and Setup.....	10 Min
Discussion of Continuous Delivery for Machine Learning (CD4ML).....	45 Min
Break.....	10 Min
Workshop.....	1h 45 Min
Doing the plumbing: Set up the pipeline and see if it is working.....	20 Min
Data Science: Develop the model and test it (with TDD).....	50 Min
Continuous Deployment: Bring the model into production.....	20 Min
Monitor the model “in the wild”.....	15 Min
Questions and Answers.....	10 Min



ThoughtWorks®

7000 technologists with 40 offices in 14 countries



Partner for technology driven business transformation



100+
books written

#1
in Agile and
Continuous Delivery

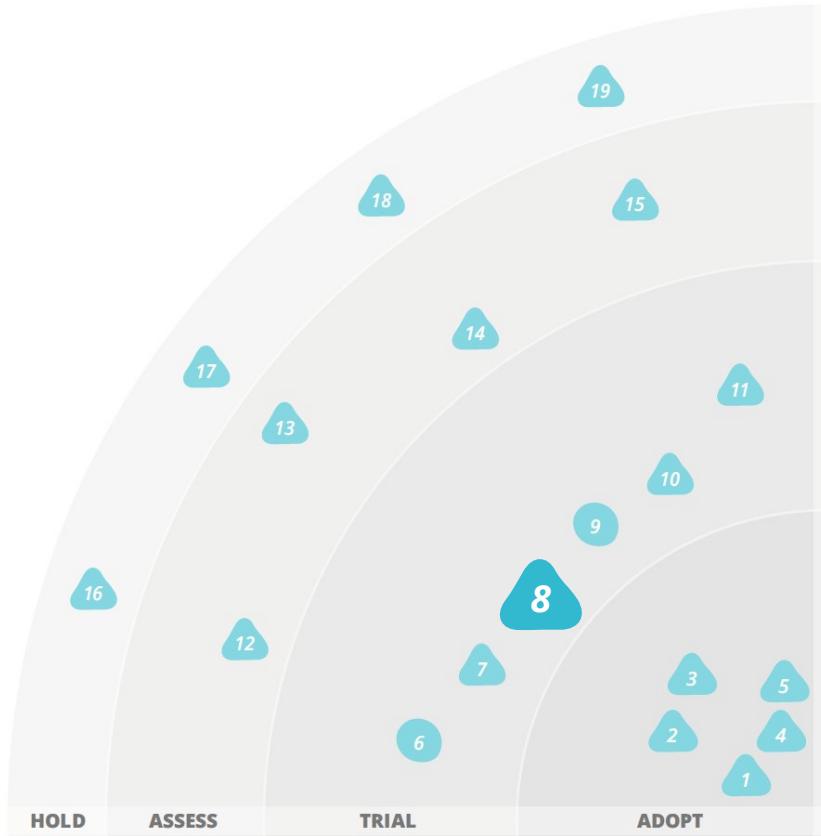




TECHNIQUES

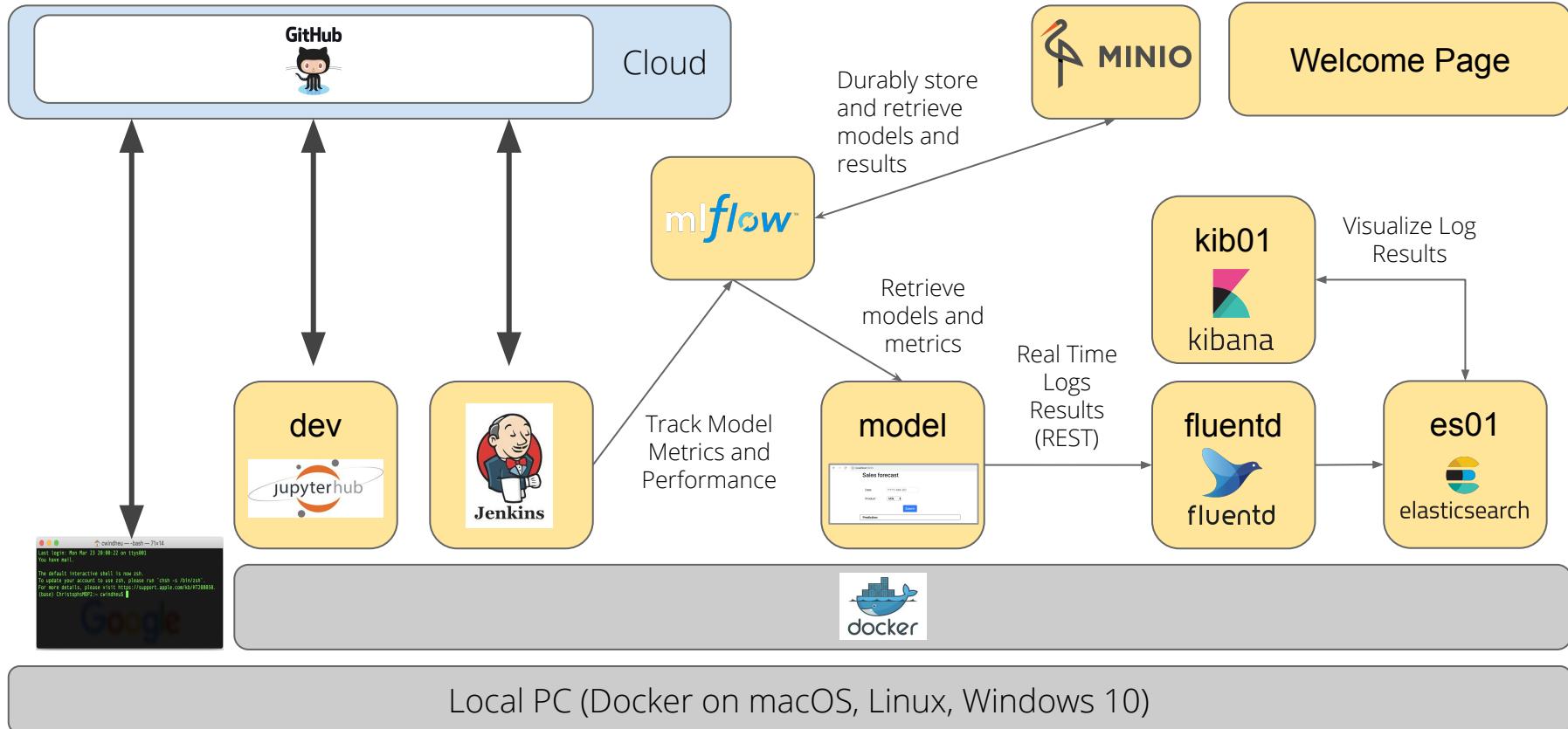
Continuous delivery #8
for machine
learning (CD4ML)
models

TRIAL



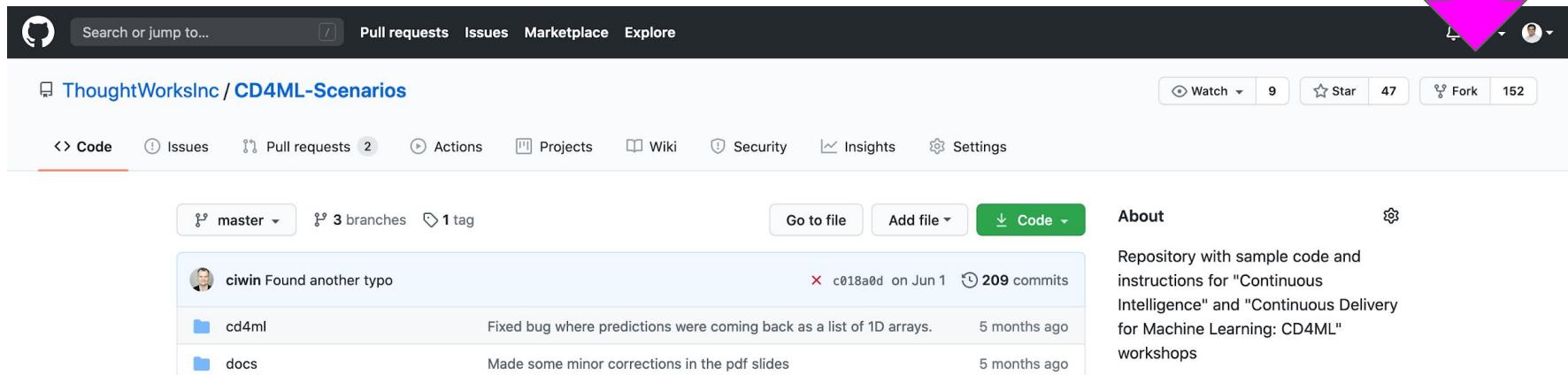
Setting up your local environment

Big Picture of Workshop Services



Installation (1)

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



A screenshot of a GitHub repository page for "ThoughtWorksInc / CD4ML-Scenarios". The page has a dark header with navigation links: Search or jump to..., Pull requests, Issues, Marketplace, Explore. On the right of the header is a user profile icon. Below the header is a light-colored navigation bar with links: Code, Issues, Pull requests (2), Actions, Projects, Wiki, Security, Insights, Settings. To the right of the navigation bar are buttons for Watch (9), Star (47), Fork (152), and a gear icon. The main content area shows a dropdown menu for the master branch, 3 branches, and 1 tag. A list of recent commits is displayed:

- ciwin Found another typo (c018a0d, Jun 1, 209 commits)
- cd4ml Fixed bug where predictions were coming back as a list of 1D arrays. (5 months ago)
- docs Made some minor corrections in the pdf slides. (5 months ago)

To the right of the commit list is an "About" section with the following text:

Repository with sample code and instructions for "Continuous Intelligence" and "Continuous Delivery for Machine Learning: CD4ML" workshops

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. Do not

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's profile icon is in the top right corner. The main area shows the "Developer settings" section with tabs for GitHub Apps, OAuth Apps, and Personal access tokens. The "Personal access tokens" tab is selected. A title "New personal access token" is displayed, followed by a note explaining that personal access tokens function like ordinary OAuth access tokens. A "Note" field contains the text "ODSC-Jenkins". A "What's this token for?" question is present. Below, a "Select scopes" section allows the user to choose access levels. The "repo" scope is checked, granting full control of private repositories. Other checked scopes include repo:status, repo_deployment, public_repo, and repo:invite. The "user" scope is also listed with options for read:user, user:email, and user:follow.

Search or jump to... / Pull requests Issues Marketplace Explore

Settings / Developer settings

GitHub Apps

OAuth Apps

Personal access tokens

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 9 pull requests, 47 stars, 152 forks, and 152 issues. The 'Code' tab is selected, showing the file '1-SystemSetup.md'. The file was last updated by dave31415 on April 14, 2020. It contains 93 lines (70 sloc) and is 4.39 KB. The content of the file includes the heading 'Setting up your environment'.

ThoughtWorksInc / CD4ML-Scenarios

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master CD4ML-Scenarios / instructions / 1-SystemSetup.md Go to file ...

dave31415 minor docs Latest commit 9037273 on Apr 14 History

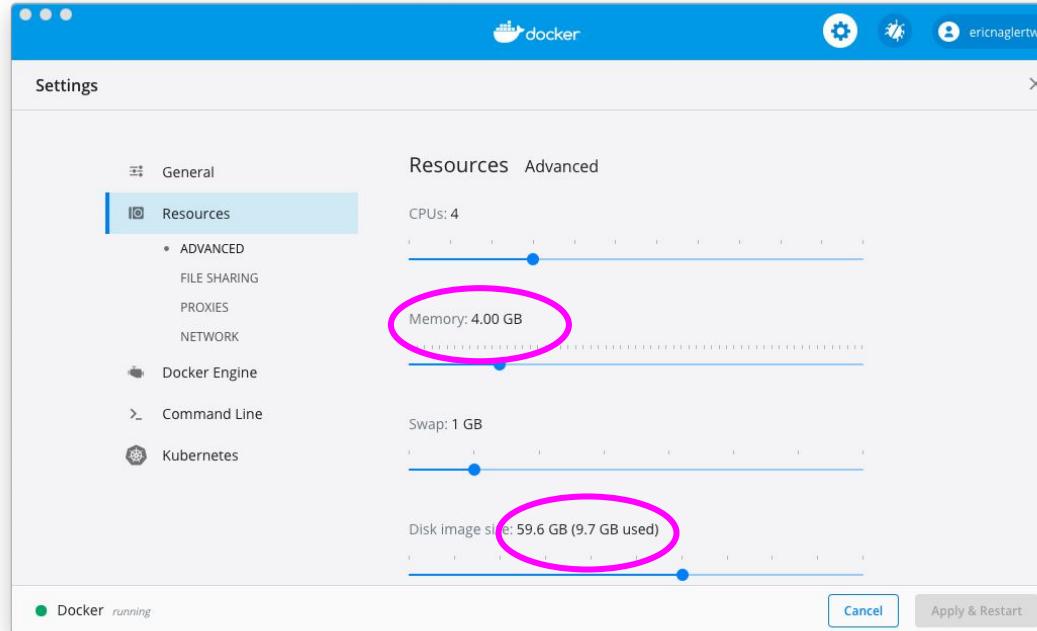
3 contributors

93 lines (70 sloc) | 4.39 KB Raw Blame

Setting up your environment

Installation (4)

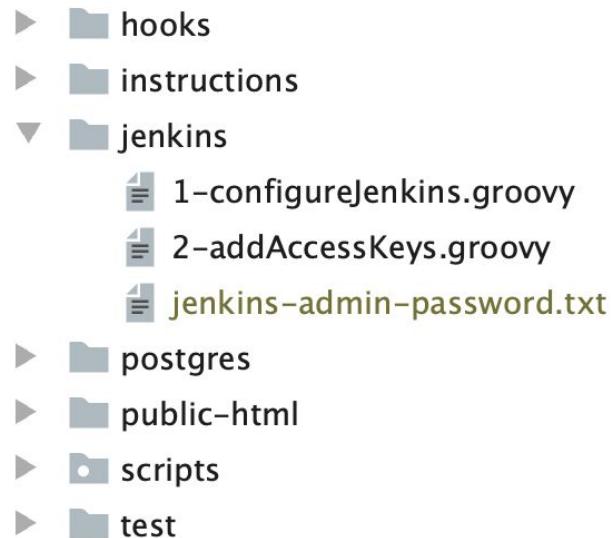
- Increase the RAM allocated to docker to 4 Gigabytes (or more)
- Ensure you have about 20 GB Free on your Docker image



Installation (5)

Configuring your jenkins password

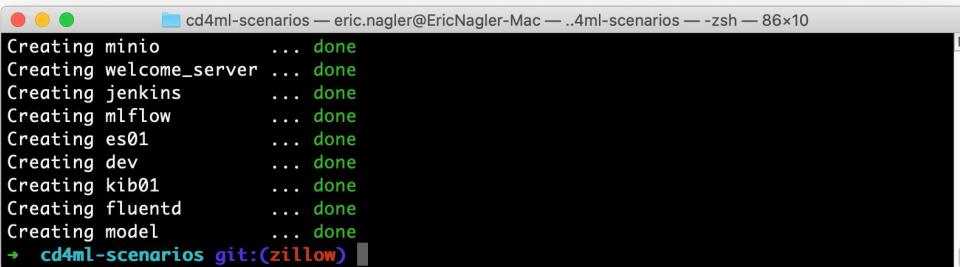
- Create a file in jenkins/jenkins-admin-password.txt
- Open the file and put in a secure password on the first line. Save and close the file.
- When you log into Jenkins your username will be 'admin'



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
```



The screenshot shows a terminal window on a Mac OS X desktop. The title bar reads "cd4ml-scenarios — eric.nagler@EricNagler-Mac — ..4ml-scenarios — zsh — 86x10". The terminal content displays the output of a "docker-compose up -d --build" command. It lists the creation of several services: minio, welcome_server, jenkins, mlflow, es01, dev, kib01, fluentd, and model. Each service is shown with a status of "... done". At the bottom of the terminal, the prompt "cd4ml-scenarios git:(zillow)" is visible.

```
Creating minio      ... done
Creating welcome_server ... done
Creating jenkins     ... done
Creating mlflow       ... done
Creating es01         ... done
Creating dev           ... done
Creating kib01        ... done
Creating fluentd      ... done
Creating model        ... done
+ cd4ml-scenarios git:(zillow)
```

Homepage

- Navigate to homepage - <http://localhost:3000>

CD4ML Go to Github

CONTINUOUS DELIVERY FOR MACHINE LEARNING

Select where you would like to go!

The diagram illustrates the six core components of CD4ML:

- DISCOVERABLE AND ACCESSIBLE DATA (Icon: Stacked layers)
- VERSION CONTROL AND ARTIFACT REPOSITORIES (Icon: Hexagon with a keyhole)
- CONTINUOUS DELIVERY ORCHESTRATION TO COMBINE PIPELINES (Icon: Rocket)
- INFRASTRUCTURE FOR MULTIPLE ENVIRONMENTS AND EXPERIMENTS (Icon: Cloud with a rocket)
- MODEL PERFORMANCE ASSESSMENT TRACKING (Icon: Target)
- MODEL MONITORING AND OBSERVABILITY (Icon: Eye)

Below each component, there are logos for specific tools:

- DISCOVERABLE AND ACCESSIBLE DATA: Apache Beam
- VERSION CONTROL AND ARTIFACT REPOSITORIES: Git
- CONTINUOUS DELIVERY ORCHESTRATION TO COMBINE PIPELINES: Jenkins
- INFRASTRUCTURE FOR MULTIPLE ENVIRONMENTS AND EXPERIMENTS: Docker
- MODEL PERFORMANCE ASSESSMENT TRACKING: mlflow
- MODEL MONITORING AND OBSERVABILITY: fluentd, elasticsearch, kibana

Additional Locations

- Machine Learning Model
- Jupyter Lab Environment
- Minio Object Storage

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

- Schema
- Sampling over Time
- Volume
- ...

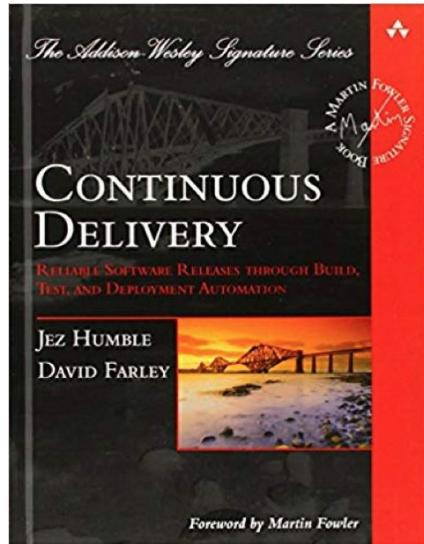
Model

- Research, Experiments
- Training on New Data
- Performance
- ...

Code

- New Features
- Bug Fixes
- Dependencies
- ...

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

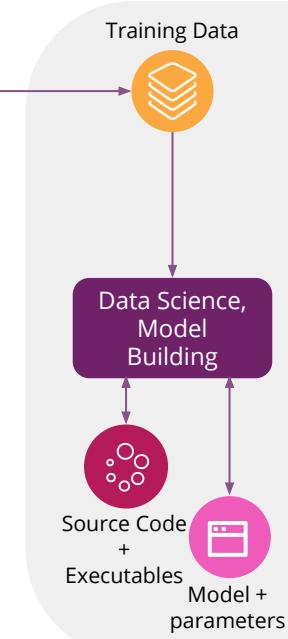
*“Continuous Delivery is the ability to get **changes of all types** — including new features, configuration changes, bug fixes and experiments — into production, or into the hands of users, **safely and quickly** in a **sustainable way**.”*

- 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

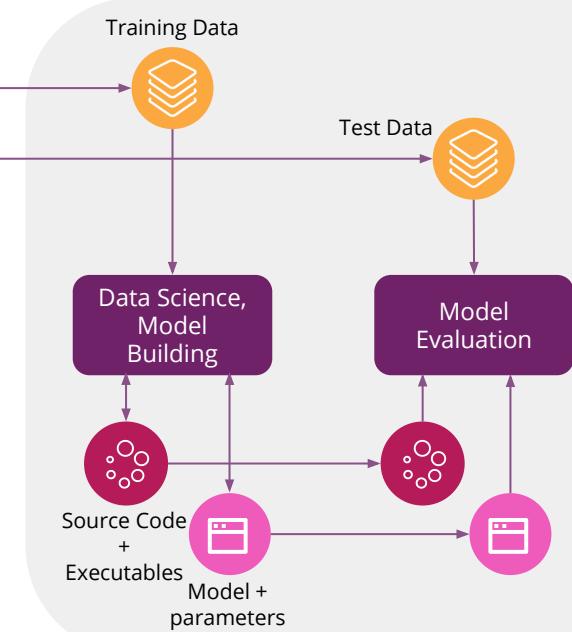
Discoverable and Accessible Data



CD Tools and Repositories

HOW DOES IT WORK

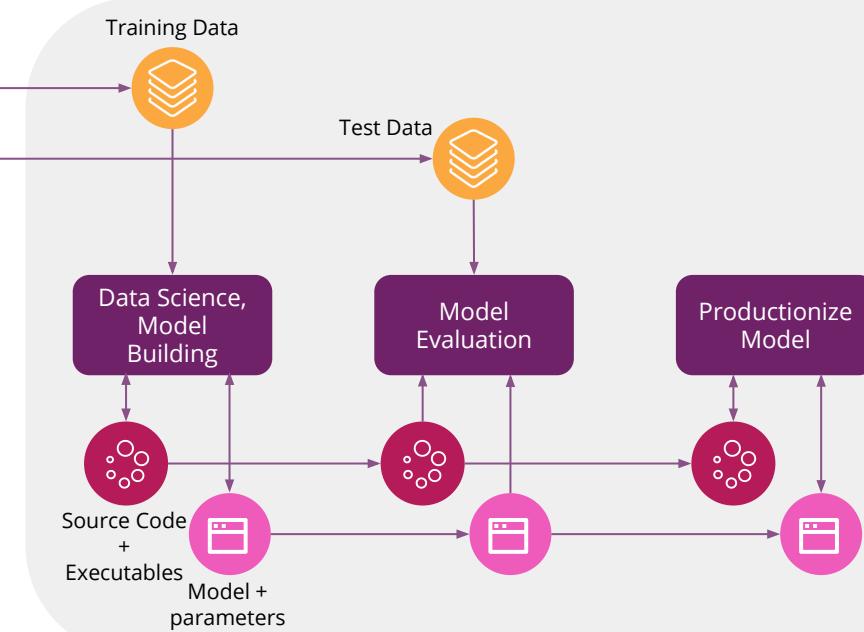
Discoverable and Accessible Data



CD Tools and Repositories

HOW DOES IT WORK

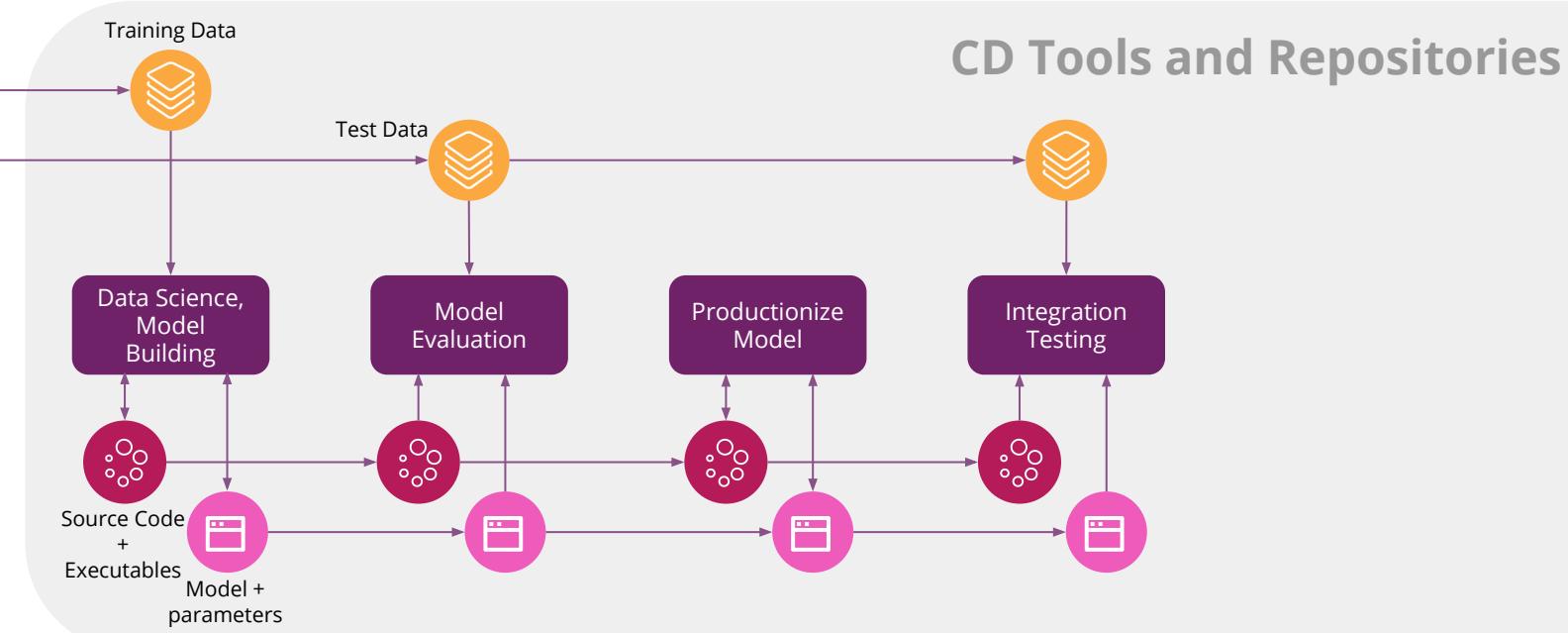
Discoverable and Accessible Data



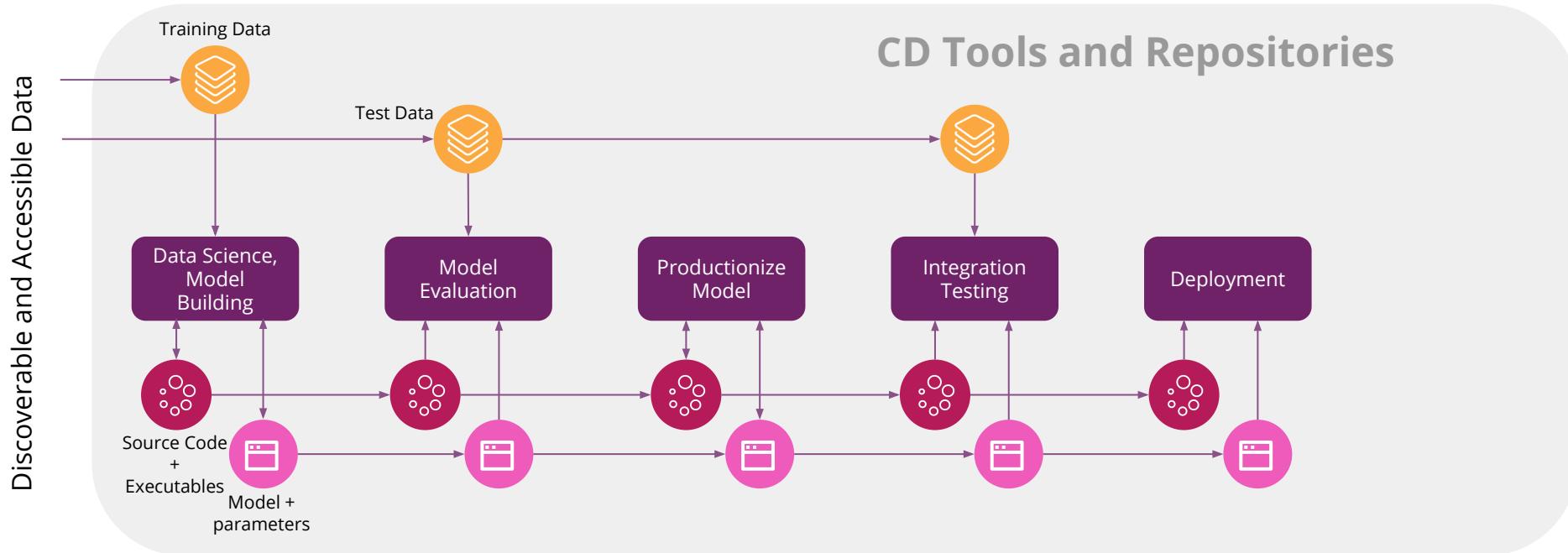
CD Tools and Repositories

HOW DOES IT WORK

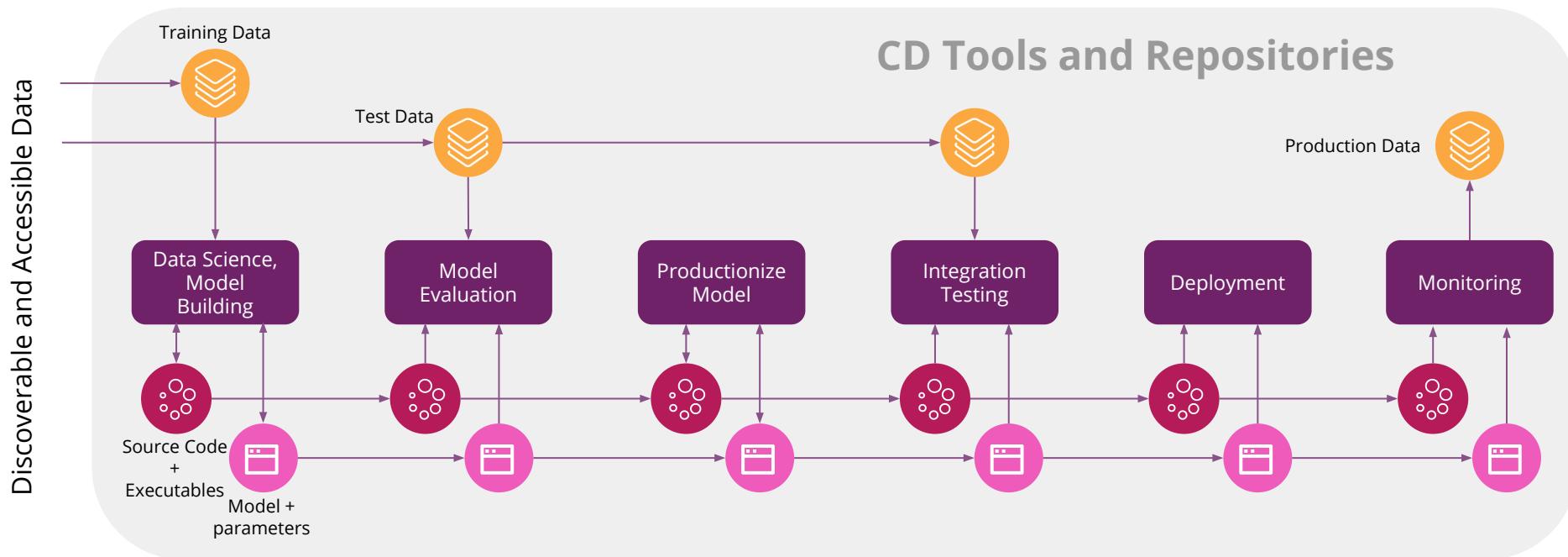
Discoverable and Accessible Data



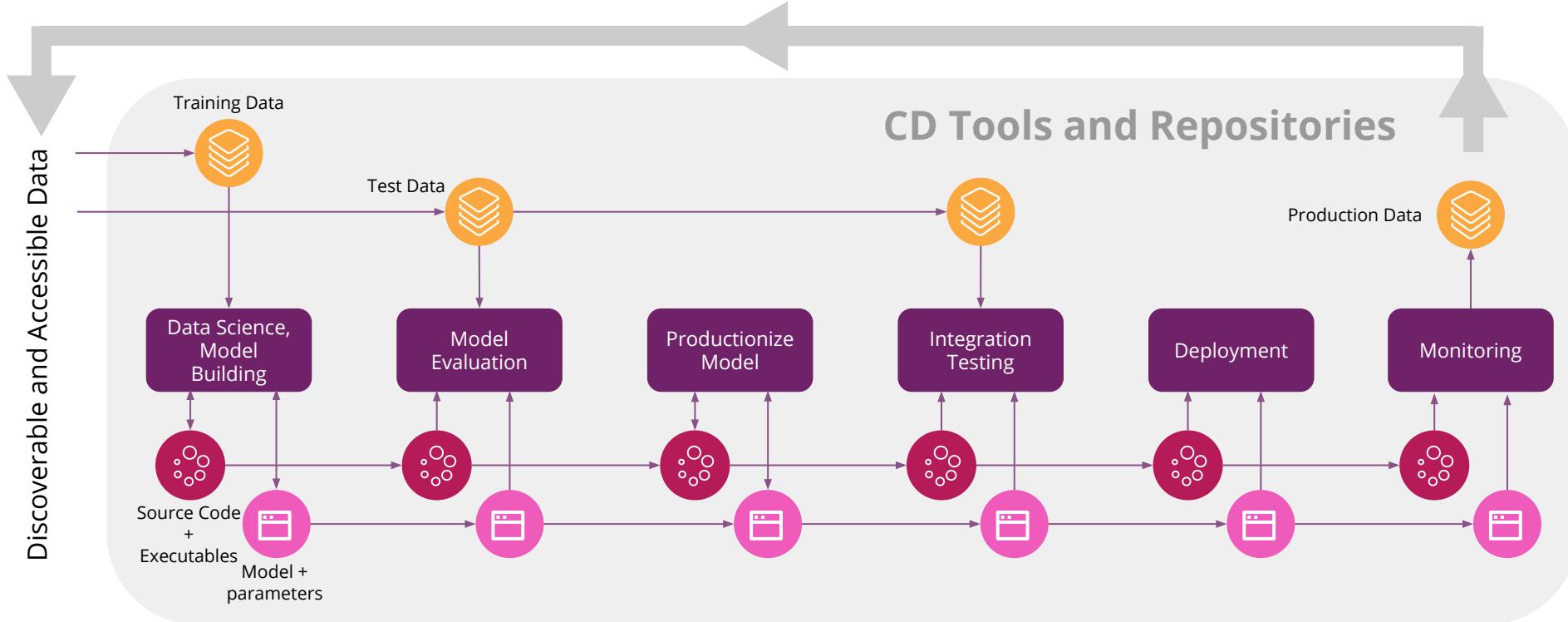
HOW DOES IT WORK



HOW DOES IT WORK



HOW DOES IT WORK



WHAT DO WE NEED IN OUR CD4ML STACK?



THE BUSINESS PROBLEM: PREDICT HOUSE PRICES

Make a pricing prediction based on a multitude of data:

- ZIP code
- Lot size and square feet
- No. of rooms
- Year built
- Style
- Extras (parking, pool, etc.)

Based on publicly available data from the [Zillow Group](#).



Picture from <https://www.zillowgroup.com/>

THE DATA

To train our machine learning models, we are using a publicly available data set from the Zillow Group containing:

- You can find the data [here](#)
- The original data are from the [kaggle platform](#)



WE WILL BUILD A SIMPLIFIED WEB APPLICATION

As a seller or a buyer, I want to get a price estimation based on my preferences.

Kitchen_refurbished (required)

Square_feet (required)

Pool (required)

Parking (required)

Multi_family (required)

Zipcode (required)

Style (required)

Hit submit to update the prediction

Submit

Prediction: 346576.1

WHAT WE WILL USE IN THIS WORKSHOP

There are many options for tools and technologies to implement CD4ML

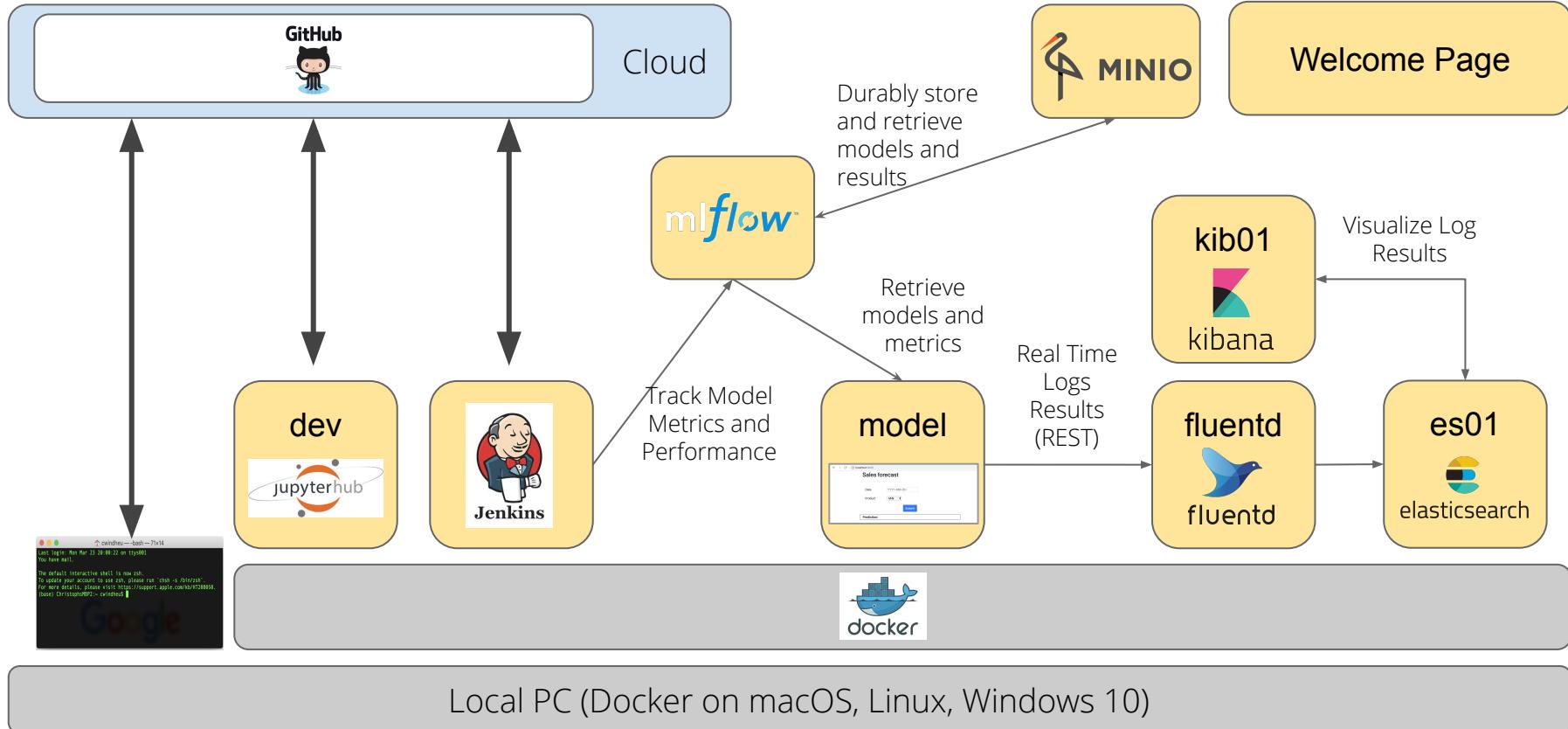


Break (10 min)

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 TDD)
 - a. Run the ML pipeline
 - b. Run experiments
 - c. Change parameters, improve score and monitor models in ML Flow
 - d. Overview of codebase and design
 - e. Testing (TDD)
3. **Continuous Deployment:** Setup a performance test of the model, which only allows automatic deployment if the model passes the quality threshold
4. **Monitor** the model “in the wild”

The Workshop Infrastructure



SETUP JENKINS

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

The screenshot shows a GitHub repository page for 'ThoughtWorksInc / CD4ML-Scenarios'. The repository has 7 watches, 1 star, and 3 forks. The main content is the file '2-SetupJenkins.md'. A commit by 'ciwin' was made 12 hours ago, adding changes to the instructions. The file contains 29 lines (16 sloc) and is 1.13 KB in size. The content of the file is as follows:

```
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

```

LOG IN TO JENKINS



Welcome to Jenkins!

admin

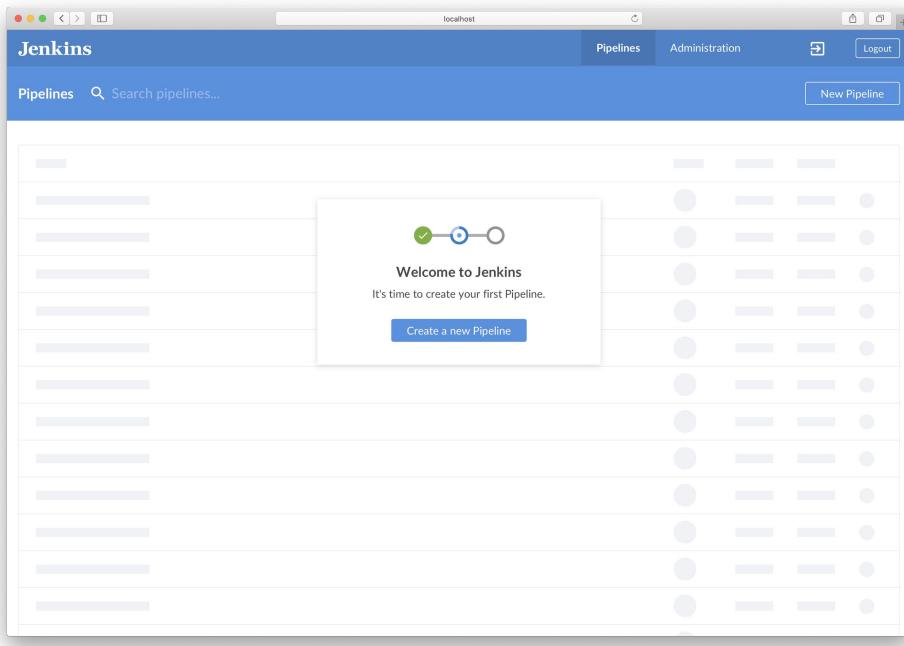
.....

Sign in

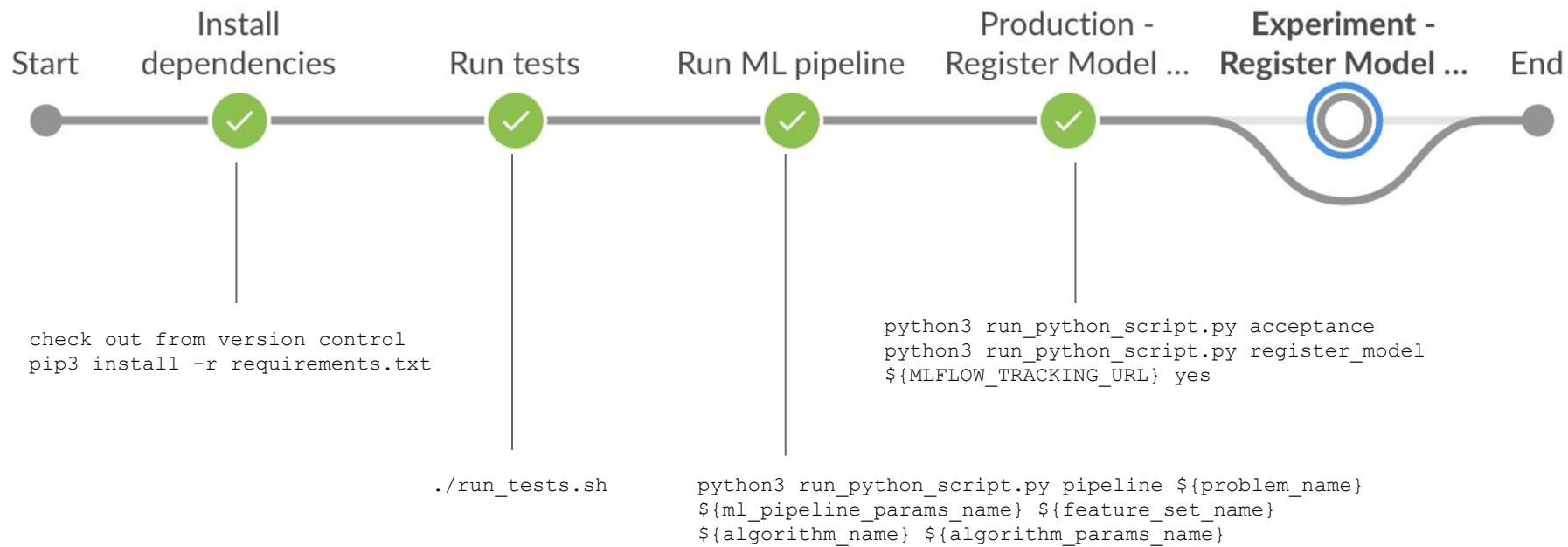
Keep me signed in

<http://localhost:10000/blue>

SETUP THE PIPELINE



OUR END GOAL JENKINS PIPELINE



RUN THE WEB APPLICATION (1)



CD4ML Home Groceries ▾ Housing ▾

Welcome! Select the scenario you want to try!



Sales Forecasting Scenario

Predict the amount of a product required to order on a particular date

[Use latest valid model »](#)

[View All Models »](#)

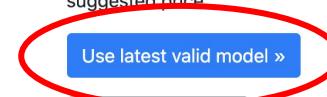


Housing Price Prediction Scenario

Prediction the difference between the actual price of a house and the suggested price

[Use latest valid model »](#)

[View All Models »](#)



<http://localhost:11000/>

RUN THE WEB APPLICATION (2)

Kitchen_refurbished *(required)*

Square_feet *(required)*

Pool *(required)*

Parking *(required)*

Multi_family *(required)*

Zipcode *(required)*

Style *(required)*

Hit submit to update the prediction

Prediction: 346576.1

<http://localhost:11000/>

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 TDD)
 - a) Run the ML pipeline
 - b) Run experiments
 - c) Change parameters, improve score and monitor models in ML Flow
 - d) Overview of codebase and design
 - e) Testing (TDD)
3. **Continuous Deployment:** Setup a performance test of the model, which only allows automatic deployment if the model passes the quality threshold
4. **Monitor** the model “in the wild”

Machine Learning Frameworks Principles

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

Jack of all concerns

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 факт_код == sales_hist_dates факт_код[idx]) & (orders_all дата_заказа < last_date)]
    order_endcust = order_cust[['Date', 'Completion', 'UnitSales']]
    order_endcust = order_endcust[(order_endcust['VolumeSold'] > 0)]
    order_endcust_agg = order_endcust.groupby(['Component'], as_index=False).agg(
        {'volume': 'mean', 'std', 'cnt'})
    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
```

Annotations from top-left to bottom-right:

- Date transformations
- Aggregation
- Statistics
- Feature generation
- Column Filtering
- Restructuring
- Row Filtering
- Feature generation
- Row Filtering
- Restructuring
- Row Filtering
- Row Filtering

- 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))
```

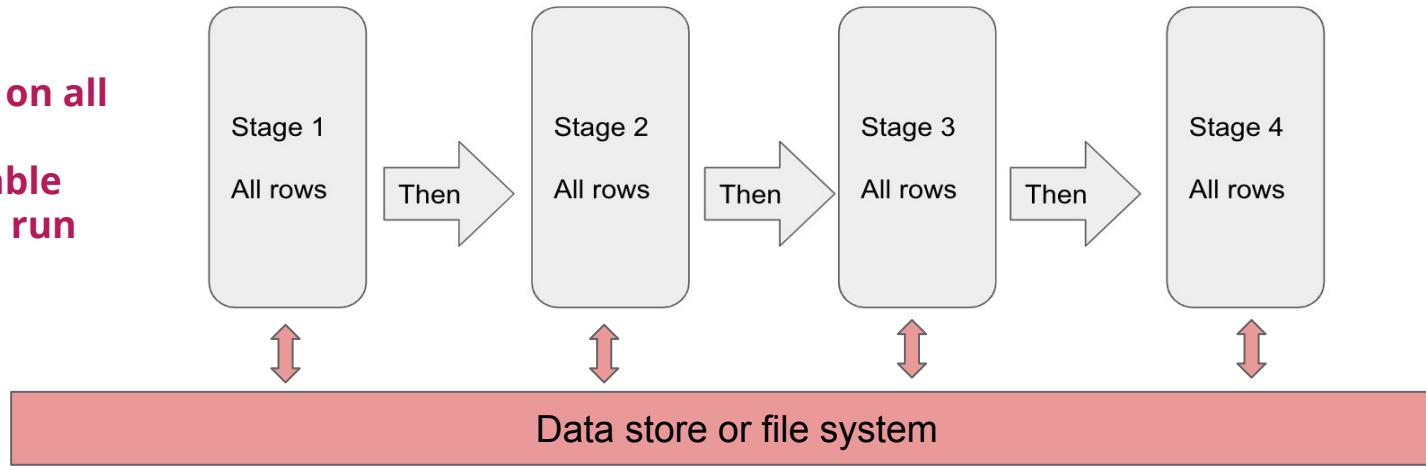
```
{"id": "88221657", "date": "2016-08-16", "item_nbr": "1963838"}
```

Nice properties of streams

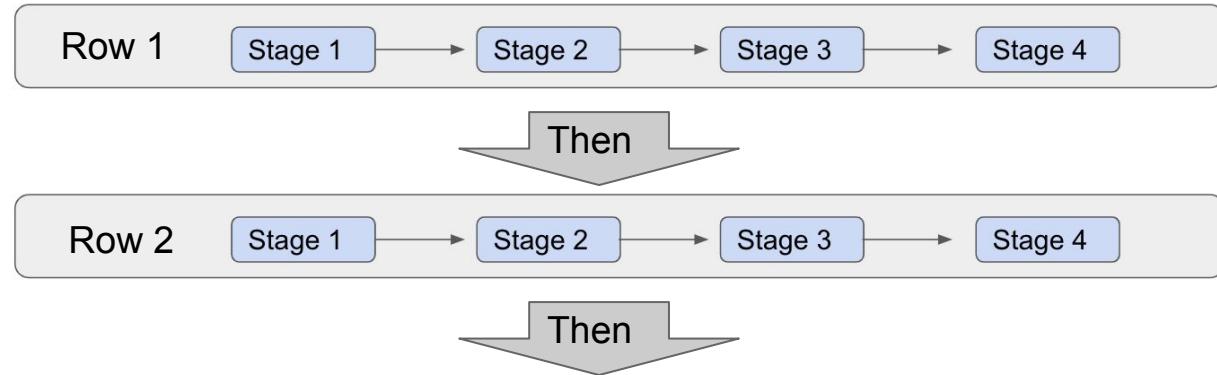
- They are generic. Encapsulates the ultimate source and its properties.
- Stream transformations are simple: stream_2 = (transform(i) for i in stream)
- Encourages modular code and testing and separation of concerns
- Memory efficient
- Leads to higher productivity development / debugging. Depth-first.

Row first, batch stages versus stage-first streaming model

Run Stage 1 on all rows. Write results to table or file. Then run stage 2 etc.



Run all stages on each row before moving on to second row.



Advantages of stage-first streaming model

- Data access / IO can be more easily be separated from the transformation logic. All logic runs the same wherever the stream of rows comes from.
- Data sources can be migrated trivially. File -> DB -> Spark -> Kafka
- Code logic can all be developed from a simulated or reduced-size data set in any environment, even local (i.e. laptop). No fighting the system to develop. Much higher productivity. Separation of concerns.
- Easier to test and debug
 - What if Row # 580,005 fails at Stage 3? Can write a single unit test that fails for that row and fix code to make it pass without having to rerun the whole pipeline and wait hours to reproduce failure. Fast feedback through all stages. Easier to identify root cause of problem.
- Usually more performant. Reduces the tendency to rely on SQL or database to perform transformation logic. In memory processing is usually faster. At least, allows easier performance tuning.

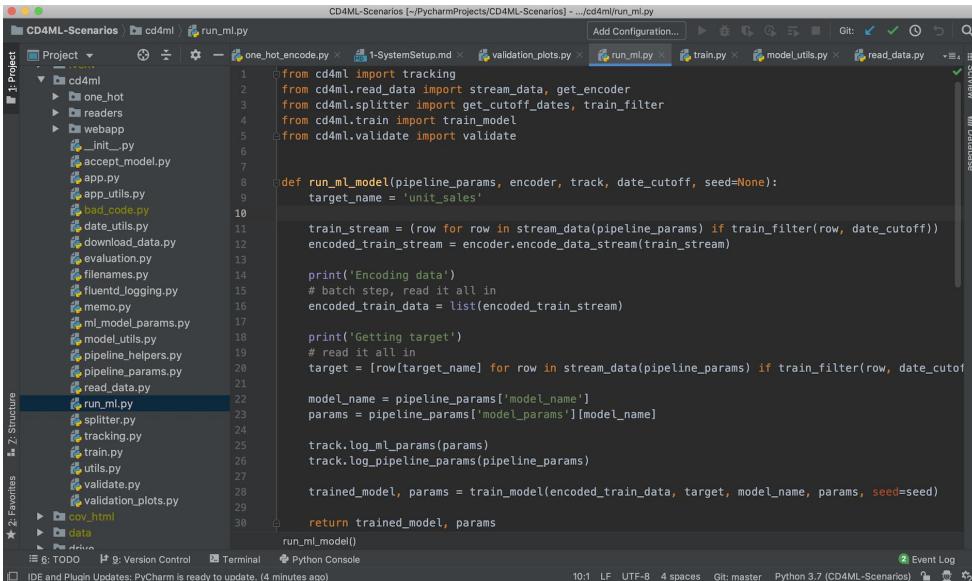
STEP BY STEP

1. **Doing the plumbing:** Set up the pipeline and see if it is working

2. **Data Science:** Develop the model and test it (with TDD)
 - a) Run the ML pipeline
 - b) Run experiments
 - c) Change parameters, improve score and monitor models in ML Flow
 - d) Overview of codebase and design
 - e) Testing (TDD)

3. **Continuous Deployment:** Setup a performance test of the model, which only allows automatic deployment if the model passes the quality threshold
4. **Monitor** the model “in the wild”

OVERVIEW OF THE CODE BASE AND SCENARIOS



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

- Project:** CD4ML-Scenarios
- Run:** cd4ml
- File:** run_ml.py
- Code Content (run_ml.py):**

```
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
31 run_ml_model()
```

- Toolbars and Status Bar:** Event Log, Version Control, Terminal, Python Console.
- Status Bar:** 10:1 LF, UTF-8, 4 spaces, Git: master, Python 3.7 (CD4ML-Scenarios).

- Structure of code base
- How to run the pipeline
After making changes
- How to view results of
models and keep track of
changes
- Changing algorithms,
params etc and not causing
chaos

ML Flow

The screenshot shows the MLflow web interface. At the top, there's a navigation bar with the 'mlflow' logo, 'Experiments', 'Models', 'GitHub', and 'Docs' buttons.

The main area is titled 'Experiments' and shows a list of experiments: 'Default' and 'houses'. The 'houses' experiment is selected, indicated by a blue background.

For the 'houses' experiment, the following details are displayed:

- Experiment ID: 1
- Artifact Location: s3://cd4ml-ml-flow-bucket/1
- Notes: None
- Search Runs: metrics.rmse < 1 and params.model = "tree" and tags.mlflow.source.type = "LOCAL"
- State: Active
- Buttons: Search, Clear, Compare, Delete, Download CSV

A table below shows one matching run:

	Start Time	Run Name	User	Source	Version	Algorithm	Algorithm	Features	mad_sco	num_valid	r2_score	DidPassA	BuildNur
<input type="checkbox"/>	2020-09-05 15: 58c6...	root	run.py	7281bd	default	default	default	1034...	10044	0.456	yes	9	

<http://localhost:11000/>

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 TDD)
 - a. Run the ML pipeline
 - b. Run experiments
 - c. Change parameters, improve score and monitor models in ML Flow
 - d. Overview of codebase and design
 - e. Testing (TDD)
3. **Continuous Deployment:** Setup a performance test of the model, which only allows automatic deployment if the model passes the quality threshold
4. **Monitor** the model “in the wild”

CONTINUOUS DEPLOYMENT

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

The screenshot shows a GitHub repository page for the organization **ThoughtWorksInc / CD4ML-Scenarios**. The repository has 10 pull requests, 51 stars, and 152 forks. The **Code** tab is selected, showing the file **4-ContinuousDelivery.md**. The file was last updated 1 hour ago by **ericnagler** with the commit message "Adjust instructions and images". There is 1 contributor listed. The file contains 48 lines (38 sloc) and is 2.7 KB in size. The content of the file is partially visible, showing the heading **Exercise 4 - Continuous Delivery**.

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.”

- Ensure that the experiment that we just ran with lasso can be promoted to production using the “default” algorithm.
- Change the minimum acceptance criteria to simulate an unexpected model performance drop and watch the jenkins pipeline capture the error
- Transfer the contents from lasso big_alpha.json to default.json
- We will also observe that this will not impact the model running in production

Continuous Deployment

How are we checking for performance (accept_model.py)

```
def is_model_accepted(model_id):
    model_files = get_model_files(model_id)
    params = get_json(model_files['ml_pipeline_params'])

    metric_name = params['acceptance_metric'].replace("", "")
    threshold_min = float(params['acceptance_threshold_min'])
    threshold_max = float(params['acceptance_threshold_max'])

    metrics = get_json(model_files['model_metrics'])
    metric_value = metrics[metric_name]

    accepted = threshold_min <= metric_value <= threshold_max

    message = get_message(model_id, metric_name, metric_value,
                          threshold_min, threshold_max, accepted)

    return accepted, message
```

}

Retrieve the machine learning pipeline parameters

}

Retrieve the r2 score of the model

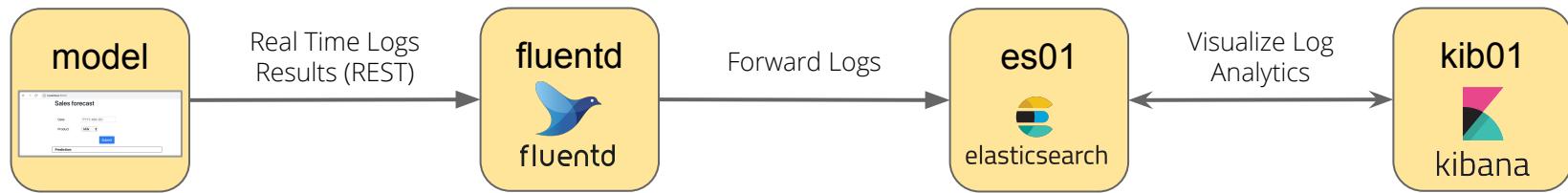
}

Check the that the threshold score is within range

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 TDD)
 - a. Run the ML pipeline
 - b. Run experiments
 - c. Change parameters, improve score and monitor models in ML Flow
 - d. Overview of codebase and design
 - e. Testing (TDD)
3. **Continuous Deployment:** Setup a performance test of the model, which only allows automatic deployment if the model passes the quality threshold
4. **Monitor** the model “in the wild”

Technical Details of the setup of fluentd, elasticsearch and kibana (EFK Stack)



- **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 our case, we collect the logs from the model and forward them to ElasticSearch.
- **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 are then visualized in Kibana.
- **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/housing/5-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 file '6-KibanaLogVisualization.md' is selected. The file content is as follows:

```
Branch: master -> CD4ML-Scenarios / instructions / 6-KibanaLogVisualization.md
ericnagler Adjust exercise number 9b7f082 23 seconds ago
1 contributor

45 lines (27 sloc) 2.34 KB
```

Exercise 6: Model Monitoring and Observability

Goals

- 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. A red box highlights the search bar and the 'Search predictions' button. Another red box highlights the filter section and the 'Filter predictions' button. A third red box highlights the field selection section and the 'Select fields' button. A fourth red box highlights the table and JSON preview section and the 'Inspect table and json file' button.

Search predictions

Filter predictions

Select fields

Inspect table and json file

Discover

New Save Open Share Inspect

Search

Add filter

model-*

Search field names

Filter by type 0

Selected fields

_source

Available fields

Popular

- @timestamp
- t _type
- t item_name
- prediction
- t _id
- t _index
- # score

Count

12 hits

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

2020-04-05 00:00 2020-04-06 00:00 2020-04-07 00:00 2020-04-08 00:00 2020-04-09 00:00 2020-04-10 00:00 2020-04-11 00:00

@timestamp per 3 hours

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

WRAP UP

- You learned:
 - The Principles of CD4ML
 - The components of a tech stack you need to run CD4ML
 - Data Science best practice in the industry
 - Bringing ML model into production
 - Run CD4ML on your laptop!
- Continue the conversation in our Google group!
 - <https://groups.google.com/a/thoughtworks.com/g/cd4ml-webinar>
- Reach out and ask questions on slack channel or with a direct message, we would love to hear your feedback!

REFERENCES / LINKS

- **Github**

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

- **Google Group**

<https://groups.google.com/a/thoughtworks.com/g/cd4ml-webinar>

- **Articles**

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

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

<https://www.thoughtworks.com/continuous-delivery-for-machine-learning>

<https://opendatascience.com/continuous-delivery-for-machine-learning/>

QUESTIONS

THANK YOU

ThoughtWorks®