

# Continuous Intelligence

***Moving Machine Learning  
Application into Production Reliably***

Christoph Windheuser  
Danilo Sato  
Emily Gorcenski  
Arif Wider  
ThoughtWorks Inc.

WORKSHOP ON WHY  
AND HOW TO APPLY  
**CONTINUOUS  
DELIVERY TO  
MACHINE LEARNING  
(CD4ML)**

# Structure of Today's Workshop

- INTRODUCTION TO THE TOPIC
- EXERCISE 1: SETUP
- EXERCISE 2: DEPLOYMENT PIPELINE
- BREAK
- EXERCISE 3: ML PIPELINE
- EXERCISE 4: TRACKING EXPERIMENTS
- EXERCISE 5: MODEL MONITORING



ThoughtWorks®

*5000+ technologists with 40 offices in 14 countries*



*Partner for technology driven business transformation*



100+

books written

#1

in Agile and  
Continuous Delivery

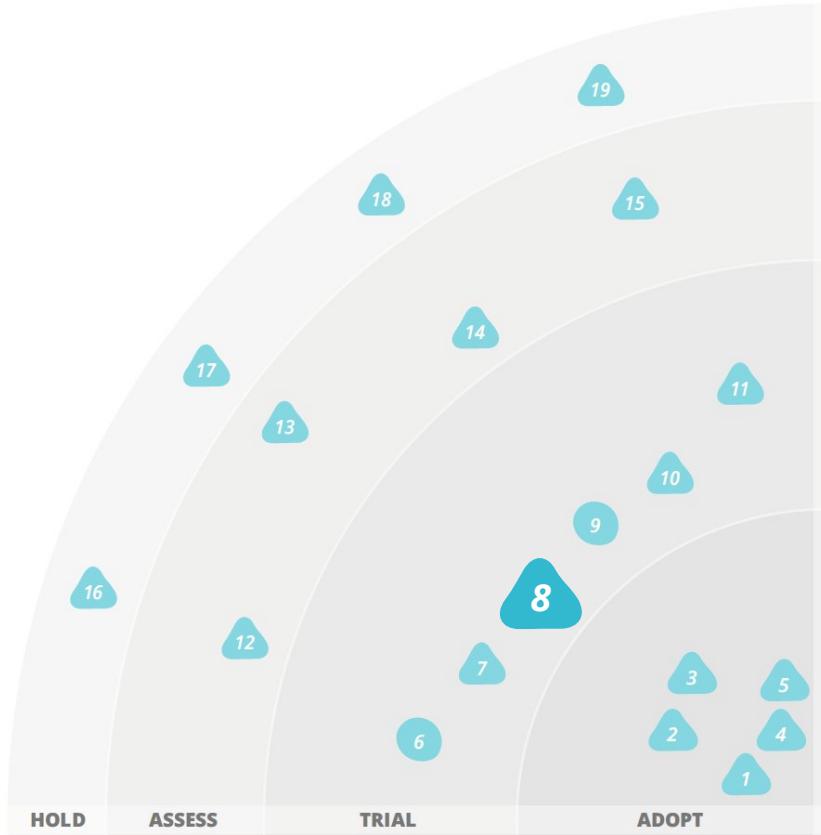




## TECHNIQUES

Continuous delivery #8  
for machine  
learning (CD4ML)  
models

ASSESS



# CONTINUOUS INTELLIGENCE CYCLE

## 5. Execute

Changing the real world!

## 4. Productionize

Planning and prioritizing actions. Hypothesis testing.

## 3. Model

Understanding, predicting, forecasting and pattern discovery.

## 1. Acquire

Values attributed to parameters.

## 2. Store, clean, curate, featurize

Data that has meaning and is fit for consumption and analysis.



# 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

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.

# 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:

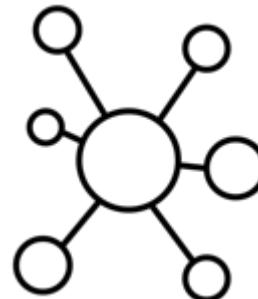
- Non-deterministic
- Hard to test
- Hard to explain
- Hard to improve

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

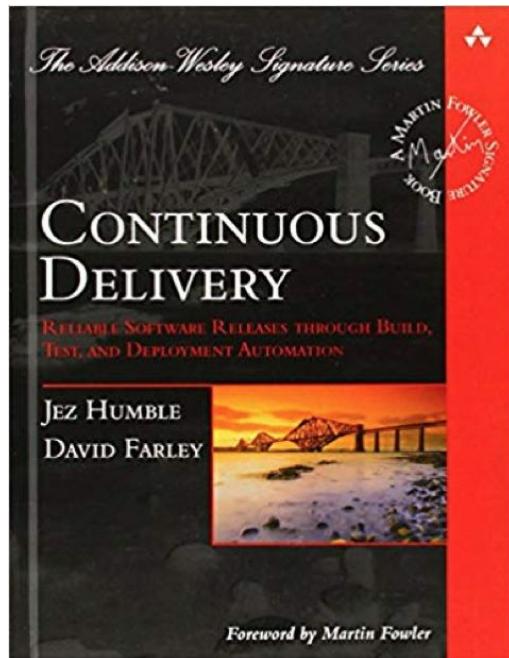
Dependencies

...

*“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

# PRINCIPLES OF CONTINUOUS DELIVERY



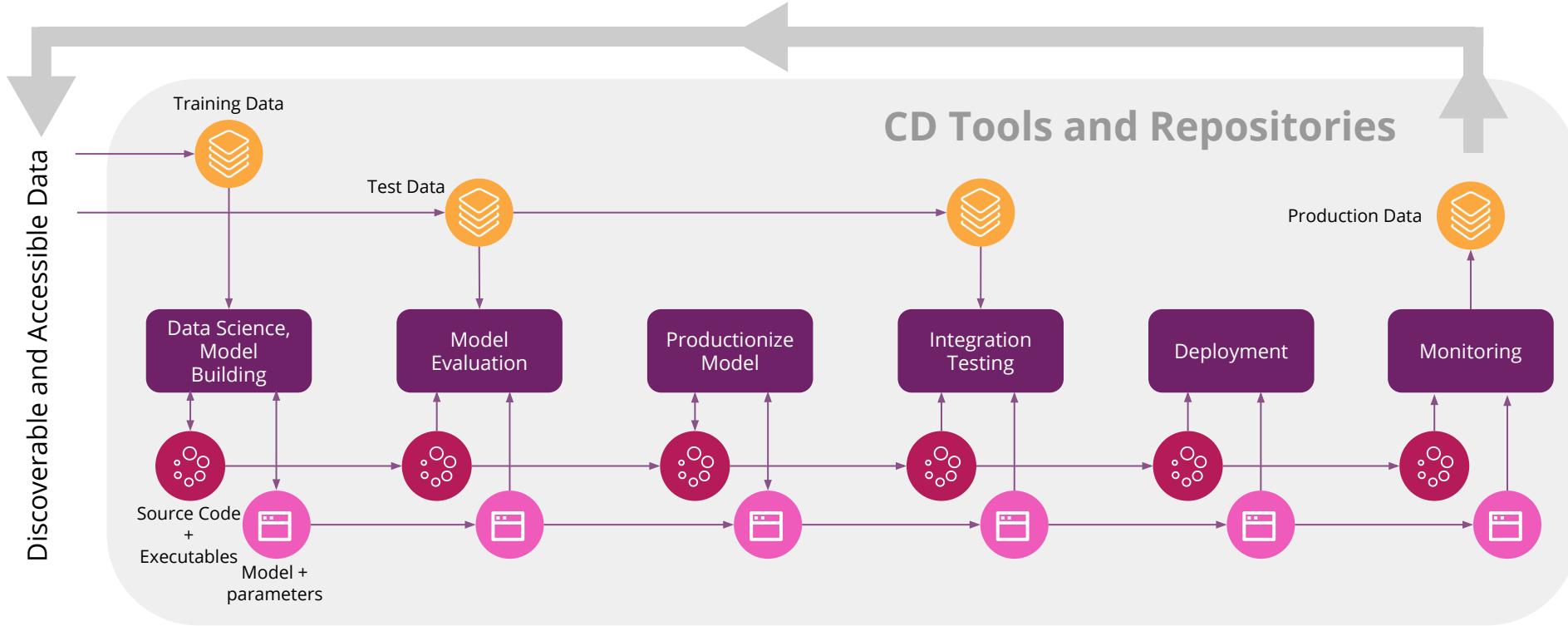
- Create a Repeatable, Reliable Process for Releasing Software
- Automate Almost Everything
- Build Quality In
- Work in Small Batches
- Keep Everything in Source Control
- Done Means "Released"
- Improve Continuously

# WHAT DO WE NEED IN OUR STACK?

*Doing CD with Machine Learning is still a hard problem*

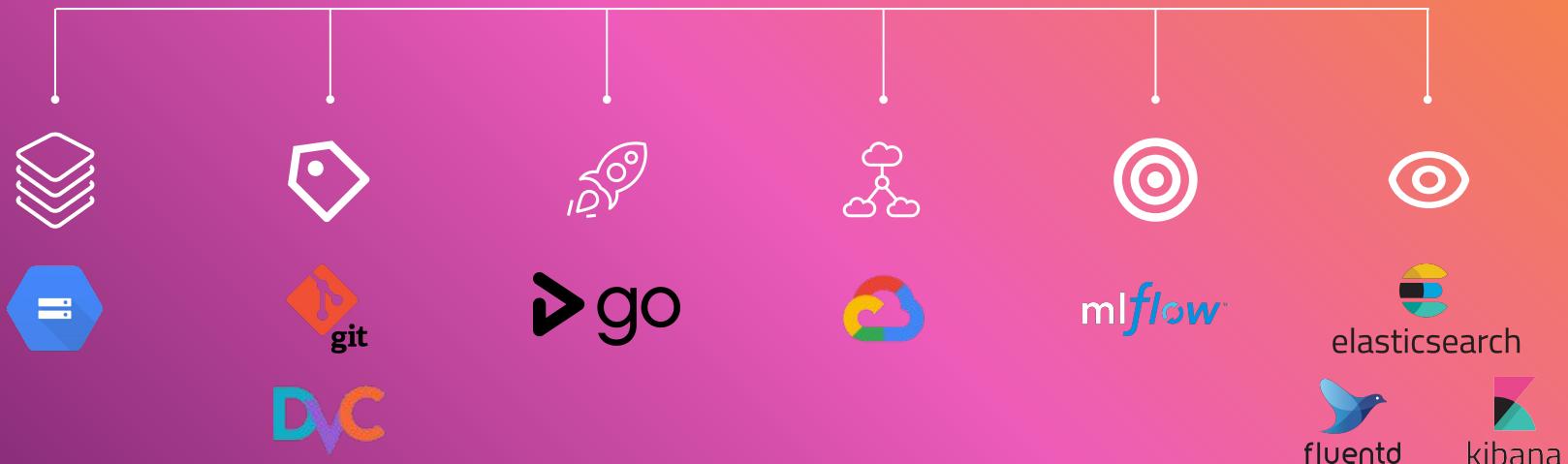


# PUTTING EVERYTHING TOGETHER



# WHAT WE WILL USE IN THIS WORKSHOP

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



# THE MACHINE LEARNING PROBLEM WE ARE EXPLORING TODAY

# A REAL BUSINESS PROBLEM

## RETAIL / SUPPLY CHAIN

Loss of sales, opportunity cost, stock waste, discounting

## REQUIRES

Accurate Demand Forecasting

## TYPICAL CHALLENGES

- Predictions Are Inaccurate
- Development Takes A Long Time
- Difficult To Adapt To Market Change Pace

# SALES FORECASTING FOR GROCERY RETAILER

Make predictions based on data from:

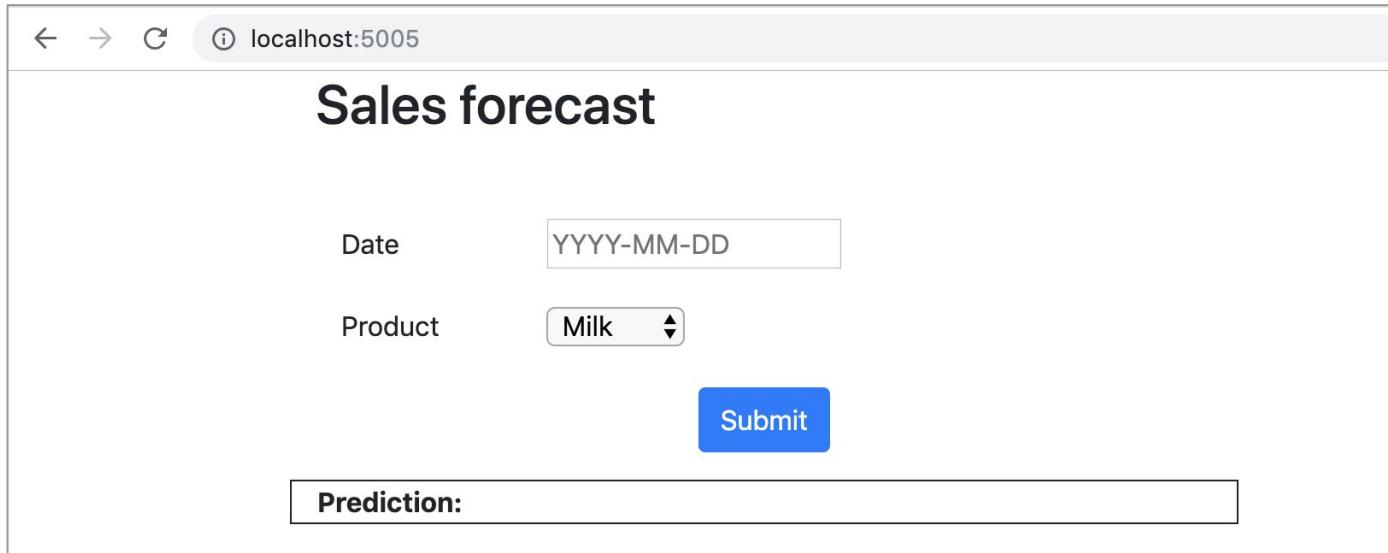
- 4,000 items
- 50 stores
- 125,000,000 sales transactions
- 4.5 years of data

TASK:

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

# THE SIMPLIFIED WEB APPLICATION

*As a buyer, I want to be able to **choose a product** and  
**predict how many units the product will sell** at a **future date**.*



The screenshot shows a web browser window with the URL `localhost:5005` in the address bar. The page title is **Sales forecast**. There are two input fields: one for **Date** (with placeholder `YYYY-MM-DD`) and one for **Product** (with dropdown menu showing **Milk**). Below these is a blue **Submit** button. At the bottom, there is a long input field labeled **Prediction:** which is currently empty.

# EXERCISE 1: SETUP

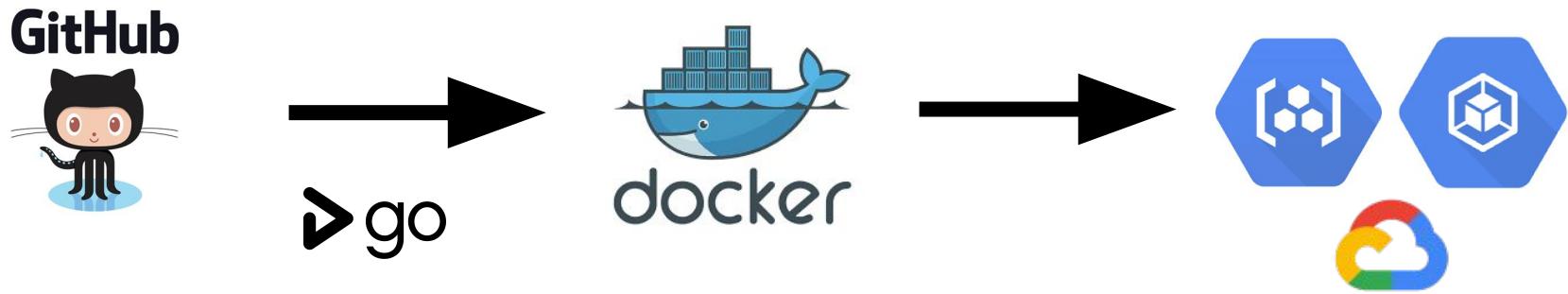
<https://github.com/ThoughtWorksInc/continuous-intelligence-workshop>

- Click on **instructions → 1-setup.md**
- Follow the steps to setup your local development environment

User ID assignment sheet: <http://bit.ly/cd4ml-strata19>

# DEPLOYMENT PIPELINE

*Automates the process of building, testing, and deploying applications to production*



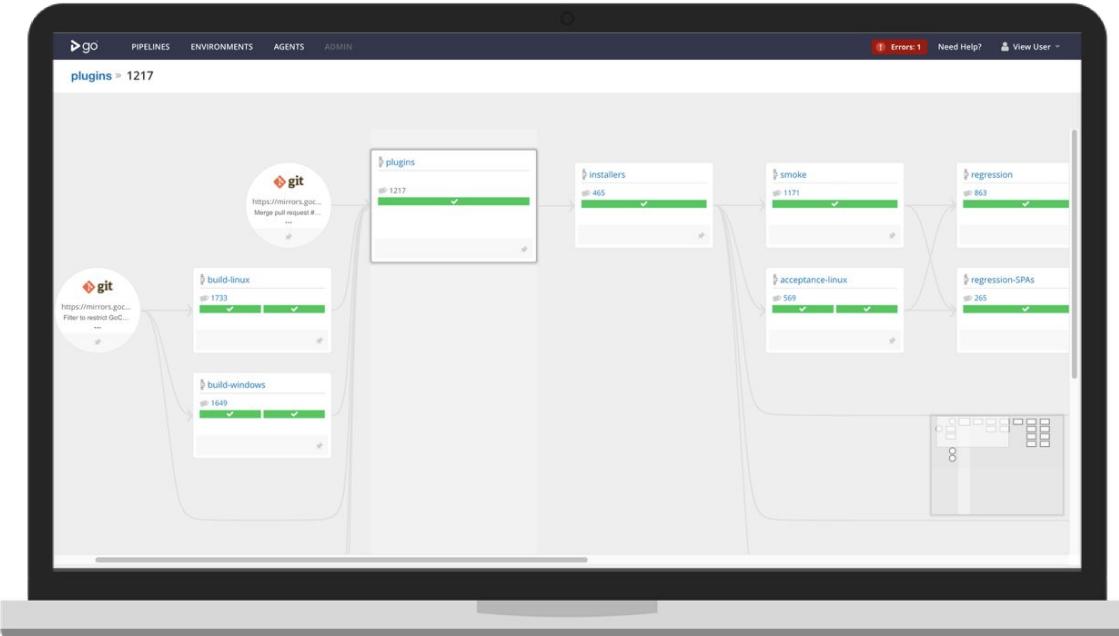
Application **code** in  
version control  
repository

Container image as  
**deployment**  
**artifact**

Deploy container  
to **production**  
**servers**

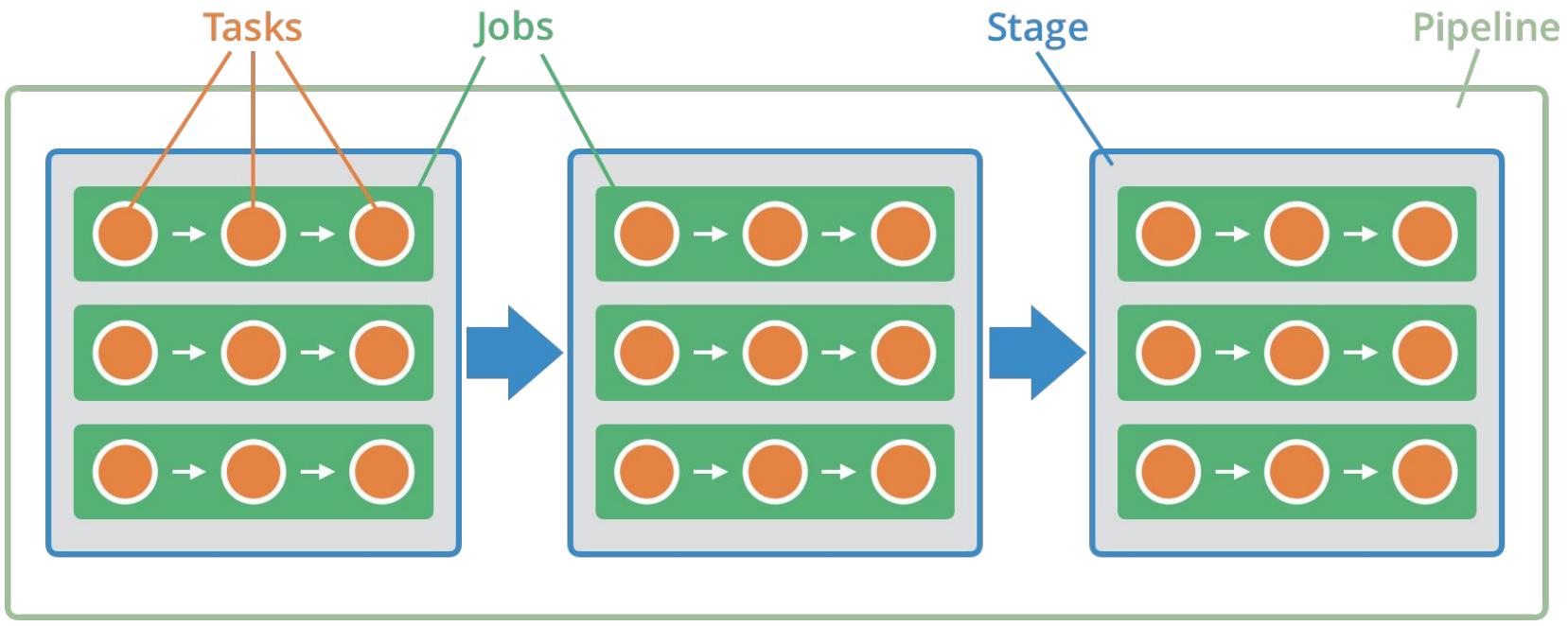
# >go

An Open Source Continuous Delivery server to model and visualise complex workflows



# ANATOMY OF A GOCD PIPELINE

Pipeline Group



# EXERCISE 2: DEPLOYMENT PIPELINE

<https://github.com/ThoughtWorksInc/continuous-intelligence-workshop>

- Click on **instructions → 2-deployment-pipeline.md**
- Follow the steps to setup your deployment pipeline
- GoCD URL: <https://gocd.cd4ml.net>

# BUT NOW WHAT?

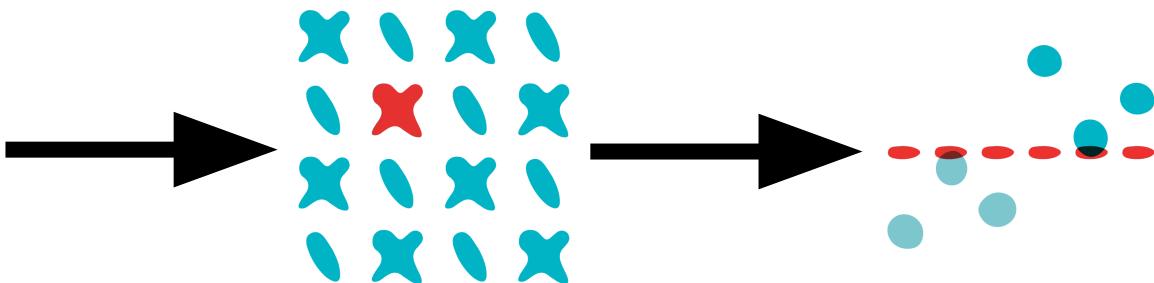
Once your model is in production...

- How do we **retrain** the model **more often**?
- How to **deploy** the retrained model **to production**?
- How to make sure we **don't break anything** when deploying?
- How to make sure that our modeling approach or parameterization is **still the best fit** for the data?
- How to **monitor** our model "**in the wild**"?

# BASIC DATA SCIENCE WORKFLOW



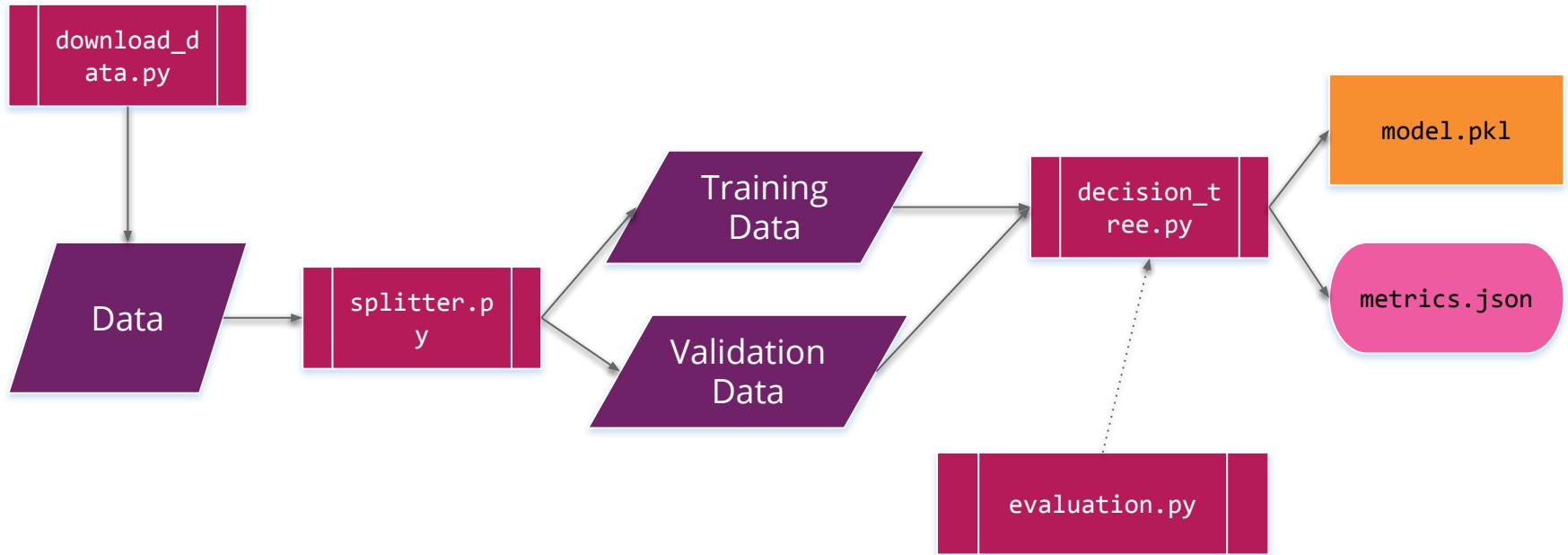
Gather data and extract features



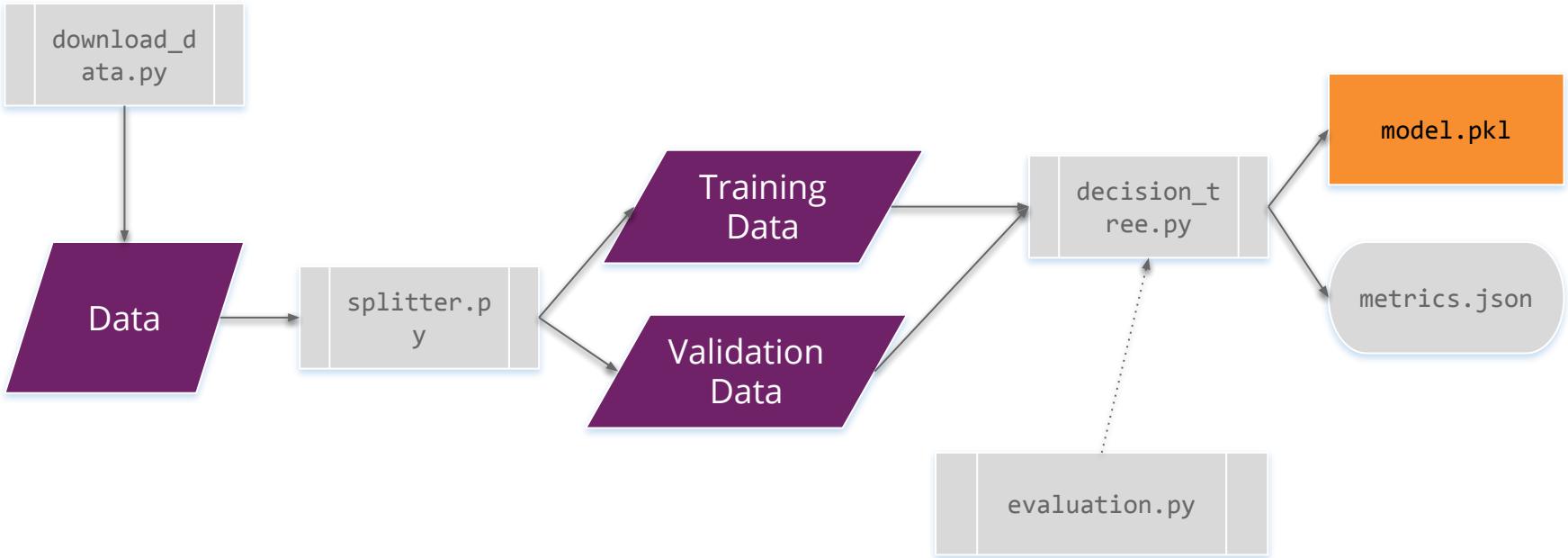
Separate into training and validation sets

Train model and evaluate performance

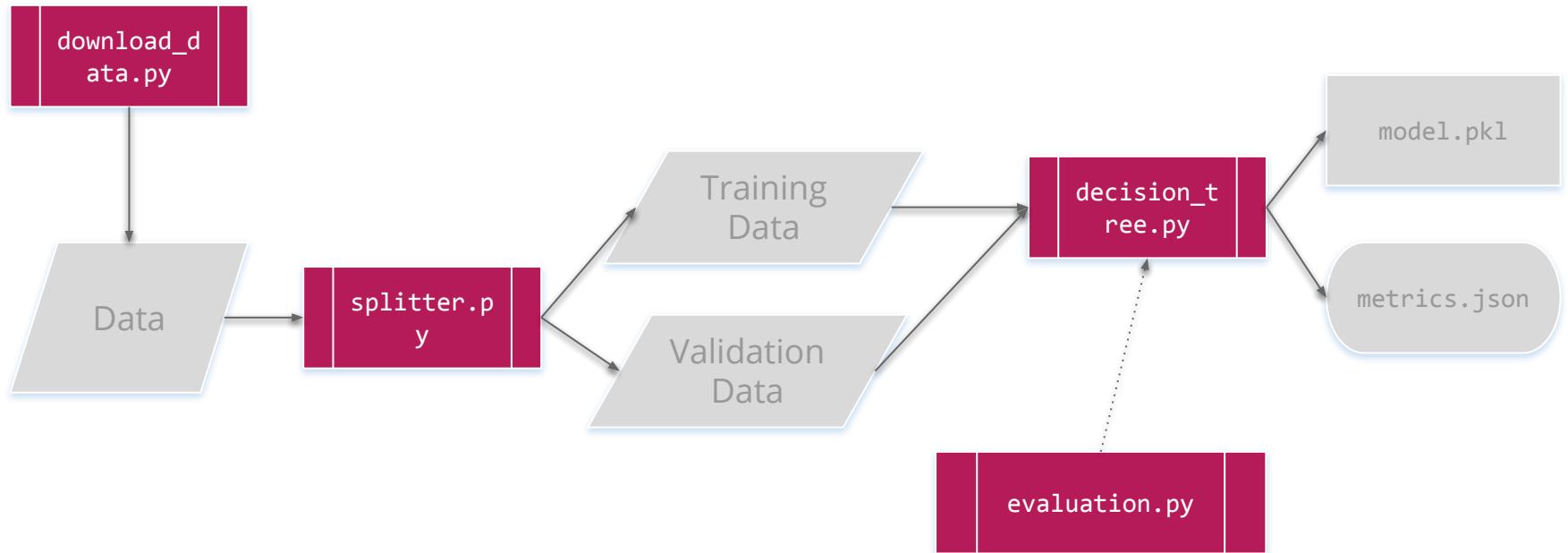
# SALES FORECAST MODEL TRAINING PROCESS



# CHALLENGE 1: THESE ARE LARGE FILES!



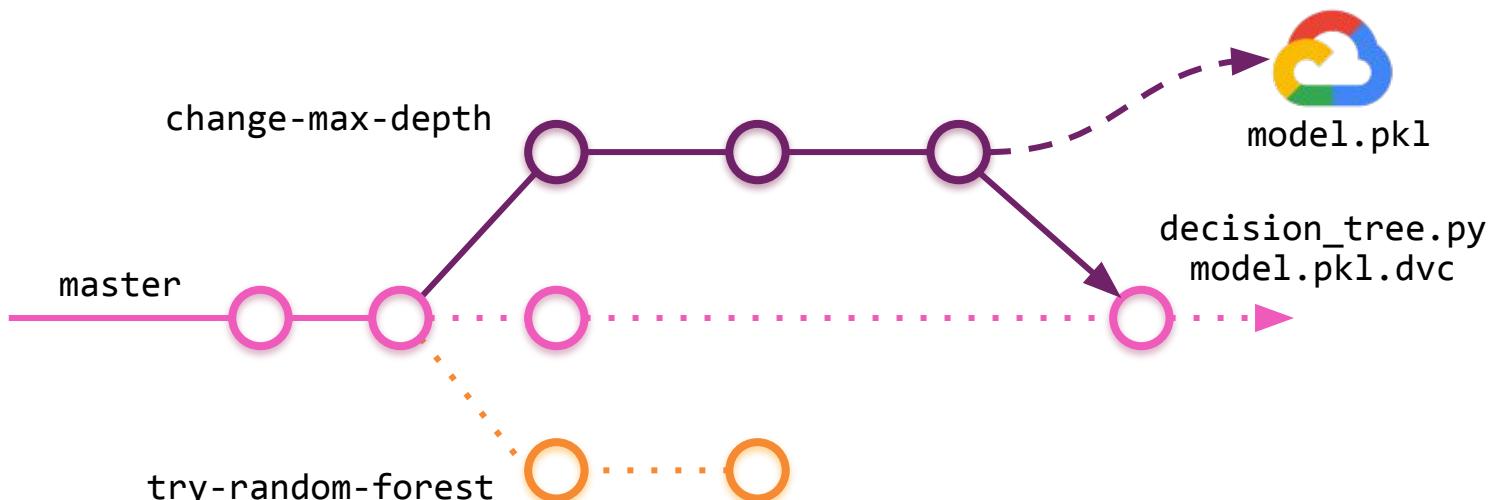
# CHALLENGE 2: AD-HOC MULTI-STEP PROCESS



# SOLUTION: dvc

*data science version control*

- dvc is **git porcelain** for storing large files using cloud storage
- dvc connects model training steps to create **reproducible workflows**



# ANATOMY OF A DVC COMMAND

```
dvc run -d src/download_data.py  
-o data/raw/store47-2016.csv python src/download_data.py
```

This runs a **command** and creates a **.dvc** file. The dvc file points to the **dependencies**. The **output files** are versioned and stored in the cloud by running **dvc push**.

When you use the output files (**store47-2016.csv**) as dependencies for the next step, a *pipeline* is automatically created.

You can re-execute an entire pipeline with one command: **dvc repro**

# EXERCISE 3: MACHINE LEARNING PIPELINE

<https://github.com/ThoughtWorksInc/continuous-intelligence-workshop>

- Click on **instructions** →  
**3-machine-learning-pipeline.md**
- Follow the steps on your local development environment  
and in GoCD to create your Machine Learning pipeline

# HOW DO WE TRACK EXPERIMENTS?

**We need to track the scientific process and evaluate our models:**

- Which experiments and hypothesis are being explored?
- Which algorithms are being used in each experiment?
- Which version of the code was used?
- How long does it take to run each experiment?
- What parameter and hyperparameters were used?
- How fast are my models learning?
- How do we compare results from different runs?



An Open Source platform for managing end-to-end machine learning lifecycle

mlflow GitHub Docs

## Experiments < user1

user2 Experiment ID: 1 Artifact Location: gs://cd4ml-mlflow-tracking/1

user1

Search Runs: metrics.rmse < 1 and params.model = "tree" State: Active ▾ Search

Filter Params: alpha, lr Filter Metrics: rmse, r2 Clear

1 matching run Compare Delete Download CSV

	Date ▾	User	Run Name	Source	Version	Parameters	Metrics
<input type="checkbox"/>	2019-04-28 00:03:29	go	5	decision_tree.py	b24402	model: RANDOM_FOREST, n_estimators: 10	nwrmsle: 0.743, r2_score: 0.109

# EXERCISE 4: TRACKING EXPERIMENTS

<https://github.com/ThoughtWorksInc/continuous-intelligence-workshop>

- Click on **instructions** →  
**4-tracking-experiments.md**
- Follow the steps to track ML training in mlflow
- MLflow URL: <https://mlflow.cd4ml.net>

# HOW TO LEARN CONTINUOUSLY?

We need to capture production data to improve our models:

- Track model usage
- Track model inputs to find training-serving skew
- Track model outputs
- Track model interpretability outputs to identify potential bias or overfit
- Track model fairness to understand how it behaves against dimensions that could introduce unfair bias

# EFK STACK

*Monitoring and Observability infrastructure*



Open Source **data collector** for unified logging



## elasticsearch

Open Source **Search Engine**



## kibana

Open Source web UI to **explore and visualise data**



*An Open Source UI that makes it easy to explore and visualise the data index in Elasticsearch*

The screenshot shows the Kibana Discover interface. On the left, a sidebar lists navigation options: Discover (selected), Visualize, Dashboard, Timelion, Dev Tools, and Management. The main area displays a histogram titled 'tag:user1.prediction' with a time range from April 28th 2019, 00:11:55.605 to April 28th 2019, 00:26:55.605. The histogram shows two bars at the end of the timeline, both reaching a count of 3. Below the histogram is a table with columns 'Time' and '\_source'. Three log entries are listed:

Time	_source
April 28th 2019, 00:26:41.000	tag: user1.prediction prediction: 32.2 date: 2019-04-26 item_nbr: 99197 family: GROCERY I class: 1067 perishable: 0 transactions: 1000 year: 2019 month: 4 day: 26 dayofweek: 4 days_til_end_of_data: 0 dayoff: false @timestamp: April 28th 2019, 00:26:41.000 _id: tm8fYw0BhP4U-0zcXkvk _type: _doc _index: logstash-2019.04.27 _score: -
April 28th 2019, 00:26:37.000	tag: user1.prediction prediction: 32.2 date: 2019-04-26 item_nbr: 105574 family: GROCERY I class: 1045 perishable: 0 transactions: 1000 year: 2019 month: 4 day: 26 dayofweek: 4 days_til_end_of_data: 0 dayoff: false @timestamp: April 28th 2019, 00:26:37.000 _id: Um8fYw0BQ4eaJlTUTQNh _type: _doc _index: logstash-2019.04.27 _score: -
April 28th 2019, 00:26:32.000	tag: user1.prediction prediction: 32.1 date: 2019-04-17 item_nbr: 105574 family: GROCERY I class: 1045 perishable: 0 transactions: 1000 year: 2019 month: 4 day: 17 dayofweek: 2 days_til_end_of_data: 0 dayoff: false @timestamp: April 28th 2019, 00:26:32.000 _id: tG8fYw0BhP4U-0zcPEso _type: _doc _index: logstash-2019.04.27 _score: -

# EXERCISE 5: MODEL MONITORING

<https://github.com/ThoughtWorksInc/continuous-intelligence-workshop>

- Click on **instructions** → **5-model-monitoring.md**
- Follow the steps to log prediction events
- Kibana URL: <https://kibana.cd4ml.net>

# SUMMARY - WHAT HAVE WE LEARNED?

# CD4ML

- Proper data/model versioning tools enable reproducible work to be done in parallel.
- No need to maintain complex data processing/model training scripts.
- We can then put data science work into a Continuous Delivery workflow.
- **Result: Continuous, on-demand AI development and deployment, from research to production, with a single command.**
- **Benefit: production AI systems that are always as smart as your data science team.**

# join.thoughtworks.com

## THANK YOU!

Danilo Sato

Christoph Windheuser

Emily Gorcenski

Arif Wider

[\(dsato@thoughtworks.com\)](mailto:dsato@thoughtworks.com)

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

[\(egorcens@thoughtworks.com\)](mailto:egorcens@thoughtworks.com)

[\(awider@thoughtworks.com\)](mailto:awider@thoughtworks.com)

**ThoughtWorks®**