CLOUDERA

DATA SCIENCE BEST PRACTICES with CDSW

سابك

sabic

Objectives of the workshop

- Clarify the architecture and inner workings of CDSW
- Provide guidelines for Data Science / ML development
- Provide guidelines for model deployment

Agenda

9h30 - 10h00 Introduction (Cloudera Data Science / ML offering)

10h00 - 10h30 CDSW Architecture

10h30 - 11h 20 DS / ML development guidelines

11h20 - 11h30 PAUSE

11h30 - 12h15 Deployment guidelines

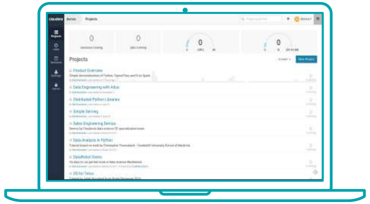
12h15 - 12h30 Next Steps + Questions

INTRODUCTION
(ML an DS @Cloudera)

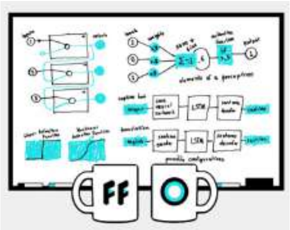
Cloudera ML offering

Modern enterprise platform, tools and expert guidance to help you unlock business value with ML/AI

Open **platform** to build, train, and deploy many scalable ML applications



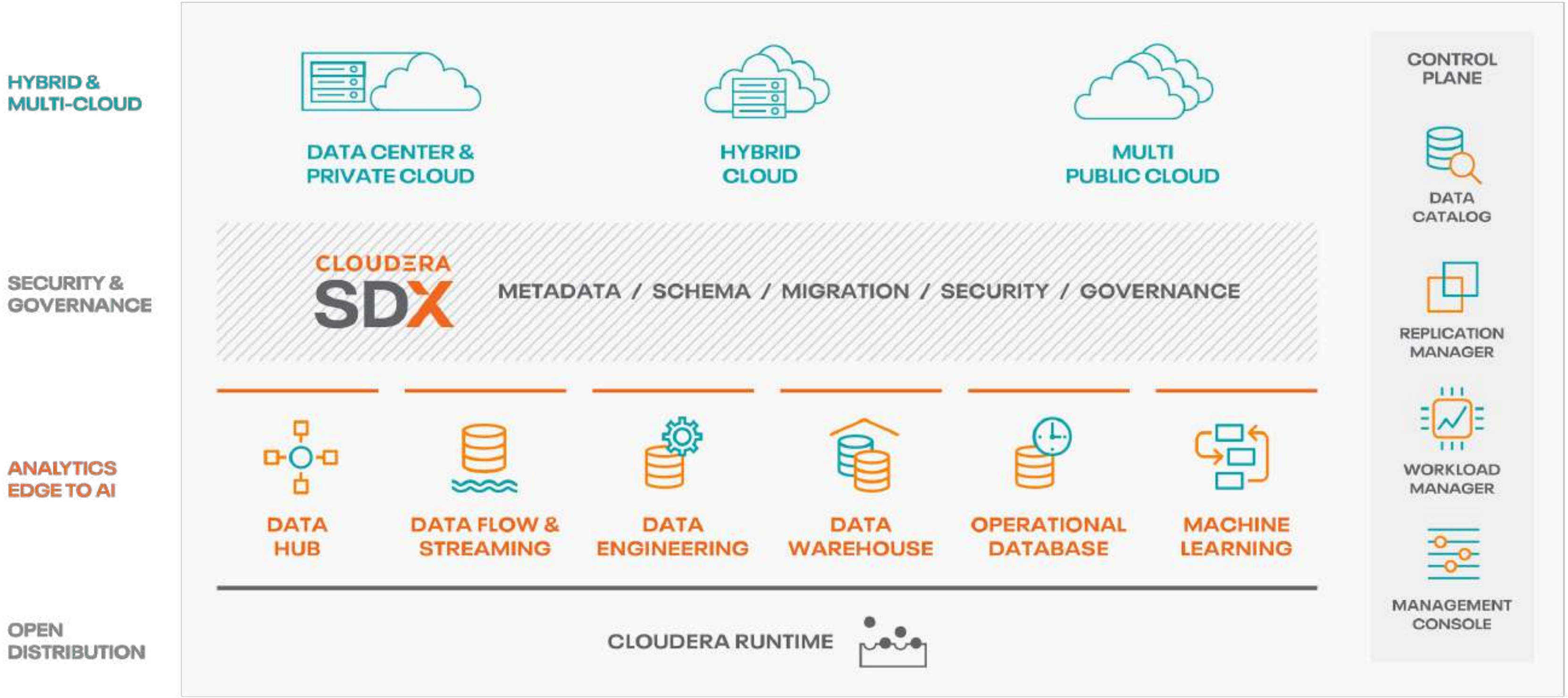
Comprehensive data science **tools** to accelerate team productivity



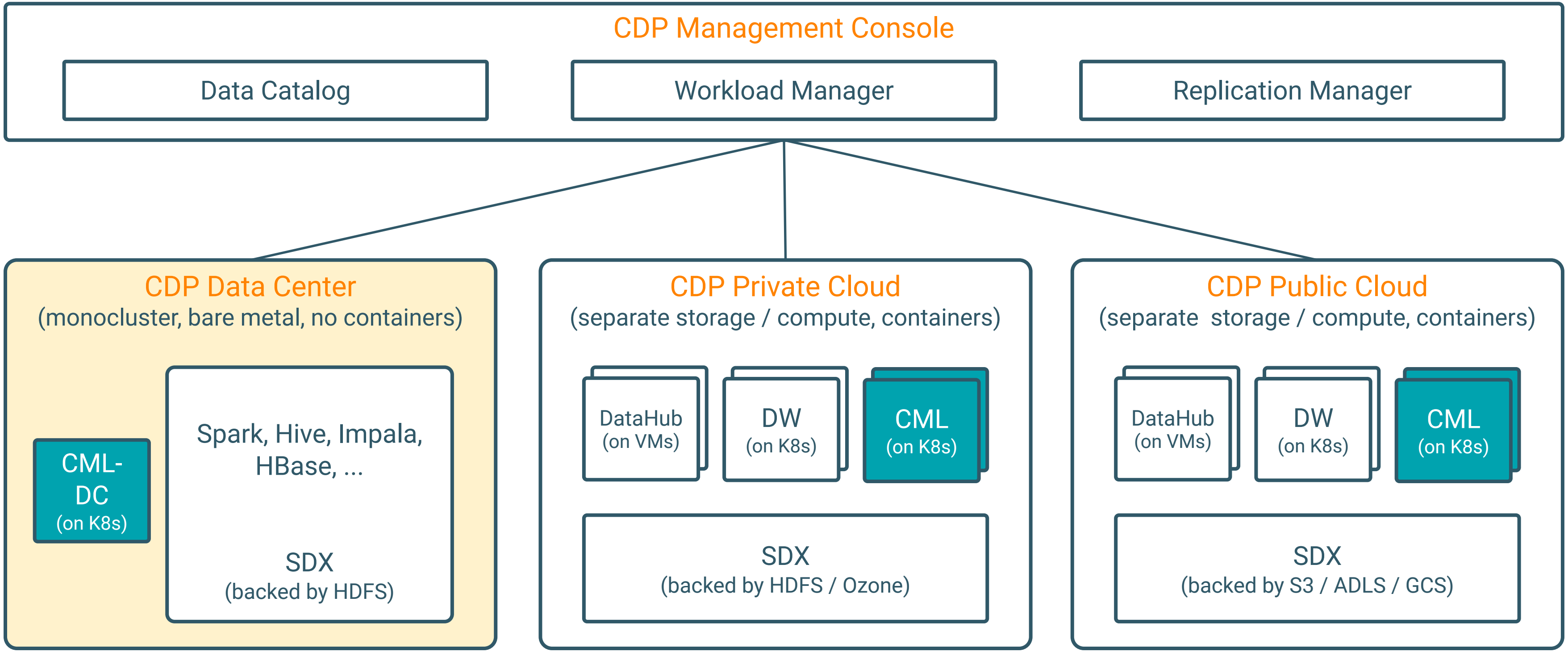
Expert guidance & services to fast track value & scale

PLATFORM

CLOUDERA DATA PLATFORM



CDP OFFERS HYBRID, MULTI-CLOUD ARCHITECTURE

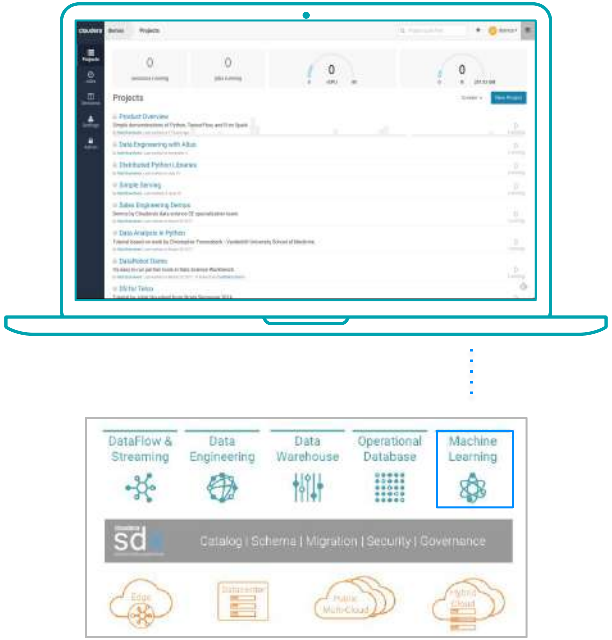


CLOUDERA DATA SCIENCE WORKBENCH

SELF SERVICE DATA SCIENCE TOOLING

CLOUDERA DATA SCIENCE WORKBENCH

Accelerate machine learning from research to production



For data scientists

- **Experiment faster**
  Use R, Python, or Scala with on-demand compute and secure CDH/HDP data access
- **Work together**
  Share reproducible research with your whole team
- **Deploy with confidence**
  Get to production consistently without recoding

For IT professionals

- **Bring data science to the data**
  Give your data science team more freedom while reducing the risk and cost of silos
- **Secure by default**
  Leverage common security and governance across workloads
- **Run anywhere**
  On-premises or in the cloud

CLOUDERA DATA SCIENCE WORKBENCH

Accelerate and simplify machine learning from research to production

ANALYZE DATA

- Explore data securely and share insights with the team

TRAIN MODELS

- Run, track, and compare reproducible experiments

DEPLOY APIs

- Deploy and monitor models as APIs to serve predictions

MANAGE SHARED RESOURCES

- Provide a secure, collaborative, self-service platform for your data science teams

A Generic Code Execution Engine

Any Language / Any Framework

Other

(custom Engines)

SERVICES

SMARTML

Cloudera's ML services for the journey to industrialized AI

	Strategy/Foundation	Develop	Deploy	Scale
PLATFORM & TOOLS	PLATFORM & TOOLS EDC + CDSW/CML			
EXPERT GUIDANCE	ML STRATEGY ENGAGEMENT Strategy prescription for use cases, organizational and process design ON DEMAND CDSW TRAINING		ADVISING & RESEARCH White glove concierge service including ML expert & SW/platform advising plus access to CFFL research reports and prototypes. RESIDENT DATA SCIENTIST CFFL expert onsite with your teams 1 week/month	
SERVICES FOR PRODUCTION & SCALE	ML PLATFORM ENABLEMENT Deploy and enable CDSW in your environment per best practices	ML APP DEVELOPMENT Identify, build, and document an ML app in your environment	ML APP DEPLOY Deploy existing ML app to an existing Cloudera production environment ML OPERATIONS Apply operational tooling, automated processes, and best practices to maintain production ML models	ML APP SCALE Increase data and/or performance for existing production ML app

CLOUDERA'S ML SERVICES FOR THE JOURNEY TO INDUSTRIALIZED AI

STRATEGY / FOUNDATION	DEVELOP	DEPLOY	SCALE

PLATFORM & TOOLS

PLATFORM & TOOLS
EDC + CDSW/CML

EXPERT GUIDANCE

ML STRATEGY ENGAGEMENT
Strategy prescription for use cases, organizational and process design

ON DEMAND CDSW TRAINING

RESIDENT DATA SCIENTIST
CFFL expert onsite with your teams 1 week/month

SERVICES FOR PRODUCTION & SCALE

ML PLATFORM ENABLEMENT
Deploy and enable CDSW in your environment per best practices

ML APP DEVELOPMENT
Identify, build, and document an ML app in your environment

ML APP DEPLOY
Deploy existing ML app to an existing Cloudera production environment

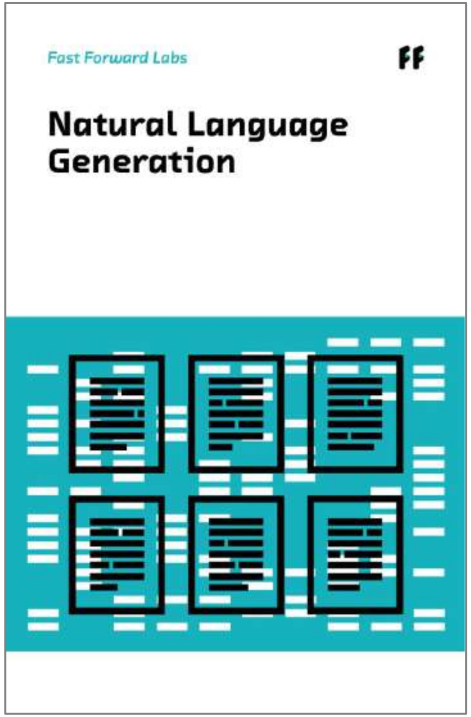
ML APP SCALE
Increase data and/or performance for existing production ML app

ML OPERATIONS
Apply operational tooling, automated processes, and best practices to maintain production ML models

FFL RESEARCH TOPICS

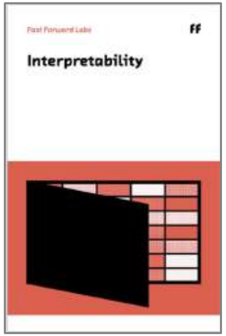
What's inside?

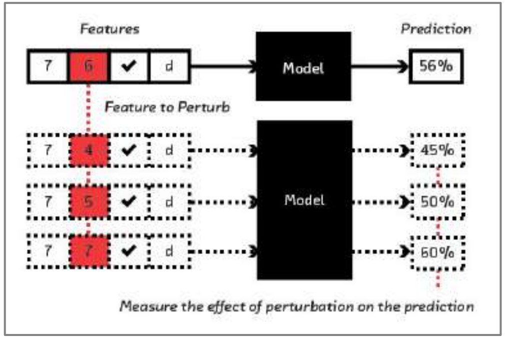
- A rigorous but conceptual explanation of a state-of-the-art algorithm:
  how it works, limitations, alternatives, data and hardware requirements
- A prototype that showcases the algorithm
- Commercial and open source landscape
- Ethics, future future implications, sci-fi short story



RESEARCH TOPICS

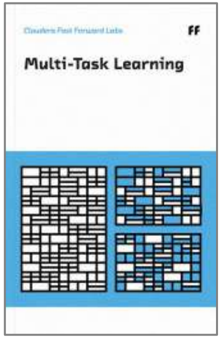
Practical guidance for implementing the recently possible

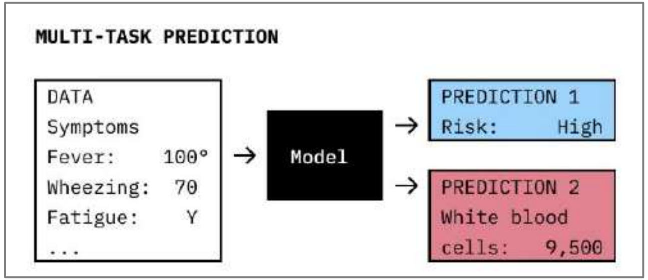




Understanding what's inside the black boxes

- Regulatory compliance and bias testing
- Customer churn reasoning
- Reverse engineering 3rd party models

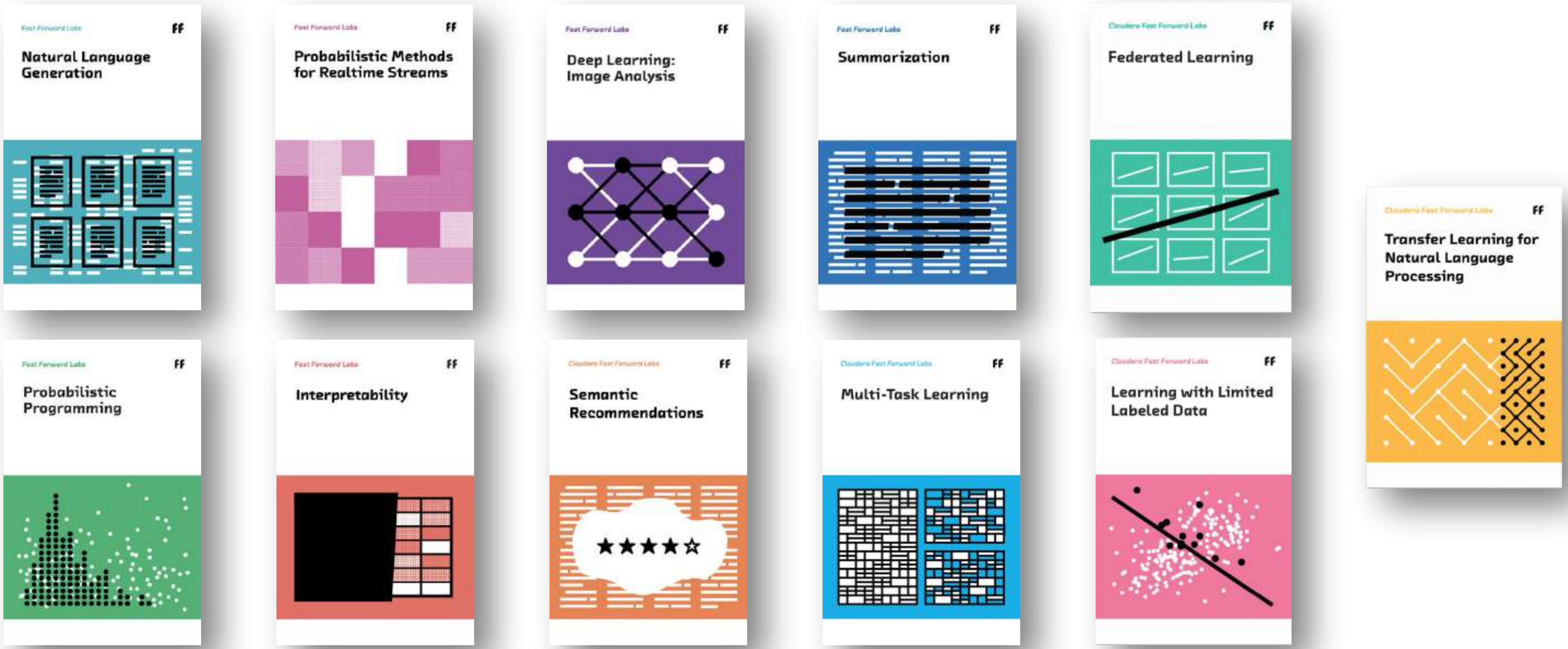




A novel approach to ML for deeper insights

- Analyzing review sentiment across retail vs luxury brands
- Detecting suspicious employee activity across bull vs bear markets
- Predicting IoT device failure across operating conditions

FAST FORWARD LABS – RESEARCH REPORTS



CDSW / CML Architecture

A MODERN DATA SCIENCE ARCHITECTURE

Containerized environments with scalable, on-demand compute

Built with Docker and Kubernetes

- Isolated, reproducible user environments

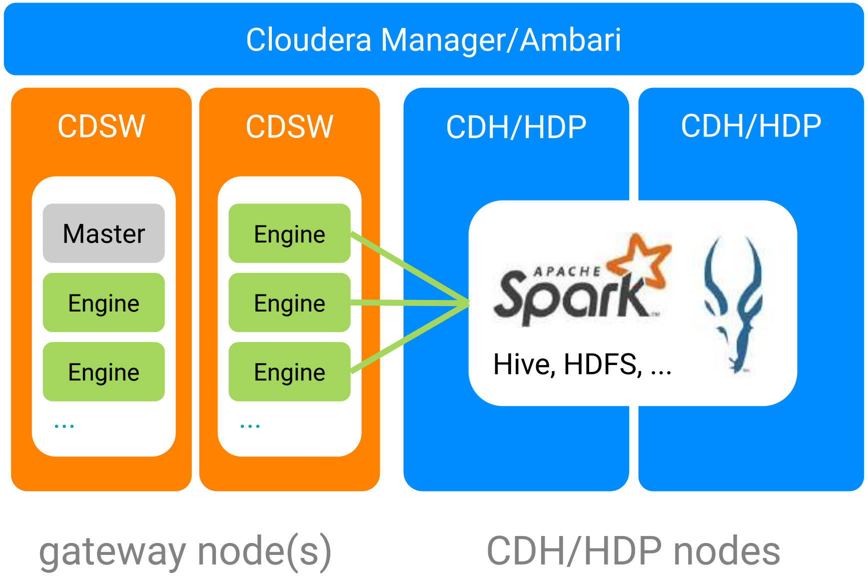
Supports both big and small data

- Local Python, R, Scala runtimes
- Schedule & share GPU resources
- Run Spark, Impala, and other CDH services

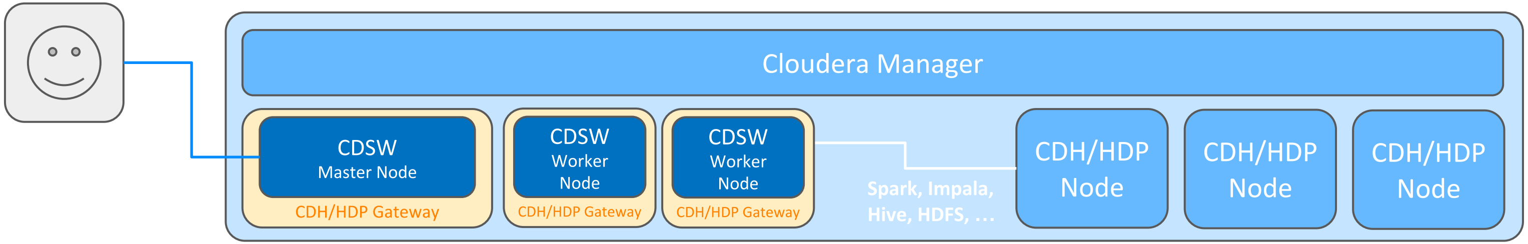
Secure and governed by default

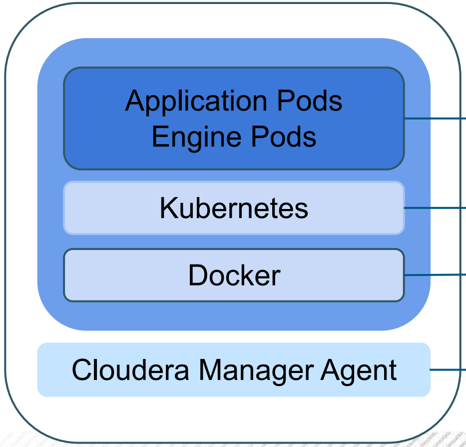
- Easy, audited access to Kerberized clusters
- Leverages SDX platform services

Deployed with Cloudera Manager



OVERALL ARCHITECTURE OVERVIEW





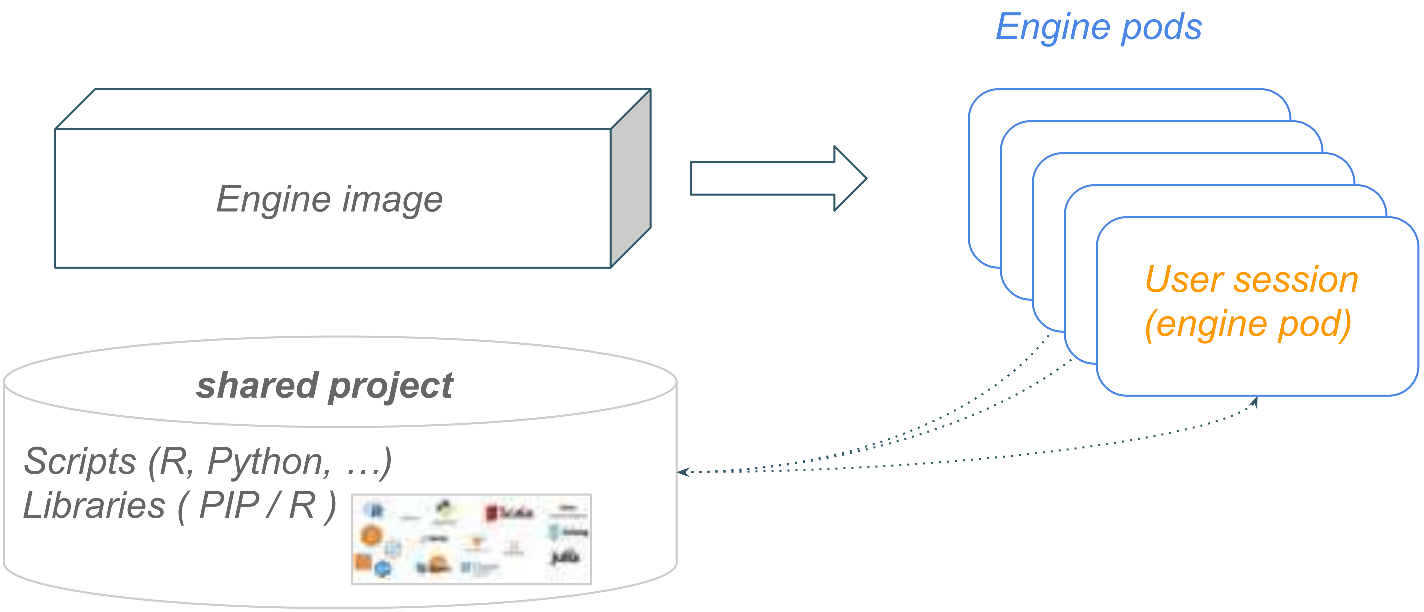
Cloudera Data Science Workbench application components and user workloads

Container orchestration service

Container runtime

Local management of CDH services

User sessions and project sharing



EXECUTION MODES

LOCAL VS DISTRIBUTED

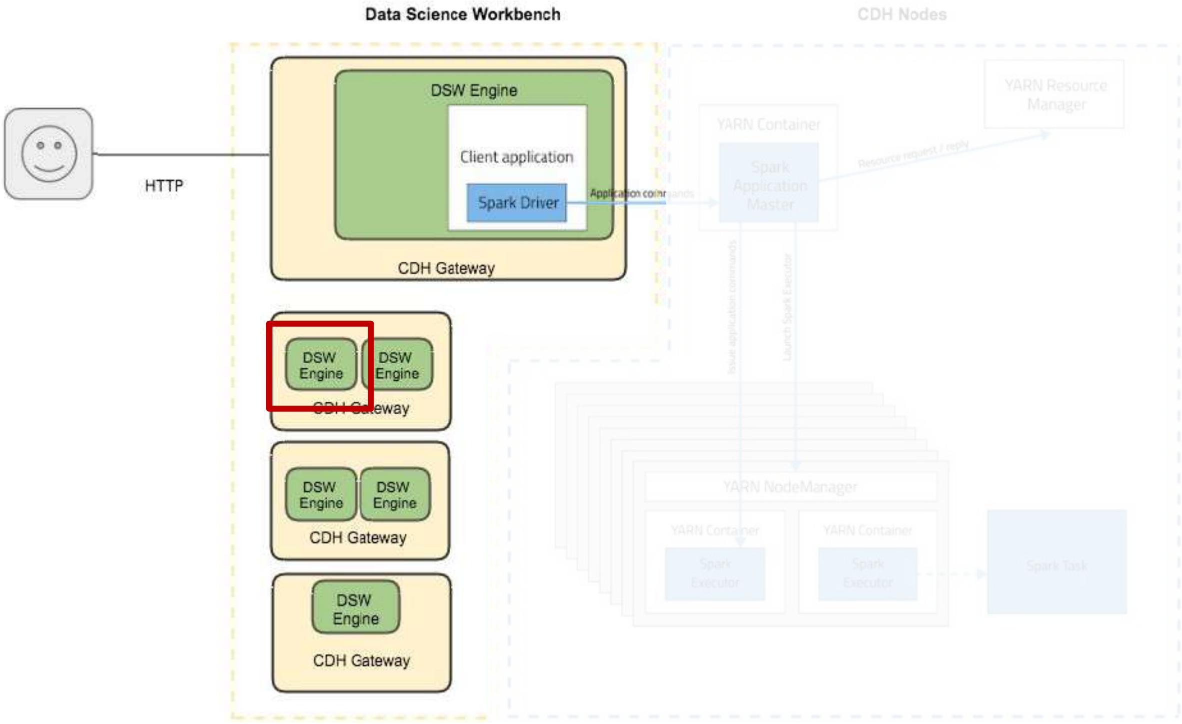
LOCAL - SESSION ON CDSW

Code execution within the context of a CDSW session

- Complete control of the environment :
  - Resources (CPU / Mem)
  - Language (R / Python / Scala / Java /….)
  - Packages / Libraries
- Potentially parallelized (Multi-core / shared mem)

Ideal for :

- Small to medium size data set
- Non distributed frameworks (R ; Sklearn ; Keras ; TensorFlow ; …)



DISTRIBUTED - ON CLUSTER

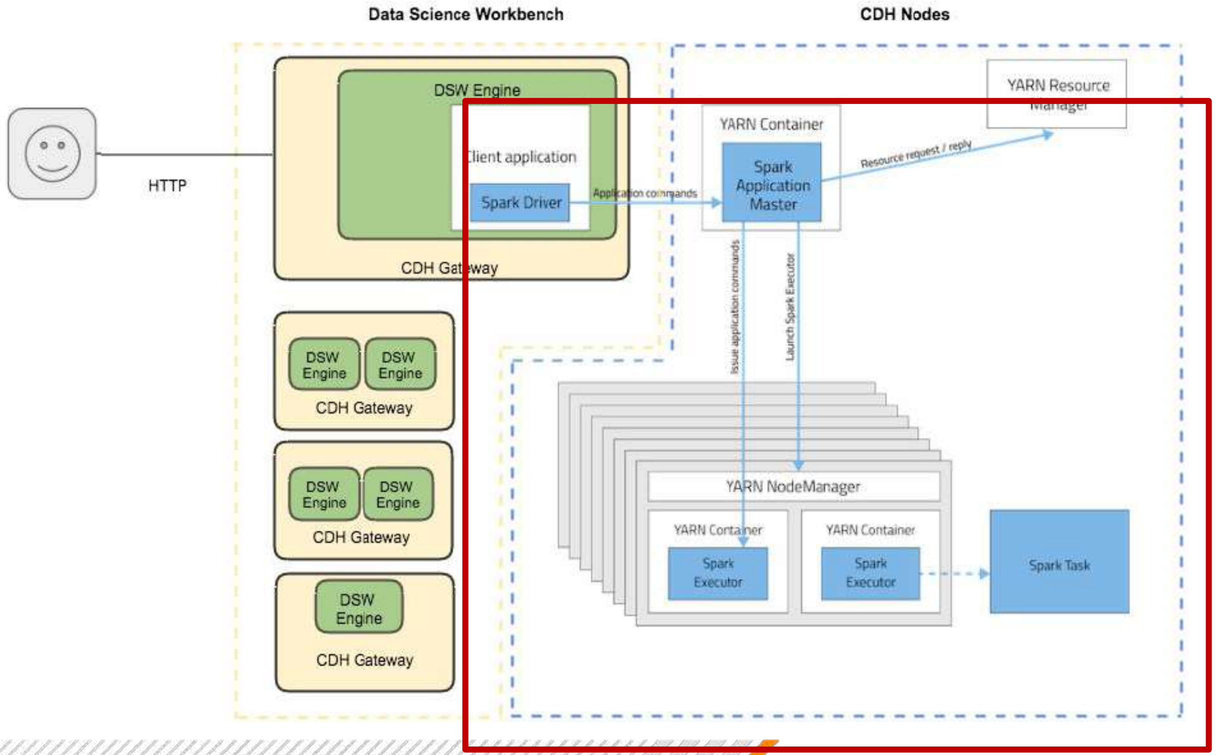
Code execution submitted to Cluster

- Distributed computing using SPARK
  - Scala (Native)
  - Python : PySpark
  - R : SpakR or sparklyr
- **Manual dependency management**
  - Scala / Java : Dependency shipping (see spark dependency mgmt)
  - R/Python : Dependent libraries installed on cluster (managed independently)
    or
    Dependency shipping (advanced)

Ideal for :

- Large data set / Complex learning

Tip : As much a possible leverage spark native functions



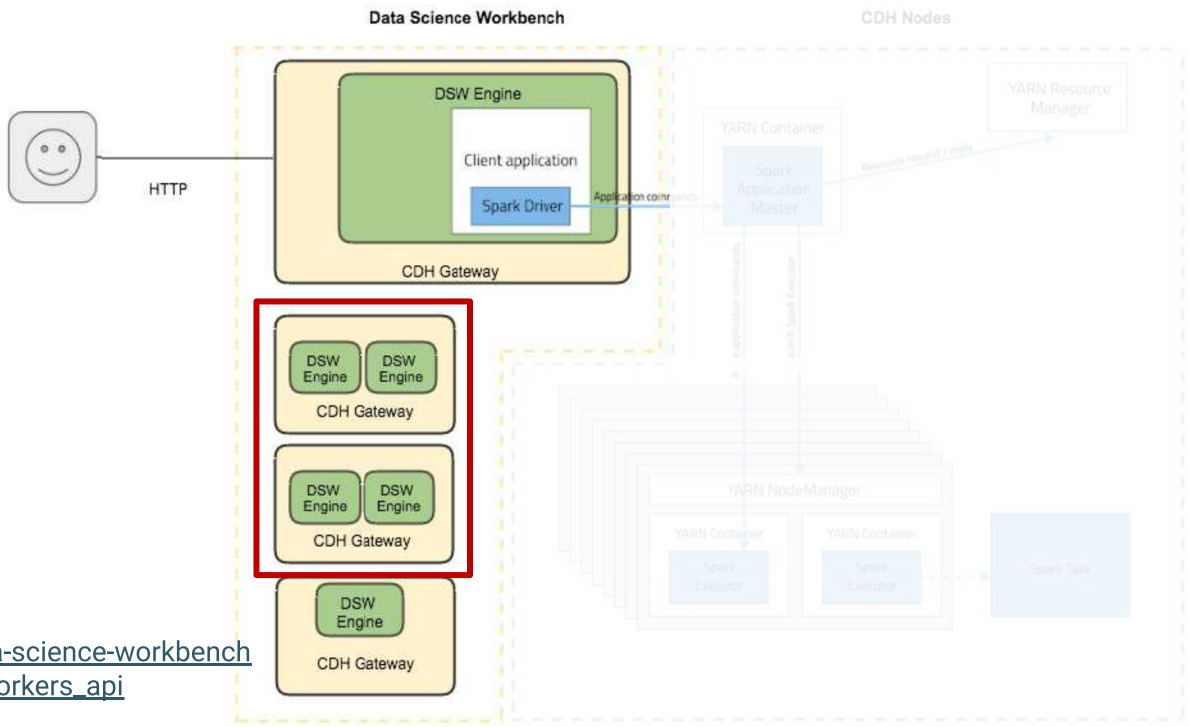
ADVANCED : DISTRIBUTED - ON CDSW

Code execution within the context of a session – with multiple workers

- Ability to launch multiple workers to distribute computing
- Automatic dependency management
- **Manual Data distribution**
- Some framework / distributed approaches may require extra configuration (ex : topology description)

Ideal for :

- Parallel Hyperparameter optimization
- Natively distributed frameworks : TensorFlow distributed, DASK, …



https://www.cloudera.com/documentation/data-science-workbench/1-6-x/topics/cdsw_parallel_computing.html#workers_api

DS / ML development guidelines

Tip 1 :
The right framework for the right data (size)

Not one shape fits all !!!

Use the right framework for the data size you are working with

Small — Medium — Large

Non distributed frameworks → Distributed frameworks

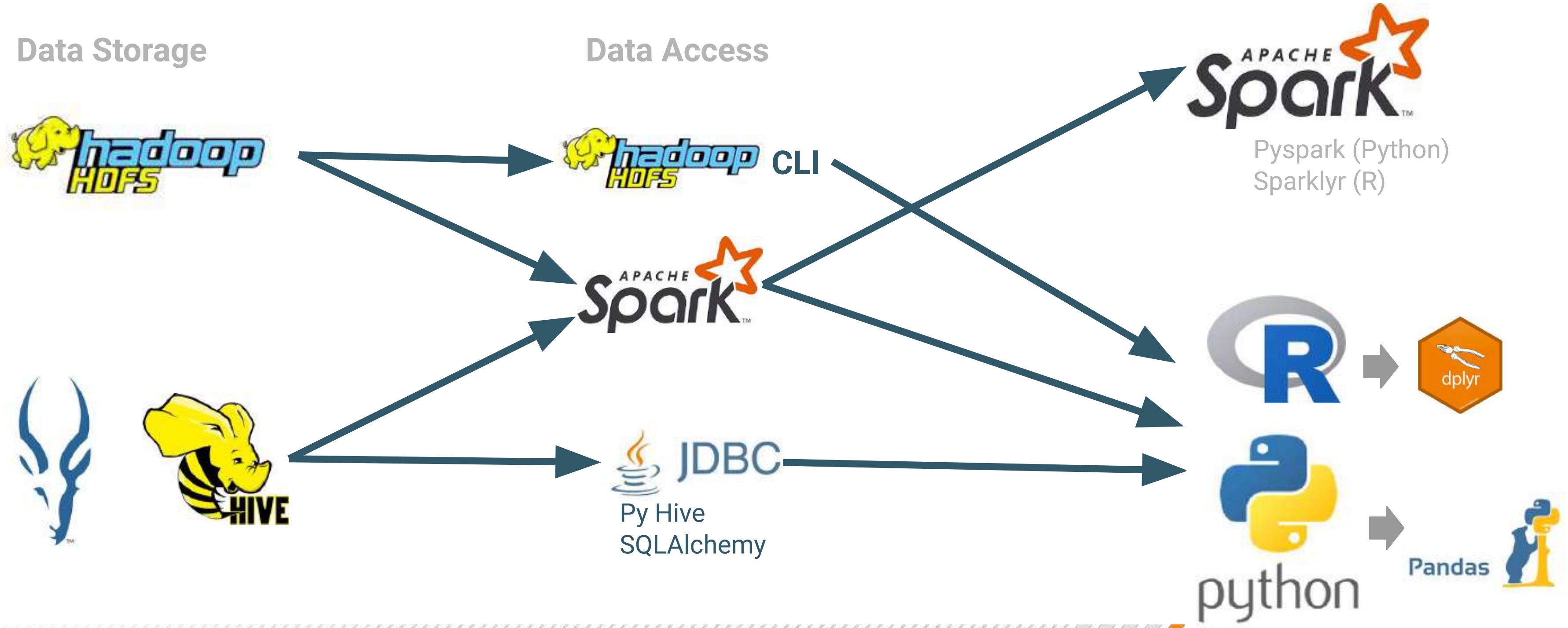
Data volume (rule of thumb)	Up to ~1 M lines Up to ~15 columns	~1M to ~50 M lines ~15 to ~50 columns	> 50 M lines > 50 Columns
Frameworks	Non distributed R, Pandas, scikit learn	Non distributed With optims R, scikit learn, Pandas Distributed Horovod, DASK, Spark	Horovod, DASK, Spark
Comments	**Data fit in memory and single machine perf OK** Distributed frameworks are often slower due to to coordination overhead	**Data fits in memory (barely) but performance lacks** **Non distributed**: Memory and Performance optimisations necessary **Distributed frameworks** can significantly increase performance of computationaly intensive tasks (ex training / hyper param tuning)	**Data does NOT fit in memory** Distributed frameworks required for distributed data management and computation

Tip 2 :
Use data where it is (don't copy it locally)

USE DATA WHERE IS IT STORED, don't copy it

Data access patterns in CDSW

Data Transformation



Lab 1

Git : https://github.com/frenchlam/CDSW_RW_Basics.git

Tip 3 :
How to right size a session

Session sizing - Clarifications - CPU

CPU :

- CPU represents min. CPU quota available
- CDSW does not impose strict max CPU usage quotas (documentation). Therefore a session may use more CPU cores - if they are available

Recommendation :

- Use low CPU - 1(light loads) to 4(heavy loads) - session definition

Session sizing - Clarifications - Memory

Memory :

- CDSW imposes strict memory quota on sessions
- Sessions have memory overhead (linux kernel, getty terminal, interpreter,...) of ~1GB
- Jupyter Notebook increase overhead by ~ 128 to 256 mb

Recommendations :

- Min. session : 2GB with workbench / 3GB with Jupyter notebook
- Sessions must be accurately sized according to data and processing need
- Rule of thumb : keep 20 to 40 % headroom to "play around"
- Libraries such as memory-profiler can help in that respect

Session size - defined by admin

Example

Engines Profiles

vCPU is expressed in fractional virtual cores and allows bursting. Memory is expressed in fractional GiB and is enforced by memory killer. GPU indicates the number of GPUs that need to be used by the engine. Configurations larger than the maximum allocatable CPU, memory and GPU per node will be unschedulable.

	Description	vCPU (burstable)	Memory (GiB)	Actions
Small	1 vCPU / 2 GiB Memory	1	2	Edit Delete
Medium	2 vCPU / 4 GiB Memory	2	4	Edit Delete
Large	2 vCPU / 8 GiB Memory	2	8	Edit Delete
XLarge	2 vCPU / 16 GiB Memory	2	16	Edit Delete
	1 vCPU (burstable), 1.75 GiB memory	1	1,75	Add

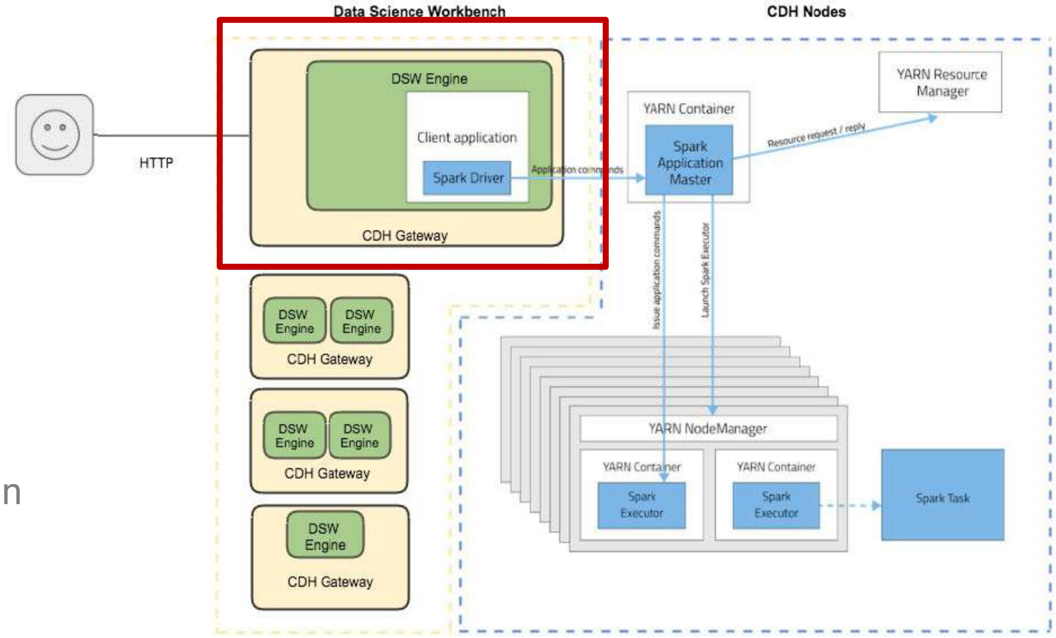
Session sizing - Clarifications - Spark

Only the Spark Driver runs in the CDSW session

- Requires limited amount of resources - if used right

Recommendation

- Only aggregated data should be retrieved locally
- All data transformation should stay on cluster / HDFS
- Open smallish session :
  - 1 CPU / 2 - 6GB mem



Tip 4 :
Memory optimisations tricks in Pandas

Pandas tip 1 : Read only the data you need

When working with larger datasets, try to limit the data loaded in memory :

For Wide datasets (many columns) :

- Always filter to use only columns you need

```
import pandas as pd
columns=["date", "loc", "x"]
df = pd.read_parquet("timeseries_wide.parquet")[columns]
# Warning: Entire dataset is read before filter
```

- Better - load only columns needed

```
# for CVS files use "usecols"
import pandas as pd
columns=["date", "loc", "x"]
df = pd.read_csv(csv_file ,
        header=0,
        index_col=["date", "loc"],
        usecols=columns,
        parse_dates=["date"])

# For columnar formats (orc / Parquet ), use columns option
df = pd.read_parquet(csv_file, columns=columns)
```

For long Datasets (many rows)

- Try working with samples first

- Try Chunking (rreaking up dataset into easily identifiable subsets) and work only on that

https://pandas.pydata.org/pandas-docs/stable/user_guide/scale.html#use-chunking

=> Applicable to simple transformation pipelines that are easily treatable independently

NOTE : If you get to that point, consider using distributed frameworks such as Spark

Pandas tip 2 : Optimize data types

Default datatypes in Pandas are **not** memory efficient

Try downcasting to numeric types and save (a lot) of memory

Numeric :

- Float - default dtype -> float64
  Can be downcast to float32
  Ex :
  s = pd.Series(['1.0', '2', -3])
  pd.to_numeric(s, downcast='float')

- INT - default dtype -> int64
  Can de downcast to :
  - signed - dtype int8 -> All signed integers
  - unsigned -dtype uint8 -> Smallest footprint

Text :

- All categorical columns can be cast to categorical

  -> Essentially a dictionary of values

  Ex :

```
df2['name'] = df2['name'].astype('category')
```

https://pandas.pydata.org/pandas-docs/stable/user_guide/scale.html#use-efficient-datatypes

Pandas tip 3 : Clean up after yourself

1. Delete all unused variable (and dataframe copies in particular) ex :

```
df = pd.read_parquet("timeseries_wide.parquet")
columns=["date", "loc", "x"]
df_filtered = df[columns]
# remove inital copy
del df
```

2. (use with ***CAUTION***) - use operations with "inplace"

```
df = pd.read_csv('some'file')
df.sort_values(inplace=True)
=> 2x mem for for comp but 1x for result
```

```
df = pd.read_csv('some'file')
df = df.sort_values()
=> 2x mem for comp, and 2x mem footprint (copy of original object )
```

```
Note : another approach for the above statement is could be to use method chaining
df = pd.read_csv('some'file').sort_values()
```

https://docs.python.org/3.6/tutorial/datastructures.html?highlight=del#the-del-statement

Lab 2

Git : https://github.com/frenchlam/pandas_mem_tips.git

Selected resources

Speed

- From Pandas documentation : https://pandas.pydata.org/pandas-docs/stable/user_guide/enhancingperf.html
- From PyCon https://speakerdeck.com/pycon2017/sofia-heisler-no-more-sad-pandas-optimizing-pandas-code-for-speed-and-efficiency

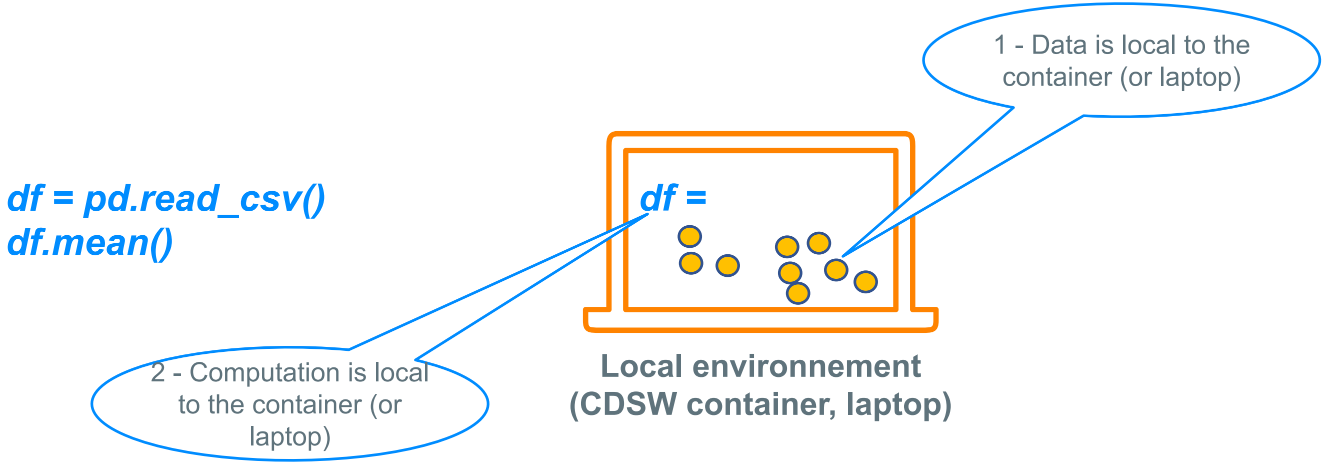
Memory efficiency

- From Pandas documentation (scale-out) : https://pandas.pydata.org/pandas-docs/stable/user_guide/scale.html
- From Pandas documentation (Sparse data) : https://pandas.pydata.org/pandas-docs/stable/user_guide/sparse.html

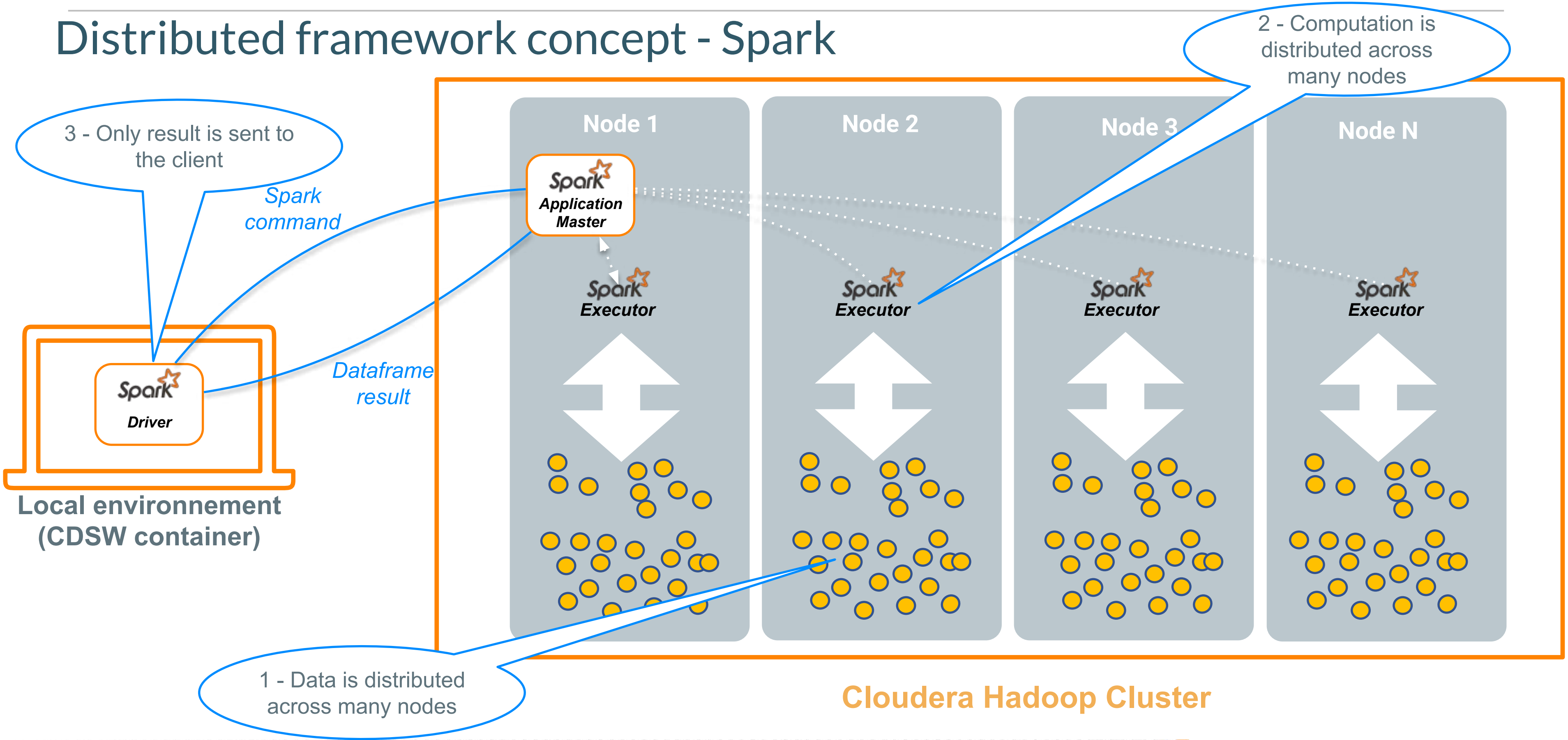
Tip 5 :
Spark concepts

Non-distributed framework concept

Python (Pandas, Scikit, …) / R / (also Spark in local mode)

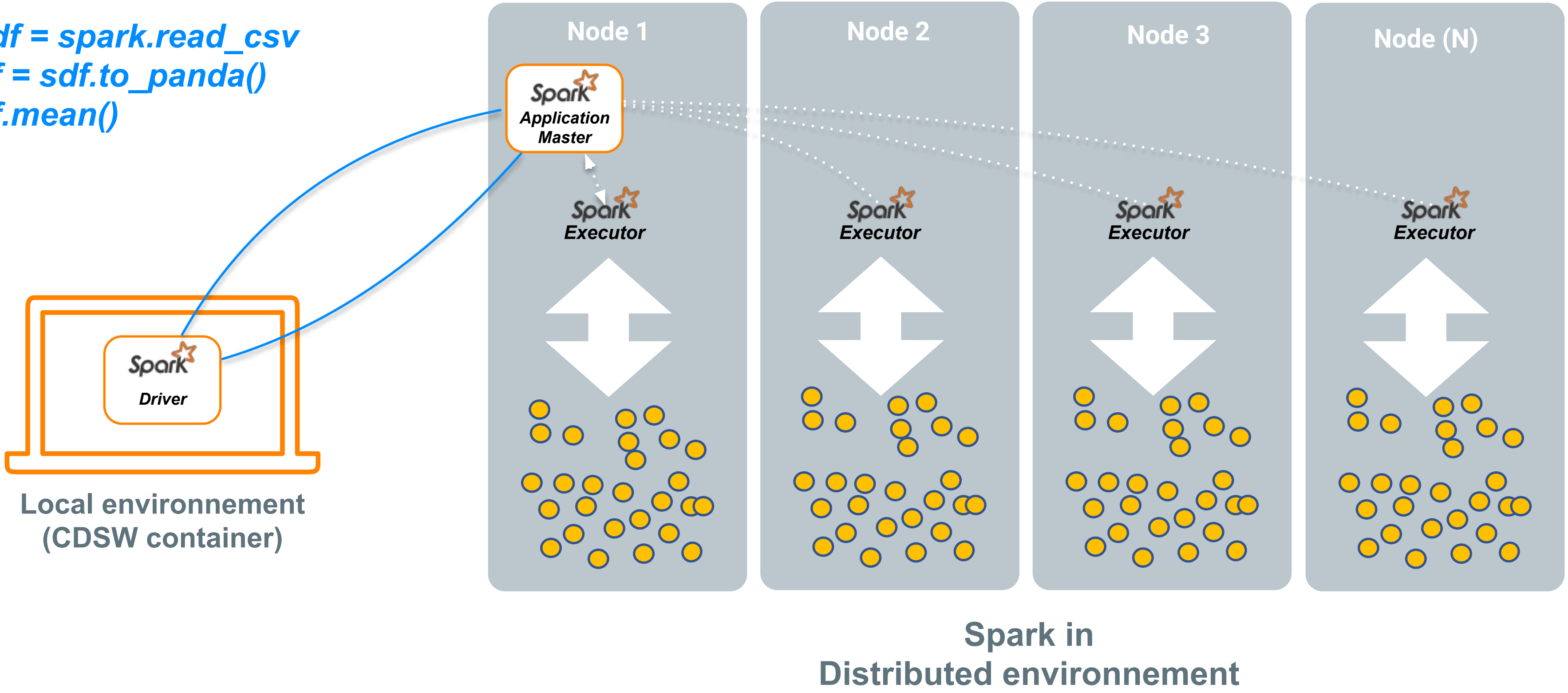


Distributed framework concept - Spark



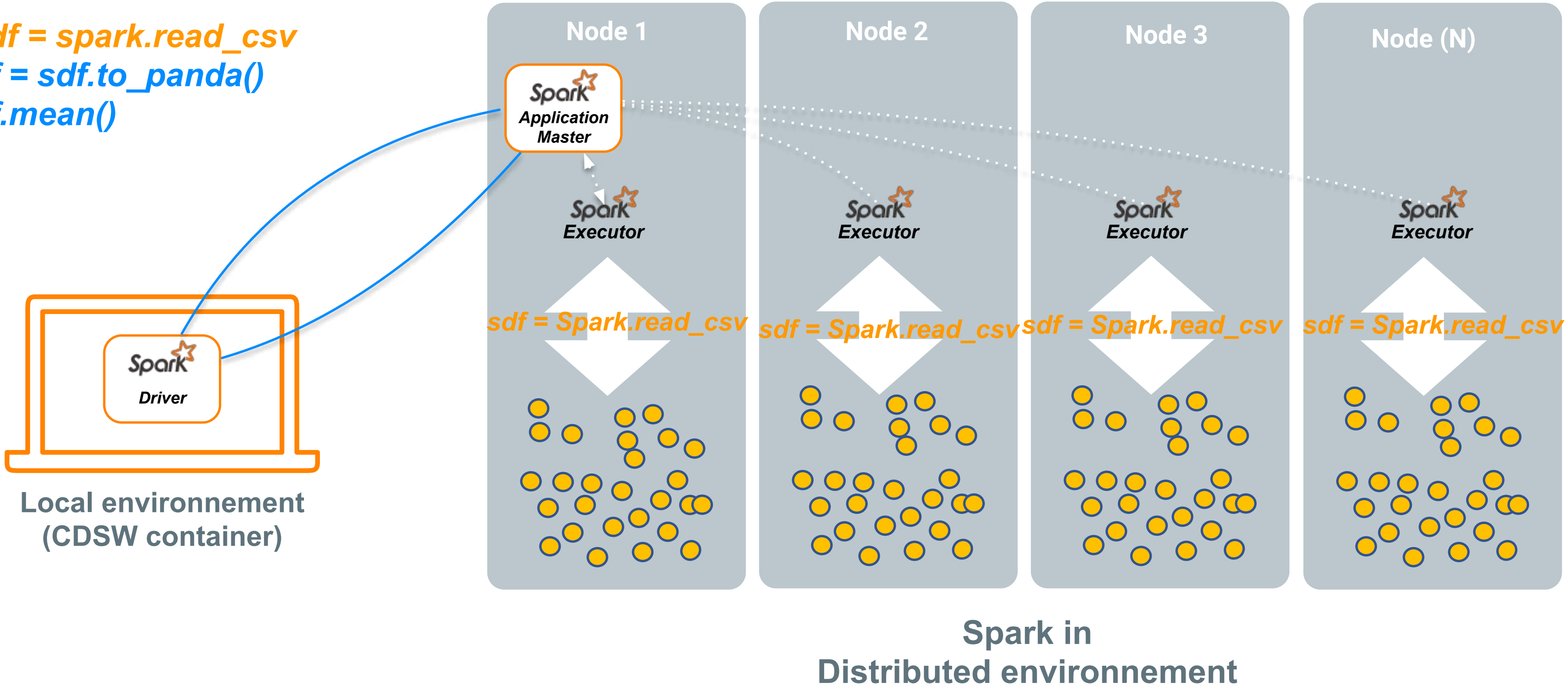
SPARK common mistake

sdf = spark.read_csv
df = sdf.to_panda()
df.mean()



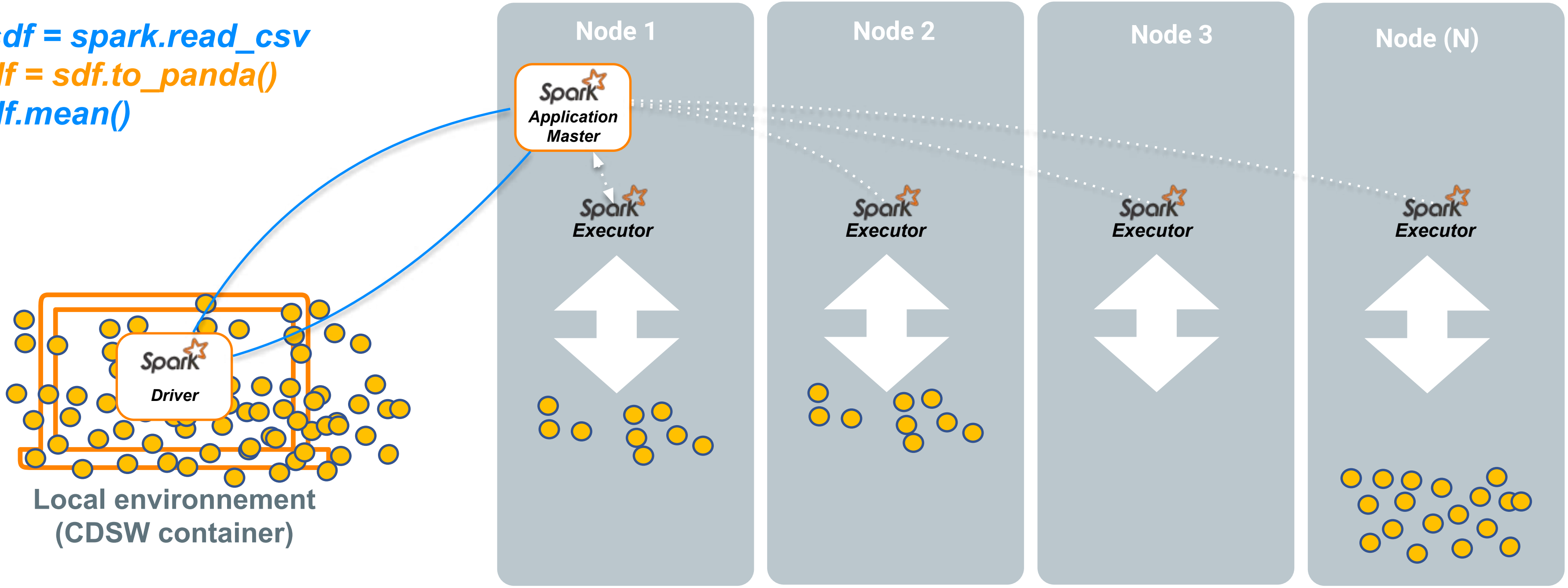
SPARK common mistake

sdf = spark.read_csv
df = sdf.to_panda()
df.mean()



SPARK common mistake

sdf = spark.read_csv
df = sdf.to_panda()
df.mean()

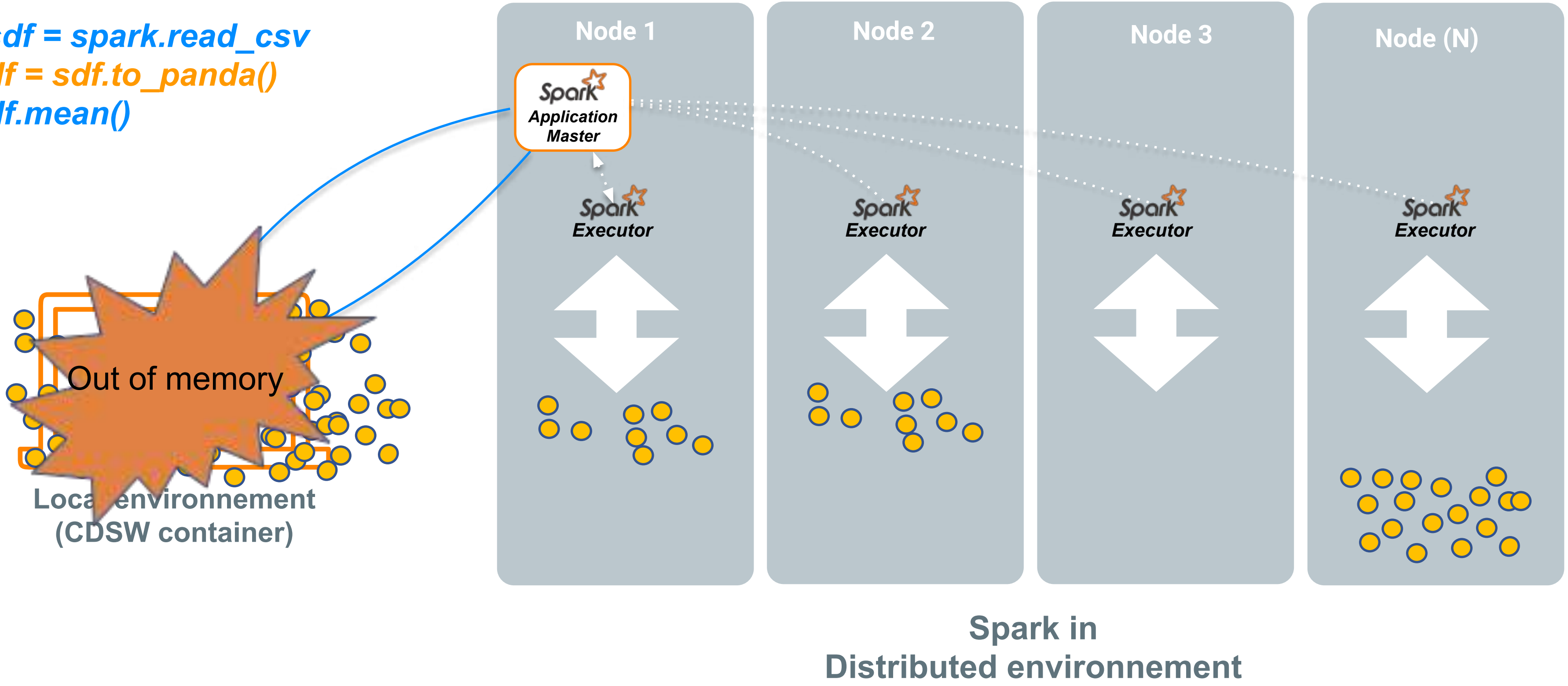


Local environnement (CDSW container)

Spark in Distributed environnement

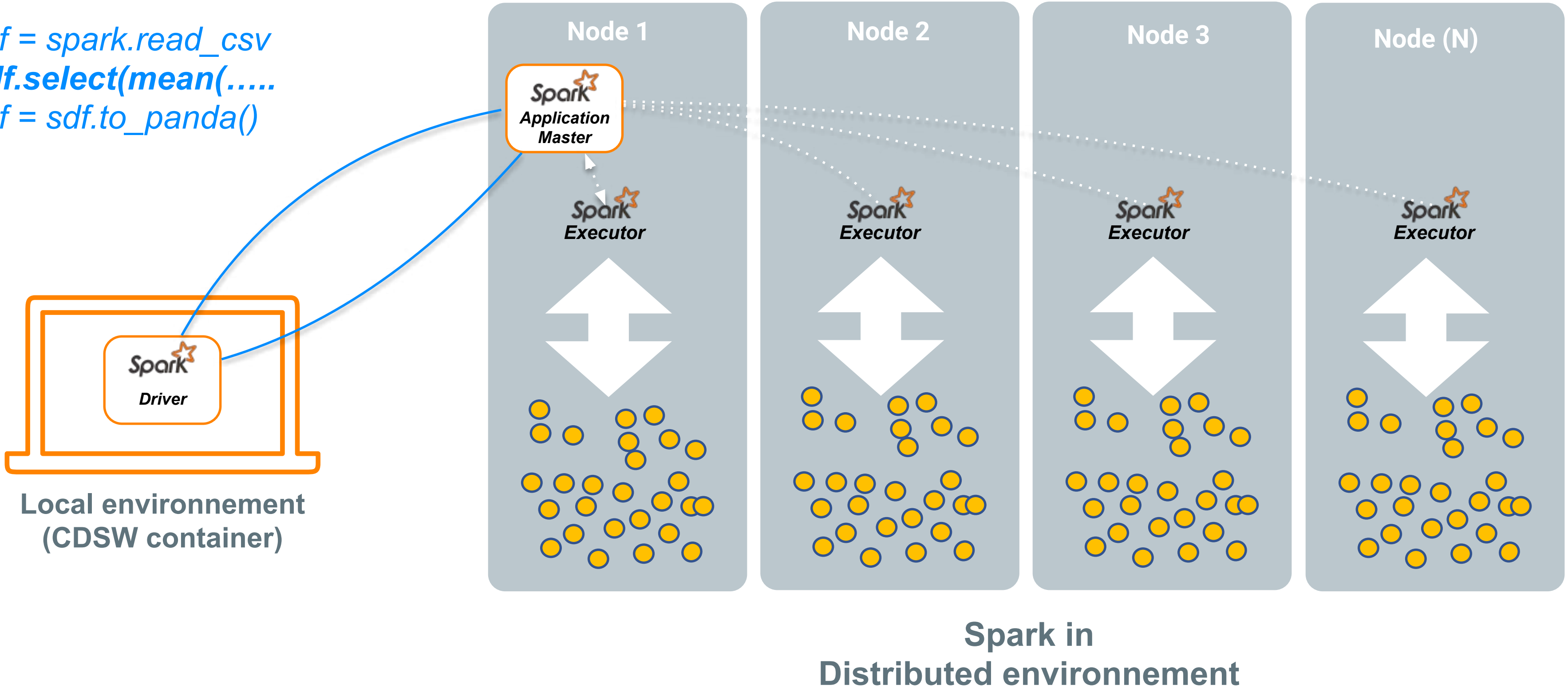
SPARK common mistake

sdf = spark.read_csv
df = sdf.to_panda()
df.mean()



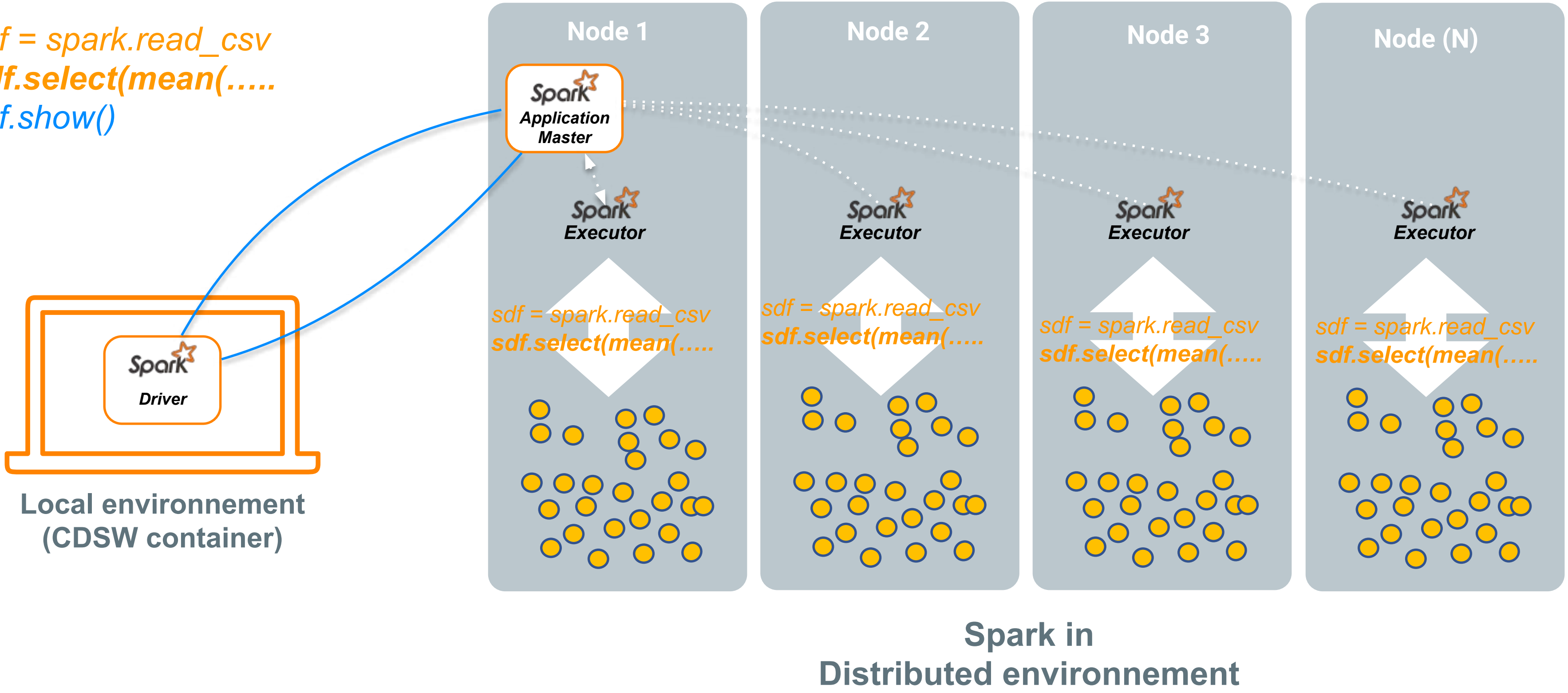
SPARK The good way

sdf = spark.read_csv
sdf.select(mean(.....
sdf = sdf.to_panda()



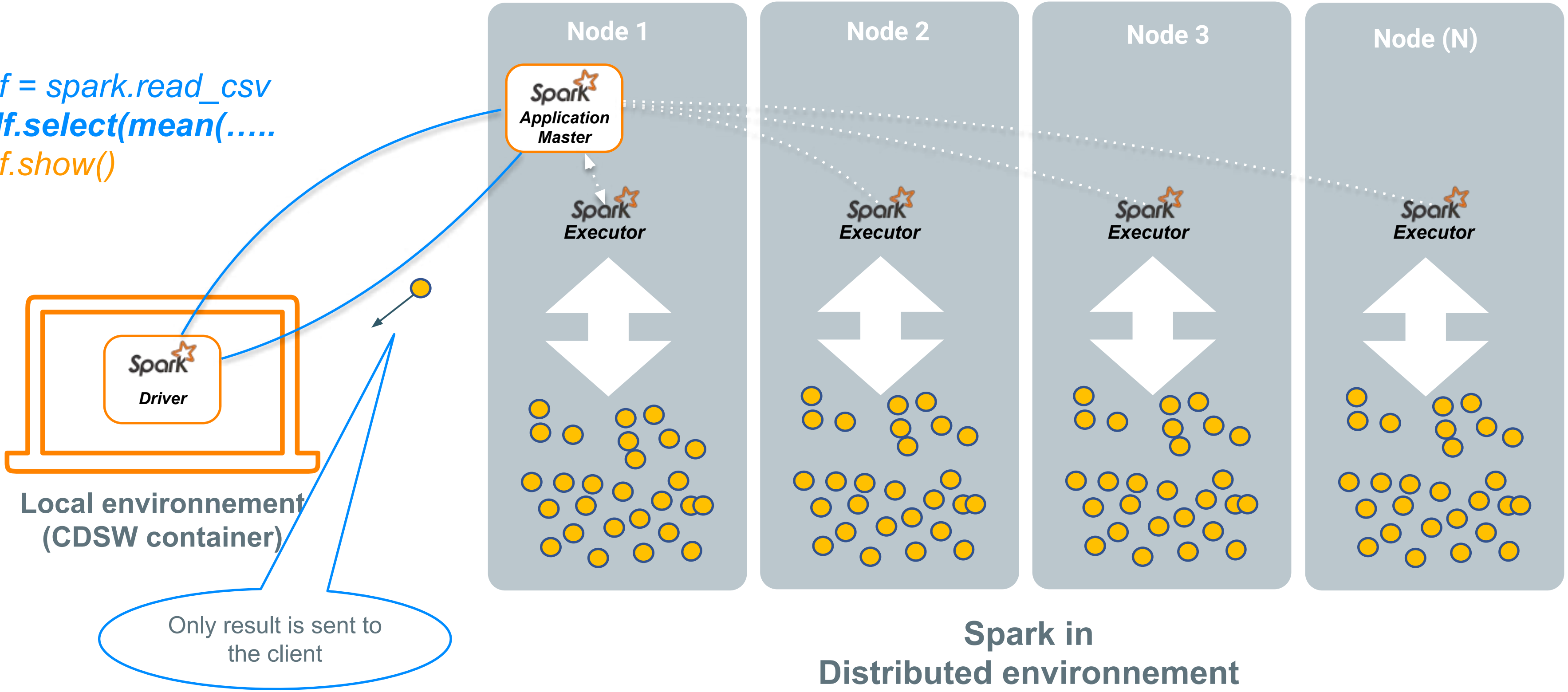
SPARK The good way

sdf = spark.read_csv
sdf.select(mean(.....
sdf.show()



SPARK The good way

sdf = spark.read_csv
sdf.select(mean(…..
sdf.show()



Recommendations

- Use native spark functions
- Only bring aggregated data back -> perform all computations on the cluster
- Be careful with dependencies (especially with pyspark)
- Understand and if possible plan the dataflow
  - Be careful with shuffles (groupBy, sorts, …)
  - Be careful with joins
- Be aware of data skews
- Cartesians products **really** don't play well with big data

Resources and References

Spark

http://blog.cloudera.com/blog/2015/03/how-to-tune-your-apache-spark-jobs-part-1/

http://blog.cloudera.com/blog/2015/03/how-to-tune-your-apache-spark-jobs-part-2/

http://www.slideshare.net/hkarau/presentations

http://spark.apache.org/docs/latest/tuning.html

R

https://www.rdocumentation.org/packages/sparklyr/versions/1.0.4

https://rdrr.io/cran/sparklyr/api/

https://spark.rstudio.com/

PySpark

https://spark.apache.org/docs/latest/api/python/

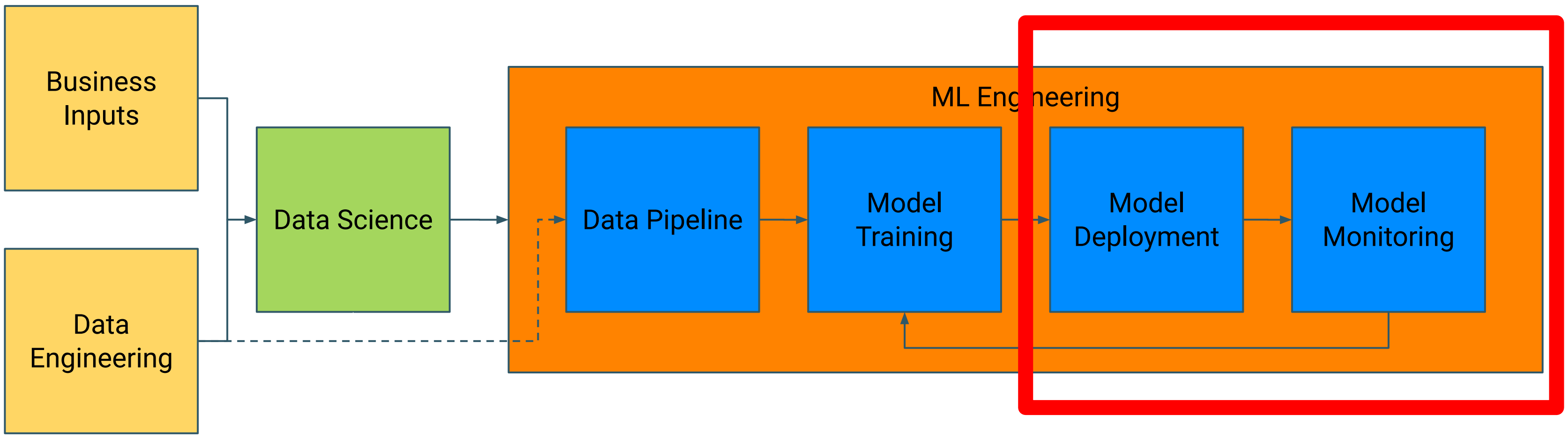
Workshop (examples of Data Engineering and ML using Spark)

https://github.com/fletchjeff/ml_at_scale_workshop

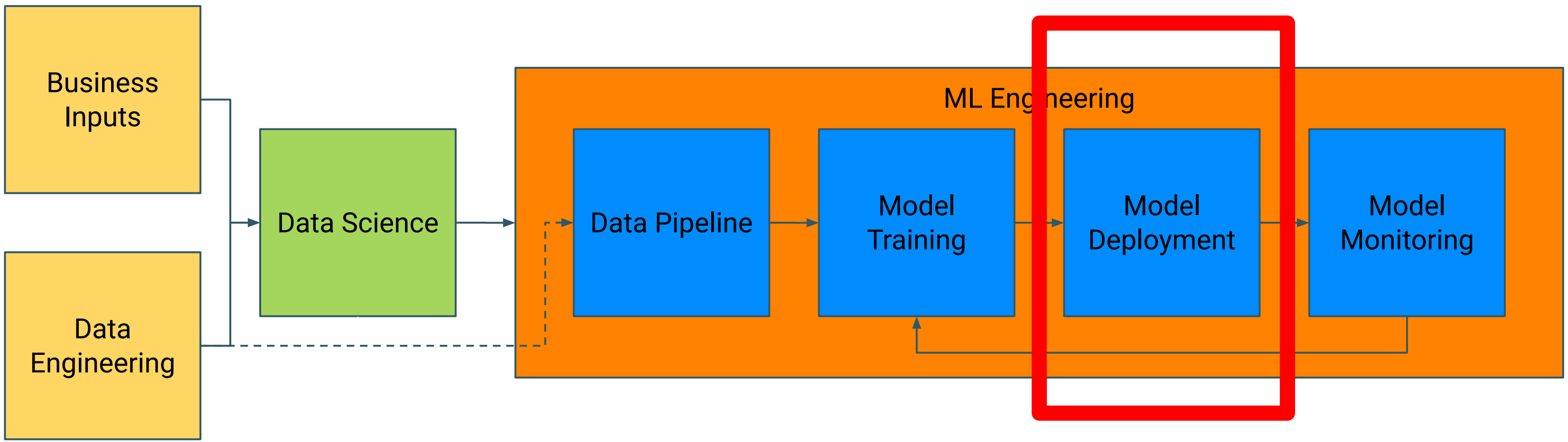
PAUSE

Deployment guidelines

MACHINE LEARNING MODEL DEPLOYMENT & OPERATIONS



MACHINE LEARNING MODEL DEPLOYMENT



MODELS API

Function as a Service FaaS REST API (More than just Model)

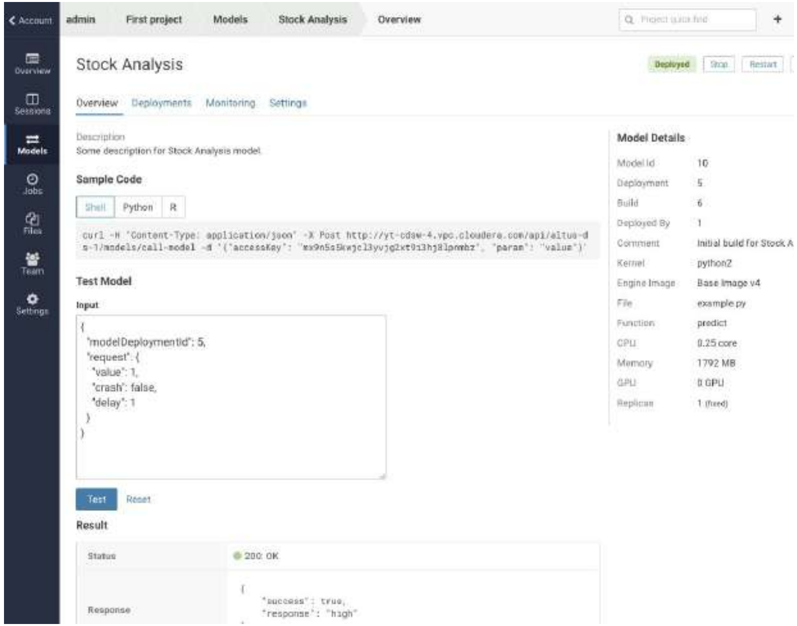
1. Choose file, e.g. `score.py`
2. Choose **function**, e.g. `forecast`

```
f = open('model.pk', 'rb')
model = pickle.load(f)

def forecast(data):
  return model.predict(data)
```

3. Choose resources
4. Deploy!

Containers also have access to CDH/HDP for data lookups.



HOW IT WORKS - Source 2 Image (S2I)

Experiments and Models leverage a new way of building images from source

- When running an **experiment** or **deploying** a model:

SOURCE

- Stage 1: Git snapshot of source, respecting `.gitignore` (before – to be sure to ignore any local Python/R environment!)

IMAGE

- Stage 2: Docker build from source; `cdsw-build.sh` defines build steps, e.g.:

```
#!/bin/bash
pip3 install -r requirements.txt
```

RUN

- Stage 3: Run versioned image as Experiment (batch) or Model (online) in Kubernetes

- Provides a declarative pathway from version control to experiment or model
- Sets the stage for deployment from other environments, e.g. laptops, webhooks

MODEL BUILD - Best Practices

Best practices for cdsw-build.sh scripts

The script should include :

1. All dependencies :

Python : PIP requirements

R : Setup script

```r
install.packages("sparklyr")

#list.of.packages <- c("sparklyr","jsonlite", dependencies=TRUE)
#new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,'Package'])]
#if (length(new.packages)) {
#   install.packages(new.packages, repos='https://cran.revolutionanalytics.com/')
#}
```

```
### Replacer avec l'adresse de vos repos python ###
--index-url https://pypi.org/simple/

### necessaire si le serveur n'utilise pas TLS ###
--trusted-host https://pypi.org/

#### libraries ####
#### Mettre le chemin absolut des fichiers si installation locale ####
/home/cdsw/cdsw_test/FrenchLefffLemmatizer.zip
joblib
scikit-learn==0.21.2
lightgbm==2.2.3
imbalanced-learn==0.5.0
```

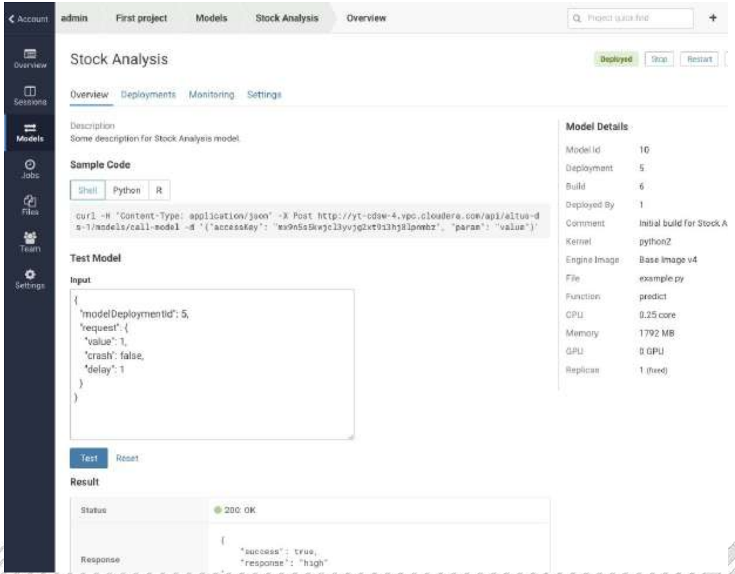
MODEL BUILD - Best Practices

Best practices for cdsw-build.sh scripts

The script should include :

If stored in remote location (HDFS / S3) :
=> **Should be copied locally** (to ensure versioning)

2. All artefacts
(for models)



MODEL BUILD GOTCHAS

Known limitations

Git snapshot cannot handle artefacts larger than 50mb

Work-around :

- Add artefact to .gitignore file
- Copy explicitly in cdsw-build.sh script

https://docs.cloudera.com/documentation/data-science-workbench/1-6-x/topics/cdsw_engines_models_experiments.html#snapshot

Do not create a git repo INSIDE a project (cf. DSE-4657)

Work-around :

- rename the **.git** directory (for example, NO.git) and re-build the model.

If model is stuck in deployment, check that :

- CDWS script ran correctly
  => Builds tab
- Model / Function initialises correctly
  => Monitoring tab
- Check that CDSW cluster has sufficient resources
  (no error message)

```
Error: 2 UNKNOWN: Unable to schedule build: [Unable to create a checkpoint of current source: [Unable to push sources
to git server: ...
```

Lab 3

Git : https://github.com/frenchlam/wine_pred_CDSW.git

APPLICATION DEPLOYMENT

Deploy interactive application via Sessions or Jobs

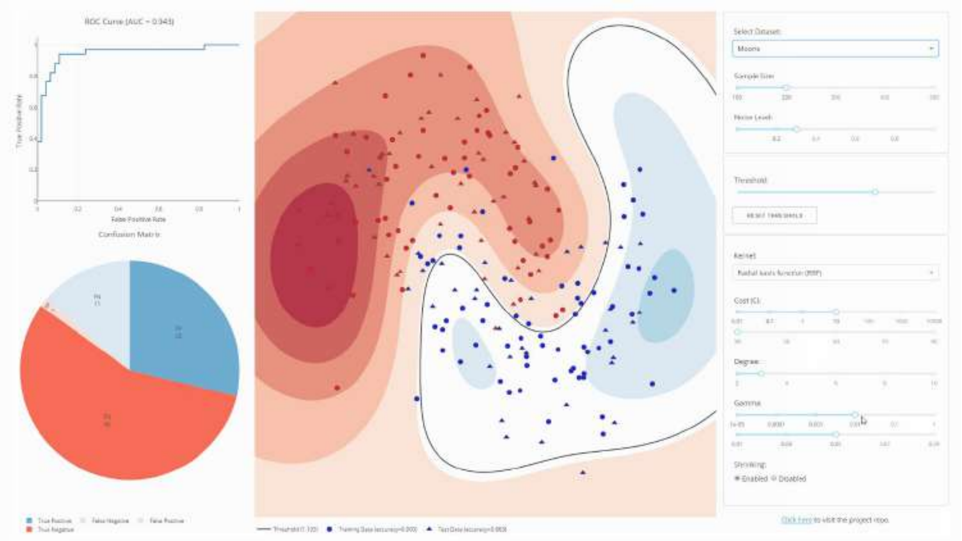
- CDSW can run any type of arbitrary code in Scala, Python or R
- Different ports are exposed through sessions or Jobs to expose web apps such as
  - Shiny or
  - Dash
- App are deployed and exposed via the CDSW infrastructure

test_app Running

By CDEP CREATED ACCOUNT — Python 3 Session — 2 vCPU / 4 GiB Memory — just now See job details

Session Logs Collapse Share

```
> import os
> engine_id = os.environ.get('CDSW_ENGINE_ID')
> cdsw_domain = os.environ.get('CDSW_DOMAIN')
> print("app running at http://{}.{}".format(engine_id,cdsw_domain))
  app running at http://90j7dethdwyno58q.sfrmlamairesse-4.vpc.cloudera.com
> !python3 app.py
```

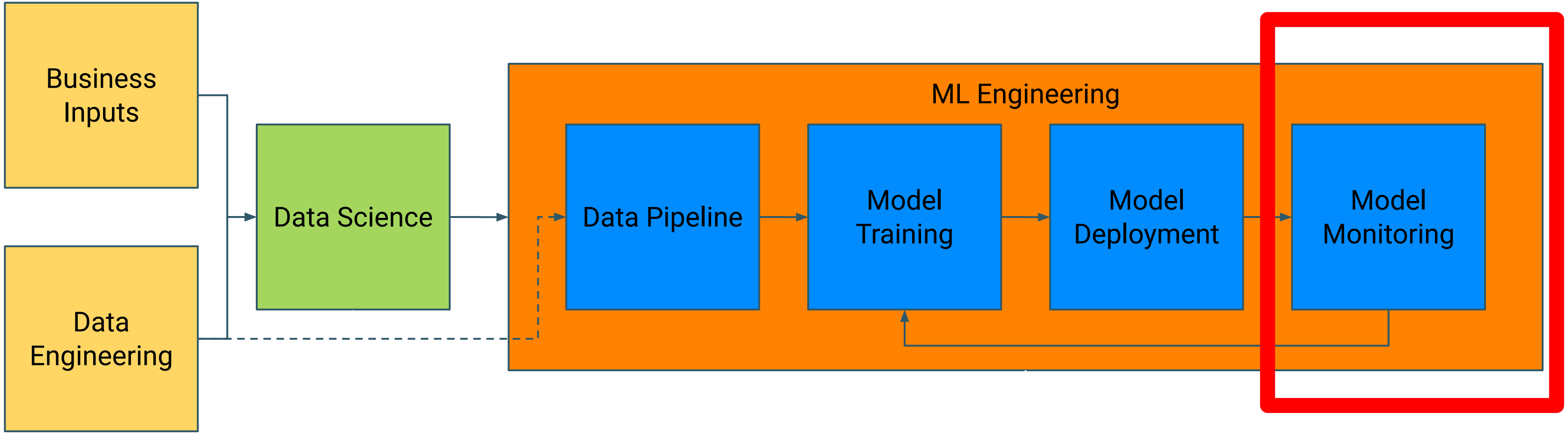


CLOUDERA

Lab 4

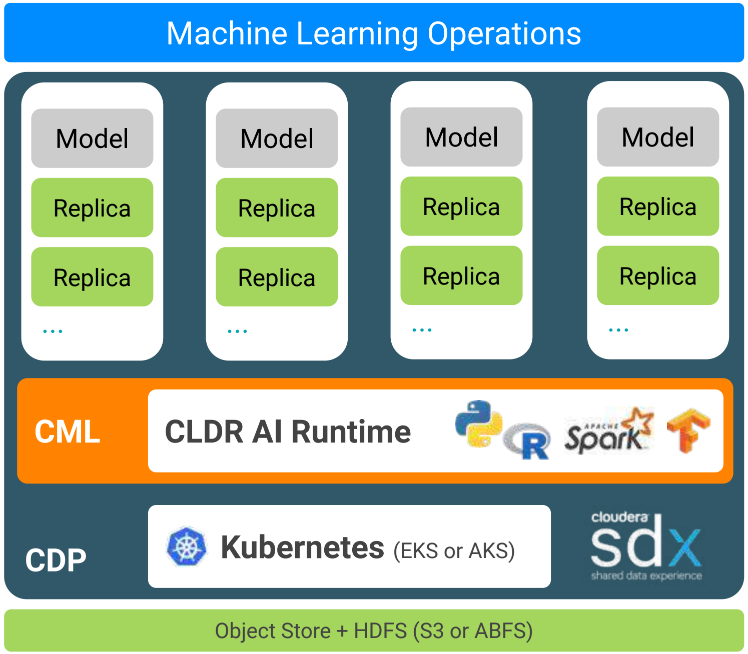
Git : https://github.com/frenchlam/dash-svm.git

MACHINE LEARNING MODEL OPERATIONS



To Come H1/2020

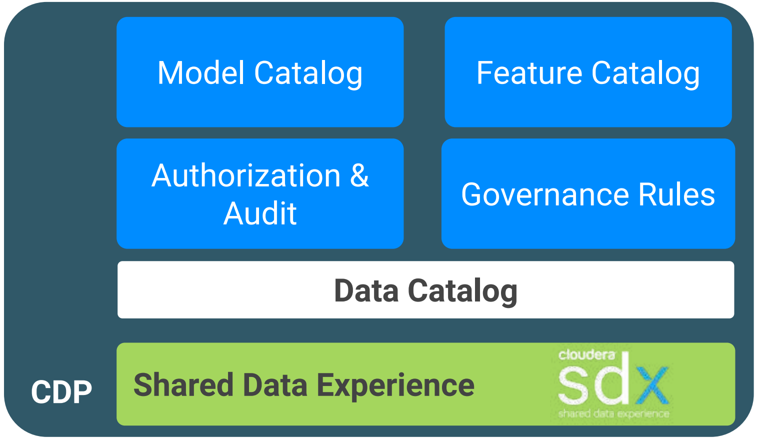
A look ahead - MLOPS: Operating models at enterprise scale



- **Centralized Monitoring Solution**
  - Single pane of glass for all models
  - Alerting and external system integration
- **Track technical metrics**
  - Uptime
  - Status
  - SLA adherence
- **Track mathematical metrics**
  - E.g. prediction distribution, drift, input distribution
  - Customizable to model

A look ahead - MLOPS: Governing hundreds of models

- **Centralized Catalog**
  - Track all models along their lifecycle
  - Track all features their lifecycle
  - Understand model and feature relationships
  - Understand features and their relationship with the data catalog (i.e. lineage)
- **Authorization & Audit**
  - Protect models
  - Track access



A look ahead - MLOPS: Feature overview

Example only.

Investments in active research and development

Feature Store

Best practices and support for storing, sharing, and using ML features, outcomes, and predictions.

```
x = load("customer_features")
y = load("customer_churn")
model = train(y, x)
p = model.predict(x)
store("customer_pchurn", p)
```

Job/Model Deploy

Production pipeline orchestration, scheduling, and monitoring, with tight integration to ML-X runtime (R, Python, Spark).

```
cldr run experiment “foo.py”
cldr flow deploy
cldr model deploy
```

Model Store

Stores for model artifacts (e.g. container, serialized model, diagnostics, and metrics) and ongoing monitoring updates.

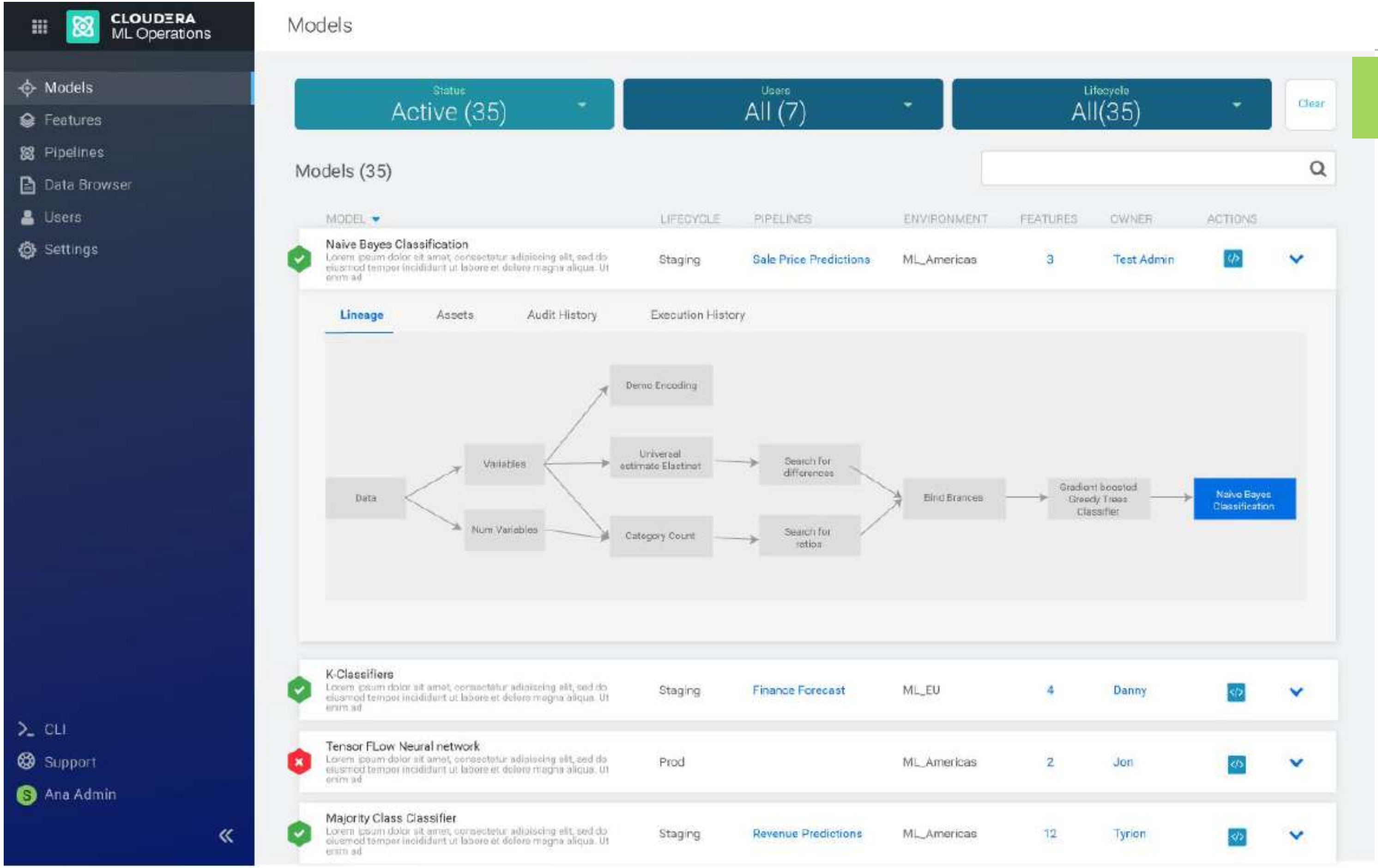
```
m = ModelStore(“churn”)
m.store(model)
m.track_metric(auc)
```

Model Operations

Monitor and maintain models, anywhere (think: WXM/APM for models). Track drift, manage risk, run experiments, collect data

```
def predict(x):
  p = model.predict(x)
  cldr.track(x, p)
  return p
```

...and resulting actions...



Example only.

Other roadmap items

MACHINE LEARNING ROADMAP

Accelerating our roadmap to support the industrialization of AI

WHAT ARE WE TRYING TO DO?	Enable data scientists	On a common platform	For the real world
WHAT ARE THE CHALLENGES?	Users: Developer friction IT: Security, governance	Monolithic, inelastic architecture with complex dependency management	Limited management for models in production
SO WHAT ARE WE DOING?	Better core experience	Evolve the platform	Production ML

Better core experience to enable data Scientist

Roadmap for ML Runtimes

SPECIALIZED ML RUNTIMES FOR COMMON USE-CASES

Reduced size of container images — better agility and performance in autoscaling clusters

Improved security — smaller surface area for security vulnerabilities

Simplified architecture — enable deeper level of customizations

VERSATILE & NATIVE CUSTOM EDITOR SUPPORT

Streamlined IDE provisioning — Optimized dev. Workflows for data science team productivity

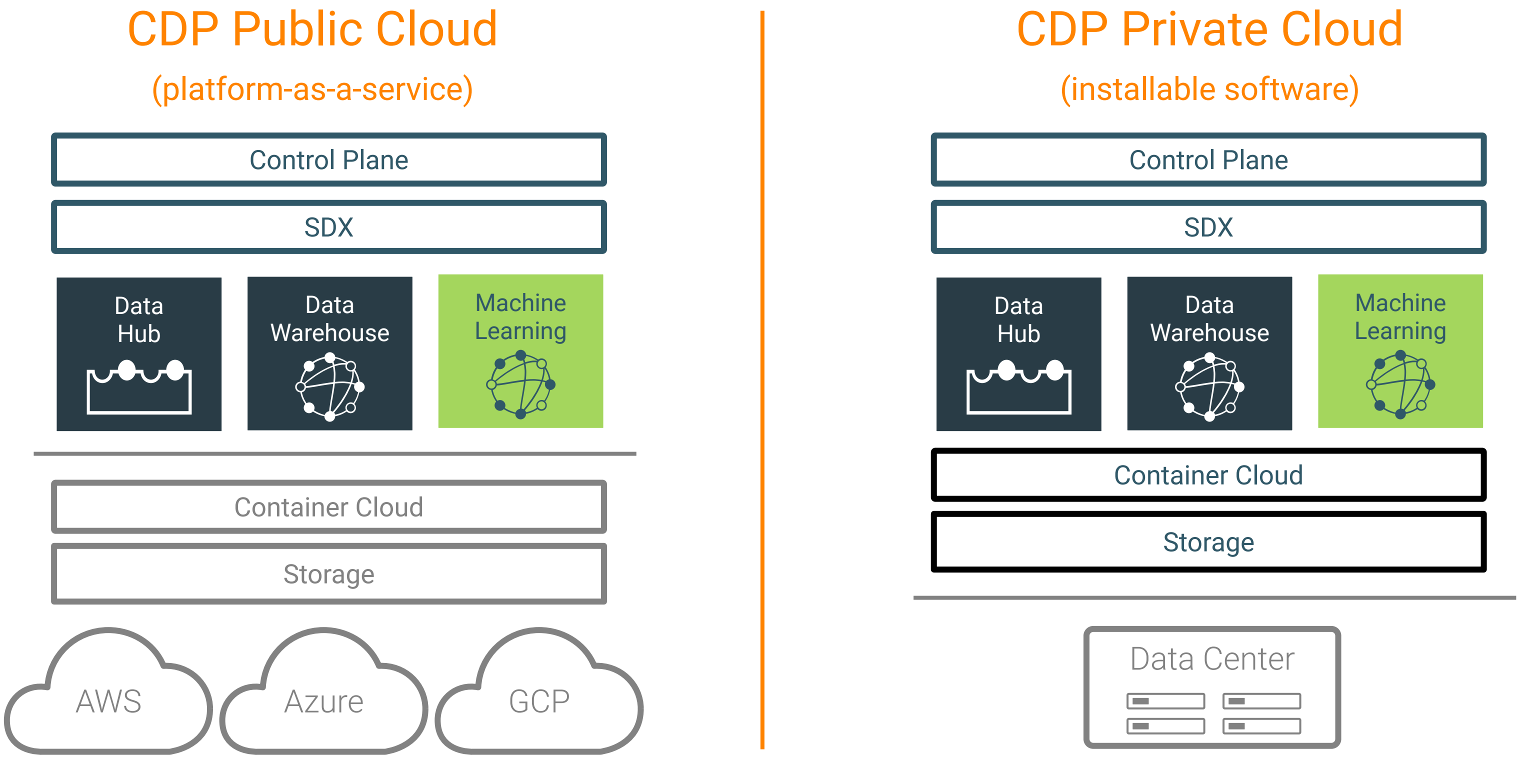
OOTB alternatives for the Workbench experience — support for JupyterLab and Zeppelin

ENABLE SELF-SERVICE RUNTIME CUSTOMIZATION

Secure root level Runtime customization for data scientists — no need for IT assistance

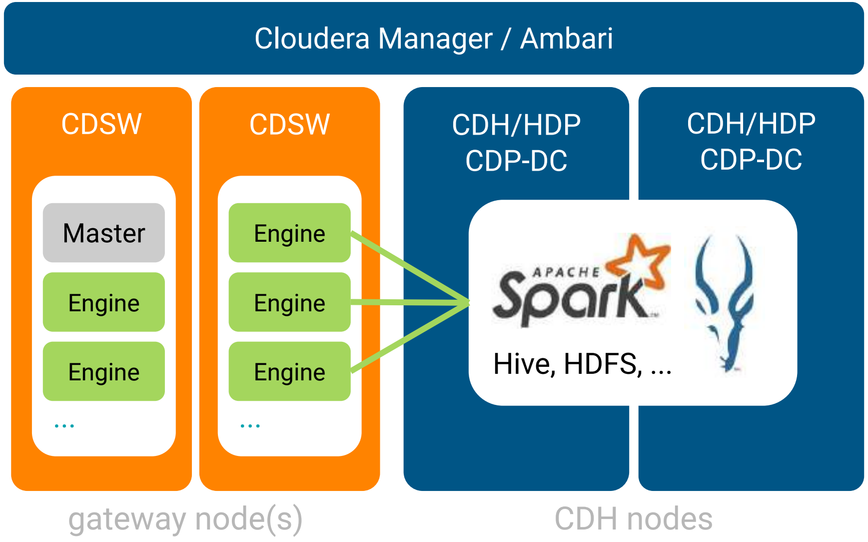
Support environment sharing between projects — enable data scientist collaboration workflows

Evolve the Platform to make it run anywhere



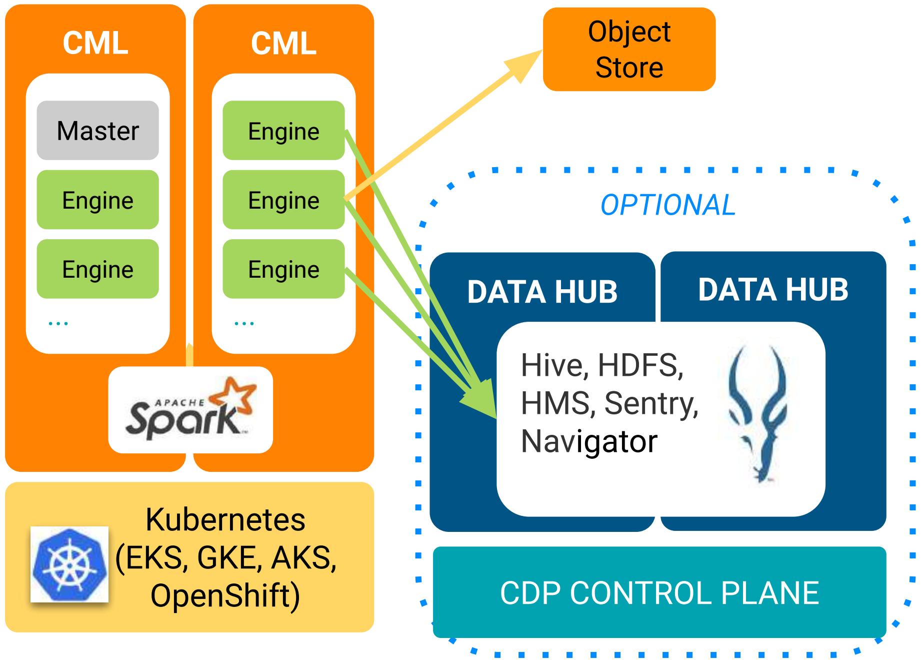
CDSW - SPARK ON YARN

TWO FORM FACTORS, ONE USAGE



Requires and extends CDH/HDP/CDP-DC, pushing distributed compute to the cluster

CML - SPARK ON KUBERNETES



Self-contained and manages own distributed compute; can optionally use CDH/HDP

Production ML to facilitate the industrialisation of use cases

Roadmap for Model Operations

OPERATIONS

- **Centralized Monitoring Solution**
  - SDK & UI
  - Alerting and external system integration
- **Track technical metrics**
  - Uptime
  - Status
  - SLA adherence
- **Track business metrics**
  - Input/Output Tracking
  - Ground Truth Matching
  - Customizable to model

DEPLOYMENT AND SERVING

- **Enterprise Capabilities**
  - Highly Available
  - Autoscaling
  - Secure by default (strong Auth)
- **Models & Jobs CLI**
  - Programmatically deploy models

GOVERNANCE

- **Model Catalog**
  - Metadata and lineage for models
  - Supports custom metadata
  - Links to data lineage for tracking source data used in training

NEXT STEPS

Consider training - specifically on SPARK

Broad portfolio across multiple platforms

ADMINISTRATOR	**Administrator** CDH | HDP	**Security** CDH | HDP	**AWS Fundamentals for CDP**	**Cloudera on MS Azure***				**CCA Administrator** **HDPC Administrator**
DATA ANALYST	**Data Analyst** CDH	**Hive3** HDP	**Kudu** CDH	**SQL at Scale*** Coursera	**CDW** CDP			**CCA Data Analyst**
DEVELOPER & DATA ENGINEER	**Spark** CDH | HDP	**Spark Application Performance Tuning** CDH	**NiFi** CDF	**Kafka Operations** CDH	**Search | Solr** CDH	**Architecture Workshop** CDH	**HBase** CDH	**CCA Developer** **CCP Data Engineer**
DATA SCIENTIST	**Data Science** CDSW | HDP	**Cloudera DS Workbench** CDSW		**CML** CDP				

Private Public Class OnDemand OnDemand (coming soon)

Consider PS to help onboard teams on CML

SMARTML

Cloudera's ML services for the journey to industrialized AI

	Strategy/Foundation	Develop	Deploy	Scale
PLATFORM & TOOLS	PLATFORM & TOOLS EDC + CDSW/CML			
EXPERT GUIDANCE	ML STRATEGY ENGAGEMENT Strategy prescription for use cases, organizational and process design ON DEMAND CDSW TRAINING		ADVISING & RESEARCH White glove concierge service including ML expert & SW/platform advising plus access to CFFL research reports and prototypes. RESIDENT DATA SCIENTIST CFFL expert onsite with your teams 1 week/month	
SERVICES FOR PRODUCTION & SCALE	ML PLATFORM ENABLEMENT Deploy and enable CDSW in your environment per best practices	ML APP DEVELOPMENT Identify, build, and document an ML app in your environment	ML APP DEPLOY Deploy existing ML app to an existing Cloudera production environment ML OPERATIONS Apply operational tooling, automated processes, and best practices to maintain production ML models	ML APP SCALE Increase data and/or performance for existing production ML app

SMARTML: ML PLATFORM ENABLEMENT

Deploys and enables CDSW in customer's environment per best practices

This engagement lays the foundations for customer success with CDSW by addressing both the technical and non-technical aspects of CDSW adoption for platform teams and end users. Installation and configuration of CDSW is just one (optional) aspect. Key focus areas for end users include best practices for dependency management, collaboration and source control and patterns for bringing their own data.

Services	• Install and configure CDSW; KT with platform team • Dependency management (custom engines, parcels etc.) • Enable use of GPUs	• Integrate with third-party editors • KT with DS/ML teams on customer's ML platform • Migrate projects from another data science tool
Outcomes	• Platform team enabled to provide and support an effective machine learning and data science environment • Data science team enabled to use CDSW and wider Cloudera platform most effectively in their unique environment	

Estimated Effort	Estimated Pricing
2-4 weeks for typical engagements; longer for complex migrations	Time & materials

Customer Benefits	• Reduce time investment of customer platform team by leveraging prior Cloudera experience • Reduce time to productive adoption of CDSW by DS/ML teams • Ensure administration and use of CDSW complies with best practices for most effective use of the platform
Assumptions	• Customer has provisioned, installed and networked all hardware (including GPUs if applicable) • Customer has completed prerequisites as documented in "ML Platform Enablement Engagement Prerequisites" • DS/ML teams have received CDSW training prior to knowledge transfer sessions

EDUCATION SERVICES OFFERINGS

Flexible delivery options catering to all enterprises

PRIVATE

- ✓ Role-based Training
- ✓ Tailoring
- ✓ Customization
- ✓ Onsite or Virtual
- ✓ Flexible dates

LIVE

PUBLIC

- ✓ Role-based Training
- ✓ Published Schedule
- ✓ Global coverage
- ✓ Multi-time zones
- ✓ Training facility or Virtual

LIVE

ONDEMAND

- ✓ OnDemand Lab Access
- ✓ 100% self-paced
- ✓ High quality pre-recorded video & audio instruction

CERTIFICATION

- ✓ Performance-based hands-on exams
- ✓ Role-based
- ✓ Live proctor
- ✓ Schedule exam 24/7
- ✓ Digital badge

TRAINING CREDITS

edu

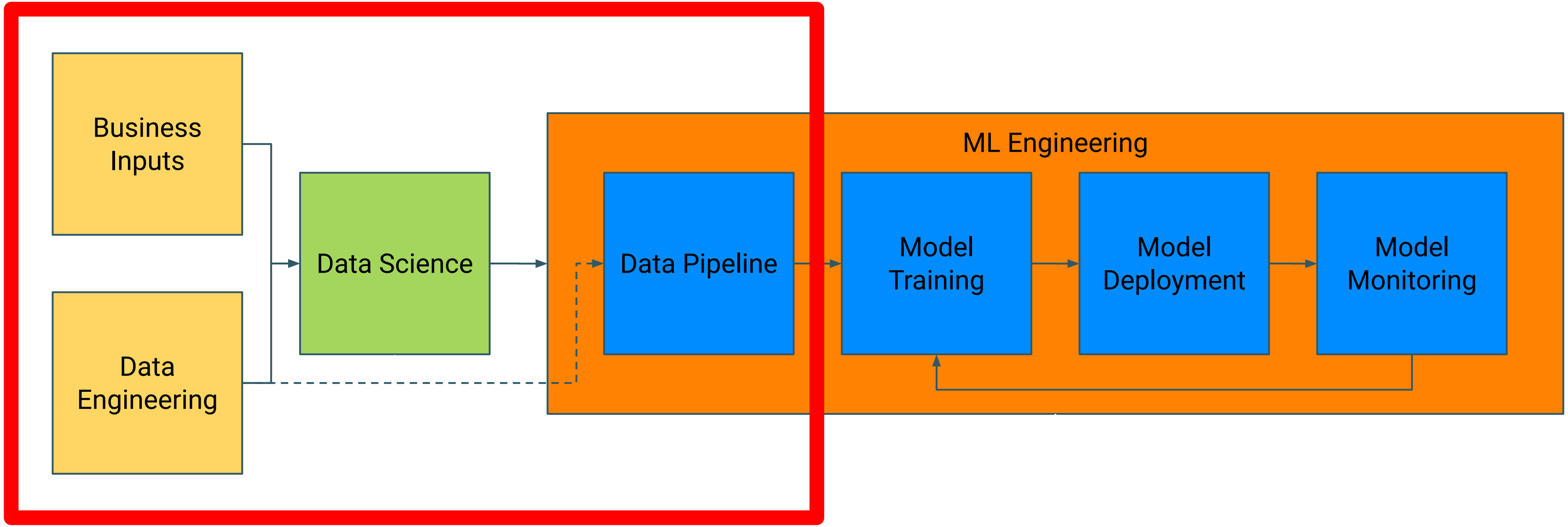
- ✓ Prepaid training credits
- ✓ Use on ANY training product or service
- ✓ Expires 1 year from purchase

Questions ?

BACKUPS

CDSW CONCEPTS / FEATURES

MACHINE LEARNING MODEL OPERATIONS WORKFLOW

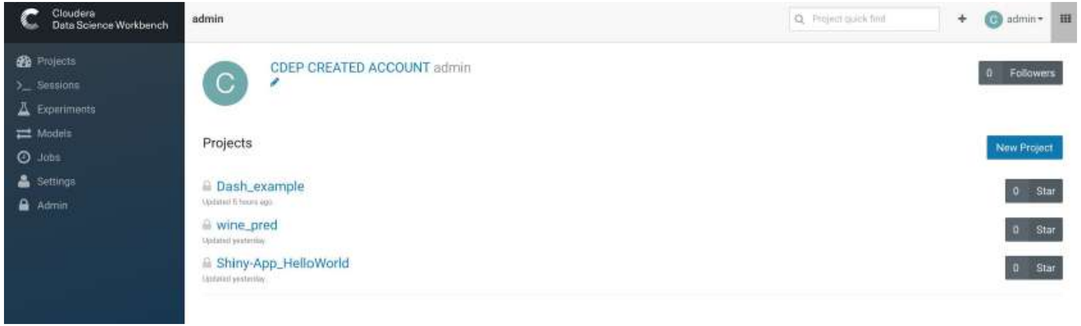


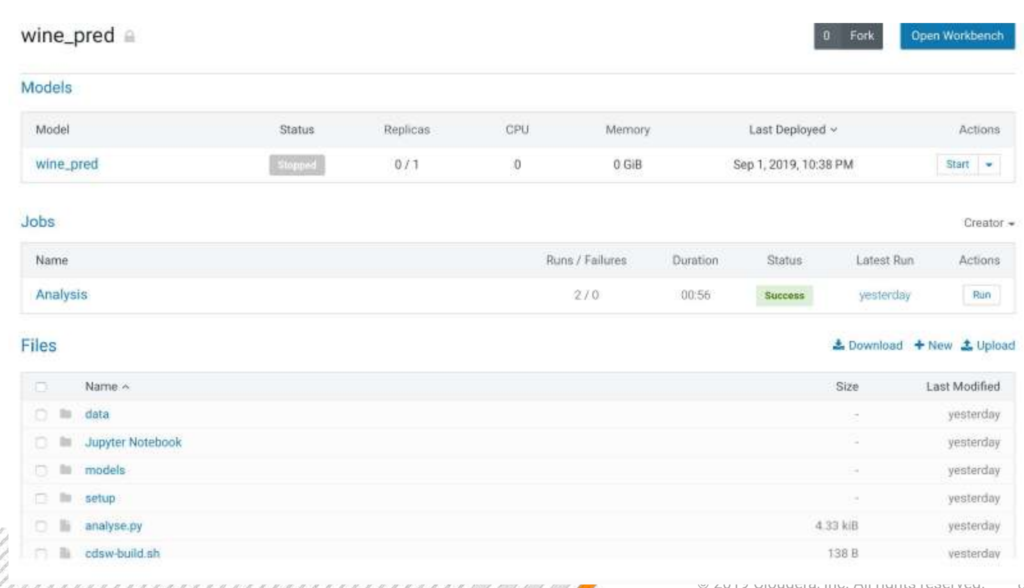
PROJECT

Basic collaborative unit for collaborative work

Data scientists can :

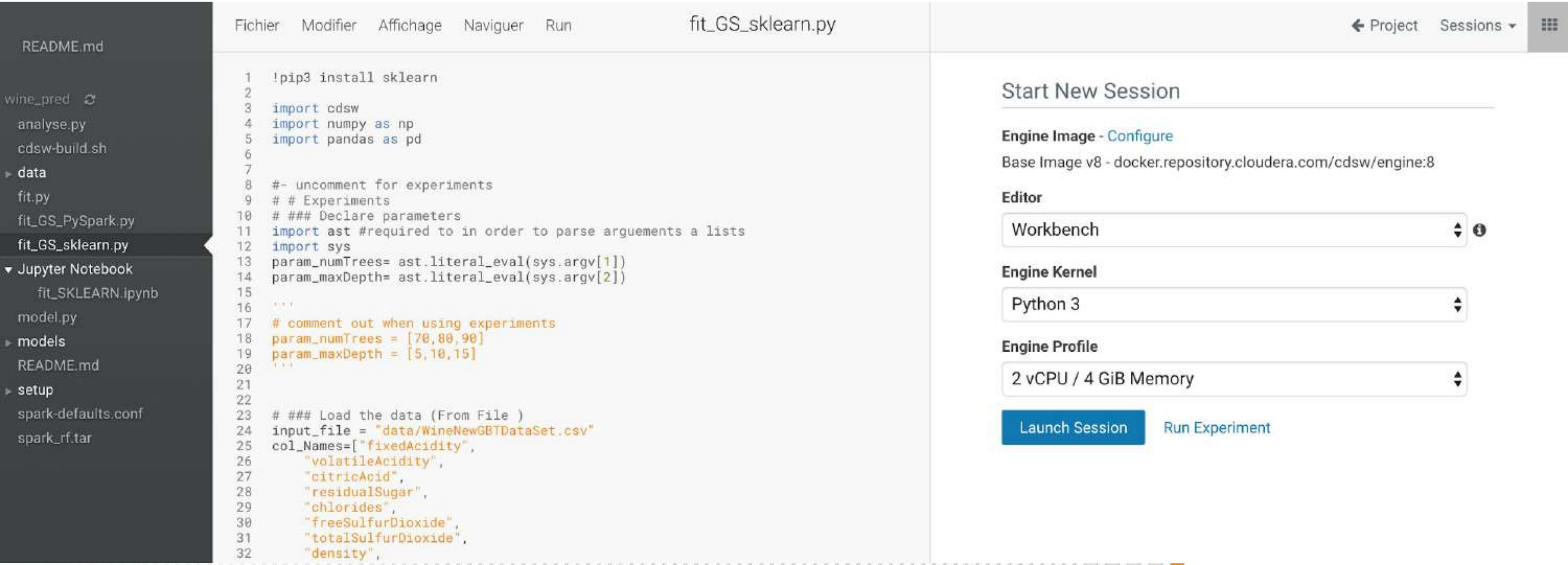
- Create personal / shared or public projects
- Structure and collaborate on common work
- Instantiate and customize individual execution environments (sessions)
- Schedule analysis, report, retraining, application (Jobs)
- Deploy models for online scoring





WORKBENCH (EDITOR + SESSION)

Code Editing to : Analyse data / Develop Scripts / Applications



DATA SCIENTIST EDITOR PREFERENCES

One size does not fit all

Software engineering backgrounds

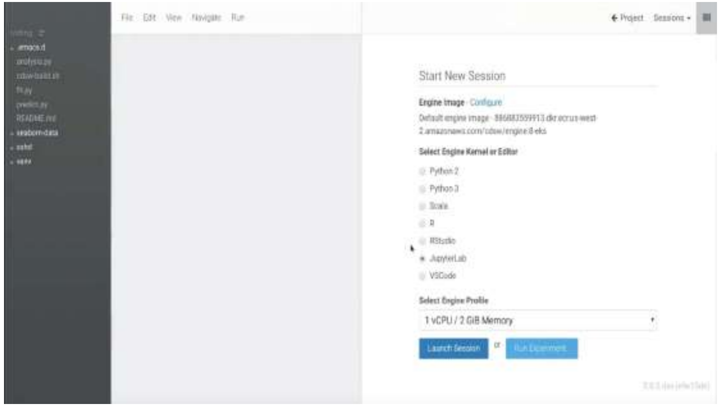
- Mostly favor IDEs e.g., PyCharm
  - Richness of features
  - Familiarity and personal preference

Other code-first data scientists

- Mostly favor Notebooks and RStudio
  - Interactivity of Notebooks
  - Familiarity and personal preference

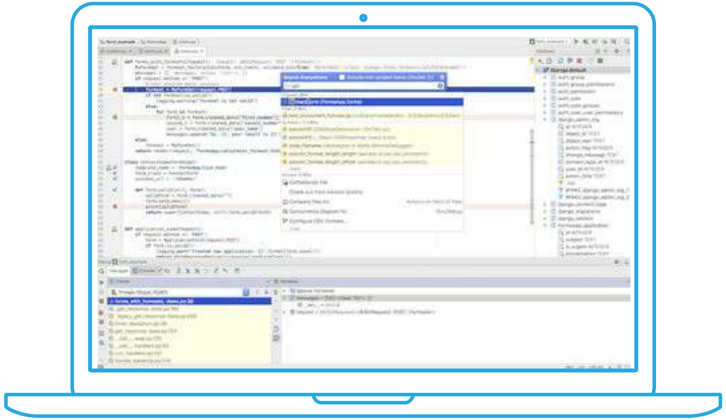
CDSW 1.6 THIRD PARTY EDITOR SUPPORT

Browser-based editors



- Popular editors (RStudio, JupyterLab)
- shipped as a docker image with CDSW
- Third-party editors are enabled within CDSW

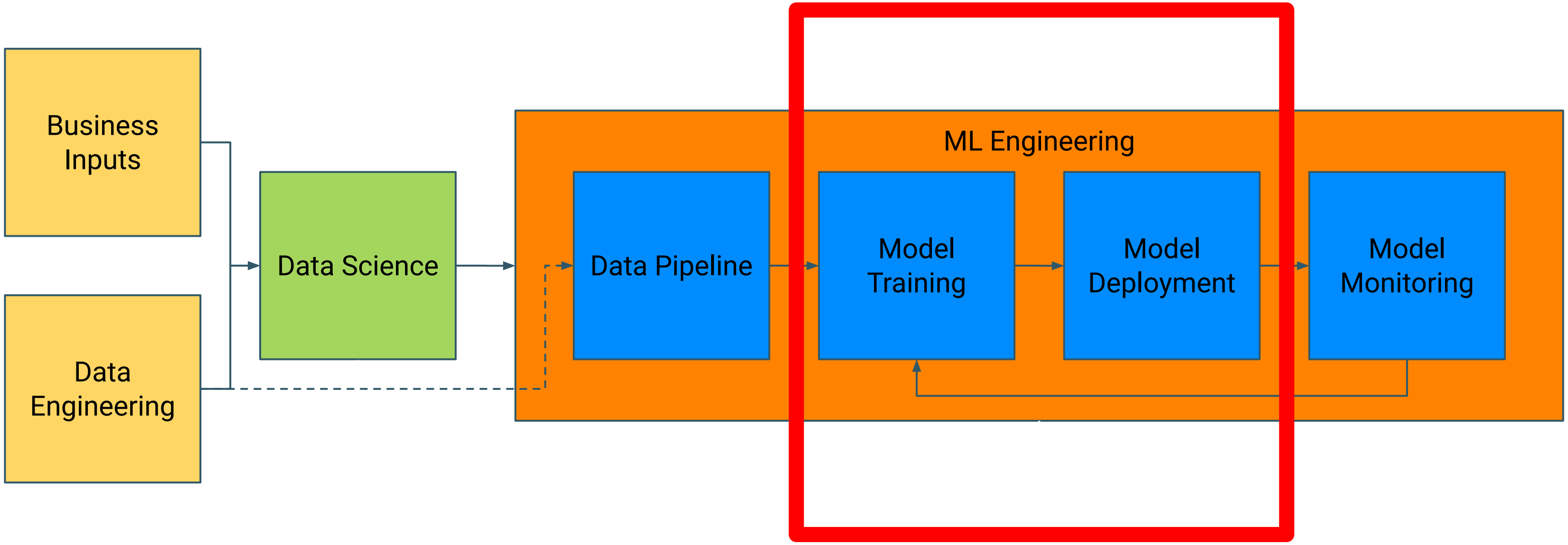
Local editors



CDSW (Remote)

- Code sync with CDSW via Git
- Remote execution in CDSW

MACHINE LEARNING MODEL OPERATIONS WORKFLOW

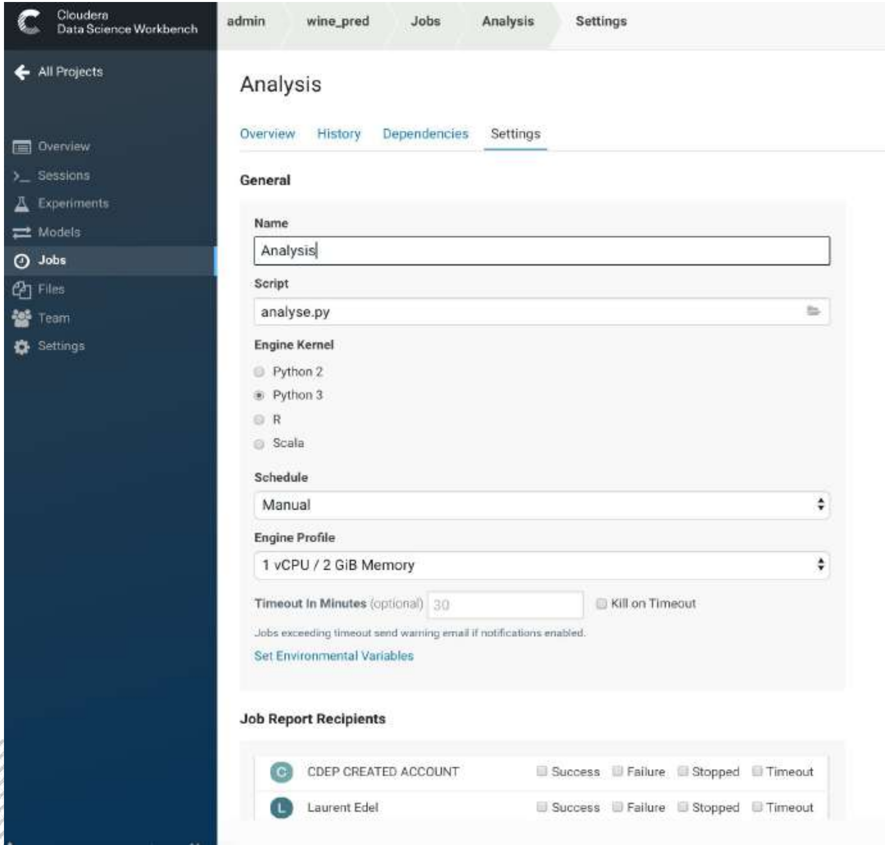


Jobs

Integrated scheduler for job management and automation

The Jobs interface allows DataScientists to:

- Schedule jobs on a manual or repeatable basis
- Create pipeline of dependent scripts
- Send results of job results and status
- Integrate with other tools using its open API

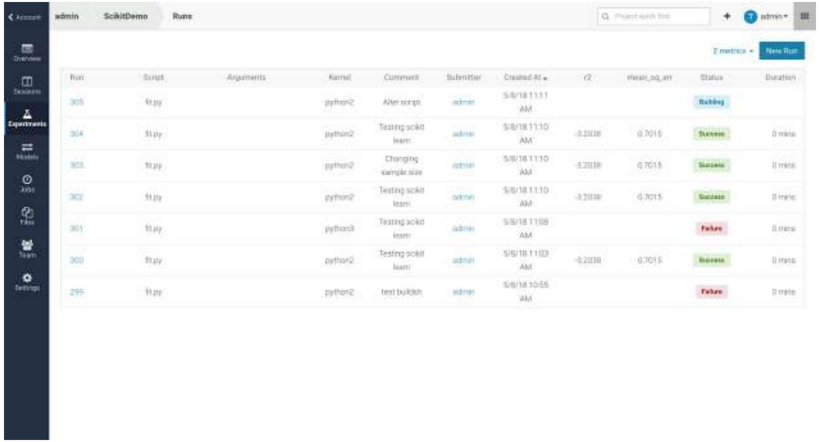


EXPERIMENTS

Versioned model training runs for evaluation and reproducibility

With experiment data scientists can :

- Create a snapshot of model code, dependencies, and configuration necessary to train the model
- Build and execute the training run in an isolated container
- Track specified model metrics, performance, and model artifacts
- Inspect, compare, or deploy prior models



MODELS

Machine learning models as one-click microservices (REST APIs)

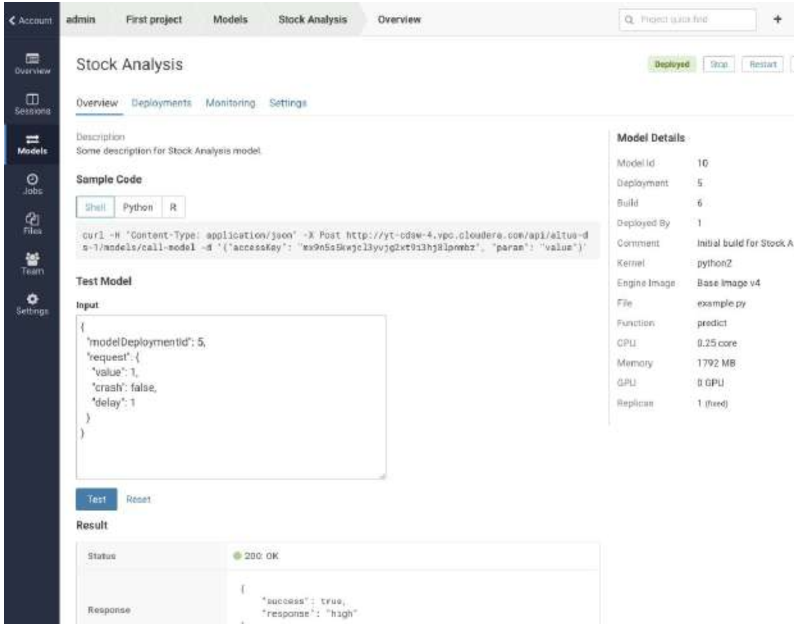
1. Choose file, e.g. `score.py`
2. Choose function, e.g. `forecast`

```
f = open('model.pk', 'rb')
model = pickle.load(f)

def forecast(data):
    return model.predict(data)
```

3. Choose resources
4. Deploy!

Running model containers also have access to CDH for data lookups.



APPLICATION DEPLOYMENT

Deploy interactive application via Sessions or Jobs

- CDSW can run any type of arbitrary code in Scala, Python or R
- Different ports are exposed through sessions or Jobs to expose web apps such as
  - Shiny or
  - Dash
- App are deployed and exposed via the CDSW infrastructure

test_app Running

By CDEP CREATED ACCOUNT — Python 3 Session — 2 vCPU / 4 GiB Memory — just now See job details

Session Logs Collapse Share

```
> import os
> engine_id = os.environ.get('CDSW_ENGINE_ID')
> cdsw_domain = os.environ.get('CDSW_DOMAIN')
> print("app running at http://{}.{}".format(engine_id,cdsw_domain))
  app running at http://90j7dethdwyno58q.sfrmlamairesse-4.vpc.cloudera.com
> !python3 app.py
```