

CLUSTERA

DATA SCIENCE BEST PRACTICES with CDSW

سابک
sebk

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

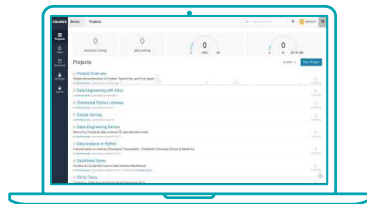
(ML an DS @Cloudera)

Cloudera ML offering

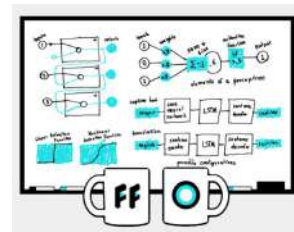
Modern enterprise platform, tools and expert guidance to help you unlock business value with ML/AI



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

CLUSTERA DATA PLATFORM

HYBRID &
MULTI-CLOUD



DATA CENTER &
PRIVATE CLOUD



HYBRID
CLOUD



MULTI
PUBLIC CLOUD

SECURITY &
GOVERNANCE

CLUSTERA
SDX

METADATA / SCHEMA / MIGRATION / SECURITY / GOVERNANCE

ANALYTICS
EDGE TO AI



DATA
HUB



DATA FLOW &
STREAMING



DATA
ENGINEERING



DATA
WAREHOUSE



OPERATIONAL
DATABASE



MACHINE
LEARNING

OPEN
DISTRIBUTION

CLUSTERA RUNTIME



CONTROL
PLANE



DATA
CATALOG



REPLICATION
MANAGER

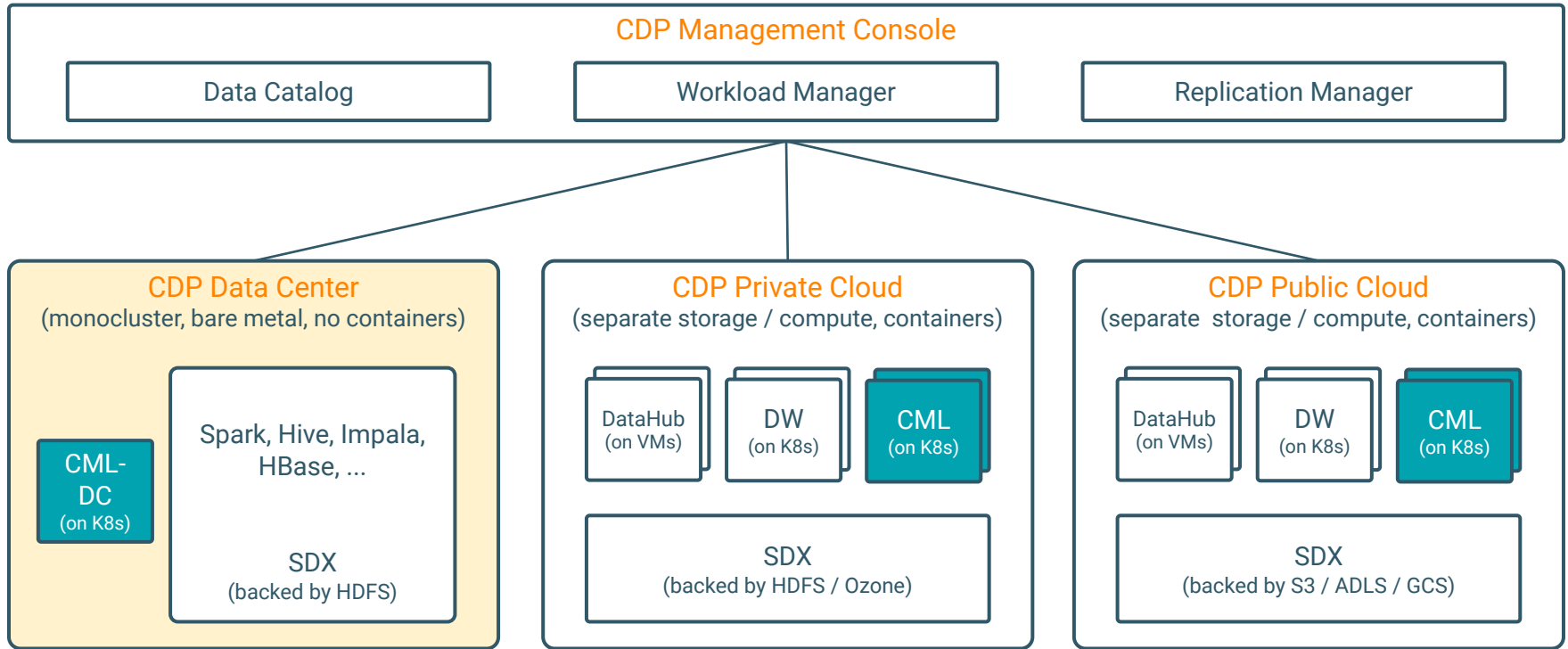


WORKLOAD
MANAGER



MANAGEMENT
CONSOLE

CDP OFFERS HYBRID, MULTI-CLOUD ARCHITECTURE

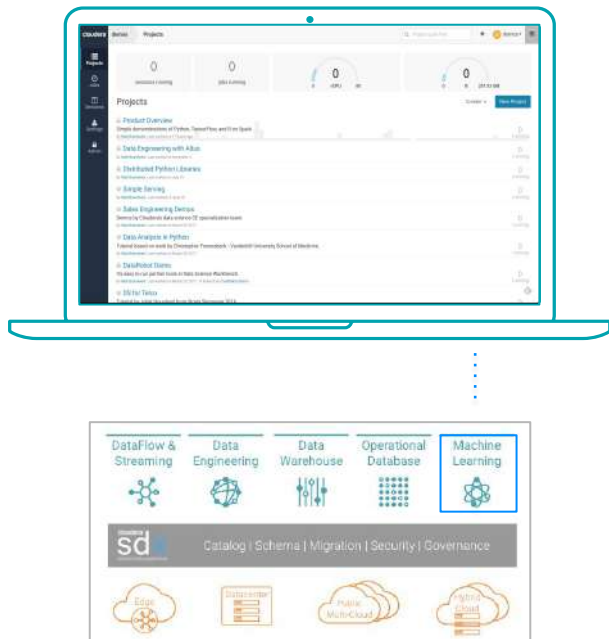


CLOUDERA DATA SCIENCE WORKBENCH

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)



Golang



Dash
byplotly



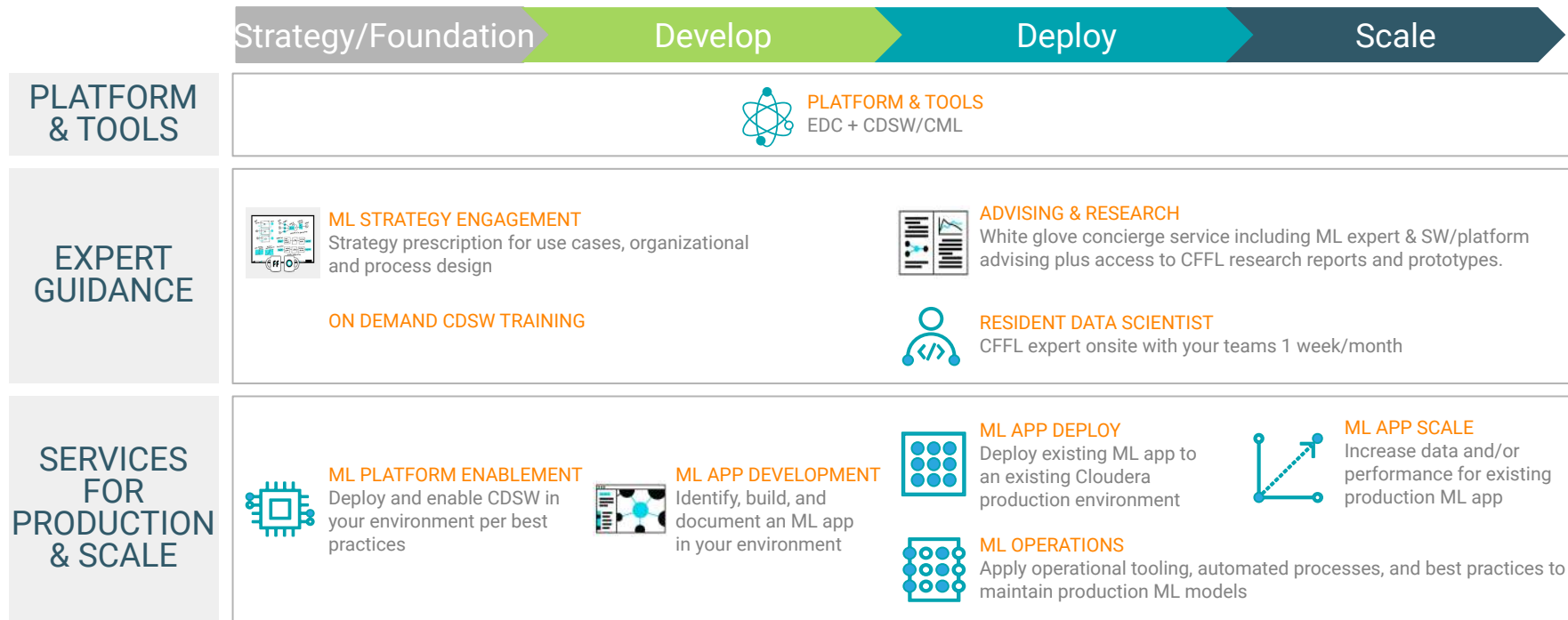
XGBoost



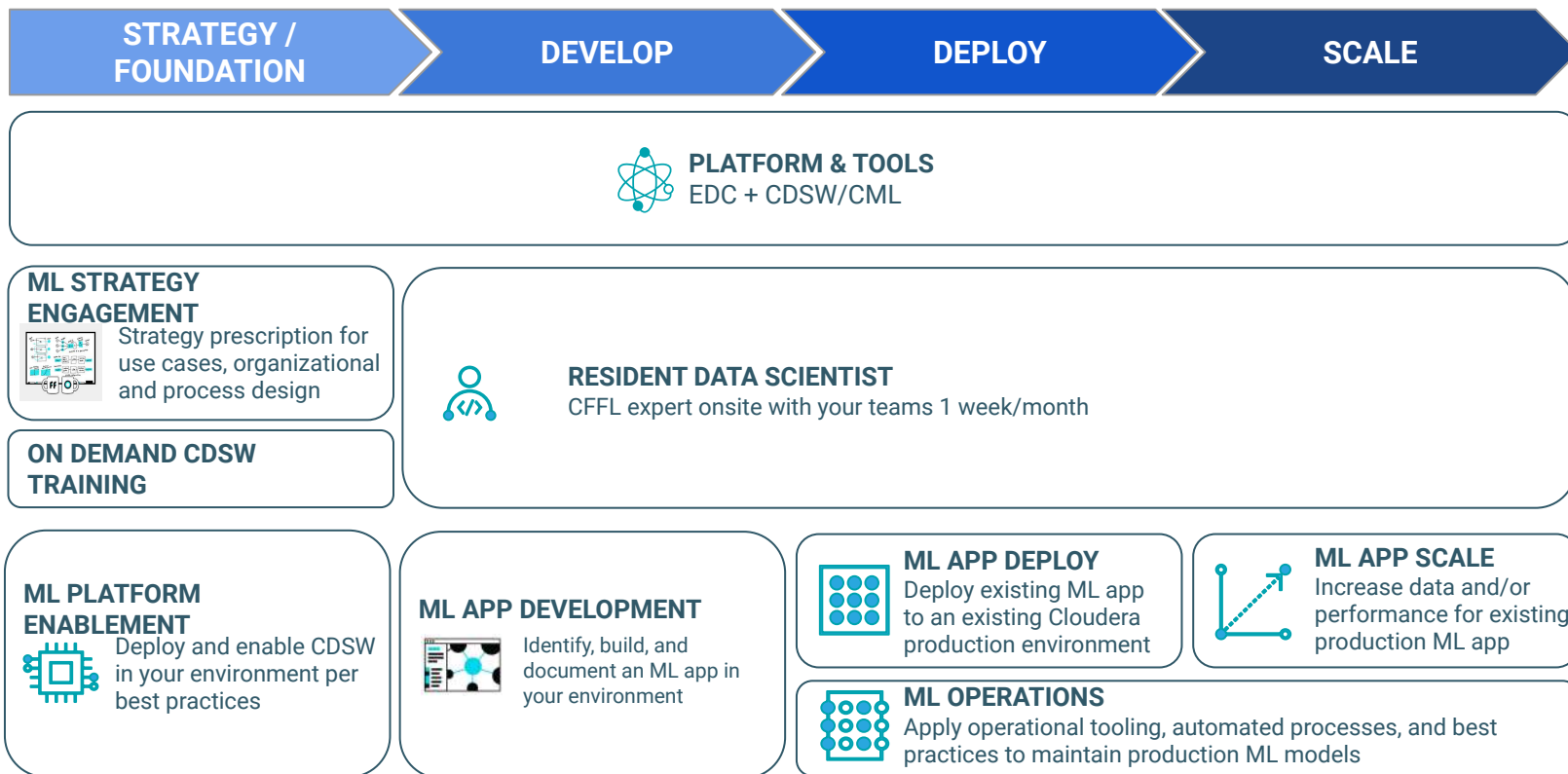
SERVICES

SMARTML

Cloudera's ML services for the journey to industrialized AI



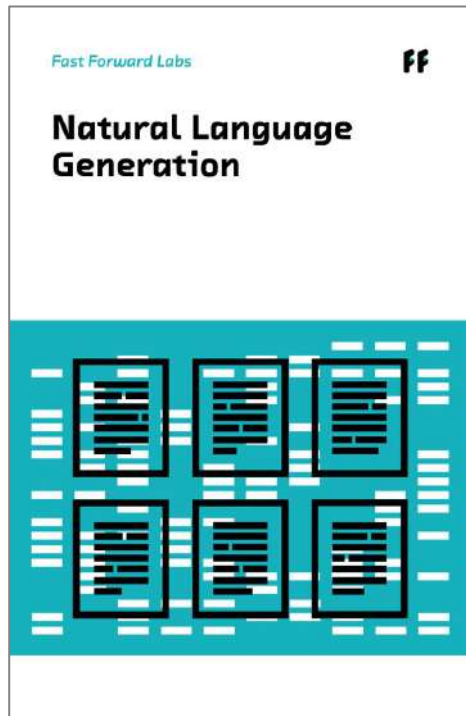
CLOUDERA'S ML SERVICES FOR THE JOURNEY TO INDUSTRIALIZED AI



FFL RESEARCH TOPICS

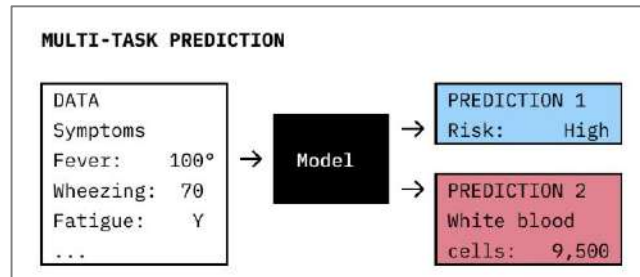
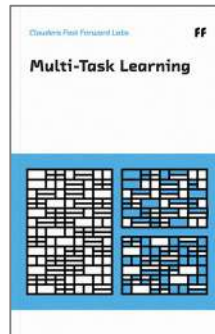
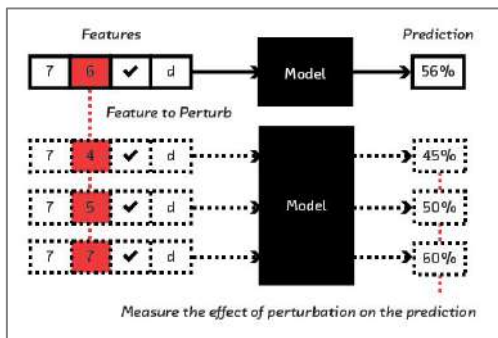
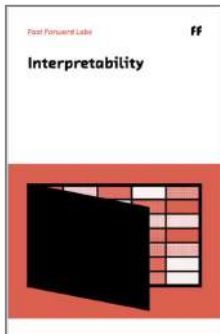
What's inside?

- 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



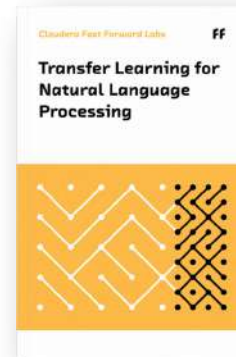
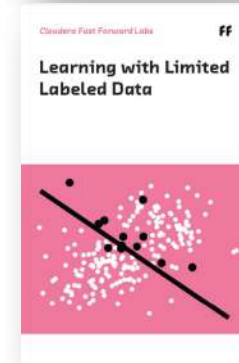
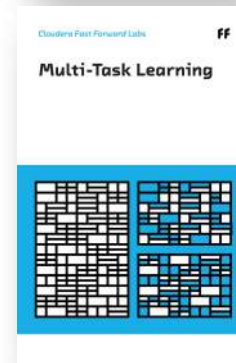
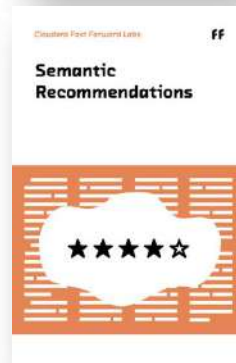
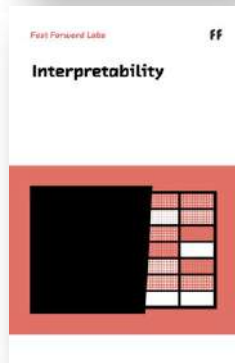
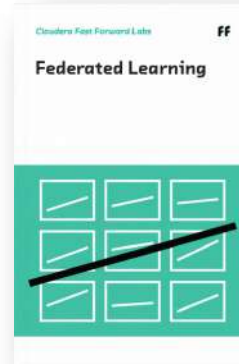
Understanding what's inside the black boxes

- 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

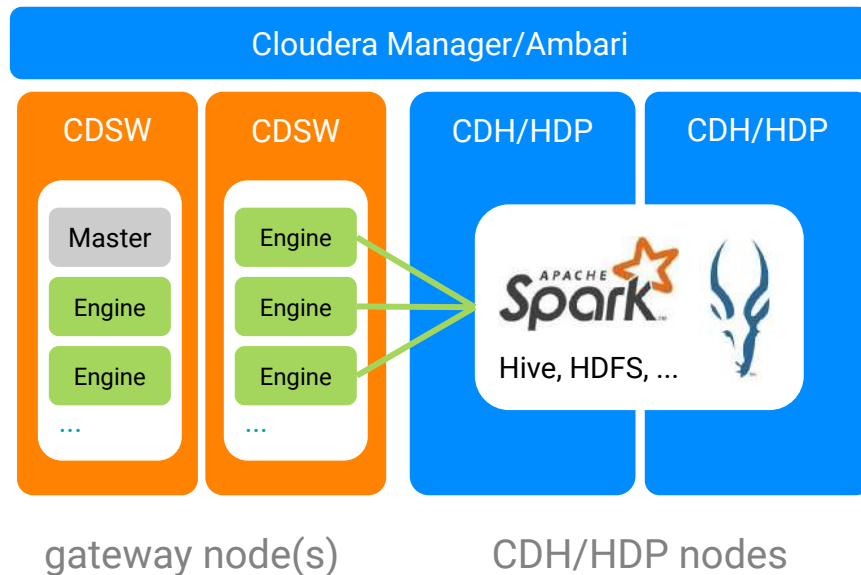
Supports both big and small data

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

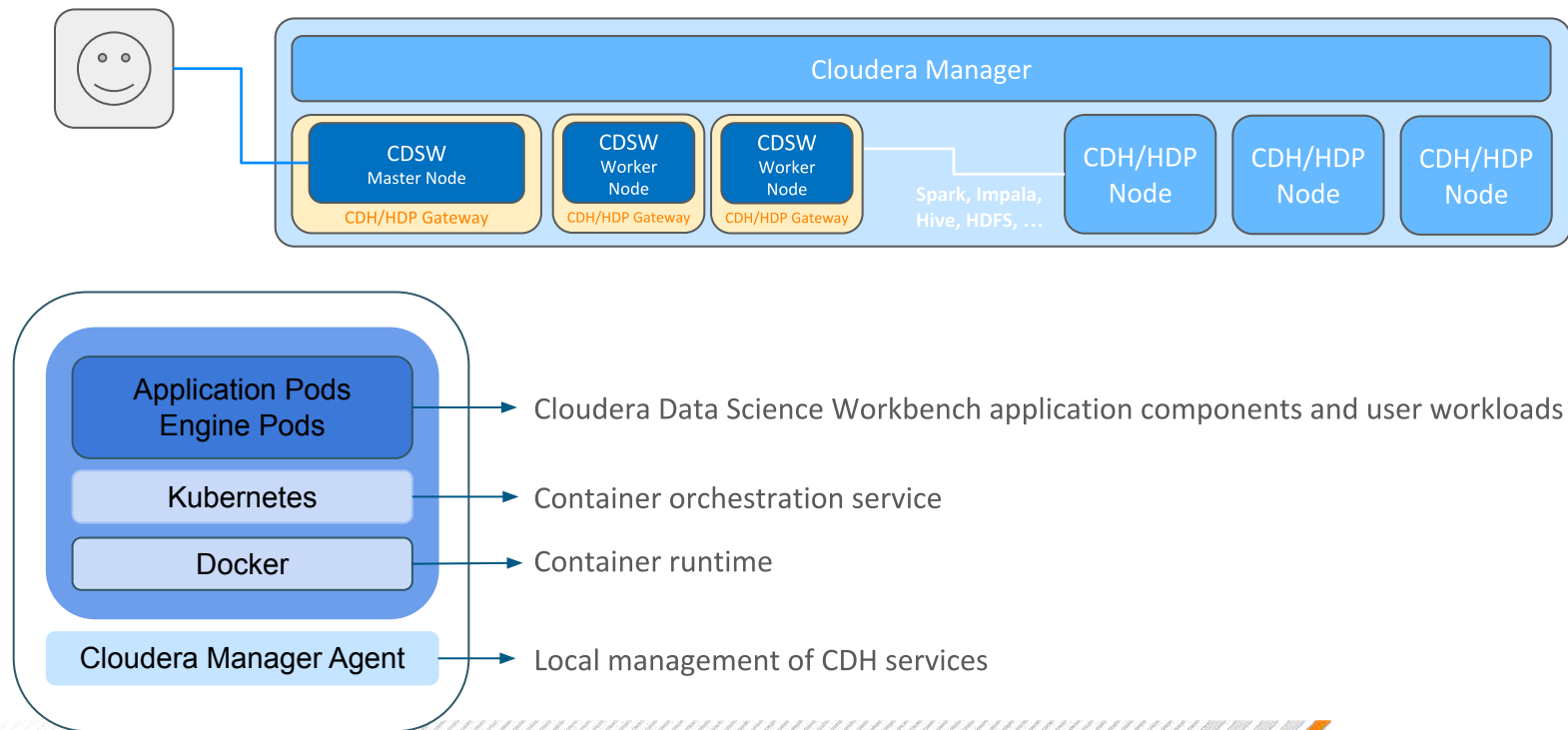
Secure and governed by default

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

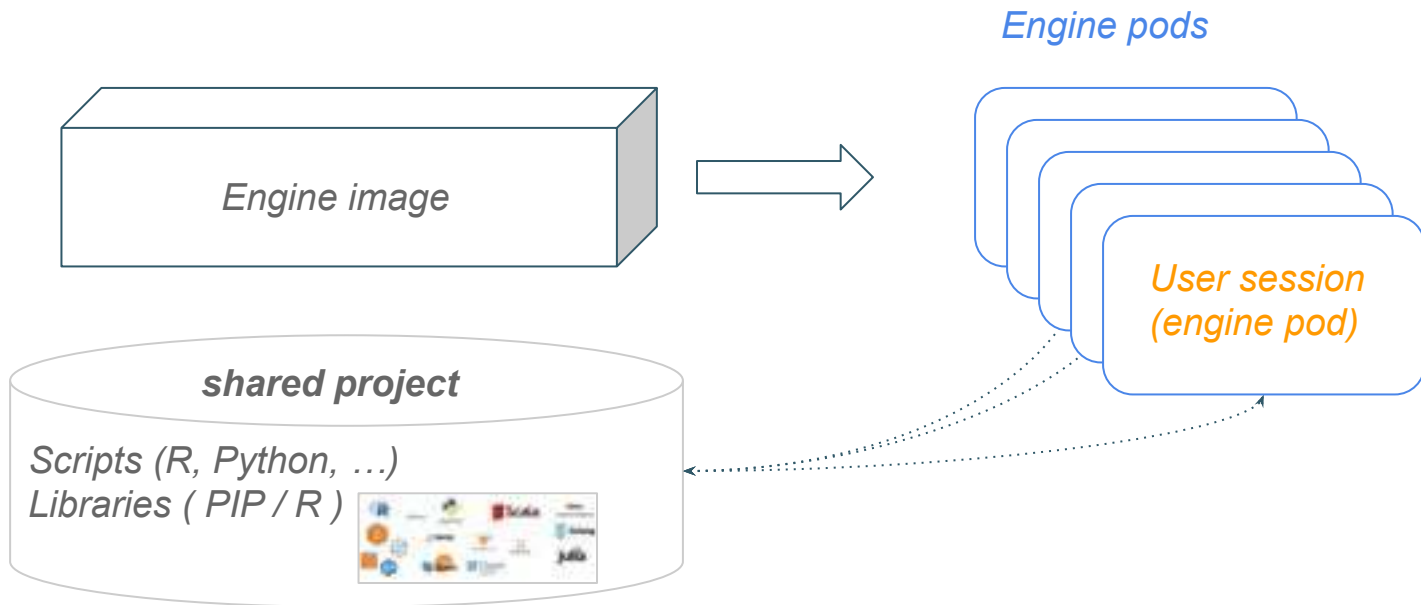
Deployed with Cloudera Manager



OVERALL ARCHITECTURE OVERVIEW



User sessions and project sharing



EXECUTION MODES

LOCAL VS DISTRIBUTED

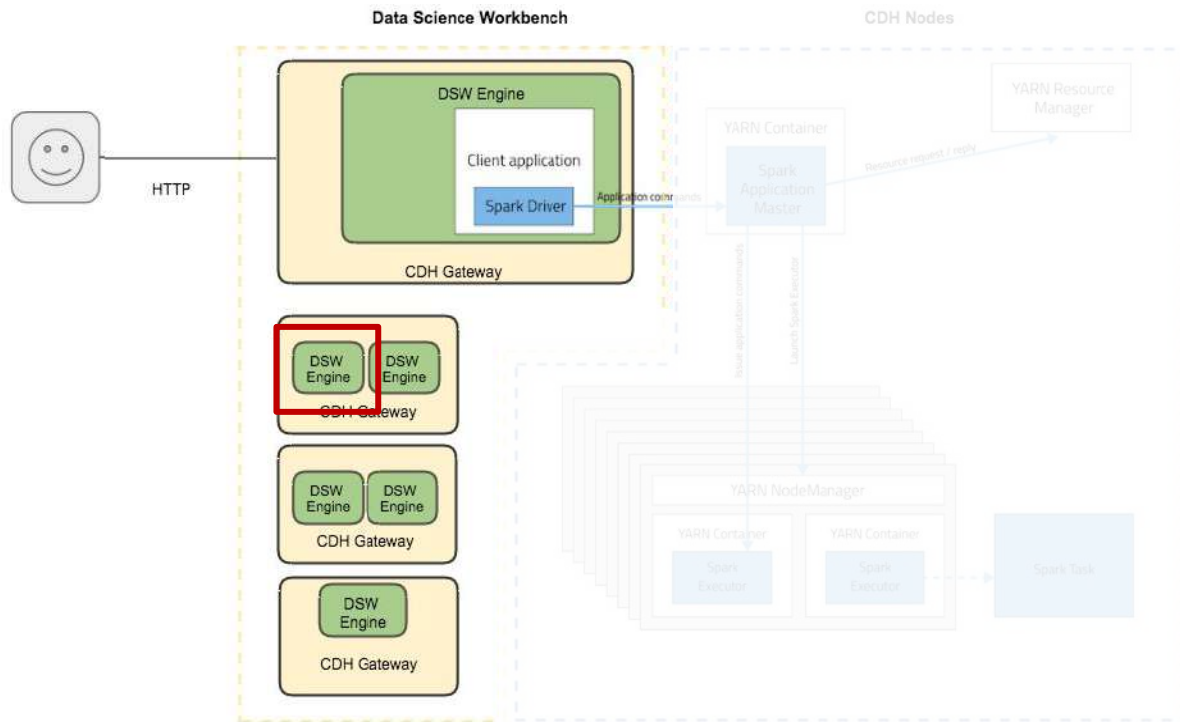
LOCAL - SESSION ON CDSW

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

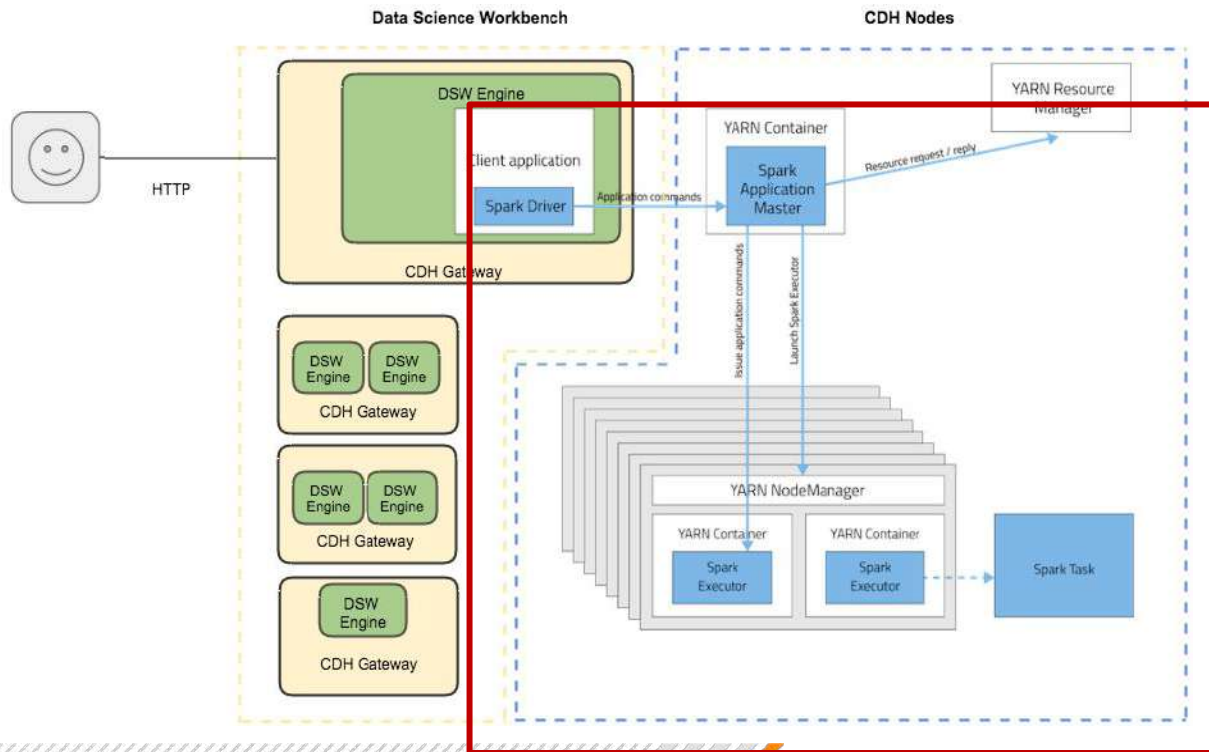
Code execution submitted to Cluster

- 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 as possible leverage spark native functions



ADVANCED : DISTRIBUTED - ON CDSW

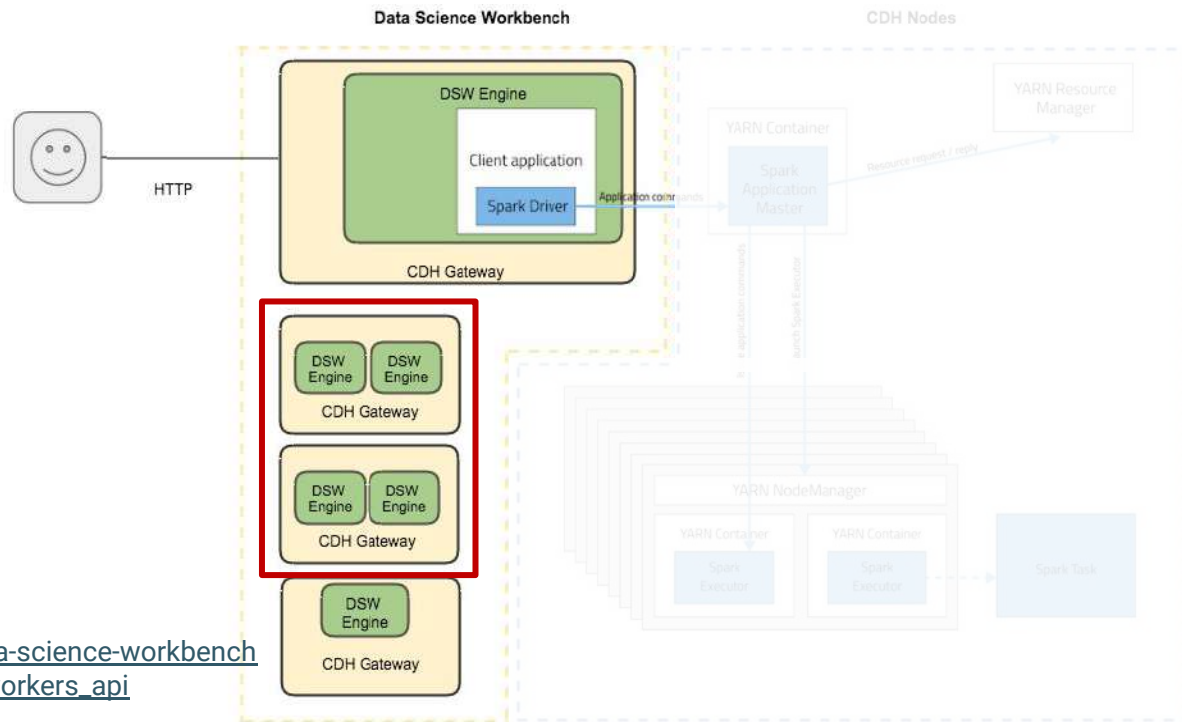
Code execution within the context of a session – with multiple workers

- 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 	Non distributed With optimis  Distributed 	
Comments	Data fit in memory and single machine perf OK Distributed frameworks are often slower due 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 computationally 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 Storage



Data Access

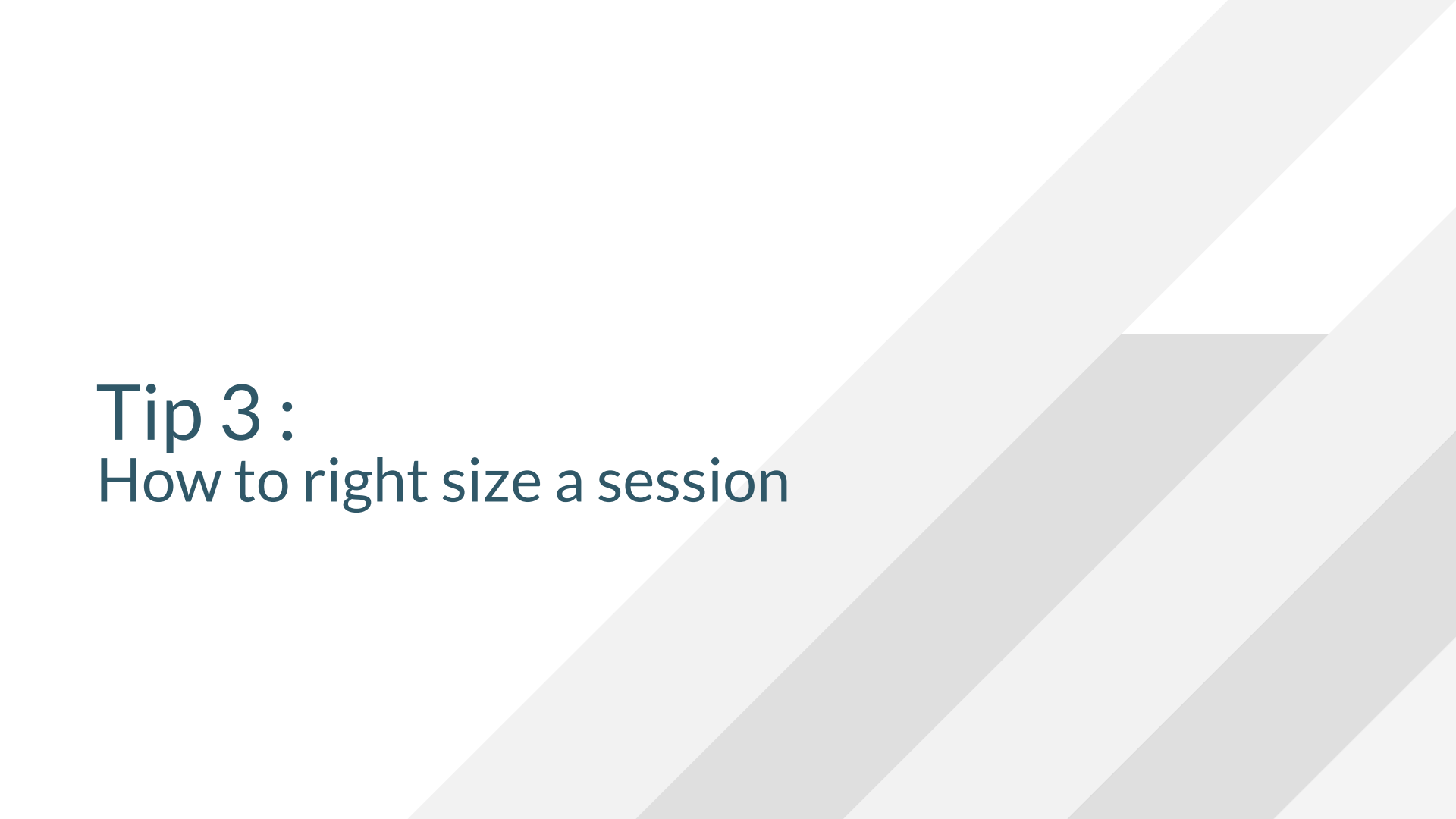


Data Transformation



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	<input type="text" value="1"/>	<input type="text" value="1,75"/>	Add	

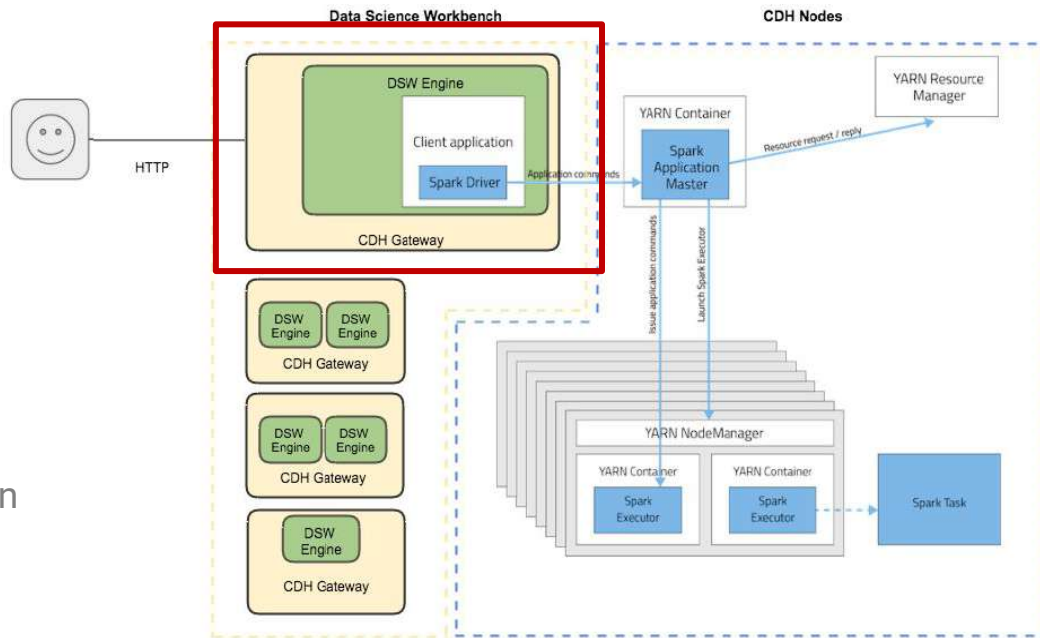
Session sizing - Clarifications - Spark

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 be 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 initial copy
del df
```

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

<pre>df = pd.read_csv('some'file') df.sort_values(inplace=True)</pre>	<pre>df = pd.read_csv('some'file') df = df.sort_values()</pre>
---	--

=> 2x mem for for comp but 1x for result

=> 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

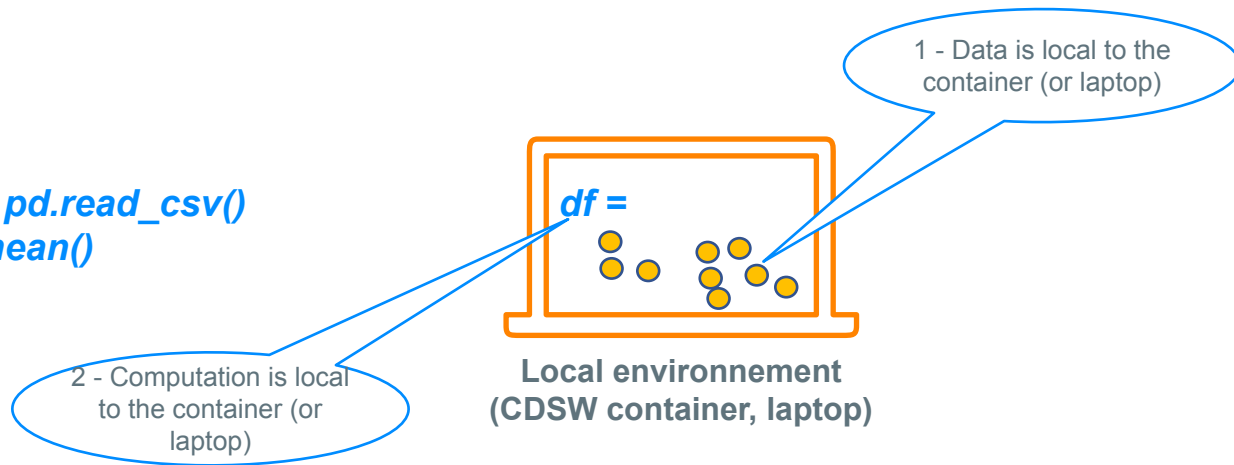
Tip 5 : Spark concepts

The background of the slide features several parallel diagonal stripes in shades of light gray and white, running from the bottom-left towards the top-right.

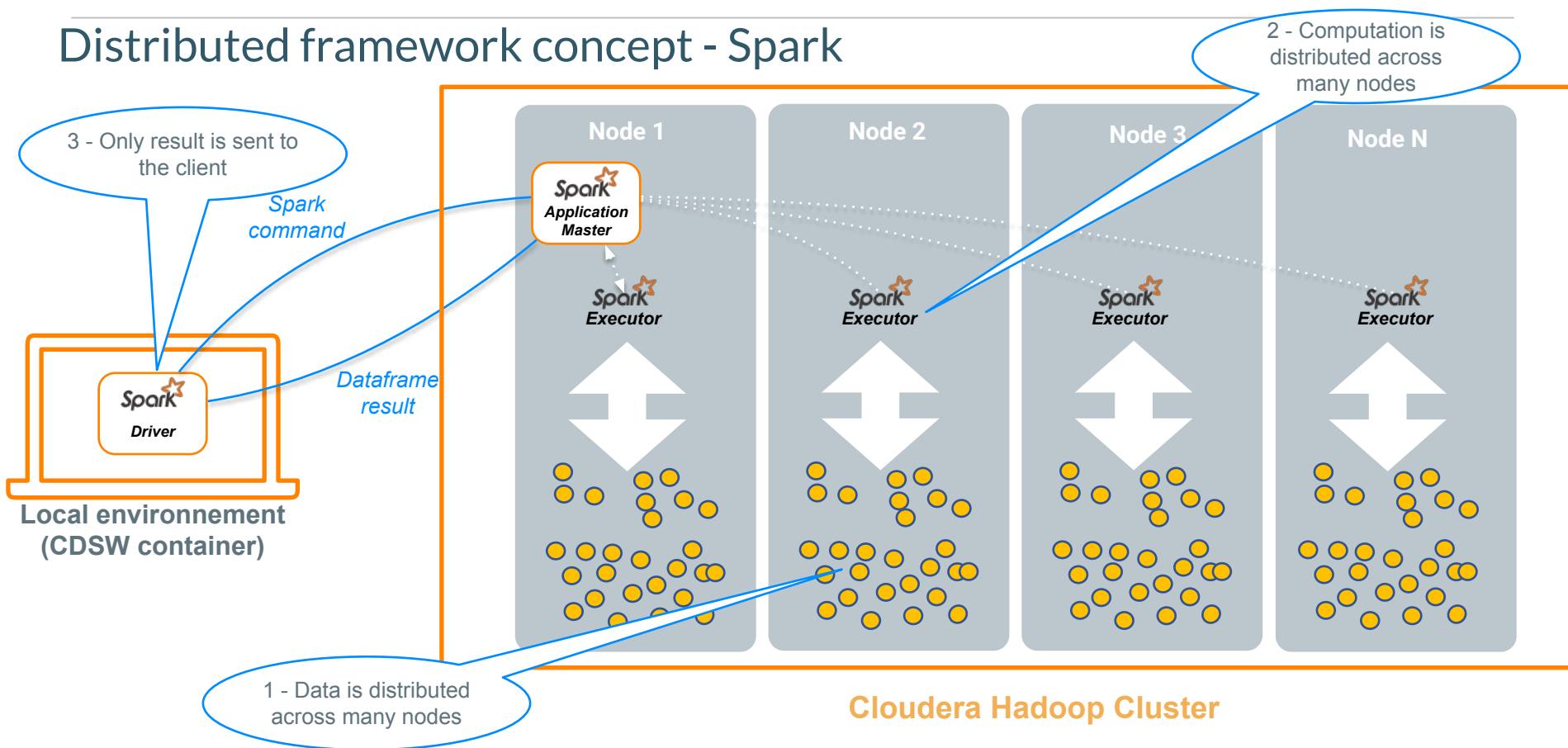
Non-distributed framework concept

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

```
df = pd.read_csv()  
df.mean()
```

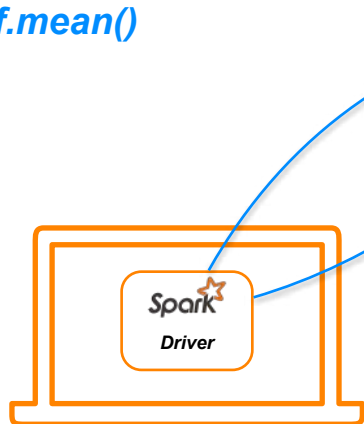


Distributed framework concept - Spark

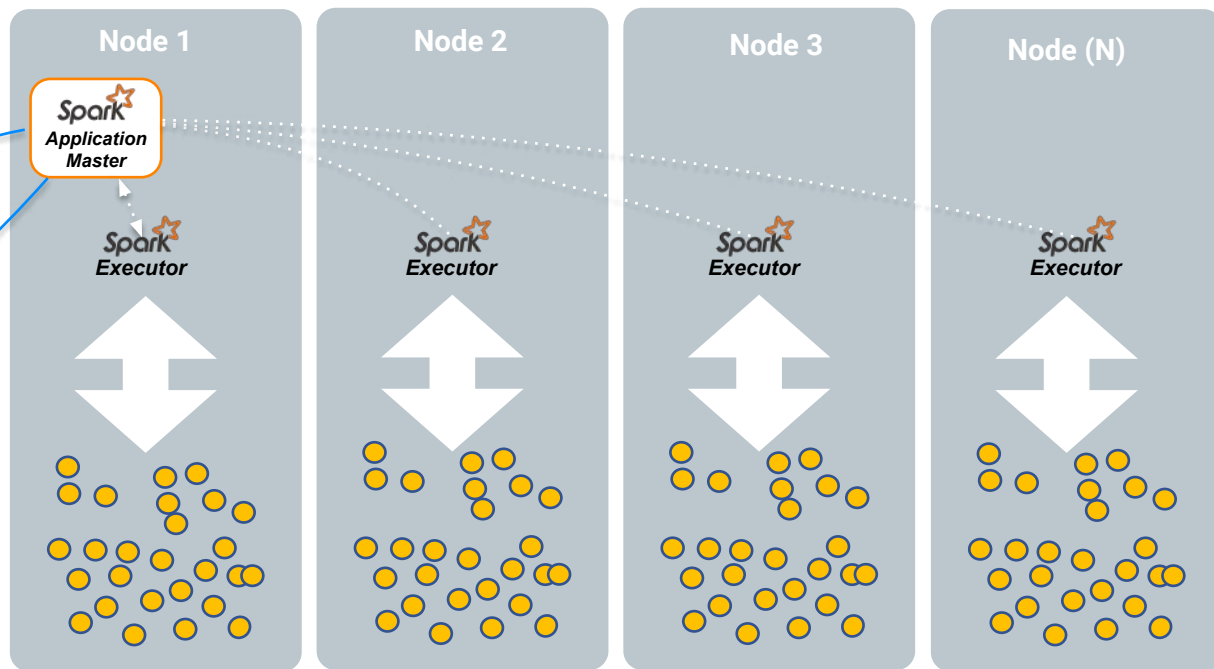


SPARK common mistake

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



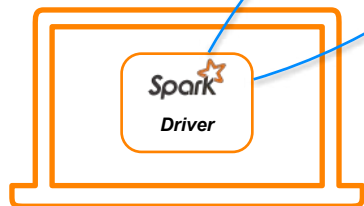
Local environnement
(CDSW container)



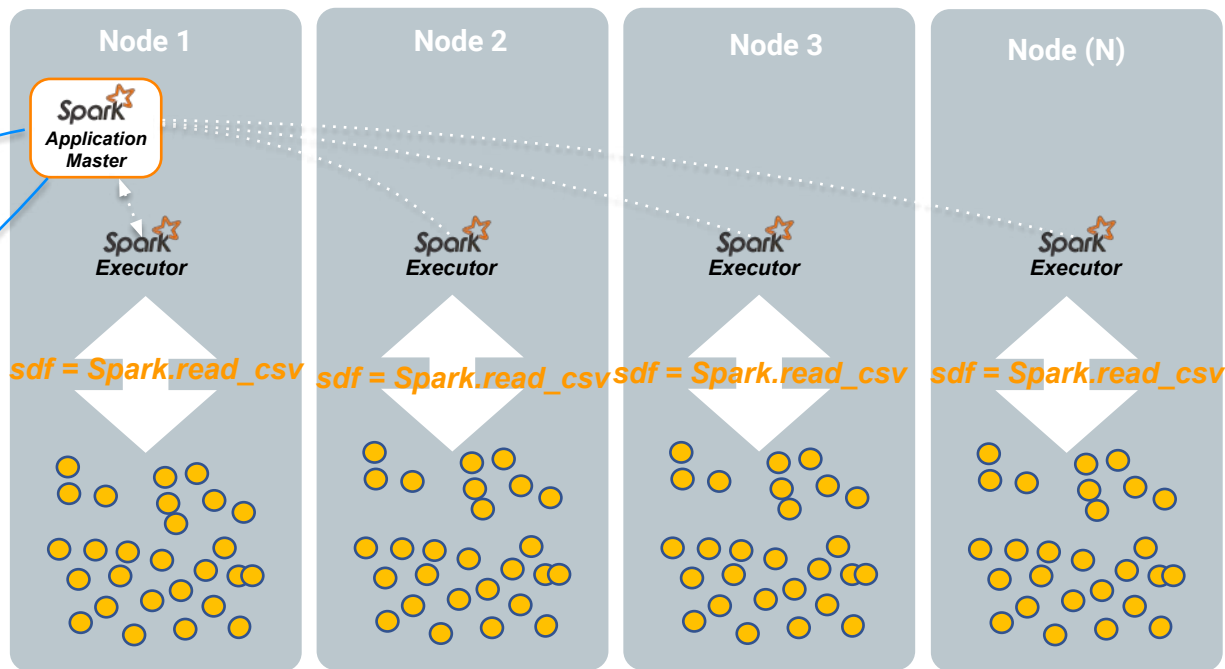
Spark in
Distributed environnement

SPARK common mistake

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



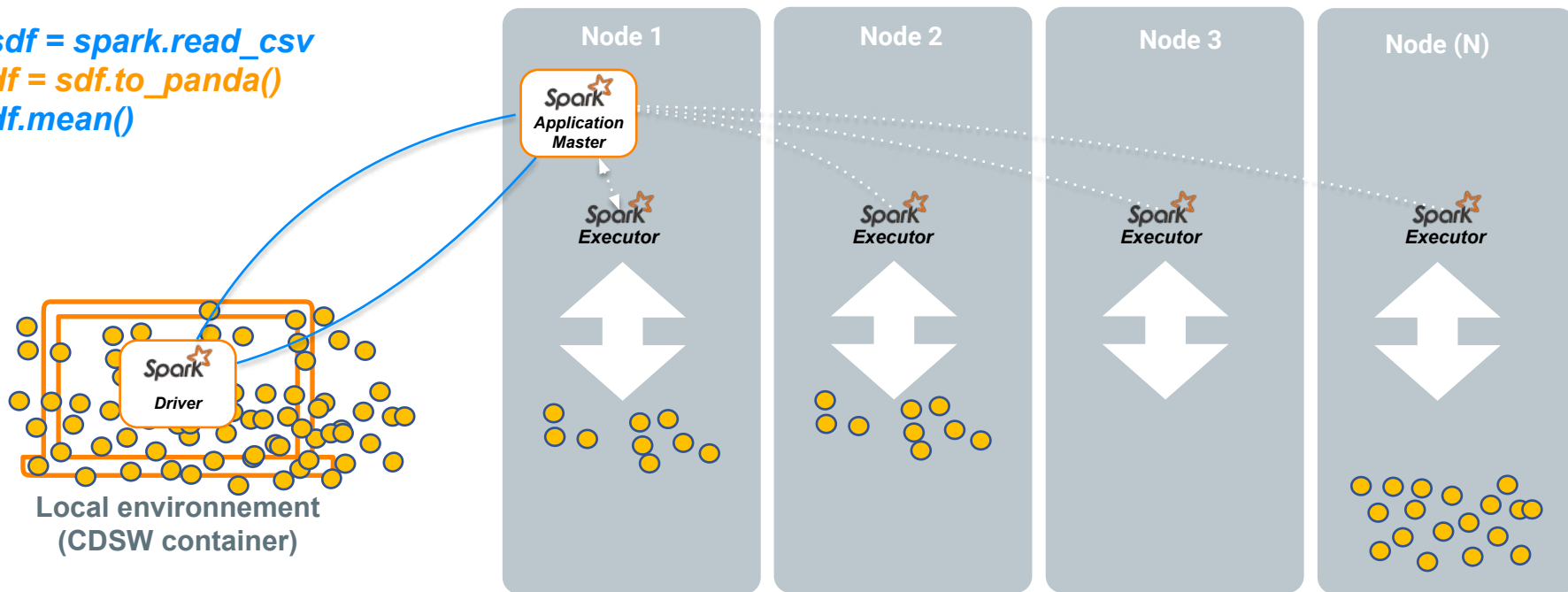
Local environnement
(CDSW container)



Spark in
Distributed environnement

SPARK common mistake

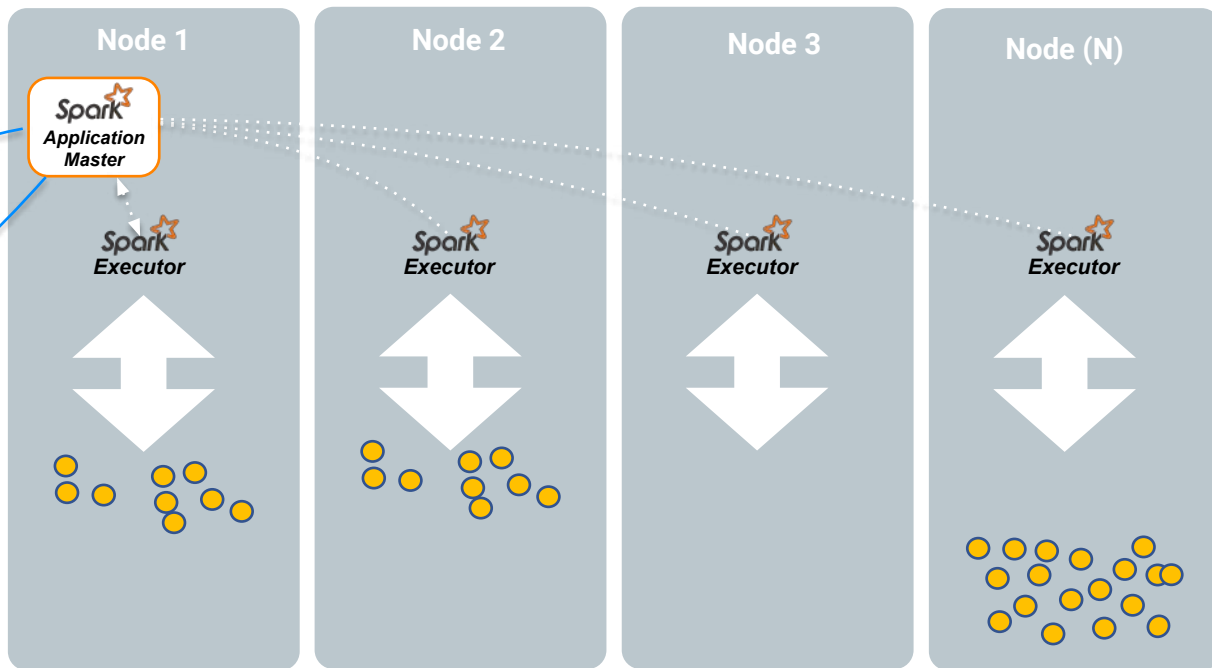
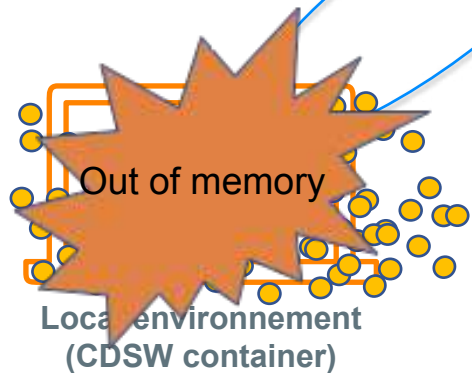
```
sdf = spark.read_csv  
df = sdf.to_panda()  
df.mean()
```



Spark in
Distributed environment

SPARK common mistake

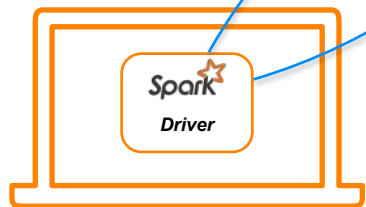
```
sdf = spark.read_csv  
df = sdf.to_panda()  
df.mean()
```



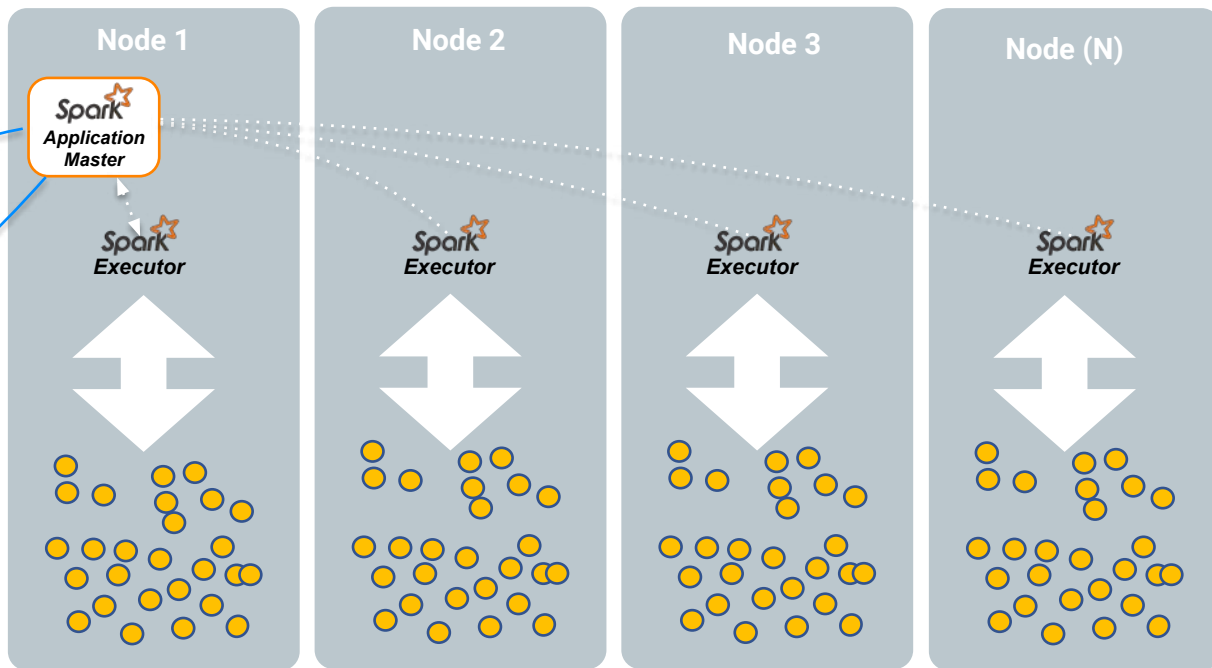
Spark in
Distributed environnement

SPARK The good way

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



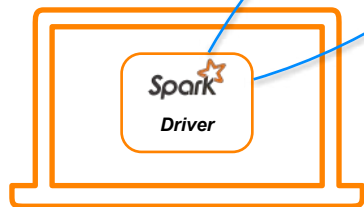
Local environnement
(CDSW container)



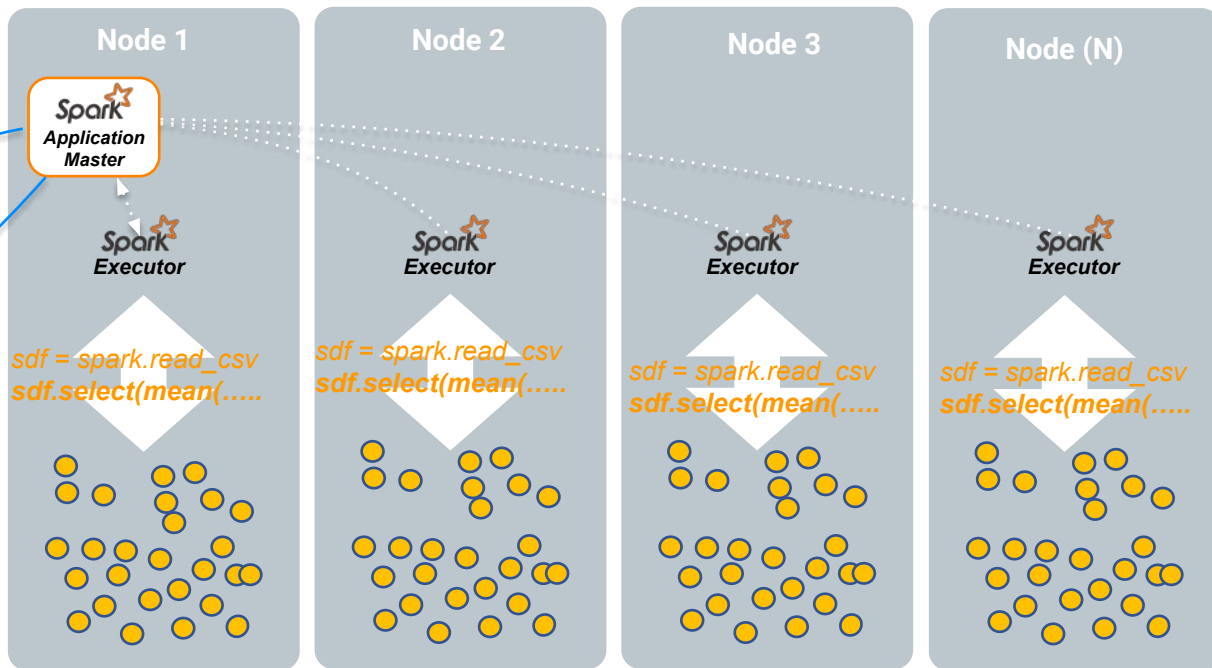
Spark in
Distributed environnement

SPARK The good way

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



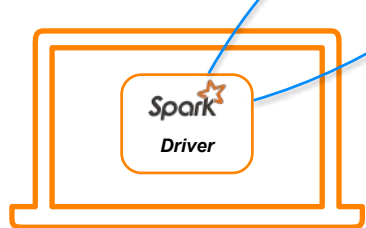
Local environnement
(CDSW container)



Spark in
Distributed environnement

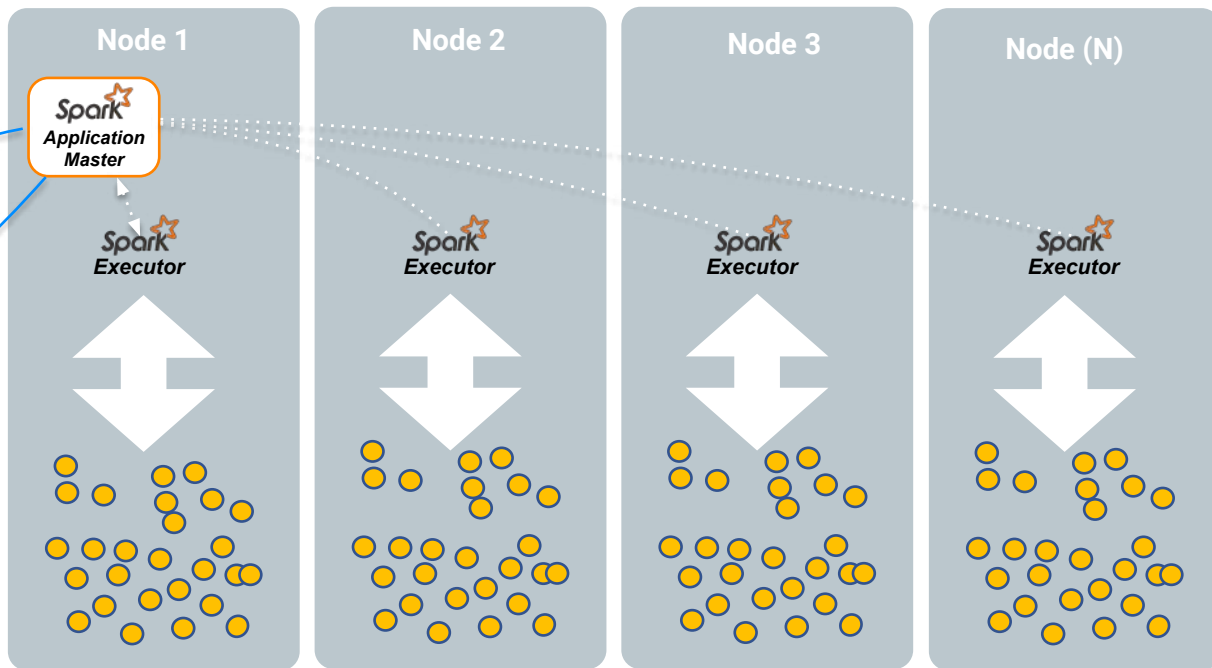
SPARK The good way

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



Local environnement
(CDSW container)

Only result is sent to
the client



Spark in
Distributed environnement

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
- Cartesian 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://rdr.io/cran/sparklyr/api/>

<https://spark.rstudio.com/>

PySpark

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

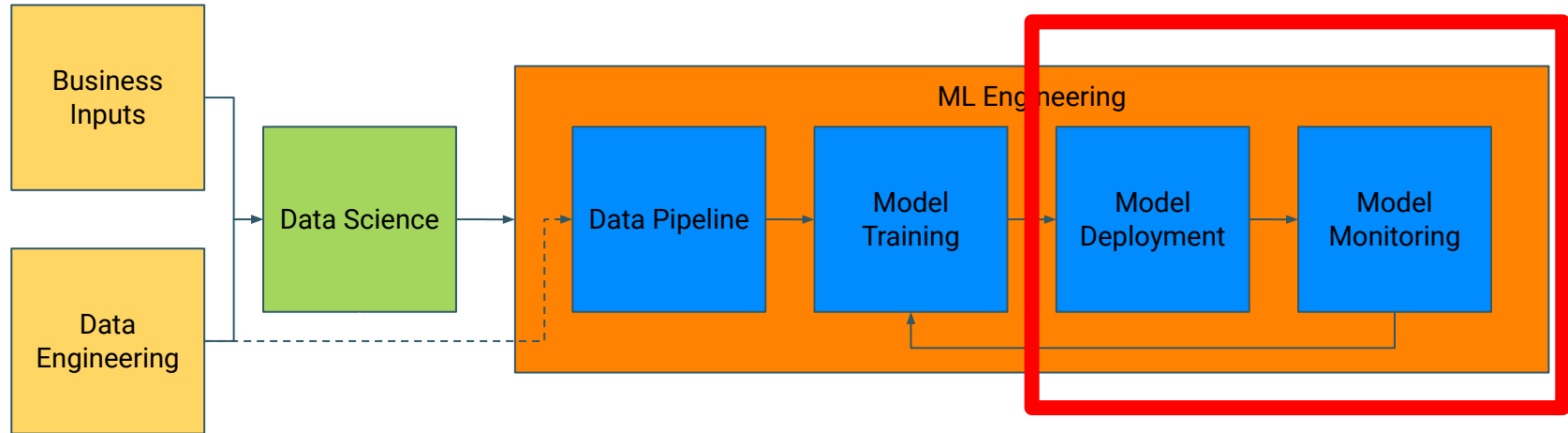
Workshop (examples of Data Engineering and ML using Spark)

https://github.com/fletchjeff/ml_at_scale_workshop

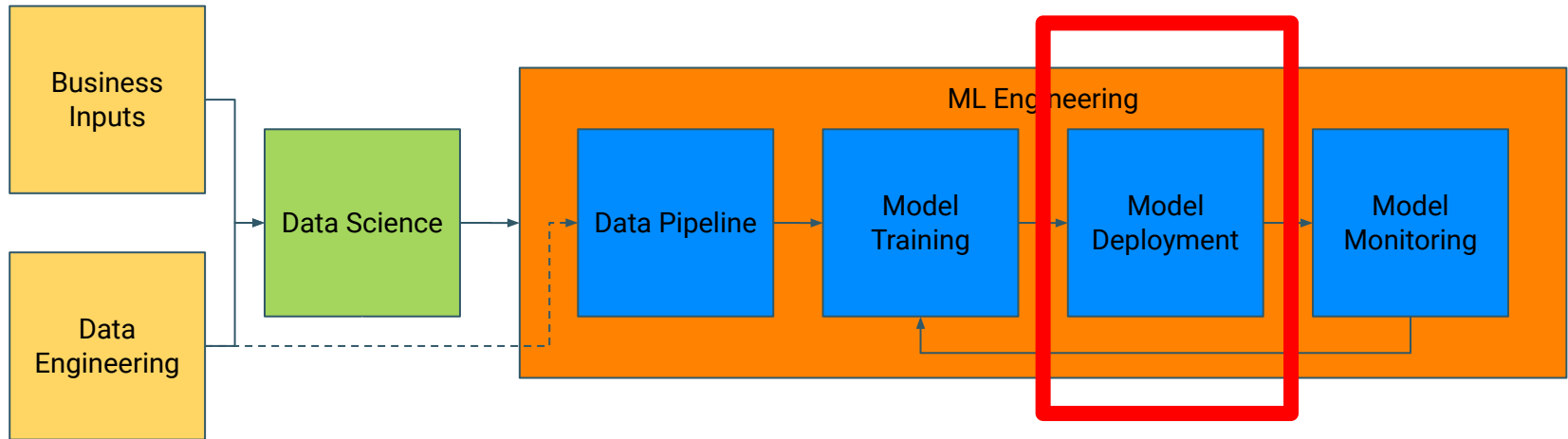
PAUSE

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.

The screenshot shows the Cloudera Model Management interface for a model named 'Stock Analysis'. The left sidebar contains navigation links: Account, Overview, Sessions, Models, Jobs, Files, Team, and Settings. The main panel has tabs for Overview, Deployments, Monitoring, and Settings. The Overview tab is active, showing a description of the model and a sample code block. Below the code is a 'Test Model' section with an input field containing a JSON request. At the bottom, a 'Result' section shows the status as '200 OK' and the response as a JSON object indicating success and a high prediction value. On the right, a 'Model Details' table lists various attributes of the model.

Model Details	
Model Id	10
Deployment	5
Build	6
Deployed By	1
Comment	Initial build for Stock A
Kernel	python2
Engine Image	Base Image v4
File	example.py
Function	predict
CPU	0.25 core
Memory	1792 MB
GPU	0 GPU
Replicas	1 (fixed)

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



```
1 install.packages("sparklyr")
2
3 #list.of.packages <- c("sparklyr","jsonlite", dependencies=TRUE)
4 #new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,'Package'])]
5 #if (length(new.packages)) {
6 #   install.packages(new.packages, repos='https://cran.revolutionanalytics.com/')
7 #}
```

```
### Remplacer 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 absolu 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)

The screenshot displays the Cloudera Data Science Workbench (CDSW) interface for a model named 'Stock Analysis'. The interface is divided into several sections:

- Navigation Bar:** Includes 'Account', 'admin', 'First project', 'Models', 'Stock Analysis', and 'Overview'. A search bar is present on the right.
- Model Overview:** Shows the model name 'Stock Analysis' with buttons for 'Deployed', 'Stop', and 'Restart'. Below this are tabs for 'Overview', 'Deployments', 'Monitoring', and 'Settings'.
- Description:** A brief description of the model.
- Sample Code:** A code editor showing a shell script for calling the model's API.
- Test Model:** A section for testing the model with input parameters.
- Result:** A section showing the status and response of the model test.
- Model Details:** A sidebar on the right providing technical specifications for the model.

Sample Code:

```
curl -H "Content-Type: application/json" -X Post http://yt-cdsw-4-ygo.cloudera.com/api/altus-ds-1/models/call-model -d '{"accessToken": "w9d5d5k9c3lyvg2k9b3h3lamebz", "param": {"value": "value"}}
```

Test Model Input:

```
{
  "modelDeploymentId": 5,
  "request": {
    "value": 1,
    "crash": false,
    "delay": 1
  }
}
```

Test Result:

```
Status: 200 OK
Response: {
  "success": true,
  "response": "high"
}
```

Model Details:

Model Details	
Model id	10
Deployment	5
Build	6
Deployed By	1
Comment	Initial build for Stock A
Kernel	python2
Engine Image	Base Image v4
File	example.py
Function	predict
CPU	0.25 core
Memory	1792 MB
GPU	0 GPU
Replicas	1 (head)

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.



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

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)

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](#)

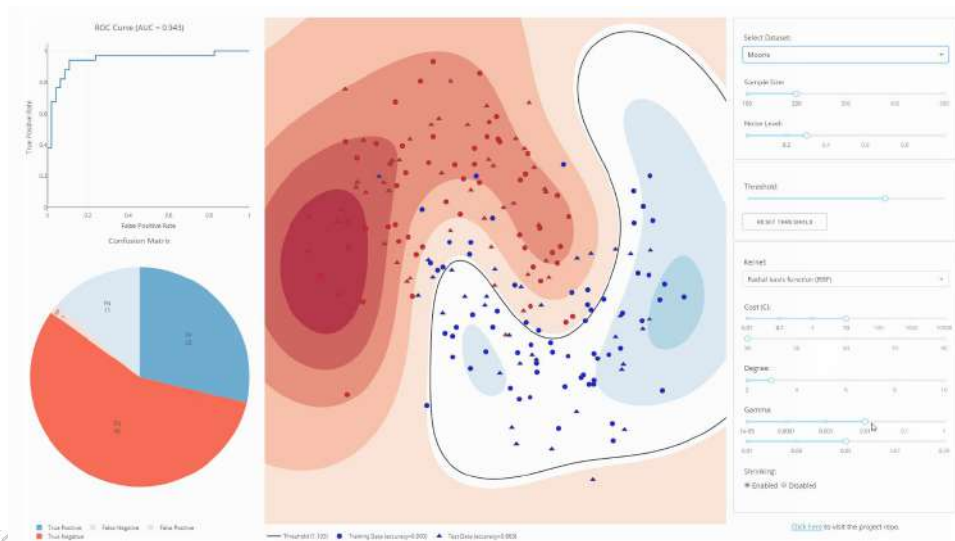
Session [Logs](#)

[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.sfrmlamairese-4.vpc.cloudera.com

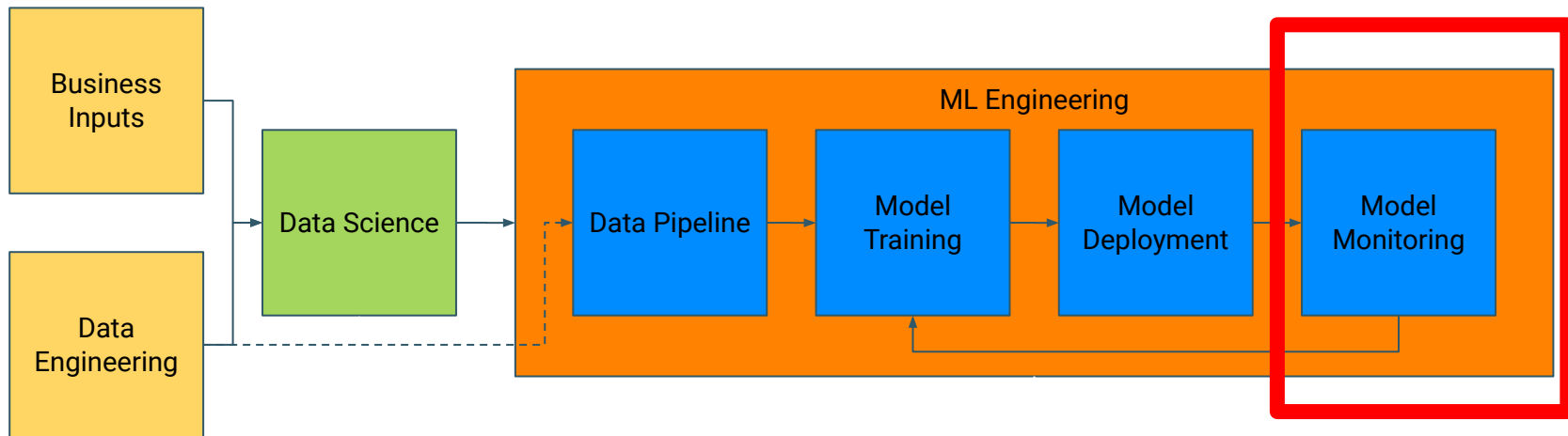
!python3 app.py
```



Lab 4

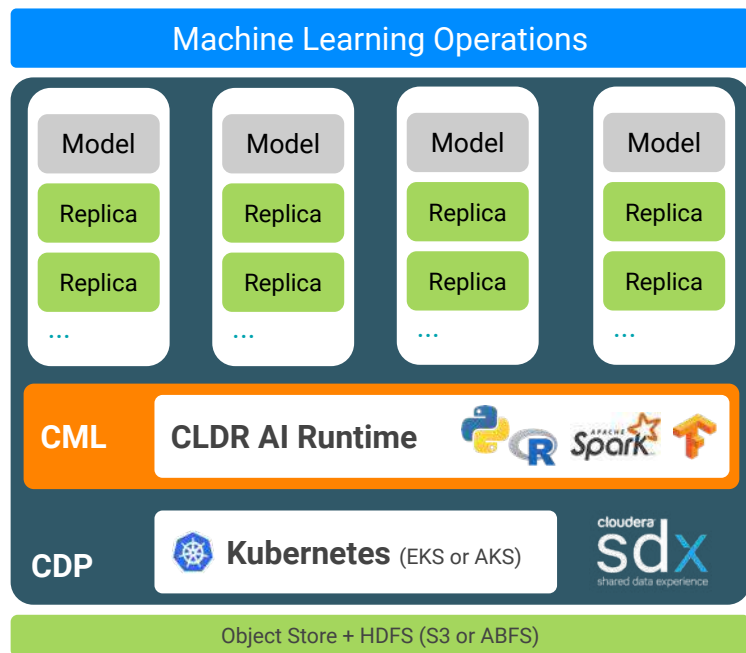
Git : <https://github.com/frenchlam/dash-svm.git>

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

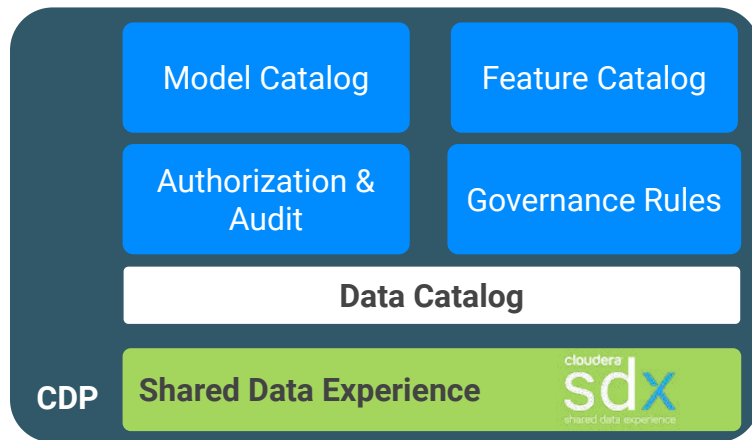
A look ahead - MLOPS: Governing hundreds of models

- **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

Investments in active research and development

Example only.

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...

Status

Active (35)

Users

All (7)

Lifecycle

All(35)

Clear

Models (35)

MODEL

LIFECYCLE

PIPELINES

ENVIRONMENT

FEATURES

OWNER

ACTIONS

Naive Bayes Classification

Staging

Sale Price Predictions

ML_Americas

3

Test Admin

🔍

⌵

Lineage

Assets

Audit History

Execution History

K-Classifier

Staging

Finance Forecast

ML_EU

4

Danny

🔍

⌵

TensorFlow Neural network

Prod

ML_Americas

2

Jon

🔍

⌵

Majority Class Classifier

Staging

Revenue Predictions

ML_Americas

12

Tyrien

🔍

⌵

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

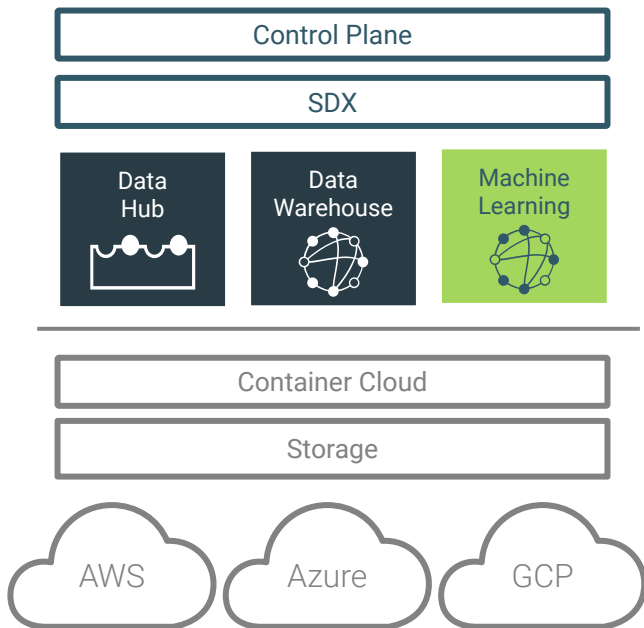
Secure root level Runtime customization for data scientists — no need for IT assistance

Support environment sharing between projects — enable data scientist collaboration workflows

Evolve the Platform to make it run anywhere

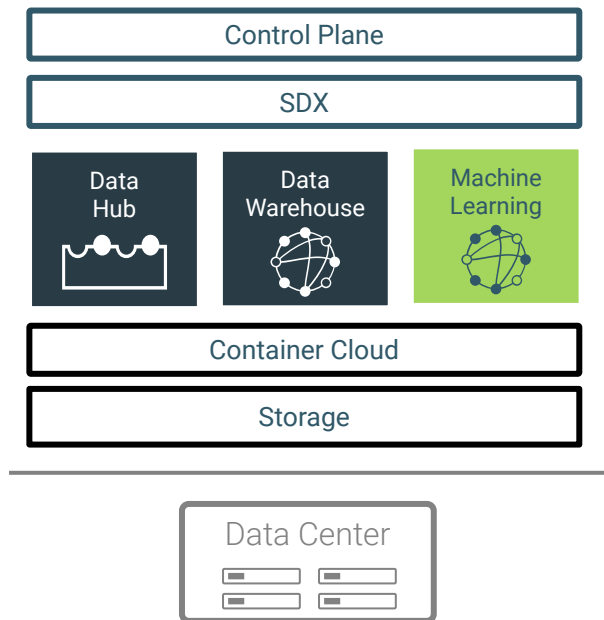
CDP Public Cloud

(platform-as-a-service)



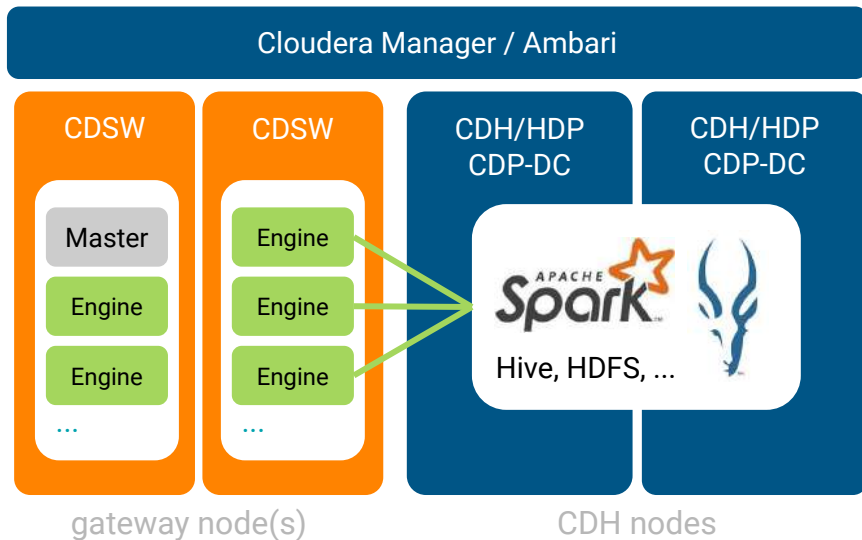
CDP Private Cloud

(installable software)



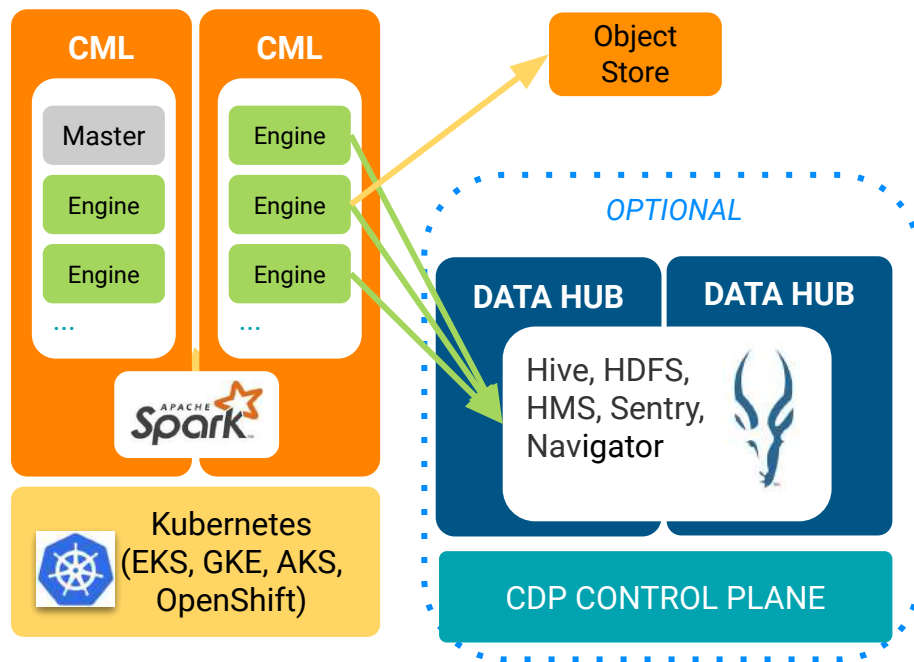
CDSW - SPARK ON YARN

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 HDP 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 ●●	CCA Developer CCP Data Engineer
DATA SCIENTIST	Data Science CDSW HDP ●●	Cloudera DS Workbench CDSW ●		CML CDP ●			

● Private

● Public Class

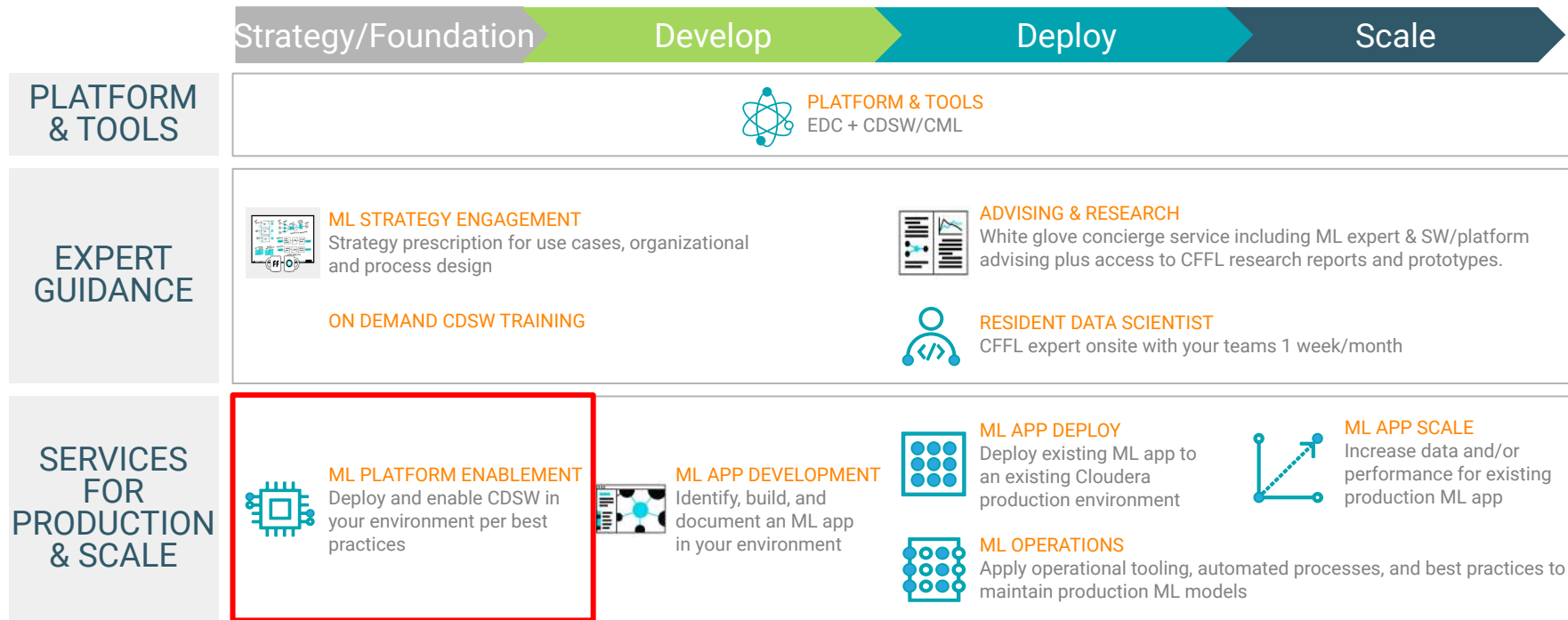
● OnDemand

● OnDemand (coming soon)

Consider PS to help onboard teams on CML

SMARTML

Cloudera's ML services for the journey to industrialized AI



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	<ul style="list-style-type: none">• Install and configure CDSW; KT with platform team• Dependency management (custom engines, parcels etc.)• Enable use of GPUs	<ul style="list-style-type: none">• Integrate with third-party editors• KT with DS/ML teams on customer's ML platform• Migrate projects from another data science tool
Outcomes	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• 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

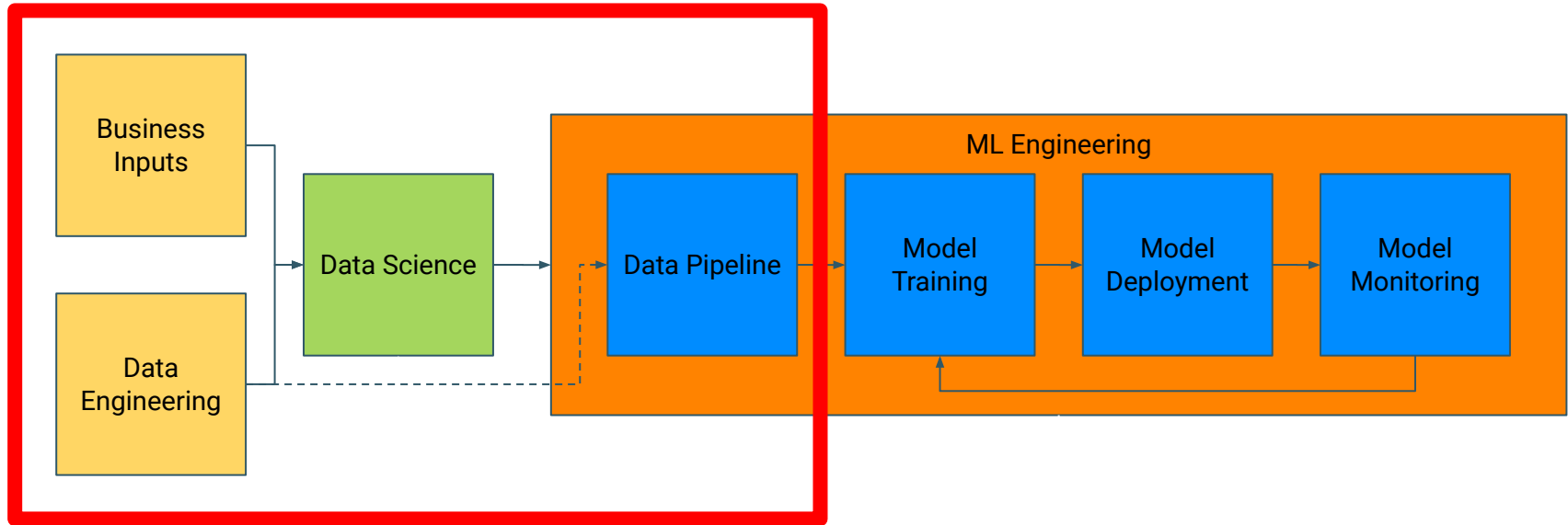
PRIVATE	PUBLIC	ONDEMAND	CERTIFICATION	TRAINING CREDITS
				
<ul style="list-style-type: none">✓ Role-based Training✓ Tailoring✓ Customization✓ Onsite or Virtual✓ Flexible dates	<ul style="list-style-type: none">✓ Role-based Training✓ Published Schedule✓ Global coverage✓ Multi-time zones✓ Training facility or Virtual	<ul style="list-style-type: none">✓ OnDemand Lab Access✓ 100% self-paced✓ High quality pre-recorded video & audio instruction	<ul style="list-style-type: none">✓ Performance-based hands-on exams✓ Role-based✓ Live proctor✓ Schedule exam 24/7✓ Digital badge	<ul style="list-style-type: none">✓ 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

The screenshot displays the Cloudera Data Science Workbench interface for the 'admin' user. The left sidebar contains navigation links: Projects, Sessions, Experiments, Models, Jobs, Settings, and Admin. The main content area shows a list of projects: 'Dash_example' (updated 5 hours ago), 'wine_pred' (updated yesterday), and 'Shiny-App_HelloWorld' (updated yesterday). Each project has a 'Star' button. Below the project list, the 'wine_pred' project is selected, showing options to 'Fork' or 'Open Workbench'. The 'Models' section displays a table with columns: Model, Status, Replicas, CPU, Memory, Last Deployed, and Actions. The 'wine_pred' model is listed with a 'Stopped' status, 0/1 replicas, 0 CPU, 0 GIB memory, and was last deployed on Sep 1, 2019, at 10:38 PM. The 'Jobs' section shows a table with columns: Name, Runs / Failures, Duration, Status, Latest Run, and Actions. The 'Analysis' job is listed with 2/0 runs, a duration of 00:56, a 'Success' status, and was last run yesterday. The 'Files' section shows a list of files and folders: 'data', 'Jupyter Notebook', 'models', 'setup', 'analyse.py', and 'cdsw-build.sh', with columns for Name, Size, and Last Modified.

Cloudera Data Science Workbench

admin

Project quick find

CDEP CREATED ACCOUNT admin

0 Followers

New Project

0 Star

0 Star

0 Star

wine_pred

0 Fork Open Workbench

Models

Model	Status	Replicas	CPU	Memory	Last Deployed	Actions
wine_pred	Stopped	0 / 1	0	0 GIB	Sep 1, 2019, 10:38 PM	Start

Jobs

Name	Runs / Failures	Duration	Status	Latest Run	Actions
Analysis	2 / 0	00:56	Success	yesterday	Run

Files

Download New Upload

Name	Size	Last Modified
data	-	yesterday
Jupyter Notebook	-	yesterday
models	-	yesterday
setup	-	yesterday
analyse.py	4.33 kB	yesterday
cdsw-build.sh	138 B	yesterday

WORKBENCH (EDITOR + SESSION)

Code Editing to : Analyse data / Develop Scripts / Applications

The screenshot displays the Cloudera Workbench interface. On the left is a dark sidebar with a file explorer showing a project structure with files like README.md, wine_pred, analyse.py, and a Jupyter Notebook named fit_SKLEARN.ipynb. The main area is split: the left pane shows a Python script named fit_GS_sklearn.py with line numbers 1 through 32, containing imports for sklearn, cdsw, numpy, and pandas, and a section for loading and processing data from a CSV file. The right pane, titled 'Start New Session', contains configuration options for 'Engine Image' (Base Image v8), 'Editor' (Workbench), 'Engine Kernel' (Python 3), and 'Engine Profile' (2 vCPU / 4 GiB Memory). At the bottom of this panel are two buttons: 'Launch Session' and 'Run Experiment'.

```
1 !pip3 install sklearn
2
3 import cdsw
4 import numpy as np
5 import pandas as pd
6
7
8 #- uncomment for experiments
9 # # Experiments
10 # ### Declare parameters
11 import ast #required to in order to parse arguments a lists
12 import sys
13 param_numTrees= ast.literal_eval(sys.argv[1])
14 param_maxDepth= ast.literal_eval(sys.argv[2])
15
16
17 # comment out when using experiments
18 param_numTrees = [70,80,90]
19 param_maxDepth = [5,10,15]
20
21
22
23 # ### Load the data (From File )
24 input_file = "data/WineNewGBDataSet.csv"
25 col_Names=["fixedAcidity",
26            "volatileAcidity",
27            "citricAcid",
28            "residualSugar",
29            "chlorides",
30            "freeSulfurDioxide",
31            "totalSulfurDioxide",
32            "density",
```

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

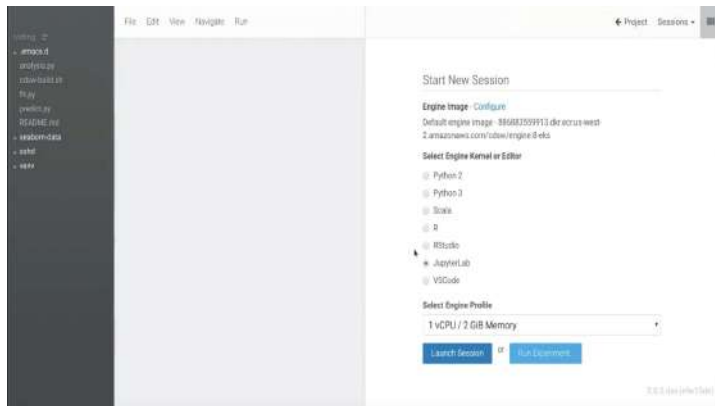
Other code-first data scientists

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



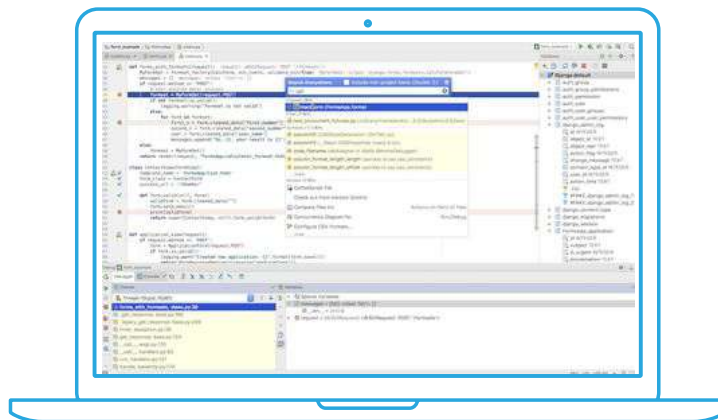
CDSW 1.6 THIRD PARTY EDITOR SUPPORT

Browser-based editors



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

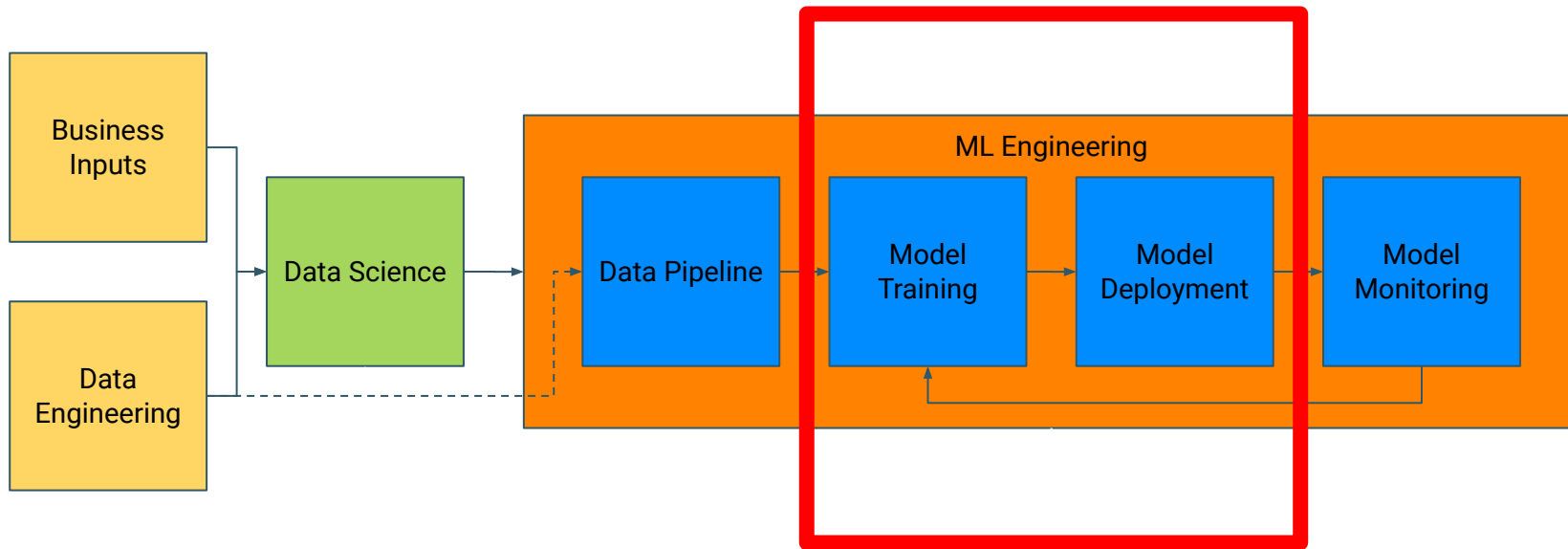
Local editors



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

The screenshot shows the Cloudera Data Science Workbench interface. On the left is a dark sidebar with a menu: 'All Projects', 'Overview', 'Sessions', 'Experiments', 'Models', 'Jobs' (highlighted), 'Files', 'Team', and 'Settings'. The main area has a top navigation bar with 'admin', 'wine_pred', 'Jobs', 'Analysis' (selected), and 'Settings'. Below this is the 'Analysis' configuration page with tabs for 'Overview', 'History', 'Dependencies', and 'Settings'. The 'General' section contains fields for 'Name' (Analysis), 'Script' (analyse.py), 'Engine Kernel' (Python 3 selected), 'Schedule' (Manual), 'Engine Profile' (1 vCPU / 2 GiB Memory), and 'Timeout in Minutes (optional)' (30). There is a 'Kill on Timeout' checkbox and a link to 'Set Environmental Variables'. At the bottom, the 'Job Report Recipients' section lists 'CDEP CREATED ACCOUNT' and 'Laurent Edel', each with checkboxes for 'Success', 'Failure', 'Stopped', and 'Timeout'.

Cloudera Data Science Workbench

admin wine_pred Jobs Analysis Settings

Analysis

Overview History Dependencies Settings

General

Name: Analysis

Script: analyse.py

Engine Kernel: Python 2, Python 3, R, Scala

Schedule: Manual

Engine Profile: 1 vCPU / 2 GiB Memory

Timeout in Minutes (optional): 30 Kill on Timeout

Jobs exceeding timeout send warning email if notifications enabled.

[Set Environmental Variables](#)

Job Report Recipients

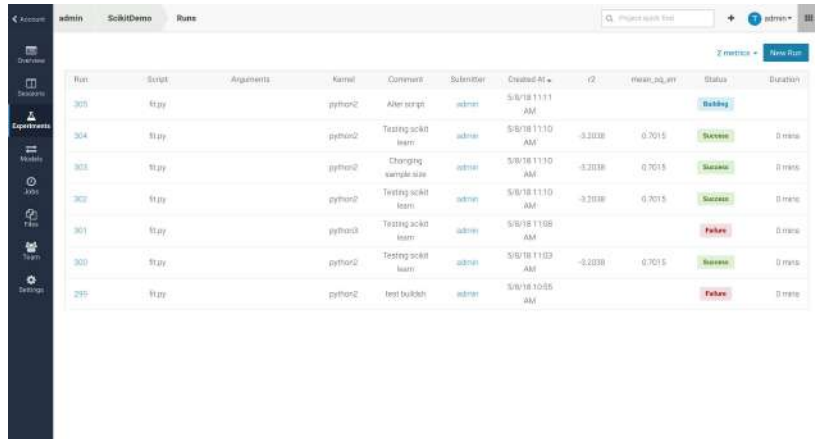
CDEP CREATED ACCOUNT	Success	Failure	Stopped	Timeout
Laurent Edel	Success	Failure	Stopped	Timeout

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



The screenshot displays the Cloudera Experiments interface. On the left is a dark sidebar with navigation icons for Overview, Experiments, Models, Jobs, Files, Teams, and Settings. The main panel shows a table of runs under the 'admin' user and 'SckidDemo' project. The table has columns for Run ID, Script, Arguments, Kernel, Comment, Submitter, Created At, r2, mean_rmse, Status, and Duration. The first run (303) is in 'Building' status. Subsequent runs (304, 305, 302, 301, 300) are 'Success', while the last run (299) is 'Failure'.

Run	Script	Arguments	Kernel	Comment	Submitter	Created At	r2	mean_rmse	Status	Duration
303	fit.py		python2	After script	admin	5/6/18 11:11 AM			Building	
304	fit.py		python2	Testing sckid learn	admin	5/6/18 11:10 AM	-0.2038	0.7015	Success	0 mins
305	fit.py		python2	Changing sample size	admin	5/6/18 11:10 AM	-0.2038	0.7015	Success	0 mins
302	fit.py		python2	Testing sckid learn	admin	5/6/18 11:10 AM	-0.2038	0.7015	Success	0 mins
301	fit.py		python2	Testing sckid learn	admin	5/6/18 11:08 AM			Failure	0 mins
300	fit.py		python2	Testing sckid learn	admin	5/6/18 11:03 AM	-0.2038	0.7015	Success	0 mins
299	fit.py		python2	test bulksh	admin	5/6/18 10:55 AM			Failure	0 mins

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.

The screenshot displays the Cloudera Machine Learning (CML) web interface. The top navigation bar includes 'Account', 'admin', 'First project', 'Models', 'Stock Analysis', and 'Overview'. A search bar on the right says 'Project quick find'. The left sidebar contains icons for Overview, Sessions, Models, Jobs, Files, Train, and Settings. The main content area is titled 'Stock Analysis' and has tabs for Overview, Deployments, Monitoring, and Settings. The 'Overview' tab is active, showing a description of the 'Stock Analysis' model. Below the description is a 'Sample Code' section with buttons for Shell, Python, and R. A code block shows a curl command to call the model's REST API. The 'Test Model' section has an 'Input' field with a JSON payload and 'Test' and 'Reset' buttons. The 'Result' section shows the status as '200 OK' and the response as a JSON object indicating success and a 'high' prediction. On the right, a 'Model Details' table lists various attributes.

Model Details	
Model Id	10
Deployment	5
Build	6
Deployed By	1
Comment	Initial build for Stock A
Kernel	python2
Engine Image	Base Image v4
File	example.py
Function	predict
CPU	0.25 core
Memory	1792 MB
GPU	0 GPU
Replicas	1 (fixed)

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.sfrmlamairese-4.vpc.cloudera.com

!python3 app.py
```

