CLOUDERA

Spark 3 & Iceberg in CDP

Overview & Demo

Paul de Fusco, Data Services Specialist

Sept 13th, 2024

AGENDA

Slides & Demos: Spark 3 Updates & New Features

Slides & Demos: Apache Iceberg with PySpark

Spark 3 Updates & New Features

SPARK 3

Updates

- CDP, Python, Java, Scala versions
- Spark Connector Support Changes
- Refactoring Recommendations
- Pre-Migration Planning Workbook (TP)

CDP, Python, Java, Scala Versions

More FYI's in Workbook

Scala Version Change

- Spark Scala applications must be recompiled with Scala 2.12 and adjust the dependencies to use Spark 3 version binaries provided by Cloudera in public maven repository and Scala 2.12 version of third-party libraries.
- Scala version change can require source code changes, please refer to the Scala language manual.

CDP, Python, Java Version Support

- Spark 3.3.0: CDP 7.2.16, CDS 3.3 for CDP 7.1.8 supports Python 3.7 to 3.10, Java 8, 11
- Spark 3.3.1: CDP 7.2.16, CDS 3.3 for CDP 7.1.8 supports Python 3.7 to 3.10, Java 8, 11, 17
- Spark 3.3.2: CDP 7.2.17, CDS 3.3 CHF 2 for CDP 7.1.8 supports Python 3.7 to 3.10, Java 8, 11, 17
- Spark 3.3.3: CDP 7.2.17, CDS 3.3 CHF 2 for CDP 7.1.8 supports Python 3.7 to 3.10, Java 8, 11, 17

Useful Links:

- Cloudera Spark, Scala, Python Supportability Matrix
- Cloudera Spark Python Supportability Matrix

Spark Connector Support Changes

More FYI's in Workbook

Spark 3 connectors are supported from certain versions. If Spark 2 connectors are used, please take the connectors into account when choosing the minimum CDP / CDS parcel version you need to upgrade to when migrating a Spark 2 application to Spark 3.

- Hive Warehouse Connector for Spark 3 is supported from CDP 7.1.8 + CDS 3.3.0 or CDP 7.2.18
- HBase connector for Spark 3 is supported from CDP 7.1.7 + CDS 3.2 or CDP 7.2.17
- Phoenix connector for Spark 3 is supported from CDP 7.1.8 + CDS 3.3.0 or CDP 7.2.18
- Oozie for Spark 3 is supported from CDP 7.1.9 + CDS 3.3.2 or CDP 7.2.18
- Solr for Spark 3 is supported from CDP 7.1.9 + CDS 3.3.2 or CDP 7.2.18

Refactoring Recommendations

More FYI's in Workbook

- **In Spark 3.2**, ShuffleBytesWritten and shuffleRecordsWritten are removed. Use bytesWritten and recordsWritten instead of ShuffleBytesWritten and shuffleRecordsWritten available in class org.apache.spark.status.api.v1.OutputMetrics.
- **In Spark 2.4**, org.apache.spark Class Accumulator is used. **In Spark 3.2**, org.apache.spark.util.AccumulatorV2 is used. Replace org.apache.spark.Accumulator with org.apache.spark.util.AccumulatorV2
- **In Spark 2.4**, For non-struct type, for example, int, string, array, **Dataset.groupByKey** results in a grouped dataset with key attribute is wrongly named as "**value**". **In Spark 3.2**, For non-struct type, for example, int, string, array, **Dataset.groupByKey** results to a grouped dataset with key attribute is named as "**key**". Check the code if the "value" attribute is used in logic, refactor the "value" attribute to "key". Or to preserve the old behavior, set ***spark.sql.legacy.dataset.nameNonStructGroupingKeyAsValue to false*.**
- **In Spark 2.4,** Path option is overwritten if one path parameter is passed to DataFrameReader.load(), DataFrameWriter.save(), DataStreamReader.load(), or DataStreamWriter.start(). **In Spark 3.2**, path option cannot coexist when the following methods are called with path parameter(s): DataFrameReader.load(), DataFrameWriter.save(), DataStreamReader.load(), or DataStreamWriter.start(). Remove the path option if it's the same as the path parameter, or add it to the load() parameter if you do want to read multiple paths. To ignore this check, set 'spark.sql.legacy.pathOptionBehavior.enabled' to 'true'.

Continued in Workbook

On the Pre-Migration Planning Workbook (TP)

Disclaimers

- **The workbook will be shared with you at the end of the session.**
- **The workbook is not to be used as a comprehensive guide to all potential Spark Migration Planning issues or topics.**
- **The Workbook is in Tech Preview and is an ongoing effort by Cloudera to include recommendations from different stakeholders. It is subject to change. Check with your account team for updates.**
- **Cloudera always recommends engaging Cloudera Professional Services in order to plan Spark Migrations in CDP.**

SPARK 3

New Features

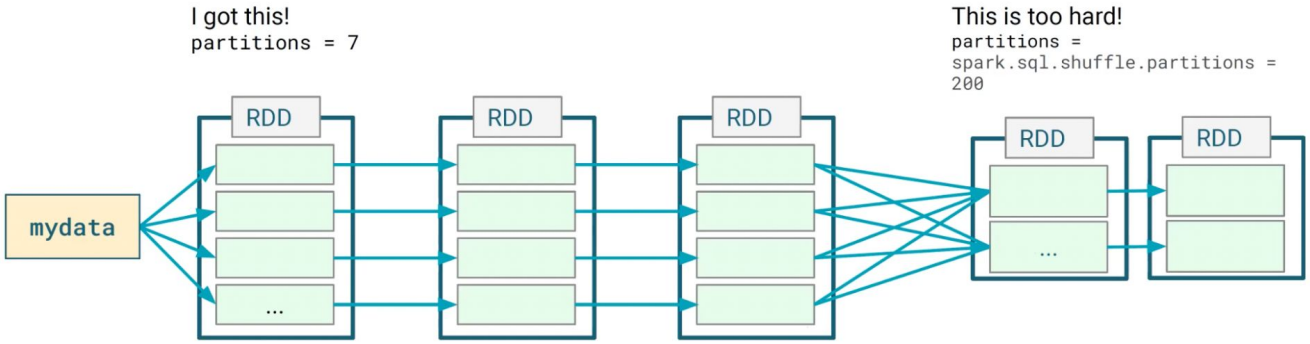
- Adaptive Query Execution
- Dynamic Partition Pruning

- Other Notable Features
  - GPU Support: CDP RAPIDS
  - Pandas on Spark API (aka Koalas)
  - Spark on Kubernetes

Adaptive Query Execution

Catalyst Optimizer Limitations

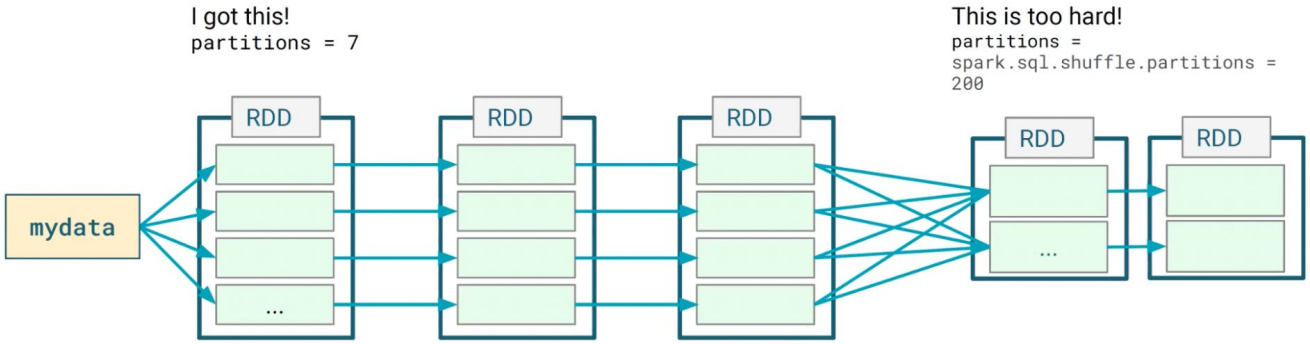
- The Catalyst Optimizer optimizes partitions for first DAG stage
- But uses a default setting of 200 for later stages



Adaptive Query Execution

Catalyst Optimizer Limitations

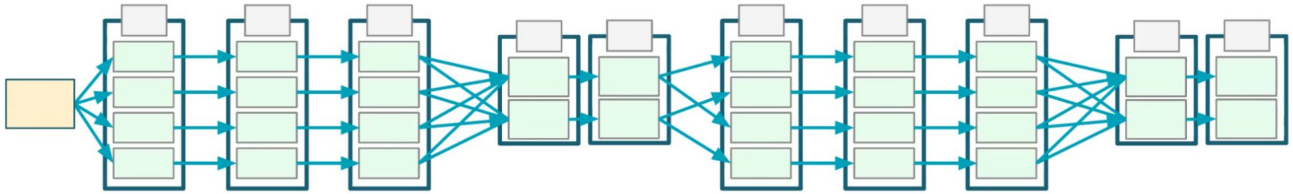
- Challenge: potentially suboptimal partitioning in later stages
- Solution in Spark 2: manually tune *spark.sql.shuffle.partitions*



Adaptive Query Execution

Catalyst Optimizer Limitations

- Even so, repartitioning is extremely difficult with complex DAGs
- You can only set *spark.sql.shuffle.partitions* only once before execution



Adaptive Query Execution

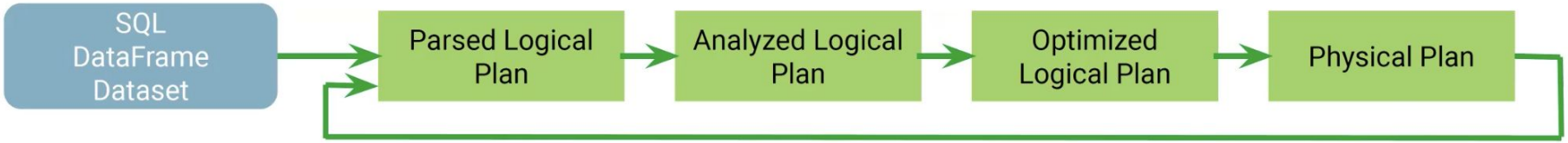
Catalyst Optimizer Limitations

- The plans are completely finalized before any execution
- Stages likely perform increasingly less optimally as the DAG gets larger
- Finding out information about data required to optimize the partitioning would require performing the transformations
- Hence the Catalyst Resorts to the “Magic Number” of 200 Default Partitions

Adaptive Query Execution

AQE Design Principle

- Break down the static monolithic plan into new “query stages” abstractions delimited by stages
- Allow Catalyst to plan for one query stage.
- Execute the query stage
- Plan the next query stage using results from the previous stage and apply all the following optimizations…



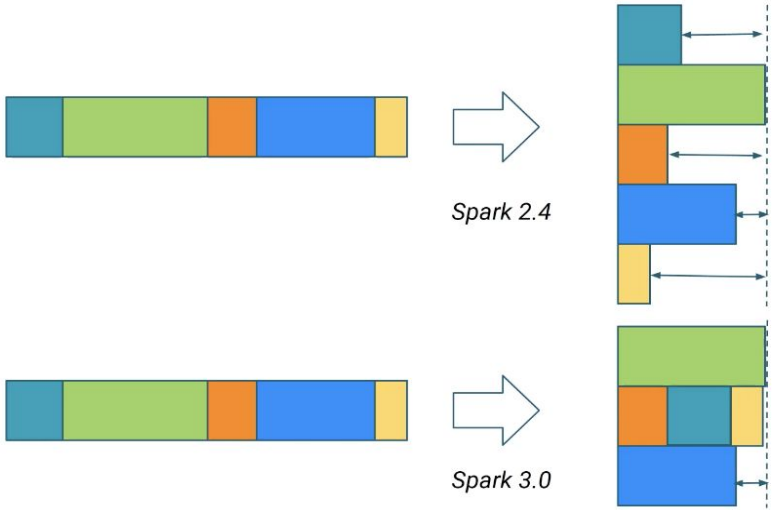
Adaptive Query Execution in Summary

AQE is a layer on top of the CO that will modify the Spark Plan on the Fly

- Coalescing Post Shuffle Partitions
  - The number of shuffle partitions is automatically adjusted (no longer defaulting to 200)
- Handling Skew Joins
  - Spark recognizes skew in the query and then automatically redistributes the data to join faster.
- Converting sort-merge join to broadcast join
  - AQE converts sort-merge join to broadcast hash join when the runtime statistics of any join side is smaller than the adaptive broadcast hash join threshold
- On by default in Spark 3.2
  - Or enabled via *sparkConf.set("spark.sql.adaptive.enabled", "true")*

Adaptive Query Execution

Dynamically Coalesce Shuffle Partitions



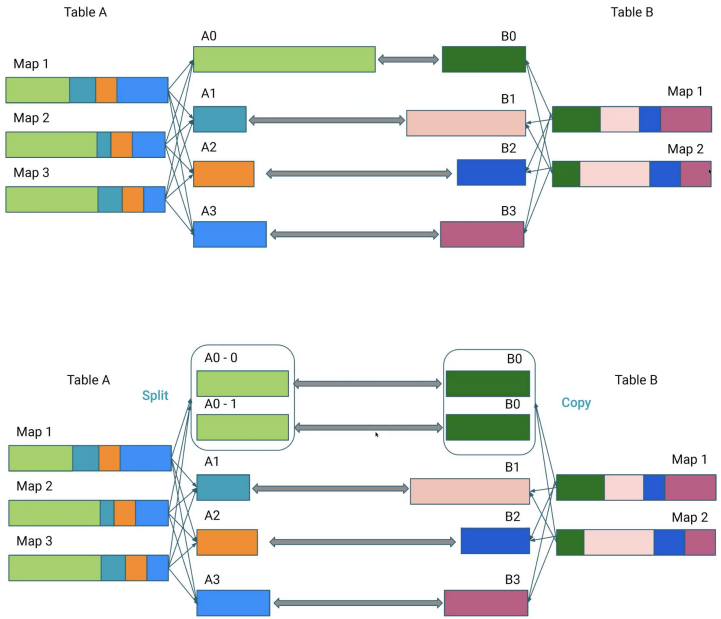
If the number of shuffle partitions is greater than the number of group by keys a lot of CPU cycles are lost due to unbalanced key distribution

When both properties are True, Spark will coalesce contiguous shuffle partitions according to the target size

spark.sql.adaptive.enabled -> True
spark.sql.adaptive.coalescePartitions.enabled -> True

Adaptive Query Execution

Optimizing Skew Joins with AQE



A Dataset is considered to be skewed for a Join operation when the distribution of join keys across the records in the dataset is skewed towards a small subset of keys.

AQE can detect skew and break partitions down so they run into a separate job as “subpartitions”. No need to write extra code.

AQE is detecting skew with Partition A0 and creating two subpartitions A0 -1 and A0 - 2

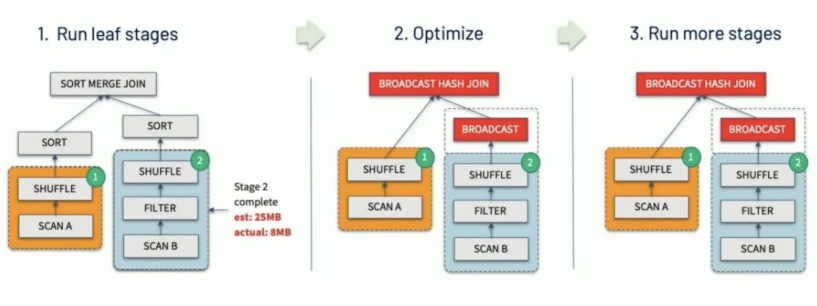
Adaptive Query Execution

AQE Skew Join Properties

Property Name	Default	Meaning
`spark.sql.adaptive.skewJoin.enabled`	True	When true and `spark.sql.adaptive.enabled` is true, Spark dynamically handles skew in sort-merge join by splitting (and replicating if needed) skewed partitions.
`spark.sql.adaptive.skewJoin.skewedPartitionFactor`	5	A partition is considered as skewed if its size is larger than this factor multiplying the median partition size and also larger than `spark.sql.adaptive.skewedPartitionThresholdInBytes`.
`spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes`	256MB	A partition is considered as skewed if its size in bytes is larger than this threshold and also larger than `spark.sql.adaptive.skewJoin.skewedPartitionFactor` multiplying the median partition size. Ideally this config should be set larger than spark.sql.adaptive.advisoryPartitionSizeInBytes.

Adaptive Query Execution

Converting SHUFFLE-SORT-MERGE to BROADCAST HASH Joins



Spark Defaults Joins to SORT-MERGE which require shuffling data. The Broadcast Join stores the smaller DataFrame on the executors memory.

The Broadcast Join is more efficient as long as the smaller DataFrame is small enough to fit into Executors memory.

AQE allows you to set a threshold to automatically convert MERGE-SORT to BROADCAST JOIN.

spark.sql.adaptive.autoBroadcastJoinThreshold
(configures max size in bytes)

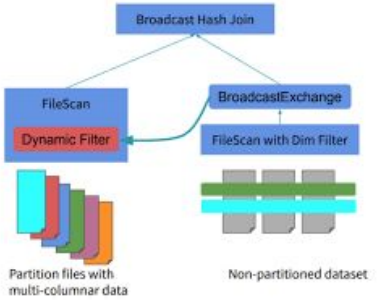
Dynamic Partition Pruning

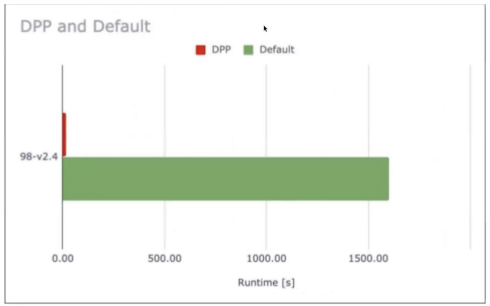
Increasing Spark Performance

- Read only the data you need
- DPP optimization implemented both on the logical plan optimization and the physical planning
- Create Logical Plan By: Column pruning, constant folding, filter push down
- Logical planning level to find the dimensional filter and propagate it across the join to the other side of the scan
- Physical level to wire it together in a way that this filter executes only once on the dimension side and then creates broadcast variable
- Then the results of the filter are directly reused for the table scan

Dynamic Partition Pruning

Filter / Predicate Pushdown + Broadcast Hash Join





The smaller table is queried and filtered. A hash table is built as part of the filter query.

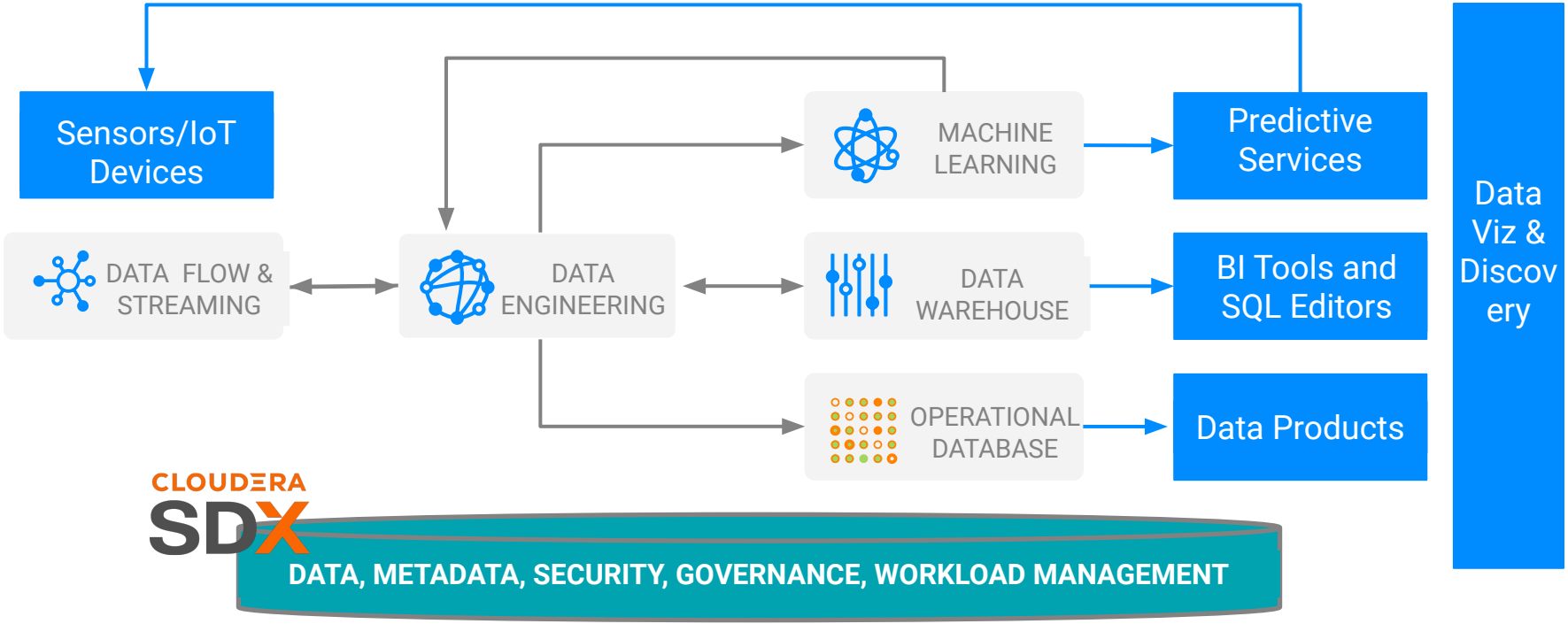
Spark uses the result of this query (and hash table) to create a broadcast variable. Then, it will broadcast the filter to each executor

At runtime, Spark's physical plan is changed so that the dynamic filter is applied to the bigger (fact) table.

This dynamic filter is created as an internal subquery built from the filter applied to the smaller table.

Apache Iceberg

WHAT INDUSTRIALIZED DATA ENGINEERING LOOKS LIKE

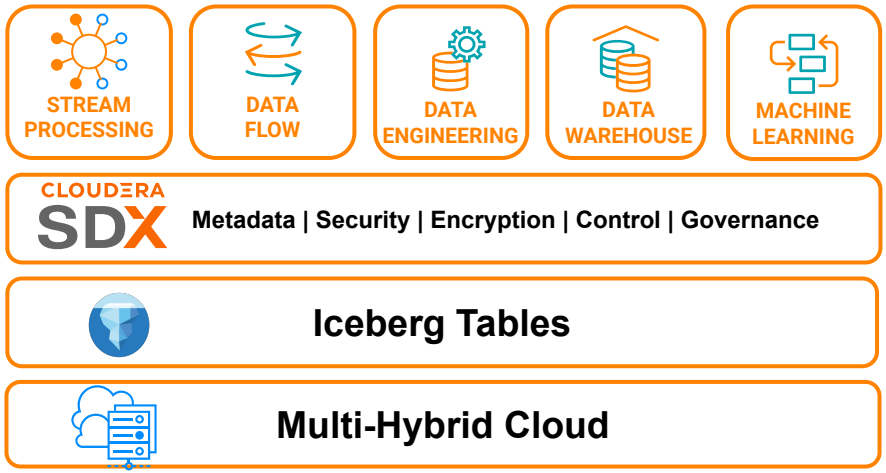


A Flexible, Performant & Scalable Table Format

APACHE ICEBERG

- Donated by **Netflix** to the Apache Foundation in 2018
- Flexibility
  - Hidden partitioning
  - Full schema evolution
- Data Warehouse Operations
  - Atomic Consistent Isolated Durable (ACID) Transactions
  - Time travel and rollback
- Supports best in class SQL performance
  - High performance at Petabyte scale

Cloudera's Open Data Lakehouse



- Multi-function analytics for Streaming, Data Engineering, Data Warehouse and AI/ML with integrated data services
- Performant, scalable, and flexible for PB scale data
- Common security and governance policies and data lineage with SDX integration
- Common dataset with all CDP analytics engines without data duplication and replication
- Deployment freedom with Multi-Hybrid Cloud

Introducing **Apache Iceberg** in CDP

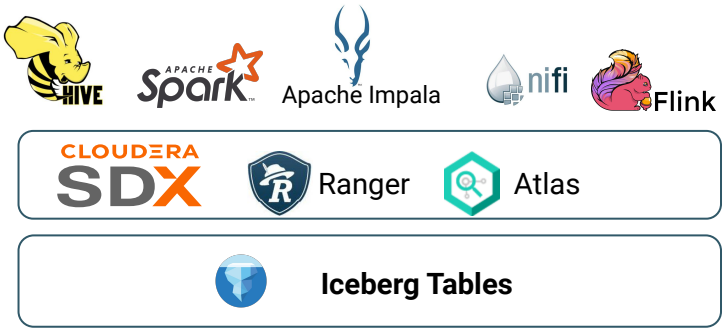
A Comparison with Hive

	Iceberg	Hive ACID
ACID Transaction	✔	✔
Partition Evolution	✔	✘
Schema Evolution	✔	✔
Time Travel	✔	✘
Major Users & Contributors	Netflix, LinkedIn, Tencent, Alibaba, Dremio, Apple, Cloudera, ...	Cloudera, Qubole
Compute Engines	Spark, Presto, Flink, Dremio, Hive; Impala, and Nifi forthcoming	Primarily Hive; Presto; read support in Impala; Spark through HWC

Apache Iceberg is selected for CDP integration due to 100% open table format, strong OS community engagement, and a growing list of compute engine support

Compute Engines Interoperability & Fine Grained Access Control

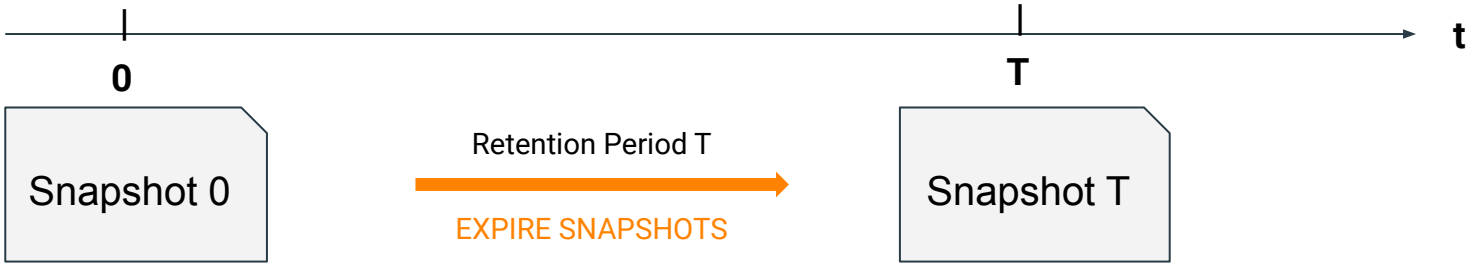
Compute Engine Interoperability & SDX Integration



- **Snapshot isolation** ensures **consistent** data access and processing with various compute engines including **Hive**, **Spark**, **Impala**, **Flink** and **Nifi**
- **Security & Governance** support (e.g. FGAC) through **Ranger** integration
- Data **lineage** support through **Atlas** integration

Time Travel and Table Rollback

Time Travel & Table Rollback



Standard SQL operations:

- Queries
- DDL
- DML

Time Travel & Rollback operations:

- SELECT … AS OF …
- ALTER TABLE … EXECUTE ROLLBACK ( )…

- **Time Travel** enables reproducible queries that use exactly the same table snapshot, or lets users easily examine changes
- **Rollback** allows users to quickly correct problems by resetting tables to a good state

Time Travel & Table Rollback Operations

	SQL Examples

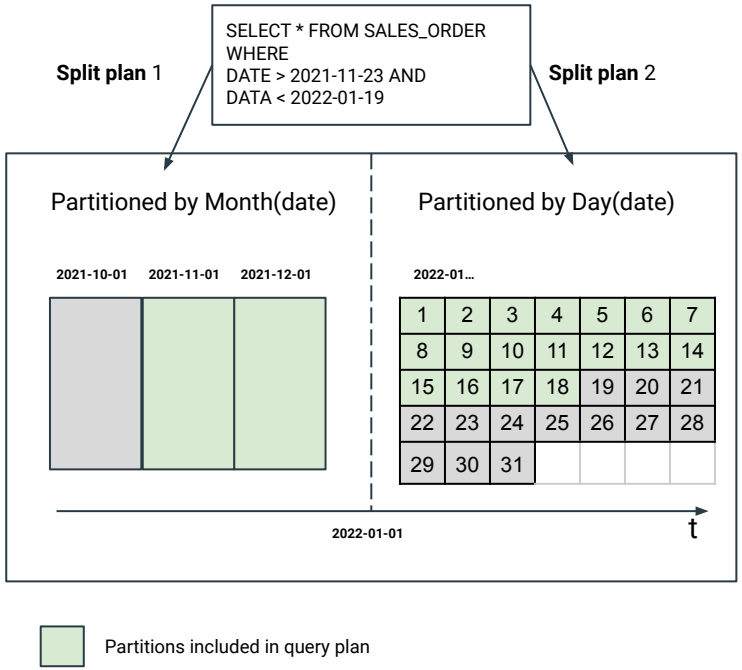
Time Travel	`SELECT * FROM table FOR SYSTEM_TIME AS OF '2021-08-09 10:35:57';`
	
	`SELECT * FROM table FOR SYSTEM_VERSION AS OF 1234567;`

Table Rollback	`ALTER TABLE table EXECUTE ROLLBACK ('2021-08-09 10:35:57')`
	
	`ALTER TABLE table EXECUTE ROLLBACK (1234567)`

***Describe History** SQL statement for timestamp and snapshot ID

In-Place Partition Evolution

In-Place Partition Evolution



- Existing big data solution doesn't support **in-place** partition evolution. Entire table must be completely rewritten with new partition column
- With Iceberg's **hidden partition**, a separation between physical and logical, users are not required to maintain partition columns.
- Iceberg tables can **evolve partition** schemas over time as data volume changes.
- **Benefits**:
  - No costly table rewrites or table migration
  - No query rewrites
  - Reduce downtime and improve SLA

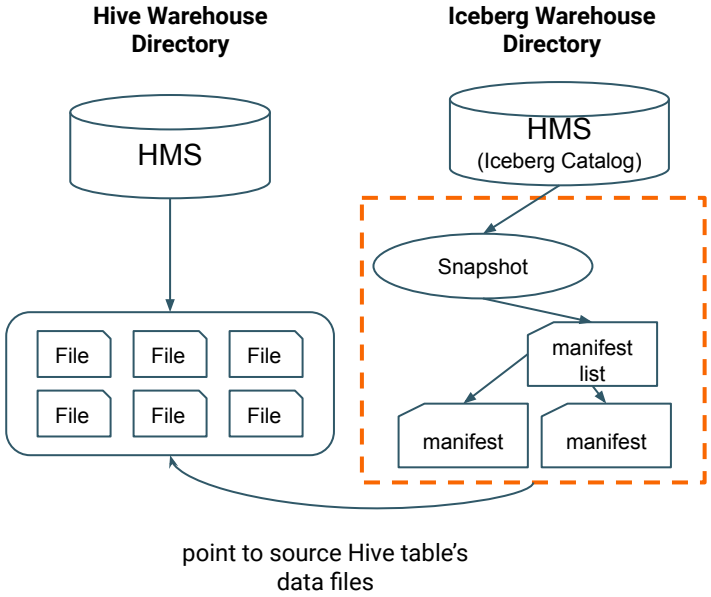
In-Place Partition Evolution SQL examples

Engine	SQL Examples
Hive / Impala	`// Partition evolution to hour` `ALTER TABLE t SET PARTITION SPEC (hour(ts))`
Spark SQL	`// Partition evolution to hour` `ALTER TABLE t ADD PARTITION FIELD (hour(ts))`

Table Migration

- from Hive to Iceberg table using SQL commands

Table Migration In-Place



Avoids rewriting data files, just **re-write** the metadata

- **Hive table migration**:

```
ALTER TABLE tbl SET TBLPROPERTIES
('storage_handler'='org.apache.iceberg.mr.hive.Hive
IcebergStorageHandler')
```

- **Spark 3:**

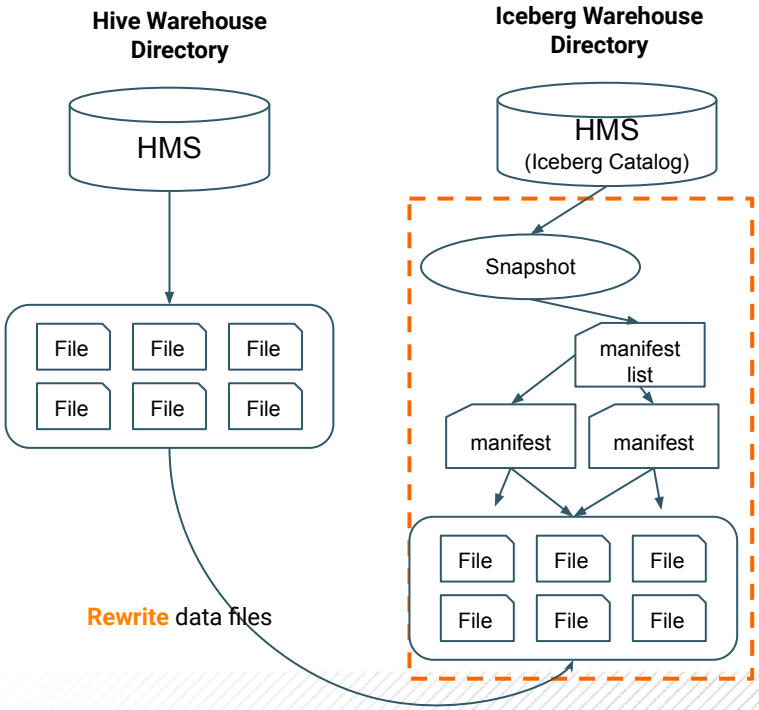
  a. Import Hive tables into Iceberg

```
spark.sql("CALL <catalog>.system.snapshot('<src>',
'<dest>')")
```

  b. Migrate Hive tables to Iceberg tables

```
spark.sql("CALL
<catalog>.system.migrate('<src>')")
```

Table Migration CTAS / RTAS



Data files **recreated** in addition to creation of Iceberg tables and corresponding metadata

```
CREATE TABLE ctas PARTITIONED BY(z) STORED
BY ICEBERG AS SELECT x, y FROM t;

spark.sql("CREATE TABLE ctas PARTITIONED
BY(z) USING iceberg AS (SELECT x, y FROM
t)")

spark.sql("REPLACE TABLE rtas PARTITIONED
BY(z) USING iceberg AS (SELECT x, y FROM
t)")
```

Other Notables

Spark On CDP Private Cloud (PVC) Base

ENTERPRISE GRADE MULTI-VERSION SPARK

Spark 3.3 GA
Spark 2.4

GPU ACCELERATED ETL WORKLOADS

Spark 3 RAPIDs integration
Up To 20X faster with zero code change

3RD PARTY INTEGRATION

Livy JDBC/Thrift Server support added with improved security & fault tolerance
Added Spark 3
Hue support SparkSQL

TRAINSMART: ENTERPRISE ENABLEMENT BUNDLE

Blend of Live , OnDemand Training & Certification

Live Training

- Dedicated sessions in which team can guide discussions to topics overtly relevant to internal goals.
- Access to tenured, expert instructors
- Access to curated Materials and Labs environment
- Covering Breadth of Topics for Multiple Audiences

OnDemand

- OnDemand Libraries provide comprehensive, and expanding, coverage of CDP and foundational technology topics
- One year, 100-hours lab time to practice in a safe environment.
- 24x7x365 access

Certification

- Role-based Certification
- Live proctor
- Schedule exam 24/7
- Digital badge

CLOUDERA TRAINING PORTFOLIO

Administrator	Administrator Private Cloud Base CDP 7.1 ●●	Administrator CDP Public Cloud ● AWS | Azure	Kafka Operations CDP 7.1 ●	Security CDP 7.1 ●			
Data Steward	Data Governance CDP ●●						
Data Analyst	Data Analyst: Hive/Impala CDH ● CDP ●	CDP Data Visualization ●* CDP					
Developer Data Engineer	Cloudera Data Engineering - Developing Applications with Apache Spark CDH ● CDP ●	Spark Performance Tuning CDP ●	NiFi Flow Management CDF ●●	CDF Stream Processing CDF ●	Architecture CDP ● (Q4)	HBase CDH ●●	Flink CDP ●
Data Scientist	CDSW CDSW ●	Data Science CDH ●	CML CDP ●				
General	**OnDemand Library** **CDH | HDP | CDP | CDF** ●	**"CDP"** Essentials, Pvt Cloud Fundamentals, HDP/CDH to CDP, AWS	**"Just Enough"** Python, Git, Scala	**Tech Overviews** CDW, Kudu, Kafka, Cloudera Manager, CDH on Azure	**HDP** Self Paced Library	**One Year Subscription** 100 hrs. Lab Time Message Forum Early Access Content	

● Live
● OnDemand

Demos

https://github.com/pdefusco/Spark3_Demo

THANK YOU

CLOUDERA

Why DE & Data Services

DATA SERVICES ARE THE FUTURE OF PVC

Compute-layer innovation will come in Data Services going forward

DATA CLUSTERS

SPARK, HIVE, IMPALA, CDSW

- Servers
- Monolithic
- Co-Located Storage & Compute
- HW Dependent
- Operator Focused
- Optimized for Existing Applications
- Forklift Upgrades
- Static Workloads

DATA SERVICES

DATA ENGINEERING

DATA WAREHOUSE

MACHINE LEARNING

- Services
- Modular
- Separated Storage & Compute
- SW Defined
- Practitioner Focused
- Optimized for New Applications
- Independent Upgrades
- Portable Workloads

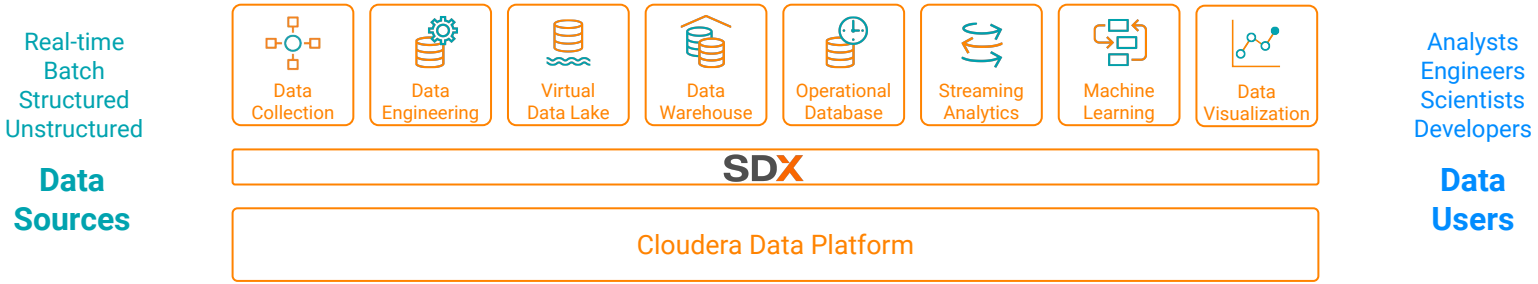
CLOUDERA

THE ENTERPRISE DATA CLOUD COMPANY

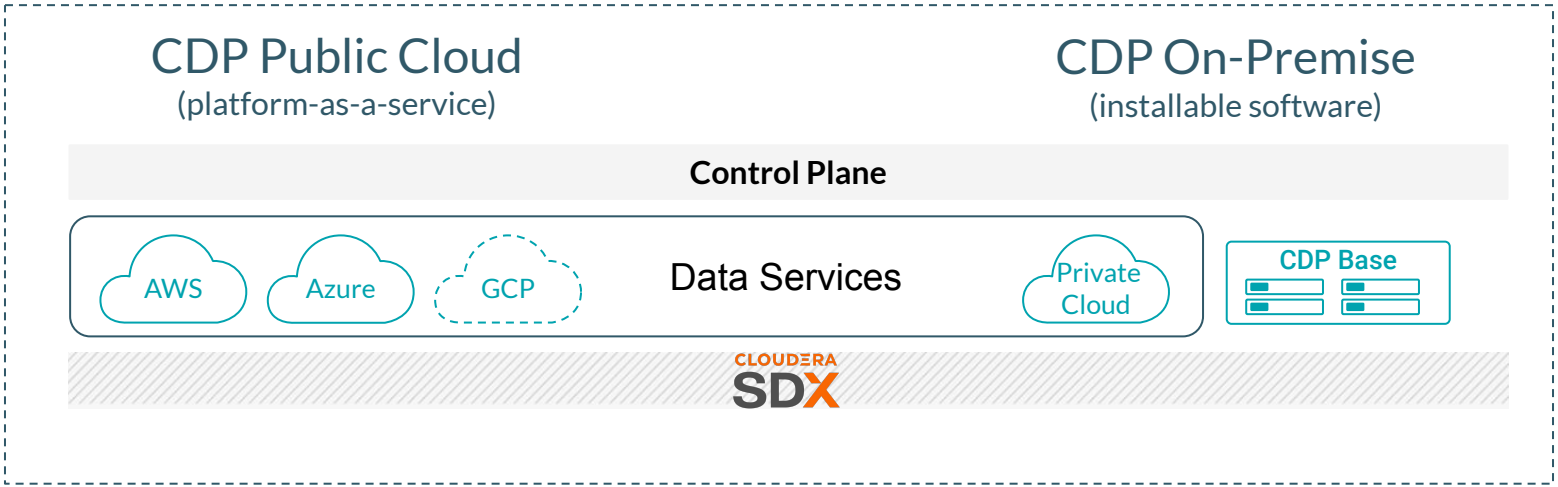
A HYBRID DATA CLOUD *AND* ANALYTICS FOR THE DATA LIFECYCLE

Storage & compute separation
Containers
SDX

Delivered as data architectures for key workload patterns



AVAILABLE IN PUBLIC CLOUD AND ON-PREMISE



PVC Data Services

PVC CDE VALUE ABOVE BASE

Strategic — Practitioner

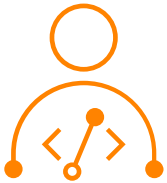
HYBRID PORTABILITY

Move workloads where needed:

across all of CDP, in public, private, hybrid or multi-cloud

IT / PLATFORM ADMINS

- Faster tenant on-boards w/**multi-version Spark** support
- Higher utilization of hardware
- **Per-tenant upgrades**



DATA ENGINEERS

- Centralized pipeline monitoring
- Better orchestration with **Airflow for modernizing** ETL pipelines
- **First class APIs** for DevOps automation

CDE PROVIDES AN ECOSYSTEM OF DE TOOLS AROUND SPARK

Self-service + DevOps meets DE

DevOps

First Class APIs

Next-gen Orchestration

Self-service

Performance profiling

Centralized Monitoring

Admin

Multi-version Spark

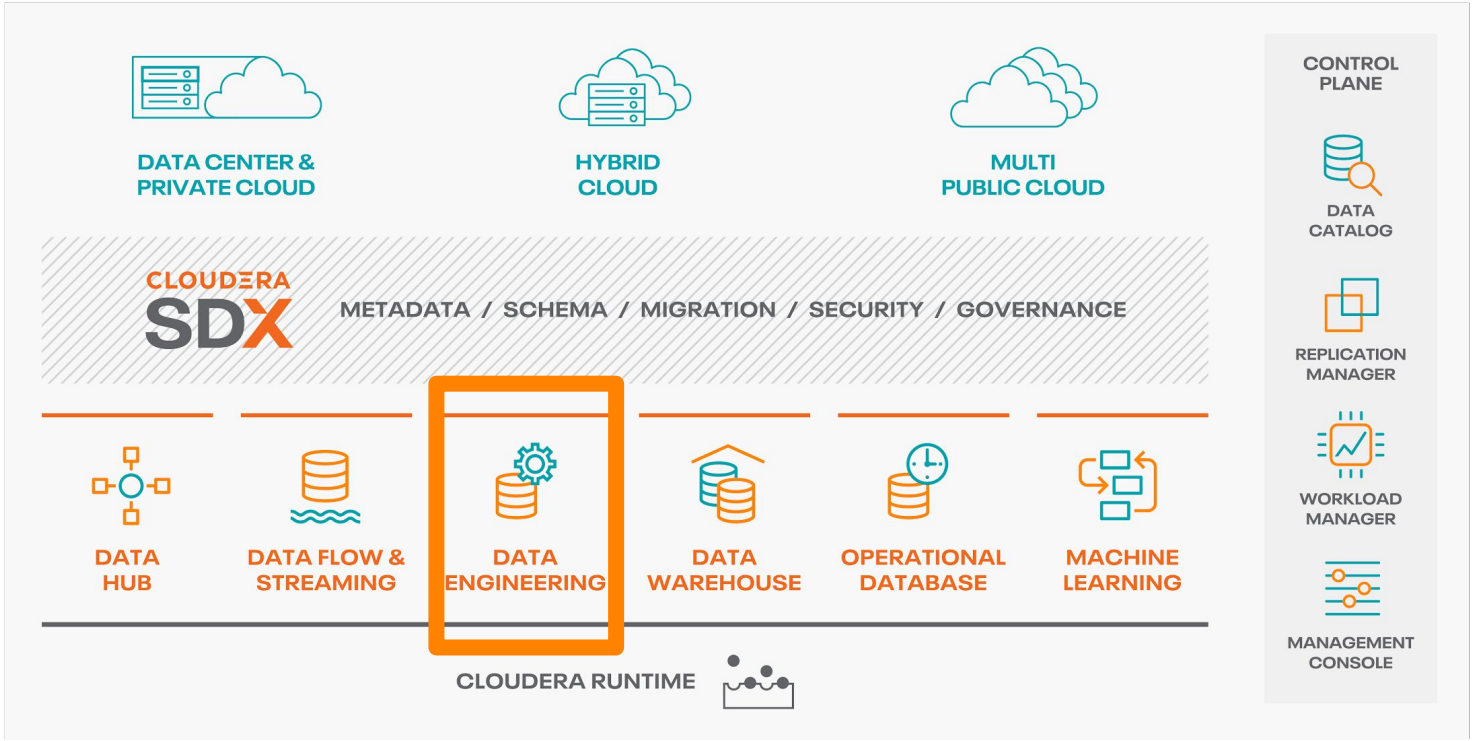
Tenant Isolation

Fast on-boarding

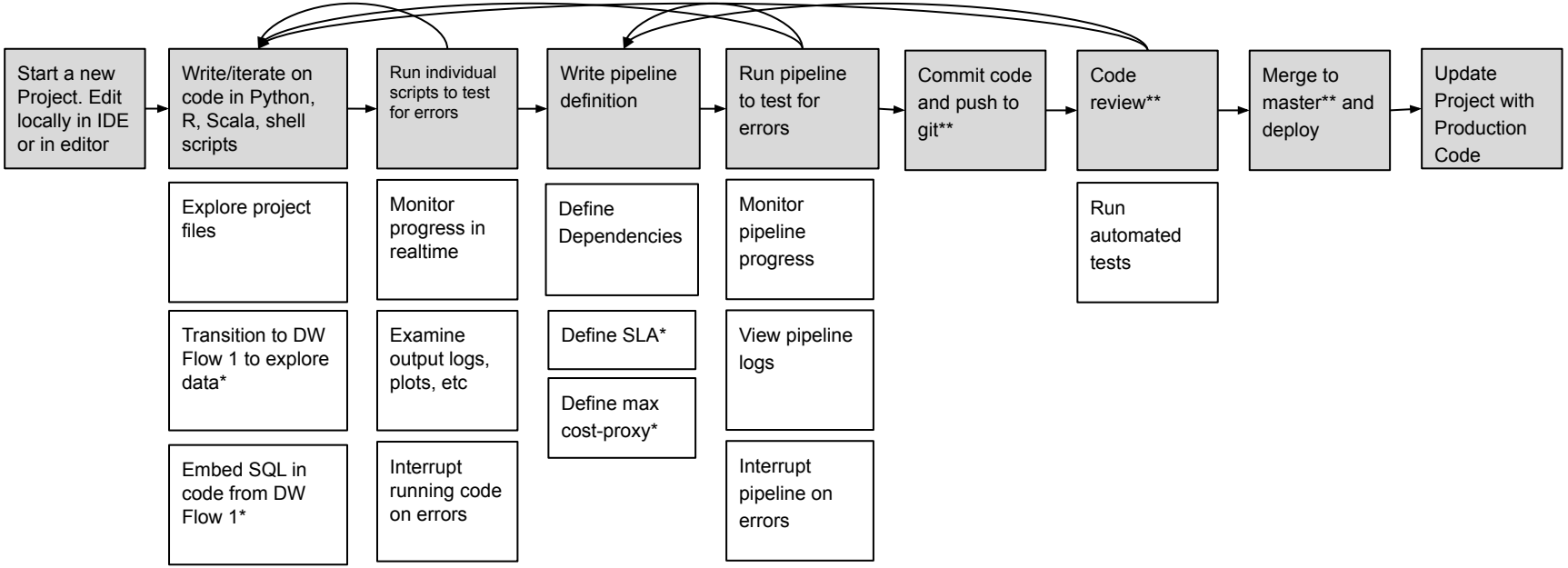
APACHE Spark™

Optimized spark on K8s

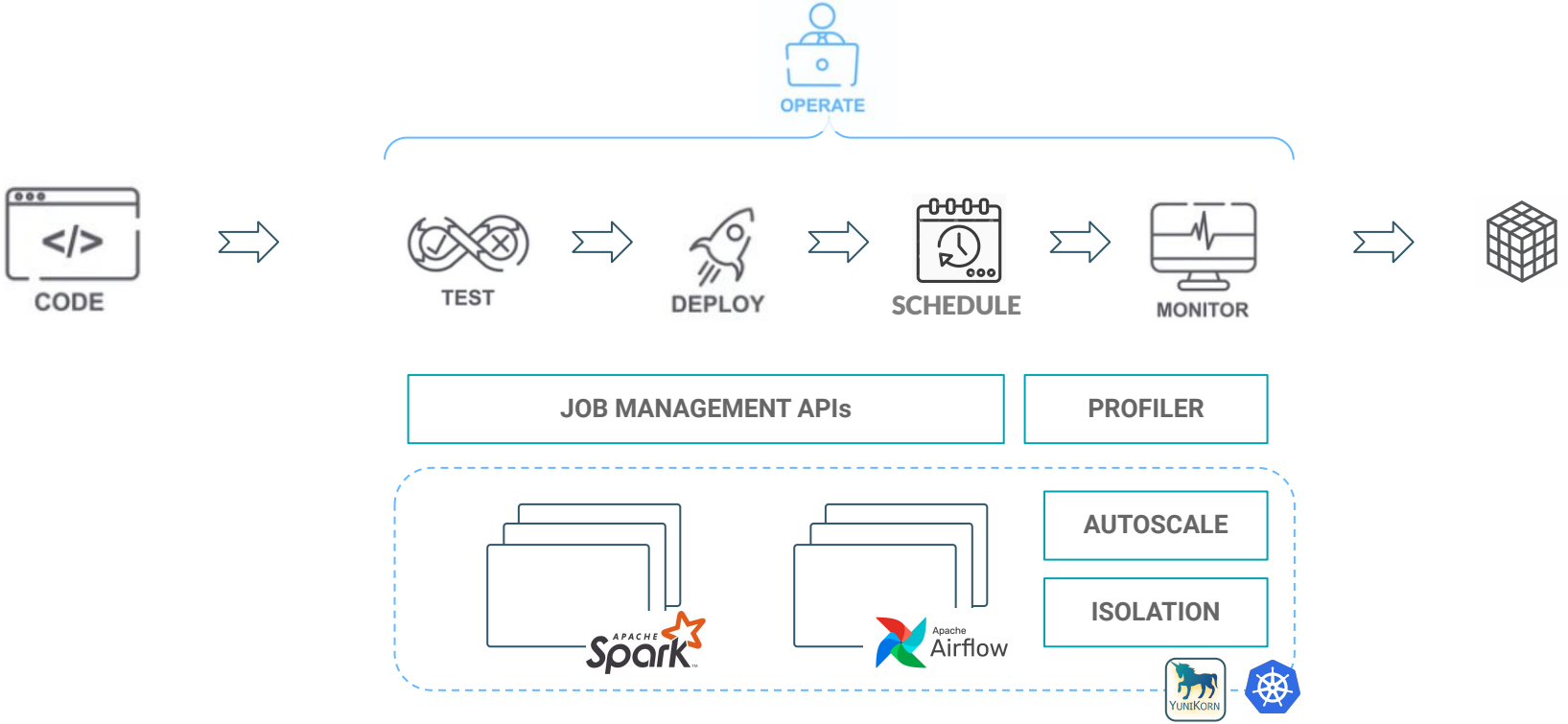
CDE: Run Spark Workloads in CDP Public/Private Cloud



Data Engineer Typical Challenges



Modernizing Data Ops with Cloudera Data Engineering



CLOUDERA DATA ENGINEERING DATA SERVICE

An Integrated, Purpose Built Experience for Data Engineers

CONTAINERIZED, MANAGED SPARK SERVICE

- Multiple version deployments
- Autoscaling compute
- Governed & secure with Cloudera SDX

APACHE AIRFLOW SCHEDULING

- Open preferred tooling
- Orchestrate complex data pipelines
- Manage & schedule dependencies easily

TUNING & VISUAL TROUBLESHOOTING

- Resolve issues fast with real-time visual performance profiling
- Complete monitoring & alerting capabilities

SIMPLIFIED JOB MANAGEMENT & APIS

- Full lifecycle mgmt.
- API-driven pipeline automation for any service
- Any language: SQL, Java, Scala, Python

CLOUDERA DATA ENGINEERING

An Integrated, Purpose Built Experience for Data Engineers

CONTAINERIZED, MANAGED SPARK SERVICE

- Autoscaling compute
- Governed & secure with Cloudera SDX
- Mix version deployments
- K8s optimized resource scheduling (Yunikorn)

What is Apache YuniKorn

An alternative to the K8s default scheduler

YuniKorn is a K8s scheduler, fully compatible with K8s APIs. Use as a drop-in replacement.

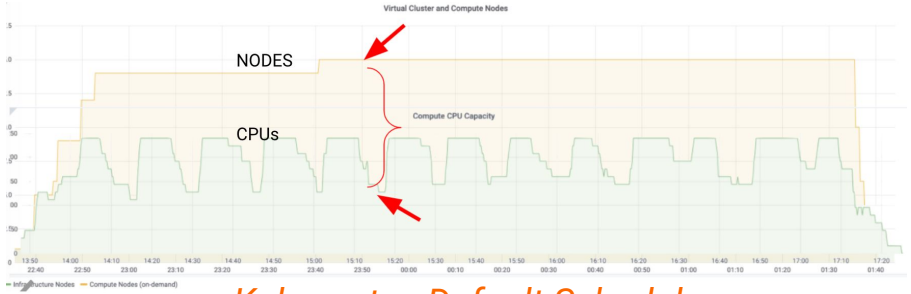
Built for Big Data on K8s

Bring YARN like resource management to K8s, multi-tenancy ready, and better performance.

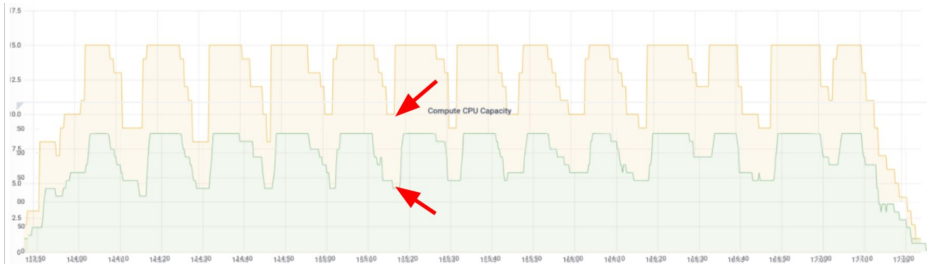
Yunikorn in action

Resource scheduling optimized for k8s

- Policies: FIFO, Gang, Bin-packing
- At scale
  - 200+ nodes, 24xlarge for ~5 days, with 1K+ executors



Kubernetes Default Scheduler



w/ Yunikorn

Performance

Schedule 50,000 pods on 2,000/4,000 nodes.

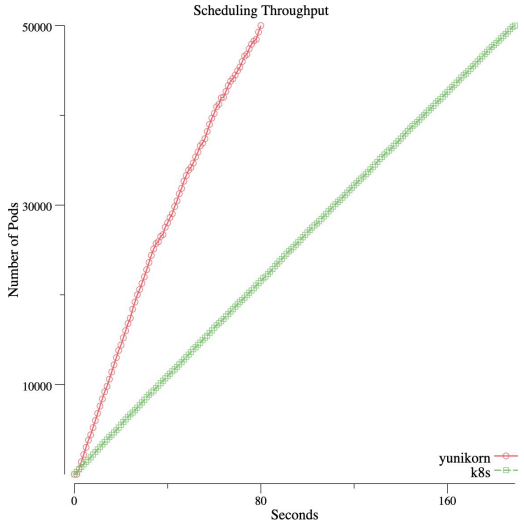
Compare Scheduling throughput (Pods per second allocated by scheduler)

Red line (**YuniKorn**)
Green line (Default Scheduler)

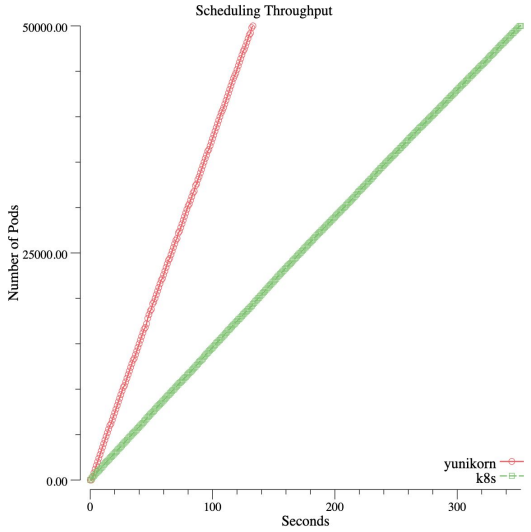
617 vs 263 ↑ 2.3x

373 vs 141 ↑ 2.6x

50k pods on 2k nodes



50k pods on 4k nodes

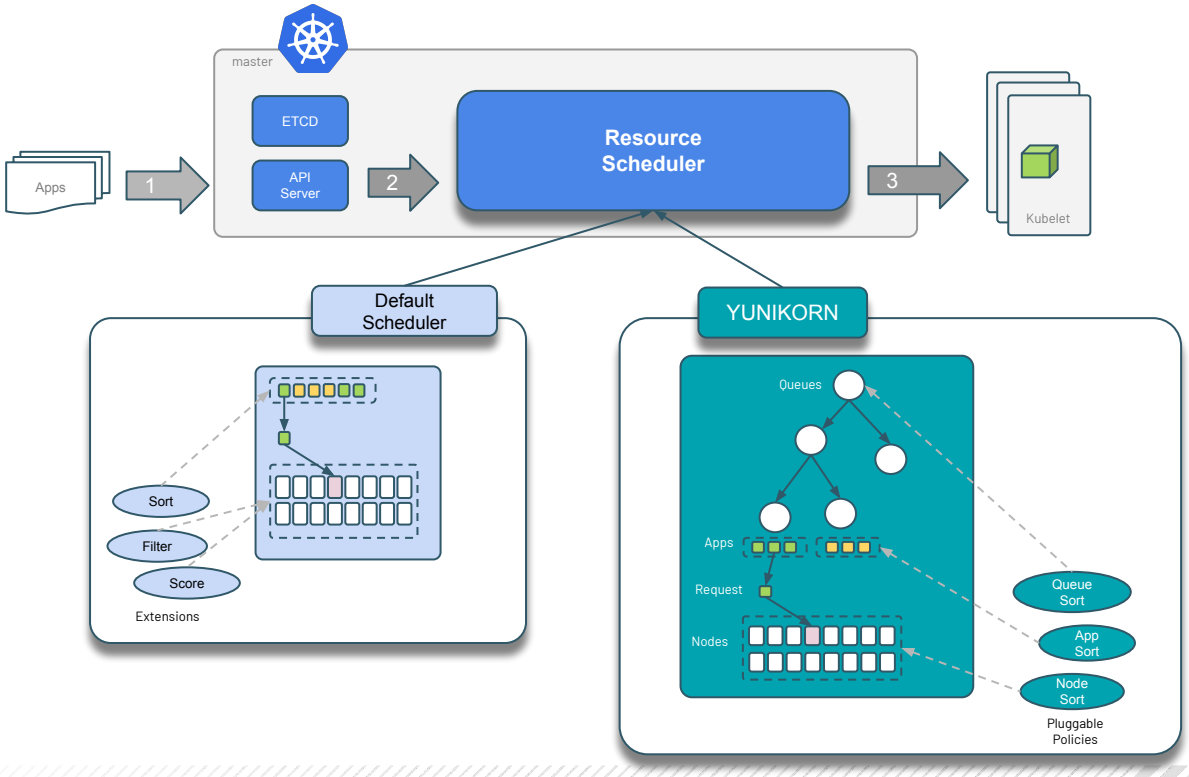


Detail report:
https://yunikorn.apache.org/docs/next/performance/evaluate_perf_function_with_kubemark/

Why is YuniKorn a Differentiator?

Feature	Default Scheduler	YUNIKORN	Note
Scheduling at app dimension	✗	✓	App is the 1st class citizen in YuniKorn, YuniKorn schedules apps with respect to, e,g their submission order, priority, resource usage, etc.
Job ordering	✗	✓	YuniKorn supports FIFO/FAIR/Priority (WIP) job ordering policies
Fine-grained resource capacity management	✗	✓	Manage cluster resources with hierarchy queues, queue provides the guaranteed resources (min) and the resource quota (max).
Resource fairness	✗	✓	Inter-queue resource fairness
Natively support Big Data workloads	✗	✓	The default scheduler is main for long-running services. YuniKorn is designed for Big Data app workloads, it natively supports Spark/Flink/Tensorflow, etc.
Scale & Performance	✗	✓	YuniKorn is optimized for performance, it is suitable for high throughput and large scale environments.

How Does YuniKorn work?



YuniKorn QUEUE, APP concepts are critical to provide advanced job scheduling and fine-grained resource management capabilities

CLOUDERA DATA ENGINEERING

An Integrated, Purpose Built Experience for Data Engineers

CONTAINERIZED, MANAGED SPARK SERVICE

- Autoscaling compute
- Governed & secure with Cloudera SDX
- Mix version deployments

APACHE AIRFLOW ORCHESTRATION

- Open preferred tooling
- Orchestrate complex data pipelines
- Manage & schedule dependencies easily

NEXT GEN ORCHESTRATION

An Integrated, Purpose Built for Data Engineers

Apache Airflow

SIMPLIFY PIPELINES

- Construct complex pipelines from simpler steps
- Easier to diagnose and optimize
- Modular & increase reusability

OPEN

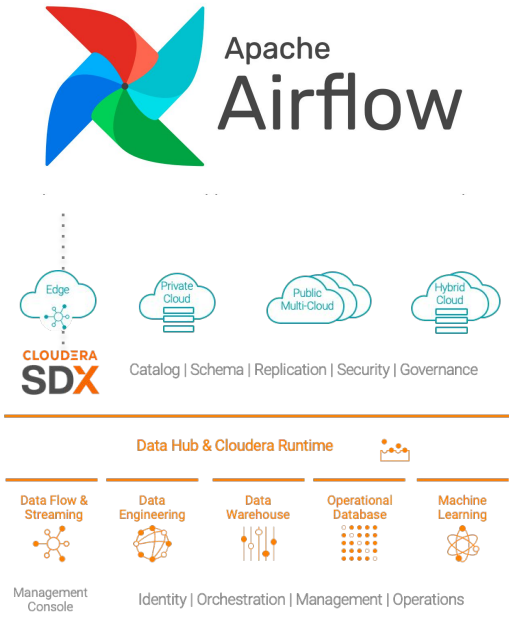
- Preferred tooling for DE and ML practitioners
- 100s of existing operators
- Strong community support

INTEGRATED LIFECYCLE

- Out of the box integration with CDP
- Flexibility to run both Spark and Hive jobs
- Extendible to other CDP end points

Top Reasons to Use Apache Airflow in CDE

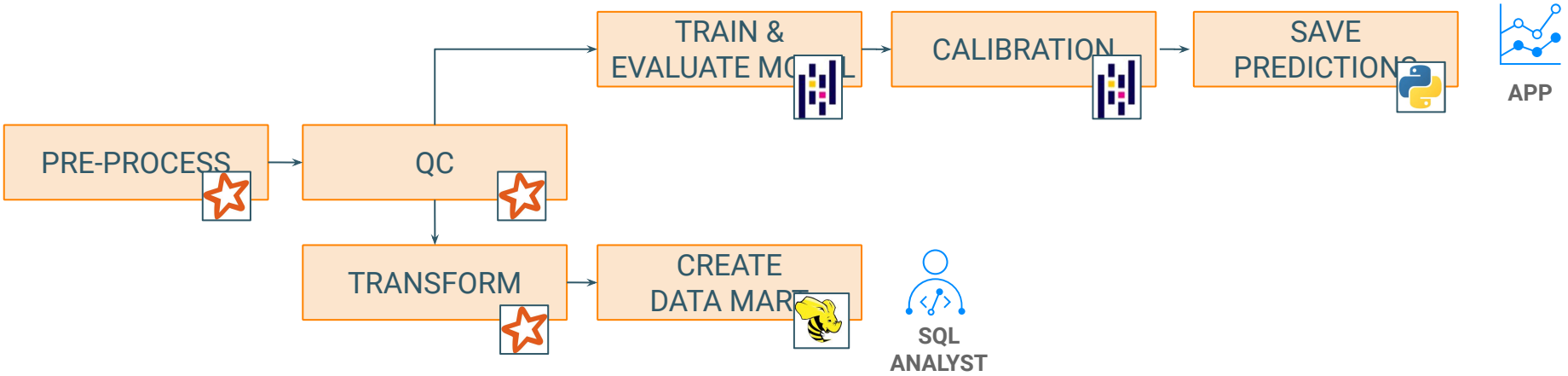
Built for multi-functional analytics anywhere



- **Python:** create arbitrarily complex pipelines using anything you can dream
- **Drag-n-drop UI:** self-service editor for no-code creation of multi-step pipelines
- **Managed Service**: Airflow in CDE does not require installing and operating the Airflow instance. It is managed for you by the CDE Service.
- **Fully Integrated with SDX:** CDE Airflow runs within a CDE Virtual Cluster and automatically inherits CDP Security and Governance
- **Airflow's rich web interface** provides an easy view for monitoring the results of your pipeline runs and debugging any failures that may have occurred.
- **Orchestrate Cross Service Pipelines:** trigger jobs in CDP Data Hub, CML, CDW, and virtually any Cloud or On-Prem service from a single CDE instance

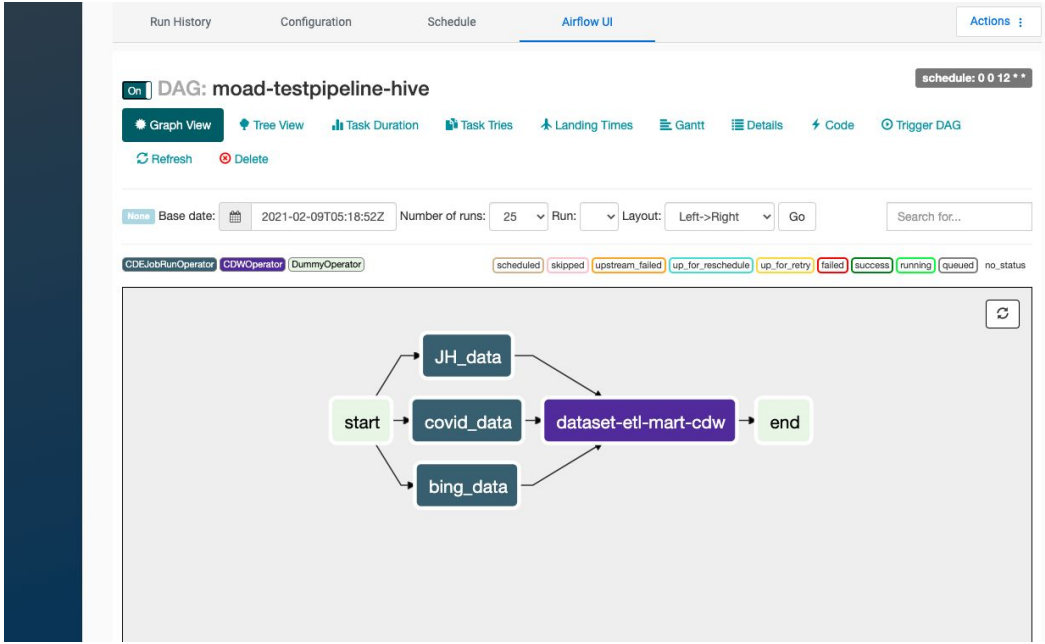
PIPELINE ORCHESTRATION

- Multi-step
- Mixed analytics
- Stateful (dependencies & branching)



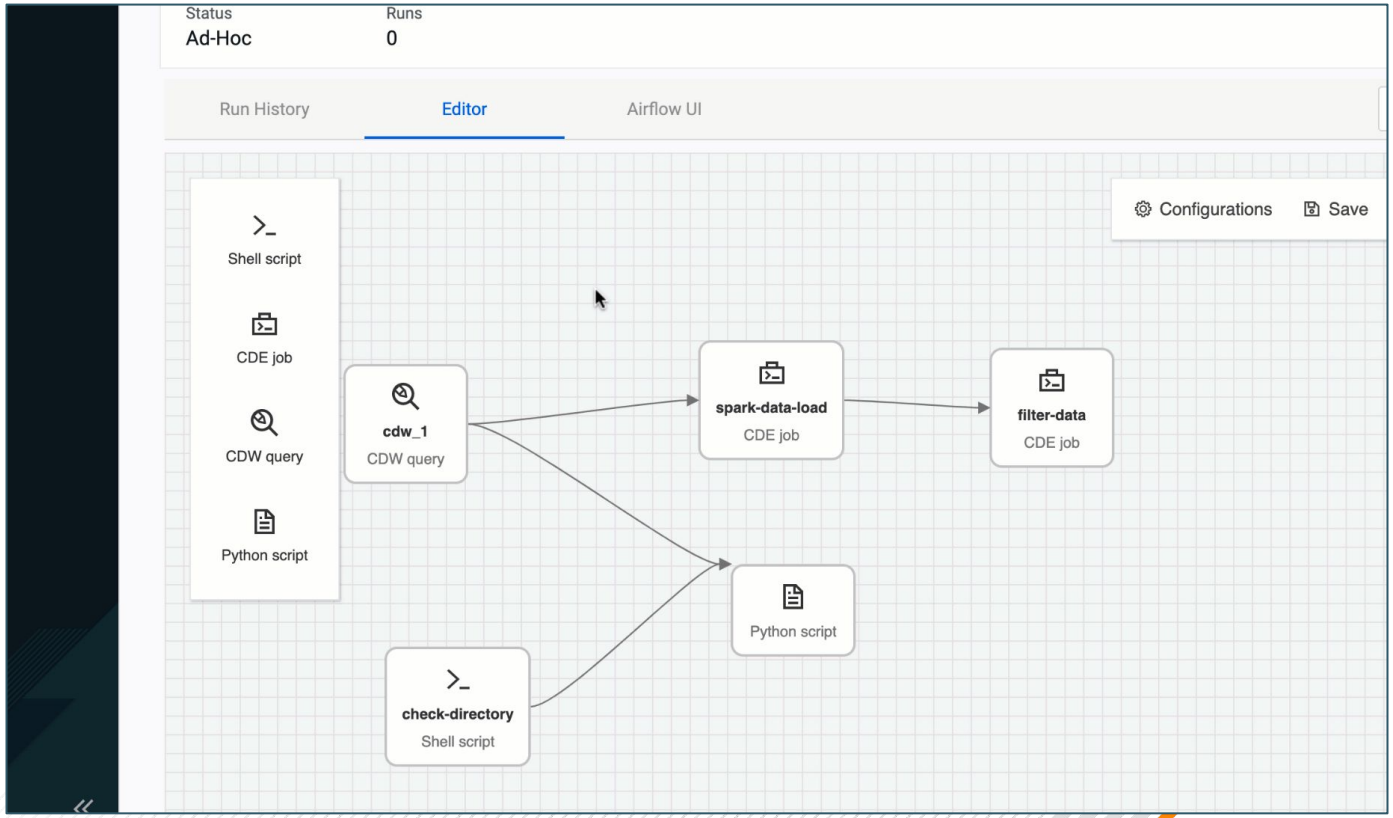
DEPLOY CUSTOM AIRFLOW DAGS

- Orchestrate Hive, Spark, Python, Bash scripts all in Airflow
- Open source CDP Providers & operators - CDE (spark), CDW (Hive)



https://github.com/cloudera/cloudera-airflow-plugins/tree/main/cloudera_airflow_provider

Self-service, low code / no-code Airflow **pipeline authoring**



CLOUDERA DATA ENGINEERING

An Integrated, Purpose Built Experience for Data Engineers

CONTAINERIZED, MANAGED SPARK SERVICE

- Autoscaling compute
- Governed & secure with Cloudera SDX
- Mix version deployments
- K8s optimized resource scheduling (Yunikorn)

APACHE AIRFLOW SCHEDULING

- Open preferred tooling
- Orchestrate complex data pipelines
- Manage & schedule dependencies easily

TUNING & VISUAL TROUBLESHOOTING

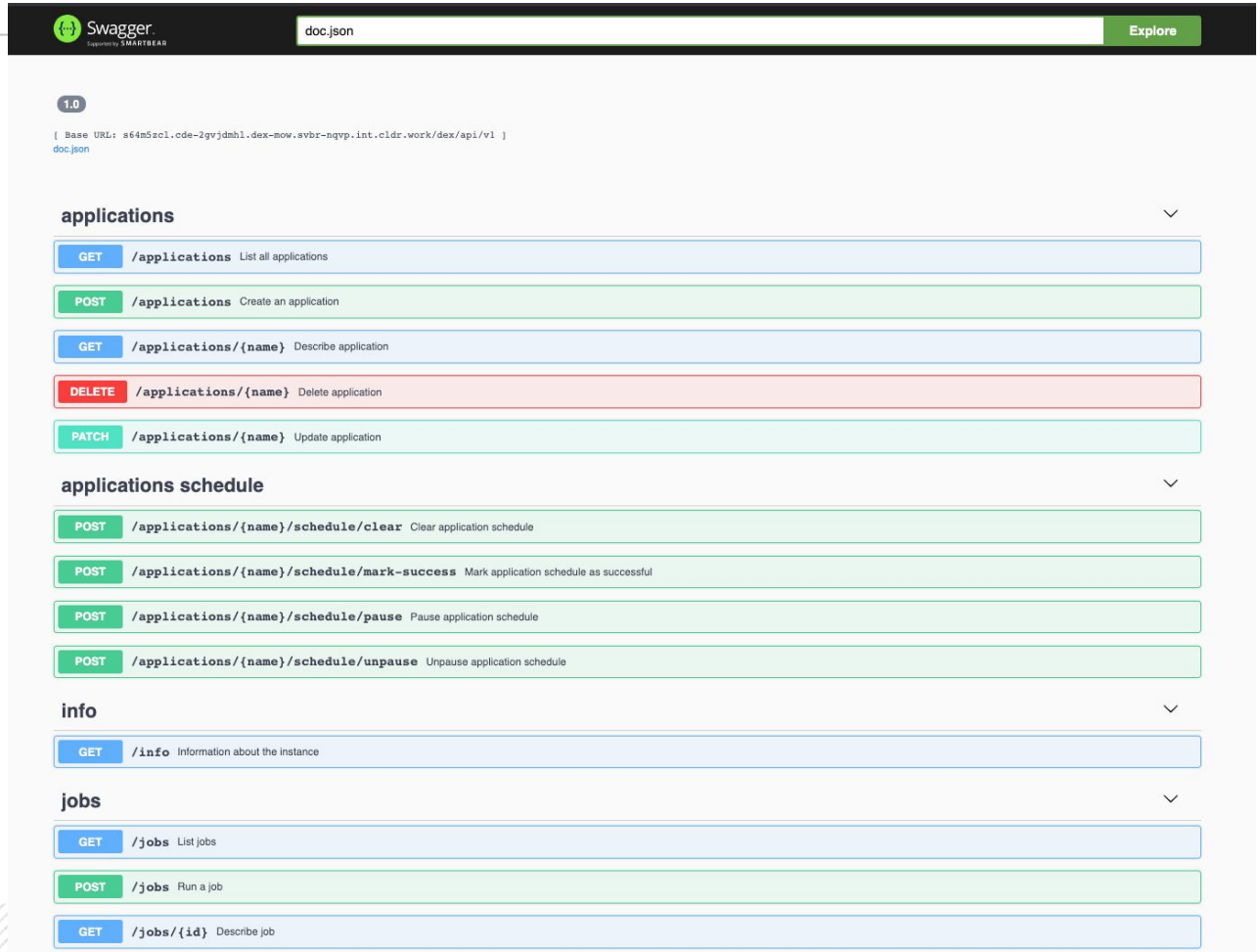
- Resolve issues fast with real-time visual performance profiling
- Complete monitoring & alerting capabilities

SIMPLIFIED JOB MANAGEMENT & APIS

- Full lifecycle mgmt.
- API-driven pipeline automation for any service
- Any language: SQL, Java, Scala, Python

RICH API

Used by >80% Customers



CDE DEMOS

We would be happy to collaborate with you

Spark 3 & Iceberg: a quick intro of Time Travel Capabilities with Spark 3 https://github.com/pdefusco/Spark3_Iceberg_CML

Getting Started with the CDE CLI https://github.com/pdefusco/CDE_CLI_demo

CDE CLI Basics https://github.com/pdefusco/CDE_CLI_Simple

GitLab2CDE: a CI/CD pipeline to orchestrate Cross Cluster Workflows - Hybrid/Multicloud Data Engineering
https://github.com/pdefusco/Gitlab2CDE

CML2CDE: a CI/CD Pipeline to deploy Spark ETL at Scale with Python and the CDE API https://github.com/pdefusco/CML2CDE

Postman2CDE: using the Postman API to bootstrap CDE Services https://github.com/pdefusco/Postman2CDE

Oozie2CDE: an API to easily migrate Oozie Workflows to Spark/Airflow CDE Jobs
https://github.com/pdefusco/Oozie2CDE_Migration

Spark & Airflow Data Engineering Workshop

Overview:

In this workshop we will implement an end-to-end data engineering workflow.

- ½ Hour: Slide Deck
- 1 ½ - 2 ½ Hours: Hands On Labs
- The workshop is free…

Stakeholders
Data Engineers
Operations Teams
IT Teams
CDP Admins (optional)
Data Scientists (optional)

Attendees gain hands on experience with Spark, Airflow and Iceberg in the Data Engineering Service

Why Cloudera Professional Services?

Partner with the Industry Leader in Universal Data Distribution

Expertise

Depth of experience unmatched in real-world DataFlow implementations

Industry-specific domain knowledge and resources

Hands-on technical depth for implementations

Speed time-to-value with architecture, design and security expertise

All of Cloudera

Large global consulting team -- not just the consultant on the ground

Internal knowledge-based articles and best practices

Product and engineering accessibility

Alignment with Cloudera support

Partnership

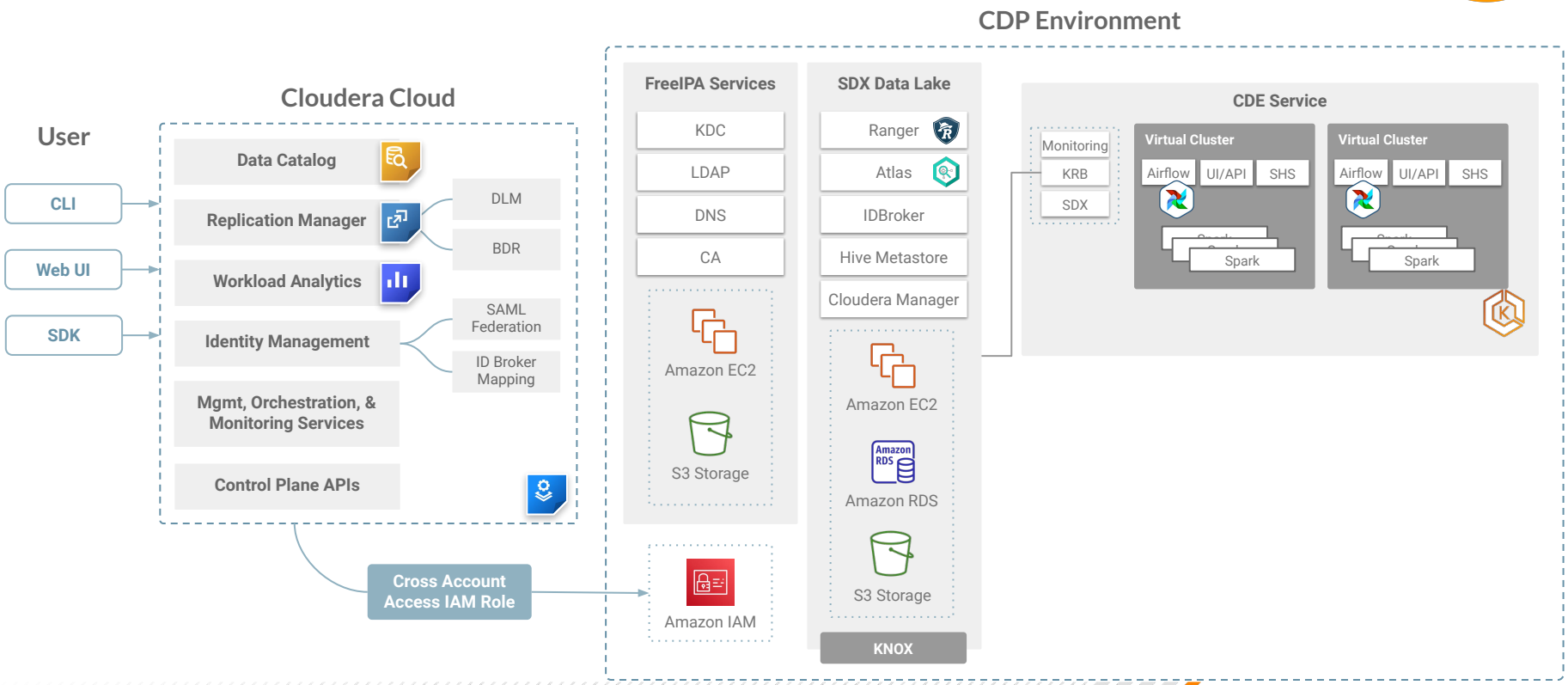
No “black box” solutions

Knowledge transfer to enable teams and drive project expansions

Partnership to solve business problems together

Trustworthy stakeholder advice backed by experience

CDP - AWS HIGH LEVEL ARCHITECTURE



Recommendations for Operationalizing CDE

Please reach out to the Account Team for more CDE Best Practices

CDE Virtual Clusters:

1. Keep Virtual Clusters on the same version. CDE allows for In-Place upgrades as of CDE 1.14
2. Use your favorite CI/CD tool to backup CDE Resources via CDE API/CLI
3. Do not apply VC autoscale ranges that too closely map to CDE Job driver/executor resources
4. Always create Spark CDE Jobs first and then Airflow CDE Jobs

CDE Integrations:

1. Always try Airflow rather than 3rd party tools to orchestrate CDE Jobs. CDE Airflow is already integrated with CDP SDX.
2. If you cannot use CDE Airflow but already have Open Source Airflow, you connect the two instances
3. If you must orchestrate each single CDE Job with a 3rd party use the CDE API

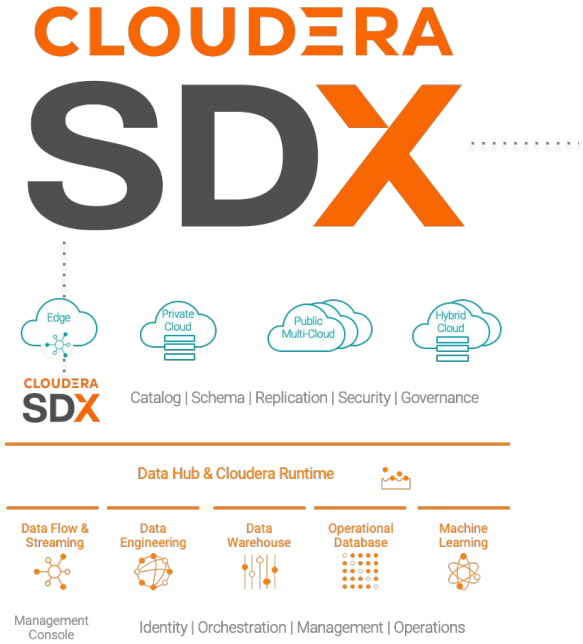
CDE Support Tickets:

1. Always upload CDE Virtual Cluster Diagnostic Bundle to the Ticket first
2. If k8s warnings or errors are shown, please visit the k8s dashboard for your cluster and identify any obvious errors
3. Do not use kubectl commands unless supervised by Cloudera Support technical team

Appendix

CONSISTENT SECURITY AND GOVERNANCE

Built for multi-functional analytics anywhere

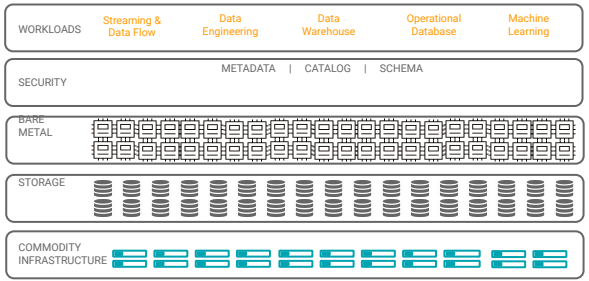


- **Data Catalog:** a comprehensive catalog of all data sets, spanning on-premises, cloud object stores, structured, unstructured, and semi-structured
- **Schema:** automatic capture and storage of any and all schema and metadata definitions as they are used and created by platform workloads
- **Replication:** deliver data as well as data policies there where the enterprise needs to work, with complete consistency and security
- **Security:** role-based access control applied consistently across the platform. Includes full stack encryption and key management
- **Governance:** enterprise-grade auditing, lineage, and governance capabilities applied across the platform with rich extensibility for partner integrations

EVOLUTION OF DATA ARCHITECTURE

Disaggregated Storage/Compute + SaaS Experiences

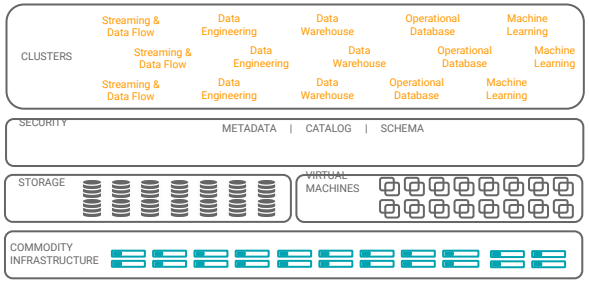
On Premise



Co-located Storage/Compute
Large, Shared Clusters
(CDH / HDP)

gen-1

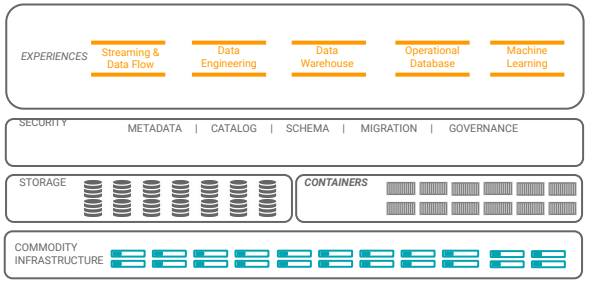
Public Cloud



Disaggregated Storage/Compute
Multiple Clusters
(Cloudbreak / Director)

gen-2

Multi-Public/Private/Hybrid Cloud



Disaggregated Storage/Compute
Multi-Tenant, Containerized "SaaS Experiences"
(CDP)

gen-3

Evolution

FULLY INTEGRATED WITH CDP

Data Engineering At The Edge Of Platform Innovation

SELF SERVICE, TURN-KEY SPARK

Streamlined Spark service with no manual setup or time-intensive configuration

OPEN & FLEXIBLE ORCHESTRATION

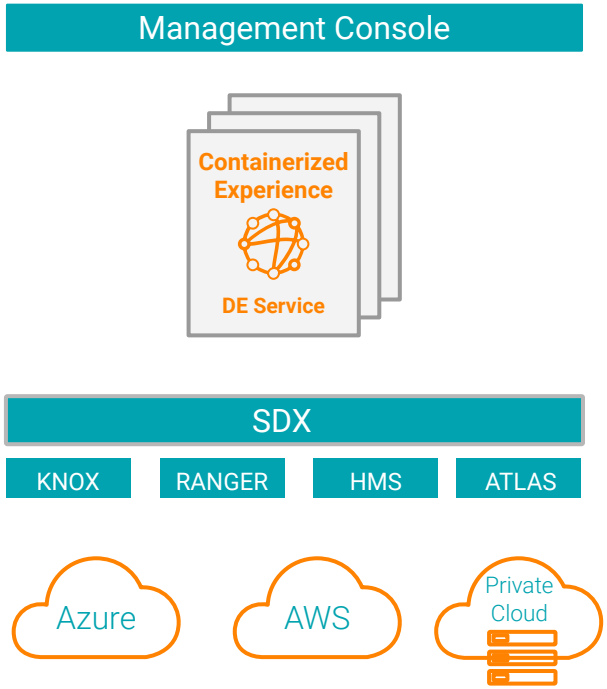
Airflow for data pipeline orchestration to integrated CDP experiences like CML, CDW, and COD

ELASTIC ISOLATED COMPUTE & STORAGE

Ephemeral and elastic compute for getting jobs done faster at lower cost

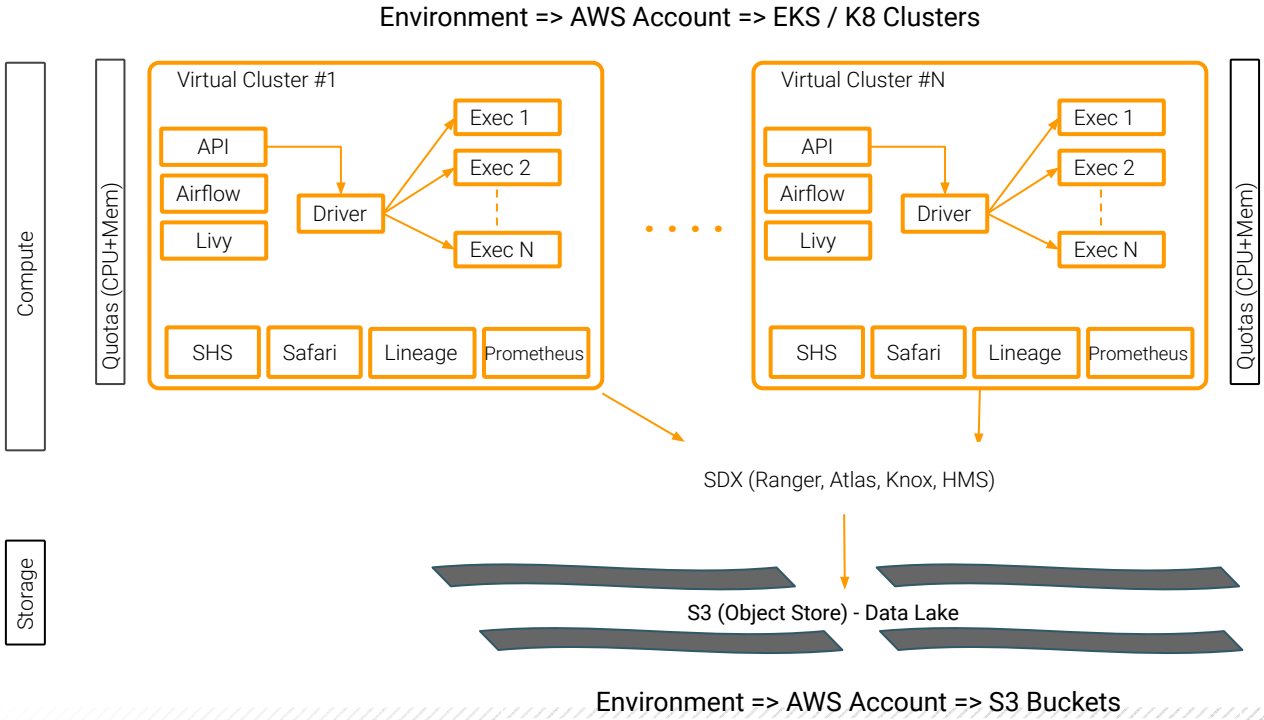
KUBERNETES & CONTAINERS

Isolated, service based architecture for flexibility and agility throughout

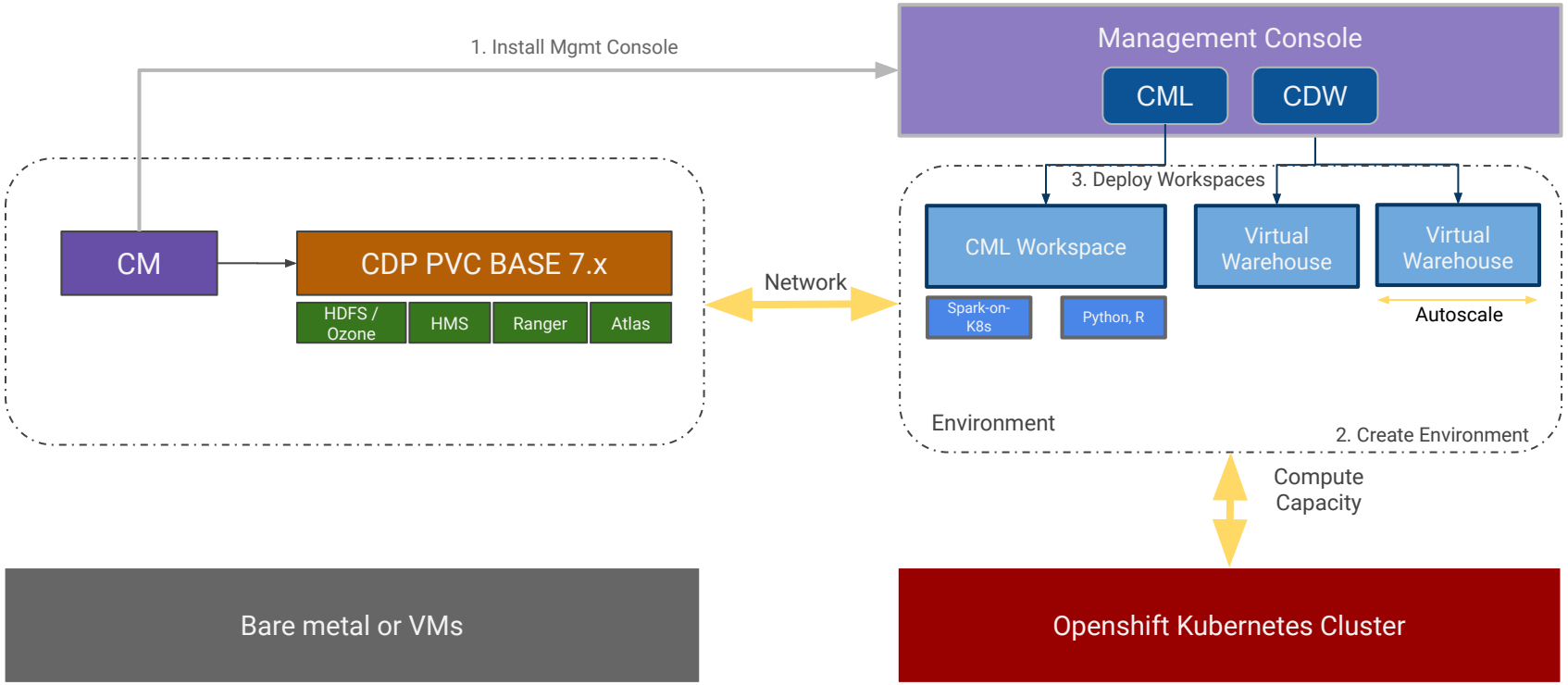


INSIDE THE CONTAINERIZED DE SERVICE

PROVISION MULTIPLE CLUSTERS WITH QUOTAS, ACCESS SHARED DATA IN THE LAKE



CDP Private Cloud – Dedicated Kubernetes



Model Lineage Tracking is done in Atlas