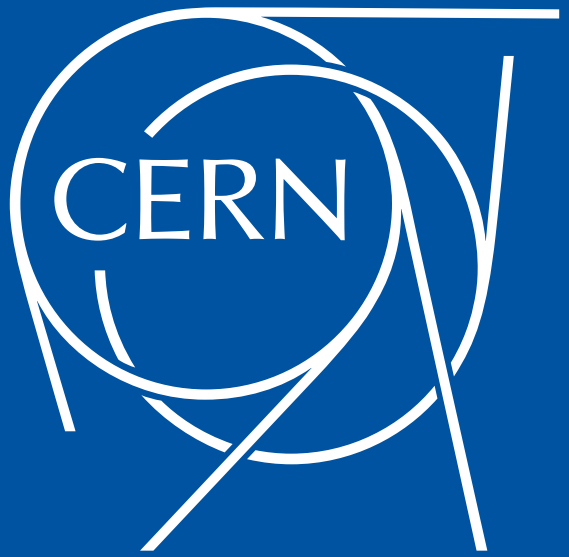

CERN

Boosting Complex IoT Analysis with Autonomous Data Warehouse Cloud

Customer Case Study Session

https://indico.cern.ch/event/767130/

Eric Grancher, Manuel Martin Marquez, Sébastien Masson

Outline

Introduction

ADWC Setup

Moving Data to ADWC

Performance Insights

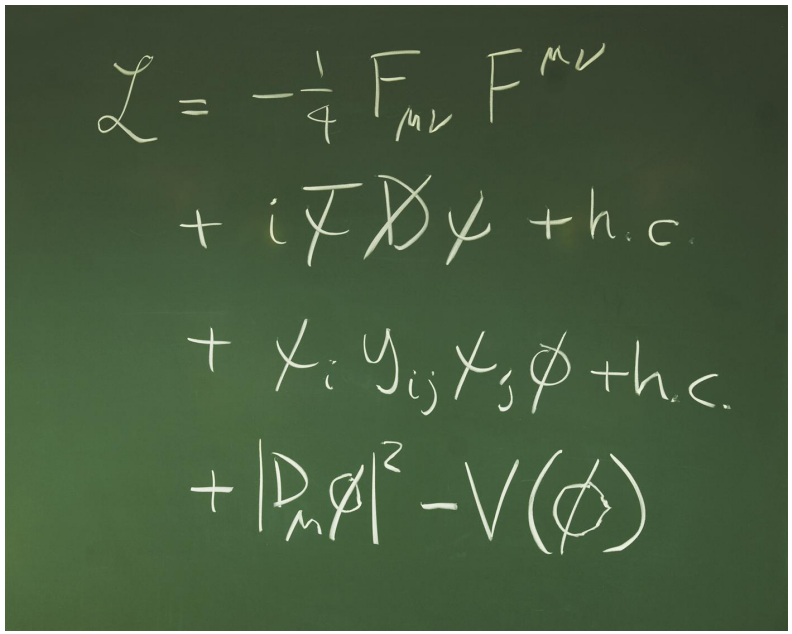
Conclusion

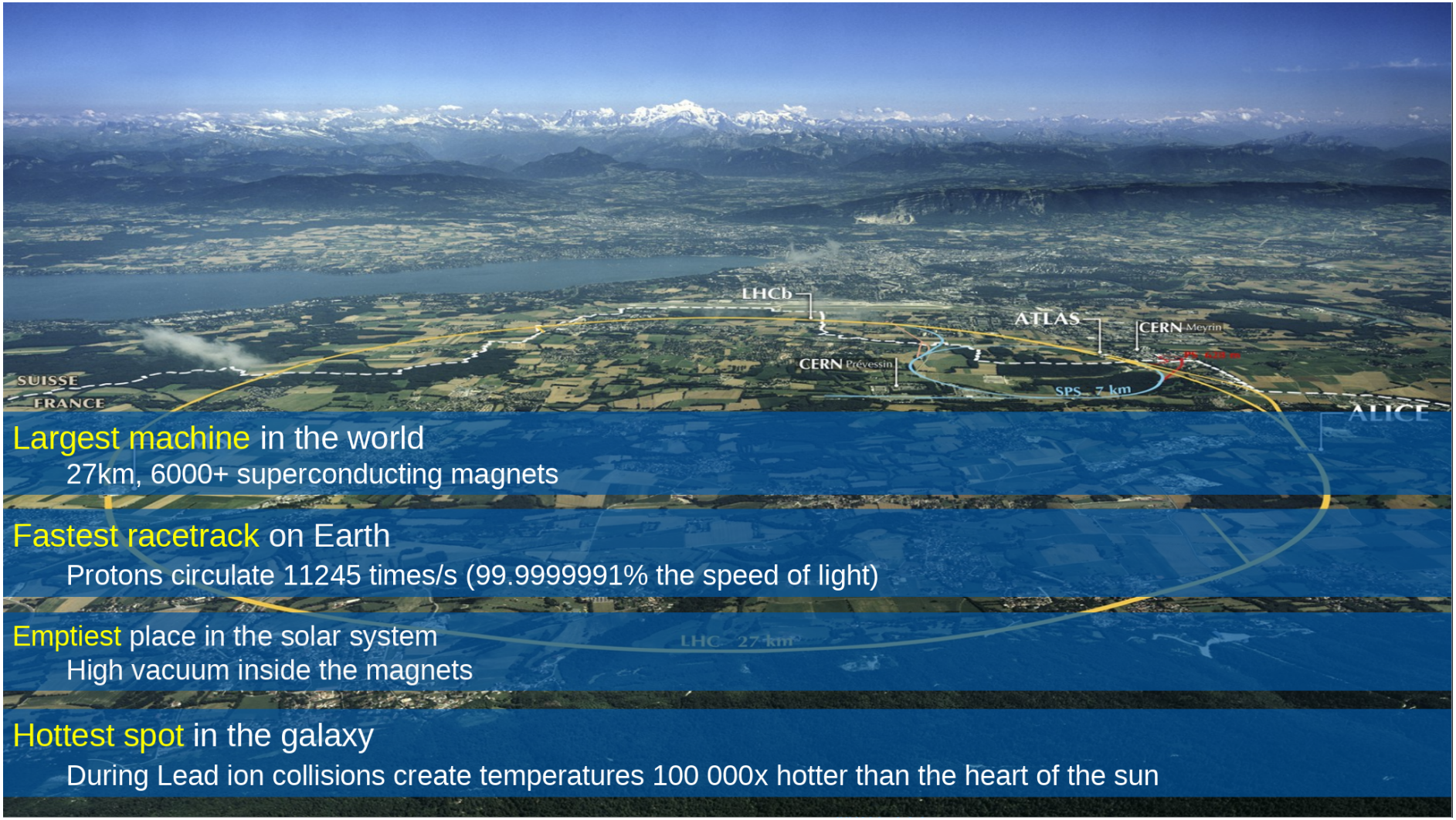
CERN

- CERN - European Council for Nuclear Research
- Founded in 1954 by 12 countries for fundamental physics research in the post-war Europe
- Today 22 members states (see `https://home.cern/about/member-states`) and world-wide collaborations
  - About 1100 MCHF/$ yearly budget
  - 2 300 CERN personnel
- More information at `https://home.cern/about`
- Can be visited, see `https://visit.cern/`

Fundamental Research

- What is 95% of the Universe made of?
- Why do particles have mass?
- Why is there no antimatter left in the Universe?
- What was the Universe like, just after "Big Bang"?





Largest machine in the world

27km, 6000+ superconducting magnets

Fastest racetrack on Earth

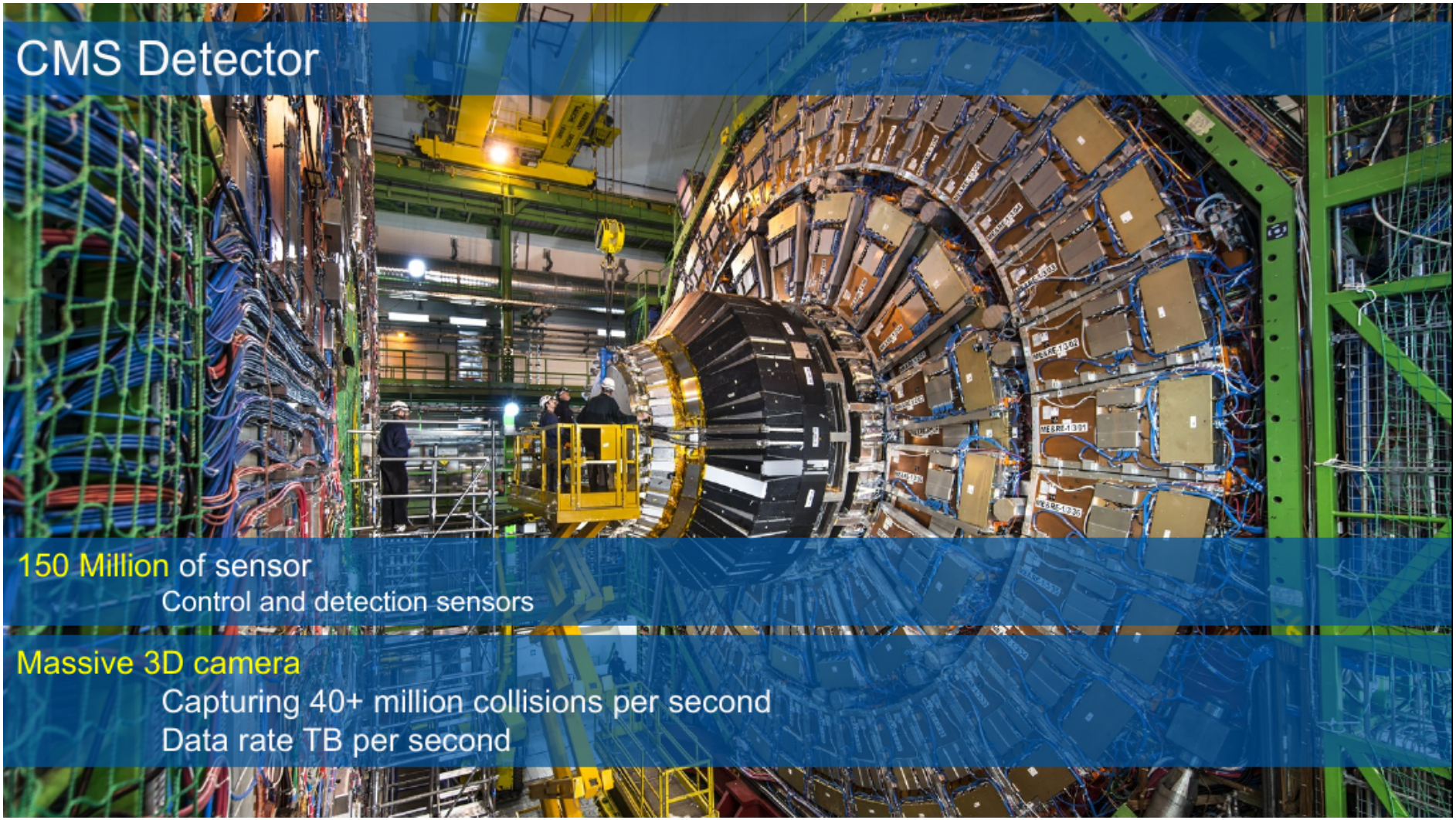
Protons circulate 11245 times/s (99.9999991% the speed of light)

Emptiest place in the solar system

High vacuum inside the magnets

Hottest spot in the galaxy

During Lead ion collisions create temperatures 100 000x hotter than the heart of the sun


CMS Detector
150 Million of sensor
Control and detection sensors
Massive 3D camera
Capturing 40+ million collisions per second
Data rate TB per second

CERN openlab

- Public-private partnership, through which CERN collaborates with leading ICT companies and other research organizations.
  - Evaluate state-of-the-art technologies in a challenging environment and improve them. Test in a research environment today's technologies that will be used in many business sectors tomorrow. Train the next generation of engineers/researchers. Promote education and cultural exchanges. Communicate results and reach new audiences. Collaborate and exchange ideas to create knowledge and innovation.
- Oracle is a member since 2003.

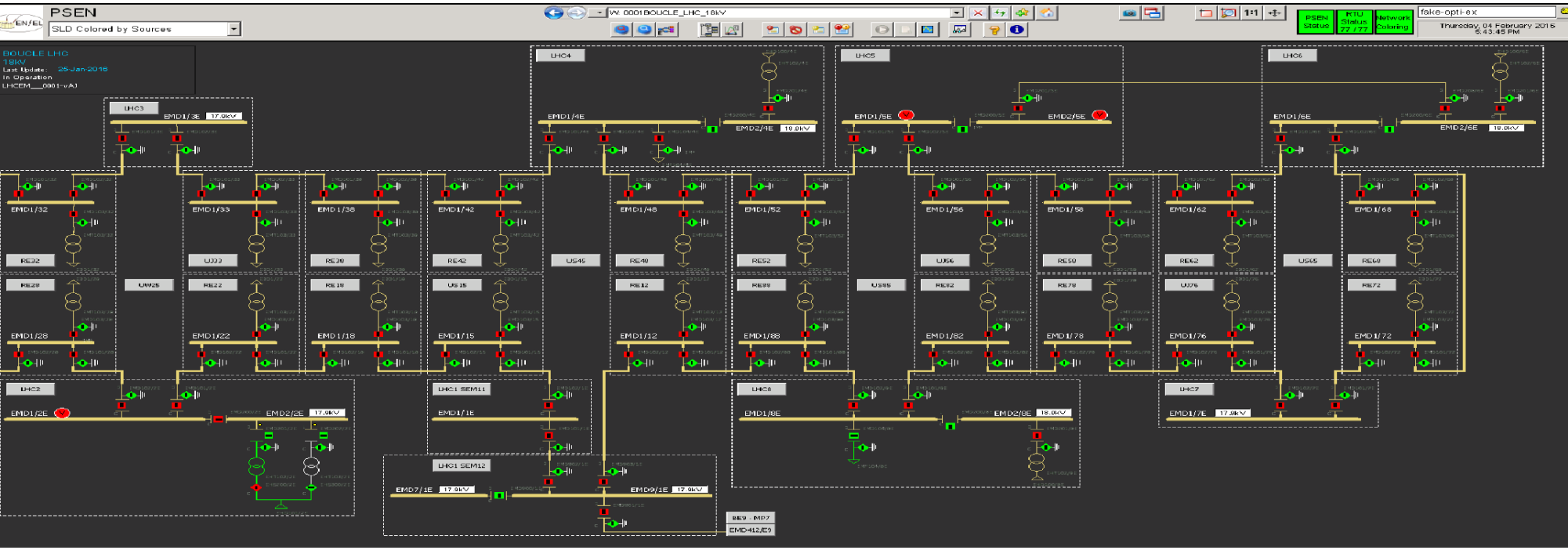
CERN's control system

- In addition of physics data, CERN's produces a lot of data for its SCADA (Supervisory Control And Data Aquisition) systems.
- SCADA scope is very wide:
  - **Accelerator systems**: cryogenics, vacuum, machine Protection, radiations...
  - **Detector Control System**: ATLAS, CMS, ALICE and LHCb
  - **Technical Infrastructure**: electrical network, cooling and ventilation systems

CERN Accelerator Logging System

- 2,057,960 defined signals produce more than 2.5TB data per day. These signals range from scalars, to arrays and 2D arrays of up-to 4 million elements. Data diverse in nature, with examples being accelerator running modes, equipment statuses, magnet currents, Cryogenics temperatures, particle beam positions, intensities, losses etc.
- System heaving used by more than 1000 individuals and 130 expert applications from all over CERN. The long term average number of requests to extract data for one or more signals exceeds 5 million requests per day. The CALS system is highly tuned in terms of making use of Oracle database features (range partitioned, compressed IOTs and optimised PL/SQL) and Oracle-specific JDBC configurations.
- Credit: Chris Roderick

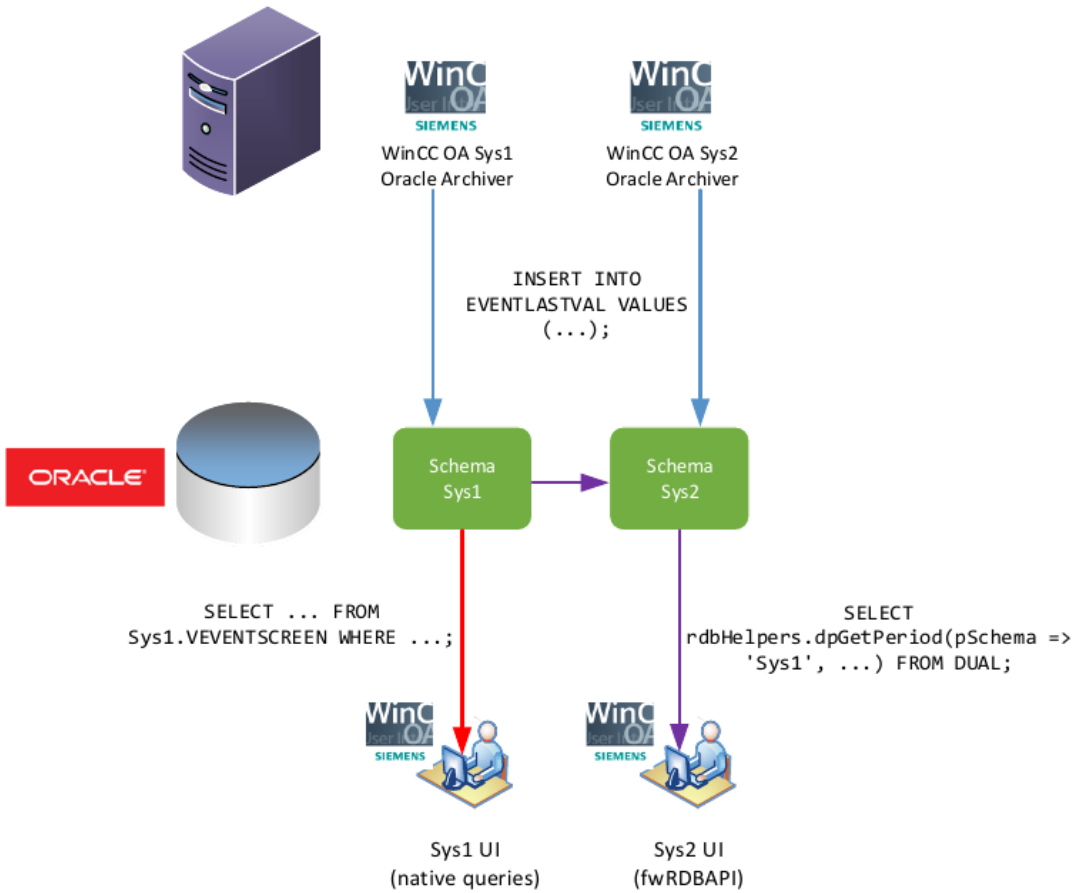
SCADA: Electrical Network



WinCC OA and CERN's extensions

- The software in use at CERN is Siemens WinCC OA
- Some extensions were needed to satisfy our requirements, mostly to be able to query against archived data:
  - New tables to store archived data
  - Joint work Oracle-Siemens-CERN to scale the archiver to 150k changes sustained
  - Triggers to populate new tables
  - Etc.

WinCC OA and CERN's extensions



Database environments on premise

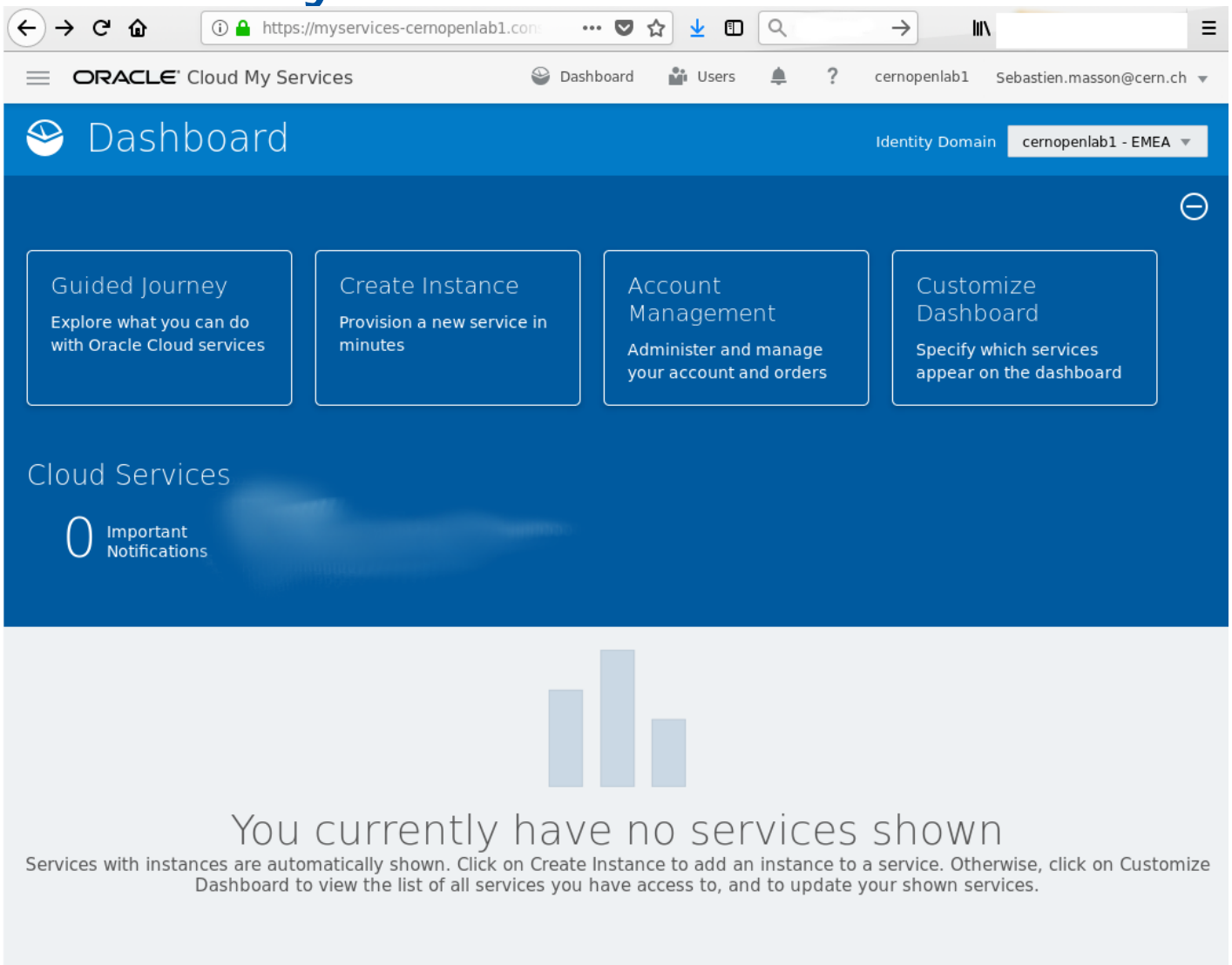
- Oracle 12.1.0.2 and 11.2.0.4 EE database, the database in question has a size of 20TB
- Several schemas containing (SCADA) Supervisory Control And Data Acquisition data for accelerators complex.
- Multiple connections to satisfy all monitoring purposes.

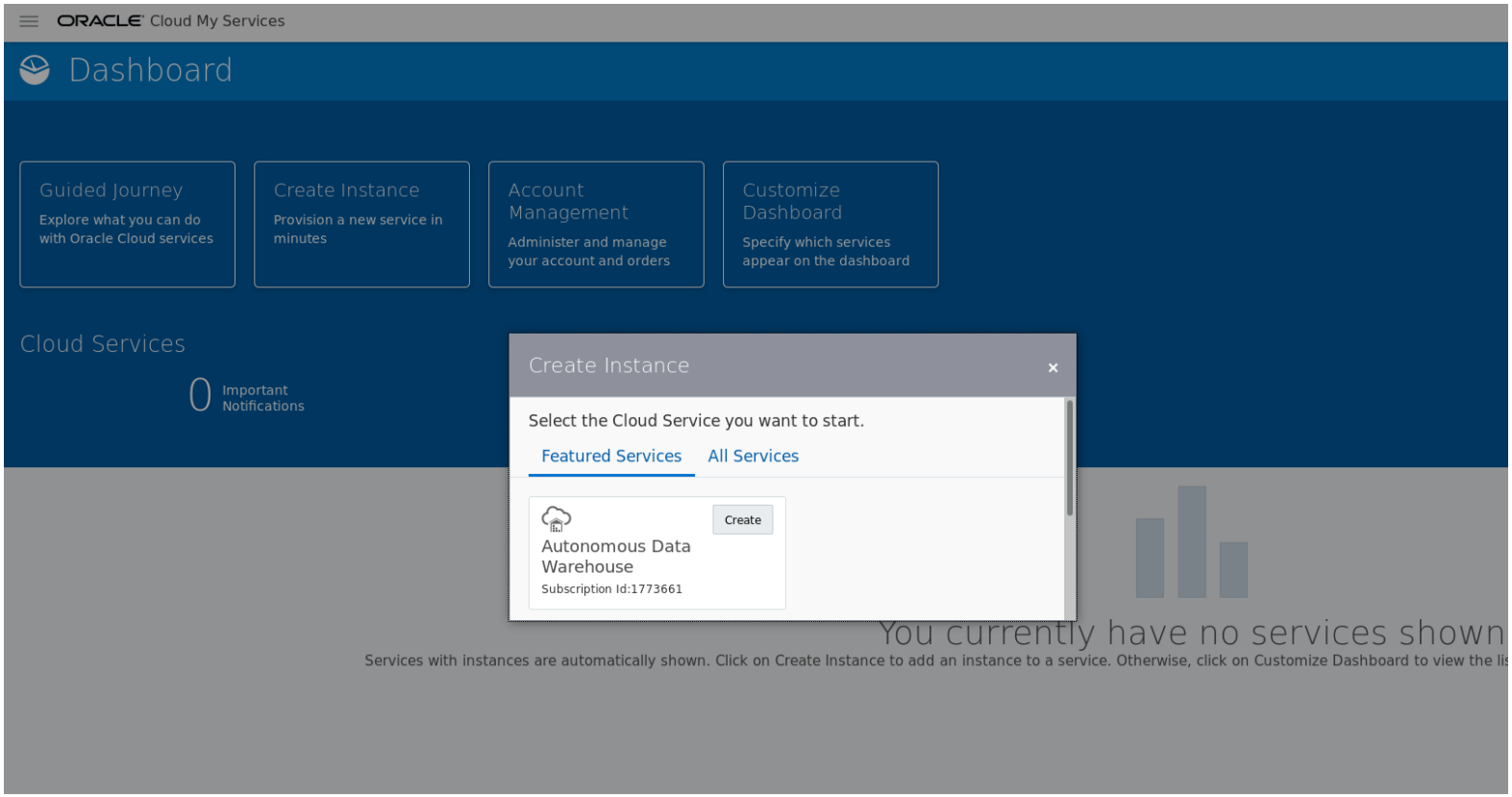
PSEN schema

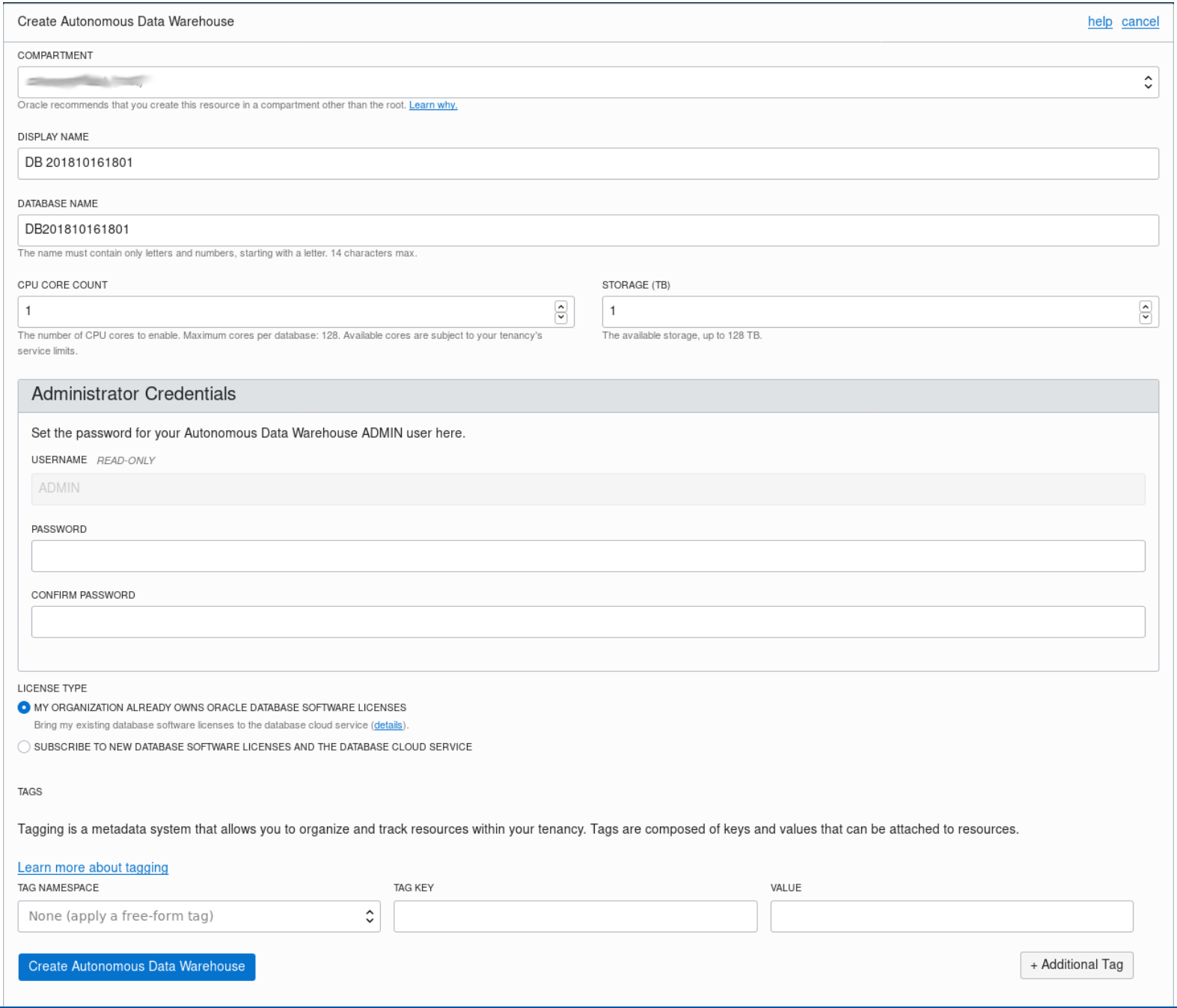
- 750GB of SCADA data about Electrical Network
- Contains IOT partitioned tables
- One big IOT partitioned table of 620G

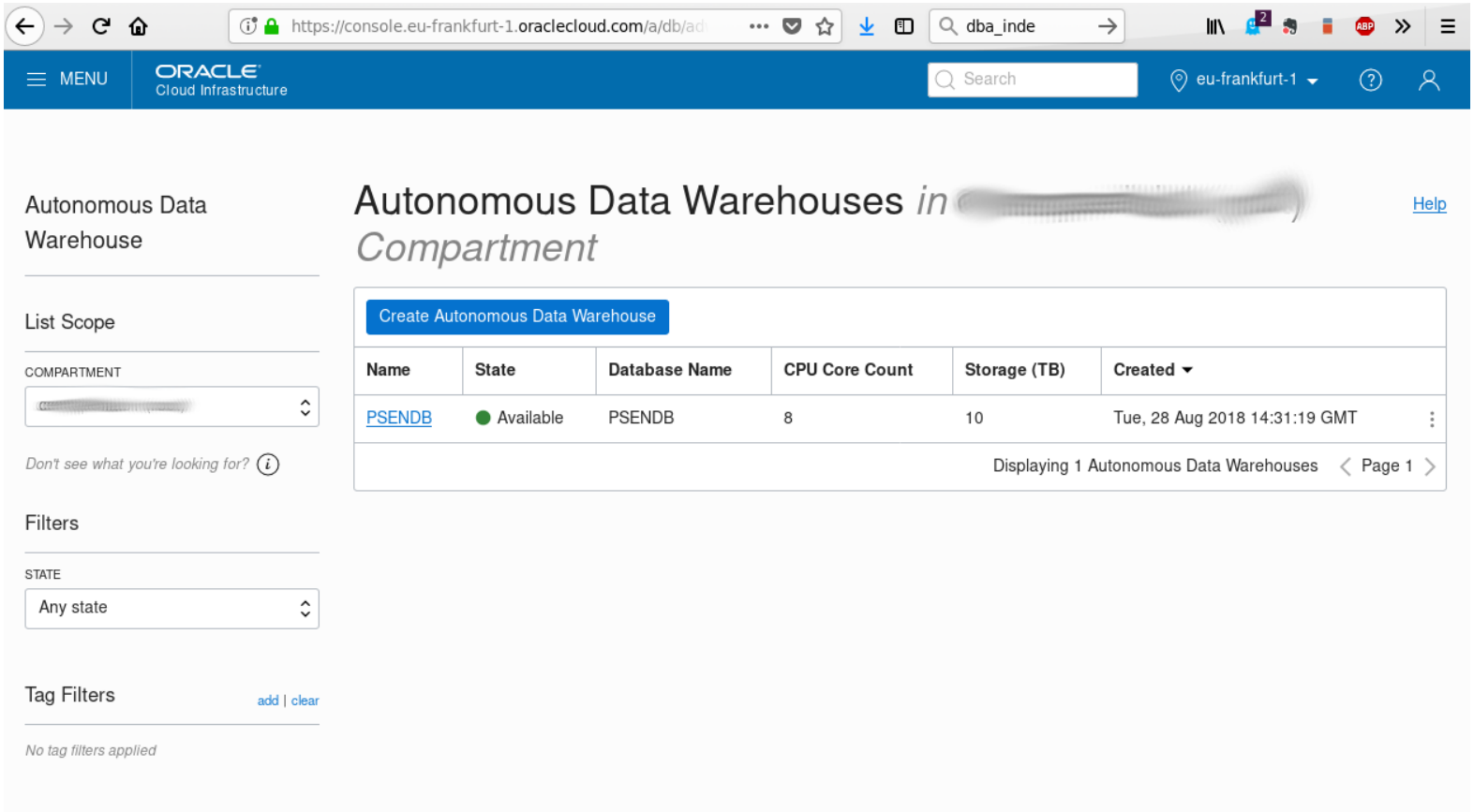
ADWC Setup

Creation of your ADWC instance: GUI




ORACLE Cloud My Services
Dashboard
Guided Journey
Explore what you can do with Oracle Cloud services
Create Instance
Provision a new service in minutes
Account Management
Administer and manage your account and orders
Customize Dashboard
Specify which services appear on the dashboard
Cloud Services
0 Important Notifications
Create Instance
Select the Cloud Service you want to start.
Featured Services
All Services
Create
Autonomous Data Warehouse
Subscription Id:1773661
You currently have no services shown
Services with instances are automatically shown. Click on Create Instance to add an instance to a service. Otherwise, click on Customize Dashboard to view the lis


Create Autonomous Data Warehouse
help cancel
COMPARTMENT
Oracle recommends that you create this resource in a compartment other than the root. Learn why.
DISPLAY NAME
DB 201810161801
DATABASE NAME
DB201810161801
The name must contain only letters and numbers, starting with a letter. 14 characters max.
CPU CORE COUNT
1
The number of CPU cores to enable. Maximum cores per database: 128. Available cores are subject to your tenancy's service limits.
STORAGE (TB)
1
The available storage, up to 128 TB.
Administrator Credentials
Set the password for your Autonomous Data Warehouse ADMIN user here.
USERNAME READ-ONLY
ADMIN
PASSWORD
CONFIRM PASSWORD
LICENSE TYPE
MY ORGANIZATION ALREADY OWNS ORACLE DATABASE SOFTWARE LICENSES
Bring my existing database software licenses to the database cloud service (details).
SUBSCRIBE TO NEW DATABASE SOFTWARE LICENSES AND THE DATABASE CLOUD SERVICE
TAGS
Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values that can be attached to resources.
Learn more about tagging
TAG NAMESPACE
None (apply a free-form tag)
TAG KEY
VALUE
Create Autonomous Data Warehouse
+ Additional Tag


https://console.eu-frankfurt-1.oraclecloud.com/a/db/ad
dba_inde
MENU
ORACLE
Cloud Infrastructure
Search
eu-frankfurt-1
Autonomous Data Warehouse
List Scope
COMPARTMENT
Don't see what you're looking for?
Filters
STATE
Any state
Tag Filters
add | clear
No tag filters applied
Autonomous Data Warehouses in
Compartment
Help
Create Autonomous Data Warehouse
Name
State
Database Name
CPU Core Count
Storage (TB)
Created
PSENDB
Available
PSENDB
8
10
Tue, 28 Aug 2018 14:31:19 GMT
Displaying 1 Autonomous Data Warehouses
Page 1

Creation of your ADWC instance: OCI

- You can also use Oracle Cloud Infrastructure CLI or OCI CLI command-line to perform this kind of operation.
- Tool to install locally to manage cloud infrastructure tasks by calling OCI REST APIs.

```
Commands:
  audit     Audit
  bv        Block Volume Service
  ce        Container Engine for Kubernetes
  compute   Compute Service
  db        Database Service
  dns       Public DNS Service
  email     Email Delivery Service
  fs        File Storage Service
  iam       Identity and Access Management Service
  kms       Key Management Service
  lb        Load Balancing Service
  network   Networking Service
  os        Object Storage Service
  search    Search Service
  setup     Setup commands for CLI
```

Creation of your ADWC instance: OCI

```
[oracle@itdbsma ~]$ oci db autonomous-data-warehouse
Usage: oci db autonomous-data-warehouse [OPTIONS] COMMAND [ARGS]...

  An Oracle Autonomous Data Warehouse.

  **Warning:** Oracle recommends that you avoid using any confidential
  information when you supply string values using the API.

Options:
  -?, -h, --help  Show this message and exit.

Commands:
  create   Creates a new Autonomous Data Warehouse.
  delete   Deletes the specified Autonomous Data...
  get      Gets the details of the specified Autonomous...
  list     Gets a list of Autonomous Data Warehouses.
  restore  Restores an Autonomous Data Warehouse based...
  start    Starts the specified autonomous Data...
  stop     Stops the specified Autonomous Data...
  update   Updates the specified Autonomous Data...
```

Creation of your ADWC instance: OCI

- Easy to install:

```
curl -L "https://raw.githubusercontent.com/oracle/oci-cli/master/scripts/install/install.sh" |bash
```

- And to configure:

```
oci setup config
```

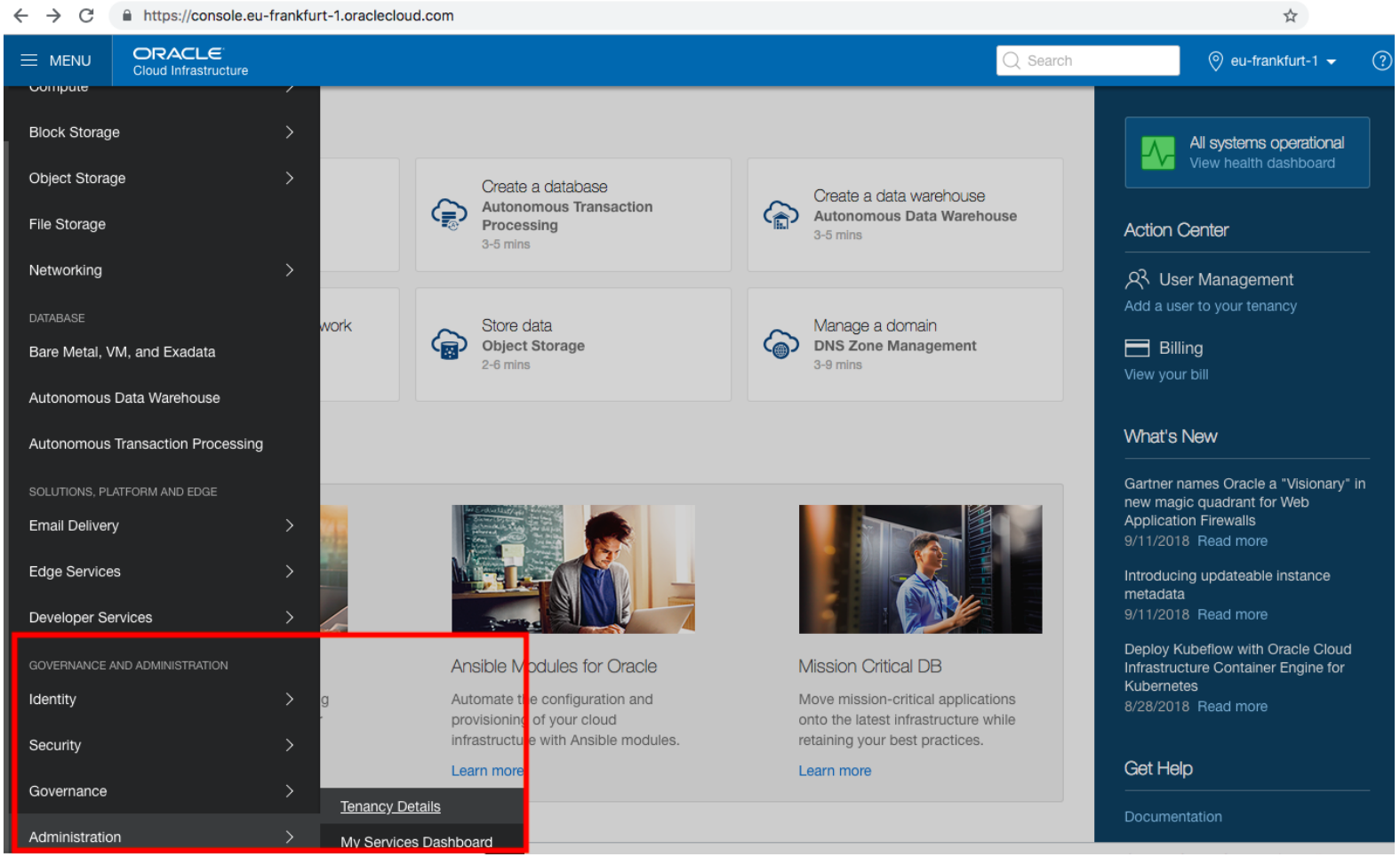
- Environment details OCIDs are needed:

Environment details and resource OCIDs

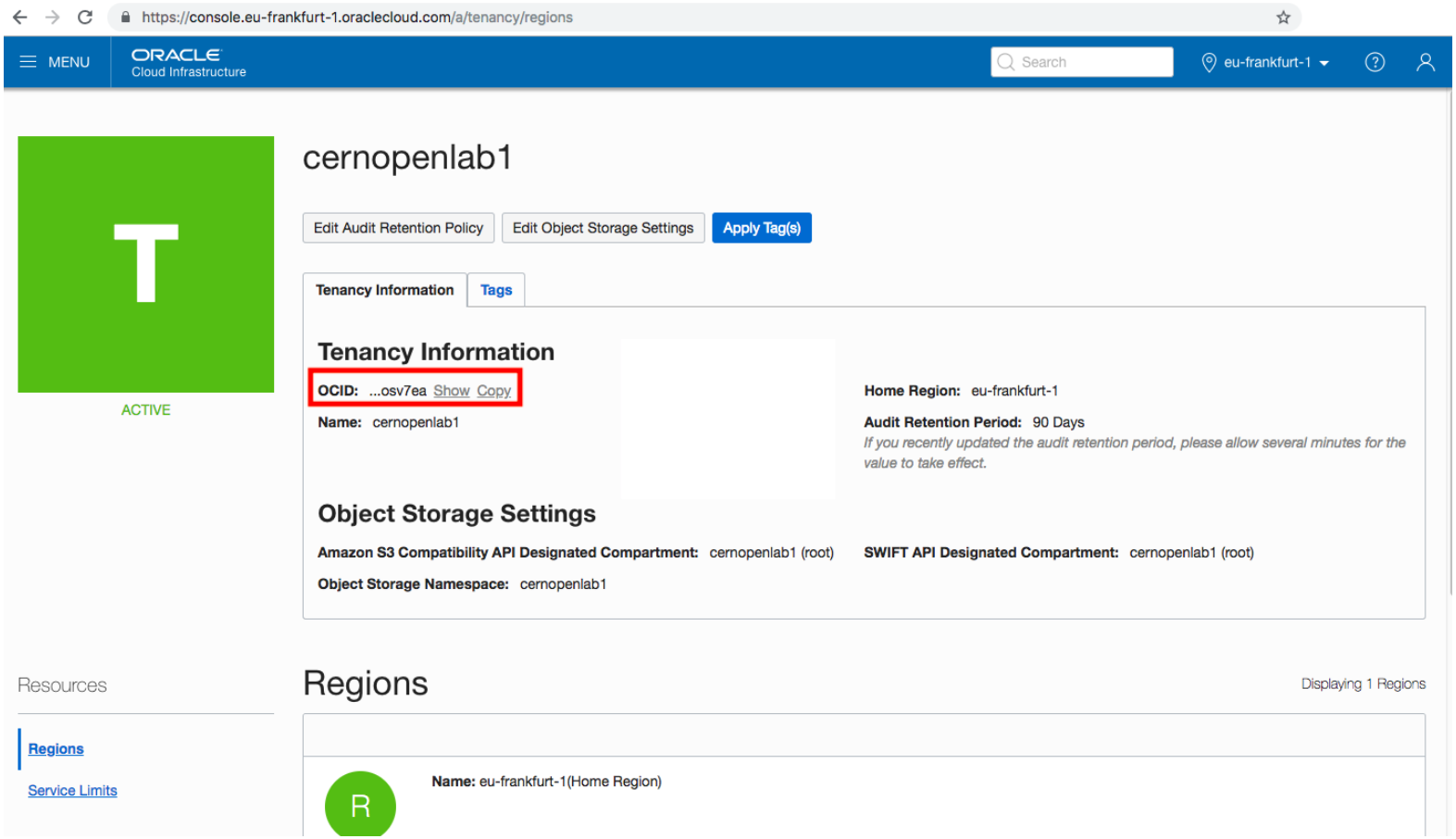
Every Oracle Cloud Infrastructure resource has an Oracle-assigned unique ID called an Oracle Cloud Identifier (OCID). It is included as part of the resource's information in both the Console and API.

- tenancy : ocid1.**tenancy**.oc1..aaaaaaaanssi********
- user : ocid1.**user**.oc1..aaaaaaaai3*********
- region : eu-frankfurt-1|us-ashburn-1|uk-london-1|us-phoenix-1

Environment details and resource OCIDs

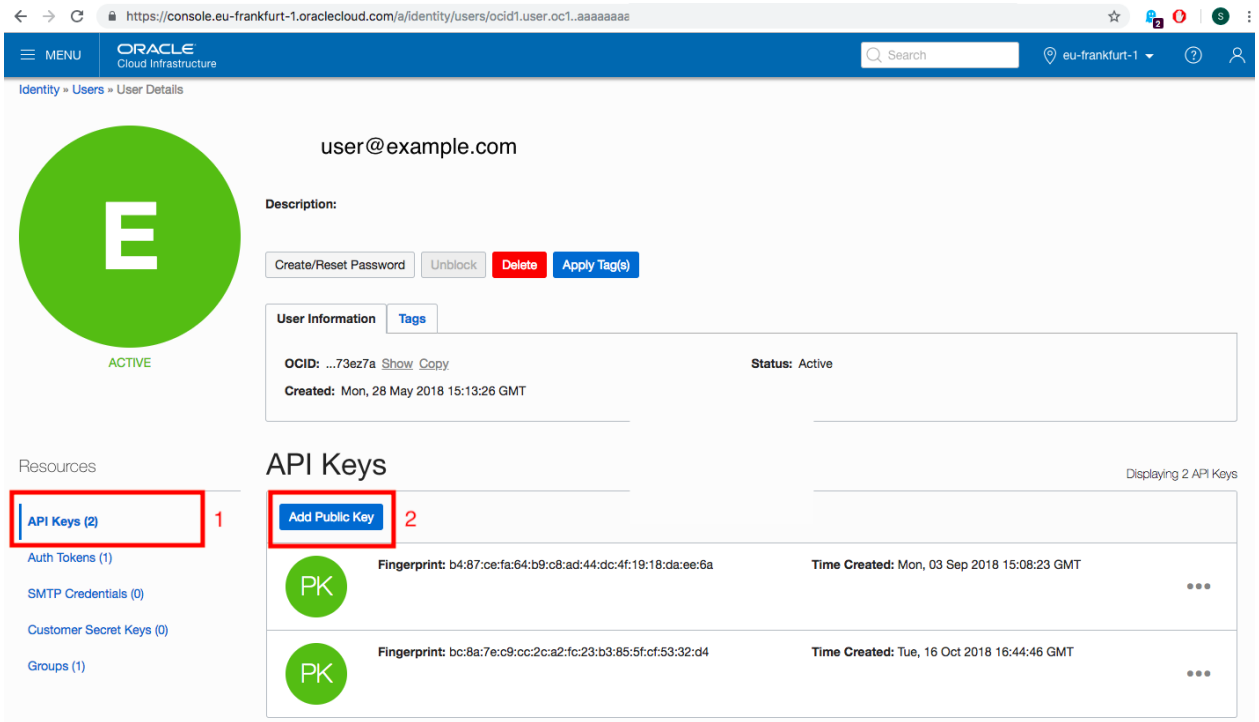


Environment details and resource OCIDs



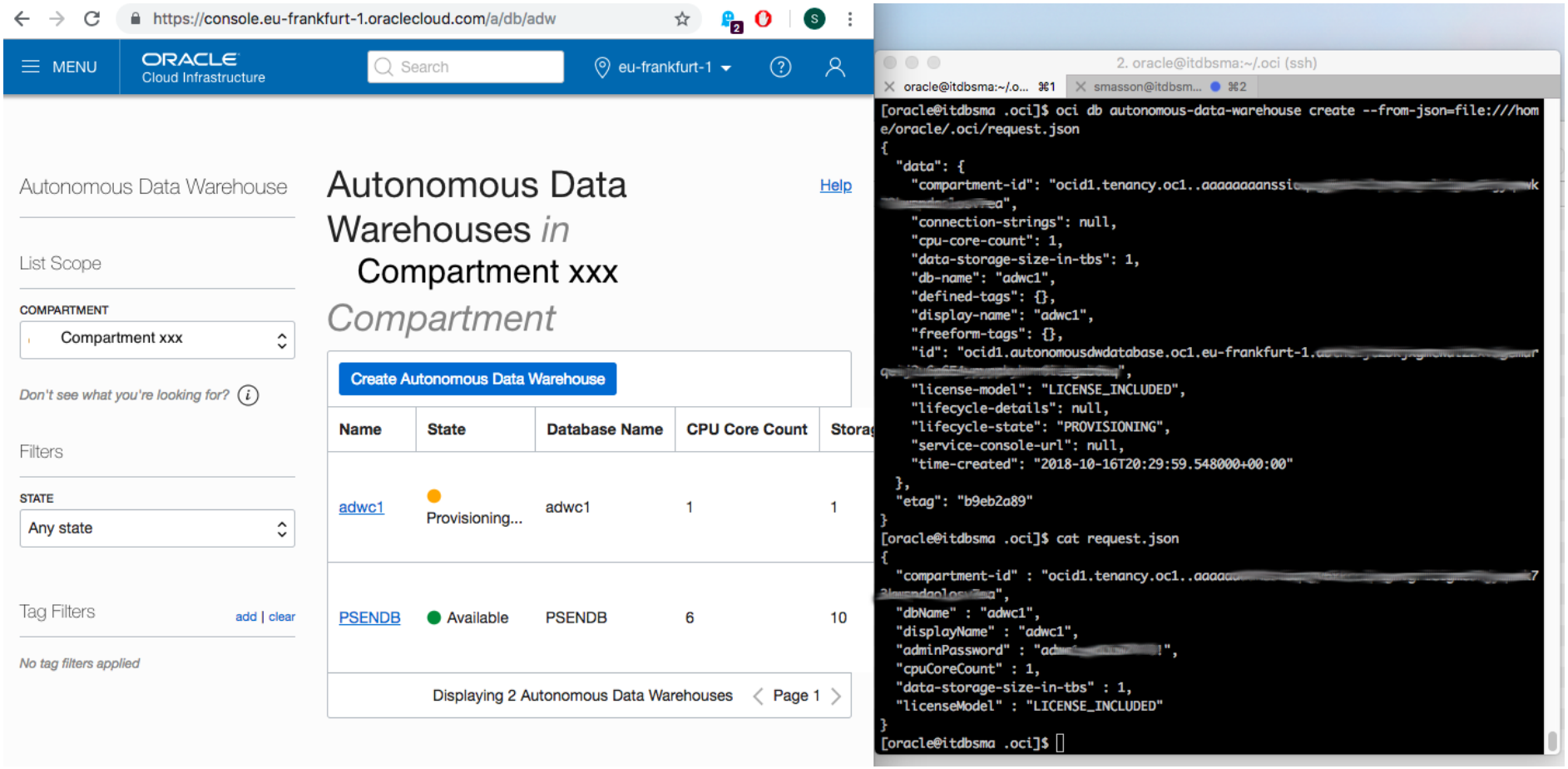
Creation of your ADWC instance: OCI

Once OCI is configured locally, you have to upload your public key in your user configuration:



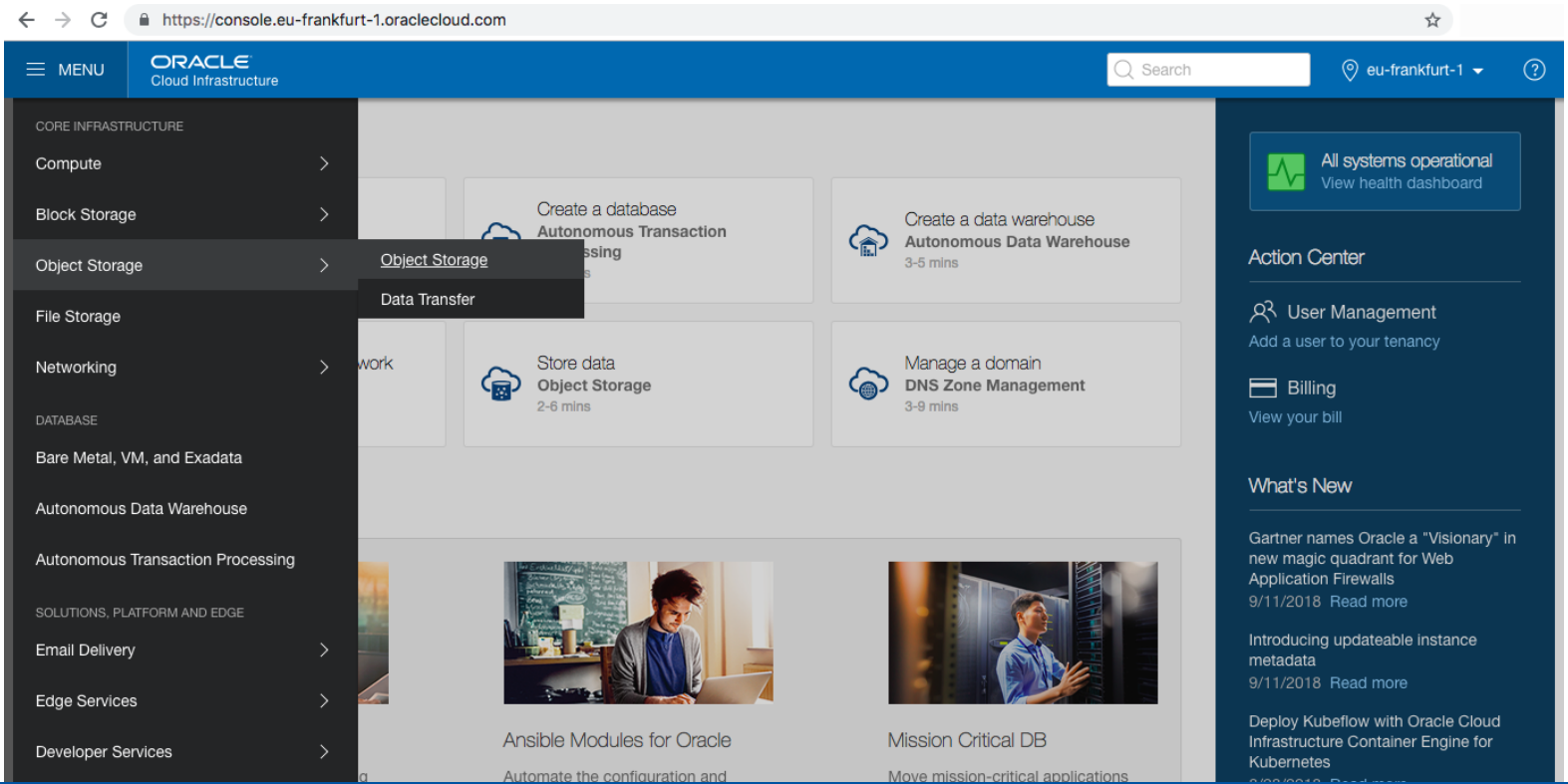
Then you can start using OCI

Creation of your ADWC instance: OCI

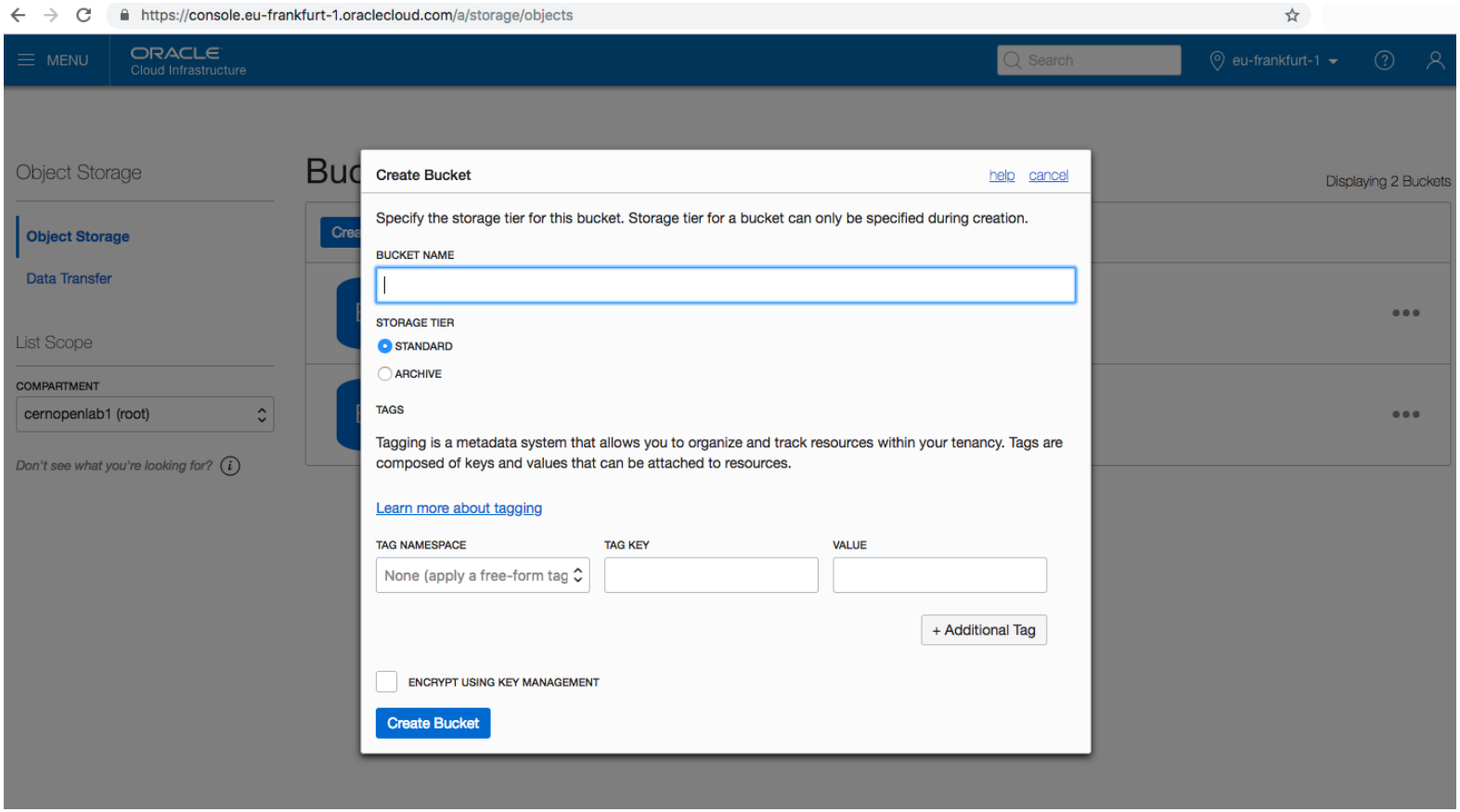


Creation of your Object Storage Bucket: GUI

Now, you need to configure your cloud object storage to upload your data pump export files. You have to create a bucket in one of your compartment.



Creation of your Object Storage Bucket: GUI



Creation of your Object Storage Bucket: OCI

```
[oracle@itdbsma .oci]$ oci os bucket create --name test_bucket
{
  "data": {
    "compartment-id": "ocid1.tenancy.oc1..aa[redacted]ea",
    "created-by": "ocid1.user.oc1..aaaaaaaai[redacted]a",
    "defined-tags": {},
    "etag": "bf1e[redacted]e6f018b",
    "freeform-tags": {},
    "kms-key-id": null,
    "metadata": {},
    "name": "test_bucket",
    "namespace": "cernopenlab1",
    "object-lifecycle-policy-etag": null,
    "public-access-type": "NoPublicAccess",
    "storage-tier": "Standard",
    "time-created": "2018-10-17T09:29:49.567000+00:00"
  },
  "etag": "bf1e0fef-ffe7-4d5e-84d2-39e39e6f018b"
}
```

OCI

OCI works with JMESPath (JSON Matching Expression paths) queries:

```
[oracle@itdbsma .oci]$ oci db autonomous-data-warehouse list \
> --output table \
> --query "data [*].{\
> DB:\"db-name\",\
> SIZE_TB:\"data-storage-size-in-tbs\",\
> CPUs:\"cpu-core-count\"}"
+------+--------+---------+
| CPUs | DB     | SIZE_TB |
+------+--------+---------+
| 1    | adwc1  | 1       |
| 6    | PSENDB | 10      |
+------+--------+---------+
```

```
[oracle@itdbsma ~]$ oci db autonomous-data-warehouse list \
> --query "data[?\"db-name\"=='adwc1'].\"connection-strings\"[].low"
[
  "adb.eu-frankfurt-1.oraclecloud.com:1522/                    oraclecloud.com"
]
```

Moving Data to ADWC

Moving Data to ADWC

- ADWC principle: data is imported in its simplest form:
  - No indices
  - No partitions
  - No IOTs
  - No materialized views
  - ...
- Data pump allows the needed transformations (contrary to RMAN)

Data pump export

- Oracle documentation recommends the following expdp parameters:

```
exclude=index, cluster, indextype, materialized_view, materialized_view_log,
materialized_zonemap, db_link
data_options=group_partition_table_data
parallel=n
schemas=schema name
dumpfile=export%u.dmp
```

Data pump export

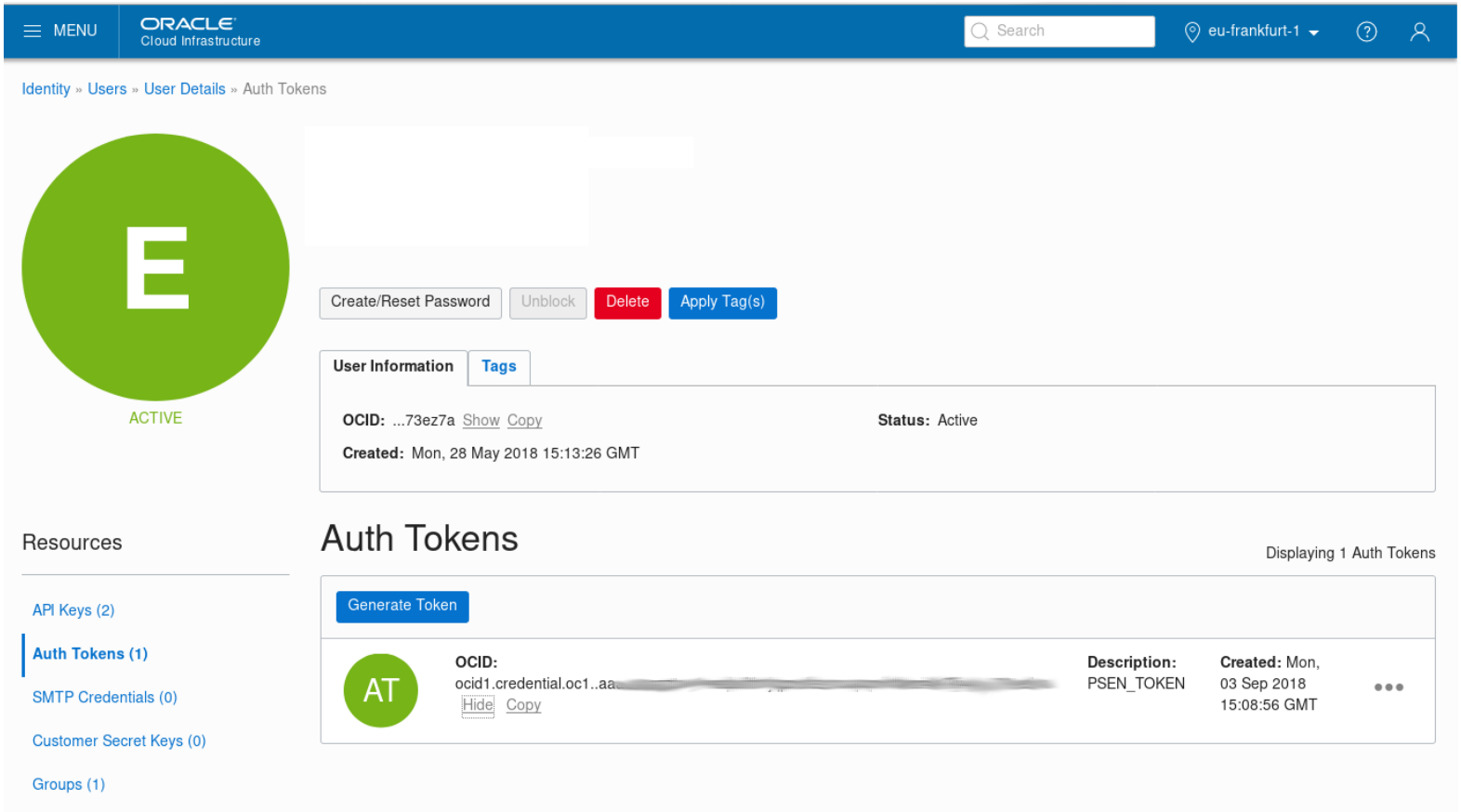
- data_option=group_partition_table_data :
  - Modifies storage parameters for partitions to allow faster import afterwards
  - **Requires 12.2 data pump client and database !**
- data_option=group_partition_table_data parameter not available in 11.2.0.4

Send data to object storage

- OCI is a very convenient way to send expdp files to object_storage
- OCI can split files and upload them in parallel. Files are automatically merged afterwards.

```
oci os object bulk-upload --bucket-name PSEN_BUCKET_1 --src-dir
/mnt/oci/ --part-size 64 --parallel-upload-count 10
```

Send data to object storage



Data pump import: parameters

- Oracle documentation recommends the followwing parameters:

```
directory=data_pump_dir
credential=def_cred_name
dumpfile= https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/adwc/adwc_user/sh.dmp
```

- DATA_PUMP_DIR directory is provided by default. Only related to log file location in this context.
- Every data pump export file needs to be mentioned
- Credentials to access object storage are needed

Data pump import: credentials parameter

- Credentials are required for data pump to access dump files
- Credentials are created with :

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'adwc_user@oracle.com',
    password => 'password'
  );
END;
/
```

- The password is called **auth token**, also referred as **swift password** before (Swift is the OpenStack object store service)

Data pump import: parameters

```
parallel=4
partition_options=merge
transform=segment_attributes:n
transform=dwcs_cvt_iots:y
transform=constraint_use_default_index:y
exclude=index, cluster, indextype, materialized_view, materialized_view_log, materialized_zonemap, db_link
```

- Parallel set to the number of CPUs you have
- Partitioned tables are converted to non-partitioned tables
- All segment attributes are ignored
- IOTs are converted to regular tables
- PK and unique indexes renamed to constraint name
- Same exclusion as during export

Data pump import

In our case, we could not use data_option=group_partition_table_data so we did not get any parallelism:

SID	EVENT	MODULE	SQL_TEXT
4301	wait for unread message on broa...	udi@dbnile-clie...	BEGIN :1 := sys.kupc$que_int.get_status(:2, :3); END;
6986	wait for unread message on broa...	Data Pump Master	BEGIN :1 := sys.kupc$que_int.receive(:2); END;
7344	enq: TM - contention	Data Pump Worker	INSERT /*+ APPEND ENABLE_PARALLEL_DML PARALLEL("EVENTHISTORY_00000008",1)+*/
7879	enq: TM - contention	Data Pump Worker	INSERT /*+ APPEND ENABLE_PARALLEL_DML PARALLEL("EVENTHISTORY_00000008",1)+*/
8419	Datapump dump file I/O	Data Pump Worker	INSERT /*+ APPEND ENABLE_PARALLEL_DML PARALLEL("EVENTHISTORY_00000008",1)+*/
8596	enq: TM - contention	Data Pump Worker	INSERT /*+ APPEND ENABLE_PARALLEL_DML PARALLEL("EVENTHISTORY_00000008",1)+*/

Data pump import

And even worse...

```
. . imported "PSEN"."EVENTHISTORY_00000008":"EVH_00000008_2017042800" 603.1 MB 9586464 rows
. . imported "PSEN"."EVENTHISTORY_00000008":"EVH_00000008_2015082700" 573.0 MB 9115136 rows
. . imported "PSEN"."EVENTHISTORY_00000008":"EVH_00000008_2015071100" 574.4 MB 9115083 rows
. . imported "PSEN"."EVENTHISTORY_00000008":"EVH_00000008_2016102300" 593.9 MB 9428783 rows
ORA-39014: One or more workers have prematurely exited.
ORA-39029: worker 10 with process name "DW09" prematurely terminated
ORA-31671: Worker process DW09 had an unhandled exception.
ORA-00600: internal error code, arguments: [4832], [0xABB5E23C8], [], [], [], [], [], [], [], [], [], []
ORA-06512: at "SYS.DBMS_STATS", line 1726
ORA-06512: at "SYS.DBMS_STATS", line 14728
ORA-06512: at "SYS.DBMS_STATS", line 31265
ORA-06512: at line 1
ORA-06512: at "SYS.DBMS_SQL", line 1721
ORA-06512: at "SYS.KUPD$DATA", line 1148
ORA-06512: at "SYS.KUPD$DATA", line 1252
ORA-06512: at "SYS.KUPD$DATA", line 3424
ORA-06512: at "SYS.KUPD$DATA", line 4719
ORA-06512: at "SYS.KUPD$DATA", line 6417
ORA-06512: at "SYS.KUPW$WORKER", line 22345
ORA-06512: at "SYS.KUPW$WORKER", line 5628
ORA-06512: at "SYS.KUPW$WORKER", line 13365
ORA-06512: at "SYS.KUPW$WORKER", line 2397
ORA-06512: at line 2
```

Data pump import

- Hopefully in the meantime, partitioned tables became available.
- New import without partition_options=merge parameter.
- We also imported indexes which became available too.
- Still under investigation with Oracle Support.

Data pump import

- During this second attempt we got:

```
. . imported "PSEN"."EVENTHISTORYVALUES_00000008":"EVHV_00000008_2015020100"  1.000 MB   11251 rows
KUP-11007: conversion error loading table "PSEN"."EVENTHISTORYVALUES_00000008"
ORA-12899: value too large for column VALUE_DYNSTRING (actual: 4019, maximum: 4000)

KUP-11009: data for row: VALUE_DYNSTRING : 0X'246473506C6F744461746 13A56616C7565206F766572207469'
```

- This is due to the migration from a single-byte character set to a multi-byte one.
- The fix was to recreate the table, change column definition from byte to char and import data again:

```
select column_name, char_used, data_length, data_type  from dba_tab_columns where
table_name='EVENTHISTORYVALUES_00000008' and column_name='VALUE_DYNSTRING';

COLUMN_NAME                    C    DATA_LENGTH DATA_TYPE
------------------------------ - -------------- --------------------
VALUE_DYNSTRING                B           4000 VARCHAR2

alter table psen.EVENTHISTORYVALUES_00000008 modify VALUE_DYNSTRING varchar2(4000 char);

select column_name, char_used, data_length, data_type  from dba_tab_columns where
table_name='EVENTHISTORYVALUES_00000008' and column_name='VALUE_DYNSTRING';

COLUMN_NAME                    C    DATA_LENGTH DATA_TYPE
------------------------------ - -------------- --------------------
VALUE_DYNSTRING                C           4000 VARCHAR2
```

Where is my alert.log?

Something that is a bit confusing when you start using ADWC.

```
col ORIGINATING_TIMESTAMP format a50
SQL> col MESSAGE_TEXT format a90
SQL> set line 400 pages 2000
SQL> select ORIGINATING_TIMESTAMP, message_text
  2    from V$DIAG_ALERT_EXT
  3  where ORIGINATING_TIMESTAMP between
  4  to_date('13/10/2018 01:00:00','DD/MM/YYYY HH24:MI:SS')
  5  and
  6  to_date('13/10/2018 10:00:00','DD/MM/YYYY HH24:MI:SS')
  7  order by ORIGINATING_TIMESTAMP;

ORIGINATING_TIMESTAMP                             MESSAGE_TEXT
------------------------------------------------- ------------------------------------------------------------------
--------------------------
13-OCT-18 01.11.47.784000000 AM +00:00            Setting Resource Manager plan DWCS_PLAN via parameter
13-OCT-18 01.11.51.906000000 AM +00:00            Resize operation completed for file# 4576, old size 1457520640K,
 new size 1468006400K
```

Where are my trace files?

```
select PAYLOAD from V$DIAG_TRACE_FILE_CONTENTS where
  2      TRACE_FILENAME='ehs1pod8_ora_115533.trc' order by line_number fetch
  3     first 15 rows only
  4  /

PAYLOAD
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------
Trace file /u02/app/oracle/diag/rdbms/ehs1pod/ehs1pod8/trace/ehs1pod8_ora_115533.trc
Oracle Database 18c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
Build label:    RDBMS_PT.DWCS_LINUX.X64_181006
ORACLE_HOME:    /u02/app/oracle/product/12.2.0.1/dbhome_1
System name:    Linux
Node name:      xxxxxxxx
Release:        4.1.12-94.7.8.el6uek.x86_64
Version:        #2 SMP Thu Jan 11 20:41:01 PST 2018
Machine:        x86_64
VM name:        Xen Version: 4.4 (HVM)
Storage:        Exadata
Instance name: ehs1pod8
Redo thread mounted by this instance: 8
Oracle process number: 960
```

And what about my data pump logs?

```
SQL> col object_name for a50
SQL> SELECT * FROM DBMS_CLOUD.LIST_FILES('DATA_PUMP_DIR');

OBJECT_NAME                                                BYTES
---------------------------------------------------------- ----------
dp.log                                                        129
export_PSEN.log                                               170
import.log                                                    336
import_PSEN_03082018.log                                    64547
import_PSEN_12092018.log                                    60235
import_PSEN_EVENTHISTORYVALUES_00000008.log                 15932
```

And what about my data pump logs?

```
SET SERVEROUTPUT ON SIZE 1000000
DECLARE
  l_file         UTL_FILE.file_type;
  l_location     VARCHAR2(100) := 'DATA_PUMP_DIR';
  l_filename     VARCHAR2(100) := 'import_PSEN_03082018.log';
  l_text         VARCHAR2(32767);
BEGIN
  -- Open file.
  l_file := UTL_FILE.fopen(l_location, l_filename, 'r', 32767);


  BEGIN
    LOOP
      UTL_FILE.get_line(l_file, l_text, 32767);
      DBMS_OUTPUT.put_line( l_text) ;
    END LOOP;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      NULL;
  END;

  -- Close the file.
  UTL_FILE.fclose(l_file);
END;
/
```

Performance Insights

Direct comparison between on premise database and ADWC is difficult...

- ADWC and on premise schemas are very different:
  - Transformations of partitioned, compressed IoTs to partitioned, compressed tables + PK index.
  - Execution plans changed a lot.
  - Our case is perhaps too specific to give a conclusion.
- Better to give some insights.

Statistics

Some statistics were missing after data pump import:

- So we needed to gather them on our schema
- A check to the documentation told us it was expected if you do not use recommended parameters (partition merge related parameters in our case).

Compression

Interesting case of EVENTHISTORY_00000008 table:

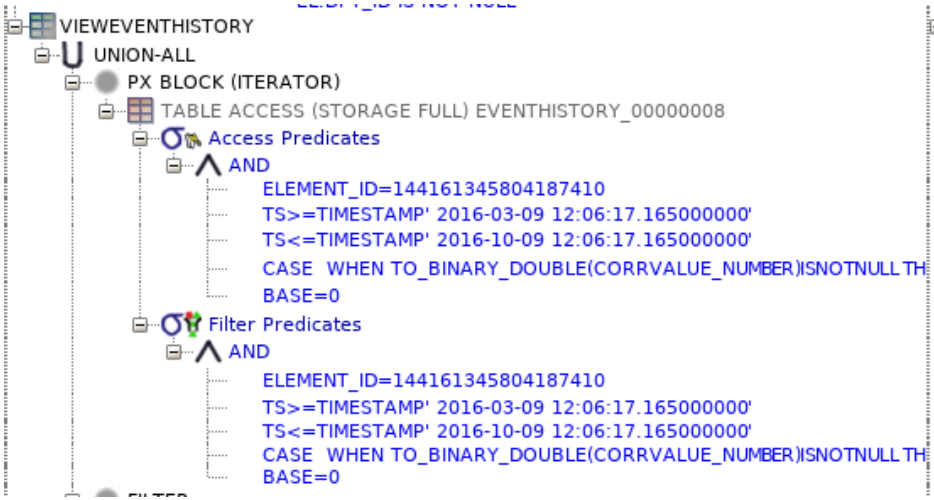
- This table is 620GB Index Organized Table (IOT) partitioned, compressed in our local database
- Transformed as non-IOT (but still partitioned and compressed) + Primary Key index on ADWC
  - Thanks to Hybrid Columnar Compression the table is now 70GB: full scans are smaller!
  - The ratio is the same for most of our tables.
- Is our physical model still a good one?

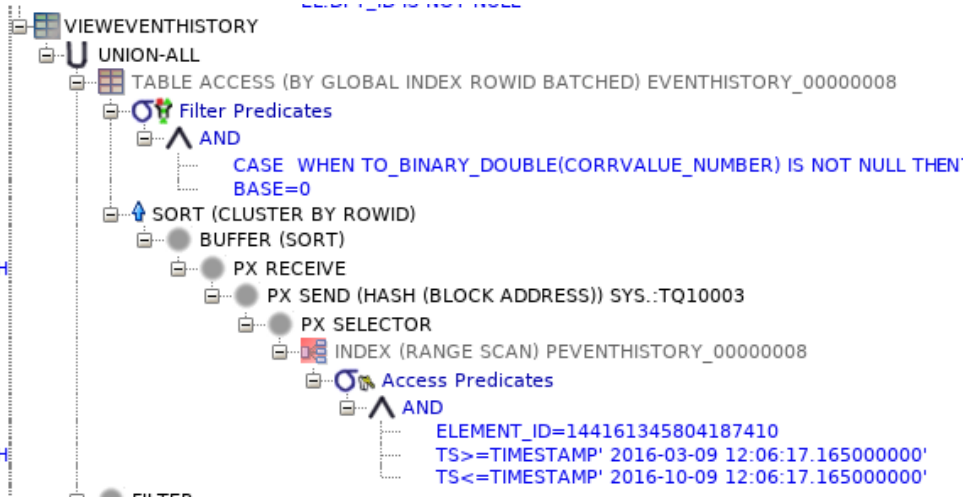
Indexes

Default scenario is to not use indexes on ADWC but it is useful in some cases:

```
select
    to_char(OFFVALUE_NUMBER),
    to_char(TS,'YYYY.MM.DD HH24:MI:SS.FF')STIME
  from
    VEVENTSCREEN
 where
    ELEMENT_ID = 144161345804187410
       and TS between TO_TIMESTAMP('2016.03.09 12:06:17.165000000','YYYY.MM.DD HH24:MI:SS.FF')
                  and TO_TIMESTAMP('2016.10.09 12:06:17.165000000','YYYY.MM.DD HH24:MI:SS.FF')
       and OFFVALUE_NUMBER is not null
 order by
    TS;
```

Indexes





In this case, only 3232 rows out of 10 billions are needed from EVENTHISTORY_00000008.

Indexes

```
Statistics
----------------------------------------------------------
         80  recursive calls
          0  db block gets
        934  consistent gets
          0  physical reads
        132  redo size
     212920  bytes sent via SQL*Net to client
       2937  bytes received via SQL*Net from client
        217  SQL*Net roundtrips to/from client
          9  sorts (memory)
          0  sorts (disk)
       3232  rows processed
```

```
Statistics
----------------------------------------------------------
         80  recursive calls
          0  db block gets
    2377599  consistent gets
    1124850  physical reads
   20969664  redo size
     212920  bytes sent via SQL*Net to client
       2937  bytes received via SQL*Net from client
        217  SQL*Net roundtrips to/from client
          8  sorts (memory)
          0  sorts (disk)
       3232  rows processed
```

Indexes

In this case:

- HCC helps a lot to reduce the Full Scan workload
- Indexes can still be more efficient when a high selectivity can be achieved

Indexes

"Test Drive Automatic Index Creation in Oracle Autonomous Database Cloud"
Today 04:45 PM - 05:30 PM — Moscone West - Room 3003
See you there!

Elastic scaling

- CPU and storage can be adjusted online, at anytime in few seconds!

```
oci db autonomous-data-warehouse update –cpu-core-count 8
```

- So you can start with few resources and grow only if needed

Conclusion

Some takeaways

- This presentation gives a number of tips to successfully create an ADWC IoT, especially for data loading and benefits of ADWC
- Work on significantly large "Industrial Internet of Things"
- Exadata nature of ADWC provides key features (HCC, Smart Scans, fast IO subsystem)
- Automation of ADWC helps create and manage the system including patching, online scale-up or scale-down, etc.
- Evolution of ADWC to be followed this week!

Thank you!

Questions, suggestions most welcome, now or via email.

manuel.martin.marquez@cern.ch,
sebastien.masson@cern.ch,
eric.grancher@cern.ch.

`https://indico.cern.ch/e/IOTADWC`


CERN