ORACLE
AI World

Operational Life Hacks

with Oracle AutoUpgrade

Mike Dietrich

Vice President

- in mikedietrich
- 🦋 @mikedietrichde.com
- B https://mikedietrichde.com

Alex Zaballa

Distinguished Product Manager

in alexzaballa

@alexzaballa.bsky.social

https://alexzaballa.com

Martin Berger

Database Platforms - Data Eng, Mgmt & Governance Manager

Accenture

- in martin-berger-ch
- @martinberger.bsky.social
- https://martinberger.com

What is a Life Hack?

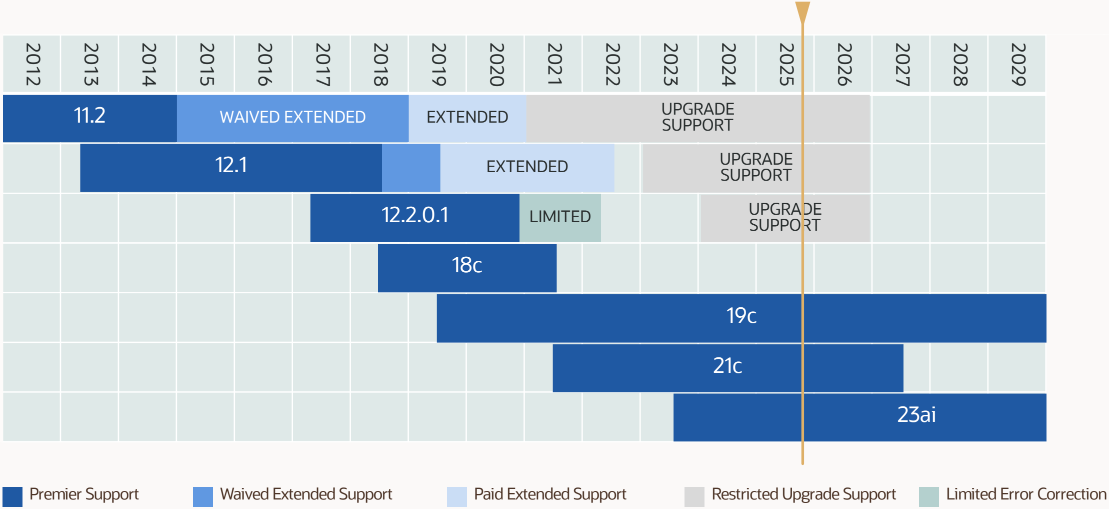
A life hack is any trick, shortcut, skill, or novelty method that increases productivity and efficiency.

By Jonathan Aquino from Victoria BC, Canada - IMG_0372, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=91896154

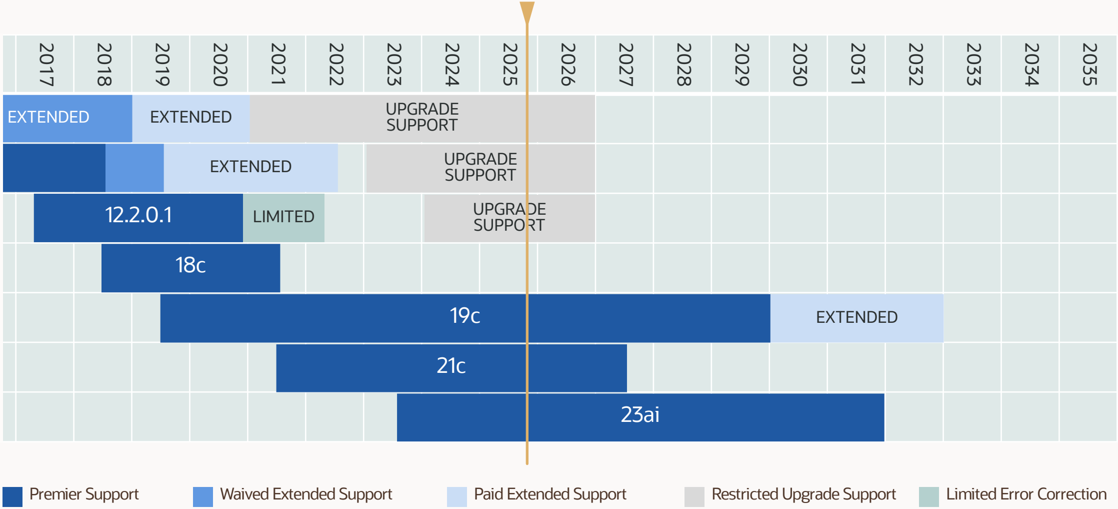
Introduction

Release Cycle and AutoUpgrade

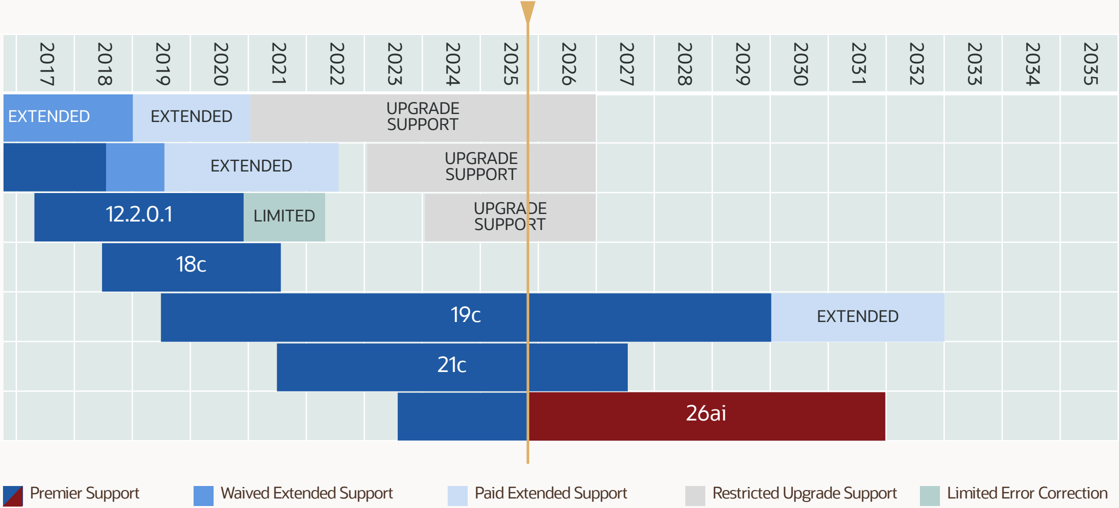
Lifetime Support Policy



Lifetime Support Policy



Lifetime Support Policy



26ai

When is a database upgrade required?

Oracle Database 19c → Oracle Database 23ai → Oracle AI Database 26ai

UPGRADE

Oracle Database 19c ⇛ Oracle AI Database 26ai

UPGRADE

Oracle Database 19c ⇒ Oracle Database 23ai ⇒ Oracle AI Database 26ai

UPDATE

your key to

Successful Database Upgrades

AutoUpgrade

Only supported method for upgrading to Oracle AI Database 26ai

AutoUpgrade

1 Upgrading

2 Non-CDB to PDB

3 Patching

AutoUpgrade

Upgrading

Non-CDB to PDB

Patching

Oracle
Database 19c

Oracle
AI Database 26ai

AutoUpgrade

Upgrading

Non-CDB to PDB

Patching

Non-CDB

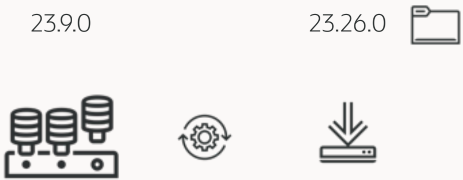
Multitenant

AutoUpgrade

Upgrading

Non-CDB to PDB

Patching



One single tool for everything
 - on all platforms

```
wget
https://download.oracle.com/otn-pub/otn_software/autoupgrade.jar
```

Operational Life Hack 1

Always collect all the logs

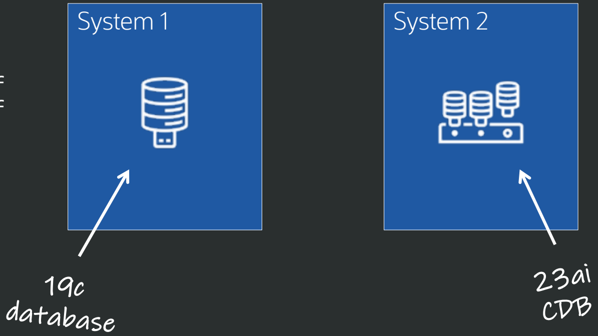
```
--Collect all logs files for all phases
--including the alert.log, broker logs and more

java -jar autoupgrade.jar -config config.cfg -zip
```

Operational Life Hack 2

Cloning makes testing so much easier

Upgrade via Refreshable Clone PDB

```
system01.dbf
sysaux01.dbf
users01.dbf
undo01.dbf
...
```



System 1

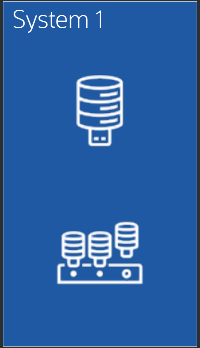
System 2

19c
database

23ai
CDB

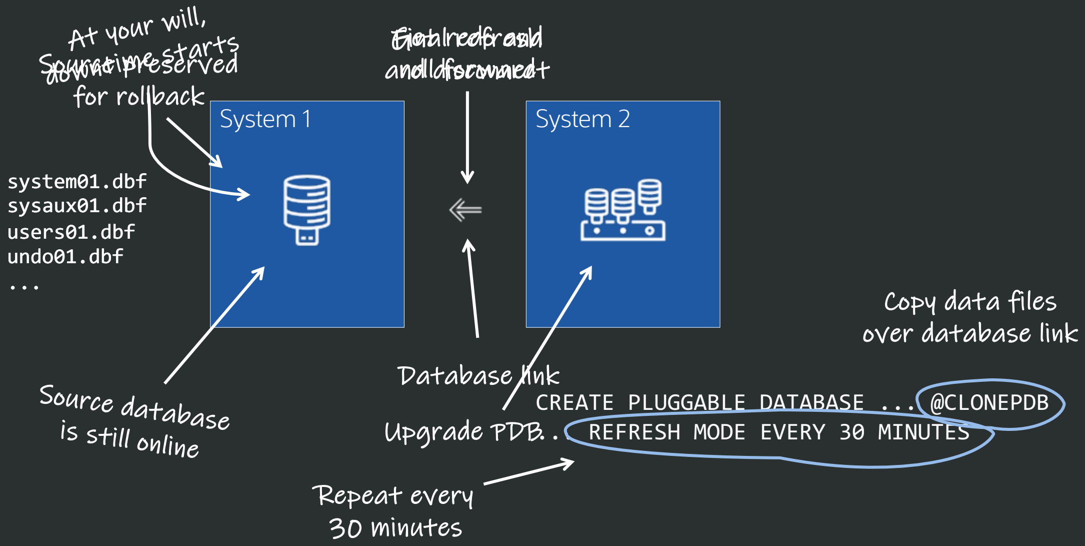
Upgrade via Refreshable Clone PDB

```
system01.dbf
sysaux01.dbf
users01.dbf
undo01.dbf
...
```



← Could be same system as well

Upgrade via Refreshable Clone PDB



At your will, downtime starts

Source preserved for rollback

Get redo and roll forward

Final refresh and disconnect

System 1

System 2

```
system01.dbf
sysaux01.dbf
users01.dbf
undo01.dbf
...
```

Copy data files over database link

Database link

```
CREATE PLUGGABLE DATABASE ... @CLONEPDB
REFRESH MODE EVERY 30 MINUTES
```

Upgrade PDB...

Source database is still online

Repeat every 30 minutes

Do this smarter with AutoUpgrade

Setup

Source non-CDB | **Target CDB**

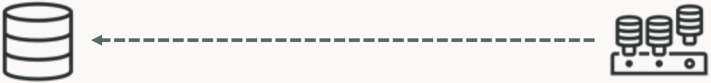
```
CREATE USER dblinkuser
       IDENTIFIED BY ... ;

GRANT CREATE SESSION,
      CREATE PLUGGABLE DATABASE,
      SELECT_CATALOG_ROLE TO dblinkuser;

GRANT READ ON sys.enc$ TO dblinkuser;
```

```
CREATE DATABASE LINK CLONEPDB
       CONNECT TO dblinkuser
       IDENTIFIED BY ...
       USING 'noncdb-alias';
```

Refreshable Clone PDB

Source non-CDB

Target CDB



```
autoupgrade.jar ... -mode analyze
```

```
autoupgrade.jar ... -mode fixups
```

```
autoupgrade.jar ... -mode deploy
```

Database Link

Source non-CDB

Target CDB

```
upg1.source_home=/u01/app/oracle/product/19
upg1.target_home=/u01/app/oracle/product/26
upg1.sid=NONCDB1
upg1.target_cdb=CDB1
upg1.target_version=23.26
upg1.source_dblink.NONCDB1=CLONEPDB
```

Source non-CDB

Target CDB

```
upg1.source_home=/u01/app/oracle/product/19
upg1.target_home=/u01/app/oracle/product/26
upg1.sid=NONCDB1
upg1.target_cdb=CDB1
upg1.target_version=23.26
upg1.source_dblink.NONCDB1=CLONEPDB 300
```

Rename your PDB to avoid name collision

- If CDB is on same host,
  it also registers for the default service

Refreshable Clone PDB

Source non-CDB | Target CDB

```
upg1.source_home=/u01/app/oracle/product/19
upg1.target_home=/u01/app/oracle/product/26
upg1.sid=NONCDB1
upg1.target_cdb=CDB1
upg1.target_version=23.26
upg1.source_dblink.NONCDB1=CLONEPDB 300
upg1.target_pdb_name.NONCDB1=PDB1
```

You can drop user and database link after migration

Refreshable Clone PDB

Source non-CDB	Target CDB

```
upg1.source_home=/u01/app/oracle/product/19
upg1.target_home=/u01/app/oracle/product/26
upg1.sid=NONCDB1
upg1.target_cdb=CDB1
upg1.target_version=23.26
upg1.source_dblink.NONCDB1=CLONEPDB 300
upg1.target_pdb_name.NONCDB1=PDB1
upg1.drop_dblink=yes
```

Limit the network load

Refreshable Clone PDB

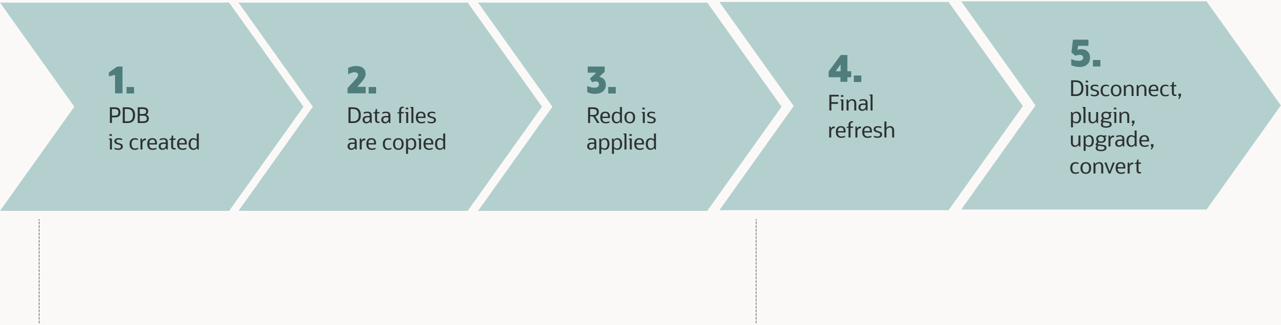
Source non-CDB

Target CDB

```
upg1.source_home=/u01/app/oracle/product/19
upg1.target_home=/u01/app/oracle/product/26
upg1.sid=NONCDB1
upg1.target_cdb=CDB1
upg1.target_version=23.26
upg1.source_dblink.NONCDB1=CLONEPDB 300
upg1.target_pdb_name.NONCDB1=PDB1
upg1.parallel_pdb_creation_clause=4
```

Adjust the start time dynamically

Refreshable Clone PDB



```
autoupgrade.jar ... -mode deploy
```

```
upg1.start_time=21/10/2025 22:30:00
```

```
--When a job is in REFRESHPDB stage, you can force it to start immediately
--Check documentation for other options

upg> proceed -job 101
```

```
--When a job is in REFRESHPDB stage, you can force it to start immediately
--Check documentation for other options

upg> proceed -job 101


--Or postpone it
upg> proceed -job 101 -newstarttime +2h30m
```

```
--When a job is in REFRESHPDB stage, you can force it to start immediately
--Check documentation for other options

upg> proceed -job 101


--Or postpone it
upg> proceed -job 101 -newstarttime +2h30m


--Or reschedule it
upg> proceed -job 101 -newstarttime 15/10/2025 15:45:00
```

Works for *unplug-plug* upgrades as well

Perfect for ExaScale migrations, too

The source remains untouched

```
upg1.close_source=no
```

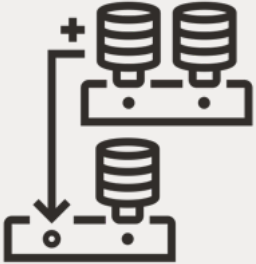
- Default: YES

Refreshable clone works only with deferred recovery on standby database

- You must restore the PDB on standby database after disconnect from non-CDB / PDB

Further Information

Refreshable Clone PDBs



- After creating the refreshable clone PDB, don't restart the source database
- In the source database, refreshable clone PDB supports:
  - Creating new tablespaces
  - Extending existing data files
  - Adding new data files

Further Information

Refreshable Clone PDBs

Data Guard and Refreshable Clone PDBs when using ASM and OMF

- MOS Note: 2273304.1
  Reusing the Source Standby Database Files When Plugging a non-CDB as a PDB into the Primary Database of a Data Guard Configuration
- MOS Note: 1916648.1
  Making Use of Deferred PDB Recovery and the STANDBYS=NONE Feature with Oracle Multitenant

Key Benefits of Upgrade via Refreshable Clone PDB

1. No interruption
2. Excellent testing option
3. Fully automated by AutoUpgrade

Operational Life Hack 3

Refreshable Clone PDBs with Oracle GoldenGate

Portuguese Government Agency

Move from 19c PDB to 23ai on-prem with very large databases and very little downtime

Customer Case | Government Agency - Portugal

Customer

Project

Constraints

Preparation

Upgrade

Success?

Remarks

Financial services in public sector

- Very important for Portugal

Customer Case | Government Agency - Portugal

Customer

Project

Constraints

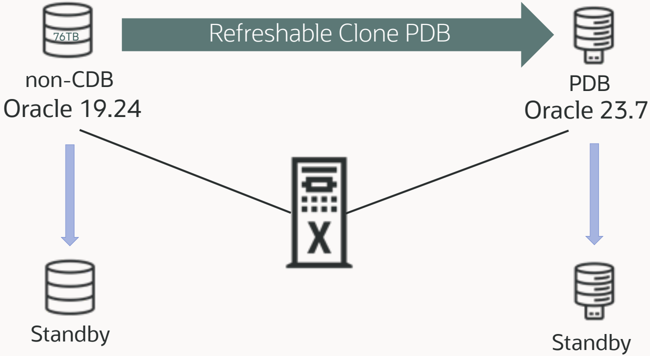
Preparation

Upgrade

Success?

Remarks

Upgrade and migrate 76TB Data Lake



Customer Case | Government Agency - Portugal

Customer

Project

Constraints

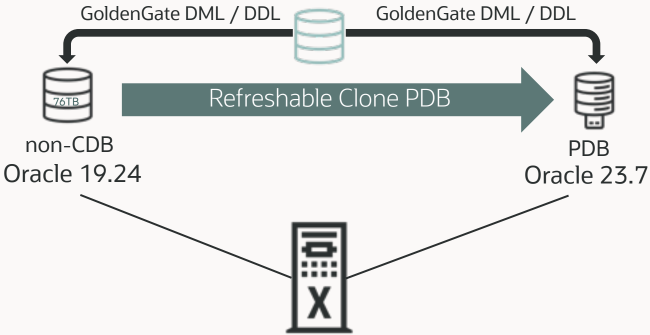
Preparation

Upgrade

Success?

Remarks

Upgrade and migrate 76TB Data Lake



Customer Case | Government Agency - Portugal

Customer

Project

Constraints

Preparation

Upgrade

Success?

Remarks

Downtime less 1 hour

Rebuild Standby environment

Keep operational:

- Oracle GoldenGate
- Audit Vault and Database Firewall

Customer Case | Government Agency - Portugal

Customer

Project

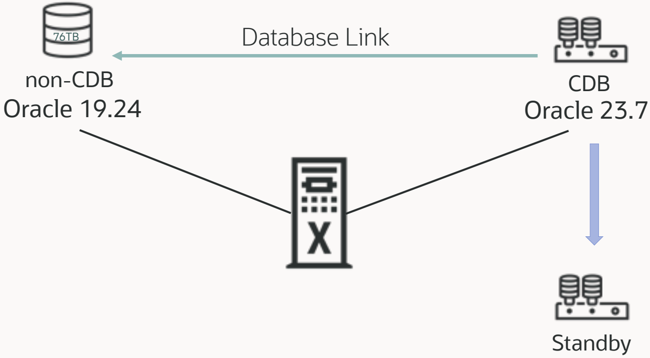
Constraints

Preparation

Upgrade

Success?

Remarks



Customer Case | Government Agency - Portugal

Customer

Project

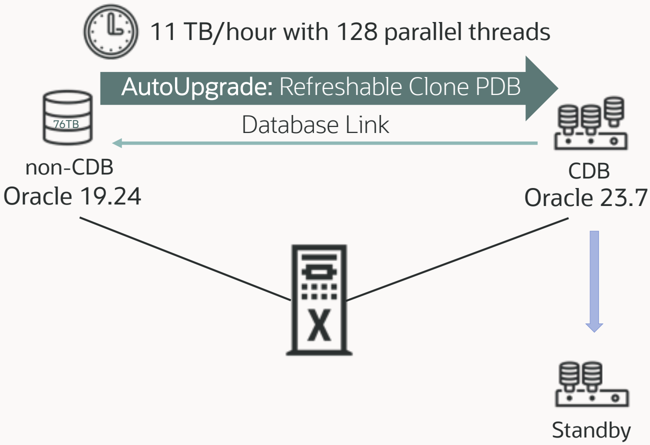
Constraints

Preparation

Upgrade

Success?

Remarks



11 TB/hour with 128 parallel threads

AutoUpgrade: Refreshable Clone PDB

Database Link

non-CDB
Oracle 19.24

CDB
Oracle 23.7

Standby

Customer Case | Government Agency - Portugal

Customer

Project

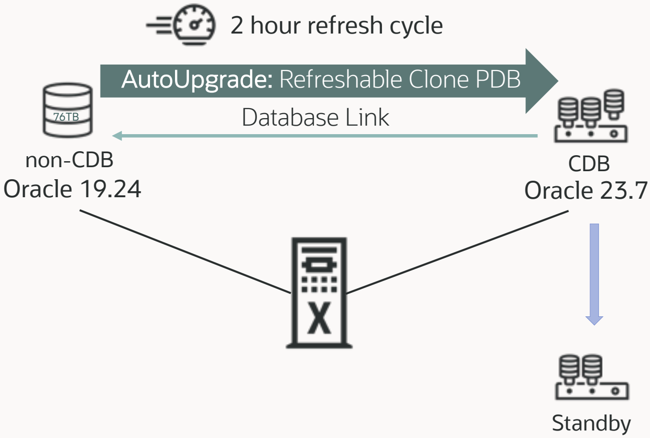
Constraints

Preparation

Upgrade

Success?

Remarks



Customer Case | Government Agency - Portugal

Customer

Project

Constraints

Preparation

Upgrade

Success?

Remarks

AutoUpgrade `proceed` feature

1. Define `START_TIME` in the future, e.g.
   `upg1.START_TIME=31/10/2025 13:00:00`
2. Run the clone operation until it completes and refreshes
3. Then adjust AutoUpgrade and set a new `START_TIME`:
   - `proceed -job 100 -newStartTime 15/10/2025 15:45:00`

Customer Case | Government Agency - Portugal

Customer

Project

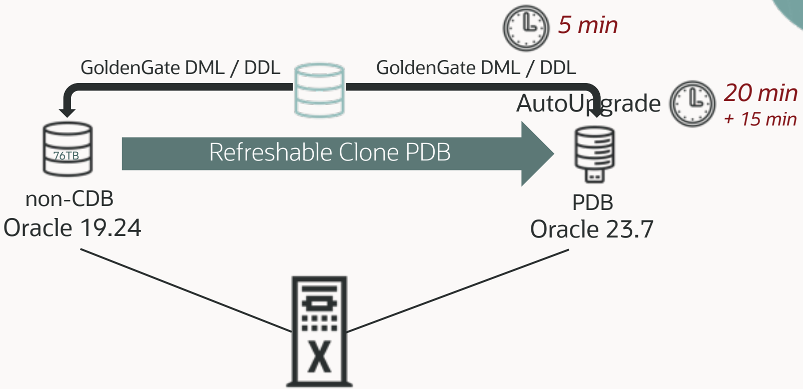
Constraints

Preparation

Upgrade

Success?

Remarks



Customer Case | Government Agency - Portugal

Customer

Project

Constraints

Preparation

Upgrade

Success?

Remarks

Massive success!!

Downtime less than 45 minutes

Kickoff for future projects

- More upgrades into PDBs in Oracle Database 23ai
- AutoUpgrade proven to be THE solution
- 80TB DWH
- Migrations from legacy zLinux

Customer Case | Government Agency - Portugal

Customer

Project

Constraints

Preparation

Upgrade

Success?

Remarks

Standby building

Oracle GoldenGate move to 23ai

- Schema cleanout needed

Operational Life Hack 4

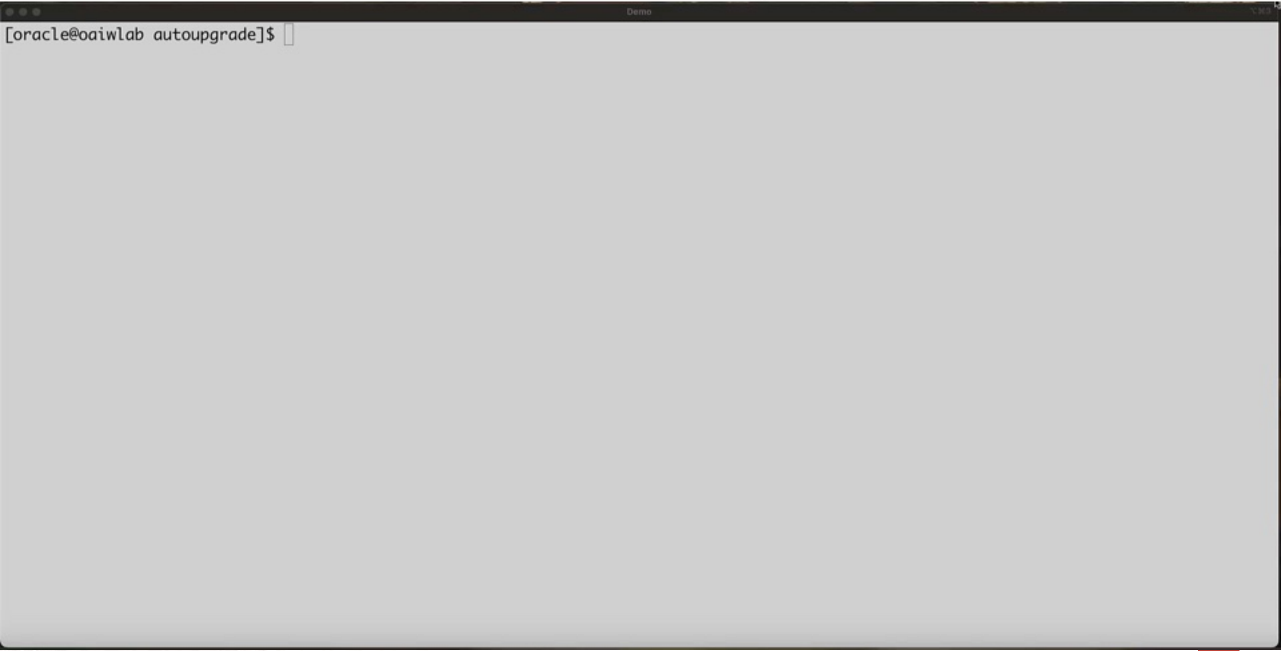
Download all your software

```
$ cat DB19.cfg

global.keystore=/home/oracle/autoupgrade-patching/keystore
patch1.source_home=/u01/app/oracle/product/19/dbhome_19_27_0
patch1.target_home=/u01/app/oracle/product/19/dbhome_19_28_0
patch1.sid=DB19
patch1.folder=/home/oracle/autoupgrade-patching/patch
patch1.patch=RECOMMENDED,OCW
```

```
./opatch lspatches

37777295;DATAPUMP BUNDLE PATCH 19.27.0.0.0
37499406;OJVM RELEASE UPDATE: 19.27.0.0.250415 (37499406)
37642901;Database Release Update : 19.27.0.0.250415 (37642901)
29585399;OCW RELEASE UPDATE 19.3.0.0.0 (29585399)
```

```
[oracle@oaiwlab:~]$ 
```

```
$ cat DB19.cfg

global.global_log_dir=/home/oracle/autoupgrade/logs
global.keystore=/home/oracle/autoupgrade/keystore

patch1.target_version=19
patch1.platform=LINUX.X64
patch1.folder=/home/oracle/autoupgrade/patches
patch1.patch=OCW
```



```
[oracle@oaiwlab autoupgrade]$ 
```

OJVM is embedded in Release Updates

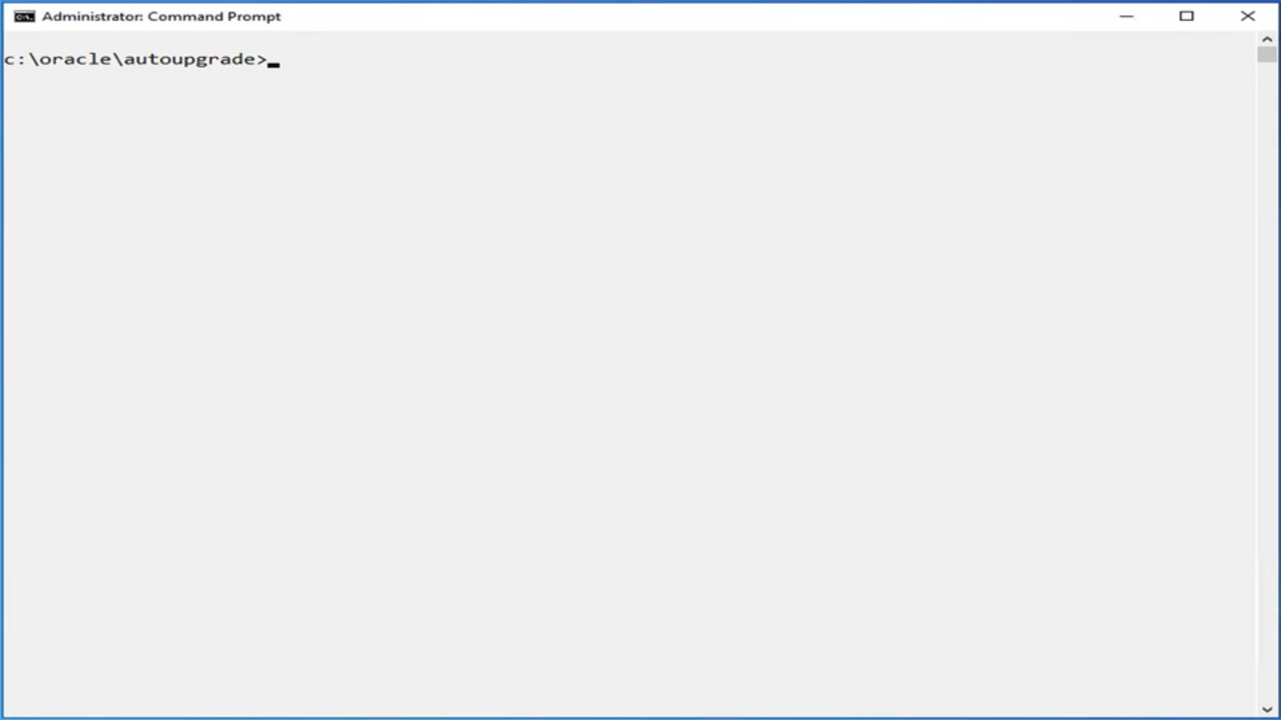
- No separate download
- Complete RAC Rolling patching support

```
$ cat install-OH.cfg

global.global_log_dir=/home/oracle/autoupgrade/logs
install1.target_version=23
install1.patch=RECOMMENDED,37693383,37393792
install1.folder=/u01/app/oracle/software
install1.download=no
install1.target_home=/u01/app/oracle/product/dbhome_23_9
install1.home_settings.edition=EE
```

```
$ cat install-OH.cfg

global.global_log_dir=/home/oracle/autoupgrade/logs
install1.target_version=23
install1.patch=RECOMMENDED,37693383,37393792
install1.folder=/u01/app/oracle/software
install1.download=no
install1.target_home=/u01/app/oracle/product/dbhome_23_9
install1.home_settings.edition=EE
```

```
java -jar autoupgrade.jar -config install-OH.cfg -mode create_home
```

```
$ cat download.cfg

global.keystore=c:\oracle\autoupgrade\keystore
global.patch=RU:19.28,OPATCH,OJVM,DPBP,OCW

patch1.platform=LINUX.X64
patch1.folder=f:\nfs\oracle\patches
```



```
c:\oracle\autoupgrade>
```

Download mode is available now for **non-admin** users on MS Windows, too

Operational Life Hack 5

Set all the secret underscores

```
--This is a list of our most favorite underscore parameters

alter system set "_cursor_obsolete_threshold"=1024;
alter system set "_sql_plan_directive_mgmt_control"=0;
alter system set "_column_tracking_level"=1;
alter system set "_exclude_pdb_seed_view"=false;
```

```
vi my_underscores.ora
```

```
_cursor_obsolete_threshold=1024
_sql_plan_directive_mgmt_control=0
_column_tracking_level=1
_exclude_pdb_seed_view=false
```

```
global.autoupg_log_dir=/home/oracle/logs/autoupgrade-UPGR

upg1.source_home=/u01/app/oracle/product/19_27
upg1.target_home=/u01/app/oracle/product/19_28
upg1.sid=UPGR
upg1.add_after_upgrade_pfile=/home/oracle/my_underscores.ora
```

But you don't want underscores?

```
global.autoupg_log_dir=/home/oracle/logs/autoupgrade-UPGR

upg1.source_home=/u01/app/oracle/product/19_27
upg1.target_home=/u01/app/oracle/product/19_28
upg1.sid=UPGR
upg1.remove_underscore_parameters=yes
```

Operational Life Hack 6

Sample Config

```
java -jar autoupgrade.jar -create_sample_file config
Created sample configuration file /home/oracle/sample_config.cfg
```

```
global.global_log_dir=<$ORACLE_BASE/cfgtoollogs/upgrade or /tmp/upgrade>
upg1.sid=<$ORACLE_SID or {SID}}>
upg1.source_home=<$ORACLE_HOME or /u01/app/oracle/product/12.2/dbhome_1>
upg1.target_home=<$ORACLE_TARGET_HOME or /u01/app/oracle/product/23/dbhome_1>

#global.keystore=/u01/app/oracle/admin/ORCL/keystore
#upg1.drop_grp_after_upgrade=
#upg1.restoration=
#upg1.add_after_upgrade_pfile=
#upg1.drop_after_upgrade_pfile=
#upg1.before_action=/u01/app/oracle/admin/ORCL/before_upgrade.sh
#upg1.after_action=/u01/app/oracle/admin/ORCL/after_upgrade.sh
#upg1.drop_win_src_service=
#upg1.wincredential=C:Usersoraclecred
#upg1.log_dir=/u01/app/oracle/admin/ORCL/upgrade_logs
#upg1.raise_compatible=
#upg1.run_dictionary_health=
#upg1.timezone_upg=
```

Operational Life Hack 7

AutoConfig makes it even easier

```
$ env | grep ORA

ORACLE_SID=UPGR
ORACLE_BASE=/u01/app/oracle
ORACLE_HOME=/u01/app/oracle/product/19
```

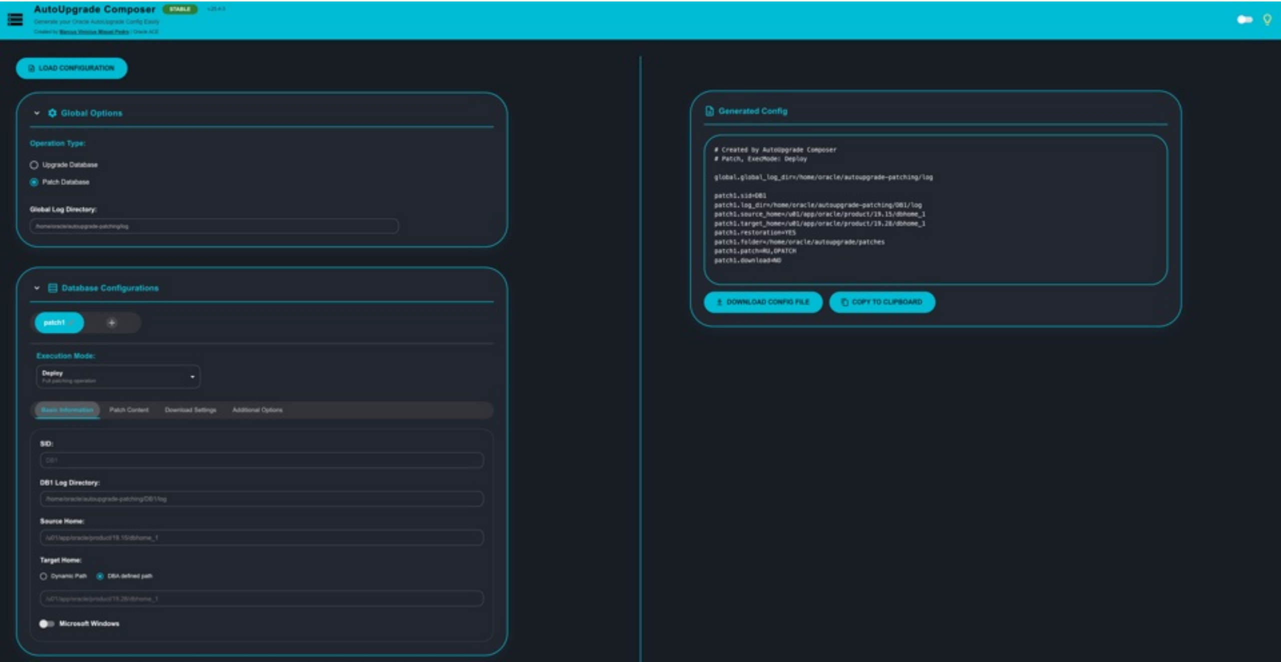
```
$ env | grep ORA

ORACLE_SID=UPGR
ORACLE_BASE=/u01/app/oracle
ORACLE_HOME=/u01/app/oracle/product/19
```

```
java -jar autoupgrade.jar -auto_config
```

```
[oracle@oaiwlab:~]$ 
```

AutoUpgrade Composer

https://viniciusdba.com.br/autoupgrade-composer/



Operation Type:

○ Upgrade Database

● Patch Database

Global Log Directory: /home/oracle/autoupgrade-patching/log

Execution Mode: Deploy
Full patching operation

SID: DB1

DB1 Log Directory: /home/oracle/autoupgrade-patching/DB1/log

Source Home: /u01/app/oracle/product/19.15/dbhome_1

Target Home:

○ Dynamic Path ● DBA-defined path

/u01/app/oracle/product/19.28/dbhome_1

Microsoft Windows

```
# Created by AutoUpgrade Composer
# Patch, ExecMode: Deploy

global.global_log_dir=/home/oracle/autoupgrade-patching/log

patch1.sid=DB1
patch1.log_dir=/home/oracle/autoupgrade-patching/DB1/log
patch1.source_home=/u01/app/oracle/product/19.15/dbhome_1
patch1.target_home=/u01/app/oracle/product/19.28/dbhome_1
patch1.restoration=YES
patch1.folder=/home/oracle/autoupgrade/patches
patch1.patch=RU,OPATCH
patch1.download=NO
```

Operational Life Hack 8

Using Refreshable Clone PDBs from a Standby Database

```
DGMGRL> convert database '...' to snapshot standby;
```

```
alter pluggable database ... open;
```

- In source standby, create the cloning user
- In target CDB, create database link pointing to source standby
- AutoUpgrade config file
- Start AutoUpgrade in deploy mode

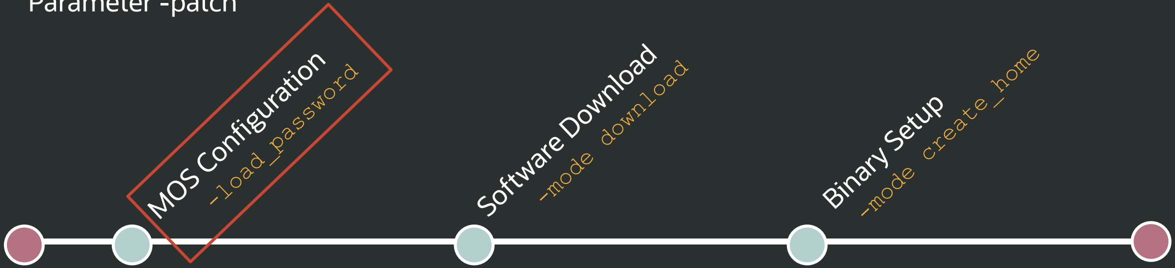
```
DGMGRL> convert database '...' to physical standby;
```

Operational Life Hack 9

Let's hear from a customer

Autoupgrade – A new ORACLE_HOME in Online Mode

Parameter -patch



```
java -jar autoupgrade.jar
-config create-home-web.config
-patch -mode create_home
```

Pro Tip: Expect

Used to automate interactions with programs

```
spawn java
-jar "{{ autoupgrade_base }}/autoupgrade.jar"
-config "{{ autoupgrade_base }}/get-patches.config"
-patch -load_password

expect {
    "Enter password:" {
        send "$keystore_password\r"
        exp_continue
    }
```

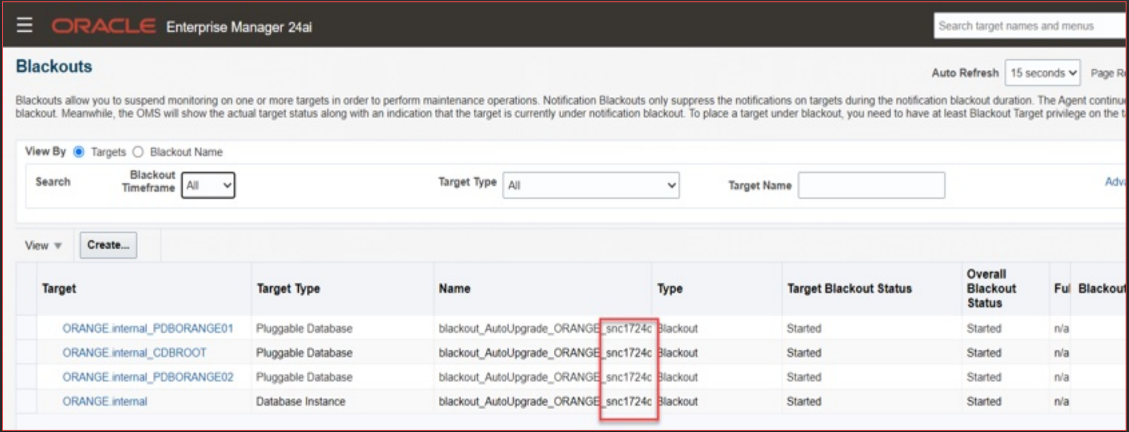
The EMCLI-Autoupgrade Combo

Streamline upgrades with your Monitoring

```
# Global Settings
global.autoupg_log_dir=/u01/app/oracle/cfgtoollogs/autoupgrade
upg1.log_dir=/u01/app/oracle/cfgtoollogs/autoupgrade
upg1.sid=ORANGE
upg1.source_home=/u01/app/oracle/product/19.0.0/dbhome_1
upg1.target_home=/u01/app/oracle/product/19.0.0/dbhome_2
upg1.start_time=NOW
upg1.upgrade_node=dbengine02.internal
upg1.emcli_path=/u01/app/oracle/emcli
upg1.em_target_name=ORANGE.internal
upg1.em_blackout_suffix=snc1724o195
```

Pro Tip: Use suffixes

Identify your autoupgrade jobs.



Blackouts

Auto Refresh: 15 seconds ▾   Page R

Blackouts allow you to suspend monitoring on one or more targets in order to perform maintenance operations. Notification Blackouts only suppress the notifications on targets during the notification blackout duration. The Agent continu blackout. Meanwhile, the OMS will show the actual target status along with an indication that the target is currently under notification blackout. To place a target under blackout, you need to have at least Blackout Target privilege on the t

View By ● Targets ○ Blackout Name

Search: Blackout Timeframe: All ▾   Target Type: All ▾   Target Name:

Target	Target Type	Name	Type	Target Blackout Status	Overall Blackout Status	Ful	Blackout
ORANGE.internal_PDBORANGE01	Pluggable Database	blackout_AutoUpgrade_ORANGE_snc1724c	Blackout	Started	Started	n/a	
ORANGE.internal_CDBROOT	Pluggable Database	blackout_AutoUpgrade_ORANGE_snc1724c	Blackout	Started	Started	n/a	
ORANGE.internal_PDBORANGE02	Pluggable Database	blackout_AutoUpgrade_ORANGE_snc1724c	Blackout	Started	Started	n/a	
ORANGE.internal	Database Instance	blackout_AutoUpgrade_ORANGE_snc1724c	Blackout	Started	Started	n/a	

Ansible ♥ Autoupgrade

Consistency & Reliability

Automation ensures every new **ORACLE_HOME** is created the same way, reducing human error and configuration drift.

Speed & Efficiency

Tasks that normally take hours can be executed in minutes, allowing DBAs to focus on higher-value activities instead of repetitive manual steps.

Scalability

With automation, rolling out new ORACLE_HOME versions across many servers becomes a single streamlined process instead of a per-database effort.

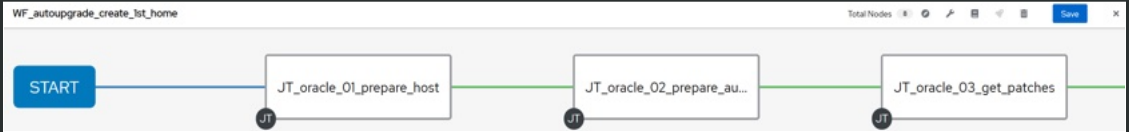
CI/CD

Automated ORACLE_HOME creation fits seamlessly into Oracle AutoUpgrade patching pipelines, ensuring end-to-end automation from software installation to database upgrade.

Pro Tip: Use the Power of Automation

17 min

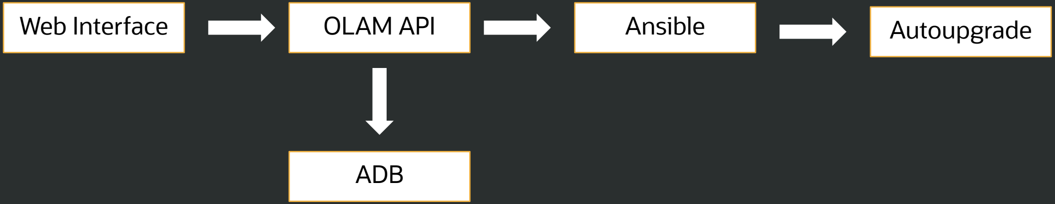
Oracle Linux Automation Manager 2.x


WF_autoupgrade_create_1st_home
Total Nodes
Save
START
JT_oracle_01_prepare_host
JT
JT_oracle_02_prepare_au...
JT
JT_oracle_03_get_patches
JT

```
# Run the autoupgrade command
- name: Run autoupgrade command to download patches
  ansible.builtin.command:
     java -jar {{ autoupgrade_base }}/autoupgrade.jar
       -config {{ autoupgrade_base }}/get-patches-web.config
       -patch -mode download"
  become: true
  become_user: oracle
  when: autoupgrade_source == "web"
```

Autoupgrade Automation Framework

Ansible, Oracle Linux Automation Manager, Oracle Autonomus Database, Python3 (Flask)



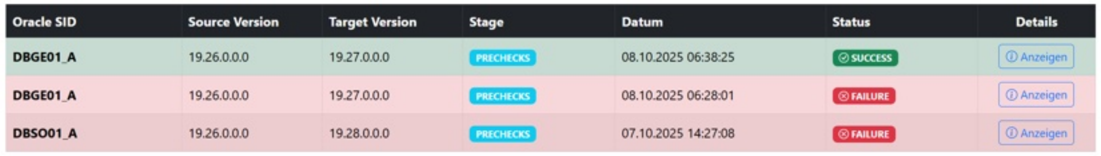
Web Interface

OLAM API

Ansible

Autoupgrade

ADB



Oracle SID	Source Version	Target Version	Stage	Datum	Status	Details
DBGE01_A	19.26.0.0.0	19.27.0.0.0	PRECHECKS	08.10.2025 06:38:25	✓ SUCCESS	ⓘ Anzeigen
DBGE01_A	19.26.0.0.0	19.27.0.0.0	PRECHECKS	08.10.2025 06:28:01	⊗ FAILURE	ⓘ Anzeigen
DBSO01_A	19.26.0.0.0	19.28.0.0.0	PRECHECKS	07.10.2025 14:27:08	⊗ FAILURE	ⓘ Anzeigen

3 Tips for Beginners in Ansible Automation

Configuration

Invest in a solid inventory and variables management – strictly split configuration and plays.

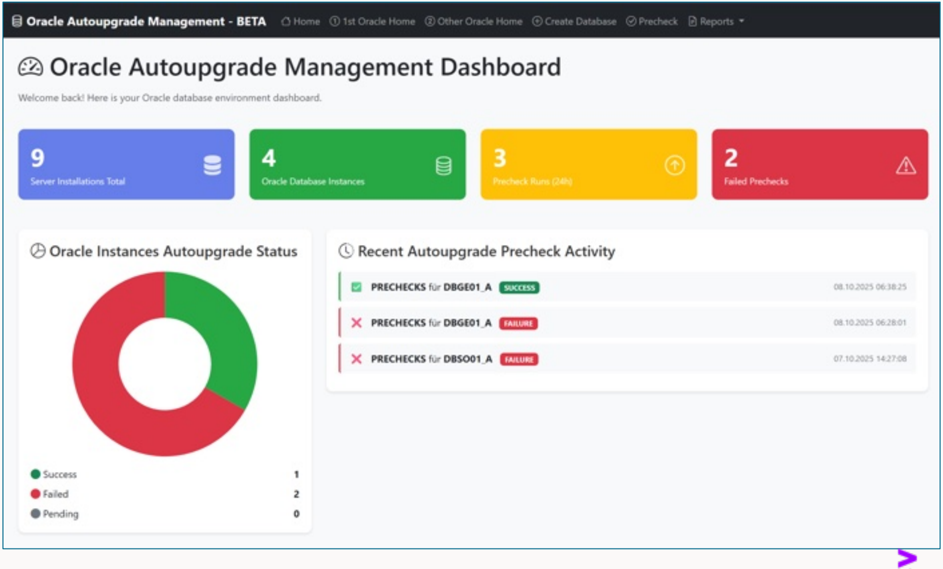
Use Cases

Real-world use-cases, not just for fun.

AI Coding

Use AI tools like Claude, ChatGPT to review your Ansible code, adding comments, recommend improvements – not to write your playbooks.

Screenshot 1



Oracle Autoupgrade Management Dashboard

Welcome back! Here is your Oracle database environment dashboard.

9 Server Installations Total

4 Oracle Database Instances

3 Precheck Runs (24h)

2 Failed Prechecks

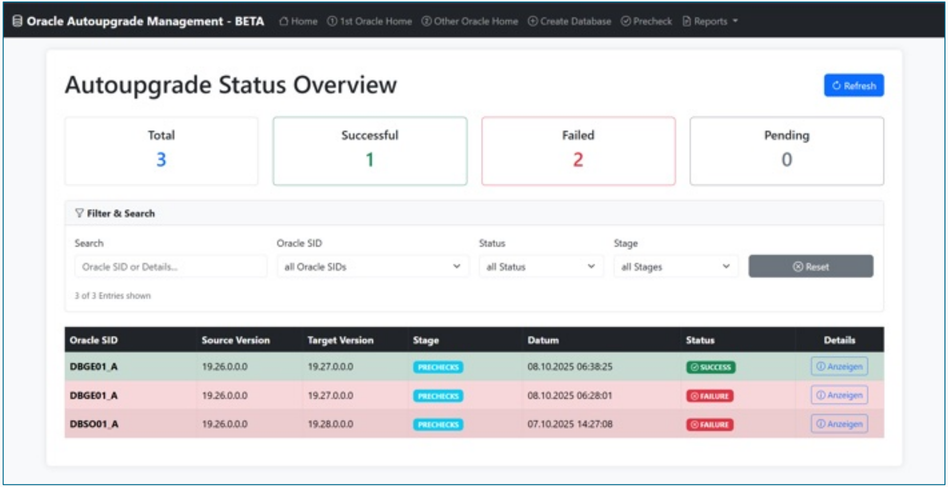
Oracle Instances Autoupgrade Status

🟢 Success	1
🔴 Failed	2
● Pending	0

Recent Autoupgrade Precheck Activity

☑	**PRECHECKS** für **DBGE01_A**	SUCCESS	08.10.2025 06:38:25
✗	**PRECHECKS** für **DBGE01_A**	FAILURE	08.10.2025 06:28:01
✗	**PRECHECKS** für **DBSO01_A**	FAILURE	07.10.2025 14:27:08

Screenshot 2



Autoupgrade Status Overview

Total	Successful	Failed	Pending
3	1	2	0

Search: Oracle SID or Details...

Oracle SID: all Oracle SIDs

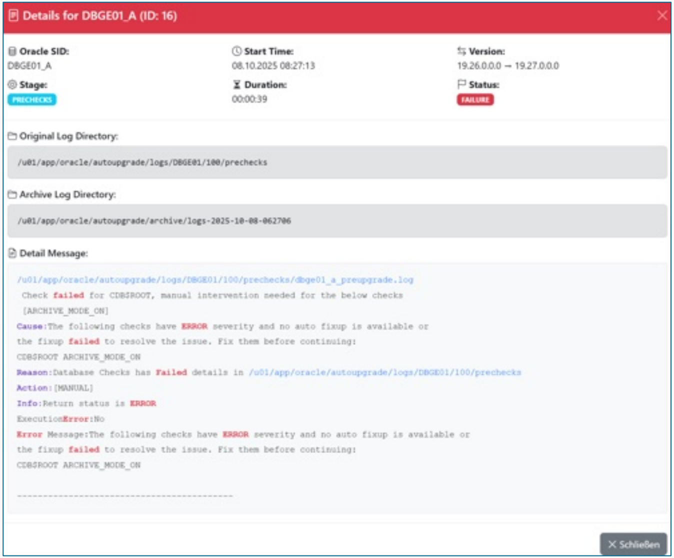
Status: all Status

Stage: all Stages

3 of 3 Entries shown

Oracle SID	Source Version	Target Version	Stage	Datum	Status	Details
DBGE01_A	19.26.0.0.0	19.27.0.0.0	PRECHECKS	08.10.2025 06:38:25	SUCCESS	Anzeigen
DBGE01_A	19.26.0.0.0	19.27.0.0.0	PRECHECKS	08.10.2025 06:28:01	FAILURE	Anzeigen
DBSO01_A	19.26.0.0.0	19.28.0.0.0	PRECHECKS	07.10.2025 14:27:08	FAILURE	Anzeigen

Screenshot 3



Details for DBGE01_A (ID: 16)

Oracle SID: DBGE01_A

Start Time: 08.10.2025 08:27:13

Version: 19.26.0.0.0 → 19.27.0.0.0

Stage: PRECHECKS

Duration: 00:00:39

Status: FAILURE

Original Log Directory:

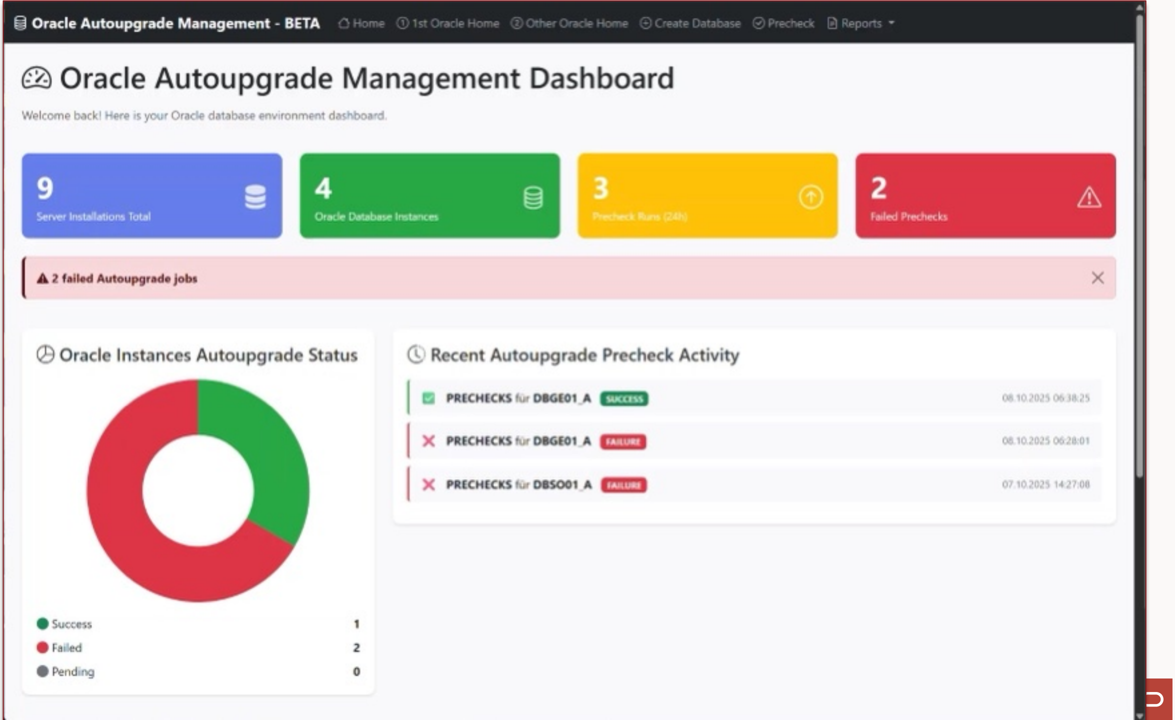
```
/u01/app/oracle/autoupgrade/logs/DBGE01/100/prechecks
```

Archive Log Directory:

```
/u01/app/oracle/autoupgrade/archive/logs-2025-10-08-062706
```

Detail Message:

```
/u01/app/oracle/autoupgrade/logs/DBGE01/100/prechecks/dbge01_a_preupgrade.log
 Check failed for CDB$ROOT, manual intervention needed for the below checks
 [ARCHIVE_MODE_ON]
Cause:The following checks have ERROR severity and no auto fixup is available or
the fixup failed to resolve the issue. Fix them before continuing:
CDB$ROOT ARCHIVE_MODE_ON
Reason:Database Checks has Failed details in /u01/app/oracle/autoupgrade/logs/DBGE01/100/prechecks
Action:[MANUAL]
Info:Return status is ERROR
ExecutionError:No
Error Message:The following checks have ERROR severity and no auto fixup is available or
the fixup failed to resolve the issue. Fix them before continuing:
CDB$ROOT ARCHIVE_MODE_ON


-------------------------------------------
```



Oracle Autoupgrade Management Dashboard

Welcome back! Here is your Oracle database environment dashboard.

9	Server Installations Total
4	Oracle Database Instances
3	Precheck Runs (24h)
2	Failed Prechecks

▲ **2 failed Autoupgrade jobs**

Oracle Instances Autoupgrade Status

● Success	1
● Failed	2
● Pending	0

Recent Autoupgrade Precheck Activity

☑ **PRECHECKS** für **DBGE01_A** SUCCESS 08.10.2025 06:38:25

✗ **PRECHECKS** für **DBGE01_A** FAILURE 08.10.2025 06:28:01

✗ **PRECHECKS** für **DBSO01_A** FAILURE 07.10.2025 14:27:08

Key Learnings

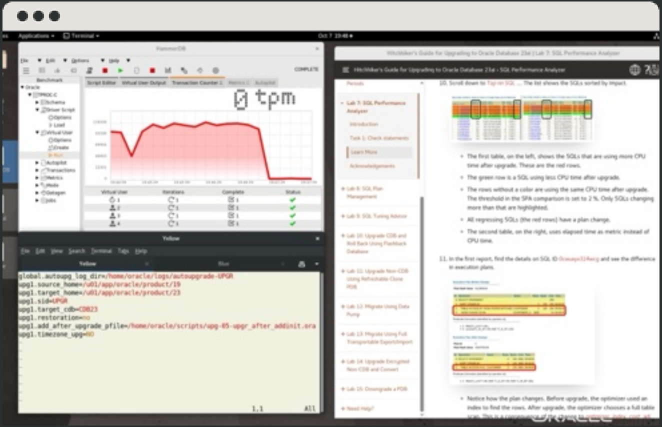
1 Start preparing today

2 Use AutoUpgrade

3 Caution with Data Guard and Multitenant conversion

Try it out – and improve your skills

Oracle LiveLabs:
[Hitchhiker's Guide for Upgrading to Oracle Database 23ai]



HammerDB
0 tpm
Yellow
Blue
Hitchhiker's Guide for Upgrading to Oracle Database 23ai › SQL Performance Analyzer
Lab 7: SQL Performance Analyzer

```
global.autoupg_log_dir=/home/oracle/logs/autoupgrade-UPGR
upg1.source_home=/u01/app/oracle/product/19
upg1.target_home=/u01/app/oracle/product/23
upg1.sid=UPGR
upg1.target_cdb=CDB23
upg1.restoration=no
upg1.add_after_upgrade_pfile=/home/oracle/scripts/upg-05-upgr_after_addinit.ora
upg1.timezone_upg=NO
```

10. Scroll down to Top-nn SQL ... The list shows the SQLs sorted by impact.

- The first table, on the left, shows the SQLs that are using more CPU time after upgrade. These are the red rows.
- The green row is a SQL using less CPU time after upgrade.
- The rows without a color are using the same CPU time after upgrade. The threshold in the SPA comparison is set to 2 %. Only SQLs changing more than that are highlighted.
- All regressing SQLs (the red rows) have a plan change.
- The second table, on the right, uses elapsed time as metric instead of CPU time.

11. In the first report, find the details on SQL ID [illegible] and see the difference in execution plans.

- Notice how the plan changes. Before upgrade, the optimizer used an index to find the rows. After upgrade, the optimizer chooses a full table scan.

Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

ORACLE