ORACLE

Rolling Upgrades

Upgrade your DB with near Zero Downtime

Francisco Munoz Alvarez

Distinguished Product Manager

Oracle Database High Availability (HA), Scalability and Maximum Availability Architecture (MAA) Team

@fcomunoz

http://www.linkedin.com/in/franciscomunozalvarez

www.oraclemaa.com

Oracle (Active) Data Guard & MAA

Impact of database downtime

$350K

average cost of downtime per hour

$10M

average cost of unplanned data center outage or disaster

87 hours

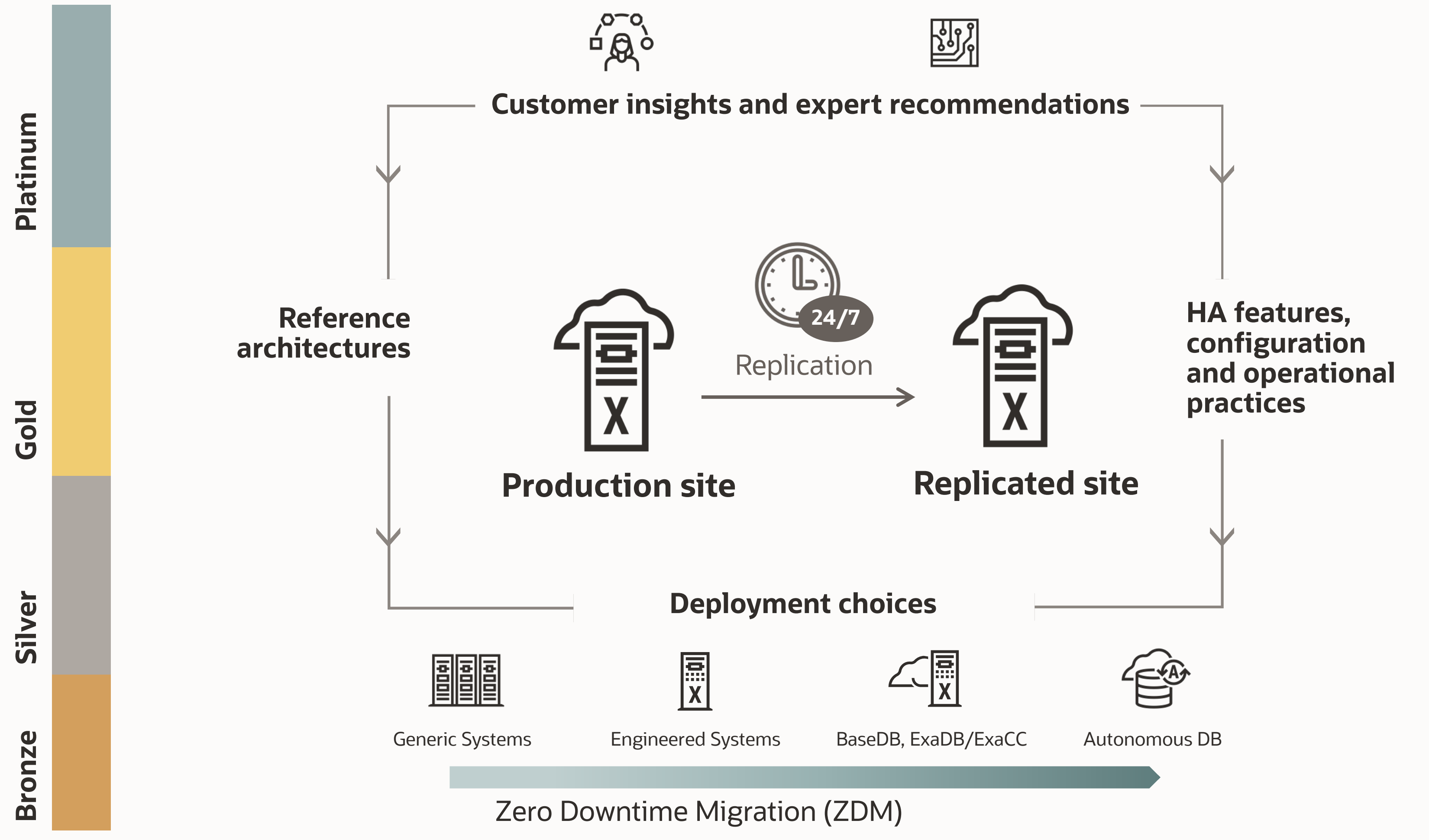
average amount of downtime per year

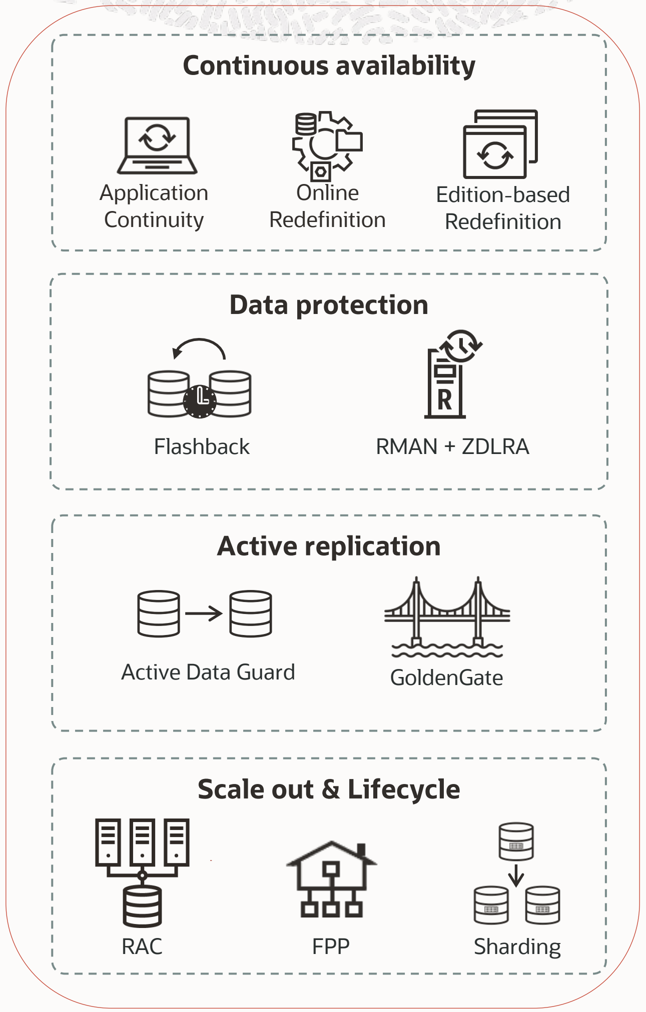
91%

percentage of companies that have experienced an unplanned data center outage in the last 24 months

Oracle Maximum Availability Architecture (MAA)

Standardized Reference Architectures for Never-Down Deployments





MAA reference architectures

Availability service levels

Bronze

Dev, test, prod

Single instance DB

Restartable

Backup/restore

Silver

Prod/departmental

Bronze +

Database HA with RAC

Application continuity

Sharding (optional)

Gold

Business critical

Silver +

DB replication with Active Data Guard

Platinum

Mission critical

Gold +

GoldenGate

Edition-based redefinition

All tiers exist with on-premises and cloud. However, platinum currently must be configured manually while bronze to gold are covered with some form of cloud automation depending on the desired MAA architecture (i.e., multiple standby databases still must be manually configured in cloud today)

Challenges of deploying highly available systems

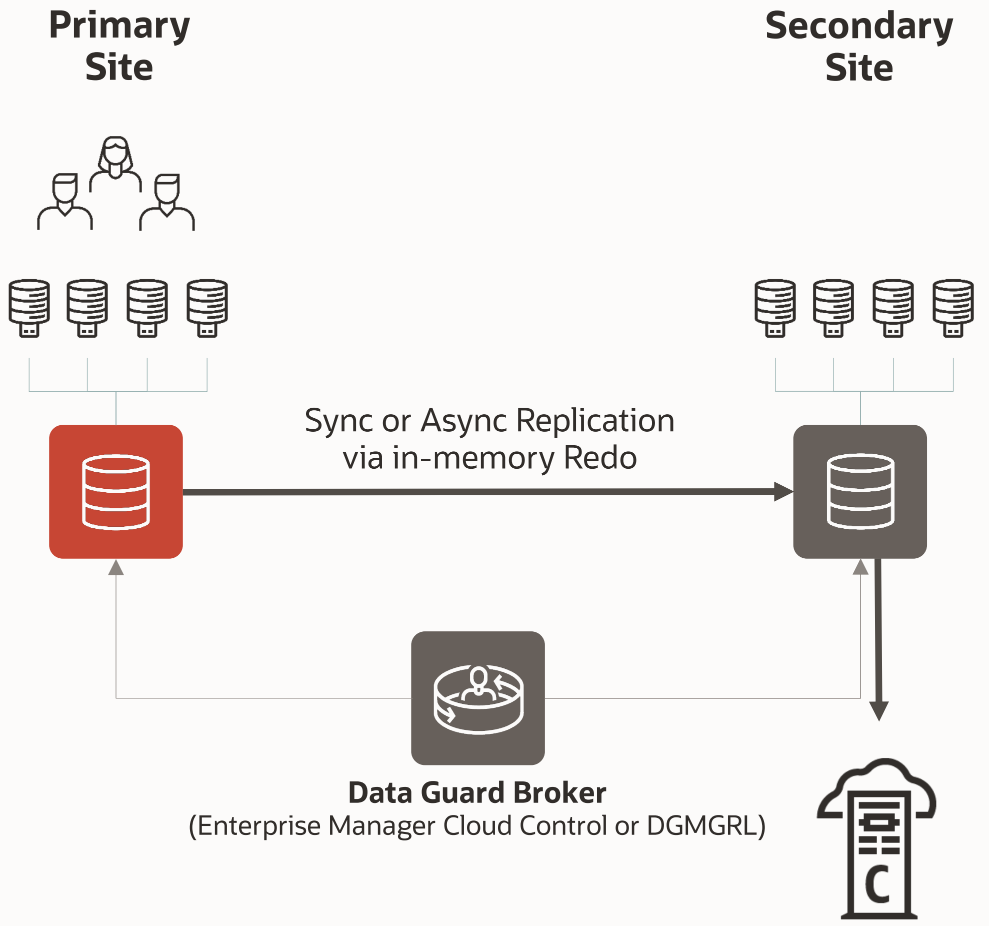
Cost and complexity

Lack of skills

Risk of failure

Oracle Data Guard Overview

Oracle Data Guard (DG)



Primary Site
Secondary Site
Sync or Async Replication via in-memory Redo
Data Guard Broker
(Enterprise Manager Cloud Control or DGMGRL)

- **Basic DR (included with DB EE)**
  - License primary and secondary sites
- **Active-passive**
  - Standby is used only for failovers
- **Automatic failover to Standby site**
- **Zero / near-zero data loss**
- **Continuous data validation**
- **Simple migrations and upgrades**

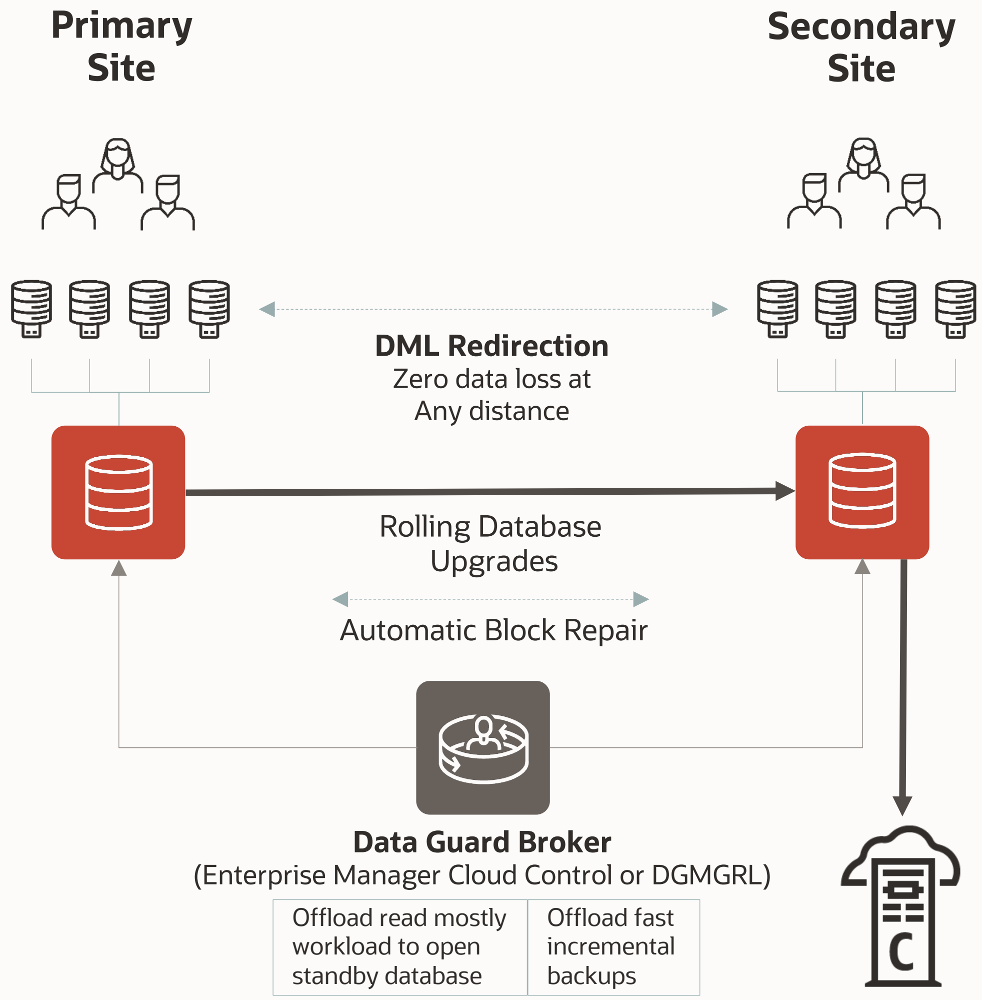
https://www.oracle.com/database/technologies/high-availability/dataguard-activedataguard-demos.html

Data Guard

Capabilities Included with Oracle Database Enterprise Edition (EE)

Data Protection	High Availability	Performance and ROI
Zero or sub-second data loss protection	Automatic database failover	Extreme throughput - supports all workloads
Strong isolation using continuous Oracle validation	Automatic client failover	Dual-purpose standby for development and test
Lost-write detection	Standby-first patch apply	Integrated management
Universal support – all data types and applications	Database rolling maintenance	
Comprehensive monitoring with Enterprise Manager	Select platform migrations	

Oracle *Active* Data Guard (ADG)


Primary Site
Secondary Site
DML Redirection
Zero data loss at Any distance
Rolling Database Upgrades
Automatic Block Repair
Data Guard Broker
(Enterprise Manager Cloud Control or DGMGRL)
Offload read mostly workload to open standby database
Offload fast incremental backups

ADG

- **Advanced Disaster Recovery**
- **Active-active***
  - Queries, reports, backups
  - Occasional updates (19c)
  - Assurance of knowing system is operational
- **Automatic block repair**
- **Application Continuity**
- **Zero data loss across any distance**
- **Many other features**

https://www.oracle.com/database/technologies/high-availability/dataguard-activedataguard-demos.html

Active Data Guard

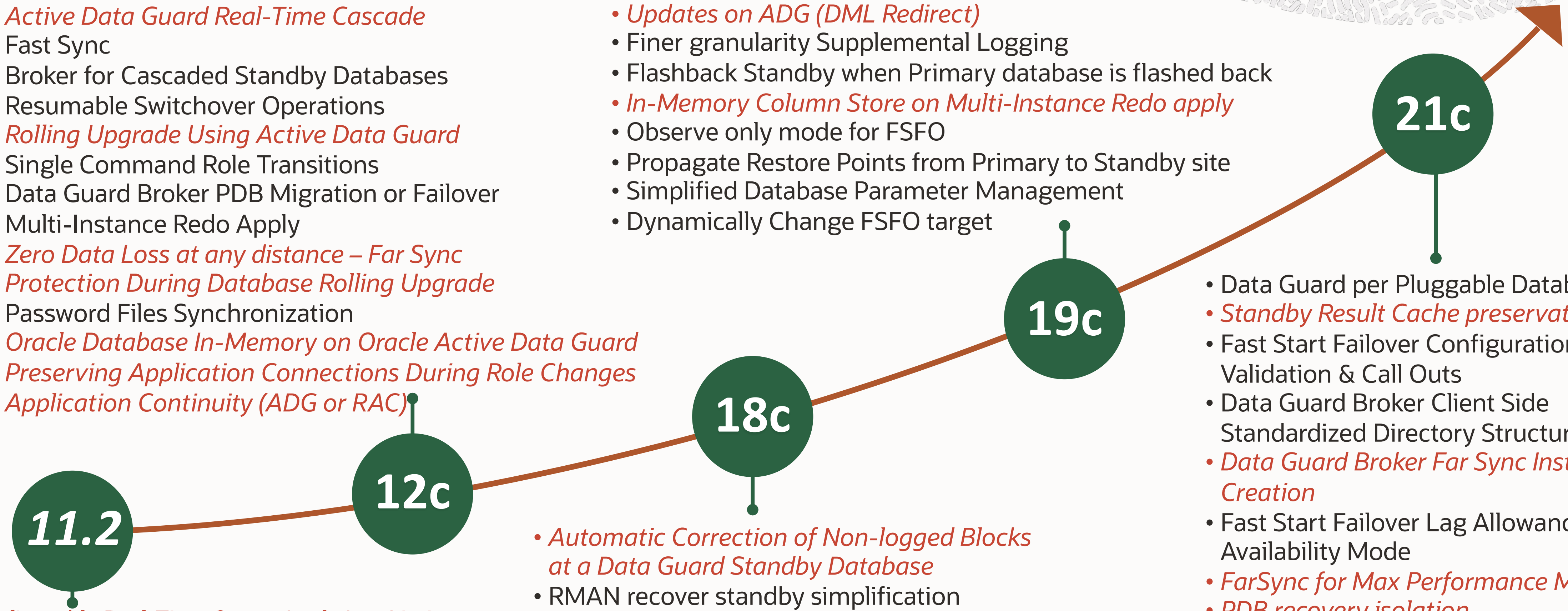
Option of Oracle Database for Advanced Capabilities and Protection

ADG

Data Protection	High Availability	Performance and ROI
Zero data loss at any distance	Automatic block repair	Extreme throughput - supports all workloads
Real-time cascade	Automated rolling database maintenance	Dual-purpose standby for development and test
Automatic Block Repair	Application continuity	Integrated management
	Service management for replicated databases	Offload network compression
	Rolling Upgrade	Intelligent load balancing for replicated databases
		Active Standby DML redirection

Oracle *Active* Data Guard

Actively protecting data for the future *both* on-premises and in the cloud



11.2

- *Configurable Real-Time Query Apply Lag Limit*
- Integrated Support for Application Failover
- *SPA Support for Active Data Guard Environment*
- Support Up to 30 Standby Databases

12c

- *Active Data Guard Real-Time Cascade*
- Fast Sync
- Broker for Cascaded Standby Databases
- Resumable Switchover Operations
- *Rolling Upgrade Using Active Data Guard*
- Single Command Role Transitions
- Data Guard Broker PDB Migration or Failover
- Multi-Instance Redo Apply
- *Zero Data Loss at any distance – Far Sync*
- *Protection During Database Rolling Upgrade*
- Password Files Synchronization
- *Oracle Database In-Memory on Oracle Active Data Guard*
- *Preserving Application Connections During Role Changes*
- *Application Continuity (ADG or RAC)*

18c

- *Automatic Correction of Non-logged Blocks at a Data Guard Standby Database*
- RMAN recover standby simplification
- Shadow Lost Write Protection
- *Transparent Application Continuity*
- *AWR reports for the standby workload*

19c

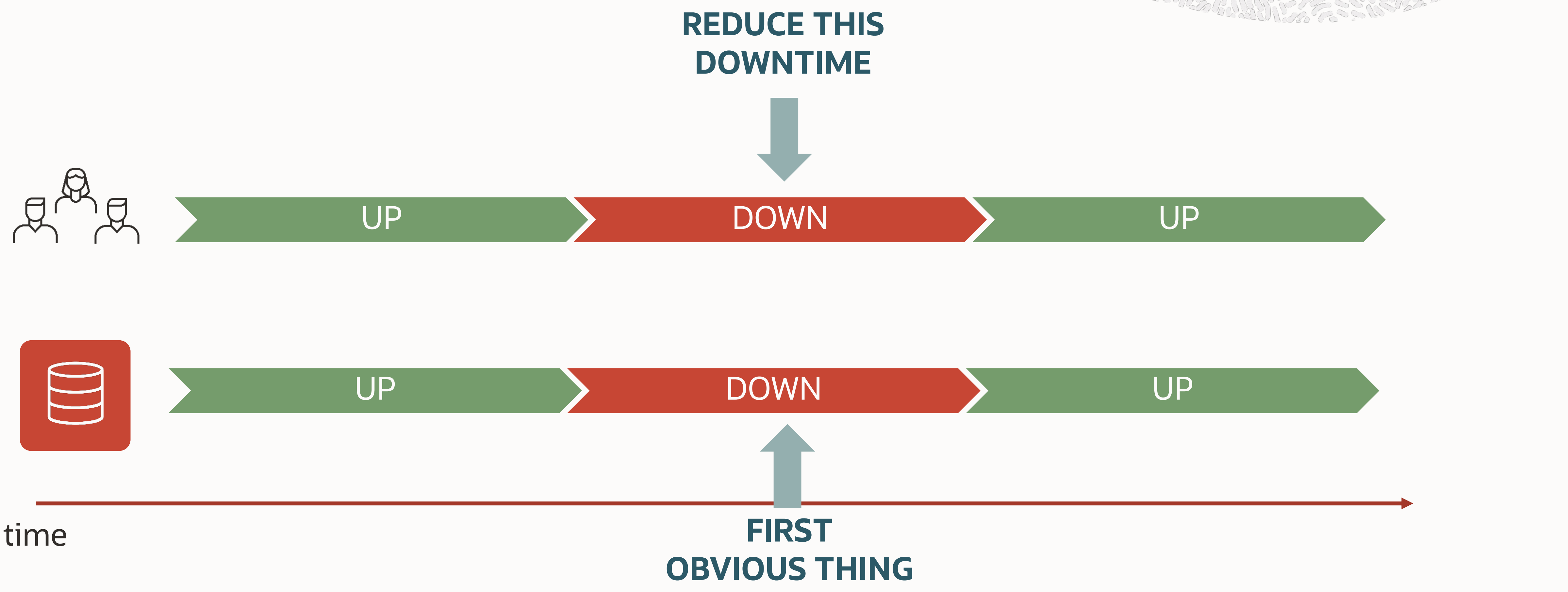
- *Updates on ADG (DML Redirect)*
- Finer granularity Supplemental Logging
- Flashback Standby when Primary database is flashed back
- *In-Memory Column Store on Multi-Instance Redo apply*
- Observe only mode for FSFO
- Propagate Restore Points from Primary to Standby site
- Simplified Database Parameter Management
- Dynamically Change FSFO target

21c

- Data Guard per Pluggable Database
- *Standby Result Cache preservation*
- Fast Start Failover Configuration Validation & Call Outs
- Data Guard Broker Client Side Standardized Directory Structure
- *Data Guard Broker Far Sync Instance Creation*
- Fast Start Failover Lag Allowance in Max Availability Mode
- *FarSync for Max Performance Mode*
- *PDB recovery isolation*

Oracle Active Data Guard
Rolling Maintenance and Upgrades

Database Downtime and Application Downtime



Fixups | Traditional

Analyze

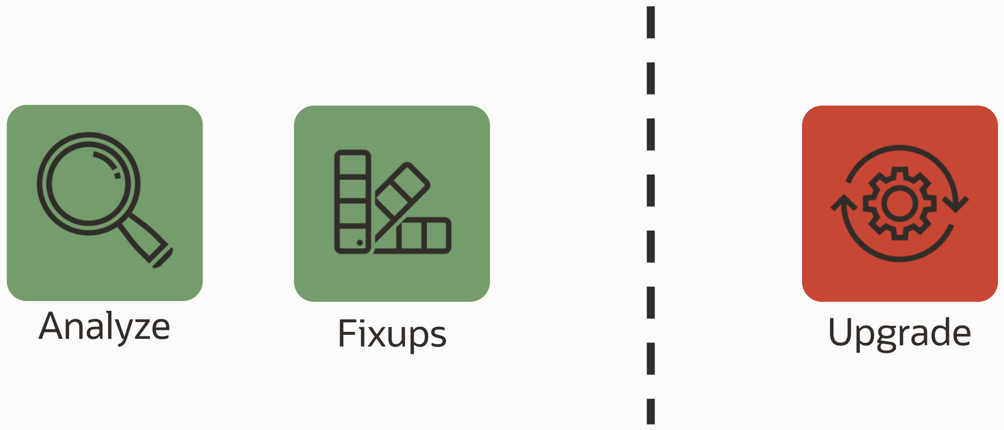
Analyze

Fixups

Upgrade

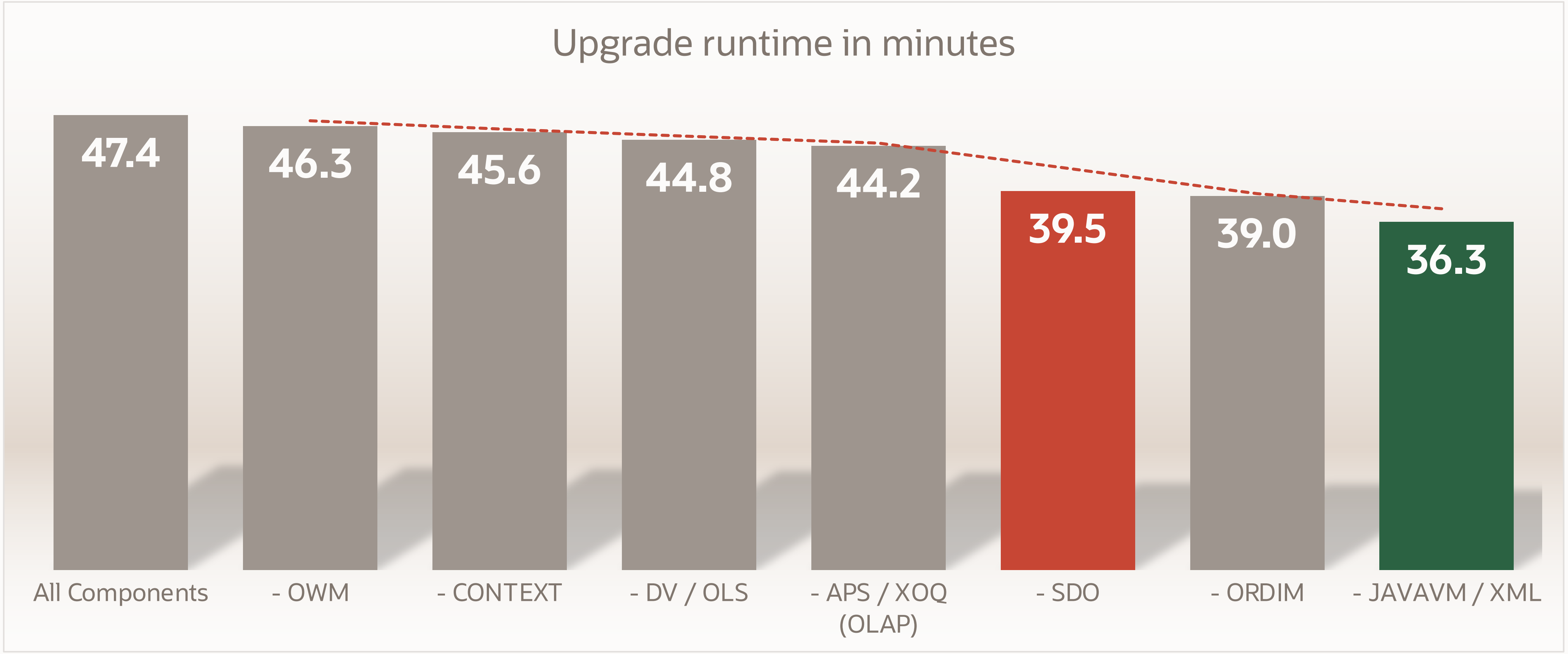
```
$ java -jar autoupgrade.jar -mode analyze

$ java -jar autoupgrade.jar -mode deploy
```

Fixups | Fast Deploy



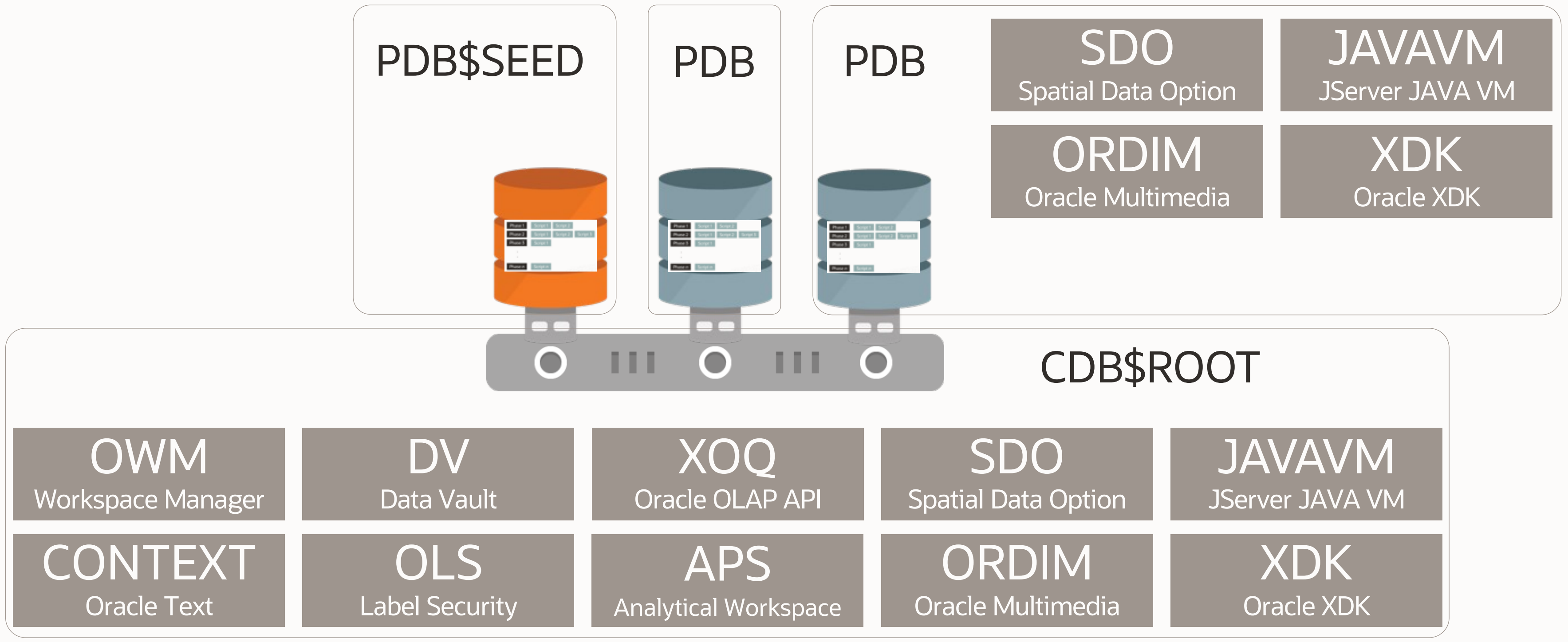
```
$ java -jar autoupgrade.jar -mode analyze
$ java -jar autoupgrade.jar -mode fixups

$ java -jar autoupgrade.jar -mode upgrade
```

Components | Impact

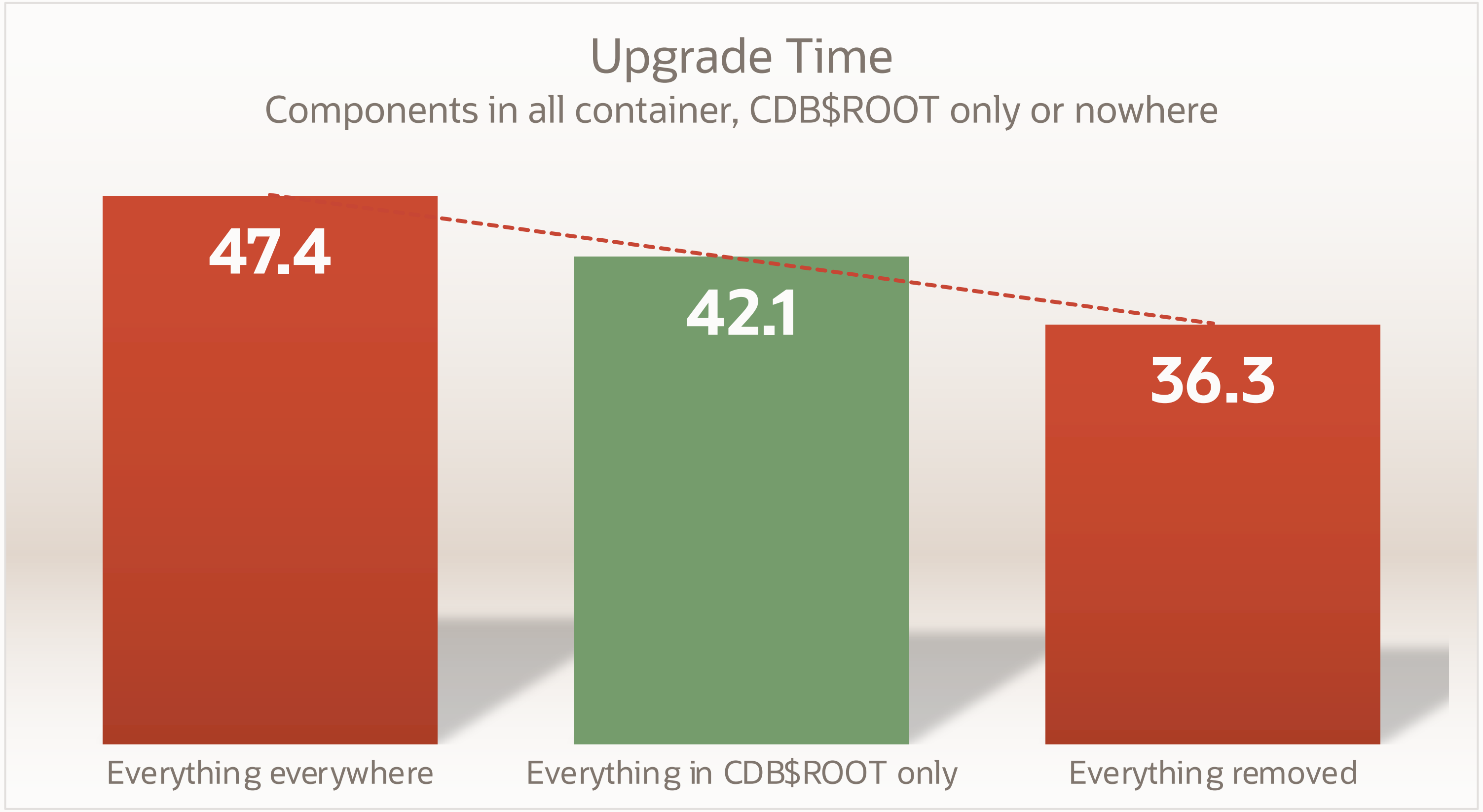


Components | CDB$ROOT vs PDB

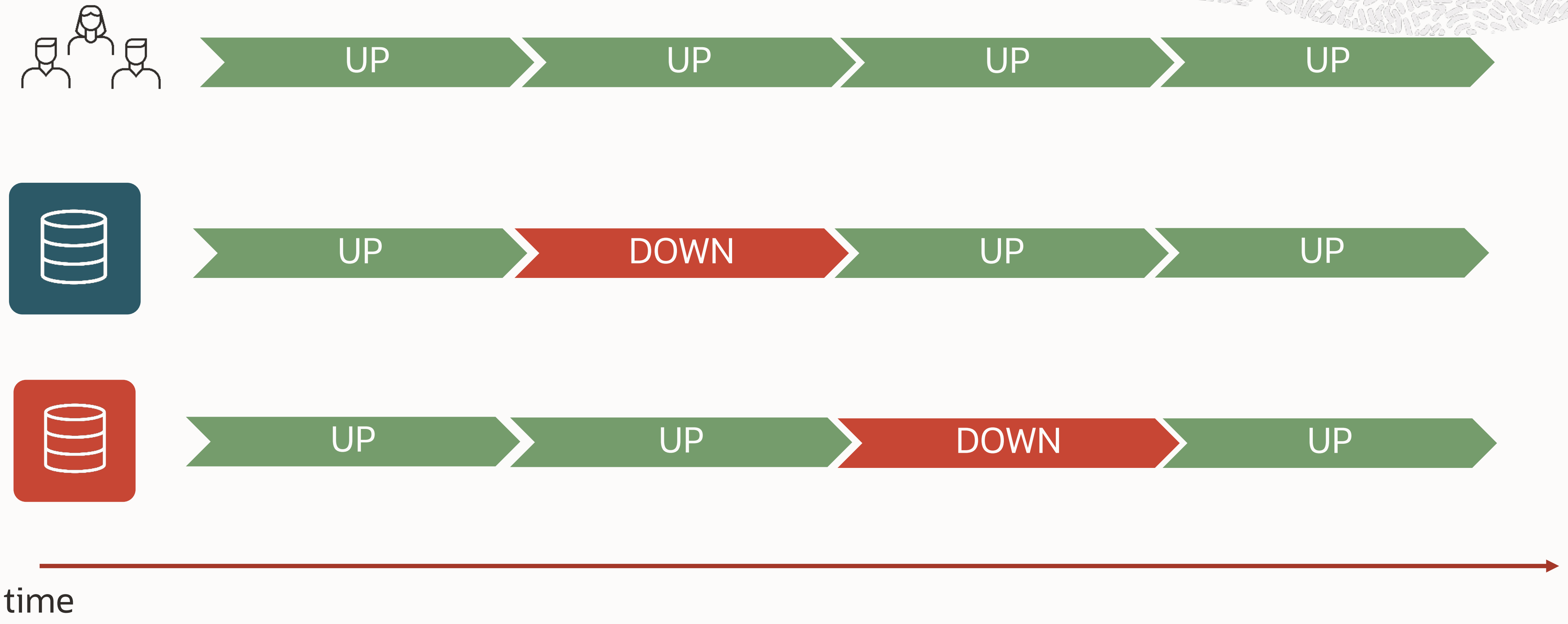
This may be a solution



Components | Compromise

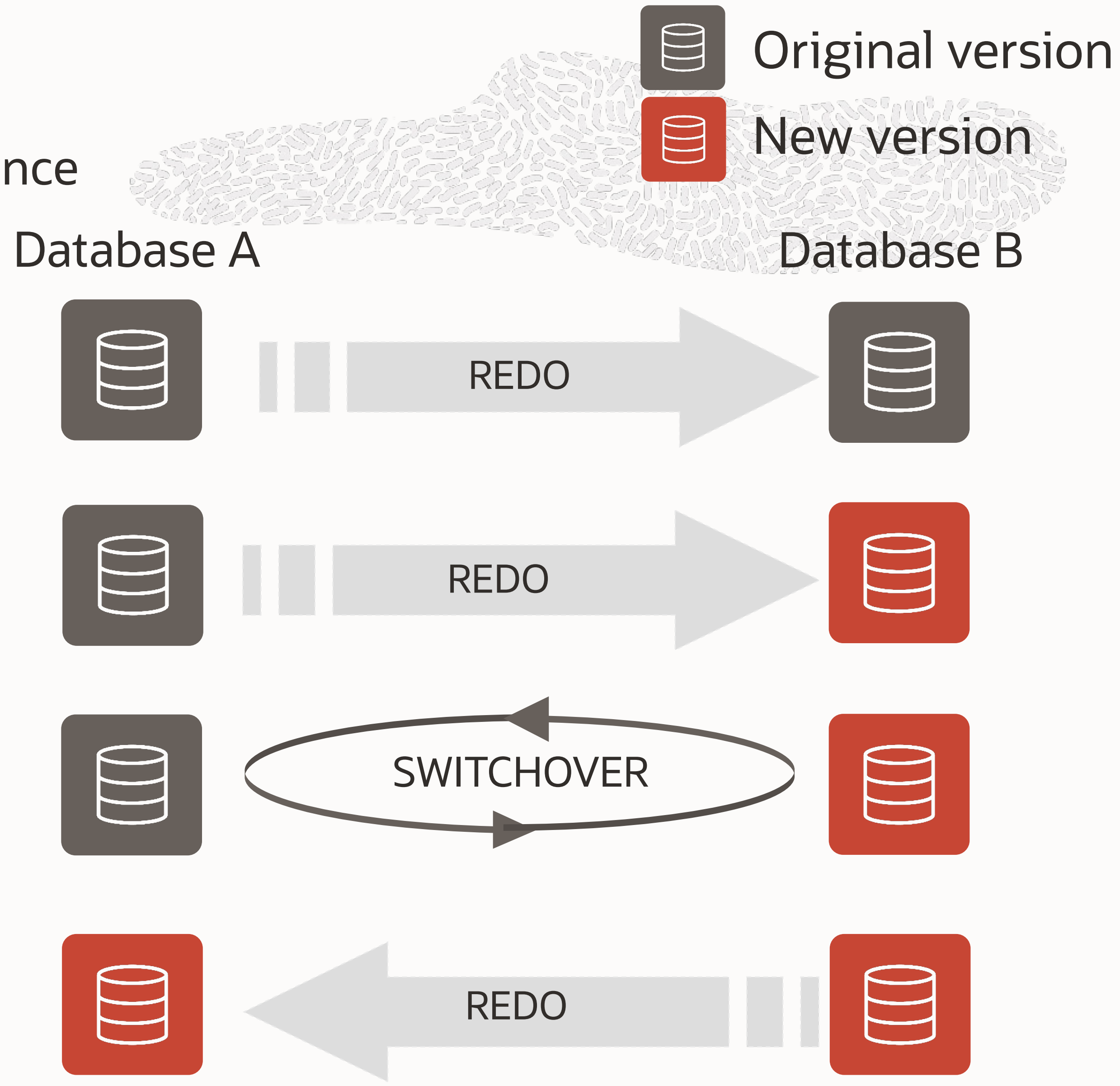


Rolling Upgrade and Application Downtime



Standby-First Patch Apply

General Process for Database Rolling Maintenance



Install new version in separate Oracle homes and defer transport

Patch or perform other maintenance on B then synchronize with production

Switch production to B, outage limited to the time needed to switch roles

Upgrade A via redo stream and synchronize

Oracle Patch Assurance - Data Guard Standby-First Patch Apply (Doc ID 1265700.1)

Solutions for Database Rolling Maintenance and Upgrades

Manual	DBMS_ROLLING	GoldenGate
Part of Enterprise Edition	Requires Active Data Guard	Requires GoldenGate
Source >= 11.1.0.7	Source >= 12.1.0.2	Source >= 11.2.0.4 (for OCI GG)
Manual approach	Automated	Manual approach
Limited feature support	Comprehensive feature support	Best feature support
		Fallback mechanism

Using SQL Apply to Upgrade the Oracle Database
https://docs.oracle.com/en/database/oracle/oracle-database/19/sbydb/using-sql-apply-to-perform-rolling-upgrade.html

Using DBMS_ROLLING to Perform a Rolling Upgrade
https://docs.oracle.com/en/database/oracle/oracle-database/19/sbydb/using-DBMS_ROLLING-to-perform-rolling-upgrade.html

Overview of Steps for Upgrading Oracle Database Using Oracle GoldenGate
https://docs.oracle.com/en/database/oracle/oracle-database/19/upgrd/converting-databases-upgrades.html#GUID-8E029631-8265-497C-983B-B8A4ACD47B98

Rolling Upgrade | DBMS_ROLLING

Use a logical standby database to upgrade with very little downtime.

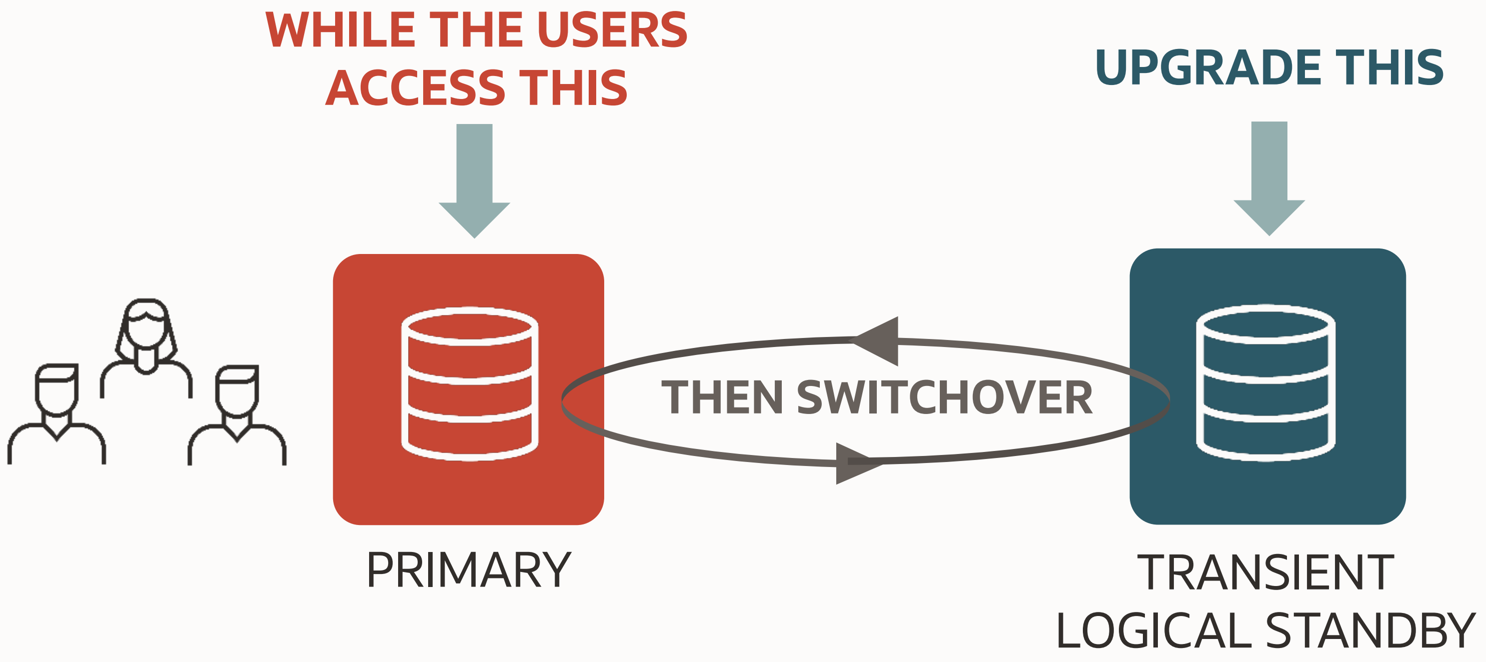
The only downtime is as little as it takes to perform a switchover.

Pro tip: Also useful for other maintenance activities

ADG

Active Data Guard Rolling Maintenance and Upgrades

Using DBMS_ROLLING package



- Use a transient logical standby database to upgrade with very little downtime.
- The only downtime is as little as it takes to perform a switchover.

DBMS_ROLLING points of attention

Do not create the logical standby on the same server as the primary database

Supplemental logging is enabled automatically which introduces an overhead and increases the amount of redo generated

When supplemental logging is enabled all DML cursors are invalidated

Not all data types and partitioning types are supported

For optimal performance all tables should have primary keys or unique keys

ADG

Important DBMS_ROLLING milestones

The driver is the SOURCE database!

SOURCE VERSION

12.1	• First version of DBMS_ROLLING for upgrades from 12.1 to higher versions
12.2	• Integration with the Data Guard broker • Services, roles changes, and instances are managed automatically by Clusterware • FAN events for Clusterware-backed databases • Support for Identity columns
21c	• FAN events without Clusterware • Support for JSON datatype

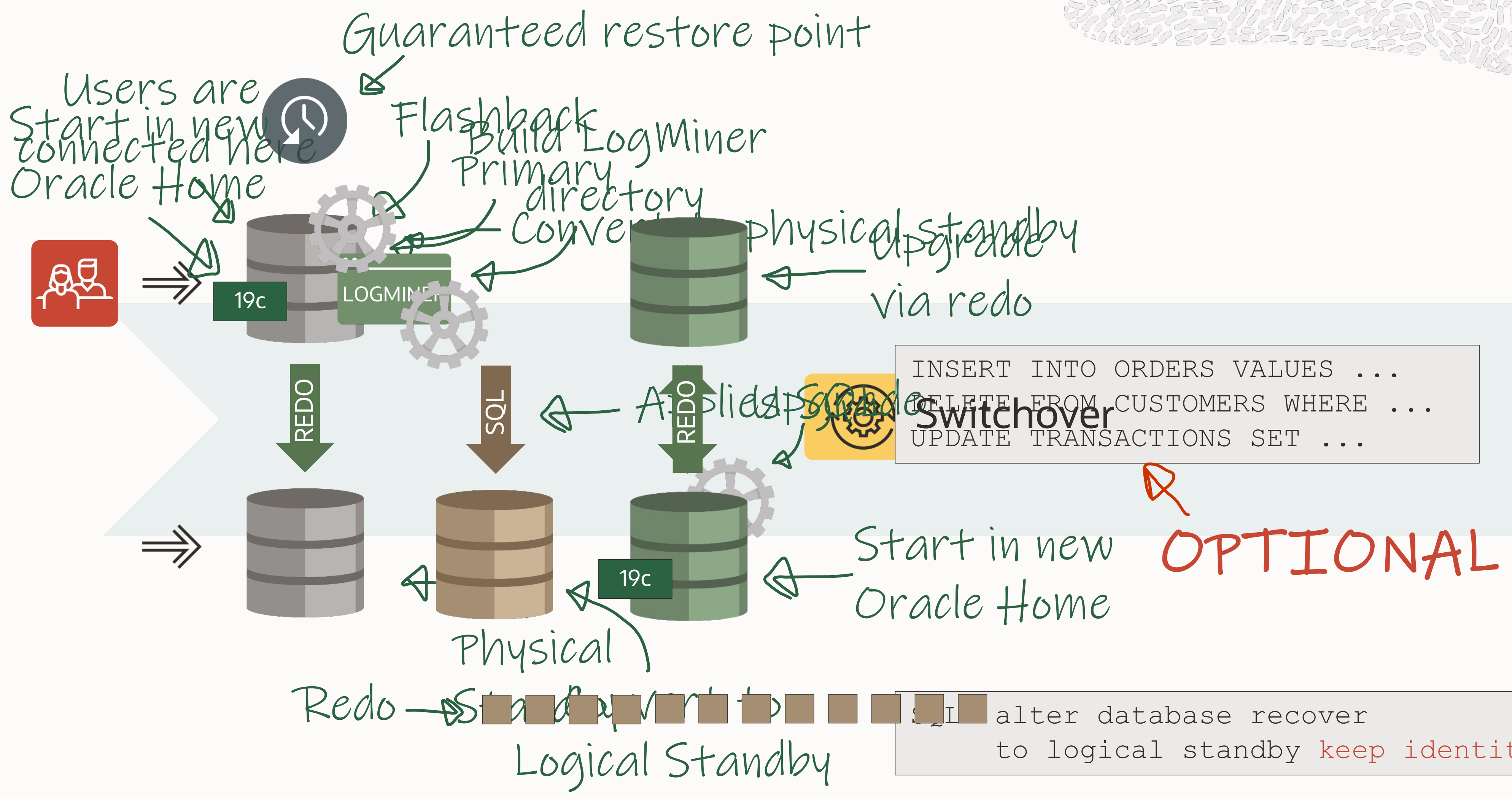
ADG

DBMS_ROLLING and client failover

DBMS_ROLLING.SWITCHOVER	Broker + OCW	Broker Only
12.1	Broker Not supported	Broker Not supported
12.2	FAN events	No FAN events
19c	FAN events	No FAN events
21c	FAN events	FAN events

AC/TAC support is in the roadmap

Rolling Upgrade | Concept



```
INSERT INTO ORDERS VALUES ...
DELETE FROM CUSTOMERS WHERE ...
UPDATE TRANSACTIONS SET ...
```

```
SQL> alter database recover
     to logical standby keep identity;
```

Rolling Upgrade | DBMS_ROLLING

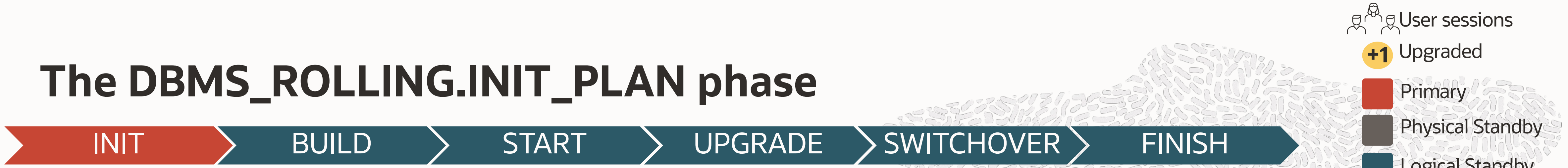
6 SIMPLE STEPS

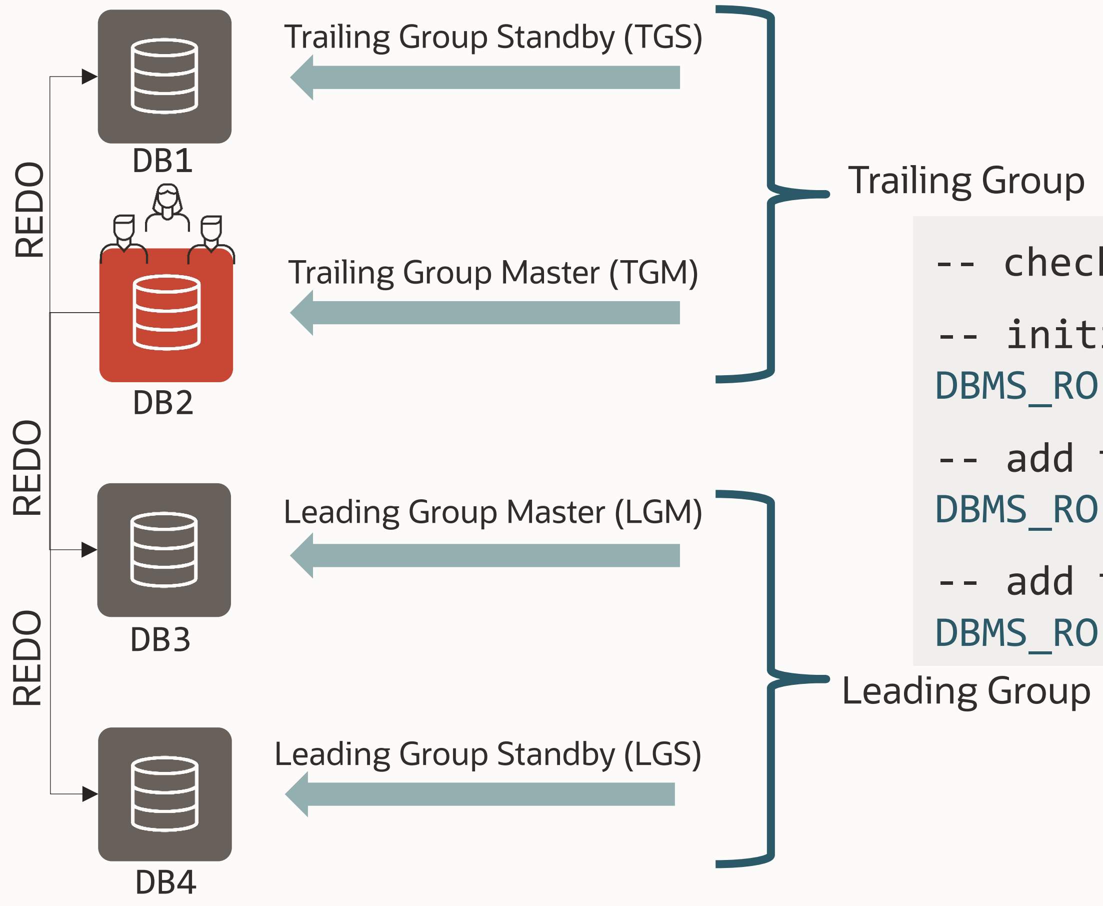
```
SQL> exec dbms_rolling.init_plan;
SQL> exec dbms_rolling.build_plan;
SQL> exec dbms_rolling.start_plan;
```

Upgrade database

```
SQL> exec dbms_rolling.switchover;
SQL> exec dbms_rolling.finish_plan;
```

The DBMS_ROLLING.INIT_PLAN phase




Trailing Group Standby (TGS)
DB1
Trailing Group Master (TGM)
DB2
Leading Group Master (LGM)
DB3
Leading Group Standby (LGS)
DB4
REDO
REDO
REDO
Trailing Group
Leading Group

```
-- check DBA_ROLLING_UNSUPPORTED for incompatible data types
-- initialize the plan and set the future primary
DBMS_ROLLING.INIT_PLAN(future_primary=>'DB3');
-- add the required standbys to the TRAILING GROUP
DBMS_ROLLING.SET_PARAMETER('DB1','MEMBER','TRAILING');
-- add the required standbys to the LEADING GROUP
DBMS_ROLLING.SET_PARAMETER('DB4','MEMBER',LEADING');
```

The DBMS_ROLLING parameters

INIT | BUILD | START | UPGRADE | SWITCHOVER | FINISH

ACTIVE_SESSIONS_TIMEOUT
ACTIVE_SESSIONS_WAIT
BACKUP_CONTROLFILE
DGBROKER
DICTIONARY_LOAD_TIMEOUT
DICTIONARY_LOAD_WAIT
DICTIONARY_PLS_WAIT_INIT
DICTIONARY_PLS_WAIT_TIMEOUT
EVENT_RECORDS
FAILOVER
GRP_PREFIX
IGNORE_BUILD_WARNINGS
IGNORE_LAST_ERROR
LAD_ENABLED_TIMEOUT
LOG_LEVEL

MEMBER
READY_LGM_LAG_TIME
READY_LGM_LAG_TIMEOUT
READY_LGM_LAG_WAIT
SWITCH_LGM_LAG_TIME
SWITCH_LGM_LAG_TIMEOUT
SWITCH_LGM_LAG_WAIT
SWITCH_LGS_LAG_TIME
SWITCH_LGS_LAG_TIMEOUT
SWITCH_LGS_LAG_WAIT
UPDATED_LGS_TIMEOUT
UPDATED_LGS_WAIT
UPDATED_TGS_TIMEOUT
UPDATED_TGS_WAIT

The DBMS_ROLLING parameters

INIT › BUILD › START › UPGRADE › SWITCHOVER › FINISH

Example:

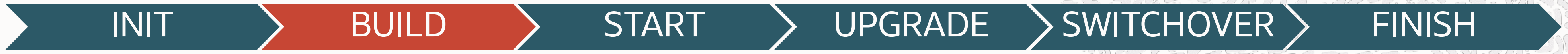
```
-- Activate full logging
exec DBMS_ROLLING.SET_PARAMETER (scope=>null,  name=>'LOG_LEVEL', value=>'FULL');


-- Wait for the SQL Apply Lag to go below 1 minute before initiating the switchover
exec DBMS_ROLLING.SET_PARAMETER('SWITCH_LGM_LAG_WAIT', '1');
exec DBMS_ROLLING.SET_PARAMETER('SWITCH_LGM_LAG_TIME', '60');
```

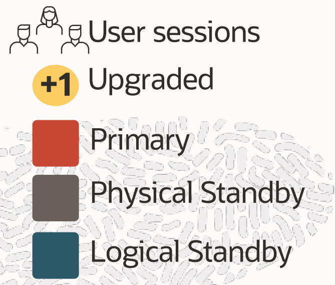
Final touches before starting

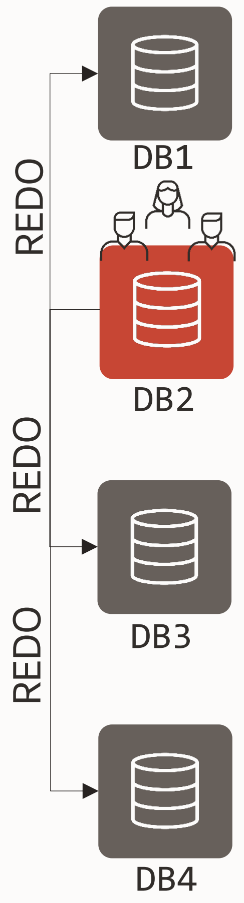
INIT > BUILD > START > UPGRADE > SWITCHOVER > FINISH

```
$ # The standby must be mounted
$ srvctl stop database -d DB3
$ srvctl start database -d DB3 -o mount

SQL> -- The PDBs must be open
SQL> alter pluggable database all open;

DGMGRL> # no FSFO or MaxProtection
DGMGRL> disable fast_start failover
DGMGRL> edit configuration set protection mode as MaxAvailability;
```

The DBMS_ROLLING.BUILD_PLAN phase







```
-- build the plan
DBMS_ROLLING.BUILD_PLAN();

-- check for any errors or warnings
SELECT * FROM DBA_ROLLING_EVENTS;

-- review the plan
SELECT * FROM DBA_ROLLING_PLAN ORDER BY INSTID;
```

Rolling Upgrade | DBMS_ROLLING

```
...
Get current redo branch of the primary database
Wait until recovery is active on the primary's redo
branch
Reduce to a single instance if database is a RAC
Verify only a single instance is active if future
primary is RAC
Stop media recovery
Execute dbms_logstdby.build
Convert into a transient logical standby
Open database including instance-peers if RAC
Verify logical standby is open read/write
Get redo branch of transient logical standby
Get reset scn of transient logical redo branch
Configure logical standby parameters
Start logical standby apply
```

86+ INSTRUCTIONS OR CHECKS

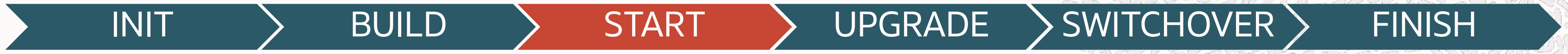
```
Stop logical standby apply
Start logical standby apply
Wait until apply lag has fallen below 600 seconds
Notify Data Guard broker that switchover to logical
standby database is starting
Log post-switchover instructions to events table
Switch database to a logical standby
Notify Data Guard broker that switchover to logical
standby database has completed
Wait until end-of-redo has been applied
...
```

The DBMS_ROLLING.BUILD_PLAN phase

ADG

```
 1 START     Notify Data Guard broker that DBMS_ROLLING has started
 2 START     Notify Data Guard broker that DBMS_ROLLING has started
 3 START     Verify database is a primary
 4 START     Verify MAXIMUM PROTECTION is disabled
 5 START     Verify database is a physical standby
 6 START     Verify physical standby is mounted
 7 START     Verify future primary is configured with standby redo logs
 8 START     Verify server parameter file exists and is modifiable
 9 START     Verify server parameter file exists and is modifiable
10 START     Verify Data Guard broker configuartion is enabled
11 START     Verify Data Guard broker configuartion is enabled
12 START     Verify Fast-Start Failover is disabled
13 START     Verify Fast-Start Failover is disabled
14 START     Verify fast recovery area is configured
15 START     Verify available flashback restore points
16 START     Verify fast recovery area is configured
17 START     Verify available flashback restore points
18 START     Stop media recovery
19 START     Drop guaranteed restore point DBMSRU_INITIAL
20 START     Create guaranteed restore point DBMSRU_INITIAL
21 START     Drop guaranteed restore point DBMSRU_INITIAL
22 START     Create guaranteed restore point DBMSRU_INITIAL
23 START     Start media recovery
24 START     Verify media recovery is running
25 START     Verify user_dump_dest has been specified
26 START     Backup control file to rolling_change_backup.f
27 START     Verify user_dump_dest has been specified
28 START     Backup control file to rolling_change_backup.f
29 START     Get current supplemental logging on the primary database
30 START     Get current redo branch of the primary database
31 START     Wait until recovery is active on the primary's redo branch
32 START     Reduce to a single instance if database is a RAC
33 START     Verify only a single instance is active if future primary is RAC
34 START     Stop media recovery
35 START     Execute dbms_logstdby.build
36 START     Convert into a transient logical standby
37 START     Open database including instance-peers if RAC
38 START     Verify logical standby is open read/write
39 START     Get redo branch of transient logical standby
40 START     Get reset scn of transient logical redo branch
41 START     Configure logical standby parameters
42 START     Start logical standby apply
43 START     Enable compatibility advance despite presence of GRPs
44 START     Log pre-switchover instructions to events table
45 START     Record start of user upgrade of DB3
46 SWITCH    Verify database is in OPENRW mode
47 SWITCH    Record completion of user upgrade of DB3
48 SWITCH    Scan LADs for presence of DB2 destination
49 SWITCH    Test if DB2 is reachable using configured TNS service
50 SWITCH    Call Data Guard broker to enable redo transport to DB3
51 SWITCH    Archive all current online redo logs
52 SWITCH    Archive all current online redo logs
53 SWITCH    Stop logical standby apply
54 SWITCH    Start logical standby apply
55 SWITCH    Wait until apply lag has fallen below 600 seconds
56 SWITCH    Notify Data Guard broker that switchover to logical standby database is starting
57 SWITCH    Log post-switchover instructions to events table
58 SWITCH    Switch database to a logical standby
59 SWITCH    Notify Data Guard broker that switchover to logical standby database has completed
60 SWITCH    Wait until end-of-redo has been applied
61 SWITCH    Archive all current online redo logs
62 SWITCH    Notify Data Guard broker that switchover to primary is starting
63 SWITCH    Switch database to a primary
64 SWITCH    Notify Data Guard broker that switchover to primary has completed
65 SWITCH    Enable compatibility advance despite presence of GRPs
66 SWITCH    Synchronize plan with new primary
67 FINISH    Reduce to a single instance for FINISH
68 FINISH    Verify only a single instance is active
69 FINISH    Verify database is mounted
70 FINISH    Flashback database
71 FINISH    Convert into a physical standby
72 FINISH    Verify database is open
73 FINISH    Save the DBID of the new primary
74 FINISH    Save the logminer session start scn
75 FINISH    Wait until transient logical redo branch has been registered
76 FINISH    Start media recovery
77 FINISH    Wait until apply/recovery has started on the transient branch
78 FINISH    Wait until upgrade redo has been fully recovered
79 FINISH    Prevent compatibility advance if GRPs are present
80 FINISH    Prevent compatibility advance if GRPs are present
81 FINISH    Drop guaranteed restore point DBMSRU_INITIAL
82 FINISH    Drop guaranteed restore point DBMSRU_INITIAL
83 FINISH    Purge logical standby metadata from database if necessary
84 FINISH    Notify Data Guard broker that DBMS_ROLLING has finished
85 FINISH    Notify Data Guard broker that DBMS_ROLLING has finished
86 FINISH    Restore Supplemental Logging
```

The DBMS_ROLLING.START phase



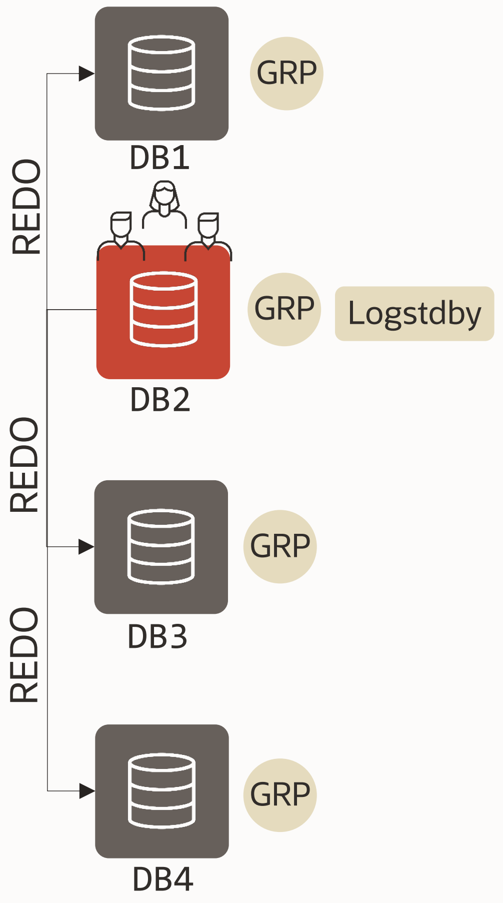
User sessions

+1 Upgraded

🟥 Primary

⬛ Physical Standby

🟦 Logical Standby

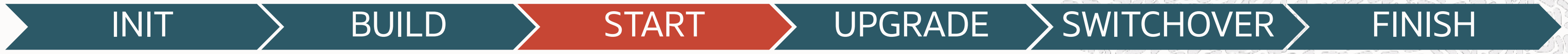


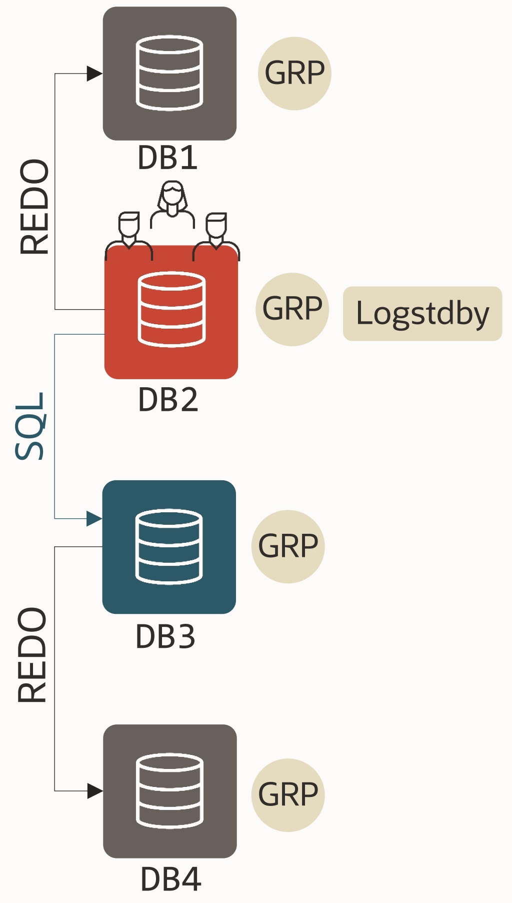
```
-- start the plan
DBMS_ROLLING.START_PLAN();
```

- Creates the Guaranteed Restore Point (GRP)
- Builds the logical standby metadata (`dbms_logstdby.build`)

User sessions
+1 Upgraded
🟥 Primary
⬛ Physical Standby
🟦 Logical Standby

The DBMS_ROLLING.START phase

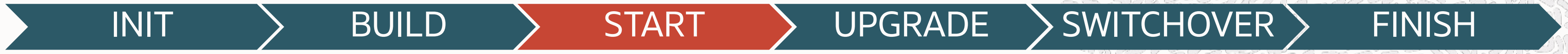




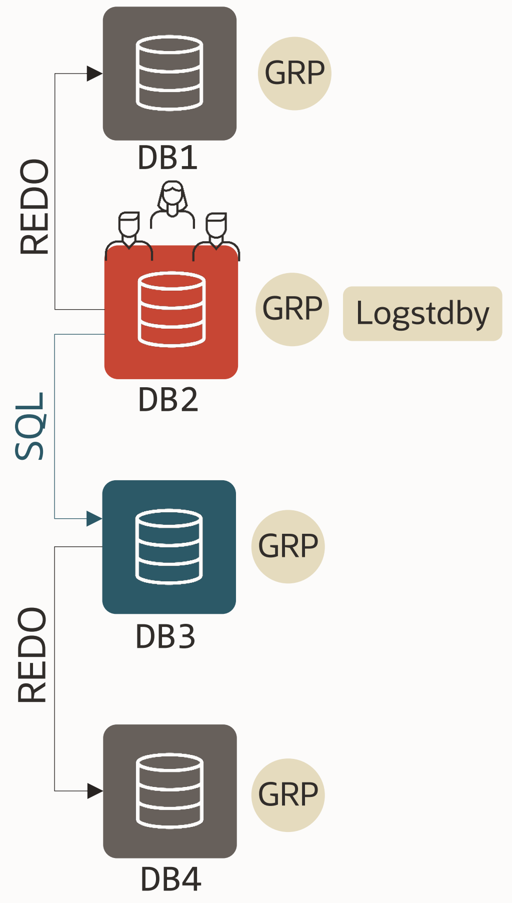
```
-- start the plan
DBMS_ROLLING.START_PLAN();
```

- Creates the Guaranteed Restore Point (GRP)
- Builds the LogMiner directory (`dbms_logstdby.build`)
- Converts the LGM to Logical Standby
- Starts SQL Apply
- With a configuration composed of 4 databases,
  the LGM and TGM are still protected by a physical standby

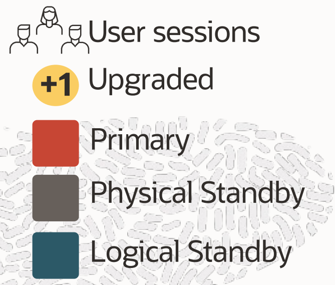
The DBMS_ROLLING.START phase



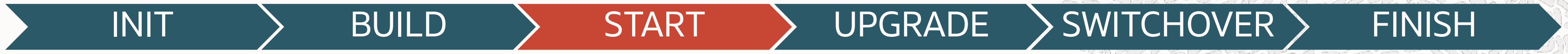
👥	User sessions
+1	Upgraded
🟥	Primary
⬛	Physical Standby
🟦	Logical Standby

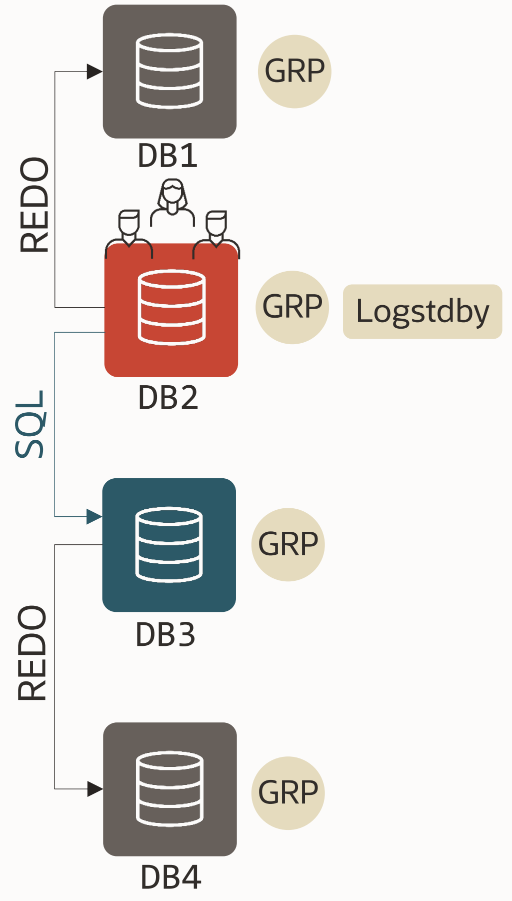


```
DGMGRL> show configuration;

Configuration - geneva

  Protection Mode: MaxAvailability
  Members:
  DB1   - Primary database
    DB3 - Physical standby database
      Warning: ORA-16854: apply lag could not be
determined

Fast-Start Failover: DISABLED

Configuration Status:
    ROLLING DATABASE MAINTENANCE IN PROGRESS
```



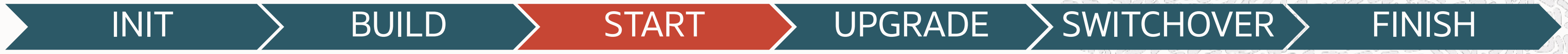
The DBMS_ROLLING.START phase

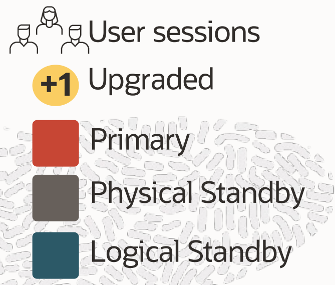


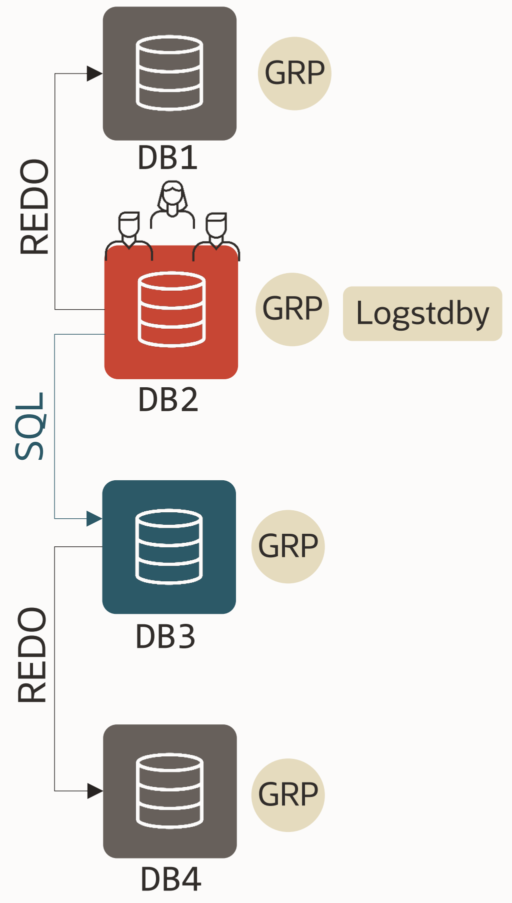


```
DGMGRL> show database DB3
...
  Role:                 PHYSICAL STANDBY
  Intended State:       APPLY-ON
  Transport Lag:        0 seconds (computed 0 seconds
ago)
  Apply Lag:            3 minutes 18 seconds (computed 0
seconds ago)
...
Database Warning(s):
    ORA-16866: database converted to transient logical
standby database for rolling database maintenance

Database Status:
WARNING
```

The DBMS_ROLLING.START phase

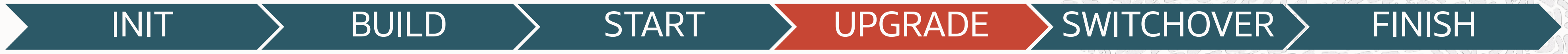


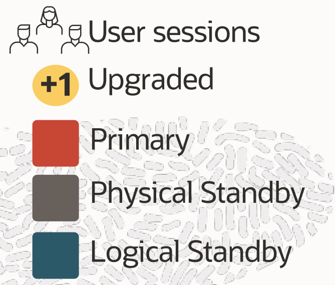


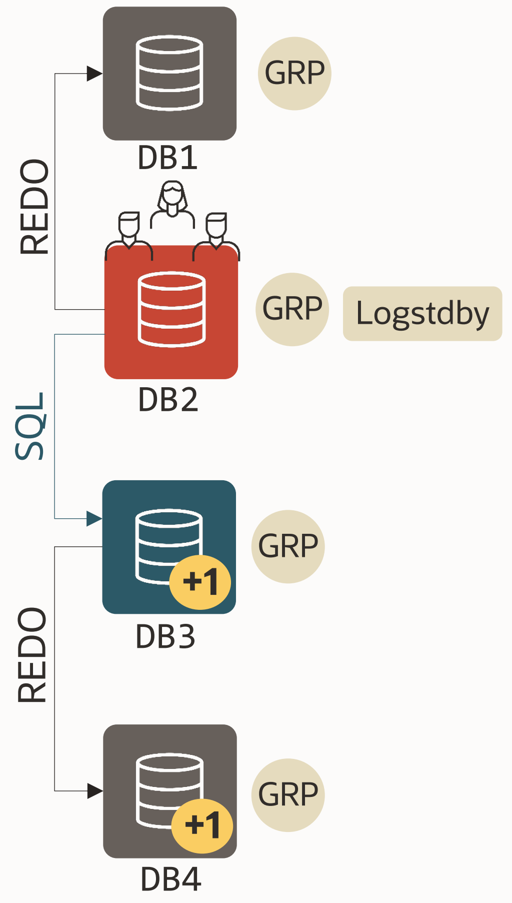


```
-- check the status of the SQL apply:
SQL> select * from V$LOGSTDBY_PROGRESS;

-- use SQL apply commands if you need
SQL> alter database start logical standby apply immediate;

-- check for logical standby error messages
SQL> select * from DBA_LOGSTDBY_EVENTS
2>    order by event_timestamp;

22-NOV-21 06.41.12  DML on "AUDSYS"."AUD$UNIFIED"
                    ORA-16129: unsupported DML encountered
22-NOV-21 06.41.13  truncate table wri$_adv_addm_pdbs
                    ORA-16247: DDL skipped on internal schema
```

The Upgrade/Maintenance phase





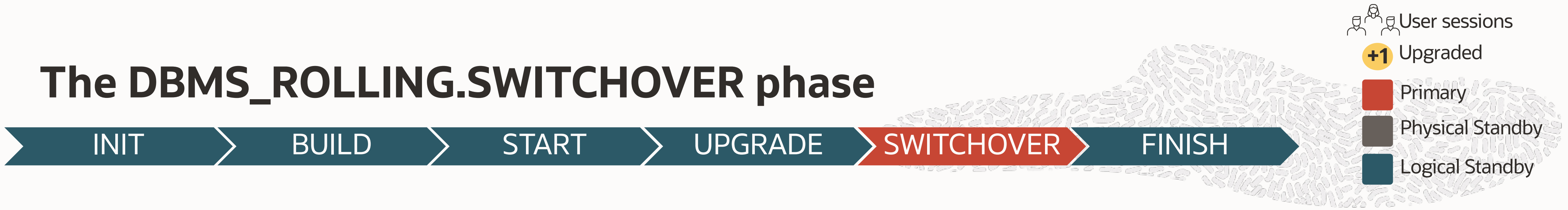


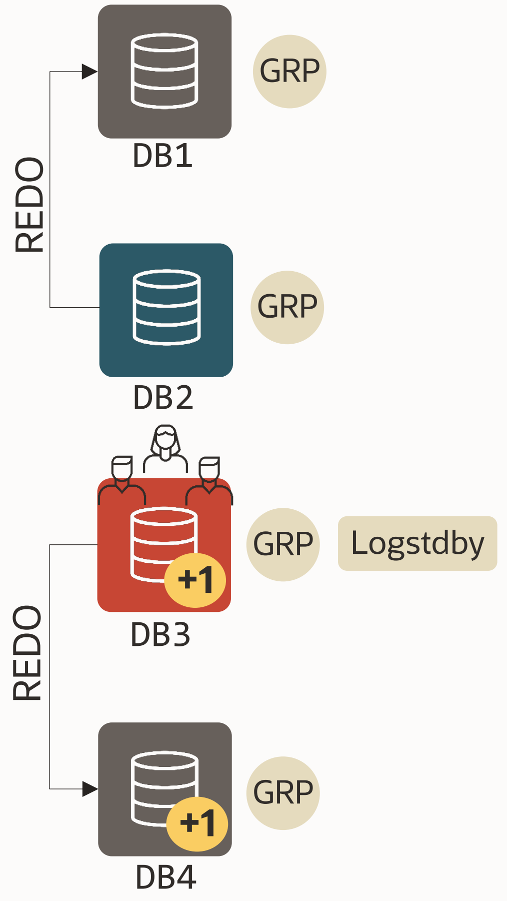
- Do the maintenance on the Leading Group Master

```
-- e.g. upgrade to a major version with AutoUpgrade
$ java -jar autoupgrade.jar -config CDB1.cfg -mode deploy
```

- This is out of DBMS_ROLLING scope (it is a manual step)
- Don't forget to align the Leading Group Standbys if necessary
- Use it for any major maintenance that requires longer downtimes (change of physical layout, structure changes, offline operations)

The DBMS_ROLLING.SWITCHOVER phase





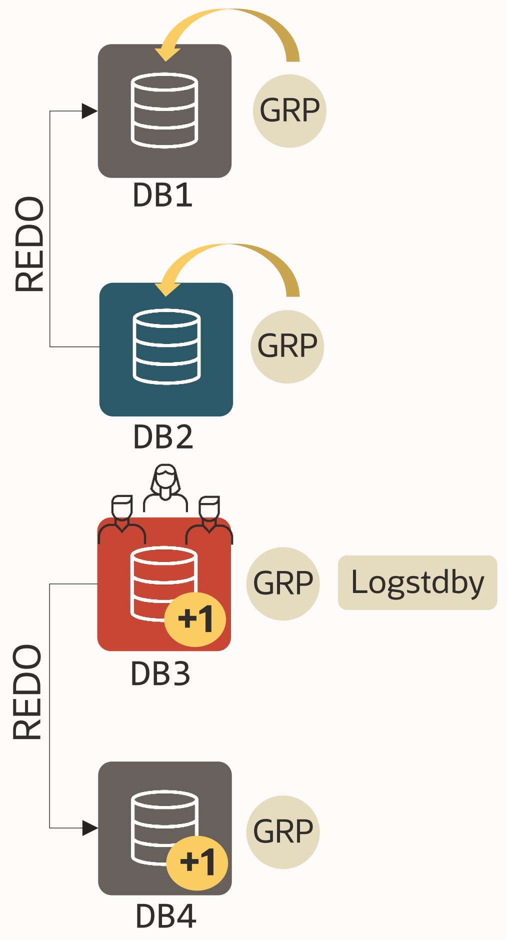
```
-- switchover to the upgraded database
DBMS_ROLLING.SWITCHOVER()
```

- Depending on the source version and HA configuration, the old connections get FAN notifications and drain automatically
- New connections go to the new primary. Application downtime is minimal.

The DBMS_ROLLING.SWITCHOVER phase

INIT › BUILD › START › UPGRADE › SWITCHOVER › FINISH

User sessions
+1 Upgraded
🟥 Primary
⬛ Physical Standby
🟦 Logical Standby

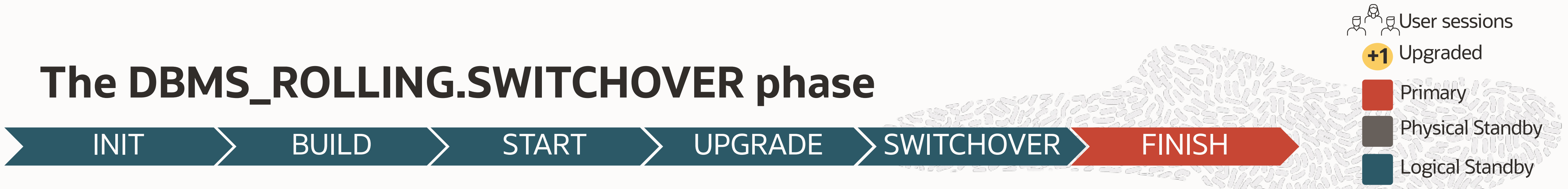


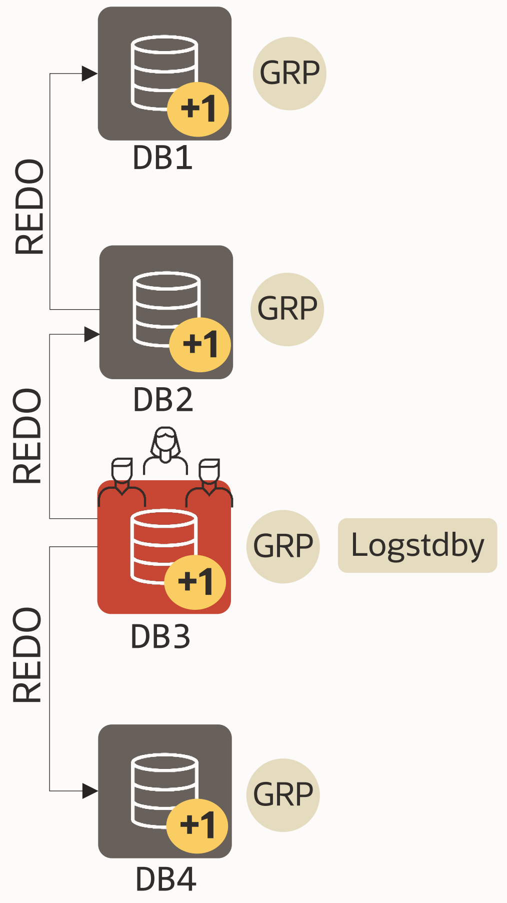
- Start the Trailing Group members with the new binaries (manual)

```
-- run the final part of the plan
DBMS_ROLLING.FINISH_PLAN()
```

- Flashes back the Trailing Group Master and Standby to the GRP

The DBMS_ROLLING.SWITCHOVER phase





- Start the Trailing Group members with the new binaries (manual)

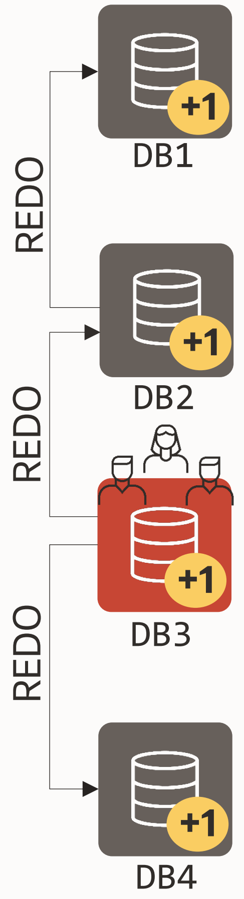
```
-- run the final part of the plan
DBMS_ROLLING.FINISH_PLAN()
```

- Flashes back the Trailing Group Master and Standby to the GRP
- Converts the Trailing Group Master to a physical standby
- Starts redo apply and catches up with the primary
- Drops the guaranteed restore points and logical standby metadata

The DBMS_ROLLING.SWITCHOVER phase

INIT > BUILD > START > UPGRADE > SWITCHOVER > FINISH

User sessions
+1 Upgraded
🟥 Primary
⬛ Physical Standby
🟦 Logical Standby



```
-- destroy the plan to clean up everything
DBMS_ROLLING.DESTROY_PLAN()
```

Rolling Upgrade | DBMS_ROLLING

6 SIMPLE STEPS

```
SQL> exec dbms_rolling.init_plan;
SQL> exec dbms_rolling.build_plan;
SQL> exec dbms_rolling.start_plan;
```

Upgrade database

```
SQL> exec dbms_rolling.switchover;
SQL> exec dbms_rolling.finish_plan;
```

ADG

DBMS_ROLLING catalog views

Evaluate	`DBA_ROLLING_UNSUPPORTED`	Check here for unsupported data types!
Initialize	`DBA_ROLLING_PARAMETERS`	Get the current parameters before building
Build	`DBA_ROLLING_DATABASES`	
	`DBA_ROLLING_PLAN`	Verify the plan before and during the execution
Monitor	`DBA_ROLLING_EVENTS`	Warning and errors are visible here
	`DBA_ROLLING_STATISTICS`	
	`DBA_ROLLING_STATUS`	

DBMS_ROLLING – Read More

Using DBMS_ROLLING to Perform a Rolling Upgrade
https://docs.oracle.com/en/database/oracle/oracle-database/19/sbydb/using-DBMS_ROLLING-to-perform-rolling-upgrade.html

DBMS_ROLLING - PL/SQL Packages and Types Reference
https://docs.oracle.com/en/database/oracle/oracle-database/19/arpls/DBMS_ROLLING.html#GUID-097F1B39-E623-43B5-BA30-DF377BFE05CF

Automated Database Upgrades using Oracle Active Data Guard and DBMS_ROLLING
https://www.oracle.com/technetwork/database/availability/database-upgrade-dbms-rolling-4126957.pdf

Oracle Database Rolling Upgrades (without DBMS_ROLLING)
https://www.oracle.com/technetwork/database/availability/database-rolling-upgrade-3206539.pdf

DBMS_ROLLING – Read More

MOS Notes:

- Transient Rolling Upgrade Using DBMS_ROLLING - Beginners Guide
- Rolling upgrade using DBMS_ROLLING - Complete Reference (Doc ID 2086512.1)
- MAA Whitepaper: SQL Apply Best Practices (Doc ID 1672310.1)
- Step by Step How to Do Swithcover/Failover on Logical Standby Environment (Doc ID 2535950.1)
- How To Skip A Complete Schema From Application on Logical Standby Database (Doc ID 741325.1)
- How to monitor the progress of the logical standby (Doc ID 1296954.1)
- How To Reduce The Performance Impact Of LogMiner Usage On A Production Database (Doc ID 1629300.1)
- Handling ORA-1403 ora-12801 on logical standby apply (Doc ID 1178284.1)
- Troubleshooting Example - Rolling Upgrade using DBMS_ROLLING (Doc ID 2535940.1)
- DBMS Rolling Upgrade Switchover Fails with ORA-45427: Logical Standby Redo Apply Process Was Not Running (Doc ID 2696017.1)
- SRDC - Collect Logical Standby Database Information (Doc ID 1910065.1)
- MRP fails with ORA-19906 after Flashback of Transient Logical Standby used for Rolling Upgrade (Doc ID 2069325.1)
- What Causes High Redo When Supplemental Logging is Enabled (Doc ID 1349037.1)

Questions & Answers

Thank you

Rolling Upgrades

Upgrade your DB with near Zero Downtime

Francisco Munoz Alvarez

Distinguished Product Manager

Oracle Database High Availability (HA), Scalability and Maximum Availability Architecture (MAA) Team

@fcomunoz

http://www.linkedin.com/in/franciscomunozalvarez

www.oraclemaa.com

ORACLE