ORACLE

AUTONOMOUS DATABASE

LEARNING LOUNGE

Build CI/CD pipelines with Autonomous Database and Oracle SQLcl Project

Autonomous Database Learning Lounge

Hosted by Marcos Arancibia

Autonomous Database Product Management

Agenda

Jeff Smith

Barry McGillin

Oracle SQlcl's Project command revolutionizes the process for maintaining your database application schemas and change management. This new workflow is a highly structured process, and you will learn all about the advantages it in this session.

Topics

- Introduction to SQLcl usage with Autonomous Database
- How to maintain your Oracle Database schemas in Git
- Automatically generate and manage your src and dist files
- Learn how to easily compare what's changed by keeping everything in easy-to-read SQL scripts
- Setup development and production pipelines to ensure consistent and reliable database schema installs and upgrades, backed up by rigorous and automated testing via CI/CD systems.

Q&A

- Product Managers will answer any questions

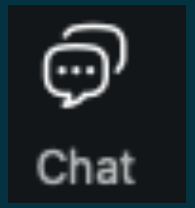
Before we begin...

This session is for you !!!

Ask your questions using **Q&A**

Product Managers are monitoring your questions

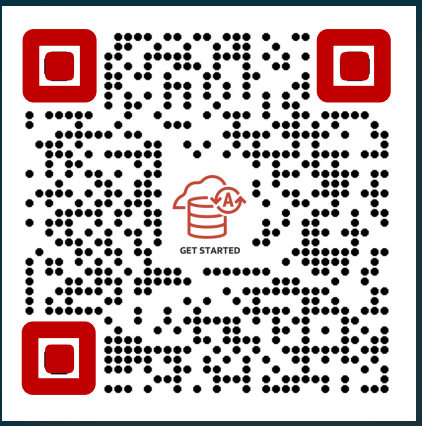
We will share links in **Chat**



The recording will be made available in a few days at
oracle.com/goto/adb-learning-lounge

Important links to bookmark

Links to get you started and to keep up to date with Autonomous Database



1 New Get Started page:
oracle.com/autonomous-database/get-started/

2 Join us:
LinkedIn
bit.ly/adb-linkedin-grp

X
@AutonomousDW

Bluesky
autonomousdb.bsky.social

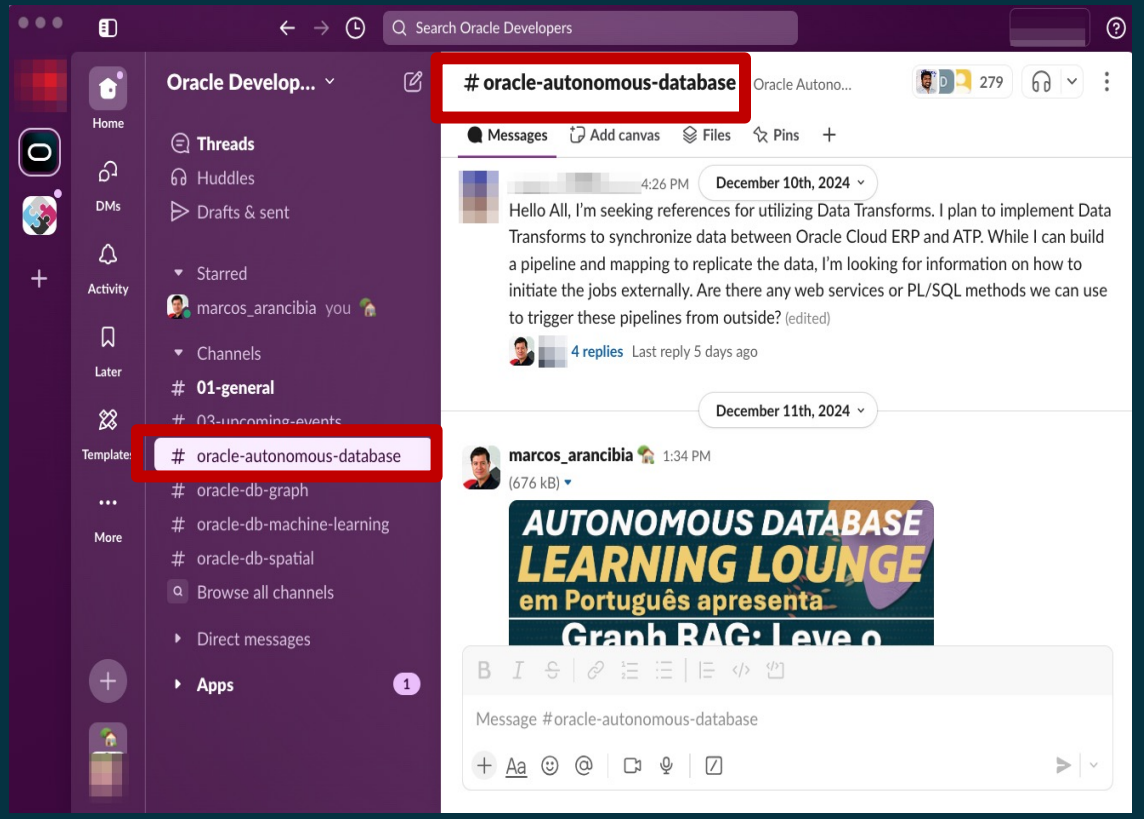
3 Got a question?
We are on stackoverflow
bit.ly/adb-stackoverflow

Join us on Developers Slack
(search #oracle-autonomous-database)
bit.ly/odevrel_slack

Join our External Slack

STEP 1: Join our Slack workspace at:
bit.ly/odevrel_slack

STEP 2: search for #oracle-autonomous-database at the top and click on the Channel



Speaker

Jeff Smith

ORACLE

Database CI/CD

App Deployment

Jeff Smith
Distinguished Product Manager
jeff.d.smith@oracle.com

About Me

Crashed my first production Oracle Database in 2000

Have been building and supporting database tools ever since!

I look after

- Oracle SQL Developer
  - Data Modeler
  - Web
  - SQLcl
  - Extension for VS Code
- Oracle REST Data Services (ORDS)

I also blog and annoy people online at… @thatJeffSmith

E-mail: jeff.d.smith@oracle.com

Why is CICD for Database Developers hard?

- No standards for the process
  - How is a database change added to git
  - How are upgrades from dev to test done
  - How are installs synchronized
- No consistent tooling
  - schema changes
  - deployment tools
  - data
- High risk deployments

Tooling for Database Change Management

Liquibase

- Changeset & Changelogs
- Audit trail in table

Flyway

- V1__create_table.sql where V1 is the version,
  and the double underscore separates the description
- Audit trail in table

Oracle SQLcl extended Liquibase support

Other Tools

- Scripts
- Pipelines

What is Oracle SQLcl

Seamless connectivity to Oracle Databases

Ease of use and lightweight install

- OTN, YUM, Docker, OCI, Homebrew, Chocolatey

Tools for Developers as standard

- Intelligent completions, file handling, formatting , error handling and diagnostics

Data Loading, formatting and unloading

DDL generation

Aliases for complex commands

Extended Liquibase support


```
sql sys/oracle@localhost:9521/freepdb1 as sysdba
42%   28 GB   24.3.1+ • 4
34) 21.3.3
enter your choice: 1


SQLcl: Release 24.3 Production on Wed Nov 06 20:56:08 2024

Copyright (c) 1982, 2024, Oracle.  All rights reserved.

login.sql found in the CWD. DB access is restricted for login.sql.
Adjust the SQLPATH to include the path to enable full functionality.

Connected to:
Oracle Database 23ai Free Release 23.0.0.0.0 - Develop, Learn, and Run for Free
Version 23.5.0.24.07

SYS@localhost:9521/freepdb1 >
viins ¦ 1:0 ¦ SYS ¦ localhost:9521/freepdb1 ¦ 24.3.1 ¦ /Users/bamcgill/development/
```

Introducing SQLcl Projects for Database Developers

Defines a prescribed process for database developers

Defines a recommended file structure

Defines tools to automate developer daily tasks

Generates releasable artifacts based atomic developer changes

Flexible to work with your existing projects

Tools of the trade

SQLcl!

Our modern Command Line Interface

```
SQL> help project
CICD SQLcl project extension

Subcommands:
  init|in
  Initialize a new project

  export|ex
  Export database objects to your repository

  config|cfg
  Display the project configuration to the screen

  gen-artifact|ga
  Generate a local artifact for the current project

  deploy|dp
  Deploy a release or release artifact to a specific database

  release|re
  Move the current set of work into a release state and start a new body of work

  verify|v
  Run a set of verification checks to ensure the validity of your project, and it is output

  stage|st
  Generate Liquibase changelogs and changesets for all src and custom SQL files
```

Command break-down

init – sets up our local directories and files, also is where you specify the schema or schemas that will be used by our project.

export – this uses a database connection to take all of your database objects

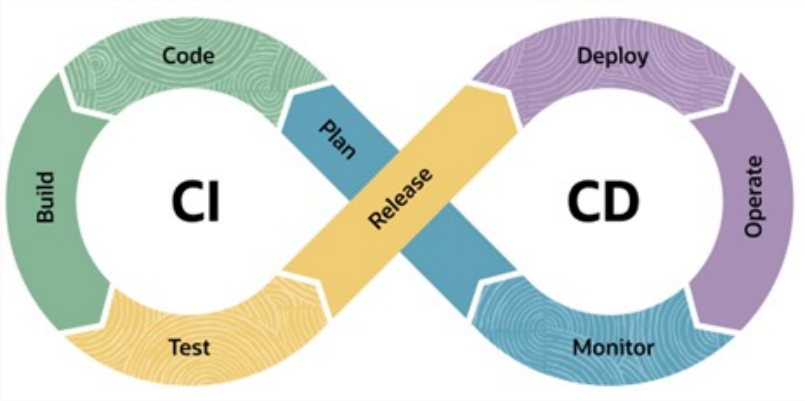
stage – this will populate the dist (distributable) directory tree with the SQL scripts to be used to deploy a ‘release’ or ‘version’ of our application schema

release – instead of working off of ‘master’ or ‘next,’ we’re going to have a release, such as 1.0.0 that will be used to deploy our project to the database (install), or say a 1.1.0 that’s used to upgrade an application from 1.0.0.

gen-artifact – this packages up the release dist files along with it’s install.sql script, and bundles it up in a Zip archive.

deploy – the simplest command, this takes an artifact (Zip), and deploys it to your database

SQLcl Projects
Workflow Tools



Project Creation

Database Export

Stage Changes

Artifact Generation

Release Deployment

Project Creation

- New Projects
- Existing projects

```
SQL> PROJECT  init
-name pname -makeroot
-schemas cicd
```

```
SQL> !git init --initial-
branch=main
SQL> !git add .
SQL> !git commit -m "chore:
initializing pname git
repository"
```

SQLcl Project Workflow

Database Export

Stage Changes

Artifact Generation

Release Deployment

SQLcl Project Workflow

Project Creation

Stage Changes

Artifact Generation

Release Deployment

Database Export

- Export source from database to files
- Control with filters in config.

```
SQL> PROJECT  export
```

```
SQL> project export
The current connection
localhost:1525/XEPDB1 HR will
be used for all operations
*** INDEXES ***
*** PROCEDURES ***
*** SEQUENCES ***
*** TABLES ***
```

SQLcl Project Workflow

Project Creation

Database Export

Stage Changes

- Create the actual changes to be applied.
- Control with filters in config.

```
SQL> PROJECT  stage -verbose
```

```
Starting execution of stage command
using the current branch
Created dir:
dist/releases/next/changes
Created dir:  dist/releases/next/code
```

Artifact Generation

Release Deployment

SQLcl Project Workflow

Project Creation

Database Export

Stage Changes

Artifact Generation

- Generate a releasable set of objects
- Generate an artifact for this set of installable objects

```
SQL> PROJECT  release -version 1.0
```

```
SQL> PROJECT  gen-artifact
```

Release Deployment

SQLcl Project Workflow

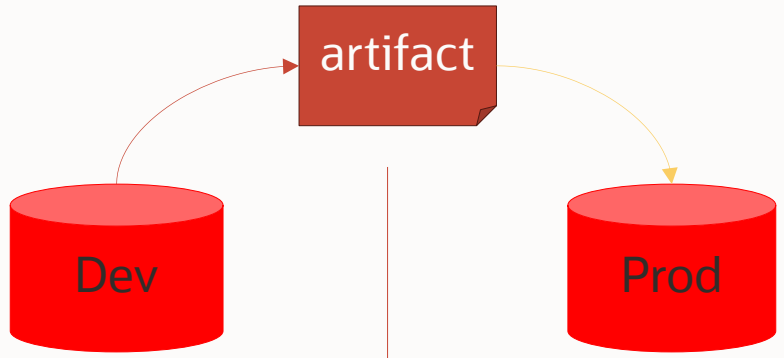
Project Creation

Database Export

Stage Changes

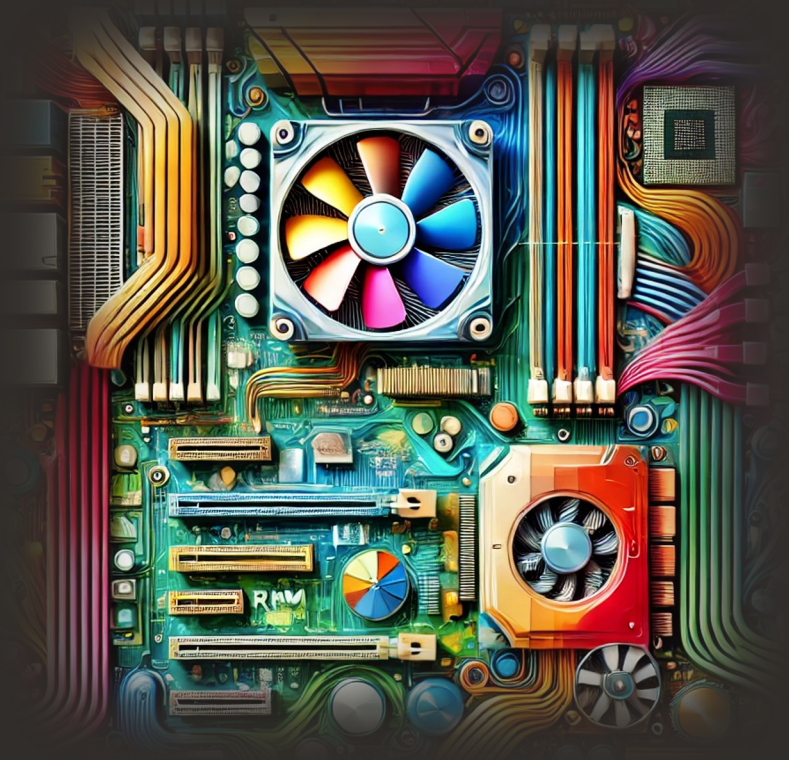
Artifact Generation

Release
Deployment



- Create objects
- Install artifact

```
SQL> PROJECT  deploy -file pname-1.0.zip
```



OK, that’s great. Now, how does it work?

Soft vs. hard objects

- Soft objects
  - DB objects like packages, views, triggers, etc.
  - Easier to work with, can always use “create or replace”
- Hard objects
  - DB objects like tables, ref constraints, indexes, constraints, etc.
  - More difficult to work with, must be “evolved” with care

Repo contains both ‘source’ *and* ‘distribution’

```
.
├── .dbtools
├── .git
├── dist
│   ├── install.sql
│   ├── releases
│   └── utils
└── src
    └── database
```

Repo contains both ‘source’ *and* ‘distribution’

State of database objects

- Helps track evolution of objects over time
- Useful for compiling code
  - Views, packages, triggers, etc.

Repo contains both ‘source’ *and* ‘distribution’

State of database objects

- Helps track evolution of objects over time
- Useful for compiling code
  - Views, packages, triggers, etc.

Immutable, ordered SQL for state in source

- Provides complete control over schema evolution
- Prevents breaking historical SQL

Repo contains both ‘source’ *and* ‘distribution’

src

```
create table emp (
  empno  number,
  fname  varchar2(50),
  lname  varchar2(50)
);
```

v1

dist

```
create table emp (
  empno  number,
  fname  varchar2(50),
  lname  varchar2(50)
);
```

Repo contains both ‘source’ *and* ‘distribution’


```
create table emp (
  empno  number,
  fname  varchar2(50),
  lname  varchar2(50),
  deptno number
);
```

v2


```
create table emp (
  empno  number,
  fname  varchar2(50),
  lname  varchar2(50)
);
```

```
alter table emp
  add deptno number;
```

Repo contains both ‘source’ *and* ‘distribution’


```
create table emp (
  empno  number,
  fname  varchar2(50),
  lname  varchar2(50),
  deptno number,
  name   varchar2(100)
);
```

v3


```
create table emp (
  empno  number,
  fname  varchar2(50),
  lname  varchar2(50)
);

alter table emp
  add deptno number;
```

```
alter table emp
  add name varchar2(100);

alter table emp
  drop column fname;

alter table emp
  drop column lname;
```

Repo contains both ‘source’ *and* ‘distribution’


```
create table emp (
  empno   number,
  fname   varchar2(50),
  lname   varchar2(50),
  deptno  number,
  name    varchar2(100)
);
```

v3

```
update emp
   set
  name = fname
         || ' '
         || lname;
```

```
create table emp (
  empno   number,
  fname   varchar2(50),
  lname   varchar2(50)
);

alter table emp
  add deptno number;
```

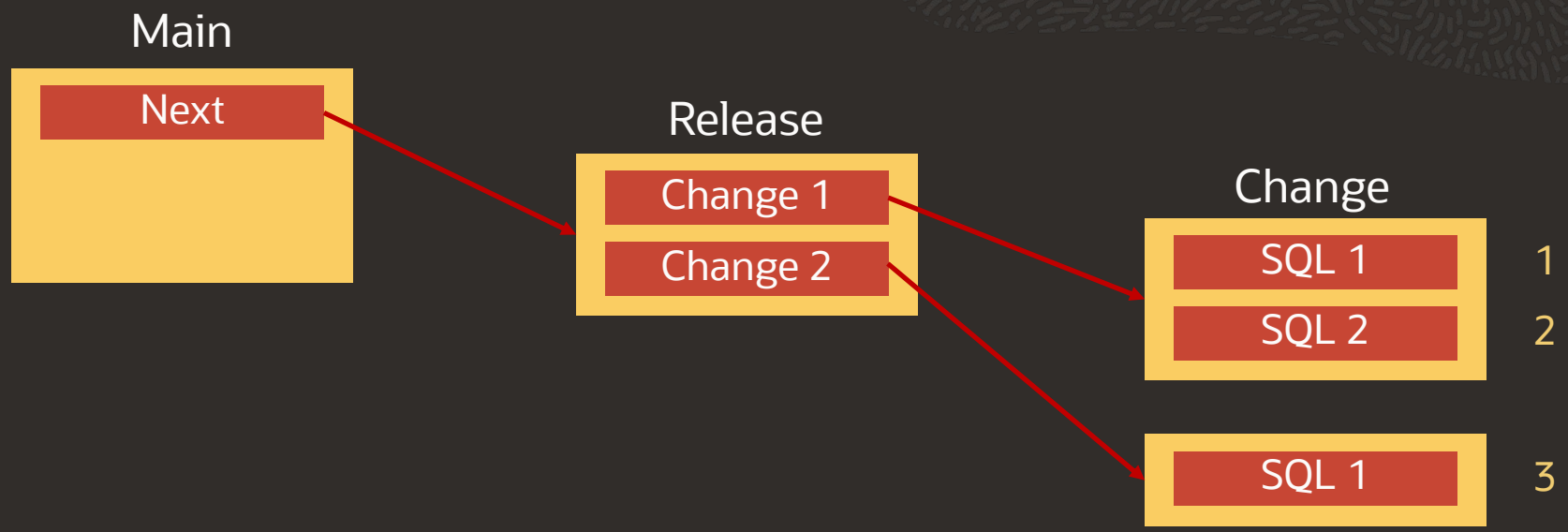
```
alter table emp
  add name varchar2(100);

alter table emp
  drop column fname;

alter table emp
  drop column lname;
```

Hierarchical changelogs in `dist`

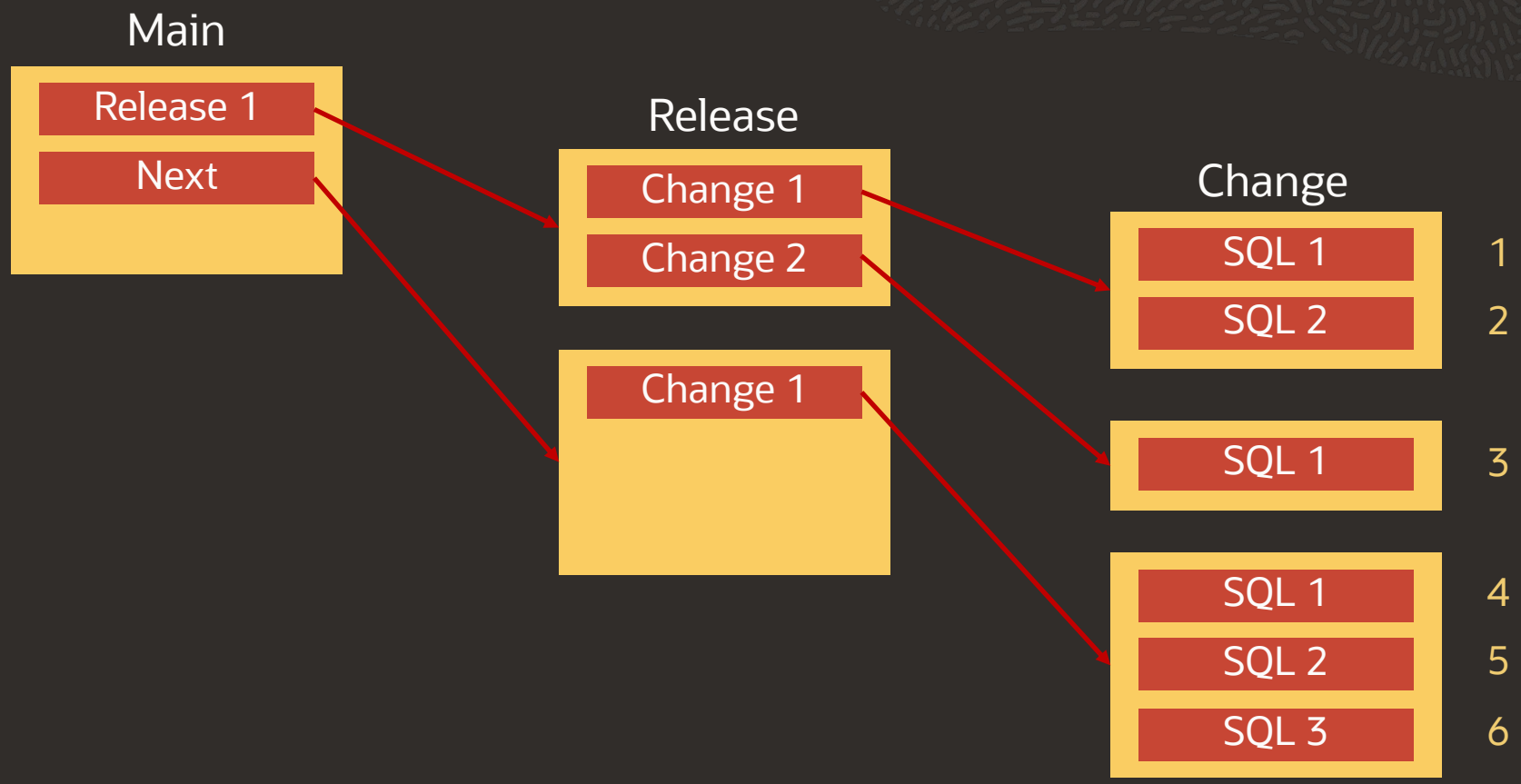
Hierarchical changelogs in `dist`

Hierarchical changelogs in `dist`

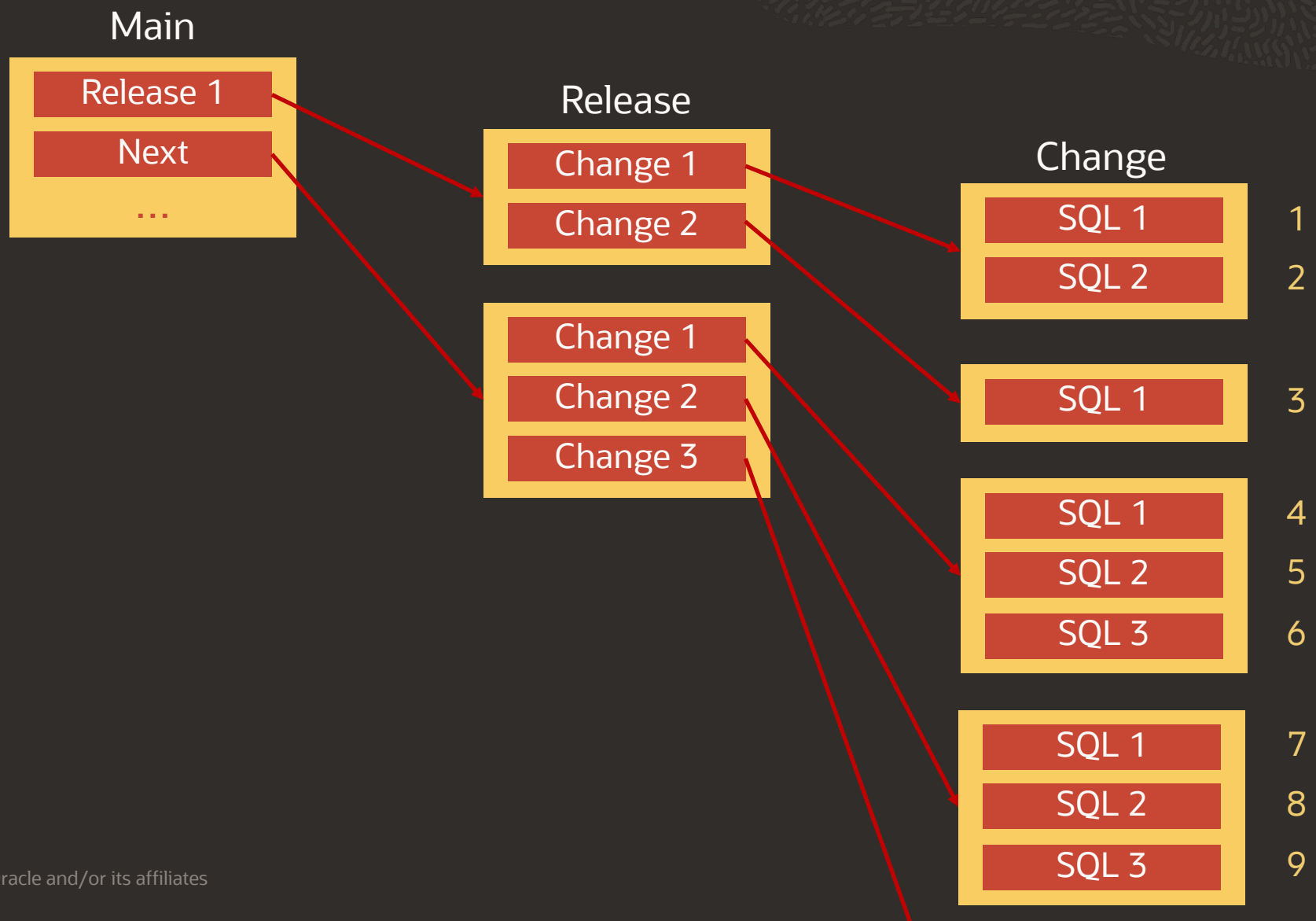
Hierarchical changelogs in `dist`



Hierarchical changelogs in `dist`



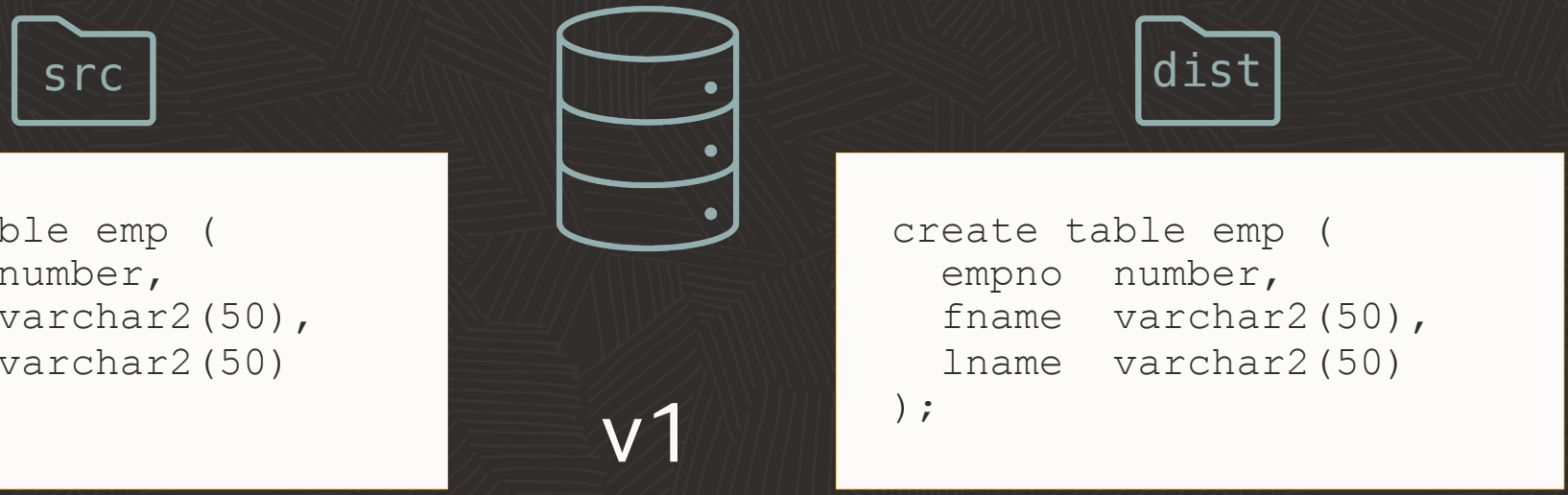
Hierarchical changelogs in `dist`



Process

1. Make changes in DEV DB
2. Sync repo with changes in DEV DB
   - Update src directory in repo using `project export`
3. Create Liquibase changelogs/sets
   - Update dist directory in repo using `project stage`
4. Test Liquibase changelogs/sets on TEST DB
5. Merge changes to repo

Using the process with SQLcl Projects



1. Make changes in DEV DB
2. Sync repo with changes in DEV DB
3. Create Liquibase changelogs/sets
4. Test Liquibase changelogs/sets on TEST DB
5. Merge changes to repo

Using the process with SQLcl Projects


```
create table emp (
  empno  number,
  fname  varchar2(50),
  lname  varchar2(50)
);
```

v2

```
create table emp (
  empno  number,
  fname  varchar2(50),
  lname  varchar2(50)
);
```

1. Make changes in DEV DB
2. Sync repo with changes in DEV DB
3. Create Liquibase changelogs/sets
4. Test Liquibase changelogs/sets on TEST DB
5. Merge changes to repo

Using the process with SQLcl Projects


```
create table emp (
  empno  number,
  fname  varchar2(50),
  lname  varchar2(50),
  deptno number
);
```

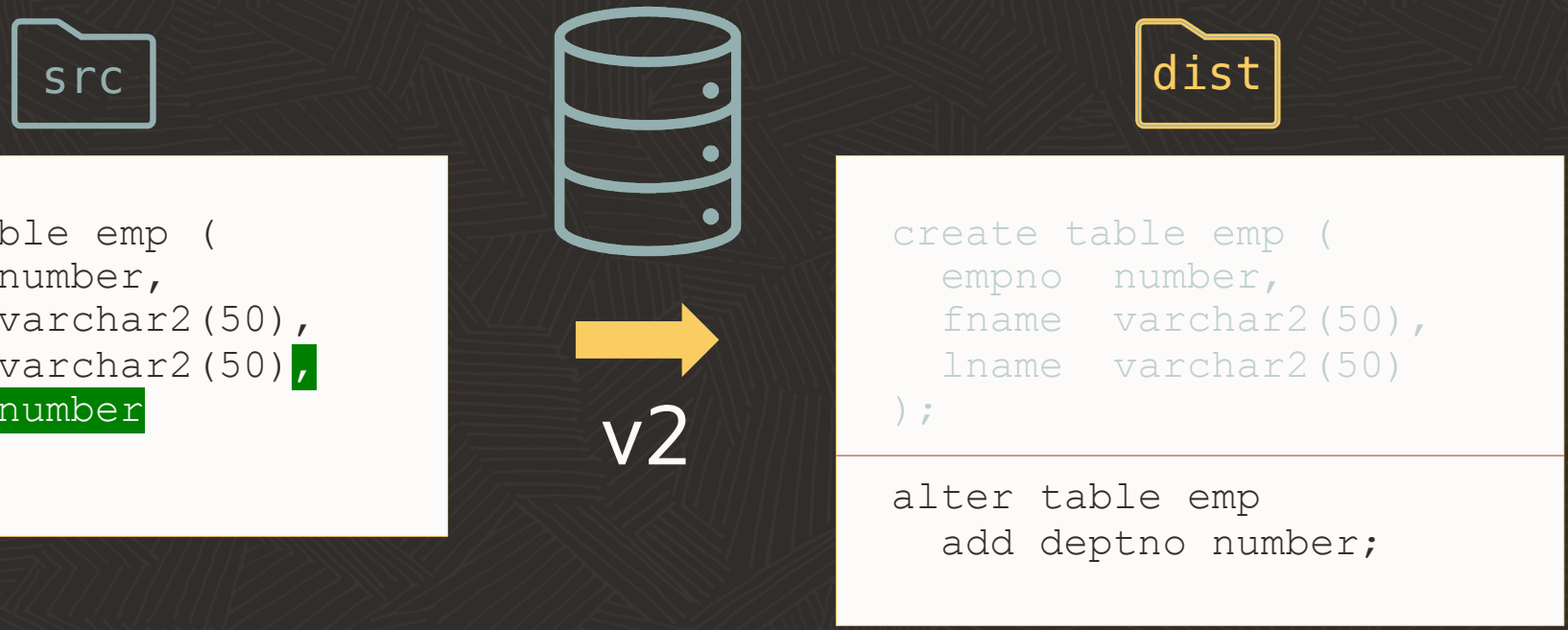
v2

```
create table emp (
  empno  number,
  fname  varchar2(50),
  lname  varchar2(50)
);
```

1. Make changes in DEV DB
2. Sync repo with changes in DEV DB
3. Create Liquibase changelogs/sets
4. Test Liquibase changelogs/sets on TEST DB
5. Merge changes to repo

```
project export -o emp
```

Using the process with SQLcl Projects



1. Make changes in DEV DB
2. Sync repo with changes in DEV DB
3. Create Liquibase changelogs/sets
4. Test Liquibase changelogs/sets on TEST DB
5. Merge changes to repo

```
project stage
```

Using the process with SQLcl Projects


```
create table emp (
  empno  number,
  fname  varchar2(50),
  lname  varchar2(50),
  deptno number
);
```

v2


```
create table emp (
  empno  number,
  fname  varchar2(50),
  lname  varchar2(50)
);
```

```
alter table emp
  add deptno number;
```

1. Make changes in DEV DB
2. Sync repo with changes in DEV DB
3. Create Liquibase changelogs/sets
4. Test Liquibase changelogs/sets on TEST DB
5. Merge changes to repo

```
project verify
project gen-artifact
project deploy
```

Using the process with SQLcl Projects


```
create table emp (
  empno  number,
  fname  varchar2(50),
  lname  varchar2(50),
  deptno number
);
```

v2

```
create table emp (
  empno  number,
  fname  varchar2(50),
  lname  varchar2(50)
);
```

```
alter table emp
  add deptno number;
```

1. Make changes in DEV DB
2. Sync repo with changes in DEV DB
3. Create Liquibase changelogs/sets
4. Test Liquibase changelogs/sets on TEST DB
5. Merge changes to repo

Summary of SQLcl Projects

Defined a process for capturing and curating database changes by developer checkins
Wiring these changes together automatically within a known framework
Giving the ability to add and order custom changes to each item added
Cut releases with known sets of changes
Generate artifacts that can be tested and then deployed

Getting Support

Oracle Developer Community Forums

https://forums.oracle.com/ords/apexds/domain/dev-community/category/sqlcl

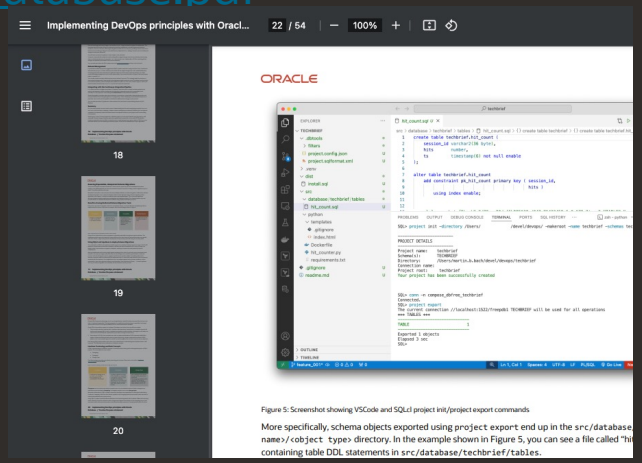
Tags: #CICD

Oracle LiveLab Hands-On CICD for a React Web app built on top of Oracle DB

https://livelabs.oracle.com/pls/apex/r/dbpm/livelabs/view-workshop?wid=4139

Oracle Database CI/CD Whitepaper

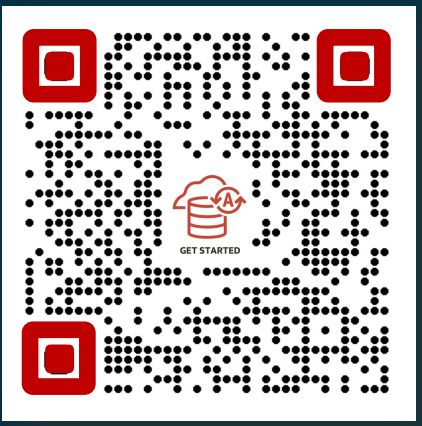
https://www.oracle.com/a/ocom/docs/database/implementing-devops-principles-with-oracle-database.pdf



Q&A Open

Important links to bookmark

Links to get you started and to keep up to date with Autonomous Database



1 New Get Started page:
oracle.com/autonomous-database/get-started/

2 Join us:
LinkedIn
bit.ly/adb-linkedin-grp

X
@AutonomousDW

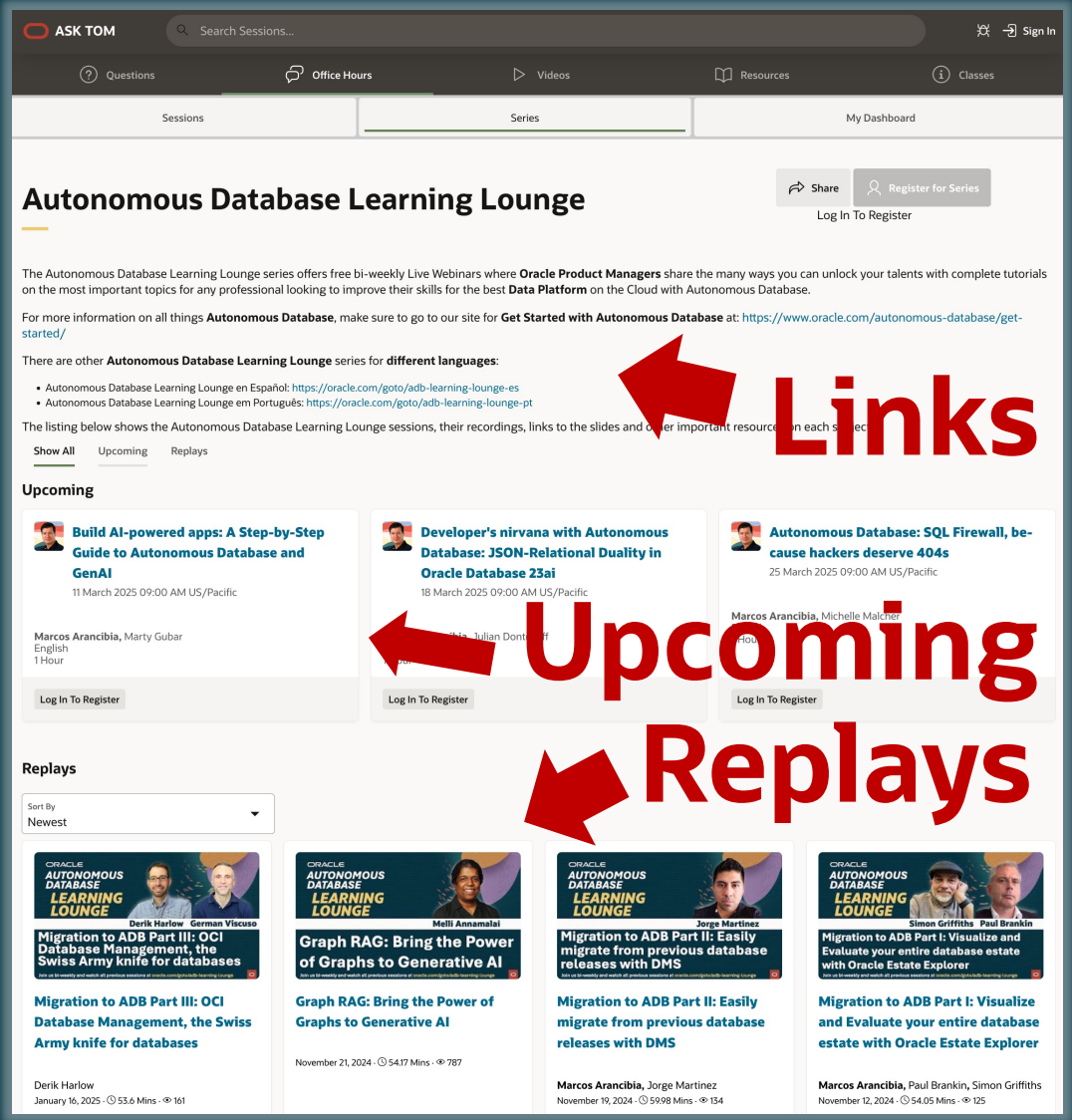
Bluesky
autonomousdb.bsky.social

3 Got a question?
We are on stackoverflow
bit.ly/adb-stackoverflow

Join us on Developers Slack
(search #oracle-autonomous-database)
bit.ly/odevrel_slack

Final Thoughts

oracle.com/goto/adb-learning-lounge



Thank you for joining !!!