

Stop editing code in Production database - a journey from 90's to CI/CD

Marcin Przepiorowski



SYMPOSIUM 42

Created by the community, to support the community

Sharing of reliable knowledge

Supporting the various user groups and individuals



@sym_42

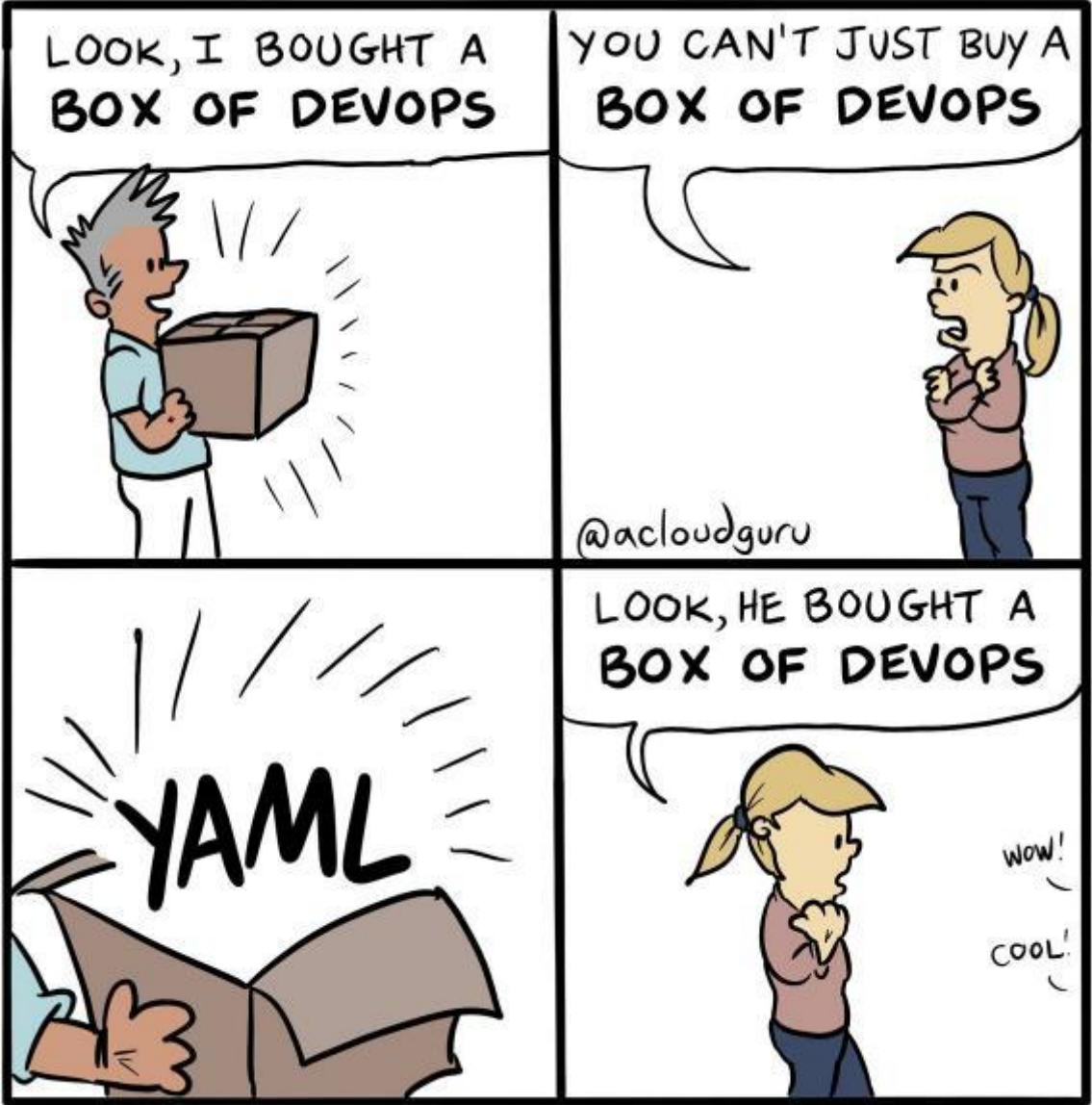


<https://sym42.org/>

Why this talk ?

A few years of observation how organizations are deploying pipelines / automation with databases

- Characteristic 1
 - The small / midsize group of very mature developers using Oracle since version 8.0 with a lot of PL/SQL and fixing code in production
 - Goal: Growth and have a control and easy to work environment
- Characteristic 2
 - Many different development group using many Oracle database for different applications and having one central “release management” team to process all changes and deploy to production
 - Goal: Going to DevOps model and increasing number of releases



LOOK, I BOUGHT A
BOX OF DEVOPS

YOU CAN'T JUST BUY A
BOX OF DEVOPS

@acloudguru

YAML

LOOK, HE BOUGHT A
BOX OF DEVOPS

WOW!
COOL!

What is important – DBA view

- All problems are caused by

CHANGE

- What has been changed
- When the change happened
- What to do to fix – best without special tools

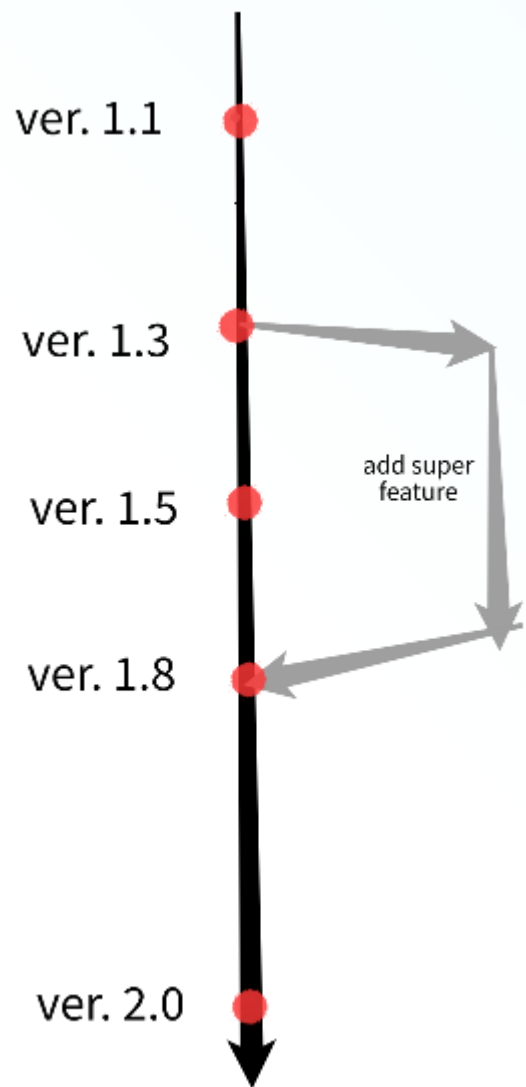
What is important – developer view

- Easy to use
- Works with different tools – even if people will keep working directly with development database(s)
- No adding to much overhead to normal development work

Challenges

- Database has a state – new deployment can't just replace a database - it has to transform database from current state to a new one
- Replacing code looks easier – although replacing a whole code it's complicated so it will be rather an incremental approach
- Data are kept in “static” objects – tables – table definition change is possible, but it should guarantee no data loss
- Size of database matter – data transformation to a new schema or even index definition change looks straight forward but execution time depend on data size
- Online vs offline changes
- Different states of databases in different environments

Challenges



DEV
V ???????



INT
v.2.0



UAT
v.1.8



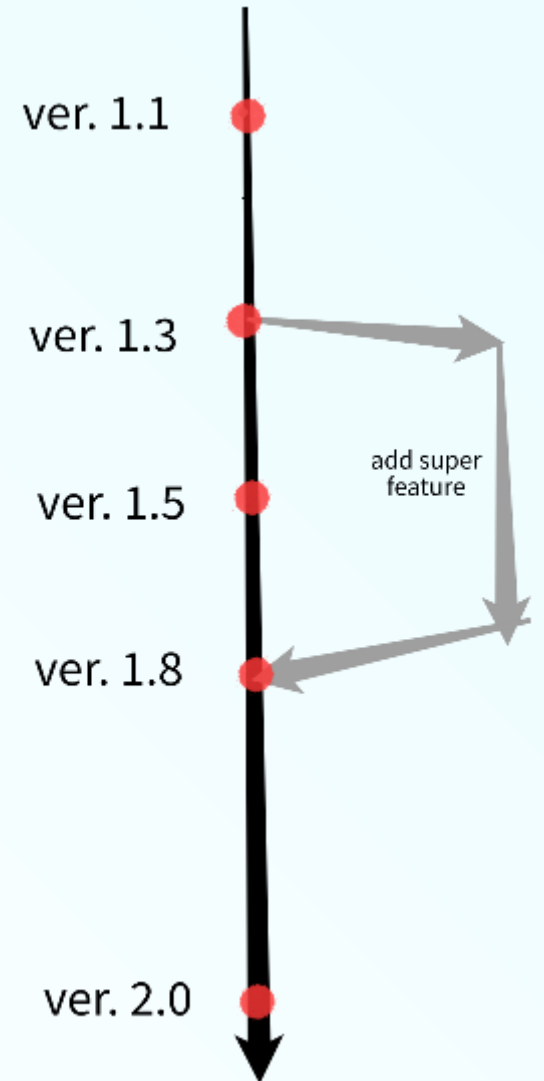
PREPROD
v.1.5



PRODUCTION
v.1.3

Challenges

- Each object can have a different version – not all changes are redeployed
- Tables doesn't have a version – maybe in comment ????
- The external tracking of deployed changes and connecting it to source control timeline is mandatory



Tools

- Investigation started with 2 tools:
 - Liquibase
 - SqlCl (leveraging Liquibase and DBMS_METADATA and DBMS_METADATA_DIFF)
- Result: "custom" scripts with git and Liquibase

Liquibase

- Changelog

Liquibase uses a changelog to explicitly list database changes in order. The changelog acts as a ledger of changes and contains a list of changesets (units of change) that Liquibase can execute on a database.

- Change set concept

Liquibase uses changesets to represent a single change to your database.

Liquibase

- Directory structure
 - One or many repos
 - Owner or owner and type ?

- One or many changes logs
 - One main
 - One object type

- How to manage PL/SQL code

Liquibase

"Static objects"

```
<changeSet author="marcinp" id="create-demotable">
  <createTable tableName="demotable">
    <column name="columnName1" type="VARCHAR2(355)">
      </column>
    </createTable>
  </changeSet>
```

```
--liquibase formatted sql
--changeset marcinp:create-demotable
CREATE TABLE demotable(
  columnName1 VARCHAR (355)
);
--rollback DROP TABLE
--rollback demotable
```

```
<databaseChangeLog ... >
<changeSet author="marcin" id="test_ver" failOnError="true">
  <sqlFile dbms="oracle" endDelimiter="/"
    path="TEST_1008_1.sql"
    relativeToChangelogFile="true"
    splitStatements="false"
    stripComments="false"/>
</changeSet>
```

```
cat SCOTT/TABLE/TEST_1008_1.sql
CREATE TABLE "SCOTT"."TEST_1008_1"
(
  "ID" NUMBER(10,0) NOT NULL ENABLE,
  "NEWCOL" VARCHAR2(20)
) PCTFREE 10 PCTUSED 40 NOCOMPRESS LOGGING
TABLESPACE "USERS" ;
```

Liquibase

"Dynamic objects"

proc.xml

```
<databaseChangeLog ... >
<changeSet author="marcin" id="test_ver" failOnError="true">
  <sqlFile dbms="oracle" endDelimiter="/"
    path="PROCEDURE_FOOPROC.sql"
    relativeToChangelogFile="true"
    splitStatements="false"
    stripComments="false"/>
</changeSet>
```

PROCEDURE_FOOPROC.sql

```
CREATE OR REPLACE PROCEDURE
"SCOTT"."FOOPROC"
as begin
    dbms_output.put_line('dev2');
end;
/
```

```
SQL> select * from DATABASECHANGELOG where filename like 'proc%' order by ORDEREXECUTED;
```

ID	AUTHOR	FILENAME	DATEEXECUTED	EXECTYPE
test_ver	marcin	proc.xml	04-AUG-22 08:13:10.445794000	EXECUTED

Liquibase

"Dynamic objects"

proc.xml

```
<databaseChangeLog ... >
<changeSet author="marcin" id="test_ver" failOnError="true">
  <sqlFile dbms="oracle" endDelimiter="/"
    path="PROCEDURE_FOOPROC.sql"
    relativeToChangelogFile="true"
    splitStatements="false"
    stripComments="false"/>
</changeSet>
```

PROCEDURE_FOOPROC.sql

```
CREATE OR REPLACE PROCEDURE
"SCOTT"."FOOPROC"
as begin
    -- lets change a code a bit
    dbms_output.put_line('dev2');
end;
/
```

Unexpected error running Liquibase: Validation Failed:

1 change sets check sum

proc1.xml::3e391ce59cbd44913879649703139346d0975cbc::marcin@delphix.com was:

8:f0263aa8cb96d8df4168a545e71f2c17 but is now: 8:34e937a19376c6a7d27074744453e251

Liquibase

"Dynamic objects"

proc.xml

```
<databaseChangeLog ... >
<changeSet author="marcin" id="test_ver" failOnError="true"
  runOnChange="true">
  <sqlFile dbms="oracle" endDelimiter="/"
    path="PROCEDURE_FOOPROC.sql"
    relativeToChangelogFile="true"
    splitStatements="false"
    stripComments="false"/>
</changeSet>
```

PROCEDURE_FOOPROC.sql

```
CREATE OR REPLACE PROCEDURE
"SCOTT"."FOOPROC"
as begin
    -- lets change a code a bit
    dbms_output.put_line('dev2');
end;
/
```

```
SQL> select * from DATABASECHANGELOG where filename like 'proc%' order by ORDEREXECUTED;
```

ID	AUTHOR	FILENAME	DATEEXECUTED	EXECTYPE
test_ver	marcin	proc.xml	04-AUG-22 08:15:10.445794000	RERAN

Sqlcl Liquibase extension

- Directory structure
 - Oracle example is a one directory per schema
 - With v1/v2 subdir but this can be replaced with git
- One or many changes logs
- Oracle example - One main per schema



Sqlcl Liquibase extension

```
SQL> lb genschema
lb genschema
[Type - TYPE_SPEC]:                153 ms
[Type - TYPE_BODY]:                29 ms
[Type - SEQUENCE]:                 48 ms
[Type - CLUSTER]:                  27 ms
[Type - TABLE]:                   36 ms
[Type - MATERIALIZED_VIEW_LOG]:    19 ms
[Type - MATERIALIZED_VIEW]:        6 ms
[Type - VIEW]:                     148 ms
[Type - REF_CONSTRAINT]:           272 ms
[Type - DIMENSION]:                23 ms
[Type - FUNCTION]:                 27 ms
[Type - PROCEDURE]:                64 ms
[Type - PACKAGE_SPEC]:             171 ms
[Type - DB_LINK]:                  14 ms
[Type - SYNONYM]:                  22 ms
[Type - INDEX]:                    202 ms
[Type - TRIGGER]:                   51 ms
[Type - PACKAGE_BODY]:             252 ms
[Method loadCaptureTable]:          1864 ms
[Method parseCaptureTableRecords]: 7342 ms
[Method sortCaptureTable]:          30 ms
[Method createExportChangeLogs]:    3 ms

Export Flags Used:
Export Grants                false
Export Synonyms              false
```

Be warned – it may be VERY slow if you have many objects



Sqlcl Liquibase extension

```
SQL> select owner, object_type, count(*) from dba_objects where object_type in (...) and owner in ('UTILITIES','XXXXDATA') group by owner, object_type;
```

OWNER	OBJECT_TYPE	COUNT(*)
XXXXDATA	INDEX	1033
XXXXDATA	PACKAGE	1
XXXXDATA	TABLE	1772
XXXXDATA	PROCEDURE	5
UTILITIES	INDEX	1060
UTILITIES	TABLE	2686
UTILITIES	FUNCTION	81
UTILITIES	PROCEDURE	417
UTILITIES	PACKAGE	513
UTILITIES	PACKAGE BODY	513
UTILITIES	VIEW	66

```
SQL> select count(*) from dba_constraints where owner in ('UTILITIES','XXXXDATA');  
COUNT(*)
```

7352

Liquibase generated by SQLcl

```
SQL> lb genobject -type PROCEDURE -name FOOPROC
Action successfully completed please review created file FOOPROC_procedure.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<databaseChangeLog
  ...
  xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.9.xsd">
  <changeSet id="859147a9947e66e53f87bed03dfefa69db1cb4fe" author="(HR2)-Generated"
  failOnError="false" >
    <n0:createOracleProcedure objectName="FOOPROC" objectType="PROCEDURE" ownerName="HR2" >
      <n0:source><![CDATA[CREATE OR REPLACE EDITIONABLE PROCEDURE "HR2"."FOOPROC" as begin null; end;
]]></n0:source>
    </n0:createOracleProcedure>
  </changeSet>
</databaseChangeLog>
```

```
echo "CREATE OR REPLACE EDITIONABLE PROCEDURE \"HR2\".\"FOOPROC\" as begin null; end;" | shasum
859147a9947e66e53f87bed03dfefa69db1cb4fe -
```

Liquibase generated by SQLcl

DDL for table – column rename

```
SQL> desc emp
```

Name	Null?	Type
EMPNO	NOT NULL	NUMBER (4)
ENAME		VARCHAR2 (10)
JOB		VARCHAR2 (9)
...		

```
SQL> alter table emp rename column ename to ename_new;
```

```
SQL> desc emp
```

Name	Null?	Type
EMPNO	NOT NULL	NUMBER (4)
ENAME_NEW		VARCHAR2 (10)
JOB		VARCHAR2 (9)
...		

```
SQL> lb genobject -type TABLE -name EMP
```

Action successfully completed please review created file emp_table.xml

Liquibase generated by SQLcl

DDL for table – column rename - deployment

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	marcin						

```
SQL> lb update -changelog emp_table1.xmlScriptRunner Executing:  
emp_table1.xml::980a4756ff1533f18092619bdb5f3d8a7dada301::(SCOTT)-Generated -- DONE
```

```
##### ERROR SUMMARY #####
```

```
Errors encountered:0
```

```
SQL>
```

```
SQL> select * from emp;
```

EMPNO	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	ENAME_NEW
10							

Liquibase generated by SQLcl

DDL for table – column rename - deployment

```
SQL> lb rollback -changelog emp_table1.xml -count 1Rolling Back
Changeset:emp_table1.xml::980a4756ff1533f18092619bdb5f3d8a7dada301:: (SCOTT) -Generated
ScriptRunner Executing: emp_table1.xml::980a4756ff1533f18092619bdb5f3d8a7dada301:: (SCOTT) -Generated -
- DONE
```

```
##### ERROR SUMMARY #####
Errors encountered:0
```

```
SQL>
```

```
SQL> select * from emp;
```

EMPNO	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	ENAME
10							

Liquibase generated by SQLcl

DDL for table – column rename - deployment

```
SQL> lb updatesql -changelog emp_table1.xml
```

```
ScriptRunner Logging: emp_table1.xml::980a4756ff1533f18092619bdb5f3d8a7dada301:: (SCOTT) -Generated --  
DONE
```

```
-- *****
```

```
-- Update Database Script
```

```
-- *****
```

```
-- Change Log: emp_table1.xml
```

```
-- Ran at: 16/08/22 08:18
```

```
-- Against: HR2@jdbc:oracle:thin:@source:1521/pdbdev
```

```
-- Liquibase version: 4.1.1
```

```
-- *****
```

```
-- Changeset emp_table1.xml::980a4756ff1533f18092619bdb5f3d8a7dada301:: (SCOTT) -Generated
```

```
ALTER TABLE "EMP" ADD ("ENAME_NEW" VARCHAR2(10))
```

```
/
```

```
ALTER TABLE "EMP" DROP ("ENAME")
```

```
/
```


Solution

- Keep all database objects in source code repository in clear form
- All commits should have a list of objects you want to deploy to the database – to create an artefact
- Use pipeline to:
 - Generate a Liquibase change sets automatically from source code – use commit SHA as ID column
 - Use Liquibase to manage an object state in database
 - Add additional checks, like object dependency, compilation, etc

How to find objects to deploy

- Git can show files changes from last commit
- It's not enough if some files was not change but it's still not exist in target system.
 - Deployment into integration fails with rollback
 - New version of body developed, package code is same (it should be always delivered together)
 - New commit has on new version of file, but more files are needed for deployment
 - Object deletion
- Each commit should have a list of files to deploy – developer has a full control what objects will be deployed

How to find objects to deploy

```
cat 5efe7694-d5e2-11ec-b0bb-acde48001122  
A, SCOTT/TABLE/TEST_1705_2.sql  
A, SCOTT/PRIMARY_KEY/TEST_1705_2_PK.sql  
D, SCOTT/INDEX/TEST_1705_2_IND.sql
```

```
cat 385ffe80-d5e8-11ec-9e14-acde48001122  
A, SCOTT/VIEW/EMP_VIEW_DEPT.sql  
M, SCOTT/PACKAGE_BODY/NEWPACK.sql
```

GIT repository with all code

```
pioro dbcode [main !?]$ ls -l SCOTT/
total 0
drwx-----  4 pioro  staff  128 17 May 14:59 FOREIGN_KEY
drwx-----  4 pioro  staff  128 17 May 14:59 FUNCTION
drwx-----  7 pioro  staff  224 17 May 15:07 INDEX
drwx-----  5 pioro  staff  160 17 May 14:59 PACKAGE
drwx-----  5 pioro  staff  160 17 May 14:59 PACKAGE_BODY
drwx----- 13 pioro  staff  416 17 May 18:17 PRIMARY_KEY
drwx-----  4 pioro  staff  128 17 May 14:59 PROCEDURE
drwx----- 24 pioro  staff  768 10 Aug 15:05 TABLE
drwx-----  6 pioro  staff  192 17 May 15:49 VIEW
```

Solution

Static objects

- Ex. tables
 - Keep a latest definition as CREATE
 - Needs DELTA files with changes

Static objects

```
git checkout 5c41704dac0cdfbb8d949f2d881527167d983724
cat SCOTT/TABLE/TEST_1705_2.sql
CREATE TABLE "SCOTT"."TEST_1705_2"
  (      "ID" NUMBER(10,0) NOT NULL ENABLE,
        "NEWCOL" VARCHAR2(20),
        "COL2" VARCHAR2(20)
  )
```

```
git checkout main
cat SCOTT/TABLE/TEST_1705_2.sql
CREATE TABLE "SCOTT"."TEST_1705_2"
  (      "ID" NUMBER(10,0) NOT NULL ENABLE,
        "NEWCOL" VARCHAR2(20),
        "COL2" VARCHAR2(20),
        "COL3" VARCHAR2(100)
  )
```

```
cat SCOTT/TABLE/DELTA/TEST_1705_2.sql
alter table TEST_1705_2 add col3 varchar2(100);
```

Solution

“Dynamic” objects

- PL/SQL, views, constraints, indexes
 - Keep a latest definition as CREATE

Dynamic objects

```
git checkout f5a416e6e719f59307765b48e4dba563133d87dc
cat SCOTT/PACKAGE_BODY/NEWPACK.sql
```

```
CREATE OR REPLACE EDITIONABLE PACKAGE BODY "SCOTT"."NEWPACK" AS

    procedure newpack_add AS
    BEGIN
        -- TODO: Implementation required for procedure NEWPACK.newpack_add
        NULL;
    END newpack_add;
END NEWPACK;
/
```


Dynamic objects

```
git checkout main
cat SCOTT/PACKAGE_BODY/NEWPACK.sql
```

```
CREATE OR REPLACE EDITIONABLE PACKAGE BODY "SCOTT"."NEWPACK" AS

    procedure newpack_add AS
        a varchar2(200);
    BEGIN
        -- TODO: Implementation required for procedure NEWPACK.newpack_add
        -- v3
        select empno into a from EMP_VIEW_DEPT;
    END newpack_add;

END NEWPACK;
/
```

PL/SQL status check with deployment

Compile

Objects are just deployed – Liquibase doesn't know if create package fails

```
<changeSet author="marcin@delphix.com" id="419335b4bfde476c8c6d855287b248e47f76bdfd"
failOnError="true" runAlways="false" >
  <sqlFile dbms="oracle"
    endDelimiter="/"
    path="PACKAGE_BODY_NEWPACK.sql"
    relativeToChangelogFile="true"
    splitStatements="false"
    stripComments="false"/>
</changeSet>
```

```
<changeSet author="marcin@delphix.com" id="Validation" runAlways="true" runOnChange="true">
  <preConditions onFail="HALT" onFailMessage="Found invalid object">
    <sqlCheck expectedResult="0"><![CDATA[
      SELECT COUNT(1) FROM user_objects
      WHERE status <> 'VALID' AND object_type = 'PACKAGE BODY' and object_name = 'NEWPACK';
    ]]></sqlCheck>
  </preConditions>
</changeSet>
```

Compile dependencies

- Compile package / package body
- Your deployed packages can be OK but what about other existing PL/SQL objects depending on deployed code
- Status needs to be checked and some objects needs to be recompiled
- In case of error – a deployment should be rollback

Restricted objects

- Sometimes PL/SQL objects are used by all other objects, for example logging functionality is used by all other objects
- Changing logging may need to stop all online sessions to replace PL/SQL without locking
- Or using Edition-Based Redefinition
- Such objects should be only deployed with extra approval as without EBR it may requires an application downtime

Rollback

- Static objects
 - no rollback – data are king
 - Always move forward
- Dynamic rollback
 - As targets may be on different patch level – rollback scripts needs to be generated per deployment

How to mark release

- Liquibase tag is used to mark a commit scope for Liquibase
- Tag is equal to commit SHA
- Liquibase rollback can use tag
- Release should be defined in git using git tags

Different patch levels or empty database

- Compare database logs with all source control
 - Replay missing commits from git
 - Risk(s):
 - Depend on source control model there could be wrong commits which never pass tests
-
- Other approach – take all files from deploy and generate one deployment
 - Risk(s):
 - Not all commits are deployed one by one – final stage may require summary of delta changes for static objects

Syncer

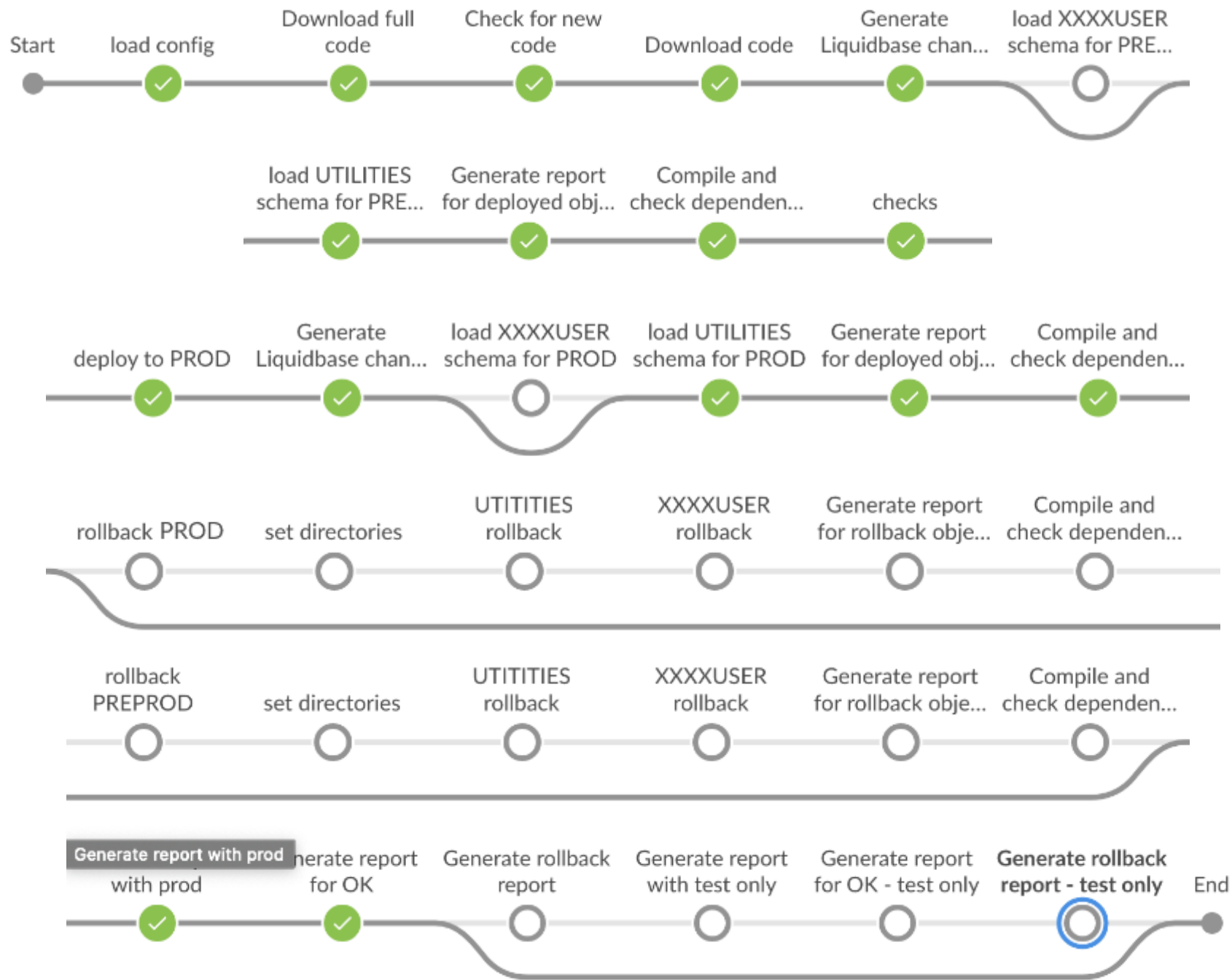
- Help developers to extract source code from database
- Support git operations like commit, push and pull
- Generate a deploy file with list of files to include in deployment
- Potentially add a formatting but this git hooks can be leverage as well

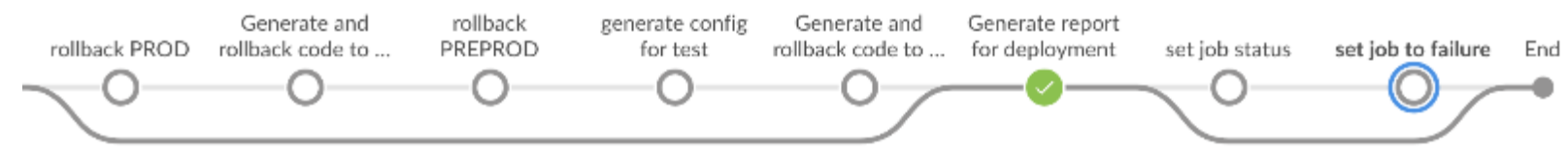
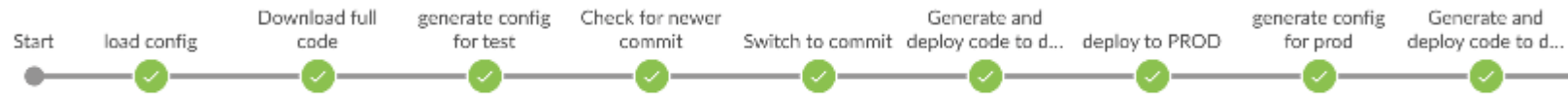
```
python syncer.py extract \  
/Users/pioro/mystuff/dbcode/SCOTT/PACKAGE/DEMO_BILL.sql \  
/Users/pioro/mystuff/dbcode/SCOTT/PACKAGE/DEPENDON.sql \  
/Users/pioro/mystuff/dbcode/SCOTT/PACKAGE_BODY/DEMO_BILL.sql\  
/Users/pioro/mystuff/dbcode/SCOTT/PACKAGE_BODY/DEPENDON.sql
```

```
python syncer.py commit --message "2 packages"
```


Generator

- Help developers to create a valid LB change logs easily from git repository
- Handle order of deployment
- Handle restrictions
- Handle LB deployments and additional checks
- Generate a report





Q & A