

24.04.2024

Hy APEX connect_ DevOps Made Easy!

with Oracle Autonomous Database using PL/SQL and Git
Timo Herwix, Senior Consultant

code of change

 **Hyand** by
GOD | MT

Who am I?



Senior Consultant at Hyand since 2019

Previously worked as a Data Warehouse Developer

Oracle APEX since 2016

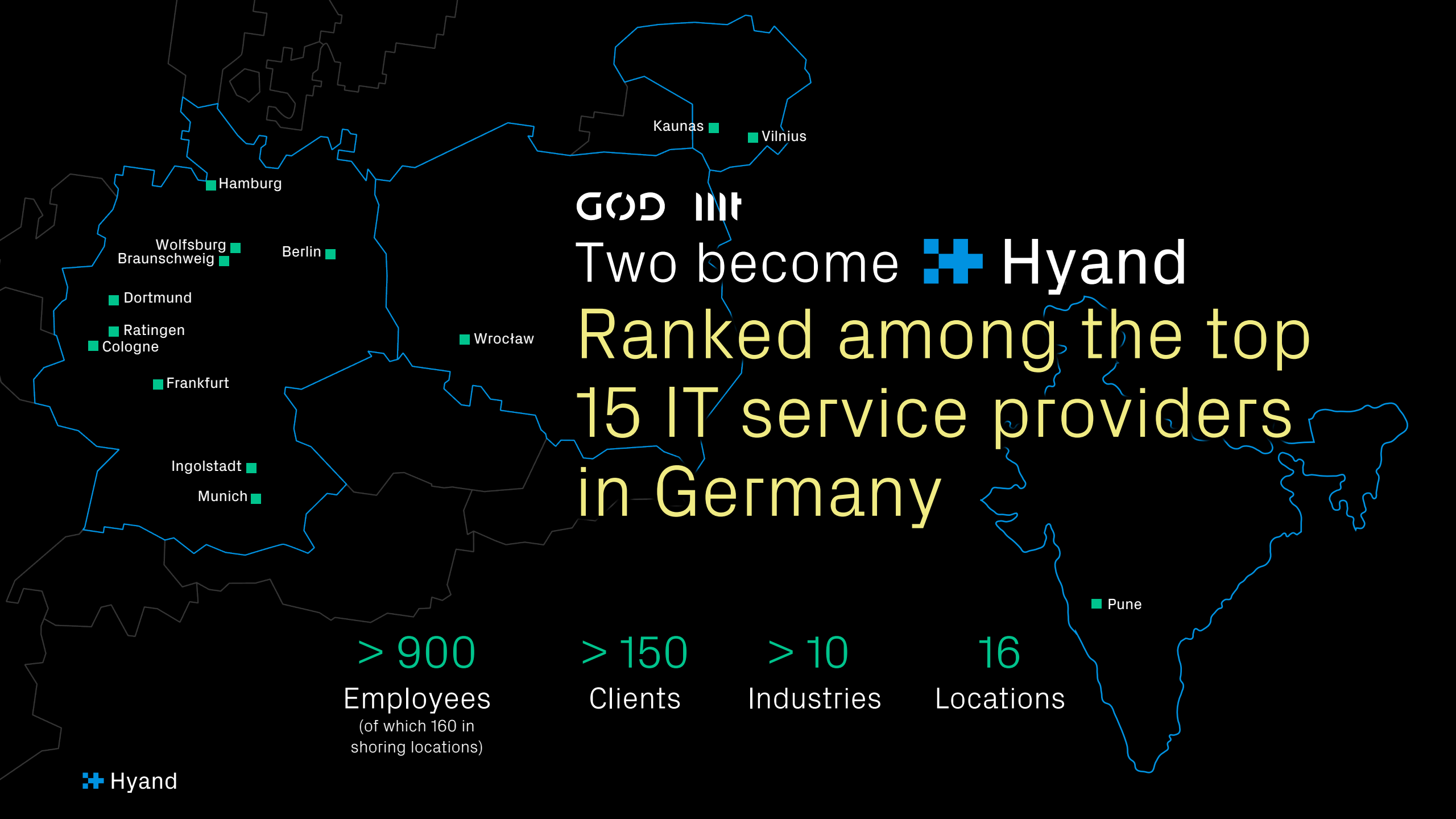
Oracle Databases since 2008

Blog author, conference speaker

Born in 1983, two children and living in Germany

Timo Herwix
Senior Consultant





GOD III

Two become  Hyand
Ranked among the top
15 IT service providers
in Germany

Hamburg

Kaunas

Vilnius

Wolfsburg
Braunschweig

Berlin

Dortmund

Ratingen
Cologne

Wrocław

Frankfurt

Ingolstadt

Munich

Pune

> 900

Employees
(of which 160 in
shoring locations)

> 150

Clients

> 10

Industries

16

Locations

1

Introduction

2

Let's dive deeper

3

Wrap-up

DevOps Made Easy?

DevOps Made Easy!

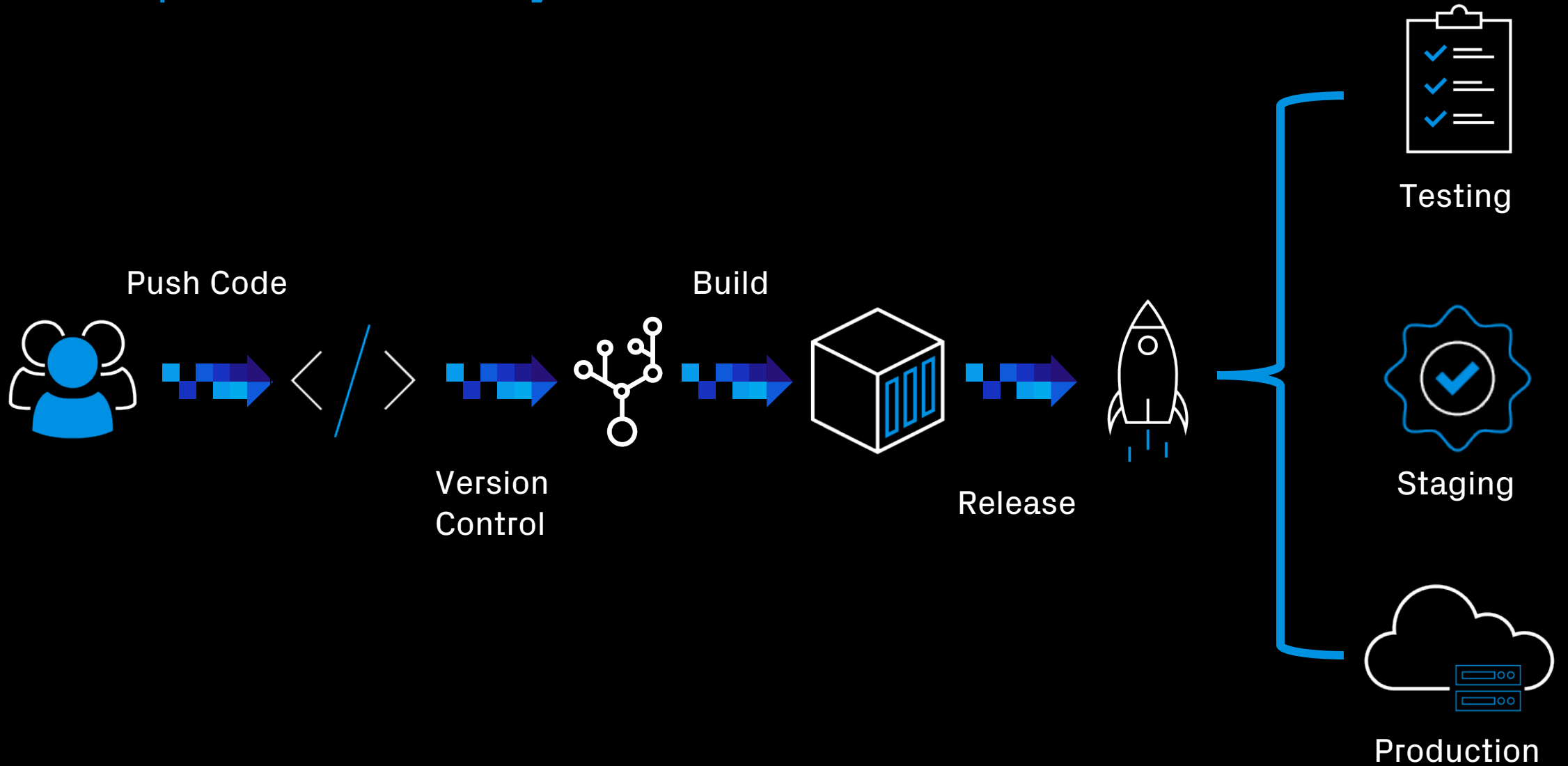
As an **APEX Developer**, you might be looking to apply modern development methodologies and tools used in other development platforms to your Oracle APEX Low-Code Projects.

This includes:

- Git-based code version management
- Code review
- Continuous delivery of apps from one instance to another
- Tracking issues
- Managing your team development



DevOps Made Easy!



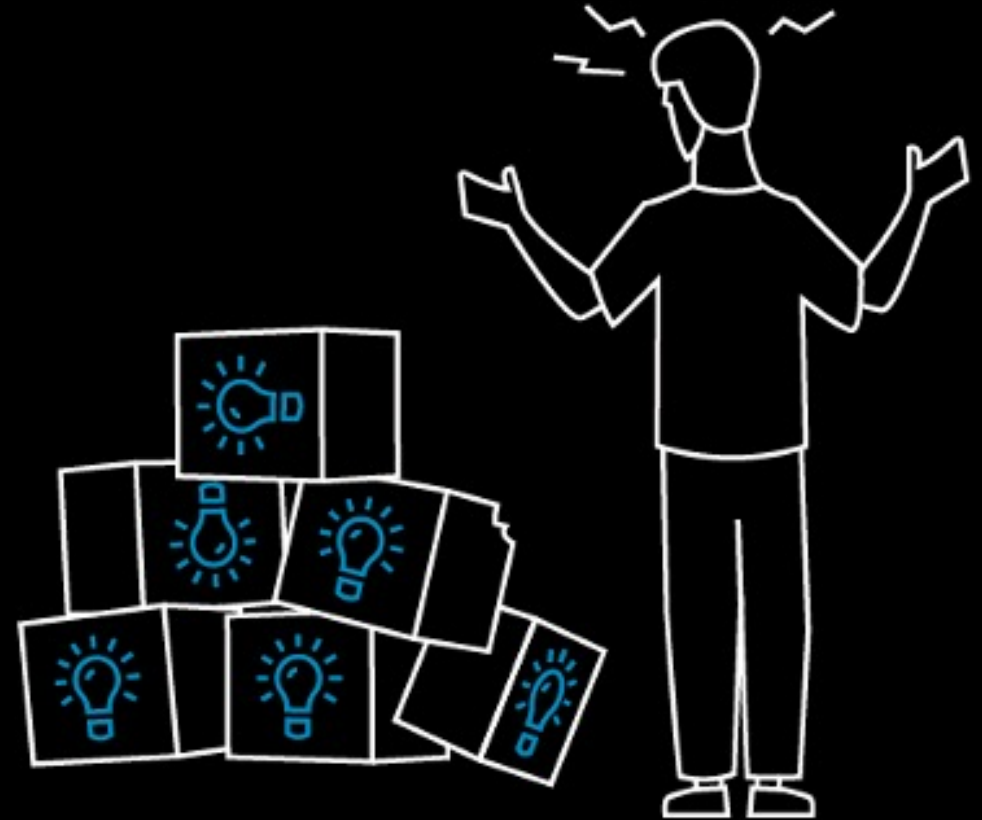
But why Easy?

But why Easy?

The Oracle Autonomous Database (ADB) includes the `DBMS_CLOUD_REPO` package, an extremely powerful package that provides **easy** access to files in Cloud Code (Git) Repositories.

With this package, you can:

- Manage repositories
- Handle code in a repository
- Export database schemas and objects
- Execute SQL statements from committed files

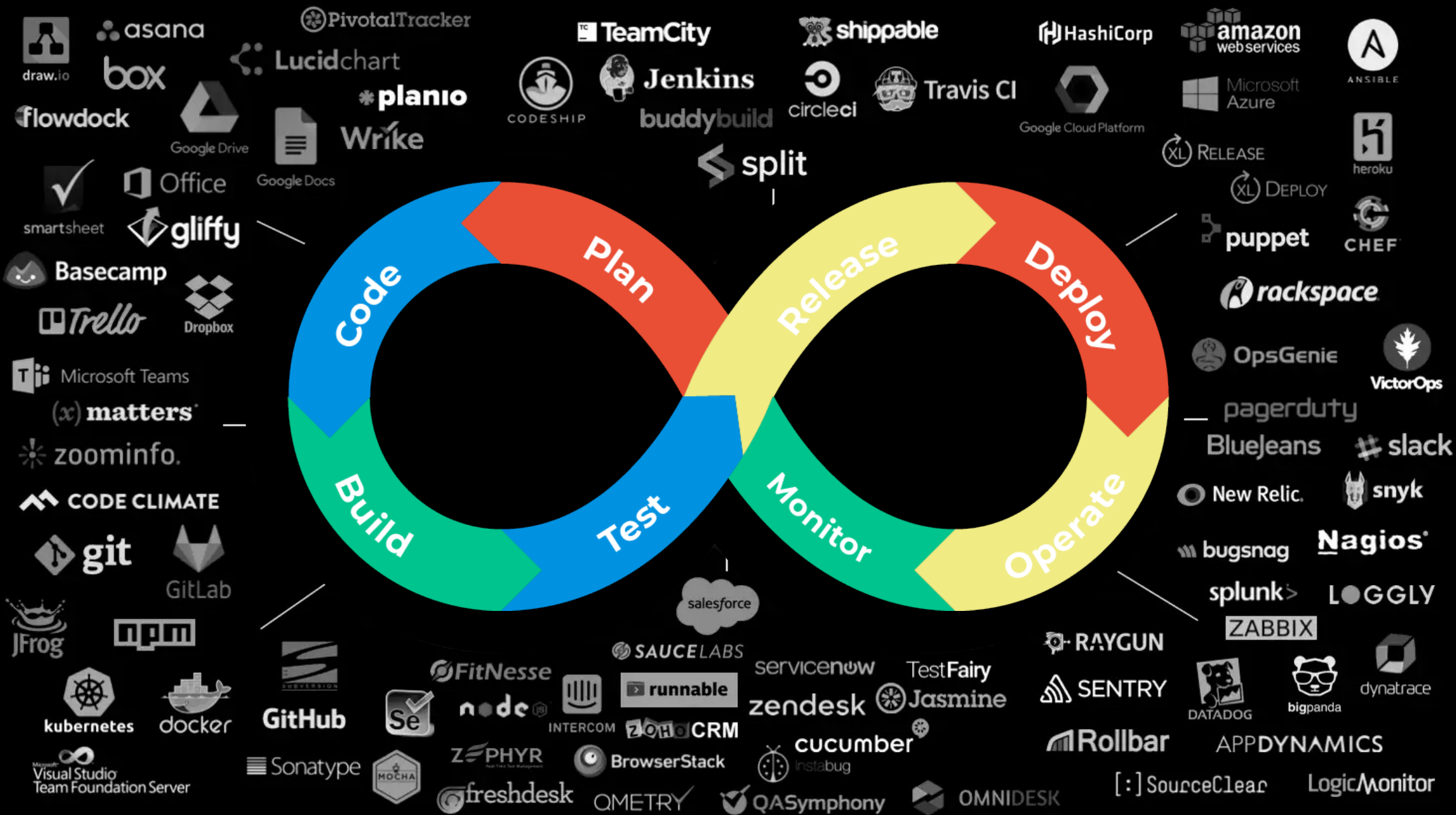


Supported Code Repositories

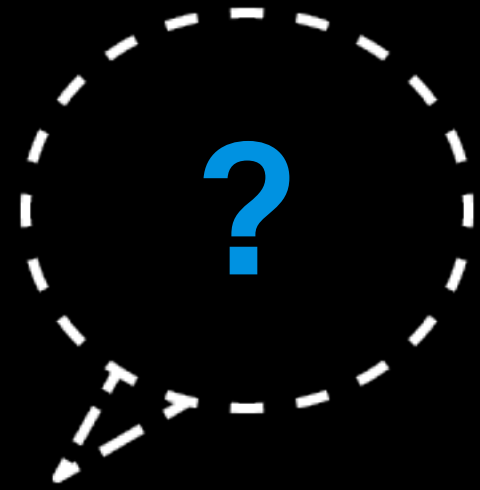
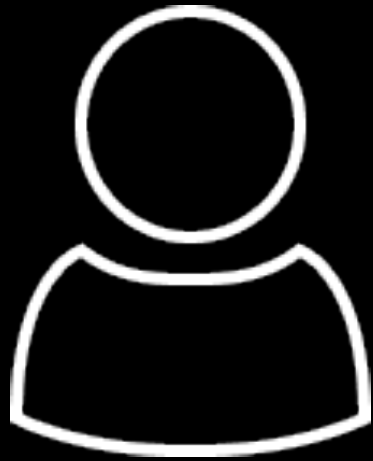
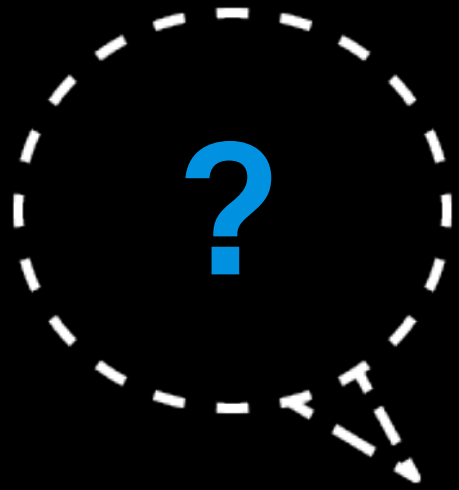


Azure Repos

Wondering why you should use
the package instead of relying
on proven tools like Jenkins?



WHASUPP!?



Simple! It's ideal for PL/SQL enthusiasts who are looking for a proven CI/CD strategy for **smaller projects with limited budgets** and don't have a **large team, the necessary expertise or enough time!**

1

Introduction

2

Let's dive deeper

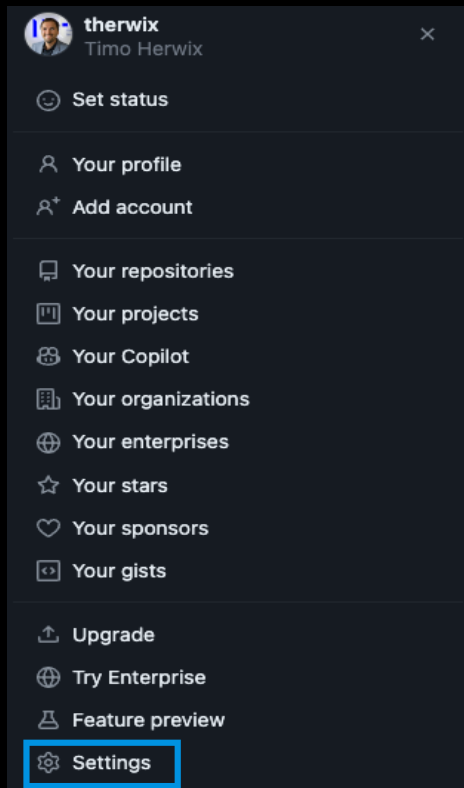
3

Wrap-up

Get started with Repository interaction!

Create the credential for interacting with your GitHub Repository

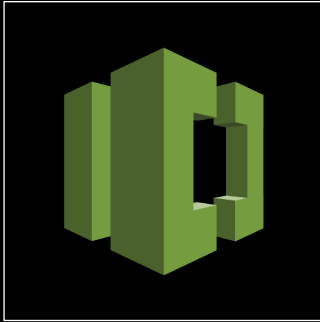
Create a Personal Access Token



Save your PAT in a Cloud Service Credential

```
1 begin
2   dbms_cloud.create_credential (
3     credential_name => 'GITHUB_CRED',
4     username        => 'therwix',
5     password        => 'github_pat_...'
6   );
7 end;
8 /
```

Subprograms for initialization operations



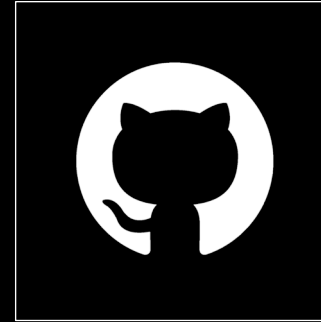
INIT_AWS_REPO

This function initializes an AWS repository handle.



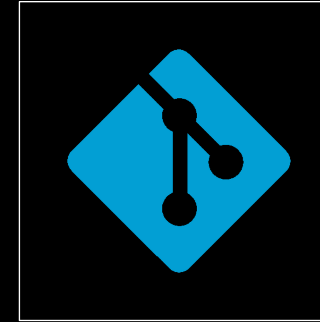
INIT_AZURE_REPO

This function initializes an Azure repository handle.



INIT_GITHUB_REPO

This function initializes a GitHub repository handle.



INIT_REPO

This function initializes a Code Repository handle.

Check if the access works

```
1 SELECT name, owner, description, created, last_modified
2 FROM dbms_cloud_repo.list_repositories(dbms_cloud_repo.init_github_repo(
3     credential_name => 'GITHUB_CRED', -- Name of the previously created credential
4     repo_name       => 'therwix',    -- Name of the GitHub Repository
5     owner           => 'therwix'     -- Name of the GitHub Repository Owner
6 ));
```



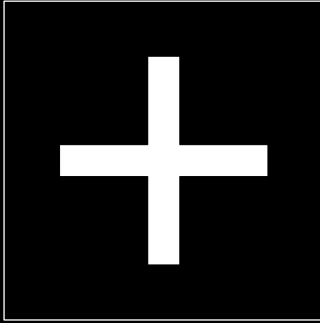
If everything works well with the credential setup, you should see a list of repositories that you can access.

All rows fetched: 4 in 1.476 seconds

	NAME	OWNER	DESCRIPTION	CREATED	LAST_MODIFIED
1	dev	therwix	(null)	06/03/20 10:55:22.000 GMT+01:00	06/03/20 10:55:22.000 GMT+01:00
2	strack-software-validate-constraints-plugin	therwix	APEX dynamic action plugin for automatic client-side constraint validations	29/10/21 12:45:48.000 GMT+02:00	29/10/21 12:45:49.000 GMT+02:00
3	tc_responsive_number_counter	therwix	(null)	14/12/23 10:28:40.000 GMT+01:00	15/01/24 23:29:08.000 GMT+01:00
4	xlsx_builder	therwix	A PL/SQL Package to create OOXML workbooks.	06/07/20 11:44:19.000 GMT+02:00	06/07/20 11:44:21.000 GMT+02:00

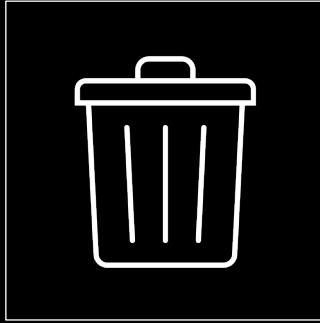
Manage your Code Repository!

Subprograms for the Repository Management Operations



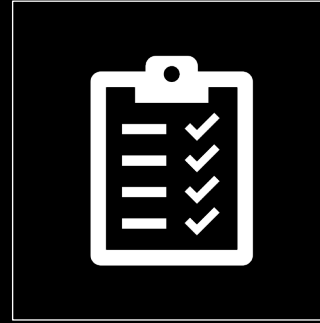
CREATE_REPOSITORY

This procedure creates a Code Repository.



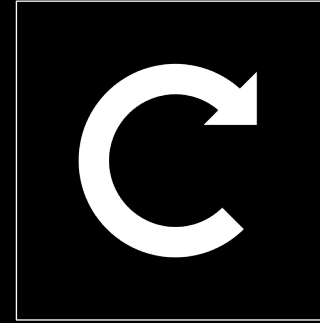
DELETE_REPOSITORY

This procedure deletes the Code Repository.



LIST_REPOSITORIES

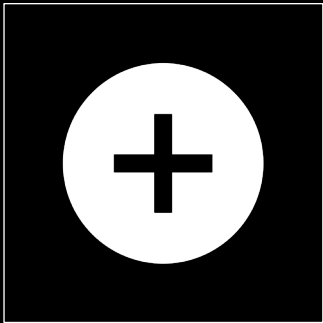
This function lists all the Code Repositories.



UPDATE_REPOSITORY

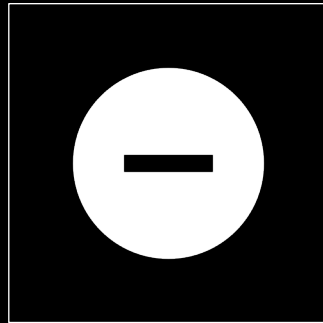
This procedure updates a Code repository.

Subprograms for the Repository Branch Management Operations



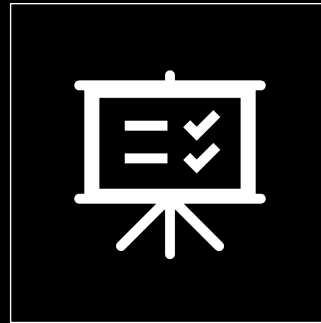
CREATE_BRANCH

This procedure creates a branch in a Code Repository.



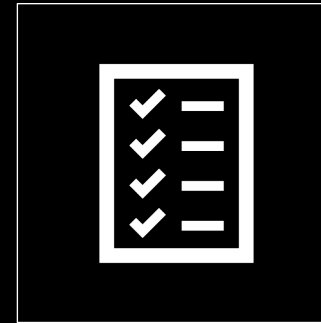
DELETE_BRANCH

This procedure deletes a branch in a Code Repository.



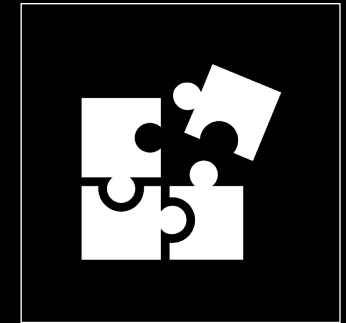
LIST_BRANCHES

This function lists all the Code Repository branches.



LIST_COMMITS

This function lists all the commits in a Code Repository branch.



MERGE_BRANCH

This procedure merges a branch into another specified branch in a Code Repository.



Create a new Repository

```
1 declare
2     repoHandle      clob;
3     repoCredential  varchar2(50 CHAR) := 'GITHUB_CRED';      -- Name of the previously created credential
4     repoName        varchar2(50 CHAR) := 'APEX_WORLD_2024'; -- Name of the new GitHub Repository
5     repoOwner       varchar2(50 CHAR) := 'therwix';         -- Name of the GitHub Repository Owner
6 begin
7     repoHandle := dbms_cloud_repo.init_github_repo(
8         credential_name => repoCredential,
9         repo_name       => repoName,
10        owner           => repoOwner
11    );
12
13    dbms_cloud_repo.create_repository(
14        repo           => repoHandle,
15        description    => 'Repo created with DBMS_CLOUD_REPO',
16        private        => TRUE
17    );
18 end;
19 /
```


Initialize a new Repository

```
1 declare
2     repoHandle      clob;
3     repoCredential  varchar2(50 CHAR) := 'GITHUB_CRED';      -- Name of the previously created credential
4     repoName        varchar2(50 CHAR) := 'APEX_WORLD_2024';  -- Name of the GitHub Repository
5     repoOwner       varchar2(50 CHAR) := 'therwix';         -- Name of the GitHub Repository Owner
6 begin
7     repoHandle := dbms_cloud_repo.init_github_repo(
8         credential_name => repoCredential,
9         repo_name      => repoName,
10        owner          => repoOwner
11    );
12
13    dbms_cloud_repo.put_file(
14        repo => repoHandle,
15        file_path => 'readme.md',
16        contents => utl_raw.cast_to_raw('APEX WORLD 2024'),
17        branch_name => 'main',
18        commit_details => json_object('message' value 'DBMS_CLOUD_REPO commit',
19                                     'author' value 'therwix',
20                                     'email' value 'timo.herwix@mt-ag.com'
21    )
22 );
23
24 end;
25 /
```

Create a new Branch



```
1 declare
2     repoHandle      clob;
3     repoCredential  varchar2(50 CHAR) := 'GITHUB_CRED'; -- Name of the previously created credential
4     repoName        varchar2(50 CHAR) := 'APEX_WORLD_2024'; -- Name of the GitHub Repository
5     repoOwner       varchar2(50 CHAR) := 'therwix'; -- Name of the GitHub Repository Owner
6 begin
7     repoHandle := dbms_cloud_repo.init_github_repo(
8         credential_name => repoCredential,
9         repo_name       => repoName,
10        owner           => repoOwner
11    );
12
13    dbms_cloud_repo.create_branch(
14        repo             => repoHandle,
15        branch_name     => 'development',
16        parent_branch_name => 'main'
17    );
18 end;
19 /
```

Display all branches in a repository



```
1 SELECT * FROM DBMS_CLOUD_REPO.LIST_BRANCHES (dbms_cloud_repo.init_github_repo(  
2                                     credential_name => 'GITHUB_CRED',  
3                                     repo_name       => 'APEX_WORLD_2024',  
4                                     owner           => 'therwix'));
```



If everything works well,
you should see a list with the branches of your Repository.



	NAME
1	development
2	main

Clean-up!

Delete a Branch

```
1 declare
2   repoHandle      clob;
3   repoCredential  varchar2(50 CHAR) := 'GITHUB_CRED'; -- Name of the previously created credential
4   repoName        varchar2(50 CHAR) := 'APEX_WORLD_2024'; -- Name of the GitHub Repository
5   repoOwner       varchar2(50 CHAR) := 'therwix'; -- Name of the GitHub Repository Owner
6 begin
7   repoHandle := dbms_cloud_repo.init_github_repo(
8     credential_name => repoCredential,
9     repo_name       => repoName,
10    owner           => repoOwner
11  );
12
13  dbms_cloud_repo.delete_branch(
14    repo             => repoHandle,
15    branch_name     => 'development'
16  );
17 end;
18 /
```

Delete a Repository

```
1 declare
2   repoHandle      clob;
3   repoCredential  varchar2(50 CHAR) := 'GITHUB_CRED'; -- Name of the previously created credential
4   repoName        varchar2(50 CHAR) := 'APEX_WORLD_2024'; -- Name of the GitHub Repository
5   repoOwner       varchar2(50 CHAR) := 'therwix'; -- Name of the GitHub Repository Owner
6 begin
7   repoHandle := dbms_cloud_repo.init_github_repo(
8     credential_name => repoCredential,
9     repo_name       => repoName,
10    owner           => repoOwner
11  );
12
13  dbms_cloud_repo.delete_repository(
14    repo             => repoHandle
15  );
16 end;
17 /
```

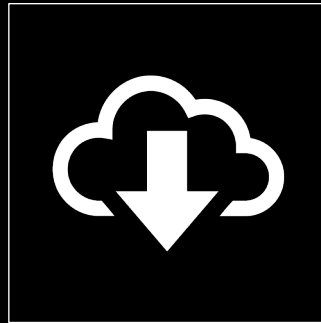
Moving content to the Code Repository!

Subprograms for File Operations



DELETE_FILE

This procedure deletes a file from the Code repository.



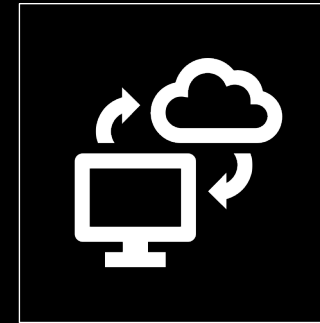
GET_FILE

The function downloads the contents of a file from the Code repository.



LIST_FILES

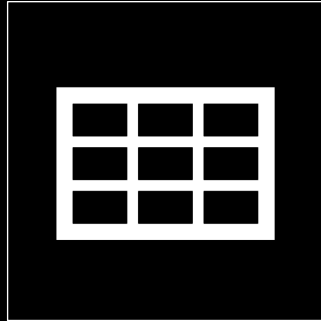
This function lists all the files in a Code Repository.



PUT_FILE

This procedure uploads a file to the Code repository.

Subprograms for Export Operations of Database Objects



EXPORT_OBJECT

This procedure uploads the DDL metadata of a database object to the Code repository.



EXPORT_SCHEMA

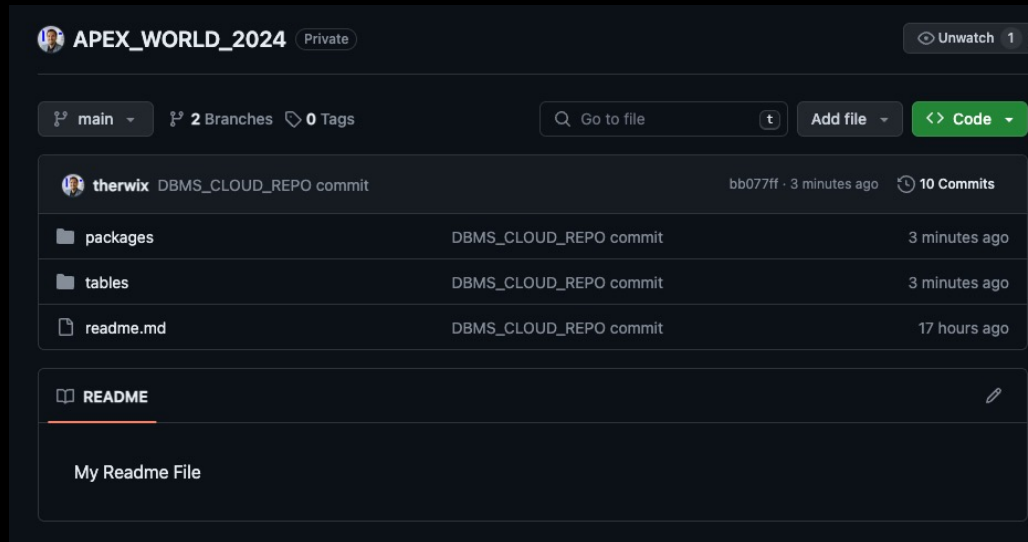
This procedure exports metadata of all objects in a schema to a Code Repository.

Export all schema objects to single files

```
1 declare
2   repoHandle    clob;
3   repoCredential varchar2(50 CHAR) := 'GITHUB_CRED';    -- Name of the previously created credential
4   repoName      varchar2(50 CHAR) := 'APEX_WORLD_2024'; -- Name of the GitHub Repository
5   repoOwner     varchar2(50 CHAR) := 'therwix';        -- Name of the GitHub Repository Owner
6 begin
7   repoHandle := dbms_cloud_repo.init_github_repo(
8     credential_name => repoCredential,
9     repo_name      => repoName,
10    owner          => repoOwner
11  );
12  for rec in (
13    select *
14    from dba_objects
15    where object_type in ('TABLE', 'VIEW', 'PACKAGE', 'PACKAGE BODY')
16          and owner = 'TMAPEX'
17  )
18  loop
19    dbms_cloud_repo.export_object(
20      repo => repoHandle,
21      file_path => case rec.object_type
22                   when 'TABLE' then 'tables/' || lower(rec.object_name) || '.sql'
23                   when 'VIEW' then 'views/' || lower(rec.object_name) || '.sql'
24                   when 'PACKAGE' then 'packages/' || lower(rec.object_name) || '.pks'
25                   when 'PACKAGE BODY' then 'packages/' || lower(rec.object_name) || '.pkb'
26                 end,
27      object_type => case rec.object_type
28                     when 'TABLE' then 'TABLE'
29                     when 'VIEW' then 'VIEW'
30                     when 'PACKAGE' then 'PACKAGE_SPEC'
31                     when 'PACKAGE BODY' then 'PACKAGE_BODY'
32                   end,
33      object_name => rec.object_name,
34      object_schema => 'TMAPEX',
35      branch_name => 'main',
36      commit_details => json_object('message' value 'DBMS_CLOUD_REPO commit',
37                                   'author' value 'therwix',
38                                   'email' value 'timo.herwix@mt-ag.com'
39                                   ),
40      append => false
41    );
42  end loop;
43 end;
44 /
```


Export all schema objects to single files

View of the Repository

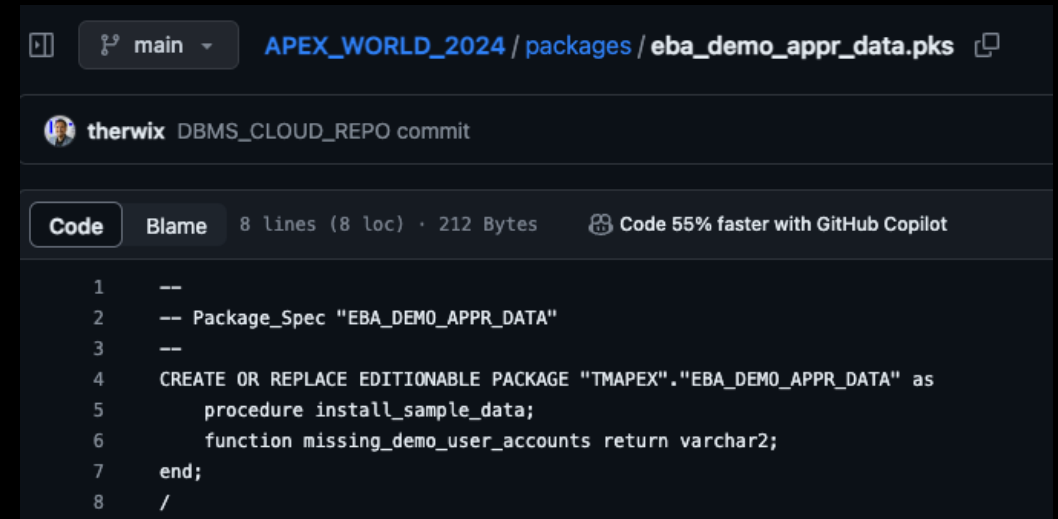


The screenshot shows the GitHub repository interface for 'APEX_WORLD_2024'. At the top, it indicates the repository is private and has 1 watcher. Below this, there are navigation options for 'main' branch, 2 branches, and 0 tags. A search bar for files and buttons for 'Add file' and '<> Code' are visible. The main content area shows a commit by 'therwix' (DBMS_CLOUD_REPO) from 3 minutes ago. Below the commit, a table lists the repository's contents:

File/Folder	Commit	Time
packages	DBMS_CLOUD_REPO commit	3 minutes ago
tables	DBMS_CLOUD_REPO commit	3 minutes ago
readme.md	DBMS_CLOUD_REPO commit	17 hours ago

At the bottom, there is a 'README' section with a 'My Readme File' placeholder.

View of a DDL-Script



The screenshot shows the view of a DDL script file named 'eba_demo_appr_data.pks' within the 'packages' directory of the 'APEX_WORLD_2024' repository. The commit is by 'therwix' (DBMS_CLOUD_REPO). The script content is as follows:

```
1  --
2  -- Package_Spec "EBA_DEMO_APPR_DATA"
3  --
4  CREATE OR REPLACE EDITIONABLE PACKAGE "TMAPEX"."EBA_DEMO_APPR_DATA" as
5      procedure install_sample_data;
6      function missing_demo_user_accounts return varchar2;
7  end;
8  /
```

Export all schema objects to a single file

```
1 declare
2     repoHandle      clob;
3     repoCredential  varchar2(50 CHAR) := 'GITHUB_CRED';      -- Name of the previously created credential
4     repoName        varchar2(50 CHAR) := 'APEX_WORLD_2024';  -- Name of the GitHub Repository
5     repoOwner       varchar2(50 CHAR) := 'therwix';         -- Name of the GitHub Repository Owner
6 begin
7     repoHandle := dbms_cloud_repo.init_github_repo(
8         credential_name => repoCredential,
9         repo_name       => repoName,
10        owner           => repoOwner
11    );
12
13    dbms_cloud_repo.export_schema(
14        repo             => repoHandle,
15        schema_name     => 'TMAPEX',
16        file_path       => 'myschema_ddl.sql'
17    );
18
19 end;
20 /
```

Exporting an APEX Application!



You can easily export APEX applications to a Code repository too. All you need to do is call the `APEX_EXPORT` package and pass the application ID to the `GET_APPLICATION` function.

But, there's a tiny thing to remember: the output of `GET_APPLICATION`, which is a CLOB, needs to be converted to a BLOB to work with the `DBMS_CLOUD_REPO.PUT_FILE` procedure.

However, `APEX_UTIL` has a helpful function to do this.

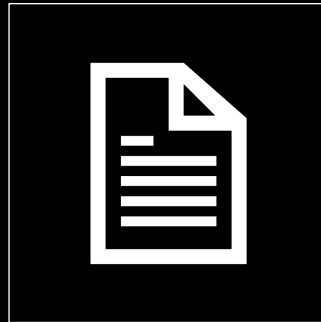
Exporting an APEX Application

```
1 declare
2     repoHandle      clob;
3     repoCredential  varchar2(50 CHAR) := 'GITHUB_CRED';      -- Name of the previously created credential
4     repoName        varchar2(50 CHAR) := 'APEX_WORLD_2024';  -- Name of the GitHub Repository
5     repoOwner       varchar2(50 CHAR) := 'therwix';         -- Name of the GitHub Repository Owner
6
7     l_file          apex_t_export_files;
8     l_app_id        number := 108;                          -- App-ID of the exported application
9     l_name          varchar2(255 CHAR);                      -- Name of the exported application
10    l_app_clob       clob;
11    l_app_blob       blob;
12 begin
13     repoHandle := dbms_cloud_repo.init_github_repo(
14         credential_name => repoCredential,
15         repo_name      => repoName,
16         owner          => repoOwner
17     );
18
19     l_file := apex_export.get_application(p_application_id => l_app_id);
20     l_name := l_file(1).name;
21
22     l_app_clob := l_file(1).contents;
23     l_app_blob := apex_util.clob_to_blob(l_app_clob);
24
25     dbms_cloud_repo.put_file(
26         repo      => repoHandle,
27         file_path => 'apex/' || l_name,
28         contents  => l_app_blob,
29         branch_name => 'main',
30         commit_details => json_object('message' value 'DBMS_CLOUD_REPO commit',
31                                     'author'   value 'therwix',
32                                     'email'    value 'timo.herwix@mt-ag.com'
33     )
34 );
35 end;
36 /
```



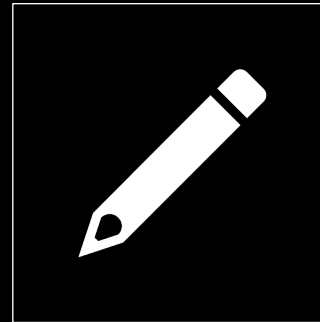
Perform SQL Operations from Code Repositories!

Subprograms for SQL Install Operations



INSTALL_FILE

This procedure installs SQL statements from a file in the Code repository.



INSTALL_SQL

This procedure installs SQL statements from a buffer given as input.



Creating an installation script

```
1 declare
2     repoHandle      clob;
3     repoCredential  varchar2(50 CHAR) := 'GITHUB_CRED';    -- Name of the previously created credential
4     repoName        varchar2(50 CHAR) := 'APEX_WORLD_2024'; -- Name of the GitHub Repository
5     repoOwner       varchar2(50 CHAR) := 'therwix';       -- Name of the GitHub Repository Owner
6 begin
7     repoHandle := dbms_cloud_repo.init_github_repo(
8         credential_name => repoCredential,
9         repo_name       => repoName,
10        owner           => repoOwner
11    );
12
13    dbms_cloud_repo.put_file(
14        repo => repoHandle,
15        file_path => 'install_db.sql',
16        contents => utl_raw.cast_to_raw('
17            @@tables/eba_demo_appr_approvers.sql
18            @@tables/eba_demo_appr_dept.sql
19            @@tables/eba_demo_appr_emp.sql
20            @@tables/eba_demo_appr_laptop_requests.sql
21            @@tables/eba_demo_appr_sal_history.sql
22            @@packages/eba_demo_appr.pks
23            @@packages/eba_demo_appr_data.pks
24            @@packages/eba_demo_appr.pkb
25            @@packages/eba_demo_appr_data.pkb'),
26        branch_name => 'main',
27        commit_details => json_object('message' value 'DBMS_CLOUD_REPO commit',
28                                     'author'   value 'therwix',
29                                     'email'    value 'timo.herwix@mt-ag.com'
30    )
31    );
32
33 end;
34 /
```

Execute the installation script

```
1 declare
2     repoHandle      clob;
3     repoCredential  varchar2(50 CHAR) := 'GITHUB_CRED';      -- Name of the previously created credential
4     repoName        varchar2(50 CHAR) := 'APEX_WORLD_2024';  -- Name of the GitHub Repository
5     repoOwner       varchar2(50 CHAR) := 'therwix';         -- Name of the GitHub Repository Owner
6 begin
7     repoHandle := dbms_cloud_repo.init_github_repo(
8         credential_name => repoCredential,
9         repo_name       => repoName,
10        owner           => repoOwner
11    );
12
13    dbms_cloud_repo.install_file(
14        repo             => repoHandle,
15        file_path        => 'install_db.sql',
16        branch_name     => 'main',
17        stop_on_error   => true
18    );
19
20 end;
21 /
```

Deploy an APEX Application from a Script

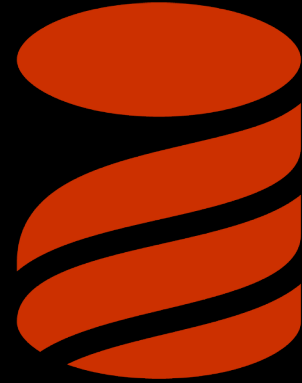
```
1 declare
2     repoHandle      clob;
3     repoCredential varchar2(50 CHAR) := 'GITHUB_CRED';      -- Name of the previously created credential
4     repoName        varchar2(50 CHAR) := 'APEX_WORLD_2024'; -- Name of the GitHub Repository
5     repoOwner       varchar2(50 CHAR) := 'therwix';        -- Name of the GitHub Repository Owner
6     repoApp         clob;
7     l_file          apex_t_export_files;
8     l_app_id        number := 108;
9 begin
10    repoHandle := dbms_cloud_repo.init_github_repo(
11        credential_name => repoCredential,
12        repo_name       => repoName,
13        owner           => repoOwner
14    );
15
16    repoApp := dbms_cloud_repo.get_file(
17        repo           => repoHandle,
18        file_path      => 'apex/f' || l_app_id || '.sql',
19        branch_name    => 'main'
20    );
21
22    l_file := apex_t_export_files (
23        apex_t_export_file (
24            name      => 'apex/f' || l_app_id || '.sql',
25            contents => repoApp));
26
27    apex_util.set_workspace('DEMO');
28
29    apex_application_install.set_application_id (
30        p_application_id => l_app_id);
31
32    apex_application_install.install(
33        p_source => l_file,
34        p_overwrite_existing => true);
35
36 end;
37 /
```

Deploy an APEX Application from a Script

The screenshot displays the Oracle APEX App Builder interface. At the top, there is a navigation bar with the APEX logo, 'App Builder' dropdown, 'SQL Workshop', 'Team Development', and 'Gallery' menus. A search bar and user profile 'DE demo' are also present. Below the navigation bar, four main action buttons are visible: 'Create', 'Import', 'Dashboard', and 'Workspace Utilities'. A search bar with a 'Go' button and 'Actions' dropdown is located below these buttons. The main workspace area shows a list of applications, with 'Sample Approvals' (108) highlighted by a red rectangular box. To the right, a sidebar contains 'About' information, 'Recent' applications (listing 'Sample Approvals - 108'), and 'Tasks' (listing 'Manage Backups' and 'Browse by Facets'). The bottom of the interface includes a footer with user information 'demo', 'demo', 'en', copyright text 'Copyright © 1999, 2023, Oracle and/or its affiliates.', and the version 'Oracle APEX 23.1.5'.

What is about the table differences
between our environments???

Get started with Liquibase



Liquibase

...or you are a PL/SQL nerd and use
DBMS_METADATA_DIFF 🧐



This solution works well if you make Database Links between your Autonomous instances.

Take a look at the differences in our tables and push them



```
1 select dbms_metadata_diff.compare_alter(object_type => 'TABLE',
2                                         name1       => 'EBA_DEMO_APPR_APPROVERS',
3                                         network_link1 => 'TMAPEX_DB_LINK',
4                                         name2       => 'EBA_DEMO_APPR_APPROVERS',
5                                         schema2     => 'TMAPEX')
6 from dual;
```



```
ALTER TABLE "TMAPEX"."EBA_DEMO_APPR_APPROVERS"
ADD ("MAX_SALARY" NUMBER);
```



Take a look at the differences in our tables and push them

```
1 declare
2   l_sql      varchar2(32000 CHAR);
3   l_output   varchar2(32000 CHAR);
4   repoHandle clob;
5   repoCredential varchar2(50 CHAR) := 'GITHUB_CRED';    -- Name of the previously created credential
6   repoName    varchar2(50 CHAR) := 'APEX_WORLD_2024';   -- Name of the GitHub Repository
7   repoOwner   varchar2(50 CHAR) := 'therwix';          -- Name of the GitHub Repository Owner
8 begin
9   l_sql := 'select dbms_metadata_diff.compare_alter(
10              object_type      => ''TABLE'',
11              name1            => ''EBA_DEMO_APPR_APPROVERS'',
12              network_link1    => ''TMAPEX_DB_LINK'',
13              name2            => ''EBA_DEMO_APPR_APPROVERS'',
14              schema2          => ''TMAPEX'')
15          from dual';
16   execute immediate l_sql into l_output;
17
18   repoHandle := dbms_cloud_repo.init_github_repo(
19     credential_name => repoCredential,
20     repo_name       => repoName,
21     owner           => repoOwner
22   );
23
24   dbms_cloud_repo.put_file(
25     repo      => repoHandle,
26     file_path => 'install_alter.sql',
27     contents  => utl_raw.cast_to_raw(l_output),
28     branch_name => 'main',
29     commit_details => json_object('message' value 'DBMS_CLOUD_REPO commit',
30                                  'author'   value 'therwix',
31                                  'email'    value 'timo.herwix@mt-ag.com'
32     )
33   );
34 end;
35 /
```

1

Introduction

2

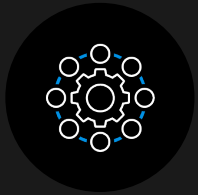
Let's dive deeper

3

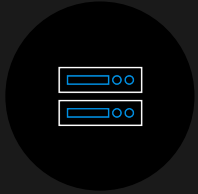
Wrap-up

Wrap-up

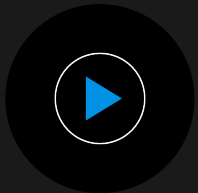
In conclusion, the DBMS_CLOUD_REPO package simplifies CI/CD workflows for APEX applications on Autonomous Database by providing a single interface for managing Code Repositories.



Managing branches and repositories



Exporting database schemas



Executing SQL scripts directly



Wrap-up

However, it is only intended as a simple solution and cannot replace more complex procedures.

It's perfect for:



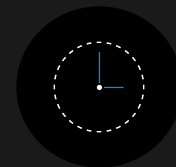
Smaller projects



Limited budgets



When expertise is limited



When time is tight



Blog



Scan me!

A screenshot of the ā'pěks blog website. The header features the site logo 'ā'pěks (#orclapex)' and navigation links: Home, Cloud Computing, Secure Access Made Easy, APEX Tips, Other Series, About Us, and Newsletter. The main content area displays a featured article titled 'Start Generating Dynamic PDFs in APEX Easily with OCI Pre-Built Functions' by Timo Herwix, with a 12-minute read time. To the right, there are two smaller article teasers: 'Implementing Social Sign-In with Microsoft Azure/Office 365 in APEX' (5 min read) and 'Mastering Identity Lifecycle Management for OCI IAM and Azure AD' (4 min read). The footer contains a list of hashtags: #oracle, #orclapex, #lowcode, #plsql, #JavaScript, #AutonomousDatabase, and #Cloud.

Are you interested?



Timo Herwix
Senior Consultant

Telefon: +49 2102 30 961-0
Mobil: +49 176 20185455
Mail: timo.herwix@hyand.com



Timo Herwix



@Therwix



tm-apex.hashnode.dev