



Touchless Deployment

User Guide

(HANA and Oracle)

THE BEST RUN 

Overview	3
About Liquibase	3
Enabling Touchless Deployment in Commissions	4
Touchless Deployment Rest Service	6
Testing Connectivity and Authorizations	6
Deploying Schema Changes	6
Get Deployment Status.....	7
Tagging Database Changelog	9
RollbackToTag	10
Rollback Last Changesets	11
Rollback to Date	12
Creating and Uploading the Liquibase Changelog	13
Best Practices.....	14
Best Practices for XML changelog	14
Best Practices for SQL changelog	15
Touchless Deployment Client / Shell Command	16
Downloading the TDClient.sh Shell Command	16
Pre-requisite to run the TDClient.sh Shell Command.....	16
How To Setup/Environment Variables	17
Using ./TDClient.sh.....	17
Additional parameters.....	18
Examples.....	18
CI/CD Integration with GitHub	19
GitHub Project Structure of Sample Repository	19
Set Up GitHub Repository for ./TDClient.sh Usage	19
View CI/CD Action Log via GitHub Actions	20

Overview

Touchless Deployment is a self-service capability that fully empowers customers to develop and deploy custom code in the EXT Schema. Customers can automatically deploy database changes into the EXT schema in the production and non-production environments, enabling them to continuously test and deploy database schema changes in their tenant. This option makes database refactoring much simpler for customers:

- Database changes can be deployed automatically without any manual interventions (no need to create service tickets).
- Users can easily develop and deploy tables, sequences, procedures, views, and more.
- The deployment starts immediately (if no pipelines are running).
- All deployment details and errors are accessible to enable troubleshooting in case the deployment fails.

Touchless Deployment service for EXT schema is based on an open-source tool – **Liquibase**.

Note: Deploying changes related to registering a stage hook in a pipeline or in the Commissions Data Loader would still require customers to create a support ticket. Other schema changes can be deployed automatically using the Touchless Deployment option.

About Liquibase

Liquibase is an open-source tool that can be used for database refactoring.

Liquibase lists all the changes made to the database in a sequential manner. The changes are recorded in a changelog in the order in which they need to be deployed. Liquibase then uses the changelog to inspect and execute the updates that are not yet applied to the database. Customers need to upload the changelog (Liquibase formatted change log - zip file) and SPM will use Liquibase to execute the changes.

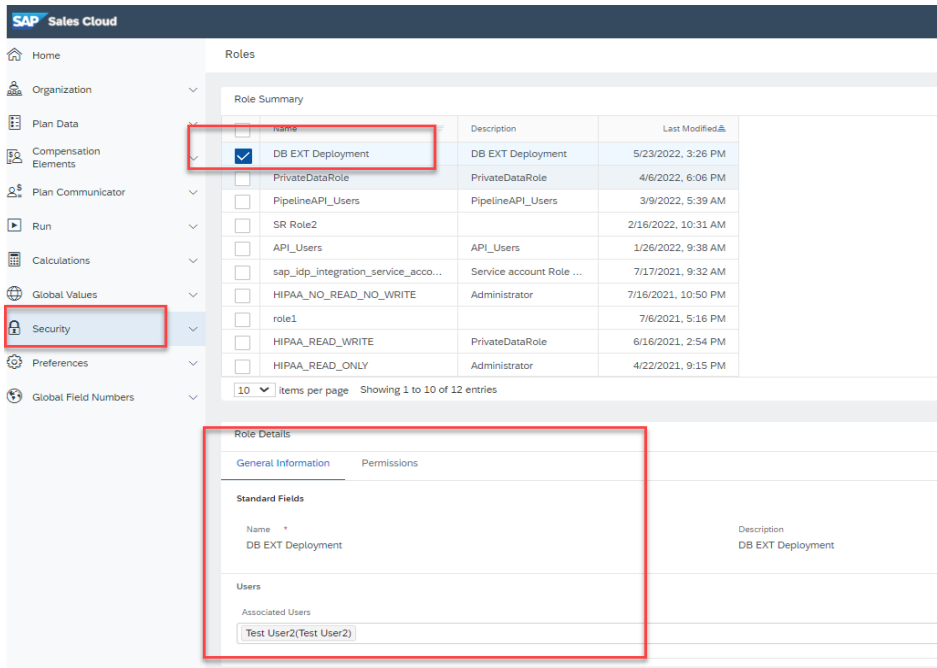
Enabling Touchless Deployment in Commissions

The Touchless Deployment feature is in pilot phase and is currently disabled by default. To enable this feature, create a support ticket. Support will enable the Touchless Deployment Rest Service.

The Touchless Deployment permission is added in **Security > Roles > Administration**. The Commissions user who deploys the database changes must be granted with this permission (through a role).

To configure permissions for Touchless Deployment:

1. In **Manage Setup > Security > Roles**, select the Role that has the permissions to deploy database changes.



The screenshot shows the SAP Sales Cloud interface. On the left, the 'Security' menu item is highlighted. The main area displays a table of roles. The role 'DB EXT Deployment' is selected, and its details are shown in a pop-up window. The 'Permissions' tab is active, showing the 'Touchless Deployment (Update)' permission assigned to the role.

Role Summary	Description	Last Modified
☒ DB EXT Deployment	DB EXT Deployment	5/23/2022, 3:26 PM
☐ PrivateDataRole	PrivateDataRole	4/6/2022, 6:06 PM
☐ PipelineAPI_Users	PipelineAPI_Users	3/9/2022, 5:39 AM
☐ SR Role2		2/16/2022, 10:31 AM
☐ API_Users	API_Users	1/26/2022, 9:38 AM
☐ sap_idp_integration_service_acco...	Service account Role ...	7/17/2021, 9:32 AM
☐ HIPAA_NO_READ_NO_WRITE	Administrator	7/16/2021, 10:50 PM
☐ role1		7/6/2021, 5:16 PM
☐ HIPAA_READ_WRITE	PrivateDataRole	6/16/2021, 2:54 PM
☐ HIPAA_READ_ONLY	Administrator	4/22/2021, 9:15 PM

Role Details

General Information Permissions

Standard Fields

Name DB EXT Deployment Description DB EXT Deployment

Users

Associated Users

Test User2(Test User2)

2. In the **Permissions** tab, assign the **Touchless Deployment (Update)** permission to the role.

Plan Data
Compensation Elements
Plan Communicator
Run
Calculations
Global Values
Security
Preferences
Global Field Numbers

Name	Description	Last Modified
☒ DB EXT Deployment	DB EXT Deployment	5/23/2022, 3:26 PM
☐ PrivateDataRole	PrivateDataRole	4/6/2022, 6:06 PM
☐ PipelineAPI_Users	PipelineAPI_Users	3/9/2022, 5:39 AM
☐ SR Role2		2/16/2022, 10:31 AM
☐ API_Users	API_Users	1/26/2022, 9:38 AM
☐ sap_idp_integration_service_acc...	Service account Role ...	7/17/2021, 9:32 AM
☐ HIPAA_NO_READ_NO_WRITE	Administrator	7/16/2021, 10:50 PM
☐ role1		7/6/2021, 5:16 PM
☐ HIPAA_READ_WRITE	PrivateDataRole	6/16/2021, 2:54 PM
☐ HIPAA_READ_ONLY	Administrator	4/22/2021, 9:15 PM

10 items per page Showing 1 to 10 of 12 entries

Role Details

General Information
Permissions

ADMINISTRATION
ORGANIZATION
PLAN
Rule Element
Data Type
Classification
Logs
Models
CALCULATION
Stage Tables
Communicator

	Read	Workspace	Create	Update	Delete	Assign	Configure
Users	☐	☐	☐	☐	☐		
Roles	☐	☐	☐	☐	☐		
Customizations		☐					
Calendars		☐	☐	☐	☐		
Event Calendars		☐	☐	☐	☐		
Business Units		☐	☐	☐		☐	
Processing Units		☐	☐	☐			
Audit History	☐						
Public Searches	☐		☐	☐	☐		
System Preferences							☐
External Object		☐	☐	☐	☐	☐	
Dashboards	☐	☐	☐	☐	☐		
Private Data	☐		☐	☐	☐		
Touchless Deployment	☐			☐			

Touchless Deployment Rest Service

Testing Connectivity and Authorizations

Operation STATUS provides a basic test if the service is enabled:

GET <https://<host:port>/TrueComp-SaaS/services/rest/touchlessdeployment/status>

If the service is not enabled the result of the GET request will be 500 Internal Error.

If the user is not authorized to run TD the HTTP response will be 403.

Deploying Schema Changes

The sample screenshots below illustrate the Postman REST API client. You can also use other API clients that you prefer.

To deploy the schema changes:

1. Use POST <https://<host:port>/TrueComp-SaaS/services/rest/touchlessdeployment/update>

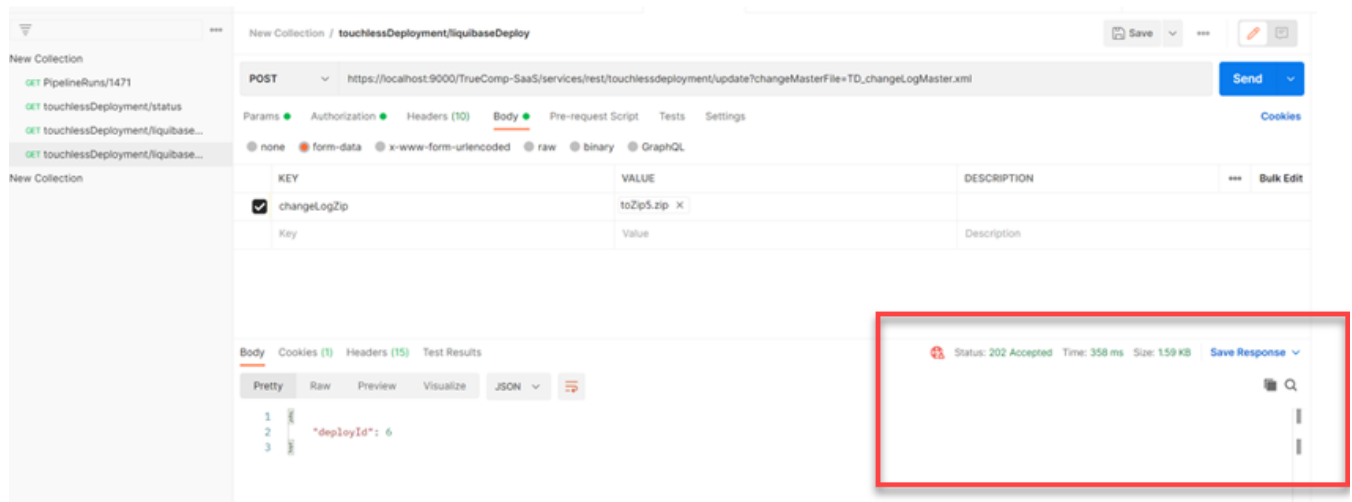
It requires two input parameters:

- Attachment: Upload the **changelog Zip** with the key **changeLogZip**. The changelog Zip is a zipped archive which contains the entire Liquibase changelog.
- URL Parameter: **changeMasterFile** is the name of the Liquibase master file in the archive.

This operation will verify if the **changelog Zip** is a valid zip archive (will extract it on the file system) and will check if the file changeMasterFile exists. If any of this fails, the rest service will return HTTP ERROR CODE 406 (Not accepted).

If successful, TouchlessDeployment will schedule the Liquibase Update Task asynchronously and it will return HTTP CODE 202 (Accepted).

The body will contain a JSON Response with a deploymentid:



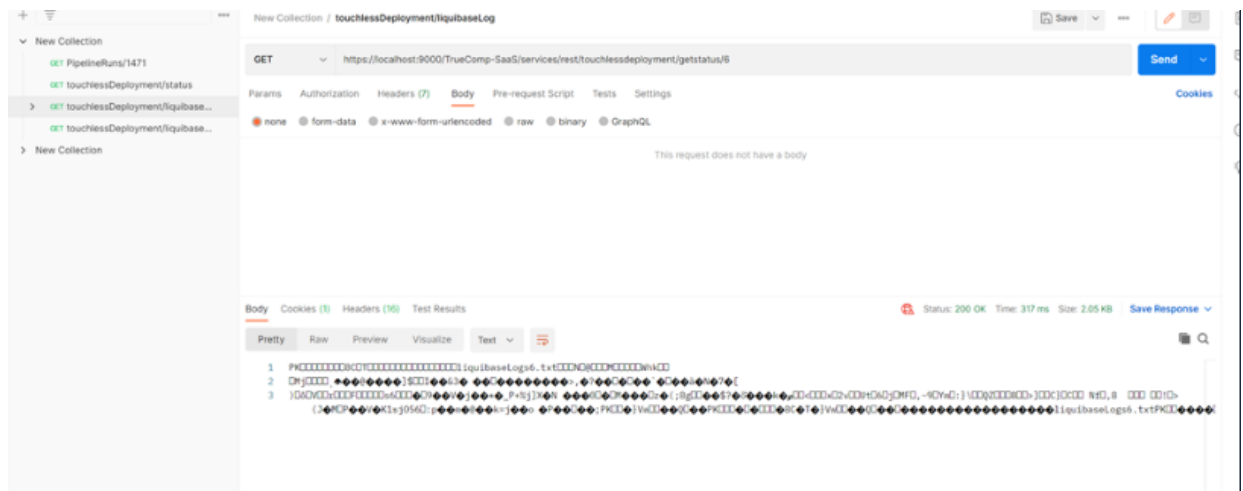
Get Deployment Status

To get the result of the Liquibase Update Task, use the Get_Status endpoint:

GET <https://<host:port>/TrueComp-SaaS/services/rest/touchlessdeployment/getstatus/<deploymentid>>

The possible responses are:

- HTTP 404 NOT FOUND – Liquibase deployment has not completed deployment yet.
- HTTP 200 (Liquibase Success) – This will contain a file attached in the body with a file
- liquibaseLogs<deploymentid>.zip – This archive contains the Liquibase execution logs.
- HTTP 400 (Liquibase Failed) – This will contain an attachment with a zipped Liquibase execution log.



Note: The [Get Status](#) endpoint can also be used to verify if the service is enabled and if the user is authorized to use it.

Save the response. The Liquibase change log shows if all the changes were executed successfully. You can view the EXT Schema to verify that the changes are executed.

Liquibase Change Log Example:

```
liquibaseLogs101.txt - Notepad
File Edit Format View Help
May 04, 2022 1:25:09 PM liquibase.ext
INFO: Set default schema name to EXT
May 04, 2022 1:25:11 PM liquibase.lockservice
INFO: Successfully acquired change log lock
May 04, 2022 1:25:13 PM liquibase.changelog
INFO: Reading from EXT.CS_TD_EXT_DATABASECHANGELOG
May 04, 2022 1:25:15 PM liquibase.changelog
INFO: Custom SQL executed
May 04, 2022 1:25:16 PM liquibase.changelog
INFO: ChangeSet tdchanges/SQLChange_6_error.sql::sql_16::JBLAGOEV ran successfully in 405ms
May 04, 2022 1:25:16 PM liquibase.lockservice
INFO: Successfully released change log lock
May 04, 2022 1:25:17 PM com.callidus.rs.touchlessdeployment.LiquibaseProcessor updateTenantDB
INFO: Liquibase execution failed : liquibase.exception.MigrationFailedException: Migration fail
change set tdchanges/SQLChange_6_error.sql::sql_17::JBLAGOEV:
Reason: liquibase.exception.DatabaseException: SAP DBTech JDBC: [288] (at 20): cannot use
table name: COMPANY_SQL: line 1 col 21 (at pos 20) [Failed SQL: (288) create column table COMPA
INT PRIMARY KEY, name VARCHAR(50), address1 VARCHAR(50), address2 VARCHAR(50), city VARCHAR(30)
```

Tagging Database Changelog

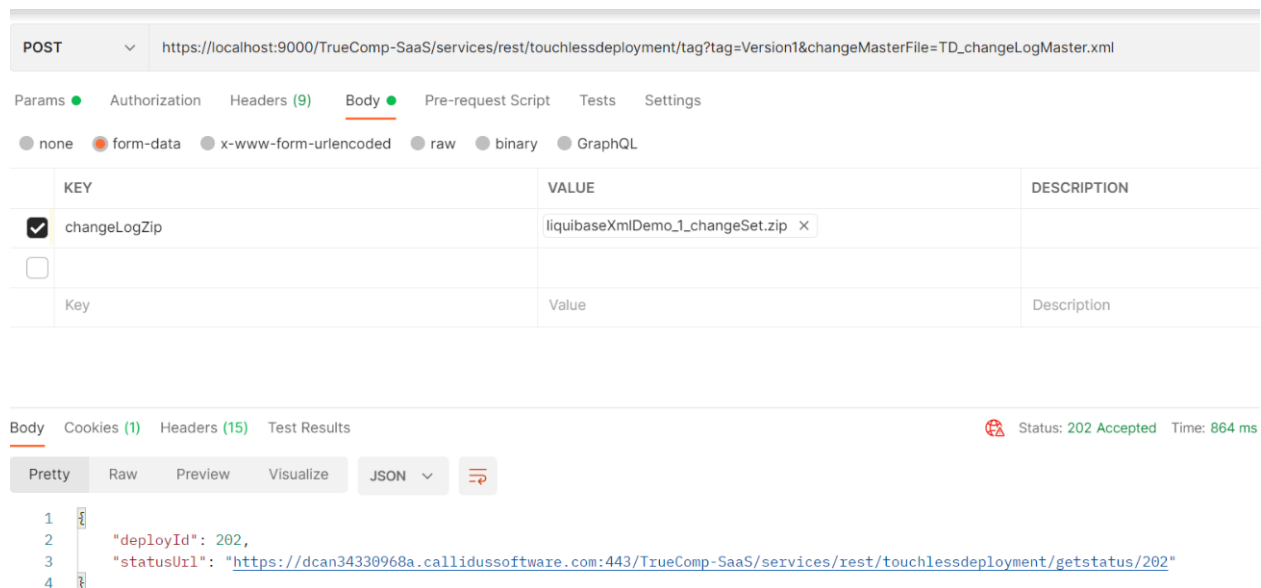
POST <https://<host:port>/TrueComp-SaaS/services/rest/touchlessdeployment/rollbackToTag>

Liquibase allows creating a tag on a changeset. The tag command enables you to mark the current database state so you can roll back changes in the future. It is recommended to use this operation to tag stable releases **before** running additional code changes. See [Liquibase Tag | Liquibase Docs](#) and [RollbackToTag](#) for more details.

The endpoint requires the same parameters as Update + tag.

- changeLogZip - http attachment representing the full changelog as a zip file (same as update)
- changeMasterFile - URL parameter indicating the change log master file inside the zip (same as update)
- tag - URL parameter which indicates the TAG

The result is the same as the UPDATE endpoint. It provides the deployId and the result of the tag must be polled through the [Get Status](#) endpoint.



POST https://localhost:9000/TrueComp-SaaS/services/rest/touchlessdeployment/tag?tag=Version1&changeMasterFile=TD_changeLogMaster.xml

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION
☒ changeLogZip	liquibaseXmlDemo_1_changeSet.zip x	
☐		
Key	Value	Description

Body Cookies (1) Headers (15) Test Results

Status: 202 Accepted Time: 864 ms

Pretty Raw Preview Visualize JSON

```

1 {
2   "deployId": 202,
3   "statusUrl": "https://dcan34330968a.callidussoftware.com:443/TrueComp-SaaS/services/rest/touchlessdeployment/getstatus/202"
4 }
```

RollbackToTag

POST <https://<host:port>/TrueComp-SaaS/services/rest/touchlessdeployment/rollbackToTag>

RollbackToTag endpoint allows you to rollback all changes which have been done after the TAG.

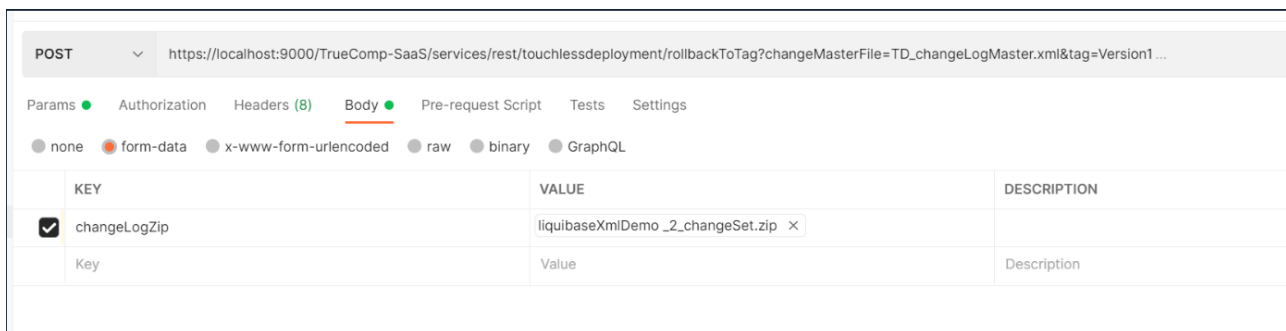
The endpoint requires the same parameters as Tag.

- changeLogZip - http attachment representing the full changelog as a zip file (same as update)
- changeMasterFile - URL parameter indicating the change log master file inside the zip (same as update)
- tag - URL parameter which indicates the TAG

Notes:

- Not all changesets can be rolled back.
- For SQL changesets you should always define ROLLBACKSQL
- For XML changesets - some of these provide automatic generation of rollback SQLs and others require explicit rollback definition. For more details, see [Liquibase Auto Rollback | Liquibase Docs](#).

The result is the same as the UPDATE endpoint. It provides the deployId and the result of the tag must be polled through the [Get_Status](#) endpoint.



POST https://localhost:9000/TrueComp-SaaS/services/rest/touchlessdeployment/rollbackToTag?changeMasterFile=TD_changeLogMaster.xml&tag=Version1 ...

Params ☒ Authorization Headers (8) **Body** ☒ Pre-request Script Tests Settings

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

KEY	VALUE	DESCRIPTION
☒ changeLogZip	liquibaseXmlDemo _2_changeSet.zip x	
Key	Value	Description

Rollback Last Changesets

POST <https://<host:port>/TrueComp-SaaS/services/rest/touchlessdeployment/rollbackLastChangesets>

This endpoint allows you to roll back the last number of changesets (the number is provided as a parameter).

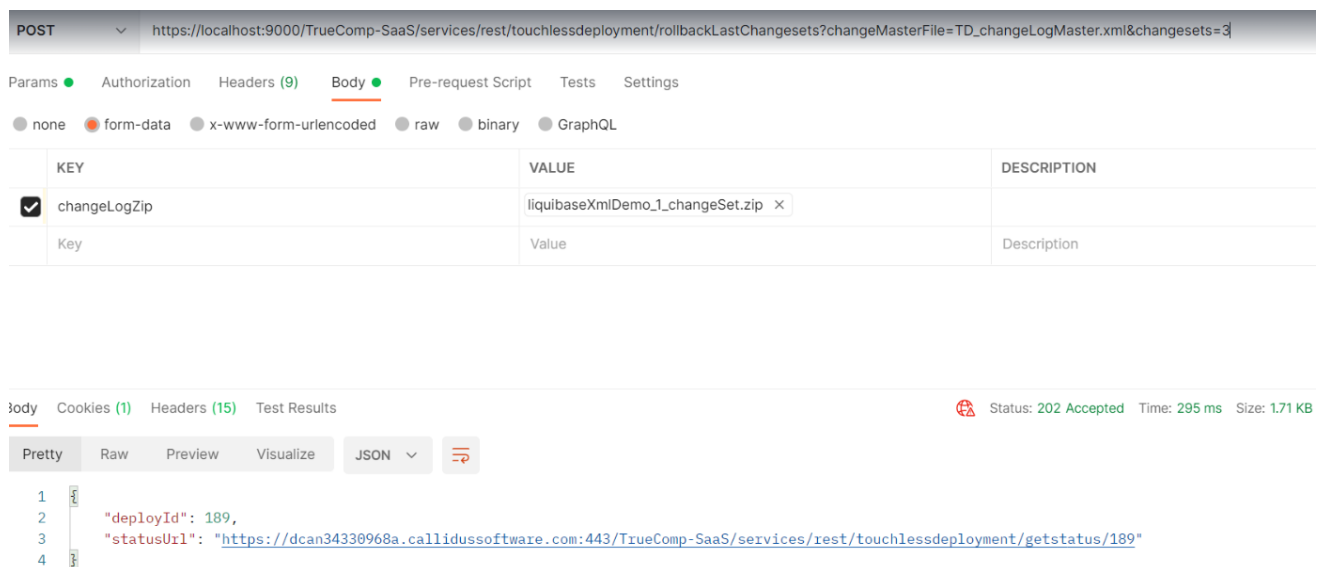
The endpoint requires the following parameters

- changeLogZip - http attachment representing the full changelog as a zip file (same as update)
- changeMasterFile - URL parameter indicating the change log master file inside the zip (same as update)
- changesets - URL parameter which indicates the number of changesets to be rolled back

Notes:

- Not all changesets can be rolled back.
- For SQL changesets you should always define ROLLBACKSQL
- For XML changesets - some of these provide automatic generation of rollback SQLs and others require explicit rollback definition. For more details see [Liquibase Auto Rollback | Liquibase Docs](#)

The result is the same as the UPDATE endpoint. It provides the deployId and the result of the tag must be polled through the [Get Status](#) endpoint.



The screenshot shows a REST client interface with a POST request to `https://localhost:9000/TrueComp-SaaS/services/rest/touchlessdeployment/rollbackLastChangesets?changeMasterFile=TD_changeLogMaster.xml&changesets=3`. The request body is a form with a checked checkbox for `changeLogZip` and a value of `liquibaseXmlDemo_1_changeSet.zip`. The response status is `202 Accepted` with a time of `295 ms` and size of `1.71 KB`. The response body is a JSON object:

```

1 {
2   "deployId": 189,
3   "statusUrl": "https://dcan34330968a.callidussoftware.com:443/TrueComp-SaaS/services/rest/touchlessdeployment/getstatus/189"
4 }
```

Rollback to Date

POST <https://<host:port>/TrueComp-SaaS/services/rest/touchlessdeployment/rollbackToDate>

This endpoint allows the user to rollback all changesets deployed after a particular date

The endpoint requires the following parameters

- changeLogZip - http attachment representing the full changelog as a zip file (same as update)
- changeMasterFile - URL parameter indicating the change log master file inside the zip (same as update)
- date - URL parameter which indicates a date for all changes after to be rolled back. The date must be in format YYYY-MM-DD or YYYY-MM-DD, hh-mm-ss

Notes:

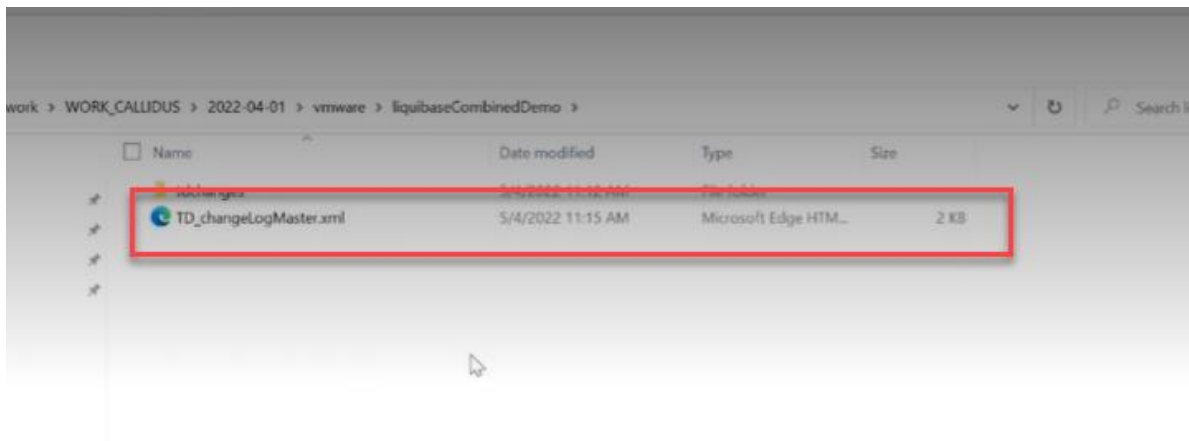
- Not all changesets can be rolled back.
- For SQL changesets you should always define ROLLBACKSQL
- For XML changesets - some of these provide automatic generation of rollback SQLs and others require explicit rollback definition. For more details see [Liquibase Auto Rollback | Liquibase Docs](#)

The result is the same as the UPDATE endpoint. It provides the deployId and the result of the tag must be polled through the [Get_Status](#) endpoint.

POST	https://localhost:9000/TrueComp-SaaS/services/rest/touchlessdeployment/rollbackToDate?changeMasterFile=TD_changeLogMaster.xml&date=2022-05-12,19:52:00		
Params	Authorization	Headers (8)	Body
Query Params			
	KEY	VALUE	DESCRIPTION
☒	changeMasterFile	TD_changeLogMaster.xml	
☒	date	2022-05-12,19:52:00	
	Key	Value	Description

Creating and Uploading the Liquibase Changelog

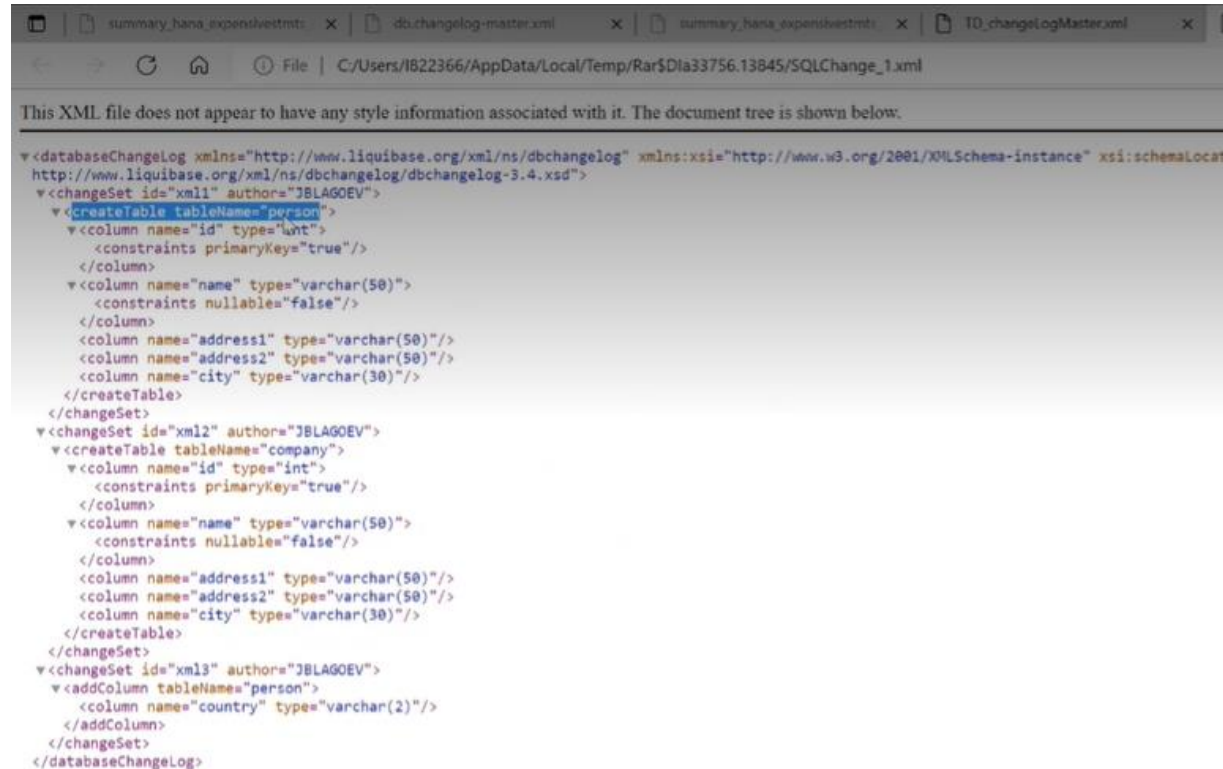
Create the change log master file according to Liquibase requirements. The structure of the change log file must meet the standards defined by Liquibase. See [Liquibase Changelogs](#) for more details.



The master file may include additional XML or SQL files as needed. The structure of the referenced XML and SQL files must also meet the standards defined by Liquibase.



For example, the XML may contain details in Liquibase format for creating a table:



```

<?xml version="1.0" encoding="UTF-8"?>
<databaseChangeLog xmlns="http://www.liquibase.org/xml/ns/dbchangelog" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.4.xsd">
  <changeSet id="xml1" author="JBLAGOEV">
    <createTable tableName="person">
      <column name="id" type="int">
        <constraints primaryKey="true"/>
      </column>
      <column name="name" type="varchar(50)">
        <constraints nullable="false"/>
      </column>
      <column name="address1" type="varchar(50)"/>
      <column name="address2" type="varchar(50)"/>
      <column name="city" type="varchar(30)"/>
    </createTable>
  </changeSet>
  <changeSet id="xml2" author="JBLAGOEV">
    <createTable tableName="company">
      <column name="id" type="int">
        <constraints primaryKey="true"/>
      </column>
      <column name="name" type="varchar(50)">
        <constraints nullable="false"/>
      </column>
      <column name="address1" type="varchar(50)"/>
      <column name="address2" type="varchar(50)"/>
      <column name="city" type="varchar(30)"/>
    </createTable>
  </changeSet>
  <changeSet id="xml3" author="JBLAGOEV">
    <addColumn tableName="person">
      <column name="country" type="varchar(2)"/>
    </addColumn>
  </changeSet>
</databaseChangeLog>

```

Best Practices

Best Practices for XML changelog

Official documentation and examples for Liquibase XML changelog are provided at [Example Changelogs: XML Format](#).

Additionally, we recommend the following:

1. For every changeset, use `failOnError="true"`. This will provide a proper indication that Liquibase has failed on some of the generated queries. It also gives you full control of what was executed and what was not together with a restart point. In rare cases, customers may use the tag `failOnError="false"`.
2. For large SQL code in procedures, libraries, views, etc., customers can include the details in the SQL file and reference it from the XML file.

3. Most of the XML changesets come with [auto-rollback options](#), but some of changesets like create-procedure do not support auto rollback. Nevertheless, the auto rollback is one of the main advantages of XML changesets over SQL changesets. Wherever auto rollback is available, we recommend using an XML changeset. If auto rollback is not available, it is best if customers always specify the rollback commands.
4. Customers may combine many SQLs into a single Liquibase changeset. All such SQLs in a single changeset will be in one transaction. For example, if the second SQL fails, the first SQL will be automatically rolled back. If a customer uses many SQLs in a single changeset, rollback options for all SQLs must be provided.
5. XML changesets generate automatic descriptions. Additional comments may be provided to clarify the changes.

Best Practices for SQL changelog

Documentation and examples for Liquibase SQL formats are provided at [Example Changelogs: SQL Format](#).

Additionally, we recommend the following:

1. For every changeset, use `failOnError=true`. In rare cases, customers may switch to `failOnError=false` (similar to the XML case).
2. SQL changesets do not provide auto-rollback options. For every changeset, it is best to provide rollback options.
3. Similar to the XML case, a customer may combine many SQLs into a single changeset. All SQLs will run in a single transaction. Make sure to provide rollback options for all SQLs in the changeset.
4. SQL changesets do not generate automatic description (the auto-description is simply SQL). Make sure to provide clear comment for every changeset. These comments will be visible in table `CS_TD_EXT_DATABASECHANGELOG`.

Touchless Deployment Client / Shell Command

TDClient.sh is a shell script which invokes the Touchless Deployment rest service.

The purpose of the shell script is to simplify and unify the integration of the customer's Continuous Integration/Continuous Delivery (CI/CD) pipeline and to simplify the process for Sales Performance Management administrators/developers.

Note: Continuous integration and continuous deployment (CI/CD) methodologies enable software release process to be automated.

The script will mostly be used for integration with CI/CD to invoke an operation update.

In few cases, it can be used by SPM Administrators or Developers to manually tag or rollback the changes.

For most of the use cases, customers must create a zip archive, post the update endpoint, and then start polling the getDeploymentStatus. The shell script helps to integrate these three steps into one. The same benefits apply to other endpoints, such as tag, rollbackToTag, and rollbackLastChangesets.

Downloading the TDClient.sh Shell Command

Customers can download the zip from the SAP Help Portal: Click [TD_Client.zip](#).

The zip contains:

- a bin directory with a main script `TDClient.sh`
- a lib directory which contains two jar files

Pre-requisite to run the TDClient.sh Shell Command

As a pre-requisite to run the TDClient.sh shell command, customers must have:

- install java (rel $\geq$ 8)
- set up the command `./java` to execute from the TDClient.sh script (JRE/bin or JDK/bin has to be part of the PATH environment).

How To Setup/Environment Variables

The client requires three mandatory environment variables (connectivity to SPM):

- export TD_BASEURL="https://...."
- export TD_USERNAME="<user>"
- export TD_PASSWORD="<password>"

Additionally, one optional environment variable can be used. It points to a directory which is used by TDClient to extract and store log files for customer auditing:

- export TD_LOGDIRECTORY="<dir>"

Using ./TDClient.sh

Usage ./TDClient.sh <arguments>

- help/-h : prints this help message
- operation/-o : Operation to execute. Possible operation values:
 - update
 - getDeployStatus
 - tag
 - rollbackToTag
 - rollbackLastChangesets
- masterfile/-m : change log master xml file.
- dir/-d : directory which contains the full change log. A zip archive will be created out of this directory and will be posted to the server.
- zip/-z : a zip file which contains the full change log. This archive will be posted to the server. Operations update/tag/rollbackToTag/rollbackLastChangesets require either one zip or dir parameter to be provided
- tag/-t : parameter tag required for operation "Tag" or "Rollback To Tag"
- lastchangesets/-l : parameter last changesets required for "Rollback last changesets"
- deployid/-id : parameter deployment id required for operation "Get deployment status"

By default, the modification operations (update/tag/rollbackToTag/rollbackLastChangesets) post the zip archive and then start polling for 30 minutes for the results of the operation.

Additional parameters

- logdirectory/-ld : this directory will be used to download and extract the execution logs for end user detail analysis. If not explicitly provided, the parameter uses the value ./executionLogs. Instead of explicitly providing this parameter, an environment variable TD_LOGDIRECTORY can be bound.
- printLiquibaseLog/-pl : Possible values Yes/No. Default No. If parameter is enabled, it will print the liquibase execution log on the console for all deployment tasks.
- poll_deploy_status : Possible values Yes/No.Default Yes. If value No is provided then modify operation(update/tag/rollbackToTag/rollbackLastChangesets) will not poll for the result and will print only the <deploy id>.
- poll_timeout : Number of minutes allowed for polling of the deployment task.Default 30 minutes.
- poll_retry_interval : Number of seconds to retry polling until success/failure result.Default 10 seconds.
- ignoreSSLCertValidation/-iSSL : Possible values Yes/No.Default No.If set to Yes ignores SSL Certificate validation when establishing an https connection. Should be used only for developer systems.

Examples

```
./TDClient.sh -o update -m changeLogMaster.xml -d /usr/local/changelog
```

Creates a zip archive from directory /usr/local/changelog (this directory must contain the master file changeLogMaster.xml) and will invoke operation UPDATE

```
./TDClient.sh -o update -m changeLogMaster.xml -z /usr/local/changelog.zip
```

Invokes operation UPDATE with the zip archive (this zip file must contain the master file changeLogMaster.xml)

```
./TDClient.sh -o getDeployStatus -id 278
```

Checks the status of deployment task 278.

```
./TDClient.sh -o tag -t STABLE_VERSION_13 -m changeLogMaster.xml -d /usr/local/changelog
```

Creates an archive and invokes operation TAG to tag the EXT schema state to STABLE_VERSION_13

```
./TDClient.sh -o rollbackToTag -t STABLE_VERSION_13 -m changeLogMaster.xml -d /usr/local/changelog
```

Creates an archive and invokes operation ROLLBACKTOTAG to rollback to tag STABLE_VERSION_13

```
./TDClient.sh -o rollbackLastChangesets -l 3 -m changeLogMaster.xml -d /usr/local/changelog
```

Creates an archive and invokes operation ROLLBACKLASTCHANGESETS to rollback the 3 latest

changesets

CI/CD Integration with GitHub

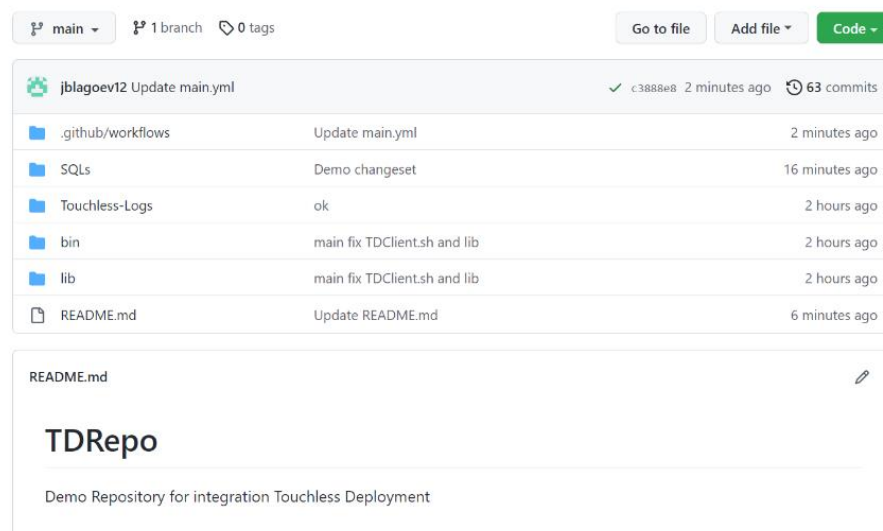
This section describes the sample repository that is set up to demonstrate how CI/CD can be accomplished with `./TDClient.sh`.

Sample Repository: <https://github.com/jblagoev12/TDRepo>

Note: For every merge request on the main branch of this repository, the GitHub Actions will invoke the `./TDClient.sh` which will call the Touchless Deployment Rest Service.

GitHub > GitHub Actions is one of the tools used for Continuous Integration. See <https://docs.github.com/en/actions> for more details.

GitHub Project Structure of Sample Repository



The screenshot displays the GitHub repository interface for `jblagoev12/TDRepo`. At the top, it shows the current branch as `main` with 1 branch and 0 tags. Below this, there are buttons for `Go to file`, `Add file`, and `Code`. The repository structure is listed as follows:

File/Folder	Commit Message	Time Ago
<code>.github/workflows</code>	Update main.yml	2 minutes ago
<code>SQLs</code>	Demo changeset	16 minutes ago
<code>Touchless-Logs</code>	ok	2 hours ago
<code>bin</code>	main fix TDClient.sh and lib	2 hours ago
<code>lib</code>	main fix TDClient.sh and lib	2 hours ago
<code>README.md</code>	Update README.md	6 minutes ago

The `README.md` file is expanded, showing the following content:

```

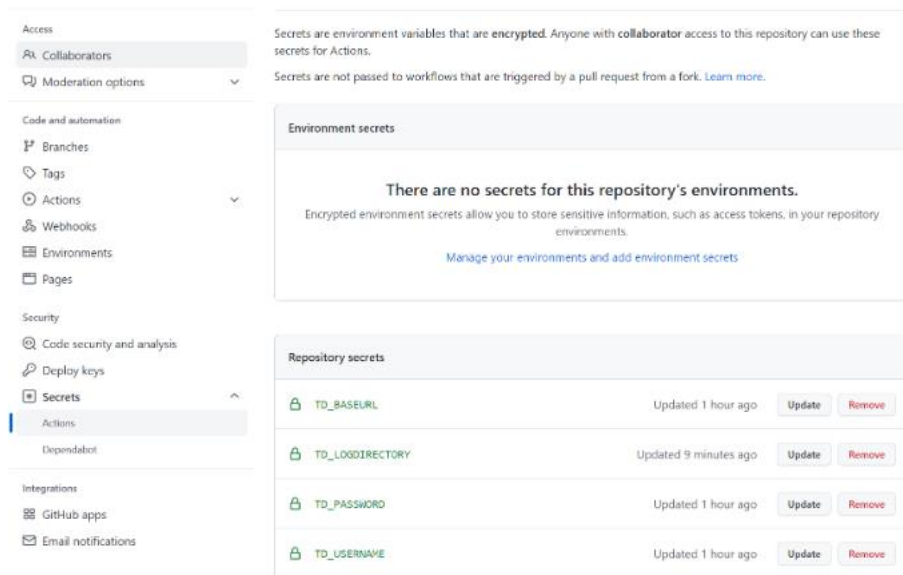
TDRepo

Demo Repository for integration Touchless Deployment
  
```

- `bin` and `lib` are the original folders for Touchless Deployment
- `SQLs` - contains the `master.xml` and all the `sql` files and subdirectories
- `.github/workflow` - contains the `yml.xml` file needed to execute the action on push or pull request

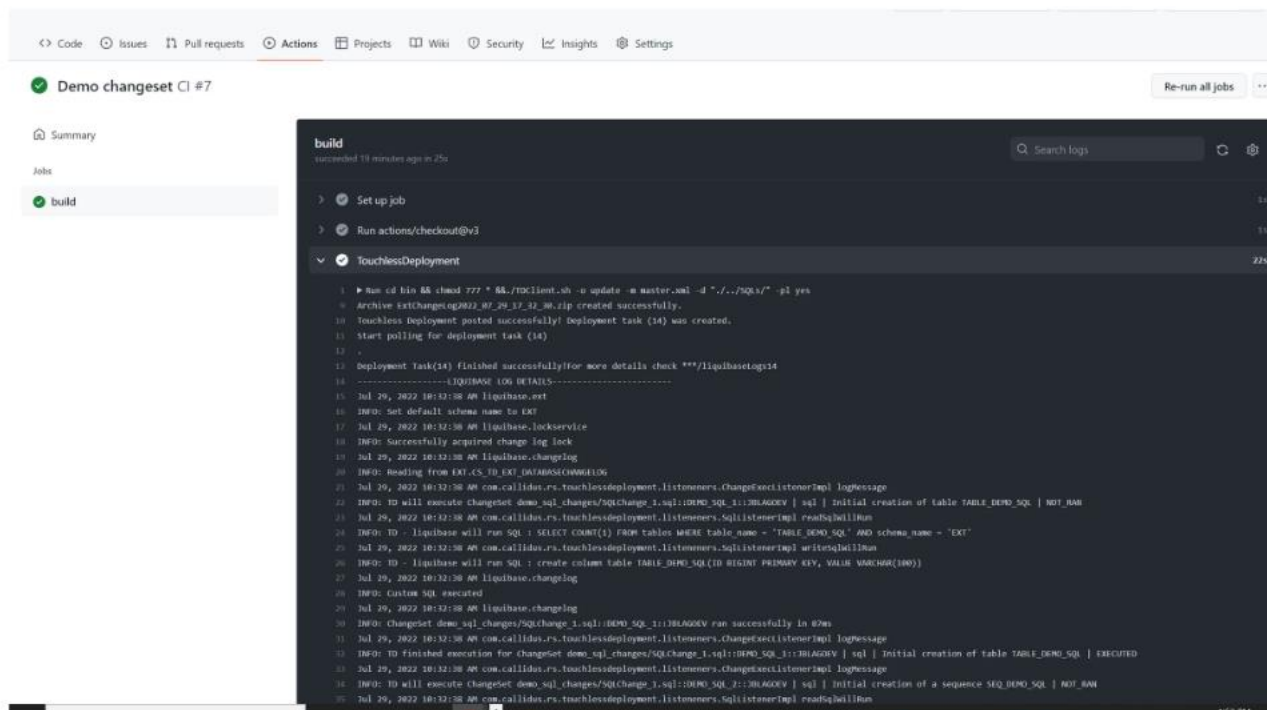
Set Up GitHub Repository for `./TDClient.sh` Usage

To setup the repository, go to **Settings > Secrets > Actions** and prepare the 4 secrets for the corresponding connection parameters.



View CI/CD Action Log via GitHub Actions

To view the action log, use the Push or Pull Requests. Sample actions triggered:



© 2022 SAP SE or an SAP affiliate company. All rights reserved.
No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.
Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliate companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.
SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.
Please see <https://www.sap.com/about/legal/trademark.html> for additional trademark information and notices.