All current Liquibase Reviewers are listed here:
https://github.com/orgs/appian/teams/liquibase-reviewers

How to become a Liquibase Reviewer

1. If you are not familiar with, please review
    - Liquibase related slides from the SE 202: Appian's Primary Datasource
    - Liquibase in Appian
    - Liquibase Recipes at Appian
2. Join the Liquibase Reviewers chat room
3. Shadow review at least three Liquibase PRs. Pay attention to what other reviewers are pointing out. You can take a look at past PRs that have already closed. Following PRs are good examples to review:
    a. *Custom task class example*
    b. *Deleting rows from tables*
    c. *Custom SQL changeset example*
4. When reviewing Liquibase PRs, you only need to cover Liquibase related changes
    a. Liquibase changelog xml or yaml files
    b. Liquibase custom change yaml and class
    c. Migration tests

Minimum Code Review Checklist

1. It is almost **NEVER ok** to modify an existing changelog file. There are exceptions to this rule. If you find yourselves having to modify an existing changelog file, please consult with other Liquibase reviewers to understand the implications.
2. If the migration is to be hotfixed to maint branches, **you MUST use the same filename and changeset ids.**
3. Naming
    a. Make sure all constraints are named

```
<column name="uuid" type="${uuidType}">
   <constraints nullable="false" unique="true" uniqueConstraintName="external_sys_uuid_uc"/>
</column>


<addUniqueConstraint columnNames="uuid" tableName="deployment" constraintName="uuid_uc"/>
```

    b. Make sure all column names are 30 characters or less
        i. This is required by Oracle: http://www.dba-oracle.com/sf_ora_00972_identifier_is_too_long.htm
    c. All identifier names are lowercase_underscore_form
    d. Abbreviations are standard and/or sensible
    e. Table names are singular, Join Tables are plural - justified as plural since each row has >1 detail

    f. Table names not repeated in column names
    g. Column type is not repeated in column name
    h. id is simply id, uuid is uuid
    i. Foreign keys in form _id unless multiple to same table - N/A
    j. FKs end in _fk, Unique constraints in _uc, Indices in _idx
4. Add indexes to the table, especially if there will be frequent queries based on certain columns in the WHERE clause or when sorting. Though keep in mind that adding indexes can slow down insert and update operations on the table.
5. No preconditions avoiding index creation
6. For non-nullable columns, both value and default value populated
7. Ensure rdbms-migration-daily has successfully completed *for **all databases***
8. Migration testing
    a. In general, we default to ensuring each migration has a migration test
    b. The only scenario where a migration test can be excluded is if the migration is adding a table that has no relationship to any other tables (i.e. no fks)
    c. If a test is not included, it requires justification on the PR as to explicitly why it was not included.
    d. For all new indexes, ensure `` `verifyIndexCreationForTable` `` is used to assert the index creation. (e.g. https://github.com/appian/ae/pull/59115/files)
9. **Data Migrations**
    a. **When there is a data migration and an addition of a unique or nonNullable constraint or other types of potentially breaking changes (type changes, etc), we need to verify that error scenarios are covered by the migration test**
10. Make sure there are no mysql/mariadb preconditions that try to avoid index-creation on those dbs, since we don't need them and prefer to avoid that complexity.
11. When adding a non-nullable column, ensure the migration specifies the value for this column for the existing rows

```
- changeSet:
    author: appian
    id: '000386.1.0'
    comment: Add show_header column to portal
    addColumn:
      tableName: portal
      columns:
        - column:
            name: show_header
            type: ${booleanType}
            valueBoolean: false
            defaultValueBoolean: false
            constraints:
              nullable: false
```

Furthermore, Unit test should be added to

    a. populate at least one row in the database with schema prior to this migration.
    b. migrate to the new schema
    c. confirm the new field is populated correctly after loading it using the updated Dao
    d. DataSourceManagerSchemaMigrationTo000384Test.java is a good example
12. JPA
    a. Cascading behavior specified in Liquibase and JPA
    b. JPA @column nullable configuration matches Liquibase
    c. All JPA string @columns set length
    d. JPA names use camelCase
    e. FetchType.LAZY preferred over FetchType.EAGER for lists - N/A
    f. @fetch(FetchMode.SELECT) and @batchsize used when necessary - N/A
    g. Implement toString on Entities, excluding lazily-loaded data

LiquibaseChangelogAnalysisTest Enhancements

LiquibaseChangelogAnalysisTest checks if the certain rules automatically. If this test is failing, you must fix your changelog. There are more rules we can automatically check. If you have ideas, list them below. Better yet, use your indie time to implement them yourself.

(Add your ideas about automating the Liquibase migration checks here)

1.