Apache Airavata Database Versioning:
Getting the Airavata Registry under Version Control

Sneha Tilak, Suresh Marru, Marlon Pierce
Science Gateway Research Center, Indiana University, Bloomington

ABSTRACT

How do you record and preserve a user's cyberinfrastructure sessions?

Gateways are open source software projects where multiple developers from different organizations may be modifying the code base. There is a need for preservation of records as the gateway's codebase evolves effectively. From Legacy systems to newer versions.

We must implement a database versioning tool in Apache Airavata to ensure traceability, visibility, fault-tolerance and convenience.

APACHE AIRAVATA

- Apache Airavata framework supports the execution of computational scientific applications and workflows, grid-based systems, remote clusters and cloud-based systems.
- Registry is used to store metadata about user experiments, computational resources, and scientific applications
- Registry is implemented using OpenJPA over a relational database.

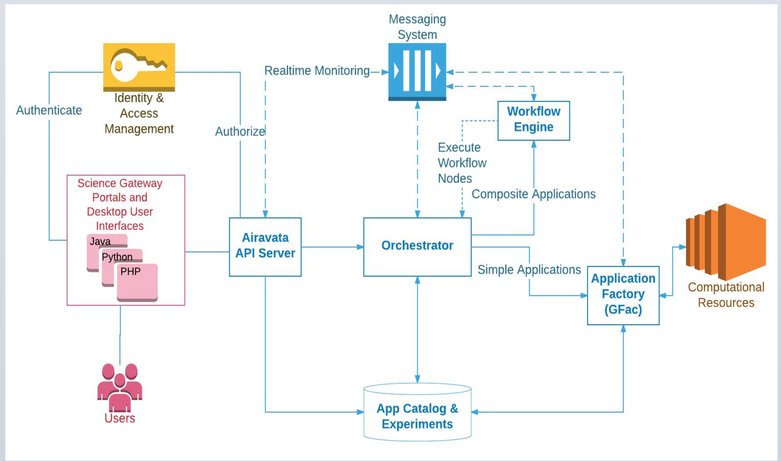

Fig 1: Apache Airavata's conceptual framework

PROBLEM STATEMENT

Challenges:

1. Managing the various versions of the framework.
2. Adapting the Registry to fit in new data structures.
3. Maintaining database schema and the data stored when multiple developers work on the framework.

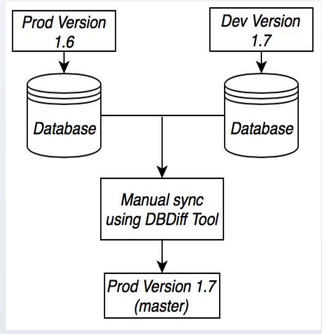

Fig 2: Current scenario in Airavata

Solution:

The most efficient way to manage data in any distributed system is to version it. Database versioning refers to the process of managing and tracking the changes made to a database.

LIQUIBASE

- An open-source library
- Database-independent
- Tracks, manages and applies database schema changes
- Maintains a *changeLog* file to keep track of the changes.
- Can be integrated with Apache Maven (build tool Airavata uses) using a JAR plugin.
- Provides Rollback support in Maven.

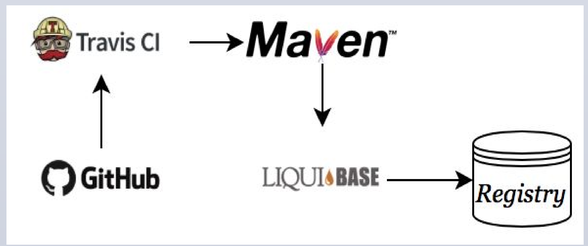

Fig 3: The workflow of a Maven based application using a liquibase-maven-plugin

USE CASE

Database migrations refer to the changes made to a stable database that is already in production. In this case, the developer must perform some operations to conduct the migration.

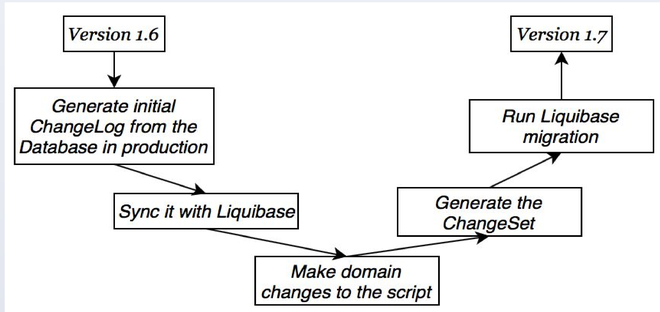

Fig 4: Deploying changes made to the Database using Liquibase

Liquibase maps the old structure of the schema and database to the new structure without us having to make changes manually to the database that needs to be modified.

CONCLUSION

By including the database scripts in the source control, we can maintain an audit trail. Through shared management, it provides better integration with the application code. And lastly, we can automate all our deployments as we need not worry about the conflicts between schemas on the production and development environments.

To learn more about Apache Airavata, get involved and contribute, visit ***www.airavata.apache.org*** *Thanks to our mentors Suresh and Marlon for all their support and guidance throughout this project. This project is supported by the Google Summer of Code 2017 program.*