Automated Testing of Tableau Dashboards

KINESIS
Powered by 360Suite

Abstract

Companies make business-critical decisions every day based on data from their business intelligence systems. It is therefore essential that Business Intelligence (BI) reports display correct data in an efficient way that is interactive and easy to understand for business users. BI reports are usually validated manually, but there are other efficient ways of conducting testing which are more reliable and can reduce testing time. In this whitepaper, we explore alternatives to manual testing of Tableau dashboards and describe how software developers have been relying on automated testing via Continuous Integration and techniques like Test-Driven Development to improve software stability and reduce maintenance costs. We will explore how Kinesis CI supports these practices for Tableau.

Why Test Dashboard Changes?

Business intelligence dashboards constantly evolve as business and reporting needs change. Such changes can potentially alter existing unrelated functionality and cause unintended reporting errors, which is why testing dashboard deployments is essential. In large organizations, it is not uncommon to have a high number of changes committed on any given day to a dashboard or a number of dashboards. These changes can only be effectively validated with automated testing.

We can also use automated testing to ensure that our reports display correct data and, that our data refresh processes and manual modifications are executed correctly either by comparing the results with pre-set values or referencing data in the underlying database. Performing such test tasks in an automated fashion on a Continuous Integration server can greatly reduce costs and speed up delivery.

Background

Modern software development practices encourage making testing a fundamental part of the project. If a software project does not have reasonable test coverage it is considered as “legacy code”: any change to such a project is a potential risk and makes maintenance harder and more costly.

There are a number of software test categories but some of the most commonly used ones are:

- *Unit*
- *Functional*
- *Performance*

Each software test category provides different values: in unit testing, it is possible to write simple and quick tests and validate low-level logic, while functional tests are higher level, take longer to write but give more confidence in the overall correctness of the system. Performance tests can help identify slowdowns and bottlenecks introduced by changes. It is common to use a combination of several test types in a software project to get optimal results.

Ideally, we should be able to use a similar approach and perform different kinds of tests on our Tableau dashboards to gain complete confidence in our changes before promoting the new version to our clients.

Business Intelligence Testing

Drawing parallels between BI dashboard and software testing can be helpful to understand important similarities, but there is a major difference with business intelligence systems: software testing tools and frameworks are typically designed for traditional programming languages and are not suitable for BI projects. Testing Business Intelligence applications is different from testing traditional transactional applications, as it requires a data-centric testing approach and you need to work with third-party BI tools, resulting in limited alternatives for applicable testing tools. In addition, the volume, variety, and complexity of the data make it difficult to create robust test cases, and specialized skills are required to execute the data validation and verification processes.

Despite limited testing options with Tableau, we can still implement a full test suite to validate the accuracy and performance of our dashboards.

We can implement some of the following test types in our BI testing strategy:

- **Regression** to verify new changes don’t impact existing functionality.
- **Functional** to ensure that the delivered changes are in line with requirements.
- **Cross - Environment** to compare different Tableau environments (i.e. sites or servers) to make sure changes can be safely deployed.
- **Performance** to make sure no performance regressions are introduced in our changes.

Testing Tableau Dashboards

Testing is an integral and very important part of most areas of software development. However in BI, it is not yet widespread, despite the fact that BI developers not only visualize data but they also make a lot of data transformations, implementing various business logic into the visualization layer, blending data from multiple sources, etc. This simply results in a very high risk of potentially showing data incorrectly.

BI professionals who recognize the need for testing their BI projects can implement one of the following approaches, or a combination of them.

1. **Manual testing.** Compile a list of test cases that need to be verified after every change. These can be maintained in a simple spreadsheet and updated with new test cases as dashboard development progresses. As this kind of testing does not require any additional software products it is the simplest one to implement. However, it is the slowest and most expensive one to maintain.
2. **Automated testing via Selenium**. Since Tableau Server can be accessed with a web browser, automation can be built with a headless browser using Selenium WebDriver. This means we can automate manual test executions, but it does take a considerable investment to build out the initial automation framework. Adding and maintaining tests requires software development skills depending on how much abstraction our framework does. It is also not possible with Selenium to test data or Tableau-specific functionalities.
3. **Automated testing with specialized Tableau testing tools, like Kinesis CI**. Kinesis CI is a Tableau testing framework that uses headless browser automation with an easy-to-use user interface to set-up and maintains test cases. Kinesis CI allows users to describe dashboard interactions and user journeys in their test cases, and execute the tests using the UI or through the command-line interface. This is so that tests can be driven from continuous integration servers like **Jenkins**.

At present, manual testing is by far the most prevalent. BI analysts usually implement and follow a manual testing process. To follow a manual testing routine, companies are reliant on QA resources, resulting in an increased headcount. Manual testing is also prone to human errors and thus not the most efficient process.

Selenium tests are designed for web applications. However, they cannot deal with the challenges of a data-centric testing approach, where data can be changing at frequent intervals. Writing and maintaining test cases in Selenium is also a lengthy process, creating overhead and requiring skills data analysts and BI developers do not necessarily have.

Specialized Tableau testing tools can efficiently deal with the challenges of changing data, and specific features are designed to provide interactivity for Tableau users.

How Kinesis CI can Help Your Organization

Kinesis CI is a testing tool designed for BI projects to cover testing requirements, including functional, regression, and performance testing, and to implement test-driven development for Tableau dashboards. Kinesis CI offers a versatile and flexible test framework for Tableau that can reduce the high costs of repetitive manual testing by shortening your BI development cycle and increasing the reliability of your BI dashboards.

- **Write tests for your Tableau reports**: using Kinesis Designer, analysts can quickly put together a comprehensive test plan for their dashboards, including functional, regression, cross-environment, and performance testing. Tests can be executed directly within the tool.
- **Use source control for your tests**: as the test cases are described in JSON file format, they can be easily kept in source control systems. Kinesis Designer integrates with Git so users don't need to leave the tool to clone or upload tests.
- **Implement continuous integration**: As reports evolve, new changes can break existing functionality. The Kinesis command-line interface tool can be easily integrated with your organization's continuous integration server, and every dashboard change can trigger a re-run of the entire test suite against the latest dashboard version. This ensures that the changes do not cause problems before the dashboard is published to the live environment.
- **Test Driven Development**: find issues at the earliest possible phase when developing new features.
- **Continuous Delivery**: automatically deploy changes after successful test runs to improve productivity.
- **Scheduled Test Runs**: run the test suite on a regular basis to ensure the Tableau environment is always in a healthy state.

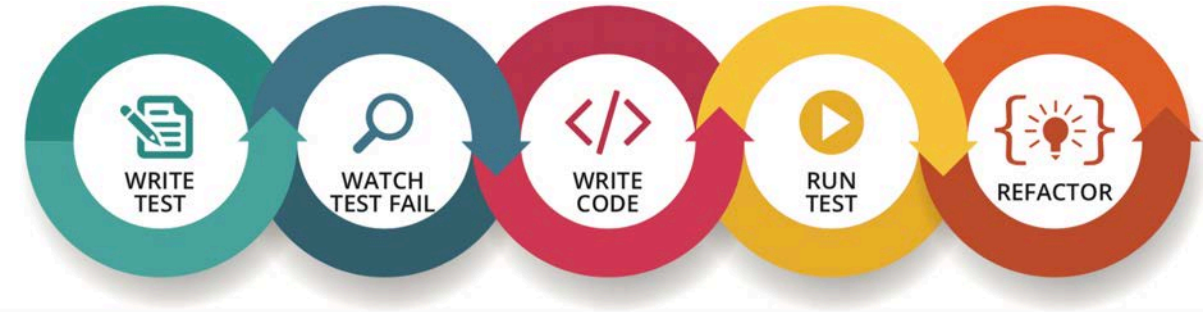
Advantages of Kinesis Over In-house Test Automations:

In order to be able to use the Kinesis CI tool, users do not need special skills other than working knowledge of Tableau. Users can quickly learn how to use Kinesis Designer and start implementing tests. In-house test automation solutions without a simplified user experience usually require training and continuous assistance from the software engineering group.

Cost of ownership: implementing a custom automation solution for Tableau is non-trivial. Taking Selenium as an example to drive test automation, even a minimal solution that can be used by non-technical users will likely require hundreds of man-hours of development time and continuous maintenance to keep up with changes introduced by new Tableau releases. In contrast, the total cost of ownership of a ready-made solution like Kinesis CI will be much less and is practically maintenance-free.

Test-Driven Development

In most areas of software development, adopting CI and other agile methodologies have changed the way people test. Development teams are required to do more testing, faster and more often. Testing is done earlier in the development lifecycle and the focus is on automated testing as developers try to move away from manual testing.



Test-Driven Development (TDD) as a concept is widely used in most areas of software development. Software developers using TDD start with writing test cases, based on what the software or piece of code is expected to do, then write the code, so that the test passes. They then keep repeating these steps on a regular basis.

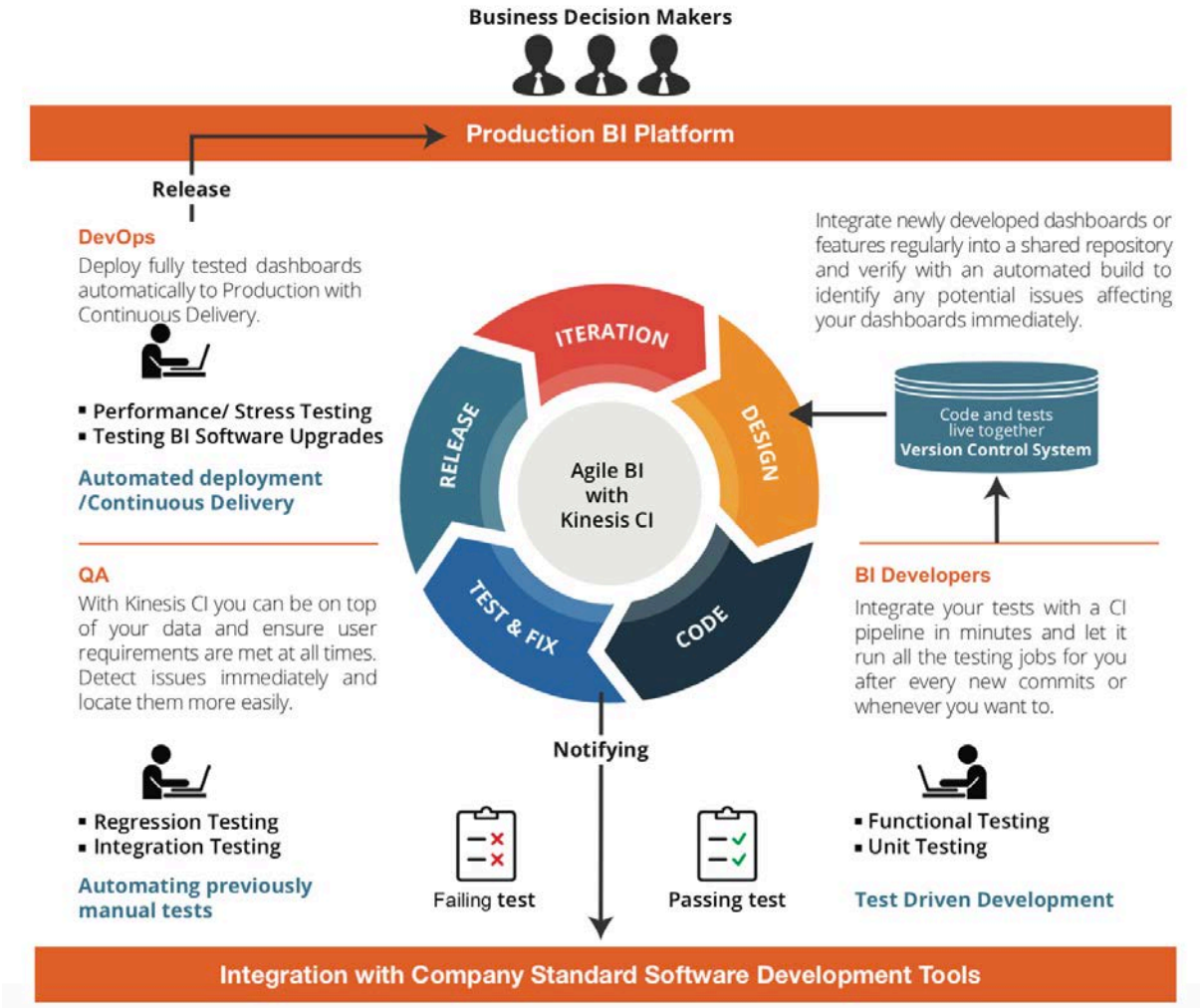
This concept enables developers to find issues at the earliest possible phase when developing new features, so they will know of any issues before merging the changes into the existing codebase. A large number of studies prove that writing tests are crucial when developing software, and will result in a decrease in production issues and of course less headache with fixing bugs.

In Business Intelligence, test-driven development can be implemented with Kinesis CI, as this tool is designed for BI applications and is able to test dashboards with changing data, complex calculation logic, and interactive fields. Kinesis CI is using the Tableau JS/REST APIs and further internal communication methods to interact with the visualizations where web testing frameworks are not efficient. Moreover, it comes with built-in data comparison tasks that are mandatory to validate the figures shown on the dashboard.

Agile BI Development

Agile software development is an approach whereby software solutions evolve through continuous collaboration between teams. It is widespread in modern software development as changes in requirements can be dealt with more efficiently, and results in delivering solutions to end-users faster and more accurately.

With Kinesis CI, users can evolve their BI projects into an agile process. Automated testing can be implemented at all stages of dashboard development and used by various teams, starting with BI developers or data analysts developing functional tests when developing the dashboards themselves. These can be run whenever a change is committed to a dashboard to make sure existing functionality is preserved. Regression tests can be run by QA analysts to keep track of unexpected changes to dashboards. Finally, fully tested dashboards can be deployed automatically to the production environment.

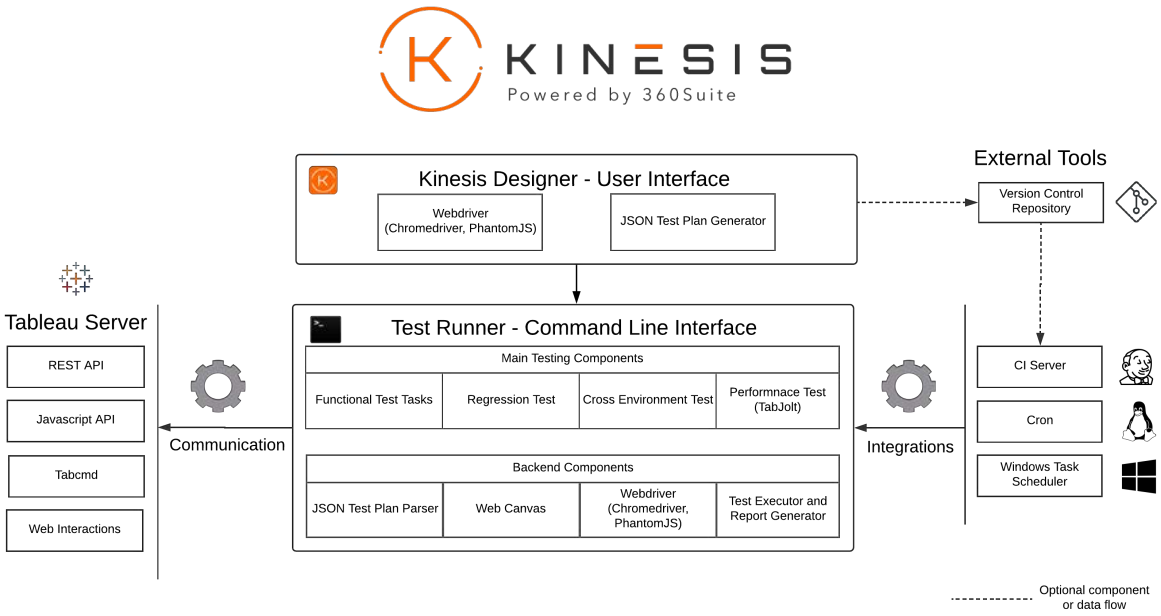


Software Architecture

Tests are created in the User Interface, Kinesis Designer. Kinesis Designer generates a JSON file that includes the test properties necessary for the Command Line Interface to run the test cases.

To set up test cases, i.e. creating snapshots from Tableau dashboards containing metadata, Kinesis Designer is using a web driver.

Tests are run by the Command Line Interface. Communication with Tableau is done by using a combination of REST API, Javascript API, Tabcmd, and web interactions.



Architectural Diagram

Test Types

There are four testing modules within Kinesis CI: functional, regression, cross-environment, and performance testing.

- **Functional Testing** is designed to test different components of Tableau dashboards, such as filters, parameters, or the layout. Users can validate data by a number of criteria. Data can be tested against expected results, based on user-defined rules using flexible formulas. Equally, data displayed on Tableau dashboards can be tested directly against the content of the underlying data source at the moment of running the test case. Users can simulate user journeys and test every step of these journeys in Kinesis CI. Functional testing is highly modular and flexible.
- **Regression Testing** enables users to compare the actual status of a Tableau dashboard to a baseline of the same Tableau dashboard taken earlier from Tableau Server to verify unexpected changes. Any future test runs will be compared to this baseline.
- **Cross-Environment Testing** enables users to compare the same dashboard on two different Tableau environments at the same time.
- **Performance Testing** drives load to your Tableau server and assesses response times based on SLA requirements. You can work with multiple concurrent users and dashboards. Performance testing in Kinesis CI is based on the Apex score, which is commonly used for web applications.

After every test run, a report which gives an overview of the test case and a detailed illustration for any failures is generated. This enables users to easily identify and fix issues before they get escalated or cause bigger problems down the line.

Functional Testing

Functional Testing in Kinesis CI is designed to simulate user journeys on Tableau dashboards. It offers options to test dashboard elements, such as filters and parameters, to simulate user clicks, and more importantly to test the underlying data.

Functional testing is very flexible. Every functional test in Kinesis CI is made up of a series of tasks that will be run by the Kinesis test runner one after the other. You can simulate user interactions and describe complete user journeys with Functional testing, such as setting and asserting filters and parameters, or user clicks.

You can also test your underlying data based on user-defined rules against an expected data set or query your underlying data source with an SQL command, and compare the content of your database to the data displayed on your Tableau server in real-time. Functional testing offers you the opportunity to design and implement a flexible and comprehensive test plan to ensure the functionality of your Tableau Dashboards is preserved at all times.

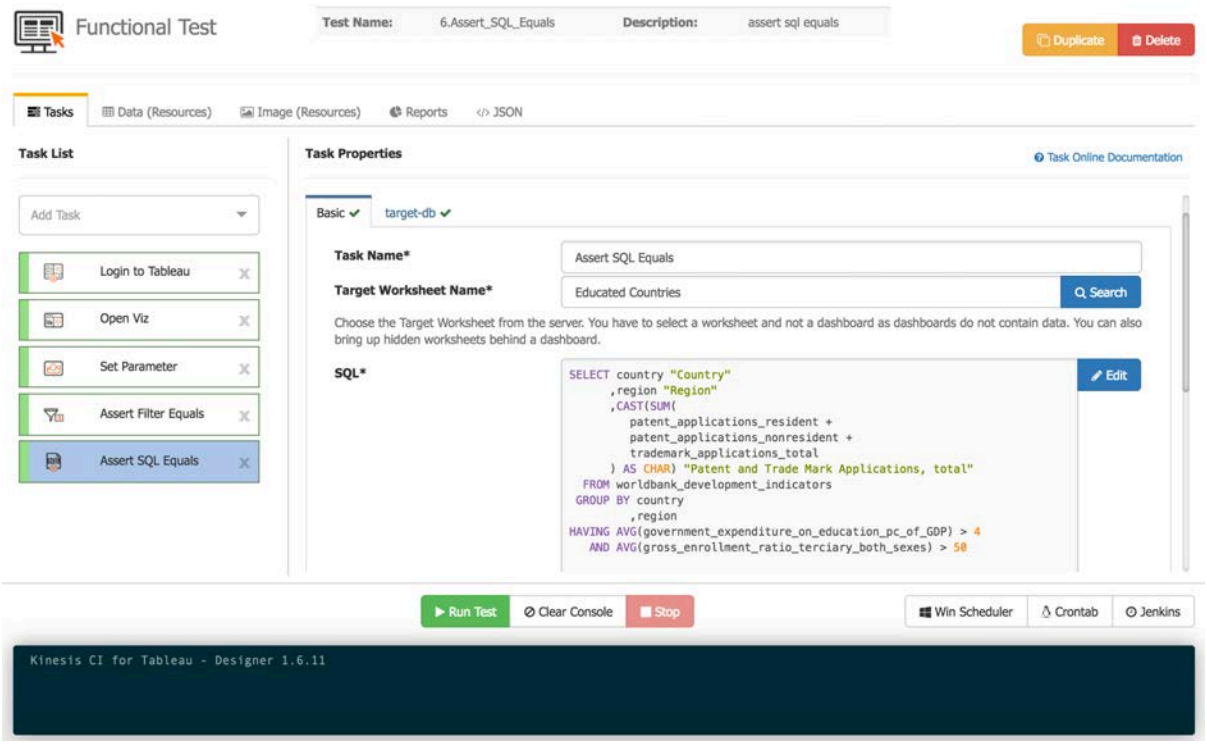
Tasks in Functional Testing

In Kinesis CI at present, the following tasks are available within Functional Testing:

1. Login to Tableau - Logs in to Tableau Server
2. Open Viz - Opens a Tableau Visualization
3. Publish to Tableau - Publishes a Tableau file to Tableau Server (twb, twbx, tds, tdsx)
4. Refresh Extracts - Refreshes Data Extracts based on data source
5. Load Data - Loads data from a CSV file into a database table
6. Set Filter - Validates Tableau filters (separate List, Date, and Date range options available)
7. Set Parameter - Validates Tableau parameters
8. Assert Data Equals - DDownloads and compares the underlying data on a worksheet to an expected result defined by the user
9. Assert Image Equals - Downloads and compares the layout in a picture format to an expected result defined by the user
10. Assert SQL Equals - Compares values from the underlying database, queried by an SQL command, to the data on the Tableau worksheet/ dashboard
11. Assert Data Rules - Validates data displayed on Tableau Server to rules defined by the user

12. Assert Filter Equals - Verifies the existence of a filter and compares the value of a filter to an expected result
13. Assert Parameter Equals - Verifies the existence of a parameter and compares the value of a parameter to an expected result
14. Select Marks - Selects marks on your Tableau visualization to simulate clicks/ user interactions
15. Drive Browser - Run custom web driver commands based on individual requirements, i.e. when using SAML SSO
16. Switch Tab - Switches between worksheets within a Tableau workbook
17. Run Command - Gives additional flexibility to run a command in the command-line interface.

The above testing tasks can be combined in a sequence to describe test cases. Please see below an example for Functional testing.



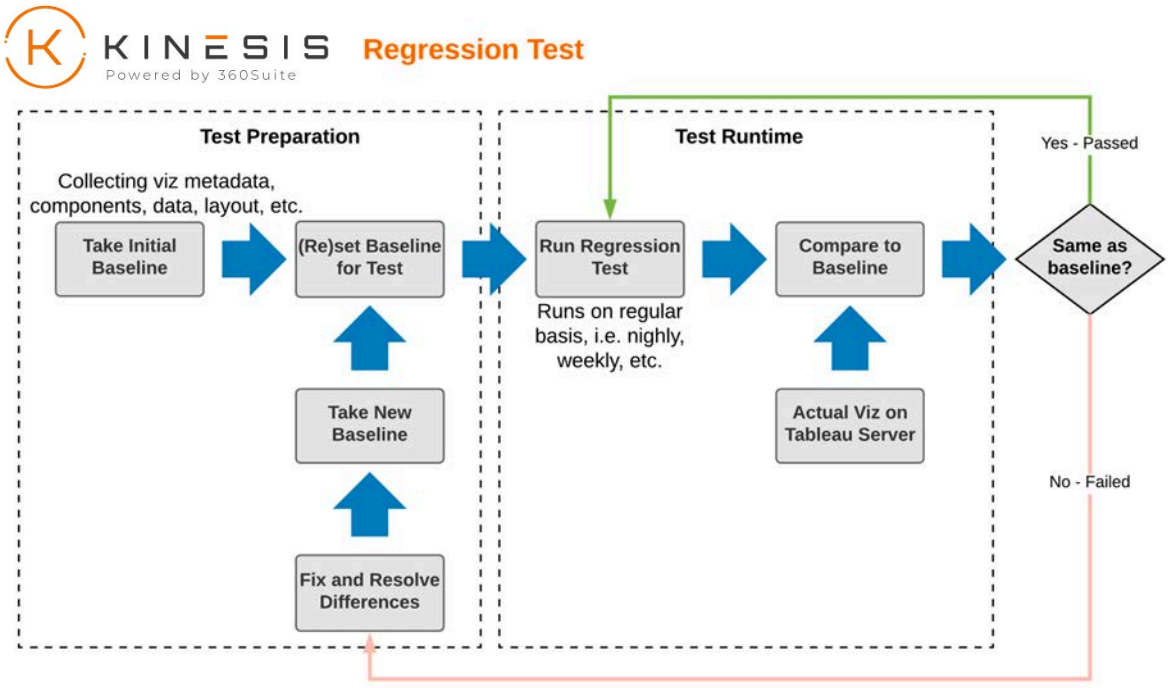
Validating underlying data with Assert SQL Equals task

Regression Testing

In regression testing, users can compare a Tableau dashboard to a baseline taken earlier to check for regression over time. Users take an initial baseline of the dashboard they wish to test. The baseline contains all metadata such as worksheets, data, layout, filters, and parameters available on the dashboard.
Any future test runs will be compared to this baseline.

When a regression test is run, Kinesis CI compares the actual content of your Tableau server to the baseline taken earlier.

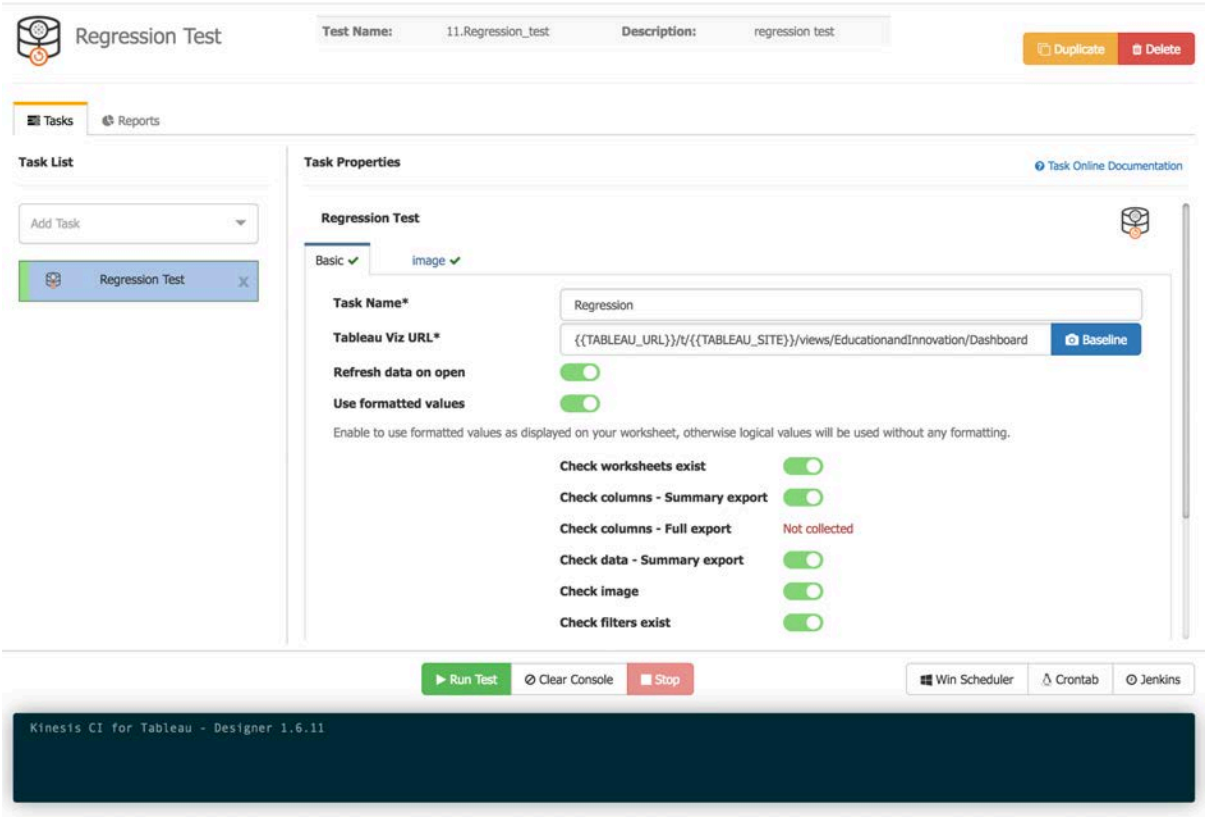
If the baseline and the actual dashboard are the same, the test passes. If there is a difference, the test fails. To update the Baseline, tests can be re-based. Following a re-base, the baseline will be updated and any test runs will compare metadata to the updated baseline.



Flow diagram of Regression Testing

Regression Testing

Kinesis CI also gives flexibility for users to determine what to include in their Regression test cases. For instance, if data is changing on a regular basis, users can exclude data testing in regression, and focus on the structure of the workbooks and elements such as filters and parameters.



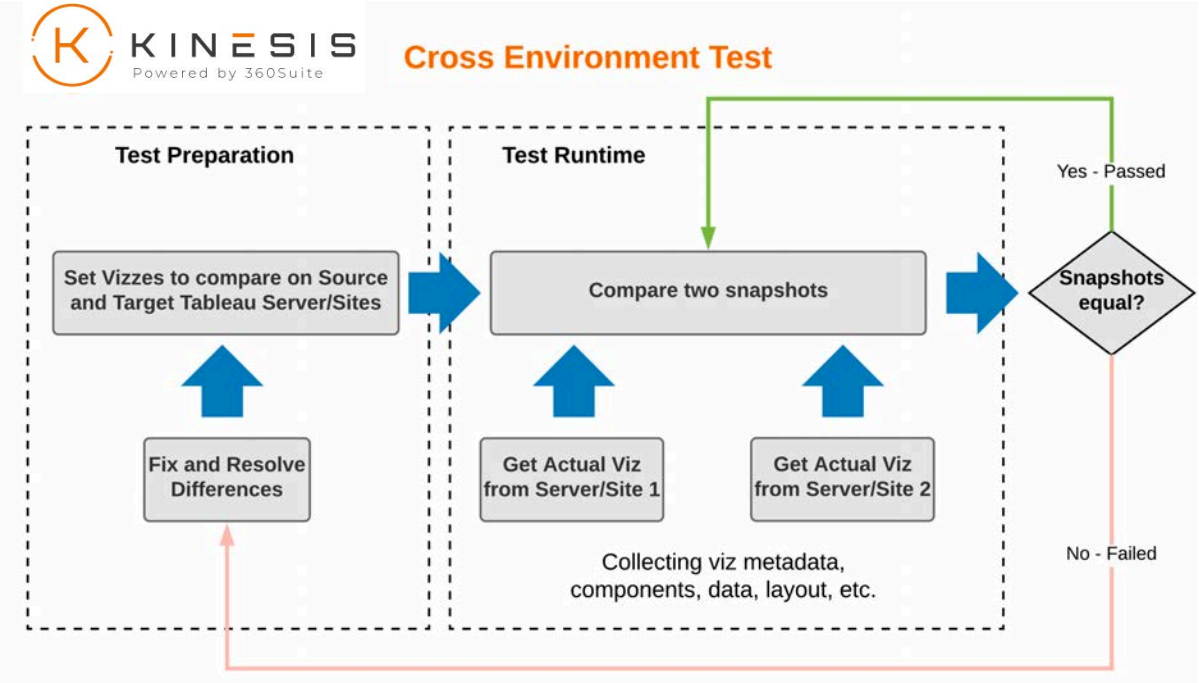
Regression testing task properties

Cross-Environment Testing

With Cross-Environment testing, users can compare the same dashboard on two different Tableau environments at the same time. These will be the Source and Target Environments on Tableau.

For instance, a dashboard on a dev server can be compared to the same dashboard already deployed on the prod server. Alternatively, when conducting a Tableau Server upgrade you have the old and the new server running in parallel -- this means you can automatically compare the dashboards on those servers to accelerate pre-upgrade testing.

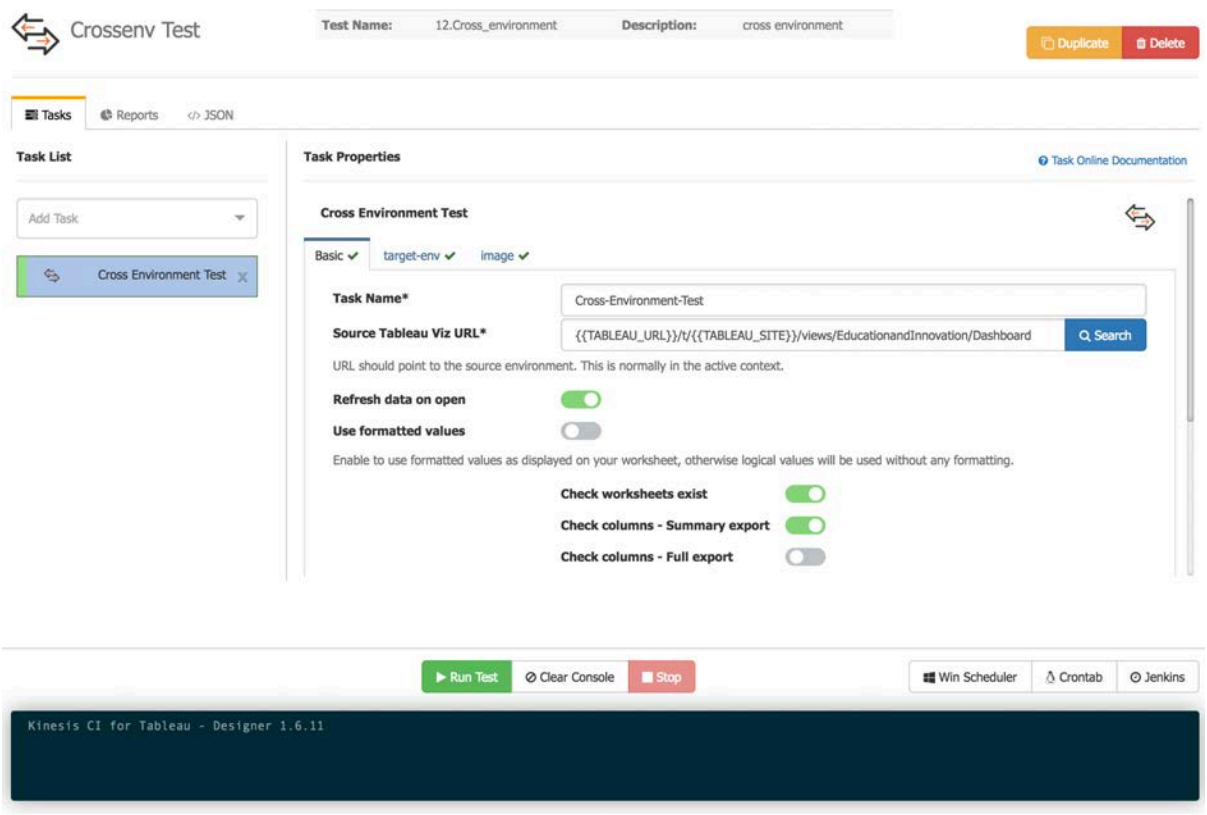
When the test is run, Kinesis takes a snapshot of both the source and the target environment and compares the two. If the two snapshots are the same, the test passes, if they differ, the test fails.



Flow diagram of Cross-Environment Testing

Cross-Environment Testing

Similar to Regression testing, in Cross-Environment testing users can also select what to include in the scope of the Cross-Environment test, i.e. worksheets, data columns, the actual data, the layout, as well as filter and parameter names, data types, and values.



Cross-Environment testing task properties

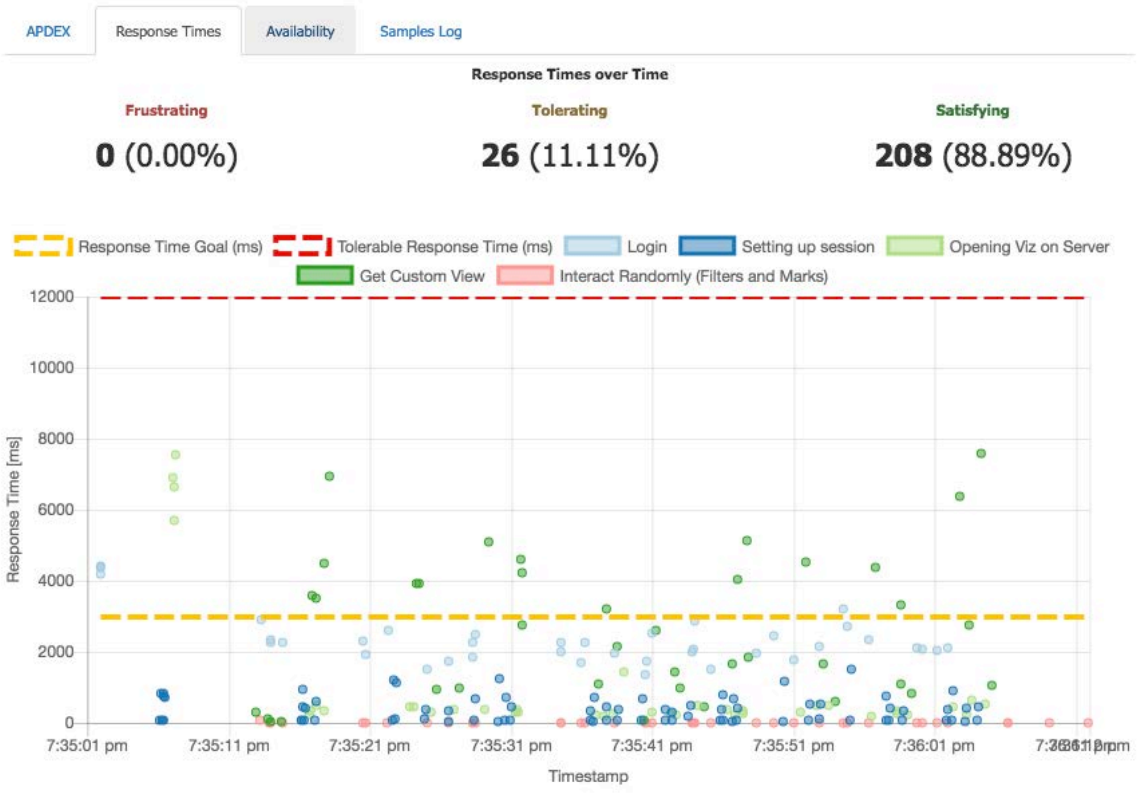
Performance Testing

Performance testing in Kinesis CI drives load to your Tableau server and assesses response times based on SLA requirements. You can work with multiple concurrent users and dashboards. Users can define SLA's (Service Level Agreements) with multiple goals, such as Performance Goal, Availability Goal, or Apdex Score.

Apex (Application Performance Index) score is commonly used for web applications. Users can define a performance and an availability goal. Apdex is a combination of availability and performance metrics, where fast and error-free responses increase the score, and long responses and errors reduce the score. It is similar to Service Level Agreements (SLAs) but more tolerant of rare occasional delays. Apdex ranges from 0 to 1, where 1 means all users are happy, and 0 means all users are frustrated.

In Performance testing, users can include up to 100 concurrent clients to test the Performance of the Tableau Server and they can determine a pool of Tableau views to include in the test case.

Performance test run reports are illustrative and easy to understand, featuring an overview of response times, availability score, and samples logs.



Performance Test Report - Response Times

Flexibility through Context Variables

Tableau server connection details can be set up in context variables. Users can set-up multiple contexts, enabling them to run tests on multiple sites or servers, changing only the Active Context -- there is no need to refactor test cases.

With changing context, continuous deployment of Tableau dashboards can also be implemented.

Importing Tableau Dashboards into Kinesis CI

Users can also import Tableau files, including twb and twbx workbooks and data sources into Kinesis. This way, dashboards can be kept together with the respective test cases, promoting test-driven development. Using this approach, users can constantly monitor their dashboards to make sure existing functionality is preserved, even when new a functionality or features are added.

Compatibility with Enterprise Solutions

Kinesis CI is also compatible with Single Sign-On systems and other custom login implementations.

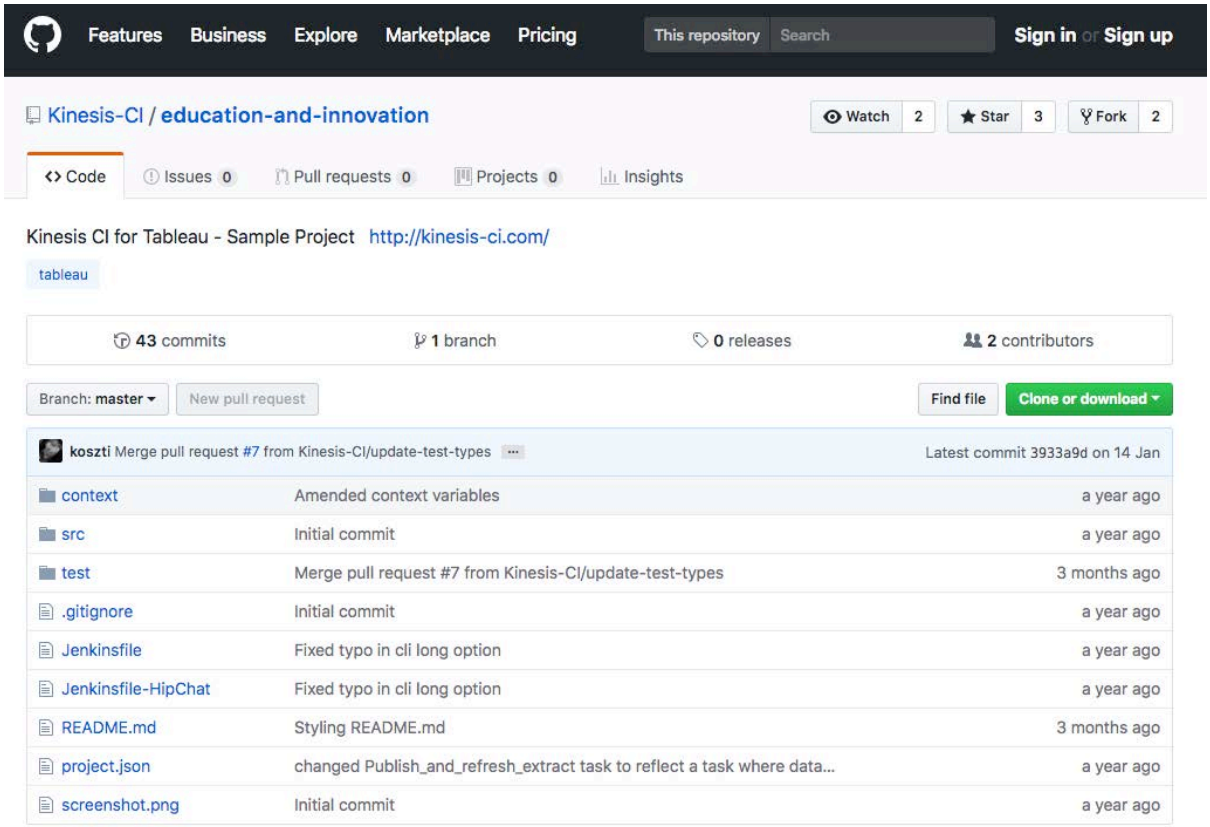
Test Runs

The tests are run by the command-line interface, and Tableau Server interactions are executed by a web driver.

Test runs can be executed from the UI one by one or automated in Windows Task Scheduler or Crontab. Users can also integrate Kinesis CI with a CI tool of your choice, such as Jenkins, Bamboo, or TeamCity.

Version Control

Kinesis CI projects use a Kinesis CI Project Standard Directory Layout that only contains text files (JSON files mostly), which enables easy collaboration between developers and version control.

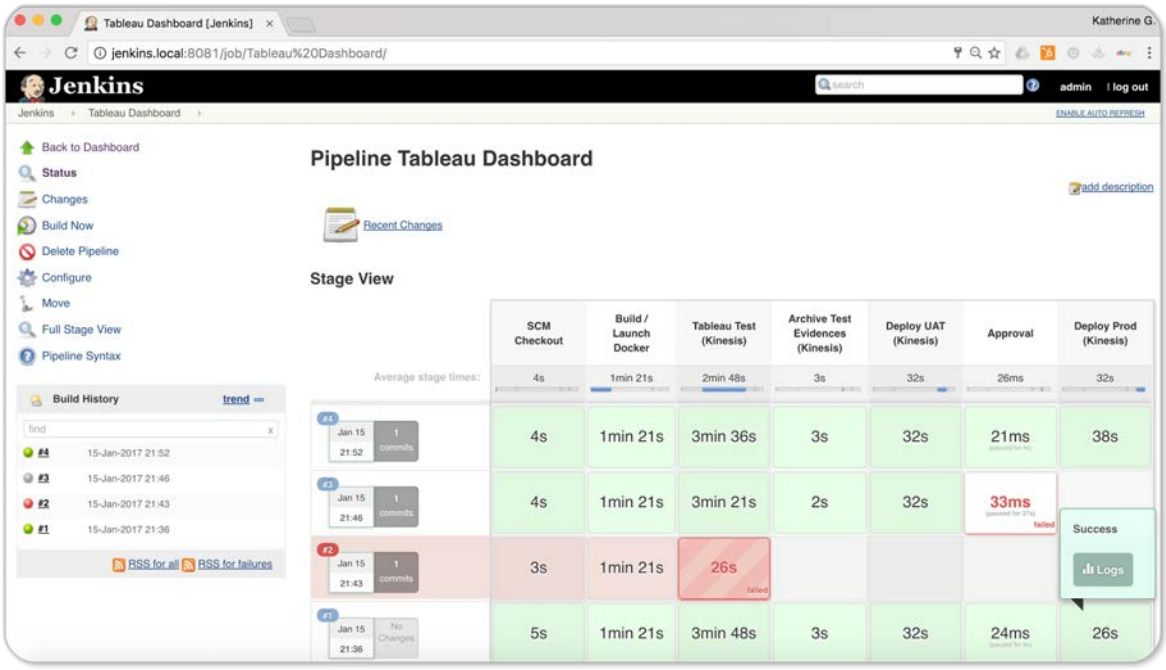


Sample Kinesis Project published to GitHub

Continuous Integration

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems immediately and locate them more easily.

To run tests automatically on a Continuous Integration Server, i.e. Jenkins, TeamCity, etc., users need to install the Kinesis Command Line Interface to the server where their CI server is running and integrate it with the CI tool of choice.



Example Jenkins Pipeline with stages running Kinesis tests

Deployment

Kinesis CI is shipped in bundle packages including the following two software components:

- **Kinesis Designer:** User Interface for building and running test cases in an easy-to-use environment.
- **Command Line Interface (CLI):** Used for running tests, enables automated test runs and continuous integration.

The bundle packages contain self-executable files and no further installation is required. Kinesis Designer is installed locally at the BI developers' computer and Kinesis CLI may be deployed on a server to run tests automatically.

Operating Systems

- Windows
- MacOS
- Linux (CLI Only)

About Kinesis

Testing should be an integral part of our BI development process to deliver reliable reports in a cost-effective manner. Kinesis CI is a unique tool designed for Tableau that provides a sophisticated and comprehensive test environment to cover all your BI testing needs.

Kinesis CI is powered by 360Suite, which offers industry-leading solutions for Tableau, helping BI and Data Analytics professionals strive towards an insight-driven organization. Allowing customers to trust every decision they make and action they take is our mission. Kinesis CI is a test framework solution for Tableau that increases trust in data and improves user experience, which combined, leads to higher user adoption and better decision-making.

Learn More About Kinesis

KINESIS
Powered by 360Suite

contact@kinesis-ci.com