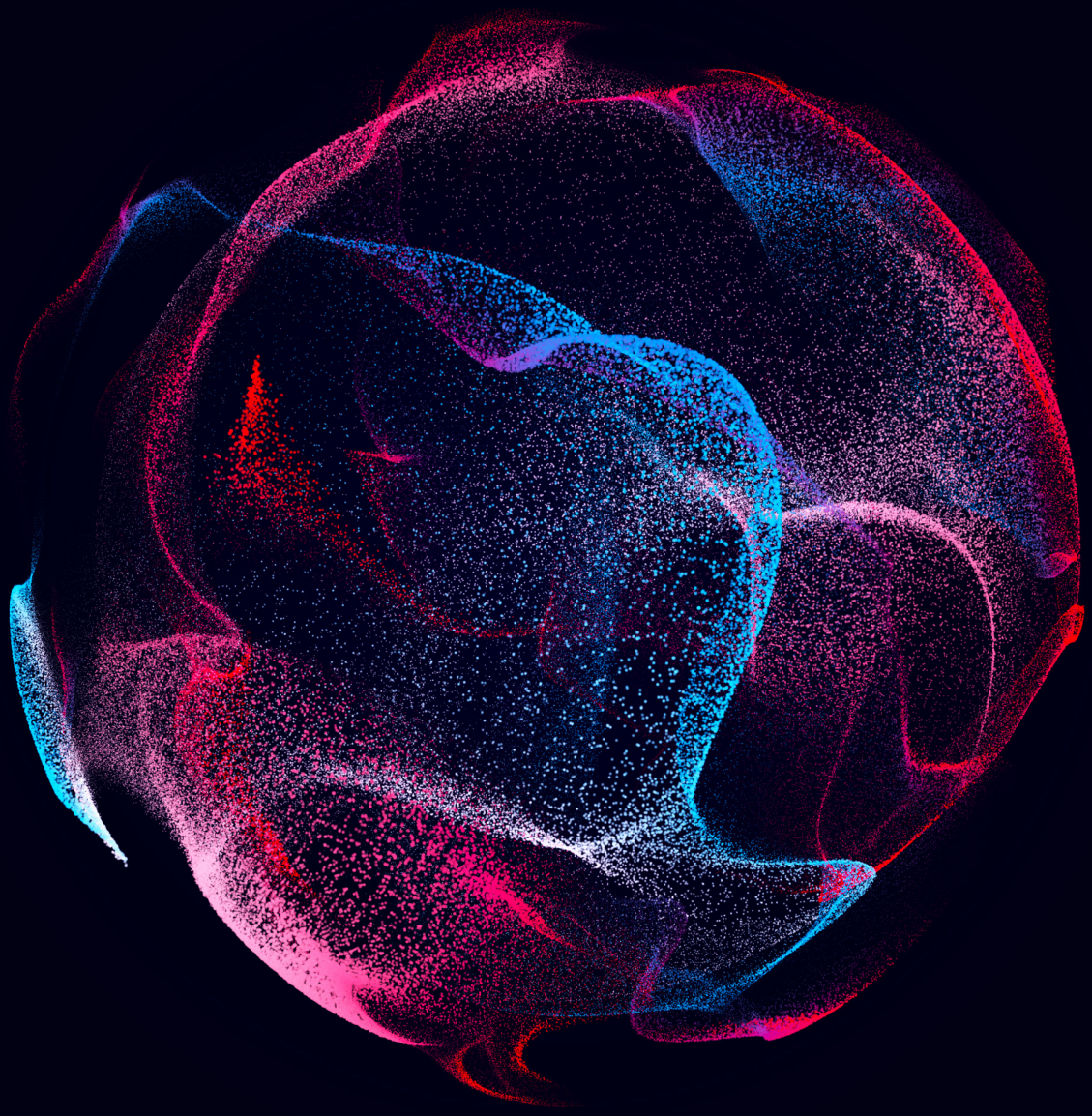
zensar

Guidelines for Guidewire Test Automation Cloud Migration

Whitepaper



An RPG Company

1. Introduction

As the insurance industry continues its shift toward cloud-native operations, cloud-based test automation is a foundational capability, not only for quality assurance, but also for the broader goal of building a resilient, scalable, and future-ready technology ecosystem.

Insurers planning a Guidewire cloud migration should not view it as merely an application infrastructure transformation; it is also a major quality engineering and test automation transformation. Cloud-based test automation provides elastic scalability for parallel test execution across Guidewire's PolicyCenter, BillingCenter, ClaimCenter, and Digital Portals, reduces infrastructure overhead, and integrates seamlessly with modern CI/CD pipelines to support continuous delivery.

Migrating Guidewire test automation to the cloud is not simply a lift-and-shift of scripts and runners; it also requires assessing, adapting, stabilizing, and optimizing existing automation assets to support the cloud-based Guidewire architecture.

Drawing on our experience with successful migration projects, this whitepaper examines key considerations, challenges, best practices, and implementation strategies for migrating Guidewire test automation to the cloud. It also offers practical guidance on evaluating frameworks, managing test data, ensuring environment readiness, securing cloud-native testing, optimizing costs, and governing cloud-native testing processes.

2. Migration objectives for Guidewire test automation

A clear migration objective is like a flight plan for a long journey. It defines the destination, sets the route, identifies critical checkpoints, and ensures that every team works toward the same outcome. Without it, automation efforts can become fragmented, focusing on script execution rather than on confirming whether the migrated Guidewire platform is truly ready to support business operations.

For Guidewire cloud migration, test automation must go beyond verifying that existing test cases continue to pass. It must provide measurable assurance that business-critical workflows remain functionally correct, integrated systems continue to operate reliably, migrated data retains its integrity, and the cloud platform can support ongoing releases without increasing operational risk.

The migration objectives below provide a common direction for automation design, execution, governance, and cloud-readiness decisions.

Objective	Description
Business workflow validation	Ensure PolicyCenter, BillingCenter, ClaimCenter, and Portal workflows function correctly after migration.
Risk reduction	Detect defects before UAT or production release.
Regression acceleration	Reduce manual regression cycle time through automated execution.
Framework reusability	Adapt existing automation assets for future cloud releases and migrations.
Environment flexibility	Enable the same automation suite to run across multiple cloud environments.
Quality governance	Create clear checkpoints, guidelines, and acceptance criteria for cloud readiness.
Continuous quality enablement	Create a reusable automation foundation that supports future Guidewire cloud upgrades, releases, and continuous testing practices.

3. Framework evaluation strategy

Remodeling a home without first evaluating its foundation and structural condition can lead to repeated repairs, unexpected delays, and costs that exceed rebuilding. Similarly, demolishing a structurally sound home without assessment can waste an investment that could have been modernized efficiently.

The same principle applies when selecting a framework for a Guidewire cloud migration. Over the years, insurers may have invested heavily in their on-premises automation framework, test scripts, reusable components, test data, utilities, integrations, reporting capabilities, and team knowledge.

Before beginning the migration, organizations should evaluate whether to:

- Reuse or modernize the existing automation framework.
- Adopt the Guidewire GT Framework.
- Use a hybrid approach that combines existing assets with GT Framework capabilities.
- Build a new cloud-ready framework.

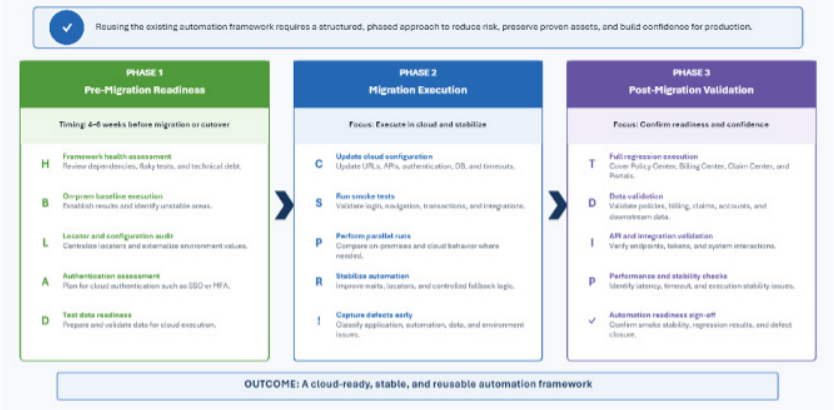
A proof of concept should compare the existing framework and the Guidewire GT Framework using representative workflows such as policy submission, billing, claims, portal access, document validation, and API integration. The assessment should consider cloud compatibility, Guidewire alignment, asset reusability, team familiarity, implementation effort, stability, scalability, and long-term cost.

The objective is not simply to retain the current framework or select the GT Framework. It is to choose the solution that best balances reuse, cloud readiness, reliability, implementation speed, and long-term sustainability. A proper evaluation at the outset prevents costly rework and establishes a strong foundation for the automation migration.

4. Recommended automation migration approach

If the POC outcome clearly favors the existing automation framework for migration, the focus should shift from framework selection to controlled modernization and cloud enablement. Reusing the framework does not mean moving the existing scripts to the cloud without change. The framework must be assessed, adapted, and stabilized to address differences in configuration, authentication, UI behavior, integrations, test data, and performance.

A structured three-phase approach helps protect the organization's existing automation investment while reducing migration risk. It establishes a clear progression from preparing and baselining the current framework to validating it in the cloud and, finally, confirming readiness for production and future Guidewire cloud releases.



Phase 1: Pre-migration readiness

Recommended timing: Four to six weeks before migration or cutover

This phase establishes a reliable baseline and prepares the existing automation assets for cloud execution.

- Conduct a framework health assessment.
- Execute the existing suite in the on-premises environment to establish a baseline.
- Audit and update unstable or environment-dependent locators.
- Review and externalize configuration values.
- Assess cloud authentication requirements.
- Prepare and validate cloud-ready test data.

Phase 2: Migration execution

This phase focuses on executing the adapted framework in the cloud environment and stabilizing scripts based on cloud-specific behavior.

- Update cloud URLs, API endpoints, authentication settings, database connections, timeouts, and browser configurations.
- Run smoke tests covering login, navigation, critical Guidewire transactions, and integration touchpoints.
- Perform parallel on-premises and cloud executions where comparison is required.
- Stabilize scripts by updating locators, improving waits, and adding controlled fallback logic.
- Identify defects early and classify them as application, automation, data, integration, or environment issues.

Phase 3: Post-migration validation

This phase confirms that the migrated framework provides sufficient functional coverage, execution stability, and production confidence.

Criteria	Key questions
Current framework health	Is the existing framework stable, reliable, and free from significant technical debt or recurring failures?
GT framework suitability	Does the Guidewire GT Framework provide capabilities and standardization that materially improve automation effectiveness?
Existing asset reusability	How many current test scripts, utilities, data components, and integrations can be reused without extensive redevelopment?
Setup time	How quickly can the framework be made cloud-ready?
Team familiarity and learning curve	Does the team already understand the framework, or will it require additional training and onboarding?
Stability	Has the framework been proven in previous releases?
Maintainability	Can locators, configurations, data, and utilities be easily updated?
Cloud compatibility	Can it support cloud URLs, authentication, API tokens, and dynamic waits?
Execution scalability	Can it run smoke, sanity, and regression suites efficiently?
Risk level	Will migration introduce framework-level instability?
Future sustainability	Can the framework support future Guidewire cloud upgrades, application changes, and continuous testing requirements?

- Execute full regression across PolicyCenter, BillingCenter, ClaimCenter, and Portals.
- Validate policy, billing, claims, account, and downstream system data.
- Verify cloud APIs, authentication tokens, and system integrations.
- Perform performance and stability checks for latency- and timeout-prone workflows.
- Complete the automation readiness sign-off based on smoke-test stability, regression results, defect closure, and business-critical workflow coverage.

This phased approach ensures the existing framework is not merely transferred to the cloud, but transformed into a stable, reusable, cloud-ready automation capability that supports both migration assurance and future Guidewire releases.

5. Key automation changes required for cloud migration

Migrating an existing automation framework to the cloud requires targeted changes to address differences between on-premises and cloud environments. This section covers the most common updates and how to implement them in any cloud automation migration project.

5.1 Locator modernization

Cloud migration can change UI rendering behavior, component identifiers, dynamic attributes, page load timing, and DOM structure. Hardcoded locators that worked in on-prem environments may fail in the cloud.

- Use centralized locator files.
- Avoid hardcoded IDs that contain environment-specific or dynamic values.
- Prefer stable attributes where available.
- Use parameterized locators.
- Review Guidewire-specific widgets, dropdowns, tables, and modal dialogs carefully.

5.2 Authentication flexibility

Authentication often changes during cloud migration. Teams may move from legacy SSO to OKTA, modern SSO, MFA-enabled authentication, or token-based authentication.

- Create reusable login utilities.
- Externalize authentication type through configuration.
- Support multiple authentication flows.
- Keep credentials out of code.
- Use secure vaults or secret managers for sensitive data.

5.3 Dropdown and UI interaction standardization

Guidewire applications contain complex UI components such as dropdowns, editable fields, dynamic lists, search screens, popups, and tables. Cloud migration may expose inconsistencies in how automation interacts with these elements.

- Standardize dropdown selection methods.
- Use reliable utility methods instead of one-off script-level logic.
- Add validation after selection.
- Use explicit waits before interacting with dynamic elements.
- Maintain separate handling for standard HTML dropdowns and custom Guidewire widgets.

5.4 Resilient click and interaction handling

Cloud environments may introduce timing delays or rendering differences. Elements may appear visible but may not be immediately clickable. A layered interaction strategy improves resilience without hiding genuine application defects.

1. Standard click
2. JavaScript click
3. Actions class click
4. Keyboard interaction such as the Enter key
5. Retry with explicit wait

5.5 API and database updatest

Cloud migration usually changes API URLs, authentication methods, database connectivity, and timeout requirements.

Component	On-prem pattern	Cloud pattern
API URL	API URL	HTTPS cloud endpoint
API authentication	Basic authentication	Bearer token or OAuth-based flow
Database	Direct SQL connection	Secure cloud connection with SSL
Timeout	Lower timeout	Higher timeout due to network/cloud latency
Secrets	Local config or scripts	Vault or secret manager

6. Cloud migration automation checklistt

A structured checklist helps ensure that critical automation activities are completed at the appropriate stage of cloud migration. It provides teams with a consistent way to track framework readiness, cloud execution, defect resolution, and final validation while reducing the risk of missed dependencies or late-stage rework. Our recommended checklist for each stage is outlined below:

Stage	Checklist items
Pre-migration	• Framework reviewed • Flaky tests corrected • On-prem baseline regression completed • Test data inventory prepared • Environment values externalized • Hardcoded locators removed • Authentication strategy reviewed • API endpoints identified • Database access confirmed • Reporting reviewed
Migration	• Cloud URLs configured • Smoke tests executed • Login flow validated • Key workflows validated • API authentication validated • Database connectivity validated • Defects categorized • Daily execution reports published
Post-migration	• Full regression completed • High-severity defects closed • Data validation completed • API and integration validation completed • Performance-sensitive scripts reviewed • Final pass percentage confirmed • Lessons learned documented

7. Common challenges and recommended solutions

Cloud migration can introduce technical differences that disrupt previously stable automation. Understanding the likely impact of these changes and applying the appropriate corrective measures helps teams resolve failures faster, improve execution reliability, and maintain steady progress throughout the migration. The most common challenges and our recommended solutions are as follows:

Challenge	Impact	Recommended solution
Environment differences	Tests pass on-prem but fail in the cloud.	Externalize all environment-specific values.
Authentication changes	Login scripts fail repeatedly.	Build configurable authentication handlers.
Locator failures	UI elements are not found.	Centralize and parameterize locators.
Network latency	Tests fail due to timeouts.	Use dynamic waits, retries, and an improved timeout strategy.
Database connectivity	Automation cannot validate backend data.	Update secure connection strings and SSL settings.
API authentication	API tests fail after migration.	Replace basic auth with token-based authentication where required.
Test data gaps	Scripts fail due to missing data.	Prepare cloud-ready test data before execution.
Flaky tests	Regression confidence decreases.	Stabilize before migration and track recurring failures separately.

8. Do's and don'ts

Successful cloud migration depends not only on changes to the automation framework, but also on how consistently teams adhere to cloud-ready practices. The following do's and don'ts help teams improve framework portability, execution stability, security, and maintainability while avoiding common migration risks.

Do's

- Parameterize configuration values so the same framework can run across multiple environments.
- Centralize and standardize locators to simplify maintenance and reduce duplication.
- Build flexible login utilities to accommodate changes in authentication mechanisms.
- Validate beyond the UI by including API, database, integration, and data-level checks.
- Use explicit waits and controlled retry strategies to manage cloud latency and dynamic application behavior.
- Run automation frequently in cloud environments to identify issues early.
- Secure credentials and sensitive information using approved vaults or secret-management solutions.
- Document recurring failures and resolutions to support faster troubleshooting and future migrations.

Don'ts

- Do not hardcode IP addresses, URLs, credentials, or environment-specific values.
- Do not migrate broken or flaky tests without stabilizing them first.
- Do not assume cloud performance will be identical to on-premises performance.
- Do not rely solely on UI validation to confirm business and data integrity.
- Do not overlook API, database, or integration changes introduced by the cloud migration.
- Do not implement isolated fixes within individual test scripts when the solution should be added to reusable framework utilities.
- Do not begin full regression testing before smoke tests are stable.
- Do not store passwords, tokens, or other secrets in scripts, repositories, or property files.

9. Recommended governance model

A clearly defined governance model ensures that automation migration activities are coordinated across functional, technical, cloud, security, and test-data teams. Establishing ownership and accountability for each critical area enables faster decision-making, proactive risk management, and consistent quality controls throughout the Guidewire cloud migration.

Role	Responsibility
Automation lead	Owns automation migration strategy and framework readiness.
Functional QA lead	Confirms business scenario coverage.
Guidewire technical lead	Supports application-specific changes and troubleshooting.
DevOps/cloud lead	Provides environment, pipeline, and cloud connectivity support.
Security lead	Reviews authentication, credentials, and secret handling.
Test data lead	Ensures availability of cloud-compatible test data.
Testing manager	Tracks readiness, risks, milestones, and sign-offs.

10. Key success factors

A successful automation migration depends on more than adapting scripts to run in the cloud. It requires early planning, controlled reuse, resilient framework design, comprehensive validation, and continuous learning. The following success factors improve migration stability, reduce delivery risk, and establish a sustainable automation foundation for future Guidewire cloud releases.

- **Start early:** Begin automation assessment and migration several weeks before cutover to allow sufficient time for updates and stabilization.
- **Reuse proven assets:** Retain existing frameworks, utilities, and test scripts when they are stable, maintainable, and familiar to the team.
- **Parameterize configurations**: Keep URLs, credentials, environment names, timeouts, API endpoints, and database settings configurable.
- **Design for authentication changes:** Build flexible login mechanisms that can support multiple authentication providers and security requirements.
- **Stabilize before scaling:** Ensure smoke tests are reliable before progressing to full regression execution.
- **Validate beyond the UI:** Include data, API, database, integration, and backend validations to confirm end-to-end business integrity.
- **Invest in reporting:** Produce clear execution reports that distinguish application defects from automation, data, integration, and environment issues.
- **Capture lessons learned:** Convert migration knowledge, recurring issues, and proven solutions into reusable guidelines for future releases.

11. Benefits of our structured automation migration strategy

Without a structured strategy, automation migration can become a series of reactive fixes, resulting in duplicated effort, unstable execution, and delayed validation. Our recommended approach aligns teams around common quality objectives, prioritizes critical workflows, and introduces the right controls at each stage. This reduces migration risk while establishing a reliable and reusable automation capability for future Guidewire cloud releases.

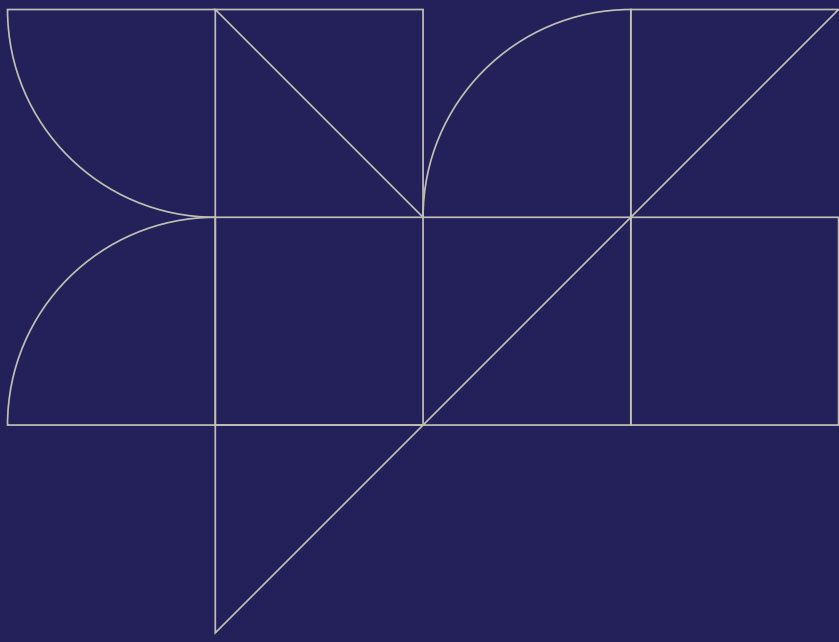
Benefit	Description
Faster regression	Automated execution shortens regression cycles and reduces manual testing effort.
Earlier defect detection	Frequent and repeatable validation identifies issues before UAT or production.
Lower migration risk	Critical business workflows are validated consistently throughout the migration.
Improved team productivity	Teams spend less time on repetitive testing and more time on analysis and defect resolution.
Reusable cloud-ready framework	Adapted automation assets can support future Guidewire releases and cloud upgrades.
Greater stakeholder confidence	Clear results and quality evidence improve confidence in migration and release readiness.
Stronger governance	Defined quality gates, reporting, and sign-offs support informed migration decisions.

12. Conclusion

Guidewire cloud migration requires test automation to evolve beyond simply executing existing scripts in a new environment. A successful transition depends on selecting the right framework strategy, assessing existing assets, addressing cloud-specific changes, and following a structured approach across pre-migration readiness, migration execution, and post-migration validation.

Ultimately, a well-planned automation migration does more than support a successful cutover. By following the guidelines in this whitepaper, you can establish a resilient, reusable, cloud-ready quality engineering foundation that improves migration confidence, reduces regression effort, enables earlier defect detection, strengthens production confidence, and supports future Guidewire cloud releases.

Insurers that invest in this transformation today will be better positioned to support faster Guidewire upgrades, respond to regulatory and market changes with confidence, and deliver higher-quality digital experiences to all the stakeholders in their ecosystem.



zensar

An RPG Company

At Zensar, we're 'experience-led everything.' We are committed to conceptualizing, designing, engineering, marketing, and managing digital solutions and experiences for over 145+ leading enterprises. Using our 3Es of experience, engineering, and engagement, we harness the power of technology, creativity, and insight to deliver impact.

Part of the $4.8 billion RPG Group, we are headquartered in Pune, India. Our 10,000+ employees work across 30+ locations worldwide, including Milpitas, Seattle, Princeton, Cape Town, London, Zurich, Singapore, and Mexico City.

For more information, please contact: **info@zensar.com | www.zensar.com**