Performance Testing, Scale Center *and more...*

Svet Voloshin

What is Performance Testing?

- Salesforce performance testing is a specialized process to evaluate the efficiency, scalability, and reliability of a Salesforce application under various conditions.
- The primary goal is to ensure that the application performs well for end-users, especially under high load or stress.
- This kind of testing is crucial in environments where Salesforce is heavily customized or extensively used.

Where does it happen?

Staging Environment (Full Copy):

- Purpose: To replicate real-world usage as closely as possible before going live.
- Process: This environment is a near-exact replica of the production environment, used for final performance evaluations.

Production Environment:

- Purpose: To monitor and optimize performance in the live environment.
- Process: This involves real-time monitoring and analysis, especially useful for identifying issues that only appear under actual user loads.

Full Copy Sandbox

A Full Copy Sandbox is a type of environment in Salesforce that provides a complete replica of your production environment, including all data and metadata. This sandbox type is used for various purposes like testing, training, and development.

Key Characteristics of Full Copy Sandbox:

- Data Replication: It contains all your production organization's data, including records, attachments, and metadata.
- Purpose: Ideal for performance testing, load testing, staging, and user training, as it provides a realistic environment that closely mirrors your live Salesforce instance.
- Refresh Interval: The Full Copy Sandbox can be refreshed every 29 days. This means you can update it to reflect the current state of your production environment at nearly monthly intervals.
- Storage Limit: The storage limit is generally the same as your production environment, allowing for a full-scale testing and development platform.

Full Copy Considerations

Advantages:

- Realistic Testing Environment: Because it's a full copy, it provides the most accurate environment for testing.
- Safe for Experimentation: Changes in a sandbox do not affect your production environment, making it safe for experimentation.
- Data-Rich: Having a complete set of data is particularly useful for complex testing scenarios that require detailed data.

Limitations:

- Resource Intensive: Requires more storage and might have more overhead in terms of setup and maintenance compared to other sandbox types.
- Refresh Limitations: The 29-day refresh interval might be restrictive for some rapid development cycles.

Management and Best Practices:

- Regular Refreshes: Plan refreshes strategically to ensure the sandbox aligns with key development or testing phases.
- Data Masking: Even though it’s a secure environment, consider masking sensitive data for training or testing purposes.
- Monitor Usage: Since it's a complete copy, it's important to monitor usage to avoid performance degradation.

Just any Full Copy Sandbox?

- Today, you cannot test beyond a particular volume in a pre-production environment, and even if you attempt to do so, you may get throttled;
- **Scale Testing Service** is your solution to that problem. With STS add, you will be able to leverage “production-like” infrastructure temporarily based on a reserved time window and run your test at higher loads.

Understand Org Throttling

- To manage capacity and availability, Salesforce sometimes applies a throttle to your org in Sandbox or Production instances. Requests are placed in a queue and executed at a slower rate compared to the incoming request rate, typically 50% of the incoming rate of requests. The impact on your org depends on your implementation and can include increased request times, latency, or timeout errors.
- Salesforce has automation and recurring processes that increase horizontal and vertical scale. Throttles are applied when these processes are exhausted and guardrails aren't sufficient. Customer success is a shared responsibility and Salesforce regularly notifies customers about phases in the customer lifecycle journey.

Common Reasons for an Org Throttle

- Execution of an inefficient SOQL
- Unapproved performance testing, as performance testing is only allowed on Sandbox instances via the performance testing approval process
- A sudden spike in inbound synchronous requests per second that includes browser requests and SOAP or REST API calls
- A high number of inbound asynchronous requests per second that include Bulk API jobs, @future jobs, and Apex Queueable jobs
- Rare Salesforce infrastructure issues that cause automatic or manual throttles

How is it done?

- Using Automated Testing Tools: Tools like Apache JMeter or LoadRunner are used to simulate multiple users and transactions on the Salesforce platform.
- Monitoring and Analysis Tools: Salesforce offers tools like the Developer Console, Lightning Platform API, and Workbench for real-time monitoring and analysis.
- Custom Scripts and Scenarios: Performance tests often involve custom scripts that mimic real user interactions with the application, including accessing various Salesforce objects and executing Apex code.
- Governor Limits: Salesforce has specific governor limits to ensure shared resources are used efficiently. Performance testing must consider these limits to ensure the application adheres to Salesforce's best practices.
- Cloud-Based Testing: Since Salesforce operates in the cloud, performance testing often involves cloud-based tools and environments, which provide scalability and a more accurate representation of user interaction.

What about Load Testing?

Load Testing

- Objective: Load testing specifically focuses on how the application behaves under expected user loads. It is a subset of performance testing.
- Scope:
  - Determine if the application can **handle the anticipated number of users or transactions simultaneously**.
  - Measure the system's **response time** and **throughput** under various load levels.
- Conditions: Performed by simulating a specific number of users or transactions over a defined period to test the application's load-handling capabilities.
- Use Cases: Critical in ensuring that the application can meet its performance goals when it goes live and is subjected to real-world user loads.

Key Differences

Focus:

- Performance testing is **broader**, assessing the application's overall performance characteristics.
- Load testing is **more focused**, examining how the application handles specific, expected user loads.

Test Conditions:

- Performance testing includes a **variety of conditions** – not just high load scenarios.
- Load testing **specifically simulates expected user load conditions**.

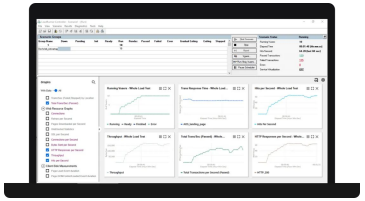
Objectives:

- Performance testing aims to uncover any performance-related issues and assess various performance metrics across different scenarios.
- Load testing aims to ensure that the application can handle the expected number of users or transactions **without performance degradation**.

Outcome:

- Performance testing provides a **comprehensive view of the application's performance**, including **how it scales, uses resources, and handles different load conditions**.
- Load testing primarily determines the **maximum operating capacity of the application** and helps in identifying **at what point the application's performance starts to degrade under load**.

LoadRunner



LoadRunner is a widely used performance testing tool, developed by Micro Focus (now OpenText) (formerly by Hewlett-Packard). It is designed to test the performance of various applications under load, providing insights into how systems behave when accessed by multiple users simultaneously.

Versatile Testing Capabilities:

- Supports various types of performance testing, including **load, stress, endurance, and spike testing**.
- Can **simulate thousands of users concurrently using application services** to identify performance bottlenecks.

Protocol Support:

- Offers extensive support for various protocols and technologies, including Web/HTTP, Web Services, SAP, Oracle, and many others.
- This allows LoadRunner to test a wide range of applications, from simple websites to complex enterprise software.

Scripting:

- Utilizes VuGen (Virtual User Generator) for scripting. It allows testers to create custom test scenarios to mimic real user interactions with the application.
- Supports scripting languages like C.

Real-time Analysis and Reporting:

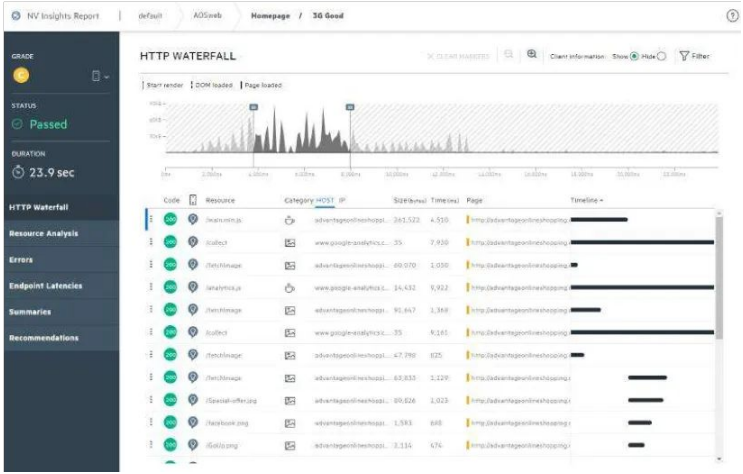
- Provides powerful analysis and reporting tools.
- Enables real-time monitoring of performance metrics during test execution.

Integration Capabilities:

- Can be integrated with other tools and platforms, including continuous integration environments, for comprehensive testing strategies.

How LoadRunner Works

1. Creating Test Scripts:
    - Test scenarios are created using VuGen. These scripts replicate user interactions with the application.
2. Generating Load:
    - LoadRunner uses these scripts to generate load on the application by simulating multiple virtual users (VUsers).
3. Monitoring and Analysis:
    - During the test, LoadRunner monitors and captures performance data from various sources.
    - This data includes response times, throughput rates, resource utilization, and more.
4. Reporting:
    - After the test, LoadRunner provides detailed reports and graphs that help in analyzing the application's performance under load.



LoadRunner Use Cases & Considerations

Use Cases:

- Performance Benchmarking: Before going live, to ensure that applications meet performance expectations.
- Capacity Planning: To determine how many users an application can handle before performance degrades.
- Identifying Bottlenecks: To find and fix performance-related issues in the application.

Advantages:

- Scalability: Can simulate a large number of users with minimal hardware.
- Accuracy: Provides detailed and accurate performance metrics.
- Flexibility: Supports a wide range of applications and protocols.

Limitations:

- Complexity: Can be complex to set up and requires expertise in scripting and performance testing.
- Cost: Being a commercial tool, it might be expensive for small projects or organizations.

VuGen

VuGen, short for Virtual User Generator, is a key component of Micro Focus LoadRunner, a performance testing tool. VuGen is used for creating scripts that simulate the actions of real users on a software application. These scripts are essential for conducting performance, load, and stress tests using LoadRunner. Here's a detailed overview:

1. **Script Creation**: VuGen enables the creation of test scripts that mimic user interactions with an application. These scripts are used by LoadRunner to generate virtual users (VUsers) during performance tests.

2. **Protocol Support**: It supports a wide range of protocols, including HTTP/HTTPS, Web Services, Database, SAP, Oracle, and more, allowing it to test a variety of applications.

3. **Recording and Manual Scripting**:
    - Recording: VuGen can record the actions of a user interacting with an application, which can then be converted into a script. This feature is especially useful for web-based applications.
    - Manual Scripting: Advanced users can write scripts manually, often using C language, for more complex or customized testing scenarios.

4. **Parameterization and Correlation**:
    - Parameterization: Allows dynamic input of data into the scripts, enhancing the realism of the tests.
    - Correlation: Automatically handles dynamic values like session IDs, which are unique for each user session.

5. **Integrated Development Environment**: VuGen provides an IDE-like environment with features like syntax highlighting, code completion, and debugging tools, making script development and testing more efficient.

Importance in Performance Testing

- Realism: VuGen scripts help in creating realistic load testing scenarios that closely mimic user behavior.
- Scalability: These scripts are essential for simulating a large number of virtual users, essential for load and stress testing.
- Flexibility: The ability to handle a wide range of protocols makes VuGen versatile for testing different types of applications.

Lightning Experience Performance Optimization

Component Optimization:

- Efficient Component Design: Design Lightning components to be efficient in data handling and rendering. Avoid unnecessary server calls and complex client-side computations.
- Lazy Loading: Implement lazy loading in components, where data is loaded only when needed, reducing initial page load times.

Data and Server Interaction:

- Minimize Server Calls: Reduce the frequency and volume of server calls. Use caching and store data locally when possible.
- Optimized Apex and SOQL: Ensure that Apex classes and SOQL queries are optimized for performance. Avoid queries inside loops and limit data retrieval to only what's necessary.

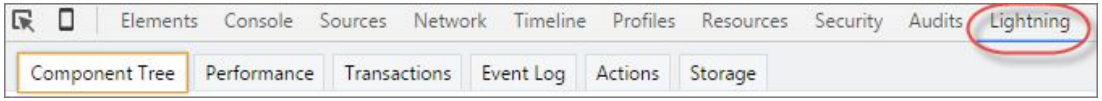
User Interface Design:

- Simplified Layouts: Design page layouts to be simple and intuitive. Overly complex layouts with too many components can slow down page loading.
- Conditional Rendering: Use conditional rendering to display components only when necessary, reducing the load on the client-side.

Resource Loading and Management:

- Minimize Resource Size: Keep the size of resources like images and scripts minimal. Utilize Salesforce's Content Delivery Network (CDN) for faster loading.
- Asynchronous Loading: Load JavaScript and other resources asynchronously to improve page load times.

LEPO (continued...)



5. Testing and Monitoring:

- Performance Testing: Regularly test the performance of the Lightning Experience using tools like the Lightning Console Performance Inspector.
- Monitoring: Utilize Salesforce's built-in monitoring tools to track performance and user experience metrics.

6. Network and Browser Performance:

- Optimized Network Usage: Ensure that network interactions are efficient. This can include minimizing the data transferred and optimizing API calls.
- Browser Caching and Performance: Leverage browser caching capabilities and ensure compatibility and performance across different browsers.

7. Best Practices Implementation:

- Follow Salesforce Best Practices: Salesforce provides guidelines and best practices for optimizing Lightning Experience performance. Keeping abreast of these recommendations is crucial.

8. Customization with Care:

- Balanced Customization: While customization is a powerful feature of Salesforce, it should be done thoughtfully. Unnecessary or inefficient customizations can adversely affect performance.

9. Training and User Adoption:

- User Training: Educate users on efficient use of the Lightning Experience, which can also contribute to overall performance.

Org Scalability

Salesforce org scalability refers to the ability of a Salesforce organization (org) to handle increased loads and complexities as a business grows. This involves not only managing more data, users, and transactions but also ensuring that the system remains efficient, reliable, and maintainable over time. Here's a detailed look into the various aspects of Salesforce org scalability:

Key Factors in Salesforce Org Scalability:

- Data Volume Management:
    - Large data volumes can impact performance. Efficient data management strategies, like archiving old data and optimizing data models, are crucial.
    - Implementing data archiving and purging policies helps in maintaining optimal performance levels.
- Governor Limits:
    - Salesforce imposes governor limits to ensure shared resources are used efficiently across the multi-tenant environment.
    - Designing applications and customizations with these limits in mind is essential to prevent scalability issues.
- Application Design and Customization:
    - Scalable application design involves efficient Apex code, optimized SOQL queries, and effective use of Salesforce features like Process Builder, Workflows, and Lightning Components.
    - Avoid complex and nested triggers, and use asynchronous Apex (like Batch Apex, Queueable Apex) when processing large data sets.

Org Scalability (continued...)

- User and License Management:
  - As the number of users grows, managing user roles, profiles, and permissions efficiently becomes important.
  - Choosing the right mix of Salesforce licenses based on user roles and needs can impact both cost and scalability.
- Integration Scalability:
  - As organizations grow, they often integrate Salesforce with other systems. Scalable integrations require efficient API usage and robust error handling.
  - Utilizing Salesforce's integration patterns and best practices ensures that integrations are scalable and maintainable.
- Performance Testing:
  - Regular performance testing is vital, especially after major releases or when significant changes are made to the org.
  - Tools like LoadRunner can be used for performance testing.

Org Scalability (continued...)

- Monitoring and Maintenance:
  - Regular monitoring of system performance, user activity, and storage usage helps in identifying potential scalability issues early.
  - Scheduled maintenance and reviews of the org setup and customizations ensure ongoing scalability.
- Capacity Planning:
  - Anticipating future growth and planning accordingly is key. This includes considerations for data storage, API limits, and feature usage.
- Training and Change Management:
  - As the org scales, ensuring that users are adequately trained and that changes are managed effectively is important for overall system health.

Challenges and Best Practices

Challenges in Scalability

- Balancing Customization and Performance: Excessive customization can lead to performance bottlenecks.
- Staying Within Limits: Adhering to Salesforce’s governor limits while scaling can be challenging.
- Change Management: Managing changes efficiently in a rapidly scaling org requires robust processes.

Best Practices for Scalability

- Regular Health Checks: Use Salesforce’s Health Check and other tools to regularly evaluate the org’s health.
- Efficient Use of Resources: Optimize resource usage (like SOQL queries, API calls) for better scalability.
- Scalable Architecture Design: Design a scalable architecture from the outset, considering future growth and potential changes.

What if it doesn’t scale?

If a Salesforce organization (org) doesn't scale effectively, several issues can arise, **impacting both the user experience and the overall functionality of the Salesforce system**. The ability to scale is crucial in Salesforce, given its role as a central platform for managing customer relationships, sales processes, and business operations. Here are some of the key consequences of poor scalability in a Salesforce org:

1. Performance Degradation:

- Slow Response Times: Users may experience slower page loads and delayed responses to actions, reducing productivity and efficiency.
- Timeouts and Errors: In extreme cases, users might encounter system timeouts or errors due to overloaded servers or exceeded governor limits.

2. Data Management Challenges:

- Data Overload: As data volume grows without proper scaling strategies, it can become increasingly difficult to manage and navigate, leading to inefficiencies.
- Difficulty in Reporting: Large data volumes can slow down or complicate reporting processes, impacting decision-making capabilities.

3. Governor Limit Issues:

- Exceeded Limits: Salesforce enforces strict governor limits to ensure shared resources are used efficiently. Failure to scale properly can lead to frequent hitting of these limits, disrupting normal operations.
- Blocked Processes: Exceeding limits can lead to processes being blocked or terminated, which can disrupt business workflows and automation.

Scalability Challenges (continued...)

4. User Experience and Adoption:

- Frustrated Users: Poor system performance can lead to user frustration and reduced adoption of the Salesforce platform.
- Reduced Efficiency: If the system becomes cumbersome to use, it can reduce overall organizational efficiency and productivity.

5. Scalability of Integrations:

- Integration Failures: Integrations with other systems may fail or perform poorly if not scaled properly, leading to data inconsistencies and workflow issues.
- API Limitations: Hitting API limits can disrupt integrations and external applications reliant on Salesforce data.

6. Maintenance and Upkeep Challenges:

- Increased Maintenance Effort: A non-scalable org can require more effort to maintain and update, increasing the workload for admins and developers.
- Cost Implications: Poor scalability often leads to higher costs due to the need for additional resources to manage the growing complexity.

7. Business Impact:

- Impeded Growth: A Salesforce org that doesn't scale well can become a bottleneck, impeding the growth and agility of the business.
- Risk of Data Loss: In severe cases, scalability issues could lead to data integrity problems or loss.

8. Security and Compliance Risks:

- Increased Security Risks: Overloaded systems may be more vulnerable to security risks.
- Compliance Issues: Scalability issues might also lead to challenges in maintaining compliance with data management and privacy regulations.

Mitigation Strategies

- Regular Performance Audits: Regularly assess and optimize performance.
- Scalable Architecture: Design a scalable architecture from the start.
- Data Management Policies: Implement effective data management and archiving policies.
- Monitoring and Proactive Management: Monitor system usage actively and manage resources proactively.

Scale Testing Service

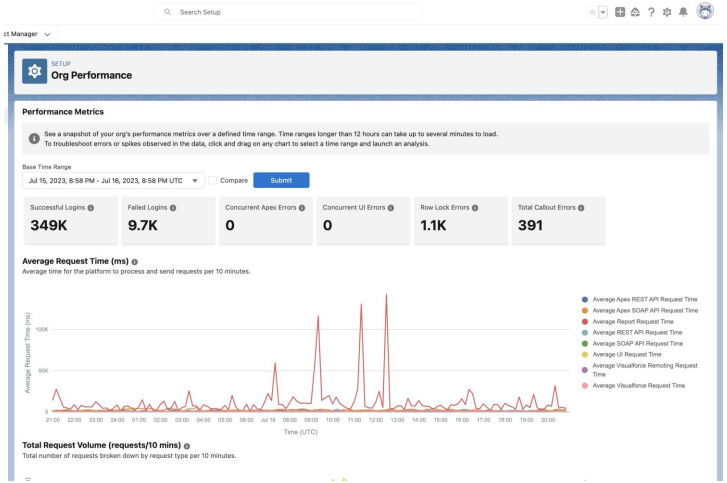
- With Full copy Sandboxes, you can create a copy of your entire production org, including all metadata and varied levels of customer data.
- However, there is no easy way to be confident of your application's scalability when preparing for your peak event.
- For example, if you are a retail customer, you want to be ready for Thanksgiving and the peak holiday season, or if you are a healthcare customer, you care about open enrollment season or a financial customer who wants to be ready for their tax season.
- Today, you cannot test beyond a particular volume in a pre-production environment, and even if you attempt to do so, you may get throttled; Scale Testing Service is your solution to that problem.
- With STS add, you will be able to leverage "production-like" infrastructure temporarily based on a reserved time window and run your test at higher loads.

Scale Testing Service

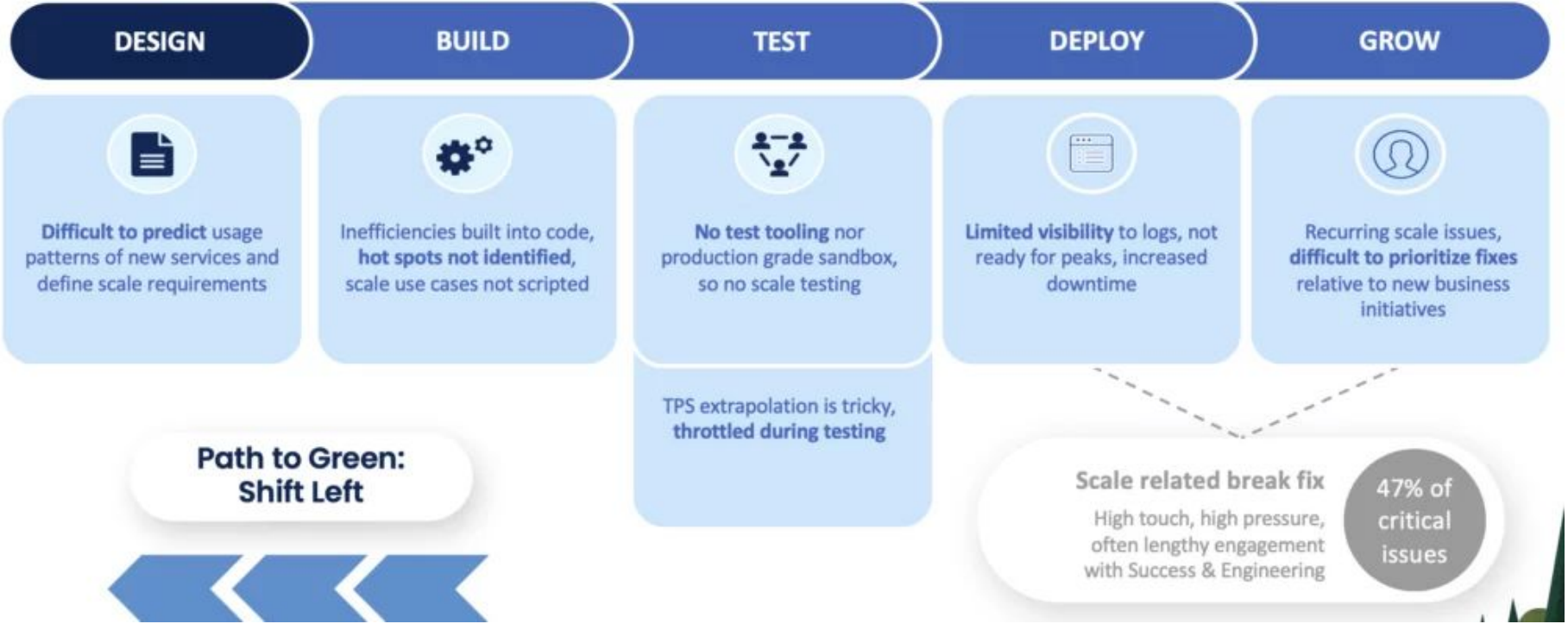
Download Link (PDF)

Salesforce Scale Center

- Salesforce Scale Center lets customers see a snapshot of their org performance metrics over a defined time range.
- Customers are provided with a single pane of glass view into common errors, including failed logins, **concurrent Apex errors**, and **Rowlock errors**.
- Additionally, customers are able to view key metrics visualized in a chart over the duration of the time interval, including **Average Request Time, Database CPU Time, and Total Errors**.

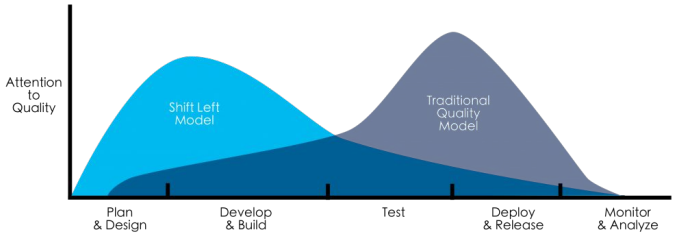


Salesforce Scale Center Lifecycle



Shift Left Methodology

- **Early and Continuous Testing**: Incorporating testing early in the development cycle and continuing it throughout.
- Increased Collaboration: Enhancing cooperation among developers, testers, and operations teams.
- Proactive Issue Resolution: Identifying and addressing issues early to reduce later fixing costs.
- Automated Testing: Utilizing automation for efficient and frequent testing.
- Quality Focus: Prioritizing quality from the beginning of the development process.
- Cost and Time Efficiency: Lowering time and costs associated with late-stage bug fixes.
- Effective Risk Management: Continuously assessing and managing potential risks.



Features of Scale Center

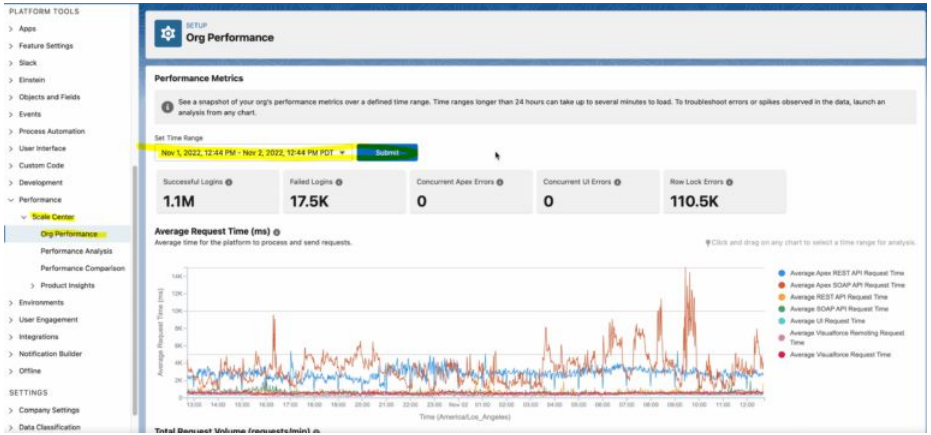
It helps identify performance and scale hotspots. Launch analysis to troubleshoot errors and compare two different periods. Key features that have been enabled at GA:

- Get near Real-Time Visibility into your essential Org Performance and Scale Metrics and critical errors like RowLocks, Concurrent Apex, UI, and callout Errors.
- Run deep dive investigations at precise intervals where you see spikes or anomalies—eight different types of investigations.
- Compare Org performance across different time intervals. A pre-production use case compares the results of the Scale Testing Service for other test runs.

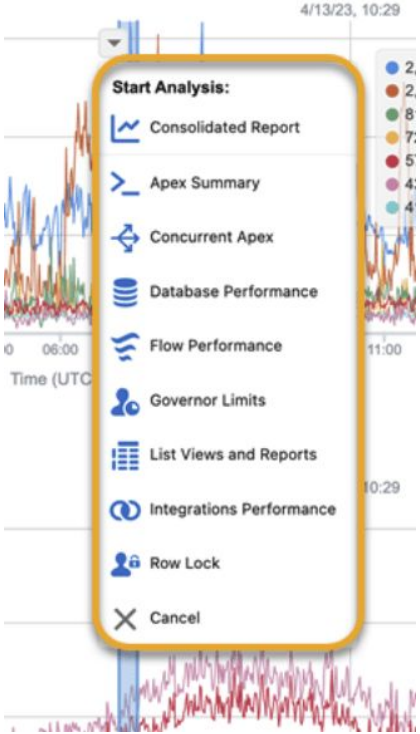
Org Performance

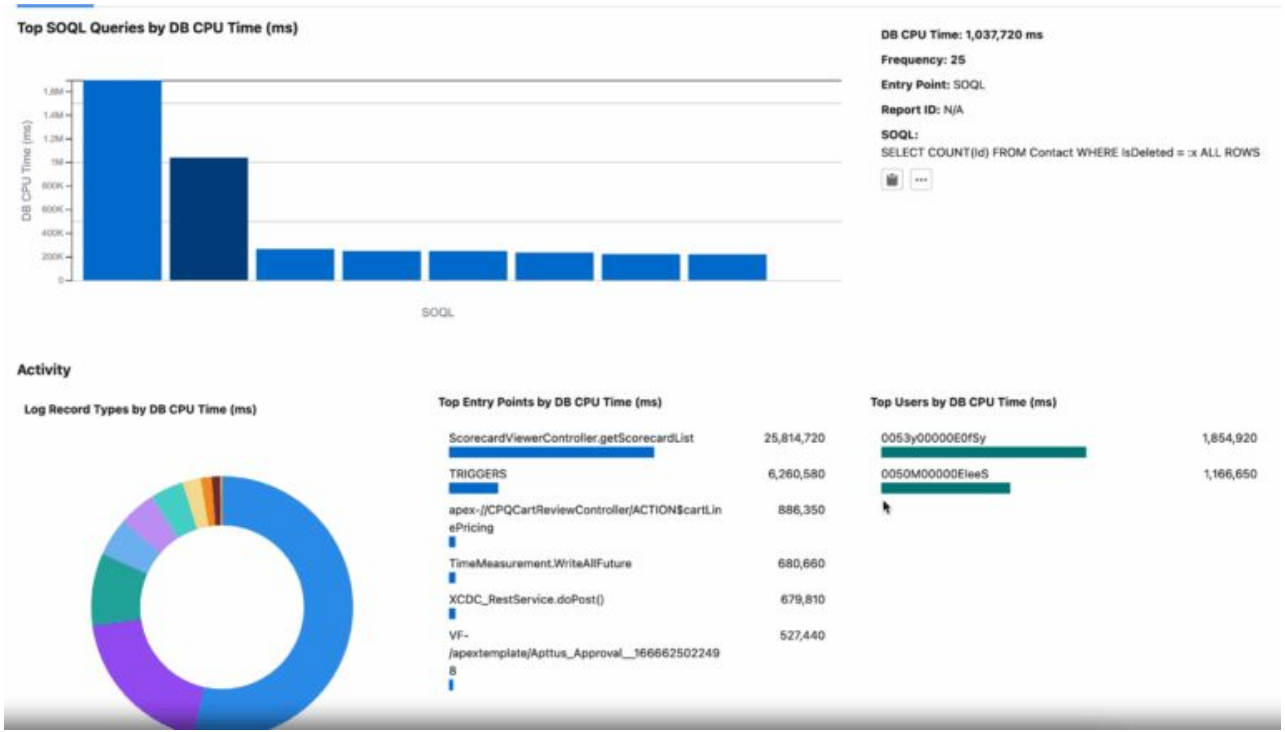
With org performance, you can find the org metric for the **last seven days or 30 days**. It will show all successful logins, **failed logins, concurrent Apex Errors, UI errors, and Row lock errors**. It provides a chart (on the right).

- Average Request Time: Average Request Time for the platform to process and send a request.
- Total Request Volume: Total Request per minute, broken down by request type.
- Total DB CPU Time: Cumulative CPU time spent on database cells.
- Total APP CPU Time: Total App CPU time spent on user code execution per minute.
- Total Login: Total number of successful logins.
- Error Per Minute: Row Lock errors, concurrent apex errors, and UI error per minute.

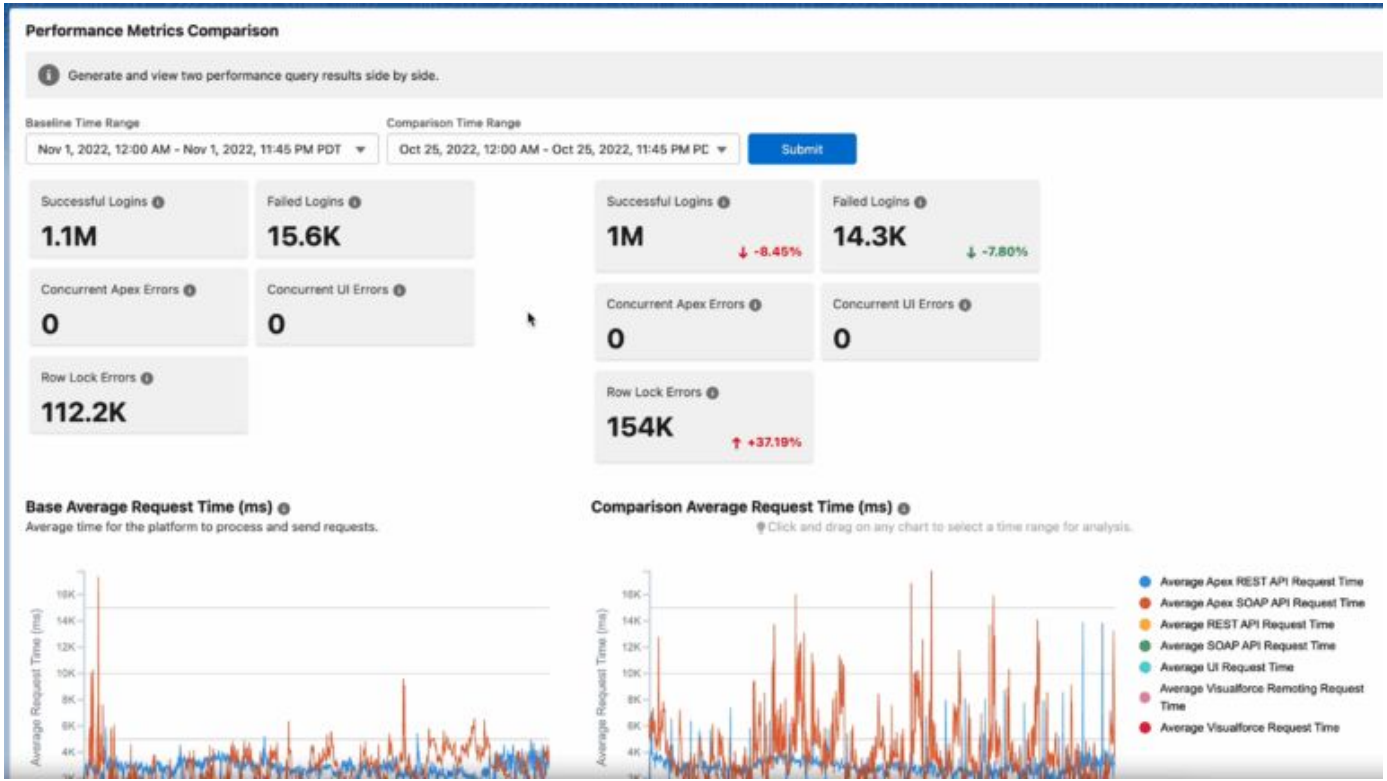


Performance Analysis





Performance Comparison



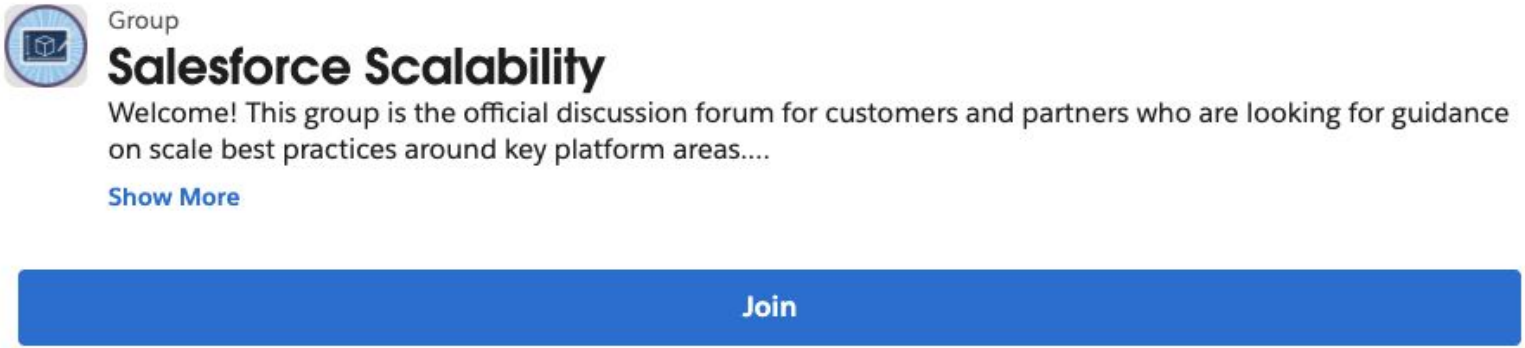
FAQ

Is Salesforce Scale Center FREE?

- Scale Center is free to use and is accessible in all **UE production and full-copy sandbox org**.

How far back in time can I retrieve data from the Scale Center

- Last 30 days.



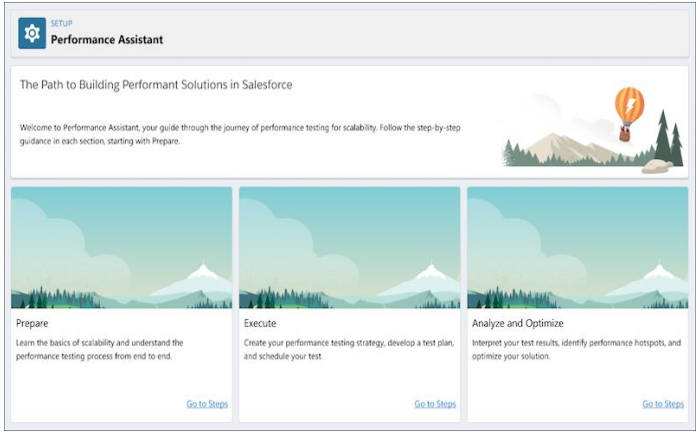
Performance Assistant

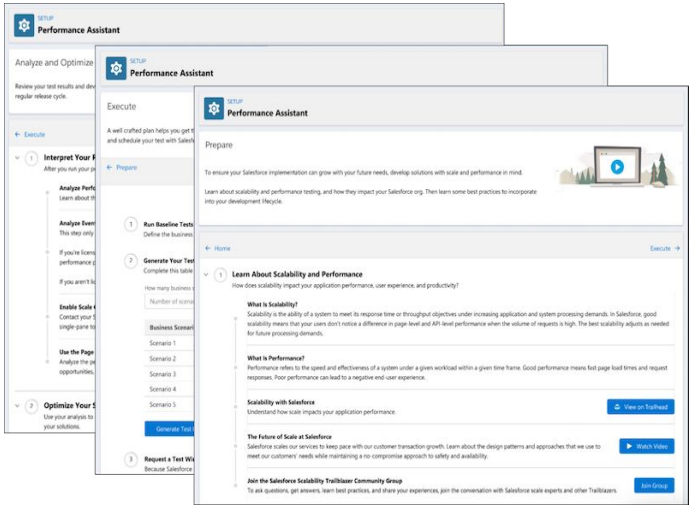
Performance Assistant is your **central hub of information** and resources about **scalability and performance testing** with Salesforce.

Use the step-by-step instructions, articles, and tools to help you architect your system, conduct performance testing, and interpret your results.

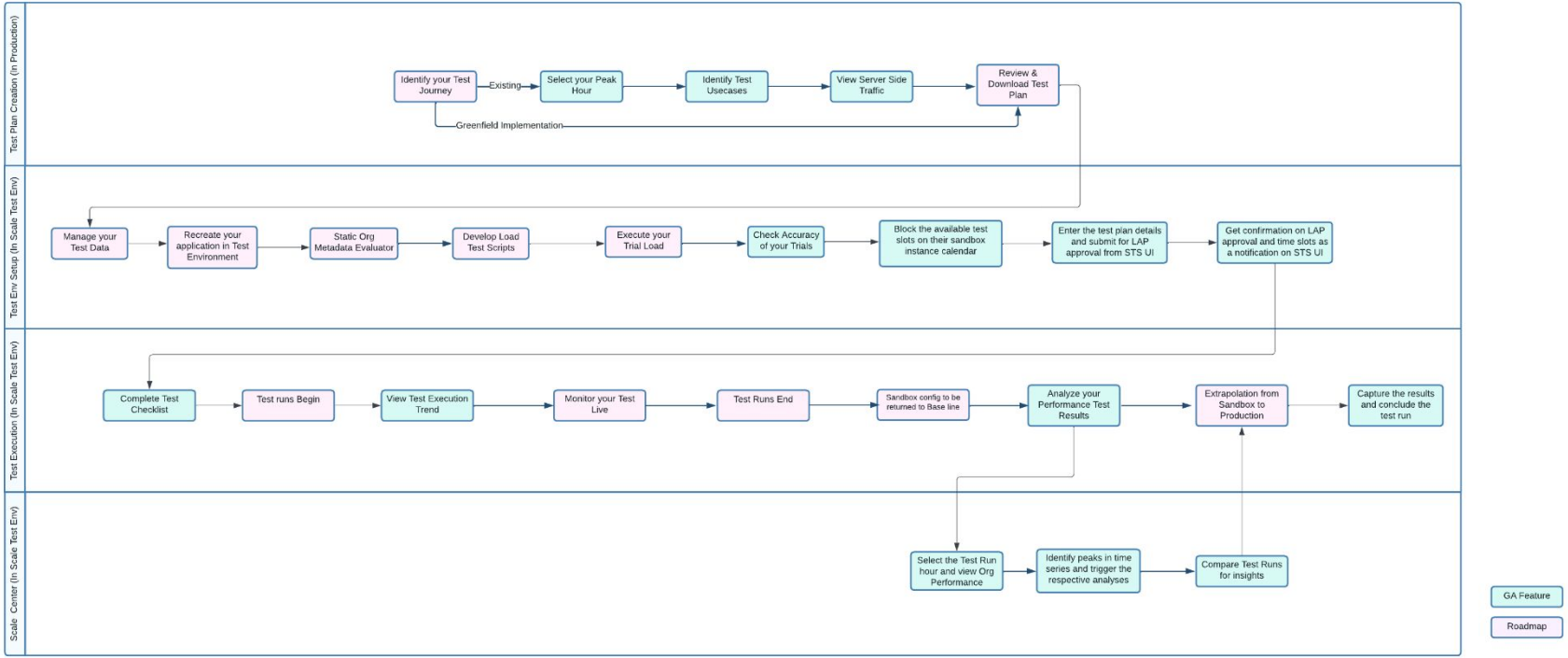
- Where: This change applies to Lightning Experience in Enterprise, Essentials, Unlimited, Professional, and Developer editions.
- Who: To use Performance Assistant, users need the View Setup and Configuration user permission.
- How: From Setup, in the Quick Find box, enter Performance Assistant, and then select Performance Assistant.

Performance test FAQs





STS - FY24 Test Tooling Features Overview



ApexGuru

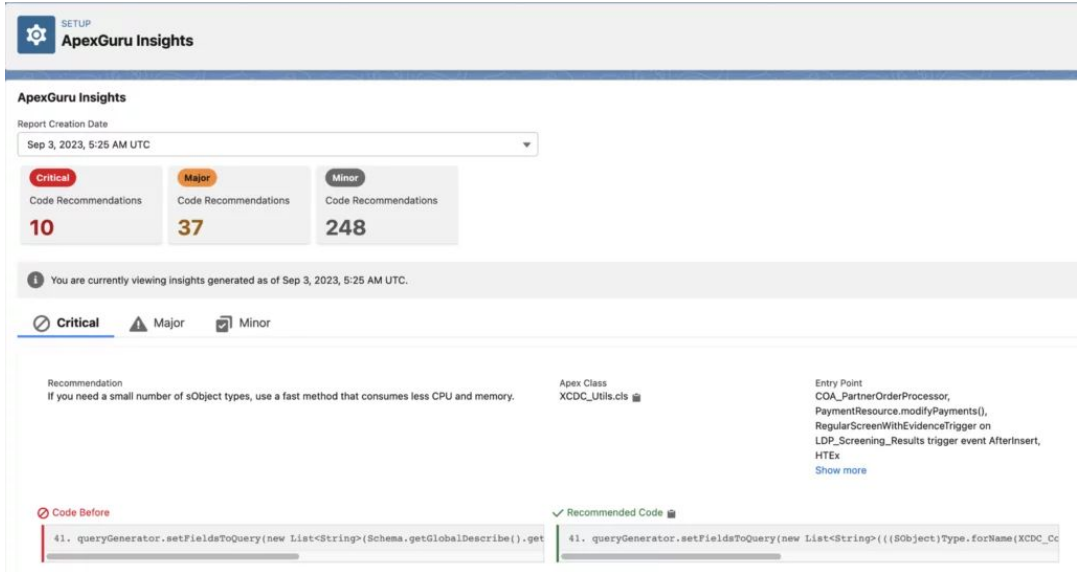
ApexGuru automates the detection of critical anti-patterns and performance hotspots in your apex code (runtime profiles) and provides customers with AI-driven insights and prescriptive code recommendations.

It's powered by Generative Artificial Intelligence (CodeT5 Model).

Pinpoint the exact class and line of code where the **anti-pattern** is detected, provide the recommended code fix, and prioritize these into critical, significant, and minor.

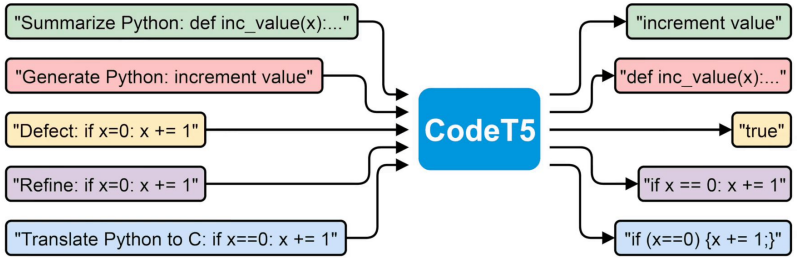
Feature of Apex Guru

- Code recommendations prioritize it as Critical, Major, and Minor based on the expected impact.
- Working SOQL's antipatterns, slowest classes, hot methods
- Proactively improve code quality
- Promising Impact Metrics we have received so far.
- 20% overall AppCPU Savings.
- 80% AppCPU savings for top methods



CodeT5 and CodeT5+

- **TL;DR:** CodeT5+ is a new family of open code large language models (LLMs) with improved model architectures and training techniques.
- CodeT5+ achieves the state-of-the-art performance among the open-source LLMs on many challenging code intelligence tasks, including zero-shot evaluation on the code generation benchmark HumanEval.
- Text-to-code generation: generate code based on the natural language description.
- Code autocompletion: complete the whole function of code given the target function name.
- Code summarization: generate the summary of a function in natural language description.



Continued Learning

- Salesforce Scale Center
- Introduction to Performance Testing (Developers' Blog)
- Performance test FAQs
- Develop Your Performance Testing Strategy (Trailmix)
- Previewing Performance Assistant and Scale Center (Medium.com)
- Amplify your ALM with Scale Center and Apex guru (Apex Hours video)