N

Nagabhushanam Rangisetty

Presenter Designation

Benefits of Integrating SDLC Tools in Enterprises

Benefits of Integrating SDLC Tools in Enterprises

Understanding Horizon as an SDLC Enterprise Platform

1. **Integrated Suite of Tools**

   Horizon offers an integrated suite of tools that supports agile software delivery.

2. **Standardized Tools and Processes**

   It utilizes standardized tools and processes to enhance efficiency in software development.

3. **Automation of Manual Functions**

   Horizon aims to automate manual functions, reducing the workload on development teams.

4. **Continuous Integration (CI)**

   The platform employs Continuous Integration (CI) to streamline the delivery process.

5. **Continuous Delivery (CD)**

   Horizon also incorporates Continuous Delivery (CD) pipelines to facilitate swift deployment.

Overview of Litmus Test Tool

Definition of Litmus Test Tool

Litmus Test is a test harness tool developed as an in-house solution.

Purpose of Litmus Test

It is used to rapidly test and increase efficiency in the testing process.

Integration with SDLC

The tool integrates testing with build and deployment processes on the SDLC pipeline.

Functions of Litmus Test in the SDLC Pipeline

1	Validation of Build and Deployment Events	Litmus Test is utilized to validate build and deployment events within the Software Development Life Cycle (SDLC).
2	Auto Promotion of Developer Code	It aids in automatically promoting developer code through various environments, streamlining the development process.
3	User Interface for Testing Management	Litmus Test provides a user interface that allows users to define, classify, and pool execution machines or agents for effective testing.

Configuring Test Sets with Litmus Test

Definition of Test Sets

Test sets can be defined for scheduled execution, allowing for systematic testing.

Event Triggered Execution

Facilitates build and deploy event triggered execution through delivery pipelines.

User Interface Functionality

The UI interface enables users to view test summary reports, enhancing visibility into testing outcomes.

Success Criteria Definition

Users can define success criteria for execution when triggered from the build or event.

Monitoring Execution Status with Litmus Test UI

with Litmus Test UI Litmus Test UI is useful to monitor the status of execution and also monitor the status of execution agents. Users can set up notification alerts from the interface.

Components of Litmus Test

Litmus Test Overview

The Litmus Test consists of two main components: the Litmus Test Agent and the Litmus Test UI.

Litmus Test Agent

Agents are installed on user execution machines and are responsible for running automation scripts.

Script Management

The agent downloads scripts and automation code to the machines.

Execution Process

Once the scripts are downloaded, the agent starts the execution process.

Results Reporting

After execution, the agent reports the results back to the Litmus Test server.

Target State Achievable with Litmus Test

Current State of Build and Deployment

The current process involves manual status checks and validations, leading to inefficiencies.

Manual Automation Scripts Execution

Execution of automation scripts is conducted manually, which can slow down the process.

Manual Maintenance of Execution Machines

Machines and agents used for execution are maintained manually, adding to operational overhead.

Delayed Error Detection

Errors in build and deployment are detected only after the event has occurred, resulting in potential setbacks.

Automated Validations with Litmus Test

With Litmus Test, build and deployment validations are conducted automatically, enhancing efficiency.

Scheduled Execution of Automation Events

The ability to schedule automation build and deployment events is enabled, streamlining processes.

Automatic Inventory and Execution Status Maintenance

Inventory and execution statuses are maintained automatically, reducing manual workload.

Immediate Error Detection

Errors are detected immediately and published to stakeholders post-execution, allowing for quick resolutions.

Integrating Litmus with Continuous Delivery Pipeline

1. **Integration with Orchestration Tools**

   Litmus can be configured with build and release orchestration tools like XLR.

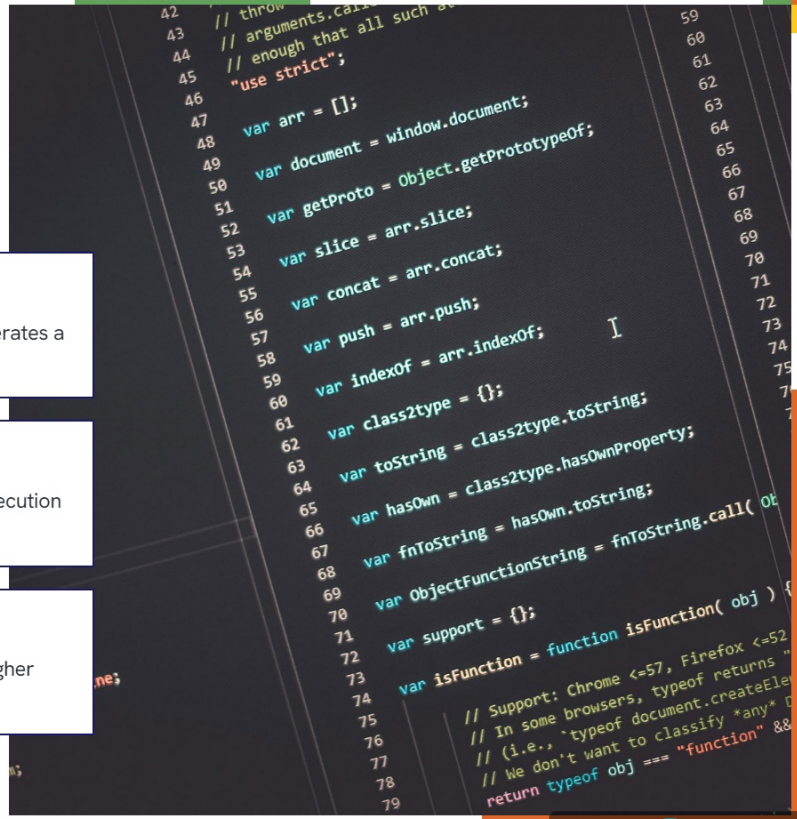
2. **Deployment Trigger Notification**

   Whenever a deployment is triggered, Litmus is notified to initiate automation execution.

3. **Artifact Preparation and Deployment**

   During a deployment event, artifacts are prepared and deployed to the targeted server using Ansible Tower.

Promoting Code through Environments with Litmus



Deployment Completion

Once the deployment is completed and execution is successful, it generates a Litmus event.

Automation Triggering

The generated Litmus event allows automation scripts to trigger on execution machines.

Promotion of Development Code

If the execution is successful, the development code is promoted to higher environments such as QA and Pre-Prod.

Continuous Deployment Process with Litmus

Automation Triggering

Litmus enables the SDLC pipeline to trigger automation scripts automatically.

Threshold Pass Criteria

Deployments proceed only if certain threshold pass criteria are met.

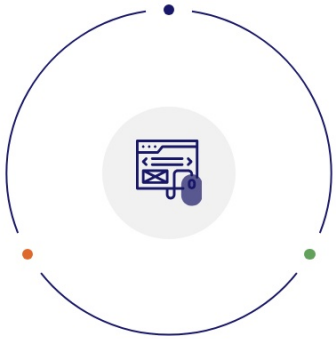
Execution Results Criteria

Build deployments continue from one environment to the next based on the execution results criteria.

How XLR Listens to Developer Code Commits

Configuration of XLR

XLR can be configured to monitor not only the test automation repository but also the developer's code repository.



Methods of Notification

XLR gets notified through either a webhook or polling mechanism.

Notification of Code Commits

When a developer commits new code, such as a bug fix, XLR receives a notification.

Triggering Test Automation Pipeline with XLR

Detecting Code Commit

The process begins by detecting a new code commit in the developer repository.

Integration with Litmus-XLR

The integration of Litmus-XLR is utilized to trigger the automation test pipeline.

Separate Test Automation Repository

The automation test pipeline is stored in a different test automation repository.

Pulling Latest Automation Scripts

XLR pulls the latest version of the automation scripts from the test automation code repository.

Running Tests Against Updated Code

The pulled automation scripts are executed against the newly updated application code.

Workflow Overview of XLR Integration

Developer commits code

Developer commits code to the developer repository (e.g., bug fix).

1

XLR detects new commit

XLR detects the new commit using webhook or polling.

2

Trigger test automation pipeline

XLR triggers the test automation pipeline by pulling the latest test scripts from the test automation repository.

3

Run tests on new code

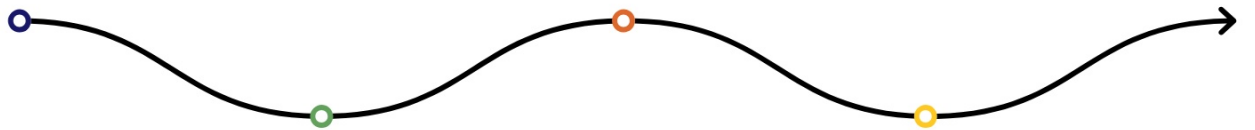
XLR runs the tests on the new developer code.

4

Report test results

The test results (pass/fail) are reported back.

5



Key Integration Points of XLR

Integration with Repositories

XLR can be integrated with both repositories based on SPK.

Listening for Code Commits

XLR listens to the developer code repository for new code commits.

Triggering Test Automation

It triggers test automation from the test automation repository.

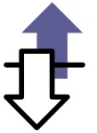
Independent Workflows

This setup allows both developer and test automation teams to maintain independent workflows.

Automatic Testing Trigger

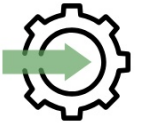
Testing is automatically triggered whenever new changes are made.

Configuration Details for Test Automation Repository



Integration Requirement

The test automation team must provide configuration details to the DevOps team.



Purpose of Integration

This integration allows XLR to trigger and execute test automation scripts.

Event Trigger

Test automation scripts will be executed whenever a new code commit occurs in the developer repository.

Challenges with Deposits Automation

1

Mandatory Selection of Session ID

Automation requires the mandatory selection of the HLLPI API session ID 'A' to recognize the mainframe screen, limiting the automation capabilities.

2

Limitation on Multiple Tabs

Due to the session ID requirement, users cannot utilize multiple tabs with automation tools like LEanFT.

3

Functional Teams' Workaround

Functional teams can open multiple instances without selecting the HLLPI session, which remains blank.

Security Restrictions in Mainframe Applications

1

User Login Restrictions

Once a user logs into one instance of a mainframe application, they cannot log into another instance due to application-level restrictions.

2

LEANFT Runtime Engine Management

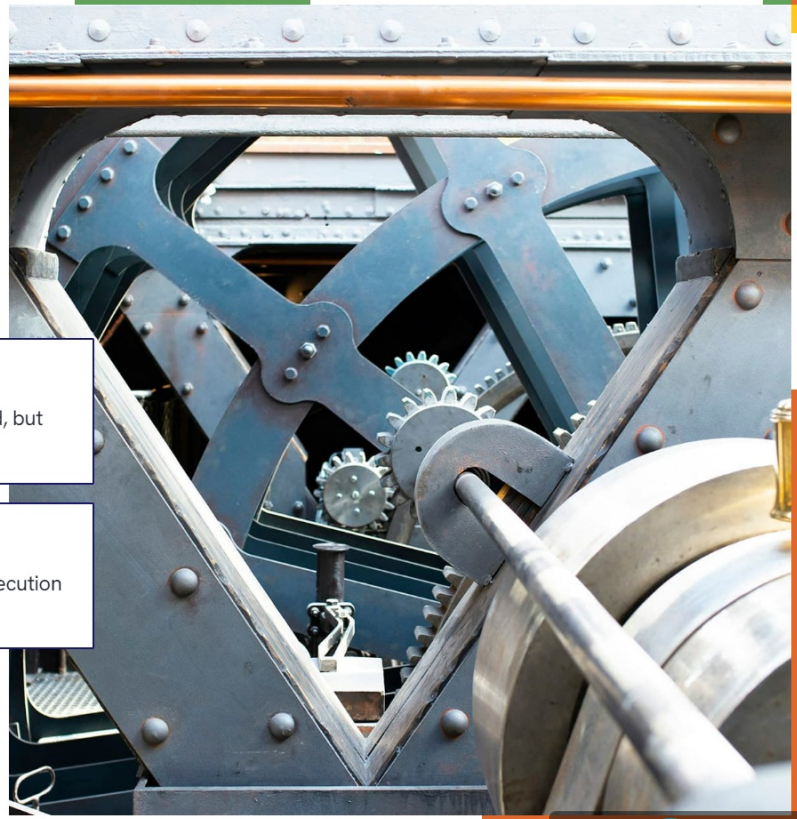
The LEANFT Runtime engine must remain active to successfully trigger the execution of automation scripts.

3

Inactivity Consequences

If the LEANFT runtime engine becomes inactive due to user inactivity, it will enter an idle state, hindering automation processes.

Automation Execution Challenges with HVD



HVD Sleep Mode Impact

When an HVD goes into sleep mode, automation script execution may be triggered, but the tool could fail to execute the scripts, leading to unsuccessful results.

Single User Login Limitation

Due to the single user login requirement for the mainframe application, parallel execution from available machines or machine groups is not feasible.

Advantages of CI with Litmus and XLR CI/CD Pipeline

Continuous Integration Overview

Continuous Integration (CI) is a development practice that automatically tests code changes.

Role of XLR

XLR manages changes to the code and ensures that updates are continuously integrated.

Automated Testing

XLR automatically runs tests on updated code to identify issues early.

Issue Prevention

Early detection of issues helps prevent them from escalating into larger problems.

Reducing Manual Work through Automation Tools

Use of Automation Tools

We are utilizing automation tools such as Selenium, Jenkins, and XLR to enhance our workflow.

1

Reduction of Manual Work

These tools significantly minimize the manual effort required for updates.

2

Thorough Testing of Updates

Every update, whether introducing a new feature or fixing a bug, is rigorously tested.

3

Smooth Integration

The testing ensures that updates are seamlessly integrated into the system.

4