IICS CICD pipelines with GitHub and Jenkins

Prabir Kumar Dhar
Principal Consultant IPS
Agenda

1. Introduction
2. IICS CICD overview
3. Supported platform details
4. Rest api endpoint for CICD
5. Demo
6. Q & A
IICS-Git: Supported Integrations

<table>
<thead>
<tr>
<th>Product</th>
<th>SaaS</th>
<th>Self-hosted</th>
</tr>
</thead>
<tbody>
<tr>
<td>GitHub</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>GitLab</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Bitbucket</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Azure DevOps</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>git Generic repo</td>
<td>N/A</td>
<td>✓</td>
</tr>
</tbody>
</table>
IICS-Git: SaaS Git repo integration

IICS Design Environment

HTTPS

GitHub

SaaS Git Provider

HTTPS

SaaS

Customer Infrastructure

HTTPS

IICS Secure Agent

© Informatica. Proprietary and Confidential.
IICS-Git: On-premise Git repo integration

IICS Design Environment

SaaS

Customer Infrastructure

On Premise Git Servers

GitHub

GitLab
IICS Org-Git integration: Org Level Branch

Developers can work concurrently on separate assets

Org/SubOrg 1 (Branch - Dev)

Org/SubOrg 2 (Branch - Main/Prod)

Org/SubOrg 3 (optional branches)

Each Org can be mapped to different branch in same repo. Branches can be merged in Git (outside IICS)
Recent API enhancements

- Source Control integration APIs

- pull
- pullByCommitHash
- checkout
- undoCheckout
- checkin
- commit
- commitHistory
- sourceControlAction
- Pull status
IICS Orgs - CI/CD with Git + Jenkins

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Cloud Apps
Cloud Sources/Targets

SaaS
Customer Infrastructure

On-Premise Sources/Targets

Connections

© Informatica. Proprietary and Confidential.
Step 1: a. Developer checks in revisions to Dev branch
Step 1: a. Developer checks in revisions to Dev branch

Check in Dev Env

New Commit hash (H1) in Dev branch

Dev Org -> Test Org -> Prod Org

Check in Dev Env

CI Pipeline

CD Pipeline

Informatica. Proprietary and Confidential.
Step 1: b. Git passes hash H1 via webhook to CI Pipeline
Step 1: c. CI retrieves list of assets for H1 via REST API
Step 1: c. CI retrieves list of assets for H1 via REST API

V3/Commit/H1 return assets with type
Step 1: d. CI executes tests via REST API
Step 1: d. CI executes tests via REST API
Step 1: e. CI send status notification to developer

Communicate whether CI tests passed or failed.
Step 2: a. Developer requests promotion of H1 to next Environment

If CI tests passed, request CD workflow to promote H1 to Test branch.
Step 2: b. Git cherry-pick from H1 to Test Branch
Step 2: c. Git passes hash H2 via webhook to CI Pipeline
Step 2: d. Pull by Commit Hash H2 into Test Org
Step 2: e. CI repeats steps 1c, 1d, 1e in Test Environment

Communicate whether CI tests passed or failed

Dev Org

Test Org

Prod Org

Dev RTE

Test RTE

Prod RTE

Connections

SaaS

Customer Infrastructure

Cloud Apps

Cloud Sources/Targets

On-Premise Sources/Targets

Applications

Data Warehouse

Legacy Databases

On-Premise Data & Applications

Informatica. Proprietary and Confidential.
Step 3: a. Developer requests promotion of H2 to next/Prod Env

If CI tests passed, request CD workflow to promote H2 to Prod branch.
Step 3: b. CD and CI pipelines repeat step 2b – 2d for Prod Env
Step 3: c. CI send final status notification to developer

Communicate successful promotion to Prod env

CI Pipeline
CD Pipeline

Dev Org → Test Org → Prod Org

Dev RTE → Test RTE → Prod RTE

Connections

Cloud Apps
Cloud Sources/Targets

SaaS
Customer Infrastructure

On-Premise Sources/Targets

Applications
Data Warehouse
Legacy Databases
On-Premise Data & Applications