

Intelligent CI/CD Platform for SailPoint IdentityIQ: Automated Plugin Deployment & AI-Based Failure Analysis

In many SailPoint IdentityIQ environments, plugin deployment is manual and error-prone, while CI/CD failures require time-consuming log analysis.

To address this, I built an intelligent CI/CD platform that automates plugin deployment and leverages AI to analyze build failures in real time.

1. Install Plugin using Pipeline

PROBLEM

In most SailPoint IdentityIQ setups, plugin deployment is still manual.

- Developers build the plugin → generate ZIP → login to UI → upload → install
- If something goes wrong, we repeat the same steps again
- No version control, no traceability, no consistency across environments

WHAT I DID (SOLUTION)

To solve this, I automated the entire process using Jenkins.

- Connected Git repository directly to Jenkins
- Built a parameter-driven pipeline (environment, plugin name, reinstall option)
- Automated build + packaging (Maven/Ant detection)
- Implemented safe reinstall (uninstall → install → verify)
- Added approval gates for production

Now deployment is reduced to a simple flow: select environment → trigger → deploy

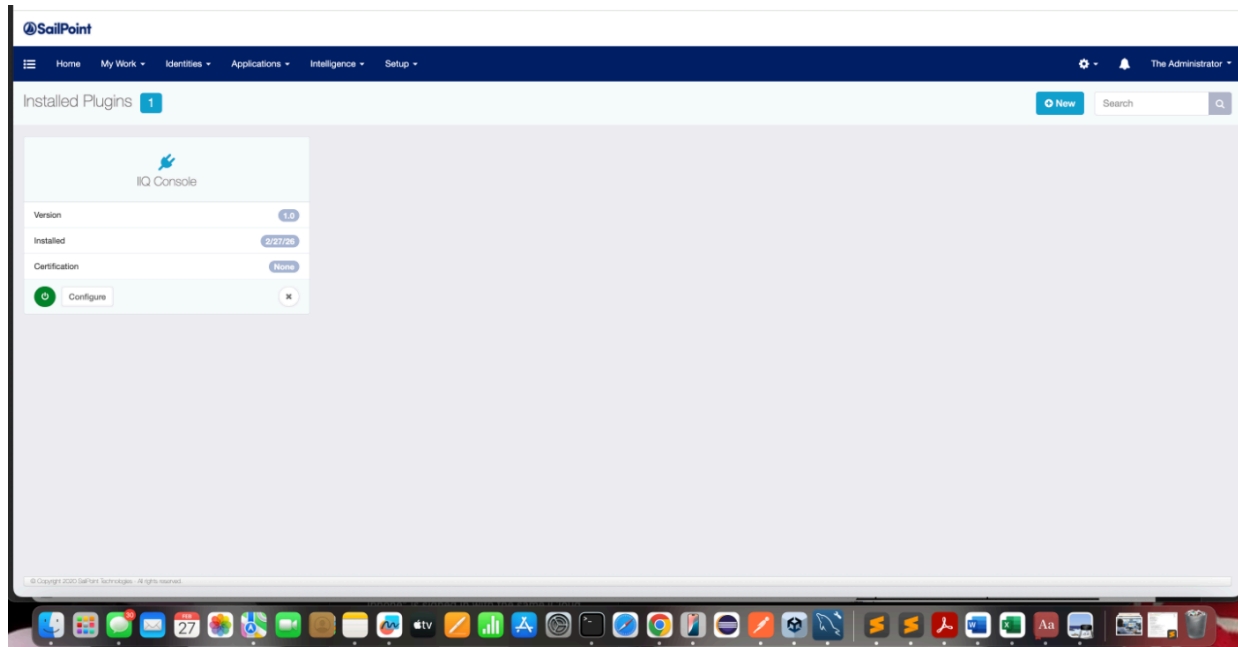
OUTCOME

- No more manual UI uploads
- Same deployment flow across DEV → QA → PROD
- Full traceability of what was deployed and when
- Reduced risk during production releases

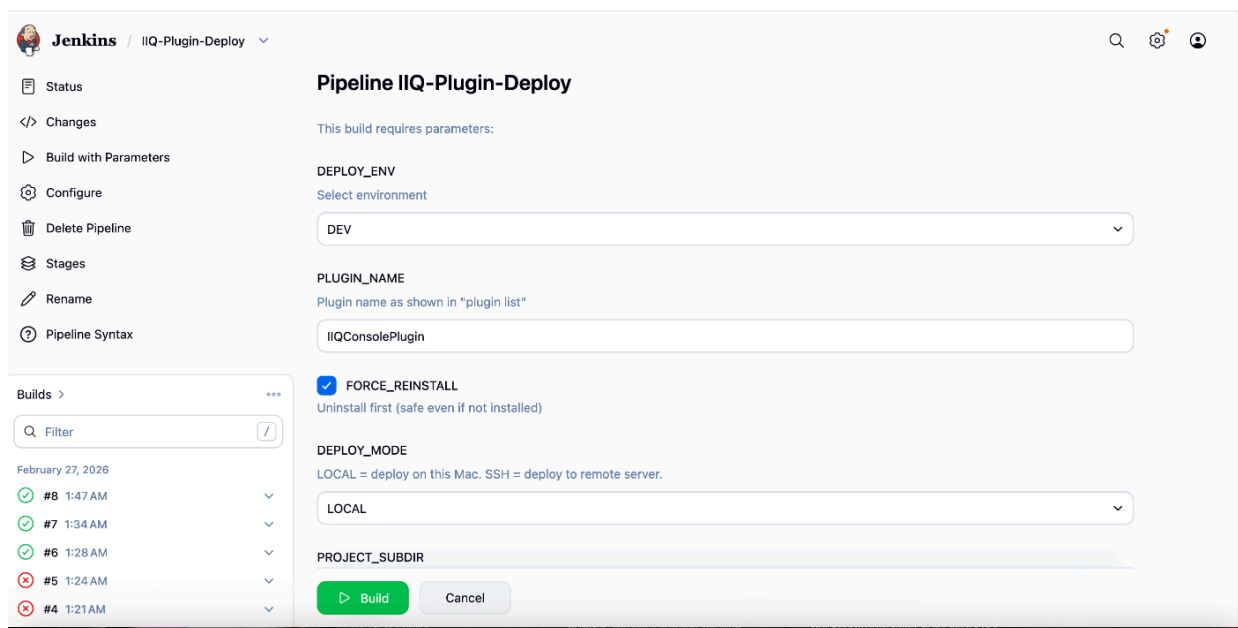
Deployment became **repeatable, safe, and fast**

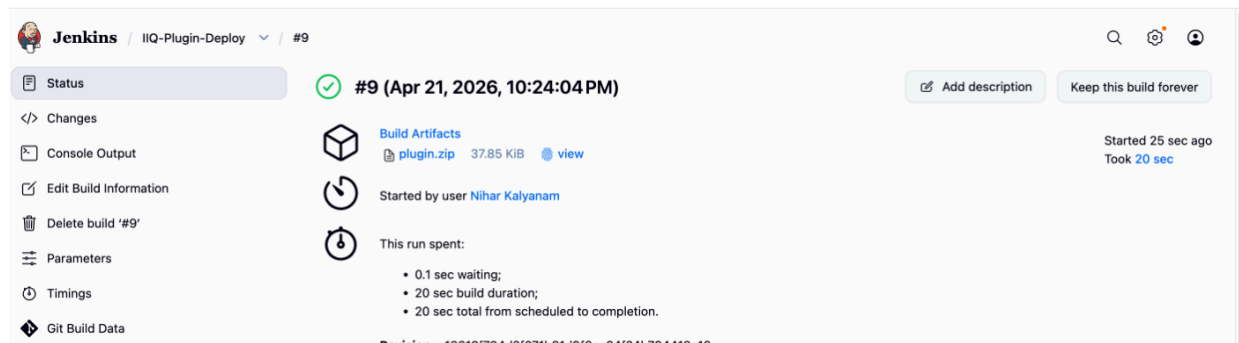
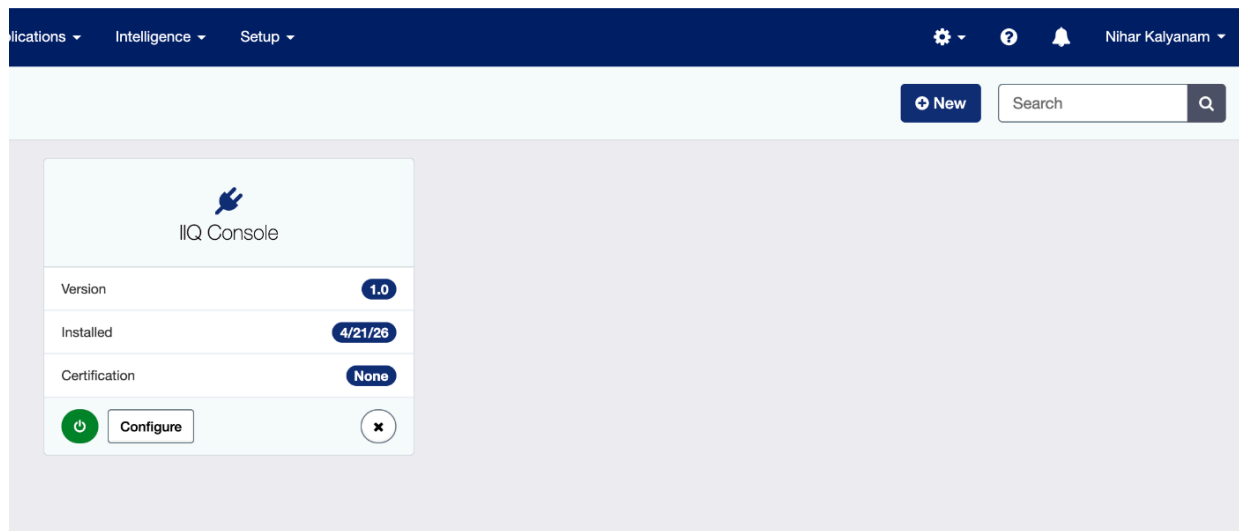
Steps:

Before Jenkins pipeline, we can see that we have IIQ Console plugin with 2/27/2026 date



Running the Jenkins Pipeline to uninstall and import the latest plugin from the GIT to Sailpoint Identityiq Environment.





2. AI-Powered CI/CD Failure Analysis & Interactive Debugging

PROBLEM

Even after CI/CD automation, one major issue remained:

Debugging failures

- Jenkins logs are huge and hard to read
- Engineers spend time scrolling logs
- Root cause is not always obvious
- Same failures keep repeating
- Non-technical users cannot understand failures

This slows down:

- debugging
- releases
- productivity

WHAT I BUILT (SOLUTION)

I built **JenkinsAI**, an AI-powered failure analysis system.

When a build fails:

1. Jenkins sends failure event via webhook
2. Logs are captured and processed
3. AI analyzes the logs
4. System returns:
 - root cause
 - fix suggestion
 - confidence score

On top of that, I added:

“Ask AI” feature

- Users can ask:
 - “Why did this fail?”
 - “How to fix this?”
- Supports multi-step conversation over logs

Also:

- Stores all failures
- Detects patterns
- Shows trends in dashboard

OUTCOME

- No need to manually read logs
- Faster root cause identification
- Debugging becomes simple even for non-experts
- Recurring issues are identified automatically

Failure analysis moved from **manual** → **intelligent**

LIVE FAILURE FEED

Shows real-time failed builds with AI-generated insights including root cause and suggested fixes.

DETAILED FAILURE ANALYSIS

Displays structured breakdown of the issue, including error snippet, reasoning, and recommended resolution steps.

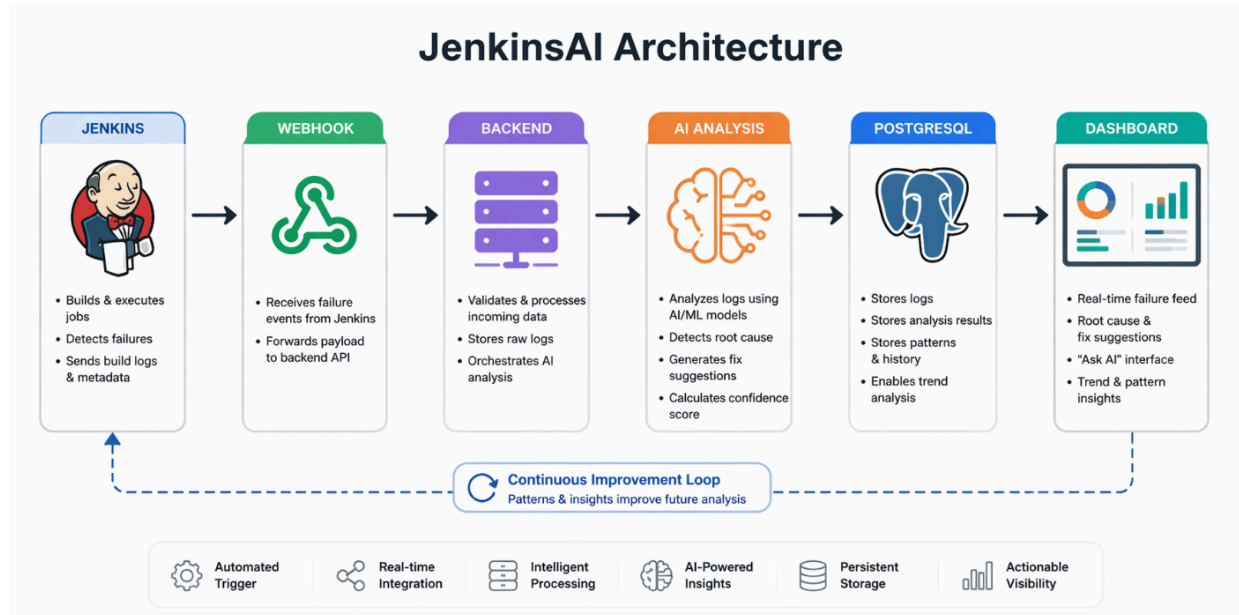
ASK AI

Allows users to interact with build failures using natural language, making debugging conversational and intuitive.

FAILURE PATTERN TRENDS

Highlights recurring issues across builds, helping teams identify and fix systemic problems.

SYSTEM ARCHITECTURE OVERVIEW



- Built an AI-powered failure analysis system integrated with **Jenkins** to automatically capture and analyze build logs on failures.
- Implemented real-time log ingestion using webhook-based integration, enabling immediate analysis without manual intervention.
- Developed AI-driven root cause detection with actionable fix suggestions, reducing the need for manual log debugging.
- Introduced an "Ask AI" interface that allows users to query build failures in natural language and understand issues without reading logs.
- Designed a centralized system to store failure analysis and identify recurring patterns across builds.
- Built a dashboard to visualize failures, insights, and trends, improving visibility and decision-making.

In addition to plugin deployment, I also automated operational pipelines such as Tomcat start/stop/restart to support end-to-end environment management.

Jenkins

Tomcat Server -- Start_Stop

🔍

🔔

👤

Status

</> Changes

▶ Build with Parameters

⚙️ Configure

🗑️ Delete Pipeline

📁 Stages

✏️ Rename

🔍 Pipeline Syntax

Builds >

🔍 Filter

March 21, 2026

❌ #55 2:55 AM

❌ #54 2:36 AM

❌ #53 2:19 AM

❌ #52 2:07 AM

❌ #51 1:59 AM

Pipeline Tomcat Server -- Start_Stop

This build requires parameters:

DEPLOY_ENV
Target environment
DEV

ACTION
Tomcat action
RESTART

TOMCAT_PORT
Optional override. Leave blank to use environment default port

EMAIL_TO
Optional email recipients (comma-separated)

▶ Build

Cancel

Jenkins

Tomcat Server -- Start_Stop

#56

🔍

🔔

👤

Status

</> Changes

📄 Console Output

📄 Edit Build Information

🗑️ Delete build '#56'

⚙️ Parameters

🕒 Timings

🔗 Pipeline Overview

🔄 Restart from Stage

🔄 Replay

📋 Pipeline Steps

📁 Workspaces

← Previous Build

❌ #56 (Apr 21, 2026, 10:47:58 PM)

📄 Add description

Keep this build forever

🕒 Started by user Nihar Kalyanam

Started 4 min 57 sec ago
Took 4 min 20 sec

🕒 This run spent:

- 58 ms waiting;
- 4 min 20 sec build duration;
- 4 min 20 sec total from scheduled to completion.

</> No changes.

REST API

Jenkins 2.541.3

JenkinsAI

Live Feed

Trends

Ask AI

Backend online · Jenkins: localhost:1993

⚡ Live Failure Feed

Auto-refreshes every 15 seconds · 2 failures recorded

Refresh

Tomcat Server -- Start_Stop

Build #56

RUN ACTION

INVALID_HOST_PARAMETER

1 SCRIPT ISSUE

1m ago

▼

Tomcat Server -- Start_Stop

Build #55

RUN ACTION

INVALID_HOST_PARAMETER

1 SCRIPT ISSUE

31d ago

▼

JenkinsAI

Live Feed

Trends

Ask AI

Backend online · Jenkins: localhost:1993

⚡ Live Failure Feed

Auto-refreshes every 15 seconds · 2 failures recorded

Refresh

Tomcat Server -- Start_Stop

Build #56

RUN ACTION

INVALID_HOST_PARAMETER

1 SCRIPT ISSUE

1m ago

▲

ROOT CAUSE

Malformed HOST_IP environment variable with value '127.0.0.1BBBB' causing unreachable health check URL.

ERROR SNIPPET

HEALTH_URL : http://127.0.0.1BBBB:1992/identityId/

FIX SUGGESTION

Correct the HOST_IP environment variable to a valid IP address, e.g., '127.0.0.1'.

SCRIPT ISSUES FOUND IN JENKINSFILE (1)

Invalid HOST_IP environment variable causing malformed URL.

environment block, line with HOST_IP

Change 'HOST_IP = 127.0.0.1BBBB' to 'HOST_IP = 127.0.0.1'.

AI Confidence: 95% Analyzed in 4183ms · Log: 20.0KB

Tomcat Server -- Start_Stop

Build #55

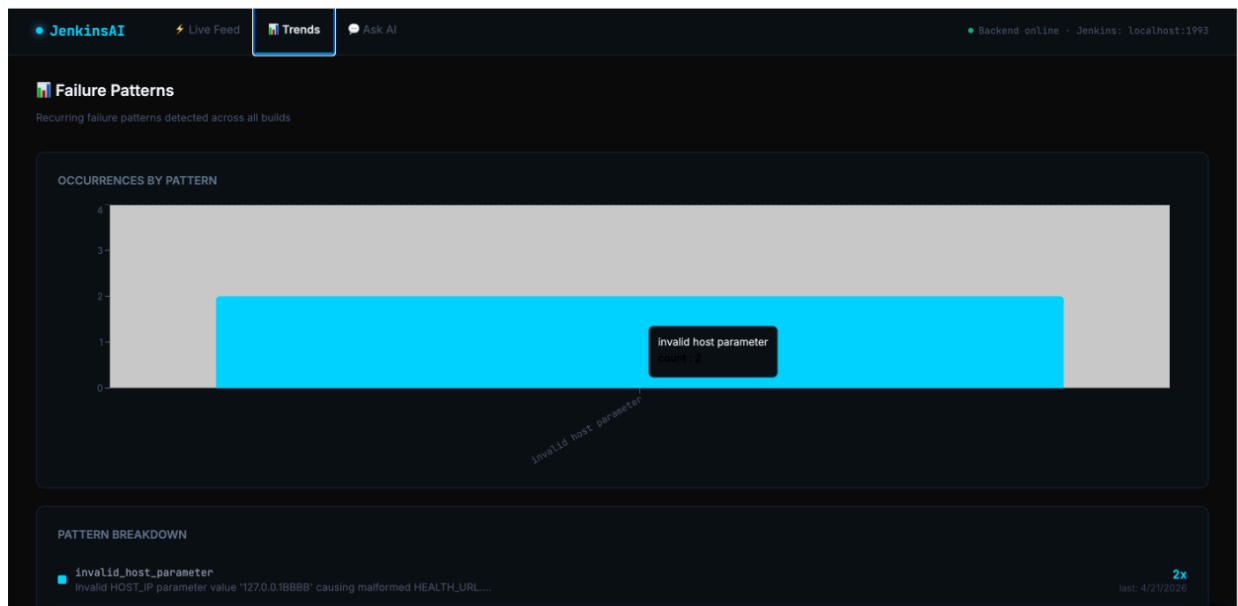
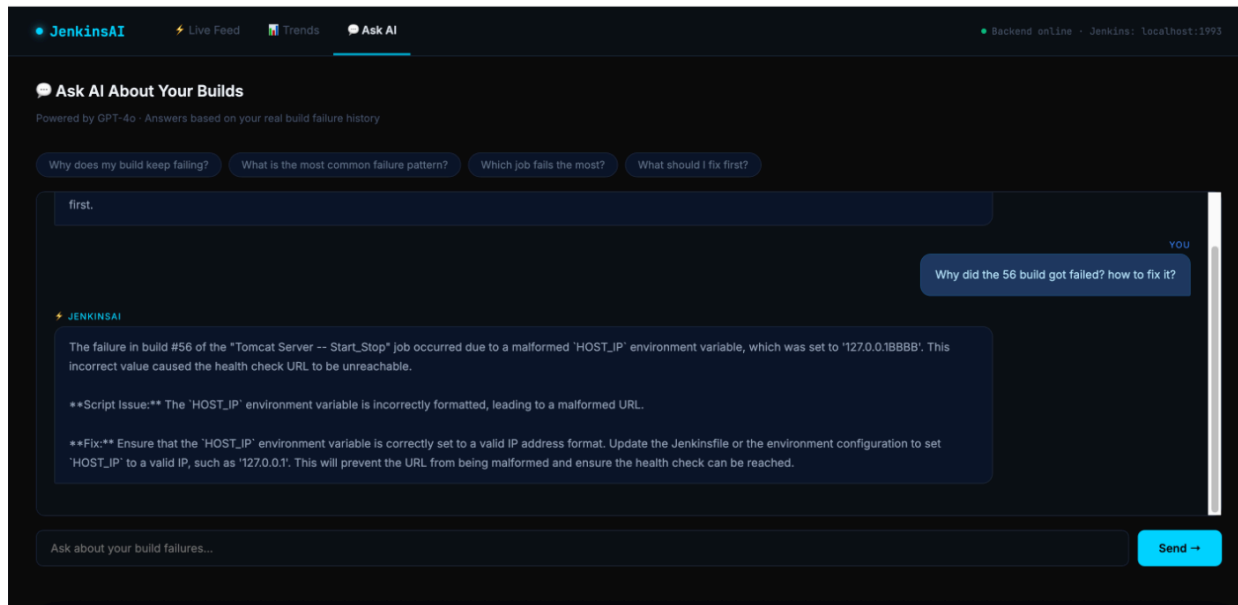
RUN ACTION

INVALID_HOST_PARAMETER

1 SCRIPT ISSUE

31d ago

▼



BUSINESS IMPACT

- Eliminates manual plugin deployment effort
- Reduces CI/CD failure debugging time
- Enables non-experts to understand build failures using AI
- Improves reliability of SailPoint deployments
- Provides centralized intelligence for recurring issues

This approach combines automation and AI to transform traditional CI/CD into an intelligent system that not only deploys efficiently but also understands and explains failures, enabling faster and more reliable delivery.