Databricks Package Installation Troubleshooting: Sonar Nexus Quarantined Dependencies

This document provides a comprehensive guide for engineering teams, including data science and other development groups, to understand, diagnose, and resolve issues related to Databricks package installations when CVS's Sonar Nexus repository quarantines dependencies. The aim is to empower teams to independently address these challenges, thereby minimizing the need for platform team intervention.

1. Understanding the Problem: Sonar Nexus and Package Quarantines

What is Sonar Nexus?

Sonar Nexus (or similar artifact repositories like Artifactory) serves as a central repository for software components, libraries, and build artifacts within CVS. Its critical functions include:

- **Caching:** Accelerates build processes by caching frequently used external dependencies.
- **Security Scanning:** Conducts scans for known vulnerabilities within packages.
- **Policy Enforcement:** Enables the enforcement of CVS's organizational policies related to licensing, security, and approved software versions.

Why Do Packages Get Quarantined?

Packages or specific versions of packages may be "quarantined" or "blocked" within Sonar Nexus for various reasons, such as:

- **Security Vulnerabilities:** The package contains identified security flaws.
- **Licensing Issues:** The package's license is incompatible with CVS's organizational policies.
- **Compliance Breaches:** The package does not adhere to internal compliance standards.
- **Deprecated Versions:** Older versions may be blocked to encourage the adoption of more stable or secure newer versions.

Impact on Databricks Clusters

When a package or its dependency is quarantined, the following impacts may be observed:

- **Installation Failures:** Attempts to install the package on a Databricks cluster (whether via the UI, init scripts, or notebook commands) will fail.

- **Cluster Restart Issues:** If a package previously installed (and whose dependency subsequently became quarantined) is part of the cluster's default libraries or an init script, the cluster may fail to start or restart successfully due to the inability to re-resolve and install the blocked dependency.

2. Case Study: pymc and Quarantined numpy

This section details a common scenario recently encountered by a data science team.

The Scenario

A data science team utilized pymc, a probabilistic programming library, which has a dependency on numpy. A specific version of numpy, required by pymc, was quarantined by CVS's Sonar Nexus.

Initial Symptoms

Upon restarting the Databricks cluster, the pymc installation failed. The observed symptoms included:

- Cluster failing to attach or start.
- Error messages indicating library installation failures.
- pymc code failing to import or execute.

3. Debugging Steps: A Detailed Guide for Self-Resolution

Follow these steps to independently diagnose and resolve package installation issues caused by quarantined dependencies.

Step 1: Review Databricks Cluster Logs

The initial step involves identifying the root cause of the package installation failure.

- **Access Event Logs:**
  1. Navigate to your Databricks cluster's configuration page.
  2. Select the **"Event Log"** tab.
  3. Examine recent events for entries such as **"Library installation failed"** or similar errors related to cluster startup/attachment.
- **Access Driver Logs:**
  1. On the same cluster configuration page, select the **"Driver Logs"** tab.
  2. Review the stderr and stdout outputs. These logs typically contain detailed output from pip install or conda install commands.
  3. **Keywords for Diagnosis:** Search for terms such as "quarantined," "blocked," "denied," "failed to resolve dependency," or specific HTTP error codes (e.g.,

403 Forbidden) indicating issues with package repository access.

Example Log Snippet (Illustrative):ERROR: Could not install packages due to an OSError: [Errno 13] Permission denied:
'/databricks/python/lib/python3.9/site-packages/numpy'
WARNING: You are using pip version 21.2.4; however, version 23.3.1 is available.
You should consider upgrading via the '/databricks/python/bin/python -m pip install --upgrade pip' command.

Failed to install library: pymc
Caused by: com.databricks.backend.daemon.driver.library.LibraryInstallFailed: Library installation failed.
...
[ERROR] Blocked by Sonar Nexus: numpy-1.20.0.tar.gz is quarantined due to [Security Vulnerability CVE-XXXX-YYYY]. Please use a newer version.

In this example, the critical line for diagnosis is [ERROR] Blocked by Sonar Nexus: numpy-1.20.0.tar.gz is quarantined....

Step 2: Identify the Quarantined Package and Version

Based on the log analysis, precisely identify the package and its version that has been quarantined. In the case study, this was numpy==1.20.0.

Step 3: Find an Approved Alternative Version of the Dependency

Once the blocked dependency and version are identified, locate an alternative version that is not quarantined.

- **Search for Newer Versions:**
  - Consult PyPI (Python Package Index) at pypi.org to search for the blocked package (e.g., numpy).
  - Review the available versions to identify a more recent or suitable alternative.
- **Check Quarantined Package List:**
  - Refer to the official CVS Sonar Nexus quarantined package list for known blocked versions: [Insert Link to CVS Quarantined Package List Here]
- **Internal Communication (If Necessary):**
  - If an approved list of alternative versions is not readily available, or if uncertainty persists, a brief consultation with the platform team may be necessary to confirm an approved version. The goal is to obtain a version that is known to be compliant and not quarantined.

Step 4: Install the Approved Dependency First

After an approved, non-quarantined version of the dependency is identified, install it

prior to attempting to install your primary package (pymc).

Two primary methods for library installation on Databricks are available:

- **Method A: Cluster-wide Installation (Recommended for persistent dependencies)**
  1. Navigate to your Databricks cluster's configuration page.
  2. Click on the **"Libraries"** tab.
  3. Select **"Install New"**.
  4. Choose **"PyPI"** as the source.
  5. In the "Package" field, enter the approved version of the dependency, e.g., numpy==1.24.3 (assuming 1.24.3 is the approved version).
  6. Click **"Install"**. The cluster will restart to apply the new library.
- Method B: Notebook-scoped Installation (Suitable for quick testing or specific notebook requirements)
  Within a Databricks notebook, utilize the %pip magic command to install the dependency. This installation is scoped to the current notebook session.
  %pip install numpy==1.24.3

  Note: Replace 1.24.3 with the actual approved version.

Step 5: Install the Primary Package

Once the approved dependency has been successfully installed, proceed with the installation of your primary package (pymc).

- If using Cluster-wide Installation (Method A):
  Add pymc as another PyPI library in the cluster configuration, following the same procedure used for numpy.
- If using Notebook-scoped Installation (Method B):
  In the same notebook, after the numpy installation, install pymc:
  %pip install pymc
  ```pip` will now leverage the already installed, approved `numpy` version, enabling `pymc` to install successfully.

Step 6: Verify Installation

Conclude by verifying that both the dependency and your primary package are
  ```

correctly installed with the desired versions.

In a Databricks notebook, use %pip show:

```
%pip show pymc
%pip show numpy
```

This command will display information about the installed packages, including their versions. Confirm that numpy is the approved version you installed.

4. Debugging Flowchart

This flowchart outlines the self-service debugging process:

```
graph TD
    A[Start: Package Installation Fails or Cluster Fails to Start] --> B{Review Databricks Cluster Logs (Event & Driver)};
    B --> C[Identify Quarantined Dependency & Version (e.g., numpy==X.Y.Z)];
    C --> D{Find Approved Alternative Version of Dependency (PyPI, Internal Lists)};
    D -- If needed --> D_Platform[Consult Platform Team for Approved Version];
    D_Platform --> E[Install Approved Dependency Version First (e.g., numpy==A.B.C)];
    E --> F[Install Original Package (e.g., pymc)];
    F --> G[Verify Installed Versions (%pip show)];
    G --> H[End: Issue Resolved];
```

5. Best Practices and Prevention

This section will be populated following internal alignment with relevant teams to ensure comprehensive and organization-specific guidance.

By diligently following these steps, engineering teams can effectively and independently resolve package installation challenges arising from Sonar Nexus quarantines, thereby minimizing operational disruptions.