Complete Troubleshooting Guide: Connecting Supabase to n8n

Author: Manus AI

Date: July 22, 2025

Version: 1.0

Executive Summary

This comprehensive guide addresses the most common issues encountered when connecting Supabase databases to n8n workflows, with particular focus on resolving the "Host not found" error and other connection failures. Based on extensive research of community forums, official documentation, and real-world troubleshooting cases, this guide provides step-by-step solutions that have been proven to work across different environments and configurations.

The primary cause of connection failures between Supabase and n8n stems from confusion between Supabase's API credentials and PostgreSQL database credentials. Many users attempt to use the wrong credential set or connection method, leading to persistent connection errors that can be easily resolved with the correct configuration approach.

Understanding the Problem

The Root Cause of "Host not found" Errors

The "Host not found" error that appears when connecting n8n to Supabase typically occurs due to one of several fundamental misconfigurations. The most common issue involves using Supabase's API endpoint URL instead of the proper PostgreSQL database host credentials. When users see their Supabase project URL in the format

`https://projectid.supabase.co`, they often mistakenly believe this is the database host they should use for PostgreSQL connections.

However, Supabase provides two distinct sets of credentials for different purposes. The API credentials are designed for frontend applications using Supabase's REST or GraphQL APIs, while the PostgreSQL credentials are specifically for direct database connections through tools like n8n, pgAdmin, or other database clients. This fundamental distinction is crucial to understand before attempting any connection configuration.

Network and Infrastructure Considerations

Modern cloud environments introduce additional complexity through IPv6 networking requirements and SSL certificate validation. Supabase's direct database connections use IPv6 by default, which can cause connectivity issues in environments that don't fully support IPv6 routing. This is particularly common with certain cloud hosting providers and containerized environments where IPv6 support may be limited or disabled.

Furthermore, SSL certificate validation can present challenges when connecting through pooled connections. Different connection pooling modes have varying SSL requirements and certificate chain validation processes that must be properly configured to establish successful connections.

Step-by-Step Solution Guide

Solution 1: Using Transaction Pooler Credentials (Recommended)

The most reliable method for connecting n8n to Supabase involves using the Transaction Pooler credentials, which are specifically designed for applications that make multiple short-lived database connections. This approach has proven successful in the majority of troubleshooting cases documented in community forums.

To obtain the correct Transaction Pooler credentials, navigate to your Supabase project dashboard and locate the "Connect" button prominently displayed at the top of the interface. Clicking this button reveals multiple connection options, each designed for different use cases and environments.

Scroll down to the "Transaction Pooler" section, which is specifically optimized for serverless and automation tools like n8n. The Transaction Pooler provides several advantages over direct connections, including better handling of connection limits, improved performance for short-lived connections, and support for both IPv4 and IPv6 networking.

Within the Transaction Pooler section, you'll find a "View parameters" dropdown that reveals the individual connection components needed for n8n configuration. These parameters include the host address, port number, database name, username, and other essential connection details.

The host address for Transaction Pooler connections typically follows the format `aws-0-[REGION].pooler.supabase.com`, where the region corresponds to your Supabase project's geographic location. This is fundamentally different from the API URL that many users mistakenly attempt to use.

The port number for Transaction Pooler connections is 6543, not the standard PostgreSQL port 5432. This distinction is critical because using the wrong port will result in connection timeouts or "host not found" errors.

The username format for pooled connections includes your project identifier, typically in the format `postgres.yourprojectid`. This extended username format is required for the pooling system to properly route your connection to the correct database instance.

Solution 2: Session Mode Pooler for Persistent Connections

For environments where n8n runs as a persistent service rather than serverless functions, the Session Mode pooler often provides better performance and stability. Session Mode maintains longer-lived connections that are ideal for applications that make frequent database queries over extended periods.

The Session Mode connection string uses port 5432, the standard PostgreSQL port, but connects through Supabase's pooling infrastructure rather than directly to the database. This approach provides the benefits of connection pooling while maintaining compatibility with applications that expect standard PostgreSQL connection behavior.

Session Mode pooler addresses IPv6 connectivity issues that can affect direct connections in certain environments. By routing connections through Supabase's pooling infrastructure, Session Mode provides reliable connectivity regardless of the underlying network configuration.

To configure Session Mode connections, follow the same process of accessing the "Connect" button in your Supabase dashboard, but select the "Session Pooler" option instead of Transaction Pooler. The connection parameters will be similar but with different port numbers and slightly different host addresses.

Solution 3: Direct Connection with IPv6 Considerations

Direct connections to Supabase databases can provide the lowest latency and highest performance, but they require proper IPv6 support in your hosting environment. Direct connections bypass the pooling layer entirely, establishing a direct TCP connection to your PostgreSQL instance.

The direct connection string format follows standard PostgreSQL conventions: `postgresql://postgres:[YOUR-PASSWORD]@db.projectid.supabase.co:5432/postgres`. The key difference from pooled connections is the host address, which uses the `db.` subdomain prefix and connects directly to port 5432.

However, direct connections present several challenges that make them less suitable for many n8n deployments. The primary issue is IPv6 requirement, as Supabase's direct database endpoints only accept IPv6 connections. Many hosting environments, particularly older cloud platforms and containerized deployments, may not have complete IPv6 support.

If your environment supports IPv6 and you prefer direct connections for performance reasons, ensure that your network configuration properly handles IPv6 routing and that any firewalls or security groups allow IPv6 traffic on port 5432.

Configuration Parameters for n8n

PostgreSQL Node Configuration

When configuring the PostgreSQL node in n8n with Supabase credentials, each field must be populated with the correct values extracted from your chosen connection method. The following table provides the exact mapping between Supabase connection parameters and n8n configuration fields:

n8n Field	Transaction Pooler	Session Pooler	Direct Cc
Host	aws-0-[region].pooler.supabase.com	aws-0-[region].pooler.supabase.com	db.proje
Port	6543	5432	5432
Database	postgres	postgres	postgres
User	postgres.projectid	postgres.projectid	postgres
Password	[Your database password]	[Your database password]	[Your da passwor
SSL	Require	Require	Require

The host field should contain only the hostname without any protocol prefixes. Many users make the mistake of including `https://` or `postgresql://` in the host field, which causes connection failures. The host should be entered exactly as shown in the Supabase connection parameters, without any additional formatting.

The database name is consistently "postgres" across all connection methods, as this is the default database created with every Supabase project. While you can create additional databases within your Supabase project, the initial connection should always target the "postgres" database unless you have specific requirements for custom databases.

SSL Configuration Requirements

SSL configuration is mandatory for all Supabase connections, regardless of the connection method chosen. Supabase enforces encrypted connections to protect data in transit and comply with security best practices. The SSL mode should be set to "Require" in n8n's PostgreSQL node configuration.

In some cases, particularly with certain versions of n8n or specific hosting environments, SSL certificate validation may encounter issues with Supabase's certificate chain. If you

encounter SSL-related errors such as "self-signed certificate in certificate chain," you may need to adjust the SSL configuration.

For environments where SSL certificate validation fails, you can try setting the SSL mode to "Prefer" instead of "Require." This configuration attempts to establish an SSL connection but falls back to unencrypted connections if SSL negotiation fails. However, this approach should only be used as a temporary workaround, as unencrypted database connections pose security risks.

Password and Authentication

The database password used for n8n connections is the same password you set when creating your Supabase project or the password you've configured through the database settings. This password is distinct from any API keys or service role keys used for Supabase's REST API.

If you're uncertain about your database password or suspect it may have been changed, you can reset it through the Supabase dashboard. Navigate to Settings > Database and use the "Reset database password" option. After resetting the password, you'll need to update any existing connections that use the old password.

Authentication failures typically manifest as "FATAL: password authentication failed" errors rather than "host not found" errors. If you're seeing authentication errors, double-check that you're using the correct database password and not confusing it with API keys or other credentials.

Common Error Messages and Solutions

"Host not found, please check your host name"

This error message indicates that n8n cannot resolve the hostname you've provided in the connection configuration. The most common causes include using the wrong hostname format, including protocol prefixes in the host field, or attempting to use API URLs instead of database hostnames.

To resolve this error, verify that you're using the correct hostname from the appropriate Supabase connection method. The hostname should not include `https://` or any other protocol prefixes. For Transaction Pooler connections, the hostname should follow the format `aws-0-[region].pooler.supabase.com`.

If you're certain the hostname is correct, the issue may be related to DNS resolution in your hosting environment. Some containerized environments or restricted networks may have limited DNS resolution capabilities. In such cases, try using a different connection method or contact your hosting provider about DNS resolution issues.

"connect ENETUNREACH" with IPv6 Addresses

This error specifically indicates that your environment cannot establish IPv6 network routes to Supabase's direct database endpoints. The error message typically includes an IPv6 address in the format `2600:1f1c:...`, confirming that the connection attempt is using IPv6.

The solution is to switch from direct connections to either Session Mode or Transaction Mode pooler connections, which support both IPv4 and IPv6. Pooled connections route through Supabase's infrastructure that handles the IPv6 connectivity requirements transparently.

Alternatively, if your hosting provider supports IPv6 but it's not properly configured, you may need to enable IPv6 support in your network configuration. However, switching to pooled connections is generally the more reliable solution.

"self-signed certificate in certificate chain"

SSL certificate validation errors occur when the SSL certificate presented by Supabase's pooler endpoints cannot be validated by your environment's certificate authority store. This is more common with older systems or environments with restricted certificate authority configurations.

The immediate solution is to adjust the SSL configuration in n8n to be less strict about certificate validation. However, this should be done carefully to maintain security. If possible, update your environment's certificate authority store to include the certificates needed for Supabase connections.

In some cases, the issue may be related to intermediate certificates in the certificate chain. Supabase's infrastructure uses standard SSL certificates, but some environments may not have the complete certificate chain needed for validation.

Advanced Troubleshooting Techniques

Network Connectivity Testing

Before attempting complex configuration changes, it's valuable to test basic network connectivity to Supabase endpoints. This can help isolate whether the issue is related to network routing, DNS resolution, or configuration problems.

From your n8n hosting environment, attempt to resolve the Supabase hostnames using standard network tools. For Transaction Pooler connections, test DNS resolution of `aws-0-[region].pooler.supabase.com` where the region matches your Supabase project location.

If DNS resolution succeeds but connections still fail, test TCP connectivity to the appropriate ports. Transaction Pooler uses port 6543, while Session Pooler and direct connections use port 5432. Network connectivity issues at the TCP level often indicate firewall restrictions or routing problems.

Connection Pooling Considerations

Understanding how n8n manages database connections can help optimize your Supabase configuration for better performance and reliability. n8n maintains connection pools for PostgreSQL nodes, reusing connections across multiple workflow executions to reduce overhead.

The "Maximum Number of Connections" setting in n8n's PostgreSQL configuration controls how many concurrent connections n8n will maintain to your Supabase database. Setting this value too high can exhaust Supabase's connection limits, while setting it too low may create bottlenecks in high-volume workflows.

Supabase's free tier includes connection limits that may be reached if you configure too many concurrent connections or if connections are not properly released after use. Monitor

your Supabase dashboard for connection usage and adjust n8n's connection pool settings accordingly.

Alternative Connection Methods

If standard PostgreSQL connections continue to fail despite proper configuration, consider alternative approaches that may work better in your specific environment. One effective workaround involves creating a proxy service that handles Supabase connections and exposes a simpler interface for n8n.

This proxy approach involves creating a small web service (such as an Express.js application) that connects to Supabase using standard database libraries and exposes HTTP endpoints for n8n to call. While this adds complexity, it can resolve network and SSL issues that are difficult to address directly in n8n.

Another alternative is to use n8n's HTTP Request node with Supabase's REST API instead of direct database connections. This approach uses Supabase's API credentials rather than database credentials and can be more reliable in environments with network restrictions.

Environment-Specific Considerations

n8n Cloud vs Self-Hosted

The connection method that works best often depends on whether you're using n8n Cloud or a self-hosted n8n instance. n8n Cloud environments have specific network configurations that may affect Supabase connectivity, particularly regarding IPv6 support and SSL certificate validation.

n8n Cloud users should prioritize Transaction Pooler or Session Pooler connections, as these methods are most compatible with n8n's cloud infrastructure. Direct connections may encounter IPv6 routing issues in n8n Cloud environments.

Self-hosted n8n instances have more flexibility in network configuration but may require additional setup for IPv6 support or SSL certificate management. The optimal connection method depends on your hosting environment's capabilities and network configuration.

Docker and Containerized Deployments

Containerized n8n deployments introduce additional networking considerations that can affect Supabase connectivity. Docker networks may not have complete IPv6 support by default, making direct Supabase connections unreliable.

For Docker deployments, ensure that your container network configuration supports the connection method you're attempting to use. Session Mode and Transaction Mode poolers are generally more compatible with containerized environments than direct connections.

If you're running both n8n and other services in Docker containers, consider the network isolation and routing requirements. Supabase connections originate from your n8n container and must be able to reach external networks through your Docker network configuration.

Security Best Practices

Credential Management

Proper management of Supabase credentials is essential for maintaining security while ensuring reliable connections. Database passwords should be stored securely and rotated regularly according to your organization's security policies.

Avoid hardcoding database credentials in n8n workflows or configuration files. Use n8n's credential management system to store Supabase connection details securely. This approach also makes it easier to update credentials across multiple workflows when passwords are rotated.

Consider using environment variables or external credential management systems for storing sensitive connection details, particularly in production environments. This approach provides better security and makes credential rotation more manageable.

Network Security

Supabase connections should always use SSL encryption to protect data in transit. While it may be tempting to disable SSL to resolve connection issues, this creates significant security vulnerabilities that could expose sensitive data.

If you encounter SSL-related connection issues, work to resolve the underlying certificate or network problems rather than disabling encryption. Supabase's SSL implementation follows industry standards and should work correctly in properly configured environments.

Consider implementing additional network security measures such as IP allowlisting if your Supabase plan supports it. This can provide an additional layer of protection by restricting database access to known IP addresses.

Performance Optimization

Connection Pool Tuning

Optimizing n8n's connection pool settings can significantly improve performance and reliability when working with Supabase databases. The optimal configuration depends on your workflow patterns, execution frequency, and Supabase plan limits.

For workflows that execute frequently with database operations, maintaining a larger connection pool can reduce the overhead of establishing new connections. However, be mindful of Supabase's connection limits to avoid exhausting available connections.

Monitor connection usage patterns through both n8n's execution logs and Supabase's dashboard metrics. This data can help you identify optimal connection pool sizes and detect potential issues before they affect workflow performance.

Query Optimization

While connection configuration is crucial for establishing reliable Supabase connectivity, query optimization is equally important for maintaining good performance. Efficient queries reduce connection time and resource usage, improving overall workflow reliability.

Use appropriate indexes on frequently queried columns in your Supabase tables. This is particularly important for workflows that perform lookups or filtering operations on large datasets.

Consider using Supabase's built-in query optimization features and monitoring tools to identify slow queries that may be affecting n8n workflow performance.

Monitoring and Maintenance

Connection Health Monitoring

Implementing monitoring for your Supabase connections can help detect issues before they affect critical workflows. n8n provides execution logs that can reveal connection problems, but additional monitoring may be beneficial for production environments.

Monitor key metrics such as connection establishment time, query execution time, and connection failure rates. Sudden changes in these metrics can indicate network issues, configuration problems, or capacity constraints.

Set up alerts for connection failures or performance degradation to enable rapid response to issues. Early detection of connection problems can prevent workflow failures and minimize business impact.

Regular Maintenance Tasks

Establish regular maintenance procedures to ensure continued reliability of your Supabase-n8n integration. This includes monitoring credential expiration, reviewing connection pool settings, and testing backup connection methods.

Periodically test your connection configuration to ensure it continues to work correctly as both n8n and Supabase evolve. Software updates, infrastructure changes, or configuration drift can affect previously working connections.

Document your connection configuration and troubleshooting procedures to enable faster resolution of future issues. Include specific details about your environment, connection methods, and any custom configurations that were necessary.

Conclusion

Successfully connecting Supabase to n8n requires understanding the distinction between API credentials and database credentials, choosing the appropriate connection method for your environment, and properly configuring SSL and network settings. The most common

"Host not found" errors stem from using incorrect hostnames or credential types, issues that can be resolved by following the step-by-step procedures outlined in this guide.

Transaction Pooler connections provide the most reliable solution for most n8n deployments, offering good performance while avoiding the IPv6 and SSL complications that can affect direct connections. Session Mode poolers offer an alternative for persistent deployments that need longer-lived connections.

By following the troubleshooting procedures and configuration guidelines provided in this guide, you should be able to establish reliable, secure connections between your n8n workflows and Supabase databases. Remember to prioritize security best practices while optimizing for performance and reliability in your specific environment.

References

[1] n8n Community Forum - Cannot connect n8n Postgres Account to my Supabase PostgreSQL database: https://community.n8n.io/t/help-cannot-connect-n8n-postgres-account-to-my-supabase-postgresql-database/86234

[2] n8n Community Forum - Problem connecting to supabase with postgres credentials: https://community.n8n.io/t/problem-connecting-to-supabase-with-postgres-credentials/75027

[3] n8n Documentation - Supabase node common issues: https://docs.n8n.io/integrations/builtin/app-nodes/n8n-nodes-base.supabase/common-issues/

[4] Supabase Documentation - Connect to your database: https://supabase.com/docs/guides/database/connecting-to-postgres

[5] n8n Documentation - Supabase node documentation: https://docs.n8n.io/integrations/builtin/app-nodes/n8n-nodes-base.supabase/