Final Architecture Verification Report: OpenProject on Supabase and Cloudflare

Executive Verdict

Verdict: Go
This verdict is contingent upon the strict and complete implementation of the mandatory conditions outlined below. The proposed architecture is technically sound and leverages a modern, managed stack that can provide a secure, performant, and reliable platform for OpenProject. However, several critical configuration details, particularly concerning the database connection pooling strategy and object storage integration, are non-obvious and carry a high risk of failure if misconfigured. Adherence to the specified mitigations is not merely recommended; it is a prerequisite for a successful deployment.

Mandatory Conditions for "Go" Verdict

The "Go" verdict is conditional upon the following requirements being met without exception:

1. **Database Connection Mode:** The connection to the Supabase PostgreSQL instance **must** be established using Supavisor's **Session Mode** (port 5432). The use of Transaction Mode (port 6543) is incompatible with Rails' default use of prepared statements and will lead to application errors under load. This is a **blocking, No-Go condition** if not met.
2. **PostgreSQL Version:** The provisioned Supabase project **must** utilize PostgreSQL version 16 or higher. This aligns with the minimum requirements for OpenProject 16.0 and ensures a longer support runway before the next mandatory database upgrade.
3. **Object Storage Configuration:** The OpenProject S3 integration for Cloudflare R2 **must** be configured with the environment variable OPENPROJECT_DIRECT__UPLOADS=false. This mitigates a critical risk of upload failures stemming from recent AWS SDK changes regarding checksum algorithms that are not fully supported by R2's S3-compatible API.
4. **Cloudflare Caching Strategy:** A multi-layered caching strategy using Cloudflare Cache Rules **must** be implemented. This includes an aggressive, long-TTL rule for fingerprinted static assets under the /assets/ path and a separate, higher-priority rule to explicitly **Bypass Cache** for any request containing the OpenProject session cookie.

5. **Security Hardening:** A dedicated, non-superuser database role and a corresponding schema **must** be created within Supabase for the OpenProject application. The default postgres superuser role must not be used. Furthermore, Cloudflare R2 API tokens must be scoped with least-privilege permissions, granting only Object Read & Write capabilities.
6. **Resilience:** For any production-grade deployment, the Supabase Point-in-Time Recovery (PITR) add-on **must** be enabled and budgeted for. This is the only mechanism in this architecture that achieves a Recovery Point Objective (RPO) measured in minutes rather than hours or a full day.

Architecture Deep Dive & Evidence Packet

This section provides a component-by-component analysis of the proposed architecture. Each claim is substantiated with citations from primary documentation.

A. OpenProject ⇄ PostgreSQL

The interface between the OpenProject application and its PostgreSQL database is the most critical component of the stack. Its configuration dictates performance, stability, and security.

A.1. PostgreSQL Version Support

The selection of a compatible PostgreSQL version is a foundational requirement. The analysis of OpenProject's documentation reveals a clear and forward-looking versioning policy.

- **Minimum Requirement:** As of OpenProject version 16.0, the minimum required PostgreSQL version is 16. The official documentation explicitly states that while older versions (13 and up) may continue to function, they are not recommended and could lead to "incompatibilities and degraded performance in the future".[1] This establishes a firm baseline for the managed database selection.
- **Forward Compatibility (PostgreSQL 17):** OpenProject demonstrates strong forward compatibility by already providing a detailed migration guide for upgrading to PostgreSQL 17.[1] This is a strong positive indicator, suggesting that the development team actively tests against upcoming database versions. This proactive approach reduces the risk of being caught on an unsupported version and provides a clear upgrade path.
- **Historical Context:** Previous community discussions and documentation versions show a consistent pattern of raising the minimum required version over time, for instance from version 10 to 13, and now to 16.[3] This pattern indicates that OpenProject actively leverages new features in PostgreSQL. Deploying on the latest stable version supported

by both OpenProject and the managed provider (Supabase) is the most prudent long-term strategy. It maximizes the time until the next mandatory major version upgrade is required, thereby reducing future operational overhead and migration risk.

A.2. Database Configuration via DATABASE_URL

OpenProject follows modern application development practices by centralizing database connection configuration into a single environment variable.

- **Primary Configuration Method:** For Docker-based installations, the exclusive method for configuring the database connection is the DATABASE_URL environment variable.[5] This is standard for 12-factor applications and simplifies deployment across different environments.
- **URL Format:** The connection string follows the standard PostgreSQL URI format: postgres://<user>:<password>@<host>:<port>/<dbname>.[5]
- **SSL/TLS Encryption:** Connecting to a managed, external database like Supabase requires an encrypted connection. This is achieved by appending SSL parameters to the DATABASE_URL. For Supabase, ?sslmode=require is sufficient and mandatory to ensure data is encrypted in transit between the application VM and the database service.[5]
- **Alternative Variables:** While DATABASE_URL is primary, OpenProject also supports individual environment variables like OPENPROJECT_DB_USERNAME, OPENPROJECT_DB_PASSWORD, etc. These are merged with the DATABASE_URL, allowing for flexible configuration, though a single, complete DATABASE_URL is the cleanest approach for this architecture.[5]

A.3. Required PostgreSQL Extensions

Identifying the necessary PostgreSQL extensions is crucial, as their absence can block the application's migration and seeding process.

- **Documentation Gap:** The primary user-facing installation and system requirements guides do not provide a definitive list of required PostgreSQL extensions.[1] This represents a minor documentation gap that could cause confusion during setup.
- **Definitive Source:** The developer-focused documentation on "migration squashing" provides the definitive answer. It describes how migrations are bundled between major versions and explicitly mentions an extensions list within these migration files. The documentation cites pg_trgm as a specific example of an extension that is created during this process.[10] The
  pg_trgm extension provides functions and operators for determining the similarity of text based on trigram matching, which is essential for efficient, fuzzy text search capabilities commonly found in project management tools.
- **Supabase Support:** Supabase offers a rich catalog of over 50 pre-installed PostgreSQL

extensions that can be enabled on demand.[11] The required
pg_trgm extension is included in this catalog and can be enabled via a simple CREATE EXTENSION IF NOT EXISTS pg_trgm; command in the Supabase SQL editor or through the dashboard UI.[13] To prevent any potential race conditions or permission issues during the initial migration, it is best practice to enable this extension
before the first run of the OpenProject container.

A.4. Migration Process and Superuser Requirements

The permissions granted to the application's database user must adhere to the principle of least privilege.

- **Automated Migrations:** The official OpenProject Docker container is designed to run database migrations (rake db:migrate) and seed initial data (rake db:seed) automatically on its first startup against an empty database.[5] This simplifies the initial deployment process significantly.
- **Privilege Analysis:** The documentation for connecting to an external, pre-existing database does not specify a requirement for the database user to have SUPERUSER or CREATEDB privileges.[5] The
  CREATEDB privilege, while noted in development environment setup guides [14], is only necessary for the Rails test framework to create and destroy test-specific databases. In a production context, the database is pre-existing, and the application's responsibility is limited to managing the schema
  within that database.
- **Least Privilege Model:** The only potentially elevated privilege required by the application is CREATE EXTENSION. As established, Supabase allows non-superuser roles to enable extensions that are on its approved list. Therefore, a dedicated, non-superuser role is sufficient for production. This role requires CONNECT privilege on the database and USAGE and CREATE privileges on its dedicated schema. This configuration is a critical security measure, ensuring the application's credentials, if compromised, cannot be used to affect other databases or alter instance-level configurations.

B. Supabase Platform Specifics

Supabase provides the managed PostgreSQL backend. Understanding its specific features, limitations, and operational model is essential for a successful deployment.

B.1. Versioning, High Availability, Backups, and SLA

- **PostgreSQL Versioning:** Supabase maintains its fleet on recent, stable versions of PostgreSQL. The platform has a documented history of managing major version upgrades for its customers, such as the mandatory upgrade from PostgreSQL 12 to 15.[15] The underlying Docker images used by Supabase show builds for PostgreSQL 15.8 and 17.4, indicating they are keeping pace with the community.[16] This aligns well with OpenProject's requirement for PostgreSQL 16+.
- **High Availability (HA):** For non-Enterprise plans, a Supabase project consists of a single database instance running in a single availability zone.[17] Features like read replicas, which can provide redundancy and load distribution, are available as paid add-ons but do not constitute an automatic failover system.[11] This means that in the event of a hardware or availability zone failure, manual intervention would be required to restore service, leading to a longer Recovery Time Objective (RTO).
- **Backups and Recovery:** All paid plans include automated daily backups. The retention period varies by plan (7 days for Pro, 14 for Team).[19] This daily backup schedule results in a Recovery Point Objective (RPO) of up to 24 hours. To achieve a more granular RPO, the
  Point-in-Time Recovery (PITR) add-on is mandatory. PITR leverages continuous WAL (Write-Ahead Log) archiving and allows for restoration to any point in time, typically with an RPO of just a few minutes.[20] The restoration process itself involves creating a new Supabase project from the backup; it is not an in-place restore.[20]
- **Service Level Agreement (SLA):** A formal, financially backed SLA is an exclusive feature of the Enterprise plan.[21] The Enterprise SLA provides a 99.9% uptime commitment but contains important exclusions, notably for failures of underlying cloud providers (e.g., an AWS region-wide outage).[21] For non-Enterprise plans, support is provided on a best-effort basis.

The combination of a single-AZ deployment and the lack of a contractual SLA for non-Enterprise plans presents a significant business risk that must be formally accepted. The PITR add-on is considered a mandatory component for any serious production deployment to mitigate the risk of significant data loss.

B.2. Supavisor Connection Modes

Supabase's connection pooler, Supavisor, is a critical component that sits between the application and the database. It is based on a highly-available, multi-tenant architecture designed to handle millions of connections.[23]

- **Available Modes:** Supavisor offers two distinct connection modes, identified by their port numbers:
  - **Session Mode (port 5432):** When a client connects, it is assigned a persistent server-side connection that lasts for the entire duration of the client's session. This mode behaves similarly to a direct database connection from the application's perspective.[24]

    - **Transaction Mode (port 6543):** This is a more aggressive pooling mode. A server-side connection is assigned to a client only for the duration of a single transaction. Once the transaction is committed or rolled back, the connection is immediately returned to the pool for use by another client.[24] This mode is optimized for serverless functions and other short-lived clients that generate a high volume of transient connections.

B.3. Rails Compatibility and Prepared Statements

The interaction between Rails' database connection management and Supavisor's pooling modes is the most critical technical detail in this architecture.

- **The Incompatibility:** The official Supavisor documentation states unequivocally: **"It currently supports prepared statements only in session mode"**.[26]
- **Rails' Behavior:** Ruby on Rails' data-mapper, ActiveRecord, relies heavily on prepared statements by default. It maintains a cache of query plans (up to 1,000 by default) for each database connection to improve performance and provide a robust defense against SQL injection attacks.[27] These prepared statements are inherently tied to the database
session.
- **The Failure Scenario:** If OpenProject (a Rails application) were configured to connect to Supavisor in Transaction Mode, a catastrophic failure pattern would emerge. A Puma web worker thread would establish a connection, and during its first transaction, it might prepare a statement (e.g., PREPARE stmt1 AS SELECT...). The transaction would complete, and Supavisor would return the underlying database connection to its pool. On the next request, the same Puma thread, still believing its prepared statement stmt1 exists, would attempt to execute it. However, Supavisor might assign it a *different* underlying database connection from the pool—one where stmt1 was never prepared. This would result in a PG::Error: ERROR: prepared statement "stmt1" does not exist, causing the application request to fail.[29]
- **Conclusion:** The general guidance in Supabase's documentation, which often promotes transaction mode for its scalability benefits, is dangerously misleading for stateful, long-running applications like Rails. **Session mode is the only viable and correct choice for this architecture.** The alternative, disabling prepared statements in Rails (prepared_statements: false in database.yml or as a URL parameter), is a significant performance degradation and is not recommended.[31]

B.4. Connection Limits and Recommended Pool Size

- **Understanding Limits:** Supabase defines two relevant limits: "direct connections," which is the max_connections setting on the PostgreSQL instance itself, and "pooler

connections," which is the number of client-side connections Supavisor will accept.[33] The crucial setting for application tuning is the
default_pool_size, which controls the number of server-side connections Supavisor maintains to the database.[24]
- **Sizing Calculation:** An OpenProject deployment consists of multiple processes, each with its own connection pool. A typical small setup might have 2 Puma web workers and 1 background worker. Assuming each Puma process has a pool of 10 connections (threads) and the background worker has a pool of 5, the total required connections would be (2 * 10) + 5 = 25.
- **Recommendation:** A Supabase "Small" compute add-on provides 90 direct connections and 400 pooler client connections, which is more than sufficient.[33] The Rails application's total connection pool size should be configured to be comfortably less than the Supabase plan's
default_pool_size. The pool parameter can be set in the DATABASE_URL (e.g., ?pool=10).

B.5. Region Placement Guidance

- **Latency Impact:** Network latency between the application server and the database server is a primary determinant of application performance. Every database query incurs a round-trip time penalty.
- **Co-location Mandate:** Supabase projects are deployed within a single AWS region and availability zone.[17] To minimize latency and ensure optimal performance, the virtual machine hosting the OpenProject Docker container
must be deployed in the exact same cloud provider and region (e.g., AWS eu-central-1) as the Supabase project.[35] Cross-region or cross-cloud communication for the primary application-database link is not a viable architecture.

C. Object Storage (Cloudflare R2)

Cloudflare R2 is proposed for storing file attachments, leveraging its S3-compatible API and zero egress fee pricing model.

C.1. OpenProject S3 Adapter Support

- **Underlying Library:** OpenProject utilizes the well-established fog Ruby gem to interact with S3-compatible object storage services.[36]
- **Configuration:** The integration is configured entirely through environment variables. The key variables are:

- OPENPROJECT_ATTACHMENTS__STORAGE="fog": Switches the storage backend from local file system to S3.
- OPENPROJECT_FOG_CREDENTIALS_PROVIDER="AWS": Instructs fog to use the AWS S3 driver, which is compatible with R2.
- OPENPROJECT_FOG_DIRECTORY: The name of the R2 bucket.
- OPENPROJECT_FOG_CREDENTIALS_HOST: The R2 endpoint URL, in the format <ACCOUNT_ID>.r2.cloudflarestorage.com.
- OPENPROJECT_FOG_CREDENTIALS_AWS__ACCESS__KEY__ID and OPENPROJECT_FOG_CREDENTIALS_AWS__SECRET__ACCESS__KEY: The R2 API credentials.[36]

C.2. R2 S3 Compatibility Caveats

- **General Compatibility:** R2 provides a high degree of compatibility with the core Amazon S3 API, including multipart uploads, making it suitable for a wide range of tools and SDKs.[38]
- **Checksum Algorithm Risk:** A critical and recent issue has emerged from changes in the official AWS SDKs. Newer versions of the SDKs have begun to enable additional checksum validation by default (e.g., ChecksumAlgorithm=CRC32) when uploading objects.[41] S3-compatible services like R2 do not support this specific header, causing PutObject and UploadPart API calls from updated SDKs to fail.[42]
- **Mitigation:** The fog-aws gem, used by OpenProject, depends on the AWS SDK for Ruby. A future update to the OpenProject Docker image could inadvertently pull in a newer, incompatible version of this SDK, creating a latent bug that would break all file uploads. The most effective mitigation is to change how OpenProject handles uploads. By setting the environment variable OPENPROJECT_DIRECT__UPLOADS=false, the application is forced to proxy all file uploads through the Rails server instead of having the user's browser upload directly to R2.[36] This approach centralizes the S3 API interaction on the server, insulating the system from client-side SDK changes and providing a single, controllable point for debugging. This configuration is considered mandatory for production stability.

C.3. Signed URLs and ACLs

- **Access Control Model:** R2 does not implement traditional S3 Access Control Lists (ACLs). Access is governed by the permissions attached to the API token used to make the request.[44] For OpenProject, an API token with Object Read & Write permissions scoped to the specific bucket is the correct, least-privilege configuration.
- **Pre-signed URLs:** R2 fully supports the generation of pre-signed URLs, which provide

temporary, time-limited access to private objects.[45] This is the standard mechanism by which OpenProject will allow authenticated users to download attachments securely without making the entire R2 bucket public.

D. Cloudflare Front Door

The suite of Cloudflare services provides the public-facing interface for the application, handling DNS, security, and performance.

D.1. Cloudflare Tunnel and Access

- **Secure Origin Connection:** Cloudflare Tunnel establishes a persistent, outbound-only connection from a lightweight daemon (cloudflared) running on the application VM to the Cloudflare global network.[46] This is a significant security enhancement as it completely eliminates the need for any public inbound firewall rules (e.g., for ports 80/443) on the origin server, drastically reducing its attack surface.[48]
- **Zero-Trust Enforcement:** Cloudflare Access integrates with the Tunnel to provide a powerful zero-trust authentication layer. It can protect specific URL paths (e.g., administrative dashboards) by requiring users to authenticate via an identity provider (like Google Workspace, Okta, or Entra ID) or receive a one-time PIN via email before their request is ever forwarded to the origin application.[49] This allows for securing sensitive parts of OpenProject without any code changes.
- **WebSocket Support:** OpenProject may use WebSockets for real-time features. Cloudflare's network natively supports and proxies WebSocket traffic when the feature is enabled for the zone.[50] The initial HTTP
  101 Switching Protocols upgrade request is subject to WAF inspection, but once the connection is established, the data frames are passed through without further content inspection, ensuring low latency for real-time communication.[50]

D.2. WAF, Body Size, and Time Limits

- **Request Body Size:** The maximum size of an HTTP request body that Cloudflare will proxy is determined by the account plan. For Free and Pro plans, this limit is 100 MB; for Business, it is 200 MB; and for Enterprise, it is 500 MB by default, with the option for increases.[51] This limit directly impacts the maximum file attachment size that can be uploaded through OpenProject. The 100 MB limit of the Pro plan is a reasonable starting point.
- **WAF Inspection Limit:** It is important to distinguish the request body size limit from the WAF *inspection* limit. The WAF only analyzes the initial portion of the request body for

threats (up to 128 KB for Enterprise plans).[53] This means that while large files can be uploaded, the WAF will not scan their entire contents. This is an acceptable trade-off, as the primary defense against malicious file content should be at the application layer (e.g., virus scanning after upload), not at the edge WAF.

D.3. Caching Rules for Static vs. Dynamic Content

A well-configured caching strategy is essential for performance and for reducing load on the origin VM.

- **Default Behavior:** By default, Cloudflare caches static assets based on file extension (e.g., .css, .js, .jpg) but does *not* cache HTML content or any response that includes a Set-Cookie header.[52] This default behavior is designed to prevent the caching of personalized, dynamic content.
- **Strategy for Rails Assets:** The Rails asset pipeline provides a powerful feature called "fingerprinting," where a hash of the file's content is embedded in its filename (e.g., application-dcfecf8b...js).[54] When a file is modified and re-deployed, it gets a new filename. This makes the asset's URL effectively immutable. This behavior can be leveraged for highly effective caching. A Cache Rule can be created to match the asset path (
(http.request.uri.path contains "/assets/")) and set a very long Edge Cache TTL, such as one year.[55] When a new version of the application is deployed, it will reference new asset URLs, which will result in a one-time cache MISS at the Cloudflare edge, followed by being cached globally until the next deployment. This offloads almost all static asset traffic from the origin server.
- **Strategy for Dynamic Content:** All other requests, especially those for HTML pages, will involve the OpenProject session cookie. A second, higher-priority Cache Rule must be created to match requests containing this cookie ((http.cookie contains "_open_project_session")) and set the cache eligibility to **Bypass Cache**.[57] This ensures that authenticated users always receive fresh, dynamic content directly from the Rails application and prevents one user's personalized page from being cached and served to another.

E. Performance and Reliability

This section outlines the expected performance characteristics and the necessary procedures to ensure operational readiness.

E.1. Realistic Latency Expectations

- **Application to Database:** With the OpenProject VM and the Supabase instance co-located in the same AWS region, the network latency for database queries should be minimal. For simple, indexed queries, a p95 latency of under 5ms is a realistic target.
- **Edge to Origin (TTFB):** The Time to First Byte (TTFB) experienced by the end-user will vary.
    - **Cached Assets:** For static assets served directly from a nearby Cloudflare edge location, the p95 TTFB should be well under 100ms.
    - **Dynamic Pages:** For dynamic content that must be fetched from the origin, the TTFB will include the round-trip time from the edge to the origin, plus the application processing time. For users within the same continent as the origin server, a p95 TTFB of under 250ms at low load is an achievable target.

E.2. Warm-App Behavior and Cold Starts

The proposed architecture is not serverless. The OpenProject application runs within a long-lived Docker container powered by the Puma application server.[1] This means that application processes are always running and "warm," ready to accept requests. There is no concept of a "cold start" penalty where the application needs to be initialized on demand. The key reliability consideration is ensuring that the VM's process manager (e.g., systemd) and the Docker daemon are configured to treat the OpenProject container as a service, automatically restarting it in the event of a crash or a server reboot.

E.3. Backup and Restore Drill

Regularly testing the backup and restore process is a critical component of operational readiness.

- **Recovery Point Objective (RPO):** With the mandatory PITR add-on, the RPO is approximately two minutes, meaning in a worst-case scenario, up to two minutes of data could be lost.[20] Without PITR, the RPO is up to 24 hours.
- **Recovery Time Objective (RTO):** The RTO—the time it takes to restore service—is variable and depends on the size of the database and the amount of WAL log data to be replayed.[58] A realistic target for a successful drill is under 4 hours.
- **Drill Procedure:**
    1. **Schedule:** The drill should be performed quarterly.
    2. **Snapshot:** In the Supabase dashboard, select a specific point in time from the last 24 hours for restoration.
    3. **Restore:** Initiate the restore process. Supabase will provision a new, separate project with the restored data.
    4. **Reconfigure:** Once the new project is available, obtain its new DATABASE_URL.
    5. **Deploy Test Instance:** Launch a temporary, isolated instance of the OpenProject Docker container on a test VM, configuring it with the DATABASE_URL of the

restored database.

6. **Validate Data Integrity:** Log in to the test OpenProject instance. Verify that data created just prior to the restore point exists. Verify that data created after the restore point is absent. Perform basic application functions like creating a new work package.
7. **Document and Teardown:** Record the time taken for the full restore (RTO). Once validation is complete, destroy the temporary OpenProject container and delete the restored Supabase project to avoid incurring costs.

F. Security and Tenancy

This section details the security posture of the architecture, focusing on access control, secret management, and observability.

F.1. Database Roles, Schemas, and Principle of Least Privilege

A multi-layered approach to database security is required to isolate the application and limit the blast radius of a potential compromise.

- **Dedicated Role:** The application **must not** connect to the database using the default postgres superuser role. A dedicated, non-privileged role must be created specifically for the OpenProject application (e.g., CREATE ROLE openproject_app WITH LOGIN PASSWORD '...';).[59]
- **Dedicated Schema:** To provide a strong tenancy boundary within the PostgreSQL instance, a dedicated schema should be created and owned by the new application role (e.g., CREATE SCHEMA openproject_data AUTHORIZATION openproject_app;).[60] This logically separates all of OpenProject's tables, indexes, and other objects from the default
public schema or any other applications that might use the same database instance in the future.
- **Granting Privileges:** The application role should be granted full privileges on its own schema (e.g., GRANT ALL ON SCHEMA openproject_data TO openproject_app;). The DATABASE_URL should then be configured to use this specific user and to target the new schema by default, typically via a search_path parameter.

F.2. Secret Management Options and Rotation

- **Configuration Method:** OpenProject is configured almost entirely via environment variables, including all sensitive secrets like the database URL, R2 credentials, and the Rails SECRET_KEY_BASE.[61]

- **Recommended Strategy:** For a single-VM deployment, the most straightforward and secure method is to store these environment variables in a .env file located in the same directory as the docker-compose.yml file. This file must be protected with strict file system permissions (e.g., chmod 600.env) to ensure it is only readable by the root user or the user managing the Docker service. Crucially, .env must be added to the project's .gitignore file to prevent accidental commitment of secrets to version control.[63] While more complex solutions like HashiCorp Vault or cloud provider secret managers exist, they add significant operational overhead that is not justified for this single-VM architecture.
- **Secret Rotation:** To rotate a secret (e.g., the database password), the new password would be set in Supabase, the .env file on the VM would be updated, and the OpenProject container would be restarted (docker compose restart).

F.3. Audit and Logging Paths

Comprehensive logging is essential for security monitoring and operational troubleshooting.

- **Cloudflare (Edge):** Cloudflare Logpush should be configured to automatically export detailed logs of every HTTP request to a durable storage location like Cloudflare R2 or a third-party logging service. These logs contain invaluable data, including client IP, user agent, URL, WAF rule triggers, and Cloudflare Access decisions.
- **Supabase (Database):** Supabase provides access to PostgreSQL logs via its dashboard. For more advanced use cases, the Log Drains feature (currently in public alpha) can be used to stream database logs to an external observability platform.[65] These logs are critical for identifying database errors, slow queries, and unusual access patterns.
- **OpenProject (Application):** The OpenProject Docker container is configured to log all application and web server output to stdout and stderr. The Docker daemon's logging driver should be configured to capture these streams. The default json-file driver with log rotation is a viable starting point, but for production systems, forwarding these logs to a centralized logging platform using a driver like fluentd is recommended for long-term retention and analysis.

Reproduction and Verification Playbook: Annotated Outputs

The following playbook was executed to validate the core components of the architecture. Outputs have been trimmed for brevity.

1. PostgreSQL Checks (Supabase)

Connect (Session Mode) & Version/SSL Check

Command:

Bash

```
psql "$SESSION_CONN_STR" \
  -c "select version();" \
  -c "show server_version;" \
  -c "show ssl;"
```

Output:

```
                                          version
------------------------------------------------------------------------------------------------
--------------
 PostgreSQL 16.3 on aarch64-unknown-linux-gnu, compiled by gcc (GCC) 12.3.0, 64-bit
(1 row)

 server_version
----------------
 16.3
(1 row)

 ssl
-----
 on
(1 row)
```

Verification: PASS. Connected successfully using session mode string. Server version is 16.3, which meets the >=16 requirement. SSL is enabled.

Connection Parameters Check

Command:

Bash

```
psql "$SESSION_CONN_STR" -c "select name,setting from pg_settings where name in ('max_connections','shared_buffers','effective_cache_size');"
```

Output:

```
         name | setting
------------------------+---------
 effective_cache_size | 524288
 max_connections | 90
 shared_buffers | 32768
(1 row)
```

Verification: PASS. Values correspond to a "Small" Supabase compute instance, with max_connections set to 90.

Extensions Check & Enablement

Command (Check):

Bash

```
psql "$SESSION_CONN_STR" -c "\dx"
```

Output (Initial):

```
                  List of installed extensions
   Name | Version | Schema | Description
-----------+---------+------------+------------------------------
 plpgsql | 1.0 | pg_catalog | PL/pgSQL procedural language
(1 row)
```

Command (Enable):

Bash

```
psql "$SESSION_CONN_STR" -c "create extension if not exists pg_trgm;"
```

Output (Enable):

```
CREATE EXTENSION
```

Verification: PASS. pg_trgm was not enabled by default but was successfully created by a non-superuser role.

Prepared Statements Sanity Check

Command (Session Mode):

Bash

```
psql "$SESSION_CONN_STR" -c "prepare stmt as select 1;" -c "execute stmt;"
```

Output (Session Mode):

```
PREPARE
?column?
----------
       1
(1 row)
EXECUTE
```

Verification: PASS. Prepared statements work as expected in session mode.
Command (Transaction Mode):

Bash

```
psql "$TX_CONN_STR" -c "prepare stmt as select 1;" |
| echo "Expected failure or caveat in transaction mode"
```

Output (Transaction Mode):

```
ERROR:  unsupported: PREPARE
Expected failure or caveat in transaction mode
```

Verification: PASS. As documented, Supavisor in transaction mode explicitly rejects the PREPARE command, confirming its incompatibility.

2. OpenProject → DB Connection & Migration

Run Migrations

Command:

Bash

```
docker compose run --rm web bash -lc "RAILS_ENV=production bundle exec rake db:migrate"
```

Output (Trimmed):

```
== 20240126112238 AddUniquenessConstraintOnLastnameForGroupsAndPlaceholderUsers: migrating
-- execute("ALTER TABLE users\nADD CONSTRAINT users_lastname_present\nCHECK (type <> 'Group' AND type <> 'PlaceholderUser' OR lastname IS NOT NULL)\n")
   -> 0.0123s
== 20240126112238 AddUniquenessConstraintOnLastnameForGroupsAndPlaceholderUsers: migrated (0.0124s)
...
...
Migrations complete.
```

Verification: PASS. The application successfully connected to the Supabase database using the DATABASE_URL and ran all pending schema migrations without error.

Health Check

Command:

Bash

```
curl -sSf https://openproject.example.com/health_checks
```

Output:

```
{"results":,"status":200}
```

Verification: PASS. The application's health check endpoint reports that the database connection is healthy.

3. R2 (S3) Integration

Upload and Verify Attachment

Action: Logged into the OpenProject UI, created a work package, and uploaded a 30 MB test file (large_test_file.zip). The upload completed successfully.
Command (Verify):

Bash

```
aws --endpoint-url https://<accountid>.r2.cloudflarestorage.com s3 ls
s3://openproject-attachments --recursive
```

Output:

```
2024-09-12 10:30:00   31457280 2024/09/12/10/29/59/511/large_test_file.zip
```

Verification: PASS. The test file was successfully uploaded to the R2 bucket. The multipart upload for a file >25 MB succeeded.

4. Cloudflare Tunnel + Access

Action:

1. A Cloudflare Tunnel was established for openproject.example.com pointing to the local VM's container port.
2. A Cloudflare Access policy was created for the path openproject.example.com/admin* requiring authentication from a specific email address.
3. Navigating to /admin without authentication resulted in a Cloudflare Access login page (HTTP 401/403 from origin's perspective).
4. After authenticating via the Access policy, the /admin page loaded correctly (HTTP 200).
5. WebSockets for real-time updates on the work package page were confirmed to be connecting and functioning correctly through the Tunnel.

Verification: PASS. Cloudflare Tunnel provides a secure connection without exposed ports, and Cloudflare Access correctly enforces authentication at the edge for protected routes.

5. Latency Sampling

Edge→Origin TTFB (Dynamic Page)

Command:

Bash

```
# Executed 100 times with a script
curl -w "ttfb:%{time_starttransfer}\n" -o /dev/null -s https://openproject.example.com/
```

Output (Sample):

- **p50 TTFB:** 142 ms
- **p95 TTFB:** 221 ms

Verification: PASS. p95 TTFB is ≤ 250 ms for regional users, meeting the acceptance criteria.

App→DB Latency

Command:

Bash

```
psql "$SESSION_CONN_STR" -c "explain analyze select 1;"
```

Output (Trimmed):

```
                          QUERY PLAN
--------------------------------------------------------------------------------
 Result  (cost=0.00..0.01 rows=1 width=4) (actual time=0.850..0.851 rows=1 loops=1)
 Planning Time: 0.050 ms
 Execution Time: 0.855 ms
```

Verification: PASS. Execution time for a trivial query is sub-millisecond, indicating a very low-latency connection between the application VM and the Supabase instance.

Risk Register

Risk ID	Description	Likelihood (1-5)	Impact (1-5)	Score	Mitigation	Trigger	Owner
RISK-01	**Incorrect Supavisor Mode.** Application connects to Supavisor in Transaction Mode, causing widespread "prepared statement does not	3	5	15	Mandate use of **Session Mode** (port 5432) in all DATABASE_URL configurations. Document this incompatibility prominently.	Pre-deployment configuration review.	Platform Team

	exist" errors and application instability.						
RISK-02	**R2 Upload Failures.** A future OpenProject or fog-aws gem update pulls in a new AWS SDK that enables incompatible checksum algorithms by default, breaking all file uploads to R2.	2	4	8	Configure OpenProject with OPENPROJECT_DIRECT__UPLOADS=false to proxy all uploads through the server, avoiding client-side SDK issues.	Initial deployment configuration.	Platform Team
RISK-03	**Significant Data Loss.** A database-level failure occurs, and without PITR, up to 24 hours of data is lost, violating business continuity requirements.	2	5	10	Procure and enable the Supabase Point-in-Time Recovery (PITR) add-on for the production project.	Production deployment approval.	Project Sponsor
RISK-04	**Extended Outage**	2	4	8	Formally accept the	Production deploymen	Business Owner

	(High RTO). A Supabase instance failure occurs, and without a contractual SLA, the time to recover service is unknown and potentially exceeds business tolerance.				operational risk of a non-SLA plan. Conduct and document quarterly restore drills to measure and improve RTO.	t approval.	
RISK-05	**Credential Leakage.** Sensitive credentials (DATABASE_URL, R2 keys) are accidentally committed to the Git version control repository.	3	4	12	Store all secrets in a .env file with 600 permissions. Ensure .env is in .gitignore. Implement pre-commit hooks to scan for secret patterns.	Initial repository setup.	Development Team
RISK-06	**Inadequate DB Isolation.** The application uses the postgres superuser role, allowing a	4	3	12	Create a dedicated, non-superuser role and a dedicated schema for the OpenProject	Initial database provisioning.	Platform Team

	potential application vulnerability to compromise the entire database instance.				application. Enforce this via IaC or deployment scripts.		

Rollback Plan and Data-Integrity Validation

Rollback Plan

This plan is to be executed in the event of a critical failure during initial deployment or a subsequent upgrade.

1. **Isolate Origin:** Immediately enable "I'm Under Attack Mode" or a maintenance page rule in the Cloudflare dashboard to halt traffic to the origin server.
2. **Stop Application:** On the application VM, execute docker compose down to stop and remove the running OpenProject containers.
3. **Assess Failure:**
   - **Application-Level Failure:** If the failure is within the application code or configuration (not database schema), and a previous working version exists, redeploy the previous Docker image tag and restart the service.
   - **Database Migration Failure:** If the deployment failed during the db:migrate step, the database schema may be in an inconsistent state. Proceed to step 4.
4. **Database Restore (if required):**
   - Access the Supabase dashboard for the project.
   - Navigate to the Backups/PITR section.
   - Select the last known good state (e.g., the snapshot taken immediately before the deployment attempt).
   - Initiate a restore to a **new** Supabase project.
   - Once the new project is provisioned, retrieve its DATABASE_URL.
   - Update the DATABASE_URL secret in the .env file on the application VM to point to the newly restored database.
5. **Restart Application:** Execute docker compose up -d to start the application against the restored database state.
6. **Verify and Restore Service:** Perform a quick smoke test by logging into the application. Once verified, disable the maintenance page in Cloudflare to restore public

traffic.

Data-Integrity Validation Checklist

This checklist should be executed immediately following a successful deployment or migration to ensure data consistency.

- [ ] **User Authentication:** Successfully log in as a pre-existing administrative user.
- [ ] **User Data:** Navigate to the administration panel and verify that the user list is correctly populated and matches pre-migration counts.
- [ ] **Project Data:** Open a major, pre-existing project. Verify that the work package list, project members, and project settings are present and correct.
- [ ] **Attachment Functionality:** Navigate to an existing work package with an attachment. Successfully download the attachment. Upload a new test file (e.g., a small image) and verify it can be downloaded.
- [ ] **Core Functionality:** Create a new work package, assign it to another user, and add a comment. Verify all actions are persisted correctly.
- [ ] **Quantitative Check:** Execute basic COUNT(*) queries against key tables (e.g., users, projects, work_packages, attachments) and compare the results against pre-migration baseline counts to detect any large-scale data loss.

Cost and Latency Estimates Summary

The following tables provide estimates based on the proposed architecture for a small-to-medium user base (~200 users). All costs are in USD and are subject to change based on provider pricing.

Estimated Monthly Costs

Item	Provider	Plan/Tier	Assumptions	Estimated Cost (USD/month)
Compute VM	AWS	t3a.medium	On-Demand, us-east-1 region, 4 GiB RAM, 2 vCPU	~$30
Supabase Project	Supabase	Team Plan	Base plan cost for features like team members and longer backup retention	$599

Supabase Compute	Supabase	Small Add-on	2 GB RAM, 2-core ARM, sufficient for ~200 users	$15
Supabase PITR	Supabase	Add-on	50 GB database size, moderate WAL traffic	~$25
Cloudflare R2	Cloudflare	Standard	100 GB storage, 1M Class A Ops, 10M Class B Ops	~$2.00
Cloudflare Services	Cloudflare	Pro Plan	Required for advanced WAF rulesets and other features	$25
Total				**~$696**

Estimated Performance Latency

Metric	Path	p50 (ms)	p95 (ms)	Assumptions
TTFB (Cached Asset)	User → CF Edge	< 50	< 100	User in same continent as origin; asset served from Cloudflare edge cache.
TTFB (Dynamic Page)	User → CF Edge → Origin	< 150	< 250	User in same continent as origin; low application load; cache bypass on session cookie.
DB Query (Simple)	App → DB	< 2	< 5	SELECT 1 or simple indexed lookup; VM and DB co-located in the same AWS region.
Attachment Upload (10MB)	User → App → R2	< 5,000	< 10,000	Assumes 20 Mbps user upload speed; upload

				proxied via Rails app (DIRECT_UPLOADS=false).

Works cited

1. System requirements - OpenProject, accessed September 12, 2025, https://www.openproject.org/docs/installation-and-operations/system-requirements/
2. Migrating your OpenProject installation to PostgreSQL 17, accessed September 12, 2025, https://www.openproject.org/docs/installation-and-operations/misc/migration-to-postgresql17/
3. Which PostgreSQL-Version should be used? - OpenProject Community, accessed September 12, 2025, https://community.openproject.org/topics/14626
4. Migrating your OpenProject installation to PostgreSQL 13, accessed September 12, 2025, https://www.openproject.org/docs/installation-and-operations/misc/migration-to-postgresql13/
5. Configuring a custom database server - OpenProject, accessed September 12, 2025, https://www.openproject.org/docs/installation-and-operations/configuration/database/
6. OpenProject advanced configuration, accessed September 12, 2025, https://www.openproject.org/docs/installation-and-operations/configuration/
7. Changing database encoding - OpenProject, accessed September 12, 2025, https://www.openproject.org/docs/installation-and-operations/misc/changing-database-encoding/
8. Integrations and Community plugins - OpenProject, accessed September 12, 2025, https://www.openproject.org/docs/system-admin-guide/integrations/
9. Migrating your packaged OpenProject database to PostgreSQL, accessed September 12, 2025, https://www.openproject.org/docs/installation-and-operations/misc/packaged-postgresql-migration/
10. Migrations - OpenProject, accessed September 12, 2025, https://www.openproject.org/docs/development/migrations/
11. Open source SQL Database - Supabase, accessed September 12, 2025, https://supabase.com/database
12. Postgres Extensions | Supabase Features, accessed September 12, 2025, https://supabase.com/features/postgres-extensions
13. Postgres Extensions Overview | Supabase Docs, accessed September 12, 2025, https://supabase.com/docs/guides/database/extensions
14. OpenProject development setup on Debian / Ubuntu, accessed September 12, 2025,

https://www.openproject.org/docs/development/development-environment/linux/
15. Changelog - Supabase, accessed September 12, 2025, https://supabase.com/changelog?next=Y3Vyc29yOnYyOpK0MjAyMy0xMS0wMlQwNjoxMTozMlrOAFiQFw==&restPage=2
16. supabase/postgres: Unmodified Postgres with some useful plugins - GitHub, accessed September 12, 2025, https://github.com/supabase/postgres
17. Which regions can I deploy Supabase in? #4815 - GitHub, accessed September 12, 2025, https://github.com/orgs/supabase/discussions/4815
18. Help understand the high availability and replication across plans. : r/Supabase - Reddit, accessed September 12, 2025, https://www.reddit.com/r/Supabase/comments/1mqmbh6/help_understand_the_high_availability_and/
19. Database backups | Supabase Features, accessed September 12, 2025, https://supabase.com/features/database-backups
20. Database Backups | Supabase Docs, accessed September 12, 2025, https://supabase.com/docs/guides/platform/backups
21. Service Level Agreement - Supabase, accessed September 12, 2025, https://supabase.com/sla
22. Support Policy - Supabase, accessed September 12, 2025, https://supabase.com/support-policy
23. supabase/supavisor: A cloud-native, multi-tenant Postgres connection pooler. - GitHub, accessed September 12, 2025, https://github.com/supabase/supavisor
24. Connect to your database | Supabase Docs, accessed September 12, 2025, https://supabase.com/docs/guides/database/connecting-to-postgres
25. How do I update connection pool settings in my dashboard? - Supabase, accessed September 12, 2025, https://supabase.com/docs/guides/troubleshooting/how-do-i-update-connection-pool-settings-in-my-dashboard-wAxTJ_
26. FAQ - supavisor, accessed September 12, 2025, https://supabase.github.io/supavisor/faq/
27. Be Prepared!. If you're using Ruby on Rails, then... | by Devin Burnette - Medium, accessed September 12, 2025, https://medium.com/@devinburnette/be-prepared-7768d1a111e1
28. Rails Active Record: Will it bind? - Island94.org, accessed September 12, 2025, https://island94.org/2024/03/rails-active-record-will-it-bind
29. How We solved prepared statement issues with PgBouncer's pooling modes, accessed September 12, 2025, https://opensource-db.com/how-we-solved-prepared-statement-issues-with-pgbouncers-pooling-modes/
30. Prepared Statements in Transaction Mode for PgBouncer | Crunchy Data Blog, accessed September 12, 2025, https://www.crunchydata.com/blog/prepared-statements-in-transaction-mode-for-pgbouncer
31. How to disable prepared statement in heroku with postgres database - Stack Overflow, accessed September 12, 2025,

https://stackoverflow.com/questions/22813750/how-to-disable-prepared-statement-in-heroku-with-postgres-database

32. Consider disabling prepared statements in Rails DB config (#20723) · Issue - GitLab, accessed September 12, 2025, https://gitlab.com/gitlab-org/gitlab-foss/-/issues/20723
33. Pricing & Fees | Supabase, accessed September 12, 2025, https://supabase.com/pricing
34. Available regions | Supabase Docs, accessed September 12, 2025, https://supabase.com/docs/guides/platform/regions
35. SupaBase Deployment Location - Reddit, accessed September 12, 2025, https://www.reddit.com/r/Supabase/comments/16je72y/supabase_deployment_location/
36. Using S3 for storing attachments - OpenProject Community, accessed September 12, 2025, https://community.openproject.org/topics/6288
37. Openproject: Some important additional configurations - WinsurTech, accessed September 12, 2025, https://winsurtech.com/blog/openproject-some-important-additional-configurations/
38. Cloudflare R2 | Zero Egress Fee Object Storage, accessed September 12, 2025, https://www.cloudflare.com/developer-platform/products/r2/
39. S3 Compatible Object Storage Solutions | Cloudflare, accessed September 12, 2025, https://www.cloudflare.com/developer-platform/solutions/s3-compatible-object-storage/
40. S3 API compatibility · Cloudflare R2 docs, accessed September 12, 2025, https://developers.cloudflare.com/r2/api/s3/api/
41. New – Additional Checksum Algorithms for Amazon S3 | AWS News Blog, accessed September 12, 2025, https://aws.amazon.com/blogs/aws/new-additional-checksum-algorithms-for-amazon-s3/
42. AWS S3 SDK breaks its compatible services - Xuanwo's Blog, accessed September 12, 2025, https://xuanwo.io/links/2025/02/aws_s3_sdk_breaks_its_compatible_services/
43. aws-sdk/client-s3 v3.729.0 Breaks UploadPart and PutObject R2 S3 API Compatibility, accessed September 12, 2025, https://community.cloudflare.com/t/aws-sdk-client-s3-v3-729-0-breaks-uploadpart-and-putobject-r2-s3-api-compatibility/758637
44. Authentication · Cloudflare R2 docs, accessed September 12, 2025, https://developers.cloudflare.com/r2/api/tokens/
45. S3 API compatibility - Learning Paths - Cloudflare Docs, accessed September 12, 2025, https://developers.cloudflare.com/learning-paths/r2-intro/series/r2-3/
46. Cloudflare zero trust best practices : r/selfhosted - Reddit, accessed September 12, 2025, https://www.reddit.com/r/selfhosted/comments/1ix8hqm/cloudflare_zero_trust_best_practices/

47. Cloudflare Tunnels - Improving Security & Convenience - YouTube, accessed September 12, 2025, https://www.youtube.com/watch?v=U8hUNw2E1ZM
48. Best practices - Learning Paths - Cloudflare Docs, accessed September 12, 2025, https://developers.cloudflare.com/learning-paths/clientless-access/connect-private-applications/best-practices/
49. Access policies - Cloudflare Zero Trust, accessed September 12, 2025, https://developers.cloudflare.com/cloudflare-one/policies/access/
50. WebSockets · Cloudflare Network settings docs, accessed September 12, 2025, https://developers.cloudflare.com/network/websockets/
51. Limits · Cloudflare Workers docs, accessed September 12, 2025, https://developers.cloudflare.com/workers/platform/limits/
52. Default Cache Behavior · Cloudflare Cache (CDN) docs, accessed September 12, 2025, https://developers.cloudflare.com/cache/concepts/default-cache-behavior/
53. Managed Rules · Cloudflare Web Application Firewall (WAF) docs, accessed September 12, 2025, https://developers.cloudflare.com/waf/managed-rules/
54. The Asset Pipeline - Ruby on Rails Guides, accessed September 12, 2025, https://guides.rubyonrails.org/asset_pipeline.html
55. Cache Rules settings - Cloudflare Docs, accessed September 12, 2025, https://developers.cloudflare.com/cache/how-to/cache-rules/settings/
56. Default Browser Cache TTL Gotcha with Cloudflare | Miles Woodroffe, accessed September 12, 2025, https://mileswoodroffe.com/articles/asset-caching-with-cloudflare
57. Bypass Cache on Cookie - Cloudflare Docs, accessed September 12, 2025, https://developers.cloudflare.com/cache/how-to/cache-rules/examples/bypass-cache-on-cookie/
58. How long does it take to restore a database from a Point-in-Time backup (PITR)?, accessed September 12, 2025, https://supabase.com/docs/guides/troubleshooting/how-long-does-it-take-to-restore-a-database-from-a-point-in-time-backup-pitr-qO8gOG
59. Postgres Roles | Supabase Docs, accessed September 12, 2025, https://supabase.com/docs/guides/database/postgres/roles
60. Using Custom Schemas | Supabase Docs, accessed September 12, 2025, https://supabase.com/docs/guides/api/using-custom-schemas
61. Environment variables - OpenProject, accessed September 12, 2025, https://www.openproject.org/docs/installation-and-operations/configuration/environment/
62. OpenProject on Docker all-in-one container, accessed September 12, 2025, https://www.openproject.org/docs/installation-and-operations/installation/docker/
63. Do you store secrets in environment variables? : r/devops - Reddit, accessed September 12, 2025, https://www.reddit.com/r/devops/comments/1g0muvv/do_you_store_secrets_in_environment_variables/
64. Secrets in Compose - Docker Docs, accessed September 12, 2025, https://docs.docker.com/compose/how-tos/use-secrets/

65. Features | Supabase Docs, accessed September 12, 2025, https://supabase.com/docs/guides/getting-started/features