SYNTHAX CODES | FREE ENGINEERING RESOURCE

Multi-Tenant Architecture Checklist.

A practical guide for SaaS founders and CTOs deciding between RLS, schema isolation, and hybrid approaches.

25+ Decision points

7 Architecture areas

3 Isolation models

WHAT'S INSIDE

synthax.codes | v1.0 | February 2026

01 ISOLATION STRATEGY SELECTION

	Row-Level Security	Schema-per-Tenant	Hybrid Approach
Best for	< 1K tenants, uniform schema	Regulated, custom schemas	Mixed small + enterprise
Complexity	Low — single schema	High — N migrations	Medium — two-tier mgmt
Performance	Excellent with indexing	Good, no cross-tenant locks	Variable per tier
Migration	One for all tenants	One per tenant	Shared + selective

- [ ] Identified tenant count projection (6-month, 12-month, 3-year)
- [ ] Evaluated regulatory requirements for data isolation
- [ ] Confirmed whether tenants need custom schema modifications
- [ ] Benchmarked query performance with tenant_id filtering
- [ ] Documented isolation strategy decision and rationale

→ Our default recommendation

Start with RLS. It handles 95% of SaaS use cases, simplifies migrations, and scales to thousands of tenants. Only move to schema isolation for specific regulatory or enterprise customization requirements.

02 DATA LAYER ARCHITECTURE

- [ ] Configured connection pooling (PgBouncer / RDS Proxy)
  500 tenants × 10 connections = 5,000. PostgreSQL limit is ~200 without pooling.
- [ ] Implemented tenant context middleware
  SET app.current_tenant = ? on every request before any query executes.
- [ ] Created RLS policies on all tenant-scoped tables
- [ ] Tested cross-tenant aggregate queries
  How do you answer 'total users across all tenants' with RLS enabled?
- [ ] Verified migration strategy at current tenant count
- [ ] Added tenant_id to all foreign key relationships
- [ ] Indexed tenant_id columns on high-traffic tables

03 AUTHENTICATION & AUTHORIZATION

- [ ] Tenant resolved from subdomain, header, or JWT claim
  e.g. acme.app.com → tenant_id = 'acme' before auth middleware
- [ ] Tenant context injected into every database session
- [ ] RBAC scoped per tenant (admin in Tenant A ≠ admin in Tenant B)
- [ ] Cross-tenant access denied at middleware level
  Defense in depth: even if RLS fails, middleware blocks cross-tenant requests.
- [ ] Session/token invalidation works across tenant switches

- [ ] Super-admin / platform-admin role for internal operations
- [ ] Invitation flow handles users joining multiple tenants

> ⚠ **Critical: Default-deny on missing tenant context**
> If your middleware fails to set tenant_id, the system should return zero rows — not all rows. RLS with current_setting('app.tenant', true) handles this correctly.

04 INFRASTRUCTURE & SCALING

- [ ] Database connection limits documented and monitored
- [ ] Caching strategy is tenant-aware
  Redis keys prefixed with tenant_id. Cache invalidation per-tenant.
- [ ] Background jobs tagged with tenant context
- [ ] Rate limiting applied per-tenant, not globally
- [ ] Monitoring dashboards segment metrics by tenant
  Identify noisy neighbors. Alert on > X% resource consumption.
- [ ] Auto-scaling rules account for tenant growth patterns
- [ ] File storage uses tenant-prefixed paths
  s3://bucket/tenants/{tenant_id}/uploads/ — never shared namespace.
- [ ] CDN and edge caching respect tenant isolation

05 SECURITY & COMPLIANCE

- [ ] Data encryption at rest enabled (AES-256 / RDS encryption)
- [ ] Encryption in transit enforced (TLS 1.2+, no fallback)
- [ ] Audit logging captures tenant_id on every write operation
- [ ] PII handling documented per jurisdiction (GDPR, CCPA)
- [ ] Tenant data deletion fully purges all related records
  Test: delete tenant, verify zero rows in all tables, S3, cache, logs.
- [ ] Penetration testing includes cross-tenant access attempts

06 OPERATIONS & MIGRATION

- [ ] Tenant onboarding automated (provision in < 30 seconds)
- [ ] Tenant offboarding includes data export + purge workflow
- [ ] Database backups support per-tenant restoration
  Can you restore Tenant A without affecting Tenant B?
- [ ] Migration rollback plan tested with production-like data
- [ ] Seed data / demo tenant provisioning automated
- [ ] Runbook documented for tenant isolation failures

07 COMMON MISTAKES TO AVOID

AVOID **Choosing schema isolation ‘for security’ without regulatory need**
RLS with proper policies provides equivalent isolation at 10× lower operational cost.

AVOID **No connection pooling strategy**
500 tenants × 10 connections = 5,000. PostgreSQL stops being happy around 200.

AVOID **Tenant context in application code instead of middleware**
One forgotten WHERE clause = data leak. Middleware-level enforcement is non-negotiable.

AVOID **No cross-tenant query strategy**
When the PM asks ‘how many total users?’ and aggregate queries break RLS.

AVOID **Testing with 3 tenants, deploying to 300**
Migration times, connection counts, and cache sizes scale non-linearly. Load test early.

NEXT STEP

Need help implementing multi-tenancy?

We've built 15+ multi-tenant SaaS products.
Book a free architecture review.

synthax.codes/book-a-call

or email dimitry@synthax.codes

SYNTHAX CODES
Los Angeles | Kentucky | Moldova
Senior engineers | No outsourcing | Moldova IT Park | 5.0 on Clutch