CLOUDNIX TECHNICAL REPORT · CNX-TR-2026-07

Engineering Multi-tenant SaaS Platforms

Elango, Akshay Dalvi, Avinash KS, Ajmal Abubacker and Avinash Nishant

Cloudnix Software Labs, Bengaluru, India

CNX-TR-2026-07 · Version 1.1 · November 2026

Abstract

Multi-tenancy is easy to begin and hard to get right: one missing filter exposes another customer's data, unenforced limits and unbilled seats leak revenue, and plan changes applied before payment invite abuse. We describe how we engineer multi-tenant platforms we have built and run: shared-schema isolation with per-request tenant resolution and namespaced files, caches, settings and keys; tiered, per-seat and subscription plan models with limits enforced at the point of use and plan changes applied only on verified provider events; white-label tenancy; integrity controls and hidden-test grading in an assessment platform; commission snapshots, wallets and double approval in a partner network; and signed documents.

Keywords: *SaaS; multi-tenancy; tenant isolation; billing; metering; webhooks; white-label; proctoring; commission; e-signature*

Contents

1. Executive summary
2. Tenancy models
3. Shared-schema isolation in practice
4. Plans, seats and metering
5. Assessment integrity
6. Partner networks
7. Documents and e-signatures
8. Security
9. Checklist: 15 questions
10. References

1 Executive summary

Multi-tenancy is easy to begin and hard to get right. A single missing filter exposes one customer's data to another; limits that are not enforced and seats that are not billed leak revenue; upgrades applied before payment invite abuse; and each domain adds its own hard problem. This report describes how we engineer multi-tenant platforms we have built and run: shared-schema isolation in a commerce platform and a workflow platform; plans, seats and verified billing; white-label tenancy; integrity in an assessment platform; commission and wallets in a partner network; and signed documents.

Principles.

1. **Resolve the tenant once, early.** Every request is bound to a tenant before logic runs.
2. **Scope everything.** Rows, files, caches, settings and keys.
3. **Enforce limits in the product.** At the point of use, not in a report.
4. **Change plans on verified events.** Never on a browser redirect alone.
5. **Snapshot what money depends on.** Rates and prices are copied onto the transaction.
6. **One release for every tenant.** No forks per customer.

2 Tenancy models

Model	Isolation	Cost	When
Shared schema, tenant key	Logical, per query	Lowest	Many small and mid tenants
Schema per tenant	Stronger	Medium	Fewer, larger tenants
Database per tenant	Strongest	Highest	Regulated or very large tenants

Our platforms use a shared schema with a tenant key on every row, and route special tenants (such as a demonstration organisation) to a separate database [1].

3 Shared-schema isolation in practice

3.1 *Resolution*

In the commerce platform the tenant (a store) is resolved from the host name against a unique domain column; in the workflow platform from the organisation in the signed-in session. The resolved tenant determines templates, a media namespace (`media/<domain>/`), a cache namespace, settings and payment credentials for the rest of the request.

3.2 *Scoping*

The store key appears across all 53 tables of the commerce schema; the organisation key across 35 tables of the workflow schema. Queries add the key explicitly. Because scoping is not enforced by a framework, it is checked in code review and should be covered by tests that attempt cross-tenant access [2].

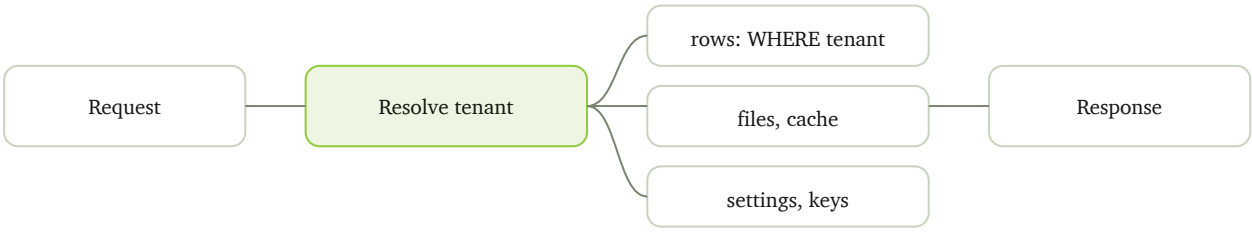


Figure 1. One resolution, everything scoped.

4 Plans, seats and metering

We have run three plan models in production:

Model	Mechanics
Tiered	Four tiers sold for three or twelve months, tax added; product limits enforced at upload; trial flag and expiry date
Per seat	Seats up to a cap of 100 users, unlimited above it; seats added mid-term charged pro rata
Subscription	Monthly plans with trials, a customer portal and trial reminder emails

```
# pro-rata seat addition
charge = seats_added × price_per_seat_per_day × days_remaining_in_term
# plan changes only on a verified provider event
verify(signature_header, raw_body, webhook_secret) → apply(plan, limits)
```

Listing 1. Example exchange or schema.

Plan changes are applied only after the payment provider's signed event is verified against the raw request body [3, 4]. Browser redirects are treated as hints, never as proof.

4.1 *White-label*

Tenants carry their own domain, theme, logo, email templates and documents; the platform's own plans can be sold through the platform itself.

5 Assessment integrity

An assessment platform must make cheating hard and leave evidence when it is attempted. Signals recorded during a test:

Signal	Mechanism
Webcam snapshots	Periodic capture through the browser camera
Screen snapshots	Periodic screen capture
Full-screen exits	Counted
Tab switches	Counted, with a configurable limit
Copy, paste, selection	Blocked after the first question
AI helper extensions	Matching page elements removed every five seconds

Programming answers are written in an in-browser editor with templates for 66 languages, compiled by a separate service and run against hidden test cases, with several attempts allowed.

6 Partner networks

A partner network has five operating levels below the super user: administration, channel partners, dealers (stations), and consumers or industry buyers. Each partner has direct, bulk and online commission rates; each order snapshots the rates in force when it was placed, so later changes never rewrite history. Partners hold prepaid wallets with a full ledger, topped up online; offline payments require two separate approvals. Scheduled jobs fetch daily prices, alert on low station stock and back-fill tax values.

7 Documents and e-signatures

Invoices and agreements are rendered as PDF, stamped with a QR code linking to the record, and signed in one of two modes: Aadhaar eSign through a signing service provider's gateway [5], or a remote digital signature certificate through a server-side signing request [6]. Signing state is tracked per document; signed copies are stored with the record.

8 xOnboarding and provisioning



Figure 2. Self-serve provisioning.

A tenant is created from a verified sign-up (one-time password to phone or email), receives defaults copied from a template tenant (theme and settings in commerce; workflows in the workflow platform), and can attach a custom domain. Provisioning is idempotent: a retried request never creates a second tenant.

8.1 *Trials*

Trials carry an end date and reminder emails before expiry; at expiry, write actions stop while data remains readable, so conversion does not require re-entering anything.

9 xRoles, permissions and API keys

Permissions are defined per function and mode, and checked at each guarded function. Across our platforms, permission checks number more than 1,100 call sites (255 in commerce, 208 in workflow, 106 in assessment). Each staff member has an individual login; API keys are personal, hashed at rest, and inherit the user's permissions.

Layer	Example
User type	super user, administrator, member, submitter, public
Function	create workflow, edit price, export orders
Mode	view, add, edit, delete
Scope	organisation, department, project, store, outlet

10 xReleases and configuration

All tenants run one codebase and one release. Differences between tenants live in configuration: settings, themes, templates, workflows and feature switches. Each change is developed on its own branch, tested on a staging copy and merged to a live branch; rollback is the reverse merge. Database changes are additive where possible so that a release never needs a per-tenant migration.

Why no forks. A fork per customer multiplies every fix and freezes customers on old versions. Configuration and feature switches keep every tenant on the current release.

11 xNoisy neighbours and fairness

- Request-rate limits per IP and user agent, logged and cleaned up on a schedule.
- Per-tenant caches so one tenant's cache churn does not evict another's.
- Heavy work (feeds, imports, exports, reports) in scheduled jobs, not in requests.

- Plan limits on products, users and requests.
- Separate infrastructure for tenants that need it (dedicated cloud or on-premise).

12 xData lifecycle

Stage	Practice
Backups	nightly, three copies, monthly restore drills
Export	tenant data exportable on request
Retention	logs and change tables pruned on schedules
Deletion	expired tenants removed by scheduled jobs after notice

13 Security

Area	Control
Tenancy	Key on every row; namespaces for files, caches, settings and keys
Identity	OTP, Google sign-in, roles and permissions per function
API	Bearer key per user, hashed at rest; OpenAPI-documented endpoints
Billing	Signed provider events verified on the raw body
Money movement	Rates snapshotted; offline payments double-approved

14 Checklist: 15 questions

1	Is the tenant resolved before any logic runs?
2	Do tests attempt cross-tenant access?
3	Are files, caches and keys namespaced?
4	Are limits enforced at the point of use?
5	Are seats charged pro rata?
6	Do plans change only on verified events?
7	Is there one release for all tenants?
8	Can tenants use their own domain and brand?
9	Are demo tenants kept apart?
10	Is assessment evidence recorded, not just blocked?
11	Are hidden tests used for code?
12	Are commission rates snapshotted per order?
13	Do offline payments need two approvals?
14	Are signed documents linked to their records?
15	Can a tenant export its data?

15 xCase study: an assessment platform

A hiring platform for companies screening candidates.

Each company is a tenant with its own branding, question banks and plans. Candidates take timed objective and programming tests; the platform records webcam and screen snapshots, full-screen exits and tab switches, blocks copy and paste after the first question and removes AI helper extensions; programs compile in a separate service and run against hidden tests in 66 languages. Plans are monthly subscriptions with trials, a customer portal and verified billing events. An AI job-description generator runs under a monthly quota per tenant.

16 xTenant resolution, step by step

```
host  = lowercase(strip_www(request.host))
store = SELECT id, theme, currency, gateway FROM store WHERE domain = :host   -- unique
if not store: 404 "can't find the shop"
ctx.tenant = store.id
ctx.paths  = { media: "media/" + host, cache: host + "/cache" }
-- every later query:  ... WHERE store_id = :ctx.tenant
```

Listing 2. Pseudocode.

The same pattern applies with an organisation id from the session in the workflow platform. Tests should create two tenants with identical data and assert that no request for one ever returns the other's rows.

17 xBilling events, worked through

Event	Verification	Effect
Checkout completed	signature over raw body	plan and term set; limits raised
Renewal paid	signature	term extended
Payment failed	signature	grace period; reminders
Subscription cancelled	signature	limits fall at term end
Seats added	signature on pro-rata charge	seat cap raised

Events are idempotent (stored by event id) so retries from the provider never apply twice. The browser return after checkout only shows a waiting state until the event arrives.

18 xWhite-label in depth

Surface	Per tenant
Domain	custom domain mapped to the tenant; certificates per domain
Storefront or portal	theme, logo, colours, widgets
Emails	templates and sender identity
Documents	invoice and agreement templates, numbering series
Apps	store listings under the tenant’s name where offered

19 xAssessment grading, in detail

1. The candidate writes code in a browser editor with a language template.
2. On submit, the code is sent with the language to a separate compile service, isolated from the application.
3. The program runs against hidden test cases stored with the problem; outputs are compared.
4. Attempts per problem are limited; the best result counts.
5. Evidence (snapshots, counts) is stored with the attempt for reviewers.

20 xImplementation roadmap

Phase	Outcome
1. Tenancy design	Model, resolution, scoping, namespaces
2. Plans	Tiers, limits, seats, billing events
3. Pilot tenants	Onboarding, white-label, support
4. Scale	Fairness, dedicated options, data lifecycle

21 xGlossary

Tenant	One customer organisation on the platform.
Tenant key	The column identifying a row's tenant.
Pro rata	Charged in proportion to time remaining.
Billing event	A signed notification from the payment provider.
White-label	The tenant's brand in place of the platform's.

22 References

1. Microsoft. *Architect multitenant solutions on Azure: tenancy models*, Azure Architecture Center.
2. OWASP Foundation. *Application Security Verification Standard*, v4.0, chapter V4: Access Control.
3. Stripe. *Webhooks: verify signatures*, documentation.
4. Razorpay. *Webhooks: validate signatures*, documentation.
5. Controller of Certifying Authorities, India. *e-Authentication guidelines for eSign: Online Electronic Signature Service*.
6. Government of India. *Information Technology Act*, 2000, sections on electronic signatures.

About this report. Prepared by the Cloudnix engineering team from systems we have designed, built and operated. Client and product names are withheld; illustrative values are marked as examples. Cloudnix Software Labs, Bengaluru, India · cloudnix.com.
This report may be shared freely in its entirety.

Document control

Report	CNX-TR-2026-07	Version	1.1
Issued	November 2026	Status	Published
File	cloudnix-multi-tenant-saas-v1.1.pdf		