Workspace = Subdomain + Optional Custom Domain — The Story

0) The baseline
• Your product runs on the apex saltifysaas.com (marketing, login/register).
• A wildcard like *.saltifysaas.com is wired to your app so any something.saltifysaas.com can work instantly.
• A tiny resolver in the backend knows how to map any Host (subdomain or custom domain) → workspace (+ tenant).

1) Sign-up → instant workspace on a subdomain
1) User lands on saltifysaas.com/register.
The form asks: Name, Company, Email, Password, Workspace Subdomain (e.g., alpha).
2) On submit, the backend:
• Creates the auth user.
• Creates a Tenant (company) and a Workspace (e.g., "Main").
• Reserves the global subdomain alpha (unique across the platform).
• Links the user as owner of that tenant/workspace.
3) Redirect to https://alpha.saltifysaas.com/onboarding— it resolves immediately because *.saltifysaas.com is already routed to your app.

What the user experiences: "I typed a subdomain, clicked Create, and boom—I'm in my own workspace URL."

2) How routing works under the hood (every request)
• The Next.js middleware (or edge function) reads the Host header:
• If it's *.saltifysaas.com, it extracts alpha and asks the backend: "Which workspace is this?"
• If it's a custom domain (e.g., pages.acme.com), it asks: "Which workspace owns this hostname?"
• The backend returns { workspaceId, tenantId }.
Middleware stamps those ids into request headers so all pages and APIs automatically scope data to that workspace/tenant.
• If nothing matches, show a branded 404 ("Workspace not found").

Result: every page/API call is multi-tenant safe by default because the workspace/tenant context is always present.

3) Adding a custom domain (later, from settings)
1) The owner opens Workspace → Domains and clicks "Connect domain".
2) They enter pages.acme.com. You show them two DNS steps:
• TXT record for verification (proves ownership)
• CNAME pointing pages.acme.com → your platform edge (so traffic reaches you)
3) They hit Verify. Your backend (or your host like Vercel/Cloudflare) confirms DNS has been set and issues SSL.
4) Mark the domain verified and active.
From now on, https://pages.acme.com serves the same workspace as https://alpha.saltifysaas.com.

Nice touch: let them choose a Primary domain. That becomes the canonical in emails/links.

4) Everyday use (multi-workspace, branding, links)
• A user can belong to multiple tenants/workspaces. The App Launcher shows each by name and

domain; clicking takes them straight to that workspace's host.
• Branding (logo/colors) is stored per workspace. On request, the server reads x-workspace-id and renders the right theme instantly—so alpha.saltifysaas.com and pages.acme.com can look identical or different, your choice.
• Emails & deep links always use the primary domain for that workspace (custom if set, otherwise the subdomain).

5) Safety rails you'll have from day one
• Reserved subdomains: block www, api, cdn, docs, etc.
• Global uniqueness: no two workspaces can claim the same subdomain.
• Domain hijack prevention: a custom domain is inactive until DNS TXT proves ownership and SSL is issued.
• Strict scoping: every DB/API call receives tenantId/workspaceId from middleware; server checks these before returning data.
• Graceful errors: unknown host → workspace 404; domain pending → show "Domain not verified yet" page (optional).

6) Operator view (your side)
• You keep a simple "Host map" in your DB:
• workspaces.subdomain (e.g., alpha)
• domains[] (e.g., pages.acme.com, verified or not)
• A tiny Host Resolver service answers two questions fast:
• "Given alpha.saltifysaas.com, which workspace is this?"
• "Given pages.acme.com, which workspace is this?"
• Your edge provider (Vercel/Cloudflare) handles routing + certificates once you tell it about the domain; your app handles who owns what.

7) The UX flow, as a user
• Day 0: I sign up, choose alpha → I'm instantly inside alpha.saltifysaas.com.
• Day 1: I invite my team, build forms/pages.
• Day 2: I add my domain pages.acme.com, follow DNS steps, click Verify → now my clients see my brand at my URL.
• Always: bookmarks, emails, and deep links open on the correct host with the right data and theme.

8) Rollout plan (short, then sweet)
Phase 1 — MVP
• Middleware + Host Resolver
• Register flow that captures subdomain and provisions tenant/workspace
• Auto-redirect to subdomain.saltifysaas.com
• Basic reserved-name checks

Phase 2 — Custom Domains
• Domains settings UI (add → show DNS instructions → verify)
• Provider integration (Vercel/Cloudflare) for SSL & activation
• Primary domain toggle + canonical link generation

Phase 3 — Polish & scale
• Workspace switcher, branded error pages, analytics per domain
• Rate-limits, audit logs, export, webhooks

9) What changes in your code (conceptually)
• Frontend: Add subdomain field in Register; add middleware that stamps x-workspace-id/x-tenant-id; small Domains settings page.
• Backend: Add a /resolve/host endpoint; extend /auth/register to accept a chosen subdomain and create the workspace; add minimal /domains endpoints for add/verify/list.
• Infra: One-time wildcard for *.saltifysaas.com. For custom domains, your provider APIs handle certs after you confirm DNS.