Authentication (Auth0) & Billing (Stripe) — Implementation Plan

Crivline — Feature Flag SaaS Stack: Next.js 15 (App Router), Prisma, PostgreSQL, Redis, Auth0, Stripe Plans: Free ($0) / Pro ($29/mo) / Team ($69/mo)

What Exists Today

Area	Status
Prisma schema (User, Tenant, Subscription, Usage)	Exists, needs Plan enum update
Auth0 integration	Nothing — no SDK, no routes, no middleware
Stripe integration	Nothing — no SDK, no routes, no webhooks
Dashboard auth	None — all routes open, hardcoded user
Onboarding flow	None
Billing UI	None (landing page has pricing component)

Phase 1: Schema Migration (do first)

1.1 Update Plan Enum

The schema currently has `FREE / STARTER / PRO`. Update to match the landing page:

```
enum Plan {
  FREE
  PRO
  TEAM
}
```

- `STARTER` → `PRO`
- `PRO` → `TEAM`
- If you have existing data rows using the old values, you'll need a SQL data migration

1.2 Add Auth0 ID to User Model

```
model User {
  auth0Id String? @unique @map("auth0_id")
  // ... keep all existing fields
}
```

This links the Auth0 `sub` claim (e.g., `auth0|abc123`) to the local User. Used for lookup on every authenticated request.

1.3 Run Migration

```
cd packages/db
npx prisma migrate dev --name auth-and-plan-tiers
```

Files to touch

- `packages/db/prisma/schema.prisma`
- `packages/db/prisma/seed.ts` (update if it references old Plan values)

Phase 2: Auth0 Setup

2.1 Auth0 Dashboard (external setup)

1. Create an Auth0 **Application** (type: Regular Web Application)
2. Set **Allowed Callback URLs**: `http://localhost:3000/api/auth/callback`
3. Set **Allowed Logout URLs**: `http://localhost:3000`
4. Note your **Domain**, **Client ID**, **Client Secret**

2.2 Install SDK

```
cd apps/dashboard
pnpm add @auth0/nextjs-auth0
```

2.3 Environment Variables

Add to `.env.example` and create `.env.local`:

```
AUTH0_SECRET=<random 32+ char string - run `openssl rand -hex 32`>
AUTH0_BASE_URL=http://localhost:3000
AUTH0_ISSUER_BASE_URL=https://YOUR_TENANT.auth0.com
AUTH0_CLIENT_ID=<from Auth0 dashboard>
AUTH0_CLIENT_SECRET=<from Auth0 dashboard>
```

2.4 Auth API Route (catch-all)

Create `apps/dashboard/app/api/auth/[auth0]/route.ts`:

```
import { handleAuth } from '@auth0/nextjs-auth0';

export const GET = handleAuth();
```

This single file gives you: - `GET /api/auth/login` — redirects to Auth0 login page - `GET /api/auth/callback` — handles the OAuth callback - `GET /api/auth/logout` — logs the user out - `GET /api/auth/me` — returns the user's session data

2.5 Middleware (route protection)

Create `apps/dashboard/middleware.ts`:

```
import { withMiddlewareAuthRequired } from '@auth0/nextjs-auth0/edge';

export default withMiddlewareAuthRequired();

export const config = {
  matcher: ['/dashboard/:path*', '/onboarding/:path*'],
};
```

This protects all `/dashboard/*` and `/onboarding/*` routes. Unauthenticated users get redirected to `/api/auth/login` automatically.

2.6 Auth Provider (root layout)

Wrap your root layout with `<UserProvider>`:

```
// apps/dashboard/app/layout.tsx
import { UserProvider } from '@auth0/nextjs-auth0/client';

export default function RootLayout({ children }: { children: ReactNode }) {
```

```
  return (
    <html lang="en">
      <body>
        <UserProvider>{children}</UserProvider>
      </body>
    </html>
  );
}
```

This makes the `useUser()` hook available in all client components.

Files to create/modify

File	Action
`apps/dashboard/app/api/auth/[auth0]/route.ts`	Create
`apps/dashboard/middleware.ts`	Create
`apps/dashboard/app/layout.tsx`	Modify (add UserProvider)
`.env.example`	Modify (add Auth0 vars)

Phase 3: User Provisioning & Onboarding

3.1 First-Login Detection

After Auth0 authenticates a user, check if they exist in your DB:

- In a server component or the Auth0 `afterCallback` hook, look up User by `email` (from the Auth0 session)
- **Found** → continue to dashboard
- **Not found** → redirect to `/onboarding`

The simplest approach: check this in a dashboard layout server component. If the user has no DB record yet, `redirect('/onboarding')`.

3.2 Onboarding Wizard (3 steps)

Route: `apps/dashboard/app/onboarding/page.tsx` (client component with local step state)

Step 1 — Organization - Text input: organization name - Auto-generate slug from name (lowercase, hyphens, no special chars) - Validate: slug must be unique

Step 2 — First Project - Text input: project name - Optional: description - Inform user: "We'll auto-create Development, Staging, and Production environments"

Step 3 — Invite Team (optional) - Text input: comma-separated email addresses - "Skip for now" button - Store invites in DB but don't send emails yet (email integration later)

On final submit — server action or API route:

Everything in a Prisma `$transaction`:

1. Create `Tenant` (name, slug)
2. Create `User` (email, name, avatarUrl from Auth0 profile, role: `OWNER`, auth0Id from session `sub`, linked to tenant)
3. Create `Subscription` (tenantId, plan: `FREE`, status: `ACTIVE` — no stripeCustomerId yet, Stripe customer gets created lazily on first checkout)
4. Create `Project` (name, slug, linked to tenant)

5. Create 3 × `Environment`:
   - Development (slug: `dev`, color: `#22C55E`, sortOrder: 0)
   - Staging (slug: `staging`, color: `#EAB308`, sortOrder: 1)
   - Production (slug: `prod`, color: `#EF4444`, isProduction: true, sortOrder: 2)
6. Redirect to `/dashboard`

Files to create

File	Action
`apps/dashboard/app/onboarding/page.tsx`	Create
`apps/dashboard/app/onboarding/_components/step-org.tsx`	Create
`apps/dashboard/app/onboarding/_components/step-project.tsx`	Create
`apps/dashboard/app/onboarding/_components/step-invite.tsx`	Create
`apps/dashboard/app/api/onboarding/route.ts`	Create

Phase 4: Wire Auth into the Dashboard

4.1 `getCurrentUser()` Helper

Create `apps/dashboard/lib/auth.ts`:

```ts
import { getSession } from '@auth0/nextjs-auth0';
import { prisma } from '@crivline/db';

export async function getCurrentUser() {
  const session = await getSession();
  if (!session?.user) return null;

  const user = await prisma.user.findUnique({
    where: { auth0Id: session.user.sub },
    include: {
      tenant: {
        include: { subscription: true },
      },
    },
  });

  return user;
}
```

Use this at the top of every server component in the dashboard to get the current user + tenant + subscription.

4.2 Tenant Scoping

Every Prisma query in the dashboard needs to be scoped to the current tenant. Example:

```ts
// Before (fetches ALL flags)
const flags = await prisma.flag.findMany();

// After (fetches only this tenant's flags)
const user = await getCurrentUser();
const flags = await prisma.flag.findMany({
```

```
  where: { project: { tenantId: user.tenantId } },
});
```

Go through every page and add this scoping: - `apps/dashboard/app/dashboard/page.tsx` - `apps/dashboard/app/dashboar`
- Any future pages

4.3 Sidebar — Real User Data

Replace the hardcoded "Alex Rivera" in `sidebar.tsx` with real user data. Either: - Pass user as a prop from a server component parent - Or use Auth0's `useUser()` hook (gives Auth0 profile data — name, picture, email)

Files to create/modify

File	Action
`apps/dashboard/lib/auth.ts`	Create
`apps/dashboard/app/dashboard/_components/sidebar.tsx`	Modify
`apps/dashboard/app/dashboard/page.tsx`	Modify
`apps/dashboard/app/dashboard/flags/page.tsx`	Modify

Phase 5: Stripe Integration

5.1 External Setup (Stripe Dashboard)

1. Create **Products**:
   - **Crivline Pro** — two Prices: `$29/mo` and `$290/yr`
   - **Crivline Team** — two Prices: `$69/mo` and `$690/yr`
2. Note all 4 Price IDs (`price_xxx`)
3. No product for Free — Free is the absence of a subscription

5.2 Install & Configure

```
cd apps/dashboard
pnpm add stripe
```

Create `apps/dashboard/lib/stripe.ts`:

```
import Stripe from 'stripe';

export const stripe = new Stripe(process.env.STRIPE_SECRET_KEY!, {
  apiVersion: '2024-12-18.acacia', // pin the version
});
```

Add to `.env.example`:

```
STRIPE_SECRET_KEY=sk_test_...
STRIPE_WEBHOOK_SECRET=whsec_...
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_test_...
```

5.3 Plan Limits Config

Create `apps/dashboard/lib/billing/plans.ts`:

```
import { Plan } from '@crivline/db';
```

```ts
export const PLAN_LIMITS = {
  [Plan.FREE]:  { maxFlags: 5,  maxProjects: 1,  maxEnvironments: 2,  maxMAU: 1_000 },
  [Plan.PRO]:   { maxFlags: -1, maxProjects: 5,  maxEnvironments: 5,  maxMAU: 10_000 },
  [Plan.TEAM]:  { maxFlags: -1, maxProjects: -1, maxEnvironments: -1, maxMAU: 50_000 },
} as const;

// Map Stripe Price IDs → Plan (fill in your real price IDs)
export const PRICE_TO_PLAN: Record<string, Plan> = {
  'price_pro_monthly':  Plan.PRO,
  'price_pro_annual':   Plan.PRO,
  'price_team_monthly': Plan.TEAM,
  'price_team_annual':  Plan.TEAM,
};
```

`-1` means unlimited.

5.4 Checkout API Route

`apps/dashboard/app/api/billing/checkout/route.ts`

1. Authenticate (read Auth0 session)
2. Get current user + tenant
3. Check: does tenant already have an active subscription? If yes → return error ("Already subscribed, use billing portal to change plans")
4. Get or create Stripe Customer:
   - If `subscription.stripeCustomerId` exists → use it
   - If not → `stripe.customers.create({ email, metadata: { tenantId } })` → save the ID
5. Create Checkout Session:

   ```ts
   stripe.checkout.sessions.create({
     customer: stripeCustomerId,
     mode: 'subscription',
     line_items: [{ price: priceId, quantity: 1 }],
     success_url: `${baseUrl}/dashboard/settings/billing?success=true`,
     cancel_url: `${baseUrl}/dashboard/settings/billing`,
     metadata: { tenantId },
   })
   ```

6. Return `{ url: session.url }`

5.5 Portal API Route

`apps/dashboard/app/api/billing/portal/route.ts`

1. Authenticate
2. Get tenant's `stripeCustomerId`
3. Create Portal Session:

   ```ts
   stripe.billingPortal.sessions.create({
     customer: stripeCustomerId,
     return_url: `${baseUrl}/dashboard/settings/billing`,
   })
   ```

4. Return `{ url: session.url }`

Important: Configure the Customer Portal in Stripe Dashboard first (which plans can be swapped, whether cancellation is allowed, etc.)

5.6 Webhook Handler

`apps/dashboard/app/api/webhooks/stripe/route.ts`

Critical: Read the body as raw text, not JSON. This is the #1 Stripe webhook bug in Next.js.

```
export async function POST(request: Request) {
  const body = await request.text(); // NOT request.json()
  const signature = request.headers.get('stripe-signature')!;

  const event = stripe.webhooks.constructEvent(body, signature, webhookSecret);

  switch (event.type) {
    case 'checkout.session.completed': { /* ... */ }
    case 'invoice.paid': { /* ... */ }
    case 'invoice.payment_failed': { /* ... */ }
    case 'customer.subscription.updated': { /* ... */ }
    case 'customer.subscription.deleted': { /* ... */ }
  }

  return new Response('OK', { status: 200 });
}
```

Event handlers:

Event	What to do
`checkout.session.completed`	Extract `tenantId` from metadata. Fetch full subscription from Stripe. Upsert Subscription record with plan (mapped from priceId), status, period dates.
`invoice.paid`	Update Subscription period dates. Confirm status is ACTIVE.
`invoice.payment_failed`	Set Subscription status to PAST_DUE.
`customer.subscription.updated`	Sync everything: plan, status, priceId, cancelAtPeriodEnd, period dates. Use upsert.
`customer.subscription.deleted`	Set Subscription status to CANCELED. Set plan to FREE.

All handlers use upsert (not create) for idempotency. Stripe may deliver the same event more than once.

Local development:

```
stripe listen --forward-to localhost:3000/api/webhooks/stripe
```

Use the webhook signing secret from the CLI output as `STRIPE_WEBHOOK_SECRET`.

Files to create

File	Action
`apps/dashboard/lib/stripe.ts`	Create
`apps/dashboard/lib/billing/plans.ts`	Create
`apps/dashboard/app/api/billing/checkout/route.ts`	Create
`apps/dashboard/app/api/billing/portal/route.ts`	Create
`apps/dashboard/app/api/webhooks/stripe/route.ts`	Create

Phase 6: Billing Dashboard Page

Route: `apps/dashboard/app/dashboard/settings/billing/page.tsx`

What it shows:

For **Free** users: - Current plan: "Free" with a badge - Usage summary (X/5 flags, X/1 projects) - Pricing cards for Pro and Team with "Upgrade" buttons - "Upgrade" clicks → POST to `/api/billing/checkout` → redirect to Stripe Checkout

For **Paid** users (Pro/Team): - Current plan name + ACTIVE badge - Billing period: "Jan 15 — Feb 15, 2026" - Usage summary against plan limits - "Manage Subscription" button → POST to `/api/billing/portal` → redirect to Stripe Portal - If `cancelAtPeriodEnd` is true: show "Your plan will be downgraded to Free on [date]"

Sidebar

Add a "Billing" link in `sidebar.tsx`, either as a top-level nav item or nested under Settings.

Files to create/modify

File	Action
`apps/dashboard/app/dashboard/settings/billing/page.tsx`	Create
`apps/dashboard/app/dashboard/_components/sidebar.tsx`	Modify

Phase 7: Landing Page Alignment

Update Pricing Component

In `apps/web/src/components/landing/pricing.tsx`:

- Make sure tier names/prices match: Free ($0), Pro ($29/mo), Team ($69/mo)
- Update features lists to match `PLAN_LIMITS`
- Change CTA `href` values:
  - "Get Started" (Free) → `http://localhost:3000/api/auth/login` (dashboard login)
  - "Start Trial" (Pro) → same URL (user signs up, then upgrades from billing page)
  - "Contact Sales" (Team) → `/contact` or same login URL

Implementation Order

```
1. Schema migration   → 2. Auth0 setup   → 3. Onboarding wizard


          6. Billing page     5. Stripe     4. Wire auth into dashboard


          7. Landing page alignment
```

Start with the schema because everything depends on the updated models. Then auth (you can't do anything else without knowing who the user is). Then onboarding (first-time user path). Then wire auth into existing pages. Then Stripe. Then billing UI. Then landing page cleanup.

Verification Checklist

- ☐ Visit dashboard → redirected to Auth0 login
- ☐ Sign up with new Auth0 account → lands on onboarding wizard
- ☐ Complete onboarding → Tenant + Project + 3 Environments created, land on `/dashboard`
- ☐ Sidebar shows real user name + avatar
- ☐ Dashboard page + flags page only show current tenant's data
- ☐ Billing page shows Free plan + upgrade options
- ☐ Click "Upgrade to Pro" → Stripe Checkout opens
- ☐ Pay with test card `4242 4242 4242 4242` → webhook fires → plan updates to PRO
- ☐ Billing page now shows Pro plan + "Manage Subscription" button
- ☐ "Manage Subscription" → Stripe Portal opens
- ☐ Cancel in portal → webhook fires → plan set to cancel at period end
- ☐ Log out → redirected to Auth0 login
- ☐ Log back in → goes straight to dashboard (no onboarding again)

Key Libraries

Package	Install where	Purpose
`@auth0/nextjs-auth0`	`apps/dashboard`	Authentication, session, middleware
`stripe`	`apps/dashboard`	Stripe API (server-side only)

Key Environment Variables

```
# Auth0
AUTH0_SECRET=
AUTH0_BASE_URL=http://localhost:3000
AUTH0_ISSUER_BASE_URL=https://YOUR_TENANT.auth0.com
AUTH0_CLIENT_ID=
AUTH0_CLIENT_SECRET=

# Stripe
STRIPE_SECRET_KEY=sk_test_...
STRIPE_WEBHOOK_SECRET=whsec_...
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_test_...

# Existing
DATABASE_URL=postgresql://crivline:crivline@localhost:5432/crivline?schema=public
REDIS_URL=redis://localhost:6379
```

Generated: 2026-02-09