Toggly Next.js

App Router · Server Components · Client Components · Edge Middleware

Install

ALL PACKAGES (RECOMMENDED)

```
npm i @ops-ai/nextjs-toggly-core \
  @ops-ai/nextjs-toggly-server \
  @ops-ai/nextjs-toggly-client \
  @ops-ai/nextjs-toggly-edge
```

SERVER ONLY

```
npm i @ops-ai/nextjs-toggly-core \
  @ops-ai/nextjs-toggly-server
```

Environment variables

```
# .env.local — server (private)
TOGGLY_APP_KEY=your-app-key

# client (public)
NEXT_PUBLIC_TOGGLY_APP_KEY=your-app-key
```

Server init

Process-wide singleton. Fetches definitions-signed and evaluates locally.

```
import { initServerToggly }
  from '@ops-ai/nextjs-toggly-server';

await initServerToggly({
  appKey: process.env.TOGGLY_APP_KEY!,
  environment: 'Production',
  featureDefaults: { 'my-flag': false },
  onError: (msg, err) => console.warn(msg, err),
});
```

Server components

```
import { Feature, FeatureVariant } from
  '@ops-ai/nextjs-toggly-server';
<Feature featureKey="new-dashboard">
  <NewDashboard />
</Feature>
<Feature featureKey="new-dashboard" negate>
  <OldDashboard />
</Feature>

<Feature featureKey="OrderBadge"
  context={order} contextKind="Order">
  <Badge />
</Feature>
```

Server actions & checks

```
import { checkFeature,
  checkFeatureGate,
  isServerFeatureOn } from
  '@ops-ai/nextjs-toggly-server';

'use server'
export async function submitForm() {
  const on = await checkFeature(
    'new-checkout', userId);
  if (!on) {
    throw new Error('Feature disabled');
  }
}
await isServerFeatureOn('OrderBadge', {
  context: order,
  contextKind: 'Order',
});
const gate = await checkFeatureGate({
  featureKeys: ['beta', 'premium'],
  requirement: 'any',
});
```

Programmatic (server)

For ambient context, use isServerFeatureOn.

```
import { getServerToggly } from
  '@ops-ai/nextjs-toggly-server';
const toggly = getServerToggly();
const on = await toggly?.isFeatureOn(
  'ExpressCheckout', order, 'Order');
```

SSR → client hydration

```
// layout.tsx (server)
import { initServerToggly,
  getServerFeatures } from
  '@ops-ai/nextjs-toggly-server';

await initServerToggly({ appKey: … });
const features = getServerFeatures();
// providers.tsx (client)
<TogglyProvider
  config={{ appKey: PUBLIC_KEY }}
  initialFeatures={features}>
  {children}
</TogglyProvider>
```

Toggly Next.js

App Router · Server Components · Client Components · Edge Middleware

Client provider

```
'use client'
import { TogglyProvider } from
  '@ops-ai/nextjs-toggly-client';

<TogglyProvider
  config={{
    appKey: process.env
      .NEXT_PUBLIC_TOGGLY_APP_KEY!,
    environment: 'Production',
    onError: (msg, err) ⇒ …,
  }}
  autoInit>
  {children}
```

Client hooks

```
import { useFeatureFlag,
  useFeatureGate, useFeatures,
  useToggly } from
  '@ops-ai/nextjs-toggly-client';

const { isEnabled, isLoading } =
  useFeatureFlag('new-ui');

const { isEnabled } = useFeatureGate(
  ['a','b'], { requirement: 'any' });

const { features, refresh, setIdentity }
  = useToggly();
```

NAMED VARIANTS (ENABLEVARIANTS: TRUE)

```
import { useVariant } from
  '@ops-ai/nextjs-toggly-client';

const { variant, variantValue } =
  useVariant('checkout-flow');
```

Client components

```
import { Feature, FeatureGate,
  FeatureSwitch } from
  '@ops-ai/nextjs-toggly-client';

<Feature featureKey="beta">
  <BetaUI />
</Feature>
<Feature featureKey="beta" negate>
  <StableUI />
</Feature>

<FeatureGate featureKeys={['a','b']}
  requirement="all">
  <PremiumContent />
```

Edge middleware

Edge caches signed definitions and evaluates request overrides locally. Keep user targeting in per-call overrides.

```
import { createFeatureMiddleware }
  from '@ops-ai/nextjs-toggly-edge';

const gate = createFeatureMiddleware({
  appKey: process.env.TOGGLY_APP_KEY!,
  environment: 'Production',
});

export async function middleware(req) {
  return gate(req, {
    featureKey: 'beta-access',
    redirectTo: '/waitlist',
  });
}

export const config = {
  matcher: '/beta/:path*',
};
```

Pitfalls

- Use `TOGGLY_APP_KEY` on the server and `NEXT_PUBLIC_TOGGLY_APP_KEY` in Client Components — never expose management keys.
- Call `initServerToggly` in the layout before any server `Feature` or `checkFeature` call. The server package evaluates definitions locally.
- Pass per-call `identity` or `{ context, contextKind }` — do not set a per-request user id on the shared client.
- Pass `initialFeatures` from the server to avoid a hydration flash on the client.
- Always set `featureDefaults` for offline / API failure scenarios.
- Client `setIdentity` does not roll back identity on refresh errors; old flags may remain.
- Edge middleware runs in Edge Runtime — use `@ops-ai/nextjs-toggly-edge`, not the server package.

Resources

- Docs: `docs.toggly.io/sdks/nextjs`
- Dashboard: `app.toggly.io`