Toggly Gatsby

Gatsby plugin · React hooks · Build-time page gating

Install

```
npm i @ops-ai/gatsby-feature-flags-toggly
npm i nanostores @nanostores/react
```

Requires Gatsby 5+, React 18+, Node.js 18+.

Plugin config

GATSBY-CONFIG.JS

```
{
  resolve: '@ops-ai/gatsby-feature-flags-toggly',
  options: {
    appKey: process.env.TOGGLY_APP_KEY,
    environment: 'Production',
    flagDefaults: {
      'new-dashboard': false,
    },
    // build everything; filter at edge/runtime
    allFeaturesEnabledDuringBuild: true,
    featureFlagsRefreshInterval: 180000,
    isDebug: process.env.NODE_ENV === 'development',
    onError: (msg, err) => console.warn(msg, err),
  },
}
```

Build strategies

- `allFeaturesEnabledDuringBuild: true` — single build, filter at edge or runtime (recommended).
- `false` — fetch real flags at build; smaller bundle, requires rebuild to change flags.

Hooks

USEFEATUREFLAG

```
import { useFeatureFlag } from
  '@ops-ai/gatsby-feature-flags-toggly';

const { enabled, isReady } =
  useFeatureFlag('new-dashboard');

if (!isReady) return <Loading />;
return enabled ? <NewUI /> : <OldUI />;
```

USEFEATUREGATE

```
const { enabled } = useFeatureGate(
  ['premium-tier', 'api-access'],
  'all'   // or 'any'
);
```

USETOGGLY

```
const { flags, isReady, error,
  refreshFlags, setIdentity } = useToggly();
```

Components

```
import { Feature, FeatureGate } from
  '@ops-ai/gatsby-feature-flags-toggly';

<Feature flag="new-hero">
  <NewHero />
</Feature>
<Feature flag="new-hero" negate>
  <OldHero />
</Feature>

<FeatureGate flags={['a','b']}
  requirement="any">
  <ProtectedContent />
</FeatureGate>
<FeatureGate flags={['a','b']}
  requirement="any" negate>
  <SignUp />
</FeatureGate>

<Feature flag="legacy" negate>
  <LegacyNotice />
</Feature>
```

Toggly Gatsby

Gatsby plugin · React hooks · Build-time page gating

Page-level gating

MDX FRONTMATTER

```
---
title: Beta Feature Page
x-feature: beta-feature
---

# Beta Feature
<!-- entire page gated by flag -->
```

Plugin generates `public/toggly-page-features.json` at build. Put Toggly.CloudflareWorker in front of the origin for edge 404 and `data-feature` stripping.

Identity

```
const { setIdentity, clearIdentity } = useToggly();

// on login
setIdentity(user.id); // starts refresh

// on logout
clearIdentity(); // starts anonymous refresh
```

Without identity, % rollouts and targeting won't apply consistently.

Entity context

```
<Feature flag="OrderBadge"
  context={{ kind: 'Order', key: String(order.id),
    attributes: { Vip: order.vip } }}>
  <span className="badge">Featured</span>
</Feature>
```

Build-time vs runtime

- Static HTML embeds build-time flag values — client hooks can refresh after hydration.
- User-specific flags require client-side `setIdentity`, which starts a refresh.
- After first load, refresh failures preserve last-known-good flags (not defaults).

Pitfalls

- Always set `flagDefaults` for offline / error scenarios.
- Check `isReady` before rendering — show a loading state while flags fetch.
- Handle `error` from `useToggly()` with a fallback experience.
- Run `gatsby clean` if plugin config changes don't take effect.
- Enable Client Side API in Toggly dashboard for your App Key.

Resources

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