Toggly Nuxt

Nuxt 3 / 4 module · Nitro server · Vue composables · directives

Install

Nuxt 3 and Nuxt 4 (peer ^3.0.0 || ^4.0.0). APIs here require the matching current family versions; see the SDK guide.

```
npm i @ops-ai/nuxt-toggly
```

Bundles core, server, and client packages. Composables and components are auto-imported.

Module config

NUXT.CONFIG.TS

```
export default defineNuxtConfig({
  modules: ['@ops-ai/nuxt-toggly'],
  toggly: {
    appKey: process.env.TOGGLY_APP_KEY,
    environment: 'Production',
    featureDefaults: { 'my-flag': false },
    refreshInterval: 180000,
  },
})
```

Server handlers

USEEVENTTOGGLY

```
// server/api/data.ts
export default defineEventHandler(async (event) => {
  const toggly = useEventToggly(event);
  if (await toggly.isFeatureOn('new-api')) {
    return getNewApiData();
  }
  return getLegacyApiData();
});
```

QUICK CHECK

```
if (await isEventFeatureOn(event,
    'new-checkout')) {
  return processNewCheckout(event);
}
```

Server gates & middleware

MULTI-FLAG GATE

```
const hasAccess =
  await evaluateEventFeatureGate(
    event,
    ['premium-tier', 'api-access'],
    'all');

if (!hasAccess) {
  throw createError({ statusCode: 403 });
}
```

PROTECTED HANDLER

```
export default defineFeatureHandler(
  'beta-api',
  async (event) => getBetaData(event),
);
```

ROUTE MIDDLEWARE

```
export default defineFeatureMiddleware({
  featureKey: 'beta-access',
  statusCode: 404,
});
```

Server + client split

Server routes use Nitro utilities only — no Vue bundled. Client code uses composables and components — no server imports.

Toggly Nuxt

Nuxt 3 / 4 module · Nitro server · Vue composables · directives

Composables

```
// auto-imported — no import needed
const { isEnabled, isLoading } =
  useFeatureFlag('new-dashboard');

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

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

Components & directives

FEATURE COMPONENT

```
<Feature feature-key="beta-feature">
  <BetaContent />
</Feature>
<Feature feature-key="beta-feature" negate>
  <StableContent />
</Feature>
```

DIRECTIVES

```
<div v-feature="'SignUp'">…</div>
<div v-feature-show="'DarkMode'">…</div>
<div v-feature-class:highlight="'Beta'">…</div>
```

Identity

```
async function onLogin(userId) {
  await setIdentity(userId);
  // flags refresh automatically
}

async function onLogout() {
  await setIdentity('');
}
```

Variants

NUXT.CONFIG.TS

```
toggly: { enableVariants: true }
```

USEVARIANT

```
const { variant, variantValue } =
  useVariant('pricing-test');

// variant?.name === 'B' | 'C' | …
```

USETOGGLY / SERVER

```
const { getVariant } = useToggly();
const v = getVariant('pricing-test');

// server/api handler
import { getServerVariant }
  from '@ops-ai/nuxt-toggly-server';
const sv = getServerVariant('pricing-test');
```

Vue template patterns

```
<script setup>
const { isEnabled } = useFeatureFlag('new-dashboard')
</script>

<template>
  <NewDashboard v-if="isEnabled" />
  <OldDashboard v-else />
</template>
```

Pitfalls

- Composables are auto-imported — don't import from the server package in `.vue` files.
- Use `useEventToggly(event)` in Nitro handlers — not the client composables.
- Set `featureDefaults` in module config for offline resilience.
- Call `setIdentity` after login for targeting and consistent rollouts.
- Check `isReady` / `isLoading` before rendering flag-dependent UI.

Resources

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