Adding A/B Tests with Posthog

A complete guide to adding an A/B test to a software product using PostHog and React.

Adding a Feature Flag **2**
Creating an A/B test **7**

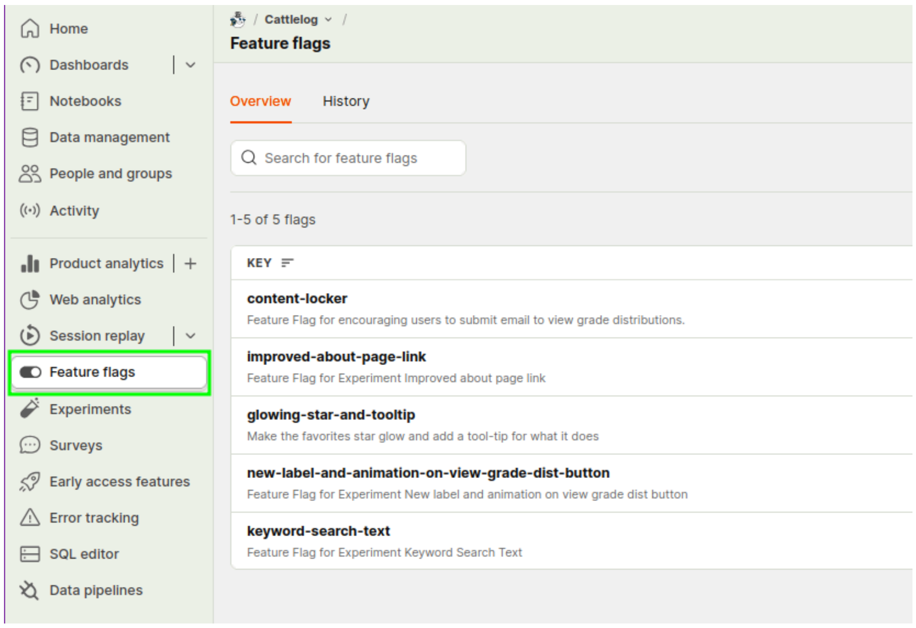
v0.1.0

To add an A/B test, you first need to create a feature flag.

Adding a Feature Flag

Adding a feature flag involves both adding code and configuring your feature flag in PostHog.

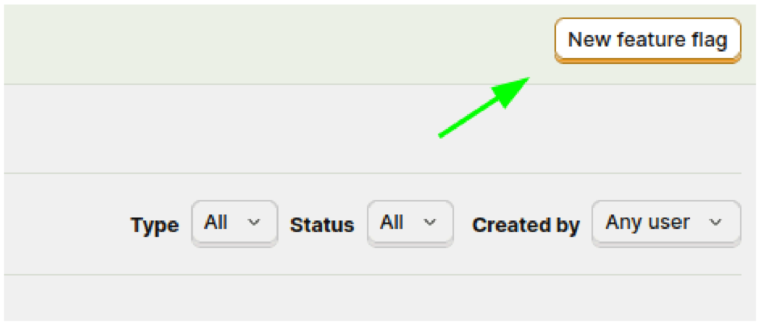
Here is the process of configuring the feature flag in PostHog:



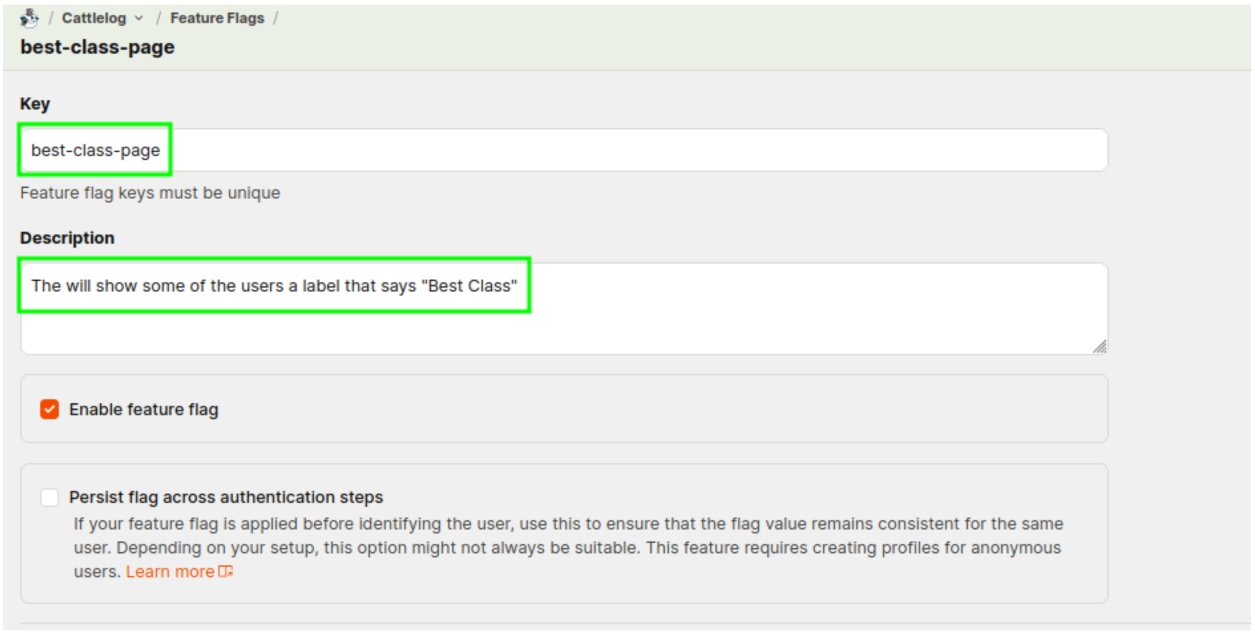
1-5 of 5 flags

KEY
content-locker Feature Flag for encouraging users to submit email to view grade distributions.
improved-about-page-link Feature Flag for Experiment Improved about page link
glowing-star-and-tooltip Make the favorites star glow and add a tool-tip for what it does
new-label-and-animation-on-view-grade-dist-button Feature Flag for Experiment New label and animation on view grade dist button
keyword-search-text Feature Flag for Experiment Keyword Search Text

Click on “Feature Flags” to view and create a new feature flag.



Next, click “New feature flag”.


Cattlelog / Feature Flags /

best-class-page

Key

best-class-page

Feature flag keys must be unique

Description

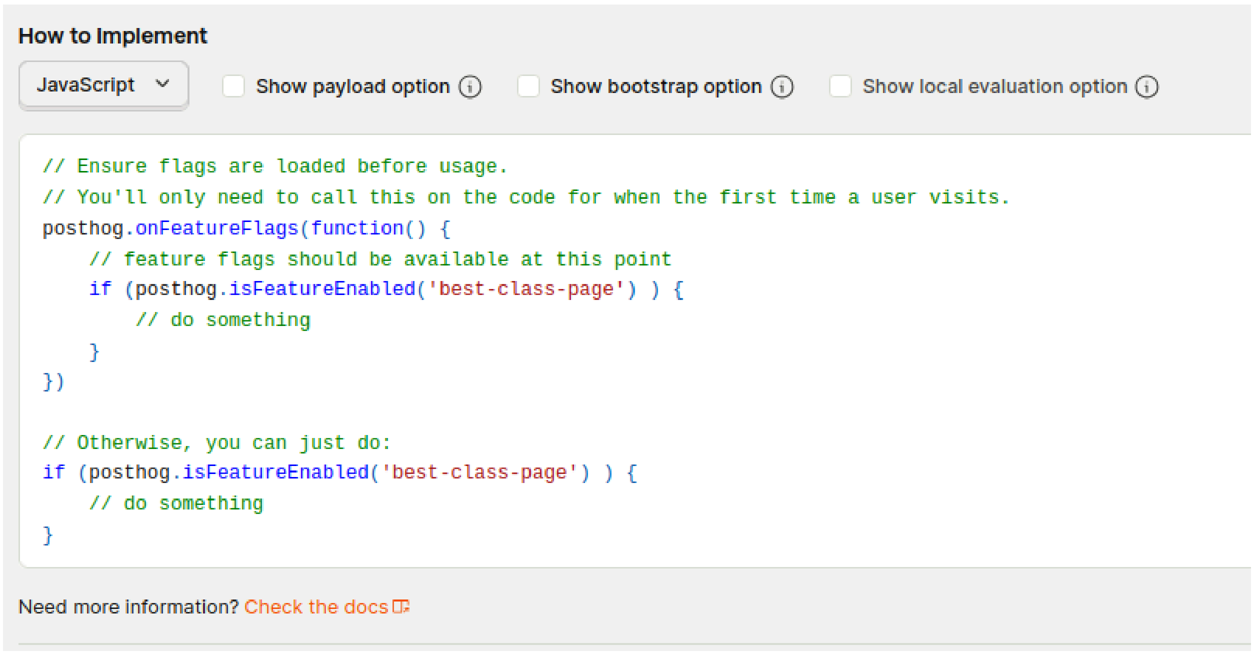
The will show some of the users a label that says "Best Class"

☑ Enable feature flag

☐ Persist flag across authentication steps

If your feature flag is applied before identifying the user, use this to ensure that the flag value remains consistent for the same user. Depending on your setup, this option might not always be suitable. This feature requires creating profiles for anonymous users. Learn more

Fill in the Key and the description. It's important to use Kebob Case for the key.

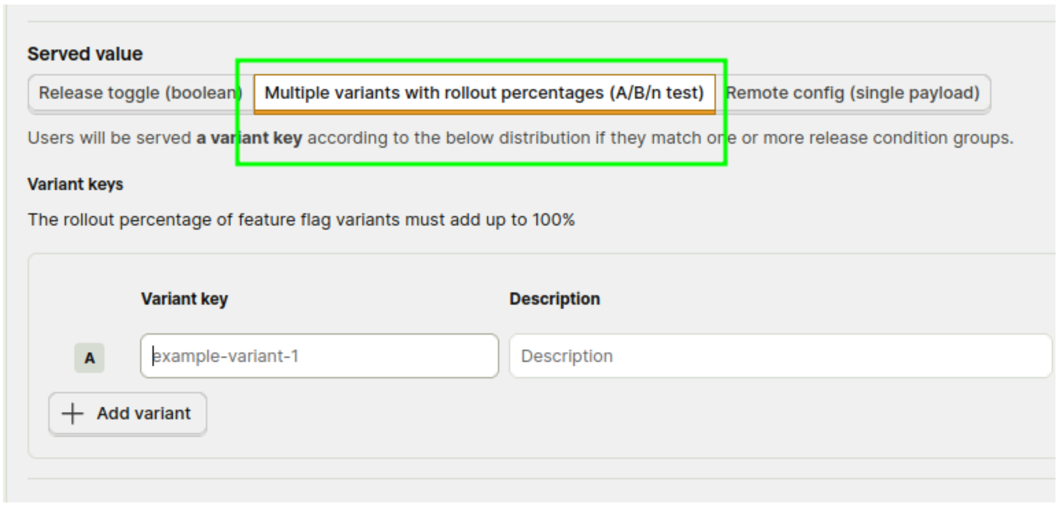

How to Implement

JavaScript ☐ Show payload option ☐ Show bootstrap option ☐ Show local evaluation option

```
// Ensure flags are loaded before usage.
// You'll only need to call this on the code for when the first time a user visits.
posthog.onFeatureFlags(function() {
    // feature flags should be available at this point
    if (posthog.isFeatureEnabled('best-class-page') ) {
        // do something
    }
})

// Otherwise, you can just do:
if (posthog.isFeatureEnabled('best-class-page') ) {
    // do something
}
```

Need more information? Check the docs

At the bottom, you can find example code for how to implement the feature flag in code. We will use this later.


Served value

Release toggle (boolean) | Multiple variants with rollout percentages (A/B/n test) | Remote config (single payload)

Users will be served **a variant key** according to the below distribution if they match one or more release condition groups.

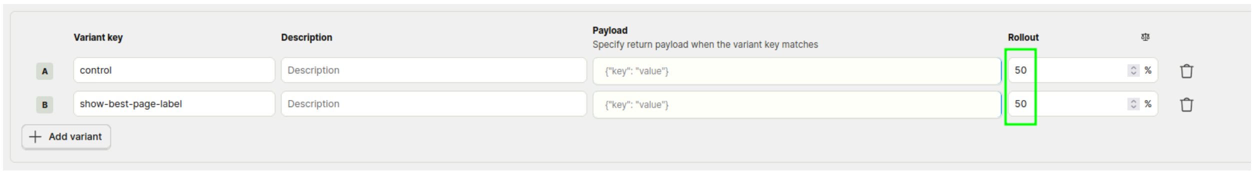
Variant keys

The rollout percentage of feature flag variants must add up to 100%

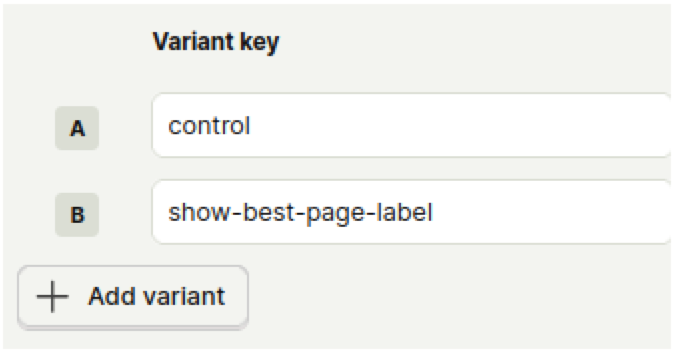
	Variant key	Description
A	example-variant-1	Description

Set the served value to the one that says “A/B/n test”.

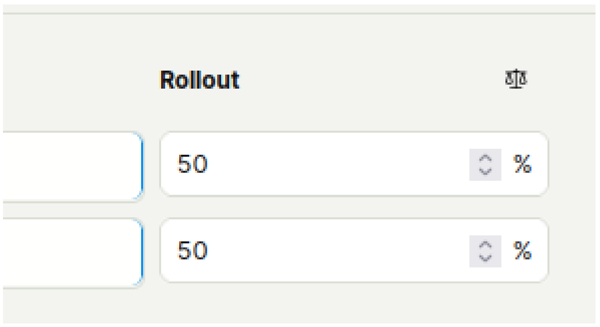
Create your variants! One needs to be called “control” but the rest are up to you. See later screenshots for a more zoomed in version of each part.


	Variant key	Description	Payload Specify return payload when the variant key matches	Rollout
A	control	Description	{"key": "value"}	50 %
B	show-best-page-label	Description	{"key": "value"}	50 %

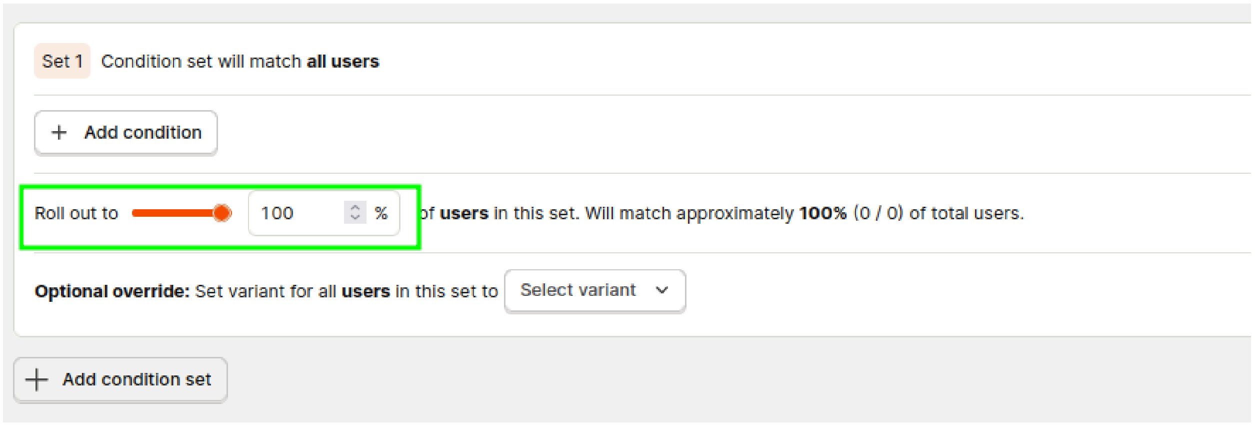
Here I created the labels “control” and “show-best-page-label”. Please use Kebob Case here too.


	Variant key
A	control
B	show-best-page-label

I then set the rollout to **50%** each.


Rollout
50 %
50 %

Then set the rollout to 100% of users.


Set 1 Condition set will match **all users**

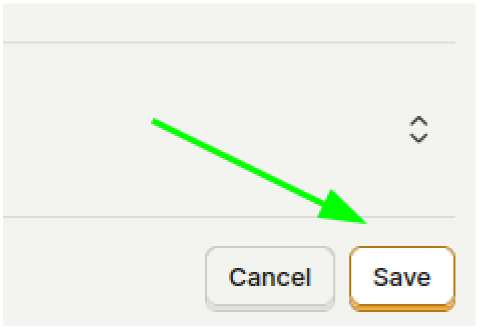
Add condition

Roll out to 100 % of **users** in this set. Will match approximately **100%** (0 / 0) of total users.

Optional override: Set variant for all **users** in this set to Select variant

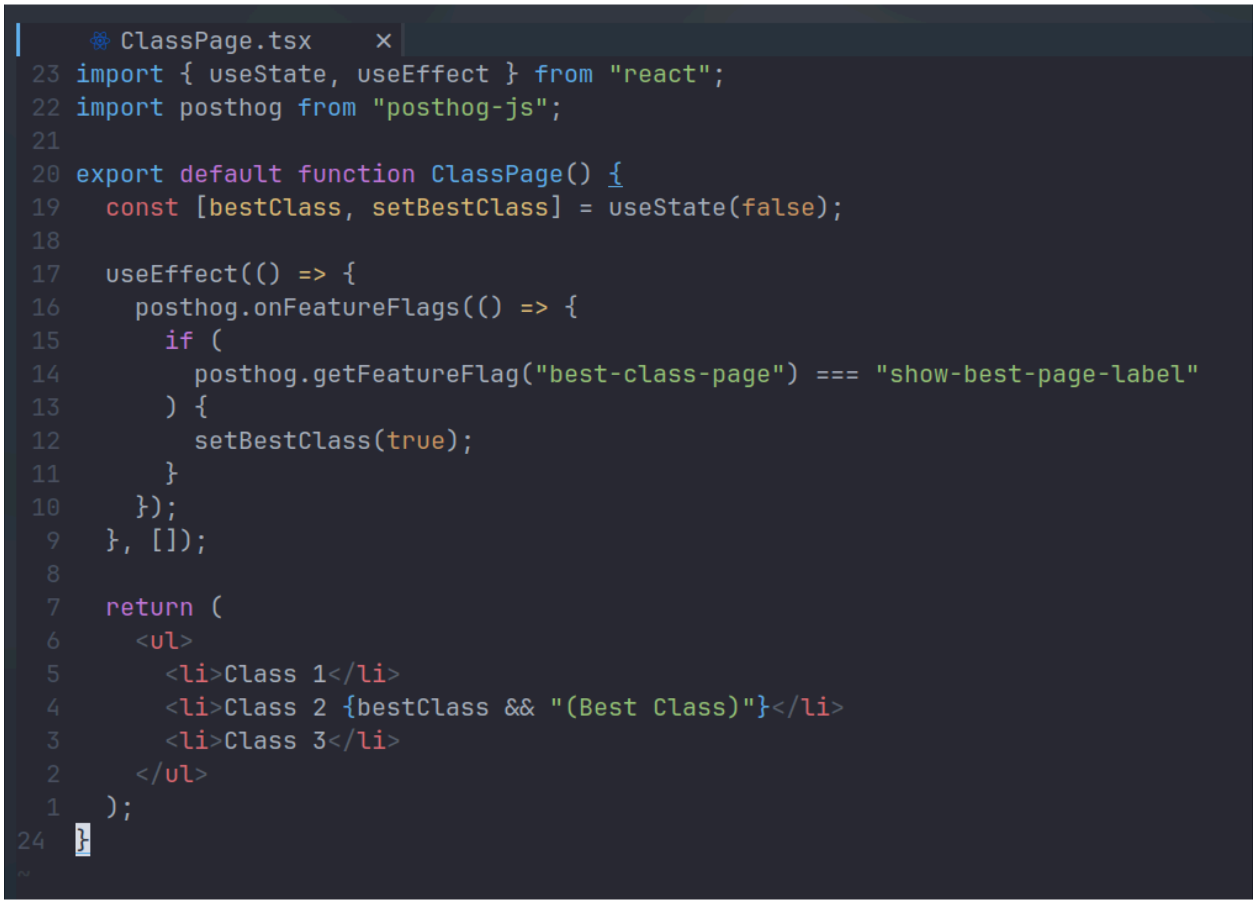
Add condition set

This essentially means that 100% will be given the option to have either A or B for their view.



Then **click save**! I've forgotten this a few times.

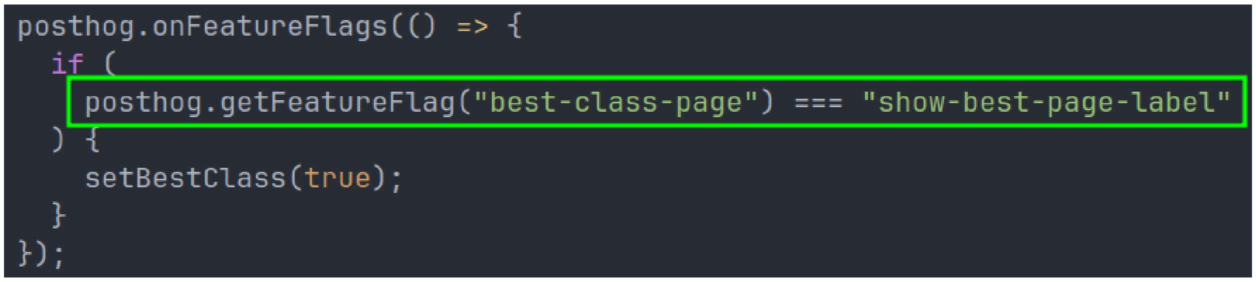
Now is time to add the code code to use the feature flag.



```
ClassPage.tsx ×
import { useState, useEffect } from "react";
import posthog from "posthog-js";

export default function ClassPage() {
  const [bestClass, setBestClass] = useState(false);

  useEffect(() => {
    posthog.onFeatureFlags(() => {
      if (
        posthog.getFeatureFlag("best-class-page") === "show-best-page-label"
      ) {
        setBestClass(true);
      }
    });
  }, []);

  return (
    <ul>
      <li>Class 1</li>
      <li>Class 2 {bestClass && "(Best Class)"}</li>
      <li>Class 3</li>
    </ul>
  );
}
```

This will call the feature flag and see if it's enabled. If it is, it will render the version that includes the information added by the feature flag.

The important part is where we call the PostHog library to check if the feature is enabled.



```
posthog.onFeatureFlags(() => {
  if (
    posthog.getFeatureFlag("best-class-page") === "show-best-page-label"
  ) {
    setBestClass(true);
  }
});
```

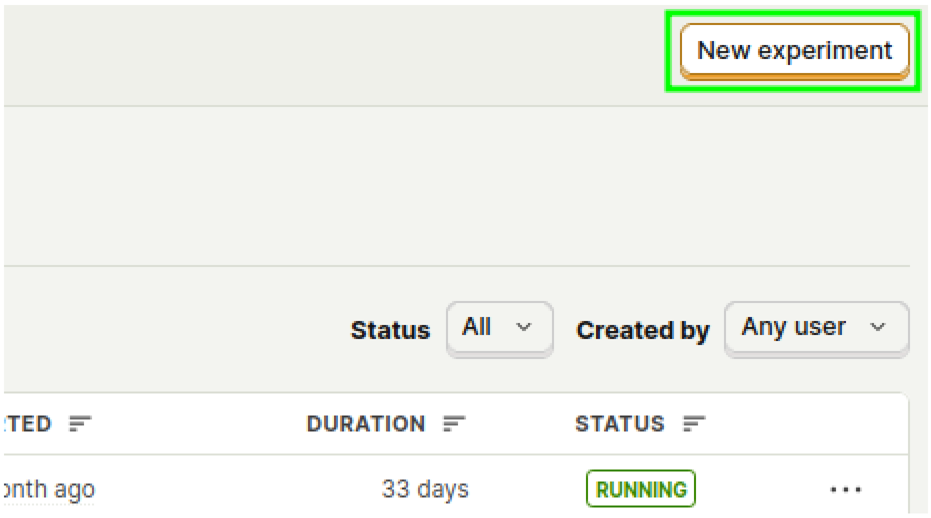
We do not need to check for "control" as that's the default and we usually use an "else" to represent the "control" variant.

This is the whole process for adding a feature flag and this process is helpful even without adding an A/B test. Say you want to be able to disable a feature remotely without changing the code; feature flags are great for this!

In addition to being useful for enabling/disabling features, we can also use these feature flags to run tests to see how users react.

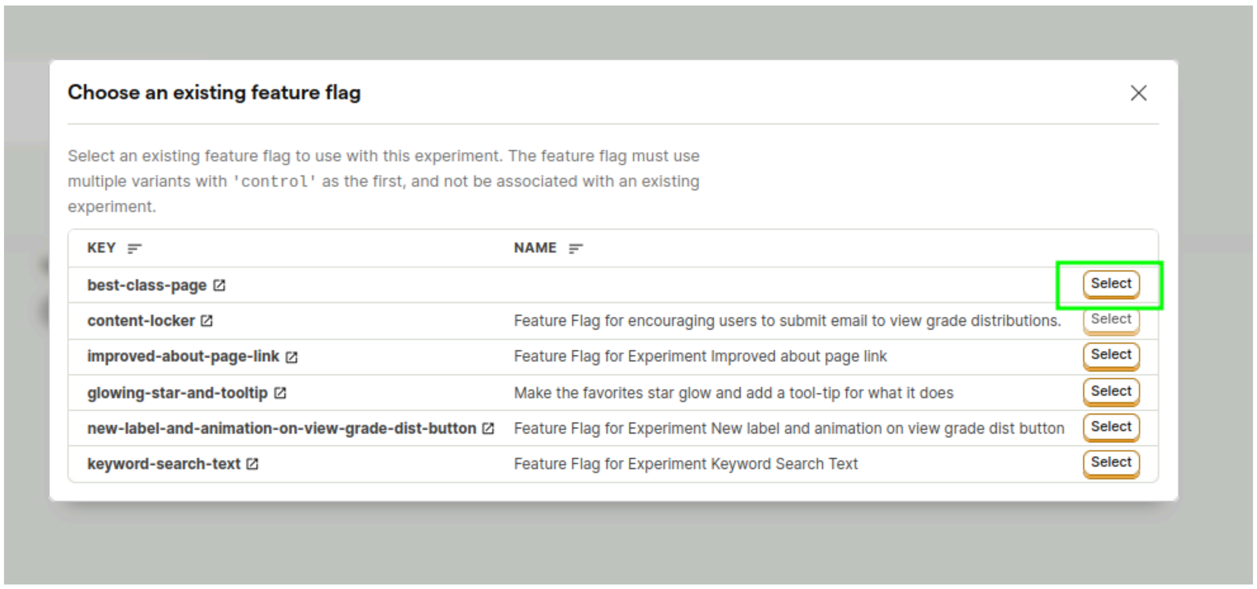
Creating an A/B test

This chapter requires a feature flag to work. See the above section if you haven't completed this.



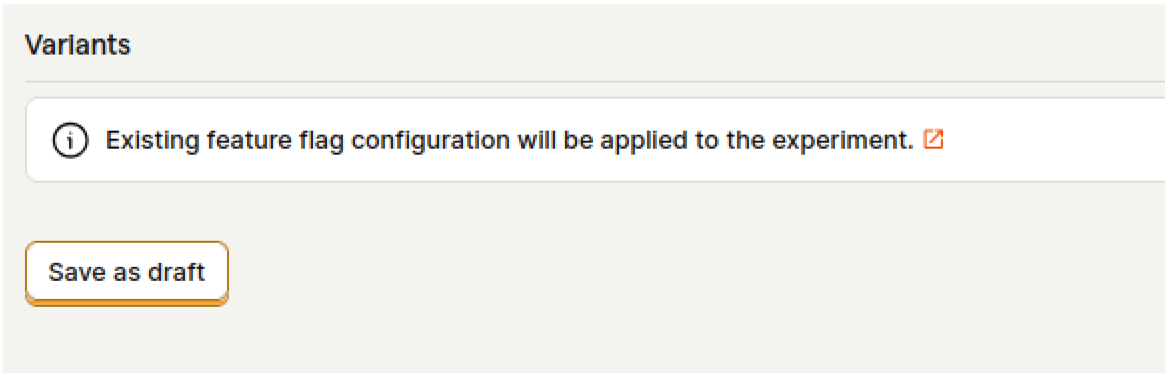
To create an A/B test, go to the Experiments section on PostHog and click "New experiment" in the top right corner.

Add a Name and then click "Link to existing feature flag". You will see the following menu. Go ahead and click "Select" for the key for the new feature flag we just created.


Choose an existing feature flag

Select an existing feature flag to use with this experiment. The feature flag must use multiple variants with `'control'` as the first, and not be associated with an existing experiment.

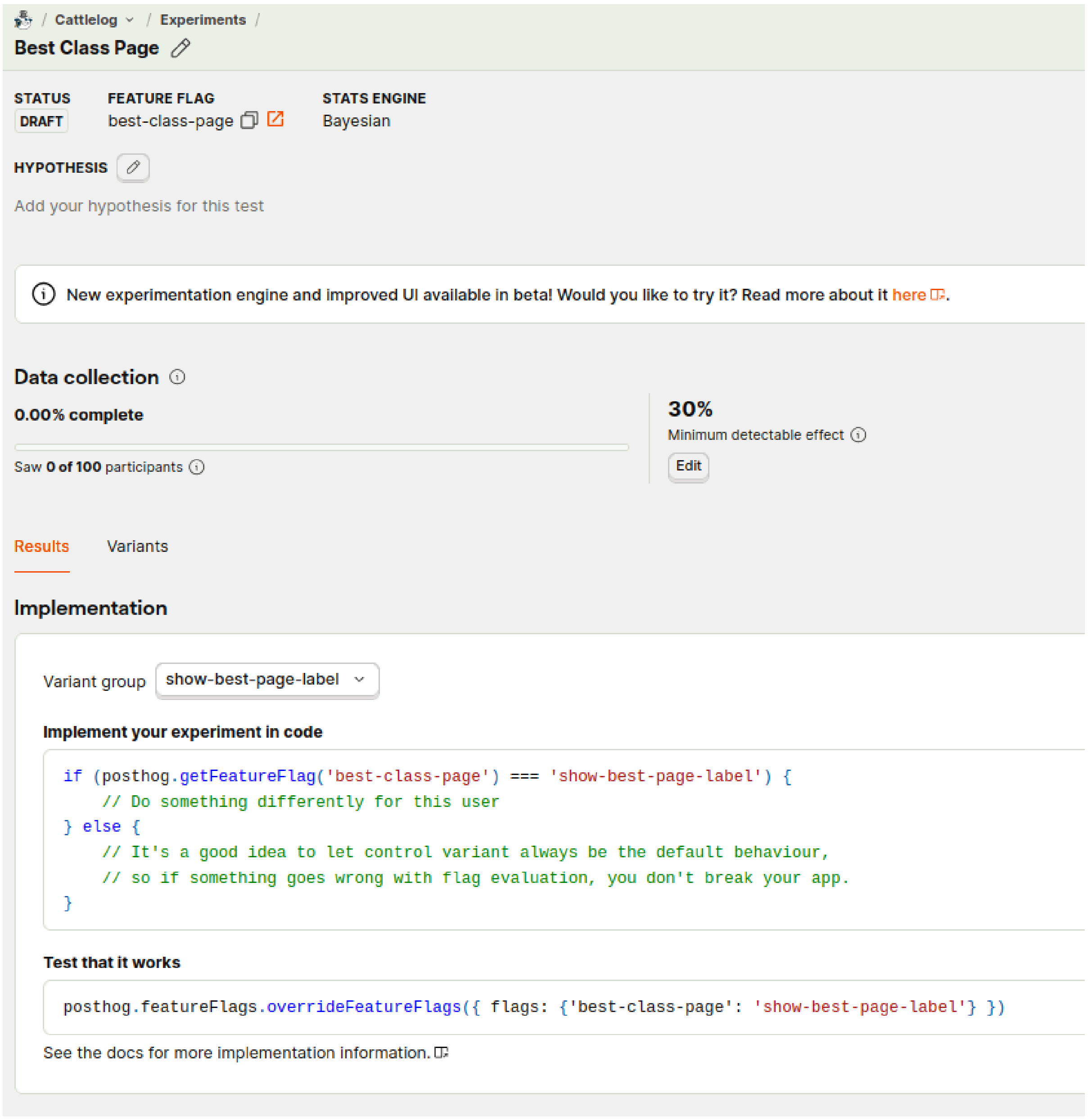
KEY	NAME	
best-class-page		Select
content-locker	Feature Flag for encouraging users to submit email to view grade distributions.	Select
improved-about-page-link	Feature Flag for Experiment Improved about page link	Select
glowing-star-and-tooltip	Make the favorites star glow and add a tool-tip for what it does	Select
new-label-and-animation-on-view-grade-dist-button	Feature Flag for Experiment New label and animation on view grade dist button	Select
keyword-search-text	Feature Flag for Experiment Keyword Search Text	Select


Variants

Existing feature flag configuration will be applied to the experiment.

It will say this, so go and click “Save as draft”

You should now see something like this:



Best Class Page

STATUS DRAFT

FEATURE FLAG best-class-page

STATS ENGINE Bayesian

HYPOTHESIS

Add your hypothesis for this test

New experimentation engine and improved UI available in beta! Would you like to try it? Read more about it here.

Data collection

0.00% complete

Saw 0 of 100 participants

30%

Minimum detectable effect

Implementation

Variant group show-best-page-label

Implement your experiment in code

```
if (posthog.getFeatureFlag('best-class-page') === 'show-best-page-label') {
    // Do something differently for this user
} else {
    // It's a good idea to let control variant always be the default behaviour,
    // so if something goes wrong with flag evaluation, you don't break your app.
}
```

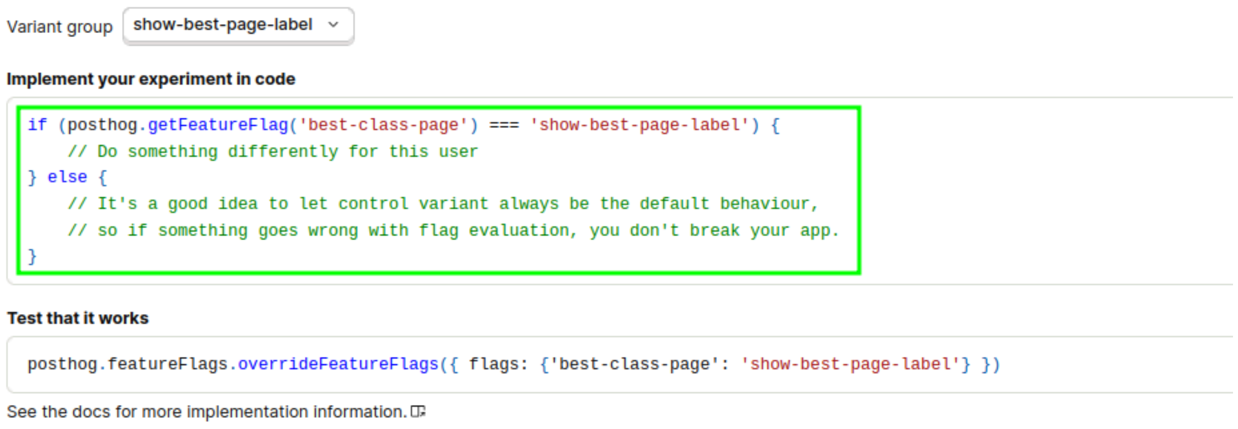
Test that it works

```
posthog.featureFlags.overrideFeatureFlags({ flags: {'best-class-page': 'show-best-page-label'} })
```

See the docs for more implementation information.

Leave the “Minimum detectable effect” to what the default is. It will get automatically adjusted to the amount of traffic your last experiments have gotten. Sometimes it’s a value of 1% and other times it’s as high as 30%.

Here we can even see the example code that is very similar to what we implemented.


Variant group show-best-page-label

Implement your experiment in code

```
if (posthog.getFeatureFlag('best-class-page') === 'show-best-page-label') {
    // Do something differently for this user
} else {
    // It's a good idea to let control variant always be the default behaviour,
    // so if something goes wrong with flag evaluation, you don't break your app.
}
```

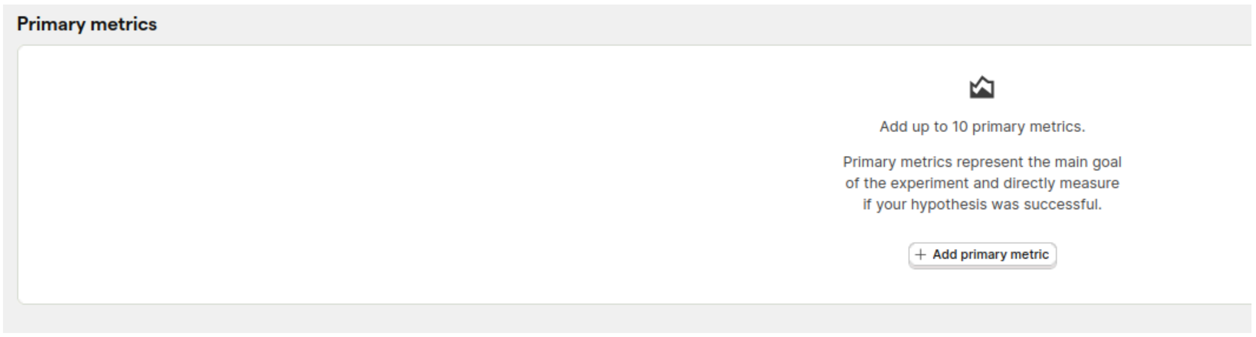
Test that it works

```
posthog.featureFlags.overrideFeatureFlags({ flags: {'best-class-page': 'show-best-page-label'} })
```

See the docs for more implementation information.

PostHog is very good at adding relevant information for developers in their application.

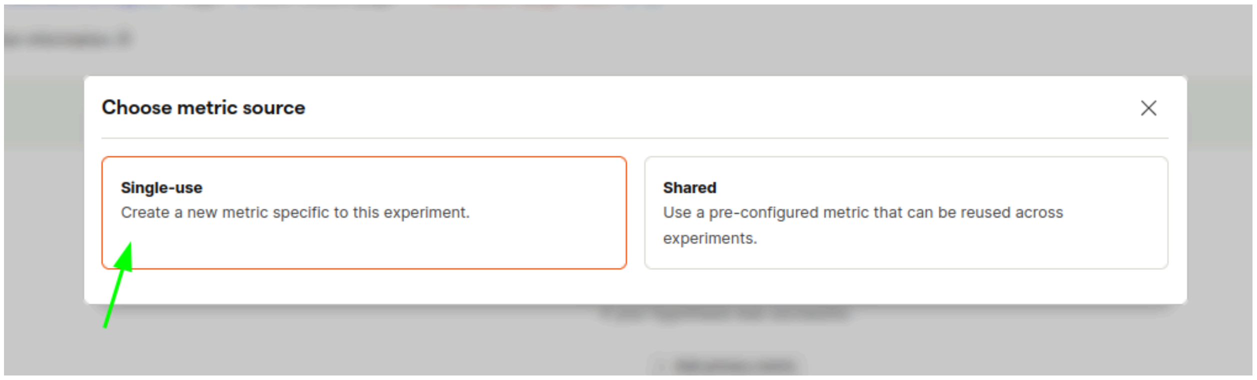
You can now create a primary metric to see how things are changed by the feature being presented.


Primary metrics

Add up to 10 primary metrics.

Primary metrics represent the main goal of the experiment and directly measure if your hypothesis was successful.

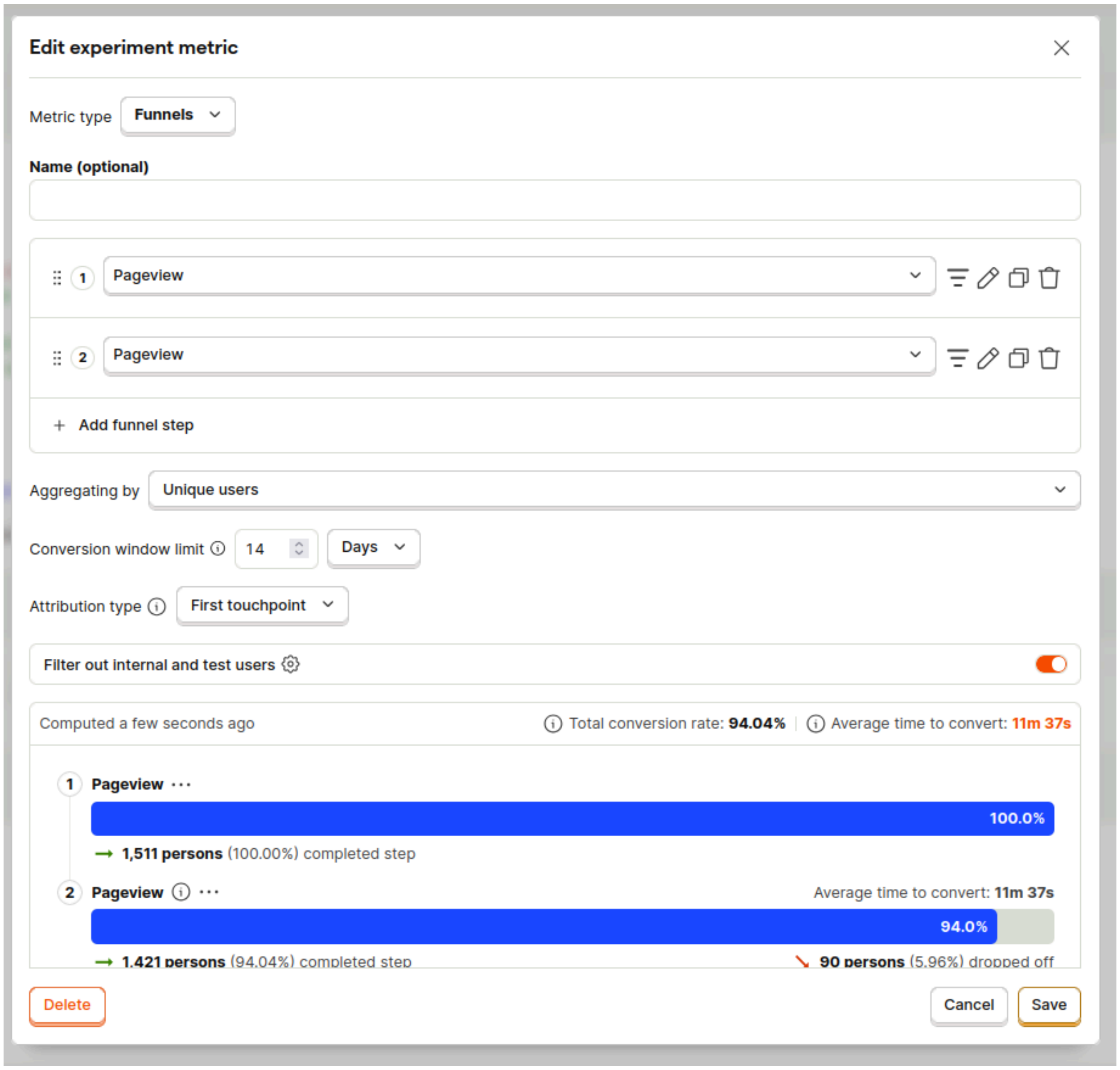
I always do “Single use” because it’s more simple and works great for our purposes but this is something to experiment with later.


Choose metric source

Single-use
Create a new metric specific to this experiment.

Shared
Use a pre-configured metric that can be reused across experiments.

You should see a page that looks like this:



Edit experiment metric

Metric type: Funnels

Name (optional)

1 Pageview

2 Pageview

+ Add funnel step

Aggregating by: Unique users

Conversion window limit: 14 Days

Attribution type: First touchpoint

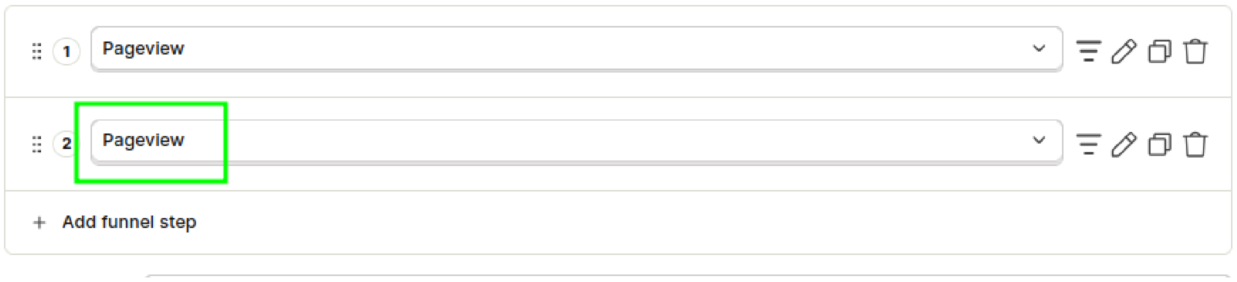
Filter out internal and test users

Computed a few seconds ago | Total conversion rate: **94.04%** | Average time to convert: **11m 37s**

1 **Pageview** ···
100.0%
→ **1,511 persons** (100.00%) completed step

2 **Pageview** ··· Average time to convert: **11m 37s**
94.0%
→ **1,421 persons** (94.04%) completed step ↘ **90 persons** (5.96%) dropped off

Go ahead and edit the second step in the funnel see here:

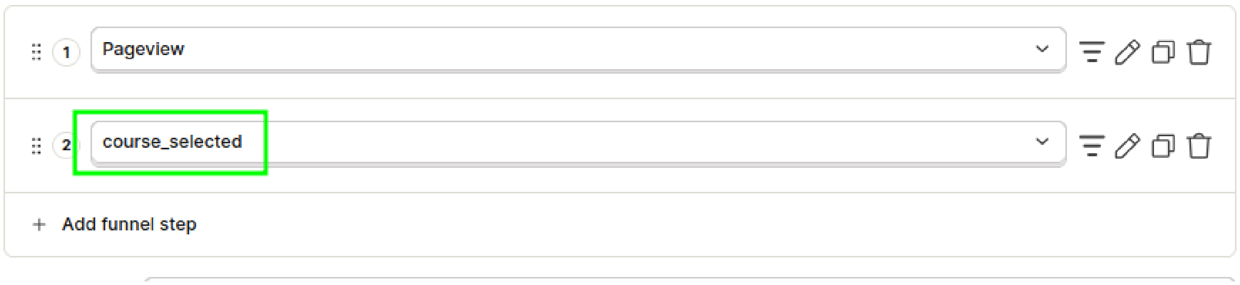


1 Pageview

2 Pageview

+ Add funnel step

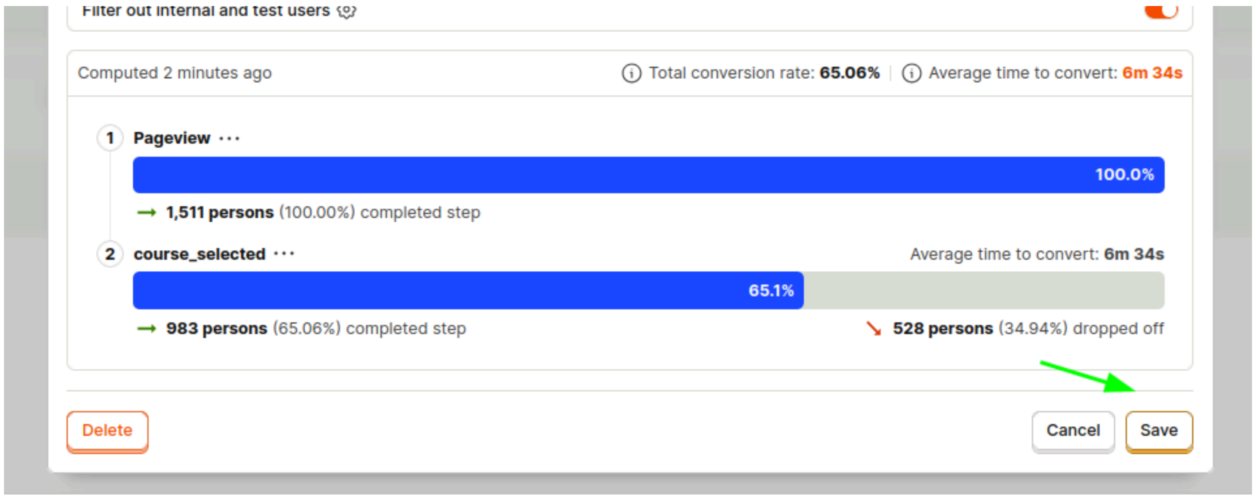
Here I set it to “course_selected”.



Pageview

course_selected

The hypothesis we are testing is that adding a “Best Class” label to one of the courses will improve the rate that *any* course is clicked. We could test if it improves the specific class, but for this example, we just care that more classes are being clicked, which would improve our north star metric as well. An improvement here is something we really care about.



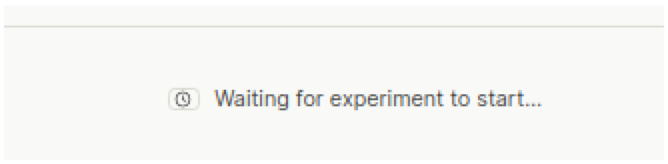
Filter out internal and test users

Computed 2 minutes ago — Total conversion rate: **65.06%** | Average time to convert: **6m 34s**

1 **Pageview** ··· 100.0%
→ **1,511 persons** (100.00%) completed step

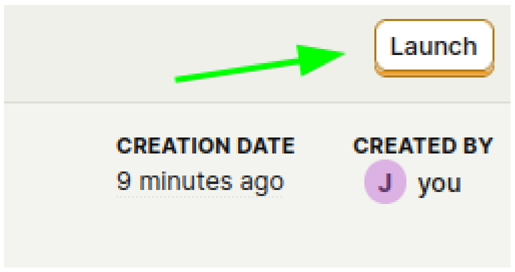
2 **course_selected** ··· Average time to convert: **6m 34s**
65.1%
→ **983 persons** (65.06%) completed step — ↘ **528 persons** (34.94%) dropped off

You can leave everything else as the default and then click “Save”.



Waiting for experiment to start...

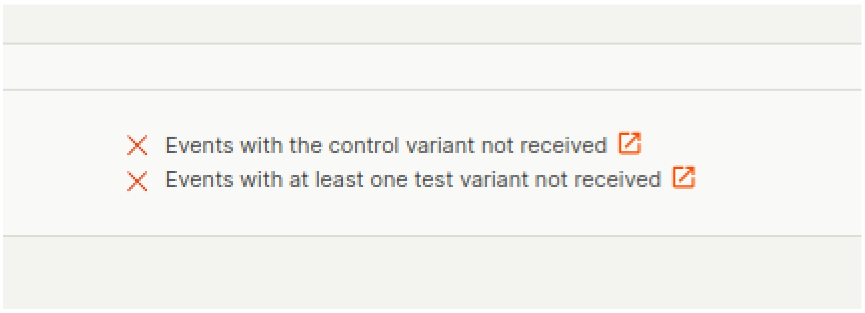
It will show that the experiment is waiting to start. That’s a good sign!



CREATION DATE: 9 minutes ago

CREATED BY: J you

Scroll back to the top and click launch!



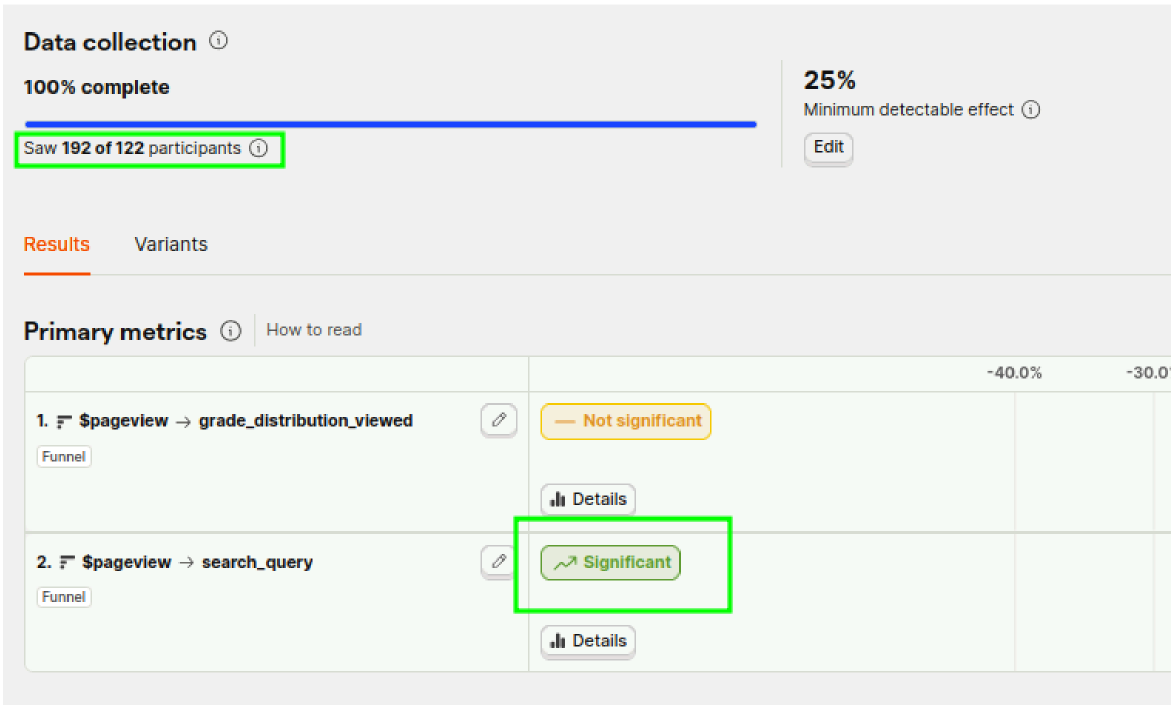
Events with the control variant not received

Events with at least one test variant not received

At first we will see these errors saying that events are not received. We need to first deploy the code (or test locally) to then see the events happen.

Click around on the deployed version and you should see some amount of events being called.

Here is an example of an A/B test we started a month ago.



Data collection

100% complete

Saw 192 of 122 participants

25%

Minimum detectable effect

Edit

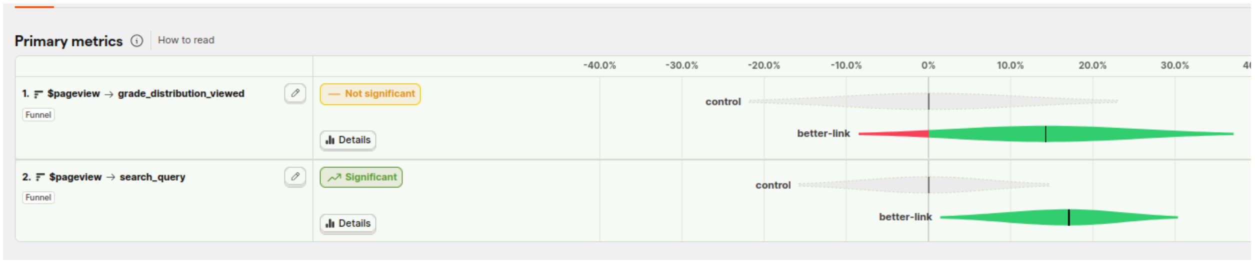
Results Variants

Primary metrics How to read

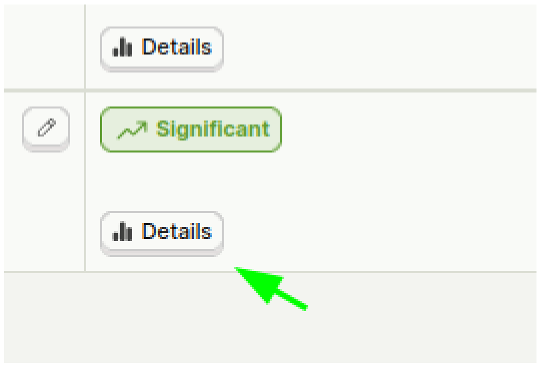
		-40.0%	-30.0
1. $pageview → grade_distribution_viewed Funnel	Not significant Details		
2. $pageview → search_query Funnel	Significant Details		

This test saw 192 users so far and one of the primary metrics is statistically significant.

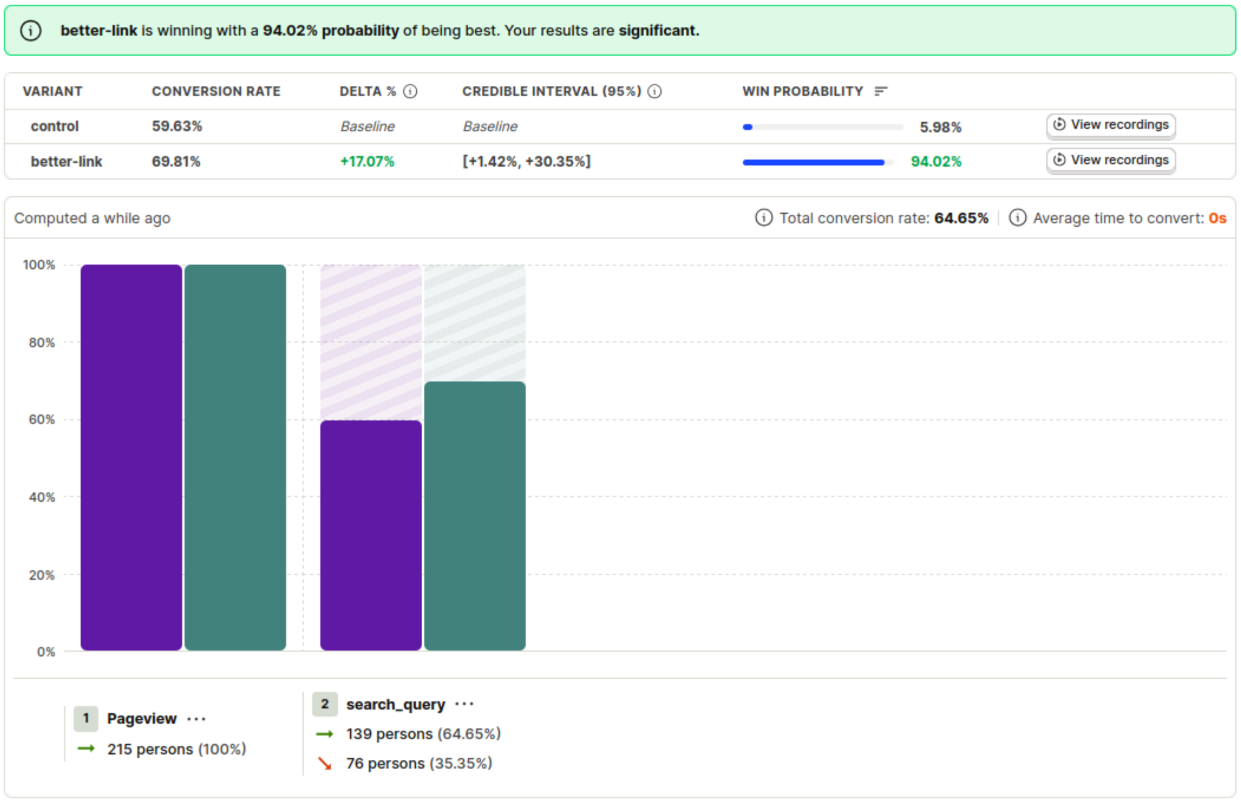
We can see the graph here on the right of the rows here:



We notice that the second primary metric of improving the quantity of searches was more significant so far. We know that the mean amount of conversion is high and we believe this with a 94.02% confidence.



We can even see a more detailed view of the results after clicking Details.



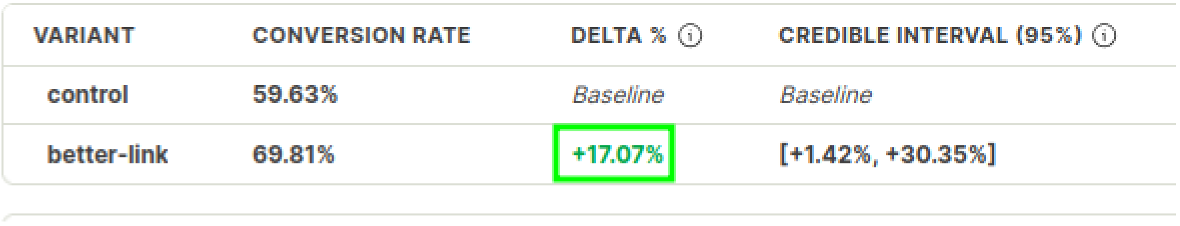
better-link is winning with a **94.02% probability** of being best. Your results are **significant.**

VARIANT	CONVERSION RATE	DELTA %	CREDIBLE INTERVAL (95%)	WIN PROBABILITY	
control	59.63%	*Baseline*	*Baseline*	5.98%	View recordings
better-link	69.81%	+17.07%	[+1.42%, +30.35%]	94.02%	View recordings

Computed a while ago

Total conversion rate: **64.65%** | Average time to convert: **0s**

The first graph (two bars on the left) show the percentage of pageviews for control versus the pageviews for our variant “better-link”. Notice both are 100% because this is the start of our funnel. The second graph (on the right) shows us how the “better-link” variant affects the quantity of the “search_query”.

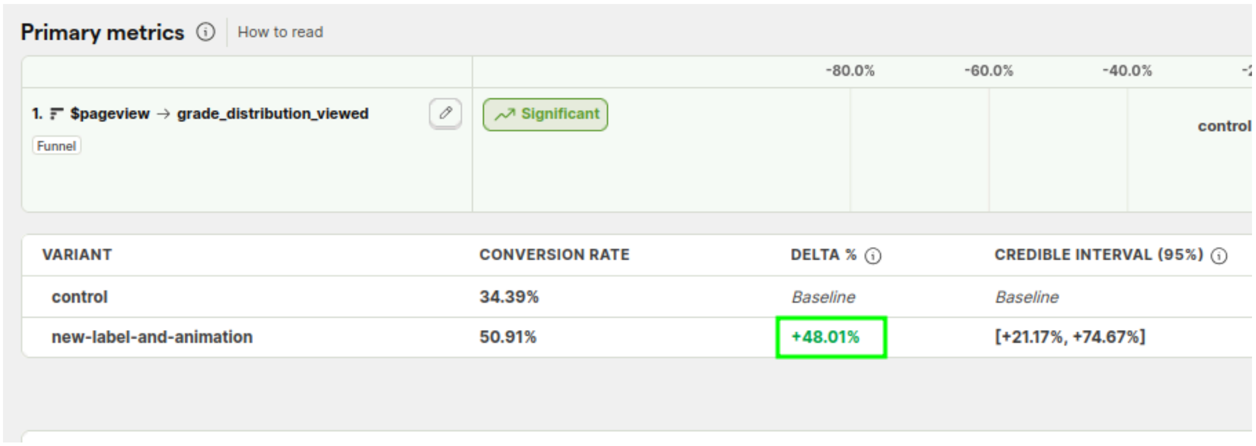


VARIANT	CONVERSION RATE	DELTA %	CREDIBLE INTERVAL (95%)
control	59.63%	*Baseline*	*Baseline*
better-link	69.81%	+17.07%	[+1.42%, +30.35%]

We can also see the percentage delta (the improvement) of the feature flag as well as a 95% confidence interval.

This test, although significant for one primary metric, hasn't been running quite long enough to make any huge judgement call. We will wait for this test to get more traffic and confirm with a higher degree of confidence the actual result.

Here is a completed test with a really high confidence level and a great effect.



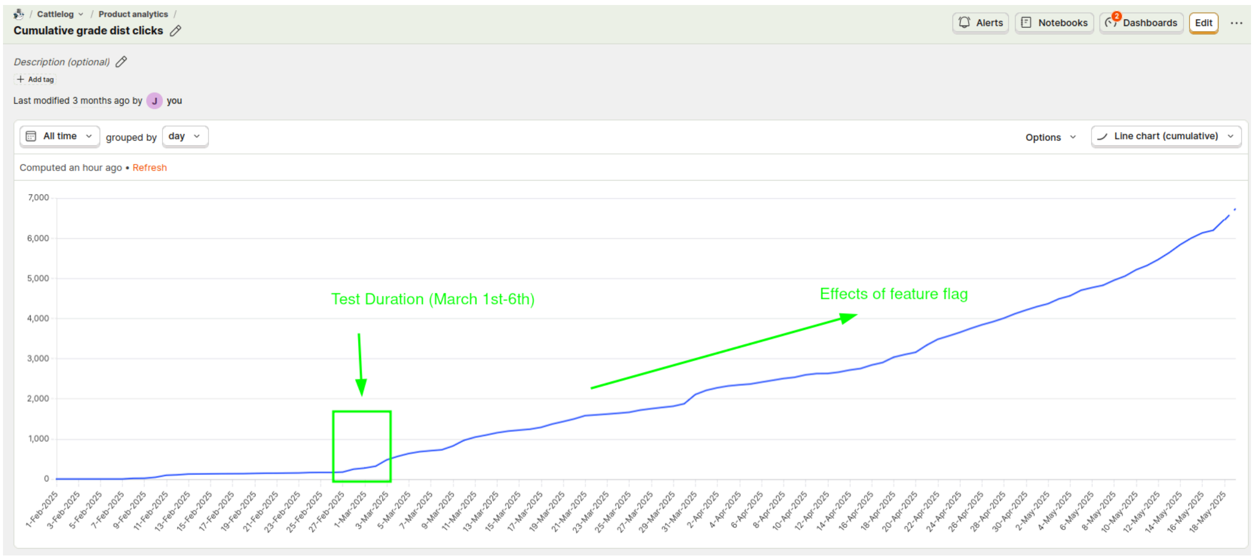
Primary metrics How to read

1. $pageview → grade_distribution_viewed

Funnel

VARIANT	CONVERSION RATE	DELTA %	CREDIBLE INTERVAL (95%)
control	34.39%	*Baseline*	*Baseline*
new-label-and-animation	50.91%	+48.01%	[+21.17%, +74.67%]

This test had a **48%** improvement from the baseline, which means that the users who saw the "new-label-and-animation" were 48% more likely to click the grade distribution label, which contributed to our 6,700+ grade distribution views.



Cattlelog / Product analytics /

Cumulative grade dist clicks

Description (optional)

Last modified 3 months ago by J you

All time grouped by day

Options Line chart (cumulative)

Computed an hour ago • Refresh

You can even see here, before the A/B test, the grade distribution page existed but the line was almost flat compared to the experiment and then the subsequent growth of the feature. The grade distribution feature was available for about a month before we did the test, and the line looks comparatively flat. This is the effect of targeted A/B tests.

Thank you for reading. If you have any feedback, please email me@jr0.org.