

Demystifying Incremental Static Regeneration

And raising the Jamstack ceiling

Phil Hawksworth, Netlify



These slides and links

findthat.at/incremental



Why this talk?

Terminology / confusion / doubt / opportunity

Jamstack

JavaScript / APIs / Markup

(but Jamstack means more than what it stands for)

Jamstack

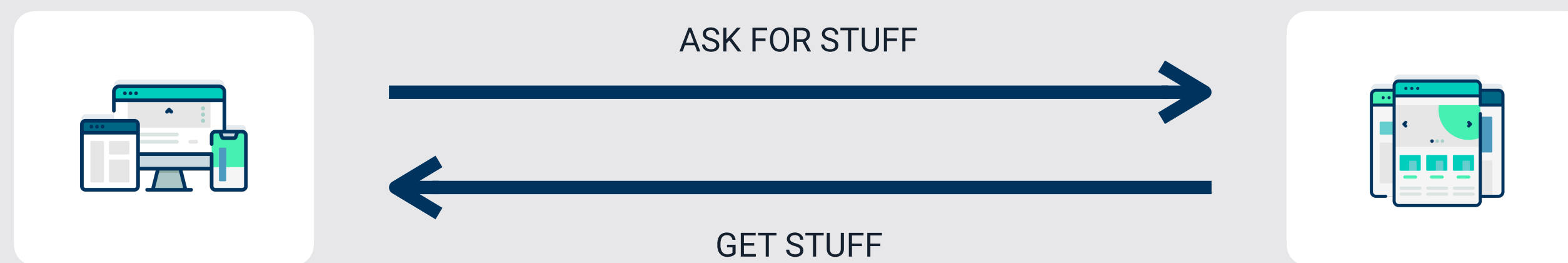
A way of thinking about how to build for the web. The UI is compiled, the frontend is **decoupled**, and data is pulled in as needed.

Decoupling

Compiling

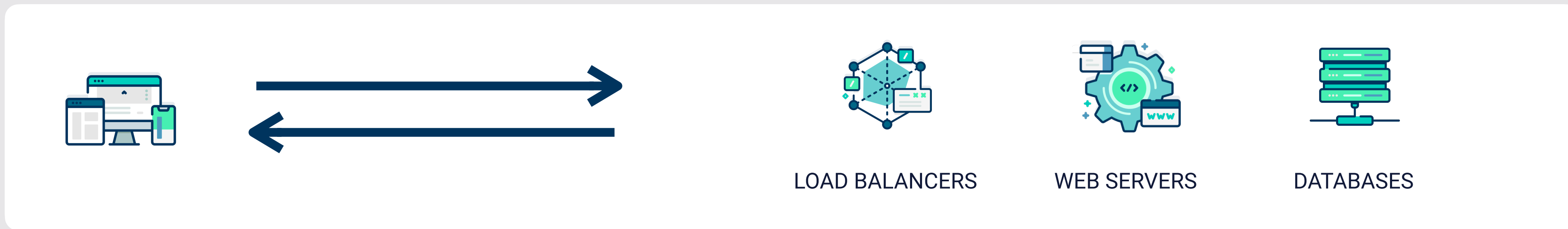
Pre-generating

It used to be so simple



Time and context

TRADITIONAL STACK



JAMSTACK



Decoupling

Front-end code is
no longer limited to
being a product of
a **back-end** system

Capable of being
served directly
from a CDN

Offloading hosting complexity as a solved problem

**Pre-generation
of a site**

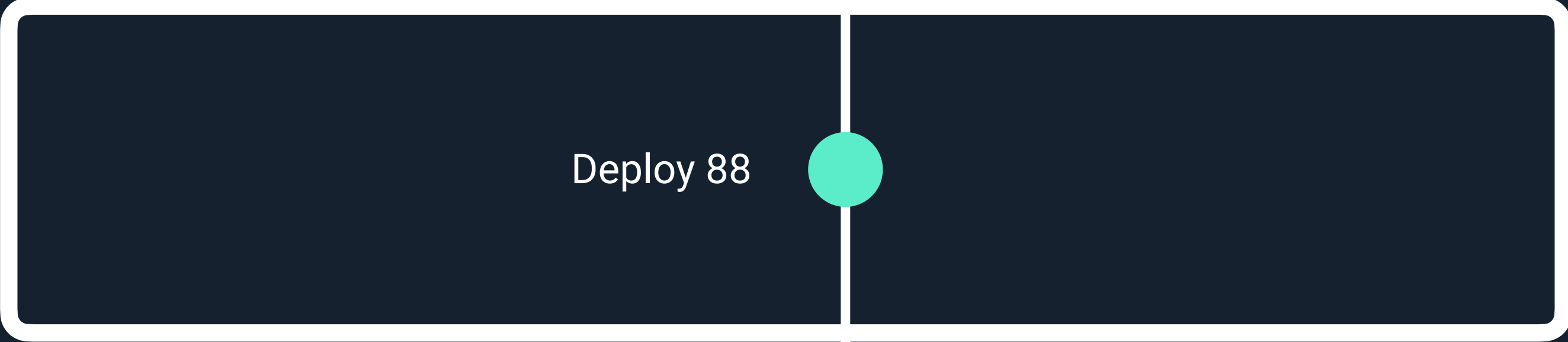
+

**The workflows and
automation that
unlocks**

HUGE BENEFIT

**Deploys are
immutable and atomic**

Live

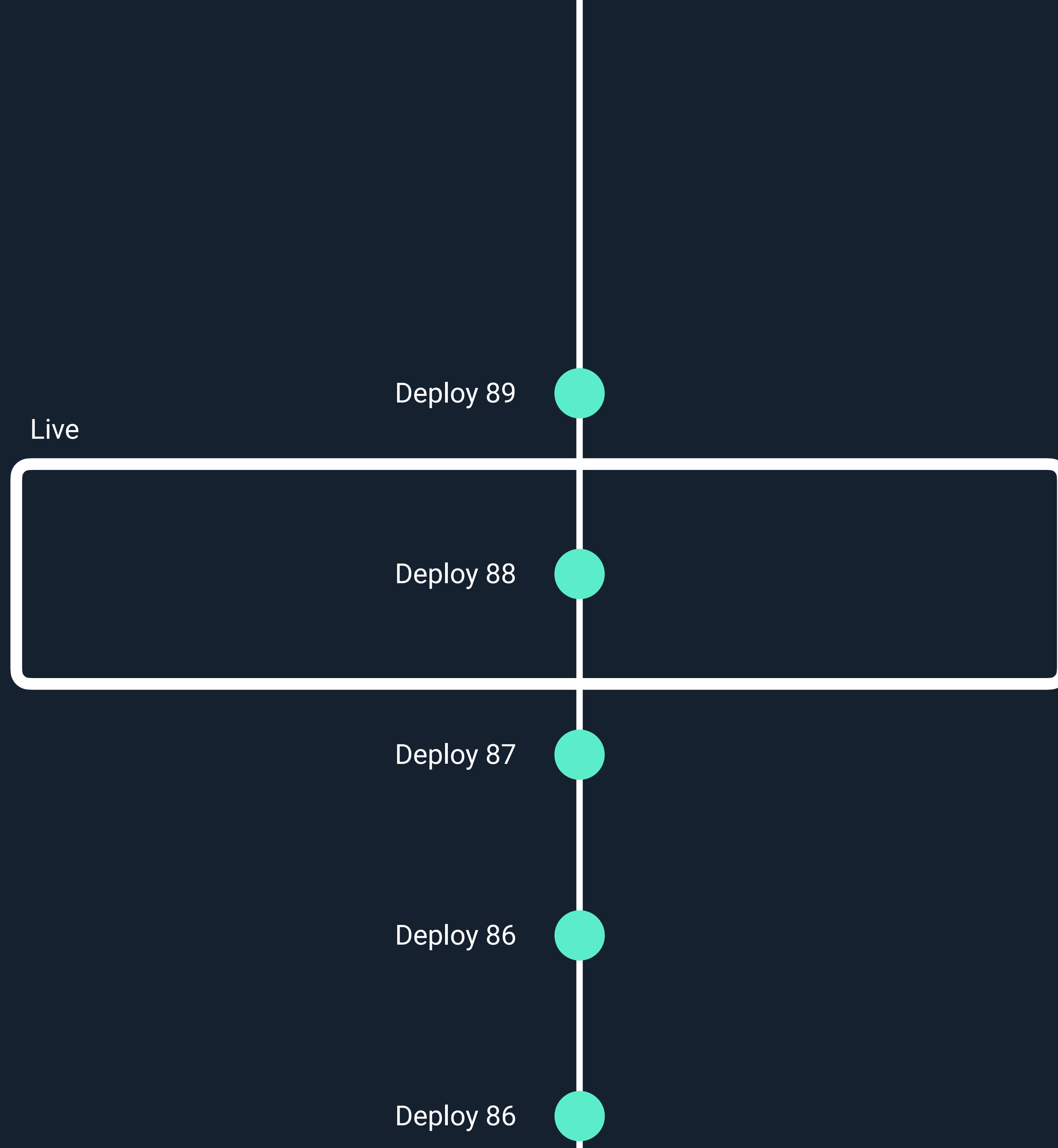


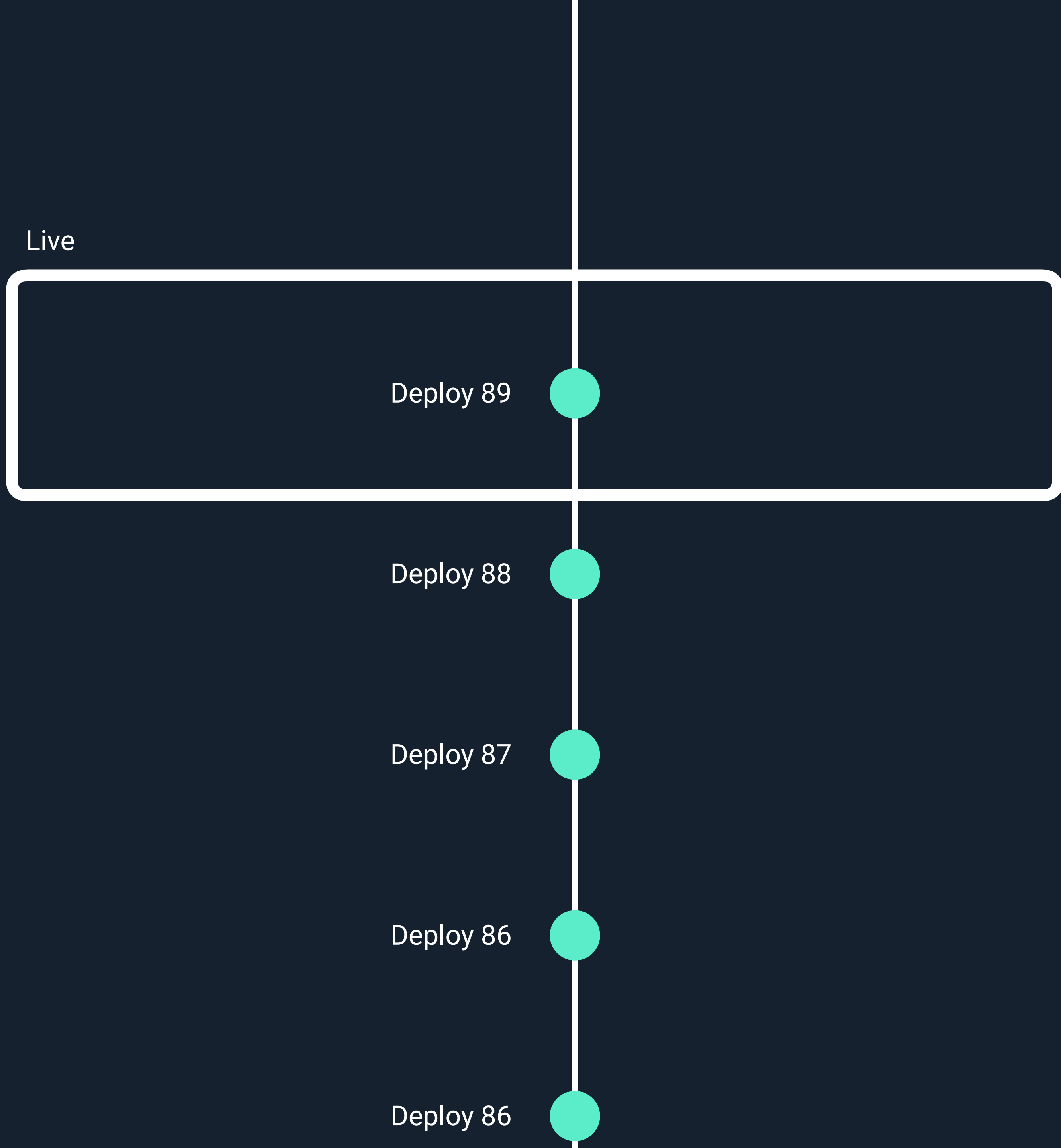
Deploy 88

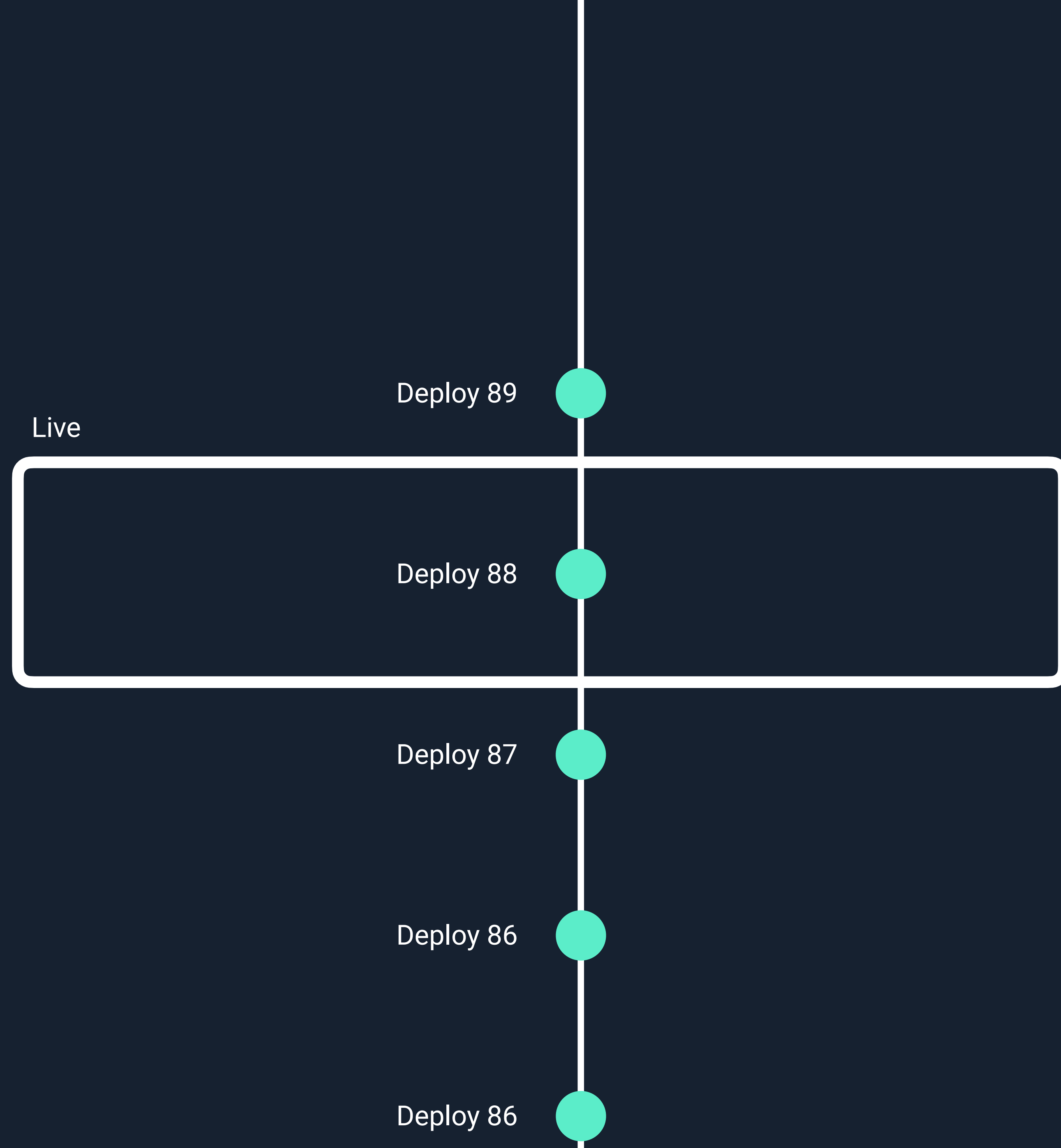
Deploy 87

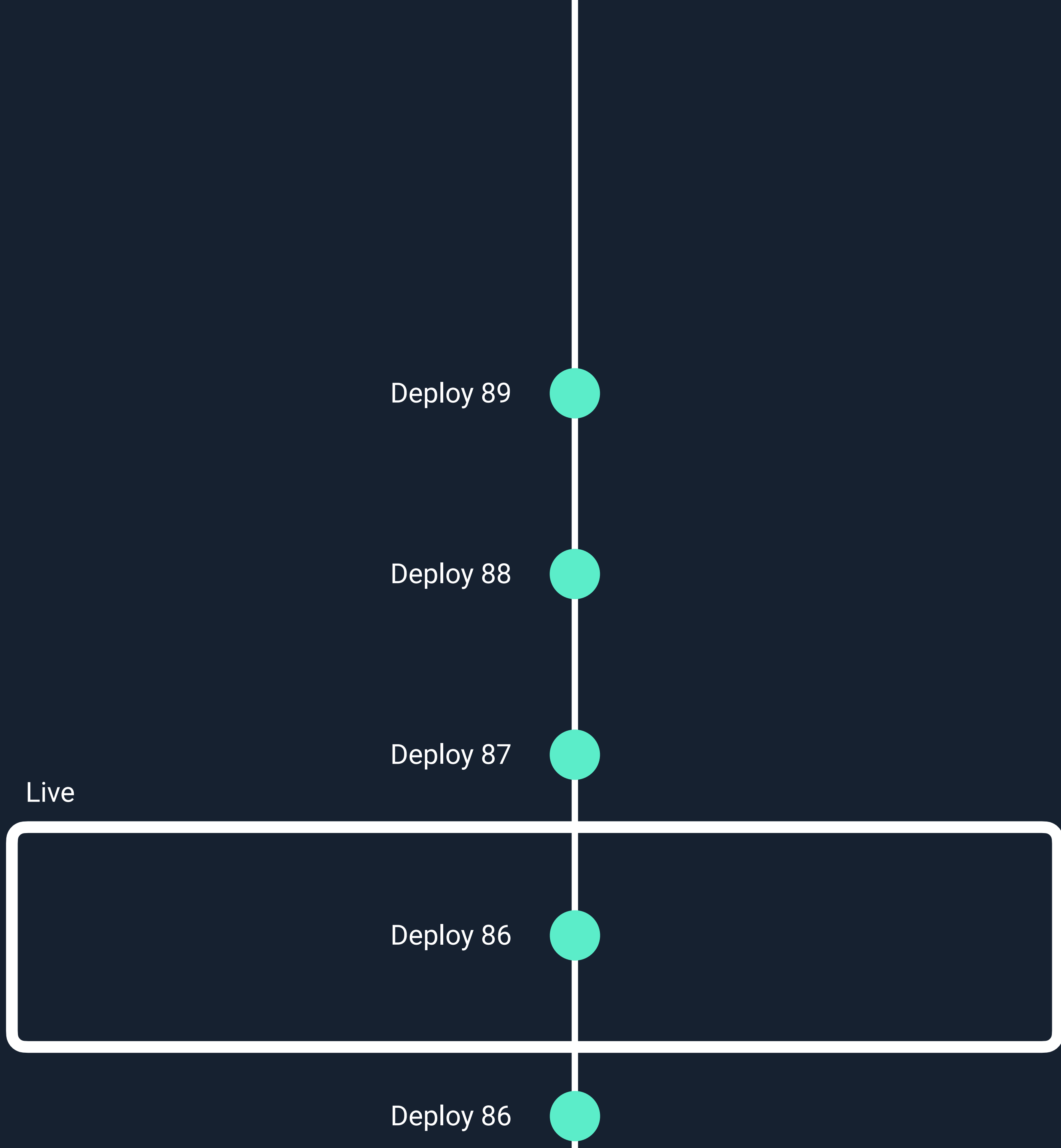
Deploy 86

Deploy 86









**Zero burden on the
developers for
caching logic**

**Caching is one of the
easiest things in
computer science**

— Nobody ever.

Confidence & certainty



But...
(And potentially a big but)

WHAT ABOUT?

User generated content?

WHAT ABOUT?

**Sites with huge
numbers of pages?**

Has Jamstack reached its ceiling?

WHAT ABOUT?

Sites with huge
numbers of pages?

WHAT ABOUT?

User generated
content?

Incremental builds

@ PhilHawksworth

Director of Developer Experience, Netlify



findthat.at/jamstack/book



findthat.at/interesting

These slides and links

findthat.at/incremental



Let's talk

1

Approaches to
delivering
incremental builds

2

Understanding the
benefits and the
sacrifices

3

A practical example of
a first step to
incremental builds

1

Approaches to delivering incremental builds

APPROACHES

ISR

Incremental
Static
Regeneration



DPR

Distributed
Persistent
Rendering



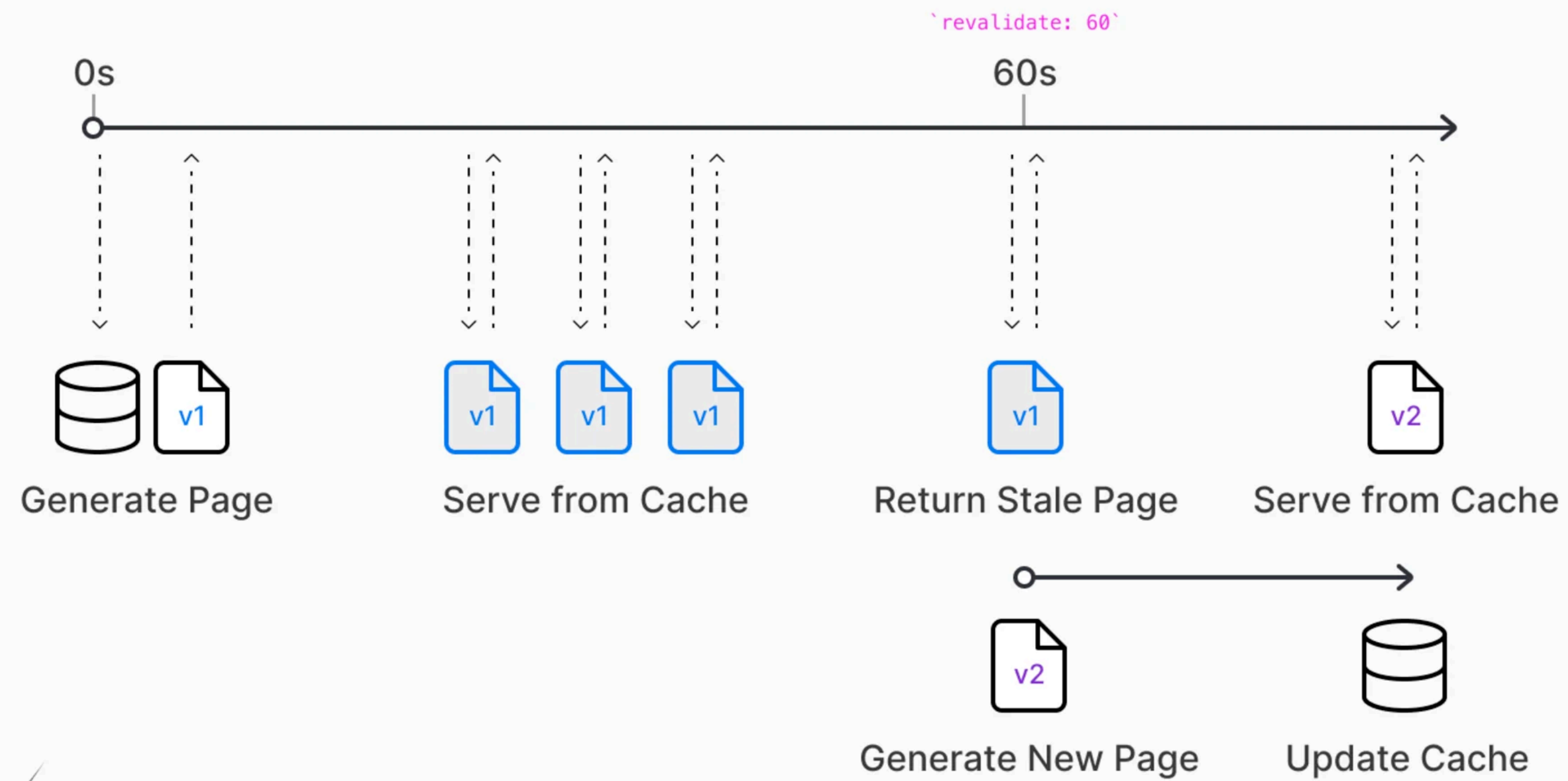
BYO

Bring
Your
Own

Incremental Static Regeneration (ISR)

Added to Next.js by Vercel

Incremental Static Regeneration



NEXT.js

When to render?

Specifying the fallback

```
fallback: false
```

If the page was not generated in the build, request will 404

```
fallback: true
```

Serve a stale page or holding page but update the cache for future requests

```
fallback: blocking
```

Generate a page on-demand and cache it for future

An excellent enhancement to Next.js

**Managing caching
behaviours needs
careful handling**

EARLIER

**Zero burden on the
developers for
caching logic**

Distributed Persistent Rendering (DPR)

(Builds with a long tail)

findthat.at/dpr

Code

Code

Build



Deploy number 75

Code

Build



On demand



Deploy number 75



Code

Build



On demand



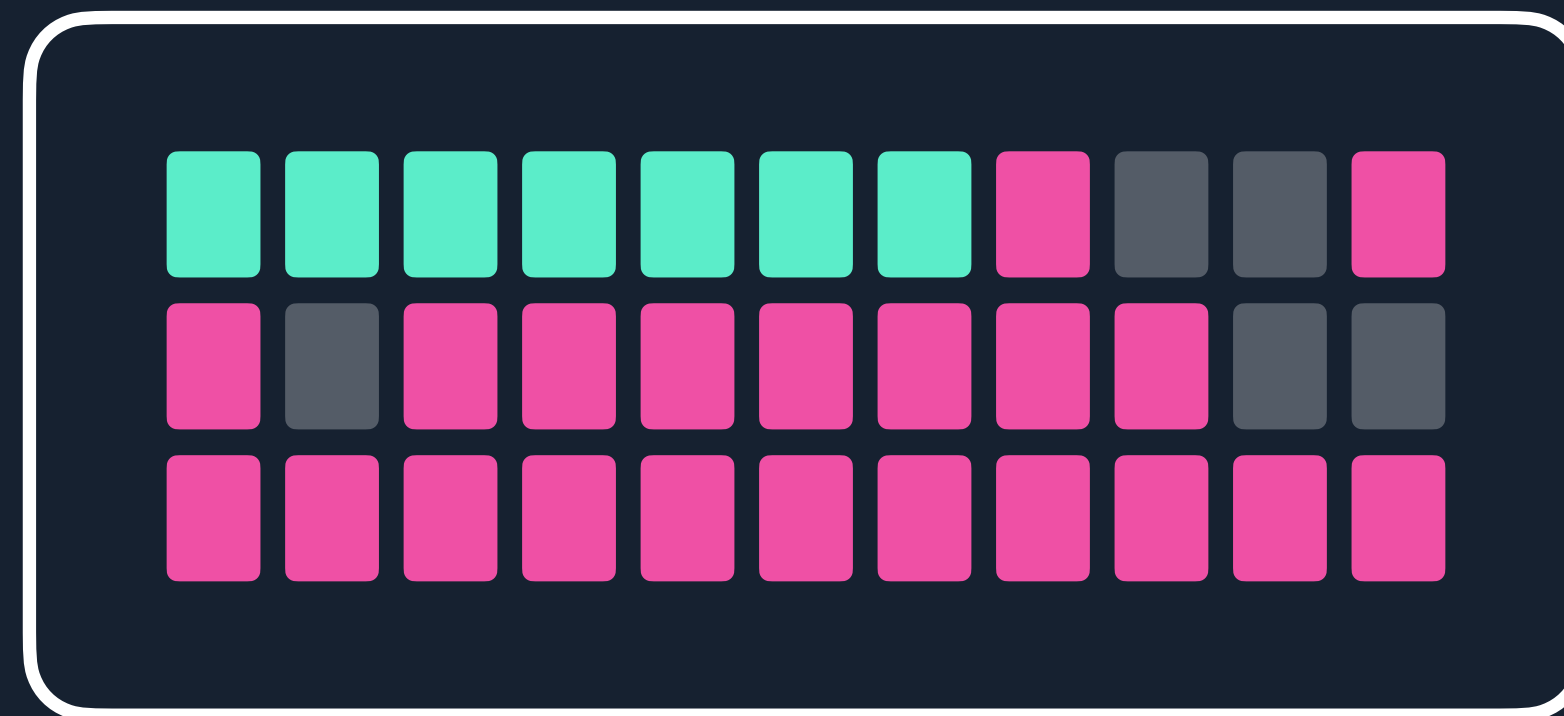
Deploy number 75



Code

Build

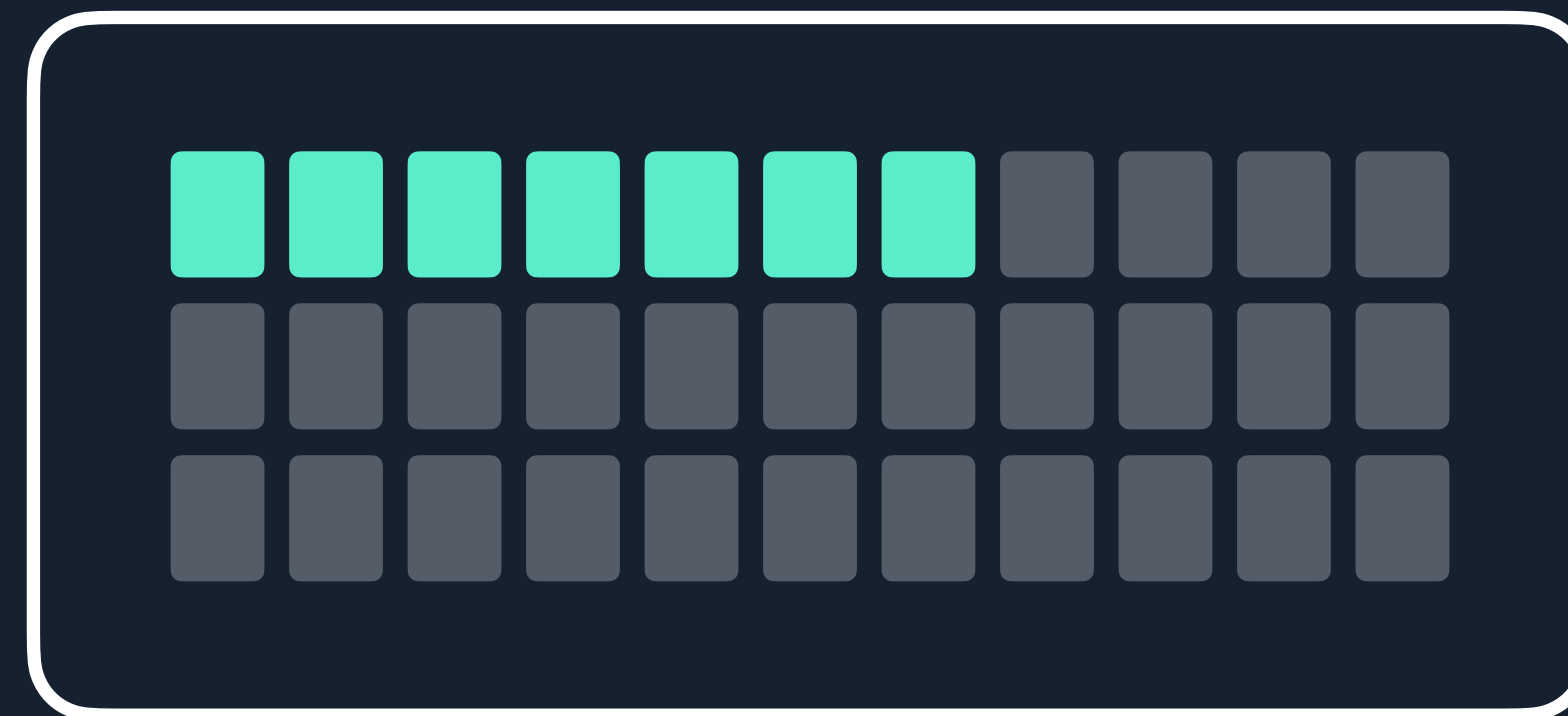
On demand



Deploy number 75

Code

Build



Deploy number 76

Code

Build



On demand



Deploy number 76



Code

Build



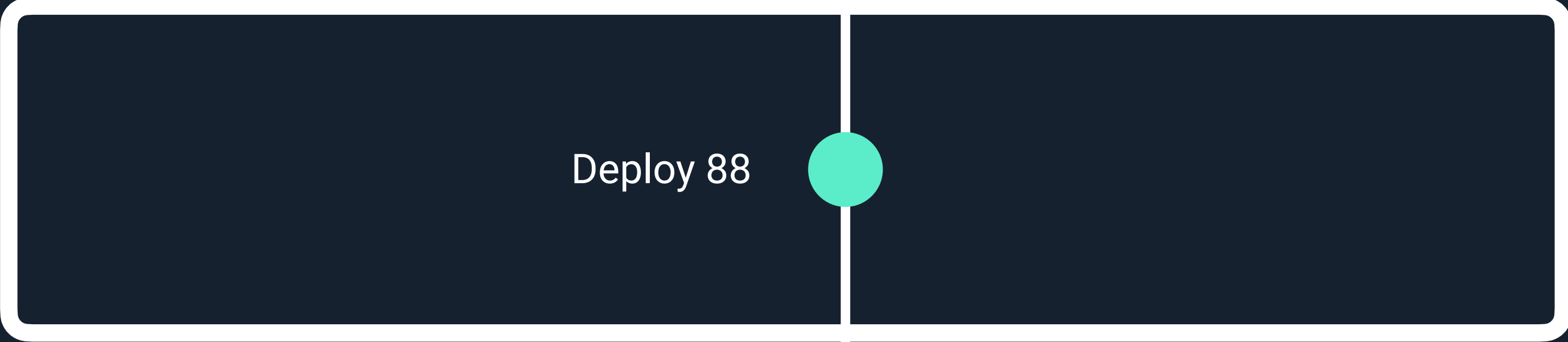
On demand



Deploy number 76



Live

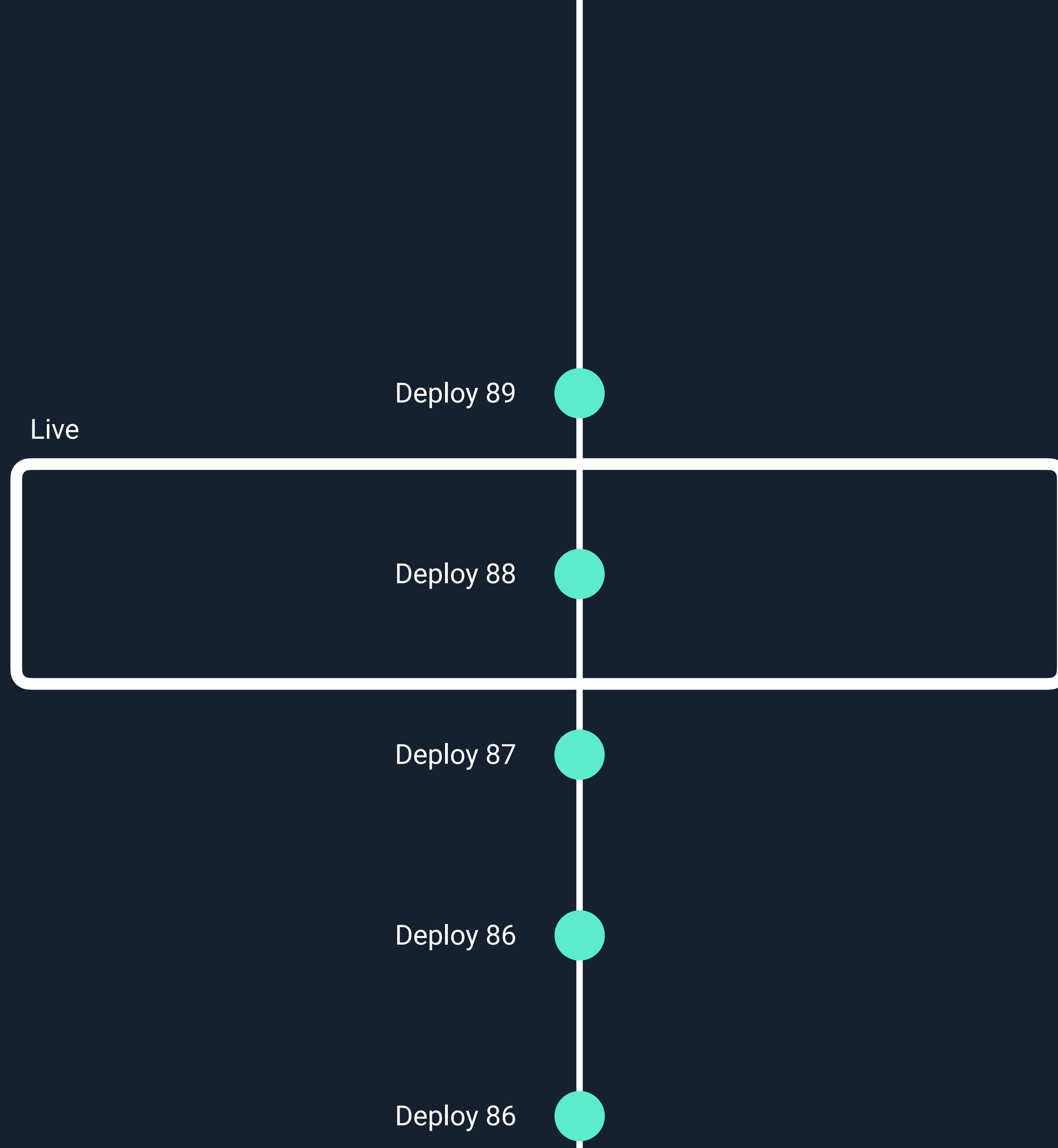


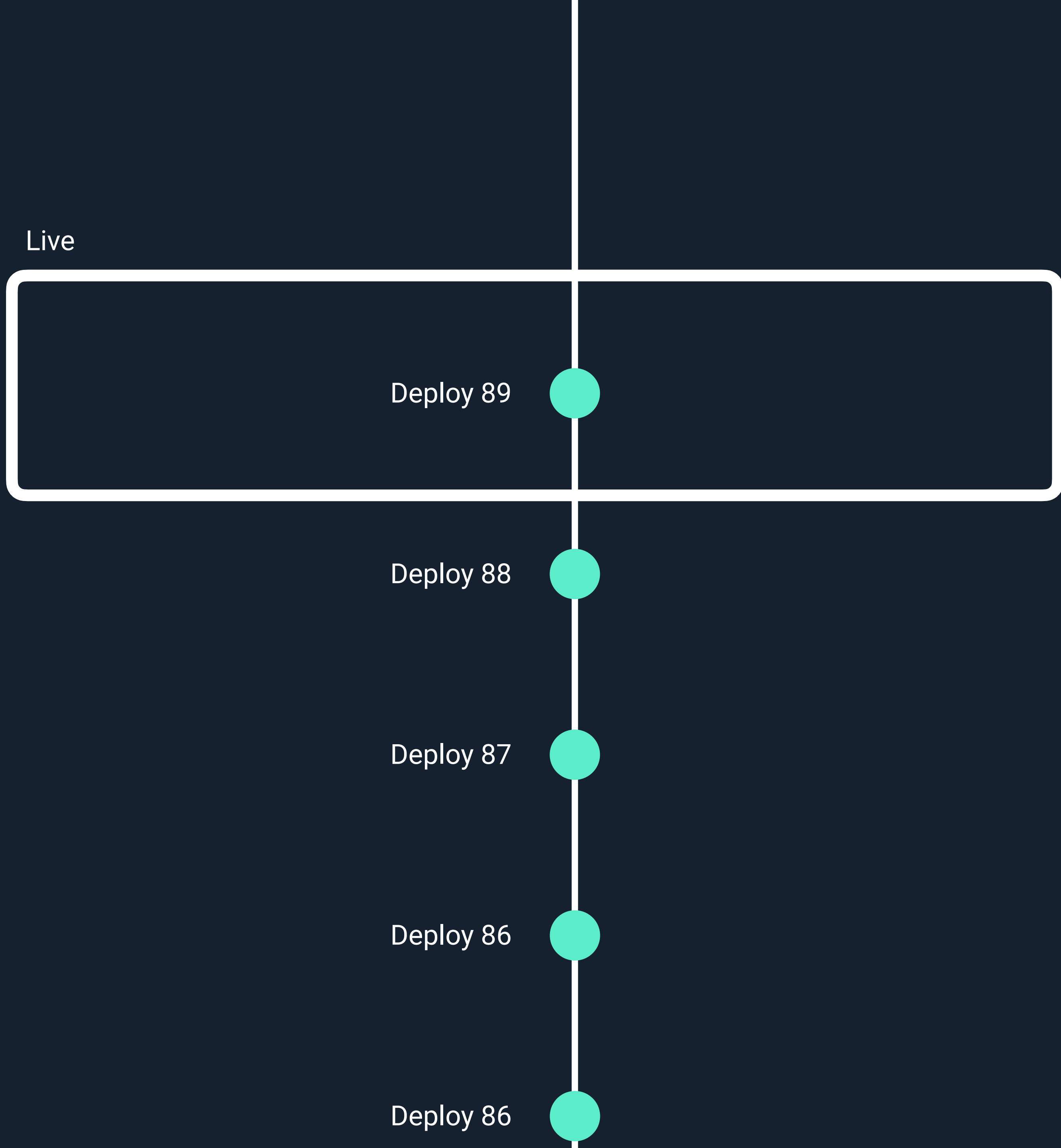
Deploy 88

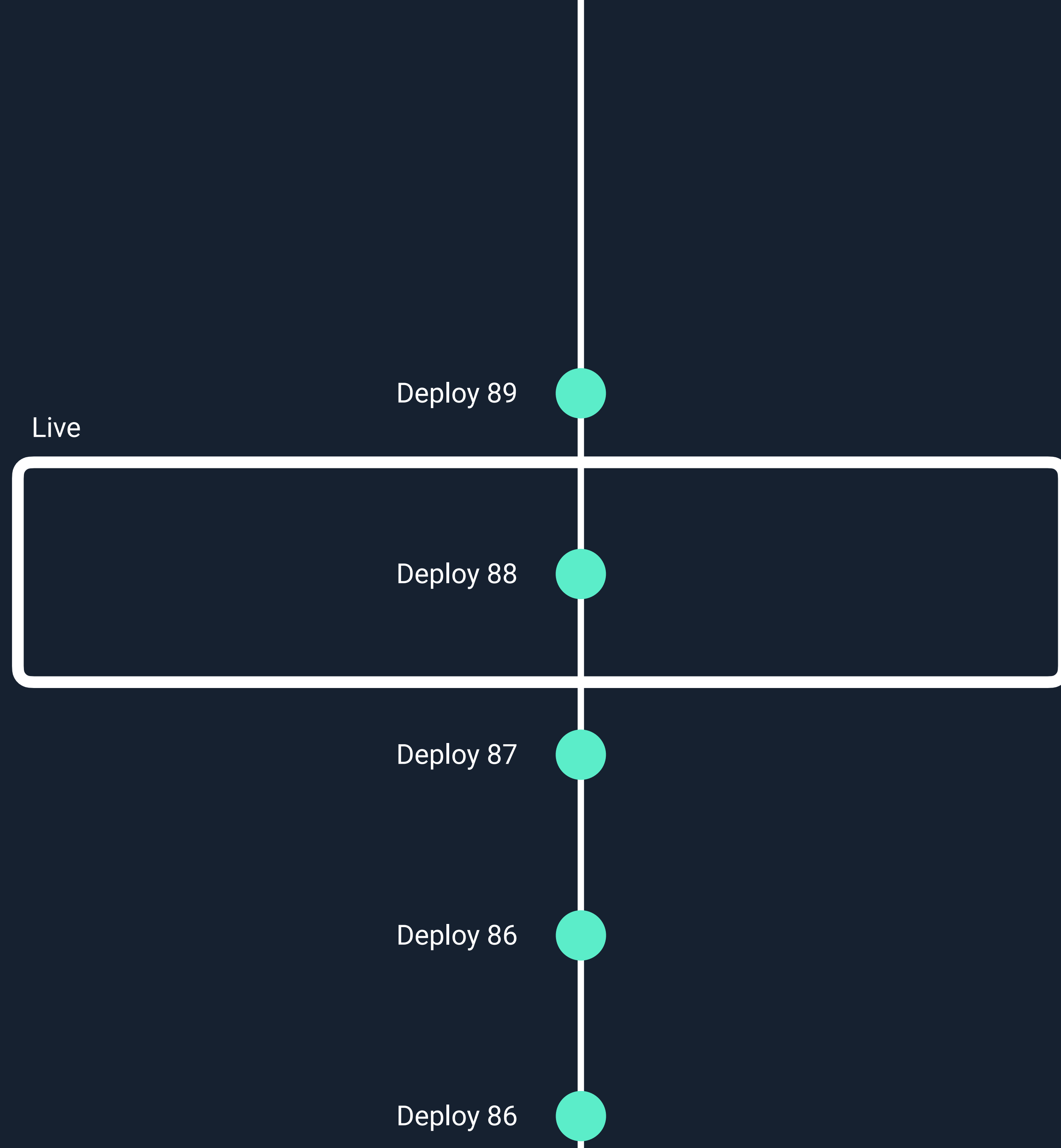
Deploy 87

Deploy 86

Deploy 86







A serverless function

```
const handler = async event => {  
  // return a view  
};
```

A serverless function

```
const { builder } = require("@netlify/functions");  
  
const handler = async event => {  
  // return a view  
};  
  
exports.handler = builder(handler);
```


Help to refine this pattern

findthat.at/dpr/rfc

findthat.at/dpr/csstricks

Bring Your Own

What are the underlying primitives you need?

**Use with any generator
or framework**

**A site without complex
interdependencies
between pages**

+

**An tool which can
generate pages from
a set of data**

+

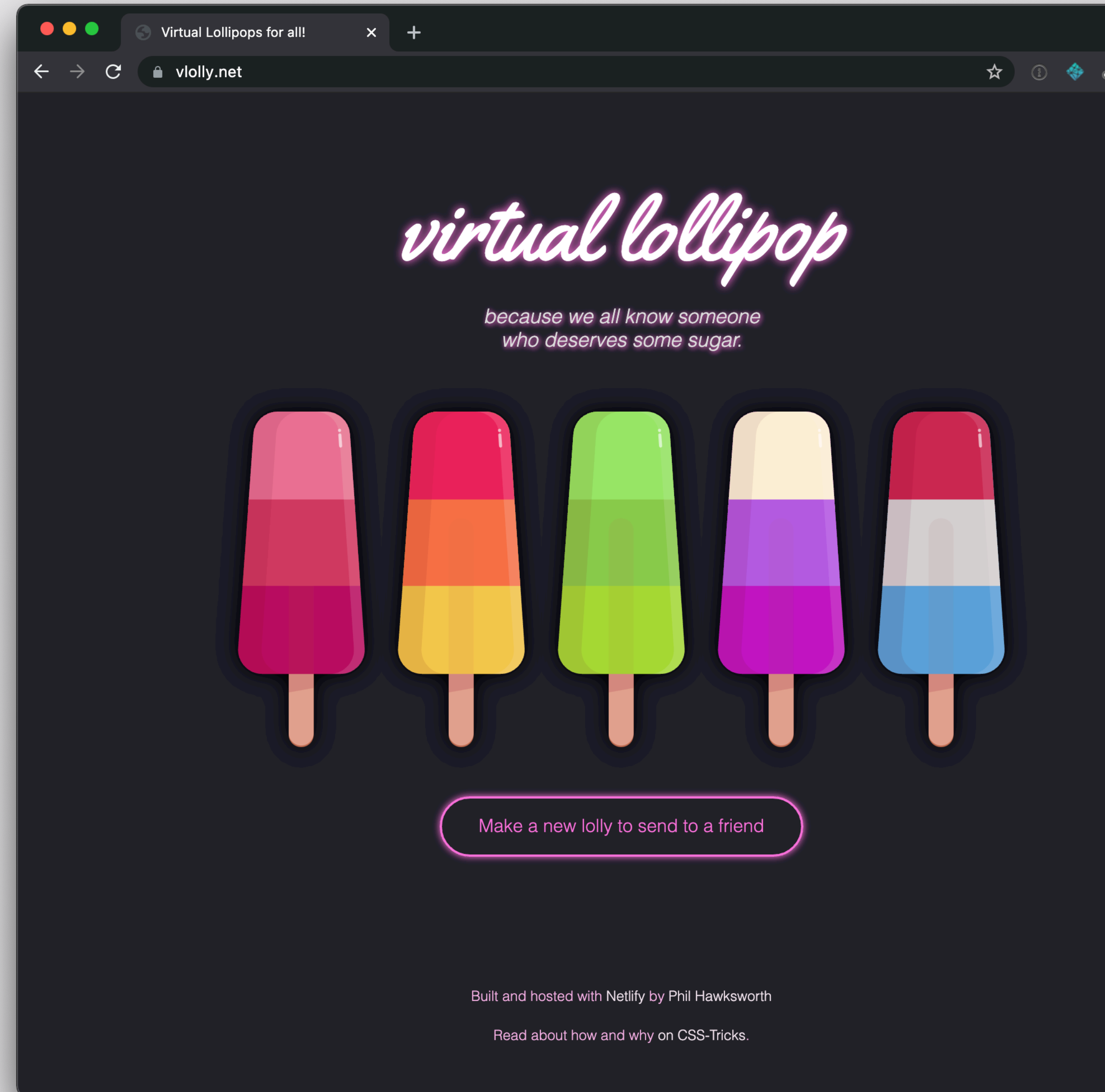
**The ability to cache
things between builds**

lolly.net

**thousands of pages of
user content**

**Each new deploy comes
from the build cache**

**And then generates new
pages and adds them to
the build cache**



2

Understanding the benefits and the sacrifices

Generating pages on-demand

BRINGS

**The ability to serve far
larger sites**

**The ability to deliver
previously unknown,
dynamic content**

BUT

Generating pages on-demand

**Can re-introduce the uncertainty
of what lives at a URL**

**Can violate the contract of
atomic, immutable builds**

**Can make architectures harder
to reason about**

So consider your approach carefully

Evaluate the functional and non-functional requirements of your project

3

**A practical example
of a first step to
incremental builds**

findthat.at

findthat.at/thecode

findthat.at - A URL shortener p x +

hawksworx.com/blog/find-that-at/

hawksworx blog speaking about search

January 4th 2019 by [Phil Hawksworth](#)

FINDTHAT.AT - A URL SHORTENER POWERED BY NETLIFY

#netlify
#tips

Previously posted:
[Keeping Sass simple and speedy on Eleventy](#)


The source code of this site is available on [GitHub](#) and is hosted and updated by [Netlify](#) automatically after each [code commit](#).

Other than where specified, the content on this site is published under a [Creative Commons Attribution 3.0 licence](#).

Subscribe to a [feed](#) of blog posts on this site.

I've recently started to roll my own short URLs. Using Netlify's optimised edge redirects via Netlify's redirects API is incredibly efficient and gives me URLs which I manage on my own domain instead of farming that out to a third-party provider who might go away.

It is remarkably easy to set up. Here's how.



The redirect API

Netlify allows you to run redirects directly on their ADN by adding a `_redirects` file to your project. You can read more details in the [documentation](#), but that file contains lines of config which look a little like this:

```
/a-short-path http://some-place-on-the-web  
...
```

`_redirects`

```
/thecode      https://github.com/philhawksworth/findthat
/incremental  https://noti.st/philhawksworth
/supasupa    https://www.netlify.com/blog/2021/06/28/sa
/magic-roundabout https://en.wikipedia.org/wiki/Magic_Rounda
/learn/graphql https://graphql.org/learn/
/eleventail   https://www.hawksworx.com/blog/eleventail-
/cal          https://calendar.google.com/calendar/u/0/r
/jamstack/london https://app.experiencewelcome.com/events/9
/jamstack/growing https://noti.st/philhawksworth/5Zh3rm/jams
```



findthat.at/thecode



findthat.at/qr/thecode



findthat.at/thecode

<https://github.com/philhawsworth/findthat.at>



BEHAVIOURS

**Render the page
the first time it is requested**

**Persist the page as part of
the latest deploy**

**Start fresh after
each new deploy**

We'll need

1

A function to build the pages on-demand and persist them

2

A way to direct requests to our function

`_redirects`

```
/magic-roundabout https://en.wikipedia.org/wiki/Magic_Roundabout
/learn/graphql https://graphql.org/learn/
/eleventail https://www.hawksworx.com/blog/eleventail-
/cal https://calendar.google.com/calendar/u/0/r
/jamstack/london https://app.experiencewelcome.com/events/9
/jamstack/growing https://noti.st/philhawksworth/5Zh3rm/jams

/qr/* /.netlify/functions/show-qr 200
```

show-qr.js

```
const { builder } = require("@netlify/functions");
const QRCode = require('qrcode');
const fetch = require('node-fetch');
const pageTemplate = require('../includes/page.js');
const rootURL = "https://findthat.at";

const handler = async event => {

  // Get the original short URL (without the qr part of the path)
  const path = event.path.split("/qr/")[1];
  const shortURL = `${rootURL}/${path}`;

  // follow the redirect to get the destination to display
  const destinationURL = await fetch(shortURL);

  // make a QR code and then return a page displaying it
  return QRCode.toString(shortURL, {type:'svg'})
  .then(svg => {

    // render the data into the template
    return {
      statusCode: 200,
      body: pageTemplate({
        shortURL : shortURL,
        destinationURL : destinationURL.url,
        svg: escape(svg)
      })
    };
  })
  .catch(err => {
    console.error(err)
  })
};

exports.handler = builder(handler);
```

show-qr.js

```
const handler = async event => {  
  
  // Get the original short URL (without the qr part of the path)  
  const path = event.path.split("/qr/")[1];  
  const shortURL = `${rootURL}/${path}`;  
  
  // follow the redirect to get the destination to display  
  const destinationURL = await fetch(shortURL);  
  
  // make a QR code and then return a page displaying it  
  return QRCode.toString(shortURL, { 'type': 'svg' } )  
  .then(svg => {  
  
    // render the data into the template  
    return {  
      statusCode: 200,  
      body: pageTemplate({  
        shortURL : shortURL,  
        destinationURL : destinationURL.url,  
        svg: escape(svg)  
      })  
    };  
  })  
  .catch(err => {  
    console.error(err)  
  })  
};
```

show-qr.js

```
const { builder } = require("@netlify/functions");
const QRCode = require('qrcode');
const fetch = require('node-fetch');
const pageTemplate = require('../includes/page.js');
const rootURL = "https://findthat.at";
```

page.js

```
module.exports = (data) => {  
  return `<!DOCTYPE html>  
  ...  
  <body>  
    <header>  
      <h1><a href="${ data.shortURL }">${ data.shortURL }</a></h1>  
      <a class="dest" href="${ data.destinationURL }">${ data.destinationURL }</a>  
    </header>  
    <main>  
      <img src='data:image/svg+xml;utf8,${ data.svg }'>  
    </main>  
    ...  
  `;  
}
```

show-qr.js

```
const { builder } = require("@netlify/functions");
const QRCode = require('qrcode');
const fetch = require('node-fetch');
const pageTemplate = require('../includes/page.js');
const rootURL = "https://findthat.at";

const handler = async event => {

  // Get the original short URL (without the qr part of the path)
  const path = event.path.split("/qr/")[1];
  const shortURL = `${rootURL}/${path}`;

  // follow the redirect to get the destination to display
  const destinationURL = await fetch(shortURL);

  // make a QR code and then return a page displaying it
  return QRCode.toString(shortURL, {type:'svg'})
  .then(svg => {

    // render the data into the template
    return {
      statusCode: 200,
      body: pageTemplate({
        shortURL : shortURL,
        destinationURL : destinationURL.url,
        svg: escape(svg)
      })
    };
  })
  .catch(err => {
    console.error(err)
  })
};

exports.handler = builder(handler);
```

show-qr.js

```
const { builder } = require("@netlify/functions");
```

```
const handler = async event => {
```

```
  ...
```

```
};
```

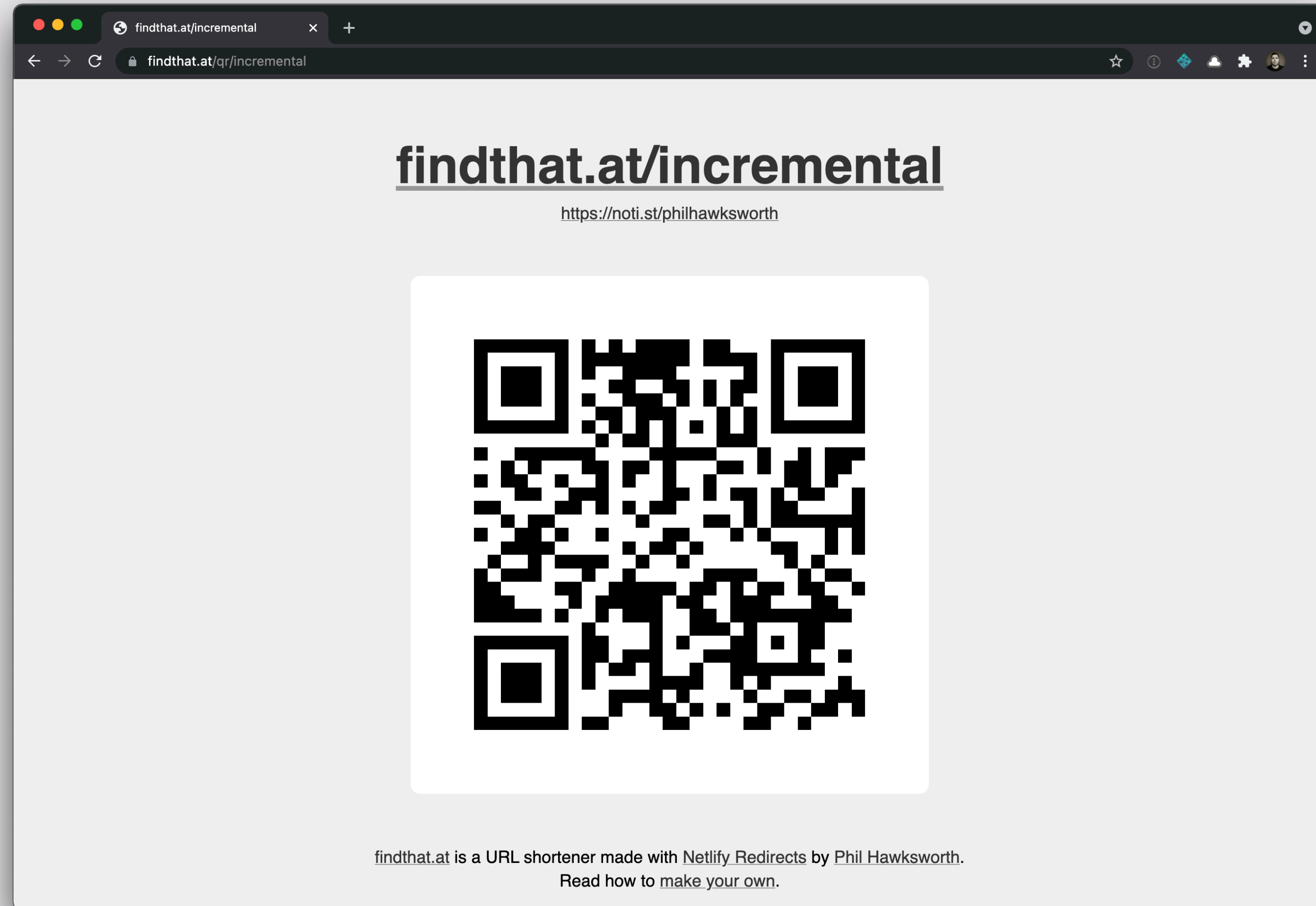
```
exports.handler = builder(handler);
```

These slides and links

findthat.at/incremental
findthat.at/qr/incremental



findthat.at/qr/incremental



Wrapping up

REMEMBER

**Incremental builds are
possible right now**

**Sites built with any
framework or generator
can be enhanced**

**Embracing the law of
least power helps protect
an architecture you can
reason about**

REMEMBER

**The benefits of a
Jamstack architecture
are worth protecting**

Examine your use cases

**Please don't make me
have to understand and
manage end-to-end
caching**

Jamstack Community Survey 2021

findthat.at/jamstack/survey

Jamstack Conf - Speak!

jamstackconf.com/cfp

Thanks
@philhawksworth

findthat.at/incremental