



# Building cross-site search

Supporting a continuous delivery  
solution across six sites

Chad Carlson  
Technical Writer  
Platform.sh  
[github.com/chadwcarlson](https://github.com/chadwcarlson)  
[chad.carlson@platform.sh](mailto:chad.carlson@platform.sh)

# Agenda

---

Platform.sh

Search and the great docs migration

Cross-site search on public docs

Public docs + search CD

Extending to other sites



**Chad Carlson**

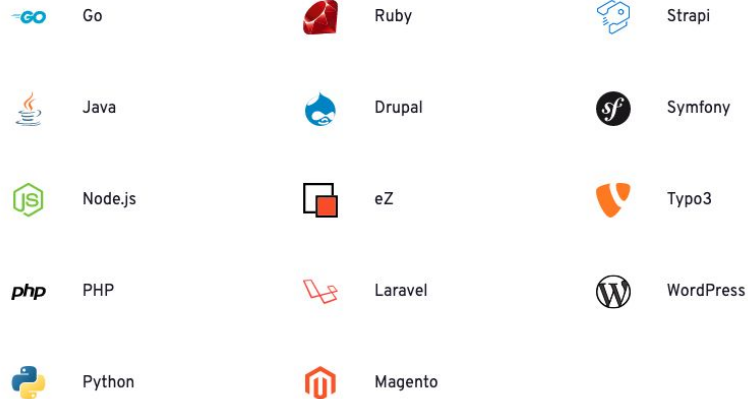
Technical writer

[github.com/chadwcarlson](https://github.com/chadwcarlson)

Writer, programmer, experimenter, ex-scientist.

# Platform.sh

- Platform-as-a-Service = Infrastructure abstraction
  - Polyglot (9 languages)
  - Batteries included services (13)
  - Explicit config for your cluster (3+ YAMLS)
- Built for continuous deployment, tying
  - each branch to an environment
  - each commit to a deployment
  - External integrations to your projects
- Makes merges to production
  - depend on successful builds
  - depend on successful tests
  - depend on successful deployment
- Results in
  - More frequent deployments (experiment)
  - Deploy Friday! (same image)
- Find out more at [docs.platform.sh](https://docs.platform.sh)



## Scale and secure your whole web app fleet

10 sites or 10,000? Platform.sh helps your team balance governance and

# Platform.sh

- Platform-as-a-Service = Infrastructure abstraction
  - Polyglot (9 languages)
  - Batteries included services (13)
  - Explicit config for your cluster (3+ YAMLS)
- Built for continuous deployment, tying
  - each branch to an environment
  - each commit to a deployment
  - External integrations to your projects
- Makes merges to production
  - depend on successful builds
  - depend on successful tests
  - depend on successful deployment
- Results in
  - More frequent deployments (experiment)
  - Deploy Friday! (same image)
- Find out more at [docs.platform.sh](https://docs.platform.sh)

The screenshot displays the Platform.sh web interface. At the top, there's a navigation bar with 'Projects > Platform.sh | Docs'. Below this, there are tabs for 'PROJECT Platform.sh | Docs' and 'ENVIRONMENT Select environment'. The main content area is divided into 'OVERVIEW', 'INTEGRATIONS', and 'SETTINGS'. The 'OVERVIEW' tab is active, showing 'Platform.sh | Docs' with a 'Live' button. Below this, there's a preview of the documentation page, which includes sections like 'Introduction', 'Configuration', and 'Best practices'. To the right of the documentation preview, there's a table with environment statistics:

22	Europe EU-3
No. of environments	Region
7	5 GB
Members	Storage

Below the table, there's a 'Services' section. On the far right, there's a sidebar titled 'Environments' with a 'Master' environment. Below this, there's a list of pull requests (PRs) with their titles:

- PR #1081: WIP initial Kafka documentation
- PR #1341: Revise sizing page to account for th
- PR #1413: Note about the build process
- PR #1423: Set conf\_dir to empty string.
- PR #1467: Document stoppable cron syntax.
- PR #1475: work around a composer\_manager
- PR #980: Build cache

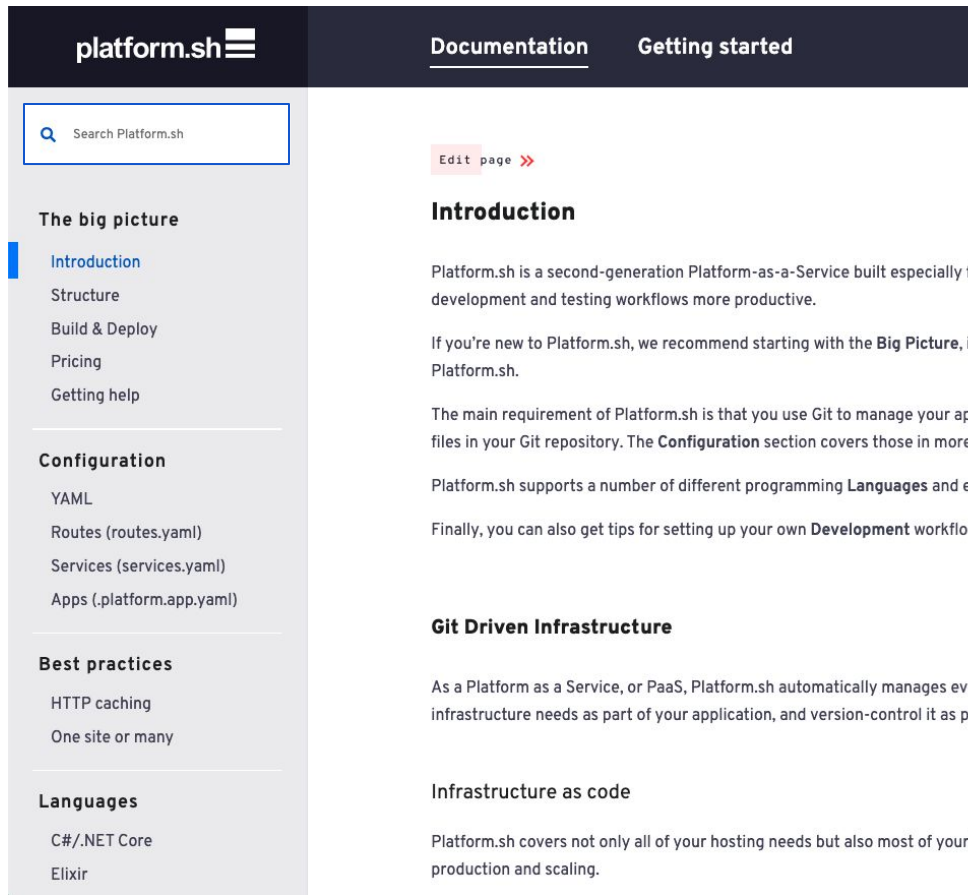
# Docs + search

- Previously used Gitbook
  - Node.js
  - Deprecated module ecosystem
  - Used Algolia plug-in for search
- Migrate to Hugo
  - Replicate Algolia?
- Meilisearch (Rust)
  - Executable search engine (multi-app)
  - Need to self-index docs (Hugo custom outputs)
- Side effect: Cross-site search!
  - Could scrape our other sites during build
  - Format documents/pages for Meilisearch
  - Include in the final index

The screenshot shows the Platform.sh documentation website. At the top, there is a dark navigation bar with the 'platform.sh' logo on the left and 'Documentation' and 'Getting started' links on the right. Below the navigation bar is a search bar with the placeholder text 'Search Platform.sh'. To the left of the main content area is a table of contents with sections: 'The big picture' (containing 'Introduction', 'Structure', 'Build & Deploy', 'Pricing', 'Getting help'), 'Configuration' (containing 'YAML', 'Routes (routes.yaml)', 'Services (services.yaml)', 'Apps (.platform.app.yaml)'), 'Best practices' (containing 'HTTP caching', 'One site or many'), and 'Languages' (containing 'C#/.NET Core', 'Elixir'). The main content area displays the 'Introduction' page, which includes an 'Edit page' link, the title 'Introduction', and several paragraphs of text describing Platform.sh as a second-generation Platform-as-a-Service, its requirements for using Git, and its support for various programming languages and development workflows.

# Docs + search

- Previously used Gitbook
  - Node.js
  - Deprecated module ecosystem
  - Used Algolia plug-in for search
- Migrate to Hugo
  - Replicate Algolia?
- Meilisearch (Rust)
  - Executable search engine (multi-app)
  - Need to self-index docs (Hugo custom outputs)
- Side effect: Cross-site search!
  - Could scrape our other sites during build
  - Format documents/pages for Meilisearch
  - Include in the final index



The screenshot shows the Platform.sh documentation website. At the top, there is a dark navigation bar with the Platform.sh logo on the left and the links "Documentation" and "Getting started" on the right. Below the navigation bar is a search bar with the placeholder text "Search Platform.sh". To the left of the main content area is a sidebar menu with the following sections: "The big picture" (containing links for Introduction, Structure, Build & Deploy, Pricing, and Getting help), "Configuration" (containing links for YAML, Routes (routes.yaml), Services (services.yaml), and Apps (.platform.app.yaml)), "Best practices" (containing links for HTTP caching and One site or many), and "Languages" (containing links for C#/.NET Core and Elixir). The main content area displays the "Introduction" page, which includes an "Edit page" link, a heading "Introduction", and several paragraphs of text describing Platform.sh as a second-generation Platform-as-a-Service built for development and testing workflows. It also mentions that users should use Git to manage their applications and that the documentation covers configuration, languages, and development workflows.

# Docs + search

- Previously used Gitbook
  - Node.js
  - Deprecated module ecosystem
  - Used Algolia plug-in for search
- Migrate to Hugo
  - Replicate Algolia?
- Meilisearch (Rust)
  - Executable search engine (multi-app)
  - Need to self-index docs (Hugo custom outputs)
- Side effect: Cross-site search!
  - Could scrape our other sites during build
  - Format documents/pages for Meilisearch
  - Include in the final index

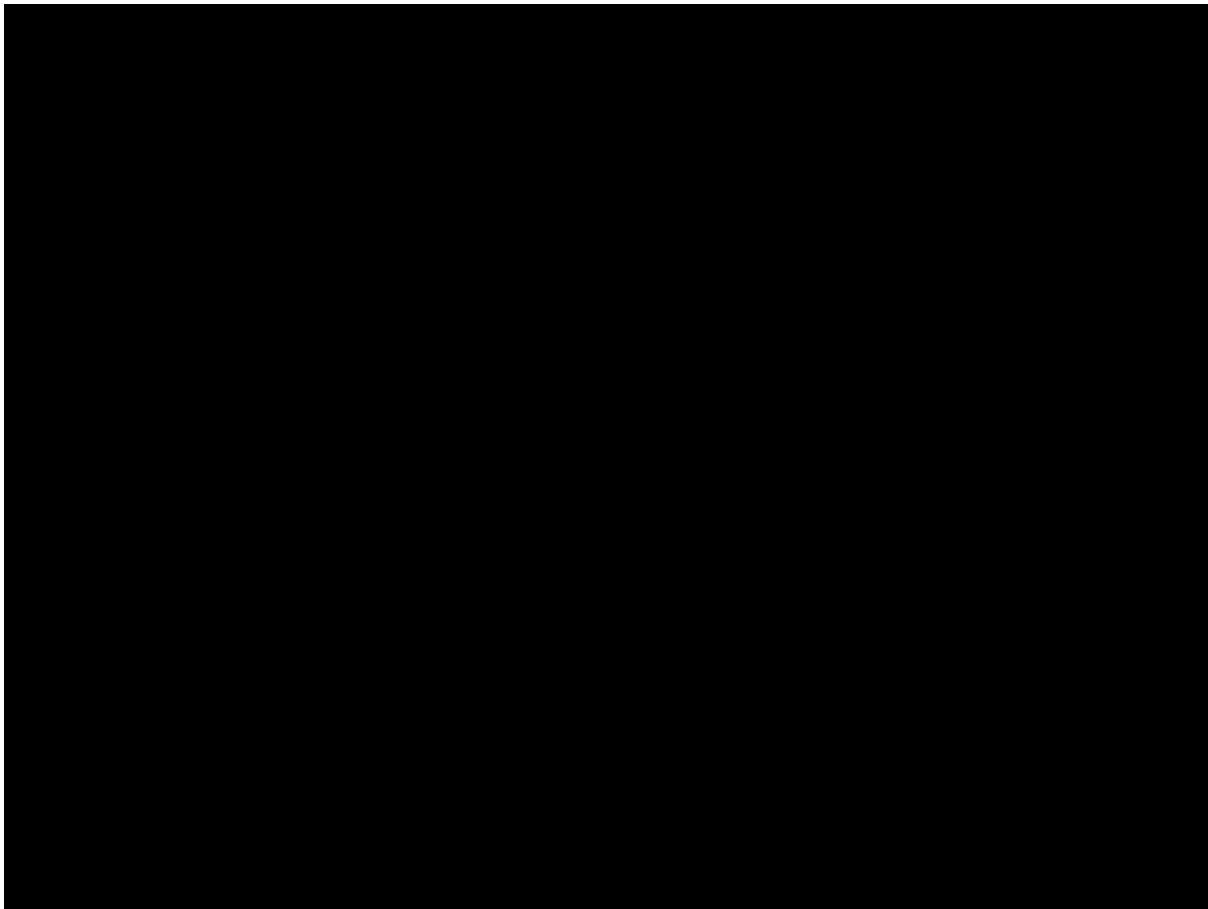
Search or jump to... Pull requests Issues Marketplace Explore

platformsh / platformsh-docs

<> Code Issues 8 Pull requests 7 Actions Projects Security Insights

Branch: master

	Crell committed 6a514cf 7 days ago	...
📁 .platform	Remove ancient redirects.	
📁 docs	Merge pull request #1473 from dev	
📁 search	Small fixes.	
📄 .editorconfig	Add a new empty book skeleton to	
📄 CONTRIBUTING.md	Removes redundant context. Adds	
📄 LICENSE.md	Rename license file for consistency	
📄 README.md	Fix bulk text parse/search. adds syn	





# XSS CD on docs

- Each pull request triggers new environment creation
- Each commit = new deployment
- Limits on merges, dependent on successful deploy
- Builds/deploys themselves depend on:
  - Indexing for search engine
    - If scraper tests fail, deploy fails
    - If scrape fails, deploy fails
    - If self-index fails, deploy fails
  - Posting to Meilisearch
    - Invalid document format = failed deploy
    - Unsuccessful POST = failed deploy
  - Hugo build
    - Fails if index did not create private key
- All of these must pass to be able to merge

The screenshot shows the Platform.sh | Docs dashboard. The top navigation bar includes 'Projects > Platform.sh | Docs', 'PROJECT Platform.sh | Docs', and 'ENVIRONMENT Select environment'. Below this are tabs for 'OVERVIEW', 'INTEGRATIONS', and 'SETTINGS'. The main content area is titled 'Platform.sh | Docs' with a 'Live' status indicator and 'Plan Medium'. It features a preview of the documentation page, a table of environment statistics, and a list of pull requests under the 'Environments' section.

22	Europe	EU-3
No. of environments	Region	
7	5 GB	
Members	Storage	

### Environments

Master

- PR #1081: WIP initial Kafka documentation
- PR #1341: Revise sizing page to account for th
- PR #1413: Note about the build process
- PR #1423: Set conf\_dir to empty string.
- PR #1467: Document stoppable cron syntax.
- PR #1475: work around a composer\_manager
- PR #980: Build cache

### Services

# XSS CD on docs

- Each pull request triggers new environment creation
- Each commit = new deployment
- Limits on merges, dependent on successful deploy
- Builds/deploys themselves depend on:
  - Indexing for search engine
    - If scraper tests fail, deploy fails
    - If scrape fails, deploy fails
    - If self-index fails, deploy fails
  - Posting to Meilisearch
    - Invalid document format = failed deploy
    - Unsuccessful POST = failed deploy
  - Hugo build
    - Fails if index did not create private key
- All of these must pass to be able to merge

```
name: 'search'
```

```
type: 'golang:1.14'
```

```
hooks:
```

```
  build: !include
```

```
    type: string
```

```
    path: build.sh
```

```
  post_deploy: !include
```

```
    type: string
```

```
    path: post_deploy.sh
```

```
web:
```

```
  commands:
```

```
    # Run the Meilisearch server
```

```
    start: "./meilisearch --http-addr localhost:${PORT}"
```

```
## build.sh ##
```

```
# Install Meilisearch
```

```
curl -L https://install.meilisearch.com | sh
```

```
# Build & test the indexer app
```

```
go test
```

```
go build
```

```
#####
```

```
## post_deploy.sh ##
```

```
# Build the index and post to meilisearch
```

```
./pshindex --index-config=config/index.yaml --meili-config=config/meili
```

# XSS CD on docs

- Each pull request triggers new environment creation
- Each commit = new deployment
- Limits on merges, dependent on successful deploy
- Builds/deploys themselves depend on:
  - Indexing for search engine
    - If scraper tests fail, deploy fails
    - If scrape fails, deploy fails
    - If self-index fails, deploy fails
  - Posting to Meilisearch
    - Invalid document format = failed deploy
    - Unsuccessful POST = failed deploy
  - Hugo build
    - Fails if index did not create private key
- All of these must pass to be able to merge



## Review required

At least 1 approving review is required by reviewers with write access. [Learn](#)



## All checks have passed

1 successful check



 **platformsh** — Platform.sh: Environment deployed



## Merging is blocked

Merging can be performed automatically with 1 approving review.

Merge pull request



You can also [open this in GitHub Desktop](#) or view co

Search Platform.sh

**The big picture**

- Introduction
- Structure
- Build & Deploy
- Pricing
- Getting help

**Configuration**

- YAML
- Routes (routes.yaml)
- Services (services.yaml)
- Apps (platform\_app.yaml)

**Best practices**

- HTTP caching
- One site or many

**Languages**

- C#/.NET Core
- Elisp

EDIT PAGE

### Introduction

Platform.sh is a second-generation Platform-as-a-Service built especially for continuous deployment. It allows you to host web applications on the cloud while making your development and testing workflows more productive.

If you're new to Platform.sh, we recommend starting with the **Big Picture**, in particular **Structure**, and **Build & Deploy** will get you started on the right track to best leverage Platform.sh.

The main requirement of Platform.sh is that you use Git to manage your application code. Your project's configuration is driven almost entirely by a small number of YAML files in your Git repository. The Configuration section covers those in more detail and can serve as both tutorial and quick-reference.

Platform.sh supports a number of different programming Languages and environments, and it features recommended optimizations for a number of Featured Frameworks.

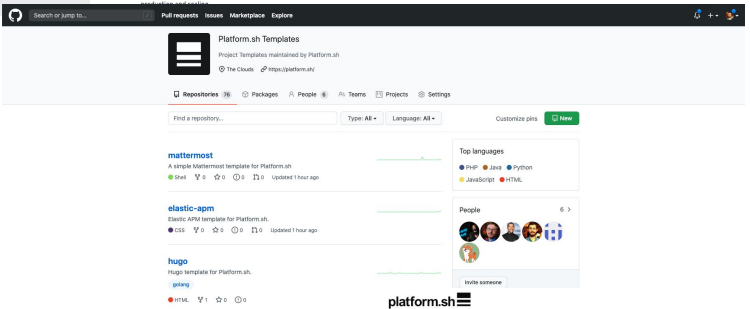
Finally, you can also get tips for setting up your own Development workflow and Administering your Platform.sh account.

### Git Driven Infrastructure

As a Platform as a Service, or PaaS, Platform.sh automatically manages everything your application needs in order to run. That means you can, and should, view your infrastructure needs as part of your application, and version-control as part of your application.

### Infrastructure as code

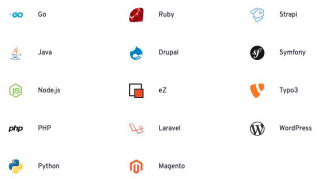
Platform.sh covers not only all of your hosting needs but also most of your DevOps needs. It is a simple, single tool that covers the application life-cycle from development to deployment and operations.



## Polyglot? Hyperpolyglot? We got you.

Platform.sh supports an array of cool languages and frameworks. Monoliths, Microservices. So you can keep your options open. Flexible. And future-proof.

Deploy 60+ software packages with a click



## Scales and secures your whole web app fleet

10 sites or 10,000? Platform.sh helps your team balance governance and

## TEMPLATES

- Types of templates
- Template Builder
- External Templates
- Template Definition Files
- Template Indexer

### Types of templates

Templates are Git repositories that a user may use to initialize a Platform.sh project. They are all intended to be work-out-of-the-box, or as close to that as is feasible. Templates include fully-ready applications (Drupal, Jenkins, WordPress), frameworks (Symfony, Ruby on Rails, Spring Boot), and proof-of-concept templates for specific language configurations. (The latter may be removed at some point in the future.)

Templates are primarily intended as starter kits, not examples, although they can and should be viewed as our officially recommended way to run certain systems.

There is no intention or expectation that users will update their projects post-install from a template. If they wish to do so (to get updated glue code, for instance), that is an entirely manual process.

At this time, there are two sources of templates: Template Builder and External.

### Template Builder

- Domain Management
- Cert Management
- Project Variables
- Repository
- Third-Party integrations
- Support

### Environment

On Platform.sh, an environment encompasses a single instance of your entire application stack, the services used by the application, the application's data storage, and the environment's backups.

In general, an environment represents a single branch or merge request in the Git repository backing a project. It is a virtual Cluster of read-only application and service containers with read-write mounts for application and service data.

On Platform.sh, the `default` branch is your production environment—thus, merging changes to master will put those changes to production.

### Activate an environment

Set the specified environment's status to active

AUTHORIZATIONS: OAuth2

PATH PARAMETERS:

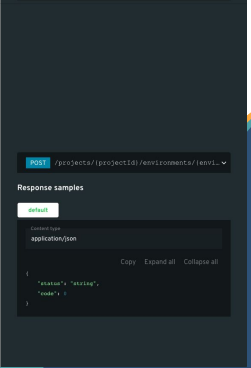
```

projectId string
environmentId string

```

Responses

> default



## Platform.sh Community

Welcome to the Platform.sh Community! Ask or answer any questions in the Q&A section, and browse the How-To Guides to learn how to do cool things with Platform.sh.

### Quick Links

- Getting Started
- Going Live
- Troubleshooting
- Support on Slack

Questions & Answers Latest Top Edit New Topic

Topic	Replies	Views	Activity
Is it possible to kill a build or deploy?	1	920	Mar '19
Using a custom certificate and Let's Encrypt certificates on the same project?	1	315	Jul '19
What does "WARNING: [pool web] server reached max_children setting (2), consider raising it?" mean?	1	961	May '19
How do I fix "fatal: the remote end hung up unexpectedly" while pushing from a Bitbucket pipeline to Platform.sh?	2	668	Mar '12



# XSS for all!

- Replicate the same model on each site
- Prioritize results for each with independent search server (all become multi-apps)
- We need a way to make index updates easy, regular
  - Source operations
  - Daily cron tests + re-indexes all sites
  - Can be triggered manually via CLI/API
- Now we have
  - A fleet of sites that can synchronize
  - Contain daily updates for each
  - An endpoint created to trigger on all, for big content launches (new language released; EOL announcements, etc.)
  - Each with inbuilt tests where merging to production is blocked if index generation, index POST, or site build fail anywhere.

The screenshot shows the platform.sh documentation website. The header includes the platform.sh logo and navigation links for 'Documentation' and 'Getting started'. A search bar is present with the text 'Search Platform.sh'. The main content area is titled '[Beta] Source operations' and includes a 'Contents' section with links to 'Source Operations with an external Git integration' and 'Automated Source Operations using cron'. A 'Note' section states that Source Operations are currently in Beta. A code block shows a configuration snippet for source operations, including an 'update' command that runs 'composer update' and commits the changes. The footer of the code block explains that the 'update' key is the name of the operation.

platform.sh

Documentation Getting started

Edit page >>

## [Beta] Source operations

Contents:

- [Source Operations with an external Git integration](#)
- [Automated Source Operations using cron](#)

An application can define a number of operations that apply to its source code.

**Note:**

Source Operations are currently in Beta. While the syntax is not expected to change, it is subject to change.

A basic, common source operation could be to automatically update Composer dependencies.

```
source:
  operations:
    update:
      command: |
        set -e
        composer update
        git add composer.lock
        git commit -m "Update Composer dependencies."
```

The `update` key is the name of the operation. It is arbitrary, and multiple source operations can be defined.

# XSS for all!

- Replicate the same model on each site
- Prioritize results for each with independent search server (all become multi-apps)
- We need a way to make index updates easy, regular
  - Source operations
  - Daily cron tests + re-indexes all sites
  - Can be triggered manually via CLI/API
- Now we have
  - A fleet of sites that can synchronize
  - Contain daily updates for each
  - An endpoint created to trigger on all, for big content launches (new language released; EOL announcements, etc.)
  - Each with inbuilt tests where merging to production is blocked if index generation, index POST, or site build fail anywhere.

```
# .platform.app.yaml
crons:
  update:
    # Run the 'update' source operation every day at 1:00am.
    spec: '0 1 * * *'
    cmd: |
      set -e
      if [ "$PLATFORM_BRANCH" = update-index ]; then
        platform environment:sync code data --no-wait --yes
        platform source-operation:run update_index --no-wait
      fi
source:
  operations:
    update_index:
      command: |
        set -e
        go test
        ./pshindex --index-config=config/index.yaml --meilisearch
        git add .
        git commit -m "Update Meilisearch index."
```

# THANK YOU!

Meet me in the Network  
Chat Lounge for questions