

The Serverless PHP Application

Rob Allen

The Online PHP Conference, January 2021

A black and white photograph of a factory floor. In the center, several workers wearing white caps and aprons are operating machinery. A large conveyor belt system with multiple levels moves numerous rectangular boxes or packages through the facility. The floor is made of metal grating, and overhead lighting creates strong shadows.

Serverless?

Platform options

Physical servers (Dell/HP)

Platform options

Virtual machines (EC2)

Physical servers (Dell/HP)

Platform options

Containers (Kubernetes)

Virtual machines (EC2)

Physical servers (Dell/HP)

Platform options

Platform (CloudFoundry)

Containers (Kubernetes)

Virtual machines (EC2)

Physical servers (Dell/HP)

Platform options

Serverless (OpenWhisk)

Platform (CloudFoundry)

Containers (Kubernetes)

Virtual machines (EC2)

Physical servers (Dell/HP)

Platform options



Abstraction

Serverless (OpenWhisk)

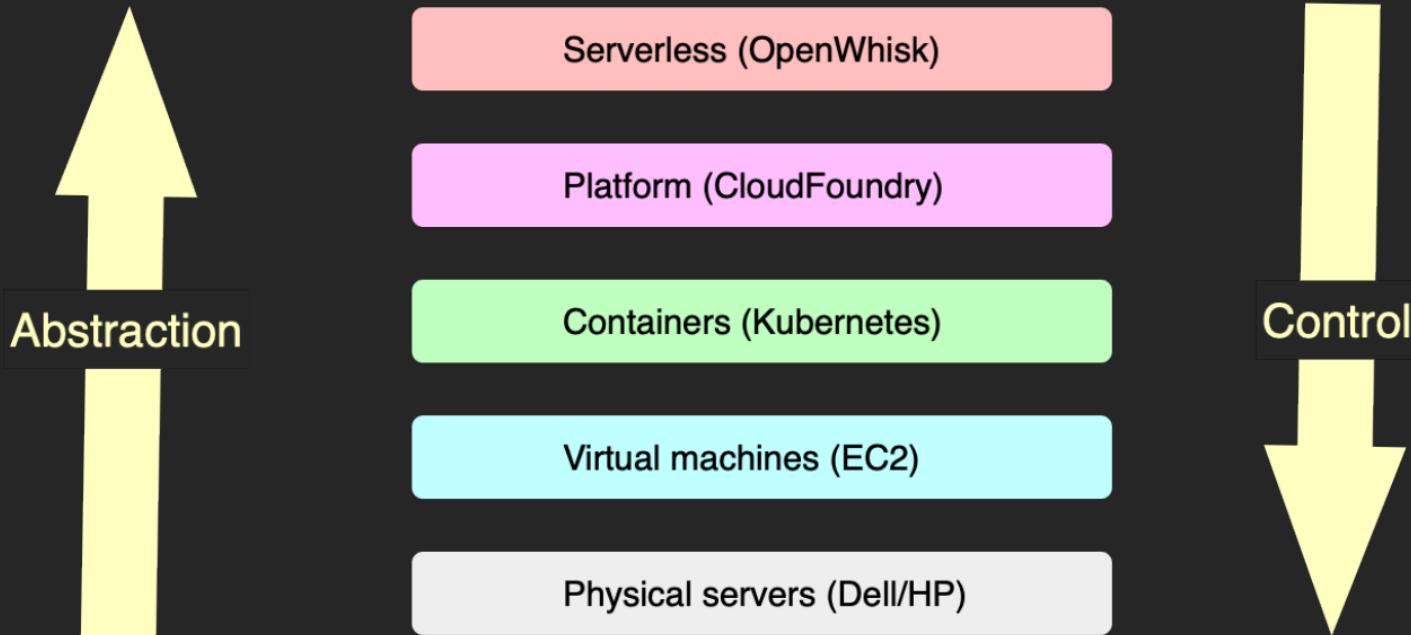
Platform (CloudFoundry)

Containers (Kubernetes)

Virtual machines (EC2)

Physical servers (Dell/HP)

Platform options



Serverless

Serverless is all about composing software systems from a collection of cloud services.

With serverless, you can lean on off-the-shelf cloud services resources for your application architecture, focus on business logic and application needs.

Nate Taggart, CEO Stackery

FaaS

Your code

FaaS

Deployed to the cloud

FaaS

Runs when needed

FaaS

Scaled automatically

FaaS

Pay only for execution

Where are the servers?





Rob Allen ~ @akrabat

Use-cases

Use-cases

Synchronous

Service is invoked and provides immediate response
(HTTP requests: APIs, chat bots)



Use-cases

Synchronous

Service is invoked and provides immediate response
(HTTP requests: APIs, chat bots)

Asynchronous

Push a message which drives an action later
(web hooks, timed events, database changes)



Benefits

Benefits

- No need to maintain infrastructure

Benefits

- No need to maintain infrastructure
- Concentrate on application code

Benefits

- No need to maintain infrastructure
- Concentrate on application code
- Pay only for what you use, when you use it

Benefits

- No need to maintain infrastructure
- Concentrate on application code
- Pay only for what you use, when you use it
- Language agnostic

Challenges

Challenges

- Start up latency

Challenges

- Start up latency
- Time limit

Challenges

- Start up latency
- Time limit
- State is external

Challenges

- Start up latency
- Time limit
- State is external
- Different way of thinking

When should you use serverless?

When should you use serverless?

- Responding to web hooks

When should you use serverless?

- Responding to web hooks
- Additional features without extending current platform



When should you use serverless?

- Responding to web hooks
- Additional features without extending current platform
- PWA/Static site contact form, et al.

When should you use serverless?

- Responding to web hooks
- Additional features without extending current platform
- PWA/Static site contact form, et al.
- Variable traffic levels

When should you use serverless?

- Responding to web hooks
- Additional features without extending current platform
- PWA/Static site contact form, et al.
- Variable traffic levels
- When you want your costs to scale with traffic

It's about value



Beau @BeauVrolyk · 30m

Replies to @akrabat @kelseyhightower

1) "Serverless" is a point on the path to true app isolation. Apps want to just run, their authors don't care about infrastructure at all.



1



2



Beau @BeauVrolyk · 29m

Replies to @akrabat @kelseyhightower

2) The App author should not need to know, anymore than a Journalist knows about printing presses or what the voltage of the power used.



1



2



Beau @BeauVrolyk · 25m

Replies to @akrabat @kelseyhightower

3) We are relearning what was known in the time-share days. Pricing needs to be based on something customers value, not infra. items like VMs



Serverless platforms



Google Cloud Platform



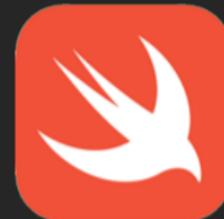
IBM Cloud



APACHE
OpenWhisk



Serverless languages



Serverless platforms with PHP support



Hello World

AWS Lambda (Bref):

```
<?php  
  
return function ($event) {  
    $name = $event['name'] ?? 'world';  
    return 'Hello ' . $name;  
};
```

Hello World

Apache OpenWhisk:

```
<?php

function main(array $args): array
{
    $name = $args['name'] ?? 'world';
    return ["greeting" => 'Hello ' . $name];
}
```

Hello World

OpenFaaS

```
<?php  
  
class Handler  
{  
    public function handle(string $data): void {  
        $decoded = json_decode($data, true);  
        $name = $decoded['name'] ?? 'world';  
        return 'Hello ' . $name;  
    }  
}
```

Hello World

Google Cloud Functions (alpha)

```
<?php

use Psr\Http\Message\ServerRequestInterface as Request;

function helloHttp(Request $request)
{
    $name = $request->getQueryParams( 'name' ) ?? 'world';
    return 'Hello ' . $name;
}
```



APACHE
OpenWhisk

The anatomy of an action

```
function main(array $args): array
{
    // Marshall inputs from event parameters
    $name = $args['name'] ?? 'world';
    // Do the work
    $message = 'Hello ' . $name
    // Return result
    return ["body" => $message];
}
```



Hello World

Entry point
Event parameters

```
function main(array $args): array
{
    // Marshall inputs from event parameters
    $name = $args['name'] ?? 'world';
    // Do the work
    $message = 'Hello ' . $name
    // Return result
    return ["body" => $message];
}
```

Hello World

```
function main(array $args): array
{
    // Marshall inputs from event parameters
    $name = $args['name'] ?? 'world';
    // Do the work
    $message = 'Hello ' . $name
    // Return result
    return ["body" => $message];
}
```



Hello World

```
function main(array $args): array
{
    // Marshall inputs from event parameters
    $name = $args['name'] ?? 'world';
    // Do the work
    $message = 'Hello ' . $name
    // Return result
    return ["body" => $message];
}
```



Hello World

```
function main(array $args): array
{
    // Marshall inputs from event parameters
    $name = $args['name'] ?? 'world';
    // Do the work
    $message = 'Hello ' . $name
    // Return result
    return ["body" => $message];
}
```



Deploy to OpenWhisk

```
$ zip -q hello.zip hello.php
```



Deploy to OpenWhisk

```
$ zip -q hello.zip hello.php
$ wsk action update --kind php:7.4 hello hello.zip
ok: updated action hello
```



Run it

```
$ wsk action invoke hello --result --param name Rob
```

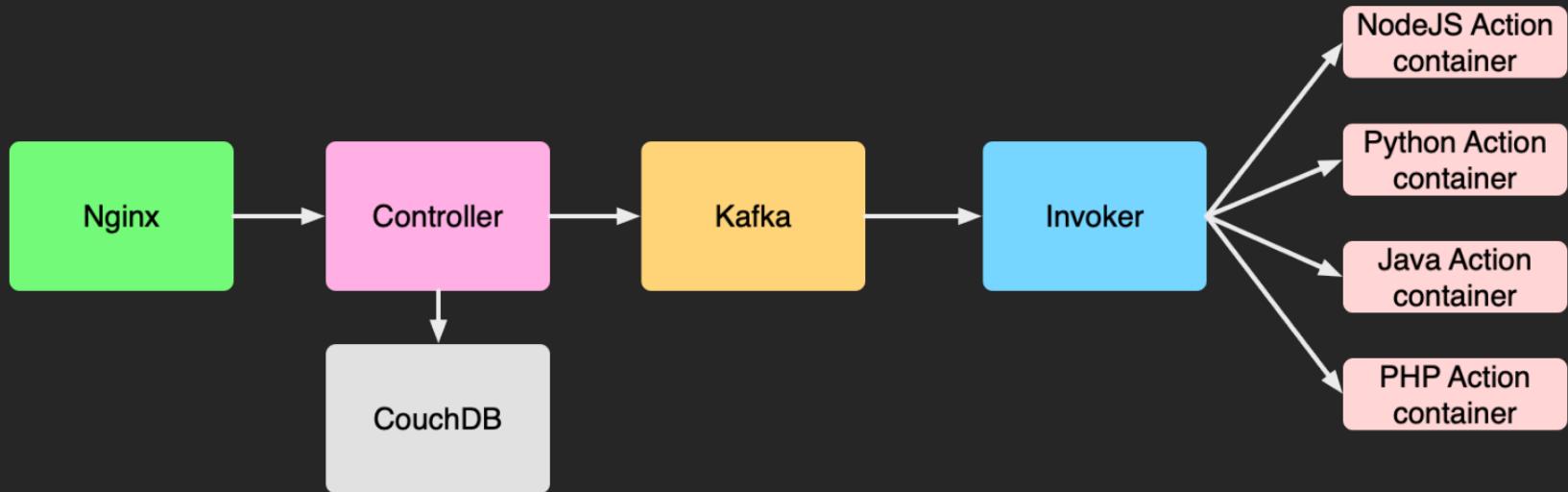
Run it

```
$ wsk action invoke hello --result --param name Rob
{
  "body": "Hello Rob!"
}
```

A teal and white classic car with its hood open, showing the engine. The car is parked on a street with other vehicles and buildings in the background.

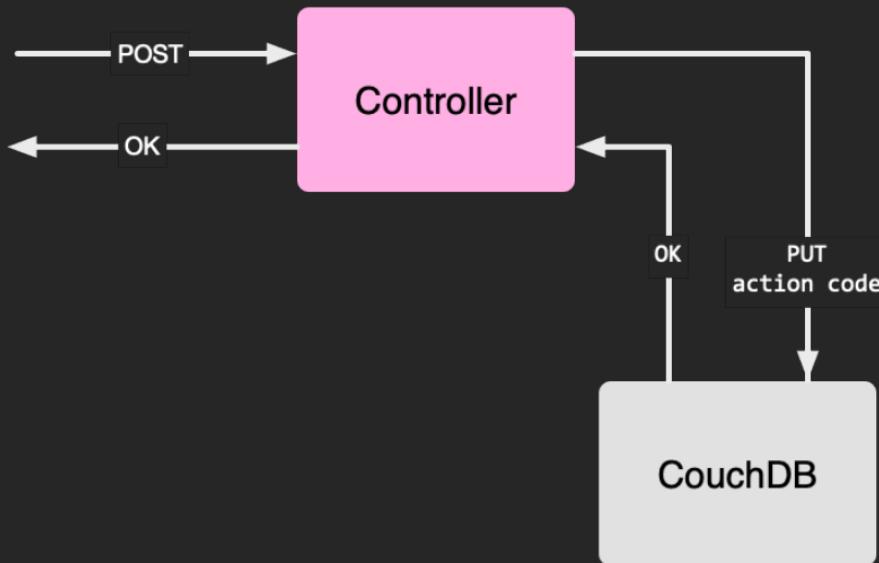
Under the hood

OpenWhisk's architecture



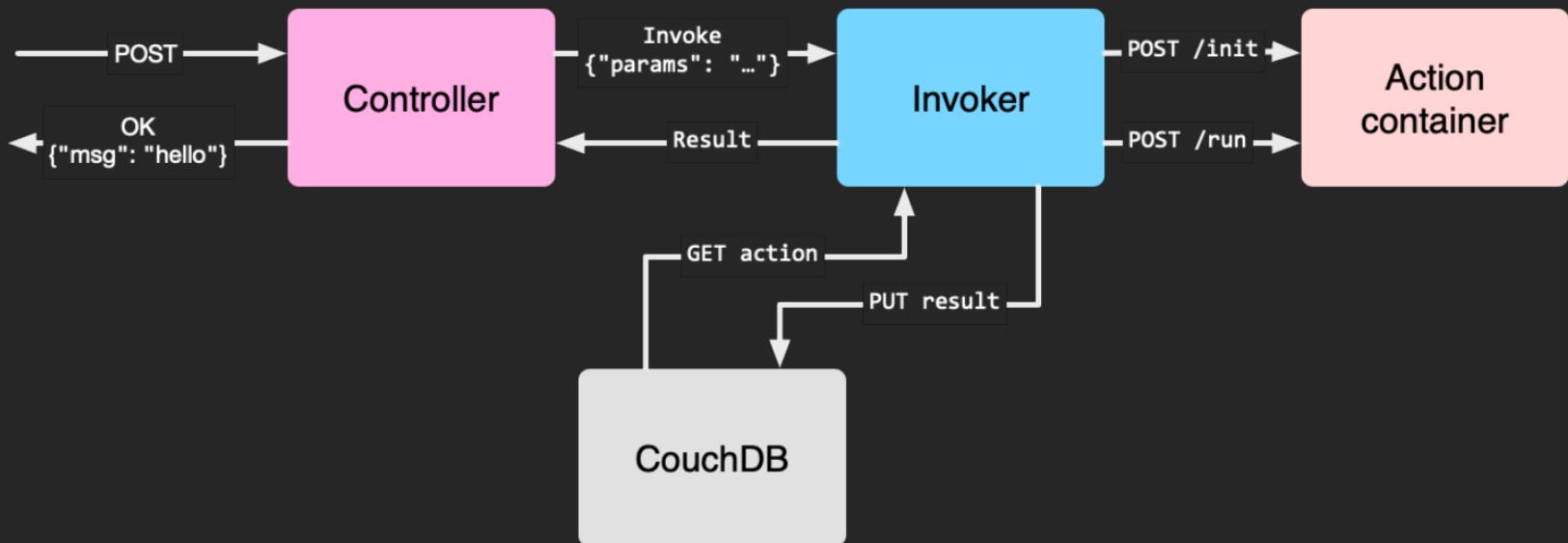
Create an action

```
$ wsk action create hello hello.php
```



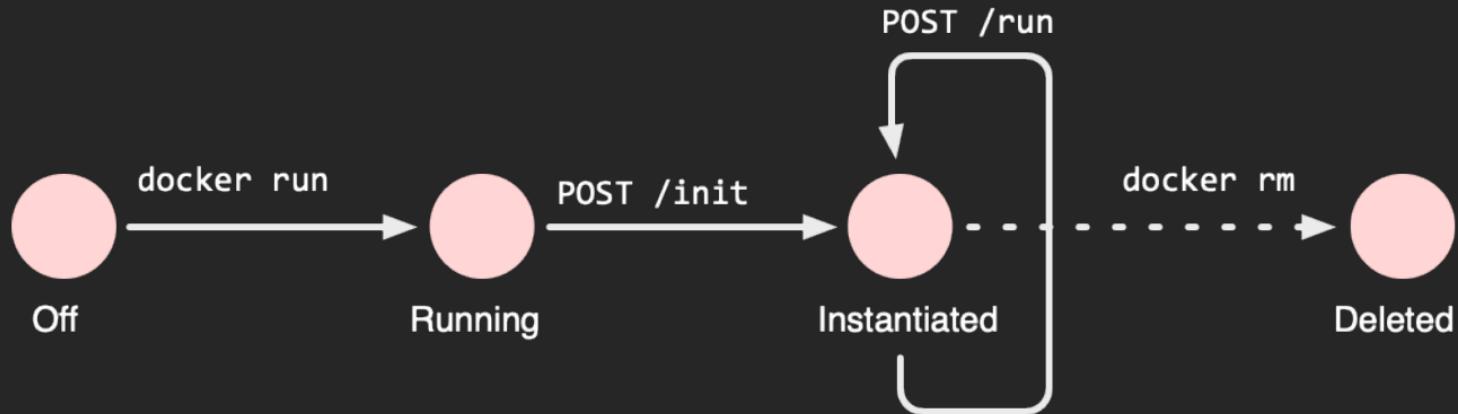
Invoke an action

```
$ wsk action invoke hello -r
```



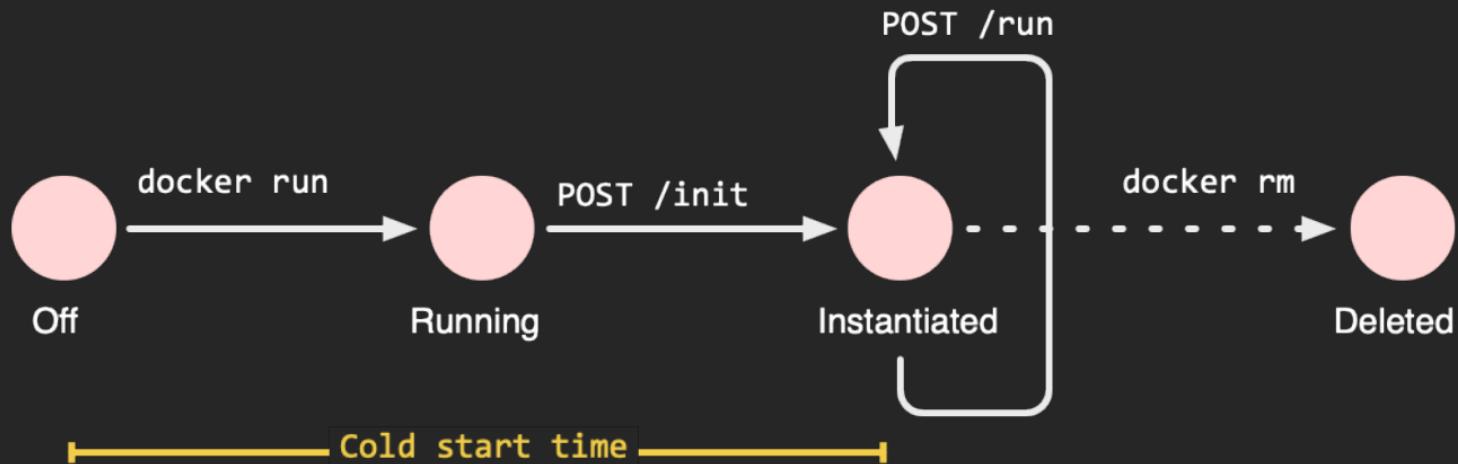
Action container lifecycle

- Hosts the user-written code
- Controlled via two end points: /init & /run



Action container lifecycle

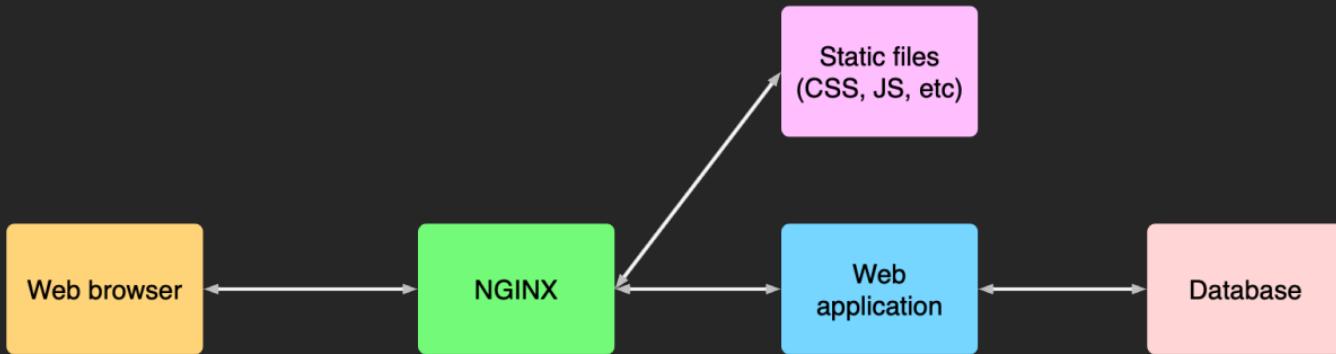
- Hosts the user-written code
- Controlled via two end points: /init & /run



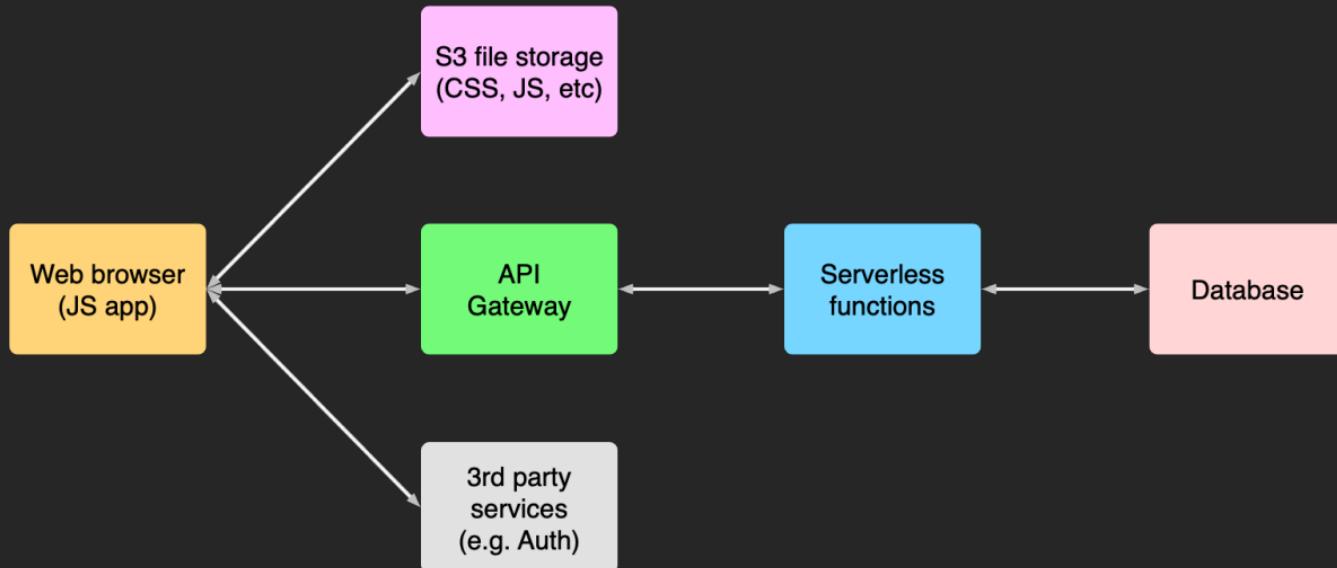


Architecture

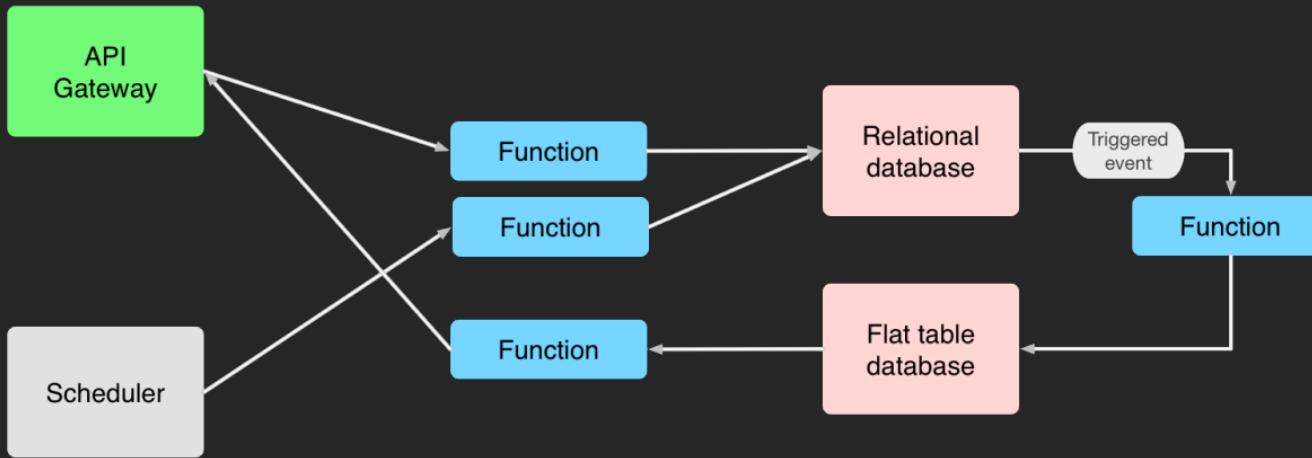
Monolith architecture



Serverless architecture



Serverless architecture pattern



Functions are key

Functions are the
Unit of Deployment



Functions are the
Unit of Scale



Functions are Stateless

Functions have
Structure

Structure

If it's non-trivial, software engineering principles apply!

- Use multiple methods

Structure

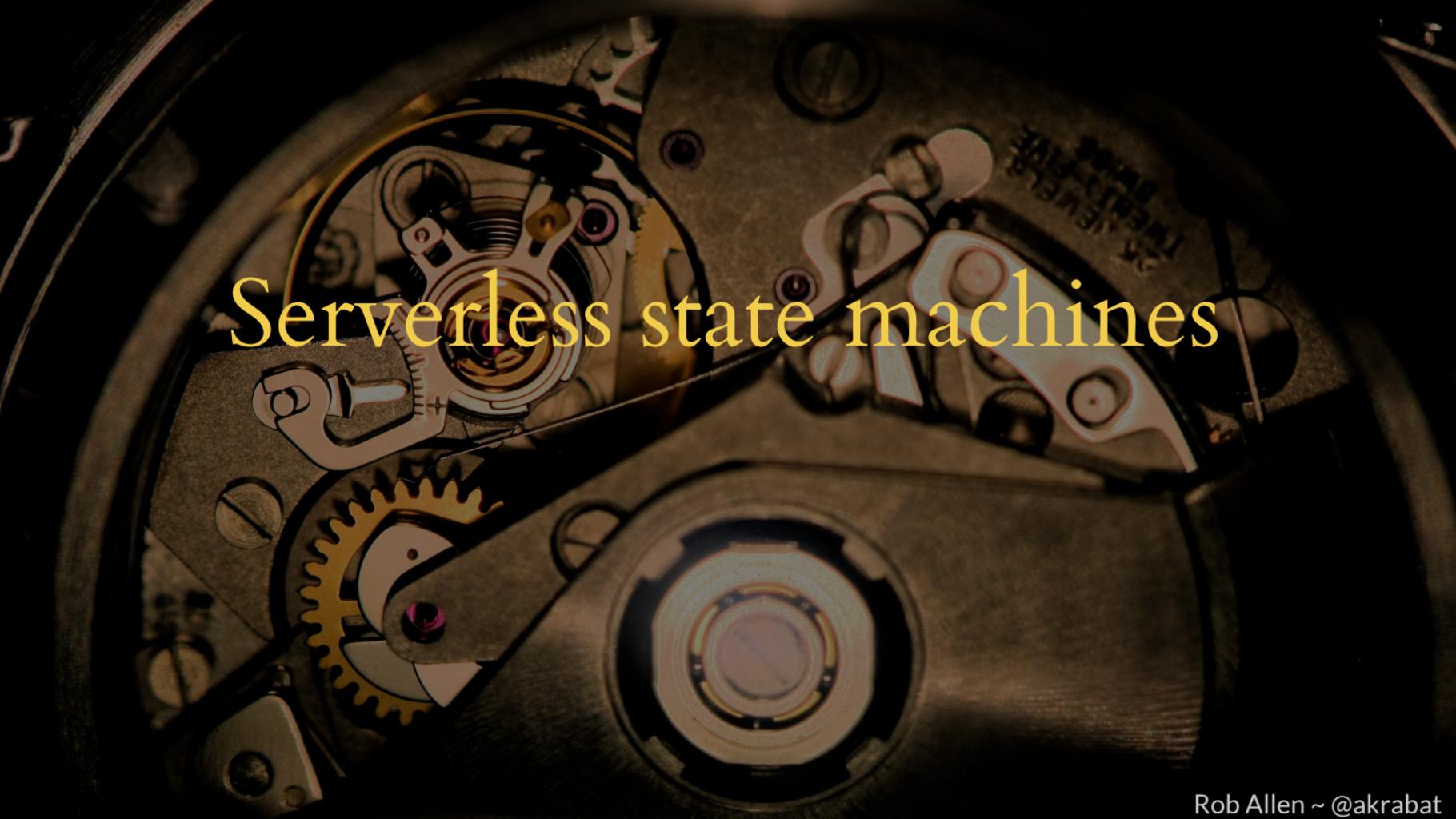
If it's non-trivial, software engineering principles apply!

- Use multiple methods
- Use multiple files

Structure

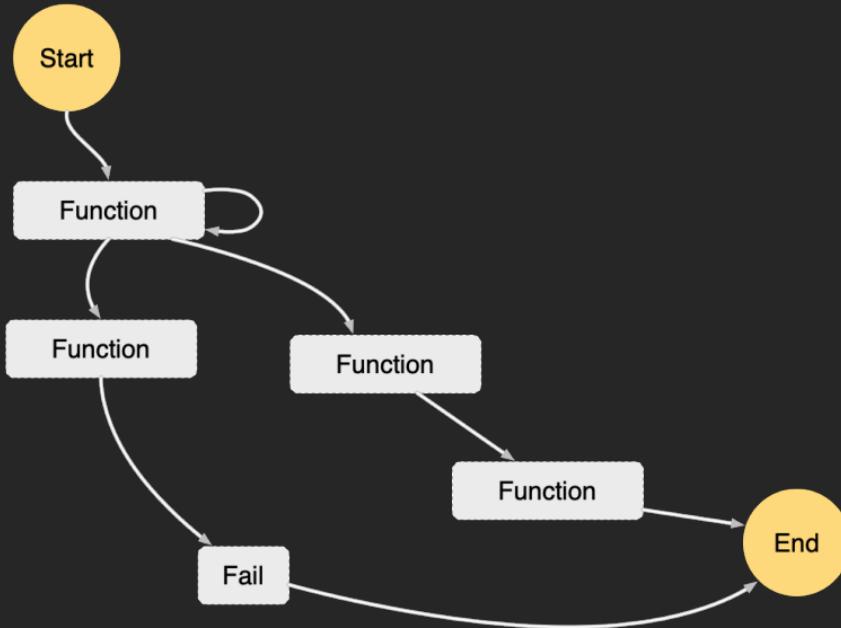
If it's non-trivial, software engineering principles apply!

- Use multiple methods
- Use multiple files
- Integrate reusable dependencies

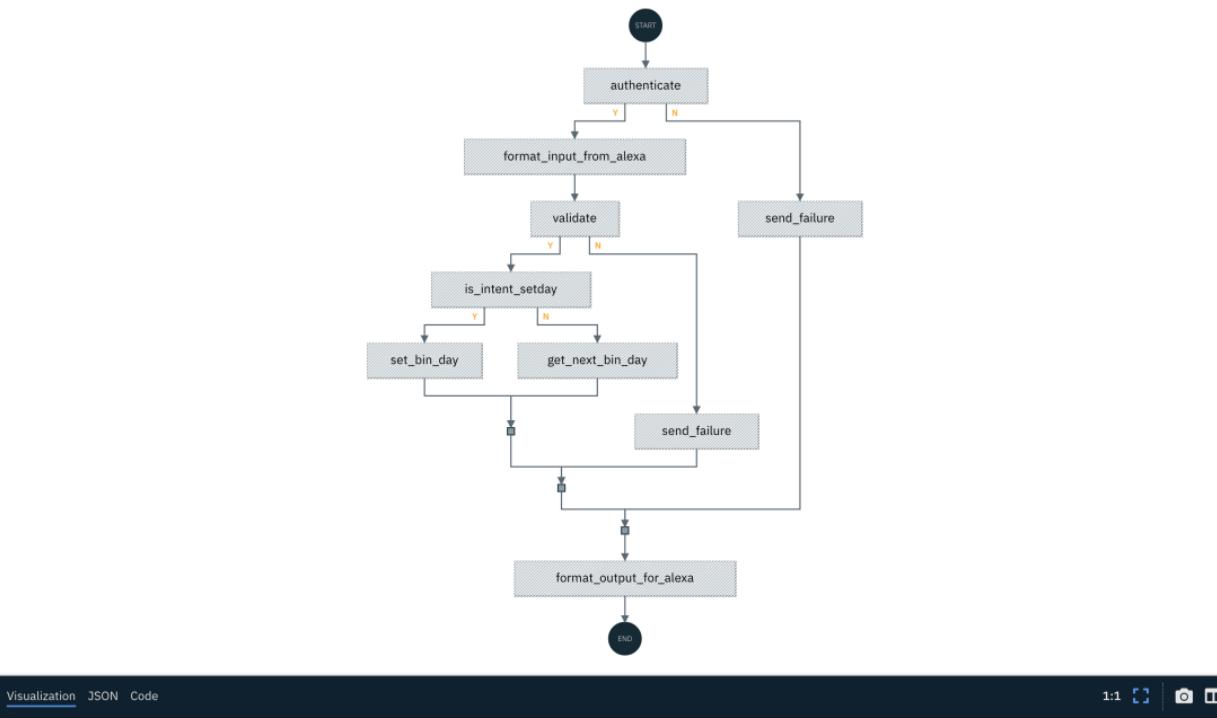


Serverless state machines

Serverless state machines

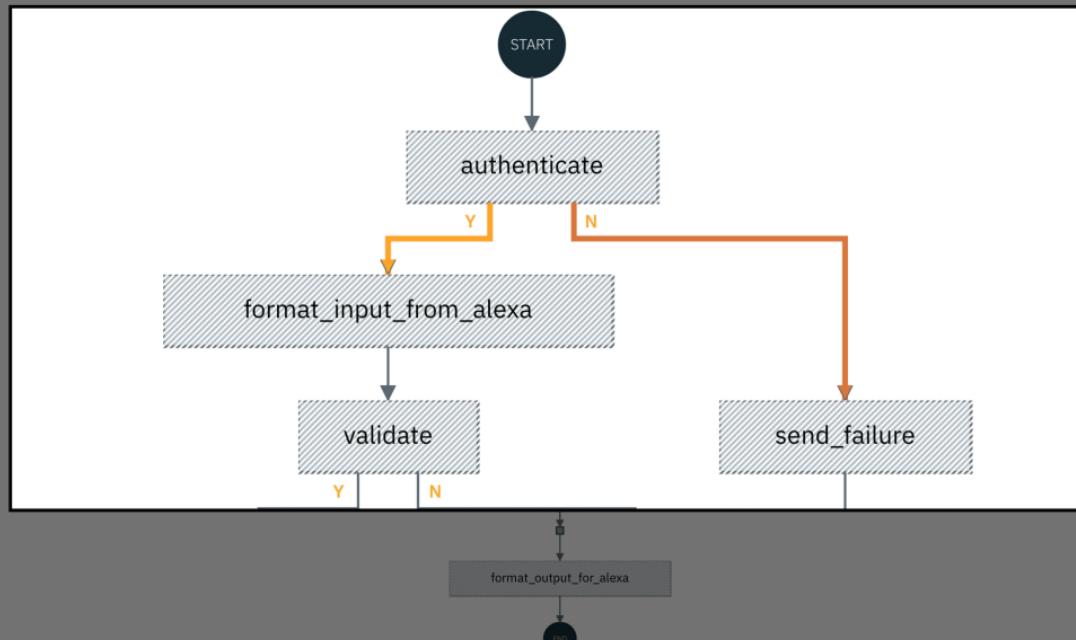


binday.js



binday.js

Composer





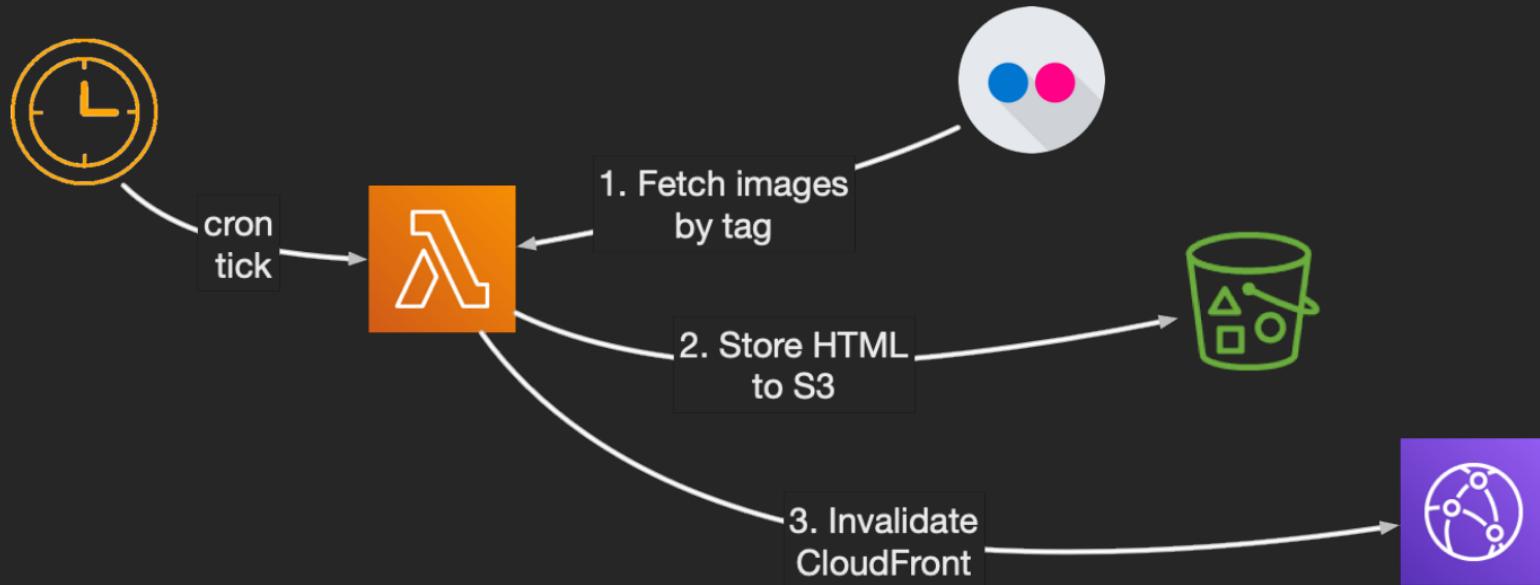
Case study Project 365 photo website

Project 365

Static website to display my photo-a-day picture for each day of the year.

- Hosted on S3
- CloudFront CDN
- Lambda/PHP function

Lambda/PHP function



Infrastructure as code

```
serverless.yml:  
  functions:  
    update:  
      handler: update.php  
      events:  
        - schedule:  
            name: project365-build  
            rate: cron(0 */2 * * ? *)
```



Infrastructure as code

```
functions:
```

```
update:
```

```
  handler: update.php
```

```
events:
```

```
  - schedule:
```

```
    name: project365-build
```

```
    rate: cron(0 */2 * * ? *)
```

Infrastructure as code

```
functions:  
update:  
  handler: update.php  
events:  
  - schedule:  
    name: project365-build  
    rate: cron(0 */2 * * ? *)
```

Process

1. Gather credentials from environment
2. Download photos from Flickr API
3. Create HTML page
4. Upload to S3
5. Invalidate CloudFront cache

main()

```
return function ($event) {
    $year = $event['year'] ?? date('Y');

    $photos = (new PhotoFetcher())->fetchForYear($year);
    $html = (new PageCreator())->create($year, $photos);

    $uploader = new Uploader();
    $uploader->uploadOne($year, $html, $s3Bucket);

    $uploader->invalidateCache(['/'. $year]);
}
```

main()

```
return function ($event) {
    $year = $event['year'] ?? date('Y');

    $photos = (new PhotoFetcher())->fetchForYear($year);
    $html = (new PageCreator())->create($year, $photos);

    $uploader = new Uploader();
    $uploader->uploadOne($year, $html, $s3Bucket);

    $uploader->invalidateCache( [ '/' . $year ] );
}
```

main()

```
return function ($event) {
    $year = $event['year'] ?? date('Y');

    $photos = (new PhotoFetcher())->fetchForYear($year);
    $html = (new PageCreator())->create($year, $photos);

    $uploader = new Uploader();
    $uploader->uploadOne($year, $html, $s3Bucket);

    $uploader->invalidateCache( [ '/' . $year ] );
}
```

main()

```
return function ($event) {
    $year = $event['year'] ?? date('Y');

    $photos = (new PhotoFetcher())->fetchForYear($year);
    $html = (new PageCreator())->create($year, $photos);

    $uploader = new Uploader();
    $uploader->uploadOne($year, $html, $s3Bucket);

    $uploader->invalidateCache( [ '/' . $year ] );
}
```

main()

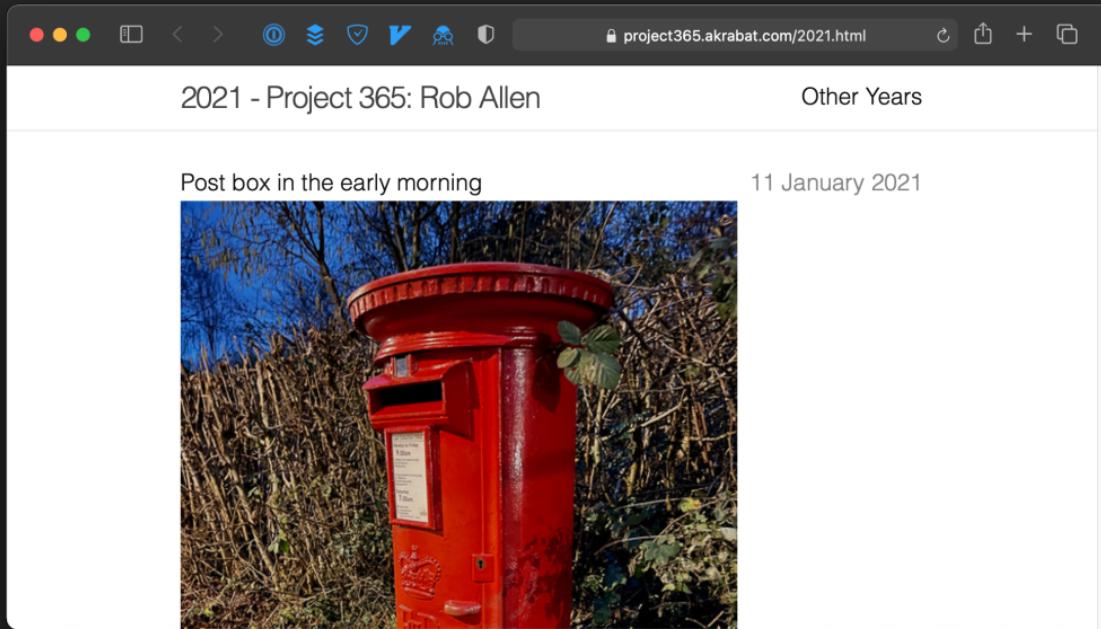
```
return function ($event) {
    $year = $event['year'] ?? date('Y');

    $photos = (new PhotoFetcher())->fetchForYear($year);
    $html = (new PageCreator())->create($year, $photos);

    $uploader = new Uploader();
    $uploader->uploadOne($year, $html, $s3Bucket);

    $uploader->invalidateCache(['/' . $year]);
}
```

The finished website



A photograph of a rocket launching from a launch pad. The rocket is positioned vertically in the center of the frame, with its plume of white smoke and orange fire at the base. The background shows a dark sky transitioning to a warm orange and yellow glow of the setting sun. Several tall metal towers stand behind the launch pad. The foreground is a flat, open landscape.

To sum up

Thank you!

Photo credits

- Assembly line: <https://www.flickr.com/photos/adiram/3886212918>
- Under the hood: <https://www.flickr.com/photos/atomichotlinks/7736849388>
- Pantheon: <https://www.flickr.com/photos/shawnstilwell/4335732627>
- Watch mechanism: <https://www.flickr.com/photos/shinythings/2168994732>
- Holiday snaps: <https://www.flickr.com/photos/kjgarbutt/5358075923>
- Rocket launch: <https://www.flickr.com/photos/gsfc/16495356966>
- Stars: <https://www.flickr.com/photos/gsfc/19125041621>