The Serverless Application

Rob Allen

November 2019


Serverless?

Platform options

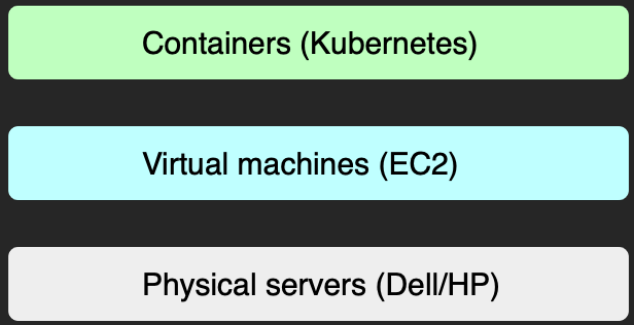
Physical servers (Dell/HP)

Platform options

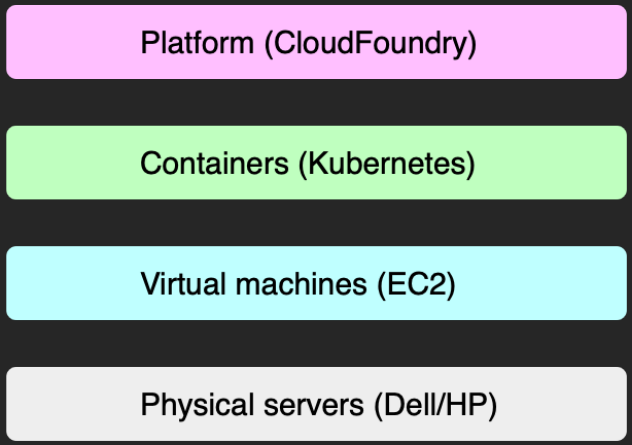
Virtual machines (EC2)

Physical servers (Dell/HP)

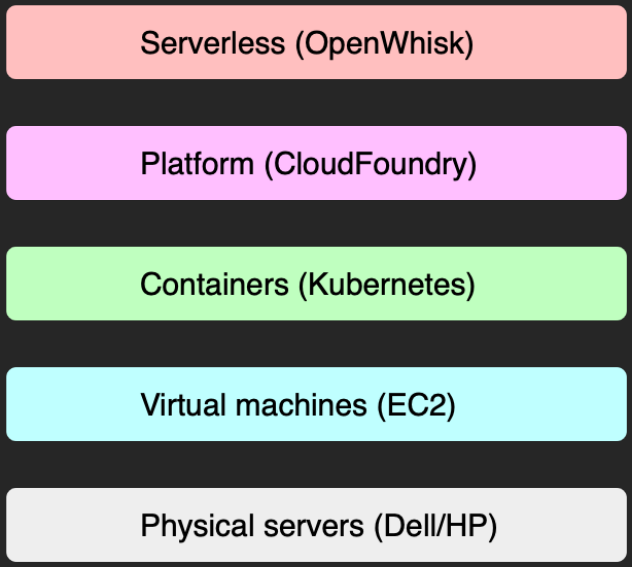
Platform options



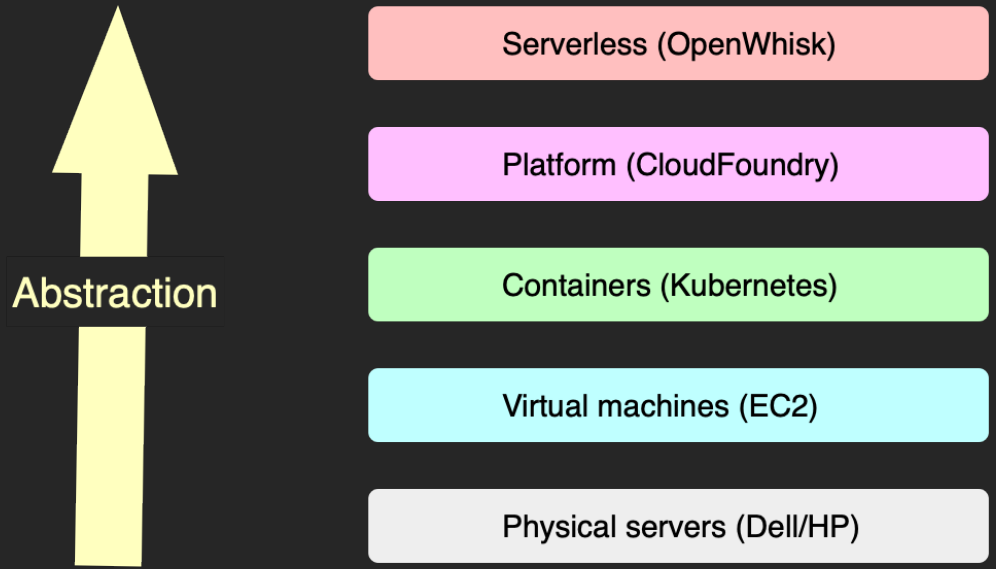
Platform options



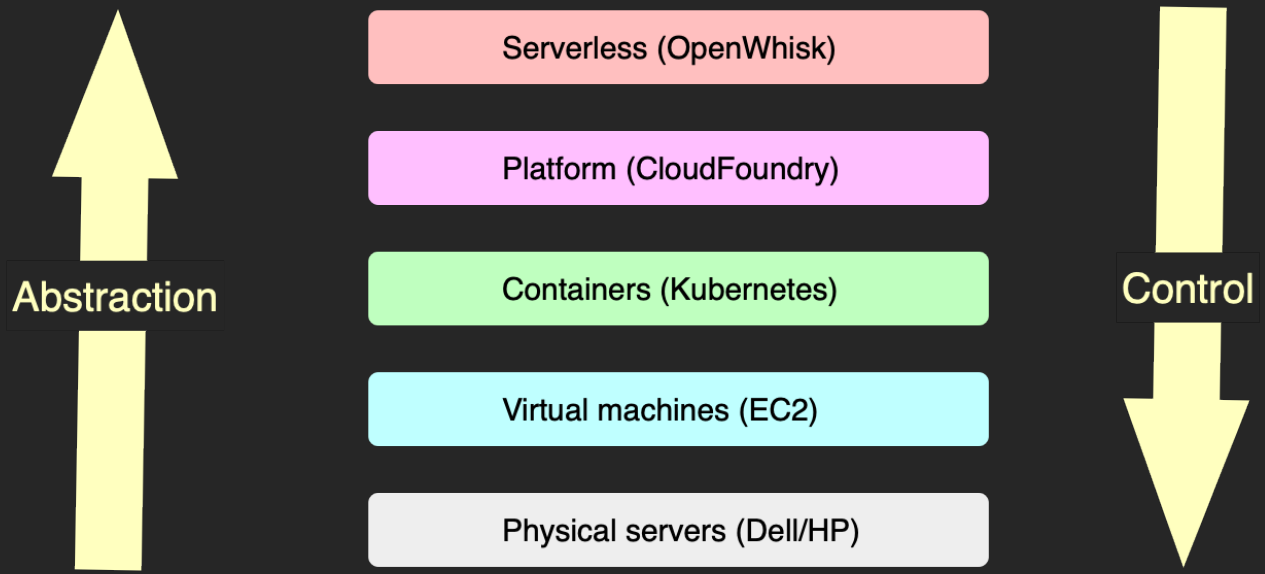
Platform options



Platform options



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?


@akrabat

Not Your PROBLEM!

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

Serverless platforms

Hello World

AWS Lambda:

```
def my_handler(event, context):
    name = event.get("name", "World")
    message = 'Hello {}!'.format(name)

    return {'message': message}
```

Hello World

Apache OpenWhisk:

```
def main(args):
    name = args.get("name", "World")
    message = 'Hello {}!'.format(name)

    return {'message': message}
```

Hello World

Google Cloud Functions

```
def hello_http(request):
    name = request.json.get("name", "World")
    message = 'Hello {}!'.format(name)

    return message
```

Hello World

Azure Cloud Functions

```
import azure.functions as func

def main(req: func.HttpRequest,
         msg: func.Out[func.QueueMessage]) -> str:

    name = req.params.json.get("name", "World")
    message = 'Hello {}!'.format(name)

    msg.set(message)
    return message
```

APACHE OpenWhisk


JS

.NET
Core


php


GO

Java


The anatomy of an action

```
def main(args):
  # Marshall inputs from event parameters
  name = args.get("name", "World")
  # Do the work
  message = 'Hello {}!'.format(name)
  # return result
  return {'body': message}
```

Hello World


```
def main(args):
  # Marshall inputs from event parameters
  name = args.get("name", "World")
  # Do the work
  message = 'Hello {}!'.format(name)
  # return result
  return {'body': message}
```

Hello World

```
def main(args):
  # Marshall inputs from event parameters
  name = args.get("name", "World")
  # Do the work
  message = 'Hello {}!'.format(name)
  # return result
  return {'body': message}
```

Hello World

```
def main(args):
  # Marshall inputs from event parameters
  name = args.get("name", "World")
  # Do the work
  message = 'Hello {}!'.format(name)
  # return result
  return {'body': message}
```

Hello World

```
def main(args):
  # Marshall inputs from event parameters
  name = args.get("name", "World")
  # Do the work
  message = 'Hello {}!'.format(name)
  # return result
  return {'body': message}
```

Deploy to OpenWhisk

```
$ zip -q hello.zip hello.py
```

Deploy to OpenWhisk

```
$ zip -q hello.zip hello.py
$ wsk action update --kind python:3.7 hello hello.zip
ok: updated action hello
```

Run it

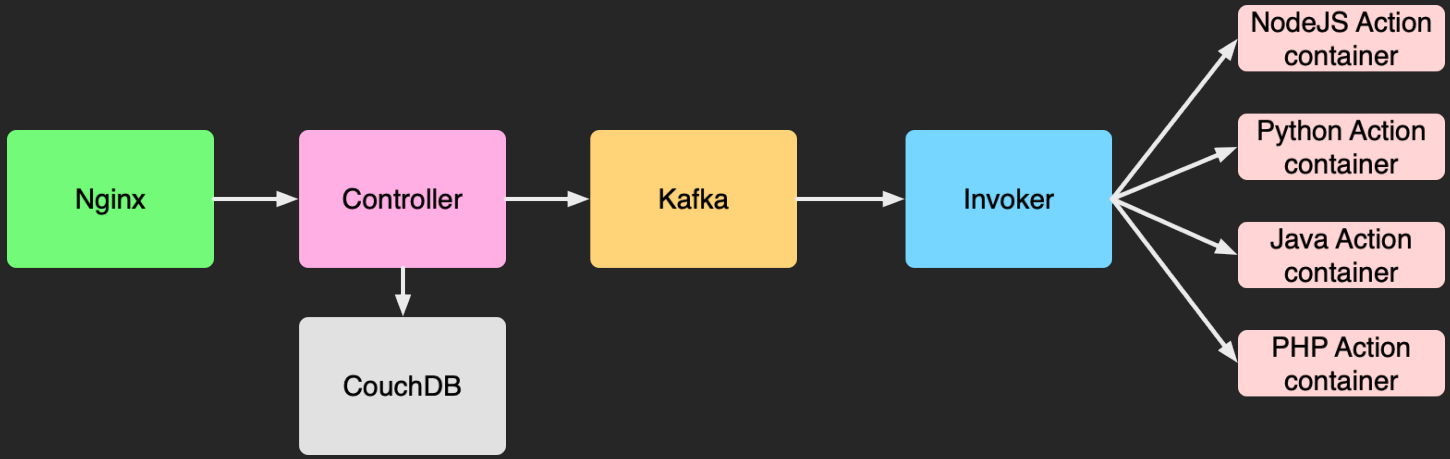
```
$ wsk action invoke hello --result --param name Rob
```

Run it

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

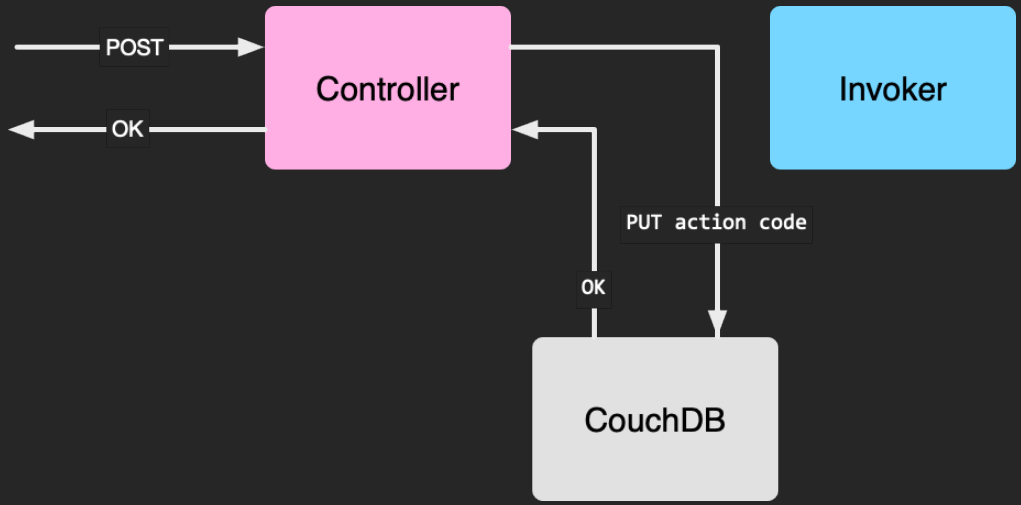

Under the hood

OpenWhisk's architecture



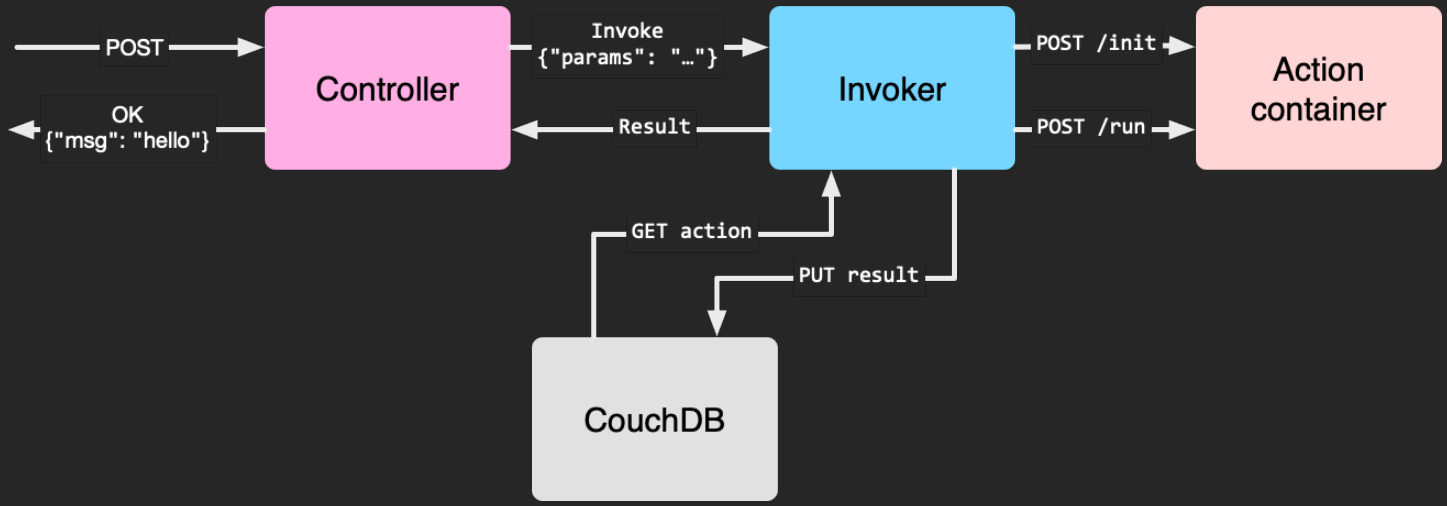
Create an action

```
$ wsk action create hello hello.py
```



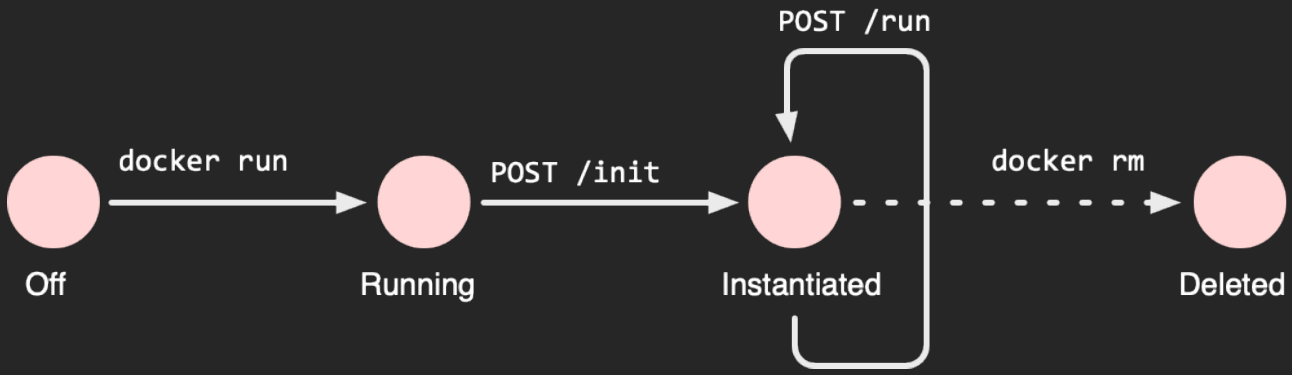
Invoke an action

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



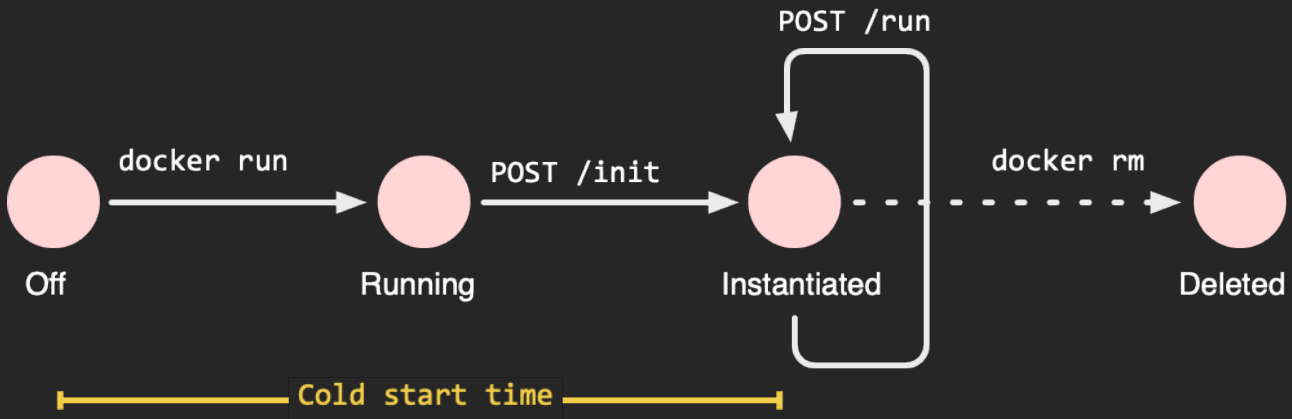
Action container lifecycle

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



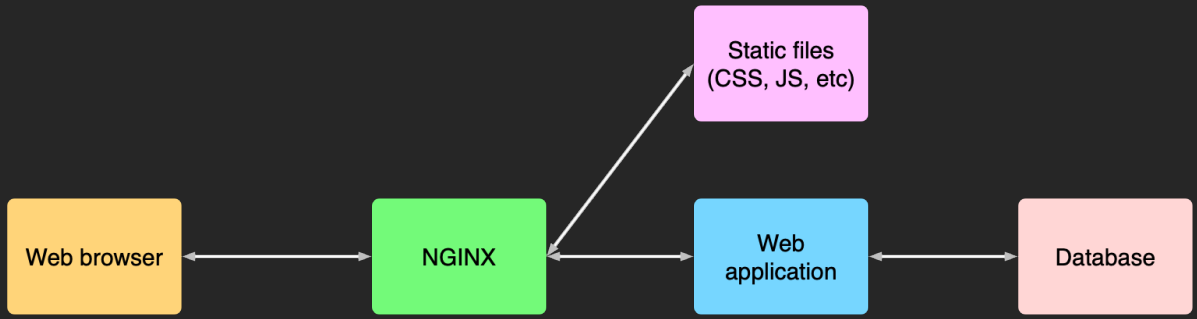
Action container lifecycle

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

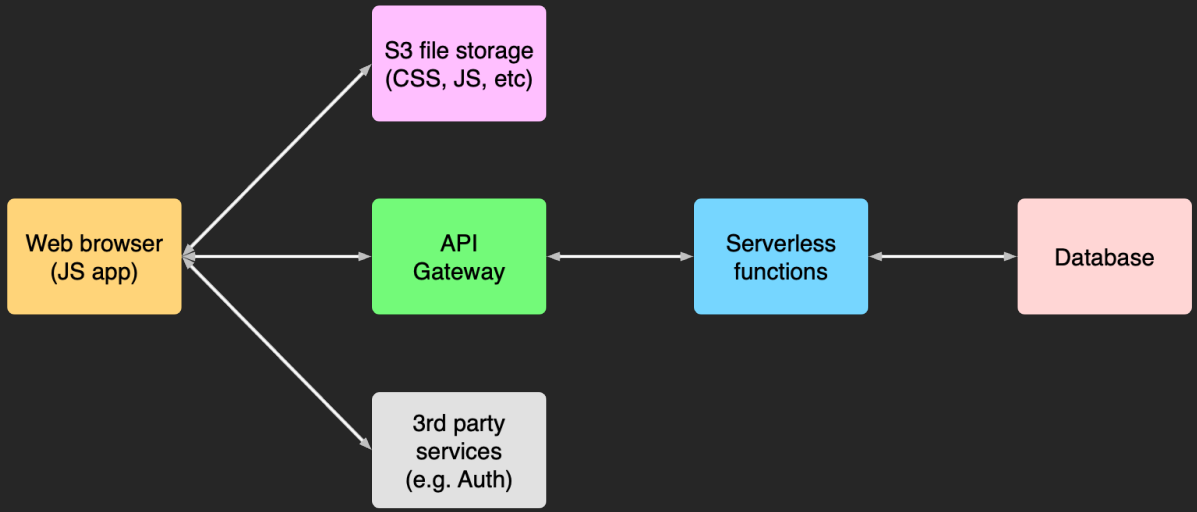


Architecture

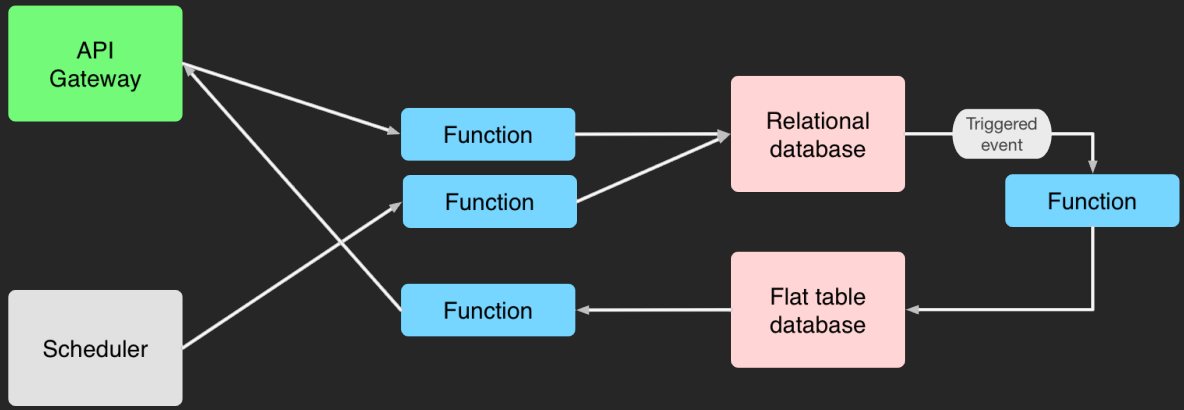
Monolith architecture



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

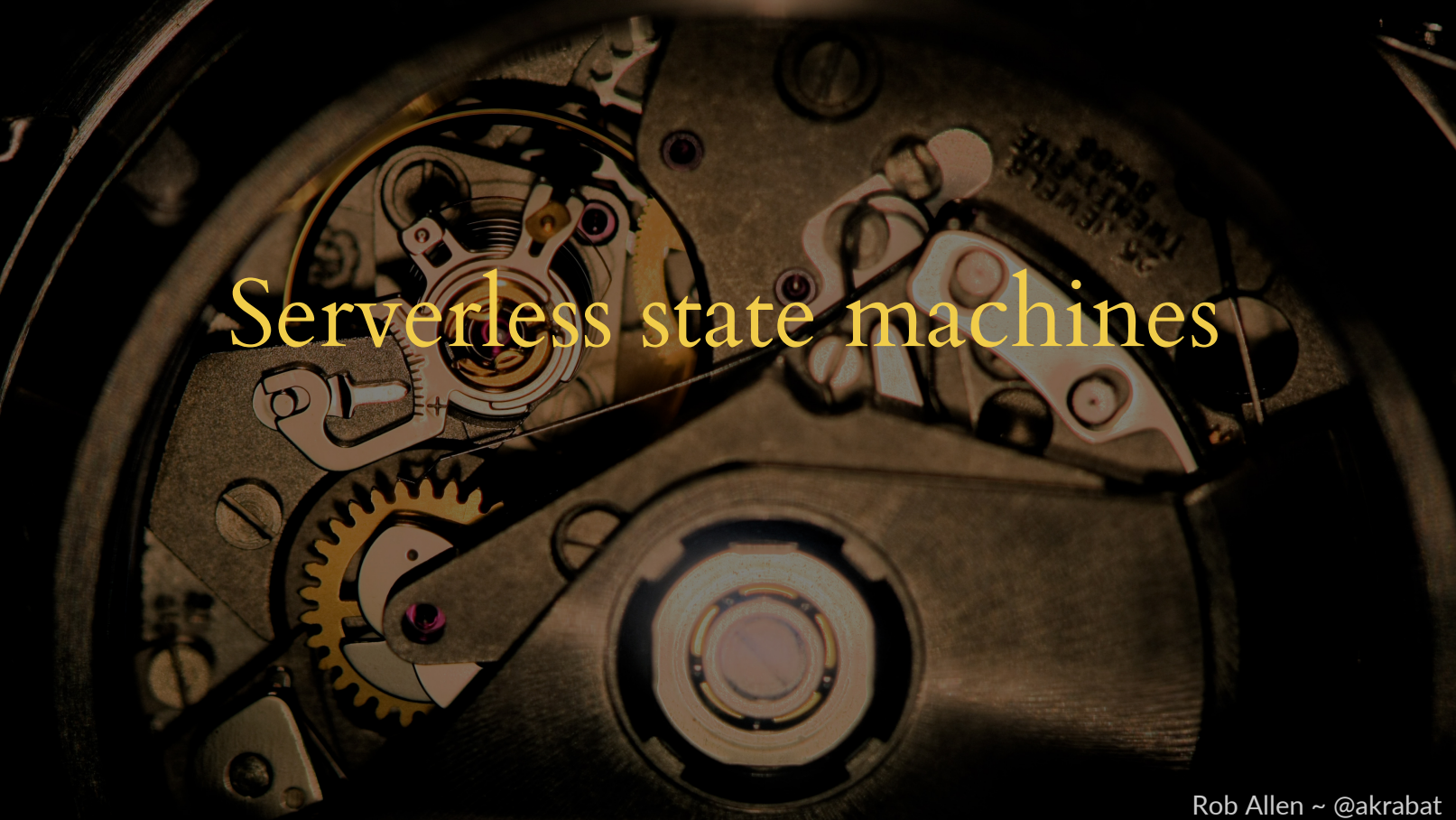
If it's non-trivial, software engineering principles apply!

- Use multiple methods
- Use multiple files

Structure

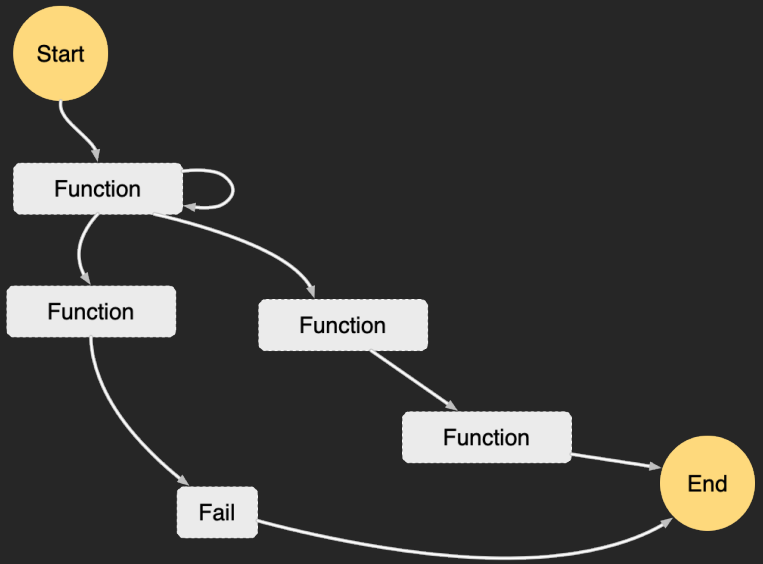
If it's non-trivial, software engineering principles apply!

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



Rob Allen ~ @akrabat

Serverless state machines

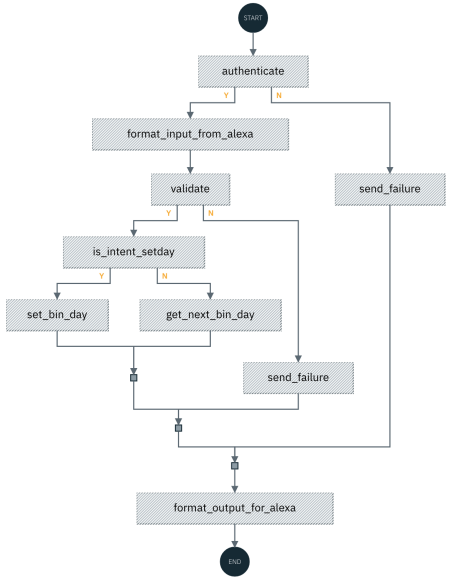


IBM Cloud Shell

HELP

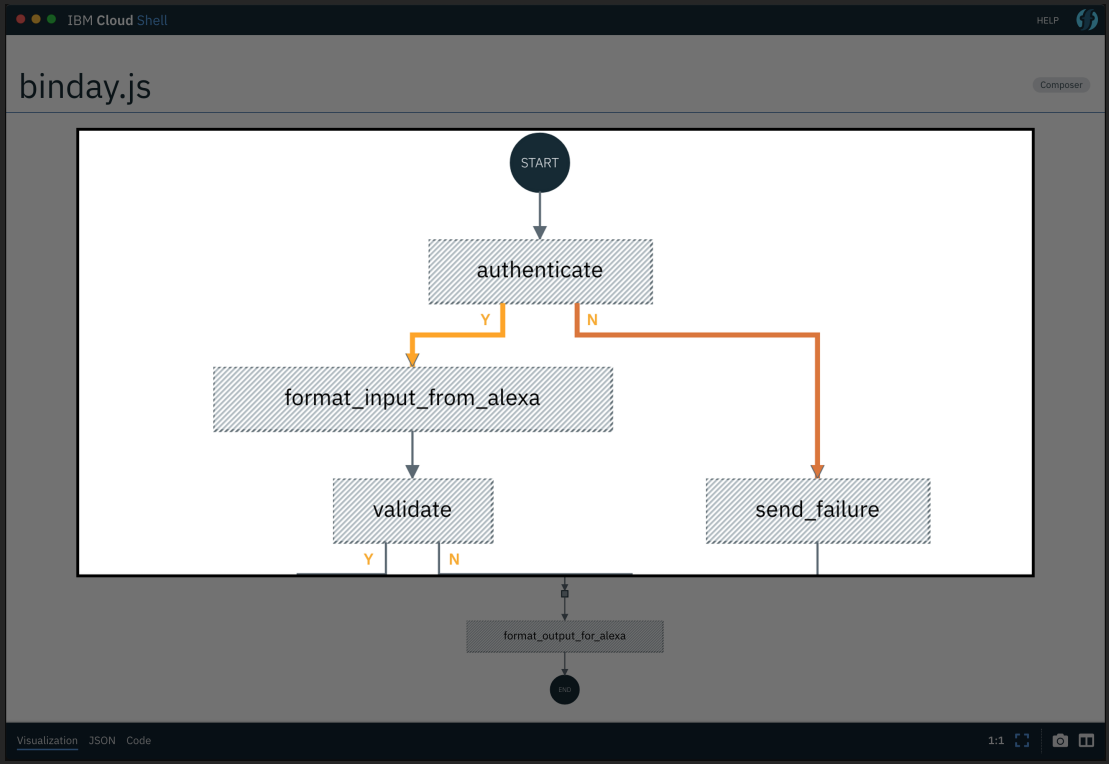
binday.js

Composer



Visualization JSON Code

1:1


IBM Cloud Shell
HELP
binday.js
Composer
START
authenticate
Y
N
format_input_from_alexa
validate
send_failure
Y
N
format_output_for_alexa
END
Visualization
JSON
Code
1:1

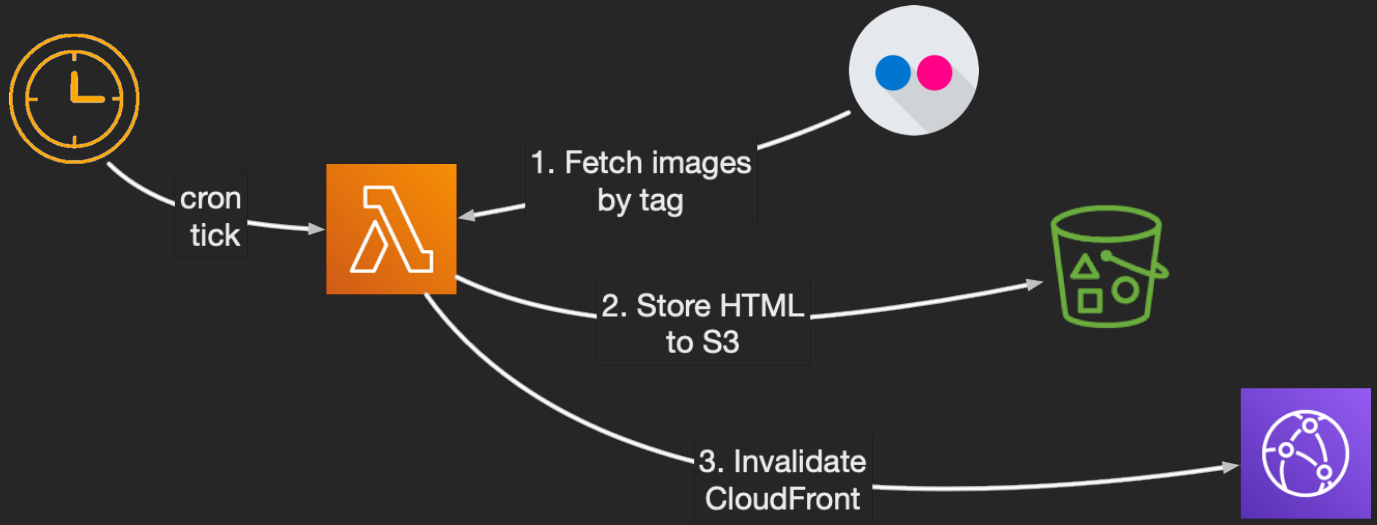
Case study
Project 365 photo website

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

```
functions:
  update:
    handler: src/actions/update.main
    layers:
      - {Ref: PhpLambdaLayer}
    environment:
      FLICKR_API_KEY: ${self:custom.FLICKR_API_KEY}
      FLICKR_USER_ID: ${self:custom.FLICKR_USER_ID}
    events:
      - schedule:
          name: project365-build
          rate: cron(0 */2 * * ? *)
```

Infrastructure as code

```
functions:
  update:
    handler: src/actions/update.main
    environment:
      FLICKR_API_KEY: ${self:custom.FLICKR_API_KEY}
      FLICKR_USER_ID: ${self:custom.FLICKR_USER_ID}
    events:
      - schedule:
          name: project365-build
          rate: cron(0 */2 * * ? *)
```

Infrastructure as code

```
functions:
  update:
    handler: src/actions/update.main
    environment:
      FLICKR_API_KEY: ${self:custom.FLICKR_API_KEY}
      FLICKR_USER_ID: ${self:custom.FLICKR_USER_ID}
    events:
      - schedule:
          name: project365-build
          rate: cron(0 */2 * * ? *)
```

Infrastructure as code

```
functions:
  update:
    handler: src/actions/update.main
    environment:
      FLICKR_API_KEY: ${self:custom.FLICKR_API_KEY}
      FLICKR_USER_ID: ${self:custom.FLICKR_USER_ID}
    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()

```php
function main(array $eventData) : array
{
  $apiKey = getEnvVar('P365_FLICKR_API_KEY');
  $userId = getEnvVar('P365_FLICKR_USER_ID');
  $year = $eventData['year'] ?? date('Y');

  $pageCreator = new PhotoPageCreator($apiKey);
  $html = $pageCreator->update($year, $userId);
  $uploader = new Uploader($cloudFrontId);
  $uploader->uploadOne($filename, $html, $s3Bucket);
  $uploader->invalidate(['/'.$filename]);
}
```

main()

```
function main(array $eventData) : array
{
  $apiKey = getEnvVar('P365_FLICKR_API_KEY');
  $userId = getEnvVar('P365_FLICKR_USER_ID');
  $year = $eventData['year'] ?? date('Y');

  $pageCreator = new PhotoPageCreator($apiKey);
  $html = $pageCreator->update($year, $userId);
  $uploader = new Uploader($cloudFrontId);
  $uploader->uploadOne($filename, $html, $s3Bucket);
  $uploader->invalidate(['/'.$filename]);
}
```

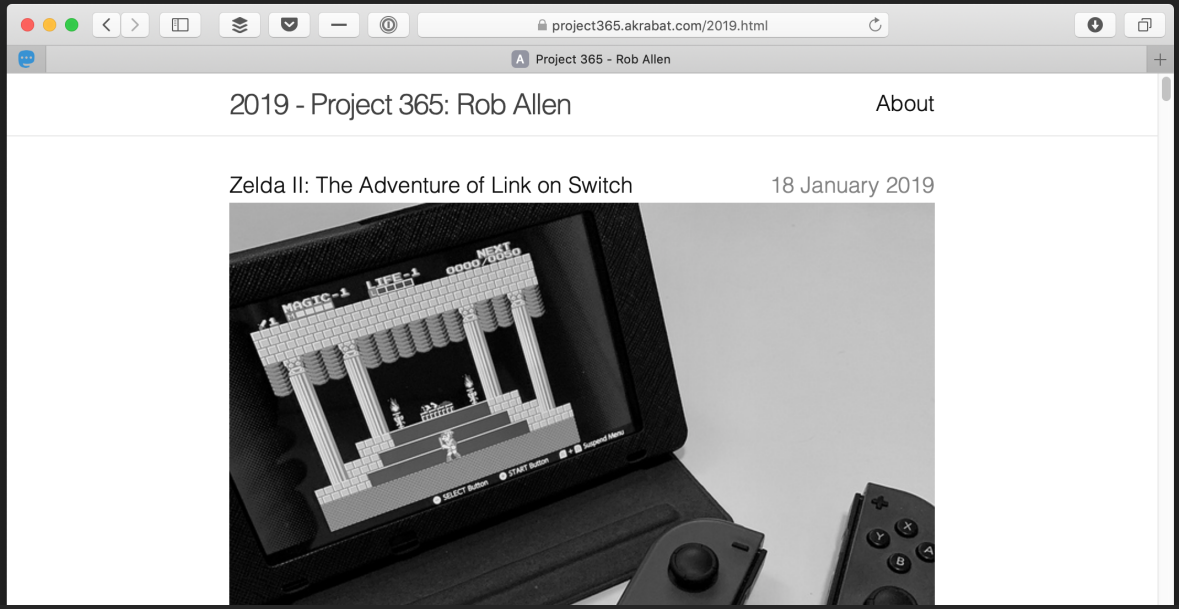
main()

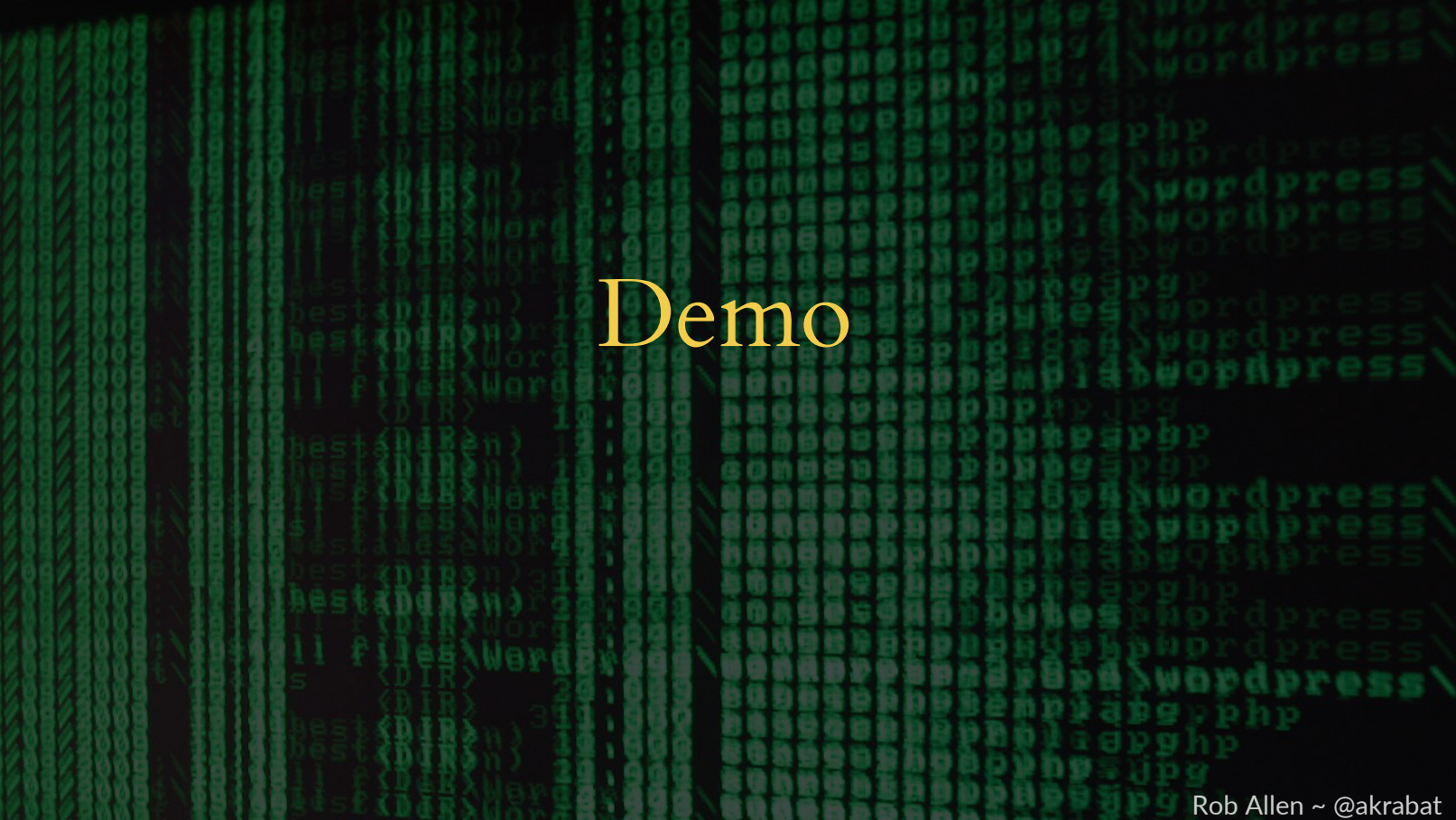
```
function main(array $eventData) : array
{
  $apiKey = getEnvVar('P365_FLICKR_API_KEY');
  $userId = getEnvVar('P365_FLICKR_USER_ID');
  $year = $eventData['year'] ?? date('Y');

  $pageCreator = new PhotoPageCreator($apiKey);
  $html = $pageCreator->update($year, $userId);
  $uploader = new Uploader($cloudFrontId);
  $uploader->uploadOne($filename, $html, $s3Bucket);
  $uploader->invalidate(['/'.$filename]);
}
```

main()

```
function main(array $eventData) : array
{
  $apiKey = getEnvVar('P365_FLICKR_API_KEY');
  $userId = getEnvVar('P365_FLICKR_USER_ID');
  $year = $eventData['year'] ?? date('Y');

  $pageCreator = new PhotoPageCreator($apiKey);
  $html = $pageCreator->update($year, $userId);
  $uploader = new Uploader($cloudFrontId);
  $uploader->uploadOne($filename, $html, $s3Bucket);
  $uploader->invalidate(['/'.$filename]);
}
```

The finished website




Demo


To sum up

Resources

- https://akrabat.com
- https://www.martinfowler.com/articles/serverless.html

- https://github.com/akrabat/ow-php-todo-backend
- https://github.com/akrabat/project365-photos-website

- http://www.openwhisk.org
- https://aws.amazon.com/lambda/
- https://bref.sh


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
- Computer code: https://www.flickr.com/photos/n3wjack/3856456237
- Rocket launch: https://www.flickr.com/photos/gsfc/16495356966
- Stars: https://www.flickr.com/photos/gsfc/19125041621