Serverless PHP Applications

Rob Allen

Nineteen Feet

JUNE 6
A Serverless solution is one that costs you nothing to run if nobody is using it

... excluding data storage costs.

Paul D. Johnston
as a Service

Storage as a Service
Database as a Service
Cache as a Service
Auth as a Service
Search as a Service
Function as a Service
Function as a Service

- Your code
- Deployed to the cloud
- Runs when needed
- Scaled automatically
- Pay only for execution

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)
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
It's about value

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.

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.

3) We are relearning what was known in the time-share days. Pricing needs to be based on something customers value, not infa. items like VMs
When should you use serverless?

• Responding to web hooks
When should you use serverless?

- Responding to web hooks
- PWA/Static site contact form, et al.
When should you use serverless?

- Responding to web hooks
- PWA/Static site contact form, et al.
- Additional features without extending current platform
When should you use serverless?

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

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

Rob Allen ~ @akrabat
Serverless platforms

- Azure
- Google Cloud Platform
- aws
- IBM Cloud
- fn
- Openfaas
Serverless platforms with PHP support

- Azure
- AWS
- IBM Cloud
- fn
- Google Cloud Platform

Rob Allen ~ @akrabat
Concepts

Sources → Event Trigger → Rules → Actions (code) → Results

HTTP requests → CLI

Rob Allen ~ @akrabat
<?php

function main(array $args) : array
{
    $name = $args["name"] ?? "World";
    return [ "msg" => "Hello $name" ];
}

Hello world in PHP

```php
<?php
function main(array $args): array {
    $name = $args["name"] ?? "World";
    return [ "msg" => "Hello $name" ];
}
```
Upload your action

$ wsk action update hello hello.php
ok: updated action hello
Run your action

$ wsk action invoke hello --result
{
  "msg": "Hello World"
}
Segue: How did it do this?
OpenWhisk's architecture

- Nginx
- Controller
- Consul
- CouchDB
- Kafka
- Invoker
- Action container
- Action container
- Action container
- Action container

Rob Allen ~ @akrabat
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

Rob Allen ~ @akrabat
Action container lifecycle

- Hosts the user-written code
- Controlled via two end points: /init & /run
End Segue
Turn it into an API

Add the --web flag:

$ wsk action update hello hello.php --web true
Turn it into an API

Add the --web flag:

```bash
$ wsk action update hello hello.php --web true
$ curl https://openwhisk.ng.bluemix.net/api/v1/web/19FT_demo/default/hello.json
{
    "msg": "Hello World"
}
```
API Gateway

When you want to do more with HTTP endpoints

- Route endpoint methods to actions
- Custom domains
- Rate limiting
- Security (API keys, OAuth, CORS)
- Analytics
$ wsk api create /demo /hello GET hello
ok: created API /demo/hello GET for action /_/hello
API Gateway

```
$ curl https://ow.akrabat.com/demo/hello?name=Rob
{
  "message": "Hello Rob!"
}
```
A Serverless API

Todo-Backend

a shared example to showcase backend tech stacks

The Todo-Backend project helps showcase and compare different language and framework combinations for building web services. This website defines a simple web API in the form of a todo list and users can create their own identical APIs using various tech stacks. This website then provides a spec runner to verify that the user implementation is identical to the base implementation.

The Todo-Backend project was inspired by the TodoMVC project, and some code (specifically the todo client app) was borrowed directly from TodoMVC.

featuring HTTP APIs built with:

- Node.js
- Express
- MongoDB
- CouchDB
- C#
Todo-Backend

An OpenWhisk PHP implementation of a to-do list API

Rob Allen ~ @akratab
Serverless Framework

Deployment tooling for serverless applications

`serverless.yml`:

```yaml
service: ow-todo-backend

provider:
  name: openwhisk
  runtime: php

plugins:
  - serverless-openwhisk
```

Rob Allen ~ @akrabat
Configure action

functions:
  edit-todo:
    handler: "src/actions/editTodo.main"
    name: "todo-backend/edit-todo"
Configure action

functions:
edit-todo:
    handler: "src/actions/editTodo.main"
    name: "todo-backend/edit-todo"
Configure action

functions:
  edit-todo:
    handler: "src/actions/editTodo.main"
    name: "todo-backend/edit-todo"
Configure API Gateway

functions:
  edit-todo:
    handler: "src/actions/editTodo.main"
    name: "todo-backend/edit-todo"
  events:
    - http:
      path: /todos/{id}
      method: patch
Configure API Gateway

functions:
  edit-todo:
    handler: "src/actions/editTodo.main"
    name: "todo-backend/edit-todo"
    events:
      - http:
          path: /todos/{id}
          method: patch
Project files

- src/
- vendor/
  - composer.json
  - composer.lock
  - serverless.yml
Project files

```
.
├── src/
│   ├── Todo/
│       ├── Todo.php
│       └── TodoMapper.php
│   ├── TodoTransformer.php
│   └── actions/
│       ├── addTodo.php
│       ├── deleteTodo.php
│       ├── editTodo.php
│       └── listTodos.php
└── AppContainer.php
```
function main(array $args): array
{
    try {
        $parts = explode("/", $args['__ow_path']);
        $id = (int)array_pop($parts);
        $data = json_decode(base64_decode($args['__ow_body']), true);
        if (!is_array($data)) {
            throw new InvalidArgumentException('Missing body', 400);
        }
        $container = new AppContainer($args);
        $mapper = $container[TodoMapper::class];
        $todo = $mapper->loadById($id);
        $mapper->update($todo, $data);
        $resource = new Item($todo, $mapper, 'todos');
        $fractal = $container[Manager::class];
        return [
            'statusCode' => 200,
            'body' => $fractal->createData($resource)->toArray(),
        ];
    } catch (Throwable $e) {
        var_dump((string)$e);
        $code = $e->getCode() < 400 ? $e->getCode() : 500;
        return [
            'statusCode' => $code,
            'body' => ['error' => $e->getMessage()],
        ];
    }
}
Function main(array $args) : array
{
    try {
        // do stuff
    }
    catch (Throwable $e) {
        var_dump((string)$e);
        $code = $e->getCode() < 400 ? $e->getCode() : 500;
        return [
            'statusCode' => $code,
            'body' => ['error' => $e->getMessage()]
        ];
    }
}
function main(array $args) : array
{
    try {
        // do stuff
    } catch (Throwable $e) {
        var_dump((string)$e);
        $code = $e->getCode() < 400 ? $e->getCode() : 500;
        return [
            'statusCode' => $code,
            'body' => ['error' => $e->getMessage()]
        ];
    }
}
function main(array $args) : array
{
    try {
        // do stuff
    }
    catch (Throwable $e) {
        var_dump((string)$e);
        $code = $e->getCode() < 400 ? $e->getCode() : 500;
        return [
            'statusCode' => $code,
            'body' => ['error' => $e->getMessage()]
        ];
    }
}
function main(array $args) : array
{
    try {
        // do stuff
    }
    catch (Throwable $e) {
        var_dump((string)$e);
        $code = $e->getCode() < 400 ? $e->getCode() : 500;
        return [
            'statusCode' => $code,
            'body' => ['error' => $e->getMessage()]];
    }
}
$parts = explode("/", $args['__ow_path']);
$id = (int)array_pop($parts);
editTodo.php: Grab input

```php
$parts = explode("/", $args['__ow_path']);
$id = (int)array_pop($parts);

$body = base64_decode($args['__ow_body']);
$data = json_decode($body, true);
if (!is_array($data)) {
    throw new Exception('Missing body', 400);
}
```
$parts = explode("/", $args['__ow_path']);
$id = (int)array_pop($parts);

$body = base64_decode($args['__ow_body']);
$data = json_decode($body, true);
if (!is_array($data)) {
    throw new Exception('Missing body', 400);
}
editTodo.php: Do the work

    $container = new AppContainer($args);
    $mapper = $container[TodoMapper::class];

    $todo = $mapper->loadById($id);
    $mapper->update($todo, $data);
$container = new AppContainer($args);
.mapper = $container[TodoMapper::class];

todo = $mapper->loadById($id);
.mapper->update($todo, $data);
$container = new AppContainer($args);
$mapper = $container[TodoMapper::class];

$todo = $mapper->loadById($id);
$mapper->update($todo, $data);
editTodo.php: Present results

```php
$transformer = $container[TodoTransformer::class];
$resource = new Item($todo, $transformer, 'todos');
$fractal = $container[Manager::class];

$output = $fractal->createData($resource);

return [
    'statusCode' => 200,
    'body' => $output->toArray(),
];
```
$transformer = $container[TodoTransformer::class];

$resource = new Item($todo, $transformer, 'todos');

$fractal = $container[Manager::class];

$output = $fractal->createData($resource);

return [
    'statusCode' => 200,
    'body' => $output->toArray(),
];
editTodo.php: Present results

$transformer = $container[TodoTransformer::class];
$resource = new Item($todo, $transformer, 'todos');
$fractal = $container[Manager::class];

$output = $fractal->createData($resource);

return [
    'statusCode' => 200,
    'body' => $output->toArray(),
];
$ serverless deploy
Serverless: Packaging service...
Serverless: Compiling Functions...
Serverless: Compiling Packages...
Serverless: Compiling API Gateway definitions...
Serverless: Compiling Rules...
Serverless: Compiling Triggers & Feeds...
Serverless: Compiling Service Bindings...
Serverless: Deploying Packages...
Serverless: Deploying Functions...
Serverless: Deploying API Gateway definitions...
[...]
Rob Allen ~ @akrabat
A working API!

These tests are targeting:
https://service.eu.apiconnect.ibmcloud.com/gws/apigateway/api/2b6669604cd65f6fa459d90a6fphptodo-backend/todos
choose a different server to target

todo-backend API residing at
https://service.eu.apiconnect.ibmcloud.com/gws/apigateway/api/2b6669604cd65f6fa459d90a6fphptodo-backend/todos
the pre-requsites
 ✓ the api root responds to a GET (i.e. the server is up and accessible, CORS headers are set up)
 ✓ the api root responds to a POST with the todo which was posted to it
 ✓ the api root responds successfully to a DELETE
 ✓ after a DELETE the api root responds to a GET with a JSON representation of an empty array
storing new todos by posting to the root url
 ✓ adds a new todo to the list of todos at the root url
 ✓ sets up a new todo as initially not completed
 ✓ each new todo has a url
 ✓ each new todo has a url, which returns a todo
working with an existing todo
 ✓ can navigate from a list of todos to an individual todo via urls
 ✓ can change the todo title by DELETEing to the todo's url

Rob Allen ~ @akrabat
AWS Lambda with PHP

Only *sensibly* possible since November 2018 with the introduction of *layers*
AWS Lambda with PHP

Only *sensibly* possible since November 2018 with the introduction of *layers*

Process:

1. Create a layer containing:
   1. the PHP executable
   2. a bootstrap script
2. Write the PHP function!
bref running PHP made simple

Everything you need to easily deploy and run serverless PHP applications.
Bref

- Maintained PHP runtimes for AWS Lambda
- Deployment via Serverless Framework
- Great documentation!
<?php declare(strict_types=1);

require __DIR__ . '/vendor/autoload.php';

return function($event) {
    return 'Hello ' . ($event['name'] ?? 'world');
};
service: helloapp
provider:
  name: aws
  runtime: provided
functions:
  hello:
    handler: index.php
    layers:
      - ${bref:layer.php-73}
serverless.yml

service: helloapp
provider:
  name: aws
  runtime: provided

functions:
  hello:
    handler: index.php
    layers:
      - ${bref:layer.php-73}
serverless.yml

service: helloapp
provider:
  name: aws
  runtime: provided

functions:
  hello:
    handler: index.php
    layers:
      - ${bref:layer.php-73}
$ serverless deploy
Serverless: Packaging service...
...
Serverless: Stack update finished...

Service Information
service: helloapp
stage: dev
stack: helloapp-dev
functions:
  hello: helloapp-dev-hello
Run

$ serverless invoke -f hello
"Hello world"
Run

$ serverless invoke -f hello
"Hello world"

$ serverless invoke -f hello -d '{"name": "Rob"}"
"Hello Rob"
Run locally in Docker

$ serverless invoke local --docker -f hello
Serverless: Building Docker image...
...
REPORT RequestId: 6a653a94-ee51-1f3e-65c7-1f1954842f29
    Init Duration: 265.93 ms Duration: 145.37 ms
    Billed Duration: 200 ms Memory Size: 1024 MB
    Max Memory Used: 27 MB

"Hello world"

Xdebug also works!

Rob Allen ~ @akrabat
Add AWS API Gateway

serverless.yml:

functions:
  hello:
    handler: index.php

  events:
  - http: "GET /hello"
  - http: "GET /hi/{name}"
Return a PSR-15 RequestHandler

class HelloHandler implements RequestHandlerInterface
{
    public function handle(ServerRequest $request): Response
    {
        $name = ($request->getQueryParams() ['name'] ?? 'world');
        $body = 'Hello ' . $name;

        return new Response(200,
            ['Content-Type' => 'text/plain'],
            $body);
    }
}
Deploy

$ serverless deploy
Serverless: Packaging service...
...

Service Information
service: helloapp
stage: dev
stack: helloapp-dev
endpoints:
  GET - https://l1v6cz13zb.execute-api.eu-west-2.amazonaws.com/dev/hello
Test

```
$ curl -i https://l1v6cz...naws.com/dev/hello?name=Rob
HTTP/2 200
content-type: text/plain
content-length: 9

Hello Rob
```
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

1. Fetch images by tag
2. Store HTML to S3
3. Invalidate CloudFront

cron tick

Rob Allen ~ @akrabat
Serverless configuration

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

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

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

Rob Allen ~ @akrabat
function main(array $eventData) : array
{
    $year = $eventData['year'] ?? date('Y');

    $pageCreator = new PhotoPageCreator();
    $html = $pageCreator->update($year);
    $uploader = new Uploader($cloudFrontId);
    $uploader->uploadOne($year, $html, $s3Bucket);
    $uploader->invalidateCache(['/'. $year]);
}
function main(array $eventData) : array
{
    $year = $eventData['year'] ?? date('Y');

    $pageCreator = new PhotoPageCreator();
    $html = $pageCreator->update($year);
    $uploader = new Uploader($cloudFrontId);
    $uploader->uploadOne($year, $html, $s3Bucket);
    $uploader->invalidateCache(['/'. $year]);
}
main()

function main(array $eventData) : array
{
    $year = $eventData['year'] ?? date('Y');

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

Rob Allen ~ @akrabat
function main(array $eventData) : array
{
	$year = $eventData['year'] ?? date('Y');

$pageCreator = new PhotoPageCreator();
$html = $pageCreator->update($year);
$ uploader = new Uploader($cloudFrontId);
$uploader->uploadOne($year, $html, $s3Bucket);
$uploader->invalidateCache(['/'.$year]);
}
function main(array $eventData) : array
{
    $year = $eventData['year'] ?? date('Y');

    $pageCreator = new PhotoPageCreator();
    $html = $pageCreator->update($year);
    $uploader = new Uploader($cloudFrontId);
    $uploader->uploadOne($year, $html, $s3Bucket);
    $uploader->invalidateCache(['/'.$year]);
}
Fetch photos from Flickr

```php
$url = '?' . http_build_query([  'api_key' => $this->flickrApiKey,  'user_id' => $flickrUserId,  'extras' => 'url_z, date_taken, owner_name',  'method' => 'flickr.photos.search',  'tags' => $year,]);

$response = $this->client->get($url);
$data = json_decode($response->getBody(), true);
return $data['photos'];
```
Fetch photos from Flickr

$url = '?'. http_build_query([ 'api_key' => $this->flickrApiKey, 'user_id' => $flickrUserId, 'extras' => 'url_z, date_taken, owner_name', 'method' => 'flickr.photos.search', 'tags' => $year, ]); 

$response = $this->client->get($url);
$data = json_decode($response->getBody(), true);
return $data['photos'];
Fetch photos from Flickr

$url = '?' . http_build_query([  'api_key' => $this->flickrApiKey,  'user_id' => $flickrUserId,  'extras' => 'url_z, date_taken, owner_name',  'method' => 'flickr.photos.search',  'tags' => $year,  ]);  

$response = $this->client->get($url);
$data = json_decode($response->getBody(), true);
return $data['photos'];
Fetch photos from Flickr

$url = '?' . http_build_query([  
    'api_key' => $this->flickrApiKey,  
    'user_id' => $flickrUserId,  
    'extras' => 'url_z, date_taken, owner_name',  
    'method' => 'flickr.photos.search',  
    'tags' => $year,  
  ]);  
$response = $this->client->get($url);  
$data = json_decode($response->getBody(), true);  
return $data['photos'];
Upload to S3

$s3 = new S3Client([  
    'version' => 'latest',  
    'region' => getenv('AWS_DEFAULT_REGION')  
]);

$s3->putObject([  
    'Bucket' => $bucketName,  
    'ACL' => 'public-read',  
    'Key' => '/$year.html',  
    'Body' => $data,  
    'ContentType' => 'text/html',  
]);
Upload to S3

$s3 = new S3Client([  
    'version' => 'latest',  
    'region'  => getenv('AWS_DEFAULT_REGION')  
]);

$s3->putObject([  
    'Bucket' => $bucketName,  
    'ACL'    => 'public-read',  
    'Key'    => '/$year.html',  
    'Body'   => $data,  
    'ContentType' => 'text/html',  
]);
Upload to S3

$s3 = new S3Client([  
    'version' => 'latest',  
    'region'  => getenv('AWS_DEFAULT_REGION')
]);

$s3->putObject([  
    'Bucket' => $bucketName,  
    'ACL'    => 'public-read',  
    'Key'    => '/$year.html',  
    'Body'   => $data,  
    'ContentType' => 'text/html',
  ]);
Invalidate CloudFront

```php
$cft = new CloudFrontClient([ .. ]);,

$result = $cft->createInvalidation(
    ['DistributionId' => $cloudFrontId,
     'InvalidationBatch' => [
         'CallerReference' => date('YmdHis'),
         'Paths' => [
             ['Items' => ["/$year.html"],
             'Quantity' => 1,
             ],
             ],
          ],
    ]);
```
Invalidate CloudFront

```php
$cft = new CloudFrontClient([ .. ]);

$result = $cft->createInvalidation(
    'DistributionId' => $cloudFrontId,
    'InvalidationBatch' => [
        'CallerReference' => date('YmdHis'),
        'Paths' => [
            'Items' => ['/$_year.html'],
            'Quantity' => 1,
        ],
    ],
);
```
$cft = new CloudFrontClient([..]);

$result = $cft->createInvalidation(['DistributionId' => $cloudFrontId,
   'InvalidationBatch' => [
      'CallerReference' => date('YmdHis'),
      'Paths' => [
         'Items' => ['/year.html'],
         'Quantity' => 1,
      ],
   ],
]);
Invalidate CloudFront

```php
$cft = new CloudFrontClient([..]);

$result = $cft->createInvalidation(['DistributionId' => $cloudFrontId,
  'InvalidationBatch' => [
    'CallerReference' => date('YmdHis'),
    'Paths' => [
      'Items' => ["/$year.html"],
      'Quantity' => 1,
    ],
  ],
]);
```
<table>
<thead>
<tr>
<th>Years</th>
<th>5 March 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link Amiibo</td>
<td><img src="image" alt="Link Amiibo" /></td>
</tr>
</tbody>
</table>
To sum up

Rob Allen ~ @akrabat
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!

Rob Allen - http://akrabat.com - @akrabat