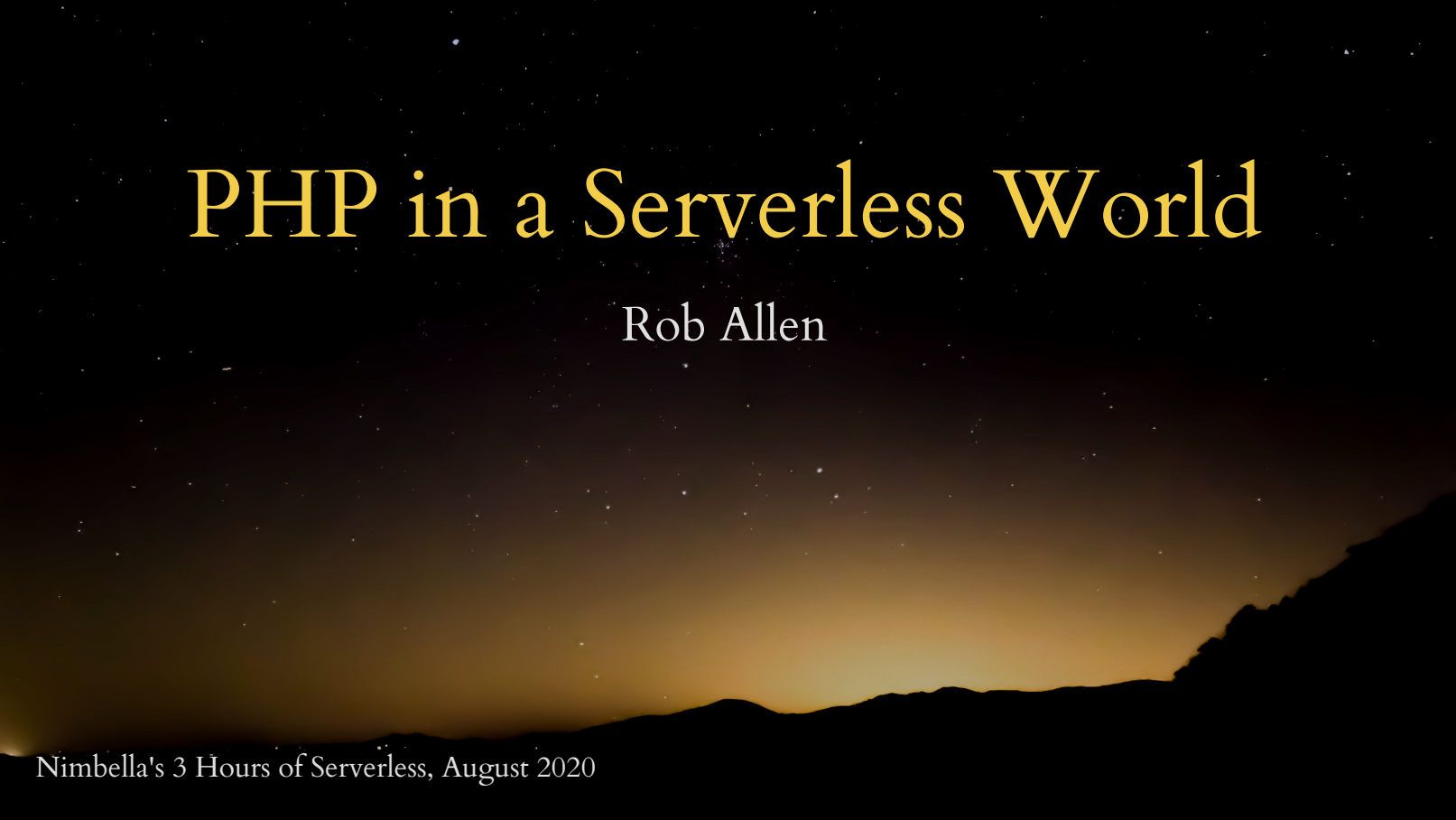

PHP in a Serverless World
Rob Allen
Nimbella's 3 Hours of Serverless, August 2020

John Arundel
@bitfield

Lessons of serverless, no. 1: You will have no choice but to use JavaScript

6 things I've learned in my first 6 months using serverless
The serverless world is pretty awesome once you find the right tools — and burn the middle layer to the ground
read.acloud.guru

8:14 PM · May 13, 2018 · Buffer


JS

.NET
Core


php



GO

Java


JS
php
GO
java


Why?
Why?
Why?
Why?
Why?

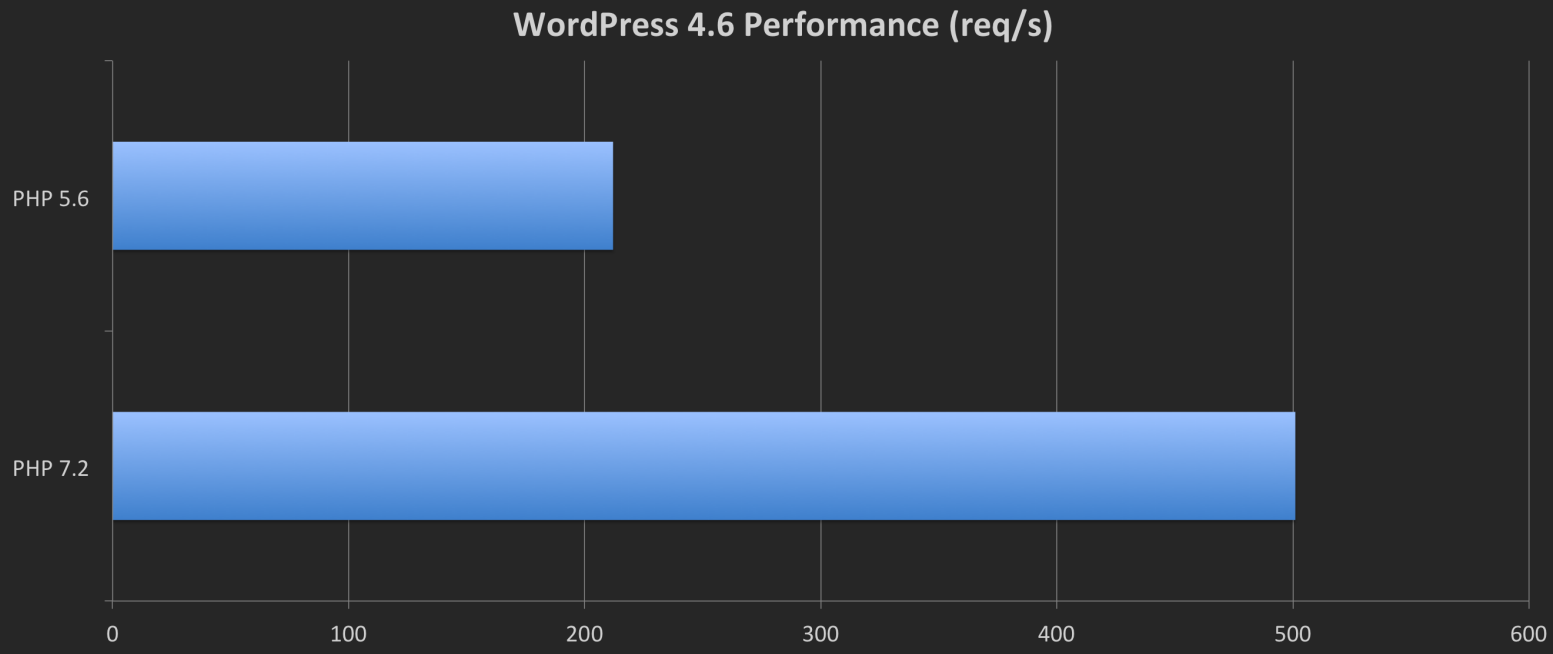
Stateless

Synchronous

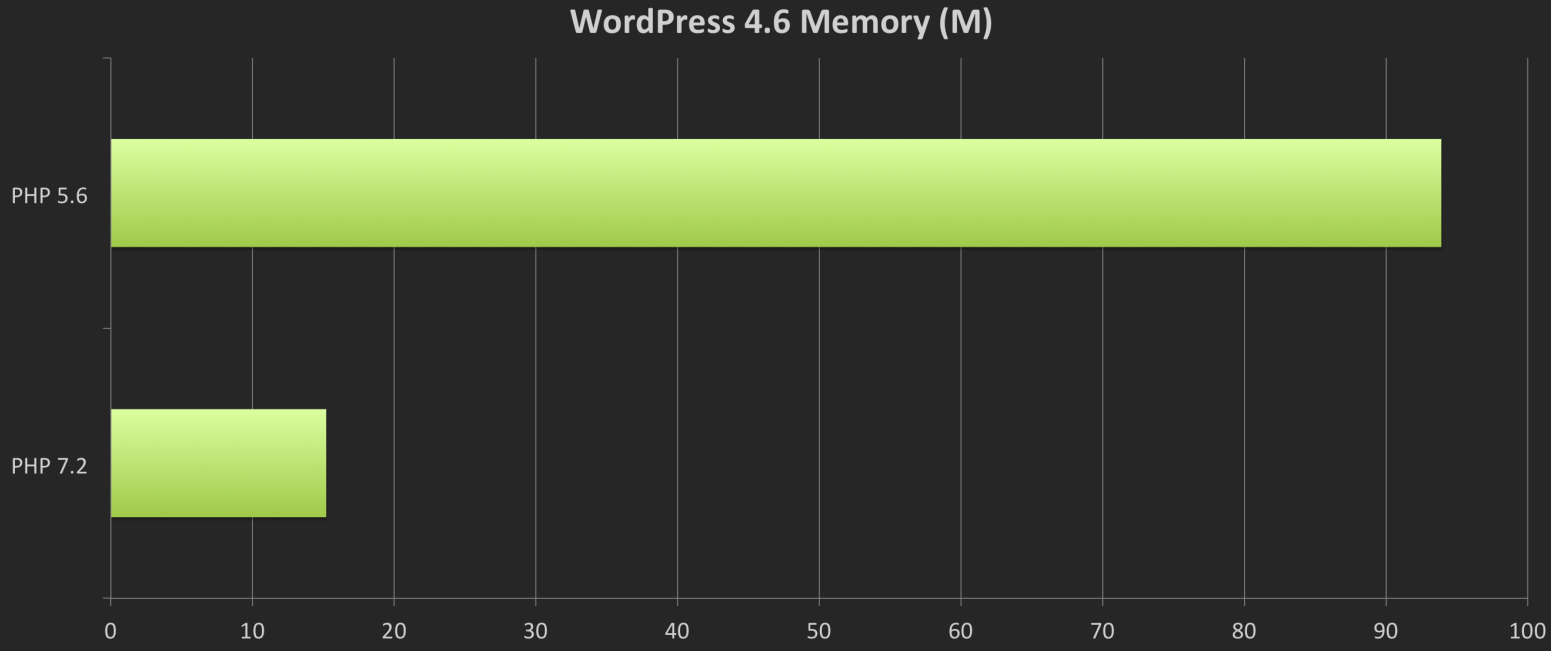
PHP 7

This is not the PHP you remember!

PHP 7: Faster!


WordPress 4.6 Performance (req/s)
PHP 5.6
PHP 7.2
0
100
200
300
400
500
600

PHP 7: Lower memory



Serverless platforms

Serverless platforms with PHP support

Hello World

AWS Lambda (Bref):

```php
<?php

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

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

Hello World

Apache OpenWhisk (OSS, IBM, Nimbella):

```php
<?php

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

Hello World

OpenFAAS (OSS):

```php
<?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
<?php

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

APACHE OpenWhisk

The anatomy of an action

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

Hello World


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

Hello World

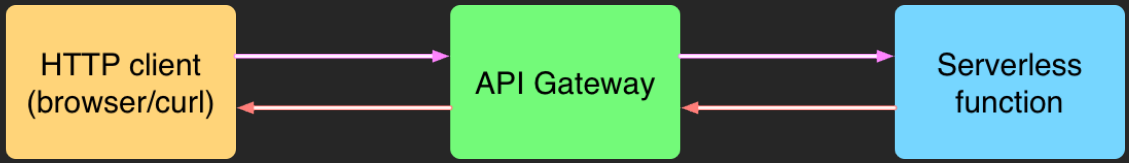
```php
function main(array $args) : array
{
    // Marshall inputs from event parameters
    $name = $args['name'] ?? 'world';
    // Do the work
    $message = 'Hello ' . $name
    // Return result
    return ["message" => $message];
}
```

HTTP Access

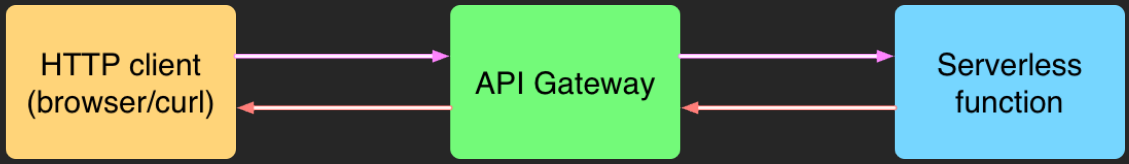
API Gateway provides:

- API routing
- Rate limiting
- Authentication
- Custom domains

API Gateway

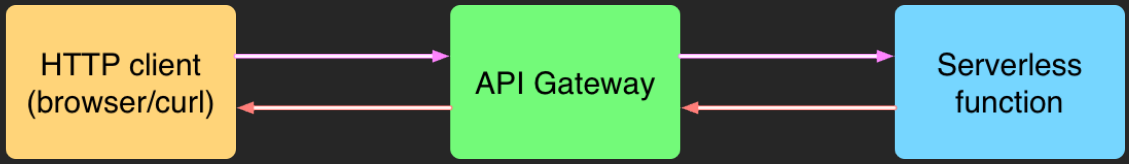


API Gateway



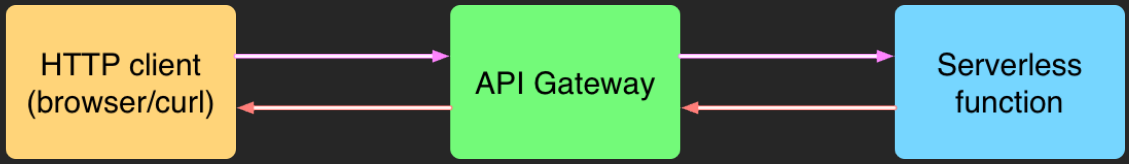

```
$ wsk api create /myapp /hello GET hello
```

API Gateway




```
$ wsk api create /myapp /hello GET hello
ok: created API /myapp/hello GET for action /_/hello
```

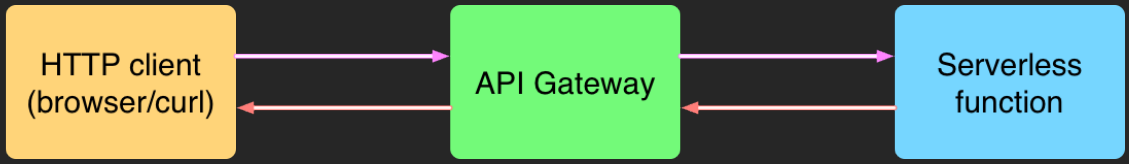
API Gateway




```
$ wsk api create /myapp /hello GET hello
ok: created API /myapp/hello GET for action /_/hello
```

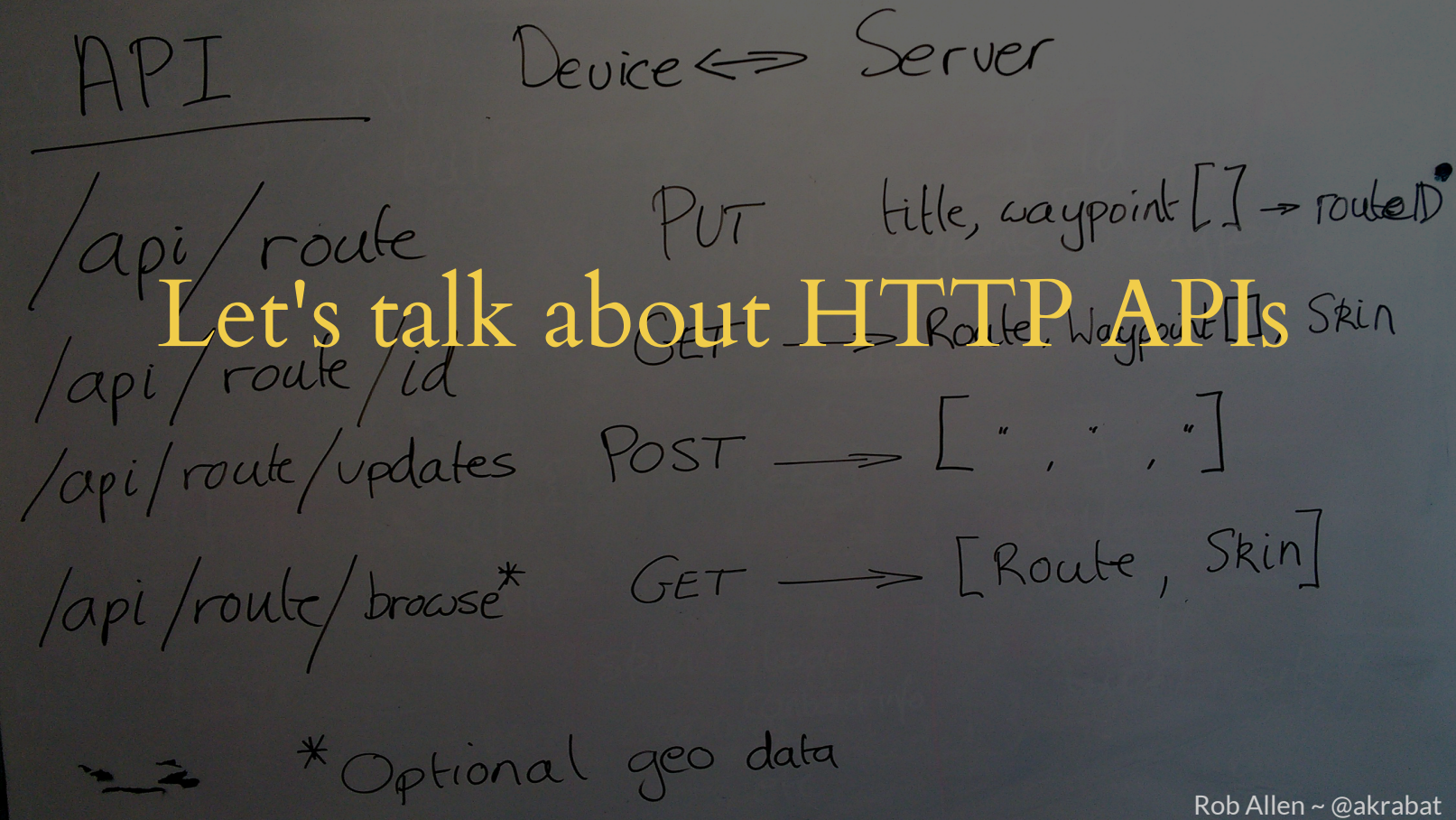
```
$ curl https://example.com/myapp/hello?name=Rob
```

API Gateway




```
$ wsk api create /myapp /hello GET hello
ok: created API /myapp/hello GET for action /_/hello
```

```
$ curl https://example.com/myapp/hello?name=Rob
{
  "message": "Hello Rob!"
}
```



Let's talk about HTTP APIs

HTTP APIs

Just because it's *serverless* doesn't mean we can ignore the basics!

- Status codes
- HTTP method negotiation
- Content-type handling
- Error handling
- Media type format

Status codes

Send the right one for the right situation!

1xx	Informational
2xx	Success
3xx	Redirection
4xx	Client error
5xx	Server error

Full control over response

```
function main(array $args) : array
{
  // our code here to do something

  return [
    "statusCode" => 201,
     "headers" => [
         "Content-Type" => "application/json",
     ],
    "body" => [
      "result" => "Item created",
    ],
  ];
}
```

HTTP verbs

Method	Used for	Idempotent?
GET	Retrieve data	Yes
PUT	Change data	Yes
DELETE	Delete data	Yes
POST	Change data	No
PATCH	Update data	No

Prefer Idempotent whenever possible

HTTP request information

Parameters arrive in `args` associative array:

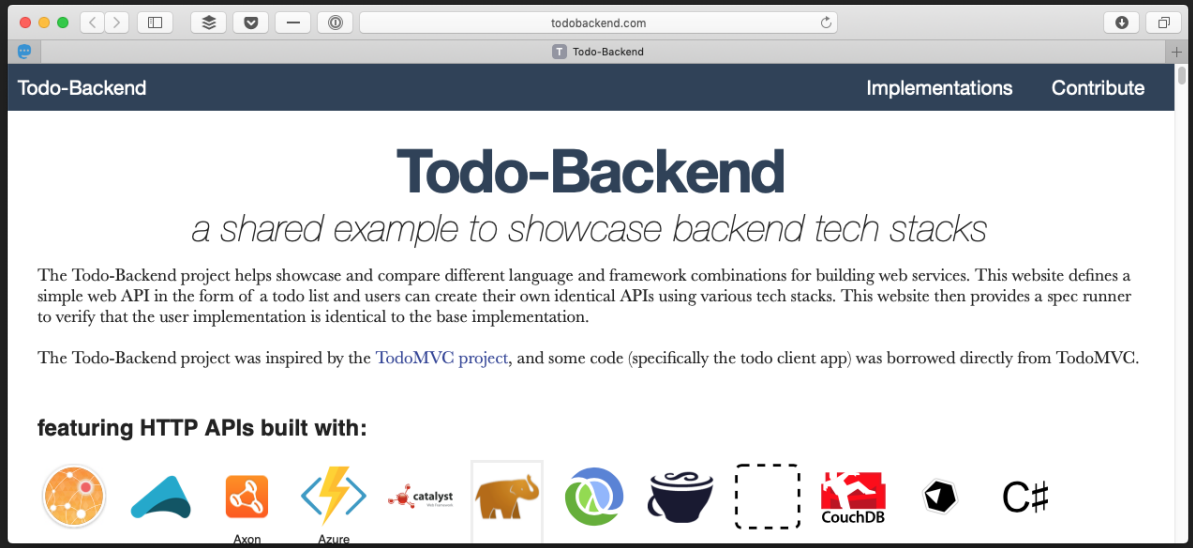
`$args['name']`	`name` parameter in body or query
`$args['date']`	`date` parameter in body or query

`__ow`-prefixed keys for for HTTP data:

`$args['__ow_method']`	HTTP method
`$args['__ow_path']`	URL path
`$args['__ow_headers']`	HTTP headers
`$args['__ow_body']`	Base64 encoded HTTP body

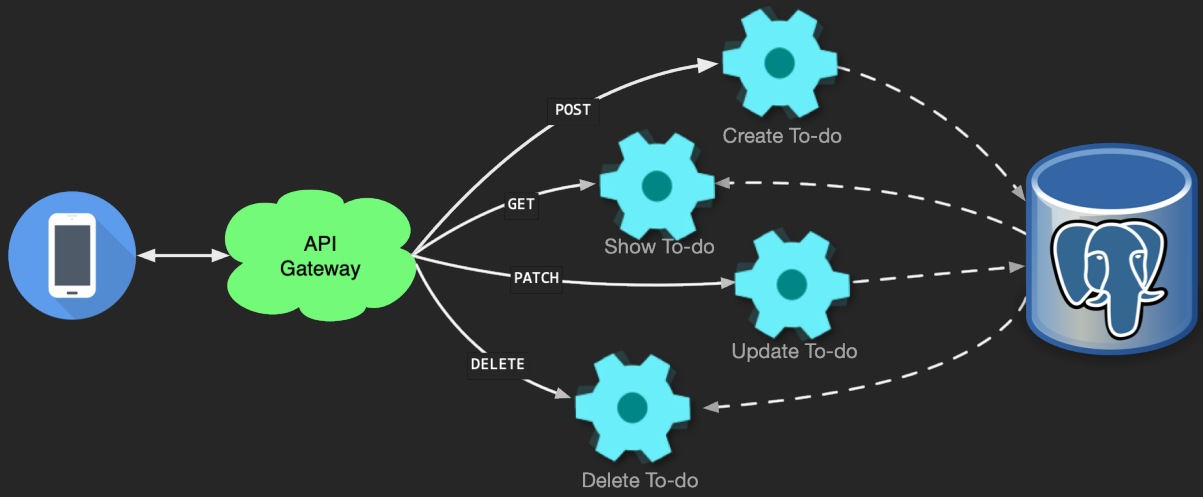
Let's look at a case study

A Serverless API



Todo-Backend

An OpenWhisk PHP implementation of a to-do list API



Serverless Framework

Deployment tooling for serverless applications

`serverless.yml:`

```
service: ow-todo-backend

provider:
  name: openwhisk
  runtime: php

plugins:
  - serverless-openwhisk
```

Configure action

```yaml
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/
├── vendor/
├── composer.json
├── composer.lock
└── serverless.yml
```

```
src/
├── Todo/
│   ├── Todo.php
│   ├── TodoMapper.php
│   └── TodoTransformer.php
├── actions/
│   ├── addTodo.php
│   ├── deleteTodo.php
│   ├── editTodo.php
│   ├── listTodos.php
│   └── showTodo.php
└── AppContainer.php
```

editTodo.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);

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

      return [
          'statusCode' => 200,
          'body' => $fractal->createData($resource)->toArray(),
      ];
  } catch (Throwable $e) {
      error_log((string)$e);
      $code = $e->getCode() < 400 ? 500 : $e->getCode();
      return [
          'statusCode' => $code,
          'body' => ['error' => $e->getMessage()]];
  }
}
```

editTodo.php: Error handling

```
function main(array $args) : array
{
  try {
      // do stuff
  } catch (Throwable $e) {
      error_log((string)$e);
      $code = $e->getCode() < 400 ? 500 : $e->getCode();
      return [
          'statusCode' => $code,
          'body' => ['error' => $e->getMessage()]];
  }
}
```

editTodo.php: Error handling

```
function main(array $args) : array
{
  try {
      // do stuff
  } catch (Throwable $e) {
      error_log((string)$e);
      $code = $e->getCode() < 400 ? 500 : $e->getCode();
      return [
          'statusCode' => $code,
          'body' => ['error' => $e->getMessage()]];
  }
}
```

editTodo.php: Read input

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

editTodo.php: Read input

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

// decode HTTP body
$body = base64_decode($args['__ow_body'];
$data = json_decode($body), true);
```

editTodo.php: Do the work

```
$todo = $mapper->loadById($id);
$mapper->update($todo, $data);
```

editTodo.php: Present results

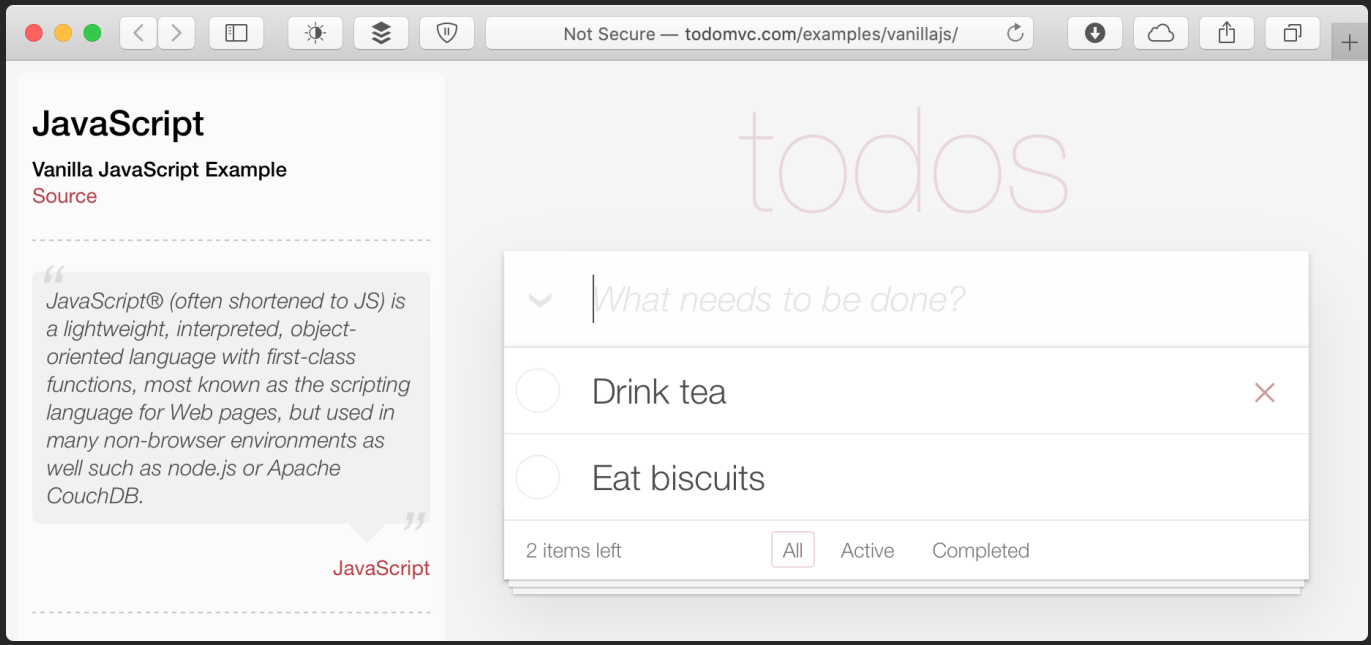
```php
$resource = new Fractal\Item($todo, $transformer, 'todos');
$viewData = $fractal->createData($resource);

return [
    'statusCode' => 200,
    'body' => $viewData->toArray(),
];
```

Deploy

```
$ 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...
[...]
```

A working API!




Thank you!

Photo credits

- Road sign: https://www.flickr.com/photos/ell-r-brown/6804246004
- Stars: https://www.flickr.com/photos/gsfc/19125041621