Implementing Serverless PHP

Under the hood of OpenWhisk

Rob Allen, Nineteen Feet

I write APIs

Serverless?

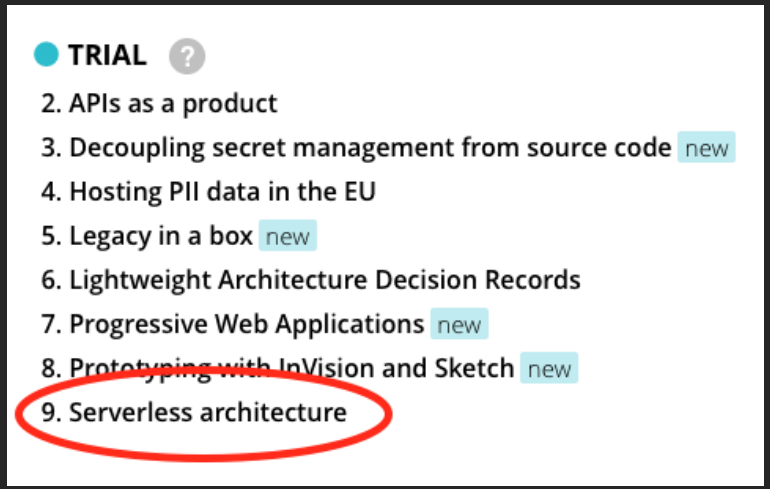
The first thing to know about serverless computing is that "serverless" is a pretty bad name to call it.

- Brandon Butler, Network World

AKA: Functions as a Service

- A runtime to execute your functions
- No capacity planning or load balancing; just tasks being executed.
- Pay for execution, not when idle

ThoughtWorks Technology Radar



"Our teams like the serverless approach; it's working well for us and we consider it a valid architectural choice.""

2017 Technology Radar

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)

Streaming

Continuous data flow to be processed

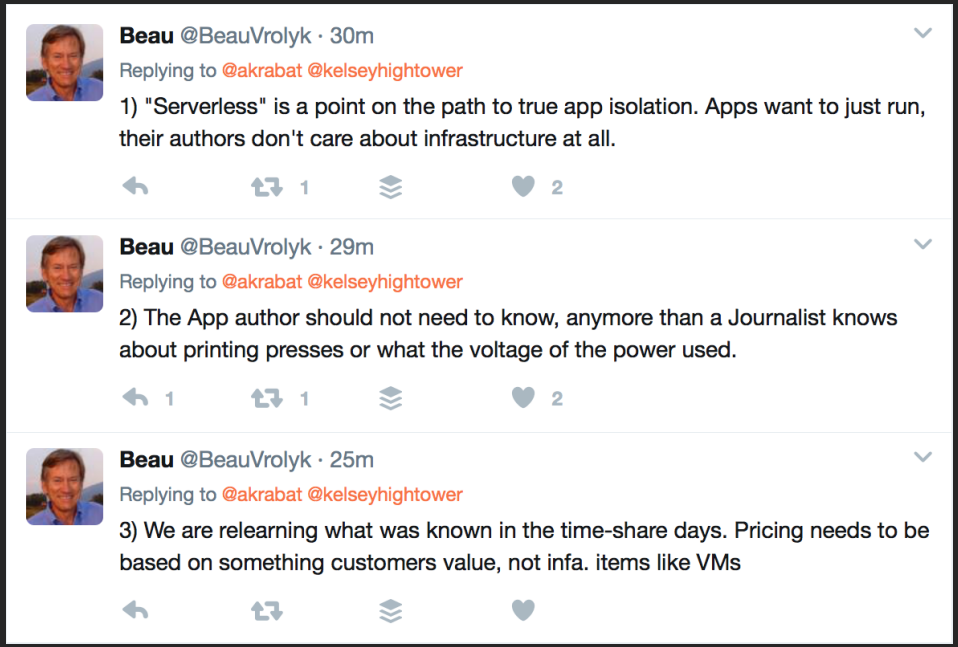
Benefits

- No need to think about servers
- Concentrate on application code
- Pay only for what you use, when you use it
- Language agnostic: NodeJS, Swift, Python, Java, C#, etc

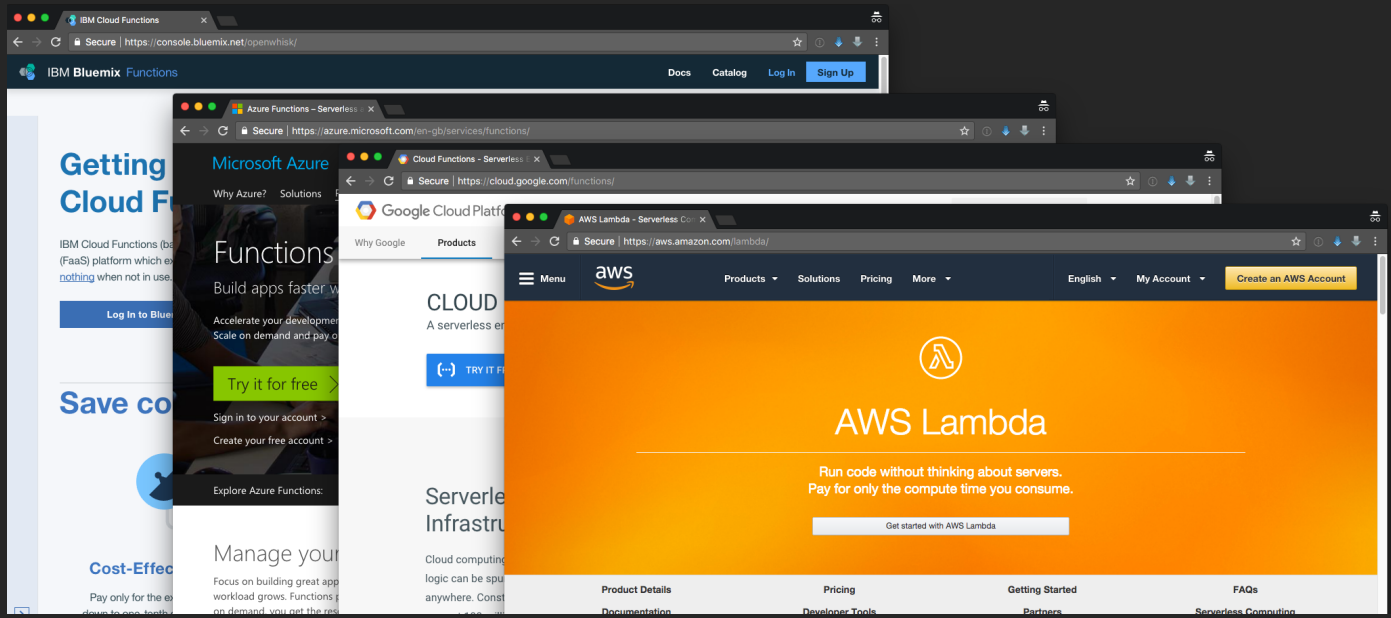
Challenges

- Start up latency
- Time limit
- State is external
- DevOps is still a thing

It's about value



Serverless providers



OpenWhisk

OpenWhisk

OpenSource; multiple providers:

- IBM
- RedHat
- Adobe (for Adobe Cloud Platform APIs)

&, of course, self-hosted

JavaScript

Hello world in JS

`hello.js`:

```
1 function main(params)
2 {
3     name = params.name || "World"
4     return {msg:  'Hello ' + name}
5 }
```

Create action:

```
$ wsk action create helloJS hello.js --kind nodejs:6
ok: updated action helloJS
```

Hello world in JS

Execute:

```
$ wsk action invoke -r helloJS -p name Rob
{
    "msg": "Hello Rob"
}
```

or:

```
$ curl -k https://192.168.33.13/api/v1/web/guest/default/helloJS.json
{
  "msg": "Hello World"
}
```

PHP

Hello world in PHP

`hello.php`:

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

The old way: Dockerise it

Create a bash script called **exec**:

```
#!/bin/bash

# Install PHP
if [ ! -f /usr/bin/php ]; then
    echo "http://dl-cdn.alpinelinux.org/alpine/edge/community" \
        >> /etc/apk/repositories
    apk add --update php7 php7-json
fi

# Run PHP action
/usr/bin/php -r 'require "/action/hello.php"';
echo json_encode(main(json_decode($argv[1], true)));' -- "$@"
```

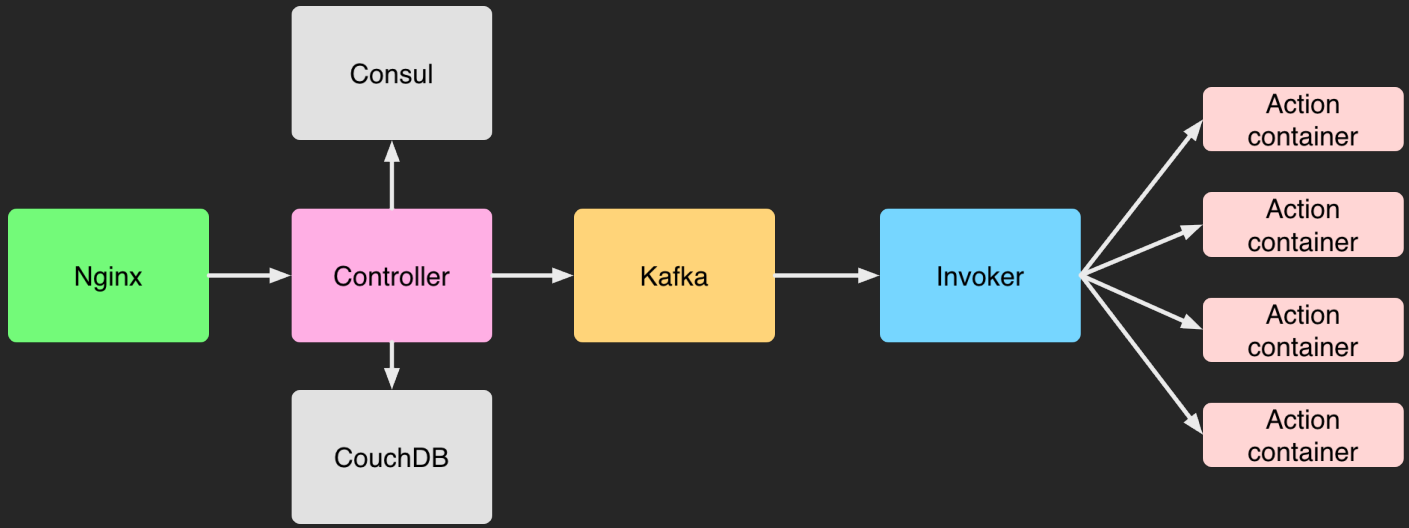
Create & execute action

```
$ zip -r hello.zip hello.php exec

$ wsk action create helloPHP hello.zip --native
ok: updated action helloPHP

$ wsk action invoke -r helloPHP -p name Rob
{
    "msg": "Hello Rob"
}
```

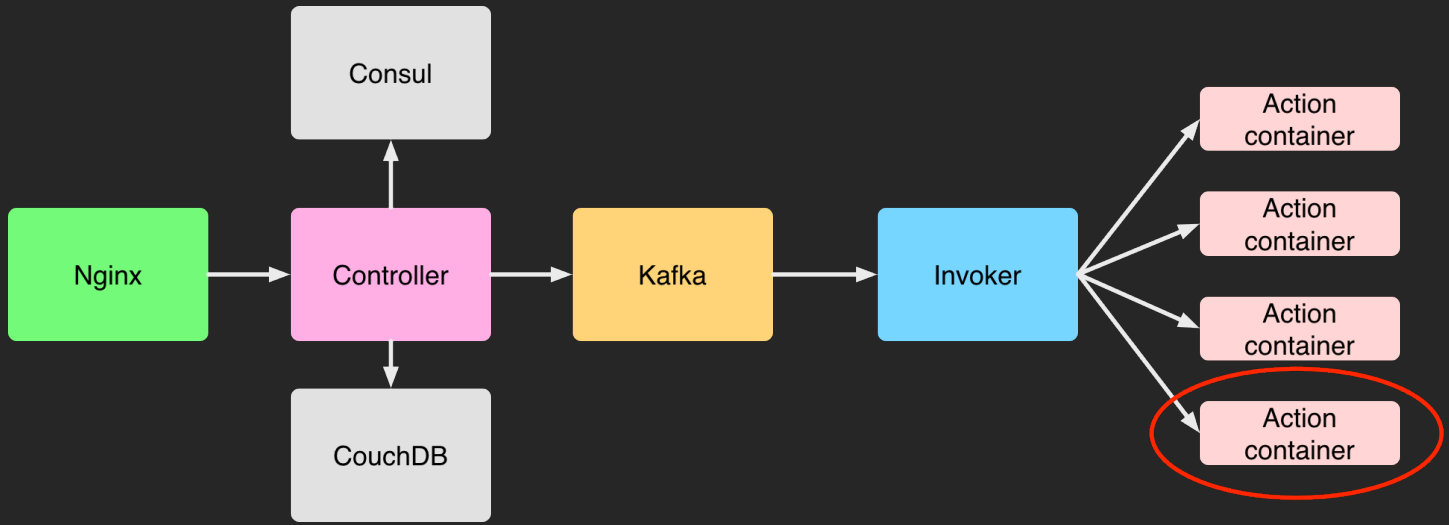
Time for first-run: 2 seconds

Solution:
Write a PHP action runner!

OpenWhisk's architecture

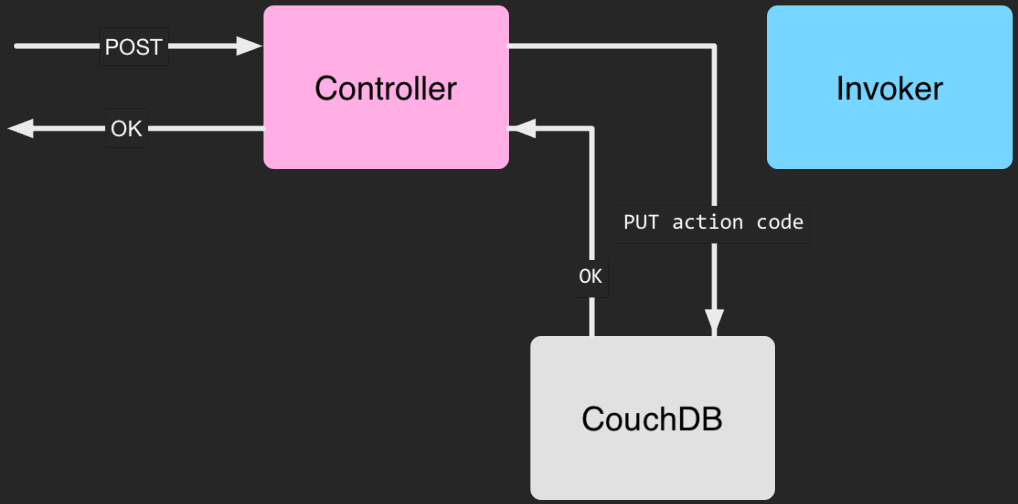


OpenWhisk's architecture



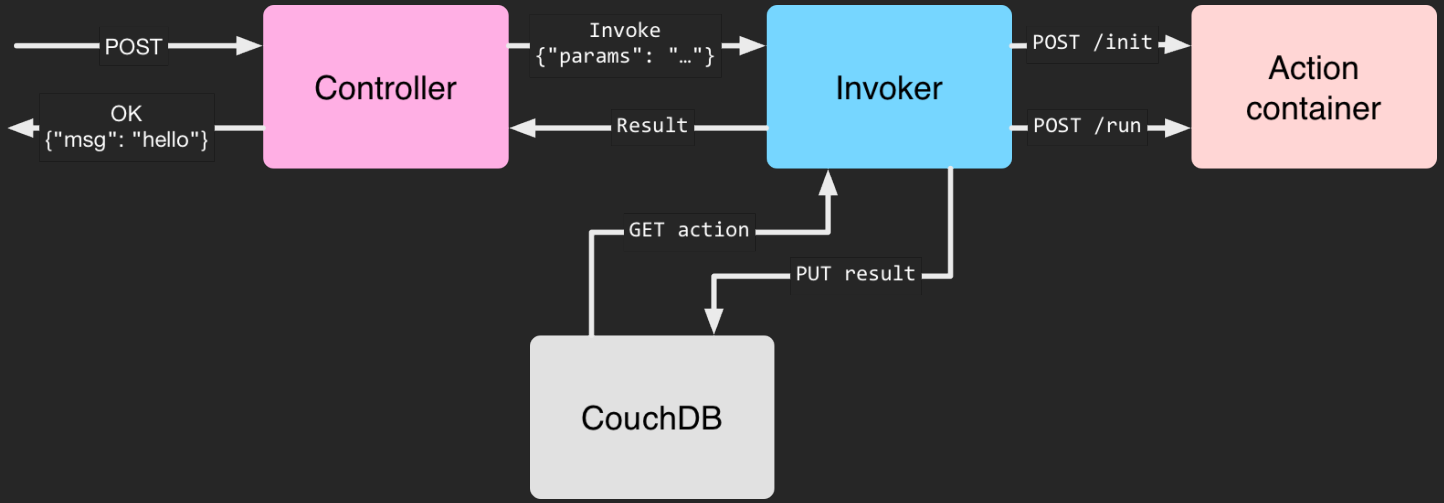
Create an action

```
wsk action create hello
```



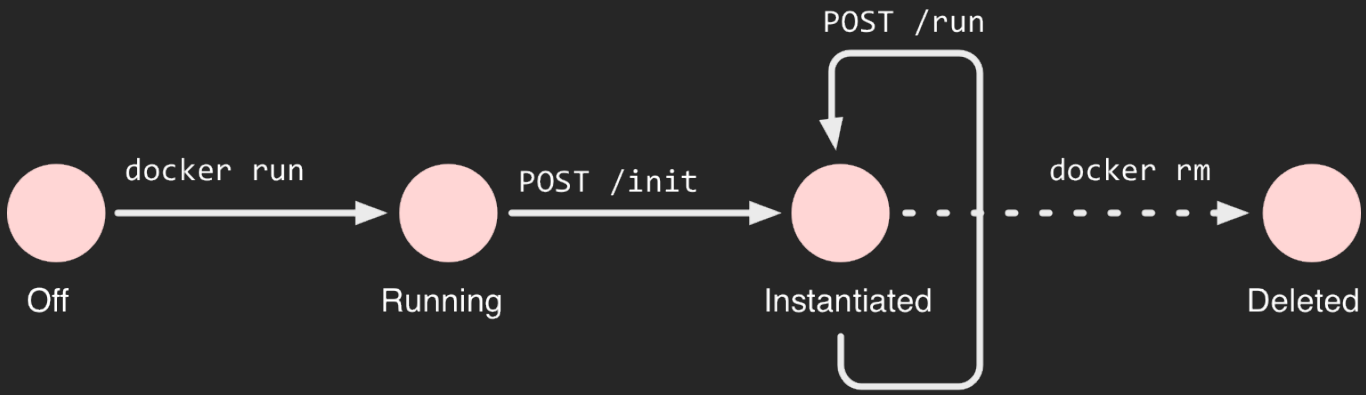
Invoke an action

```
wsk action invoke hello
```



Action container lifecycle

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



Action container API

input

```
{
  "value": {
    "name"  : "helloPHP",
    "main"  : "main",
    "binary": false,
    "code"  : "<?php …",
  }
}
```

```
{
  "value": {
    "name"  : "Rob",
  }
}
```

output

```
{ "OK": true}
```

```
`{"msg": "Hello Rob" }
```

Writing a PHP action container

We need:

- A container
- Code to handle endpoints (router.php)
- Execute the user code (runner.php)

Dockerfile

```
FROM php:7.1-alpine

# copy required files
ADD router.php /action
ADD runner.php /action

# Start webserver on port 8080
EXPOSE 8080
CMD [ "php", "-S", "0.0.0.0:8080", "/action/router.php" ]
```

Responding to the invoker: router.php

router.php

```php
<?php
if ($_SERVER['REQUEST_URI'] == '/init') {
    $result = init();
} elseif ($_SERVER['REQUEST_URI'] == '/run') {
    $result = run();
}

/* send response */
header('Content-Type: application/json');
echo json_encode((object)$result);
```

router.php: init()

```php
function init()
{
  $post = file_get_contents('php://input');
  $data = json_decode($post, true)['value'];

  file_put_contents('index.php', $data['code']);

  $config = ['function' => $data['main'], 'file' => 'index.php'];
  file_put_contents('config.json', json_encode($config));

  return ["OK" => true];
}
```

router.php: run()

```php
function run()
{
  $args = file_get_contents('config.json');
  $stdin = json_decode(file_get_contents('php://input'), true)['value'];

  list($rtncode, $stdout, $stderr) = runAction($args, $stdin);

  file_put_contents("php://stderr", $stderr);
  file_put_contents("php://stdout", $stdout);

  $pos = strrpos($stdout, PHP_EOL) + 1;
  $lastLine = trim(substr($stdout, $pos));

  return $lastLine;
}
```

router.php: run()

```
 1 function run()
 2 {
 3   $args = file_get_contents('config.json');
 4   $stdin = json_decode(file_get_contents('php://input'), true)['value'];
 5
 6   list($rtncode, $stdout, $stderr) = runAction($args, $stdin);
 7
 8   file_put_contents("php://stderr", $stderr);
 9   file_put_contents("php://stdout", $stdout);
10
11   $pos = strrpos($stdout, PHP_EOL) + 1;
12   $lastLine = trim(substr($stdout, $pos));
13
14   return $lastLine;
15 }
```

router.php: runAction()

```php
function runAction(array $args, string $stdin = '') : array
{
  $args = implode(' ', array_map('escapeshellarg', $args));

  /* execute runner and open file pointers for input/output */
  $pipes = [];
  $process = proc_open(
    '/usr/local/bin/php -f runner.php' . $args,
    [
        0 => ['pipe', 'r'], /* descriptor for stdin  */
        1 => ['pipe', 'w'], /* descriptor for stdout */
        2 => ['pipe', 'w']  /* descriptor for stderr */
    ],
    $pipes,
  );
```

router.php: runAction()

```php
function runAction(array $args, string $stdin = '') : array
{
  $args = implode(' ', array_map('escapeshellarg', $args));

  /* execute runner and open file pointers for input/output */
  $pipes = [];
  $process = proc_open(
    '/usr/local/bin/php -f runner.php' . $args,
    [
        0 => ['pipe', 'r'], /* descriptor for stdin  */
        1 => ['pipe', 'w'], /* descriptor for stdout */
        2 => ['pipe', 'w']  /* descriptor for stderr */
    ],
    $pipes,
  );
```

router.php: runAction()

```php
function runAction(array $args, string $stdin = '') : array
{
  $args = implode(' ', array_map('escapeshellarg', $args));

  /* execute runner and open file pointers for input/output */
  $pipes = [];
  $process = proc_open(
    '/usr/local/bin/php -f runner.php' . $args,
    [
        0 => ['pipe', 'r'], /* descriptor for stdin  */
        1 => ['pipe', 'w'], /* descriptor for stdout */
        2 => ['pipe', 'w']  /* descriptor for stderr */
    ],
    $pipes,
  );
```

router.php: runAction()

```
1   /* write to the process's stdin */
2   $bytes = fwrite($pipes[0], $stdin); fclose($pipes[0]);
3
4   /* read the process's stdout */
5   $stdout = stream_get_contents($pipes[1]); fclose($pipes[1]);
6
7   /* read the process's stderr */
8   $stderr = stream_get_contents($pipes[2]); fclose($pipes[2]);
9
10  /* close process & get return code */
11  $returnCode = proc_close($process);
12
13  return [$returnCode, $stdout, $stderr];
14 }
```

router.php: run()

```
 1 function run()
 2 {
 3   $args = file_get_contents('config.json');
 4   $stdin = json_decode(file_get_contents('php://input'), true)['value'];
 5
 6   list($rtncode, $stdout, $stderr) = runAction($args, $stdin);
 7
 8   file_put_contents("php://stderr", $stderr);
 9   file_put_contents("php://stdout", $stdout);
10
11   $pos = strrpos($stdout, PHP_EOL) + 1;
12   $lastLine = trim(substr($stdout, $pos));
13
14   return $lastLine;
15 }
```

router.php: run()

```php
function run()
{
  $args = file_get_contents('config.json');
  $stdin = json_decode(file_get_contents('php://input'), true)['value'];

  list($rtncode, $stdout, $stderr) = runAction($args, $stdin);

  file_put_contents("php://stderr", $stderr);
  file_put_contents("php://stdout", $stdout);

  $pos = strrpos($stdout, PHP_EOL) + 1;
  $lastLine = trim(substr($stdout, $pos));

  return $lastLine;
}
```

router.php: run()

```php
function run()
{
  $args = file_get_contents('config.json');
  $stdin = json_decode(file_get_contents('php://input'), true)['value'];

  list($rtncode, $stdout, $stderr) = runAction($args, $stdin);

  file_put_contents("php://stderr", $stderr);
  file_put_contents("php://stdout", $stdout);

  $pos = strrpos($stdout, PHP_EOL) + 1;
  $lastLine = trim(substr($stdout, $pos));

  return $lastLine;
}
```

Running the action code: runner.php

runner.php

Runs the user's code in a separate process

```php
<?php
$config = json_decode($argv[1], true);
$functionName = $config['function'] ?? 'main';

require '/action/vendor/autoload.php';
require '/action/src/index.php';
$result = $functionName(
  json_decode(file_get_contents('php://stdin') ?? [], true)
);

/* last line of output is response */
echo json_encode((object)$result);
```

runner.php

Runs the user's code in a separate process

```php
<?php
$config = json_decode($argv[1], true);
$functionName = $config['function'] ?? 'main';

require '/action/vendor/autoload.php';
require '/action/src/index.php';
$result = $functionName(
  json_decode(file_get_contents('php://stdin') ?? [], true)
);

/* last line of output is response */
echo json_encode((object)$result);
```

runner.php

Runs the user's code in a separate process

```php
<?php
$config = json_decode($argv[1], true);
$functionName = $config['function'] ?? 'main';

require '/action/vendor/autoload.php';
require '/action/src/index.php';
$result = $functionName(
  json_decode(file_get_contents('php://stdin') ?? [], true)
);

/* last line of output is response */
echo json_encode((object)$result);
```

runner.php

Runs the user's code in a separate process

```php
<?php
$config = json_decode($argv[1], true);
$functionName = $config['function'] ?? 'main';

require '/action/vendor/autoload.php';
require '/action/src/index.php';
$result = $functionName(
  json_decode(file_get_contents('php://stdin') ?? [], true)
);

/* last line of output is response */
echo json_encode((object)$result);
```

Hello world in PHP

`hello.php`:

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

Execute natively in OpenWhisk

```
$ wsk action create hello hello.php --kind php:7.1
ok: updated action hello

$ wsk action invoke -r hello -p name Rob
{
    "msg": "Hello Rob"
}
```

Time for first-run: 400 ms

Demo

Summary

Resources

- https://www.martinfowler.com/articles/serverless.html
- http://www.openwhisk.org
- https://medium.com/openwhisk

- http://github.com/apache/incubator-openwhisk/pull/2415
- https://github.com/akrabat/ow-php-ftime

Thank you!

Rob Allen ~ @akrabat
PHP Training: @phptraininguk