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)

Rob Allen ~ @akrabat
Platform options

Abstraction

- Serverless (OpenWhisk)
- 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)
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
Your code
Deployed to the cloud
FaaS

Runs when needed

Rob Allen ~ @akrabat
Scaled automatically
FaaS

Pay only for execution
Where are the servers?
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

• 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
Replying 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.

Beau @BeauVrolyk · 29m
Replying 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.

Beau @BeauVrolyk · 25m
Replying 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

- Azure
- Google Cloud Platform
- AWS
- IBM Cloud
- fn
- Apache OpenWhisk
Serverless languages

- JavaScript
- .NET Core
- Docker Logo (likely representing Docker)
- PHP
- Python
- Go
- Java
- Ruby
- Swift

Rob Allen ~ @akrabat
Serverless platforms with PHP support

Azure
aws
IBM Cloud
Google Cloud Platform
fn
Google App Engine

Rob Allen ~ @akrabat
Hello World

AWS Lambda (Bref):

```php
<?php

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

Apache OpenWhisk:

```php
<?php

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

```
```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;
}
```

Rob Allen ~ @akrabat
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

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];
}
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];
}
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];
}
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!"
}
Under the hood
OpenWhisk's architecture

- Nginx
- Controller
- Kafka
- Invoker
  - NodeJS Action container
  - Python Action container
  - Java Action container
  - PHP Action container

CouchDB
Create an action

```bash
$ 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`
Monolith architecture

- Web browser
- NGINX
- Web application
- Database
- Static files (CSS, JS, etc)
Serverless architecture

- Web browser (JS app)
- API Gateway
- Serverless functions
- Database
- S3 file storage (CSS, JS, etc)
- 3rd party services (e.g. Auth)
Serverless architecture pattern

API Gateway

Function
Function
Function

Relational database
Flat table database

Scheduler

Triggered event

Function

Rob Allen ~ @akrabat
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
Case study

Project 365 photo website

Rob Allen ~ @akrabat
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
Infrastructure as code

serverless.yml:

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

Rob Allen ~ @akrabat
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]);
};
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

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 January 2021</td>
<td>Post box in the early morning</td>
</tr>
</tbody>
</table>
To sum up
Thank you!

Rob Allen ~ @akrabat
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