Welcome!

I am Wojtek Erbetowski. You pronounce my first name like "Voytech". And this talk is about Serverless computing.

I you want to mention me, I am:
* @erbetowski on Twitter
* wojtek.erbetowski pretty much everywhere else

Who am I?

Generalist.

#java #machinelearning #scala #groovy #android #python #microservices #django #spring #gradle #docker #ecmascript #reactjs

Passionate about the people, then the process.

MSc in mathematics.

What is Serverless?

What is Serverless?

1. Abstract from the runtime (server)
2. Stateless
3. Event driven
4. Ephemeral

another name is **FaaS** or **Function as a Service**

a nice comparison is that

Serverless is to **computing**

what **S3** is to **file storage**

What Serverless is not?

1. BaaS (or is it?)
2. parallel computing engine

Current options are

1. Amazon AWS Lambda
2. Google Cloud Functions
3. Microsoft Azure Functions
4. IBM OpenWhisk
5. OpenStack Picasso
6. minor frameworks

AWS Lambda

Event driven

1. S3
2. DynamoDB
3. Simple Notification Service
4. CloudWatch
5. API Gateway
6. and many more

Supports variety of languages

-> Node.js – v4.3.2 and 6.10.2

-> Java – Java 8

-> Python – Python 3.6 and 2.7

-> .NET Core – .NET Core 1.0.1 (C#)

-> (extra) language of your choice

Vendor lock-in

1. APIs differ between cloud providers
2. API is pretty minimal though

Pricing

Lambdas are priced per:

1. number of requests
1. 100s of milliseconds (depends on declared memory)

Free tier is very generous.

Pricing

$0.20 per 1 million requests.

1024 MB: 1$ -> 600k units -> 16 hours

128 MBs: 1$ -> 5M units -> 133 hours

Free tier

First 1 million requests per month.

512 MB app can run for 800,000s (~200 hours).

Lambda > Functions > myFunctionName

Qualifiers | Test | Actions

Code | Configuration | Triggers | Monitoring

Code entry type: Edit code inline

```
1  from __future__ import print_function
2
3  import json
4
5  print('Loading function')
6
7
8  def lambda_handler(event, context):
9      print("Received event.")
10
11     operation = event['httpMethod']
12
13     response = {
14         'statusCode': '200',
15         'body': 'Hello from lambda fun! V2!',
16         'headers': {
17             'Content-Type': 'application/json',
18         },
19     }
20
21     return response
22
23
```

Serverless frameworks

Raising the API level

1. Serverless.com
2. Apex
3. Chalice
4. Kappa
5. Sparta
6. Zappa

Serverless framework

1. Deployments
2. Declarative (YAML)
3. Resource management (S3, DynamoDB, VPC)
4. Logs
5. Triggers, APIs
6. Local execution

set up

```yaml
service: my-service

provider:
  name: aws
  runtime: nodejs6.10

functions:
  hello:
    handler: handler.hello
    events:
      - http:
          path: hello
          method: get
```

handle

```
module.exports.hello = (event, context, callback) => {
  const response = {
    statusCode: 200,
    body: JSON.stringify({
      message: 'Hello Polyconf!',
      input: event,
    }),
  };

  callback(null, response);
};
```

deploy

```
$ serverless deploy
Serverless: Packaging service...
Serverless: Uploading CloudFormation file to S3...
Serverless: Uploading artifacts...
Serverless: Uploading service .zip file to S3 (307 B)...
Serverless: Validating template...
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
.............
Serverless: Stack update finished...
Service Information
service: my-service
stage: dev
region: us-east-1
api keys:
  None
endpoints:
  GET - https://.../dev/hello
functions:
  hello: my-service-dev-hello
```

partial deploy

```
$ serverless deploy function -f hello
```

Use

```
$ serverless invoke -f hello
```

or

```
$ curl "https://.../dev/hello"
```

output:

```
{
  message: "Hello Polyconf!",
  input: {
    resource: "/hello",
    path: "/hello",
    httpMethod: "GET",
    headers: {
      ...
}
```

how to async?

```
export function helloPromise(event, context, callback) {
  console.log('Running hello promise');
  fetch('https://api.github.com/users/github')
    .then(res => res.json())
    .then(json => {
      const location = json['location'];

      callback(null, {
        status: 200,
        body: JSON.stringify({ location }),
      });
    }, err => callback(null, {
      status: 200,
      error: err,
    }));
}
```

how to async?

```yaml
plugins:
  - serverless-webpack
```

```
export async function helloAsync(event, context, callback) {
  const res = await fetch('https://api.github.com/users/github');
  const json = await res.json();
  const location = json['location'];

  callback(null, {
    status: 200,
    body: JSON.stringify({ location }),
  });
}
```

working with S3

```
fetch('image URL')
  .then(res => {
    return s3.putObject({Bucket, Key, Body: res.body}).promise();
  }).then(res => {
    callback(null, res);
  }).catch(err => {
    callback(err, null);
  });
```

working with S3

```yaml
functions:
  users:
    handler: users.handler
    events:
      - s3:
          bucket: photos
          event: s3:ObjectCreated:*
          rules:
            - prefix: uploads/
            - suffix: .jpg
```

scheduling tasks

```yaml
functions:
  cron:
    handler: handler.run
    events:
      # Invoke Lambda function every minute
      - schedule: rate(1 minute)
  secondCron:
    handler: handler.run
    events:
      # Invoke Lambda function every 2nd minute from Mon-Fri
      - schedule: cron(0/2 * ? * MON-FRI *)
```

Perfect use cases

1. S3 images processing
2. Chatbots
3. Websites
4. Inconsistent traffic
5. Log analysis on the fly
6. Event Sourcing

Tools that are easy to use take over the world

Limitations

1. Disk space
2. Deployment package size
3. Memory
4. Time

Pain points

1. Monitor usage
2. Debugging
3. Wall of requests hit AWS
4. Cost management

Biggest wins

1. Scalability
2. Pricing
3. NoOps

Serverlessify regular app

Project Golem


Server less
λ