Chalice microframework 101

Październik 2017

Wojciech Lichota

STX Next

wojciech@lichota.pl

http://lichota.pl


STX NEXT
python powerhouse
ZAWINĘLIŚMY
DO
GDAŃSKA
SMILE
ŻEGLUGA
GDAŃSK

Serverless

AWS Lambda

Chalice

Use case

Serverless

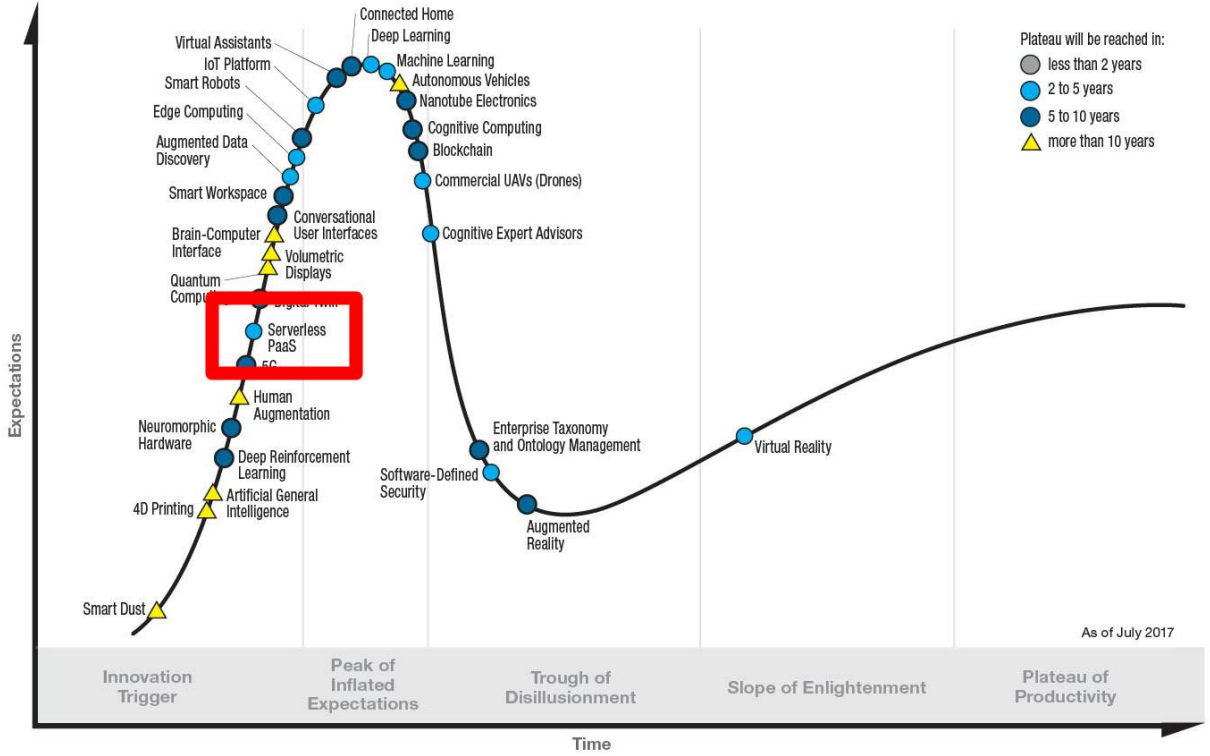
AWS Lambda

Chalice

Use case

Hype Cycle

Gartner Hype Cycle for Emerging Technologies, 2017



gartner.com/SmarterWithGartner

Gartner

Serverless vs. ...

Dedyk.	Nabywca	Dostawca
IaaS	Nabywca	Dostawca
PaaS	Nabywca	Dostawca
Serverless	Nabywca	Dostawca
SaaS	Nabywca	Dostawca

Usługa

Funkcja

Aplikacja (proces)

System operacyjny

Serwer

Infrastruktura sieciowa

Przykłady

	IaaS	PaaS	Serverless
Amazon Web Services	EC2	Elastic Beanstalk	Lambda
Google Cloud Platform	Compute Engine	App Engine	Cloud Functions
Microsoft Azure	Virtual Machines	App Service	Functions
Pozostałe	Rackspace	Heroku	Apache OpenWhisk

Model płatności

Dedykowany	IaaS	PaaS	Serverless
Stała miesięczna stawka	Czas pracy maszyny wirtualnej (w godzinach)	Czas pracy aplikacji (w godzinach)	Ilość wywołań
	Klasa maszyny wirtualnej (CPU i RAM)	Klasa środowiska wykonawczego (CPU i RAM)	Czas pracy (milisekundy CPU)
			Zużyta pamięć (milisekundy RAM)

Serverless

AWS Lambda

Chalice

Use case

- ★ Uruchomienie 11.2014
- ★ Python, JavaScript (Node.js), Java, C# (.NET)
- ★ Python 2.7 i 3.6 (od 04.2017)
- ★ Event-driven
  - ○ Zdarzenia wewnętrzne
  - ○ API Gateway
- ★ RAM: 128 MB - 1536 MB
- ★ CPU: ? (więcej MB -> więcej GHz)

Cennik:

- ★ Zaokrąglenie do 100ms
- ★ $0.0000006 per request
- ★ $0.00005001 per GB-second
- ★ Free tier
  - ○ 1 000 000 requests
  - ○ 400 000 GB-seconds

Porównanie kosztów Lambda i EC2

https://www.trek10.com/blog/lambda-cost/

Function Execution Memory & Time	Requests per Hour Required for Lambda Cost to Equal EC2 (m4.large) Cost	Requests per Second
100 ms @ 128 MB	295 000	81.9
200 ms @ 512 MB	64 000	17.8
200 ms @ 1 GB	34 000	9.4
1 sec @ 1 GB	7 100	2.0


```python
from time import time
import os

def lambda_handler(event, context):
    start = time()
    response = {
        "event": event,
        "context": vars(context),
        "environ": dict(os.environ),
    }
    del response["context"]["identity"]
    print('EXEC TIME: {:.2f} ms'.format((time() - start) * 1000))
    return response
```

Serverless

AWS Lambda

Chalice

Use case

Chalice

microframework

Python Serverless Microframework for AWS

- ★ Ułatwia deklarowanie endpointów
- ★ Upraszcza obsługę żądań HTTP
- ★ Narzędzie do “deployowania”
- ★ Automatycznie tworzy IAM policy
- ★ Lokalny serwer
- ★ Przeglądanie logów

Chalice

microframework

Instalacja

```
mkvirtualenv --python=`which python3` chalice
pip install chalice awscli
aws configure
chalice new-project demo
cd demo
```

Chalice

microframework

```python
import os
from time import time
from chalice import Chalice

app = Chalice(app_name='demo')

@app.route('/')
def index():
    start = time()
    response = {
        "request": app.current_request.to_dict(),
        "environ": dict(os.environ),
    }
    print('EXEC TIME: {:.2f} ms'.format((time() - start) * 1000))
    return response
```

Chalice

microframework

- ★ Lokalne uruchomienie

```
chalice local --port=8080
```

- ★ “Deployowanie”

```
chalice deploy
```

```
cat .chalice/policy-dev.json
```

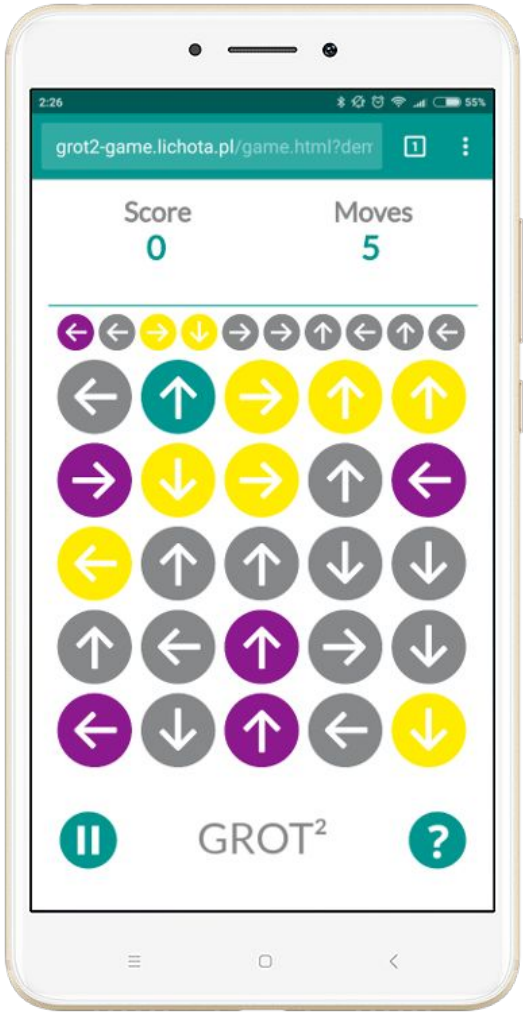
```
unzip -l .chalice/deployments/*.zip
```

Serverless

AWS Lambda

Chalice

Use case

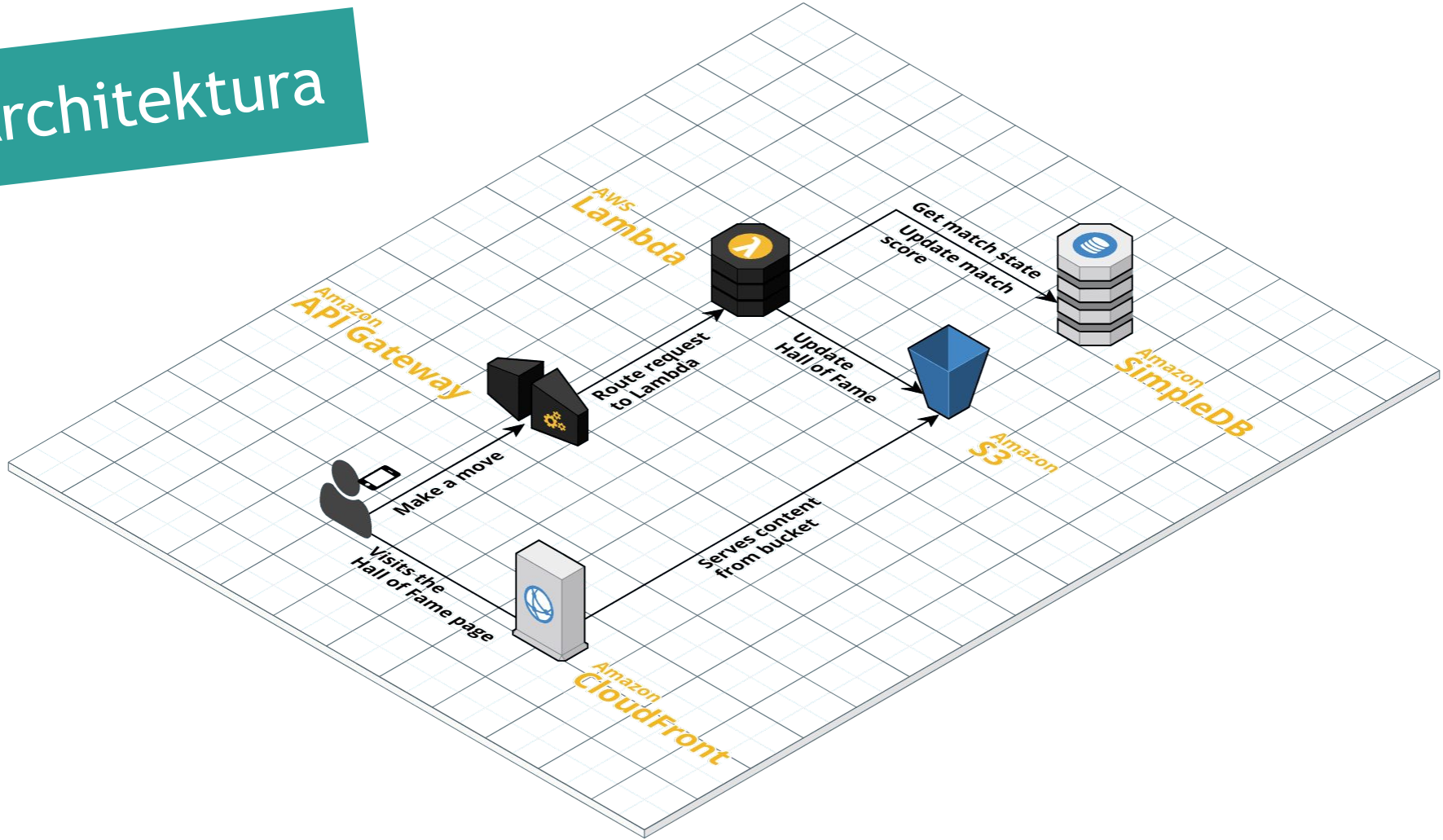


GROT² game

http://grot2-game.lichota.pl/

https://github.com/sargo/grot2

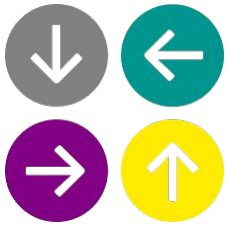
Architektura




```
import os
from chalice import Chalice, CORSConfig
from chalicelib import s3, sdb, settings

app = Chalice(app_name='demo')
app.debug = settings.DEBUG

cors_config = CORSConfig(
    allow_origin=os.environ.get(
        'CORS_ALLOW_ORGIN', settings.CORS_ALLOW_ORGIN),
     max_age=86400,
)
```



GROT²

game

```
@app.route(
    '/match/{match_id}',
    methods=['POST'],
    cors=cors_config,
    api_key_required=True,
)
def make_move(match_id):
    api_key = app.current_request.context['identity']['apiKey']
    match = sdb.get_match(api_key, match_id)
    data = app.current_request.request.json_body
    if 'row' not in data or 'col' not in data:
        raise BadRequestError('row or col is missing')
    match.start_move(data['row'], data['col'])
    if not match.is_active():
        s3.update_hall_of_fame(match)
    return match.get_state()
```

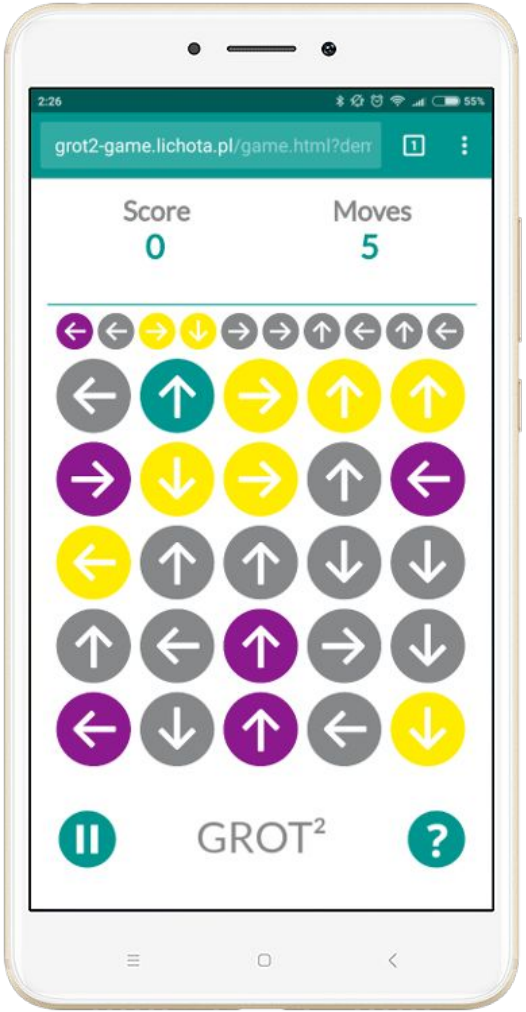
Porady

- ★ Zwiększaj RAM (więc i GHz) tak aby większość żądań była lekko poniżej 100ms
- ★ Ogranicz komunikację z innymi serwisami, bo płacisz za oczekiwanie na nie
- ★ Ustaw Usage Plan w API Gateway aby ograniczyć ilość żądań
- ★ Przyjrzyj się Zappa (WSGI na AWS Lambda)

Podsumowanie

- ★ Popularność Serverless rośnie
- ★ AWS Lambda pozwala optymalizować koszty
- ★ Chalice ułatwia tworzenie “Lambd”
- ★ Chalice nie służy do budowy całych aplikacji
- ★ Budując w oparciu o Chalice uzależniasz się całkowicie od AWS

ZAWINĘLIŚMY
DO
GDAŃSKA



Konkurs

http://bit.do/grot2