Chalice

AWS Lambda microframework

April 2018

Wojciech Lichota

STX Next

wojciech@lichota.pl

http://lichota.pl

- Serverless
- AWS Lambda
- Chalice
- Use case

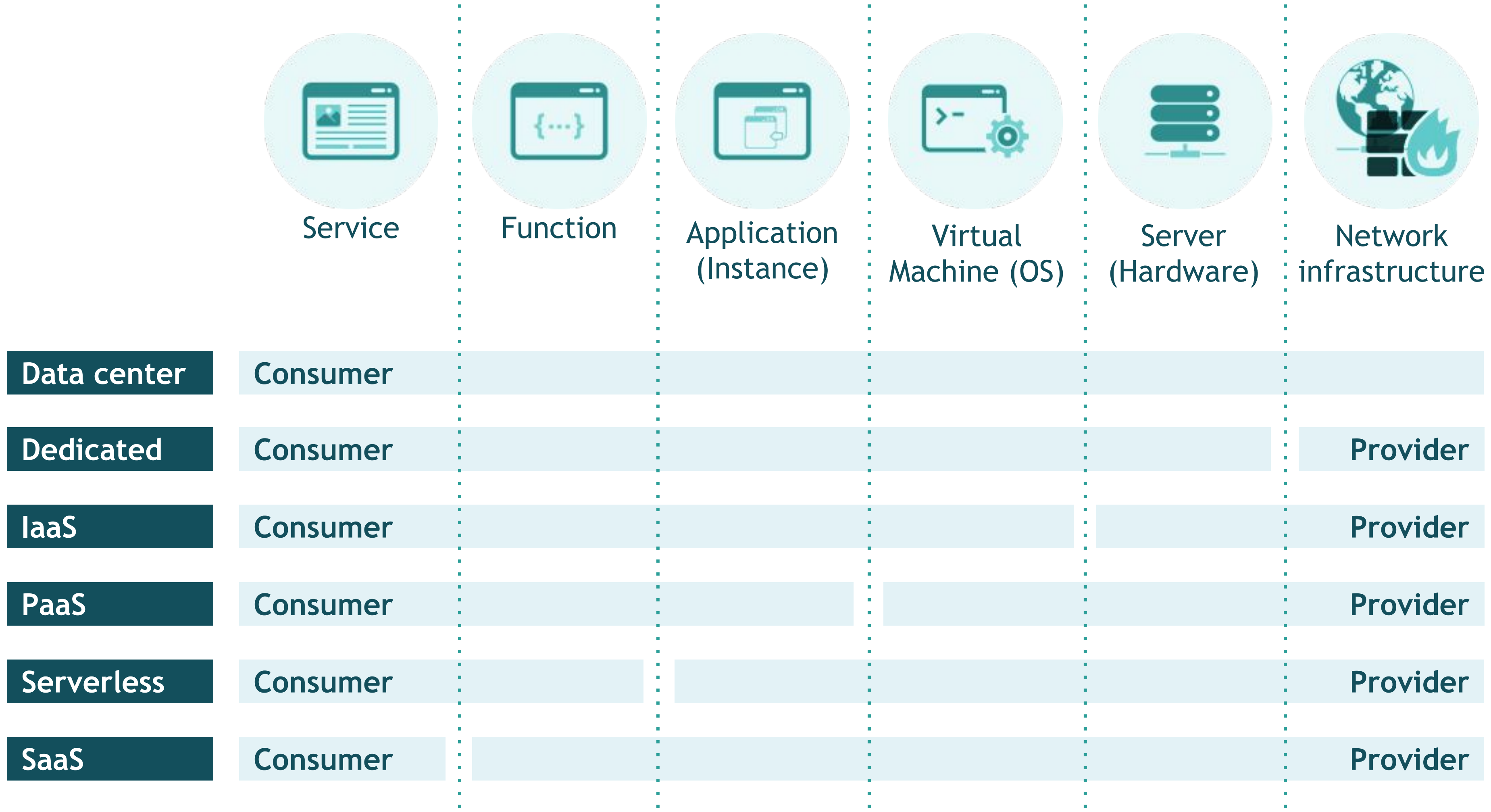
Serverless

AWS Lambda

Chalice

Use case

Serverless


Service
Function
Application (Instance)
Virtual Machine (OS)
Server (Hardware)
Network infrastructure
Data center
Consumer
Dedicated
Consumer
Provider
IaaS
Consumer
Provider
PaaS
Consumer
Provider
Serverless
Consumer
Provider
SaaS
Consumer
Provider

Examples

	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
	Rackspace	Heroku	Apache OpenWhisk

Serverless

AWS Lambda

Chalice

Use case

AWS

Lambda

- ★ Started in Nov 2014
- ★ Python, JavaScript (Node.js), Java, C# (.NET)
- ★ Python 2.7 i 3.6 (added in Apr 2017)
- ★ RAM: 128 MB - 1536 MB
- ★ CPU: ? (more MB -> more GHz)

- ★ Event-driven
  - ○ Internal events
  - ○ API Gateway

AWS
Lambda

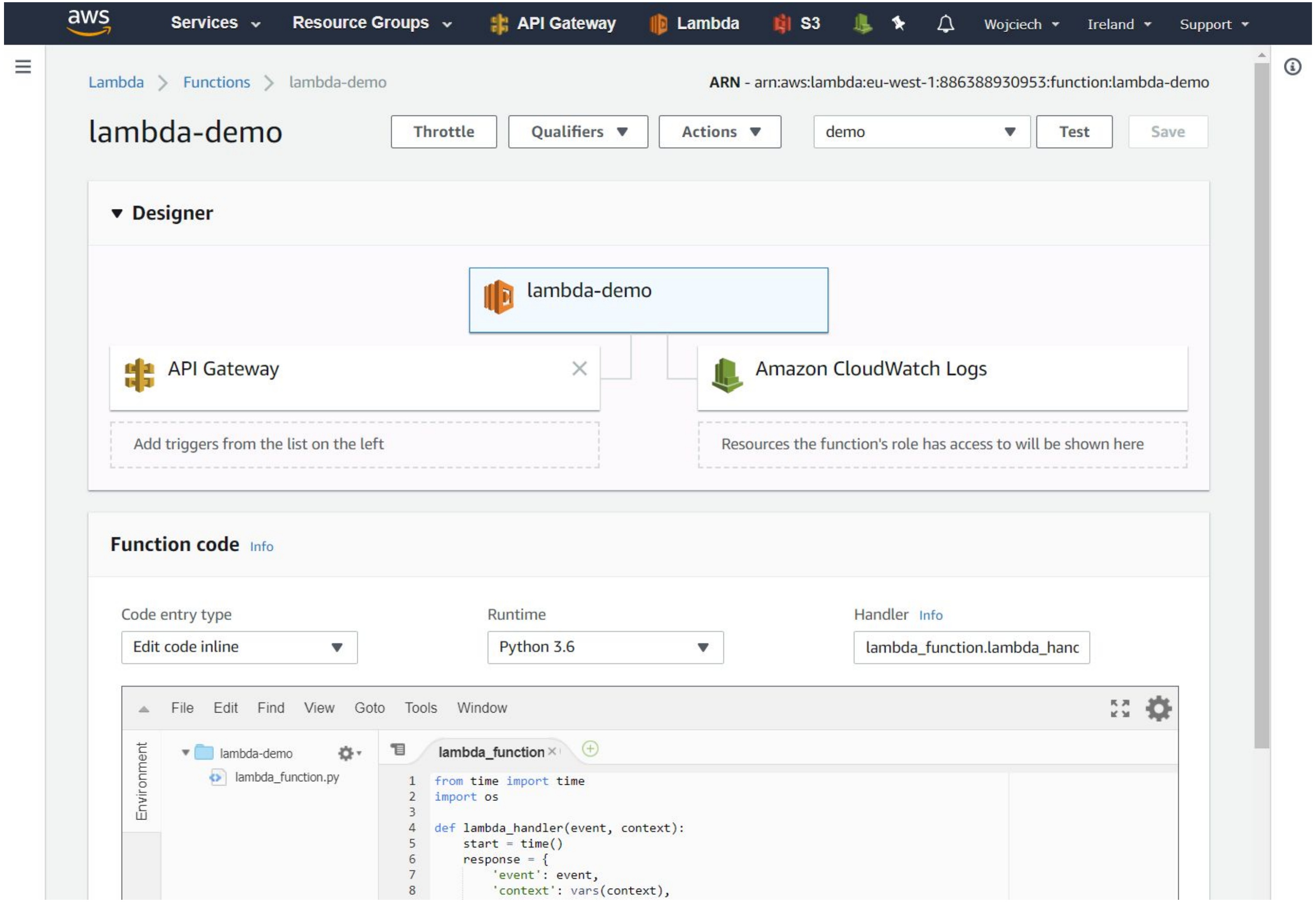
```python
from time import time
import os

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
```

AWS Lambda

```
{
  "event": {},
  "context": {
    "aws_request_id": "3cc979b0-3d8d-11e8-9deb-973978474979",
    "log_group_name": "/aws/lambda/lambda-demo",
    "log_stream_name": "2018/04/11/[$LATEST]7fd62d99a7bc48d984a4dfd68dbdcabc",
    "function_name": "lambda-demo",
    "memory_limit_in_mb": "128",
    "function_version": "$LATEST",
    "invoked_function_arn": "arn:aws:lambda:eu-west-1:886388930953:function:lambda-demo",
    "client_context": null
  },
  "environ": {
    "PATH": "/var/lang/bin:/usr/local/bin:/usr/bin/:/bin",
    "LANG": "en_US.UTF-8",
    "LAMBDA_RUNTIME_DIR": "/var/runtime",
    "AWS_REGION": "eu-west-1",
    ...
  }
}
```

AWS

Lambda



Lambda > Functions > lambda-demo

ARN - arn:aws:lambda:eu-west-1:886388930953:function:lambda-demo

lambda-demo

Designer

Add triggers from the list on the left

Resources the function's role has access to will be shown here

Function code Info

Code entry type: Edit code inline

Runtime: Python 3.6

Handler Info: lambda_function.lambda_hanc

```
from time import time
import os

def lambda_handler(event, context):
    start = time()
    response = {
        'event': event,
        'context': vars(context),
```

Serverless

AWS Lambda

Chalice

Use case

Chalice

microframework

Python Serverless Microframework for AWS

- ★ Helps in endpoint declaration
- ★ Simplifies access to HTTP request
- ★ Automatically creates IAM policy
- ★ Deployment tool
- ★ Local server
- ★ Logs viewer

Chalice

microframework

Installation

```
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

★ Run locally

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

★ Deploy

```
chalice deploy
```

```
(chalice) sargo@prv:~/workspace/demo $ chalice deploy
Creating deployment package.
Updating policy for IAM role: demo-dev
Updating lambda function: demo-dev
Creating Rest API
Resources deployed:
  - Lambda ARN: arn:aws:lambda:eu-west-1:886388930953:function:demo-dev
  - Rest API URL: https://6xa33b359a.execute-api.eu-west-1.amazonaws.com/api/
(chalice) sargo@prv:~/workspace/demo $
```

Chalice

microframework

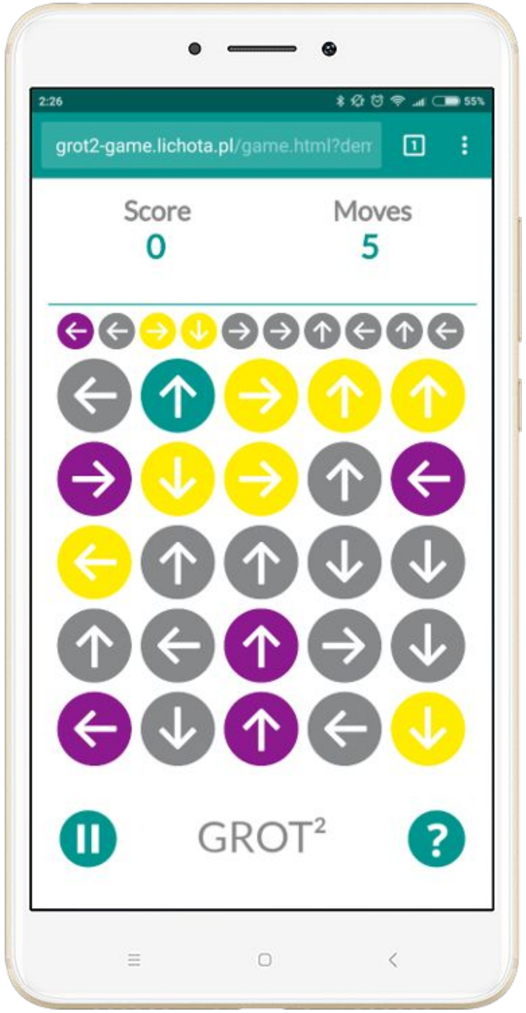
```
{
  "request": {
    "query_params": null,
    "headers": {},
    "uri_params": null,
    "method": "GET",
    "context": {
      "path": "/",
      "stage": "test-invoke-stage",
      "identity": {
        "apiKey": "test-invoke-api-key",
        ...
      "resourcePath": "/",
      "httpMethod": "GET",
      "extendedRequestId": "test-invoke-extendedRequestId",
    },
    "stage_vars": null
  },
  "environ": {
    ...
}
```

Serverless

AWS Lambda

Chalice

Use case

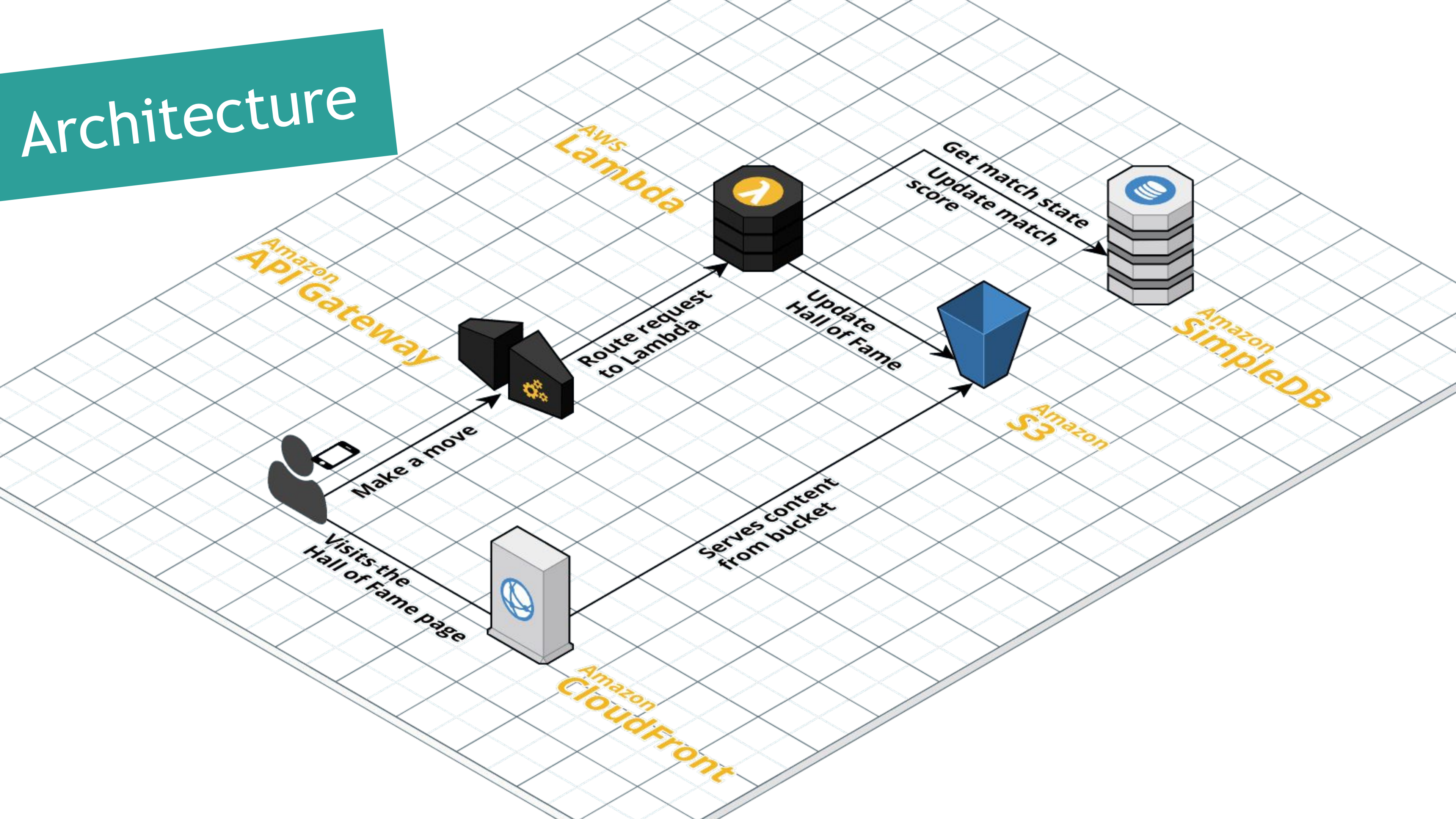


GROT² game

http://bit.ly/grot-2

https://github.com/sargo/grot2

Architecture



GROT^2

game

```
import os
from chalice import Chalice, CORSConfig, BadRequestError
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,
)
```

GROT^2

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()
```

Tips

★ Decrease communication with external services because you're paying for wait time
★ Increase RAM (increase GHz) until most of request will take less than 100ms
★ Set Usage Plan in API Gateway limit number of requests

Tips

★ Combine rarely used functions with often used ones to decrease chance of lambda warm up

★ Use CloudWatch to configure alerts and monitor execution times

★ Check Zappa (WSGI on AWS Lambda)

Summary

- ★ Chalice simplifies writing “Lambdas” and deploying them
- ★ Chalice is mainly focused on API applications based on API Gateway
- ★ Using Chalice you will became fully dependent on AWS (vendor lock-in)

THANK YOU!
STX NEXT