How to go serverless with AWS Lambda

Roman Plessl, nine (AWS Partner)

Zürich, AWSomeDay

12. September 2018

About myself and nine

Roman Plessl

Working for nine as a Solution Architect, Consultant and Leader.

As a Cloud Navigator we offer you AWS Consulting, Migrations and also Operations.

Besides nine I work as Consultant and Teacher (ETH, Foundation's)

Motivation for AWS Lambda

With AWS Lambda you don't need to worry about provisioning servers or scaling them.

Learn which workloads work best and how to go serverless in this hands-on session.

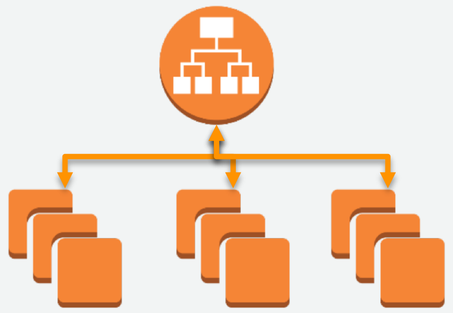
Motivation for AWS Lambda

What is Serverless Computing?

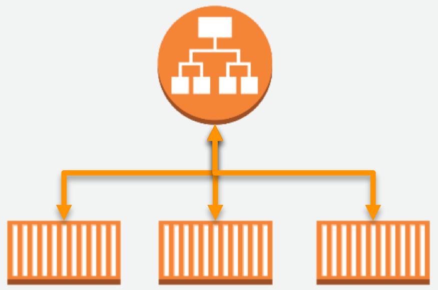
What is AWS Lambda?

Hands-on with Examples

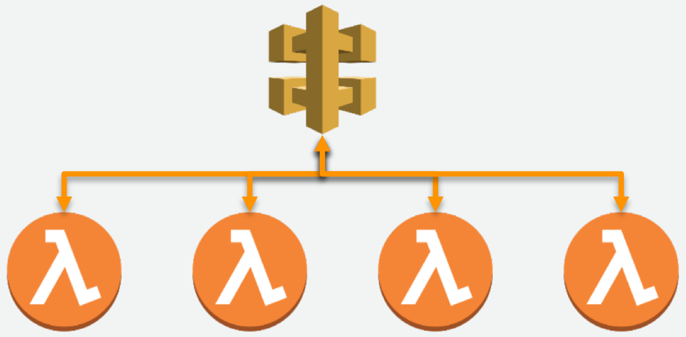
What is Serverless Computing?
Architecture Paradigms



EC2 running behind Load-Balancer



Container (ECS, EKS) running behind Load-Balancer



AWS Lambda running behind API Gateway

What is Serverless Computing?
Architecture Paradigms

EC2

- *Machine* as the unit of scale
- Virtual servers in the cloud
- I configure machines, storage, networking and OS

ECS / EKS

- *Application* as the unit of scale
- Container Management Service for running Docker containers
- Fargate (PaaS)
- I run container management servers, configure apps, control scaling

Lambda

- *Function* as the unit of scale
- Abstracts the language runtime
- I run my code when it's needed

What is Serverless Computing?
Benefit of Serverless Applications

- It’s hip, next to containerizing apps:
  (and good for staffing developers)

- Server Management:
  - No servers to manage
  - Continuous scaling
  - No idle / cold servers

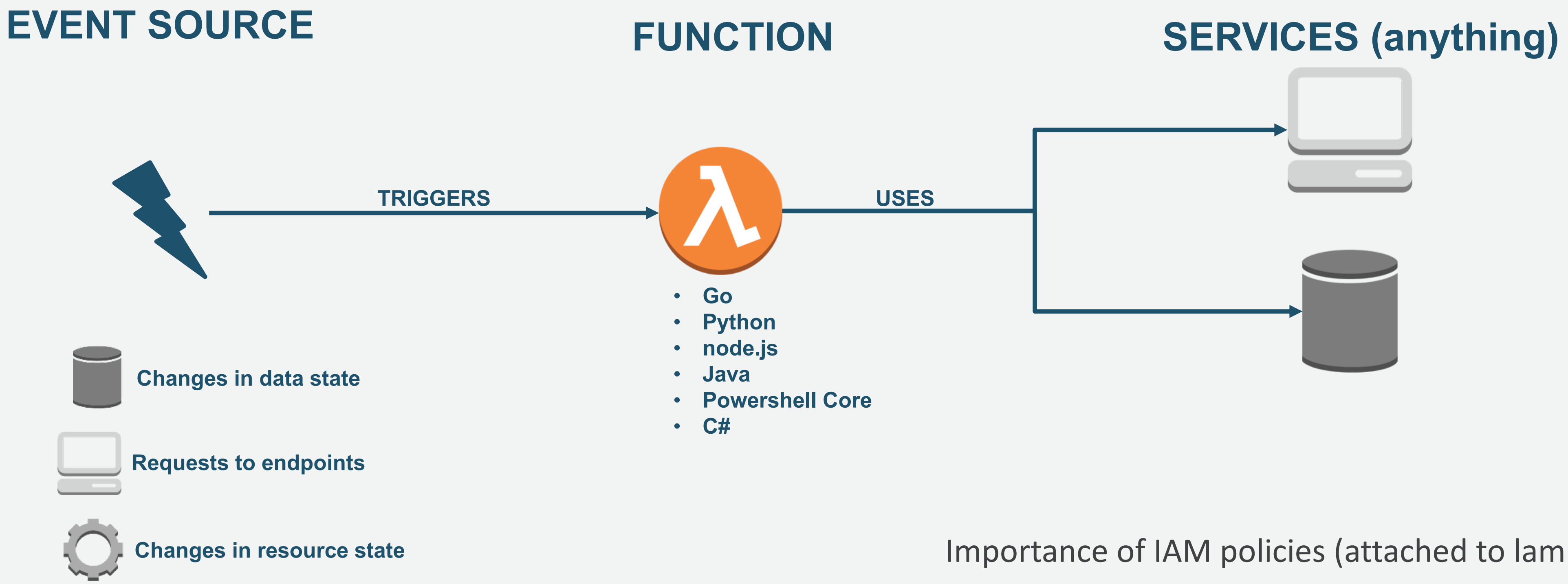
- Cost sensitive:
  - Pay per request, low charge
  - Buy compute time in mini increments
  - No hourly, daily or monthly minimums
    - No per-device fees
    - Never pay for idle

The serverless compute manifesto

https://blog.rowanudell.com/the-serverless-compute-manifesto

- 🌟 Functions are the unit of deployment and scaling.
- 🌟 No machines, VMs, or containers visible in the programming model.
- 🌟 Permanent storage lives elsewhere.
- 🌟 Scales per request; Users cannot over- or under-provision capacity.
- 🌟 Never pay for idle (no cold servers/containers or their costs).
- 🌟 Implicitly fault-tolerant because functions can run anywhere.
- 🌟 BYOC - Bring Your Own Code.
- 🌟 Metrics and logging are a universal right.

Lambda API (Sources, Functions, Services)

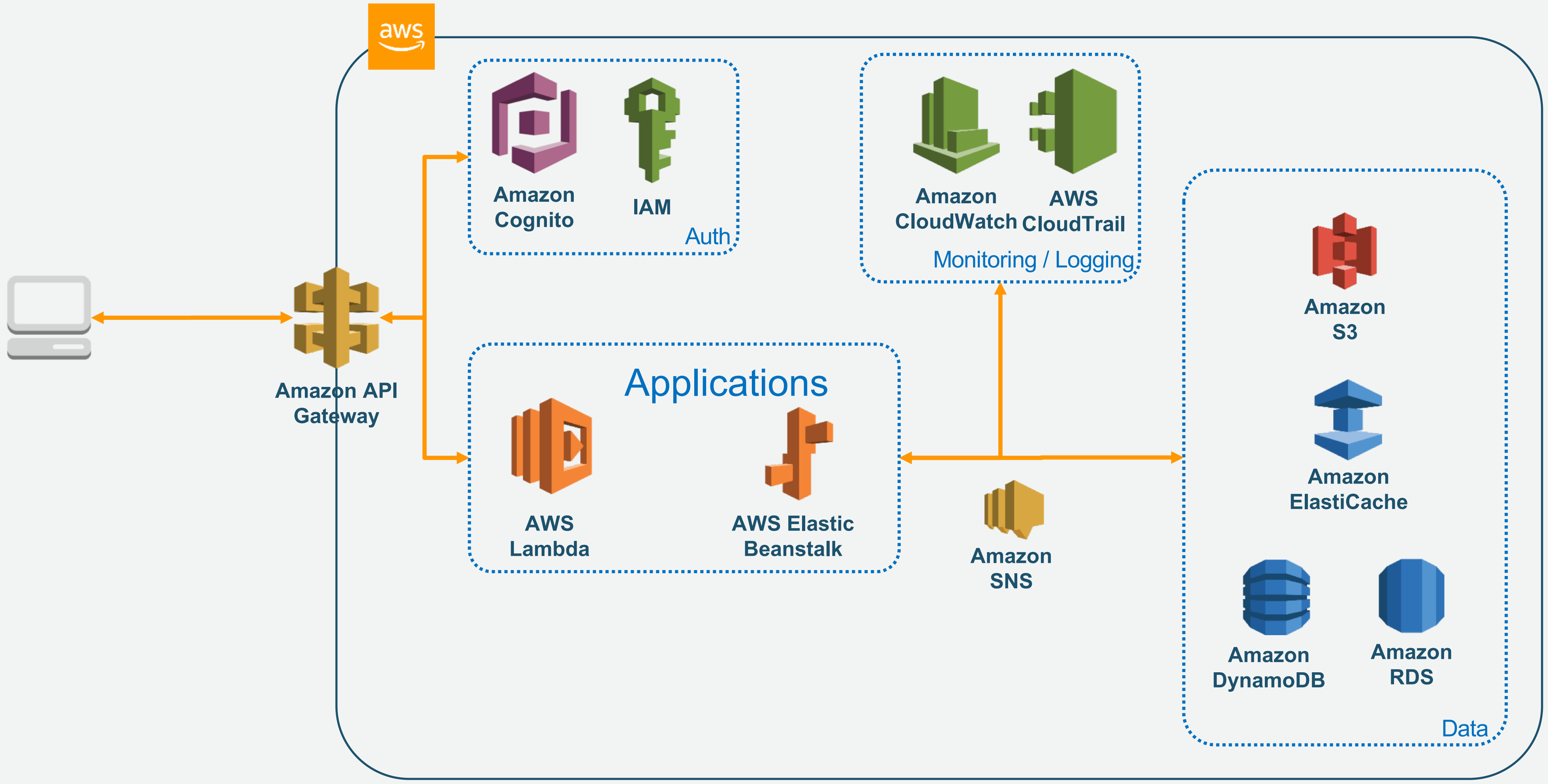


- Go
- Python
- node.js
- Java
- Powershell Core
- C#

Importance of IAM policies (attached to lambdas)

List event sources / triggers: https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html

Typical AWS Lambda Architecture



Using AWS Lambda

Bring your own code

- Multiple languages (Go, Python, Node.js, Java, Powershell Core, C#)
- Bring your own libraries

Simple resource model

- Select power rating based on memory size from 128 MB to 3000 MB
- CPU and network are allocated proportionally
- Reports / metrics actual usage

Using AWS Lambda

Flexible use

- Call or send events
- Integrated with other AWS services
- Build server less ecosystems

Flexible authorization

- Securely grant access to resources, incl. VPCs
- Fine-grained control over who can call your function (Namespaces)

Using AWS Lambda

Programming Model

- Built-in AWS SDK
- Front end is Lambda

Authoring functions

- Author directly using the Cloud9 editor (some kind with version control)
- Package code as a ZIP and upload to S3 / Lambda console

Using AWS Lambda

Stateless

- Persist Data using DynamoDB, AWS Aurora Serverless, S3 or ElastiCache (states)
- No affinity to infrastructure (can’t login to host)

Monitoring and Logging

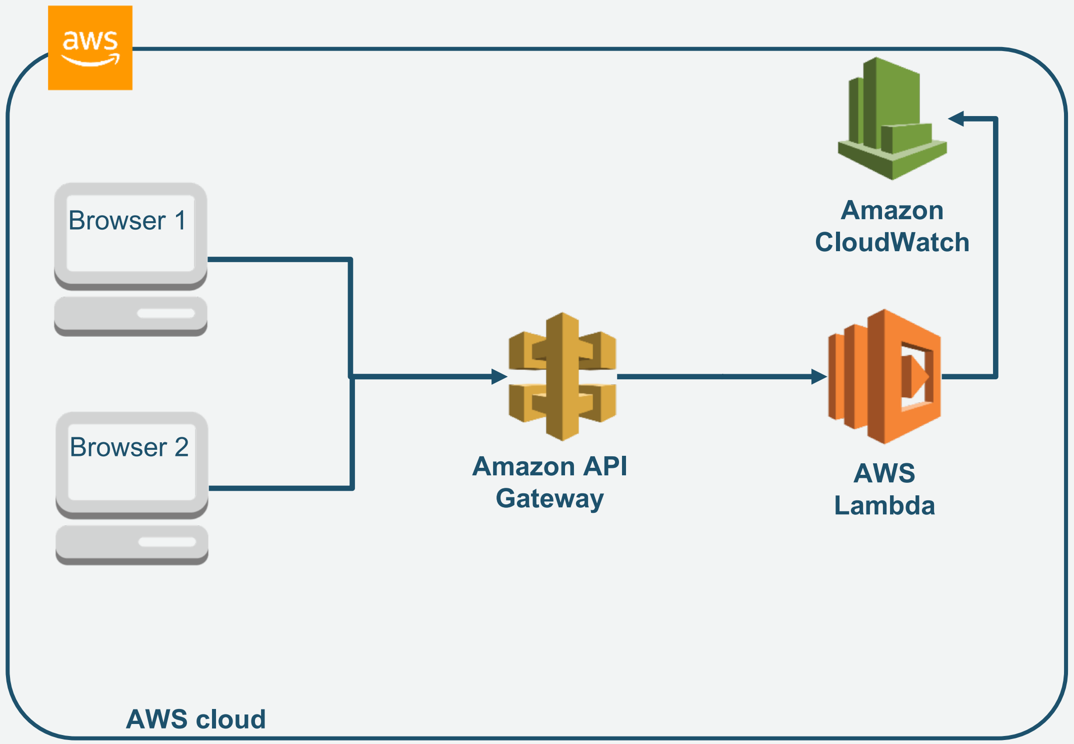
- Built in metrics for requests, latency, errors and throttles
- Built in logging with CloudWatch

AWS Lambda Pricing

- https://aws.amazon.com/lambda/pricing/
- Free tier: 1 million request, and 400’000 GB/s of compute every month, for every customer
- AWS Lambda is cost effective when compared to EC2
- Keep in mind with AWS Lambda there is no infrastructure to maintain

DEMO AND HANDS-ON TIME

Example 1 – TicTacToe
didactic approach - always start with something amusing



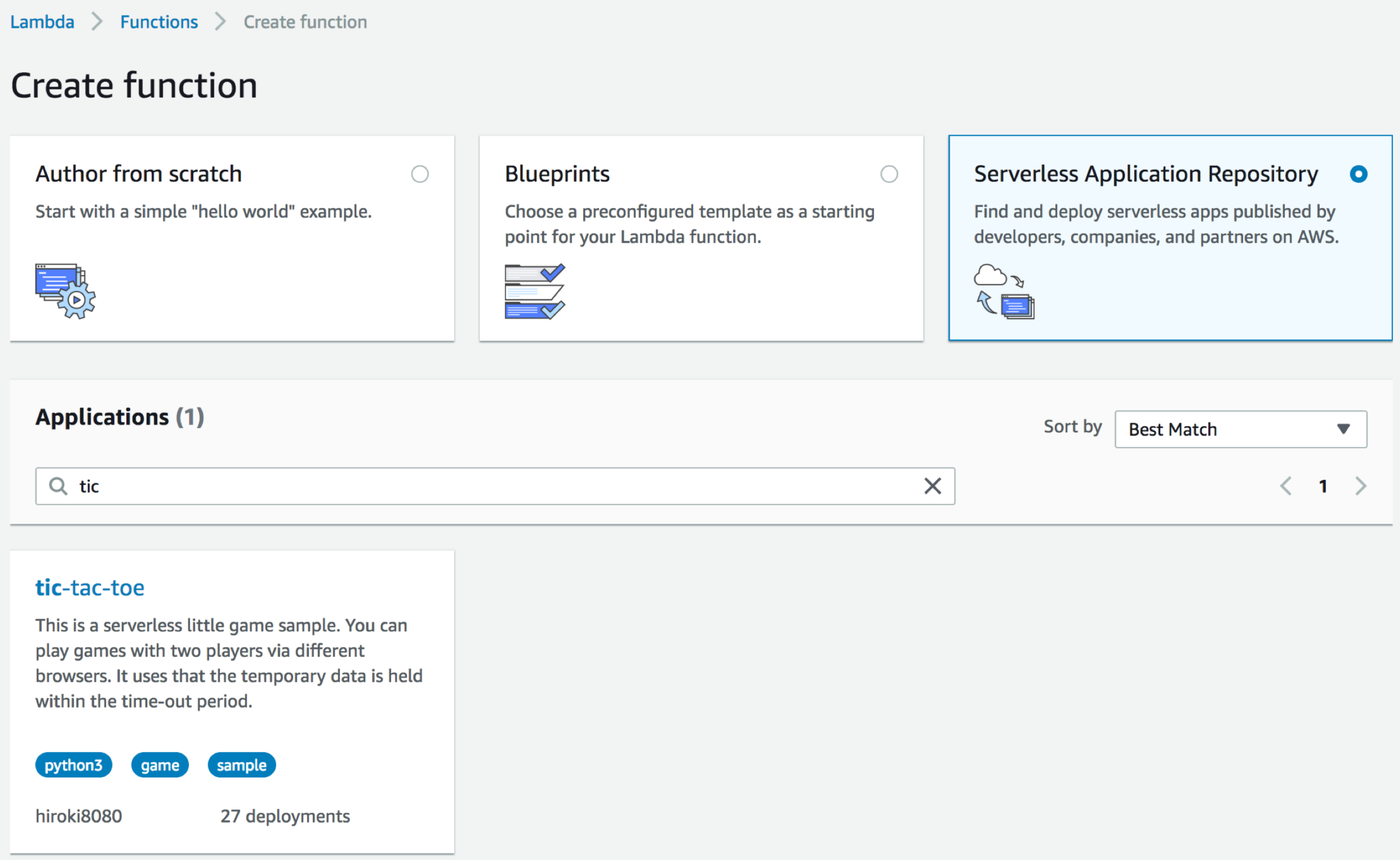
- AWS Lambda: Using in the GUI the Serverless Application Repository
- Select: *tic-tac-toe*

- *Learning how stuff works concretely from examples*

- Uses "poor-man-database" in filesystem
- Timeout is 180 seconds

Example 1 – TicTacToe

didactic approach - always start with something amusing



Lambda > Functions > Create function

Create function

Author from scratch
Start with a simple "hello world" example.

Blueprints
Choose a preconfigured template as a starting point for your Lambda function.

[✓ Serverless Application Repository]
Find and deploy serverless apps published by developers, companies, and partners on AWS.

Applications (1)

Sort by: Best Match

tic

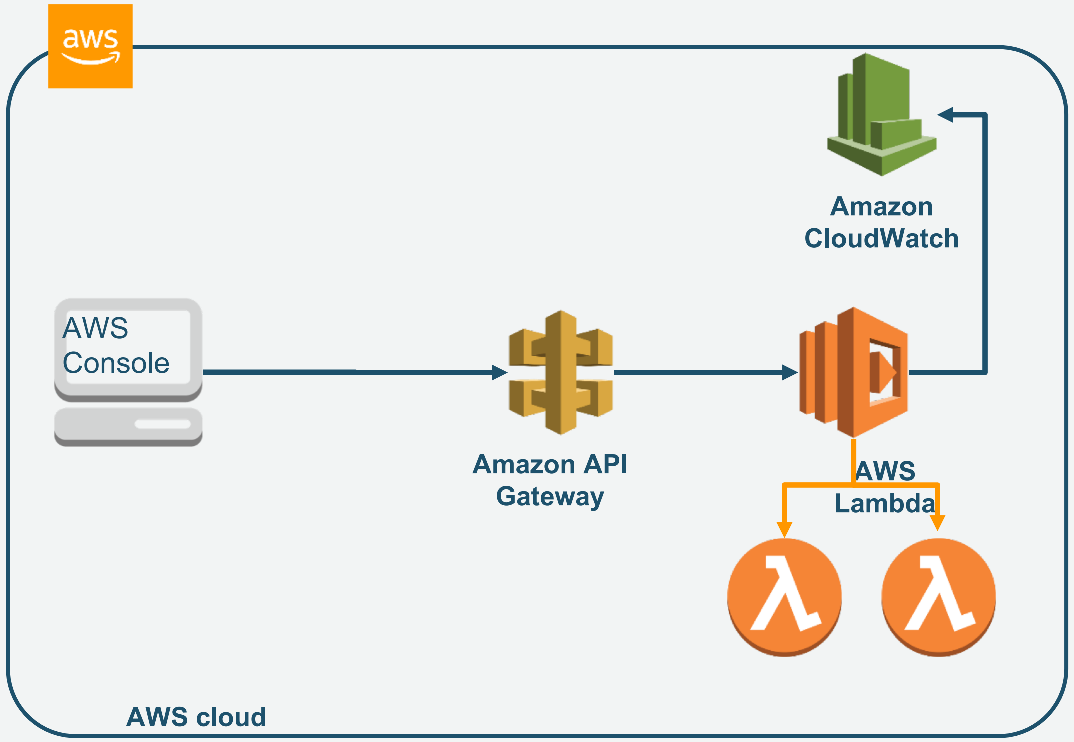
tic-tac-toe
This is a serverless little game sample. You can play games with two players via different browsers. It uses that the temporary data is held within the time-out period.

python3 game sample

hiroki8080 27 deployments

Example 2 – Hello World with Versions and Aliases

implement best practices with an easy example

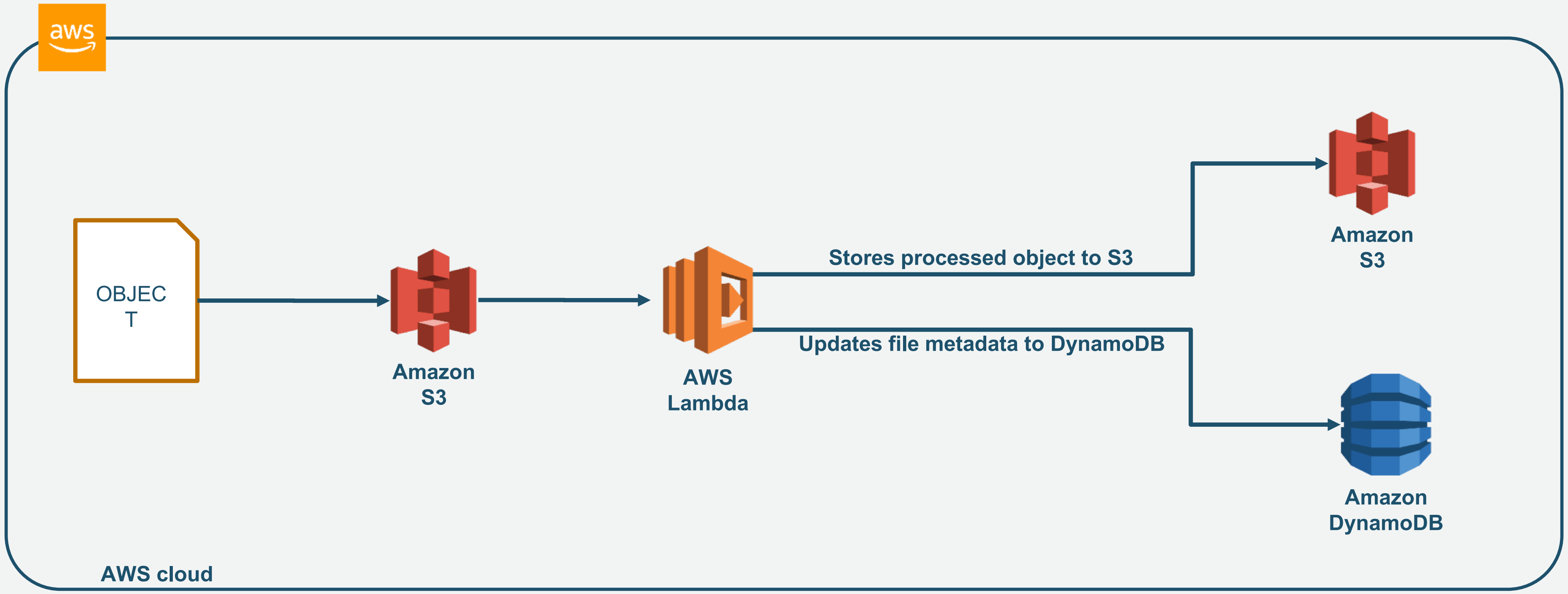


- AWS Lambda: Using in the GUI the Serverless Application Repository
- Select: *hello-world-python3*

- *Learning how to use versions and aliases*

- Run 2 adapted versions of the same function
- … and split traffic to one of those two instances

Example 3 – S3 as an Input Source
Dynamic Data Ingestion with Lambda + S3

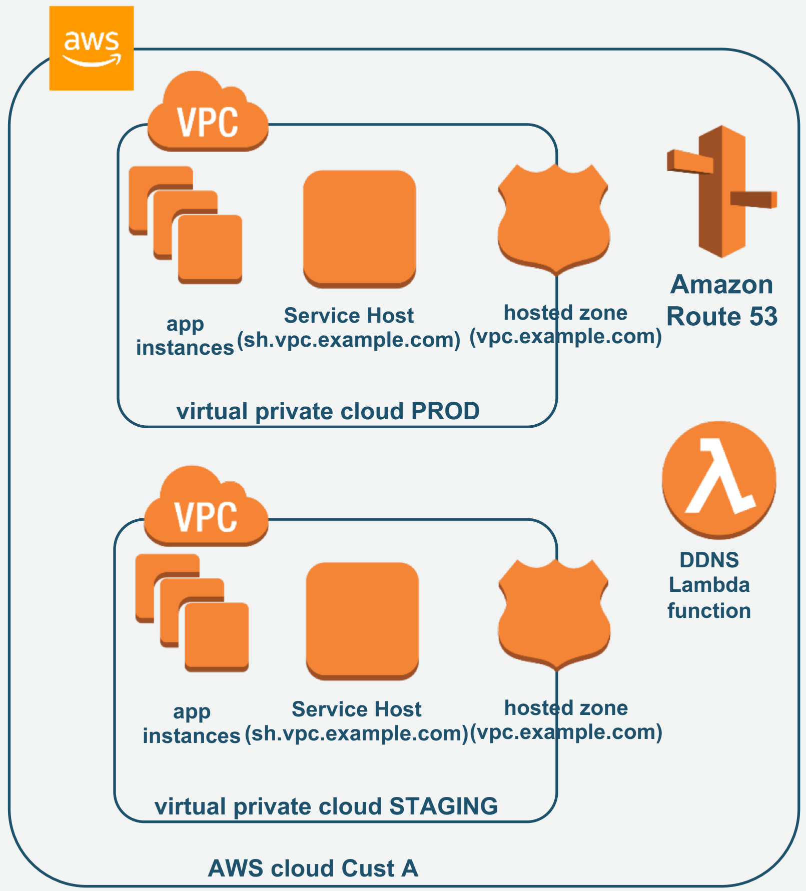


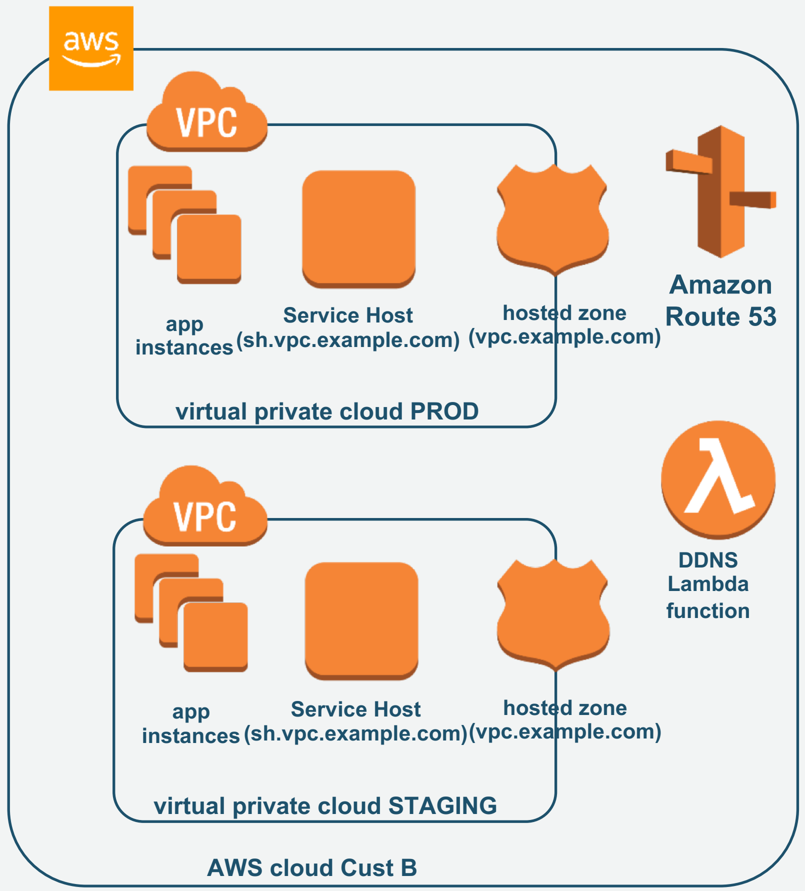
OBJECT
Amazon S3
AWS Lambda
Stores processed object to S3
Updates file metadata to DynamoDB
Amazon S3
Amazon DynamoDB
AWS cloud

Example 3 – S3 as an Input Source
Dynamic Data Ingestion with Lambda + S3

1. Create S3 Bucket (e.g. YOURNAME-lambda-s3-upload)
2. Use *s3-get-object-python3* blueprint with your bucket name
3. Upload a File with S3 Console
4. Have a look to the CloudWatch Logs

made of Lambda and EC2 instances


aws
VPC
app instances
Service Host (sh.vpc.example.com)
hosted zone (vpc.example.com)
Amazon Route 53
virtual private cloud PROD
VPC
app instances
Service Host (sh.vpc.example.com)
hosted zone (vpc.example.com)
DDNS Lambda function
virtual private cloud STAGING
AWS cloud Cust A


aws
VPC
app instances
Service Host (sh.vpc.example.com)
hosted zone (vpc.example.com)
Amazon Route 53
virtual private cloud PROD
VPC
app instances
Service Host (sh.vpc.example.com)
hosted zone (vpc.example.com)
DDNS Lambda function
virtual private cloud STAGING
AWS cloud Cust B

Solving a real life “problem”,
where the app servers
need a special common
but different gateway,
for each AWS account
and each AWS VPC context

Example 4 – Service Host local DNS resolution using DDNS made of Lambda and EC2 instances

https://github.com/rplessl/aws-lambda-ddns

```
class ServiceHostLambdaDDns:
  ZoneName = 'vpc.example.com.'

  def __init__(self, event, context):
    …
    self.r53 = ServiceHostLambdaDDnsRoute53(ServiceHostLambdaDDns.ZoneName)
    …
    if self._state() == 'running':
      ec2 = boto3.resource('ec2')

      instance = ec2.Instance(self.instance_id())

      target = instance.private_dns_name

      self.r53.add_cname('sh', target)
      self.r53.add_cname('*.sh', target)
```

Example 5 – Using serverless framework (not SAM)

```
# see also serverless.com for documentation

# install node.js
brew install node.js

# install serverless framework
npm install -g serverless

# create node.js example and
serverless create --template aws-nodejs --path example1
cd example1
serverless deploy –v
serverless deploy function -f hello
serverless invoke -f hello –l
serverless logs -f hello –t
serverless remove

# improve example, see https://github.com/rplessl/aws-serverless-examples
# with https://docs.serverless.com and https://serverless-stack.com
```

AWS Lambda Best Practices (in general)

- Use Versions and Aliases in AWS Lambda Deployments
- AWS Lambda functions should be stateless
  (focus: there will be used another sandbox)
- Using larger memory sizes will automatically provide more CPU power
- Use CloudWatch to monitor AWS Lambda request latency
- Keep startup code to a minimum
- New instances can be started any time

AWS Lambda Best Practices (Testing)

- Use pipelines for coding with automated Tests
  - Use mock's for testing
  - E.g. https://www.npmjs.com/package/aws-sdk-mock or https://github.com/spulec/moto
- Debug locally using
  - AWS SAM Local
- Use small function parts
  - Small functions with tests
  - Extract functions to libraries
- Use Sample Event Data
  - Prepare sample test events and document them

Good Resources to Learn and Practice more with AWS Lambda and Serverless

- **Wild Rydes Serverless Workshops**
  Good full blown self-teaching workshops for creating 5 different serverless use cases
  https://github.com/aws-samples/aws-serverless-workshops

- **Examples from the serverless toolkit**
  https://github.com/serverless/examples

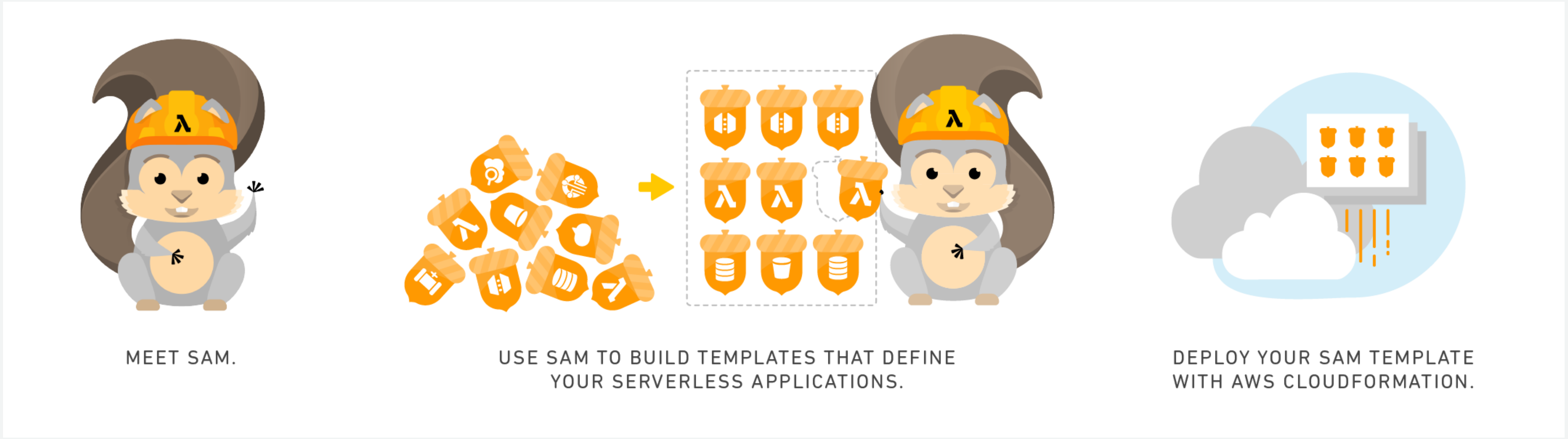
And keep in mind!

“No server is easier to manage than no server”

@Werner, Re:Invent 2016

Further Tools and Hints

Serverless Application Model



MEET SAM.

USE SAM TO BUILD TEMPLATES THAT DEFINE YOUR SERVERLESS APPLICATIONS.

DEPLOY YOUR SAM TEMPLATE WITH AWS CLOUDFORMATION.

What is the Serverless Application Model

- Typical applications consist of AWS services in addition to AWS Lambda functions

  implementation by hand can / will be error prone
- SAM
  - uses CloudFormation to define all components of a serverless application and architecture
  - supports a simplified syntax for serverless resources (Lambda, API Gateway, DynamoDB)
  - supports the resources CloudFormation supports.
- blueprints in the AWS lambda console

AWS Step functions

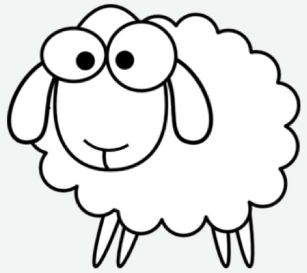
When running complex functions the following must-be or questions arises:

- I will have sequential flow(s) of my functions
- I want to run functions in parallel
- I will use a third party app on a EC2 or a container
- I have code that runs for hours

AWS Step functions makes it easy to coordinate components of distributed applications using a visual workflow.

Step function task can be on AWS Lambda, ECS or EC2.

Further Tools and Integrations



- **lambCI**
  Serverless continuous integration (like Travis or CircleCI)
  https://github.com/lambci/lambci
  including a CloudFormation Launch Stack

- **lambda-packages**
  "How to integrate libraries with C bindings like python?..."
  Various Python libraries precompiled and selected for using with AWS Lambda
  https://github.com/Miserlou/lambda-packages

Thank you!

Roman Plessl
github.com/rplessl

Reach us

nine, AWS Partner
www.nine.ch/en/aws