universität innsbruck

Serverless Computing – Cheap, Easy and Powerful

DIH-WEST

Sashko Ristov

Agenda

- Introduction
  - About me
  - Introduction for the workshop
- FaaS
  - Task 1: Create a Simple Function
- BaaS Services
  - Task 2: Recognize faces from images
- Infrastructure-as-a-Code
  - Task 3: Deploy a function with Terraform
- Automation
  - Task 4: GitHub Actions + Terraform

Positions

- (June 2022 -) Assistant Professor (QE)
- 6 years Postdoctoral University Assistant (DPS)
- 3 years Assistant Professor (Skopje, N. Macedonia)
- 4 years Teaching and Research Assistant (Skopje, N. Macedonia)
- 8 years in Macedonian Telecom

Publications in 2022

- Journals / Magazines $\times 4$
  - **S. Ristov**, D. Kimovski, T. Fahringer, "FaaScinating Resilience for Serverless Function Choreographies in Federated Clouds," in *IEEE Trans. on Network and Service Management*, 2022.
  - **S. Ristov**, S. Pedratscher, T. Fahringer, “xAFCL: Run scalable function choreographies across multiple FaaS systems,” in *IEEE Transactions on Services Computing*, pp. 1–1, 2022
  - S. Pedratscher, **S. Ristov**, T. Fahringer, "M2FaaS: Transparent and Fault Tolerant FaaSification of Node.js Monolith Code Blocks", in *Elsevier Fut. Gen. Comp. Sys.*, Accepted on Apr. 19, 2022.
  - D. Kimovski, **S. Ristov**, R. Prodan, "Decentralized Machine Learning for Intelligent Health Care Systems on the Computing Continuum", in *IEEE Computer*, Accepted Jan. 2022.
- Top Conferences $\times 3$
  - **S. Ristov**, M. Hautz, C. Hollaus, R. Prodan. “Simulate Serverless Workflows and Their Twins and Siblings in Federated FaaS,” *ACM Symp. on Cloud Computing* (*SoCC*’22), San Francisco, Nov. 2022
  - **S. Ristov** and P. Gritsch. "FaaSt: Optimize makespan of serverless workflows in federated commercial FaaS," (*IEEE CLUSTER ’22*). Heidelberg, Germany, 182–194, 2022.
  - **S. Ristov**, S. Brandacher, M. Felderer, R. Breu, “GoDeploy: Portable Deployment of Serverless Functions in Federated FaaS”, *IEEE Cloud Summit 2022*, Fairfax, Virginia, USA, Oct. 2022
- Workshops $\times 2$
  - **S. Ristov**, C. Hollaus, and M. Hautz. "Colder Than the Warm Start and Warmer Than the Cold Start! Experience the Spawn Start in FaaS Providers," in *ApPLIED ’*22). Salerno, Italy, 35–39, 2022.
  - M. Gusev, **S. Ristov**, et al. „CardioHPC: Serverless Approaches for Real-Time Heart Monitoring of Thousands of Patients“, *WORKS’*22 Workshop (Supercomputing) , Dallas, USA, Nov. 2022

Serafin Plattner

- Bachelor student
- Bachelor thesis in serverless
- pyfOps: A pipeline for one-touch FaaSification, deployment, and testing of Python serverless functions
- On demand, we can show his pipeline, which integrates FaaSification and tests

Organization (Social)

- Show agenda on WEB
- Refreshments
- Two Breaks

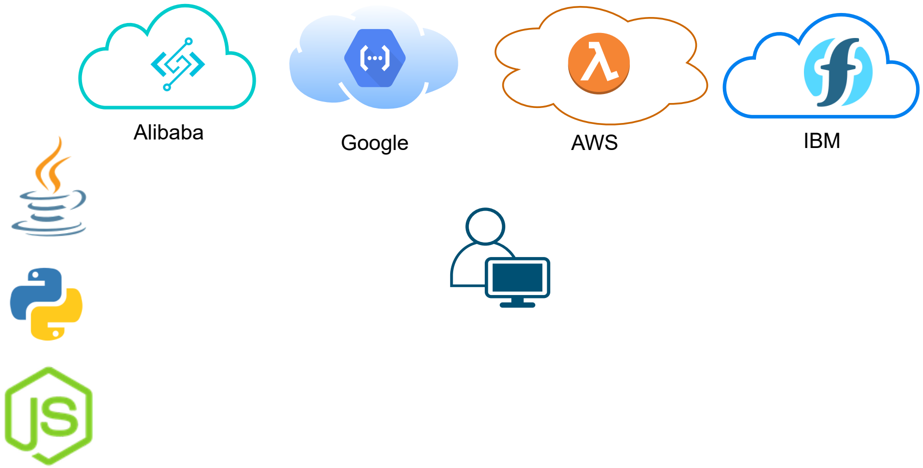
Organization (Technical)

- FaaS familiarity?!
- Fork the repository `https://github.com/FaaSTools/faas-tutorial`
- Install Terraform
- AWS Account

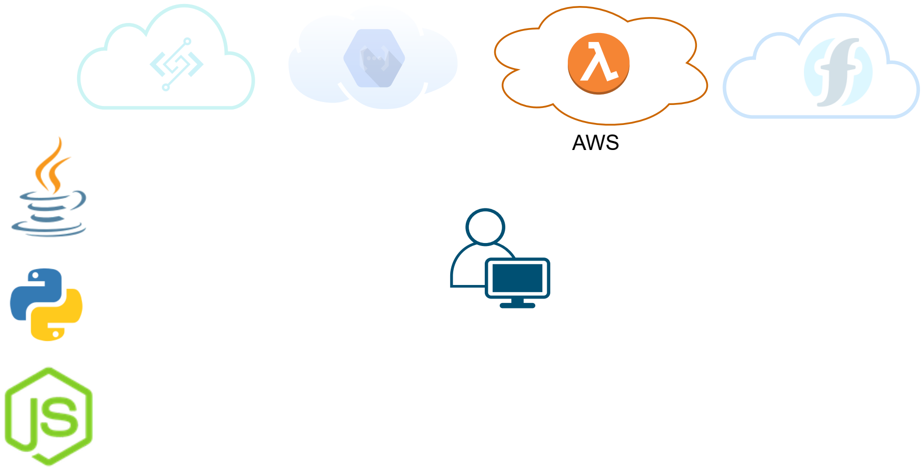
Agenda

- Introduction
  - About me
  - Introduction for the workshop
- FaaS
  - Task 1: Create a Simple Function
- BaaS Services
  - Task 2: Recognize faces from images
- Infrastructure-as-a-Code
  - Task 3: Deploy a function with Terraform
- Automation
  - Task 4: GitHub Actions + Terraform

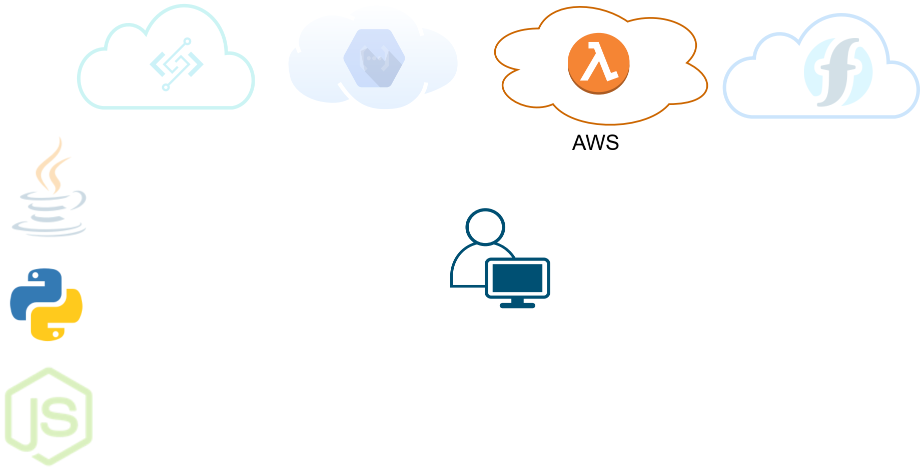
FaaS Options



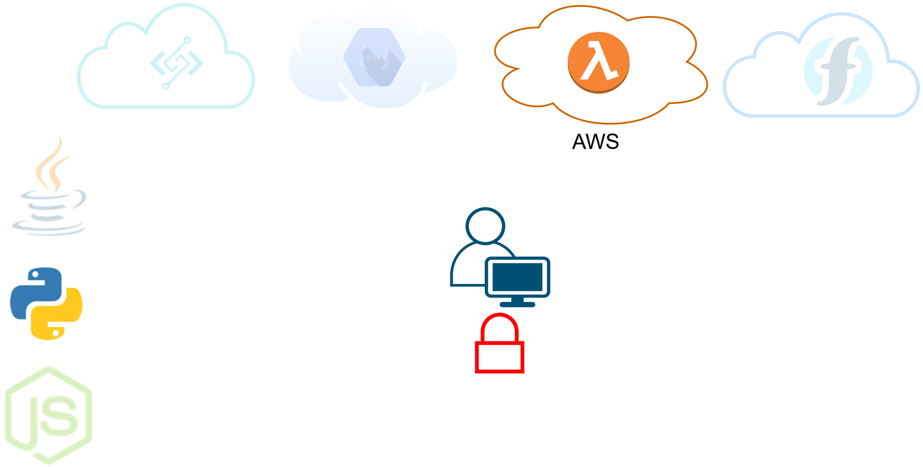
FaaS Options



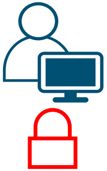
FaaS Options



FaaS Options



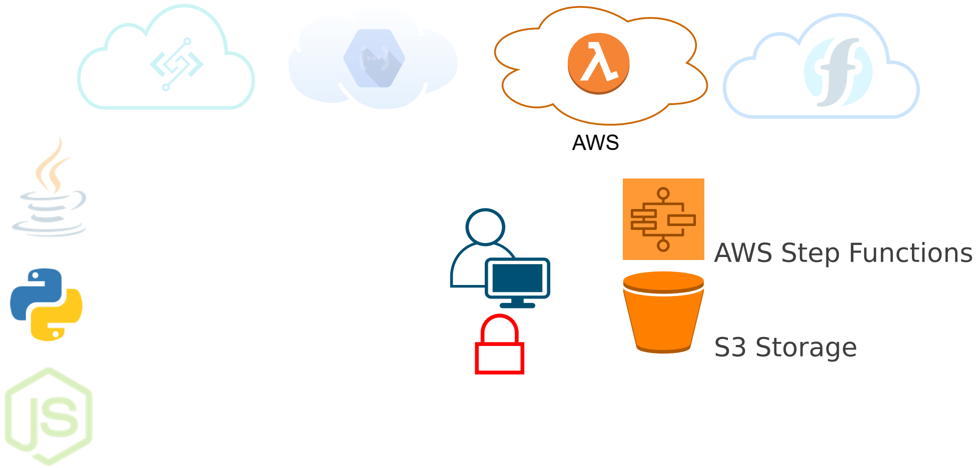
FaaS Options



AWS

AWS Step Functions

FaaS Options



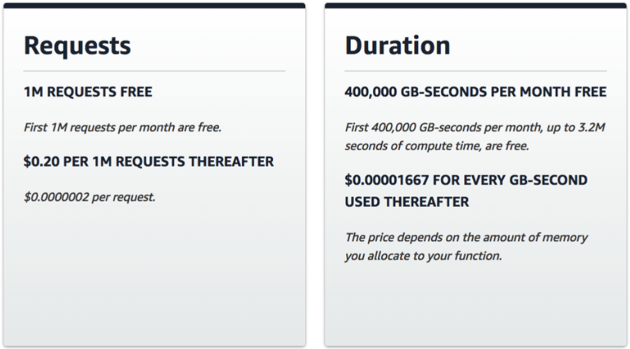
Pay as you go

- **Upload** a function **in any language**
- NodeJS, Swift, Java, Go, Scala, Python, PHP, Ruby
- **Other** programming languages
  - Prepare your **own container**

Charging?

- Fuel
  - Up to the 1/1000 (third digit after the "comma")

Charges for AWS Lambda



Requests

1M REQUESTS FREE

First 1M requests per month are free.

$0.20 PER 1M REQUESTS THEREAFTER

$0.0000002 per request.

Duration

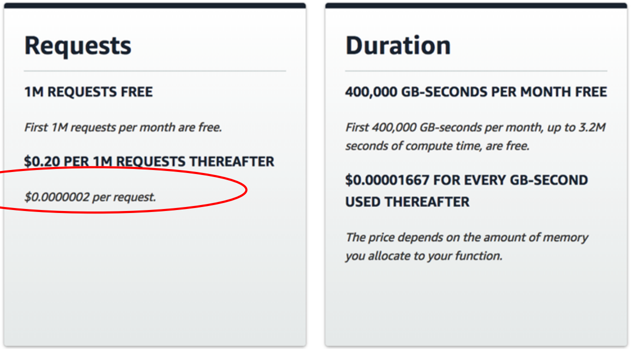
400,000 GB-SECONDS PER MONTH FREE

First 400,000 GB-seconds per month, up to 3.2M seconds of compute time, are free.

$0.00001667 FOR EVERY GB-SECOND USED THEREAFTER

The price depends on the amount of memory you allocate to your function.

Charges for AWS Lambda



Requests

1M REQUESTS FREE

First 1M requests per month are free.

$0.20 PER 1M REQUESTS THEREAFTER

$0.0000002 per request.

Duration

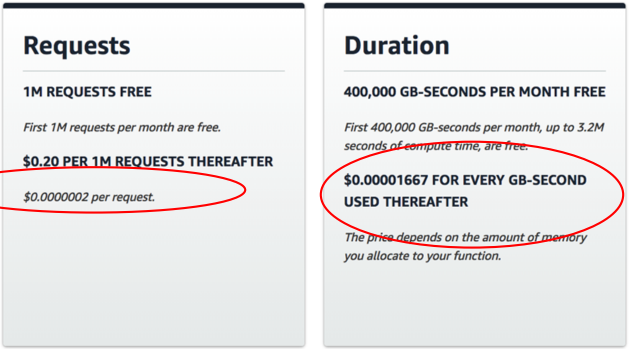
400,000 GB-SECONDS PER MONTH FREE

First 400,000 GB-seconds per month, up to 3.2M seconds of compute time, are free.

$0.00001667 FOR EVERY GB-SECOND USED THEREAFTER

The price depends on the amount of memory you allocate to your function.

Charges for AWS Lambda



Requests

1M REQUESTS FREE

First 1M requests per month are free.

$0.20 PER 1M REQUESTS THEREAFTER

$0.0000002 per request.

Duration

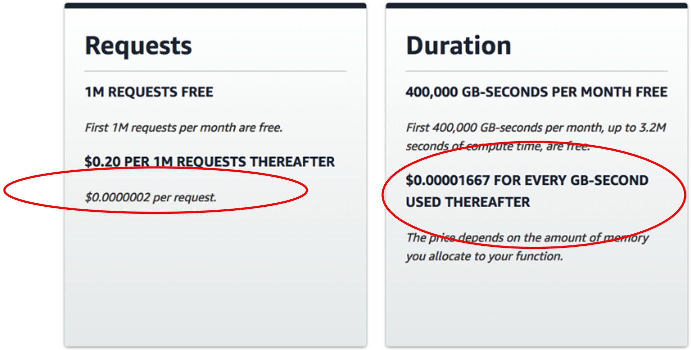
400,000 GB-SECONDS PER MONTH FREE

First 400,000 GB-seconds per month, up to 3.2M seconds of compute time, are free.

$0.00001667 FOR EVERY GB-SECOND USED THEREAFTER

The price depends on the amount of memory you allocate to your function.

Charges for AWS Lambda



Requests

1M REQUESTS FREE

First 1M requests per month are free.

$0.20 PER 1M REQUESTS THEREAFTER

$0.0000002 per request.

Duration

400,000 GB-SECONDS PER MONTH FREE

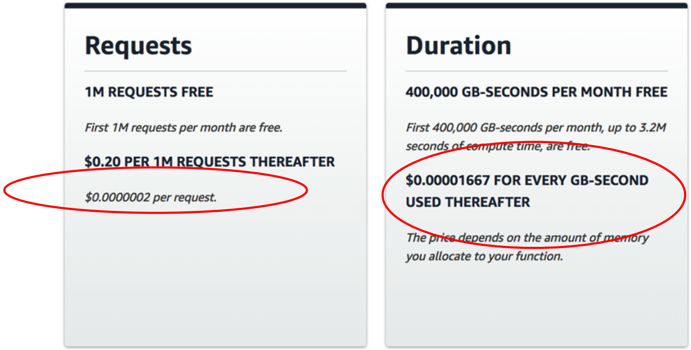
First 400,000 GB-seconds per month, up to 3.2M seconds of compute time, are free.

$0.00001667 FOR EVERY GB-SECOND USED THEREAFTER

The price depends on the amount of memory you allocate to your function.

Free tiers both for requests and duration/resources

Charges for AWS Lambda



Requests

1M REQUESTS FREE

First 1M requests per month are free.

$0.20 PER 1M REQUESTS THEREAFTER

$0.0000002 per request.

Duration

400,000 GB-SECONDS PER MONTH FREE

First 400,000 GB-seconds per month, up to 3.2M seconds of compute time, are free.

$0.00001667 FOR EVERY GB-SECOND USED THEREAFTER

The price depends on the amount of memory you allocate to your function.

Free tiers both for requests and duration/resources

Some FaaS providers do not charge invocations!

How to Code?

- Handler method
- Read keys from the JSON input data `event['key']`
- Code the functions
- Return the output `returnJSON`
- Show AWS Console

How to Deploy?

- Handler method format
- Region
- Memory
- Timeout
- Runtime Environment
- Role
- Show AWS Console

How to Run?

- Test input
- Cold/Warm start
- Timeout
- Runtime Environment
- Show AWS Console

Agenda

- Introduction
  - About me
  - Introduction for the workshop
- FaaS
  - Task 1: Create a Simple Function
- BaaS Services
  - Task 2: Recognize faces from images
- Infrastructure-as-a-Code
  - Task 3: Deploy a function with Terraform
- Automation
  - Task 4: GitHub Actions + Terraform

How to Code?

- Boto3 for Python
- `boto3.resource('s3')`
- `boto3.client('rekognition')`
- layers
- Deployment package is ZIP

Agenda

- Introduction
  - About me
  - Introduction for the workshop
- FaaS
  - Task 1: Create a Simple Function
- BaaS Services
  - Task 2: Recognize faces from images
- Infrastructure-as-a-Code
  - Task 3: Deploy a function with Terraform
- Automation
  - Task 4: GitHub Actions + Terraform

Infrastructure as a Code

- Automate and manage
  - infrastructures
  - platforms
  - services

Declarative vs. Imperative Approaches

- Declarative approach (IDEMPOTENT) - **Desired state** :
  - 20 servers
  - 2 FW
  - 3 VPCs

- Imperative approach - give **instructions**:
  - add 2 servers in AWS North Virginia
  - remove 2 servers in IBM Tokyo
  - add a FW
  - remove a security group

Infrastructure Provisioning / Coding Tools / Standard

- Terraform
- Open Tosca
- AWS Cloud Formation (provider specific)
- AWS SAM (provider specific)
- Serverless.com

Terraform

- tool for infrastructure provisioning, and platform and service deployment
- open source
- multi provider support

Terraform

- tool for infrastructure provisioning, and platform and service deployment
- open source
- multi provider support
- declarative approach

Terraform Use Cases

- Provision the environment
- Update the environment
- Replicate environment
  - From DEVELOPMENT to TESTING or PRODUCTION

Terraform Architecture

- Inputs
  - TF-config (.tf)
    - What to create / configure
  - TF-State

- Core (how to reach the desired state from the current one?)
  - compares the config and the state
  - what needs to be deployed / updated / destroyed

- Providers
  - IaaS (AWS, IBM, ...)
  - PaaS (Kubernetes)
  - Serverless (FaaS)

Terraform Actions

- `refresh`
  - query the provider about the current state
- create an execution plan (`init`)
  - which actions are necessary to be taken (no execution yet)
- `apply`
  - execute the plan
- `destroy`
  - destroy the infrastructure according to the plan (opposite from `apply`)

Agenda

- Introduction
  - About me
  - Introduction for the workshop
- FaaS
  - Task 1: Create a Simple Function
- BaaS Services
  - Task 2: Recognize faces from images
- Infrastructure-as-a-Code
  - Task 3: Deploy a function with Terraform
- Automation
  - Task 4: GitHub Actions + Terraform

GitHub Actions

- A pipeline to automatize actions after some push/pull on GitHub
- YAML-based script
- create `.github/workflows/main.yaml`
- https://docs.github.com/en/actions/using-workflows/workflow-syntax-for-github-actions

GitHub Actions Example

Open Actions -> Python application

QUESTIONS?

Email: sashko.ristov@uibk.ac.at