Marcin Copik, Alexandru Calotoiu, Torsten Hoefler, Michał Podstawski, Laurin Brandner, Larissa Schmid, Nico Graf, Grzegorz Kwaśniewski, Kacper Janda, Mateusz Knapik, Jakub Czerski, Mahla Sarifi, Paweł Żuk, Sascha Kehrli, Abhishek Kumar, Prajin Khadka, Horia Mercan, and many others!

Demystifying Serverless Performance: A Hands-on Tutorial with Serverless Benchmark Suite SeBS

SPCL

HiPEAC 2026
Cracow

Monday, January 26th, 14:00
Room: S12

Goals of the Tutorial

Goals of the Tutorial

✓ **Learn what serverless is and isn’t – no prior knowledge is required.**

Goals of the Tutorial

- ✓ **Learn what serverless is and isn’t – no prior knowledge is required.**
- ✓ **Understand differences, capabilities and limitations of FaaS platforms.**

Goals of the Tutorial

- ✓ **Learn what serverless is and isn’t – no prior knowledge is required.**
- ✓ **Understand differences, capabilities and limitations of FaaS platforms.**
- ✓ **Get up-to-date with the current frontier of research in serverless.**

Goals of the Tutorial

- ✓ **Learn what serverless is and isn't – no prior knowledge is required.**
- ✓ **Understand differences, capabilities and limitations of FaaS platforms.**
- ✓ **Get up-to-date with the current frontier of research in serverless.**
- ✓ **Learn a benchmarking tool for serverless platforms.**

Goals of the Tutorial

SeBS Slack

- ✓ Learn what serverless is and isn’t – no prior knowledge is required.
- ✓ Understand differences, capabilities and limitations of FaaS platforms.
- ✓ Get up-to-date with the current frontier of research in serverless.
- ✓ Learn a benchmarking tool for serverless platforms.

slack Join SeBS Slack channel: https://serverlessbenchmark.slack.com

Agenda

SeBS Slack

slack Join SeBS Slack channel: https://serverlessbenchmark.slack.com

Agenda

SeBS Slack

- ✓ Part I
  - ✓ What is Serverless?
  - ✓ Benchmarking Suite SeBS
  - ✓ Working with SeBS

slack **Join SeBS Slack channel: https://serverlessbenchmark.slack.com**

Agenda

SeBS Slack

- ✓ Part I
  - ✓ What is Serverless?
  - ✓ Benchmarking Suite SeBS
  - ✓ Working with SeBS
- ✓ Hands-on I: Local Deployment & Storage

slack Join SeBS Slack channel: https://serverlessbenchmark.slack.com

Agenda

- ✓ Part I
  - ✓ What is Serverless?
  - ✓ Benchmarking Suite SeBS
  - ✓ Working with SeBS
- ✓ Hands-on I: Local Deployment & Storage
- ✓ Part II
  - ✓ Communication and Data
  - ✓ Serverless Workflows
  - ✓ Experiments in SeBS

SeBS Slack

slack Join SeBS Slack channel: https://serverlessbenchmark.slack.com

Agenda

- ✓ Part I
  - ✓ What is Serverless?
  - ✓ Benchmarking Suite SeBS
  - ✓ Working with SeBS
- ✓ Hands-on I: Local Deployment & Storage
- ✓ Part II
  - ✓ Communication and Data
  - ✓ Serverless Workflows
  - ✓ Experiments in SeBS
- ✓ Hands-on II: FaaS Platforms & Experiments

SeBS Slack

slack Join SeBS Slack channel: https://serverlessbenchmark.slack.com

Agenda

SeBS Slack

- ✓ Part I
  - ✓ What is Serverless?
  - ✓ Benchmarking Suite SeBS
  - ✓ Working with SeBS
- ✓ Hands-on I: Local Deployment & Storage
- ✓ Part II
  - ✓ Communication and Data
  - ✓ Serverless Workflows
  - ✓ Experiments in SeBS
- ✓ Hands-on II: FaaS Platforms & Experiments
- ✓ Part III
  - ✓ Research Directions in Serverless
  - ✓ Development of SeBS

slack

Join SeBS Slack channel: https://serverlessbenchmark.slack.com

Agenda

- ✓ Part I
  - ✓ What is Serverless?
  - ✓ Benchmarking Suite SeBS
  - ✓ Working with SeBS
- ✓ Hands-on I: Local Deployment & Storage
- ✓ Part II
  - ✓ Communication and Data
  - ✓ Serverless Workflows
  - ✓ Experiments in SeBS
- ✓ Hands-on II: FaaS Platforms & Experiments
- ✓ Part III
  - ✓ Research Directions in Serverless
  - ✓ Development of SeBS

SeBS Slack

slack

Join SeBS Slack channel: https://serverlessbenchmark.slack.com

Does Serverless Have Servers?

> serverless architecture
> *looks inside*
> servers

Does Serverless Have Servers?

> serverless architecture
> *looks inside*
> servers

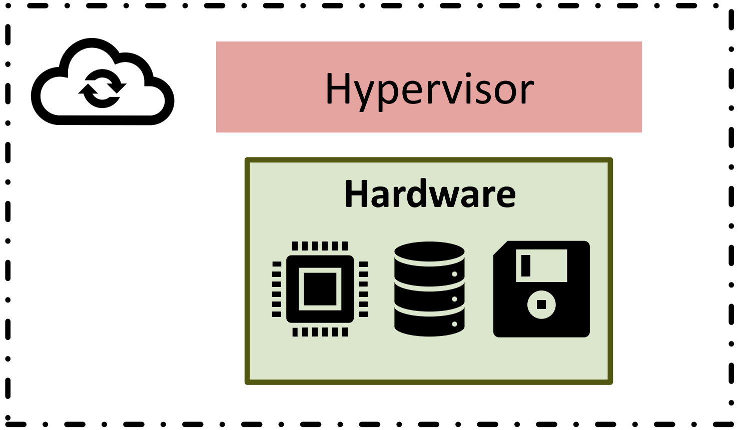
Does Serverless Have Servers?

> serverless architecture
> *looks inside*
> servers

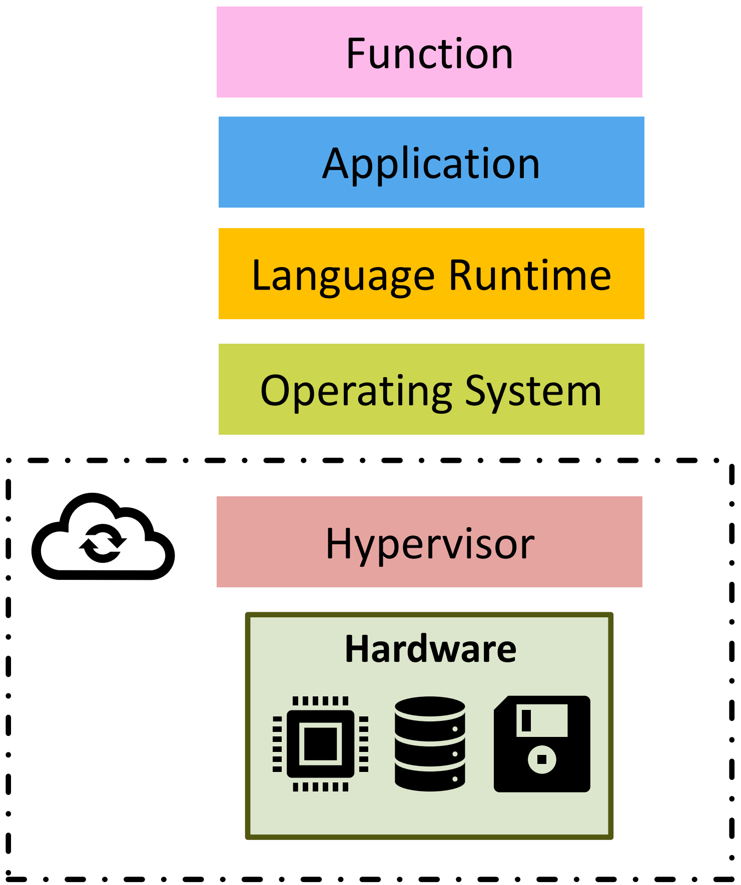
WAIT, SO THERE IS NO SERVER?
NEVER HAS BEEN ONE
imgflip.com

Cloud and Serverless

Cloud and Serverless

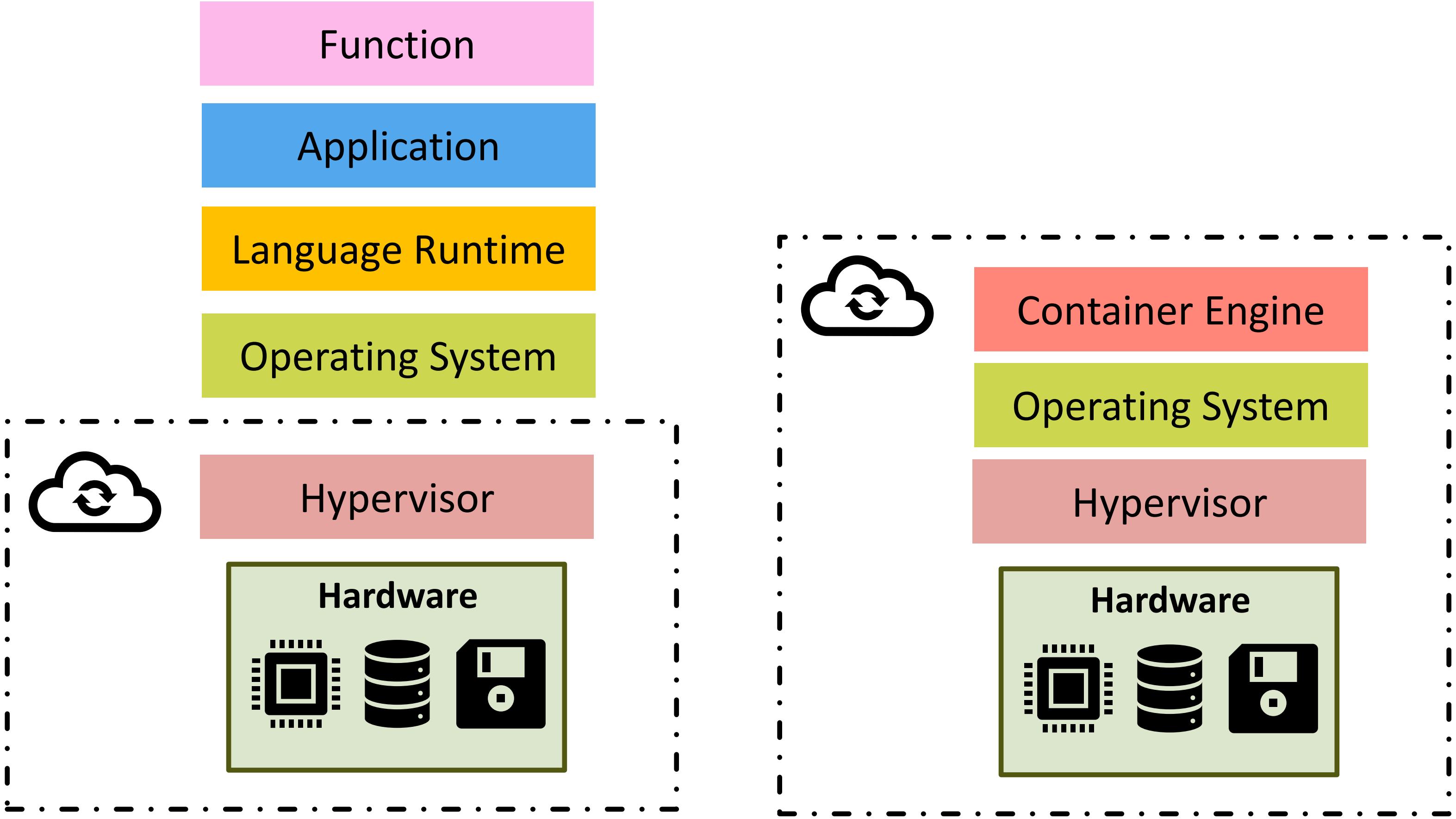


Cloud and Serverless



Virtual Machine (Infrastructure-as-a-Service)

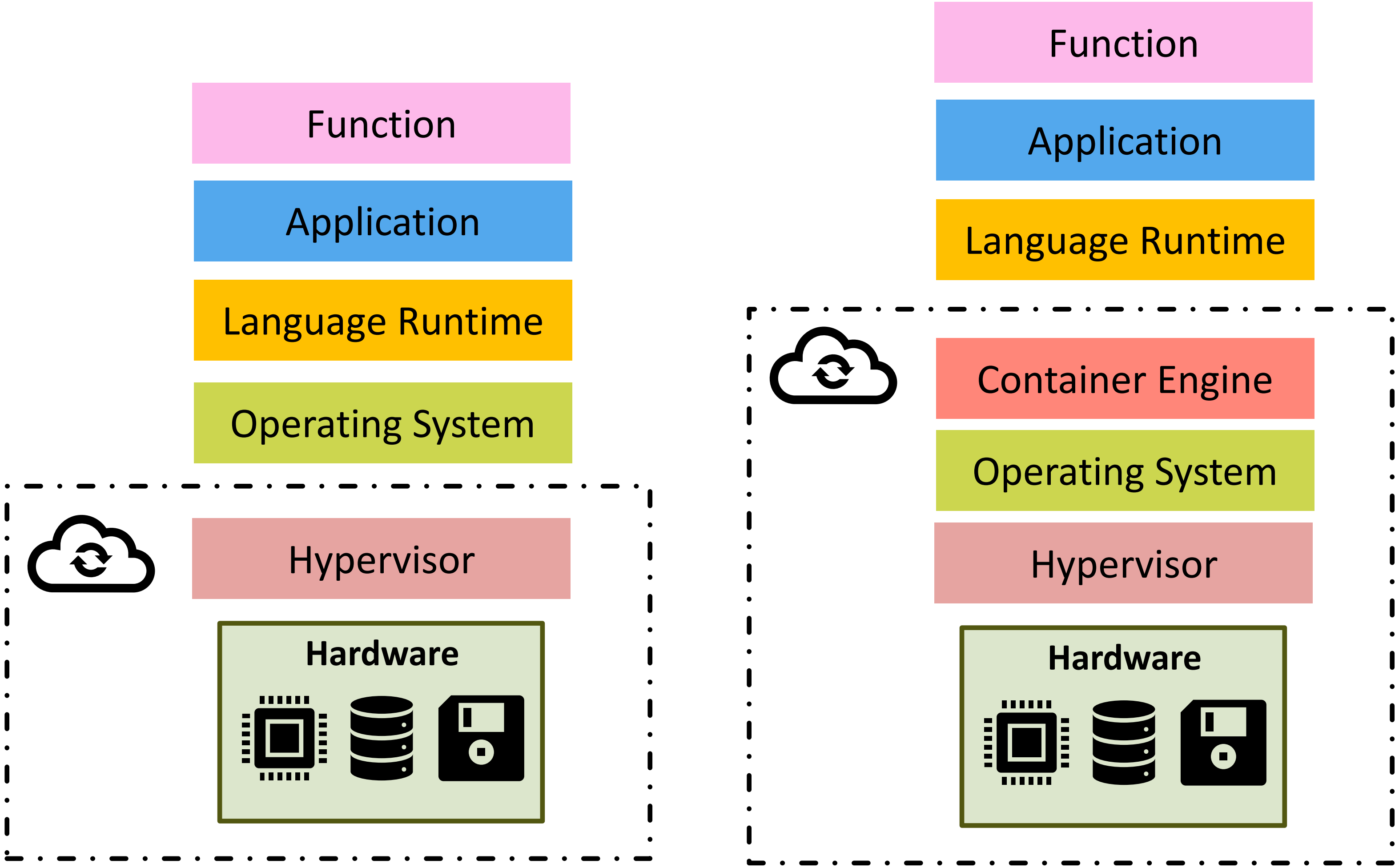
Cloud and Serverless



Virtual Machine (Infrastructure-as-a-Service)

Containers (Container-as-a-Service)

Cloud and Serverless



Function

Application

Language Runtime

Operating System

Hypervisor

Hardware

Function

Application

Language Runtime

Container Engine

Operating System

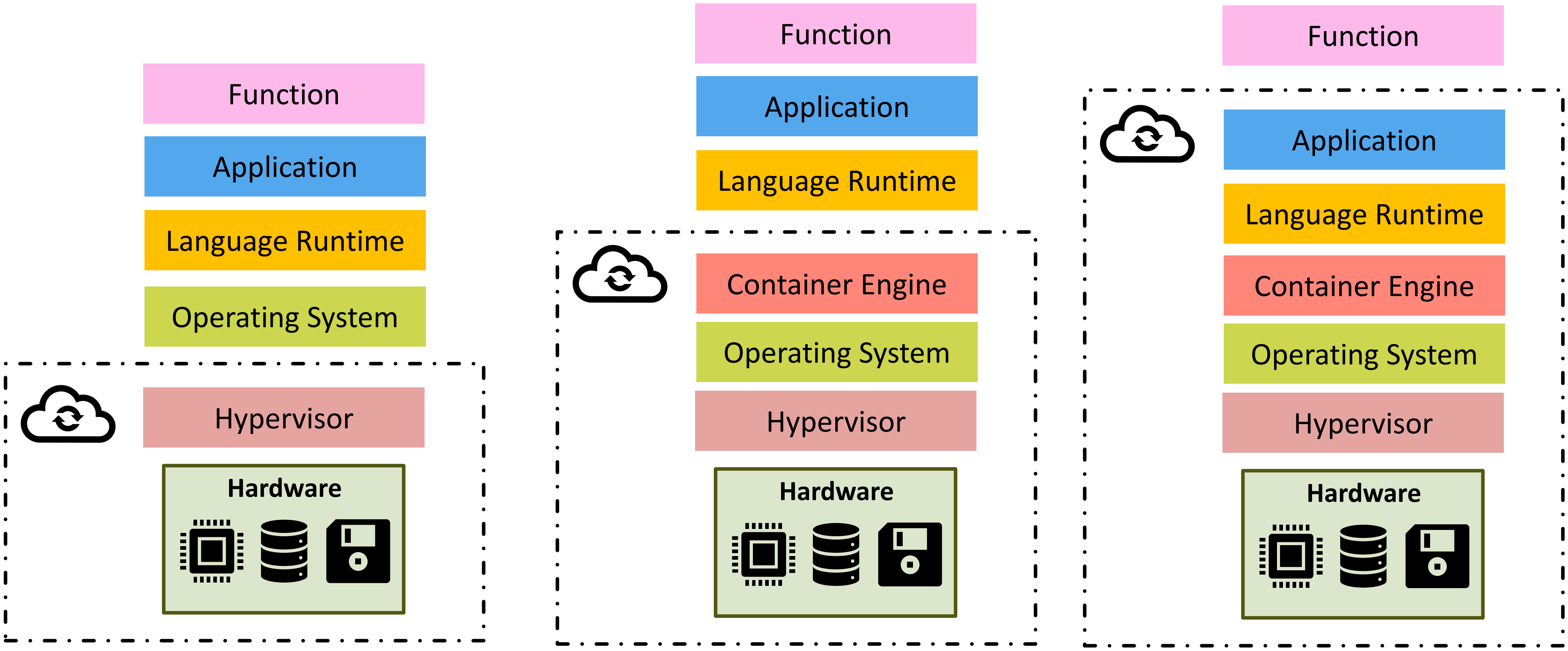
Hypervisor

Hardware

Virtual Machine (Infrastructure-as-a-Service)

Containers (Container-as-a-Service)

Cloud and Serverless

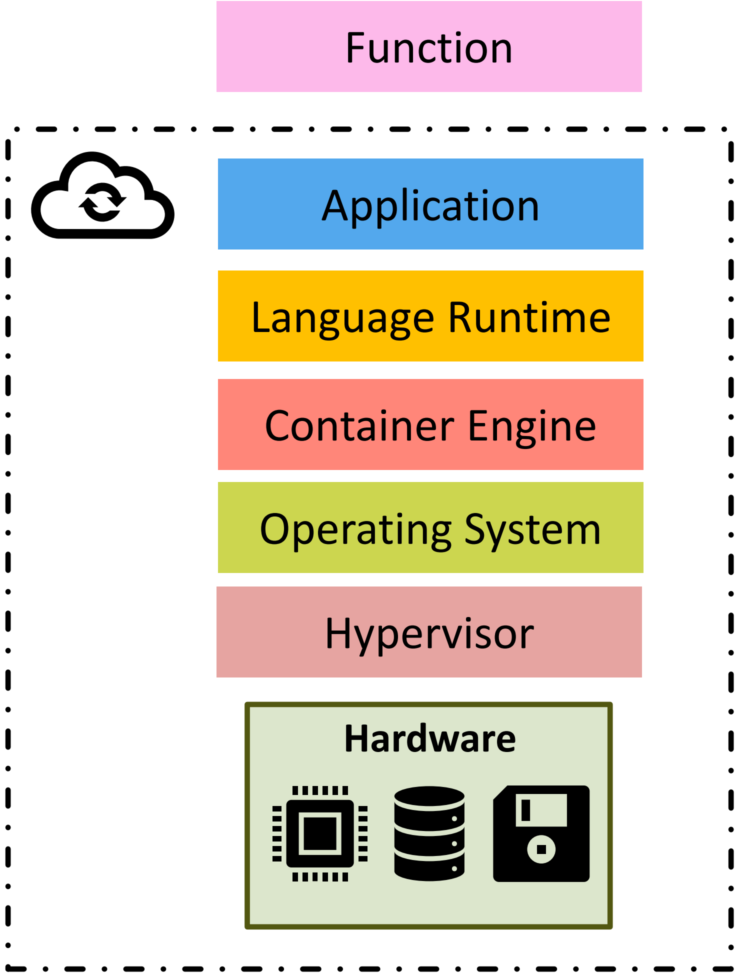


Virtual Machine (Infrastructure-as-a-Service)

Containers (Container-as-a-Service)

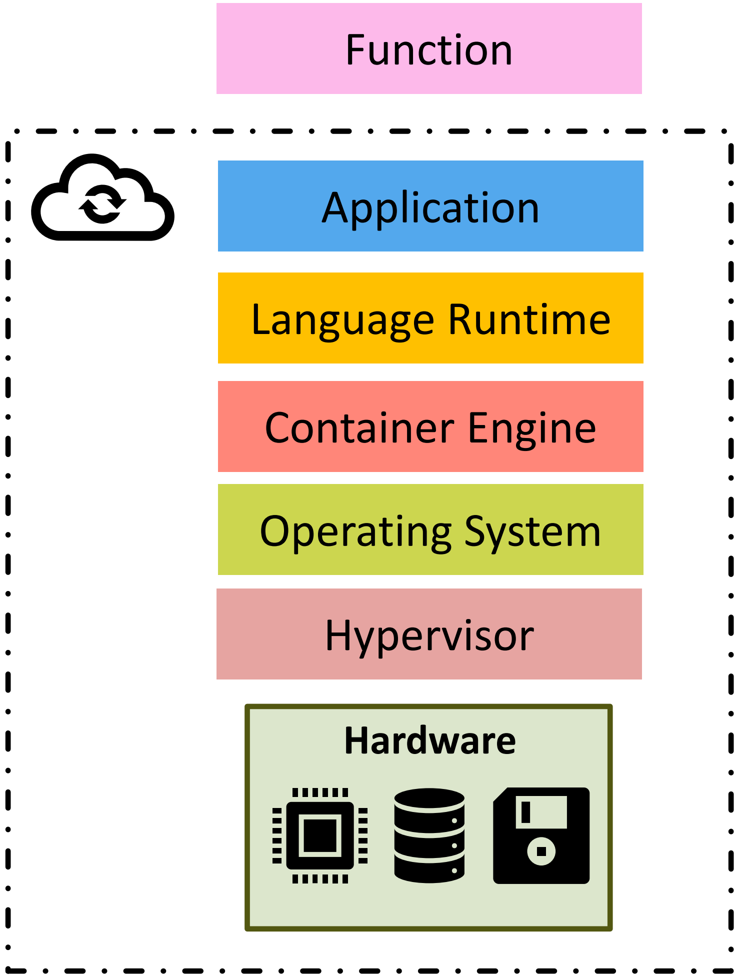
Functions (Function-as-a-Service)

Cloud and Serverless

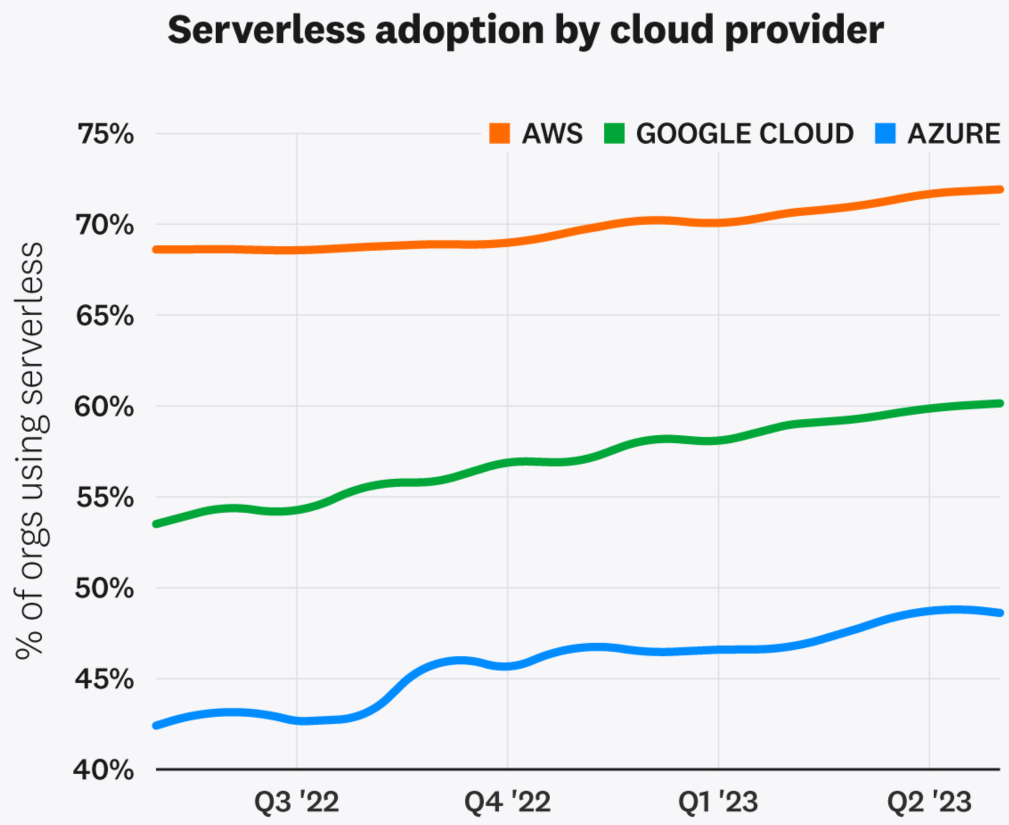


Functions (Function-as-a-Service)

Cloud and Serverless



Functions (Function-as-a-Service)



“State of Serverless 2023”, Datadog

How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):
    new_data = process_logic(request['op‘], request[‘data‘])
    return stamp
```

How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):
    new_data = process_logic(request['op‘], request[‘data‘])
    return stamp
```

+

Configuration: CPU RAM ⏰

How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):

    new_data = process_logic(request['op‘], request[‘data‘])

    return stamp
```

+

Configuration: CPU RAM ⏰

⬇

☁

How does Function-as-a-Service (FaaS) work?

Web Application

```
def handler_function(request: dict, context: dict):
    new_data = process_logic(request['op‘], request[‘data‘])
    return stamp
```

+

Configuration: CPU RAM 🐍 ⏰

⬇

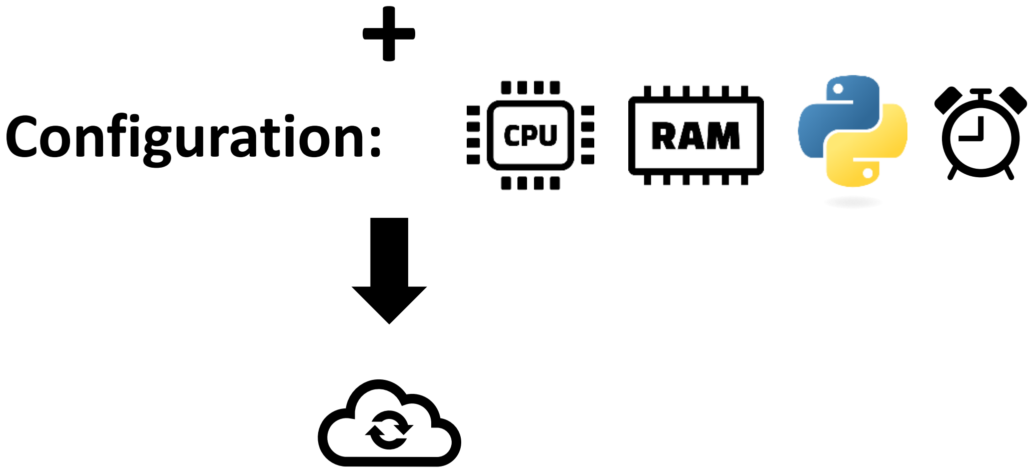
☁

How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):

    new_data = process_logic(request['op‘], request[‘data‘])

    return stamp
```

Web Application

Image Processing



+

Configuration:

How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    new_data = process_logic(request['op'], data)

    stamp = cloud_storage.write(request['id'], new_data)

    return stamp
```

+

Configuration: CPU RAM ⏰

⬇

☁

Web Application

Image Processing

How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):
    data = cloud_storage.read(request['id'])
    new_data = process_logic(request['op'], data)
    stamp = cloud_storage.write(request['id'], new_data)
    return stamp
```

+

Configuration: CPU RAM ⏰

⬇

☁

Web Application

Image Processing

Microservice

How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):
    data = redis_cache.read(request['id'])
    new_data = process_logic(request['op'], data)
    stamp = redis_cache.write(request['id'], new_data)
    return stamp
```

+

Configuration: CPU RAM ⏰

⬇

Web Application

Image Processing

Microservice

How does Function-as-a-Service (FaaS) work?

```python
def handler_function(request: dict, context: dict):
    data = redis_cache.read(request['id'])
    new_data = process_logic(request['op'], data)
    stamp = redis_cache.write(request['id'], new_data)
    return stamp
```

+

Configuration: CPU RAM

⬇

Web Application

Image Processing

Microservice

Distributed Compilation

How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):
    data = fs_read(path.join('/mnt/', request['id']))
    new_data = process_logic(request['op'], data)
    stamp = fs_write(path.join('/mnt/', request['id']), new_data)
    return stamp
```

+

Configuration: CPU RAM ⏰

⬇

☁

Web Application

Image Processing

Microservice

Distributed Compilation

How does Function-as-a-Service (FaaS) work?

```
def handler_function(request: dict, context: dict):
    data = fs_read(path.join('/mnt/', request['id']))
    new_data = process_logic(request['op'], data)
    stamp = fs_write(path.join('/mnt/', request['id']), new_data)
    return stamp
```

+

Configuration: CPU RAM

↓

Web Application

Image Processing

Microservice

Distributed Compilation

ML Inference

How does Function-as-a-Service (FaaS) work?

```
model = model_init(cloud_storage, 'resnet50')

def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    new_data = process_logic(request['op'], data)

    stamp = cloud_storage.write(request['id'], new_data)

    return stamp
```

+

Configuration: CPU RAM ⏰

↓

Web Application

Image Processing

Microservice

Distributed Compilation

ML Inference

How does Function-as-a-Service (FaaS) work?

```
model = model_init(cloud_storage, 'resnet50')

def handler_function(request: dict, context: dict):

    return stamp
```

+

Configuration: CPU RAM

Web Application

Image Processing

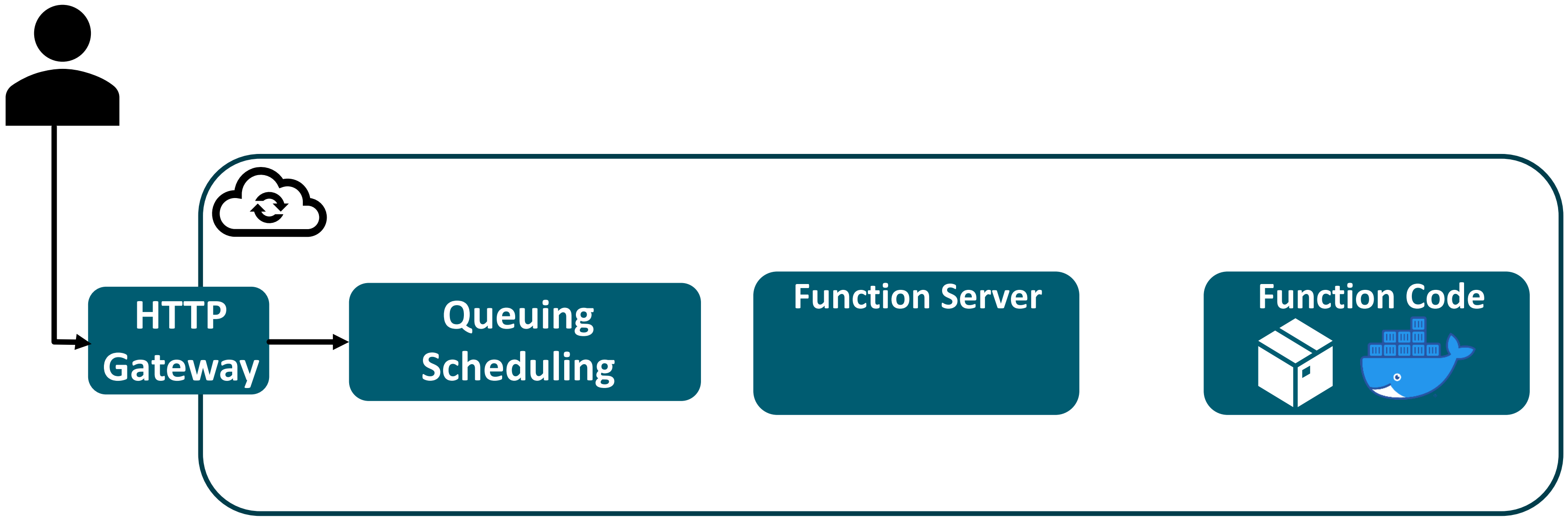
Distributed Compilation

ML Inference

#1 Disaggregated Compute from Storage

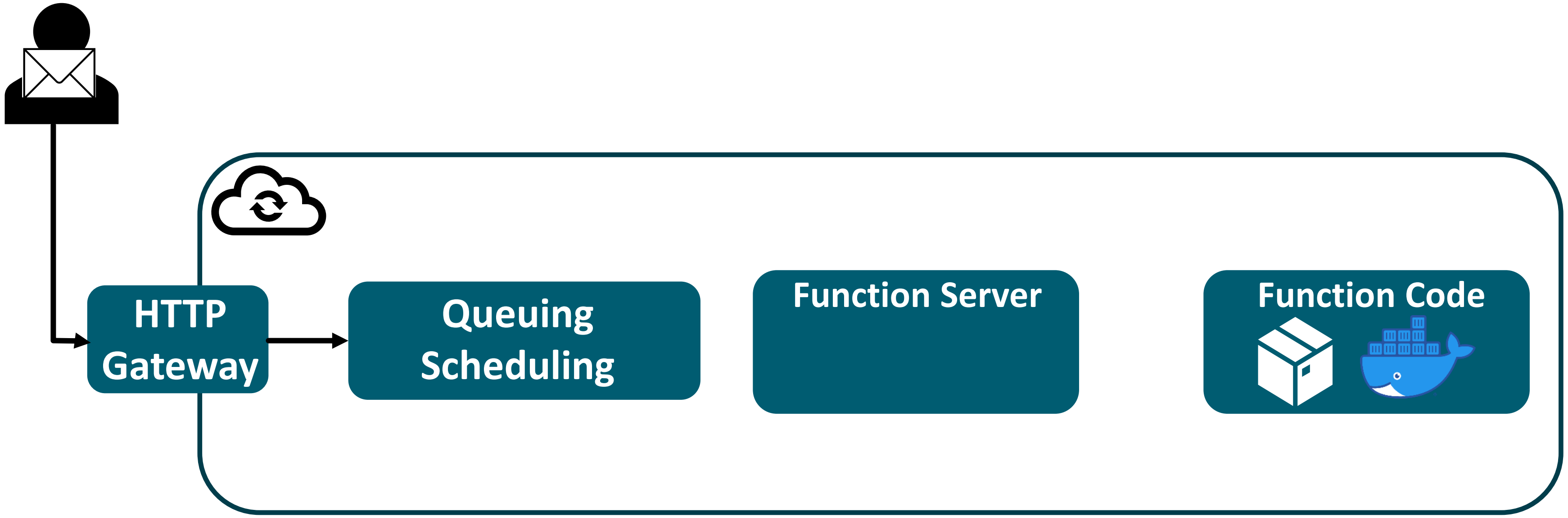
#2 “Stateless” Functions

How does Function-as-a-Service (FaaS) work?



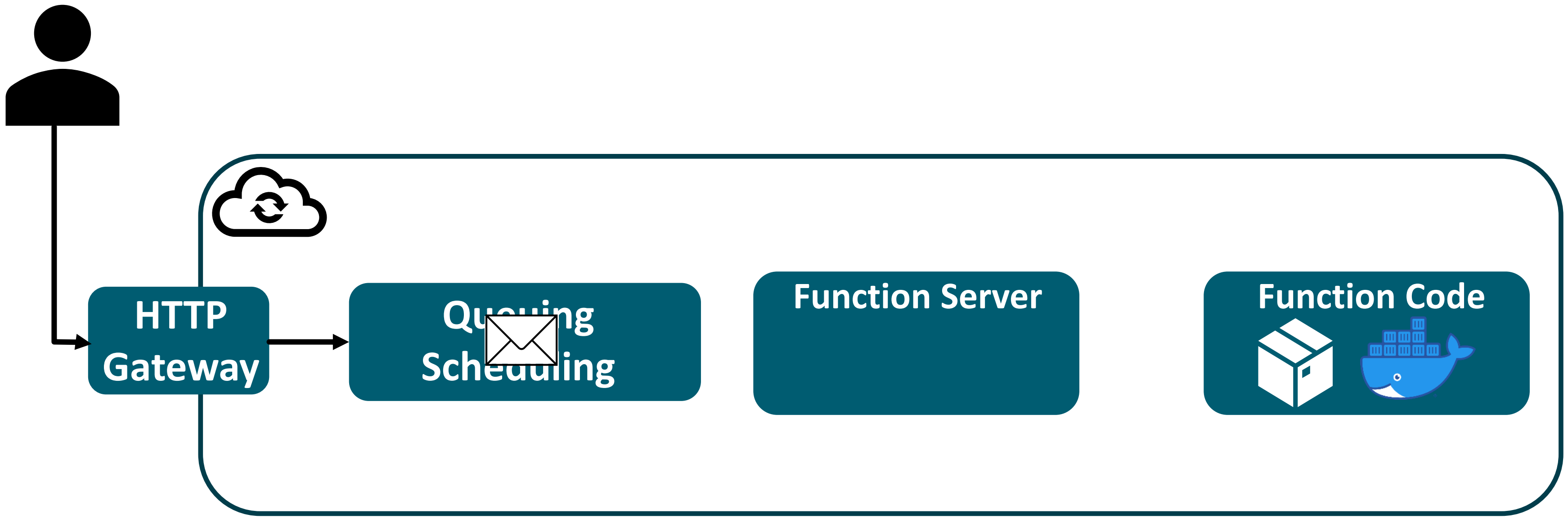
How does Function-as-a-Service (FaaS) work?

Functions is triggered by sending input data.

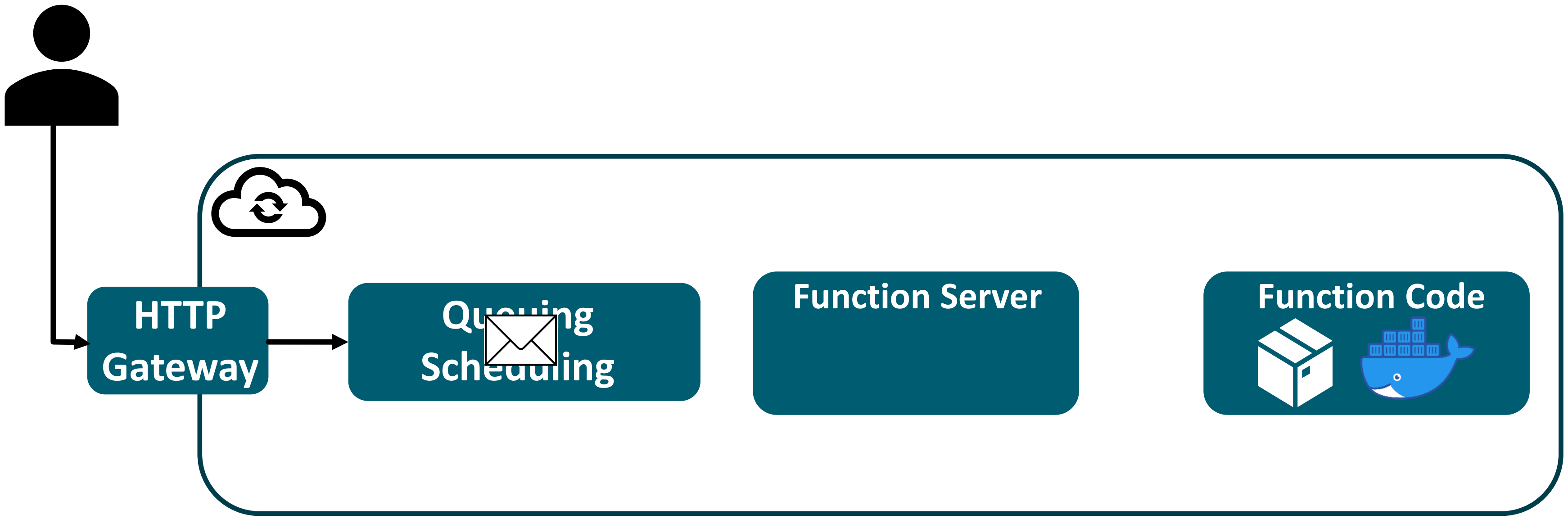


How does Function-as-a-Service (FaaS) work?

Functions is triggered by sending input data.



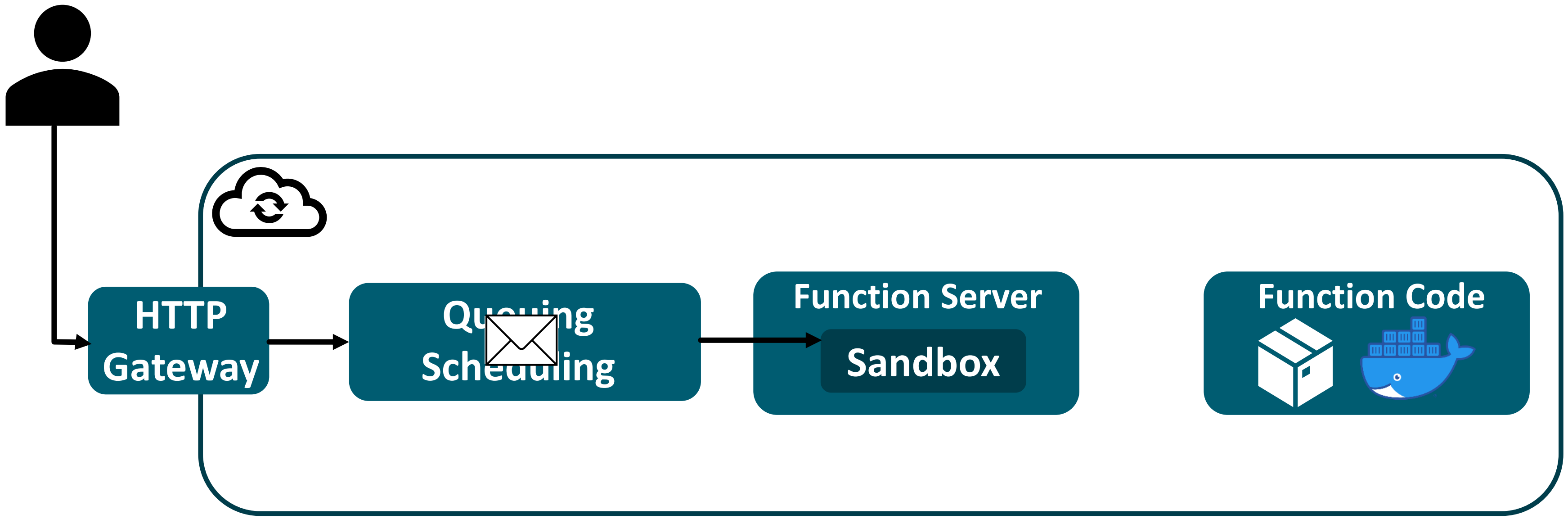
How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

FaaS system looks for a sandbox to run the function.

How does Function-as-a-Service (FaaS) work?

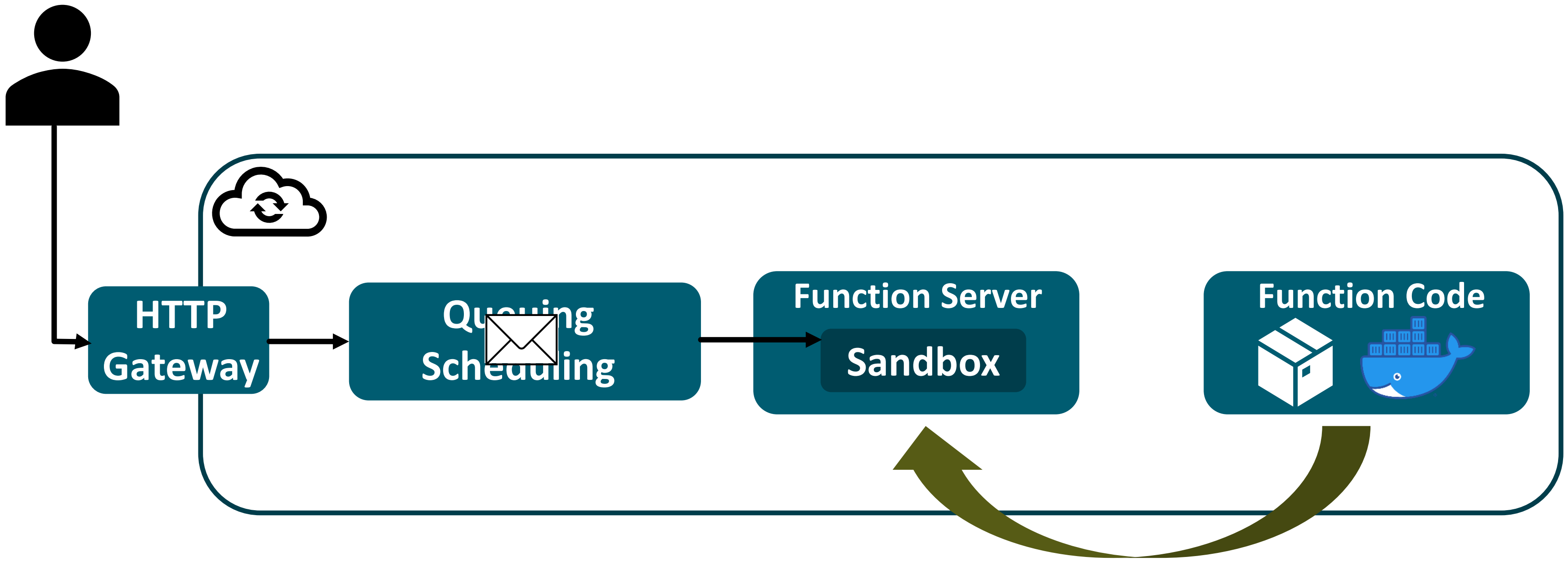


Functions is triggered by sending input data.

FaaS system looks for a sandbox to run the function.

A new sandbox is initialized to handle the invocation.

How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

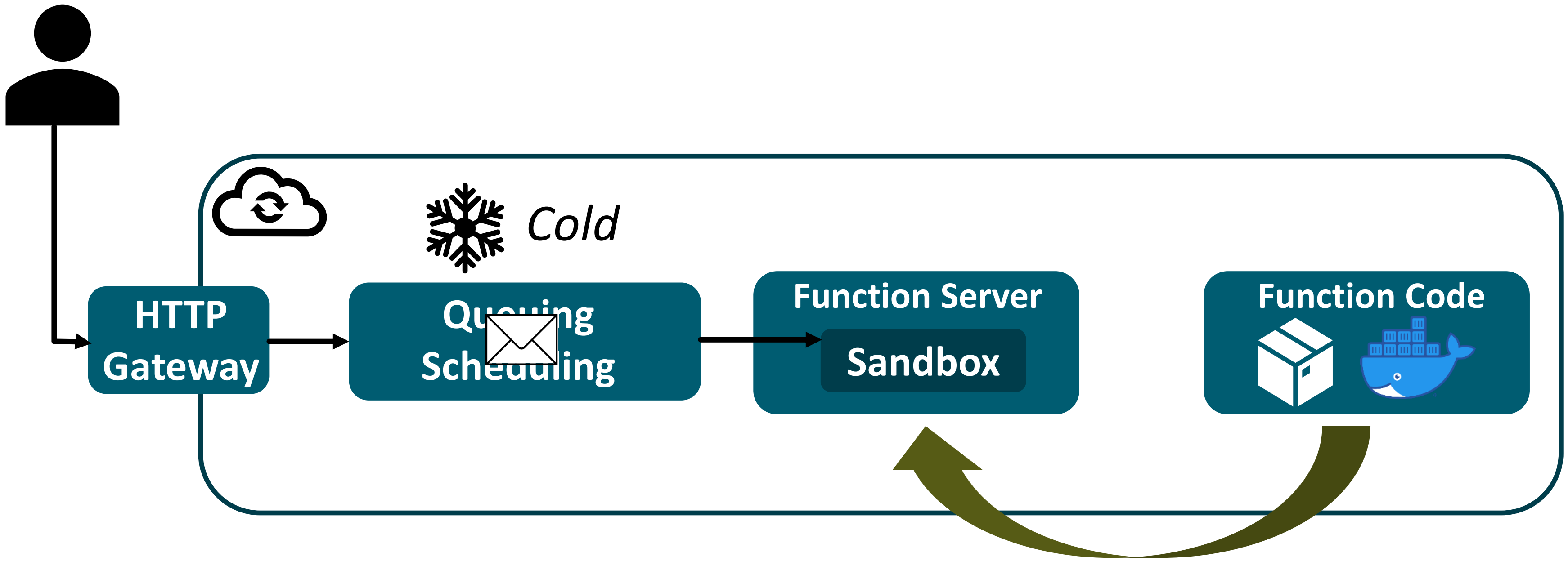
↓

FaaS system looks for a sandbox to run the function.

↓

A new sandbox is initialized to handle the invocation.

How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

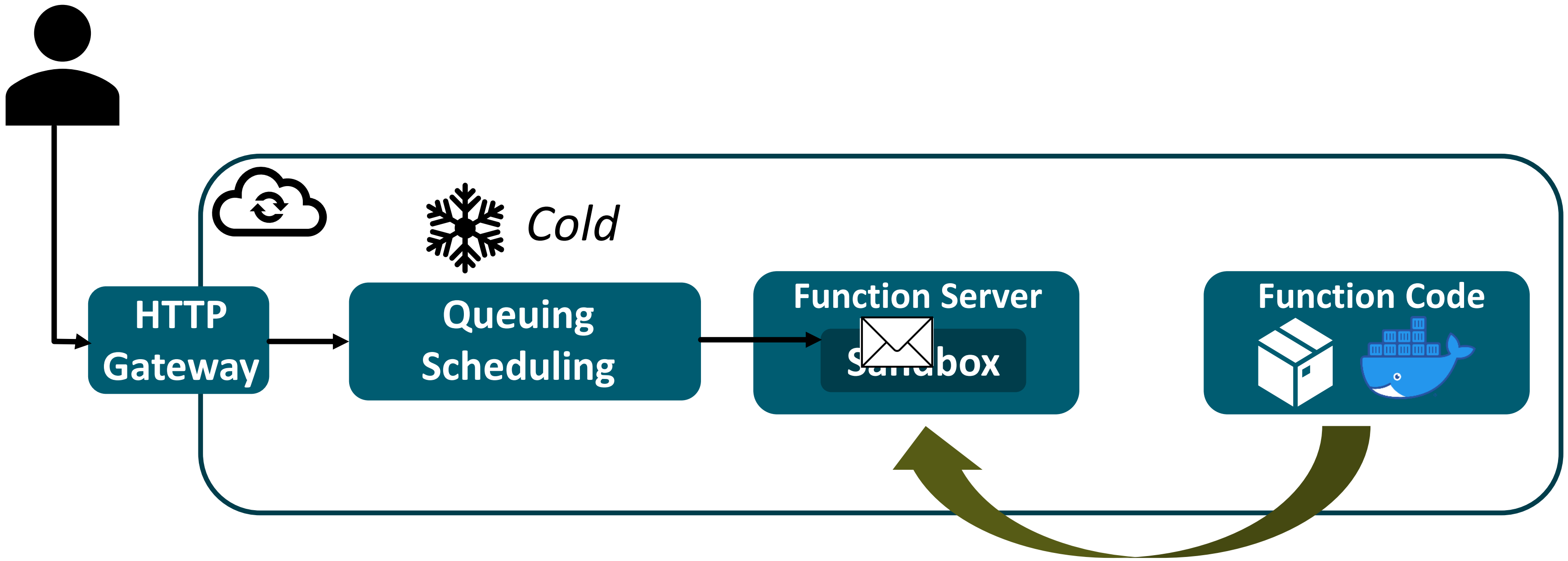
↓

FaaS system looks for a sandbox to run the function.

↓

A new sandbox is initialized to handle the invocation.

How does Function-as-a-Service (FaaS) work?

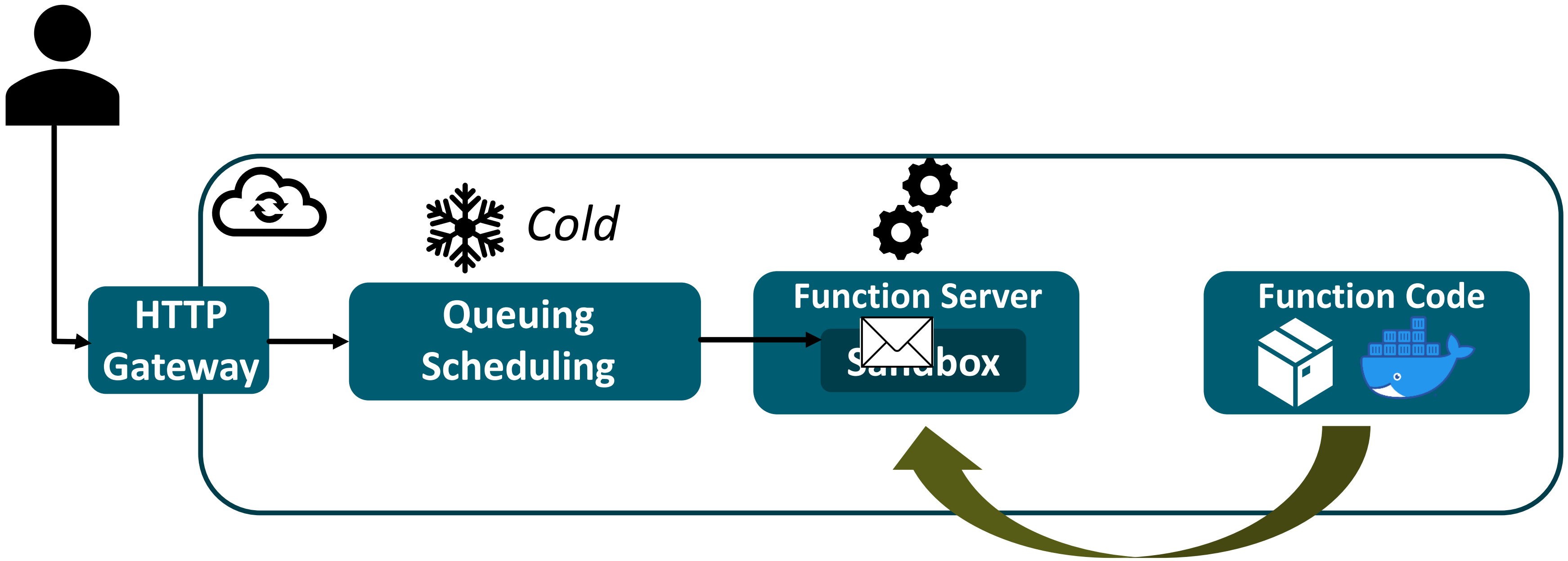


Functions is triggered by sending input data.

FaaS system looks for a sandbox to run the function.

A new sandbox is initialized to handle the invocation.

How does Function-as-a-Service (FaaS) work?



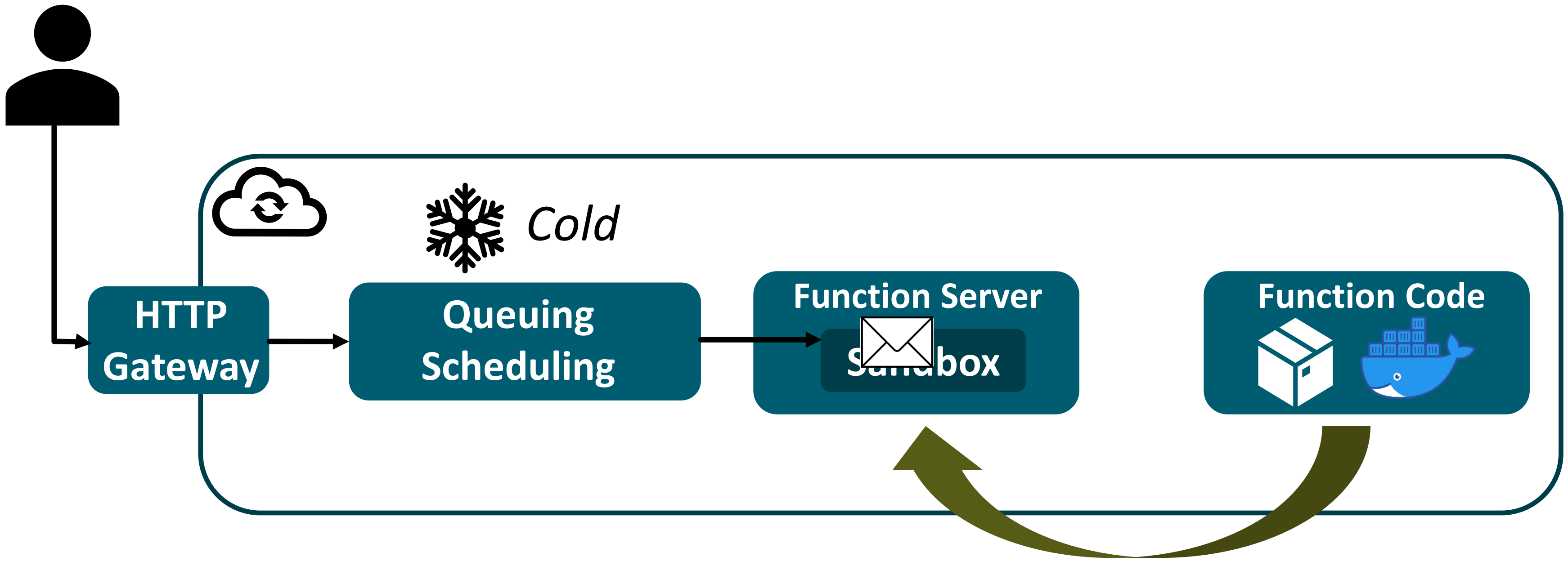
Functions is triggered by sending input data.

FaaS system looks for a sandbox to run the function.

A new sandbox is initialized to handle the invocation.

Function is executed ***somewhere*** in the cloud.

How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

↓

FaaS system looks for a sandbox to run the function.

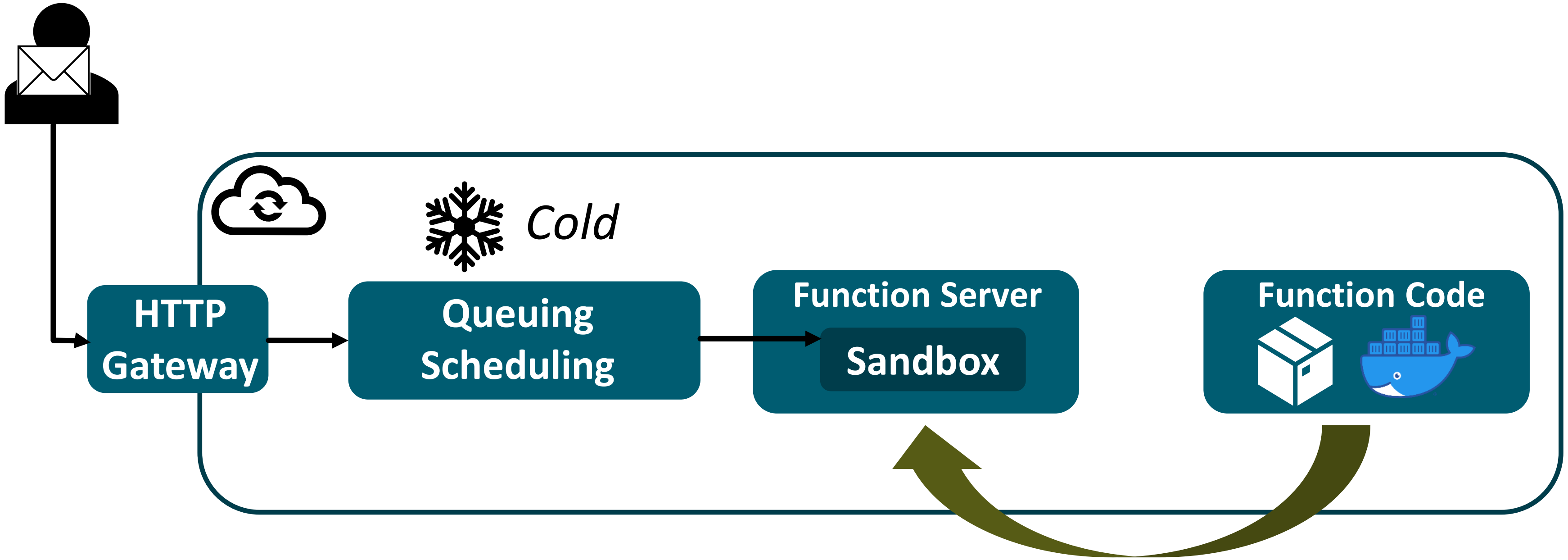
↓

A new sandbox is initialized to handle the invocation.

↓

Function is executed ***somewhere*** in the cloud.

How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

↓

FaaS system looks for a sandbox to run the function.

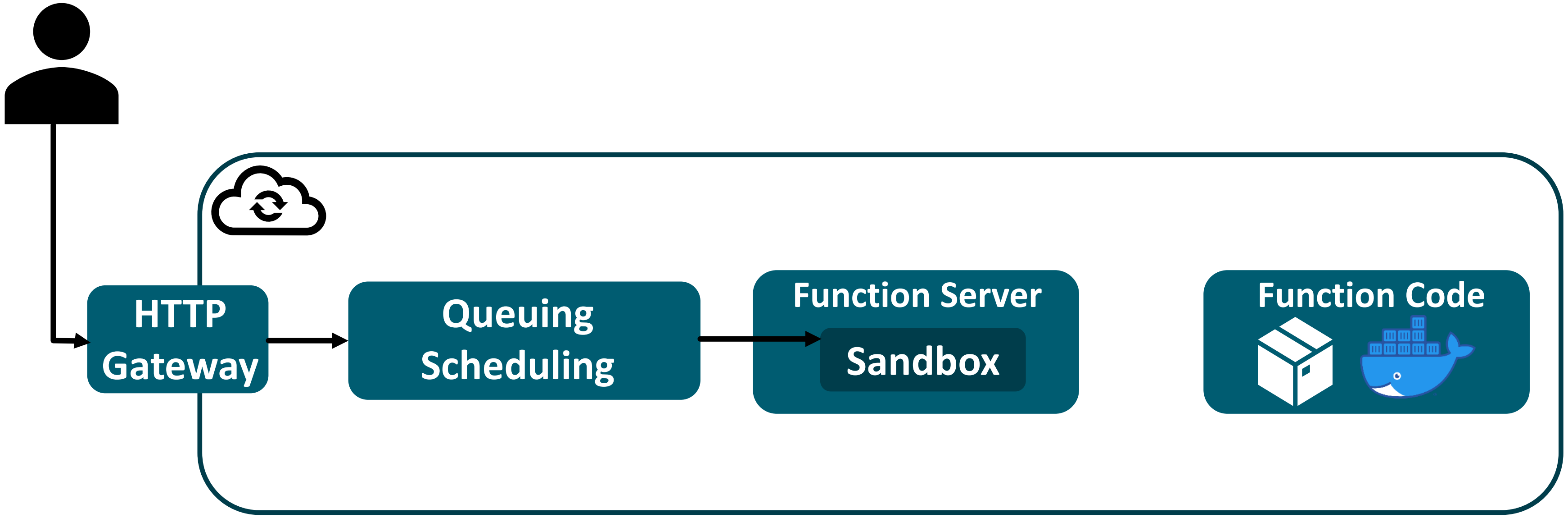
↓

A new sandbox is initialized to handle the invocation.

↓

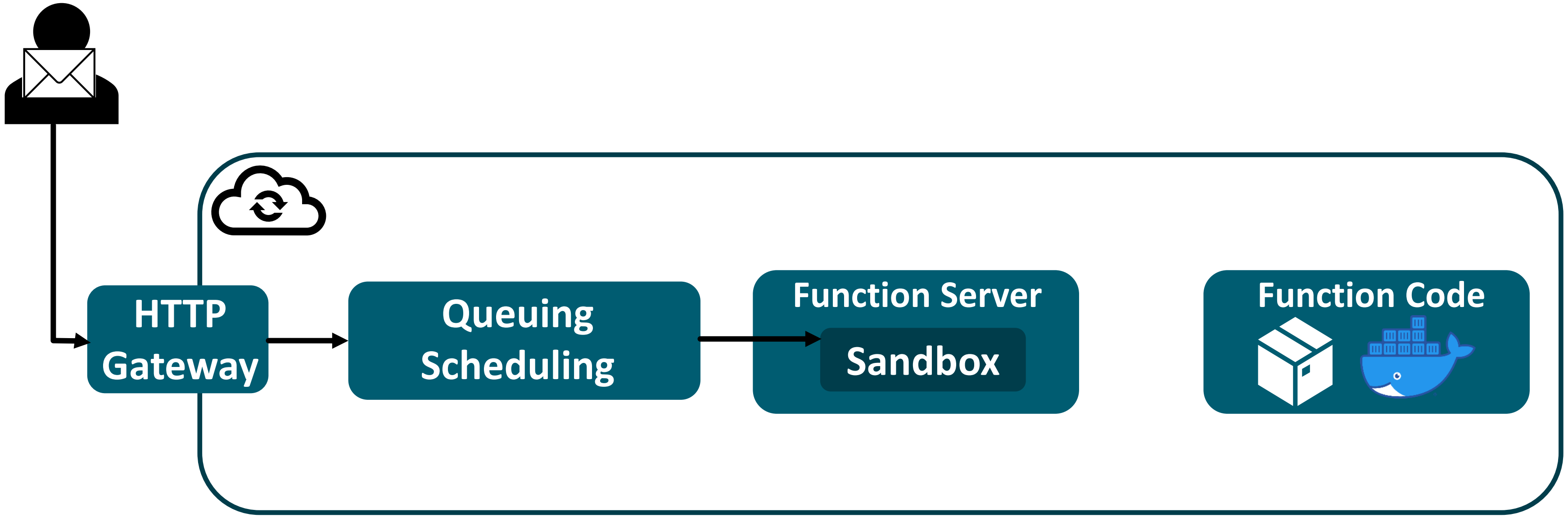
Function is executed ***somewhere*** in the cloud.

How does Function-as-a-Service (FaaS) work?



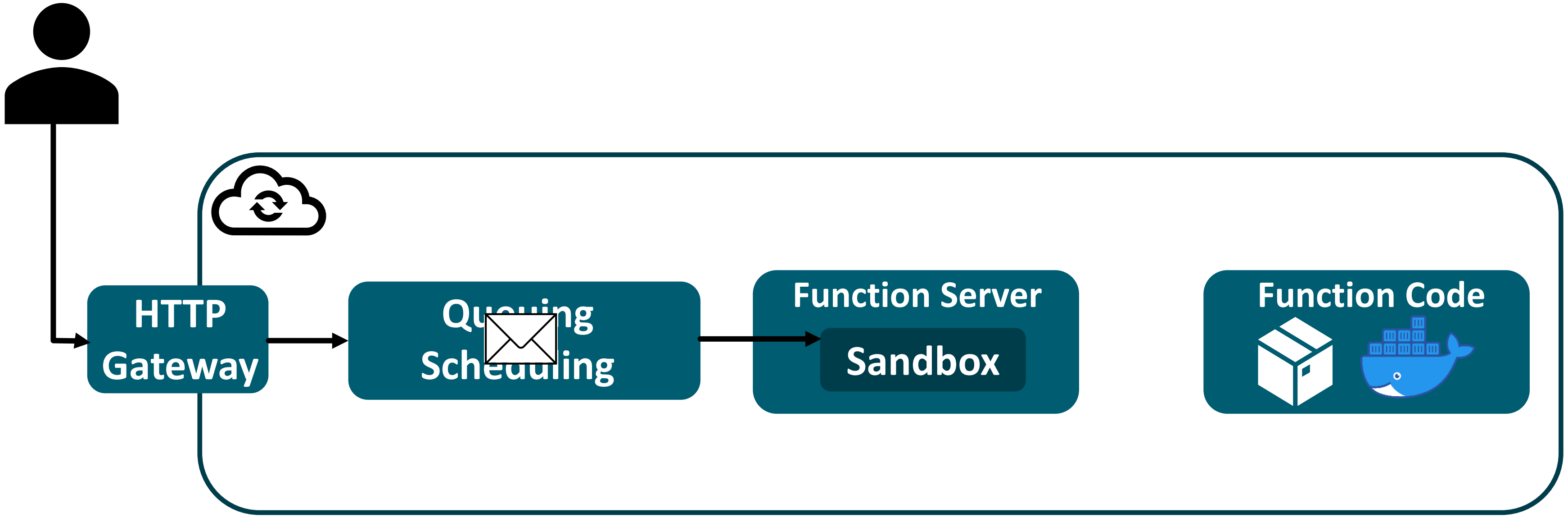
How does Function-as-a-Service (FaaS) work?

Functions is triggered by sending input data.

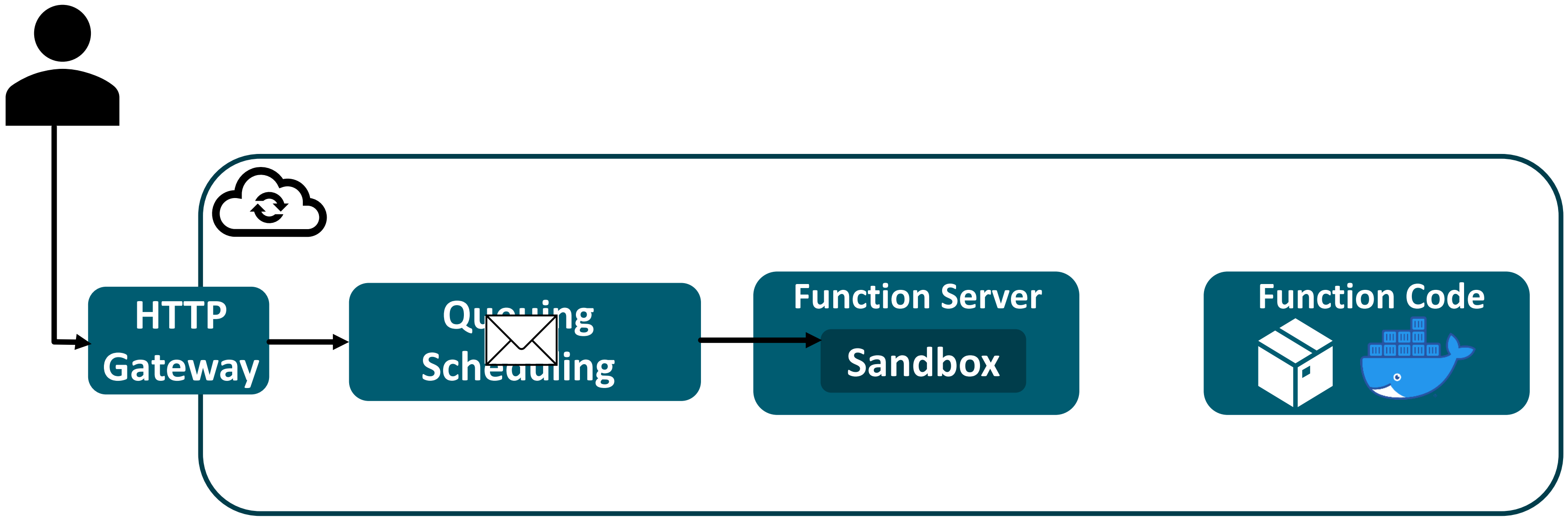


How does Function-as-a-Service (FaaS) work?

Functions is triggered by sending input data.



How does Function-as-a-Service (FaaS) work?

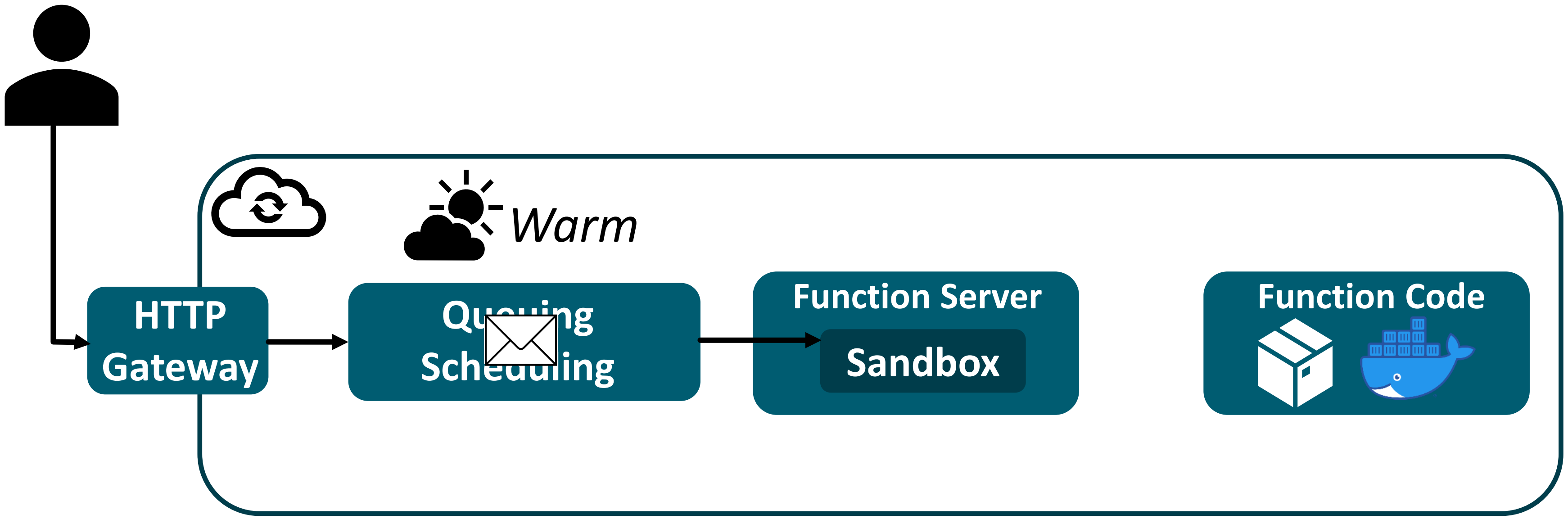


Functions is triggered by sending input data.

↓

FaaS system looks for a sandbox to run the function.

How does Function-as-a-Service (FaaS) work?

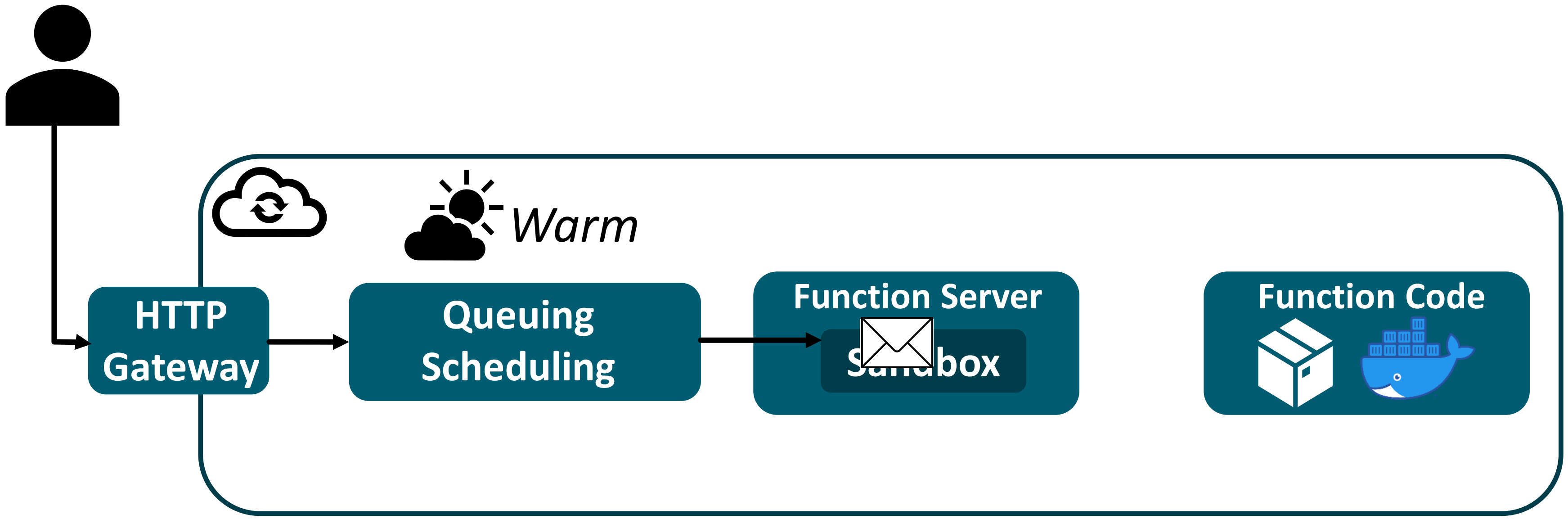


Functions is triggered by sending input data.

↓

FaaS system looks for a sandbox to run the function.

How does Function-as-a-Service (FaaS) work?

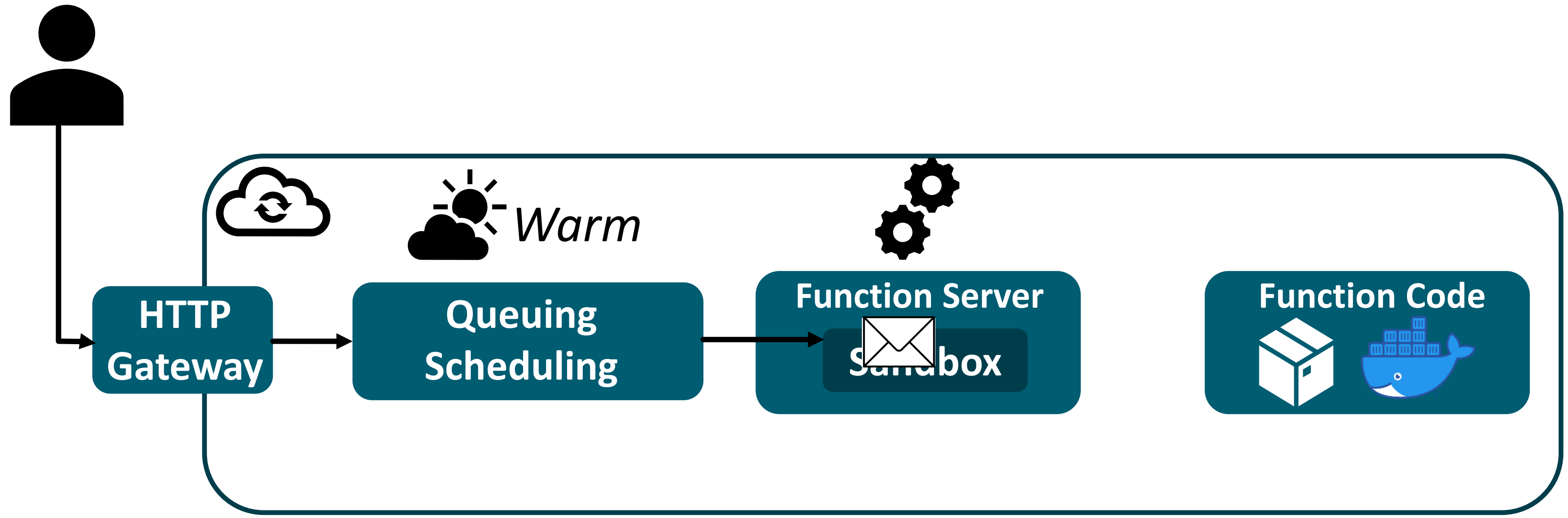


Functions is triggered by sending input data.

↓

FaaS system looks for a sandbox to run the function.

How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

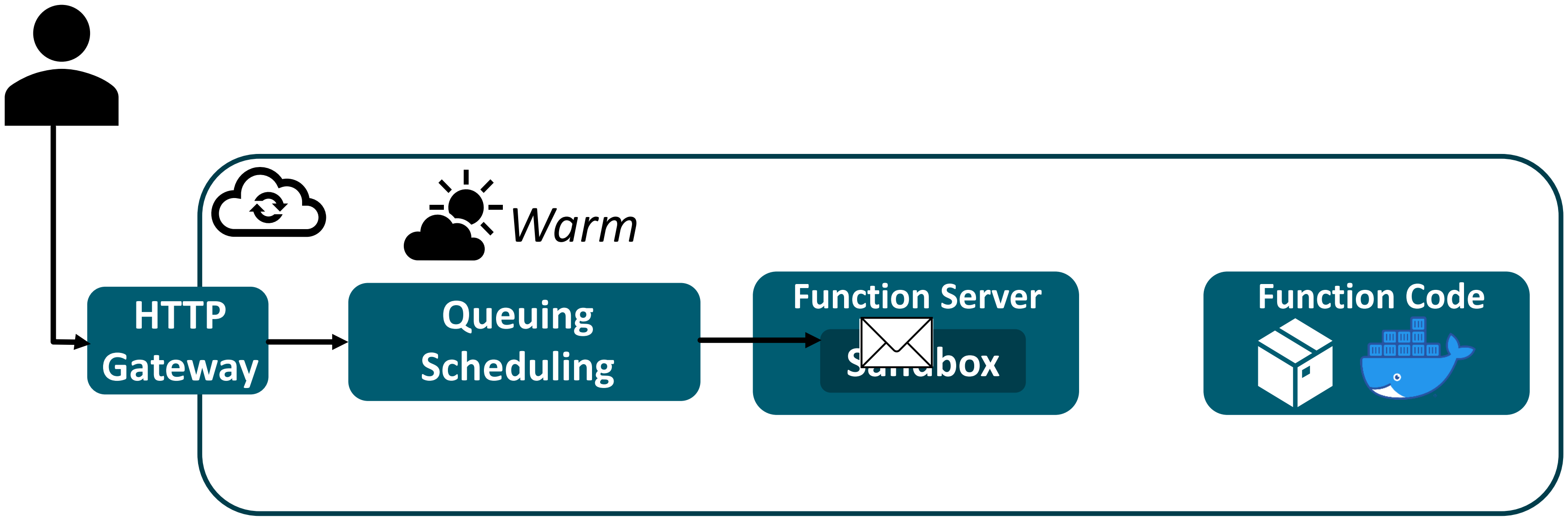
↓

FaaS system looks for a sandbox to run the function.

↓

Function is executed ***somewhere*** in the cloud.

How does Function-as-a-Service (FaaS) work?

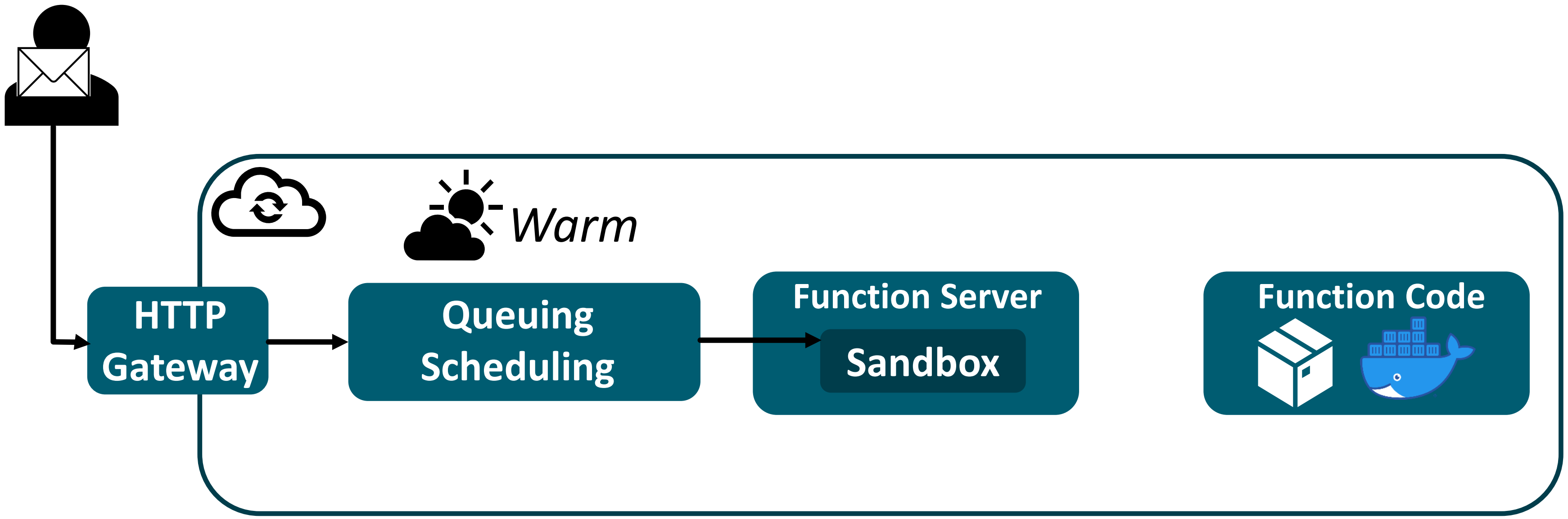


Functions is triggered by sending input data.

FaaS system looks for a sandbox to run the function.

Function is executed *somewhere* in the cloud.

How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

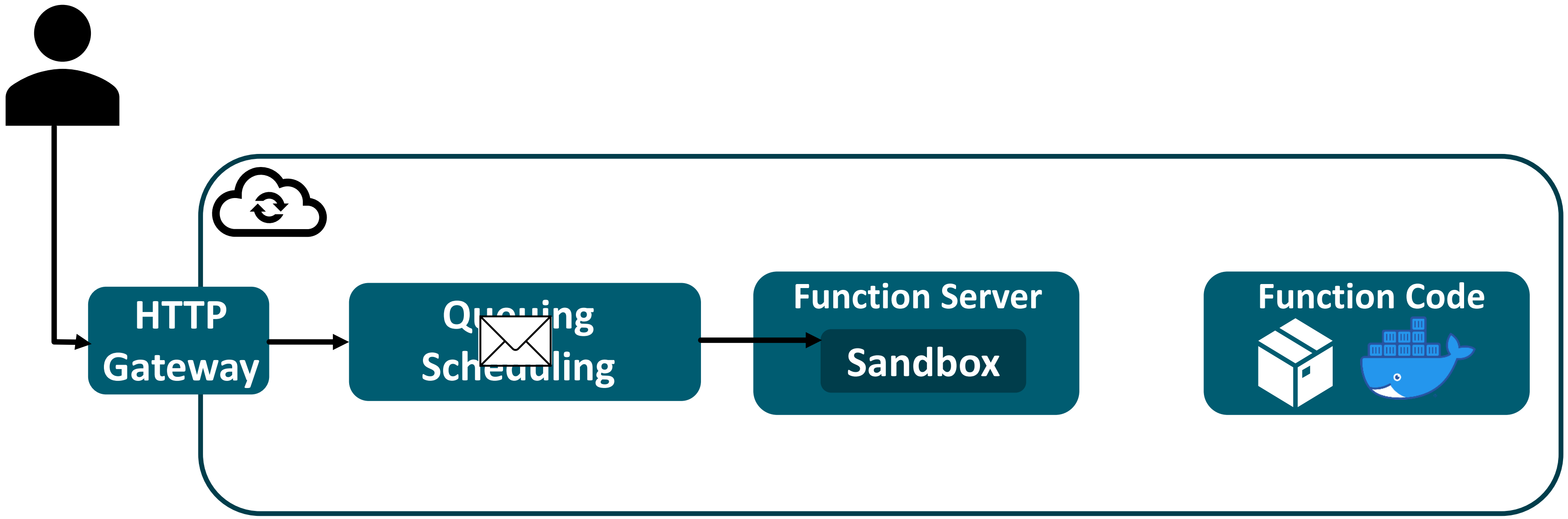
↓

FaaS system looks for a sandbox to run the function.

↓

Function is executed ***somewhere*** in the cloud.

How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

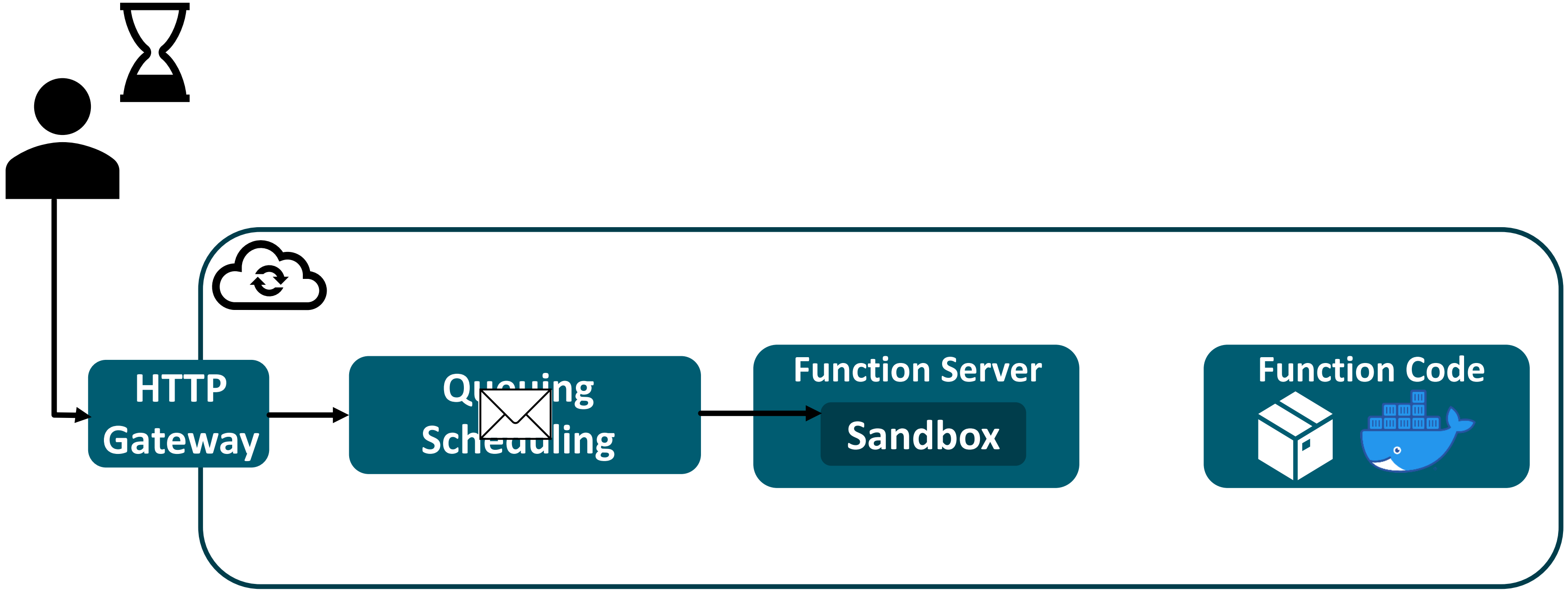
↓

FaaS system looks for a sandbox to run the function.

↓

Function is executed *somewhere* in the cloud.

How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

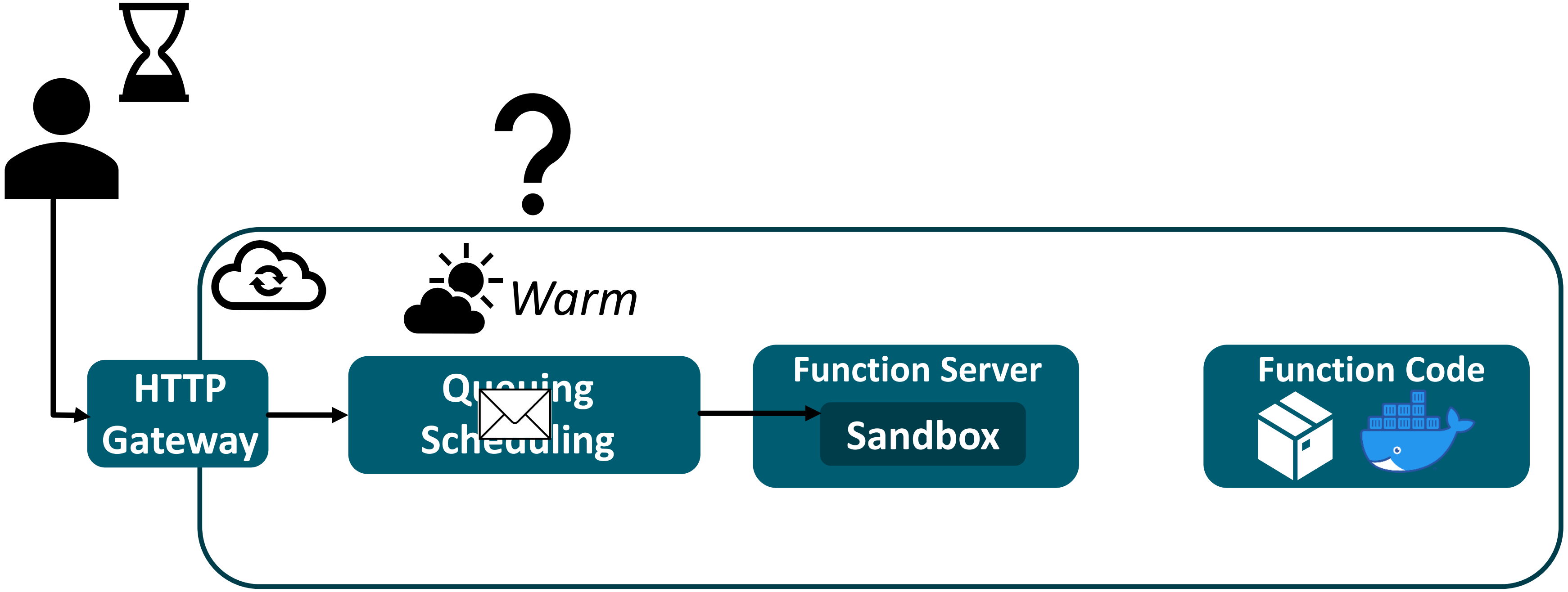
↓

FaaS system looks for a sandbox to run the function.

↓

Function is executed ***somewhere*** in the cloud.

How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

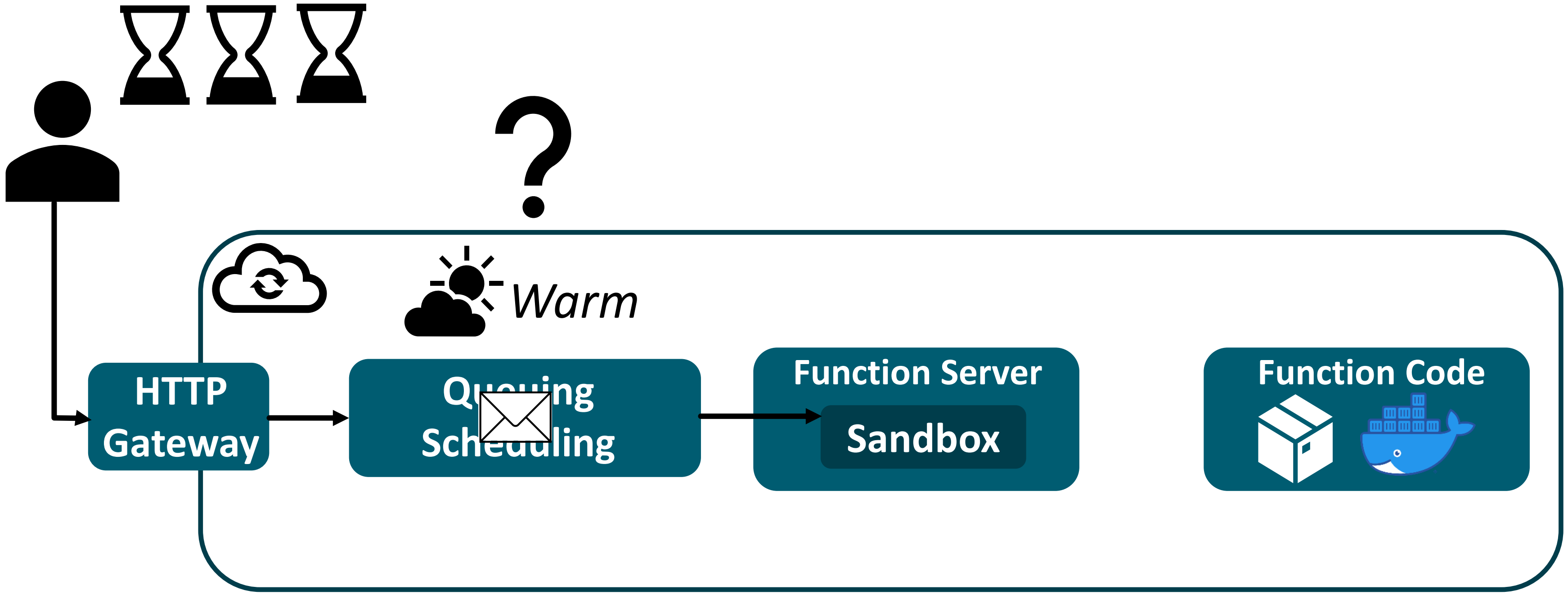
↓

FaaS system looks for a sandbox to run the function.

↓

Function is executed ***somewhere*** in the cloud.

How does Function-as-a-Service (FaaS) work?

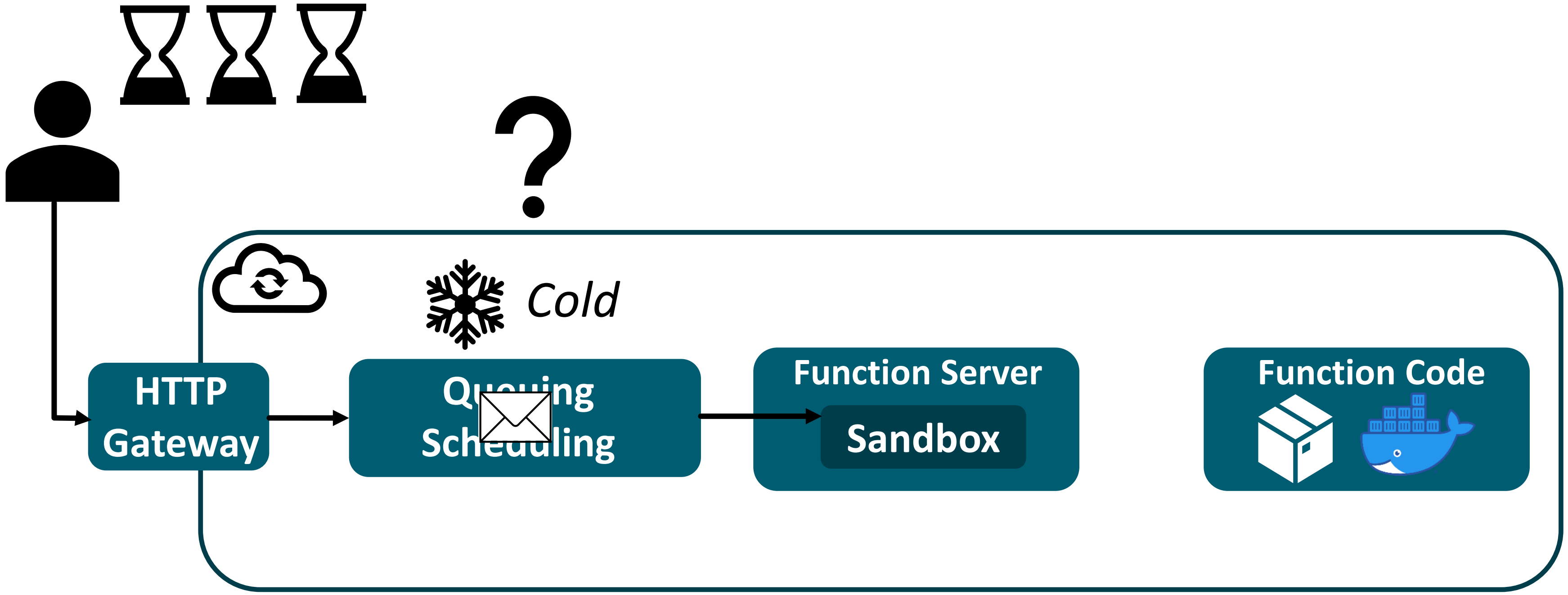


Functions is triggered by sending input data.

FaaS system looks for a sandbox to run the function.

Function is executed *somewhere* in the cloud.

How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

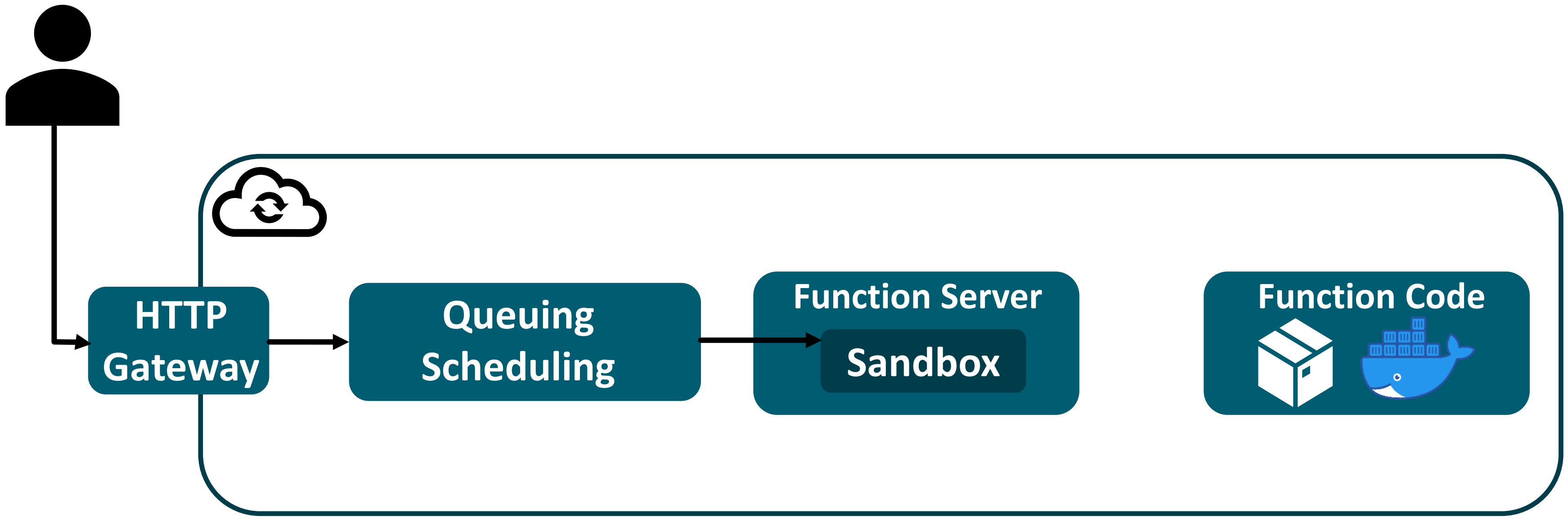
↓

FaaS system looks for a sandbox to run the function.

↓

Function is executed ***somewhere*** in the cloud.

How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

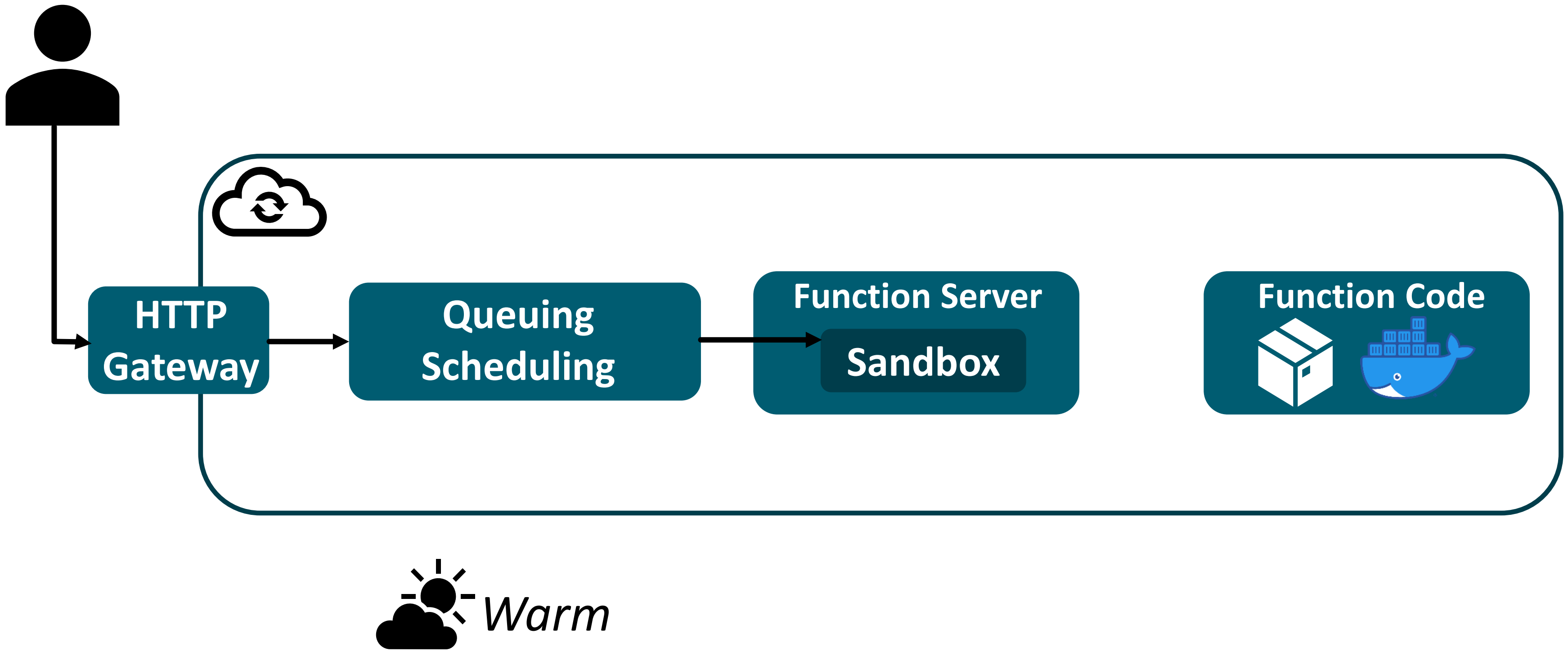
↓

FaaS system looks for a sandbox to run the function.

↓

Function is executed *somewhere* in the cloud.

How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

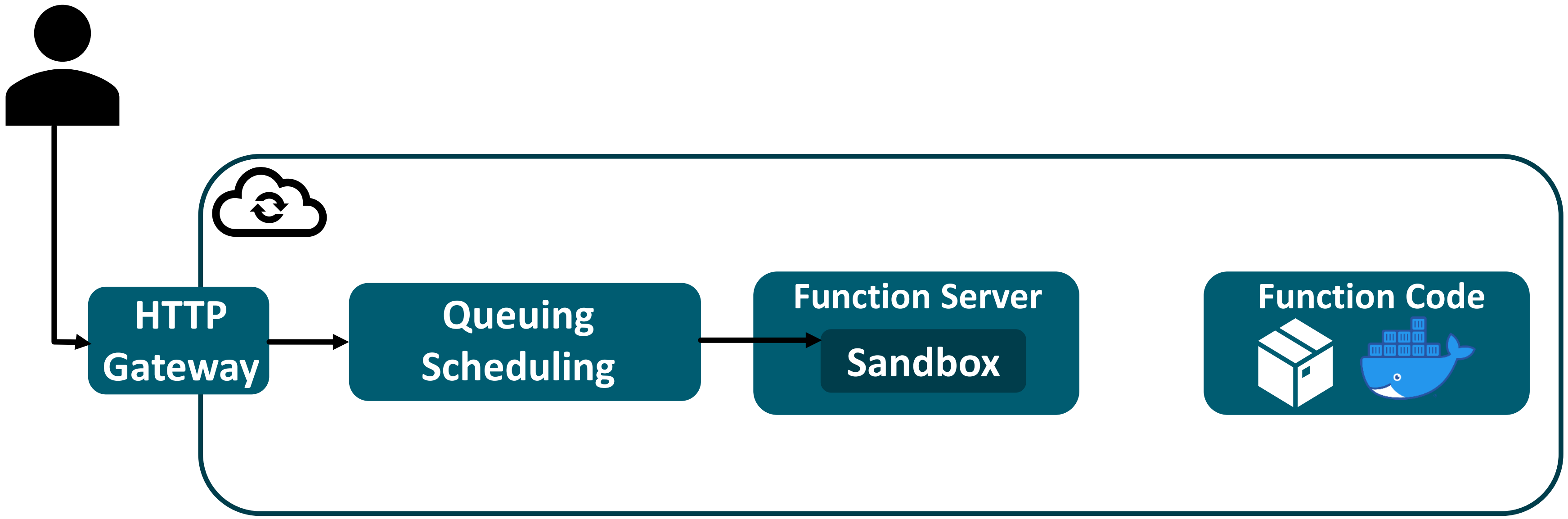
↓

FaaS system looks for a sandbox to run the function.

↓

Function is executed *somewhere* in the cloud.

How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

↓

FaaS system looks for a sandbox to run the function.

↓

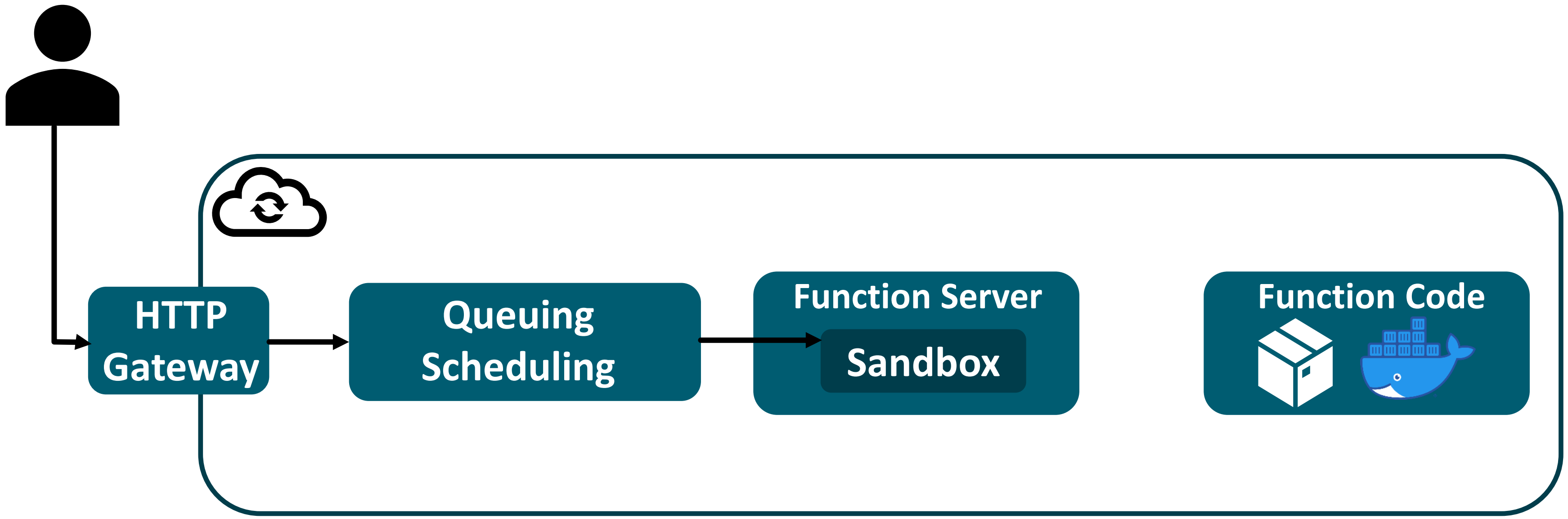
Function is executed ***somewhere*** in the cloud.

Warm

global var = 1

echo ‘a’ > /tmp/dir

How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

↓

FaaS system looks for a sandbox to run the function.

↓

Function is executed *somewhere* in the cloud.

Warm

global var = 1

echo ‘a’ > /tmp/dir

➡

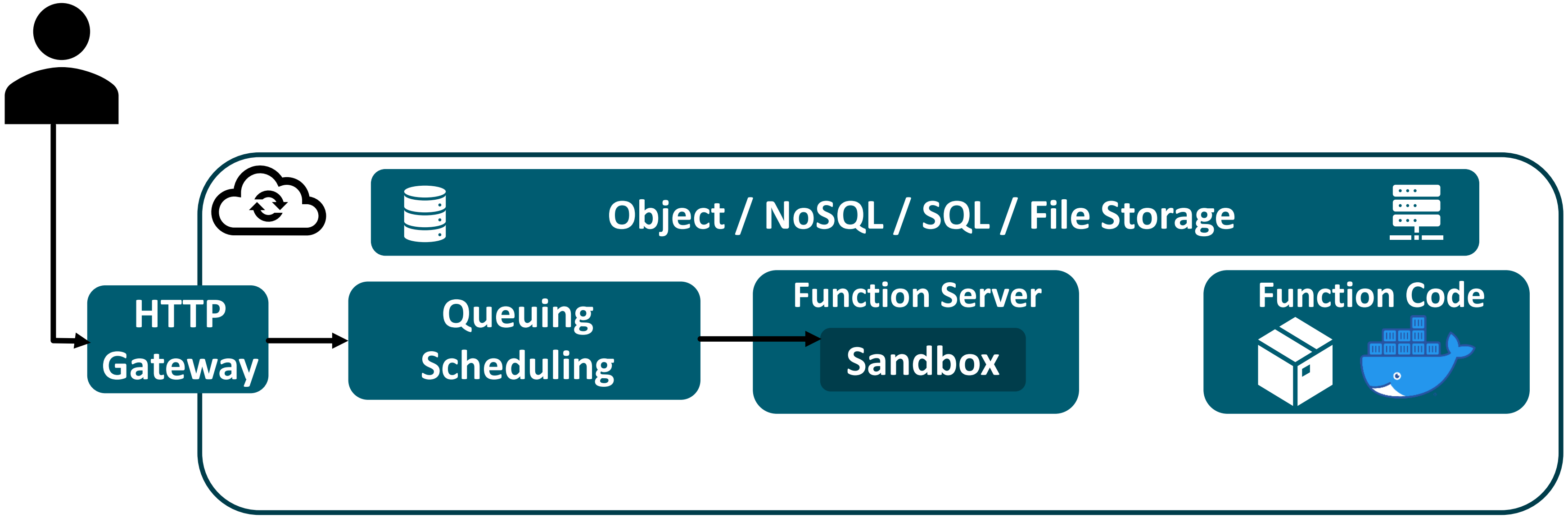
Warm

print(var)

cat /tmp/dir

?

How does Function-as-a-Service (FaaS) work?



Functions is triggered by sending input data.

↓

FaaS system looks for a sandbox to run the function.

↓

Function is executed ***somewhere*** in the cloud.

Warm

global var = 1

echo 'a' > /tmp/dir

➡

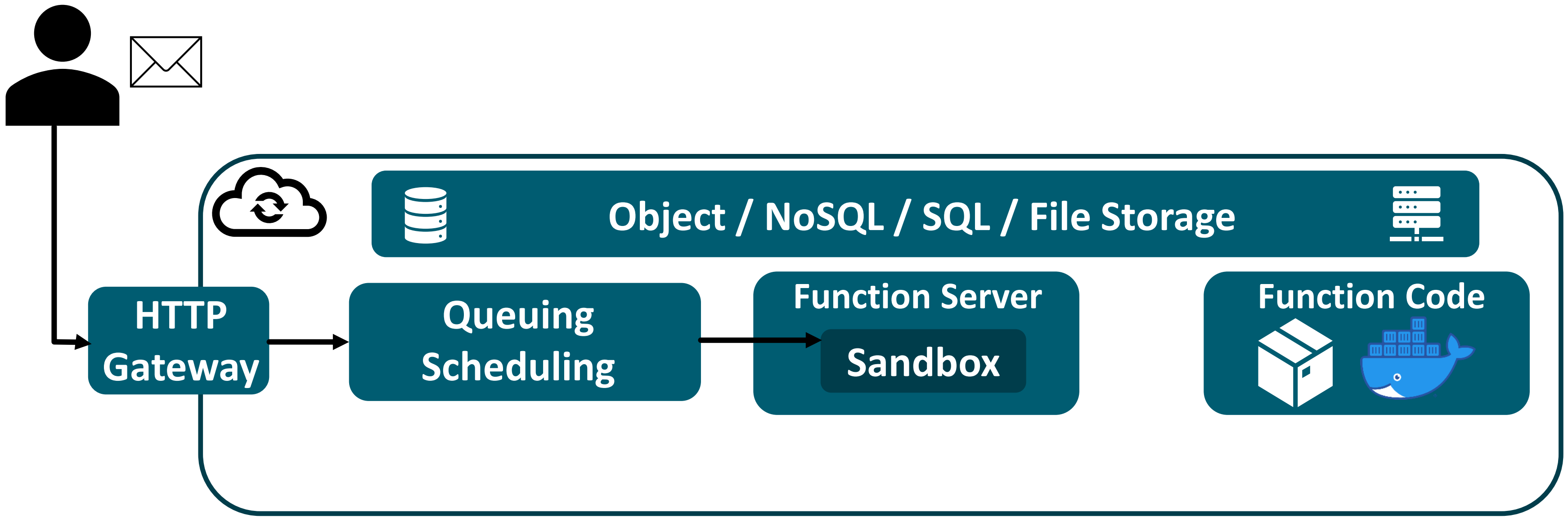
Warm

print(var)

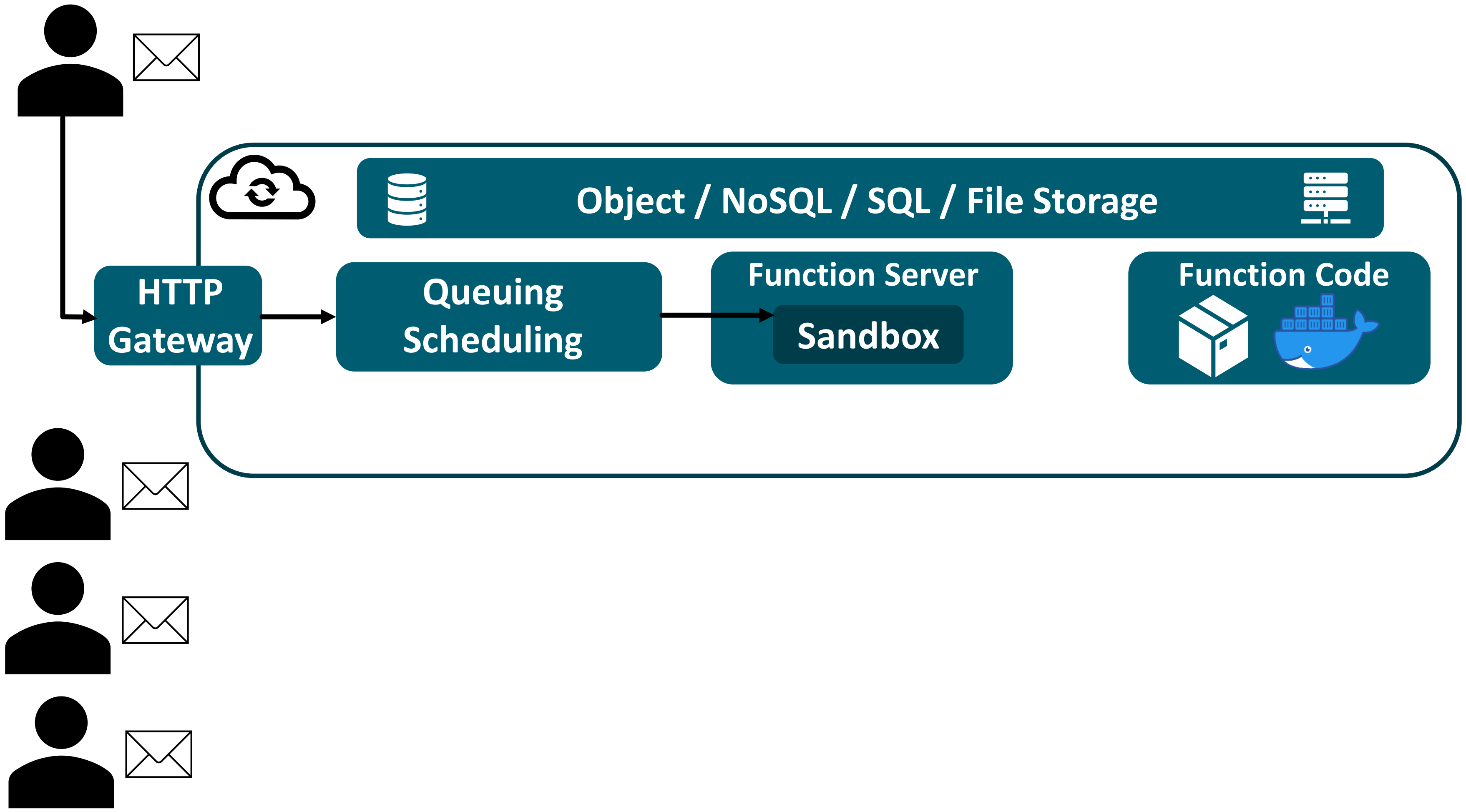
cat /tmp/dir

?

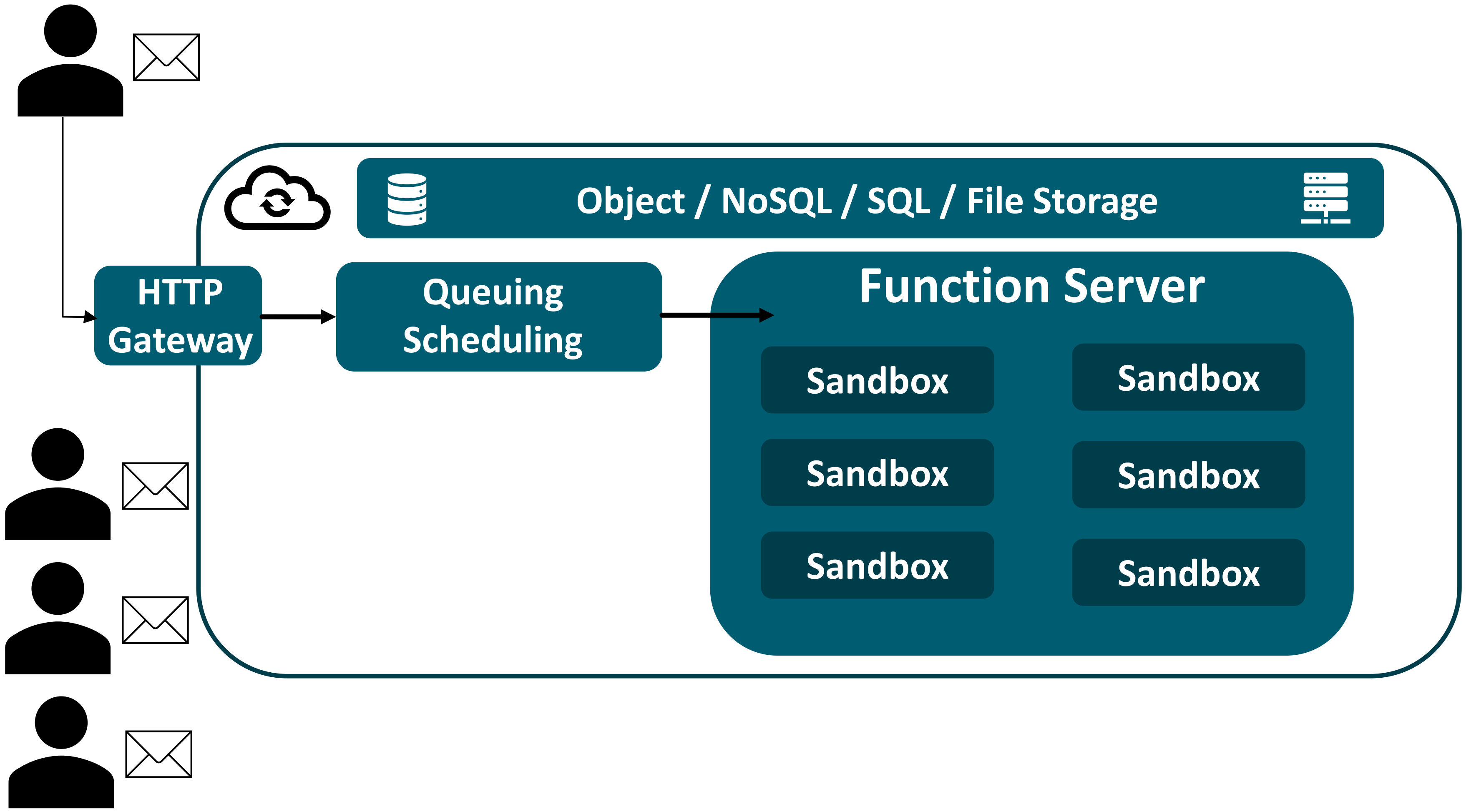
How does Function-as-a-Service (FaaS) work?



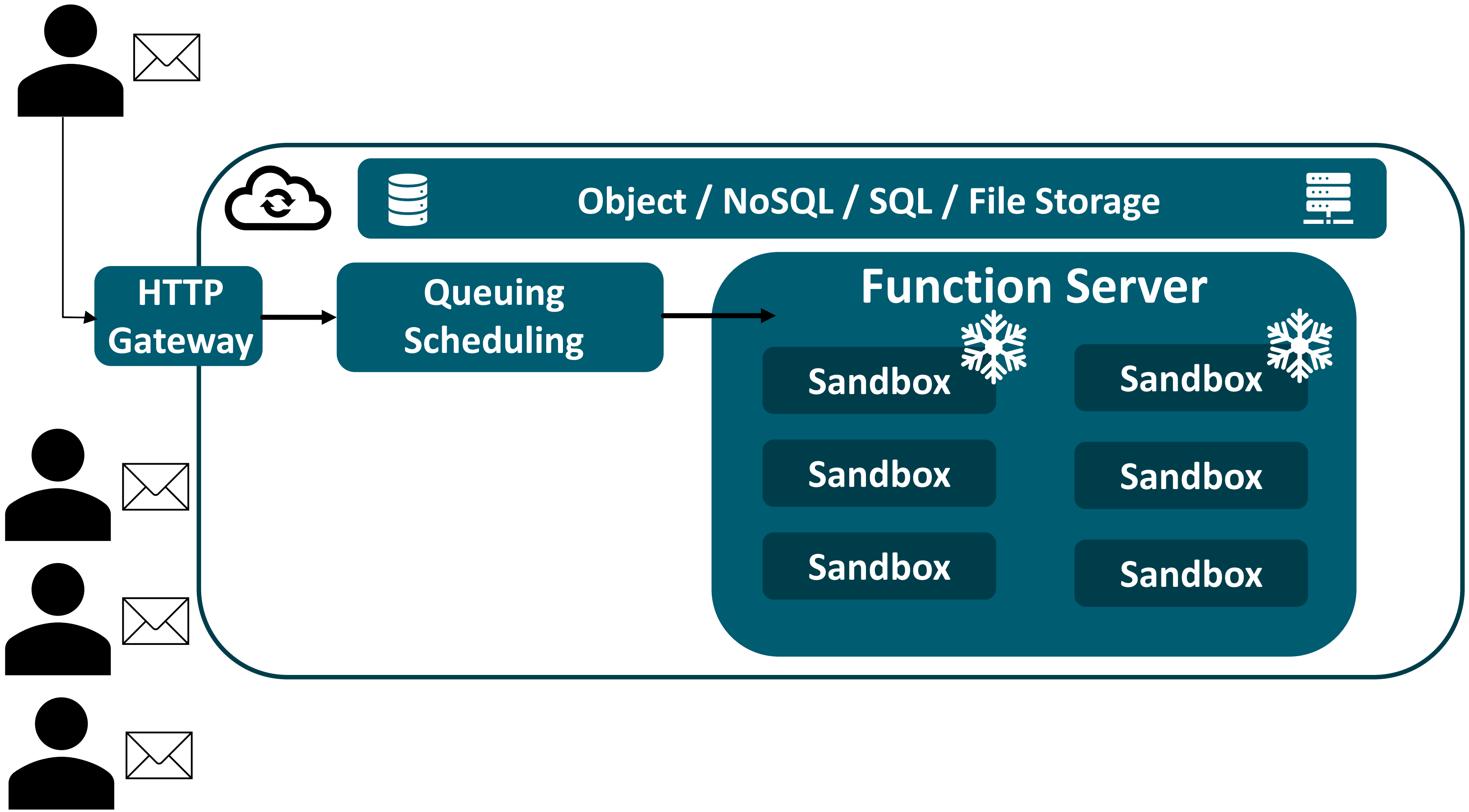
How does Function-as-a-Service (FaaS) work?



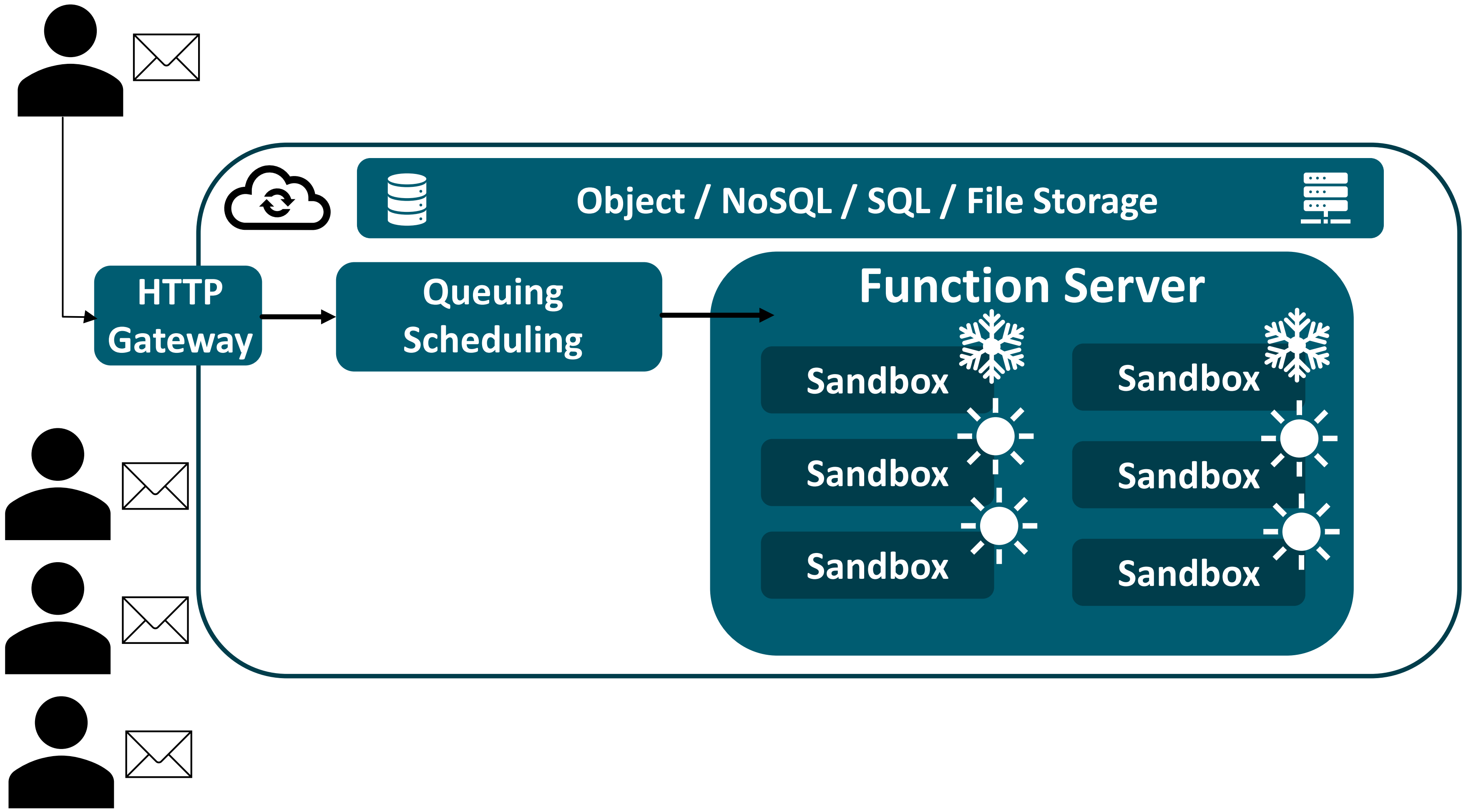
How does Function-as-a-Service (FaaS) work?



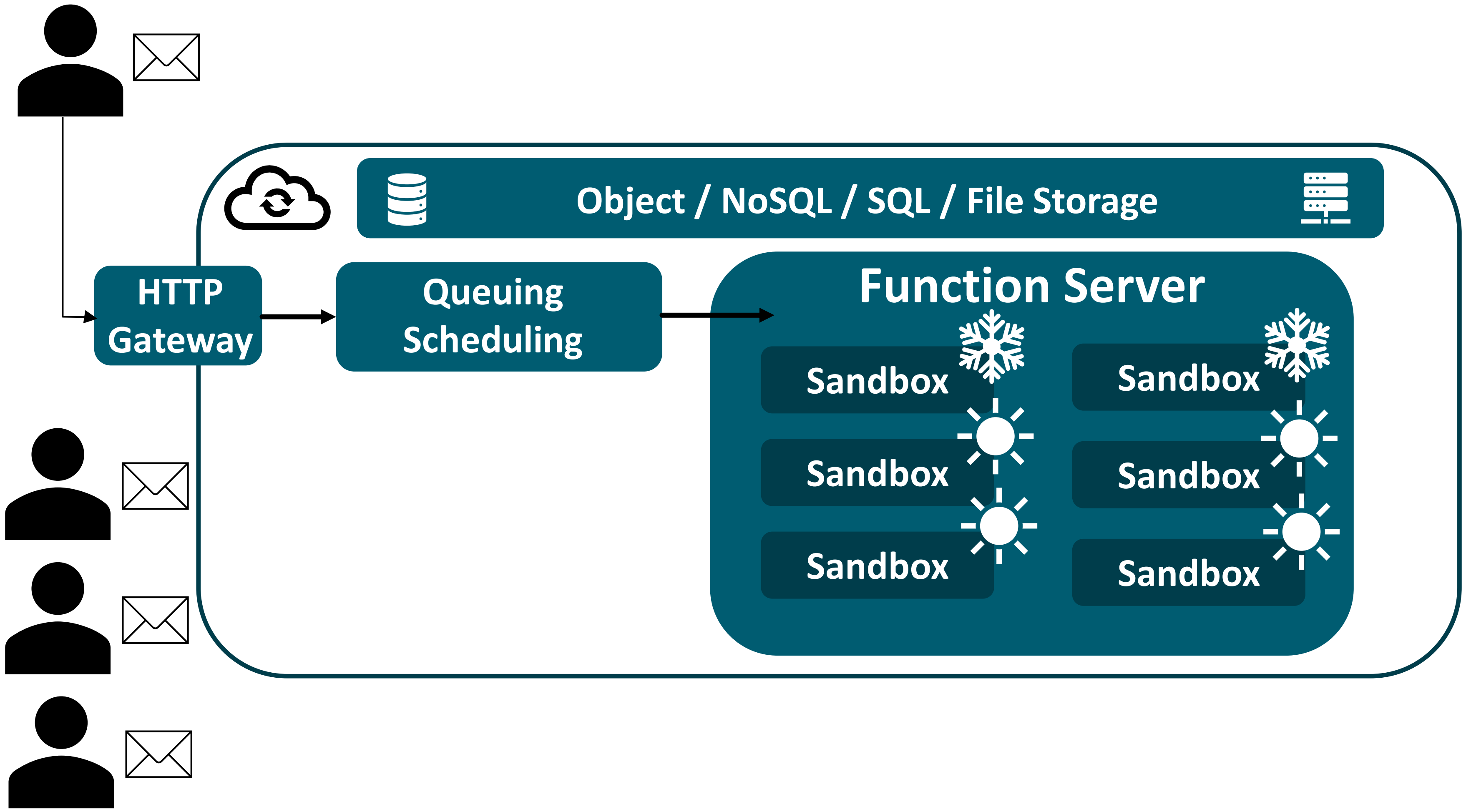
How does Function-as-a-Service (FaaS) work?



How does Function-as-a-Service (FaaS) work?



How does Function-as-a-Service (FaaS) work?



How much time a sandbox stays warm?

Will the same sandbox be reused?

How are parallel invocations handled?

How much time it takes to invoke a function?

What is the overhead of a cold start?

How does Function-as-a-Service (FaaS) work?

HTTP
Gateway

How much time a sandbox stays warm?

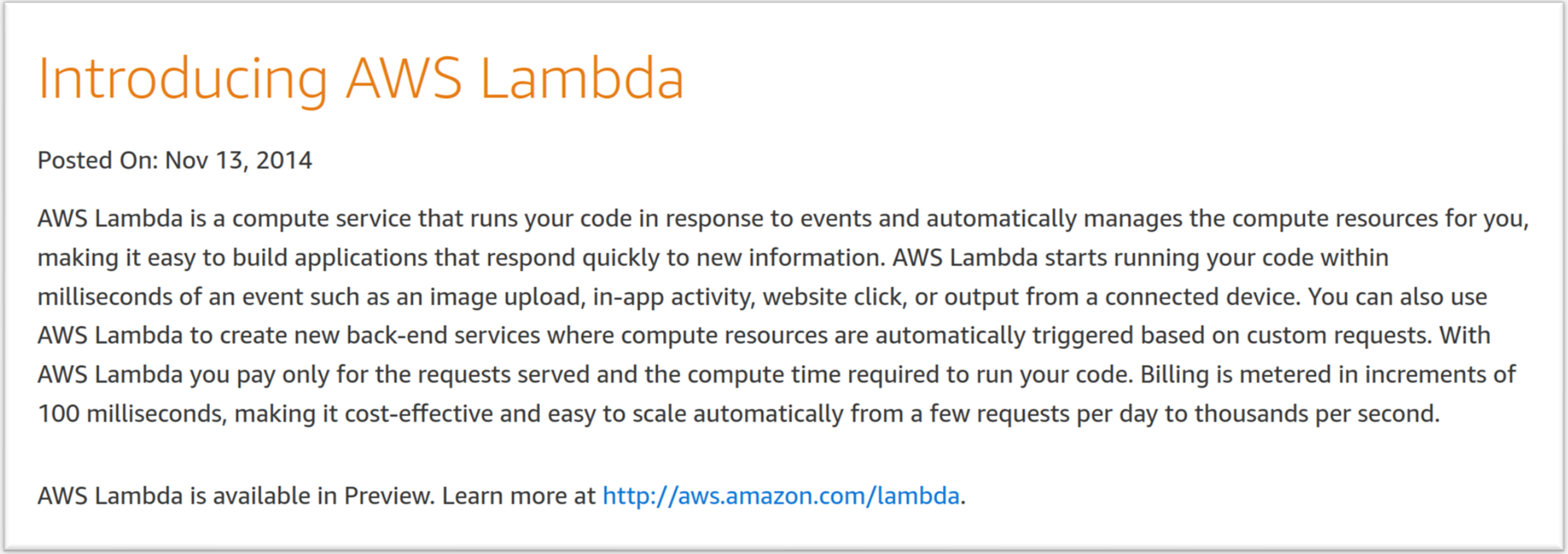
Will the same sandbox be reused?

How are parallel invocations handled?

How much time it takes to invoke a function?

What is the overhead of a cold start?

Humble Beginnings



Introducing AWS Lambda

Posted On: Nov 13, 2014

AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources for you, making it easy to build applications that respond quickly to new information. AWS Lambda starts running your code within milliseconds of an event such as an image upload, in-app activity, website click, or output from a connected device. You can also use AWS Lambda to create new back-end services where compute resources are automatically triggered based on custom requests. With AWS Lambda you pay only for the requests served and the compute time required to run your code. Billing is metered in increments of 100 milliseconds, making it cost-effective and easy to scale automatically from a few requests per day to thousands per second.

AWS Lambda is available in Preview. Learn more at http://aws.amazon.com/lambda.

Humble Beginnings

“Toy”
Serverless

Humble Beginnings

“Toy”
Serverless

⚠ Functions run in single-tenant VMs

Humble Beginnings

“Toy” Serverless

⚠ Functions run in single-tenant VMs

⚠ One language: Node

Humble Beginnings

“Toy”
Serverless

⚠ Functions run in single-tenant VMs

⚠ One language: Node

⚠ Deployment as zip files <= 250 MB

Humble Beginnings

“Toy” Serverless

⚠ Functions run in single-tenant VMs

⚠ One language: Node

⚠ Deployment as zip files <= 250 MB

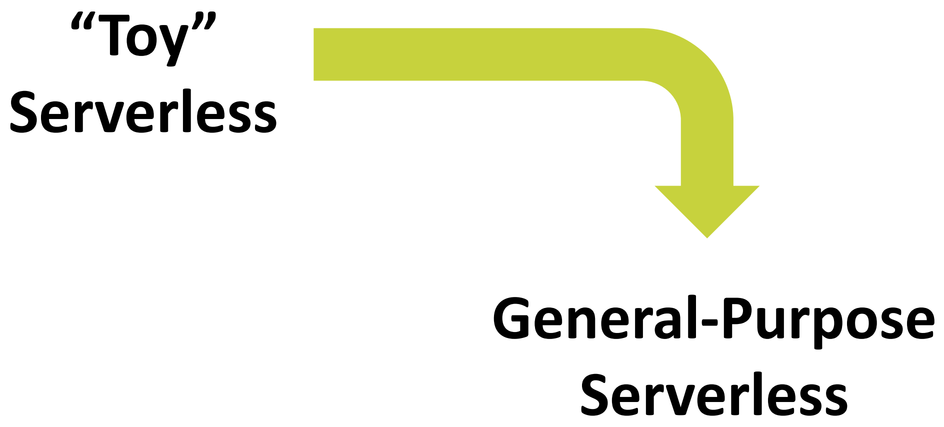
⚠ 1 vCPU, up to 1 GB memory

Humble Beginnings

**“Toy”
Serverless**

- ⚠ Functions run in single-tenant VMs
- ⚠ One language: Node
- ⚠ Deployment as zip files <= 250 MB
- ⚠ 1 vCPU, up to 1 GB memory
- ⚠ Simple HTTP triggers

Serverless Functions Grew Larger and More Powerful



“Toy” Serverless

General-Purpose Serverless

⚠ Functions run in single-tenant VMs

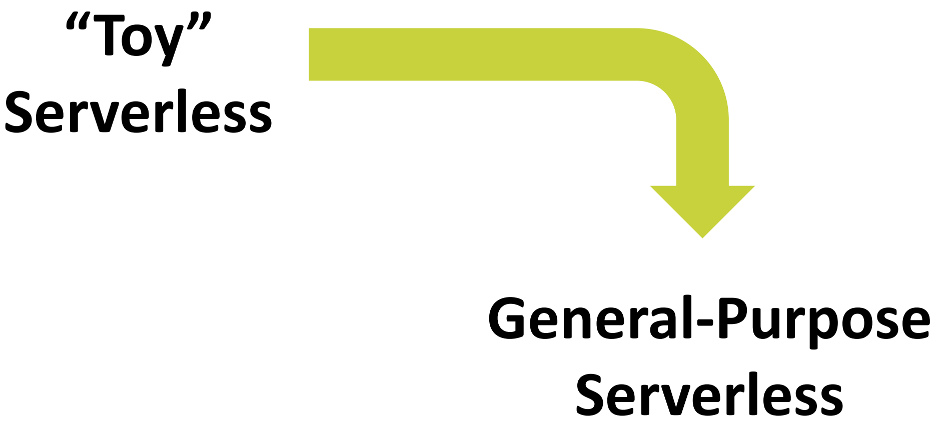
⚠ One language: Node

⚠ Deployment as zip files <= 250 MB

⚠ 1 vCPU, up to 1 GB memory

⚠ Simple HTTP triggers

Serverless Functions Grew Larger and More Powerful



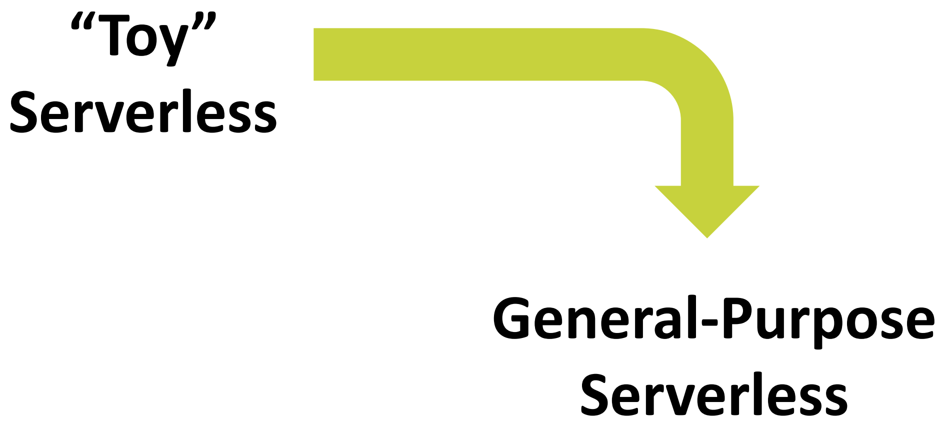
“Toy” Serverless

General-Purpose Serverless

- ⚠ Functions run in single-tenant VMs
- ⚠ One language: Node
- ⚠ Deployment as zip files <= 250 MB
- ⚠ 1 vCPU, up to 1 GB memory
- ⚠ Simple HTTP triggers

⏩ Dedicated sandboxes and microVMs

Serverless Functions Grew Larger and More Powerful



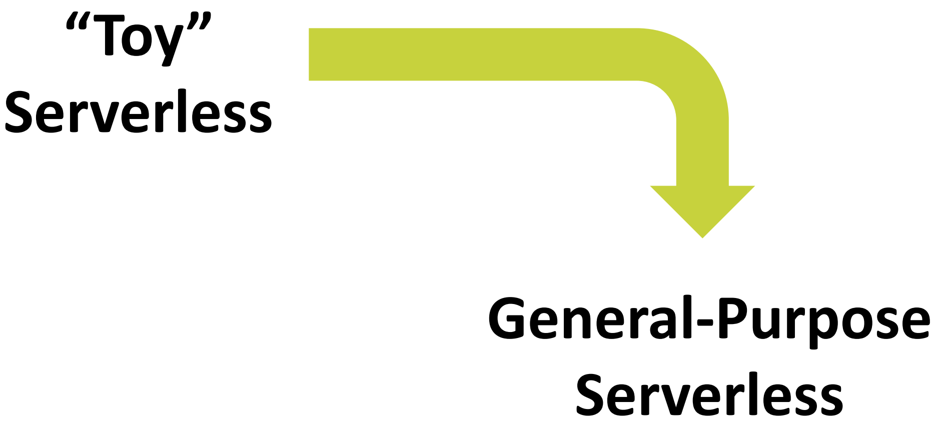
“Toy” Serverless

General-Purpose Serverless

- ⚠ Functions run in single-tenant VMs
- ⚠ One language: Node
- ⚠ Deployment as zip files <= 250 MB
- ⚠ 1 vCPU, up to 1 GB memory
- ⚠ Simple HTTP triggers

- ⏩ Dedicated sandboxes and microVMs
- ⏩ Many languages, including compiled

Serverless Functions Grew Larger and More Powerful



“Toy” Serverless

General-Purpose Serverless

- ⚠ Functions run in single-tenant VMs
- ⚠ One language: Node
- ⚠ Deployment as zip files <= 250 MB
- ⚠ 1 vCPU, up to 1 GB memory
- ⚠ Simple HTTP triggers

- ⏩ Dedicated sandboxes and microVMs
- ⏩ Many languages, including compiled
- ⏩ Containers & snapshots

Serverless Functions Grew Larger and More Powerful

“Toy” Serverless

General-Purpose Serverless

AWS Lambda Today: up to 60,000 vCPUs

- ⚠ Functions run in single-tenant VMs
- ⚠ One language: Node
- ⚠ Deployment as zip files <= 250 MB
- ⚠ 1 vCPU, up to 1 GB memory
- ⚠ Simple HTTP triggers

- ⏩ Dedicated sandboxes and microVMs
- ⏩ Many languages, including compiled
- ⏩ Containers & snapshots
- ⏩ Multi-core functions

Serverless Functions Grew Larger and More Powerful

“Toy” Serverless

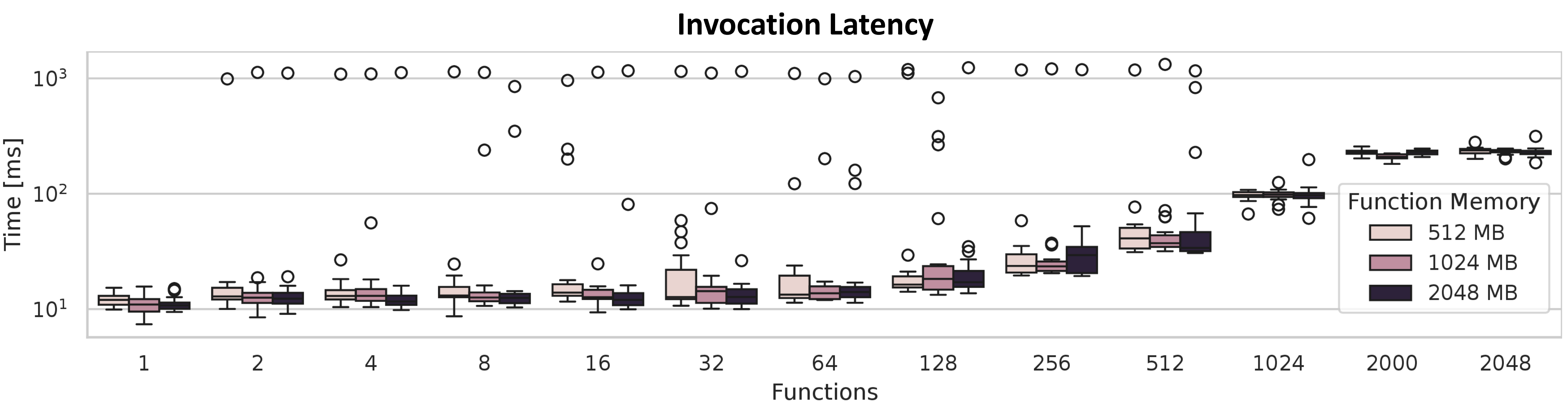
General-Purpose Serverless

AWS Lambda Today: up to 60,000 vCPUs

- ⚠ Functions run in single-tenant VMs
- ⚠ One language: Node
- ⚠ Deployment as zip files <= 250 MB
- ⚠ 1 vCPU, up to 1 GB memory
- ⚠ Simple HTTP triggers

- ⏩ Dedicated sandboxes and microVMs
- ⏩ Many languages, including compiled
- ⏩ Containers & snapshots
- ⏩ Multi-core functions
- ⏩ Serverless workflows

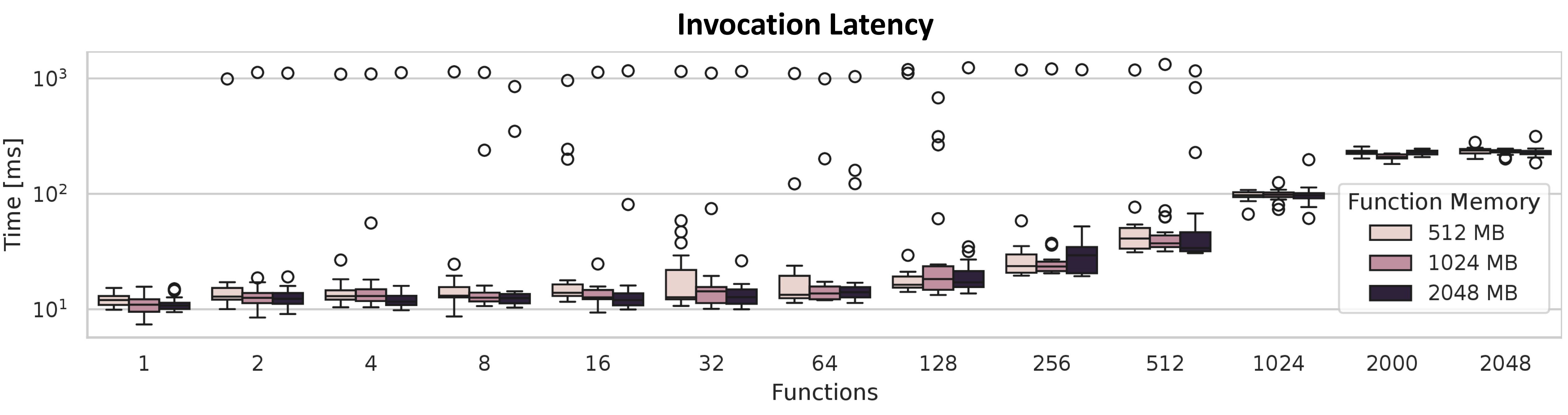
Serverless Functions Grew Larger and More Powerful



- One language: Node
- Deployment as zip files <= 250 MB
- 1 vCPU, up to 1 GB memory
- Simple HTTP triggers

- Many languages, including compiled
- Containers & snapshots
- Multi-core functions
- Serverless workflows

Serverless Functions Grew Larger and More Powerful



One language: Node

Many languages, including compiled

Learn more at our paper talk tomorrow, at 10:00!

"Cppless: Single-Source and High-Performance Serverless Programming in C++"

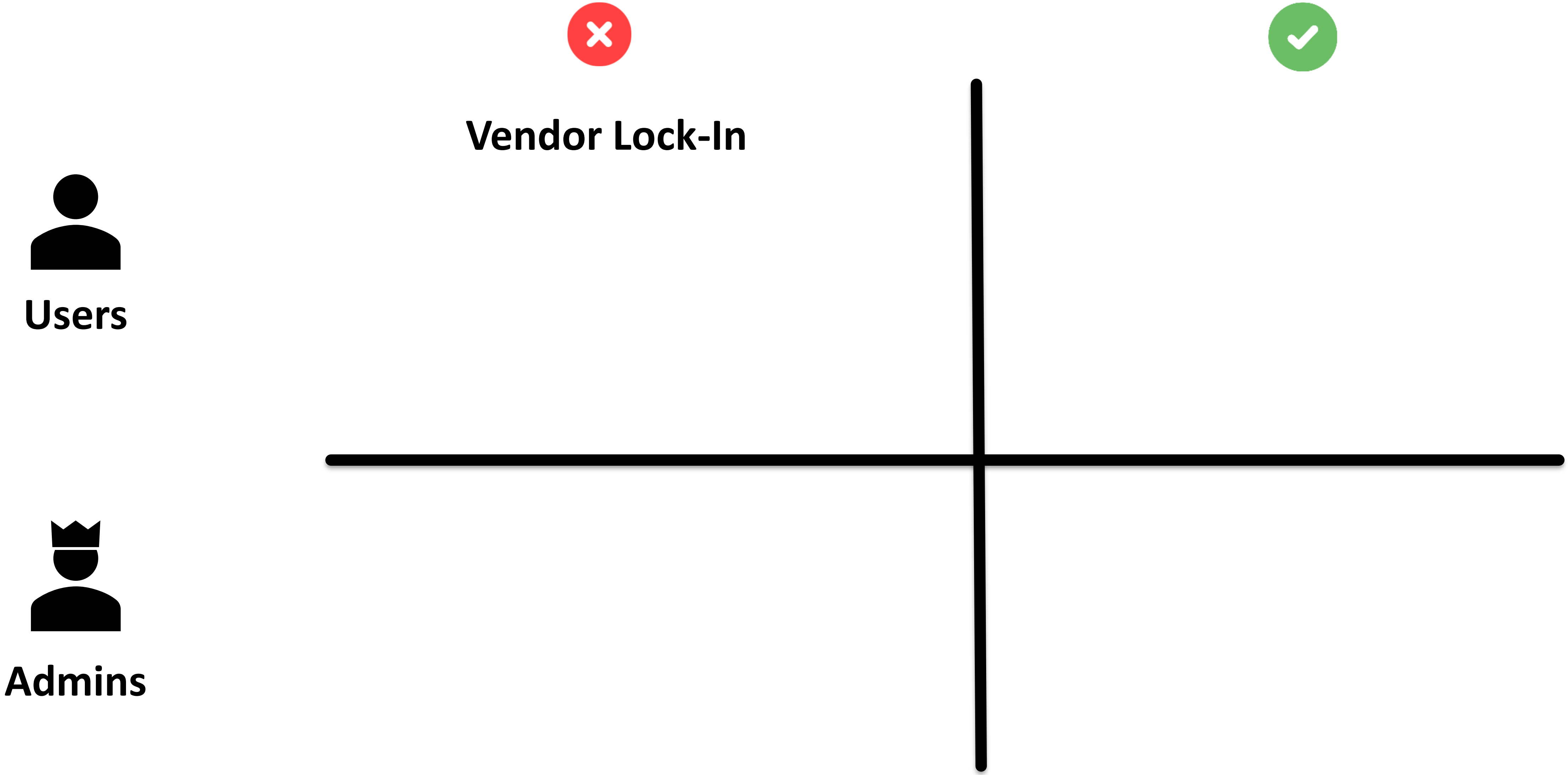
Simple HTTP triggers

Serverless workflows

Good & Bad of Serverless

	✗	✓
Users		
Admins		

Good & Bad of Serverless



✗

✓

Vendor Lock-In

Users

Admins

Good & Bad of Serverless

	✗	✓
Users	Vendor Lock-In Performance Variability	
Admins		

Good & Bad of Serverless

	✗	✓
Users	Vendor Lock-In Performance Variability Stateless & Data locality	
Admins		

Good & Bad of Serverless

	✗	✓
Users	Vendor Lock-In Performance Variability Stateless & Data locality	Simplicity
Admins		

Good & Bad of Serverless

	✗	✓
Users	Vendor Lock-In Performance Variability Stateless & Data locality	Simplicity Scalability
Admins		

Good & Bad of Serverless

	✗	✓
Users	Vendor Lock-In Performance Variability Stateless & Data locality	Simplicity Scalability Pay-as-you-go Billing
Admins		

Good & Bad of Serverless

	✗	✓
Users	Vendor Lock-In Performance Variability Stateless & Data locality	Simplicity Scalability Pay-as-you-go Billing
Admins	Hardware Heterogeneity	

Good & Bad of Serverless

	✗	✓
Users	Vendor Lock-In Performance Variability Stateless & Data locality	Simplicity Scalability Pay-as-you-go Billing
Admins	Hardware Heterogeneity Platform Costs	

Good & Bad of Serverless

	✗	✓
Users	Vendor Lock-In Performance Variability Stateless & Data locality	Simplicity Scalability Pay-as-you-go Billing
Admins	Hardware Heterogeneity Platform Costs	High Utilization

Good & Bad of Serverless

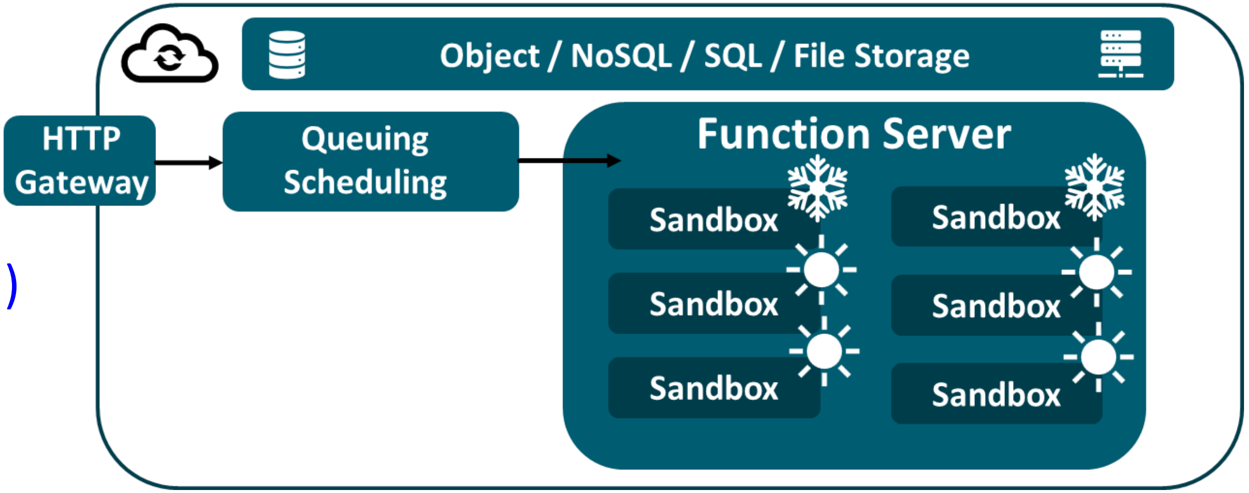
	✗	✓
Users	Vendor Lock-In Performance Variability Stateless & Data locality	Simplicity Scalability Pay-as-you-go Billing
Admins	Hardware Heterogeneity Platform Costs	High Utilization Higher Margins

Serverless Platforms in Practice

```python
def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    new_data = process_logic(request['op'], data)

    stamp = cloud_storage.write(request['id'], new_data)

    return stamp
```

+

Configuration: CPU RAM



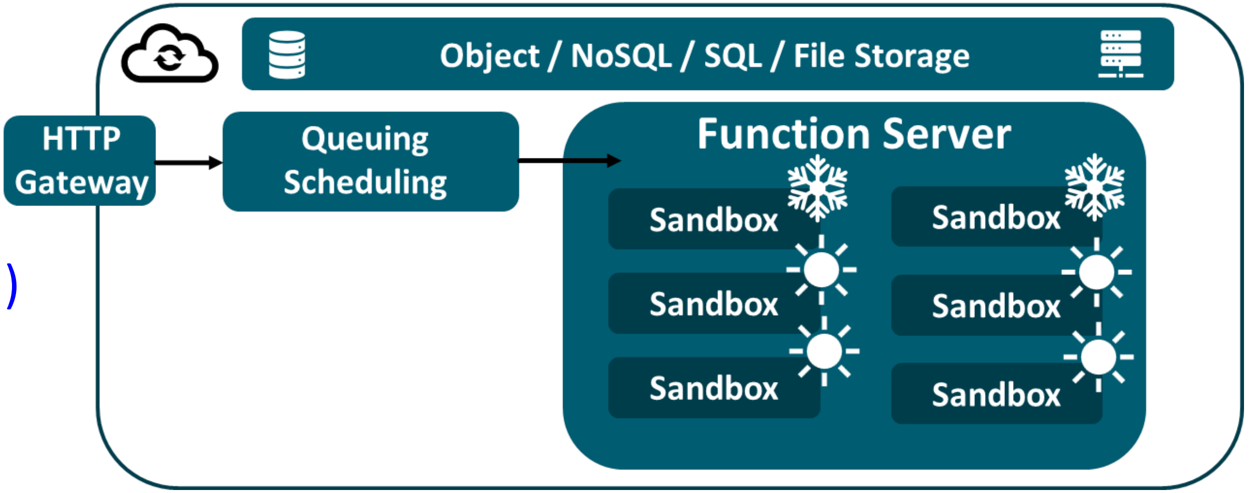
Serverless Platforms in Practice

```
def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    new_data = process_logic(request['op'], data)

    stamp = cloud_storage.write(request['id'], new_data)

    return stamp
```

+

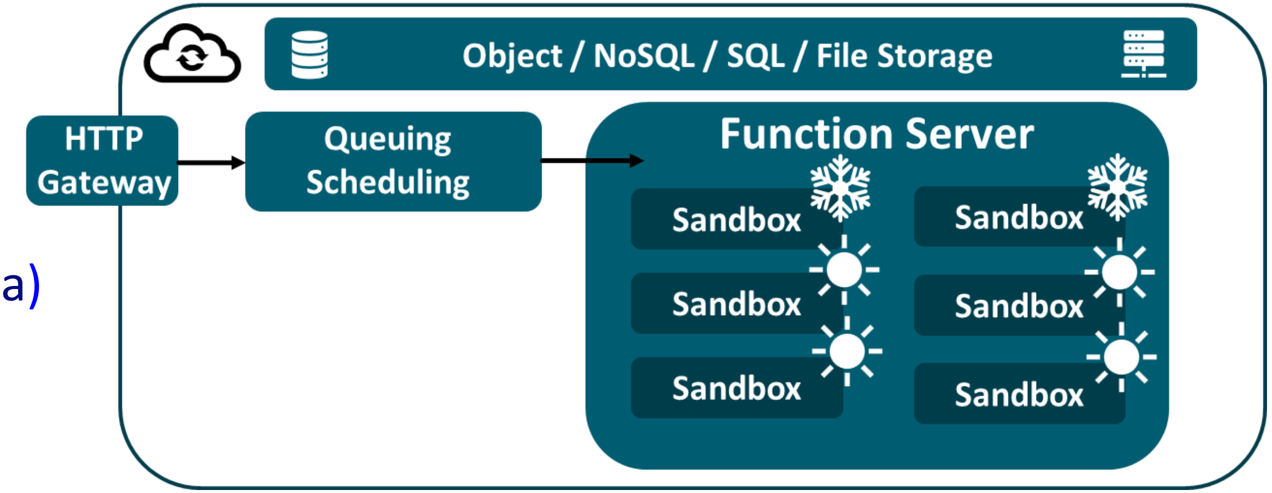
Configuration: CPU RAM

↓

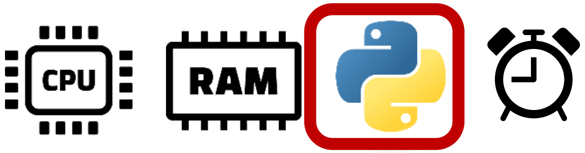


Serverless Platforms in Practice

```
def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    new_data = process_logic(request['op'], data)

    stamp = cloud_storage.write(request['id'], new_data)

    return stamp
```



+

Configuration: 

⬇ ☁

3.9? 3.11? PyPy?

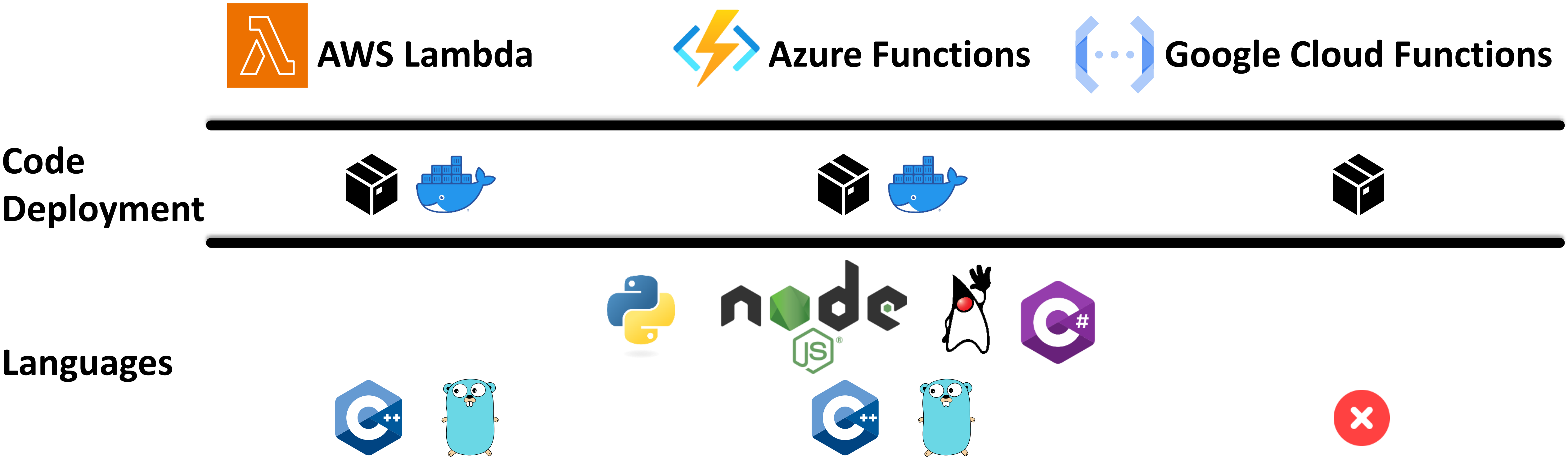
Serverless Platforms in Practice: Functions

AWS Lambda Azure Functions Google Cloud Functions

Serverless Platforms in Practice: Functions

	AWS Lambda	Azure Functions	Google Cloud Functions
Code Deployment	📦 🐳	📦 🐳	📦

Serverless Platforms in Practice: Functions



AWS Lambda

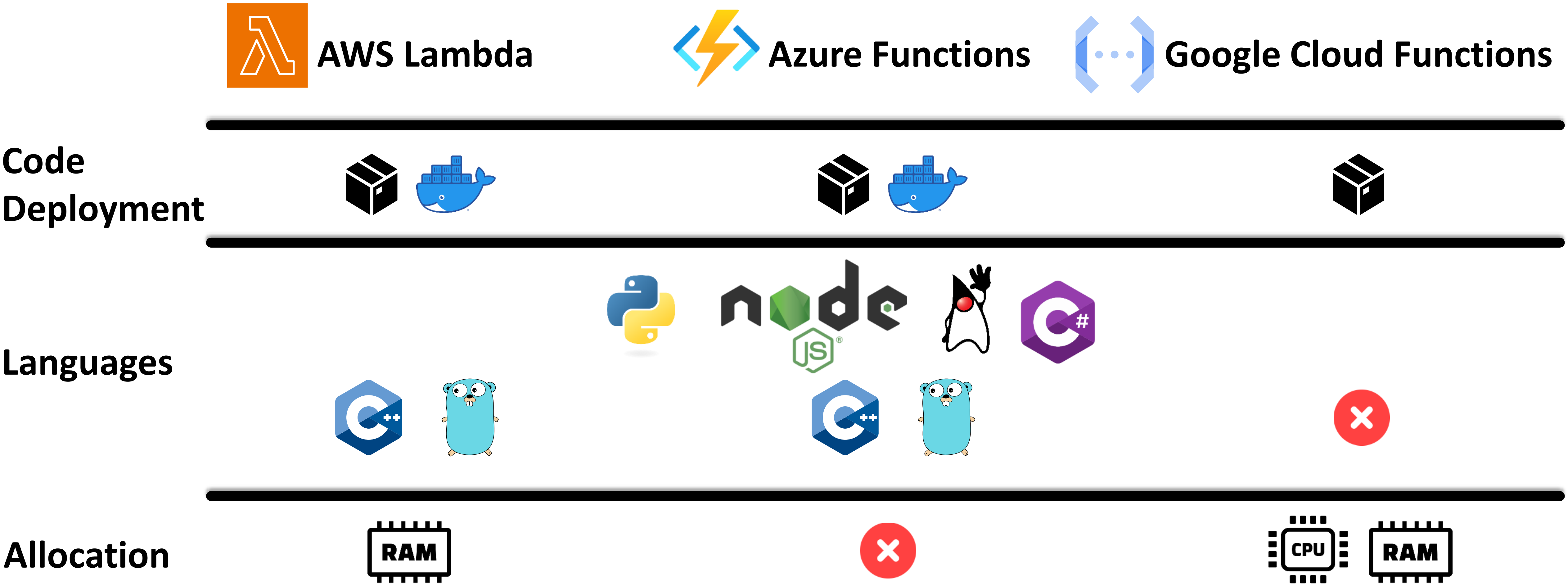
Azure Functions

Google Cloud Functions

Code Deployment

Languages

Serverless Platforms in Practice: Functions



AWS Lambda

Azure Functions

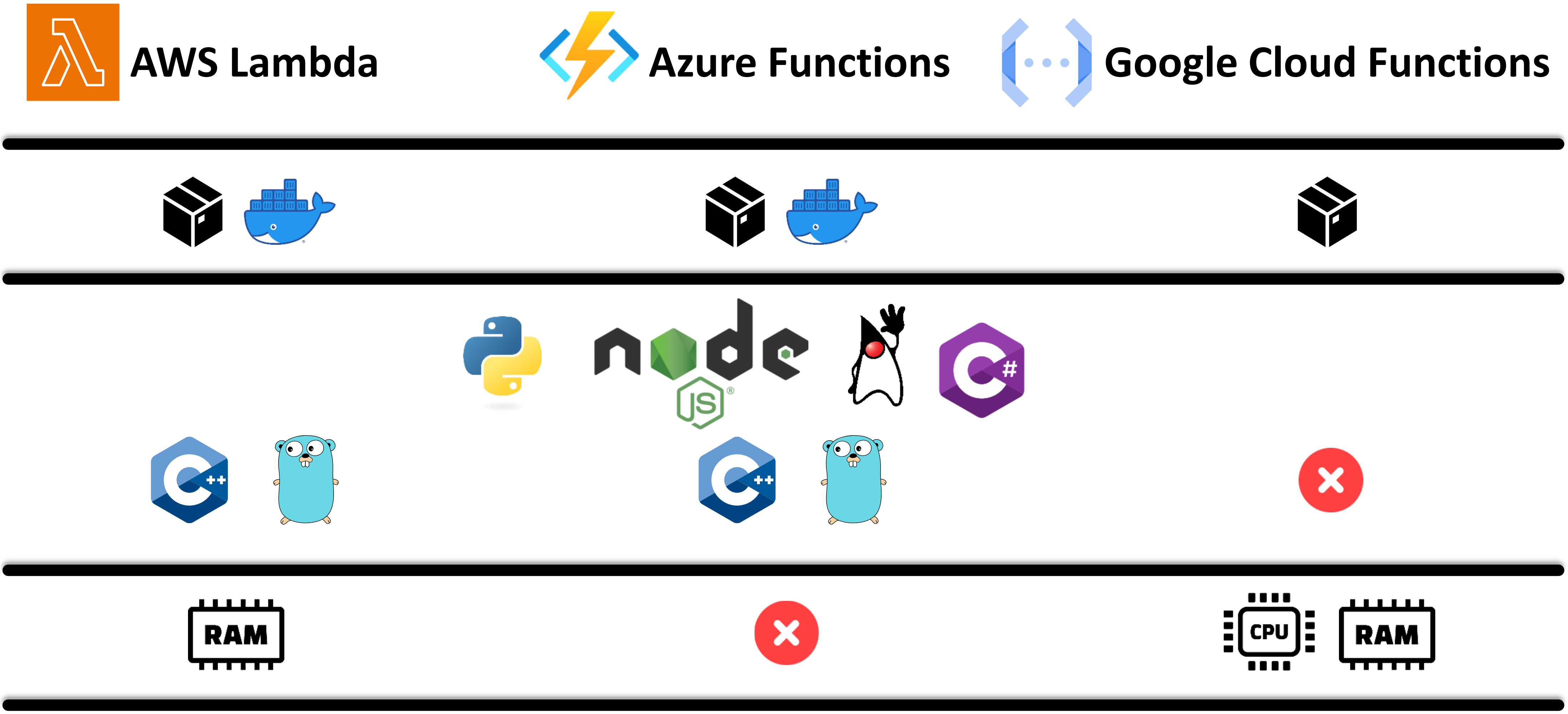
Google Cloud Functions

Code Deployment

Languages

Allocation

Serverless Platforms in Practice: Functions



Code Deployment

Languages

Allocation

	AWS Lambda	Azure Functions	Google Cloud Functions
Execution	One-to-one	One-to-many	One-to-many

Serverless Platforms in Practice: Cost

Serverless Platforms in Practice: Cost

$

❑ Invocation Fee
❑ Execution Time
❑ Memory
❑ vCPUs

Serverless Platforms in Practice: Cost

$

- □ Invocation Fee
- □ Execution Time
- □ Memory
- □ vCPUs

Time * Memory Allocated

Time * Memory Used

Time * (vCPU + Memory Allocated)

Serverless Platforms in Practice: Cost

$

□ Invocation Fee
□ Execution Time
□ Memory
□ vCPUs

$$

□ Storage Access
□ Network Egress

λ Time * Memory Allocated

<⚡> Time * Memory Used

{···} Time * (vCPU + Memory Allocated)

Serverless Platforms in Practice: Cost

$

- □ Invocation Fee
- □ Execution Time
- □ Memory
- □ vCPUs

$$

- □ Storage Access
- □ Network Egress

$$$

- □ Prewarm Functions
- □ Function Snapshot
- □ Ephemeral Storage

Time * Memory Allocated

Time * Memory Used

Time * (vCPU + Memory Allocated)

Serverless Functions: Simple but Complex

Runtime

Resource Configuration

Triggers

Deployment Package

Serverless Platforms: Simple but Complex

Event-Driven Execution

Scale Down to Zero

Pay-as-you-go Billing

Serverless Platforms: Simple but Complex

Event-Driven Execution

Scale Down to Zero

Pay-as-you-go Billing

Cold & Warm Start

Stateless Functions

Disaggregated Storage

Agenda

- ✓ Part I
  - ✓ What is Serverless?
  - ✓ **Benchmarking Suite SeBS**
  - ✓ **Working with SeBS**
- ✓ Hands-on I: Local Deployment & Storage
- ✓ Part II
  - ✓ Communication and Data
  - ✓ Serverless Workflows
  - ✓ Experiments in SeBS
- ✓ Hands-on II: FaaS Platforms & Experiments
- ✓ Part III
  - ✓ Research Directions in Serverless
  - ✓ Development of SeBS

SeBS Slack

slack **Join SeBS Slack channel: https://serverlessbenchmark.slack.com**

SeBS: The Serverless Benchmark Suite

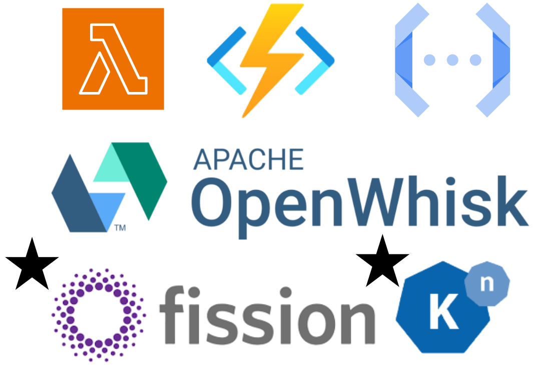
spcl/serverless-benchmarks

"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

SeBS: The Serverless Benchmark Suite

spcl/serverless-benchmarks

Cloud-Agnostic



“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

SeBS: The Serverless Benchmark Suite

spcl/serverless-benchmarks

Cloud-Agnostic

APACHE
OpenWhisk
★
fission
★

Representative Benchmarks

python™
node
JS
★

“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

SeBS: The Serverless Benchmark Suite

spcl/serverless-benchmarks

Cloud-Agnostic

APACHE OpenWhisk

★ fission ★

Representative Benchmarks

python™

node JS ★

Automatic Experiments

Performance & Cost
Invocation Overhead
Container Eviction
Communication★

“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

SeBS: The Serverless Benchmark Suite

spcl/serverless-benchmarks

Cloud-Agnostic

APACHE OpenWhisk

★ fission ★

Representative Benchmarks

python™

node JS ★

Automatic Experiments

Performance & Cost
Invocation Overhead
Container Eviction
Communication★

Insights

$

"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

SeBS: The Serverless Benchmark Suite

spcl/serverless-benchmarks

Cloud-Agnostic

APACHE OpenWhisk

★ fission ★

Representative Benchmarks

python™

node JS

★

Automatic Experiments

Performance & Cost
Invocation Overhead
Container Eviction
Communication★

Insights

$

Adoption & Community

90+ forks
20+ contributors

Google Summer of Code

"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

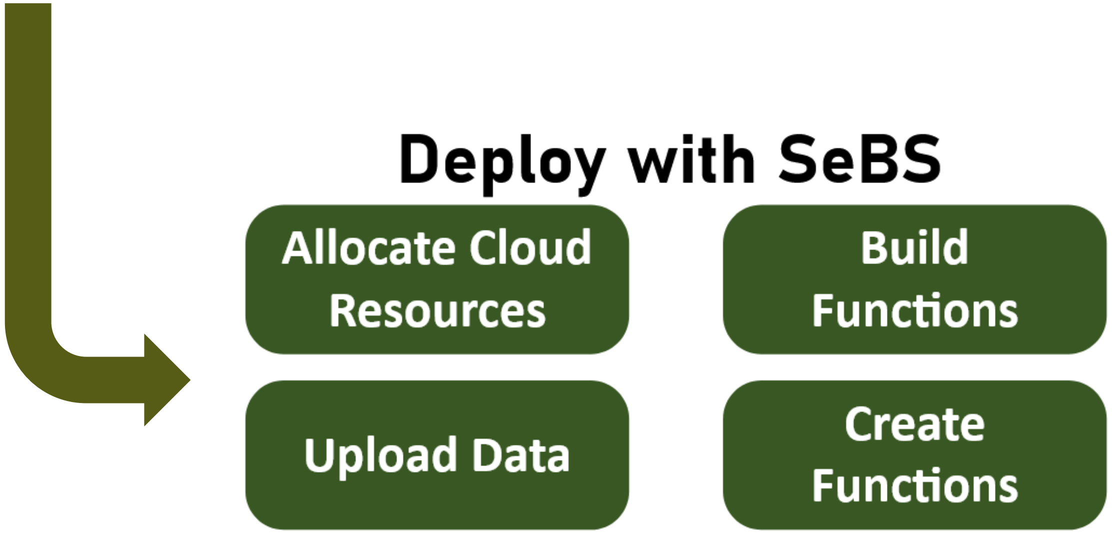
SeBS: The Serverless Benchmark Suite

“Which platform is best for my workload? Can my application scale? What are the performance limits of each platform?”

“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

SeBS: The Serverless Benchmark Suite

“Which platform is best for my workload? Can my application scale? What are the performance limits of each platform?”



Deploy with SeBS

Allocate Cloud Resources

Build Functions

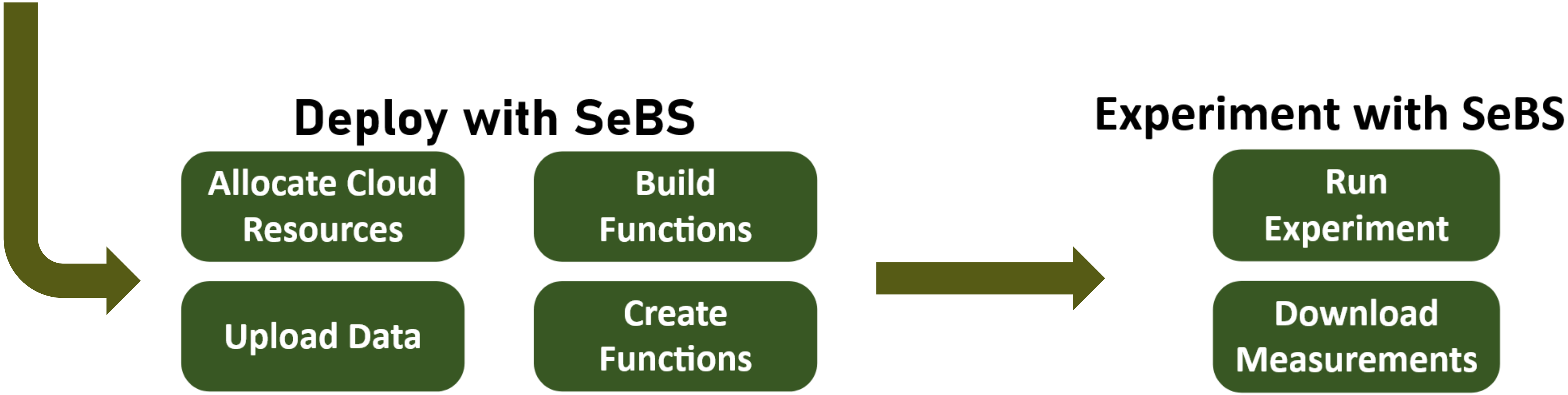
Upload Data

Create Functions

“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

SeBS: The Serverless Benchmark Suite

"Which platform is best for my workload? Can my application scale? What are the performance limits of each platform?"

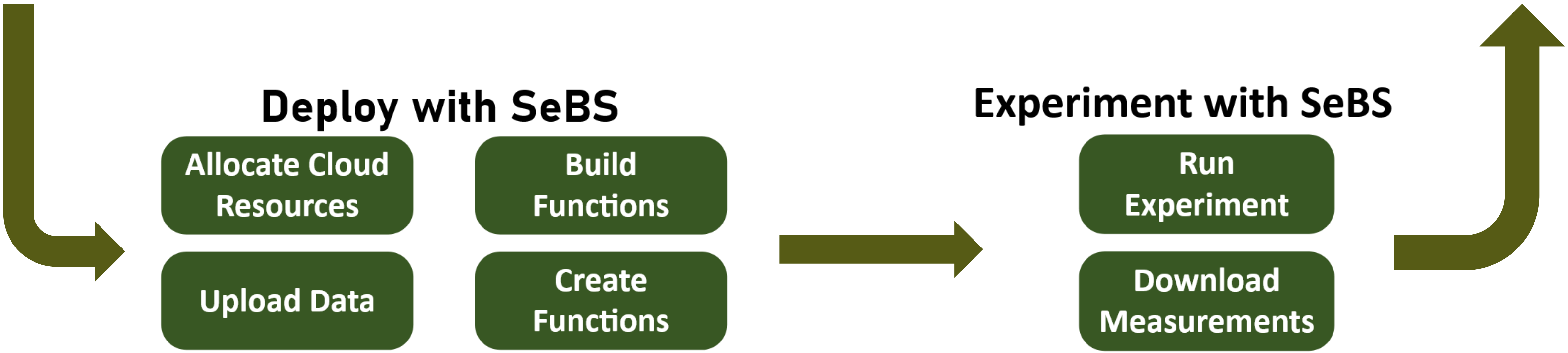


"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

SeBS: The Serverless Benchmark Suite

“Which platform is best for my workload? Can my application scale? What are the performance limits of each platform?”

“The end-to-end latency is not what I expected because of lower scalability and variable I/O performance.”



Deploy with SeBS

- Allocate Cloud Resources
- Build Functions
- Upload Data
- Create Functions

Experiment with SeBS

- Run Experiment
- Download Measurements

“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

SeBS: The Serverless Benchmark Suite

"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

SeBS: The Serverless Benchmark Suite

Principles

- ✓ Usability
- ✓ Portability
- ✓ Extensibility
- ✓ Scientific

"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

SeBS: The Serverless Benchmark Suite

Principles

- ✓ Usability
- ✓ Portability
- ✓ Extensibility
- ✓ Scientific

Applications

- ✓ Realistic workloads
- ✓ Single implementation
- ✓ Varying computational characteristics

"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

SeBS: The Serverless Benchmark Suite

Principles

- ✓ Usability
- ✓ Portability
- ✓ Extensibility
- ✓ Scientific

Applications

- ✓ Realistic workloads
- ✓ Single implementation
- ✓ Varying computational characteristics

Metrics

- ✓ Cloud time
- ✓ User time
- ✓ Resource utilization
- ✓ I/O

“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

SeBS: The Serverless Benchmark Suite

Principles

- ✓ Usability
- ✓ Portability
- ✓ Extensibility
- ✓ Scientific

Applications

- ✓ Realistic workloads
- ✓ Single implementation
- ✓ Varying computational characteristics

Metrics

- ✓ Cloud time
- ✓ User time
- ✓ Resource utilization
- ✓ I/O

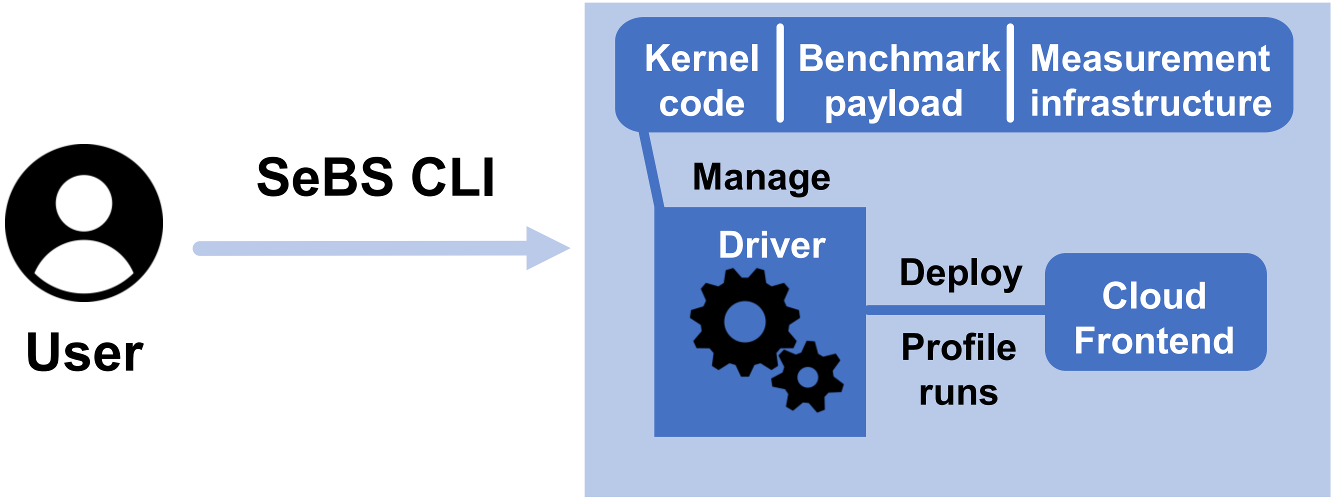
Experiments

- ✓ Performance and cost
- ✓ FaaS vs IaaS
- ✓ Invocation overhead
- ✓ Container eviction

“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

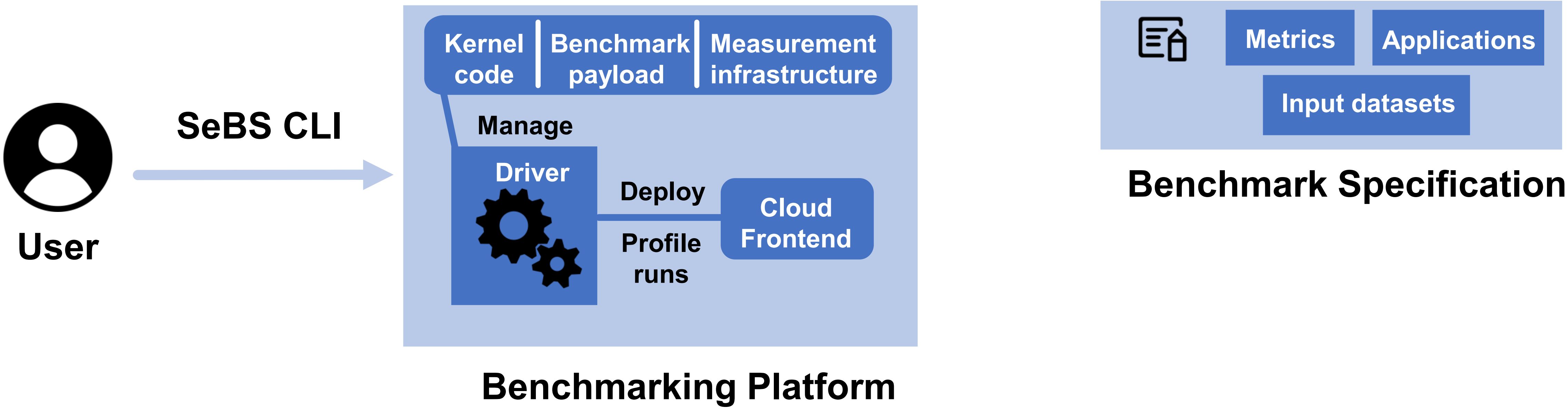
SeBS: Architecture

SeBS: Architecture

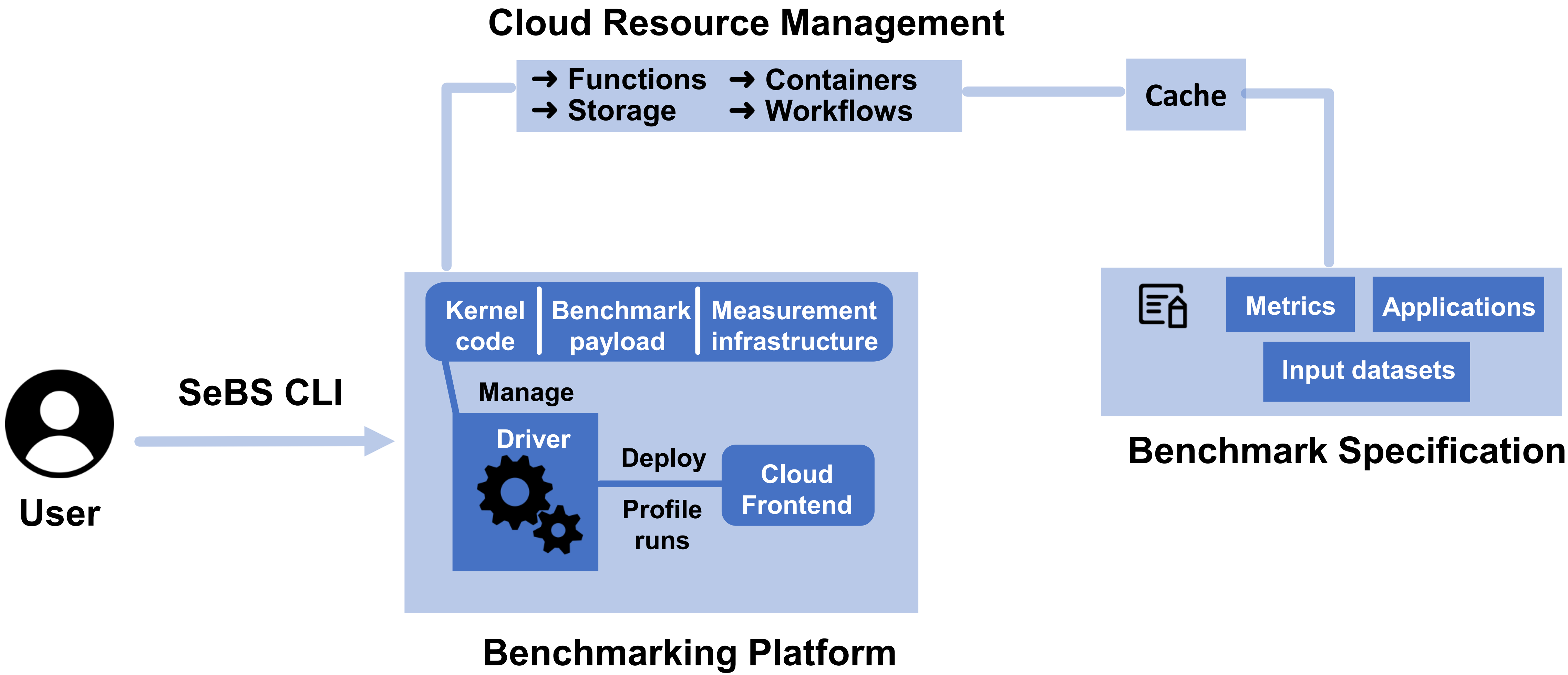


Benchmarking Platform

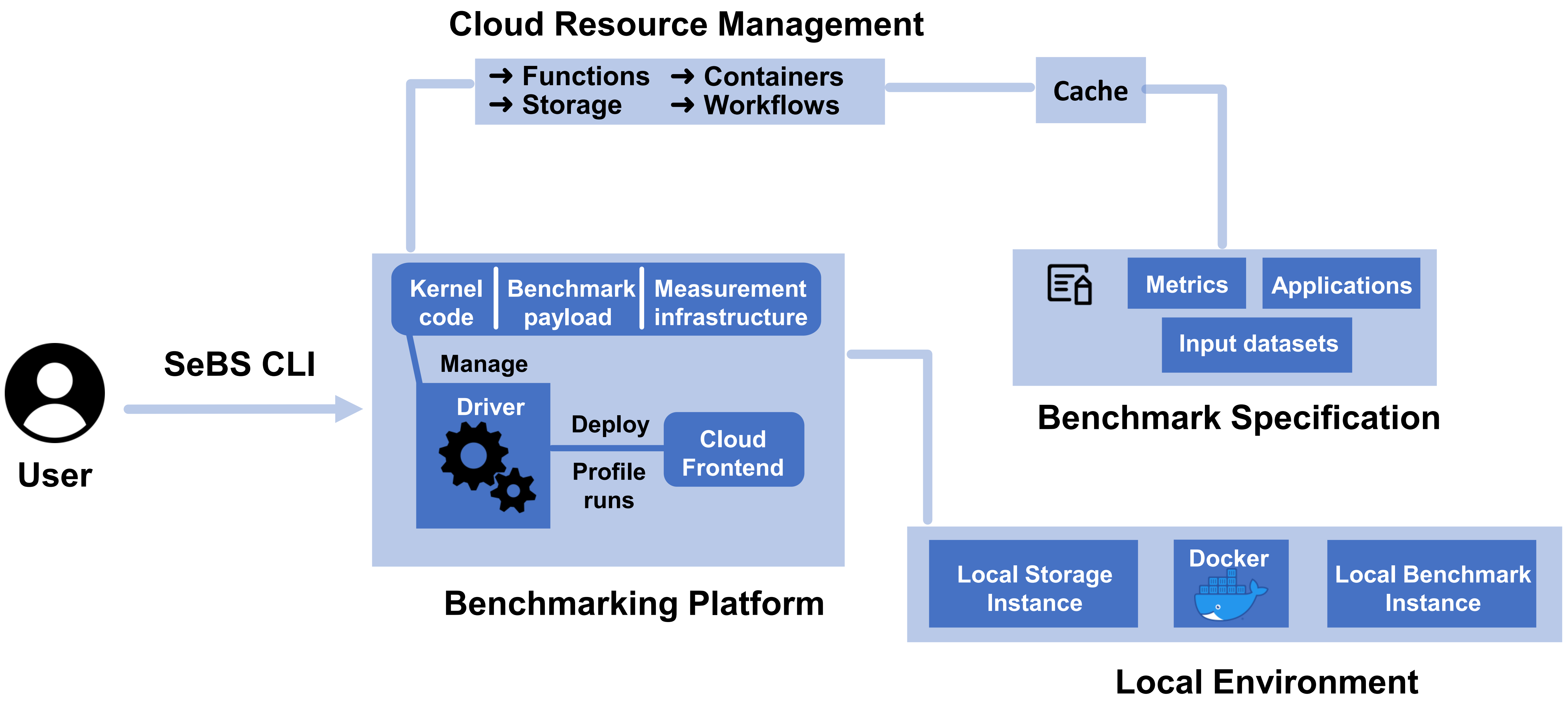
SeBS: Architecture



SeBS: Architecture



SeBS: Architecture



SeBS: Cloud Platforms

SeBS: Cloud Platforms

Big Three: similar capabilities, different semantics.

SeBS: Cloud Platforms

Big Three: similar capabilities, different semantics.

APACHE OpenWhisk™

Open-source FaaS, run on any Kubernetes.

SeBS: Cloud Platforms

Big Three: similar capabilities, different semantics.

Open-source FaaS, run on any Kubernetes.

Experimental support: see GitHub PRs for the branch.

SeBS: Cloud Platforms

Big Three: similar capabilities, different semantics.

APACHE OpenWhisk

Open-source FaaS, run on any Kubernetes.

fission

Experimental support: see GitHub PRs for the branch.

Local

Debugging, local measurements, hardware counters.

SeBS: Benchmarks

Type	Name	Language

SeBS: Benchmarks

Type	Name	Language
Webapps	dynamic-html, uploader, crud-api	Python, Node.js

SeBS: Benchmarks

Type	Name	Language
Webapps	dynamic-html, uploader, crud-api	Python, Node.js
Multimedia	thumbnailer	Python, Node.js, C++

SeBS: Benchmarks

Type	Name	Language
Webapps	dynamic-html, uploader, crud-api	Python, Node.js
Multimedia	thumbnailer	Python, Node.js, C++
Utilities	compression	Python, Node.js*

SeBS: Benchmarks

Type	Name	Language
Webapps	dynamic-html, uploader, crud-api	Python, Node.js
Multimedia	thumbnailer	Python, Node.js, C++
Utilities	compression	Python, Node.js*
Inference	image-recognition	Python, C++

SeBS: Benchmarks

Type	Name	Language
Webapps	dynamic-html, uploader, crud-api	Python, Node.js
Multimedia	thumbnailer	Python, Node.js, C++
Utilities	compression	Python, Node.js*
Inference	image-recognition	Python, C++
Scientific	graph-bfs/pagerank/mst, dna-visualizer	Python, C++

SeBS: Benchmarks

Type	Name	Language
Webapps	dynamic-html, uploader, crud-api	Python, Node.js
Multimedia	thumbnailer	Python, Node.js, C++
Utilities	compression	Python, Node.js*
Inference	image-recognition	Python, C++
Scientific	graph-bfs/pagerank/mst, dna-visualizer	Python, C++
Workflows	trip-booking, video-analysis, …	Python

SeBS: Benchmarks

Type	Name	Language
Webapps	dynamic-html, uploader, crud-api	Python, Node.js
Multimedia	thumbnailer	Python, Node.js, C++
Utilities	compression	Python, Node.js*
Inference	image-recognition	Python, C++
Scientific	graph-bfs/pagerank/mst, dna-visualizer	Python, C++
Workflows	trip-booking, video-analysis, …	Python

Programming Languages

SeBS: Benchmarks

Type	Name	Language
Webapps	dynamic-html, uploader, crud-api	Python, Node.js
Multimedia	thumbnailer	Python, Node.js, C++
Utilities	compression	Python, Node.js*
Inference	image-recognition	Python, C++
Scientific	graph-bfs/pagerank/mst, dna-visualizer	Python, C++
Workflows	trip-booking, video-analysis, …	Python

Programming Languages

Workloads Classes

SeBS: Benchmarks

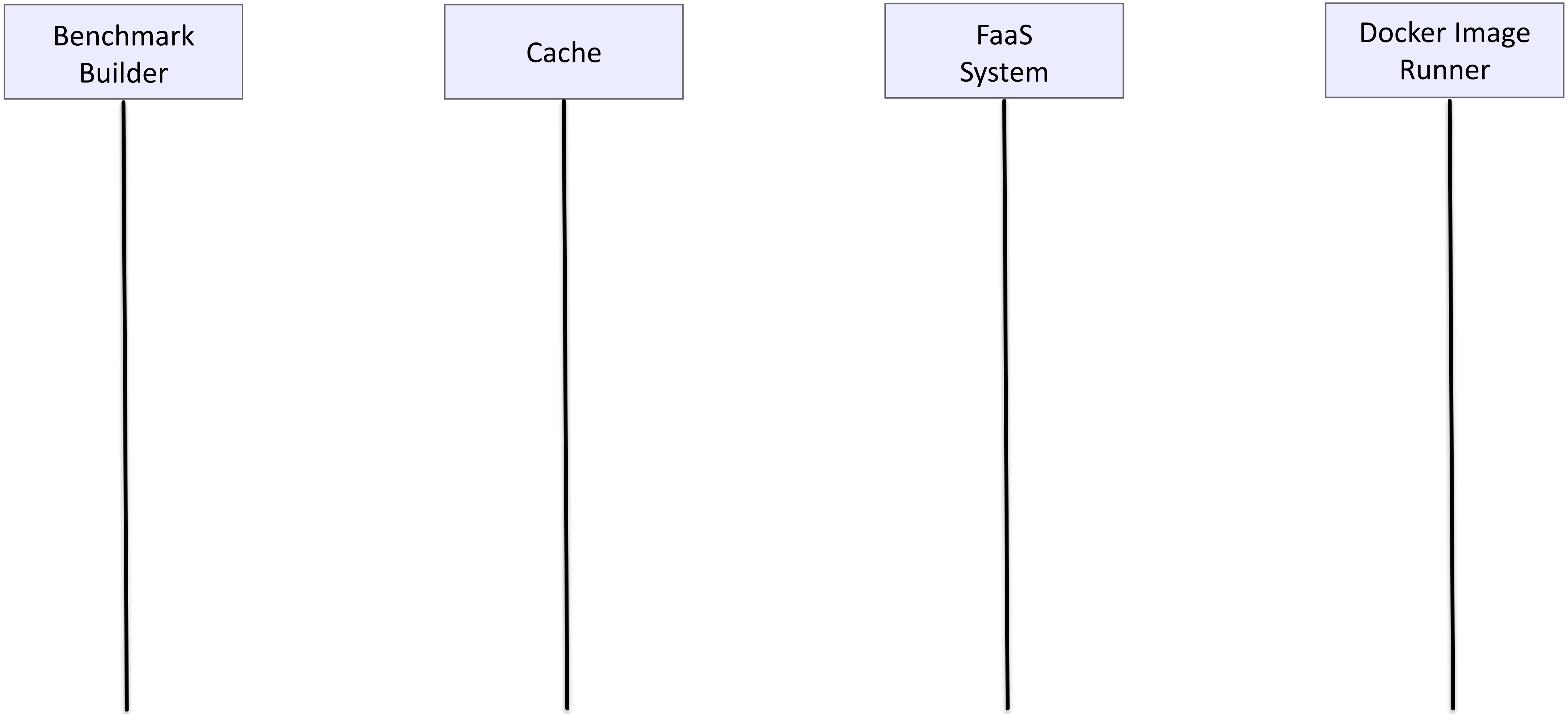
Type	Name	Language
Webapps	dynamic-html, uploader, crud-api	Python, Node.js
Multimedia	thumbnailer	Python, Node.js, C++
Utilities	compression	Python, Node.js*
Inference	image-recognition	Python, C++
Scientific	graph-bfs/pagerank/mst, dna-visualizer	Python, C++
Workflows	trip-booking, video-analysis, …	Python

Programming Languages

Workloads Classes

Deployment Complexity

SeBS: Deployment Pipeline (Code Package)



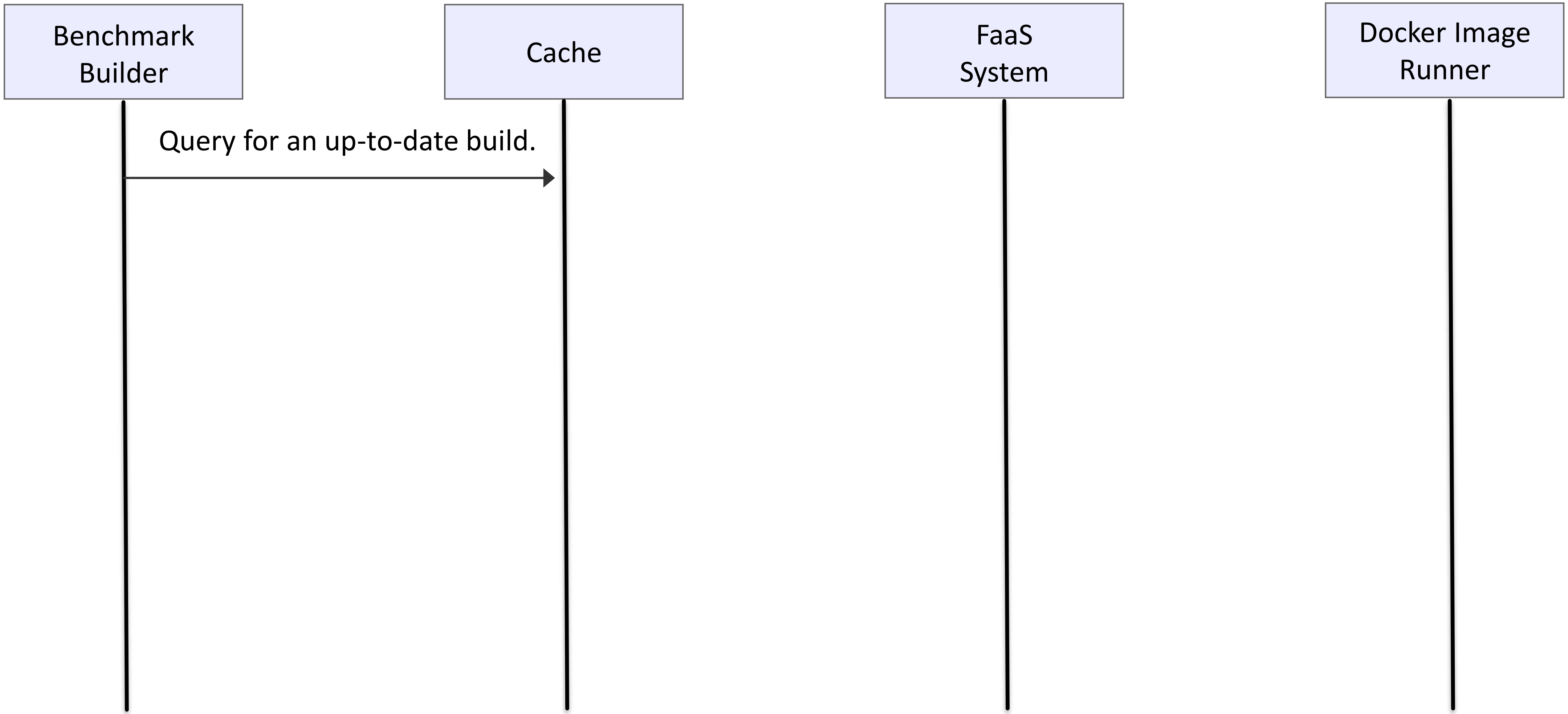
Benchmark Builder

Cache

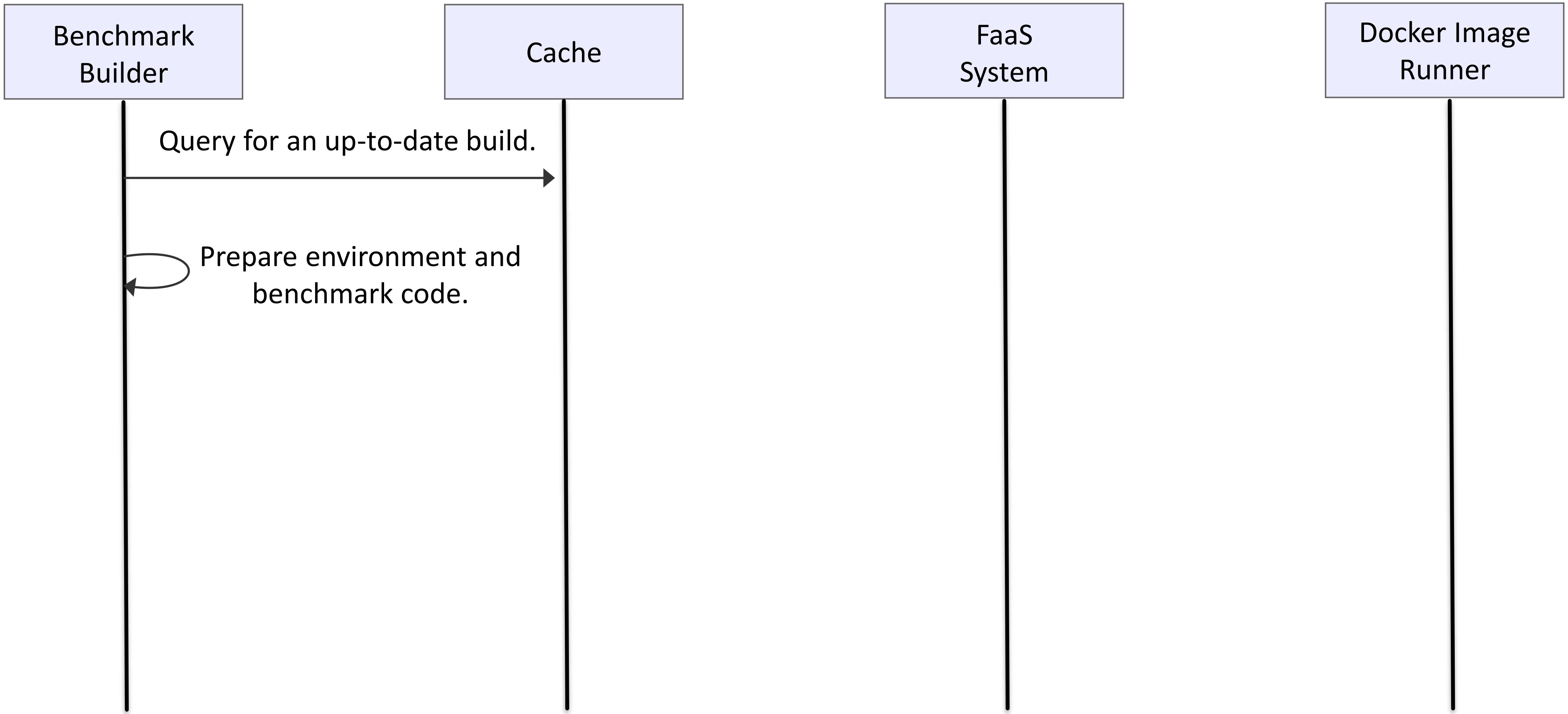
FaaS System

Docker Image Runner

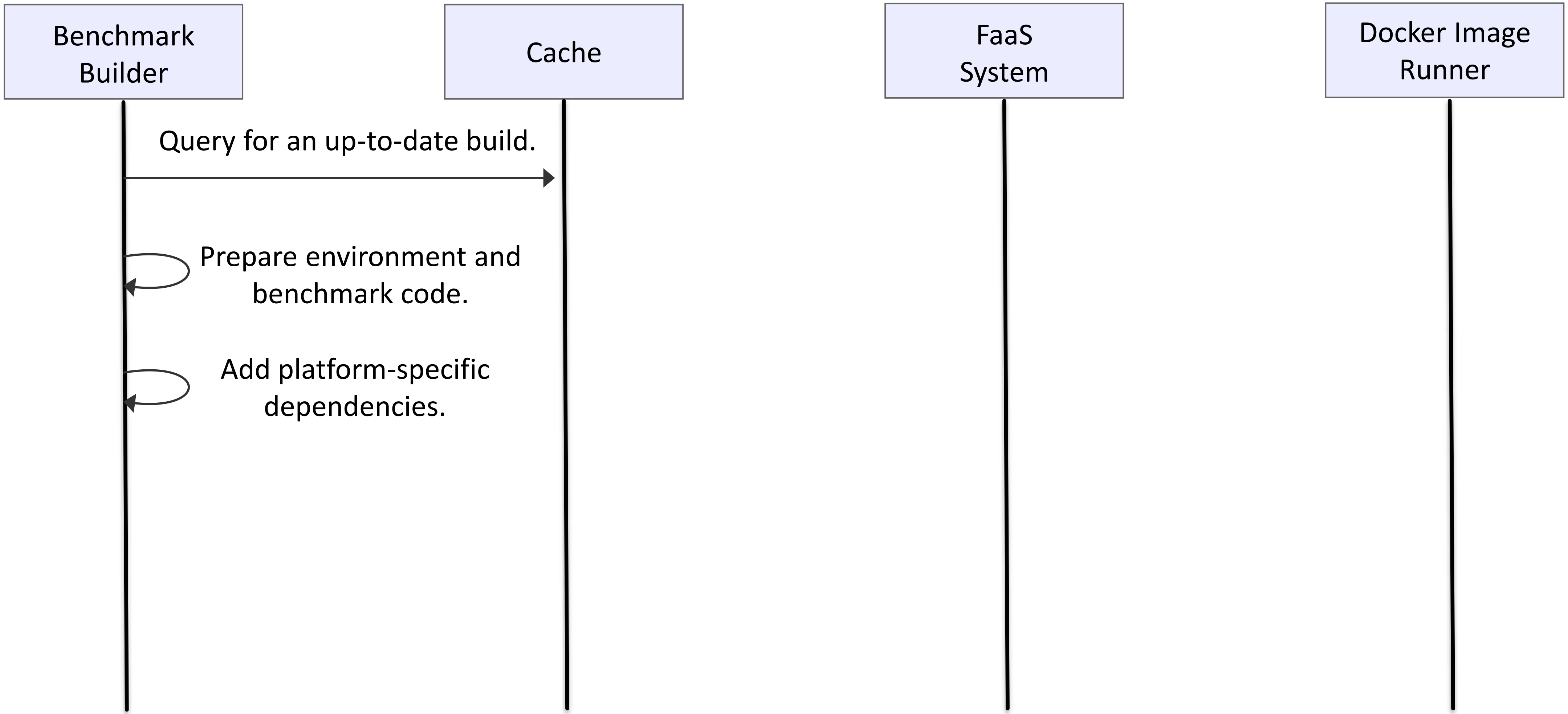
SeBS: Deployment Pipeline (Code Package)



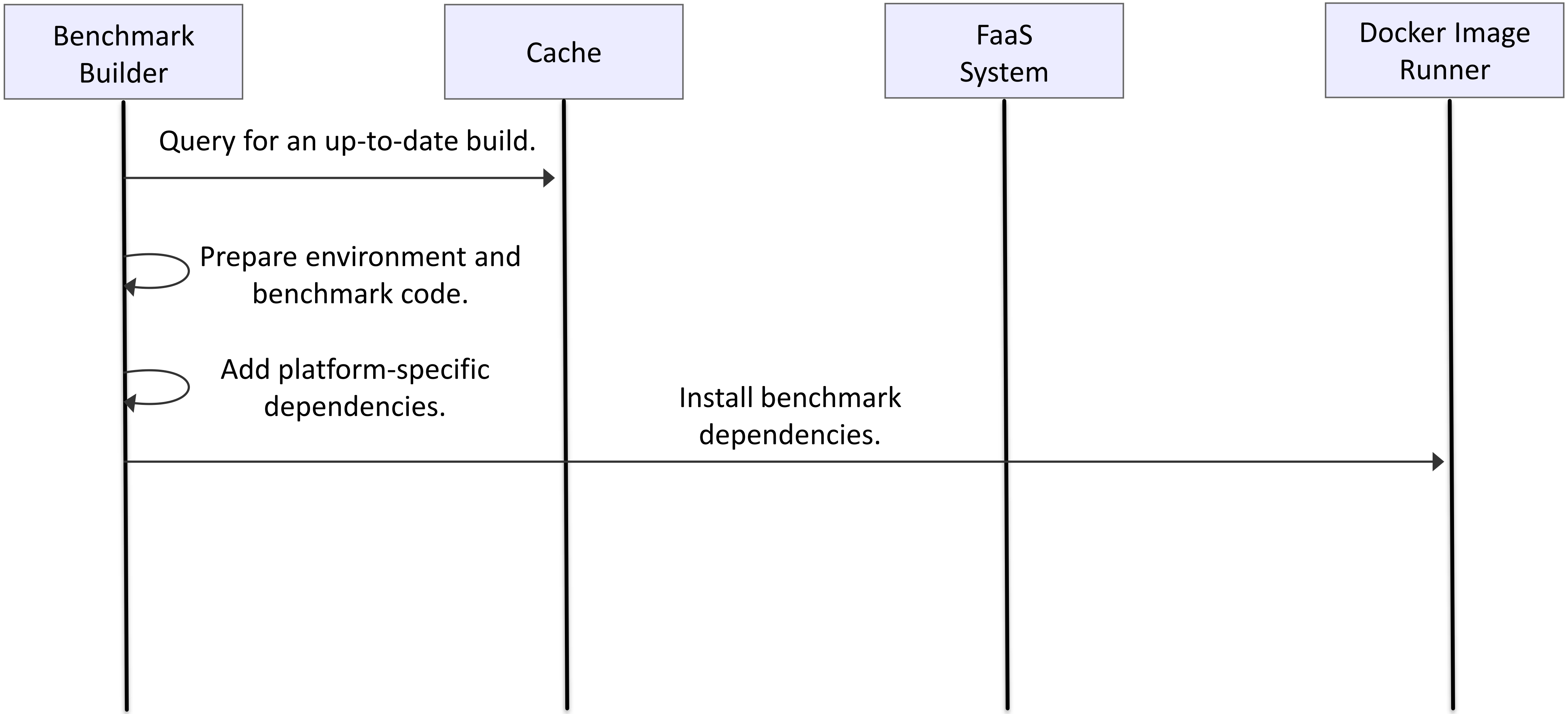
SeBS: Deployment Pipeline (Code Package)


Benchmark Builder
Cache
FaaS System
Docker Image Runner
Query for an up-to-date build.
Prepare environment and benchmark code.

SeBS: Deployment Pipeline (Code Package)


Benchmark Builder
Cache
FaaS System
Docker Image Runner
Query for an up-to-date build.
Prepare environment and benchmark code.
Add platform-specific dependencies.

SeBS: Deployment Pipeline (Code Package)



Benchmark Builder

Cache

FaaS System

Docker Image Runner

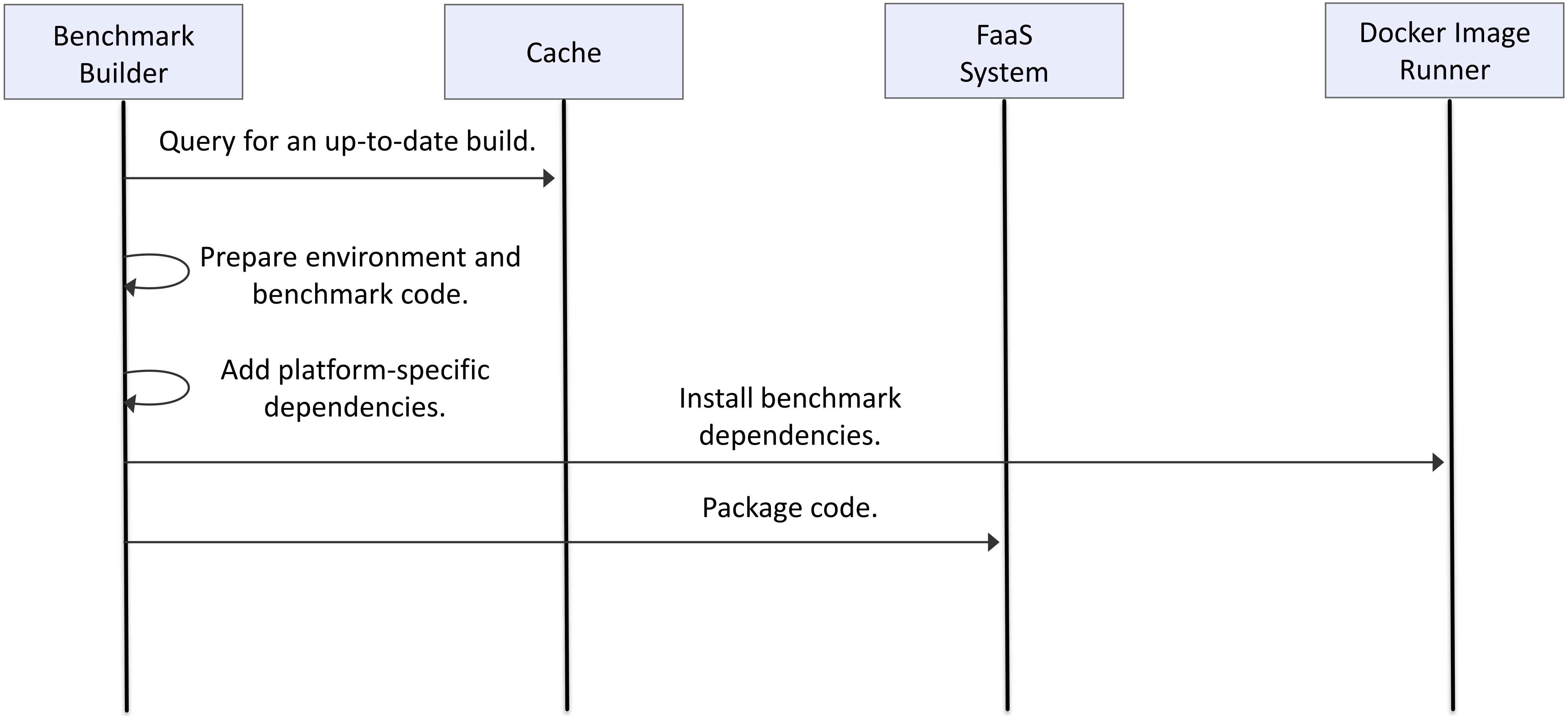
Query for an up-to-date build.

Prepare environment and benchmark code.

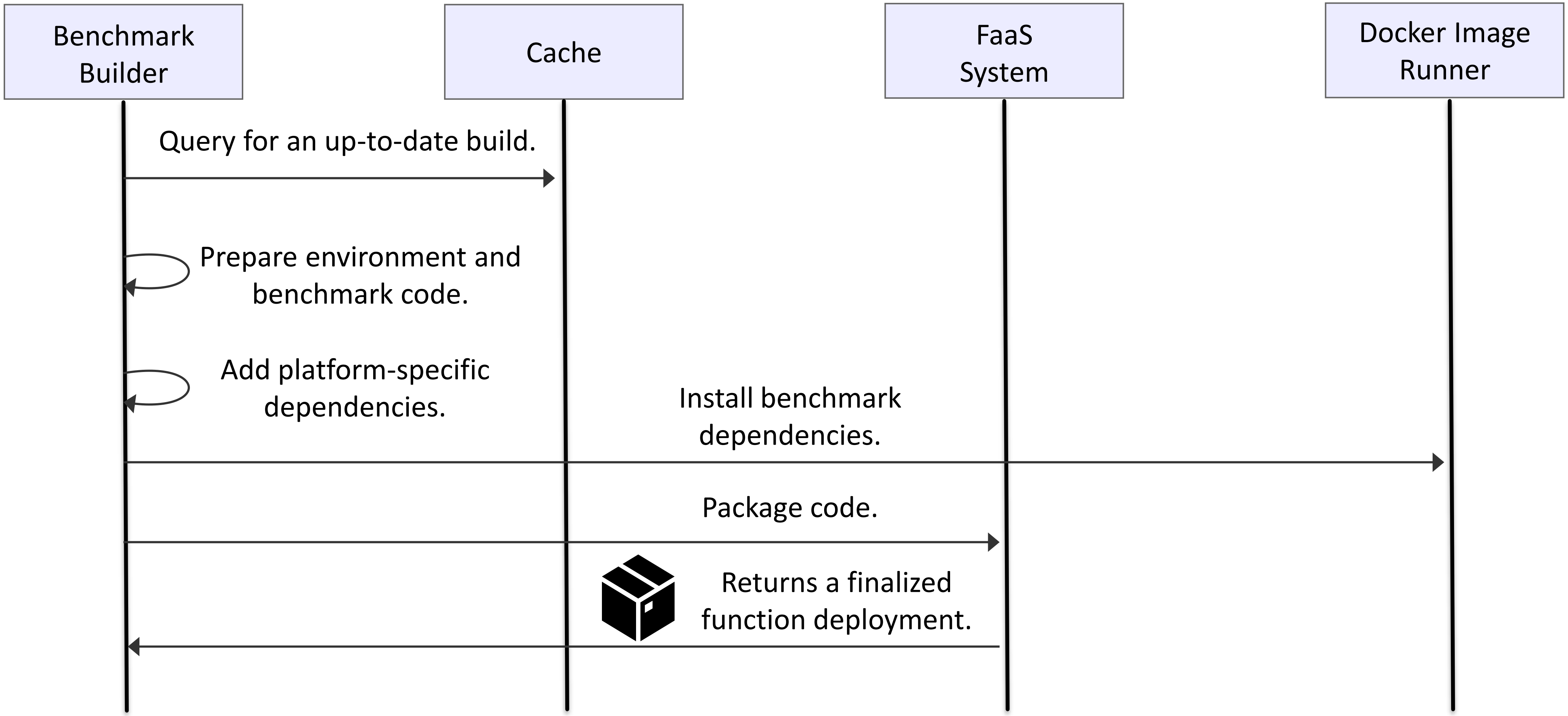
Add platform-specific dependencies.

Install benchmark dependencies.

SeBS: Deployment Pipeline (Code Package)


Benchmark Builder
Cache
FaaS System
Docker Image Runner
Query for an up-to-date build.
Prepare environment and benchmark code.
Add platform-specific dependencies.
Install benchmark dependencies.
Package code.

SeBS: Deployment Pipeline (Code Package)



Benchmark Builder

Cache

FaaS System

Docker Image Runner

Query for an up-to-date build.

Prepare environment and benchmark code.

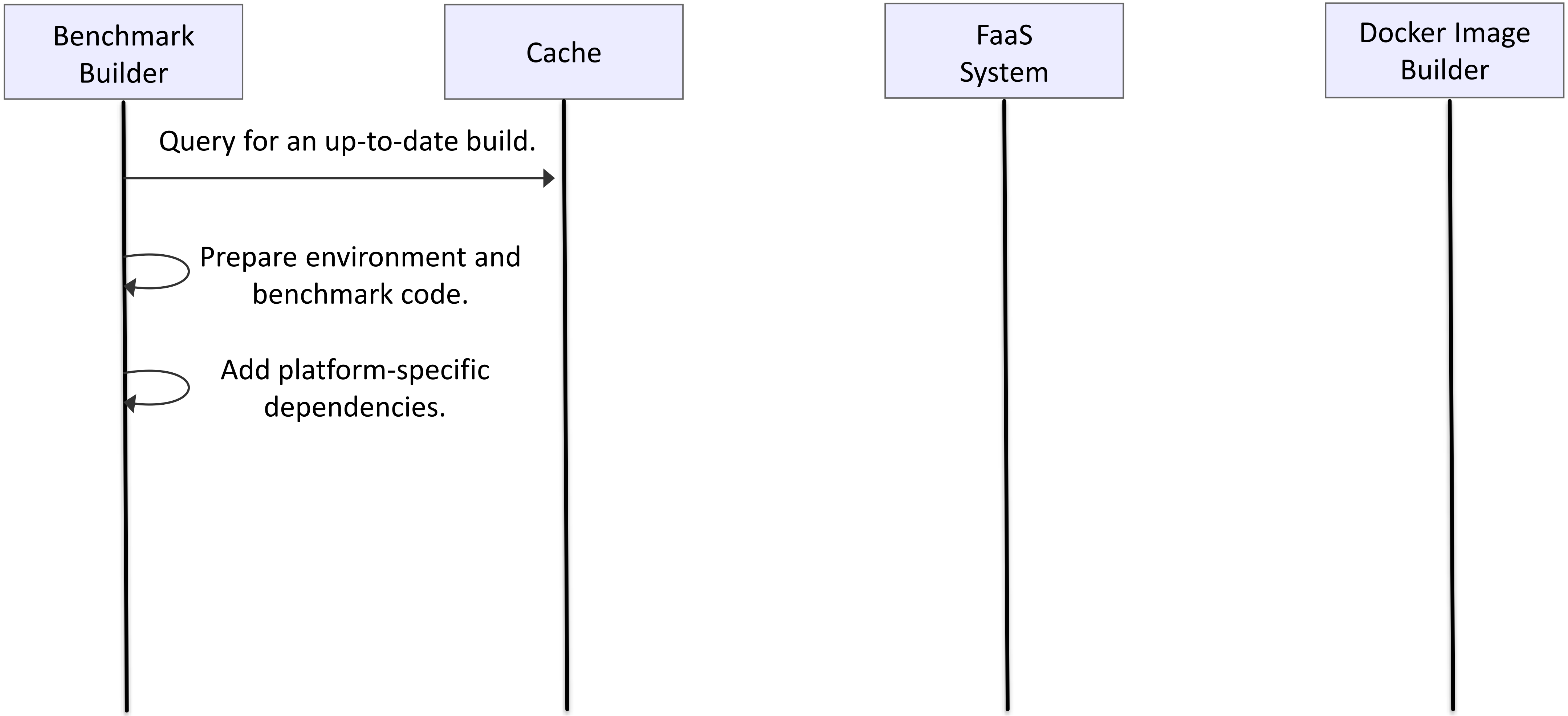
Add platform-specific dependencies.

Install benchmark dependencies.

Package code.

Returns a finalized function deployment.

SeBS: Deployment Pipeline (Container)



Benchmark Builder

Cache

FaaS System

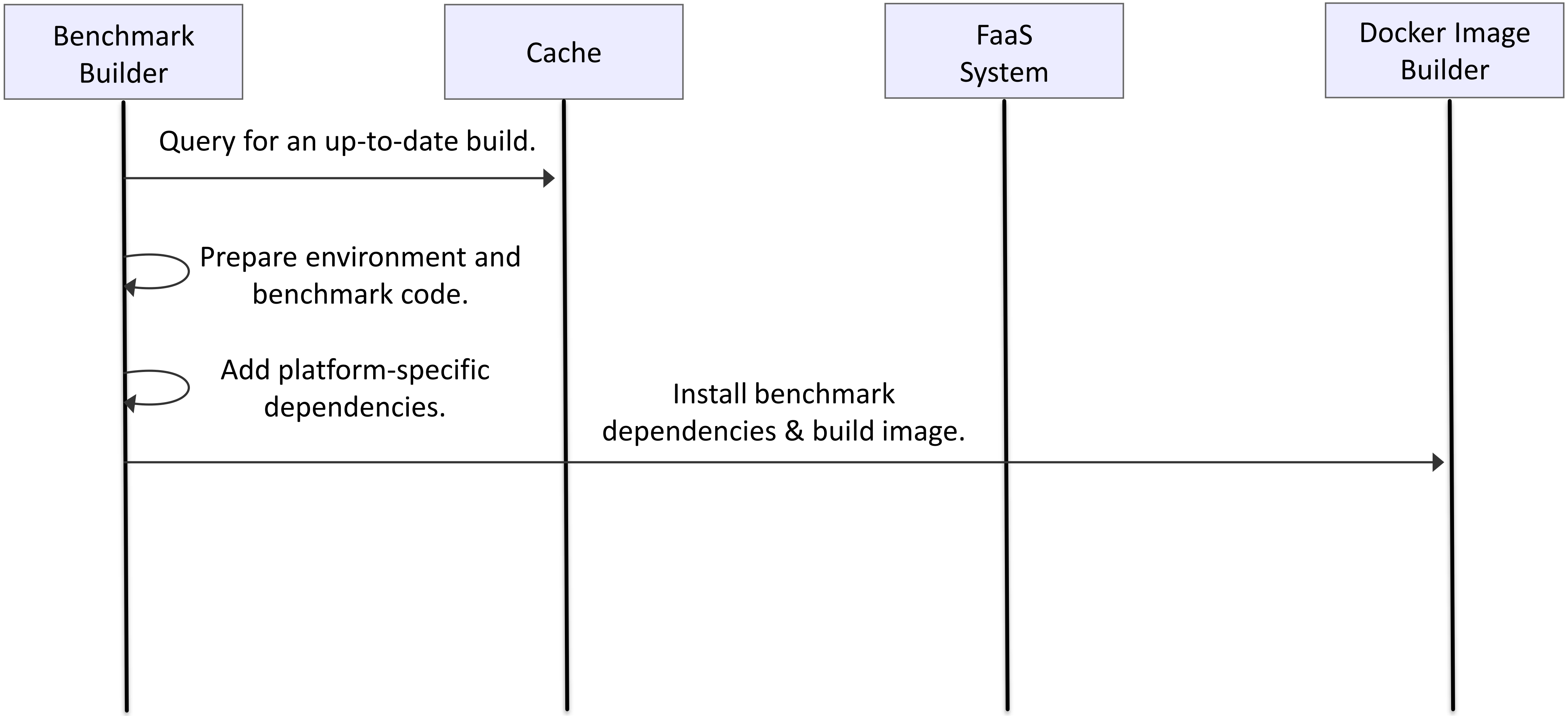
Docker Image Builder

Query for an up-to-date build.

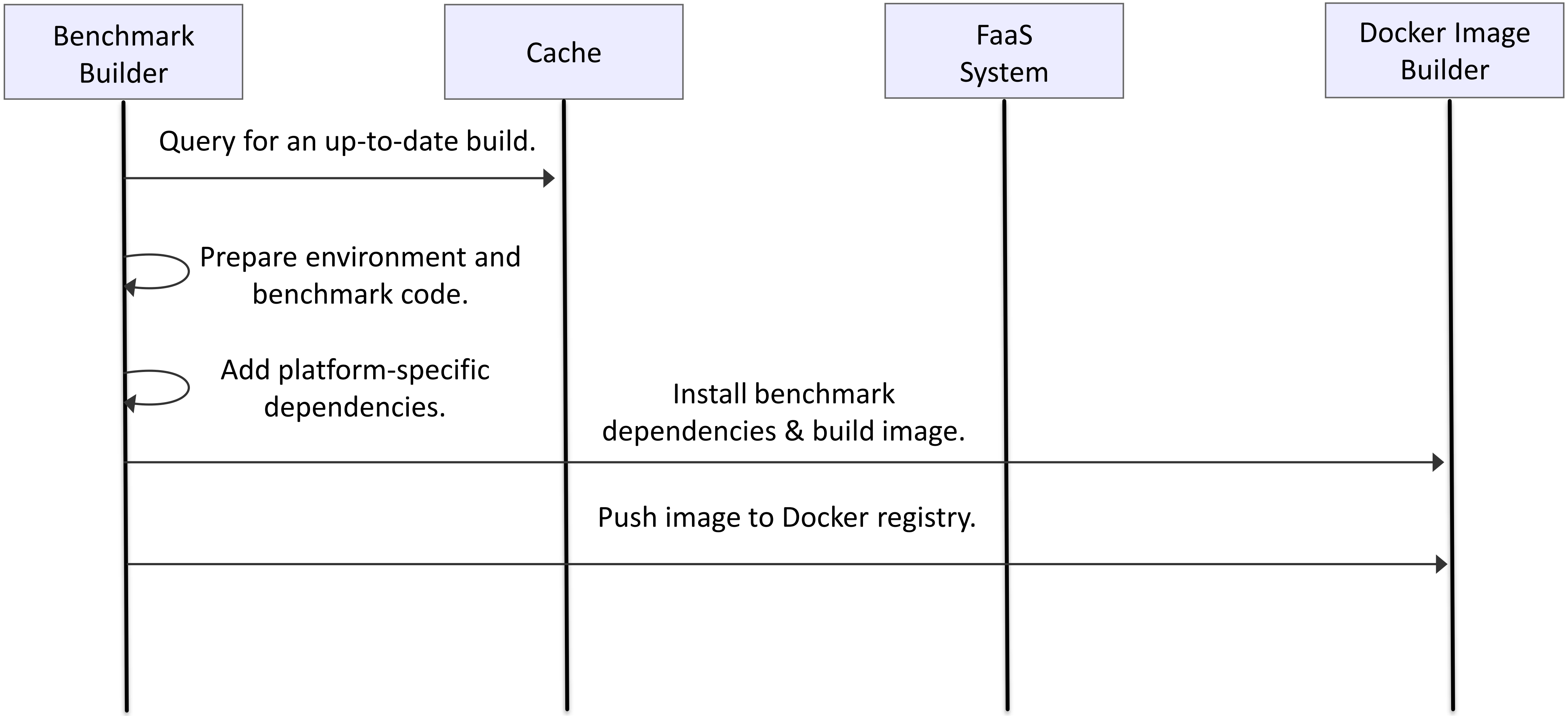
Prepare environment and benchmark code.

Add platform-specific dependencies.

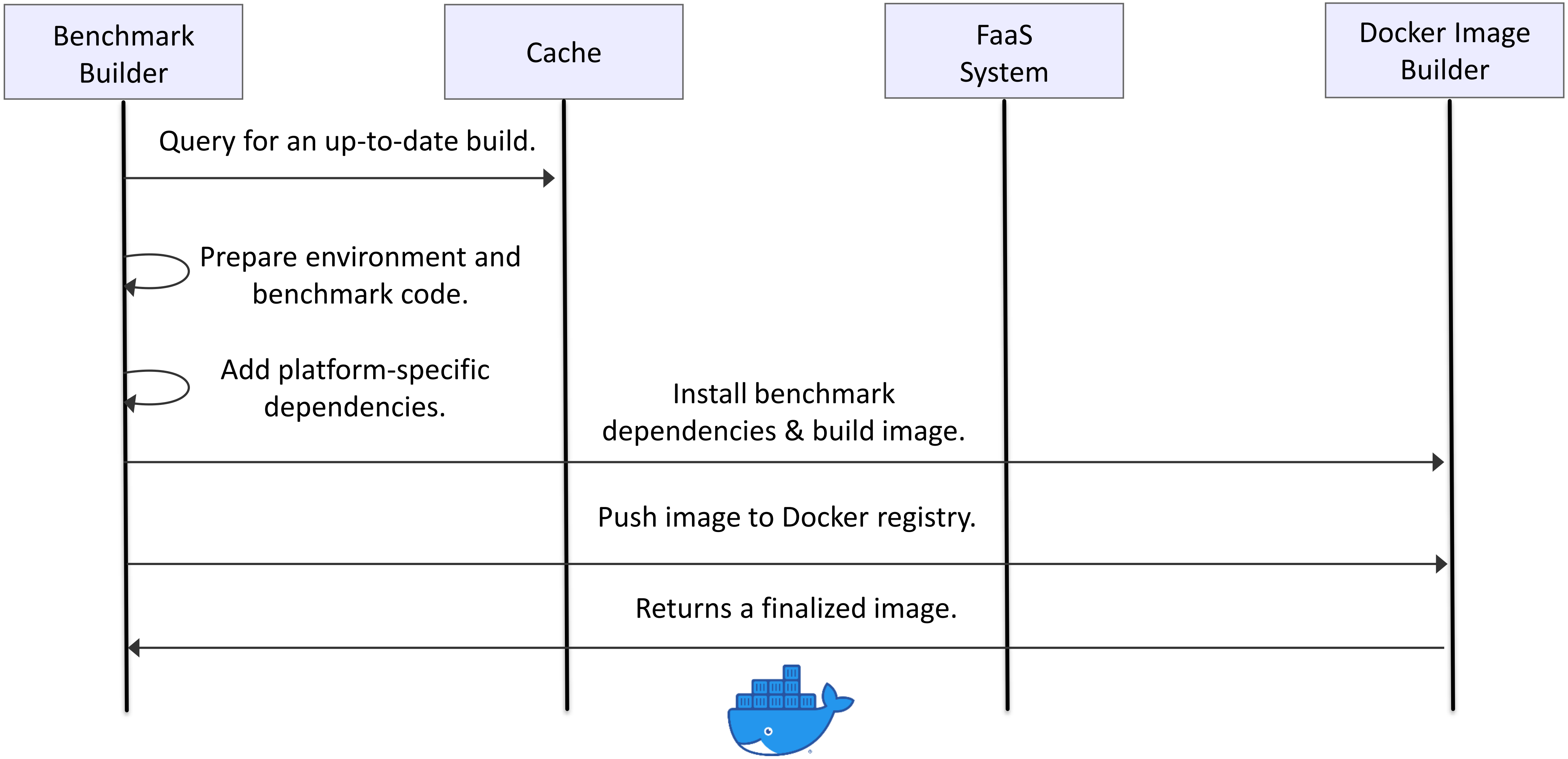
SeBS: Deployment Pipeline (Container)


Benchmark Builder
Cache
FaaS System
Docker Image Builder
Query for an up-to-date build.
Prepare environment and benchmark code.
Add platform-specific dependencies.
Install benchmark dependencies & build image.

SeBS: Deployment Pipeline (Container)


Benchmark Builder
Cache
FaaS System
Docker Image Builder
Query for an up-to-date build.
Prepare environment and benchmark code.
Add platform-specific dependencies.
Install benchmark dependencies & build image.
Push image to Docker registry.

SeBS: Deployment Pipeline (Container)


Benchmark Builder
Cache
FaaS System
Docker Image Builder
Query for an up-to-date build.
Prepare environment and benchmark code.
Add platform-specific dependencies.
Install benchmark dependencies & build image.
Push image to Docker registry.
Returns a finalized image.

SeBS: Resource Management

Function

Cloud System

SeBS: Resource Management

Function

Cloud System

- ❖ Timeout.
- ❖ Memory allocation.
- ❖ Dependencies.

SeBS: Resource Management

Function

- Timeout.
- Memory allocation.
- Dependencies.

Cloud System

- Functions.
- Containers.
- Container registries (AWS).
- Storage buckets.
- NoSQL tables.
- Storage accounts (Azure).
- Role management.

SeBS: Resource Management

Function

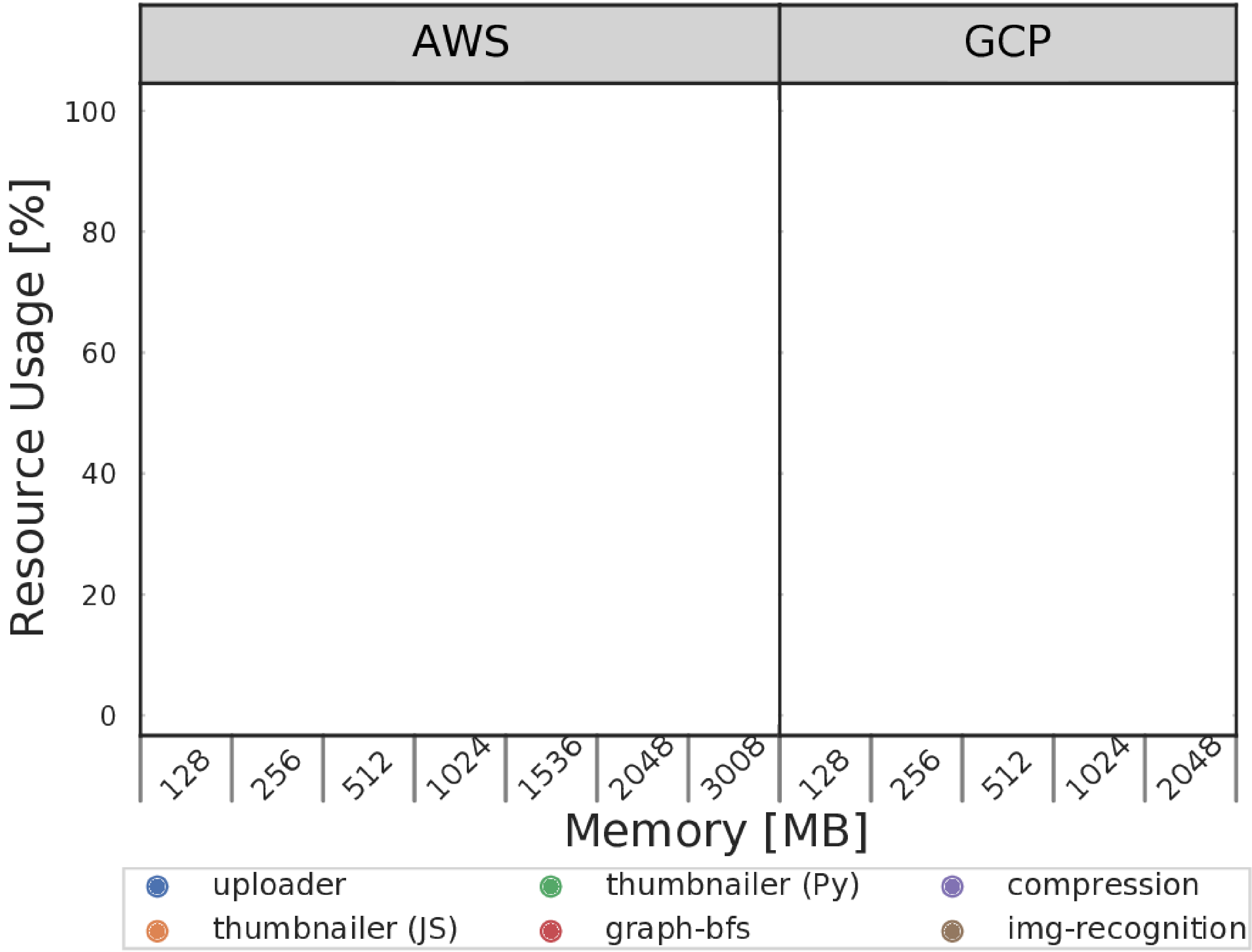
- ❖ Timeout.
- ❖ Memory allocation.
- ❖ Dependencies.

Cloud System

- ❖ Functions.
- ❖ Containers.
- ❖ Container registries (AWS).
- ❖ Storage buckets.
- ❖ NoSQL tables.
- ❖ Storage accounts (Azure).
- ❖ Role management.

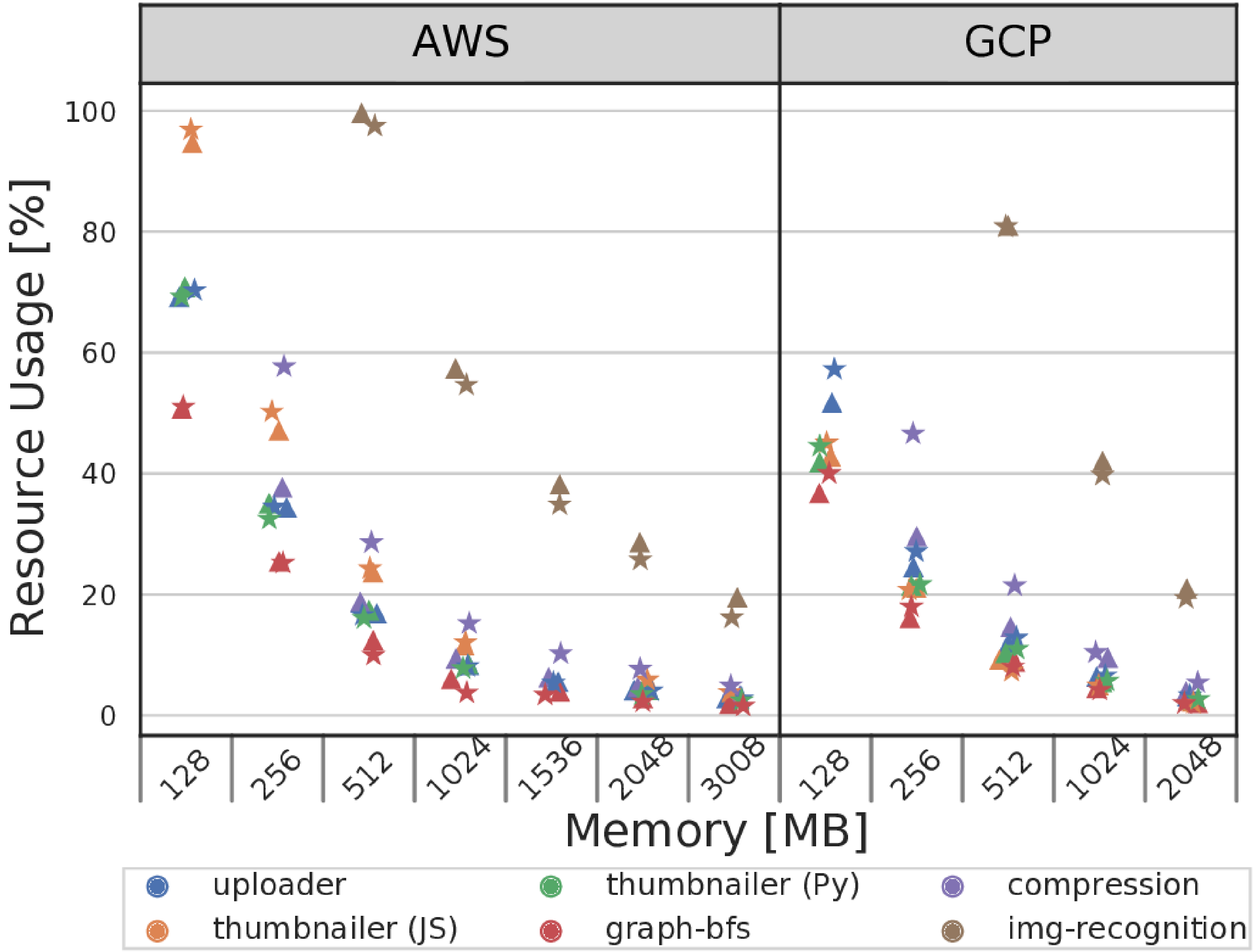
All resources are tagged with SeBS resource ID for easy lookup and cleanup!

Cost Analysis: Resource Usage



"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

Cost Analysis: Resource Usage



"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

Cost Analysis: FaaS vs IaaS

		Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS							
FaaS							

“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

Cost Analysis: FaaS vs IaaS

		Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS							
FaaS							

Configuration:

- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM, 100% utilization, $0.0116/h.
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.

“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

Cost Analysis: FaaS vs IaaS

		Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS	Cloud Storage [req/h]						
FaaS							

Configuration:

- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM, 100% utilization, $0.0116/h.
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.

"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

Cost Analysis: FaaS vs IaaS

		Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS	Cloud Storage [req/h]	11371	27503	18819	1284	15312	117153
FaaS							

Configuration:

- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM, 100% utilization, $0.0116/h.
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.

“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

Cost Analysis: FaaS vs IaaS

		Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS	Cloud Storage [req/h]	11371	27503	18819	1284	15312	117153
FaaS	Eco 1M [$]						
	Eco Break-Even						
	Perf 1M [$]						
	Perf Break-Even						

Configuration:

- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM, 100% utilization, $0.0116/h.
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.

“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

Cost Analysis: FaaS vs IaaS

		Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS	Cloud Storage [req/h]	11371	27503	18819	1284	15312	117153
FaaS	Eco 1M [$]	3.54	2.29	3.75	32.1	15.8	2.08
	Eco Break-Even	3275	5062	3093	362	733	5568
	Perf 1M [$]	6.67	3.34	10	50	19.58	2.5
	Perf Break-Even	1740	3480	1160	232	592	4640

Configuration:

- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM, 100% utilization, $0.0116/h.
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.

“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

Cost Analysis: FaaS vs IaaS

		Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS	Cloud Storage [req/h]	11371	27503	18819	1284	15312	117153
FaaS	Eco 1M [$]	3.54	2.29	3.75	32.1	15.8	2.08
	Eco Break-Even	3275	5062	3093	362	733	5568
	Perf 1M [$]	6.67	3.34	10	50	19.58	2.5
	Perf Break-Even	1740	3480	1160	232	592	4640

Configuration:

- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM, 100% utilization, $0.0116/h.
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.

“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

Agenda

SeBS Slack

- ✓ Part I
  - ✓ What is Serverless?
  - ✓ Benchmarking Suite SeBS
  - ✓ Working with SeBS
- ✓ **Hands-on I: Local Deployment & Storage**
- ✓ Part II
  - ✓ Communication and Data
  - ✓ Serverless Workflows
  - ✓ Experiments in SeBS
- ✓ Hands-on II: FaaS Platforms & Experiments
- ✓ Part III
  - ✓ Research Directions in Serverless
  - ✓ Development of SeBS

slack **Join SeBS Slack channel: https://serverlessbenchmark.slack.com**

SeBS in Practice

SeBS Slack

spcl/sebs-tutorial

SeBS in Practice

SeBS Slack

spcl/sebs-tutorial

```
git clone https://github.com/spcl/sebs-tutorial.git
```

SeBS in Practice

spcl/sebs-tutorial

SeBS Slack

SeBS in Practice

spcl/sebs-tutorial

SeBS Slack

What do we need?

- Docker running on the system
- Python 3.7+
- libcurl
- Virtualenv
- Works on Linux. WSL should also work ☺

SeBS in Practice

spcl/sebs-tutorial

SeBS Slack

What do we need?

- Docker running on the system
- Python 3.7+
- libcurl
- Virtualenv
- Works on Linux. WSL should also work ☺

Where to execute functions?

- Cloud? You need credentials and set up account.
- Open source? Deploy OpenWhisk – instructions are provided. You need DockerHub account.
- Local test environment?

SeBS in Practice

spcl/sebs-tutorial

SeBS Slack

What do we need?

- Docker running on the system
- Python 3.7+
- libcurl
- Virtualenv
- Works on Linux. WSL should also work ☺

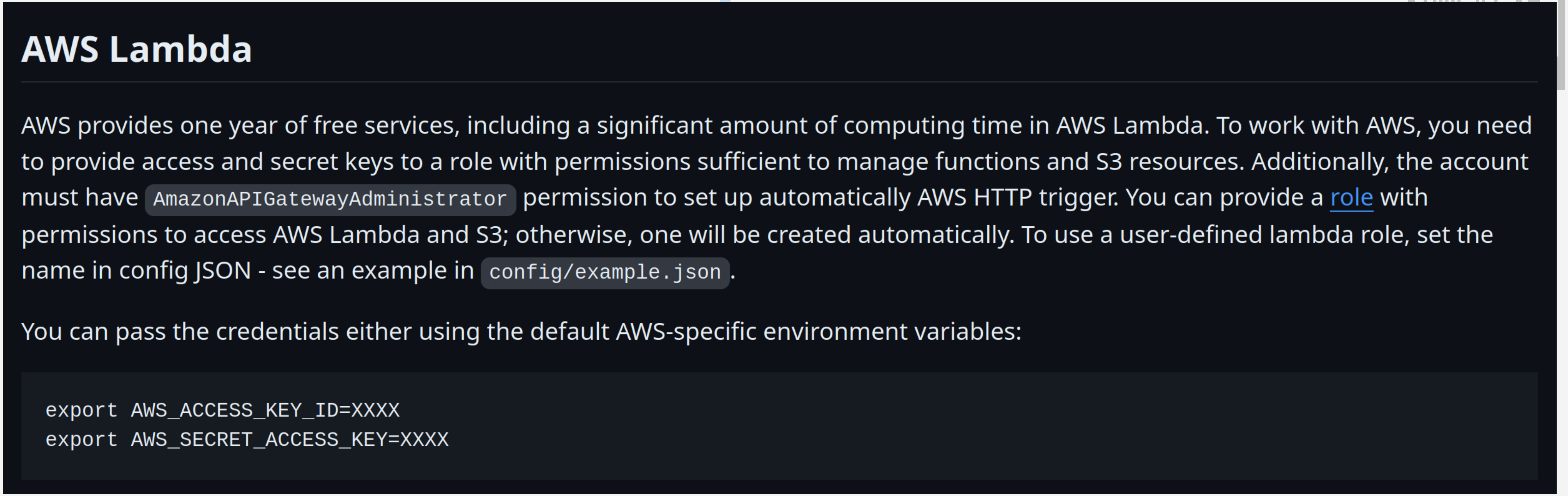
Where to execute functions?

- Cloud? You need credentials and set up account.
- Open source? Deploy OpenWhisk – instructions are provided. You need DockerHub account.
- Local test environment?

SeBS in Practice

spcl/sebs-tutorial

What do we need?



AWS Lambda

AWS provides one year of free services, including a significant amount of computing time in AWS Lambda. To work with AWS, you need to provide access and secret keys to a role with permissions sufficient to manage functions and S3 resources. Additionally, the account must have `AmazonAPIGatewayAdministrator` permission to set up automatically AWS HTTP trigger. You can provide a role with permissions to access AWS Lambda and S3; otherwise, one will be created automatically. To use a user-defined lambda role, set the name in config JSON - see an example in `config/example.json`.

You can pass the credentials either using the default AWS-specific environment variables:

```
export AWS_ACCESS_KEY_ID=XXXX
export AWS_SECRET_ACCESS_KEY=XXXX
```

- Local test environment?

SeBS in Practice

spcl/sebs-tutorial

```
{
  "experiments": {
    "deployment": "aws",
    "update_code": false,
    "update_storage": false,
    "download_results": false,
    "architecture": “x64",
    "container_deployment": true,
    "runtime": {
      "language": "python",
      "version": "3.9"
    },
  },
}
```

SeBS Slack

SeBS in Practice

```
{
  "experiments": {
    "deployment": "aws",
    "update_code": false,
    "update_storage": false,
    "download_results": false,
    "architecture": “x64",
    "container_deployment": true,
    "runtime": {
      "language": "python",
      "version": "3.9"
    },
  },
}
```

```
"deployment": {
    "name": "aws",
    "aws": {
      "region": "us-east-1",
      "lambda-role": ""
    },
    "azure": {
      "region": "westeurope"
    },
    "gcp": {
      "region": "europe-west1",
      "project_name": "",
      "credentials": ""
    },
    "local": {
      "storage": {}
    },
    "openwhisk": {
      "storage": {},
      "wskBypassSecurity": "true",
      "wskExec": "wsk",
      ...
    }
}
```

spcl/sebs-tutorial

SeBS Slack

SeBS in Practice

spcl/sebs-tutorial

SeBS Slack

```
{
  "experiments": {
    "deployment": "aws",
    "update_code": false,
    "update_storage": false,
    "download_results": false,
    "architecture": “x64",
    "container_deployment": true,
    "runtime": {
      "language": "python",
      "version": "3.9"
    },
  },
}
```

JSON configuration can be overridden via CLI options.

```
"deployment": {
    "name": "aws",
    "aws": {
      "region": "us-east-1",
      "lambda-role": ""
    },
    "azure": {
      "region": "westeurope"
    },
    "gcp": {
      "region": "europe-west1",
      "project_name": "",
      "credentials": ""
    },
    "local": {
      "storage": {}
    },
    "openwhisk": {
      "storage": {},
      "wskBypassSecurity": "true",
      "wskExec": "wsk",
      ...
    }
}
```

SeBS in Practice

SeBS Slack

spcl/sebs-tutorial

git clone https://github.com/spcl/sebs-tutorial.git

slack Join SeBS Slack channel: https://serverlessbenchmark.slack.com