SERVERLESS RUST

@DDPRRT - RUST-LINZ.AT - FETTBLOG.EU - RUST-TRAINING.EU

HI 👋

I’M STEFAN
@DDPRRT
CO-FOUNDER + CO-ORGANIZER RUST LINZ

TYPESCRIPT IN 50 LESSONS 📖

NOV 2020 WITH SMASHING MAGAZINE
TYPESCRIPT-BOOK.COM
FETTBLOG.EU

LET'S TALK ABOUT SERVERLESS

WHAT IS IT...

AUTOSCALING

Care about servers, less

Scale-out happens automatically

No infrastructure management

Consumption based billing

Examples

Google Cloud Run

AWS Fargate

FUNCTIONS AS A SERVICE

Write less servers

Focus on business logic

Stateless development mindset

Glue logic between services

Examples

AWS Lambda

Azure Functions

AUTOSCALING

Care about servers, less

Scale-out happens automatically

No infrastructure management

Consumption based billing

Examples

Google Cloud Run

AWS Fargate

FUNCTIONS AS A SERVICE

Write less servers

Focus on business logic

Stateless development mindset

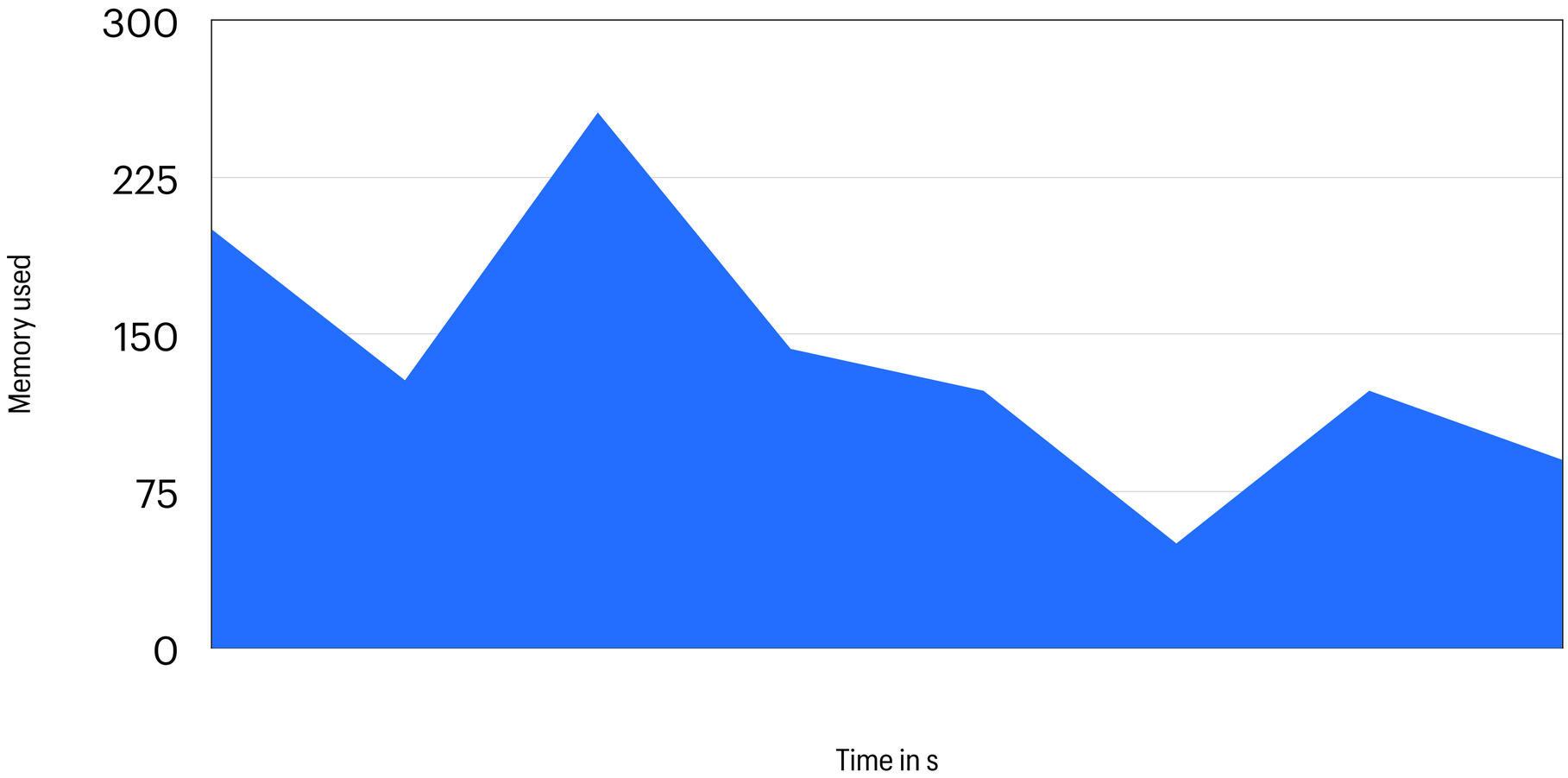
e logic between services

Examples

AWS Lambda

Azure Functions

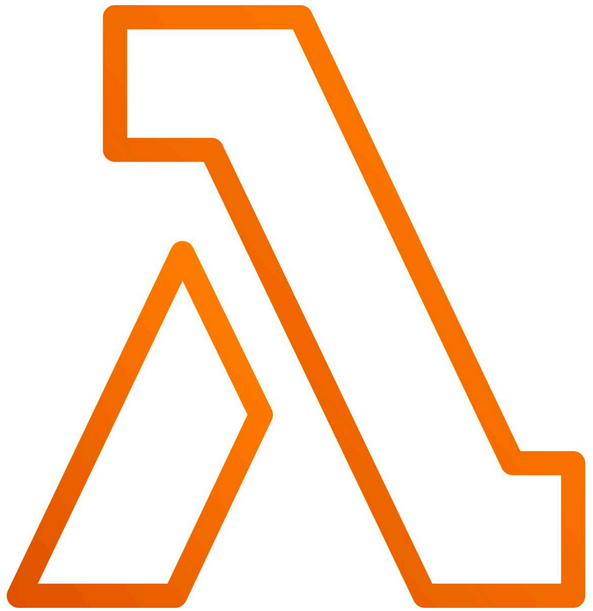
BILLING



WHY DO WE WANT TO USE RUST WITH SERVERLESS?

RUST IS REALLY GOOD AT MEMORY AND SPEED

AWS LAMBDA



AWS LAMBDA

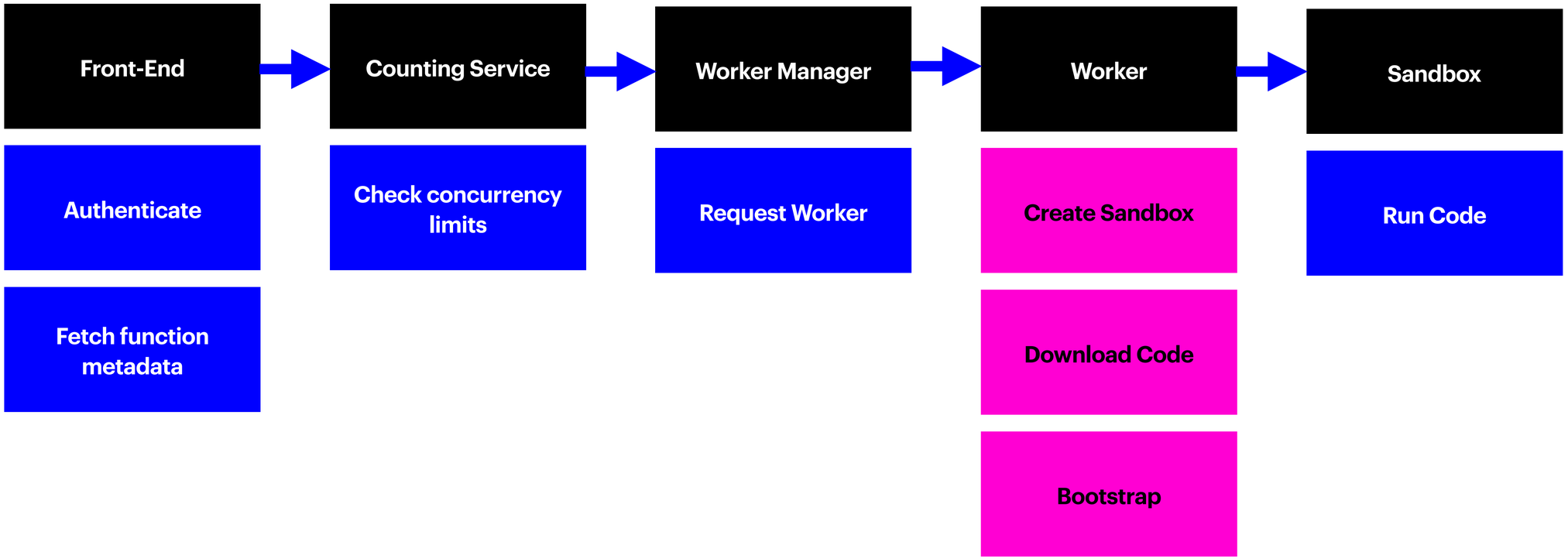
Glue code between AWS Services

Unaware of Triggers, just takes event

Runs in lightweight Micro VMs (Firecracker)

Workers, not servers

LAMBDA EXECUTION LIFECYCLE



HTTPS://WWW.DYNATRACE.COM/NEWS/BLOG/A-LOOK-BEHIND-THE-SCENES-OF-AWS-LAMBDA-AND-OUR-NEW-LAMBDA-MONITORING-EXTENSION/

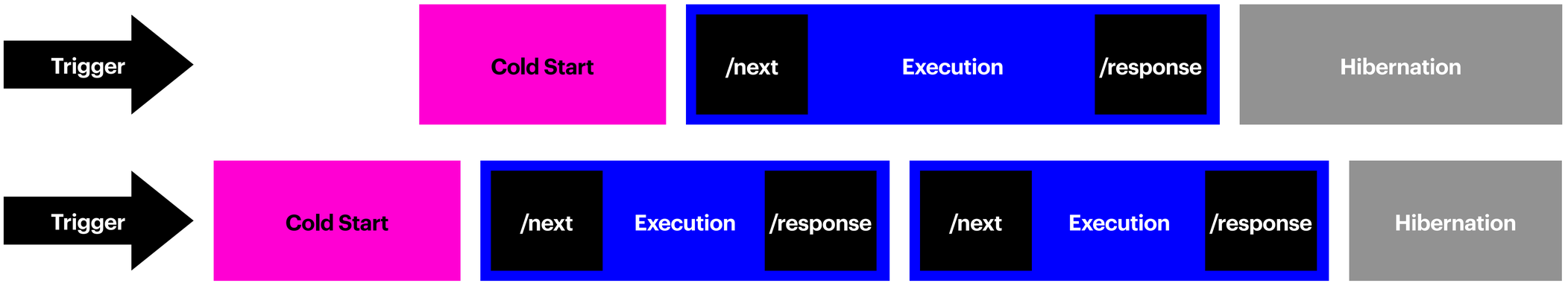
Always

Coldstart

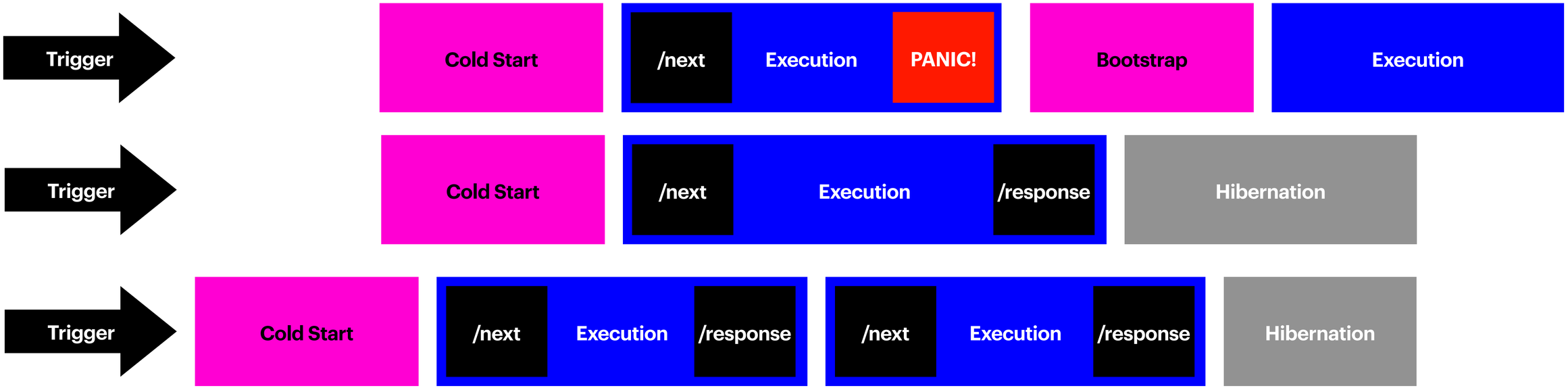
INVOCATION


Trigger
Cold Start
/next
Execution
/response
/next
Execution
/response
Hibernation

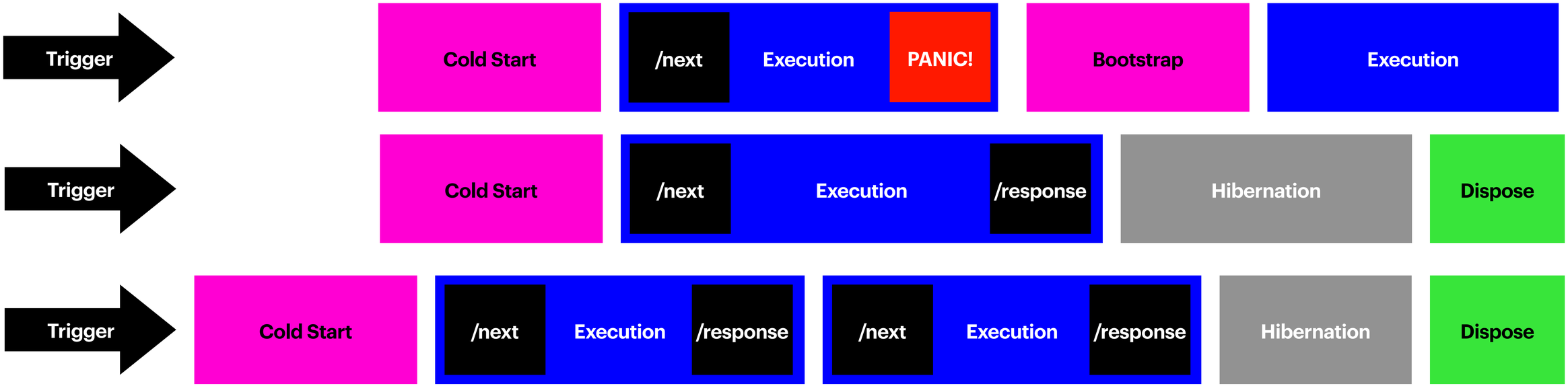
INVOCATION


Trigger
Cold Start
/next
Execution
/response
Hibernation
Trigger
Cold Start
/next
Execution
/response
/next
Execution
/response
Hibernation

INVOCATION


Trigger
Cold Start
/next
Execution
PANIC!
Bootstrap
Execution
Trigger
Cold Start
/next
Execution
/response
Hibernation
Trigger
Cold Start
/next
Execution
/response
/next
Execution
/response
Hibernation

INVOCATION


Trigger
Cold Start
/next
Execution
PANIC!
Bootstrap
Execution
Trigger
Cold Start
/next
Execution
/response
Hibernation
Dispose
Trigger
Cold Start
/next
Execution
/response
/next
Execution
/response
Hibernation
Dispose

GOOD TO KNOW

RAM freely scalable from 128MB to 10GB in 1MB steps

Costs scale the same per ms

vCPU does the same

E.g. 2x the RAM = 2x the cost = 2x the vCPU

In A LOT of cases it's also twice as fast!

AWS LAMBDA IN RUST?

THIS IS WHAT YOU READ IN BLOG POSTS 😭

AWS LAMBDA IN RUST?

NO!!!

THIS IS WHAT YOU READ IN BLOG POSTS 😭

AWS LAMBDA RUNTIME

CRATES.IO/CRATES/LAMBDA_RUNTIME

```rust
use lambda_runtime::{Context, handler_fn};
use serde_json::{Value, json};

#[tokio::main]
async fn main() -> Result<(), lambda_runtime::Error>{
    let func = handler_fn(handler);
    lambda_runtime::run(func).await?;
    Ok(())
}

async fn handler(event: Value, _: Context) -> Result<Value, lambda_runtime::Error> {
    //...
}
```

COMPILE TO X86_64-UNKNOWN-LINUX-GNU
BINARY NAME: BOOTSTRAP

RESULTS - LAMBDA 128MB

	Hello World	Palindrome Products 10-99	Palindrome Products 100-999	Palindrome Products 1000-9999
Node 🥶	~200ms			
Node 🔥	2ms	2ms	~500ms	~70s
Rust 🥶	< 20ms			
Rust 🔥	< 1ms	< 1ms	~45ms	~8s

BENEFITS OF RUST IN LAMBDA

Very small binaries instead of Language Runtime + Code + Dependencies

Works great on low vCPU! (128 MB RAM = 1/13th vCPU)

Low RAM usage

Less variation in execution

It’s super fun!

AZURE FUNCTIONS

AZURE FUNCTIONS

Triggers are part of the function definition (eg. HTTP)

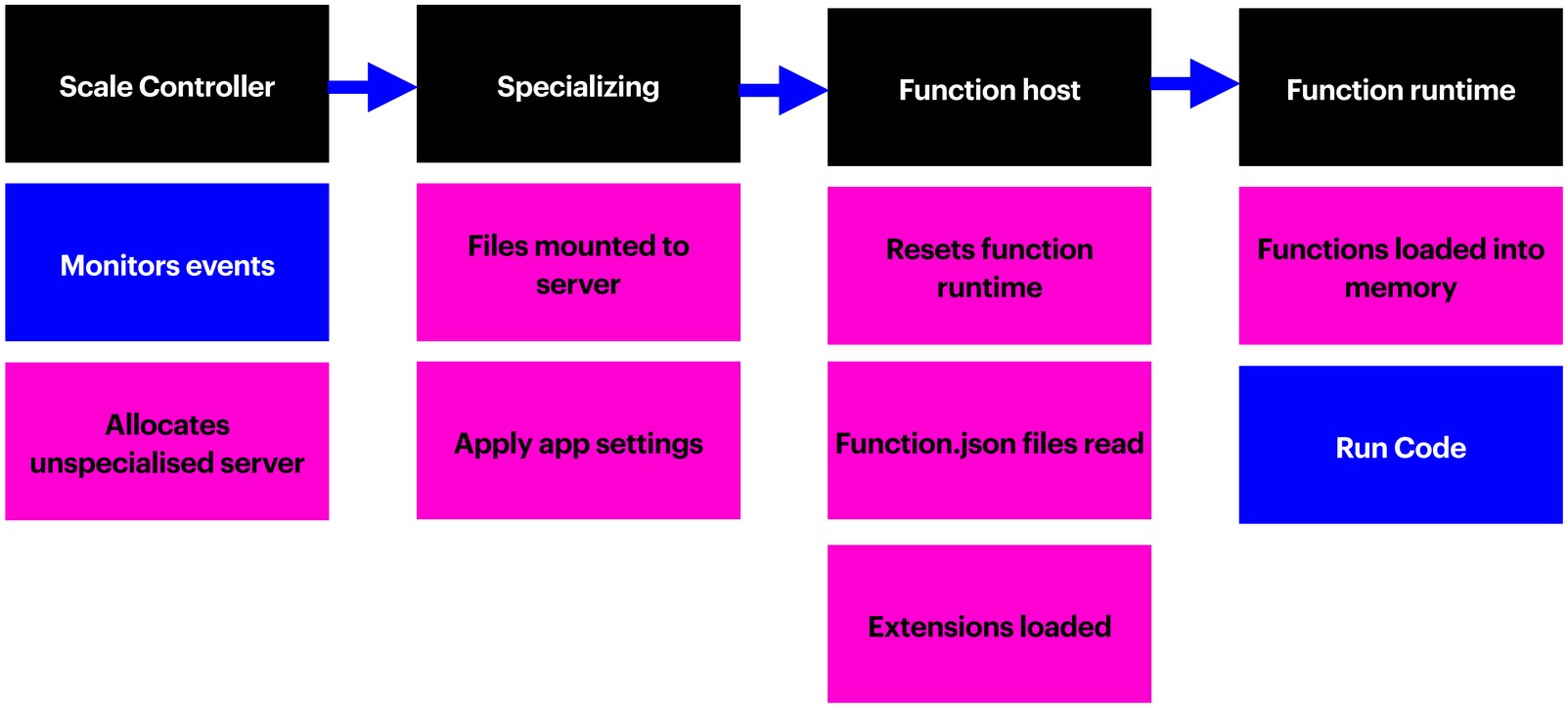
Multiple target bindings

Builds up on Azure WebJobs

Unit of deployment / scale are *function apps*

Needs Azure Storage to store function apps

AF EXECUTION LIFECYCLE

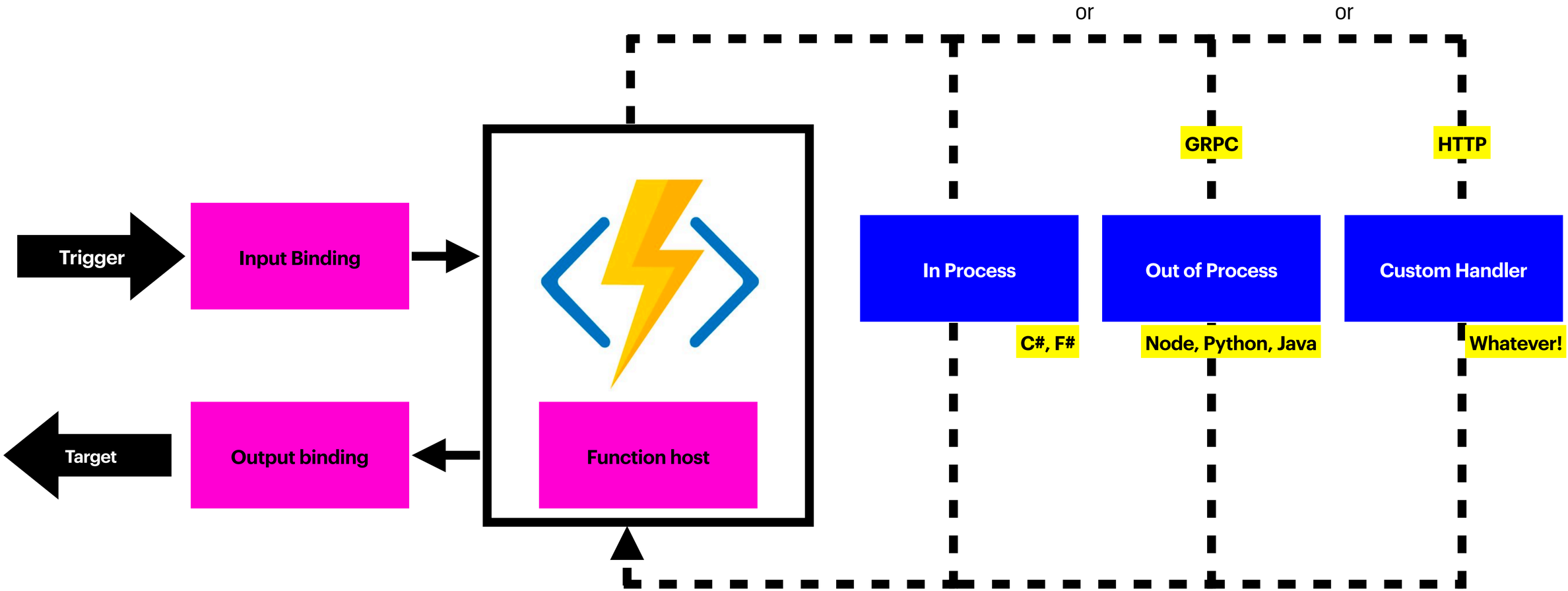


AZURE.MICROSOFT.COM/EN-US/BLOG/UNDERSTANDING-SERVERLESS-COLD-START/

Always

Coldstart

AZURE FUNCTIONS HOST



GOOD TO KNOW

RAM scales to your needs

vCPU scales to your needs

Cold starts are slow, but you get as much CPU as you need

Cold starts are for the entire *function app*

Consumption plan allocates max 100 ACUs per Server (~ 1 vCPU)

DEFINE THE FUNCTION HOST

HOST.JSON

```
{
    "version": "2.0",
    "extensionBundle": {
      "id": "Microsoft.Azure.Functions.ExtensionBundle",
      "version": "[2.*, 3.0.0)"
    },
    "customHandler": {
      "description": {
        "defaultExecutablePath": "handler",
        "workingDirectory": "",
        "arguments": []
      },
      "enableForwardingHttpRequest": true
    }
  }
```

CREATE BINDINGS

PALINDROMES/FUNCTION.JSON

```
{
    "bindings": [
      {
        "authLevel": "anonymous",
        "type": "httpTrigger",
        "direction": "in",
        "name": "req",
        "methods": [
          "get",
          "post"
        ]
      },
      {
        "type": "http",
        "direction": "out",
        "name": "res"
      }
    ]
  }
```

ANY SERVER WILL DO

```rust
#[tokio::main]
async fn main() {
    let server = warp::get()
        .and(warp::path("api"))
        .and(warp::path("palindromes"))
        .and(warp::query::<HashMap<String, String>>())
        .map(|p: HashMap<String, String>| {
            // ...
        });

    let port_key = "FUNCTIONS_CUSTOMHANDLER_PORT";
    let port: u16 = match env::var(port_key) {
        Ok(val) => val.parse().expect("Custom Handler port is not a number"),
        Err(_) => 3000,
    };

    println!("Starting at {}", port);

    warp::serve(server)
        .run((Ipv4Addr::UNSPECIFIED, port))
        .await
}
```

COMPILE TO X86_64-UNKNOWN-LINUX-GNU
BINARY NAME: HANDLER

ANY SERVER WILL DO

```
#[tokio::main]
async fn main() {
    let server = warp::get()
        .and(warp::path("api"))
        .and(warp::path("palindromes"))
        .and(warp::query::<HashMap<String, String>>())
        .map(|p: HashMap<String, String>| {
            // ...
        });

    let port_key = "FUNCTIONS_CUSTOMHANDLER_PORT";
    let port: u16 = match env::var(port_key) {
        Ok(val) => val.parse().expect("Custom Handler port is not a number"),
        Err(_) => 3000,
    };

    println!("Starting at {}", port);

    warp::serve(server)
        .run((Ipv4Addr::UNSPECIFIED, port))
        .await
}
```

COMPILE TO X86_64-UNKNOWN-LINUX-GNU
BINARY NAME: HANDLER

RESULTS - AZURE FUNCTIONS

	Hello World	Palindrome Products 10-99	Palindrome Products 100-999	Palindrome Products 1000-9999
Node 🥶	~700ms			
Node 🔥	1ms	9ms	~80ms	~10s
Rust 🥶	< 100ms			
Rust 🔥	< 1ms	< 5ms	~15ms	~1s

BENEFITS OF RUST IN AF

Significantly lower cold starts

It’s just a server! Take your web server and let it scale automatically!

Easy migration!

Cold starts also depend on the size of the function app. Less of a problem in Rust

Again, it’s super fun!

SUMMARY

Rust should definitely be considered when writing Serverless Functions!

Rust can help significantly with cold start times and duration times

Execution time costs are one thing

- Azure Storage Costs?

- AWS API Gateway Costs?

(Billing is a nightmare...)

MATERIALS

- https://azure.microsoft.com/en-us/blog/understanding-serverless-cold-start/

- https://docs.microsoft.com/en-us/azure/azure-functions/create-first-function-vs-code-other

- https://www.dynatrace.com/news/blog/a-look-behind-the-scenes-of-aws-lambda-and-our-new-lambda-monitoring-extension/

- https://aws.amazon.com/blogs/opensource/rust-runtime-for-aws-lambda/

- https://github.com/ddprrt/serverless-rust

LEARN RUST WITH US 🦀

RUST-TRAINING.EU
RUST-LINZ.AT
STEFAN@SCRIPTCONF.ORG

THANK YOU!