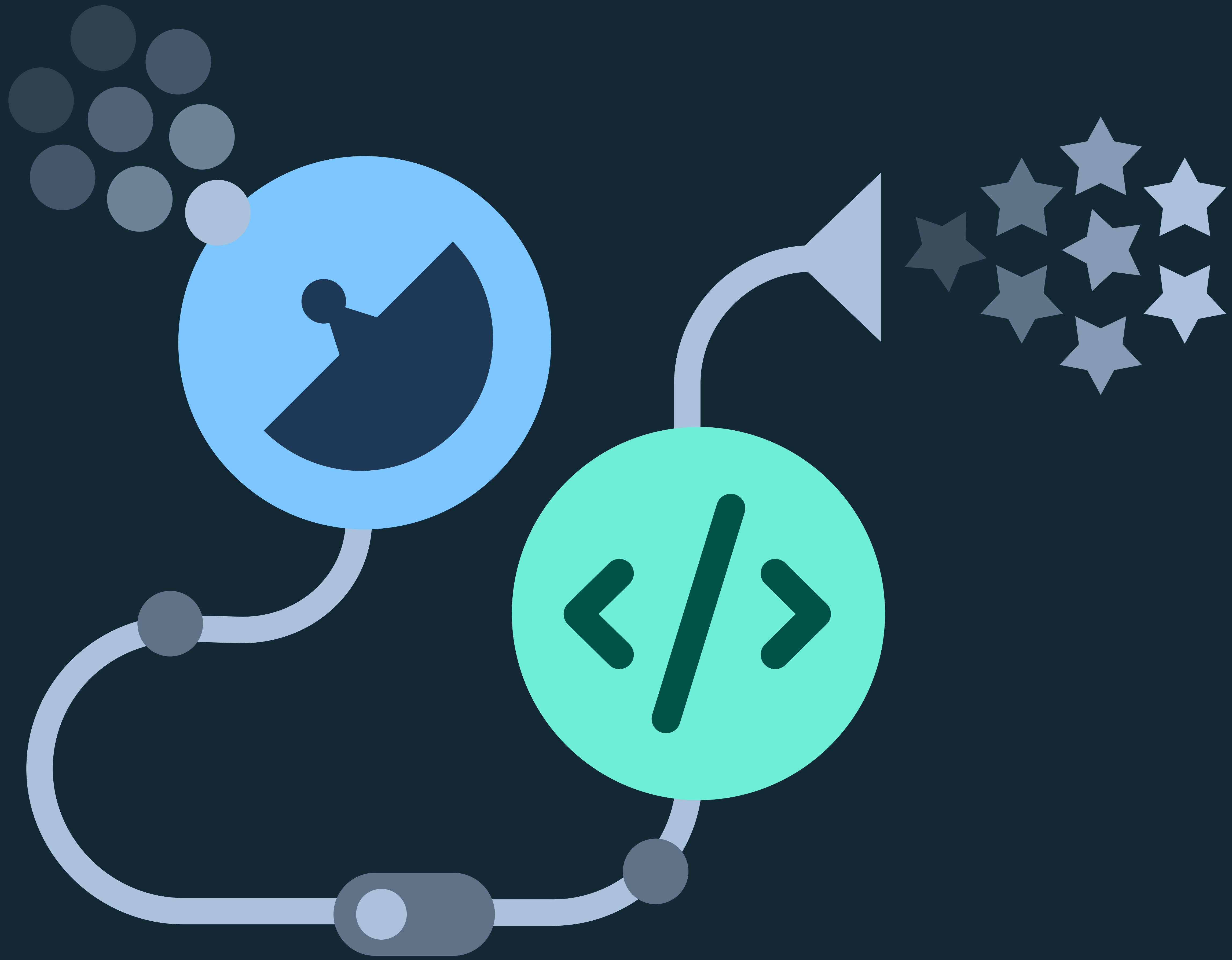
Serverless Computing

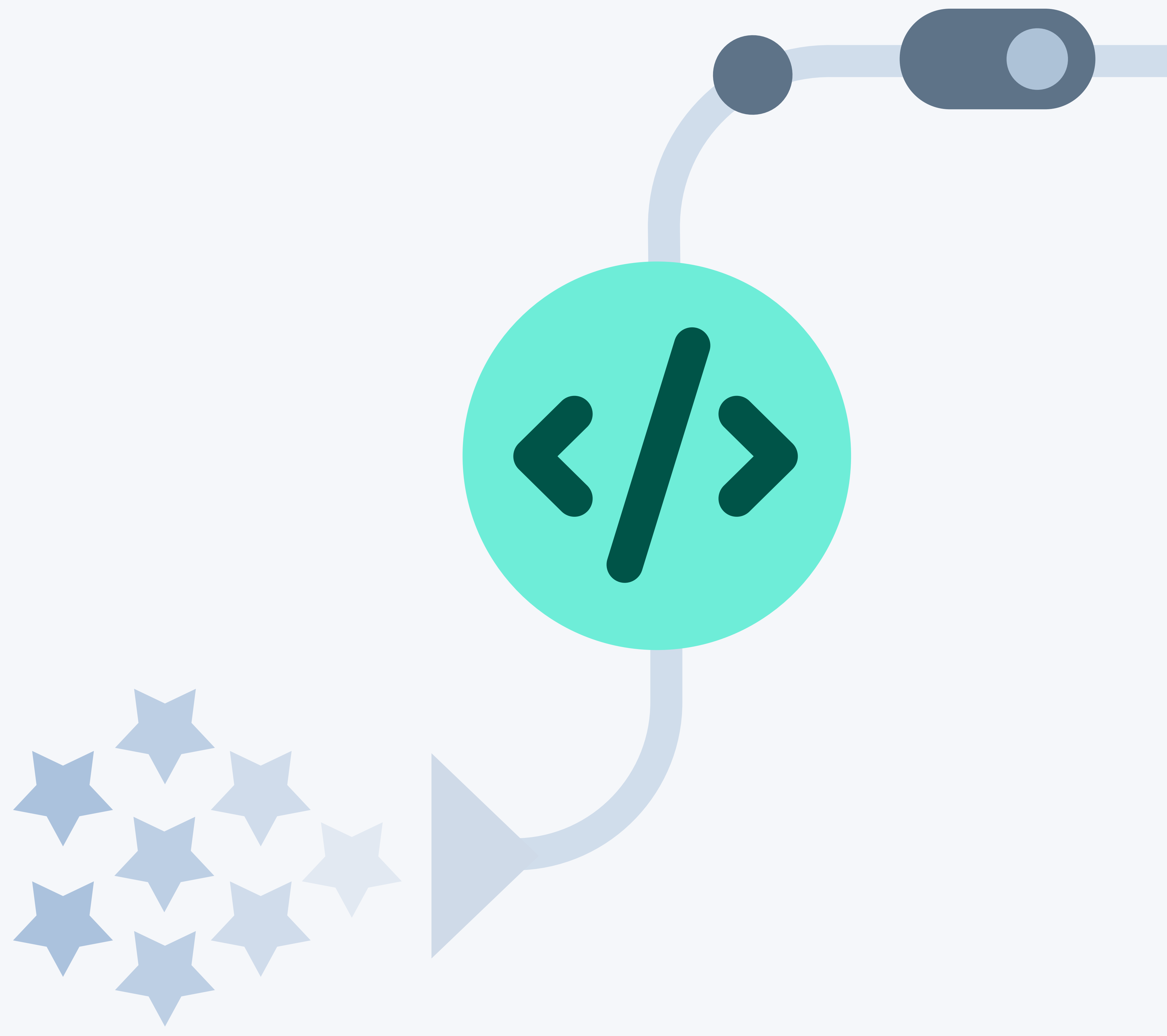
Yves Debeer
IBM Developer Advocate

@yvesdebeer



Agenda

1. What is serverless computing ?
2. The Serverless Ecosystem
3. Serverless Use Cases
4. Serverless Best Practices
5. Serverless Resources



In a nutshell:

When to use serverless?

“I have code or containers I want it to be executed only when needed - at any scale”

What is Serverless Computing ?

Features

1. Major language support (Java, Node, Go, Python)
2. Support for other languages via shims or community workarounds

What is Serverless Computing ?

Cloud Deployment

1. Usually writing for one provider, although frameworks can less pain of switching (e.g serverless framework)
2. Openwhisk allows you to create your own FAAS platform.

What is Serverless Computing ?

Pay-per-invocation

1. Pricing is usually based per-invocation and the amount of memory used
2. Most providers have generous free teers

What is Serverless Computing ?

Scaling

1. SEP: someone else’s problem

Serverless Ecosystem

Major Players

1. AWS Lambda
2. IBM Cloud Functions
3. Google Cloud Functions
4. Azure Functions
5. Other Players: Twillio Functions, Webtask from Auth0, Iron.io IronFunctions, StdLib, SpotInst, ...

Serverless Use Cases

Low-Volume

1. Contact Forms
2. Static (brochureware) websites
3. Automated backups
4. Bots
5. Ops task (policy enforcement, uptime checks)

Serverless Use Cases

Bursty Compute

1. Dataprocessing (especially ETL)
2. Data pipelines of all flavours
3. Event-driven computing (e.g. IoT)

Serverless Use Cases

Heavy Backend Tasks

1. Generating PDFs
2. Image resizing
3. Batch processing

Serverless Use Cases

Tiny Apps

1. APIs
2. Microservices (sometimes called nanoservices)

Don’t run a server all the time if you don’t need a server all the time !

Best Practices

Think stateless ! - Functions may have a clean state each time ... or maybe not

Think Event-Based ! - Make sure you have a plan to handle failed or lost events

Security ! - Keep roles tight to limit access to functions, and to limit what resources functions can access

Resources ! - Tune your functions to use the right amount of memory and watch your execution time (e.g. download of dependencies)

Thing to consider

Development

- Can you develop locally ?
- How to handle unit end integration testing ?

Deployment

- How will you handle versioning ? Rollback ?
- How will your CI/CD process change ?

Monitoring

- How will you profile your code ?
- What will debugging look like ?
- How will you manage and inspect your function logs ?

Service Discovery

- How to kee track of service credentials ?

Gotchas

Size

- If you’re using big libraries, you may run into the upper limit of deployable function size
- Know the limit on the number of concurrent functions and number of deployed functions

Recursion

- If your function calls itself, or call other functions, watch out for infinite loops !

Datastores

- Your function may scale, but can your DB handle the write load ?

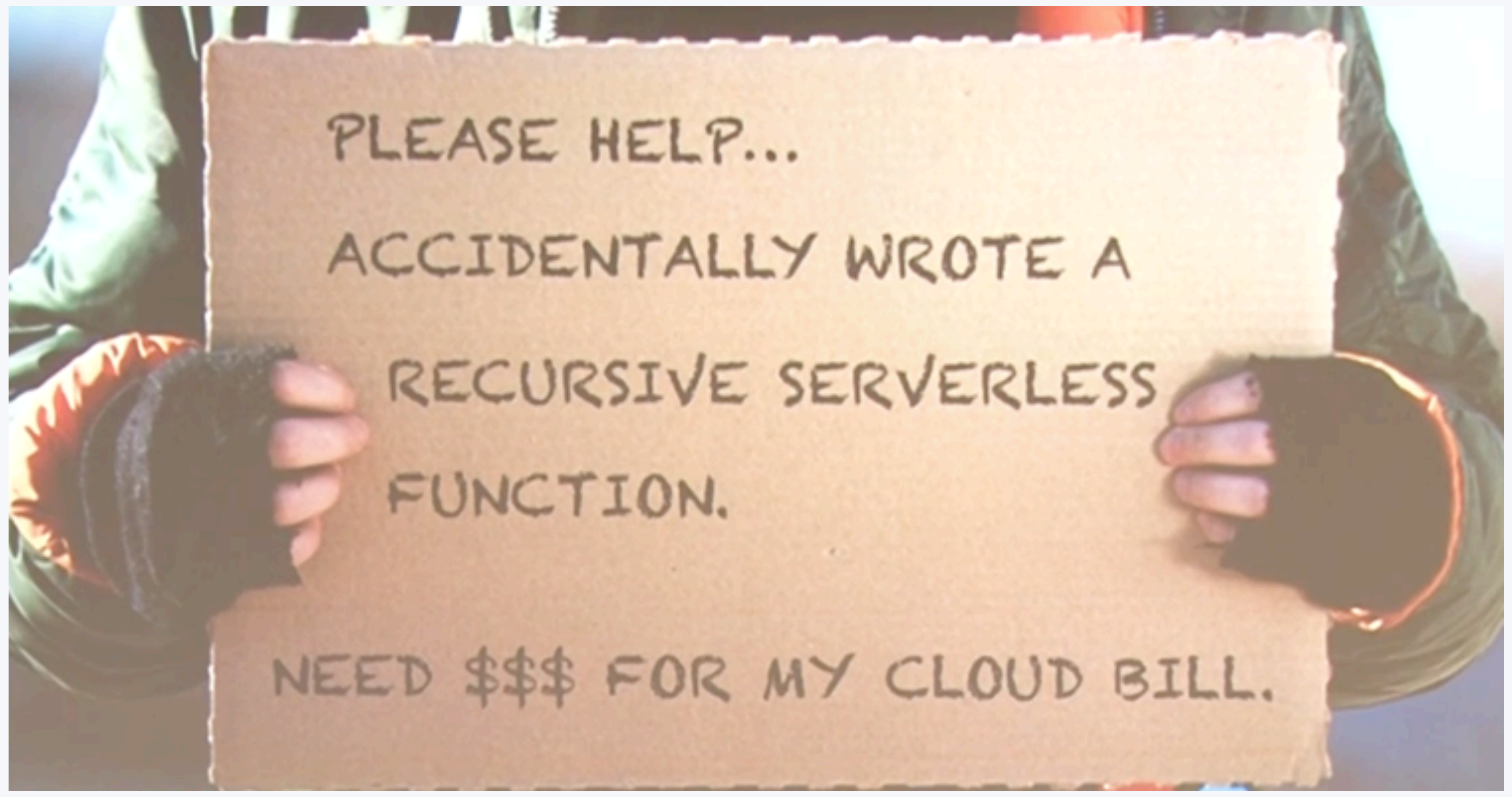
Latency

- Cold starts are a problem !
- Consider keeping your function “warm” with scheduled invocations

Gotchas

Keep in mind warning signs of a malfunctioning serverless function

- recursion
- execution time - amount of memory used
- crashing of dependency out of date


PLEASE HELP...
ACCIDENTALLY WROTE A RECURSIVE SERVERLESS FUNCTION.
NEED $$$ FOR MY CLOUD BILL.

Resources

Lists

github.com/anaibol/awesome-serverless

github.com/pmuens/awesome-serverless

twitter.com/tmclaughbos/lists/serverless

Code & Architecture

developer.ibm.com/code/patters/category/serverless

github.com/serverless/example

martinfowler.com/articles/serverless.html

Documentation

cloud.ibm.com/openwhisk

https://www.ibm.com/blogs/bluemix/2019/02/a-recap-of-the-key-advantages-offered-by-ibm-cloud-functions/

Lab Exercise

Hands-on Lab

cloud.ibm.com/docs/openwhisk?topic=cloud-functions-serverless-api-webapp#serverless-api-webapp

developer.ibm.com/tutorials/learn-serverless-computing-app-filter-mustache