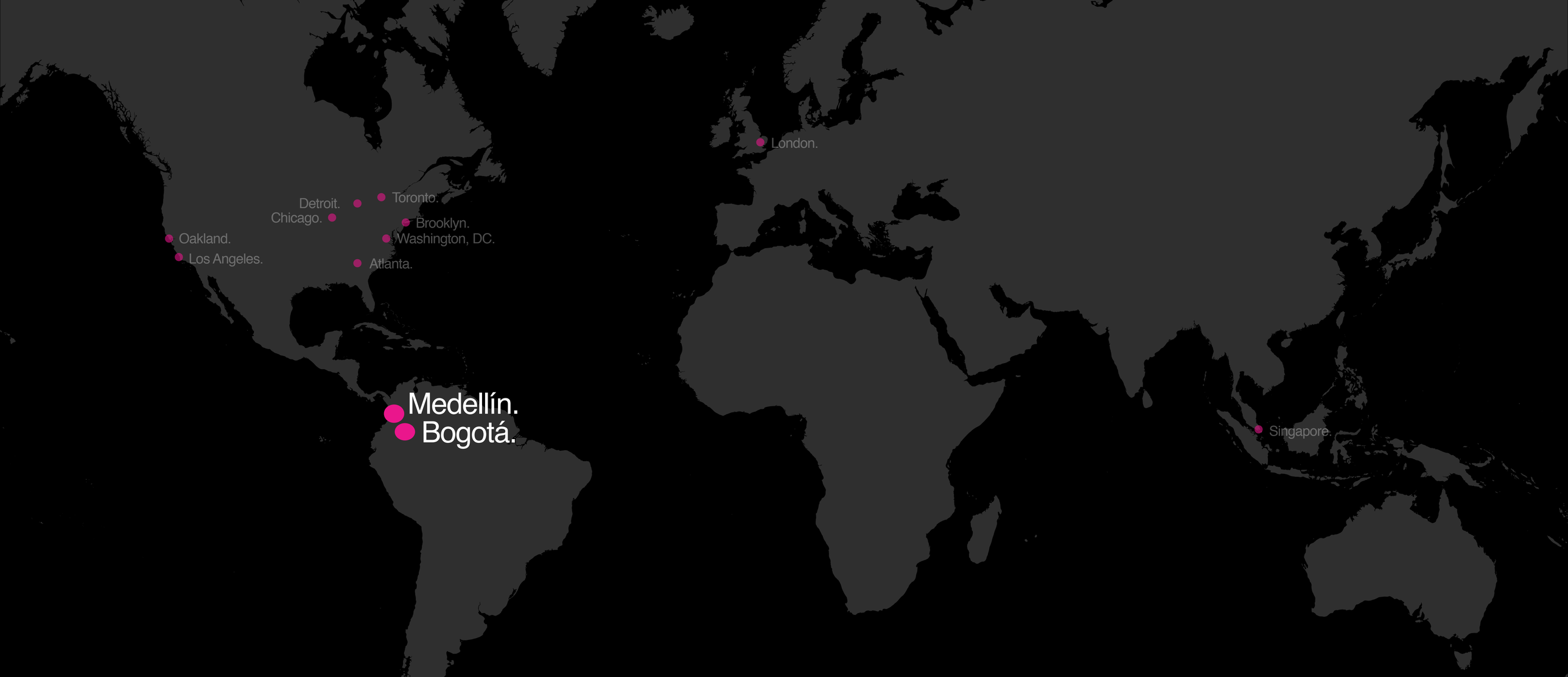
HUGE

Hello Python Medellín.

HUGE

Jorge Galvis
Tech Lead at Huge
@jorlugaqui

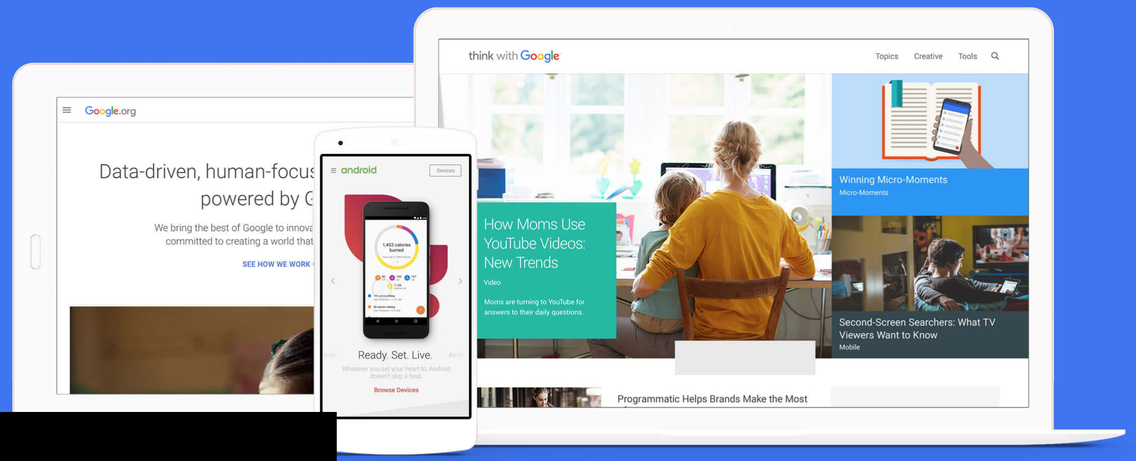


Huge Colombia

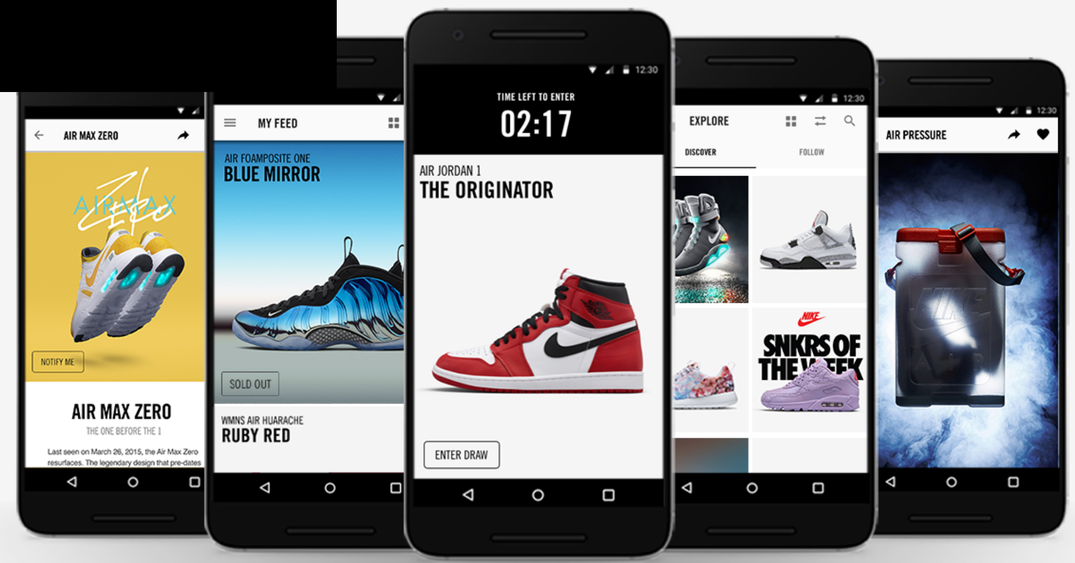
+ 200 people across Bogota and Medellin

H


Born in Brooklyn
in 1999.



We make things people love.



HUGE

Going FaaS with Python.

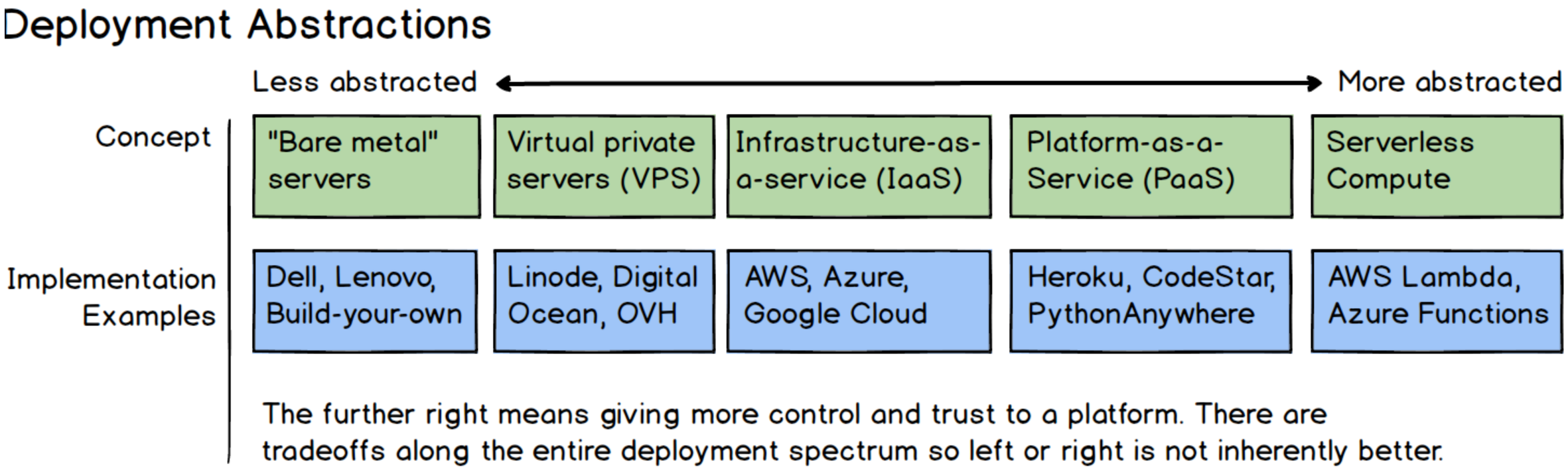
Agenda.

1. What is Serverless/FaaS?
2. Should I go Serverless?
3. Going Serverless with Python.
4. Demo.
5. Conclusions.

1

What is Serverless?

Deployment Abstractions.



Taken from https://www.fullstackpython.com/serverless.html

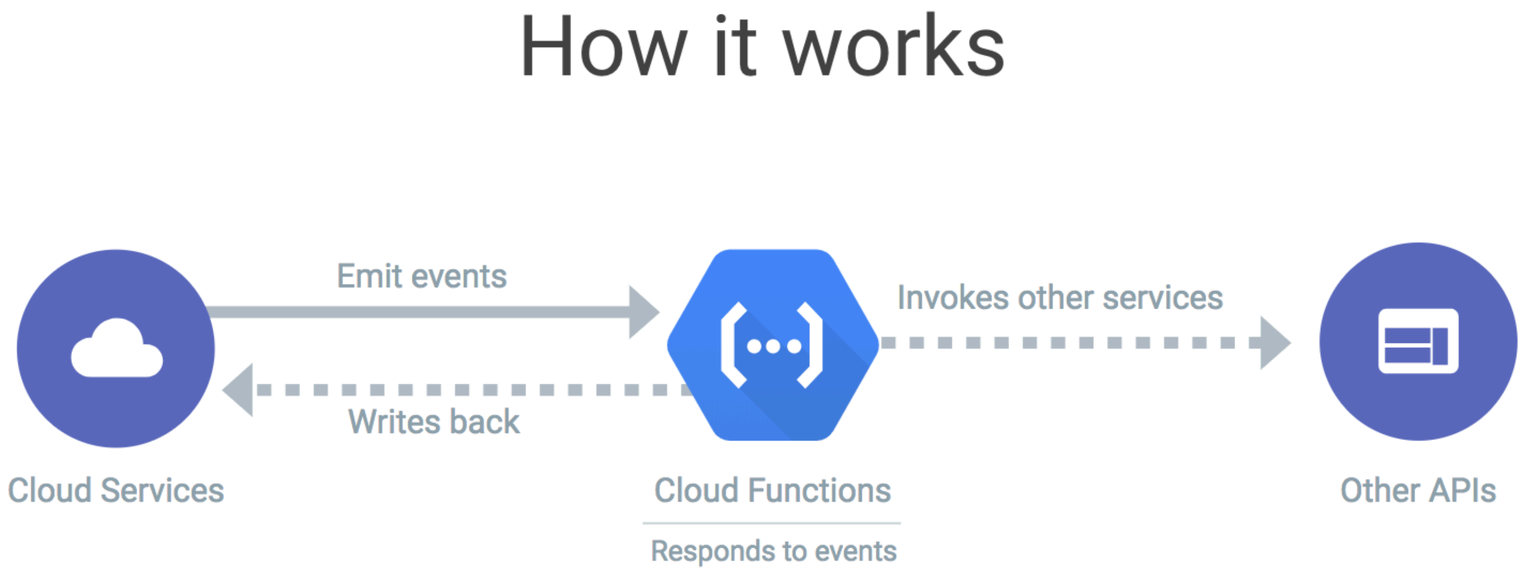
"Serverless is a deployment architecture where servers are not explicitly provisioned by the deployer."

Taken from https://www.fullstackpython.com/serverless.html

“Code is instead executed based on developer-defined events that are triggered…”

Taken from https://www.fullstackpython.com/serverless.html

How it works.



Taken https://cloud.google.com/functions/

Advantages.

Not server managing.

Advantages.

It scales out of the box.

Advantages.

Paying for the compute you use.

Design principles.

Speedy, simpler, singular.

Design principles.

Share nothing.

Assume no hardware affinity.

Design for failures and duplicates.

Should I go Serverless?

2

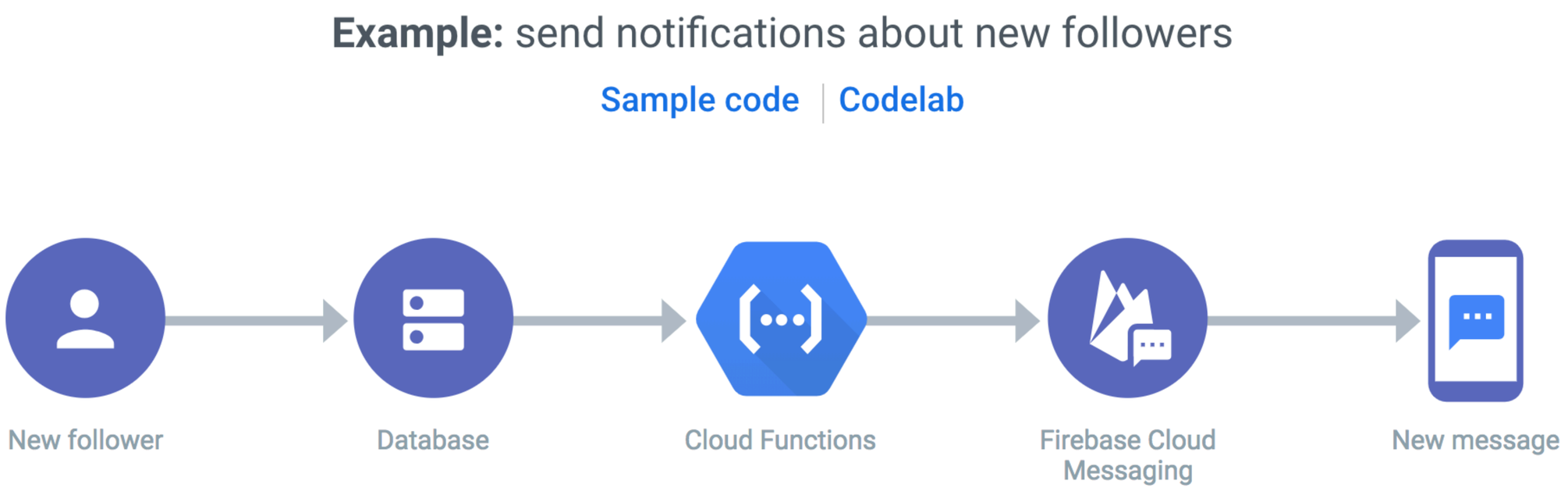
No suitable apps for Serverless.

Apps with web-sockets don’t fit here.
Apps serving static content.

Suitable apps for Serverless.

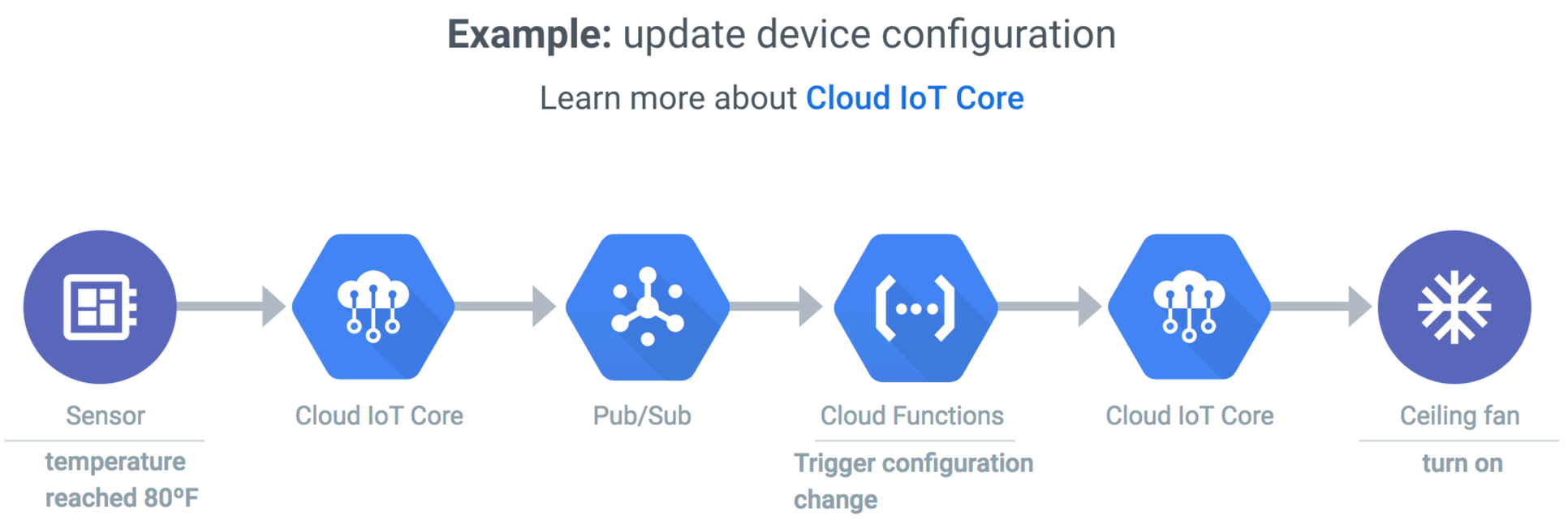
1. Apps with spikes or periods of low activity.
2. Stateless apps (REST APIs).
3. Workers (sending an email, uploading a file, push a notification).
4. IoT Backends.
5. Data Pipelines.

Mobile Backend.



Taken from https://cloud.google.com/functions/use-cases/serverless-application-backends

IoT Backends.

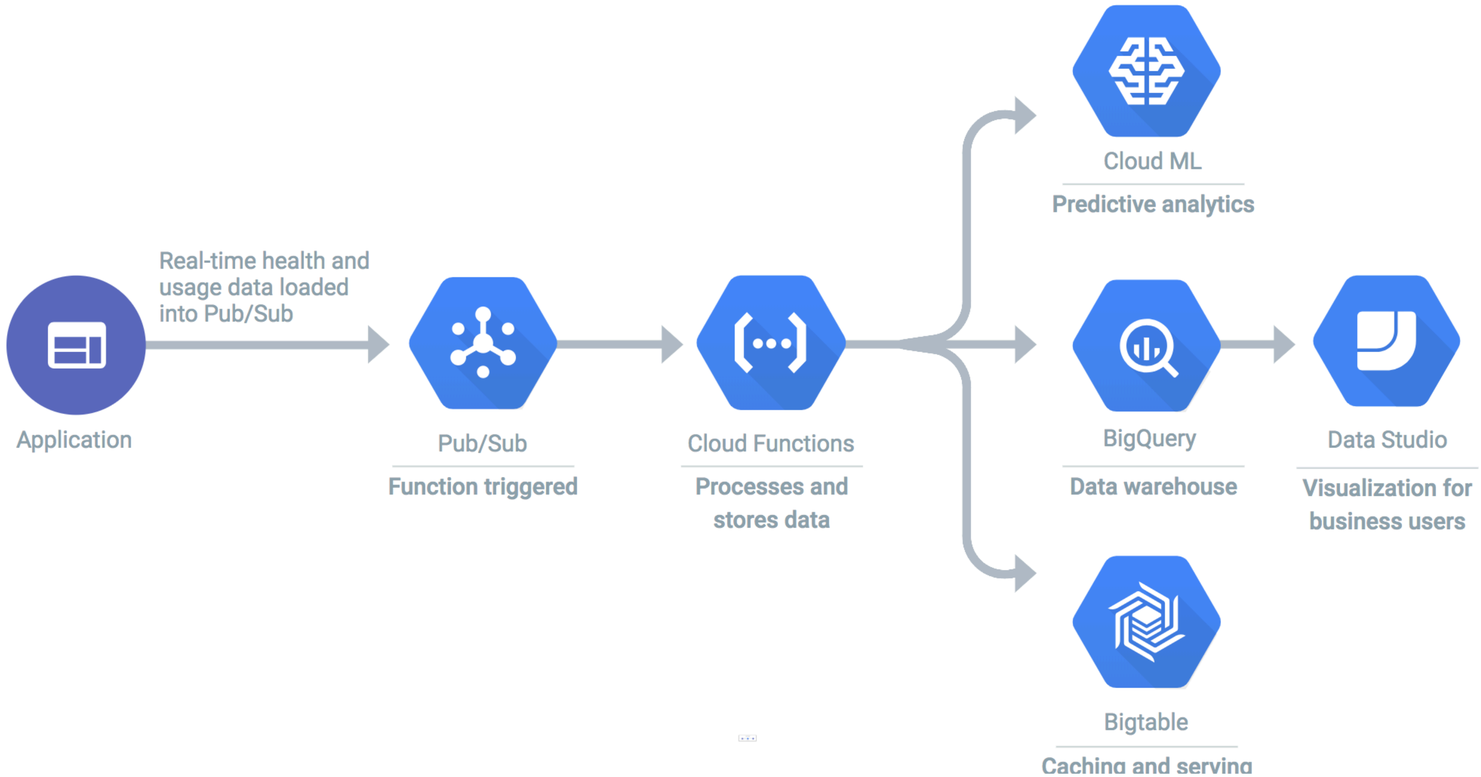


Taken from https://cloud.google.com/functions/use-cases/serverless-application-backends

Real Time Data Processing.

Example: quality-of-service tracking application

Sample Code | Tutorial

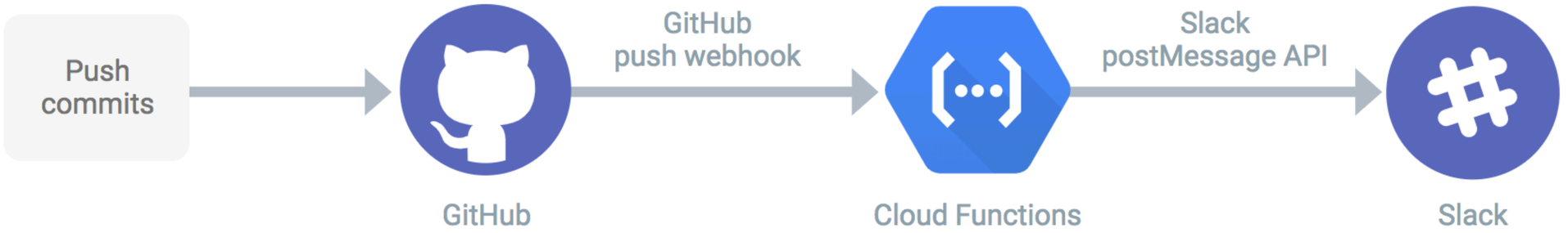


Taken from https://cloud.google.com/functions/use-cases/real-time-data-processing

Integration with APIs.

Example: post a comment on Slack channel in response to a GitHub commit

Sample code | Tutorial



Taken from https://cloud.google.com/functions/use-cases/serverless-application-backends

Vendor lock-in.

Tradeoffs.

Cold starts.

Vendor restrictions (time, size, etc).

Observability is harder.

Tradeoffs.

Taken from https://www.thundra.io/

3

Going Serverless (FaaS) with Python.

Providers.

1. AWS Lambda.
2. Azure Functions.
3. Google Cloud Functions.

On-premise solutions.

1. OpenFaaS.
2. OpenWhisk.
3. Fission.io.

Frameworks.

1. Manual Creation (UI/CLI).
2. Serverless Framework.
3. Zappa/Chalice.

1. GUI (browser - console UI).
2. CLI (gcloud).

4

Conclusions.

Conclusions.

1. FaaS is another architectural approach for apps.
2. Not all apps fit on this approach.
3. FaaS allows to save money if it is used wisely.
4. It autoscales.

References.

1. https://www.fullstackpython.com/serverless.html
2. https://read.iopipe.com/the-right-way-to-do-serverless-in-python-e99535574454
3. https://d1.awsstatic.com/whitepapers/architecture/AWS-Serverless-Applications-Lens.pdf
4. https://servers.lol/
5. https://www.iopipe.com/what-is-amazon-lambda/
6. https://read.iopipe.com/understanding-aws-lambda-coldstarts-49350662ab9e
7. https://thenewstack.io/serverless-architecture-five-design-patterns/
8. https://serverless.com/blog/flask-python-rest-api-serverless-lambda-dynamodb/
9. https://martinfowler.com/articles/serverless.html
10. https://code.roygreenfeld.com/comparisons/zappa-vs-serverless-framework.html
11. https://hackernoon.com/aws-lambda-serverless-framework-python-part-1-a-step-by-step-hello-world-4182202aba4a
12. https://itnext.io/writing-google-cloud-functions-with-python-3-49ac2e5c8cb3
13. http://www.iamondemand.com/blog/serverless-for-humans-a-simplified-overview/
14. https://medium.com/google-cloud/deploying-a-python-serverless-function-in-minutes-with-gcp-19dd07e19824
15. https://cloudonair.withgoogle.com/events/app-dev/watch?watching=amer-track1-fundamentals

I wrote something.

http://www.iamondemand.com/blog/serverless-for-humans-a-simplified-overview/

HUGE

Done.