The future will be SERVERLESS

Luciano Mammino (@loige)

Front Conf

Munich, 9 Dec 2017

loige.link/serverless-future

loige.link/serverless-future

Luciano... who?

... just a Fullstack developer

Let's connect

Twitter (@loige)

GitHub (lmammino)

Linkedin

https://loige.co

Agenda

Chapter 1: from bare metal to Serverless

Chapter 2: Serverless, WTF?!

Chapter 3: Understanding Serverless

Chapter 4: A Serverless use case

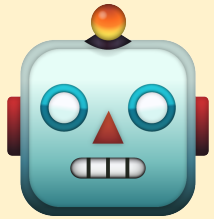
Chapter 5: PROs n' CONs

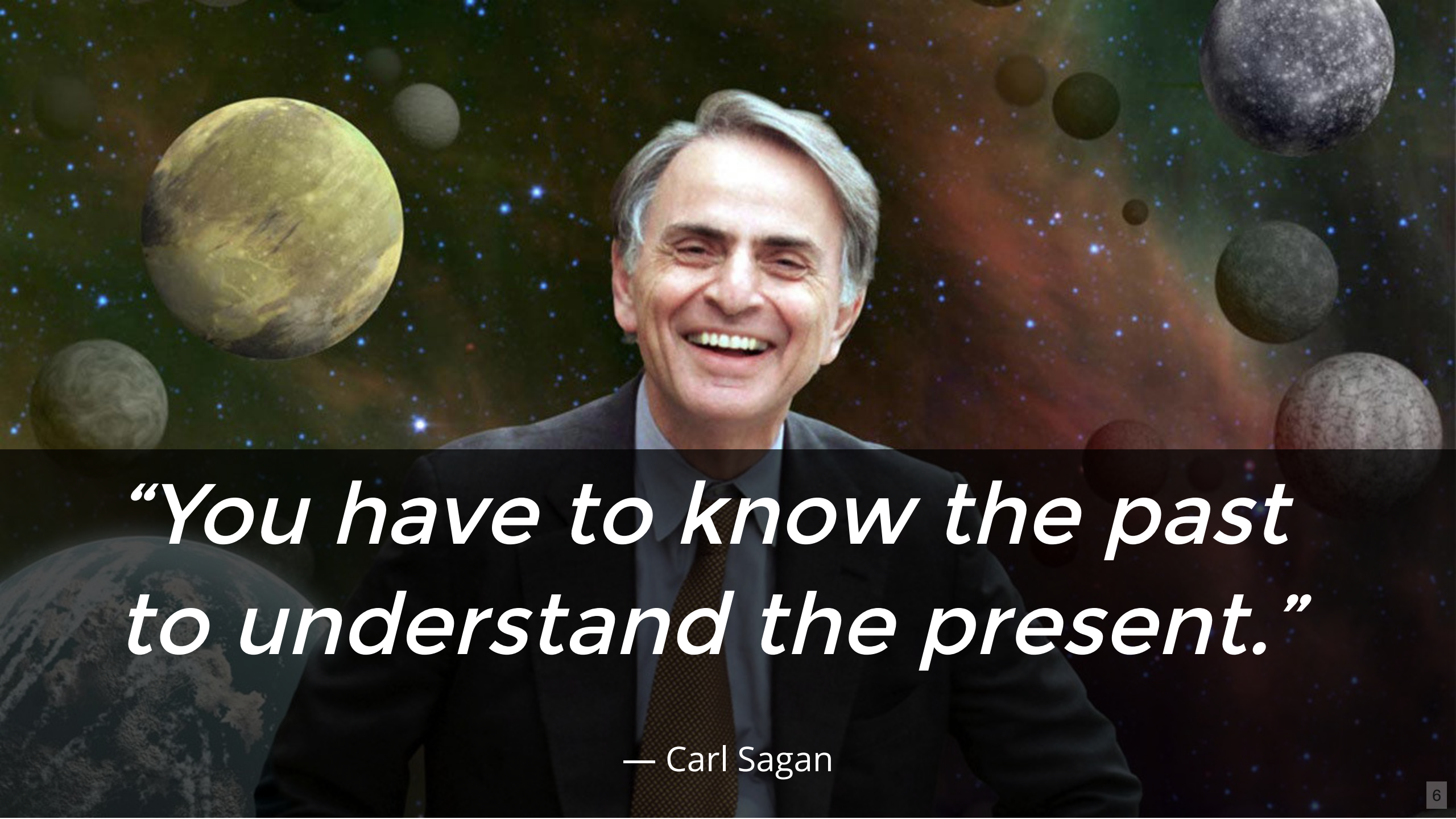
Chapter 6: It's time to get started

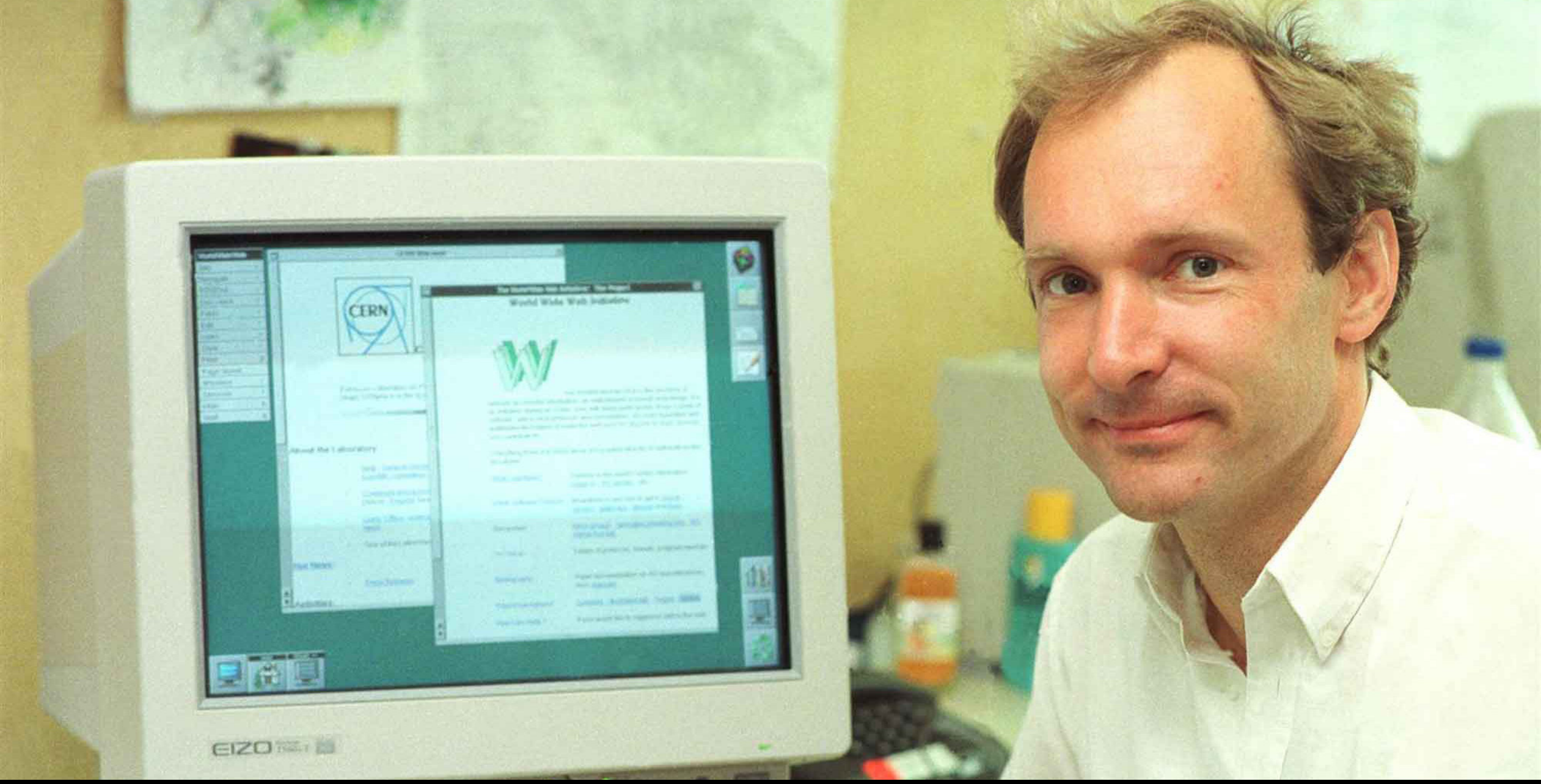
Chapter 1

from *bare metal* to *Serverless*






“You have to know the past
to understand the present.”
— Carl Sagan



1989-1991 — Sir Tim Berners-Lee invented the World Wide Web

1991-1995 — The bare metal age

1995 — The invention of web hosting

Marc Benioff

1999 — Salesforce introduces the concept of Software as a Service (SaaS)

2001 — VMWare releases ESXi, "server virtualization" becomes a thing

2002-2006 — AWS is born (IaaS), people talk about "Cloud computing"

James Lindenbaum

Adam Wiggins

Orion Henry

2009 — Heroku and the invention of the "Platform as a Service" (PaaS)

James Tamplin and Andrew Lee

2011 — Envolve/Firebase, real time database as a service (RTDaaS???)

Tikhon Bernstam

Ilya Sukhar

James Yu

Kevin Lacker

2012 — Parse.com and the first Backend as a Service (BaaS)

2013 — Docker, "containers are better than virtual machines"

2013-2015 — Kubernetes / Swarm / Nomad / CoreOs (containers at scale)

2014 — Launch of AWS Lambda (FaaS)

Chapter 2

Serverless, WTF?!*

*What's The Fun

"Serverless most often refers to **serverless applications**. Serverless applications are ones that **don't require you to provision or manage any servers**. You can **focus on your core product and business logic** instead of responsibilities like operating system (OS) access control, OS patching, provisioning, right-sizing, scaling, and availability. By building your application on a serverless platform, the platform manages these responsibilities for you."

— Amazon Web Services
loige.link/serverless-apps-lambda

> *The essence of the **serverless** trend is the absence of the server concept **during software development**.*
>
> *— Auth0*

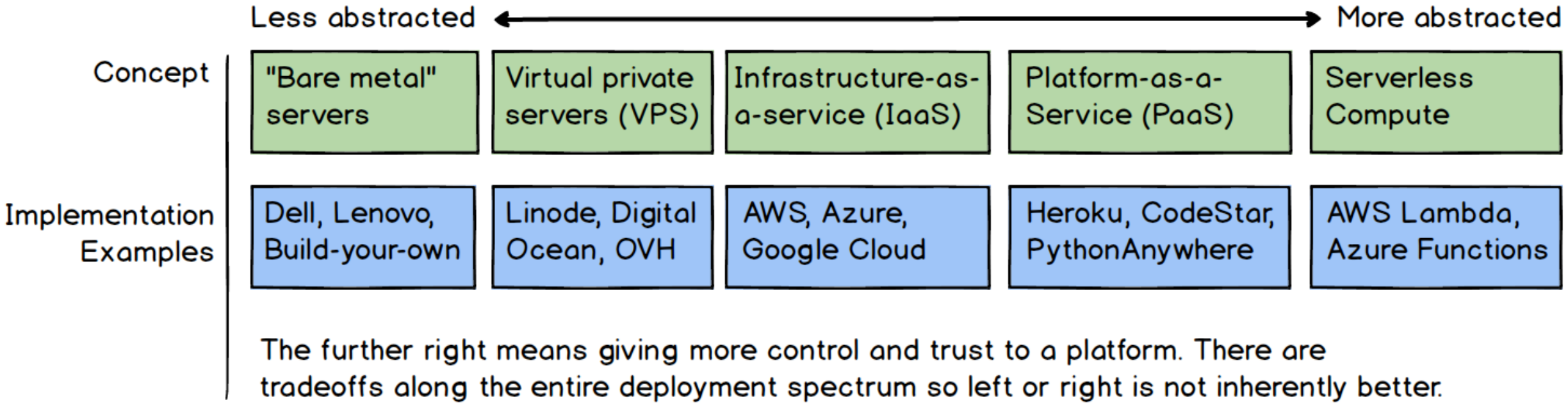
loige.link/what-is-serverless

loige.link/serverless-commitstrip

Chapter 3

Understanding Serverless

Deployment Abstractions



loige.link/serverless-abstraction

The 4 pillars of serverless

(TLDR; It's not only about servers)

No server management

You don't know how many and how they are configured

Flexible scaling

If you need more resources, they will be allocated for you

High availability

Redundancy and fault tolerance are built in

Never pay for idle

Unused resources cost $0

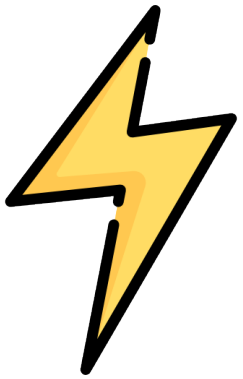
The serverless layers

(TLDR; It's not only "FaaS")

- 👉 Compute
- 👉 Data
- 👉 Messaging and Streaming
- 👉 User Management and Identity
- 👉 Monitoring and Deployment
- 👉 Edge

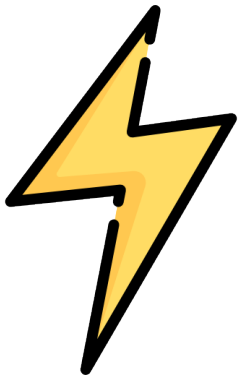
Stuff that we can build

- 📱 Mobile Backends
- 🔌 APIs & Microservices
- 📦 Data Processing pipelines
- ⚡ Webhooks
- 🤖 Bots and integrations
- ⚙️ IoT Backends
- 💻 Single page web applications



execution model

Event → f



"IF this THEN that" model

IF _____

THEN _____

Serverless and JavaScript

Frontend

🌏 Serverless Web hosting is **static**, but you can **build SPAs**

(React, Angular, Vue, etc.)

Backend

👌 **Node.js** is supported by every provider

⚡ **Fast startup** (as opposed to Java)

📦 Use all the modules on **NPM**

🤓 Support other languages/dialects

(TypeScript, ClojureScript, ESNext...)

Anatomy of a Node.js lambda on AWS

```
exports.myLambda = function (
    event,
    context,
    callback
) {

  // get input from event and context

  // use callback to return output or errors

}
```

Chapter 4

A serverless use case

ID/TITLE: Duos invoicing data

DESCRIPTION:

In order to consume DuoS data for invoicing

As an engineer

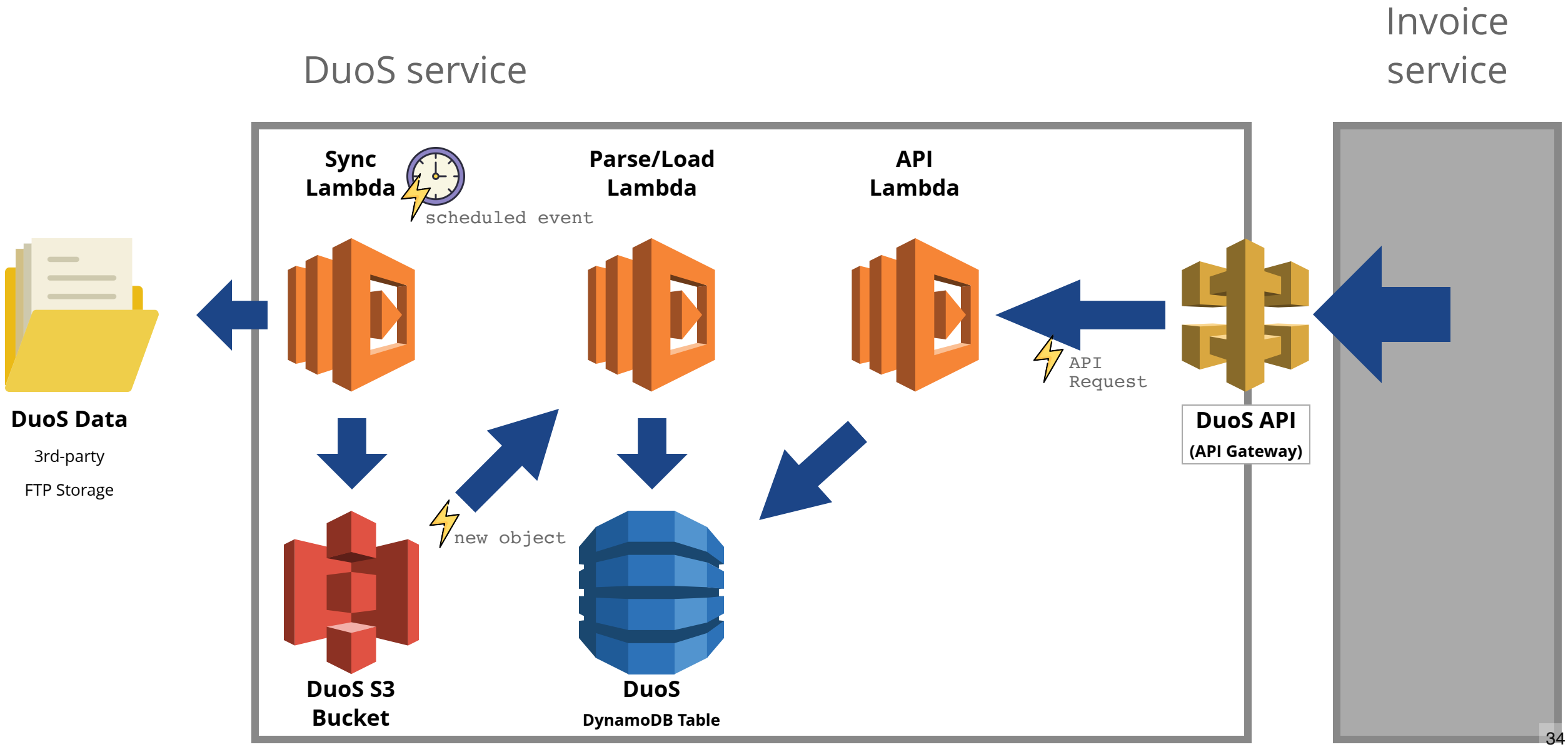
I Want to have a process that keeps data from the source FTP in sync

And exposes it as a REST API

POINTS: A lot

PRIORITY: For yesterday

A serverless implementation (on AWS)



(Many) things I didn't have to worry about...

- What kind of virtual machine do I need?
- What operative system?
- How to keep OS/System updated?
- How much disk space do I need?
- How do I handle/retry failures?
- How do I collect and rotate logs?
- What about metrics?
- What machine do I need to run the database?
- How do I backup the database?
- How do I scale the database?
- Which web server should I use and how to configure it?
- Throttling? Managing API Keys? API caching?

Chapter 5

PROs n' CONs

Focus on delivering business value / Fast time to market

Less "Tech-freedom™" / Tight vendor lock-in!

Optimal resource allocation

Not-magic™!

You still have to write configuration

Auto-scalability

Cold start problem

loige.link/cold-start

High Availability

Soft/Hard Limits

loige.link/lambda-limits

Pay per usage

(don't pay for idle!)

Local development, Testing, Debugging

Growing ecosystem

Older technologies might not integrate well

Chapter 6

It's time to get started

Who is already adopting Serverless

Pick one and start to have fun!

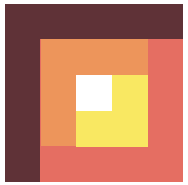
Cloud based

IBM Cloud Functions

AWS Lambda

Azure Functions

Google Cloud Functions



Auth0 Webtask

Iron.io FaaS

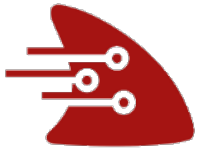
Spotinst Functions

stdlib service Functions

Self-hosted / Open Source

Apache OpenWhisk

Fission



Fn

Kubeless

effe

LeverOS

Why is this the right direction for the future?

2 main reasons

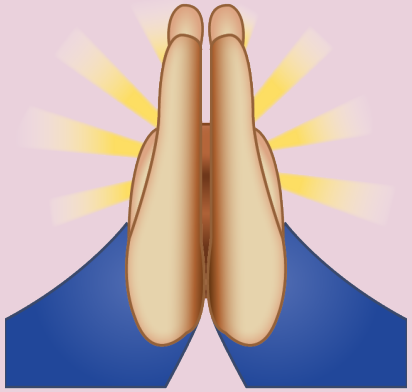
Opportunity to deliver value to customer quickly

Pay only for the used resources

Should I migrate all my apps to serverless?

Approach this with care...

Thanks!



Questions?

Now or later to @loige :)

If your company wants to get started with serverless on AWS, be sure to check out serverlesslab.com

loige.link/serverless-future

Credits

- webfoundation.org/about/vision/history-of-the-web/
- en.wikipedia.org/wiki/Web_hosting_service
- loige.link/web-hosting-history
- computerweekly.com/feature/A-history-of-cloud-computing
- salesforceben.com/brief-history-salesforce-com
- aws.amazon.com/about-aws
- fullstackpython.com/serverless.html
- en.wikipedia.org/wiki/Parse_(platform)
- en.wikipedia.org/wiki/Firebase
- en.wikipedia.org/wiki/Kubernetes
- en.wikipedia.org/wiki/Container_Linux_by_CoreOS
- en.wikipedia.org/wiki/AWS_Lambda
- en.wikipedia.org/wiki/Serverless_computing
- loige.link/aws-serverless-lens
- loige.link/serverless-apps-lambda
- start.jcolemorrison.com/aws-lambda-vs-the-world
- m.subbu.org/serverless-looking-back-to-see-forward-74dd1a02cb62
- github.com/anaibol/awesome-serverless

a **HUGE** thanks to:

@acambas_sasa

@katavic_d

@Podgeypoos79

@lakatos88

Cover photo by Tobias Zils on Unsplash
High Res Emojis by emojiisland.com