inovex

Vier Deployments für ein Halleluja

Aus dem Leben eines Softwaretherapeuten

Michael Bruns

Karlsruher Entwicklertag, 04.06.2019

Let’s start with a story...

- Real-life conversation from a project:
  - Manager: “We need Microservices!”
  - Me: “Ok. Why?”
  - Manager: “Err…”
  - Me: “Did you at least talk to Dev and Ops?”
  - Manager: “Err…”

Let’s start with a story...

- Real-life conversation from a project:
  - Me: “Can we operate the VMs with our team?”
  - Manager: “No, the Ops team does that.”
  - Me: “Can we shift people from the Ops team to our team?”
  - Manager: “No.”
  - Me: “Ok… Then I need to talk to the Ops team.”
  - Manager: “You can’t, they’re too busy with their roadmap.”

Let’s start with a story...

- Real-life conversation from a project:
  - Me: “We need two small VMs, one per data center.”
  - Ops: “We don’t do it that way. You’ll get three per data-center, 64 GB RAM and 6 CPUs each.”
  - Me: “What?! Well, alright... Is it possible to get them by this afternoon?”
  - Ops: “Nah, average delivery time is eight weeks.”

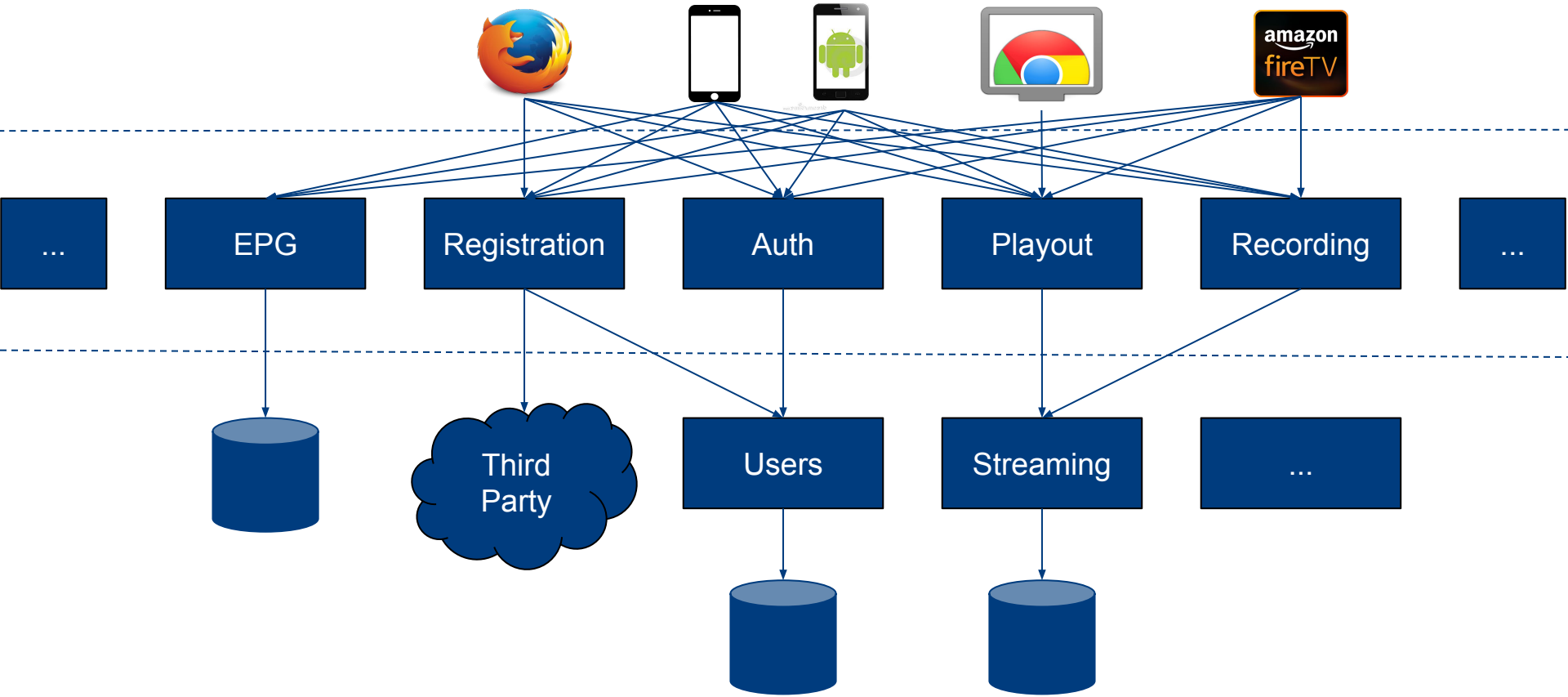
Michael Bruns

inovex GmbH

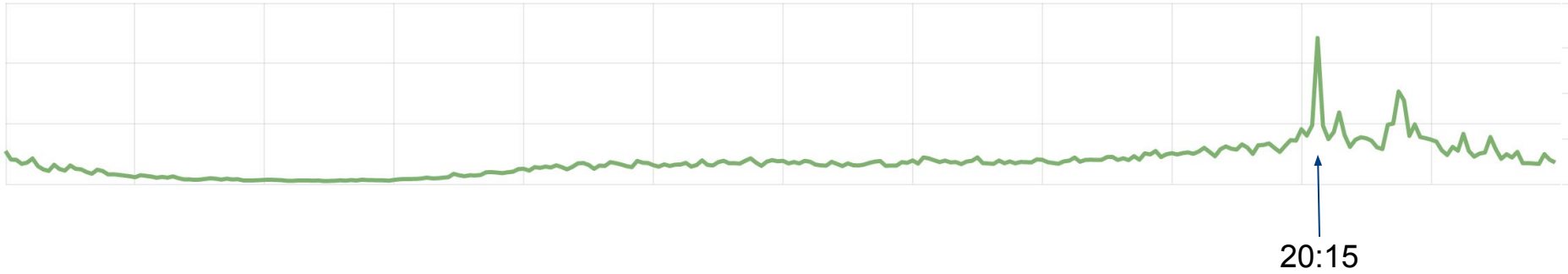
Please allow me to introduce myself:

- Software developer, architect, therapist, teamlead, ...
- Loves boring solutions
- Twitter: @der_miggel

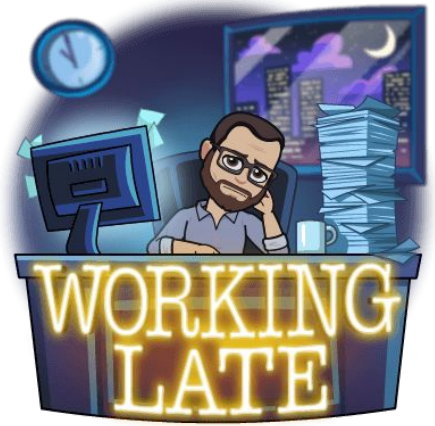
Example architecture from a project



How do people use it?



- Anybody who likes working at 20:15 every day?
- Including and especially on weekends?



A short quiz

- Q: How many people watched Tatort on 27 May 2018?
- A: 12.00m
- Q: What's the market share of this?
- A: 37.8%

(Source: http://bit.ly/2S2Ya4v)

How did we build it?

- Don’t build a platform, use one: AWS, Azure, GCP, ...
- Use what’s already there
- Add shared stuff (e.g. JVM, nginx) to base image (AMI)
- Leave the rest to the services

How did we build it?

- Foster DevOps, i.e. tear down all political and technological barriers
- Make infrastructure reproducible
- Choose the right tool for the job:
  Terraform, GitLab CI, Prometheus, …

Use the right tool for the job...

...and use it wisely!

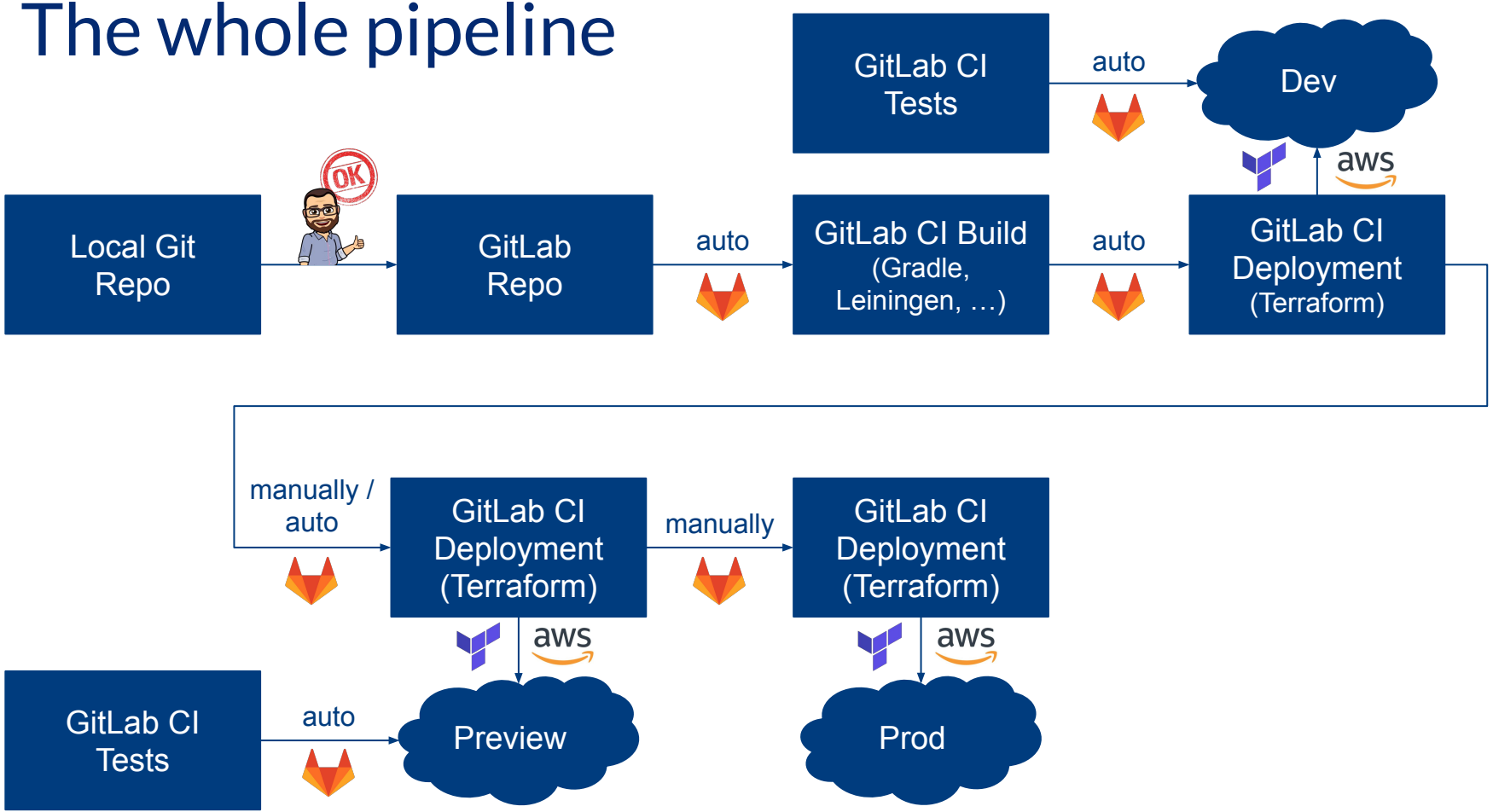
Terraform

- Infrastructure as Code
- Independent of provider (AWS, Azure, OpenStack, …)
- Reproducible
- Multi-cloud strategy is possible
- https://www.terraform.io/

GitLab CI

- Continuous Integration & Deployment
- Pipelines
- Stages, environments, variables, tags, …
- Easy integration of runners

The whole pipeline



Automatic scaling

- Launch Configuration for services
- Automatic scaling of services based on
  - Network/CPU usage
  - Number of requests
  - Schedules

Automatic scaling - Example on AWS

Filter: Filter scheduled actions... 1 to 2 of 2 Scheduled Actions

Name	Start Time	End Time	Recurrence	Desired Capacity	Min	Max
daily-peak-start	2019 May 3 19:45:00 UTC+2		45 17 * * *	4	2	6
daily-peak-end	2019 May 3 22:45:00 UTC+2		45 20 * * *	2	1	4

Another short quiz

- Fact 1: If 500 customers record a show, 500 copies of the show have to be made
- Fact 2: Shows are recorded 10 minutes “too long”
- Q: Which two shows put the highest load on the recording backend because they overlapped?
- A: Der Bachelor and Ich bin ein Star, holt mich hier raus

“Micro”services

- Loosely coupled, small services
- Isolated from each other
- Supposed to fit into your head
- But what if they **don’t** just because of the sheer number of services?!

Microservices - Some anti-patterns

The more the merrier

(Source: https://www.flickr.com/photos/frans16611/9138604666)

Microservices - Some anti-patterns

Microservices as the panacea

Microservices - Some anti-patterns

Microservices as the goal of a project

(Source: https://pxhere.com/en/photo/1364310)

Microservices - Some anti-patterns

Scattershot approach

(Source: https://pxhere.com/de/photo/181542)

Microservices - Some anti-patterns

Flying before you learned to walk

(Source: https://funnycrazyanimals.blogspot.com/2009/01/crash-landing-duck-in-snow-pic.html)

Unmaintainable monolith in the cloud

- You will get automatic scaling (of errors)
- You will get continuous delivery (of a big ball of mud)
- You will get a better distribution (of the stuff you already didn't know how to debug before and which you now can't even find because you aren't writing the logs to a central store yet and once the machine is gone all logs are gone and now you sit in your office yelling and…)

Diagram of two microservices and their shared database

Source: https://twitter.com/davecheney/status/1125288279044120576

Most important advice

Don’t put your unmaintainable monolith into the cloud and expect things to get better!

Just deploy it to Kubernetes!

ryan huber
@ryanhuber

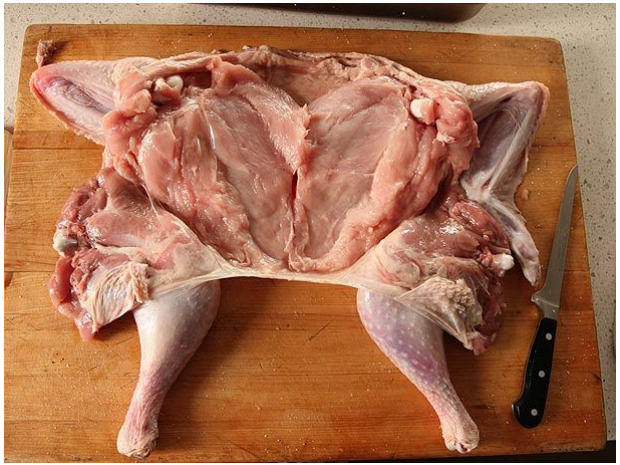
Folgen

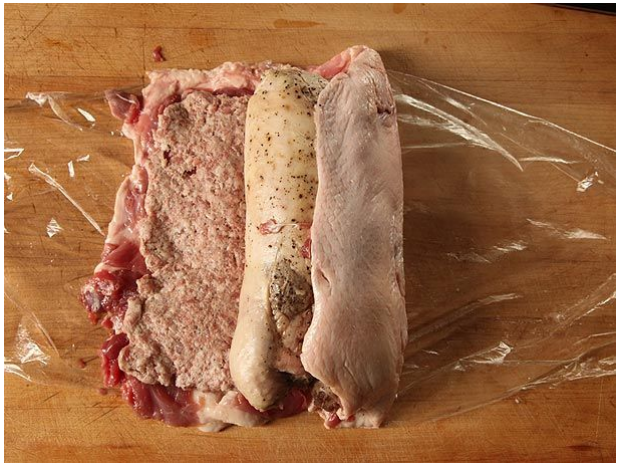
Kubernetes: Because every problem can be solved with a Google sized solution.

15:41 - 20. März 2019

Source: https://twitter.com/ryanhuber/status/1108498814854283264

The Turducken - Creation





http://www.seriouseats.com/2012/11/the-food-lab-the-ulimate-turducken.html

The Turducken - Result

http://www.seriouseats.com/2012/11/the-food-lab-the-ulimate-turducken.html

The Turducken - My two cents on containers

- Kubernetes & Docker is a cloud in the cloud
- It adds several layers of abstraction
- You should prepare the interior first, and add the outer layers when you know how the interior works
- It takes an experienced person to handle it

Corey Quinn
@QuinnyPig

Folgen

If you run Kubernetes in production, you must be incredibly intelligent.

That's not a compliment; it's a requirement.

10:13 - 25. Apr. 2019

Source: https://twitter.com/QuinnyPig/status/1121462122741940226

The bad news

- A lot of the things I talked about so far are just tools
- A fool with a tool is still a fool
- Tools won't fix your people issues

Why do people become developers?

- 2%: Because they want fame
- 8%: Because they want money
- 12%: Because they want to have fun when coding
- 78%: Because they want to interact with people

FAKE!

The social aspects

- Talk to each other
- Take responsibility
- Don’t point your finger at someone
- Don’t fall into this trap:
  “Weeks of coding can save you hours of planning!”

Source: https://twitter.com/davecheney/status/1125288279044120576

Share the operational load

- Firefighter for bugs or urgent issues
- Take turns
- Create an easy way to notify the person in charge (e.g. @firefighter)
- Don’t be afraid to ask for help

Embrace change

- Be open for constant change
- However, don’t change everything just because you can
  - Sometimes legacy is not necessarily a bad thing
  - Some changes might have consequences you don’t expect

Track change

- Create a central board to request improvements
- Let the team make decisions which improvements are most important
- Accept if the others think that your suggestion isn’t useful

The Mantra

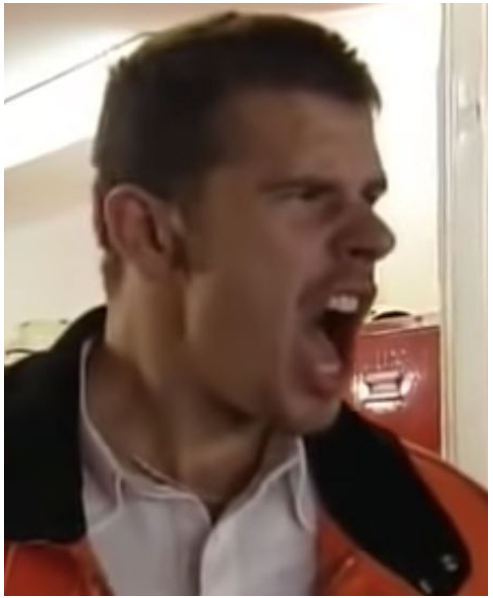
- Repeat these sentences ten times every day:
  - We are not Netflix
  - We are not Google
  - We are not Facebook
  - We are not Spotify

Coupling services

- Don’t build a single point of failure
- Couple your services not as loosely as possible, but as loosely as feasible
- Do you really need Kafka, Event Sourcing, CQRS, Service Mesh, …?

Monitor your services

- ALARM! ALAHAARM!
- Send alarms to Slack/Teams/…
- Use tools like Grafana to review changes in the load, requests etc.



DevOps - How not to do it

“One common anti-pattern when introducing DevOps to an organization is to assign someone the role of 'DevOps' or to call a team a 'DevOps team'. Doing so perpetuates the kinds of silos that DevOps aims to break down and prevents DevOps culture and practices from spreading and being adopted by the wider organization.”

(Rouan Wilsenach, https://martinfowler.com/bliki/DevOpsCulture.html)

The fear of DevOps

› There's this one person constantly calling me at the most awkward times because something isn't working:

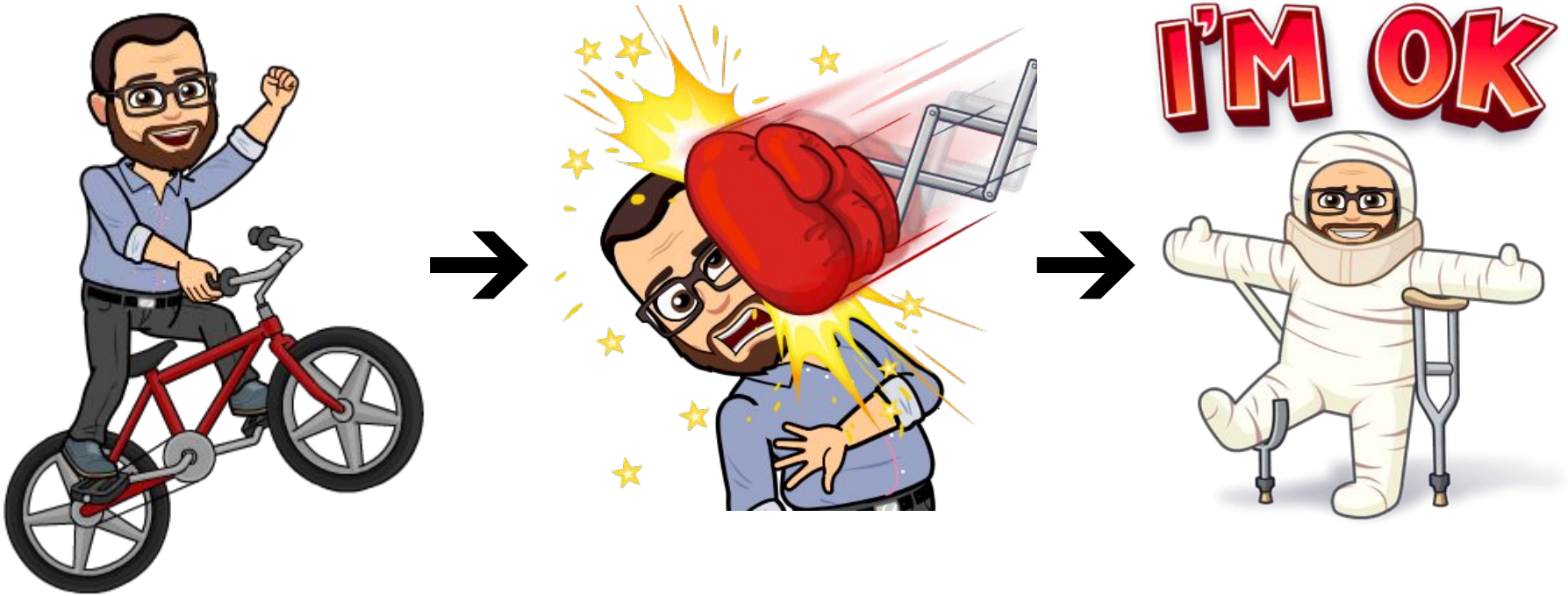
› But my mum's phone not working the way she wants it to work has nothing to do with DevOps!

Of course this isn't really my mum!
And her calls are perfectly fine.

Another real-life conversation

- Dev 1: “We need to update service X to fix issue Y.”
- Dev 2: “Meh. We haven’t touched it for ages. We haven’t updated the CI pipeline. I have absolutely no idea if the deployment still works. Dev 3 built this in 2015 and he’s on vacation.”
- That’s not DevOps, that’s DevOops

One final note...



One final note...

- Wear a helmet when riding your bike, i.e.:
- Expect failure
- The person to trust least is you
- Accept help

Vielen Dank

Michael Bruns

inovex GmbH
Ludwig-Erhard-Allee 6
76131 Karlsruhe

michael.bruns@inovex.de
Twitter: @der_miggel

Links

› Inspiration for anti-patterns: https://microservices.io//microservices/general/2018/11/04/potholes-in-road-from-monolithic-hell.html