KubeCon | CloudNativeCon

Europe 2019

Serverless Operations

From dev to production

Erwin van Eyk

PLATFORM9

fission

@Large Research
Massivizing Computer Systems

spec RG Cloud

Why serverless computing?

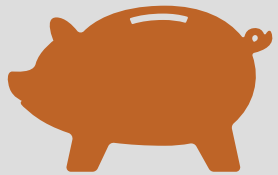
Operational Model

Minimal operational logic

“Infinite” autoscaling

Built-in tooling: monitoring, tracing, health checking, etc.

Cost Model



Pay for what you use

No upfront/periodic costs

Granular billing

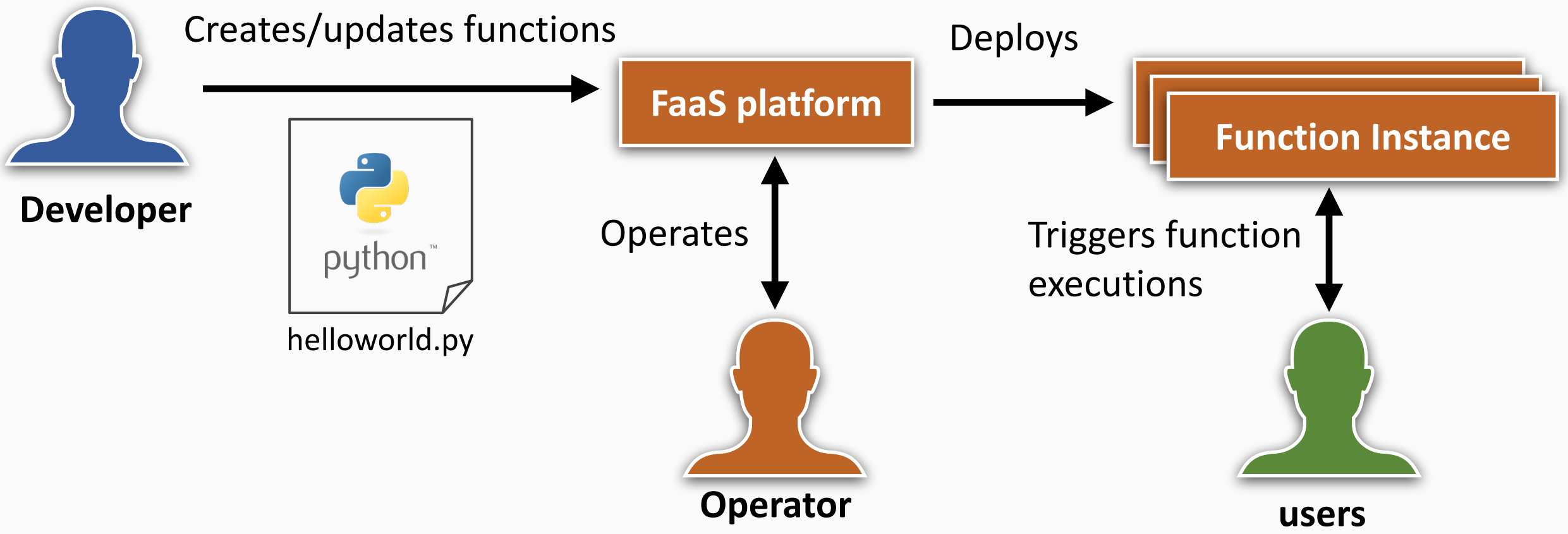
Development Model

High-level abstractions

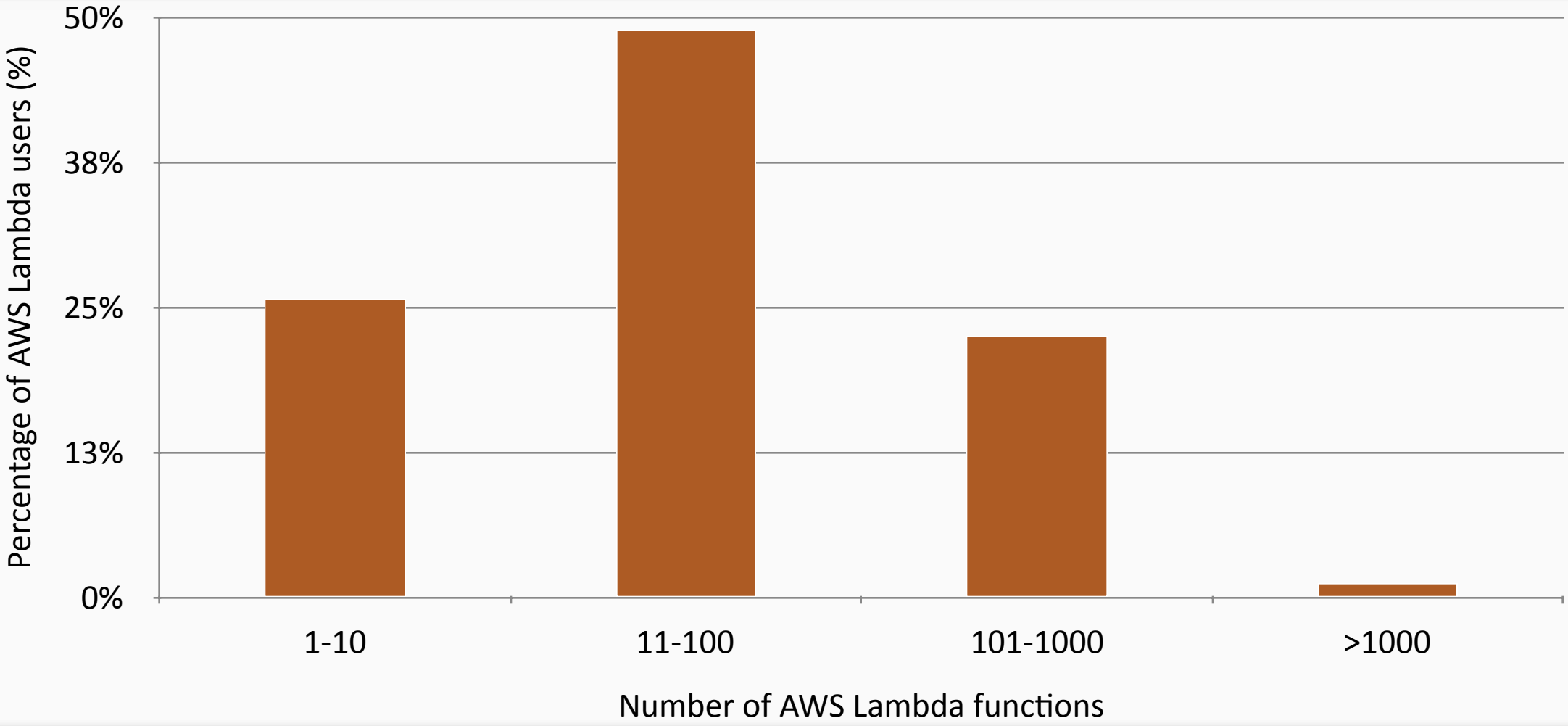
Pre-provided integrations

Language-agnostic

Function-as-a-Service (FaaS) in a nutshell



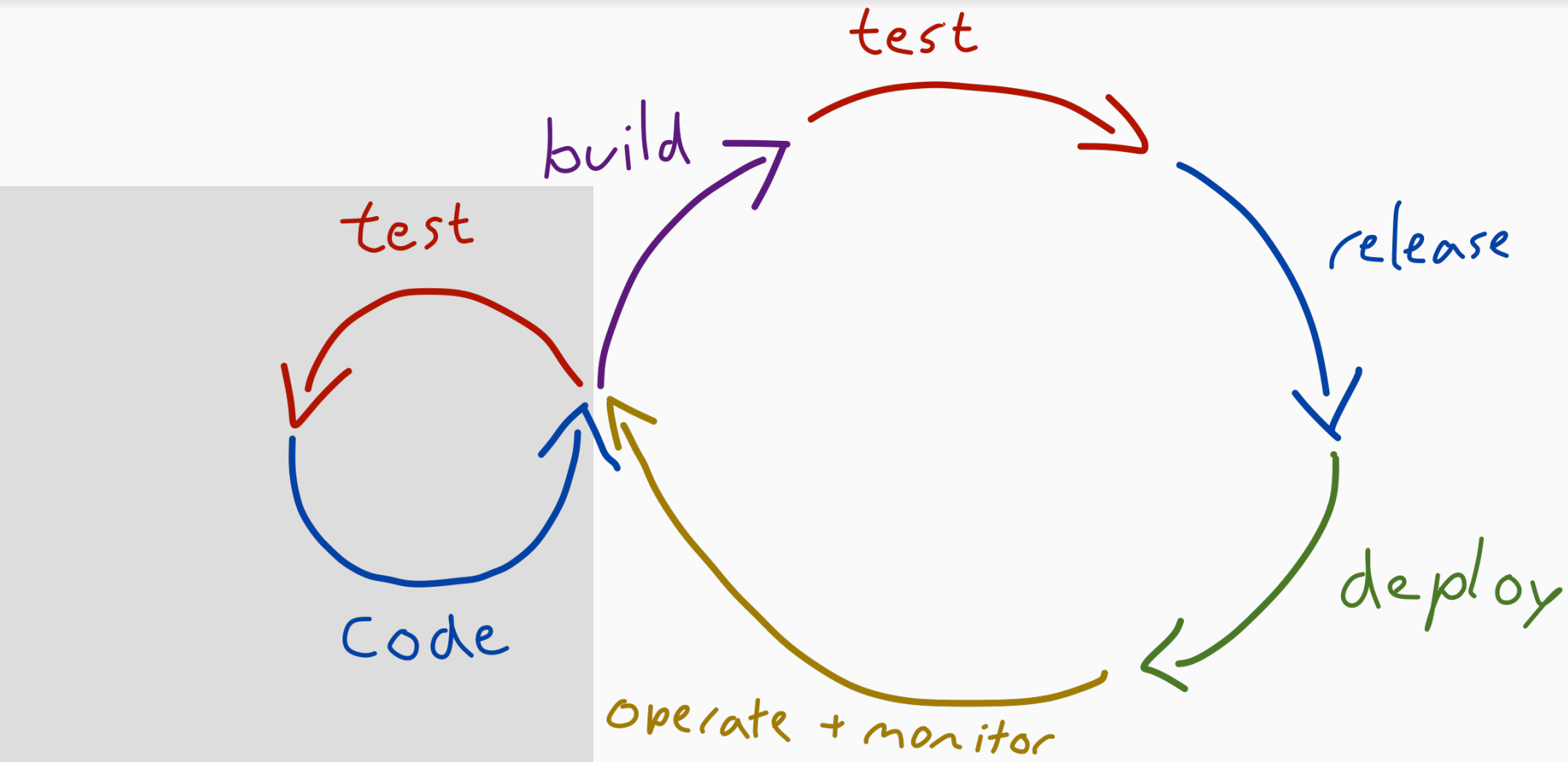
Serverless users use more functions…



…for which we need structured operational processes.

source: https://thenewstack.io/what-aws-lambdas-performance-stats-reveal/

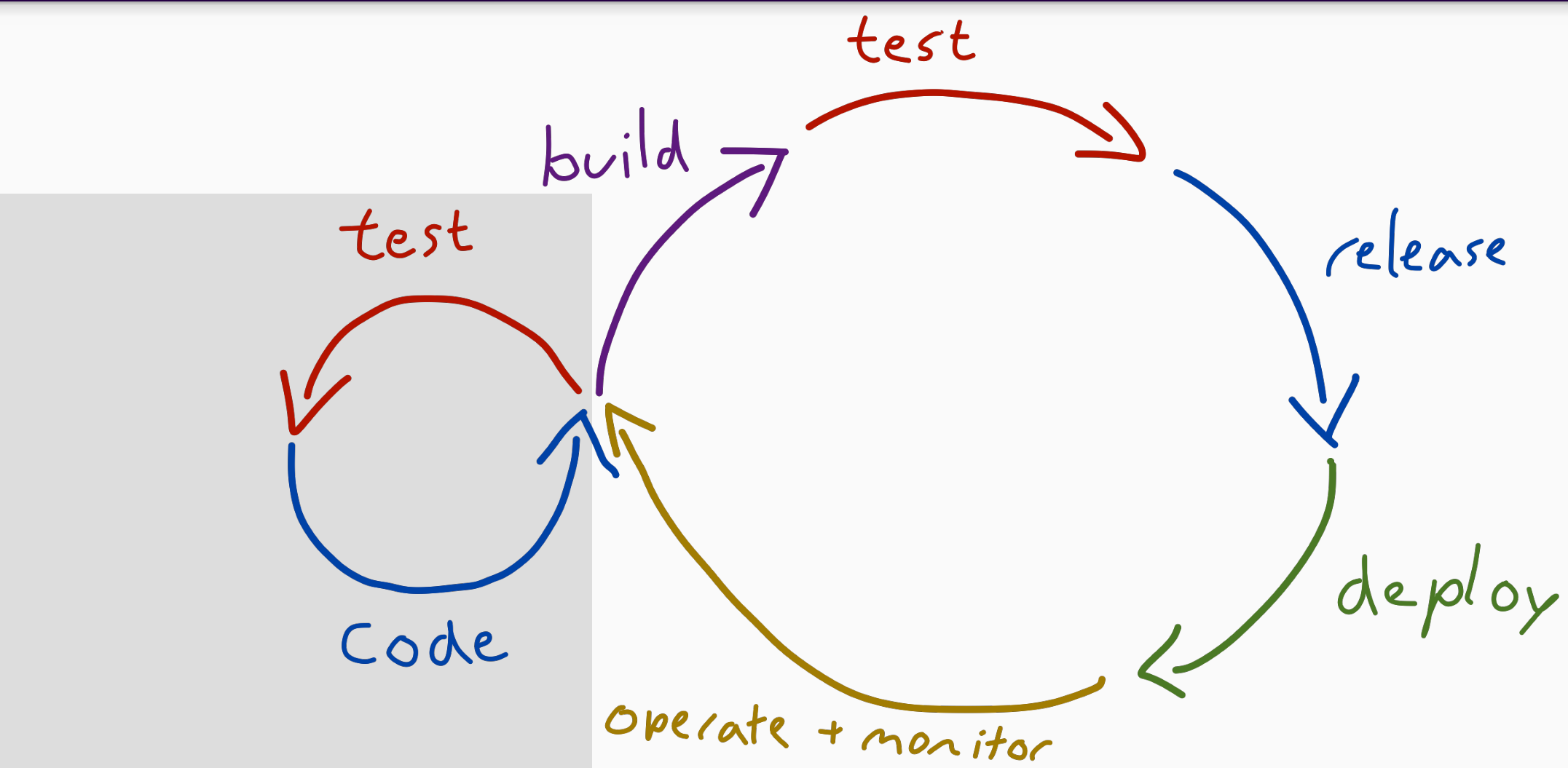
The traditional DevOps lifecycle



Development Loop

Operation Loop

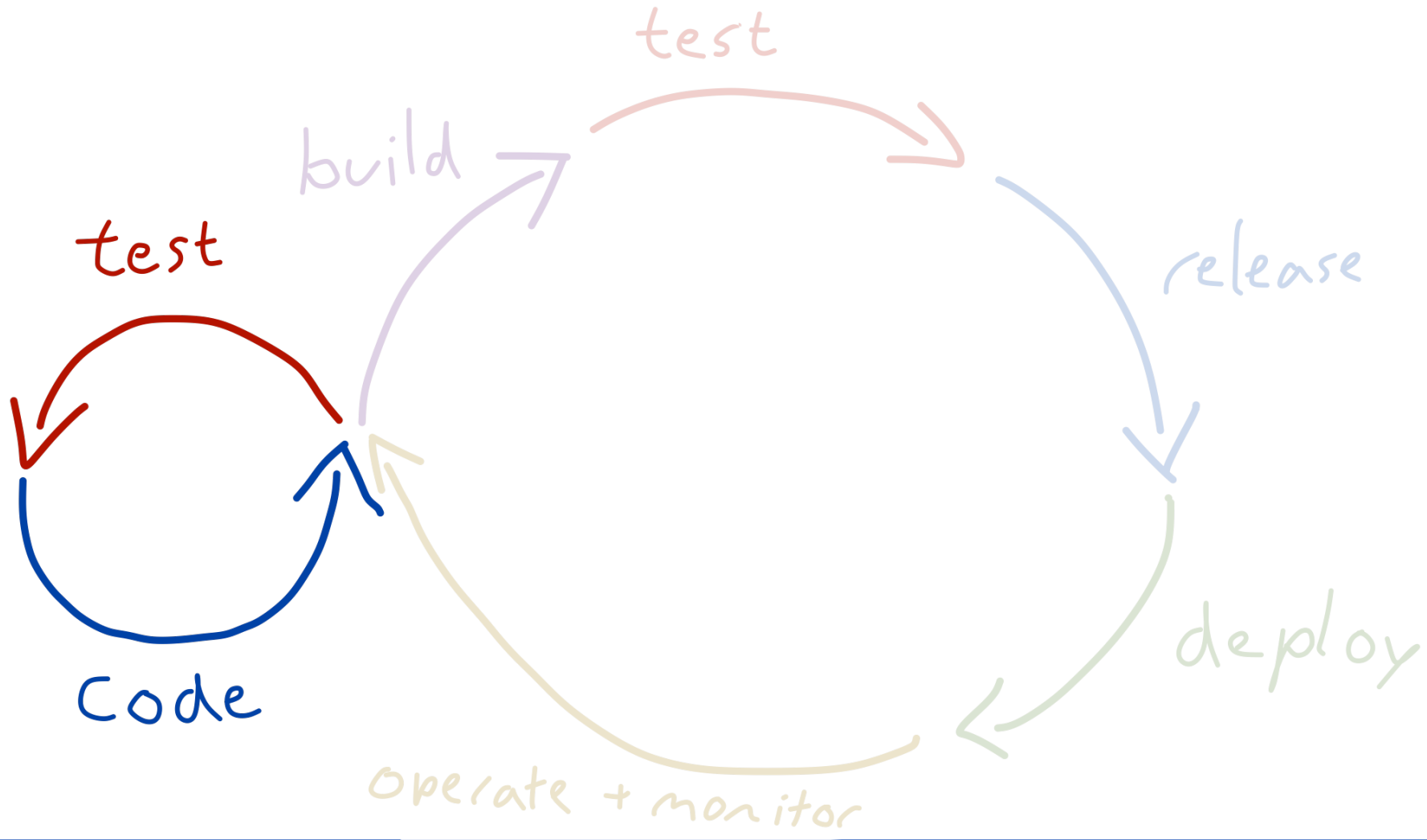
The serverless DevOps lifecycle



What are good practices (specifically) for serverless operations?

Development Loop

code, test.



Keep the development loop fast

Developing and testing within the production-level cloud platform can be slow.

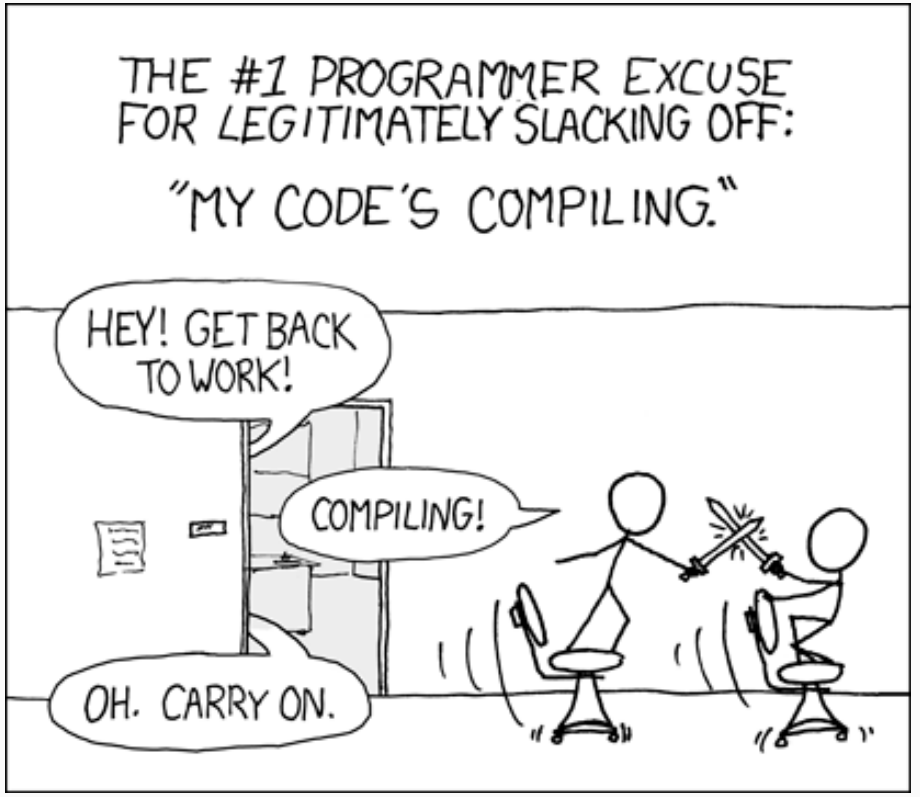
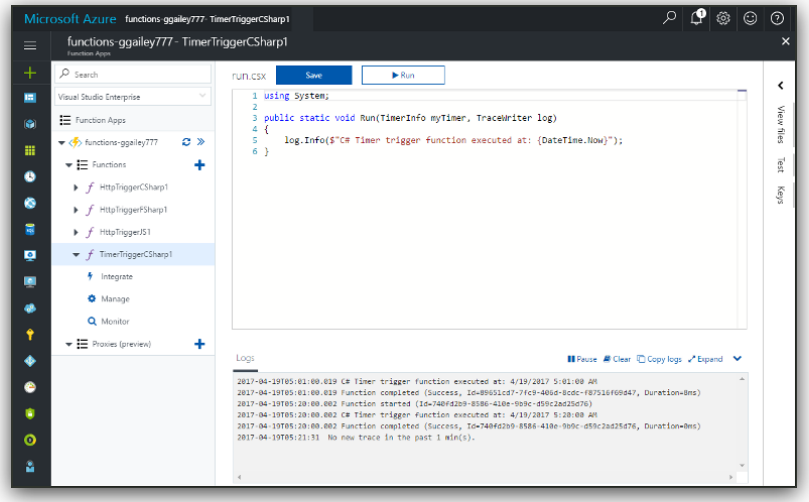
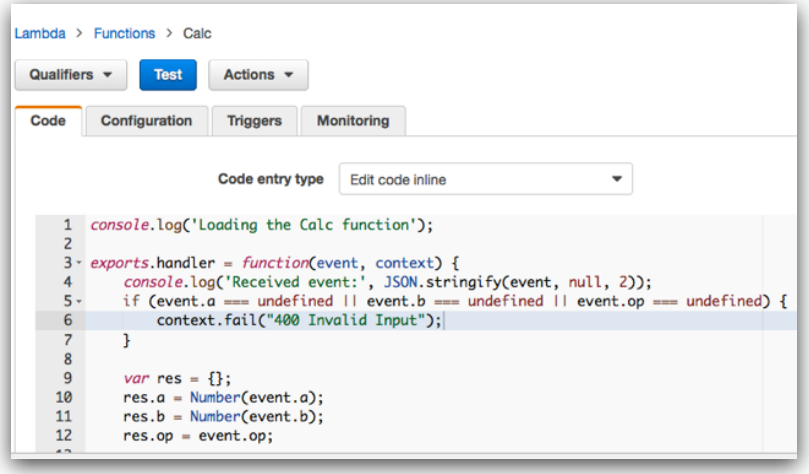


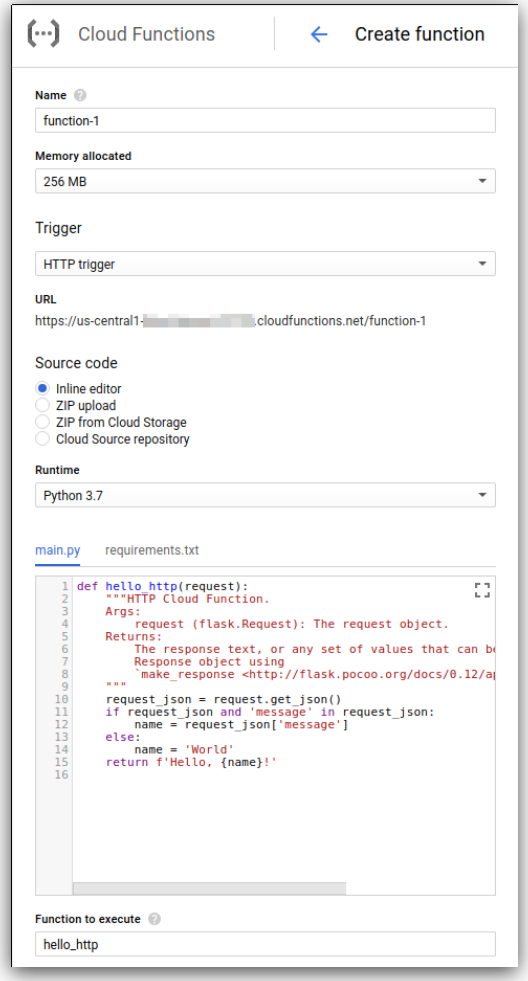
image source: https://xkcd.com/303/

Limitations of online serverless development

- Most cloud providers offer online editors for functions.
- Great for getting started.
- Limitations:
  - Limited functionality
  - Missing typical IDE functionality
  - Minimal version control







Develop locally

Many (third-party) projects aim to improve the local development experience:

- AWS SAM—https://aws.amazon.com/serverless/sam/
- Serverless Framework—https://github.com/serverless/serverless
- Azure Core Tools—https://github.com/Azure/azure-functions-core-tools

Test locally (1): FaaS emulators

Speed up testing by deploying a local FaaS emulator.

Emulators exist for most major FaaS providers:

- Azure Function Core Tools—https://github.com/Azure/azure-functions-core-tools
- Google Cloud Function Emulator—https://cloud.google.com/functions/docs/emulator
- serverless.com emulator—https://github.com/serverless/emulator
- AWS SAM local—https://github.com/awslabs/aws-sam-cli
- Docker Lambda—https://github.com/lambci/docker-lambda

Test locally (2): cloud emulators

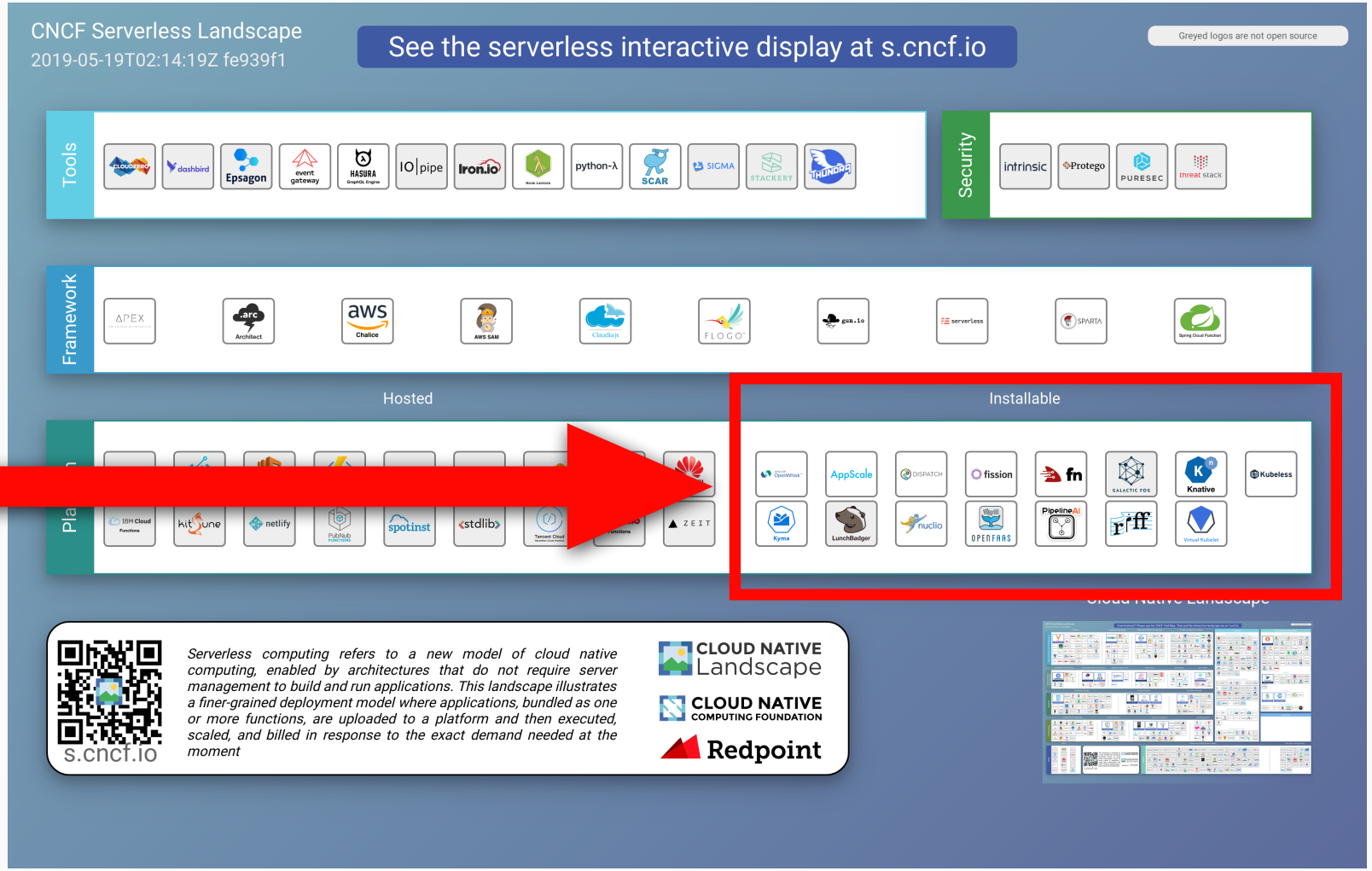
Attempts to emulate FaaS and related services:

- localstack—https://github.com/localstack/localstack
- AWS SAM local—https://github.com/awslabs/aws-sam-cli

⚠️ Can become a time sink to get and keep working.

Test locally (3): open-source FaaS platforms

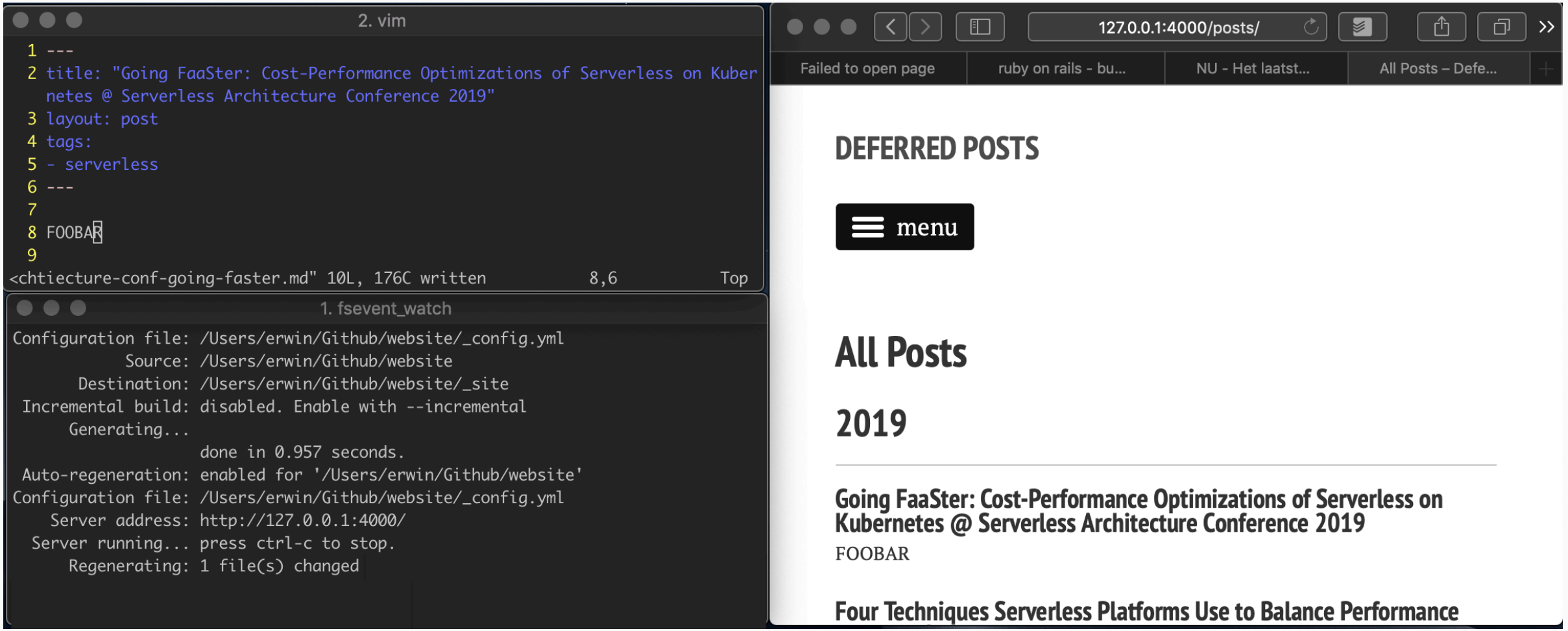
Open-source FaaS platforms allow you to fully replicate the platform locally.



source: https://github.com/cncf/wg-serverless

Live-reloading

Automate function updates to provide instant “frontend”-like feedback.

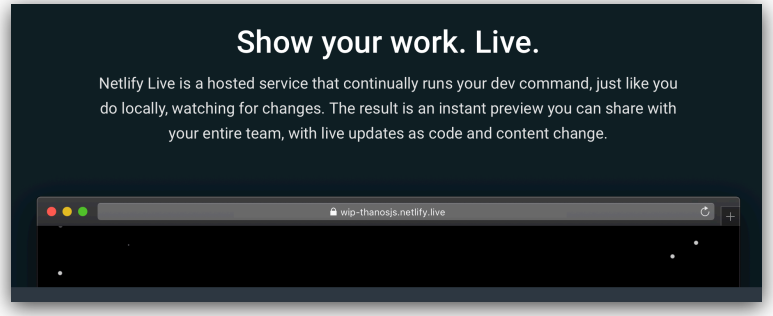


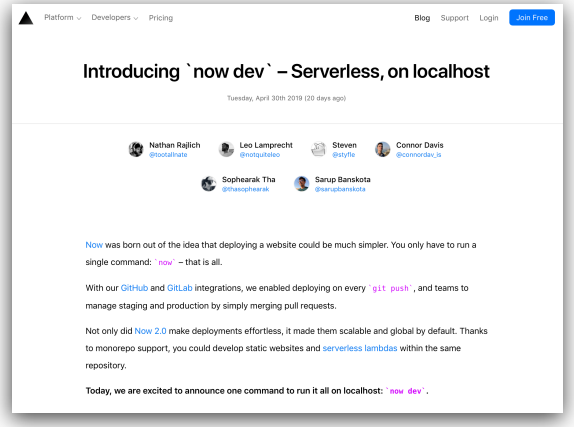
Live-reloading in FaaS platforms

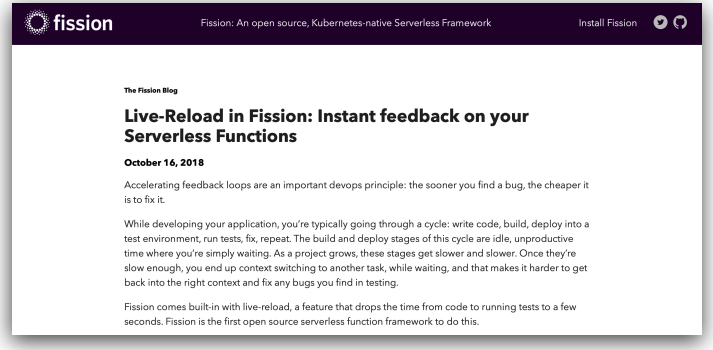
DIY basic live-reloading:

```
fswatch /my/serverless/project | deploy_dev_function.sh
```

Or, built-in:

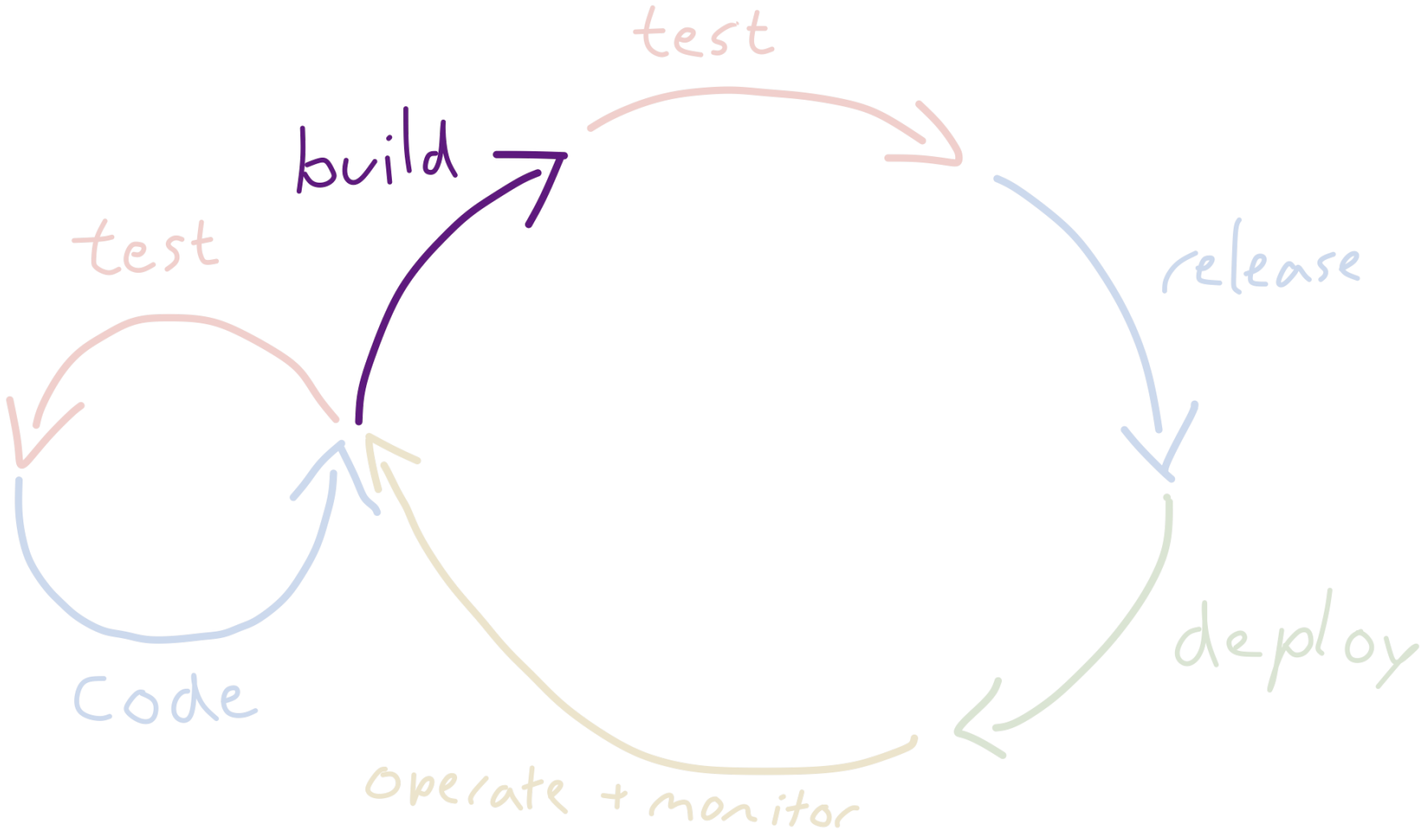






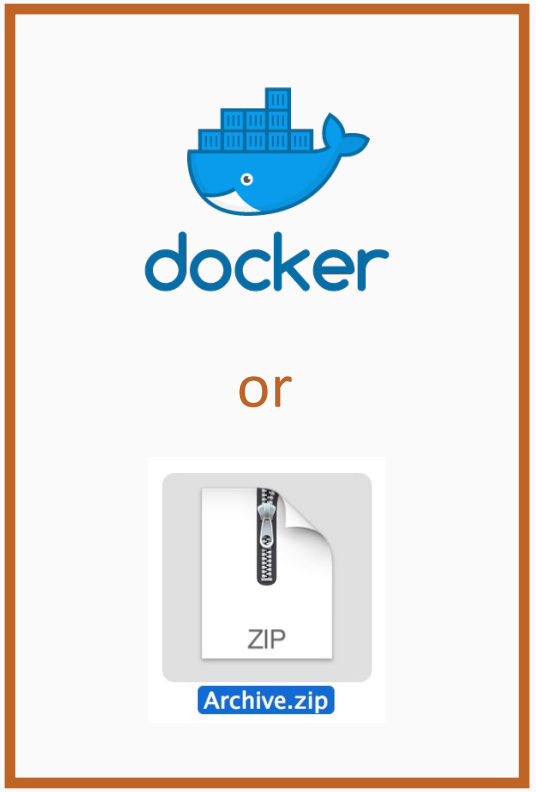
Operation Loop

build.



Automated, reproducible builds

- resolving dependencies
- compiling source code
- packaging assets
- …

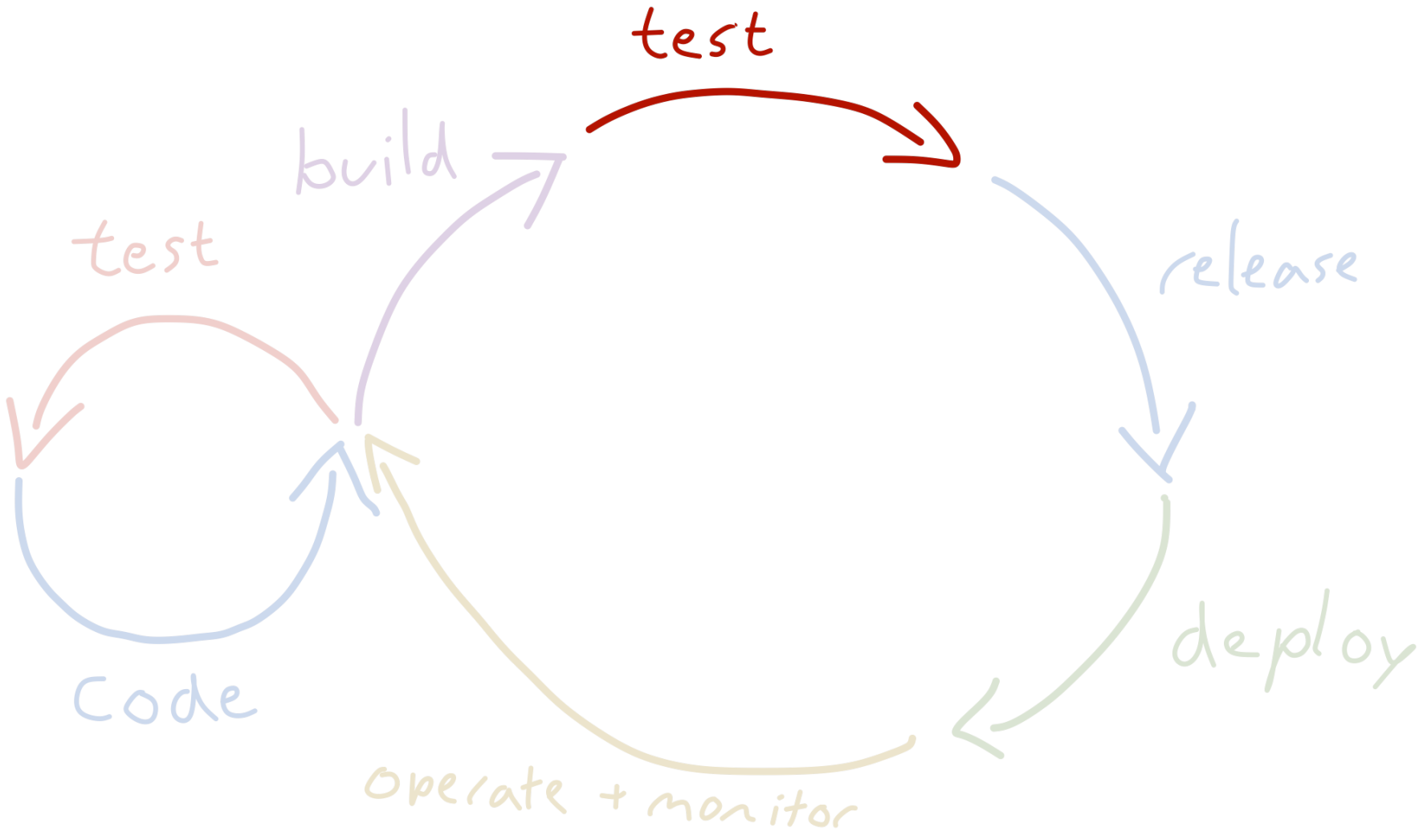


function source(s)

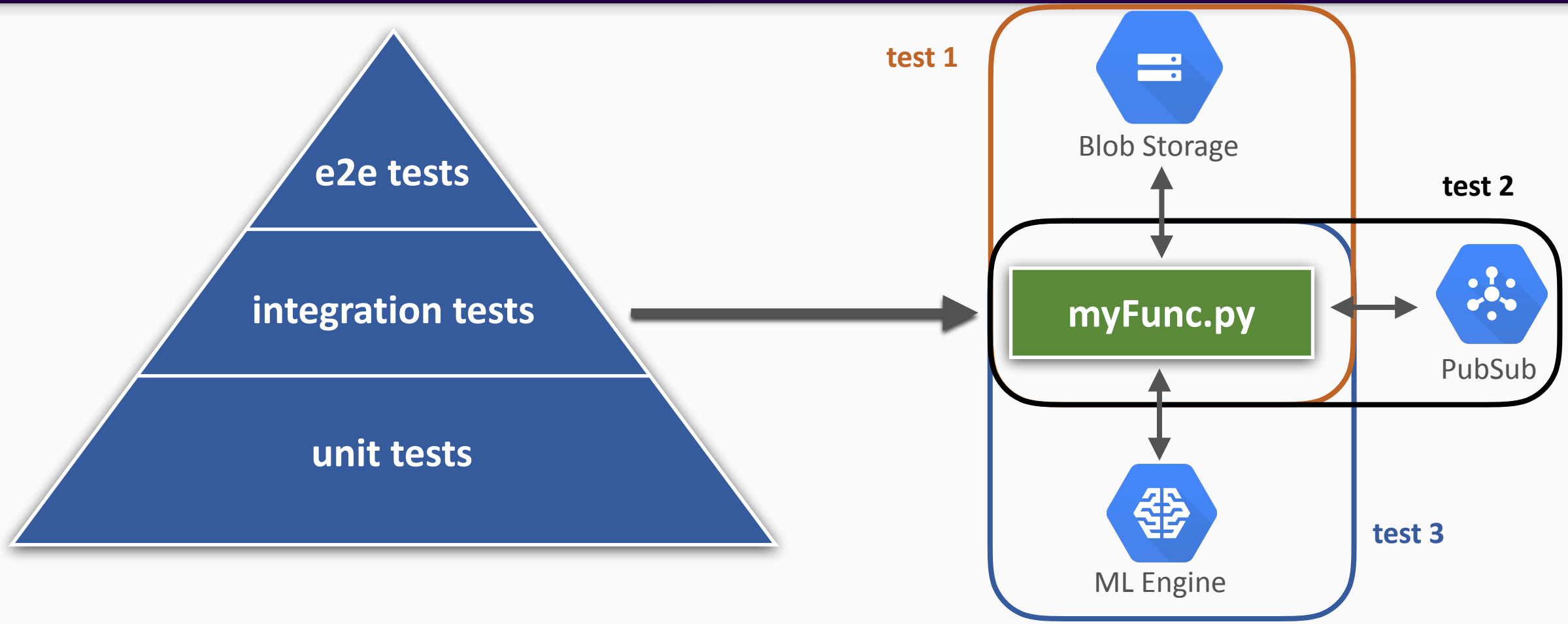
deployable function(s)

Operation Loop

test.



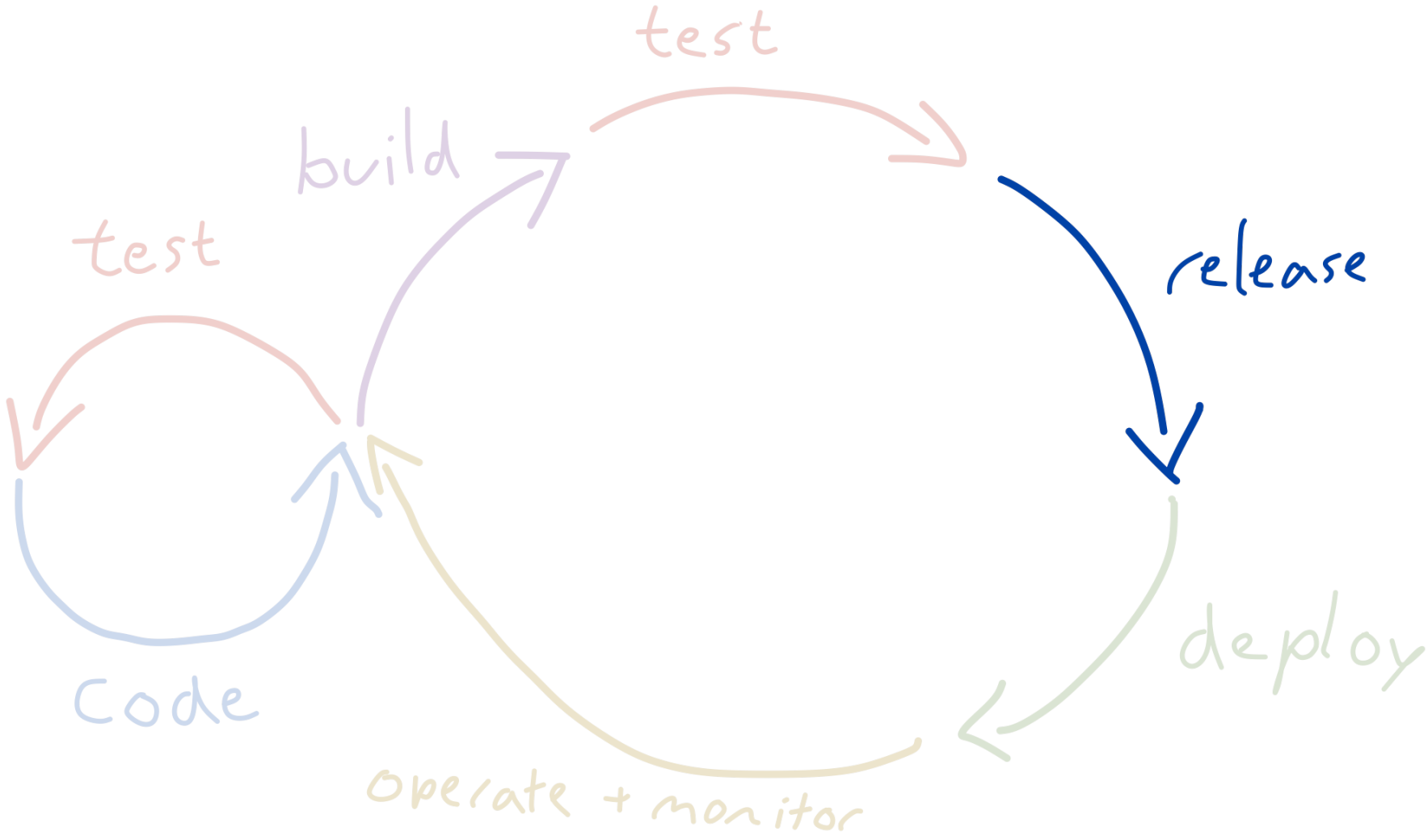
Comprehensive testing in a serverless world



- Avoid hard-coding service-related details.
- Fully isolate the testing from production.
- Set limits and alerts (for run-away functions!).

Operation Loop

release.



GitOps

"Operations by pull request"

Imperative configuration…

deploy.sh

```
# [edit counter.py]
fission environment create --name python --image fission/python-env
fission function create --name counter --env python --src ./counter.py
fission trigger create --name counter-get --method GET --url /counter
```

…vs. declarative configuration

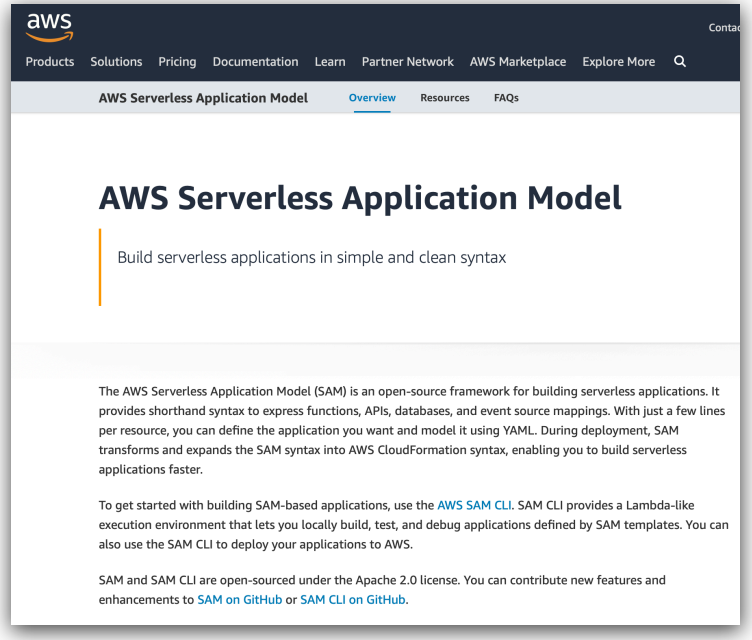
```yaml
apiVersion: fission.io/v1
kind: Environment
metadata:
  name: python
  namespace: default
spec:
  version: 2
  runtime:
    image: fission/python-env:1.1.0
---
apiVersion: fission.io/v1
kind: Function
metadata:
  name: counter
  namespace: default
spec:
  environment:
    name: python
    namespace: default
# [...]
```

deploy.sh

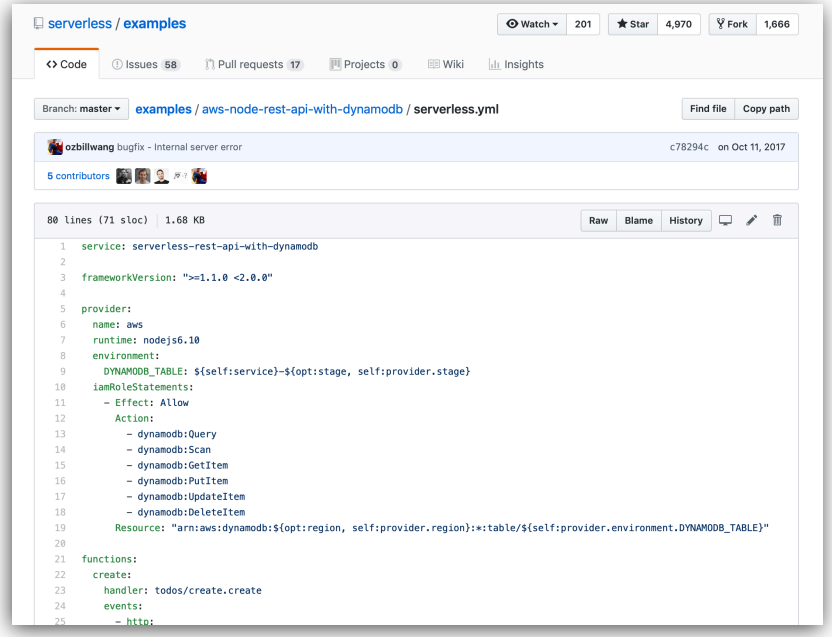
```
# [edit counter.py and counter.yaml]
# create function deployment specs:
kubectl apply -f counter.yaml

# [edit counter.py]
# OR, using the fission API:
fission spec init
fission spec apply # Generates and applies Kubernetes specs
```

Examples of projects focusing on declarative configurations



AWS SAM



Serverless framework

Kubeless

APACHE OpenWhisk™

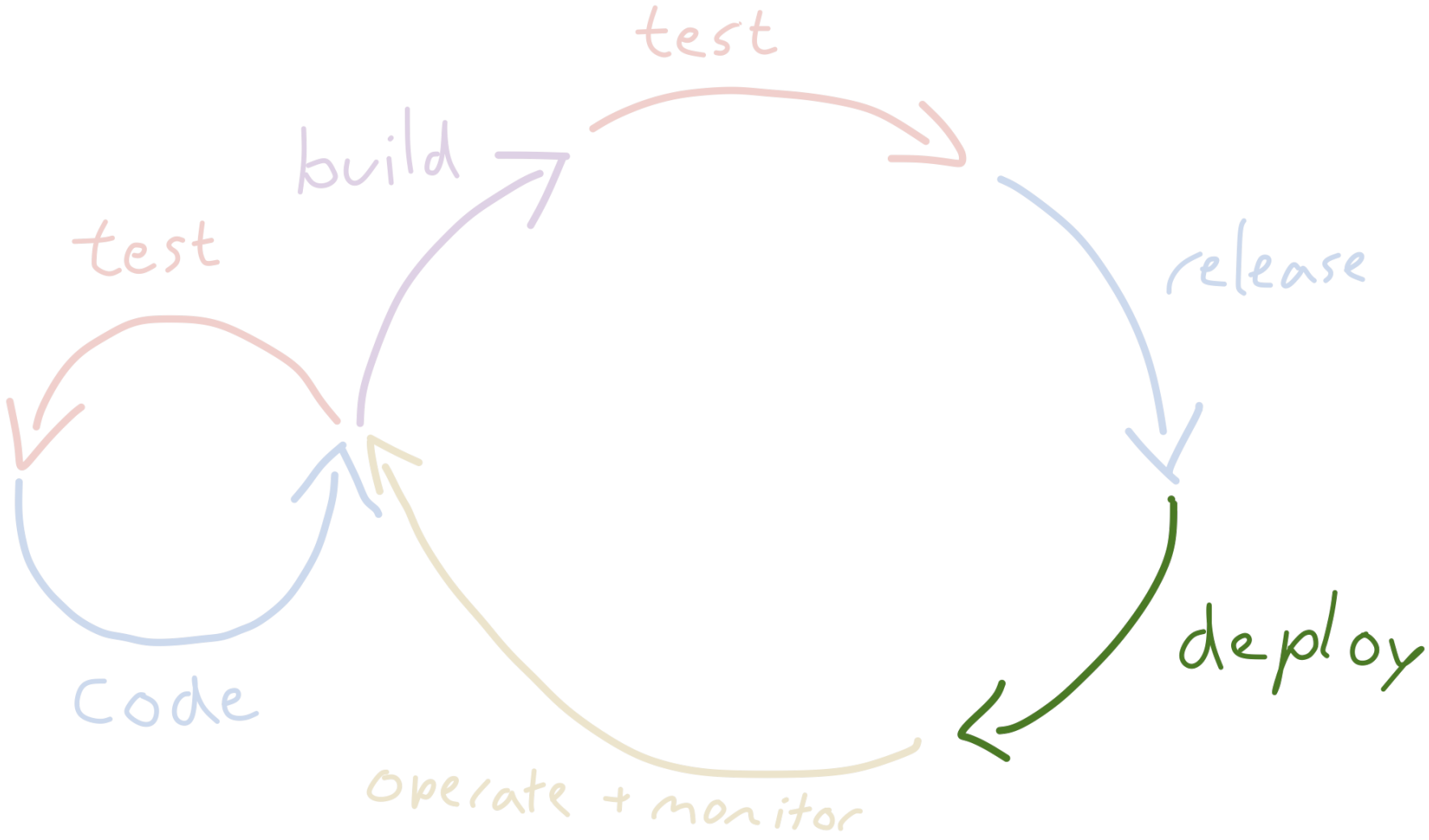
fission

OPENFAAS

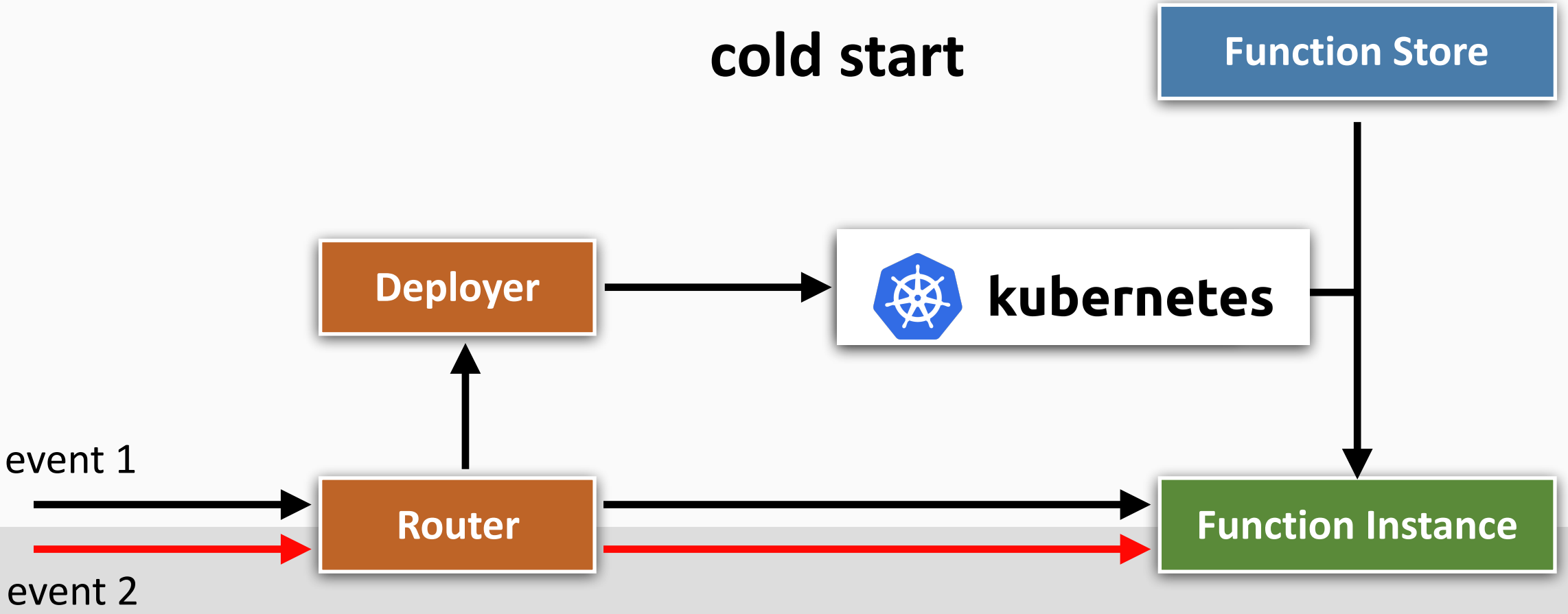
Most open-source FaaS platforms

Operation Loop

deploy.



Understanding the FaaS deployment process



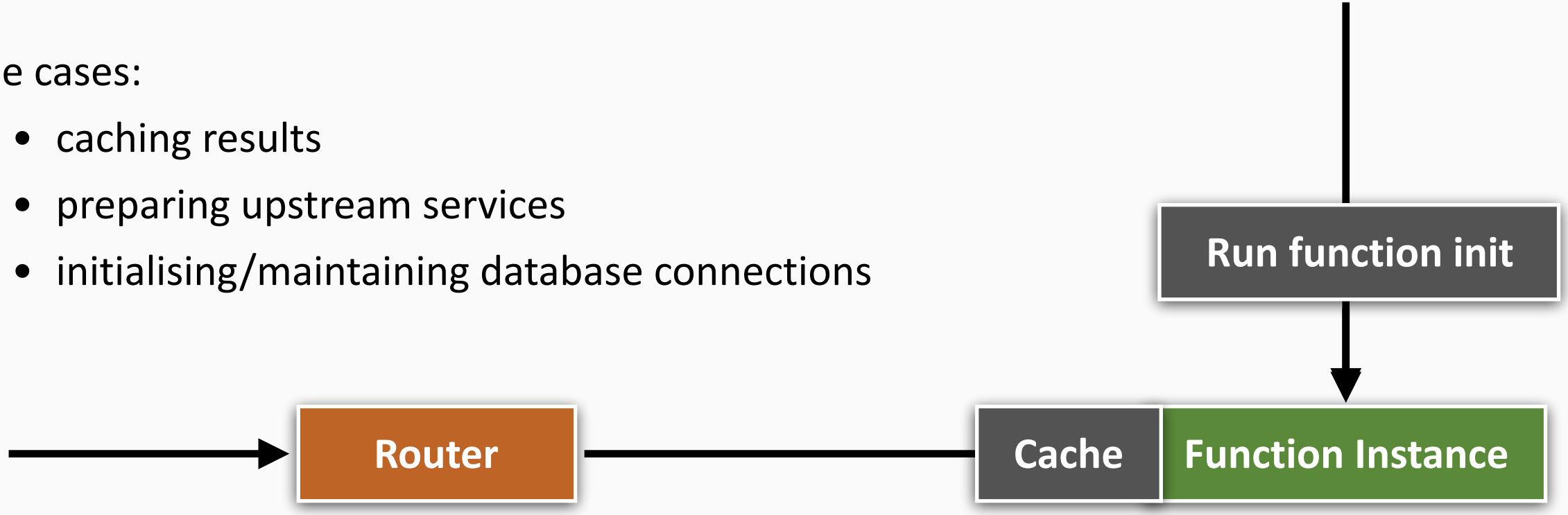
warm execution

Caching in deployed function instances

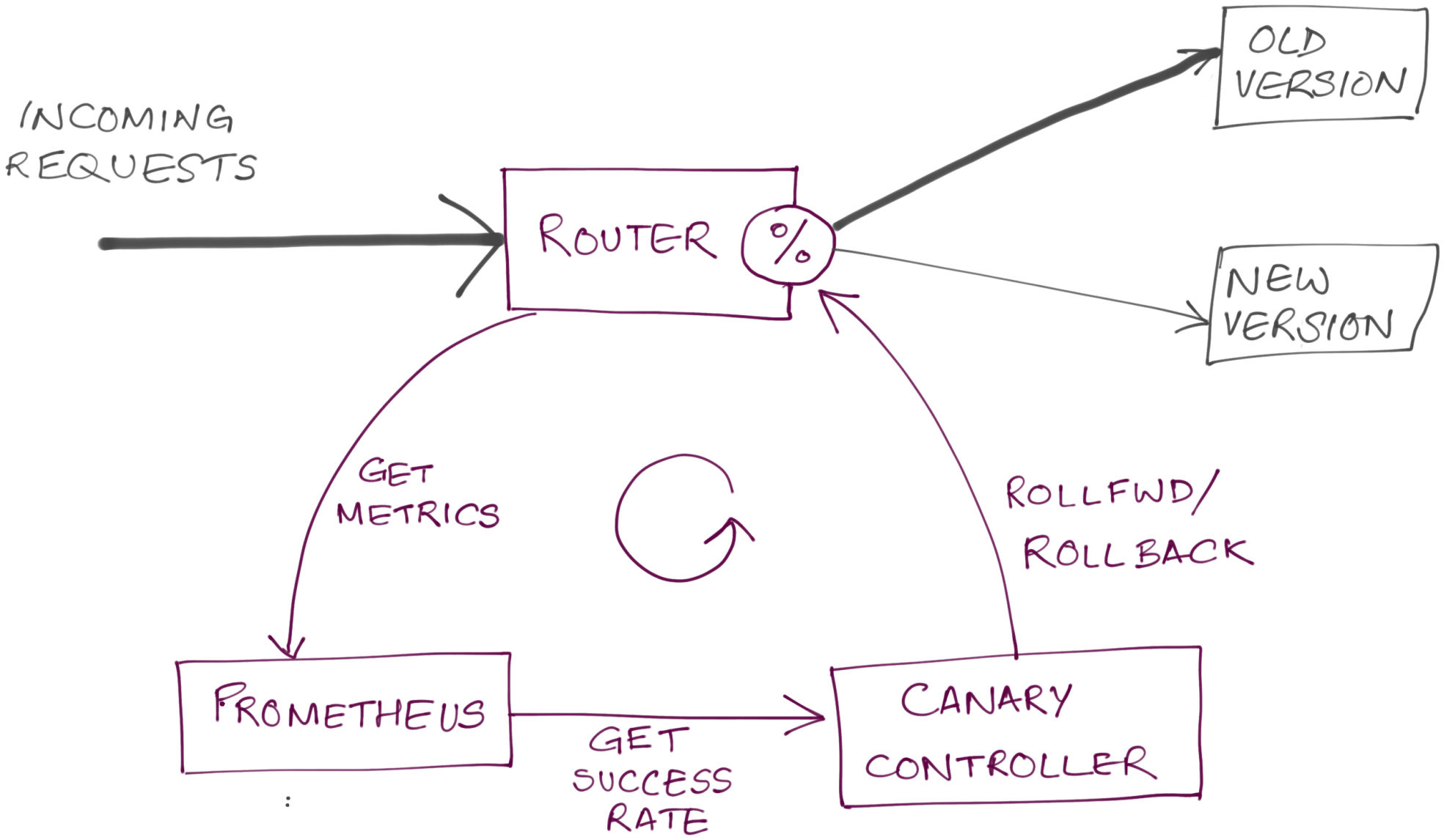
Functions are stateless, but can maintain non-persistent state.

Use cases:

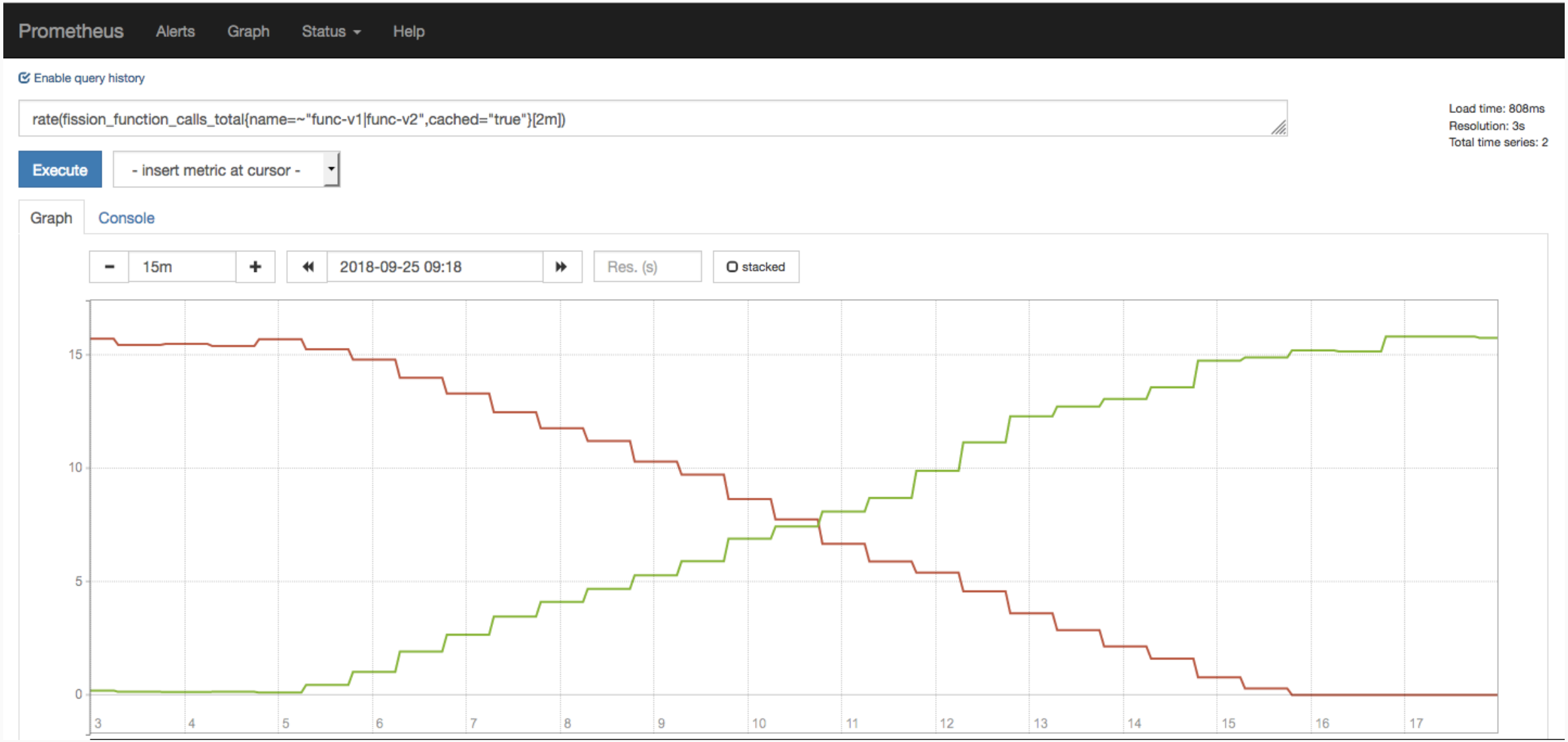
- caching results
- preparing upstream services
- initialising/maintaining database connections



Canary Deployments

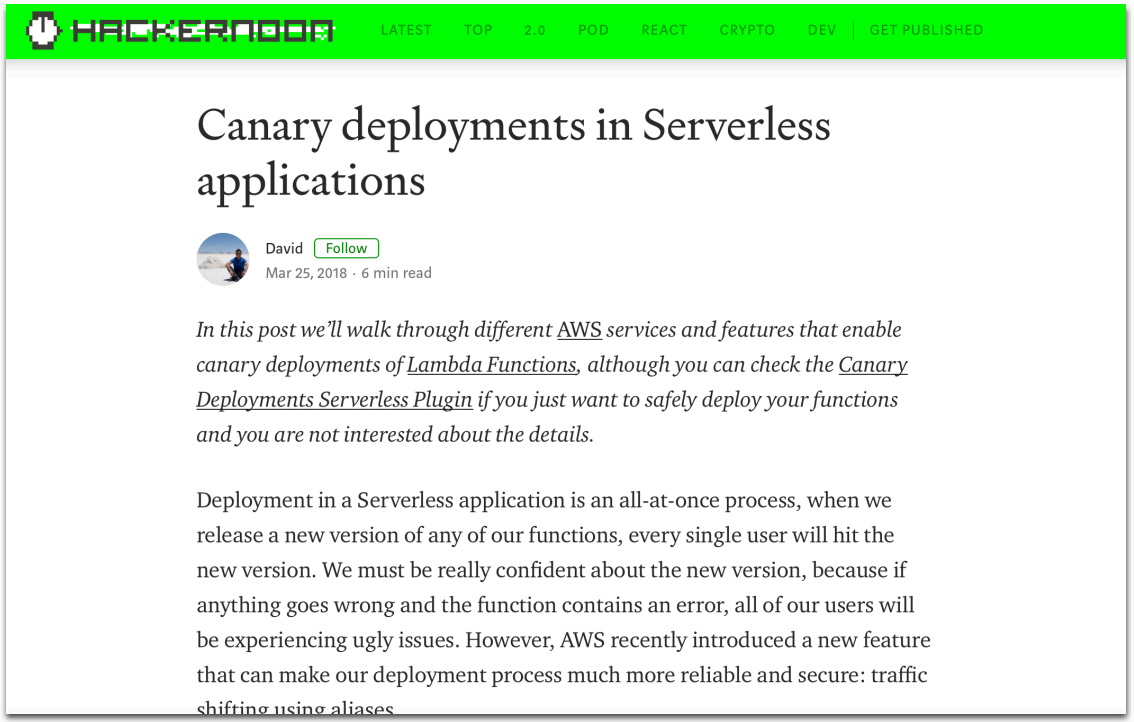


Canary Deployments (2)

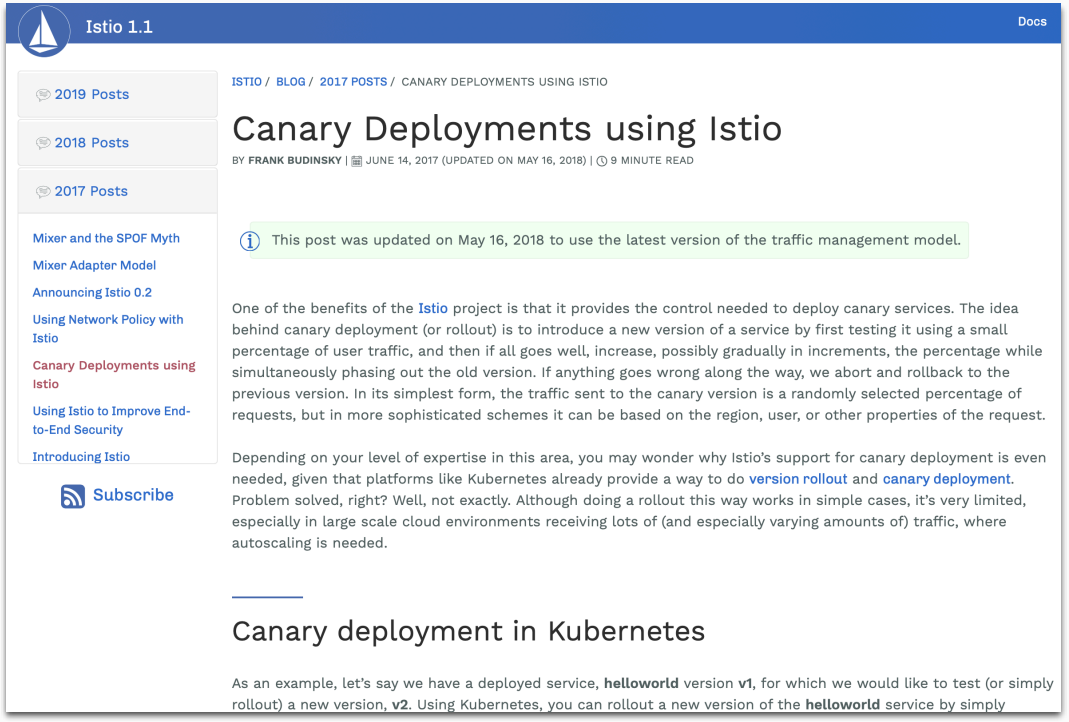


https://blog.fission.io/posts/automated-canary-deployments/

Canary Deployments (3)



AWS CodeDeploy + AWS Lambda

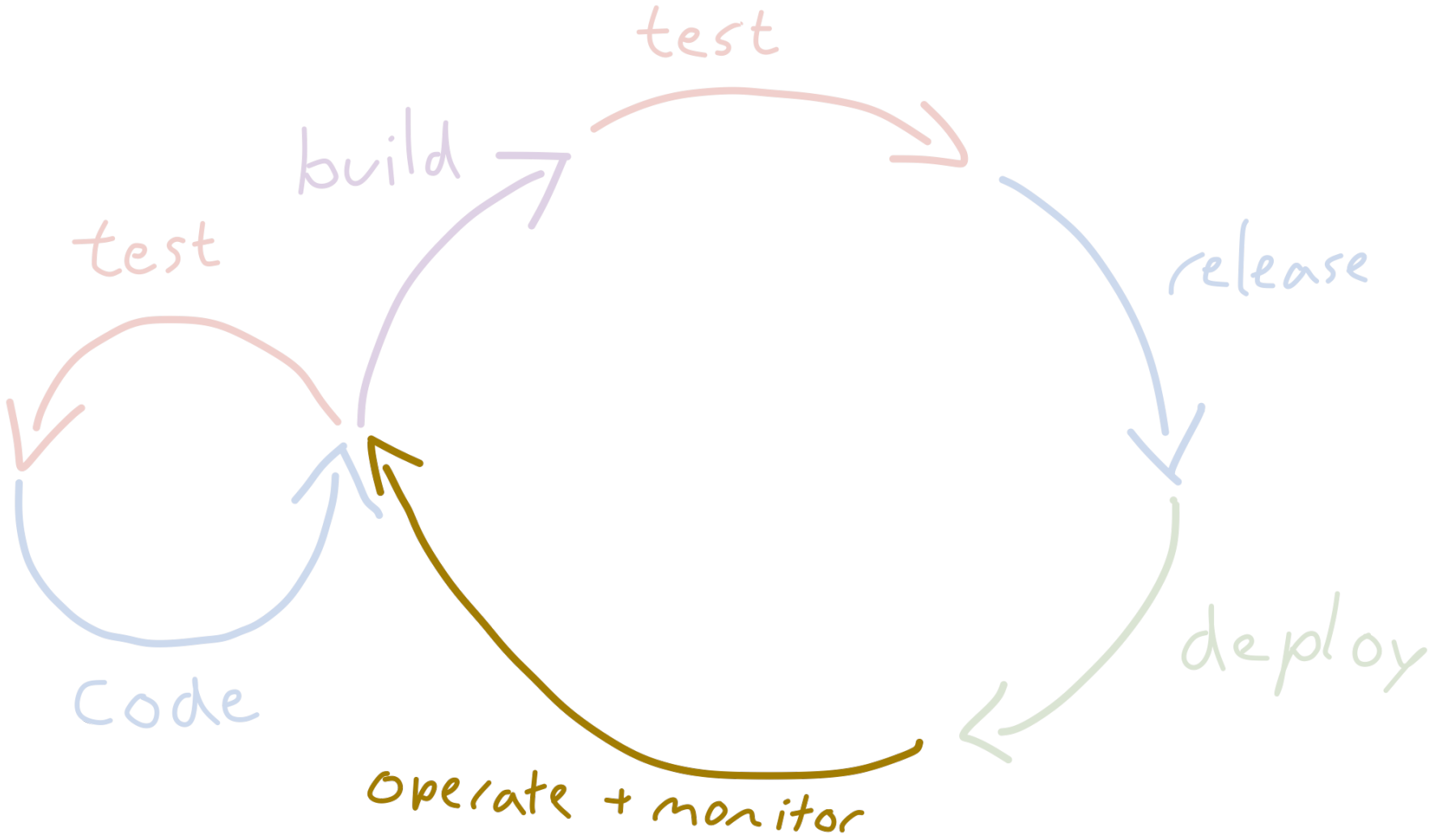


Service mesh-based FaaS platforms (e.g., knative)

https://hackernoon.com/canary-deployments-in-serverless-applications-b0f47fa9b409
https://istio.io/blog/2017/0.1-canary/

Operation Loop

operate, monitor.



Monitoring, tracing, logging

Serverless applications still need to be monitored

Serverless platforms help with…

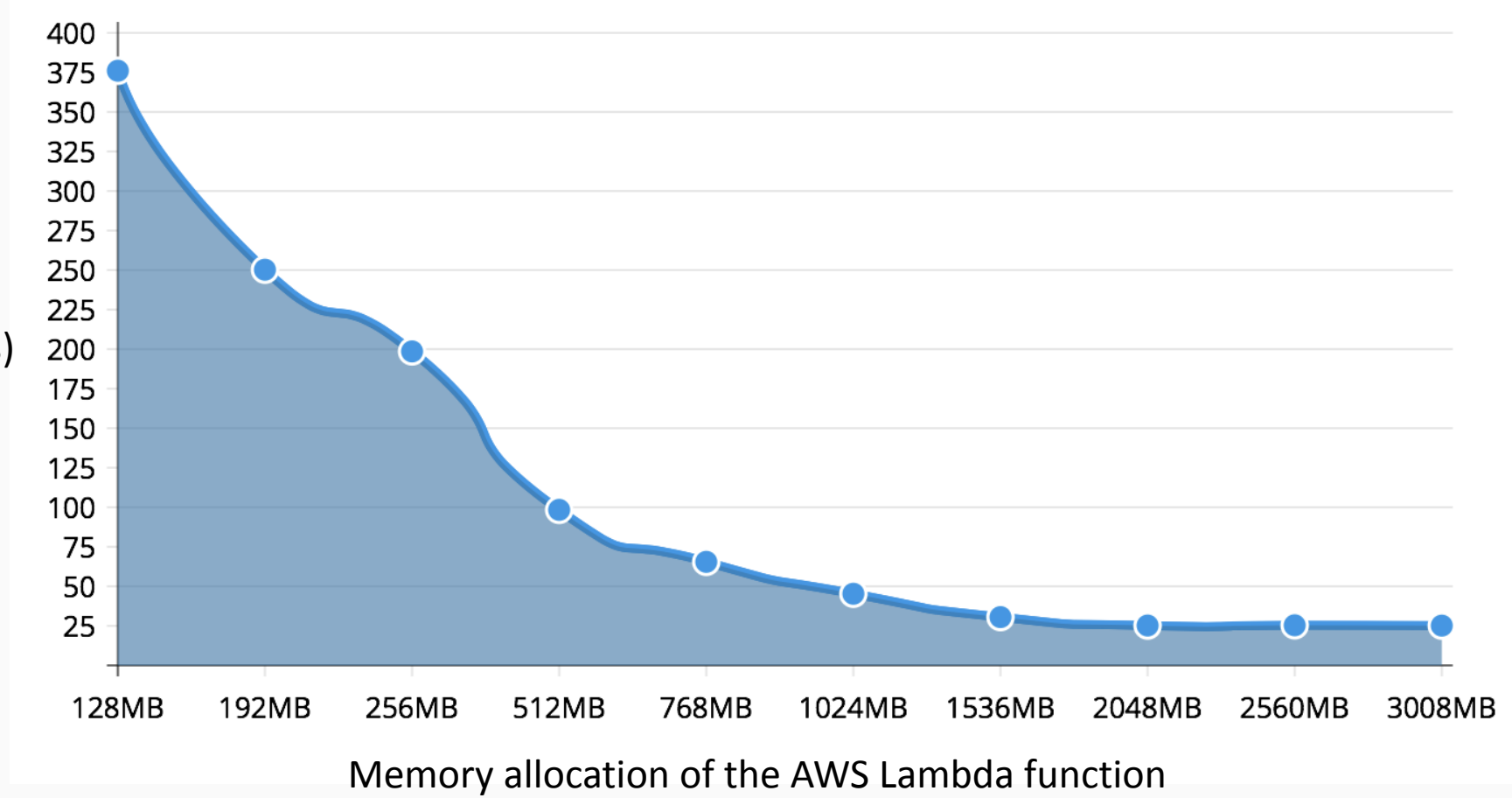
- metric monitoring (system-level and some user-level metrics)
- log aggregation
- distributed tracing

Potential pitfalls:

- (implicit) costs
- proprietary formats
- vendor lock-in

Memory allocation is linked to CPU/IO shares

Average runtime calculating Fibonacci numbers 1 to 100



source: https://epsagon.com/blog/how-to-make-lambda-faster-memory-performance-benchmark/

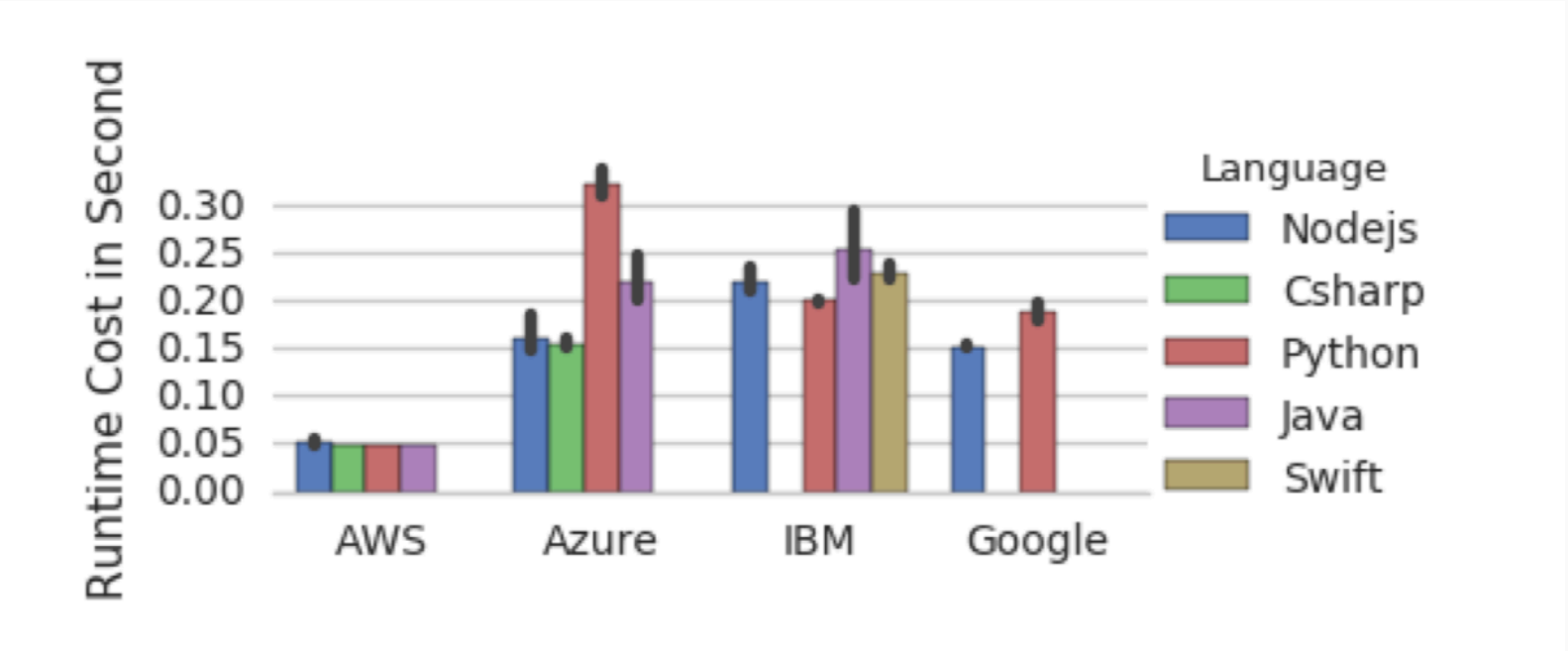
Never scaling down to zero

Keep function instances alive at all times to avoid (most) cold starts.

- Not well supported in managed FaaS platforms.
  - Azure supports minimal instances in its “premium plan”.
- Well supported in open-source FaaS platforms.

The language can impact performance

- In theory, languages should perform similar.
- In practice, maturity of languages differs.



Lee, Hyungro, Kumar Satyam, and Geoffrey Fox. "Evaluation of production serverless computing environments." *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 2018.

Conclusion

Lessons learned:

1. Keep the serverless development loop as fast as possible.
2. Prefer declarative over imperative configuration.
3. Automate the build and deployment process (canaries!).
4. Use the function deployment process to your advantage.
5. Be wary of performance effects of function configuration changes.

Takeaway: The DevOps lifecycle is (still) relevant in a serverless world!

Thanks!

fission	http://fission.io + https://github.com/fission
Slack	http://slack.fission.io/
Twitter	@fissionio

Erwin van Eyk
Software Engineer @ Platform9 Systems
Chair @ SPEC CLOUD RG Serverless
Researcher @ AtLarge Research

@erwinvaneyk
erwin@platform9.com
https://erwinvaneyk.nl

Further reading

- More on **FaaS internals and performance**:
  - My talk at KubeCon China 2018: https://erwinvaneyk.nl/kubecon-china-2018-serverless-performance/
  - Or, the blogpost derived from that talk: https://www.infoq.com/articles/serverless-performance-cost
  - Wang, Liang, et al. "Peeking behind the curtains of serverless platforms." 2018. https://www.usenix.org/system/files/conference/atc18/atc18-wang-liang.pdf

- More on **serverless concepts:**
  - Serverless is More (2018) - which covers the emergence, current state, and future of serverless: https://erwinvaneyk.nl/internet-computing-serverless-is-more/
  - CNCF Serverless WG - Serverless Overview Whitepaper and serverless landscape (2017): https://github.com/cncf/wg-serverless

Additional Slides

Who is who in serverless computing

Cloud user

- Or just the "**(software) developer**" in the context of this talk.
- Uses cloud (and serverless) services to develop applications.
- **Examples:** Spotify, Netflix, you(?)…

Cloud operator

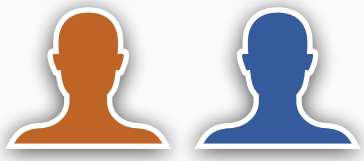
- Provides cloud (and serverless) services to cloud users.
- Can be a public or private, in-house cloud provider.
- **Examples:** AWS, Platform9, internal DevOps team…

Application user

- Generates events which trigger the execution of (cloud) applications.
- Can be downstream services, or actual (physical) users.
- **Examples:** frontend/UI, workqueue…

Serverless in a nutshell

cloud operator manages…



cloud user (you) manages…

bare-metal	IaaS	kubernetes	serverless
Application	Application	Application	Application
Application Middleware	Application Middleware	Application Middleware	Application Middleware
Cluster Resource Mgt	Cluster Resource Mgt	Cluster Resource Mgt	Cluster Resource Mgt
Virtualization	Virtualization	Virtualization	Virtualization
Hardware	Hardware	Hardware	Hardware