A Million Ways to Fail in Production

Embracing Catastrophes for Fun and Profit

Jonatan Ivanov
2025-02-28

About Me

- Spring Team
  - Micrometer
  - Spring Cloud, Spring Boot
  - Spring Observability Team
- Java Champion
- Seattle Java User Group
- develotters.com
- **@jonatan_ivanov**

What can go wrong? 🙃

What can go wrong?

- You get an error (something that contains an error)
- Or nothing (timeout, connection reset, TLS, etc.)
- Latency
- Fallacies of distributed computing
- Network, Hardware, Apps, DBs, Caches, Streams, Queues, etc.

Why do we care?

- We need to fix them
- Increasing complexity
- Something is always broken on huge scale
- Chaotic environments
- Unknown unknowns
- Things can be perceived differently by observers

What can we do about it?

Observability: Data about your components

- Logging: What happened? (Why?)
- Metrics: What’s the context? How bad is it?
- Distributed Tracing: Why did it happen?
- But there is so much more…

Memory leak only in PROD

Memory leak only in PROD

- The platform/container was homegrown 😬
- Had memory leak on Java 5 🧨
- It was fine on Java 6 😎 … … 🧨
- We deployed new features (automated) 🚚
- Also updated Java: 6u123 ➜ 6u124 ☕
- The app was leaking! 😱
- But only in prod! 🧐
- Investigations for weeks! 🤔

What happened?

- I “accidentally” ran **`java -version`**
- It said Java 5 ...
- The 6u124 folder contained Java 5
- The deployment script used that
- The app was running on Java 5
- And leaking ...

What can we learn from it?

- Do not write platforms/containers on your own
- Do not 100% trust your deployment pipeline
- Only your app knows its environment
  - Java version, vendor, ...
  - OS version, arch, ...
  - ENV vars, properties, ...

Solution: Ask the app!

```
"java": {
   "version": "23",
   "vendor": {
      "name": "BellSoft"
   },
   "jvm": {
      "name": "OpenJDK 64-Bit Server VM",
      "vendor": "BellSoft",
      "version": "23+38"
   }
}
```

Memory leak or not?

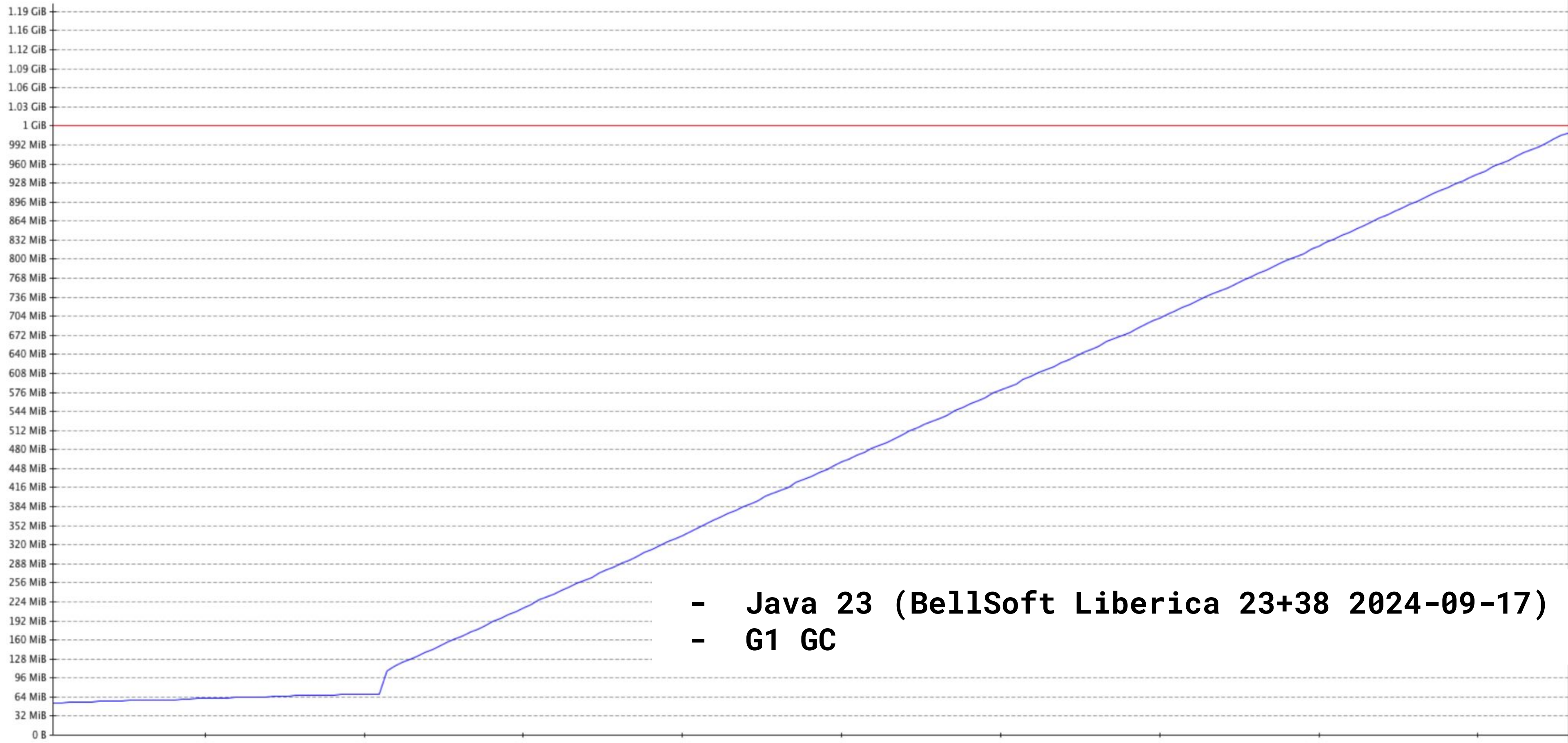
Memory leak or not?

- We deployed new features (manual steps) 🚚
- One instance did not receive any traffic 😬
- No one noticed it 🙈
- The app had a scheduled job (infrequent) 🕒
- An alert was triggered! 🧨
- The heap utilization was high (95+%)! 😱
- Only on the “no-traffic” instance! 🧐
- Investigations, load tests…

What happened?

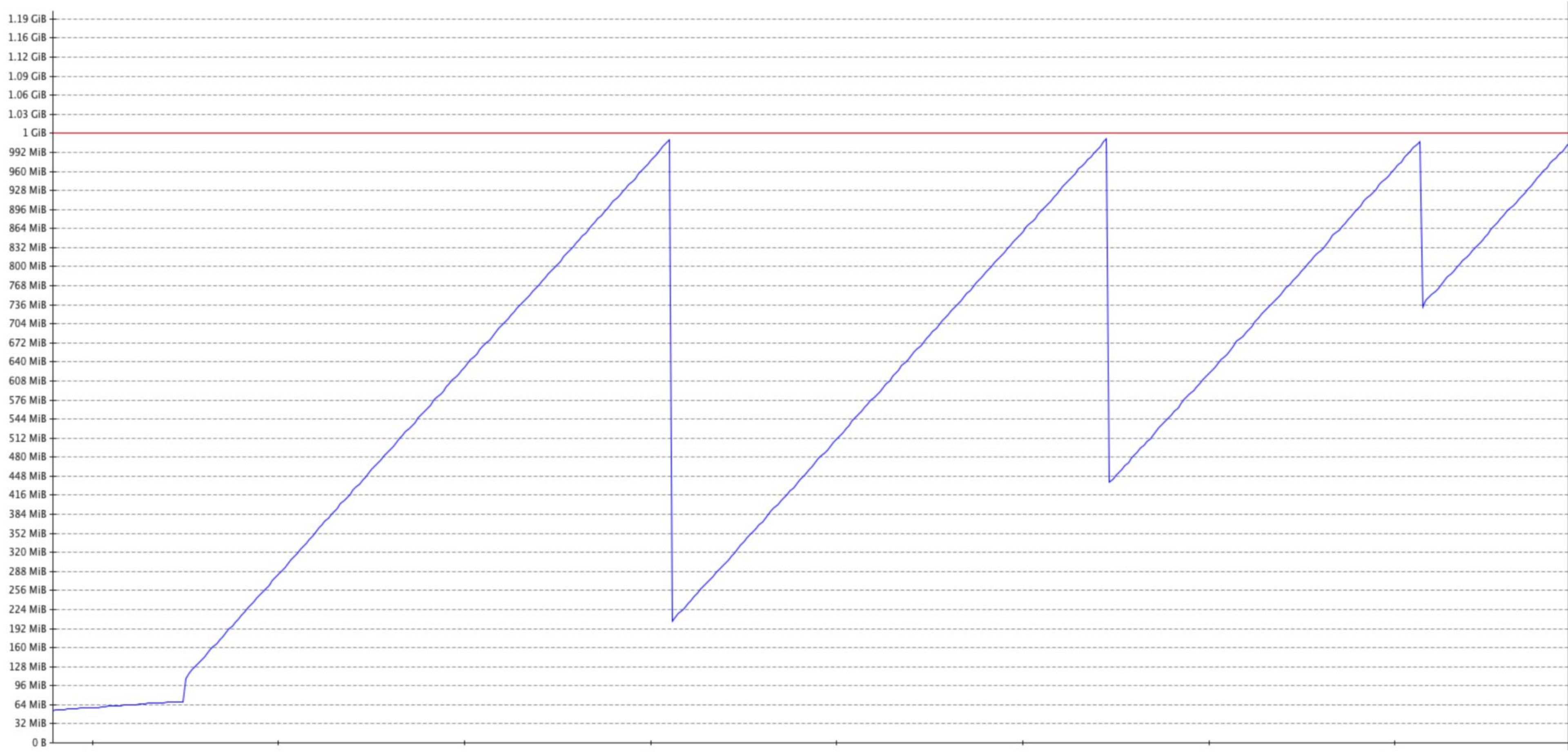
- A Generational GC was used…
- Let’s play “Memory leak or not”…

Memory leak or not?

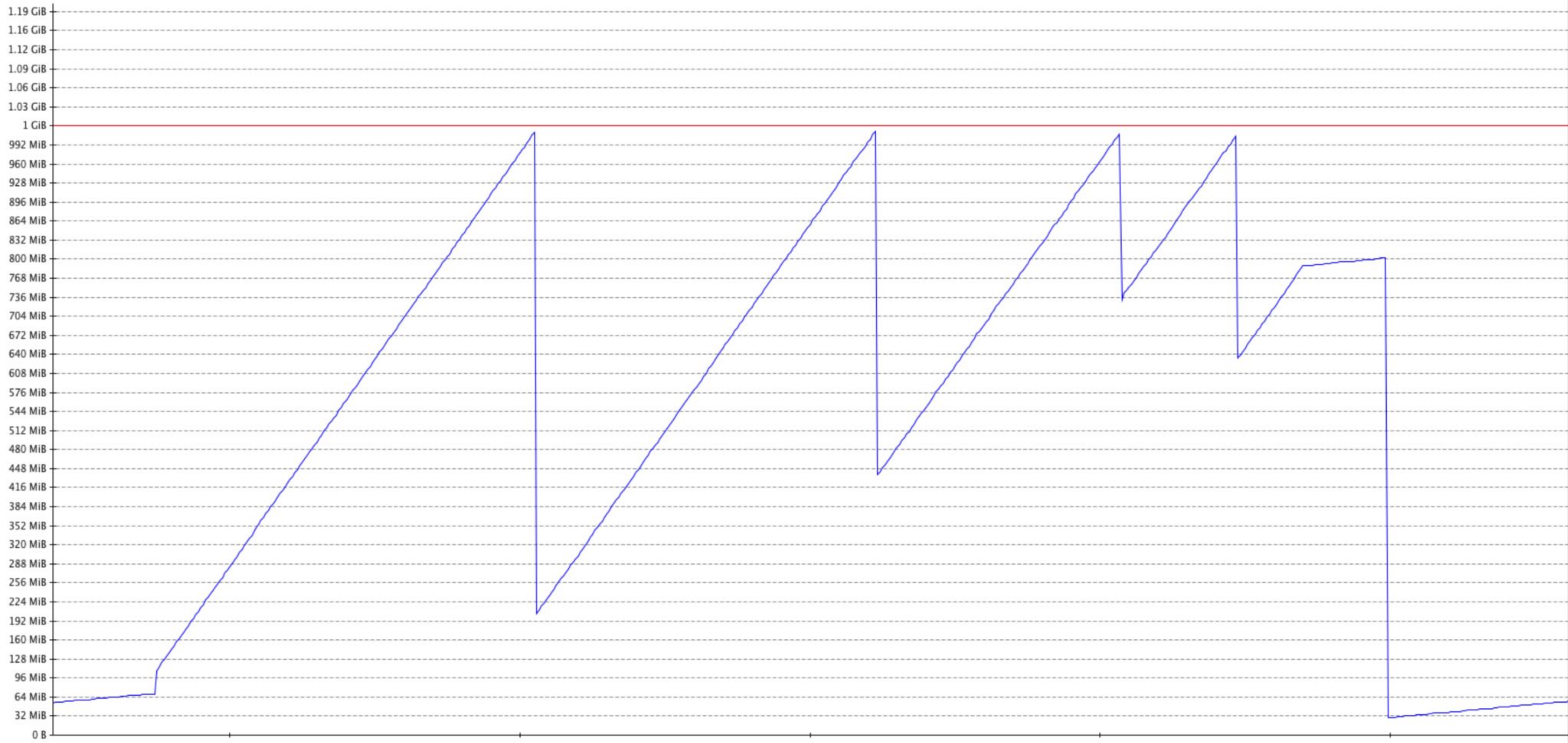


- Java 23 (BellSoft Liberica 23+38 2024-09-17)
- G1 GC

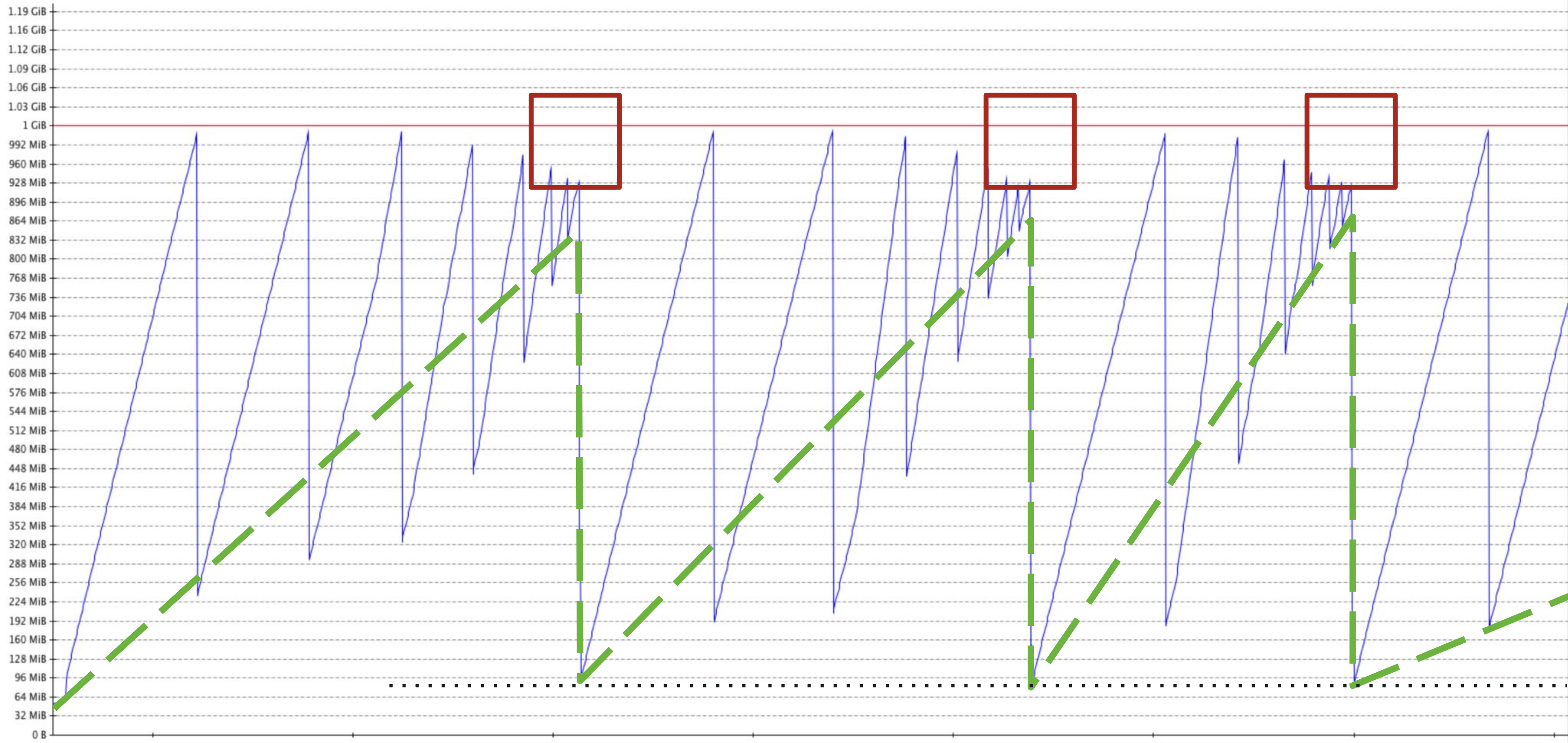
Memory leak or not?



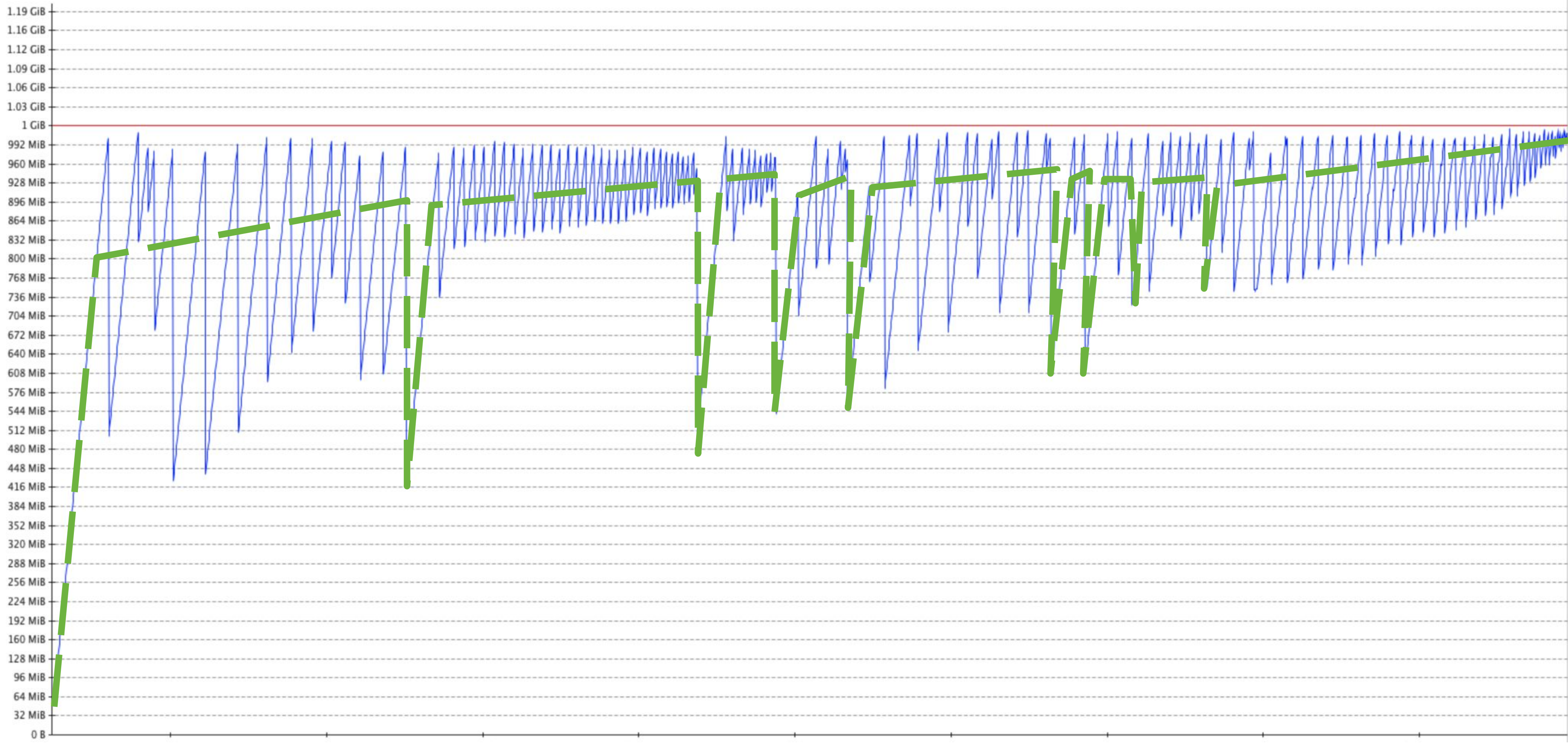
Memory leak or not?



Memory leak or not?



Memory leak or not?



What happened?

- A Generational GC was used
- No “double-sawtooth pattern” in heap metrics
- No full GC (can’t say it’s a leak)
- Increase in heap usage was low (no traffic)
- So the GC was ok with high heap usage
- So didn’t trigger full GC (it’s expensive)
- Eventually it did
- Everything was back to normal

What can we learn from it?

- Human errors are inevitable
- GC’s are complicated
- Even more complicated than that
- Nope, they are even more complicated

Solution

- Fully automate your deployments
- Observe and alert on traffic patterns
- Learn the JVM basics
- Observe heap usage
- Observe GC events

The app from the past

The app from the past

- We had a Service Registry (Eureka) 🎉
- An app was transferred to another team ➡️
- They removed the Eureka Client from it 🧨
- Two years later the app appeared in Eureka 🧐
- The team did not put the Eureka Client back 😱

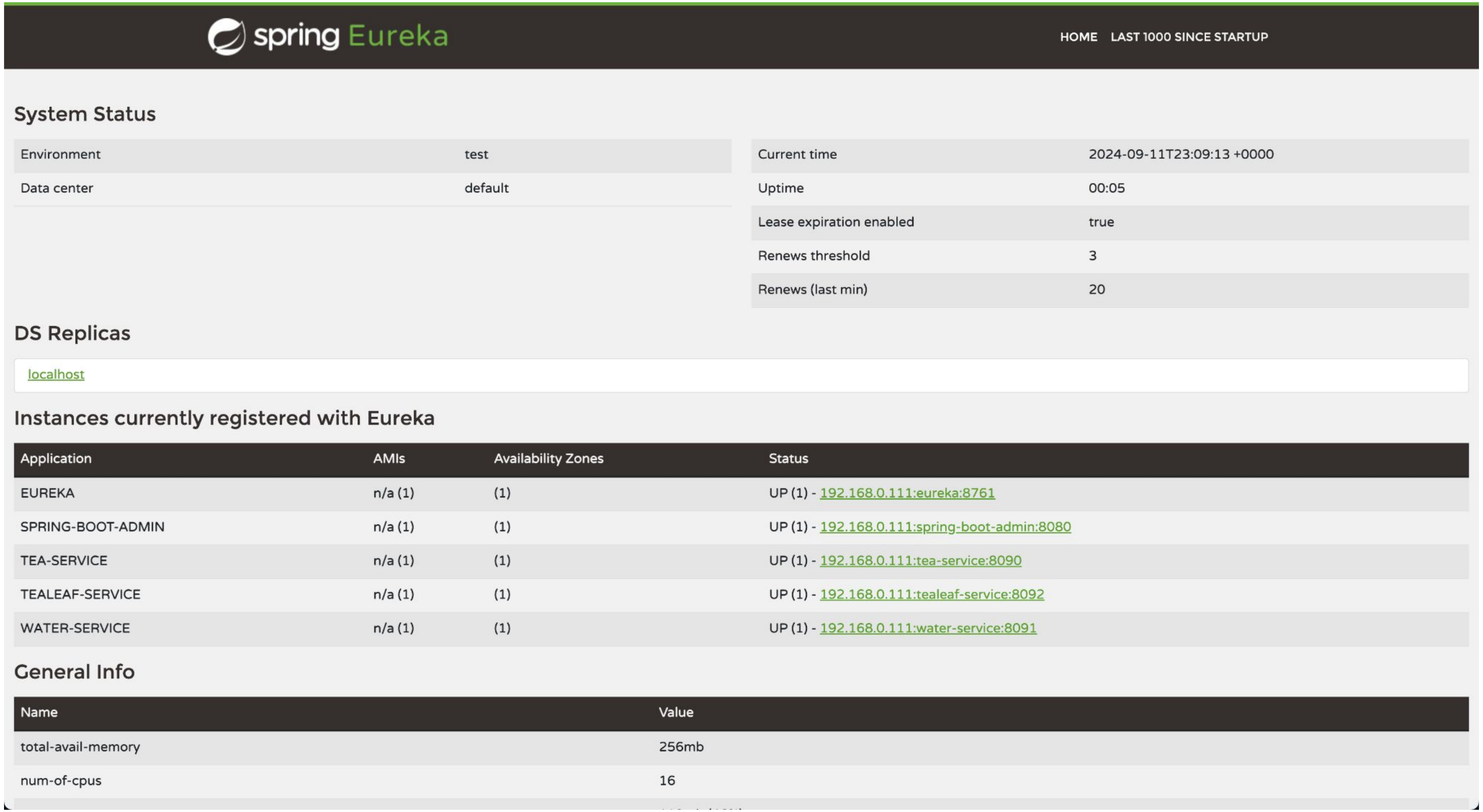
What happened?

- A 2+ years old version of the app started
- The app was a data processing bot
- It processed data

What can we learn from it?

- We need more data about what is running!
- What apps are running (by env)?
- What versions are running (by env)?
- Where are they (by env)?
  (host+port, instance, region, cloud provider, ...)
- How many instances (by env)?
- Service starts/stops (deployments, restarts)?

Solution: Service Registry



System Status

Environment	test
Data center	default

Current time	2024-09-11T23:09:13 +0000
Uptime	00:05
Lease expiration enabled	true
Renews threshold	3
Renews (last min)	20

DS Replicas

localhost

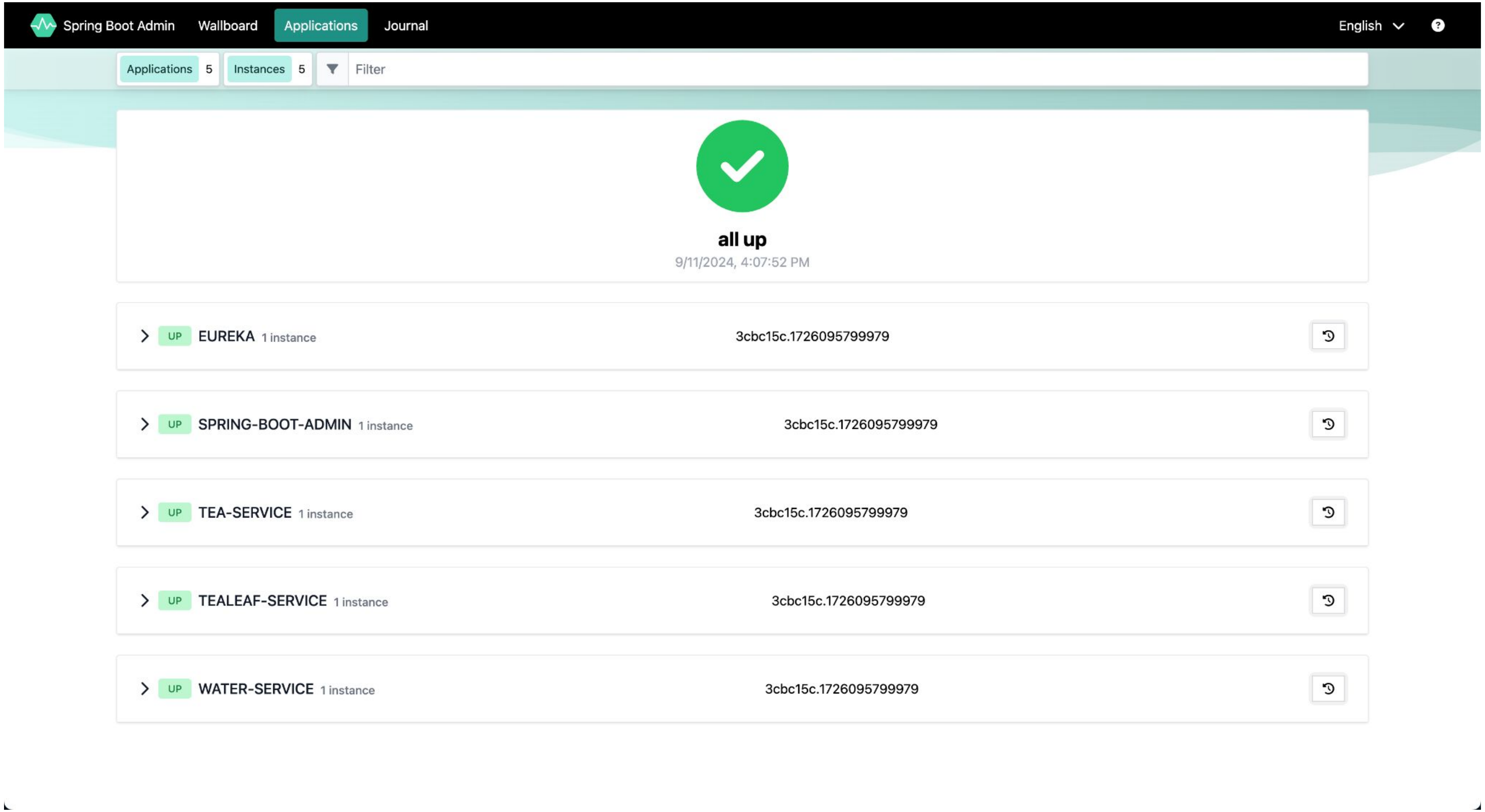
Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
EUREKA	n/a (1)	(1)	UP (1) - 192.168.0.111:eureka:8761
SPRING-BOOT-ADMIN	n/a (1)	(1)	UP (1) - 192.168.0.111:spring-boot-admin:8080
TEA-SERVICE	n/a (1)	(1)	UP (1) - 192.168.0.111:tea-service:8090
TEALEAF-SERVICE	n/a (1)	(1)	UP (1) - 192.168.0.111:tealeaf-service:8092
WATER-SERVICE	n/a (1)	(1)	UP (1) - 192.168.0.111:water-service:8091

General Info

Name	Value
total-avail-memory	256mb
num-of-cpus	16

Solution: Service Registry



all up
9/11/2024, 4:07:52 PM

UP EUREKA 1 instance 3cbc15c.1726095799979

UP SPRING-BOOT-ADMIN 1 instance 3cbc15c.1726095799979

UP TEA-SERVICE 1 instance 3cbc15c.1726095799979

UP TEALEAF-SERVICE 1 instance 3cbc15c.1726095799979

UP WATER-SERVICE 1 instance 3cbc15c.1726095799979

The one you won’t believe

The one you won’t believe

- Trying to reproduce an intermittent issue (fixed)
- Running the same app in a loop 100 times
- Thousands of executions (start-work-verify-stop)
- One time, the app crashed 😱

```
java.lang.ClassFormatError:
Unknown constant tag 41 in class file
io/rsocket/frame/FrameLengthCodec
```

What happened?

- `ClassFormatError`: the class file is malformed
- The class was loaded many times except once
- No dynamic class loading or byte-code generation
- Nothing was changed between executions
- No disk or memory issue was found
- Single-Event Upset (SEU)?

What can we learn from it?

- Anything that can go wrong, will go wrong
- Even if you think it cannot
- Unknown Unknowns
- Umwelt

Solution

- Observability

Expired TLS certificate

Expired TLS certificate

- What happened?
  - The certificate expired. 😢
- What can we learn?
  - Don’t let the certificate expire! 🙃
- Solution
  - Alert before it happens! ⚠️

    (server/client certs, LB, API GW, etc.)

Ask your apps (and your LB, API GW, etc.)

```
"certificates": [
    {
        "subject": "CN=localhost,OU=Spring,L=Seattle,ST=WA,C=US",
        "issuer": "CN=root,OU=Spring,L=Seattle,ST=WA,C=US",
        "version": "V3",
        "serialNumber": "64d019d1dd94eee0",
        "signatureAlgorithmName": "SHA256withRSA",
        "validityStarts": "2024-06-21T21:32:22Z",
        "validityEnds": "2024-06-22T21:32:22Z",
        "validity": {
            "status": "WILL_EXPIRE_SOON",
            "message": "Will expire within threshold (72h) at …"
        }
    }
]
```

What else?

- Kubernetes CPU rq and Memory limit
- Clock skew (or wrong timezone)
- Deploying the wrong version
- Deploying the wrong profile/config
- Unpatched OS
- Unpatched dependency (SBOM)
- Restarted the wrong instance

Thank you!

@jonatan_ivanov

develotters.com

slack.micrometer.io

GH: jonatan-ivanov/resourceater