Benchmarking Function Hook Latency in Cloud-Native Environments

November 7, 2023

PRESENTER

Mario KAHLHOFER
Dynatrace
Research

CO-AUTHOR

Patrick KERN
Dynatrace
Research

CO-AUTHOR

Sören HENNING
Johannes Kepler
University Linz &
Dynatrace Research

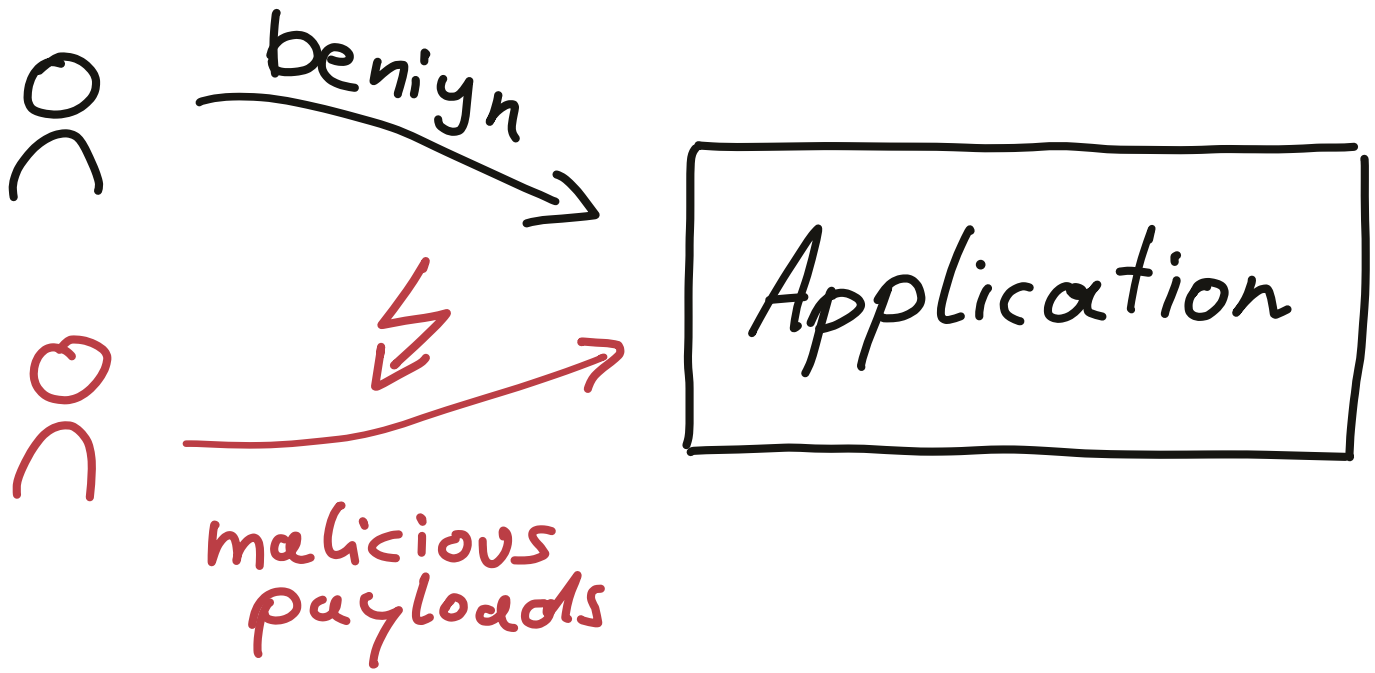
CO-AUTHOR

Stefan RASS
Johannes Kepler
University Linz

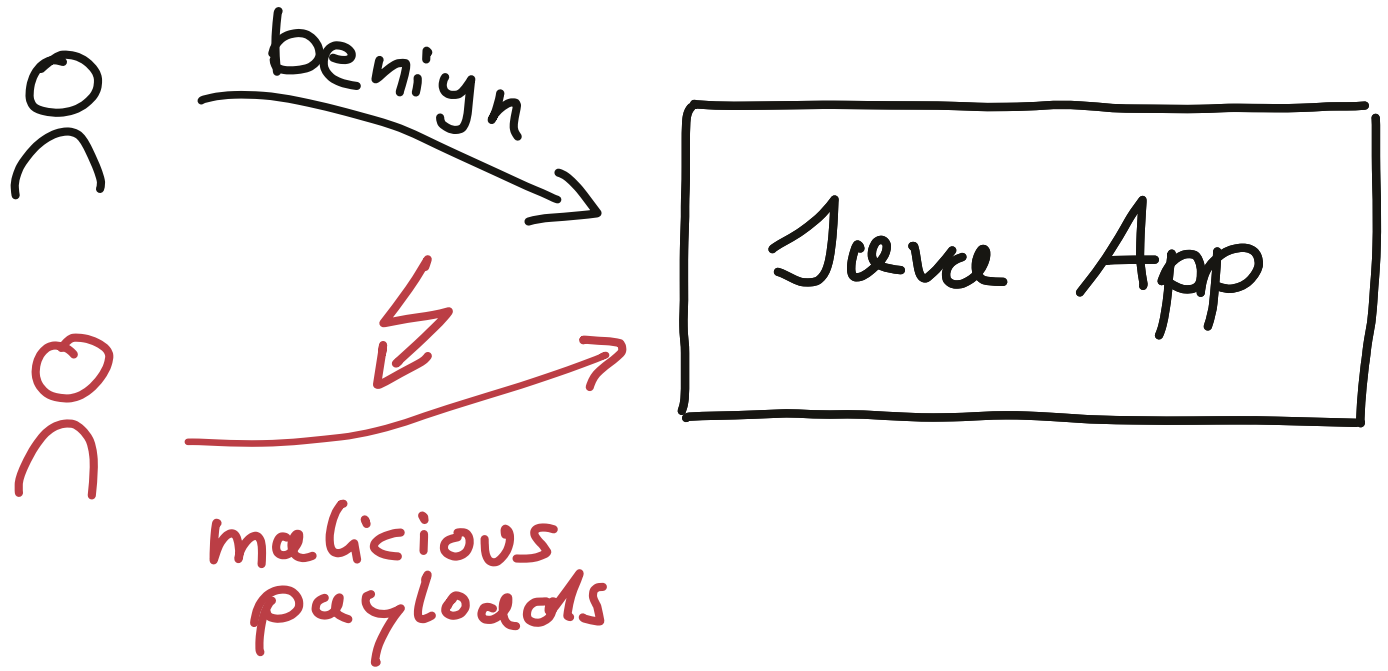
dynatrace

JOHANNES KEPLER UNIVERSITÄT LINZ

USE CASE: Blocking malicious payloads



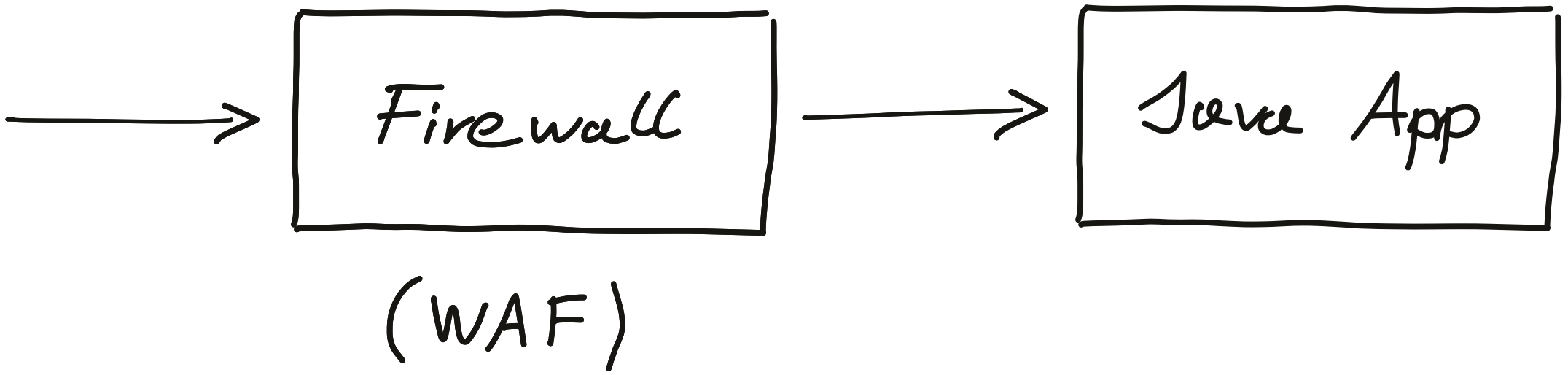
USE CASE: Blocking malicious payloads (cont.)



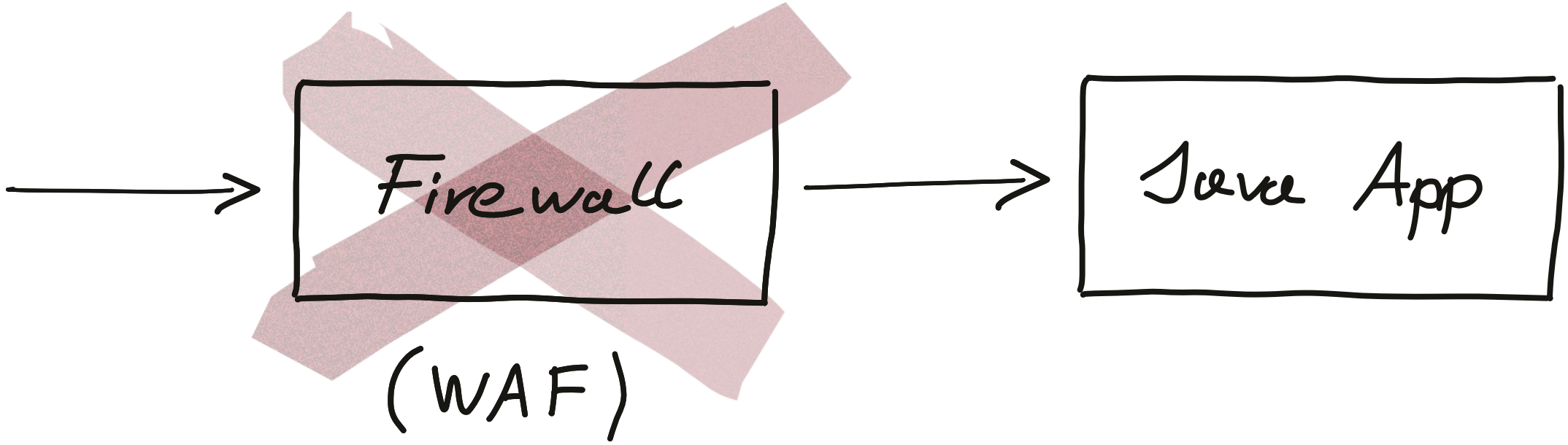
USE CASE: Blocking malicious payloads (cont.)

```
def accept_request(payload):
    if contains_keyword(payload):
        block()
    continue_processing()
```

USE CASE: Blocking malicious payloads with a firewall

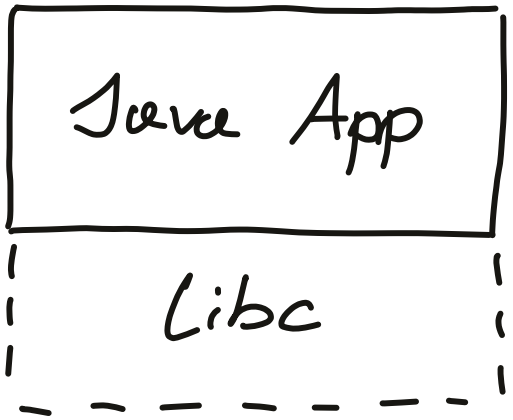


USE CASE: Blocking malicious payloads with a firewall (cont.)

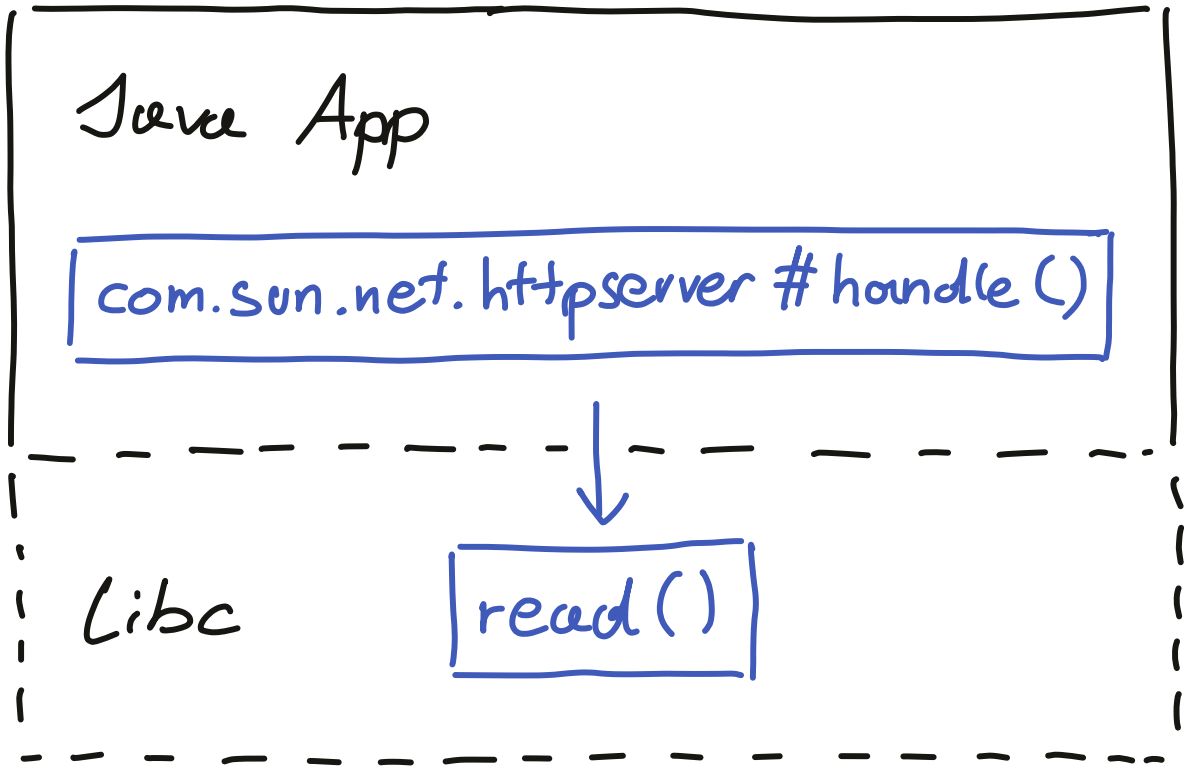


USE CASE: Blocking malicious payloads with function hooks

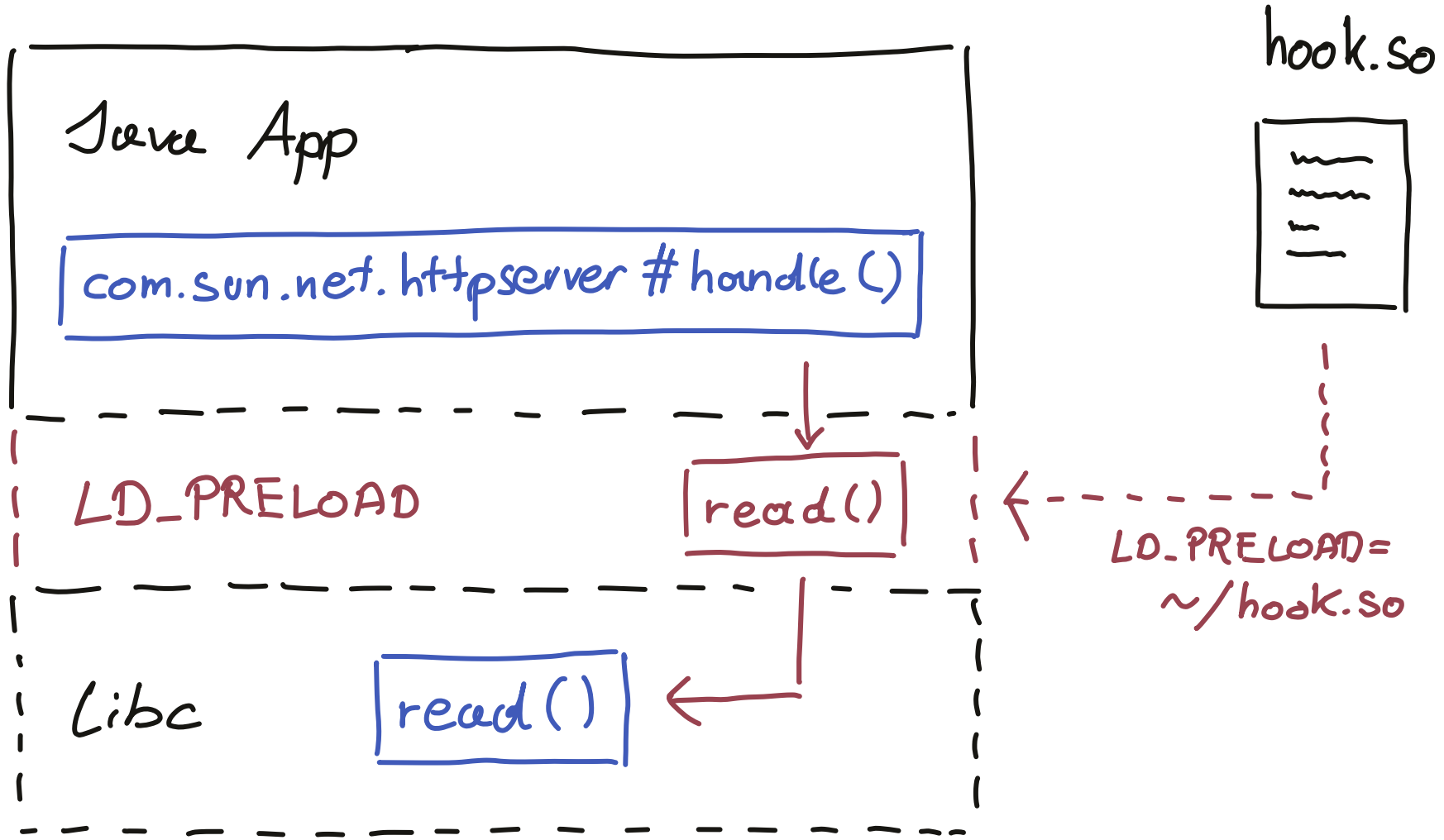
USE CASE: Blocking malicious payloads with function hooks (cont.)



USE CASE: Blocking malicious payloads with function hooks (cont.)



USE CASE: Blocking malicious payloads with function hooks (cont.)

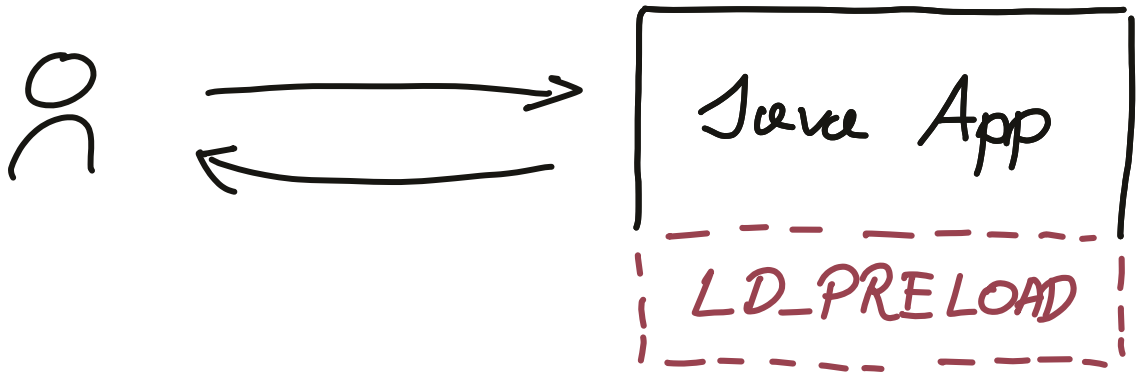


USE CASE: Blocking malicious payloads with function hooks (cont.)

hook.so

```
ssize_t read(int fd, void *buf, size_t count) {
  read_t read_ptr = (read_t)dlsym(RTLD_NEXT, "read");
  ssize_t bytes_read = read_ptr(fd, buf, count);
  if (is_http_socket(fd)) {
    if (contains_keyword(buf, count)) {
      // trace or block call
    }
  }
  return bytes_read;
}
```

PROBLEM: Measure function hook latency

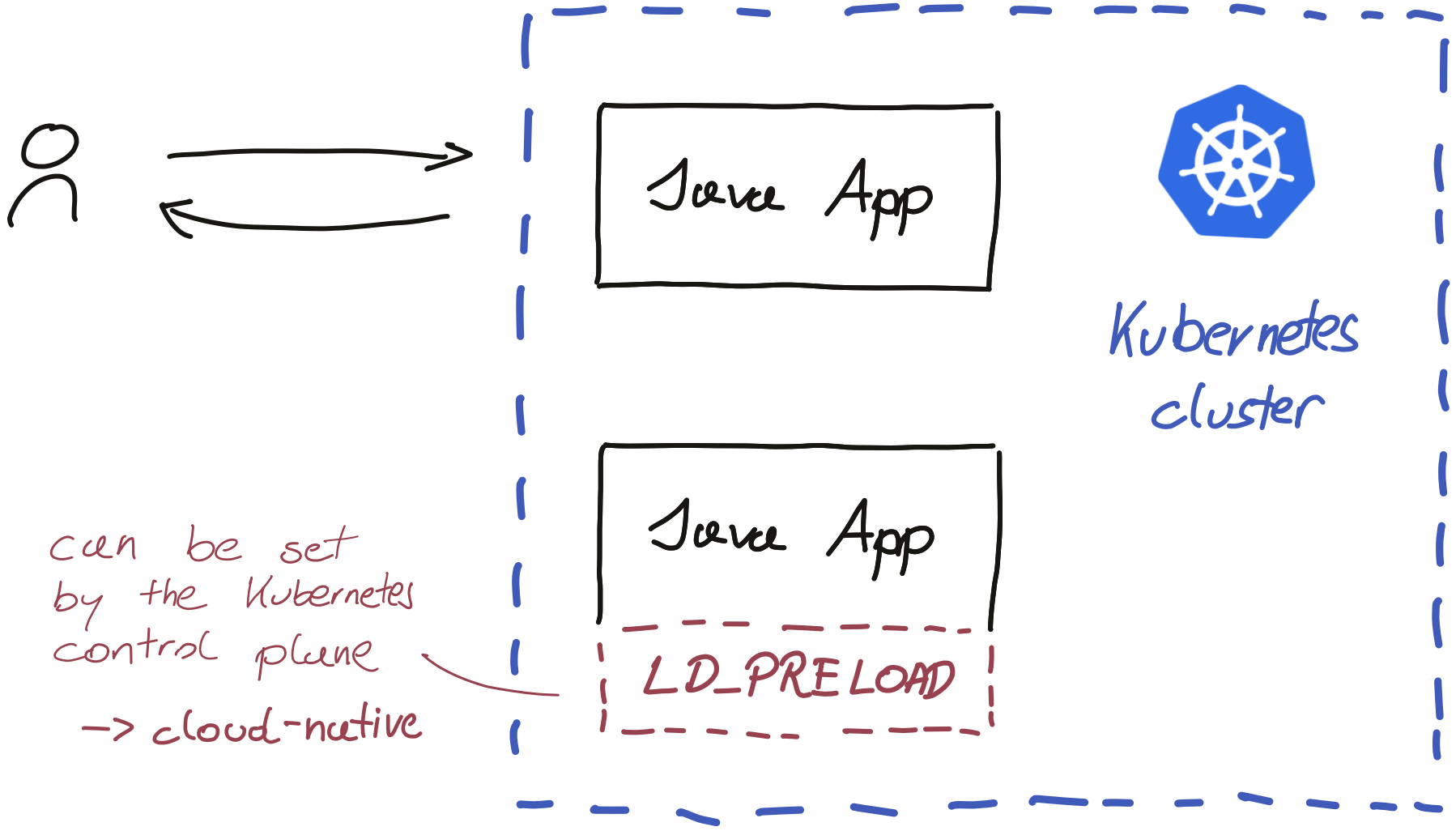


~~CPU~~

~~Memory~~

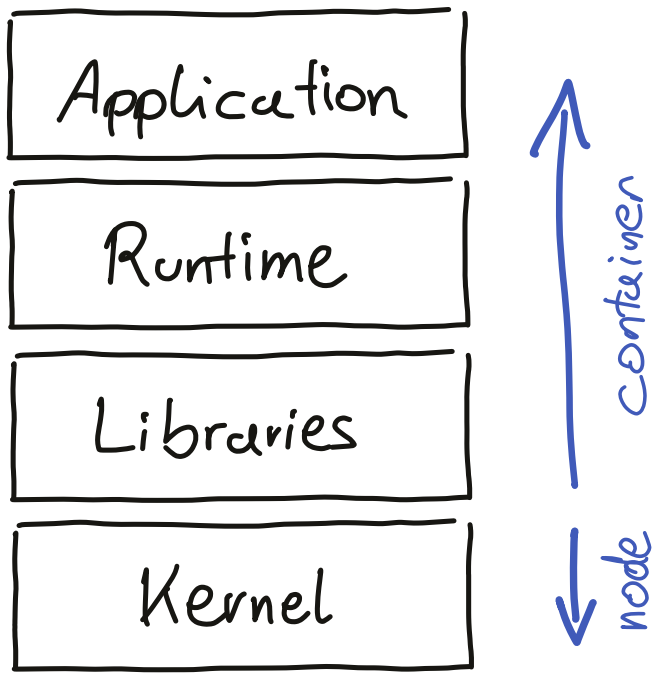
Latency

PROBLEM: Measure function hook latency in Kubernetes



#5 Monitor close by your function hook

```
def accept_request(payload):
    start = time.now()
    if contains_keyword(payload):
        block()
    duration = start - time.now()
    continue_processing()
```



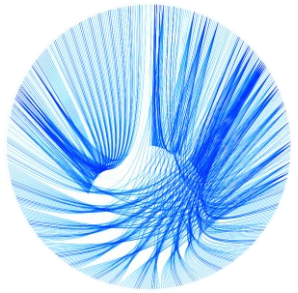
#6 Conduct micro- and macro-benchmarks

Micro benchmarks

Measure the raw hooking overhead in isolation, e.g., by crafting a synthetic app that utilizes the hooked function heavily.

Macro benchmarks

Represent a reference application with the hook injected into it.



DeathStarBench

Unguard

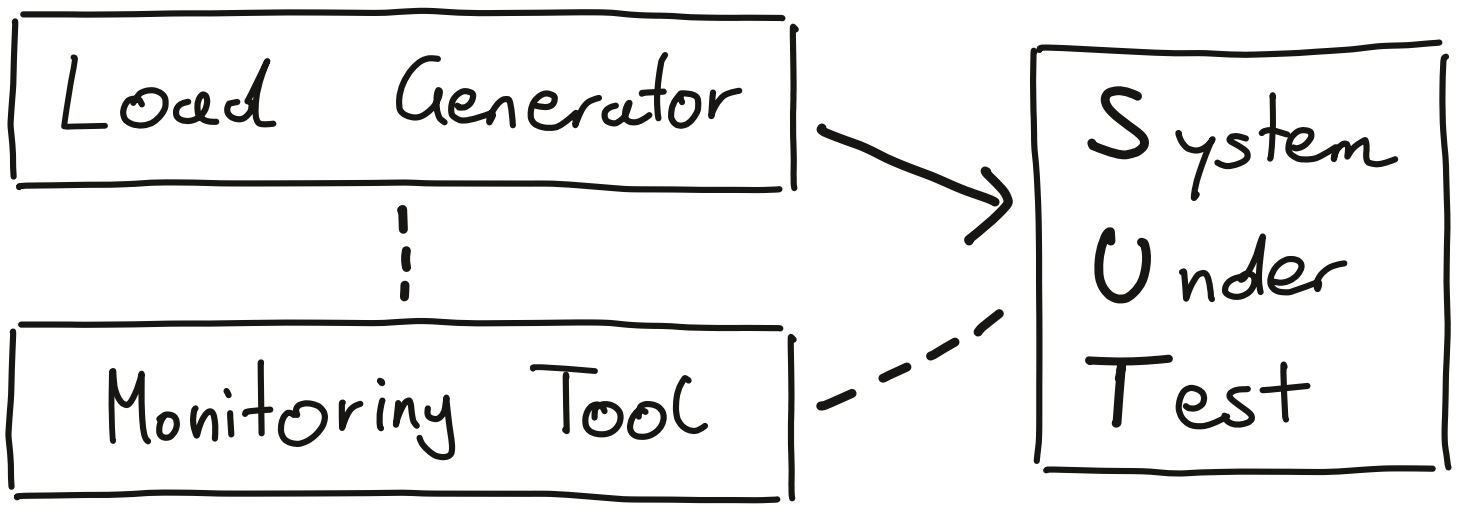
TeaStore

#7 Observe how your hooked function is used

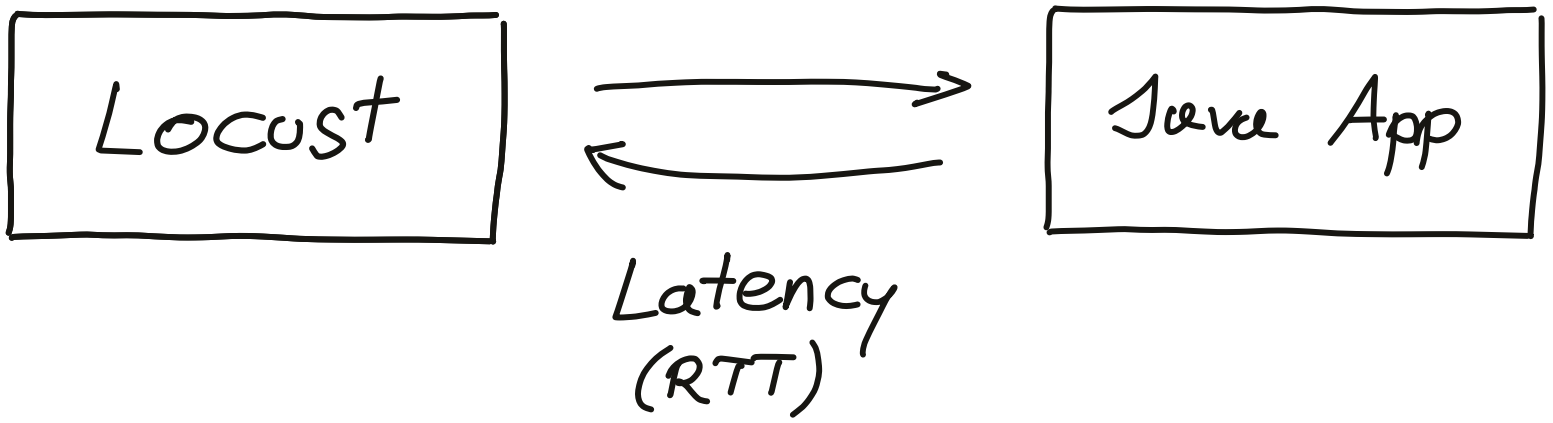
hook.so

```
$ strace curl www.google.at
execve("/home/mario/bin/curl", ["curl", "www.google.at"],
0x7ffe225bcb08 /* 34 vars */) = 0
brk(NULL) = 0x55672d1cf000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe893284a0) = -1 EINVAL
access("/etc/ld.so.preload", R_OK) = -1 ENOENT
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=46190, ...}) = 0
mmap(NULL, 46190, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f224611c000
close(3) = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libcurl.so.4") = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\1\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=596616, ...}) = 0
```

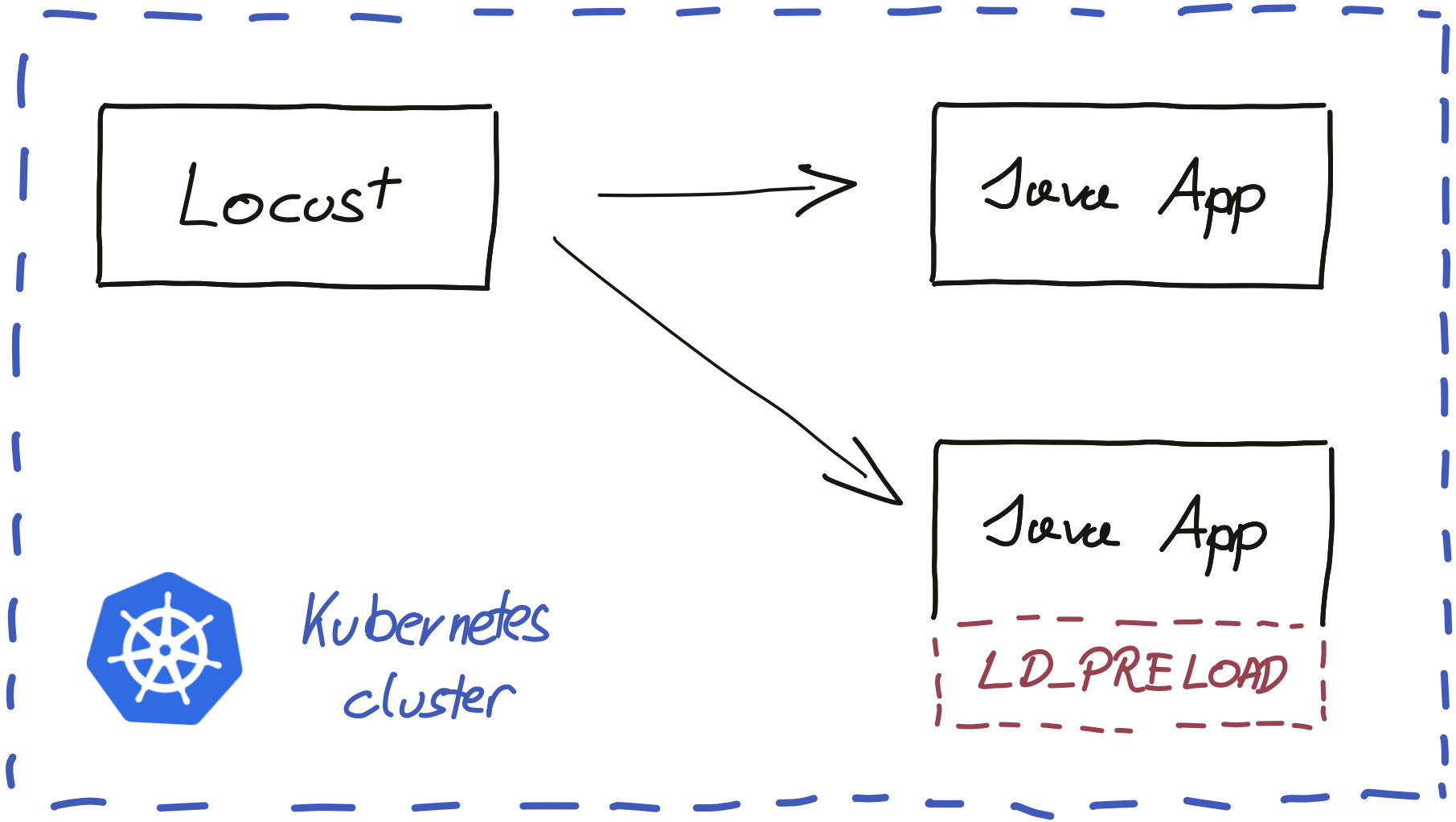
PROBLEM: Benchmarks in the cloud



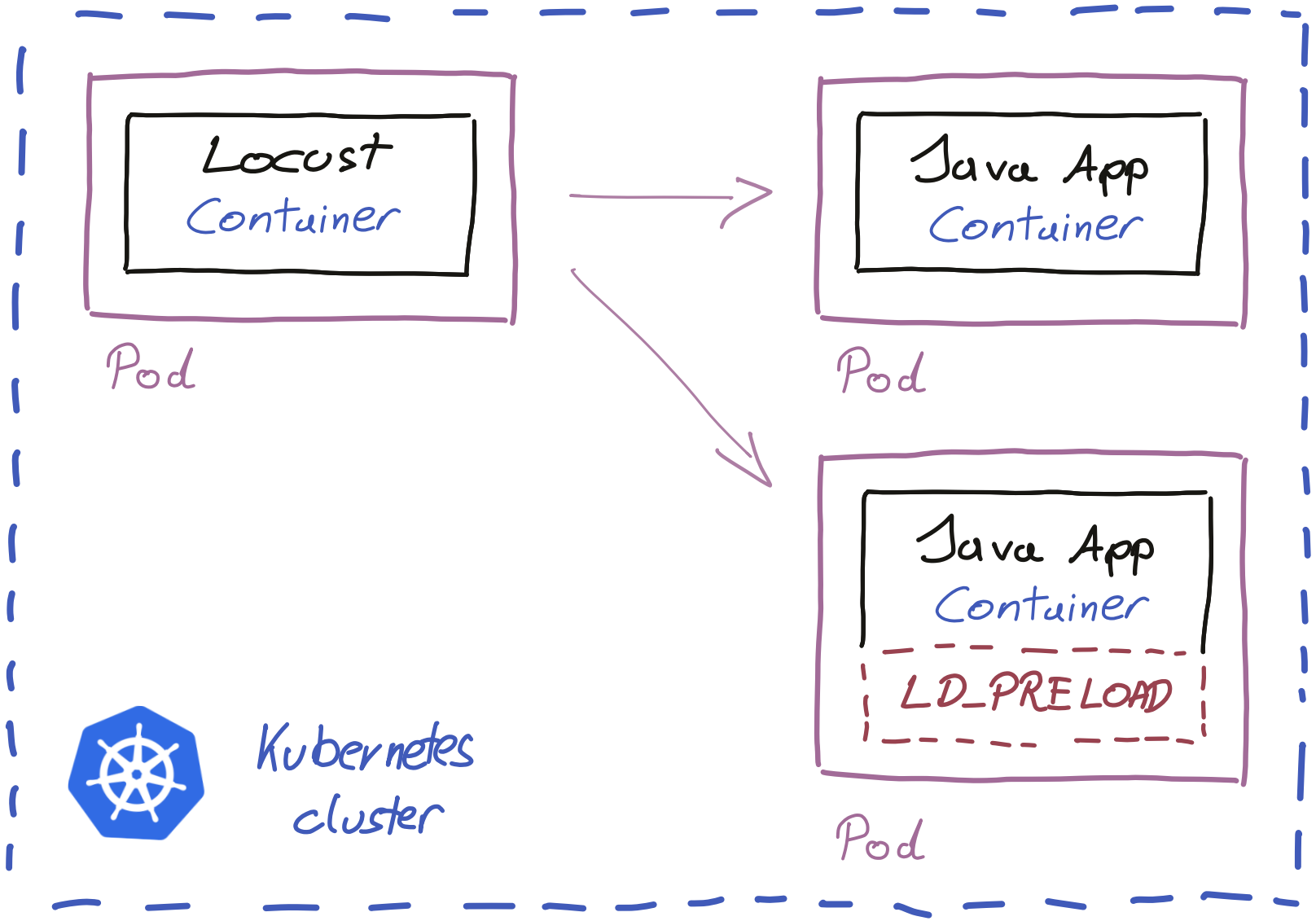
PROBLEM: Benchmarks in the cloud (cont.)



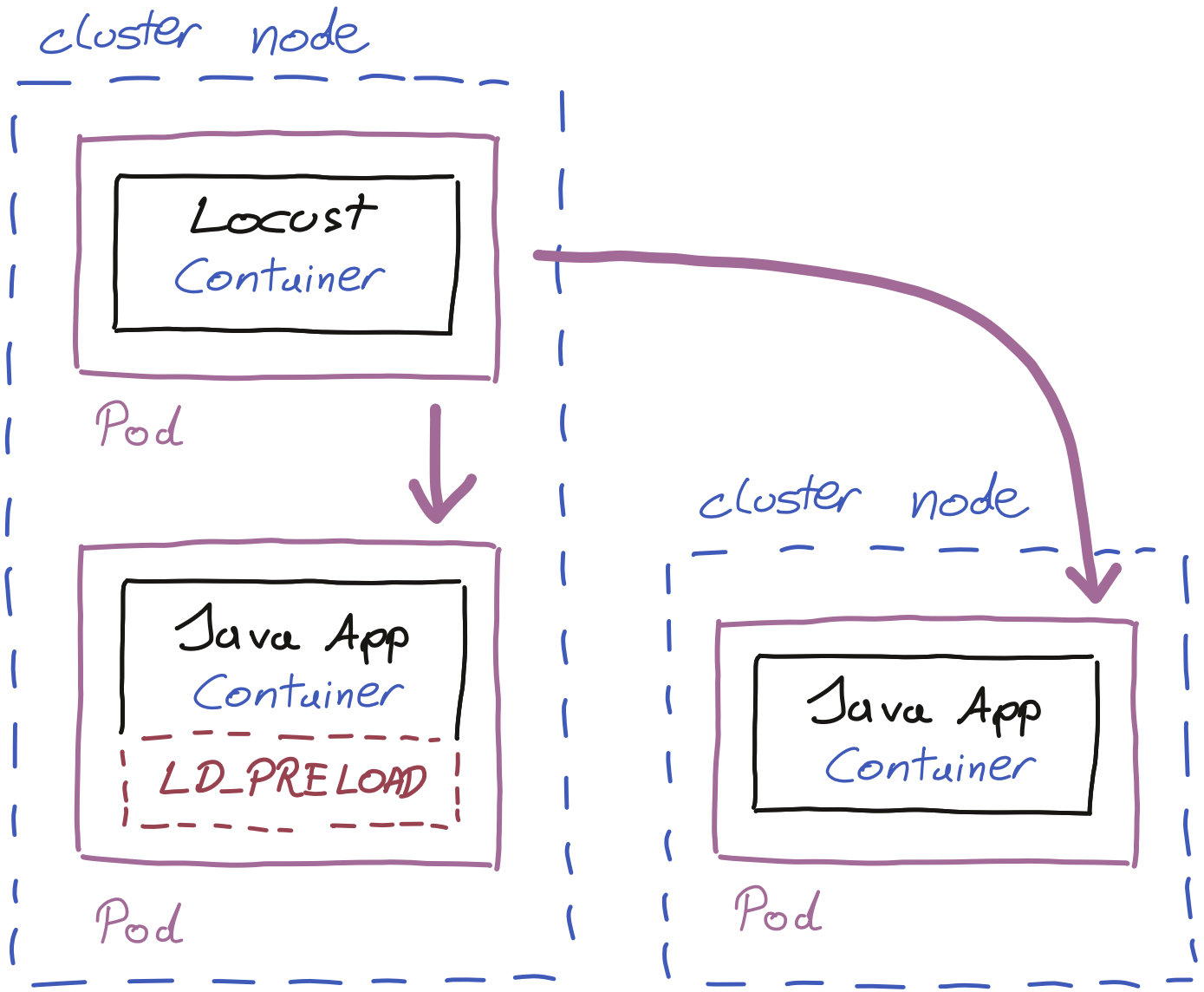
PROBLEM: Benchmarks in Kubernetes clusters



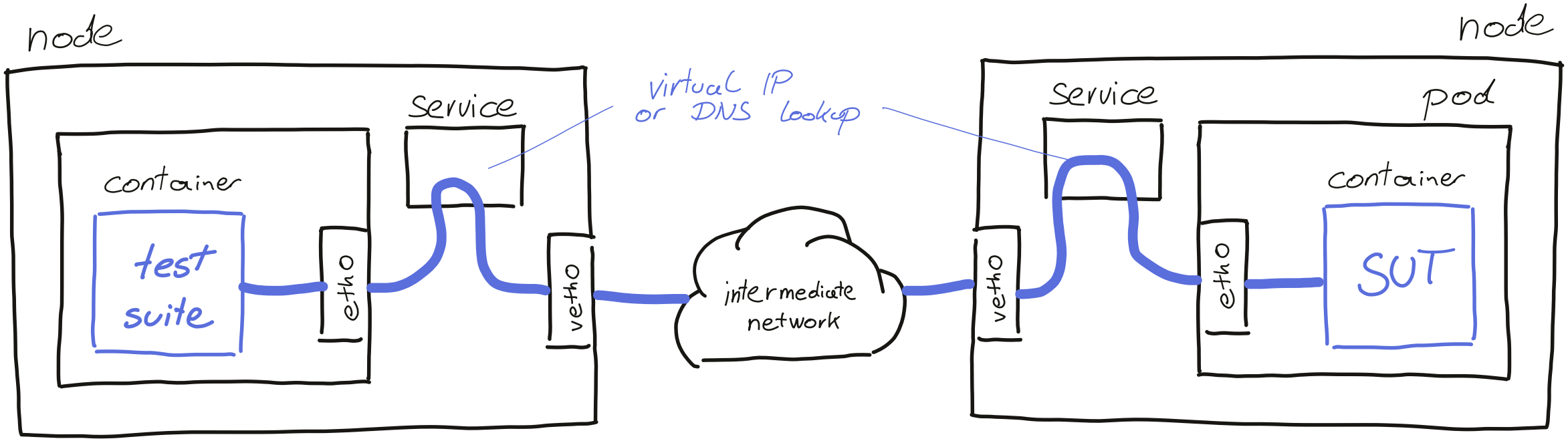
PROBLEM: Benchmarks in Kubernetes clusters (cont.)



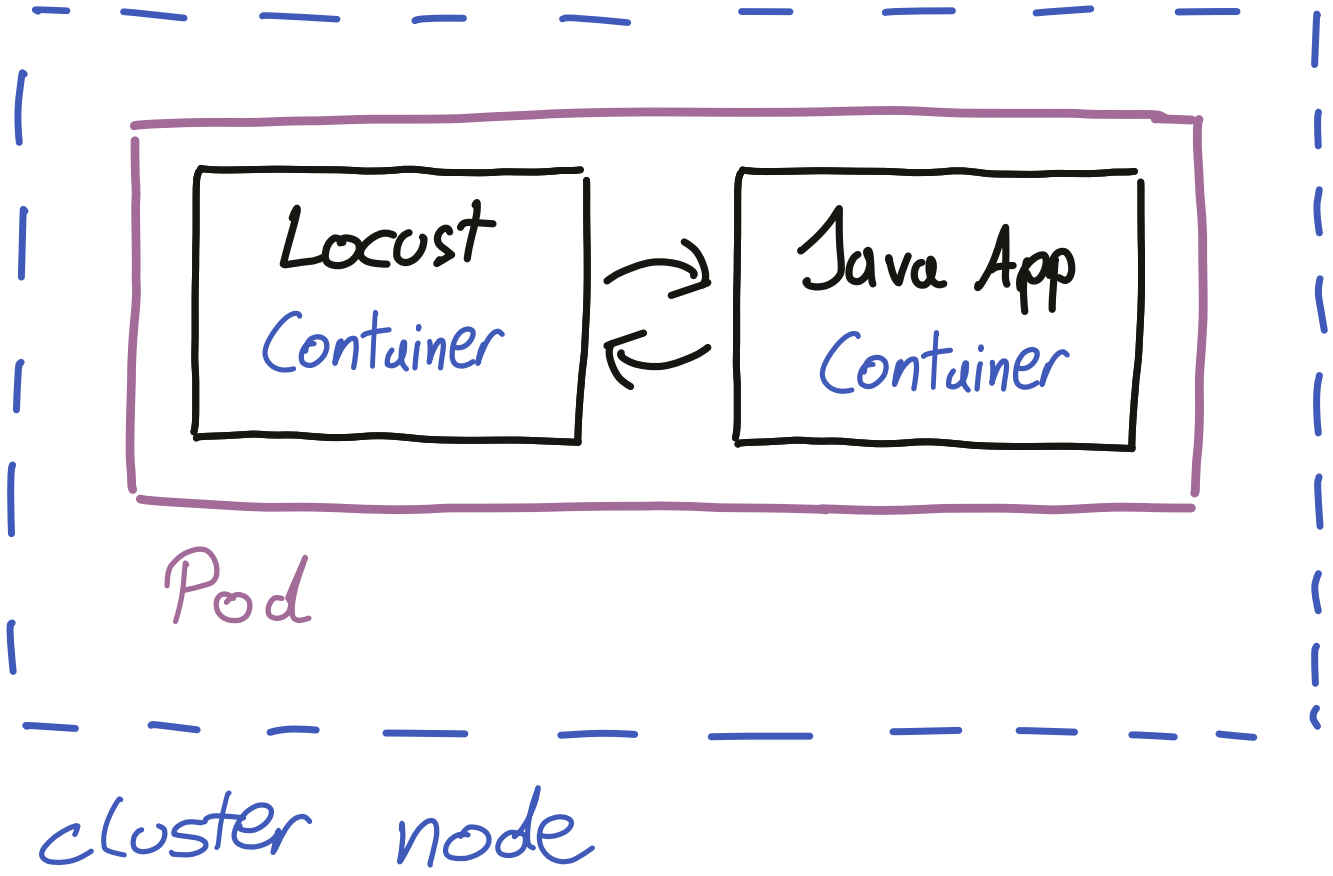
PROBLEM: Benchmarks in Kubernetes clusters (cont.)



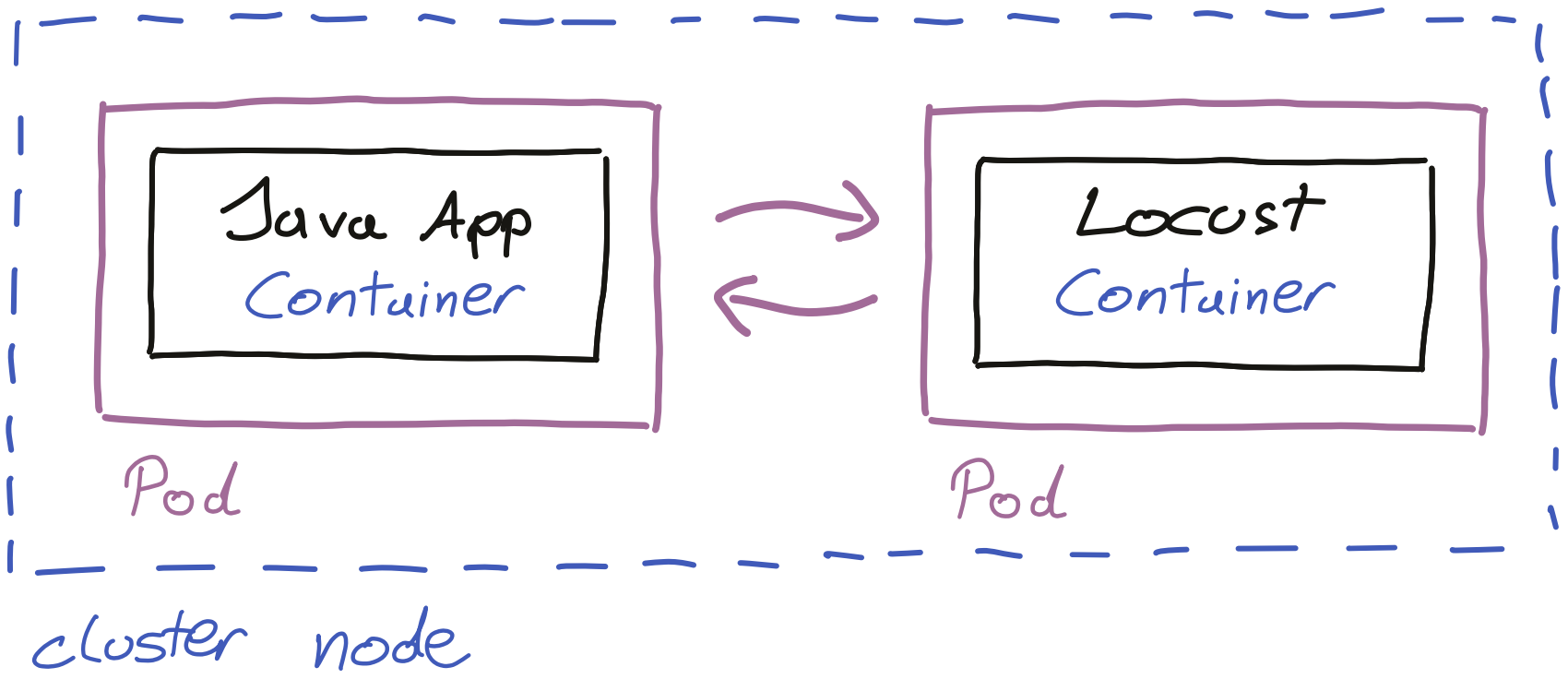
PROBLEM: Benchmarks in Kubernetes clusters (cont.)



#1 Put the load generator and the SUT in the same pod

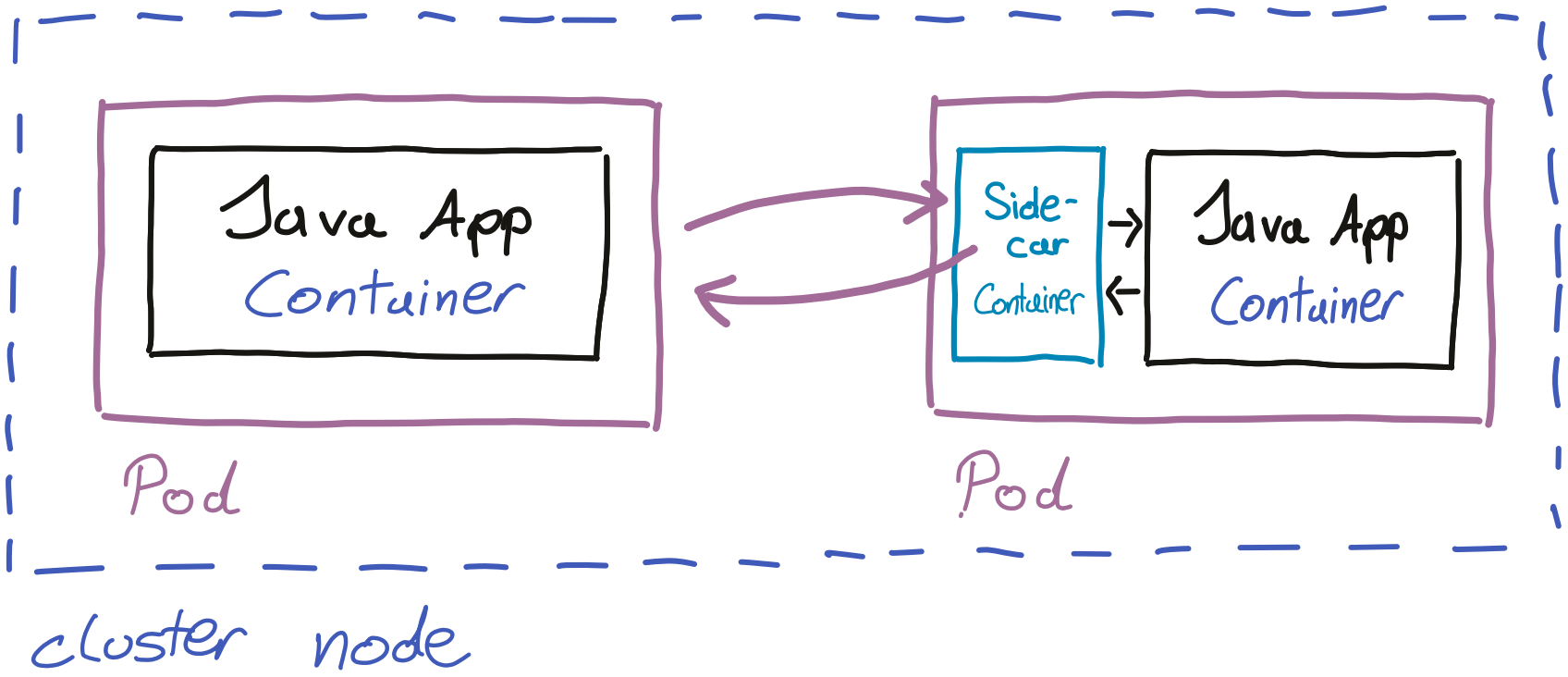


#2 … or at least on the same cluster node

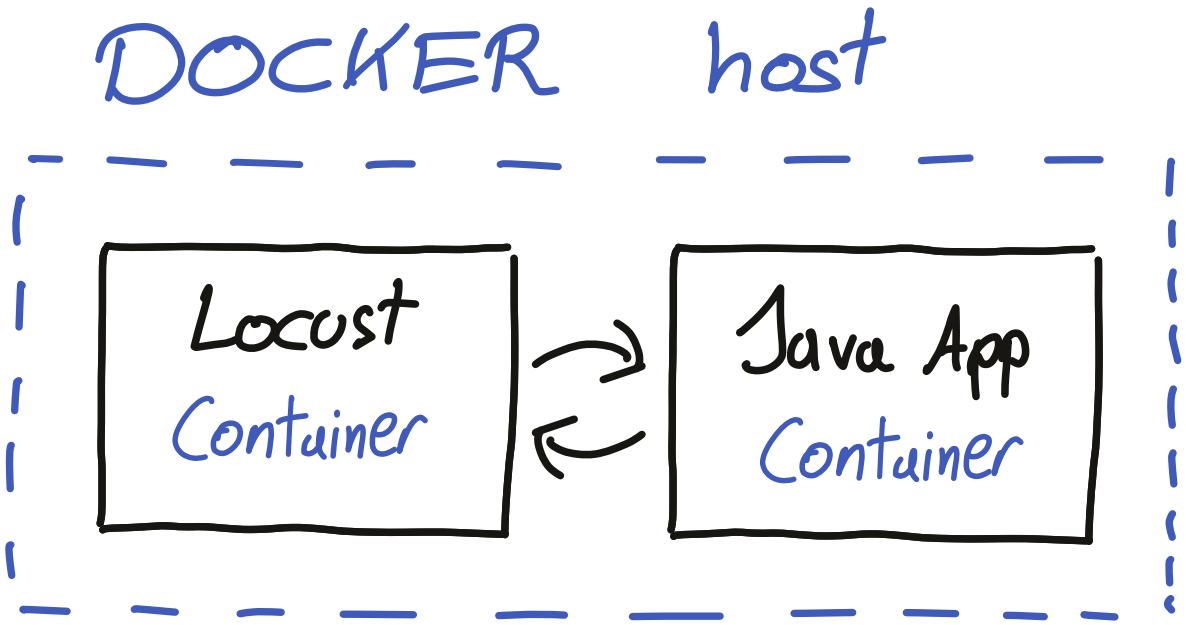


#3 Be mindful about the latency overhead of service meshes

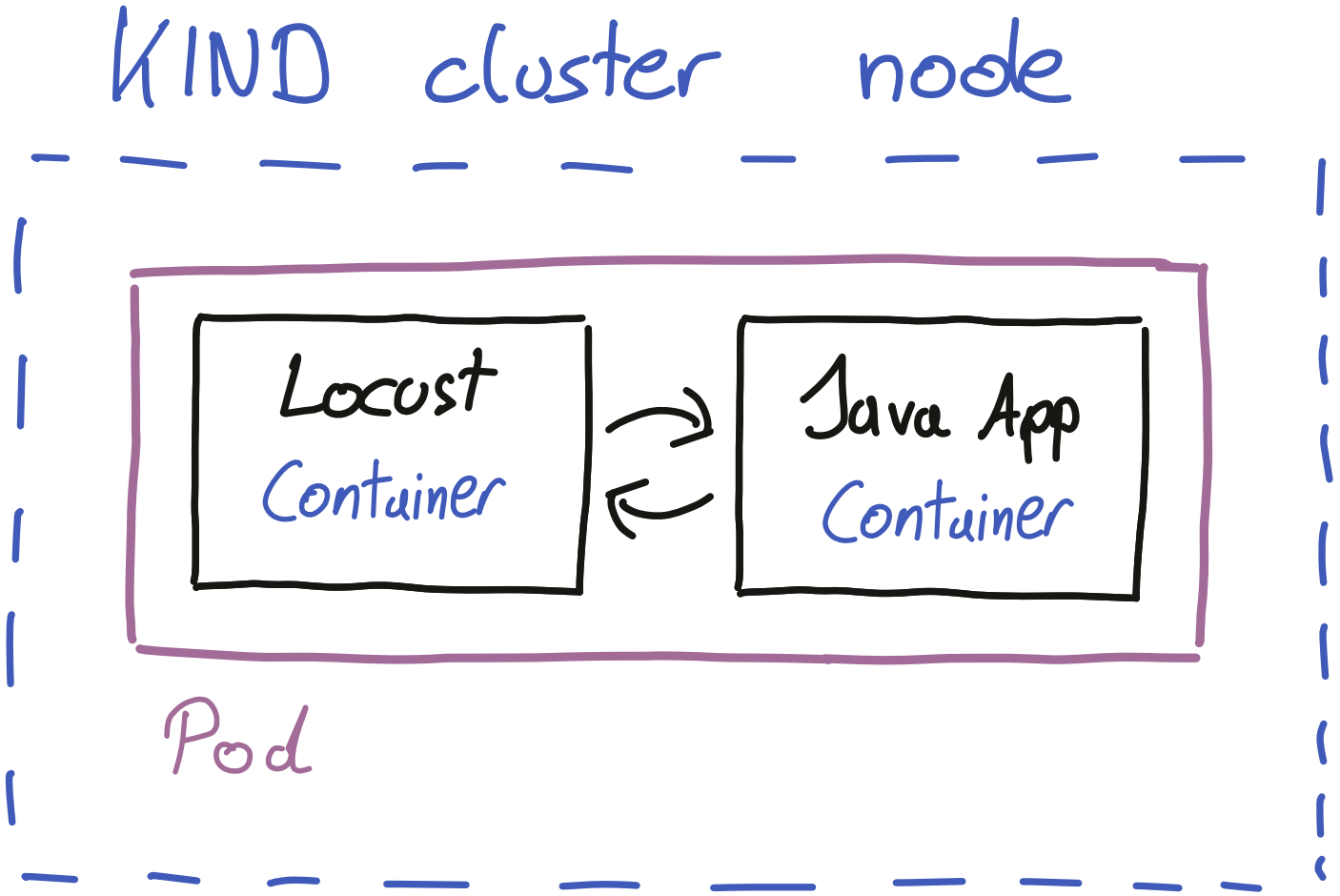
#4 Avoid benchmarks in multi-tenancy clusters



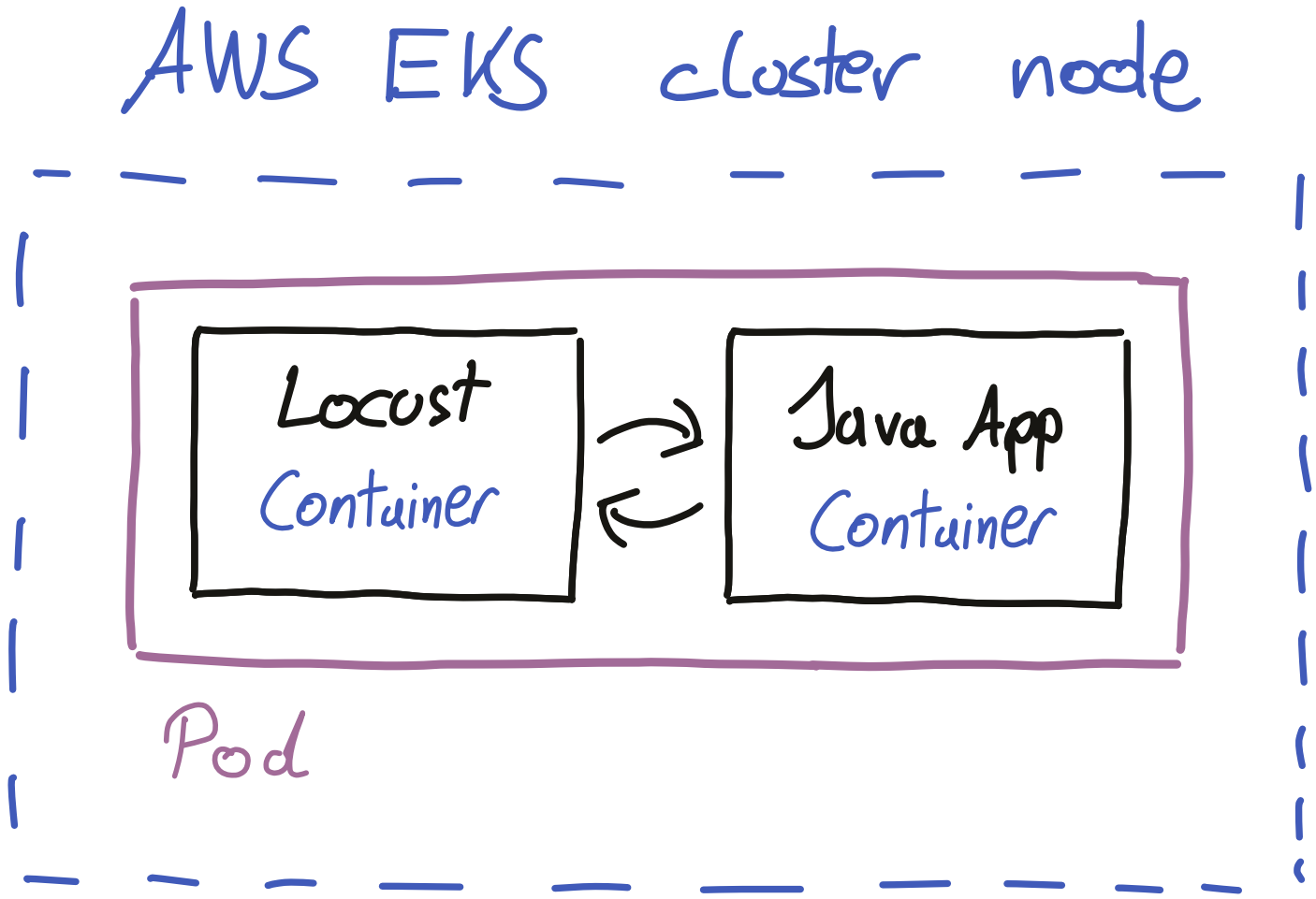
DEMO #A: Docker host with two containers



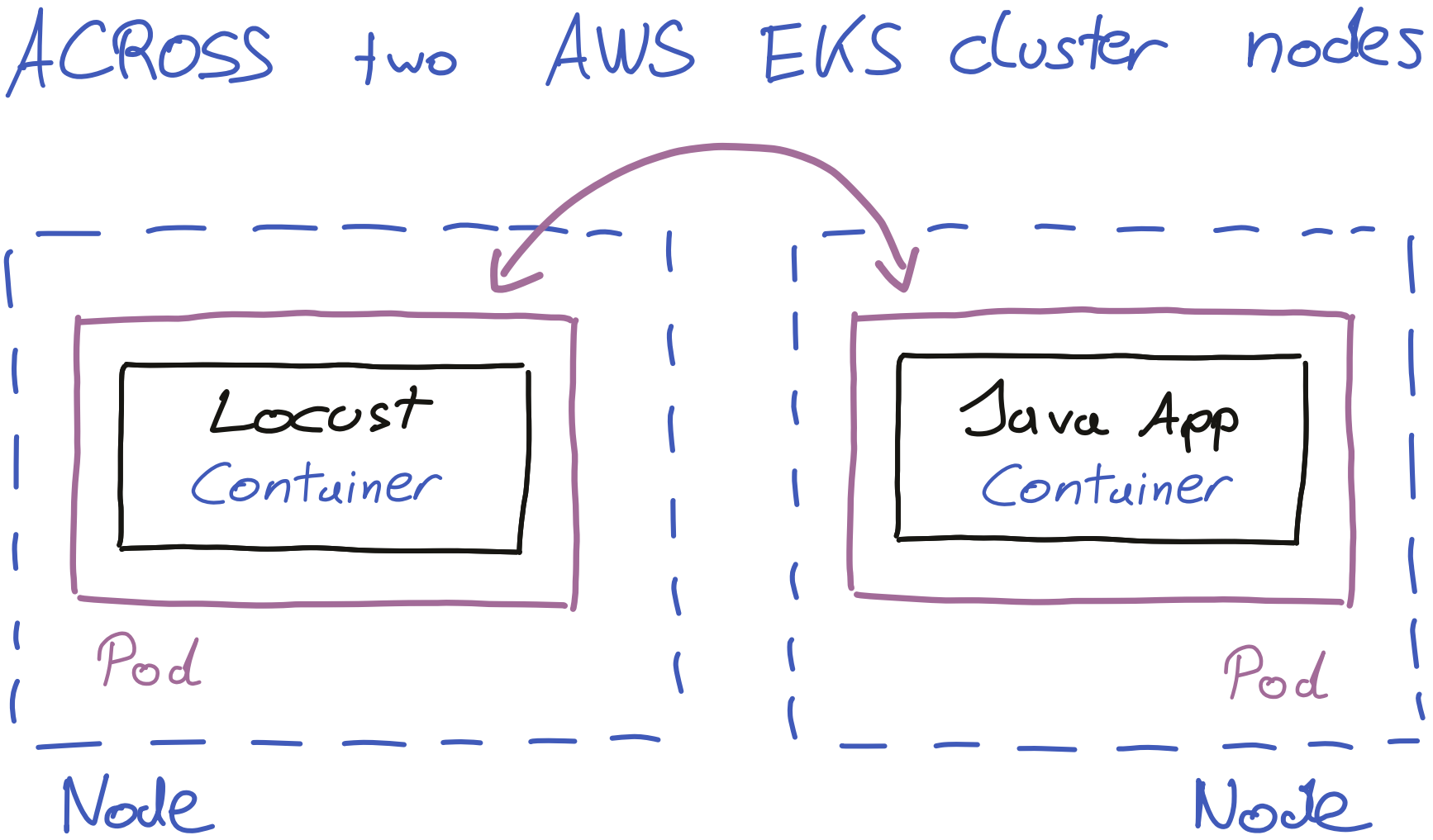
DEMO #B: Kind cluster with two containers in one pod



DEMO #C: AWS EKS cluster with two containers in one pod

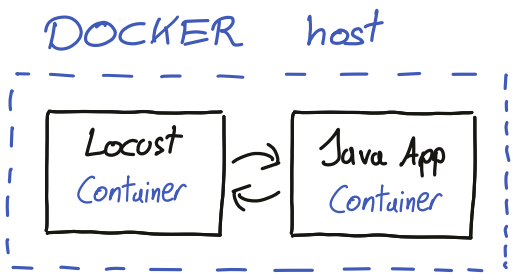


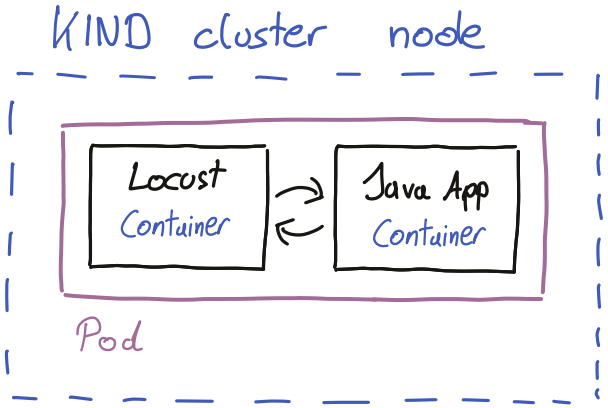
DEMO #D: AWS EKS cluster with separate pods, on different nodes

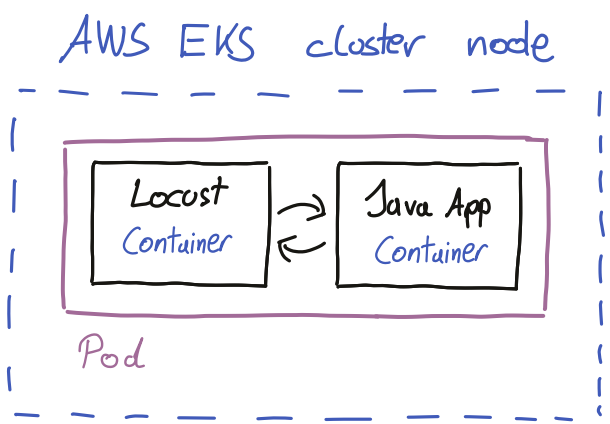


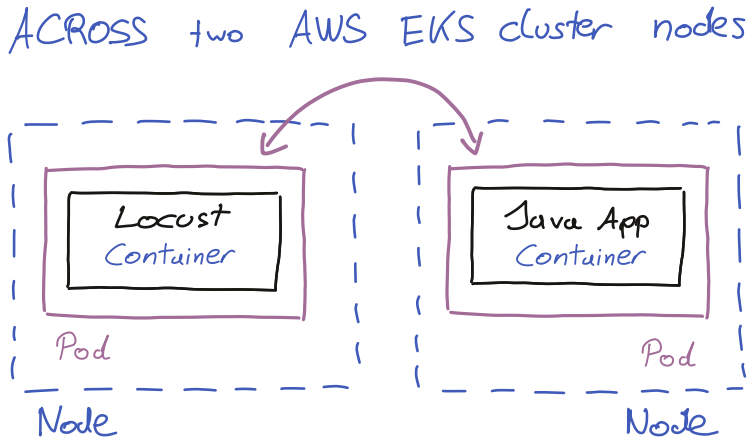
DEMONSTRATION: Overview of the four conditions

#10 Benchmark in diverse cloud environments



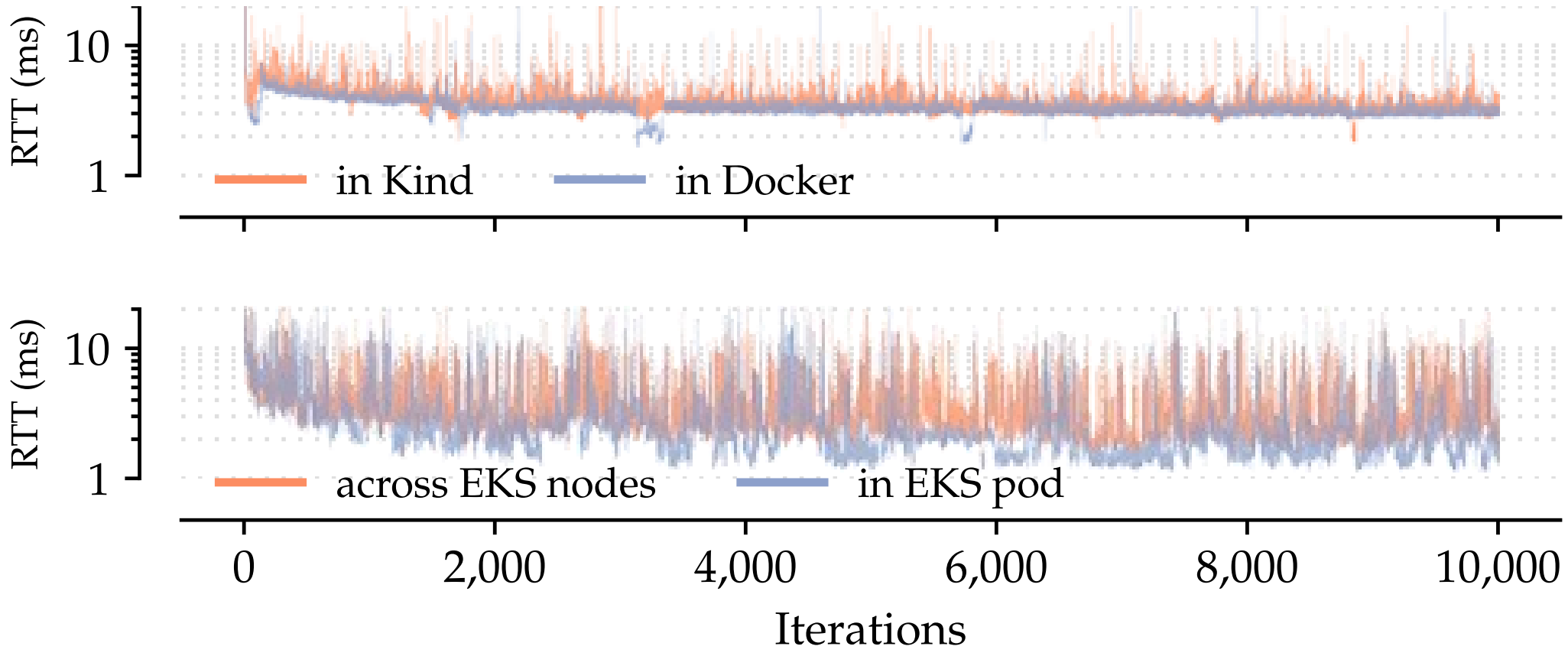






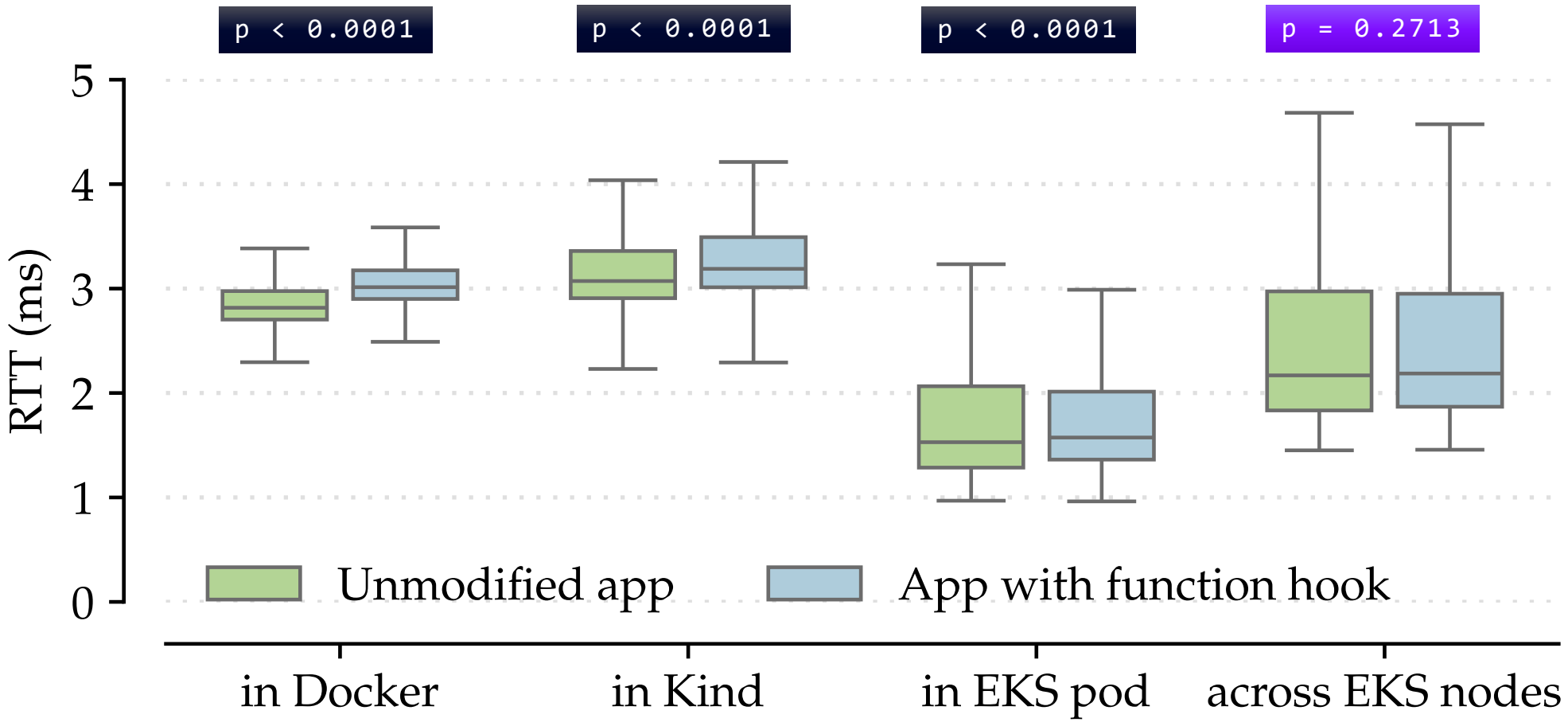
RESULTS: Logarithmic lag plots on measured RTT

#8 Monitor resource limits to avoid resource contention



RESULTS: Measured RTT per condition, without warm-up

#9 Increase sample size to regain statistical power



CONCLUSION

https://github.com/dynatrace-research/function-hook-latency-benchmarking

10

recommendations for benchmarking function hook latency in cloud-native environments

4

conditions demonstrated

AWS EKS showed the highest variance

1

open-source repository to aid reproducibility

Also read empirical standards for software benchmarking [1] and methodological principles for performance evaluation in cloud computing [2]

[1] P. Ralph et al., "Empirical Standards for Software Engineering Research." arXiv, Mar. 04, 2021. doi: 10.48550/arXiv.2010.03525.
[2] A. V. Papadopoulos et al., "Methodological Principles for Reproducible Performance Evaluation in Cloud Computing," TSE, vol. 47, no. 8, pp. 1528–1543, Jul. 2019, doi: 10.1109/TSE.2019.2927908.