Inspektor Gadget: An eBPF Based Tool to Observe Containers
FOSDEM 2023

Francis Laniel

`flaniel@linux.microsoft.com`

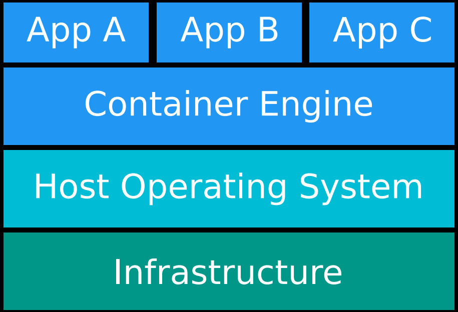
5th February 2023

The containers

The containers rely on several features offered by the kernel:

The `namespaces`: Security isolation [1, 2, 3].

The `cgroups`: Resources isolation [1, 4].



Container (`docker`, `lxc`, `podman`, etc.)

Containers can be hard to debug

Using containers pose several problems to debug applications, among others:

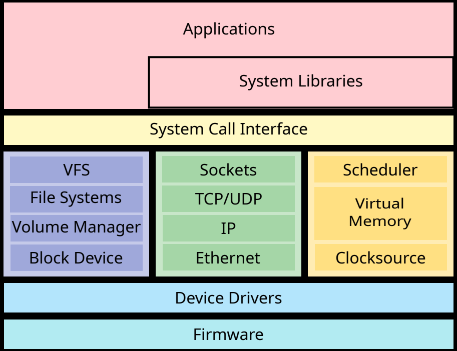
- Harder to attach a debugger to running application.
- One have to take into account the communications between different containers.

Inspektor Gadget

Presentation

A swiss army knife based on eBPF [5]:

- `local-gadget`
- `kubectl-gadget`



Applications

System Libraries

System Call Interface

VFS

File Systems

Volume Manager

Block Device

Sockets

TCP/UDP

IP

Ethernet

Scheduler

Virtual Memory

Clocksource

Device Drivers

Firmware

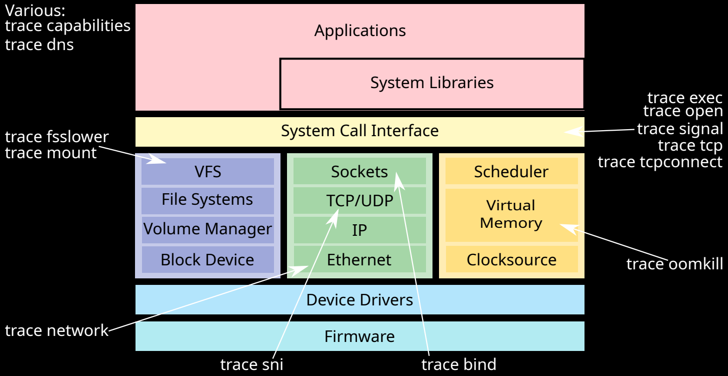
The different tools offered by Inspektor Gadget.

Inspektor Gadget

Presentation

A swiss army knife based on eBPF [5]:

- `local-gadget`
- `kubectl-gadget`



Various:
trace capabilities
trace dns

Applications

System Libraries

System Call Interface

trace exec
trace open
trace signal
trace tcp
trace tcpconnect

trace fsslower
trace mount

VFS
File Systems
Volume Manager
Block Device

Sockets
TCP/UDP
IP
Ethernet

Scheduler
Virtual Memory
Clocksource

trace oomkill

Device Drivers

Firmware

trace network

trace sni

trace bind

The different tools offered by Inspektor Gadget.

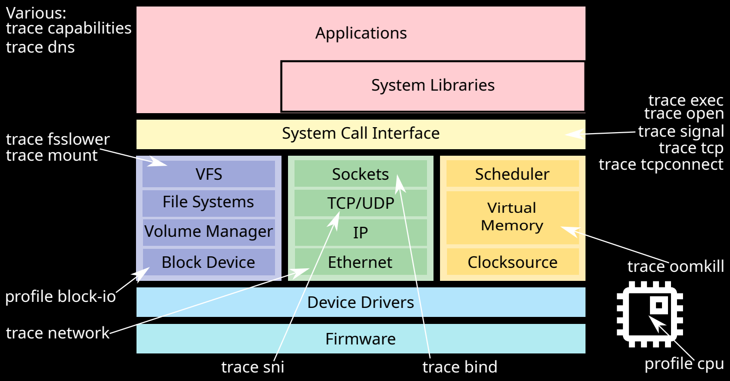
Inspektor Gadget

Presentation

INSPEKTOR GADGET

A swiss army knife based on eBPF [5]:

- `local-gadget`
- `kubectl-gadget`



Various:
trace capabilities
trace dns

Applications

System Libraries

System Call Interface

trace exec
trace open
trace signal
trace tcp
trace tcpconnect

trace fsslower
trace mount

VFS

File Systems

Volume Manager

Block Device

Sockets

TCP/UDP

IP

Ethernet

Scheduler

Virtual Memory

Clocksource

trace oomkill

profile block-io

trace network

Device Drivers

Firmware

trace sni

trace bind

profile cpu

The different tools offered by Inspektor Gadget.

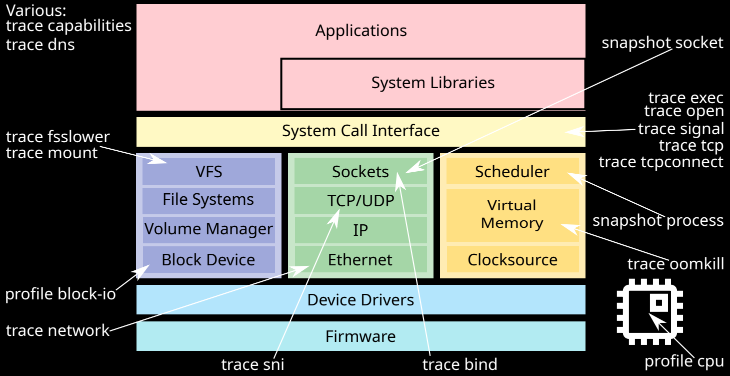
Inspektor Gadget

Presentation

INSPEKTOR GADGET

A swiss army knife based on eBPF [5]:

- `local-gadget`
- `kubectl-gadget`



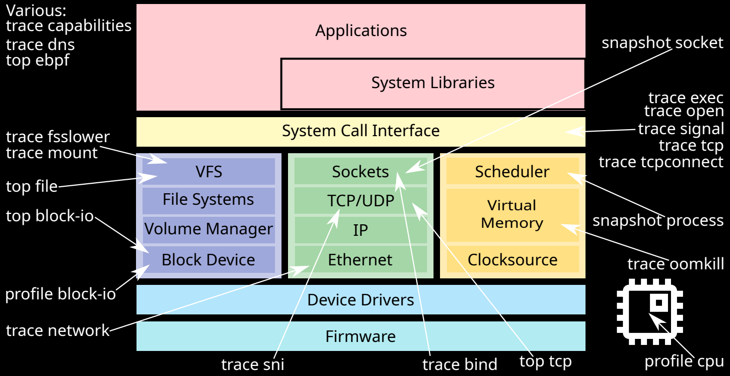
The different tools offered by Inspektor Gadget.

Inspektor Gadget

Presentation

A swiss army knife based on eBPF [5]:

- `local-gadget`
- `kubectl-gadget`



Various:
trace capabilities
trace dns
top ebpf

Applications

System Libraries

System Call Interface

VFS

File Systems

Volume Manager

Block Device

Sockets

TCP/UDP

IP

Ethernet

Scheduler

Virtual Memory

Clocksource

Device Drivers

Firmware

trace fsslower
trace mount

top file

top block-io

profile block-io

trace network

trace sni

trace bind

top tcp

profile cpu

snapshot socket

trace exec
trace open
trace signal
trace tcp
trace tcpconnect

snapshot process

trace oomkill

The different tools offered by Inspektor Gadget.

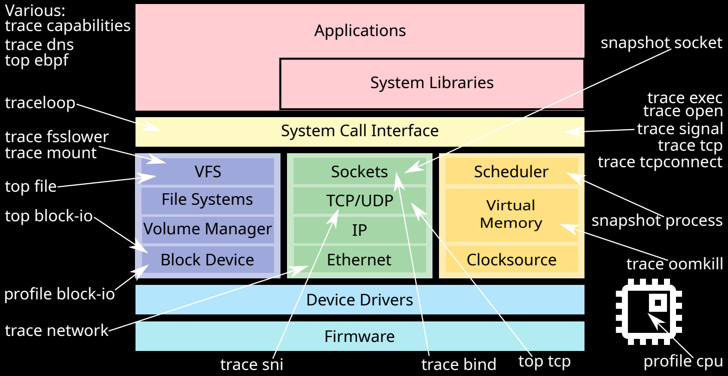
Inspektor Gadget

Presentation

INSPEKTOR GADGET

A swiss army knife based on eBPF [5]:

- `local-gadget`
- `kubectl-gadget`



The different tools offered by Inspektor Gadget.

Inspektor Gadget

Short demo

Comparing `local-gadget trace exec` to `execsnoop` [6].

Inspektor Gadget

What is eBPF?

According to Brendan Gregg [7]:

> *eBPF does to Linux what JavaScript does to HTML. [...] [W]ith eBPF, instead of a fixed kernel, you can now write mini programs that run on events like disk I/O, which are run in a safe virtual machine in the kernel.*

What is eBPF?

According to Brendan Gregg [7]:

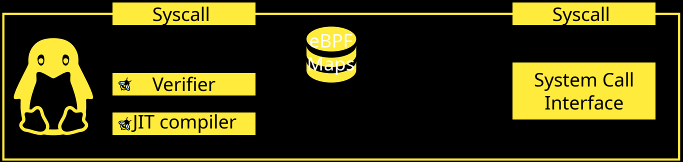
> *eBPF does to Linux what JavaScript does to HTML. [...] [W]ith eBPF, instead of a fixed kernel, you can now write mini programs that run on events like disk I/O, which are run in a safe virtual machine in the kernel.*

eBPF programs safety comes with some limitations, among others:

- It is impossible to write an infinite or a not statically bounded loop.
- There is no function like `malloc`.

Inspektor Gadget

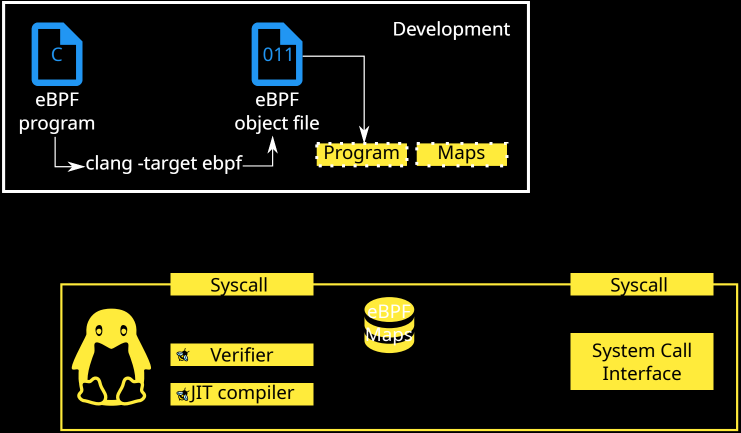
eBPF in the Linux kernel



Development, loading and execution of an eBPF program [8].

Inspektor Gadget

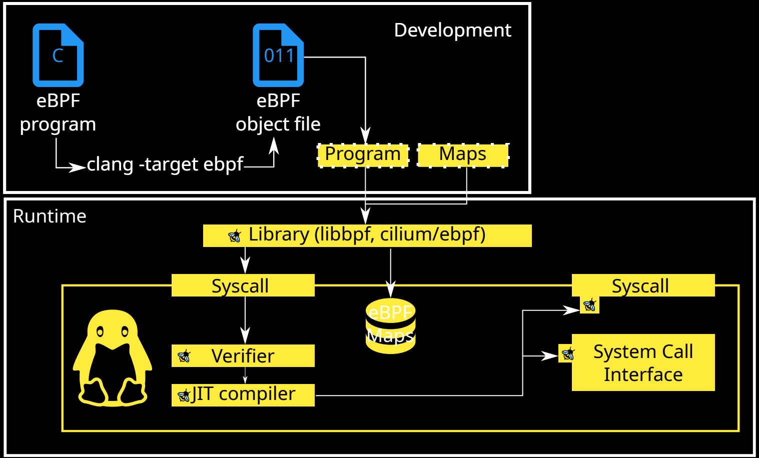
eBPF in the Linux kernel



Development, loading and execution of an eBPF program [8].

Inspektor Gadget

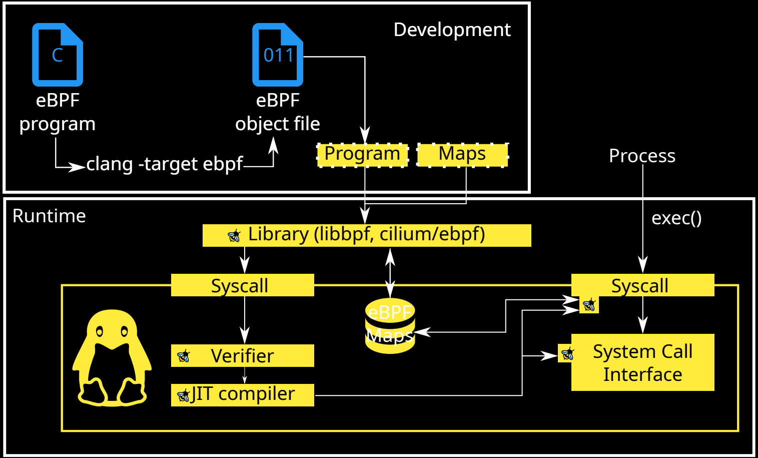
eBPF in the Linux kernel



Development, loading and execution of an eBPF program [8].

Inspektor Gadget

eBPF in the Linux kernel



Development, loading and execution of an eBPF program [8].

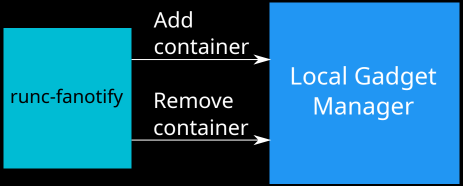
Inspektor Gadget

Internal architecture of `local-gadget`

Local Gadget Manager

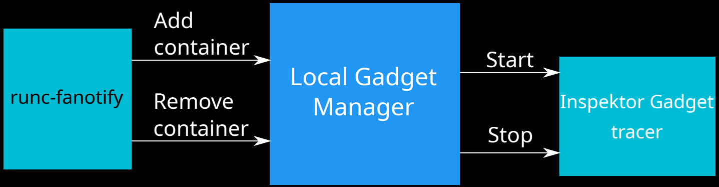
Inspektor Gadget

Internal architecture of `local-gadget`



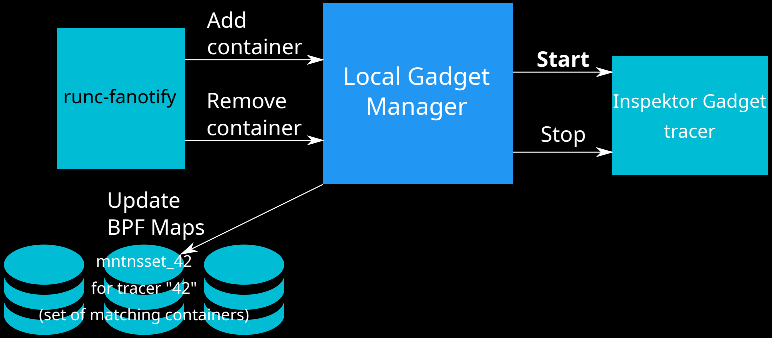
Inspektor Gadget

Internal architecture of `local-gadget`



Inspektor Gadget

Internal architecture of `local-gadget`



runc-fanotify

Add container

Remove container

Local Gadget Manager

Start

Stop

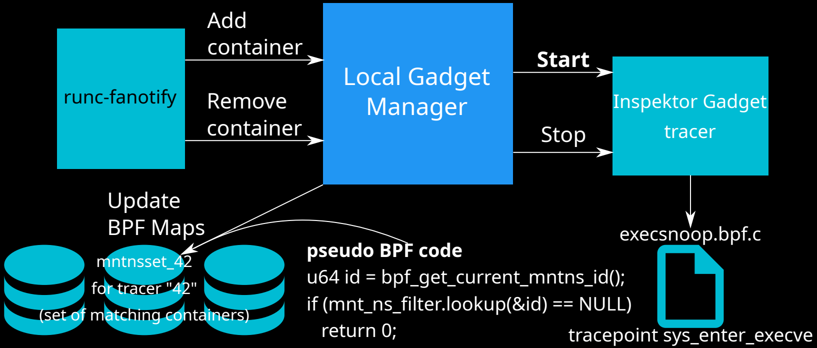
Inspektor Gadget tracer

Update BPF Maps

mntnsset_42 for tracer "42" (set of matching containers)

Inspektor Gadget

Internal architecture of `local-gadget`



pseudo BPF code

```
u64 id = bpf_get_current_mntns_id();
if (mnt_ns_filter.lookup(&id) == NULL)
  return 0;
```

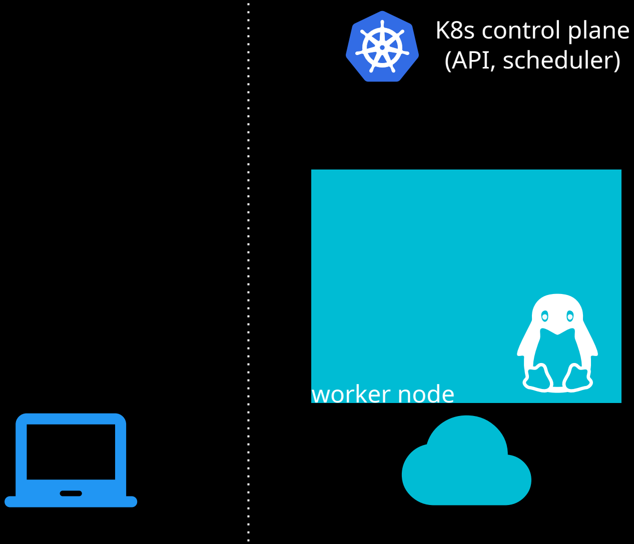
Inspektor Gadget

Real world demonstration of `local-gadget`

How to use `local-gadget` to verify `seccomp` profile?

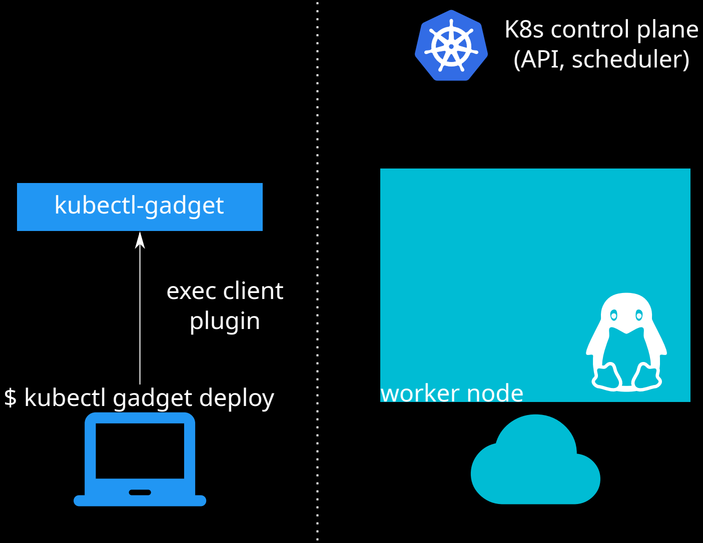
Inspektor Gadget

In Kubernetes



Inspektor Gadget

In Kubernetes



K8s control plane
(API, scheduler)

kubectl-gadget

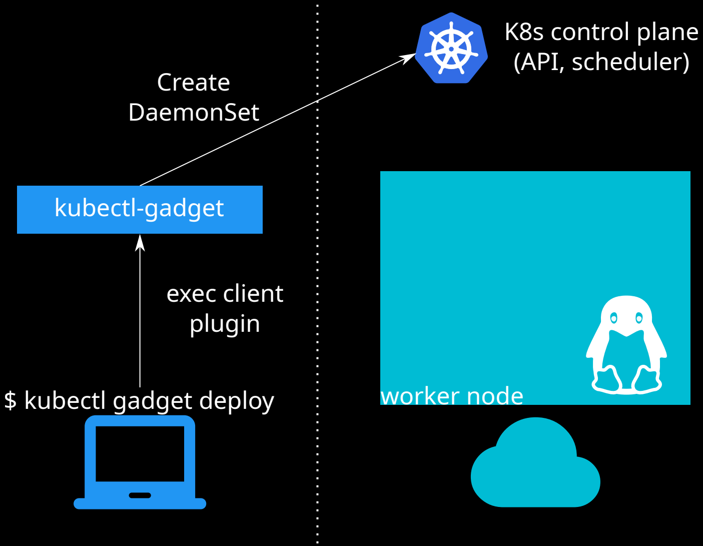
exec client
plugin

$ kubectl gadget deploy

worker node

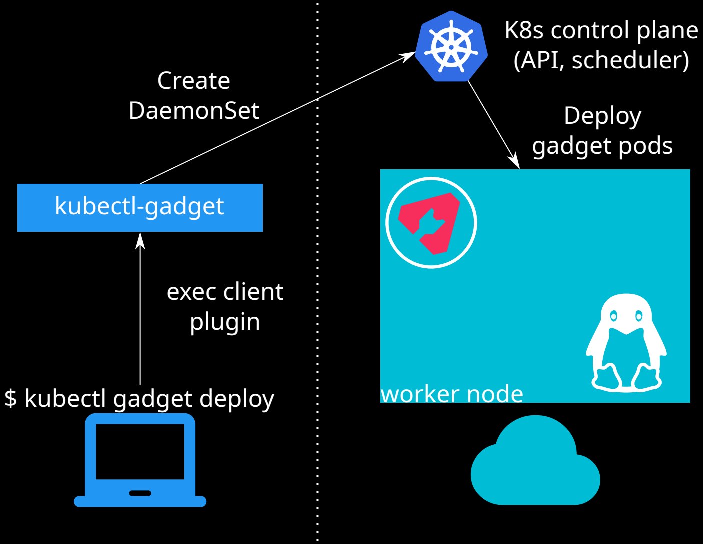
Inspektor Gadget

In Kubernetes



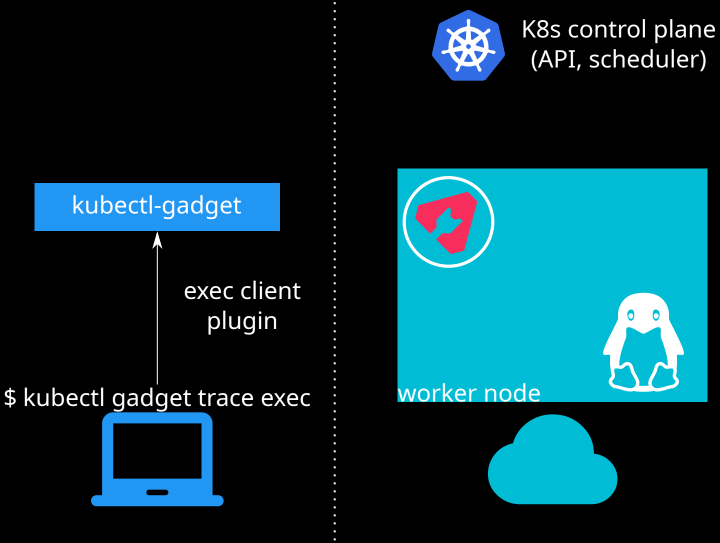
Inspektor Gadget

In Kubernetes



Inspektor Gadget

In Kubernetes



K8s control plane (API, scheduler)

kubectl-gadget

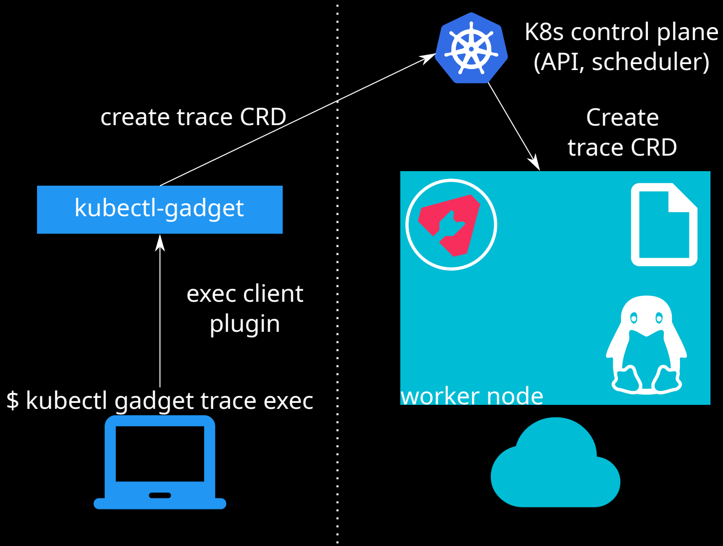
exec client plugin

```
$ kubectl gadget trace exec
```

worker node

Inspektor Gadget

In Kubernetes



K8s control plane
(API, scheduler)

create trace CRD

Create
trace CRD

kubectl-gadget

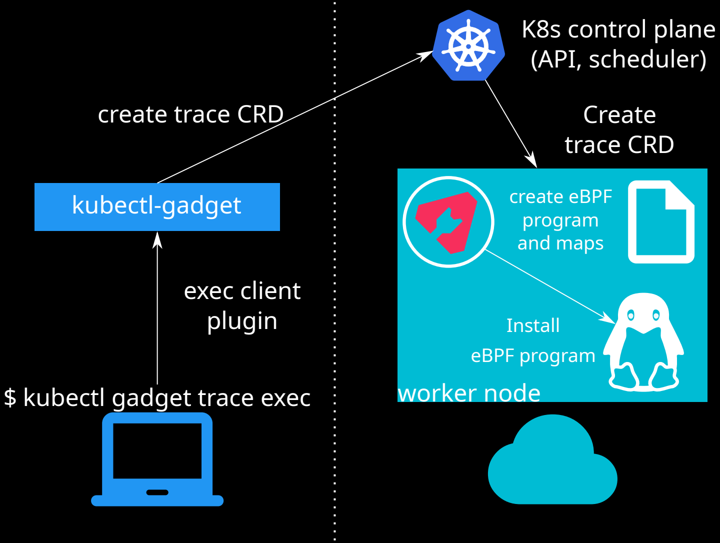
exec client
plugin

worker node

$ kubectl gadget trace exec

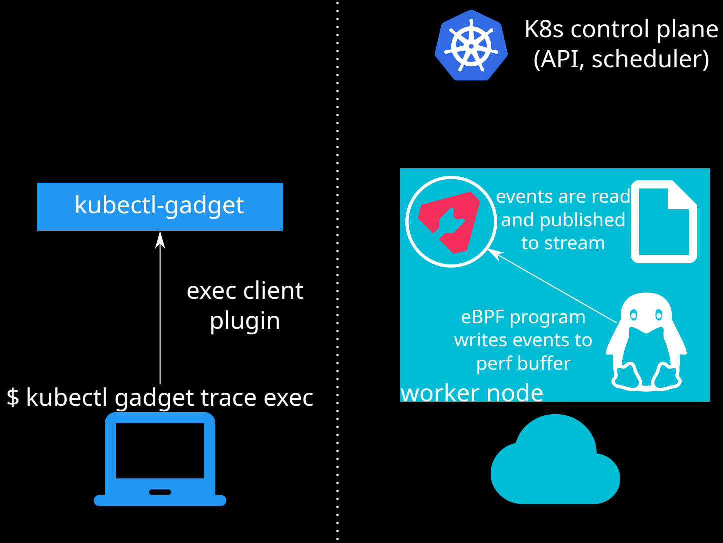
Inspektor Gadget

In Kubernetes



Inspektor Gadget

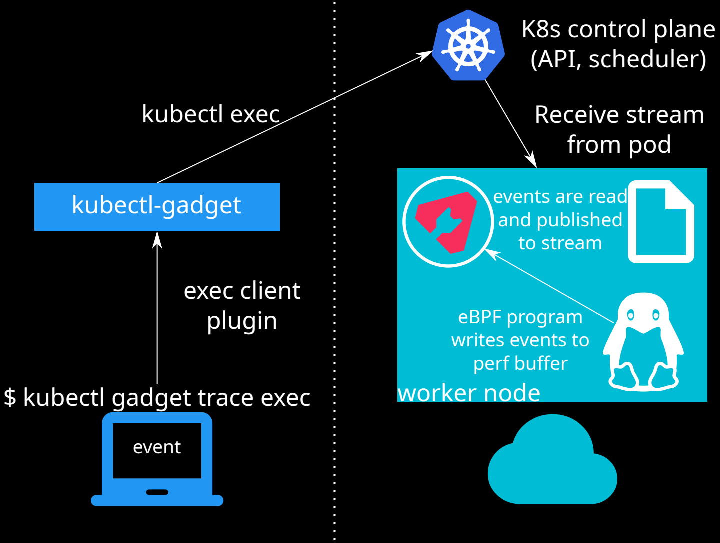
In Kubernetes



K8s control plane
(API, scheduler)
kubectl-gadget
exec client plugin
$ kubectl gadget trace exec
events are read and published to stream
eBPF program writes events to perf buffer
worker node

Inspektor Gadget

In Kubernetes



Inspektor Gadget

Real world demonstration of `kubectl-gadget`

How to use `kubectl-gadget` to verify the containers capabilities?

Conclusion and future works

Conclusion:

1. Inspektor Gadget permits monitoring of containers.
2. It is of precious help to debug these applications.

Conclusion and future works

Conclusion:

1. Inspektor Gadget permits monitoring of containers.
2. It is of precious help to debug these applications.

Future works:

1. Improve scaling.
2. Addition of new gadgets.

Conclusion and future works

Conclusion:

1. Inspektor Gadget permits monitoring of containers.
2. It is of precious help to debug these applications.

Future works:

1. Improve scaling.
2. Addition of new gadgets.

Where to find us:

- `inspektor-gadget.io`
- `github.com/inspektor-gadget/inspektor-gadget`
- `#inspektor-gadget (k8s slack)`

Bibliographie I

[1] Rami Rosen, “Namespace and cgroups, the basis of Linux containers,” Seville, Spain, Feb. 2016. [Online]. Available: https://www.netdevconf.org/1.1/proceedings/slides/rosen-namespaces-cgroups-lxc.pdf

[2] M. Kerrisk, “LCE: The failure of operating systems and how we can fix it,” Nov. 2012, publication Title: LWN. [Online]. Available: https://lwn.net/Articles/524952/

[3] ——, “Namespaces in operation, part 1: namespaces overview,” Jan. 2013, publication Title: LWN. [Online]. Available: https://lwn.net/Articles/531114/

[4] K. Hiroyu, “Cgroup And Memory Resource Controller,” Nov. 2008. [Online]. Available: https://www.static.linuxfound.org/jp_uploads/seminar20081119/CgroupMemcgMaster.pdf

[5] Inspektor Gadget contributors, “Inspektor Gadget.” [Online]. Available: https://github.com/inspektor-gadget/inspektor-gadget/

[6] iovisor/bcc contributors, “execsnoop.” [Online]. Available: https://github.com/iovisor/bcc/blob/master/libbpf-tools/execsnoop.bpf.c

[7] B. Gregg, “Learn eBPF Tracing: Tutorial and Examples,” Jan. 2019. [Online]. Available: https://www.brendangregg.com/blog/2019-01-01/learn-ebpf-tracing.html

[8] eBPF contributors, “What is eBPF?” [Online]. Available: https://ebpf.io/what-is-ebpf