CLOUD NATIVE

eBPF DAY

NORTH AMERICA

Who Needs an API Server to Debug a Kubernetes Cluster?

Jose Blanquicet

Who Needs an API Server to Debug a Kubernetes Cluster?

CLOUD NATIVE eBPF DAY NORTH AMERICA

October 24, 2022 | Detroit, MI

Jose Blanquicet
Senior Software Engineer
Microsoft

How would you debug a Kubernetes cluster if the API server goes down?

Agenda

- Demo #1: Debug an API server issue **using BCC and standard Linux tools**
- Introducing **Local Gadget** project
- Demo #2: Debug an API server issue **using Local Gadget**
- **The future** of Local Gadget (Roadmap)

Demo #1: Debug an API Server issue using BCC and standard Linux tools

Demo #1: What issues did we find?

- Need to **manually** retrieve container information (PID1, namespaces, etc.).
- **Extracting/Filtering** the data of interest is difficult.
- Switching between Linux namespaces to run tools in the **correct context**.

Local Gadget

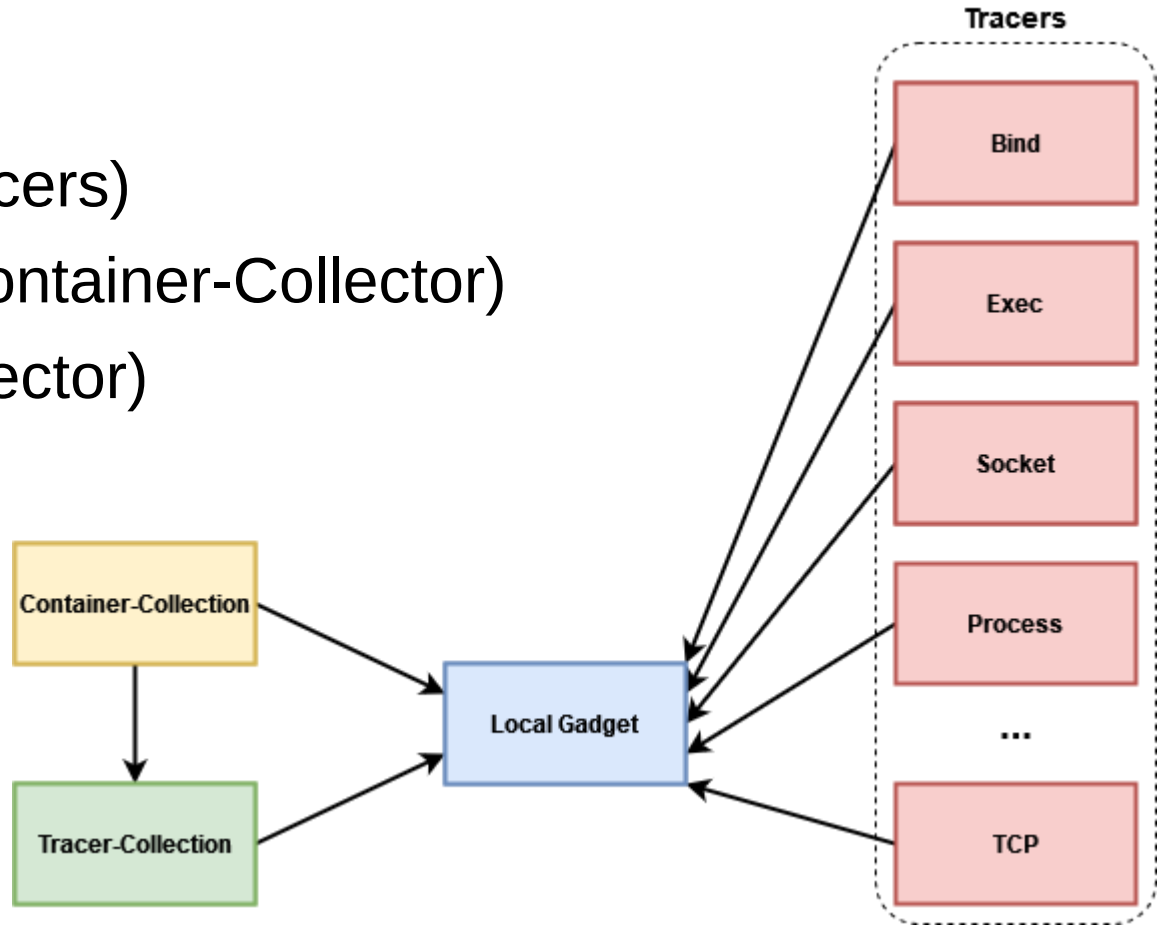
Local Gadget: What is it?

- It is a **single binary (statically linked)**.
- Allows you to trace local containers using **eBPF**.
- Enriches events with **Kubernetes metadata**.
- Can be used for trace Kubernetes and **non**-Kubernetes containers.
- Available tools (or “gadgets”): Some based on **BCC tools** (e.g., trace bind, exec, open events), as well as some developed by our team for other use cases (e.g., snapshot processes and sockets, trace DNS events, audit seccomp policies).

Local Gadget: Architecture

Three main tasks:

- Collect insights (Tracers)
- Data enrichment (Container-Collector)
- Filtering (Trace-Collector)



Collect insights (Tracers)

We wrote the control plane in Go, so that it can be easily used/integrated:

```go
func main() {
    if err := rlimit.RemoveMemlock(); err != nil {
        return
    }

    eventCallback := func(event execTypes.Event) {
        fmt.Printf("A new %q process with pid %d was executed\n",
            event.Comm, event.Pid)
    }

    tracer, err := execTracer.NewTracer(
        &execTracer.Config{},
        nil,
        eventCallback,
    )
    if err != nil {
        fmt.Printf("creating tracer: %s\n", err)
        return
    }
    defer execTracer.Stop()

    exit := make(chan os.Signal, 1)
    signal.Notify(exit, syscall.SIGINT, syscall.SIGTERM)
    <-exit
}
```

```go
func NewTracer(
    config *Config,
    enricher gadgets.DataEnricher,
    eventCallback func(types.Event),
) (*Tracer, error) {
```

```go
type Config struct {
    // Filtering
    MountnsMap *ebpf.Map
}
```

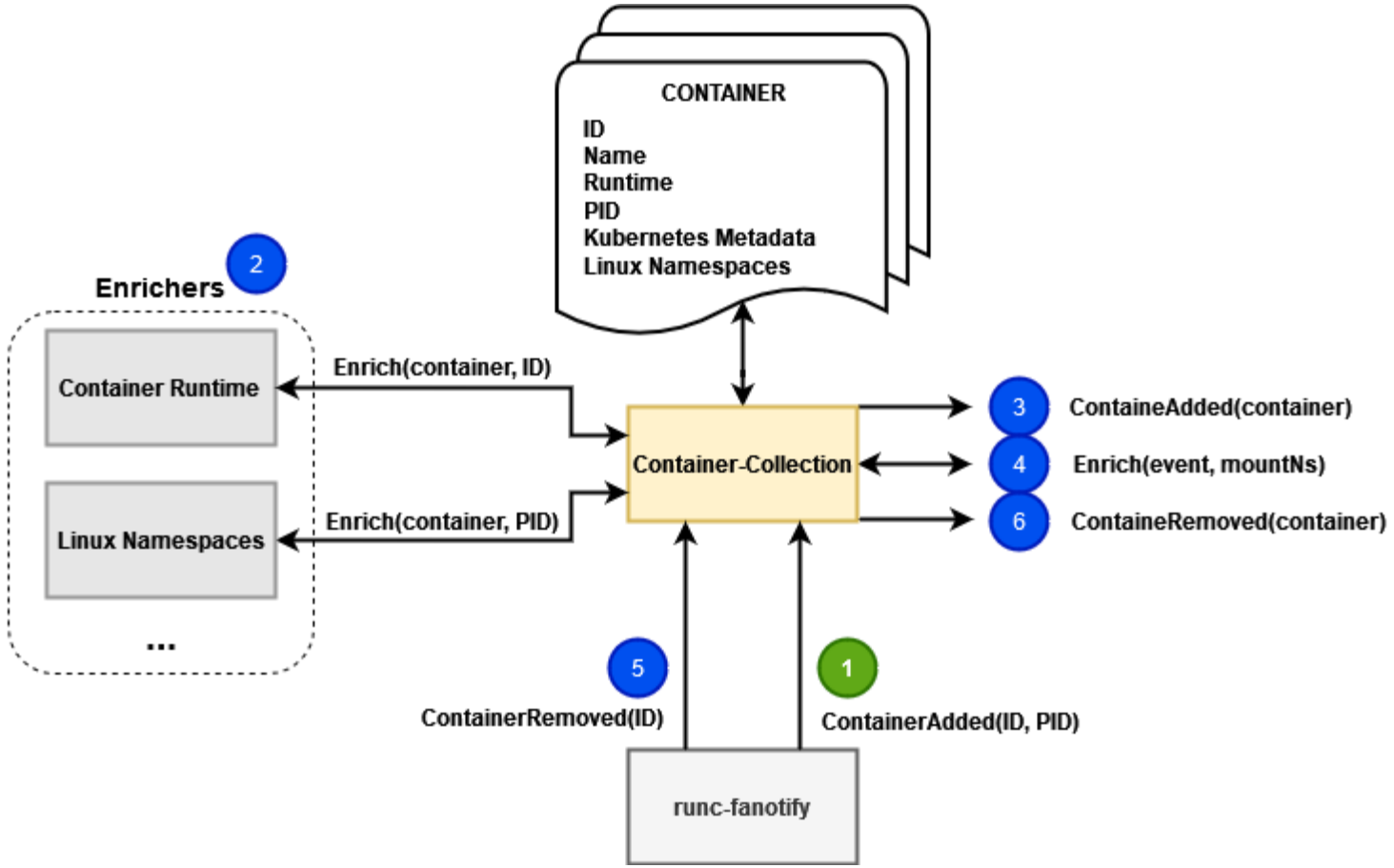
```go
type Event struct {
    types.CommonData

    Pid       uint32
    Ppid      uint32
    Comm      string
    Retval    int
    Args      []string
    UID       uint32
    MountNsID uint64
}
```

```
$ go build -o exec .
$ sudo ./exec
A new "calico" process with pid 118594 was executed
A new "portmap" process with pid 118606 was executed
A new "bandwidth" process with pid 118611 was executed
A new "runc" process with pid 118616 was executed
A new "docker-init" process with pid 118623 was executed
^C
```

Data enrichment (Container-Collection)

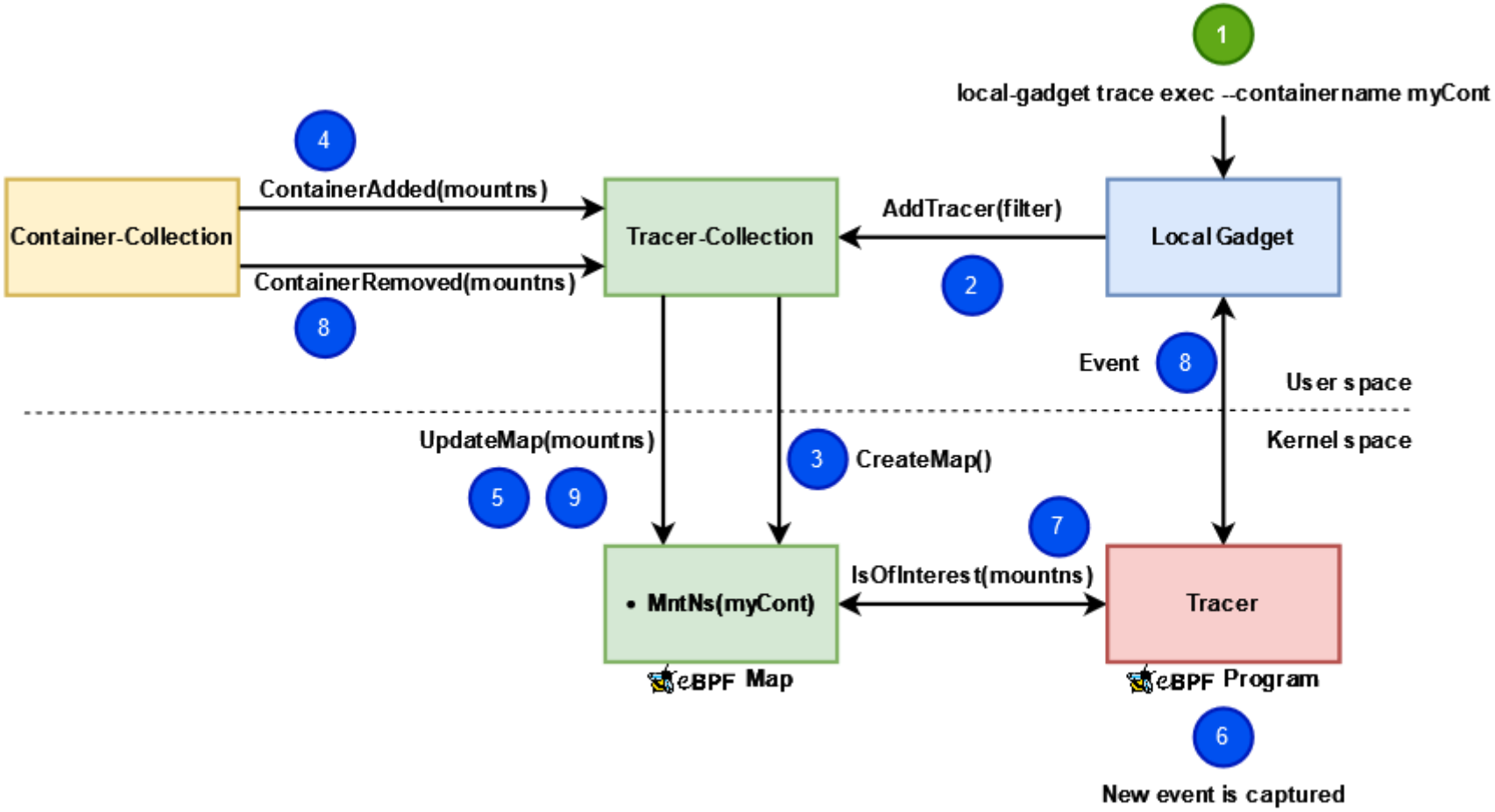
- **Enriches** events.
- Notifies about **container creation/deletion**.
- Get Kubernetes info from the **Container Runtime**:
  - Docker through the Engine API.
  - Containerd and CRI-O through the CRI.



Filtering (Trace-Collection)

CLOUD NATIVE eBPF DAY NORTH AMERICA

Manage eBPF maps for filtering



Local-Gadget: Internal modules

Do you want to know more about these components?

- Blog: https://www.inspektor-gadget.io/blog
- Examples: https://github.com/kinvolk/inspektor-gadget/tree/main/examples

Local Gadget: Use-Cases

- Debug **without API server** is difficult.
- You are implementing a tool that needs to **get insights from the node**:
  - Include the local-gadget binary in your container image, and your app simply execs local-gadget (JSON format).
  - If your app is in Go, you can run our tracers using the packages we created.
- Observing and debugging containers also **outside** Kubernetes environments.

Demo #2: Debug an API server issue **using Local Gadget**

Notes from Local Gadget demo

- Debug Kubernetes containers even if the API server is down.
- Enrichment of Kubernetes metadata.
- No manual steps for filtering.
- Don’t lose any event at container startup.

The future of Local Gadget

- Support **filtering by Kubernetes resources**: `--k8s-namespace, --k8s-pod, --k8s-container`.
- Support **non-Kubernetes** containers created by other container runtimes.
- Continue adding **more and more gadgets** … Is there a use-case where you think Local Gadget could be useful? **Reach us**:
  - Repository: https://github.com/kinvolk/inspektor-gadget
  - Slack: https://kubernetes.slack.com/messages/inspektor-gadget/

Thanks!