Contribfest: Enhancing Kubernetes Debugging and Observability with Inspektor Gadget

KubeCon NA 2024

Jose Blanquicet

Mauricio Vasquez

Environment Setup

- We have few virtual machines ready for you
- Join our Slack channel and write to **Maya Singh** to get your credentials
- If you don’t have Slack, talk to Maya in this room to get them
- https://github.com/inspektor-gadget/inspektor-gadget

Agenda

- Introduction to Inspektor Gadget
- Environment Setup
- Using Gadgets
- Creating Gadgets
- Open discussion time
- Closing remarks

Inspektor Gadget

A batteries-included tool for eBPF observability

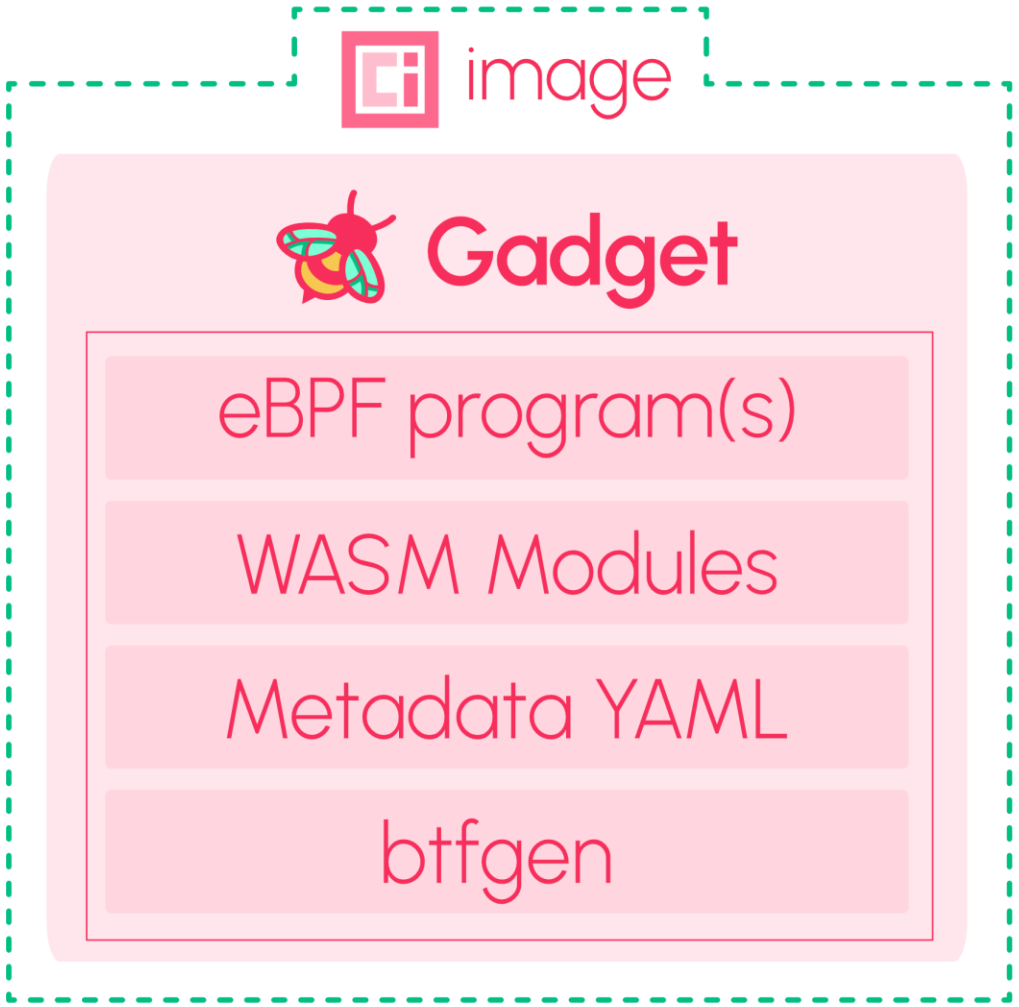
Inspektor Gadget is a set of tools and framework for data collection and system inspection on Kubernetes clusters and Linux hosts using eBPF

Inspektor Gadget’s key features

- Build and package eBPF programs into OCI images called Gadgets
- Distributes and manages eBPF programs across your cluster
- Collect and export data to observability tools with a simple command and declarative configuration
- Security mechanisms to restrict and lock-down which Gadgets can be run
- Automatic enrichment: map kernel data to high-level resources like Kubernetes and container runtimes
- Supports WebAssembly modules to post-process; using any WASM-supported language
- Supports many modes of operation; cli, client-server, API, embeddable via Golang library

Gadgets

- eBPF: Gather information from the kernel
- WASM: Post process information
- Metadata: Describes how the gadget behaves
- Btfgen: Compatibility with older kernels



Environment Setup

- We have few virtual machines ready for you
- Join our Slack channel and write to **Maya Singh** to get your credentials
- If you don’t have Slack, talk to Maya in this room to get them
- https://github.com/inspektor-gadget/inspektor-gadget

Using Gadgets

Contribfest repository

https://github.com/inspektor-gadget/kubecon-na-2024/

Creating Gadgets

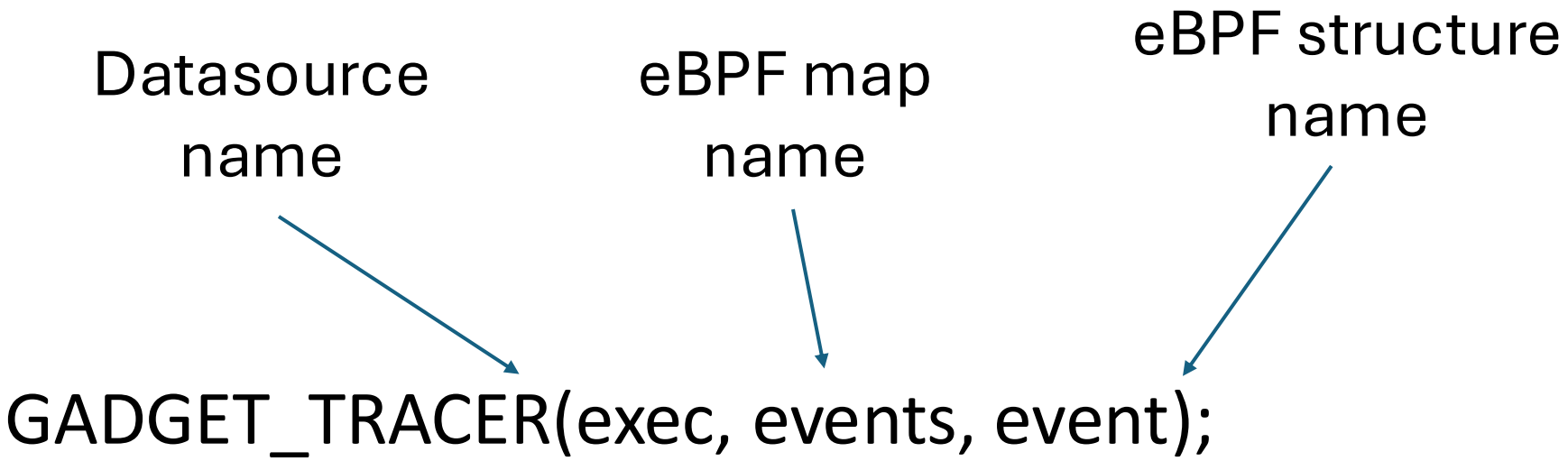
Gadgets: Deep Dive

- Datasources
- eBPF Gadget helper API
- Metadata
- Building
- Running

Datasources

- Indicate Inspektor Gadget how to gather and show information provided by Gadgets
- There are different kinds
  - Tracer: Provides a stream of events
  - Map Iter: Saves statistics (metrics) in a hash map
  - Snapshot: Provides a list of things
  - More coming later
- Registered by using macros
- They can be configured (annotated) by the gadget author or user
- More information here

Datasources



Gadget eBPF Helper API

- Set of functions and types provided by Inspektor Gadget
- Functions
  - Filling common information
  - Filtering events by container information
  - Correlating process information from a socket object
- Enriched types
  - Container enrichment
  - Human readable representation of a timestamp
  - String representation for an IP address
  - Many more
- More details here

Metadata

- The metadata file contains extra information about the gadget
  - Name of the gadget, description, etc.
  - Enable metrics collection
- It's used to configure how data is formatted
  - Print this field aligned to the right
  - Hide this field from the output
- It's optional
- More info here

Hands on

- Follow the guide and ask any questions you might have

Open Discussion

Closing Remarks

Closing Remarks

- We're happy to hear about your use cases
- Please reach out to us on Slack