Kubernetes... Minus the Blind Spots

Solving K8s Observability Challenges with Real-Time Topology

Conf42 DevOps January 2025



Today's Roadmap

- Kubernetes is great (until you have to troubleshoot)
- Challenges with the traditional observability toolkit
- Real-time topology mapping: what, why, how
- Real-world use cases
- eBPF as real-time topology enabler
- Getting started





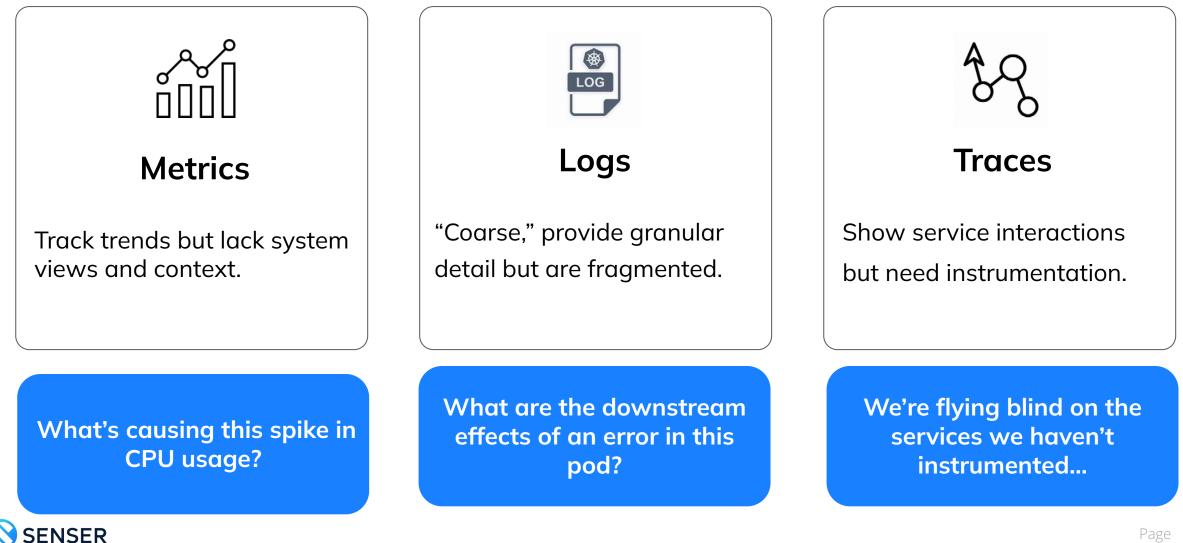
Kubernetes is great (until you have to troubleshoot)

- Blurred line between app and infra issues
- Ephemeral environments
- Complex networking (CoreDNS, kube-proxy, CNI plugins, etc.)
- Shared resources and noisy neighbors
- Black box containers

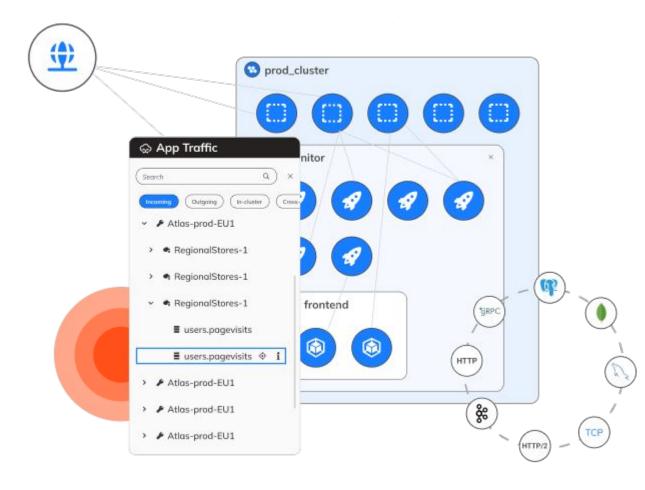
Automation complexity

& Deployment) fro	ontend			•tg _		
ime Range: Sep 18	nly applies to this	s pane		0 Indirect Logs Events Deployments		
occurs of medica	ficult (1993)	TESTION, ITAIN	c ones oe	e manoco caga carena ocpanyinana		
TIMESTAMP	POD	CONTAINER	LEVEL	MASSAGE		
18.9.2024 09:16:57.397	frontend- 56d8b595d8- 9tqpx	server	error	1802, serverEndpoint: redis-cart-6379, conn-sec: tv2. aoc: 0, mc: 1/1/0, mgr. 10 of 10 available, client Name: cartservice-7989787b95-k6hvx;SE.Redis-v2.6.122.38350), IOCP: (Busy-0,Free-1000.Min-1, Max-1000), WORKER: (Busy-1,Free-32766,Min-1,Max-32767), POOL: (Threads=6,QueuedItems- O,CompletedItems-376638543/Timers=6), v2.6.122.38350 [Please take a look at this article for so me comman client-side issues that can cause timeouts: https://stackexchange.github.io/StackExchan ge.Redis/Timeouts/Iv/u003e StackExchange.R Show Mare:		
18.9.2024 09.16:57.397	frontend- Sédöb595d8- 9tqpx	server	error	[Percel*Could not retrieve cart: type error: cade - FailedPrecondition desc - Can't access cart storag a. StackExchange Redis RedisConnectionException: The message timed out in the backlog attamptin gto send bacause no connection became available (5000ms) - Last Connection Exception: It was no t possible to connect to the redis server(s). ConnectTimeout, command-HMGET, timeout 5000, inst (o, cu: 3, cg: 0, oux: False, bux: CheckingFoTTimeout, command-HMGET, timeout 5000, inst out; o, cu: 0, cg: 0, oux: False, bux: CheckingFoTTimeout, command-HMGET, timeout 5000, inst (Julio, mgr: 10 of 10 available, clientName: cartservice-7989787b95-k6hws/SE.Redis-v26.122.38350 (7), IPOCL: (Threads=6),QueuedItems=1.CompletedItems=378638534,Timers=7), v2.6.122.38350 (7), IPOCL: (Threads=6),QueuedItems=1.CompletedItems=378638554,ChengeRedisRedisCohes actionException: It was not possible to connect to the redis server(s), ConnectTimeouth= End of 1 nert acception: It was not possible to connect to the redis server(s), ConnectTimeouth= End of 1 nert acception: It was not possible.RedisCohes ethandRefreahAsync(String key, Boalean getData, ConcellationToken token)) at Microsoft.Extensions Caching StackExchangeRedisCartStore GetCartAsync(String key, ConcellationToken token) in a tar secture.actatore RedisCartStore GetCartAsync(String key, ConcellationToken token) in a tar secture.actatore RedisCartStore GetCartAsync(String key, ConcellationToken token) in a tar secture.actatore RedisCartStore GetCartAsync(String key, ConcellationToken token) in a tar secture.actatore		
18.9.2024 09:16:56.396	frontend- 56d8b595d8- 9tqpx	server	unknown	("http:req.id": '5/21fee8-5184-4734-8/2a-cebe8ccf6cdd","http:req.method": 'GET', "http:req.path": '/pr oduct/22YFj3GM2N', 'http:res.pbytes', '6166, 'http:res.ptatus', '500, 'http:res.ptatok, me", '5299, 'messa ge": 'request complete', 'session'.' co6cbd1b-1669-4bc7-9452-8c1767o48b8a', 'severity'.''debug", 'ti mestame'', '2024-09-1870: 1556-3969388282')		
18.9.2024	frontend- 56d8b595d8-	server	error	['error ^{1,1} 'could not retrieve cart: rpc error: code – FailedPrecondition desc – Can't access cart storag e. StackExchange Redis RedisConnectionException: The message timed out in the backlog attemptin g to send bacause no connection became available (5000ms) – Last Connection Exception: It was no t possible to connect to the redis server(s). ConnectTimeout, command=HMGET, timeout: 5000, inst 0, qu. 5, qu. 0, av: False, bw: CheckingForTimeout, last-in: 0, sum: op:: 0, sign: op:: 2232 1802, server Indigent: redis-artis379, com-sec: nda, aoc: 0, mc: 1/10, mg; 10 of 10 available, client Neuron contention of the standard statement of the sta		

Challenges with the traditional observability toolkit



Real-time topology provides the context required to eliminate blind spots and troubleshoot, fast.



What it is

A live map of system relationships (pods, services, networks)

Why it matters

Connect individual data points into a full-system view

How it works

Continuously updates to reflect real-time changes (every new service, every deployment)



Real-time topology provides the context required to eliminate blind spots and troubleshoot, fast.

	-K8s 🕶 🔟
Aug 27, 2024, 2:07 PM - Aug 27, 2024, 2:08	🗄 Camponents 🔰 Protocol overlay: 🗉 📖 💷 🌒 🌒 🕼 🍪 🐨 💌 📵 🔥 😨 🚫 🤫 🎯
. NA KA AMA KA KA KAKA KA KA KA KA KA KA KA KA KA	人名英格兰英格兰英格兰英格兰英语 医马克尔 医马克尔 医马克尔氏的
t ne na nationa na amin'ny faritr'ora amin'ny faritr'ora amin'ny faritr'ora amin'ny faritr'ora amin'ny faritr'o	t na sa sana na sa sana sa sa sa sa sa sa sa sa sa
人名英格兰 医外周 化分离子 医结核 医子宫	(我这些书书的话,我的是我的话,我们的书书的书书,我们的书书的书书。
, na ha kawa ka ka ka kuna ka ka kawa maka a ka k	计标准 电环 电调频 电波 医结果 经济 电连接 医结核 医液体的 医神经神经炎
计数据 化合物 网络鼻科 医结束 化化化物 化化物化物 化化化物化化物 化化化物化化物 化化物化物 化化物	人名英格里 网络黑卡属 医海绵管 医黑毛菌 医输出 医医子宫 医副子宫的
, na na mang na panamana na na malaha na na mana na na na na hasa hasa na na	
r na na amin'ny kaodim-paositra 1970. Ny kaodim-paositra na kaodim-paositra 60 kaodim-paositra 60 kaodim-paosit	
	i na sa sana na sa sana sa sa sa ka ka ka 🚺 🛁
Cluster	
Cluster	External
AWS-ECOMMERCE	
	· · · · · · · · · · · · · · · · · · ·
Cluster	
Cluster	
	Inter
	Inter
AZURE-ECOMMERCE	· * * * * * * * * * * * * * * * * * * *
计数据 化过度 國際 化过度 化试验剂 化活动过度 化进行放出 化过度 化过度 化过度 化过度 化过度 医碘化合物 化过度过	
	人名法 医铁 医颈体 标准 医颈 医关系 化法 医颈法 化法 化准 电弧 医颌的



Use case #1: Probe readiness failure

The scenario

A stateful application in Kubernetes fails readiness probes due to high disk I/O, leading to pods being repeatedly restarted by the kubelet. Without proper visibility, it's difficult to correlate the high disk usage with other factors like database queries or backup operations.

How real-time topology helps

Mapping pod dependencies and resource usage, enabling teams to identify the root cause of the disk I/O spike and adjust the probe configurations or workload scheduling.



cartservice e Initial () 24.05.2023 08:42:28 - 08:53:21							
aluation modelity Time orsdow 39% 28d	Related changes a Component Name uhippingsentice	nd deployments Deployment time Prinutou	Deployment Type New container image	Image Version 1.15 - 1.16			
mpilanaa ne wixidaw Eroc budget Recoolining Taine To Depiletien 98 2% 1.8% 304 millio I nome Availetsility	Impact Map			2 Impact			
Prod_cadar	(BPod) cartservice		ndegeneral) Addiension I	StD Checkent			

Use case #2: Throttling in autoscaling

The scenario

An application under high traffic triggers Horizontal Pod Autoscaling (HPA), but new pods experience throttling due to CPU limits set too low. This slows down traffic handling and results in degraded performance.

How real-time topology helps

Visualizing resource allocation and pod scaling behavior, enabling teams to adjust resource requests and limits to align with the workload and improve scaling efficiency.

Cartservice Initial () 24.05.2023 08:42:28 - 08:53:21							
ervice Level Objective i station method Time orders 39% 288	Related changes a Component Name uhppingsarvice	nd deployments Deployment time Preinatos	Deployment Type New container image	Image Version 1.15 - 1.14			
mpRanas Intervention Error budget Reenairing Time To Depiction 88 88.2% 1.8% 304 mins	Impact Map			0			
J nome Availability				Impact			
opology				(SLC) Checkout			
Poul Akatar	(@Pod) cartservice		npleprent) Addervice 1				
0000				(ZSerska) internelCort			



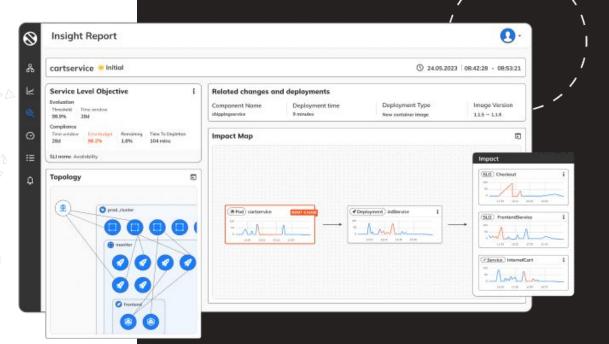
Use case #3: Misconfigured network policies

The scenario

A microservice deployed in Kubernetes fails to communicate with its dependent services due to overly restrictive Network Policies. The lack of detailed logs or visibility into blocked connections makes it difficult to diagnose the issue.

How real-time topology helps

Mapping service-to-service communication paths and highlighting blocked connections, enabling teams to update Network Policies and restore communication between services.





What enables real-time topology



مر،



What is **eBPF**?

eBPF (extended Berkeley Packet Filter) is a Linux kernel feature that allows safe, low-level instrumentation and monitoring.





- **Traffic analysis**: Trace network requests, messages, and data flows.
- Infra, App & Performance monitoring: Observe system calls, resource usage, and latency in real-time.



Security auditing: Detect anomalies or unauthorized access at a granular level.

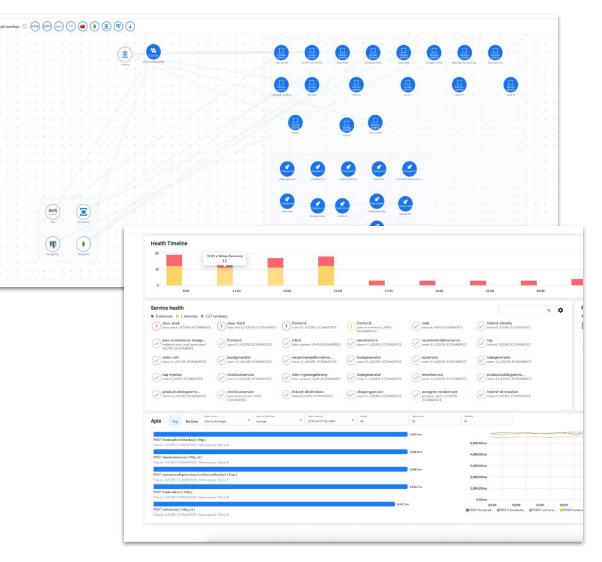
Why it matters

Enables **deep observability** directly from the kernel, making it invaluable for **complex**, **distributed environments**.



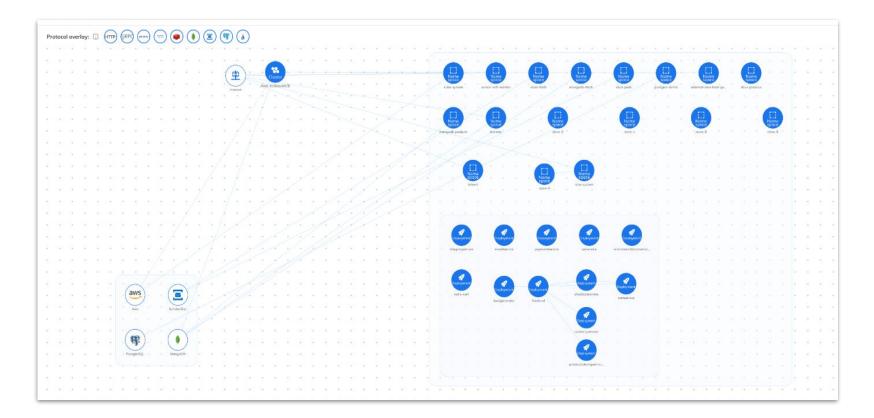
Why eBPF is ideal for distributed systems

- **Low-latency monitoring**. Captures and displays real-time insights without hindering system performance.
- **Enhanced visibility**. Sees all network interactions and system calls to provide comprehensive insights.
- Improved efficiency. Reduces need for multiple diagnostic tools, centralizing observability with eBPF.
- **Scalability**. Adapts to cloud-native and microservices architectures, providing a consistent observability approach.





Auto-discovery and mapping of your environment



Al capabilities that use Kubernetes metadata and traffic inspection to discover dependencies and create a dynamic service graph – without requiring manual updates.



Common pitfalls (and how to avoid them)

- Incomplete instrumentation: Missing critical data like kubelet logs, API server events, or container runtime metrics leaves significant gaps in the topology, reducing its usefulness. (Technologies like eBPF can help.)
- Stale topology maps. Kubernetes environments are constantly changing. If topology updates aren't automatic and real-time, you risk basing decisions on outdated information.
- Assuming topology alone solves everything. Topology mapping is powerful for troubleshooting but by itself rarely a "smoking gun" in root cause analysis. (This is where AIOps shines.)



Tips for getting started

- Leverage Kubernetes metadata. Use tools that automatically pull data from Kubernetes-native sources like pod labels, ConfigMaps, and events to build accurate, context-rich maps.
- Automate topology maps. Choose solutions that dynamically update the topology to keep up with Kubernetes' rapid scaling and frequent state changes.
- Avoid context-building pitfalls. Begin with critical applications where better visibility will have the greatest operational benefit, then expand as you refine your process.





Thank you!

