TIGERA SOLUTION BRIEF

Addressing Kubernetes Observability Challenges with Calico

Kubernetes is the de facto platform to orchestrate containerized workloads and services for hybrid and multi-cloud deployments and multi-cluster environments, which are the building blocks of cloud-native applications. These Kubernetes workloads are highly dynamic, ephemeral, and are deployed on a distributed and agile cloud infrastructure; making it harder for DevOps and site reliability engineers to monitor and troubleshoot. Therefore, DevOps teams, site reliability engineers (SREs), and platform owners require better visibility into their application behavior, services, dependencies, how they are interconnected, and which applications and services access them.

While Kubernetes helps to meet the needs of deploying and managing cloud-native applications, its observability challenges require a Kubernetes-native approach.

Kubernetes Observability Challenges

Data collection, correlation, and aggregation

Disparate data collection happens at different layers (infrastructure, cluster, and application) of the Kubernetes environment, creating data silos with limited data correlation between them. Context has to be built after data collection, which requires time, effort, and resources like extra compute, memory, and storage. Also, Kubernetes creates a large number of ephemeral objects that generate data across a distributed environment; this data needs to be aggregated and correlated in order to visualize the interactions and activities in the environment. With the high volume of granular data generated in a distributed manner with each ephemeral object, this approach is not scalable.

Kubernetes context

Kubernetes adds a layer of abstraction on top of hosts and VMs. While collecting and aggregating data from individual containers and hosts is important, the data needs to be correlated and aggregated at different levels of Kubernetes abstractions.

Kubernetes policy knowledge

Kubernetes network and security policies determine access in the cluster. Real-time mapping of these policies to traffic flow in the Kubernetes cluster is critical to understanding the behavior of a deployment. Due to the dynamic and ephemeral nature of Kubernetes, traditional monitoring tools are unable to map policies and flows.

Addressing Kubernetes Observability Challenges with Calico

Kubernetes Observability with Calico

Calico addresses the above challenges by providing full-stack Kubernetes observability.

Smart in-line aggregation with Kubernetes context and policy knowledge

Calico performs smart data aggregation and correlation at the source level and packages this with Kubernetes context for live observability. A streamlined, context-driven view showing dependencies is provided—something traditional monitoring tools, which are not Kubernetes-native, cannot do.

While traditional monitoring tools might need to generate 20 logs to understand each interaction, Calico provides a single, context-rich log showing up to 40 different dependencies. Since Calico is Kubernetes-native, this information is shown during data collection, saving time, effort, and resources such as memory, compute, and network bandwidth.

Calico also provides a policy engine to define and enforce security policies for the entire Kubernetes stack. With real-time access to data and policy knowledge, Calico provides a live policy context for traffic flow at all levels of the Kubernetes stack, with Kubernetes context.

Data

- DNS Flows
- Application Flows
- Service Information
- Kubernetes Activity, Audit Logs
- Network Flows, TCP/UDP Status
- Sockets Stats
- Process Information

Policies

- Application-level policies
- Network-level policies
- DNS policies
- Firewall policies
- Load balancer policies
- Select 3rd-party firewall and load balancer policies

Addressing Kubernetes Observability Challenges with Calico

Rapid problem detection and resolution with live observability

With Kubernetes context and policy knowledge correlated to traffic, Calico provides live observability for Kubernetes environments. Calico provides a real-time view of the Kubernetes cluster, showing how workloads within the cluster communicate across namespaces, ports, and protocols, as well as how security policies are being applied.

Calico offers the user a dynamic, live mapping of data flow and security policy enforcement, as both separate and combined views. Users can preview, stage, and deploy policies across heterogeneous environments and quickly troubleshoot policy violations. Calico Cloud's live view is ideal for real-time problem detection, live troubleshooting, and violation findings.

- Dynamic Service Graph
- Application-level observability
- Dynamic Packet Capture
- Flow visualizer
- Dashboards
- Alerts

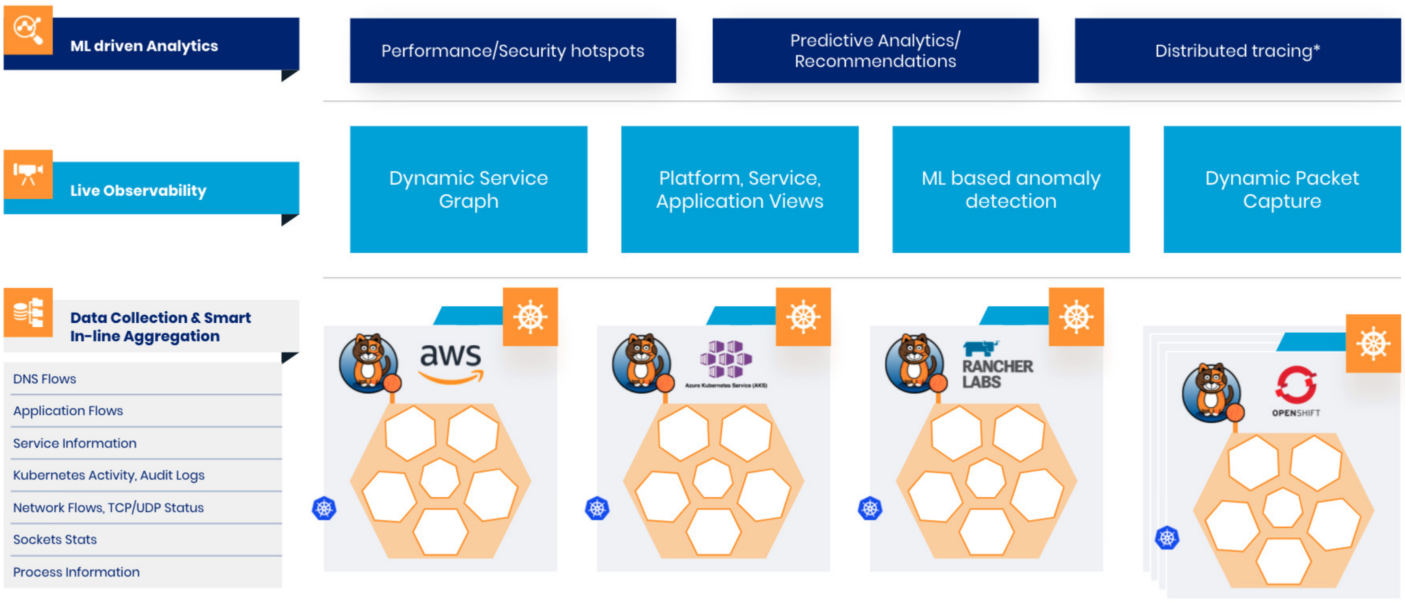
Real-time troubleshooting and prevention with ML-driven analytics

Calico provides real-time analytics to identify issues such as performance hotspots and vulnerabilities. Also, Calico uses machine learning to identify anomalies and make policy recommendations to proactively remedy the anomalies.

- Anomaly detection
- Performance & security hotspots
- Predictive analytics with recommendations

TIGERA SOLUTION BRIEF

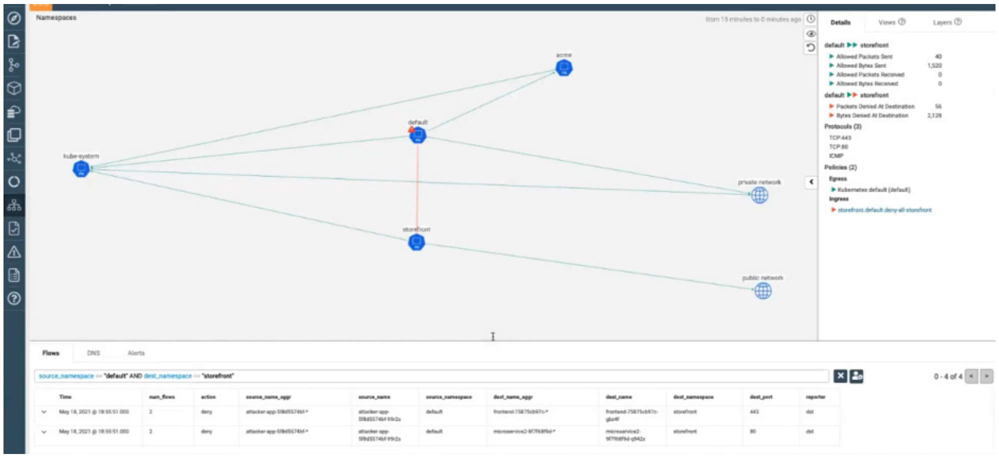
Addressing Kubernetes Observability Challenges with Calico


ML driven Analytics
Performance/Security hotspots
Predictive Analytics/ Recommendations
Distributed tracing*
Live Observability
Dynamic Service Graph
Platform, Service, Application Views
ML based anomaly detection
Dynamic Packet Capture
Data Collection & Smart In-line Aggregation
DNS Flows
Application Flows
Service Information
Kubernetes Activity, Audit Logs
Network Flows, TCP/UDP Status
Sockets Stats
Process Information
aws
Azure Kubernetes Service (AKS)
RANCHER LABS
OPENSHIFT

Calico features for Kubernetes Observability

Dynamic Service Graph

Dynamic Service Graph provides a point-to-point, topographical representation of traffic within the cluster that allows users to observe Kubernetes environment behavior and troubleshoot connectivity issues or identify performance hotspots. It enables DevOps teams, SREs, platform operators, and application developers to understand communication between namespaces, services, and deployments. The Dynamic Service Graph also includes a rich set of tools to filter resources and save views. For example, selecting a node or edge on the graph displays detailed networking and DNS activity, while automatically filtering the raw flow log data.



default ▶▶ storefront
- ▶ Allowed Packets Sent 40
- ▶ Allowed Bytes Sent 1,520
- ▶ Allowed Packets Received 0
- ▶ Allowed Bytes Received 0

default ▶▶ storefront
- ▶ Packets Denied At Destination 56
- ▶ Bytes Denied At Destination 2,128

Protocols (3)
- TCP:443
- TCP:80
- ICMP

Policies (2)
- Egress
  - ▶ Kubernetes default (default)
- Ingress
  - ▶ storefront.default.deny-all-storefront

source_namespace == "default" AND dest_namespace == "storefront"

0 - 4 of 4

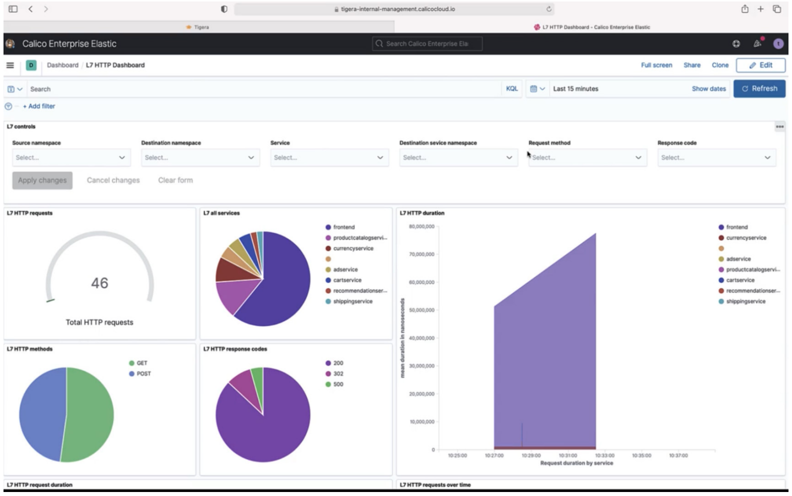
Time	num_flows	action	source_name_aggr	source_name	source_namespace	dest_name_aggr	dest_name	dest_namespace	dest_port	reporter
May 18, 2021 @ 18:55:51.000	2	deny	attacker-app-5f8d5574bf-*	attacker-app-5f8d5574bf-99r2s	default	frontend-75875cb97c-*	frontend-75875cb97c-gbz4f	storefront	443	dst
May 18, 2021 @ 18:55:51.000	2	deny	attacker-app-5f8d5574bf-*	attacker-app-5f8d5574bf-99r2s	default	microservice2-9f7f68f9d-*	microservice2-9f7f68f9d-q942x	storefront	80	dst

TIGERA SOLUTION BRIEF

Addressing Kubernetes Observability Challenges with Calico

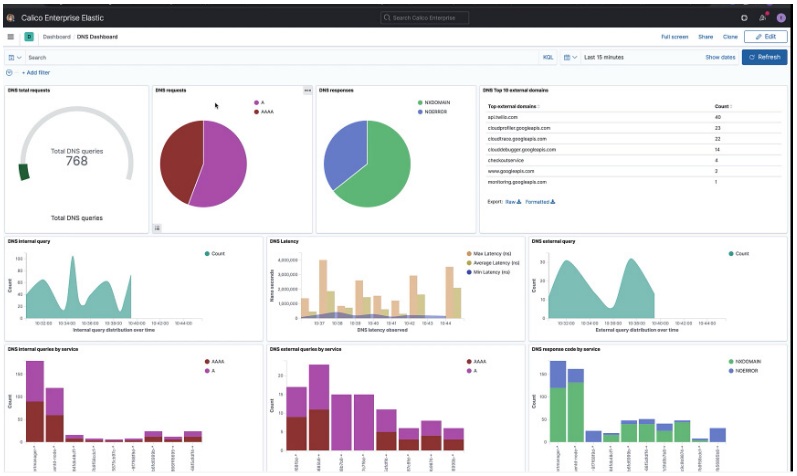
Application-level observability

Calico provides application-level observability with an Envoy integration in your data plane. Developers can get application-level information such as transaction/request throughput, error rates, performance, and latency metrics in an interactive and customizable dashboard, which allows them to monitor and troubleshoot connectivity issues, identify performance hotspots, and detect operational anomalies.



DNS Dashboard

Calico's DNS Dashboard observes and monitors traffic flows to look for DNS queries and replies, and generates a special set of DNS log data for all Kubernetes activity. It simplifies the troubleshooting process and speeds up problem resolution with an interactive UI displaying related DNS information categorized by microservices and workloads.



DNS Top 10 external domains

Top external domains	Count
api.twilio.com	40
cloudprofiler.googleapis.com	23
cloudtrace.googleapis.com	22
clouddebugger.googleapis.com	14
checkoutservice	4
www.googleapis.com	2
monitoring.googleapis.com	1

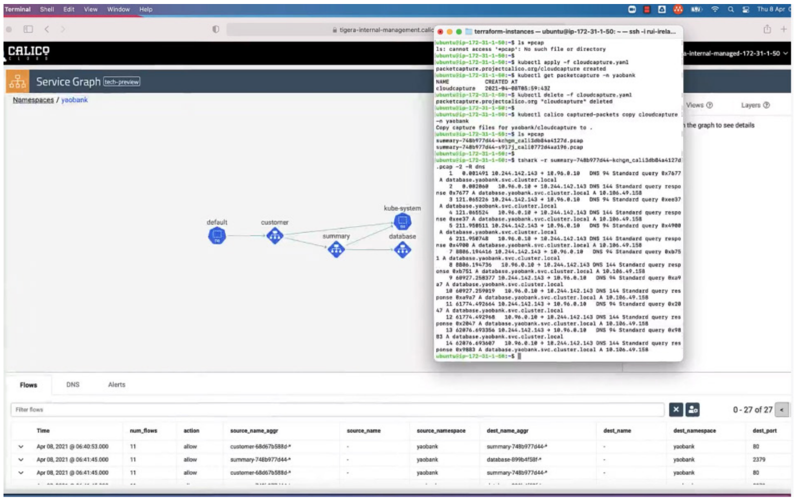
Export: Raw Formatted

TIGERA SOLUTION BRIEF

Addressing Kubernetes Observability Challenges with Calico

Dynamic Packet Capture

Calico's Dynamic Packet Capture is a Kubernetes-native way to capture packets from a specific pod or collection of pods with specified packet sizes and duration, in order to troubleshoot performance hotspots and connectivity issues faster. A command-line interface makes it easy to use tools like Wireshark to transfer generated PCAP files that are distributed across nodes, directly to your local machine for analysis.



```
ubuntu@ip-172-31-1-50:~$ ls *pcap
ls: cannot access '*pcap': No such file or directory
ubuntu@ip-172-31-1-50:~$
ubuntu@ip-172-31-1-50:~$ kubectl apply -f cloudcapture.yaml
packetcapture.projectcalico.org/cloudcapture created
ubuntu@ip-172-31-1-50:~$ kubectl get packetcapture -n yaobank
NAME           CREATED AT
cloudcapture   2021-04-08T05:59:43Z
ubuntu@ip-172-31-1-50:~$ kubectl delete -f cloudcapture.yaml
packetcapture.projectcalico.org "cloudcapture" deleted
ubuntu@ip-172-31-1-50:~$
ubuntu@ip-172-31-1-50:~$ kubectl calico captured-packets copy cloudcapture
-n yaobank
Copy capture files for yaobank/cloudcapture to .
ubuntu@ip-172-31-1-50:~$ ls *pcap
summary-748b977d44-kchgm_cali3db84a4127d.pcap
summary-748b977d44-s917j_cali0772d4aa196.pcap
ubuntu@ip-172-31-1-50:~$
ubuntu@ip-172-31-1-50:~$ tshark -r summary-748b977d44-kchgm_cali3db84a4127d
.pcap -2 -R dns
    1   0.001491 10.244.142.143 → 10.96.0.10   DNS 94 Standard query 0x7677
 A database.yaobank.svc.cluster.local
    2   0.002060   10.96.0.10 → 10.244.142.143 DNS 144 Standard query respo
nse 0x7677 A database.yaobank.svc.cluster.local A 10.106.49.158
    3 121.065226 10.244.142.143 → 10.96.0.10   DNS 94 Standard query 0xee37
 A database.yaobank.svc.cluster.local
    4 121.065524   10.96.0.10 → 10.244.142.143 DNS 144 Standard query respo
nse 0xee37 A database.yaobank.svc.cluster.local A 10.106.49.158
    5 211.950511 10.244.142.143 → 10.96.0.10   DNS 94 Standard query 0x4900
 A database.yaobank.svc.cluster.local
    6 211.950748   10.96.0.10 → 10.244.142.143 DNS 144 Standard query respo
nse 0x4900 A database.yaobank.svc.cluster.local A 10.106.49.158
    7 8806.194416 10.244.142.143 → 10.96.0.10   DNS 94 Standard query 0xb75
1 A database.yaobank.svc.cluster.local
    8 8806.194736   10.96.0.10 → 10.244.142.143 DNS 144 Standard query resp
onse 0xb751 A database.yaobank.svc.cluster.local A 10.106.49.158
    9 60927.258377 10.244.142.143 → 10.96.0.10   DNS 94 Standard query 0xa9
a7 A database.yaobank.svc.cluster.local
   10 60927.259019   10.96.0.10 → 10.244.142.143 DNS 144 Standard query res
ponse 0xa9a7 A database.yaobank.svc.cluster.local A 10.106.49.158
   11 61774.492664 10.244.142.143 → 10.96.0.10   DNS 94 Standard query 0x20
47 A database.yaobank.svc.cluster.local
   12 61774.492968   10.96.0.10 → 10.244.142.143 DNS 144 Standard query res
ponse 0x2047 A database.yaobank.svc.cluster.local A 10.106.49.158
   13 62076.693356 10.244.142.143 → 10.96.0.10   DNS 94 Standard query 0x98
83 A database.yaobank.svc.cluster.local
   14 62076.693607   10.96.0.10 → 10.244.142.143 DNS 144 Standard query res
ponse 0x9883 A database.yaobank.svc.cluster.local A 10.106.49.158
ubuntu@ip-172-31-1-50:~$
```

	Time	num_flows	action	source_name_aggr	source_name	source_namespace	dest_name_aggr	dest_name	dest_namespace	dest_port
⌄	Apr 08, 2021 @ 06:40:53.000	11	allow	customer-68d67b588d-*	-	yaobank	summary-748b977d44-*	-	yaobank	80
⌄	Apr 08, 2021 @ 06:41:45.000	11	allow	summary-748b977d44-*	-	yaobank	database-899b4f58f-*	-	yaobank	2379
⌄	Apr 08, 2021 @ 06:41:45.000	11	allow	customer-68d67b588d-*	-	yaobank	summary-748b977d44-*	-	yaobank	80

Kubernetes Observability as Code

The declarative nature of Kubernetes makes it extremely simple to do observability right. DevOps teams, SREs, and service owners can declare a high-level language construct around how they want to secure and observe the system, and Kubernetes can take care of the implementation. Observability can be treated as code so that it gets wired in as an integral part of the application, and then travels with the application so that it can run on any cloud, infrastructure, network, or application. The observability challenges posed by data silos, data volume, and granular components, and Kubernetes abstraction can be addressed by using an observability-as-code approach, which utilizes the declarative nature of Kubernetes. This leads to faster troubleshooting and a shorter time to a resolution if your application is experiencing performance, breakdown, or timeout issues.

To address Kubernetes observability challenges, start with a **free trial** of Calico Cloud.

Get a Free Trial

Tigera, Inc.
58 Maiden Lane, Fl 5
San Francisco, CA 94108
+1 (415) 612-9546 / www.tigera.io