DATA SHEET

# Project Calico v3.0

## Benefits

- **Simplicity.** Traditional Software Defined Networks (SDNs) are complex, making them hard to deploy and troubleshoot. Calico removes that complexity, with a simplified networking model designed for the demands of today's cloud-native applications.
- **Scale.** Unlike SDNs requiring a central controller that limits scalability, Calico is built on a fully distributed, scale-out architecture. So it scales smoothly from a single developer laptop to large enterprise deployments.
- **Performance.** Calico's architectural simplicity and use of the standard Linux data plane effectively deliver bare metal performance for virtual workloads.
- **Security.** Defining secure network policy used to be reserved for skilled network engineers. Calico's powerful micro-segmentation capabilities build on a simple policy language that naturally expresses the developer's intent.
- **Open Source, Open Community.** Calico is 100% open source, meaning it is free to use and modify. An active community of hundreds of contributors and thousands of users ensures the vibrancy and sustainability of the project.

## Overview

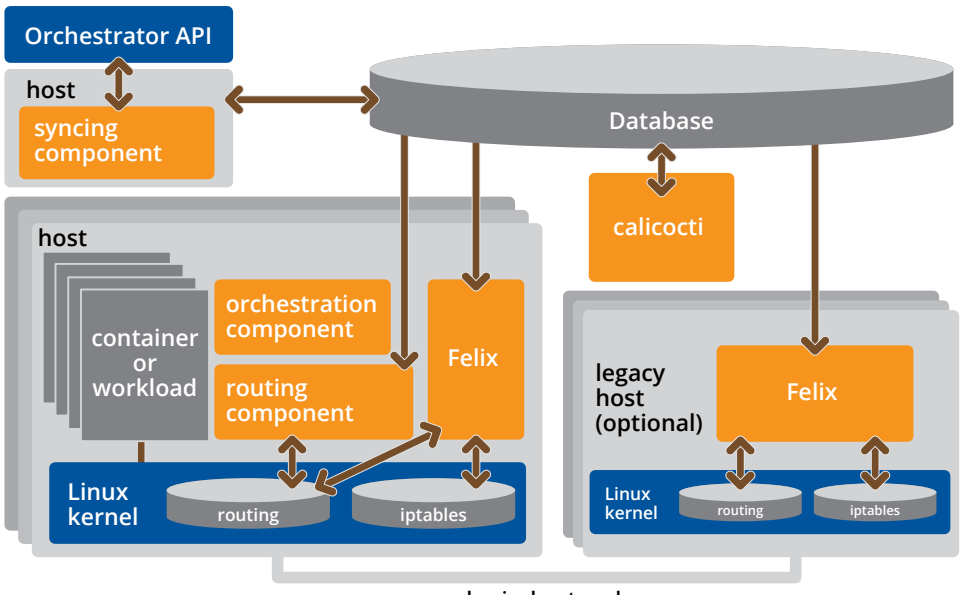
Project Calico provides secure network connectivity for containers and virtual machine workloads.

Calico creates and manages a flat Layer 3 network, assigning each workload a fully routable IP address. Workloads can communicate without IP encapsulation or network address translation for bare metal performance, easier troubleshooting, and better interoperability. In environments that require an overlay, Calico uses IP-in-IP tunneling or can work with other overlay networking such as flannel.

Calico also provides dynamic enforcement of network security rules. Using Calico's simple policy language, you can achieve fine-grained control over communications between containers, virtual machine workloads, and bare metal host endpoints.

Calico is proven in production at scale with a variety of orchestrators. Calico features integrations with Kubernetes, OpenShift, Docker, Mesos, DC/OS, and OpenStack. (Note that Calico 3.0 supports Kubernetes and standalone host endpoints only; other platforms are supported by Calico 2.6.4 and planned for Calico 3.1.)

## Architecture and Key Components




*Generalized Calico Architecture*

Calico applies networking (routing) and network policy rules to virtual interfaces for orchestrated containers and virtual machines, as well as enforcement of network policy rules on host interfaces for servers and virtual machines.

Each orchestrator (e.g., OpenStack, Kubernetes, Mesos, Docker) has its own plug-in mechanisms, but the general architecture is the same across all orchestrators.

- A Calico plug-in interfaces with the orchestrator to listen for events related to workload creation/destruction and (if applicable) network policy configuration via the orchestrator APIs.
- The calicoctl CLI command provides operator access to commands and configuration options complementing the features available via the orchestrator plug-in.
- Either Calico or the orchestrator, depending on the platform, assigns IP addresses to each new workload.
- Global state (workload identity, IP addresses, and network policy) is shared among all nodes via a highly available distributed key/value store. Generally Calico uses etcd for this function, but when running in Kubernetes it can also use the Kubernetes API datastore (instead of accessing etcd directly). Note that there are some small functional restrictions when using the Kubernetes API datastore.
- The Calico Felix agent runs on each node, programs kernel routes to local workloads, and calculates and enforces the local filtering rules required by the current policies applied to the cluster.
- The Calico routing agent communicates with other nodes (and optionally the underlying network infrastructure) to ensure routability between all nodes within the cluster.

Additional platform-specific components include:

- **Typha** – Kubernetes API Server fan-out for scalability
- **Calico-node** – container packaging of Felix, routing and configuration deployed as an agent on each worker node (containers only; not OpenStack)
- **DHCP agent** – OpenStack only
- **Calico libnetwork driver** – container network model (CNM) integration for Docker
- **Calico Network Policy Controller** – supports Kubernetes network policy API
- **CNI plug-in** – orchestrator integration for Kubernetes and other CNI-compatible platforms
- **Neutron plug-in or ML2 driver** – orchestrator integration for OpenStack

# Features

## Routing

| Features | Platform | Description |
|---|---|---|
| BGP Peering – Mesh mode | All | Automatically configures the Border Gateway Protocol control plane between hosts without any additional configuration or external route reflectors required. |
| BGP Peering – Infrastructure mode | All | Enables manual Border Gateway Protocol configuration to infrastructure routers and route reflectors. |
| Configurable 4-byte AS number | All | Enables advanced networking configuration with larger autonomous system (AS) address space |
| Native cloud networking (no overlay) | All | Enables communication between workloads without any overlay/encapsulation – traffic is routed via standard IP protocols |
| Flannel integration (Canal) | K8s | Enables Calico policy to be applied in conjunction with Flannel networking |
| Floating IPs* | OpenStack | Enables OpenStack floating IP functionality - assigning a virtual IP to a specific VM |
| Neutron host routes* | OpenStack | For VMs with multiple NICs, enables specification of which NIC should be used for data to particular prefixes. |
| Multiple IPs per VM* | OpenStack | Enables assignment of multiple IP addresses (including mixed IPv4/v6) to a workload. |
| Route Reflector | All | Calico includes a basic virtual route reflector for deployment scenarios an external route reflector is not available |

## IP Address Management

| Features | Platform | Description |
|---|---|---|
| Calico IPAM | K8s (etcd only), Mesos*, Docker* | IP Address Management (when used, Calico is responsible for assigning IP addresses to workloads; if not used, assignment is performed by the orchestrator)<br>• Dual stack (IPv4/v6)<br>• Allows multiple pools, each specified by an IPv4 or IPv6 CIDR<br>• Address aggregation: /26 per node, /26 advertised via BGP; rack aggregation<br>• Per-pod address or IP pool selection (Enables selection of a specific address, or which IP pool to assign an address from, via a pod annotation) |
| CNI IPAM | K8s | Support for CNI IPAM enables Calico to interoperate with external IP address management solutions<br>• Host Local IPAM, with endpoints advertised as multiple host routes (when using etcd) or aggregated (when using Kubernetes API datastore)<br>• Dual Stack (IPv4/v6) |
| OpenStack Neutron IPAM* Integration | OpenStack | Calico interoperates with Neutron assignment of IP addresses to workloads (advertised as multiple host routes) |
| IPAM override | K8s | Manual IPv4 and IPv6 address assignment to pod, bypassing IPAM, via Kubernetes pod annotation |

## Control Plane

| Features | Platform | Description |
|---|---|---|
| Direct etcd access | All | Calico global state is stored in an etcd distributed data store (configurable, can be dedicated or shared) |
| Kubernetes API datastore | K8s | Calico global state is stored via calls to the Kubernetes API server (ultimately in the Kubernetes etcd) |

## Data Plane

| Features | Platform | Description |
|---|---|---|
| Dual stack IPv4/6 | All | Data plane supports IPv4 and IPv6 datagrams |
| Packet forwarding | All | Linux kernel Layer 3 forwarding |
| L3/4 filtering | All | Filtering at Layer 3/4 via Linux kernel iptables with ipsets Access Control Lists |
| Packet format options | All | Options for formatting of IP packets:<br>• Unencapsulated<br>• IP-IP (always or cross-subnet)<br>• VxLAN (via Flannel) |
| Configurable MTU | All | The maximum transmission unit (MTU) size is configurable (either via CNI or calicoctl) for optimal network performance |
| SNAT | All | Optional outgoing SNAT configured per IP pool |

## Network Policy

| Features | Platform | Description |
|---|---|---|
| Dynamic enforcement of fine-grained network access control | All | Calico policies are enforced in real-time, typically within a small number of milliseconds, as workloads are created/moved/destroyed, labels are applied/updated/deleted, and policies are created/updated/deleted. |
| Kubernetes Network Policy API | K8s | Supports Kubernetes network policy API features<br>• Namespace scope<br>• Ingress<br>• Egress (from Kubernetes 1.8)<br>• Allow/deny access to/from namespaces<br>• Labels<br>• CIDRs (ipBlock) (from K8s 1.8)<br>• TCP/UDP ports |
| Calico policy | All | Calico policies support the following capabilities:<br>• Ingress<br>• Egress<br>• Global or namespace scope<br>• Labels<br>• CIDRs<br>• Layer 4 protocols: TCP, UDP, ICMP, SCTP, UDPlite, numbered (1-255); all or selected port range<br>• Positive or negative match on protocol, CIDR, selector label expression<br>• ICMP type & code<br>• Ordering<br>• Optional conntrack bypassing<br>• Pre-DNAT enforcement<br>• Actions: Allow/Deny |

### Network Policy (con't)

| Features | Platform | Description |
|---|---|---|
| Policy enforcement – host interfaces | All | Supports enforcement of network policies on host interfaces:<br>• Local host protection – for non-virtualized workloads<br>• Gateway – for transit/forwarded traffic |
| Policy enforcement – containers | K8s, Mesos*, Docker* | Supports enforcement of network policies on container virtual ethernet interfaces |
| Policy enforcement – virtual machines* | OpenStack | Supports enforcement of network policies on VM tap interfaces |

### Monitoring

| Features | Platform | Description |
|---|---|---|
| Readiness and liveness checks (for Felix and Typha components) | K8s | Integrates with Kubernetes readiness/liveness checking |
| Prometheus statistics (beta) | All | Enables reporting of key Calico performance metrics via Prometheus time series database |
| Logging (via syslog) | All | Supports logging with:<br>• Multiple levels (Felix): none, debug, info, warning, error, critical<br>• Configurable local log file<br>• Configurable syslog threshold |
| Horizon liveness reporting* | OpenStack | Liveness reporting to the Horizon dashboard for Felix agent and each configured TAP interface |

## Supported Platforms

### Orchestrators

Calico supports the following versions. Other platforms and/or versions may work but have not been tested.

| Orchestrator | Versions |
|---|---|
| Kubernetes | 1.7, 1.8, 1.9 |
| OpenStack* | Mitaka, Ocata, Pike |
| OpenShift | 3.6, 3.7 |
| Apache Mesos* | 1.3 |
| DC/OS* (Marathon, with Navstar for DNS) | 1.8, 1.9, 1.10 |
| Docker* (via libnetwork, not Swarm Mode) | 1.9 or later |

### Public Cloud

Calico has been successfully deployed on the following public clouds:

- AWS
- Microsoft Azure
- Google Cloud Platform
- IBM Cloud
- Digital Ocean
- Packet.net

In addition, Calico is integrated with

- Microsoft Azure Container Service Engine (ACS Engine)
- Google Container Engine
- IBM Cloud Container Service

Calico support is planned for:

- Amazon Elastic Container Service for Kubernetes (EKS)
- Microsoft Azure Container Service (AKS)

## About Project Calico

Calico is a community-driven open source (Apache 2.0 licensed) project, led by Tigera (www.tigera.io), with many contributors from across its diverse user and vendor community. For more information including access to community forums, documentation, and commercial support services, visit www.projectcalico.org.