chrislovecnm**.com**

Choosing a CNI Network Provider for Kubernetes

11 Nov 2017

[`kubernetes` `cni`]

The Container Network Interface (CNI) is a library definition, and a set of tools under the umbrella of the Cloud Native Computing Foundation project. For more information visit their GitHub **project**. Kubernetes uses CNI as an interface between network providers and Kubernetes networking.

Why Use CNI

Kubernetes default networking provider, kubenet, is a simple network plugin that works with various cloud providers. Kubenet is a very basic network provider, and basic is good, but does not have very many features. Moreover, kubenet has many limitations. For instance, when running kubenet in AWS Cloud, you are limited to 50 EC2 instances. Route tables are used to configure network traffic between Kubernetes nodes, and are limited to 50 entries per VPC. Moreover, a cluster cannot be set up in a Private VPC, since that network topology uses multiple route tables. Other more advanced features, such as BGP, egress control, and mesh networking, are only available with different CNI providers.

Choosing a Provider

> *Which CNI provider should I use?*

The above question is repeatedly asked on the #kops Kubernetes slack channel. There are many different providers, which have various features and options. Deciding which provider to use is not a trivial decision.

I am not going to say in this blog post: "Use this provider". There is not one provider that meets everyones needs, and there are many different options. The focus of this post is not to tell you which provider to use. The goal of this blog article is to educate and provide data so that you can make a decision.

CNI in kops

At last count, kops supports seven different CNI providers besides kubenet. Choosing from seven different network providers is a daunting task.

Here is our current list of providers that can be installed out of the box, sorted in alphabetical order.

- **Calico**

- **Canal (Flannel + Calico)**
- **flannel**
- **kopeio-vxlan**
- **kube-router**
- **romana**
- **Weave Net**

Any of these CNI providers can be used without kops. All of the CNI providers use a daemonset installation model, where their product deploys a Kubernetes Daemonset. Just use kubectl to install the provider on the master once the K8s API server has started. Please refer to each projects specific documentation.

Summary of the Providers

Calico

Mike Stowe provided a summary of both Calico and Canal.

Calico provides simple, scalable networking using a pure L3 approach. It enables native, unencapsulated networking in environments that support it, including AWS, AZ's and other environments with L2 adjacency between nodes, or in deployments where it's possible to peer with the infrastructure using BGP, such as on-premise. Calico also provides a stateless IP-in-IP mode that can be used in other environments, if necessary. Beyond scalable networking, Project Calico also offers policy isolation, allowing you to secure and govern your microservices/container infrastructure using advanced ingress and egress policies. With extensive Kubernetes support, you're able to manage your policies, in Kubernetes 1.8+.

Canal

Canal is a CNI provider that gives you the best of Flannel and Project Calico, providing simple, easy to use/out of the box VXLAN networking, while also allowing you take advantage of policy isolation with Calico policies.

This provider is a solution for anyone who wants to get up and running while taking advantage of familiar technologies that they may already be using.

flannel

Brandon Phillips' views on flannel.

Flannel is a simple and easy way to configure a layer3 network fabric designed for Kubernetes. No external database (uses Kubernetes API), simple performant works anywhere VXLAN default, can be layered with Calico policy engine (Canal). Oh, and lots of users.

Tectonic, CoreOS's Commercial Kubernetes product, uses a combination of flannel and Felix from Calico, much like Canal.

kopeio-networking

Justin Santa Barbara the founder of kopeio provided this:

kopeio-networking provides Kubernetes-first networking. It was purpose built for Kubernetes, making full use of the Kubernetes API, and because of that is much simpler and more reliable than alternatives that were retrofitted. The VXLAN approach is the most commonly used mode (as used in weave & flannel), but it also supports layer 2 (as used in calico), with more experimental support for GRE (the replacement for IPIP), and for IPSEC (for secure configurations). It does all of this with a very simple codebase

kube-router

Kube-router is a purpose-built network solution for Kubernetes ground up. It aims to provide operational simplicity and performance. Kube-router delivers a pod networking solution, a service proxy, and network policy enforcer as all-in-one solution, with single daemon set. Kuber-router uses Kubernetes native functionality like annotations, pod CIDR allocation by kube-controller-manger. So it does not have any dependency on a data store and does not implement any custom solution for pod CIDR allocation to the nodes. Kube-router also uses standard CNI plug-ins so does require any additional CNI plug-in. Kube-router is built on standard Linux networking toolset and technologies like ipset, iptables, ipvs, and lvs.

Murali Reddy founder of kube-router.

romana

Chris Marino summarized romana.

Romana uses standard layer 3 networking for pod networks. Romana supports Kubernetes Network Policy APIs and does not require an overlay, even when a cluster is split across network availability zones. Romana supports various network toplogies including flat layer 2 and routed layer 3 networks. Routes between nodes are installed locally and when necessary, distributed to network devices using either BGP or OSPF. In AWS deployments, Romana installs aggregated routes into the VPC route table to overcome the 50 node limit. This lets Romana use native VPC networking across availability zones for HA clusters. The current release uses its own etcd cluster, but the next version will optionally allow the Kubernetes etcd cluster to be used as a datastore.

Weave Net

Paul Fremantle’s synopsis of Weave Net.

Weave Net supports an overlay network that can span different cloud networking configurations, simplifying running legacy workloads on Kubernetes. For example, Weave supports multicast, even when the underlying network doesn’t. Weave can configure the underlying VPC networking and bypass the overlay when running on AWS. This provider forms a mesh network of hosts that are partitionable and eventually consistent; meaning that the setup is almost zero-config, and it doesn’t need to rely on an Etcd. Weave supports encryption and Kubernetes network policy, ensuring that there is security at the network level.

By the Numbers

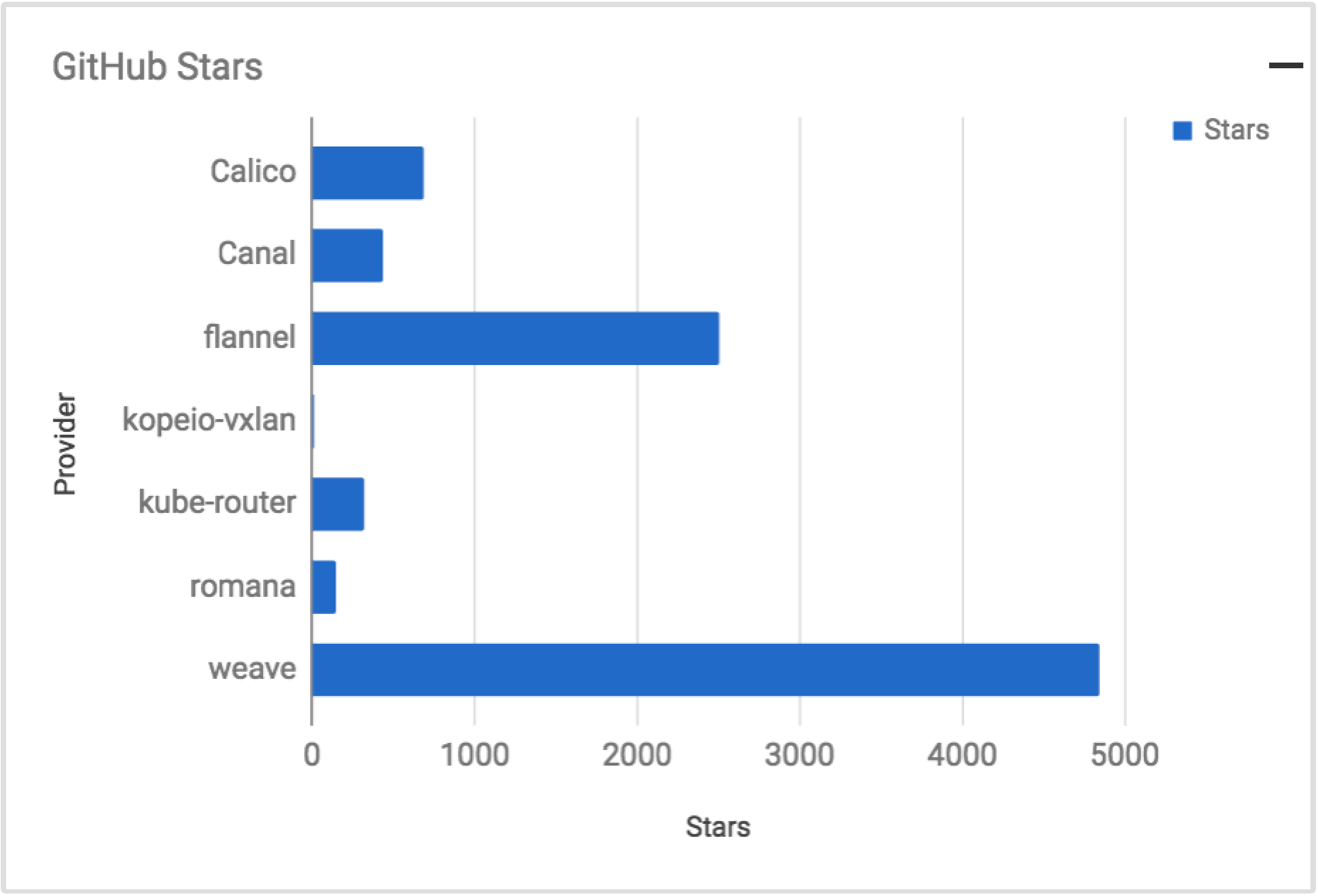
Kops does not track usage numbers of the different CNI providers, and I hope never does. When making a product selection of software hosted on GitHub, I look at three different numbers:

1. GitHub Stars - Likes on GitHub
2. GitHub Forks - Number of copies of the repo
3. GitHub Contributors - Number of people with merged code

The activity level of a project is critical. These are some of the metrics that I use to judge the activity level.

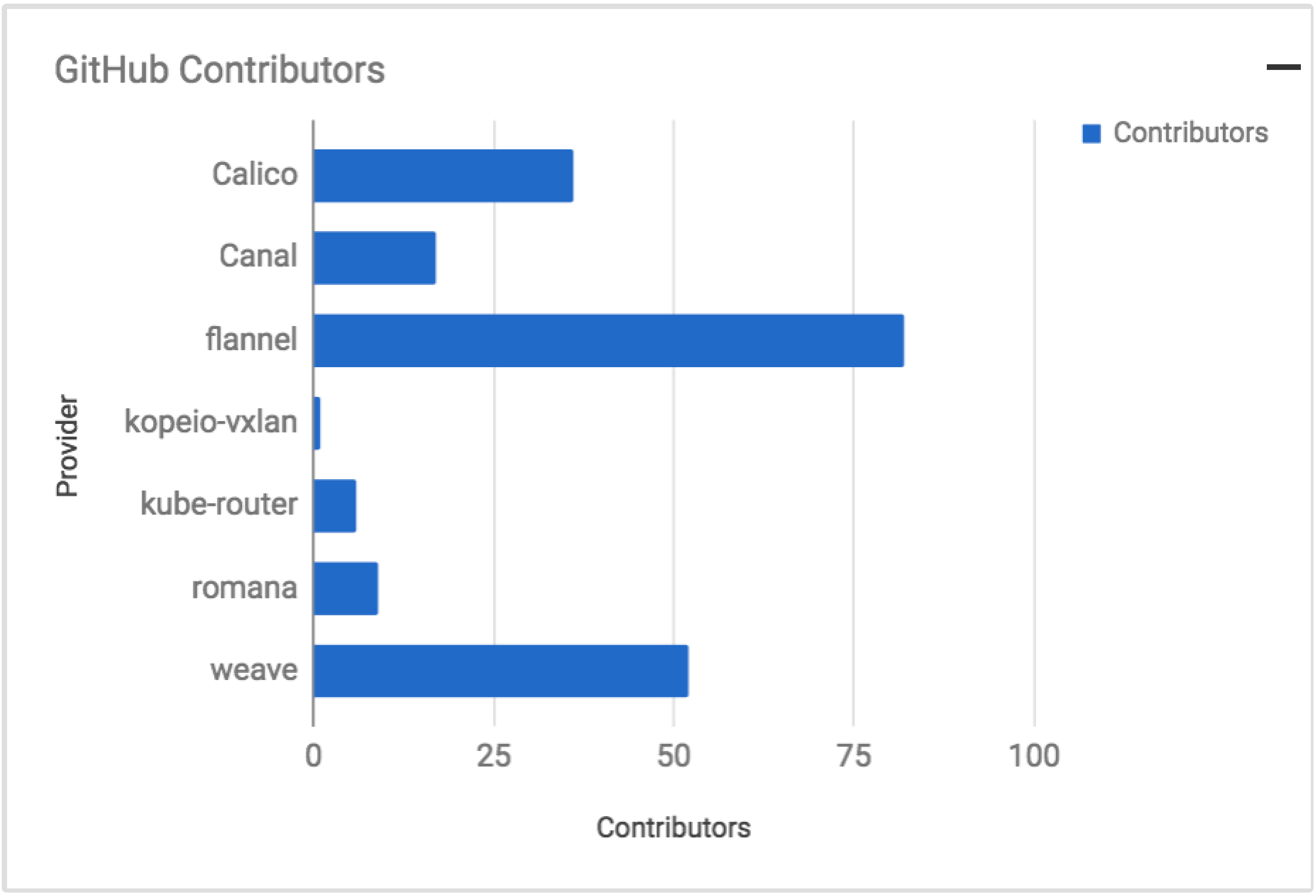
GitHub Stars

GitHub Stars are akin to likes on a Social Media platform.



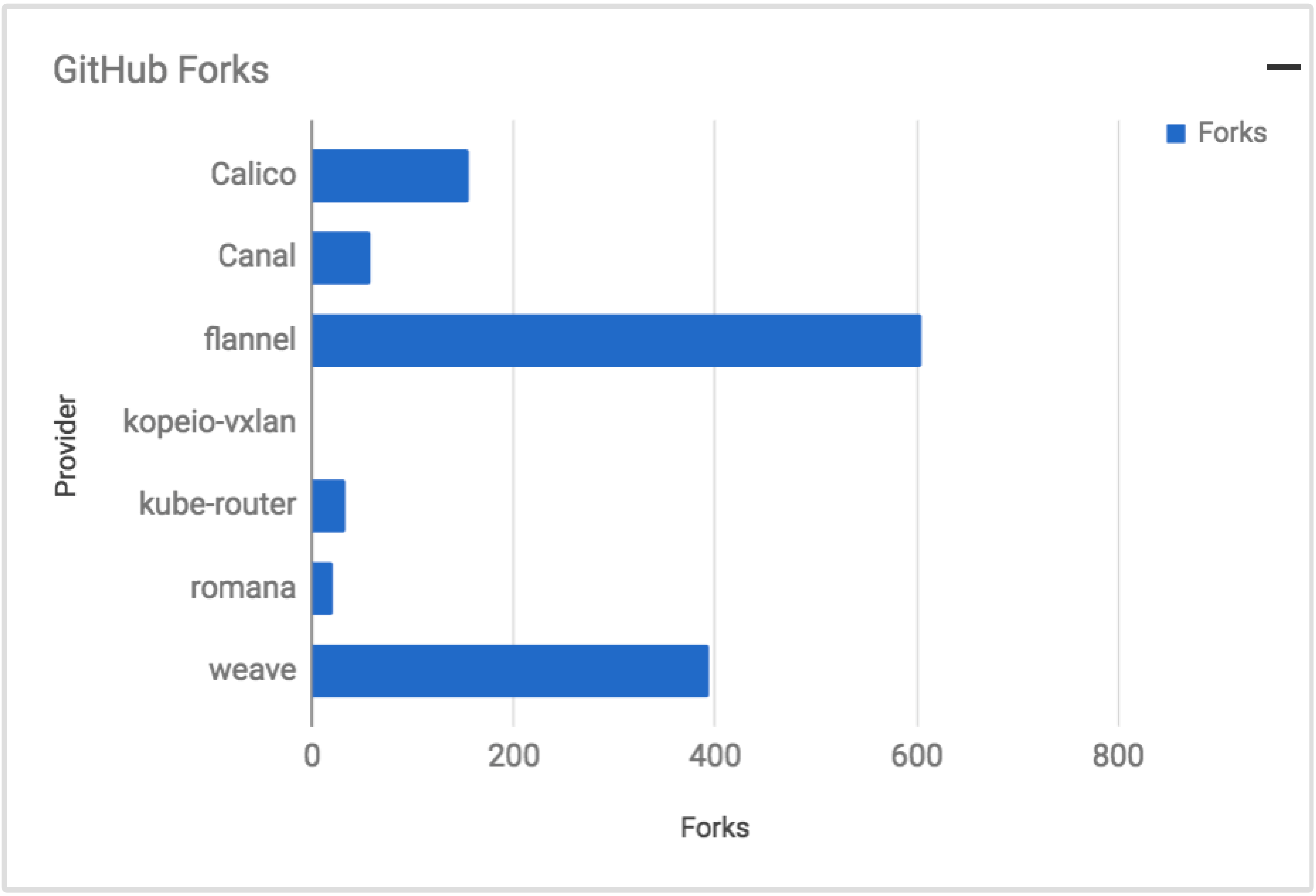
GitHub Contributors

The number of contributors talks to the number of people maintaining the code base and documentation. Active projects have a high number of contributors.



GitHub Forks

The number of forks is a mix of likes and contributors. Contributors typically have to fork the repo. Other people will fork the project to build a custom copy, push code to a feature branch that they own, or for various reasons.



Support Matrix

Here is a table of different features from each of the CNI providers mentioned:

Provider	Network Model	Route Distribution	Network Policies	Mesh	External Datastore	Encryption	Ingress/Egress Policies	Commercial Support
Calico	Layer 3	Yes	Yes	Yes	Etcd [1]	Yes	Yes	Yes
Canal	Layer 2 vxlan	N/A	Yes	No	Etcd [1]	No	Yes	No
flannel	vxlan	No	No	No	None	No	No	No
kopeio-networking	Layer 2 vxlan [2]	N/A	No	No	None	Yes [3]	No	No
kube-router	Layer 3	BGP	Yes	No	No	No	No	No
romana	Layer 3	OSPF	Yes	No	Etcd	No	Yes	Yes
Weave Net	Layer 2 vxlan [4]	N/A	Yes	Yes	No	Yes	Yes [5]	Yes

1. Calico and Canal include a feature to connect directly to Kubernetes, and not use Etcd.
2. kopeio CNI provider has three different networking modes: vlan, layer2, GRE, and IPSEC.
3. kopie-network provides encryptions in IPSEC mode, not the default vxlan mode.
4. Weave Net can operate in AWS-VPC mode without vxlan, but is limited to 50 nodes in EC2.
5. Weave Net does not have egress rules out of the box.

Table Details

Network Model

The Network Model with providers is either encapsulated networking such as VXLAN, or unencapsulated layer 2 networking. Encapsulating network traffic requires compute to process, so theoretically is slower. In my opinion, most use cases will not be impacted by the overhead. More about VXLAN on **wikipedia**.

Route Distribution

For layer 3 CNI providers, route distribution is necssary. Route distribution is typically via BGP. Route distribution is nice to have a feature with CNI, if you plan to build clusters split across network segments. It is an exterior gateway protocol designed to exchange routing and reachability information on the internet. BGP can assist with pod to pod networking between clusters.

Network Policies

A kubernetes.io blog post about network policies in 1.8 **here**.

> *Kubernetes now offers functionality to enforce rules about which pods can communicate with each other using network policies. This feature is has become stable Kubernetes 1.7 and is ready to use with supported networking plugins. The Kubernetes 1.8 release has added better capabilities to this feature.*

Mesh Networking

This feature allows for “Pod to Pod” networking between Kubernetes clusters. This technology is not Kubernetes federation, but is pure networking between Pods.

Encyption

Encrypting the network control plane, so all TCP and UDP traffic is encrypted.

Ingress / Egress Policies

The network policies are both Kubernetes and Non-Kubernetes routing control. For instance, many providers will allow an administrator to block a pod communicating with an EC2 instance meta and data service on 169.254.169.254.

Summary

If you do not need the advanced features that a CNI provider delivers, use kubenet. It is stable, and fast. Otherwise, pick one. If you do need run more than 50 nodes on AWS, or need other advanced features, make a decision quickly (don’t spend days deciding), and test with your cluster. File bugs, and develop a relationship with your network provider. At this point in time, networking is not boring in Kubernetes. It is getting more boring every day! Monitor test and monitor more.

Thanks

Appreciate all of the feedback that I received on the pull request for this **blog post**. Also, all of the summaries that were provided by the different people that work on the different projects.

Related Posts

GKE On-Prem 18 Aug 2018

Kubernetes Engine Demos 25 Jul 2018

Update Google Cloud Load Balancer SSL Cert 11 Jan 2018

For help with all-things Kubernetes have me contact you today!

Name

Email

Subject

Message

Send Message