Enable multiple network interfaces for Rook storage providers

Project Proposal for GSoC 2019 by Giovan Isa Musthofa

Abstract

This project aims to create new API to enable multiple network interface for Rook storage providers. Currently, Rook providers only choice is to use `hostNetwork` or not. The new API will be used to define networks resource for Rook clusters. Rook operators will be able to consume those definitions and manage them. Therefore, it enables more fine-grained control over storage providers network access.

Current Implementation

Inside a Rook cluster, there are all kinds of services such as Rook Operator, Rook Agents, OSD, MON, RGW, iSCSI, NFS, and vice versa. A minimal Rook storage provider will need to manage its own data by storing and serving data, while Rook operator will do the rest. Kubernetes flat pod networking model is still sufficient for this.

However, for highly available storage providers such as Ceph, EdgeFS, or CockroachDB, they need to communicate with other instances across pods/nodes for data replication or building consensus. To enable this, user needs to exposes all node interface to the Rook providers using `hostNetwork`. This however is a bad practice and might give an attacker direct access to Kubernetes nodes network once the attacker gained access of the Rook pod. Another working approach is to separates public and private subnet for Ceph OSDs. While the latter approach works to provide more consistent bandwidth, support for more robust multi-homed cluster with multiple network interfaces to improve data security still persists.

Benefits

Once the new API is implemented Rook will be able to set up proper multi-homed Rook clusters with multiple network interfaces. The biggest benefit of this project will be to *entirely* avoid using `hostNetwork` at all. In the future, Rook clusters will have OSDs across nodes connected to a dedicated overlay network for OSD to OSD only network, another dedicated overlay network for MONs quorum, a default network to expose storage provider to the load balancer, and other networks as the Rook operator/cluster administrator see fit. The new network architecture will lend itself to better data security and access control, better tail latency/QoS/SLA, network containment, and easier monitoring.

Findings

Issues

- Generic support for multi-homed Rook clusters #2621
  The main issue the project is trying to address.
- Investigation: Host network security improvements for Ceph #2031
  Security issues surrounding usage of `hostNetwork`, plus a bonus video.
- Using separate networks for Ceph cluster #1523
  Common use case for the new API to enable network separation. The current solution is to use subnets after `hostNetwork`.

Related Projects

- Multus-CNI
  Multus CNI is a container network interface (CNI) plugin for Kubernetes that enables attaching multiple network interfaces to pods. Typically, in Kubernetes each pod only has one network interface (apart from a loopback) -- with Multus you can create a multi-homed pod that has multiple interfaces. This is accomplished by Multus acting as a "meta-plugin", a CNI plugin that can call multiple other CNI plugins.

Deliverables

- PoC Network API
  Research storage providers networking requirements and Multus-CNI capabilities. Then, write proposal sub-resource to propose API and configuration designs and have it commented by mainteners in a PR. Implements the proposed design with unit tests.
- PoC of running multi-homed with multiple-interface Rook EdgeFS cluster
  Gather some metrics around the network performance and overall throughput. Publish it in a blog posts to give sneak peak for the new feature.
- Documentation
  Document the new API and how users would configure the Rook clusters using the new options. Also give warning to users who still runs their cluster using `hostNetwork`. Also add docs to migrate existing Rook cluster using `hostNetwork` to the new cluster with multiple network interfaces.
- Benchmarks
  Write benchmarks around the new configuration to illustrate the performance difference between separated network vs `hostNetwork` with subnet vs `hostNetwork` clusters.
- Start migrating the existing test/example clusters using the new configuration.

Timeline

Community Bonding/Research Phase | May 7 - May 26

- Run local development tools and exercise debugging to submit patches.
- Research differents storage providers and configurations (EdgeFS, Ceph, CockroachDB, Cassandra, Minio, and NFS) network requirements and compares them.
- Research Multus-CNI capabilities and possible deployment options.
- Publish previous research results and propose design based on those findings on a PR.
- Discuss with mentors and maintainers regarding the proposed design. Research other options based on existing sample or crude prototypes.

Coding Phase 1 | May 27 - June 25

- Implement the agreed design requirements. First rough PoC of the Network API with bare minimum support for EdgeFS storage providers. (Week 01)
- Continue work on the PoC so it's presentable by June 10. (Week 02)
- Work on preparation to demo the PoC on Shanghai KubeCon June 22. This work includes ironing bugs, and performing benchmarks. (Week 03)
- Based on feedback of the first PoC and demo, make changes accordingly, provide tests and documentation. Submit the new API as `v1alpha1` for EdgeFS and have it merged to master. (Week 04)

Coding Phase 2 | June 26 - July 23

- Improve Network API implementation for EdgeFS to include more/complete capabilities (e.g. portMappings) as defined in the proposed design. Updated docs and tests will be made available and PR will be submitted. (Week 05)
- Implement Network API to work on Ceph storage providers so that it has comparable features with EdgeFS storage providers. Updated docs and tests will be made available and PR will be submitted (Week 06).
- Attempt to implement Network API for more storage providers (CockroachDB, Cassandra) and provide more generic Network API implementations. (Week 07)
- Write benchmarks to showcase the early improvement made by the new Network API and publish it over a blogpost. (Week 08)
- Ask people from the community to try out the early feature and gather feedbacks/bug reports from them. (Week 08)

Coding Phase 3 | July 24 - August 22

- Write e2e integration tests to deploy and consume multi-homed Rook clusters with multiple network interface to catch regressions. (Week 09)
- Research migration mechanism from existing `hostNetwork` enabled Rook cluster. (Week 10)
- Document the migration, including common cases, and errors, and how to troubleshoots those errors. (Week 11)
- Have the final code and docs reviewed and merged to master. (Week 12)

About Me

Hi, my name is Giovan Isa Musthofa. I'm a (19 years old) sophomore in Computer Science undergraduate program at Universitas Indonesia. I live in Jakarta, but during most of my summer, I will be home in rural area (Rembang, Central Java). I've coded Pascal and C# during high school and most of my course in college requires Python or Java. Actually, my first experience with open-source started two years ago when I was a GCI student, therefore I'm already somewhat familiar with open-source development model that doesn't exist in my undergraduate program. However, being a regular contributor was still beyond my ability back then. I hope with GSoC I can be regular contributor or even maintainer of OSS project.

For now, I'm a senior member of Network Security and Operating System SIG in my faculty where we develop security and operational skills around Linux/CTFs. I have run an internal CTF on my faculty and another public one due at the end of the next August. I gained most of my computer networking skills from those experience.

My experience in Go mostly came from my attempt to rewrite our CTF deployment automation tools using proper docker engine API SDK that previously written using scripts around docker and docker-compose.

Lately, I also joined Kubernetes community in Indonesia. Once or twice a month there will be talks by speakers from Indonesian tech company sharing their experience using Cloud Native technology such as Telepresence, Helm, Istio, etc. I also took qwiklabs course provided by them to learn more about cloud native solutions available in GCP.

Notes

This section contains recommended things to do outside the project

- Maintain a google document during coding phase with **daily** update to have a proper log containing my progress and roadblocks
- Write a **weekly** blog post detailing actionable items delivered that week
- Contact mentor daily to report updates
- Attempt to participate (or follow up) in the weekly community meeting
- Set up a meta-tracker repository containing:
    - Links to my gdoc with daily update
    - List of Issues I worked on
    - List of PRs I worked on
    - List of blog posts I posted