A distributed, reliable key-value store for the most critical data of a distributed system

Henry Saptono (henry@nurulfikri.ac.id)

Apa itu etcd ?

- Open source software
- Apache 2.0 licensed
- Written in Go
- Layanan Konsensus dan Penemuan (discovery)
- Penyimpanan kunci/nilai (key/value)
- Terdistribusi dan sangat tersedia (highly available)
- Fault tolerance
- Didesain untuk mudah dimengerti dan kesederhanaan
- Aplikasi dapat membaca dan menulis data ke etcd.
- Studi kasus sederhana adalah untuk menyimpan detail koneksi basis data atau flag-flag fitur kedalam etcd sebagai pasangan nilai dan kunci.

Apa itu etcd ?

- etcd adalah penyimpanan (database) nilai/kunci terdistribusi open source yang digunakan untuk menyimpan dan mengelola informasi penting yang harus terus dijalankan oleh sistem terdistribusi. Terutama, etcd untuk mengelola data konfigurasi, data state (status), dan metadata untuk Kubernetes (platform orkestrasi container populer)

Apa itu etcd ?

- Seperti semua beban kerja yang terdistribusi, beban kerja container memiliki persyaratan manajemen yang kompleks yang menjadi lebih kompleks dengan skala beban kerja.
- Kubernetes menyederhanakan proses pengelolaan beban kerja ini dengan mengoordinasikan tugas-tugas seperti konfigurasi, penyebaran, penemuan layanan, penyeimbangan beban, penjadwalan pekerjaan, dan pemantauan kesehatan di seluruh cluster, yang dapat berjalan di beberapa mesin di beberapa lokasi.

Apa itu etcd ?

- Tetapi untuk mencapai koordinasi ini, Kubernetes membutuhkan penyimpanan data yang menyediakan sumber kebenaran tunggal yang konsisten tentang status sistem — semua kluster dan pod serta contoh aplikasi di dalamnya — pada titik waktu tertentu. etcd adalah penyimpanan data yang digunakan untuk membuat dan memelihara versi kebenaran ini.

Apa itu etcd ?

- etcd melayani peran serupa untuk Cloud Foundry — open source, Platform-as-a-Service (PaaS) multicloud — dan merupakan opsi yang layak untuk mengoordinasikan sistem kritis dan metadata lintas kluster aplikasi yang didistribusikan.
- Nama "etcd" berasal dari konvensi penamaan dalam struktur direktori Linux: Di UNIX, semua file konfigurasi sistem untuk satu sistem terkandung dalam folder yang disebut "/etc;" "D" adalah singkatan dari "distributed” (didistribusikan)

Mengapa etcd?

- Bukan tugas kecil untuk berfungsi sebagai tulang punggung data (*data backone*) yang membuat beban kerja terdistribusi berjalan. Tetapi etcd dibangun untuk tugas tersebut, dirancang dari bawah ke atas untuk kualitas berikut:
  - **Sepenuhnya direplikasi (*fully replicated*)**: Setiap node dalam gugus etcd memiliki akses penyimpanan data lengkap.
  - **Sangat tersedia (*highly available*)**: etcd dirancang untuk tidak memiliki titik kegagalan dan dengan ‘sabar’ mentolerir kegagalan perangkat keras dan partisi jaringan.
  - **Sangat konsisten (*Reliably consistent*)**: Setiap data 'baca' mengembalikan data terbaru 'tulis' di semua cluster.
  - **Cepat (*fast*)**: etcd telah di-benchmark pada 10.000 kali menulis per detik.

Mengapa etcd?

- **Aman (*Secure*)**: etcd mendukung Transport Layer Security (TLS) otomatis dan otentikasi sertifikat klien secure socket layer (SSL). Karena etcd menyimpan data konfigurasi yang vital dan sangat sensitif, administrator harus menerapkan kontrol akses berbasis peran dalam penyebaran dan memastikan bahwa anggota tim yang berinteraksi dengan etcd terbatas pada tingkat akses paling tidak istimewa yang diperlukan untuk melakukan pekerjaan mereka.
- **Sederhana (*Simple*)**: Aplikasi apa pun, dari aplikasi web sederhana hingga mesin orkestrasi wadah (*container*) yang sangat kompleks seperti Kubernetes, dapat membaca atau menulis data ke etcd menggunakan alat HTTP / JSON standar.

Raft consensus algorithm

- etcd dibangun di atas **algoritma konsensus Raft** untuk memastikan konsistensi penyimpanan data di semua node dalam sebuah cluster — ‘tabel taruhan’ untuk sistem terdistribusi yang toleran terhadap kesalahan.
- Raft mencapai konsistensi ini melalui node pemimpin (***leader***) terpilih yang mengelola replikasi untuk node lain di cluster, yang disebut pengikut (***follower***). Pemimpin (leader) **menerima permintaan dari klien,** yang kemudian diteruskan ke node pengikut (follower). Setelah pemimpin memastikan bahwa **mayoritas node pengikut** telah menyimpan setiap permintaan baru sebagai entri log, entri itu berlaku ke semua keadaan mesin dalam cluster dan mengembalikan hasil eksekusi "tulis" itu - kepada klien. **Jika pengikut crash atau paket jaringan hilang**, pemimpin mencoba lagi sampai semua pengikut menyimpan semua entri log secara konsisten.

Raft consensus algorithm

- Jika ***node*** pengikut (***follower***) gagal menerima pesan dari pemimpin (***leader***) dalam interval waktu tertentu, pemilihan diadakan untuk memilih pemimpin baru. Pengikut menyatakan dirinya sebagai kandidat, dan pengikut lainnya memilihnya atau node lain berdasarkan ketersediaannya.
- Setelah pemimpin baru terpilih, ia mulai mengelola replikasi, dan prosesnya berulang. **Proses ini** memungkinkan semua node etcd untuk mempertahankan (menjaga), dan mereplikasi penyimpanan data secara konsisten dan dengan ketersediaan yang tinggi.

etcd dan Kubernetes

- etcd termasuk di antara komponen inti Kubernetes dan berfungsi sebagai penyimpan nilai kunci utama untuk membuat kluster Kubernet berfungsi dan toleran terhadap kesalahan.
- Server API Kubernetes menyimpan data status setiap cluster di etcd. Kubernetes menggunakan fungsi “watch” yang disediakan etcd untuk memonitor data ini dan untuk mengkonfigurasi ulang dirinya sendiri ketika terjadi perubahan. Fungsi "watch" menyimpan nilai yang mewakili keadaan aktual dan ideal dari cluster dan dapat memulai respons ketika mereka berbeda(berubah).
- Untuk mempelajari cara menghubungkan aplikasi layanan Kubernetes menggunakan etcd, silhakan tinjau tutorial ini (https://cloud.ibm.com/docs/databases-for-etcd?topic=cloud-databases-tutorial-k8s-app).

CoreOS dan sejarah etcd

- etcd (https://etcd.io) dibuat oleh tim yang sama yang bertanggung jawab untuk merancang CoreOS Container Linux, sistem operasi kontainer yang banyak digunakan yang dapat dijalankan dan dikelola secara efisien dalam skala besar. Mereka awalnya membangun etcd di Raft untuk mengoordinasi banyak salinan Container Linux secara bersamaan, untuk memastikan waktu operasi aplikasi yang tidak terputus.
- Pada bulan Desember 2018, tim menyumbangkan etcd ke Cloud Native Computing Foundation (CNCF), organisasi nirlaba netral yang mempertahankan kode sumber, domain, layanan host, infrastruktur cloud, dan properti proyek etcd sebagai sumber daya open source untuk komunitas pengembang cloud berbasis kontainer. CoreOS telah bergabung dengan Red Hat.

etcd vs. ZooKeeper vs. Consul

- Basis data lain telah dikembangkan untuk mengelola informasi koordinasi antara seluruh cluster aplikasi terdistribusi.
- Ada dua yang paling umum dibandingkan dengan etcd adalah ZooKeeper dan Consul.

etcd vs. ZooKeeper vs. Consul

- ZooKeeper awalnya dibuat untuk mengoordinasikan data konfigurasi dan metadata di seluruh cluster Apache Hadoop. (Apache Hadoop adalah kerangka kerja sumber terbuka, atau kumpulan aplikasi, untuk menyimpan dan memproses volume besar data pada kelompok perangkat keras komoditas).
- ZooKeeper lebih tua dari etcd, dan pelajaran yang didapat dari bekerja dengan ZooKeeper mempengaruhi desain etcd.

etcd vs. ZooKeeper vs. Consul

- Akibatnya, etcd memiliki beberapa kemampuan penting yang tidak dimiliki ZooKeeper. Misalnya, tidak seperti ZooKeeper, etcd dapat melakukan hal berikut:
  - memungkinkan untuk konfigurasi ulang keanggotaan cluster yang dinamis.
  - Tetap stabil saat melakukan operasi baca / tulis di bawah beban tinggi.
  - Mempertahankan model data kontrol konkurensi multi-versi.
  - Menawarkan pemantauan kunci yang andal yang tidak pernah membatalkan acara tanpa memberikan pemberitahuan.
  - Menggunakan konkurensi primitif yang memisahkan koneksi dari sesi.
  - Mendukung berbagai bahasa dan kerangka kerja (ZooKeeper memiliki protokol RPC kustom sendiri yang mendukung binding bahasa yang terbatas).

etcd vs. ZooKeeper vs. Consul

- Consul adalah solusi layanan jaringan untuk sistem terdistribusi, kemampuannya yang berada di suatu tempat di antara etcd dan service mesh Istio untuk Kubernetes.
- Seperti etcd, Consul mencakup penyimpanan nilai kunci terdistribusi berdasarkan algoritma Raft dan mendukung antarmuka pemrograman aplikasi (API) HTTP / JSON. Keduanya menawarkan konfigurasi keanggotaan cluster dinamis, tetapi Konsul tidak mengontrol dengan kuat terhadap beberapa versi konkuren dari data konfigurasi, dan ukuran basis data maksimum yang bekerja dengannya akan lebih kecil.

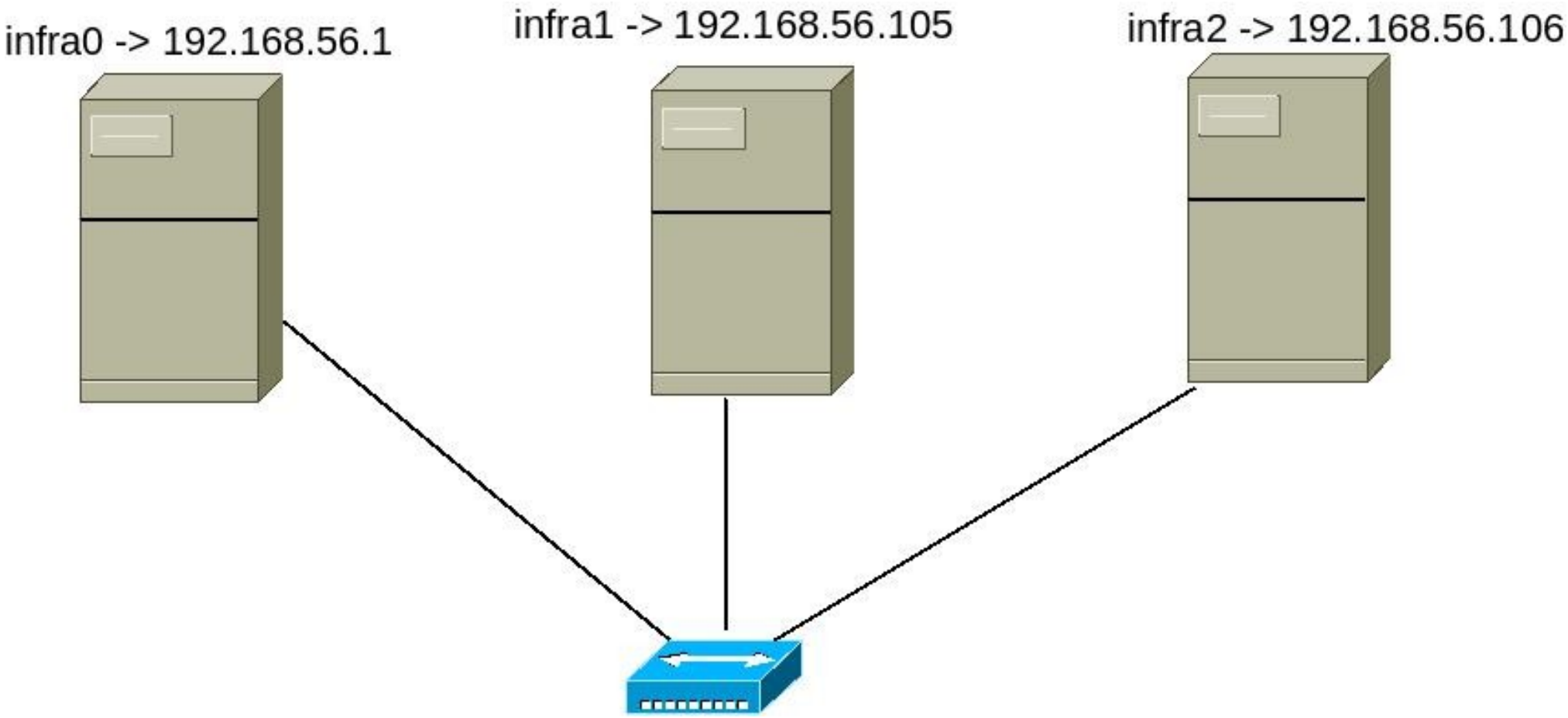
etcd vs. Redis

- Seperti etcd, Redis adalah alat open source, tetapi fungsi dasarnya berbeda.
- Redis adalah penyimpanan data dalam memori dan dapat berfungsi sebagai pangkalan data, cache, atau pesan. Redis mendukung jenis dan struktur data yang lebih luas daripada etcd dan memiliki kinerja baca / tulis yang jauh lebih cepat.

etcd vs. Redis

- Tetapi etcd memiliki toleransi kesalahan yang unggul, failover yang lebih kuat dan kemampuan ketersediaan data yang berkelanjutan, dan, yang paling penting, etcd tetap menyimpan semua data yang tersimpan ke disk, pada dasarnya mengorbankan kecepatan untuk keandalan yang lebih besar dan konsistensi yang terjamin. Untuk alasan ini, Redis lebih cocok untuk berfungsi sebagai sistem cache memori terdistribusi daripada untuk menyimpan dan mendistribusikan informasi konfigurasi sistem.

Menggunakan etcd



Instalasi etcd

- Instalasi etcd pada semua node (sistem ubuntu)
  - Pada node infra0, infra1 dan infra2

```
# sudo apt-get install etcd
```

Download and build the latest version

  - https://etcd.io/docs/v3.4.0/dl-build/

Menjalankan cluster sistem etcd

- Pada node infra0 :

```
# etcd --name infra0 --initial-advertise-peer-urls http://192.168.56.1:2380 \
 --listen-peer-urls http://192.168.56.1:2380 \
--listen-client-urls http://192.168.56.1:2379,http://127.0.0.1:2379 \
--advertise-client-urls http://192.168.56.1:2379 \
--initial-cluster-token etcd-cluster \
--initial-cluster
infra0=http://192.168.56.1:2380,infra1=http://192.168.56.105:2380,infra2=
http://192.168.56.106:2380 \
--initial-cluster-state new
```

Menjalankan cluster sistem etcd

- Pada node infra1 :

```
# etcd --name infra1 --initial-advertise-peer-urls http://192.168.56.105:2380 \
--listen-peer-urls http://192.168.56.105:2380 \
--listen-client-urls http://192.168.56.105:2379,http://127.0.0.1:2379 \
--advertise-client-urls http://192.168.56.105:2379 \
--initial-cluster-token etcd-cluster \
--initial-cluster infra0=http://192.168.56.1:2380,infra1=http://192.168.56.105:2380,infra2=http://192.168.56.106:2380 \
--initial-cluster-state new
```

Menjalankan cluster sistem etcd

- Pada node infra2 :

```
#etcd --name infra2 --initial-advertise-peer-urls http://192.168.56.106:2380 \
--listen-peer-urls http://192.168.56.106:2380 \
--listen-client-urls http://192.168.56.106:2379,http://127.0.0.1:2379 \
--advertise-client-urls http://192.168.56.106:2379 \
--initial-cluster-token etcd-cluster \
--initial-cluster infra0=http://192.168.56.1:2380,infra1=http://192.168.56.105:2380,infra2=http://192.168.56.106:2380 \
--initial-cluster-state new
```

Menampilkan daftar node dalam cluster etcd

- Perintah berikut ini bisa dijalankan di semua node etcd.

```
# ETCDCTL_API=3 etcdctl member list
```

Mengetahui status peran node dalam cluster etcd

- Perintah berikut ini bisa dijalankan di semua node etcd.

```
# ETCDCTL_API=3 etcdctl endpoint status -w table
```

Menulis data

- Menulis atau menyimpan data (key/value)

  # ETCDCTL_API=3 etcdctl put data1 "Hello World"

  # ETCDCTL_API=3 etcdctl put data2 "Hai Dunia”

Membaca data

- Membaca data (key/value)

  # ETCDCTL_API=3 etcdctl get data1

  # ETCDCTL_API=3 etcdctl get data --prefix=true

Watch data

- Watch data (key/value)

  # ETCDCTL_API=3 etcdctl watch data1

  # ETCDCTL_API=3 etcdctl watch data1 --endpoints=http://192.168.56.105:2379

Menulis dan membaca data via HTTP/ JSON menggunakan curl

- Menulis data:

  # curl -L http://localhost:2379/v3alpha/kv/put \

  -X POST -d '{"key": "Zm9v", "value": "YmFy"}'

  catatan:

  Zm9v → base64 encode dari ‘foo’

  YmFy → base64 encode dari ‘bar’

Menulis dan membaca data via HTTP/ JSON menggunakan curl

- Membaca data:

  # curl -L http://localhost:2379/v3alpha/kv/range \

  -X POST -d '{"key": "Zm9v"}'

  catatan:

  Zm9v → base64 encode dari ‘foo’