Sri Ganesh Reddy Yerra Kondappagari
290685@ust.com

ETCD: A Distributed Key-Value Store

Introduction to ETCD

ETCD is an open-source, distributed key-value store designed for distributed systems. It provides a reliable way to store data across a cluster of machines, ensuring consistency and availability even in the face of machine failures. ETCD is widely used for configuration management, service discovery, and leader election in distributed applications, such as Kubernetes.

Key Features

1. **Strong Consistency**: ETCD uses the Raft consensus algorithm to ensure all data is consistent across the cluster.
2. **Hierarchical Data Storage**: Data is stored in a hierarchical key-value format, making it easy to organize and retrieve.
3. **High Availability**: ETCD can recover from machine or network failures without data loss.
4. **Watch Mechanism**: Clients can subscribe to changes in specific keys or directories for real-time updates.
5. **Secure Communication**: Supports TLS encryption and role-based access control (RBAC).

Architecture of ETCD

ETCD operates as a cluster of nodes, where each node can read and write data. A leader node manages writes, ensuring changes are replicated to follower nodes using the Raft protocol.

Components

- **Cluster Nodes**: Machines running ETCD instances.
- **Client Libraries**: Interface for applications to interact with the ETCD API.
- **Raft Protocol**: Ensures consistency and manages leader election.

Workflow

1. **Client Request**: Clients send read or write requests to any cluster node.

Sri Ganesh Reddy Yerra Kondappagari
290685@ust.com

2. **Leader Election**: The leader processes write requests and replicates changes to followers.
3. **Data Persistence**: Data is written to disk on each node for durability.

Setting Up ETCD

Installation

ETCD can be installed via package managers or binaries. For example, on Linux:

```
wget https://github.com/etcd-io/etcd/releases/download/<version>/etcd-<version>-linux-amd64.tar.gz

tar -xvf etcd-<version>-linux-amd64.tar.gz

sudo mv etcd /usr/local/bin/
```

Cluster Configuration

1. **Single Node**: Useful for development environments.
2. **Multi-Node Cluster**: Recommended for production to ensure high availability.

Example of starting an ETCD cluster:

```
etcd --name node1 --initial-advertise-peer-urls http://<node1-ip>:2380 \

    --listen-peer-urls http://<node1-ip>:2380 \

    --listen-client-urls http://<node1-ip>:2379,http://127.0.0.1:2379 \

    --initial-cluster-token etcd-cluster-1 \

    --initial-cluster node1=http://<node1-ip>:2380,node2=http://<node2-ip>:2380 \

    --initial-cluster-state new
```

Key ETCD Operations

CRUD Operations

- **Create**: etcdctl put <key> <value>
- **Read**: etcdctl get <key>
- **Update**: Similar to create, as put overwrites the value.

Sri Ganesh Reddy Yerra Kondappagari
290685@ust.com

- **Delete**: etcdctl del <key>

Watch

Monitor changes to specific keys:

```
etcdctl watch <key>
```

Transactions

ETCD supports atomic transactions to group multiple operations:

```
etcdctl txn <<EOF
if
key "foo" = "bar"
then
put "status" "updated"
else
put "status" "failed"
end
EOF
```

Use Cases of ETCD

1. **Configuration Management**: Store and distribute configuration files across clusters.
2. **Service Discovery**: Applications register themselves with ETCD, enabling other services to discover them.
3. **Leader Election**: Elect a leader node in a distributed system using ETCD's strong consistency.
4. **Distributed Locks**: Ensure only one instance of a task runs at a time.

Maintenance and Monitoring

Backups

Create regular backups of your ETCD data:

```
etcdctl snapshot save <backup-file>
```

Sri Ganesh Reddy Yerra Kondappagari
290685@ust.com

Restore from a snapshot:

```
etcdctl snapshot restore <backup-file>
```

Monitoring

- **Metrics**: Use the ETCD metrics endpoint (/metrics) for monitoring.
- **Logs**: Analyze ETCD logs for troubleshooting.
- **Alerting**: Set up alerts for key metrics, such as latency or leader changes.

Performance Optimization

1. **Hardware**: Use SSDs for faster disk operations.
2. **Cluster Size**: Optimal size is 3, 5, or 7 nodes to maintain quorum.
3. **Request Load**: Distribute read operations across followers to reduce the leader's load.

Conclusion

ETCD is a robust, feature-rich solution for distributed systems, ensuring consistency, availability, and scalability. Its capabilities make it a critical component in modern cloud-native architectures, especially in systems like Kubernetes. By following best practices in setup and maintenance, ETCD can provide reliable performance and support for your distributed applications.