etcd/clientv3

docs latest go documentation

`etcd/clientv3` is the official Go etcd client for v3.

Install

```
go get go.etcd.io/etcd/client/v3
```

Warning: As etcd 3.5.0 was not yet released, the command above does not work. After first pre-release of 3.5.0 #12498, etcd can be referenced using:

```
go get go.etcd.io/etcd/client/v3@v3.5.0-pre
```

Get started

Create client using `clientv3.New`:

```go
cli, err := clientv3.New(clientv3.Config{
    Endpoints:   []string{"localhost:2379", "localhost:22379", "localhost:32379"},
    DialTimeout: 5 * time.Second,
})
if err != nil {
    // handle error!
}
defer cli.Close()
```

etcd v3 uses `gRPC` for remote procedure calls. And `clientv3` uses `grpc-go` to connect to etcd. Make sure to close the client after using it. If the client is not closed, the connection will have leaky goroutines. To specify client request timeout, pass `context.WithTimeout` to APIs:

```go
ctx, cancel := context.WithTimeout(context.Background(), timeout)
resp, err := cli.Put(ctx, "sample_key", "sample_value")
cancel()
if err != nil {
    // handle error!
}
// use the response
```

For full compatibility, it is recommended to install released versions of clients using go modules.

Error Handling

etcd client returns 2 types of errors:

1. context error: canceled or deadline exceeded.

2. gRPC error: see api/v3rpc/rpctypes.

Here is the example code to handle client errors:

```
resp, err := cli.Put(ctx, "", "")
if err != nil {
    switch err {
    case context.Canceled:
        log.Fatalf("ctx is canceled by another routine: %v", err)
    case context.DeadlineExceeded:
        log.Fatalf("ctx is attached with a deadline is exceeded: %v", err)
    case rpctypes.ErrEmptyKey:
        log.Fatalf("client-side error: %v", err)
    default:
        log.Fatalf("bad cluster endpoints, which are not etcd servers: %v", err)
    }
}
```

Metrics

The etcd client optionally exposes RPC metrics through go-grpc-prometheus. See the examples.

Namespacing

The namespace package provides `clientv3` interface wrappers to transparently isolate client requests to a user-defined prefix.

Request size limit

Client request size limit is configurable via `clientv3.Config.MaxCallSendMsgSize` and `MaxCallRecvMsgSize` in bytes. If none given, client request send limit defaults to 2 MiB including gRPC overhead bytes. And receive limit defaults to `math.MaxInt32`.

Examples

More code examples can be found at GoDoc.