klog

klog is a permanent fork of https://github.com/golang/glog.

Why was klog created?

The decision to create klog was one that wasn't made lightly, but it was necessary due to some drawbacks that are present in glog. Ultimately, the fork was created due to glog not being under active development; this can be seen in the glog README:

> The code in this repo [...] is not itself under development

This makes us unable to solve many use cases without a fork. The factors that contributed to needing feature development are listed below:

- `glog` presents a lot "gotchas" and introduces challenges in containerized environments, all of which aren't well documented.
- `glog` doesn't provide an easy way to test logs, which detracts from the stability of software using it
- A long term goal is to implement a logging interface that allows us to add context, change output format, etc.

Historical context is available here:

- https://github.com/kubernetes/kubernetes/issues/61006
- https://github.com/kubernetes/kubernetes/issues/70264
- https://groups.google.com/forum/#!msg/kubernetes-sig-architecture/wCWiWf3Juzs/hXRVBH90CgAJ
- https://groups.google.com/forum/#!msg/kubernetes-dev/7vnijOMhLS0/1oRiNtigBgAJ

Release versioning

Semantic versioning is used in this repository. It contains several Go modules with different levels of stability: - `k8s.io/klog/v2` - stable API, `vX.Y.Z` tags - `k8s.io/hack/tools` - no stable API yet (may change eventually or get moved to separate repo), `hack/tools/v0.Y.Z` tags - `examples` - no stable API, no tags, no intention to ever stabilize

Exempt from the API stability guarantee are items (packages, functions, etc.) which are marked explicitly as `EXPERIMENTAL` in their docs comment. Those may still change in incompatible ways or get removed entirely. This can only be used for code that is used in tests to avoid situations where non-test code from two different Kubernetes dependencies depends on incompatible releases of klog because an experimental API was changed.

How to use klog

- Replace imports for `"github.com/golang/glog"` with `"k8s.io/klog/v2"`
- Use `klog.InitFlags(nil)` explicitly for initializing global flags as we no longer use `init()` method to register the flags
- You can now use `log_file` instead of `log_dir` for logging to a single file (See `examples/log_file/usage_log_file.go`)
- If you want to redirect everything logged using klog somewhere else (say syslog!), you can use `klog.SetOutput()` method and supply a `io.Writer`. (See `examples/set_output/usage_set_output.go`)
- For more logging conventions (See Logging Conventions)
- See our documentation on pkg.go.dev/k8s.io.

NOTE: please use the newer go versions that support semantic import versioning in modules, ideally go 1.11.4 or greater.

Coexisting with klog/v2

See this example to see how to coexist with both klog/v1 and klog/v2.

Coexisting with glog

This package can be used side by side with glog. This example shows how to initialize and synchronize flags from the global `flag.CommandLine` FlagSet. In addition, the example makes use of stderr as combined output by setting `alsologtostderr` (or `logtostderr`) to `true`.

Community, discussion, contribution, and support

Learn how to engage with the Kubernetes community on the community page.

You can reach the maintainers of this project at:

- Slack
- Mailing List

Code of conduct

Participation in the Kubernetes community is governed by the Kubernetes Code of Conduct.

---

glog

Leveled execution logs for Go.

This is an efficient pure Go implementation of leveled logs in the manner of the open source C++ package https://github.com/google/glog

By binding methods to booleans it is possible to use the log package without paying the expense of evaluating the arguments to the log. Through the -vmodule flag, the package also provides fine-grained control over logging at the file level.

The comment from glog.go introduces the ideas:

```
Package glog implements logging analogous to the Google-internal
C++ INFO/ERROR/V setup.  It provides functions Info, Warning,
Error, Fatal, plus formatting variants such as Infof. It
also provides V-style logging controlled by the -v and
-vmodule=file=2 flags.

Basic examples:

    glog.Info("Prepare to repel boarders")

    glog.Fatalf("Initialization failed: %s", err)

See the documentation of the V function for an explanation
of these examples:

    if glog.V(2) {
        glog.Info("Starting transaction...")
    }

    glog.V(2).Infoln("Processed", nItems, "elements")
```

The repository contains an open source version of the log package used inside Google. The master copy of the source lives inside Google, not here. The code in this repo is for export only and is not itself under development. Feature requests will be ignored.

Send bug reports to golang-nuts@googlegroups.com.