Example

This directory includes example logger setup allowing users to easily check and test impact of logging configuration.

Below we can see examples of how some features work.

Default

Run:

```
go run ./staging/src/k8s.io/component-base/logs/example/cmd/logger.go
```

Expected output:

```
I0329 11:36:38.734334   99095 logger.go:44] "Oops, I shouldn't be logging yet!"
This is normal output via stdout.
This is other output via stderr.
I0329 11:36:38.734575   99095 logger.go:76] Log using Infof, key: value
I0329 11:36:38.734592   99095 logger.go:77] "Log using InfoS" key="value"
E0329 11:36:38.734604   99095 logger.go:79] Log using Errorf, err: fail
E0329 11:36:38.734619   99095 logger.go:80] "Log using ErrorS" err="fail"
I0329 11:36:38.734631   99095 logger.go:82] Log with sensitive key, data: {"secret"}
I0329 11:36:38.734653   99095 logger.go:87] "Now the default logger is set, but using the
one from the context is still better."
I0329 11:36:38.734674   99095 logger.go:90] "Log sensitive data through context" data=
{Key:secret}
I0329 11:36:38.734693   99095 logger.go:94] "runtime" duration="1m0s"
I0329 11:36:38.734710   99095 logger.go:95] "another runtime" duration="1m0s"
```

JSON

Run:

```
go run ./staging/src/k8s.io/component-base/logs/example/cmd/logger.go --logging-format
json
```

Expected output:

```
I0329 11:38:01.782592   99945 logger.go:44] "Oops, I shouldn't be logging yet!"
This is normal output via stdout.
This is other output via stderr.
{"ts":1648546681782.9036,"caller":"cmd/logger.go:76","msg":"Log using Infof, key:
value\n","v":0}
{"ts":1648546681782.9392,"caller":"cmd/logger.go:77","msg":"Log using
InfoS","v":0,"key":"value"}
{"ts":1648546681782.9763,"caller":"cmd/logger.go:79","msg":"Log using Errorf, err: fail\n"}
{"ts":1648546681782.9915,"caller":"cmd/logger.go:80","msg":"Log using ErrorS","err":"fail"}
{"ts":1648546681783.0208,"caller":"cmd/logger.go:82","msg":"Log with sensitive key, data:
{\"secret\"}\n","v":0}
{"ts":1648546681783.0364,"caller":"cmd/logger.go:87","msg":"Now the default logger is set,
but using the one from the context is still better.","v":0}
{"ts":1648546681783.0552,"caller":"cmd/logger.go:90","msg":"Log sensitive data through
context","v":0,"data":{"key":"secret"}}
```

```
{"ts":1648546681783.1091,"caller":"cmd/logger.go:94","msg":"runtime","v":0,"duration":"1m0s"}

{"ts":1648546681783.1257,"caller":"cmd/logger.go:95","msg":"another
runtime","v":0,"duration":"1h0m0s","duration":"1m0s"}
```

Verbosity

```
go run ./staging/src/k8s.io/component-base/logs/example/cmd/logger.go -v1
```

The expected output now includes `Log less important message`:

```
I0329 11:38:23.145695  100190 logger.go:44] "Oops, I shouldn't be logging yet!"
This is normal output via stdout.
This is other output via stderr.
I0329 11:38:23.145944  100190 logger.go:76] Log using Infof, key: value
I0329 11:38:23.145961  100190 logger.go:77] "Log using InfoS" key="value"
E0329 11:38:23.145973  100190 logger.go:79] Log using Errorf, err: fail
E0329 11:38:23.145989  100190 logger.go:80] "Log using ErrorS" err="fail"
I0329 11:38:23.146000  100190 logger.go:82] Log with sensitive key, data: {"secret"}
I0329 11:38:23.146017  100190 logger.go:83] Log less important message
I0329 11:38:23.146034  100190 logger.go:87] "Now the default logger is set, but using the
one from the context is still better."
I0329 11:38:23.146055  100190 logger.go:90] "Log sensitive data through context" data=
{Key:secret}
I0329 11:38:23.146074  100190 logger.go:94] "runtime" duration="1m0s"
I0329 11:38:23.146091  100190 logger.go:95] "another runtime" duration="1m0s"
```

Contextual logging

Contextual logging enables the caller of the function to add a string prefix and additional key/value pairs to a logger and then pass the updated logger into functions via a `context` parameter.

At the moment, this functionality is controlled in Kubernetes with the `ContextualLogging` feature gate and disabled by default. `klog.LoggerWithValues`, `klog.LoggerWithName`, `klog.NewContext` just return the original instance when contextual logging is disabled. `klog.FromContext` doesn't check the context for a logger and instead returns the global logger.

```
go run ./staging/src/k8s.io/component-base/logs/example/cmd/logger.go --feature-gates
ContextualLogging=true
```

The expected output now includes `example` (added by caller) and `myname` (added by callee) as prefix and the caller's `foo="bar"` key/value pair:

```
I0329 11:47:36.830458  101057 logger.go:44] "Oops, I shouldn't be logging yet!"
This is normal output via stdout.
This is other output via stderr.
I0329 11:47:36.830715  101057 logger.go:76] Log using Infof, key: value
I0329 11:47:36.830731  101057 logger.go:77] "Log using InfoS" key="value"
E0329 11:47:36.830745  101057 logger.go:79] Log using Errorf, err: fail
E0329 11:47:36.830760  101057 logger.go:80] "Log using ErrorS" err="fail"
I0329 11:47:36.830772  101057 logger.go:82] Log with sensitive key, data: {"secret"}
I0329 11:47:36.830795  101057 logger.go:87] "Now the default logger is set, but using the
```

```
one from the context is still better."
I0329 11:47:36.830818  101057 logger.go:90] "example: Log sensitive data through context"
foo="bar" data={Key:secret}
I0329 11:47:36.830841  101057 logger.go:94] "example/myname: runtime" foo="bar"
duration="1m0s"
I0329 11:47:36.830859  101057 logger.go:95] "example: another runtime" foo="bar"
duration="1m0s"
```