记录和管理 API 日志

本节核心内容

- Go 日志包数量众多，功能不同、性能不同，本小册介绍一个笔者认为比较好的日志库，并给出原因
- 介绍如何初始化日志包
- 介绍如何调用日志包
- 介绍如何转存（rotate）日志文件

> 本小节源码下载路径：[demo03 (https://github.com/lexkong/apiserver_demos/tree/mas](https://github.com/lexkong/apiserver_demos/tree/mas)
>
> 可先下载源码到本地，结合源码理解后续内容，边学边练。
>
> 本小节的代码是基于 [demo02 (https://github.com/lexkong/apiserver_demos/tree/mas](https://github.com/lexkong/apiserver_demos/tree/mas) 来开发的。

日志包介绍

apiserver 所采用的日志包 [lexkong/log (https://github.com/lexkong/log)](https://github.com/lexkong/log) 是笔者根据开发经验，并调研 GitHub 上的 开源log 包后封装的一个日志包，也是笔者所在项目使用的日志包。它参考华为 [paas-lager (https://github.com/ServiceComb/paas-lager)](https://github.com/ServiceComb/paas-lager)，做了一些便捷性的改动，功能完全一样，只不过更为便捷。相较于 Go 的其他日志包，该日志包有如下特点：

- 支持日志输出流配置，可以输出到 stdout 或 file，也可以同时输出到 stdout 和 file
- 支持输出为 JSON 或 plaintext 格式
- 支持彩色输出
- 支持 log rotate 功能
- 高性能

初始化日志包

在 conf/config.yaml 中添加 log 配置

在 config/config.go 中添加日志初始化代码

```
package config

import (
    ....
    "github.com/lexkong/log"
    ....
)
....
func Init(cfg string) error {
    ....
```

```
    // 初始化配置文件
    if err := c.initConfig(); err != nil {
        return err
    }

    // 初始化日志包
    c.initLog()
    ....
}

func (c *Config) initConfig() error {
    ....
}

func (c *Config) initLog() {
    passLagerCfg := log.PassLagerCfg {
        Writers:
viper.GetString("log.writers"),
        LoggerLevel:
viper.GetString("log.logger_level"),
        LoggerFile:
viper.GetString("log.logger_file"),
        LogFormatText:
viper.GetBool("log.log_format_text"),
        RollingPolicy:
viper.GetString("log.rollingPolicy"),
        LogRotateDate:
viper.GetInt("log.log_rotate_date"),
        LogRotateSize:
viper.GetInt("log.log_rotate_size"),
        LogBackupCount:
viper.GetInt("log.log_backup_count"),
    }
```

```
    log.InitWithConfig(&passLagerCfg)
}

// 监控配置文件变化并热加载程序
func (c *Config) watchConfig() {
    ....
}
```

这里要注意，日志初始化函数 c.initLog() 要放在配置初始化函数 c.initConfig() 之后，因为日志初始化函数要读取日志相关的配置。func (c *Config) initLog() 是日志初始化函数，会设置日志包的各项参数，参数为：

- writers：输出位置，有两个可选项 —— file 和 stdout。选择 file 会将日志记录到 logger_file 指定的日志文件中，选择 stdout 会将日志输出到标准输出，当然也可以两者同时选择
- logger_level：日志级别，DEBUG、INFO、WARN、ERROR、FATAL
- logger_file：日志文件
- log_format_text：日志的输出格式，JSON 或者 plaintext，true 会输出成非 JSON 格式，false 会输出成 JSON 格式
- rollingPolicy：rotate 依据，可选的有 daily 和 size。如果选 daily 则根据天进行转存，如果是 size 则根据大小进行转存
- log_rotate_date：rotate 转存时间，配合rollingPolicy: daily 使用
- log_rotate_size：rotate 转存大小，配合 rollingPolicy: size 使用
- log_backup_count：当日志文件达到转存标准时，log 系统会将该日志文件进行压缩备份，这里指定了备份文件的最大个

数

调用日志包

日志初始化好了，将 [demo02](https://github.com/lexkong/apiserver_demos/tree/master/c) 中的 log 用 [lexkong/log (https://github.com/lexkong/log)](https://github.com/lexkong/log) 包来替换。替换前（这里 grep 出了需要替换的行，读者可自行确认替换后的效果）：

```
$ grep log * -R
config/config.go:   "log"
config/config.go:       log.Printf("Config file
changed: %s", e.Name)
main.go:    "log"
main.go:            log.Fatal("The router has no
response, or it might took too long to start
up.", err)
main.go:        log.Print("The router has been
deployed successfully.")
main.go:    log.Printf("Start to listening the
incoming requests on http address: %s",
viper.GetString("addr"))
main.go:
log.Printf(http.ListenAndServe(viper.GetString("a
ddr"), g).Error())
main.go:        log.Print("Waiting for the
router, retry in 1 second.")
```

替换后的源码文件见 [demo03](https://github.com/lexkong/apiserver_demos/tree/master/c)

编译并运行

1. 下载 apiserver_demos 源码包（如前面已经下载过，请忽略此步骤）

```
$ git clone
https://github.com/lexkong/apiserver_demos
```

2. 将 apiserver_demos/demo03复制为 $GOPATH/src/apiserver

```
$ cp -a apiserver_demos/demo03/
$GOPATH/src/apiserver
```

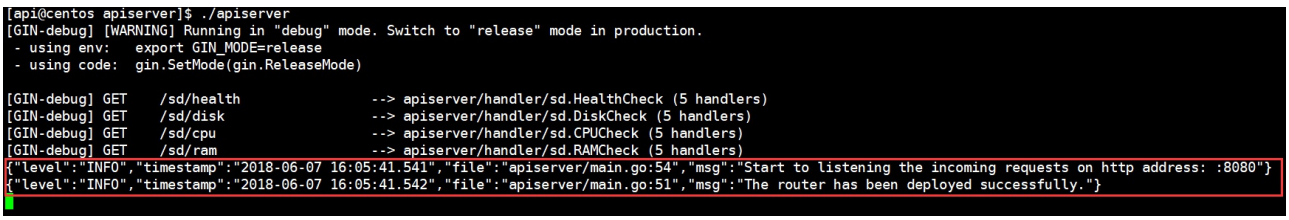
3. 在 apiserver 目录下编译源码

```
$ cd $GOPATH/src/apiserver
$ gofmt -w .
$ go tool vet .
$ go build -v .
```

4. 启动 apiserver

```
$ ./apiserver
```

启动后，可以看到 apiserver 有 JSON 格式的日志输出：

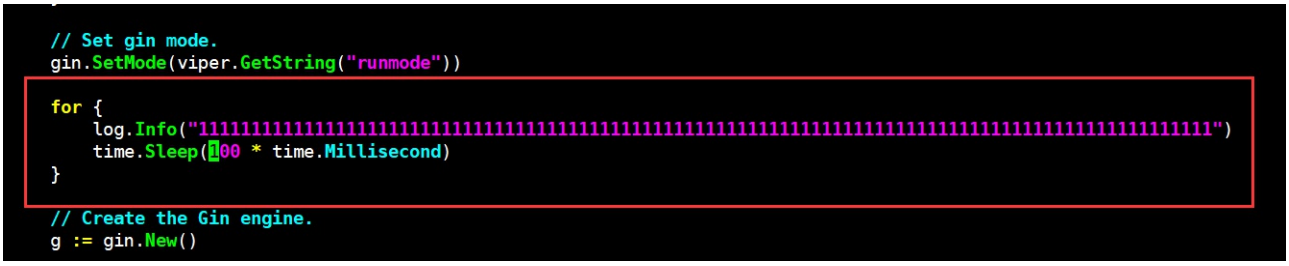


管理日志文件

这里将日志转存策略设置为 size，转存大小设置为 1 MB

```
  rollingPolicy: size
  log_rotate_size: 1
```

并在 main 函数中加入测试代码：



启动 apiserver 后发现，在当前目录下创建了 log/apiserver.log 日志文件：

```
$ ls log/
apiserver.log
```

程序运行一段时间后，发现又创建了 zip 文件：

```
$ ls log/
apiserver.log
apiserver.log.20180531134509631.zip
```

该 zip 文件就是当 apiserver.log 大小超过 1MB 后，日志系统将之前的日志压缩成 zip 文件后的文件。

小结

本小节通过具体实例讲解了如何配置、使用和管理日志。