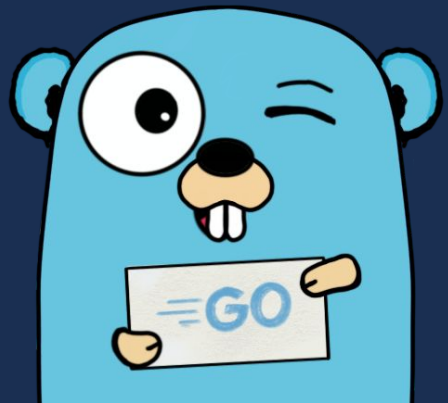
iSpring

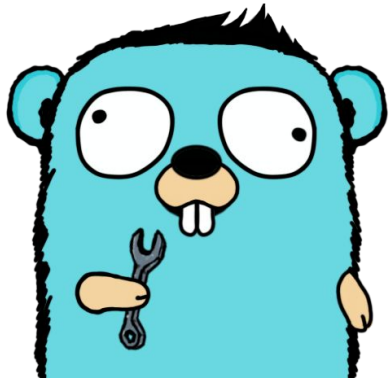
Настройка инструментов стандартной библиотеки Golang для production

Богданов Илья
iSpring



Обо мне и докладе

- 5 микросервисов на production за 2019 год
- 1 год опыта Go
- 4 года на C++
- Увлечения трехмерной графикой и микроконтроллерами
- Много шишек



О чем я буду рассказывать

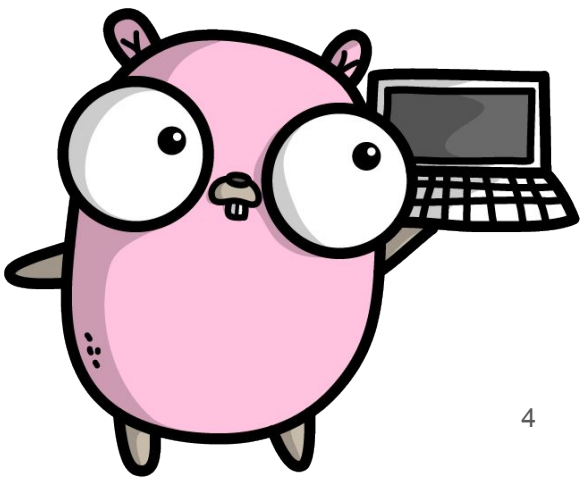
Доклад для начинающих разработчиков

- Как мы словили ошибки соединения с БД на production
- http.Client и что с ним не так
- http.Server и с чем его едят
- Пара бонусов

sql.DB

- connection pool.
- expired соединения чистятся отдельной горутиной каждую секунду.

```
db, err := sql.Open("mysql", "root:localhost/test")
if err != nil {
   return err
}
defer db.Close()

_, err := db.Exec("...")
```

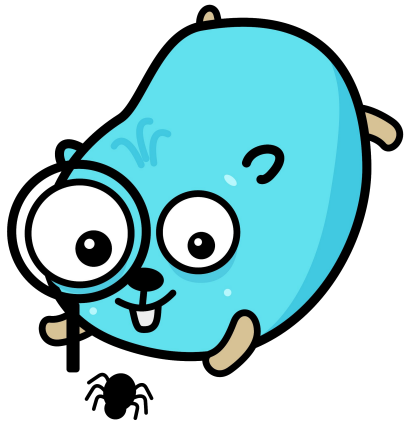


```
failed to begin a transaction: invalid connection

unexpected EOF
```

https://github.com/ashleymcnamara/gophers

func (*DB) SetMaxOpenConns(n int)

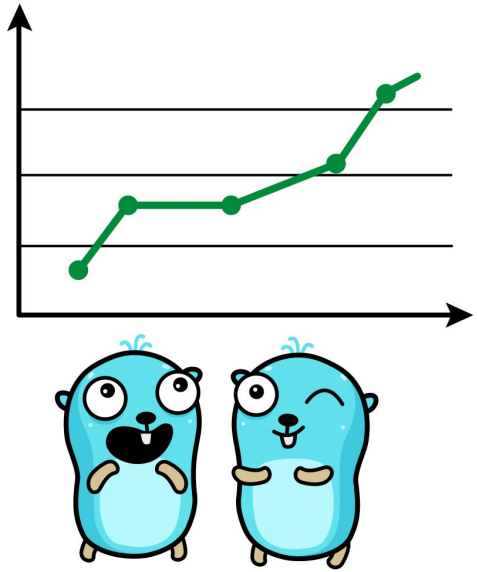
- Ограничивает максимальное количество соединений с базой
- Если n <= 0, количество соединений не ограничено
- По-умолчанию **0**



func (*DB) SetMaxOpenConns(n int)

- Ограничивает максимальное количество соединений с базой
- Если n <= 0, количество соединений не ограничено
- По-умолчанию **0**

SetMaxOpenConns	ns/op	B/op	allocs/op
1	497081	649	14
2	360376	650	14
3	251083	652	14
5	156420	652	14
10	110926	735	13
20	108629	719	12
0	110477	715	12



https://gist.github.com/Warboss-rus/2020ea4f0b1c6269304e7890d070c2da

func (*DB) SetMaxOpenConns(n int)

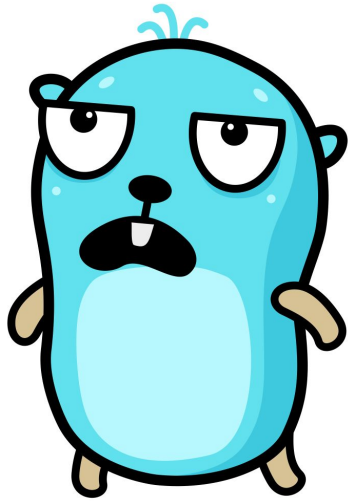
- Ограничивает максимальное количество соединений с базой
- Если n <= 0, количество соединений не ограничено
- По-умолчанию **0**

```
[mysqld]
max_connections=10
```

```
Benchmark_MaxOpenConns20
    Benchmark_MaxOpenConns20: main_test.go:69: Error 1040: Too many connections
    Benchmark_MaxOpenConns20: main_test.go:69: Error 1040: Too many connections
    Benchmark_MaxOpenConns20: main_test.go:69: Error 1040: Too many connections
     ...
-- FAIL: Benchmark_MaxOpenConns20
FAIL
```

func (*DB) SetMaxIdleConns(n int)

- Количество соединений, которые могут остаться открытыми для переиспользования.
- Если 0 - соединения не будут повторно использоваться и на каждый запрос будет создаваться новое.
- По-умолчанию 2.



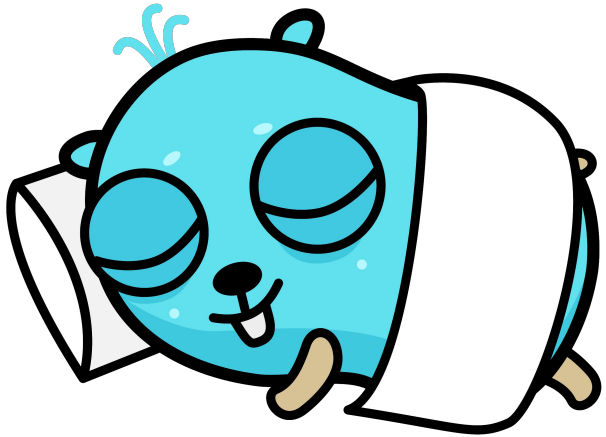
func (*DB) SetMaxIdleConns(n int)

- Количество соединений, которые могут остаться открытыми для переиспользования.
- Если 0 - соединения не будут повторно использоваться и на каждый запрос будет создаваться новое.
- По-умолчанию 2.

SetMaxIdleConns	ns/op	B/op	allocs/op
0	194023	6433	40
1	130755	1973	18
2	109399	709	12
5	106797	524	12
10	109346	524	12

func (*DB) SetConnMaxLifetime(d time.Duration)

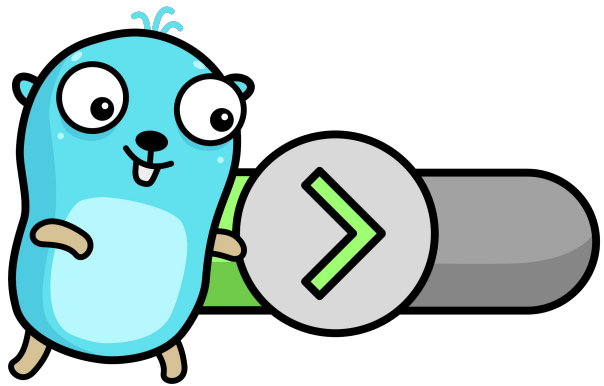
- Время, в течение которого соединение может быть переиспользовано.
- По-умолчанию 0 (не ограничено).



func (*DB) SetConnMaxLifetime(d time.Duration)

- Время, в течение которого соединение может быть переиспользовано.
- По-умолчанию 0 (не ограничено).

SetConnMaxLifetime	ns/op	B/op	allocs/op
100 µs	613533	6465	42
1 ms	537581	5340	37
10 ms	172004	1130	15
100 ms	129907	728	12
1 s	130169	724	12
0 (unlimited)	127519	712	12



Что в итоге?

- Отсутствует лимит на количество соединений
- Соединение может переиспользоваться после долгого простоя
- Долгоживущее соединение разрывается базой в одностороннем порядке
- Ограничение времени жизни соединений и максимального их количества решает проблему

Как подобрать настройки для моего сервиса?

- Оцените нагруженность сервиса
- Проанализируйте конфигурацию базы данных
- func (*DB) Stats() - вернет информацию о текущих соединениях и статистику по ожиданию соединения
- Читайте документацию к языку!

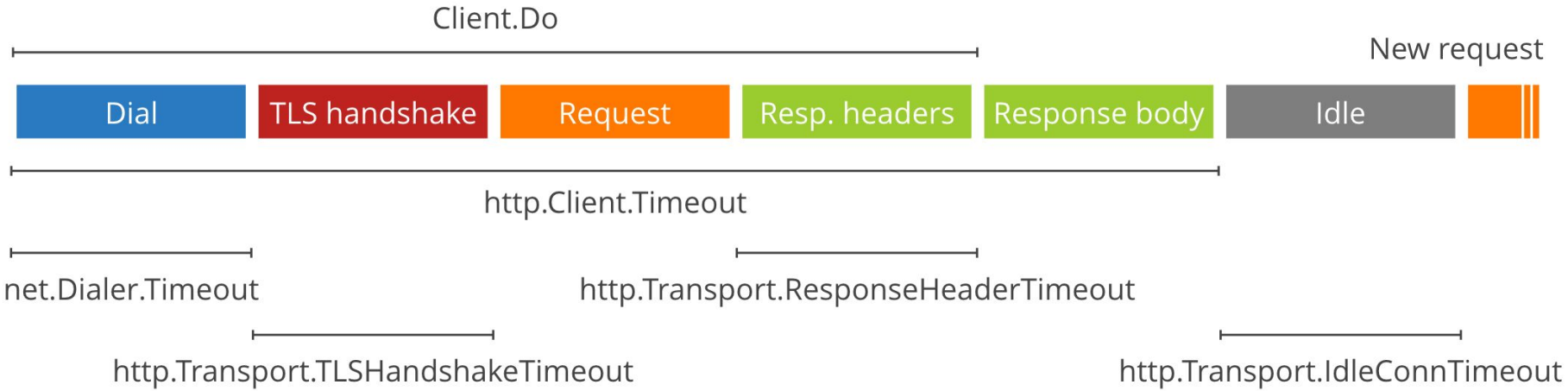
Пример конфигурации для низконагруженного сервиса

```
db, err := sql.Open("mysql", "root:localhost/test")
if err != nil {
   return err
}
defer db.Close()

db.SetMaxOpenConns(25)
db.SetMaxIdleConns(2)
db.SetConnMaxLifetime(time.Minute)

_, err := db.Exec("...")
```

http.Client

- Не имеет таймаута по-умолчанию!
- Странное поведение Keep-Alive

Таймауты http.Client



Динамический таймаут

```
c := &http.Client{}
resp, _ := c.Get("https://ispringsolutions.com")
defer resp.Body.Close()

timer := time.AfterFunc(5*time.Second, func() {
   resp.Body.Close()
})

bodyBytes := make([]byte, 0)
for {
   timer.Reset(5 * time.Second)

   _, err = io.CopyN(bytes.NewBuffer(bodyBytes), resp.Body, 256)
   if err == io.EOF {
       break // данные закончились, выходим
   } else if err != nil {
       panic(err)
   }
}
```

Повторное использование соединений (Keep-Alive)

- *http.Transport:* до 100 соединений с таймаутом 90 секунд
- 2 соединения на хост (*MaxIdleConnsPerHost*)
- Если сервис работает только с одним хостом, либо производится нагрузочное тестирование имеет смысл увеличить *MaxIdleConnsPerHost*
- Если тело ответа не было считано, то Keep-Alive ломается

```
res, err := client.Do(req)
io.Copy(ioutil.Discard, res.Body)
res.Body.Close()
```

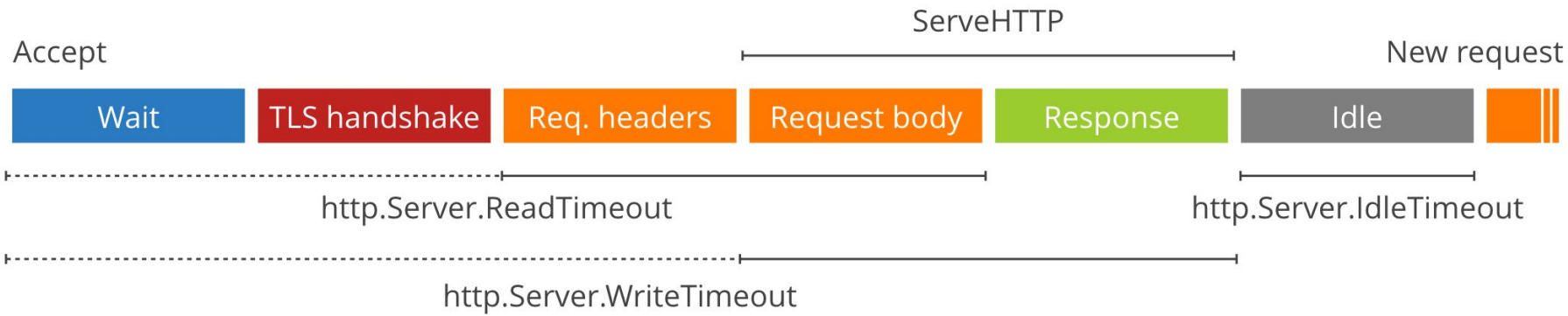
Примеры конфигов

- Для API - **http.Client.Timeout = 5*time.Second**
- Для скачивания и загрузки файлов:
  - **динамический таймаут** 5 *time.Second
  - **net.Dialer.Timeout** 1*time.Second
  - **http.Transport.ResponseHeaderTimeout** 5*time.Second

http.Server

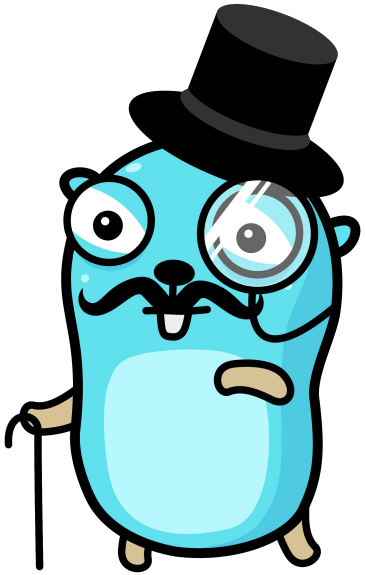
Без таймаутов по-умолчанию!

http.Server


ServeHTTP
Accept
New request
Wait
TLS handshake
Req. headers
Request body
Response
Idle
http.Server.ReadTimeout
http.Server.IdleTimeout
http.Server.WriteTimeout

Продвинутые таймауты http.Server

- **http.TimeoutHandler** - обертка над http.Handler
- **http.Server.ReadHeaderTimeout** - таймаут до окончания чтения заголовков запроса. Несовместим с ReadTimeout
- **Динамический таймаут**
- **context.Context**



Примеры конфигов

Только API:

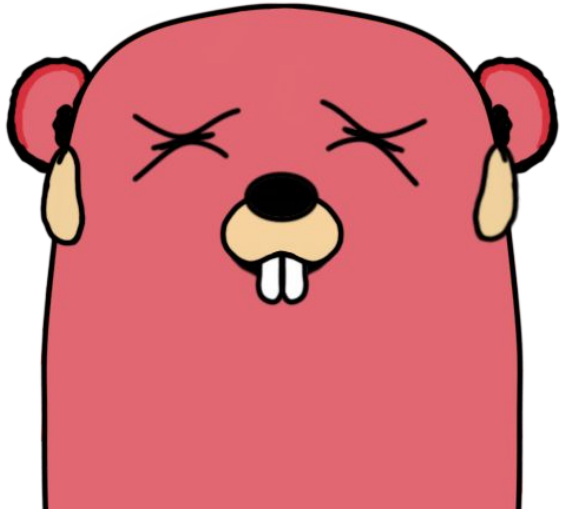
```
srv := &http.Server{
    ReadTimeout:  5 * time.Second,
    WriteTimeout: 10 * time.Second,
    IdleTimeout:  120 * time.Second,
}
```

Файлы, стриминг:

```
srv := &http.Server{
    ReadHeaderTimeout: 5 * time.Second,
    IdleTimeout:      120 * time.Second,
}

//+динамический таймаут
```

Бонус: пользовательские блокировки MySQL

- MySQL: GET_LOCK, RELEASE_LOCK
- Ошибка при попытке освободить lock:
  `cannot release a lock that is not acquired`
- проблема: освобождать лок нужно в том же соединении, в котором вы его заблокировали
- решение: блокировать внутри транзакции



Бонус: пользовательские блокировки MySQL

БЫЛО

```
var result int
err := db.Get(&result, “SELECT GET_LOCK(`lockname`, 60)”)
// execute some sql transaction
err := db.Get(&result, “SELECT RELEASE_LOCK(`lockname`)”)
if result != 1 {
     panic(“cannot release a lock that is not acquired”)
}
```

СТАЛО

```
tx, err := db.Begin()
var result int
err := tx.Get(&result, “SELECT GET_LOCK(`lockname`, 60)”)
// execute some sql commands
err := tx.Get(&result, “SELECT RELEASE_LOCK(`lockname`)”)
if result != 1 {
     panic(“cannot release a lock that is not acquired”)
}
err := tx.Commit();
```

Бонус: реконнект к amqp

Официальный клиент: github.com/streadway/amqp

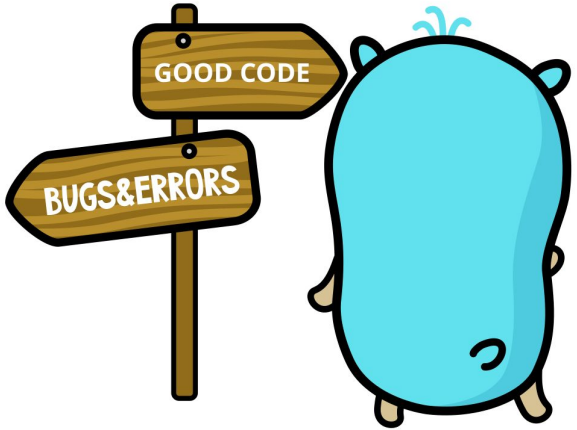
Не поддерживает автоматический реконнект

func (c *Connection) NotifyClose(receiver chan *Error) chan *Error

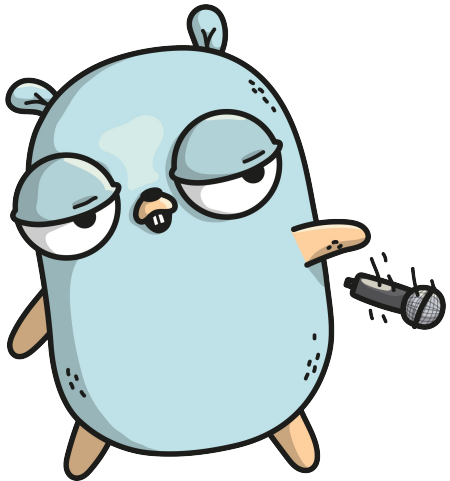
```
func connect() err {
  conn, err := amqp.Dial(“url”);
  connErrorChan := conn.NotifyClose(make(chan *amqp.Error))
  go processConnectErrors(connErrorChan)
  return err
}
func processConnectErrors(ch chan *amqp.Error) {
  err := <- ch
  for {
    err := connect()
    if err != nil {
      break
    }
  }
}
```

Выводы

- Не полагайтесь на стандартные конфигурации
- Читайте документацию и умные статьи :)
- Тестируйте и профилируйте микросервисы



Спасибо за внимание



Ссылки

Гоферы :)

- https://github.com/MariaLetta/free-gophers-pack
- https://github.com/ashleymcnamara/gophers
- https://quasilyte.dev/gopherkon

Бенчмарк:

- https://gist.github.com/Warboss-rus/2020ea4f0b1c6269304e7890d070c2da

MySQL Memory Calculator:

- https://mysqlcalculator.com

Ссылки

Соединение с БД

- https://making.pusher.com/production-ready-connection-pooling-in-go/
- https://www.alexedwards.net/blog/configuring-sqldb

http.Client и http.Server

- https://blog.cloudflare.com/the-complete-guide-to-golang-net-http-timeouts/
- https://blog.gopheracademy.com/advent-2016/exposing-go-on-the-internet/
- https://medium.com/@simonfrey/go-as-in-golang-standard-net-http-config-will-break-your-production-environment-1360871cb72b
- http://tleyden.github.io/blog/2016/11/21/tuning-the-go-http-client-library-for-load-testing/