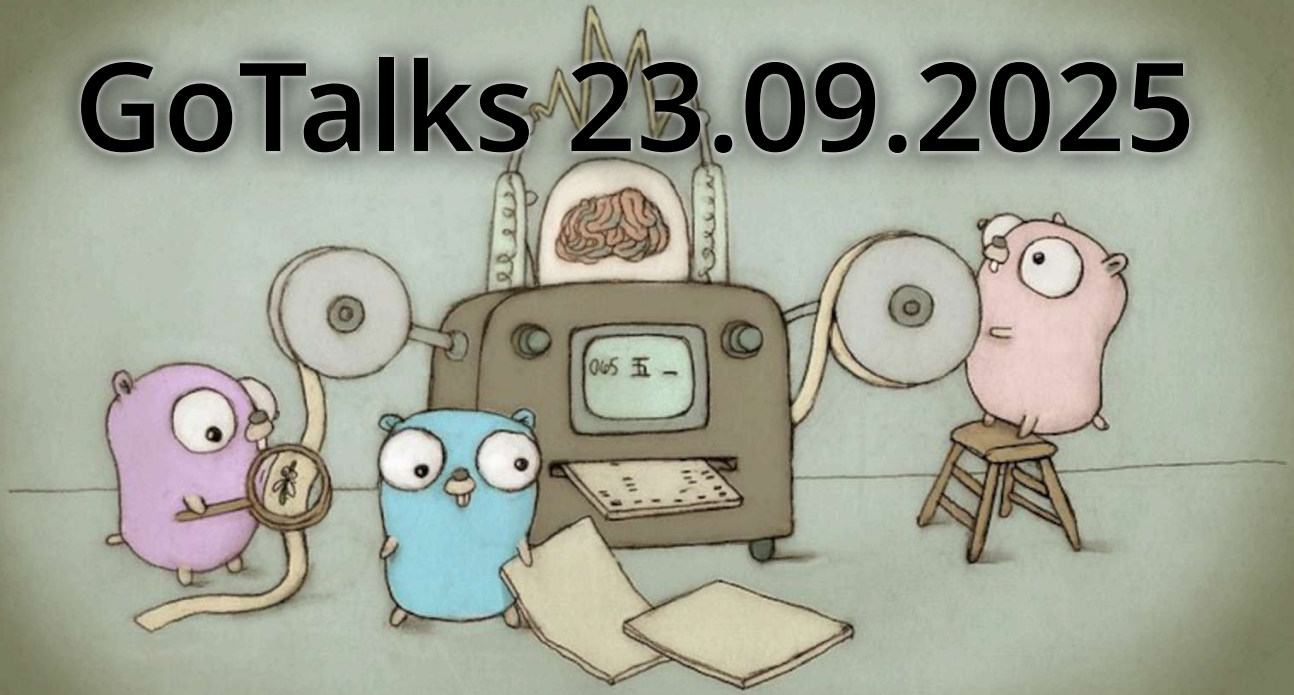
GoTalks 23.09.2025




TVZ

How to reach us

meetup.com/Golang-ZG

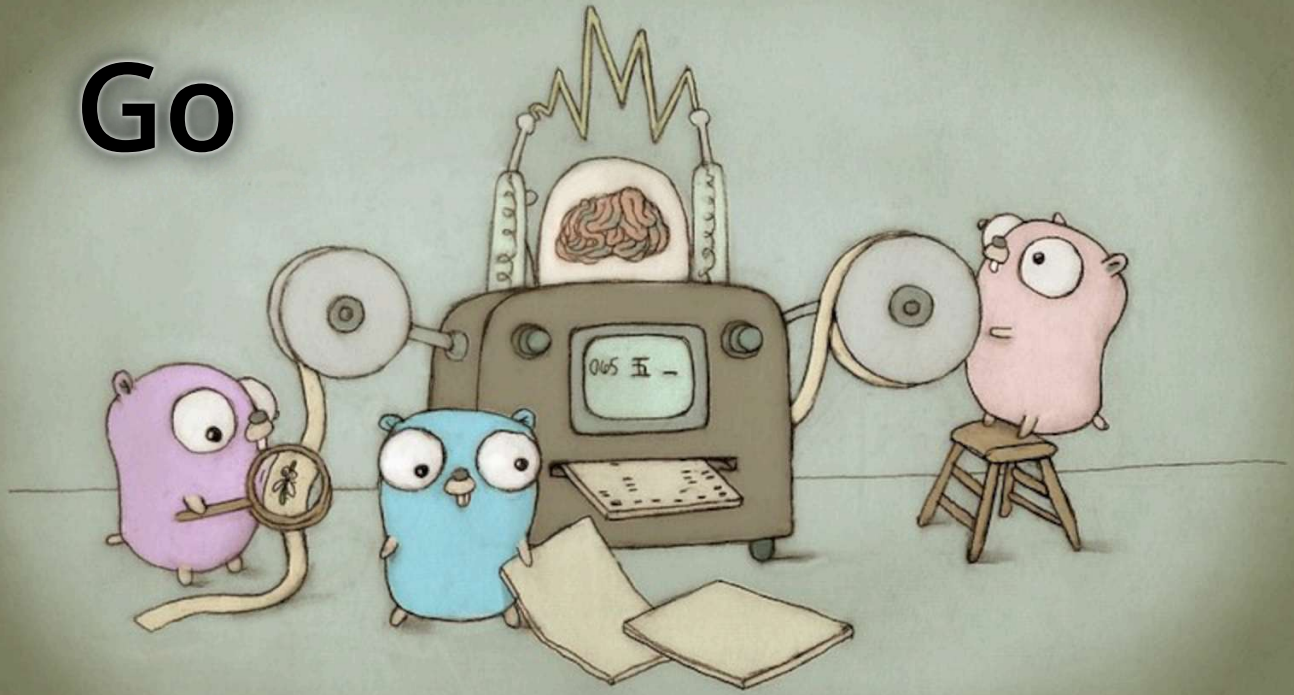
@golangzg

NEW github.com/golang-zg/meetups ⭐

@golangzg.bsky.social

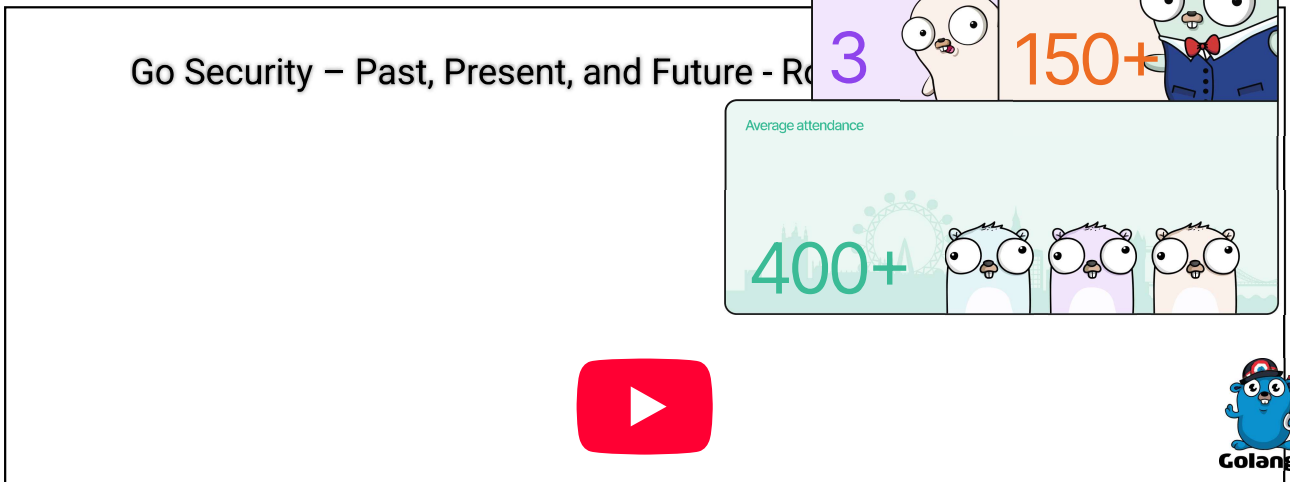
invite.slack.golangbridge.org

Go



GopherCon UK 2025

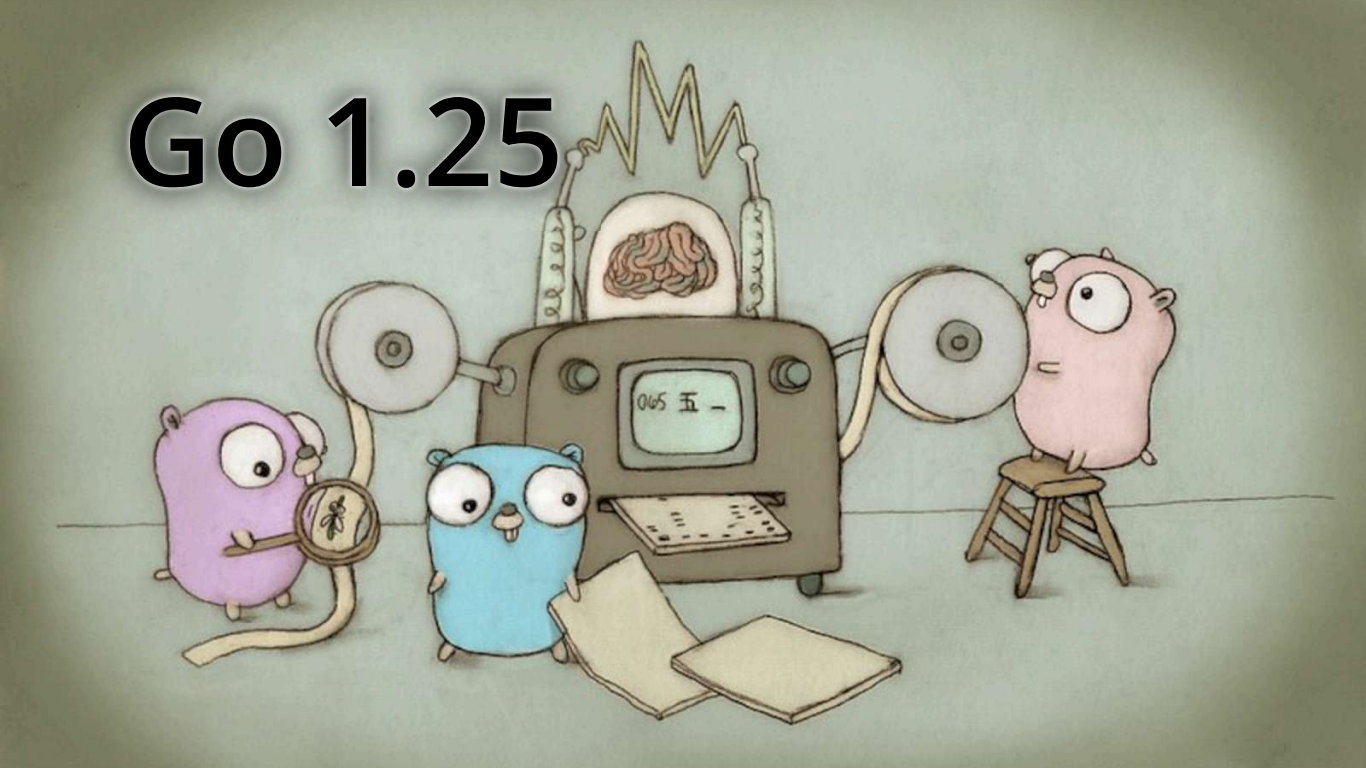
www.youtube.com/playlist?list=...



Go Developer Survey 2025

- How has Go has been working out for you?
  - go.dev/blog/survey2025-announce

Go 1.25



Container-aware `GOMAXPROCS`

- The default behavior of the `GOMAXPROCS` has changed.
  - In prior versions of Go, `GOMAXPROCS` defaults to runtime.NumCPU.
  - Go 1.25 introduces two changes:
    - On Linux, the runtime considers the CPU bandwidth limit of the cgroup containing the process.
    - **periodically** updates `GOMAXPROCS`
- disabled if `GOMAXPROCS` is set manually
  - `GOMAXPROCS` environment variable
  - call to `runtime.GOMAXPROCS`

New experimental garbage collector

- improves the performance of marking and scanning small objects through better locality and CPU scalability.
- expected somewhere between a `10–40%` reduction in garbage collection overhead in real-world programs that heavily use the garbage collector.

testing/synctest package

- Runs a test function in an isolated `bubble`. Within the bubble, time is virtualized
  - clock moves instantaneously if all goroutines in the bubble are blocked
- The Wait function waits for all goroutines in the current bubble to block.

```
func TestWait(t *testing.T) {
    synctest.Test(t, func(t *testing.T) {
        x := 0
        go func() {
            x = 1
            time.Sleep(time.Second)
            // ...
        }()

        synctest.Wait()
    })
}
```

experimental encoding/json/v2

- go.dev/blog/jsonv2-exp
- antonz.org/go-json-v2/
- new packages are available:
  - new `encoding/json/v2`
  - new `encoding/json/jsontext`
- when the “jsonv2” GOEXPERIMENT is enabled `encoding/json` package uses the new JSON implementation
- `encoding` performance is at parity between the implementations
- `decoding` is substantially faster in the new one

experimental encoding/json/v2

```
func Marshal(in any, opts ...Options) (out []byte, err error)
func MarshalWrite(out io.Writer, in any, opts ...Options) err
func MarshalEncode(out *jsontext.Encoder, in any, opts ...Opt

func Unmarshal(in []byte, out any, opts ...Options) error
func UnmarshalRead(in io.Reader, out any, opts ...Options) er
func UnmarshalDecode(in *jsontext.Decoder, out any, opts ...O
```

experimental encoding/json/v2

- json. `MarshalToFunc`

```
boolMarshaler := json.MarshalToFunc(
    func(enc *jsontext.Encoder, val bool) error {
        if val {
            return enc.WriteToken(jsontext.String("enabled"))
        }
        return enc.WriteToken(jsontext.String("disabled"))
    },
)
```

```
data, err := json.Marshal(vals, json.WithMarshalers(boolMarshaler
```

experimental encoding/json/v2

- json.`MarshalToFunc`

```
boolStrMarshaler := json.MarshalToFunc(
    func(enc *jsontext.Encoder, val string) error {
      if val == "enabled" || val == "true" {
          return enc.WriteToken(jsontext.String("true"))
      }
      if val == "disabled" || val == "false" {
          return enc.WriteToken(jsontext.String("false"))
      }
      return json.SkipFunc
    },
)
```

experimental encoding/json/v2

```
type Age struct {
    DOB time.Time `json:"dob,format:DateOnly"`
}
```

```
    a := Age{
        DOB: time.Date(
            2025, 9, 23, 18, 35, 0, 0, time.UTC),
    }
    b, err := json.Marshal(a)
    fmt.Println(string(b))
```

```
{"dob":"2025-09-23"}
```

experimental encoding/json/v2

```
type Image struct {
    Data []byte `json:"data"`
}
```

```
    img := Image{
        Data: []byte{0x01, 0x02, 0x03, 0x04, 0x05},
    }
```

```
{"data":"AQIDBAU="}
```

experimental encoding/json/v2

```
type ImageArray struct {
    Data []byte `json:"data,format:array"`
}
```

```
    b, err := json.Marshal(ImageArray{
        Data: img.Data,
    })
    fmt.Println(string(b))
    if err != nil {
        fmt.Println(err)
    }
```

```
{"data":[1,2,3,4,5]}
```

experimental encoding/json/v2

```go
type ImageBase64 struct {
    Data []byte `json:"data,format:base64"`
}
```

```go
    b, err := json.Marshal(ImageBase64{
        Data: img.Data,
    })
    fmt.Println(string(b))
    if err != nil {
        fmt.Println(err)
    }
```

```
{"data":"AQIDBAU="}
```

experimental encoding/json/v2

```
type ImageHex struct {
    Data []byte `json:"data,format:hex"`
}
```

```
    b, err := json.Marshal(ImageHex{
        Data: img.Data,
    })
    fmt.Println(string(b))
    if err != nil {
        fmt.Println(err)
    }
```

```
{"data":"0102030405"}
```

experimental encoding/json/v2

```
type Data struct {
    List []int           `json:"list"`
    Map  map[string]int `json:"map"`
}
```

```
    b, err := json.Marshal(d)
    fmt.Println(string(b))
    if err != nil {
        fmt.Println(err)
    }
```

```
{"list":[],"map":{}}
```

experimental encoding/json/v2

```go
type Data struct {
    List []int          `json:"list"`
    Map  map[string]int `json:"map"`
}
```

```go
    b, err := json.Marshal(d,
        json.FormatNilSliceAsNull(true),
        json.FormatNilMapAsN
    )
    fmt.Println(string(b))
```

```
{"list":null,"map":null}
```

experimental encoding/json/v2

```go
type Person struct {
    Name             string `json:"name"`
    AttendingMeetups bool
    // Collect all unknown Person fields
    Data map[string]any `json:",unknown"`
}
```

```go
    b, err := json.Marshal(
        jsontext.Multiline(
    )
    fmt.Println(string(b))
```

```
{
"name": "Zlatko",
"AttendingMeetups": true
}
```

experimental encoding/json/v2

```
type Durations struct {
  A time.Duration `json:"dob,format:units"`
  B time.Duration `json:"time,format:sec"`
  C time.Duration `json:"time,format:nano"`
  C time.Duration `json:"time,format:iso8601"`
}
```

experimental encoding/json/v2

```
type Age struct {
  DOB      time.Time `json:"dob     ,format:DateOnly"`
  Time     time.Time `json:"time    ,format:TimeOnly"`
  DateTime time.Time `json:"datetime,format:2006/01/02 15:04"`
  RFC      time.Time `json:"datetime,format:RFC3339"`
}
```

experimental encoding/json/v2

```
type Data struct {
  List1 []int          `json:"list,format:emitnull"`
  List2 []int          `json:"list,format:emitnull"`

  Map1  map[string]int `json:"map,format:emitempty"`
  Map2  map[string]int `json:"map,format:emitempty"`
}
```

sync.WaitGroup

```
wg.Add(1)
go func() {
    defer wg.Done()
    // code to run
}()

wg.Go(func() {
    // code to run
})

wg.Wait()
```

go mod - ignore

```
ignore (
  ./ui/node_modules
  static
)
```

```
go test ./...
go install ./...
```

Root

```
os.OpenInRoot(dir, name)
```

```
os.Root(dir)
```

- Chmod, Chown, Chtimes, Lchown, Link, MkdirAll, ReadFile, Readlink, RemoveAll, Rename, Symlink, WriteFile

other

- slices
  - The compiler can now allocate the backing store for slices on the stack in more situation.
- log/slog
  - GroupAttrs creates a group Attr from a slice of Attr values.
- CrossOriginProtection
  - implements protections against Cross-Site Request Forgery (CSRF) by rejecting non-safe cross-origin browser requests.
- go doc -http
  - starts a web server to serve documentation pages for Go packages.
- reflect.TypeAssert
  - permits converting a Value directly to a Go value of the given type (avoids interface)