Context cancel

Common pitfalls

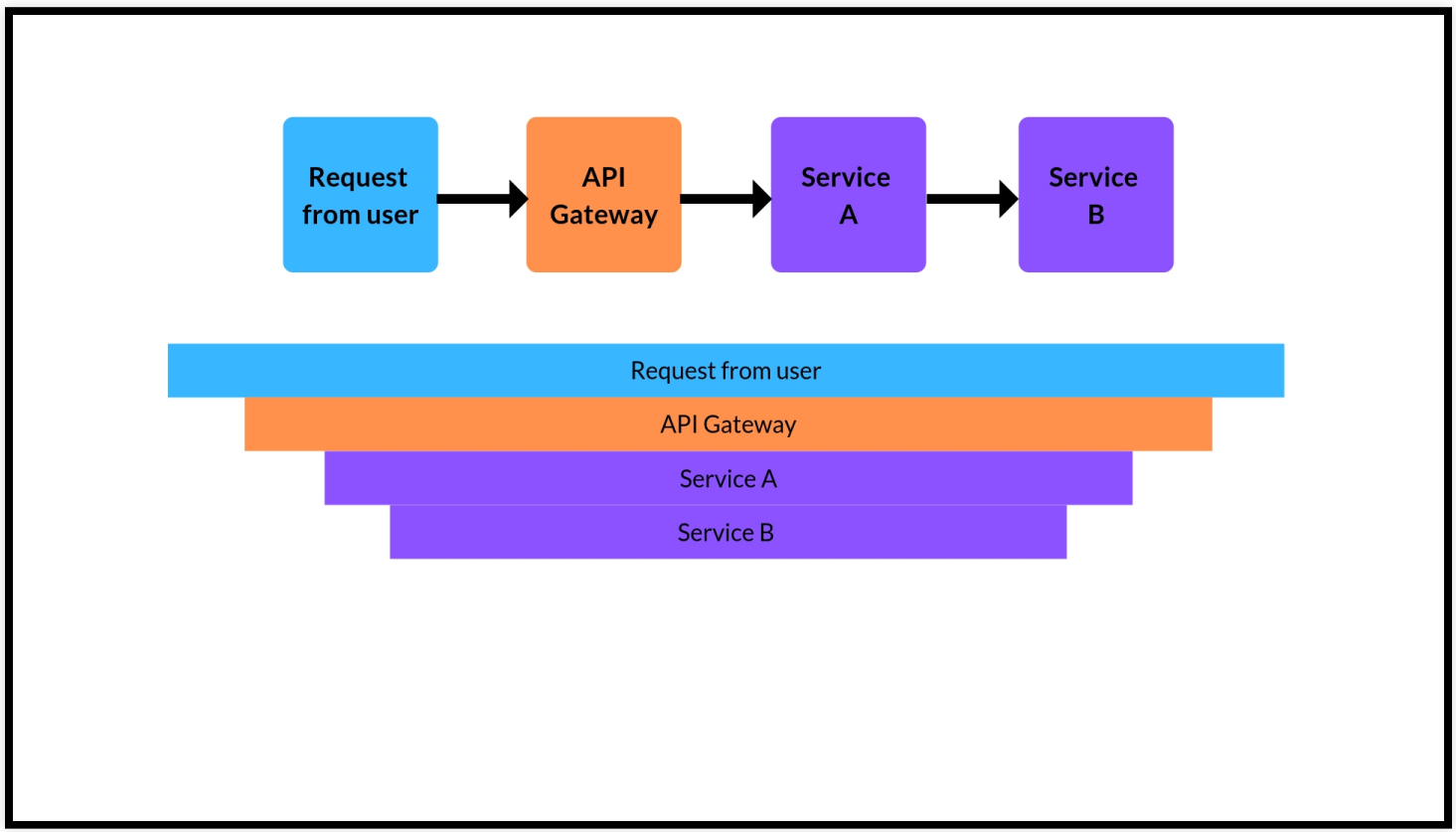
Tobiasz Heller

Ingrid

Definition

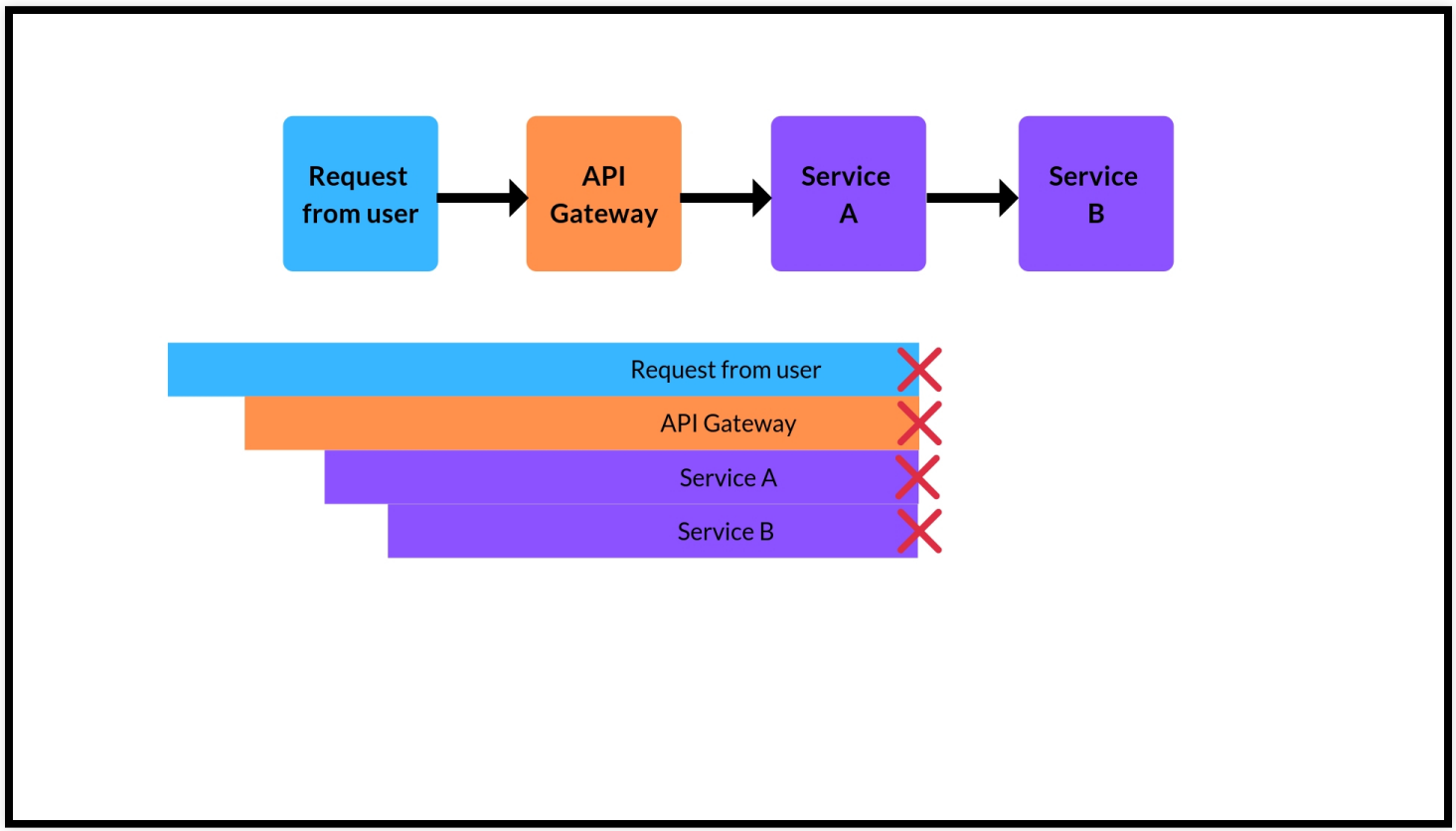
Context carries deadlines, cancelation signals, and other request-scoped values across API boundaries.

Why ctx cancel is useful?



Why ctx cancel is useful?

Why ctx cancel is useful?



How to cancel ctx

```
ctx, cancel := context.WithCancel(ctx)
```

```
req, err := http.NewRequest(http.MethodGet, "www.google.com", nil)
if err != nil {
    log.Fatal(err)
}
ctx, cancel := context.WithCancel(context.Background())
// cancel ctx, all calls with given ctx will be canceled immediately.
cancel()
res, err := http.DefaultClient.Do(req.WithContext(ctx))
if err != nil {
    log.Fatal(err)
    //output: Get www.google.com: context canceled
}
```

Context inspection

ctx.Err()

```
if ctx.Err() != nil {
    fmt.Printf("%v", err)
    //output: Get www.google.com: context canceled
}
```

<- ctx.Done()

```
select {
case <-ctx.Done():
    fmt.Printf("Context finished: %v\n", ctx.Err())
    //output: Context finished: context canceled
}
```

Common pitfalls

gRPC

Server

```
func (s *server) SayHello(ctx context.Context, in *pb.HelloRequest) ...
    go insertToCache(ctx, in)
    return &pb.HelloReply{Message: "Hello " + in.Name}, nil
}

func insertToCache(ctx context.Context, in *pb.HelloRequest) {
    select {
    case <-time.After(100 * time.Millisecond):
        fmt.Println("Inserted to cache")
    case <-ctx.Done():
        fmt.Printf("Insert canceled: %v\n", ctx.Err())
    }
}
```

Common pitfalls

gRPC

Client

```
...
c := pb.NewGreeterClient(conn)
r, err := c.SayHello(context.Background(), &pb.HelloRequest{Name: name})
if err != nil {
    log.Fatalf("could not greet: %v", err)
}
fmt.Printf("Greeting: %s\n", r.Message)
//Output: Greeting: Hello world
...
```

Server output:

```
Insert canceled: context canceled
```

Common pitfalls

gRPC

Context is a request scoped paradigm.

Common pitfalls

Metrics on fallback

```
// "github.com/prometheus/client_golang/prometheus"

type Counter interface {
    Inc()
    ...
}
```

Common pitfalls

Metrics on fallback

```
type geocoder interface {
    Geocode(context.Context, address) (*coordinates, error)
}

func geocode(ctx context.Context, a address, clients []geocoder) (*coordi
    for _, c := range clients {
        coords, err := c.Geocode(ctx, a)
        if err == nil {
            return coords, nil
        }
    }
    return nil, fmt.Errorf("failed to geocode address %v", a)
}
```

Common pitfalls

Metrics on fallback

```
type client struct {
    name  string
    sleep time.Duration
}

func (c *client) Geocode(ctx context.Context, a address) (*coordinates, e
    metrics.WithLabel(c.name).Inc()

    req, err := http.NewRequest(http.MethodGet, c.url, nil)
    req.Header.Set("x-sleep", c.sleep.String())
    ...
    resp, err := http.DefaultClient.Do(req.WithContext(ctx))
    ...
}
```

Common pitfalls

Metrics on fallback

```
clientA := client{
    name:  "client A",
    sleep: time.Second,
}
clientB := client{
    name:  "client B",
    sleep: time.Millisecond,
}
clients := []geocoder{&clientA, &clientB}

ctx, cancel := context.WithTimeout(ctx, 100*time.Millisecond)
defer cancel()

coords, err := geocode(ctx, a, clients)
//Metrics: Label: client A, count 1
//Metrics: Label: client B, count 1
```

Common pitfalls

MySQL ctx cancel

```
now := time.Now()
ctx, cancel := context.WithTimeout(ctx, 200*time.Millisecond)
defer cancel()
_, err := db.QueryContext(ctx, "SELECT SLEEP(10)")
if err != nil {
    fmt.Println(err)
}
fmt.Printf("Flight time %v\n", time.Since(now))
//Output: context deadline exceeded
//Output: Flight time 202.724294ms
```

Common pitfalls

MySQL ctx cancel

```
+-----+------+---------+------+------------+----------------------+
| Id  | db   | Command | Time | State      | Info                 |
+-----+------+---------+------+------------+----------------------+
| 348 | test | Query   | 0    | starting   | show full processlist |
| 359 | test | Query   | 4    | User sleep | SELECT SLEEP(10)     |
+-----+------+---------+------+------------+----------------------+
```

https://github.com/go-sql-driver/MySQL/issues/731

Summary

- Run async work with new context
- Beware functions without context parameter when continue on error
- Understand/question 3rd party libraries/drivers

Sources

- context package
- Using context cancellation in Go blog post
- justforfunc #9: The Context Package

Thank you