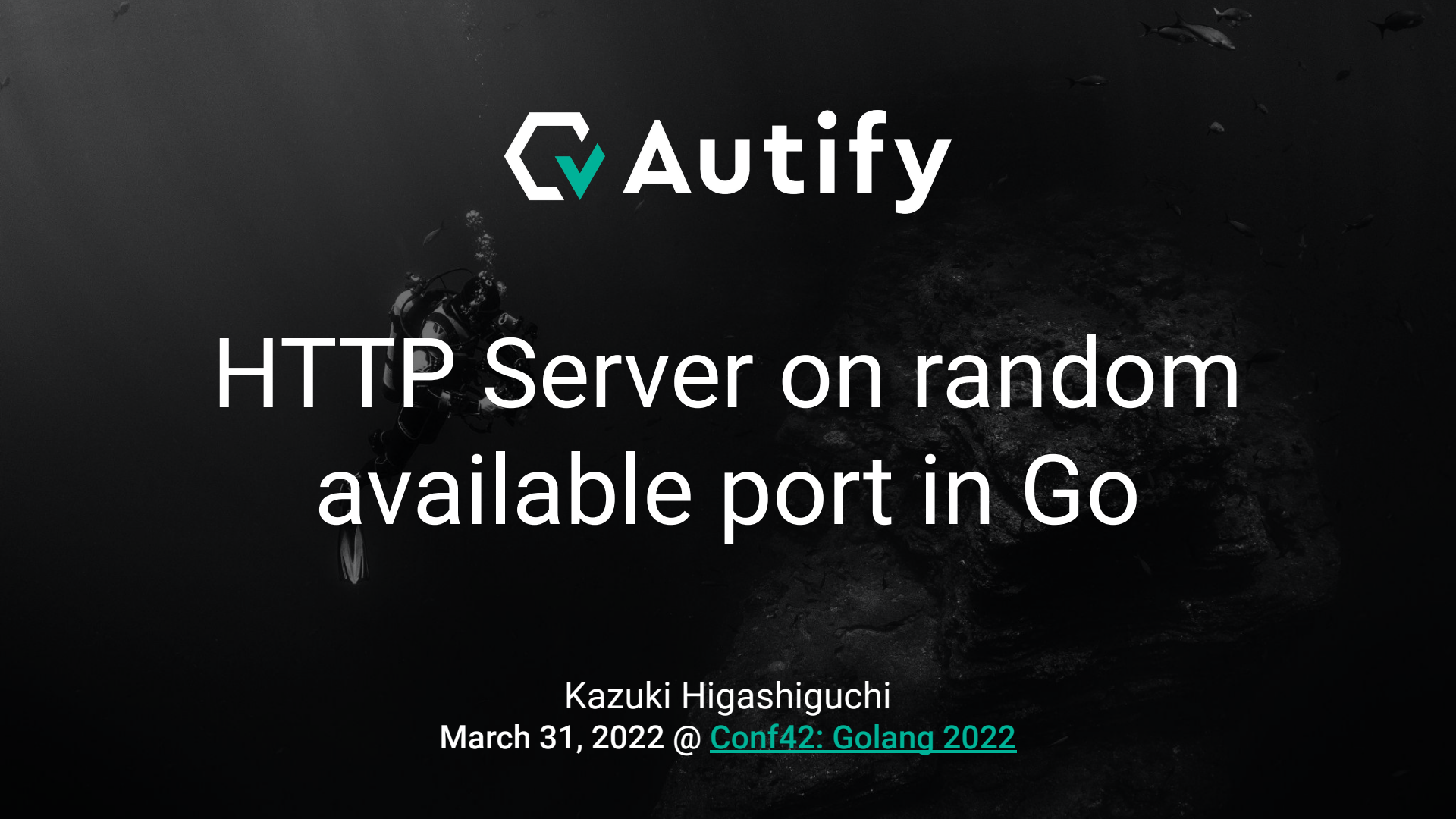


HTTP Server on random available port in Go

Kazuki Higashiguchi

March 31, 2022 @ Conf42: Golang 2022

About Me

Kazuki Higashiguchi

Backend Engineer at Autify.

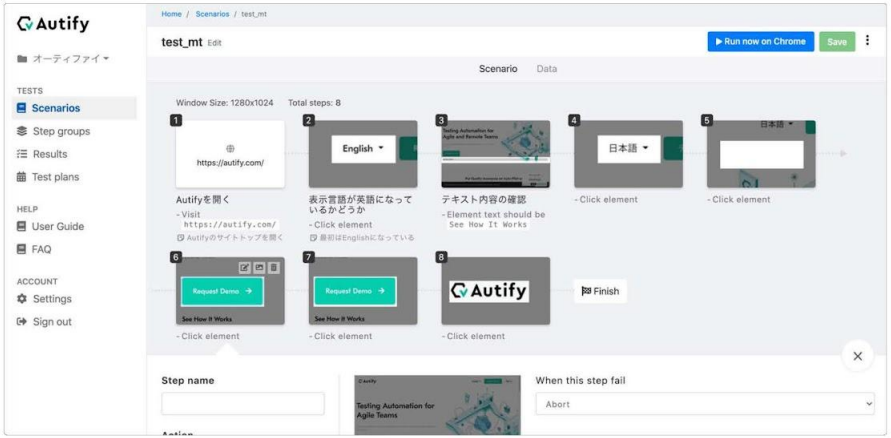
No-code AI-powered software testing automation platform

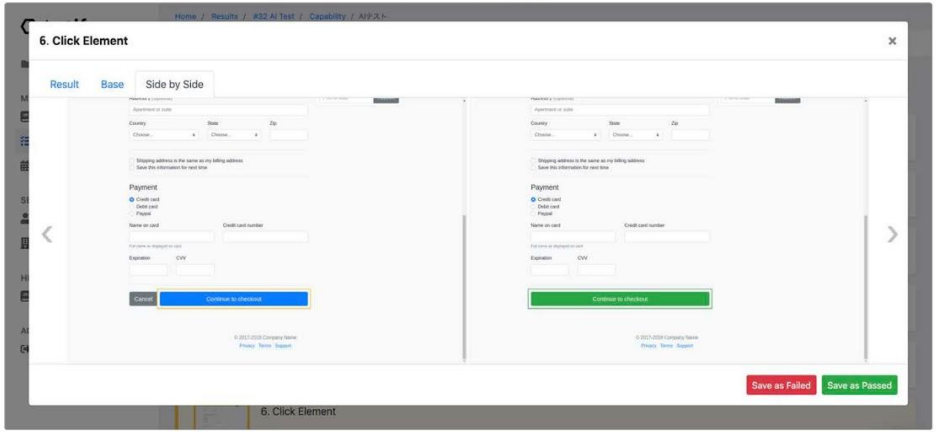
Follow @hgsgtk on Twitter

Autify Solution

No code unlocks automation at scale

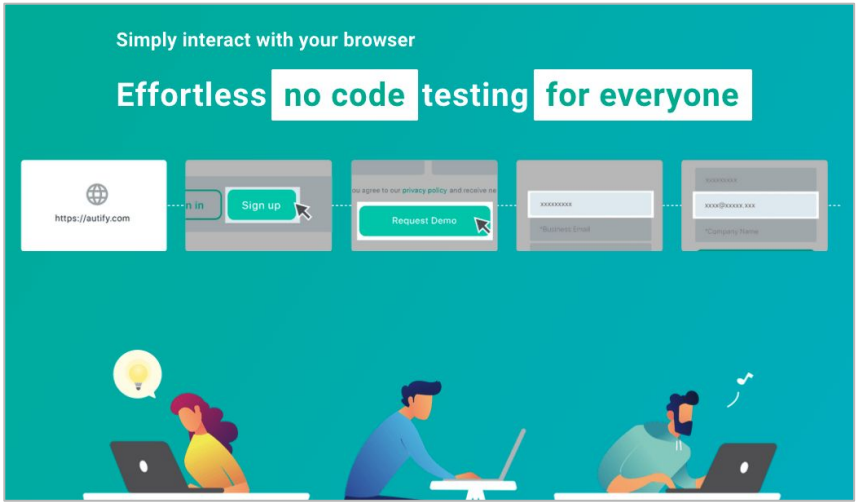
AI automatically maintenance test scripts



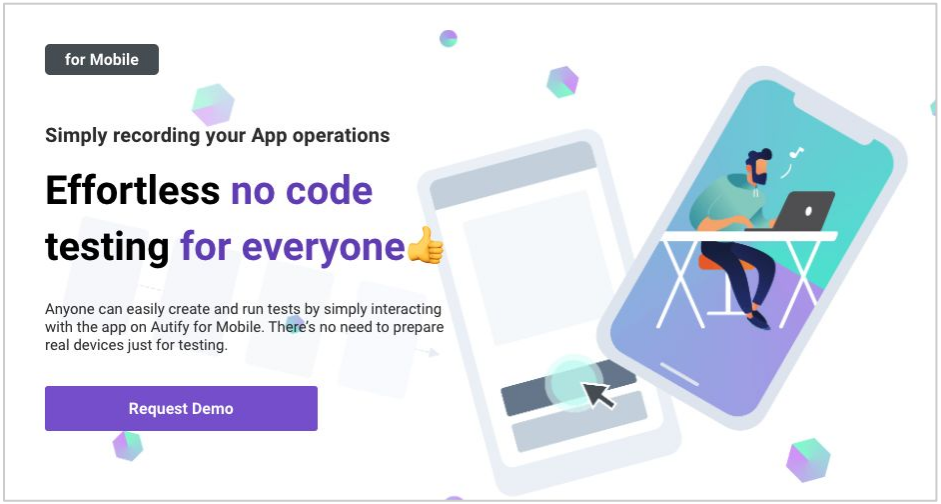


Autify for Web / for Mobile

Autify for Web



Autify for Mobile



We are taking demo requests https://autify.com/

HTTP server on a random available port

go run main.go

HTTP server

Bind & Listen

Port: ?????

No port specified

Available ports

60012 | 50123 | 45342 | ????? | ?????

Implementation using Go

```go
func main() {
    l, err := net.Listen("tcp", ":0")
    if err != nil {
        panic(err)
    }

    port := l.Addr().(*net.TCPAddr).Port
    log.Printf("using port: %d", port)
    if err := http.Serve(l, nil); err != nil
{       panic(err)
    }
}
```

using port: 57645
using port: 57473
using port: 57464
…

net.Listen

```
1 func Listen(network, address string) (Listener, error)
```

"If the port in the address parameter is empty or "0", as in "127.0.0.1:" or "[::1]:0", a port number is automatically chosen"

https://pkg.go.dev/net#Listen

net/http/httptest invokes net.Listen with “0”

```go
func TestServer(t *testing.T) {
    ts := httptest.NewServer(http.HandlerFunc(testHandler))
    t.Cleanup(func() { ts.Close() })

    got, err := http.Get(ts.URL)
```

net/http/httptest/server.go

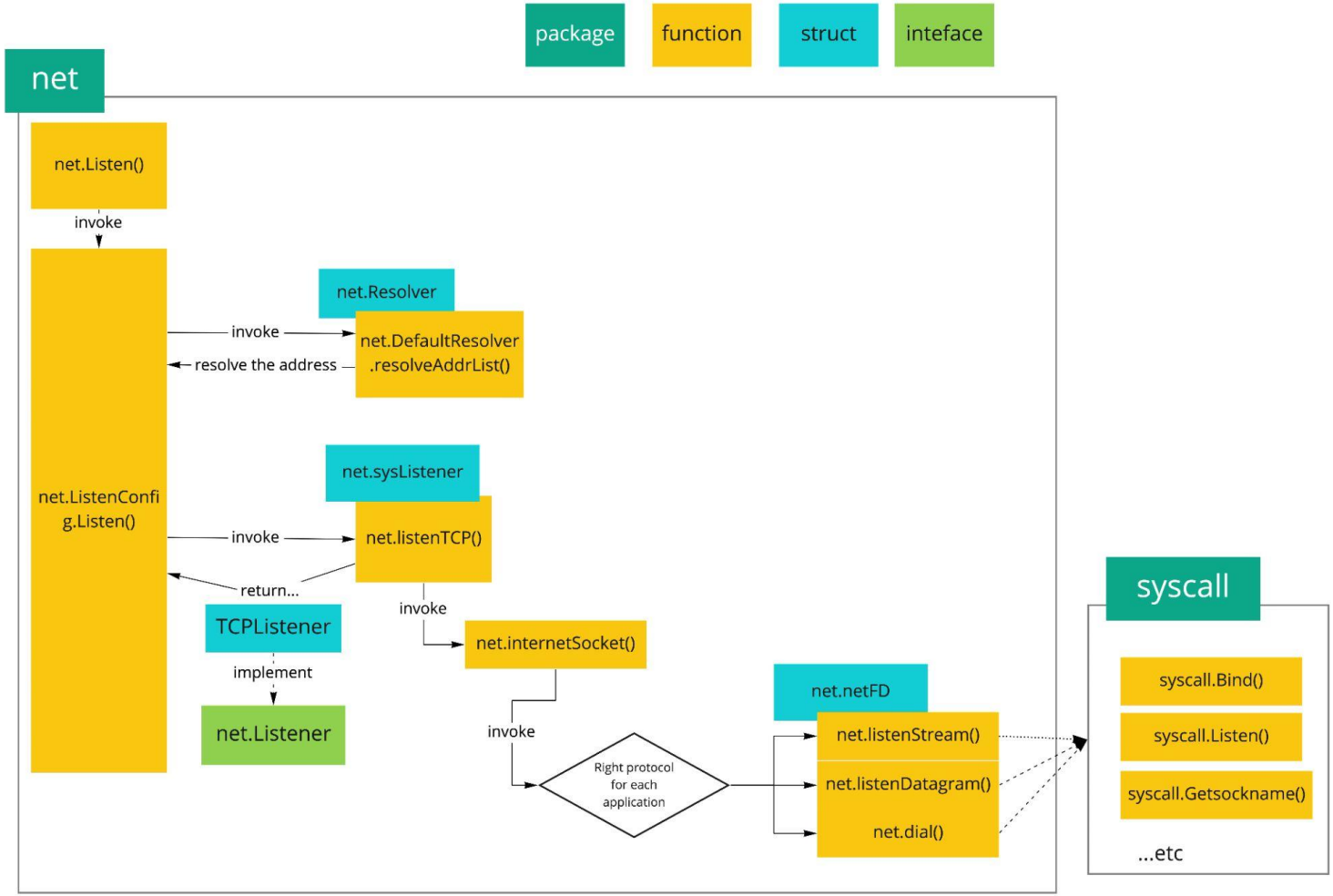
```go
func newLocalListener() net.Listener {
    // Omit
    l, err := net.Listen("tcp", "127.0.0.1:0")
    if err != nil {
        if l, err = net.Listen("tcp6", "[::1]:0"); err != nil {
            // Omit
        }
    }
    // Omit
}
```

https://cs.opensource.google/go/go/+/refs/tags/go1.18:src/net/http/httptest/server.go;drc=refs%2Ftags%2Fgo1.18;l=60

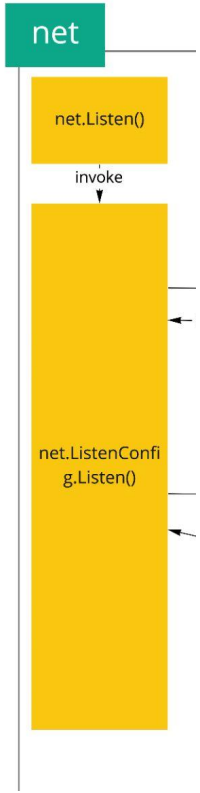

Dive into the
standard libraries
https://unsplash.com/photos/AN2SypyyOnA

Signature: “network”

```
1 // func Listen(network, address string) (Listener, error)
2
3 l, err := net.Listen("tcp", ":0") // IPv4 or IPv6
4 l, err := net.Listen("tcp4", ":0") // IPv4 only
5 l, err := net.Listen("tcp6", ":0") // IPv6 only
6 l, err := net.Listen("unix", ":0")
7 l, err := net.Listen("unixpacket", ":0")
```


package
function
struct
inteface
net
net.Listen()
invoke
net.ListenConfig.Listen()
net.Resolver
invoke
net.DefaultResolver
.resolveAddrList()
resolve the address
net.sysListener
invoke
net.listenTCP()
return...
TCPListener
implement
net.Listener
invoke
net.internetSocket()
invoke
Right protocol
for each
application
net.netFD
net.listenStream()
net.listenDatagram()
net.dial()
syscall
syscall.Bind()
syscall.Listen()
syscall.Getsockname()
...etc

1: net.Listen -> net.ListenConfig.Listen

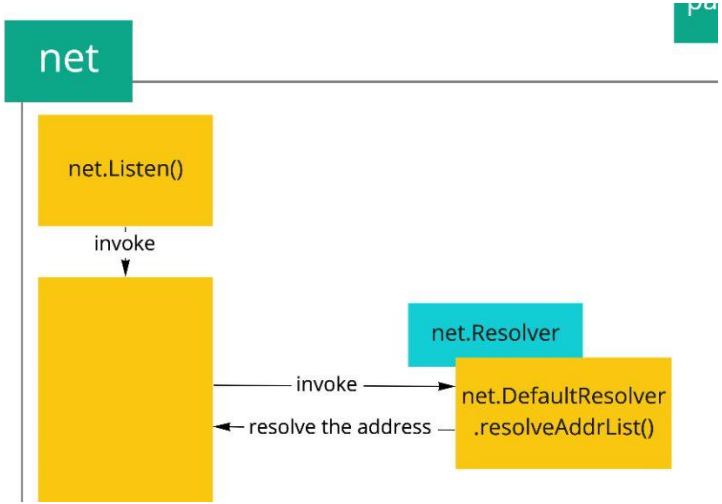


net/ipsock.go

```
708 // Listen uses context.Background internally; to specify the context, use
709 // ListenConfig.Listen.
710 func Listen(network, address string) (Listener, error) {
711   var lc ListenConfig
712   return lc.Listen(context.Background(), network, address)
713 }
```

https://cs.opensource.google/go/go/+/refs/tags/go1.18:src/net/dial.go;l=710

2. net.ListenConfig.Listen -> net.DefaultResolver



DefaultResolver resolves the network IP address

net/dial.go

```
625 func (lc *ListenConfig) Listen(ctx context.Context, network, address string) (Listener
626   addrs, err := DefaultResolver.resolveAddrList(ctx, "listen", network, address, nil)
627   if err != nil {
628     return nil, &OpError{Op: "listen", Net: network, Source: nil, Addr: nil, Err: err}
629   }
```

https://cs.opensource.google/go/go/+/refs/tags/go1.18:src/net/dial.go;l=625

3. net.DefaultResolver -> net.LookupPort

LookupPort looks up the port for the given network

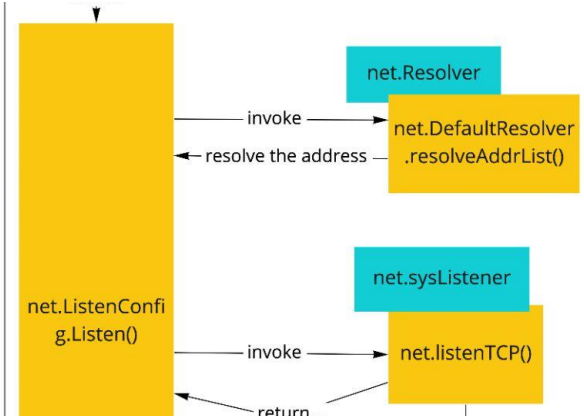
e.g. LookupPort("127.0.0.1", "0") -> returned port: 0

net/ipsock.go

```
248 func (r *Resolver) internetAddrList(ctx context.Context, net, addr string)
249   var (
250     err        error
251     host, port string
252     portnum    int
253   )
254   switch net {
255   case "tcp", "tcp4", "tcp6", "udp", "udp4", "udp6":
256     if addr != "" {
257       if host, port, err = SplitHostPort(addr); err != nil {
258         return nil, err
259       }
260       if portnum, err = r.LookupPort(ctx, net, port); err != nil {
261         return nil, err
262       }
263     }
```

https://cs.opensource.google/go/go/+/refs/tags/go1.18:src/net/ipsock.go;l=260;drc=refs%2Ftags%2Fgo1.18

4: net.ListenConfig.Listen -> sysListener.listenTCP

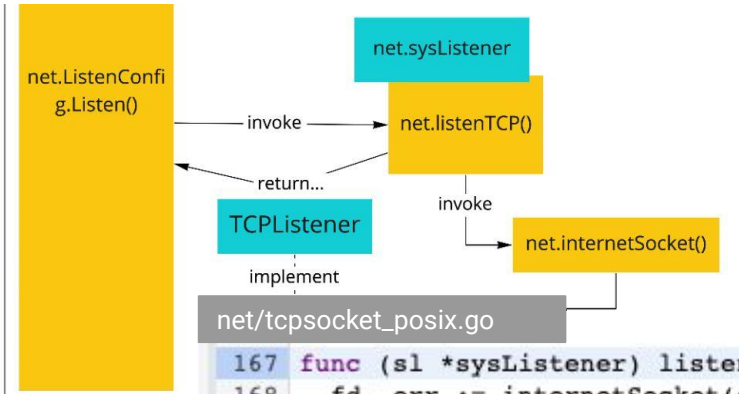


net/dial.go

```
629   }
630   sl := &sysListener{
631     ListenConfig: *lc,
632     network:      network,
633     address:      address,
634   }
635   var l Listener
636   la := addrs.first(isIPv4)
637   switch la := la.(type) {
638   case *TCPAddr:
639     l, err = sl.listenTCP(ctx, la)
640   case *UnixAddr:
641     l, err = sl.listenUnix(ctx, la)
642   default:
643     return nil, &OpError{Op: "listen", Net: sl.network, Source: nil, A
644   }
```

https://cs.opensource.google/go/go/+/refs/tags/go1.18:src/net/dial.go;l=639;drc=refs%2Ftags%2Fgo1.18

5: sysListener.listenTCP -> internalsocket



net/tcpsocket_posix.go

```
167 func (sl *sysListener) listenTCP(ctx context.Context, laddr *TCPAddr) (*TCPListener, error) {
168   fd, err := internetSocket(ctx, sl.network, laddr, nil, syscall.SOCK_STREAM, 0, "listen", sl.L
169   if err != nil {
170     return nil, err
171   }
172   return &TCPListener{fd: fd, lc: sl.ListenConfig}, nil
173 }
```

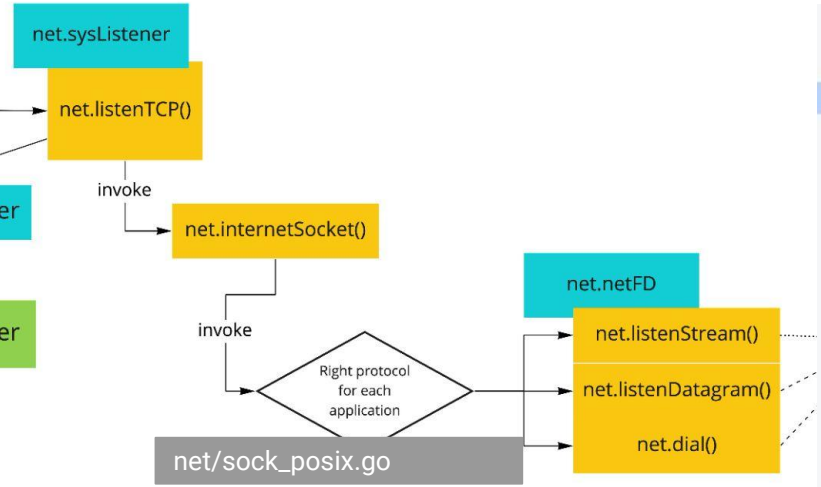
Return a network file descriptor

https://cs.opensource.google/go/go/+/refs/tags/go1.18:src/net/tcpsock_posix.go;l=167;drc=refs%2Ftags%2Fgo1.18

socket type: SOCK_STREAM

Network type	Socket type	Description
“tcp” (Transmission Control Protocol)	SOCK_STREAM	Stream-oriented Sequenced, reliable, two-way, connection-base byte streams.
“unix” (Unix domain sockets)		
“udp” (User Datagram Protocol)	SOCK_DGRAM	Datagram-oriented. Connectionless, unreliable messages
“unixgram”		
“unixpacket”	SOCK_SEQPACKET	Datagram-oriented. Sequenced, reliable, two-way, connection-base byte streams.

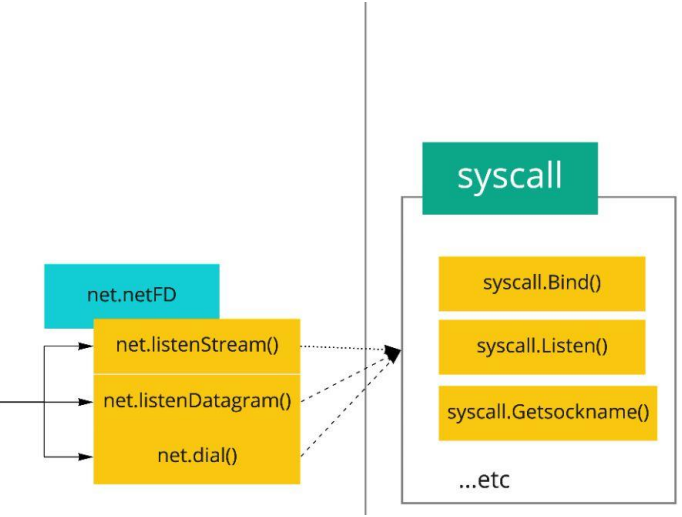
6: internalsocket -> net.listenStream




```
54  if laddr != nil && raddr == nil {
55    switch sotype {
56    case syscall.SOCK_STREAM, syscall.SOCK_SEQPACKET:
57      if err := fd.listenStream(laddr, listenerBacklog(), ctrlFn
58        fd.Close()
59        return nil, err
60      }
61      return fd, nil
62    case syscall.SOCK_DGRAM:
63      if err := fd.listenDatagram(laddr, ctrlFn); err != nil {
64        fd.Close()
65        return nil, err
66      }
67      return fd, nil
68    }
69  }
70  if err := fd.dial(ctx, laddr, raddr, ctrlFn); err != nil {
71    fd.Close()
72    return nil, err
73  }
74  return fd, nil
75 }
```

https://cs.opensource.google/go/go/+/refs/tags/go1.18:src/net/sock_posix.go;l=56;drc=refs%2Ftags%2Fgo1.18

7: execute three system calls

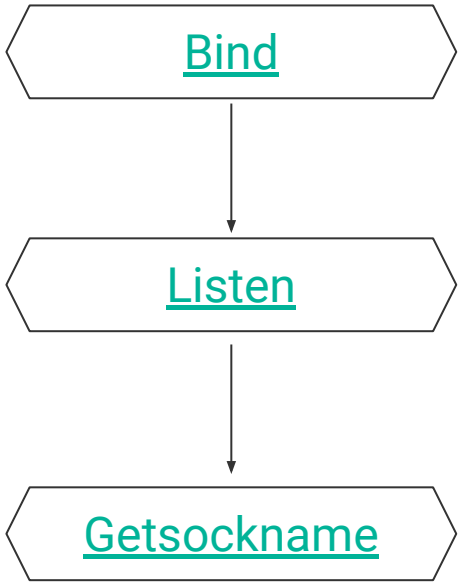


net/sock_posix.go

```
175 func (fd *netFD) listenStream(laddr sockaddr, ba
176   var err error
177   if err = setDefaultListenerSockopts(fd.pfd.Sysfd); err != nil {
178     return err
179   }
180   var lsa syscall.Sockaddr
181   if lsa, err = laddr.sockaddr(fd.family); err != nil {
182     return err
183   }
184   if ctrlFn != nil {
185     c, err := newRawConn(fd)
186     if err != nil {
187       return err
188     }
189     if err := ctrlFn(fd.ctrlNetwork(), laddr.String(), c); err != nil {
190       return err
191     }
192   }
193   if err = syscall.Bind(fd.pfd.Sysfd, lsa); err != nil {
194     return os.NewSyscallError("bind", err)
195   }
196   if err = listenFunc(fd.pfd.Sysfd, backlog); err != nil {
197     return os.NewSyscallError("listen", err)
198   }
199   if err = fd.init(); err != nil {
200     return err
201   }
202   lsa, _ = syscall.Getsockname(fd.pfd.Sysfd)
203   fd.setAddr(fd.addrFunc()(lsa), nil)
204   return nil
205 }
```

https://cs.opensource.google/go/go/+/refs/tags/go1.18:src/net/sock_posix.go;drc=refs%2Ftags%2Fgo1.18;l=175

bind, listen, getsockname

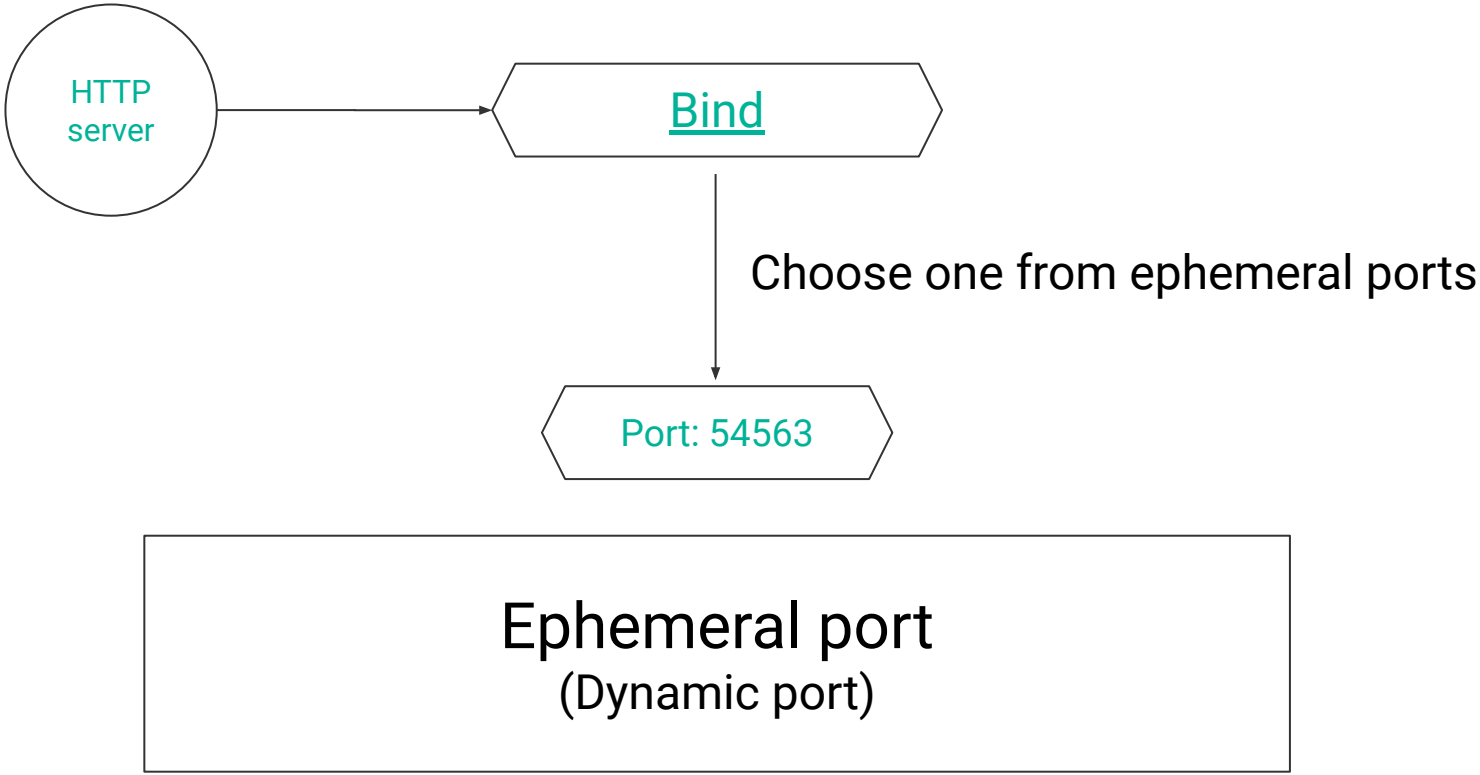


Assign the socket address to the socket referred to by the file descriptor

Mark the socket as a passive socket

Return the current socket address

Bind to an ephemeral port



Specification of bind

IANA defines the port range is 49152 - 65535

OS	When a port number is zero	Port range
Linux	Attempt to bind to an ephemeral port	(Basically) 49152 - 65535
Windows	Assign a unique port from the dynamic client port range	(On Windows Vista and later) 49152 - 65535 (Windows Server 2003 and earlier) 1025 - 5000

Key consideration

1. Confirm the “bind” specification
2. Check your infrastructure can use an ephemeral port
3. Check the range of ephemeral port

Thank you

See more detail in the article on dev,to.

Kazuki Higashiguchi

Posted on Jan 13 • Updated on Jan 14

Edit Manage Stats

Go: Deep dive into net package learning from TCP server

#go #webdev