

INDEX

A

Address Resolution Protocol (ARP), 26
Amazon Web Services (AWS) Lambda.
 See AWS (Amazon Web Services) Lambda
ARP (Address Resolution Protocol), 26
ASN (autonomous system number), 33–34
autonomous system number. *See* ASN
 (autonomous system number)
AWS (Amazon Web Services) Lambda,
 333–340
 command line interface, 333–335
 configuring, 333–335
 installing, 333
 compiling, packaging, and
 deploying, 339–340
 creating a Lambda function,
 336–338
 creating a role, 335–336
 testing a Lambda function, 340
Azure Functions. *See* Microsoft Azure
 Functions

B

bandwidth, 6–7
 vs. latency, 7
big package. *See* math/big package
bits, 9–10
Border Gateway Protocol (BGP), 34
broadcasting, 25–26
bufio package, 74, 76–78
 Scanner struct, 74, 76–78
bytes package, 79, 83–86, 89, 109–114,
 120, 122–123, 126–131, 133,
 146–147, 149, 151–153, 179–
 181, 188–189, 253, 255, 260,
 265, 299–302, 306, 320, 325

Buffer struct, 85–86, 89, 122, 126,
 128–129, 180–181, 299–301, 306
Equal(), 110, 112, 114, 147, 151,
 153, 255, 265
HasPrefix(), 325
NewBuffer(), 123, 127, 130
NewBufferString(), 189
NewReader(), 83, 127–128, 133
Repeat(), 147
Split(), 325

C

Caddy, 217–239
 automatic HTTPS, 237–238
 building from source code, 220
 configuration, 220–224
 adapters, 225–226
 administration endpoint, 220
 modifying, real time, 222–224
 traversal, 222
 using a configuration file, 224
 downloading, 219
 extending, 224–232
 configuration adapters,
 225–226
 injecting modules, 231–232
 middleware, 226–230
 Let’s Encrypt integration, 218
 reverse proxying, 219, 232–238
 diagram, 219
CDN (content delivery network), 7, 16
certificate pinning, 247, 252–255, 292
Classless Inter-Domain Routing
 (CIDR), 20–22
Cloud Functions. *See* Google Cloud
 Functions
content delivery network (CDN), 7, 16

- context package, 57–62, 65–66, 69, 107–111, 113, 144–146, 148–150, 152, 177–178, 183, 213, 249–250, 253, 260, 286–287, 290–291, 293
 - WithCancel(), 59, 66, 69, 109, 111, 113, 146, 150, 152, 178, 253, 260
 - Canceled error, 59–62
 - WithDeadline(), 58, 60–62
 - DeadlineExceeded error, 58, 61–62, 177
- crypto/ecdsa package, 256–257
 - GenerateKey(), 257
- crypto/elliptic package, 256–257
 - P256(), 257
- crypto/rand package, 75, 256, 258
 - Int(), 256
 - Read(), 75
 - Reader variable, 256
- crypto/sha512 package, 137
 - Sum512_256(), 137
- crypto/tls package, 245–251, 253–255, 260–264
- crypto/x509 package, 253–254, 256–260, 262–263, 292
 - Certificate struct, 256–257, 262
 - CreateCertificate(), 258
 - key usage, 257, 262
 - ExtKeyUsageClientAuth constant, 257, 262
 - ExtKeyUsageServerAuth constant, 257
 - MarshalPKCS8PrivateKey(), 259
 - NewCertPool(), 254, 260, 262, 292
 - VerifyOptions struct, 262–263
- crypto/x509/pkix package, 256–257
 - Name struct, 257

D

- DDOS (distributed denial-of-service)
 - attack, 34
- Defense Advanced Research Projects Agency (DARPA), 12
- delimited data, reading from a network. *See* bufio package, Scanner struct
- DHCP (Dynamic Host Configuration Protocol), 14

- distributed denial-of-service (DDOS)
 - attack, 34
- DNS (Domain Name System), 34–41
 - domain name resolution, 34–35
 - domain name resolver, 35
 - privacy and security considerations, 40–41
 - resource records, 35–40
 - Address (A), 36
 - Canonical Name (CNAME), 38
 - Mail Exchange (MX), 38–39
 - Name Server (NS), 37
 - Pointer (PTR), 39
 - Start of Authority (SOA), 37
 - Text (TXT), 39–40
- DNS over HTTPS (DoH), 41
- DNS over TLS (DoT), 41
- DNSSEC (Domain Name System Security Extensions), 41
- DoH (DNS over HTTPS), 41
- Domain Name System. *See* DNS (Domain Name System)
- Domain Name System Security Extensions (DNSSEC), 41
- DoT (DNS over TLS), 41
- dynamic buffers, reading into, 79–86
- Dynamic Host Configuration Protocol (DHCP), 14

E

- ecdsa package. *See* crypto/ecdsa package
- elliptic curve, 247
- elliptic package. *See* crypto/elliptic package
- encoding/binary package, 79–85, 120, 122–123, 126–130
 - BigEndian constant, 80–83, 85, 122–123, 126–130
 - Read(), 80–83, 123, 127–130
 - Write(), 80–81, 85, 122, 126, 128–129
- encoding/gob package, 278–279
 - NewDecoder(), 279
 - NewEncoder(), 279
- encoding/json package, 179–180, 225–226, 228, 277, 342–343
 - Marshal(), 226, 343
 - NewDecoder(), 179, 277, 343
 - NewEncoder(), 180, 277
 - Unmarshal(), 228

encoding/pem package, 256, 258–259, 270
 Block struct, 258
 Encode(), 258
external routing protocol, 34

F

filepath package. *See* path/filepath package
fixed buffers, reading into, 74–76
flag package 96–97, 135, 137, 157–158, 211, 232–233, 256, 271–272, 276, 288–290, 292–293, 317, 320–321, 323
 Arg(), 97, 276, 293
 Args(), 137, 158, 276, 293
 CommandLine variable, 157, 272, 290
 Output(), 157, 272, 290
 Duration(), 96
 Int(), 96
 NArg(), 97
 Parse(), 97, 135, 137, 158, 211, 233, 256, 276, 288, 292, 323
 PrintDefaults(), 96, 137, 157, 272, 290
 String(), 135, 211, 233, 256, 317, 321
 StringVar(), 272, 288, 290
 Usage(), 97
fragmentation. *See* UDP (User Datagram Protocol), fragmentation

G

global routing prefix (GRP), 27–28
Gob
 decoding. *See* encoding/gob package
 encoding. *See* encoding/gob package
 serializing objects, with. *See* serializing objects, Gob
Google Cloud Functions, 341–346
 defining a Cloud Function, 342–344
 deploying a Cloud Function, 344–345
 enable billing and Cloud Functions, 342
 installing the software development kit, 341–342
 testing a Cloud Function, 345–346

GRP (global routing prefix), 27–28
gRPC, 284–294
 client, 289–294
 connecting services, 284–286
 server, 286–289

H

handlers, 193–202
 advancing connection deadline, 68–70
 dependency injection, 200–202
 implementing the http.Handler interface, 198–200
 testing, 195–196
 writing the response, 196–198
heartbeat, 64–70
hextets, 26–28
html/template package, 194
HTTP (HyperText Transfer Protocol), 10, 23, 41, 165–215
 client-side, 165–184
 default client, 174–175
 requests, from client, 167–170
 responses, from server, 170–172
 request-response cycle, 172–173
 server-side, 187–215
 anatomy of a Go HTTP server, 188
 Caddy. *See* Caddy
http package. *See* net/http package
httptest package. *See* net/http/httptest package
HTTP/1. *See* HyperText Transfer Protocol (HTTP)
HTTP/2 Server Pushes, 209–214
HyperText Transfer Protocol (HTTP). *See* HTTP (HyperText Transfer Protocol)

I

IANA (Internet Assigned Numbers Authority), 23, 28, 35
ICMP (Internet Control Message Protocol), 31–32, 96, 98
 destination unreachable, 31
 echo, 32
 echo reply, 32

- ICMP (*continued*)
 - fragmentation, checking, 115–116
 - redirect, 32
 - time exceeded, 32
 - IETF (Internet Engineering Task Force), 26
 - instrumentation, 316–326
 - counters, 317–318
 - gauges, 318–319
 - histograms and summaries, 319–320
 - HTTP server, instrumentation, 320–326
 - Internet Assigned Numbers Authority (IANA), 23, 28, 35
 - Internet Control Message Protocol. *See* ICMP (Internet Control Message Protocol)
 - Internet Engineering Task Force (IETF), 26
 - Internet Protocol (IP), 12, 18
 - internet service provider (ISP), 7
 - inter-process communication (IPC), 141
 - io package, 54–55, 63–66, 70, 74–84, 87–96, 126, 176, 179–180, 182, 189, 194, 196, 198, 207, 255, 273, 277–279, 283, 297–298, 301–302, 306–307, 314–315, 324
 - Copy(), 87–89, 92, 126, 176, 180, 182, 194, 198, 207, 314, 324
 - CopyN(), 89, 126
 - Eof error, 54–55, 63–64, 70, 75, 78, 88, 90–91, 94, 126, 255
 - MultiWriter(), 87, 93–96, 297–298
 - TeeReader(), 87, 93–96
 - ioutil package. *See* io/ioutil package
 - io/ioutil package, 135, 137, 146, 149, 152, 175, 179, 183, 188, 190, 194, 198–199, 203–204, 207, 209, 253–254, 260, 283, 289, 292, 302, 309, 312, 314–315, 321, 324–325, 330–331
 - Discard variable, 175, 179, 194, 198, 207, 314, 324
 - ReadAll(), 183, 190, 194, 199, 204, 209, 283, 325, 331
 - ReadFile(), 135, 137, 254, 260, 292
 - TempDir(), 146, 149, 152, 309, 315
 - IP (Internet Protocol), 12, 18
 - IPC (inter-process communication), 141
 - IPsec, 31
 - IPv4 addressing, 18–26
 - host ID, 19–20
 - localhost, 23
 - network ID, 19–22
 - network prefix, 20–22
 - subnets, 20
 - IPv6 addressing, 28–33
 - address categories, 28
 - anycast address, 29–30
 - multicast address, 29
 - unicast address, 28
 - advantages over IPv4, 30
 - interface ID, 27–28, 30–31
 - simplifying, 27
 - subnet ID, 27–28
 - subnets, 28
- J**
- JSON
 - encoding and decoding, 179–180, 225–226, 228, 277, 342–343
 - serializing objects with, 276–278
- K**
- keepalive messages, 99, 175, 192
- L**
- Lambda. *See* AWS (Amazon Web Services) Lambda
 - latency, 7
 - reducing latency, 7
 - vs. bandwidth, 7
 - Let’s Encrypt, integration with Caddy, 218
 - linger, 99–100
 - log package, 93–94, 131, 135, 155, 157, 201, 211, 232, 256, 271, 288–289, 297–302, 321, 242, 249
 - Ldate constant, 297
 - levels, 300–301
 - Lmsgprefix constant, 299
 - Lshortfile constant, 201, 297, 299–300
 - LstdFlags constant, 297

Ltime constant, 297
New(), 94, 201, 297, 299–300
lumberjack. *See* zap logger, log rotation

M

MAC (media access control) address,
 10, 24
math/big package, 256
 Int type, 256
 NewInt(), 256
maximum transmission unit (MTU),
 115–116
mDNS (Multicast DNS), 40
media access control (MAC) address,
 10, 24
metrics. *See* instrumentation
Microsoft Azure Functions, 346–353
 command line interface, 346–347
 configuring, 347
 installing, 346–347
 custom handler, 348–353
 creating, 348–349
 defining: 349–350
 deploying, 351–352
 testing, locally, 350–351
 testing, on Azure, 353
 installing core tools, 347
middleware, 202–206
 protecting sensitive files, 204–206
 time out slow clients, 203–204
mime/multipart package, 179, 181
 NewWriter(), 181
monitoring network traffic, 89–92
MTU (maximum transmission unit),
 115–116
Multicast DNS (mDNS), 40
multicasting, 23, 106
multiplexers, 207–209

N

NAT (network address translation), 24
net package, 51–70, 73–75, 77, 84–103,
 107–117, 131–134, 143–153,
 155, 156, 158–159, 189, 192,
 221, 248, 250–252, 257, 262,
 270, 288, 313, 322
 binding, 51–52

Conn interface, 52–54, 56, 73–74,
 77, 87, 89–92, 98–99, 102, 107,
 113–115, 144, 146, 156, 159,
 252, 270, 322
 SetDeadline(), 62–63, 69, 74,
 114, 252
 SetReadDeadline(), 62, 74, 133
 SetWriteDeadline(), 62, 74
Dial(), 54, 63, 69, 75, 77, 85, 88,
 91–92, 95, 113–115, 134,
 147–148, 152–153
DialContext(), 58–61
Dialer struct, 56–60, 248
DialTimeout(), 56–58, 97
Error interface, 55–58, 63, 86–87,
 97, 133
Listen(), 51–54, 60, 62, 68, 75, 77,
 84, 90–91, 94, 145, 189, 192,
 250, 288, 322
Listener interface, 51–52, 189,
 250–251
ListenPacket(), 107–111, 113, 117,
 131, 143, 146, 148–150
ListenUnix(), 143, 146, 149, 158–159
LookupAddr(), 262
OpError struct, 55
PacketConn interface, 107–110,
 113–115, 117, 131–132, 149
ParseIP(), 257
ResolveTCPAddr(), 98–99
SplitHostPort(), 313
TCPConn struct, 89, 98–101
UnixConn struct, 155–156, 159
net/http package, 168, 171, 173–174,
 Client struct, 190, 246, 324
 Get(), 174, 176–177, 180, 185,
 314, 325
 Head(), 174, 176, 185
 Post(), 176, 181, 185
 Error(), 180, 194–195, 197–199,
 202, 205, 229
 FileServer(), 204–206, 212, 236
 FileSystem interface, 204, 206
 Handle(), 200–201
 HandlerFunc(), 177, 180, 193–195,
 199–203, 205, 207, 212, 233,
 245, 313–314, 323

- net/http package (*continued*)
 - Handler interface, 193–195, 198–205, 207, 215, 226, 227, 309, 313–315, 321
 - NewRequest(), 190
 - NewRequestWithContext(), 177–178, 183
 - Pusher interface, 212–213
 - Request struct, 176, 179, 193–196, 198–203, 205, 207–208, 212, 227, 229, 233, 245, 313–314, 321
 - persistent TCP connections, disable, 178–179
 - time-out, cancellation, 176–178
 - Response struct, 174, 177, 180–181, 183, 190, 204, 209, 246–247, 314, 324–325
 - body, closing, 175–176
 - ResponseWriter interface, 176, 179, 193–196, 198–203, 205, 207, 212, 227, 229, 233, 245, 312–314, 321
 - WriteHeader(), 180, 196, 203, 207, 245
 - ServeMux struct, 207–208, 212, 233, 323
 - Server struct, 189, 191–192, 195, 213, 233, 322
 - connection state, 322
 - time-out settings, 191–192
 - TLS support, adding, 192–193
 - StripPrefix(), 205–206, 212
 - TimeoutHandler(), 189, 200, 203–204
 - Transport struct, 246–247, 324, 326
 - default transport, 324
- net/http/httptest package, 177, 180, 196–197, 203, 205, 209, 245–246, 314
 - NewRecorder(), 196–197, 203, 205, 209
 - NewRequest(), 196–197, 203, 205, 209
 - NewServer(), 177, 180, 314
 - NewTLSServer(), 245–246
 - ResponseRecorder struct, 196
- network address translation (NAT), 24
- network errors. *See* net package, Error interface

- network topology, 3–6
 - bus, 4
 - daisy chain, 4
 - hybrid, 6
 - mesh, 5–6
 - point-to-point, 4
 - ring, 5
 - star, 5
- nibble, 26

O

- octets, 18
- Open Systems Interconnection (OSI)
 - reference model, 7–12
 - encapsulation, 10
 - datagrams, 12
 - frame, 12
 - frame check sequence (FCS), 12
 - horizontal communication, 10–11
 - message body, 10
 - packet, 12
 - payload, 10
 - segments, 12
 - service data unit (SDU), 10
 - layers, 8–9
 - application (layer 7), 8
 - data link (layer 2), 9
 - network (layer 3), 9
 - physical (layer 1), 9
 - presentation (layer 6), 9
 - session (layer 5), 9
 - transport (layer 4), 9
- os package, 93, 96–97, 137, 143, 146–150, 152, 157–158, 179, 182, 211, 213, 232–233, 256, 258, 271–273, 289–290, 299, 302, 304, 310–312, 315, 349
- Args slice, 96, 137, 157, 272, 290
- Chmod(), 143, 147, 150, 152
- Chown(), 143
- Create(), 258, 273, 311
- Exit(), 97, 304, 315
- Getpid(), 146, 150, 152
- IsNotExist(), 272
- LookupEnv(), 349
- Open(), 182, 272
- OpenFile(), 258

- Remove(), 148, 311
- RemoveAll(), 146, 149, 152, 310, 315
- Signal type, 158, 213, 233
- Stat(), 272
- TempDir(), 158
- OSI. *See* Open Systems Interconnection reference model (OSI)

P

- path package, 204, 211, 302
 - Clean(), 205
- path/filepath package, 146, 149–150, 152, 157–158, 179, 182, 212, 271–272, 289–290, 310, 315
 - Base(), 157, 182, 272, 290
 - Join(), 146, 150, 152, 158, 212, 310, 315
- pem package. *See* encoding/pem package
- ping TCP ports, 96–98
- pkix package. *See* crypto/x509/pkix package
- ports, 23
- posting data over HTTP, 179–184
 - multipart form with file attachments, 181–184
- protocol buffers, 280–284
- proxying network data, 87–93

R

- rand package. *See* crypto/rand package
- receive buffer, connection set, 100–101
- reflect package, 78, 85
 - DeepEqual(), 78, 85
- Request for Comments (RFC), 15
- routing, 17, 32–33

S

- scanning delimited data, 76–78
- serializing objects, 270–284
 - Gob, 278–280
 - JSON, 276–278
 - Protocol Buffers, 280–284
 - transmitting. *See* gRPC
- Simple Mail Transfer Protocol (SMTP), 13
- SLAAC (stateless address autoconfiguration), 30–31
- socket address, 23–25, 106–107, 142–144, 221–222, 238

- sort package, 198–199
 - Strings(), 199
- stateless address autoconfiguration (SLAAC), 30–31
- strconv package, 271, 275, 289, 291–292
 - Atoi(), 275, 291–292
- strings package, 120, 124, 130, 198–199, 205, 228–229, 245–246, 253, 256–257, 260, 262–263, 271, 274, 276, 289, 291, 293
 - Contains(), 246, 253, 263
 - HasPrefix(), 205, 229
 - Join(), 199, 276, 293
 - Split(), 205, 229, 257, 262, 274, 291
 - ToLower(), 124, 276, 293
 - TrimRight(), 123–124, 130
 - TrimSpace(), 274, 291
- structured logging. *See* zap logger
- subdomain, 35

T

- TCP (Transmission Control Protocol), 8, 11–14, 45–102
 - flags, 47–50
 - acknowledgment (ACK), 47–50
 - finish (FIN), 50
 - reset (RST), 51
 - selective acknowledgment (SACK), 48
 - synchronize (SYN), 47–48
 - handshake, 47
 - maximum segment lifetime, 50
 - receive buffer, 48
 - reliability, 46
 - sequence number, 47–48
 - sliding window, 49
 - termination, 50
 - transport layer, 14
 - window size, 48–49
- TCP/IP model, 12–15
 - end-to-end principle, 12
 - layers, 13–15
 - application, 13
 - internet, 14
 - link, 15
 - transport, 14
- temporary errors, 55, 86, 97–98, 102

- TFTP (Trivial File Transfer Protocol), 119–139
 - downloading files over UDP, 135–138
 - server implementation, 131–135
 - handling read requests, 132–134
 - starting, with payload, 135
 - types, 120–130
 - acknowledgment, 128–129
 - data packet, 124–127
 - error packet, 129–130
 - read request (RRQ), 121–124
- time package, 56, 58–60, 62–63, 65–70, 87, 96–97, 113–114, 131, 133, 174, 176, 179, 181, 188, 202–203, 211, 232, 245, 249, 252–253, 255–257, 302–303, 308, 321, 323, 349
 - NewTimer(), 65
 - Parse(), 174
 - Round(), 67, 174
 - Since(), 67, 69–70, 97, 321
 - Sleep(), 58–59, 87, 97, 203, 255, 308
 - Truncate(), 69–70
- time-out errors, 55, 57–58, 63–64, 133
- TLS (Transport Security Layer), 41, 192–193, 212–214, 241–266, 288–289, 292–293
 - certificate authorities, 243–244
 - client-side, 245–248
 - recommended
 - configuration, 247
 - certificate pinning, 252–255
 - forward secrecy, 243
 - how to compromise TLS, 244
 - leaf certificate, 263
 - mutual TLS authentication, 255–265
 - generating certificates for authentication, 256–259
 - implementation, 259–265
 - server-side, 249–252
 - recommended
 - configuration, 251
- tls package. *See* crypto/tls package
- top-level domain, 35
- topology. *See* network topology

- Transmission Control Protocol. *See* TCP (Transmission Control Protocol)
- Transport Security Layer (TLS), 41, 192–193, 212–214, 241–266, 288–289, 292–293
- Trivial File Transfer Protocol. *See* TFTP (Trivial File Transfer Protocol)
- type-length-value encoding, 79–86

U

- UDP (User Datagram Protocol), 14, 105–117, 119–120, 131–136, 139, 144, 148, 150–151, 221
 - fragmentation, 115–117
 - maximum transmission unit. *See* MTU (maximum transmission unit)
 - packet structure, 106
 - sending and receiving data, 107–115
 - listening for incoming data, 110–112
 - transport layer, 14
- unicasting, 25
- uniform resource locator (URL), 165–167
- Unix domain sockets, 141–161
 - authentication, 154–160
 - peer credentials, 154–156
 - testing with Netcat, 159–160
 - binding, 143
 - changing ownership and permissions, 143–144
 - types, 144–154
 - unix streaming socket, 144–148
 - unixgram datagram socket, 148–151
 - unixpacket sequence packet socket, 151–154
- unix package, 154–156
 - GetsockoptUcred(), 155–156
- URL (uniform resource locator), 165–167
- User Datagram Protocol. *See* UDP (User Datagram Protocol)

W

write buffer, connection set, 100–101

X

x509 package. *See* crypto/x509 package

Z

zap logger, 301–316

encoder configuration, 302–303

encoder usage, 305

logger options, 303–304

log rotation, 315–316

multiple outputs and encodings,
306–307

on-demand logging, 309–312

sampling, 307–309

wide event logging, 312–315

zero window errors, 101–102