Server + Middleware — Notes & Best Practices

Purpose

- Quick reference for setting up an Express server with common middleware and a graceful shutdown pattern.

Install (one-liner)

- npm install helmet winston morgan cors
- Why: helmet for security headers, winston for structured logging, morgan for HTTP request logs, cors for cross-origin requests.

Middleware — explanation and why to use (point-wise)

- **helmet()**
  - **What:** Adds many HTTP headers to improve security (HSTS, frameguard, etc.)
  - **Why:** Reduces common web vulnerabilities with minimal effort.
- **cors()**
  - **What:** Enables controlled Cross-Origin Resource Sharing.
  - **Why:** Allows browsers to call your API from other origins safely; configure origins and methods as needed.
- **express.json()**
  - **What:** Parses incoming JSON request bodies and populates req.body.
  - **Why:** Required to handle JSON payloads without manual parsing.
- **morgan()**
  - **What:** HTTP request logger middleware.
  - **Why:** Lightweight request-level logging; combine with winston for production logs.
- **winston**
  - **What:** Flexible logging library (file, console, transports).
  - **Why:** Structured, level-based logging suitable for production and aggregation systems.

Health endpoint

- **Purpose:** Quickly check whether the server is responsive.
- **Example:** GET /health returning 200 and basic info (uptime, status).
- **Use in** orchestration (Kubernetes, load balancers) for liveness/readiness checks.

Process uptime

- **process.uptime()**
  - Returns the number of seconds the current Node.js process has been running.
  - Useful in /health responses and for debugging restarts.

Graceful shutdown — why it matters (point-wise)

- Avoids killing active requests in-flight.

- Prevents data corruption or partial writes.
- Ensures connections (DB, queues) can close cleanly.
- Important for microservices and orchestration systems (Kubernetes expects graceful termination).

Common questions (Q&A)

- **Q: Should I use http.createServer(app) or app.listen()?**
  - A: app.listen(...) internally creates and returns an http.Server. Use app.listen for brevity and it still returns the server object for graceful shutdown.
- **Q: How long should I wait before force-killing?**
  - A: Choose a timeout slightly longer than your typical longest request (e.g., 30s). Coordinate with load balancer/k8s terminationGracePeriodSeconds.
- **Q: How to handle new connections during shutdown?**
  - A: Stop accepting new requests (server.close()) and let existing requests finish. Optionally respond with 503 for readiness probes beforehand.

Recommended minimal server + graceful shutdown snippet

- Use app.listen(...) and keep the same graceful shutdown logic; this is concise and returns an http.Server.

```
// Example server snippet (illustrative)
const { createApp } = require('./app');
const config = { port: 3000 };

const app = createApp();
const server = app.listen(config.port, () => {
  console.log(`Server is running on port ${config.port}`);
});

// Graceful shutdown
const SHUTDOWN_TIMEOUT = 30000; // ms
let isShuttingDown = false;

function shutdown(signal) {
  if (isShuttingDown) return;
  isShuttingDown = true;
  console.log(`Received ${signal} — shutting down gracefully`);
  // Stop accepting new connections
  server.close(err => {
    if (err) {
      console.error('Error during server close', err);
      process.exit(1);
    }
    // Close DB/connections here, then exit
    console.log('Closed remaining connections, exiting');
    process.exit(0);
  });
  // Force kill after timeout
  setTimeout(() => {
    console.warn('Forcing shutdown after timeout');
    process.exit(1);
```

```
  }, SHUTDOWN_TIMEOUT);
}

process.on('SIGTERM', () => shutdown('SIGTERM'));
process.on('SIGINT', () => shutdown('SIGINT'));
```

Notes & tips

- Combine morgan with winston by streaming morgan output into a winston transport for unified logs.
- Keep health endpoint simple and fast — avoid heavy DB calls in liveness checks.
- Document shutdown behavior for your deployment environment (Kubernetes, systemd, etc.).
- Example command to run: node src/server.js or npm start (depending on project scripts).