
```
perf_hooks
```

`perf_hooks`

`--prof` `--prof-process`

`clinic flame`

`clinic doctor`

`node:perf_hooks`

`--prof` `--prof-process`

`clinic doctor`

`clinic flame`

```
                                  autocannon

autocannon -c 50 -d 20 http://localhost:3000/api/users

     -c                        -d
```

JSON.parse JSON.stringify
stream-json

fs.readFileSync crypto.pbkdf2Sync zlib.gzipSync *Sync

safe-
regex vuln-regex-detector

process.nextTick Promise.resolve()
setImmediate

EXPLAIN ANALYZE

NODE_ENV=production

--inspect --prof

bench(name, fn, iterations = 10_000) fn iterations performance.now() += Array.join

EventLoopMonitor start() stop() warnThreshold report()

profileMiddleware(options)
performance.mark measure
PerformanceObserver
GET /metrics

`WeakRef`

`FinalizationRegistry`

`dispose()` `close()`

`using`

`try/finally` `[Symbol.dispose]()`

`WeakMap`

`Map`				
`WeakMap`				
`Set`				
`WeakSet`				

```
              --expose-gc         global.gc()
```

```
node --expose-gc test.js

let obj = { big: new Array(1_000_000) };
const ref = new WeakRef(obj);
obj = null; // remove the strong reference
global.gc(); // force collection
console.log(ref.deref()); // undefined — collected
```

```
          --expose-gc
```

max_memory_restart

memoryUsage()
heapUsed

SoftCache
get(key) set(key, value)
has(key) size --expose-gc
size global.gc()

processAll(items, fn, { concurrency, onProgress })
concurrency onProgress({ done, total,
heapUsedMB })

`ws`

```
ws.close(code, reason)

ws.terminate()
```

`ws.send(string)` `ws.send(Buffer)` `isBinary` `message`

```
wss.clients
```

1000 1001 1006
1011
4000–4999 4001 4002 4003

`send()`

`ws.bufferedAmount`

[10:45:32] your
message
{ type: "stats", count: N }

{ type: "who", room }
username
{ type: "leave", username }

{ type: "subscribe", sensor: "temp" }
{ type: "unsubscribe", sensor: "temp" }
ws.bufferedAmount

```

```

`Object.prototype`

`Object.prototype`

`if (obj.admin)`

`$.extend`

`npm audit`

`for (key in source)`

`target[key]`

`169.254.169.254`

	`lodash` `lodahs`	`npm audit`
		`@yourcompany/pkg` `registry` `.npmrc`
	`postinstall` `npm install`	`ignore-scripts=true` `.npmrc`
		`package.json` `npm audit signatures`

```
node --experimental-permission --allow-fs-read=/uploads server.js

/etc/passwd
--allow-fs-write --allow-net --allow-child-process --allow-worker
```

`exec()`

`execFile()` `spawn()`

`process.env`

`createSafeLogger`

`helmet` `app.use(require('helmet')())` `X-Content-Type-Options` `X-Frame-Options` `Strict-Transport-Security` `Content-Security-Policy` `X-Powered-By: Express`

`express-rate-limit` `rateLimit({ windowMs: 15 * 60 * 1000, max: 100 })` `slow-down`

`deepMerge(target, ...sources)` `__proto__` `constructor` `prototype` `__proto__` `constructor.prototype` `Object.prototype`

`testRegexSafety(pattern, inputs, timeoutMs)` `{ safe: boolean, results: [{ input, matched, durationMs, timedOut }] }` `safe` `/^a+$/` `/^(a+)+$/` `'a'.repeat(30) + 'b'` `safe: false`

`ssrfGuard(allowedDomains)` `url`

`req.safeUrl`

```
EX 300              EX 5
```

```
redis.del(`user:${id}`)
```

```
user:v3:42
```

```
FLUSHDB
```

```
tag:product:99
```

```
SADD tag:product:99 "key1" "key2"
```

```
SMEMBERS
```

```
DEL
```

`user:42:orders` `orders` `products:en:42` `` `search:${userInput}` ``

```
    entity:id:field       user:42 user:42:orders product:99:stock
                                                                                redis-cli --scan
--pattern "user:*"
                                                myapp:user:42
```

```
            LRUCache
       get(key)                                  set(key, value, ttlMs)
                                  delete(key)                      size
```

```
      RedisCache                                getOrSet(key, fetcher, ttlSec)
                                  stats()         { hits, misses, hitRate }
invalidate(...keys)                                                          ioredis-mock
```

```
     stampedeSafeGet(redis, key, fetcher, ttlSec)
 SET NX EX
                                       fetcher
                ioredis-mock                              jest.useFakeTimers()
```

	await	

Worker

@bull-board/express

/admin/queues

SimpleQueue add(name, data, { priority, delayMs }) process(name, concurrency, fn) maxAttempts 'completed' 'failed'

enqueueWelcome(to, name) enqueueReminder(to, delayMs) enqueueDigest()

job.waitUntilFinished ioredis-mock

`GET /queue-health`

`"healthy"` `"degraded"` `"critical"`

`crypto.timingSafeEqual`

```
            X-Correlation-Id
                                crypto.randomUUID()
```

```
reservation.cancelled
```

```

```

```

```

`CircuitBreaker`
`failureThreshold` `resetTimeoutMs`
`fallback` `fire(fn)` `fn`
`state` `stats()`
`'open'` `'close'` `'halfOpen'`

`withRetry(fn, options)`
`shouldRetry(err)` `[{ attempt, durationMs, error|null }]` `{ value, attempts }`

`setTimeout` `jest.useFakeTimers()`

`createHealthRouter(checks, { timeoutMs })` `GET /health/live`
`GET /health/ready`

`Promise.allSettled` `timeoutMs`
`'timeout'`

		`package.json` `package-lock.json` `npm ci`
		`process.env.DATABASE_URL`
		`npm ci`
		`app.listen(process.env.PORT)`

`/metrics`

```
                          USER appuser
docker scout cves
```

```
npm ci    npm install                                    npm audit --audit-level=high
```

```
           .env
```

```

```

-
- `process.env.PORT`
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-

`createServer(app, options)`

`/health/live` `/health/ready`

`/metrics`

`{ server, logger, shutdown }`

`audit-dockerfile.js <path>`

`:latest`

`FROM … AS` `npm ci` `npm install`

`COPY .env`

`createObservability({ serviceName, logLevel })`

`crypto.randomUUID()`

`req.route.path` `req.path`

`error`