

# Index

## Numeric

5-question fitness test

accessing real-time data, 334–335

AI actions, 335

AI tool change, 335–336

integration redundancy, 335

security and governance  
requirements, 335

## A

abstraction, 28, 288, 335, 350

abstraction layer, 15

agent loop, 251, 252, 261–263

aggregators, 92–93

AI. *See* artificial intelligence (AI)

aiolimiter, 328

aiomysql, 273, 274, 293

AI-to-AI communication servers, 368–369

Amplitude tool, 366

Anaconda Prompt, 62, 63, 175, 176, 180,  
184, 192, 196, 209, 222, 225, 228,  
229, 232, 273

analytics and monitoring solutions, 365

Amplitude, 366

Datadog, 366

Grafana, 366

Mixpanel, 366

Anthropic, 1, 11, 21, 32, 91, 92, 99, 123

`@app.get_prompt()`, 258

Application Programming Interfaces (APIs)

authentication, 222–226

adding to server, 222–223

client and server testing, 225–226

client update, 224

Claude Desktop vs, 32–35

gateway integration, 337–338

integration path, 33–34

applications, 33

limitations, 33

users, 33

`@app.list_prompts()`, 258

artificial intelligence (AI)

assistants, 1, 3, 9, 11–13, 18, 22–25,  
27, 28, 268

available resources, 18

as collaborator, 23

functions, 17–19

interact with external systems, 15

needs external data, 10–11

tools, 18

async design, 175

`AsyncExitStack`, 214

asynchronous server, 184, 319–320, 322

`asyncpg`, 320

- audit logs, 310
- authentication, 285, 310
  - application program interfaces (APIs), 222–226
  - HTTP and SSE transports need, 376
  - JSON Web Token (JWT), 226–232
    - adding secret, 226–227
    - client modification, 231–232
    - modified client-server testing, 232
    - server modification, 229–231
    - token generation, 227–229
  - for MCP, 375–376
- automation, 24, 25
- availability tax, 331
- available resources, 16–18, 21
- awareness training, 346
- Awesome MCP Servers, 93

## B

- backup and recovery, high availability architectures (HA), 332
- BI. *See* business intelligence (BI)
- BigQuery, 362
- bi-server, 297–300
- blue-green deployment, 331
- Brave Search server, 96, 125
- Brew, 234
- browser automation (Playwright and Puppeteer), 367–368
- buggy search tool, 278
- buggy ticket tool, 278
- build-*versus*-buy decision, 125
- business intelligence (BI), 287
  - data analysis and, 23–24
  - database schema, 288–291
  - generalized pattern, 304
  - letting large language model (LLM), 287
  - metrics, 291–293
  - packages installation, 291
  - production considerations, 302–304
    - cost, 302
    - failed queries, 303
    - freshness, 302
    - numerical integrity, 304
    - permissions, 302–303
  - questions and responses, 297–302
  - server creation, 293–296
  - server implementation, 288
  - tool, 287, 288
  - wiring server into Claude, 296–297
- business outcome metrics, 351–352
- business users, 11, 94
  - Brave Search server, 96
  - Fetch server, 94–95
  - Filesystem server, 94
  - GitHub server, 95–96
  - Google Drive server, 96
  - MySQL server, 94
  - Puppeteer server, 95
  - SQLite server, 96
  - YouTube Transcript server, 95
- business value
  - cost reduction, 26
  - ecosystem leverage, 28–29

- flexibility and future-proofing, 27–28
- innovation enablement, 28
- security and compliance
  - enhancement, 27
- time to value, 26–27

## C

- cachetools, 282
- caching strategies, 316–318
  - bad caching candidates, 317
  - good caching candidates, 316
- invalidation, 318–319
  - event-based invalidation, 319
  - time-based expiration (TTL), 319
  - versioned keys, 319
- layers of, 317–318
- call\_tool() function, 79
- call tools, 160, 179, 255, 256, 259–263, 265
- canary deployment, 331
- catalog tools, in pattern recognition, 306
- change management
  - accommodating stakeholder perspectives, 344–345
- champions identification and nurturing, 345
- establishing governance, 346–347
- job security question, 345–346
- training and enablement, 346
- checklist
  - Model Context Protocol (MCP) servers, 123
  - prototype to production, 308
    - authentication and network, 310
    - correctness, 308
    - data, 310–311
    - deployment, 309–310
    - documentation and operations, 311
    - final sniff test, 312
    - human layer, 311
    - observability, 309
    - performance, 309
    - safety, 308–309
- circuit breaker, 324
- Claude Desktop, 2, 9, 10, 22, 28, 100–102, 105, 133–135, 137, 153, 157, 259, 268, 269, 285, 298
- vs Application Programming Interface (API) integration, 32–35
- applications, 33
- architecture
  - extensibility, 55
  - security and isolation, 55
  - separation of concerns, 54
  - standardization, 55
- configuration, 74–76
  - claude\_desktop\_config.json, 37, 38, 42, 47, 48, 51, 52, 56
  - to add Fetch server, 108
  - to add Filesystem server, 100
  - to add MySQL server, 104–105
  - to add Puppeteer server, 114–115
  - to add YouTube Transcript server, 111

## Claude Desktop (*continued*)

- Document Q&A System, 66–76
  - add search functions, 70–73
  - build Python-based Model Context Protocol (MCP) Server, 67–70
  - Claude Desktop configuration, 74–76
- installation, 35–40
  - download application, 36
  - install application, 36–37
  - on macOS, 36
  - restart application, 39
  - sign in, 37
  - verification, 39–40
  - on Windows, 37
- limitations, 33
- Model Context Protocol (MCP)
  - support, 37–39
  - configuration structure, 38–39
  - find configuration file, 37
  - open configuration file, 38
  - working status, 39–40
- path, 32–33
- troubleshooting, 51
  - architecture, 54–55
  - boot sequence, 52–53
  - configuration file's role, 56
  - prompts, 58
  - request workflow, 53–55
  - resources, 57
  - tools, 58
  - users, 33
- CLI. *See* command-line interface (CLI)

- client-server architecture, 1, 14
- cloud storage, 154–158. *See also* Google Drive
- command-line interface (CLI), 391
- Common Table Expressions (CTEs), 362
- communication servers, AI-to-AI, 368–369
- community-built browser automation servers, 367–368
- community-built game changers, 367
  - browser automation (Playwright and Puppeteer), 367–368
  - home automation (Home Assistant), 368
  - music and media (Spotify), 368
- community-driven development, 25
- community hubs, 92–93
- compliance enhancement, 27
- compliance frameworks, 243–246
  - general data protection regulation (GDPR), 243–244
  - The Health Insurance Portability and Accountability Act (HIPAA), 244–245
  - production deployment security checklist, 245–246
  - system and organization controls 2 (SOC 2), 245–246
- concurrency, 319, 321, 322. *See also* multiple concurrent connections handling
- Conda, 62
- configuration file role, Claude Desktop, 56
- connectors, 10–12
  - database and data warehouse, 361–362
- continuous integration (CI), 381
- COO, 287, 297, 298, 302

- core components, 15
  - complete picture, 21
  - prompts, 19–21
  - resources, 16–17
  - tools, 17–19
- core patterns, 60, 160
- core value proposition, 334
- correctness, 308
- cost of building, Model Context Protocol (MCP) servers
  - ongoing maintenance capability, 126
  - technical skills, 126
  - testing, 126
  - time, 126
- CPU-heavy work, 322
- `create_ticket`, 269, 277–278
- CRM. *See* customer relationship management (CRM)
- cross-tool connectivity, 25
- CTEs. *See* Common Table Expressions (CTEs)
- curated surface over a powerful backend, in pattern recognition, 305
- custom code, 12–13, 27
- customer support, 21, 24
  - automation with knowledge base, 268
  - architecture, 269–270
  - authentication, 285
  - benefits of, 286
  - build Model Context Protocol (MCP) server, 268–269, 274–278
  - caching, 282–283
  - install package, 273
  - knowledge base schema, 270–273
  - make server production-ready, 281
  - observability, 283–285
  - starting server, 278
  - testing, 285–286
  - user trust, 286
  - wiring server into Claude, 278–282
- custom integrations, 10–12, 26, 27
  - critical problems of, 10
  - multiplication, 10
  - reinvention, 10
  - security, 10

## D

- data, 310–311
- data analysis, 23–24
- database and data warehouse
  - connectors, 361
  - BigQuery, 362
  - PostgreSQL and SQLite, 361–362
  - Snowflake, 362
- database Model Context Protocol (MCP) server, 17
- database queries, 303, 314–315
- data connection
  - cloud storage, 154–158
    - Google Drive setting, 155–156
  - model context protocol (MCP) server setting, 156–158
- file systems and document repositories, 136–140
  - CSV files, 137–140
  - Filesystem server setting, 136–137

- data connection (*continued*)
  - Gmail integration, 143–154
    - API enabling, 143–149
    - credentials file creation, 149–151
    - model context protocol (MCP)
      - config file, 152
    - model context protocol (MCP) server
      - testing, 152–154
    - new project creation, 143
  - PDFs files, 140–142
    - Claude summarizes, 142
    - Markdown server setting, 140–142
  - SQLite integration, 130–136
    - Claude Desktop, 133–135, 137, 153, 157
    - database creation, 130–132
    - macOS users, 130
    - model context protocol (MCP) server
      - setting, 132–133
    - model context protocol (MCP) server
      - testing, 134–136
    - Windows users, 130–131
  - Datadog MCP integration, 366
  - data integration problem, 10
  - data migration considerations, 350
  - data sources, 9, 11–15, 366, 371
  - data streams, 369
  - debugging, 56–57, 378, 381
  - decision tree, 34, 127
  - defense layers, in pattern recognition
    - business-rule validation, 306
    - parameterized queries or escaped
      - commands, 306
    - schema validation, 306
  - degradation, 322–324
  - developer tools, 22–23
  - developer training, 346
  - development and DevOps utilities, 364
    - Docker and Kubernetes, 365
    - GitHub, 364
    - GitLab, 364
    - terminal/shell, 365
  - development environment, 2, 22, 23, 302
  - disk cache, 318
  - Docker and Kubernetes tool, 365
  - docstring, 275, 276, 278, 286, 307, 308
  - documentation, 1, 15, 21, 24, 95, 124, 311
  - Document Q&A System, 59–60
    - components of, 60
      - knowledge base, 60
      - Model Context Protocol (MCP)
        - server, 60
      - prompt strategy, 60
  - connect Claude to documents, 66–76
    - add search functions, 70–73
    - build Python-based Model Context
      - Protocol (MCP) server, 67–70
    - Claude Desktop configuration, 74–76
  - demonstrate value, 60
  - expandability, 60
  - knowledge base set up, 61–66
    - add metadata, 66
    - documents selection, 61–62
    - file organization, 62
    - Python environment set up, 62–64
    - standard format conversion, 64–65

- prompts, test and refine, 76–88
  - testing basic queries, 76–84
  - view log files, 84–88
- teach core patterns, 60
- usefulness, 59
- do-not rules, 264
- do\_stuff, 269

## E

- ecosystem, 10, 11, 22, 91–94, 114, 125
  - leverage, 28–29
- embedded integration pattern, 338–339
- encryption, 221, 233–241
  - certificate creation, 235–236
  - client-server testing, 240–241
  - configuration file creation, 236–239
  - Nginx installation, 234–235
  - reverse proxy, 233–234
  - Secure Socket Layer (SSL) verification, 239–240
- enterprise knowledge management, 23–24
- .env.example trick, 384–385
- error handling, 173, 221, 242, 322–323
  - circuit breakers, 324
  - exponential back-off, 323
  - useful errors, 324
- evaluation suite, 286
- event-based invalidation, 319
- expandability, 60
- expected errors, 322

- experimental servers to watch, 368
  - AI-to-AI communication servers, 368–369
  - multimodal servers, 369
  - real-time data streams, 369
- EXPLAIN PLAN, 315
- exposed, meaning of, 16
- extensibility, 55
- external application program interface (API) calls, 314

## F

- FastMCP tool, 341
- fatal errors, 323
- Fetch MCP server, 94–95, 360
  - installation and configuration, 108–110
- file-based integration, 343
- file-based logger, 296
- file systems and document repositories, 136–140
  - CSV files, 137–140
- Filesystem server, 41–46, 94, 100, 360
  - adding files to folders, 42–46
  - installation and configuration, 41–42, 99–103
  - in video-to-action workflows, 119–120
  - setting, 136–137
- final sniff test, 312
- Finance-focused MCP servers, 367
- fitness test, MCP, 336
- flame graph, 316
- flexibility, 15, 27–28, 350, 376
- future-proofing, 27–28

## G

- general data protection regulation (GDPR), 219, 243–244
- generic chatbots, 268
- genuine bugs, 379
- `get_article`, 269, 276–277, 280–283
- `get_prompt()`, 259
- GitHub MCP server, 46–51, 58, 95–96, 360, 364, 370
  - configuration, 47–49
  - get access token, 47
  - installation, 46–47
  - testing, 49–50
  - troubleshoot connection, 51
- GitHub repository, 46, 51, 123, 368, 370
- GitLab, 364
- Gmail integration, 143–154
  - API enabling, 143–149
  - credentials file creation, 149–151
  - download link, 146, 148
  - model context protocol (MCP) config file, 152
  - model context protocol (MCP) server testing, 152–154
  - new project creation, 143
- Google Drive, 154–158
  - Model Context Protocol (MCP) server, 17, 96, 156–158
  - setting, 155–156
- Google Workspace MCP servers, 363
- GPT-5, 10
- Grafana, 327, 366
- green test, 382
- Gutenberg, 109

## H

- HA. *See* high availability architectures (HA)
- handler function, 70, 258, 314, 321, 382, 390
- hard-coding configuration values, 383
  - configuration, 383–384
  - `.env.example` trick, 384–385
- healthcare and life sciences server, 367
- health check endpoint, 330, 331
- The Health Insurance Portability and Accountability Act (HIPAA), 219, 243–245
- high availability architectures (HA), 329–330
  - backup and recovery, 332
  - deployments, 331
  - multiple instances, 330–331
    - health checks, 330, 331
    - load balancer, 330
    - stateless servers, 330
  - survival, 331
- HIPAA. *See* The Health Insurance Portability and Accountability Act (HIPAA)
- Home Assistant MCP server, 368
- HTTP and SSE transports, 376
- Hugging Face, 93
- human layer, 311

## I

- IDE. *See* Integrated Development Environment (IDE)
- idempotency keys, 305–306

- implementation, 27, 79, 91–93, 173, 179, 288, 306, 348
    - adopters and reference, 22
    - customer support and service automation, 24
    - data analysis and business intelligence, 23–24
    - developer tools, 22–23
    - enterprise knowledge management, 23–24
    - Integrated Development Environment (IDE) integration, 22–23
    - open source ecosystem, 25
    - personal productivity and automation, 25
  - industry servers, specialized, 366
    - financial data (Bloomberg and Alpha Vantage), 367
    - healthcare and life sciences, 367
    - legal research (Westlaw and Casetext), 367
  - in-memory cache, 317–318, 338
  - innovation enablement, 28
  - Integrated Development Environment (IDE), 22–23
  - integration failures, 354
  - integration metrics
    - business outcome metrics, 351–352
    - cost/benefit profile, 352
    - technical performance metrics, 350, 351
    - user experience (UX) metrics, 351
  - integration patterns and architectures
    - API gateway integration, 337–338
    - embedded integration, 338–339
    - shared service (MCP hub), 339
    - sidecar server, 337
    - SSE transport, 340
    - stdio, 340
    - summary of, 339
  - integration test, 381, 382
  - invalidation, caching, 318–319
    - event-based invalidation, 319
    - time-based expiration (TTL), 319
    - versioned keys, 319
  - I/O-heavy work, 322
  - isolation, 55
- ## J
- JavaScript Object Notation (JSON), 38, 48, 254, 255, 315
    - schema, 255–256
  - JavaScript Object Notation Remote Procedure Call (JSON-RPC), 12, 14, 56
  - job IDs, in pattern recognition, 306
  - job security question, 345–346
  - JSON. *See* JavaScript Object Notation (JSON)
  - JSON-RPC. *See* JavaScript Object Notation Remote Procedure Call (JSON-RPC)
  - JSON Web Token (JWT), 303
    - authentication, 226–232
      - adding secret, 226–227
      - client modification, 231–232
      - modified client-server testing, 232
      - server modification, 229–231
      - token generation, 227–229

## K

knowledge-base-server, 268  
    create\_ticket, 269, 277–278  
    get\_article, 269, 276–277  
    search\_articles, 269, 275–276  
knowledge base set up, in Document Q&A System, 61–66  
    add metadata, 66  
    documents selection, 61–62  
    file organization, 62  
    Python environment set up, 62–64  
    standard format conversion, 64–65  
Kubernetes, 331

## L

large language models (LLMs), 9, 138, 157, 250, 261, 262, 287, 304  
    tool calling  
        server-sent events (SSE)  
            implementation, 193–197  
        stdio implementation, 175–183  
legacy integration ladder, 340–341  
legacy systems  
    authentication challenges, 343–344  
    data formats, 342–343  
    file-based integration, 343  
    legacy integration ladder, 340–341  
    relational database, 341–342  
    Robotic Process Automation (RPA)  
        integration, 343  
    Scheduled Extract Transform Load (ETL), 343  
    screen scraping, 343

legal MCP servers, 367  
limit variable, 276  
List Directory tool, 43  
list\_metrics, 288, 296, 300, 302  
list\_prompts(), 259  
LLMs. *See* large language model (LLMs)  
load balancer, 330  
local files access, Filesystem server, 41–46  
    adding files to folders, 42–46  
    configuration, 42  
    installation, 41–42  
log file, 74–76, 84–88, 284–285  
logging, 269, 283, 307, 325–326, 353, 379  
long system prompt, 307

## M

macOS  
    Claude Desktop configuration file on, 38  
    Claude Desktop installation on, 36  
    log file, 285  
Markdown server setting, 140–142  
MCP development, mistakes in, 374  
    failing to plan for scale, 387  
        the 1 client today, 100  
        tomorrow test, 389  
    components to defer, 388–389  
    components to get right early  
        on, 387–388  
forgetting about user  
    experience, 389–390  
hard-coding configuration values, 383  
    configuration, 383–384  
    .env.example trick, 384–385

- ignoring error handling, 377–379
- inadequate testing coverage, 380–382
- neglecting documentation, 385
  - README, 385–387
- overcomplicating the first project, 376–377
- poor resource and tool design, 379–380
- self-audit, 20-minute, 394
- skipping security in development, 374
  - authentication for MCP specifically, 375–376
  - minimum security foundation, 375
- trying to do everything with MCP, 391–393
  - quick decision guide, 393
- MCP ecosystem, contributing to, 370
  - building the first server, 371
  - publishing and sharing, 371
  - reporting issues and improving docs, 370–371
- `mcp.log`, 86
- MCP SDK, 274, 371
- `mcp-server.conf`, 237, 238
- `mcp-server-docs-qa.log`, 86
- `mcp-server-*.log`, 86
- `mcpServers` key, 279
- MCP.so, 93
- measurement infrastructure, 353
- Memcached, 318
- memory and resource caps, 329
- memory server, 361
- metadata, 17, 66, 157
- metrics, in observability, 325–327
  - errors, 327
  - latency, 327
  - saturation, 327
  - traffic, 327
- migration strategies
  - data migration considerations, 350
  - decision making, 347
  - hard-coded API integrations, replacement of, 348–349
  - retrieval-augmented generation (RAG) pipelines, 349–350
  - strangler fig pattern, 348
- Miniconda, 62–63
- Mixpanel tool, 366
- Model Context Protocol (MCP) client, 14, 22, 25, 33, 53–54, 70, 79, 87, 250, 256, 258, 260
  - customizing with system prompt, 251–253
- Model Context Protocol (MCP) servers, 9, 13, 14, 16, 18–20, 249, 250, 259, 359
  - aggregators, 92–93
  - analytics and monitoring solutions, 365
    - Amplitude, 366
    - Datadog, 366
    - Grafana, 366
    - Mixpanel, 366
- Anthropic’s reference servers, 92
- client functions, 160–165
  - agentic workflow, 160–161
  - deployment methods, 162–164
  - initialization handshake, 162
  - Python virtual environment, 164–165

- Model Context Protocol (MCP)
  - servers (*continued*)
  - vs client interaction, 220
  - community-built game changers, 367–368
    - browser automation (Playwright and Puppeteer), 367–368
    - home automation (Home Assistant), 368
    - music and media (Spotify), 368
  - community hubs, 92–93
  - compliance frameworks, 243–246
    - general data protection regulation (GDPR), 243–244
    - The Health Insurance Portability and Accountability Act (HIPAA), 244–245
    - production deployment security checklist, 245–246
    - system and organization controls 2 (SOC 2), 245–246
  - contributing to MCP ecosystem, 370–371
  - database and data warehouse connectors, 361–362
    - BigQuery, 362
    - PostgreSQL and SQLite, 361–362
    - Snowflake, 362
  - deployment methods
    - local execution, 162–163
    - remote access, 163–164
  - development and DevOps utilities, 364–365
    - Docker and Kubernetes, 365
    - GitHub, 364
    - GitLab, 364
    - terminal/shell, 365
  - evaluate new servers, 369–370
    - community adoption, 370
    - documentation quality, 370
    - maintenance and activity, 370
    - security first, 369
  - experimental servers to watch, 368–369
    - AI-to-AI communication servers, 368–369
    - multimodal servers, 369
    - real-time data streams, 369
  - Fetch server, 94–95
  - fitness test, 336
  - industry servers, specialized, 366–367
    - financial data (Bloomberg and Alpha Vantage), 367
    - healthcare and life sciences, 367
    - legal research (Westlaw and Casetext), 367
  - installation and configuration of, 96–97
    - Fetch server, 108–110
    - Filesystem server, 99–103
    - MySQL server, 104–107
    - Puppeteer server, 114–117
    - YouTube Transcript server, 110–114
  - multiple connections, 197–217
    - multiserver client, 210–217
    - two servers creation, 198–209
  - multiple servers integration, 117
    - limit management, 121–122
    - multiserver workflow example, 117–119
    - video-to-action workflows, 119–121
  - MySQL server, 94
  - npx (Node Package Execute) tool, 98–99

- official anthropic servers, 360–361
  - Fetch server, 360
  - Filesystem server, 360
  - memory server, 361
- performance, 314–316
  - optimizing, 315–316
  - tend to slow down, 314–315
- privacy and safe data handling, 241–243
  - cache lifetime, 243
  - privacy-aware patterns, 242
  - secure logging, 242
- productivity tool integrations, 362–363
  - Google Workspace (Gmail, Calendar, Drive), 363
  - Notion and Obsidian, 363
  - Slack, 363
- prompts by, 19–20
- Puppeteer server, 95
- quality and trustworthiness, 123
  - build own server, 125–126
  - build vs buy, 125
  - cost of building, 126
  - decision tree, 127
  - evaluation checklist, 123
  - pause before installing, 124–125
  - prebuilt servers as default choice, 125
  - principle of least privilege, 124
- search with specific tool, 93
- startup behavior, 52–53
  - always available, 52
  - resource usage, 52
  - startup time, 52

- uvx tool, 99
- via server-sent events (SSE)
  - implementation, 184–197
  - large language model (LLM) tool calling, 193–197
  - server implementation, 184–193
- via stdio implementation, 165–183
  - client implementation, 173–175
  - large language model (LLM) tool calling, 175–183
  - server implementation, 165–173
- application program interfaces (APIs)
  - authentication, 222–226
- authentication, 221–232
- encryption, 233–241
- JSON Web Token (JWT)
  - authentication, 226–232
- adding secret, 226–227
- client modification, 231–232
- modified client-server testing, 232
- server modification, 229–231
- token generation, 227–229
- security questions, 221
- YouTube Transcript server, 95
- model ignorance, of system
  - prompt, 263–265
- call limit tool, 265
- confirm prompt transmission, 264
- do-not rules, 264
- explicit language, 264
- move rules to top, 265
- modern artificial intelligence (AI) language models, 10

- monitoring, 325–327, 365–366, 368
- multimodal servers, 369
- multiple concurrent connection handling, 319
  - avoid shared mutable state, 321
  - connection pooling, 320–321
  - server for concurrency, 322
  - synchronous vs asynchronous, 319–320
- multiple servers integration, 117
  - limit management, 121–122
  - video-to-action workflows, 119–121
- multiplication, 10
- multiregional setups, 331
- multiserver workflow, 117–119
  - limit management in, 121–122
- music and media (Spotify), 368
- MySQL Community Edition, 104
- MySQL server, 94, 124, 270–271
  - create database in, 270–271, 274
  - installation and configuration, 104–107
  - in video-to-action workflows, 121, 122

## N

- Network Attached Storage (NAS), 119
- network effect, 13, 25, 28
- Node.js, 41, 95, 98, 99, 104, 114
- Node Package Manager (npm), 41, 42, 93, 98, 99, 123
- Notion and Obsidian MCP server, 363
- npx (Node Package Execute) tool, 98–99

## O

- OAuth flows, 376
- observability, 283–285, 309, 325–327
- official anthropic servers, 360
  - Fetch server, 360
  - Filesystem server, 360
  - memory server, 361
- Ollama, 159, 176, 179, 251, 253, 255, 256, 259
- open authorization (OAuth), 146, 148, 154, 155
- open source, 11, 93, 123
- open source ecosystem, 25
- Oracle database, 341

## P

- parsing, 315
- pattern recognition
  - catalog tools, 306
  - curated surface over a powerful backend, 305
  - defense layers, 306–307
  - idempotency keys on writes, 305–306
  - job IDs, 306
  - read replicas, 307
  - search, fetch, act, 304–305
  - small system prompt, 307
  - split by security profile, 305
  - structured logging, 307
- PDF documents, 140–142
  - Claude summarizes, 142
  - MarkItDown server setting, 140–142

- performance, 53, 309
  - of Model Context Protocol (MCP) server, 314–316
    - handler function, 314
    - optimizing, 315–316
    - tend to slow down, 314–315
- Personally Identifiable Information (PII), 310
- personal productivity, 24
- Playwright, 367–368
- PostgreSQL database, 93, 320, 321, 361–362
- prebuilt Model Context Protocol (MCP) servers, 125
  - Anthropic reference servers, 92
  - community hubs aggregators, 92–93
  - ecosystem of, 2
  - Fetch server, 94–95
  - installation and configuration of, 96–97
    - Filesystem server, 99–103
    - MySQL server, 104–107
    - Puppeteer server, 114–117
    - YouTube Transcript server, 110–114
- multiple servers integration, 117
  - limit management, 121–122
  - multiserver workflow example, 117–119
  - video-to-action workflows, 119–121
- MySQL server, 94
- npm (Node Package Execute) tool, 98–99
- Puppeteer server, 95
- quality and trustworthiness, 123
  - build own server, 125–126
  - build vs buy, 125
  - cost of building, 126
  - decision tree, 127
  - evaluation checklist, 123
  - pause before installing, 124–125
  - prebuilt servers as default choice, 125
  - principle of least privilege, 124
- testing with
  - Filesystem server, 41–46
  - GitHub server, 46–51
- uvx tool, 99
- YouTube Transcript server, 95
- predictable errors, 378
- principle of least privilege, 124
- privacy and safe data handling, 241–243
  - cache lifetime, 243
  - privacy-aware patterns, 242
  - secure logging, 242
- productivity tool integrations, 362–363
  - Google Workspace (Gmail, Calendar, Drive), 363
  - Notion and Obsidian, 363
  - Slack, 363
- Prometheus, 327
- prompt(s), 19–21, 58
  - Document Q&A System, 76–88
    - testing basic queries, 76–84
    - view log files, 84–88
  - engineering skills, 3
  - purposes of
    - consistency, 19
    - discovery, 19
    - efficiency, 19
    - expertise, 19

- proprietary internal system, 125–126
- protected health information (PHI), 244
- Puppeteer, 95, 367–368
  - installation and configuration, 114–117
- Python, 62–64, 320
  - decorator, 70
  - interpreter, 273
  - virtual environment, 164–165
- Python-based Model Context Protocol (MCP) server, 67–70
- Python Package Index (PyPI), 93, 165

## Q

- quantum computing, 141, 142
- query\_metric, 288, 296, 300

## R

- race condition, 321
- random jitter, 323
- rate-limiting, 328–329
  - global, 328
  - memory and resource caps, 329
  - per-tool, 328
  - per-user, 328
  - token bucket, 328–329
- read\_document(), 73
- read\_document tool, 73
- README, 311
- read-only MySQL, 124
- read replicas, in pattern recognition, 307
- real-time data streams, 369
- Redis, 318

- reference servers, of Anthropic, 92
- Registry Pattern, 70
- reinvention, 10
- relational database, 341–342
- repository access, GitHub server, 46–51
  - configuration, 47–49
  - get access token, 47
  - installation, 46–47
  - testing, 49–50
  - troubleshoot connection, 51
- Representational State Transfer (REST) API, 269
- resource management, 329
- resources, 2, 16–17, 43, 52, 57, 58, 67, 70, 256, 335, 338, 350
- retrieval-augmented generation (RAG) pipelines, 349–350
- Robotic Process Automation (RPA) integration, 343
- rolling deployment, 331
- run\_agent\_loop() function, 258

## S

- safety, 308–309
- Scheduled Extract Transform Load (ETL), 343
- schemas, 17, 24, 362
- screen scraping, 343
- search, fetch, act, in pattern recognition, 304–305
- search\_articles, 269, 275–276, 279
- search\_documents(), 73
- search\_documents tool, 73, 87–88

- search functions, 70–73
- Secure Socket Layer (SSL), 235, 236, 239, 240
- security, 10, 55
  - code, 220
  - debt, 374
  - enhancement, 27
  - requirements, 126
  - skipping security in development, 374
    - authentication for MCP specifically, 375–376
    - minimum security foundation, 375
  - teams, 11, 27
- self-contained token, 226
- separation of concerns, 54
- serialization, 163, 315
- `@server.call_tool()` function, 72–73
- `@server.list_resources()` function, 70
- `@server.list_tools()` function, 70
- server quality, 123
  - build own server, 125–126
  - build vs buy, 125
  - cost of building, 126
  - decision tree, 127
  - evaluation checklist, 123
  - pause before installing, 124–125
  - prebuilt servers as default choice, 125
  - principle of least privilege, 124
- `@server.read_resource()` function, 70
- `SERVICES` dict, 214
- Server-Sent Events (SSE), 160, 163, 184, 192, 193, 214
- service automation, 24
- shared cache, 318
- shared service (MCP hub), 339
- shell MCP server, 365
- sidecar pattern, 337
- Slack MCP server, 363
- small system prompt, in pattern recognition, 307
- smoke test, 381
- Snowflake MCP integration, 362
- Software as a Service (SaaS), 268
- split by security profile, in pattern recognition, 305
- Spotify, community servers for, 368
- SQL, 287–291, 293, 300, 304, 305
- SQLite database connection, 130–136
  - Claude Desktop, 133–135, 137, 153, 157
  - database creation, 130–132
  - macOS users, 130
  - model context protocol (MCP) server setting, 132–133
  - model context protocol (MCP) server testing, 134–136
  - Windows users, 130–131
- SQLite server, 96, 132, 361–362
- stakeholder groups and primary concerns, 344–345
- standard format conversion, 64–65
- standard input/output (stdio), 250, 278, 285
  - transport, 376
- standardization, 15, 27, 55
- standardized communication layer, 12
- standard protocol, 13, 106, 146

- starter documents, 61
- stateless servers, 330
- stdio. *See* standard input/output (stdio)
- strangler fig pattern, 348
- structlog, 326
- structured logging, in pattern recognition, 307
- synchronous server, 319–320, 322
- System and Organization Controls (SOC 2), 219, 243, 245–246
- system prompt, 250–251, 303, 307
  - customizing Model Context Protocol (MCP) client, 251–253
  - model ignorance of, 263–265
    - call limit tool, 265
    - confirm prompt transmission, 264
    - do-not rules, 264
    - explicit language, 264
    - move rules to top, 265
  - testing, 259–261
  - tools, 253–259
- system\_prompt, 259

## T

- team adoption
  - accommodating stakeholder perspectives, 344–345
  - champions identification and nurturing, 345
  - establishing governance, 346–347
  - job security question, 345–346
  - training and enablement, 346
- technical performance metrics, 350, 351

- tenant isolation, 310
- Terminal or Anaconda Prompt, 175, 180, 184, 192, 196, 209, 222, 225, 228, 229, 232
- Thunderbolt, 11
- thundering herd fix, 323
- time-based expiration, 319
- time to live (TTL), 282
- time to value, 26–27
- TLS. *See* Transport Layer Security (TLS)
- token bucket, 328–329
- tools, 17–19, 58, 67
  - in artificial intelligence (AI), 17
  - catalog, 306
  - core components, 17–19
  - developer, 22–23
  - discovery, 174
  - invocation, 175
  - name prefixing, 214
  - role message, 214
  - routing, 214
  - troubleshooting, 58
- tools parameter, 254–255
- traced, 283–284, 293, 296
- @traced, 283, 284
- traces, 325, 366
- tracking, 326–327
- traditional MCP servers, 369
- transient errors, 322–323, 378–379
- Transport Layer Security (TLS), 233, 234, 239–240
- travel-assistant prompt, 256–257

trustworthiness, 123

- build own server, 125–126
- build vs buy, 125
- cost of building, 126
- decision tree, 127
- evaluation checklist, 123
- pause before installing, 124–125
- prebuilt servers as default choice, 125
- principle of least privilege, 124

TTL. *See* time-based expiration (TTL)

TypeScript, 336, 371

## U

Unicode Transformation Format - 8-bit (UTF-8), 70

Uniform Resource Identifiers (URIs), 16, 17

unit tests, 285, 305, 380, 381

- for each tool's error paths, 381
- for each tool's happy path, 381

URIs. *See* Uniform Resource Identifiers (URIs)

USB-C, 11

user experience (UX) metrics, 351

user interface (UI), 300, 390

user training, 346

UTF-8. *See* Unicode Transformation Format - 8-bit (UTF-8)

uvx tool, 98, 99

## V

versioned keys, 319

video-to-action workflows, 119–121

Virtual Private Server (VPS), 233, 234

vulnerability analysis, 220–241

- application program interfaces (APIs)
  - authentication, 222–226
- encryption, 233–241
  - certificate creation, 235–236
  - client-server testing, 240–241
  - configuration file creation, 236–239
  - Nginx installation, 234–235
  - reverse proxy, 233–234
  - Transport Layer Security (TLS)
    - verification, 239–240
- JSON Web Token (JWT)
  - authentication, 226–232
  - adding secret, 226–227
  - client modification, 231–232
  - modified client-server testing, 232
  - server modification, 229–231
  - token generation, 227–229
- security questions, 221

## W

web traffic, 233

Wilder, Thornton, 109

Windows

- Claude Desktop configuration file on, 38
- Claude Desktop installation on, 37
- log file, 285

“The Woman of Andros” (Wilder), 109

## Y

YouTube Transcript server, 95

- installation and configuration, 110–114
- in video-to-action workflows, 119–122

