

Moleculer Cheat Sheet

Getting Started & CLI

Moleculer is a fast, modern, and highly scalable microservices framework for Node.js.

- **Install CLI:** `npm install -g moleculer-cli`
- **Initialize Project:** `moleculer init project my-project`
- **Create Service:** `moleculer create service user`
- **Start Runner (Dev Mode):** `npm run dev` (or `moleculer-runner --repl`)
- **Interactive REPL Commands:** `call users.list, services, actions, nodes, emit user.created`

ServiceBroker Architecture

Central orchestrator running on each node.

```
import { ServiceBroker } from
"moleculer";

const broker = new ServiceBroker({
  nodeID: "node-1",
  transporter: "NATS", // Supports
Redis, MQTT, TCP, RabbitMQ
  cacher: "Redis", // Memory, Redis
  caching
  serializer: "JSON", // MsgPack,
ProtoBuf, Avro
  logger: true,
  metrics: true,
  tracing: { enabled: true, exporter:
"Jaeger" },
});

broker.start().then(() => {
  broker.call("math.add", { a: 5, b:
3 }).then(res => console.log(res));
});
```

Service Definition & Actions

```
broker.createService({
  name: "math",
  settings: { defaultOffset: 0 },

  actions: {
    add: {
      params: {
```

```
      a: { type: "number" },
      b: { type: "number" },
    },
    cache: { keys: ["a", "b"], ttl:
30 },
    async handler(ctx) {
      return ctx.params.a +
ctx.params.b;
    },
  },

  // Inter-service action call
  async multiply(ctx) {
    const { a, b } = ctx.params;
    const res = await
ctx.call("otherService.calc", { a, b });
    return res;
  },

  methods: {
    // Internal private helper method
    formatResult(val) { return `Result:
${val}`; }
  }
});
```

Context & Action Invocations

Inside action handlers, access invocation meta-data and remote services via `ctx`.

- `ctx.params`: Received payload parameters
- `ctx.meta`: Metadata dictionary for distributed context & auth
- `ctx.call('service.action', params, opts)`: Remote/local action execution
- `ctx.emit('event.name', payload)`: Balanced event emission
- `ctx.broadcast('event.name', payload)`: Broadcast event to all instances

Event Bus & Pub/Sub

```
broker.createService({
  name: "notifications",
  events: {
    // Balanced event (only one instance
among consumer group receives)
    "user.created"(ctx) {
```

```
      console.log("Send welcome email:",
ctx.params.email);
    },
    // Broadcast event (all service
nodes receive)
    "$services.changed"(ctx) {
      console.log("Services list
updated");
    }
  }
});
```

API Gateway (moleculer-web)

Expose microservice actions over HTTP / Web-Sockets.

```
import ApiService from "moleculer-web";

broker.createService({
  mixins: [ApiService],
  settings: {
    port: 3000,
    routes: [
      {
        path: "/api",
        whitelist: ["users.*",
"posts.*"],
        aliases: {
          "GET users": "users.list",
          "POST users": "users.create",
          "REST posts": "posts", //
Auto-maps RESTful CRUD
        },
        autoAliases: true,
      },
    ],
  },
});
```

Fault Tolerance & Resilience

- **Circuit Breaker:** `circuitBreaker: { enabled: true, threshold: 0.5, minRequestCount: 20 }`
- **Retry Policy:** `retryPolicy: { enabled: true, retries: 3, delay: 100 }`
- **Call Timeout:** `broker.call("service.action", {}, { timeout: 3000 })`

- **Bulkhead:** Limits concurrent in-flight requests per action