Namazu Elements in Five Minutes or Less

Getting Started · https://namazustudios.com/docs/getting-started/elements-in-five-minutes-or-less/

Spin up a local Elements instance and explore a full game backend runtime in minutes.

What Is Namazu Elements?

Namazu Elements is a server-side runtime for connected games.

It can:

- run authoritative game logic
- manage players, inventory, rewards, missions, and leaderboards
- host real-time or turn-based games
- act as your game server via Custom Elements

You can run Elements locally to prototype, test integrations, or build full gameplay logic without deploying anything.

What You'll Get Locally

When you run Elements locally, you get:

- A full Elements backend (API + database)
- Swagger API documentation
- Admin / CMS tools
- Support for Custom Elements (server-side Java code)

This environment behaves the same way as a production deployment, just running on your machine.

Prerequisites

Before you start, make sure you have:

1. Docker Desktop or Docker Compose
2. Git
3. ...that's it!

Run Elements Locally

From a terminal run these three commands:

```
git clone https://github.com/NamazuStudios/docker-compose.git

cd docker-compose

docker compose up
```

After a short startup, Elements will be running locally, and your Elements runtime is live and ready to use.

> More information about the Elements Docker Compose project is available here: https://github.com/Elemental-Computing/docker-compose

---

Verify It's Running

Once started, you can immediately:

- Open the admin UI (also known as the Content Management System, or CMS): http://localhost:8080/admin/login
- Explore the API with Swagger: http://localhost:8080/doc/swagger/

If those load, Elements is up and ready.

Your First Interaction

Here’s a simple way to confirm everything is working.

Create a user via the API:

```
curl -X POST http://localhost:8080/api/rest/user \
  -H "Content-Type: application/json" \
  -d '{"name":"player1","email":"player1@example.com", "password":"example"}'
```

You should receive a JSON response with a user ID and other user info.

You’ve just interacted with your local Elements backend.

Where Does Game Logic Go?

Elements is designed to run authoritative code, which is logic that players cannot modify.

That logic can live:

- inside Custom Elements (running within Elements itself), or
- in external services that call Elements APIs

If this is your first time working with server-side game logic, we recommend starting with the Fundamentals section.

Start here:

- Why You Need Authority
- Elements as a Game Runtime
- Lifecycles and Flows

These pages explain how Elements fits into a game architecture - without assuming backend experience.

Running Locally in the IDE

If you need custom logic or an API that isn't covered by the Elements Core features, try making your own custom Element! Elements makes it easy to develop and debug by letting you run the entirety of Elements with your custom code directly in your IDE. This allows you to view logs and set breakpoints to fully test your application before even making your first deployment.

> Since Elements uses Java, we highly recommend using IntelliJ as your IDE.

Head over to the Custom Code section to get an overview, or check out the platform specific setup guides here:

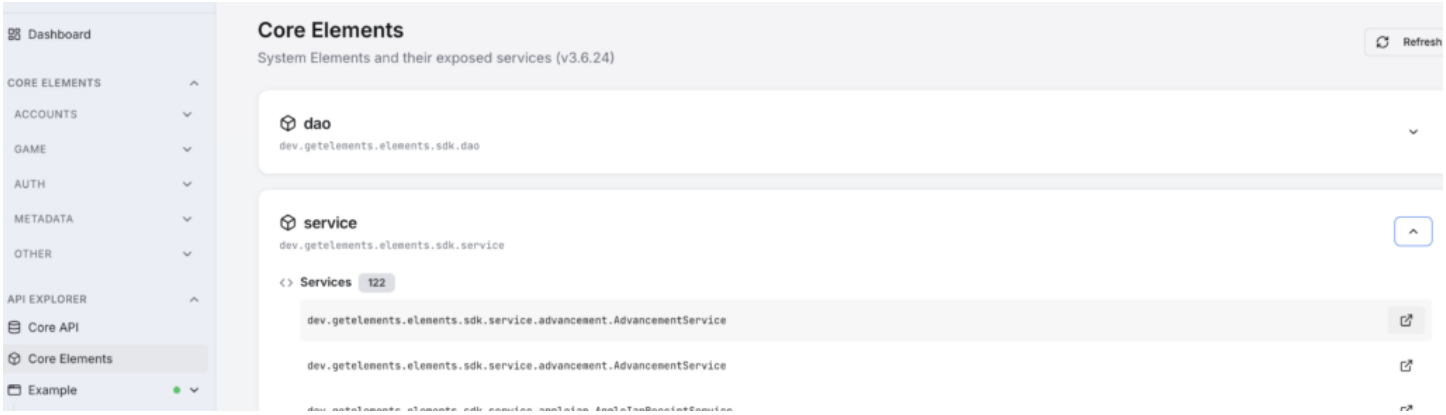
- Windows
- Mac
- Linux

Once you're done, check out our example project here to see how to build a custom Element, debug your work locally, and make a deployment.

Accessing API Documentation

Each instance of Elements provides a copy of the documentation specific to the release. It also provides Swagger json files which you can access. The documentation will vary based on the version. For example: https://javadoc.getelements.dev/3.6.24/index.html

These pages are also accessible via the CMS under the Core Elements section, like so:



What's Next?

Now that you’re running Elements locally, start by:

1. Exploring the fundamentals:
    - Learn how authority, lifecycles, and Custom Elements work together.
2. Or dive right in!
    - Exploring the Content Management System (CMS)
    - Creating your first user
    - Exploring the API with Swagger
    - Understanding core concepts
    - Building a Custom Element