The idea behind ....

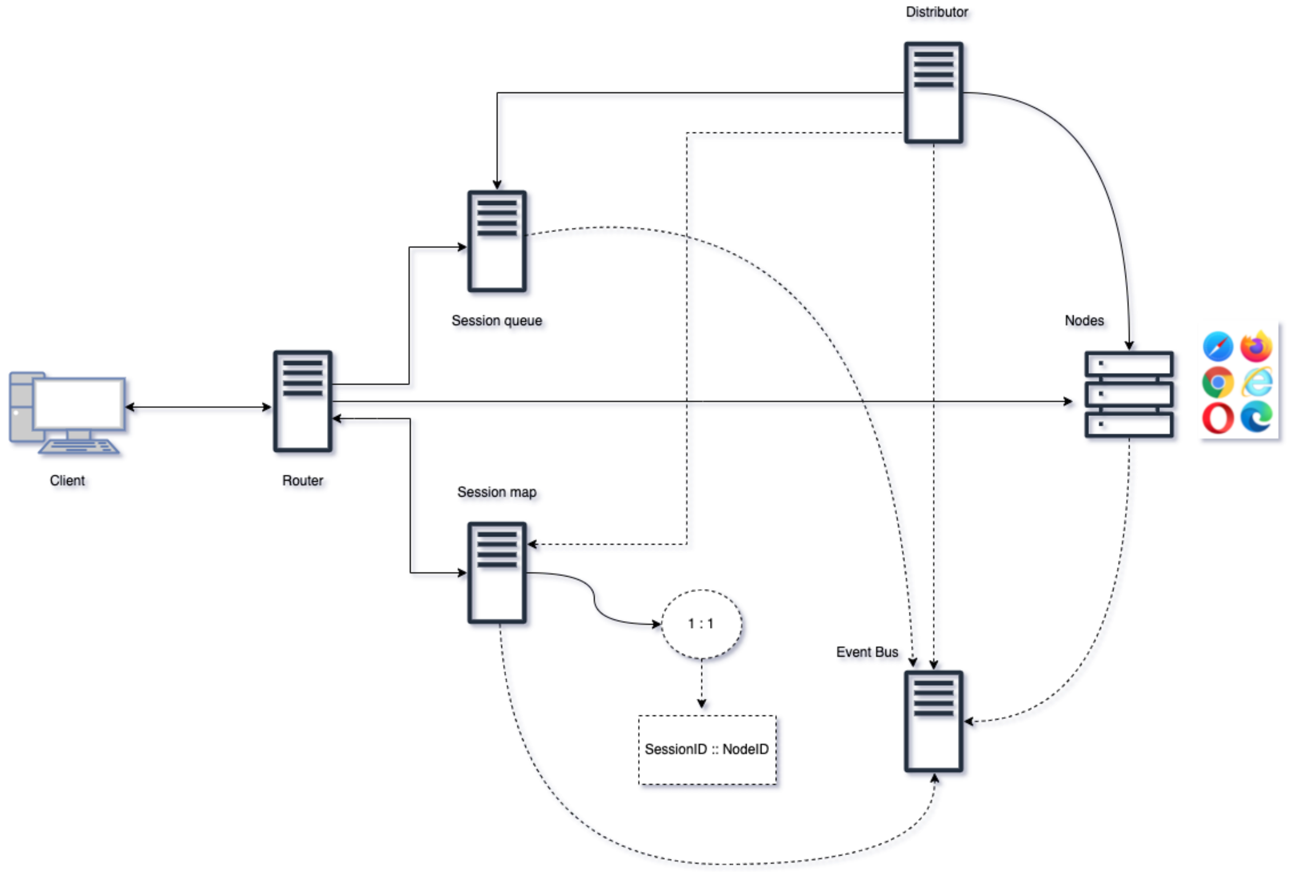
- Better efficiency.
- Higher availability.
- Run tests concurrently without burdening a single resource.
- Derive capabilities to execute tests in parallel for faster execution.

The open source way to do it ....

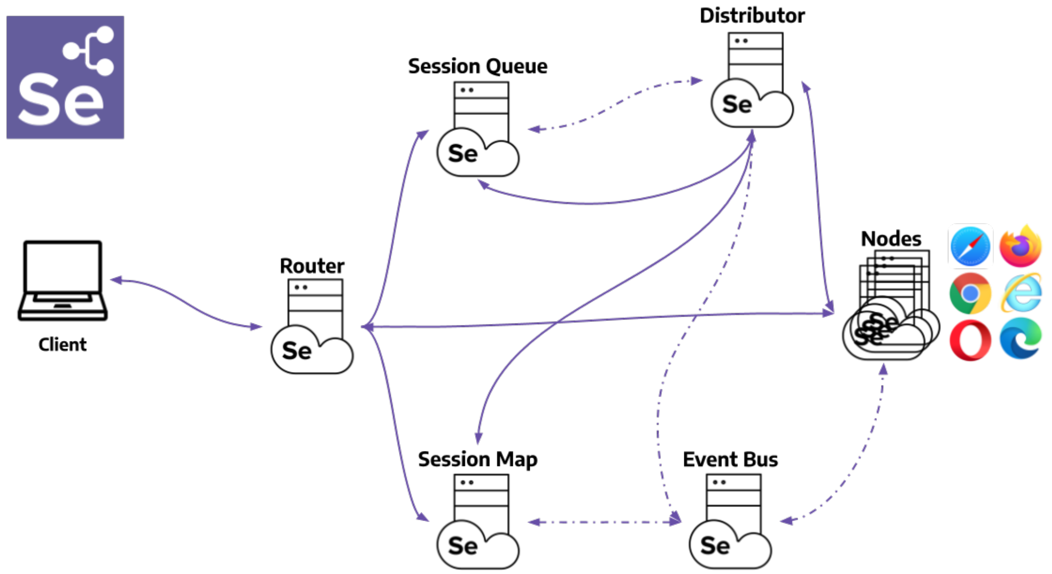
Selenium Grid

- Selenium Grid 4 was the easy option.
  - Comes packaged with docker.
  - Browser options comes pre-packaged with docker.
  - Hub comes pre-packaged with docker.
  - Scale nodes as per demand.
  - Easy to deploy on cloud prem.

Architecture ....



Components ....



- Router
- Distributor
- Session Map
- Session Queue
- Node
- Event Bus

Flow ....

Router

The **Router** received all external requests, and forwards them.

If the Router receives a new session request, it will be forwarded to the **New Session Queue**.

If the request belongs to an existing session, the **Router** will query the **Session Map** to get the **Node ID** where the session is running, and then the request will be forwarded directly to the Node.

Distributor

The **Distributor** has two main responsibilities:

Register and keep track of all Nodes and their capabilities

A Node registers to the Distributor by sending a Node registration event through the Event Bus. The Distributor reads it, and then tries to reach the Node via HTTP to confirm its existence.

Query the New Session Queue and process any pending new session requests

The Distributor will poll the **New Session Queue** for pending new session requests, and then finds a suitable Node where the session can be created.

After the session has been created, the Distributor stores in the **Session Map** the relation between the **session id** and **Node** where the session is being executed.

Session Map

The **Session Map** is a data store that keeps the relationship between the session id and the Node where the session is running.

It supports the **Router** in the process of forwarding a request to the Node. The **Router** will ask the **Session Map** for the **Node** associated to a **session id**.

New Session Queue

The New Session Queue holds all the new session requests in a **FIFO** order.

The New Session Queue regularly checks if any request in the queue has timed out, if so the request is rejected and removed immediately.

The **Distributor** polls the **New Session Queue** for the first matching request. The Distributor then attempts to create a new session.

Once the requested capabilities match the capabilities of any of the free **Node** slots, the Distributor attempts to get the available slot. If all the slots are busy, the **Distributor** will send the request back to the queue.

After a session is created successfully, the Distributor sends the session information to the New Session Queue, which then gets sent back to the Router, and finally to the client.

Flow (Continued) ....

Node

A Grid can contain multiple **Nodes**. Each Node manages the slots for the available browsers of the machine where it is running.

By default, the **Node** auto-registers all browser drivers available on the path of the machine where it runs. It also creates one slot per available CPU. Through a specific configuration, it can run sessions in Docker containers.

A **Node** only executes the received commands, it does not evaluate, make judgments, or control anything other than the flow of commands and responses.

Event Bus

The **Event Bus** serves as a communication path between the **Nodes**, **Distributor**, **New Session Queue**, and **Session Map**.

The Grid does most of its internal communication through messages, avoiding expensive HTTP calls.

When starting the Grid in its fully distributed mode, the **Event Bus** is the first component that should be started.