JASECI

Getting Started

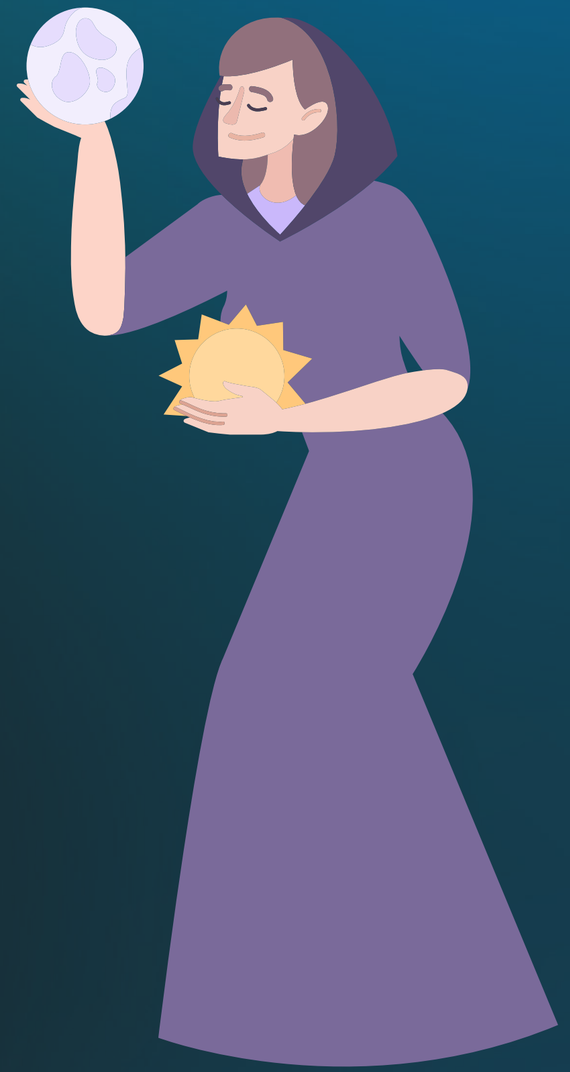
Coursework Series : Week 4

Coursework Outline & Outcomes

Jaseci in Production

What are Sentinels?

sentinel is the overseer of walkers, nodes and edges. It is the abstraction Jaseci uses to encapsulate compiled walkers and architype nodes and edges. The key operation with respesct to sentinel is "register" a sentinel. You can think of registering a sentinel as a compiling your jac program. The walkers of a given sentinel can then be invoked and run on arbitrary nodes of any graph.



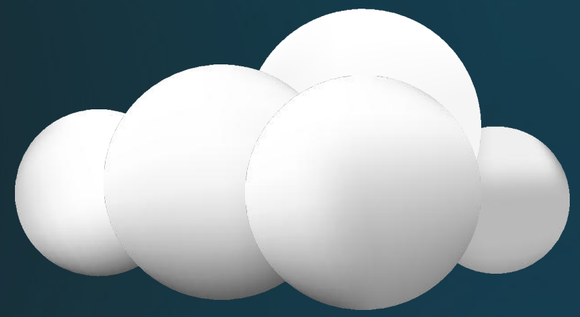
Using Sentinels

Let's register our jac program

```
jaseci > sentinel register tesla_ai.jir -set_active true -mode ir
```

Three things are happening here:

- First, we registered the jir we compiled earlier to new sentinel. This means this new sentinel now has access to all of our walkers, nodes and edges. -mode ir option speciifes a jir program is registered instead of a jac program.
- Second, with -set_active true we set this new sentinel to be the active sentinel. In other words, this sentinel is the default one to be used when requests hit the Jac APIs, if no specific sentinels are specified.
- Third, sentinel register has automatically creates a new graph (if no currently active graph) and run the init walker on that graph. This behavior can be customized with the options -auto_run and -auto_create_graph.

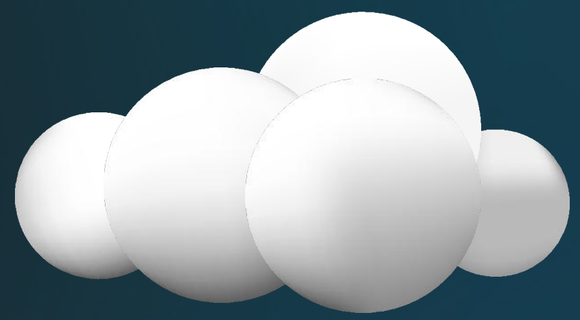


Using Sentinels

Once a sentinel is registered, you can update its jac program with

`jaseci > sentinel set -snt SENTINEL_ID -mode ir tesla_ai.jir`
To get the sentinel ID, you can run one of the two following commands

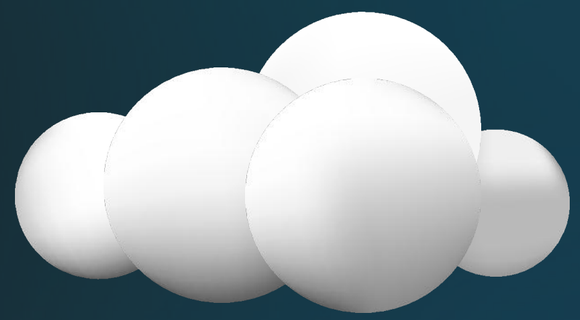
`jaseci > sentinel get` or `jaseci > sentinel list`
sentinel get returns the information about the current active sentinel, while sentinel list returns all available sentinels for the user. The output will look something like this



Using Sentinels

With a sentinel and graph, we can now run walker with

```
jaseci > walker run talk -ctx "{\"question\": \"I want to schedule a
test drive\"}"
```



JASECI

Tests

Just like any program, a set of automatic tests cases with robust coverage is essential to the success of the program through development to production. Jac has built-in tests support and here is how you create a test case in jac.

```
import {*} with "tesla_ai.jac";

test "testing the Tesla conv AI system"
with graph::tesla_ai by walker::talk(question="Hey I would like to go on a test drive"){
    res = std.get_report();
    assert(res[-1] == "To set you up with a test drive, we will need your name and addre
}
```

To run jac tests, save the test case(s) in a file (say tests.jac) and import the neccessary walkers and graphs. Then run

```
jaseci > jac test tests.jac
```

This will execute all the test cases in tests.jac squentially and report success or any assertion failures.

Using Jaseci as Backend

With a jaseci web server (i.e., jsserv), we also get access to the full suite of Jaseci RESTful APIs. You can go to http://localhost:8000/docs/ to checkout the list of available APIs and their documentation and request and response payload format.

Every jsctl command has a corresponding API endpoint. Click on the triangle to the right of the endpoint to see details on its request and response format. The command line argument to the jsctl command becomes the fields in the request payload of the API endpoint. jsctl is great for rapid development and testing but for deploying a jaseci application to production, a set of RESTful API endpoints is the industry standard. Here are the most commonly used endpoints that you should pick up and get familarized first.

Next week

1. **Introduction to Jaseci AI Kit. Create an AI Application**

ASSIGNMENT 2 (Deadline 17th April)
Groups of 5, Need to add a new feature to the Ancestry Example. Need to Submit a Video of the feature in action and the Source code.

Allowed to make changes to every aspect of the app including nodes, edges, walkers, and dataset.

Example - You can add a new field to the node saying hobbies, you can cluster people who have similar hobbies etc.