REST API

Each Inductor Agent has a REST API with which we can access its current state and session information. The REST resources are as follows:

GET *http://<IP>:<Port>/node*

Returns compute node technical details. An example response can be seen in listing 4.5.

```
{
    "disk": 499963170816,
    "machine": "x86_64",
    "memory": {
        "swap": 4294967296,
        "total": 17179869184
        },
    "node": "hal985m.local",
    "processor": {
        "cores_logical": 8,
        "cores_physical": 4,
        "frequency_current": 2700,
        "frequency_max": 2700,
        "frequency_min": 2700,
        "type": "i386"
        },
    "release": "18.7.0",
    "system": "Darwin",
    "version": "Darwin Kernel Version 18.7.0: Sat Oct
       (cont.)12 00:02:19 PDT 2019;
    root:xnu-4903.278.12~1/RELEASE_X86_64"
}
```

Listing 4.5: Node details

GET *http://<IP>:<Port>/chaos/session*

Returns the currently set session descriptor data. If none was yet defined by the user a default one will be shown.

PUT *http://<IP>:<Port>/chaos/session*

Used to send a session descriptor as shown in listings 4.3 and 4.4 respectively.

GET *http://<IP>:<Port>/chaos/session/execute*

Get the anomalies to be executed in the current session descriptor.

PUT *http://<IP>:<Port>/chaos/session/execute*

Generates the final session, no payload is required. If different distributions and stagger time is set they will be now added. Also, if randomization was defined it will take effect now. It should be noted that this action is required. It is not enough to upload the initial session descriptor, this resource has to be activate in order to generate the final session descriptor. Now the GET request will return the final generated session descriptor instead of the initial one.

POST *http://<IP>:<Port>/chaos/session/execute*

Executes the final session descriptor. If the previous resource is not used to generate the final sessions descriptor it will be generated automatically together with a warning message. This resource is a protected one thus it is necessary to include the *x-access-tokens* header together with the appropriate token. This token can be viewed as a shared secret between the CI service and Inductor Agents and has to be generated during initial deployment.

```
{
    "0373b6b6-ad5c-4f5b-a16d-ac29f0a4798a": {
        "anomaly": "dummy",
        "options": {
            "stime": 10
            }
    },
    "1acc58a9-0baf-4a8b-b3c3-d36c5d662a95": {
        "anomaly": "cpu_overload",
        "options": {
            "half": 1,
            "time_out": 15
            }
    }
}
```

Listing 4.6: Session execution example

Listing 4.6 show an example output. It should be noted that this will contain a *job_id* for all started anomalies.

GET *http://<IP>:<Port>/chaos/session/execute/jobs*

Returns a complete listing of all the current scheduled jobs/anomalies. They are split into 4 statuses or registries; *failed*, *finished*, *queued*, *started*. Listing 4.7 show

an example response.

```
{
    "jobs": {
        "started": [
            "2374c649-ec96-4a26-a273-a35172c4cc0c",
            "dae6fe4f-c47b-4877-be78-c25bf223b5a8",
            "9fbca075-1dfb-4751-ab93-b899efc833dd",
            "d6f8e2ec-5dbb-4244-88e4-2ff062a67795",
            "35cf653d-13fb-49c3-afed-17411f95c9ad",
            "36c8f740-e08d-4cf2-baf0-08a8fec75ac4"
            ],
        "finished": [
            "9ff0f7bd-cebc-4c7d-802f-e8387ceafb43",
            "447559f8-700f-4152-a704-a2140df18874",
            "d3442b85-fb02-445d-9f57-df1d66501f85",
            "d0d00ec2-5b30-4632-86df-d3bfbec4555d"
            ],
        "queued": [
            "78efbf00-88c6-4e8f-97a1-32e9fd0cee4a",
            "8299df3e-43fb-4c7c-9d66-4bc20b6358c2",
            "bcc580b2-d4dd-41a7-97e0-7b4217b77215"
            ],
        "failed": [
            "077660a3-3700-4b4c-b0de-3676a4b5b8f3"
            ]
    }
}
```

Listing 4.7: Example job status

DELETE *http://<IP>:<Port>/chaos/session/execute/jobs*

Resets all registries by deleting all jobs from the queue.

GET *http://<IP>:<Port>/chaos/session/execute/jobs/<job_id>*

Returns the current status of a job based on it's *job_id*.

DELETE
http://<IP>:<Port>/chaos/session/execute/jobs/<job_id>

Deletes a job based on it's *job_id*. This is done by removing the job from the queued registry. If the job is currently in the started registry CI will try to stop the process however, some anomalies might still hang or refuse to exit correctly particularly if

the compute node is under heavy load. This fact is marked in the Inductor agent log file and is accounted for during dataset generation.

GET *http://<IP>:<Port>/chaos/session/execute/jobs/logs*

Each anomaly will log its progress and timestamps in a log file. This log information will be used to generate the labeled datasets. This resource will show all available logs.

POST *http://<IP>:<Port>/chaos/session/execute/jobs/logs*

Once this request has been sent an archive containing all available logs will be created and sent as a response.

DELETE *http://<IP>:<Port>/chaos/session/execute/jobs/logs*

Deletes all logs.

GET
http://<IP>:<Port>/chaos/session/execute/jobs/logs/<x>.log

Fetches a particular log file. The log file naming conventions are <*anomaly_name*>-*out.log*.

GET *http://<IP>:<Port>/workers*

Returns all available inducer agent workers and their status as seen in listing 4.8.

```
{
    "workers": [
        {
            "id": "ff32c347f9c4437bb4039f2c8369c29f",
            "pid": 27486,
            "status": true
        }
    ]
}
```

Listing 4.8: Available workers

POST *http://<IP>:<Port>/workers*

Starts a new inducer agent worker. This allows the execution of more than one anomaly simultaneously. The *x-access-tokens* header is required.

DELETE *http://<IP>:<Port>/workers*

Stops an inducer agent worker instance. The *x-access-tokens* header is required.

Generation of training datasets for EDE

Each anomaly implementation generates a standardized log format containing a timestamp (UTC format), flags for each step of execution (i.e. *started*, *applied modifier*, *finished*, *failed*). Based on the data available in these logs we can easily generate accurate labeled datasets once combined with the monitoring data from the ASPIDE monitoring solution.

In order to support a large scale distributed deployment the Command and Control API (*EciCommand* class instance) is capable of controlling all Inductor agents via their REST API. This API is also responsible for collecting and generating the labeled datasets mentioned previously. It will match the metric timestamps from the monitoring platform with the anomaly log timestamps.

Prototyping Testbeds

We have also created two portable testbed clusters. These contain the basic building blocks necessary to run a distributed application on Spark and monitor it using Prometheus. These portable testbeds can be found in this repository:

https://github.com/IeAT-ASPIDE/prometheus-test-cluster