Genisys Documentation

Release 0.1.1

Anthony Lapenna

October 02, 2015

Contents

1 Contents **3**

1.1 Setup 3

1.2 Configuration 4

1.3 HTTP API 5

1.4 Connector 7

2 Indices and tables **9**

Genisys is one of the main components of the Skynet stack.

It's main goal is to manage resources associated to a service.

It is able to communicate with different kinds of computes (such as AWS, Docker, VMWare VSphere...) via specific connectors.

Each connector is in charge of managing the resources (scale up/down) of its associated compute, Genisys is the component in charge of balancing the scale orders to the different computes.

CHAPTER 1

Contents

1.1 Setup

1.1.1 Docker

A public Docker image is available and can be used to start the component:

```
$ docker run -p "7001:7001" cyberdynesystems/genisys:latest
```

Do not forget to map the port 7001 of the container to a specific port on the Docker host.

Overriding the configuration

You can map your own configuration file in the container file system:

```
$ docker run -p "7001:7001" -v "/path/to/config/genisys.yml:/app/genisys.yml" cyberdynesystems/genisy
```

1.1.2 From sources

Requirements

Ensure you have *python* >= 3.4 and *git* installed on your system.

Installation

Clone this repository and install the dependencies using **pip**:

```
$ git clone https://github.com/cyberdyne-corp/genisys && cd genisys
$ pip install -r requirements.txt
```

Start

Start the component:

```
$ python main.py
```

1.2 Configuration

1.2.1 Configuration file

The configuration file *genisys.yml* is written in YAML format.

```
genisys:
  # Server address to bind to.
  bind: 127.0.0.1

  # Application port
  port: 7001

  # Path to compute definitions
  compute_file: ./computes.py

consul:
  # Consul host
  host: localhost

  # Consul port
  port: 8500

  # Prefix of services managed by genisys
  service_prefix: 'skynet_'

connector:
  # How frequently to poll all connectors for service status
  poll_interval: 15s
```

Genisys section

This section is related to the component configuration.

bind

The server address to bind to.

port

The port that will be used to communicate with the component via HTTP.

compute_file

A python file that defines an optional list of computes that will be loaded by the component during startup.

See *Compute definition* below for more information on the format of the file.

Consul section

This section is related to the Consul service registry.

host

The host where is located the Consul server.

port

Port associated to the Consul server.

service_prefix

Genisys will poll informations about a selected list of services from the Consul service registry. This option is used to configure which services will be targeted by Genisys management.

Connector section

This section is related to Genisys connectors.

poll_interval

Determines how frequently Genisys will poll its connectors to retrieve each service status.

1.2.2 Compute definition

A compute definition defines a link to a connector that will manage specific compute resources (Docker containers, AWS instances...) associated with a service.

A compute definition looks like:

```
myCompute = {
  "name": "myCompute",
  "connector": "http://localhost:7051"
}
```

A compute definition must include a *name* and a *connector*.

The *connector* field is the URL to the genisys connector used to manage compute resources associated to this compute.

An optional *service_file* (see *compute_file*) can be used to define services using the format defined above. These definitions will be loaded during the connector startup.

1.3 HTTP API

Genisys exposes a HTTP API. It can be used to perform CRUD actions on computes and trigger remote procedure calls on services.

NOTE: The examples use the httpie CLI to query the API.

1.3.1 Compute HTTP endpoint

The following endpoints are exposed:

- */compute*: List service definitions or register a new service definition
- */compute/<compute_name>*: Retrieve or update a service definition

/compute

This endpoint is used to list existing compute definitions or to create a new compute definition.

It supports the following methods: POST and GET.

When hitting the endpoint with a GET, it returns a JSON body like this:

```
{
  "compute_nameA": {
      "name": "compute_nameA",
      "connector": "http://connector.domain:port",
  },
  "compute_nameB": {
      "name": "compute_nameB",
      "connector": "http://other-connector.domain:port",
  }
}
```

When hitting the endpoint with a POST, it expects a JSON request body that must look like:

```
{
  "name": "compute_name",
  "connector": "http://connector.domain:port",
}
```

All fields are mandatory.

The *compute* field is used to identify the compute.

The *connector* field specifies the URL to a genisys connector that will manage the backend.

Example:

```
$ http POST :7001/compute name="localdc" connector="http://localhost:7051"
```

/compute/<compute_name>

This endpoint is used to retrieve a compute definition or to update it.

It supports the following methods: PUT and GET.

When hitting the endpoint with a GET, it returns a JSON body like this:

```
{
  "connector": "http://localhost:7051",
  "name": "localdc"
}
```

When hitting the endpoint with a PUT, it expects a JSON request body that must look like:

```
{
      "connector": "http://localhost:7051"
}
```

The *connector* field is mandatory.

The *connector* field specifies the URL to a genisys connector that will manage the backend.

Example:

```
$ http PUT :7001/compute/local connector="http://localhost:7052"
```

1.3.2 Service HTTP endpoint

The following endpoints are exposed:

- */service/<service_name>/scale*: Ensure that a specific number of compute resource is running for a service

`/service/<service_name>/scale`

This endpoint is used to ensure that specific number of compute resources associated to a service are running.

It expects a JSON request body to be POST.

The request body must look like:

```
{
  "number": "number_of_compute_resources",
  "compute": "compute_name"
}
```

The *number* field is mandatory.

The *compute* field is used to identify the compute in which the compute resource will be created. If not specified, Genisys will automatically pick up the first compute defined.

Example:

```
$ http POST :7001/service/myService/scale number=3 compute="local"
```

1.4 Connector

1.4.1 Available connectors

List of existing connectors:

- Docker: genisys-connector-docker

1.4.2 How can I create my connector?

A connector must expose a HTTP API with specific endpoints:

- */service/<service_name>/scale*: Endpoint used to ensure that a number of containers are running for a service
- */service/<service_name>/status*: Endpoint used to return the number of running resources for a service

These are the **mandatory** endpoints. *Genisys* will use them when trying to automatically scale services.

The connector is not limited to these and could expose other endpoints (for example, a specific endpoint to define a service).

/service/<service_name>/scale

This endpoint must ensure that a specific number of containers associated to a service are running.

It must expects a JSON request body to be POST.

The request body must look like:

```
{
  "number": number_of_containers,
}
```

The *number* field is mandatory.

/service/<service_name>/status

This endpoint must returns the number of running resources for a service managed by this connector.

When hitting the endpoint with a GET, it must returns a JSON body like this:

```
{
  "running_resources": number_of_running_resources,
}
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search