GalaxyCloudRunner Documentation

Release 0.3.0+dev

Galaxy and GVL projects

Jul 14, 2019

Contents:

GalaxyCloudRunner enables bursting of user jobs to remote compute resources for the Galaxy application. It provides several dynamic job rules that can be plugged into Galaxy, enabling Galaxy to submit jobs to remote compute nodes.

CHAPTER 1

How it works

The GalaxyCloudRunner provides a library of rules that can be plugged into Galaxy through its configuration via `job_conf.xml`. Once configured, Galaxy jobs can be automatically routed to a Galaxy remote job runner, called Pulsar, on nodes running on the cloud. Adding a new node is a simple matter of visiting the CloudLaunch site and launching a new worker node on your desired cloud. The GalaxyCloudRunner will discover what Pulsar nodes are available by querying the CloudLaunch API.

CHAPTER 2

Getting Started

Getting started with the GalaxyCloudRunner is a simple process:

1. Configure Galaxy to use GalaxyCloudRunner job destination rules
2. Launch as many worker nodes as you need through CloudLaunch
3. Submit jobs as usual

2.1 Configuring Galaxy

2.1.1 Configuring Galaxy 19.01 or higher

1. Edit your *job_conf.xml* in the <*galaxy_home*>*/config* folder and add the highlighted sections to it.

   You will need to add your own value for the `cloudlaunch_api_token` to the file. Instructions on how to obtain your CloudLaunch API key are given below.

Note: If you do not have the Galaxy configuration file (i.e., *config/galaxy.yml*), either create it (by making a copy of the *.sample* version of the file) or explicitly install the *galaxycloudrunner* library into Galaxy's virtual env, as per docs below (section on installing GCR for Galaxy <v19.01).

```
<?xml version="1.0"?>
<job_conf>
    <plugins>
        <plugin id="local" type="runner" load="galaxy.jobs.runners.local:LocalJobRunner" workers="4"/>
        <plugin id="pulsar" type="runner" load="galaxy.jobs.runners.pulsar:PulsarRESTJobRunner"/>
    </plugins>
    <destinations default="galaxycloudrunner">
        <destination id="local" runner="local"/>
```

(continues on next page)

(continued from previous page)

```
        <destination id="galaxycloudrunner" runner="dynamic">
            <param id="type">python</param>
            <param id="function">cloudlaunch_pulsar_burst</param>
            <param id="rules_module">galaxycloudrunner.rules</param>
            <param id="cloudlaunch_api_endpoint">https://launch.usegalaxy.org/
↪cloudlaunch/api/v1</param>
            <!-- Obtain your CloudLaunch token by visiting: https://launch.usegalaxy.
↪org/profile -->
            <param id="cloudlaunch_api_token">37c46c89bcbea797bc7cd76fee10932d2c6a2389
↪</param>
            <!-- id of the PulsarRESTJobRunner plugin. Defaults to "pulsar" -->
            <param id="pulsar_runner_id">pulsar</param>
            <!-- Destination to fallback to if no nodes are available -->
            <param id="fallback_destination_id">local</param>
            <!-- Pick next available server and resubmit if an unknown error occurs --
↪>
            <resubmit condition="unknown_error and attempt &lt;= 3" destination=
↪"galaxycloudrunner" />
        </destination>
    </destinations>
    <tools>
        <tool id="upload1" destination="local"/>
    </tools>
</job_conf>
```

2. Launch as many worker nodes as you need through CloudLaunch. The job rule will periodically query CloudLaunch, discover these new nodes, and route jobs to them. Instructions on how to launch new Pulsar nodes are below.

3. Submit jobs as usual.

2.1.2 Configuring Galaxy versions lower than 19.01

1. First install the GalaxyCloudRunner into your Galaxy virtual environment.

```
cd <galaxy_home>
source .venv/bin/activate
pip install --upgrade galaxycloudrunner
```

2. For prior prior to Galaxy 19.01, you will need to add a GalaxyCloudRunner job rule to your Galaxy configuration by pasting the following file contents into your Galaxy job rules folder in: *<galaxy_home>/lib/galaxy/jobs/rules/*.

   Create a file named galaxycloudrunner.py and paste the following contents into the file at the location above.

```
from galaxycloudrunner.runners.cl_pulsar_burst import get_destination


def cloudlaunch_pulsar_burst(app, referrer,
                             cloudlaunch_api_endpoint=None,
                             cloudlaunch_api_token=None,
                             pulsar_runner_id="pulsar",
                             pulsar_file_action_config=None,
                             fallback_destination_id=None):
    return get_destination(app, referrer,
                           cloudlaunch_api_endpoint,
```

(continues on next page)

(continued from previous page)

```
                                cloudlaunch_api_token,
                                pulsar_runner_id,
                                pulsar_file_action_config,
                                fallback_destination_id)
```

3. Edit your *job_conf.xml* in the *<galaxy_home>/config* folder and add the highlighted sections to it.

   You will need to add your own `cloudlaunch_api_token` to the file. Instructions on how to obtain your CloudLaunch API key are given below. If you have a Galaxy version prior to 19.01, the line *<param id="rules_module">galaxycloudrunner.rules</param>* passed to your destination will not work. This is the reason that we need to perform step 2.

```
<?xml version="1.0"?>
<job_conf>
    <plugins>
        <plugin id="local" type="runner" load="galaxy.jobs.runners.local:LocalJobRunner" workers="4"/>
        <plugin id="pulsar" type="runner" load="galaxy.jobs.runners.pulsar:PulsarRESTJobRunner"/>
    </plugins>
    <destinations default="galaxycloudrunner">
        <destination id="local" runner="local"/>
        <destination id="galaxycloudrunner" runner="dynamic">
            <param id="type">python</param>
            <param id="function">cloudlaunch_pulsar_burst_compat</param>
            <param id="cloudlaunch_api_endpoint">https://launch.usegalaxy.org/cloudlaunch/api/v1</param>
            <!-- Obtain your CloudLaunch token by visiting: https://launch.usegalaxy.org/profile -->
            <param id="cloudlaunch_api_token">37c46c89bcbea797bc7cd76fee10932d2c6a2389</param>
            <!-- id of the PulsarRESTJobRunner plugin. Defaults to "pulsar" -->
            <param id="pulsar_runner_id">pulsar</param>
            <!-- Destination to fallback to if no nodes are available -->
            <param id="pulsar_fallback_destination_id">local</param>
        </destination>
    </destinations>
    <tools>
        <tool id="upload1" destination="local"/>
    </tools>
</job_conf>
```

4. Launch as many worker nodes as you need through CloudLaunch. The job rule will periodically query CloudLaunch, discover these new nodes, and route jobs to them. Instructions on how to launch new worker nodes are following.

5. Submit your jobs as usual.

2.1.3 Reducing data transfers

If you would like to control the data transfer configurations for Pulsar, an additional option can be specified in the `job_conf` destination for the GalaxyCloudRunner rule. This is particularly useful for Galaxy's reference data because the remote Pulsar nodes have been configured to mount the Galaxy public file system repository with preformatted reference data for a number of tools. In turn, this speeds up job execution and reduces data transfers from your Galaxy instance because the relevant files do not need to be transferred to the remote node with each job.

Note that this configuration is necessary only if your file system paths differ from those on the remote Pulsar nodes. Specifically for the reference data, Pulsar nodes mount Galaxy Project's CVMFS repository, which is available under `/cvmfs/data.galaxyproject.org/` directory. The layout of that directory can be inspected here: https://gist.github.com/afgane/b527eb857244f43a680c9654b30deb1f

To enable this feature for the GalaxyCloudRunner, it is necessary to add the following `param` to the existing job destination in `job_conf.xml`:

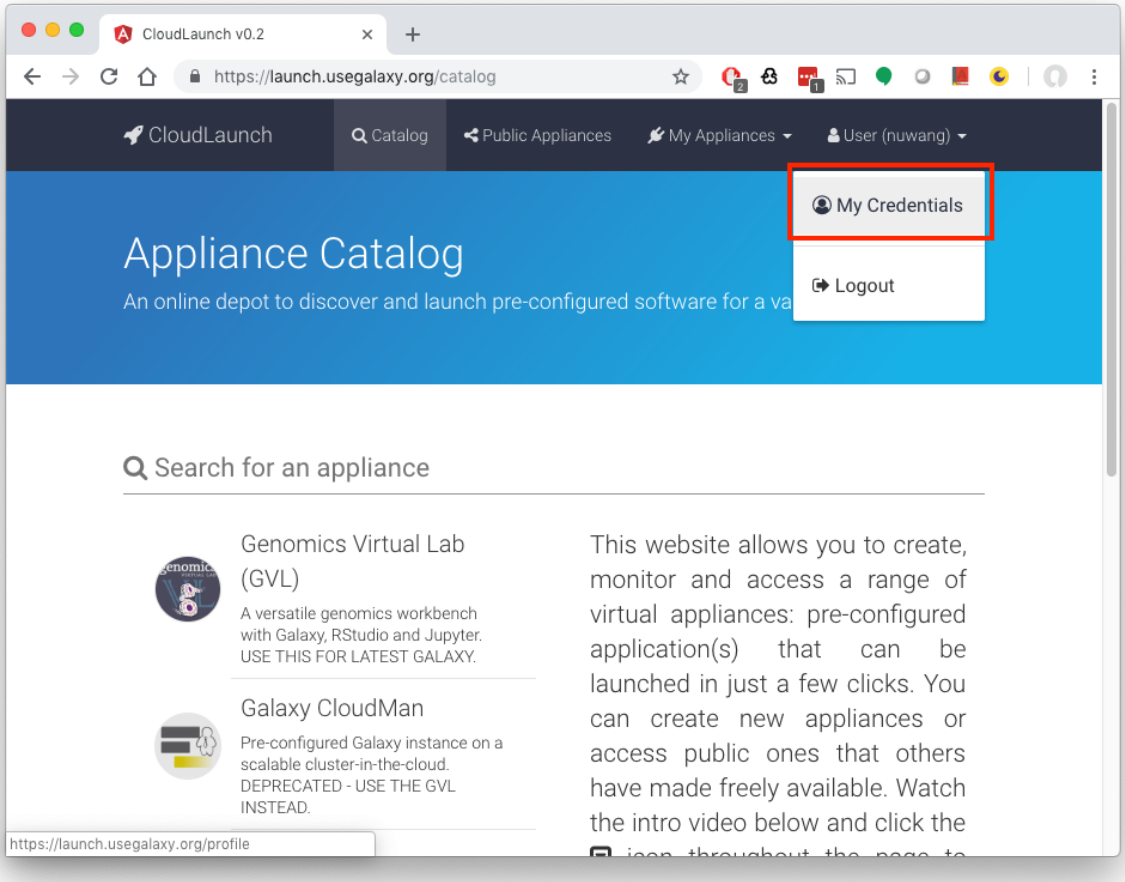
```
<!-- Path for the Pulsar destination config file for path rewrites. -->
<param id="pulsar_file_action_config">config/pulsar_actions.yml</param>
```

In addition, *transfer actions* need to be defined that specify how paths should be translated between the systems. This is done in a dedicated file pointed to in the above `param` tag, in above example `config/pulsar_actions.yml`. A basic example of the file is available below while complete details about the available transfer action options are available as part of the Pulsar documentation.

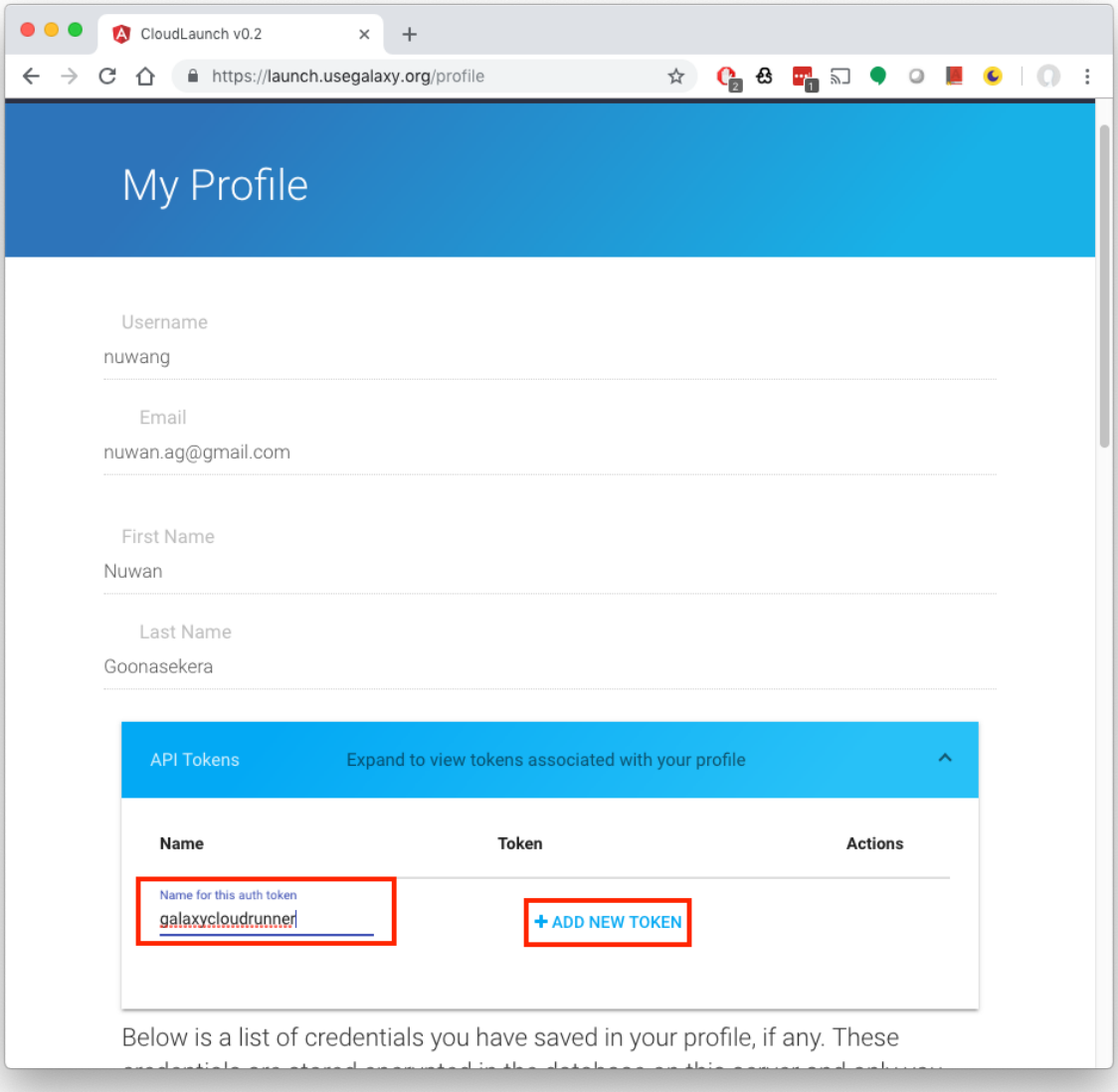
```
paths:
  - path: /galayx/server/tool-data/sacCer2/bwa_mem_index/sacCer2/
    path_types: unstructured
    action: rewrite
    source_directory: /galaxy/server/sacCer2/bwa_mem_index/sacCer2/
    destination_directory: /cvmfs/data.galaxyproject.org/managed/bwa_mem_index/
↪sacCer2/
```

2.2 Obtaining a CloudLaunch API key

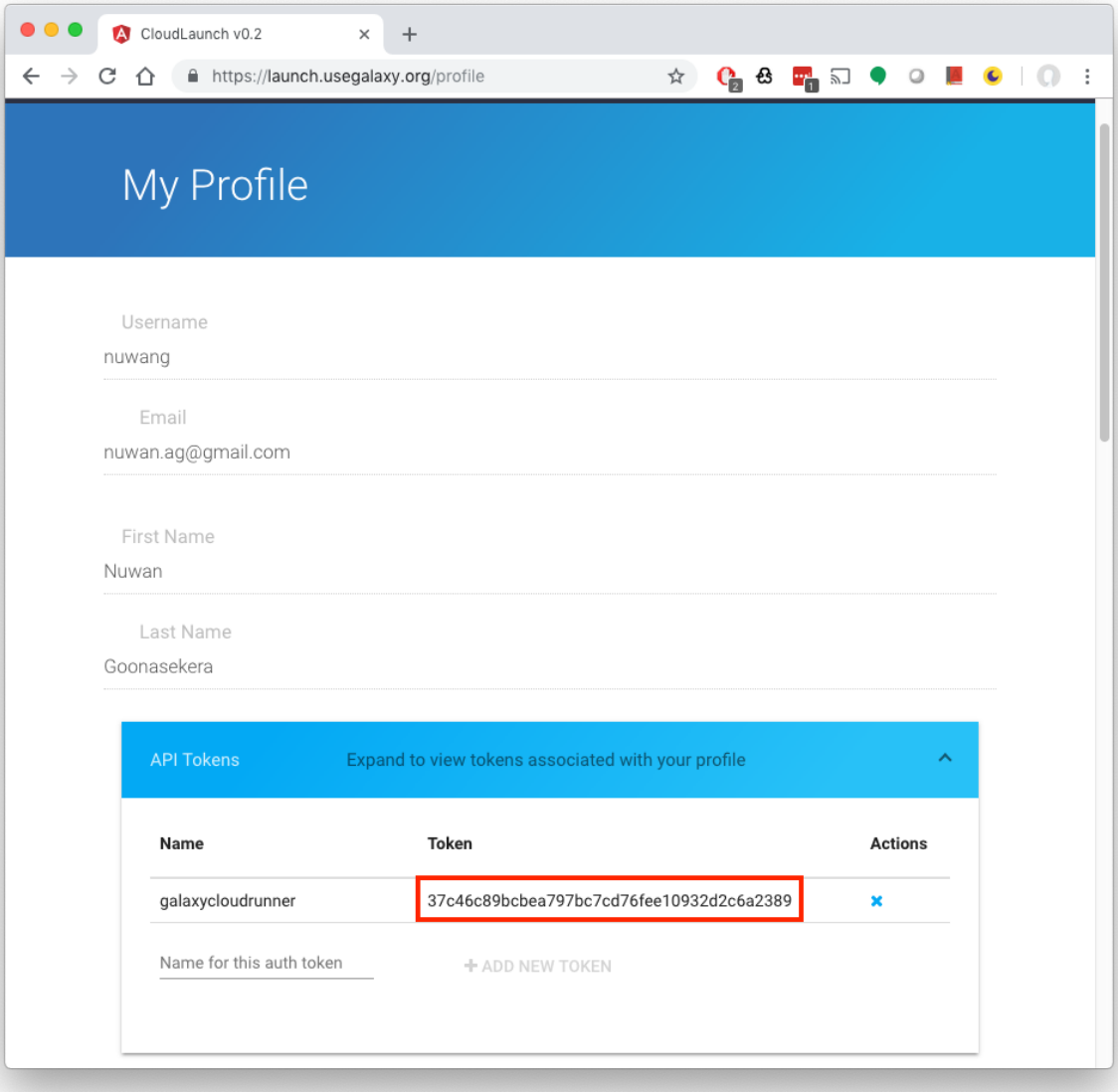
1. Visit the CloudLaunch site: https://launch.usegalaxy.org/
2. Select *Login* on the top menu bar and sign in through a 3rd party provider.
3. Once logged in, select the 'My Profile' option from the menu bar as shown.



4. Get a new API token for CloudLaunch by expanding the collapsed *API Tokens* panel. You can give the API key any name you like (we have given `galaxycloudrunner`) and click the *Add New Token* button.

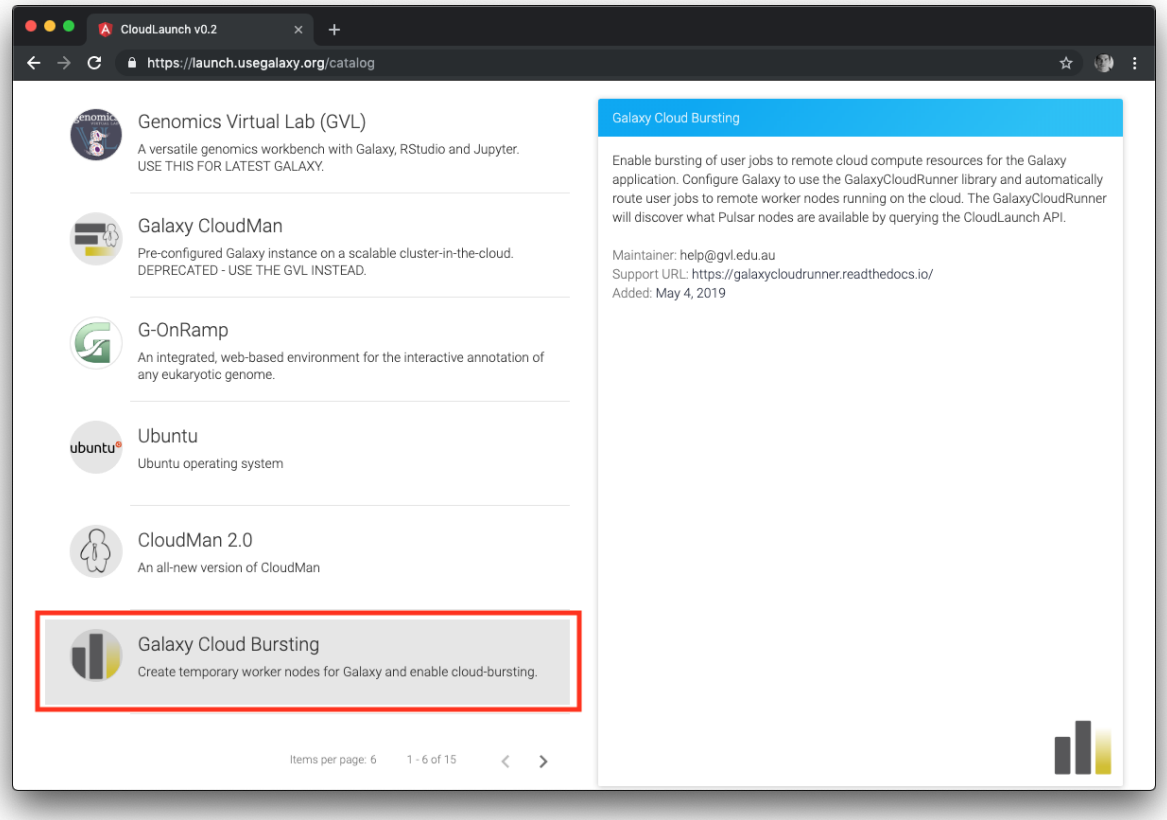


5. Copy the token value and paste it into your `job_conf.xml`.

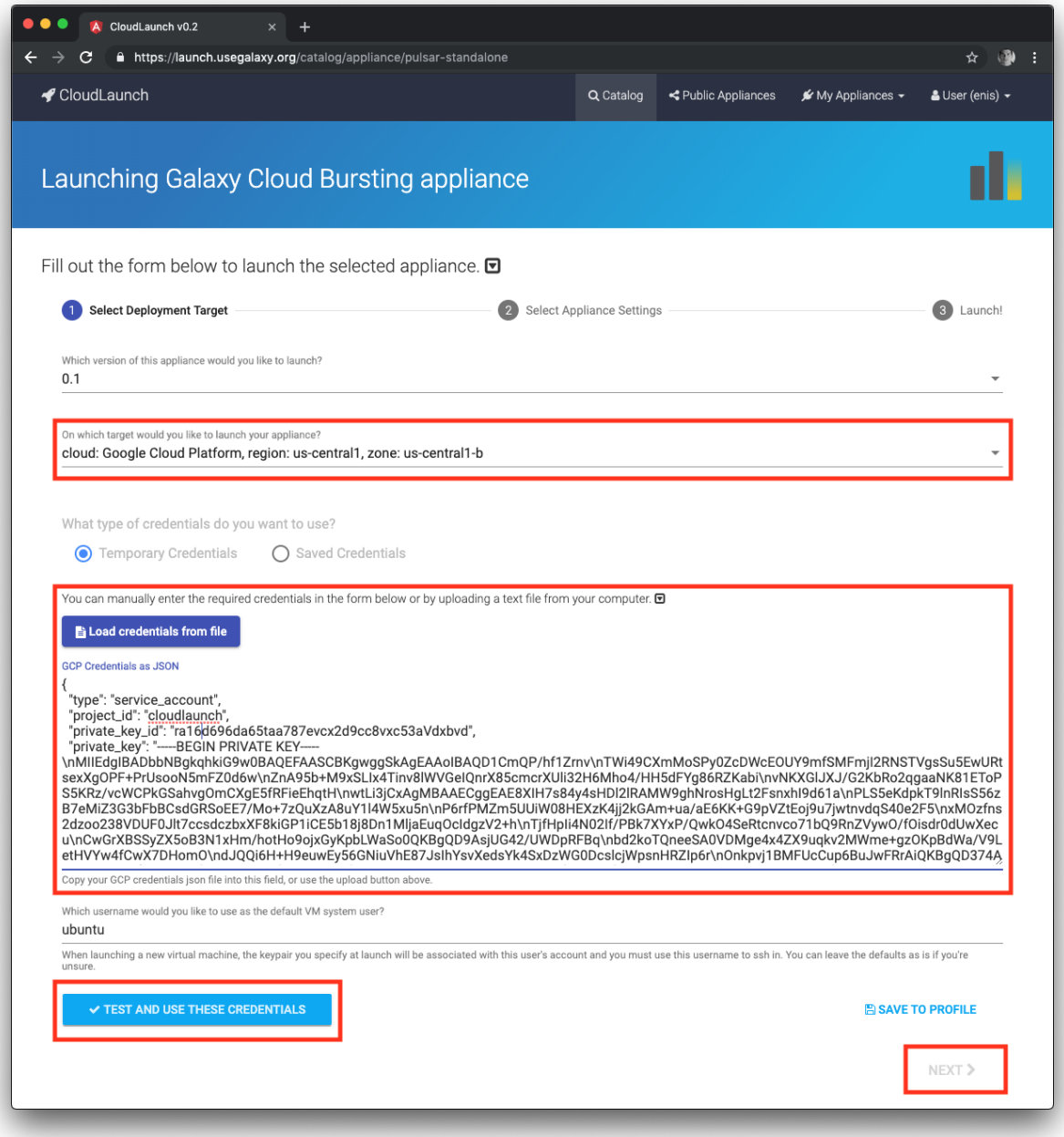


2.3 Adding new worker nodes

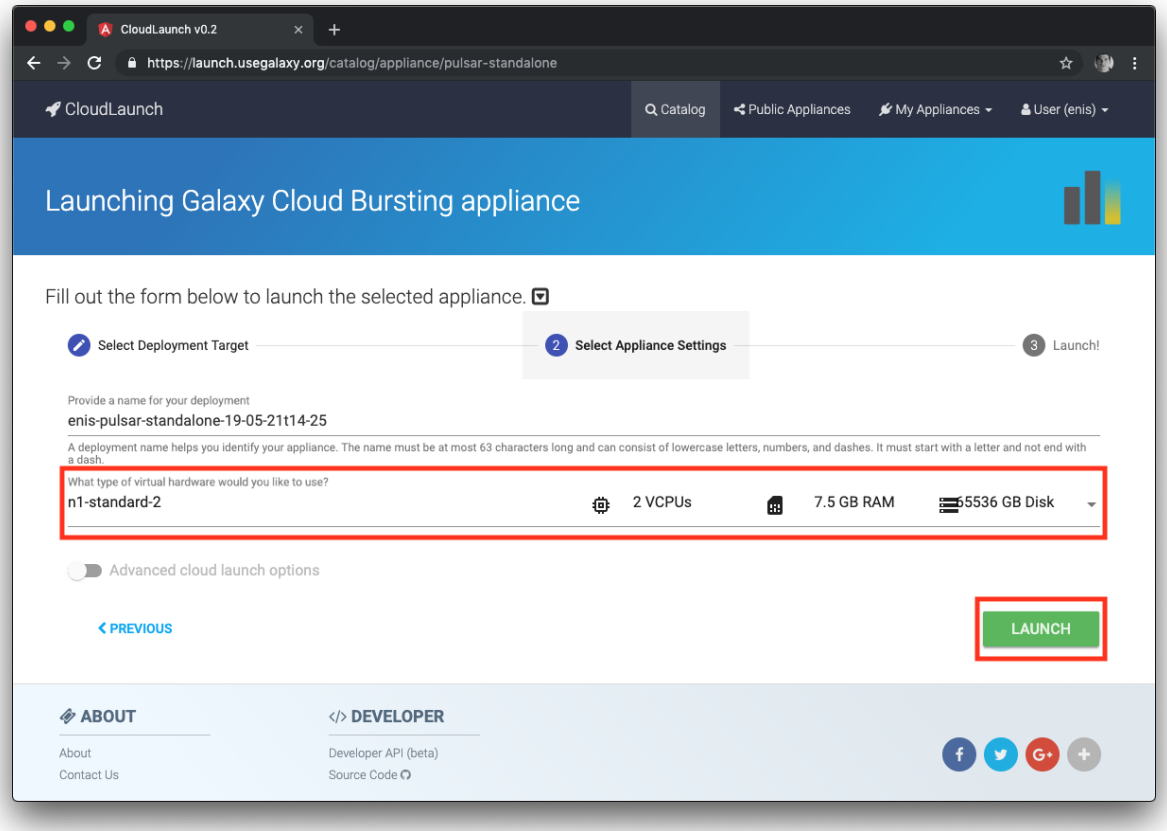
1. To launch a new Pulsar node, go to https://launch.usegalaxy.org/catalog/appliance/pulsar-standalone. We are using the `Galaxy Cloud Bursting` appliance, which is a leveraging the Pulsar application as a remote Galaxy job runner.



3. You may be asked to login through a social network provider.
4. Once logged in, fill in the following fields:
    a. The target cloud you want to launch in
    b. Provide or choose your credentials for the selected cloud
    c. Click the 'Test and use these Credentials button' to validate them
    d. Click next



5. Finally, select the size of the Virtual Machine you want, and click Launch.



6. Simply launching the node is enough, the GalaxyCloudRunner will now pick up your new nodes by querying the CloudLaunch API.

2.4 Job configuration for Galaxy 19.01 or higher

2.4.1 Simple configuration

The following is a simple job configuration sample that you can use to get started.

```
<?xml version="1.0"?>
<job_conf>
    <plugins>
        <plugin id="local" type="runner" load="galaxy.jobs.runners.local:LocalJobRunner" workers="4"/>
        <plugin id="pulsar" type="runner" load="galaxy.jobs.runners.pulsar:PulsarRESTJobRunner"/>
    </plugins>
    <destinations default="galaxycloudrunner">
        <destination id="local" runner="local"/>
        <destination id="galaxycloudrunner" runner="dynamic">
            <param id="type">python</param>
            <param id="function">cloudlaunch_pulsar_burst</param>
            <param id="rules_module">galaxycloudrunner.rules</param>
            <param id="cloudlaunch_api_endpoint">https://launch.usegalaxy.org/cloudlaunch/api/v1</param>
```

(continues on next page)

(continued from previous page)

```
            <!-- Obtain your CloudLaunch token by visiting: https://launch.usegalaxy.org/profile -->
            <param id="cloudlaunch_api_token">37c46c89bcbea797bc7cd76fee10932d2c6a2389</param>
            <!-- id of the PulsarRESTJobRunner plugin. Defaults to "pulsar" -->
            <param id="pulsar_runner_id">pulsar</param>
            <!-- Destination to fallback to if no nodes are available -->
            <param id="fallback_destination_id">local</param>
            <!-- Pick next available server and resubmit if an unknown error occurs -->
            <resubmit condition="unknown_error and attempt &lt;= 3" destination="galaxycloudrunner" />
        </destination>
    </destinations>
    <tools>
        <tool id="upload1" destination="local"/>
    </tools>
</job_conf>
```

In this simple configuration, all jobs are routed to GalaxyCloudRunner by default. This works as follows:

1. If a Pulsar node is available, it will return that node.
2. If multiple Pulsar nodes are available, they will be returned in a round-robin loop.
3. You can add or remove Pulsar nodes at any time. However, there's a caching period (currently 5 minutes) to avoid repeatedly querying the server, which will result in a short period of time before the change is detected by the GalaxyCloudRunner. This has implications for node addition and in particular removal. When adding a node, there could be a delay of a few minutes before the node is picked up. If a Pulsar node is removed, your jobs may be routed to a dead node for the duration of the caching period. Therefore, we recommend attempting a job resubmission through the resubmit tag as shown in the example. See *Additional Configuration and Limitations* on how to change this cache period.
4. If no node is available, it will return the `fallback_destination_id`, if specified, in which case the job will be routed there. If no `fallback_destination_id` is specified, the job will be re-queued till a node becomes available.

2.4.2 To burst or not to burst?

In the above example, all jobs are routed to the GalaxyCloudRunner by default. However, it is often the case that jobs should be routed to the remote cloud nodes only if the local queue is full. To support this scenario, we recommend a configuration like the following.

```
<?xml version="1.0"?>
<job_conf>
    <plugins>
        <plugin id="local" type="runner" load="galaxy.jobs.runners.local:LocalJobRunner" workers="4"/>
        <plugin id="drmaa" type="runner" load="galaxy.jobs.runners.drmaa:DRMAAJobRunner">
        <plugin id="pulsar" type="runner" load="galaxy.jobs.runners.pulsar:PulsarRESTJobRunner"/>
    </plugins>
    <destinations default="burst_if_queued">
        <destination id="local" runner="local"/>
        <destination id="burst_if_queued" runner="dynamic">
```

(continues on next page)

(continued from previous page)

```
            <param id="type">burst</param>
            <param id="from_destination_ids">local,drmaa</param>
            <param id="to_destination_id">galaxycloudrunner</param>
            <param id="num_jobs">2</param>
            <param id="job_states">queued</param>
        </destination>
        <destination id="galaxycloudrunner" runner="dynamic">
            <param id="type">python</param>
            <param id="function">cloudlaunch_pulsar_burst</param>
            <param id="rules_module">galaxycloudrunner.rules</param>
            <param id="cloudlaunch_api_endpoint">https://launch.usegalaxy.org/
↪cloudlaunch/api/v1</param>
            <!-- Obtain your CloudLaunch token by visiting: https://launch.usegalaxy.
↪org/profile -->
            <param id="cloudlaunch_api_token">37c46c89bcbea797bc7cd76fee10932d2c6a2389
↪</param>
            <!-- id of the PulsarRESTJobRunner plugin. Defaults to "pulsar" -->
            <param id="pulsar_runner_id">pulsar</param>
            <!-- Destination to fallback to if no nodes are available -->
            <param id="fallback_destination_id">local</param>
            <!-- Pick next available server and resubmit if an unknown error occurs --
↪>
            <resubmit condition="unknown_error and attempt &lt;= 3" destination=
↪"galaxycloudrunner" />
        </destination>
    </destinations>
    <tools>
        <tool id="upload1" destination="local"/>
    </tools>
</job_conf>
```

Note the emphasized lines. In this example, we route to the built-in rule `burst_if_queued` first, which determines whether or not the cloud bursting should occur. It examines how many jobs in the `from_destination_ids` are in the given state (`queued` in this case), and if there are above `num_jobs`, routes to the to the `to_destination_id` destination (`galaxycloudrunner` in this case). If bursting should not occur, it routes to the first destination in the `from_destination_ids` list. This provides a simple method to scale to Pulsar nodes only if a desired queue has a backlog of jobs. You may need to experiment with these values to find ones that work best for your requirements.

2.4.3 Advanced bursting

In this final example, we show how a complex chain of rules can be used to exert fine-grained control over the job routing process.

```
<?xml version="1.0"?>
<job_conf>
    <plugins>
        <plugin id="local" type="runner" load="galaxy.jobs.runners.
↪local:LocalJobRunner" workers="4"/>
        <plugin id="drmaa" type="runner" load="galaxy.jobs.runners.
↪drmaa:DRMAAJobRunner">
        <plugin id="pulsar" type="runner" load="galaxy.jobs.runners.
↪pulsar:PulsarRESTJobRunner"/>
    </plugins>
    <destinations default="burst_if_queued">
        <destination id="local" runner="local"/>
```

(continues on next page)

(continued from previous page)

```
        <destination id="burst_if_queued" runner="dynamic">
            <param id="type">burst</param>
            <param id="from_destination_ids">local,drmaa</param>
            <param id="to_destination_id">burst_if_size</param>
            <param id="num_jobs">2</param>
            <param id="job_states">queued</param>
        </destination>
        <destination id="burst_if_size" runner="dynamic">
            <param id="type">python</param>
            <param id="function">to_destination_if_size</param>
            <param id="rules_module">galaxycloudrunner.rules</param>
            <param id="max_size">1g</param>
            <param id="to_destination_id">galaxycloudrunner</param>
            <param id="fallback_destination_id">local</param>
        </destination>
        <destination id="galaxycloudrunner" runner="dynamic">
            <param id="type">python</param>
            <param id="function">cloudlaunch_pulsar_burst</param>
            <param id="rules_module">galaxycloudrunner.rules</param>
            <param id="cloudlaunch_api_endpoint">https://launch.usegalaxy.org/
↪cloudlaunch/api/v1</param>
            <!-- Obtain your CloudLaunch token by visiting: https://launch.usegalaxy.
↪org/profile -->
            <param id="cloudlaunch_api_token">37c46c89bcbea797bc7cd76fee10932d2c6a2389
↪</param>
            <!-- id of the PulsarRESTJobRunner plugin. Defaults to "pulsar" -->
            <param id="pulsar_runner_id">pulsar</param>
            <!-- Destination to fallback to if no nodes are available -->
            <param id="fallback_destination_id">local</param>
            <!-- Pick next available server and resubmit if an unknown error occurs --
↪>
            <resubmit condition="unknown_error and attempt &lt;= 3" destination=
↪"galaxycloudrunner" />
        </destination>
    </destinations>
    <tools>
        <tool id="upload1" destination="local"/>
    </tools>
</job_conf>
```

Jobs are first routed to the built-in `burst_if_queued` rule, which determines whether the bursting should occur. If it should, it is then routed to the `burst_if_size` destination, which will check the total size of the input files. If they are less than 1GB, they are routed to the `galaxycloudrRunner` destination. If not, they are routed to a local queue.

2.5 Job configuration for Galaxy versions lower than 19.01

2.5.1 Simple configuration

The following is a simple job configuration sample that you can use to get started.

```
<?xml version="1.0"?>
<job_conf>
    <plugins>
```

(continues on next page)

(continued from previous page)

```
        <plugin id="local" type="runner" load="galaxy.jobs.runners.local:LocalJobRunner" workers="4"/>
        <plugin id="pulsar" type="runner" load="galaxy.jobs.runners.pulsar:PulsarRESTJobRunner"/>
    </plugins>
    <destinations default="galaxycloudrunner">
        <destination id="local" runner="local"/>
        <destination id="galaxycloudrunner" runner="dynamic">
            <param id="type">python</param>
            <param id="function">cloudlaunch_pulsar_burst_compat</param>
            <param id="cloudlaunch_api_endpoint">https://launch.usegalaxy.org/cloudlaunch/api/v1</param>
            <!-- Obtain your CloudLaunch token by visiting: https://launch.usegalaxy.org/profile -->
            <param id="cloudlaunch_api_token">37c46c89bcbea797bc7cd76fee10932d2c6a2389</param>
            <!-- id of the PulsarRESTJobRunner plugin. Defaults to "pulsar" -->
            <param id="pulsar_runner_id">pulsar</param>
            <!-- Destination to fallback to if no nodes are available -->
            <param id="pulsar_fallback_destination_id">local</param>
        </destination>
    </destinations>
    <tools>
        <tool id="upload1" destination="local"/>
    </tools>
</job_conf>
```

In this simple configuration, all jobs are routed to GalaxyCloudRunner by default. This works as follows:

1. If a Pulsar node is available, it will return that node.
2. If multiple Pulsar nodes are available, they will be returned in a round-robin loop.
3. You can add or remove Pulsar nodes at any time. However, there's a caching period (currently 5 minutes) to avoid repeatedly querying the server, that will result in a short period of time before the change is detected by the GalaxyCloudRunner. This has implications for node addition and in particular removal. When adding a node, there could be a delay of a few minutes before the node is picked up. If a Pulsar node is removed, your jobs may be routed to a dead node for the duration of the caching period. Therefore, we recommend a job resubmission through a resubmit tag. However, Galaxy versions prior to 19.01 do not support resubmissions for Pulsar, and you may need to change the cache period to zero to handle this scenario. See *Additional Configuration and Limitations* on how to change this cache period.
4. If no node is available, it will return the `fallback_destination_id`, if specified, in which case the job will be routed there. If no `fallback_destination_id` is specified, the job will be re-queued till a node becomes available.

Note that you must manually add the galaxy rule as described here: *Configuring Galaxy versions lower than 19.01*

2.5.2 To burst or not to burst?

In the above example, all jobs are routed to the GalaxyCloudRunner by default. However, it is often the case that jobs should be routed to the remote cloud nodes only if the local queue is full. To support this scenario, we recommend a configuration like the following.

```
<?xml version="1.0"?>
<job_conf>
```

(continues on next page)

(continued from previous page)

```
    <plugins>
        <plugin id="local" type="runner" load="galaxy.jobs.runners.local:LocalJobRunner" workers="4"/>
        <plugin id="drmaa" type="runner" load="galaxy.jobs.runners.drmaa:DRMAAJobRunner">
        <plugin id="pulsar" type="runner" load="galaxy.jobs.runners.pulsar:PulsarRESTJobRunner"/>
    </plugins>
    <destinations default="galaxycloudrunner">
        <destination id="local" runner="local"/>
        <destination id="galaxycloudrunner" runner="dynamic">
            <param id="type">python</param>
            <param id="function">cloudlaunch_pulsar_burst_compat</param>
            <param id="cloudlaunch_api_endpoint">https://launch.usegalaxy.org/cloudlaunch/api/v1</param>
            <!-- Obtain your CloudLaunch token by visiting: https://launch.usegalaxy.org/profile -->
            <param id="cloudlaunch_api_token">37c46c89bcbea797bc7cd76fee10932d2c6a2389</param>
            <!-- id of the PulsarRESTJobRunner plugin. Defaults to "pulsar" -->
            <param id="pulsar_runner_id">pulsar</param>
            <!-- Destination to fallback to if no nodes are available -->
            <param id="pulsar_fallback_destination_id">local</param>
            <param id="burst_enabled">true</param>
            <param id="burst_from_destination_ids">local,drmaa</param>
            <param id="burst_num_jobs">2</param>
            <param id="burst_job_states">queued</param>
        </destination>
    </destinations>
    <tools>
        <tool id="upload1" destination="local"/>
    </tools>
</job_conf>
```

Galaxy versions prior to 19.01 do not support chaining dynamic rules, and therefore, we have provided a single monolithic rule that can handle both scenarios.

Note the `burst_enabled` flag, which will activate the bursting rule. This rule will determine whether or not the cloud bursting should occur. It examines how many jobs in the `burst_from_destinations` are in the given state (`queued` in this case), and bursts to pulsar only if they are above `burst_num_jobs`. If bursting should not occur, it routes to the first destination in the `from_destinations` list. This provides a simple method to scale to Pulsar nodes only if a desired queue has a backlog of jobs. You may need to experiment with these values to find ones that work best for your requirements.

2.5.3 Advanced bursting

In this final example, we expand this compound rule to also filter jobs by size.

```
<?xml version="1.0"?>
<job_conf>
    <plugins>
        <plugin id="local" type="runner" load="galaxy.jobs.runners.local:LocalJobRunner" workers="4"/>
        <plugin id="drmaa" type="runner" load="galaxy.jobs.runners.drmaa:DRMAAJobRunner">
```

(continues on next page)

(continued from previous page)

```
        <plugin id="pulsar" type="runner" load="galaxy.jobs.runners.pulsar:PulsarRESTJobRunner"/>
    </plugins>
    <destinations default="galaxycloudrunner">
        <destination id="local" runner="local"/>
        <destination id="galaxycloudrunner" runner="dynamic">
            <param id="type">python</param>
            <param id="function">cloudlaunch_pulsar_burst_compat</param>
            <param id="cloudlaunch_api_endpoint">https://launch.usegalaxy.org/cloudlaunch/api/v1</param>
            <!-- Obtain your CloudLaunch token by visiting: https://launch.usegalaxy.org/profile -->
            <param id="cloudlaunch_api_token">37c46c89bcbea797bc7cd76fee10932d2c6a2389</param>
            <!-- id of the PulsarRESTJobRunner plugin. Defaults to "pulsar" -->
            <param id="pulsar_runner_id">pulsar</param>
            <!-- Destination to fallback to if no nodes are available -->
            <param id="pulsar_fallback_destination_id">local</param>
            <param id="burst_enabled">true</param>
            <param id="burst_from_destination_ids">local,drmaa</param>
            <param id="burst_num_jobs">2</param>
            <param id="burst_job_states">queued</param>
            <param id="dest_if_size_enabled">true</param>
            <param id="dest_if_size_max_size">1g</param>
            <param id="dest_if_size_fallback_destination_id">local</param>
        </destination>
    </destinations>
    <tools>
        <tool id="upload1" destination="local"/>
    </tools>
</job_conf>
```

Enable the `dest_if_size_enabled` flag as highlighted to filter by size. This will make sure that the job is routed to Pulsar only if the total size of the input files are less than 1GB. If not, they are routed to `dest_if_size_fallback_destination_id`, which in this case, is a local queue.

2.6 Additional Configuration and Limitations

1. Configuring the query timeout

   You can set the environment variable `CLOUDLAUNCH_QUERY_CACHE_PERIOD` before starting Galaxy to control the caching period (in seconds). Setting this to 0 will allow you to get around the node removal issue where, if a Pulsar node is removed, jobs may be routed to a dead node for the duration of the caching period. However, we recommend setting a value greater than 0 to avoid repeatedly querying a remote server during each job submission.

2. Incompatible tools

   Due to the nature of how Galaxy collects metadata on datasets, certain tools are not compatible with job execution in the bursting mode. Some of these issues will be resolved once Pulsar is upgraded to collect metadata itself but for the time being the following is an (incomplete) list of tools and tool classes that will not operate when executed via the GalaxyCloudRunner: upload tool, data managers, tools that use metadata input, and tools that use custom data discovery.

3. Auto-scaling

Currently, the GalaxyCloudRunner does not support automatic scaling, you must manually add and remove individual nodes but you can add as many as you would like. We will be adding autoscaling features as part of CloudMan v2.0 in future.

4. Galaxy versions prior to 19.01

   Galaxy versions prior to 19.01 do not support certain features required by GalaxyCloudRunner and therefore, need more complex configuration steps.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search