Cloud Resource Federation for Galaxy

Enis Afgan
Galaxy Team
Johns Hopkins University
Jan 23, 2019
Galaxy platform as a science gateway

Galaxy

ToolShed

*NIX tools

BioBlend

Domain users

Training

Admin

Tool developers

Stampede2

Jetstream

CONDA
130,000 registered users
20M jobs run
2PB user data
100 training events (2018 & 2019)

Stats for Galaxy Main (usegalaxy.org) in Dec 2018
usegalaxy.* federation - a group of public Galaxy servers

- Present a similar experience to users no matter which they use
- Guarantee a minimum service
  - Tools & versions
  - Reference Data
  - Reproducibility
  - Training materials
- Starting with USA, Europe and Australia, more welcome!
- Manage with community assets/repositories
- Don’t prescribe hardware resources
125+ platforms for using Galaxy

Public servers
Academic and commercial clouds
Container images
Virtual Machines
Galaxy is well-adopted by a broad community
Scaling challenges: quotas

- 3-4 small jobs
- &
- 2 parallel jobs

Storage: 250GB
Scaling challenges: **silos and fragmentation**

Which Galaxy has the data / tool / workflow I need?

- **Galaxy Main**
  - useGalaxy.org
  - Galaxy cluster + Jetstream + Stampede2

- **Private**
  - Galaxy Server
  - localhost:8080
  - Laptop → cluster

- **Public**
  - Galaxy Servers
  - www
  - Private infrastructure

- **Galaxy on the Cloud**
  - Galaxy
  - on the
  - Cloud
  - launch.usegalaxy.org
  - AWS | Azure | OpenStack

- **SlipStream**
  - Galaxy Appliance
  - www.bioteam.net
  - Purchased server

Manual process
Each server is custom-crafted and centrally administered.
Galaxy-as-a-Service

useGalaxy.*

Galaxy without Quotas!
Galaxy-as-a-Service: towards a federated Galaxy

GaaS core components

- Compute
- Storage
- AuthNZ
Compute: attach compute resources to a session
Storage: allow a user to link to object stores
Auth: handle user identity and resource ownership

- Rely on identity that can span Galaxy instances
- Remove, and at least minimize, storing user cloud credentials
- Be compatible with a variety of resource providers

A tool suite for cloud virtual environments: **CloudVE**

- CloudAuthz
- CloudLaunch
- CloudBridge
- CloudMan
- HelmsMan

[http://cloudve.org](http://cloudve.org)
Today: a closer look at compute bursting
Enabling cloud bursting for life sciences within Galaxy

Enis Afgan¹,²,c,†, Nate Coraor³, John Chilton³, Dannon Baker¹, James Taylor¹ and The Galaxy Team

¹Department of Biology, Johns Hopkins University, Baltimore, MD, USA
²Centre for Informatics and Computing, Rudjer Boskovic Institute (RBI), Zagreb, Croatia
³Department of Biochemistry and Molecular Biology, Penn State University, University Park, PA, USA

SUMMARY

Fueled by the radically increased capacity to generate data of research, the need for data storage and analysis platform for life science research has been constrained by the ability to analyze data. Galaxy, a data sharing and analysis platform for life science research, has been developed to address this need. However, the scale of data and the scope of tools required has made any monolithic deployment of the Galaxy application unpopular. We share our approach to utilizing compute and storage resources is necessary to avoid deploying and scaling in creating a ubiquitous platform capable of simultaneously utilizing remote resources. Specifically, the requirements, process, and impact of this solution are described in detail. Copyright © 2010 John Wiley & Sons, Ltd.

Received 18 January 2015; Revised 23 March 2015; Accepted 14 April 2015

KEY WORDS: cloud computing; cloud bursting; data analysis; performance; design; experimentation
2019: GalaxyCloudRunner

- Enables bursting of user jobs to remote compute resources for the Galaxy application
- Integrated with Galaxy 19.01 release but also applicable to older releases
- Enables bursting per Galaxy instance
- Documentation available at galaxycloudrunner.readthedocs.io
GalaxyCloudRunner usage

1. Install galaxycloudrunner Python library into your Galaxy’s virtual environment
2. Add a job rule to Galaxy which will determine the Pulsar node to route to
3. Configure your job_conf.xml to use this rule
4. Launch as many Pulsar nodes as you need through CloudLaunch
5. Submit your jobs as usual
What is Pulsar?

- Python server application
- Allows a Galaxy server to run jobs on a remote system
- No shared file system required
- Configurable
- Securable
- Can submit jobs to HPC queueing system
- Automatically handles tool dependency management

https://pulsar.readthedocs.io/
How Pulsar works

1. User clicks “Execute”
2. Galaxy packs up and sends:
   ○ Data
   ○ Config files
   ○ Tool name & version
   ○ Parameters and other job metadata
3. Pulsar accepts the job
4. Pulsar checks if tool is installed locally
   ○ If not - Installs tool with Conda or Docker
5. Pulsar submits job to local queue
6. Pulsar waits until job complete
7. Pulsar packs up result and sends it back to Galaxy
What is CloudLaunch?

A gateway for discovering and launching applications on a variety of clouds.

**Cloud-agnostic**
Backed by CloudBridge, use native cloud capabilities for infrastructure management

**Pluggable and extensible**
Arbitrary launch process and UI are supported, via an isolated plug-in mechanism

**UI and REST API**
UI available for end-users but it is all API driven for integration into external apps

Try it at [https://launch.usegalaxy.org/](https://launch.usegalaxy.org/)

Why CloudLaunch?

AWS Marketplace

GCE Solutions

Azure Marketplace

Jetstream Atmosphere VMs

Consistent interface
Single, uniform API
Multi-cloud
Shared global data via CVMFS

**Stratum 0**: The canonical source
- Transactional updates

**Stratum 1**: Multiple servers
- Mirrors Stratum 0 server
- Continuous updates

**User servers**: Many multiple servers
- Mounts repo from stratum 1
- Based on GEO-API
- With fallback to other stratum 1s
Configuring Galaxy

Make use of *dynamic destinations* to define `galaxycloudrunner` as the default destination

```xml
<?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>
      <!-- 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>
```
Support for opportunistic bursting

Route jobs to the remote cloud nodes only if the local queue is full.

GalaxyCloudRunner is extensible so can add your own rules.

In addition, can burst based on input file size.
Galaxy cloud bursting in a picture

1. One-time setup

2. Launch cloud nodes as desired

3. GalaxyCloudRunner checks availability

4. Submit jobs as normal

job_conf.xml

<destination>
  ...
</destination>
Looking forward and beyond Galaxy
(Auto)-scaling, via CloudMan

Currently, each cloud node is a single, independent resource. Scale can be achieved by adding multiple nodes.

Submit and manage jobs via Pulsar API

Control and manage resources in response to settings or load

Provision compute and storage infrastructure (resources from IaaS cloud providers)

Actions

Run application jobs submitted by end users via Galaxy

Configure resources into cluster nodes, to run Slurm, Kubernetes, etc.

Give me a VM, disk, etc.

https://github.com/galaxyproject/cloudman/tree/v2.0
Beyond Galaxy use cases

- **CloudBridge** is a general-purpose, multi-cloud library for interacting with the IaaS resources

- **CloudLaunch** leverages CloudBridge and can launch a variety of applications; each appliance is a plugin with custom back-end and front-end components

- **CloudMan** is a cloud manager for orchestrating a running cloud deployment, primarily focusing on managing Kubernetes clusters for multiple clouds

- **HelmsMan** is a manager for Helm applications, currently integrated with CloudMan