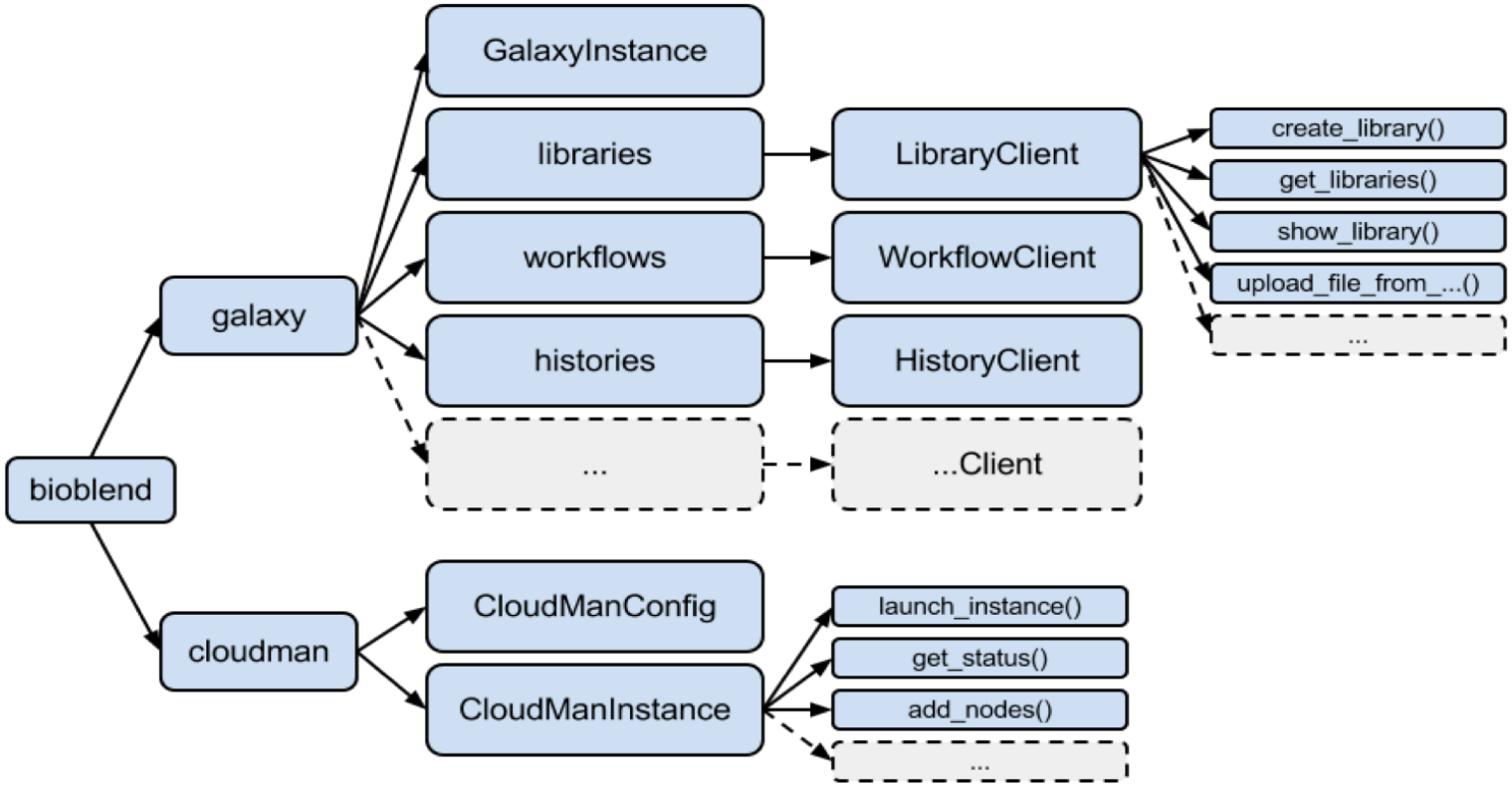
BioBlend - Enabling Pipeline Dreams

Clare Sloggett[1], Nuwan Goonasekera[1,2], Enis Afgan[1,3]

[1]VLSCI, University of Melbourne
[2]VERSI, University of Melbourne
[3]CIR, Ruđer Bošković Institute (RBI)

BOSC 2013, Berlin

BioBlend is a Python library which wraps the Galaxy API and the CloudMan API (both REST)



Goal is to enable creation of automated and scalable pipelines.

Galaxy | Analyze Data | Workflow | Shared Data | Visualization | Cloud | Help | User | Using 1%

Tools

search tools

Get Data
Send Data
ENCODE Tools
Lift-Over
Text Manipulation
Convert Formats
FASTA manipulation
Filter and Sort
Join, Subtract and Group
Extract Features
Fetch Sequences
Fetch Alignments
Get Genomic Scores
Operate on Genomic Intervals
Statistics
Graph/Display Data
Regional Variation
Multiple regression
Multivariate Analysis
Evolution
Motif Tools
Multiple Alignments
Metagenomic analyses
Genome Diversity
Phenotype Association
EMBOSS

NGS TOOLBOX BETA
NGS: QC and manipulation

Tools

Sharing

Scalable

Cuffdiff (versio

Transcripts:

5: http://galaxy-vic.genome.edu.au/datasets/870093c60d62e4ae,

A transcript GFF3 or GTF file produced by cufflinks, cuffcompare, or other source.

Perform replicate analysis:

No

Perform cuffdiff with replicates in each group.

Jobs

SAM or BAM file of aligned RNA-Seq reads:

9: https://galaxy-vic.genome.edu.au/datasets/a6aaf8a0aa187613

tracking_id	class_code	nearest_ref_id	gene_id	gene_short_name	tss_id	locus
FBtr0089063	-	-	FBgn0016126	CaMKI	TSS14551	chr4:466119-474566
FBtr0089064	-	-	FBgn0016126	CaMKI	TSS12579	chr4:466119-474566
FBtr0089065	-	-	FBgn0016126	CaMKI	TSS8216	chr4:466119-474566
FBtr0089066	-	-	FBgn0016126	CaMKI	TSS17320	chr4:466119-474566
FBtr0089067	-	-	FBgn0016126	CaMKI	TSS4313	chr4:466119-474566
FBtr0089068	-	-	FBgn0016126	CaMKI	TSS4313	[illegible]
FBtr0089069	-	-	FBgn0016126	CaMKI	TSS16559	[illegible]
FBtr0089070	-	-	FBgn0004607	zfh2	TSS5731	[illegible]
FBtr0089075	-	-	FBgn0040324	Ephrin	TSS11650	[illegible]
FBtr0089076	-	-	FBgn0040324	Ephrin	TSS11650	chr4:592382-598159
FBtr0089077	-	-	FBgn0040324	Ephrin	TSS11650	chr4:592382-598159
FBtr0089078	-	-	FBgn0040324	Ephrin	TSS11650	chr4:592382-598159
FBtr0089079	-	-	FBgn0039911	CG1909	TSS6869	chr4:603736-609976
FBtr0089080	-	-	FBgn0039911	CG1909	TSS14820	chr4:603736-609976
FBtr0089082	-	-	FBgn0025936	Eph	TSS8588	chr4:631311-641552
FBtr0089083	-	-	FBgn0025936	Eph	TSS8588	chr4:631311-641552

Data

Published items | jeremy | Trackster Demo 2: chr19 891,949 - 2,358,938 Import visualization

1,000,000 2,000,000

GM12878 Cufflinks assembled transcripts BEST

h1-hESC Cufflinks assembled transcripts BEST

Visualization

History

Random samples

1.1 GB

26: Cuffdiff on data 9, data 9, and data 5: transcript FPKM tracking

246 lines

format: tabular, database: dm3

cuffdiff v2.1.1 (4046M) cuffdiff --no-update-check -q --library-norm-method geometric --dispersion-method pooled -p 8 -c 10 --FDR 0.050000

/galaxy/main_pool/pool5/files/004/3

005/1

Provenance

1	2	3
tracking_id	class_code	nearest_ref_id
FBtr0089063	-	-
FBtr0089064	-	-
FBtr0089065	-	-
FBtr0089066	-	-
FBtr0089067	-	-

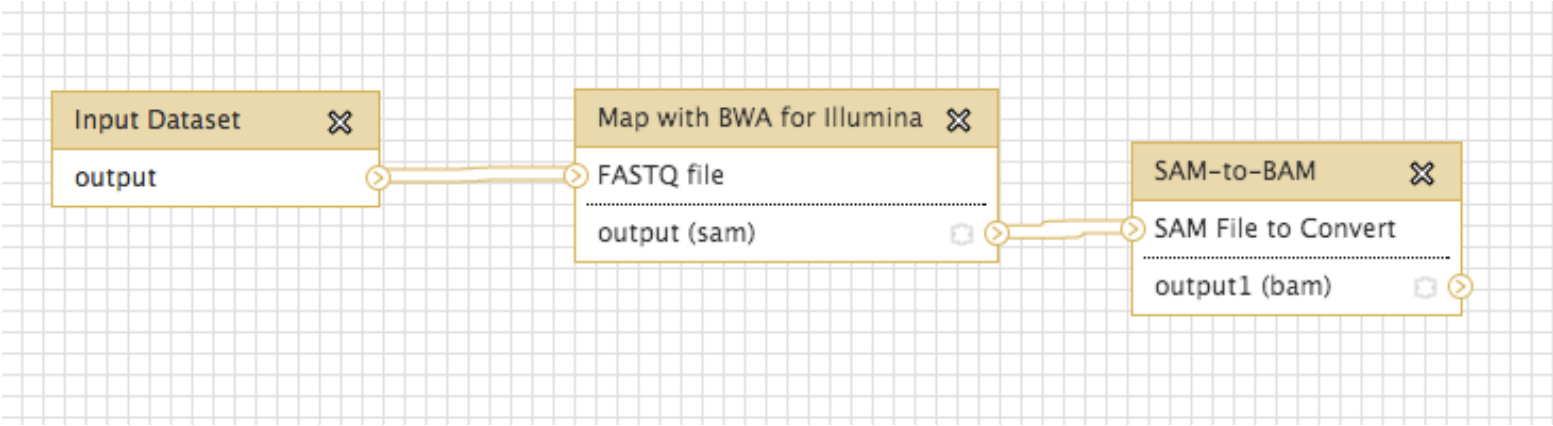
25: Cuffdiff on data 9, data 9, and data 5: transcript differential expression testing

24: Cuffdiff on data 9, data 9, and data 5: gene FPKM tracking

23: Cuffdiff on data 9, data 9, and data 5: gene differential expression testing

http://usegalaxy.org

Simple: single-end mapping to reference



Sample1.fastq → Sample1.bam

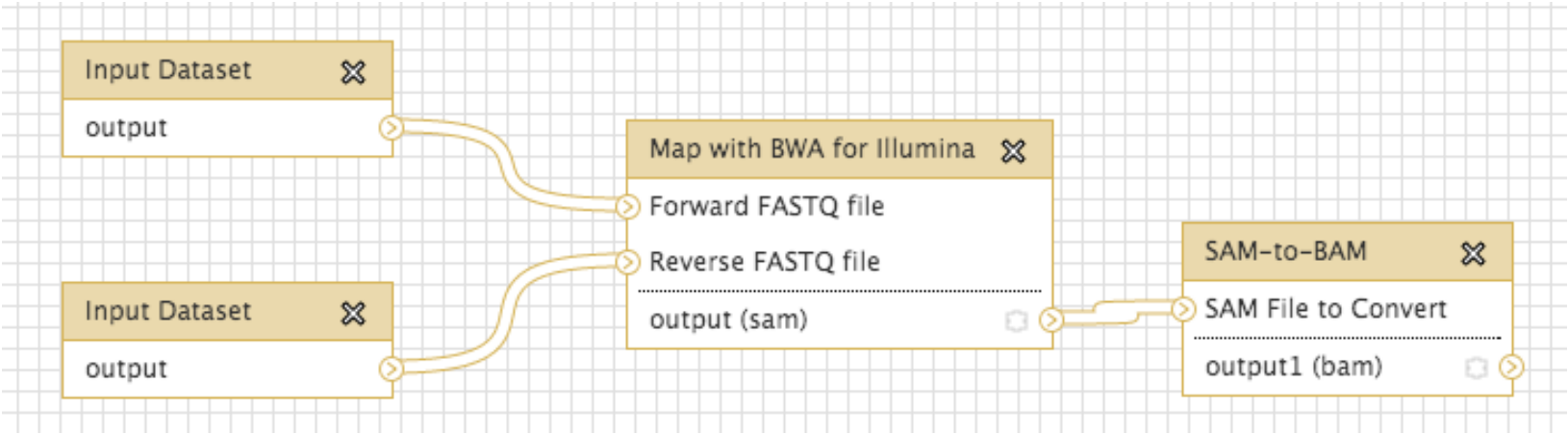
Sample2.fastq → Sample2.bam

Sample3.fastq → Sample3.bam

For some workflows the GUI batch mode isn't enough, because we need to know which file is which;

we need to base the logic on metadata fields like sample, patient, or experimental run.

Less simple: paired-end mapping to reference



Sample1_R1.fastq
Sample1_R2.fastq
Sample1.bam

Sample2_R1.fastq
Sample2_R2.fastq
Sample2.bam

Galaxy REST API

http://galaxy-dist.readthedocs.org/
http://bitbucket.org/galaxy/galaxy-dist/src

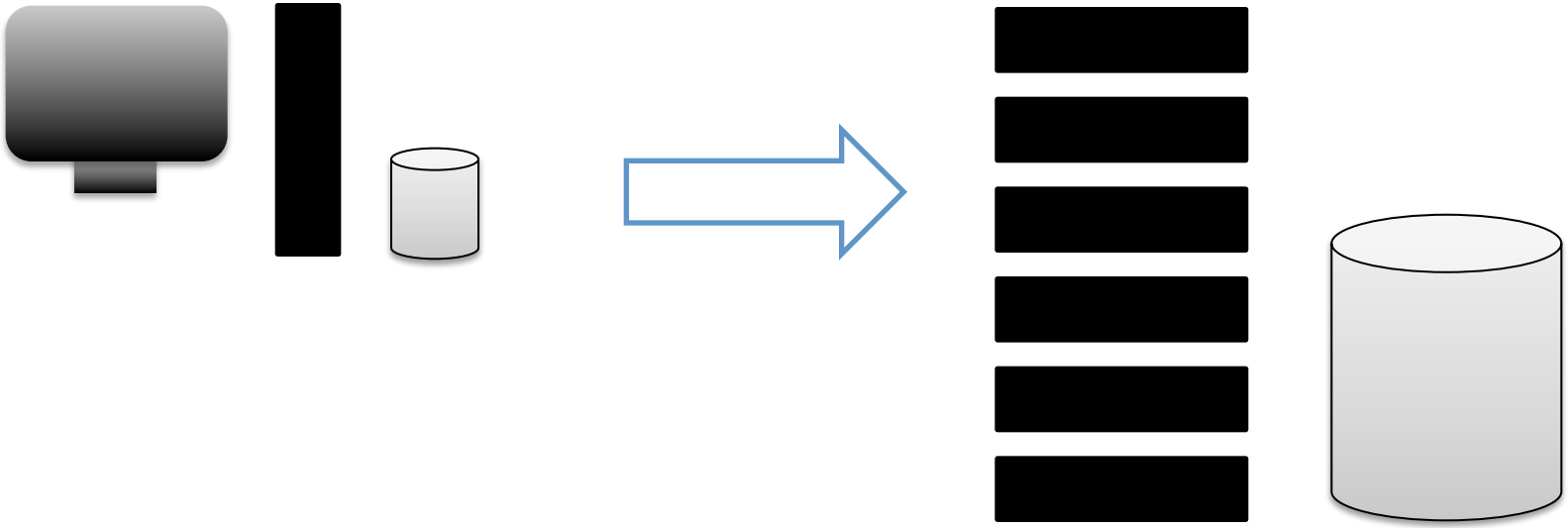
Admin and end-user functionality, e.g.

- manage users, quotas, roles
- manage Data Libraries
- import and run Workflows
- upload and download data; automate analysis as soon as data is available

Galaxy REST API

https://main.g2.bx.psu.edu/api/histories/0a7b7992a7cabaec?key=123456789abcdef1011

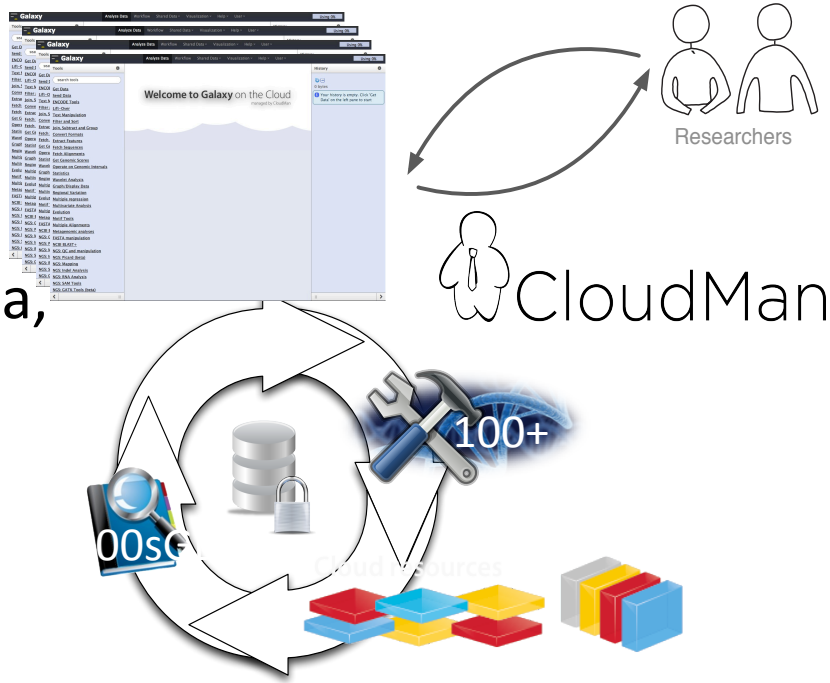

```
{
    "annotation": "",
    "contents_url": "/api/histories/0a7b7992a7cabaec/contents",
    "id": "0a7b7992a7cabaec",
    "name": "New output history",
    "nice_size": "1.4 MB",
    "state": "ok",
    "state_details": {
        "discarded": 0,
        "empty": 0,
        "error": 0,
        "failed_metadata": 0,
        "new": 0,
        "ok": 8,
        "paused": 0,
        "queued": 0,
        "running": 0,
        "setting_metadata": 0,
        "upload": 0
    },
```



CloudMan

CloudMan is a platform for running a "virtual cluster" in the cloud

- SGE cluster, attached storage
- Built with CloudBioLinux
- Galaxy installed and configured
- Tools and genomes pre-installed



Runs on AWS (Amazon), OpenNebula, Eucalyptus, OpenStack (Australian Research Cloud)

http://usecloudman.org/

CloudMan from Galaxy

Admin | Report bugs | Wiki | Screencast

CloudMan Console

Welcome to CloudMan. This application allows you to manage this instance cloud cluster and the services provided within. Your previous data store has been reconnected. Once the cluster has initialized, use the controls below to manage services provided by the application.

Terminate cluster | Add nodes ▾ | Remove nodes | Access Galaxy

Status

Cluster name: galaxy-dev-vic

Disk status: 0 / 0 (0%)

Worker status: **Idle**: 0 **Available**: 0 **Requested**: 0

Service status: Applications ● Data ●

Autoscaling is off.
Turn on?

Cluster status log

05:54:31 - Master starting
05:54:33 - Completed the initial cluster startup process. Configuring a previously existing cluster of type SGE
05:54:37 - SGE service prerequisites OK; starting the service
05:54:44 - Setting up SGE...
05:56:58 - post_start_script found and saved to '/mnt/cm/post_start_script'; running it now (note that this may take a while)
05:57:27 - Done running post_start_script
05:57:27 - All cluster services started; the cluster is ready for use

CloudMan from Galaxy

Admin | Report bugs | Wiki | Screencast

CloudMan Admin Console

This admin panel is a convenient way to gain insight into the status of individual CloudMan services as well as to control those services.
Services should not be manipulated unless absolutely necessary. Please keep in mind that the actions performed by these service-control 'buttons' are basic in that they assume things will operate as expected. In other words, minimal special case handling for recovering services exists. Also note that clicking on a service action button will initiate the action; there is no additional confirmation required.

Galaxy controls

Use these controls to administer functionality of Galaxy.

- Access Galaxy
- Current Galaxy admins: system@genome.edu.au
- Add Galaxy admin users What will this do?

  CSV list of emails to be added as admins | Add admin users
- Running Galaxy at revision: 10003:b4a373d86c51
- Update Galaxy from a provided repository What will this do?

  http://bitbucket.org/galaxy/galaxy-dist | Update Galaxy

Services controls

Use these controls to administer individual application services managed by CloudMan. Currently running a 'Galaxy' type of cluster.

Service name	Status						
Galaxy	Running	Log	Stop	Start	Restart	Update DB	
PostgreSQL	Running	Log	Stop	Start	Restart		
SGE	Running	Log	Stop	Start	Restart	Q conf	qstat
Galaxy Reports		Log	Stop	Start	Restart		

File systems

Name	Status	Usage	Controls	
transient_nfs	Running			Details
galaxy	Running			Details
galaxyIndices	Running			Details

+ Add new

System controls

Use these controls to administer CloudMan itself as well as the underlying system.

- We have Galaxy workflows as an execution engine
- We have CloudMan as an infrastructure manager

e.g. REST call:

```
http://main.g2.bx.psu.edu/api/histories/
  0a7b7992a7cabaec?key=123456789abcdef10
```

Returns JSON:

```
{ 'id': '0a7b7992a7cabaec',
  'name': 'Output history',
  'state_details': { 'discarded': 0,
                     'empty': 0,  ......
```

in BioBlend becomes Python:

```
GalaxyInstance.histories.show_history('0a7b7992a7cabaec')
```

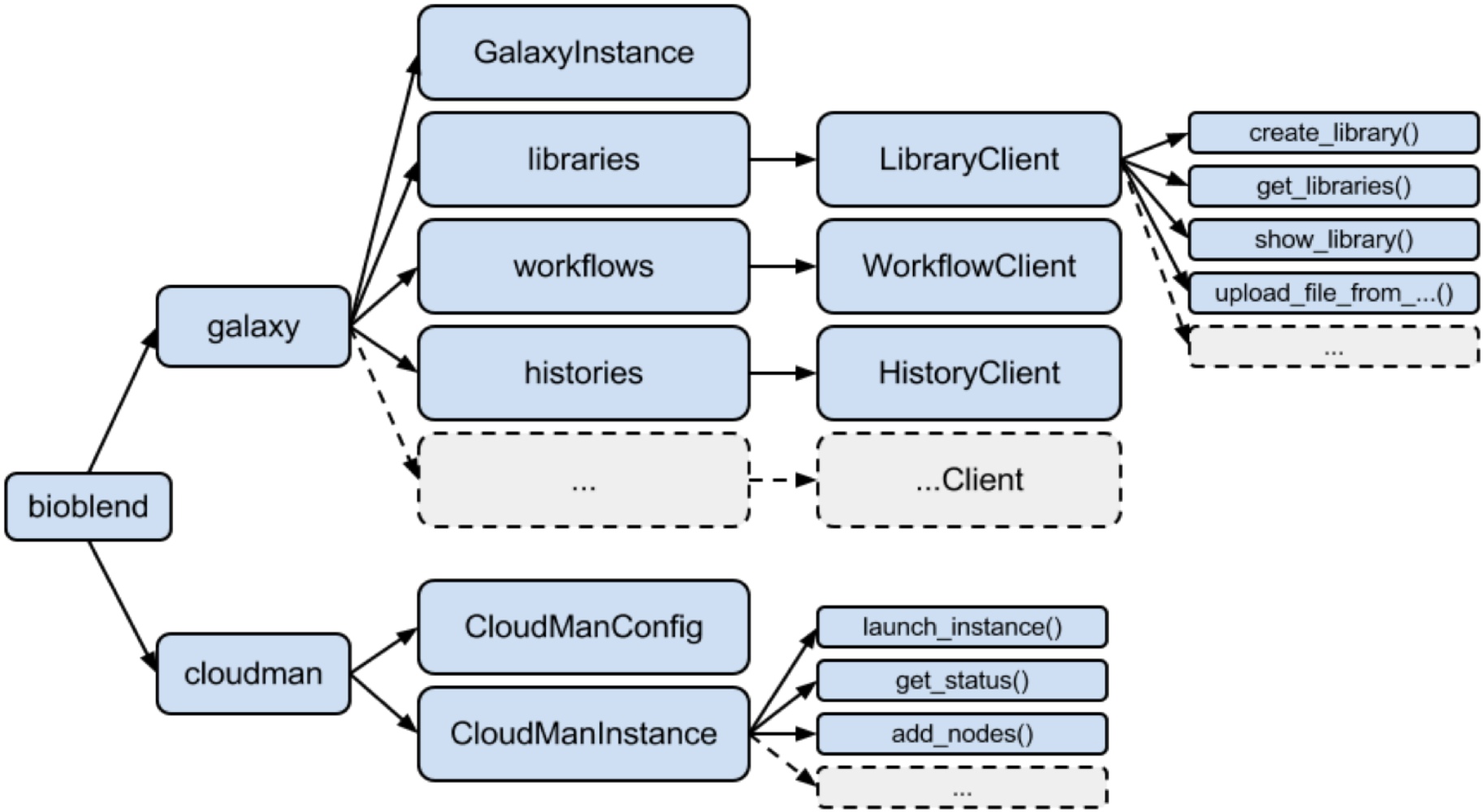
Returns corresponding Python dictionary

```
{ 'id': '0a7b7992a7cabaec',
  'name': 'Output history',
  'state_details': { 'discarded': 0,
                     'empty': 0,  ......
```

Reduces boilerplate code

```
class HistoryClient(Client):
    def __init__(self, galaxy_instance):
        self.module = 'histories'
        super(HistoryClient, self).__init__(galaxy_instance)

    def show_history(self, history_id, contents=False):
        """
        Get details of a given history. By default, just get the
        history meta information. If ``contents`` is set to ``True``,
        get the complete list of datasets in the given history.
        """
        return Client._get(self, id=history_id, contents=contents)
```


bioblend
galaxy
cloudman
GalaxyInstance
libraries
workflows
histories
...
LibraryClient
WorkflowClient
HistoryClient
...Client
create_library()
get_libraries()
show_library()
upload_file_from_...()
...
CloudManConfig
CloudManInstance
launch_instance()
get_status()
add_nodes()
...

```
cloud = Bunch(id='-1',
              name="NeCTAR",
              cloud_type='openstack',
              bucket_default='cloudman-os',
              region_name='NeCTAR',
              region_endpoint='nova.rc.nectar.org.au',
              ec2_port=8773,
              ec2_conn_path='/services/Cloud',
              cidr_range='115.146.92.0/22',
              is_secure=True,
              s3_host='swift.rc.nectar.org.au',
              s3_port=8888,
              s3_conn_path='/')
  # Create an instance of the CloudManConfig class and launch a CloudMan instance
  cmc = CloudManConfig(ak, sk, name,  ami, inst_type, pwd, cloud_metadata=cloud,
    cloudman_type=cm_type, initial_storage_size=2, placement='melbourne-np')
  print "Configured an instance; waiting to launch and boot..."
  cmi = CloudManInstance.launch_instance(cmc)
  print "Done! CloudMan IP is {0}".format(cmi.cloudman_url)
```

bioblend/docs/examples/run_imported_workflow.py

```
1  """
2  This example demonstrates running a tophat+cufflinks workflow over paired-end da
3  This is a task we could not do using Galaxy's GUI batch mode, because the inputs
4  The workflow is imported from a json file (previously exported from Galaxy), and
5
6  This example creates a new Data Library, so you must be a Galaxy Admin on the in
7
8  Also note that a Galaxy Workflow will only run without modification if it finds
9  installed on the Galaxy instance. This is to ensure reproducibility.
10 In this case we expect Tophat wrapper 1.5.0 and Cufflinks wrapper 0.0.5.
11
12 Usage: python run_imported_workflow.py <galaxy-url> <galaxy-API-key>
13 """
14
15 import sys
16 from bioblend import galaxy
17
18 ## ------------------------------------
19 ## Config information for this example
20
21 # Specify workflow and data to import into Galaxy
22
23 workflow_file = 'tophat_cufflinks_pairedend_workflow.ga'
24
25 import_file_pairs = [
26     ('https://bioblend.s3.amazonaws.com/C1_R1_1.chr4.fq', 'https://bioblend.s3.a
27     ('https://bioblend.s3.amazonaws.com/C1_R2_1.chr4.fq', 'https://bioblend.s3.a
28     ('https://bioblend.s3.amazonaws.com/C1_R3_1.chr4.fq', 'https://bioblend.s3.a
29 ]
```

http://bioblend.readthedocs.org/

bioblend.readthedocs.org/en/latest/

BioBlend 0.2.3-dev documentation »

Project Versions

latest

RTD Search

Go

Full-text doc search.

Table Of Contents

Next topic

API documentation for interacti

This Page

Show Source
Show on GitHub
Edit on GitHub

BioBlend

About

BioBlend is a Python (2.6 or higher) library for interacting with CloudMan and Galaxy's API.

Conceptually, it makes it possible to script and automate the process of cloud infrastructure p via Galaxy. In reality, it makes it possible to do things like this:

- Create a CloudMan compute cluster, via an API and directly from your local machine:

```
from bioblend.cloudman import CloudManConfig
from bioblend.cloudman import CloudManInstance
cfg = CloudManConfig('<your cloud access key>', '<your cloud secret key>
cmi = CloudManInstance.launch_instance(cfg)
cmi.get_status()
```

- Reconnect to an existing CloudMan instance and manipulate it:

```
from bioblend.cloudman import CloudManInstance
cmi = CloudManInstance("<instance IP>", "<password>")
cmi.add_nodes(3)
cluster_status = cmi.get_status()
cmi.remove_nodes(2)
```

- Interact with Galaxy via a straightforward API:

```
from bioblend.galaxy import GalaxyInstance
gi = GalaxyInstance('<Galaxy IP>', key='your API key')
libs = gi.libraries.get_libraries()
gi.workflows.show_workflow('workflow ID')
gi.workflows.run_workflow('workflow ID', input_dataset_map)
```

50 bioblend/galaxy/toolshed/__init__.py View file @ 70066e9

```
@@ -0,0 +1,50 @@
+"""
+Interaction with Galaxy Tool shed
+
+"""
+from bioblend.galaxy.client import Client
+from os.path import basename
+
+
+class ToolShedClient(Client):
+
+    def __init__(self, galaxy_instance):
+        self.module = 'tool_shed_repositories'
+        super(ToolShedClient, self).__init__(galaxy_instance)
+
+    def get_tools(self):
+        """
+        Get a list of all tools in galaxy tool shed repository
+
+        :rtype: list
+        :return: Returns a list of dictionaries containing information about tools present in the tool shed repositori
+                 For example::
+                 [{u'changeset_revision': u'4afe13ac23b6',
+                 u'deleted': False,
+                 u'dist_to_shed': False,
+                 u'error_message': u'',
+                 u'name': u'velvet_toolsuite',
+                 u'owner': u'edward-kirton',
+                 u'status': u'Installed'}]
+
+        """
+        return Client._get(self)
+
```

Contribute!

```
+        return Client._get(self)
```

Basically 1 line of code!

Thanks Philip Mabon!

http://bioblend.readthedocs.org/

https://github.com/afgane/bioblend

https://pypi.python.org/pypi/bioblend

blend4j: https://github.com/jmchilton/blend4j

clj-blend: https://github.com/chapmanb/clj-blend

Thanks to…

genomics VIRTUAL LAB

nectar

vlsci LIFE SCIENCES COMPUTATION

RB

VeRSI

Galaxy