Galaxy LIMS documentation

Introduction

Galaxy offers by default a Sample Tracking system. In the NGS LIMS project we decided to use and extend this system.

Administrator Manual

Quick start

This is a summary to get a populated version of the Galaxy LIMS

Summary

- Check Dependencies
- hg clone https://bitbucket.org/jelle/galaxy-central-tron-lims galaxy-central-tron-lims
- cd galaxy-central-tron-lims && ./run.sh --daemon && tail -f paster.log
- Check correct running interface on http://localhost:8080 and inspect the paster.log file
- python scripts/lims/add_defaults.py universe_wsgi.ini demo
- Login using you@company.none/demolims

Once you have the system running, you can now start creating requests and adding samples by following this short introduction Create and manage requests
To test the automatic flowcell processing, go to Watch and analyze sequence run directories

About

The code provided by the TRON repository is the modified version of galaxy-central. We tend to regulary merge the code repositories to keep up2date with all the latest goodies. Although our modifications have been aimed at the Sample Tracking part, you might see some differences throughout our instance compared to the base - those should be minor though.
The documentation below explains and shows how to setup the TRON Galaxy LIMS - assumed is you are somewhat familiar with Galaxy.

Dependencies

The following python modules are required.

- pyyaml
- PIL **with zlib(PNG/ZIP) support**
- xhtml2pdf
- qrcode

In addition you may also want to use virtualenv as also proposed by the Galaxy 'Production Server' Manual.

One way to install the above mentioned modules is by using easy_install. In order for easy_install to build these modules you might also need the **python-dev** package (e.g. apt-get install python-dev).

To get easy_install:

When running python 2.6:

```
wget 'http:
//pypi.python.org/packages/2.6/s/setuptools/setuptools-0.6c11-py2.6.egg#md5=bfa92100bd772d5a213eedd35
sh setuptools-0.6c11-py2.6.egg
```

or for 2.7

```
wget 'https:
//pypi.python.org/packages/2.7/s/setuptools/setuptools-0.6c11-py2.7.egg#md5=fe1f997bc722265116870bc79
sh setuptools-0.6c11-py2.7.egg
```

And then:

```
easy_install pyyaml
easy_install PIL #check for zlib *-see note below*
easy_install xhtml2pdf
easy_install qrcode
```

Make sure PIL builds with zlib support. You might need to check that libz.so is located in /usr/lib see also: easy_install PIL does not install zlib

If PIL fails to install with zlib support, you will first have to remove the package!
Removing python packages can be made easy with pip e.g. easy_install pip && pip uninstall PIL

You can now continue following the Quick start or continue reading

Get the code and unpack

```
hg clone https://bitbucket.org/jelle/galaxy-central-tron-lims galaxy-central-tron-lims
```

You can now continue following the Quick start or with running galaxy, e.g. ./run.sh or ./run.sh --daemon && tail -f paster.log.

universe_wsgi.ini

If you are not running the 'demo' settings, then there are a few specific settings in the ini file that you will have to change to optimally use this Galaxy instance.

For testing purposes, you can leave all default settings.

Main

```
# TRON START sample_tracking MAIN
# Use same method for adding default forms as B. Chapman did for nglims extension.
lims_config_file = tool-data/lims.yaml
# Sequencing request library - you will have to create this library through galaxies 'manage data
libraries' interface
# This library will contain all sequencing requests
sequencing_library = Sequencing Projects
# Flowcell library will contain all info about your flowcells. You want to keep those accesible by
admins only.
flowcell_library = Flowcell Data
# which user account is used for the api calls and workflow selection
ngs_bot_user_email = ngs-galaxy@company.none
# which user account(s) will be the lims administrators
lims_admin_users = you@company.none, lab-geek@company.none
# Leave following settings alone - e.g. alternatives are not implemented/tested!!
use_extended_sample_form = True
# are illumina indexes mutable when designing the flowcell - Not functioning
flowcell_mutable_indexes = False
# TRON END
```

The comments should be pretty explanatory. The lims_config_file specifies where the yaml file is located, which describes the default form definitions, request types, default roles, users and requests. The given location is where you can find the current yaml file.
As in the comment, the sequencing_library is the "Data Library" that will hold the sequencing requests data by default - e.g. when a new request is created a child folder will be created owned by and named after the request owner. Another folder below that folder is created which is named after the request name. The name of this folder will change together with the request name.

Add default forms and settings

Start Galaxy

Before adding the default forms, you will have to start Galaxy once and close/shut it down after you have been able to reach the Galaxy interface - as that tells you that your settings were correct and the database has been initialized. In this stage, you can setup your Galaxy system following the instruction provided on the Galaxy Wiki. If you are just testing, you can directly continue our documentation.

add_defaults.py

Summary

```
python scripts/lims/add_defaults.py universe_wsgi.ini demo
```

Intro

After you have shutdown Galaxy you are now able to add the forms that are optimized for TRONs sample tracking. You can change and/or add forms in before loading them through this step or afterwards in the web interface. As mentioned before, the yaml file describes the locations of the form definitions and which forms will be used for a request type.

Different from the standard Galaxy sample tracking is that a request type also relates to a flowcell form, which is only used for the default available illumina indexes - i.e. [ADVANCED] if you want to add more default indexes, add or change a flowcell form. Every field with the keyword 'index' in its name will be used to build the multiplexing index list. The first letter of the name will be prefixed to the index number(e.g. for Sure Select Indexes : S01 ATCACG. This behavior might change as it is a quick fix.

The yaml file as set in the universe_wsgi.ini by the option lims_config_file points by default to three different form definitions. One "Sample Form", describes all fields presented on the add/edit sample pages, a "Request Form", describes the fields needed in the "New/Edit Request" page and a "Flowcell Form", which describes how many lanes the flowcell design will have and the by default available illumina indexes.
The forms are defined in CSV's files which are then parsed by the add_defaults.py script, they follow practically the same format as used in the csv import function for importing forms that can be found in the web interface - however they might differ over time.

Demo

To get you started quickly you can add some more defaults settings like users, roles, libraries and demo requests. You can configure parts of this demo by modifying the mentioned yaml file.
You can add more users, define their default permission roles, create more libraries and add more requests.
When executing:

```
python scripts/lims/add_defaults.py universe_wsgi.ini demo
```

The following (default) objects will be created for the purpose of demoing:

- **User-Types**:
  - Internal
  - External
- **Users** (passwords: demolims)
  - Customer1: customer-1@demo.none
  - Admins, also named in the universe_wsgi.ini
    - You-Admin: you@company.none
  - Lims-admins, also named in the universe_wsgi.ini
    - Lab-Admin: lab-geek@company.none
  - NGS Bot that can be used to upload sequence request data
    - ngs-bot: ngs-galaxy@company.none
- **Roles**:
  - NGS Customer
    - Default members: Customer1, You-Admin, Lab-Admin
  - Employee
    - Default members: You-Admin, Lab-Admin, ngs-bot
  - NGS Admin
    - Default members: You-Admin, Lab-Admin, ngs-bot
- **Libraries**:
  - Sequencing Projects
    - Accessible by the roles: NGS Customer, NGS Admin
  - Flowcell Data
    - Accessible by the roles: NGS Admin
- **Forms**:
  - Sample
    - All the sample information fields
  - Flowcell
    - Containing available multiplexing indexes
  - Request
    - Describing the minimal info about the request
  - User forms:
    - Internal
    - External
      - Containing more fields like Address, Account number etc
- **Request types**:
  - NGS Request

        - Consists of the forms: Sample, Flowcell & Request
- **Requests**:
    - Demo1
        - Owned by Customer1
    - Demo2
        - Owned by Customer1

Execute

To add the default forms:

```
python scripts/lims/add_defaults.py universe_wsgi.ini demo
```

Or without all the default accounts and roles

```
python scripts/lims/add_defaults.py universe_wsgi.ini
```

This script will output the specific changes it made. If all worked well, you can now start your Galaxy instance and see the default forms under "Manage form definitions".

Administration

Admin
The next steps requires a Galaxy admin account

Intro

If you did not execute the add_defaults.py with the demo option, you still have to create user accounts, roles and libraries yourself. Below is explained how to do so.
Note You can also update the settings in the yaml file(tool-data/lims.yaml) and run the scripts/lims/add_defaults.py script to add your specific wishes.
Running the script with the update command like: "python scripts/lims/add_defaults.py universe_wsgi.ini demo **update**" certain entities will be updated. Always **backup** your database before doing so as the changes are hard to reverse otherwise!

Summary
*Create Customer role e.g. "NGS_customer"
*Create Lab admin role e.g. "NGS_admin"
*Add role "Customer" to those users who need access to the "Lab"
*Add roles "Customer" and "Lab admin" to "Request Type" e.g. "NGS Request"
*Create two data libraries with the name specified in ini file with the parameter "sequencing_library" and "flowcell_library"
*Add "Customer" role to the access permission of the "sequencing_library" library
*Add "Lab admin" role to the access permission of the "flowcell_library" library
*Add "ngs_bot_user" account

Roles and permissions

This step is not necessary but ensures only specific user have access to the "Sequencing Request" part of your Galaxy instance.
You can create users as admin or let users create their own accounts. After an account is created you can add roles to these accounts. You can create any role and name it as you like, for example "NGS_customer". You can then add this role to those users who will be allowed to send "Sequencing Request".
To allow a user to submit a "Sequencing Request" with the role "NGS_customer" you will have to add this role to the request type. Under "Manage request types" are the available request types and when following this guide, you will have the "NGS Request" request type. Click on "Edit permissions" and add the "NGS_customer" role. Now your users with this role have access to the "Lab".

Add data library

In the universe_wsgi.ini file you have specified a name for the "sequencing_library" and "flowcell_library", you will now have to create these libraries under "manage data library". You can limit the "access" to these libraries to those who only have the correct roles. After creating the libraries, change the access role to the "NGS_customer" role for the "sequecing_library".
Now as someone creates a new request, a folder with his specified e-mail address is created in this library and a folder below with the sequence request name. This folder is then linked to this request and should be used to send around files, e.g. Gel photos, raw and analyzed data.
When a flowcell is created, a folder is created under the "flowcell_library" library. Now an admin can "Add data" to this folder, data can be anything, like used samplesheets or QC forms.

Add ngs_bot_user

To be able to select workflows for samples and automatically run them on sequencing completion create the user that you indicated in the universe_wsgi.ini under the option ngs_bot_user_email.

Watch and analyze sequence run directories

Here we describe how you can setup your LIMS to automatically wait for new flowcell data and process them through CASAVA, upload fastq files to the request owners library and execute a given workflow set-up in Galaxy.
What we describe here is directly applicable to the HiSeq2000 sequencing process, you will have to modify the methods and scripts if you have a different sequencer or 'workflow'.

Demo data workflow

Demo
Unfortunately we do not provide any raw Illumina sequencing data, so if you don't have your own a complete data flow is not possible.
However, setting the option "demo" to True (default) in the api_scripts.ini, does show most of the processes that would otherwise be started.

```
• cd scripts/api; python watchscript_flowcell.py
```

Non-Demo data workflow

Summary

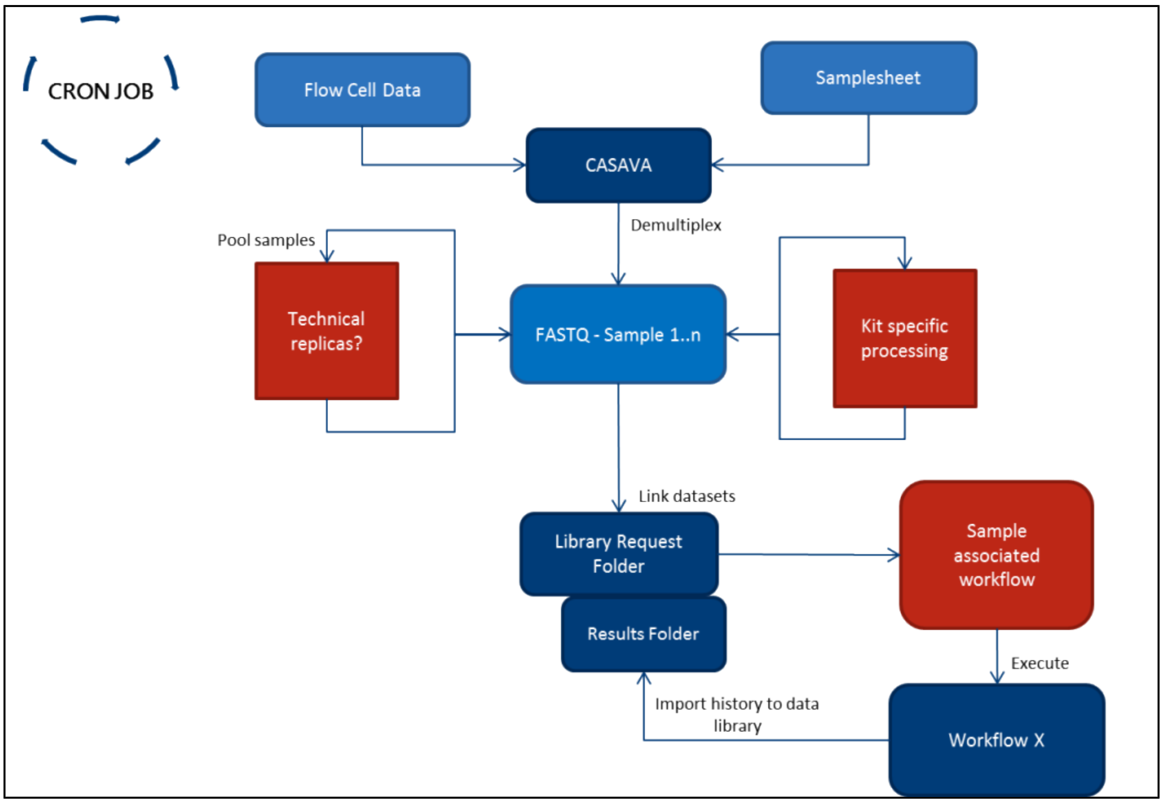
- Using the ngs-bot user, create one or more workflows that take a fastq file as input
- Add samples/requests and as lims admin specify a workflow for each sample
- Create a flowcell
- Edit scripts/api/api_scripts.ini - provide the api key of the ngs-bot user
- Setup cronjob like: "0 * * * * galaxy cd $GALAXYHOME/scripts/api; python watchscript_flowcell.py"
- Output sequencing data into the 'sequencing_output_path' with the flowcell identifier at the end of the directory name
- See the result in the request owner library

Sequence processing dependencies

- CASAVA1.8
- Illumina Sequencing data

How it works

The script watchscript_flowcell.py will check for any new flowcells in the 'sequencing_output_path' directory and will note any new flowcell directory in the file specified by 'processed_flowcells'. If the file RTAComplete.txt is found, this will tell the script that this flowcell is ready to be processed by CASAVA. A samplesheet will be fetched from Galaxy, using the script 'scripts/api/get_samplesheet.py', and CASAVA is started with this samplesheet to create fastq files, which are placed in 'casava_output_path'. Then each fastq will be linked to the request library folder in galaxy - using the samplesheet to identify the request identifier. If any workflow was set, the fastq will be processed using that workflow under the ngs-bot account/history. All non-hidden files will then be copied to the request library folder and permissions are set to the request owner.
Currently the workflows to be executed and that can be selected when modifying requests will be linked to the ngs-bot user. To understand better how the processing of flowcells works, the best way is to look in the two scripts: scripts/api/watchscript_flowcell.py and scripts/api/process_flowcell.py



Customization

As you may have noticed there are some default TRON logos and texts in place. You may replace those with your own. Please be aware of the picture sizes and formats.

Submission form:

The submission form is defined in:

- templates/requests/common/submission_form.mako
  - static/images/submission_logo.png # shown in the middle-top of the submission form, size: 308x75px
  - static/images/QR_logo_60.png # shown within the qr barcode, size: 60x60px
    Feel free to modify this template. As you will notice, most information is like an html file. Be aware though, that not all css styling will be converted/displayed in the resulting pdf. Changing this template does not require you to restart Galaxy.

Welcome.mako

The welcome.mako replaces the default welcome.html. You can modify this file to change the first page as you enter Galaxy.

- templates/welcome.mako
  - static/images/Logo.png # The logo shown in the bottom-center of the page, size: 300x300px

Job section
If you don't like the job section on the welcome page, you can remove the blocks between the ##JOBS comments (in the static/welcome.mako)
Furthermore, feel free to remove static/jobs.xml

Wiki

summary

- Check Requirements
- Configure Galaxy to send notifications (Configuration)
- Set-up rabbitmq
- Configure 'listener.py' to send to your wiki setup, or set both to false to send to messages to STDOUT

Intro

To let a wiki be updated automatically on each request change, you will have to configure an amqp server, as used by the Galaxy team, we

have been using RabbitMQ.
The scripts attached have been successfully run in our environment to update a Atlassian Confluence (3.x) wiki. Furthermore, we have been using this script to update a mediawiki with semantics support (using the SMW bundle).

Make sure you have all dependencies/requirements in place:

Requirements

- RabbitMQ-Server set-up -> apt-get install rabbitmq-server
- For reading messages from AMQP:
  - pika 0.5.2 -> pip install pika==0.5.2
- For Mediawiki:
  - python-wikitools -> http://code.google.com/p/python-wikitools/
  - Pywikipediabot -> http://www.mediawiki.org/wiki/Manual:Pywikipediabot
- For Confluence:
  - xmlrpclib -> easy_install xmlrpclib
- Additional modules:
  - libxml2 -> apt-get install libxml2-dev
  - libxslt -> apt-get install libxslt1-dev
  - lxml -> easy_install lxml

Configuration

You will have to setup galaxy to send request&flowcells updates to an amqp channel:

Set-up rabbitmq

For demo purposes the default configuration can be used for rabbitmq. (re)start galaxy on the same host and you can leave the default settings in the universe_wsgi.ini and in the script mentioned for updating a wiki. Just remove the comment marks and **set notify_wiki to true**
.

Galaxy

```
# TRON START
# Wiki AMQP settings - if you want to send sequencing request updates to a wiki. Confluence and
Mediawiki supported.
notify_wiki = True
notify_amqp_host = localhost:5672
notify_amqp_user = guest
notify_amqp_pass = guest
notify_amqp_virtual_host= /
notify_amqp_exchange = galaxy
notify_amqp_type = direct
notify_amqp_routing_key = sequencing_requests
notify_amqp_queue = requests
# TRON END
```

Script

Under scripts/lims you will find a script called listener.py
You will need to set the correct settings in this script.

STDOUT: If you do not have a wiki or if you are interesting only in seeing the messages, set both confluence and wiki to False in **listener.py**. This way the messages will only be send to STDOUT.

This script can be executed and put in the background or in a screen to always run and wait for new messages on the queue.

Error handling: if you get an error like: "TypeError: channel() takes at least 2 arguments (1 given)" then you are probably running a newer version of pika. Either make sure you are running pika 0.5.2 or update the code to support your version.

If you do not like how pages appear in your wiki, you will need to edit galaxy2wiki.py directly.

Troubleshooting

Fetching eggs fails

If you get the error message "Fetch failed." after running ./run.sh you might have to correct your http_proxy settings if you are behind a proxy.
To see if your python instance can access the internet try:

```
$: python
>>> from urllib import urlopen
>>> print urlopen("http://www.google.com").read()
```

And correct based on any error message you get.

User manual

Create and manage requests

This quick 'step-by-step' guide you through the basics of creating and managing requests. You will have to have Galaxy LIMS installed to try this out. We assume you can access your instance through http://localhost:8080 and you added the default values by executing 'python scripts/lims/add_defaults.py universe_wsgi.ini demo'
The 'customer' account has the email address customer-1@demo.none and the lab account, the one managing the request, the email address lab-geek@company.none

Create a request

1. Browse to the main page: http://localhost:8080
2. Login with the customer account (default password: demolims) using the "User->Login" option
3. From the "Analyse Data" view, "Go to the Lab" section ("Lab->Sequencing Requests")
You will now see 2 requests, they are however without any samples.
4. Create a new request by pressing the right upper button "Create new request" or edit one of the existing
5. After providing at the minimal a name to the request, you can now after saving add some samples

Adding samples

1. On the "Add Samples" page, you start out with one sample.
2. The best way to list your samples is to first provide information for this first sample.
3. Then, at the bottom of the page specify how many copies of the sample you want to create and press "Add sample".
Specifying 4 samples will get you a total of 5 samples in this request, each with the same information. Change according to the information you want to provide
4. Press "Save" to store the request. You will be warned if some values are missing or bad depending on the sample type you specified.

Uploading data

If you have any (quality) information about your samples in a file (e.g. pictures), you can upload those with your request for the Lab personnel to check
1. Go to "Associate Data"
2. Pick a file to upload from your local drive and provide some description if you want
3. "Upload to library"
4. The resulting file will show below in the section "Associated datasets", where you can also see the other files belonging to your request.

Submit request

1. Press "Submit request" button in the top right corner
2. After succesful submission you can now get the submission form, that you can sign and sent with your samples to the lab

ADMIN: Request overview

1. Login with the lab account (default password: demolims) using the "User->Login" option
2. You will now see the section "Lab Admin" in the header and below that the "Seqeuncing requests" option, go there.
Here you will see multiple requests that have been created by you or other users

ADMIN: Library preparation information

1. Select a request (that has one or more samples)
2. Go to the "QC details" tab
3. Here you are able to provide your information regarding quality control and library preparation
4. For each section, you are able to select if the customer is allowed to see this information when he/she is looking at the request
5. Furthermore, each 'QC line' can be copied to add more information about one sample

ADMIN: Uploading data

Apart from directly interacting with the data library under "Shared Data" you can, like the 'customer' upload data to the request directly, this can be a quality control file or the raw or analysed data.
1. Go to the "Associate Data" tab
2. Pick a file to upload from your local drive and provide some description if you want
3. Check the checkbox "Show Customer" if you want to allow the customer to see this dataset after upload
4. "Upload to library"
5. The resulting file will show below in the section "Associated datasets". It will list all files 'associated' with this request and show if the request owner is able to see the file.

ADMIN: Submit request

The admin can reject, submit or delete request without any form validation

ADMIN: Create a flowcell

1. If there are **submitted** requests one can start to design a flowcell
2. Go to "Create flowcell" by using the left panel or through the "Lab Admin->Browse Flowcells->Create flowcell" buttons
3. "Find" samples, by default samples that are passed the "Sequencing Finished" state (after a flowcell is finished) will not be found
4. Select your samples and "Create flowcell"
5. Either by creating or editing a flowcell you can now rearrange your samples on the flowcell
– You will be warned if samples cannot fit in one lane due to muliplexing problems
6. If you want to define technical replicates, you can press the two-blue-circles on that sample when its in the bucket to clone it. When using the 'processed_flowcell.py' script technical replicate will automatically be pooled (based on the same name and index).
7. "Save"

ADMIN: Manage flowcell

1. Once defined you can get a samplesheet for this flowcell
2. When not using the "watchscript_flowcell.py", you can update your flowcell manually
3. In the "Browse Flowcells" section, you can select any flowcell and set the state to "Submit"(flowcell cannot be editted anymore) and "Finished".
4. If put to "Finished" all samples on that flowcell will be updated to the "Finished state", furthermore, if all samples of a request are finished, the request will also be marked.

ADMIN: Associate data to flowcell

1. If you have any data you want to associate to the flowcell, you can use the "Add data" when selecting a flowcell
2. Data associated, can also be viewed by browsing the "Shared Data->Data Libraries->flowcell_library"