
Tripal Galaxy Documentation

Release 7.x-1.x

Connot Wytko, Shawna Spoor, Brian Soto, Stephen Ficklin

Mar 29, 2022

Tripal Galaxy Exetension Module

1	Install Tripal Galaxy	3
1.1	Installation	3
1.2	Set Permissions	4
2	Option 1: Automatic Webform Creation	7
2.1	Site-wide Files	7
2.2	Adding a Remote Galaxy Server	10
2.3	Adding a New Workflow	13
2.4	Set User Quotas	21
3	Option 2: Power an Application	23
4	Workflow Settings	25
4.1	Customizing Emails	26
5	Monitoring Usage	29
5.1	The Job Queue	29
5.2	Usage Stats	32
6	Additional Settings	35
7	Executing Jobs and Tasks	37
7.1	Automatic Job Execution	37
7.2	Automatic Job Status Checking	37
7.3	Automatic Housekeeping	38
7.4	Manually Execute Jobs	38
8	Tripal Galaxy API	41
8.1	Tripal Galaxy API Overview	41
8.1.1	Adding a Galaxy Server	41
8.1.2	Get a List of Galaxies	42
8.1.3	Connect to Galaxy	42
8.1.4	Test if a Galaxy server is accessible.	43
8.1.5	Add a Workflow to Tripal	43
8.1.6	Prepare a Workflow for Execution	43
8.1.7	Get a History Name	44
8.1.8	Create a History and Upload Files	45

8.1.9	Invoke a Workflow	46
8.1.10	Check the status of a workflow submission	49
8.1.11	Retrieving Results from Galaxy	49
8.1.11.1	Retrieving a Dataset	49
8.1.11.2	Option 1: Download Files to the Server	52
8.1.11.3	Option 2: Providing Download Links for Users	52
8.2	Tripal Galaxy API Functions	53
	Index	61

Note: The Tripal Galaxy module is currently compatible only with release 19.01 of the Galaxy Project.

The Tripal Galaxy module is designed to support integration of [Tripal](#) with [Galaxy](#). In the past, community databases have often provided analytical tools that come prepared with site-specific data. Examples include BLAST, CAP3, and InterProScan servers, for example. These tools eased the informatic burden for some researchers by providing tools with site-specific data in easy to use web interfaces. With larger datasets and more complicated analytical workflows creating online tools becomes a more challenging task.

By integrating Tripal with Galaxy, the Tripal-based community database can offer more complicated analytical tools that support larger data sets using Galaxy as a backend. To this end, analytical workflows are created by site developers or bioinformaticists inside of the Galaxy user interface. Once tested and ready, the Tripal Galaxy module communicates with the Galaxy server to provide a web front-end for that workflow. Users can execute the workflow within the Tripal site, providing a user interface that site-users are familiar and comfortable with. Users need not know how to use Galaxy to execute the workflow, although, attribution is appropriately provided to the Galaxy server that provides the computation.

The Tripal Galaxy module provides more than just a “wrapper” for Galaxy. Site administrators can provide files to help end-users easily integrate data from the site within workflows. On Tripal v3 sites, user’s can create data collection containing data gleaned from the site which in turn can be used in Galaxy workflows. Quotas are provided to prevent users from overrunning the storage space of the server and usage statistics help a site admin learn which workflows are most used and who are the biggest users.

Development of the Tripal Galaxy module and accompanying starter workflows was funded by the [National Science Foundation](#) award #1443040 and is part of the [Tripal Gateway Project](#).

Install Tripal Galaxy

1.1 Installation

The Tripal Galaxy module is available as a full Drupal module. Therefore, it can be installed following the typical Drupal module installation either via the GUI or via Drush. However it requires a few dependencies:

- curl
- php-curl
- blend4php

Follow the typical method for installation of tools for your operating system to install `curl` and `php-curl`. For example, on Ubuntu systems you can install both `curl` and `php-curl` with the following command:

```
sudo apt-get install php-curl
```

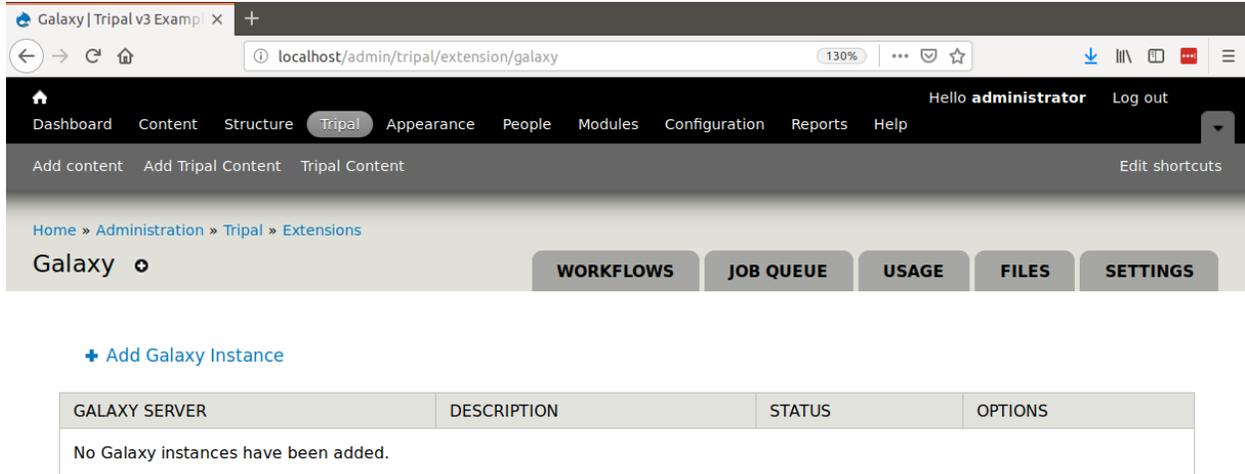
Additionally, this module requires the presence of the `blend4php` library. The `blend4php` library was developed as part of the Tripal Gateway Project as stand-alone PHP bindings to the Galaxy RESTful API. It allows any PHP-based web application to communicate with a remote Galaxy instance. To install `blend4php`, navigate to the `sites/all/libraries` directory of your Drupal installation and issue the following command:

```
git clone https://github.com/galaxyproject/blend4php.git
```

That's it! The library is now available for Drupal to find. Now, install the Tripal Galaxy module. We'll do so here using a Drush command:

```
drush pm-enable tripal_galaxy
```

You will be asked if you would like to download and then enable the module. Now that the Tripal Galaxy module is installed, navigate to the Administration > Tripal > Extensions > Galaxy page via the administrative menu. At this location is the administrative interface for interacting with remote Galaxy servers. Near the top of the page are several tabs including **Workflows**, **Job Queue**, **Usage**, **Files** and **Settings**.



1.2 Set Permissions

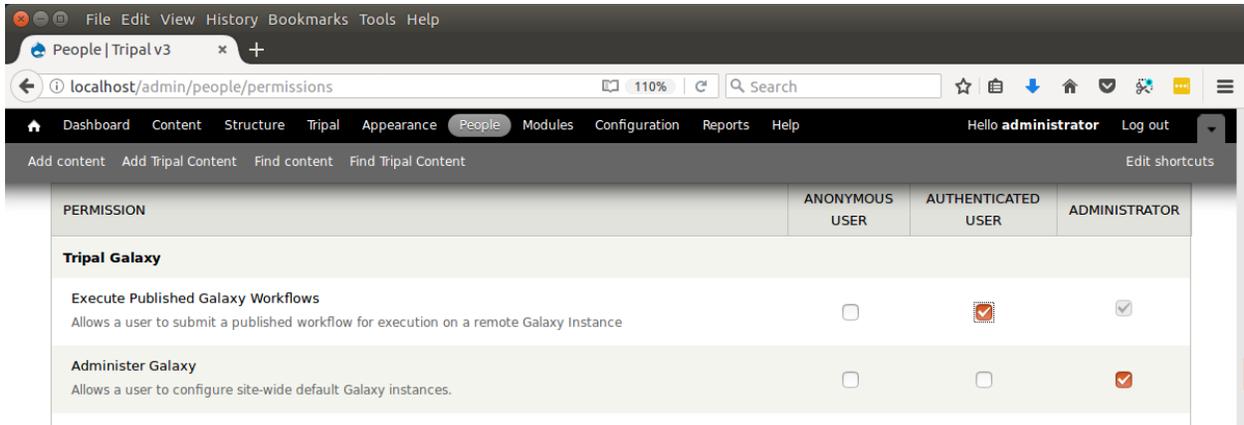
Now that the Tripal Galaxy module is installed, we must set permissions so that users can add submit workflows and administrators can manage the workflows. To set permissions, navigate to **People** page and click the **Permissions** tab in the top right. To allow end-users to submit workflows, you must set the following two permissions found in the **Tripal Galaxy** and **Tripal** sections of the permissions page:

- Tripal Galaxy > Execute Published Galaxy Workflows
- Tripal > Upload files

To allow a user to administer the Tripal Galaxy workflows you must set three permissions for the given role:

- Tripal Galaxy > Administer Galaxy
- Tripal Galaxy > Execute Published Galaxy Workflows
- Tripal > Upload files

The following screenshot shows an example of the permissions in the **Tripal Galaxy** section of the **Permission** page:



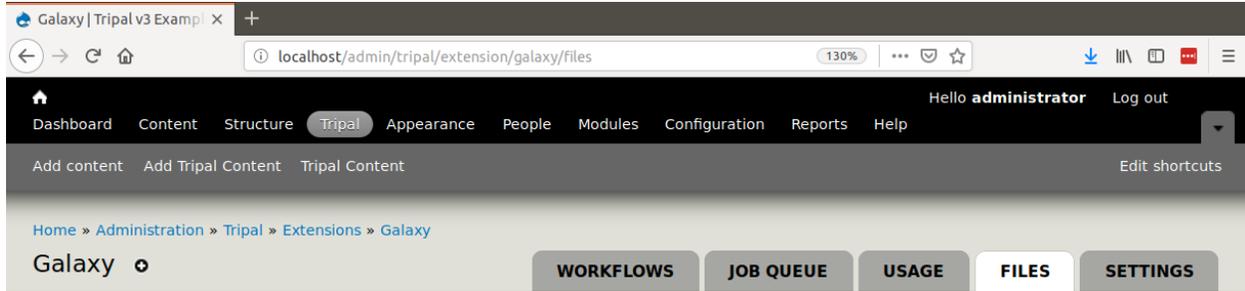
Warning: If your site is not new you may have additional roles. As a rule you should not give the anonymous user either permission.

Option 1: Automatic Webform Creation

The Tripal Galaxy module can automatically connect to a remote Galaxy server, and retrieve information about a workflow. That information is then used to construct an online web form that a site visitor can use to submit the workflow using their own datasets and/or data provided by the site.

2.1 Site-wide Files

The Galaxy workflows allow users to provide their own files, or to use site-wide files that are provided by the site administrators. As an administrator, you can provide site-wide files for anyone to use in a Galaxy workflow by navigating to the Galaxy administrative page and clicking the **Files** tab near the top. The following page appears:



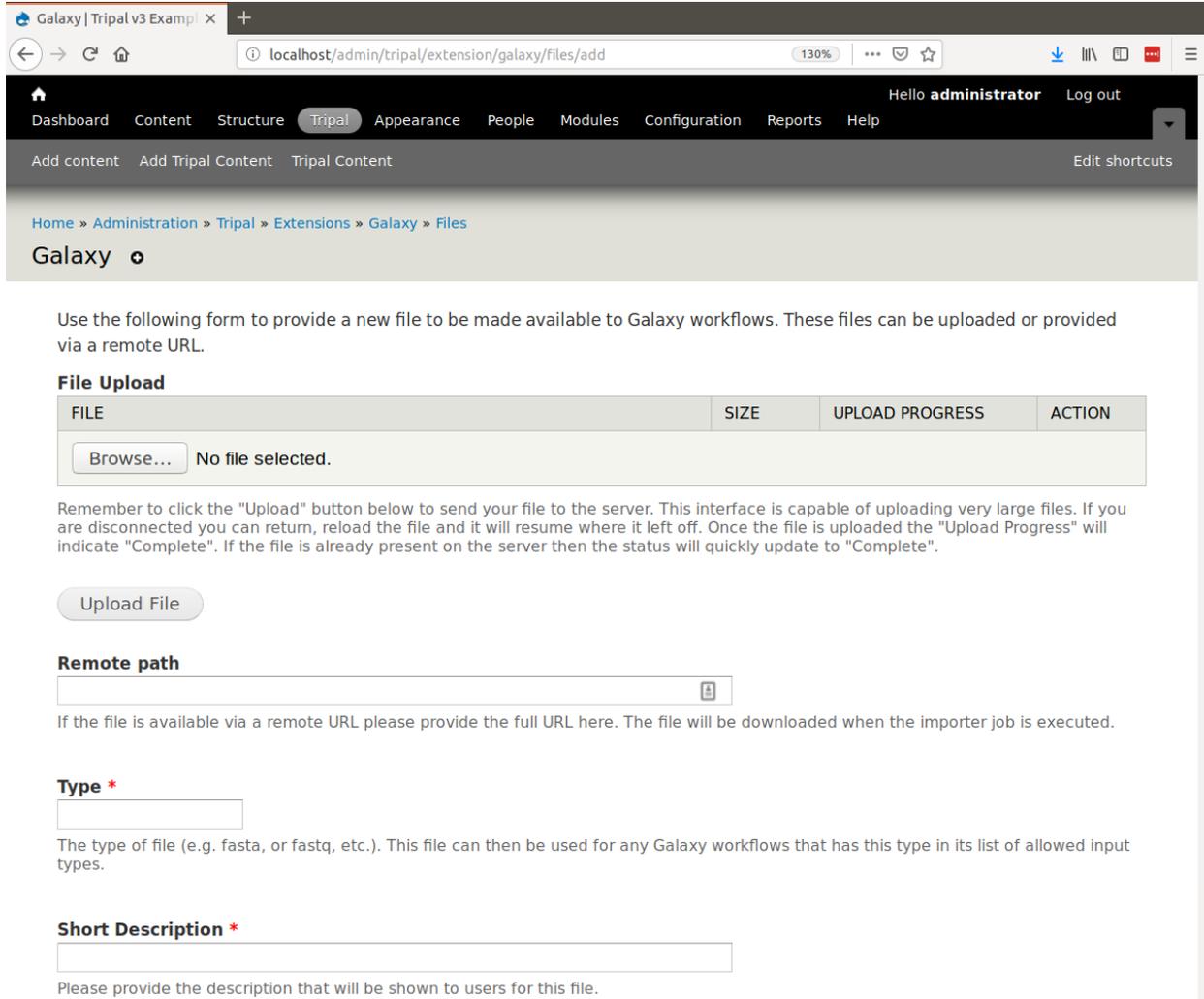
[+ Add a New File](#)

The following files will be made available to users inside of Galaxy workflows if the allowed types of the data input match the specified file type.

Files for Workflows

FILE	SHORT DESC	PATH	TYPE	ENABLED	ACTIONS
There are no site-wide files currently shared with for workflows.					

Initially, there are no site-wide files available. You can add a file (such as a whole genome assembly FASTA file) for use in workflows by clicking the **Add a New File** link. The following page appears.



Here you have two options for providing a file. Either upload the file using the File Upload form or add the URL to the file if it is available from a remote source. Typically files from remote sources are available with an HTTP URL or FTP URL. For example, if your site provides whole genome data and the FASTA file for the genome assembly is available for retrieval from an FTP site, you should enter the full FTP URL for the file into the **Remote Path** field. Next, you must provide the type of file. Examples of file types include `fasta`, `fastq`, `CSV`, etc. You must provide a file type because Galaxy workflows sometimes require specific file types as input. If an input requires a FASTA file then it expects a fasta file be provided. Sometimes remote files do not have an appropriate extension to properly determine the file type. Therefore, you must indicate the file type. Finally, the site's users should understand what the file is. Enter a very brief description of the file to help users recognize what it contains. When completed, click the Add File button.

In keeping with the example provided by the [Tripal v3 User's Guide](#) for Setup of an Example Genomics Site, the following screen shot shows a site-wide file for the JGI Citrus sinensis v1.1 assembly FASTA file that has been added. Any workflow that expects a FASTA file as input will now be able to use this file in a workflow:

Successfully added new site-wide file for Galaxy workflows.

[+ Add a New File](#)

The following files will be made available to users inside of Galaxy workflows if the allowed types of the data input match the specified file type.

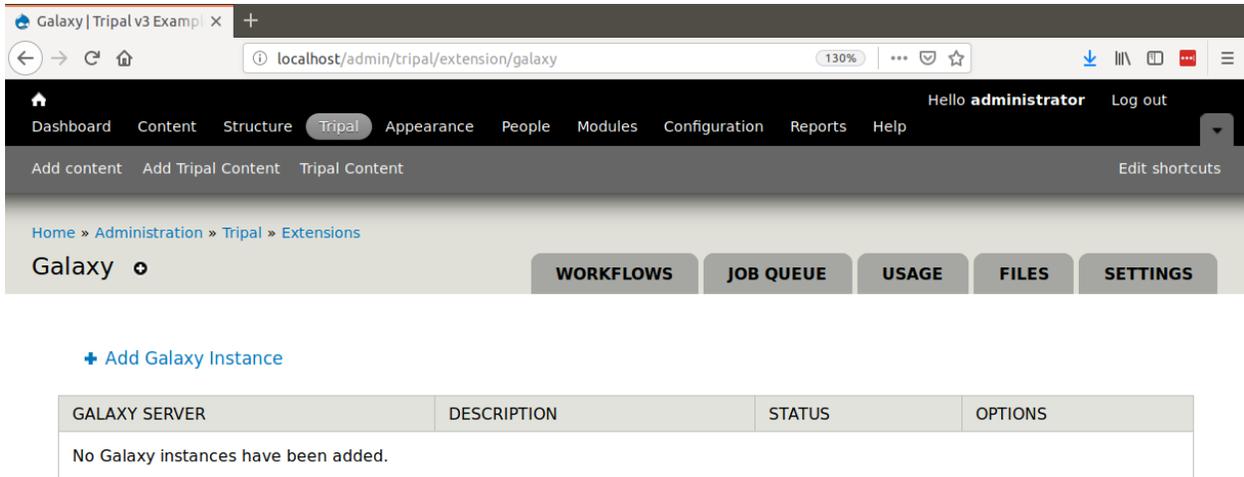
Files for Workflows

FILE	SHORT DESC	PATH	TYPE	ENABLED	ACTIONS
Csinensis_154_v1.fa	JGI v1.1 Citrus sinensis genome assembly	ftp://ftp.bioinfo.wsu.edu/species/Citrus_sinensis/C.sinensis_JGI_v1.1_genome/assembly/Csinensis_154_v1.fa	fasta	Yes	Edit Disable Delete

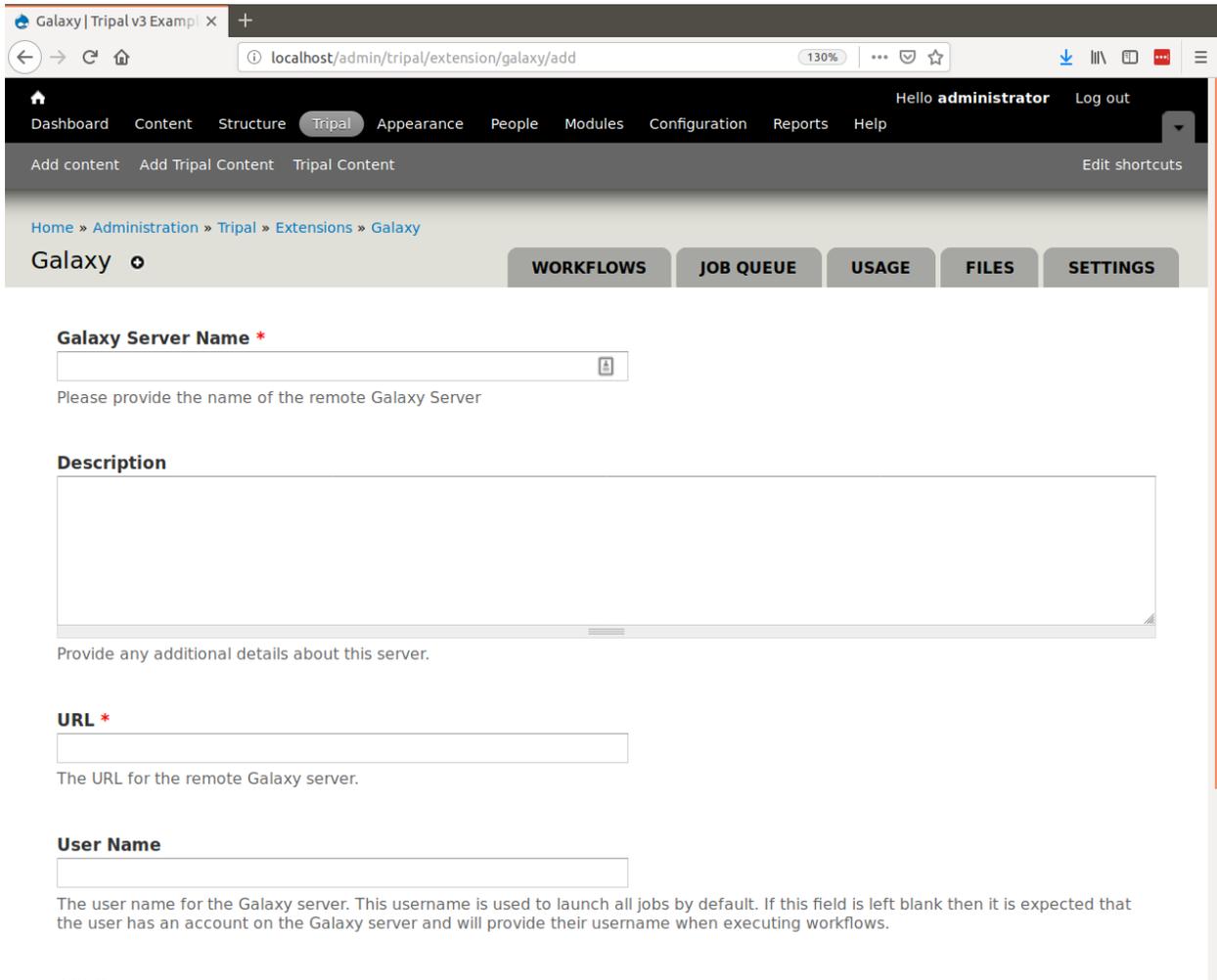
If you no longer need a specific file to be available for users you can either disable the file by clicking the **Disable** link in the Actions column, or you can delete the file by clicking the **Delete** link. Disabling the file will allow the file to remain as a site-wide file but exclude it from appearing for use in future workflow submissions.

2.2 Adding a Remote Galaxy Server

Before any workflows can be run, at least one remote Galaxy server must be integrated with Tripal. This can be done on the Tripal Galaxy administrative page. This page is available by navigating to **Administration > Tripal > Extensions > Galaxy**. The following page is presented:



We can add a new remote Galaxy server by clicking the Add Galaxy Instance link. The following page appears:



Here you can provide the details for the remote Galaxy instance. This can be a Galaxy instance that was setup and created specifically and dedicated for your site to use, a shared Galaxy instance setup at your institution or one of

several [public Galaxy instances](#). For this tutorial we will use the Galaxy Project’s “Use Galaxy” public instance. The following values are therefore provided to the fields:

Table 1: Example Galaxy Field Values

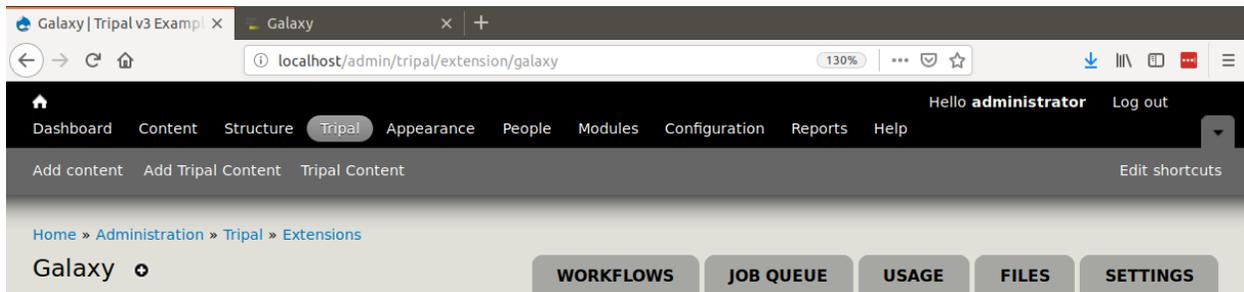
Field Name	Value
Galaxy Server Name	Use Galaxy Public Server
Description	The Galaxy Project’s public server. Does not provide all tools that may be needed, size and responsiveness is limited.
URL	https://usegalaxy.org/
User Name	<i>[The name of a user account on the Galaxy server]</i>
API Key	<i>[The API Key of the user]</i>

The fields described above provide everything needed to connect to a remote Galaxy instance. These include the URL, user name of a user on the remote Galaxy server and the API Key for that user. The **Galaxy Server Name** field and **Description** field are meant for you and the site’s users to know where computation is happening and the Description field is for the site admin.

The user name provided will be the account under which all Galaxy workflows will be executed, but the Tripal Galaxy module will ensure that each workflow submission runs independent of all others within this user account. It is recommended to create a **service** account on the remote Galaxy instance that is independent of a real person. This service account can be specific for your website only.

On Galaxy, each user has an API Key. It is via this key that Galaxy will allow remote connections from a client application like Tripal. To retrieve the API key for your service account user, log on to the remote Galaxy instance and navigate to **User > Preferences > Manage API Key**. By default, user’s do not have an API key. You can generate a key by clicking the link to generate a new key. Once the key is generated you must cut-and-paste the key into the **API Key** field of the Tripal Galaxy Instance setup form (shown above). Click the **Test Connection** button to make sure all details are correct and you can successfully connect using the user name and API Key. Then once successful, click the **Submit** button when done.

Now we can see that we have a Galaxy server available:



✓ Galaxy instance has been added

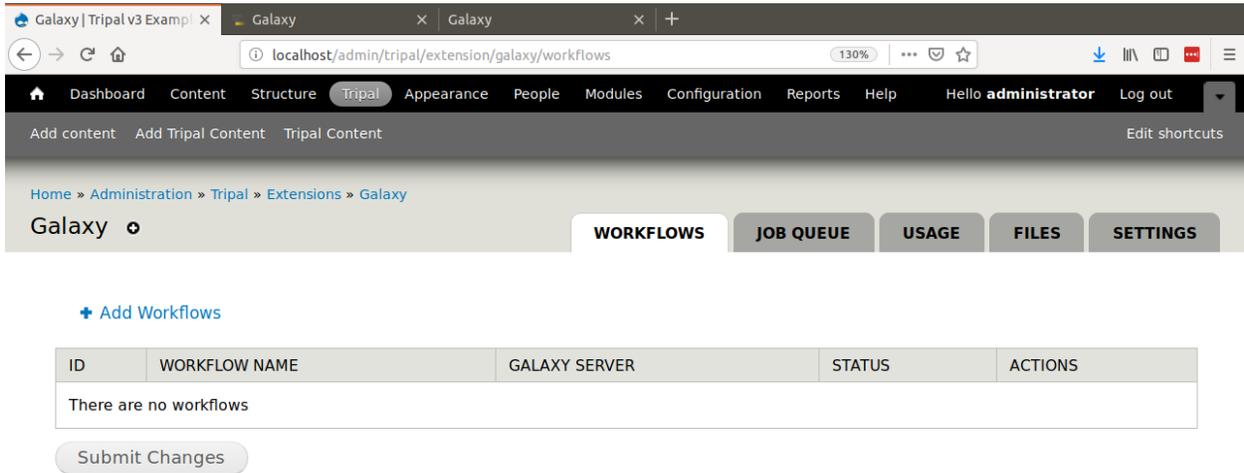
+ Add Galaxy Instance

GALAXY SERVER	DESCRIPTION	STATUS	OPTIONS
Use Galaxy Public Server	The Galaxy Project’s public server. Does not provide all tools that may be needed, size and responsiveness is limited.		edit test

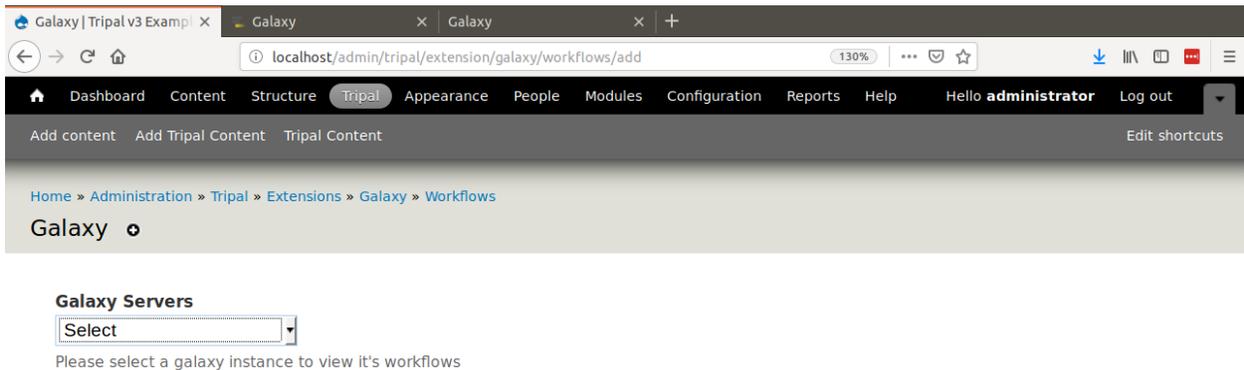
You can edit or re-test connectivity to the server by clicking the **edit** or **test** links in the **Options** column.

2.3 Adding a New Workflow

Now that Tripal is aware of a Remote Galaxy server we can integrate workflows with the site. To do this, navigate to the Tripal Galaxy administration page at **Administration > Tripal > Extensions > Galaxy** and click the **Workflows** tab.



Currently, there are no workflows available. A workflow can be added by clicking the **Add workflows** link. A page appears with a single select box containing a list of Galaxy servers that Tripal knows about.

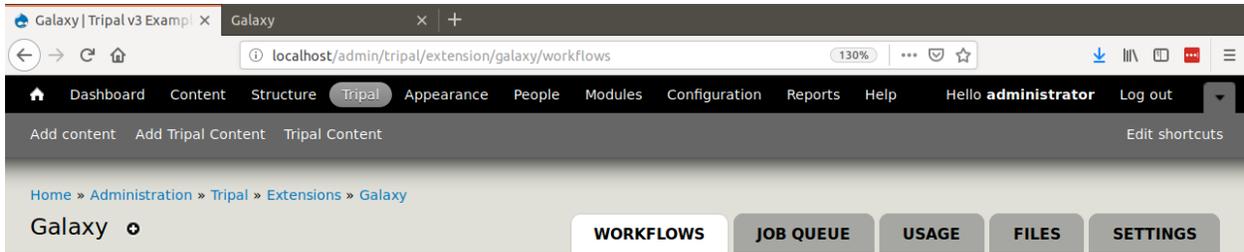


Select a Galaxy server and a list of workflows appears. In the previous *Adding a Remote Galaxy Server* section we added the UseGalaxy public server. However, for the remainder of this tutorial we will use a local Galaxy server which has several pre-installed workflows from the [Staton Lab Galaxy Workflows library](#).

The screenshot shows the Tripal Galaxy administration interface. The browser address bar is `localhost/admin/tripal/extension/galaxy/workflows/add`. The navigation menu includes Dashboard, Content, Structure, Tripal, Appearance, People, Modules, Configuration, Reports, and Help. The user is logged in as administrator. The breadcrumb trail is Home » Administration » Tripal » Extensions » Galaxy » Workflows. The main heading is 'Galaxy Servers'. A dropdown menu is set to 'Local Galaxy'. Below it is the text 'Please select a galaxy instance to view it's workflows'. A table lists five workflows with checkboxes for selection. A 'Submit' button is at the bottom.

ADD WORKFLOW	WORKFLOW NAME	WORKFLOW ID
<input type="checkbox"/>	wgcna-analysis (imported from uploaded file)	1cd8e2f6b131e891
<input type="checkbox"/>	mapping-hisat2-single-end	33b43b4e7093c91f
<input type="checkbox"/>	differential-expression-deseq2-analysis (imported from uploaded file)	ebfb8f50c6abde6d
<input type="checkbox"/>	blast-protein-sequence (imported from uploaded file)	f597429621d6eb2b
<input type="checkbox"/>	bowtie2-alignment-pe-dataset-collection (imported from uploaded file)	f2db41e1fa331b3e

Here we can select any number of workflows to enable on our Tripal site by checking the select box to the left of each workflow and clicking the **Submit** button. The Tripal Galaxy module will connect to the remote Galaxy server, retrieve information about the workflow and automatically create a web form for users to submit. For this example we will enable the workflow named `wgcna-analysis` (imported from uploaded file) which performs gene co-expression network (GCN) construction.



✓ The workflow was successfully added!

Sometimes these workflows need a bit of tweaking to make them more intuitive for users. Click the title of the workflow below to customize how it will appear to users. Consider improving:

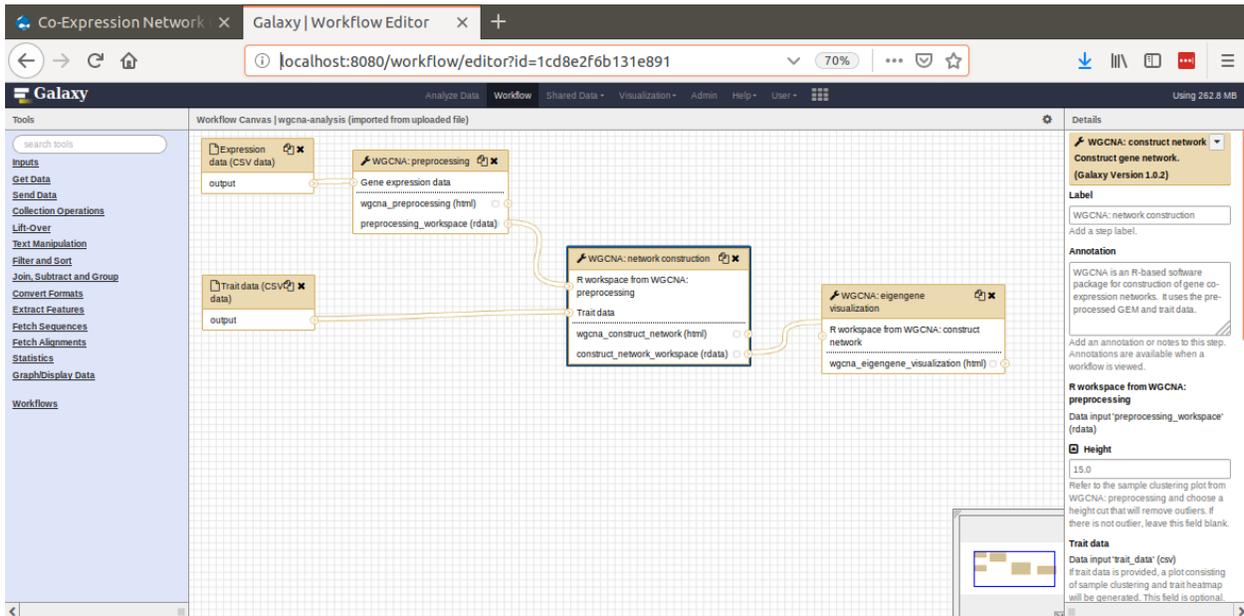
- The workflow title.
- The instructions for the workflow.
- The "About this step" field for each tool to help the user understand the workflow.
- Each tool setting descriptions as desired.

+ Add Workflows

ID	WORKFLOW NAME	GALAXY SERVER	STATUS	ACTIONS
80	<p><i>Webform Title:</i> wgcn-analysis (imported from uploaded file)</p> <p><i>Workflow Name:</i> wgcna-analysis (imported from uploaded file)</p>	Local Galaxy	Active	Remove Disable

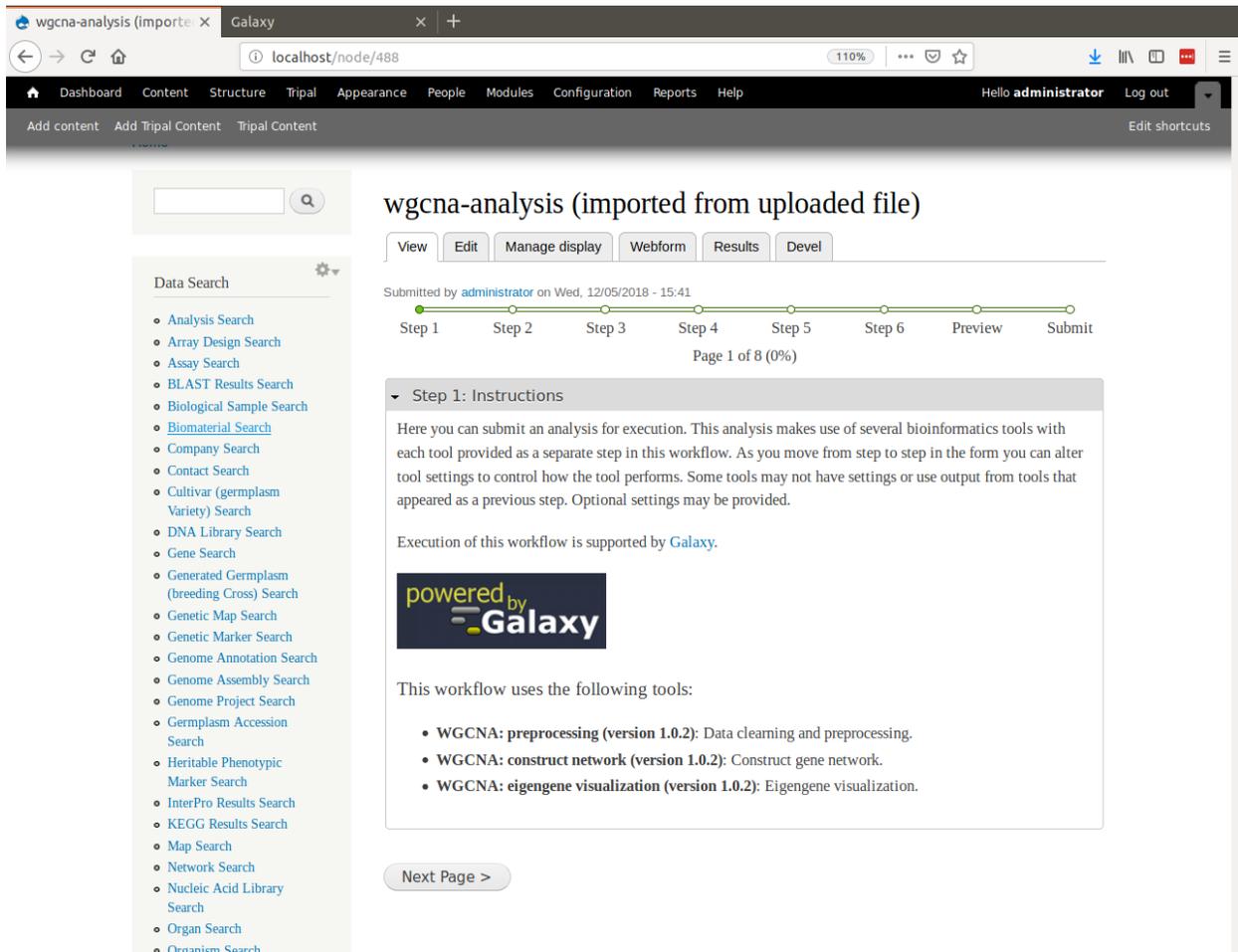
Submit Changes

Below is an image of the WGCNA workflow on the Galaxy server:

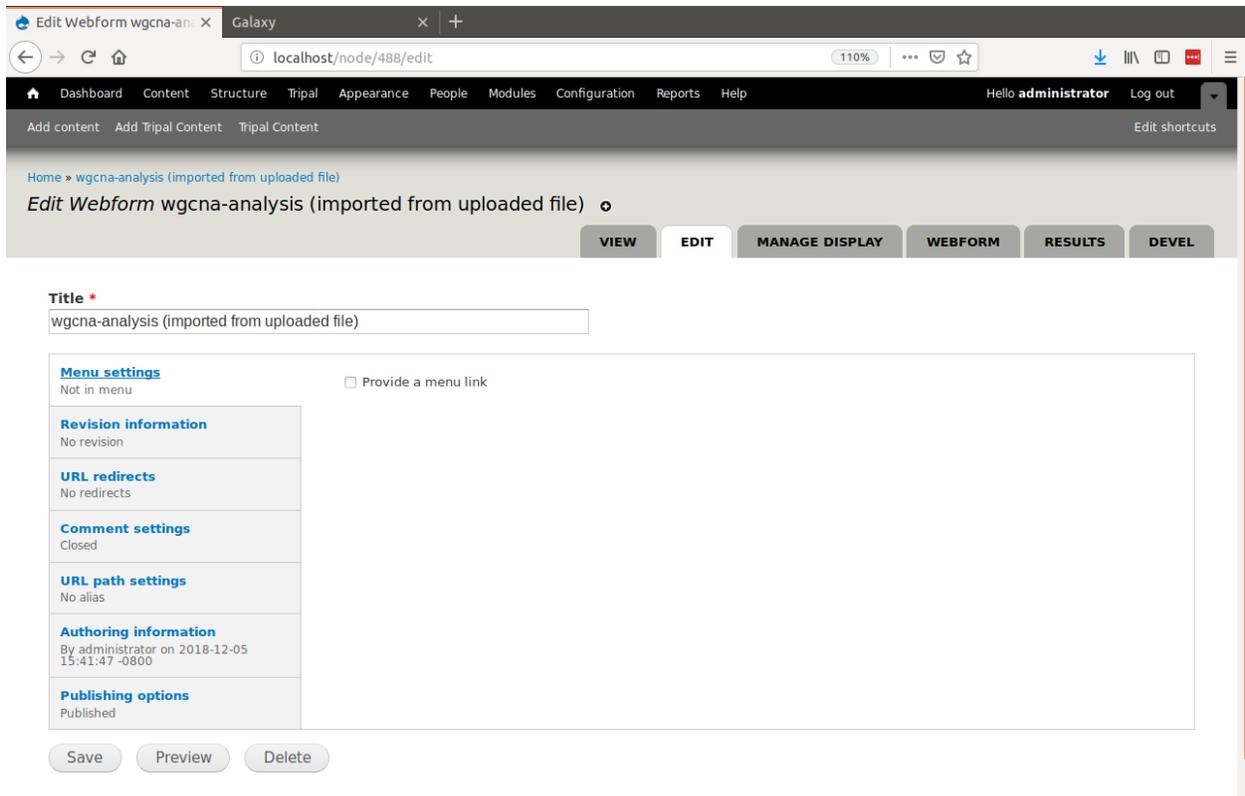


After the workflow has been added to Tripal a message is provided indicating that we may want to adjust the title, instructions and other information about the workflow. We may want to do this to help improve the user experience for a user who wants to run a workflow and needs details or instructions specific to your site. To do this, click the link

to the workflow in the **Workflow Name** column. The following page appears:



Tripal Galaxy will provide a set of default introductory instructions which are shown on the page. The title of the workflow is also not very user friendly. We can first change the title by clicking on the **Edit** tab. Here you can change the title to something such as “Co-Expression Network Construction (WGCNA)”



Click the **Save** button.

Next, to update the instructions, click the **Webform** tab. Here you will see a list of every field in every step of the workflow.

The screenshot shows the Tripal Galaxy Workflow Editor interface. The browser address bar indicates the URL is localhost/node/488/webform. The navigation menu includes Dashboard, Content, Structure, Tripal, Appearance, People, Modules, Configuration, Reports, and Help. The user is logged in as administrator. The main content area is titled 'Co-Expression Network Construction (WGCNA)' and features several tabs: VIEW, EDIT, MANAGE DISPLAY, WEBFORM (selected), RESULTS, and DEVEL. Below these are sub-tabs for Form components, Conditionals, E-mails, Form settings, Export, and Import. A 'Show row weights' link is also present. The main table lists various form components for three steps.

LABEL	FORM KEY	TYPE	VALUE	REQUIRED	OPERATIONS
+ Step 1: Instructions	galaxy_webform	Fieldset	-		Edit Clone Delete
+ Step 1: Instructions	galaxy_webform	Markup	<p>Here you can submit an...		Edit Clone Delete
+ Step 2	step_2_pagebreak	Page break	-		Edit Clone Delete
+ Description	Step_0_annotation_fieldset	Fieldset	-		Edit Clone Delete
+ workflow step annotation	Step_0_annotation_markup	Markup	At this step the Gene...		Edit Clone Delete
+ Step 2: Data File	Step_0_fieldset	Fieldset	-		Edit Clone Delete
+ Data File	step_2_7	File	-		Edit Clone Delete
+ Step 3	step_3_pagebreak	Page break	-		Edit Clone Delete
+ Description	Step_1_annotation_fieldset	Fieldset	-		Edit Clone Delete
+ workflow step annotation	Step_1_annotation_markup	Markup	At this step, a comma...		Edit Clone Delete
+ Step 3: Data File	Step_1_fieldset	Fieldset	-		Edit Clone Delete
+ Data File	step_3_12	File	-		Edit Clone Delete

Warning: Be cautious editing any of the fields on the **Webform** tab as they are needed for the workflow. It is safe to edit any field of type **Markup** or, to change the **Label** and **Description**. Do not change the **Form Key** or any other default values.

To change the instructions, click the **Edit** link for the column with the **Label** *Step : Instructions* of type **Markup** (the field on the second row). The following page appears:

Note:

If the **value** field on the form shows pure HTML you may want to install a WYSIWYG editor such as `ckeditor` using a command such as:

```
drush pm-enable ckeditor
```

Edit the instructions to your liking and click the **Save component** button at the bottom of the page. Click the **View** tab to see how things have changed:

Co-Expression Network Construction (WGCNA)

Submitted by administrator on Wed, 12/05/2018 - 15:41

Step 1 Step 2 Step 3 Step 4 Step 5 Step 6 Preview Submit

Page 1 of 8 (0%)

▼ Step 1: Instructions

Gene Co-Expression Networks GCNs are often used to help identify groups of genes that are co-functional. The analysis requires a file containing a Gene Expression Matrix (GEM) containing gene expression levels derived from either RNA-seq or microarray data sets. Genes that tend to be expressed at similar levels across a set of samples are considered co-expressed. A correlation value (often Pearson's or Spearman's correlation) is assigned to each gene pair. WGCNA is an R-based package that performs pair-wise gene correlations and uses a method referred to as soft-thresholding to cull pair-wise correlation values that are non significant. To learn more about WGCNA and co-expression network construction please see the [WGCNA homepage](#).

EditExecution of this workflow is supported by [Galaxy](#).

powered by Galaxy

This workflow uses the following tools:

- **WGCNA: preprocessing (version 1.0.2):** Data cleaning and preprocessing.
- **WGCNA: construct network (version 1.0.2):** Construct gene network.

You can now browse through the workflow, to ensure that the end-user experience matches your expectations. Click the **Next Page** button to see the next step in the workflow.

If you see a description or help text that does not meet your liking you may change it by clicking the **Webform** tab, finding the correct component and editing appropriately. Remember to only change labels, descriptions and help text. Do not change the form keys or default values or conditionals.

Once you have completed any edits to the workflow you can make it available for end-users by adding the URL for the workflow to your sites Menu as needed. Notice in the previous screenshot that our workflow can be found on our site at the relative URL `node/488`. Navigate to the **Administer > Structure > Menus** and add the workflow to the most appropriate menu for your site.

Note: We do not show here how to create menus as that is covered by the Tripal v3 User's Guide and Tripal documentation.

2.4 Set User Quotas

One of the most important settings to configure before exposing workflows is User quotas. Setting Quotas is a function provided by Tripal itself and can be found at **Administer > Tripal > User File Management** under the **User Quotas** tab. Instructions for setting User Quotas are found in the Tripal v3 User's Guide.

Option 2: Power an Application

You may not want to expose workflows to end-users via the automatical web forms. However, you can use the *Tripal Galaxy API Functions* and `blend4php` to directly integrate with Galaxy through your own Tripal application. To do so, please visit the *Tripal Galaxy API Overview* pages for more details.

An example application that uses this approach is the *Cartogratree* application available on the *TreeGenes* website.

CHAPTER 4

Workflow Settings

Regardless if you add workflows using option #1 or #2 there are a few settings you can set to customize the way users interact with workflows. On the workflow administration page at `Administer > Tripal > Extension > Galaxy > Workflows` you will see a `Settings` link for each workflow.

Galaxy | Tripal v3 Example - Mozilla Firefox

Galaxy | Tripal v3 Examp | X +

localhost/admin/tripal/extension/galaxy/workflows 130%

Hello administrator Log out

Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help

Add content Add Tripal Content Tripal Content Edit shortcuts

Home » Administration » Tripal » Extensions » Galaxy

Galaxy

WORKFLOWS JOB QUEUE USAGE FILES SETTINGS

+ Add Workflows

ID	WORKFLOW NAME	GALAXY SERVER	STATUS	ACTIONS
81	<i>Webform Title:</i> Differential Gene Expression (DESeq2) <i>Workflow Name:</i> differential-expression-deseq2-analysis (imported from uploaded file)	Local Galaxy	Active	Settings Disable Remove
80	<i>Webform Title:</i> Co-Expression Network Construction (WGCNA) <i>Workflow Name:</i> wgcna-analysis (imported from uploaded file)	Local Galaxy	Active	Settings Disable Remove

Clicking the `Settings` link beside a workflow will take you to the `Settings` page for that workflow.

4.1 Customizing Emails

Currently the setting page allows you to customize the email text that is sent to end-users when the workflow is invoked and also when it either fails or succeeds.

Workflow Settings: Co-Expression Network Construction (WGCNA) | Tripal v3 Example - Mozilla Firefox

Workflow Settings: Co-E: X

localhost/admin/tripal/extension/galaxy/workflows/settings/80

130%

Hello administrator Log out

Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help

Add content Add Tripal Content Tripal Content Edit shortcuts

Home » Administration » Tripal » Extensions » Galaxy » Workflows

Workflow Settings: Co-Expression Network Construction (WGCNA)

EMAIL MESSAGE ON SUCCESS

Set the subject and body for emails that are sent to users when the workflow successfully completes. You may use the tokens as placeholders for values appropriate for the workflow, site and user. If nothing is provided then the defaults will be used.

Do not send email on success

Subject Line

Your job at [site_name] has completed!

Enter the text that should appear on the subject line.

Email Body

You recently submitted a job titled: [workflow_title].
 This job has completed.
 You may view results of this job at [results_link].
 Thank you.
 Site Name: [site_name]

Enter the text that should appear in the body of the email.

AVAILABLE TOKENS

EMAIL MESSAGE ON FAILURE

You can change the text of the subject line and the body of the email by altering the text in the fields. Additionally, you can use tokens as placeholders for text that you want included in the messages. You can find a list of available tokens by opening the Available Tokens fieldset. If you do not want a workflow to send email you can disable the email by clicking the checkbox whose title begins: do not send email.

Workflow Settings: Co-E: X +

localhost/admin/tripal/extension/galaxy/workflows/settings/80

130%

Hello administrator Log out

Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help

Add content Add Tripal Content Tripal Content Edit shortcuts

Enter the text that should appear in the body of the email.

AVAILABLE TOKENS

Use any of these tokens in the subject line or body of your email. Appropriate values will be substituted in their place.

TOKEN	DESCRIPTION
[site_name]	The name of this site
[workflow_title]	The title of the workflow.
[submission_init]	The time when the workflow was submitted to Tripal for execution.
[submission_start]	The time when the submitted workflow started.
[submission_end]	The time when the submitted workflow ended.
[submission_status]	The current status of the submitted workflow.
[user_name]	The name of the user that submitted the workflow.
[jobs_link]	A link to the page where all submitted workflows are listed for the user.
[results_link]	A link to the results page for the workflow. This link is only valid if the workflow completes successfully.

EMAIL MESSAGE ON FAILURE

EMAIL MESSAGE ON SUCCESS

For example, if you want to include the submission time of the workflow in the body of the email that gets sent when the workflow is invoked, you should add the token `[submission_init]`. When the email is sent this token will be replaced with the submission time. Be sure to click the `Submit` button after making any changes.

Monitoring Usage

Once workflows are made available to end-users you will want to monitor usage. There are two tools to assist with this. Navigate to the Tripal Galaxy administration pages at **Administer > Tripal > Extensions > Galaxy**. You will see two tabs named **Job Queue** and **Usage**.

5.1 The Job Queue

The job queue is a tool that lists all of the currently active jobs. It provides details about who submitted the job, the job state, submission, start and end time.

The screenshot shows the Tripal Galaxy administration interface. The breadcrumb trail is: Home » Administration » Tripal » Extensions » Galaxy. The page title is "Analysis Results". There are tabs for WORKFLOWS, JOB QUEUE (selected), USAGE, FILES, and SETTINGS. Below the tabs, there are filter fields for Workflow (set to "Is equal to"), User, and Submission Status. An "Apply" button is present. Below the filters is a table of active jobs.

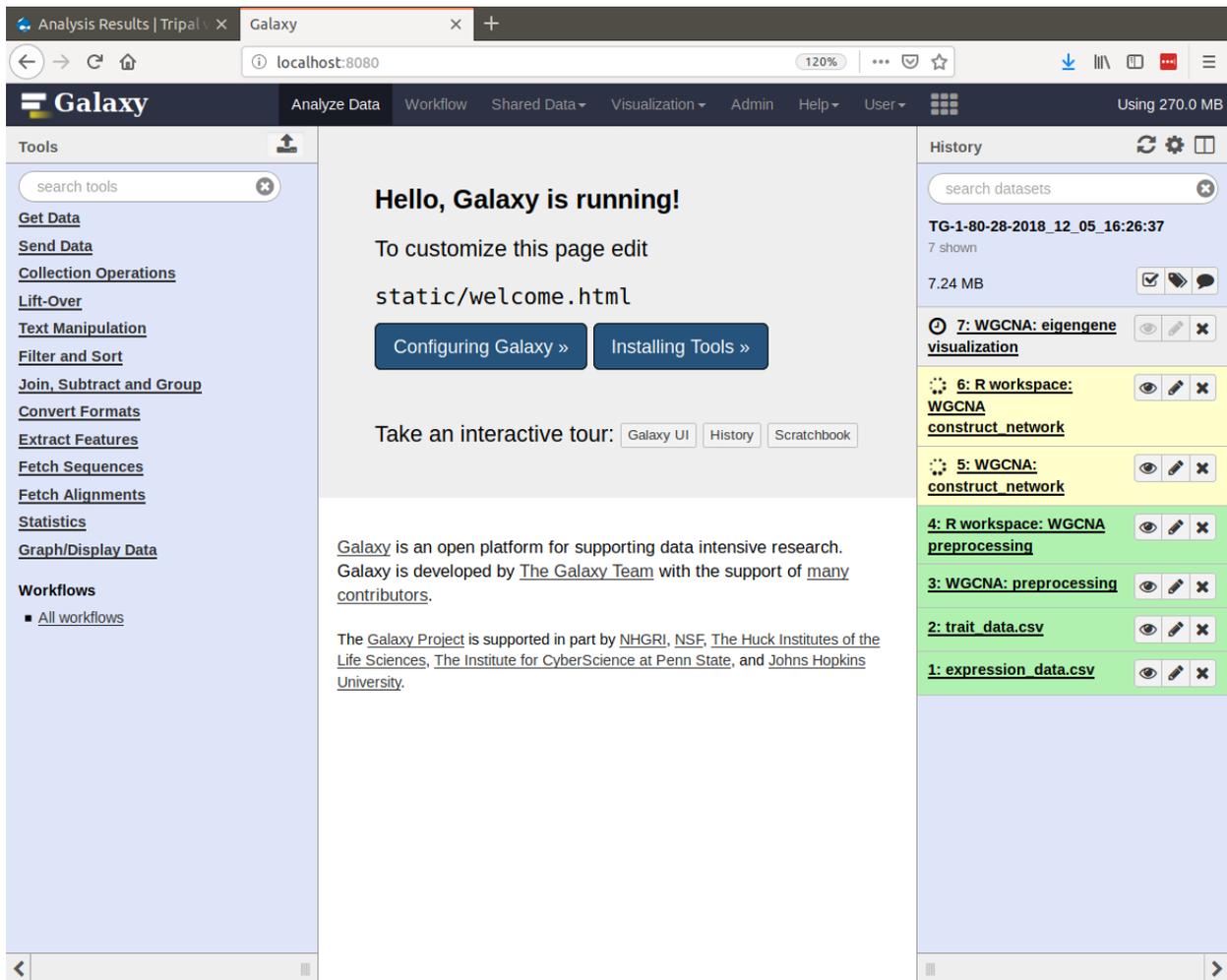
ID	WORKFLOW	GALAXY SERVER	USER	SUBMISSION TIME	START TIME	END TIME	STATUS	RESULTS
28	Co-Expression Network Construction (WGCNA)	Local Galaxy	administrator	12/05/2018 - 16:26			Waiting	View

powered by Galaxy

The example above shows the first submission of a Galaxy workflow on our example site. Currently the workflow has a status of ‘Waiting’. The job will stay in the waiting state until the Tripal Job launcher invokes the job. If you have not setup Tripal for automated job submission, please review the [instructions in the Tripal v3 User’s Guide](#).

When a job is invoked by the Tripal Job launcher, the following occurs: 1. A history is created uniquely for this workflow submission on the remote Galaxy site. The Tripal Galaxy module ensures that each workflow submission executes in a unique history. 2. All files needed for the workflow are uploaded to the remote site into the newly created history. 3. Any data collections (groups of files such as paired-data) are organized in the history on the Galaxy server. 4. The workflow is invoked and beings executing on the Galaxy server once resources are available. 5. An email is sent to the end-user indicating that the job has been submitted.

If you were to visit the Galaxy server you should be able to find the history for the submitted workflow and see that the job is running.



To do this, you need to know the history name. You can find the history ID for this submission by clicking on the **View Results** link in the job queue. The resulting page will indicate that no results are available for non completed jobs, but if you open the **Submission Details** field set you can find the history name in the table.

The screenshot shows a web browser window with the URL `localhost/admin/tripal/extension/galaxy/workflows/report/28`. The page title is "Workflow Submission Details". The navigation menu includes "Dashboard", "Content", "Structure", "Tripal", "Appearance", "People", "Modules", "Configuration", "Reports", "Help", "Hello administrator", and "Log out".

Analysis Name
Co-Expression Network Construction (WGCNA)

SUBMISSION DETAILS

WORKFLOW NAME	Co-Expression Network Construction (WGCNA)
SUBMISSION ID	28
WORKFLOW ID	1cd8e2f6b131e891
INVOCATION ID	e85a3be143d5905b
STATUS	Processing
SUBMISSION DATE	Wed, 12/05/2018 - 16:26
HISTORY NAME	TG-1-80-28-2018_12_05_16:26:37
START TIME	
END TIME	
GALAXY SERVER	Local Galaxy 

Results

Currently, no results exist for this job. The current state of the job is: *Processing*

As the workflow proceeds the status in the job queue is updated until finally the workflow completes:

The screenshot shows the Tripal Galaxy Administration interface. The browser address bar indicates the URL is localhost/admin/tripal/extension/galaxy/job-queue. The navigation menu includes Dashboard, Content, Structure, Tripal, Appearance, People, Modules, Configuration, Reports, Help, and a user profile for administrator. The breadcrumb trail is Home » Administration » Tripal » Extensions » Galaxy. The main content area is titled 'Analysis Results' and has tabs for WORKFLOWS, JOB QUEUE, USAGE, FILES, and SETTINGS. The JOB QUEUE tab is active, showing a filter section with a Workflow dropdown set to 'Is equal to', and empty input fields for User and Submission Status. An 'Apply' button is present. Below the filter is a table of workflow submissions:

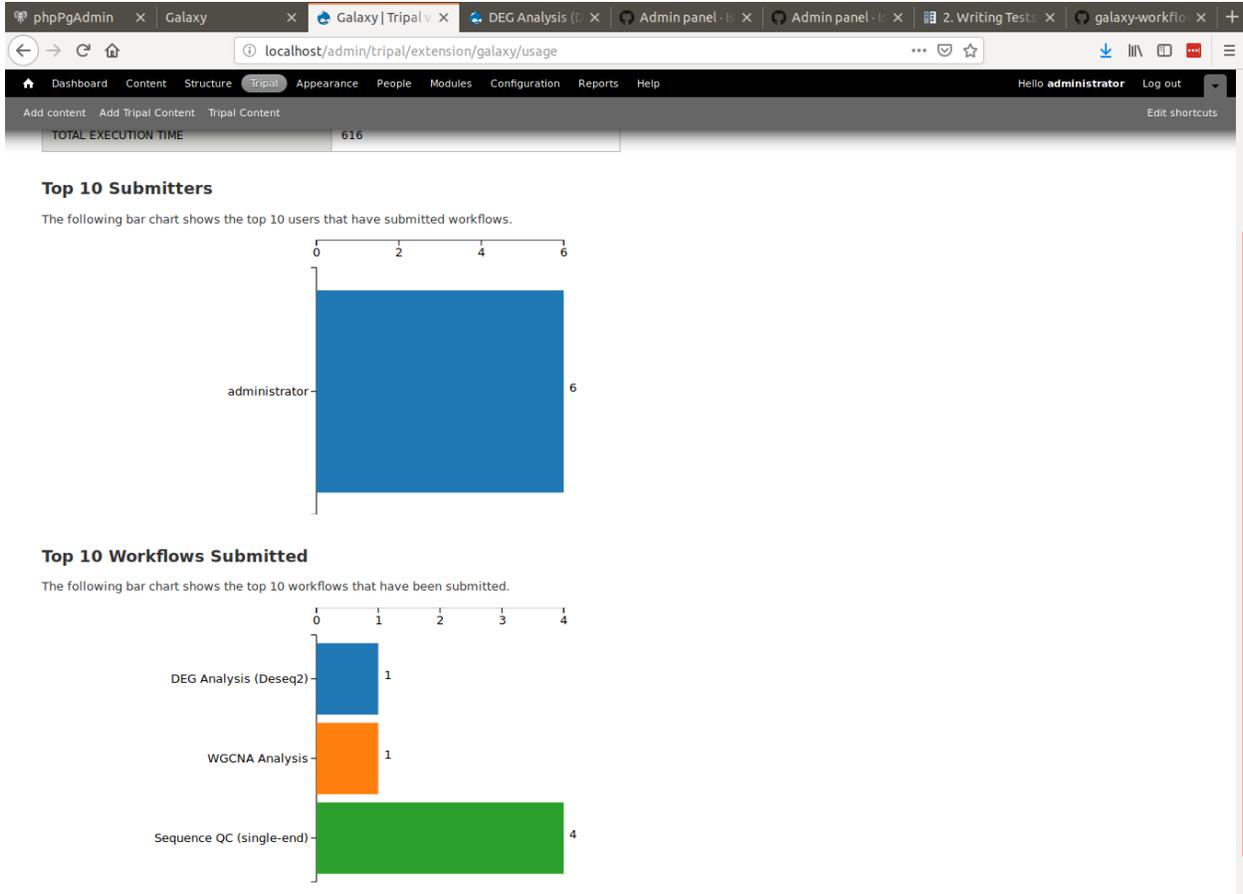
ID	WORKFLOW	GALAXY SERVER	USER	SUBMISSION TIME	START TIME	END TIME	STATUS	RESULTS
28	Co-Expression Network Construction (WGCNA)	Local Galaxy	administrator	12/05/2018 - 16:26	12/05/2018 - 16:34	12/05/2018 - 16:43	Completed	View

At the bottom left of the page, there is a logo that says 'powered by Galaxy'.

5.2 Usage Stats

The Tripal Galaxy Module also provides some simple bar charts indicating the top workflows submitted and users of workflows. To retrieve these reports navigate to **Administer > Tripal > Extensions > Galaxy** and click the **Usage** tab.

The following is an example results for a new site that has had just a few workflows executed by the administrator user (most likely just for testing).



Additional Settings

There are a few additional settings you may want to adjust to fine tune integration with Galaxy. You can adjust settings by navigating to the **Administer > Tripal > Extensions > Galaxy** page and click the **Settings** tab. Here you will find two settings. One for altering the expiration date for histories on remote Galaxy servers and one for adjusting how often the the Tripal Galaxy module should run the cron.

Galaxy | Tripal v3 Example x +

localhost/admin/tripal/extension/galaxy/settings 120%

Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help Hello administrator Log out

Add content Add Tripal Content Tripal Content Edit shortcuts

Home » Administration » Tripal » Extensions » Galaxy

Galaxy **WORKFLOWS** **JOB QUEUE** **USAGE** **FILES** **SETTINGS**

History Age Expiration (in days) *

60

Delete histories on remote galaxy server after how many days?

Galaxy cron run (in minutes) *

60

How frequently the galaxy cron run should happen. We recommend at least once per day as workflow expiration is tied into this.

Save Changes

The first setting ensures that results submitted by users do not pile up and use up your file storage quota on the remote Galaxy server. The second settings adjust how often the Tripal Galaxy cron job runs. When the cron runs the following actions occur:

1. **Server Status Checks:** All Galaxy servers registered with Tripal are checked to see if they are available. If not then workflows are automatically marked as disabled preventing further submissions. If workflows are disabled, a message is added to the Drupal dashboard.

2. **History Cleanup:** All histories on the remote Galaxy servers that are older than the number of days specified by the **History Age Expiration** value are deleted.

Executing Jobs and Tasks

7.1 Automatic Job Execution

In a production setting, jobs submitted by end-users should be executed automatically. There are two ways to automate job submission. First, if you are writing your own application using *Option 2: Power an Application*, then you will most likely invoke your workflow using the Tripal Galaxy API. In this case you are invoking the workflow instantly and nothing further is needed. If you are using *Option 1: Automatic Webform Creation*, then user submitted workflows are added to the Tripal job management system. You can therefore automate job execution by following the directions in the [Tripal User's Guide](#) for job automation.

Note: For near instantaneous invocation of workflows consider using the [Tripal Daemon Module](#). The Tripal Daemon runs as a background service that constantly checks for new jobs and then executes them.

7.2 Automatic Job Status Checking

When submitted workflows are added to the Tripal Jobs queue or invoked via the API you will want to check the status of those jobs. For this we need automation to ensure that jobs are checked on a regular bases. The best way to do this is to setup an independent [Cron entry](#) for the site. You can also learn about Cron integration with Drupal at the [Tripal User's Guide](#). To automate job status checks, the `drush trp-galaxy-status` command-line tool is available. Suppose our site is at `http://my.tripal.site` and it is found on the server at the path `/var/www/html` then we could add a new cron entry to check for job status updates every 5 minutes. On an Ubuntu system, to edit the cron execute this command:

```
sudo crontab -e
```

Then add an entry similar to the following

```
0,5,10,15,20,25,30,35,40,45,50,55 * * * * drush trp-galaxy-status --uri=http://my.  
↳tripal.site/ --root=/var/www/html
```

Note that we specified the URL of the site with the `--uri` argument and the location of our Drupal site with the `--root` option.

Warning: It is critical that the `--uri` argument is provided. When checking a job status if the job has failed or completed then an email will be sent. Without this argument all links will default to site name of `http://default/` which is incorrect.

7.3 Automatic Housekeeping

In the *Additional Settings* section of this document you learned that you can control how frequent cleanup of old histories on the Galaxy server occurs. These tasks do not use the Tripal Job's management system but rather the Drupal cron, and unlike job execution, you do not want the Cron to execute too often, perhaps once an hour or a few times a day.

You should setup the Drupal cron according to the instructions provided in the [Tripal User's Guide](#).

Once the Drupal cron is setup and running then housekeeping will begin.

7.4 Manually Execute Jobs

Note: Except for testing or debugging you should avoid manual execution of jobs in a production environment. Rather, jobs should be executed automatically.

As a site administrator you have the ability to manually invoke and check the status of a job. There are two drush commands for this: `drush trp-galaxy-invoke` and `drush trp-galaxy-status`.

To manually invoke a workflow submission you can execute the `trp-galaxy-invoke` command. It requires that you know the submission ID and that you set the URI of the website. For example, suppose our Tripal site is named `http://my.tripal.site` and our submission ID is 30. The following on the command-line will invoke the workflow.

```
drush trp-galaxy-invoke --submission=30 --uri=http://my.tirpal.site/
```

To check the status use the same arguments for the `trp-galaxy-status` command:

```
drush trp-galaxy-status --submission=30 --uri=http://my.tirpal.site/
```

You must provide the `--uri` argument because both of these commands may result in an email being sent to the end-user. By default, drush is not able to resolve the domain name of the site and emails will be sent with a URL of `http://default/`.

To find the submission ID for any submitted workflow, navigate to Adminster > Tripal > Extensions > Galaxy > Job Queue. The submission ID appears in the first column of the resulting table.

The screenshot shows a web browser window displaying the 'Analysis Results | Tripal v3 Example - Mozilla Firefox' page. The URL is localhost/admin/tripal/extension/galaxy/job-queue. The user is logged in as 'administrator'. The page has a navigation menu with options like Dashboard, Content, Structure, Tripal, Appearance, People, Modules, Configuration, Reports, and Help. Below the navigation is a breadcrumb trail: Home » Administration » Tripal » Extensions » Galaxy. The main content area is titled 'Analysis Results' and has tabs for WORKFLOWS, JOB QUEUE (selected), USAGE, FILES, and SETTINGS. There are filter fields for Workflow (set to 'Is equal to'), User, and Submission Status, with an 'Apply' button. Below the filters is a table with the following data:

ID	WORKFLOW	GALAXY SERVER	USER	SUBMISSION TIME	START TIME	END TIME	STATUS	RESULTS
30	Differential Gene Expression (DESeq2)	Local Galaxy	administrator	12/06/2018 - 14:42	12/15/2018 - 19:58	12/15/2018 - 19:58	Completed	View
29	Co-Expression Network Construction (WGCNA)	Local Galaxy	administrator	12/06/2018 - 14:41	12/15/2018 - 20:09	12/15/2018 - 20:16	Completed	View
28	Co-Expression Network Construction (WGCNA)	Local Galaxy	administrator	12/05/2018 - 16:26	12/15/2018 - 19:53	12/15/2018 - 20:00	Completed	View

At the bottom left of the page, there is a logo that says 'powered by Galaxy'.

Tripal Galaxy API

Note: Use of the Tripal Galaxy API requires the installation of the Tripal Galaxy module and the [blend4php library](#)..

The Tripal Galaxy API allows you to integration Tripal and Galaxy and customize some or all of the process. By default the Tripal Galaxy module provides an interface that allows a site to offer a step-by-step interface for a workflow, maintaining the look-and-feel of the site. If that functionality is sufficient you will not need this API documentation.

The blend4php library was specifically built to provide a native PHP interface to the Galaxy API such that any PHP application can connect to the RESTful web services of a remote Galaxy server. While blend4php provides much of the functions needed, the Tripal Galaxy module does provide a few other functions that help with Tripal integration.

The API documentation is provided in two forms: an overview and a list of functions. The overview provides a written description for using the API and the list of functions provides the list of all functions, their specific arguments and return values.

8.1 Tripal Galaxy API Overview

Here code examples and brief descriptions are provided for common applications of the Tripal Galaxy API. Not all arguments for all API functions are thoroughly described. Please see the *Tripal Galaxy API Functions* page for specific details.

8.1.1 Adding a Galaxy Server

The Tripal Galaxy module allows a site administrator to add a new Galaxy Server to Tripal using an online web interface. However, you can programmatically add a new Galaxy server using the `tripal_galaxy_add_galaxy` function. For example:

```
// Get the current user.  
global $user;
```

(continues on next page)

(continued from previous page)

```
// Prepare the information needed for the server.
$data = array(
  'servername' => 'Use Galaxy Public',
  'description' => 'The public server provided by the Galaxy Project',
  'url' => 'https://usegalaxy.org/',
  'username' => 'fakeuser',
  'api_key' => '48eead1213e5dd6d1876811b38b66e51',
  'uid' => $user->uid,
);

// Add the Galaxy server.
$galaxy = tripal_galaxy_add_galaxy($data);
```

A use case when you may want to programmatically add a new Galaxy server is if you want to provide your own interface for allowing others to add new servers besides the site admin.

8.1.2 Get a List of Galaxies

You can retrieve a list of all Galaxy servers known to Tripal by calling the `tripal_galaxy_get_galaxies` function.

```
// Get the list of galaxies.
$galaxies = tripal_galaxy_get_galaxies();

// Iterate through that list to perform actions on each, or retrieve
// information.
foreach ($galaxies as $galaxy) {

  // Get the galaxy ID
  $galaxy_id = $galaxy->galaxy_id;

  // Do stuff here.
}
```

8.1.3 Connect to Galaxy

Before any communication between Tripal and Galaxy can happen a connection between the two must be made. Blend4php will allow you to connect directly to a Galaxy instance if you know the URL. However, it is recommended to use the Tripal API to make the connection.

If the site admin has already added a Galaxy server using the web interface then you can connect to the Galaxy server using its internal ID. The `tripal_galaxy_get_connection` function can be used. It returns an instance of a `GalaxyInstance`.

For example:

```
// Connect to the Galaxy instance using a galaxy ID we retrieved in an
// earlier step.
$galaxy = tripal_galaxy_get_connection($galaxy_id);

// It is always good to make sure we got a valid connection. If we didn't
// then we can retrieve any connection errors using the getError() member
// function.
if (!$galaxy) {
```

(continues on next page)

(continued from previous page)

```

$error = $galaxy->getError();
drupal_set_message('Could not connect to Galaxy server. ' . $error['message'],
↳ 'error');
}

```

In the example code above, the `tripal_galaxy_get_connection` function returns an instance of the `blend4php` `GalaxyInstance` class that we named `$galaxy`. This instance is used by all other functions used by `blend4php` to interact with Galaxy. To learn more about this object and the functionality it provides please see the [blend4php_docs](#).

8.1.4 Test if a Galaxy server is accessible.

If a Galaxy server is not accessible no actions can be performed including workflow submissions, status updates, or results display. A server may not be accessible if it is offline. You can check the status of a galaxy workflow with the `tripal_galaxy_test_connection` function.

```

$is_connected = tripal_galaxy_test_connection(['galaxy_id' => $galaxy_id]);
if (!$is_connected) {
    // Do something here.
}

```

8.1.5 Add a Workflow to Tripal

The Tripal GALaxy module does not allow a user to create Galaxy workflows. Workflows should always be created on the Galaxy interface. Once a workflow is created it can be added to Tripal. A site administrator can add new workflows to Tripal using the web interface. However, you can programmatically add a new workflow using the `tripal_galaxy_add_workflow` function. To do this you must know have the ID of the galaxy server and the name or `workflow_id` of the workflow on the Galaxy server. You can obtain the name or `workflow_id` from the Galaxy server by:

1. Using the `blend4php` `GalaxyWorkflows::index()` function and examining the resulting array.
2. Directly retrieving a JSON array of results from the Galaxy server's RESTful API services available at the URL `api/workflows` (e.g. <https://usegalaxy.org/api/workflows>). You can view results in a web browser if you have a plugin for viewing JSON arrays. You must first login to Galaxy before using the RESTful web services.

```

$values = [
    'workflow_id' => 'ebfb8f50c6abde6d',
];
$workflow = tripal_galaxy_add_workflow($galaxy_id, $values, TRUE);

```

The `tripal_galaxy_add_workflow` will connect to the remote Galaxy server, retrieve the workflow information and add it to Tripal. If you want your site visitors to be able to submit the workflow for execution set the last argument to `TRUE` to create the web form for the workflow.

8.1.6 Prepare a Workflow for Execution

If a web form was created for a workflow then your site visitors can submit the workflow for execution using the web interface to provide input data and settings. However, you can submit a workflow programmatically. To do this, use the `tripal_galaxy_init_submission` function to first prepare the submission. For this function you must specify the workflow object and the user that is submitting it. The workflow object can be retrieved using the `tripal_galaxy_get_workflows` by providing values to uniquely select it.

```
// Get the current user.
global $user;

// Provide the values to uniquely find the workflow.
$values = [
  'workflow_id' => 'ebfb8f50c6abde6d',
  'galaxy_id' => $galaxy_id,
]

// Find the workflow using the values. This function always returns an
// array of workflows that match the criteria. By providing the workflow_id
// and the galaxy_id it should only ever match one workflow.
$workflows = tripal_galaxy_get_workflows($values);
$workflow = $workflows[0];

// We can now initialize the workflow submission.
$ssid = tripal_galaxy_init_submission($workflow, $user);

// The submission ID will uniquely identify this submission. Next get the
// submission object.
$submission = tripal_galaxy_get_submission($ssid);
```

The workflow is now ready to be invoked.

8.1.7 Get a History Name

Before we can invoke a workflow we need to understand Galaxy histories. All data in Galaxy is housed in a data collection referred to as a “history”. Before workflows can be executed, input data must be placed in a history, and after workflow execution, resulting data is found in the history. For more information about histories in Galaxy you can view the [histories tutorial page](#).

When Tripal Galaxy invokes a workflow within Galaxy it will ensure that each invocation uses a unique history with a unique name. By default Tripal Galaxy module uses a naming schema for histories: *TG-[UID]-[WID]-[SID]-[Date]*.

Where

- [UID] is the user ID of the Drupal user who is submitting/submitted the workflow
- [WID] is the Tripal Galaxy module’s ID for the workflow
- [SID] is the Tripal Galaxy submission ID for the workflow submission and
- [Date] is the date that the submission was made.

For example the following is history name that follows this scheme: TG-1-53-19-2018_10_03_09:31:02

Before invoking a workflow you will need to create the history and you should follow the Tripal naming scheme to name that history. To retrieve the name for any history use the `tripal_galaxy_get_history_name` function.

```
// Retrieve the $submission object using a known submission ID.
$submission = tripal_galaxy_get_submission($ssid);

// Get the history name.
$history_name = tripal_galaxy_get_history_name($submission);
```

8.1.8 Create a History and Upload Files

Before we invoke a workflow we must ensure that the required input data files are in a history on the remote Galaxy server. For loading files from your local Tripal site into Galaxy use the `tripal_galaxy_upload_file` function. This function expects that any files you upload to Galaxy are also known by Drupal, therefore you must use the [Drupal 7 File API](#). The following shows an example for how to do this:

```
// The current user owns the file.
global $user;

// The Drupal File API uses Streams (i.e public://, private://, etc) which
// point to files within the Drupal directory heirarchy.
$uri = 'public://reads_1.fastq';

// Create a new file object.
$file = new stdClass();
$file->uri = $uri;
$file->filename = 'reads_1.fastq';
$file->filemime = file_get_mimetype($uri);
$file->uid = $user->uid;
$file->status = FILE_STATUS_PERMANENT;
$file = file_save($file);
$fid = $file->fid;

// We don't want the file to disappear when Drupal performs it's cleaning so
// we have to tell Drupal that the file is being used. See the Drupal
// documentation for the meaning of the input arguments.
file_usage_add($file, 'my_module', 'workflow1_reads', $sid);
```

Now that Drupal is aware of the file there are a few additional steps before we can upload it to Galaxy. First, we must ensure that a history exists on the remote Galaxy server. Remember, that all workflow submissions use a unique history name. We should use that history name to create the history. The history is created using the `tripal_galaxy_create_history` function.

```
// Retrieve the $submission object using a known submission ID.
$submission = tripal_galaxy_get_submission($sid);

// Get the history name.
$history_name = tripal_galaxy_get_history_name($submission);
$history = tripal_galaxy_create_history($galaxy, $history_name);
if ($history === FALSE) {
  $error = $galaxy->getError();
  throw new Exception($error['message']);
}
```

Next, we need the current contents of the history. If the history did not exist then the history should be empty. However, the same workflow submission can be invoked multiple times so we need to get the contents of the history to pass into the `tripal_galaxy_upload_file` function.

```
// Get the history contents so we don't upload the same file more than once
// in the event that this invocation occurs more than once.
$ghistory_contents = new GalaxyHistoryContents($galaxy);
$history_contents = $ghistory_contents->index(['history_id' => $history['id']]);
if ($history_contents === FALSE) {
  $error = $galaxy->getError();
  throw new Exception($error['message']);
}
```

Now we have sufficient information to upload our file to Galaxy. Our file is known to Drupal, we have a history, and we have the history contents.

```
// Upload the file to Galaxy.
$galaxy_file = tripal_galaxy_upload_file($galaxy, $fid, $history['id'], $history_
↳contents);
```

The returned `$galaxy_file` variable contains information about the file on the galaxy server. We will need this later when invoking the workflow.

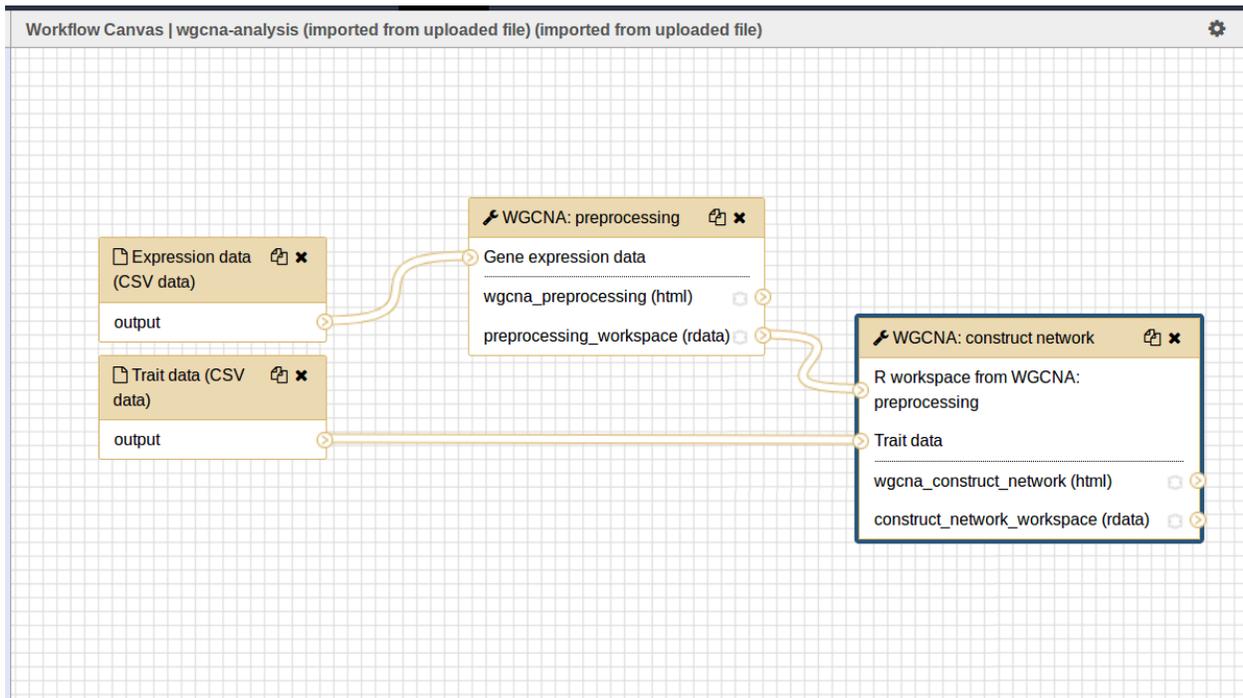
8.1.9 Invoke a Workflow

Workflows are the multistep process through which data is submitted, analysed, processed, and then results provided. Workflows are created in Galaxy, they cannot be created in Tripal Galaxy.

For more information on creating and editing workflows please see the [online tutorial](#).

Before invoking the workflow all data files need to be uploaded to a Galaxy history. Examples for how to do this were shown in the previous section. To invoking a workflow use the `tripal_galaxy_invoke_workflow` function. It requires two complex arrays: `$parameters` and `$inputs`. These provide the tool settings for the workflow and the input files needed to start the workflow. Construction of these arrays will require a good understanding of each workflow's needs.

For example, consider the following workflow. It has four steps, with two initial steps providing only input data and the other two using actual analytical tools.



The first two steps require input files. Therefore, we will indicate where these are using the `$inputs` array. This is an associative array containing the files used by the workflow and their location in the Galaxy server. An input array may look like the following:

```
$inputs = [
  [0] => [
```

(continues on next page)

(continued from previous page)

```
[id] => 70eec96181a992f8,
[src] => hda,
],
[1] => [
  [id] => 8317ee2b0d0f62d9,
  [src] => hda,
],
];
```

The first-level keys are numeric (e.g. 0 and 1). These indicate the steps in the workflow that require input. For each there is an `id` and `src` key. The `id` is the Galaxy ID for the file. We can retrieve this id from the `$galaxy_file` array we retrieved above when we called the `tripal_galaxy_upload_file` function.

The `src` indicates where the file can be found. The value `hda` indicates the file is in the History Dataset Association (HDA) which is the default history storage location. See the `tripal_galaxy_invoke_workflow` function definition on the [Tripal Galaxy API Functions](#) page for a more in-depth meaning of the `$inputs` array structure.

Similarly, the `$parameters` argument is an associative array whose first-level keys are the numeric index for the steps in the workflow and the settings for the tool at each step are provided.

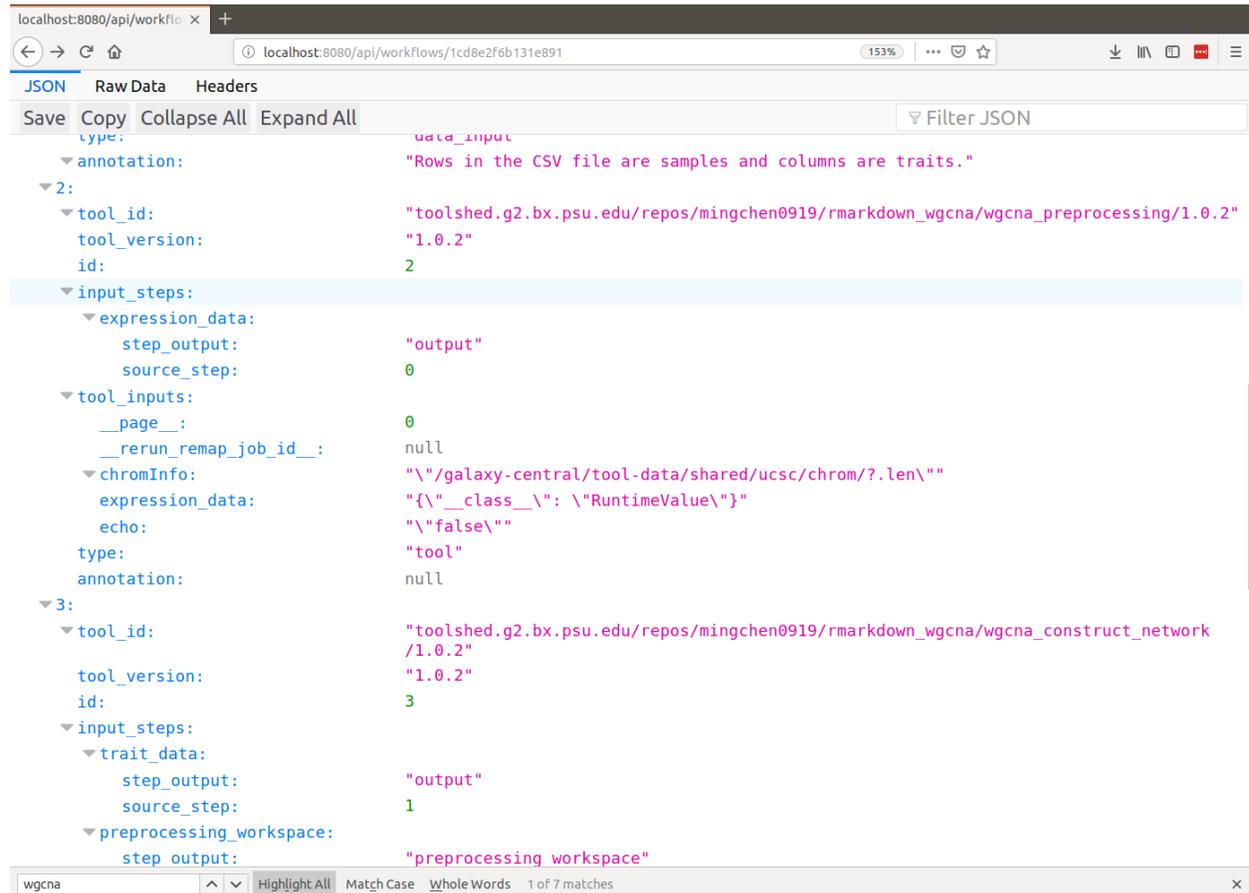
```
$parameters = [
  [0] => [
    [Data File] => 70eec96181a992f8,
  ],
  [1] => [
    [Data File] => 8317ee2b0d0f62d9,
  ],
  [2] => [
    [expression_data] => [
      [step_output] => output,
      [source_step] => 0,
    ],
    [echo] => no,
  ],
  [3] => [
    [preprocessing_workspace] => [
      [step_output] => preprocessing_workspace,
      [source_step] => 2,
    ],
    [height_cut] => 15,
    [trait_data] => [
      [step_output] => output,
      [source_step] => 1,
    ],
    [echo] => no,
  ],
];
```

Note that for steps 0 and 1 the setting name is `Data File` and the value is the file ID used in the `$inputs` array. For all other steps, the settings are specified.

Note: Every workflow is different and as such the `$input` and `$parameters` arrays will be different from those shown in the examples above.

It can be very challenging to determine the proper elements and structure of the `$parameters` array. For example, the following view of the JSON for a workflow shows how a workflow is structured in Galaxy. To build

the `$parameters` array one would need to find them in the `tool_inputs` for each step and be sure to use the `input_steps` info for inputs that use values from previous steps. It can be quite complicated.



To simplify this process, you can retrieve a pre-populated `$parameters` array and change the values as needed. You can do so by calling the `tripal_galaxy_get_workflow_defaults` function:

```
$parameters = tripal_galaxy_get_workflow_defaults($galaxy, $workflow_id)
```

Now we have everything we need to invoke a workflow:

1. A submission record
2. A history on the remote Galaxy ID
3. Uploaded files into the history
4. An `$inputs` array indicating where input files are found in the galaxy server.
5. A `$parameters` array containing all of the tool settings.

```
// Call the Tripal Galaxy API function to invoke this workflow.
tripal_galaxy_invoke_workflow($galaxy, $submission, $parameters,
    $input_datasets, $history);
```

An email will be sent to the user indicating that the workflow has been submitted for execution.

Note: It can take quite a while to upload files and invoke workflows because of the time needed to upload large files. Consider uploading files and invoking workflows using the Tripal Job system. This way you can submit a job to invoke

the workflow, your script will return immediately rather than wait, and the next time the Tripal Job queue is launched your workflow will be invoked.

8.1.10 Check the status of a workflow submission

By invoking a workflow you indicate to the Galaxy server that the workflow should be executed. You must then wait for Galaxy to execute your workflow submission once resources become available. Additionally, depending on the amount of data and the tools used the workflow can take a long time to complete. You therefore should periodically check the status of your workflow. You can do so by calling the `tripal_galaxy_check_submission_status` function to update the status of the submitted workflow. A workflow submission on Tripal Galaxy will have one of 4 statuses: Waiting, Submitted, Completed or Error.

```
// First ask Tripal to update the status of the submission.
tripal_galaxy_check_submission_status($submission->sid);

// Next retrieve the submission again to check the new status.
$submission = tripal_galaxy_get_submission($submission->sid);
$status = $submission->status;
```

8.1.11 Retrieving Results from Galaxy

There are two primary cases in which result files from an executed workflow can be made available to end-users. First, you may wish to retrieve the files so that they can be analyzed for visualization or for use in other workflows. This results in the files being stored locally to the Drupal server, and retrieved files will count towards Tripal's user file quota. Second, you may simply wish to provide a download link to these files but not download them to the server. This way, the files can remain on the Galaxy server, the files don't count towards the Tripal user quota, yet users can still retrieve them from within your application.

8.1.11.1 Retrieving a Dataset

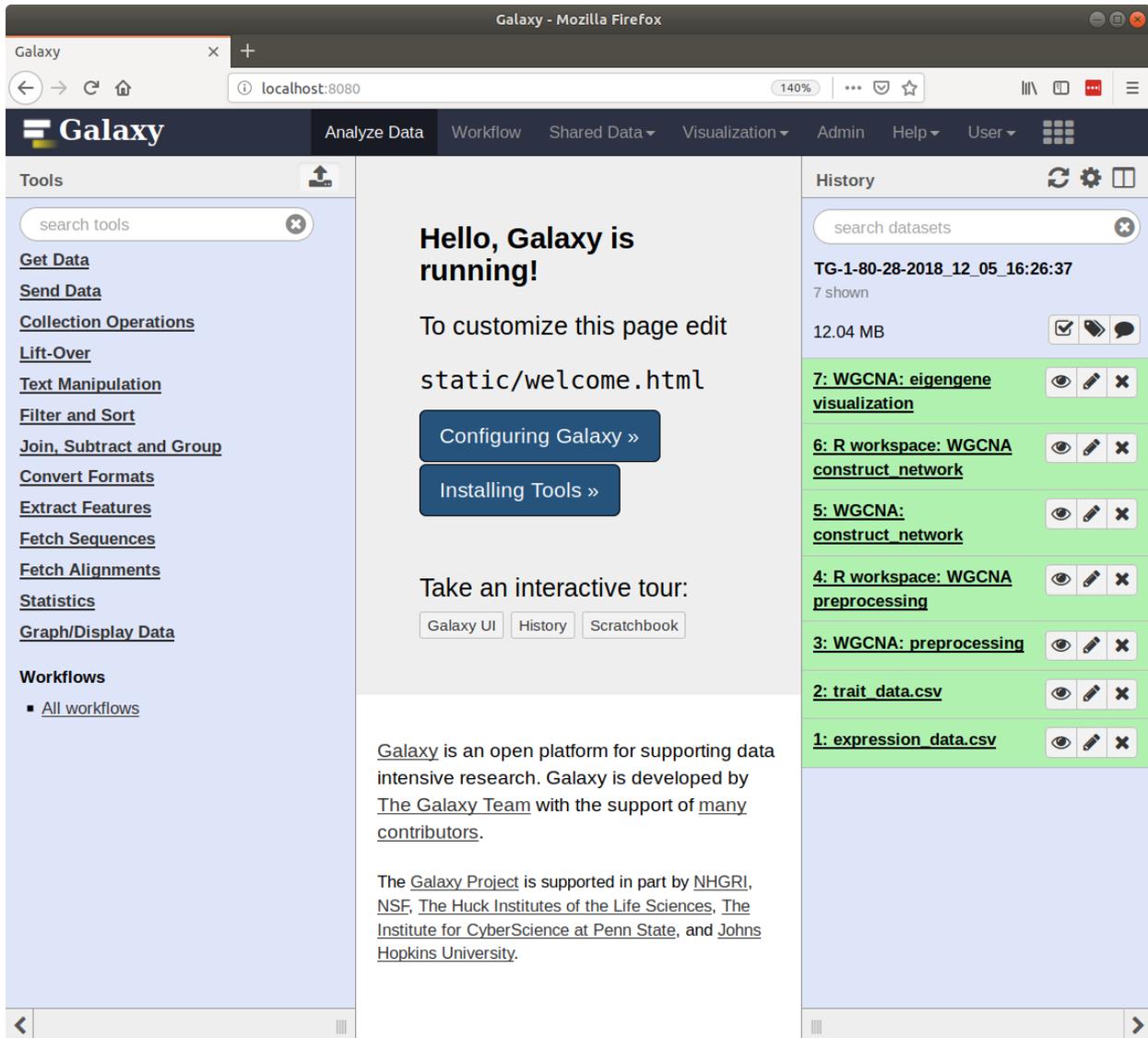
For either option, you must first know the dataset ID. Result files are stored within the history on the remote Galaxy server. The quickest way to find the dataset ID is to retrieve a list of datasets using the `tripal_galaxy_get_datasets` function.

```
$datasets = tripal_galaxy_get_datasets($submission);
```

The `$datasets` variable now contains a list of datasets for the workflow invocation.

Note: You must be sure that the workflow has completed before calling the `tripal_galaxy_get_datasets()` function.

Suppose we have a history that appears as follows on the Galaxy server.



Note in the image above the right sidebar with the 7 datasets. When the history contents are retrieved for such a history it will be an array describing each dataset. Below is an example element in the `$datasets` array for the 'WGCNA: eigengene visualization' item.

```
[
  [accessible] => 1
  [type_id] => dataset-40876639881ca029
  [file_name] => /galaxy/database/files/000/dataset_58.dat
  [resubmitted] =>
  [create_time] => 2018-12-06T00:35:05.157630
  [creating_job] => 52e496b945151ee8
  [dataset_id] => 40876639881ca029
  [file_size] => 1427219
  [file_ext] => html
  [id] => 40876639881ca029
  [misc_info] => ''
  [hda_ldda] => hda
  [download_url] => /api/histories/0c5ffef6d88a1e97/contents/40876639881ca029/display
```

(continues on next page)

(continued from previous page)

```
[state] => ok
[display_types] => []
[display_apps] => []
[metadata_dbkey] => ?
[type] => file
[misc_blurb] => 1.4 MB
[peek] => <table cellpadding="0" cellspacing="3"><tr><td>HTML file</td></tr></table>
[update_time] => 2018-12-06T00:43:13.749954
[data_type] => galaxy.datatypes.text.Html
[tags] => []
[deleted] =>
[history_id] => 0c5ffef6d88a1e97
[meta_files] => []
[genome_build] => ?
[hid] => 7
[model_class] => HistoryDatasetAssociation
[metadata_data_lines] => 150
[annotation] =>
[permissions] => [
  [access] => []
  [manage] => []
]
[history_content_type] => dataset
[name] => WGCNA: eigengene visualization
[extension] => html
[visible] => 1
[url] => /api/histories/0c5ffef6d88a1e97/contents/40876639881ca029
[uuid] => c5178a2d-eef5-4218-bb09-6984e56a49e3
[visualizations] => [
  [0] => [
    [embeddable] =>
    [href] => /plugins/visualizations/charts/show?dataset_id=40876639881ca029
    [html] => Charts
    [target] => galaxy_main
  ]
]
[rerunnable] => 1
[purged] =>
[api_type] => file
]
```

By iterating over all of the datasets we can find the dataset id of the file we are interested in.

```
$dataset_id = NULL;
$datasets = tripal_galaxy_get_datasets($submission);
foreach ($datasets as $dataset) {
  if ($dataset['name'] == 'WGCNA: eigengene visualization') {
    $dataset_id = $dataset['id'];
  }
}
```

Note: The Tripal Galaxy module does have a setting for how long files can stay on a remote Galaxy server before they are automatically removed. See the [Additional Settings](#) document for more details.

8.1.11.2 Option 1: Download Files to the Server

Once the workflow has completed you can retrieve any resulting files onto the Drupal server. This is useful if you need to re-use the results in other workflows or if you want to make those results available to users on their Profile pages (i.e. via the Files link).

To retrieve the file simply call the `tripal_galaxy_download_file` function.

```
$file = tripal_galaxy_download_file($submission, $dataset_id, $uid);
```

This function downloads the file to the Drupal server and registers it with Drupal as a file managed by Tripal. This ensures that the file will be associated with the user ID specified by the `$uid` argument. Therefore, the `$file` variable returned will be a Drupal File object, and the file will count against the user's quota and show up on their profile page.

Warning: Be sure that you have sufficient storage space when downloading files. To help with this, Tripal provides a user quota system to which downloaded are applied. However, be cautious of how much space is being utilized. You can check how much space a user is currently consuming using the `tripal_get_user_usage()` function. If space is a concern you probably want to check the user's usage before invoking the workflow.

8.1.11.3 Option 2: Providing Download Links for Users

Once the workflow has completed you may want to provide links for the end-user to view or download results from the workflow. As opposed to option #1, this does not result in files being downloaded to the Drupal server and will not affect the user's file quota. To create a link for either viewing or downloading you should use the `tripal_galaxy_get_proxy_url`. This function returns a URL you can use in your application for viewing or downloading. For example, the following retrieves a URL for downloading a dataset:

```
global $user;

// Get the dataset.
$dataset = tripal_galaxy_get_dataset($submission, $dataset_id);

// Now get the Proxy URL and create the link.
$proxy_url = tripal_galaxy_get_proxy_url($submission, $dataset, $uid, 'download');
$link = l($dataset['name'], $proxy_url) . ' (' . $dataset['file_size'] . ')';
```

The `$link` variable now contains an HTML formatted link for downloading the file. It also lists the file size next to the link. You can use this variable anywhere that you create content for a page in your application. Notice that the last argument to the `tripal_galaxy_get_proxy_url()` function is the string `download`. This is the action that the proxy URL should perform.

Similarly to retrieve a URL for viewing the dataset:

```
global $user;

// Get the dataset.
$dataset = tripal_galaxy_get_dataset($submission, $dataset_id);

// Any files that are smaller than 1MB can be shown in a browser.
if ($dataset['file_size'] < pow(10, 6)) {
  $proxy_url = tripal_galaxy_get_proxy_url($submission, $dataset, $uid, 'viewer');
  $link = l($dataset['name'], $proxy_url);
}
```

(continues on next page)

(continued from previous page)

```
else {
  $link = 'Result file is too large to view. Please download.';
}
```

Note in the call to `tripal_galaxy_get_proxy_url` the last argument is `viewer`. With this action, when the link is clicked the file will be shown on a results page provided by the Tripal Galaxy module. The page displays results in an HTML IFrame. If you prefer to show results in a stand-alone page use `viewer-full` as the last argument.

8.2 Tripal Galaxy API Functions

group `tripal_galaxy_api`

programming interface (API) for integrating Tripal and Galaxy. It uses the `blend4php` library to communicate with a remote Galaxy instances but ensures that other developers who want to integrate with the features of the Tripal Galaxy module can do so.

Functions

`tripal_galaxy_add_galaxy(array $values)`

Adds a new Galaxy server to Tripal.

Return object The newly created Galaxy object or FALSE on error. If the record already exists the object is returned and no duplicated record is added.

Parameters

- `$values`: An associative array containing the following key/value pairs: - `servername`: A human-readable name for the Galaxy server. The servername must be unique and not already present. - `description`: (optional) a description about this server. - `url`: The full URL for the server (e.g. <https://usegalaxy.org/>) - `username`: The name of the user to connect as. - `api_key`: The API key of the user. This allows Tripal to access the Galaxy server's RESTful API services. - `uid`: The Drupal User ID who created this record.

`tripal_galaxy_add_workflow(int $galaxy_id, array $values, bool $create_webform = TRUE)`

Adds a remote Galaxy Workflow to Tripal.

Return object A workflow object or FALSE on error. If the workflow has already been added, then the workflow object is returned and no duplicate record is added.

Parameters

- `$galaxy_id`: An ID of the galaxy server.
- `$values`: An associative array used to specify the workflow. The workflow can be identified using the Galaxy workflow ID provided using the `'workflow_id'` key or using the workflow name provided using the `'workflow_name'` key.
- `$create_webform`: If TRUE, creates a webform for the end-user to submit this workflow using the Tripal website. It will automatically create a history for this workflow on the remote Galaxy server as well.

`tripal_galaxy_check_submission_status(int $sid = NULL, bool $force = FALSE)`

Checks and updates the status of a Galaxy workflow.

Return bool Returns TRUE on successful checking of the status, FALSE if a problem occurred.

Parameters

- `$sid`: The submission ID of the workflow.
- `$force`: If a workflow submission is already completed this function will quickly return and not check the status again. Setting the `$force` argument to `TRUE` will force the function to check the status.

tripal_galaxy_create_history(*GalaxyInstance* `$galaxy`, *string* `$history_name`)
Creates a history on the Galaxy server.

Return array A history array for the created history.

Parameters

- `$galaxy`:
- `$history_name`:

tripal_galaxy_delete_expired_histories ()
Deletes expired histories.

Walks through the `tripal_galaxy_workflow_submission` table and deletes any workflows older in days than specified in the `tripal_galaxy_history_days_limit` system variable.

tripal_galaxy_delete_remote_history(*int* `$galaxy_id`, *string* `$history_name`)
Deletes a single remote history from the remote galaxy server.

Return bool `TRUE` if the deletion was successful, `FALSE` otherwise.

Parameters

- `$galaxy_id`: A unique ID for the galaxy server. If this is provided no other arguments are needed.
- `$history_name`: The name of the history to retrieve. If the history doesn't exist then it will be created.

tripal_galaxy_download_file(*stdClass* `$submission`, *array* `$dataset`, *int* `$uid`)
Downloads a file from a Galaxy history.

Downloads a single file from a history. The file will be stored in the user's Tripal space and count towards their quota. If the file has already been downloaded it will be retrieved again and overwrite any existing file.

Return object Returns a Drupal File object.

Parameters

- `$submission`: A submission object as obtained by the `tripal_galaxy_get_submission()` function.
- `$dataset`: An array describing the dataset as returned by the `tripal_galaxy_get_dataset()` or `tripal_galaxy_get_datasets()` functions.
- `$uid`: The user that should own this file after it is downloaded.

Exceptions

- `Exception`:

tripal_galaxy_get_active_submissions ()
Retrieves a list of all workflow submissions that have not completed.

Return array An array of submission objects.

tripal_galaxy_get_connection(int \$galaxy_id)

Retrieves a GalaxyInstance objects using a galaxy_id.

Return GalaxyInstance A galaxyInstance object or FALSE on error.

Parameters

- \$galaxy_id: The ID of a galaxy server.

tripal_galaxy_get_dataset(stdClass \$submission, \$dataset_id)

Retrieves a single datasets for a workflow invocation.

This function returns a single datasets as reported by the remote Galaxy server. The key/value pairs of the array are dependent on the Galaxy server version.

Return array An array of datasets as reporte by the Remote Galaxy server.

Parameters

- \$submission: An object for the submission as returned by the tripal_galaxy_get_submission() function.
- The: ID of the dataset to retrieve.

Exceptions

- Exception:

tripal_galaxy_get_datasets(stdClass \$submission)

Retrieves a list of datasets for a workflow invocation.

This function returns an array of datasets as reported by the remote Galaxy server. The key/value pairs of the array are dependent on the Galaxy server version.

Note, this function may take a few seconds to complete as it must communicate with the remote Galaxy server to retrieve dataset information.

Return array An array of datasets as reporte by the Remote Galaxy server.

Parameters

- \$submission: An object for the submission as returned by the tripal_galaxy_get_submission() function.

Exceptions

- Exception:

tripal_galaxy_get_files_dir()

Returns the URI where the Tripal Galaxy module stores files.

This function also ensures that the path exists by creating it.

Return string|bool A Drupal URI indicating the location where Galaxy files are housed. Returns FALSE if the location does not exist or cannot be created.

tripal_galaxy_get_galaxies()

Retrieves a list of Galaxy servers that are known to Tripal.

Return array An associative array of galaxy server objects. Note, the returned objects are different from the GalaxyInstance objects provided by blend4php. These objects house the information that Tripal maintains about each of these servers.

tripal_galaxy_get_galaxy(int \$galaxy_id)

Retrieves a Galaxy object.

Return object A galaxy object Note, the returned object is different from the GalaxyInstance object provided by blend4php. This objects house the information that Tripal maintains about each the servers. Returns FALSE on error.

Parameters

- `$galaxy_id`: The ID of the galaxy server to retrieve. A list of all galxy servers known to Tripal can be retrieved using the *tripal_galaxy_get_galaxies()* function.

tripal_galaxy_get_history(GalaxyInstance \$galaxy, string \$history_name)

Retrieves a history by name from Galaxy.

You must first create the history using *tripal_galaxy_create_history()* if it does not already exist.

Return array|bool A history array for the specified history or FALSE if the history could not be found.

Parameters

- `$galaxy`: A GalaxyInstance object.
- `$history_name`: The name of the history to retrieve. If the history doesn't exist then it will be created.

tripal_galaxy_get_history_name(\$submission)

Constructs the history name for a given submission.

Return string The history name.

Parameters

- `$submission`: A submission object that contains the `galaxy_workflow_id`, `sid`, and `submit_date` properties.

tripal_galaxy_get_submission(int \$sid)

Retrieves a workflow submission object using the submission ID.

The returned submission object includes all information about the submission. If the submission has been invoked then the 'errors' element will have more information about the run including the history information, the history name, and the state of the history_contents from the last time the status of the submission was checked. If the status of the submission is 'Completed' then the history_contents should contain all the history contents.

Return object An object containing the submission information.

Parameters

- `$sid`: The submission ID of the workflow.

tripal_galaxy_get_user_submissions(int \$userid)

Retrieve all submission IDs for a user.

Return array A list of submission ids.

Parameters

- `$userid`: User id number.

tripal_galaxy_get_workflow_defaults(GalaxyInstance \$galaxy, string \$workflow_id)

Retrieves an array of settings used by the workflow.

This is a helpful function to help a developer understand how the \$parameters array should be structured when passing into the *tripal_galaxy_invoke_workflow()* function.

Return boolean An array compatible with the \$parameters argument for the *tripal_galaxy_invoke_workflow()*. Values are populated with appropriate defaults. Returns FALSE if the workflow settings could not be retrieved.

Parameters

- \$galaxy: A GalaxyInstance object.
- \$workflow_id: The ID of the workflow to retrieve settings for. This is the Galaxy ID for the workflow not Tripal's internal workflow ID.

tripal_galaxy_get_workflows(array \$values = [])

Retrieves a list of workflows integrated with Tripal.

The returned workflows can be filtered using a set of matching criteria given by the \$values argument. This list only includes workflows that have been integrated with Tripal and not the list of workflows available on a remote galaxy server. Use *blend4php* functions to retrieve a list of workflows available on a remote Galaxy server.

Return array An array of workflow objects.

Parameters

- \$values: An associative array used to find workflows. The following keys are supported: - galaxy_id: finds all workflows that match the given galaxy_id. - id: finds the workflow with this specific workflow ID. - workflow_id: the ID of the workflow on the remote Galaxy instance. - name: finds the workflow with a given name. Note: the workflow name is not guaranteed to be unique. - status: finds all workflows whose status matches the value provided. Any combination of the keys can be used.

tripal_galaxy_init_submission(\$workflow, \$user)

Initializes a new workflow submission record.

This function creates the record for the submission.

Return int|bool The submission ID on success, FALSE on failure.

Parameters

- \$workflow: An workflow object as generated by the *tripal_galaxy_get_workflow()* function.
- \$user: The Drupal User object. This is the user who owns the submission.

tripal_galaxy_invoke_workflow(GalaxyInstance \$galaxy, \$submission, array \$parameters,

Invokes all submitted workflows that are in the 'Waiting' state.

This function can be called by the tripal Job system hence the \$job argument. For Tripal v2 the job_id is passed, for Tripal v3 a job object is passed so we'll handle both cases.

Parameters

- \$galaxy: An instance of a GalaxyInstance object.
- \$submission: A Galaxy workflow submission object. This object can be retrieved using the *tripal_galaxy_get_submission()* function.

- `$parameters`: A mapping of tool parameters that are non-datasets parameters. The map must be in the following format:

```
[
  {step_id_or_UUID} => [
    {param_name} => {value}
  ],
  {step_id_or_UUID} => [
    {param_name} =>
      {value}
  ]
];
```

- `$inputs`: An array of file inputs. These files should already be uploaded to the history on the Galaxy server. This array contains a mapping of workflow inputs to datasets and dataset collections. The datasets source can be a LibraryDatasetDatasetAssociation (ldda), LibraryDataset (ld), HistoryDatasetAssociation (hda), or HistoryDatasetCollectionAssociation (hdca). The map must be in the following format.

```
[
  {step index} => [
    'id' => {encoded dataset ID},
    'src' => {'ldda'|'ld'|'hda'|'hdca'},
  ],
];
```

The id's are data set IDs and can be found using the data set class's `index()` function. The data set must be present in a history, and the data set 'state' must be 'ok' and 'deleted' must be set to FALSE. The {step index} is the numeric value of the step in the workflow where the file is used.

- `$history`: A history record as returned by the function `tripal_galaxy_get_history()`.

tripal_galaxy_test_connection(array \$connect)

Tests if a Galaxy server is accessible.

Return bool Returns TRUE if accessible. FALSE otherwise. A Drupal message is also provided that indicates if the test was successful.

Parameters

- `$connect`: An array of the following: - `galaxy_id`: A unique ID for the galaxy server. If this is provided no other arguments are needed. - `host`: The DNS hostname of the galaxy server. - `port`: The TCP port for the server. - `use_https`: Set to TRUE if the server uses HTTPS. If the 'galaxy_id' is provided then no other values are needed. Use the host, port and use_https arguments if testing connection to a server that has not yet been added.

tripal_galaxy_upload_file(\$galaxy, int \$fid, string \$history_id, array \$history_contents)

Uploads a file to a given history on Galaxy.

Return array An array of the dataset details from Galaxy for the uploaded file.

Parameters

- `$galaxy`: An instance of a Galaxy server object.
- `$fid`: The Drupal managed file ID.
- `$history_id`: The history ID.
- `$history_contents`: The Galaxy history contents array as returned by the `GalaxyHistoryContents::index` function.

Exceptions

- Exception:

T

`tripal_galaxy_delete_expired_histories`
(C++ *function*), 54

`tripal_galaxy_get_active_submissions`
(C++ *function*), 54

`tripal_galaxy_get_files_dir` (C++ *function*),
55

`tripal_galaxy_get_galaxies` (C++ *function*),
55