Galaxy-on-StarCluster Documentation

Release 1.0.0

Eric A. McDonald

October 23, 2013

CONTENTS

1 "Quick" Guide for Experts **3**

2 Introductory Discussion **5**

3 MIT StarCluster **7**
- 3.1 StarCluster Installation . . . 7
- 3.2 StarCluster Configuration . . . 7
- 3.3 Using StarCluster . . . 12

4 Galaxy Installation and Configuration **15**
- 4.1 Installation of Galaxy . . . 15
- 4.2 Configuration and Testing . . . 15

5 Galaxy Tool Development and Deployment **19**
- 5.1 Preparing a Tool . . . 19
- 5.2 Development of New Tools . . . 20
- 5.3 Deployment of New Tools . . . 21

6 Integrating Tools from Outside Galaxy **23**
- 6.1 Some Unsorted Notes . . . 23

7 Indices and tables **25**

Contents:

CHAPTER

ONE

"QUICK" GUIDE FOR EXPERTS

1. If you have questions about why a certain procedure is being recommended, then please refer to the introduction or other relevant chapter of the manual for more details.
2. Configure a new MIT StarCluster template.
   - You only need a single node.
   - Create a new EBS volume and assign it to your cluster template.
3. Log into an EC2 instance created with your StarCluster template.
4. Create a Python virtual environment somewhere (maybe at the mount point for the EBS volume which you created).
5. Create a `galaxy-python` directory at your EBS volume mount point or in a directory beneath it.
6. Symlink the `python` executable in your virtual environment into the `galaxy-python` directory.
7. Clone http://bitbucket.org/galaxy/galaxy-dist into a sibling directory of your `galaxy-python` directory.
8. Follow the instructions at *Configuring the Main Galaxy Server*.
9. Follow the instructions at *Testing the Main Galaxy Server*.
10. Make a mental note of the Galaxy serer's unruliness mentioned in *Stopping the Main Galaxy Server*.
11. Follow the instructions at *Configuring the Galaxy Tool Shed Server*.
12. Follow the instructions at *Testing the Galaxy Tool Shed Server*.
13. Follow the instructions at *Securing the Galaxy Servers*.
14. Follow the instructions at *Connecting the Galaxy Tool Shed to the Main Server*.
15. Follow the instructions at *Testing the Tool Shed Connection*.
16. Acquire the sources for the tool you are trying to install. (Maybe dump them into a sibling directory of `galaxy-dist`.)
17. Configure and build any binaries which you need.
18. If the sources are Pythonically-packaged, then consider building a `setuptools`-style egg package, as Galaxy can be configured to use eggs on launch. If you need help, then please take at *Building Eggs*.
19. Use the instructions at *Creating Tool Categories* and *Adding a New Tool Repository*, if you are not already familiar with the operation of a Galaxy tool shed.
    - Clone the http://github.com/ged-lab/sc-galaxy.git repo and look in the `tools` subdirectory for an example.
20. To learn how to find and edit an uploaded repo on the server, please see *Editing a Tool Repository*.

21. Use the instructions at *Refreshing Tool Shed Records*, if you need to know how to update the Galaxy toolshed's view of your changes.
22. Use the instructions at *Deployment of New Tools*, if you need to know how to transport your tool from the toolshed server to the main server.

CHAPTER

TWO

INTRODUCTORY DISCUSSION

The purpose of this document is to provide a detailed set of instructions on how to deploy a Galaxy server and develop new tools for it. There are a number of ways to perform these tasks. The various options are here discussed.

Because Galaxy hosts Web services on TCP/IP ports which may be firewalled, one criterion is that the user, who is running a Galaxy server, must be able to expose those ports to the outside world without the involvement of a third party. This precludes the installation of Galaxy at most HPC centers and even on the resources provided by most campuses – by regular users anyway. Cloud computing provides a way around this limitation and the focus of this document is upon setting up a Galaxy server in a cloud environment.

Many different cloud providers exist. As Galaxy is a Unix-centric tool, cloud providers which cater to Windows, such as Microsoft Azure, can be effectively be ruled out. Within the Linux world, Amazon EC2 (Elastic Compute Cloud) is currently the most popular cloud provider. Thus, we will examine the list of options available for using Amazon as the cloud provider.

The Galaxy development team provides AMIs (Amazon Machine Images) which come with a cloud-enabled Galaxy already installed. A tool, called *CloudMan*, is integrated into these AMIs and allows a Galaxy administrator to allocate clusters of Galaxy workers, monitor these clusters, and dynamically grow and shrink them. While this is some fairly powerful functionality to have available, especially the dynamic resizing, this approach does have some drawbacks. These drawbacks are:

- AMI configuration and initialization occur via Amazon's EC2 web console. There are no command-line tools available for these tasks.
- Cluster management occurs via the CloudMan web interface. There might not be any command-line tools available for this.
- The administrator must supply his or her EC2 credentials via some metadata associated with the initial virtual machine (VM) instance of the AMI. While these credentials are transmitted securely, it is still a dubious security practice to store them as part of the machine metadata. For example, it may give a dishonest Amazon employee access to information which he or she would not otherwise be able to view. (The metadata fields are clear text; none of your credentials are stored in hashed or encrypted form.)

The STAR (Software Tools for Academics and Researchers) program at MIT provides a wonderful command-line tool called *StarCluster*. This tool has a number of subcommands, which can be used to create, manage, login to, stop, and destroy clusters of one or more VM instances on EC2. Although StarCluster does not natively support Galaxy (yet), its value as an extremely convenient, general purpose EC2 management tool cannot be denied. A more detailed examination of this tool will follow.

If you are interested in maintaining a cluster of Galaxy workers in the long term, then it is conceivable that CloudMan may be the better option for you. Of course, in such a case, you may wish to provide some dedicated hardware for this purpose and avoid the pay-as-you-go scheme associated with cloud computing. This document, however, focuses on running a Galaxy server for the purpose of developing tools. For this purpose, using MIT StarCluster as a basis is eminently useful and more detailed discussion will proceed along those lines.

As a final note, it should be mentioned that the Galaxy developers do provide a public-facing development system, whereat you can upload and test your own tools. While this has the convenience of being ready-made, the rather open permissions model leaves a lot to be desired. Rogue tools can be uploaded to this public-facing system. Developers are also at the mercy of the maintenance schedule for this system, which may prove disruptive to development. Also, the act of development and testing is not as efficient as if one had behind-the-scenes access to his or her own Galaxy server, as we will see below.

CHAPTER

THREE

MIT STARCLUSTER

3.1 StarCluster Installation

Acquiring and configuring MIT StarCluster are both quite simple tasks. StarCluster is distributed as a Python package, called `StarCluster`. Thus, the usual Python package management tools, such as `easy_install` or `pip`, may be used to install it. For example:

```
easy_install StarCluster
```

You may also find that your particular Linux distribution has a package for StarCluster. For example, on Gentoo Linux, the package is named `dev-python/starcluster`. Using your Linux distribution's package management system is usually preferable, unless you are installing an experimental version of StarCluster to a non-standard location. Your distribution's package management system will help ensure that all necessary dependencies are properly built and working and make it easier to perform automatic upgrades in the future.

Please see the StarCluster intallation web page for more details: http://star.mit.edu/cluster/docs/latest/installation.html

3.2 StarCluster Configuration

3.2.1 Generating the Configuration File

When you run StarCluster for the first time, it will create a configuration subdirectory for you in your home directory, if such subdirectory does not already exist. This subdirectory is named `.starcluster` by default. To tell StarCluster to create this subdirectory, you can use something like the `starcluster ls` subcommand:

```
$ starcluster ls
StarCluster - (http://web.mit.edu/starcluster) (v. 0.93.3)
Software Tools for Academics and Researchers (STAR)
Please submit bug reports to starcluster@mit.edu

!!! ERROR - config file /home/em/.starcluster/config does not exist

Options:
--------
[1] Show the StarCluster config template
[2] Write config template to /home/em/.starcluster/config
[q] Quit

If everything was successful, then you should see that the ``.starcluster``
subdirectory contains a file named ``config``. This file was generated from a
template shipped with StarCluster.
```

```
The configuration file will need some tweaking before you can create a VM
instance suitable for use with Galaxy.

Please enter your selection: 2

>>> Config template written to /home/em/.starcluster/config
>>> Please customize the config template
```

You can verify the existence of your configuration subdirectory and its contents by listing them:

```
$ ls -l ~/.starcluster
total 12
-rw-r--r-- 1 em em 12106 Oct 16 10:19 config
drwxr-xr-x 2 em em    22 Oct 16 10:14 logs
drwxr-xr-x 2 em em     6 Oct 16 10:14 plugins
```

Note the presence of a file named `config`. This file was generated from a template that comes with the StarCluster software. By default, this file will not suffice for providing VM instances which are suitable for running Galaxy. We will need to edit it.

3.2.2 Editing the Global Configuration

The configuration file is monolithic; it contains your EC2 credentials, your desired cluster configurations, your desired firewall settings, your desired persistent data volumes, etc.... StarCluster provides an *includes* mechanism, which allows you to break out portions of this file into separate files. This is highly recommended, at least in the case of your EC2 credentials, to help ensure that you do not accidentally share them with someone else if you helpfully give a copy of your configuration to a colleague or friend. You may also use the inclusion mechanism simply to break the file up into more manageable pieces, if you desire.

At the top of the configuration file, there is a `global` section. Here, we are interested in changing two variables, `DEFAULT_TEMPLATE` and `INCLUDE`.

```
[global]
DEFAULT_TEMPLATE=solo
INCLUDE=~/...PRIVATE/starcluster-auth
```

We will create a new template named `solo` in a later section. (You can call it whatever you want, but `solo` will be used for the purpose of these instructions.) We will also create a separate directory and a separate file therein which will hold your EC2 credentials. To perform the first part of this latter task now, use the `mkdir` command with a secure access mode setting:

```
$ mkdir -m 700 ~/...PRIVATE
```

3.2.3 Securing Your EC2 Credentials

Back in the configuration file, you might notice that the next two sections concern your credentials and other information about your SSH keys. This is the very information which we want to exclude from the main configuration and place into a separate secure file. So, the following procedure is recommended:

- Copy your configuration file over to your private, secure directory and name it `starcluster-auth`. Secure the file as well.

  ```
  $ install -m 600 ~/.starcluster/config ~/...PRIVATE/starcluster-auth
  ```

  Or, if the `install` command is not available to you:

```
$ cp ~/.starcluster/config ~/...PRIVATE/starcluster-auth
$ chmod 600 ~/...PRIVATE/starcluster-auth
```

- Edit `~/...PRIVATE/starcluster-auth`, removing all sections except the `aws info` section and any `key` sections.
- Edit `~/...PRIVATE/starcluster-auth`, filling out the `aws info` section and any `key` sections as appropriate. You will likely need to consult your Amazon EC2 console web page to gather some of the needed information. (This is a one-time activity. You should not need to use the web page after this.)
- Edit `~/.starcluster/config`, removing the `aws info` section and any `key` sections.

3.2.4 Defining a New Cluster

Next, we want to create a new `cluster` section in `~/.starcluster/config`. Place the following in the file (preferably in the same area as the other `cluster` sections):

```
[cluster solo]
KEYNAME             = amazon-ged-ctb
CLUSTER_SIZE        = 1
CLUSTER_USER        = sgeadmin
CLUSTER_SHELL       = bash
NODE_IMAGE_ID       = ami-999d49f0
NODE_INSTANCE_TYPE  = m1.large
AVAILABILITY_ZONE   = us-east-1a
PERMISSIONS         = ssh, galaxy-http-main, galaxy-http-shed
# VOLUMES             = galaxy-devel
```

Warning: Please adjust the `KEYNAME` variable to match the name of a `key` section which you defined in your `~/...PRIVATE/starcluster-auth` file.

Warning: If you are using a different EC2 availability zone than `us-east-1a`, then please alter the `AVAILABILITY_ZONE` variable accordingly.

Note: The StarCluster project may release new AMIs from time to time. Please consult the output of the `starcluster listpublic` subcommand for the current list.

You can adjust the `NODE_INSTANCE_TYPE` to fit your purposes and possibly save some money. A list of node instance types can be found in the `cluster smallcluster` section of the configuration file.

Warning: You will notice that the `cluster smallcluster` section is not commented out. Please either comment out this section or adjust the `KEYNAME` variable to the same value which you used in the `cluster solo` section.

The definition of the `solo` cluster has references to some firewall permissions that are not part of the configuration file by default. Likewise, there is a reference to a volume which needs to be defined. We will handle the definition of these things next.

3.2.5 Setting Firewall Permissions

If you scroll through the configuration file, you will eventually come across some `permission` sections. You will notice that there are a couple of `permission http` sections, a `permission https` section, and a `permission`

ssh section. These are all commented out by default.

Remove the comments from in front of the permission ssh section. (You may leave the CIDR_IP assignment commented out, if you wish. You might want to do this if you intend to perform your development from a wide variety of locations with disparate IPs. If you intend to only develop from machines on the premises of a particular organization, such as a university, you can tighten down security by removing the comment from in front of the CIDR_IP assigment. CIDR stands for Classless Inter-Domain Routing. The short story is that every network on the Internet has a *network number*. The network number is calculated from an *IP address* and a *netmask* (*network number mask*). You can get the necessary information from your organization's network administrators, if you don't already know it and are uncomfortable crawling through Internet network registries (whois databases). You may also make an educated, overly-narrow guesstimate of your network number by taking your computer's IP address (try hostname -i or /sbin/ifconfig), replacing the last of the four octets with 0, and placing a /24 after it. Or, you can just use your computer's current IP address and place a /32 after it.)

For the purposes of running Galaxy, you do not need the standard web server ports (80 and 443). By default, Galaxy is configured to use ports 8080 and 9009. Port 8080 is, however, well-known as it gets used for a variety of auxiliary web services, particularly administration tools of various sorts. This makes it a target for intrusion attempts. If you plan on leaving your Galaxy server running for any length of time on EC2, then you will either want to configure Galaxy to use a different port number or you will want to set a CIDR_IP variable for it. (You can do both, of course.) These instructions will use ports 9007 and 9009. (If you find this choice of numbers to be odd, no one will dispute that.) Create the following two sections in your configuration file, using whichever port numbers you have decided upon (the port numbers should be greater than 1023):

```
# non-standard - 8080 is standard but intrusion attempts probe there
[permission galaxy-http-main]
PROTOCOL = tcp
FROM_PORT = 9007
TO_PORT = 9007
# CIDR_IP = <your_ip>/32

[permission galaxy-http-shed]
PROTOCOL = tcp
FROM_PORT = 9009
TO_PORT = 9009
# CIDR_IP = <your_ip>/32
```

3.2.6 Defining a Galaxy Volume

We need a place to persistently store Galaxy. Amazon provides persistent storage volumes known as *EBS Volumes*. The EBS (Elastic Block Store) service is another pay-as-you-go service like EC2. Since the size of the volume factors in to how much you pay, you want to optimize between price and having enough capacity for your needs. A reasonable starting configuration, which should accommodate software you might wish to install plus some sizable test data, might be 20 gigabytes.

Before an EBS volume can be referenced, it must exist. To create it, we will use:

```
$ starcluster cv -s 20 us-east-1a
StarCluster - (http://web.mit.edu/starcluster) (v. 0.93.3)
Software Tools for Academics and Researchers (STAR)
Please submit bug reports to starcluster@mit.edu

>>> No keypair specified, picking one from config...
>>> Using keypair: amazon-ged-ctb
>>> Creating security group @sc-volumecreator...
>>> No instance in group @sc-volumecreator for zone us-east-1a, launching one now.
Reservation:r-1156d977
>>> Waiting for volume host to come up... (updating every 30s)
```

```
>>> Waiting for all nodes to be in a 'running' state...
1/1 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Waiting for SSH to come up on all nodes...
1/1 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Waiting for cluster to come up took 1.551 mins
>>> Checking for required remote commands...
>>> Creating 20GB volume in zone us-east-1a
>>> New volume id: vol-a0cf0add
>>> Waiting for new volume to become 'available'...
>>> Attaching volume vol-a0cf0add to instance i-58fbf225...
>>> Formatting volume...

>>> Detaching volume vol-a0cf0add from volhost-us-east-1a
>>> Terminating node: volhost-us-east-1a (i-58fbf225)
>>> Waiting for cluster to terminate...
>>> Removing @sc-volumecreator security group
```

Note that if anything in the above process fails, then it means that you have something wrong with your configuration file and need to go back and carefully review the previous steps. If nothing failed, then congratulations, you just successfully created an EC2 instance and EBS volume and terminated the EC2 instance without using Amazon's EC2 console.

Warning: The Amazon zone 'us-east-1a' is sometimes "full" and blocked from starting new machines. If you get an error that says something to this effect, then modify `AVAILABILITY_ZONE` in your `.starcluster/config` file to be something else, like 'us-east-1d'. You will also need to modify the `starcluster cv` command to use 'us-east-1d'.

Now, we need to edit the configuration file again. First, stop by the `cluster solo` section which you created earlier. Remove the comment from in front of the `VOLUMES` variable. After that, find the area where the `volume` sections reside and create a `volume galaxy-devel` section:

```
[volume galaxy-devel]
VOLUME_ID       = vol-a0cf0add
MOUNT_PATH      = /opt/sw/Galaxy
```

Warning: You need to set the `VOLUME_ID` variable to match the volume ID given in the output from your `starcluster cv` command. (In the above example, the volume ID is `vol-a0cf0add`.)

You can adjust the `MOUNT_PATH` variable to whatever you want. If the path to the mount point does not yet exist in the instance's file system, then it will be automatically created by StarCluster before attempting to mount the volume there.

3.2.7 Miscellany

Your configuration of StarCluster should now be complete. To learn more about configuring this software, it is recommended that you read the helpful comments in the configuration file and that you consult http://star.mit.edu/cluster/docs/latest/manual/configuration.html

3.3 Using StarCluster

3.3.1 Getting Help

The `starcluster help` command is your friend. Note that you can get detailed help on subcommands by supplying the subcommand name as an additional argument. For example:

```
$ starcluster help cv
StarCluster - (http://web.mit.edu/starcluster) (v. 0.93.3)
Software Tools for Academics and Researchers (STAR)
Please submit bug reports to starcluster@mit.edu

Usage: createvolume [options] <volume_size> <volume_zone>

    Create a new EBS volume for use with StarCluster
```

The StarCluster manual is a useful resource, especially since it contains details that the built-in help does not provide. (E.g., the manual tells you what units the `<volume_size>` parameter uses for the `createvolume` subcommand.) The manual can be found on the web at http://star.mit.edu/cluster/docs/latest/contents.html

3.3.2 Short Example

To create an EC2 instance with the above configuration, the following command can be used, for example:

```
$ starcluster start -c solo galaxy
StarCluster - (http://web.mit.edu/starcluster) (v. 0.93.3)
Software Tools for Academics and Researchers (STAR)
Please submit bug reports to starcluster@mit.edu

>>> Validating cluster template settings...
>>> Cluster template settings are valid
>>> Starting cluster...
>>> Launching a 1-node cluster...
>>> Creating security group @sc-galaxy...
>>> Opening tcp port range 9009-9009 for CIDR 0.0.0.0/0
>>> Opening tcp port range 9007-9007 for CIDR 0.0.0.0/0
Reservation:r-abd1ded2
>>> Waiting for cluster to come up... (updating every 30s)

...

>>> Configuring cluster took 0.925 mins
>>> Starting cluster took 2.065 mins
```

To log into the newly created EC2 instance once it has finished starting, the following command can be used, for example:

```
$ starcluster sshmaster galaxy
StarCluster - (http://web.mit.edu/starcluster) (v. 0.93.3)
Software Tools for Academics and Researchers (STAR)
Please submit bug reports to starcluster@mit.edu

The authenticity of host 'ec2-54-234-37-131.compute-1.amazonaws.com
(54.234.37.131)' can't be established.
ECDSA key fingerprint is 4e:2c:7e:7c:08:1e:ed:df:7f:5d:4a:91:8b:ab:22:22.
Are you sure you want to continue connecting (yes/no)? yes
```

```
Warning: Permanently added
'ec2-54-234-37-131.compute-1.amazonaws.com,54.234.37.131' (ECDSA) to the
list of known hosts.

...
```

CHAPTER

FOUR

GALAXY INSTALLATION AND CONFIGURATION

4.1 Installation of Galaxy

4.1.1 Setting up the Python Environment

```
$ cd /opt/sw/Galaxy
$ mkdir MERCURIAL
$ cd MERCURIAL
$ virtualenv --no-site-packages PYTHON-ENV
New python executable in PYTHON-ENV/bin/python
Installing distribute...................done.
Installing pip..............done.
$ mkdir galaxy-python
$ ln -s $PWD/PYTHON-ENV/bin/python galaxy-python
$ . PYTHON-ENV/bin/activate
```

4.1.2 Cloning the Galaxy Repository

```
$ hg clone https://bitbucket.org/galaxy/galaxy-dist/
destination directory: galaxy-dist
requesting all changes
adding changesets
adding manifests
adding file changes
added 7829 changesets with 30613 changes to 6311 files
updating to branch default
3926 files updated, 0 files merged, 0 files removed, 0 files unresolved
```

4.2 Configuration and Testing

4.2.1 Configuring the Main Galaxy Server

```
$ mkdir galaxy-IMPORT
$ cd galaxy-dist
```

```
$ cp universe_wsgi.ini{.sample,}
$ cp tool_shed_wsgi.ini{.sample,}
```

Edit `universe_wsgi.ini`. Configure the `server:main` section, adding new variable assignments as follows:

```
[server:main]
port = 9007
host = 0.0.0.0
```

If you chose a different port number than 9007 when you were setting up the `permissions` sections of your StarCluster configuration file, then you need to reflect that in the `port` variable in the Galaxy configuration.

Configure the `app:main` section, adding new variable assignments as follows:

```
[app:main]
library_import_dir = /opt/sw/Galaxy/MERCURIAL/galaxy-IMPORT
allow_library_path_paste = True
```

4.2.2 Testing the Main Galaxy Server

Start the Galaxy server with the `run.sh` script:

```
$ ./run.sh
... bunch of text ...
Starting server in PID 3139.
serving on 0.0.0.0:9007 view at http://127.0.0.1:9007
```

On the computer where you have the StarCluster tools installed, you can get the DNS name of your Galaxy server. (Replace the `sctest` argument in the command pipeline below with the name of your EC2 cluster.)

```
$ starcluster lc | grep -A10 sctest | grep 'master running'
master running i-060c047b ec2-23-20-53-82.compute-1.amazonaws.com
```

From this information form, a HTTP URL with the DNS name of the Galaxy server and port 9007. For example, `http://ec2-23-20-53-82.compute-1.amazonaws.com:9007`. Point your favorite web browser at this URL. If all has gone according to plan, then you should see be able to bask in the glow of a running Galaxy instance. But, don't get too cocky because there are still plenty of opportunities for failure in the work that remains to be done.

While you are at the Galaxy web page, click on the `User` dropdown menu in the menu bar along the top of the web page and select the `Register` menu item. Setup an account for yourself. You will want to do this so that you can add yourself to the administrators list for the server.

After you have setup an account for yourself, you will want to stop the Galaxy server so that you can continue configuring it. (Technically, a `--reload` argument can be supplied to the run `run.sh` script which allows the server to monitor configuration files for changes and reload its state on-the-fly. But, it is still safer to perform a shutdown and restart of it.)

4.2.3 Stopping the Main Galaxy Server

Unfortunately, the Galaxy server is not guaranteed to stop via a keyboard interrupt. You may have noticed that near the end of the server initialization output to your terminal that the PID of the server was provided. You can use this PID to kill the server process. If you cannot find the process identifier, then it is fairly safe to use a command such as `pkill` or `killall` with `python` as the argument. For example:

```
$ kill 3139
```

Or:

```
$ pkill python
```

4.2.4 Configuring the Galaxy Tool Shed Server

Edit the `tool_shed_wsgi.ini` file. In the `server:main` section, ensure that the following assigment is made:

```
[server:main]
host = 0.0.0.0
```

In the `app:main` section, ensure that the `database_file` assignment is uncommented:

```
[app:main]
database_file = database/community.sqlite
```

4.2.5 Testing the Galaxy Tool Shed Server

Start the Galaxy tool shed server with the `run_community.sh` script.

```
$ sh run_tool_shed.sh
```

Compose a HTTP URL for your web browser, as you did to access the main Galaxy server, using port `9009` or whatever port number you chose instead. (E.g., `http://ec2-23-20-53-82.compute-1.amazonaws.com:9009`.) Point your web browser at this URL and check out your newly erected tool shed (if everything went well). Shiny, isn't it?

While you are at the Galaxy tool shed web page, click on the `User` dropdown menu in the menu bar along the top of the web page and select the `Register` menu item. Setup an account for yourself. You will want to do this so that you can add yourself to the administrators list for the server.

After you have setup an account for yourself, it is recommended that you stop the tool shed server.

4.2.6 Securing the Galaxy Servers

Assuming that you have created an account for yourself on both the main server and the tool shed server, you are now ready to elevate yourself to an administrative role.

Edit the `universe_wsgi.ini` and `community_wsgi.ini` files and provide the following assignments in their `app:main` sections.

```
[app:main]
admin_users = em@msu.edu
require_login = True
allow_user_creation = False
```

Note: In the above, the value of the `admin_users` assignment should be changed to the email address which you provided when you registered your Galaxy account.

4.2.7 Connecting the Galaxy Tool Shed to the Main Server

Once you have developed tools in your Galaxy tool shed, you may wish to transport those tools to your main server to perform proper integration testing. Galaxy provides a transport mechanism for this purpose.

```
$ mkdir ../galaxy-SHED
```

Edit `tool_sheds_conf.xml` and add a new `tool_shed` entry, replacing the value of the `url` assignment with the HTTP URL for your tool shed server:

```
<?xml version="1.0"?>
<tool_sheds>
    <tool_shed name="Galaxy main tool shed" url="http://toolshed.g2.bx.psu.edu/"/>
    <tool_shed name="Galaxy test tool shed" url="http://testtoolshed.g2.bx.psu.edu/"/>
    <tool_shed name="My Tool Shed" url="http://ec2-23-20-53-82.compute-1.amazonaws.com:9009/"/>
</tool_sheds>
```

Note: You cannot use `localhost` in the HTTP URL for your toolshed server. This is possibly because the URL is used on a web page rendered on your client web browser and Galaxy processes the submitted URL in such a way that `localhost` would be considered relative to your client rather than the Galaxy server. Unfortunately, the limitation forces you to update this configuration file every time you are working with a new instance or else break the connection between the main server and the tool shed server.

Edit `shed_tool_conf.xml` (not the same as the above file - don't blame me for the confusing nomenclature) and change the value of the `tool_path` assignment to the path to your tool shed directory (`../galaxy-SHED` is used in this documentation).

```
<?xml version="1.0"?>
<toolbox tool_path="../galaxy-SHED">
</toolbox>
```

Edit `universe_wsgi.ini` and add the following assignment to the `app:main` section:

```
[app:main]
tool_config_file = tool_conf.xml,shed_tool_conf.xml
```

4.2.8 Testing the Tool Shed Connection

Fire up the Galaxy main and tool shed servers with their respective scripts, `run.sh` and `run_community.sh`. Once and if they are up, login to the main Galaxy server. Click on the `Admin` menu in the menu bar across the top of the web page. A sidebar with various administrative tasks will present itself. Find the `Tool sheds` section and click on the `Search and browse tool sheds` link. You should see a button labeled `My Tool Shed`. Click on this button and select `Browse valid repositories` from the ensuing dropdown menu. If this task is completed without error, then you should be good to go.

Onward to actually developing and deploying tools....

4.2.9 Resuming Galaxy on a New EC2 Instance

From time to time, you may need to stop your EC2 instance and then start a new one at a later point. Some steps will need to be done to restart the Galaxy servers. There are recapitulated below:

```
cd /opt/sw/Galaxy/MERCURIAL/galaxy-dist
. ../PYTHON-ENV/bin/activate
bash run.sh &
bash run_community.sh &
```

CHAPTER

FIVE

GALAXY TOOL DEVELOPMENT AND DEPLOYMENT

5.1 Preparing a Tool

What follows is a specific example of how to prepare a tool for inclusion in Galaxy.

5.1.1 Acquiring the Sources

As a staging area for the sources you wish to acquire, it is recommended you move out of the Galaxy installation directory.

```
$ cd /opt/sw/Galaxy
```

In this example, two Git repositories are cloned.

```
$ git clone http://github.com/ged-lab/screed.git
Cloning into screed...
remote: Counting objects: 1821, done.
remote: Compressing objects: 100% (661/661), done.
remote: Total 1821 (delta 1192), reused 1758 (delta 1136)
Receiving objects: 100% (1821/1821), 372.63 KiB, done.
Resolving deltas: 100% (1192/1192), done.
$ git clone http://github.com/ged-lab/khmer.git
Cloning into khmer...
remote: Counting objects: 7450, done.
remote: Compressing objects: 100% (2363/2363), done.
remote: Total 7450 (delta 5197), reused 7262 (delta 5025)
Receiving objects: 100% (7450/7450), 62.33 MiB | 724 KiB/s, done.
Resolving deltas: 100% (5197/5197), done.
```

5.1.2 Building Eggs

If you have a `distutils`-based Python package, then you can apply a patch similar to following to use the `setuptools` package, which should have been installed into your Python virtual environment.

```
diff --git a/python/setup.py b/python/setup.py
index 4ff584a..94326c7 100644
--- a/python/setup.py
+++ b/python/setup.py
```

```
@@ -1,4 +1,7 @@
-from distutils.core import setup, Extension
+try:
+    from setuptools import setup, Extension
+except ImportError:
+    from distutils.core import setup, Extension

 # the c++ extension module (needs to be linked in with ktable.o ...)
 extension_mod = Extension("khmer._khmermodule",
```

Build the egg(s) for your Python package. For example:

```
$ (cd screed && python setup.py bdist_egg)
$ (cd khmer && make && cd python && python setup.py bdist_egg)
```

Yes, `python setup.py install` could be used to install the packages into the Python virtual environment which we are using. However, the intention is to simulate how Galaxy would use Python eggs in a production environment. For this purpose, we want to have eggs on hand.

Copy the eggs into Galaxy's egg basket. For example:

```
$ cp khmer/python/dist/khmer-0.4-py2.7-linux-x86_64.egg galaxy-dist/eggs
$ cp screed/dist/screed-0.7-py2.7.egg galaxy-dist/eggs
```

5.2 Development of New Tools

5.2.1 Creating Tool Categories

Before you can place any tools into your tool shed, you need to provide one or more categories in which the tools can be placed. To do this, login to the tool shed server. Click on the `Admin` menu in the menu bar across the top of the web page. A sidebar with various administrative tasks will present itself. Find the `Categories` section and click on the `Manage categories` link. You should see a `Categories` frame appear. In the upper right-hand corner of the frame is a button labeled `Add new category`. Create categories, have fun.

5.2.2 Adding a New Tool Repository

Once you have created your desired categories, click on the `Repositories` menu in the menu bar along the top of the web page. A sidebar with various repository-related actions should appear. Find the `Available Actions` section and click on the `Create new repository` link. Fill out the required metadata and then click the `Save` button. You should get a message indicating that your repository has been created. In the upper, right-hand corner of the resulting frame, you should notice a button, labeled `Upload files to repository`. Click on this button. You can upload a plain Unix tar file or a gzip- or bzip2-compressed variant thereof.

5.2.3 Editing a Tool Repository

Your first tool shed's repository is stored at `database/community_files/000/repo_1`. For your convenience, you can edit the repository's files directly in that location. When you are finished, you should commit your changes as the user who created the repository.

```
$ hg ci -u eric -m "Fix directory structure."
```

Note: In the above, you will want to replace the argument to the `-u` option with the display name of your Galaxy user account. When you registered for the account, this was the fourth field that you were asked to fill out; it is not your email address.

5.2.4 Refreshing Tool Shed Records

After you commit changes, you need to make the Galaxy tool shed database aware of the changes. To do this, you need to be viewing your repository. In the upper, right corner of the frame, there is a `Respository Actions` button. Click on this button and select the `Reset all respository metadata` menu itme from the dropdown menu. (This is not as scary as it sounds.)

5.3 Deployment of New Tools

5.3.1 Staging a Tool to the Main Galaxy Server

Once you have developed something that you are satisfied with, you may wish to stage it over to the main Galaxy server. To do this, switch over to your web page for the main server and go into the administrator tasks. Select the `Browse repositories` menu item from the dropdown menu for your tool shed. You should see the tool repository that you added in the category to which you added it. Click on the button bearing the name of your tool repository and select the `Preview and install` menu item. In the upper, right corner of the resulting frame, click on the `Install to local Galaxy` button.

That's it. If everything worked, then your tool should appear under the category which you selected in the sidebar associated with the `Analyze Data` menu.

CHAPTER

SIX

INTEGRATING TOOLS FROM OUTSIDE GALAXY

6.1 Some Unsorted Notes

6.1.1 Velvet

If you plan to build Oases, you may wish to symlink the Velvet directory. For example:

```
ln -s velvet_1.2.08 velvet
```

6.1.2 Trinity

If you plan to build Trinity, you will probably want to place the directory as a sibling of the `galaxy-dist` directory and symlink it as follows:

```
ln -s trinityrnaseq_r2012-10-05 trinityrnaseq
```

The example tools expect to find `Trinity.pl` there.

CHAPTER

SEVEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*