PANDA Project Documentation

Release 1.0.0

PANDA Project

August 03, 2012

CONTENTS

1 About **1**

2 What is PANDA? **3**

3 Setup **5**
- 3.1 Local development & testing . . . 5
- 3.2 Production deployment . . . 7

4 Configuration **13**
- 4.1 Configuring DNS . . . 13
- 4.2 Configuring Email . . . 14
- 4.3 Performance . . . 15
- 4.4 Configuring SSL . . . 16

5 Administration **19**
- 5.1 User management . . . 19
- 5.2 Managing dataset categories . . . 19
- 5.3 Managing user API keys . . . 20

6 Server maintenance **21**
- 6.1 Connecting with SSH . . . 21
- 6.2 Systems Administration (Ops) . . . 23
- 6.3 Backing up your PANDA . . . 25
- 6.4 Adding more storage to your PANDA . . . 26
- 6.5 Upgrading your PANDA . . . 27

7 Extending PANDA **31**
- 7.1 API Import Tutorial . . . 31
- 7.2 API Documentation . . . 34

8 Authors **51**

9 License **53**

10 Changelog **55**
- 10.1 1.0.0 . . . 55
- 10.2 0.2.0 . . . 56
- 10.3 0.1.4 . . . 57
- 10.4 0.1.3 . . . 57
- 10.5 0.1.2 . . . 58

10.6 0.1.1 58
10.7 0.1.0 59

11 Search this documentation **61**

CHAPTER

ONE

ABOUT

PANDA is your newsroom data appliance. It provides a place for you to store data, search it and share it with the rest of your newsroom.

The PANDA Project is 2011 Knight News Challenge winner. The team would like to thank the Knight Foundation for their generous support of free and open source software for newsrooms.

Logistical support and fiscal agency for PANDA have been provided by Investigative Reporters and Editors. Our sincere thanks to them helping make it a reality.

Note: **Are you a reporter?** Documentation for users can be found at the PANDA Project Cookbook.

CHAPTER

TWO

WHAT IS PANDA?

PANDA is:

- A place for journalists to store data.
- A search engine for your news data.
- A private archive of your newsworthy datasets.
- *Extensible*.
- *Self-hosted*.

PANDA is *not*:

- A publishing system.
- A universal backend for your newsapps.
- A platform for data visualizations.
- A highly structured datastore.
- Software as a Service.

See our `Frequently Asked Questions (FAQ)` for much more.

CHAPTER

THREE

SETUP

3.1 Local development & testing

3.1.1 Install basic requirements

Use the tools appropriate to your operating system to install the following packages. For OSX you can use Homebrew. For Ubuntu you can use Apt.

- pip
- virtualenv
- virtualenvwrapper
- PostgreSQL
- Mercurial (hg)

3.1.2 Set up PANDA

This script will setup the complete application, *except* for Solr. Be sure to read the comments, as some steps require opening additional terminals:

```
# Get source and requirements
git clone git://github.com/pandaproject/panda.git
cd panda
mkvirtualenv --no-site-packages panda
pip install -r requirements.txt

# Create log directory
sudo mkdir /var/log/panda
sudo chown $USER /var/log/panda

# Create data directories
mkdir /tmp/panda
mkdir /tmp/panda_exports

# Enter "panda" when prompted for password
createuser -d -R -S -P panda
createdb -O panda panda
python manage.py syncdb --noinput
python manage.py migrate --noinput
python manage.py loaddata panda/fixtures/init_panda.json
```

```
python manage.py loaddata panda/fixtures/test_users.json

# Start PANDA
python manage.py runserver
```

Open a new terminal in the PANDA directory and enter:

```
# Start the task queue
workon panda
python manage.py celeryd
```

Open *another* terminal in the PANDA directory and enter:

```
# Run a local email server
workon panda
fab local_email
```

3.1.3 Set up Solr

Installing Solr can be tricky and will vary quite a bit depending on your operating system. The following will get you up and running on OSX Lion, using Homebrew. If you've just started the PANDA server, open a new terminal in the PANDA directory and enter these commands:

```
# Get into the env
workon panda

# Install solr 3.4.0
brew update
brew install solr

# Create Solr home directory
sudo mkdir /var/solr
sudo chown $USER /var/solr

# This command will install all Solr configuration
fab local_reset_solr

# To start Solr
fab local_solr
```

3.1.4 Checking your PANDA

Your PANDA should now be running at:

```
http://localhost:8000/
```

A PANDA installed locally will not run through the normal setup mode procudure. Instead, two default users will be created.

You can login using the default user credentials:

```
Username: user@pandaproject.net
Password: user
```

Or the default administrator credentials:

```
Username: panda@pandaproject.net
Password: panda
```

3.1.5 Running Python unit tests

To run the unit tests, start Solr and execute the test runner, like so:

```
# Ensure you are in the PANDA source directory and your virtualenv is active
# You may need to customize the fabfile so it can find your Solr installation.
fab local_solr

# Quite a bit of output will be printed to the screen.
# Wait until you see something like
# 2011-11-02 14:15:54.061:INFO::Started SocketConnector@0.0.0.0:8983
# Then, open another terminal and change to your PANDA source directory.
workon panda
python manage.py test panda
```

3.1.6 Running Javascript unit tests

Running the Javascript unit tests requires that the application server is running (to render the the JST template map). To run the Javascript tests, first start the test server with `python manage.py runserver`, then open the file `client/static/js/SpecRunner.html` in your browser (e.g. `file://localhost/Users/onyxfish/src/panda/client/static/js/SpecRunner.html`.

3.2 Production deployment

We provide documentation for two different ways of installing PANDA for production use. Non-advanced users and those who desire the simplest setup possible should install their PANDA on Amazon's EC2 service. This is the best-supported option.

Those who are concerned about the security of storing their data in the cloud or who have access to dedicated hardware may choose to install PANDA on their own server. Although we make every effort to support this, we can not offer any guarantee of compatibility.

Have more questions about hosting? See the Hosting questions of our FAQ.

I am...

3.2.1 Installing on Amazon EC2

Step 1. Register for EC2

If you don't already have an Amazon Web Services account, you will need to register for one and set up your credentials. Visit the EC2 homepage and follow the "Sign Up Now" link.

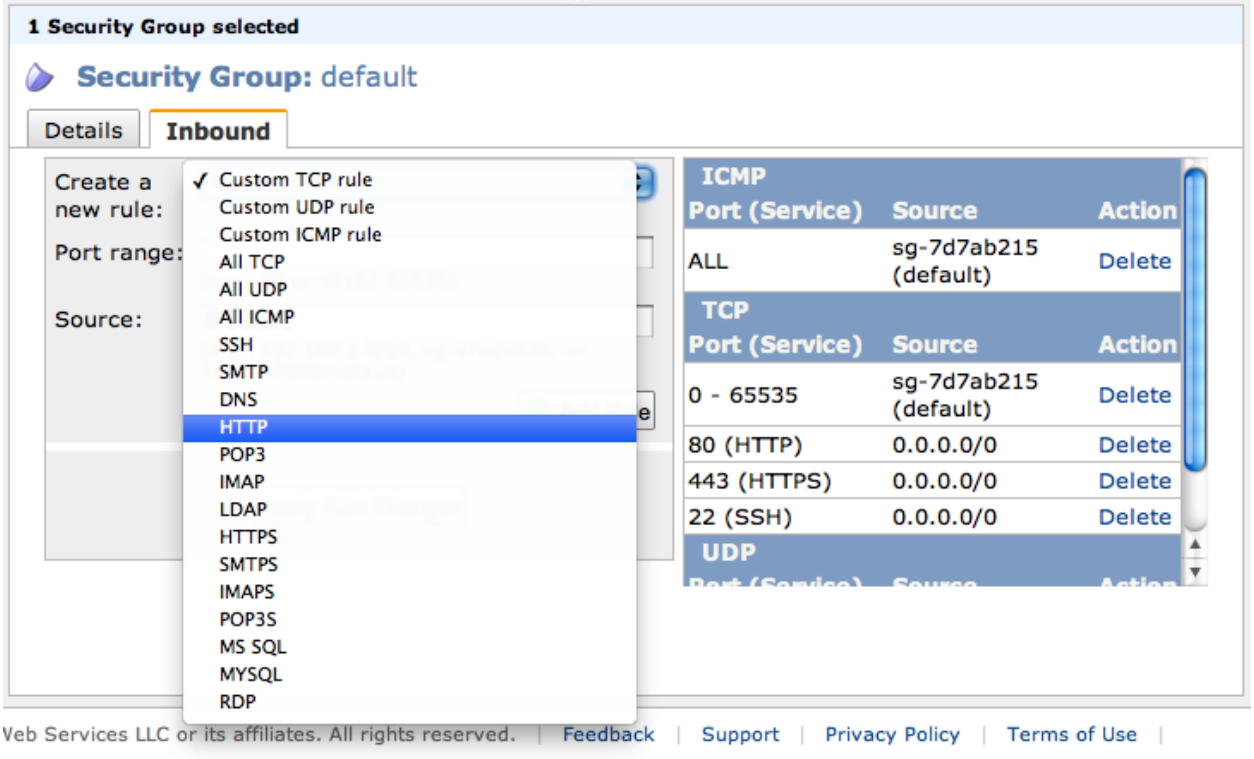
This process may take a few minutes and you will be required to verify your identity using a phone.

Note: Although every effort has been made to make this process as streamlined as possible, if you've never set up a server before, you may find it rather daunting. In this case we suggest pairing up with an engineer until you are through the setup process.

Step 2. Configure your Security Group

Before setting up your PANDA server, you will need to configure your security group so that web requests will be able to reach it.

To do this, visit the Security Groups section of the EC2 Management Console. Select the "default" security group from the list, and then click the "Inbound" tab in the bottom pane of the window. Add rules for `HTTP`, `HTTPS` and `SSH`.



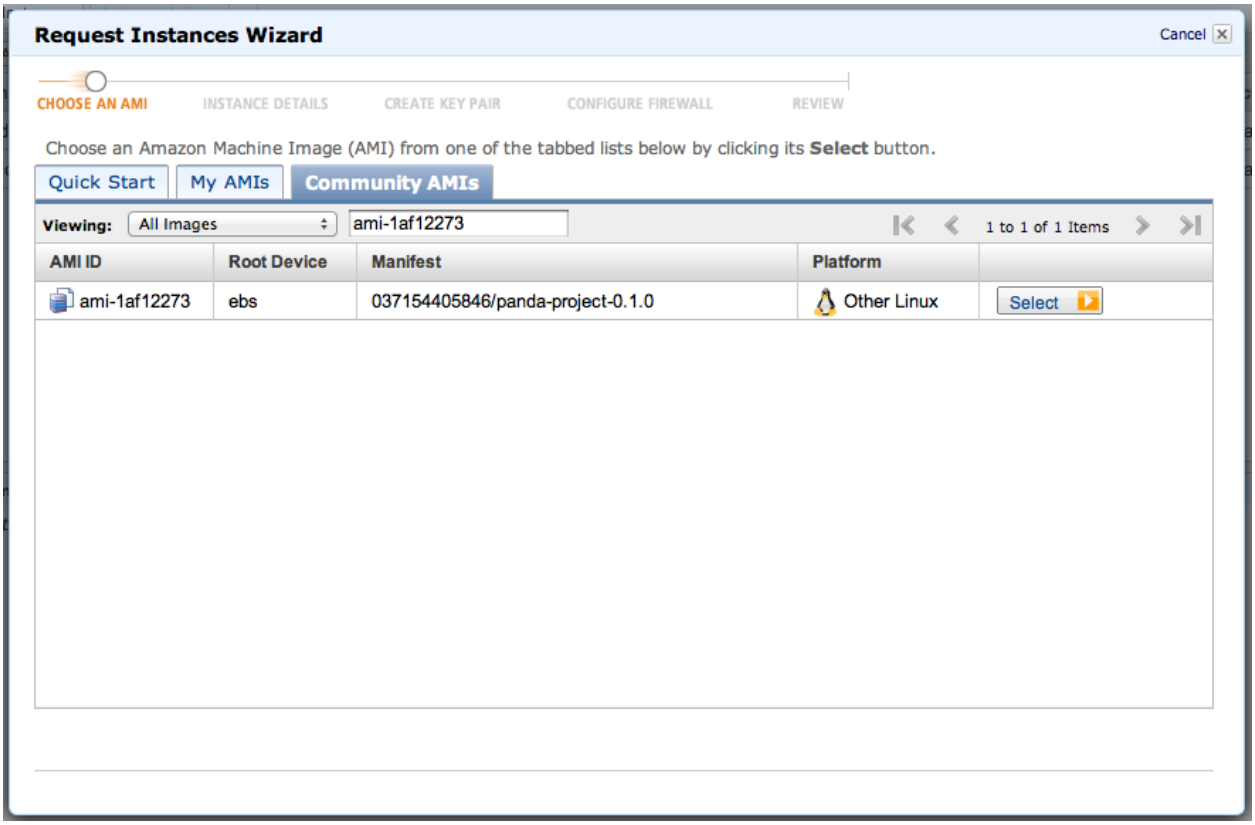
If you don't mind your PANDA being accessible to anyone on the internet, you can enter `0.0.0.0/0` in the **Source** field for each. **This will make your PANDA visible to the public.** (Although it's highly unlikely anyone would find it unless you gave them the link.) More discerning users will want to enter a private IP or subnet assigned to their organization.

Note: If you're not sure what to enter for the **Source** field it would be wise to consult with your IT department to find out if your organization has a private IP or subnet.

Step 3. Launch your server

Method #1 - Use an Amazon Machine Instance (AMI)

This is the absolute simplest way to make a PANDA. Visit the Instances section and click "Launch Instance". Select "Launch Classic Wizard" and click "Continue". Click the "Community AMIs" tab and search for `ami-553e963c`. It may take a moment to return a result. When it does, click "Select". On the next page you'll need to select an **Instance Type**. You are welcome to use (and pay for) a more powerful server, but PANDA has been optimized with the expectation that most organizations will run it on an `m1.small` instance. (At a cost of roughly $70 per month.)



This should provide adequate capacity for small- to medium-sized groups. We don't recommend trying to run it on a `t1.micro` unless you will only be using it for testing.

Once you've selected your instance type, skip over **Availability Zone** and click "Continue". Keep clicking "Continue" and accepting all the default options until the "Continue" button becomes a "Launch" button. Click "Launch".

Note: If you have never used Amazon EC2 before you will be required to create a keypair. Even if you don't know what this is, **don't lose it or delete it**. It is the only way to log into your PANDA server if you ever need to.

Method #2 - Use a script over SSH

This method is slightly more complex and assumes you have some experience operating servers. It also provides greater feedback for users who want to understand more about how PANDA works.

Visit the Instances section and click "Launch Instance". Select "Launch Class Wizard" and click "Continue". Click the "Community AMIs" tab and search for `ami-8cfa58e5`. This is the official Ubuntu 12.04 AMI. It may take a moment to return a result. When it does, click "Select".

On the next page you'll need to select an **Instance Type**. See the notes above regarding instance types. We recommend you select `m1.small`.

Click "Continue" and keep clicking "Continue" and accepting all the default options until the "Continue" button becomes a "Launch" button. Click "Launch".

Once your new server is available, SSH into it and execute the following commands:

```
wget https://raw.github.com/pandaproject/panda/1.0.0/setup_panda.sh
sudo bash setup_panda.sh
```

The disadvantage of this method is that you will need to wait while the setup script is run. This normally takes 15-20 minutes.

Note: An installation log will be created at `/var/log/panda-install.log` in case you need to review any part of the process.

Step 4. Setting up your PANDA

Once you've completed your selected installation method you can will the web interface to complete setup. You can browse directly using to your instance using its "Public DNS Name". Navigate to the EC2 Instances section and select your instance. The public DNS name will be listed among the instance details in the bottom pane. It will look something like this: `ec2-50-16-157-39.compute-1.amazonaws.com`. Visit this in your browser, like so:

```
http://ec2-50-16-157-39.compute-1.amazonaws.com/
```

Your PANDA will be running in setup mode. This guided process will give you an opportunity to create an administrative user. Once you've completed the setup you will be directed to login to your PANDA with your new administrative user.

You may also wish to configure *DNS*, *E-mail* and/or *Secure connections (SSL)*.

3.2.2 Installing on your own hardware

Installation

Server requirements

PANDA requires a server running Ubuntu 12.04. Whether you want to run PANDA in a virtual machine or on the old Compaq under your desk, as long as it can run Ubuntu 12.04, you should be fine. (Performance, of course, may vary widely depending on the hardware.)

Running the install script

SSH into your server and run the following setup commands:

```
wget https://raw.github.com/pandaproject/panda/1.0.0/setup_panda.sh
sudo bash setup_panda.sh
```

Note that the setup script **must** be run with sudo.

An installation log will be created at `/var/log/panda-install.log` in case you need to review any part of the process.

Setting up your PANDA

Once the installation is complete your PANDA will be running on port 80 at the public IP address of the server you installed it on.

Your PANDA will be running in setup mode. This guided process will give you an opportunity to configure the time zone and create an administrative user. Once you've completed the setup you will be directed to login to your PANDA with your new administrative user.

You may also wish to configure *DNS*, *E-mail* and/or *Secure connections (SSL)*.

CHAPTER

FOUR

CONFIGURATION

4.1 Configuring DNS

4.1.1 Getting a domain name

To use a custom domain (or subdomain) with your PANDA, you will first need to requisition one from your IT department or purchase one from a registrar such as Namecheap or dnsimple.

Note: If you're using Amazon EC2, you will now want to set up an "Elastic IP" for your instance. This will give you a permanent address for your server. To do this, visit the EC2 Dashboard and click "Elastic IPs" in the left-hand rail. Click "Allocate New Address" in the toolbar and then "Yes, Allocate". Select the new IP address in the list and click "Associate Address" in the toolbar. Select your PANDA instance from the drop-down and click "Yes, Associate." You'll use this new IP address in the next step.

Create an A (Address) record for your new domain, pointed to your server's IP address. The details of how to do this will depend on your registrar (or the procedures of your IT staff), but typically it's as simple as pasting the IP address into the correct box and clicking "save." It may take some time for your domain to become available, but often it is instantaneous. Trying visiting your new domain in your web browser and PANDA should load. If it does not, wait a while and try again.

4.1.2 Configuring PANDA

Once you have your new domain name directed to your PANDA, you'll want to configure outbound email to use the new domain. You can do this on the settings page:

```
http://localhost:8000/admin/settings
```

Replace `localhost:8000` with your PANDA's domain name.

You will be prompted to log in as an administrative user. Once logged in you will see a list of configuration options. In the section titled "Site domain settings," fill in your new domain name and click "Update Settings".

To test the new domain name, click the "Home" link at the top of the screen and then the link to "Add" a new User. Fill in your own email address and click "Save." You should get an activation email in your inbox. Click the activation link and verify that you return to your PANDA.

4.2 Configuring Email

4.2.1 Getting an SMTP server

Before you configure PANDA for email you're going to need access to an SMTP (email) server. There are several ways you can get access to one of these.

Using your own SMTP

If your organization already has an SMTP server then you may be able to use it for PANDA. Note, however, that if you host PANDA on EC2 and your SMTP server is internal to your organization you may not be able to reach it. It's best to discuss this possibility with the IT staff in charge of your email servers.

If you can use your own SMTP server then make sure you have its address, port number, username and password ready so you can fill them in later.

Using CritSend

If you don't have access to an SMTP server for your organization your best bet is probably to use an email service that provides SMTP access. We suggest CritSend because they are inexpensive (first 1,000 emails are free, next 20,000 are $10) and their registration process doesn't require that your website is public facing.

To sign up for CritSend visit their website and enter your email address in the signup box. You'll receive an activation email. Follow it and then click "My Account" and fill out the information they require. In order to prevent SPAM they do manually validate accounts, so make sure you use legitimate contact information. Once you save it will take up to 24 hours for your registration to be validated.

Once you receive notification that your registration has been validated, you'll be able to use the following settings to configure your PANDA:

- Enabled: `True`
- Host: `smtp.critsend.com` (if you're hosting on Amazon EC2 then use `aws-smtp.critsend.com`)
- Port: `587`
- User: `YOUR_CRITSEND_USERNAME`
- Password: `YOUR_CRITSEND_PASSWORD`
- Use TLS: `True`
- From address: `YOUR_FROM_ADDRESS`

If you wish to setup advanced features such as SPF and Domain Keys to ensure your email is not flagged as SPAM, CritSend has documentation on how to do this.

Using Gmail

Another possiblity is to use Gmail's SMTP servers for sending email. Note that this solution is somewhat hacky and probably shouldn't be relied on as a permanant solution.

You'll need to register for a new Gmail. Do **not** use your primary email account for this, but do register with a real name and contact information.

That's it. To use the Gmail SMTP servers, you'll use the following configuration for PANDA:

- Enabled: `True`

- Host: `smtp.gmail.com`
- Port: `587`
- User: `YOUR_NEW_EMAIL_ADDRESS`
- Password: `YOUR_NEW_PASSWORD`
- Use TLS: `True`
- From address: `YOUR_NEW_EMAIL_ADDRESS`

4.2.2 Configuring PANDA

Once you have your SMTP connection details ready. You're ready to configure your PANDA. Visit the settings page at:

```
http://localhost:8000/admin/settings
```

Replace `localhost:8000` with your PANDA's domain name.

You'll be prompted to log in as an administrative user. Once logged in you'll see a list of configuration options. In the section titled "Email settings", fill in the details of your SMTP connection and then click "Update Settings".

To test the new connection click the "Home" link at the top of the screen and then the link to "Add" a new User. Fill in your own email address and click Save. You should get an activation email in your inbox!

4.3 Performance

The default PANDA configuration has been optimized for an EC2 Small server environment. However, even if that is the hosting solution it is possible the default configuration may not be optimal for you.

4.3.1 Task throttling

If you are running on an instance smaller or larger than our recommended server you may wish to configure a shorter or longer time to wait between importing batches of data. This small "throttle" value allows the server to periodically catch up on user requests while importing data. You'll find this configuration option in the *Performance* section of the admin settings page.

Increasing the number will provided additional time for user requests and should improve PANDA's responsiveness, at the cost of imports taking longer. Decreasing the number will allow less time for user requests, which is appropraite if your server has multiple CPUs. In the latter case you may even be able to set this value to zero and still have a very responsive server.

Note: In the vast majority of cases the default value for this option is fine, so if you are not sure how to tune it, you should probably just leave it alone.

4.3.2 Solr

You may wish to adjust how much memory you give to Solr. You will find the relevant configuration in `/etc/init/solr.conf`:

```
description "Solr server for PANDA"
start on runlevel [2345]
stop on runlevel [!2345]
respawn
exec sudo -u solr sh -c "java -Xms256m -Xmx512m -Dsolr.solr.home=/opt/solr/panda/solr -Djetty.home=/
```

The startup parameters `-Xms` and `-Xmx` control the minimum and maximum memory that Solr will consume while indexing documents and performing queries.

If you're running on anything larger than an EC2 Small you will almost certainly want to increase these numbers. Even on an EC2 Small you may need to increase them if your are storing a large amount of data. As a rule of thumb you will want to leave between `768m` and `1g` of the system's total memory for other processes.

4.3.3 A note on dataset size

In general, uploading very large datasets (greater than 100MB) to PANDA is fine. It is even encouraged! However, there is a caveat regarding extremely wide datasets. Datasets with hundreds or thousands of columns can cause PANDA to perform very poorly. This is not a function of the search being slow, but rather of the amount of data PANDA needs to deliver to and render in the browser. For this reason it is best to avoid uploading very wide datasets.

4.4 Configuring SSL

4.4.1 Getting your certificate

If you've decided to host your PANDA on Amazon's EC2 service or anywhere else that is accessible over the public internet then you should secure your site with an SSL certificate. Broadly, there are two ways you might go about this.

Buying a certificate

The best option is purchase a certificate from an official signing authority such as VeriSign or Digicert. However, a fully validated SSL certificate can cost hundreds of dollars per year.

Fortunantely, for the purposes of PANDA you should be fine with a much less expensive "Domain Validation" only certificate. This are available from many DNS registrars, such as Namecheap as well as dedicated providers like RapidSSL. Domain Validation certificates typically verify the email address in the WHOIS records of your domain registration and then issue instantly. Ideally, you wouldn't want to use one of these to secure a public website, but because PANDA is only used by members of your organization it is sufficient.

Once you've acquired a certificate you can copy it your server by running the following commands in the directory with the files:

```
scp -i <MY_EC2_KEY.pem> <MY_CERT.crt> ubuntu@<MY_PANDA_SERVER.com>:~/panda.crt
scp -i <MY_EC2_KEY.pem> <MY_KEY.key> ubuntu@<MY_PANDA_SERVER.com>:~/panda.key
```

Creating your own certificate

If you are operating your PANDA in a no-budget environment, then you may choose to generate your own "self-signed" certificate instead. This will provide all the benefits of encryption to your users, however, **each user will need to click through a browser warning that the site is not verified**. You will need to explicitly communicate to your users that this error message is normal. (In most browers they will only see it once.)

SSH into your server and run these commands:

```
# You'll be prompted for a passphrase.
# Use anything, but remember what it is.
openssl genrsa -des3 -out panda.key 1024

# After entering your passphrase you'll be prompted for a
# variety of information which you can either fill in or leave blank.
openssl req -new -key panda.key -out panda.csr

cp panda.key panda.key.org

# You'll need your passphrase again here.
openssl rsa -in panda.key.org -out panda.key
openssl x509 -req -days 365 -in panda.csr -signkey panda.key -out panda.crt

rm panda.csr
rm panda.key.org
```

4.4.2 Installing your certificate

Once you've purchased or created a certificate you'll need to install both it and your signing key in the correct location:

```
sudo mv panda.crt /etc/nginx/panda.crt
sudo chown root:root /etc/nginx/panda.crt
sudo chmod 644 /etc/nginx/panda.crt

sudo mv panda.key /etc/nginx/panda.key
sudo chown root:root /etc/nginx/panda.key
sudo chmod 644 /etc/nginx/panda.key
```

4.4.3 Turning it on

The last thing you'll need to do is reconfigure PANDA's webserver (Nginx) to use the new SSL certificate. Fortunantely, there is a one-size-fits-all solution for this. All you need to is SSH into your PANDA server and run the following commands:

```
sudo wget https://raw.github.com/pandaproject/panda/1.0.0/setup_panda/nginx_ssl -O /etc/nginx/sites-a
sudo service nginx restart
```

Your PANDA should now redirect all unsecured requests to a secure `https://` url.

CHAPTER

FIVE

ADMINISTRATION

5.1 User management

5.1.1 Creating a new admin user

Visit the admin users page:

```
http://localhost:8000/admin/auth/user/
```

Replace `localhost:8000` with your PANDA's domain name.

In the upper-right corner click "Add user". Type in the email address and name and click "Save". If you've already setup *Email* then you will receive a registration email with an activation link. Ignore it. On the details page for your new user click the "change password form" link underneath the **Password** field. Enter your password and click "Change Password". Check the **Active**, **Staff status** and **Superuser status** checkboxes and click "Save".

5.1.2 Creating new PANDA users

Note: Setup *Email* before you do this.

Visit the admin users page:

```
http://localhost:8000/admin/auth/user/
```

Replace `localhost:8000` with your PANDA's domain name.

In the upper-right corner click "Add user". Type in the email address. You may choose to also enter the user's name or leave it blank. They will have an opportunity to add/update it. Click "Save". Your new user will receive a registration email with an activation link.

5.1.3 Creating new users in bulk

If you need to create a lot of users you can also use your administrative API key to create new users via the *API*.

5.2 Managing dataset categories

PANDA allows you to organize your datasets into categories that you define. By default it comes configured with a small handful of categories we imagine everyone will need, but you will almost certainly want to add your own.

Note: Categories are intended to be used as **broad** groupings of datasets ("Crime" or "Education"), not projects ("Medicaid Investigation 2012") or tags ("president"). Trying to use them in these latter ways is strongly discouraged.

Categories can be maintained by visiting the categories section of the admin:

```
http://localhost:8000/admin/panda/category/
```

Replace `localhost:8000` with your PANDA's domain name.

This interface should be largely self-explanatory. When creating a new category a url slug will automatically be generated based on the name you provide. In less you feel the need to edit them for brevity, these default slugs are usually fine.

Warning: PANDA automatically manages categories called "all" and "uncategorized". These will not appear in the adminstrative list of categories, but you should not attempt to create categories with exactly these names.

5.3 Managing user API keys

Under the hood PANDA relies on API Keys to allow users to access the application. These keys also allow the user to *programmatically access PANDA* if they have the know-how. Every user is automatically issued an API key when they are created. It is not possible to use PANDA without a valid API key.

From time to time it may be necessary to revoke a user's API key and issue them a new one. Normally you would want to do this if you were concerned that their key had been leaked to a third-party or otherwise compromised by someone who should not have access.

Warning: If it is the user whom you are trying to keep from accessing PANDA, the right way to do so is to deactivate their account. The only reason to delete a user's API key is if you plan on creating a new one for them.

To regenerate a user's API key go to the user admin page:

```
http://localhost:8000/admin/auth/user/
```

Replace `localhost:8000` with your PANDA's domain name.

Select a user from the list and scroll down to the bottom of their page. Check the **Delete** checkbox on the right-hand side of the "Api Keys" section. Click "Save and continue editing".

Scroll to the bottom of the page once more and click "Add another Api Key". Click "Save". Your user now has a new, valid API key and the old one can no longer be used to access your PANDA.

CHAPTER

SIX

SERVER MAINTENANCE

6.1 Connecting with SSH

When performing *upgrades*, adding *storage* or doing advanced maintainence on your PANDA you will need to login over SSH. At its most basic, SSH is a way of communicating with your server via a secure text interface. This quick guide will explain how to SSH into your PANDA server using Amazon EC2's Java SSH client.

Note: If for some reason the Java SSH client doesn't work for you, you will need to connect with a different SSH client. Describing this is beyond the scope of this documentation, however, Amazon does provide complete SSH documentation for both Linux/Mac and Windows.

To use to the Java SSH client first navigate to your EC2 Instances tab. Find your instance in the list, right-click on it and select "Connect". You should see a dialog like this:

In this dialog set the **User name** to `ubuntu` as shown in the image above.

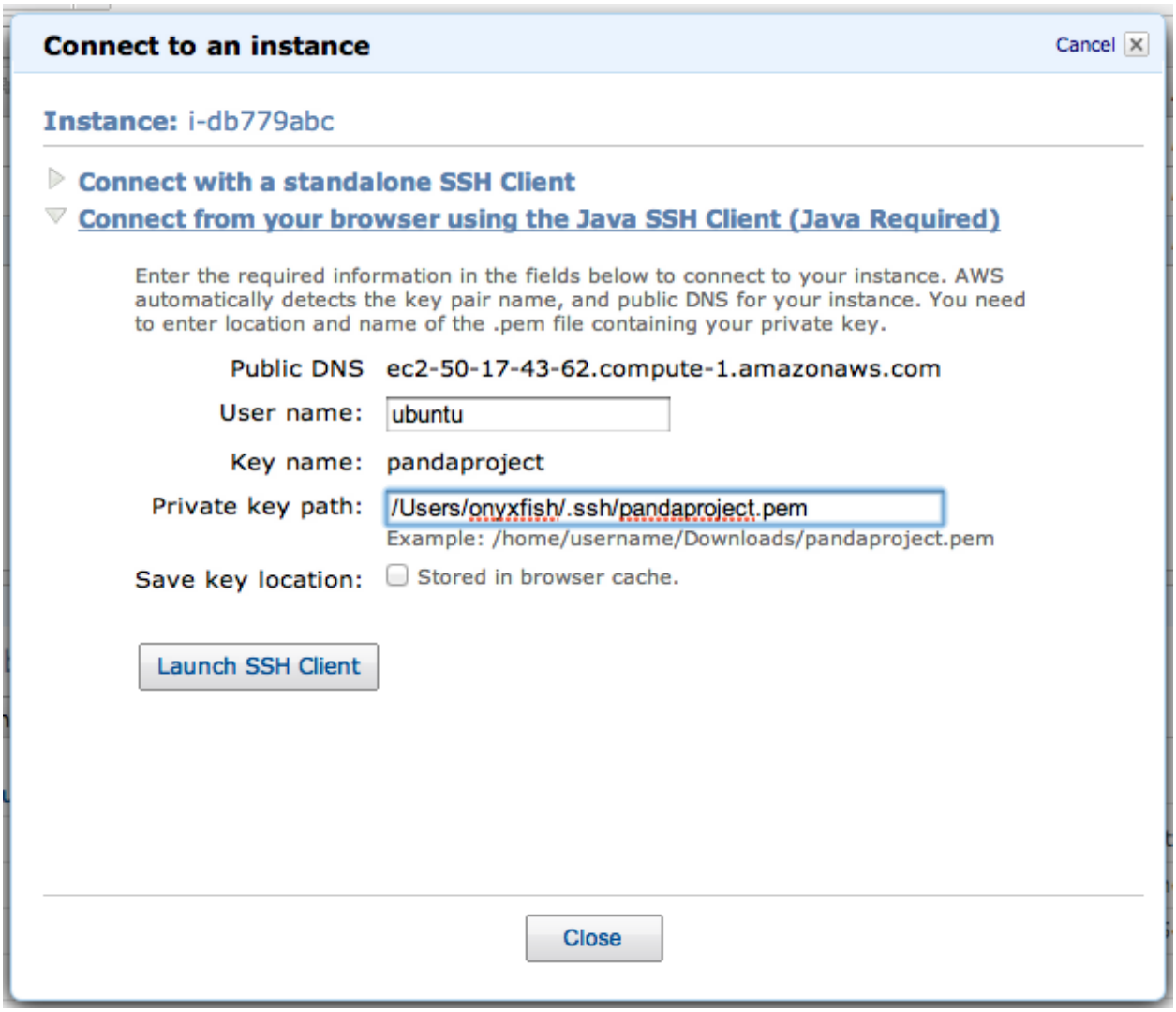
You will also need to set the **Private key path**. This is the location on your computer of the private key file you created when you setup your EC2 account. The name of your key will be the one specified in the **Key name** field and it will have a `.pem` extension. For example, the key in the above example is named `pandaproject.pem`. Unfortunantely, Amazon doesn't provide a "file picker" to use to find the file, so you will need to type out the complete location of the file. Here are some example locations.

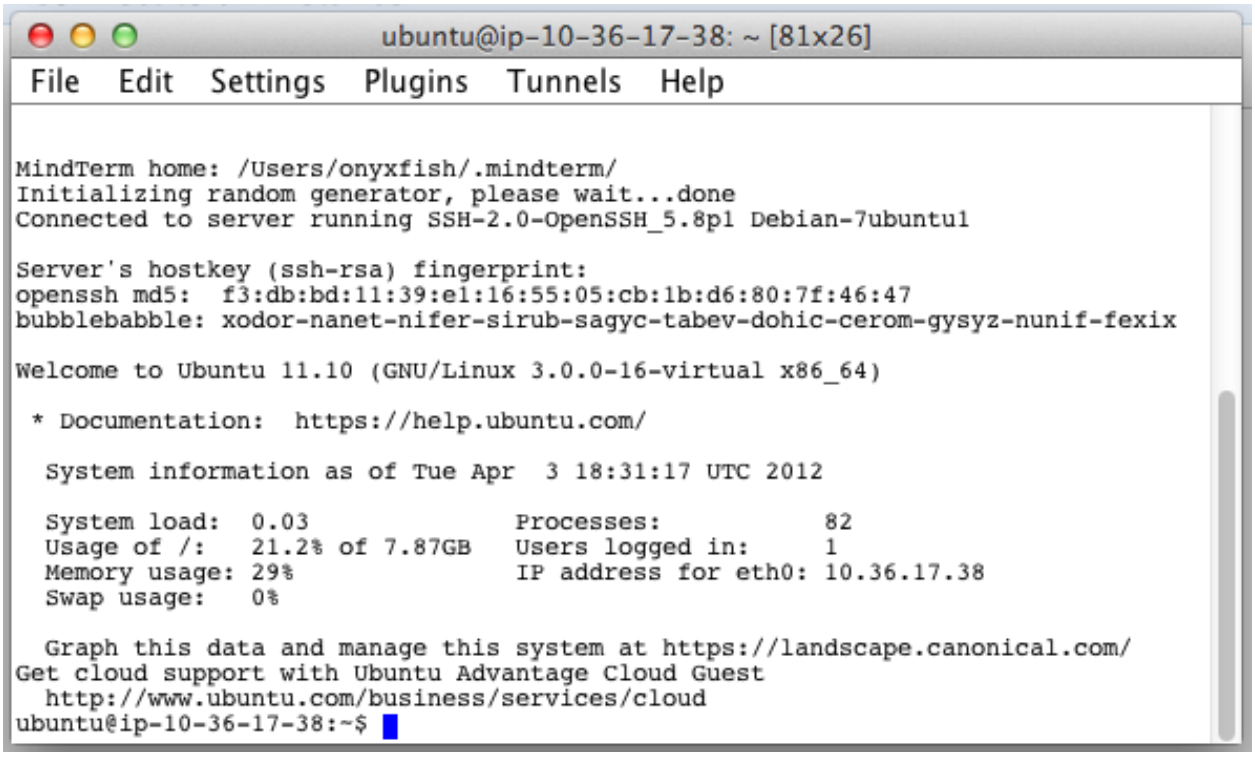
- If you are on Windows XP and your key is in `My Documents` then the full path might be `C:\Documents and Settings\USERNAME\My Documents\KEYNAME.pem`.
- If you are on Windows 7 and your keys in `Downloads` then the full path might be `C:\Users\USERNAME\Downloads\KEYNAME.pem`.
- If you are on Mac OSX and your key is in `Downloads` then the full path might be `/Users/USERNAME/Downloads/KEYNAME.pem`.
- If you are on Linux and your key is in your home directory then the full path might be `/home/USERNAME/KEYNAME.pem`.

Once you have this information keyed in, click "Launch SSH Client".

You will see several dialogs prompting you to allow the software to access your computer and to accept the software's Terms of Service. It is safe to accept each of these. After connecting the software will also prompt you to create some folders on the server. Accept these as well. Once you've made your way through these dialogs, you should see a window that looks something like this:

You've made it! This is your SSH terminal. From here you can enter commands as described in any of the other sections of this documentation. When you've finished using the SSH connection, select "Exit" from the "File" menu to disconnect.


Connect to an instance
Cancel
Instance: i-db779abc
Connect with a standalone SSH Client
Connect from your browser using the Java SSH Client (Java Required)
Enter the required information in the fields below to connect to your instance. AWS automatically detects the key pair name, and public DNS for your instance. You need to enter location and name of the .pem file containing your private key.
Public DNS ec2-50-17-43-62.compute-1.amazonaws.com
User name: ubuntu
Key name: pandaproject
Private key path: /Users/onyxfish/.ssh/pandaproject.pem
Example: /home/username/Downloads/pandaproject.pem
Save key location: Stored in browser cache.
Launch SSH Client
Close



6.2 Systems Administration (Ops)

If you are an advanced user and you're going to run a PANDA server, you may want to know how it all works. This page lays out the users, services and files that are part of the standard PANDA setup.

This might also be thought of as a guide to what you may need to change in the event you want to run PANDA in a non-standard configuration.

6.2.1 Users

panda

Runs the `uwsgi` and `celeryd` services and owns their logs. Owns `/var/lib/panda/media` (compressed assets) and `/var/lib/panda/uploads` (uploaded files). Only this user should be used to execute Django management commands, etc:

```
sudo -u panda -E python manage.py shell
```

Note: Note when executing command as the `panda` user, it's often necessary to use the `-E` option to sudo, so that the `DEPLOYMENT_TARGET` environment variable will be preserved.

---

postgres

Runs the `postgresql` service and owns its database files and logs.

root

Owns everything else, including the panda source code in `/opt/panda`.

solr

Runs the `solr` service and owns its files (`/opt/solr`), indices and logs.

ubuntu

The standard Ubuntu login user. Must be used to SSH into the system and run sudo. Has read-only access to files and logs, but can not execute any system commands.

www-data

Runs the `nginx` service and owns its logs.

6.2.2 Services

celeryd

Task processing.

nginx

Web server. Runs on port 80.

postgresql

Database. Runs on port 5432 locally. Not accessible from remote hosts.

solr

Search engine. Runs on 8983. Not accessible from remote hosts.

uwsgi

Python application server. Runs over a socket at `/var/run/uwsgi/wsgi.sock`.

6.2.3 Files

- `/var/lib/panda/media` - compressed assets
- `/var/lib/panda/uploads` - file uploads
- `/opt/panda` - application source code
- `/opt/solr` - Solr application

- `/var/log/celeryd` - destination for celery logs if output was not captured and routed to panda logs (exists only as a failsafe)
- `/var/log/nginx` - destination for access logs
- `/var/log/panda` - destination for application logs (errors and task processing)
- `/var/log/uwsgi.log` - destination for uwsgi logs (normally just startup and shutdown)

6.3 Backing up your PANDA

If you are using PANDA in production you will want to ensure that you perform frequent backups. For archival purposes the most crucial thing to store is the data files themselves, however, you will probably also want to backup your search indexes and database.

6.3.1 Amazon EC2 backups

Creating a backup

If you are hosting your PANDA on Amazon EC2 then you can make use of EC2's "snapshot" ability for your backups. In essence a snapshot is an exact copy of any EBS volume. You can use this ability to create backups your PANDA server and, if you have *migrated your files and indexes*, the volumes they reside on as well. We provide a script to automate this process. You will need to have at least some experience with administering a server in order to use this script.

To perform a one-time backup *SSH into your server* and execute the following commands:

```
cd /opt/panda/scripts
sudo python backup_volumes.py
```

When asked for your Amazon Access and Secret keys you can paste them into the console. They will not be displayed or logged. Your PANDA will be unavailable during the backup process, which may take some time (especially if your volumes are large).

Once the script has completed your backup will be created and your PANDA will be running again.

Restoring a backup

Restoring a backup created with snapshots is a matter of restoring each volume and ensuring they are mounted correctly. However, because one of the volumes that will need to be restored is the root volume of the instance, we have to do a little magic. To restore the root volume:

- On the EC2 Instances tab locate the instance you wish to restore.
- Right-click that instance and select "Launch More Like This".
- Proceed through the wizard, modifying the **Instance Type**, if desired.
- Launch your new instance.
- Once the new instance is available, select it and find its instance ID near the top of the bottom pane. It will look like `i-76acf913`. Note this, you will need it in a minute.
- Right-click on your instance and "Stop" it. (Don't "Terminate" it!)
- Once it's stopped, navigate to the Volumes tab.
- Find the volume which has the instance ID you noted in the "Attachment Information" column.

- Select this volume and click "Detach Volume."
- Once it is detached, delete it.
- On the EC2 Snapshots tab locate the latest snapshot of your root volume, in the description it will say "mounted at /". Note the Snapshot ID. It will look like `snap-f8b6c49f`. You will need this in a moment.
- Select your snapshot and click "Create Volume". Make it 8GB.
- Got back to the Volumes tab. You should see your new volume in the list. Its identifiable by the snapshot ID you noted in the last step.
- Select it and click "Attach Volume".
- Find your instance in the list using the instance ID you noted before. It should still be in "stopped" state. Set the **Device** to `/dev/sda1`. Attach it!

If you have run either the files or indexes *storage migration scripts* then you will also need to restore your additional storage volumes. These are more straight-forward:

- Create a volume from each snapshot in the same manner you created the root volume.
- Make note of the device in the description of each volume. It looks like `/dev/sdg`.
- When attaching each volume in the instance, use this value as the **Device**.

Once all volumes have been created and attached, navigate back to the Instances tab, find your instance in the list, right-click it and select "Start". Once it is finished booting up you will have successfully restored your PANDA backup!

6.3.2 Self-install backups

If you chose to self-install PANDA your backup options may vary widely. If possible you should consider periodically disabling the PANDA services and backing up the entire filesystem. If you have files and indexes stored on separate devices you will, of course, want to back those up as well. Using compressed and/or incremental backups will likely save you a significant amount of storage space.

6.4 Adding more storage to your PANDA

PANDA stores both raw data files and search indexes. Both of these can be quite large. At some point you may find you need to upgrade your storage for one or both. This document describes how to do this.

> **Warning:** All procedures described on this page **can destroy your data**. Please take every precaution to ensure your data is *backed up* before beginning.

6.4.1 Amazon EC2 upgrade

If you are hosting your PANDA on Amazon EC2 and used our *installation guide* then adding additional storage is easy. PANDA comes with scripts which will create new storage volumes, attach them to your server, and migrate existing files.

Based on Amazon's defaults for EC2 instances and the size of the PANDA software itself, you need to consider adding storage when you have more than 1.5 GB total data you want to upload. An initial PANDA installation leaves slightly less than 6 GB available storage for files and search indexes. While circumstances vary, search indexes seem to use about three times as much disk space as the uploaded files they index. If you use the *API* to add data to PANDA, you will use more search index space but not more file storage space.

Before you begin, estimate how much storage you need. Additional storage on Amazon incurs a monthly cost based on the allocated disk size. If you want to start small, you could simply add an 18GB volume for the search index storage. This would give you capacity for slightly less than 6 GB worth of uploaded files. If you expect to need more than that, remember that you should allocate at least 3 times more space for your search index volume than for your files volume.

To upgrade your search index storage, *SSH into your server* and execute the following commands:

```
cd /opt/panda/scripts
sudo python migrate_solr_volume.py
```

To upgrade your file index storage, the second command would be:

```
sudo python migrate_files_volume.py
```

Both scripts will prompt you for some information. When asked for your Amazon Access and Secret keys you can paste them into the console. They will not be displayed or logged. You will also need to enter a size (in gigabytes) for your new volume.

Once the script has completed your files or indexes will be on the new volume and your PANDA will be running again.

Note: If you run this script more than once (i.e. migrating from an added volume to a new added volume) then the old volume will not be detached from your instance or destroyed. Describing how to do this is beyond the scope of this documentation. **You will be billed for EBS volumes until you destroy them, even if they are not actively being used.**

6.4.2 Self-install upgrade

Although we can't provide a detailed upgrade guide for self-installed PANDA instances, the basic outline of what you need to do is simple:

- Take your PANDA out of service.
- Add a new storage device and mount it at a temporary location.
- Copy the contents of either `/var/lib/panda` (for files) or `/opt/solr/panda/solr` to the temporary location.
- Unmount the device, delete the original files and remount the device at the original location.
- Restart your PANDA.

Don't forget to update `/etc/fstab` so that the new device will be automatically mounted after a reboot.

6.5 Upgrading your PANDA

6.5.1 Before you get started

Although we strive to make upgrades as simple as possible, upgrading your PANDA will require that you know how to SSH into your server. If you need help with this see our guide to *Connecting with SSH*. And don't be afraid to ask for help on the PANDA Users Group.

Warning: Your PANDA will be unavailable while upgrading. Typically this will not last more than five minutes, but it will vary by release. You should plan to perform PANDA upgrades during off hours.

The following release are in **reverse** version order. They **must** be performed in sequence (from lowest version number to highest version number–reverse order on this page).

6.5.2 0.2.0 to 1.0.0

To upgrade your PANDA from the 0.2.0 release to the 1.0.0 release, *SSH* into your server and execute the following commands:

```
wget https://raw.github.com/pandaproject/panda/1.0.0/scripts/migrations/0.2.0-to-1.0.0.sh
sudo bash 0.2.0-to-1.0.0.sh
```

Your PANDA will be stopped, the upgrade will be applied and it will then be restarted. A log of this process will be put in `/var/log/panda-upgrade-1.0.0.log`.

After running this upgrade your PANDA will be reset into *setup mode*, giving you an opportunity to set your local timezone and create an administrative user. You should visit your PANDA soon after the upgrade to complete this process. If already have an administrative user configured, simply create one with dummy data and then deactivate or delete it after completing the setup. **DO NOT attempt to recreate a user that already exists.** Doing so will render that account unable to login.

Note: This upgrade will automatically upgrade your server's Ubuntu distrubution to version 12.04. This long-release version of Ubuntu will be supported by Canonical (the company behind Ubuntu) for five years. If you have made any customizations to your PANDA's server environment be aware that this upgrade could have unintended consequences.

Check out the *Changelog* to see all the new features and bug fixes in this release!

6.5.3 0.1.4 to 0.2.0

To upgrade your PANDA from the 0.1.4 release to the 0.2.0 release, *SSH* into your server and execute the following commands:

```
wget https://raw.github.com/pandaproject/panda/0.2.0/scripts/migrations/0.1.4-to-0.2.0.sh
sudo bash 0.1.4-to-0.2.0.sh
```

Your PANDA will be stopped, the upgrade will be applied and it will then be restarted. A log of this process will be put in `/var/log/panda-upgrade-0.2.0.log`.

Note: As part of this upgrade all existing activation keys will be voided. New activation keys can be generated via the admin.

Check out the *Changelog* to see all the new features and bug fixes in this release!

6.5.4 0.1.3 to 0.1.4

To upgrade your PANDA from the 0.1.3 release to the 0.1.4 release, *SSH* into your server and execute the following commands:

```
wget https://raw.github.com/pandaproject/panda/0.1.4/scripts/migrations/0.1.3-to-0.1.4.sh
sudo bash 0.1.3-to-0.1.4.sh
```

Your PANDA will be stopped, the upgrade will be applied and it will then be restarted. A log of this process will be put in `/var/log/panda-upgrade-0.1.4.log`.

Note: This version adds an option to explicitly enable or disable sending email. If you've previously configured email you will need to visit the settings page for your PANDA (http://MY-PANDA/admin/settings) and check the "Enable email?" checkbox.

Check out the *Changelog* to see all the new features and bug fixes in this release!

6.5.5 0.1.2 to 0.1.3

To upgrade your PANDA from the 0.1.2 release to the 0.1.3 release, *SSH* into your server and execute the following commands:

```
wget https://raw.github.com/pandaproject/panda/0.1.3/scripts/migrations/0.1.2-to-0.1.3.sh
sudo bash 0.1.2-to-0.1.3.sh
```

Your PANDA will be stopped, the upgrade will be applied and it will then be restarted. A log of this process will be put in `/var/log/panda-upgrade-0.1.3.log`.

Check out the *Changelog* to see all the new features and bug fixes in this release!

6.5.6 0.1.1 to 0.1.2

To upgrade your PANDA from the 0.1.1 release to the 0.1.2 release, *SSH* into your server and execute the following commands:

```
wget https://raw.github.com/pandaproject/panda/0.1.2/scripts/migrations/0.1.1-to-0.1.2.sh
sudo bash 0.1.1-to-0.1.2.sh
```

Your PANDA will be stopped, the upgrade will be applied and it will then be restarted. A log of this process will be put in `/var/log/panda-upgrade.log`.

Check out the *Changelog* to see all the new features and bug fixes in this release!

6.5.7 0.1.0 to 0.1.1

To upgrade your PANDA from the first beta release to the 0.1.1 release, *SSH* into your server and execute the following commands:

```
wget https://raw.github.com/pandaproject/panda/0.1.1/scripts/migrations/0.1.0-to-0.1.1.sh
sudo bash 0.1.0-to-0.1.1.sh
```

Your PANDA will be stopped, the upgrade will be applied and it will then be restarted. A log of this process will be put in `/var/log/panda-upgrade.log`.

Check out the *Changelog* to see all the new features and bug fixes in this release!

CHAPTER

SEVEN

EXTENDING PANDA

7.1 API Import Tutorial

PANDA's API is designed to allow you to programmatically import (and, to a lesser extent, export) data from PANDA. In this tutorial we will show you how you can use the PANDA API to pull data froma web scraper into PANDA.. Our example will be written in Python using the Requests module, but should be easily portable to any language.

Note: If you just want to skip to the code, check out all our API examples on Github.

7.1.1 The problem of synchronization

Writing a PANDA scraper usually means that you are trying to *synchronize* PANDA with some data source. True synchornization, where all new, changed and deleted rows get replicated to PANDA is often not possible. Because of this it is important to think in advance about how you will design your import process.

The first question to to ask yourself is if each row of data has a unique id that will allow you to identify it. (A *primary key* in SQL parlance.) In PANDA we call this value the `external_id`, because it is generated *external* to PANDA. An `external_id` could be anything from a row number to a social security number.

If you can provide an `external_id` for your data you will be able to read and update your individual rows of data at a unique URL:

```
GET http://localhost:8000/api/1.0/dataset/[slug]/data/[external_id]/
```

If your data doesn't have an `external_id` then you won't be able to read or update individual rows of data. (You can still find them via search or delete them in bulk.) In this case the only way to *synchronize* changes between PANDA and your source dataset will be to delete and reimport all rows.

Even if you do have an `external_id` you may still need to delete all rows if your source doesn't provide a *changelog*. A changelog is a stream of metadata that describes when rows of data are modified. Without a changelog it will be impossible to tell if a row of data has been *deleted*. (Because, by definition, it won't be there anymore to find out about.) CouchDB is a rare example of a database that provides a changelog. (See our CouchDB example.)

In most cases you will have an `external_id`, but not a changelog, so you will need to decide if it is important that rows deleted in the source dataset are also deleted in your PANDA. If so, you will need to wipe out all the data before importing the new data.

Note: There is no technical difference between creating and updating a row in PANDA. In either case data will be overwritten if there is a row with a matching `external_id` already in the dataset, otherwise a new row will be created.

7.1.2 Our source data

In this tutorial we are going to be scraping the results of a very simple web scraper, hosted on Scraperwiki, that is aggregating all tweets about the PANDA Project. Because Scraperwiki has an API, we can write a simple script that will allow us to import the results and then run that script as often as we like.

The data we are importing has three simple columns:

- `text`, the full-text of the tweet.
- `id`, Twitter's unique id for the tweet.
- `from_user`, the username of the person who sent the tweet.

Although Twitter doesn't provide a changelog, they do provide a unique `id`, which we can use as our `external_id`. We won't be able to track deleted tweets, but in this case that may actually be to our advantage, since deleted tweets can themselves be interesting. So for this data we will take the approach of reimporting all data, but not deleting anything. This will ensure we pick up any new tweets and any changes (though tweets should never change).

7.1.3 Step 1: Getting setup

Before we get started we are going to need to import the Python standard JSON module, as well as Requests. If you don't have requests you can install it with `pip install requests` (or `easy_install requests`).

```
import json
import requests
```

Now we will define some global variables for values we are going to reuse throughout the script.

```
# Replace "localhost:8000" with your PANDA's DNS name or IP address
PANDA_API = 'http://localhost:8000/api/1.0'

# Replace these parameters with your adminstrator's email and api key
PANDA_AUTH_PARAMS = {
    'email': 'panda@pandaproject.net',
    'api_key': 'edfe6c5ffd1be4d3bf22f69188ac6bc0fc04c84b'
}

# This will be the slug of the dataset we are creating/updating
PANDA_DATASET_SLUG = 'twitter-pandaproject'

# This is the url where the dataset will live, we send our GET/PUT requests here
PANDA_DATASET_URL = '%s/dataset/%s/' % (PANDA_API, PANDA_DATASET_SLUG)

# This is the url under which all data for this dataset lives
PANDA_DATA_URL = '%s/dataset/%s/data/' % (PANDA_API, PANDA_DATASET_SLUG)

# Change this value to configure how many rows should be sent to PANDA in each batch
PANDA_BULK_UPDATE_SIZE = 1000

# This is the url of the Scraperwiki endpoint for our Twitter scraper
# This was generated from: https://scraperwiki.com/docs/api#sqlite
SCRAPERWIKI_URL = 'https://api.scraperwiki.com/api/1.0/datastore/sqlite?format=jsonlist&name=basic_tw

# These are the three columns in our dataset
COLUMNS = ['text', 'id', 'from_user']
```

Next we will define two convenience methods that handle reading from and writing to PANDA. These will save us repeating ourselves each time we need to send an authenticated request to PANDA.

```
# Wrapper around a GET request
def panda_get(url, params={}):
    params.update(PANDA_AUTH_PARAMS)
    return requests.get(url, params=params)

# Wrapper around a PUT request
def panda_put(url, data, params={}):
    params.update(PANDA_AUTH_PARAMS)
    return requests.put(url, data, params=params, headers={ 'Content-Type': 'application/json' })
```

7.1.4 Step 2: Creating the dataset

Before we start importing our data we will first check to see if the dataset exists. If it has not yet been created we will create it.

```
# Attempt to fetch the dataset at its url
response = panda_get(PANDA_DATASET_URL)

# If it doesn't exist the response will be a 404 (not found)
if response.status_code == 404:
    # This is will be serialized as JSON and sent to PANDA to create the dataset
    dataset = {
        'name': 'PANDA Project Twitter Search',
        'description': 'Results of the scraper at <a href="https://scraperwiki.com/scrapers/basic_twi
    }

    # In addition to the name and description, we also use the "column" querystring parameter
    # to define the dataset's columns. You will always need to do this when creating datasets
    # via the API.
    response = panda_put(PANDA_DATASET_URL, json.dumps(dataset), params={
        'columns': ','.join(COLUMNS),
    })
```

Note: See also: complete documentation for the Datasets API.

7.1.5 Step 3: Fetching data from Scraperwiki

To get data from Scraperwiki we simply request the url we defined above. Because we specified the `jsonlist` parameter the response is a JSON document containing an array of keys and an array of rows.

```
# Request the latest data
response = requests.get(SCRAPERWIKI_URL)

# The response is json, so deserialize it
data = json.loads(response.content)
```

7.1.6 Step 4: Load the data into PANDA!

Now that we have our data from Scraperwiki we simply iterate over the rows and convert them into batches of data to be sent to PANDA.

```
# This is the data structure that will be sent to PANDA
put_data = {
    'objects': []
}

# The row data from Scraperwiki is inside the "data" key
# We enumerate the rows as we go so we can load the data in batches
for i, row in enumerate(data['data']):
    # Each row we send to PANDA consists of "data", an array of column values
    # and the "external_id", which *must* be unicode
    put_data['objects'].append({
        'data': row,
        'external_id': unicode(row[1])
    })

    # Everytime we've processed 1000 records, we send them to PANDA
    if i and i % PANDA_BULK_UPDATE_SIZE == 0:
        panda_put(PANDA_DATA_URL, json.dumps(put_data))
        put_data['objects'] = []

# At the end we will probably have records left over, so we send the rest
if put_data['objects']:
    print 'Updating %i rows' % len(put_data['objects'])
    response = panda_put(PANDA_DATA_URL, json.dumps(put_data))
```

Note: See also: complete documentation for the Data API.

7.1.7 Step 5: PANDAmonium!

And that's it, we're done! PANDA now has the data. We can run this script as frequently as we like. Existing rows will be overwritten with any changes and new rows will be added. For very large datasets we suggest running these scripts during off hours.

You can see the complete script on Github.

7.2 API Documentation

The PANDA application is built on top of a REST API that can be used to power custom applications or import/export data in novel ways.

The PANDA API follows the conventions of Tastypie except in important cases where doing so would create unacceptable limitations. If this documentation seems incomplete, refer to Tastypie's page on Interacting with the API to become familiar with the common idiom.

Note: You will probably want to try these URLs in your browser. In order to make them work you'll need to use the `format`, `email`, and `api_key` query string parameters. For example, if you had a user named panda@pandaproject.net you might append the following query string to any url described on this page:

```
?format=json&email=panda@pandaproject.net&api_key=edfe6c5ffd1be4d3bf22f69188ac6bc0fc04c84b
```

Unless otherwise specified, all endpoints that return lists support the `limit` and `offset` parameters for pagination. Pagination information is contained in the embedded `meta` object within the response.

7.2.1 Users

User objects can be queried to retrieve information about PANDA users. Passwords and API keys are not included in responses.

> **Warning:** If accessing the API with normal user credentials you will only be allowed to fetch/list users and to update your own data. Superusers can update any user, as well as delete existing users and create new ones.

Example User object:

```
{
    date_joined: "2011-11-04T00:00:00",
    email: "panda@pandaproject.net",
    first_name: "Redd",
    id: "1",
    is_active: true,
    last_login: "2011-11-04T00:00:00",
    last_name: "",
    resource_uri: "/api/1.0/user/1/"
}
```

Schema

```
GET http://localhost:8000/api/1.0/user/schema/
```

List

```
GET http://localhost:8000/api/1.0/user/
```

Fetch

```
GET http://localhost:8000/api/1.0/user/[id]/
```

Create

New users are created by POSTing a JSON document containing at least the `email` property to the user endpoint. Other properties such as `first_name` and `last_name` may also be set. If a `password` property is specified it will be set on the new user, but it will not be included in the response. If `password` is omitted and email is enabled the new user will be sent an activation email.

```
POST http://localhost:8000/api/1.0/user/
```

```
{
    "email": "test@test.com",
    "first_name": "John",
    "last_name": "Doe"
}
```

Update

PANDA supports updating users via a simulated PATCH verb. To update a user PUT to the user's URL, with `patch` as a query string parameter. In the body of your request include only those attributes of the user you want to change.

```
PUT http://localhost:8000/api/1.0/user/[id]/?patch=true

{
    "last_name": "My New Last Name"
}
```

7.2.2 Tasks

The Task API allows you to access data about asynchronous processes running on PANDA. This data is read-only.

Example Task object:

```
{
    end: "2011-12-12T15:11:25",
    id: "1",
    message: "Import complete",
    resource_uri: "/api/1.0/task/1/",
    start: "2011-12-12T15:11:25",
    status: "SUCCESS",
    task_name: "panda.tasks.import.csv",
    traceback: null
}
```

Schema

```
GET http://localhost:8000/api/1.0/task/schema/
```

List

```
GET http://localhost:8000/api/1.0/task/
```

List filtered by status

List tasks that are PENDING (queued, but have not yet started processing):

```
GET http://localhost:8000/api/1.0/task/?status=PENDING
```

Note: Possible task statuses are `PENDING`, `STARTED`, `SUCCESS`, `FAILURE`, `ABORT REQUESTED` and `ABORTED`.

List filtered by date

List tasks that ended on October 31st, 2011:

```
GET http://localhost:8000/api/1.0/task/?end__year=2011&end__month=10&end__day=31
```

Fetch

```
GET http://localhost:8000/api/1.0/task/[id]/
```

7.2.3 Data Uploads

Due to limitations in upload file-handling, it is not possible to create Uploads via the normal API. Instead data files should be uploaded to http://localhost:8000/data_upload/ either as form data or as an AJAX request. Examples of how to upload files with curl are at the end of this section.

Example DataUpload object:

```
{
    columns: [
        "id",
        "first_name",
        "last_name",
        "employer"
    ],
    creation_date: "2012-02-08T17:50:09",
    creator: {
        date_joined: "2011-11-04T00:00:00",
        email: "user@pandaproject.net",
        first_name: "User",
        id: "2",
        is_active: true,
        last_login: "2012-02-08T22:45:28",
        last_name: "",
        resource_uri: "/api/1.0/user/2/"
    },
    data_type: "csv",
    dataset: "/api/1.0/dataset/contributors/",
    dialect: {
        delimiter: ",",
        doublequote: false,
        lineterminator: "\r\n",
        quotechar: "\"",
        quoting: 0,
        skipinitialspace: false
    },
    encoding: "utf-8",
    filename: "contributors.csv",
    "guessed_types": ["int", "unicode", "unicode", "unicode"],
    id: "1",
    imported: true,
    original_filename: "contributors.csv",
    resource_uri: "/api/1.0/data_upload/1/",
    sample_data: [
        [
            "1",
            "Brian",
            "Boyer",
            "Chicago Tribune"
        ],
        [
            "2",
            "Joseph",
```

```
            "Germuska",
            "Chicago Tribune"
        ],
        [
            "3",
            "Ryan",
            "Pitts",
            "The Spokesman-Review"
        ],
        [
            "4",
            "Christopher",
            "Groskopf",
            "PANDA Project"
        ]
    ],
    size: 168,
    title: "PANDA Project Contributors"
}
```

Schema

```
GET http://localhost:8000/api/1.0/data_upload/schema/
```

List

```
GET http://localhost:8000/api/1.0/data_upload/
```

Fetch

```
GET http://localhost:8000/api/1.0/data_upload/[id]/
```

Download original file

```
GET http://localhost:8000/api/1.0/data_upload/[id]/download/
```

Upload as form-data

When accessing PANDA via curl, your email and API key can be specified with the headers `PANDA_EMAIL` and `PANDA_API_KEY`, respectively:

```
curl -H "PANDA_EMAIL: panda@pandaproject.net" -H "PANDA_API_KEY: edfe6c5ffd1be4d3bf22f69188ac6bc0fc04
-F file=@test.csv http://localhost:8000/data_upload/
```

Upload via AJAX

```
curl -H "PANDA_EMAIL: panda@pandaproject.net" -H "PANDA_API_KEY: edfe6c5ffd1be4d3bf22f69188ac6bc0fc04
--data-binary @test.csv -H "X-Requested-With:XMLHttpRequest" http://localhost:8000/data_upload/?qqfil
```

Note: When using either upload method you may specify the character encoding of the file by passing it as a parameter, e.g. `?encoding=latin1`

Update

Data uploads may be updated by PUTing new data to the object's endpoint. However, only the `title` field is writeable. All other fields are read-only.

7.2.4 Related Uploads

As with Data Uploads, it is not possible to create Uploads via the normal API. Instead related files should be uploaded to http://localhost:8000/related_upload/ either as form data or as an AJAX request. Examples of how to upload files with curl are at the end of this section.

Example RelatedUpload object:

```
{
    creation_date: "2012-02-08T23:14:35",
    creator: {
        date_joined: "2011-11-04T00:00:00",
        email: "user@pandaproject.net",
        first_name: "User",
        id: "2",
        is_active: true,
        last_login: "2012-02-08T22:45:28",
        last_name: "",
        resource_uri: "/api/1.0/user/2/"
    },
    dataset: "/api/1.0/dataset/master-4/",
    filename: "PANDA.1.png",
    id: "1",
    original_filename: "PANDA.1.png",
    resource_uri: "/api/1.0/related_upload/1/",
    size: 58990,
    title: "PANDA Logo"
}
```

Schema

```
GET http://localhost:8000/api/1.0/related_upload/schema/
```

List

```
GET http://localhost:8000/api/1.0/related_upload/
```

Fetch

```
GET http://localhost:8000/api/1.0/related_upload/[id]/
```

Download original file

```
GET http://localhost:8000/api/1.0/related_upload/[id]/download/
```

Upload as form-data

When accessing PANDA via curl, your email and API key can be specified with the headers `PANDA_EMAIL` and `PANDA_API_KEY`, respectively:

```
curl -H "PANDA_EMAIL: panda@pandaproject.net" -H "PANDA_API_KEY: edfe6c5ffd1be4d3bf22f69188ac6bc0fc04
-F file=@README.txt http://localhost:8000/related_upload/
```

Upload via AJAX

```
curl -H "PANDA_EMAIL: panda@pandaproject.net" -H "PANDA_API_KEY: edfe6c5ffd1be4d3bf22f69188ac6bc0fc04
--data-binary @README.txt -H "X-Requested-With:XMLHttpRequest" http://localhost:8000/related_upload/1
```

Update

Related uploads may be updated by PUTing new data to the object's endpoint. However, only the `title` field is writeable. All other fields are read-only.

7.2.5 Categories

Categories are referenced by slug, rather than by integer id (though they do have one).

Example Category object:

```
{
    dataset_count: 2,
    id: "1",
    name: "Crime",
    resource_uri: "/api/1.0/category/crime/",
    slug: "crime"
}
```

Schema

```
http://localhost:8000/api/1.0/category/schema/
```

List

When queried as a list, a "fake" category named "Uncategorized" will also be returned. This category includes the count of all Datasets not in any other category. It's slug is `uncategorized` and its id is 0, but it can only be accessed as a part of the list.

```
http://localhost:8000/api/1.0/category/
```

Fetch

```
http://localhost:8000/api/1.0/category/[slug]/
```

7.2.6 Exports

Example Export object:

```
{
    creation_date: "2012-07-12T14:38:00",
    creator: "/api/1.0/user/2/",
    dataset: null,
    filename: "search_export_2012-07-12T14:38:00.078677+00:00.zip",
    id: "1",
    original_filename: "search_export_2012-07-12T14:38:00.078677+00:00.zip",
    resource_uri: "/api/1.0/export/1/",
    size: 287,
    title: "search_export_2012-07-12T14:38:00.078677+00:00.zip"
}
```

Schema

```
http://localhost:8000/api/1.0/export/schema/
```

List

```
http://localhost:8000/api/1.0/export/
```

Fetch

```
http://localhost:8000/api/1.0/export/[id]/
```

Download

```
http://localhost:8000/api/1.0/export/[id]/download/
```

7.2.7 Datasets

Dataset is the core object in PANDA and by far the most complicated. It contains several embedded objects describing the columns of the dataset, the user that created it, the related uploads, etc. It also contains information about the history of the dataset and whether or not it is currently locked (unable to be modified). Datasets are referenced by slug, rather than by integer id (though they do have one).

Example Dataset object:

```
{
    categories: [ ],
    column_schema: [
        {
```

```
            indexed: false,
            indexed_name: null,
            max: null,
            min: null,
            name: "first_name",
            type: "unicode"
        },
        {
            indexed: false,
            indexed_name: null,
            max: null,
            min: null,
            name: "last_name",
            type: "unicode"
        },
        {
            indexed: false,
            indexed_name: null,
            max: null,
            min: null,
            name: "employer",
            type: "unicode"
        }
    ],
    creation_date: "2012-02-08T17:50:11",
    creator: {
        date_joined: "2011-11-04T00:00:00",
        email: "user@pandaproject.net",
        first_name: "User",
        id: "2",
        is_active: true,
        last_login: "2012-02-08T22:45:28",
        last_name: "",
        resource_uri: "/api/1.0/user/2/"
    },
    current_task: {
        creator: "/api/1.0/user/2/",
        end: "2012-02-08T17:50:12",
        id: "1",
        message: "Import complete",
        resource_uri: "/api/1.0/task/1/",
        start: "2012-02-08T17:50:12",
        status: "SUCCESS",
        task_name: "panda.tasks.import.csv",
        traceback: null
    },
    data_uploads: [
        {
            columns: [
                "first_name",
                "last_name",
                "employer"
            ],
            creation_date: "2012-02-08T17:50:09",
            creator: {
                date_joined: "2011-11-04T00:00:00",
                email: "user@pandaproject.net",
                first_name: "User",
```

```
            id: "2",
            is_active: true,
            last_login: "2012-02-08T22:45:28",
            last_name: "",
            resource_uri: "/api/1.0/user/2/"
        },
        data_type: "csv",
        dataset: "/api/1.0/dataset/contributors/",
        dialect: {
            delimiter: ",",
            doublequote: false,
            lineterminator: "
            ",
            quotechar: """,
            quoting: 0,
            skipinitialspace: false
        },
        encoding: "utf-8",
        filename: "contributors.csv",
        id: "1",
        imported: true,
        original_filename: "contributors.csv",
        resource_uri: "/api/1.0/data_upload/1/",
        sample_data: [
            [
                "Brian",
                "Boyer",
                "Chicago Tribune"
            ],
            [
                "Joseph",
                "Germuska",
                "Chicago Tribune"
            ],
            [
                "Ryan",
                "Pitts",
                "The Spokesman-Review"
            ],
            [
                "Christopher",
                "Groskopf",
                "PANDA Project"
            ]
        ],
        size: 168,
        title: "Contributors"
    }
],
description: "",
id: "1",
initial_upload: "/api/1.0/data_upload/1/",
last_modification: null,
last_modified: null,
last_modified_by: null,
locked: false,
locked_at: "2012-03-29T14:28:02",
name: "contributors",
```

```
    related_uploads: [ ],
    resource_uri: "/api/1.0/dataset/contributors/",
    row_count: 4,
    sample_data: [
        [
            "Brian",
            "Boyer",
            "Chicago Tribune"
        ],
        [
            "Joseph",
            "Germuska",
            "Chicago Tribune"
        ],
        [
            "Ryan",
            "Pitts",
            "The Spokesman-Review"
        ],
        [
            "Christopher",
            "Groskopf",
            "PANDA Project"
        ]
    ],
    slug: "contributors"
}
```

Schema

```
GET http://localhost:8000/api/1.0/dataset/schema/
```

List

```
GET http://localhost:8000/api/1.0/dataset/
```

List filtered by category

```
GET http://localhost:8000/api/1.0/dataset/?category=[slug]
```

List filtered by user

A shortcut is provided for listing datasets created by a specific user. Simply pass the `creator_email` parameter. Note that this paramter can not be combined with a search query or other filter.

```
GET http://localhost:8000/api/1.0/dataset/?creator_email=[email]
```

Search for datasets

The Dataset list endpoint also provides full-text search over datasets' metadata via the `q` parameter.

Note: By default search results are complete Dataset objects, however, it's frequently useful to return simplified objects for rendering lists, etc. These simple objects do not contain the embedded task object, upload objects or sample data. To return simplified objects just add `simple=true` to the query.

```
GET http://localhost:8000/api/1.0/dataset/?q=[query]
```

Fetch

```
GET http://localhost:8000/api/1.0/dataset/[slug]/
```

Create

To create a new Dataset, POST a JSON document containing at least a `name` property to the dataset endpoint. Other properties such as `description` may also be included.

```
POST http://localhost:8000/api/1.0/dataset/

{
    "title": "My new dataset",
    "description": "Lets fill this with new data!"
}
```

If data has already been uploaded for this dataset, you may also specify the `data_upload` property as either an embedded DataUpload object, or a URI to an existing DataUpload (for example, `/api/1.0/data_upload/17/`).

If you are creating a Dataset specifically to be updated via the API you will want to specify columns at creation time. You can do this by providing a `columns` query string parameter containing a comma-separated list of column names, such as `?columns=foo,bar,baz`. You may also specify a `column_types` parameter which is an array of types for the columns, such as `column_types=int,unicode,bool`. Lastly, if you want PANDA to automatically indexed typed columns for data added to this dataset, you can pass a `typed_columns` parameter indicating which columns should be indexed, such as `typed_columns=true,false,true`.

Import

Begin an import task. Any data previously imported for this dataset will be lost. Returns the original dataset, which will include the id of the new import task:

```
GET http://localhost:8000/api/1.0/dataset/[slug]/import/[data-upload-id]/
```

Export

Exporting a dataset is an asynchronous operation. To initiate an export you simple need to make a GET request. The requesting user will be emailed when the export is complete:

```
GET http://localhost:8000/api/1.0/dataset/[slug]/export/
```

You can export only results which match a query by appending the `q` querystring parameter. You can export only results after a certain time by appending the `since` querystring parameter. These may be combined:

```
GET http://localhost:8000/api/1.0/dataset/[slug]/export/?q=John&since=2012-01-01T00:00:00
```

Reindex

Reindexing allows you to add (or remove) typed columns from the dataset. You initiate a reindex with a GET request and can supply `column_types` and `typed_columns` fields in the same format as documented above in the section on creating a Dataset.

```
GET http://localhost:8000/api/1.0/dataset/[slug]/reindex/
```

7.2.8 Data

Data objects are referenced by a unicode `external_id` property, specified at the time they are created. This property must be unique within a given `Dataset`, but does not need to be unique globally. Data objects are accessible at per-dataset endpoints (e.g. `/api/1.0/dataset/[slug]/data/`). There is also a cross-dataset Data search endpoint at `/api/1.0/data/`, however, this endpoint can only be used for search–not for create, update, or delete. (See below for more.)

> **Warning:** The `external_id` property of a Data object is the only way it can be referenced via the API. In order to work with Data via the API you **must** include this property at the time you create it. By default this property is `null` and the Data can not be accessed except via search.

An example Data object with an `external_id`:

```
{
    "data": [
        "1",
        "Brian",
        "Boyer",
        "Chicago Tribune"
    ],
    "dataset": "/api/1.0/dataset/contributors/",
    "external_id": "1",
    "resource_uri": "/api/1.0/dataset/contributors/data/1/"
}
```

An example `Data` object **without** an `external_id`, note that it also has no `resource_uri`:

```
{
    "data": [
        "1",
        "Brian",
        "Boyer",
        "Chicago Tribune"
    ],
    "dataset": "/api/1.0/dataset/contributors/",
    "external_id": null,
    "resource_uri": null
}
```

> **Warning:** You can not add, update or delete data in a **locked** dataset. An error will be returned if you attempt to do so.

Schema

There is no schema endpoint for Data.

List

When listing data, PANDA will return a simplified Dataset object with an embedded `meta` object and an embedded `objects` array containing Data objects. The added Dataset metadata is purely for convenience when building user interfaces.

```
GET http://localhost:8000/api/1.0/dataset/[slug]/data/
```

Search

Full-text queries function as "filters" over the normal `Data` list. Therefore, search results will be in the same format as the list results described above:

```
GET http://localhost:8000/api/1.0/dataset/[slug]/data/?q=[query]
```

For details on searching Data across all Datasets, see below.

Fetch

To fetch a single Data object from a given Dataset:

```
GET http://localhost:8000/api/1.0/dataset/[slug]/data/[external_id]/
```

Create and update

Because Data is stored in Solr (rather than a SQL database), there is no functional difference between Create and Update. In either case any Data with the same `external_id` will be overwritten when the new Data is created. Because of this requests may be either POSTed to the list endpoint or PUT to the detail endpoint.

An examplew with POST:

```
POST http://localhost:8000/api/1.0/dataset/[slug]/data/

{
    "data": [
        "column A value",
        "column B value",
        "column C value"
    ],
    "external_id": "123456"
}
```

An example with PUT:

```
PUT http://localhost:8000/api/1.0/dataset/[slug]/data/123456/

{
    "data": [
        "new column A value",
        "new column B value",
        "new column C value"
    ]
}
```

Bulk create and update

To create or update objects in bulk you may PUT an array of objects to the list endpoint. Any object with a matching `external_id` will be deleted and then new objects will be created. The body of the request should be formatted like:

```
{
    "objects": [
        {
            "data": [
                "column A value",
                "column B value",
                "column C value"
            ],
            "external_id": "1"
        },
        {
            "data": [
                "column A value",
                "column B value",
                "column C value"
            ],
            "external_id": "2"
        }
    ]
}
```

Delete

To delete an object send a DELETE request to its detail url. The body of the request should be empty:

```
DELETE http://localhost:8000/api/1.0/dataset/[slug]/data/[external_id]/
```

Delete all data from a dataset

In addition to deleting individual objects, its possible to delete all Data within a Dataset, by sending a DELETE request to the root per-dataset data endpoint. The body of the request should be empty.

```
DELETE http://localhost:8000/api/1.0/dataset/[slug]/data/
```

7.2.9 Global search

Searching all data functions slightly differently than searching within a single dataset. Global search requests go to their own endpoint:

```
http:://localhost:8000/api/1.0/data/?q=[query]
```

The response is a `meta` object with paging information and an `objects` array containing simplified Dataset objects, each of which contains its own `meta` object and an `objects` array containing Data objects. **Each Dataset contains a group of matching Data.**

When using this endpoint the `limit` and `offset` parameters refer to the Datasets (that is, the **groups**) returned. If you wish to paginate the result sets within each group you can use `group_limit` and `group_offset`, however, this is rarely useful behavior.

- Source code repository
- Issue tracker
- Project wiki

CHAPTER

EIGHT

AUTHORS

The PANDA Board:

- Brian Boyer
- Joe Germuska
- Ryan Pitts

Lead Developer:

- Christopher Groskopf

Contributors:

- Dane Springmeyer
- David Eads
- Niran Babalola
- Justin Edwards

CHAPTER

NINE

LICENSE

CHAPTER

TEN

CHANGELOG

10.1 1.0.0

- Proper error messages when a search fails. (#600)
- Upgrade to Django 1.4.1 security release.
- Wrote new user documentation at pandaproject.net.
- Remove nginx file size upload limitation. (#832)
- Stale data in Solr indexes will no longer break search. (#793)
- Add form for adding users in bulk to admin. (#762)
- Fix broken Change Password and Logout buttons on admin pages. (#795)
- Implement global search results export. (#788)
- Increase uWSGI buffer size to prevent overflows. (#792)
- Fix emails so they can be longer than 30 characters. (#789)
- Add “Related Links” section to dataset pages. (#763)
- Make progress bars degrade gracefully in IE. (#426)
- Ensure new pages always begin scrolled to the top. (#790)
- Fix broken export links in notifications. (#783)
- Fix so that hitting enter in modal dialogs submits the form. (#781)
- Add descriptions for data and related uploads. (#757)
- Fix major bug in IE8 with handling encoding errors. (#688)
- Upgrade all deployments to Ubuntu 12.04 LTS release. (#684)
- User login cookies now last 30 days. (#639)
- Reword “Column filters” to be easier to understand. (#591)
- Major performance improvements for user pages. (#778)
- Automatically match nicknames in search. (#761)
- Search for data within dataset categories. (#760)
- Add links to data exports to user pages. (#759)
- Implement “Text” column filter. (#758)

- Make modal dialogs focus first form field on open. (#674)
- Fix display of notifications in navbar. (#749)
- Datasets are no longer locked for editing during exports. (#659)
- PANDA instances now have unique SECRET_KEYS for added security. (#465)
- A password confirmation is now required to edit a user profile. (#677)
- Datasets with aborted imports no longer appear broken. (#705)
- Add refresh page link to dataset import status alerts. (#738)
- Fix search subscription text when not using any full-text query (#747)
- Add delete button for data uploads (also deletes data imported from that upload). (#754)
- Improve appearance of the dashboard. (#765)
- Eliminate default accounts and demo mode. (#769)
- Implement "Welcome to PANDA" setup page. (#752)

10.2 0.2.0

- Implement automated search notifications. (#703)
- Add landing pages for exports so they use the login logic. (#711)
- Prompt for metadata during upload. (#710)
- Add a human-readable description to tasks in the admin. (#709)
- Fix longstanding bug where task error notifications may fail. (#700)
- Add row count to import and reindex notification emails. (#697)
- Fix filters so it is possible to search for a zero. (#696)
- Add "Resend activation code" button to User admin pages. (#690)
- Escape dataset names on User pages. (#685)
- Reorganize import errors to be more readable. (#617)
- Don't disable dataset search during import. (#483)
- Update dataset full text when a user's name is changed via the admin. (#351)
- Update dataset full text when a user's name is changed via the API. (#350)
- Handle SMTP connection errors gracefully. (#335)
- Normalize API behavior for Notications. (#237)
- Fix bug where column filters could not be searched alone. (#694)
- Fix bug where column filter ranges would not display. (#695)
- Warn when uploading a file if low on storage space. (#693)
- Add "Abort" link to TaskStatus admin page. (#419)

10.3 0.1.4

- Added “Forgot Password?” link. (#536)
- Added setting to explicitly enable/disable email. (#680)
- Pushing enter in a modal form now submits it. (#577)
- Fix broken “expand search to all categories” link. (#673)
- Improved usability of notifications. (#666)
- Fix unescaped quotes in searches breaking pagination. (#665)
- Fix “Browse all datasets” link on homepage. (#662)
- Fix major bug with how search filters are applied. (#661)
- Export cross-dataset search results. (#657)
- Export single-dataset search results. (#641)
- Users can now change their password. (#299)
- Users can now update their email address. (#652)
- Users can now update their name. (#150)
- Per-user pages instead of email links. (#650)

10.4 0.1.3

- Add documentation links to the header of the app. (#623)
- Reorganize documentation so users come first. (#624)
- Add metrics dashboard.
- Fix long-standing bug where dataset metadata might not get created. (#618)
- Intelligently handle currency symbols in numeric columns. (#538)
- Ensure related files are deleted when datasets are. (#612)
- Implement dataset paging. (#609)
- Overhaul datasets search urls.
- Use human-readable data types. (#588)
- Fix bug preventing new users from activating. (#614)
- Fix bug preventing new users from being created. (#614)
- Write user documentation. (#601)
- Fix bug where an empty cell in a datetime column could break import. (#608)
- Escape dataset names in email. (#610)

10.5 0.1.2

- Revise release process to prevent docs from referencing the master branch.
- Fix bug where users could not login if their username was changed. (#511)
- Wrote documentation for connecting via SSH.
- Made max upload size configurable via admin settings. (#508)
- Upgraded to Django 1.4. (#561)
- Fixed escaping in dataset titles displayed in notifications (#598)
- Doubled the amount of memory allocated to Solr.
- Made Solr much less likely to run out of memory during queries.

10.6 0.1.1

- Fix bug where quotes in dataset titles could break the view page. (#570)
- Fig bug where a dataset search url with no query would result in a 404. (#569)
- Prevent datasets created via the API from having duplicate slugs. (#550)
- Fix an issue where empty columns in a CSV could crash dataset creation.
- Make it possible to specify column types via API at dataset creation time. (#518)
- Import and search specific data columns (search filters). (#494, #506, et al)
- Implement a configurable throttle for asynchronous tasks. (#502)
- Added Justin Edwards to AUTHORS.
- Added favicon.ico. (#512)
- Better handling of timeout errors during file uploads. (#529)
- Implement ability to import specific columns with type data, i.e. filters. (#496)
- Fix bug where blank headers in XLSX files would be labelled “None”.
- Fix “unpack requires string length of 14” error during xls upload. (#486)
- Implement locks so import/export processes can not occur simultaneously. (#353)
- Improve handling of queries with inline punctuation. (#461)
- Fix error when adding a category via Admin. (#470)
- Add Niran Babalola to AUTHORS. (#464)
- Suggest deactivating instead of deleting users. (#464)
- Add “view all rows” link to dataset page. (#455)
- Fix bug that caused changes to a dataset made during import to be lost. (#452)
- Add feedback link. (#451)

10.7 0.1.0

- Initial beta release.

CHAPTER

ELEVEN

SEARCH THIS DOCUMENTATION

- *search*