Yabi Documentation

Release

Centre for Comparative Genomics, Murdoch University

February 06, 2016

Contents

CHAPTER 1

About Yabi

Yabi is a 3-tier application stack to provide users with an intuitive, easy to use, abstraction of compute and data environments. Developed at the Centre for Comparative Genomics, Yabi has been deployed across a diverse set of scientific disciplines and high performance computing environments.

1.1 Key features

Yabi has a few key features:

- simplified web based access to High Performance Computing
- easy tool addition and maintenance
- handling of disparate compute and storage resouces ie. PBSPro, SGE, Torque, SSH, SFTP, Amazon S3, Swift
- easy and powerful workflow creation environment

1.2 A brief history

The Centre for Comparative Genomics has been addressing the problem of facilitating transparent access to HPC for over a decade. As technologies have advanced we have gone through various iterations to address this issue. Now we have drawn on our experience in HPC and Internet based solutions to produce Yabi, an intuitive abstraction of compute and data environments.

1.3 Architecture

The Yabi architecture consists of three main components:

- The Client (typically a web browser or yabish)
- The Yabi Application
- The Yabi Backend (Yabi Resource Manager)

1.3.1 Client

The client is typically a web browser, although a command line client also exists (yabish). The command line client interacts with the front end application in the same way as a web browser, that is:

- cookies are used to maintain a user session
- all traffic is via HTTPS
- users are required to log in to activate a session

1.3.2 Yabi Application

The Yabi application is a Python web application running under Apache 2 via mod_wsgi. HTTP and HTTPS are required, although the application will insist on HTTPS and redirect any HTTP requests to HTTPS. The application runs under Apache using mod_wsgi so does not require any additional accounts, privileges or ports to be created/opened.

The Yabi application is intended to be run on an Internet facing server as it serves the HTML/CSS/Javascript application that users typically interact with as well as a REST style interface for the command line client. Naturally, it can be deployed on an internal network if access over the Internet is not desired.

1.3.3 Yabi Backend

The Yabi Backend Server makes use of the Celery Task Engine to queue user jobs before distribution to the appropriate execution backend via a message broker (typically RabbitMQ but this is configurable.)

The Yabi Backend server is responsible for the communication with individual data and compute resources.

1.4 Source code

The Yabi source code is available on Bitbucket: https://bitbucket.org/ccgmurdoch/yabi.

1.5 Licence

GNU GPL v3 for non-commercial use only. Please contact the Centre for Comparative Genomics if you require a licence other than GPL for legal or commercial reasons.

1.6 Authors

Yabi is developed at the Center for Comparative Genomics at Murdoch University under the direction of Professor Matthew Bellgard. Developers working on the project are Adam Hunter, Tamas Szabo and Lee Render with acknowledgements to Andrew Macgregor and Crispin Wellington.

1.7 Acknowledgements

We would like to acknowledge the valuable contributions from the following groups and individuals:

- Dr Paula Moolhuijzen, Adam Harvey, Nick Takayama, David Schibeci, Mark O'Shea and Brett Chapman, Centre for Comparative Genomics (CCG).
- Dr Cas Simons, Pierre-Alain Chaumeil, Nick Rhodes, Queensland Facility for Advanced Bioinformatics (QFAB)

- Professor Andreas Wicenec, Research Associate Professor Kevin Vinsen, International Centre for Radio Astronomy Research (ICRAR)
- Australian Research Collaboration Service
- Australian National Data Service
- Bioplatforms Australia
- National Collaborative Research Infrastructure Strategy Program
- Education Investment Fund Super Science Initiative
- iVEC

CHAPTER 2

Installation

We currently install Yabi on CentOS servers which is the supported platform. There is nothing preventing you to install Yabi on any other Linux distribution, but installing on CentOS is easiest using the RPMs we provide.

Theoretically, you could install on any platform that can run a Python WSGI application and Celery. Additionally you will need access to a database server (Postgresql recommended or MySQL), a message broker (we use and recommend RabbitMQ), and Memcached.

2.1 Components

As it was already mentioned in *Architecture* Yabi consists of three main components. Of those at least the Yabi web application and the Yabi Backend has to be installed on the server. The yabish command line client can also be installed on user machines if required.

The Yabi Backend is running the Celery worker processes that among other things are responsible for file operations. In case you have a lot of large files that are transferred the responsiveness of the web application could be effected.

That being said the simplest method of installation is to install both the Yabi web application and the Yabi Backend on the same server. We use this setup on all our servers and we recommend you start with the same because it is simpler and you can always optimise later if required.

The Yabi web application is a Django application. It needs a WSGI container and a web server to serve static files. We currently use and recommend Apache + mod_wsgi, if you use our RPMs you would be using the same.

The Yabi Backend is based on Celery and it is recommended that you use a dedicated message broker.We use and recommend RabbitMQ, installation instructions are provided for CentOS.

Both the Yabi web application and the Yabi Backend make use of a shared database. As a database server we use and recommend Postgresql. We assume you already have your database server installed. We do provide instructions on how to set up the Yabi database on an existing database server.

Yabi also uses Django caching and although any caching method supported by Django should work we use and recommend Memcached. We assume you have your Memcached servers already installed, we provide instructions on how to set up Yabi to use them.

Yabi sessions are also stored in Memcache. You can do the same or opt for another session backend supported by Django.

2.2 How to install

In the *how to install* section we assume that you are going to install Yabi on CentOS using our provided RPMs and that both the web application and the Yabi Backend are going to be installed on the same server.

2.2.1 Prerequisites

Extra CentOS repositories

The EPEL, IUS and CCG rpm repositories are needed to install packages that are not in the CentOS base repositories.

The way to add the EPEL and IUS repositories, depends on the version of CentOS you're having. The following example is for CentOS 6.6, for other versions of CentOS please follow the instructions at IUS Repos:

```
# yum install https://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
# yum install https://dl.iuscommunity.org/pub/ius/stable/CentOS/6/x86_64/ius-release-1.0-14.ius.cento
```

To add the CCG repository:

```
# yum install http://repo.ccgapps.com.au/repo/ccg/centos/6/os/noarch/CentOS/RPMS/ccg-release-6-2.noar
```

RabbitMQ

We recommend using RabbitMQ as Celery's message broker. RabbitMQ requires Erlang:

```
# yum install erlang
```

To install RabbitMQ:

```
# rpm --import https://www.rabbitmq.com/rabbitmq-signing-key-public.asc
# yum install https://www.rabbitmq.com/releases/rabbitmq-server/v3.5.3/rabbitmq-server-3.5.3-1.noarch
```

Start the RabbitMQ service with:

```
# service rabbitmq-server start
```

2.2.2 Installation of the Yabi web application under Apache

The IUS repository provides a `httpd24u` package that unfortunately conflicts with `httpd`. Therefore if you try to install `yabi-admin` and you don't have one of the `httpd` packages already installed you will get a conflict error. The recommended way (in the email announcing httpd24u) to get around this problem is to install the httpd package first and only after that install yabi-admin:

```
# yum install httpd mod_ssl
# yum install yabi-admin
```

This will add an Apache conf file to `/etc/httpd/conf.d` called `yabiadmin.ccg`. Please feel free to read through it and edit if required. When you are happy with the contents create a symbolic link for Apache to pick this config up automatically:

```
# pushd /etc/httpd/conf.d && ln -s yabiadmin.ccg yabiadmin.conf && popd
```

2.2.3 Database Setup

We assume that you have a database server (preferably Postgres) installed, that is accessible from the server you're performing the Yabi installation on.

Create the Yabi database

Please create a database (ex. `yabi_prod`) that will be used by the Yabi application. We recommend creating a user called `yabiapp` with no special privileges that will be the owner of the database.

Configure Yabi to use Yabi database

To change the database that Yabi points at you will need to alter the Yabi settings file. Open up `/etc/yabi/yabi.conf` in your editor and make sure the variables in the *database options* section (`dbserver`, `dbname`, `dbuser` etc.) are set correctly.

For more details see settings.

Initialise the Yabi database

To initialise the database you will have to run the Django migrations in the Yabi project:

```
# yabi migrate
```

These will create the schema, insert setup data, and create initial users.

Make sure Apache can connect to the database

SELinux on CentOS systems will prevent Apache to connect to any network services by default. Yabi needs to be able to connect to your database, so you will need to set at least the first of the following SELinux booleans to on:

httpd_can_network_connect_db (HTTPD Service) Allow HTTPD scripts and modules to network connect to databases.

httpd_can_network_connect (HTTPD Service) Allow HTTPD scripts and modules to connect to the network.

2.2.4 Caching and Sessions

By default Yabi uses database caching and file-based sessions but we recommend using memcached for both.

Changing both caching and session to memcached is therefore easy. Assuming you already have one or more memcached servers ready to go, all you need to do is open `/etc/yabiadmin/yabiadmin.conf` in your editor and set the `memcache` variable to a space-separated list of memcache servers.

This will make Yabi switch to memcached for both caching and sessions.

See:

- settings
- Django caching
- Django sessions

2.2.5 Last steps

At this stage we should have everything installed, the database, caching and sessions configured.

As a last step before starting the applications you should go through all the variables in `/etc/yabiadmin/yabiadmin.conf` and make sure everything is set to sensible values. See :ref:`settings`.

Restart apache

To start up the Yabi web application restart Apache:

```
# /etc/init.d/httpd restart
```

Start Celery

To start up the Yabi Backend start Celery:

```
# /etc/init.d/celeryd start
```

At this stage you should be able to access the Yabi web application by browsing to https://YOURHOST/yabi/.

The Yabi default installation creates two users *demo* and *admin*, where *demo* is a normal user and *admin* is Yabi administrator. The password for *demo* is *demo* and for *admin* is *admin*.

In order to test the system, log in with the *demo* user and create a workflow that has one job running the `hostname` tool. You should be able to submit this workflow and it should run to completion.

2.2.6 Setting up Yabish

Yabish is the command line client that you might want to install on machines used by end users of Yabi.

Make sure you've added the CCG repo as described in *Extra CentOS repositories*. Then install the `yabi-shell` package:

```
# yum install yabi-shell
```

This installs the CLI for Yabi in `/usr/bin/yabish`.

CHAPTER 3

Logging

Yabi uses the standard Python and Django logging mechanisms and is controlled by the logging section in the settings. If you wish to add file logging for instance you would need to add a file logging handler to the handlers section and then add name of the handler into the yabi logger section. For further details consult these pages:

http://docs.python.org/library/logging.html

https://docs.djangoproject.com/en/dev/topics/logging/

CHAPTER 4

Testing

4.1 Running Tests

Yabi has an end to end test suite for testing the full Yabi stack. The tests can run against Postgresql or MySql. You can run them by:

```
$ ./develop.sh test_postgresql
```

to run them against Postgresql or

```
$ ./develop.sh test_mysql
```

to run them against MySql.

4.1.1 Lint

To lint all the Python code in Yabi run:

```
$ ./develop.sh lint
```

To lint all the JavaScript code in Yabi run:

```
$ ./develop.sh jslint
```

CHAPTER 5

Yabi Authentication

Yabi currently supports the following authentication methods:

- Database Authentication - this is the default
- LDAP Authentication
- Kerberos and LDAP Authentication

5.1 Database Authentication

When using database authentication users and all their details (including passwords) are stored in the database. This is very similar to Django's ModelBackend with the only difference that Yabi usernames are case insensitive.

This is the default authentication, therefore to use this you don't have to configure anything. You will have to manually create all users using the Yabi Administration interface. See *Adding Users*.

5.2 LDAP Authentication

This setup is a good choice if you already have users set up in a central LDAP repository and you would like them to use the same password to log in to Yabi. Another benefit is that you won't have to manually add all users to Yabi.

You will need 2 special LDAP User Groups, the Yabi Users group and the Yabi Administrators group. Members of the Yabi User group will have access to Yabi, members of the Yabi Administrators group will have access to Yabi and the Yabi Administration Interface.

After you've created these 2 groups, adding users to Yabi is as simple as adding your LDAP users to one of these groups. Yabi user accounts will be automatically created in the database when the user logs in the first time into Yabi. Additionally the user's name and email is also fetched from LDAP and set in the database.

Group membership can be defined either on the user object or on the group object. The user objects can have their `AUTH_LDAP_MEMBER_OF_ATTR` attribute (typically *"memberOf"* or similar) to the DN of a group(s) they are member of. From the other end the group object can have their `AUTH_LDAP_MEMBER_ATTR` attribute (typically *"uniqueMember"*, or *"member"*) to the DN of the user(s) that are members of this group. As long as these settings are set correctly Yabi will check both ends to check for group membership, therefore you can have some users having their groups set on the user object, others on the group object etc.

To set up LDAP authentication you will have to configure the following settings in your settings.

5.2.1 Required settings:

Setting	Description
AUTH_TYPE	Authenticaton type. Set it to *"ldap"*.
AUTH_LDAP_SERVER	List of at least 1 ldap servers. Servers will be tried in order. Ex. *["ldaps://ldap1.your.domain", "ldap://ldap2.your.domain"]*
AUTH_LDAP_USER_BASE	Parent DN of place where your users are stored. Ex. *"ou=People, dc=your_domain"*
AUTH_LDAP_YABI_GROUP_DN	DN of the Yabi user group. Ex. *"cn=Yabi, ou=Groups, dc=your_domain"*
AUTH_LDAP_YABI_ADMIN_GROUP_DN	DN of the Yabi Administrators user group. Ex. *"cn=Yabi Administrators, ou=Groups, dc=your_domain"*

5.2.2 Optional settings:

All these settings but `AUTH_LDAP_SYNC_USER_ON_LOGIN` are provided for cases when your LDAP config differs from the standard, therefore in most cases you wouldn't want to change them.

Setting	Description
AUTH_LDAP_SYNC_USER_ON_LOGIN	Each time a user logs in fetch their details (name, email, is a Yabi Administrator) and set it in the database. Otherwise setting the database values from LDAP happens only the first time the user logs in. Default is `True`.
AUTH_LDAP_USER_FILTER	LDAP search filter used when searching for your users in AUTH_LDAP_USER_BASE. Default is *"(objectclass=person)"*.
AUTH_LDAP_MEMBER_ATTR	LDAP group attribute used to add members to a group. Default is *"uniqueMember"*.
AUTH_LDAP_MEMBER_ATTR_USER_ATTR	The user's object attribute that is stored in the AUTH_LDAP_MEMBER_ATTR above. Can be *"username"* or *"dn"*. Default is *"dn"*.
AUTH_LDAP_MEMBER_OF_ATTR	LDAP user attribute used to add members to a group. Default is *"memberOf"*.
AUTH_LDAP_USERNAME_ATTR	LDAP user attribute for username. Default is *"uid"*.
AUTH_LDAP_EMAIL_ATTR	LDAP user attribute for email. Default is *"mail"*.
AUTH_LDAP_LASTNAME_ATTR	LDAP user attribute for last name. Default is *"sn"*.
AUTH_LDAP_FIRSTNAME_ATTR	LDAP user attribute for first name. Default is *"givenName"*.
AUTH_LDAP_REQUIRE_TLS_CERT	Require server to have a valid TLS certificate. Default is True.

5.3 Kerberos and LDAP Authentication

This authentication method is very similar to the LDAP authentication method above. The only difference is that the username and password will be checked against your Kerberos server not your LDAP server.

After that everything described in LDAP authentication applies. Users will have to be members of the Yabi and/or Yabi Administrator LDAP groups to be able to log in into Yabi. The database user accounts will be created automatically on first login, the user detail will be fetched from LDAP and set in the database at that time and on each login if `AUTH_LDAP_SYNC_USER_ON_LOGIN` is set.

To set up *Kerberos and LDAP* authentication you will have to configure all the settings in the LDAP authentication section **and** the following setting in your settings.

Setting	Description
AUTH_TYPE	Authentication type. Set it to *"kerberos+ldap"*.
AUTH_KERBEROS_REALM	Your Kerberos realm. Ex. *"CCGMURDOCH"*.

Note: For Kerberos Authentication to work properly the Kerberos must be configured properly on this machine. That will likely mean ensuring that the edu.mit.Kerberos preference file has the correct realms and KDCs listed.

5.4 Fallback on Database Authentication

When the setting `AUTH_ENABLE_DB_FALLBACK` is set to `True` (default), your users will always be authenticated against the database as the last step, if the main authentication method failed.

This feature can be useful to avoid being locked out of Yabi if there is some temporary problem with your LDAP or Kerberos server. In case you have an admin user in the Database you will always be able to log in into Yabi using that user if `AUTH_ENABLE_DB_FALLBACK` is set to `True`.

CHAPTER 6

Adding Users

To add users to the Yabi system you will need to be logged into the Adminstration part of the application.

6.1 Adding Regular Users

At the main page you will see a list of all tables available for editing. Of interest are the Auth User table and the Yabi User table.

The Auth User is the main authentication table used by Django Auth to store user details. Add a record to this table ensuring that the Active checkbox is ticked and the Staff and Superuser checkboxes are **not** ticked.

When you create an Auth User record you will find that a corresponding Yabi User record is automatically created. This is where additional user information can be stored. If you edit the Yabi User record you will see two options:

- User option access - controls whether the Account and Change Password screens are available to the user
- Credential access - determines whether the Credential modification screen is available to the user

6.1.1 Using Different Authentication Backends

If you are using a different authentication method such as LDAP or "Kerberos and LDAP" you don't have to add users manually to the Yabi. After you set up Yabi to use one of these authentication methods all you have to do is add the user to the Yabi or Yabi Administrators LDAP group and their user account will be created automatically the first time they log in into Yabi.

See *Yabi Authentication* for more detail on setting up LDAP or "Kerberos and LDAP" authentication.

6.2 Adding Adminstrator Users

To add an adminstrator follow the steps above to add a regular user and the also ensure that the Staff and Superuser checkboxes are ticked.

If you wish to you could use the Django Permissions system to give finer grained permissions to users i.e. permission to add tools only.

6.3 Viewing a User's Setup

There are a couple of admin tools that allow an administrator to view a users setup. First click on Users under the Yabi section (**not** under the Auth section). This will show you a list of users you have added to Yabi. Clicking on the `Tools` or `Backend` links will give you further detail.

6.3.1 User Tools

This listing will show you all the tools a user has access to and the tool groups the tools belong to.

6.3.2 User Backends

This listing will show you all the execution and filesystem backend credentials set up for the user.

CHAPTER 7

Backends

In Yabi there are two types of backend - execution and storage. As the names suggest, an execution backend is where tasks will be run, and a storage backend is where files are stored.

Both types of backends can be static or dynamic. Dynamic backends are automatically created by Yabi for example using Amazon Web Services, Open Stack or other providers. see *Dynamic Backends*.

To get things up and running the best idea is to set up your first backend to a resource that you can connect to via SSH. Before trying to add the backend to Yabi you should be able to connect to it via the command line using ssh keys from the box running Yabi.

7.1 Setting up an SSH File System Backend

Click on Backends under Yabi heading and add a backend, using *scp* as the scheme. For example you would fill in the fields like this:

Name	Example File Server
Description	My example file server
Dynamic backend	Leave unchecked or see Dynamic Backends
Dynamic backend configuration	Leave unchecked or see Dynamic Backends
Scheme	scp
Hostname	exampleserver.localdomain
Port	22
Path	/home/

7.1.1 A note about the Path field

Yabi uses the information in the Backend record and the Backend Credential record to construct a URI that it uses to access data or execution resources. In setting up this backend I am adding `/home/` to the path. This is where all the Yabi users on this machine will find their home directories. So the backend can construct a URI from the scheme, hostname and path fields that will look like this:

```
scp://exampleserver.localdomain/home/
```

In a later step (*Backend Credentials*) we will setup the link between Backends and a Users Credentials. In that step we will add to the *User Directory* field `andrew/yabi/`. This leads to my home directory (`/home/andrew/`) and then limits Yabi access to a directory within it (`yabi/`).

So when Yabi needs to access files on that backend it combines the fields from the Backend and the Backend Credential records to derive the URI:

```
scp://exampleserver.localdomain/home/andrew/yabi/
```

Please note: While Yabi refers to the directory on the Backend Credential record as User Directory this can really point to any directory the user has access to.

7.1.2 Additional Fields

The *Lcopy* and *Link* checkboxes come into play if an execution and storage backend have a shared filesystem. You can set this backend up to use local copy and link, then later specific tools can request that lcopy or link be used.

7.2 Adding an Execution Backend

Next we are going to add an execution backend so we can run some jobs. In this example we are going to use pbspro qsub via ssh. In this way we do not need to have torque and qsub set up on the same box as Yabi. Instead we just need to have our ssh credentials (*Backend Credentials*) set up to access the execution host.

Again, click on Backends under Yabi heading and add a backend, this time using *ssh+pbspro* as the scheme. For example you would fill in the fields like this:

```
Name                          Example Execution Server
Description                   My example execution server
Dynamic backend               Leave unchecked or see Dynamic Backends
Dynamic backend configuration Leave unchecked or see Dynamic Backends
Scheme                        ssh+pbspro
Hostname                      exampleserver.localdomain
Port                          22
Path                          /
```

7.2.1 Submission Template Field

On an execution backend you can add a template for any submission scripts that will be used. Please see *Submission Templates*.

7.3 Troubleshooting SSH

Take a look at these FAQ

- *I'm having trouble setting up an SSH backend, what am I doing wrong?*

7.4 Null Backend

The null backend is used by tools that should not be executed, such as a file selection tool. A null backend is created automatically for every new installation of Yabi, but in case you have to create one yourself you can do it by using these values:

```
Name            Null Backend
Description     Use this null backend when tools should not be executed.
Scheme          null
Hostname        localhost.localdomain
Port
Path            /
```

7.5 S3 Backend

An S3 filesystem backend can be created by using the schema `s3`. The domain of the hostname should be set to `amazonaws.com` and the hostname to a S3 bucket name.

For example `mybucket.amazonaws.com` as the hostname will access the bucket `mybucket` on amazon.

In setting up the credential for access to S3, your remote username is ignored, so you can place any text in here you like. You will need to fill in two fields: password and key. Into the yabi key field put the Amazon ACCESS ID and into the yabi password field put the Amazon SECRET KEY.

7.6 OpenStack Swift Backend

Yabi can use OpenStack Object Storage (commonly known as *Swift*) as a filesystem backend. Swift is similar to S3 in that it is a key-value store, not a heirachical file system. Yabi will present the keys in a directory tree, using forward slashes (`/`) to separate directory paths.

The OpenStack cluster must use Keystone 2.0 auth. To set up the Swift backend, set its hostname to the hostname part of the Keystone API endpoint. For example, if the Keystone auth URL is `https://keystone.bioplatforms.com/v2.0/`, then the backend hostname will be `keystone.bioplatforms.com`.

OpenStack users are associated with projects (also called "tenants"). In Swift, files are collected in "containers" which belong to the project. Each project has its own set of containers. The backend path must specify both the project and container.

Backend Option	Setting
Name	OpenStack Swift
Description	OpenStack object storage
Scheme	`swift`
Hostname	*The hostname part of the Keystone API endpoint.*
Port	*Not required, defaults to 443*
Path	`/tenant/container/`
...	*All other options left blank.*

When creating a credential entry for the Swift backend, use Keystone credentials.

Credential	Setting
Description	OpenStack Keystone credentials for *user name*.
Username	The Keystone user name.
Password	The Keystone password.
Cert	
Key	
User	The Yabi user.
Expires on	A date in the future.

CHAPTER 8

Credentials

Any tasks carried out by Yabi are always run **as the user** not as Yabi. For this reason Yabi needs to be able store and make use of user credentials.

8.1 A note about encryption

All credentials are stored encrypted within the Yabi system. This includes within the database and within memcache or other caching mechanisms. When you enter details into the Add Credential screen in plain text, they are encrypted before they are added to the database.

Encryption uses one of the following. If data has been newly added it is encrypted using the SECRET_KEY in the `settings.py` file (see settings). As soon as a user logs in any of their credentials that are encrypted with the SECRET_KEY are decypted and stored re-encrypted using their Yabi password.

8.2 How credentials are used by the backend

When a user logs into Yabi, their password is used to decrypt their credentials, which are then re-encrypted with the SECRET_KEY from the `settings.py` file and temporarily stored in memcache or other caching system. When the backend then needs access to a resource, it is able to get the credential from memcache and decrypt it with the SECRET_KEY so it can use it.

In this way the credentials are stored **encrypted** both permanently (in database) and temporarily (in memcache).

8.2.1 Adding Credentials

In the Yabi Credential table select Add Credential and fill in the form for each user's credential. Not all the fields will be needed, depending on the credential type.

Description: A name or description to identify the credential to the administrator.

Username: This is the username on the backend that the credential matches. This may be different than the Yabi username. For example in Yabi my username may be andrew while on a cluster my username may be amacgregor.

Password: The password for the credential, either for the key itself or for the backend matching this credential.

Cert: The certificate for the matching backend.

Key: The key for the matching backend.

User: The Yabi user that this credential belongs to.

Expires On: An expiry date for the credential. At the moment the expiry date is required. If the key does not have one just pick a date far in the future.

Credentials for SSH and SCP Backends

The SSH protocol supports a number of ways to authenticate yourself to a remote host. This simplest is by a password. You can also authenticate via a RSA or DSS private/public keypair. The private side of this keypair has the option of being protected by a passphrase. All of these authentication styles are supported by yabi via the same Credential object. The way that is attempted is dependent on which fields of the Credential object are filled. The following applies:

Authentication Type	Password field	Cert field	Key field
Password	Account password	empty	empty
Passphraseless key	empty	empty	the private keyfile contents
Passphrase protected key	Key passphrase	empty	the private keyfile contents

8.2.2 Credential Actions

On the screen listing all Credentials there are several actions that can be taken using the Action dropdown near the top left of the screen.

Cache: Will decrypt credentials and cache them in memcache. NB: You must know the login password that has encrypted the credentials.

Delete: Will delete credentials.

Duplicate: Will make a copy, again you must know the login password that has encrypted the credentials.

Purge: Will remove the credentials from memcache.

CHAPTER 9

Backend Credentials

Backend Credentials is a linking table between a Credential and a Backend, used simply because a user may have one credential (say an SSH key) that is valid across multiple backends.

9.1 Adding Backend Credentials

- Add a Backend Credential and select the Backend and associated Credential.
- Add the users home directory or subdirectory (see *A note about the Path field* for details).
- Check *Visible* so the backend will appear to the user.
- Select Default Stageout to indicate that this user's job results should stageout to this filesystem

9.2 Default Stageout Explained

Yabi uses Default Stageout as a single directory where all the user's results will be staged out to. Each user should only have one default stageout, usually on a file server they have access to from outside of Yabi i.e. via ssh

9.3 What the user will see

At this point if everything has worked the user should be able to log into Yabi and under the Files tab they will be able to see and view files on the backends you have set up.

CHAPTER 10

Dynamic Backends

Dynamic Backends are backends that are created automatically by Yabi before running a job.

Tools can be configured to run on Dynamic Backends. When Yabi is about to run a Job of a tool that has been configured with a Dynamic Backend it will automatically create the server, it will run the Job, then it will destroy the server. The servers are created based on dynamic backend configurations created using Yabi admin.

Currently only AWS EC2 instances (both On-Demand and Spot) are supported, but we plan on adding support for more providers.

10.1 AWS Credentials

The AWS credentials used to create your AWS EC2 instances have to be provided to Yabi.

Please set the `aws_access_key_id` and the `aws_secret_access_key` in your environment or config file for prod as described in settings.

10.2 OpenStack Credentials

Similarly, to create OpenStack Nova instances you will have to provide your OpenStack credentials used to create those instances.

Please set the `openstack_user` and the `openstack_password` in your environment or config file for prod as described in settings.

10.3 Dynamic Backends in Admin

Dynamic Backends are created in Admin similarly to other Backends in Admin. The differences are that you have to mark the Backend dynamic by checking the `Dynamic backend` checkbox and you have to choose a `Dynamic backend configuration`. The `Dynamic backend configuration` will be used to create the Dynamic Backend.

The hostname of the backend can't be know at this time, because it will be assigned dynamically on creation. Therefore it is recommended set `Hostname` to `DYNAMIC`.

10.4 Dynamic Backend Configurations

Dynamic Backend Configuration have to be created through Yabi Admin. These are named configuration parameters used by Yabi when creating a Dynamic Backend.

Please edit them under Yabi/DynamicBackendConfigurations in Yabi Admin:

```
Name            The name of the configuration (ex. "CCG AWS t1.micro")
Configuration   A JSON dictionary containing the provider specific configuration.
```

The JSON configuration will always have to contain at least the `instance class` parameter, that specifies the type of the instance. Currently, only `ec2` and `ec2spot` are supported. Use `ec2` for AWS On-Demand instances and `ec2-spot` for AWS Spot instances.

10.4.1 AWS specific configuration

The following configuration parameters are currently supported for AWS instances.

Field	Description
region	The AWS region
ami_id	The id of the AMI image
size_id	The instance type
keypair_name	Name of the SSH key
security_group_names	List of Security groups
spot_price	(Spot Instance Only) The max price your willing to pay for an hour of usage

An example for an AWS Spot instance configuration:

```
{
   "instance_class": "ec2spot",

   "spot_price": "0.08",

   "region": "ap-southeast-2",
   "ami_id": "ami-3f821e05",
   "size_id": "t1.micro",

   "keypair_name": "ccg-syd-staging",

   "security_group_names": [
       "default",
       "ssh"
   ]
}
```

An example for an AWS On-Demand instance configuration:

```
{
   "instance_class": "ec2",

   "region": "ap-southeast-2",
   "ami_id": "ami-3f821e05",
   "size_id": "t1.micro",

   "keypair_name": "ccg-syd-staging",

   "security_group_names": [
```

```
        "default",
        "ssh"
    ]
}
```

The only difference between the two configuration is the `instance_class` (`ec2` vs. `ec2spot`) and there is no `spot_price` for the On-Demand instance.

10.4.2 OpenStack specific configuration

The following configuration parameters are currently supported for OpenStack Nova instances.

Field	Mandatory?	Description
auth_url	Yes	The keystone URL used for authentication
auth_version	No	Apache Libcloud auth_version. Default is "2.0_password"
tenant	Yes	Tenant name
service_type	No	Catalog entry for service type. Default is "compute"
service_name	No	Catalog entry for service name. Default is "nova"
service_region	No	Catalog entry for service region. Default is "RegionOne"
flavor	Yes	Flavor of the instance
image_name	Yes	Name of image to boot the instance from
keypair_name	Yes	Name of the SSH key to install into the instance
availability_zone	No	The zone you would like your instance to be created in
security_group_names	No	List of Security groups

An example would be the following configuration that it is used to start up NeCTAR instances:

```
{
    "instance_class": "nova",
    "auth_url": "https://keystone.rc.nectar.org.au:5000/v2.0/tokens/",
    "tenant": "pt-8173",
    "service_region": "Melbourne",
    "service_name": "Compute Service",

    "availability_zone": "tasmania",
    "flavor": "m1.small",
    "image_name": "NeCTAR Ubuntu 14.04 (Trusty) amd64",
    "keypair_name": "tszabo",
    "security_group_names": [
        "default", "ssh", "icmp"
    ]
}
```

CHAPTER 11

Submission Templates

Yabi uses a submission template every time it submits a job. The submission template allows the user to customise the command according to the needs of the backend or tool.

Here is an example submission template for a PBS execution backend:

```
#!/bin/sh
#PBS -l walltime=${walltime}
#PBS -l mem=${memory}
#PBS -l nodes=1:ppn=${cpus}
%if queue == 'debugq':
#PBS -q debugq
%else:
#PBS -q routequeue
%endif
#PBS -W group_list=my_account_id
% for module in modules:
    module load ${module}
% endfor
cd '${working}'
${command}
```

As you can see, the template is a script that will be passed to qsub and it allows us to use PBS directives, load modules etc.

The variables placed inside *${}* are Submission Template Variables that Yabi makes available for each template. Please refer to *Submission Template Variables* for a description of variables available and their meaning.

The submission templates are using the *Mako templating system <http://www.makotemplates.org/>*.

11.1 Submission Template Lookup Order

Submission templates can be defined on any of the Tools, Backend Credentials or Backends. When a task is submitted Yabi will use the following lookup order to decide which submission template to use:

1. Tool submission template used if defined
2. Backend Credential submission template used if defined
3. Backend submission template used

Usually only Backend submission templates are defined and used for most Tools. Tools that need further customisation will define their own submission template.

11.2 Submission Template Variables

The following is a table of the variables available in the template, and their meaning. Variables are to be placed inside *${}* in the template.

Key	Value
working	The full path of the working directory for the job.
command	The full command line for the final task. The executable and all its passed in arguments.
modules	A list of all the modules that need loading.
cpus	Number of cpus that should be utilised.
memory	The amount of RAM that should be partitioned for the job.
walltime	The walltime for the job.
yabiusername	The username of the yabi user that submitted the job. Can be different from the backend username.
username	The username the job is to be run as
host	The hostname the job is being run on
queue	The name of the queue the job should be submitted to
tasknum	The number (from 1 to tasktotal) of this particular task in the job set.
tasktotal	The total number of tasks in the job set.
envvars	User suppled environment variables. Please see *User-supplied Environment Variables*

11.3 User-supplied Environment Variables

Yabi allows further customisation of the submission template by using user-supplied submission template variables.

Any tool can define an input file parameter called `.envvars.yabi` which is a *special* file. If a job has an input file called `.envvars.yabi` all the variables in it will be made available in the submission template variable `envvars`.

The format of the `.envvars.yabi` file is JSON.

For example, if the following `.envvars.yabi` is passed to a Job:

```
{
   "aMap":
         ["first", "second"],
   "anotherMap": {
         "nestedMapKey": "nestedValue"
   }
}
```

In the submission template you could use:

```
echo "This will evaluate to     'second'      : ${envvars['aMap'][1]}"
echo "And this will evaluate to 'nestedValue' : ${envvars['anotherMap']['nestedMapKey']}"
```

Allowing any valid JSON in the environment file and Mako syntax in the submission template allows a lot of flexibility when running jobs through Yabi.

Furthermore, the `.envvars.yabi` file doesn't have to be static, it can be dynamically generated on the server by another Tool. This way you can have a Tool that runs on the server and by writing out the environment file it can effect how a subsequent Tool is executed.

The following example `count-files.sh` is a simple bash script that counts the number of arguments passed to it and writes it to the Yabi environment file:

```
#!/bin/bash

cat <<EOS > .envvars.yabi
{
  "job_array": "1-$#"
}
EOS
```

Then another Yabi Tool *"cp Job Array on Torque"* can be set up for the standard UNIX tool `cp` with the following Submission Template:

```
#!/bin/sh
#PBS -t ${envvars['job_array']}
cd '${working}'
${command} ../input/testfile_$PBS_ARRAYID.txt testfile_$PBS_ARRAYID.COPY.txt
```

The two Tools above can be combined together to run a Job Array of the size determined dynamically by `count-files.sh` based on the number of files passed to it.

The Job Array will have consist of one Subjob for each file of the form `testfile_INDEX.txt` (where INDEX is 1,2,3,...) and it will make a copy for each file named `testfile_INDEX.COPY.txt`.

CHAPTER 12

Adding Tools

All tools that you wish to run through Yabi must be installed on the machine where they will be executed. Yabi can run any command line tool based on a description of the tool stored in the Yabi Tool table.

When you add a tool to the Yabi Tool table there are a number of fields to fill out. A tool definition is split into two objects:

1. A description of the tool and the parameters it accepts (*ToolDesc*)
2. Information about how to run the tool on a backend (*Tool*)

Both objects are required to use a tool. A single tool description can be used on multiple execution backends.

12.1 Tool Description

Field	Meaning
Name	A unique name used for identification in the database.
Description	The description of the tool that will be seen by the user.
Accepts Input	If checked, this tool will accept inputs from prior tools rather than presenting file select widgets. This should usually be checked.
Tool Output Extensions	This is list of all file extension patterns that this tool will output. You can enter patterns either inline by clicking the green plus beside the extension list or via the Yabi File Extension table. The extensions should be entered using a Unix Glob pattern such as `*.txt`.

12.2 Tool Parameter Fields

Tool parameter fields are used to describe each switch or flag or parameter given to the tool.

12.2.1 Switch

The actual text for the switch that the tool will use i.e. a tool may use `-i` for input, so you would enter `-i` in this field.

12.2.2 Switch Use

This field describes how the switch should be used. There are some basic patterns that are repeated across many tools. They are represented with Yabi using Python string interpolation. Switch use records maybe added inline by clicking the green plus or via the Yabi Parameter Switch Uses table.

To give an example, if a tool takes one switch `-i my_input_file` we would add a Parameter Switch use which we describe as "Both" and represent this as `%(switch)s %(value)s`. Yabi when running this tool would then produce `-i my_input_file`.

The most commonly used parameter switch uses that we use are:

Display	Format String	Description
redirect	>%(value)s	Use this to redirect the output of stdout into a file.
combined with equals	%(switch)s=%(value)s	Both the switch and the value will be passed in the argument list. They will be separated joined with an equals(=) character with no spaces.
nothing		The switch and the value won't be passed in the argument list.
combined	%(switch)s%(value)s	Both the switch and the value will be passed in the argument list. They will be joined together with no space between them.
both	%(switch)s %(value)s	Both the switch and the value will be passed in the argument list. They will be separated by a space.
valueOnly	%(value)s	Only the value will be passed in the argument list (ie. the switch won't be used)
switchOnly	%(switch)s	Only the switch will be passed in the argument list.

12.2.3 Rank

The order in which the switches should appear when running the tool. Leave this blank if the order is unimportant.

12.2.4 Mandatory

Check this box if the user **must** provide an input for this parameter.

12.2.5 Output File

Check this if this parameter relates to an output file i.e. `--output`

12.2.6 Extension Param

If an extension is selected then this extension will be appended to the filename. This should only be set for specifying output files.

12.2.7 Possible Values

This field accepts a JSON snippet that will be presented to the user as a dropdown select widget. Your JSON should look like this:

```
{"value":[
{"display":"option1","value":"value1"},
{"display":"option2","value":"value2"},
{"display":"option3","value":"value3"},
{"display":"option4","value":"value4"},
{"display":"option5","value":"value5"}
]}
```

12.2.8 Default Value

The default value that should be used for this parameter. If you have used Possible Values above this value should match one of the values in the JSON snippet.

12.2.9 Helptext

The help text that is passed to the frontend for display to the user.

12.2.10 File Assignment

Specifies how to deal with files that match the accepted filetypes setting.

- No input files - This parameter does not take any input files as an argument
- Single input file - This parameter can only take a single input file, and batch jobs will need to be created for multiple files if the user passes them in
- Multiple input file - This parameter can take a whole string of onput files, one after the other. All matching filetypes will be passed into it

12.2.11 Use Output Filename

You can set a tool in Yabi to name its output file based on an input file from another parameter. i.e. If your tool runs like this: `mytool -i inputfile.txt` and produces a `.html` output you can set Use Output Filename to `-i` and your output will be named `inputfile.txt.html`. When entering your tool you should enter all other parameters first, save the record, edit it again and set this parameter. That way the dropdown select widget only shows relevant switches.

12.2.12 Accepted Filetypes

The extensions of accepted filetypes for this switch. When searching for input files Yabi will only consider those that match extensions in this list. Again, the extensions should be entered using a Unix Glob pattern such as `*.txt`.

12.3 Tool

A *Tool* object links a tool description and an execution backend for the tool to run on.

Field	Meaning
Tool	The tool description which should be used for this tool.
Path	The path to the binary for this file. This will often just be binary name but can be a full path.
Display Name	The tool name that the user will see.
Enabled	If checked the tool will appear to users, otherwise it will not.
Exec Backend	The execution backend where this tool will be run.
FS Backend	The file system associated with the execution backend.
CPUs	The number of cpus that should be requested in the submission script when running this tool.
Walltime	The walltime that should be requested in the submission script when running this tool.
Module	Yabi supports the use of the Environment Modules project to manage the dynamic modification of a user's environment via modulefiles. If you are using this system you can add a comma-separated list of modules that should be loaded when running this tool.
Queue	The queue that should be requested in the submission script when running this tool.
Max memory	The maximum memory that should be requested in the submission script when running this tool.
Job type	The job type that should be requested in the submission script when running this tool.
Submission	Submission template to be used when submitting a task running this tool. Please see *Submission Templates*.
Lcopy supported, Link supported	If the backend this tool will run on supports local copy or symbolic link, check these boxes to make the tool use local copy orsymlink. See also *Local Copy and Link*.

CHAPTER 13

Importing and Exporting Tools

It is possible to import/export tools in json format.

13.1 Exporting tools in JSON

In the tool listing under YabiTools you'll see the link called `View`. This will give you an overview of the tool including a link to `View JSON`. Clicking on `View JSON` will present a window with a plain text JSON representation of the tool.

13.2 Importing tools in JSON

If you have the JSON for a tool you can use the `Add Tool` page to add in the tool. Go to: https://127.0.0.1/yabi/admin/addtool/ or similar and paste in your JSON.

Once you have done this the tool will be added and you'll be shown the tool edit page. You will need to check all the details, usually you will need to change the backends and check things like the module settings. You will have to add the new tool to the appropriate *Toolsets and Toolgroups* etc.

CHAPTER 14

Toolsets and Toolgroups

Before a user will see a tool in the front end you need to set up assign access to it using Toolsets and Toolgroups.

14.1 Toolsets

Adding toolsets allows grouping of users to determine which tools they have access to. We typically have an `allusers` set, a `dev` set and maybe a `testing` set. On top of that we might have sets for individual research groups etc.

Grouping like this also means that if we are running a licenced tool, we can assign it only to groups of users who are included in the licence.

14.2 Toolgroups

Toolsgroups determine how tools are grouped in the user interface. The groups that you add here will be reflected in the menu on the left of user interface.

NB: If a user has access to a tool more than once through a couple of toolsets, they will only see the tool once in the front end.

CHAPTER 15

Frequently Asked Questions

15.1 Admin

15.1.1 I've installed Yabi but when I try to login why do I see this error "Unable to create a new session key."?

This is most likely because you do not have caching set up and Django cannot write its session information to cache. If you are running memcached caching have you started memcached? If you are using file based caching, is the cache directory writable and readable by Yabi?

15.1.2 What backend should a file select tool use or why won't my file select run?

You have run into an evolutionary quirk of Yabi. See *Null Backend*. If you set up Yabi from our code repository you will find a *file select* tool was installed by default for you to use.

15.1.3 What is the default stageout field for?

See *Default Stageout Explained*.

15.1.4 How does Yabi handle the URI construction with the Backend and Backend Credential records?

See *A note about the Path field*.

15.1.5 I'm trying to add a tool that takes an input directory which it will use as a working directory, how do I do this?

It helps in setting up these tools to know that Yabi always behaves in the same way when executing jobs.

1. It creates a temp directory with a unique name
2. In that directory it creates an input and an output directory
3. It stages all input files into the input directory
4. It cds to the output directory and runs the command
5. It stages out whatever is in the output directory

So in Yabi we don't have an option to pass in dir names as the command always runs in the output dir with inputs from the inputs dir.

In the case of such tools you could write a wrapper (in python etc) to move the input files to the output directory. Then you specify the inputdir to be `.` The other option is to use this in the command field:

```
cp ../input/* .  ; command_to_be_run
```

We would typically then set the tool up so the parameter specifying `.` as the working directory was hidden to the user.

15.1.6 I'm having trouble setting up an SSH backend, what am I doing wrong?

To troubleshoot setting up a backend with SSH here is what we normally do:

1. On the box running Yabi, as the user running Yabi, we will try to ssh to the backend resource. If we want to do that from Yabi with a private key we'll try to use that key i.e. *ssh -i my_priv_key user@hostname*.

2. As long as that works we will then paste the plain text of the key into the correct credential in Yabi. Note that if the private key has a password, then it is this password that goes in the credential record, **not** the user's password for the backend resource.

3. From there we'll log into the front end as the user and see if we can use the files tab. If we're logged in the front end we can also check in the admin under Yabi Users whether we can access the backend (see *Viewing a User's Setup*).

Using just a password for ssh (not a private key) should be the same steps.

15.1.7 SSH works, but I'm having trouble submitting jobs with YABI?

In case you are getting errors when Yabi is trying to submit your jobs, you should look at Syslog table in the Admin interface of Yabi to find out more about the problems you are encountering.

As a general rule the user on the Execution Backend should be able to run the scheduling programs (like qsub, qstat, qdel etc.) via ssh. To ensure you are set up correctly you should run the command below successfully:

Note: use sinfo instead of qstat for SLURM Backends.

$ ssh user_on_backend@backend qstat

15.1.8 How do I get symlinking working?

The backend will use symlinks if these conditions are met:

- the File System backend has Link Supported enabled
- the tools in the workflow have Link Supported enabled
- all the tools use the same File System backend

A gotcha here is that by default the File Select tool uses nullbackend for the File System backend and Execution backend. Make sure that you change the File System backend on File Select to be localfs or scp etc, the same as the tools that will follow it. This should ensure symlinks are used rather than copying input files.

Of course, if you select a file from a file system that is separate from the execution file system then Yabi has to make a copy to stage it in.