Deploying Arch

1. Introduction

This document is written for webmasters. It provides simple step-by-by step instructions for the installation of Arch. Some basic familiarity with managing websites is assumed.

Arch is a multi-component package: it includes Arch indexer (based on Apache Nutch) and Arch search server (based on Apache Solr) with, optionally, one or more PHP front-ends. See Arch White Paper for more details on Arch architecture. Arch also requires a number of freely available software packages: Java, Tomcat, a Web server, PHP and a relational database management system (RDBMS) such as MySQL. These are widely used applications, and some, or all, of these may already be installed.

Note: Arch 1.7 and later versions use own Jetty server and embedded RDBMS (H2) by default. This significantly simplifies deployment. All you have to do to get it going is run Ant build script and insert your seed URLs into Arch script. **For a quick start, see Arch Quick Start Guide**. The document below describes Arch installation for advanced users.

2. System requirements

- A Linux/Unix Operating System, or Windows Vista/7 with Cygwin installed;
- Java 7, or later version;
- Tomcat 6, or later version (optional);
- Apache Ant and Ant-options packages. These are required for building Arch binaries (section 3.7);
- Apache Ivy;
- Optionally, for PHP front-ends, a Web server, such as Apache, with PHP;
- At least 2 GB RAM. The amount of memory required will depend on the size of the web site(s) and the number of web pages to crawl at one iteration. As a very rough guide 2 GB of RAM should be sufficient to crawl approximately 10,000 web pages at one iteration. There can be several iterations in each crawl.
- Free disk space for index data and temporary files. The space for temporary files is used in the indexing process and then released. Again, the amount required will depend on the size of the web sites. As an indication around 50 GB of temporary disk space should be sufficient to crawl about 10,000 pages at one iteration. The size of produced index is small compared to the amount of temporary space required.

3. Downloading and installing Arch and other packages

3.1 Arch package

The Arch package is available from the Arch home page:

http://www.atnf.csiro.au/computing/software/arch/

The Arch package includes source files needed to build and use Arch. Some parts may not be needed for a specific installation. The following folders are included:

ivy – Apache ivy files.

jetty – Jetty servlet engine home directory.

lib – a placeholder for Hadoop native libraries, see lib/native/README.txt

src/bin – executable scripts.

src/conf – Nutch, Arch and Solr configuration.

src/conf/arch – a sample Arch configuration directory. This contains sample Arch configuration files and site folder.

src/java – Java source files.

src/PHP – PHP source files.

src/plugin – source files of Nutch plugins (Java).

src/test and src/testresources – files related to Nutch testing.

src/web – various files required by Solr web application.

3.2 Installing Apache Tomcat (optional)

Arch includes Jetty servlet engine and is configured to use it by default. However, you can use Arch with Tomcat. To install Apache Tomcat see:

http://wwwcsif.cs.ucdavis.edu/~cs160t/tomcat-tutorial.html

3.3 Installing LAMP (optional)

If you are going to deploy PHP front-ends to Arch, you will need a web server with PHP installed. LAMP is an acronym for 'Linux Apache MySQL PHP'. These three Linux applications are often installed as a group.

To install LAMP see:

https://help.ubuntu.com/community/ApacheMySQLPHP

http://wiki.debian.org/LaMp

http://en.opensuse.org/Linux_Apache_MySQL_PHP_Server_%28lamp%29

http://geekzine.org/2008/10/28/install-centos-52-as-a-server-lampbindntp/

Alternatively, to install Apache, MySQL and PHP separately, see sections 3.4 to 3.6.

3.4 Installing MySQL (optional)

Arch includes an embedded RDBMS (H2) and is configured to use it by default. You can also make Arch use MySQL. To install MySQL, see:

http://www.webdevelopersnotes.com/tutorials/sql/installing_mysql_on_linux.php3

http://www.yolinux.com/TUTORIALS/LinuxTutorialMySQL.html

3.5 Installing Apache (optional)

PHP front-ends to Arch need a web server with PHP. To install Apache web server, see:

http://articles.sitepoint.com/article/installing-apache-tutorial

3.6 Installing PHP (optional)

PHP front-ends to Arch need a web server with PHP. To install PHP, see:

http://dan.drydog.com/apache2php.html

http://www.php.net/manual/en/install.unix.apache2.php

3.7 Building Arch

As an example - to install Arch indexer to a directory `/opt/arch-1.9`:

Create a directory `/opt/arch-1.9`, then download the package there and unpack it.

```
#$> cd /opt/arch-1.9
#$> ant
```

These commands will build a subdirectory `ArchHome` inside `/opt/arch-1.9`. Note that if you start the build process again, it will delete `ArchHome` and re-build it completely.

4 Installing the Arch indexer

Tasks described in sections 4.1 – 4.5 can be performed using a GUI based wizard-driven configuration tool (see 4.9). However, it is strongly recommended that you read these sections first.

4.1 Setting up Arch configuration directory

The `ArchHome/conf/arch` directory contains global and local website configuration files and web log files. The log files are optional. If available, they are used to estimate relative document weights.

The build process creates a sample data directory that can serve as a starting point. In this document this directory is referred to as `ArchHome/conf/arch`.

4.2 Creating web sites directories (optional)

Arch is designed to index multiple web sites. They can be defined implicitly by seed URLs provided in the Arch crawling script, or explicitly, if you want to take advantage of Arch features that require configuration.

For advanced configuration, each web site can have its own subfolder under `ArchHome/conf/arch/sites`. For example, if you have three websites, called apple, orange and lemon to be indexed then create folders with the names `ArchHome/conf/arch/sites/apple`, `ArchHome/conf/arch/sites/orange`, and `ArchHome/conf/arch /sites/lemon`.

Folder names must contain only alphanumeric and underscore characters, and have a maximum length of 30 characters.

To create site folders, copy and/or rename the template `ArchHome/conf/arch/sites/template` folder.

4.3 Setting up global configuration

Configuration parameters shared by multiple sites are located in the Arch global configuration file `ArchHome/conf/arch /config.txt`. Open this file in a text editor. There is an explanation for each parameter in the comment lines. The parameters `temp.dir` and `target.db` must be set.

If a PHP front-end is to be used, then the parameter `frontend.profile` should also be set. Note that an unlimited number of front-ends can be used. Each front-end can have a separate profile configuration.

To use the Apache file based authenticator that comes with Arch, set the parameters in the authentication section and make sure that the passwords and groups files exist and contain valid information.

4.4 Configuring individual web sites

Each web site can have its own configuration file in its folder. This contains site-specific parameters and will override the same parameters if these are defined in the global file.

Open the file `data/sites/<your-site-name>/config.txt` in a text editor. As in the global config file, all parameters are explained.

A site can be divided into several 'areas' for more efficient indexing and searching. At least one area must be defined. In the simplest case this is an entire web site. For an explanation of the configuration details see the sample site configuration file.

For example setting:

```
area = mysite

enabled.mysite = on

root.mysite = http://www.mysite.com/

include.mysite = http://www.mysite.com/
```

will include an entire site in one area. Note that `root.mysite` is used as a seed URL for crawling. If this does not reach all of the web pages on the site, then additional root records can be included. For example:

```
root.mysite = http://www.mysite.com/well-connected-page.html

root.mysite = http://www.mysite.com/sitedirectory/index.html
```

Arch can be easily configured to include as many web sites, and as many site areas as desired.

To set the base URL of a web site:

```
url = http://www.mysite.com/
```

It is a good idea to set initial permissions for public and protected areas of your site. For example:

```
permissions = d | http://www.mysite.com/ | public | staff | admin |
admin | admin | s

permissions = d | http://www.mysite.com/protected/ | staff | staff |
admin | admin | admin | s
```

Each site can be configured to use a separate database and a separate authentication plugin. To do this, override relevant parameters.

PHP front-ends can also be set up to use a site authentication instead of a global authentication. However, if this is implemented, then other sites become invisible to this front-end. This is done for transparent search engine sharing, when multiple sites are hosted together, but administered independently.

4.5 Configuring Nutch

Nutch is treated as a component in Arch. It is likely that in the future it will be configured automatically, but currently a couple of files need to be tweaked.

Open `conf/nutch-site.xml` and modify a few parameters that are associated with the identity of the web crawler and how this processes a 'robots permissions' file (`robots.txt`). This is explained further in the file comments.

Open `conf/suffix-urlfilter.txt` and add extensions of files that must **not** be retrieved and indexed. **IMPORTANT:** some binary files may upset parser plugins when Arch attempts to parse them and cause crawls waste time. To prevent this, block processing of these files by placing their extensions to the `conf/suffix-urlfilter.txt` file.

4.6 Modifying the Arch script

Open `bin/arch` in a text editor and set `ARCH_HOME and JAVA_HOME`, where `ARCH_HOME` is the `ArchHome` directory referred above. Also uncomment path to one of Hadoop native libraries, depending on your platform.

There are other parameters in this script that overlap with parameters in Arch configuration files, such as depth of crawling and number of parallel threads used. These parameters, if provided, override parameters in the configuration file.

This script can also be used for defining web sites to index. Just include seed URLs for these sides in the `CRAWLING_SEED` list.

IMPORTANT: When running Arch on Windows with Cygwin, use "Apache-style" Windows paths in arch configuration files , but Cygwin form of paths in `bin/arch`. For example, a path may look like `C:/bin/Arch` (note '/' instead or standard Windows '\'), in a configuration file and `/cygdrive/c/bin/Arch in bin/arch`.

The `bin/arch` script automatically detects Cygwin environment and modifies paths accordingly. When using Arch with Cygwin **you have to use a JVM installed on Windows, not inside Cygwin**.

Cygwin must be started with Administrator privileges, else Arch will complain about failing to create certain directories and exit.

You don't have to run Solr/Tomcat on Cygwin either. See section 5.

4.7 Adding log files

The default location of log files is in a "logs" folder in your site configuration folder (e.g. `ArchHome/conf/arch/sites/apple/logs)`. There are no requirements for log file names. The log files can be packed in gzip format, but time will be spent packing/unpacking them, so, unless there are space limitations, it is better to keep them unpacked.

Log files should be in the combined log format or a compatible format. If a different format is used then the log parser plugin should be replaced.

Starting with version 1.6, Arch supports remote log directories. To use them, you specify "logs" parameter in site configuration file. There may be several remote log directories for one site, each with optional/multiple name filters defined by Java regular expressions. The logs directory URI and filters are separated by "|". For example:

```
logs = sftp://user:pass@server.com:22/var/logs/ | ^2012.+ | ^access.+
```

This will look for logs in directory `/var/logs/` on server `server.com`, using `"user"` and `"pass"` for authentication. In this directory, it will only use files with names starting with `"2012"` or `"access".`

The supported protocols are FILE, FTP, SFTP, FTPS, HTTP and HTTPS. Section 4.7.1 below describes an alternative method of using remote logs.

Use of log files in Arch is optional. It greatly improves quality of search results, but if there are no log files available, Arch will still work.

4.7.1 Using log files of remote sites

It may be that one or more of your sites are located remotely from your Arch host. In this case, placing their log files in directories where Arch expects to find them is difficult. It would involve periodic copying of large files across a couple of firewalls. Similarly, if you are providing search services to your clients, it is also difficult to ensure that Arch has access to fresh logs of their web servers. Often these logs contain confidential information and your clients will not want to export them.

Starting with version 1.42, Arch is able to generate and accept files that contain document weight information in Sitemap (XML) format. Let's call them "log digests" because they do not have to contain full information as specified by the Sitemap protocol.

For each remote site, you can use Arch log processor to process logs on the web server (produce a log digest), compress the resulting file, encrypt it and make available at a certain URL. Compression and encryption are optional. Encryption allows you to use publicly accessible sites to host your log digests. We will describe the full process, but if you are not going to use encryption, do not generate encryption keys and do not include the –k option in the digest generation command.

To set up remote logs processing:

1. Use Arch to generate public and private keys used for log digests encryption. Encryption keys used in Arch are 2048 bits. Asymmetric encryption is slow, but in Arch it is used only to encrypt randomly generated keys used for symmetric encryption. To generate a key pair, uncomment and use a command that is commented out in `ArchHome/bin/arch`. (Don't forget to comment out current active command, or you will end up running them both). Place the generated keys in `ArchHome/conf/arch`.

2. Copy Arch home directory to the host where you are going to be processing logs. You don't have to copy the whole directory. The critical subfolders to copy are `bin`, `lib`, `plugins` and `conf`. In `conf/arch/sites`, only copy folder of the site for each you are setting up log processing.

3. On the remote host, give Arch access to a database. The simplest way to do it is to use the embedded Apache Derby database. To do this, replace relevant records in your configuration files (the root one and/or the site one) on the remote host with these:

   ```
   database = Derby

   target.db = jdbc:derby:archDB;create=true;user=me;password=mine

   db.driver = org.apache.derby.jdbc.EmbeddedDriver
   ```

4. Now you are ready to process logs on the remote host. You can do it by uncommenting and modifying relevant command in **`ArchHome/bin/arch`**. To check what options are available, run the LogProcessor class without any options. If you set encryption enabled, Arch will produce the encrypted digest file (suppose, `logdigest.gz`) and also encrypted symmetric key (in this example, `logdigest.gz.key`). Place both these files in a directory where Arch can get them from. This place can be on a public web server. This is safe to do because the digest is encrypted with a strong symmetric encryption method, and the key is encrypted with Arch 2048 bit public key.

5. On Arch host, tell Arch where to get this site's log digest from. To do this, either edit the site's configuration file manually and set parameter `sitemap.url` to the URL of your log digest file, or use the GUI control tool to set this parameter in the `Log Processing` tab.

This step is optional. You don't have to do it, if you use for your digest the default location, which is `<your-site-base-URL>/arch/sitemap.dat`. Correspondingly, by default, Arch looks for the encryption key at `<your-site-base-URL>/arch/sitemap.dat.key`.

After you've done it, Arch will be getting document weights information from the digest file you've made available. Just keep it fresh.

4.8 Enabling access to RDBMS (optional)

By default, Arch uses an embedded RDBMS (H2), but can also use others, like MySQL:

```
#$> mysql -p -u root
Enter password:
<…>
mysql> create database arch ;
Query OK, 1 row affected (0.00 sec)
mysql> grant all on arch.* to 'archer'@'localhost' identified by 'archpass' ;
Query OK, 0 rows affected (0.00 sec)
mysql>
```

4.9 Using browser based configuration tool

Steps 4.1 – 4.5 (except modifying `conf/suffix-urlfilter.txt`) can be performed using a browser based configuration tool. Some people will find this GUI tool more efficient and simple to use than editing text configuration files. **This tool is new and at a beta stage** in the current Arch version, so, use with care and expect some (hopefully, minor) problems. To start the tool, you can access the tool at this URL:

`http://your-arch-address:port/arch/control`

This tool currently allows configuring Arch in a user friendly, wizard-driven mode. We plan to add more functions in the future.

Note that familiarity with Arch architecture (see Arch White Paper) and this manual helps to do this task right. Please read sections 4.1 - 4.5 even if you plan to use the tool to perform the tasks described in these sections.

5. Installing Arch search server

5.1 Deploying the Arch Tomcat application (optional)

By default, Arch is configured to use it's own copy of Jetty servlet engine, but you can make Arch use Tomcat.

Open `ArchHome/conf/arch.xml` and change the `docBase` attribute of the Context tag and the value `attribute` of the `Environment` tag. Make contents of the file look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<Context docBase="/opt/arch-1.9/ArchHome/jetty/webapps/arch" debug="0"
                                                        crossContext="true">
```

```
  <Environment name="solr/home" type="java.lang.String"
           value="/opt/arch-1.4/ArchHome" override="true"/>
</Context>
```

When finished, close the file and place it to your Tomcat `conf/Catalina/localhost` directory.

Tomcat must have writing access to Arch data, configuration and log directories:

```
#$> chown -R <tomcat user> ArchHome/
```

Restart Tomcat now.

The Arch search form should be available at this address **after first crawl is done**:

`http://your-tomcat-address:port/arch/search`

If anything is wrong, see the Tomcat log files for the reason. The form is not available before crawling is done because data from the database is used to make lists of sites and areas in the form's select boxes.

The setup above assumes that `ArchHome` and Tomcat server are on the same box. In this simple configuration, the ArchHome directory is shared by the crawler and the search server. It is also possible to have them on different boxes or have one search server and multiple crawlers, each on a separate box. To do it, you will have to replicate `ArchHome` and keep the configuration files synchronised.

5.2 Changing authentication in Arch

The authentication plugin that comes with Arch is included as an example. This implements the Apache file based authentication scheme. This scheme is used only by relatively small organizations. For other authentication schemes, the plugin should be replaced. Just look at the example code and follow it to build your own implementing whatever scheme is used in your organisation. If the reference plugin functionality is sufficient for your purposes, you can configure it using relevant parameters in the security section in `ArchHome/conf/arch/config.txt`.

It is strongly recommended to keep security disabled until you get your search working as expected. Else it may get in the way and it will be hard to tell what is causing problems. To switch off security, set `security.enabled` parameter to `false` (this is its default value, to avoid security related problems at the deployment and trial stages).

Changing the authentication for users with PHP front-ends is discussed below.

5.3 Changing the look and feel

To change the look and feel of the JSP interface, modify the Velocity template files in `ArchHome/conf/language/<site language>`, css files in `ArchHome/web/css` and images in `ArchHome/web/images`.

Arch can dynamically switch its visual appearance and comes with three visual themes. These are "bare bones" that are provided for demonstration.

Changing the look and feel of PHP front-ends is discussed below.

5.4 Changing language

To translate the Arch JSP interface to a different language, switch to `ArchHome/conf/arch/language`, copy the directory `en` to a directory with name `<your_language_code>` and translate Velocity templates in it to your language. To make Arch use this file, set the parameter `lang = <your_language_code>` with your requests.

5.6 Doing a trial crawl

You can use parameters in the Arch crawling script `ArchHome/bin/arch`. The text below explains how to do the same using the configuration files.

Open the global configuration file `ArchHome/conf/arch/config.txt`, set `depth = 2` and save the file. If you overrode the depth parameter for individual areas, you will have to change all of them.

Switch to the directory `ArchHome/bin` and type `./arch`.

Watch the output. If nothing alarming happened and it seems that the process finished OK, start a web browser and go to Arch search page: `http://<your-tomcat-or-jetty-server-address-and-port>/arch/search`. Submit a query "http". The returned results page will show the number of found results **which you have permission to see**. If you configured Arch to restrict access to certain pages and your search does not have required access privileges, you will not see these pages in the results. If you see what you expect, then the crawl is successful. However, only a small fraction of the site has been indexed.

The created index is located in `ArchHome/data` and can be browsed with Luke (http://code.google.com/p/luke/) to make sure that it is not empty and makes sense.

5.7 Configuring Arch to crawl password protected sites

Arch can crawl password protected sites where NTLM, Basic or Digest authentication schemes are used. To configure Arch to crawl password protected sites:

1. Change parameter `plugin.includes` in `ArchHome/conf/nutch-default.xml` file or override this parameter in `ArchHome/conf/nutch-site.xml` file. In this parameter, change `protocol-http` to `protocol-httpclient` and leave everything else as is.
2. Add authentication parameters to `ArchHome/conf/httpclient-auth.xml`. Here are detailed instructions on how to do this: http://wiki.apache.org/nutch/HttpAuthenticationSchemes
3. Start crawling "as normal". Now the `protocol-httpclient` plugin will take care of authentication.

5.8 Starting a production crawl

Delete the trial index: type `./clean`. This will make Arch clean the database and Solr index before next crawl.

Open the global configuration file `ArchHome/conf/arch/config.txt`, set the `depth` to a real value and save the file.

Note that the pair of parameters `<depth, max.urls>` puts an upper limit on the number of URLs that will be indexed in each area. For example, if the depth is set to 20 and max.urls to 50,000, then there will be no more than 20 iterations, with a maximum of 50,000 URLs indexed at each iteration. So, the theoretical limit is 1,000,000 URLs. The actual amount will be smaller than this as first iterations will be much smaller than 50,000 URLs.

Once the configuration has been set, switch to bin, type `./arch` and let it run.

5.9 Optimizing crawling performance

1. Make sure that Hadoop can use the native libraries. If it can't, you should see it complaining about this in the logs. Arch comes with them, but for Linux 32 and 64 bit platforms only. They can be compiled for other platforms. See Hadoop documentation for instructions. They make a **huge** difference. In our tests, a job that was running for two days without them, took only 5 hours to finish when they were available.

2. Try to achieve as even as possible distribution of fetched URLs over crawling iterations. Fetching 100000 of URLs in one iteration will take much longer than fetching 10000 URLs in ten iterations. The difference is especially prominent if native libraries are not used.

`Max.urls` can be set to -1, corresponding to an unlimited number. However, this may lead to a very large numbers of URLs fetched during the peak iterations, taking disproportionally long to finish. It is recommend that the `max.urls` would be limited, spreading the load evenly, but doing enough iterations to cover all URLs.

5.10 Switching on faceted search

Arch comes with faceted search implemented for three fields: `ar_area` (area names), `ar_site` (site names) and `type` (page formats). It would seem natural to also enable faceting on file sizes and modification dates. However, web servers often do not send these fields for a variety of reasons, so, faceting on them would look broken and confusing at best.

To enable built-in faceting, set at least one of `facet.areas`, `facet.sites` or `facet.formats` parameters to true. These parameters can be in the root configuration file or in a site configuration file.

Faceting parameters set for sites override root configuration if parameter "domain" is used in the request. Another method to override faceting settings is to add `facet=true` and other Solr faceting parameters to the request. If Arch finds a "`facet`" (can be `facet=false as well`) field in request, it ignores configuration parameters. This allows implementing custom faceting solutions. You can switch on faceting on any fields that you might add using custom index filters or Solr updaters. Add relevant faceting parameters to the query (see Solr documentation), and Arch will form data structures that you can use in Velocity (if you are customizing the Java interface) or Smarty (if you are customizing the PHP interface) templates. You don't have to modify Arch Java classes or JavaScript.

For example of using facets in Velocity, see `ArchHome/conf/arch/language/en/results.vm`. For example of using facets in Smarty, see `ArchHome/PHP/frontend/en/results.tpl`.

5.11 Cleaning documents before indexing

Some parts of web pages should never be indexed. These include common headers and footers, navigational menus and advertising inserts. Arch can remove these parts before indexing, if this feature is switched on and proper configuration parameters are provided. Arch comes with a basic pruner that can be used to remove fragments of text identified by start and end strings. More advanced pruning plugins can be created and used in the pruning chain. See the provided sample configuration file or GUI help for detailed description of pruning configuration parameters.

5.12 Enabling threat scanning and security alerts

Note: this feature is experimental. Use at your own risk.

Arch can be used to quickly locate suspicious pages on your site and report them to the webmaster. It also lets webmasters to be aware of important changes on their web sites, such as new pages, new or changed scripts, new or changed pages with forms, added or removed links. This is very convenient for large multiuser web sites as it gives webmasters a chance to react quickly if someone creates a vulnerability in the site by using unsafe code.

Arch has an excellent ability to find isolated pages and backdoors installed by hackers because it does not depend on web links to find pages,

Arch security scanner searches pages for suspicious strings defined in configuration files. For better security, both output (what the client receives) and sources (such as PHP code) of pages must be scanned. To do this, Arch must be able to download sources. This is not hard to setup using a simple script that serves page source on request.

For details of configuring security scanning in Arch, see the provided sample configuration file or GUI help.

5.13 Running Arch in "watch" mode

When web server logs are available, Arch can check them periodically to find new URLs and index them. If security scanning is enabled, Arch will also scan new pages for threats in the process. The cost of this operation is not much more than reading changes made to the log files since the previous run. Therefore, it can be performed a few times per hour, keeping indexes fresh and detecting new threats very fast.

This mode can't be used when a "normal" crawl is in progress. Therefore, if watch mode is required, it is necessary to deploy two copies of Arch: one for "watch" mode and one for "normal" mode. They must have different databases, temporary and configuration directories, but same Solr index server address.

To run Arch in watch mode, set parameter `watch.mode` to `true` in the root configuration file and start Arch normally.

Note that setting `watch.mode` to `true` automatically overrides `depth` to 1, `depth.loglinks` to 1, `parallel.indexing` to `true` and `remove.duplicates` to `false`.

It is possible to exclude sites from watch mode processing. To do this, set `enabled.loglinks` to `false` in these sites' configuration files.

6.1 Deploying a PHP front-end for Arch

Move the files from `arch-1.9/src/PHP/frontend` to a directory `arch` under your web server document root. Create a "local" subdirectory in this directory and move `config.php` there. The example below is for a local server, but it does not matter where the web server is.

```
#$> mkdir /opt/apache2/docroot/arch
#$> mv /opt/arch-1.9/src/PHP/frontend/* /opt/apache2/docroot/arch/
#$> mkdir /opt/apache2/docroot/arch/local
#$> mv /opt/apache2/docroot/arch/config.php /opt/apache2/docroot/arch/local/
```

Now open the file `arch/local/config.php` and edit the configuration parameters. The meaning of the parameters is explained in the comment lines. Provide the address of the Arch Solr index server application and the authentication parameters: the front-end id and password.

The id and password must match those given in the `frontend.profile` record in the Arch global configuration file discussed above (section 4.3). For site-based authentication (as opposed

to global authentication), set the site name in the “domain” parameter and set the front-end id and password to match those set in the configuration file for the site. As discussed in section 4.4, for site-based authentication, only the data from the site is visible to searches.

6.2 Changing the authentication in PHP front-ends

The default authentication module is located in `arch/auth`. This implements the Apache file based authentication scheme. This scheme is used only by relatively small organizations. If you use a different scheme and need authentication, you will have to replace this module. Just look at the example code and follow it.

6.3 Changing the look and feel

To change the look and feel of the PHP interface, modify the Smarty templates located in the folder arch/`language/en`, images located in the folder `arch/images` and css files in the folder `arch/css`.

Arch can dynamically switch between visual themes and comes with three default visual themes. These are provided as examples for demonstration purposes.

6.4 Changing language

To translate the Arch PHP interface to a different language, create a folder `arch/language/<your_language_code>`, copy files from the folder `arch/language/en/` to the new folder and translate them. To use this file, send parameter `lang = <your_language_code>` with the requests.

7. Tuning Arch

Arch comes with a PHP based tuning and evaluation module. This module allows conducting blind tests to measure Arch precision against other search engines. It also allows viewing results returned by various search engines for a particular query presented side-by-side, with relevant results highlighted. The module is expandable and generic. It comes with search plugins for Google, Arch, Nutch and Funnelback (former Panoptic). More search engines can be added by writing a relatively simple search plugin. (Note that Google have protected their service from automatic use and using the Google plugin is very problematic now).

7.1 Deploying Arch tuning and evaluation module

Move the files from `arch-1.9/PHP/tuner` to a directory `tuner` under the web server document root. Create a “local” subdirectory in this directory and move `config.php` there. The example below is for a local server, but it does not matter where the web server is.

```
#$> mkdir /opt/apache/docroot/tuner
#$> mv /opt/arch-1.1/PHP/tuner/* /opt/apache/docroot/tuner/
#$> mkdir /opt/apache/docroot/tuner/local
#$> mv /opt/apache/docroot/tuner/config.php /opt/apache/docroot/tuner/local/
```

Now open the file `tuner/local/config.php` and edit the configuration parameters. The meaning of the parameters is explained in the comment lines. Open search plugins such as search_arch.php, search_nutch.php, etc., and customize them where required. Put in the URLs

to use to access search interface of your servers. The web server must have permissions to write to the tuner directory tree. So, change permissions accordingly.

7.2 Changing the look and feel and language

This module is based on the code of Arch PHP front-end. Please see sections 6.3 and 6.4.

7.3 Using Arch tuning and evaluation module.

Open `http://<your_server_address>/tuner/index.php`. This page explains how to perform blind evaluation tests. Queries and relevance judgements submitted by users in the process of testing are saved to a file.

To see results returned by the search engines presented side-by-side, with relevant results highlighted, open `http://<your_server_address>/tuner/tune.php`. The script uses the relevance file created in the process of testing (see above). It reads and suggests queries for which it has relevance information provided by testers. Any other queries can be submitted, but, if relevance information is not available, relevant results will not be highlighted.

Getting help

For help with installing Arch, please contact:

CSIRO Astronomy and Space Science

Arkadi Kosmynin

Phone +61 2 9372 4633

Fax +61 2 9372 4444

Email Arkadi.Kosmynin@csiro.au