Grace's Guide Website Maintenance Guide

Table of contents

Executive summary

The Grace's Guide website is the charity's central repository for all publications accessed by its users, and is therefore essential to its operation. The website has for many years been maintained by a developer who is no longer available to work on it.

The Grace's Guide Website Maintenance Guide document describes the architecture and technical components of the Grace's Guide website and its underlying system infrastructure. It includes all the technical knowledge needed for any internal or external staff needing to maintain, update or support the website.

Target audience

The target audience of the document is technical users with a DevOps profile and a good understanding of Amazon Web Services, Apache, SQL databases and PHP.

Introduction

The Grace's Guide website is hosted in an Amazon Elastic Cloud Compute (EC2)[1] instance, which is part of the Amazon Web Services (AWS) offering. The EC2 instance is configured as a LAMP web server. The source files are a MediaWiki[2] distribution stored in a GitHub repository, which has been augmented with some extensions and customisations.

The website includes:

- A private member area with member registration managed with a MediaWiki extension
- Two online payment systems (PayPal for donations and Stripe[3] for payment for publication downloads in the private member area)
- A backend for site editors to update content and upload files
- Google Analytics[4], viewable through a Google account
- A standard MediaWiki search bar with some code modifications

The site also formerly included the following features, which have been discontinued:

- A blog, managed in WordPress
- A contact form, managed through MediaWiki; this has been replaced with a page with a contact email address

The following sections describe:

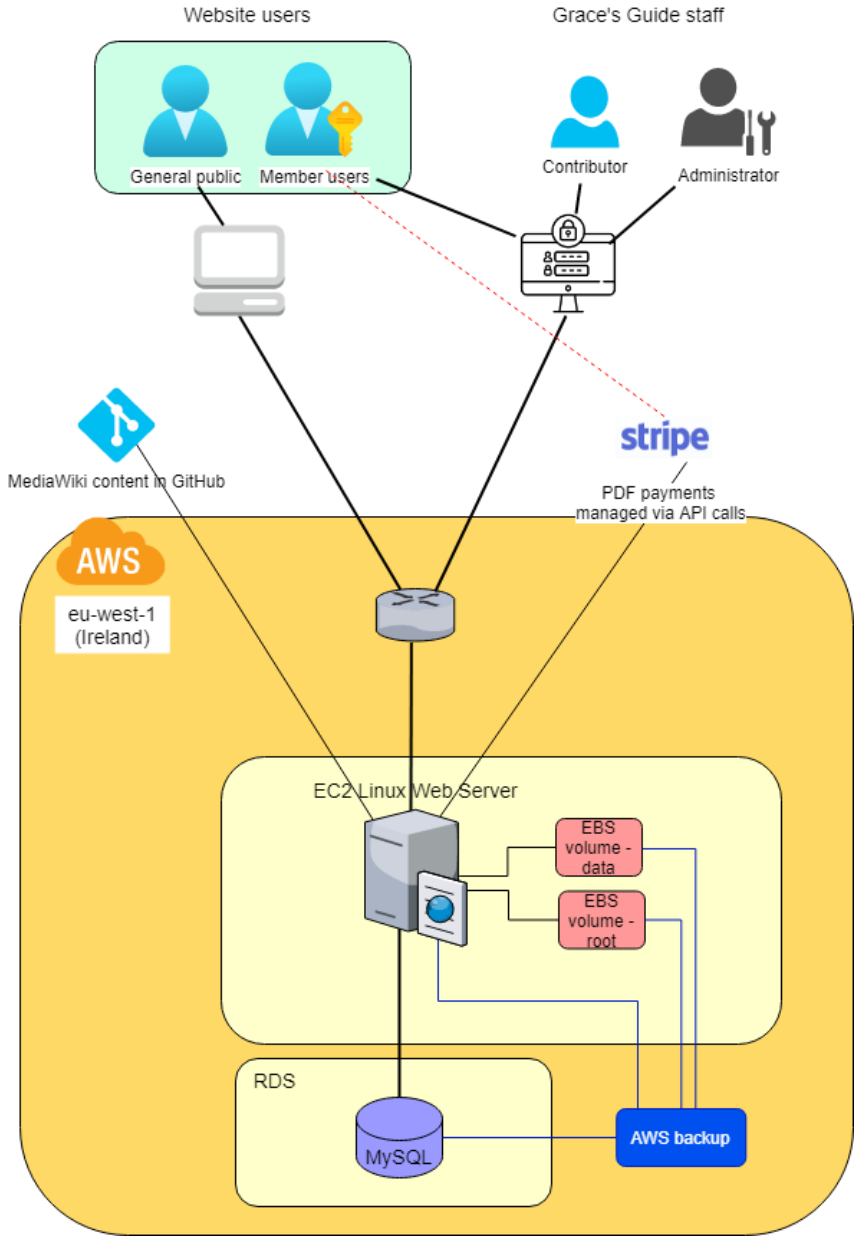
- Technical details of the site infrastructure and components underlying the website: web server, database and software (see Site infrastructure)
- Technical details of the files and components making up the website functionality: GitHub CMS, MediaWiki distribution code (including details of custom extensions and code modifications) and Google Analytics (see Site content)
- Procedures for setting up the web server and maintaining it for optimal performance (see Server maintenance)
- How to update the website components with minimal disruption (see Site updates)
- How to troubleshoot problems arising with the website (see Troubleshooting)
- In addition, the appendices list the following key reference material:
    - Component versions with information about when they need to be upgraded
    - Access information for the various components of the website, including URLs and login details

Technical details

Site infrastructure

The Grace's Guide website is hosted in a single Amazon Elastic Compute Cloud (EC2) instance running an Amazon Linux LAMP stack: Amazon Linux is configured as an Apache web server and connects internally to Amazon RDS[5], where MySQL is running.

The following diagram shows the site infrastructure. Pictured in the diagram are the four user types who access the website (see User administration for more information).



Site content

MediaWiki

The Grace's Guide website was created from a MediaWiki distribution. Editors log in to a MediaWiki backend to update the website.

MediaWiki is based on PHP. The MediaWiki distribution has undergone some customisation, in the form of:

- Extensions — additional folders and files
- Code modifications — changes to the core code

The customisations are described in detail below.

IMPORTANT! Any MediaWiki upgrade needs to take these customisations into consideration.

GitHub

The website source files are stored in a private GitHub repository. Some notable directories are:

Directory	Description
`/skins/GracesGuide.php` `/skins/gracesguide/`	Where the website stylesheets and skin information are stored.
`LocalSettings.php` (root level)	Where extensions are loaded.

Custom extensions

This section describes the customisations made to the MediaWiki implementation, to be used as a guide to changes to make after a site upgrade.

Third-party and custom extensions are as follows:

Extension	Description
`/extensions/anywebsite.php`	Third party extension for placing another page within a wiki page using iframes.
`/extensions/dynamic-what-links-here/show-list.php`	Custom extension for showing a dynamic list of

`/extensions/dynamic-what-links-here/whatlinkshere.php`	pages linking to the current page. The extension converts any occurrences of `<what-links-here/>` in an article text to a dynamically loaded list of articles that link to the article the tag is used in. It replaces the tag with a `div` and populates it using an Ajax call. Ajax calls `show-list.php` with the title of the article given as a request parameter. `show-list.php` performs a database look-up to fetch and render the list of articles that link to the given article.
`/extensions/stats.php`	Custom extension for showing global statistics. Used on the homepage. It replaces `<stats-pages></stats-pages>` and `<stats-images></stats-images>` tags within article text with the corresponding figure taken from the `SiteStats` module that is part of MediaWiki. It renders the stats counters in the page header from the custom skin in `skins/gracesguide/body.php`.
`/extensions/newitems.php`	Custom extension for showing list of recent changes. It replaces `<new-items>no-of-items</new-items>` tags

	within article text with a list of the latest articles added to the site in date order, where `no-of-items` is a number that specifies the length of the list. The list is created by querying the database. Note that certain items are excluded, such as any article with the word "test" in the title, PDFs, JPEGs, etc.
`/extensions/SpecialBulkUpload/SpecialBulkUpload.php` `/extensions/SpecialBulkUpload/SpecialBulkUpload_alias.php` `/extensions/SpecialBulkUpload/SpecialBulkUpload_i18n.php` `/extensions/SpecialBulkUpload/SpecialBulkUpload_body.php` `/extensions/SpecialBulkUpload/wikiupload.pl` `/extensions/SpecialBulkUpload/wikiuploadrc`	Custom extension for allowing bulk upload of multiple images in a zip file. This custom extension also includes code modifications. See SpecialBulkUpload in the section below for more details.
`/extensions/MemberUsers`	Provides functionality to restrict access to PDF files to member users only. Includes Stripe integration for members to pay for PDFs.
`/extensions/SpecialPaginatedCategory`	Special page to display a category using the given letter with pagination.
`/extensions/Cite`	Produces a list of references and inserts them in a ref tag (such as a tag created by `ref-insert` below).
`/extensions/PayPalDonate` `/extensions/DonateButton/donatebutton.php`	PayPal integration. `donatebutton.php` converts a special tag (`<paypal-`

	`donate>` `</paypal-donate>`) in the wiki code and displays a **Donate** button.
`/extensions/ref-insert`	Adds a button to the editor toolbar which inserts an empty `<ref>` tag on the page at the cursor location, to be used by `/Cite` (see above) to produce a list of references.
`/extensions/SocialNetworkIconLinks`	Displays social network icon link code.

Code modifications

SpecialBulkUpload

This extension includes a modification to set the upload link URL. `LocalSettings.php` contains the include to load `SpecialBulkUpload.php` and the configuration to set the URL for the **Upload** link in the site toolbox so that the bulk upload extension page appears instead. It consists of the following files:

- `SpecialBulkUpload.php` — This file is a setup file for the extension. It provides author/version info, etc. and loads the other files that implement the logic.
- `SpecialBulkUpload_alias.php` — This is just a requirement to make the extension operate within the framework.
- `SpecialBulkUpload_i18n.php` — This contains message text - it also could potentially contain translations of the text into another language.
- `SpecialBulkUpload_body.php` — This presents the main bulk upload interface. It allows the user to upload a zip file containing multiple images. Once uploaded, the zip file is extracted by this script and the contents are placed in `/tmp`. Once successfully extracted, the actual import process is passed over to a separate Perl script: `wikiupload.pl`.
- `wikiupload.pl` — This is a third-party Perl script written to perform bulk image additions. It has some slight modifications to produce output that is directly displayed on the bulk upload page so that you can see the status of each file.

Note that the Perl script has some required Perl modules that must be present. They are installed with the following commands (see step 2 of Software installation in the following section):

```
yum install perl-libwww-perl.noarch
yum install perl-XML-TreeBuilder.noarch
```

- `Wikiuploadrc` — This contains configuration information for the main Perl script.

Search modifications

A custom modification to the search filters out unwanted characters in the search results. The modification is not an extension but a core-code alteration. Therefore, you will have to modify the code after any update. The modification is in the file `/includes/specials/SpecialSearch.php` at around line 670. Instead of using the snippet text directly, it is first filtered like this:

```
// format text extract
$snippet = $result->getTextSnippet($terms);


// Strip image entries from the snippet and other unwanted cruft:
$snippet = preg_replace('/\[\[Image\:.*\]\]/Uims', , $snippet);
$snippet = preg_replace('/\[|\]/Uims', , $snippet);
$snippet = preg_replace('/\&lt\;.*\&gt\;/Uims', , $snippet); // HTML strip
$snippet = str_replace('|', , $snippet);
$snippet = str_replace('#', , $snippet);
$snippet = str_replace('*', , $snippet);
$snippet = str_replace('=', , $snippet);
$snippet = str_replace("'", , $snippet);


// Axe very small snippets:
if (strlen(trim($snippet)) <= 2) {
    $snippet = ;
}
$extract = "<div class='searchresult'>$snippet</div>";
```

In addition, in the same file the search pagination limit needs to be set to a high value, for example 1000000, in the `load()` function:

```
public function load() {
                $request = $this->getRequest();
                list( $this->limit, $this->offset ) = $request-
>getLimitOffset( 1000000, );
                $this->mPrefix = $request->getVal( 'prefix', );
```

This prevents the search results from paginating and displaying in two sections (page title matches and page text matches), which is confusing for users, as it is not obvious that there are more page title matches on the next page.

Prevent redirect to exact match page

Another modification to this file prevents the default behaviour of the search that redirects to a page if the search term exactly matches the title. This behaviour was not required and was removed, by commenting out the following code at around line 100:

```
if ( $request->getVal( 'fulltext' )
|| !is_null( $request->getVal( 'offset' ) )
|| !is_null( $request->getVal( 'searchx' ) ) )
{
$this->showResults( $search );
} else {
$this->goResult( $search );
}
```

and replacing it with:

```
$this->showResults( $search );
```

Allow PDF uploads

By default, MediaWiki does not allow PDF uploads. To allow them after upgrading the site, edit `/includes/DefaultSettings.php` and search for `wgFileExtensions =` (it should be at around line 696).

Edit the array so that it contains `'pdf'`, like this:

```
$wgFileExtensions = array( 'pdf', 'png', 'gif', 'jpg', 'jpeg' );
```

Change category paging limit

Edit `/includes/DefaultSettings.php` and change `$wgCategoryPagingLimit` to `10000` as in the example below:

```
$wgCategoryPagingLimit = 10000;
```

Modification to most linked pages report

The file `/includes/specials/SpecialMostlinked.php` has been modified to allow a parameter specified in the URL to filter the results.

Meta description

The file `/includes/OutputPage.php` has been modified to include a meta description.

At around line 3500-3600 a new meta tag is added, containing a sample of the body text from the page.

Stripe integration

As part of Stripe integration, the API was updated to allow for SCA (Strong Customer Authentication) during payment processing. For more information, see https://stripe.com/docs/payments/checkout.

Server setup and administration

This section describes the historical and current setup and administration of the web server.

> **NOTE:** The procedures in Amazon EC2 instance setup were used when migrating the Grace's Guide website to AWS for the first time. They are included here for information and troubleshooting purposes only.
>
> Any software versions referenced in the section were correct at the time, but may now be deprecated. For current software versions, see Software versions.

Amazon EC2 instance setup

For a basic overview of how to set up an Amazon EC2 instance as a LAMP web server, see https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/install-LAMP.html

This section explains how the Grace's Guide EC2 Amazon Linux instance was set up, including:

- Web server: Apache installation and web server configuration
- Database: MySQL database and RDS configuration
- Software: PHP and Perl
- Website creation: import of website files from GitHub

EC2 instance

1. First, create an EC2 instance with the following settings:
   - default VPC
   - subnet: no pref
   - auto-assign public ip: disable (this is assigned from Elastic IP below)
   - shut behav: stop
   - protect against accidental termination = on
   - root volume (ephemeral) = 8gb
   - add ebs volume @ 350gb (not deleted on termination) **NOTE:** This has since been extended by a further 50 GB. See Extending disk space for more details.
   - new security group named `ec2-app-server-sg`
   - inbound rules allow ssh from work and home

- allow port 80+443

2. Create and assign Elastic IP.
3. Associate to EC2 by instance ID (rather than by private IP).
4. Run an update on the EC2 instance:

   ```
   yum update -y
   ```

5. Set up an EBS volume in the EC2 instance (https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-using-volumes.html) by running the following commands as `root`:

   a) Determine the name of the volume:

      ```
      lsblk = xvdb
      ```

   b) Check if it is empty:

      ```
      file -s /dev/xvdb
      ```

      If empty, it will return:

      ```
      /dev/xvdb: data
      ```

   c) Create filesystem:

      ```
      mkfs -t ext4 /dev/xvdb
      ```

   d) Make a mount point:

      ```
      mkdir /data
      ```

   e) Mount it:

      ```
      sudo mount /dev/xvdb /data
      ```

   f) Check it:

      ```
      df -h
      ```

   g) Edit `fstab` so mount is maintained after reboot:

      ```
      cp /etc/fstab /etc/fstab.orig
      nano /etc/fstab
      ```

   h) Add line:

      ```
      /dev/xvdb    /data    ext4    defaults,nofail    0    2
      ```

6. Configure Apache to start at each system boot:

   ```
   chkconfig httpd on
   ```

Database configuration

1. Configure the EC2 instance to connect to RDS as follows:

   a) In AWS Management Console, go to RDS. Select the RDS instance and click the security group (`rds-launch-wizard-1`).

   b) In the inbound rules, add a rule for `MYSQL/Aurora` and enter the security group of the EC2 instance (`ec2-app-server-sg`).

2. Install MySQL client:

   ```
   yum install mysql
   ```

3. Check you can connect:

   ```
   mysql -u master -p -h aitwiki.cjy994eqsbhh.eu-west-1.rds.amazonaws.com
   aitwiki
   ```

   Password: `zFxb3VUdK5O`

4. Create a database user with limited permissions for the application to connect with in MySQL Console:

   ```
   create user 'ggapp'@'%' IDENTIFIED BY 'vjG0cWLknL';
   ```

   ```
   GRANT SELECT, INSERT, UPDATE, DELETE ON aitwiki.* TO 'ggapp'@'%';
   ```

5. Exit the console and try reconnecting with the new details:

   ```
   mysql -u ggapp -p -h aitwiki.cjy994eqsbhh.eu-west-1.rds.amazonaws.com
   aitwiki
   ```

   Password: `vjG0cWLknL`

6. Try to create a table:

   ```
   create table test;
   ```

   Access should be denied.

Software installation (PHP, Perl and Apache)

1. Install PHP and dependencies (including httpd, for Apache):

   ```
   yum install php56 php56-mysqlnd php56-mbstring
   ```

   NOTE: This version of PHP is now deprecated. See table Software versions for the currently installed version.

2. Install the required Perl libraries:

   ```
   yum install perl-libwww-perl.noarch
   ```

   ```
   yum install perl-XML-TreeBuilder.noarch
   ```

Website configuration

Create the Apache configuration as follows.

1. Create a new file `/etc/httpd/conf.d/0-gg-vhost.conf` with the following content:

```
<Directory "/data/site/www">
Options Indexes FollowSymLinks
AllowOverride All
Require all granted
</Directory>

<VirtualHost *:80>

ServerName www.gracesguide.co.uk

ServerAlias gracesguide.co.uk

DocumentRoot /data/site/www

</VirtualHost>
```

2. Start the Apache service:

```
service httpd start
```

3. Create a new test file `/data/site/www/index.php` with the following content:

```
<?php
phpinfo();
```

4. Now visit http://ec2-34-243-38-51.eu-west-1.compute.amazonaws.com/ and check you get the **phpinfo** page.
5. Copy application files to `/data/site/www` (minus `images/pdfs` — to be rsynced separately).
6. Create an `/images` directory under `/data/site/www`.
7. Copy `db_user_settings` and configure for RDS instance details — use the `ggapp` user details created in step 4 of Database configuration.
8. Recreate the `.htaccess` file.
9. Edit `LocalSettings.php` and correct the path for `db_user_settings`.
10. Fix path to db_user_settings in `/data/site/www/extensions/dynamic-what-links-here/show-list.php`.
11. Ensure ownership is `apache:apache` for web files.
12. Set up https:

    a) Run command:

```
yum install mod24_ssl
```

b) Copy `cert_files` from live in `/srv/www/cert_files` to `/data/site/cert_files`.

c) Change `/data/site/cert_files` ownership to `root` with `chmod` and set permissions to `644`.

d) Create `/etc/httpd/conf.d/ssl.conf` from the example on the previous live system. Edit as below:

i. Comment out or remove line `LoadModule ssl_module modules/mod_ssl.so`.

ii. Replace `"SSLMutex default"` with `"Mutux default"`.

iii. `DocumentRoot "/data/site/www"`.

iv. `SSLCertificateFile /data/site/cert_files/c18f73ba7da041f.crt`.

v. `SSLCertificateKeyFile /data/site/cert_files/gracesguide.key`.

vi. `SSLCertificateChainFile /data/site/cert_files/gd_bundle-g2-g1.crt`.

e) Restart Apache:

```
service httpd restart
```

13. Check the site functionality:
    - log in (Wiki user)
    - member functionality
    - bulk upload (Perl script)
    - editing
14. Check the error logs for problems that may not be visible.
15. Initial rsync of `images/` directory from `dl-aitwiki` in `/srv/www/default/images`:

```
rsync -avP /srv/www/default/images/* ec2-user@34.243.38.51:/data/site/www/images
```

16. Create symlink `/srv/www/default` to `/data/site/www` for anything left that still uses old path.

SSL certificate

The web server uses a certificate issued by Let's Encrypt[6].

Apache is configured with the certificate file, private key and key chain in `/etc/httpd/conf.d/ssl.conf` with the following lines:

```
SSLCertificateFile /etc/letsencrypt/live/www.gracesguide.co.uk/cert.pem

SSLCertificateKeyFile
/etc/letsencrypt/live/www.gracesguide.co.uk/privkey.pem

SSLCertificateChainFile
/etc/letsencrypt/live/www.gracesguide.co.uk/fullchain.pem
```

The certificate is automatically renewed and does not require any manual intervention. Auto-renewal is implemented with the following entry in `/etc/crontab`:

```
0 0,12 * * * root python -c 'import random; import time;
time.sleep(random.random() * 3600)' && /usr/local/bin/certbot-auto renew -q
```

.htaccess file configuration

The `/data/site/www/.htaccess` file is a configuration file for Apache. It is used to configure various site-specific options. The content of the file is described below, to be used when troubleshooting and updating the site.

File section	Description
`# Maintenance rule`	This section should normally be commented out, as it is below. When uncommented, it displays a maintenance page instead of the requested content. There is an exception for a given IP address, to allow for testing. `# Maintenance rule:` `#RewriteCond %{REMOTE_ADDR} !^78\.33\.44\.114$` `#RewriteCond %{REQUEST_URI} !\.(css\|gif\|ico\|jpg\|js\|png\|swf\|txt)$` `#RewriteCond %{REQUEST_URI} !/maintenance.php$` `#RewriteRule .* /maintenance.php [L]`
`# Blocked ips`	This section is for blocking access from specific IP addresses (e.g. undesirable bots): `# Blocked ips:` `Deny from 94.23.196.95` `Deny from 37.187.171.167`

	`Deny from 62.210.129.246` `Deny from 89.31.57.5` `Deny from 216.244.66.244` `Deny from 213.174.152.1/24` `Deny from 151.80.19.216`
`# Rewrite to HTTPS`	This section redirects to the https version of the site, i.e. if you visit http://www.gracesguide.co.uk/, you are redirected to https://www.gracesguide.co.uk/. It has exceptions for certain hosts, to prevent redirect from occurring in development and test environments. `# Rewrite to HTTPS:` `RewriteCond %{HTTPS} !=on` `RewriteCond %{HTTP_HOST} !^localhost [NC]` `RewriteCond %{HTTP_HOST} !^dl-gracesguide [NC]` `RewriteRule ^/?(.*) https://%{SERVER_NAME}/$1 [R,L]`
`# Restrict access to certain pages to logged in users only`	This section restricts certain pages to Contributor users only. This is to hide pages used for internal processes, procedures, etc. `# Restrict access to certain pages (tests, internal docs, etc) to logged in users only:` `RewriteCond %{HTTP_COOKIE} !aitwikiUserID` `RewriteCond %{REQUEST_URI} ^/Test:_Image_Processing [or]` `RewriteCond %{REQUEST_URI} ^/Test:_Naming_Conventions_for_Images [or]` `RewriteCond %{REQUEST_URI} ^/Test:_House_Style [or]` `RewriteCond %{REQUEST_URI} ^/Test:_Page_Processing_for_OCR [or]` `RewriteCond %{REQUEST_URI} ^/Test:_OCR_and_Merging_Using_Acrobat [or]` `RewriteCond %{REQUEST_URI} ^/Test:_Camera_Settings [or]`

	RewriteCond %{REQUEST_URI} ^/Test:_Photographing_a_Magazine [or] RewriteCond %{REQUEST_URI} ^/Test:_Uploading_Images_from_the_Camera [or] RewriteCond %{REQUEST_URI} ^/Test:_Rotation_and_Renaming_of_Images [or] RewriteCond %{REQUEST_URI} ^/Test_Poss_Discrepancies [or] RewriteCond %{REQUEST_URI} ^/Test_PDF [or] RewriteCond %{REQUEST_URI} ^/Test_File_Names [or] RewriteCond %{REQUEST_URI} ^/Test:_Linking_Images [or] RewriteCond %{REQUEST_URI} ^/Test_blockquote [or] RewriteCond %{REQUEST_URI} ^/Test_:Table_code RewriteRule ^(.*)$ /No_access [L,R=302]
`# Intercept pdf downloads to check "downloader" user membership`	This section is associated with the Member area functionality. It intercepts PDF file downloads to allow the extension to manage access: `# Intercept pdf downloads to check "downloader" user membership:` `RewriteCond %{REQUEST_URI} ^/images/.*\.pdf$` `RewriteRule ^(.*)$ /Special:MemberUsers?file=$1 [L,R=301]`
`# Main rewrite rule`	This section is the main intercept point to send all requests to the MediaWiki application. It includes exceptions for certain files, such as images, which are served directly, and verification files required by third party services. `# Main rewrite rule:` `RewriteCond %{REQUEST_URI} !^/(stylesheets|images|skins|js|securimage|extensions|gsearchtest)/`

```
RewriteCond %{REQUEST_URI}
!^/(redirect|texvc|index).php

RewriteCond %{REQUEST_URI}
!^/error/(40(1|3|4)|500).html

RewriteCond %{REQUEST_URI} !^/favicon.ico

RewriteCond %{REQUEST_URI} !^/phpinfo.php

RewriteCond %{REQUEST_URI} !^/robots.txt

RewriteCond %{REQUEST_URI}
!^/googlef68b6021b416cb49.html

RewriteCond %{REQUEST_URI} !.*\.php$

RewriteCond %{REQUEST_URI} !^/\.well-known

RewriteCond %{REQUEST_URI}
!^/DCQJTW67CJOI0cTBzIqs

RewriteCond %{REQUEST_URI} !^/godaddy.html

RewriteRule ^(.*)$ /index.php/?title=$1 [L,QSA]
```

Server maintenance

This section explains how to maintain the EC2 Amazon Linux instance and keep all of the website components installed in the instance up-to-date. It also includes server maintenance tasks, such as backup and restore, extending disk space and file maintenance.

Component updates

Apart from MediaWiki software updates, described in the Upgrading MediaWiki section below, the components needing updates are described below.

As described in Upgrade the live environment, updates need to be performed on a test system to ensure they do not break extensions. It may also be necessary to investigate accessing updated software by configuring a separate repository for the `yum` command, depending on the situation.

Software

Apache, PHP and Perl need to be updated manually as needed, according to end-of-life dates or when dependency updates require it. Updates are managed using the `yum` package manager. You can check if any updates are available with the following commands:

```
sudo yum list | grep php
sudo yum list | grep perl
sudo yum list | grep httpd
```

A given version can be installed with a command such as the following:

```
sudo yum install php73
```

Database

RDS has automatic minor updates enabled, so MySQL will remain patched. AWS will notify if any issues occur.

Backup and restore

Backup setup

Under AWS Backup service, a backup plan was created called `main`. Note that this is not a live backup. In other words, you cannot browse a backup version of the website — it requires the backed-up resources to be restored first. They do not have to be restored to the live environment.

The backup rule is called `DailyBackups` and is set to run at 5am daily, retaining for 5 weeks.

The resource assignments section specifies the following items to be included in the backup:

- RDS - `aitwiki` (the main MySQL database)
- EC2 - the web server instance

- EBS - `root` - the root volume containing the web server instance configuration files
- EBS - `data` - the data volume containing the website files including `images/pdfs`

NOTE: As backups are incremental, the charge for this service will gradually increase as the data size increases.

Restore

To restore from backup:

1. Log in to AWS Console and navigate to the **AWS Backup** service.
2. At the top of the page, click option **Restore backup**.
   A list of available backups is displayed:

Resource ID	Resource type	Last backup
instance/i-073abc55ec1d2d209	EC2	Nov 19, 2020 @ 8:04:44 AM UTC+00:00
volume/vol-035ebeaf0b953300f	EBS	Nov 20, 2020 @ 8:04:44 AM UTC+00:00
volume/vol-0c61d35715d4332e8	EBS	Nov 19, 2020 @ 8:04:44 AM UTC+00:00
aitwiki	RDS	Nov 19, 2020 @ 8:04:44 AM UTC+00:00

   - The **EC2** resource type is the web server instance.
   - The first **EBS** resource type in the list (**volume/vol-e35ebeaf0b953300f**) is the 400GB data volume containing all the website files, including PDFs and images.
   - The second **EBS** resource type in the list (**volume/vol-0c61d35715d4332e8**) is the root volume for the web server. It contains the operating system and important configuration files.
   - The **RDS** resource type is the MySQL database.
3. Choose the backup needed to replace the resources which have been lost and click the option to restore to a copy of the original.
4. Reconfigure the remaining resources to make use of the restored copy, using the instructions in the Amazon EC2 instance setup section for guidance.

Extending disk space

These steps relate to the expansion of the main data volume.

Check free space

The amount of free space on each volume can be determined by running the command `df -h`, which, before extending it by 50GB using the procedures below, produced the following result:

```
Filesystem      Size  Used Avail Use% Mounted on
```

```
devtmpfs        7.9G   64K  7.9G   1% /dev
tmpfs           7.9G     0  7.9G   0% /dev/shm
/dev/xvda1      7.8G  1.9G  5.9G  24% /
/dev/xvdb       345G  314G   14G  97% /data
```

The data volume is mounted on `/data`. As can be seen above the used space was at 97%.

Extend the volume

The first step is to take a snapshot of the existing EBS volume in case there is a problem with the expansion.

1. In AWS Management Console, go to **EC2 service > Elastic Block Store > Volumes**.
2. Select the data volume **(id: vol-035ebeaf0b953300f) > Actions > Create snapshot**.
3. Give it an appropriate name and start it (you can view its progress under the **EC2 service > Elastic Block Store > Snapshots** menu).
4. Back in **Elastic Block Store > Volumes** select the data volume and under **Actions** choose **Modify Volume**.
5. Change the size to the new desired size (in this case, 400 GB) and click **Modify** and confirm the changes. It may take a while to apply the changes.
6. Now on the EC2 instance in a command terminal, follow the guidance given in this document for extending EBS volumes: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/recognize-expanded-volume-linux.html

   In summary: the following command can be run as root against the filesystem name (`/dev/xvdb`):

   ```
   resize2fs -f /dev/xvdb
   ```

File maintenance

The unused files page (www.gracesguide.co.uk/Special:UnusedFiles) periodically gets hundreds of images in it that are not required. Because deleting them individually takes a long time the Grace's Guide team will request a bulk deletion when there are too many.

The criteria for which ones to remove vary. Here are some guidelines:

1. In a test environment, access the MySQL database:

   ```
   mysql -u aitwiki --password=syRaygbu aitwiki
   ```

2. The query executed when viewing the unused files pages can be reproduced with a query like this:

```
SELECT img_name, img_user, img_user_text, img_timestamp as value,
img_description FROM image LEFT JOIN imagelinks ON img_name=il_to WHERE
il_to IS NULL order by img_timestamp;
```

3. Delete the entries according to the criteria required by the Grace's Guide team and check the unused files page to ensure it is correct. If it all looks good then repeat the process on live..

Note that this only removes the reference to the file from the database - the files are left on disk. Some space could be cleared by identifying the image files that are no longer referenced in the database and removing them.

User administration

The website has four user types, not all of which require administration. They are as follows:

- Public users: general public users of the website. They are unregistered and therefore require no administration.
- Members: members of the public that can sign up on the site and pay for access to PDF files. This is a self-service process requiring no intervention by an administrator.
- Contributors: Grace's Guide staff who edit the site page content. They are given access to add to and edit the website by the Administrator.
- Administrator: can set up Contributors, delete pages and perform other general administration tasks.

Site updates

This section explains how to update the website content.

Upgrading MediaWiki

Upgrading MediaWiki is a straightforward process, summarised here: http://www.mediawiki.org/wiki/Manual:Upgrading

Care needs to be taken when upgrading the Grace's Guide implementation of MediaWiki, however, as new versions may cause the extensions to stop working.

General guidelines

The following general guidelines apply:

- Upgrading the version of MediaWiki is not something that can be documented step-by-step, as each upgrade will involve a unique set of issues that must be resolved.
- Due to the many custom extensions, custom skin, and other integrations, it is highly likely that an upgrade will break something. For this reason, it is very important to perform the upgrade on a test system first so that all problems can be identified. Ensure your test system is a replica of the live environment in every way that matters before continuing.
- Some of the changes made to MediaWiki cannot be made as extensions. For example, modifications to the search cannot be implemented as an extension. These changes must be made by modifying the core code. A common occurrence is that the relevant area of code will have been reworked or modified significantly from the previous release. This may mean that the required change (e.g. removing image results from the search) must be reimplemented from scratch by analysing the new code.
- For the existing core code changes, check if it becomes possible to implement the changes as an extension. In which case, that would be a sensible route to take instead of modifying core code.

Upgrade the live environment

For all site updates, the new and modified files need to be manually uploaded to the web server.

The upgrade process is as follows:

1. Check the complete upgrade process and any modifications required on a test system.
2. Taking a full backup of the existing system: a snapshot of each EBS volume, the EC2 instance and the RDS instance.

3. If required, display a maintenance page for the duration of the upgrade by uncommenting the section entitled `Maintenance rule` in `/data/site/www/.htaccess`. The maintenance page can be modified by editing `/data/site/www/maintenance.php`. There is a redirect in the page to send any users manually visiting the maintenance page to the main home page.
4. Follow the upgrade guide from the MediaWiki website, as described in the Upgrading MediaWiki section, and apply any fixes for extensions and core modifications. This will involve copying new files from the release into places and running a database update script.
5. The maintenance rule in `/data/site/www/.htaccess` has a clause to allow a given IP address access to the main site. Enter your own IP address here to allow you to inspect the site before disabling the maintenance rule.
6. At this point, you can provide an opportunity for others to test the upgrade and make sure nothing is broken, using the following section for guidance.

Test the upgrade

Pay particular attention to the following extensions when testing:

- Check the bulk upload process works. This uses a Perl script that uses the MediaWiki API to add multiple files to the wiki. This is often a problem area when upgrading.
- Check the member users functionality, including integration with Stripe.
- Check for problems with extensions caused by changes made in the new version of MediaWiki. You can use the details of each custom extension in the Custom extensions section to help identify problems. If anything is not working other than a custom extension, check the MediaWiki Release Notes for the version and any previous releases that were skipped between the version you are upgrading from and the new version. This will often give information about removed features, changes to default settings, etc., that might explain odd behaviour and what to do about it.

NOTE: You must remember to remove the maintenance rule when testing is complete.

Troubleshooting

This section outlines what to do if you encounter issues with the website.

Log files

System log files are found in `/var/log`. If you encounter problems, the main log files to check are the Apache logs in `/var/log/httpd/` – `error_log` and `access_log`.

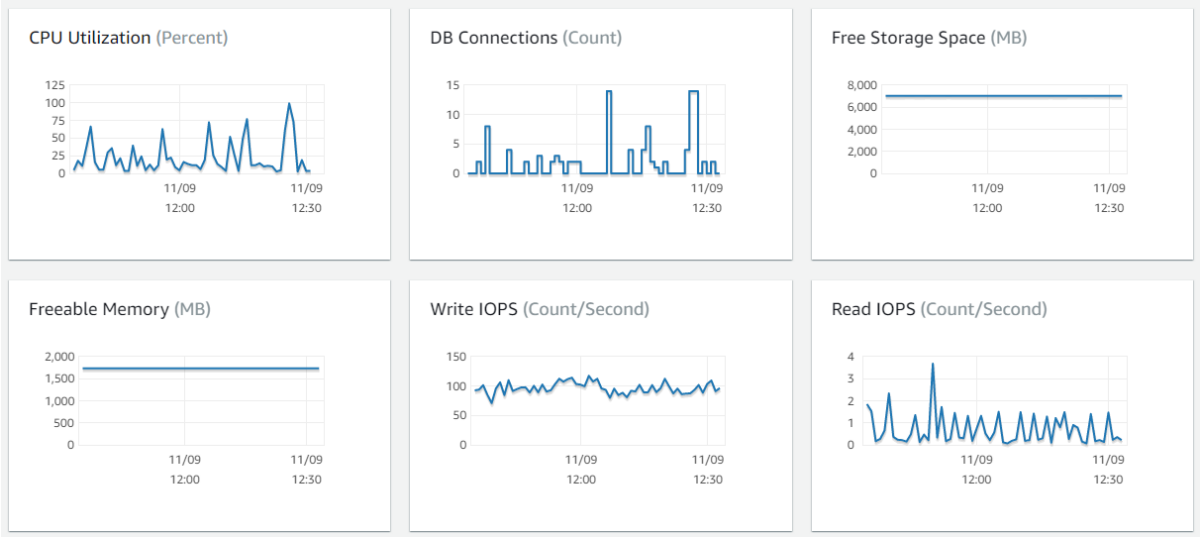
Scenarios

Site running slowly

If the site is running slowly, a likely cause is an aggressive bot crawling pages so quickly that it overloads the database.

To check this:

1. In AWS, go to **Services >RDS > DB Instances > aitwiki > Monitoring** to see information about the current database state. Normal functioning appears like this:



   Problems related to a bot would appear as consistently high CPU and a large number of database connections.

2. Now log on to the EC2 instance from the command line and run the following command:

   ```
   php /data/scripts/find_top_10_ip_requests.php
   ```

   This produces a list of the top ten IP addresses and the number of requests each has made to the site in the last 5000 entries in the log file.

3. If the top address stands out as having many more requests, this could be indicative of the problem above. Search for this IP address in the Apache log file `/var/log/httpd/access_log`.
4. Look at the user agent string for any signs it is a bot, such as a string giving a scripting language such as Python or Perl as the requester, as in the example below:

```
3.235.171.105 - - [09/Nov/2020:07:15:12 +0000] "GET / HTTP/1.1" 301 23
"-" "python-requests/2.18.1"
```

   NOTE: Some bot traffic, such as Googlebot, is legitimate and can safely be ignored.

5. Trace the origin of the IP address using a site such as https://www.ultratools.com/tools/ipWhoisLookup. Tracing the IP address to an unexpected source can indicate it is a bot.

   If you are satisfied the IP address is a bot causing significant load issues for the site and is not a necessary bot such as Googlebot, you can block it, as described in the steps below.

   Note that if you are still experiencing problems but do not find evidence of a persistent bot issue, either because there are not enough requests for it to be detectable, or because it is no longer making requests but the site has not recovered, you can still follow the database and Apache restart instructions in step 8.

6. To block a single address, edit file `/data/site/www/.htaccess` and add a line under the section titled Block ips, with the format:

   `Deny from [IP address]`

   For example:

   `Deny from 151.80.19.216`

7. To block a range of addresses (i.e. if a bot is originating from multiple sources), use CIDR notation, as described at http://www.subnet-calculator.com/cidr.php.
8. Once a problem source is blocked, reboot the RDS instance and restart Apache if the site still needs to recover.

   a) In AWS RDS Console, click **Services > RDS > DB instances** and select **aitwiki**, then in the **Actions** select **Reboot**.
   b) Restart Apache with the following command:

      `service httpd restart`

If after these steps, you are still experiencing problems, try investigating further in the Apache log files.

Registration and login problems

Problems are occasionally encountered with member user accounts, which would benefit from a new screen or adding to the existing ones in the backend administration screens. This is a suggested improvement for the future.

Appendices

Software versions

This table lists the software currently in use with the version, plus any information regarding end-of-life considerations for upgrading.

Software	Version	Comments
Amazon Linux	Amazon Linux AMI release 2017.09	The Amazon Linux AMI will end-of-life its standard support on December 31, 2020 and enter a maintenance support phase. Customers are encouraged to upgrade their applications to use Amazon Linux 2, which includes long term support through 2023. **NOTE:** In spite of the statement by Amazon above, the AMI will still receive standard support until 2023 - more information at the link above.
Apache	2.2.34	
PHP	7.0.33	This version is no longer supported. Users of this release should upgrade as soon as possible, as they may be exposed to unpatched security vulnerabilities. Upgrade to 7.3 or 7.4. Additional modules required for PHP are: • `mysql` (see below) • `pdo`

		• `opcache` • `mbstring` • `xml`
Perl	5.16.3	
MediaWiki	1.32.0	This is a legacy version. Latest version is 1.35.0, which requires PHP 7.3.19+ and MySQL 5.5.8+.
MySQL	5.6.44	

Access details

This table lists all user URL details.

Description	URL
GitHub repository	https://github.com/AshleyDown/ (A new fork will be created or access granted for this private repository)
Amazon Console	https://console.aws.amazon.com/
Google Analytics	https://analytics.google.com/analytics/web/
MediaWiki backend	https://www.gracesguide.co.uk/Special:UserLogin
GitHub login	https://github.com/login
Stripe login	https://dashboard.stripe.com/login

References

1 Amazon Elastic Compute Cloud (EC2): aws.amazon.com/ec2/

2 MediaWiki: www.mediawiki.org/wiki/MediaWiki

3 Stripe: www.stripe.com

4 Google Analytics: analytics.google.com

5 Amazon Relational Database Service (RDS): aws.amazon.com/rds/

6 Let’s Encrypt: https://letsencrypt.org/