WebFaction Software Documentation

Paragon Internet Group Limited - WebFaction is a service of Parag

CONTENTS

1 General Topics **3**
- 1.1 Accessing Logs . . . 3
- 1.2 Monitoring Memory Usage . . . 4
- 1.3 Reducing Memory Usage . . . 6
- 1.4 Setting File Permissions . . . 7
- 1.5 Scheduling Tasks with Cron . . . 9
- 1.6 Troubleshooting . . . 10

2 Installing Software from Source **13**
- 2.1 Troubleshooting . . . 14

3 Bazaar **17**
- 3.1 Installing Bazaar . . . 17
- 3.2 Publishing Bazaar Repositories with Loggerhead . . . 17

4 Custom Applications **21**
- 4.1 Creating a Custom Application . . . 21
- 4.2 Automatically Restarting a Custom Application . . . 21

5 Django **23**
- 5.1 Getting Started with Django . . . 23
- 5.2 Configuring Django . . . 28
- 5.3 Troubleshooting . . . 35

6 Drupal **45**
- 6.1 Backing Up Drupal . . . 45
- 6.2 Updating Drupal . . . 45
- 6.3 Configuring Clean URLs . . . 46
- 6.4 Serving Uploads Faster . . . 47
- 6.5 Speeding Up Drupal with Caching . . . 47
- 6.6 Sending Email from Drupal . . . 48
- 6.7 Troubleshooting . . . 48

7 Git **49**
- 7.1 Installing the Git Web Application . . . 49
- 7.2 Creating a New Repository . . . 50
- 7.3 Cloning a Repository Hosted on Your WebFaction Server . . . 50
- 7.4 Enabling Anonymous Read Access . . . 51
- 7.5 Managing Users . . . 51
- 7.6 Removing a Repository . . . 52
- 7.7 Using HTTPS . . . 52

7.8 Troubleshooting . . . 52

8 Joomla **55**
8.1 Upgrade Joomla . . . 55
8.2 Sending Email . . . 55

9 Memcached **57**
9.1 Setting Up Memcached . . . 57
9.2 Using Memcached with an Application . . . 57
9.3 Shutting Down Memcached . . . 57

10 Mercurial **59**
10.1 Installing Mercurial . . . 59
10.2 Publishing Mercurial Repositories to the Web . . . 59
10.3 Troubleshooting . . . 64

11 mod_wsgi **65**
11.1 Reducing Application Start-Up Time . . . 65
11.2 Reducing mod_wsgi Memory Consumption . . . 65
11.3 Using a virtual environment with mod_wsgi . . . 66

12 MongoDB **69**
12.1 Installing MongoDB . . . 69

13 Movable Type **71**
13.1 Configuring Movable Type to Send Mail . . . 71
13.2 Configuring Movable Type to Send Email with SMTP . . . 72

14 Node.js **75**
14.1 Starting and Stopping Node.js . . . 75
14.2 Installing Packages with `npm` . . . 75
14.3 Upgrading Node.js . . . 76

15 Perl **77**
15.1 Installing cpanminus . . . 77
15.2 Installing a Perl Module with `cpanm` . . . 77
15.3 Using Local Perl Modules in a Command-Line Script . . . 77
15.4 Using Local Perl Modules in a CGI Script . . . 78
15.5 Using Alternate Perl Versions with Perlbrew . . . 78

16 PHP **81**
16.1 PHP on the Command Line . . . 81
16.2 Configuring PHP . . . 81
16.3 Upgrade to another PHP version . . . 82
16.4 Configuring an Upload Limit . . . 82
16.5 Configuring `DOCUMENT_ROOT` . . . 83
16.6 Using FastCGI . . . 84
16.7 Installing a Custom PHP Package . . . 84
16.8 Installing and Using PEAR Packages . . . 85
16.9 Serving HTML Files as PHP Scripts . . . 86
16.10 Sending Mail from a PHP Script . . . 86
16.11 Serving an Application from a Subdirectory . . . 87
16.12 Setting PHP's Default Time Zone . . . 87
16.13 Troubleshooting . . . 88

17 Private Database Instances **89**
17.1 Private MySQL Instances . . . 89
17.2 Private PostgreSQL Instances . . . 92
17.3 Configuring Applications for Private Database Instances . . . 95

18 Pyramid **99**
18.1 Deploying an Existing Pyramid Project . . . 99
18.2 Serving Pyramid with a URL Prefix . . . 100
18.3 Starting, Stopping, and Restarting Pyramid . . . 100

19 Python **103**
19.1 Python Search Path . . . 103
19.2 Creating a `python` Alias . . . 103
19.3 Installing Python Packages . . . 104
19.4 Troubleshooting . . . 106

20 Ruby **111**
20.1 Installing Gems . . . 111

21 Ruby on Rails **113**
21.1 Installing Ruby on Rails . . . 113
21.2 Upgrading RubyGems . . . 114
21.3 Installing Gems . . . 114
21.4 Installing Multiple Gems with Bundler . . . 115
21.5 Installing ImageMagick and RMagick . . . 115
21.6 Installing *sqlite3-ruby* . . . 116
21.7 Deploying a Ruby on Rails Application . . . 117
21.8 Deploying a Ruby on Rails Application with Capistrano . . . 117
21.9 Upgrading Ruby on Rails . . . 118
21.10 Using a Database with a Ruby on Rails Application . . . 119
21.11 Configuring Action Mailer . . . 120
21.12 Troubleshooting . . . 121

22 Redmine **123**
22.1 Configuring Redmine to Send Email . . . 123

23 SQLite **125**
23.1 Installing SQLite . . . 125
23.2 Installing SQLite from Source . . . 125

24 Static Files, CGI Scripts, and PHP Pages **127**
24.1 Static-only . . . 127
24.2 Static/CGI/PHP . . . 128
24.3 Troubleshooting . . . 134

25 Subversion **139**
25.1 Send Email Notifications with a Post-Commit Hook . . . 139
25.2 Managing Users . . . 140
25.3 Reusing Usernames and Passwords Between Subversion and Trac . . . 141
25.4 Backing Up and Restoring Subversion Repositories . . . 141

26 Trac **143**
26.1 Getting Started with Trac . . . 143
26.2 Enabling Email Notifications . . . 145
26.3 Enabling reStructuredText . . . 145

26.4 Managing Users . . . 146
26.5 Disable `.htpasswd` Authentication . . . 146
26.6 Changing the Git Repository Path . . . 147
26.7 Setting a Logo . . . 147
26.8 Setting Trac's Default Time Zone . . . 148
26.9 Upgrade a Trac Application . . . 148

27 WebDAV **151**
27.1 Creating a WebDAV Application . . . 151
27.2 Adding and Removing WebDAV Users . . . 152

28 Webstats **155**
28.1 AWStats . . . 155
28.2 Webalizer . . . 155

29 WordPress **157**
29.1 Using WordPress . . . 157
29.2 Advanced WordPress . . . 161
29.3 Troubleshooting WordPress . . . 164

Index **167**

Contents:

CHAPTER

ONE

GENERAL TOPICS

1.1 Accessing Logs

Most logs are found in `~/logs/frontend` and `~/logs/user`.

Note: To prevent log files from becoming too large, logs are rotated daily at 3:00 a.m. server time. Each existing log is renamed by appending a number which indicates how many days old the log is. For example, `error_website_name.log` is named `error_website_name.log.1` initially. Each subsequent day the log is incremented by one, until the log has aged seven days, upon which it is deleted.

1.1.1 Front-end Logs

`~/logs/frontend` stores logs associated with website entries and shared Apache. There are four types of logs which appear in `~/logs/frontend`:

- **Website access logs** – Logs of the filename form beginning `access_website_name.log` record attempts to access a website. Such logs record:
  - originating IP address,
  - date and time,
  - HTTP method used,
  - the specific URL accessed,
  - and the user-agent which made the request.
- **Website error logs** – Logs of the filename form beginning `error_website_name.log` record error events associated with websites. Such logs record a date and time and an error message.
- **Shared Apache access logs** – Logs of the filename form beginning `access_website_name_php.log` record access attempts for all shared Apache-based applications reached through `website_name`. This includes all shared Apache-based applications, such as Trac, Subversion, and Static/CGI/PHP applications. Such logs record:
  - originating IP address,
  - date and time,
  - HTTP method used,
  - the specific URL accessed,
  - and the user-agent which made the request.

- **Shared Apache error logs** – Logs of the filename form beginning `error_website_name_php.log` record error events for all shared Apache-based applications reached through `website_name`. This includes all shared Apache-based applications, such as Trac, Subversion, and Static/CGI/PHP applications. Such logs record a date and time and an error message.

For example, suppose you have a Website entry with this configuration:

```
mysite
    / --> htdocs (Static/CGI/PHP)
    /blog/ --> wp (WordPress)
    /testing/ --> fancyapp (Django)
```

`mysite`'s access logs are stored in files beginning with `access_mysite.log`, while `mysite`'s error logs are stored in files beginning with `error_mysite.log`. `access_mysite_php.log` records `htdocs` and `wp`'s errors but **not** `fancyapp`'s errors. `fancyapp`'s error logs are stored elsewhere; see *User Logs* for details.

1.1.2 User Logs

`~/logs/user` stores logs associated with many applications, including popular applications such as Django. There are two types of logs which appear in `~/logs/user`:

- **Access logs** – Logs of the filename form `access_app_name.log` record attempts to access the named application. The format of such logs vary with the type of long-running server process associated with the application, but typically these logs record originating IP address, date and time, and other details.
- **Error logs** – Logs of the filename form `error_app_name.log` record error events associated with the named application. Typically, these logs record error messages and their date and time.

Note: Some older installed applications may not store their logs in `~/logs/user`. If you can't find an application's logs in `~/logs`, look in the application's directory (`~/webapps/app_name`). For example:

- Django: `~/webapps/app_name/apache2/logs/`
- Rails: `~/webapps/app_name/logs/`

1.2 Monitoring Memory Usage

To see a list of processes and how much memory they're using:

1. Open an SSH session to your account.
2. Enter `ps -u username -o rss,command`, where *username* is your WebFaction account name and press `Enter`.

The first column is the *resident set size*, the amount of memory in use by the process. The second column is the process's command along with any arguments used to start it.

Repeat these steps as needed for any additional SSH users you have created.

1.2.1 Example Memory Usage

For example, consider a user, *johndoe*, monitoring his memory usage.

```
 [johndoe@web100 ~]$ ps -u johndoe -o rss,comm
  RSS COMMAND
 1640 PassengerNginxHelperServer /home/johndoe/webapps/rails/gems/gems/passenger-2.2.8 /home/johndoe/
 7676 Passenger spawn server
  544 nginx: master process /home/johndoe/webapps/rails/nginx/sbin/nginx -p /home/johndoe/webapps/rai
  844 nginx: worker process
  896 ps -u johndoe -o rss,command
 3564 /home/johndoe/webapps/django/apache2/bin/httpd -f /home/johndoe/webapps/django/apache2/conf/htt
12504 /home/johndoe/webapps/django/apache2/bin/httpd -f /home/johndoe/webapps/django/apache2/conf/htt
 2600 /home/johndoe/webapps/django/apache2/bin/httpd -f /home/johndoe/webapps/django/apache2/conf/htt
23740 /usr/local/apache2-mpm-peruser/bin/httpd -k start
23132 /usr/local/apache2-mpm-peruser/bin/httpd -k start
 1588 sshd: johndoe@pts/1
 1472 -bash
```

The first row displays the column headers:

```
RSS COMMAND
```

RSS stands for *resident set size*. RSS is the physical memory used by the process in kilobytes. COMMAND is the process's command along with any arguments used to start it.

The next four rows show a Rails application running with Passenger and nginx:

```
1640 PassengerNginxHelperServer /home/johndoe/webapps/rails/gems/gems/passenger-2.2.8 /home/johndoe/w
7676 Passenger spawn server
 544 nginx: master process /home/johndoe/webapps/rails/nginx/sbin/nginx -p /home/johndoe/webapps/rail
 844 nginx: worker process
```

The `PassengerNginxHelperServer` and `Passenger` processes are the passenger component of the application, which handle, for example, executing Ruby code. The two `nginx` processes are the web server component of the application, which respond to the incoming HTTP requests. Altogether these processes are consuming 10,704KB or slightly more than 10 megabytes (MB).

The next row is the **ps** process itself:

```
896 ps -u johndoe -o rss,command
```

This is the command that's checking how much memory is in use.

The next three rows represent a running Django application:

```
 3564 /home/johndoe/webapps/django/apache2/bin/httpd -f /home/johndoe/webapps/django/apache2/conf/htt
12504 /home/johndoe/webapps/django/apache2/bin/httpd -f /home/johndoe/webapps/django/apache2/conf/htt
 2600 /home/johndoe/webapps/django/apache2/bin/httpd -f /home/johndoe/webapps/django/apache2/conf/htt
```

Although there are three processes, this is one ordinary Django application. These are the Apache processes used to respond to HTTP requests and to run Django itself. Together these processes are consuming 18,668KB or slightly more than 18MB of memory.

Finally, the last two lines show us *johndoe*'s connection to the server:

```
1588 sshd: johndoe@pts/1
1472 -bash
```

These processes are the SSH service and the Bash prompt, which allow *johndoe* to interact with the server from afar. They use relatively little memory, 3,060KB or under 3MB.

In total, *johndoe* is using less than 32MB of memory, which is well under the limit for his plan, so he's not at risk of having his processes terminated and having to find ways to reduce his memory consumption.

If *johndoe*'s processes had exceeded his plan limits by a small amount, he would receive a warning message. If his processes had exceeded his plan limits by a large amount, his processes would be terminated and he would receive a notification.

1.3 Reducing Memory Usage

See also:

Application specific memory consumption documentation:

- *Django*
- *mod_wsgi*

Once you've identified *where your memory is going* you can take steps to reduce your overall memory consumption. Typically, you can think of the memory your applications require in terms of *base memory* and *additional memory*.

Base memory consumption is the amount of memory a piece of software requires at startup, before handling any requests. Unfortunately, little can be done about base memory consumption. Aside from switching to a different application or modifying the software, some amount of memory must be consumed to start and run the software as expected.

The biggest gains in conserving memory typically come from reducing *additional memory* consumption. Once your software is up and running, varying amounts of additional memory will be consumed to store your data and process requests. Software might consume more or less memory based on factors such as:

- the number and duration of threads or processes,
- the size, number, and frequency of database queries,
- the size or complexity of objects retained in memory, or
- the total number of concurrent requests or sessions.

Because there are so many possible ways for an application to consume memory, there isn't a "one size fits all" solution for reducing memory consumption, but there are a number of common strategies:

- **Serve static files out-of-process.** For application types which rely on an additional server, such as *Django* (Apache) and *Ruby on Rails* (nginx), serve static files, such as style sheets, JavaScript, and images, with a separate *Static-only application*.
- **Plug memory leaks.** If the software you are using contains a memory leak, it will attempt to consume more and more memory without releasing already consumed but unneeded memory. If code under your control is leaking memory, make sure memory is deallocated or references are eliminated when particular objects or data are no longer needed. If some library or executable is leaking memory out of your control, periodically restarting the software may contain the application's memory consumption.
- **Use fewer threads or processes.** If your software relies on multiple threads or processes, try reconfiguring the software to use a more conservative number of them.
- **Complete recommended maintenance activity.** Some software may have maintenance steps which, if left unfinished, may cause increased memory consumption or deteriorated performance.
- **Don't keep unnecessary data in memory.** If certain data is not frequently accessed or is inexpensive to retrieve, try to not keep it in memory. For example, the data associated with an infrequently accessed page may not be needed until the page is actually requested.
- **Avoid making database queries that return too much information.** Not only are such queries slower, but the resulting data will consume additional memory. For example, if the result of some query must be filtered, it may be possible for some memory consumption to be eliminated by writing more specific database queries.

- **Profile your memory consumption.** There may be tools available which work with your programming language to help you identify how memory is being consumed. For example, Guppy for Python features the `heapy` memory inspection module and Xdebug is a popular profiler for PHP.

1.4 Setting File Permissions

See also:

You can also grant access to directories with the control panel. See *Granting Permissions to Additional Users* for details.

Warning: Use caution when granting other users access to your files and directories. In many cases, granting access to other users affords them the privileges of your own account. This can put you at risk of unauthorized, malicious activity, like deleting files or sending spam on your behalf. To minimize risk, grant access only to people you trust and make certain they use the same precautions as you do, like *choosing strong passwords*.

You can use the command-line tools **getfacl** and **setfacl** to manage access control lists (ACLs). Related to **chmod**, **chown**, and **chgrp**, **getfacl** and **setfacl** allow you to grant permissions to files and directories to specific users. The most common case for using ACLs is to grant permissions to another SSH/SFTP user created with the control panel.

On all servers except those >= `web500`, **getfacl** and **setfacl** can record up to 124 permissions per file or directory. On servers >= `web500`, **getfacl** and **setfacl** can record up to 25 permissions per file or directory.

The following subsections show you how to complete common tasks with **getfacl** and **setfacl**. To see complete documentation for **getfacl** or **setfacl**:

1. *Open an SSH session to your account*.
2. Enter `man getfacl` or `man setfacl` and press `Enter`.

1.4.1 Reviewing Permissions

To review the permissions for a file or directory:

1. *Open an SSH session to your account*.
2. Enter `getfacl path`, where *path* is the path to the file or directory see the permissions of, and press `Enter`.

For example, `getfacl /home/demo` returns the ACL for user `demo`'s home directory:

```
# file: .
# owner: demo
# group: demo
user::rwx
user:apache:r-x
user:nginx:r-x
group::--x
mask::r-x
other::---
```

1.4.2 Removing Access to Others (Global Access)

To prohibit read, write, and execute access for all other users (users which are neither yourself or otherwise specified in the ACL):

1. *Open an SSH session to your account*.
2. Enter `setfacl -m o::---,default:o::--- path`, where *path* is the path to a file or directory, and press `Enter`.

Note: If *path* is a directory, then you can also change permissions recursively: enter `setfacl -R -m o::---,default:o::--- path` and press `Enter`.

1.4.3 Granting Access to Specific Users

See also:

You can also grant access with the control panel. See *Granting Permissions to Additional Users* for details.

You can grant users access to individual files or directories, or a whole application.

Applications

To grant another user access to an entire application:

1. *Open an SSH session to your account*.
2. Allow the other user account to locate directories that it has access to within your home directory. Enter `setfacl -m u:secondary_username:--x $HOME`, where *secondary_username* is the other user's username, and press `Enter`.
3. Remove the other user's default access to the applications in your `$HOME/webapps` directory. Enter `setfacl -m u:secondary_username:--- $HOME/webapps/*` and press `Enter`.

Note: The above command only affects applications that are currently installed. If you create new applications, then run the command again, or run `setfacl -m u:secondary_username:--- $HOME/webapps/new_app`, where *new_app* is the name of the new application.

---

4. Grant the user read, write, and execute access to the application's files and directories. Enter `setfacl -R -m u:secondary_username:rwx $HOME/webapps/application`, where *application* is the name of the application to which the other user is to have access, and press `Enter`.
5. Grant the user read, write, and execute access to any files and directories created in the application in the future. Enter `setfacl -R -m d:u:secondary_username:rwx $HOME/webapps/application` and press `Enter`.
6. Set your account's group as the owner of any new files in the application's directory. Enter `chmod g+s $HOME/webapps/application` and press `Enter`.
7. Grant your account full access to files in the application directory, including any files created in the future by the secondary user. Enter `setfacl -R -m d:u:primary_username:rwx $HOME/webapps/application` and press `Enter`.

The other user is granted access to the application.

Tip: For convenience, the secondary user can add a symlink from their home directory to the application directory. To create the symlink:

1. Open an SSH session to the secondary user account.

2. Enter `ln -s /home/primary_username/webapps/application $HOME/application`, where *primary_username* is the name of the account which owns the application, and press `Enter`.

Files and Directories

To grant another user read, write, or execute access (or combinations thereof) to a single file or directory:

1. *Open an SSH session to your account*.
2. Enter `setfacl -m u:username:permissions path`, where
   - *username* is the username of the user to be granted permissions,
   - *permissions* is a combination of `r`, `w`, or, `x` for read, write, and execute, respectively, and
   - *path* is the path from the current directory to the desired file,

   and press `Enter`.

   Additionally, if *path* is a directory and `u` is prepended with `default:` or `d:`, then any new files created by that or any other user within the directory will default to those permissions.

The other user is granted access to the path specified.

1.5 Scheduling Tasks with Cron

You can use Cron to automatically run commands and scripts at specific times.

To review the contents of your crontab:

1. *Open an SSH session to your account*.
2. Enter `crontab -l` and press `Enter`.

To edit your crontab:

1. *Open an SSH session to your account*.
2. Enter `crontab -e` and press `Enter`. Your crontab will open in your default text editor (as specified by the `EDITOR` environment variable).

Note: *vi* is the default editor. To use a different editor to modify your crontab, set the `EDITOR` environment variable. For example, to use the *nano* text editor, enter `EDITOR=nano crontab -e` and press `Enter`.

---

3. Make any desired changes to your cron schedule.

> **Warning:** Cron jobs run under different conditions than scripts run from a shell. When you prepare your cron activity, do not rely on a specific starting directory. Use absolute paths where possible. Likewise, do not rely on environment variables like `PATH`. Changes to the environment by `.bashrc` and other "dot" files are unlikely to be available to your cron task.

Note: You can receive the output of cron jobs by setting the `MAILTO` and `MAILFROM` variables, redirecting output to `mail`, or logging to a file.

To send the output of all cron jobs to a single email address, set the `MAILTO` and `MAILFROM` variables. On a new line before any jobs, insert `MAILFROM=sender` where *sender* is the sender address for cron error

messages. To set the destination address, on a new line before any jobs (for example, immediately after the line containing `MAILFROM`), insert `MAILTO=recipient` where *recipient* is the destination address.

For example, these cron jobs run every 5 minutes and send both output to `recipient@example.com` using `sender@example.com` as their email address:

```
MAILFROM=sender@example.com
MAILTO=recipient@example.com

*/5 * * * * ps -u $USER -o pid,command
*/5 * * * * du -sh $HOME
```

If you want to specify job-specific subject lines and recipient or sender addresses, you can redirect the output of a cron job to `mail`, For example, this cron job runs once an hour and sends all output of `sample_command` to a recipient with a custom subject line, recipient address, and sender address:

```
0 * * * * example_command 2>&1 | mail -s "Example report subject" -S from=sender@example.com rec
```

Instead of email, you can store the output from a cron job by redirecting it to a log file. For example, this cron job records `cron is running` every 20 minutes to `~/logs/user/cron.log`:

```
*/20 * * * * echo "cron is running" >> $HOME/logs/user/cron.log 2>&1
```

See also:

Linux Cron Guide from Linux Help

4. Save and close the file.

1.6 Troubleshooting

Unfortunately, things don't always work as planned. Here you will find troubleshooting hints for general errors and problems sometimes seen among WebFaction users.

1.6.1 Error: *Site not configured*

The error *Site not configured* has several common causes and solutions:

- **Cause**: You recently created a new website record in the control panel.

  Solution: Wait up to five minutes for your website record to be fully configured.

- **Cause**: You created a new website record but did not include the current subdomain, such as `www`.

  Solution: Modify the website record to use the intended subdomain.

- **Cause**: You created a new domain entry in the control panel, but did not create a corresponding site website record.

  Solution: *Create a site record* which references your newly created domain entry.

- **Cause**: You accessed a website with the wrong protocol—in other words, a protocol other than the one configured in the website entry. For example, you accessed a website configured for HTTPS via HTTP or vice-versa.

  Resolution: Reenter the URL with the HTTP or HTTPS as the protocol or modify the website record to use the intended protocol.

See also:

Secure Sites (HTTPS)

- **Cause**: You attempted to access the site by the website's IP address.

  Resolution: Accessing websites by IP addresses is not supported. Please use the URL with the domain name selected in the control panel, typically one you supply or of the form `username.webfactional.com`.

- **Cause**: You account has been disabled because of a billing problem, TOS or AUP violation, or some other problem concerning your account.

  Resolution: If WebFaction disables your account, we will contact you via your account contact email address to let you know the reason and corrective action required. If you suspect that your account has been disabled and you have not been contacted, please open a support ticket.

1.6.2 Error: *Not Found*

The *Not Found* and *There is no application mounted at the root of this domain* error appears when you visit a domain at the root URL path (`/`), but no application is mounted there, but other applications are mounted elsewhere. Some causes for this error include:

- You recently modified website record to include an application at the root URL path, but opened the URL before your changes were activated in the web server and DNS configuration. Please wait a moment and refresh.
- You accessed a website with a protocol other than the one configured in the website record. For example, you added an application to the root of a website configured to use HTTPS, but opened an HTTP URL in your browser. Reenter the URL with the correct protocol, or modify your website records so that the application is available with the indented protocol.
- There is no application mounted at that domain's root. Please revisit the control panel and add the application to the website record's root URL path, verifying that root URL path is correct (for example, verify that there are no unexpected characters after `/`).

1.6.3 Error: *403 Forbidden*

A *403 Forbidden* error typically occurs when file system permissions prevent the web server from accessing the page or running a script. Please see *Setting File Permissions* for more information on changing file permissions.

1.6.4 Error: *502 Bad Gateway*

A *502 Bad Gateway* error occurs when the front-end web server cannot communicate with the application responsible for responding to the request. To resolve this error:

- Confirm that the application is running. If the application crashed, was terminated, or has not yet started following a system reboot, the application must be restarted to respond to requests. If the application was terminated, you will receive a notification email with additional information.

  To restart the application manually, please see your application type's documentation or control panel entry for instructions.

  Most stopped applications are automatically restarted during periodic cron jobs. Please see the application's entry in the control panel for details.

- Confirm that the application is listening to the correct port. Check your application's configuration to make sure that it is listening to the port assigned to the application in the control panel.

1.6.5 Error: *504 Gateway Timeout*

A *504 Gateway Timeout* error occurs when an application takes too long to respond to an incoming request. This error is often caused by an application receiving heavy traffic. This error can also be caused by an application running too slowly; optimizations may be required to avoid the error. For more information, please see the documentation for your application type.

CHAPTER

TWO

INSTALLING SOFTWARE FROM SOURCE

Many software packages can be installed to your home directory by building the software from source. To build software from source and install it to your home directory:

Note: On CentOS 5 servers (servers less than `web300`), some software may be difficult or impossible to install because system libraries such as glibc are too old. In such cases, consider *migrating to a CentOS 7 server*.

---

1. Open an SSH session to your account.
2. Create a temporary directory. Some software installations may require a user-writable temporary directory. To create the directory, enter `mkdir -p $HOME/tmp` and press `Enter`.
3. Set the `TEMPDIR` environment variable to point to the temporary directory. Enter `export TEMPDIR=$HOME/tmp` and press `Enter`.
4. Download the source to your home directory.

   Typically, source is distributed in a zip or tar archive that contains the software package's source files, configuration scripts, and documentation. To download this file directly to your home directory with wget: enter `wget url`, where *url* is the URL for the source archive, and press `Enter`. The source archive is created in the current directory.
5. Unpack the source archive.

   To unpack a zip file, enter `unzip file`, where *file* is the path to the source archive, and press `Enter`.

   To unpack a tar file, enter `tar -xf file`, where *file* is the path to the source archive, and press `Enter`.

   The new directory containing the source files is created in the current directory.
6. Switch to the source directory. Enter `cd source`, where *source* is the path to the directory containing the source files, and press `Enter`.
7. Read the software's documentation. Some software packages may require additional installation steps or configuration switches. Review the package's `README` and `INSTALL` files, if available.
8. Run the package's `configure` script. The script prepares the build environment for the package; typically, switches on this script determine where to install the software and where to find relevant dependencies. The most frequently used switches include:

   `--prefix=` This switch determines where the software will be installed, including shared libraries and executables. You will use `--prefix=$HOME` to choose your home directory as the installation destination.

   `--with-curl=` If applicable, this switch notifies the software where to find the cURL executable. You can use `--with-curl=/usr` to specify the system's cURL executable. Alternatively, you can use

`--with-curl=$HOME` to specify a different version of cURL that you may have installed in your home directory.

`--with-wget=` If applicable, this switch notifies the software where to find the wget executable. You can use `--with-wget=/usr` to specify the system's wget executable. Alternatively, you can use `--with-wget=$HOME` to specify a different version of wget that you may have installed in your home directory.

To see all of the available options, enter `./configure --help` and press `Enter`.

To run the configure script, enter `./configure --prefix=$HOME` *switches*, where *switches* are any additional, space-separated switches the software requires, and press `Enter`.

9. Build the binaries from the source: enter `make` and press `Enter`. The build runs.

   Note: This step may take a few minutes to complete.

10. Install the software. Enter `make install`. The completed software is copied to appropriate locations, such as executables in `~/bin` and libraries in `~/lib`.

2.1 Troubleshooting

2.1.1 Depending on Non-Standard Libraries and Headers Causes Errors

Some software packages may depend on non-standard libraries or headers. If these dependencies are not available, then you may encounter an error such as `cannot open shared object file: No such file or directory` or various other errors while running the software's `configure` script. These problems can be resolved with the help of some environment variables: `CPPFLAGS`, `LDFLAGS`, and `LD_LIBRARY_PATH`.

See also:

Don't forget to set appropriate `configure` switches, too. See the *Installing Software from Source* step-by-step instructions for details.

`CPPFLAGS` is used to support non-standard headers. For installations which require additional headers, you may add your home directory's `include` directory to `CPPFLAGS`. Before running `configure` or `make`, enter `export CPPFLAGS="-I$HOME/include $CPPFLAGS"` and press `Enter`.

`LDFLAGS` is used to support non-standard libraries. For installations and executions which require additional libraries, you may add your home directory's `lib` directory to `LDFLAGS`. Before running `configure` or `make`, enter `export LDFLAGS="-L$HOME/lib $LDFLAGS"` and press `Enter`.

Like `LDFLAGS`, `LD_LIBRARY_PATH` is used to support non-standard libraries. For installations and executions which require additional libraries, you may add your home directory's `lib` directory to `LD_LIBRARY_PATH`. Before running `configure`, `make`, or the software after it is installed, enter `export LD_LIBRARY_PATH=$HOME/lib` and press `Enter`.

To simplify future usage, permanently amend your `LD_LIBRARY_PATH` by adding the export statement to your `$HOME/.bash_profile` file. To permanently add `$HOME/lib` to your `LD_LIBRARY_PATH`, enter `echo "export LD_LIBRARY_PATH=$HOME/lib" >> $HOME/.bash_profile` and press `Enter`.

Finally, for commands which run outside of your shell process, like an Apache server for a Django application or a cron script, you may need to make additions to `LD_LIBRARY_PATH` elsewhere.

For example, a Django application's start script already defines the `LD_LIBRARY_PATH` environment variable. To add `$HOME/lib` to the application's `LD_LIBRARY_PATH`, edit `$HOME/webapps/app/apache2/bin/start` from this:

```
LD_LIBRARY_PATH=/home/<username>/webapps/<app>/apache2/lib /home/<username>/webapps/<app>/apache2/bi
```

to this:

```
LD_LIBRARY_PATH=/home/<username>/lib:/home/<username>/webapps/<app>/apache2/lib /home/<username>/weba
```

where `<username>` is your username and `<app>` is the name of the application.

Likewise, to change a cron job's `LD_LIBRARY_PATH`, prepend `LD_LIBRARY_PATH=$HOME/lib` to your cron job's command. For example, edit this cron task:

```
1 0 * * * command
```

to this:

```
1 0 * * * LD_LIBRARY_PATH=/home/username/lib command
```

where *username* is your username and *command* is the task to be executed.

CHAPTER

THREE

BAZAAR

Bazaar is an open source distributed version control system.

See also:

Mercurial

3.1 Installing Bazaar

To install Bazaar:

1. Open an SSH session to your account.
2. Install Bazaar. Enter `easy_install-X.Y bzr` and press `Enter`.

   Note: You can specify which Python version to use to install Bazaar by entering `easy_install-X.Y bzr` where *X.Y* is a Python version number. For example, to use Python 2.7, enter `easy_install-2.7 bzr`.

3. Configure the default name and email address to appear by default in commits made from your Bazaar installation. Enter `bzr whoami "name <email_address>"`.

You can now use the command line tool **bzr** to work with Bazaar repositories. Enter `bzr help` and press `Enter` for a list of commands. To learn more about using Bazaar, see Bazaar in five minutes and the Bazaar User Guide.

3.2 Publishing Bazaar Repositories with Loggerhead

You can create an application to serve the Loggerhead server on the web.

3.2.1 Create a Website and Application for Loggerhead

First, create a website with an application for Loggerhead:

1. Log in the WebFaction control panel.
2. Click *Domains / websites* → *Websites*. The list of websites appears.
3. Click the *Add new website* button. The *Create a new website* form appears.
4. In the *Name* field, enter a website name.
5. If applicable, in the *Machine* menu, click to select the server to host the website.

6. If applicable, in the *IP address* menu, click to select the IP address to serve the site.
7. If you want your site served over an HTTPS connection, click to select *Encrypted website (https)*.
8. For each domain name you want to use with the website, add it to the list of domains. In the *Domains* field, enter the domain name. Enter one or more domain names. If the domain has not yet been added to the control panel, click the *Create* link that appears at the bottom of the list of domains to add it.

   Note: Don't forget to *point new domains to the WebFaction name servers*.

9. Add an application for Loggerhead.
   (a) Click *Add an application* → *Create a new application*. The *Create a new web app* form appears.
   (b) In the *Name* field, enter a name for the application.
   (c) In the *App category* menu, click to select *Custom*.
   (d) In the *App type* menu, click to select *Custom app (listening on port)*.
   (e) Click the *Save* button. The application is installed and added to website's list of applications.
10. Click the *Add website* button. The website is created and added to the list of websites.
11. Make a note of the port number for the Loggerhead application. This number is required later in the installation process. Click *Domains / websites* → *Applications*. The list of applications appears. The application's port number appears next to the application's type.

3.2.2 Create a Place to Store Bazaar Repositories

Next, a directory needs to be created where the repositories themselves will be stored. Typically, this will be a directory outside of the custom application's directory.

1. *Open an SSH session to your account*.
2. Create the repository directory. Enter `mkdir path`, where *path* is the path to the new directory, and press `Enter`. For example, enter `mkdir $HOME/bzr` and press `Enter`.

3.2.3 Install Loggerhead's Dependencies

Next, install Loggerhead's dependencies. Loggerhead requires SimpleTAL, simplejson, and Paste.

1. Install the latest version of SimpleTAL for Python 2. Enter `easy_install-2.7 url`, where *url* is a URL to the latest version archive (found at Download SimpleTAL for Python 2.), and press `Enter`.
2. Install simplejson. Enter `easy_install-2.7 simplejson` and press `Enter`.
3. Install Paste. Enter `easy_install-2.7 Paste` and press `Enter`.

3.2.4 Download and Setup Loggerhead

The next step is to download Loggerhead and configure it to use the right version of Python.

1. Switch to the Custom application directory. Enter `cd ~/webapps/custom_app_name`.
2. Download the latest version
   of Loggerhead. You can find a link to the latest version from the Loggerhead download page. For example, enter

`wget http://launchpad.net/loggerhead/1.17/1.17/+download/loggerhead-1.17.tar.gz` and press `Enter`. An archive is created in the current directory.

3. Decompress the Loggerhead archive. Enter `tar -zxvf loggerhead-version_number.tar.gz` and press `Enter`. A directory called `loggerhead` is created.
4. Switch to the `loggerhead` directory. Enter `cd loggerhead` and press `Enter`.
5. Open `serve-branches` in a text editor.
6. Edit the first line of the file from this:

```
#!/usr/bin/env python
```

   to this:

```
#!/usr/bin/env python2.7
```

7. Save and close the file

3.2.5 Create a Cron Job to Automatically Start Loggerhead

1. Open an SSH session into your account.
2. Edit your crontab. Enter `crontab -e` and press `Enter`. Your crontab file will open in your text editor.
3. Add this line to your crontab:

```
10,30,50 * * * * /home/<username>/webapps/<custom_app_name>/loggerhead/serve-branches --port=<nu
```

   where

   - `<username>` is your username,
   - `<custom_app_name>` is the name of your Custom app,
   - `<number>` is the port number assigned to your Custom app,
   - and `<bzr_directory>` is the full path to the directory you created to store repositories (for example, `/home/username/bzr`).

4. Save and close the file.

Every twenty minutes your Loggerhead installation is automatically restarted. Once the Loggerhead process starts, access your repositories on the web at the domain and path you specified previously.

CHAPTER

FOUR

CUSTOM APPLICATIONS

If there is software you want to run which does not have a provided one-click installer but can respond to HTTP requests by listening to a specific port, you can use a *Custom app (listening on port)* application.

4.1 Creating a Custom Application

A *Custom app (listening on port)* application assigns a specific port number and creates a subdirectory in `~/webapps` for your use. To create a custom application:

1. Log in to the control panel.
2. Click *Domains / websites* → *Applications*. The list of applications appears.
3. Click the *Add new application* button. The *Create a new application* form appears.
4. In the *Name* field, enter a name for the application.
5. In the *App Category* menu, click to select *Custom*.
6. In the *App Type* menu, click to select *Custom app (listening on port)*.

   Note: If you plan on using WebSockets with the application, click to select *Custom websockets app (listening on port)* instead.

7. If applicable, in the *Machine* menu, click to select a server to host the application.
8. Click the *Save* button. The application is added to the list of applications. The application's *Port* field contains the port number assigned to your application.

To use the application, *create or modify a website entry* that points to your custom application. Requests to that application's URL will be proxied by your server to the port number provided. Your application can then listen and respond to requests on that port.

4.2 Automatically Restarting a Custom Application

On occasion, your custom application may stop running. There are a number of potential causes for this, such as:

- a bug in your application has caused it to crash
- your application consumed too much memory and was terminated
- the server was restarted

Since you probably won't be watching the application constantly, it's advisable to *set up a cron job* which will automatically restart your application if it is no longer running.

While some applications may gracefully restart by running the same command used to launch the program in the first place, others may have unexpected or unpleasant side-effects, such as starting numerous copies of the same program, quickly consuming your memory allotment. For such applications, we recommend wrapping your command in a script which verifies the application is not already running beforehand.

CHAPTER

FIVE

DJANGO

Django is an open source Python web development framework. You can learn more about Django and see the official documentation at the Django website.

5.1 Getting Started with Django

See also:

See our screencast for a brief demo of setting up a Django application.

In this guide, you'll learn to set up a Django site on WebFaction by:

- creating a website entry with the control panel,
- installing applications for Django and static media,
- creating a PostgreSQL database, and
- configuring Django.

When you're finished, you'll be well on your way to making a functioning Django-base web application.

5.1.1 Create a Website with Django and Static Media Applications

The first step is to create a website with applications for Django and serving static media. We're going to use a separate application for static media, instead of using Django, to reduce memory consumption.

1. Log in to the control panel.
2. Click *Domains / websites* → *Websites*. The list of websites appears.
3. Click the *Add new website* button. The *Create a new website* form appears.
4. In the *Name* field, enter a website name.
5. If applicable, in the *Machine* menu, click to select the server to host the website.
6. If applicable, in the *IP address* menu, click to select the IP address to serve the site.
7. For each domain name you want to use with the website, add it to the list of domains.

   In the *Domains* field, enter the domain name. Enter one or more domain names. If the domain has not yet been added to the control panel, click the *Create* link that appears at the bottom of the list of domains to add it.

   Note: Don't forget to *point new domains to the WebFaction name servers*.

8. Add the Django application.

(a) Click *Add an application* → *Create a new application*. The *Create a new web app* form appears.

(b) In the *Name* field, enter a name for the Django application.

(c) In the *App category* menu, click to select *Django*.

(d) Click the *Save* button. The Django application is installed and added to website's list of applications.

9. Create the static media application.

(a) Click *Add an application* → *Create a new application*. The *Create a new web app* form appears.

(b) In the *Name* field, enter a name for the static media application.

(c) In the *App category* menu, click to select *Static*.

(d) In the *App type* menu, click to select *Static only (no .htaccess)*.

(e) In the *URL* field, enter `static`.

(f) Click the *Save* button. The static media application is installed and added to the website's list of applications.

10. Click the *Save* button. The website is created and added to the list of websites.

5.1.2 Creating a Database

Next, create a database and database user.

1. Click *Databases* → *Databases*. The list of databases appears.
2. Click the *Add new database* button. The *Create a new database* form appears.
3. In the *Name* field, enter a name for the database.
4. In the *Database type* menu, click to select *PostgreSQL*.
5. Choose a database owner.

   To choose an existing user, in the *Database owner* menu, click to select a username.

   To create a new user:

   (a) In the *Database owner* menu, click to select *Create a new postgresql user*. *Username*, *Password*, and *Confirm Password* fields appear.

   (b) In the *Username* field, enter a username.

   (c) In the *Password* field, enter a password.

   See also:

   See *Strengthening Passwords* for important information about choosing passwords.

   (d) In the *Confirm Password* field, reenter the password.

6. Click the *Save* button.

The database is created and appears in the list.

5.1.3 Configuring Django

Now, configure the Django application to use the static media application, send email, and connect to the database.

Note: Do you have your own project ready to go or do you want to start a project with a name other than the default?

To substitute `myproject` by creating a new project or uploading your own:

1. *Open an SSH session to your account*.
2. Enter `cd $HOME/webapps/`*`django_app`*, where *django_app* is the name of the Django application as it appears on the control panel, and press `Enter`.
3. Enter `rm -rf ./myproject` and press `Enter`.
4. If applicable, upload your project directory to `$HOME/webapps/`*`django_app`*.

   Otherwise, enter `./bin/django-admin.py startproject `*`project`*.
5. Open `$HOME/webapps/`*`django_app`*`/apache2/conf/httpd.conf` in a text editor.
6. Change the `WSGIDaemonProcess`'s `python-path` option. Replace this line:

```
WSGIDaemonProcess <django_app> processes=2 threads=12 python-path=/home/<username>/webapps/<djan
```

   with this line:

```
WSGIDaemonProcess <django_app> processes=2 threads=12 python-path=/home/<username>/webapps/<djan
```

   where

   - `<django_app>` is the name of Django application,
   - `<username>` is your username,
   - `<python>` is the Python version associated with the application (for example, `python2.7`), and
   - `<project>` is the name of your Django project.
7. Change
   `WSGIScriptAlias / /home/`*`username`*`/webapps/`*`django_app`*`/myproject/myproject/wsgi.py`
   to
   `WSGIScriptAlias / /home/`*`username`*`/webapps/`*`django_app`*`/`*`project`*`/`*`project`*`/wsgi.py`.
8. Save your changes and close the file.

Be sure to update your Django settings to use your new database and media applications.

---

1. *Open an SSH session to your account*.
2. In a text editor, open
   `/home/`*`username`*`/webapps/`*`django_app`*`/`*`project`*`/`*`project`*`/settings.py`, where *django_app* is the name of the application as it appears on the control panel and *project* is the name of Django project (`myproject`, by default).
3. Set `ALLOWED_HOSTS` to a list of hostnames where your Django site is expected to run. For example, if you site is being served at `example.com` and `www.example.com`, then your `ALLOWED_HOSTS` setting should look like this:

```
ALLOWED_HOSTS = [
    'example.com',
    'www.example.com',
]
```

   Alternatively, you can specify a subdomain wild card with a leading period. For example, to permit any subdomain of `example.com`, then `ALLOWED_HOSTS` would look like this:

```
ALLOWED_HOSTS = [
    '.example.com',
]
```

As a security precaution, this setting is *required* for your Django site to run with `DEBUG = False`.

Configuring Django to Connect to a Database

While continuing to edit `/home/username/webapps/django_app/project/project/settings.py`:

1. Update the `DATABASES` `'default'` dictionary:
   (a) Set `ENGINE` to `'django.db.backends.postgresql_psycopg2'`.
   (b) Set `NAME` to a string containing the database name you specified in *Creating a Database*.
   (c) Set `USER` to a string containing the database username you specified in *Creating a Database*.
   (d) Set `PASSWORD` to a string containing the database user's password.
   (e) Set `HOST` to `'127.0.0.1'`.
   (f) Set `PORT` to `'5432'`.

   When you're finished, your `DATABASES` settings will look something like this:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
         'NAME': 'django_db',
         'USER': 'django_db',
         'PASSWORD': 'mysecret',
         'HOST': '127.0.0.1',
         'PORT': '5432',
    }
}
```

Configuring Django to Serve Static Media

While continuing to edit `/home/username/webapps/django_app/project/project/settings.py`:

1. If applicable, specify the paths for additional static media.

   Django automatically includes your `INSTALLED_APPS`'s static files (for example, the Django admin media or the files in `someapp/static`), but if you want to include additional directories of static files, set `STATICFILES_DIRS` to a tuple of directory paths.

   For example, you might add lines like this:

```
STATICFILES_DIRS = (
    '/home/<username>/src/extra_static_media/',
    '/home/<username>/webapps/<django_app>/even_more_static_media/',
)
```

2. Configure Django to collect static media in your static media application's directory.

Set `STATIC_ROOT` to `'/home/username/webapps/static_media_app/'` where *static_media_app* is the name of the static media application.

Configuring Django to Send Email Messages

While continuing to edit `/home/username/webapps/django_app/project/project/settings.py`:

1. Set recipients of error notifications. On a new line, set `ADMINS` to a tuple of tuples containing names and email addresses of notification recipients. For example, insert `ADMINS = (('Jane', 'jane@example.com'), ('John', 'john@example.com'))`.
2. Set the hostname of the SMTP server. On a new line, add `EMAIL_HOST = 'smtp.webfaction.com'`.
3. Set the SMTP user name. On a new line, add `EMAIL_HOST_USER = 'mailbox'`, where *mailbox* is a valid mailbox name.
4. Set the SMTP password. On a new line, add `EMAIL_HOST_PASSWORD = 'password'`, where *password* is the mailbox's password.
5. Set the *From* (origin) email address. On a new line, add `SERVER_EMAIL = 'address'`, where *address* is the sending email address.
6. Save and close `settings.py`.

5.1.4 Synchronizing the Database and Storing Static Files

Now, use `manage.py` to store your static files and synchronize the database.

1. Change to the project directory. Enter `cd /home/username/webapps/django_app/project` and press `Enter`.
2. Enter `pythonX.Y manage.py migrate` and press `Enter`. The database will be updated. You may be prompted to create a superuser; if so, create one now.
3. Enter `pythonX.Y manage.py collectstatic` and press `Enter`. Static files for installed apps and files in `STATICFILES_DIRS` are copied to your static media application.

5.1.5 Restarting Apache

The final stage of the process is to restart Apache.

1. Change to Apache's `bin` directory. Enter `cd /home/username/webapps/django_app/apache2/bin` and press `Enter`.
2. Enter `./restart` and press `Enter`.

You can now open `http://domain/admin` and admire your admin site.

See also:

If you run into any problems with your new Django application, see *Django Troubleshooting*.

5.2 Configuring Django

5.2.1 Serving Django Static Media

For better performance and resource allocation, configure a separate application to serve your Django application's static media, such as images and style sheets. To create and configure an application to serve Django's static media:

1. Log in to the control panel.
2. Add a static media application to the Django website.
   (a) Click *Domains / websites* → *Websites*. The list of websites appears.
   (b) Click on the name of the website that serves your Django application.
   (c) Click *Add an application* → *Create a new application*. The *Create a new web app* form appears.
   (d) In the *Name* field, enter a name for the static media application.
   (e) In the *App category* menu, click to select *Static*.
   (f) In the *App type* menu, click to select *Static only (no .htaccess)*.
   (g) In the *URL* field, enter `/static`.
   (h) Click the *Save* button. The application is installed and added to website's list of applications.
3. Configure Django to store and use static media from a single location.
   (a) Open `/home/username/webapps/django_app/project/project/settings.py` in a text editor, where
      - *username* is your username,
      - *django_app* is the name of your Django application, and
      - *project* is the name of your Django project (for example, `myproject`).
   (b) Verify that the django.contrib.staticfiles app is listed in your project's `INSTALLED_APPS` setting.
   (c) Django will automatically include your `INSTALLED_APPS`'s static files (for example, the files in `someapp/static` and the Django admin media), but to include additional directories of static files, set `STATICFILES_DIRS` to a tuple of directory paths. For example:

```
STATICFILES_DIRS = (
    '/path/to/static/files/',
    '/path/to/other/static/files/',
)
```

   (d) Set `STATIC_URL` to `'http://domain/static/'` where *domain* is the domain name for your Django site.
   (e) Set `STATIC_ROOT` to `'/home/username/webapps/static_media_app/'` where *static_media_app* is the name of the static media application.
   (f) Save and close the file.
4. Collect the static media into the static media application.
   (a) *Open an SSH session to your account*.
   (b) Switch to the Django project directory. Enter `cd /home/username/webapps/django_app/project/` and press `Enter`.

(c) Enter `python*X.Y* manage.py collectstatic`, where *X.Y* is the version of Python for your Django application, and press `Enter`. The static files are copied to your static media application.

5. *Restart the Django application*.

5.2.2 Configuring Django's `ALLOWED_HOSTS`

As a security measure, Django validates hostnames when *debug mode* is disabled. Django will only respond to requests for the hostnames specified in the `ALLOWED_HOSTS` list.

To configure `ALLOWED_HOSTS`:

1. Open your Django project's settings file. For example, if you're using the default project, then open `~/webapps/*django_app*/myproject/myproject/settings.py`, where *django_app* is the name of your Django application as it appears in the control panel, in a text editor.
2. Add the hostnames to serve to the `ALLOWED_HOSTS` list.

   For example, if your Django application is part of a website record that's serving the domains `mysite.example` and `www.mysite.example`, then the `ALLOWED_HOSTS` setting would look like this:

```
ALLOWED_HOSTS = [
    'mysite.example',
    'www.mysite.example',
]
```

   Alternatively, you can specify a subdomain wild card with a leading period. For example, to permit any subdomain of `mysite.example`, then the `ALLOWED_HOSTS` setting would look like this:

```
ALLOWED_HOSTS = [
    '.mysite.example',
]
```

   See also:

   See the official Django documentation for ALLOWED_HOSTS.

3. Save and close the file.
4. *Restart the Django application*.

Django now responds to requests for the specified domains. Requests for other domains raise a `SuspiciousOperation` exception.

5.2.3 Configuring Django to Use Memcached

See also:

Memcached

To configure Django to use Memcached as Django's cache backend, you must set the `CACHES` setting in `settings.py`. To set `CACHES`:

1. *Open an SSH session to your account*.
2. Install python-memcached for your Django application. Enter `PYTHONPATH=$HOME/webapps/*django_app*/lib/python2.7/ easy_install-2.7 --install-dir=$HOM` where *django_app* is the name of the Django application as it appears on the WebFaction control panel, and press `Enter`.

3. Open `$HOME/webapps/django_app/project/project/settings.py` in a text editor, where *project* is the name of the Django project.
4. Add the `CACHES` setting. Append these lines:

```
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': 'unix:<MEMCACHED_SOCKET>',
    }
}
```

   where `<MEMCACHED_SOCKET>` is the path to your memcached socket file (for example, `/home/username/memcached.sock`).
5. Save and close the file.
6. *Restart the Django application*.

5.2.4 Configuring Django to Send Mail

To configure Django to send mail through WebFaction's SMTP server:

1. *Open an SSH session to your account*.
2. Open `$HOME/webapps/django_app/project/project/settings.py` in a text editor, where *django_app* is the name of the Django application as it appears on the WebFaction control panel and *project* is the name of the Django project.
3. Add these lines:

```
EMAIL_HOST = 'smtp.webfaction.com'
EMAIL_HOST_USER = '<mailbox>'
EMAIL_HOST_PASSWORD = '<password>'
DEFAULT_FROM_EMAIL = '<address>'
SERVER_EMAIL = '<address>'
```

   where

   - `<mailbox>` is the name of a WebFaction mailbox as it appears on the control panel,
   - `<password>` is the mailbox password, and
   - `<address>` are email addresses configured on the WebFaction control panel.
4. Save and close the file.
5. *Restart the Django application*.

5.2.5 Configuring Django's Time Zone

WebFaction servers are configured to use UTC as the local time zone, but each Django installation has its own time zone configuration.

For Django 1.4 and newer, Django can be configured to use `datetime` objects that are aware of time zones. To turn on time zone awareness, set `USE_TZ = True` in your Django application's `settings` module.

For earlier versions of Django, set `TIME_ZONE` to a valid time zone name in your Django application's `settings` module.

For more information about time zone configuration, including valid time zone names and alternative configurations, see TIME_ZONE in the Django project documentation.

5.2.6 Mounting a Django Application on a Subpath

A Django application mounted on a path other than the root of a domain must set the `FORCE_SCRIPT_NAME` setting. If `FORCE_SCRIPT_NAME` is not set, Django URLs will not be routed correctly. This may cause bad links, *404 Not Found* errors, and *There is no application mounted at the root of this domain* errors.

To add the `FORCE_SCRIPT_NAME` setting:

1. *Open an SSH session to your account*.
2. Open `$HOME/webapps/django_app/project/project/settings.py` in a text editor, where
   - *django_app* is the name of the Django application as it appears in the WebFaction control panel, and
   - *project* is the name of the Django project (by default, `myproject`).
3. Add `FORCE_SCRIPT_NAME = 'path'`, where *path* is the path on the domain where the application is mounted. For example, if the Django application were mounted on `example.com/blog`, then `settings.py` would contain `FORCE_SCRIPT_NAME = '/blog'`. Do not use a trailing slash.
4. Save and close the file.
5. *Restart the Django application*.

Once the server has restarted, Django will properly handle URLs on the selected domain path.

5.2.7 Password Protecting a Django Application

You can provide password protection for a Django application by configuring the application's Apache server to use basic access authorization.

1. Create an `htpasswd` file to store permitted usernames and passwords.
   (a) *Open an SSH session to your account*.
   (b) Switch to the directory of the Django application. Enter `cd $HOME/webapps/django_app`, where *django_app* is the name of the Django application as it appears in the control panel, and press `Enter`.
   (c) Create the `htpasswd` file. Enter `htpasswd -c htpasswd username`, where *username* is the first username you wish to create, and press `Enter`. A prompt appears to enter a new password.
   (d) At the prompt, enter a new password and press `Enter`. Another prompt appears to confirm the password. Reenter the password and press `Enter`.
2. Configure the Django application's Apache web server to permit access only to users specified in the `htpasswd` file.
   (a) Open `$HOME/django_app/apache2/conf/httpd.conf` in a text editor.
   (b) Near the top of the file, find a series of `LoadModule` directives. Add these four additional directives:

```
LoadModule authn_core_module modules/mod_authn_core.so
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule authn_file_module modules/mod_authn_file.so
LoadModule authz_user_module modules/mod_authz_user.so
```

(c) At the end of the file, add these lines:

```
<Location "/">
    AuthType Basic
    AuthName "My Password Protected Area"
    AuthUserFile /home/<username>/webapps/<django_app>/htpasswd
    Require valid-user
</Location>
```

where `<username>` is the account name and `<django_app>` is the name of the Django application as it appears in the control panel.

You may also change the string following `AuthName` with the text of your choice or change the path in `<Location "/">` to a more specific subpath (for example, `<Location "/private">`).

(d) Save and close the file.

3. *Restart the Django application*.

The Django application (or specified subpath) is now password protected. If needed, you may create and update usernames and passwords. From the application directory, enter `htpasswd htpasswd username`, where *username* is a new or existing username, and press `Enter`. A series of prompts appear to enter and confirm the password.

5.2.8 Restarting a Django Application

You can restart your Django application's Apache server at any time. To restart the Apache instance and Django:

1. *Open an SSH session to your account*.
2. Run the restart script for your Django application. Enter `$HOME/webapps/django_app/apache2/bin/restart`, where *django_app* is the name of the Django application, and press `Enter`. Your Django application restarts in a few seconds.

Note: Alternatively, you may manually start and stop the Apache instance.

- To stop the server, enter `$HOME/webapps/django_app/apache2/bin/stop` and press `Enter`.
- To start the server, enter `$HOME/webapps/django_app/apache2/bin/start` and press `Enter`.

5.2.9 Setting Up a Database

To set up a database for a Django application:

1. *Optional*: If you're using a PostgreSQL or MySQL database, *create a new database*. Make a note of the database type, name, username, and password.
2. Configure the database.
   (a) *Open an SSH session to your account*.
   (b) Open the Django application's `settings` module in a text editor. By default, this file is found at `$HOME/webapps/django_app/project/project/settings.py`, where *django_app* is the name of the Django application as it appears on the control panel and *project* is the name of your Django project.

(c) Replace these lines:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.',
        'NAME': '',
        'USER': '',
        'PASSWORD': '',
        'HOST': '',
        'PORT': '',
    }
}
```

with these:

```
DATABASES = {
   'default': {
       'ENGINE': '<engine>',
       'NAME': '<db_name>',
       'USER': '<db_user>',
       'PASSWORD': '<db_password>',
       'HOST': '',
       'PORT': '',
   }
}
```

where:

- `<engine>` is `django.db.backends.postgresql_psycopg2` for PostgreSQL databases, `django.db.backends.mysql` for MySQL databases, or `django.db.backends.sqlite3` for SQLite version 3 databases,
- `<db_name>` is the path to the SQLite database file or the name of the MySQL or PostgreSQL database,
- `<db_user>` is blank for SQLite databases or a username with permission to access the database, and
- `<db_password>` is blank for SQLite databases or the password for the database user.

Note: If you're using a MySQL database, then your database's storage engine may limit some Django features. By default, MySQL uses the MyISAM storage engine, which does not provide transaction support. If you intend to use Django's transactions with a MySQL database, then choose the InnoDB storage engine before you create your tables for the first time (in other words, before you run the `migrate` management command).

To use the InnoDB storage engine, add the `OPTIONS` key to your database configuration:

```
'OPTIONS': {
    'init_command': 'SET storage_engine=INNODB',
}
```

A complete database configuration with InnoDB activated looks like this:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'demo_db',
        'USER': 'demo_db_user',
        'PASSWORD': 'mydoublesecretpassword',
```

```
                'HOST': '',
                'PORT': '',
                'OPTIONS': {
                    'init_command': 'SET storage_engine=INNODB',
                },
            }
        }
```

For more information, see the Django documentation for storage engines.

(d) Save and close the file.

3. Synchronize the database.
   (a) Switch to the Django project directory. Enter `cd $HOME/webapps/django_app/project/`, where *project* is the name of the project directory, and press `Enter`.
   (b) Enter `pythonX.Y manage.py migrate`, where *X.Y* is your Django application's version of Python and press `Enter`.
4. *Restart the Django application*.

5.2.10 Upgrading your Django Libraries

To upgrade the Django libraries used by your Django application:

1. *Open an SSH session to your account*.
2. Go to your Django application directory. Enter `cd $HOME/webapps/app_name`, where *app_name* is the name of the Django app that you are upgrading, and press `Enter`.
3. Download the Django source package. Enter `wget https://www.djangoproject.com/m/releases/X.Y/Django-X.Y.Z.tar.gz`, where *X.Y* and *X.Y.Z* are version numbers (for example, `1.6` and `1.6.5`), and press `Enter`. An archive file is created in the current directory.

   See also:

   See the Django download page for additional downloads and links to release notes.
4. Decompress the archive. Enter `tar -zxvf Django-X.Y.Z.tar.gz` and press `Enter`. A directory containing the contents of the archive is created.
5. Rename your old Django libraries to move them out of the way. Enter `mv lib/pythonA.B/Django-X.Y.Z-pyA.B.egg lib/pythonA.B/django_old.egg`, where *A.B* is the Python version used with the Django application, and press `Enter`.
6. Copy the new libraries into your Python library directory. Enter `cp -R Django-X.Y.Z/django lib/pythonA.B` and press `Enter`.
7. Copy the new management scripts into `$HOME/webapps/app_name/bin`. Enter `cp Django-X.Y.Z/django/bin/* bin` and press `Enter`.
8. Make any changes to your project code as directed by the release notes for the Django version you're upgrading to.
9. *Restart the Django application*.
10. Delete the Django source archive and directory. Enter `rm -rf Django-X.Y.Z*` and press `Enter`.

11. Delete your old Django libraries. Enter `rm -rf lib/pythonA.B/django.old` and press `Enter`.

Note: This will not change the Django version number shown in the control panel. The control panel only records originally installed version numbers.

5.2.11 Using the Latest Django Trunk

The WebFaction control panel includes an installer for the latest trunk version of Django. The installers use a nightly export of the trunk from the official Django repository, not a version-controlled checkout. To replace installed Django libraries with a Git clone of the Django repository:

1. *Open an SSH session to your account*.
2. Remove the existing Django libraries. Enter `mv $HOME/webapps/app/lib/pythonX.Y/django $HOME/webapps/app/lib/pythonX.Y/django_orig` where *app* is the name of the application and *X.Y* is the Python version used with the application, and press `Enter`.
3. Clone the Django repository. Enter `git clone https://github.com/django/django.git $HOME/django` and press `Enter`. A clone of the Django repository is created in your home directory.

   Note: You can put the Django repository anywhere you like. To use another directory, substitute `$HOME/django` with your preferred destination.

4. Create a symbolic link to the Git repository's Django library. Enter `ln -s $HOME/django/django $HOME/webapps/app/lib/pythonX.Y/django` and press `Enter`.
5. *Restart the Django application*.

Now the Django application uses the version-controlled Django checkout.

5.3 Troubleshooting

5.3.1 General Django Troubleshooting Techniques

While some specific Django problems are addressed in this guide, some problems may require additional investigation to resolve. Here are some common strategies for resolving problems with Django applications.

Reviewing Django Logs in Real Time

You can review the logs of a Django application as they are recorded using **tail** with the log file. To review the logs in real time:

1. *Open an SSH session to your account*.
2. Switch to the user logs directory. Enter `cd $HOME/logs/user` and press `Enter`.
3. Enter `tail -f error_django_app.log`, where *django_app* is the name of the Django application as it appears in the control panel and press `Enter`.

New additions to the error log will automatically appear on screen. You can now access the site and see any log entries as they appear.

See also:

Accessing Logs

Using the DEBUG Setting

Django has a built-in debug mode, provided by the `DEBUG` setting in your Django project's `settings` module. When `DEBUG` is enabled and an exception is raised, a detailed debugging output (including a stack trace) is rendered instead of attempting to render the 500 template.

See also:

Django Documentation > Settings > DEBUG

> **Warning:** Do not leave `DEBUG` set to `True` during normal operation. In addition to showing users unnecessary and potentially compromising configuration details in error pages, the `DEBUG` setting also causes Django to consume significantly more memory, which may exceed plan allotments.

To enable the Django `DEBUG` setting:

1. Open an SSH session.
2. Open `settings.py` for the Django application. Typically, this file is found in `$HOME/webapps/django_app/project/project`.
3. If it already exists, edit the line containing `DEBUG = False` to `DEBUG = True`. Otherwise, add a new line containing `DEBUG = True` to the file.
4. *Restart the Django application*.

Now, when an error occurs, a stack trace, settings information, and other valuable data is provided instead of rendering an error page. To force the appearance of the Django debugging page, add an `assert False` to a view and attempt to access it.

Using the Django Debug Toolbar

The Django Debug Toolbar is a pluggable Django app that adds debugging information (such as settings, headers, and SQL queries.) to pages.

To install and configure the Django Debug Toolbar with the default `myproject` Django project:

1. Install the Django Debug Toolbar:
   (a) *Open an SSH session to your account*.
   (b) Switch to the Django application's directory. Enter `cd $HOME/webapps/django_app`, where *django_app* is the name of the Django application as it appears on the control panel, and press `Enter`.
   (c) Add the Django application's Python directory to `PYTHONPATH`. Enter `export PYTHONPATH=$PYTHONPATH:$PWD/lib/pythonX.Y/`, where *X.Y* is the version of Python for the Django application (for example, Python 2.7 or Python 3.3), and press `Enter`.

(d) Install the Django Debug Toolbar. Enter `easy_install-X.Y --install-dir=$PWD/lib/pythonX.Y/ django-debug-toolbar` and press `Enter`.

2. Add the `WebFactionFixes` middleware class to the Django project.

   For access control, the Django Debug Toolbar uses the `REMOTE_ADDR` header to determine a request's IP address. Because WebFaction servers proxy requests to your Django application's Apache server, the `REMOTE_ADDR` header is set to your server's IP address, rather than the actual origin of the request. Adding a middleware class fixes this problem.

   (a) Open a new file, `~/webapps/django_app/myproject/myproject/middleware.py` in a text editor.

   (b) Append the following class:

```
class WebFactionFixes(object):
    """Sets 'REMOTE_ADDR' based on 'HTTP_X_FORWARDED_FOR', if the latter is
    set.

    Based on http://djangosnippets.org/snippets/1706/
    """
    def process_request(self, request):
        if 'HTTP_X_FORWARDED_FOR' in request.META:
            ip = request.META['HTTP_X_FORWARDED_FOR'].split(",")[0].strip()
            request.META['REMOTE_ADDR'] = ip
```

   (c) Save and close the file.

3. Configure the Django project to use the Django Debug Toolbar.

   (a) Open the Django project's settings file, `~/webapps/django_app/myproject/myproject/settings.py`, in a text editor.

   (b) Disable the Django Debug Toolbar's automatic setup. On a new line, add `DEBUG_TOOLBAR_PATCH_SETTINGS = False`.

   (c) Add `'debug_toolbar'` to `INSTALLED_APPS`.

   For example, edit this:

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
)
```

   to this:

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'debug_toolbar',
)
```

(d) Add `'myproject.middleware.WebFactionFixes'` and `'debug_toolbar.middleware.DebugToolbarMiddleware'` to `MIDDLEWARE_CLASSES`.

For example, edit this:

```
MIDDLEWARE_CLASSES = (
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    # ...
)
```

to this:

```
MIDDLEWARE_CLASSES = (
    'myproject.middleware.WebFactionFixes',
    'debug_toolbar.middleware.DebugToolbarMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    # ...
)
```

(e) Set which IP addresses can see the Django Debug Toolbar. One a new line, add `INTERNAL_IPS = ('address',)`, where *address* is your IP address or another address that should have access to debugging information. To add multiple IP addresses, separate each by a comma (for example, `INTERNAL_IPS = ('203.0.113.0', '203.0.113.1', '203.0.113.2')`).

(f) Save and close the file.

(g) Open the Django project's URLs file, `~/webapps/django_app/myproject/myproject/urls.py`, in a text editor.

(h) At the beginning of the file, insert these lines:

```
from django.conf import settings
from django.conf.urls import patterns
```

(i) At the end of the file, append:

```
if settings.DEBUG:
    import debug_toolbar
    urlpatterns += patterns('',
        url(r'^__debug__/', include(debug_toolbar.urls)),
    )
```

(j) Save and close the file.

4. Update static files. Enter `pythonX.Y ./myproject/manage.py collectstatic --noinput` and press `Enter`.
5. Restart Django. Enter `./apache2/bin/restart` and press `Enter`.

The Django Debug Toolbar now appears on the right side of the page when accessed from an IP address listed in `INTERNAL_IPS`.

About `manage.py runserver`

Django's `manage.py` includes the `runserver` command that provides a web server for use while developing Django applications. While neither intended nor recommended for production use, `runserver` can be configured for development use on a WebFaction server.

> **Warning:** Production use of `runserver` is not supported and strongly discouraged. `runserver` is intended as a development aid. `runserver` does not provide sufficient security, stability, and performance for production use.
> From the Django documentation regarding runserver:
>
> > DO NOT USE THIS SERVER IN A PRODUCTION SETTING. It has not gone through security audits or performance tests. (And that's how it's gonna stay. We're in the business of making Web frameworks, not Web servers, so improving this server to be able to handle a production environment is outside the scope of Django.)

1. Create a custom application listening on a port.
   (a) Log in to the control panel.
   (b) Click *Domains / websites* → *Applications*. The list of applications appears.
   (c) Click the *Add new application* button. The *Create a new application* form appears.
   (d) In the *Name* field, enter a name for the application.
   (e) In the *App category* menu, click to select *Custom*.
   (f) In the *App type* menu, click to select *Custom app (listening on port)*.
   (g) If applicable, in the *Machine* menu, click to select a web server.
   (h) Click the *Save* button. The application is created and added to the list of applications.
   (i) Make a note of the port number assigned to the application. It is required in a later step.
2. *Create a website entry* for the custom application.
3. Start the Django development server.
   (a) *Open an SSH session to your account*.
   (b) Switch to the directory of the Django project. Enter `cd $HOME/webapps/django_app/project`, where *django_app* is the name of the Django application as it appears in the WebFaction control panel and *project* is the name of the Django project, and press `Enter`.
   (c) Enter `python manage.py runserver port`, where *port* is the port number assigned to the custom application, and press `Enter`.

You can now access the Django site with a web browser. The Django development server output will appear in the SSH session.

5.3.2 Fixing `ImportError: No module named...` Exceptions

Python will raise an `ImportError` exception whenever it cannot find a particular package or module named in an import statement. This is typically caused by the named module not appearing in Python's module search path. For example, this error sometimes happens while attempting to run `manage.py`.

To correct the problem, the offending module needs to be added to the Python search path. In the case of Django module errors, you must add `$HOME/webapps/django_app/lib/python2.7` to the Python module search

path. To learn more about methods you can use to add to the Python search path, please see *Fixing ImportError Exceptions*.

5.3.3 Fixing Internal Server Errors

Django returns an HTTP status code of 500, an Internal Server Error, when Django encounters runtime errors in a view, such as a syntax error or a view failing to return an object that Django expects.

Closely related to errors in view logic is that Django itself will raise the `TemplateDoesNotExist` exception when an error is encountered in a view, the `DEBUG` setting is set to `False`, and a `500.html` template is not found.

The following two sections will help you resolve errors in views and fix Django's `TemplateDoesNotExist` exception for Internal Server Errors.

Fixing View Errors

When you encounter an Internal Server Error in your Django application, your foremost concern should be to fix your application so that it does not return an HTTP status code of 500. The best web applications respond to problems gracefully, without dumping an ugly error on the user.

The first step in fixing a view error is to identify the cause of the problem. A stack trace of the error will automatically be written to your Django application's *error log*.

If the stack trace doesn't provide you with the information you need to solve the problem, you can set `DEBUG` to `True` in your `settings` module. When `DEBUG` is `True`, a stack trace and detailed configuration information is output to the browser when the error occurs. However, be sure to set `DEBUG` to `False` when you are finished debugging: the debug output contains sensitive information (like file paths and configuration options) and SQL queries are retained in memory, vastly increasing your application's memory consumption.

See also:

General Django Troubleshooting Techniques

Finally, once you've found the source of the error in your view, be sure to update your Django application's suite of tests to keep the error from creeping back into your code.

See also:

Django Documentation > The 500 (server error) view, *Using the Django Debug Toolbar*

Fixing `TemplateDoesNotExist` for Internal Server Errors

Many Django installations raise an additional exception, `TemplateDoesNotExist`, when an error is encountered. Typically this happens when an error occurs while processing a view, no `500.html` template is found, and the `DEBUG` setting is set to `False`.

The default error page (provided by Apache) contains this text:

```
Internal Server Error

The server encountered an internal error or misconfiguration and was unable to complete your request.

Please contact the server administrator, [no address given] and inform them of the time the error occ
```

While the error is not described in detail on the page, the *error log* for the application will look something like this:

```
[Fri Nov 13 14:04:35 2009] [error] [client 127.0.0.1] mod_wsgi (pid=11586): Exception occurred proces
[Fri Nov 13 14:04:35 2009] [error] [client 127.0.0.1] Traceback (most recent call last):
[Fri Nov 13 14:04:35 2009] [error] [client 127.0.0.1]   File "/home/username/webapps/django/lib/pytho
[Fri Nov 13 14:04:35 2009] [error] [client 127.0.0.1]     response = self.get_response(request)
[Fri Nov 13 14:04:35 2009] [error] [client 127.0.0.1]   File "/home/username/webapps/django/lib/pytho
[Fri Nov 13 14:04:35 2009] [error] [client 127.0.0.1]     return self.handle_uncaught_exception(reque
[Fri Nov 13 14:04:35 2009] [error] [client 127.0.0.1]   File "/home/username/webapps/django/lib/pytho
[Fri Nov 13 14:04:35 2009] [error] [client 127.0.0.1]     return callback(request, **param_dict)
[Fri Nov 13 14:04:35 2009] [error] [client 127.0.0.1]   File "/home/username/webapps/django/lib/pytho
[Fri Nov 13 14:04:35 2009] [error] [client 127.0.0.1]     t = loader.get_template(template_name) # Yo
[Fri Nov 13 14:04:35 2009] [error] [client 127.0.0.1]   File "/home/username/webapps/django/lib/pytho
[Fri Nov 13 14:04:35 2009] [error] [client 127.0.0.1]     source, origin = find_template_source(templ
[Fri Nov 13 14:04:35 2009] [error] [client 127.0.0.1]   File "/home/username/webapps/django/lib/pytho
[Fri Nov 13 14:04:35 2009] [error] [client 127.0.0.1]     raise TemplateDoesNotExist, name
[Fri Nov 13 14:04:35 2009] [error] [client 127.0.0.1] TemplateDoesNotExist: 500.html
```

To resolve this problem, create a new template named `500.html` in your templates directory. To see what directories are currently configured as template directories:

1. *Open an SSH session to your account*.
2. Enter `cd $HOME/webapps/`*`django_app/project/project`*, where *django_app* is the name of your Django application as it appears in the control panel and *project* is the name of your Django project, and press `Enter`.
3. Enter `python -c "import settings; print settings.TEMPLATE_DIRS"` and press `Enter`.

Then, in one of the template directories, create a new file named `500.html`. Here's an example `500.html`:

```
<html>
    <head>
        <title>Server Error</title>
    </head>
    <body>
        <h1>Sorry!</h1>
        <p>Sorry, the server has encountered an error.</p>
    </body>
</html>
```

though it's recommended that you add contact information to your server error page, so your users can reach you in the event of a problem.

5.3.4 Reducing Django Start-Up Time

Django may be slow to respond on the first few requests, while mod_wsgi loads and starts the Django application. To reduce these delays, modify your Django application's mod_wsgi configuration. See *Reducing Application Start-Up Time* for step-by-step directions.

5.3.5 Reducing Memory Consumption

See also:

Monitoring Memory Usage and *Reducing mod_wsgi Memory Consumption*

Django applications can be tuned to consume more or less memory. Consider the following strategies to reduce your Django application's memory consumption, but note that some configuration changes—such as allocating fewer

processes or maximum requests—may have a negative impact on overall performance. You may want to experiment with different combinations of configuration values to suit your memory and performance needs.

- **Set `DEBUG` to False:** When Django's `DEBUG` setting is set to `True`, SQL queries and other extra data are kept in memory. See *Using the DEBUG Setting* for more details on what `DEBUG` does and how to change it.
- **Serve static media with a Static-only application:** Serving site media, like style sheets, JavaScript, CSS files, and Django admin media, with the Django application is an inefficient use of memory. Instead, let the system-wide nginx-process *serve static media with a static application*.
- **Reduce the number of objects loaded into memory**: Use the features of Django's ORM to avoid loading more objects than you need into memory.

  For example, suppose you want to retrieve all the users from your database with the first name *John*. Instead of fetching all user objects and sifting through for those you need, like this:

```
# Don't do this!
subset_of_users = []
for user in Users.objects.all():
    if user.first_name == 'John':
        subset_of_users.append(user)  # This is a terrible idea!
```

  try using Django's QuerySet API to load only the objects you need:

```
subset_of_users = Users.objects.filter(first_name__exact='John')
```

  Not only is the QuerySet version fewer lines code, but it uses less memory too.

  See also:

  Django documentation: Making queries

5.3.6 Reset an Admin Password

To reset a Django admin account password:

1. *Open an SSH session to your account*.
2. Switch to the Django project directory. Enter `cd $HOME/webapps/django_app/project`, where *django_app* is the name of the Django application as it appears on the control panel and *project* is the name of the project, and press `Enter`.
3. Start the Django shell. Enter `python2.X manage.py shell`, where *X* is the minor Python version number associated with your Django application, and press `Enter`.
4. Import the User class. Enter `from django.contrib.auth.models import User` and press `Enter`.
5. Get the object for the user whose password you want to change. Enter `u = User.objects.get(username__exact='username')` and press `Enter`.
6. Change the password. Enter `u.set_password('password')`, where *password* is the new password, and press `Enter`.

   See also:

   See *Strengthening Passwords* for important information about choosing passwords.
7. Save the change. Enter `u.save()` and press `Enter`.
8. Exit the shell. Press `Control + D`.

You can now log in with the new password.

5.3.7 Accessing REMOTE_ADDR

When a Django application's Apache instance proxies requests to Django, the `REMOTE_ADDR` header is not set with the clients's IP address. Instead, the IP address is available as the first IP address in the comma separated list in the `HTTP_X_FORWARDED_FOR` header. When you need `REMOTE_ADDR`, access the first of `HTTP_X_FORWARDED_FOR`'s IP addresses instead.

Alternatively, you can use this middleware class to automatically set `REMOTE_ADDR` to the value of `HTTP_X_FORWARDED_FOR`:

```
class WebFactionFixes(object):
    """Sets 'REMOTE_ADDR' based on 'HTTP_X_FORWARDED_FOR', if the latter is
    set.

    Based on http://djangosnippets.org/snippets/1706/
    """
    def process_request(self, request):
        if 'HTTP_X_FORWARDED_FOR' in request.META:
            ip = request.META['HTTP_X_FORWARDED_FOR'].split(",")[0].strip()
            request.META['REMOTE_ADDR'] = ip
```

CHAPTER

SIX

DRUPAL

Drupal is an open source content management system which can be used to build blogs, portals, and other web sites. WebFaction provides a control panel installer for Drupal.

See also:

Drupal is a PHP-based application. Please see *PHP* and *Static Files, CGI Scripts, and PHP Pages* for additional documentation.

6.1 Backing Up Drupal

Backing up your Drupal application is an important routine task. We also recommend that you back up your application before making major changes, such as upgrading Drupal.

To back up a Drupal application:

1. *Open an SSH session to your account*.
2. Switch to your Drupal application directory. Enter `cd $HOME/webapps/drupal_app`, where *drupal_app* is the name of your Drupal application, and press `Enter`.
3. Make `drush` executable. Enter `chmod u+x ./bin/drush` and press `Enter`.

   Note: You can skip this step if you've already done so for this Drupal installation.

4. Set the PHP version for Drush. Enter `export DRUSH_PHP=$(which php56)` and press `Enter`.
5. Back up Drupal. Enter `./bin/drush archive-backup default` and press `Enter`. An archive containing a backup of your Drupal site is created in `~/drush-backups`.

   Optionally, you can specify a destination path for your backup by adding `--destination path` to the command. For example, to back up to `~/latest.tar.gz`, enter `./bin/drush archive-backup default --destination=-$HOME/latest.tar.gz` and press `Enter`.

The Drupal backup file has been created.

6.2 Updating Drupal

Updating Drupal can protect a Drupal site from security vulnerabilities and fix bugs. To update Drupal to the latest version using Drush:

1. *Open an SSH session to your account*.
2. *Back up your Drupal site*.
3. Refresh Drush's list of known Drupal versions. Enter `./bin/drush rf` and press `Enter`.
4. Upgrade Drupal core. Enter `./bin/drush up drupal` and press `Enter`.

   Alternatively, to install only security updates, run `./bin/drush up drupal --security-only`

   You can also choose to upgrade to a specific Drupal core version by appending a version number. For example, to upgrade to Drupal 8.0.1, run `./bin/drush up drupal-8.0.1`.

The Drupal site is updated.

6.3 Configuring Clean URLs

Drupal's Clean URLs feature replaces URLs like `http://www.example.com/?q=node/123` with URLs like `http://www.example.com/node/123`.

Note: Drupal 8 applications have clean URLs enabled by default.

See also:

Drupal Clean URLs documentation

To configure Drupal to use Clean URLs:

1. Activate the `RewriteBase` Apache directive.
   (a) *Open an SSH session to your account*.
   (b) Open `~/webapps/application/.htaccess` in a text editor, where *application* is the name of the Drupal application.
   (c) Configure the `RewriteBase` directive.
   - If the Drupal application is mounted on a non-root URL path (for example, `http://domain/drupal`), replace `# RewriteBase /` with `RewriteBase /path`, where *path* is the non-root URL path on which the Drupal application is mounted.
   - If the Drupal application is mounted on the root URL path, uncomment `# RewriteBase /`. Replace `# RewriteBase /` with `RewriteBase /`.

   (d) Save and close the file.
2. Configure Drupal to use Clean URLs.
   (a) Log in to Drupal.
   (b) Click *Administer*. The administration menu appears.
   (c) Click *Modules*. The Modules configuration page appears.
   (d) In the list of modules, click to select *Path*.
   (e) Click the *Save configuration* button. The page reloads with a confirmation message.
   (f) Click *Site configuration*. The *Site configuration* page appears.
   (g) Click *Clean URLs*. The *Clean URLs* form appears.
   (h) Click to select *Enabled*.

(i) Click the *Save configuration* button. The page reloads with a confirmation message.

The Drupal application will now use clean URLs.

6.4 Serving Uploads Faster

To improve the performance of your Drupal site, create a symbolic link application to serve uploaded files directly.

See also:

If you want to use an external CDN service, see the CDN module.

To create and use a symlink application:

1. *Log in to the WebFaction control panel*.
2. Click *Domains / websites* → *Websites*. The list of websites appears.
3. Click on the name of the site that includes your Drupal application. The site's details appear.
4. Click *Add an application* → *Create a new application*. The *Create a new web app* form appears.
5. In the name field, enter a name for the application.
6. In the *App category* menu, click to select *Symbolic link*.
7. In the *Extra info* field, enter `/home/username/webapps/app/sites/default/files/`, where *username* is your username and *app* is the name of the Drupal application.
8. In the *URL* field, enter the URL path to the uploaded files. Typically, the URL is the Drupal application's URL path and `sites/default/files`.

   For example, if the Drupal application's URL path is `/blog`, enter `blog/sites/default/files`.

   If the Drupal application is at the root of the domain, enter `sites/default/files`.
9. Click the *Save* button. The application is installed and added to the website's list of applications.
10. Click the *Save* button.

Future requests for Drupal uploads are served by the server's front-end process, skipping the unneeded PHP interpreter.

6.5 Speeding Up Drupal with Caching

Some Drupal sites may see improved page load times by enabling caching.

To enable Drupal caching:

1. Log in to your Drupal site.
2. In the tool bar, click *Configuration*. The *Configuration* menu appears.
3. In the *Development* section, click *Performance*. A settings form appears.
4. In the *Page cache maximum age* menu, click to select a duration. Try experimenting with different values to find the best performance. *1 hour* is a good starting value.
5. Click the *Save configuration* button.

Future requests for cached resources will require less response time.

6.6 Sending Email from Drupal

To configure Drupal to be able to send email messages:

1. Log in to your Drupal site.
2. In the tool bar, click *Configuration*. The *Configuration* menu appears.
3. In the *System* section, click *Site information*. The *Site information* page appears.
4. In the *Email address* field, enter the outgoing email address for notification, password recovery, and other Drupal email.
5. Click the *Save configuration* button. A *The configuration options have been saved* notification appears at the top of the page.

Future outgoing messages will be sent from the designated address.

6.7 Troubleshooting

6.7.1 Strange Characters on 404 Pages

See also:

PHP > *Troubleshooting* > *Strange Characters in Output*

CHAPTER

SEVEN

GIT

Git is an open source distributed version control system.

The `git` command-line interface is installed on every WebFaction server in the `/usr/bin/` directory. By default, `git` will be in your `PATH`. To see a complete list of commands during an SSH session, enter `git help` and press `Enter`.

The Git web application can serve multiple Git repositories from your account. With the Git web application, you can clone, push, and pull over HTTP. You can also view repositories with a web browser.

See also:

Bazaar, *Mercurial*

7.1 Installing the Git Web Application

To install a Git application:

1. Log in to the control panel.
2. Click *Domains / websites* → *Applications*. The list of applications appears.
3. Click the *Add new application* button. The *Create a new application* form appears.
4. In the *Name* field, enter a name for the Git application.
5. In the *App category* menu, click to select *Git*.
6. In the *Extra info* field, enter a password for the default user.

   See also:

   See *Strengthening Passwords* for important information about choosing passwords.
7. Click the *Save* button.

The Git application is installed and added to the list of applications. To make your Git application's repositories available on the Web, *add the Git application to a website record*.

The Git application creates a directory, `~/webapps/git`, where *git* is the name of the application. The Git application directory contains:

- the CGI scripts that serve your repositories,
- `.htaccess` and `.htpasswd` files, which determine how and to whom your Git repositories are served,
- a subdirectory, `repos`, to store your Git repositories, and
- an example repository, in the `repos/proj.git` subdirectory.

7.2 Creating a New Repository

To create a new repository:

1. Open an SSH session to your account.
2. Switch to the Git application's `repos` subdirectory, where your repositories are stored. Enter `cd ~/webapps/git/repos`, where *git* is the name of the Git application as it appears on the control panel, and press `Enter`.
3. Create the repository. Enter `git init --bare {repo}.git`, where *repo* is the name of the new repository, and press `Enter`.

   Note: The `.git` extension for the new repository is required. Additionally, the rest of the repository name must not contain any dots. For example, `myrepo.git` is acceptable, while `myrepo.com.git` is not.

4. Switch to the new repository's directory. Enter `cd repo.git` and press `Enter`.
5. Enable HTTP push. Enter `git config http.receivepack true` and press `Enter`.

7.3 Cloning a Repository Hosted on Your WebFaction Server

You can clone a Git repository locally, over HTTP (anonymous or authenticated), and over SSH. All Git clone commands follow the same general formula: `git clone remote`, where *remote* is a path to a Git repository (for local repositories), a URL to a Git repository (for HTTP connections) or an `scp`-like reference to a repository (for SSH connections). Git commands issued over SSH also respect key-based authentication.

For example:

Type	Git Command
Local	`git clone /home/username/webapps/app/repos/proj.git`
HTTP (Anonymous)	`git clone http://domain/URL_path/proj.git`
HTTP (Authenticated)	`git clone http://git_user@domain/URL_path/proj.git`
SSH	`git clone username@username.webfactional.com:/home/username/webapps/app/r`

where:

- *app* is the name of the Git application as it appears on the WebFaction control panel,
- *proj* is the name of the Git repository,
- *domain* is the domain name where the Git application is available, as configured in a website record on the WebFaction control panel (for example, `git.example.com`),
- *URL_path* is the URL path (or URL mount point) where the Git application is available, as configured in a website record on the WebFaction control panel,
- *git_user* is a valid user as configured in the Git application's `.htpasswd` file,

  See also:

  Managing Users

- and *username* is your WebFaction username.

Finally, for most repositories—especially if you anticipate adding large files, numerous files, or making many changes to your repository between pushes and pulls—you should change the `postBuffer` setting on your local repository clone:

1. Switch to the local repository directory. Enter `cd path`, where *path* is the path to the repository, and press `Enter`.
2. Enter `git config http.postBuffer bytes`, where *bytes* is the maximum number of bytes permitted, and press `Enter`.

   For example, to permit pushes up to 500 megabytes, enter `git config http.postBuffer 524288000` and press `Enter`.

7.4 Enabling Anonymous Read Access

Warning: Permissions apply to all repositories within a Git application. To split permissions across different sets of repositories, *create additional Git applications*.

To permit anonymous users to view and clone your repositories:

1. Open an SSH session to your account.
2. Open `~/webapps/git/.htaccess`, where *git* is the name of the Git application, in a text editor.
3. Comment out these lines with a `#` character:

```
AuthName "Git Authentication"
AuthType Basic
Require valid-user
AuthUserFile /home/<username>/webapps/<git>/.htpasswd
```

   such that it looks like this:

```
# AuthName "Git Authentication"
# AuthType Basic
# Require valid-user
# AuthUserFile /home/<username>/webapps/<git>/.htpasswd
```

4. Save and close the file.

7.5 Managing Users

When you create a Git application, the application starts with a single user: your WebFaction control panel user name. You can manage users with the `htpasswd` utility and the `.htpasswd` file.

7.5.1 Add a User or Change a Password

To add a new user or modify an existing user's password:

1. Open an SSH session to your account.
2. Switch to the Git application directory. Enter `cd ~/webapps/git`, where *git* is the name of the Git application, and press `Enter`.

3. Enter `htpasswd .htpasswd user`, where *user* is a new or existing username, and press `Enter`.
4. Complete the password prompts.

   See also:

   See *Strengthening Passwords* for important information about choosing passwords.

7.5.2 Delete a User

1. Open an SSH session to your account.
2. Open `~/webapps/git/.htpasswd`, where *git* is the name of the git application, in a text editor.
3. Delete the line containing the username you want to remove.
4. Save and close the file.

7.6 Removing a Repository

Warning: Removing a repository cannot be undone.

To remove a repository:

1. Open an SSH session.
2. Delete the repository. Enter `rm -r ~/webapps/git/repos/repo.git`, where *git* is the name of the Git application and *repo* is the name of the repository, and press `Enter`.

7.7 Using HTTPS

If you're using Git with a domain that already has a valid security certificate set up and you've configured the Git application on a website record with HTTPS enabled, then use the HTTPS protocol (`https://`) with Git URLs.

You can also use WebFaction's security certificate by disabling Git's SSL certificate verification. While cloning, use the `GIT_SSL_NO_VERIFY` environment variable. For example:

```
export GIT_SSL_NO_VERIFY=true
git clone https://demo@demo.webfactional.com/proj.git myproj
```

Then, disable SSL certificate verification for the repository to enable push and pull operations:

```
# switch to the repository directory
git config http.sslVerify false
```

7.8 Troubleshooting

7.8.1 Error: `RPC failed; result=22, HTTP code = 411`

If you attempt to push a large set of changes to a Git repository with HTTP or HTTPS, you may get an error message such as `error: RPC failed; result=22, HTTP code = 411`. This is caused by a Git configuration

default which limits certain HTTP operations to 1 megabyte. To override this limitation, *update the postBuffer setting* on your cloned repository.

CHAPTER

EIGHT

JOOMLA

Joomla is an open source, PHP-based content management system.

See also:

Joomla is a PHP-based application. Please see *PHP* and *Static Files, CGI Scripts, and PHP Pages* for additional documentation.

8.1 Upgrade Joomla

You should upgrade Joomla after every security release (or more often, if you want to use Joomla's latest features). To upgrade Joomla:

1. Open `http://domain.example/administrator/` in a web browser, where *domain.example* is the Joomla site's domain name and URL path.
2. In the *User Name* field, enter your administrative username.
3. In the *Password* field, enter your password.
4. Click the *Log in* button. The Joomla control panel appears.
5. Check the *Quick Links* sidebar for upgrade availability. If an upgrade is available, then an *Update now!* link with the new version number appears in the sidebar.
6. Click the *Update now!* link.
7. Click the *Install the update* button. Joomla downloads and installs the update. The upgrade may take a few minutes to complete. When the update finishes successfully, a confirmation message with the Joomla version number appears.

Joomla is upgraded to the most recent release.

8.2 Sending Email

To configure Joomla to send mail:

1. Log in to the Joomla administration site with an *Administrator* or *Super Administrator* account.
2. Click *System* → *Global Configuration*. The *Global Configuration* page appears.
3. Click the *Server* tab.
4. In the *From email* field, enter an outgoing email address.
5. In the *From Name* field, enter your name as you'd like it to appear to message recipients.

6. Click the *Save* button. A confirmation message appears at the top of the page.

8.2.1 Sending Email with an SMTP Server

You may configure Joomla to send email messages using an SMTP server. This method is optional. To configure Joomla to send mail using WebFaction's SMTP server:

1. Log in to the Joomla administration site with an *Administrator* or *Super Administrator* account.
2. Click *System* → *Global Configuration*. The *Global Configuration* page appears.
3. Click the *Server* tab.
4. From the *Mailer* menu, click to select *SMTP*.
5. In the *From email* field, enter an outgoing email address.
6. In the *From Name* field, enter your name as you'd like it to appear to message recipients.
7. In the *SMTP Authentication* section, click to select *Yes*.
8. In the *SMTP Security* menu, click to select *SSL*.
9. In the *SMTP Port* field, enter `465`.
10. In the *SMTP Username* field, enter a mailbox name.
11. In the *SMTP Password* field, enter the mailbox's password.
12. In the *SMTP Host* field, enter `smtp.webfaction.com`.
13. Click the *Save* button. A confirmation message appears at the top of the page.

Outgoing messages are now sent through WebFaction's SMTP server.

CHAPTER

NINE

MEMCACHED

Memcached is an open source caching system.

9.1 Setting Up Memcached

Memcached is installed server-wide on all of our machines. To start Memcached:

1. Open an SSH session.
2. Enter
   `memcached -d -m memory -s $HOME/memcached.sock -P $HOME/memcached.pid`
   where *memory* is the maximum number of megabytes of memory you want Memcached to use, and press `Enter`.

Once your Memcached instance is up and running, you can access it by the path of the socket file (`~/memcached.sock`) with the software or library which uses memcached.

9.2 Using Memcached with an Application

To configure Django to use Memcached, please see *Configuring Django to Use Memcached*.

If you want to make custom use of Memcached, there are many libraries available which make it easier to interact with a Memcached socket. Common libraries include:

- Memcache for PHP
- python-memcached for Python
- Dalli for Ruby

9.3 Shutting Down Memcached

To stop your Memcached process:

1. Open an SSH session.
2. Enter `kill $(cat $HOME/memcached.pid)` and press `Enter`.

CHAPTER

TEN

MERCURIAL

Mercurial, commonly referred to as `Hg` (the symbol for the element Mercury), is an open source distributed version control system.

See also:

Bazaar, *Git*

10.1 Installing Mercurial

To install Mercurial:

1. Open an SSH session to your account.
2. Enter `easy_install-X.Y Mercurial`, where *X.Y* is the Python version to use to install Mercurial, and press `Enter`.

   For example, to install Mercurial with Python 2.7, enter `easy_install-2.7 Mercurial` and press `Enter`.
3. Create a `.hgrc` configuration file, if it does not already exist. Enter `touch $HOME/.hgrc` and press `Enter`.
4. Open `$HOME/.hgrc` in a text editor.
5. Add the following lines to `$HOME/.hgrc`:

```
[ui]
username = name <email>
```

   where `name` is your name and `email` is your email address as they will appear in commit messages made from your Mercurial installation.
6. Save and close the file.

You can now use the command line tool **hg** with Mercurial repositories. Enter `hg help` and press `Enter` for a complete list of commands. To learn more about using Mercurial, check out the official Quick Start guide and the Mercurial book.

10.2 Publishing Mercurial Repositories to the Web

You can create an application to serve `hgweb.cgi` (also known as *HgWeb*), which makes it possible to view repositories with a web browser, as well as push and pull changes over HTTPS.

Note: These directions assume you already have Mercurial installed. If you have not already installed Mercurial, please see *Installing Mercurial*.

10.2.1 Create an Application and Website to Serve `hgweb.cgi`

The first step in serving Mercurial repositories is to create an application and website to serve the CGI script.

1. Log in to the control panel.
2. Click *Domains / websites* → *Websites*. The list of websites appears.
3. Click the *Add new website* button. The *Create a new website* form appears.
4. In the *Name* field, enter a website name.
5. If applicable, in the *Machine* menu, click to select the server to host the website.
6. Click to select *Encrypted website (https)*.

   See also:

   Secure Sites (HTTPS)
7. For each domain name you want to use with the website, add it to the list of domains. In the *Domains* field, enter the domain name. Enter one or more domain names. If the domain has not yet been added to the control panel, click the *Create* link that appears at the bottom of the list of domains to add it.

   Note: Don't forget to *point new domains to the WebFaction name servers*.
8. For each domain name you want to use with the website, add it to the list of domains. In the *Domains* field, enter the domain name. Enter one or more domain names.
9. Click *Add an application* → *Create a new application*. The *Create a new web app* form appears.
10. In the *Name* field, enter a name for the `hgweb.cgi` application.
11. In the *App Category* menu, click to select *Static*.
12. In the *URL* field, enter the URL path.

    Note: The first application added to a website is assigned to the root URL path (`/`).
13. Click the *Save* button. The application is installed and added to website's list of applications.
14. Click the *Save* button. The website is created and added to the list of websites.

10.2.2 Create a Place to Store Mercurial Repositories

Next, a directory needs to be created where the repositories themselves will be stored. Typically, this will be a directory outside of the Static/CGI/PHP application's directory.

1. *Open an SSH session to your account*.
2. Create the repository directory. Enter `mkdir path`, where *path* is the path to a new directory for repositories, and press `Enter`. For example, enter `mkdir $HOME/hg` and press `Enter`.

10.2.3 Download and Install `hgweb.cgi`

Now, download `hgweb.cgi`, a CGI script which will make repositories available on the Web.

1. Open an SSH session to your account.
2. Change to the Static/CGI/PHP application's directory. Enter `cd $HOME/webapps/static_app/`, where *static_app* is the name of the static application, and press `Enter`.
3. Remove the existing `index.html`. Enter `rm index.html` and press `Enter`.
4. Find your Mercurial version number. Enter `hg version` and press `Enter`.
5. Download `hgweb.cgi` for your version of Mercurial. Enter `wget https://www.mercurial-scm.org/repo/hg-stable/raw-file/version/hgweb.cgi`, where *version* is the version number for your Mercurial installation, and press `Enter`.

10.2.4 Configure `hgweb.cgi`

Now, configure `hgweb.cgi` so it knows where to look for your Mercurial installation and repositories.

1. Change to the Static/CGI/PHP application's directory. Enter `cd $HOME/webapps/static_app/` and press `Enter`.
2. Make `hgweb.cgi` executable. Enter `chmod 711 hgweb.cgi` and press `Enter`.
3. Make `$HOME/webapps/static_app` executable. Enter `chmod 711 .` and press `Enter`.
4. Open `hgweb.cgi` in a text editor.
5. Configure `hgweb.cgi` to use the correct version of Python. Edit this line:

```
#!/usr/bin/env python
```

   to replace *python* with `pythonX.Y`, where *X.Y* is the version of Python you used to install Mercurial.

6. If you are using a version of Mercurial later than 1.5, edit this line:

```
config = "/path/to/repo/or/config"
```

   to this:

```
config = "/home/<username>/webapps/<static_app>/hgweb.config"
```

   where `<username>` is your username and `<static_app>` is the name of your Static/CGI/PHP application.

7. Uncomment these lines:

```
#import sys
#sys.path.insert(0, "/path/to/python/lib")
```

   and replace `/path/to/python/lib` with the path to the directory where you installed Mercurial. Typically, this directory will be `/home/username/lib/pythonX.Y`, where *username* is your username and *X.Y* is the version of Python you used to install Mercurial.

8. Save and close the file.
9. Open a new file named `hgweb.config`.

10. Configure the path to your repositories. Append the following lines:

```
[paths]
/ = /path/to/repositories/**
```

where `/path/to/repositories` is the path to your Mercurial repositories (for example, `/home/username/hg`).

Note: Both asterisks (`*`) are required for `hgweb.cgi` to find your repositories; they permit `hgweb.cgi` to find Mercurial repositories which are in a subdirectory of the path specified.

---

11. Enable cleaner URLs.

If the URL path for the application is a root of a domain, append the following lines:

```
[web]
baseurl =
```

If the URL path for the application is not a root of a domain (for example, `www.example.com/hg/`), append the following lines:

```
[web]
baseurl = urlpath
```

where `urlpath` is application's URL path (for example, `/hg`).

12. Save and close the file.

10.2.5 Configure Apache to Use `hgweb.cgi`

The next step in publishing your Mercurial repositories is to instruct the Apache web server to use `hgweb.cgi` to respond to requests.

1. Change to the Static/CGI/PHP application's directory. Enter `cd $HOME/webapps/static_app/` and press `Enter`.
2. Open or edit a file named `.htaccess`.
3. Edit the file so that it looks like the following

```
Options +ExecCGI
RewriteEngine On
# / for root directory; specify a complete path from / for others
RewriteBase /
RewriteRule ^$ hgweb.cgi  [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule (.*) hgweb.cgi/$1  [QSA,L]
```

Note: If you will not be hosting your repositories from the root URL path, replace `RewriteBase /` with `RewriteBase /path/to/directory`. For example, if you were to host your repositories at `http://www.example.com/hg` you would substitute `RewriteBase /` with `RewriteBase /hg`.

---

4. Save and close the file.

You can now visit the domain and path specified in the website entry to see your Mercurial repositories. The listing is probably empty now, however. Create new repositories with `hg init` in your repository directory to populate the list.

10.2.6 Secure and Push to Repositories Served by `hgweb.cgi`

Now that you have a working repository browser, you can enable pushing to those repositories over HTTPS for authenticated users.

1. *Open an SSH session to your account*.
2. Change to the Static/CGI/PHP application's directory. Enter `cd $HOME/webapps/static_app/` and press `Enter`.
3. Open `.htaccess` in a text editor.
4. Add the following lines to the bottom of `.htaccess` to permit pushes only by authenticated users:

```
AuthType Basic
AuthName MyHgWebDir
AuthUserFile /home/<username>/webapps/<static_app>/.htpasswd

Satisfy Any
<Limit POST PUT>
    Require valid-user
</Limit>

<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>
```

   where `<username>` is your username and `<static_app>` is the name of the Static/CGI/PHP application.

   Note: Alternatively, you can use the following lines to prohibit **all** access except for authenticated users:

```
AuthType Basic
AuthName MyHgWebDir
AuthUserFile /home/<username>/webapps/<static_app>/.htpasswd
Require valid-user

<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>
```

   ---

5. Save and close the file.
6. Now, create an initial username and password that can push to the repositories. Enter `htpasswd -c .htpasswd username` and press `Enter`. A prompt appears for a password.

   See also:

   See *Strengthening Passwords* for important information about choosing passwords.

7. Enter the password for the new user and press `Enter`. A prompt to confirm the password appears.
8. Enter the password again and press `Enter`.

9. Open `hgweb.config` in a text editor.
10. Add the following lines to the file

```
[web]
allow_push = *
```

> **Note:** Alternatively, you can replace `*` with a comma separated list of users authorized to push. Or, for more finely grained control, you can add these lines on a repository-by-repository basis in the repositories' `.hg/hgrc` files.

11. Save and close the file.

You can now push to the repositories as an authenticated user.

10.3 Troubleshooting

10.3.1 Error: *bash: hg: command not found*

If you're cloning, pulling, or pushing from Mercurial repositories via ssh (for example, `hg clone ssh://username@username.webfactional.com/path/to/repo`), you may encounter an error, *bash: hg: command not found*, which appears when the remote session is unable to locate the `hg` script.

To fix this problem:

1. Open your local `.hgrc` file in a text editor. `.hgrc` is typically found in your home directory.
2. In the `[ui]` section, add a new line containing `remotecmd = ~/bin/hg`.
3. Save and close the file.

CHAPTER

ELEVEN

MOD_WSGI

mod_wsgi is an Apache HTTP Server module for running Python software that's compatible with the Web Server Gateway Interface (**PEP 333**). A mod_wsgi application as created with the control panel provides an Apache HTTP Server with the mod_wsgi module.

11.1 Reducing Application Start-Up Time

Some mod_wsgi-based sites' first page load may take a long time as mod_wsgi imports and starts the Python application. To minimize loading delays, configure mod_wsgi to load the application on process start, instead of on first request. To configure mod_wsgi to load the application on process start:

1. *Open an SSH session to your account*.
2. Open `$HOME/webapps/app/apache2/conf/httpd.conf`, where *app* is the application name, in a text editor.
3. On the line starting with `WSGIScriptAlias`, append `process-group=<app> application-group=%{GLOBAL}`, where `<app>` is the name of your application.

   For example, with a Django application named `django_demo` running a project named `myproject`, this line:

   ```
   WSGIScriptAlias / /home/demo/webapps/django_demo/myproject/myproject/wsgi.py
   ```

   becomes:

   ```
   WSGIScriptAlias / /home/demo/webapps/django_demo/myproject/myproject/wsgi.py process-group=djang
   ```

4. Save and close the file.
5. Restart Apache. Enter `$HOME/webapps/app/apache2/bin/restart` and press `Enter`.

After restarting, subsequent first page load times are shorter.

11.2 Reducing mod_wsgi Memory Consumption

mod_wsgi applications can be tuned to consume more or less memory. These strategies may help reduce your application's memory consumption, but note that some configuration changes may degrade overall performance. Experiment with different combinations of configuration values to suit your memory and performance needs.

- **Use the `WSGIDaemonProcess` directive's `processes` setting conservatively:** Start fewer processes by setting a small value for the `WSGIDaemonProcess` directive's `process` setting.

  In `~/webapps/application/apache2/conf/httpd.conf`, where *application* is the name of the mod_wsgi-based application, edit this line:

```
WSGIDaemonProcess <app> processes=<num> python-path=/home/<user>/webapps/<app>:/home/<user>/weba
```

  where:

  - `<app>` is the name of the mod_wsgi-based application,
  - `<num>` is the number of allowed processes, and
  - `<user>` is your account name.

  For example, two to five processes will be enough for a Django site that uses a separate application to serve static media.

- **Use the `WSGIDaemonProcess` directive's `maximum-requests` setting:** If an application is not releasing memory in a way that's beyond your control (for example, a library used by your application is leaking memory), then configure mod_wsgi to serve fewer requests before stopping and replacing a child process.

  In `~/webapps/application/apache2/conf/httpd.conf`, where *application* is the name of the mod_wsgi-based application, edit this line:

```
WSGIDaemonProcess <app> processes=5 python-path=/home/<user>/webapps/<app>:/home/<user>/webapps/
```

  where:

  - `<app>` is the name of the mod_wsgi-based application,
  - `<num>` is the maximum number of requests, and
  - `<user>` is your account name.

  Reasonable values are typically between 100 and 1000 requests, though you may need to experiment to find an appropriate setting for your application.

See also:

mod_wsgi Configuration Directives

11.3 Using a virtual environment with mod_wsgi

You can use a mod_wsgi application with a Python virtual environment (also known as a virtualenv) by modifying a configuration file.

To reconfigure a mod_wsgi application (versions 3.4 or later) to use a virtual environment:

1. *Open an SSH session to your account*.
2. Open `~/webapps/application/apache2/conf/httpd.conf` in a text editor.
3. Set the path to your virtual environment. In the `WSGIDaemonProcess` directive, replace `python-path=/home/username/webapps/app/lib/python3.5` with `python-home=venv`, where *venv* is the path to your virtual environment.

   For example, if your mod_wsgi application is in `/home/demo/webapps/mywsgiapp/` and your virtual environment is in `/home/demo/webapps/mywsgiapp/venv/`, replace

`python-path=/home/demo/webapps/mywsgiapp/lib/python3.5` with `python-home=/home/demo/webapps/mywsgiapp/venv/`.

4. Optionally, set additional Python path directories, such as the path to your Python project. In the `WSGIDaemonProcess` directive, add `python-path=`*`path`*. You can add more than one `python-path`, if needed.
5. Save and close the file.
6. Restart Apache. Enter `$HOME/webapps/`*`app`*`/apache2/bin/restart` and press `Enter`.

The mod_wsgi now uses your virtual environment.

CHAPTER

TWELVE

MONGODB

MongoDB is an open source document-oriented database.

Warning: WebFaction servers less than `Web300` or less than `Dweb89` only support the 32-bit version of MongoDB. The 32-bit version of MongoDB is limited to storing approximately 2.5GB of data. For more information on this limitation, please see the MongoDB blog article "32 bit limitations."

12.1 Installing MongoDB

To install MongoDB:

1. Create a custom application to make a directory and receive a port assignment for your MongoDB installation.
   (a) Log in to the control panel.
   (b) Click *Domains / websites* → *Applications*. The list of applications appears.
   (c) Click the *Add new application* button. The *Create a new application* form appears.
   (d) In the *Name* field, enter a name for the application.
   (e) In the *App Category* menu, click to select *Custom*.
   (f) In the *App Type* menu, click to select *Custom app (listening on port)*.
   (g) If applicable, in the *Machine* menu, click to select a web server.
   (h) Click the *Save* button. The application is created and added to the list of applications.
   (i) Make a note of the application's port number. It is required in a later installation step.
2. Open an SSH session to your account.
3. Install MongoDB.
   (a) Switch to the MongoDB application directory. Enter `cd $HOME/webapps/application`, where *application* is the name of the MongoDB application as it appears on the control panel, and press `Enter`.
   (b) Download the MongoDB Linux archive. Enter `wget url`, where *url* is the download URL for MongoDB, and press `Enter`.

   See the MongoDB downloads page for download URLs. If you're using a server less than `Web300` or less than `Dweb89`, choose the Linux 32-bit legacy download URL. If you're using any other server, choose the Linux 64-bit legacy download URL.

An archive, named `mongodb-linux-architecture-version.tgz` where *architecture* is `x86_64` or `i686` and *version* is the MongoDB version number, is created in the current directory.

(c) Extract the contents of the archive. Enter `tar -xzf mongodb-linux-architecture-version.tgz` and press `Enter`. A new directory is created in the current directory that contains the MongoDB files.

4. Configure MongoDB.

(a) Create a database data directory. Enter `mkdir data` and press `Enter`.

(b) Start the MongoDB database. Enter `$HOME/webapps/application/mongodb-linux-architecture-version/bin/mongod --auth --d` where *number* is the port number provided by the control panel, and press `Enter`.

> **Warning:** Always start MongoDB with the `--auth` switch to prevent unauthorized users from tampering with your data.

(c) Open a second SSH session to your account. Leave the first session with MongoDB running.

(d) In the second session, connect to the database. Enter `$HOME/webapps/application/mongodb-linux-architecture-version/bin/mongo localhost:p` and press `Enter`. An interactive prompt (`>`) appears.

(e) Add an administrator user, which can be used to manage other users. Enter `db.createUser({user: "<username>", pwd: "<password>", roles: ["userAdminAnyData` where `<username>` is the new administrator username of your choice and `<password>` is the new user's password, and press `Enter`.

(f) Authenticate as the administrator user. Enter `db.auth("username", "password")`, where *username* is the administrator username you chose in the previous step and *password* is the user's password, and press `Enter`. If authentication is successful, a `1` appears.

You can use this administrator user to manage other users and their permissions. See the MongoDB documentation's on adding users for more detail.

See also:

For more information about user authentication, see the MongoDB Security documentation.

(g) Close the database connection. Enter `exit` and press `Enter`. You may also end the SSH session.

The database is configured and ready. You may start and stop the database as needed.

To start the database, run
`$HOME/webapps/application/mongodb-linux-architecture-version/bin/mongod --auth --dbpath $H`

To stop MongoDB while it is running in the foreground, press `Ctrl + C`. For other ways to stop the database, see the MongoDB Stopping mongod documentation.

See also:

To learn more about using MongoDB, please see the MongoDB tutorial documentation. To learn more about using your preferred programming language with MongoDB, please see the MongoDB drivers list.

CHAPTER

THIRTEEN

MOVABLE TYPE

Movable Type is a blog publishing application.

13.1 Configuring Movable Type to Send Mail

To configure Movable Type to be able to send email messages during the installation's the *Mail Configuration* page:

1. In the *Send email via* menu, click to select *Sendmail*.
2. In the *Mail address to which test email should be sent* enter an email address where you wish to receive a test email.
3. In the *From mail address* field, enter an email address configured in the WebFaction control panel.
4. Click the *Continue* button.
5. Complete the installation.
6. Log in to the Movable Type *User Dashboard*.
7. Next to *User Dashboard*, click the navigation icon and then click *System Overview*. The *System Overview* page appears.
8. Click *Settings* → *General*. The *General Settings* page appears.
9. In the *System Email* field, enter the email address Movable Type should send mail as. This address must be configured in the WebFaction control panel.
10. Click the *Save Changes* button.

To configure Movable Type to be able to send email messages at any other time:

1. Open an SSH session to your account.
2. Switch the Movable Type directory. Enter `cd ~/webapps/mt` where *mt* is the name of the Movable Type application as it appears on the control panel, and press `Enter`.
3. Open `mt-config.cgi` in a text editor (the file may be in a subdirectory depending on where you install the Movable Type application files).
4. Modify or add these configuration directives:

```
MailTransfer sendmail
SendMailPath /usr/sbin/sendmail
EmailAddressMain <address>
```

   where `<address>` is Movable Type's outgoing email address. This address must be registered on the WebFaction control panel.

5. Save and close the file.

13.2 Configuring Movable Type to Send Email with SMTP

Optionally, you may configure Movable Type to send mail with an SMTP server. WebFaction's SMTP server and others require authentication. By default, Movable Type cannot send mail through an SMTP server that requires authentication; to overcome this obstacle, install and configure a plugin to enable SMTP authentication. We recommend the SMTP Auth Plugin. To install and configure the plugin:

1. *Install these CPAN Modules*:
   - `Net::SMTP`
   - `Authen::SASL`
   - `Net::SMTP::SSL`
   - `IO::Socket::SSL`
   - `Net::SSLeay`
2. Install *SMTP Auth Plugin*.
   (a) Open an SSH session to your account.
   (b) Switch the Movable Type directory. Enter `cd ~/webapps/mt` where *mt* is the name of the Movable Type application as it appears on the control panel, and press `Enter`.
   (c) Download *SMTP Auth Plugin*. Enter `wget http://tweezersedge.com/gems/SMTPAuth-1.0.zip` and press `Enter`. An archive containing the plugin is created in the current directory.
   (d) Unzip the archive. Enter `unzip SMTPAuth-1.0.zip` and press `Enter`.
   (e) Copy the static files. Enter `cp -r SMTPAuth-1.0/mt-static/ .` and press `Enter`.
   (f) Copy the plugin files. Enter `cp -r SMTPAuth-1.0/plugins/ .` and press `Enter`.
   (g) Remove the installation files. Enter `rm -r SMTPAuth-1.0 SMTPAuth-1.0.zip` and press `Enter`.
3. Configure the plugin.
   (a) Open `~/webapps/mt/mt-config.cgi` in a text editor.
   (b) Edit or add these directives if they do not already exist:

```
MailTransfer smtpauth
EmailAddressMain <address>
EmailReplyTo 1
```

   where <*address*> is an email address registered on the WebFaction control panel.
   (c) Save and close the file.
   (d) Log in to the Movable Type *User Dashboard*.
   (e) Next to *User Dashboard*, click the navigation icon and then click *System Overview*. The *System Overview* page appears.
   (f) Click *Tools*. The *Tools* menu expands.

(g) Click *Plugins*. The *System Plugin Settings* page appears.

(h) Click *SMTP Auth 1.0*. The *SMTP Auth 1.0* plugin pane expands.

(i) Click *Settings*. The settings form appears.

(j) In the *SMTP Server*, *Account Name*, *Password*, *SMTP Connection*, and *Use secure connection (SSL)* fields, enter the SMTP server connection details. For the WebFaction SMTP server, use these values:

SMTP Server `smtp.webfaction.com`

Account Name A mailbox name.

Password The mailbox password.

Use secure connection (SSL) Yes

SMTP Connection `465`

(k) Click the *Save Changes* button.

CHAPTER

FOURTEEN

NODE.JS

Node.js (also known as "Node") is a platform for developing and running applications with JavaScript. You can install a Node.js application with the control panel.

Note: If you are working with a Node.js application during an SSH session, then you may find it easier to add the application's `bin` directory to your `PATH` environment variable. Otherwise, you must use full paths to `node`, `npm`, and other executables for the application.

To add a Node.js application's `bin` directory to your `PATH` environment variable:

1. Switch to your Node.js application's directory. Enter `cd $HOME/webapps/nodejs_app/`, where *nodejs_app* is the name of your Node.js application as it appears on the control panel, and press `Enter`.
2. Add the application's `bin` directory to your `PATH` environment variable. Enter `export PATH=$PWD/bin/:$PATH` and press `Enter`.

For the remainder of the SSH session, you can run your application's executables without entering a full path.

14.1 Starting and Stopping Node.js

To start or stop a Node.js application:

1. *Open an SSH session to your account*.
2. Run the `start` or `stop` script:
   - To start the Node.js application, enter `$HOME/nodejs_app/bin/start`, where *nodejs_app* is the name of your Node.js application as it appears on the control panel, and press `Enter`.
   - To stop the Node.js application, enter `$HOME/nodejs_app/bin/stop`, where *nodejs_app* is the name of your Node.js application as it appears on the control panel, and press `Enter`.

Note: Node.js applications are created with a cron job that starts the Node.js application every twenty minutes, if it's not already running. If you do not want the Node.js application to restarted eventually, then you must modify your crontab. For more about cron, see *Scheduling Tasks with Cron*.

14.2 Installing Packages with `npm`

`npm` is the Node.js package management tool. To install a package with `npm`:

1. *Open an SSH session to your account*.
2. Switch to your Node.js application's directory. Enter `cd $HOME/webapps/nodejs_app/` , where *nodejs_app* is the name of your Node.js application as it appears on the control panel, and press `Enter` .
3. Add the application's `bin` directory to your `PATH` environment variable. Enter `export PATH=$PWD/bin/:$PATH` and press `Enter` .
4. Enter `npm install -g package` , where *package* is the name of the package to be installed, and press `Enter` .

Note: `npm` can install packages for the current directory (in `./node_modules/`), or for the entire Node.js application (in `~/webapps/nodejs_app/lib/node_modules/`). This step assumes packages should be made available to all Node.js programs used with the current application. To install the module for the current directory only, omit the `-g` flag before running the command.

The package is installed.

14.3 Upgrading Node.js

You can use n, a Node.js version manager, to upgrade a Node.js application to a newer version of Node.js. To upgrade a Node.js application to the latest stable version of Node.js:

1. *Open an SSH session to your account*.
2. Switch to your Node.js application's directory. Enter `cd $HOME/webapps/nodejs_app/` , where *nodejs_app* is the name of your Node.js application as it appears on the control panel, and press `Enter` .
3. Add the application's `bin` directory to your `PATH` environment variable. Enter `export PATH=$PWD/bin/:$PATH` and press `Enter` .
4. Enter `npm cache clean -f` and press `Enter` .
5. Install the Node.js version manager. Enter `npm install -g n` and press `Enter` .
6. Upgrade Node.js. Enter `N_PREFIX=$PWD n stable` and press `Enter` .

   Alternatively, you can substitute `stable` with a version number, like `4.2.2` , to install a specific version of Node.js.

The version of Node.js you selected is installed. To confirm your current version of Node.js, enter `node --version` and press `Enter` .

CHAPTER

FIFTEEN

PERL

The Perl interpreter is installed on all WebFaction servers.

On CentOS 6 servers (`web300` and greater), Perl version 5.10.1 is installed. You can run the Perl interpreter as `perl` or `perl5.10.1`.

On CentOS 5 servers, Perl version 5.8.8 is installed. You can run the Perl interpreter as `perl` or `perl5.8.8`.

15.1 Installing cpanminus

We recommend using cpanminus to install CPAN modules. To install (or upgrade) cpanminus:

1. *Open an SSH session to your account*.
2. Switch to your `~/bin` directory. Enter `cd $HOME/bin` and press `Enter`.
3. Download `cpanm`. Enter `wget http://xrl.us/cpanm` and press `Enter`.
4. Make `cpanm` executable. Enter `chmod +x cpanm` and press `Enter`.

cpanminus is now installed.

15.2 Installing a Perl Module with `cpanm`

To install a package with `cpanm`:

1. If you have not already done so, *install cpanminus*.
2. *Open an SSH session to your account*.
3. Enter `cpanm module`, *module* is the name of the module to install, and press `Enter`. For example, to install Locale::gettext, enter `cpanm Locale::gettext` and press `Enter`.

The package is installed in `~/perl5`.

15.3 Using Local Perl Modules in a Command-Line Script

To use a Perl module, the module's path must appear in the `@INC` array, since Perl finds modules by searching the paths in the `@INC` array. When running command-line scripts, there are three ways to add paths to Perl's module search:

1. Use the Perl interpreter's `-I` option to specify an additional search path.

   You can add one or more paths with one or more `-I` options followed by a path. Generally, run `perl -I path script`, where *path* is the path to a directory that contains a Perl module and *script* is the path to the Perl script itself.

   For example, to use a module installed in your home directory with `cpanm`, enter `perl -I $HOME/perl5/lib/perl5/ script` and press `Enter`.

2. Use the `use lib` statement to specify an additional search path.

   You can add one or more paths with `use lib` statements. Insert `use lib "path";` in your script on a line before other modules are used.

   For example, to use a module installed in your home directory with `cpanm`, insert `use lib "/home/username/perl5/lib/perl5/";` in your script, where *username* is your username.

3. Set the `PERL5LIB` environment variable.

   You can add one or more paths by setting the `PERL5LIB` environment variable in your shell. This is useful in cases where an executable Perl script is invoked directly (for example, running `./myscript.pl` instead of `perl ./myscript.pl`), since the `-I` option is unavailable.

   - To add to `PERL5LIB` at run time, set the variable before invoking Perl, like this: enter `PERL5LIB=$PERL5LIB:path perl script` and press `Enter`.
   - To add to `PERL5LIB` for the duration a session, export the variable, like this: enter `export PERL5LIB=$PERL5LIB:path` and press `Enter`.
   - To add to `PERL5LIB` every time you login, add the export command to your `.bashrc` file, like this: enter `echo "export PERL5LIB=$PERL5LIB:path" >> $HOME/.bashrc` and press `Enter`.

   For example, to run a script using a module installed in your home directory with `cpanm`, enter `PERL5LIB=$PERL5LIB:$HOME/perl5/lib/perl5/ perl script` and press `Enter`.

15.4 Using Local Perl Modules in a CGI Script

To use a Perl module, the module's path must appear in the `@INC` array, since Perl finds modules by searching the paths in the `@INC` array. When running as a CGI script, use the `use lib` statement to specify an additional search path.

You can add one or more paths with `use lib` statements. Insert `use lib "path";` in your script on a line before other modules are used, where *path* is the path to a directory that contains a Perl module.

For example, to use a module installed in your home directory with `cpanm`, insert `use lib "/home/username/perl5/lib/perl5/";` in your script, where *username* is your username.

15.5 Using Alternate Perl Versions with Perlbrew

Perlbrew is a tool that lets you install and run isolated versions of Perl from your home directory. You can use Perlbrew to install old, new, or upcoming versions of Perl.

15.5.1 Installing Perlbrew

To install Perlbrew:

1. *Open an SSH session to your account*.
2. Download and run the Perlbrew installation script. Enter `curl -L http://install.perlbrew.pl | bash` and press `Enter`.
3. Follow the post-install instructions that appear in your terminal. You may need to modify a configuration file, such as `.bash_profile`, before continuing.
4. Start a new SSH session.
5. Confirm that Perlbrew has been installed. Enter `perlbrew --version` and press `Enter`.

If no errors appear, Perlbrew has been installed.

15.5.2 Using Perlbrew

After you have installed Perlbrew, you can install and select versions of Perl. To install and switch to a version of Perl:

1. *Open an SSH session to your account*.
2. Install a version of Perl. Enter `perlbrew install --notest version`, where *version* is a Perl version specifier, and press `Enter`.

   For example, to install the latest stable version of Perl, enter `perlbrew install --notest stable` and press `Enter`. Or to install a specific version, such as Perl 5.22.0, enter `perlbrew install --notest 5.22.0` and press `Enter`.
3. Wait for installation to complete. The installation process may take several minutes.
4. Switch to the installed version of Perl. Enter `perlbrew switch version`, where *version* is the version number you installed previously, and press `Enter`.

The Perl version you selected is active. To verify your Perl selection, run `perl --version`. For more information on using Perlbrew, run `perlbrew help`.

CHAPTER

SIXTEEN

PHP

Several application types use or may use PHP, including Static/CGI/PHP applications, WordPress, and Drupal.

16.1 PHP on the Command Line

The PHP command-line interface, `php`, is installed on all WebFaction servers. Several versions of the command-line interface are available. You can specify a version of PHP by running `phpXY`, where *XY* is the major and minor version number. For example, to use the command-line interface for PHP 5.6, run `php56`.

The default version (the version that runs when you run the `php` command) and available versions vary by server:

Operating system	Default PHP version	PHP 5.2	PHP 5.3	PHP 5.4	PHP 5.5	PHP 5.6	PHP 7.0	PHP 7.1	PHP 7.2
CentOS 5	5.2	Yes	Yes	Yes	Yes	Yes			
CentOS 6	5.2	Yes	Yes	Yes	Yes	Yes	Yes		
CentOS 7	5.6	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

To find the default version of PHP on your server, run `php --version`. To find your server's operating system, see *Finding Your Server's Operating System*.

16.2 Configuring PHP

You may configure PHP settings (also known as *directives*) using a file with a `.ini` extension located in your PHP-based application's directory. Typically, this file is called `php.ini` but any and all files named with an ending `.ini` are scanned for PHP settings.

Note: List of php.ini directives provides a table of possible settings. Only settings listed as `PHP_INI_PERDIR` or `PHP_INI_ALL` in the *Changeable* column may be changed. Settings listed as `PHP_INI_SYSTEM` settings will not work with the following directions.

Note: If you're using the optional FastCGI deployment method, please see *Using FastCGI* for additional configuration details.

To configure a PHP-based application:

1. Create `php.ini` if it does not already exist.
   (a) *Open an SSH session to your account*.

(b) Enter `touch ~/webapps/application/php.ini` where *application* is the name of the application as it appears in the control panel, and press `Enter`.

2. Edit `php.ini`.

   (a) Open `~/webapps/application/php.ini` in a text editor.

   (b) For each setting, add a new line containing `name = value`, where *name* is the name of the directive and *value* is the new value for the setting.

   (c) Save and close the file.

16.3 Upgrade to another PHP version

To upgrade to another version of PHP:

1. *Open an SSH session to your account*.
2. Switch to the PHP-based application's directory. Enter `cd $HOME/webapps/app/`, where *app* is the name of your PHP-based application, and press `Enter`.
3. Create a `.htaccess` file, if it does not already exist. Type `touch .htaccess` and press `Enter`.
4. Open the `.htaccess` file in a text editor.
5. Add the following lines to switch to PHP 7.2:

```
<FilesMatch \.php$>
    SetHandler php72-cgi
</FilesMatch>
```

   Alternatively, you may substitute `php72-cgi` with `php71-cgi` for PHP 7.1 or `php70-cgi` for PHP 7.0 or `php56-cgi` for PHP 5.6.

6. Save and close the file.

Note: If your control panel application is of the *Static/CGI/PHP* type then this change will not be reflected in the version shown in the control panel. It will continue showing the version you've choosen when you first created the application.

16.4 Configuring an Upload Limit

To configure an upload limit for PHP-based applications:

1. Create `php.ini` if it does not already exist.

   (a) *Open an SSH session to your account*.

   (b) Enter `touch ~/webapps/application/php.ini` where *application* is the name of the application as it appears in the control panel, and press `Enter`.

2. Edit `php.ini`.

   (a) Open `~/webapps/application/php.ini` in a text editor.

   (b) Add these lines to the end of the file:

```
upload_max_filesize = XM
post_max_size = XM
```

where `X` is the maximum number of megabytes you want to allow to be uploaded.

(c) Save and close the file.

16.5 Configuring `DOCUMENT_ROOT`

The `DOCUMENT_ROOT` setting has two possible default values:

- For websites with one PHP-based application, `DOCUMENT_ROOT` is set to `/home/username/webapps/application`, where *username* is your username and *application* is the name of the PHP-based application.
- For websites with two or more PHP-based applications, `DOCUMENT_ROOT` is set to `/home/username/webapps/_`.

To change the value of `DOCUMENT_ROOT`:

1. Create `set_document_root.php`.
   (a) *Open an SSH session to your account*.
   (b) Enter `touch ~/webapps/application/set_document_root.php`, where *application* is the name of the application as it appears in the control panel, and press `Enter`.
   (c) Open `~/webapps/application/set_document_root.php` in a text editor.
   (d) Insert these lines:

```
<?php
  $_SERVER['DOCUMENT_ROOT'] = '/home/username/webapps/example_application';
?>
```

   where the string following the equal sign is the new `DOCUMENT_ROOT` path.
   (e) Save and close the file.
2. Create `php.ini` if it does not already exist.
   (a) *Open an SSH session to your account*.
   (b) Enter `touch ~/webapps/application/php.ini` where *application* is the name of the application as it appears in the control panel, and press `Enter`.
3. Edit `php.ini`.
   (a) Open `~/webapps/application/php.ini` in a text editor.
   (b) Add a new line containing `auto_prepend_file = "/home/username/webapps/application/set_document_root.php"` where *application* is the name of the application as it appears in the control panel.
   (c) Save and close the file.

16.6 Using FastCGI

Note: FastCGI is available on servers `web120` and greater or `dweb61` and greater. If you have a plan on a server that does not support FastCGI, you may request a server migration. Please see *Migrating Servers* for more details.

Warning: FastCGI deployments do not use `php.ini` configuration files. If you're switching an existing PHP application to FastCGI and that application has a `php.ini` file, please rename the file to `.user.ini`.

You can use FastCGI as an alternative deployment method for PHP scripts. Although it consumes your account's memory, it has better performance characteristics for some applications. To enable FastCGI:

1. Open `$HOME/webapps/application/.htaccess` in a text editor, where *application* is the name of the PHP-based application.
2. Insert the following lines:

```
<FilesMatch \.php$>
    SetHandler <handler>
</FilesMatch>
```

   where `<handler>` is the FastCGI handler you want to use. Acceptable values are:

   - `php54-fcgi`
   - `php55-fcgi`
   - `php56-fcgi`
   - `php70-fcgi`

   Additionally, you can run up to six FastCGI processes by appending an integer 2-6. For example, to use FastCGI with PHP 5.4 and 3 processes, enter `php54-fcgi3` in place of `<handler>`.
3. Save and close the file.

Subsequent requests will use FastCGI with the PHP interpreter and the number of processes you selected.

16.7 Installing a Custom PHP Package

While WebFaction provides installers for some of the most popular PHP web applications, we cannot provide installers for all. Many packages use a familiar installation process, however. For most PHP-based packages, follow this procedure:

1. *Create a website with a new Static/CGI/PHP application*.
2. Download the software package and review its `README` and installation files. Review other documentation, like the software's website, as needed.
3. If applicable, *create a database* for your application. Make a note of the database name, username, and password.
4. Upload the package to the Static/CGI/PHP application, `~/webapps/app`, where *app* is the name of the application as it appears on the control panel.

5. Start the package's installation process. Typically, the application will have a specific URL for you to open. For example, MediaWiki uses `domain_path/config/index.php`, where *domain_path* is the domain and URL path, to start the installation process. See your software package's documentation for details. Follow the on-screen instructions.

   Some packages may not provide a web-based install procedure. For such applications, review and follow the provided documentation.

   Note: Many packages include a step to delete installation files. If it applies, don't skip this step! Failing to do so may allow others to gain control of your application and data.

16.8 Installing and Using PEAR Packages

PEAR, or the PHP Extension and Application Repository, is a collection of commonly-used PHP packages. You can use the PEAR package manager to install such packages in your home directory.

16.8.1 Installing the PEAR Package Manager

Before you can use PEAR to install packages, you must create a PEAR installation in your home directory:

1. *Open an SSH session to your account*.
2. Create a directory for PEAR. Enter `mkdir $HOME/pear` and press `Enter`.
3. Create a PEAR configuration file. Enter `pearXY config-create $HOME $HOME/.pearrc`, where *XY* is the major and minor version of PHP you want to use, and press `Enter`.

   For example, to use PHP 5.6, enter `pear56 config-create $HOME $HOME/.pearrc` and press `Enter`.
4. Install PEAR. Enter `pearXY install -o PEAR` and press `Enter`.

16.8.2 Installing PEAR Packages

1. *Open an SSH session to your account*.
2. Add the PEAR installation directory to your `PATH`. Enter `export PATH=$HOME/pear:$PATH` and press `Enter`.
3. Install a package. Enter `pear install package`, where *package* is the name of a PEAR package, and press `Enter`.

   See also:

   For a complete list available packages, see the PEAR List Packages page.

16.8.3 Using PEAR Packages

Before you can `require` or `include` an installed PEAR package in a PHP script, you must add the `~/pear/php` directory to your `include_path`.

To add add the `~/pear/php` directory to the `include_path` for a single script, include this line before any `require` or `include` statements:

```
set_include_path(get_include_path() . PATH_SEPARATOR . '/home/<username>/pear/php');
```

where `<username>` is your username.

Alternatively, you can set the `include_path` to `.:/home/username/pear/php`, where *username* is your username, in a *PHP configuration file*.

16.9 Serving HTML Files as PHP Scripts

By default, WebFaction Static/CGI/PHP and PHP-based applications treat files ending in `.html` as plain HTML to serve those pages as quickly as possible. You can override that default, but the process varies from server to server.

For servers Web120 and higher, edit or create a `.htaccess` file in the directory you want to serve HTML files as PHP scripts to contain this line:

```
AddHandler php56-cgi .html
```

For servers Web119 and lower, edit or create a `.htaccess` file in the directory you want to serve HTML files as PHP scripts to contain these lines:

```
<IfModule !php5_module>
    RewriteEngine on
    RewriteBase /
    RewriteCond %{REQUEST_URI} \.html$
    RewriteRule (.*) http://127.0.0.1:2480/$1 [P]
</IfModule>
AddHandler php5-script html
```

Alternatively, to serve files ending in `.php` as if they were `.html` files, edit or create a `.htaccess` file in the directory you want to serve the PHP files with the `.html` extension to contain these lines:

```
<IfModule !php5_module>
    RewriteCond %{REQUEST_URI} \.html$
    RewriteRule (.*) http://127.0.0.1:2480/$1 [P]
</IfModule>

RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_URI} \.html$
RewriteRule (.*).html $1.php
```

16.10 Sending Mail from a PHP Script

To send mail from a PHP script, use the built-in mail() function to send an email. For example, this script sends a test message:

```
<?php
$to = 'example@example.com';
$subject = 'Test Message';
$message = 'Hello, World!';
$from = 'sender@domain.com';
$headers = 'From: '.$from.'\r\n';
```

```
$sender = '-f '.$from;
mail($to, $subject, $message, $headers, $sender);
?>
```

Alternatively, the script may connect to and send mail from an SMTP server. There are several packages available which simplify this method of sending mail:

- Swift Mailer
- PEAR's Mail package
- Zend Framework's ZendMail

For example, this PHP script uses the PEAR Mail package to send an email:

```
<?php
require_once "Mail.php";

$from_addr = "Me <my_email_address@somedomain.example>";
$to = "Team <team@somedomain.example>";
$subject =  "Hello!";
$body = "Dear Team, here is my message text.";

$headers = array ("From" => $from_addr,
                  "To" => $to,
                  "Subject" => $subject);
$smtp = Mail::factory("smtp", array ('host' => "smtp.webfaction.com",
                                     'auth' => true,
                                     'username' => "my_mailbox_name",
                                     'password' => "password1"));

$mail = $smtp->send($to, $headers, $body);
?>
```

For this script to work, the values for *from_addr*, *my_mailbox_name*, and *password1* would be changed to an outgoing address, a valid WebFaction mailbox name, and a valid mailbox password, respectively.

16.11 Serving an Application from a Subdirectory

Some PHP frameworks call for serving a site from a subdirectory. See *Serving a Static/CGI/PHP application from a Subdirectory* to learn about using a *Symbolic link* application to serve a subdirectory.

16.12 Setting PHP's Default Time Zone

To set the default time zone for a PHP applications' date and time functions, *modify the application's PHP settings* such that `date.timezone` is your preferred time zone. For example, to set the default to the United States Eastern time zone, add `date.timezone = America/New_York` to your `php.ini`. For valid time zones, see the official PHP documentation's List of Supported Timezones.

16.13 Troubleshooting

16.13.1 Error: *500 Internal Server Error*

See also:

For additional troubleshooting recommendations, please see the general CGI *500 Internal Server Error* documentation.

Error: `Premature end of script headers: php54.cgi`

A *500 Internal Server Error* in the browser and an error like this in the `$HOME/logs/frontend/error_website_php.log` file:

```
[Thu Mar 22 18:38:49 2012] [error] [client 123.456.78.9 ] Premature end of script headers: php54.cgi
```

may be a failed attempt to use the apache_request_headers function.

There are two common solutions to this problem:

1. Disable the `apache_request_headers` function. Many software packages gracefully work around the disabled function. To disable the function, modify the application's `php.ini` file to contain:

   ```
   disable_functions = "apache_request_headers"
   ```

   See *Configuring PHP* for detailed instructions about creating and modifying `php.ini`.

2. For servers `web120` and greater or `dweb61` and greater, switch to the FastCGI PHP deployment method. See *Using FastCGI* for detailed instructions on how to enable FastCGI.

16.13.2 Strange Characters in Output

Strange hexadecimal characters may appear prepended to expected output with software which forces the `HTTP/1.1` header.

To resolve this issue, add these directives to the application's `.htaccess` file to force the correct headers:

```
SetEnv force-response-1.0 1
SetEnv downgrade-1.0 1
```

Alternatively, you may modify the software to use the correct headers. For example, this problem can be solved for Drupal by modifying the `drupal_set_header` function in `includes/common.inc`. Add this line to the end of the function definition:

```
$header = str_replace('HTTP/1.1', $_SERVER["SERVER_PROTOCOL"], $header);
```

The line replaces the `HTTP/1.1` header with the correct header based on PHP's server and execution environment information.

CHAPTER

SEVENTEEN

PRIVATE DATABASE INSTANCES

Private MySQL and PostgreSQL instances are an alternative to your server's *shared databases*. Private database instances are useful for taking greater control over how your databases are configured, or to resolve the "bad neighbor problem," where one user consumes too much of a shared database's resources.

Private database instances rely upon the server's globally-installed MySQL or PostgreSQL binaries, so WebFaction continues to apply security patches for you. Private database instances run under your user account, so the instance process counts toward your account's memory allotment, and the instance's files count toward your account's disk space allotment.

17.1 Private MySQL Instances

Private MySQL instances run under your user account, with configuration and data files stored in an application directory, `~/webapps/instance`, where *instance* is the name of the application.

The root user's password is automatically generated. The password can be found in the control panel, in the application's *Extra info* field.

On CentOS 7 servers, instances run MySQL 5.6. On CentOS 6 servers (`web300` and greater), instances run MySQL 5.5. On CentOS 5 servers, instances run MySQL 5.0.

Every instance comes with two *cron jobs*. One attempts to start the database every ten minutes, if it is not already running. The other attempts to create a dump of your database once per day.

A MySQL instance's error log file is `~/logs/user/error_instance.log`.

17.1.1 Installing a Private MySQL Instance

To install a private MySQL instance:

1. Log in to the control panel.
2. Click *Domains / websites* → *Applications*. The list of applications appears.
3. Click the *Add new application* button. The *Create a new application* form appears.
4. In the *Name* field, enter a name for the application.
5. In the *App Category* menu, click to select *MySQL*.
6. In the *App Type* menu, click to select *MySQL private instance*.
7. If applicable, in the *Machine* menu, click to select a web server.
8. Click the *Save* button.

The application is installed and added to the list of applications.

17.1.2 Starting, Restarting, and Stopping a Private MySQL Instance

To manually start, restart, or stop a private MySQL instance, *open an SSH session to your account*, then enter one of these management commands:

- Start: `$HOME/webapps/instance/bin/start`
- Restart: `$HOME/webapps/instance/bin/stop && $HOME/webapps/instance/bin/start`
- Stop: `$HOME/webapps/instance/bin/stop`

where *instance* is the name of the private database instance application as it appears on the control panel, and press `Enter`.

17.1.3 Creating Private MySQL Databases and Users

To use your database, you will likely require databases and users. To create a database and user with privileges on that database:

1. *Open an SSH session to your account*.
2. Start a MySQL session. Enter `mysql -P port -u root -p --protocol=tcp`, where *port* is the instance's port number, and press `Enter`. A password prompt appears.

   Note: To find the port number:

   (a) Log in to the control panel

   (b) Click *Domains / websites* → *Applications*. The list of applications appears.

   (c) Click the name of the application.

   The port number appears in the *Port* section.

   ---

3. Enter the password for the root user (as generated on the control panel) and press `Enter`. A prompt appears.
4. Create a database. Enter `CREATE DATABASE database_name;`, where *database_name* is the name of the database, and press `Enter`.
5. Create a user. Enter `CREATE USER 'username' IDENTIFIED BY 'pass';`, where *username* is the new username and *pass* is the user's password, and press `Enter`.

   See also:

   See *Strengthening Passwords* for important information about choosing passwords.
6. Switch to the new database. Enter `USE database_name;` and press `Enter`.
7. Grant the user privileges on the database. Enter `GRANT ALL ON database_name.* TO 'username';` and press `Enter`.
8. Quit the session. Enter `\q` and press `Enter`.

The user can now *connect to the database* and run queries.

See also:

See the official MySQL documentation for more information about creating and managing users and databases.

17.1.4 Importing Data from a Shared MySQL Database

To move your data from a shared MySQL database to a private database instance:

1. *Dump the contents of your existing shared database to a file.*
2. *Open an SSH session to your account*.
3. Enter
   `mysql -P port --protocol=tcp -u database_user -p -D database_name < dump_file`,
   where:
   - *port* is the instance's port number,
   - *database_user* is a valid database username,
   - *database_name* is a database that the user has permissions to use, and
   - *dump_file* is the path the dump containing the contents of your shared database,

   and press `Enter`.

The contents of the dump file are loaded into the database.

17.1.5 Importing and Exporting Data from a Private MySQL Database

You can use command-line tools to export (or *dump*) the contents of a MySQL database to a file, or to import the contents of such a file to a MySQL database.

Note: To minimize load and preserve performance on your server, please use `ionice` when importing or exporting databases larger than 500MB.

To use `ionice`, prefix your database commands with `ionice -c2 -n6`. For example, replace `mysqldump` with `ionice -c2 -n6 mysqldump`.

Exporting

To export the contents of a private MySQL database:

1. *Open an SSH session to your account*.
2. Enter
   `mysqldump --port=port --protocol=tcp --user database_user database_name --password > d`
   - *port* is the instance's port number,
   - *database_user* is a valid database username,
   - *database_name* is a database that the user has permissions to read, and
   - *dump_file* is the path to the dump file to be created,

   and press `Enter`.
3. When prompted, enter your database password.

Your dump file is created at the path specified.

Importing

To import a dump file into a private MySQL database:

1. *Open an SSH session to your account*.
2. Enter
   `mysql --port=port --protocol=tcp --user database_user -D database_name --password < du`
   where:
   - *port* is the instance's port number,
   - *database_user* is a valid database username,
   - *database_name* is a database that the user has permissions to use, and
   - *dump_file* is the path to the dump file,

   and press `Enter`.

   Note: If you're importing a dump file encoded with a character set other than `utf8` (for example, `latin1`), then you must invoke `mysql` with the `--default_character_set encoding` option, where *encoding* is the name of the dump file's encoding.

   Thus, the complete command for importing a non-`utf8` dump file is:

   `mysql --port=port --protocol=tcp --user database_user -D database_name --password --de`

   ---
3. When prompted, enter your database password.

The contents of the dump file are loaded into the database.

17.2 Private PostgreSQL Instances

Private PostgreSQL instances run under your user account, with configuration and data files stored in an application directory, `~/webapps/instance`, where *instance* is the name of the application.

The instance's superuser is your WebFaction account username. The superuser's password can be found in the control panel, in the application's *Extra info* field.

On CentOS 7 servers, instances run PostgreSQL 9.4. On CentOS 6 servers (`web300` and greater), instances run PostgreSQL 9.1. On CentOS 5 servers, instances run PostgreSQL 8.3.

Every instance comes with two *cron jobs*. One attempts to start the database every ten minutes, if it is not already running. The other attempts to create a dump of your database once per day.

An instance's log files can be found in the `~/webapps/instance/data/pg_log/` directory.

17.2.1 Installing a Private PostgreSQL Instance

To install a private PostgreSQL instance:

1. Log in to the control panel.
2. Click *Domains / websites* → *Applications*. The list of applications appears.
3. Click the *Add new application* button. The *Create a new application* form appears.
4. In the *Name* field, enter a name for the application.

5. In the *App Category* menu, click to select *PostgreSQL*.
6. In the *App Type* menu, click to select *PostgreSQL private instance*.
7. If applicable, in the *Machine* menu, click to select a web server.
8. Click the *Save* button.

The application is installed and added to the list of applications.

17.2.2 Starting, Restarting, and Stopping a Private PostgreSQL Instance

To manually start, restart, or stop a private PostgreSQL instance, *open an SSH session to your account*, then enter one of these management commands:

- Start: `$HOME/webapps/instance/bin/start`
- Restart: `$HOME/webapps/instance/bin/stop && $HOME/webapps/instance/bin/start`
- Stop: `$HOME/webapps/instance/bin/stop`

where *instance* is the name of the private database instance application as it appears on the control panel, and press `Enter`.

17.2.3 Creating Private PostgreSQL Databases and Users

To use your database, you will likely require databases and users. To create a database and user with privileges on that database:

1. *Open an SSH session to your account*.
2. Start a PostgreSQL interactive terminal. Enter `psql -h localhost -p port postgres`, where *port* is the instance's port number and press `Enter`. A terminal prompt appears.
3. Create a database. Enter `CREATE DATABASE database_name;`, where *database_name* is the name of the database, and press `Enter`.
4. Create a user. Enter `CREATE USER username WITH PASSWORD 'pass';`, where *username* is the new username and *pass* is the user's password, and press `Enter`.

   See also:

   See *Strengthening Passwords* for important information about choosing passwords.
5. Grant the user privileges on the database. Enter `GRANT ALL PRIVILEGES ON DATABASE database_name to username;` and press `Enter`.
6. Quit the interactive terminal. Enter `\q` and press `Enter`.

The user can now *connect to the database* and run queries.

See also:

See the official PostgreSQL documentation for more information about creating and managing users and databases.

17.2.4 Importing Data from a Shared PostgreSQL Database

To move your data from a shared PostgreSQL database to a private database instance:

1. *Dump the contents of your existing shared database to a file.*

2. *Open an SSH session to your account*.
3. Enter `psql -h localhost -p port -U database_user database_name < dump_file`, where:
   - *port* is the instance's port number,
   - *database_user* is a valid database username,
   - *database_name* is a database that the user has permissions to use, and
   - *dump_file* is the path the dump containing the contents of your shared database,

   and press `Enter`.

The contents of the dump file are loaded into the database.

17.2.5 Importing and Exporting Data from a Private PostgreSQL Database

You can use command-line tools to export (or *dump*) the contents of a PostgreSQL database to a file, or to import the contents of such a file to a PostgreSQL database.

Note: To minimize load and preserve performance on your server, please use `ionice` when importing or exporting databases larger than 500MB.

To use `ionice`, prefix your database commands with `ionice -c2 -n6`. For example, replace `pg_dump` with `ionice -c2 -n6 pg_dump`.

Exporting

To export the contents of a private PostgreSQL database:

1. *Open an SSH session to your account*.
2. Enter
   `pg_dump --host localhost --port port --username database_user --file dump_file databas`
   - *port* is the instance's port number,
   - *database_user* is a valid database username,
   - *database_name* is a database that the user has permissions to read, and
   - *dump_file* is the path to the dump file to be created,

   and press `Enter`.

Your dump file is created at the path specified.

Importing

To import a dump file into a private PostgreSQL database:

1. *Open an SSH session to your account*.
2. Enter
   `psql --host localhost --port port --username database_user database_name < dump_file`,
   where:
   - *port* is the instance's port number,

- *database_user* is a valid database username,
- *database_name* is a database that the user has permissions to use, and
- *dump_file* is the path to the dump file,

and press `Enter`.

Note: If your dump file was created as a PostgreSQL custom dump, then use `pg_restore` instead of `psql` to import the data.

Enter
`pg_restore --host localhost --port port --username database_user --dbname database_nam`
and press `Enter`.

The contents of the dump file are loaded into the database.

17.3 Configuring Applications for Private Database Instances

To connect an application to a private MySQL or PostgreSQL instance, use the following configuration details:

Host `127.0.0.1` (an IP address) or `localhost` (a hostname)

Port Use the port number assigned to your application.

Note: To find the port number:

1. Log in to the control panel
2. Click *Domains / websites* → *Applications*. The list of applications appears.
3. Click the name of the application.

The port number appears in the *Port* section.

Username A database user you've created previously.

Database A database you've created previously.

See also:

- *Creating Private MySQL Databases and Users*
- *Creating Private PostgreSQL Databases and Users*

The following sections provide examples of how to reconfigure applications for use with private database instances.

17.3.1 Configuring Django for a Private Database Instance

To reconfigure a Django application's default project to use a private PostgreSQL instance:

1. If applicable, *import data from a shared PostgreSQL database to a private database instance*.
2. In a text editor, open `~/webapps/application/myproject/myproject/settings.py`, where *application* is the name of the Django application as it appears in the control panel.
3. Find the `DATABASES` dictionary similar to the following:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.',
        'NAME': '<DATABASE>',
        'USER': '<USERNAME>',
        'PASSWORD': '<PASS>',
        'HOST': '<HOST>',
        'PORT': '<PORT_NUMBER>',
    }
}
```

and modify the dictionary such that:

- `django.db.backends.` is replaced by `django.db.backends.postgresql_psycopg2`,
- `<DATABASE>` is the name of a database you've created previously,
- `<USERNAME>` is a database user you've created previously,
- `<PASS>` is the database user's password,
- `<HOST>` is replaced by `127.0.0.1`, and
- `<PORT_NUMBER>` is the port number assigned to your private database instance.

  See also:

  To find the port number:

  (a) Log in to the control panel

  (b) Click *Domains / websites* → *Applications*. The list of applications appears.

  (c) Click the name of the application.

  The port number appears in the *Port* section.

4. Save and close the file.
5. *Restart the Django application*.

The Django application is configured to connect to the private PostgreSQL instance.

17.3.2 Configuring WordPress for a Private Database Instance

To reconfigure a WordPress application to use a private MySQL instance:

1. *Import data from the application's database to a private MySQL instance*.
2. In a text editor, open `~/webapps/application/wp-config.php`, where *application* is the name of the WordPress application.
3. Find a line starting with `define('DB_NAME',` and replace the line with `define('DB_NAME', 'database');`, where *database* is the name of a database you've created previously.
4. Find a line starting with `define('DB_USER',` and replace the line with `define('DB_USER', 'username');`, where *username* is a database user you've created previously.
5. Find a line starting with `define('DB_PASSWORD',` and replace the line with `define('DB_PASSWORD', 'password');`, where *password* is the database user's password.

6. Find a line starting with `define('DB_HOST',` and replace the line with `define('DB_HOST', '127.0.0.1:port');`, where *port* is the port number assigned to your private database instance.
7. Save and close the file.

CHAPTER

EIGHTEEN

PYRAMID

Pyramid is an open source web application framework for Python.

18.1 Deploying an Existing Pyramid Project

To deploy a Pyramid project that has been developed outside of the WebFaction environment:

1. Create a Pyramid application.
   (a) Log in to the control panel.
   (b) Click *Domains / websites* → *Applications*. The list of applications appears.
   (c) Click the *Add new application* button. The *Create a new application* form appears.
   (d) In the *Name* field, enter a name for the application.
   (e) In the *App category* menu, click to select *Pyramid*.
   (f) In the *App type* menu, click to select the version of Pyramid to use.
   (g) If applicable, in the *Machine* menu, click to select a web server.
   (h) Click the *Save* button. The application is installed and appears in the list of applications.
   (i) Make a note of the port number for the application. This number is required later in the installation process. The application's port number appears next to the application's type.
2. Install the project.
   (a) *Open an SSH session to your account*.
   (b) Switch to the application directory. Enter `cd $HOME/webapps/application` and press `Enter`.
   (c) Activate the Pyramid application's environment. Enter `source bin/activate` and press `Enter`.
   (d) Install the project. Enter `easy_install -U myproject`, where *myproject* is the name of the project package, and press `Enter`.
   (e) Deactivate the Pyramid application's environment. Enter `deactivate` and press `Enter`.
3. Update `development.ini`.
   (a) Open `$HOME/webapps/application/project/development.ini` in a text editor.
   (b) In the `[server:main]` section, replace the line starting with `port =` with `port = number` where *number* is the port number provided for the application in the WebFaction control panel.
   (c) Save and close the file.

(d) Switch to the application directory. Enter `cd $HOME/webapps/application` and press `Enter`.

(e) Create a symlink to `development.ini`. Enter `ln -fs ./project/development.ini` and press `Enter`.

See also:

If you're deploying an app to a non-root URL path (like `example.com/myapp`), see *Serving Pyramid with a URL Prefix*.

4. *Restart the Pyramid application*.

To make the Pyramid application available on the Web, *add the Pyramid application to a website record*.

18.2 Serving Pyramid with a URL Prefix

To serve a Pyramid application at a non-root URL path:

1. Open `$HOME/webapps/application/myapp/development.ini` (or `production.ini`, if your application uses it) in a text editor, where *application* is the name of the Pyramid application.
2. At the end of the `[app:main]` section (by default, immediately before the `[server:main]` section), insert the following lines:

```
filter-with = urlprefix

[filter:urlprefix]
use = egg:PasteDeploy#prefix
prefix = URL_PATH
```

where *URL_PATH* is the URL path where the application is to be served.

For example, to serve the application at the `/myapp` URL path, insert the following lines:

```
filter-with = urlprefix

[filter:urlprefix]
use = egg:PasteDeploy#prefix
prefix = /myapp
```

3. Save and close the file.
4. *Restart the Pyramid application*.

18.3 Starting, Stopping, and Restarting Pyramid

Pyramid applications include a script to start, stop, and restart the application.

18.3.1 Starting

If a Pyramid application is not already running, it is automatically started by a cron job every 20 minutes. To manually start the application:

1. Open an SSH session to your account.
2. Enter `$HOME/webapps/application/bin/start`, where *application* is the name of the application as it appears on the control panel, and press `Enter`.

18.3.2 Stopping

To stop a Pyramid application:

1. Open an SSH session to your account.
2. Enter `$HOME/webapps/application/bin/stop`, where *application* is the name of the application as it appears on the control panel, and press `Enter`.

Note: When the Pyramid application was installed, a cron job was also created to automatically start the application—if it's not already running—every 20 minutes. If you wish to prevent the application from being started automatically, this cron job must be removed. Please see *Scheduling Tasks with Cron* for details on modifying your cron jobs.

18.3.3 Restarting

To restart a Pyramid application:

1. Open an SSH session to your account.
2. Enter `$HOME/webapps/application/bin/restart`, where *application* is the name of the application as it appears on the control panel, and press `Enter`.

CHAPTER

NINETEEN

PYTHON

Python is installed on all WebFaction servers. For your convenience, several different Python versions are available. All servers have Python 2.5 through Python 3.5 installed. CentOS 5 servers also have Python 2.4 installed.

The default version of Python—the version of Python that runs when you run the `python` command—varies by server:

Operating system	Default `python` version	Default `python2` version	Default `python3` version
CentOS 5	2.4	2.4	3.3
CentOS 6	2.6	2.6	3.3
CentOS 7	2.7	2.7	3.5

To identify the default version of Python on your server, run `python -V`.

To run a specific version of Python, run `python`*`X.Y`*, where *X.Y* is the Python version number. For example, to use Python 2.7, enter `python2.7`.

19.1 Python Search Path

To make working with Python and applications easier, the Python search path has been configured with two additions:

- `$HOME/lib/python`*`X.Y`* is added to the Python search path for each version of Python.
- `$HOME/webapps/`*`app`*`/lib/python`*`X.Y`* is added to the Python search path when the current working directory is `$HOME/webapps/`*`app`*`/` or its children, where *app* is the name of an application as it appears in the control panel.

To see the contents of your Python search path, enter `python`*`X.Y`* `-c "import sys; print sys.path"` and press `Enter`.

See also:

`/usr/local/lib/python`*`X.Y`*`/sitecustomize.py` and `/usr/lib/python`*`X.Y`*`/sitecustomize.py` contain the complete implementation of the Python search path modifications.

19.2 Creating a `python` Alias

On WebFaction's CentOS 6 servers (`web300` and greater or `dweb98` and greater), the default `python` is Python 2.6. On all other servers, the default `python` command line program is Python 2.4. You can create an alias

for `python` to refer to another version during interactive SSH sessions. To create a `python` alias:

1. Open an SSH session.
2. Open `$HOME/.bash_profile` in a text editor.
3. On a new line, add `alias python=pythonX.Y` where *X.Y* is the Python version number. For example, to use Python 2.7, add a new line containing `alias python=python2.7`.
4. Save and close the file.
5. Reload `$HOME/.bash_profile`. Enter `source $HOME/.bash_profile` and press `Enter`.

Now all future interactive SSH sessions will use the specified Python version at the command line.

Note: The `python` alias will not be available in shell scripts or other situations where the alias is not defined. In such cases, the explicit Python version should be called instead (for example, `python2.7`).

19.3 Installing Python Packages

Python packages and modules are typically distributed

- automatically with **easy_install**,
- as a Python Egg,
- as an archive containing a `setup.py` file,
- or as a collection of source (`.py`) files.

You can install Python packages and modules distributed in each of these ways in your WebFaction account.

19.3.1 Installing Packages with easy_install

Many common Python packages can be automatically installed with **easy_install**. **easy_install** will look up information about the module from the Python Package Index (also known as the "cheese shop"), download, and install the module automatically.

To use **easy_install** to install a package to your home directory:

1. Make sure the destination directory exists. Enter `mkdir -p $HOME/lib/pythonX.Y`, where *X.Y* is the Python version, and press `Enter`.
2. Install the package. Enter `easy_install-X.Y package`, where *package* is the name of the package to install, and press `Enter`.

To use **easy_install** to install a package to a specific web application which makes use of the Python interpreter, enter `PYTHONPATH=$HOME/webapps/app_name/lib/pythonX.Y easy_install-X.Y --install-dir=$HOME/webap`
where *app_name* is the name of the web application as it appears in the control panel and press `Enter`.

19.3.2 Installing Eggs

Some packages are distributed as Python Eggs. These are single files which contain an entire Python package and typically have the `.egg` file extension. Eggs are installed using **easy_install**.

To install an egg to your home directory:

1. Make sure the destination directory exists. Enter `mkdir -p $HOME/lib/pythonX.Y`, where *X.Y* is the Python version, and press `Enter`.
2. Install enter `easy_install-X.Y egg_path`, where *egg_path* is the path to the egg to install, and press `Enter`.

To install an egg to a specific web application which makes use of the Python interpreter, enter `PYTHONPATH=$HOME/webapps/app_name/lib/pythonX.Y easy_install-X.Y --install-dir=$HOME/webap` where *app_name* is the name of the web application as it appears in the control panel and press `Enter`.

19.3.3 Installing Packages with `setup.py`

Many packages are distributed with a `setup.py` file created with the help of setuptools.

To install a package with `setup.py` to your home directory:

1. Make sure the destination directory exists. Enter `mkdir -p $HOME/lib/pythonX.Y`, where *X.Y* is the Python version, and press `Enter`.
2. Switch to the directory of `setup.py`. Enter `cd path`, where *path* is the path to the directory which contains `setup.py`, and press `Enter`.
3. Run `setup.py`. Enter `PYTHONPATH=$HOME/lib/pythonX.Y pythonX.Y setup.py install --install-lib=$HOME/lib/pyth` and press `Enter`.

To install a package with `setup.py` to a Python-based application:

1. Switch to the directory of `setup.py`. Enter `cd path`, where *path* is the path to the directory which contains `setup.py`, and press `Enter`.
2. Run `setup.py`. Enter `PYTHONPATH=$HOME/webapps/app_name/lib/pythonX.Y pythonX.Y setup.py install --install-l` where *app_name* is the name of the application and *X.Y* is the Python version, and press `Enter`.

Note: To install a package with `setup.py` and the `--install-lib` and `--install-scripts` options, the `--install-lib` directory must also be in Python's search path.

19.3.4 Installing Packages Manually

To install a Python package manually:

1. Make sure the destination directory exists. Enter `mkdir -p $HOME/lib/pythonX.Y`, where *X.Y* is the Python version, and press `Enter`.
2. Copy the source files to `$HOME/lib/pythonX.Y`. Alternatively, you may symbolically link to those files.

Alternatively, you can install packages directly to a web application which makes use of the Python interpreter by copying files to `$HOME/webapps/app_name/lib/pythonX.Y`, where *app_name* is the name of the web application as it appears in the control panel and press `Enter`.

19.3.5 Installing Packages with Pip

Pip, like easy_install, installs Python packages. Many packages that can be installed with easy_install can be installed with pip.

Before you can use pip to install packages, you must install pip in your home directory. To install pip:

1. Create the destination directory, if it does not already exist: Enter `mkdir -p $HOME/lib/pythonX.Y`, where *X.Y* is the Python version, and press `Enter`.
2. Install pip. Enter `easy_install-X.Y pip`, where *X.Y* is the version of Python you wish to use, and press `Enter`.

To use pip to install packages in your home directory, enter `pipX.Y install --user package`, where *package* is the name of the package, and press `Enter`. To upgrade an existing package, enter `pipX.Y install --user --upgrade package` and press `Enter`.

19.4 Troubleshooting

19.4.1 Fixing `ImportError` Exceptions

Python will raise an `ImportError` exception whenever it cannot find a particular package or module as requested by an `import` statement. This is caused by the named module not appearing in any of the module search path directories. The Python search path, represented in Python as the list `sys.path`, is constructed from Python defaults, the `PYTHONPATH` environment variable, and direct manipulations to `sys.path`.

To see the contents of `sys.path`:

1. Run Python interactively. Enter `pythonX.Y` and press `Enter`.
2. At the prompt, enter `import sys` and press `Enter`.
3. At the prompt, enter `print sys.path` and press `Enter`. The contents of `sys.path` will be displayed on screen.

By default, Python searches in this order:

1. the directory where Python is executed;
2. if applicable, the `$HOME/webapps/app_name/lib/pythonX.Y` directory where *app_name* is the name of an application and *X.Y* is the Python version number;
3. the `$HOME/lib/pythonX.Y` directory;
4. directories specified by `PYTHONPATH`;
5. the Standard Library;
6. built-in components of Python.

There are several possible fixes for this problem:

- add the path of the module to `PYTHONPATH` in your `$HOME/.bash_profile` file,
- add the path to the module to `PYTHONPATH` from the command line before executing the Python program which raises the exception,

- add the path to the module to `sys.path` before the `import` statement which raises the exception within the offending program,
- add the path to the module to `myproject.wsgi` for applications using WSGI,
- or, install offending packages directly to a path where Python will find it.

See also:

For more information on how Python loads modules, please see The Python Tutorial » 6. Modules.

Adding to `PYTHONPATH` from the `$HOME/.bash_profile` File

Adding an entry to the end of the `PYTHONPATH` environment variable is a persistent way to avoid `ImportError` exceptions. Unfortunately, this approach can cause complications while debugging Python programs, since the change to the `PYTHONPATH` environment variable happens silently, before you enter any commands at all. This is especially apparent when using two or more versions of the same module.

To add the path to a Python module to the `PYTHONPATH` environment variable from the `$HOME/.bash_profile` file:

1. Open `$HOME/.bash_profile` with a text editor.
2. Add `export PYTHONPATH="`*`path_to_module`*`:$PYTHONPATH"` to the end of the file.
3. Save and close the file.

Log out and log in for the changes to take effect.

Note: Changes to `$HOME/.bash_profile` only change `sys.path` for Python programs executed from a shell session. This approach would not resolve `ImportError` exceptions associated with web applications and other programs run with the help of cron.

Adding to `PYTHONPATH` from the Command Line

You can add to the `PYTHONPATH` environment variable with each run of a Python program. This method is not persistent, which makes it easier to switch between versions of a particular module. Unfortunately, it requires that you include the `PYTHONPATH` changes on each run of the program. You can also use this method in conjunction with the `.bash_profile` method.

To add the path to a Python module to the `PYTHONPATH` environment variable from the command line, run the program like this:

`PYTHONPATH=`*`path_to_module`*`:$PYTHONPATH python `*`program_name`*

Adding to `sys.path` within Python and WSGI Scripts

You can add to a script's Python module search path programmatically by adding entries to the `sys.path` list. It requires that you make changes to each script (like `.py` files) or WSGI file (like `.wsgi` files) for which you want to change the module search path.

To add to `sys.path` within a script, add the follow statements before the `import` statement which raises the `ImportError` exception:

```
import sys
sys.path.insert(0, "path_to_module")
```

where `path_to_module` is the path you wish to add to the module search path.

An equivalent method is to add to `sys.path` with the `+` operator like this:

```
import sys
sys.path = ["path_to_module1", "path_to_module2"] + sys.path
```

That method is common for for WSGI-based applications, where `sys.path` can be modified in the `myproject.wsgi` file found in `/home/username/webapps/appname/`. Typically, a `myproject.wsgi` file contains a line similar to this:

```
sys.path = ['/home/username/webapps/djangoapp/myproject', '/home/username/webapps/djangoapp/lib/pytho
```

Add additional paths to the list to include them in the script's search path.

Adding to `sys.path` from `httpd.conf`

For web applications running on mod_wsgi, the Python module search path can be amended in the `httpd.conf` file, found in `/home/username/webapps/appname/apache2/conf/`. The file contains a line like this:

```
WSGIDaemonProcess app_name processes=5 python-path=/home/username/webapps/app_name:/home/username/web
```

The `WSGIDaemonProcess` directive accepts a parameter `python-path` and a colon-separated list of directories to set the Python module search path.To add a directory to the search path, add a directory path and (if applicable) a colon after `python-path=`.

Note: If you're using virtualenv, then include the full path to the virtualenv's `lib/pythonX.Y` and `lib/pythonX.Y/site-packages` directories in the `python-path` list.

See also:

See the mod_wsgi documentation on Configuration Directives for details on `WSGIDaemonProcess`'s parameters and how `python-path` is handled.

Moving or Installing Packages Directly to a Path in `sys.path`

Finally, for any application which makes use of the Python interpreter, you can also install or copy modules and packages directly to a directory which is included in Python's search path.

For example, you could install additional modules to the `$HOME/webapps/app_name/lib/pythonX.Y` directory, which are included in the search path for the application by default.

For more information about installing Python packages and modules, see *Installing Python Packages*.

19.4.2 Error: `Permission denied` During Package Installation

Some Python package installations, like lxml, attempt to execute files within `/tmp`, which is not permitted on a WebFaction server. To install such packages, you must create and specify your own temporary directory before installing the package. To install a package using your own temporary directory:

1. Open an SSH session to your account.

2. Create the temporary directory. Enter `mkdir -p $HOME/tmp` and press `Enter`. A new directory, `tmp`, is created in your home directory.
3. Configure the shell to use the new temporary directory. Enter `export TEMP=$HOME/tmp` and press `Enter`.
4. *Install the package*.

CHAPTER

TWENTY

RUBY

Ruby is installed on all WebFaction servers. For your convenience, several different Ruby versions are available. All servers have Ruby 1.9 through Ruby 2.2 installed. CentOS 5 and CentOS 6 servers also have Ruby 1.8.7 and Ruby 1.8.7 Enterprise Edition installed.

The default version of Ruby—the version of Ruby that runs when you run the `ruby` command—varies by server:

Operating system	Default `Ruby` version
CentOS 5	1.8.7
CentOS 6	1.8.7
CentOS 7	2.0

To identify the default version of Ruby on your server, run `ruby --version`.

To run a specific version of Ruby, run `ruby`*`X.Y`*, where *X.Y* is the Ruby version number. For example, to use Ruby 2.2, enter `ruby2.2`.

20.1 Installing Gems

You can install Ruby programs and libraries with the RubyGems package manager.

See also:

To install gems for a specific Ruby on Rails application, see *Installing Gems in our Ruby on Rails documentation*.

To install a gem:

1. Open an SSH session to your account.
2. Create a directory for the gems. Enter `mkdir -p {gems_dir}` where *gemsdir* is the directory path (for example, `$HOME/gemhome`).
3. Set the `GEM_HOME` environment variable. Enter `export GEM_HOME=`*`gems_dir`* and press `Enter`.
4. Set the `RUBYLIB` environment variable. Enter `export RUBYLIB=`*`gems_dir`*`/lib` and press `Enter`.
5. Set the `PATH` environment variable. Enter `export PATH=`*`gems_dir`*`/bin:$PATH` and press `Enter`.
6. Enter `gem install `*`gem_name`*, where *gem_name* is the name of the Rub Gem to install, and press `Enter`.

Note: To use another Ruby version instead of the default, enter `gem`*`version`*` install `*`gem_name`*, where *version* is an available version such as 1.9 or 2.1, and press `Enter`.

The gem is downloaded and installed. To use the gem's executables, the `gems_dir/bin` directory must be in the `PATH` environment variable. Run `export PATH=gems_dir/bin:$PATH` in each session as needed, or add the statement to your `.bash_profile` file.

CHAPTER

TWENTYONE

RUBY ON RAILS

Ruby on Rails, also known as *Rails* or *RoR*, is an open source Ruby web development framework. You can learn more about Ruby on Rails at the Rails website.

Tip: Add `bin` to `PATH`

While working with a Ruby on Rails application, you may find it easier to add the application's `bin` directory to the `PATH` environment variable and set the `GEM_HOME` environment variable, instead of specifying complete paths to `$HOME/webapps/app/bin` and `$HOME/webapps/app/gems`.

To set the environment variables:

1. Switch to the directory of the Ruby application. Enter `cd $HOME/webapps/app`, where *app* is the name of the application as it appears in the WebFaction control panel, and press `Enter`.
2. Enter `export PATH=$PWD/bin:$PATH` and press `Enter`.
3. Enter `export GEM_HOME=$PWD/gems` and press `Enter`.
4. Enter `export RUBYLIB=$PWD/lib` and press `Enter`.

Now, programs like `gem` and `rake` will work as expected for that application.

To revert the changes to `PATH` and `GEM_HOME` (to work with a different Ruby on Rails application, for example), log out and log back in again.

21.1 Installing Ruby on Rails

WebFaction's Ruby on Rails installer creates an nginx instance running Passenger. To install a Rails application:

1. Log in to the control panel.
2. Click *Domains / websites* → *Applications*. The list of applications appears.
3. Click the *Add new application* button. The *Create a new application* form appears.
4. In the *Name* field, enter a name for the application.
5. In the *App Category* menu, click to select *Rails*.
6. If applicable, in the *Machine* menu, click to select a web server.
7. Click the *Save* button. The application is installed and added to the list of applications.

To make the application available on the Web, *add the application to a website record*.

21.2 Upgrading RubyGems

You can upgrade any Passenger-based application to use a more recent version of RubyGems. To upgrade Ruby Gems:

1. Open an SSH session to your account.
2. Switch to the Passenger-based application's directory. Enter `cd $HOME/webapps/app`, where *app* is the name of your Passenger-based application, and press `Enter`.
3. Set the `RUBYLIB` environment variable. Enter `export RUBYLIB=$PWD/lib` and press `Enter`.
4. Set your `GEM_HOME` environment variable. Enter `export GEM_HOME=$PWD/gems` and press `Enter`.
5. Add the Passenger-based application's `bin` directory to your `PATH`. Enter `export PATH=$PWD/bin:$PATH` and press `Enter`.
6. Create a directory to store the RubyGems files. Enter `mkdir -p src` and press `Enter`.
7. Switch to the new directory. Enter `cd src` and press `Enter`.
8. Download the RubyGems archive. Enter `wget url` where *url* is the download URL for RubyGems and press `Enter`. The RubyGems archive will be created in the current directory.

   Note: Find the latest RubyGems download link at Download RubyGems.

9. Extract the RubyGems archive. Enter `tar -xzf archive`, where *archive* is the name of the RubyGems archive (for example, `rubygems-2.4.8.tgz`) and press `Enter`. `tar` extracts the RubyGems files to a directory, `rubygems-X.Y.Z`, where *X.Y.Z* is the version number for RubyGems.
10. Switch to the RubyGems directory. Enter `cd rubygems-X.Y.Z` and press `Enter`.
11. Run the RubyGems setup file. Enter `ruby setup.rb install --prefix=$HOME/webapps/app` and press `Enter`.

The new version of RubyGems is now installed for your Passenger-based application.

21.3 Installing Gems

See also:

To install gems in your home directory, see *Installing Gems* in our Ruby documentation.

To install a Ruby Gem in your Rails application:

1. Open an SSH session to your account.
2. Enter `cd $HOME/webapps/app`, where *app* is the name of your Ruby on Rails application, and press `Enter`.
3. Enter `export GEM_HOME=$PWD/gems` and press `Enter`.
4. Enter `export RUBYLIB=$PWD/lib` and press `Enter`.
5. Enter `export PATH=$PWD/bin:$PATH` and press `Enter`.

6. Enter `gem install gem_name`, where *gem_name* is the name of the Ruby Gem you want to install, and press `Enter`.

Common Gems

To install some common gems:

- MySQL adapter MySQL/Ruby:
  `gem install mysql -- --with-mysql-config=$(which mysql_config)`
- PostgreSQL adapter `postgres`:
  - For servers `Web300` and `Dweb89` or greater:
    `gem install pg -- --with-pg-config=/usr/pgsql-9.1/bin/pg_config`
  - For other servers: `gem install pg -- --with-pg-config=/usr/bin/pg_config`
- Testing framework RSpec: `gem install rspec`
- Web framework Sinatra: `gem install sinatra`

Note: If you do not want to install the *Rdoc* or *ri* documentation, use this version of the command:
`gem install --no-rdoc --no-ri gem_name`.

The gem and any of its specified dependencies will be downloaded and installed automatically.

21.4 Installing Multiple Gems with Bundler

You can use Bundler to install multiple dependency gems in your Rails application at once. To use Bundler:

1. Open an SSH session to your account.
2. Enter `cd $HOME/webapps/app`, where *app* is the name of your Ruby on Rails application, and press `Enter`.
3. Enter `export GEM_HOME=$PWD/gems` and press `Enter`.
4. Enter `export RUBYLIB=$PWD/lib` and press `Enter`.
5. Enter `export PATH=$PWD/bin:$PATH` and press `Enter`.
6. If you have not already installed Bundler for your Rails application, install Bundler. Enter `gem install bundler` and press `Enter`.
7. Switch to your Rails project directory. Enter `cd project`, where *project* is the name of your Rails project, and press `Enter`. For example, to use the default `hello_world` Rails project, enter `cd hello_world` and press `Enter`.
8. Install the gems specified in `Gemfile`. Enter `bundle install` and press `Enter`.

Bundler will download and install the specified gems.

21.5 Installing ImageMagick and RMagick

ImageMagick is an open source image creation and editing package.

1. Install ImageMagick.
   (a) Open an SSH session to your account.
   (b) Download ImageMagick. Enter `wget ftp://ftp.imagemagick.org/pub/ImageMagick/ImageMagick.tar.gz` and press `Enter`. `ImageMagick.tar.gz` is created in the current directory.
   (c) Unpack the ImageMagick archive. Enter `tar -xf ImageMagick.tar.gz` and press `Enter`. A new directory, `ImageMagick-version`, where *version* is the ImageMagick version number, is created.
   (d) Switch to the ImageMagick directory. Enter `cd ImageMagick-version` and press `Enter`.
   (e) Configure the ImageMagick build. Enter `./configure --prefix=$HOME --without-perl` and press `Enter`.
   (f) Compile ImageMagick. Enter `make` and press `Enter`. ImageMagick compiles. This step may take several minutes.
   (g) Install ImageMagick to your home directory. Enter `make install` and press `Enter`.
   (h) Return to your home directory. Enter `cd` and press `Enter`.
   (i) Remove the installation files. Enter `rm -r ImageMagick.tar.gz ImageMagick-version` and press `Enter`.
2. Add `$HOME/bin` to your `PATH`.
   (a) Enter `echo "export PATH=$HOME/bin:\$PATH" >> .bash_profile` and press `Enter`.
   (b) Reload your `.bash_profile`. Enter `source .bash_profile` and press `Enter`.
3. Install the RMagick gem.
   (a) Enter `export LD_LIBRARY_PATH=$HOME/lib` and press `Enter`.
   (b) Enter `export PKG_CONFIG_PATH=$HOME/lib/pkgconfig/` and press `Enter`.
   (c) *Install the RMagick gem* in your Ruby on Rails application.

The gem is now installed for your Rails application.

21.6 Installing *sqlite3-ruby*

By default, Rails is configured for use with SQLite version 3 databases. To use a SQLite 3 database, the sqlite3-ruby gem must be installed. To install the gem:

1. *Install the latest version of SQLite from source* in your home directory.
2. Install the *sqlite3-ruby* gem.
   (a) Open an SSH session to your account.
   (b) Enter `cd $HOME/webapps/app`, where *app* is the name of your Ruby on Rails application, and press `Enter`.
   (c) Enter `export GEM_HOME=$PWD/gems` and press `Enter`.
   (d) Enter `export RUBYLIB=$PWD/lib` and press `Enter`.

(e) Enter `export PATH=$PWD/bin:$PATH` and press `Enter`.

(f) Enter `export LD_LIBRARY_PATH=$HOME/lib/` and press `Enter`.

(g) Enter `gem install sqlite3-ruby -- --with-sqlite3-dir=$HOME` and press `Enter`.

21.7 Deploying a Ruby on Rails Application

To start using your own application with a Passenger-based Rails application:

1. Upload your application to your Rails application directory. For example, if you have an application, `myapp`, upload it to `$HOME/webapps/app/myapp` where *app* is the name of your Rails application as it appears in the WebFaction control panel.
2. Open an SSH session to your account.
3. Switch to the Rails application directory. Enter `cd $HOME/webapps/app` where *app* is the name of your Rails application, and press `Enter`.
4. Open `./nginx/conf/nginx.conf` in a text editor.
5. Replace `hello_world` in the line containing `root /home/username/webapps/app/hello_world/public` with the name of your Rails app's directory. In the earlier example, the line would contain `root /home/username/webapps/app/myapp/public`.

   Note: You may now delete the provided `hello_world` directory.

6. Reboot your Rails application. Enter `./bin/restart` and press `Enter`.

21.8 Deploying a Ruby on Rails Application with Capistrano

To start using Capistrano with a Ruby on Rails application:

1. If you have not already done so, install Capistrano on your local workstation. Typically, you can install Capistrano by running `gem install capistrano` in a terminal session. See the Capistrano Installation page for details.
2. Add Capistrano to your Rails project.
   (a) From the root directory of your Rails application, initialize Capistrano. Enter `cap install` and press `Enter`.

   (b) Open `config/deploy.rb` in a text editor.

   (c) Set your Rails application's name. Replace `set :application, 'my_app_name'` with `set :application, 'app_name'`, where *app_name* is the name of the application. For example, if your project is named for the default `hello_world` application, enter `set :application, 'hello_world'`.

   (d) Set your repository URL. Replace `set :repo_url, 'git@example.com:me/my_repo.git'` with `set :repo_url, 'url'`, where *url* is the URL of your Git repository.

Note: These directions assume that your Ruby on Rails project is part of a publicly-accessible Git repository. For more information, see the official Capistrano documentation.

(e) Set your deployment path. Replace `# set :deploy_to, '/var/www/my_app'` with `set :deploy_to, '/home/username/webapps/railsapp'`, where *username* is your WebFaction username and *railsapp* is the name of your Rails application as it appears on the control panel.

(f) Set your temporary directory. On a new line, add `set :tmp_dir, '/home/username/tmp'`.

(g) Define a restart task. In the `namespace :deploy do` block, insert the following lines:

```
desc 'Restart application'
  task :restart do
    on roles(:app), in: :sequence, wait: 5 do
      capture("#{deploy_to}/bin/restart")
  end
end
```

(h) Set the restart task to run after deployments. On a new line, add `after 'deploy:publishing', 'deploy:restart'`.

(i) Save and close the file.

(j) Open `config/deploy/staging.rb` in a text editor.

(k) Insert these lines:

```
role :app, %w{deploy@example.com}
role :web, %w{deploy@example.com}
role :db,  %w{deploy@example.com}
```

substituting `deploy@example.com` with `username@host`, where *username* is your username and host is your WebFaction server (such as `web310.webfaction.com`).

Note: These directions assume that your are already using SSH keys. For more information about setting up SSH keys, see *Using SSH Keys*.

(l) Save and close the file.

3. Configure the application to support Capistrano deployment.

   (a) Open an SSH session to your account.

   (b) In a text editor, open `/home/username/webapps/railsapp/nginx/conf/nginx.conf`.

   (c) Replace `root /home/username/webapps/railsapp/hello_world/public` with `root /home/username/webapps/railsapp/current/public`.

   (d) Save and close the file.

4. Deploy. On your local workstation, enter `cap staging deploy` and press `Enter`.

Your Rails application is deployed.

21.9 Upgrading Ruby on Rails

From time-to-time, you may want to upgrade your Rails application, especially when new security releases are issued. Your project code may require modification to be used with the latest version of Rails or its dependencies.

Before upgrading to a new version of Rails, review the release notes for the new version and any intermediate versions of Rails and confirm that you have an up-to-date backup of your site.

See also:

For additional details and version-specific upgrade notes, see A Guide for Upgrading Ruby on Rails.

To upgrade Rails from one version to another:

1. *Open an SSH session to your account*.
2. Switch to the Rails application directory. Enter `cd $HOME/webapps/rails_app`, where *rails_app* is the name of the Rails application, and press `Enter`.
3. Enter `export PATH=$PWD/bin/:$PATH` and press `Enter`.
4. Enter `export GEM_HOME=$PWD/gems/` and press `Enter`.
5. Enter `export RUBYLIB=$PWD/lib` and press `Enter`.
6. Switch to your Rails project directory. Enter `cd project`, where *project* is the name of your Rails project, and press `Enter`. For example, to use the default `hello_world` project, enter `cd hello_world` and press Enter.
7. Open your project's `Gemfile` file in a text editor.
8. In the line containing `gem 'rails', 'version'`, where *version* is the existing Rails version number, replace the existing version number with the version number that you want to upgrade to. For example, to upgrade from version `4.0.7` to version `4.0.8`, replace `gem 'rails', '4.0.7'` with `gem 'rails', '4.0.8'`.
9. Save and close the file.
10. Update Rails and any applicable Rails dependencies. Enter `bundle update rails` and press `Enter`. When the message "Your bundle is updated!" appears, Rails is updated. Libraries that are not in Rails' dependency tree remain unchanged.
11. Restart your rails application. Enter `restart` and press `Enter`.

Your application is now running the updated version of Rails that you specified in `Gemfile`.

21.10 Using a Database with a Ruby on Rails Application

To configure a MySQL or PostgreSQL for use with a Ruby on Rails application:

1. Create a database.
   (a) Use the WebFaction control panel to create a MySQL or PostgreSQL database.

   See also:

   For step-by-step directions, see *Creating a New Database with the Control Panel*.

   (b) Make a note of the database name and password.
2. *Install the Ruby database adapter gem*. Install the `mysql` gem for MySQL. Install the `postgres` gem for PostgreSQL.
3. Configure the Ruby on Rails project to use the MySQL database.
   (a) Switch to the directory of the Ruby on Rails project. Enter `cd project`, where *project* is the name of the project (for example, `hello_world`), and press `Enter`.

(b) Open `config/database.yml` in a text editor.

(c) Edit the `production` section:

```
production:
 adapter: sqlite3
 database: db/production.sqlite3
 pool: 5
 timeout: 5000
```

to this:

```
production:
 adapter: adapter_type
 database: database_name
 host: localhost
 username: database_user
 password: database_password
```

such that

- `adapter_type` is either `mysql` for MySQL databases or `postgresql` for PostgreSQL databases,
- `database_name` is the name of the database as it appears in the WebFaction control panel,
- `database_user` is set to the name of a database user with access to the specified database, and
- `database_password` is the database user's password.

(d) Save and close the file.

The Rails application is now configured and ready to use the specified database.

21.11 Configuring Action Mailer

Note: Configuring Action Mailer applies to Ruby on Rails 3.0 or later only.

To configure your Rails application:

1. Open `$HOME/webapps/`*`rails`*`/`*`application`*`/config/environments/production.rb` in a text editor, where *rails* is the name of the Ruby on Rails application as it appears in the WebFaction control panel and *application* is the name of your Ruby on Rails project (for example, `hello_world`).
2. In the `HelloWorld::Application.configure do` block, where *HelloWorld* corresponds to your Ruby on Rails application name, add this line:

```
config.action_mailer.delivery_method = :sendmail
```

See also:

For other Action Mailer settings, see Action Mailer Basics.

3. Save and close the file.

21.12 Troubleshooting

21.12.1 Error: `We're sorry, but something went wrong.`

If you attempt to click Rails' *About your application's environment* link, you may get an error like this:

> We're sorry, but something went wrong.
>
> We've been notified about this issue and we'll take a look at it shortly.

This error occurs when the link is clicked in `production` rather than `development` mode. To switch your Rails application to `development`:

1. *Install sqlite3-ruby*.
2. Open an SSH session to your account.
3. Open `$HOME/webapps/app/nginx/conf/nginx.conf`, where *app* is the name of the Rails application as it appears in the control panel, in a text editor.
4. In the `server` block, add `rails_env development;` on a new line.
5. Save and close the file.
6. Restart the Rails application. Enter `$HOME/webapps/rails/bin/restart` and press `Enter`.

Note: Rails `development` mode consumes dramatically more resources than `production`. Be sure to return to `production` mode when `development` is no longer needed.

21.12.2 Error: `Could not spawn process for application`

If your Passenger-based application, including Rails or Redmine, fails to start as expected and causes an error such as `We're sorry, but something went wrong` in your browser and `Could not spawn process` in your application's error logs, then you may be having problems with the way Passenger starts (spawns) your application.

When Passenger starts your application, it uses a Bash login shell that loads your `~/.bash_profile` and `~/.bashrc` files. Some `.bash_profile` or `.bashrc` files may prevent Passenger from starting your application or cause the application to exit too soon.

Some problems related to `.bash_profile` and `.bashrc` files include:

- Changes to the `PATH`, `GEM_HOME`, or `RUBYLIB` environment variables that prevent Passenger from finding Ruby or gems (even if those variables are explicitly set in the application's start script or `nginx.conf` file)
- Using RVM, which may prevent Passenger from finding Ruby or gems
- Redirecting or closing standard output
- Exiting Bash

If your application fails to start or run as expected, review your `.bash_profile` and `.bashrc` files for conflicts with your application's requirements. If you need to run some commands from your `.bash_profile` or `.bashrc` file while Passenger is not running, you can check whether the file is being run under Passenger like this:

```
if [ "$SERVER_SOFTWARE" != "${SERVER_SOFTWARE%Passenger*}" ]; then
    # Passenger is running. Run Passenger-specific commands here.
else
    # Passenger is not running. Run commands that conflict with Passenger here.
fi
```

For more information, see the Passenger wiki page Debugging application startup problems.

21.12.3 Error: `Missing 'secret_token' and 'secret_key_base' for 'production' environment`

If your Rails application is running in production mode and you find this error message in your `~/webapps/rails_app/nginx/logs/error.log` file:

```
*** Exception RuntimeError in Rack application object (Missing `secret_token` and `secret_key_base` i
```

then a secret key may not be configured for your application. The error may also be accompanied by a *502 Bad Gateway* error or *Incomplete response received from application* error in your web browser.

To set your Rails application's production mode secret key:

1. *Open an SSH session to your account*.
2. Make a secret key for your application.
   (a) Switch to your Rails application directory. Enter `cd $HOME/webapps/rails_app/`, where *rails_app* is the name of your Rails application, and press `Enter`.
   (b) Generate a secret key. Enter `GEM_HOME=$PWD/gems/ $PWD/bin/rake -f $PWD/project/Rakefile secret`, where *project* is the name of your Rails project, and press `Enter`.

   For example, to generate a secret key for the default `hello_world` Rails project, enter `GEM_HOME=$PWD/gems/ $PWD/bin/rake -f $PWD/hello_world/Rakefile secret` and press `Enter`.

   A long sequence of hexadecimal characters (characters 0-9 and a-f) appears. This sequence of characters is your secret key.
   (c) Copy the secret key.
3. Configure your application to use the key you generated.
   (a) Open `~/webapps/rails_app/nginx/conf/nginx.conf` in a text editor.
   (b) Set the `SECRET_KEY_BASE` environment variable. On a new line in the `server` block, add `passenger_env_var SECRET_KEY_BASE key;`, where *key* the secret key you generated earlier.
   (c) Save and close the file.
4. Restart your Rails application. Enter `$HOME/webapps/rails_app/bin/restart` and press `Enter`.

The production mode secret key is set.

CHAPTER

TWENTYTWO

REDMINE

Redmine is an open source project management application.

22.1 Configuring Redmine to Send Email

1. Configure Redmine to send mail.
   (a) Open an SSH session to your account.
   (b) Open `$HOME/webapps/app/redmine/config/configuration.yml`, in a text editor, where *app* is the name of the Redmine application.
   (c) Under `default:` replace these lines:

```
email_delivery:
  delivery_method: :smtp
  smtp_settings:
    address: smtp.example.net
    port: 25
    domain: example.net
    authentication: :login
    user_name: "redmine@example.net"
    password: "redmine"
```

   with these lines:

```
email_delivery:
  delivery_method: :smtp
  smtp_settings:
    enable_starttls_auto: true
    address: smtp.webfaction.com
    port: 587
    domain: <domain>
    authentication: :plain
    user_name: "<mailbox>"
    password: "<password>"
```

   where:

   - `<domain>` is the domain name your Redmine application will send mail from (such as `example.com`),
   - `<mailbox>` is a mailbox name as it appears on the WebFaction control panel (for example, *`username`*`_redmine`), and

- `<password>` is the mailbox's password.

(a) Save and close the file.

(b) Restart the Redmine application. Enter `$HOME/webapps/app/bin/restart` and press `Enter`.

2. Set up a valid destination for testing Redmine email notifications.

   (a) Sign in to your Redmine website with an admin account.

   (b) Click *My account*. The *My account* form appears.

   (c) In the *Email* field, enter the email address where you want to receive Redmine notifications.

   (d) Click the *Save* button. A confirmation message appears.

3. Configure the outgoing email address and test the email notifications.

   (a) Click *Administration*. The *Administration* list appears.

   (b) Click *Settings*. The *General* configuration tab appears.

   (c) Click the *Email notifications* tab.

   (d) In the *Emission email address* field, enter the email address you want Redmine to use to send outgoing messages (such as `redmine@my.example.com`).

   (e) Click *Save*. A confirmation message appears.

   (f) Click *Send a test email*. A confirmation message appears.

4. Check your email. If a message from Redmine arrives, you have successfully configured Redmine to send email.

CHAPTER

TWENTYTHREE

SQLITE

SQLite is a public domain SQL database engine.

23.1 Installing SQLite

A version of the SQLite 3 shell is installed on all servers; however, you may want to install a newer version. To install a newer version of SQLite 3:

1. Open an SSH session to your account.
2. Switch to your `~/bin` directory. Enter `cd ~/bin` and press `Enter`.

   Note: If the directory does not already exist, create it. Enter `mkdir ~/bin` and press `Enter`.

3. Download the latest SQLite 3 precompiled binary for Linux. Enter `wget url`, where *url* is the SQLite download URL found at the SQLite Download Page, and press `Enter`. An archive containing the binary is created in the current directory.
4. Unzip the archive. Enter `unzip archive`, where *archive* is the name of the SQLite archive file, and press `Enter`. An executable file, `sqlite`, is created in the current directory.
5. Add the `~/bin` directory to your PATH.
   (a) Open `~/.bashrc` in a text editor.
   (b) Insert `export PATH=$HOME/bin/:$PATH` on a new line.
   (c) Save and close the file.

Now `sqlite` is the latest version of the SQLite 3 shell.

23.2 Installing SQLite from Source

To install a complete version of SQLite from source:

1. Open an SSH session to your account.
2. Download the latest SQLite source autoconf source. Enter `wget url`, where *url* is the SQLite download URL found at the SQLite Download Page, and press `Enter`. An archive containing the binary is created in the current directory.

3. Extract the archive. Enter `tar -xf sqlite-autoconf-version.tar.gz`, where *version* is the version number, and press `Enter`. A new directory containing the source files, `sqlite-autoconf-version`, is created in the current directory.
4. Switch to the source directory. Enter `cd sqlite-autoconf-version` and press `Enter`.
5. Configure the compilation. Enter `./configure --prefix=$HOME` and press `Enter`.
6. Compile SQLite. Enter `make` and press `Enter`.
7. Install SQLite. Enter `make install` and press `Enter`.
8. Add the `~/bin` directory to your PATH.
   (a) Open `~/.bashrc` in a text editor.
   (b) Insert `export PATH=$HOME/bin/:$PATH` on a new line.
   (c) Save and close the file.

CHAPTER

TWENTYFOUR

STATIC FILES, CGI SCRIPTS, AND PHP PAGES

Static files, CGI scripts, and PHP pages can be served by two different, but related, types of applications, *Static-only* and *Static/CGI/PHP* applications.

Static-only applications serve files through each WebFaction server's front-end nginx process, but never run CGI scripts or interpret PHP pages. While limited to serving ordinary files, like HTML, images, and CSS, Static-only applications serve media fast without additional memory consumption.

Static/CGI/PHP applications serve ordinary files like Static-only applications do, along with the ability to run CGI scripts, interpret PHP pages, and customize settings with `.htaccess` files.

See also:

For more on how each server handles incoming requests and which processes are responsible for different activities, please see *The Life Cycle of a Request and Response*.

24.1 Static-only

Static-only apps serve static files with the server's shared nginx process. A Static-only app is exclusively static; PHP files, for example, will not be interpreted in a Static-only app. We suggest Static-only apps when you need to serve static media (and only static media) fast with low memory consumption.

24.1.1 Serving Static Media

Even if most of your site is served dynamically by Django, Rails, or Node.js (to name a few), you will still have many static elements, such as style sheets, images, and videos. If these elements are served by your dynamic process, they will consume unnecessary additional memory and time to serve. You can save memory and time by serving static files with a Static-only application.

To serve static media with a Static-only application:

1. With the control panel, create a Static-only application.
2. With the control panel, modify or create a website entry which maps an unused path from your domain name (for example, `www.example.com/media`) to your new Static-only application.
3. Wait up to two minutes for propagation of the new path and domain combination to complete.
4. Copy static media files to `$HOME/webapps/static-only`.

Media served from the `/media` path of your website is now served by an nginx or Apache process, sparing your dynamic application from serving those requests.

24.1.2 Setting `expires max`

For new Static-only applications, you can enable long-lived HTTP caching headers. To enable these headers, enter `expires max` in the *Extra info* field while creating a new Static-only application.

Specifically, the `expires max` configuration value sets:

- the `Expires` header to 31 December 2037 23:59:59 GMT, and
- the `Cache-Control` header's `max-age` directive to 10 years.

24.2 Static/CGI/PHP

Static/CGI/PHP apps, in addition to their role in serving static media, add the capacity to serve PHP pages, which are processed by the PHP interpreter on the server. Static/CGI/PHP apps should only be used when you need to make use of `.htaccess` files, PHP pages, or CGI scripts.

See also:

For additional, PHP-specific configuration and troubleshooting information, please see *PHP*.

24.2.1 Adding a MIME Type

If you need to use a file extension that Apache does not recognize automatically, you can instruct Apache to use a specific MIME type with a `.htaccess` directive. To add a MIME type:

1. *Open an SSH session to your account*.
2. Switch to the Static/CGI/PHP application's directory. Enter `cd $HOME/webapps/app`, where *app* is the name of the application as it appears in the control panel, and press `Enter`.
3. Open `.htaccess` (or create the file if it does not already exist) in a text editor.
4. For each new MIME type, add a new line containing `AddType mime extension`, where *mime* is the MIME type and *extension* is the file extension (including a dot).

   For example, to set `.jnlp` files to use the `application/x-java-jnlp-file` MIME type, insert `AddType application/x-java-jnlp-file .jnlp` into the `.htaccess` file.
5. Save and close the file.

24.2.2 Blocking an IP Address or Hostname

To block an IP address or hostname for accessing a *Static/CGI/PHP* application:

1. Open an SSH session to your account.
2. Switch to the Static/CGI/PHP application's directory. Enter `cd $HOME/webapps/app`, where *app* is the name of the application as it appears in the control panel, and press `Enter`.
3. Open `.htaccess` (or create the file if it does not already exist) in a text editor.
4. Add these lines to the file:

```
order allow,deny
<deny directives>
allow from all

<Files .htaccess>
    order allow,deny
    deny from all
</Files>
```

where `<deny directives>` is any number of lines containing one directive each to deny access to IP addresses or hostnames. Such directives take the form of `deny from IP` and `deny from hostname`, where *IP* is an IP address to be blocked and *hostname* is a hostname to be blocked. For example:

```
order allow,deny
deny from example.com
deny from 123.456.78.9
allow from all

<Files .htaccess>
    order allow,deny
    deny from all
</Files>
```

5. Save and close the file.

Future requests from the blocked IP addresses and hostnames will be rejected with a 403 Forbidden error.

24.2.3 Changing the Handler for Files

By default, a Static/CGI/PHP application's instance of Apache will serve `.cgi` and `.py` files as CGI scripts. You can instruct Apache to use a different handler with a set of `FilesMatch` and `SetHandler` directives in your application's `.htaccess` file.

To modify the handler:

1. Open or create a new file `$HOME/webapps/app/.htaccess`, where *app* is the name of the application as it appears on the control panel.
2. For each new handler you would like to add, insert these lines:

```
<FilesMatch \.EXTENSION$>
  SetHandler HANDLER
</FilesMatch>
```

where *EXTENSION* is the file extension to apply the handler to and *HANDLER* is the handler to use.

For example, to treat `.py` files as ordinary text files instead of as CGI scripts, add these lines to your `.htaccess` file:

```
<FilesMatch \.py$>
  SetHandler default-handler
</FilesMatch>
```

For another example, to treat `.xyz` files as CGI scripts, add these lines to your `.htaccess` file:

```
<FilesMatch \.xyz$>
  SetHandler cgi-script
</FilesMatch>
```

3. Save and close the file.

24.2.4 Enabling Server Side Includes

Server Side Includes (SSI) direct the Apache web server to substitute portions of HTML pages with content evaluated at the time the page is served.

To enable server side includes:

1. Open an SSH session.
2. Switch to the Static/CGI/PHP application's directory. Enter `cd $HOME/webapps/app`, where *app* is the name of the application as it appears in the control panel, and press `Enter`.
3. Open `.htaccess` (or create the file if it does not already exist) in a text editor.
4. Add these lines:

```
Options +Includes
AddType text/html .shtml
AddOutputFilter INCLUDES .html
```

5. Save and close the file.

Now, you can use include directives in your pages, such as `<!--#include file="includes/header.shtml"-->`.

See also:

Apache Tutorial: Introduction to Server Side Includes

24.2.5 Granting Apache Write Access

To make a file or directory writable by the `apache` user:

1. Open an SSH session to your account.
2. Enter `setfacl -R -m u:apache:rwx path`, where *path* is the path to the file or directory, and press `Enter`.
3. Enter `setfacl -R -m default:u:username:rwx path`, where *username* is your account name, and press `Enter`.

24.2.6 Hiding Configuration Files

By default, files such as `.htaccess` and `.htpasswd` are exposed to the web. If you want to hide these files, add these lines to your `.htaccess` file:

```
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>
```

24.2.7 Password Protecting a Directory with a Static/CGI/PHP App

You can password protect the contents of any Static/CGI/PHP application by using a `.htaccess` file. This method works for all CGI and PHP-based applications, including AWStats, Drupal, Joomla, Trac, and WordPress.

To enable password protection:

1. With the control panel, create a Static/CGI/PHP app or identify an existing Static/CGI/PHP app you would like to use.
2. If necessary, create or modify a website entry which maps the Static/CGI/PHP app to a particular URL.
3. Open an SSH session into your account.
4. Switch to the `$HOME/webapps/static_cgi_php_app/` directory.
5. If it does not already exist, create a `.htaccess` file. Enter `touch .htaccess` and press `Enter`.
6. Open `.htaccess` in a text editor.
7. Add these directives to `.htaccess`:

```
AuthUserFile /home/<your_username>/webapps/<webapp_name>/.htpasswd
AuthName EnterPassword
AuthType Basic
require valid-user

# Hide files starting with a dot (.htaccess and .htpasswd in particular)
<Files .*>
order allow,deny
deny from all
</Files>
```

8. Save and close the file.
9. To create the first user with access to the directory, enter `htpasswd -c .htpasswd username` and press `Enter`. A `New password` prompt appears.

   See also:

   See *Strengthening Passwords* for important information about choosing passwords.
10. Enter the password for the new user and press `Enter`. A `Re-type new password` prompt appears.
11. Reenter the password for the new user and press `Enter`.
12. To create additional users with access to the directory, enter `htpasswd .htpasswd username` and press `Enter`, then follow the password prompts for the new user.

Note: If you would like to password protect a subdirectory, rather than the entire application, create or modify `.htaccess` in the subdirectory.

Now, when you access the URL associated with the app's protected directory, your browser will prompt you for a username and password.

24.2.8 Redirecting from HTTP to HTTPS

See also:

Secure Sites (HTTPS)

If a site is only available over HTTPS, then use this method to automatically redirect from `http://domain` to `https://domain`:

1. Create website with a static application for redirecting incoming requests.
   (a) Log in to the control panel.
   (b) Click *Domains / websites* → *Websites*. The list of websites appears.
   (c) Click the *Add new website* button. The *Create a new website* form appears.
   (d) In the *Name* field, enter a website name.
   (e) If applicable, in the *Machine* menu, click to select the server to host the website.
   (f) If applicable, in the *IP address* menu, click to select the IP address to serve the site.
   (g) For each domain name you want to redirect from HTTP to HTTPS, add it to the list of domains. In the *Domains* field, enter the domain name. Enter one or more domain names. If the domain has not yet been added to the control panel, click the *Create* link that appears at the bottom of the list of domains to add it.

   Note: Don't forget to *point new domains to the WebFaction name servers*.

   (h) Click *Add an application* → *Create a new application*. The *Create a new web app* form appears.
   (i) In the *Name* field, enter a name for the application.
   (j) In the *App category* menu, click to select *Static*.
   (k) In the *App type* menu, click to select *Static/CGI/PHP-5.6*.
   (l) Click the *Save* button. The application is installed and added to website's list of applications.
   (m) Click the *Save* button. The website is created and added to the list of websites.
2. Add the redirect rule.
   (a) In a text editor, open `~/webapps/app_name/.htaccess`, where *app_name* is the name of the new Static/CGI/PHP application.
   (b) Append the following lines:

```
RewriteEngine On
RewriteCond %{HTTP:X-Forwarded-SSL} !on
RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [R=301,L]
```

   (c) Save and close the file.

New incoming requests to the specified domains redirect to HTTPS.

24.2.9 Redirect a Domain with a Static/CGI/PHP App

You can use a Static/CGI/PHP app to redirect one domain to another by using Apache's URL rewriting feature. For example, you can use a Static/CGI/PHP app to automatically redirect requests arriving at `example.com` to `www.example.com`.

To redirect from `origin_domain` to `destination_domain`:

1. Create website with a static application for redirecting incoming requests.
   (a) Log in to the control panel.
   (b) Click *Domains / websites* → *Websites*. The list of websites appears.

(c) Click the *Add new website* button. The *Create a new website* form appears.

(d) In the *Name* field, enter a website name.

(e) If applicable, in the *Machine* menu, click to select the server to host the website.

(f) If applicable, in the *IP address* menu, click to select the IP address to serve the site.

(g) For each origin domain, add it to the list of domains. In the *Domains* field, enter the domain name. Enter one or more domain names. If the domain has not yet been added to the control panel, click the *Create* link that appears at the bottom of the list of domains to add it.

Note: Don't forget to *point new domains to the WebFaction name servers*.

(h) Click *Add an application → Create a new application*. The *Create a new web app* form appears.

(i) In the *Name* field, enter a name for the application.

(j) In the *App category* menu, click to select *Static*.

(k) In the *App type* menu, click to select *Static/CGI/PHP-5.6*.

(l) Click the *Save* button. The application is installed and added to website's list of applications.

(m) Click the *Save* button. The website is created and added to the list of websites.

2. Add the redirect rule.

(a) In a text editor, open `~/webapps/app_name/.htaccess`, where *app_name* is the name of the new Static/CGI/PHP application.

(b) Append the following lines:

```
Options +FollowSymLinks
RewriteEngine on
RewriteCond %{HTTP_HOST} ^origin_domain$ [NC]
RewriteRule ^(.*)$ http://destination_domain/$1 [R=301,L]
```

Thus, the file to redirect `example.com` to `www.example.com` would look like this:

```
Options +FollowSymLinks
RewriteEngine on
RewriteCond %{HTTP_HOST} ^example.com$ [NC]
RewriteRule ^(.*)$ http://www.example.com/$1 [R=301,L]
```

(c) Save and close the file.

All subsequent requests to a URL at `origin_domain` are rewritten to corresponding URLs at `destination_domain`.

24.2.10 Serving a Static/CGI/PHP application from a Subdirectory

Some frameworks and applications serve the publicly reachable site from a subdirectory, rather than a root directory. For example, the source for an application might have a directory structure like this:

```
demosource/
    demo/
      app/
      config/
      tests/
    www/
```

in which `www` is the only directory to be served publicly. In situations like this, use a *Symbolic link* application. A *Symbolic link* application works like a *Static/CGI/PHP* application, except that files are served from a directory you choose (instead of a `~/webapps/app_name/` directory). The directory you choose can be any directory that you have access to (including directories inside other applications).

To create a simple website that serves from a subdirectory:

1. Log in to the WebFaction control panel.
2. Click *Domains / websites* → *Websites*. The list of websites appears.
3. Click the *Add new website* button. The *Create a new website* form appears.
4. In the *Name* field, enter a website name.
5. In the domains field, enter a domain name.
6. Click *Add an application* → *Create a new application*. The *Create a new web app* form appears.
7. In the *Name* field, enter a name for the application.
8. In the *App category* menu, click to select *Symbolic link*.
9. In the *Extra info* field, enter the full path to the subdirectory to be served (for example, `/home/demo/demosource/www/`). The path must point to an existing directory; the path is not created for you.
10. Click the *Save* button. The application is installed and added to the website's list of applications.
11. Click the *Save* button. The website is created and added to the list of websites.

Now, the contents of the directory are served at the root of the specified domain.

24.2.11 Using `RewriteBase`

If you are using mod_rewrite (activated with the `RewriteEngine on` directive) you may need to use the `RewriteBase` directive to get the desired results. The `RewriteBase` directive sets the base URL for rewrites; in other words, it establishes the starting URL path for all of your subsequent URL rewrites.

In almost all cases, if you are using mod_rewrite and the `RewriteEngine on` directive, you should also set `RewriteBase` in your `.htaccess` file to the corresponding URL path in the control panel.

For example, if your application is mounted at the root URL path (`/`), then your `RewriteBase` directive should be `RewriteBase /`. If the application is mounted elsewhere, the `RewriteBase` directive should match. For example, if your application is mounted at `/blog`, then your `RewriteBase` directive should be `RewriteBase /blog`.

See also:

Apache's RewriteBase Directive documentation

24.3 Troubleshooting

24.3.1 Error: *500 Internal Server Error*

A *500 Internal Server Error* is a generic error, which indicates that the Apache failed to complete a response. In other words, Apache failed to do *something* during the request. To resolve these errors, it's best to look in the error log for the Static/CGI/PHP application responsible for the error. The error log is

`$HOME/logs/frontend/error_website_php.log`, where *website* is the name of the website entry on which the Static/CGI/PHP application is mounted.

In the following subsections, you'll find common errors explained with typical solutions.

See also:

This sections covers language agnostic causes to *500 Internal Server Error*. Please see the *PHP-specific documentation* for such errors related directly to PHP.

See also:

Accessing Logs

Error: `Permission denied`

If you encounter the `500 Internal Server Error` in the browser and this error in your `$HOME/logs/frontend/error_website_php.log` log file:

```
[Thu Sep 16 12:00:00 2010] [error] [client 12.345.678.9] (13)Permission denied: exec of '/home/<user
```

it means that the permissions for your script are set incorrectly. Your user account must have execute permissions on the file and its parent directory.

To set the correct permissions:

1. Open an SSH session to your account.
2. Switch to the directory of the script. Enter `cd script_dir`, where *script_dir* is the path to the directory where the offending script is, and press `Enter`.
3. Set the file permissions on the script and its parent directory. Enter `chmod 711 . script`, where *script* is the file name of the script, and press `Enter`.

Your script should now run without the `Permission denied` error.

Error: `Premature end of script headers`

If you encounter the `500 Internal Server Error` in the browser and this error in your log file:

```
[Thu Sep 16 12:00:00 2010] [error] [client 12.345.678.9] Premature end of script headers: <script nam
```

a likely cause is that the permissions for your CGI script are set incorrectly. Your user account must have a minimal set of execute permissions on the file and its containing directory such that:

- the script is executable by the user,
- the script's parent directories (up to its containing Static/CGI/PHP application) are executable by the user;
- the script is not writable by group or other, and
- the script's parent directories (up to its containing Static/CGI/PHP application) are not writable by group or other.

To set the correct permissions:

1. Open an SSH session to your account.
2. Switch to the directory of the script. Enter `cd script_dir`, where *script_dir* is the path to the directory where the offending script is, and press `Enter`.

3. Set the file permissions on the script and its parent directory. Enter `chmod 711 . *script*`, where *script* is the file name of the script, and press `Enter`.

   For any additional parent directories enter `chmod 711 *dirs*`, where *dirs* are a space-separated list of paths to directories, and press `Enter`.

The script's permissions should now be set appropriately.

Error: `No such file or directory`

If you encounter the `500 Internal Server Error` in the browser and this error in your `$HOME/logs/frontend/error_*website*_php.log` log file:

```
[Thu Sep 16 12:00:00 2010] [error] [client 12.345.678.9] (2)No such file or directory: exec of '/home
```

it typically indicates one of these problems:

1. a CGI script contains `\\r\\n` (or CRLF) line endings instead of Unix-standard `\\n` or LF line endings, or
2. the path specified in the script's shebang line does not exist.

Incorrect Line Endings

You can verify line endings with the `hexdump` tool.

In this example, the wrong line ending is represented as `0a0d`:

```
$ hexdump my_file
0000000 2123 752f 7273 6c2f 636f 6c61 622f 6e69
0000010 702f 7479 6f68 326e 352e 0a0d ...
```

To fix incorrect line endings, use `dos2unix`:

```
$ dos2unix my_file
dos2unix: converting file my_file to UNIX format ...
$ hexump my_file
0000000 2123 752f 7273 6c2f 636f 6c61 622f 6e69
0000010 702f 7479 6f68 326e 352e 0a0a ...
```

Now the file has the correct line endings for use on the WebFaction server. Alternatively, you may reconfigure your STFP client to save uploaded files with the proper line endings.

Incorrect Shebang Line

Many scripts begin with a shebang line to indicate which interpreter is to be used to run the script. For example, a Python 3.4 script's shebang line might look like this:

```
#!/usr/bin/env python3.4
```

If the shebang line points to an interpreter that doesn't exist, however, the script will fail with an error.

A similar error will appear when the script is run from the command line. For example:

```
$ ./test.py
-bash: ./test.py: /usr/bin/env pyton2.5: bad interpreter: No such file or directory
```

This error indicates that the path specified on the shebang line does not exist. In the example, there's a typo: `pyton`. Once the typo is corrected, the script will run without error.

Error: `Invalid command` or `Illegal option` with `.htaccess`

If you encounter the `500 Internal Server Error` in the browser and one of these errors in your `$HOME/logs/frontend/error_website_php.log` log file:

```
[Thu Sep 16 12:00:00 2010] [error] [client 12.345.678.9] /path/to/.htaccess: Invalid command '<token>
[Thu Sep 16 12:00:00 2010] [error] [client 12.345.678.9] /path/to/.htaccess: Illegal option <token>,
```

Then it means that there is an error in your `.htaccess` file. Make sure your `.htaccess` file does not contain any syntax errors or misspelled directives. Any error in the file will cause the entire `.htaccess` file to fail.

Error: `Request exceeded the limit of 10 internal redirects`

If you encounter a `500 Internal Server Error` and one of these errors in your `$HOME/logs/frontend/error_website_php.log` log file:

```
[Thu Sep 16 12:00:00 2010] [error] [client 12.345.678.9] Request exceeded the limit of 10 internal re
```

Then it means that Apache was unable to resolve a redirect in your site. Typically, this error can be resolved by *setting RewriteBase* in the application's `.htaccess` file.

24.3.2 Error: *503 Service Unavailable*

A *503 Service Unavailable* error occurs when a Static/CGI/PHP application is unable to fulfill the incoming request. This typically happens because of heavy traffic and too many requests come in during a short period of time. Minimize this error by optimizing the site to spend less time and resources per request and reducing the total number of requests to the site. For example, cache database queries and serve static content from a separate, static-only application.

24.3.3 URLs without a Trailing Slash Redirect to the Wrong Domain

Static/CGI/PHP applications automatically redirect `http://domain/path` to `http://domain/path/` (note trailing slashes) when *path* is a directory within the Static/CGI/PHP application. However, if you've configured a website record with two or more domains, these redirections may behave unexpectedly. For example, a request to `http://original_domain/path` may be redirected to `http://other_domain/path/` .

This unexpected behavior occurs because the Static/CGI/PHP application completes the redirect using an arbitrary domain chosen from the website record. This arbitrarily chosen domain may not be the original domain used in the request. To prevent such cross-domain redirections, please create a separate website record for each domain in use.

24.3.4 Uploaded Files Owned by Apache

In some cases, files uploaded through PHP or WebDAV can become owned by the *apache* user and group. When files are owned by *apache*, they cannot be modified in an SSH session or with SFTP.

To prevent this from happening, the `setgid` flag must be set on the directory where the files are being saved. This will cause files saved in the directory to be owned by the same group as the directory itself—in other words, your user—instead of *apache*'s group.

To set the `setgid` flag for a directory:

1. Open an SSH session to your account.

2. Enter `chmod g+s directory` where directory is the path to the directory where files are owned by *apache*, and press `Enter`.

CHAPTER

TWENTYFIVE

SUBVERSION

Subversion is a version control system. Like other version control systems, Subversion (also known as `svn`) allows you to track changes to files over time, to make it easier to combine multiple authors' changes to files and to maintain a record of "known good" revisions.

25.1 Send Email Notifications with a Post-Commit Hook

You can use Subversion's post-commit hook to execute a script after a complete commit has taken place. In this example, a post-commit sends notification email after each commit to the Subversion repository.

To create a post-commit hook to send an email notification:

1. Create the following script as a file named `post-commit` in the `~/webapps/svn_app/hooks` directory such that:
   - `from_addr` is the address from which the notifications will be sent,
   - `dest_addrN` are the destination addresses for the notification email,
   - `mailbox_name` is the name of the WebFaction mailbox which will be used to send the email, and
   - `mailbox_password` is the password for the WebFaction mailbox which will be used to send the email.

```
#!/usr/local/bin/python2.5

# Settings
mail_server = 'smtp.webfaction.com'
from_email = 'from_addr'
to_email = ['dest_addr1', 'dest_addr2']
mailbox_name = 'mailbox_name'
mailbox_password = 'mailbox_password'

import os
import smtplib
import sys

server = smtplib.SMTP(mail_server)
server.login(mailbox_name, mailbox_password)

repository, revision = sys.argv[1:]

# Get the diff from svnlook
cmd = "svnlook diff %(repository)s -r %(revision)s" % locals()
diff = os.popen(cmd).read()
```

```
    msg = (
    "To: %(to_email)s\r\n"
    "From: %(from_email)s\r\n"
    "Subject: Subversion post-commit: r%(revision)s\r\n"
    "Content-type: text/plain\r\n"
    "\r\n"
    "Repository: %(repository)s\r\n"
    "Revision: %(revision)s\r\n"
    "Diff:\r\n"
    "%(diff)s\r\n"
    )

    msg = msg % locals()

    server.sendmail(from_email, to_email, msg)
    server.quit()
```

2. Open an SSH session.
3. Switch to the `~/webapps/svn_app/hooks` directory.
4. Enter `chmod ogu+x post-commit` and press `Enter`.

Now, every time a new change is committed to the repository, an email will be sent with a diff of the changes committed to the list of recipients.

25.2 Managing Users

Subversion users are managed with `.htpasswd` file in `~/webapps/svn`, where *svn* is the name of the Subversion application.

To add a user or change an existing user's password:

1. Open an SSH session.
2. Switch to the Subversion directory. Enter `cd ~/webapps/svn` and press `Enter`.
3. Enter `htpasswd .htpasswd username`, where *username* is the user, and press `Enter`. A password prompt appears.
4. Enter the new password and press `Enter`.

   See also:

   See *Strengthening Passwords* for important information about choosing passwords.
5. Reenter the new password and press `Enter`.

To delete a user:

1. Open an SSH session.
2. Switch to the Subversion directory. Enter `cd ~/webapps/svn` and press `Enter`.
3. Enter `htpasswd -D .htpasswd username`, where *username* is the the user, and press `Enter`.

25.3 Reusing Usernames and Passwords Between Subversion and Trac

Subversion and Trac use `~/webapps/application/.htpasswd`, where *application* is the name of the application, to store usernames and passwords for authorized users. To use the same usernames and passwords between the two applications, create a symlink from one `.htpasswd` file to the other:

1. Open an SSH session to your account.
2. Delete one of the application's (here called the *secondary application*) `.htpasswd` files. Enter `rm ~/webapps/secondary_application/.htpasswd` and press `Enter`.
3. Create a symlink from the *primary application*'s `.htpasswd` file to the *secondary application*'s `.htpasswd` location. Enter `ln -s ~/webapps/primary_application/.htpasswd ~/webapps/secondary_applicaiton/.htpassw`

25.4 Backing Up and Restoring Subversion Repositories

While WebFaction makes regular backups of your account, it's advisable to create your own backups. You can use Subversion's `svnadmin` tool to create and restore Subversion backups.

See also:

Repository Maintenance from Version Control with Subversion

25.4.1 Creating a Complete Backup

To create a single, complete backup of your entire Subversion repository in its current state:

1. Open an SSH session to your account.
2. Switch to the directory of your Subversion application. Enter `cd ~/webapps/svn`, where *svn* is the name of the Subversion application as it appears in the control panel, and press `Enter`.
3. Enter `svnadmin dump . > dump_file`, where *dump_file* is the path to the file in which to store the backup, and press `Enter`.

Note: To prevent your dump from being stopped for excessive processor utilization (particularly for repositories with many commits or a large total file size), prefix the `svnadmin` command with the `nice` command for deference to other processes:

```
nice -n 19 svnadmin dump . > dump_file
```

25.4.2 Creating an Incremental Backup

Another popular way to back up Subversion repositories is to use an *incremental backup*. With an incremental backup, the repository can be backed up in such a way as to export only the changes made since a particular revision.

To create an incremental backup:

1. Open an SSH session to your account.

2. Switch to the directory of your Subversion application. Enter `cd ~/webapps/svn`, where *svn* is the name of the Subversion application as it appears in the control panel, and press `Enter`.
3. Enter `svnadmin dump --incremental --revision n . > dump_file`, where *n* is the revision number from which to begin backing up and *dump_file* is the path to the file in which to store the backup, and press `Enter`.

   For example, to backup all of the repository activity following revision 100, enter `svnadmin dump --incremental --revision 101 . > backup-101.dump` and press `Enter`.

The most common uses for incremental backups are in a post-commit hook to create a new incremental backup after each commit or in a *cron job* to create new incremental backups on a schedule.

25.4.3 Restoring from Backups

Backups can be restored with the `svnadmin`'s `load` command. To restore a backup:

1. Open an SSH session to your account.
2. Switch to the directory of a new Subversion application. Enter `cd ~/webapps/svn`, where *svn* is the name of the Subversion application as it appears in the control panel, and press `Enter`.
3. Enter `svnadmin load . < dump_file` where *dump_file* is the path to the backup, and press `Enter`.

   If you are restoring incremental backups, begin with the oldest backup first and progress to the newest. For example, suppose there are three incremental backup files, `r0to100.dump`, `r101to200.dump`, and `r201to300.dump`; the backups would be restored like so:

```
$ svnadmin load . < r0to100.dump
$ svnadmin load . < r101to200.dump
$ svnadmin load . < r201to300.dump
```

   Alternatively, restoration can be completed in one command by concatenating the dump files:

```
svnadmin load . <(cat r0to100.dump r101to200.dump r201to300.dump)
```

Note: To prevent your restoration from being stopped for excessive processor utilization (particularly for repositories with many commits or a large total file size), prefix the `svnadmin` command with the `nice` command for deference to other processes:

```
nice -n 19 svnadmin load . < dump_file
```

CHAPTER

TWENTYSIX

TRAC

Trac is an open source project management tool and bug tracker.

26.1 Getting Started with Trac

To set up a Trac application to work with a *Subversion* or *Git* repository, follow the steps in this section.

26.1.1 Create Applications and a Website

First, create a version control repository, Trac application, and website:

1. Create a version control repository application.
   (a) Log in to the control panel.
   (b) Click *Domains / websites* → *Applications*. The list of applications appears.
   (c) Click the *Add new application* button. The *Create a new application* form appears.
   (d) In the *Name* field, enter a name for the version control application.
   (e) In the *App Category* menu, click to select *Git* or *Subversion*.
   (f) If applicable, in the *Machine* menu, click to select a web server.
   (g) If you selected *Git*, in the *Extra info* field, enter a password for the default user.

   See also:

   See *Strengthening Passwords* for important information about choosing passwords.

   If you selected *Subversion*, you may optionally enable anonymous read access. To enable anonymous read access, in the *Extra info* field, enter `anonymous_read`.
   (h) Click the *Save* button. The application is installed and added to the list of applications.
2. Create a Trac application.
   (a) Click the *Add new application* button. The *Create a new application* form appears.
   (b) In the *Name* field, enter a name for the Trac application.
   (c) In the *App Category* menu, click to select *Trac*.
   (d) In the *App Type* menu, click to select a Trac variant. If you selected Subversion for version control, choose a Subversion Trac variant. If you selected Git for version control, choose a Git Trac variant.
   (e) If applicable, in the *Machine* menu, click to select a web server.

(f) In the *Extra info* field, enter the name of the version control repository application.

(g) Click the *Save* button.

3. Create a website to serve the version control repository and Trac applications.

(a) Click *Domains / websites* → *Websites*. The list of websites appears.

(b) Click the *Add new website* button. The *Create a new website* form appears.

(c) In the *Name* field, enter a website name.

(d) If applicable, in the *Machine* menu, click to select the server to host the website.

(e) If you want your site served over an HTTPS connection, click to select *Encrypted website (https)*.

See also:

Secure Sites (HTTPS)

(f) For each domain name you want to use with the website, add it to the list of domains. In the *Domains* field, enter the domain name. Enter one or more domain names. If the domain has not yet been added to the control panel, click the *Create* link that appears at the bottom of the list of domains to add it.

Note: Don't forget to *point new domains to the WebFaction name servers*.

(g) Click *Add an application* → *Reuse an existing application*. The *Reuse an existing web app* form appears.

(h) Click to select the Trac application.

(i) Click the *Save* button. The application is added to the website's list of applications.

(j) Click *Add an application* → *Reuse an existing application*. The *Reuse an existing web app* form appears.

(k) Click to select the version control repository application.

(l) In the *URL* field, enter a URL path for the repository (for example, `/repository`, `/svn`, or `/git`).

(m) Click the *Save* button. The application is added to the website's list of applications.

(n) Click the *Save* button. The website is created and added to the list of websites.

4. Wait two minutes for the changes to take effect.

Now your Trac site is ready to use at the specified domain.

26.1.2 Customizing the Trac Application

Now that your Trac site is working, here are some additional steps for customizing and controlling Trac:

- See *Setting a Logo*, to configure a logo for your Trac application.
- See *Enabling Email Notifications* to enable Trac ticket event notifications.
- See *Managing Users* to learn how to add, modify, and create Trac users.
- See The Trac User and Administration Guide, especially Customizing the Trac Interface (to control Trac's appearance), The Trac Configuration File (to learn about `trac.ini`), and TracAdmin (to use the command-line configuration tool).

26.2 Enabling Email Notifications

Trac can be configured to send email notifications to ticket owners, reporters, and *CC* subscribers that have provided an email address in their preferences. To enable Trac email notifications:

1. Open `~/webapps/trac/conf/trac.ini` in a text editor, where *trac* is the name of the Trac application.
2. Change the mail configuration settings in the `[notification]` section:
   (a) Replace `email_sender = SmtpEmailSender` with `email_sender = SendmailEmailSender`.
   (b) Replace `smtp_enabled = false` with `smtp_enabled = true`.
   (c) Replace `smtp_from = trac@localhost` with `smtp_from = from_address` where *from_address* is the outgoing email address to be used by your Trac application. Many users opt to use a "no-reply" address.
   (d) Replace `smtp_replyto = trac@localhost` with `smtp_replyto = replyto_address` where *replyto_address* is the email address to be used in Trac notification email messages' `Reply-To` header. Many users opt to use a "no-reply" address.
   (e) *Optional*: To have all notifications sent to a specific address (for example, a mailing list or project manager), replace `smtp_always_cc =` with `smtp_always_cc = cc_addresses`, where *cc_addresses* is a comma-separated list of email addresses.
3. Save and close the file.

Now Trac will send email notifications on ticket change events.

See also:

To make additional customizations to your Trac email notifications, please see the Trac wiki page Email Notification of Ticket Changes.

26.3 Enabling reStructuredText

Our Trac environment does not support reStructuredText (RST) by default. To enable reStructuredText, you must install docutils in your home directory or in your Trac application's `lib` directory. To install docutils:

1. Open an SSH session to your account.
2. Install docutils.
   - To install docutils to your home directory, enter `easy_install-2.6 docutils` and press `Enter`.
   - To install docutils to your Trac application's `lib` directory, enter `PYTHONPATH=$HOME/webapps/trac/lib/python2.6 easy_install-2.6 --install-dir=$HOME/w` where *trac* is the name of the Trac application as it appears in the control panel, and press `Enter`.

You can now use reStructuredText markup in Trac.

See also:

Installing Python Packages

26.4 Managing Users

Trac users are managed with `.htpasswd` file in `~/webapps/trac`, where *trac* is the name of the Trac application.

See also:

Alternatively, you may also install the Account Manager Plugin to manage users with a web interface.

To add a user or change an existing user's password:

1. Open an SSH session.
2. Switch to the Trac directory. Enter `cd ~/webapps/trac` and press `Enter`.
3. Enter `htpasswd .htpasswd username`, where *username* is the user account to modify, and press `Enter`. A password prompt appears.

   Note: Trac usernames may not be all uppercase characters.

   ---

4. Enter the new password and press `Enter`.

   See also:

   See *Strengthening Passwords* for important information about choosing passwords.

5. Reenter the new password and press `Enter`.
6. *Optional*: To grant admin privileges to the user, enter `./bin/trac-admin . permission add username TRAC_ADMIN`, where *username* is the user account to modify, and press `Enter`.

   Note: You can also grant this privilege from Trac's Admin interface with an existing admin user:

   (a) Log in to the Trac site.

   (b) Click *Admin*. The *Basic Settings* page appears.

   (c) Click *Permissions*. The *Manage Permissions* page appears.

   (d) In the *Subject* field, enter the username.

   (e) In the *Action* menu, click to select *TRAC_ADMIN*.

   (f) Click the *Add* button.

To delete a user:

1. Open an SSH session.
2. Switch to the Trac directory. Enter `cd ~/webapps/trac` and press `Enter`.
3. Enter `htpasswd -D .htpasswd username`, where *username* is the user to delete, and press `Enter`.

26.5 Disable `.htpasswd` Authentication

By default, Trac applications installed with the control panel use basic access authentication to control access. To disable authentication:

1. Open an SSH session to your account.

2. Switch to the Trac application directory. Enter `cd ~/webapps/trac`, where *trac* is the name of the Trac application as it appears on the control panel, and press `Enter`.
3. Enter `echo -e "Satisfy Any\nAllow from all" > .htaccess` and press `Enter`.

26.6 Changing the Git Repository Path

For Trac applications that use Git for version control, you can use a different Git repository by editing the value of `repository_dir` in `trac.ini`. To edit the value of `repository_dir`:

1. Open your Trac application's `trac.ini` file in a text editor. The full path to the file is `~/webapps/app/conf/trac.ini`, where *app* is the name of your Trac application.
2. Find the `repository_dir` key in the `[trac]` section of the file. It will look something like this:

```
repository_dir = /home/username/webapps/mygit/repos/proj.git
```

3. Replace the existing path with an absolute path to the root of a Git repository.
4. Save and close the file.

The Trac application will now use the repository found at the specified path.

Note: In the control panel, the Trac application's *Extra info* field will continue to show the name of the Git application used when the Trac application was originally installed.

26.7 Setting a Logo

By default, new installations of Trac do not have a working logo. To set up a logo for your Trac application:

1. Upload your logo file to `~/webapps/trac/htdocs`, where *trac* is the name of your Trac application.
2. Open `~/webapps/trac/conf/trac.ini` in a text editor.
3. Change the logo settings in the `[header_logo]` section:
   (a) Replace `src = site/your_project_logo.png` with `src = site/filename` where *filename* is the name of the logo file (for example, `project_logo.png`).
   (b) Replace `height = -1` with `height = pixels`, where *pixels* is the height of the logo in pixels.
   (c) Replace `width = -1` with `width = pixels`, where *pixels* is the width of the logo in pixels.
   (d) Replace `alt = (please configure the [header_logo] section in trac.ini)` with `alt = text`, where *text* is appropriate alternative text for your logo.
   (e) Replace `link =` with `link = dest`, where *dest* is the destination of the logo hyperlink. For example, to have the logo link return to the root of the domain, enter `link = /`.
4. Save and close the file.

Your logo will now appear at the top of Trac pages.

26.8 Setting Trac's Default Time Zone

1. *Open an SSH session to your account*.
2. Open `$HOME/webapps/trac_app/conf/trac.ini` in a text editor, where *trac_app* is the name of the Trac application.
3. In the `[trac]` section, insert a new line containing `default_timezone = GMT offset`, where *offset* is the hours and minutes from Greenwich Mean Time in the form `±XX:YY`. For example, the offset for Eastern Daylight Time is `-04:00` and the offset for Indian Standard Time is `+05:30`.

   Note: An offset is not daylight saving time-aware. If pytz is installed for the Trac application, then you may substitute *offset* with a daylight saving time-aware `Region/City Name`-style time zone name from the IANA Time Zone Database. For example, the United States Eastern time zone name is `America/New_York` and the Indian Standard Time name is `Asia/Kolkata`.

   To install pytz for the Trac application:

   (a) *Open an SSH session to your account*.

   (b) Switch to the Trac application directory. Enter `cd $HOME/webapps/trac_app/` and press `Enter`.

   (c) Install pytz. Enter `PYTHONPATH=$PWD/lib/python2.6 easy_install-2.6 -d $PWD/lib/python2.6 -s $PWD/bin p` and press `Enter`.

   ---

4. Save and close the file.

Dates and times are now shown in the default time zone for logged-out users and logged-in users who have not selected a time zone in their preferences.

26.9 Upgrade a Trac Application

To upgrade an existing Trac application to the newest version:

1. Open an SSH session to your account.
2. Switch to the Trac application directory. Enter `cd ~/webapps/trac/`, where *trac* is the name of the Trac application, and press `Enter`.
3. Get the latest version of Trac. Enter `PYTHONPATH=$PWD/lib/python2.6 easy_install-2.6 -Z -d $PWD/lib/python2.6 -s $PWD/bin py` where *version* is the Trac version number to which you're upgrading, and press `Enter`.
4. Upgrade the Trac database. Enter `./bin/trac-admin . upgrade` and press `Enter`.
5. Upgrade the Trac wiki pages. Enter `./bin/trac-admin . wiki upgrade` and press `Enter`.
6. Upgrade Trac's image and style resources. Enter `ln -f -s $PWD/lib/python2.6/Trac-version-py2.6.egg/trac/htdocs ./` and press `Enter`.

Now Trac is running the latest version.

Note: The control panel shows the originally installed version number for the Trac application. The actual version number appears in Trac's footer as *Powered by Trac 0.XY*.

CHAPTER

TWENTYSEVEN

WEBDAV

WebDAV applications expose a directory's contents (including subdirectories) to a controlled set of users over HTTP.

There are two kinds of WebDAV applications available through the WebFaction control panel: a standard WebDAV application and a symlink application. The standard application creates a directory in your `~/webapps` directory. The symlink application does not create its own directory; instead, you must provide the path to an existing directory in the application installer's *Extra info* field.

Standard and symlink WebDAV applications can only be accessed with a valid username and password. Valid usernames and passwords are stored in a `.htpasswd` file in the WebDAV application's directory (if the file doesn't already exist, the installer creates one automatically).

Note: WebDAV applications use HTTP basic access authentication. Basic access authentication is unsupported natively by Windows 7. To use a WebDAV application with Windows 7, use a stand-alone WebDAV client such as Cyberduck.

Warning: WebDAV applications can only be mounted on the root URL path of a domain (`/`). We recommend that you *create a new subdomain* for use with your WebDAV applications.

27.1 Creating a WebDAV Application

To create a WebDAV application:

1. Create the WebDAV application.
   (a) Log in to the control panel.
   (b) Click *Domains / websites* → *Applications*. The list of applications appears.
   (c) Click the *Add new application* button. The *Create a new application* form appears.
   (d) In the *Name* field, enter a name for the application.
   (e) In the *App Category* menu, click to select *WebDav*.
   (f) Choose the type of WebDAV application to create.

   If you want to serve the contents of a new directory in your `~/webapps` directory, click to select *WebDav*.

   If you want to serve the contents of an existing directory, click to select *WebDav Symlink*.
   (g) If applicable, in the *Machine* menu, click to select a web server.

(h) If you selected *WebDav Symlink*, then enter the full path to the existing directory in the *Extra info* field.

For example, if you want to serve `~/webdav`, then enter `/home/username/webdav` in the *Extra info* field.

> **Warning:** If you do not enter a valid path in the *Extra info* field, you must delete the application, then create a new application with a valid path.

If you selected a standard WebDAV application, then continue to the next step.

(i) Click the *Save* button. The application is installed and added to the list of applications.

2. *Add usernames and passwords*.

Don't forget to add the WebDAV application to a *new or existing website* to make the application available on the web.

27.2 Adding and Removing WebDAV Users

Use the *htpasswd* utility to add or remove WebDAV users.

27.2.1 Adding Users

To add a WebDAV user:

1. *Open an SSH session to your account*.
2. Switch to the WebDAV directory. Enter `cd webdav`, where *webdav* is the path to the WebDAV directory, and press `Enter`.

   For example, if you installed a standard WebDAV application, then enter `cd $HOME/webapps/appname`, where *appname* is the name of the application as it appears on the control panel, and press `Enter`.
3. Enter `htpasswd .htpasswd user`, where *user* is the new username, and press `Enter`. A prompt appears to enter the password.
4. Enter a password and press `Enter`. A prompt appears to reenter the password.

   See also:

   See *Strengthening Passwords* for important information about choosing passwords.
5. Reenter the password and press `Enter`.

The user is added.

27.2.2 Removing Users

To remove a WebDAV user:

1. *Open an SSH session to your account*.
2. Switch to the WebDAV directory. Enter `cd webdav`, where *webdav* is the path to the WebDAV directory, and press `Enter`.

   For example, if you installed a standard WebDAV application, then enter `cd $HOME/webapps/appname`, where *appname* is the name of the application as it appears on the control panel, and press `Enter`.

3. Enter `htpasswd -D .htpasswd user`, where *user* is the username to be deleted, and press `Enter`. A confirmation message appears.

CHAPTER

TWENTYEIGHT

WEBSTATS

WebFaction supports two website statistics tools, *AWStats* and *Webalizer*. These tools parse your website log files and generate reports about visits.

28.1 AWStats

AWStats is an open source log analyzer.

28.1.1 Installing AWStats

To set up AWStats, follow these steps for each website you want to monitor:

1. Log in to the control panel.
2. Click *Domains / websites* → *Applications*. The list of applications appears.
3. Click the *Add new application* button. The *Create a new application* form appears.
4. In the *Name* field, enter a name for the application.
5. In the *App Category* menu, click to select *AWStats*.
6. If applicable, in the *Machine* menu, click to select a web server.
7. In the *Extra info* field, enter the name of the website record for which you want to generate reports. For example, if you have a website record named `myblog`, enter `myblog` in the *Extra info* field.
8. Click the *Save* button.

The application is installed and added to the list of applications. Reports are generated hourly, and the first results appear in less than one hour. Add the AWStats application to a *website record* to access the reports on the web.

See also:

By default, access to AWStats is unrestricted. To password protect your AWStats application, see *Password Protecting a Directory with a Static/CGI/PHP App*.

28.2 Webalizer

Webalizer is an open source log analyzer.

28.2.1 Installing Webalizer

To set up Webalizer, follow these steps for each website you want to monitor:

1. Log in to the control panel.
2. Click *Domains / websites* → *Websites*. The list of websites appears.
3. Click the name of the website for which you want to generate statistics. The site's details appear.
4. Click *Add an application* → *Create a new application*. The *Create a new web app* form appears.
5. In the *Name* field, enter a name for the application.
6. In the *App category* menu, click to select *Webalizer*.
7. In the *URL* field, enter a URL path (for example, `/stats`).
8. Click the *Save* button. The application is installed and added to website's list of applications.
9. Click the *Save* button.

Webalizer soon generates statistics. Access the reports at the domain and URL path you selected.

See also:

By default, access to Webalizer is unrestricted. To password protect your Webalizer application, see *Password Protecting a Directory with a Static/CGI/PHP App*.

CHAPTER

TWENTYNINE

WORDPRESS

WordPress is a blog publishing application.

29.1 Using WordPress

29.1.1 Logging in to WordPress

The first time you log in to WordPress, you must use the username and password generated by the control panel. For WordPress versions 4.2.2 and later, use your WebFaction account name and the generated password. For earlier versions of WordPress, use the `admin` username and the generated password.

To log in:

1. Get the administrative user's password.
   (a) Log in to the control panel.
   (b) Click *Domains / websites* → *Applications*. The list of applications appears.
   (c) Click the name of the WordPress application.
   (d) Make a note of the password in the *Extra info* field.
2. Log in to WordPress.
   (a) Open the WordPress *Log In* page at `http://domain_path/wp-login.php` where *domain_path* is the domain and URL path where the WordPress application is mounted.
   (b) In the *Username* field, enter your username. For WordPress versions 4.2.2 and later, enter your WebFaction account name. For earlier versions of WordPress, enter `admin`.
   (c) In the *Password* field, enter the password from the control panel.
   (d) Click the *Log In* button. The WordPress *Dashboard* appears.

Now you can control all of the settings related to your WordPress blog. You can also change the admin password by following the prompt at the top of the *Dashboard*.

Note: Any change to the WordPress admin password will *not* be reflected on the control panel.

See also:

See *Strengthening Passwords* for important information about choosing passwords. and the WordPress Codex documentation site

29.1.2 Backing Up and Restoring WordPress Content

Backing Up

To back up the content of your WordPress blog to a SQL dump:

1. Log in to the control panel.
2. Click *Databases* → *Databases*
3. In the row of the WordPress's application's database, click *phpMyAdmin*. The phpMyAdmin login page appears.
4. In the *Username* field, enter the name of the database user for the WordPress application.

   Note: To find the WordPress application's database user, open `~/webapps/wp/wp-config.php` in a text editor, where *wp* is the name of the WordPress application, and look for a line like this:

   ```
   define('DB_USER', 'username_app_');
   ```

   The second parameter is database username (`username_app_` , in this example).

5. In the *Password* field, enter the database password.
6. Click the *Go* button. The phpMyAdmin site appears.
7. Click the *Databases* tab. A list of databases appears.
8. Click on the name of the WordPress database. A list of tables appears.

   Note: To find the WordPress application's database name, open `~/webapps/wp/wp-config.php` in a text editor, where *wp* is the name of the WordPress application, and look for a line like this:

   ```
   define('DB_NAME', 'username_app_');
   ```

   The second parameter is the name of the database (`username_app_` , in this example).

9. Click the *Export* tab. The dump form appears.
10. Click to select *Add DROP TABLE / VIEW / PROCEDURE / FUNCTION*.
11. Click the *Go* button. Your browser will prompt or begin the download of the SQL file.

Restoring

To restore the content of your WordPress blog from a SQL dump:

1. Log in to the control panel.
2. Click *Databases* → *Databases*
3. In the row of the WordPress's application's database, click *phpMyAdmin*. The phpMyAdmin login page appears.
4. In the *Username* field, enter the name of the database user for the WordPress application.

   Note: To find the WordPress application's database user, open `~/webapps/wp/wp-config.php` in a text editor, where *wp* is the name of the WordPress application, and look for a line like this:

   ```
   define('DB_USER', 'username_app_');
   ```

The second parameter is database username (`username_app_` , in this example).

5. In the *Password* field, enter the database password.
6. Click the *Go* button. The phpMyAdmin site appears.
7. Click the *Databases* tab. A list of databases appears.
8. Click on the name of the WordPress database. A list of tables appears.

Note: To find the WordPress application's database name, open `~/webapps/wp/wp-config.php` in a text editor, where *wp* is the name of the WordPress application, and look for a line like this:

```
define('DB_NAME', 'username_app_');
```

The second parameter is the name of the database (`username_app_` , in this example).

9. Click the *Import* tab. The import form appears.
10. Click the *Choose file* button. Your system's file selection dialog appears. Select the SQL file to be restored and dismiss the dialog.
11. Click the *Go* button. The file is processed and a confirmation message appears.

29.1.3 Sending Email from WordPress

To configure WordPress to be able to send messages for comment notifications, password recovery, and other features:

1. *Log in to WordPress*.
2. Open the *General Settings* page. Click *Settings* → *General*. The *General Settings* page appears.
3. In the *E-mail address* field, enter an outgoing email address for your WordPress site.
4. Click the *Save changes* button. The *Settings saved* notification appears at the top of the page.

29.1.4 Upgrading WordPress

You can upgrade WordPress installations in-place. By default, WordPress 2.7 and newer updates itself to the most recent minor release. For example, WordPress 2.8 updates to WordPress 2.8.1 without user intervention. To upgrade for major releases (for example, to upgrade from 2.7 to 2.8), you must follow one of the upgrade procedures in this section.

Note: Upgrading WordPress does not update the version number listed on the WebFaction control panel.

Upgrading Automatically

Note: WordPress can only be upgraded automatically with versions 2.7 or higher. Please see *Upgrading Manually* for other versions of WordPress.

To upgrade a WordPress installation automatically:

1. *Log in to WordPress*.
2. Click *Tools*. The *Tools* menu expands.

3. Click *Upgrade*. The *Upgrade WordPress* page appears.
4. Click the *Upgrade Automatically* button. The upgrade is downloaded and installed.

Upgrading Manually

Note: For more details on upgrading WordPress manually, please see the WordPress Codex documentation Upgrading WordPress Extended.

To upgrade a WordPress installation manually:

1. *Back up the WordPress database contents*.
2. Back up the WordPress installation files.
   (a) *Open an SSH session to your account*.
   (b) Switch to the WordPress application directory. Enter `cd ~/webapps/wp`, where *wp* is the name of the WordPress application as it appears on the control panel, and press `Enter`.
   (c) Create a backup archive in your home directory. Enter `tar -zcvf $HOME/wp-backup-$(date +%Y-%m-%d).tar.gz .` and press `Enter`. The files are compressed into a new archive in your home directory, `wp-backup-date.tar.gz`, where *date* is today's date.
3. Deactivate plugins.
   (a) *Log in to WordPress*.
   (b) Click *Plugins*. The *Plugins* table appears.
   (c) Select all of the plugins. Click the checkbox beside *Plugin* in the table header row.
   (d) In the *Bulk Actions* menu, click to select *Deactivate*.
   (e) Click the *Apply* button.
4. Prepare the WordPress directory for new files.
   (a) *Open an SSH session to your account*.
   (b) Switch to the WordPress application directory. Enter `cd ~/webapps/wp` and press `Enter`.
   (c) Create a directory to store existing files needed after the upgrade. Enter `mkdir -p upgrade` and press `Enter`.
   (d) Copy the required files to the upgrade directory.
      i. Enter `cp wp-config.php .htaccess upgrade` and press `Enter`.
      ii. Enter `cp -R wp-content upgrade` and press `Enter`.
   (e) Copy any additional WordPress files you have modified to the upgrade directory (for example, `index.php`). Enter `cp filenames upgrade`, where *filenames* are a space-separated list of files, and press `Enter`.
   (f) Remove the existing WordPress files.
      i. Enter `rm wp*.php .htaccess license.txt readme.html xmlrpc.php` and press `Enter`.
      ii. Enter `rm -r wp-admin wp-includes` and press `Enter`.

5. Install the new WordPress files.
   (a) Download the most recent WordPress release. Enter `wget http://wordpress.org/latest.tar.gz` and press `Enter`. A new file, `latest.tar.gz`, is created in the current directory.
   (b) Extract the archive. Enter `tar -xvf latest.tar.gz` and press `Enter`. The contents of the archive are extracted to a new directory, `wordpress`.
   (c) Copy the contents of `wordpress` to the current directory. Enter `cp -rp wordpress/* .` and press `Enter`.
   (d) Restore `wp-config.php`. Enter `cp upgrade/wp-config.php .` and press `Enter`.
6. Upgrade the WordPress database.
   (a) *Log in to WordPress*. *Database Upgrade Required* appears.
   (b) Click the *Update WordPress Database* button. After the update is finished, the *Update Complete* page appears.
   (c) Click the *Continue* button. The WordPress login form appears.
7. Restore any additional files. If you copied any additional files (such as `index.php`) to the upgrade directory, restore them to their original locations now.

   If you modified any of the default WordPress themes or plugins, restore those modifications from `upgrade/wp-content` to `wp-content` as well.
8. Remove the WordPress installation files.
   (a) *Open an SSH session to your account*.
   (b) Switch to the WordPress application directory. Enter `cd ~/webapps/wp` and press `Enter`.
   (c) Enter `rm latest.tar.gz` and press `Enter`.
   (d) Enter `rm -r wordpress/` and press `Enter`.
   (e) Remove the files for the previous version of WordPress. Enter `rm -r ./upgrade` and press `Enter`.
9. Reactivate plugins.
   (a) *Log in to WordPress*.
   (b) Click *Plugins*. The *Plugins* table appears.
   (c) Select all of the plugins. Click the checkbox beside *Plugin* in the table header row.
   (d) In the *Bulk Actions* menu, click to select *Activate*.
   (e) Click the *Apply* button.

The WordPress upgrade is now complete.

29.2 Advanced WordPress

WordPress is a powerful web publishing platform. You can use some of WordPress's more advanced features or tweak your WordPress deployment for better performance.

See also:

WordPress is a PHP-based application. Please see *PHP* and *Static Files, CGI Scripts, and PHP Pages* for additional documentation.

29.2.1 Serving Uploads Faster

To improve performance for your WordPress application, create a symbolic link application to serve your WordPress application's media directly. To create and configure the symlink application:

1. Log in to the control panel.
2. Click *Domains / websites* → *Websites*. The list of websites appears.
3. Click the name of the website that contains the WordPress application.
4. Click *Add an application* → *Create a new application*. The *Create a new web app* form appears.
5. In the *Name* field, enter a name for the application.
6. In the *App category* field, click to select *Symbolic link*.
7. In the *App type* field, click to select *Symbolic link to static-only app*.
8. In the *Extra info* field, enter `/home/username/webapps/wordpress/wp-content/uploads/`, where *username* is your username and *wordpress* is the name of the WordPress application.
9. In the *URL* field, enter `wp-content/uploads` for WordPress applications mounted at the root of a domain (`/`) or enter `path/wp-content/uploads` where *path* is the URL path for the WordPress application.
10. Click the *Save* button. The application is installed and added to website's list of applications.
11. Click the *Save* button. The changes to the website are saved.

Now the files in `wp-content/uploads` are served by your server's front-end process directly, bypassing the PHP interpreter.

29.2.2 Caching WordPress

To help your WordPress application handle a larger number of visits, you can automatically cache many pages of your WordPress application using the WP Super Cache plugin. To install WP Super Cache:

1. *Log in to WordPress*.
2. Click *Plugins* → *Add New*. The *Install Plugins* page appears.
3. In the search field, enter `WP Super Cache` and click *Search Plugins*. The *Search Results* table appears.
4. Under *WP Super Cache*, click *Install Now*. The *Installing Plugin* page appears.
5. Click *Activate Plugin*. The *Plugins* page appears with a table of installed plugins.
6. Under *WP Super Cache*, click *Settings*.
7. If *Permlink Structure Error* appears:
   (a) Click *Permalinks Options Page*. The *Permalink Settings* page appears.
   (b) In the *Common settings* section, select a permalink URL scheme other than *Default*.
   (c) Click the *Save Changes* button. A *Permalink structure updated* notification appears at the top of the page.
   (d) In the sidebar, click *Settings* → *WP Super Cache*. The *WP Super Cache Settings* page appears.
8. Click to select *Caching On (Recommended)*.

9. Click the *Update Status* button.

Now, static pages will be served to your users instead of dynamically generated pages where possible.

29.2.3 Using Multisite

As of WordPress version 3.0, WordPress MU has become part of the core WordPress software. With the Multisite features of WordPress 3.0, you can create a network of WordPress blogs from a single installation of WordPress using subpaths (for example, `example.com/blog1`, `example.com/blog2`, etc.) or subdomains (for example, `blog1.example.com`, `blog2.example.com`, etc.). To use the Multisite feature of WordPress to create a blog network:

1. Create a WordPress site and application.
   (a) Log in to the control panel.
   (b) Click *Domains / websites* → *Websites*. The list of websites appears.
   (c) Click the *Add new website* button. The *Create a new website* form appears.
   (d) In the *Name* field, enter a website name.
   (e) For each domain name you want to use with the website, add it to the list of domains. In the *Domains* field, enter the domain name. Enter one or more domain names. If the domain has not yet been added to the control panel, click the *Create* link that appears at the bottom of the list of domains to add it.

   Note: If you're planning to use subdomains for multisite blogs (like `marysblog.example.com` and `johnsblog.example.com`), add a wildcard subdomain for the domain name (or names) for the site.

   For example, if you entered `example.com` in the *Domains* field, also add `*.example.com`.

   Note: Don't forget to *point new domains to the WebFaction name servers*.

   (f) Click *Add an application* → *Create a new application*. The *Create a new web app* form appears.
   (g) In the *Name* field, enter a name for the WordPress application.
   (h) In the *App category* menu, click to select *WordPress*.
   (i) Click the *Save* button. The application is installed and added to website's list of applications.
   (j) Click the *Save* button. The website is created and added to the list of websites.
2. Wait two minutes while the website record changes take effect.
3. Enable WordPress Multisite.
   (a) Open `~/webapps/app/wp-config.php` in a text editor.
   (b) On a new line beneath `<?php`, add `define('WP_ALLOW_MULTISITE', true);`.
   (c) Save and close the file.
4. Configure WordPress Multisite.
   (a) *Log in to WordPress*.
   (b) In the menu on the left, click *Tools* → *Network*. The *Create a Network of WordPress Sites* page appears.
   (c) Click to select the address type you would like to use for blogs on the WordPress network:

- Choose *Sub-domains* for network blogs to use subdomains to form URLs (like `marysblog.example.com` and `johnsblog.example.com`).
- Choose *Sub-directories* for network blogs to use subdirectories to form URLs (like `example.com/marysblog` and `example.com/johnsblog`).

(d) In the *Network Details* section, adjust any preferences as desired.

(e) Click the *Install* button. The *Enabling the Network* page appears.

(f) The *Enabling the Network* page provides several important steps to complete the setup. Complete the directions provided before continuing.

(g) Log out of the WordPress site.

You can now make and manage new Multisite blogs: login to the WordPress Dashboard and use the *Super Admin* module in the menu.

29.3 Troubleshooting WordPress

See also:

WordPress is a PHP-based application. Please see *PHP* and *Static Files, CGI Scripts, and PHP Pages* for additional documentation.

29.3.1 Error: `Error establishing a database connection`

When WordPress is unable to connect to the MySQL database, all WordPress pages show this error: `Error establishing a database connection`. One common cause of this error is misconfigured database settings.

To verify your WordPress application's database settings:

1. Verify the settings in `wp-config.php`.

   (a) Open `~/webapps/app/wp-config.php` in a text editor, where *app* is the name of the WordPress application.

   (b) Verify the database name. The line starting `define('DB_NAME',` ends with `'app_wp');` where *app* is the name of the application. If this is not set correctly, edit the line to contain the correct value.

   Note: Some WordPress applications created before 2013 may use a database name like `username_app_wp`, where *username* is your account name.

   (c) Verify the database user. The line starting `define('DB_USER',` should end with `'app_wp');`. If this is not set correctly, edit the line to contain the correct value.

   Note: Some WordPress applications created before 2013 may use a database user like `username_app_wp`, where *username* is your account name.

   (d) Verify the database password. The line starting `define('DB_PASSWORD',` should end with `'pass');`, where *pass* is the password for the WordPress application's MySQL database. If this is not set correctly, edit the line to contain the correct value.

   If you do not already know it, you can find the correct database password using the WebFaction control panel:

i. Log in to the WebFaction control panel.

ii. Click *Domains / websites* → *Applications*. The list of applications appears.

iii. Click the name of your WordPress application in the list. The application's details appear. The database password is in *Extra info* section.

(e) Verify the database hostname. The line starting `define('DB_HOST',` should end with `'localhost');`. If this is not set correctly, edit the line to contain the correct value.

(f) Save and close the file.

2. Verify the database user's password is set correctly.

(a) Log in to the control panel.

(b) Click *Domains / websites* → *Applications*. The list of applications appears.

(c) Click the name of your WordPress application in the list. The application's details appear.

(d) Make a note of the password in the *Extra info* section.

(e) Click *Databases* → *Database Users*. The list of database users appears.

(f) In the list, click the user named `app_wp`.

Note: Some WordPress applications created before 2013 may use a database user like `username_app_wp`, where *username* is your account name.

The user's details appear.

(g) In the *Password* field, click *Change*. A *Password* field and *Confirm password* field appear.

(h) In the *Password* field, enter the password from the *Extra info* field you made a note of previously.

(i) In the *Confirm Password* field, reenter the password.

(j) Click the *Save* button. The password is changed and a confirmation message appears.

Your WordPress application's database should now be configured correctly.

29.3.2 WordPress Links Point to the Wrong Domain

Some changes to a site record with an attached WordPress application will cause WordPress's internal links to point to addresses on an incorrect domain. To fix the problem, the WordPress application must be removed and re-added to the website record.

See also:

For more information about changing a WordPress site's domain, please see the WordPress documentation page Changing The Site URL.

To remove and re-add WordPress to a site record:

1. Log in to the control panel.
2. Click *Domains / websites* → *Websites*. The list of websites appears.
3. Click the name of the site that contains the WordPress application.
4. In the *Contents* list, click the *x* in the WordPress application's row. The application is removed from the list.
5. Click the *Save* button. A confirmation appears.
6. Click the name of the site that contains the WordPress application.

7. Click *Add an application → Reuse an existing application*. The *Reuse an existing web app* form appears.
8. Select the WordPress application.
9. In the *URL* field, enter the URL path your WordPress site (for example, `/blog`, or leave blank for the root of the domain).

   Note: The first application added to a website is assigned to the root URL path (`/`).

10. Click the *Save* button. The application is added to the website's list of applications.
11. Click the *Save* button. A confirmation appears.

The WordPress application will now use the correct domain.

Symbols

403 Forbidden, 11
500 Internal Server Error, 88, 134
502 Bad Gateway, 11
503 Service Unavailable, 137
504 Gateway Timeout, 11

A

Action Mailer, 120
AWStats, 155
    Installation, 155

B

Bazaar, 15
    Installation, 17
    Loggerhead, 17
Building (from source), 12

C

Cache-Control, 127
Capistrano, 117
Compiling (from source), 12
CPAN, 77
cpanminus, 77
Cron, 9
Custom app (listening on port), 19
    Automatically restarting, 21
    Creating, 21

D

Django, 22
    Admin password, 42
    ALLOWED_HOSTS, 29
    Database configuration, 32
    DEBUG, 36
    Django Debug Toolbar, 36
    Email, 30
    FORCE_SCRIPT_NAME, 31
    ImportError, 39
    Internal Server Error, 40
    manage.py, 38
    Memcached, 29
    Memory, 41
    Password protection, 31
    REMOTE_ADDR, 43
    Restart, 32
    Start up time, 41
    Static media, 28
    TemplateDoesNotExist, 40
    Time zone, 30
    TIME_ZONE, 30
    Troubleshooting, 35
    Trunk, 35
    Tutorial, 23
    Upgrade, 34
    USE_TZ, 30
Drupal, 43
    Back up, 45
    Cache, 47
    Clean URLs, 46
    Email, 47
    Sendmail, 47
    Static media, 47
    Troubleshooting, 48
    Update, 45
    Uploads, 47

E

Expires, 127
expires max, 127

F

FastCGI, 83
File permissions, 7
FORCE_SCRIPT_NAME, 31

G

Git, 48
    Anonymous read access, 51
    HTTPS, 52
    Installation, 49
    Repository cloning, 50
    Repository creation, 49
    RPC failed, 52

Troubleshooting, 52
Users, 51

H

HTML, 86
HTTP Errors
403, 11
500, 40, 88, 134
502, 11
503, 137
504, 11
HTTPS
Redirect, 131

I

Illegal option, 136
Invalid command, 136

J

Joomla, 53
Email, 55, 56
Sendmail, 55
SMTP, 56
Upgrade, 55

L

Log in, 157
Loggerhead, 17
Logs, 3
Front-end, 3
User, 4

M

max-age, 127
Memcached, 56
Django, 29
Shut down, 57
Start up, 57
Memory usage, 4
Mercurial, 57
HgWeb, 59
Installation, 59
Publishing, 59
mod_wsgi, 64
Memory, 65
Python virtual environments, 66
Start up time, 65
MongoDB, 67
Installation, 69
Movable Type, 70
Email, 71, 72
Sendmail, 71
SMTP, 72
MySQL, 89
Databases, 90
Users, 90

N

Node.js, 73
Not Found, 11

P

PEAR, 85
Perl, 76
CPAN, 77
cpanminus, 77
Installing modules, 77
Perlbrew, 78
Permissions, 7
PHP, 79
Configuration, 81–83
DOCUMENT_ROOT, 83
Email, 86
FastCGI, 83
HTML, 86
PEAR, 85
php.ini, 81–83
Sendmail, 86
Settings, 81–83
Time zone, 87
Troubleshooting, 87
Upgrade version, 82
Upload limit, 82
PostgreSQL, 92
Databases, 93
Users, 93
Premature end of script headers, 135
Private database instances, 88
Configuration, 95
MySQL, 89
PostgreSQL, 92
ps, 4
Pyramid, 97
Deployment, 99
Restart, 100
Start, 100
Stop, 100
URL prefix, 100
Python, 101
Alias, 103
easy_install, 104
Eggs, 104
ImportError
ImportError, 106
Pip, 105
PYTHONPATH, 107
Search path, 103
setup.py, 105

Python Enhancement Proposals
PEP 333, 65
PYTHONPATH, 107

R

Rails
Secret key, 122
Redmine, 122
Email, 123
REMOTE_ADDR, 43
RPC failed, 52
Ruby, 109
Gems, 111
Ruby on Rails, 112
Action Mailer, 120
Capistrano, 117
Databases, 119
Deployment, 117
Installation, 113
Troubleshooting, 120
Upgrading, 118
RubyGems, 111

S

Site not configured, 10
SQLite, 124
Installation, 125
Static-only, 127
Static/CGI/PHP, 128
AddHandler, 129
Block a hostname, 128
Block an IP, 128
File handlers, 129
Hide configuration files, 130
HTTPS, 131
MIME types, 128
Password protection, 130
Redirect, 132
RewriteBase, 134
Server Side Includes, 130
SSI, 130
Subdirectory, 133
Symbolic link, 133
Troubleshooting, 134
Subversion, 138
Back up, 141
Email, 139
Post-commit hook, 139
Restore, 141
Users, 140

T

Time zone
Django, 30
PHP, 87
Trac, 147
Trac, 142
Email, 144
Getting started, 143
Logo, 147
reStructuredText, 145
Time zone, 147
Upgrade, 148
Users, 145

V

Version control
Bazaar, 15
Git, 48

W

Webalizer, 155
Installation, 155
WebDAV, 149
Setup, 151
Users, 152
WebSocket, 21
Webstats, 153
WordPress, 156, 157
Back up, 158
Caching, 162
Domain, 165
Email, 159
Error establishing a database connection, 164
MU, 163
Multisite, 163
Restore, 158
Sendmail, 159
Static media, 162
Troubleshooting, 164
Upgrade, 159
Uploads, 162
WP Super Cache, 162
WP Super Cache, 162