# Apache/PHP/Drupal settings:
#
# Protect files and directories from prying eyes.
<FilesMatch "\.\(engine|incl install|makemodule|profile|polish|*sql|them|twig|tpl\(\./\.*\)\?.\(\.(engine|inc|install|maked|module|profile|polish|sql|theme|twig|tpl\(\.*\)\?)|xtmpl|yml\)\(\.*|\.(\w|.)\(\.*\)\?|^\(\w|\.*\)\?\)
   <IfModule mod_authz_core\.c>
      Require all denied
   </IfModule>
   <IfModule ! mod_authz_core\.c>
      Order allow,deny
   </IfModule>
</FilesMatch>

# Don't show directory listings for URLs which map to a directory.
Options -Indexes

# Follow symbolic links in this directory.
Options +FollowSymLinks

# Set the default handler.
DirectoryIndex index\.php index\.html index\.htm

# Add correct encoding for SVGZ.
AddType image/svg+xml svg svgz
AddEncoding gzip svgz

# Most of the following PHP settings cannot be changed at runtime. See
# sites/default/default.settings.php and
# Drupal\Core\DrupalKernel::bootEnvironment() for settings that can be
# changed at runtime.

# PHP 5, Apache 1 and 2.
<IfModule mod_php5\.c>
   php_value assert\.active 0
   php_flag session\.auto_start off
   php_value mbstring\.http_input pass
   php_value mbstring\.http_output pass
   php_flag mbstring\.encoding_translation off
</IfModule>
# PHP 5.6 has deprecated $HTTP_RAW_POST_DATA and produces warnings if this is not set.

```php
php_value always_populate_raw_post_data -1
```

`</IfModule>`

# Requires mod_expires to be enabled.

```html
<!-- IfModule mod_expires.c -->

# Enable expirations.
ExpiresActive On

# Cache all files for 2 weeks after access (A).
ExpiresDefault A1209600

</FilesMatch>
```

`</IfModule>`

# Set a fallback resource if mod_rewrite is not enabled. This allows Drupal to work without clean URLs. This requires Apache version >= 2.2.16. If Drupal is not accessed by the top level URL (i.e.: http://example.com/drupal/ instead of http://example.com/), the path to index.php will need to be adjusted.

```html
<!-- IfModule !mod_rewrite.c -->

FallbackResource /index.php

</IfModule>
```

# Various rewrite rules.

```html
<!-- IfModule mod_rewrite.c -->

RewriteEngine on

# Set "protossl" to "s" if we were accessed via https://. This is used later if you enable "www." stripping or enforcement, in order to ensure that you don't bounce between http and https.
RewriteRule ^ - [E=protossl]
RewriteCond %{HTTPS} on
```

```html
RewriteRule " ^ " ^ https: //
```

`</IfModule>`
# Make sure Authorization HTTP header is available to PHP
# even when running as CGI or FastCGI.

```
RewriteRule `^-` [E=HTTP_AUTHORIZATION: %{HTTP: Authorization}]
```

# Block access to "hidden" directories whose names begin with a period. This
# includes directories used by version control systems such as Subversion or
# Git to store control files. Files whose names begin with a period, as well
# as the control files used by CVS, are protected by the FilesMatch directive
# above.

# NOTE: This only works when mod_rewrite is loaded. Without mod_rewrite, it is
# not possible to block access to entire directories from .htaccess because
# <DirectoryMatch> is not allowed here.

# If you do not have mod_rewrite installed, you should remove these
# directories from your webroot or otherwise protect them from being
# downloaded.

```
RewriteRule "(\^//)\." - [F]
```

# If your site can be accessed both with and without the 'www.' prefix, you
# can use one of the following settings to redirect users to your preferred
# URL, either WITH or WITHOUT the 'www.' prefix. Choose ONLY one option:

# To redirect all users to access the site WITH the 'www.' prefix,
# (http://example.com/foo will be redirected to http://www.example.com/foo)
# uncomment the following:
# RewriteCond %{HTTP_HOST} .
# RewriteCond %{HTTP_HOST} !^www. [NC]
# RewriteRule ^ http%{ENV:protossl}://www.%{HTTP_HOST}%{REQUEST_URI} [L,R=301]

# To redirect all users to access the site WITHOUT the 'www.' prefix,
# (http://www.example.com/foo will be redirected to http://example.com/foo)
# uncomment the following:
# RewriteCond %{HTTP_HOST} ^www\.(.+)$ [NC]
# RewriteRule ^ http%{ENV:protossl}://%1%{REQUEST_URI} [L,R=301]

# Modify the RewriteBase if you are using Drupal in a subdirectory or in a
# VirtualDocumentRoot and the rewrite rules are not working properly.
# For example if your site is at http://example.com/drupal uncomment and
# modify the following line:
# RewriteBase /drupal
#
# If your site is running in a VirtualDocumentRoot at http://example.com/,
# uncomment the following line:
# RewriteBase /

# Redirect common PHP files to their new locations.
RewriteCond %{REQUEST_URI} ^(.*)?/\(install.php\) [OR]
RewriteCond %{REQUEST_URI} ^(.*)?/\(rebuild.php\)
RewriteCond %{REQUEST_URI} !core
RewriteRule ^ %1/core/%2 [L,QSA,R=301]

# Rewrite install.php during installation to see if mod_rewrite is working
RewriteRule ^core/install.php core/install.php?rewrite=ok [QSA,L]

# Pass all requests not referring directly to files in the filesystem to
# index.php.
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_URI} !=/favicon.ico
RewriteRule ^ index.php [L]

# For security reasons, deny access to other PHP files on public sites.
# Note: The following URI conditions are not anchored at the start (^),
# because Drupal may be located in a subdirectory. To further improve
# security, you can replace '!/' with '!^/'.
# Allow access to PHP files in /core (like authorize.php or install.php):
RewriteCond %{REQUEST_URI} !/core/[^/]*\.php$
# Allow access to test-specific PHP files:
RewriteCond %{REQUEST_URI} !/core/[^/]*\.php$
# Allow access to test-specific PHP files:
RewriteCond %{REQUEST_URI} !/core/modules/system/tests/https\.php
# Allow access to Statistics module's custom front controller.
# Copy and adapt this rule to directly execute PHP files in contributed or
# custom modules or to run another PHP application in the same directory.
RewriteCond %{REQUEST_URI} !/core/modules/statistics/statistics\.php$
# Deny access to any other PHP files that do not match the rules above.
# Specifically, disallow autoload.php from being served directly.
RewriteRule "^\(.*\!/\*autoload\)/\.(php|$)" - [F]

# Rules to correctly serve gzip compressed CSS and JS files.
# Requires both mod_rewrite and mod_headers to be enabled.
<IfModule mod_headers.c>
  # Serve gzip compressed CSS files if they exist and the client accepts gzip.
  RewriteCond %{HTTP:Accept-encoding} gzip
  RewriteCond %{REQUEST_FILENAME}\.gz -s
  RewriteRule ^(.*)\.css $1\.css\.gz [QSA]

  # Serve gzip compressed JS files if they exist and the client accepts gzip.
  RewriteCond %{HTTP:Accept-encoding} gzip
  RewriteCond %{REQUEST_FILENAME}\.gz -s
  RewriteRule ^(.*)\.js $1\.js\.gz [QSA]

  # Serve correct content types, and prevent mod_deflate double gzip.
  RewriteRule \.css\.gz$ - [T=text/css,E=no-gzip:1]
  RewriteRule \.js\.gz$ - [T=text/javascript,E=no-gzip:1]
</IfModule>

.FilesMatch "(\.js\.gz|\.css\.gz)$"
  # Serve correct encoding type.
  Header set Content-Encoding gzip
  # Force proxies to cache gzipped & non-gzipped css/js files separately.
  Header append Vary Accept-Encoding
</FilesMatch>
</IfModule>
</IfModule>

# Add headers to all responses.
<IfModule mod_headers.c>
  # Disable content sniffing, since it’s an attack vector.
  Header always set X-Content-Type-Options nosniff
</IfModule>