```
<VirtualHost 192.168.167.241:80>
 ServerName cloud.pronto.de
 ServerAlias
 DocumentRoot /home/cloud.pronto.de/owncloud/
 ServerAdmin prontos@email.de
 CustomLog /var/log/apache2/cloud.pronto.de-access.log combined
 ErrorLog /var/log/apache2/cloud.pronto.de-error.log
 LogLevel warn
</VirtualHost>


<VirtualHost 192.168.167.241:443>
        ServerAdmin webmaster@localhost
        ServerName cloud.kastner.de

       DocumentRoot /home/cloud.pronto.de/owncloud/
       <Directory />
               Options FollowSymLinks
               AllowOverride None
       </Directory>
       <Directory /var/www/>
               Options Indexes FollowSymLinks MultiViews
               AllowOverride None
               Order allow,deny
               allow from all
       </Directory>

       ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
       <Directory "/usr/lib/cgi-bin">
               AllowOverride None
               Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
               Order allow,deny
               Allow from all
       </Directory>

       ErrorLog ${APACHE_LOG_DIR}/ssl_cloud.pronto.de-error.log

       # Possible values include: debug, info, notice, warn, error, crit,
       # alert, emerg.
       LogLevel warn

       CustomLog ${APACHE_LOG_DIR}/ssl_cloud.pronto.de-error.log combined

       #   SSL Engine Switch:
       #   Enable/Disable SSL for this virtual host.
       SSLEngine on

       #   A self-signed (snakeoil) certificate can be created by installing
       #   the ssl-cert package. See
       #   /usr/share/doc/apache2.2-common/README.Debian.gz for more info.
       #   If both key and certificate are stored in the same file, only the
       #   SSLCertificateFile directive is needed.
       SSLCertificateFile    /etc/ssl/cloud.cer
       SSLCertificateKeyFile /etc/ssl/cloud.key

       #   Server Certificate Chain:
       #   Point SSLCertificateChainFile at a file containing the
       #   concatenation of PEM encoded CA certificates which form the
       #   certificate chain for the server certificate. Alternatively
       #   the referenced file can be the same as SSLCertificateFile
       #   when the CA certificates are directly appended to the server
       #   certificate for convinience.
       #SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt

       #   Certificate Authority (CA):
       #   Set the CA certificate verification path where to find CA
       #   certificates for client authentication or alternatively one
       #   huge file containing all of them (file must be PEM encoded)
       #   Note: Inside SSLCACertificatePath you need hash symlinks
       #         to point to the certificate files. Use the provided
       #         Makefile to update the hash symlinks after changes.
       #SSLCACertificatePath /etc/ssl/certs/
       #SSLCACertificateFile /etc/apache2/ssl.crt/ca-bundle.crt

       #   Certificate Revocation Lists (CRL):
       #   Set the CA revocation path where to find CA CRLs for client
       #   authentication or alternatively one huge file containing all
       #   of them (file must be PEM encoded)
       #   Note: Inside SSLCARevocationPath you need hash symlinks
       #         to point to the certificate files. Use the provided
#         Makefile to update the hash symlinks after changes.
#SSLCARevocationPath /etc/apache2/ssl.crl/
#SSLCARevocationFile /etc/apache2/ssl.crl/ca-bundle.crl

#   Client Authentication (Type):
#   Client certificate verification type and depth.  Types are
#   none, optional, require and optional_no_ca.  Depth is a
#   number which specifies how deeply to verify the certificate
#   issuer chain before deciding the certificate is not valid.
#SSLVerifyClient require
#SSLVerifyDepth  10

#   Access Control:
#   With SSLRequire you can do per-directory access control based
#   on arbitrary complex boolean expressions containing server
#   variable checks and other lookup directives.  The syntax is a
#   mixture between C and Perl.  See the mod_ssl documentation
#   for more details.
#<Location />
#SSLRequire (    %{SSL_CIPHER} !~ m/^(EXP|NULL)/ \
#            and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
#            and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
#            and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
#            and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20       ) \
#           or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
#</Location>

#   SSL Engine Options:
#   Set various options for the SSL engine.
#   o FakeBasicAuth:
#     Translate the client X.509 into a Basic Authorisation.  This means that
#     the standard Auth/DBMAuth methods can be used for access control.  The
#     user name is the `one line' version of the client's X.509 certificate.
#     Note that no password is obtained from the user. Every entry in the user
#     file needs this password: `xxj31ZMTZzkVA'.
#   o ExportCertData:
#     This exports two additional environment variables: SSL_CLIENT_CERT and
#     SSL_SERVER_CERT. These contain the PEM-encoded certificates of the
#     server (always existing) and the client (only existing when client
#     authentication is used). This can be used to import the certificates
#     into CGI scripts.
#   o StdEnvVars:
#     This exports the standard SSL/TLS related `SSL_*' environment variables.
#     Per default this exportation is switched off for performance reasons,
#     because the extraction step is an expensive operation and is usually
#     useless for serving static content. So one usually enables the
#     exportation for CGI and SSI requests only.
#   o StrictRequire:
#     This denies access when "SSLRequireSSL" or "SSLRequire" applied even
#     under a "Satisfy any" situation, i.e. when it applies access is denied
#     and no other module can change it.
#   o OptRenegotiate:
#     This enables optimized SSL connection renegotiation handling when SSL
#     directives are used in per-directory context.
#SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
<FilesMatch "\.(cgi|shtml|phtml|php)$">
       SSLOptions +StdEnvVars
</FilesMatch>
<Directory /usr/lib/cgi-bin>
       SSLOptions +StdEnvVars
</Directory>

#   SSL Protocol Adjustments:
#   The safe and default but still SSL/TLS standard compliant shutdown
#   approach is that mod_ssl sends the close notify alert but doesn't wait for
#   the close notify alert from client. When you need a different shutdown
#   approach you can use one of the following variables:
#   o ssl-unclean-shutdown:
#     This forces an unclean shutdown when the connection is closed, i.e. no
#     SSL close notify alert is send or allowed to received.  This violates
#     the SSL/TLS standard but is needed for some brain-dead browsers. Use
#     this when you receive I/O errors because of the standard approach where
#     mod_ssl sends the close notify alert.
#   o ssl-accurate-shutdown:
#     This forces an accurate shutdown when the connection is closed, i.e. a
#     SSL close notify alert is send and mod_ssl waits for the close notify
#     alert of the client. This is 100% SSL/TLS standard compliant, but in
#     practice often causes hanging connections with brain-dead browsers. Use
#     this only for browsers where you know that their SSL implementation
#     works correctly.
```

```
	#   Notice: Most problems of broken clients are also related to the HTTP
	#   keep-alive facility, so you usually additionally want to disable
	#   keep-alive for those clients, too. Use variable "nokeepalive" for this.
	#   Similarly, one has to force some clients to use HTTP/1.0 to workaround
	#   their broken HTTP/1.1 implementation. Use variables "downgrade-1.0" and
	#   "force-response-1.0" for this.
	BrowserMatch "MSIE [2-6]" \
		nokeepalive ssl-unclean-shutdown \
		downgrade-1.0 force-response-1.0
	# MSIE 7 and newer should be able to use keepalive
	BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown

</VirtualHost>
```