Ubuntu Install

Here we have some step-by-step instructions for installing Tiki on Ubuntu Server with Apache2, MySQL/MARIADB, and PHP5/PHP7/PHP8.

Install Ubuntu Server

I would recommend installing only SSH Server as part of the initial install.

Patch it up

```
sudo apt-get update sudo apt-get upgrade
```

Install Lamp

You can install a LAMP (Linux, Apache, Mariadb, PHP) server with this command under Ubuntu:

```
sudo apt install lamp-server^ # this comes with Mysql 8.0 however we want to use Mariadb in order to keep usual way to make db as root in the first instance safely through a terminal window sudo apt install -y mariadb-server mariadb-client # this will replace mysql with mariadb
```

However, Ubuntu 22.04 **doesn't contain PHP 7.4 any longer**. Only PHP 8.1 is supported. Tiki versions before 26 need PHP 7.4. But there's a good PPA for PHP 7.4 available. So you install the LAMP server and then the PHP 7.4 packages from this PPA.

Use these commands to integrate the PPA in the system:

```
sudo apt-get update sudo apt -y install software-properties-common sudo add-apt-repository ppa:ondrej/php sudo apt-get update
```

You should then be able to install all the PHP 7.4 packages with this command:

```
sudo apt install libapache2-mod-php7.4 memcached php7.4 php7.4-apcu php7.4-bcmath php7.4-bz2 php7.4-common php7.4-curl php7.4-gd php7.4-intl php7.4-ldap php7.4-mbstring php7.4-memcache php7.4-memcached php7.4-mysql php7.4-opcache php7.4-pspell php7.4-soap php7.4-sqlite3 php7.4-tidy php7.4-xml php7.4-xmlrpc php7.4-zip php-apcu php-bcmath php-common php-curl php-gd php-intl php-mbstring php-memcache php-memcached php-mysql php-pear php-pspell php-sqlite3 php-tidy php-xmlrpc php-zip
```

You might also want to install some extra packages:

```
sudo apt install bsdmainutils catdoc composer curl elinks git man-db memcached odt2txt phpmyadmin poppler-utils postfix pstotext
```

Set php 7.4 as default

In case you have other php versions newer than 7.4 (such as php 8.0.x), you may set the default php version at 7.4 with these commands:

```
sudo a2dismod php8.1 sudo a2enmod php7.4 sudo service apache2 restart sudo update-alternatives --set php /usr/bin/php7.4
```

Extra steps needed

If it's a brand new server, you might be asked a few questions to configure some server programs.

To configure **postfix** (program to mange email sending and receiving from the server), you can answer, for instance (unless you know what you are doing):

- General type of mail configuration: **Internet site**
- System mail name: mail.yourdomain.org

To configure **phpmyadmin** (program to manage mysql databases through a web based GUI), you can answer, for instance (unless you know what you are doing):

- Webserver: **apache2** (you can select using the space bar and arrow keys if needed to move between the options)
- Configure database with dbconfig-common: yes
- MySQL application password for phpmyadmin: [leave empty]

Then you can enable the modules

```
#sudo phpenmod mcrypt # mcrypt is not found in php 7.2 sudo phpenmod mbstring
```

Extra packages for Media Alchemyst:

```
#sudo apt install software-properties-common #sudo add-apt-repository ppa:libreoffice/libreoffice-6-0 # not available for ubuntu 20.04 at the time of this writing sudo apt install -y libreoffice ffmpeg unoconv ghostscript php-imagick imagemagick
```

Enable the Apache rewrite rules:

Command on a console

```
sudo a2enmod rewrite sudo service apache2 restart
```

And you can enable the self-signed ssl certificates to allow connections with https:

Command on a console

```
sudo a2enmod ssl sudo a2ensite default-ssl sudo service apache2 restart
```

And add this section between the VirtualHost tags the to this file /etc/apache2/sites-enabled/000-default.conf (or equivalent for your configuration; in this case, the doc. root where tiki is installed is **/var/www/html/**, which is where ubuntu 20.04 comes pre-configured for the base doc root):

```
sudo nano /etc/apache2/sites-enabled/000-default.conf
```

Section to add (just after the Document root line):

```
<Directory /var/www/html/> Options Indexes FollowSymLinks MultiViews AllowOverride All ## The following lines to allow connections have changed syntax ## between apache 2.2 and apache 2.4 ## See: http://httpd.apache.org/docs/2.4/upgrading.html > Run-Time Configuration Changes > Access control #Order allow,deny #Allow from all Require all granted </Directory>
```

Do the same type of edit into the equivalent config file for https:

```
sudo nano /etc/apache2/sites-enabled/default-ssl.conf
```

After the change, they should look like:

Contents of /etc/apache2/sites-enabled/000-default.conf

```
<VirtualHost *:80> # The ServerName directive sets the request scheme, hostname and port that # the server uses to identify itself. This is used when creating # redirection URLs. In the context of virtual hosts, the ServerName # specifies what hostname must appear in the request's Host: header to # match this virtual host. For the default virtual host (this file) this # value is not decisive as it is used as a last resort host regardless. # However, you must set it for any further virtual host explicitly. #ServerName www.example.com ServerAdmin webmaster@localhost DocumentRoot /var/www/html <Directory /var/www/html/> Options Indexes FollowSymLinks MultiViews AllowOverride All #Order allow,deny #Allow from all Require all granted </Directory> # Available loglevels: trace8, ..., trace1, debug, info, notice, warn, # error, crit, alert, emerg. # It is also possible to configure the loglevel for particular # modules, e.g. #LogLevel info ssl:warn ErrorLog ${APACHE_LOG_DIR}/error.log CustomLog ${APACHE_LOG_DIR}/access.log combined # For most configuration files from conf-available/, which are # enabled or disabled at a global level, it is possible to # include a line for only one particular virtual host. For example the # following line enables the CGI configuration for this host only # after it has been globally disabled with "a2disconf". #Include conf-available/serve-cgi-bin.conf </VirtualHost> # vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Contents of /etc/apache2/sites-enabled/default-ssl.conf

```
<IfModule mod_ssl.c> <VirtualHost _default_:443> ServerAdmin webmaster@localhost DocumentRoot /var/www/html <Directory /var/www/html/> Options Indexes FollowSymLinks MultiViews
```

```
AllowOverride All ## The following lines to allow connections have changed syntax ## between
apache 2.2 and apache 2.4 ## See: http://httpd.apache.org/docs/2.4/upgrading.html > Run-Time
Configuration Changes > Access control #Order allow,deny #Allow from all Require all granted
</Directory> # Available loglevels: trace8, ..., trace1, debug, info, notice, warn, # error, crit, alert,
emerg. # It is also possible to configure the loglevel for particular # modules, e.g. #LogLevel info
ssl:warn ErrorLog ${APACHE_LOG_DIR}/error.log CustomLog ${APACHE_LOG_DIR}/access.log
combined # For most configuration files from conf-available/, which are # enabled or disabled at a
global level, it is possible to # include a line for only one particular virtual host. For example the #
following line enables the CGI configuration for this host only # after it has been globally disabled with
"a2disconf". #Include conf-available/serve-cgi-bin.conf # 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/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/certs/ssl-cert-snakeoil.pem SSLCertificateKeyFile /etc/ssl/private/ssl-cert-
snakeoil.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 # 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
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 </VirtualHost> </IfModule> # vim: syntax=apache ts=4 sw=4 sts=4 sr noet

1.2. Create for tiki a mysql db and a mysql user with local perms for that db

Option 1 - using console

We can create a database and user for tiki. We can do so through a terminal windows converting ourselves in root in a first step with `sudo su`, and then, we can continue with the mysql commands. Please note that when executing `mysql -+` (as root) you need to provide no password, since mysql authenticaation scheme in Ubuntu 18.04 only requires that you run this command as user root in the shell.

commands in a terminal window

```
# sudo su # mysql -p # mysql> CREATE DATABASE mytikidb CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci; # mysql> CREATE USER 'mytikiuser'@'localhost' IDENTIFIED BY 'mypassword'; # mysql> GRANT ALL ON mytikidb.* TO 'mytikiuser'@'localhost'; # mysql> FLUSH PRIVILEGES; # mysql> \q
```

If it's a brand new server, you may wish (at your own risk) to secure your mysql installation with the provided script (`sudo mysql_secure_installation`), and do the recommended setup steps indicated here:
https://vitux.com/how-to-install-and-configure-mysql-in-ubuntu-18-04-lts/

Option 2 - use phpmyadmin or adminer

Alternatively, you can use your PhpMyAdmin install:
http://example.com/phpmyadmin/

or a small php script to manage your mysql server, called Adminer, that you can download to your server with an instruction like this one in a terminal:

```
sudo wget https://www.adminer.org/latest-mysql-en.php -O "/var/www/html/tiki/adminer.php" # once you are done with managing your database and users, you are encouraged to remove adminer.php from the web access to somewhere else unaccessible from your web root for extra precaution sudo mv /var/www/html/tiki/adminer.php /var/www/
```

Then you can login to your GUI web access (either phpmyadmin or adminer). You will probably not be able to login with your mysql root password, since the msyql auth scheme in ubuntu 18.04 has changed compared to previous versions (it currently uses auth_socket instead of native_mysql_password). Therefore, you can use the `debian-sys-maint` credentials that you will find here: `/etc/mysql/debian.cnf`

- Create a new user for mysql that does NOT have all perms
  - New user is **tikiuser**, and we do not select neither database nor general perms for this user.
- Go to the start page for PhpMyAdmin/Adminer again

- Create a bbdd named **tiki24git** and as collation select utf8mb4_unicode_ci
- Go to the users tab, and go to Databases
    - Select the db tiki24git and check all perms except "grant" to that user for that db

And that's all. Then you will need to tell tiki that we have created:

- mysql db: tiki24git
- mysql user: tikiuser
- mysql password: xxxxxxx (whatever you provided as password)

It is recommended to secure your phpMyAdmin by following the steps:
https://www.atlantic.net/vps-hosting/how-to-install-and-secure-phpmyadmin-on-centos-8/

1.3. Install Tiki 24 through git

(url's taken from http://dev.tiki.org/Get+code)

We will fetch tiki24 through git shallow clone. This option is best to checkout on production servers. It uses about 214M of disk space.

Change directory to /var/www and get that tiki24 clone:

```
cd /var/www git clone --depth=1 --branch=24.x https://gitlab.com/tikiwiki/tiki.git tiki24
```

Then we run the Tiki setup script through console:

```
cd tiki24 sh setup.sh
```

The first question you will asked is about running composer to fetch the dependencies (libraries) which need to be downloaded for Tiki to run:

```
# First choose: Your choice [c]?: c # Then choose to fix file and directory permissions (classic default): Your choice [f]?: f User [www-data]: www-data Group [www-data]: www-data Multi []: # Then you can quit from the setup.sh script: Your choice [x]?: x
```

You can have a look at what the script does here:
https://gitlab.com/tikiwiki/tiki/-/blob/24.x/setup.sh

This script manages all Tiki dependencies through a program called "Composer" (it downloads and updated them when needed)

You need to make this tiki24 folder available through the webserver.
You can easily do so by means of a symbolic link

```
ln -s /var/www/tiki24 /var/www/html/tiki
```

Then you can proceed with the standard Tiki installation through the web browser (replace example.com with your domain name):
http://example.com/tiki/

and this will take us to **tiki-install.php**:
http://example.com/tiki/tiki-install.php

Once installed, you may need to manually create a synbolic link to .htaccess file in the tiki root folder, since it's needed for tiki to work properly and it might not get automatically created under some circumstances/setups, apparently. You can do so with the command:

```
ln -s /var/www/html/tiki/_htaccess /var/www/html/tiki/.htaccess
```

Tip for Spanish speakers: see the video from a course on Tiki:

- http://seeds4c.org/CT14
- http://seeds4c.org/CT14%20S1.1#Instalaci_n_del_software
  (from miunte 9' 30'' onwards)

At the screen to Setup the database connection, we provide the database name, db user and password that we previosuly created and we follow instructions on screen.

At the new installation, the first user is admin, with password admin. We replace the password and continue to log in for the first time.
You have your tiki installed, ready for you to continue using the Wizards

1.4. Get a free SSL Certificate from Let's Encrypt

Install a repository to get updated version of packages that will just work:

```
#sudo add-apt-repository ppa:certbot/certbot #sudo apt update sudo apt install python3-certbot-apache
```

Before we can start to create the SSL cert, set the domain name in the vhost configuration file. Open the default vhost file with an editor:

```
nano /etc/apache2/sites-available/000-default.conf
```

and add the line:

```
ServerName example.com
```

Replace example.com with your fully qualified domain name.

Then create the SSL Certificate with this command:

```
sudo certbot --apache
```

The command will start a wizard that asks you several questions.
If you want further customization, use other options as described or linked from here:
https://certbot.eff.org/lets-encrypt/ubuntubionic-apache

1.4.1. Add some composer packages through console

If you want, you can add some packages through console

```
sudo su cd /var/www/html/tiki/ composer require --update-no-dev --prefer-dist mpdf/mpdf composer
require --update-no-dev --prefer-dist fullcalendar/fullcalendar-scheduler composer require --update-no-
dev --prefer-dist tikiwiki/diagram # updated for Tiki 21.x instead of mxgraph-editor composer require --
update-no-dev --prefer-dist onelogin/php-saml composer require --update-no-dev --prefer-dist
thiagoalessio/tesseract_ocr # composer require --update-no-dev --prefer-dist npm-asset/pdfjs-dist-
viewer-min #composer require --update-no-dev --prefer-dist media-alchemyst/media-alchemyst
#composer require --update-no-dev --prefer-dist php-ffmpeg/php-ffmpeg #composer require --update-
no-dev --prefer-dist npm-asset/dexie #composer require --update-no-dev --prefer-dist npm-asset/lozad
composer require --update-no-dev --prefer-dist google/apiclient service apache2 restart chown www-
data:www-data composer.* exit
```

Tiki 26+

For Tiki26 or higher, you need php8.1. If you want to install this tiki, you will need to tweak the previous git commands to clone branch 26.x.

You should then be able to install the required PHP 8.1 packages with this command:

```
sudo apt install libapache2-mod-php8.1 memcached php8.1 php8.1-apcu php8.1-bcmath php8.1-bz2
php8.1-common php8.1-curl php8.1-gd php8.1-intl php8.1-ldap php8.1-mbstring php8.1-memcache
php8.1-memcached php8.1-mysql php8.1-opcache php8.1-pspell php8.1-soap php8.1-sqlite3 php8.1-tidy
php8.1-xml php8.1-xmlrpc php8.1-zip php-apcu php-bcmath php-common php-curl php-gd php-intl php-
mbstring php-memcache php-memcached php-mysql php-pear php-pspell php-sqlite3 php-tidy php-
xmlrpc php-zip
```

In this case, you may set the default php version at 8.1 with these commands:

```
sudo a2dismod php7.4 sudo a2enmod php8.1 sudo service apache2 restart sudo update-alternatives --
set php /usr/bin/php8.1
```

Install Manticore search

Since Tiki26, you can start using Manticore search for production as your unified search index. You will need to install some extra packages for manticore search:

```
wget https://repo.manticoresearch.com/manticore-repo.noarch.deb sudo dpkg -i manticore-repo.noarch.deb sudo apt update sudo apt install -y manticore manticore-extra
```

Once installed, you can start it with:

```
sudo systemctl start manticore
```

More information:
https://manticoresearch.com/install/

Related links

- https://www.howtoforge.com/tutorial/tiki-wiki-ubuntu/ - Tutorial showing installation of Tiki 21 from sf.net zip file into an Ubuntu 20.04 installation with ufw enabled (untested but looks very nice)

1.1. Install Lamp

You can install Linux, Apache, Mysql and PHP (plus a few other helper utils such as git or subversion, etc) with:

```
sudo apt install mariadb-server mariadb-client apache2 curl composer php php-tidy php-pear memcached php-gd php-xmlrpc phpmyadmin php-mbstring libapache2-mod-php php-mysql php-apcu php-curl php-intl php-sqlite3 php-zip postfix git subversion php-memcache php7.4-gettext php-pspell php-zip poppler-utils php-memcached bsdmainutils pstotext catdoc elinks man-db odt2txt php-pear pstotext php-common php-intl php-bcmath php7.4-opcache php7.4-xml php7.4-zip php7.4-ldap php7.4-bz2
```

Note
If you need to install Tiki22 or newer, you don't need to add any extra repository, since you already have php 7.4 using the default ubuntu 20.04 repositories.

Need extra repos for php 7.4+ ?

[+]
Add php 7.4 packages

```
sudo apt -y install php7.4 php7.4-tidy php7.4-gd php7.4-xmlrpc php7.4-mbstring php7.4-mysql php7.4-apcu php7.4-curl php7.4-intl php7.4-sqlite3 php7.4-zip php7.4-memcache php7.4-pspell php7.4-memcached php7.4-common php7.4-opcache php7.4-xml php7.4-ldap php7.4-bcmath php7.4-soap libapache2-mod-php7.4 php7.4-bz2
```

Set php 7.4 as default

In case you have other php versions newer than 7.4 (such as php 8.0.x), you may set the default php version at 7.4 with these commands:

```
sudo a2dismod php8.0 sudo a2enmod php7.4 sudo service apache2 restart sudo update-alternatives --set php /usr/bin/php7.4
```

Extra steps needed

If it's a brand new server, you might be asked a few questions to configure some server programs.

To configure **postfix** (program to mange email sending and receiving from the server), you can answer, for instance (unless you know what you are doing):

- General type of mail configuration: **Internet site**
- System mail name: mail.yourdomain.org

To configure **phpmyadmin** (program to manage mysql databases through a web based GUI), you can answer, for instance (unless you know what you are doing):

- Webserver: **apache2** (you can select using the space bar and arrow keys if needed to move between the options)
- Configure database with dbconfig-common: yes
- MySQL application password for phpmyadmin: [leave empty]

Then you can enable the modules

```
#sudo phpenmod mcrypt # mcrypt is not found in php 7.2 sudo phpenmod mbstring
```

Extra packages for Media Alchemyst:

```
#sudo apt install software-properties-common #sudo add-apt-repository ppa:libreoffice/libreoffice-6-0 # not available for ubuntu 20.04 at the time of this writing sudo apt install libreoffice ffmpeg unoconv ghostscript php-imagick imagemagick
```

Enable the Apache rewrite rules:

Command on a console

```
sudo a2enmod rewrite sudo service apache2 restart
```

And you can enable the self-signed ssl certificates to allow connections with https:

Command on a console

```
sudo a2enmod ssl sudo a2ensite default-ssl sudo service apache2 restart
```

And add this section between the VirtualHost tags the to this file /etc/apache2/sites-enabled/000-

default.conf (or equivalent for your configuration; in this case, the doc. root where tiki is installed is **/var/www/html/**, which is where ubuntu 20.04 comes pre-configured for the base doc root):

```
sudo nano /etc/apache2/sites-enabled/000-default.conf
```

Section to add (just after the Document root line):

```
<Directory /var/www/html/> Options Indexes FollowSymLinks MultiViews AllowOverride All ## The following lines to allow connections have changed syntax ## between apache 2.2 and apache 2.4 ## See: http://httpd.apache.org/docs/2.4/upgrading.html > Run-Time Configuration Changes > Access control #Order allow,deny #Allow from all Require all granted </Directory>
```

Do the same type of edit into the equivalent config file for https:

```
sudo nano /etc/apache2/sites-enabled/default-ssl.conf
```

After the change, they should look like:

Contents of /etc/apache2/sites-enabled/000-default.conf

```
<VirtualHost *:80> # The ServerName directive sets the request scheme, hostname and port that # the server uses to identify itself. This is used when creating # redirection URLs. In the context of virtual hosts, the ServerName # specifies what hostname must appear in the request's Host: header to # match this virtual host. For the default virtual host (this file) this # value is not decisive as it is used as a last resort host regardless. # However, you must set it for any further virtual host explicitly. #ServerName www.example.com ServerAdmin webmaster@localhost DocumentRoot /var/www/html <Directory /var/www/html/> Options Indexes FollowSymLinks MultiViews AllowOverride All #Order allow,deny #Allow from all Require all granted </Directory> # Available loglevels: trace8, ..., trace1, debug, info, notice, warn, # error, crit, alert, emerg. # It is also possible to configure the loglevel for particular # modules, e.g. #LogLevel info ssl:warn ErrorLog ${APACHE_LOG_DIR}/error.log CustomLog ${APACHE_LOG_DIR}/access.log combined # For most configuration files from conf-available/, which are # enabled or disabled at a global level, it is possible to # include a line for only one particular virtual host. For example the # following line enables the CGI configuration for this host only # after it has been globally disabled with "a2disconf". #Include conf-available/serve-cgi-bin.conf </VirtualHost> # vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Contents of /etc/apache2/sites-enabled/default-ssl.conf

```
<IfModule mod_ssl.c> <VirtualHost _default_:443> ServerAdmin webmaster@localhost DocumentRoot /var/www/html <Directory /var/www/html/> Options Indexes FollowSymLinks MultiViews AllowOverride All ## The following lines to allow connections have changed syntax ## between apache 2.2 and apache 2.4 ## See: http://httpd.apache.org/docs/2.4/upgrading.html > Run-Time Configuration Changes > Access control #Order allow,deny #Allow from all Require all granted </Directory> # Available loglevels: trace8, ..., trace1, debug, info, notice, warn, # error, crit, alert, emerg. # It is also possible to configure the loglevel for particular # modules, e.g. #LogLevel info
```

```
ssl:warn ErrorLog ${APACHE_LOG_DIR}/error.log CustomLog ${APACHE_LOG_DIR}/access.log
combined # For most configuration files from conf-available/, which are # enabled or disabled at a
global level, it is possible to # include a line for only one particular virtual host. For example the #
following line enables the CGI configuration for this host only # after it has been globally disabled with
"a2disconf". #Include conf-available/serve-cgi-bin.conf # 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/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/certs/ssl-cert-snakeoil.pem SSLCertificateKeyFile /etc/ssl/private/ssl-cert-
snakeoil.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 # 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
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 </VirtualHost> </IfModule> # vim: syntax=apache ts=4 sw=4 sts=4 sr noet

1.2. Create for tiki a mysql db and a mysql user with local perms for that db

Option 1 - using console

We can create a database and user for tiki. We can do so through a terminal windows converting ourselves in root in a first step with `sudo su`, and then, we can continue with the mysql commands. Please note that when executing `mysql -+` (as root) you need to provide no password, since mysql authenticaation scheme in Ubuntu 18.04 only requires that you run this command as user root in the shell.

commands in a terminal window

```
# sudo su # mysql -p # mysql> CREATE DATABASE mytikidb CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci; # mysql> CREATE USER 'mytikiuser'@'localhost' IDENTIFIED BY 'mypassword'; # mysql> GRANT ALL ON mytikidb.* TO 'mytikiuser'@'localhost'; # mysql> FLUSH PRIVILEGES; # mysql> \q
```

If it's a brand new server, you may wish (at your own risk) to secure your mysql installation with the provided script (`sudo mysql_secure_installation`), and do the recommended setup steps indicated here:
https://vitux.com/how-to-install-and-configure-mysql-in-ubuntu-18-04-lts/

Option 2 - use phpmyadmin or adminer

Alternatively, you can use your PhpMyAdmin install:
http://example.com/phpmyadmin/

or a small php script to manage your mysql server, called Adminer, that you can download to your server with an instruction like this one in a terminal:

```
sudo wget https://www.adminer.org/latest-mysql-en.php -O "/var/www/html/tiki/adminer.php" # once you are done with managing your database and users, you are encouraged to remove adminer.php from the web access to somewhere else unaccessible from your web root for extra precaution sudo mv /var/www/html/tiki/adminer.php /var/www/
```

Then you can login to your GUI web access (either phpmyadmin or adminer). You will probably not be able to login with your mysql root password, since the msyql auth scheme in ubuntu 18.04 has changed compared to previous versions (it currently uses auth_socket instead of native_mysql_password). Therefore, you can use the `debian-sys-maint` credentials that you will find here: `/etc/mysql/debian.cnf`

- Create a new user for mysql that does NOT have all perms
  - New user is **tikiuser**, and we do not select neither database nor general perms for this user.
- Go to the start page for PhpMyAdmin/Adminer again
- Create a bbdd named **tiki18svn** and as collation select utf8mb4_unicode_ci
- Go to the users tab, and go to Databases
  - Select the db tiki18svn, and check all perms except "grant" to that user for that db

And that's all. Then you will need to tell tiki that we have created:

- mysql db: tiki18svn
- mysql user: tikiuser
- mysql password: xxxxxxx (whatever you provided as password)

It is recommended to secure your phpMyAdmin by following the steps: https://www.atlantic.net/vps-hosting/how-to-install-and-secure-phpmyadmin-on-centos-8/

1.3. Use bash instead of dash

Ubuntu 20.04 comes with dash as the default shell interpreter. And Tiki shell scripts expect bash instead of dash. Therefore, one easy solution is to setup bash for your user with sudo perms in the server to use bash by default. You can do so with the command (for user *username*):

```
usermod --shell /bin/bash yourusername
```

More information:

- https://wiki.ubuntu.com/ChangingShells
- https://www.scivision.dev/ubuntu-default-shell/
- https://superuser.com/questions/46748/how-do-i-make-bash-my-default-shell-on-ubuntu

1.4. Install Tiki 21 through git

(url's taken from http://dev.tiki.org/Get+code)

We will fetch tiki21 through git shallow clone. This option is best to checkout on production servers. It uses about 187M of disk space.

Change directory to /var/www and get that tiki21 clone:

```
cd /var/www git clone --depth=1 --branch=21.x https://gitlab.com/tikiwiki/tiki.git tiki21
```

Then we run the Tiki setup script through console:

```
cd tiki21 bash setup.sh
```

The first question you will asked is about running composer to fetch the dependencies (libraries) which need to be downloaded for Tiki to run:

```
# First choose: Your choice [c]?: c # Then choose to fix file and directory permissions (classic default): Your choice [f]?: f User [www-data]: www-data Group [www-data]: www-data Multi []: # Then you can quit from the setup.sh script: Your choice [x]?: x
```

You can have a look at what the script does here: https://gitlab.com/tikiwiki/tiki/-/blob/21.x/setup.sh

This script manages all Tiki dependencies through a program called "Composer" (it downloads and updated them when needed)

You need to make this tiki21 folder available through the webserver.
You can easily do so by means of a symbolic link

```
ln -s /var/www/tiki21 /var/www/html/tiki
```

Then you can proceed with the standard Tiki installation through the web browser (replace example.com with your domain name):
http://example.com/tiki/

and this will take us to **tiki-install.php**:
http://example.com/tiki/tiki-install.php

Tip for Spanish speakers: see the video from a course on Tiki:

- http://seeds4c.org/CT14
- http://seeds4c.org/CT14%20S1.1#Instalaci_n_del_software
  (from miunte 9' 30'' onwards)

At the screen to Setup the database connection, we provide the database name, db user and password that we previosuly created and we follow instructions on screen.

At the new installation, the first user is admin, with password admin. We replace the password and continue to log in for the first time.
You have your tiki installed, ready for you to continue using the Wizards

1.5. Get a free SSL Certificate from Let's Encrypt

Install a repository to get updated version of packages that will just work:

```
#sudo add-apt-repository ppa:certbot/certbot #sudo apt update sudo apt install python3-certbot-apache
```

Before we can start to create the SSL cert, set the domain name in the vhost configuration file. Open the default vhost file with an editor:

```
nano /etc/apache2/sites-available/000-default.conf
```

and add the line:

```
ServerName example.com
```

Replace example.com with your fully qualified domain name.

Then create the SSL Certificate with this command:

```
sudo certbot --apache
```

The command will start a wizard that asks you several questions.
If you want further customization, use other options as described or linked from here:
https://certbot.eff.org/lets-encrypt/ubuntubionic-apache

1.5.1. Add some composer packages through console

If you want, you can add some packages through console

```
sudo su cd /var/www/html/ composer require --update-no-dev --prefer-dist mpdf/mpdf composer require --update-no-dev --prefer-dist fullcalendar/fullcalendar-scheduler composer require --update-no-dev --prefer-dist tikiwiki/diagram # updated for Tiki 21.x instead of mxgraph-editor composer require --update-no-dev --prefer-dist onelogin/php-saml composer require --update-no-dev --prefer-dist thiagoalessio/tesseract_ocr # composer require --update-no-dev --prefer-dist npm-asset/pdfjs-dist-viewer-min #composer require --update-no-dev --prefer-dist media-alchemyst/media-alchemyst #composer require --update-no-dev --prefer-dist php-ffmpeg/php-ffmpeg #composer require --update-no-dev --prefer-dist npm-asset/dexie #composer require --update-no-dev --prefer-dist npm-asset/lozad composer require --update-no-dev --prefer-dist google/apiclient service apache2 restart chown www-data:www-data composer.* exit
```

Related links

- https://www.howtoforge.com/tutorial/tiki-wiki-ubuntu/ - Tutorial showing installation of Tiki 21 from sf.net zip file into an Ubuntu 20.04 installation with ufw enabled (untested but looks very nice)

1.1. Install Lamp

You can install Linux, Apache, Mysql and PHP (plus a few other helper utils such as git or subversion, etc) with:

```
sudo apt install mysql-server mysql-client apache2 curl composer php php-tidy php-pear memcached php-gd php-xmlrpc phpmyadmin php-mbstring libapache2-mod-php php-mysql php-apcu php-curl php-intl php-sqlite3 php-zip postfix git subversion php-memcache php-gettext php-pspell php-zip poppler-utils php-memcached bsdmainutils pstotext catdoc elinks man-db odt2txt php-pear pstotext php-common php-intl php7.2-opcache php7.2-xml php7.2-zip php7.2-ldap
```

Note
If you need to install Tiki22 or newer, since they require php 7.4, you will need to add a couple of extra repository to the server, in order to be able to install the php 7.4 packages.

```
sudo apt-get update sudo apt -y install software-properties-common sudo add-apt-repository ppa:ondrej/php sudo add-apt-repository ppa:ondrej/apache2 sudo apt-get update sudo apt -y install php7.4 php7.4-tidy php7.4-gd php7.4-xmlrpc php7.4-mbstring php7.4-mysql php7.4-apcu php7.4-curl
```

```
php7.4-intl php7.4-sqlite3 php7.4-zip php7.4-memcache php7.4-pspell php7.4-memcached php7.4-common php7.4-opcache php7.4-xml php7.4-ldap php7.4-bcmath php7.4-soap php7.4-bz2 libapache2-mod-php7.4
```

In case you have other php versions newer than 7.4 (such as php 8.0.x), you may set the default php version at 7.4 with these commands:

```
sudo a2dismod php8.0 sudo a2enmod php7.4 sudo service apache2 restart sudo update-alternatives --set php /usr/bin/php7.4
```

If it's a brand new server, you might be asked a few questions to configure some server programs.

To configure **postfix** (program to mange email sending and receiving from the server), you can answer, for instance (unless you know what you are doing):

- General type of mail configuration: **Internet site**
- System mail name: mail.yourdomain.org

To configure **phpmyadmin** (program to manage mysql databases through a web based GUI), you can answer, for instance (unless you know what you are doing):

- Webserver: **apache2** (you can select using the space bar and arrow keys if needed to move between the options)
- Configure database with dbconfig-common: yes
- MySQL application password for phpmyadmin: [leave empty]

Then you can enable the modules

```
#sudo phpenmod mcrypt # mcrypt is not found in php 7.2 sudo phpenmod mbstring
```

Extra packages for Media Alchemyst:

```
sudo apt install software-properties-common sudo add-apt-repository ppa:libreoffice/libreoffice-6-0 sudo apt install libreoffice ffmpeg unoconv ghostscript php-imagick imagemagick
```

Enable the Apache rewrite rules:

Command on a console

```
sudo a2enmod rewrite sudo service apache2 restart
```

And you can enable the self-signed ssl certificates to allow connections with https:

Command on a console

```
sudo a2enmod ssl sudo a2ensite default-ssl sudo service apache2 restart
```

And add this section between the VirtualHost tags the to this file /etc/apache2/sites-enabled/000-default.conf (or equivalent for your configuration; in this case, the doc. root where tiki is installed is **/var/www/html/**, which is where ubuntu 16.04 comes pre-configured for the base doc root):

```
sudo nano /etc/apache2/sites-enabled/000-default.conf
```

Section to add (just after the Document root line):

```
<Directory /var/www/html/> Options Indexes FollowSymLinks MultiViews AllowOverride All ## The following lines to allow connections have changed syntax ## between apache 2.2 and apache 2.4 ## See: http://httpd.apache.org/docs/2.4/upgrading.html > Run-Time Configuration Changes > Access control #Order allow,deny #Allow from all Require all granted </Directory>
```

Do the same type of edit into the equivalent config file for https:

```
sudo nano /etc/apache2/sites-enabled/default-ssl.conf
```

After the change, they should look like:

Contents of /etc/apache2/sites-enabled/000-default.conf

```
<VirtualHost *:80> # The ServerName directive sets the request scheme, hostname and port that # the server uses to identify itself. This is used when creating # redirection URLs. In the context of virtual hosts, the ServerName # specifies what hostname must appear in the request's Host: header to # match this virtual host. For the default virtual host (this file) this # value is not decisive as it is used as a last resort host regardless. # However, you must set it for any further virtual host explicitly. #ServerName www.example.com ServerAdmin webmaster@localhost DocumentRoot /var/www/html <Directory /var/www/html/> Options Indexes FollowSymLinks MultiViews AllowOverride All #Order allow,deny #Allow from all Require all granted </Directory> # Available loglevels: trace8, ..., trace1, debug, info, notice, warn, # error, crit, alert, emerg. # It is also possible to configure the loglevel for particular # modules, e.g. #LogLevel info ssl:warn ErrorLog ${APACHE_LOG_DIR}/error.log CustomLog ${APACHE_LOG_DIR}/access.log combined # For most configuration files from conf-available/, which are # enabled or disabled at a global level, it is possible to # include a line for only one particular virtual host. For example the # following line enables the CGI configuration for this host only # after it has been globally disabled with "a2disconf". #Include conf-available/serve-cgi-bin.conf </VirtualHost> # vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Contents of /etc/apache2/sites-enabled/default-ssl.conf

```
<IfModule mod_ssl.c> <VirtualHost _default_:443> ServerAdmin webmaster@localhost DocumentRoot
```

```
/var/www/html <Directory /var/www/html/> Options Indexes FollowSymLinks MultiViews
AllowOverride All ## The following lines to allow connections have changed syntax ## between
apache 2.2 and apache 2.4 ## See: http://httpd.apache.org/docs/2.4/upgrading.html > Run-Time
Configuration Changes > Access control #Order allow,deny #Allow from all Require all granted
</Directory> # Available loglevels: trace8, ..., trace1, debug, info, notice, warn, # error, crit, alert,
emerg. # It is also possible to configure the loglevel for particular # modules, e.g. #LogLevel info
ssl:warn ErrorLog ${APACHE_LOG_DIR}/error.log CustomLog ${APACHE_LOG_DIR}/access.log
combined # For most configuration files from conf-available/, which are # enabled or disabled at a
global level, it is possible to # include a line for only one particular virtual host. For example the #
following line enables the CGI configuration for this host only # after it has been globally disabled with
"a2disconf". #Include conf-available/serve-cgi-bin.conf # 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/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/certs/ssl-cert-snakeoil.pem SSLCertificateKeyFile /etc/ssl/private/ssl-cert-
snakeoil.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 # 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
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 </VirtualHost> </IfModule> # vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

1.2. Create for tiki a mysql db and a mysql user with local perms for that db

Option 1 - using console

We can create a database and user for tiki. We can do so through a terminal windows converting ourselves in root in a first step with `sudo su`, and then, we can continue with the mysql commands. Please note that when executing `mysql -+` (as root) you need to provide no password, since mysql authenticaation scheme in Ubuntu 18.04 only requires that you run this command as user root in the shell.

commands in a terminal window

```
# sudo su # mysql -p # mysql> CREATE DATABASE mytikidb CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci; # mysql> GRANT ALL PRIVILEGES ON mytikidb.* TO 'mytikiuser'@'localhost' IDENTIFIED BY 'mypassword'; # mysql> FLUSH PRIVILEGES; # mysql> \q
```

If it's a brand new server, you are encouraged to secure your mysql installation with the provided script (`sudo mysql_secure_installation`), and do the recommended setup steps indicated here: https://vitux.com/how-to-install-and-configure-mysql-in-ubuntu-18-04-lts/

Option 2 - use phpmyadmin or adminer

Alternatively, you can use your PhpMyAdmin install:
http://example.com/phpmyadmin/

or a small php script to manage your mysql server, called Adminer, that you can download to your server with an instruction like this one in a terminal:

```
sudo wget https://www.adminer.org/latest-mysql-en.php -O "/var/www/html/tiki/adminer.php" # once you are done with managing your database and users, you are encouraged to remove adminer.php from the web access to somewhere else unaccessible from your web root for extra precaution sudo mv /var/www/html/tiki/adminer.php /var/www/
```

Then you can login to your GUI web access (either phpmyadmin or adminer). You will probably not be able to login with your mysql root password, since the msyql auth scheme in ubuntu 18.04 has changed compared to previous versions (it currently uses auth_socket instead of native_mysql_password). Therefore, you can use the `debian-sys-maint` credentials that you will find here: `/etc/mysql/debian.cnf`

- Create a new user for mysql that does NOT have all perms
  - New user is **tikiuser**, and we do not select neither database nor general perms for this user.
- Go to the start page for PhpMyAdmin/Adminer again
- Create a bbdd named **tiki18svn** and as collation select utf8mb4_unicode_ci

- Go to the users tab, and go to Databases
    - Select the db tiki18svn, and check all perms except "grant" to that user for that db

And that's all. Then you will need to tell tiki that we have created:

- mysql db: tiki18svn
- mysql user: tikiuser
- mysql password: xxxxxxx (whatever you provided as password)

1.3. Use bash instead of dash

Ubuntu 18.04 comes with dash as the default shell interpreter. And Tiki shell scripts expect bash instead of dash. Therefore, one easy solution is to setup bash for your user with sudo perms in the server to use bash by default. You can do so with the command (for user *username*):

```
usermod --shell /bin/bash yourusername
```

More information:

- https://wiki.ubuntu.com/ChangingShells
- https://www.scivision.dev/ubuntu-default-shell/
- https://superuser.com/questions/46748/how-do-i-make-bash-my-default-shell-on-ubuntu

1.4. Install Tiki 21 through git

(url's taken from http://dev.tiki.org/Get+code)

We will fetch tiki21 through git shallow clone. This option is best to checkout on production servers. It uses about 187M of disk space.

Change directory to /var/www and get that tiki21 clone:

```
cd /var/www git clone --depth=1 --branch=21.x https://gitlab.com/tikiwiki/tiki.git tiki21
```

Then we run the Tiki setup script through console:

```
cd tiki21 bash setup.sh
```

The first question you will asked is about running composer to fetch the dependencies (libraries) which need to be downloaded for Tiki to run:

```
# First choose: Your choice [c]?: c # Then choose to fix file and directory permissions (classic default): Your choice [f]?: f User [www-data]: www-data Group [www-data]: www-data Multi []: # Then you can quit from the setup.sh script: Your choice [x]?: x
```

You can have a look at what the script does here:

https://gitlab.com/tikiwiki/tiki/-/blob/21.x/setup.sh

This script manages all Tiki dependencies through a program called "Composer" (it downloads and updated them when needed)

You need to make this tiki21 folder available through the webserver.
You can easily do so by means of a symbolic link

```
ln -s /var/www/tiki21 /var/www/html/tiki
```

Then you can proceed with the standard Tiki installation through the web browser (replace example.com with your domain name):
http://example.com/tiki/

and this will take us to **tiki-install.php**:
http://example.com/tiki/tiki-install.php

Tip for Spanish speakers: see the video from a course on Tiki:

- http://seeds4c.org/CT14
- http://seeds4c.org/CT14%20S1.1#Instalaci_n_del_software
  (from miunte 9' 30'' onwards)

At the screen to Setup the database connection, we provide the database name, db user and password that we previosly created and we follow instructions on screen.

At the new installation, the first user is admin, with password admin. We replace the password and continue to log in for the first time.
You have your tiki installed, ready for you to continue using the Wizards

1.4.1. Additional step for H5P

It seems that you need to create a folder for h5p to work as expected (in case you have H5P enabled in your site); otherwise, when you attempt to upload a file to a file gallery, you get an error message saying that Tiki cannot write to the corresponding folder from h5p.

You can run these commands, from the tiki root folder in the server:

```
sudo mkdir storage/public/h5p sudo chmod 775 storage/public/h5p
```

1.5. Get a free SSL Certificate from Let's Encrypt

Install a repository to get updated version of packages that will just work:

```
sudo add-apt-repository ppa:certbot/certbot sudo apt update sudo apt install python-certbot-apache
```

Before we can start to create the SSL cert, set the domain name in the vhost configuration file. Open the default vhost file with an editor:

```
nano /etc/apache2/sites-available/000-default.conf
```

and add the line:

```
ServerName example.com
```

Replace example.com with your fully qualified domain name.

Then create the SSL Certificate with this command:

```
sudo certbot --apache
```

The command will start a wizard that asks you several questions.
If you want further customization, use other options as described or linked from here:
https://certbot.eff.org/lets-encrypt/ubuntubionic-apache

1.5.1. Add some composer packages through console

If you want, you can add some packages through console

```
sudo su cd /var/www/html/ composer require --update-no-dev --prefer-dist mpdf/mpdf composer require --update-no-dev --prefer-dist fullcalendar/fullcalendar-scheduler #composer require --update-no-dev --prefer-dist xorti/mxgraph-editor # deprecated for Tiki 21.x composer require --update-no-dev --prefer-dist tikiwiki/diagram # updated for Tiki 21.x instead of mxgraph-editor composer require --update-no-dev --prefer-dist onelogin/php-saml composer require --update-no-dev --prefer-dist thiagoalessio/tesseract_ocr # composer require --update-no-dev --prefer-dist npm-asset/pdfjs-dist-viewer-min #composer require --update-no-dev --prefer-dist media-alchemyst/media-alchemyst #composer require --update-no-dev --prefer-dist php-ffmpeg/php-ffmpeg #composer require --update-no-dev --prefer-dist npm-asset/dexie #composer require --update-no-dev --prefer-dist npm-asset/lozad composer require --update-no-dev --prefer-dist google/apiclient service apache2 restart chown www-data:www-data composer.* exit
```

Alias names for this page:
UbuntuInstall | Install Ubuntu | InstallUbuntu