Ubuntu Install

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

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

```
```

**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:

```bash
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 manage 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

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

Extra packages for **Media Alchemyst**:

```bash
#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:

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

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

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

And add this section between the VirtualHost tags 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**

```xml
<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**

```xml
<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
#     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.
# 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.
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 -u` (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

```bash
# 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/](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/](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:

```bash
# 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 mysql 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 `/etc/mysql/debian.cnf` credentials that you will find here:

- 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: xxxxxxxx (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://wiki.ubuntu.com/ChangingShells)
• [https://www.scivision.dev/ubuntu-default-shell/](https://www.scivision.dev/ubuntu-default-shell/)

### 1.4. Install Tiki 21 through git

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

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 previously 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 npm-asset/dexie
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/](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:

```
```

**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-ldap php7.4-memcache php7.4-memcached php7.4-common php7.4-opcache php7.4-xml php7.4-lapd 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:

```bash
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 manage 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

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

Extra packages for **Media Alchemyst**:

```bash
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
```
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
</VirtualHost>
</IfModule>
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_SERVERCERT. 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.
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 -u root` (as root) you need to provide no password, since mysql authentication scheme in Ubuntu 18.04 only requires that you run this command as user root in the shell.

### commands in a terminal window

```bash
# 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/](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/](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:

```bash

# 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 mysql 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 `/etc/mysql/debian.cnf` credentials that you will find here:

- 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*):

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

More information:

- [https://wiki.ubuntu.com/ChangingShells](https://wiki.ubuntu.com/ChangingShells)
- [https://www.scivision.dev/ubuntu-default-shell/](https://www.scivision.dev/ubuntu-default-shell/)

### 1.4. Install Tiki 21 through git

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

```bash
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:

```bash
cd tiki21
bash setup.sh
```
The first question you will be asked is about running composer to fetch the dependencies (libraries) which need to be downloaded for Tiki to run:

```bash
# 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:


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

```bash
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 previously 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:
1.5.1. Add some composer packages through console

If you want, you can add some packages through console

```bash
sudo su
cd /var/www/html/
composer require --update-no-dev --prefer-distmpdf/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/tesseractocr
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
```

1.6. Install Lamp

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

```bash
```

Then you can enable the modules

```bash
sudo phpenmod mcrypt
sudo phpenmod mbstring
```

Extra packages for **Media Alchemyst**: 

...
<table>
<thead>
<tr>
<th>Command to execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>sudo add-apt-repository ppa:libreoffice/libreoffice-5.1</td>
</tr>
<tr>
<td>sudo apt install libreoffice</td>
</tr>
<tr>
<td>sudo apt-get install ffmpeg</td>
</tr>
<tr>
<td>sudo apt-get install unoconv</td>
</tr>
<tr>
<td>sudo apt-get install ghostscript</td>
</tr>
<tr>
<td>sudo apt-get install php-imagick</td>
</tr>
<tr>
<td>sudo apt-get install imagemagick</td>
</tr>
</tbody>
</table>

Enable the Apache rewrite rules:

**Command on a console**

<table>
<thead>
<tr>
<th>Command to execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>sudo a2enmod rewrite</td>
</tr>
<tr>
<td>sudo service apache2 restart</td>
</tr>
</tbody>
</table>

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

**Command on a console**

<table>
<thead>
<tr>
<th>Command to execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>sudo a2enmod ssl</td>
</tr>
<tr>
<td>sudo a2ensite default-ssl</td>
</tr>
<tr>
<td>sudo service apache2 restart</td>
</tr>
</tbody>
</table>

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):

**Command to execute**

<table>
<thead>
<tr>
<th>Command to execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>sudo nano /etc/apache2/sites-enabled/000-default.conf</td>
</tr>
</tbody>
</table>

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
```
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:

```
<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

</VirtualHost>```

```
<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

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

    DocumentRoot /var/www/html

  </VirtualHost>

<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
```
SSLertificateKeyFile /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
#   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.

- **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.

- **OptRenegotiate:**
  - This enables optimized SSL connection renegotiation handling when SSL directives are used in per-directory context.

```c
#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:

  - **ssl-unclean-shutdown:**
    - This forces an unclean shutdown when the connection is closed, i.e. 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.

  - **ssl-accurate-shutdown:**
    - This forces an accurate shutdown when the connection is closed, i.e. 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
  ```
1.7. Create for tiki a mysql db and a mysql user with local perms for that db

Use your PhpMyAdmin install:
http://example.com/phpmyadmin/

If you can't login with your mysql root password, you can use the {debian-sys-maint} credentials that you will find here:
/etc/mysql/debian.cnf

- Go to users: We remove debian-sys-maint user, to prevent system upgrades from eventually touching the mysql db's.
- 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 again
- Create a bdd named tik18svn and as collation select UTF8_unicode_ci
- Go to the users tab, and go to Databases
  - Select the db tik18svn, 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: tik18svn
- mysql user: tikiuser
- mysql password: xxxxxxx (whatever you provided as password)

Alternatively, if we don't use control panel we can create a database and user for tiki.

```bash
# mysql -p
# mysql> CREATE DATABASE youraccountname_tiki;
# mysql> GRANT ALL PRIVILEGES ON youraccountname_tiki.* TO 'youraccountname_yourusername'@'localhost' IDENTIFIED BY 'yourpassword';
# mysql> \q
```

1.8. 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:


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 previously 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

1.9. 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:
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/ubuntuxenial-apache

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