# CENTOS

##### SETTING FIREWALL

iptables -I INPUT 5 -i eth0 -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT

[root@dhcppc2 ~]# iptables --line -vnL

Iptables

yum install policycoreutils

yum install -y iptables-services

service iptables save

# Generated by iptables-save v1.4.7 on Sun Mar 22 12:41:57 2015

\*filter

:INPUT ACCEPT [0:0]

:FORWARD ACCEPT [0:0]

:OUTPUT ACCEPT [115:19224]

-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

-A INPUT -p icmp -j ACCEPT

-A INPUT -i lo -j ACCEPT

-A INPUT -p tcp -m multiport --dports 80,9000,3306,1996 -m state

--state NEW,ESTABLISHED -j ACCEPT

-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT

-A INPUT -j REJECT --reject-with icmp-host-prohibited

-A FORWARD -j REJECT --reject-with icmp-host-prohibited

COMMIT

service iptables restart

###### REPOSITORY, EDITOR, DAN LAIN LAIN

yum -y install epel-release yum-priorities wget nano net-tools iptables

nano /etc/yum.repos.d/epel.repo

[epel]

name=Extra Packages for Enterprise Linux 7 - $basearch

#baseurl=http://download.fedoraproject.org/pub/epel/7/$basearch

mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-7&arch=$basearch

failovermethod=priority

enabled=1

priority=10

gpgcheck=1

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7

[...]

yum update ##pas coba yang ini VPS jadi ndak bise idup ( ati ati )

yum -y groupinstall 'Development Tools'

######### LEMP

# How To Install Linux, Nginx, MySQL, PHP (LEMP) stack On CentOS 7

### Introduction

A LEMP software stack is a group of open source software that is typically installed together to enable a server to host dynamic websites and web apps. This term is actually an acronym which represents the **L**inux operating system, with the **E**Nginx web server (which replaces the Apache component of a LAMP stack). The site data is stored in a **M**ySQL database (using MariaDB), and dynamic content is processed by **P**HP.

In this guide, we'll get a LEMP stack installed on an CentOS 7 VPS. CentOS will fulfill our first requirement: a Linux operating system.

## Prerequisites

Before you begin with this guide, you should have a separate, non-root user account set up on your server. You can learn how to do this by completing steps 1-4 in the [initial server setup for CentOS 7](https://www.digitalocean.com/community/articles/initial-server-setup-with-centos-7).

**Note about SELinux:** If you run into issues with Nginx not running, make sure the SELinux context of your Nginx configuration files is correct or change the SELinux mode to permissive or disabled.

## Step One — Install Nginx

In order to display web pages to our site visitors, we are going to employ Nginx, a modern, efficient web server.

To add the CentOS 7 Nginx yum repository, open terminal and use the following command:

sudo rpm -Uvh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.ngx.noarch.rpm

Since we are using a sudo command, these operations get executed with root privileges. It will ask you for your regular user's password to verify that you have permission to run commands with root privileges.

Now that the Nginx repository is installed on your server, install Nginx using the following yum command:

sudo yum install nginx

Afterwards, your web server is installed.

Once it is installed, you can start Nginx on your VPS:

sudo systemctl start nginx.service

You can do a spot check right away to verify that everything went as planned by visiting your server's public IP address in your web browser (see the note under the next heading to find out what your public IP address is if you do not have this information already):

http://server\_domain\_name\_or\_IP/

You will see the default CentOS 7 Nginx web page, which is there for informational and testing purposes. It should look something like this:



If you see this page, then your web server is now correctly installed.

Before continuing, you will want to do is enable Nginx to start on boot. Use the following command to do so:

sudo systemctl enable nginx.service

### How To Find Your Server's Public IP Address

If you do not know what your server's public IP address is, there are a number of ways you can find it. Usually, this is the address you use to connect to your server through SSH.

From the command line, you can find this a few ways. First, you can use the iproute2 tools to get your address by typing this:

ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/\/.\*$//'

This will give you one or two lines back. They are both correct addresses, but your computer may only be able to use one of them, so feel free to try each one.

An alternative method is to use an outside party to tell you how it sees your server. You can do this by asking a specific server what your IP address is:

curl http://icanhazip.com

Regardless of the method you use to get your IP address, you can type it into your web browser's address bar to get to your server.

## Step Two — Install MySQL (MariaDB)

Now that we have our web server up and running, it is time to install MariaDB, a MySQL drop-in replacement. MariaDB is a community-developed fork of the MySQL relational database management system. Basically, it will organize and provide access to databases where our site can store information.

Again, we can use yum to acquire and install our software. This time, we'll also install some other "helper" packages that will assist us in getting our components to communicate with each other:

sudo yum install mariadb-server mariadb

When the installation is complete, we need to start MariaDB with the following command:

sudo systemctl start mariadb

Now that our MySQL database is running, we want to run a simple security script that will remove some dangerous defaults and lock down access to our database system a little bit. Start the interactive script by running:

sudo mysql\_secure\_installation

The prompt will ask you for your current root password. Since you just installed MySQL, you most likely won’t have one, so leave it blank by pressing enter. Then the prompt will ask you if you want to set a root password. Go ahead and enter Y, and follow the instuctions:

Enter current password for root (enter for none):

OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB

root user without the proper authorisation.

New password: password

Re-enter new password: password

Password updated successfully!

Reloading privilege tables..

... Success!

For the rest of the questions, you should simply hit the "ENTER" key through each prompt to accept the default values. This will remove some sample users and databases, disable remote root logins, and load these new rules so that MySQL immediately respects the changes we have made.

The last thing you will want to do is enable MariaDB to start on boot. Use the following command to do so:

sudo systemctl enable mariadb.service

At this point, your database system is now set up and we can move on.

## Step Three — Install PHP

PHP is the component of our setup that will process code to display dynamic content. It can run scripts, connect to our MySQL databases to get information, and hand the processed content over to our web server to display.

We can once again leverage the yum system to install our components. We're going to include the php-mysql and php-fpm packages as well:

sudo yum install php php-mysql php-fpm

### Configure the PHP Processor

We now have our PHP components installed, but we need to make a slight configuration change to make our setup more secure.

Open the main php-fpm configuration file with root privileges:

sudo vi /etc/php.ini

What we are looking for in this file is the parameter that sets cgi.fix\_pathinfo. This will be commented out with a semi-colon (;) and set to "1" by default.

This is an extremely insecure setting because it tells PHP to attempt to execute the closest file it can find if a PHP file does not match exactly. This basically would allow users to craft PHP requests in a way that would allow them to execute scripts that they shouldn't be allowed to execute.

We will change both of these conditions by uncommenting the line and setting it to "0" like this:

cgi.fix\_pathinfo=0

Save and close the file when you are finished.

Next, open the php-fpm configuration file www.conf:

sudo vi /etc/php-fpm.d/www.conf

Find the line that specifies the listen parameter, and change it so it looks like the following:

listen = 127.0.0.1:9000

Then save and quit.

Now, we just need to start our PHP processor by typing:

sudo systemctl start php-fpm

This will implement the change that we made.

Next, enable php-fpm to start on boot:

sudo systemctl enable php-fpm.service

## Step Four — Configure Nginx to Process PHP Pages

Now, we have all of the required components installed. The only configuration change we still need to do is tell Nginx to use our PHP processor for dynamic content.

We do this on the server block level (server blocks are similar to Apache's virtual hosts). Open the default Nginx server block configuration file by typing:

sudo vi /etc/nginx/conf.d/default.conf

Currently, with the comments removed, the Nginx default server block looks like this:

server {

listen 80;

server\_name localhost;

location / {

root /usr/share/nginx/html;

index index.html index.htm;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

}

We need to make some changes to this file for our site.

* First, we need to add an index.php option as the first value of our index directive to allow PHP index files to be served when a directory is requested
* We also need to modify the server\_name directive to point to our server's domain name or public IP address
* The actual configuration file includes some commented out lines that define error processing routines. We will uncomment those to include that functionality
* For the actual PHP processing, we will need to uncomment a portion of another section. We will also need to add a try\_files directive to make sure Nginx doesn't pass bad requests to our PHP processor

The changes that you need to make are in red in the text below:

server {

listen 80;

server\_name server\_domain\_name\_or\_IP;

root /usr/share/nginx/html;

index index.php index.html index.htm;

location / {

try\_files $uri $uri/ =404;

}

error\_page 404 /404.html;

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

location ~ \.php$ {

try\_files $uri =404;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fastcgi\_params;

}

}

When you've made the above changes, you can save and close the file.

Restart Nginx to make the necessary changes:

sudo systemctl restart nginx

## Step Five — Test PHP Processing on your Web Server

In order to test that our system is configured properly for PHP, we can create a very basic PHP script.

We will call this script info.php. In order for Apache to find the file and serve it correctly, it must be saved to a very specific directory, which is called the "web root".

In CentOS 7, this directory is located at /usr/share/nginx/html/. We can create the file at that location by typing:

sudo vi /usr/share/nginx/html/info.php

This will open a blank file. We want to put the following text, which is valid PHP code, inside the file:

<?php phpinfo(); ?>

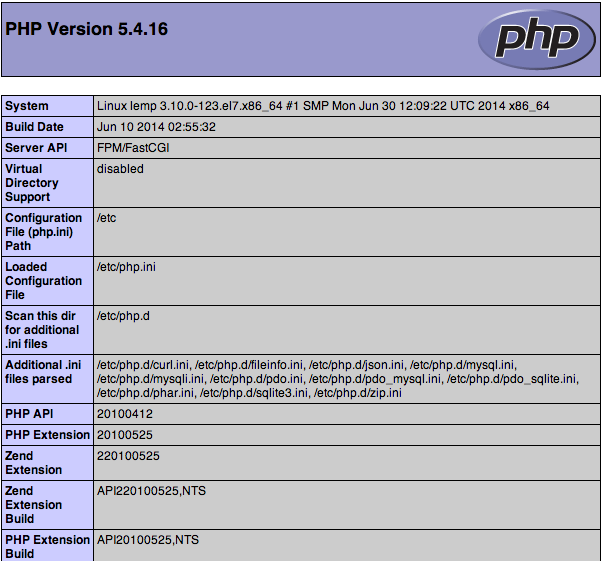
When you are finished, save and close the file.

Now we can test whether our web server can correctly display content generated by a PHP script. To try this out, we just have to visit this page in our web browser. You'll need your server's public IP address again.

The address you want to visit will be:

http://your\_server\_IP\_address/info.php

The page that you come to should look something like this:



This page basically gives you information about your server from the perspective of PHP. It is useful for debugging and to ensure that your settings are being applied correctly.

If this was successful, then your PHP is working as expected.

You probably want to remove this file after this test because it could actually give information about your server to unauthorized users. To do this, you can type this:

sudo rm /usr/share/nginx/html/info.php

You can always recreate this page if you need to access the information again later.

## Conclusion

Now that you have a LEMP stack installed, you have many choices for what to do next. Basically, you've installed a platform that will allow you to install most kinds of websites and web software on your server.

############# STATUS PHP-FPM

Nano /etc/php-fpm.d/www.conf

[uncoment]status

Nano /etc/nginx/conf.d/<watever>.conf

location ~ ^/(status|ping)$ {

access\_log off;

include fastcgi\_params;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

fastcgi\_pass 127.0.0.1:9000;

}

################ KONFIG TAMBAHAN

<http://62.108.35.100/>

ps axw -o pid,ppid,user,command | egrep '(php-fpm|PID)'

###untuk enable rewrite

location / {

rewrite ^(.\*)$ /index.php?u=$1 last;

}

### untuk lokasi di home

Chmod 775

chown

###########PHP myadmin

yum install phpmyadmin

ln -s /usr/share/phpMyAdmin /usr/share/nginx/html

t5

# UPGRADE MARIADB TO 10

MariaDB is a drop-in replacement for MySQL installed by default on CentOS 7, and offers many speed and performance improvements. MariaDB offers more storage engines than MySQL, including Cassandra (NoSQL), XtraDB (drop-in replacement for InnoDB), and OQGRAPH.

Pre-Flight Check

* These instructions are intended for upgrading from MariaDB 5.5 to MariaDB 10.0 on CentOS 7.
* I’ll be working from a Liquid Web Core Managed CentOS 7 server, and I’ll be logged in as root.

Step #1: Add the MariaDB Repository

First, you’ll follow a simple best practice: ensuring the list of available packages is up to date before installing anything new:

yum -y update

Now find which repo you should use with the [MariaDB repository generator](https://downloads.mariadb.org/mariadb/repositories/). We’re going to add the CentOS 7 (64 bit) MariaDB 10.0 repository.

For a refresher on editing files with vim see: [New User Tutorial: Overview of the Vim Text Editor](http://www.liquidweb.com/kb/overview-of-vim-text-editor/)

vim /etc/yum.repos.d/MariaDB10.repo

# MariaDB 10.0 CentOS repository list – created 2014-10-13 13:04 UTC  
# http://mariadb.org/mariadb/repositories/  
[mariadb]  
name = MariaDB  
baseurl = http://yum.mariadb.org/10.0/centos7-amd64  
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB  
gpgcheck=1

Then exit and save the file with the command :wq .

Step #2: Remove the Existing MariaDB Installation

Be sure to [backup MariaDB and/or your entire server](http://www.liquidweb.com/kb/how-to-back-up-mysql-databases-from-the-command-line/) before proceeding with the following instructions!

Removing MariaDB will remove services that depend on MariaDB!

Stop MariaDB:

systemctl stop mariadb

Remove the existing MariaDB packages:

yum remove mariadb-server mariadb mariadb-libs

Clean-up the repository cache information with the following command:

yum clean all

Step #3: Install MariaDB 10.0

At this point, installing MariaDB 10.0 is as simple as running just one command:

yum -y install MariaDB-server MariaDB-client

And then start MariaDB again:

systemctl start mysql

Be sure that MariaDB is set to start at boot:

systemctl enable mysql

Run mysql\_upgrade:

mysql\_upgrade

Verify MySQL is now MariaDB by using the command client:

mysql

Welcome to the MariaDB monitor. Commands end with ; or \g.  
Your MariaDB connection id is 4  
Server version: 10.0.14-MariaDB MariaDB Server  
  
Copyright (c) 2000, 2014, Oracle, SkySQL Ab and others.  
  
Type ‘help;’ or ‘\h’ for help. Type ‘\c’ to clear the current input statement.  
  
MariaDB [(none)]>

#### Mail with iredmail

### Set a fully qualified domain name (FQDN) hostname on your server

No matter your server is a testing machine or production server, it's strongly recommended to set a fully qualified domain name (FQDN) hostname.

Enter command hostname -f to view the current hostname:

$ hostname -f

mx.example.com

On RHEL/CentOS/Scientific Linux, hostname is set in two files:

* For RHEL/CentOS/Scientific Linux 6, hostname is defined in /etc/sysconfig/network.

HOSTNAME=mx.example.com

For RHEL/CentOS/Scientific Linux 7, hostname is defined in /etc/hostname.

mx.example.com

* /etc/hosts: hostname <=> IP address mapping. Warning: List the FQDN hostname as first item.

127.0.0.1 mx.example.com mx localhost localhost.localdomain

Verify the FQDN hostname. If it wasn't changed, please reboot server to make it work.

$ hostname -f

mx.example.com

### Disable SELinux.

iRedMail doesn't work with SELinux, so please disable it by setting below value in its config file /etc/selinux/config. After server reboot, SELinux will be completely disabled.

SELINUX=disabled

If you prefer to let SELinux prints warnings instead of enforcing, you can set below value instead:

SELINUX=permissive

Disable it immediately without rebooting your server.

# setenforce 0

### Download the latest release of iRedMail

* Visit [Download page](http://www.iredmail.org/download.html) to get the latest stable release of iRedMail.
* Upload iRedMail to your mail server via ftp or scp or whatever method you can use, login to the server to install iRedMail. We assume you uploaded it to /root/iRedMail-x.y.z.tar.bz2 (replace x.y.z by the real version number).
* Uncompress iRedMail tarball:

# cd /root/

# tar xjf iRedMail-x.y.z.tar.bz2

## Start iRedMail installer

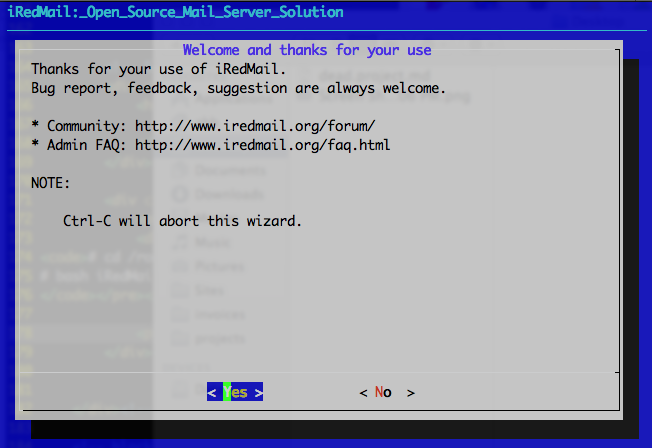
It's now ready to start iRedMail installer, it will ask you several simple questions, that's all steps to setup a full-featured mail server.

# cd /root/iRedMail-x.y.z/

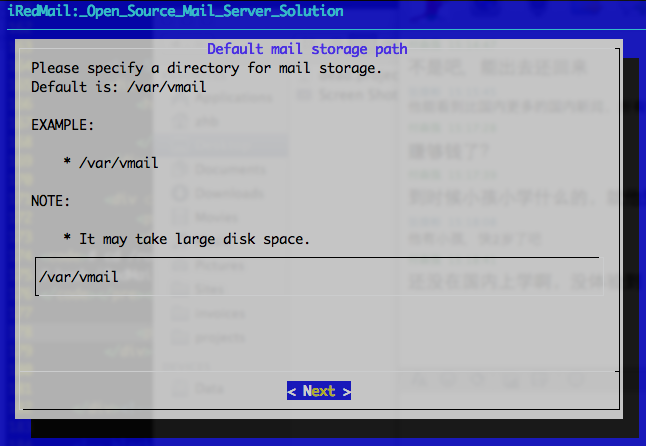
# bash iRedMail.sh

## Screenshots of installation:

* Welcome and thanks for your use



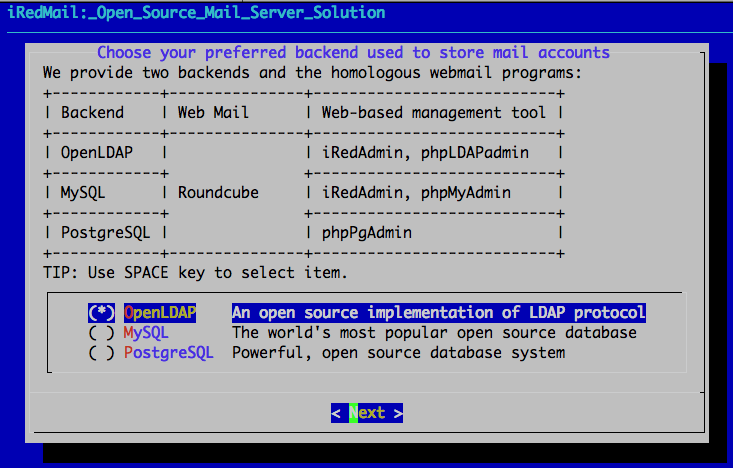
* Specify location to store all mailboxes. Default is /var/vmail/.



* Choose backend used to store mail accounts. You can manage mail accounts with iRedAdmin, our web-based iRedMail admin panel.

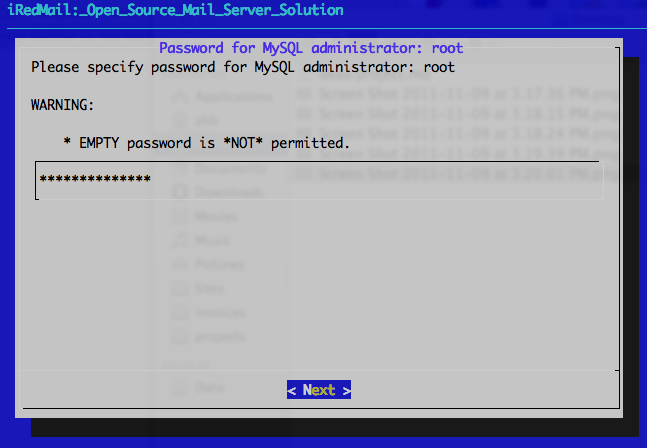
**IMPORTANT NOTE**: There's no big difference between available backends, so it's strongly recommended to choose the one you're familiar with for easier management and maintenance after installation.

( kalo kita ikutkan dari step awal ade pilihan untuk MariaDB )

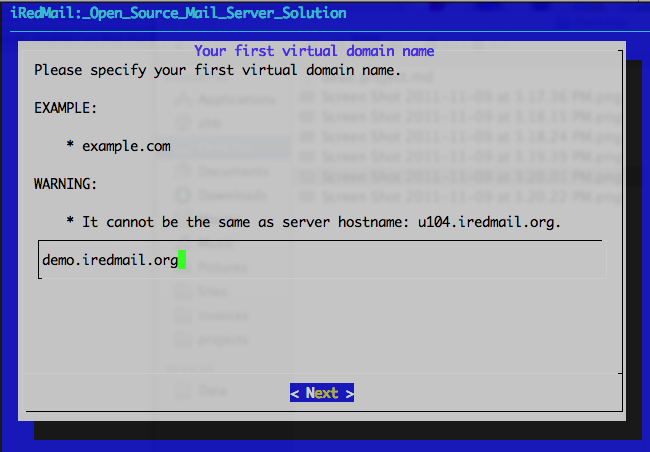


* Set password of MySQL or PostgreSQL admin user.

**NOTE**: MySQL is used to store data of other applications (e.g. Roundcube webmail, Cluebringer, Amavisd-new) if you choose OpenLDAP or MySQL as backend.

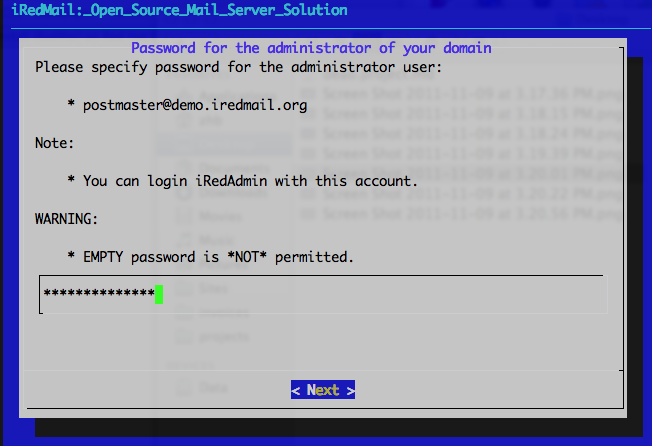


* Add your first mail domain name

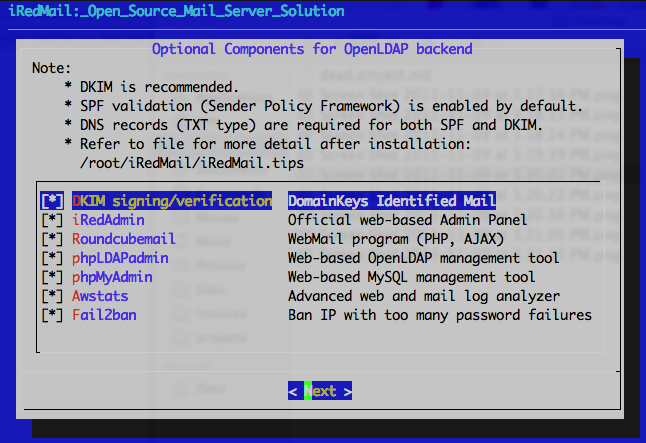


* Set password of admin account of your first mail domain.

**Note**: This account is an admin account and a mail user. That means you can login to webmail and admin panel (iRedAdmin) with this account, login username is full email address.



* Choose optional components



(setelah ini akan ade pertanyaan untuk memilih apache atau nginx, pilih jak nginx )

After answered above questions, iRedMail installer will ask your confirm to start installation. It will install and configure required packages automatically. Type y or Y and press Enter to start.

Configuration completed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* WARNING \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* Please do remember to \*REMOVE\* configuration file after installation \*

\* completed successfully. \*

\* \*

\* \* /root/iRedMail-x.y.z/config

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<<< iRedMail >>> Continue? [Y|n] # <- Type 'Y' or 'y' here, and press 'Enter' to continue

## Important things you ****MUST**** know after installation

* Read file /root/iRedMail-x.y.z/iRedMail.tips first, it contains:
  + URLs, usernames and passwords of web-based applications
  + Location of mail serve related software configuration files
  + Some other important and sensitive information
* [Setup DNS records for your mail server](http://www.iredmail.org/docs/setup.dns.html)
* [How to configure your mail clients](http://www.iredmail.org/docs/index.html#configure-mail-client-applications)
* It's highly recommended to purchase a SSL cert to avoid annonying warning message in web browser or mail clients when accessing mailbox via HTTPS/IMAPS/POP3/SMTPS. Or, you can use [free SSL cert offerred by StartSSL.com](http://www.startssl.com/?app=1).

## Access webmail and other web applications

( sebelum mulai pastikan apache (httpd) di stop kan dulu, karna biasanya ikut terinstall )

Service httpd stop  
Chkconfig httpd off

( ganti owner roundcube dari apache ke nginx )

chown -hR nginx:nginx roundcubemail-1.0.4/

Jika amavis tidak bisa start

Nano /etc/postfix/main.cf

#content\_filter = amavis:[127.0.0.1]:10024  
#smtp-amavis\_destination\_recipient\_limit = 1

* **Roundcube webmail**: <https://your_server/mail/>
* **Web admin panel (iRedAdmin)**: <httpS://your_server/iredadmin/>
* **phpLDAPadmin** (available if you choose LDAP backend): <httpS://your_server/phpldapadmin/>
* **Awstats**: <httpS://your_server/awstats/awstats.pl?config=web> (or ?config=smtp)