Systems Administration II – Spring 2195

**Lab 4 – Email and Ticketing**

# Introduction

In this lab, you will be setting up email. How you deploy your email server or servers is entirely up to you, but both domains must be able to send and receive emails. You have the option of deploying a single email server to serve both domains, or a server in each domain. In the Windows network you will need to deploy an Exchange server to provide SMTP and IMAP services. On the Linux side, you need to use PostFix for SMTP and and Dovecot as the IMAP service. You must integrate DNS with your email service.

Once you have your email service up and running you will install and configure osTicketing. Unlike your monitoring system where you set up triggers and alerts to inform your team of issues, osTicketing will be used to inform the System Administrators in your environment of problems and issues that originate from the end user.

# Lab Summary

## Linux Email Deployment with SSL

## Basic Network Setup

## SSL Configuration

## Creating The Key And Self Signed Certificate

## Generating The Certificate Signing Request (CSR)

## Configure CentOS 7 as the MTA (Postifx)

## Creating a New main.conf

## Modifying the Master.cf

## Test Smtp Via Openssl

## Configure CentOS 7 as the MDA (Dovecot)

## Recreate the dovecot.conf File

## Edit /Etc/Pam.D/Dovecot

## Test IMAP Via Openssl

## Securing Email

## Installing and Configuring SpamAssassin

## Setting up Clamav Antivirus

## Setting up Microsoft Exchnage Server

## Server Installation and Configuration

## DNS Configuration

## Enable IMAP and SMTP Protocols

## Integrating osTicket

## Install the MariaDB and PHP

## Installing Nginx and the SSl certificate

## SSL Configuration

## Email integration

# Activities

## Linux Email Deployment

## Basic network Setup

You will need clients in both domains to test the service. You will also need your email server(s) to resolve DNS, so similar to Lab 3 you will need to verify DNS resolution and enter the necessary MX resource records in your DNS server(s). Additionally, you may want to install Telnet on the server and client to test the email service locally and remotely.

Summary:

* On the email server install the Telnet service (for local testing), Postfix and Dovecot.
* On the client install the Telnet service (for possible troubleshooting) and a MTA (Mail Transfer Agent) of your choice.

You should be pretty familiar with installing software using the **yum** command, but if you need a refresher the [Red Hat Enterprise 7 System Administrator Guide](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/) is a good resource. Other resources you may find useful for this lab include the following links.

* <http://www.dovecot.org>
* <http://www.postfix.org/>

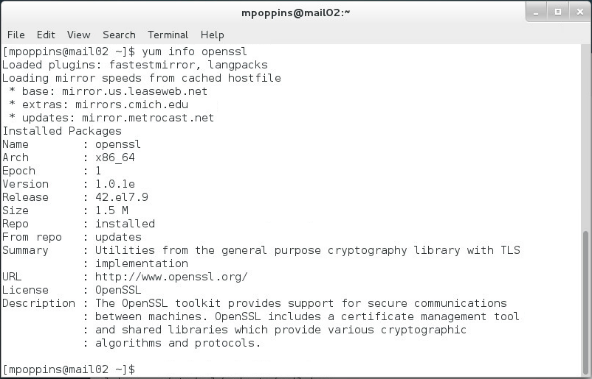
For Windows, make sure you are following the documentation for your version of Windows.

* <https://technet.microsoft.com/en-us/library/bb691354(v=exchg.160).aspx>

## SSL Configuration

In order to secure our email transmissions using SSL, we’ll want to encrypt the communication using a self-signed certificate and key. OpenSSL is installed by default on CentOS 7; if you want to check the version, type the command yum info openssl, if it is installed, it will return information similar to figure 1. While creating the certificates is not overly complicated it does involve additional configuration of the server and associated software. Keep this in mind and make note of the location of the certificate and key. You may need to create the “private” directory using the mkdir command (this directory should only be readable by root, as it will contain the private keys).

**Figure 1** – OpenSSL Information



## Creating the Key and Self Signed Certificate

To create the key and certificate type the following command as a single line in the CLI (it is a good practice to use the FQDN as the file name, such as mail.fruit.com.key and mail.fruit.com.pem, respectively, to easily identify what URL a key/cert pair is associated with).

openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout **/etc/ssl/private/mail.key** -out **/etc/ssl/certs/mailcert.pem**

**Note 1:** This key is private and should only be readable by root. Notice the location of the certificate, /etc/ssl/certs/mailcert.pem and the key, /etc/ssl/private/mail.key.

**Note 2:** It’s important you use the mail server’s fully-qualified domain name (FQDN) in the Common Name (CN) block. Most applications will show warnings, errors, or even refuse to allow access to the site if the FQDN visited does not match the CN of the certificate returned.

## Generating The Certificate Signing Request (CSR)

In a production environment, you will need to create a Certificate Signing Request (CSR). The CSR contains information about your organization and the site the certificate will be used for, and is submitted to a Certification Authority (CA), such as VeriSign, for signing. To create a CSR, type the following command as a single line in the CLI (for the purposes of this lab, a self-signed certificate, such as the mailcert.pem created in step 2.1, is sufficient).

openssl req -nodes -days 365 -newkey rsa:2048 -key /etc/ssl/private/mail.key -out mailcert.csr

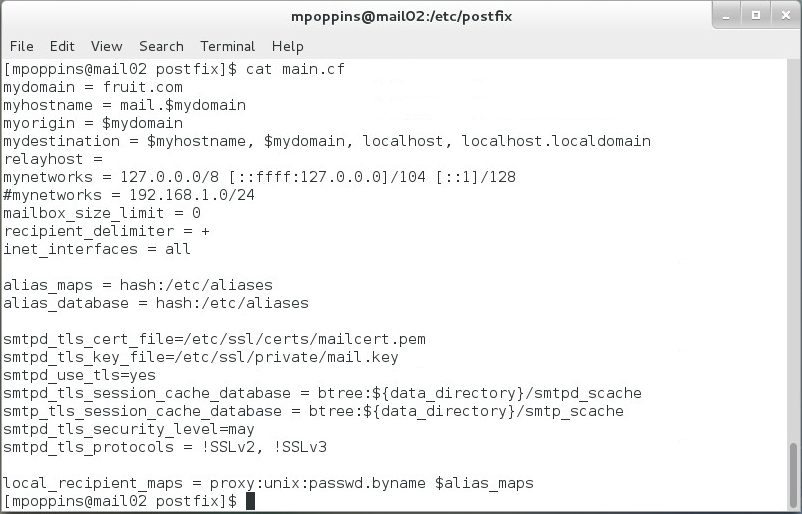
## Configure CentOS 7 as the MTA (Postfix)

For this activity we need to configure Postfix. Postfix will be responsible for receiving emails from the client to be forwarded to the mail delivery agent. You will need to edit two Postfix configuration files. It is highly recommended that for the *main.cf* and *master.cf* configuration files you make copies of the originals; we will edit the existing master.cf and create main.cf from a scratch. There are many configuration settings that impact other settings within Postfix and it will be much easier to keep track of your configuration settings with a new file. The main.cf file contains 679 lines of code, most of which we do not need to concern ourselves with, however there is plenty of information about what those settings do (in the comments), so make a copy for reference.

## Create the new Main.conf

First, make a copy of main.cf and call it main.cf.bck. Referring to figure 2 you can easily see why creating a new file is preferable to editing the original, all you need to do is add roughly 20 lines of code (Figure 2).

**Figure 2**- Sample main.cf configuration settings



Referring to figure 2, let’s cover some of the important Postfix configuration parameters. It should be obvious, but it’s worth noting that many of the configuration settings involve DNS (Figure 2, 1). Notice the mydomain, myhostname, and myorigin parameters; these are used to tell other email servers and the recipient information about where the email originated. In the example, the “*myorigin*” parameter creates the source email address of the message; when the user Mary Poppins (mpoppins) sends an email, the recipient will see that it came from [mpoppins@fruit.com](mailto:mpoppins@fruit.com), defined by the “*myhostname*” and “*mydomain*” parameters.

Referring to figure 2, 2; these parameters provide information about the network. In the example, since we are only using the local interface for testing, the loopback is used for the parameter “*mynetworks*”. To illustrate the configuration setting for an actual network, I’ve commented the statement “*#mynetworks = 192.168.1.0/24*”, if this was being used it would define a space-separated list of networks that are allowed to relay. In this lab we are only testing email within our local network so it is not needed. Finally, the “*inet\_interface*” parameter, defines the interfaces on which the Postfix service will offer SMTP. By default, it is set to the loopback only, since our device is only listening on one interface set it to “*all*”. In the real world you would want to define specific interfaces to reduce your exposure to attacks.

Next you notice many Postfix parameters used to configure SSL and certificate settings (Figure 2, 3). Early in the lab you were told to be mindful of the location of the key and the certificate locations, here you should notice the two parameters used to identify the location of these items, “*smptd\_tls\_cert\_file*” and ”*smtpd\_tls\_key\_file*”. If you are interested in learning more about the other settings visit [www.postfix.org](http://www.postfix.org) and refer to the section on “*SMTP Server specific settings*”, and click the link “*Server-side certificate and private key configuration*”. This section provides details regarding all available configuration settings and their purpose. I’ve included the link for your convenience.

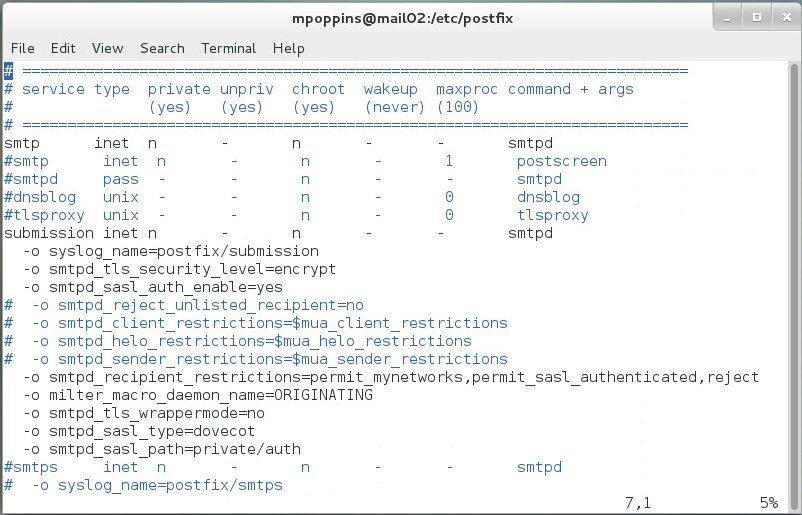
<http://www.postfix.org/TLS_README.html#server_tls>

To check for errors related to any of the configuration Postfix parameters use the **postconf** command. If you use it without any argument it will check all 816 configuration parameters in the original main.cf file, if you created the file from scratch (hopefully you did) then it will still check all the settings; try it. It should also be noted that you can modify these settings using the “-e” switch followed by the setting you want to change. For example, if I were to type **postconf mydomain**, based on the example provided it would return “**mydomain = fruit.com**” and **postconf -e mydomain=foo.bar** would change it to “foo.bar.”

## Modifying the Master.cfg file

For this activity you will be editing the master.cf file. You don’t need to make many changes to this file and these instructions tell you exactly what you need to editing it, nevertheless as a precaution it’s recommended you make a backup. First, uncomment the lines beginning with smtp and submission, as indicated by number 1 in Figure 3. Un-comment, modify, and add all the “-o” lines identified in figure 3, number 2. Follow figure 3 closely, only making changes necessary for your configuration. These lines of code are related to the smtpd process and security settings used by Postfix. If you’re interested in the details, the file has plenty of information included in the comments and you can also find more information in the Postfix master (5) manual page. There is also plenty of information on how to debug problems using the following link, http://www.postfix.org/DEBUG\_README.html.

**Figure 3** – sample master.cf configuration settings



## Test SMTP via OpenSSL

Once Postfix is configured we need to test its functionality locally, to do this we’ll use the s\_client from OpenSSL and SMTP commands. This program allows us to get a rudimentary interactive session with, as well as confirm the certificate information of, a server using SSL or TLS. The following steps illustrate a successful connection.

Let’s connect to the SMTP service by issuing the following command, using your domain name.

**openssl s\_client -starttls smtp -crlf -connect mail.fruit.com:25**

This will generate a response containing first the certificate information and TLS session ticket, then drop us into an interactive shell.

**250 DSN**

Next type…

**helo fruit.com**

This will return the following response from the server.

**250-mail.fruit.com**

Next type the command…

**mail from:** [**mpoppins@fruit.com**](mailto:mpoppins@fruit.com)

…where mpoppins is the local user. This will return the following response.

**250 2.1.0 Ok**

Next type the command…

**rcpt to:** [**mpoppins@fruit.com**](mailto:mpoppins@fruit.com)

…where mpoppins is your local user account. This will return the following response.

**250 2.1.5 Ok**

Next type the command…

**data**

**…**this will return the following.

**354 End data with <CR><LF>.<CR><LF>**

Now type whatever you would like in the body of the email message. When you are done, type the “**.**”. The server will then respond with the following message followed by an arbitrary number.

**.**

**250 2.0.0 Ok: queued as …**

Type, quit as the final step.

**quit**

If successful the server will inform you the message is located in the default mailbox, in this example, the response was…**You have mail in /var/spool/mail/mpoppins.**

## Configure CentOS 7 as the MDA (Dovecot)

Dovecot is the Mail Delivery Agent you will be using for this part of the lab. Dovecot is the default IMAP and POP3 server for Red hat Enterprise Linux and generally comes pre-installed. IMAP is preferable to POP3 because only the message header information is downloaded conserving bandwidth. In other words the message is not downloaded until the user decides they actually want to read it. This also allows the user to delete messages without having to download them. IMAP client applications can also cache the message locally removing the need to connect to the IMAP server. By doing all the work on the server side, IMAP also allows us to keep changes (read/unread/sent/deleted) synchronized across multiple clients such as web, Thunderbird on the desktop, and a smartphone. Most importantly, IMAP makes it possible to use SSL/TLS security measures which you will be configuring in this lab.

## Recreate the Dovecot.conf file

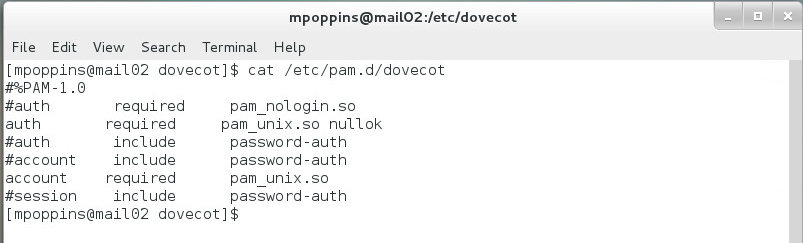
Now that we have confirmed that the SMTP service is working, we’re going to shift our focus to the IMAP service. IMAP is the protocol used by mail clients to retrieve emails located in their mailbox on the mail server. Similar to Postfix, it is recommended you back up the original *dovecot.conf* file. The modifications made to this file are fairly straight-forward. First we need to tell Dovecot how to authenticate the user and the simplest way to do this in Linux is using PAM, or Pluggable Authentication Modules, a topic that will be covered in **class**. In this file we’re going direct to Dovecot to retrieve the account information from the /etc/passwd file and Dovecot will need to use PAM for authentication (1). The next set of statements tell Dovecot that the SMTP service is being provided by Postfix and define the required permissions (2) (note the unix\_listener field, and the similarity with the smtpd\_sasl\_path field in the postfix *master.conf* file). The final set of statements activate SSL and tell Dovecot where the certificates and keys are located. Remember the names and locations where you saved these files from section 2. There is also a statement letting Dovecot know to use IMAP (4). For more information, refer to the Dovecot documentation at <http://wiki2.dovecot.org/>**.**

**Figure 4 –** Example dovecot.conf File



## Edit /etc/pam.d/dovecot

PAM or the pluggable authentication module in linux is used to grant user access to a service once the user’s identity is verified. Dovecot is a pam-aware application and the relevant settings file is located in the *“/etc/pam.d”* directory. This linux module centralizes authentication within a linux system, so we need to configure a couple lines in the pam configuration file for Dovecot. The column to the far left is the “module interface.” In this case, we need to configure authentication (auth) and the account module interfaces (account), so uncomment those lines. Since both module interfaces are “required” we can leave the middle column alone. The last column is the “module name”, and contains two shared objects (so). For the authentication interface set it to “*pam\_unix.so nullok*” and for the account interface set it to “pam\_unix.so”. And that is it for pam.

**Figure 5 –** PAM Configuration for Dovecot

## Test IMAP via OpenSSL’s s\_client

Similar to testing SMTP, we can use OpenSSL’s s\_client to test SSL and the IMAP service.

Enter the following command to test if Dovecot is listening for IMAP traffic.

**openssl s\_client -starttls imap -crlf -connect mail.fruit.com:143**

If successful the following will be returned.

**. OK Pre-login capabilities listed, post-login capabilities have more.**

Next, login using the proper credentials to verify that Dovecot is accepted the login information.

**a login mpoppins Password1**

This will return a slew of information from the server ending with something with “a OK”, if you get the “a OK” you’re in good shape.

**\* CAPABILITY IMAP4rev1…**

**a OK Logged in**

Next check that Dovecot can find the inbox for the user that just logged in by typing the following command.

**b select inbox**

This will return the following from the server.

**FLAGS (\Answered \Flagged \Deleted \Seen \Draft)**

**\* OK [PERMANENTFLAGS (\Answered \Flagged \Deleted \Seen \Draft \\*)] Flags permitted.**

**\* 1 EXISTS**

**\* 1 RECENT**

**\* OK [UIDVALIDITY 1106186941] UIDs valid**

**\* OK [UIDNEXT 2] Predicted next UID**

**b OK [READ-WRITE] Select completed.**

Finally, to verify the Dovecot configuration settings are correct for this user enter the following.

**d lsub “”\***

If the server returns the following…

**\* LSUB () "/" "INBOX"**

**d OK Lsub completed.**

Then we can confirm that the Mail Delivery Agent is working correctly.

To gracefully exit the Telnet session type the following.

**e logout**

The server will return.

**\* BYE Logging out**

**e OK Logout completed.**

## Email Security

Email is one of the most pervasive methods of gathering information about a network and users in general. Email is of particular importance because an email message as it travels across the Internet can make many stops along the way introducing countless opportunities for someone to alter the message, attach malicious code, or even use the messsage to "phish" for information in a social engineering attack. For this lab activity, you will need to install SpamAssassin to protect against unwanted and possibly malicious emails. Additionally, we’ll want to install antivirus software to protect against any viruses that hackers may attach to our user's emails.

Information on SpamAssasson and ClaimAV can be found using the following URLs.

<http://spamassassin.apache.org/>

<https://www.clamav.net/>

## Installing and configuring SpamAssassin

SpamAssassin is an open source spam filter that can be integrated with Postfix fairly easily. We’ll need to add it to our server so that we can filter suspicious or unwanted to emails.

Install the package using yum.

**yum install spamassassin**

Once we have it installed we’ll need to configure it. Edit the folling lines, by adding or uncommenting them.

**rewrite\_header Subject \*\*\*SPAM\*\*\***

**required\_hits 5.0**

**report\_safe 0**

**required\_score 5**

Next, create a new uaser group and a user to manage the service.

**groupadd spamd**

**useradd -g spamd -s /bin/false -d /var/log/spamassassin spamdchown spamd**

**chown spamd:spamd /var/log/spamassassin**

Next, we need to configure Postfix by making minor changes to the master configuration (*master.cf*) file.

First add the following line…

**-o content\_filter=spamassassin**

The entry should look similar to the following.

**smtp inet n - n - - smtpd -o content\_filter=spamassassin**

Next, we need to update with the SpamAssassin rule by adding the following to the end of the file.

**spamassassin unix - n n - - pipe flags=R user=spamd argv=/usr/bin/spamc -e /usr/sbin/sendmail -oi -f ${sender} ${recipient}**

Save, the changes and update the SpamAssassin rules by entering the following command.

**sa-update --nogpg**

Finally, restart SpamAssassin and Postifx.

## Setting up Clamav antivirus

ClamAV is an open source antivius utility and it is used to detect virus and malware in emails on Linux based systems. For this activity you will be installing and testing ClaimAV on your email server.

First install the EPEL (*Extra Packages for Enterprise* Linux) repository.

**yum install epel-release**

Next, install Clamav and all of its useful tools and **there are a lot!**

**yum install clamav-server clamav-data clamav-update clamav-filesystem**

**clamav clamav-scanner-systemd clamav-devel clamav-lib clamav-server-**

**systemd**

Set the proper SELinux context so that SELinux will let ClaimAV work with our server.

**setsebool -P antivirus\_can\_scan\_system 1**

We can check to make sure the option is active by typing the following command.

**Getsebool –a | grep virus**

Claimav comes with a default configuration, but we need to create our own. To do this we need to remove the example by typing the following command.

**sed -i '/^Example/d' /etc/clamd.d/scan.conf**

Now we need to edit the configuration file, it is located in **/etc/clad.d/scan.conf**.

**Uncomment** the folloing line.

**LocalSocket /var/run/clamd.scan/clamd.sock**

**Add** the next two lines at the end of the file and save.

**User root**

**LocalSocket /var/run/clamd.<SERVICE>/clamd.so**

We also want to make sure that the Claimav virus defeinitions are kept up to date. To do this, we need to enable **Freshclam**. But first, we should **backup the original configuration file** in case we screw it up. It happened to me and ate up 2 hours of my time. To backup the configuration make a copy.

**cp /etc/freshclam.conf /etc/freshclam.conf.bak**

Again, we need to remove the example that is provided.

**sed -i '/^Example/d' /etc/freshclam.conf**

Otherwise, if needed, we can adjust the options for a more personalized solution. We need to run Freshclam to update the database manually and to check whether the configuration is successfully set by typing **freshclam** at the command prompt.

Next, we need to create the file that will act as the service file to run the Freshclam daemon process. The file will need to be created in the system directory.

**vim /usr/lib/systemd/system/clam-freshclam.service**

Add the following code to the file.

**[Unit]**

**Description = freshclam scanner**

**After = network.target**

**[Service]**

**Type = forking**

**ExecStart = /usr/bin/freshclam -d -c 4**

**Restart = on-failure**

**PrivateTmp = true**

**[Install]**

**WantedBy=multi-user.target**

To check to see if the configuration is valid **start** the service and check the **status** by entering the following commands. Please Note: That the **–l** is the letter “el”.

**systemctl start clam-freshclam.service**

**systemctl status clam-freshclam.service -l**

If you run into any problems, use /var/log/messages and journalctl –xe to troubleshoot. Once everything check out, add it to the system startup.

**systemctl enable clam-freshclam.service**

Our next step is to create the Clamav service file. We begin by copying the example service file into the system services folder and change the name to something meanful.

**mv /usr/lib/systemd/system/clamd@.service /usr/lib/systemd/system/**

**clamd.service**

Since, the name of the file was changed, it will also need to be changed to the file that uses the service.

**/usr/lib/systemd/system/clamd@scan.service**

Change the first line by removing the @ to look like this:

**.include /lib/systemd/system/clamd.service**

Next, in the same location we need to edit the Claimav service file by adding the following lines at the end of the file.

**[Install]**

**WantedBy=multi-user.target**

Remove the **%i** from both the Description and ExecStart options, so it looks like the following.

**Description = clamd scanner daemon**

**ExecStart = /usr/sbin/clamd -c /etc/clamd.d/scan.conf --nofork=yes**

To check for any errors, run the following command. To become familiar with the command and parameters refer to the Claimav documentation.

**/usr/sbin/clamd -c /etc/clamd.d/scan.conf –nofork=yes**

If everything checks out, **enable** the service to run on boot and **start** it.

**systemctl enable clamd.service**

**systemctl enable clamd@scan.service**

**systemctl start clamd.service**

**systemctl start clamd@scan.service**

For a final verification check the status. Please note that the flag at the end is the letter “el” and not the number one..

**systemctl status clamd.service -l**

**systemctl status** [**clamd@scan.service**](mailto:clamd@scan.service) **-l**

Run and test scam but typing the following command.

**clamscan --infected --remove --recursive ./**

Read the ClaimAV documentation to understand how this command works.

## Setting up Microsoft Exchange Server

For this activity you will be installing and configuring Microsoft Exchange for the Mail Transfer and mail Delivery Agent. For this activity use Windows Server 2012 or 2012 R2, do not use Windows Server 2016, because, at the time of this writing, Exchange 2016 is not compatable with Server 2016.

**Please Note: The instructions for installing and Configuring Microsoft Exchange and osTicketing were provided by Colum McGaley.**

## Server Installation and Configuration

Add the mail server to yout existing Windows domain environment and have it join the domain, i.e. authenticate with a user account and add a computer entry to the AD DS server. You can use Windows Powershell to do this by typing the following command.

**Add-Computer –DomainName “windowsworld.com”**

Once, the server has joined the domain and you are logged in as a domain user, install the server from the Acedemic cloud.

Begin by authenticating to the Acedemic cloud with the following command, using your RIT account. The following command will allow you to access the Acedemic Cloud and the filehsare where the installer is located.

**net use \\itsnas02.main.ad.rit.edu\AcedemicCloud**

The output should look similar to Figure 6.

**Figure 6** – Successful Authentication to the Acedemic Cloud



Make sure your Windows system has the following features installed, if not use the Microsoft documentation to install them,

* Windows Management Framework 4.0 (Windows6.1-KB2819745)
* Unified Communications Managed API 4.0 (UcmaRuntimeSetup.exe)
* Microsoft Office 2010 Filter Packs
* The Remote Server Administration Tool (RSAT)

Once, you have verified the following prerequisites are installed and are part of the domain, retrieve the installer using the following command.

**\\itsnas02.main.ad.rit.edu\AcademicCloud\Software\Microsoft\Exchange2013\setup.exe**

It is recommended that you copy the folder to your local server before conitinuing.

The are two parts to the installation procedure, the Mailbox and Client Access.

First, using the PowerShell CLI interface, install the Mailbox using the following command.

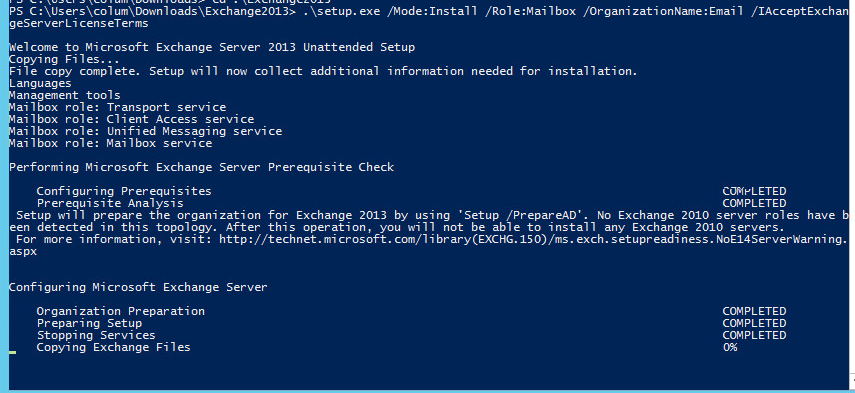
**.\setup.exe /Mode:Install /role:Mailbox /OrganizationName:Email /IAcceptExchangeServerLicenseTerms**

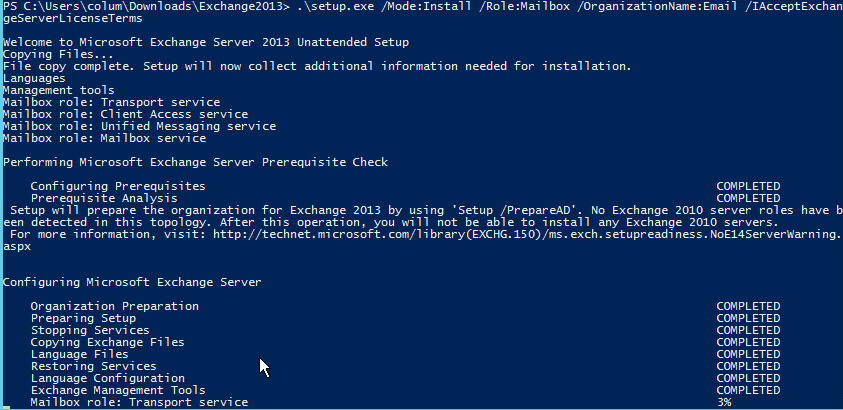
Then install Client Access by entering this command.

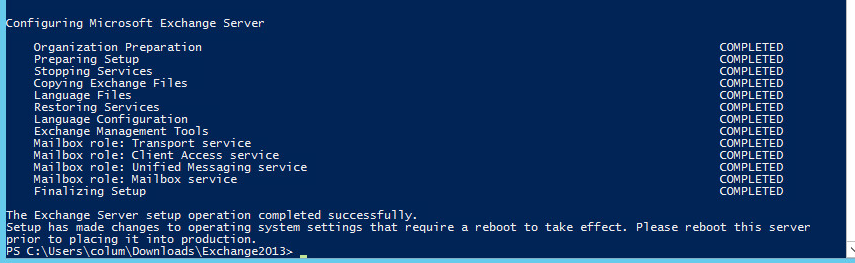
**.\setup.exe /Mode:Install /role:ClientAcces /IAcceptExchangeServerLicenseTerms**

During the installation you should see output similar to Figure 7.

**Figure 7** – Exchange Installation Output







Once the installation is complete reboot the server.

## DNS Configuration

Using Powershell you will update the DNS configuration to add the MX, A, and CNAME records to the zone database files. You may refer to the [Micrsosoft Documentation](https://technet.microsoft.com/itpro/powershell/windows/dnsserver/add-dnsserverresourcerecord) or follow the steps provided.

Add the MX resource record using the following command.

**Add-DnsServerResourceRecord –Name “.” –ZoneName email.lab –MailExchange “mail.email.lab” –Preference 10**

Add the following A records using the IP address of your mail server.

**Add-DnsServerResourceRecord –ZoneName email.lab –A –Name “mail” –Ipv4Address “172.16.7.6”**

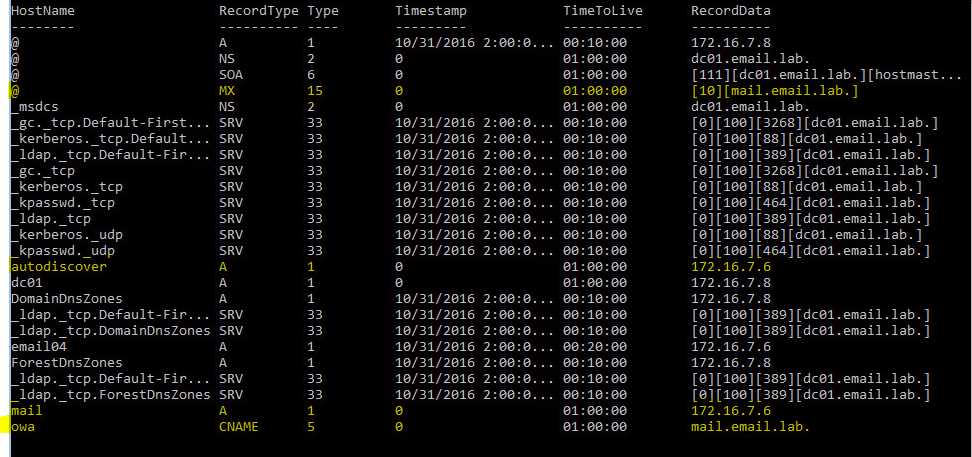
**Add-DnsServerResourceRecord –ZoneName email.lab –A –Name “autodiscover” –Ipv4Address “172.16.7.6”**

Optionally, you can add the CNAME record to hid the actual FQDN of the mail server. The following example, maps “owa” to the FQDN “*mail.email.lab.*”

**Add-DnsServerResourceRecord –ZoneName email.lab –Cname –Name “owa” –HostNameAlias “mail.email.lab”**

Use the **Get-DnsServerResourceRecord** to confirm the settings, see Figure 8.

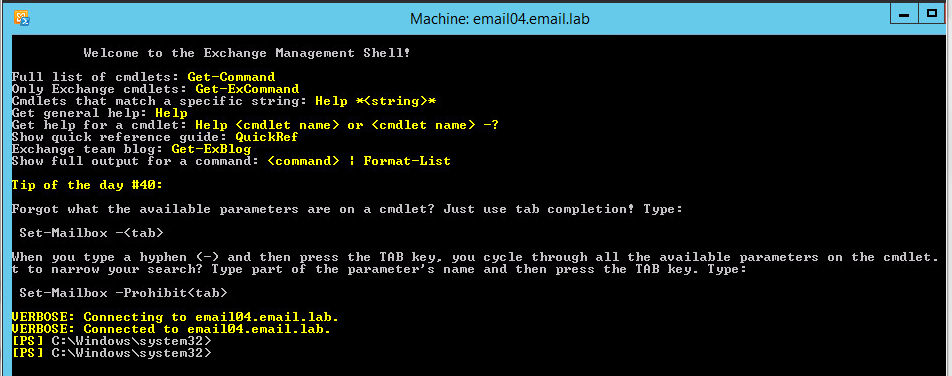
**Figure 8 –** Added DNS resource records



Next we need to open the Exchange Management Shell. To do this, navigate to **Start 🡪 All Apps 🡪 Microsoft Exchange Server 2016 🡪 Exchange Management Shell.**

Once, you open the management shell you will be presented with the Welcome Screen, Figure 9.

**Figure 9** – Microsoft Management Shell – Welcome Screen



To forward emails to other mail servers we’ll need to setup the required DNS entries and server configurations using the New-SendConnector command. Enter the following using your domain.

**New-SendConnector –Internet –Name Connector1 –AddressSpaces email.lab**

Next, using the follow resources create a couple mailboxes for your domain users and enable them.

<https://technet.microsoft.com/en-us/library/aa997663(v=exchg.160).aspx#Examples>

<https://technet.microsoft.com/en-us/library/aa998251(v=exchg.160).aspx>

Document the procedure for how to create and enable a mailbox in your Wiki.

Once you have completed these step open a web browser and enter <https://owa.email.lab/owa>, to open the Outlook Web App and verify you can authenticate to the mailbox.

## Enabling IMAP

By default the SMTP service is running, which is fine for forwarding emails, but we also want to be able to retrieve them from our mailbox and we’ll need to enable to IMAP protocol to do this. Enter the following commands.

**Set-Service MSEcxchangeIMAP4BE -StartupType automatic**

**Set-Service MSEcxchangeIMAP4 -StartupType automatic**

Test the service my sending and receiving emails to users within the local domain.

## Integrating osTicket

OsTicket is a popular ticketing software that is simple to setup and use. We will need two things for this component. First, an Email account called helpdesk and a mailbox for that account. Secondly, a Linux virtual machine to install and run osTicketing.

## Install the MariaDB and PHP

Using the yum package manager install the MariaDB and PHP packages and required dependencies. As, always update your system. At this point installing packages should be a cake walk. Start up both services and configure the database.

To create the MariaDB for osTicketing run the mysql\_secure\_installation script to configure MariaDB for osTicketing. You will need to create an osTicket user and the database.

create database osticketdb  
CREATE USER 'osticketuser'@'localhost' IDENTIFIED BY 'password';  
GRANT ALL ON osticketdb.\* TO 'osticketuser'@'localhost';  
flush privileges;  
exit;

For PHP install the following packages.

php-mysql php-gd php-ldap php-odbc php-pear php-xml php-xmlrpc php-mbstring php-snmp php-mcrypt php-imapsy

Nginx cannot spawn PHP processes by itself, so we need to use PHP-FPM to do it for us. Apache manages the PHP processes internally, which is why PHP is so easy to setup on Apache, but for nginx we need to use PHP-FPM.

PHP-FPM needs very little configuration beyond the default configuration. The only thing that we need to change the user and group that PHP-FPM spawns under. By default PHP-FPM will spawn under the user and group apache. Go into */etc/php-fpm.d/www.conf* and change the user and group to **nginx.**

We also need to make sure the session folder exists. If **/var/lib/php/session** exists and is owned by the user and group nginx

Now, we can enable and start the service.

## Install Nginx and the SSL Certificate

For this lab, we are going to add in a SSL certificate to our osTicket server.

Create the private key for our certificate by entering the following command.

**openssl genrsa –rsa -out server.key 2048**

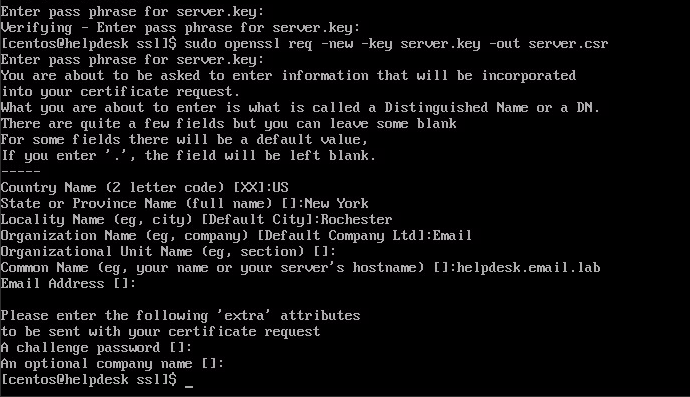
When asked to enter a password, use Password1 and document the information for future use and your Wiki.

Next, generate the Certificate Signing Request (CSR) by entering the following command.

**openssl req -new -key server.key -out server.csr**

You will then be promted to enter information pertaining to your institution. Document in your Wiki the information that needs to be included in the CSR for your organization. The information contained in the CSR is essentially used to sign the certificate, normally this would be sent to the SSL provider but for our purposes you will only be using it locally.

**Figure 10 –** Example Certificate Signing Request



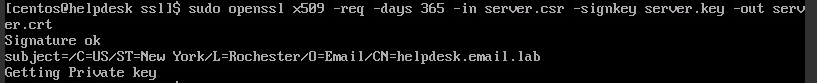
Now that we have the key and the CSR, we can create the certificate. First, we need to strip off the password from the key by entering the following:

**openssl rsa –in server.key -out server.key**

Create the certificate by entering the following information. The output should look similar to Figure 11.

**openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt**

**Figure 11** – Certifcate Confirmation

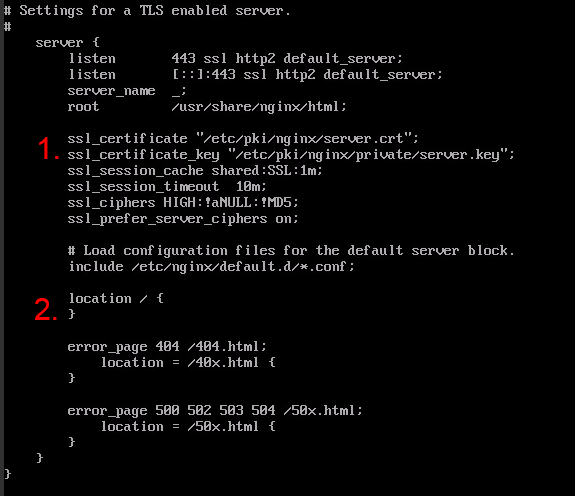


## nginx and SSL Configuration

Now that we have the certificate we can configure nginx. Make the following changes to the nginx configuration file located in /etc/nginx/nginx.conf. Using the information in Figure 12, uncomment the required lines and add the location of your certificate and key.

1. Edit these lines to identify the location of the files on your system.
2. Beneath the “location” directive, add another one for PHP

**Figure 12** – Nginx TLS configuration settings



Add the following information under the location directive in the nginx.conf file, see Figure 13.

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;   
}

**Figure 13** – Nginx Location Directive



The final step is to restart nginx and open a web browser to access it, Figure 14.

**Figure 14** – Nginx Home Page



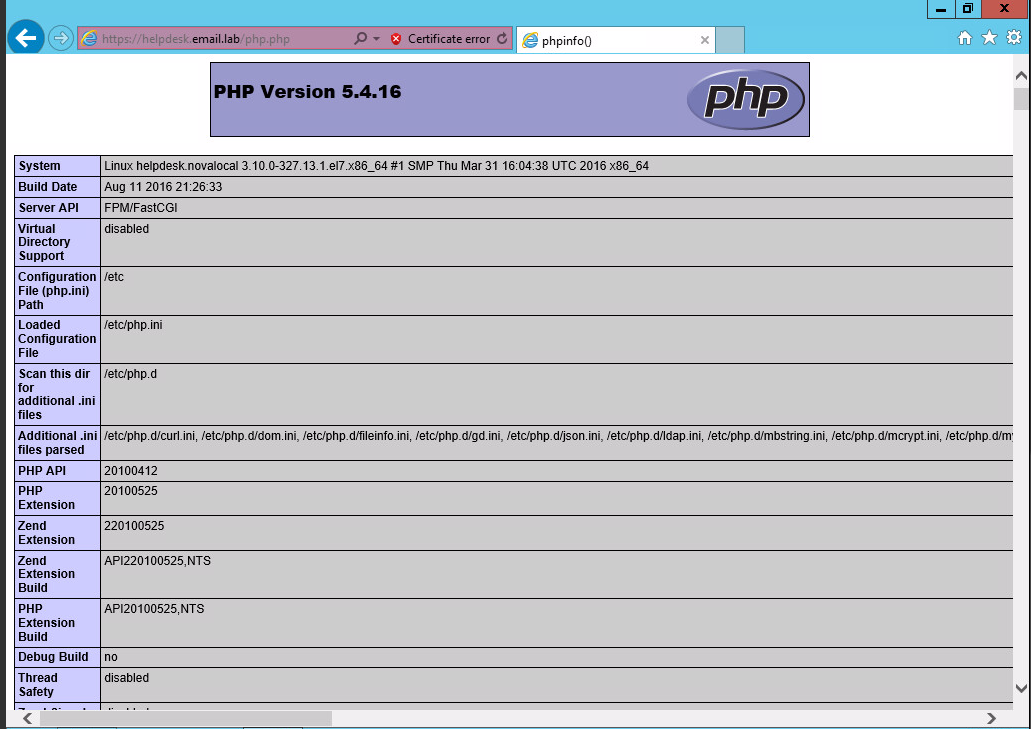
To verify PHP is configured and working correctly go to /usr/share/nginx/html and create a file called php.php. Add the following text to the file and save it.

<?php

phpinfo();

Open another tab in the web browser and navigate to <https://helpdesk.domain.com/php.php> this will produce a web page similar to Figure 15.

**Figure 15** – PHP Information



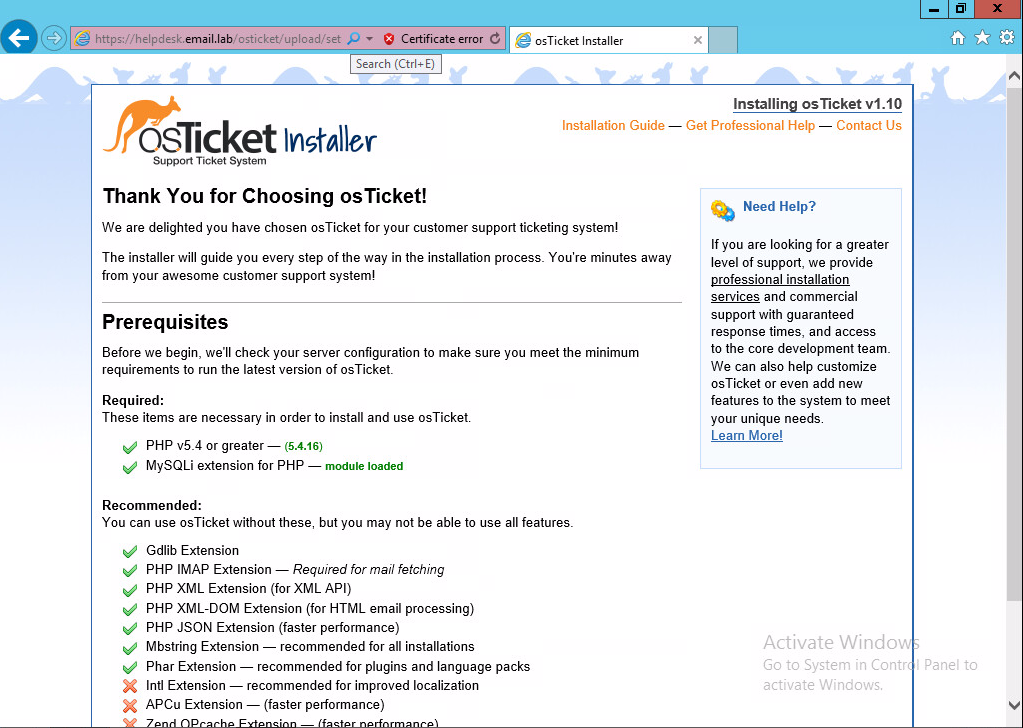
Now that we have confirmed PHP is working, we can install osTicket.

Download the compressed file from <http://osticket.com/sites/default/files/download/osTicket-v1.10.zip>. Create, a directory and unzip it, there are two directories in the zip file, *scripts* and *upload*. Confirm that the folder permissions are set to 0755. Be mindful of where the extracted files are located because you will need to know the location.

Navigate to <https://helpdesk.yourdomain.com/osTicket/upload>.

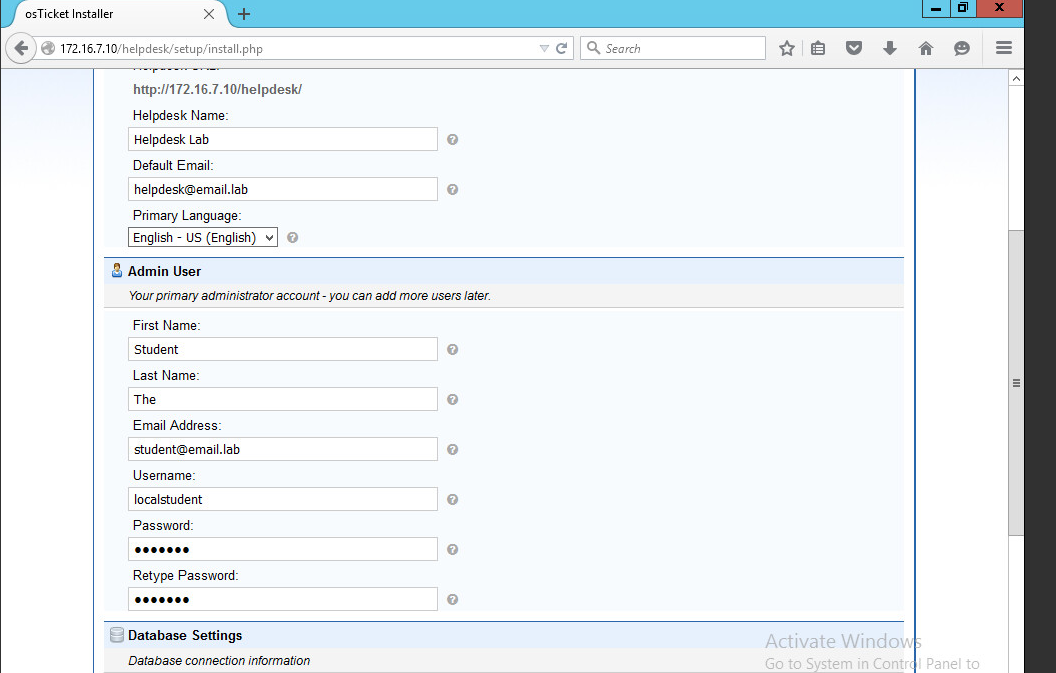
This should bring up a web page similar to Figure 16.

**Figure 16** – osTicket Installer



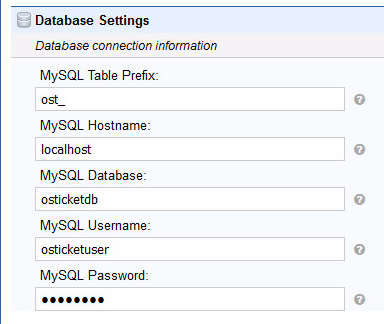
Next, create the Administrator user for osTicketing. Do not use an existing Actve Directory account, create an entirely new account, see Figure 17.

**Figure 17** – Creating the Admin User Account



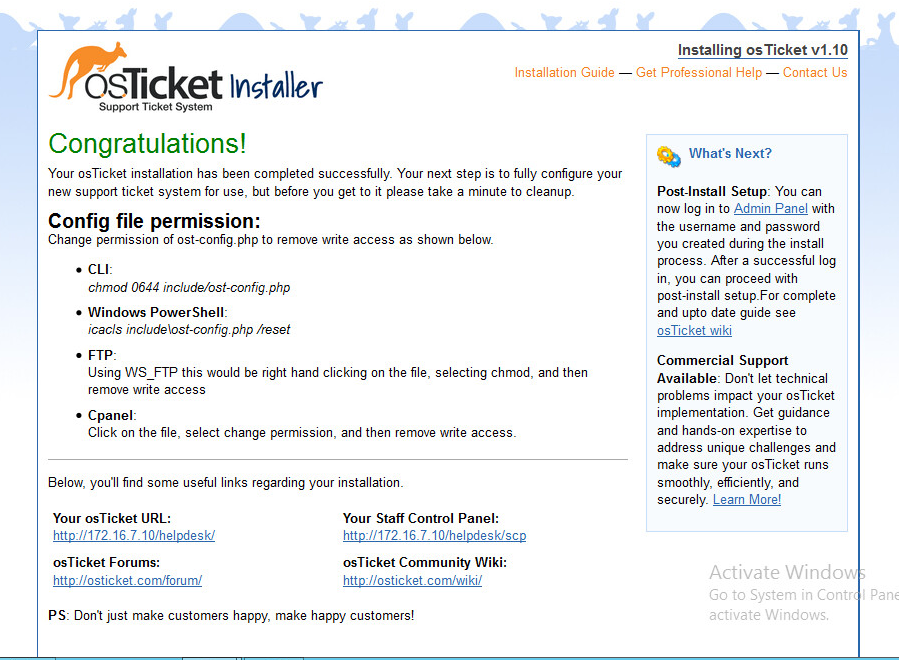
Confirm the MariaDB settings, Figure 18.

**Figure 18** – MariaDB Settings



Once the installation is complete you will be congratulated, Figure 19.

**Figure 19** – osTicket Installation Complete Confirmation



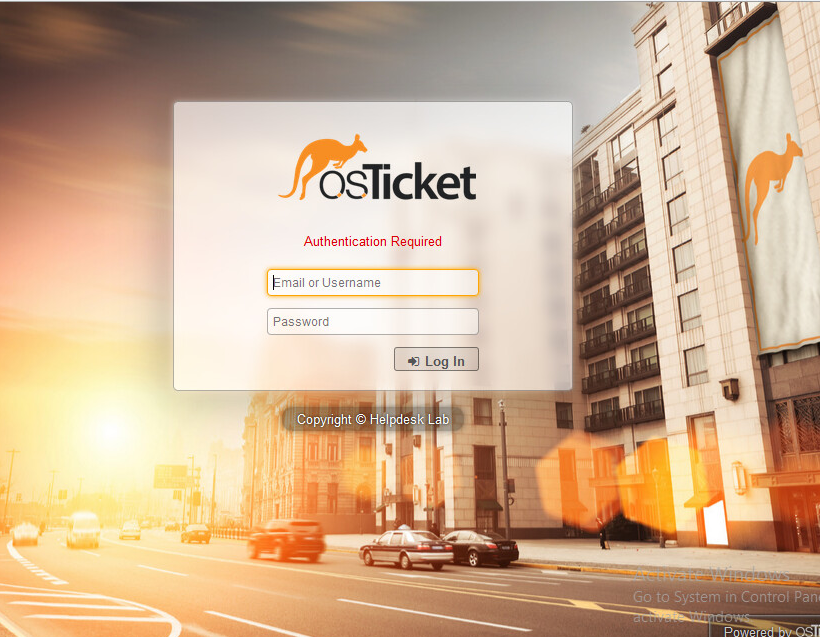
Finally, create a crontab entry to check for tickets every 5 or so seconds, this will be enabled later on as part of the osTicketing configuration.

**\*/5 \* \* \* \* nginx php /usr/share/nginx/html/helpdesk/api.php**

## Email Integration

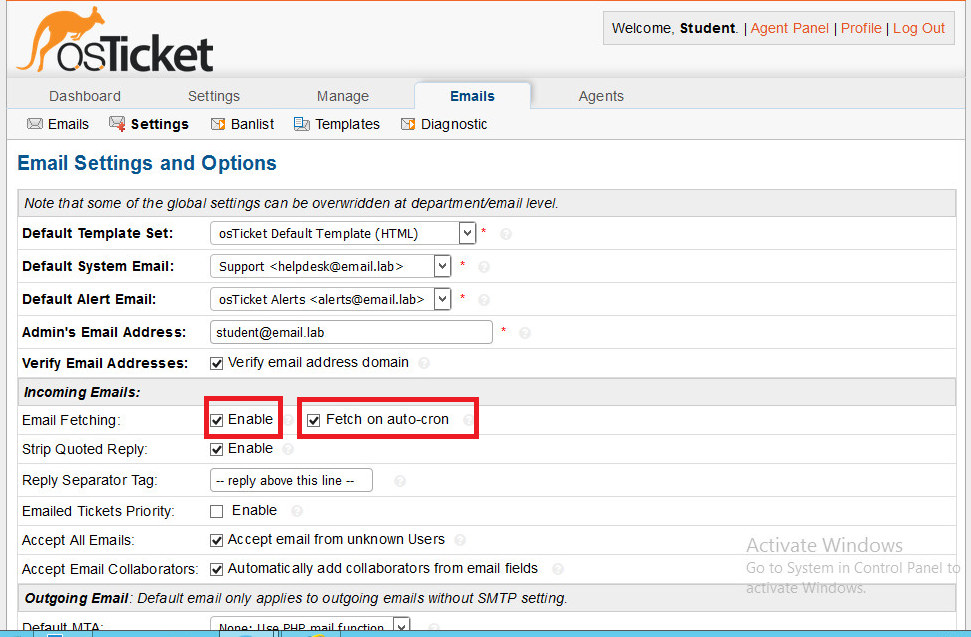
Now we need to start receiving tickets. To do this, open a web browser a login, using the credentials created earlier, Figure 20.

**Figure 20** – osTicket Log In Screen

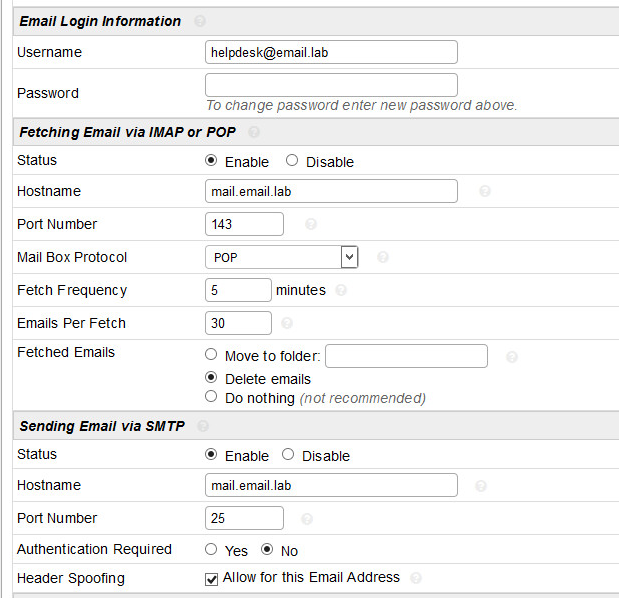


Configure osTicket to receive emails from the email server by navigating to *Email* ***🡪*** *Settings* and enable “*Email Fetching*”, by checking the boxes for **Enable** and **Fetch on auto-cron** (remember the crontab you created earilier?), see Figure 21. Configure other email settings as illustrated in Figure 22.

**Figure 21** – Configuring Incoming Emails

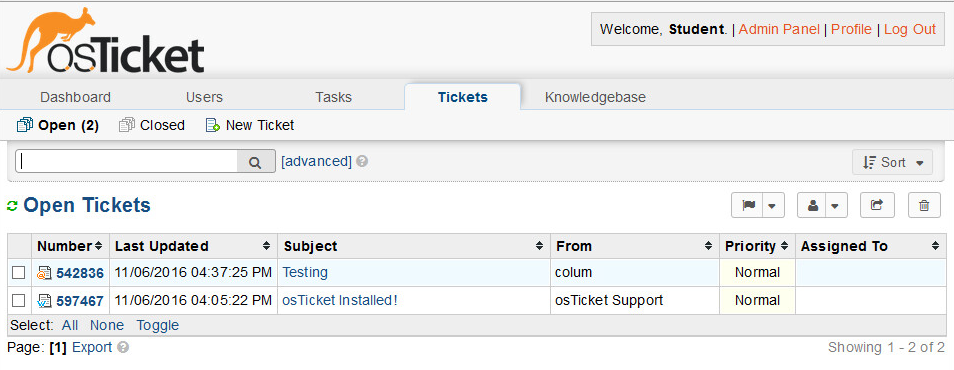


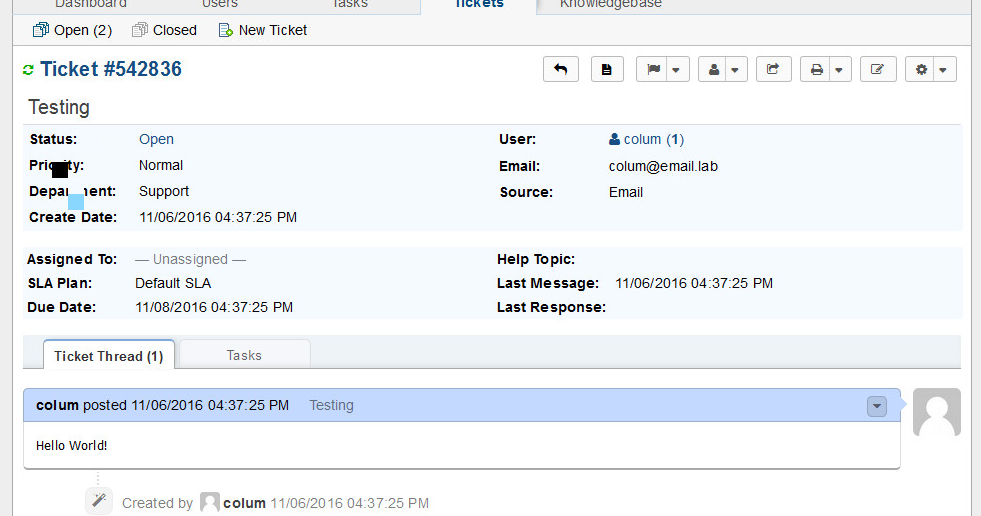
**Figure 22** – Additional Email Configuration Settings



Now that we have osTicketing configured and validated send a test email, depending on how long you configured the cron job to run the email should appear in the queue, <https://helpdesk.email.lab/helpdesk/scp/index.php>, see Figure 23.

**Figure 23** – Open Tickets





# Grading

Completion of these activities will be reviewed in RLES.