

Mesh Network Server Handbook for Emergency Responders

August 2018

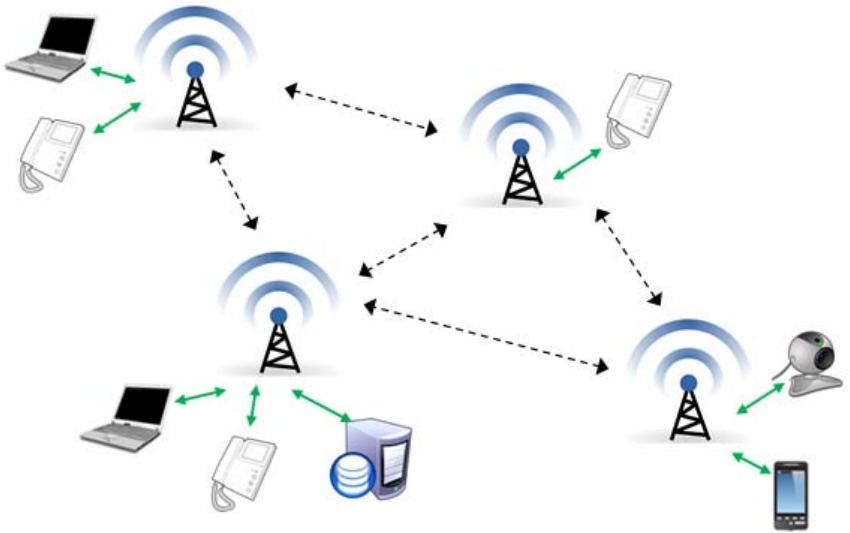


Table of Contents

- 1 INTRODUCTION..... 3**
- 2 COMMON LINUX COMMANDS..... 4**
- 3 SAMBA SERVER CONFIGURATION IN UBUNTU 14.04 LTS..... 5**
 - 3.1 BEFORE YOU BEGIN 5
 - 3.2 ANONYMOUS SAMBA SHARING 5
 - 3.3 SECURED SAMBA SERVER 8
- 4 ASTERISK & PHONE SETUP12**
 - 4.1 OVERVIEW 12
 - 4.2 CREATE THE SIP.CONF FILE 13
 - 4.3 CREATE THE EXTENSIONS.CONF FILE 14
 - 4.4 SETTING UP A ZULTYS ZIP2 PHONE..... 15
- 5 APACHE VIRTUAL HOST SETUP17**
 - 5.1 OVERVIEW 17
 - 5.2 INSTALLATION..... 17
 - 5.3 CREATE THE DIRECTORY STRUCTURE 17
 - 5.4 GRANT PERMISSIONS 18
 - 5.5 CREATE DEMO PAGES FOR EACH VIRTUAL HOST 18
 - 5.6 CREATE NEW VIRTUAL HOST FILES 19
 - 5.7 CREATE THE FIRST VIRTUAL HOST FILE..... 19
 - 5.8 ENABLE THE NEW VIRTUAL HOST FILES 21
 - 5.9 SET UP LOCAL HOSTS FILE (OPTIONAL) 22
 - 5.10 TEST YOUR RESULTS 23
 - 5.11 CONCLUSION 23
- 6 FTP SERVER.....24**
 - 6.1 OVERVIEW 24
 - 6.2 INSTALLATION..... 24
 - 6.3 ANONYMOUS FTP CONFIGURATION 24
 - 6.4 USER AUTHENTICATED FTP CONFIGURATION..... 25
 - 6.5 SECURING FTP 25
- 7 TFTP SERVER28**
 - 7.1 OVERVIEW 28
 - 7.2 INSTALLATION AND SETUP 28
 - 7.3 TESTING THE TFTP SERVER..... 29

1 Introduction

This guide is a set of validated procedures for configuring an Ubuntu server for use in a networked environment. All of the procedures listed here are excerpts or copies from excellent work posted on the internet. Links and authors are listed when known. If there were any changes made by me, it was in the area of readability, conciseness, and/or taking the writing from 1st person to 3rd person.

While I cannot guarantee that this document will work with all Linux distros, it does work on my Ubuntu 14.04 LTS. If anything, this document may offer the reader a pointer to platform specific information they he/she may encounter, or provide a sense of the complexity in performing one of these setups.

All screen shots and examples were done either on my home (192.168.1.0) or my Mesh network, and generally apply to any deployment. For the real deployment, my intent is to use this server as part of an emergency wireless mesh network in support of ARES/RACES.

System Processes

There are plenty of applications that could be deployed on any Linux server. However, for ARES/RACES purposes, those listed here are in direct support of an emergency response deployment. What's covered in this doc are listed here in no particular order of importance:

1. Samba. File sharing; supports creating file shares on a central server accessible by users on the network. As more responders bring their computers to an event, having the ability to share data can improve the effectiveness of the response.
2. Asterisk PBX. Software implementation of a telephone private branch exchange (PBX). Asterisk allows attached telephones to make calls to one another, and to connect to other telephone services, such as the public switched telephone network (PSTN) and Voice over Internet Protocol (VoIP) services. An ad-hoc phone system could be configured and set up at an Incident Command Post (ICP) thereby almost creating an office-like work environment.
3. Apache. A web server used to serve up Web Pages requested by client computers using Web Browsers such as Edge, Firefox, Opera, Chrome, or Mozilla. Event information, forms, or other services can be hosted on an ICOP website for use by responders as well as the general public that may connect to the network.
(NOTE: PHP and Mysql are also load candidates)
4. ftp Server. An application running the File Transfer Protocol (FTP) that supports exchanging files over the Internet (or network). With a smart phone FTP client app and a Wireless Access Point on the network, a user could take pictures or videos and upload them for sharing with the ICP.

5. tftp Server. Trivial File Transfer Protocol is a simple high-level protocol for transferring data that servers use to boot diskless workstations, X-terminals, routers, and in our case, VoIP phones.

2 Common Linux commands

Network Mgmt	App to add, remove, and modify network connections (sudo). nm-connection-editor
Network, config	ifconfig [-help]
Service, Restart	service <name> restart
File Commands	link
File, move or rename	mv <oldfilename> <newfilename>
File, copy	cp <oldfilename> <newfilename>
File, remove/delete	rm <filename>
Directory, create	mkdir <dirname>
File, List	ls [-al]
File, permissions	chmod <0777> <filename> owner group anybody read write execute
File, ownership	chown -R <newowner> <directory>
File, rename	rename <orig_file_name> <new_file_name>
File, delete	rm <file_name>
Directory, create	mkdir <dirname>
Directory, remove	rmdir <dirname>
User, add	useradd <username> useradd -g <groupname> username
User, assign password	passwd <username> Prompts twice for unix password Entry is added to file /etc/passwd
User, look at his groups	id <username>
Group Commands	link
Group, add	groupadd <groupname>
Group, assign user	usermod -g <groupname> <username> usermod -a -G ftp,admins,othergroup <username>
Group, user association	groups <username>

3 Samba Server Configuration in Ubuntu 14.04 LTS

Reference: (<https://www.howtoforge.com/samba-server-ubuntu-14.04-lts>), author: Srijan Kishore.

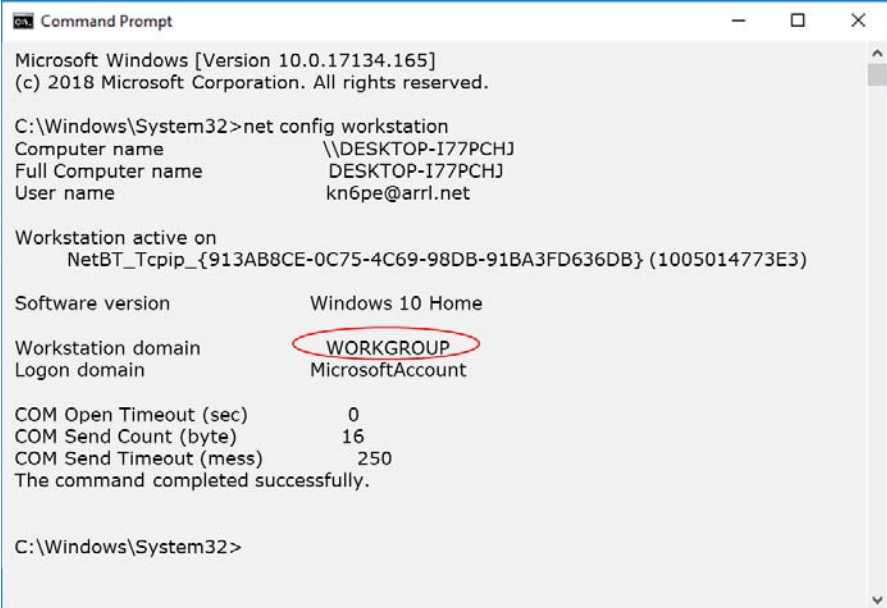
NOTE: Depending on your Linux login, all system commands may need to be preceded with "sudo" to allow root access.

3.1 Before you begin

1. You will need at least one windows machine to check the samba server that must be reachable with the ubuntu server.
2. The Windows machine must be on same workgroup. To check the value in windows machine run the command at cmd prompt

```
net config workstation
```

It will be like this:



```
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\System32>net config workstation
Computer name           \\DESKTOP-I77PCHJ
Full Computer name     DESKTOP-I77PCHJ
User name              kn6pe@arri.net

Workstation active on
  NetBT_Tcpip_{913AB8CE-0C75-4C69-98DB-91BA3FD636DB} (1005014773E3)

Software version       Windows 10 Home

Workstation domain     WORKGROUP
Logon domain          MicrosoftAccount

COM Open Timeout (sec)      0
COM Send Count (byte)     16
COM Send Timeout (mess)   250
The command completed successfully.

C:\Windows\System32>
```

Your windows machine must be at same Workstation domain as in ubuntu server, i.e. **WORKGROUP** in my case.

3.2 Anonymous samba sharing

Anonymous sharing means that anyone who is on the same network as the server can get to the file share. To set up samba with an anonymous share, do the following:

1. If samba is already installed, skip to step 2. Install samba as follows:

```
apt-get install samba samba-common python-glade2 system-  
config-samba
```

It will install samba with *version 4.1.6-Ubuntu*.

2. Create a backup of original file named `/etc/samba/smb.conf.bak`

```
mv /etc/samba/smb.conf /etc/samba/smb.conf.bak
```

3. Configure samba by editing the file `/etc/samba/smb.conf`. Use your favorite editor to make the changes.

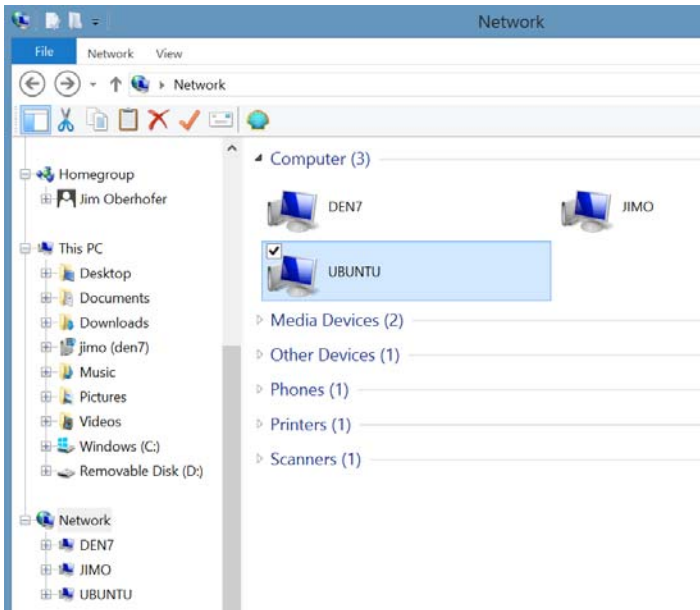
```
nano /etc/samba/smb.conf
```

```
[global]  
workgroup = WORKGROUP  
server string = Samba Server %v  
netbios name = ubuntu  
security = user  
map to guest = bad user  
dns proxy = no  
#===== Share Definitions =====  
[Anonymous]  
path = /srv/samba/anonymous  
browsable =yes  
writable = yes  
guest ok = yes  
read only = no
```

4. Create the share directory, and restart samba.

```
mkdir -p /srv/samba/anonymous  
service smbd restart
```

5. From your PC, access the ubuntu anonymous share. Click on Network in Windows Explorer. You should see the Ubuntu server. Your picture may vary depending on the version of Windows that you are running.



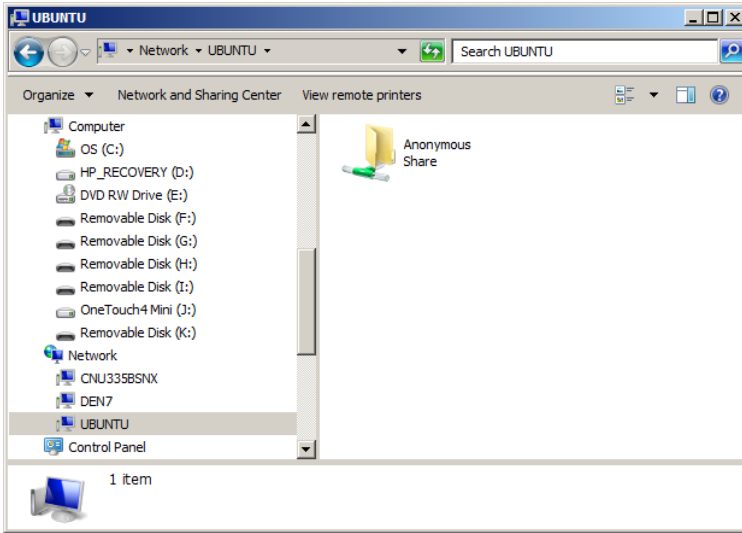
6. Check the permission for the shared folder.

```
root@server1:~# ls -l /srv/samba/
total 4
drwxr-xr-x 2 root root 4096 May 27 18:17 anonymous
root@server1:~#
```

7. To allow anonymous access, set the permissions as follows:

```
root@server1:~# cd /srv/samba
root@server1:/srv/samba# chmod -R 0755 anonymous/
root@server1:~# chown -R nobody:nogroup anonymous/
root@server1:/srv/samba# ls -l
total 4
drwxr-xr-x 2 nobody nogroup root 4096 May 27 18:17 anonymous
root@server1:/srv/samba#
```

8. Now any user can browse and create the folder contents. Create a test file on the anonymous share.



9. Back on Ubuntu, check the content at the server.

```
root@server1:/srv/samba# ls -l anonymous/
total 0
-rwxr--r-- 1 nobody nogroup 0 May 27 18:30 test_samba.txt
root@server1:/srv/samba#
```

3.3 Secured samba server

This procedure is more involved, but also offers a higher level of protection.

1. Create a group *smbgroup* and user *smbuser* to access the samba share.

Note: the user and group name are examples only. The password will not be echoed to your terminal. The string in <brackets> is for example only.

```
addgroup smbgroup
useradd smbuser -G smbgroup
smbpasswd -a smbuser
root@server1:~# smbpasswd -a smbuser
New SMB password: <smbpwd>
Retype new SMB password: <smbpwd>
Added user smbuser.
root@server1:~#
```

2. Create the folder *secured* in the */samba* folder (this could be any name) and set up the permissions as follows:


```
mkdir -p /srv/samba/secured
cd /srv/samba
chmod -R 0770 secured/
```

3. Edit the configuration file `/etc/samba/smb.conf` and add the following to the file:

```
[global]
workgroup = WORKGROUP
server string = Samba Server %v
netbios name = ubuntu
security = user
map to guest = bad user
dns proxy = no

#===== Share Definitions =====
[Anonymous]
path = /srv/samba/anonymous
browsable = yes
writable = yes
guest ok = yes
read only = no

[secured]
path = /srv/samba/secured
valid users = @smbgroup
guest ok = no
writable = yes
browsable = yes
```

4. Restart samba for the changes to take effect.

```
service smb restart
```

5. Cross-check the settings as follows:

```
root@server1:~# testparm
Load smb config files from /etc/samba/smb.conf
rlimit_max: increasing rlimit_max (1024) to minimum Windows
limit (16384)
Processing section "[Anonymous]"
Processing section "[secured]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions
<CR>
```

<you will see the listing here>

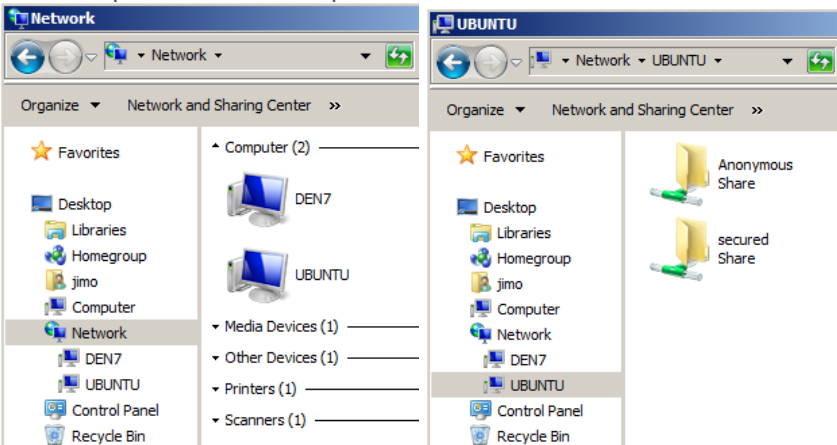
```
root@server1:~#
```

- Assign the ownership of the secured share to the user:group you created in Step 1:

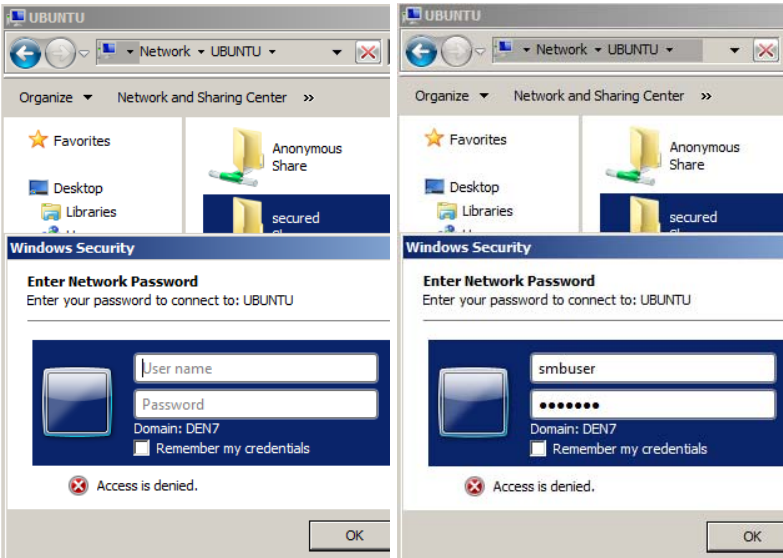
```
cd /srv/samba  
chown -R smbuser:smbgroup secured/
```

- The samba user *smbuser* now has permission to write in the folder.

- Open Windows File Explorer and double-click on the Network Icon.



- Double-clicking on the Anonymous share icon will open the share as before.
- Double-click on the secured share. You will be prompted for the user name and password. For this example,
User Name: smbuser
Password: smbpwd



11. You are now connected to the secured share.

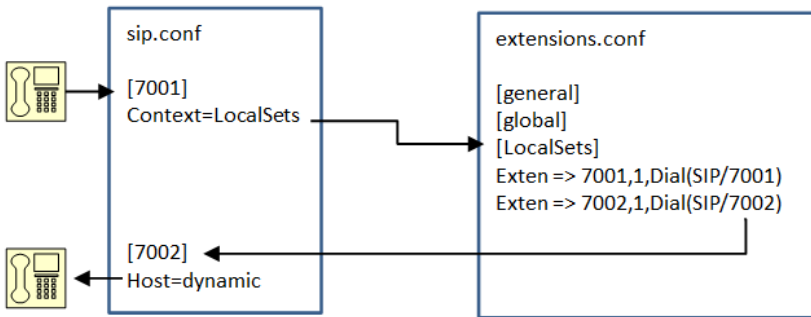
4 Asterisk & Phone Setup

Reference: <http://draalin.com/basic-asterisk-configuration-in-ubuntu/>, author: Tyler Bailey.

This section described a local extension-only phone system. External calling is not supported.

4.1 Overview

- SIP – Session initiation protocol, an application layer (signaling) control protocol for creating, modifying, and terminating sessions between 1 or more participants.
- sip.conf – used to configure devices to Asterisk.
- extensions.conf – a name for a set of actions for an extension number that is dialed.



Extension 7001 dials extension 7002

The following are the steps you need to execute to bring asterisk on line.

1. Install Asterisk. Check out this link here. This is an excellent step by step review of what it will take to get Asterisk installed on a Ubuntu server... <http://draalin.com/installing-asterisk-in-ubuntu/>
2. Edit the asterisk configuration. This will require edit and write access to asterisk configuration files in ubuntu.
3. Configure the VoIP phones.

Once at Step 2, it is good to have a plan on what you want to do. The following files will create a simple asterisk PBX configuration for any number of phones.

The configuration will support:

1. Local extension dialing. No external phone system access is defined.
2. Local conference calling
3. Phone extension read back
4. Echo test

4.2 Create the sip.conf file

The sip.conf file sets up asterisk channels for both inbound and outbound calls, essentially taking care of the setup and teardown of calls.

```
jimob@netbook:sudo gedit /etc/asterisk/sip.conf
```

```
; name:          sip.conf
; reference:     "Asterisk, the definitive guide", page 101

[general]
content=unauthenticated ; default context for incoming calls
allowguest=no           ; disabled unauthenticated calls
srvlookup=no           ; disable DNS server Lookups on outbound
                        ; calls
port=5060               ; UDP port to bind to (SIP std port = 5060)
udpbindaddr=0.0.0.0    ; IP address to bind (0.0.0.0 binds to all)
tcpenable=no           ; disable TCP support

[office-phone](!)      ; create a template for the devices
type=friend             ; channel dvr will match on 1st name, 2nd IP
context=LocalSets      ; this where calls from this device enter
                        ; the dial plan
host=dynamic            ; the device will register with asterisk
nat=force_rport,comedia ; assume device is behind NAT
dtmfmode=auto          ; accept touch tones from this device;
                        ; asterisk prefers rfc2833
disallow=all           ; reset which voice codec this device will
                        ; accept or offer
allow=g722             ; audio codec to accept from, and request
                        ; to, the device
allow=ulaw             ; in the order we prefer
allow=alaw             ;
call-limit=1          ; removes call waiting (annoying beep when
                        ; you are on another call); this setting
                        ; also disables 3-way calling

; define a device name and uses the office-phone template
[7001](office-phone)  ; device (user) name
secret=z7001          ; password

[7002](office-phone)  ; device (user) name
secret=z7002          ; password

[7003](office-phone)  ; device (user) name
secret=z7003          ; password

[7101](office-phone)  ; device (user) name
secret=z7101          ; password

; replicate the above 2 lines as many as necessary with unique
; phone numbers and a password.
```

4.3 Create the extensions.conf file

The extensions.conf file defines the Dial Plan for the PBX. A dial plan defines how Asterisk PBX will handle incoming and outgoing calls; it also contains all extension numbers.

```
jimob@netbook:sudo gedit /etc/asterisk/extensions.conf
```

```
[general]
; static=yes          ; default values for changes to this file
; writeprotect=no    ; by the Asterisk CLI

[global]
; variables go here

[default]
; variables go here

[LocalSets]
exten => 0,1,Dial(SIP/7001,20)
exten => 7001,1,Dial(SIP/7001,20)
exten => 7002,1,Dial(SIP/7002,20)
exten => 7003,1,Dial(SIP/7003,20)
exten => 7101,1,Dial(SIP/7101,20)

; Add an entry above for each extension entered in sip.conf.

; very simple test
exten => 100,1,Answer()
        same => n,Playback(hello-world)
        same => n,Hangup()

; play back with the extension
exten => 200,1,Answer()
same => n,Wait(1)
same => n,Playback(vm-extension)
same => n,SayDigits(${CALLERID(num)})
same => n,Playback(vm-goodbye)
same => n,Hangup()

; the Echo test
exten => 300,1,Answer()
exten => 300,2,Playback(demo-echotest)
exten => 300,3,Echo
exten => 300,4,Playback(demo-echodone)
exten => 300,5,Hangup

; Conference call
exten => 5001,1,Answer()
exten => 5001,2,ConfBridge(5001)
```

4.4 Setting up a Zultys Zip2 phone

The following is a sample worksheet that typically goes into setting up the Phone. To set up the Phone, you minimally need to know the following:

1. IP address of the asterisk PBX server
2. Phone ID. (logon user name) This uniquely identifies each phone. May be some variation of the phone number.
3. Phone Password. Allows the phone to log in to the Asterisk Server.

Tab	Setting	Telephone 1	Telephone 2
	MAC Address		
	IP Address	DHCP*	DHCP*
	Logon		
	Password		
	Access	Browser	Browser
SIP, SIP Server Settings	Server Addr	192.168.0.10	192.168.0.10
	Port	5060	5060
	Domain Name	192.168.0.10	192.168.0.10
	Send Ref Req	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
SIP, SIP gateway settings	Dial Plan	x.T x.# *x.T *x.#	x.T x.# *x.T *x.#
	Transport	UDP	UDP
	Phone No	7001	7002
	Caller ID	7001	7002
	Port	5060	5060
	AEC ON	ON	ON
	User Name	7001	7002
	Pwd	z7001	z7002
CODECS	G711U	20ms, OFF	20ms, OFF
	G711A	20ms, OFF	20ms, OFF
	<input checked="" type="checkbox"/> G729	20ms, OFF	20ms, OFF
CODECS, Jitter Buffer	<input checked="" type="radio"/> Adaptive Jitter Buffer	100ms	100ms
	<input type="radio"/> Fixed Jitter Buffer	40ms	40ms

Tab	Setting	Telephone 1	Telephone 2
	<input type="checkbox"/> Auto switch...	Unchecked	Unchecked
CODECS, Paging support	<input type="checkbox"/> Enable	Unchecked	Unchecked
	Paging Server		
	Paging Port	3771	3771
	Paging Codec	G700U	G700U

The following is the configuration for Phone #1 as described above.



ZULTYS VoIP Phone

Home | LAN | SIP | CODECS | System | Download | Reset

SIP | SIP Extensions | OOB Signalling | ToS/DiffServ | VLAN

SIP Configuration

SIP Server Settings (Current Server: 192.168.0.10 : 5060 ; Domain: 192.168.0.10)

* Server Address: (IP or FQDN)

* Port:

* Domain Name:

Send Registration Request

Backup Server Address: (IP or FQDN)

Backup Server Port:

Send Registration to Backup Server

Gateway Settings

Dial Plan:

Transport:

Phone Number	CallerID Name	Port	AEC On	User Name	Password
Line1: <input type="text" value="7001"/>	<input type="text" value="7001"/>	<input type="text" value="5060"/>	<input type="text" value="ON"/>	<input type="text" value="7001"/>	<input type="text" value="*****"/>

* Leaving a setting blank will force the unit to use the information obtained via DHCP and/or DNS

See the BBHN Client Handbook for a description of the setup process.

5 Apache virtual host setup

Reference: <https://www.digitalocean.com/community/tutorials/how-to-set-up-apache-virtual-hosts-on-ubuntu-14-04-lts>, by Justin Ellingwood.

5.1 Overview

The Apache web server is the most popular way of serving web content on the internet. It accounts for more than half of all active websites on the internet and is extremely powerful and flexible.

Apache breaks its functionality and components into individual units that can be customized and configured independently. The basic unit that describes an individual site or domain is called a virtual host.

These designations allow the administrator to use one server to host multiple domains or sites off of a single interface or IP by using a matching mechanism. This is relevant to anyone looking to host more than one site off of a single VPS. Each domain that is configured will direct the visitor to a specific directory holding that site's information, never indicating that the same server is also responsible for other sites. This scheme is expandable without any software limit as long as your server can handle the load.

In this guide, we will walk you through how to set up Apache virtual hosts on an Ubuntu 14.04 VPS. During this process, you'll learn how to serve different content to different visitors depending on which domains they are requesting.

5.2 Installation

You will need to have Apache installed in order to work through these steps. If you haven't already done so, you can get Apache installed on your server through apt-get:

```
sudo apt-get update
sudo apt-get install apache2
```

5.3 Create the Directory Structure

Make a directory structure that will hold the site data that we will be serving to visitors.

Our `document root` (the top-level directory that Apache looks at to find content to serve) will be set to individual directories under the `/var/www` directory. We will create a directory here for both of the virtual hosts we plan on making.

Within each of *these* directories, we will create a `public_html` directory that will hold our actual files. This gives us some flexibility in our hosting.

For instance, for our sites, we're going to make our directories like this:

```
sudo mkdir -p /var/www/example.com/public_html
sudo mkdir -p /var/www/test.com/public_html
```

The portions in red represent the domain names that we are wanting to serve from our VPS.

5.4 Grant Permissions

Now we have the directory structure for our files, but they are owned by our root user. If we want our regular user to be able to modify files in our web directories, we can change the ownership by doing this:

```
sudo chown -R $USER:$USER /var/www/example.com/public_html
sudo chown -R $USER:$USER /var/www/test.com/public_html
```

The `$USER` variable will take the value of the user you are currently logged in as when you press "ENTER". By doing this, our regular user now owns the `public_html` subdirectories where we will be storing our content.

We should also modify our permissions a little bit to ensure that read access is permitted to the general web directory and all of the files and folders it contains so that pages can be served correctly:

```
sudo chmod -R 755 /var/www
```

Your web server should now have the permissions it needs to serve content, and your user should be able to create content within the necessary folders.

5.5 Create Demo Pages for Each Virtual Host

We have our directory structure in place. Let's create some content to serve.

We're just going for a demonstration, so our pages will be very simple. We're just going to make an `index.html` page for each site.

Let's start with `example.com`. We can open up an `index.html` file in our editor by typing:

```
gedit /var/www/example.com/public_html/index.html
```

In this file, create a simple HTML document that indicates the site it is connected to. My file looks like this:

```
<html>
  <head>
    <title>Welcome to Example.com!</title>
  </head>
  <body>
    <h1>Success! The example.com virtual host is
working!</h1>
  </body>
</html>
```

Save and close the file when you are finished.

Copy this file to use as the basis for our second site:

```
cp /var/www/example.com/public_html/index.html
   /var/www/test.com/public_html/index.html
```

Open the file and modify the relevant pieces of information:

```
<html>
  <head>
    <title>Welcome to Test.com!</title>
  </head>
  <body>
    <h1>Success! The test.com virtual host is working!
</h1>
  </body>
</html>
```

Save and close the file. You now have the pages necessary to test the virtual host configuration.

5.6 Create New Virtual Host Files

Virtual host files are the files that specify the actual configuration of our virtual hosts and dictate how the Apache web server will respond to various domain requests.

Apache comes with a default virtual host file called `000-default.conf` that we can use as a jumping off point. We are going to copy it over to create a virtual host file for each of our domains.

We will start with one domain, configure it, copy it for our second domain, and then make the few further adjustments needed. The default Ubuntu configuration requires that each virtual host file end in `.conf`.

5.7 Create the First Virtual Host File

Start by copying the file for the first domain:

```
sudo cp /etc/apache2/sites-available/000-default.conf
/etc/apache2/sites-available/example.com.conf
```

Open the new file in your editor with root privileges:

```
sudo gedit /etc/apache2/sites-available/example.com.conf
```

The file will look something like this (I've removed the comments here to make the file more approachable):

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

As you can see, there's not much here. We will customize the items here for our first domain and add some additional directives. This virtual host section matches any requests that are made on port 80, the default HTTP port.

First, we need to change the `ServerAdmin` directive to an email that the site administrator can receive emails through.

```
ServerAdmin admin@example.com
```

After this, we need to *add* two directives. The first, called `ServerName`, establishes the base domain that should match for this virtual host definition. This will most likely be your domain. The second, called `ServerAlias`, defines further names that should match as if they were the base name. This is useful for matching hosts you defined, like `www`:

```
ServerName example.com
ServerAlias www.example.com
```

The only other thing we need to change for a basic virtual host file is the location of the document root for this domain. We already created the directory we need, so we just need to alter the `DocumentRoot` directive to reflect the directory we created:

```
DocumentRoot /var/www/example.com/public_html
```

In total, our `virtualhost` file should look like this:

```
<VirtualHost *:80>
    ServerAdmin admin@example.com
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot /var/www/example.com/public_html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Save and close the file.

Copy First Virtual Host and Customize for Second Domain

Create the second one by copying the above file and changing it as needed.

```
cd /etc/apache2/sites-available
sudo cp example.com.conf test.com.conf
```

Open the new file with root privileges in your editor:

```
sudo gedit /etc/apache2/sites-available/test.com.conf
```

You now need to modify all of the pieces of information to reference your second domain. When you are finished, it may look something like this:

```
<VirtualHost *:80>
    ServerAdmin admin@test.com
    ServerName test.com
    ServerAlias www.test.com
    DocumentRoot /var/www/test.com/public_html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Save and close the file when finished.

5.8 Enable the New Virtual Host Files

Apache includes some tools that allow us to enable the virtual host files. Use the `a2ensite` tool to enable each of our sites as follows:

```
sudo a2ensite example.com.conf
sudo a2ensite test.com.conf
```

When you are finished, you restart Apache to make these changes take effect:

```
sudo service apache2 restart
```

You will most likely receive a message saying something similar to:

```
* Restarting web server apache2
AH00558: apache2: Could not reliably determine the
server's fully qualified domain name, using 127.0.0.1. Set
the 'ServerName' directive globally to suppress this
message
```

This is a harmless message that does not affect our site.

5.9 Set Up Local Hosts File (Optional)

If you haven't been using actual domain names that you own to test this procedure and have been using some example domains instead, you can at least test the functionality of this process by temporarily modifying the `hosts` file on your local computer.

This will intercept any requests for the domains that you configured and point them to your VPS server, just as the DNS system would do if you were using registered domains. This will only work from your computer though, and is simply useful for testing purposes.

Make sure you are operating on your local computer for these steps and not your VPS server. You will need to know the computer's administrative password or otherwise be a member of the administrative group.

If you are on a Mac or Linux computer, edit your local file with administrative privileges by typing:

```
sudo gedit /etc/hosts
```

If you are on a Windows machine, run notepad from the command prompt with **administrative privileges**:

```
notepad c:\Windows\System32\drivers\etc\hosts
```

The details that you need to add are the public IP address of your VPS server followed by the domain you want to use to reach that VPS.

For the domains that I used in this guide, assuming that my VPS IP address is 192.168.1.71, I could add the following lines to the bottom of my `hosts` file:

```
127.0.0.1 localhost
127.0.1.1 guest-desktop
192.178.1.71 example.com
192.178.1.71 test.com
```

This will direct any requests for `example.com` and `test.com` on our computer and send them to our server at `192.168.1.71`. This is what we want if we are not actually the owners of these domains in order to test our virtual hosts. Save and close the file.

5.10 Test your Results

Now that you have your virtual hosts configured, you can test your setup easily by going to the domains that you configured in your web browser:

```
http://example.com
```

You should see a page that looks like this:

```
Success! The example.com virtual host is working!
```

Likewise, if you can visit your second page:

```
http://test.com
```

You should see a page that looks like this:

```
Success! The test.com virtual host is working!
```

If both of these sites work well, you've successfully configured two virtual hosts on the same server.

If you adjusted your home computer's hosts file, you may want to delete the lines you added now that you verified that your configuration works. This will prevent your hosts file from being filled with entries that are not actually necessary.

If you need to access this long term, consider purchasing a domain name for each site you need and setting it up to point to your VPS server.

5.11 Conclusion

At this point, you should have a single server handling two separate domain names. There is no software limit on the number of domain names Apache can handle, so feel free to make as many as your server is capable of handling.

6 FTP Server

Reference: <https://help.ubuntu.com/lts/serverguide/ftp-server.html>, Author Ubuntu Community.

6.1 Overview

File Transfer Protocol (FTP) is a TCP protocol for downloading files between computers. FTP works on a client/server model. The server component is called an FTP daemon. It continuously listens for FTP requests from remote clients. When a request is received, it manages the login and sets up the connection. For the duration of the session it executes any of commands sent by the FTP client.

Access to an FTP server can be managed in two ways:

- Anonymous
- Authenticated

In the Anonymous mode, remote clients can access the FTP server by using the default user account called "anonymous" or "ftp" and sending an email address as the password. In the Authenticated mode a user must have an account and a password. This latter choice is very insecure and should not be used except in special circumstances.

6.2 Installation

`vsftpd` is an FTP daemon available in Ubuntu. To install `vsftpd` you can run the following command:

```
sudo apt-get install vsftpd
```

6.3 Anonymous FTP Configuration

By default `vsftpd` is not configured to allow anonymous download. If you wish to enable anonymous download edit `/etc/vsftpd.conf` by changing:

```
anonymous_enable=Yes
```

During installation, an `ftp` user is created with a home directory of `/srv/ftp`. This is the default FTP directory.

If you wish to change this location, to `/srv/files/ftp` for example, create a directory in another location and change the `ftp` user's home directory:

```
sudo mkdir /srv/files/ftp
sudo usermod -d /srv/files/ftp ftp
```

After making the change restart `vsftpd`:


```
sudo restart vsftpd
```

Finally, copy any files and directories you would like to make available through anonymous FTP to `/srv/files/ftp`, or `/srv/ftp` if you wish to use the default.

6.4 User Authenticated FTP Configuration

By default `vsftpd` is configured to authenticate system users and allow them to download files. If you want users to be able to upload files, edit `/etc/vsftpd.conf`:

```
write_enable=YES
```

Now restart `vsftpd`:

```
sudo restart vsftpd
```

Now when system users login to FTP they will start in their home directories where they can download, upload, create directories, etc.

Similarly, by default, anonymous users are not allowed to upload files to FTP server. To change this setting, you should uncomment the following line, and restart `vsftpd`:

```
anon_upload_enable=YES
```

Enabling anonymous FTP upload can be an extreme security risk. It is best to not enable anonymous upload on servers accessed directly from the Internet.

The configuration file consists of many configuration parameters. The information about each parameter is available in the configuration file.

Alternatively, you can refer to the man page, `man 5 vsftpd.conf` for details of each parameter.

6.5 Securing FTP

There are options in `/etc/vsftpd.conf` to help make `vsftpd` more secure. For example users can be limited to their home directories by uncommenting:

```
chroot_local_user=YES
```

You can also limit a specific list of users to just their home directories:

```
chroot_list_enable=YES
chroot_list_file=/etc/vsftpd.chroot_list
```

After uncommenting the above options, create a `/etc/vsftpd.chroot_list` containing a list of users one per line. Then restart `vsftpd`:

```
sudo restart vsftpd
```

Also, the `/etc/ftpusers` file is a list of users that are disallowed FTP access. The default list includes `root`, `daemon`, `nobody`, etc. To disable FTP access for additional users simply add them to the list.

FTP can also be encrypted using FTPS. Different from SFTP, FTPS is FTP over Secure Socket Layer (SSL). SFTP is a FTP like session over an encrypted SSH connection. A major difference is that users of SFTP need to have a shell account on the system, instead of a nologin shell. Providing all users with a shell may not be ideal for some environments, such as a shared web host. However, it is possible to restrict such accounts to only SFTP and disable shell interaction. See the section on OpenSSH-Server for more.

To configure FTPS, edit `/etc/vsftpd.conf` and at the bottom add:

```
ssl_enable=Yes
```

Also, notice the certificate and key related options:

```
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
```

By default these options are set to the certificate and key provided by the `ssl-cert` package. In a production environment these should be replaced with a certificate and key generated for the specific host. For more information on certificates see [Certificates](#).

Now restart `vsftpd`, and non-anonymous users will be forced to use FTPS:

```
sudo restart vsftpd
```

To allow users with a shell of `/usr/sbin/nologin` access to FTP, but have no shell access, edit `/etc/shells` adding the `nologin` shell:

```
# /etc/shells: valid login shells
/bin/csh
/bin/sh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
/bin/tcsh
/usr/bin/esh
/bin/dash
/bin/bash
/bin/rbash
/usr/bin/screen
/usr/sbin/nologin
```

This is necessary because, by default `vsftpd` uses PAM for authentication, and the `/etc/pam.d/vsftpd` configuration file contains:

```
auth    required          pam_shells.so
```

The shells PAM module restricts access to shells listed in the `/etc/shells` file. Most popular FTP clients can be configured to connect using FTPS. The `lftp` command line FTP client has the ability to use FTPS as well.

7 TFTP Server

Reference: <http://askubuntu.com/questions/201505/how-do-i-install-and-run-a-tftp-server>, Author Ubuntu Community.

7.1 Overview

Trivial File Transfer Protocol (TFTP) is an Internet software utility for transferring files that is simpler to use than the File Transfer Protocol (FTP) but less capable. It is used where user authentication and directory visibility are not required.

7.2 Installation and Setup

1. Install following packages.

```
sudo apt-get install xinetd tftpd tftp
```

2. Confirm or create the file `/etc/xinetd.d/tftp` and put this entry

```
service tftp
{
  protocol          = udp
  port              = 69
  socket_type       = dgram
  wait              = yes
  user              = nobody
  server            = /usr/sbin/in.tftpd
  server_args       = /tftpboot
  disable           = no
}
```

3. Create a folder `/tftpboot` this should match whatever you gave in `server_args`

```
sudo mkdir /tftpboot
sudo chmod -R 777 /tftpboot
sudo chown -R nobody /tftpboot
```

4. Restart the `xinetd` service.

```
sudo service xinetd restart    (new systems)
or
sudo /etc/init.d/xinetd restart (older systems)
```

The tftp server is up and running.

7.3 Testing the tftp server

5. Create a file named `test` with some content and save it in the `/tftpboot` path of the tftp server.
6. Obtain the `ip` address of the tftp server using `ifconfig` command.
7. From another linux system, do the following steps.

```
tftp 192.168.1.2
tftp> get test
Sent 159 bytes in 0.0 seconds
tftp> quit
cat test
```

8. Windows had a TFTP Client that may not be enabled. To enable it,
 - a. Start > Control Panel > Programs
 - b. Click on **Turn Windows features on or off**.
 - c. In the Windows Features window, scroll down and find **TFTP Client**. Check the Box and press **OK**.
 - d. Windows run a process to enable this program (and any others you checked).
9. Run the Windows Command program
 - a. Windows 7/8/10: Right-click on the Start icon, and click on Command Prompt.
10. Enter this command:

```
C:\Users\yourlogon> tftp <ip address> GET test
Transfer successful: 46 bytes in 1 second(s), 46 bytes/s
C:\Users\yourlogon>
```

Open the file to confirm the file contents.

Cupertino ARES/RACES
10300 Torre Avenue
Cupertino, CA 95014-3255