

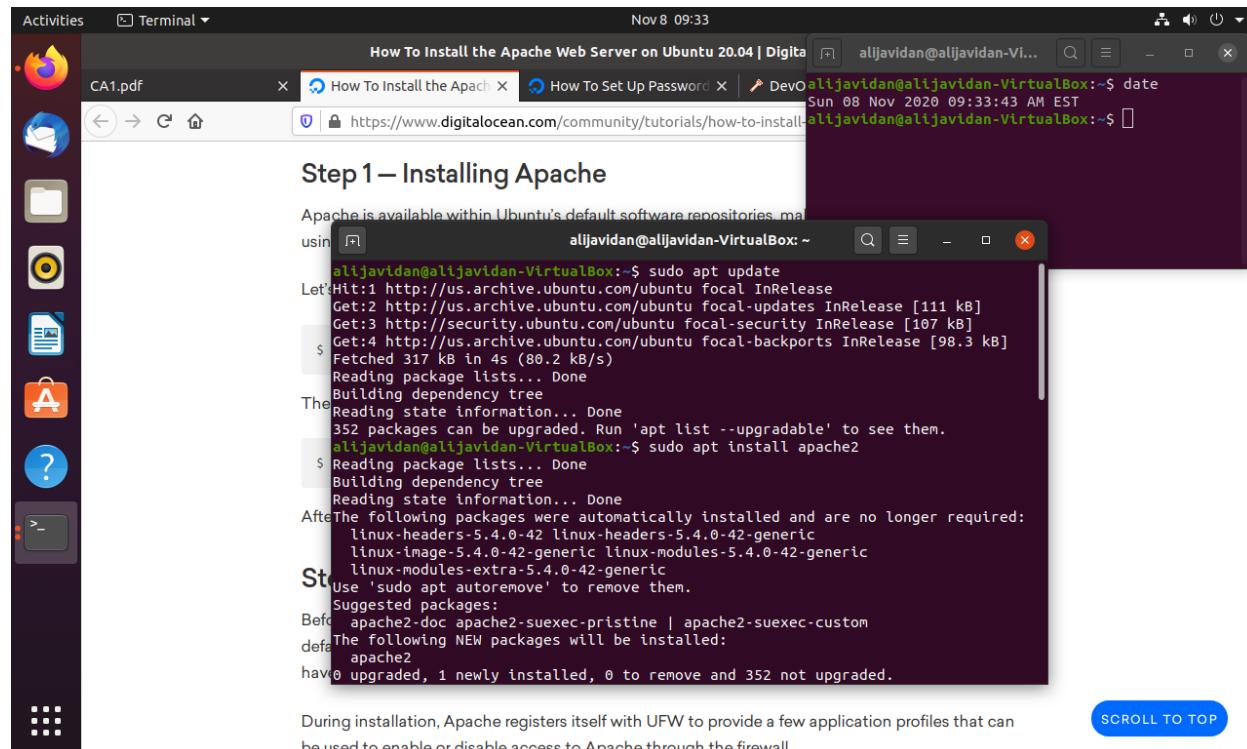
1. Set up a HTTP server with user authentication

Step 1 — Installing Apache

Let's begin by updating the local package index to reflect the latest upstream changes.

Then, install the apache2 package.

After confirming the installation, apt will install Apache and all required dependencies.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "alijavidan@alijavidan-VirtualBox:~". The terminal content shows the following command being run:

```
alijavidan@alijavidan-VirtualBox:~$ sudo apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease [111 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security InRelease [107 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease [98.3 kB]
Fetched 317 kB in 4s (80.2 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
352 packages can be upgraded. Run 'apt list --upgradable' to see them.
alijavidan@alijavidan-VirtualBox:~$ sudo apt install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-5.4.0-42 linux-headers-5.4.0-42-generic
  linux-image-5.4.0-42-generic linux-modules-5.4.0-42-generic
  linux-modules-extra-5.4.0-42-generic
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
Before installing, the following NEW packages will be installed:
  apache2
0 upgraded, 1 newly installed, 0 to remove and 352 not upgraded.
```

Below the terminal window, a status message reads: "During installation, Apache registers itself with UFW to provide a few application profiles that can be used to enable or disable access to Apache through the firewall." A blue "SCROLL TO TOP" button is located at the bottom right of the terminal window.

Step 2 — Adjusting the Firewall

During installation, Apache registers itself with UFW to provide a few application profiles that can be used to enable or disable access to Apache through the firewall.

We can list the ufw application profiles by typing:

```
sudo ufw app list
```

It is recommended that we enable the most restrictive profile that will still allow the traffic we've configured. Since we haven't configured SSL for our server yet, we will only need to allow traffic on port 80.

The screenshot shows a typical Ubuntu desktop environment with a dock on the left containing icons for the Dash, Home, Applications, and Help. A terminal window is open in the center, titled 'How To Install the Apache Web Server on Ubuntu 20.04 | DigitalOcean'. The terminal output shows the user running commands to configure Apache and UFW. The user first lists available applications with `sudo ufw app list`, then allows Apache with `sudo ufw allow 'Apache'`. They disable the firewall with `sudo ufw disable` and reset it with `sudo ufw reset`. Finally, they enable Apache again with `sudo ufw allow 'Apache'`. The terminal shows the status of the rules, with 'Apache' listed under the ALLOW column.

```
alijavidan@alijavidan-VirtualBox:~$ sudo ufw app list
Available applications:
Apache
Apache Full
Apache Secure
CUPS
S alijavidan@alijavidan-VirtualBox:~$ sudo ufw allow 'Apache'
Skipping adding existing rule (v6)
You alijavidan@alijavidan-VirtualBox:~$ sudo ufw disable
Firewall stopped and disabled on system startup
alijavidan@alijavidan-VirtualBox:~$ sudo ufw reset
S Resetting all rules to installed defaults. Proceed with operation (y/n)? y
Backing up 'user.rules' to '/etc/ufw/user.rules.20201108_093429'
Backing up 'before.rules' to '/etc/ufw/before.rules.20201108_093429'
Backing up 'after.rules' to '/etc/ufw/after.rules.20201108_093429'
Backing up 'user6.rules' to '/etc/ufw/user6.rules.20201108_093429'
Backing up 'before6.rules' to '/etc/ufw/before6.rules.20201108_093429'
Backing up 'after6.rules' to '/etc/ufw/after6.rules.20201108_093429'
alijavidan@alijavidan-VirtualBox:~$ sudo ufw allow 'Apache'
Rules updated
Rules updated (v6)
alijavidan@alijavidan-VirtualBox:~$ 
-- 
OpenSSH      ALLOW      Anywhere
Apache       ALLOW      Anywhere
```

We can verify the change by typing:

```
sudo ufw status
```

The screenshot shows the same Ubuntu desktop environment. The terminal window now displays the output of the `sudo ufw status` command. It shows that the firewall is inactive, then enables it with `sudo ufw enable`. After enabling, the status changes to active. The terminal then lists the current rules, which include a new entry for Apache allowing traffic from anywhere on port 80.

```
alijavidan@alijavidan-VirtualBox:~$ 
It is recommended that you enable the most restrictive profile that will
allow traffic on port 80:
You alijavidan@alijavidan-VirtualBox:~$ 
alijavidan@alijavidan-VirtualBox:~$ 
S Resetting all rules to installed defaults. Proceed with operation (y/n)? y
Backing up 'user.rules' to '/etc/ufw/user.rules.20201108_093429'
Backing up 'before.rules' to '/etc/ufw/before.rules.20201108_093429'
Backing up 'after.rules' to '/etc/ufw/after.rules.20201108_093429'
Backing up 'user6.rules' to '/etc/ufw/user6.rules.20201108_093429'
Backing up 'before6.rules' to '/etc/ufw/before6.rules.20201108_093429'
Backing up 'after6.rules' to '/etc/ufw/after6.rules.20201108_093429'
alijavidan@alijavidan-VirtualBox:~$ sudo ufw allow 'Apache'
Rules updated
Rules updated (v6)
alijavidan@alijavidan-VirtualBox:~$ sudo ufw status
Status: inactive
alijavidan@alijavidan-VirtualBox:~$ sudo ufw enable
Firewall is active and enabled on system startup
alijavidan@alijavidan-VirtualBox:~$ sudo ufw status
Status: active
-- 
To          Action      From
Op
Apache      ALLOW      Anywhere
OpApache (v6) ALLOW      Anywhere (v6)
Al
alijavidan@alijavidan-VirtualBox:~$ 
```

As indicated by the output, the profile has been activated to allow access to the Apache web server.

But since ufw status is inactive, we enable it by typing:

```
sudo ufw enable
```

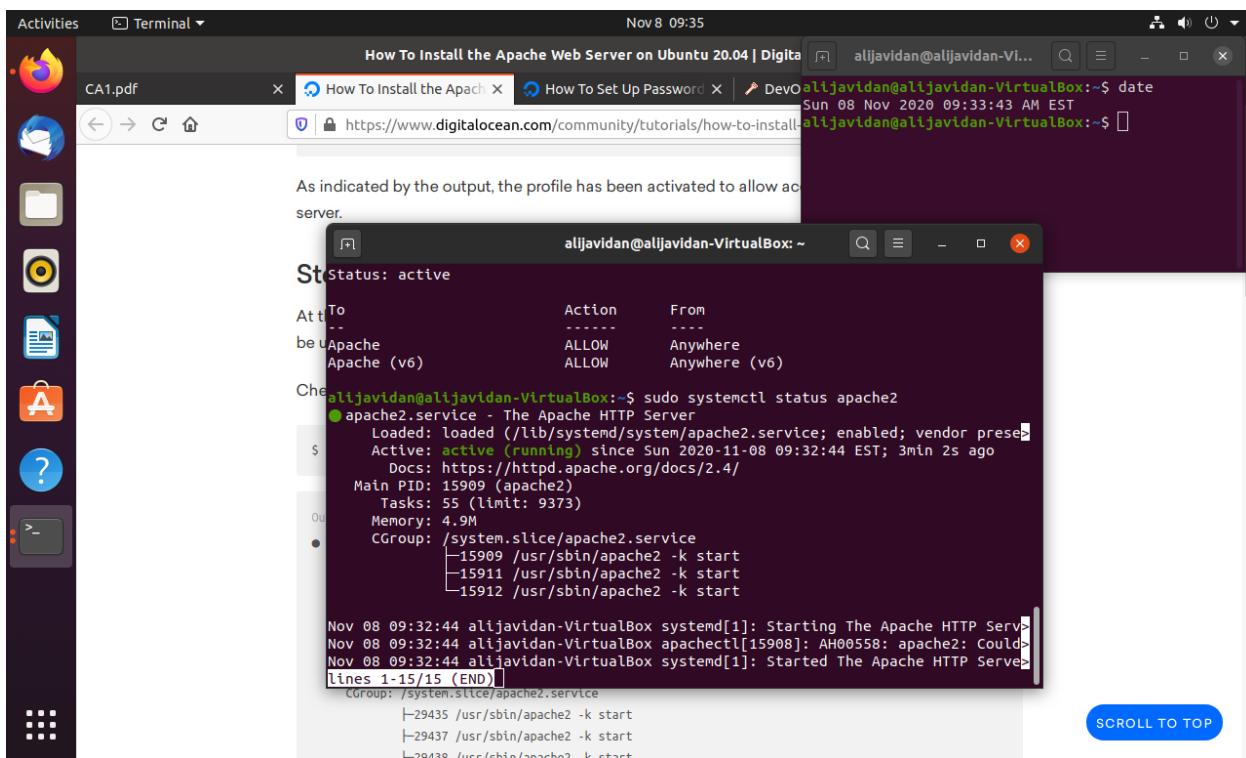
And check its status again by typing:

```
sudo ufw status
```

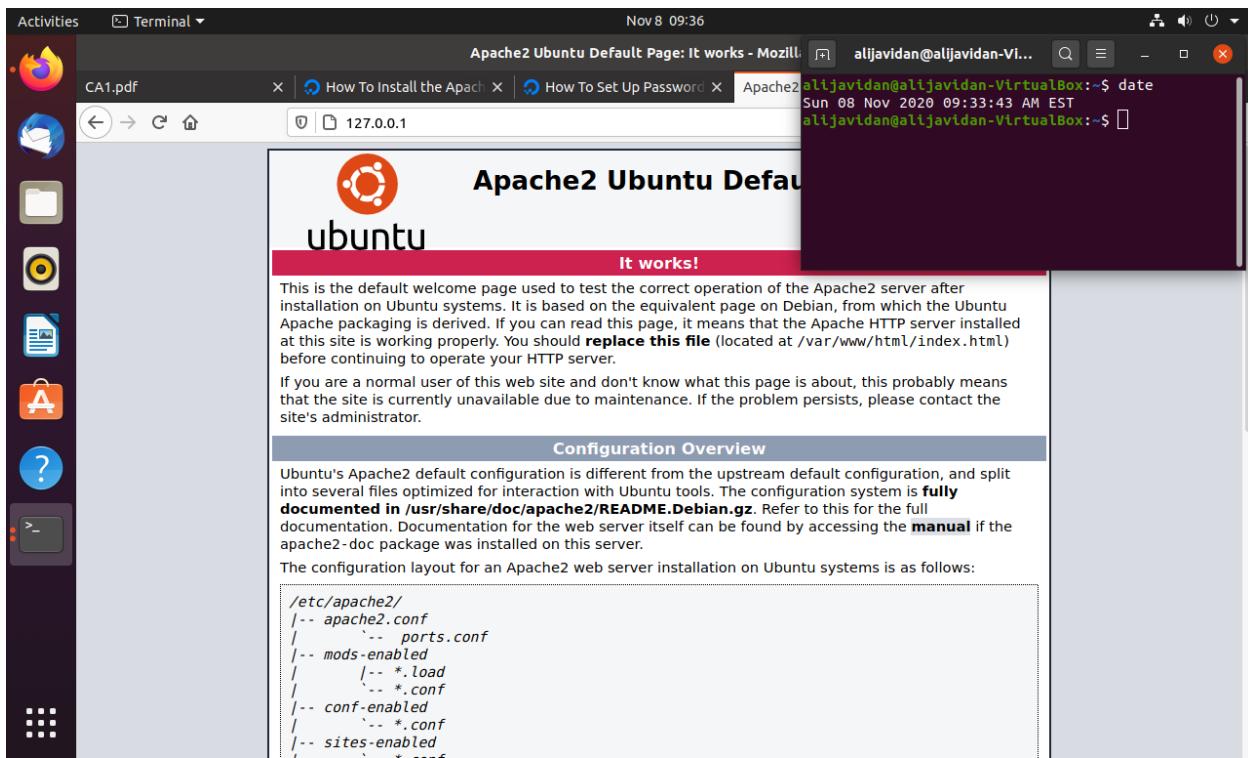
again.

Step 3 — Checking our Web Server

We check with the systemd init system to make sure the service is running.



Now we enter 127.0.0.1 into our browser's address bar, and we could see apache2 homepage gets shown up.



Set Up Password Authentication

Step 1 — Installing the Apache Utilities Package

Let's begin by updating our server and installing a package that we'll need. We will be using a utility called `htpasswd`, part of the `apache2-utils` package, to create the file and manage the username and passwords needed to access restricted content.

The screenshot shows a Linux desktop environment with a dark theme. On the left is a vertical dock containing icons for a browser, file manager, terminal, and other applications. The main window has a title bar with tabs for "How To Set Up Password Authentication with Apache on Ubuntu 18.04" and "Apache2". The content area contains a step-by-step guide with sections like "Prerequisites", "Step 1 – Installing the Apache Utilities Package", "Step 2 – Creating the Password File", "Step 3 – Configuring Apache Password Authentication", "Step 4 – Confirming Password Authentication", and "Conclusion". A search bar at the top of the content area has the command "sudo htpasswd -c /etc/apache2/.htpasswd javidan" entered. Below the search bar, a tooltip says "Apache to get set up." The terminal window below shows the execution of the command:

```
alijavidan@alijavidan-VirtualBox:~$ sudo apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [107 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease [111 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease [98.3 kB]
Fetched 317 kB in 3s (98.2 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
352 packages can be upgraded. Run 'apt list --upgradable' to see them.
alijavidan@alijavidan-VirtualBox:~$ sudo apt install apache2-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
apache2-utils is already the newest version (2.4.41-4ubuntu3.1).
The following packages were automatically installed and are no longer required:
  linux-headers-5.4.0-42 linux-headers-5.4.0-42-generic
  linux-image-5.4.0-42-generic linux-modules-5.4.0-42-generic
  linux-modules-extra-5.4.0-42-generic
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 352 not upgraded.
alijavidan@alijavidan-VirtualBox:~$
```

The first time we use this utility, we need to add the `-c` option to create the specified `passwdfile`. We specify javidan as username at the end of the command to create a new entry within the file.

With this installed, we now have access to the `htpasswd` command.

Step 2 — Creating the Password File

The `htpasswd` command will allow us to create a password file that Apache can use to authenticate users. We will create a hidden file for this purpose called `.htpasswd` within our `/etc/apache2` configuration directory.

The first time we use this utility, we need to add the `-c` option to create the specified `passwdfile`. We specify javidan as username at the end of the command to create a new entry within the file.

We will be asked to supply and confirm a password for the user.

If we view the contents of the file, we can see the username and the encrypted password for each record:

The screenshot shows a Linux desktop environment with a dark theme. On the left, there's a vertical dock containing icons for a file manager, terminal, browser, and other applications. The main window has a title bar with "Activities" and "Terminal". A terminal window is open, showing the command `sudo htpasswd -c /etc/apache2/.htpasswd javidan` and its output. The output includes instructions about adding users and upgrading packages. Below the terminal is a browser window titled "How To Set Up Password Authentication with Apache on Ubuntu 18.04", displaying a step-by-step guide with sections like "CONTENTS", "Prerequisites", and "Step 1 – Installing the Apache Utilities Package".

```
alijavidan@alijavidan-VirtualBox:~$ date
Sun Nov 08 2020 09:33:43 AM EST
alijavidan@alijavidan-VirtualBox:~$ 
alijavidan@alijavidan-VirtualBox:~$ sudo htpasswd -c /etc/apache2/.htpasswd javidan
Leave out the -c argument for any additional users you wish to add

$ sudo htpasswd /etc/apache2/.htpasswd another_user
alijavidan@alijavidan-VirtualBox:~$ 
If we Fetched 317 kB in 3s (98.2 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
352 packages can be upgraded. Run 'apt list --upgradable' to see them.
$ alijavidan@alijavidan-VirtualBox:~$ sudo apt install apache2-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
apache2-utils is already the newest version (2.4.41-4ubuntu3.1).
The following packages were automatically installed and are no longer required:
  linux-headers-5.4.0-42 linux-headers-5.4.0-42-generic
    linux-image-5.4.0-42-generic linux-modules-5.4.0-42-generic
      linux-modules-extra-5.4.0-42-generic
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 352 not upgraded.
We alijavidan@alijavidan-VirtualBox:~$ sudo htpasswd -c /etc/apache2/.htpasswd javi
dan
New password:
St Re-type new password:
Adding password for user javidan
In alijavidan@alijavidan-VirtualBox:~$ cat /etc/apache2/.htpasswd
javidan:$apr1$yHLHHLqa$FbFnFruzsoCmzbtlEMGx01
We alijavidan@alijavidan-VirtualBox:~$ 
files in the directories that need restriction. It's generally best to use the virtual host file, but if you need to allow non-root users to manage their own access restrictions, check the restrictions into version control alongside the website, or have a web application using .htaccess files for other
```

Step 3 — Configuring Apache Password Authentication

Configuring Access Control with .htaccess Files

Apache can use .htaccess files in order to allow certain configuration items to be set within a content directory.

To enable password protection using .htaccess files, we open the main Apache configuration file with a command-line text editor such as gedit.

The screenshot shows a Linux desktop environment with a terminal window and a help documentation window. The terminal window displays a command-line session for setting up password authentication on Apache 2.0.4. The help documentation window provides step-by-step instructions for the process.

```

alijavidan@alijavidan-VirtualBox:~$ date
Sun 08 Nov 2020 09:33:43 AM EST
alijavidan@alijavidan-VirtualBox:~$ 

alijavidan@alijavidan-VirtualBox:~$ sudo htpasswd -c /etc/apache2/.htpasswd javidan
New password:
Re-type new password:
Adding password for user javidan
alijavidan@alijavidan-VirtualBox:~$ cat /etc/apache2/.htpasswd
javidan:$apr1$yHLHqoSfbFnfRuzsoCmzbtlEMGX01
alijavidan@alijavidan-VirtualBox:~$ sudo gedit /etc/apache2/apache2.conf

```

Save and close the file when you are finished. If you are using nano, you can do so by pressing **CTRL+X** followed by **y** then **ENTER**.

SCROLL TO TOP

We find the <Directory> block for the /var/www directory that holds the document root. We turn on .htaccess processing by changing the AllowOverride directive within that block from None to All:

The screenshot shows a Linux desktop environment with a terminal window and a code editor window. The code editor window displays the Apache configuration file (apache2.conf) with the Apache2 directory block highlighted. The terminal window shows a command-line session for setting up password authentication on Apache 2.0.4.

```

alijavidan@alijavidan-VirtualBox:~$ date
Sun 08 Nov 2020 09:33:43 AM EST
alijavidan@alijavidan-VirtualBox:~$ 

alijavidan@alijavidan-VirtualBox:~$ sudo htpasswd -c /etc/apache2/.htpasswd javidan
New password:
Re-type new password:
Adding password for user javidan
alijavidan@alijavidan-VirtualBox:~$ cat /etc/apache2/.htpasswd
javidan:$apr1$yHLHqoSfbFnfRuzsoCmzbtlEMGX01
alijavidan@alijavidan-VirtualBox:~$ sudo gedit /etc/apache2/apache2.conf

```

SCROLL TO TOP

A screenshot of a Linux desktop environment. On the left is a dock with various icons. In the center is a terminal window titled "apache2.conf /etc/apache2". The terminal shows the Apache configuration file with several sections for directory overrides and access controls. The date command is run at the bottom of the terminal. To the right of the terminal is a file browser window showing a file named "htaccess" in the "/var/www/html" directory.

```
Nov 8 09:41 apache2.conf /etc/apache2
alijavidan@alijavidan-VirtualBox:~$ date
Sun 08 Nov 2020 09:33:43 AM EST
alijavidan@alijavidan-VirtualBox:~$ 
```

Next, we need to add an .htaccess file to the directory we wish to restrict. In our demonstration, we'll restrict the entire document root (the entire website) which is based at /var/www/html .

A screenshot of a Linux desktop environment. On the left is a dock with various icons. In the center is a terminal window titled ".htaccess /var/www/html". The terminal shows the creation of a new .htaccess file in the "/var/www/html" directory. It includes commands to generate a password hash using htpasswd and to edit the Apache configuration file (apache2.conf) using gedit. The terminal also shows a warning message about GVfs metadata support.

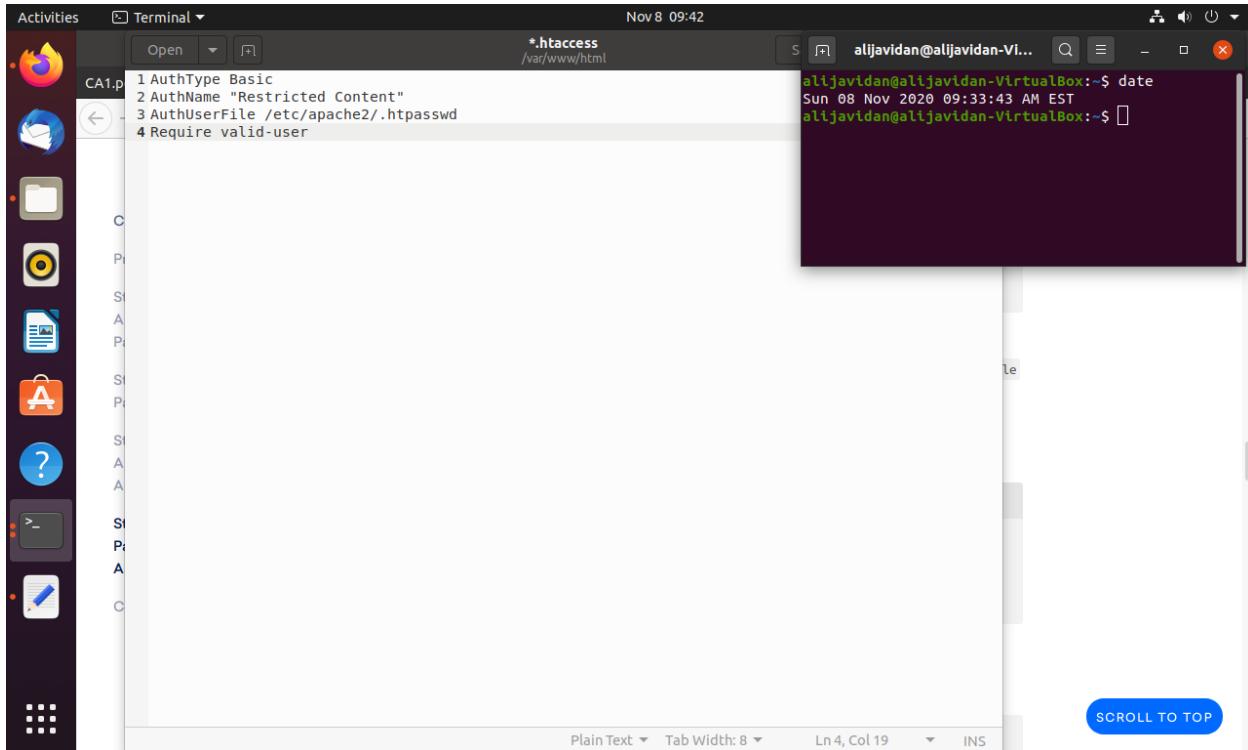
```
Nov 8 09:42 .htaccess /var/www/html
alijavidan@alijavidan-VirtualBox:~$ date
Sun 08 Nov 2020 09:33:43 AM EST
alijavidan@alijavidan-VirtualBox:~$ 
```

```
alijavidan@alijavidan-VirtualBox:~$ sudo htpasswd -c /etc/apache2/.htpasswd javidan
javidan:$apr1$yHLHHLqa$FbFnfRuzsoCnzbtIEMGX01
alijavidan@alijavidan-VirtualBox:~$ sudo gedit /etc/apache2/apache2.conf

(gedit:19364): Tepl-WARNING **: 09:41:01.331: GVfs metadata is not supported. Fallback to TeplMetadataManager. Either GVfs is not correctly installed or GVfs metadata are not supported on this platform. In the latter case, you should configure Tepl with --disable-gvfs-metadata.
alijavidan@alijavidan-VirtualBox:~$ sudo gedit /var/www/html/.htaccess

(gedit:19491): Tepl-WARNING **: 09:41:50.892: GVfs metadata is not supported. Fallback to TeplMetadataManager. Either GVfs is not correctly installed or GVfs metadata are not supported on this platform. In the latter case, you should configure Tepl with --disable-gvfs-metadata.
alijavidan@alijavidan-VirtualBox:~$ 
```

Within this file, we specify that we wish to set up Basic authentication. For the AuthName, we choose a realm name that will be displayed to the user when prompting for credentials. We use the AuthUserFile directive to point Apache to the password file we created. Finally, we will require a valid-user to access this resource, which means anyone who can verify their identity with a password will be allowed in:



The screenshot shows a Linux desktop environment with a terminal window and a code editor. The terminal window is titled 'Terminal' and shows the command 'date' being run, displaying the current date and time: 'Sun 08 Nov 2020 09:33:43 AM EST'. The code editor window is titled 'CA1.p' and contains an .htaccess file with the following content:

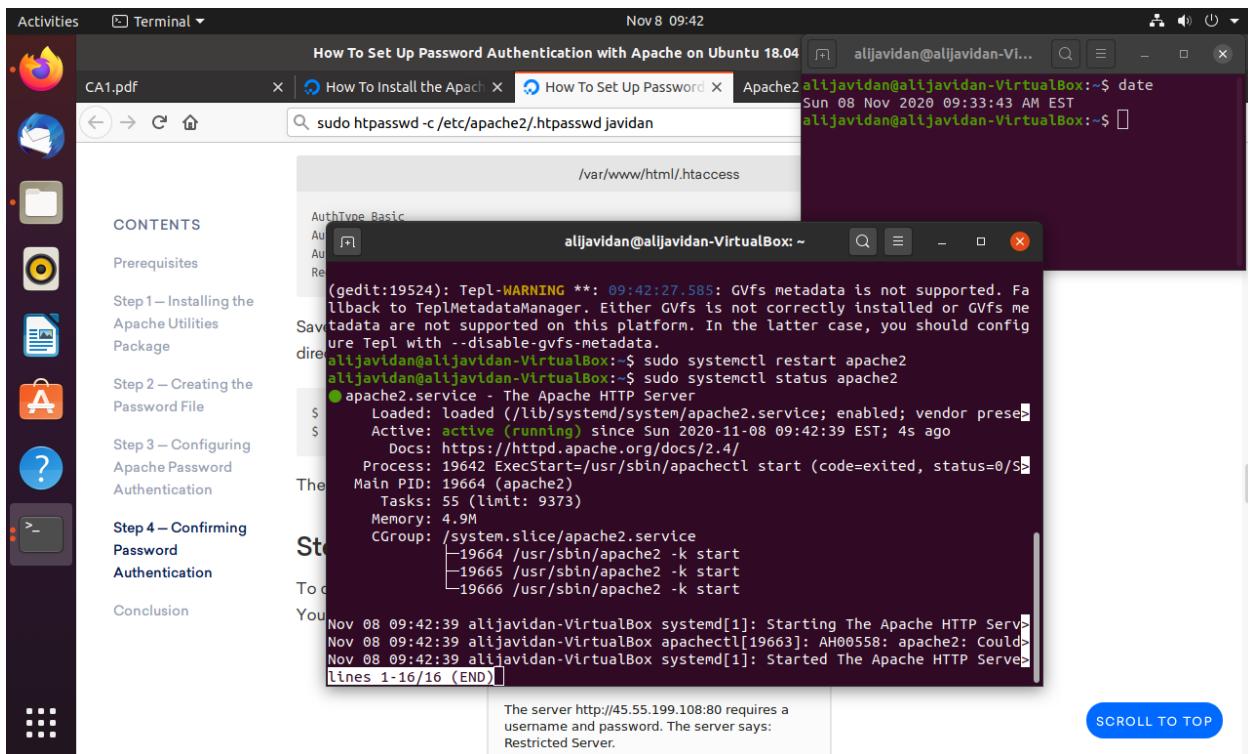
```
1 AuthType Basic
2 AuthName "Restricted Content"
3 AuthUserFile /etc/apache2/.htpasswd
4 Require valid-user
```

The desktop interface includes a dock with various application icons on the left.

We restart the web server to password protect all content in or below the directory with the .htaccess file and use

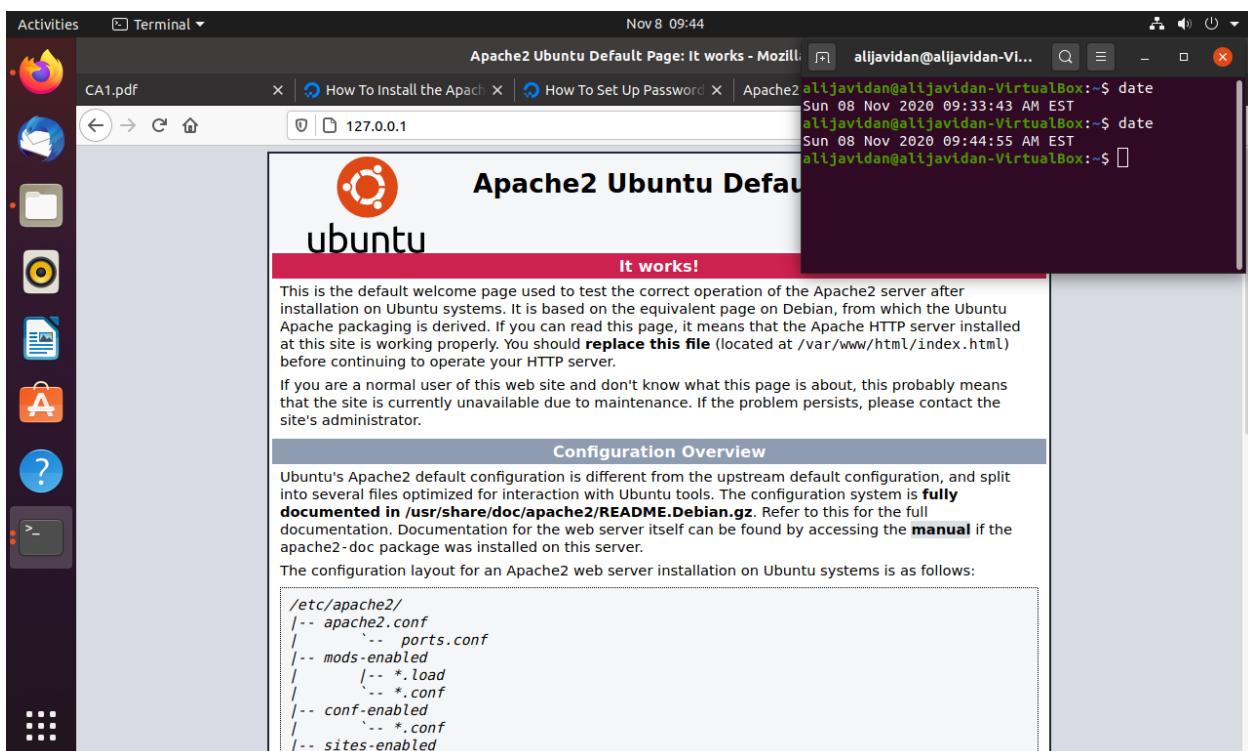
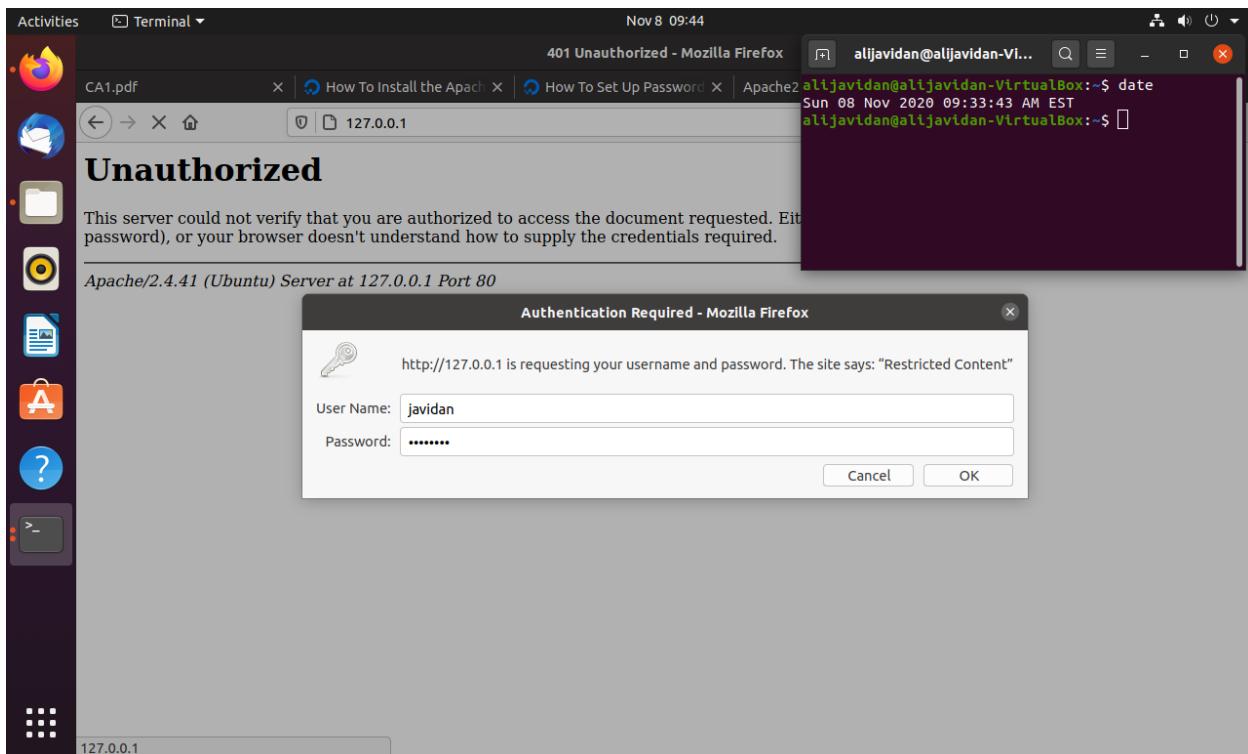
```
sudo systemctl status apache2
```

to verify the success of the restart.

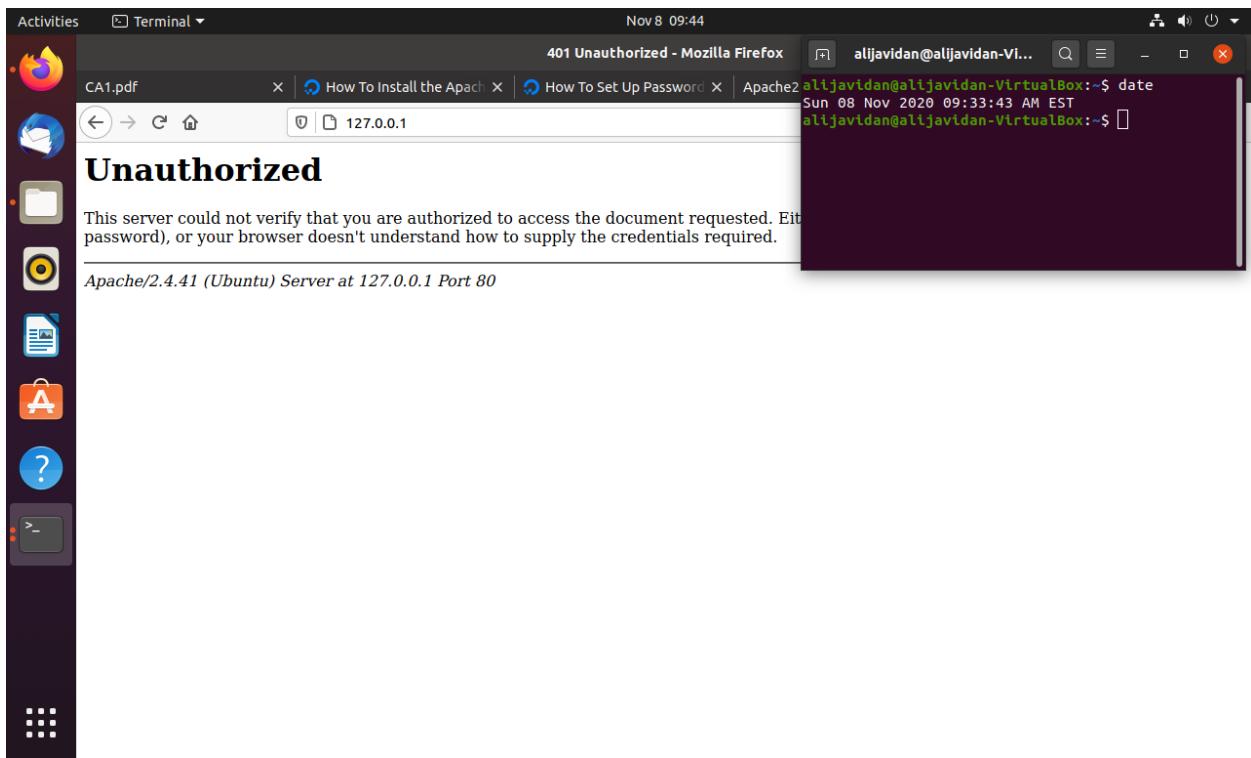


Step 4 — Confirming Password Authentication

To confirm that our content is protected, we try to access our restricted content in a web browser. We should be presented with a username and password prompt that looks like below screenshot. If we enter the correct credentials, we will be allowed to access the content.



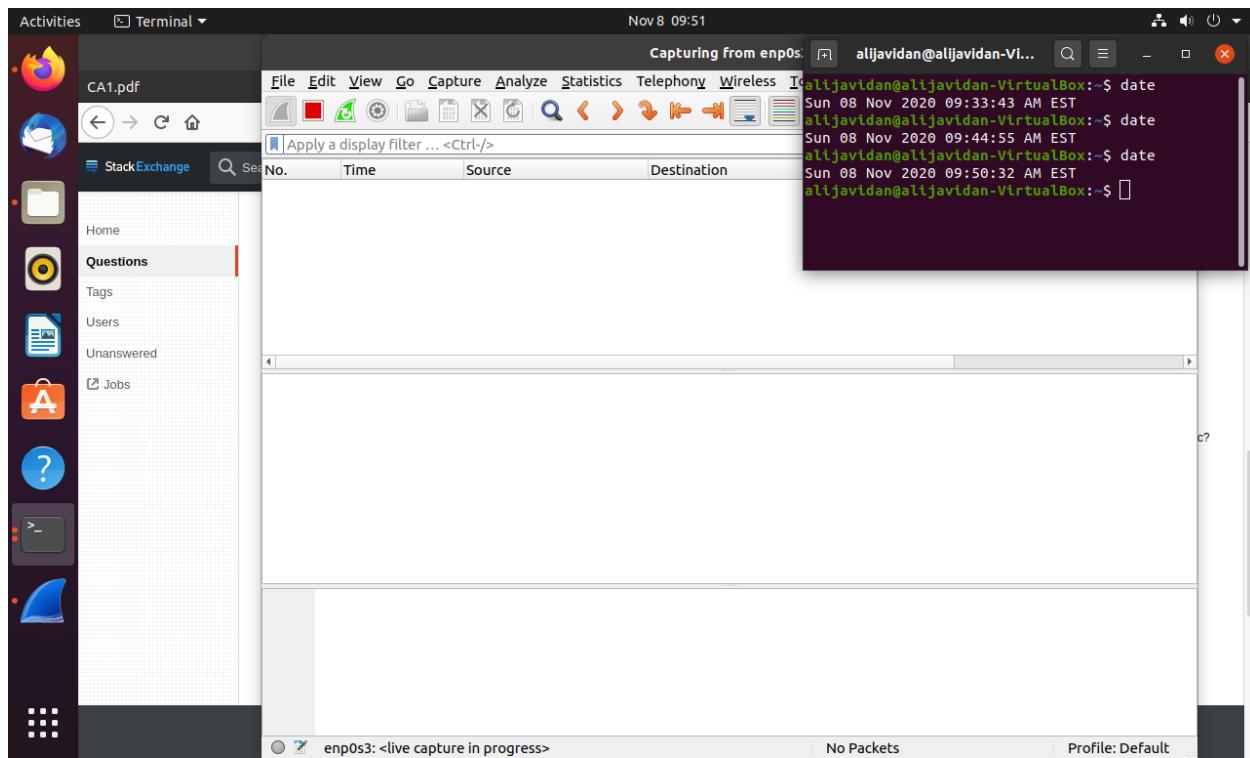
If we enter the wrong credentials or hit “Cancel”, we will see the “Unauthorized” error page:



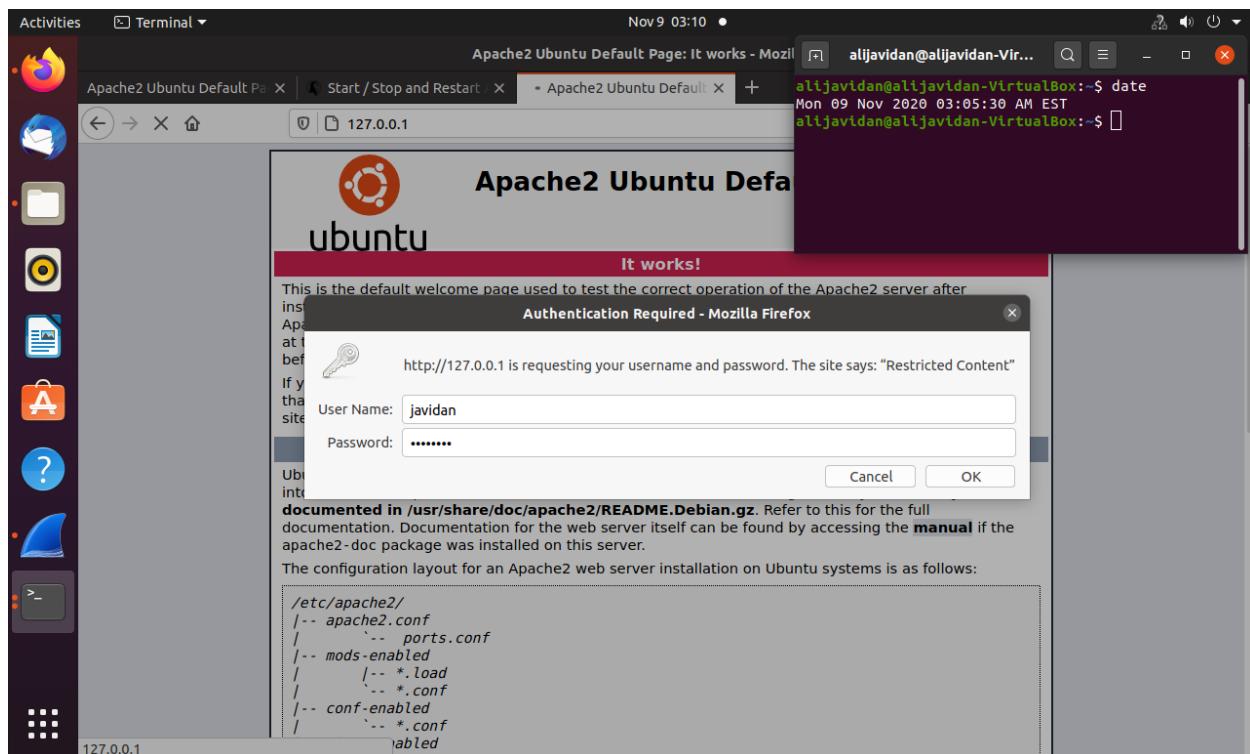
Wireshark

Since all localhost (127.0.0.1) traffic gets passed through loopback port, we could use tools like Wireshark to capture them.

We start Wireshark capture,

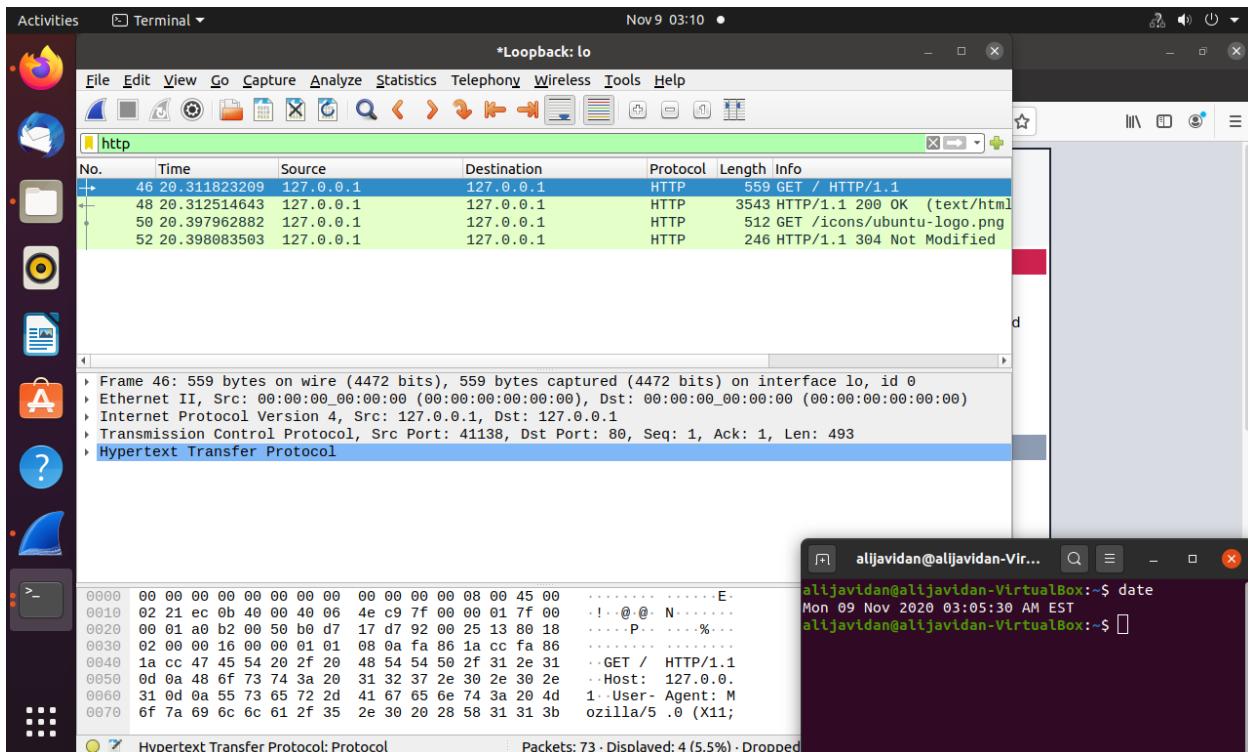


and login to Apache homepage with our creditioanal (javidan as User Name) through authentication procedure we have previously built up.

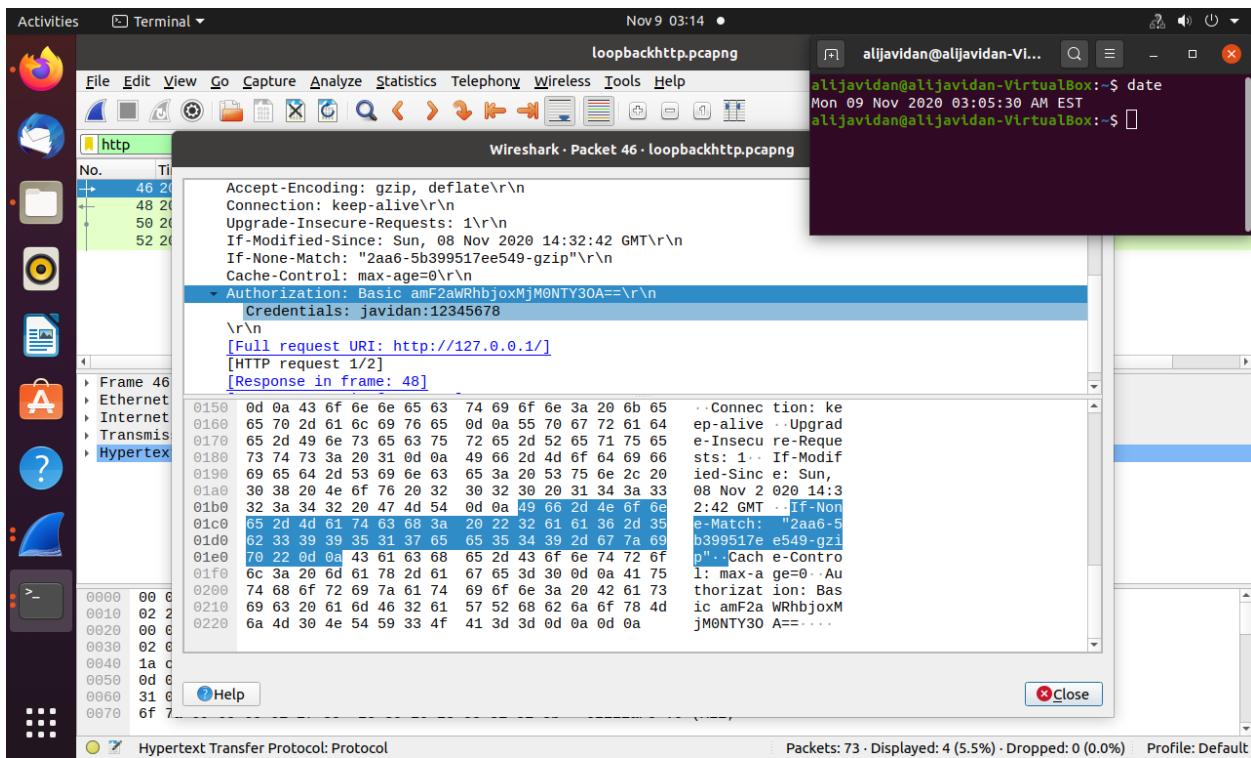




Now we stop Wireshark capture, filter out http packets as below:



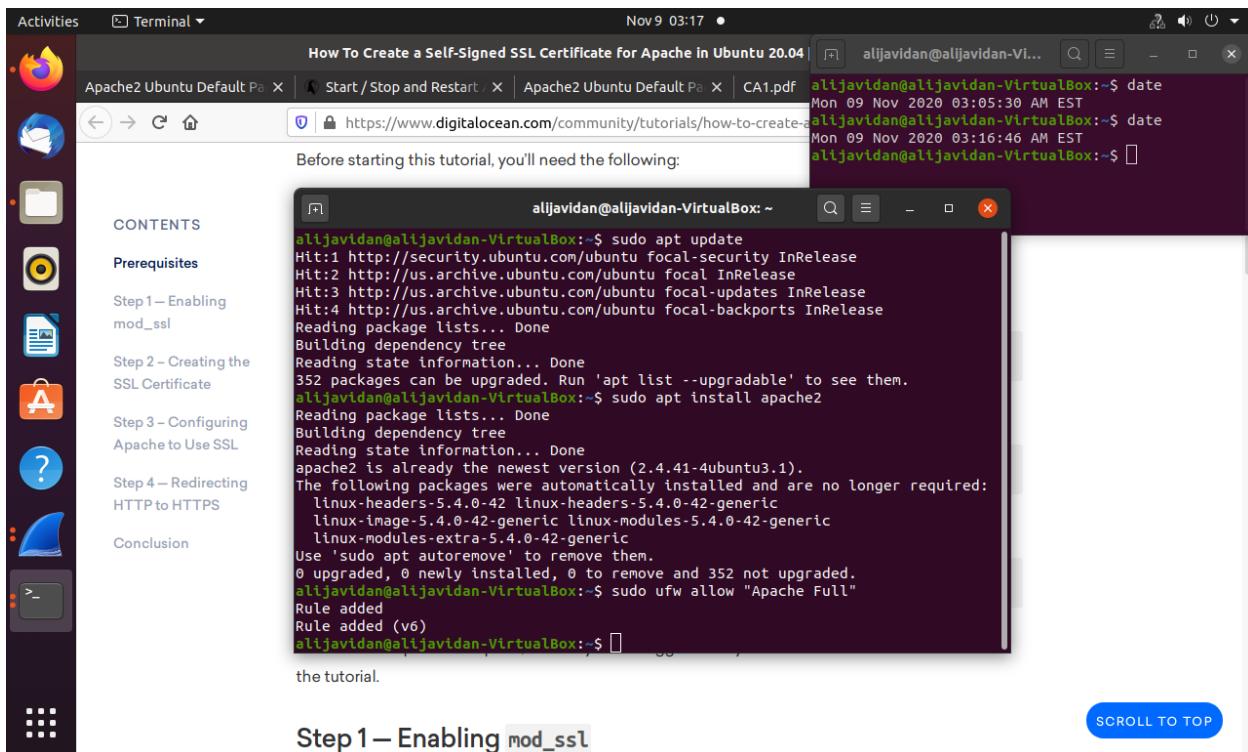
We could see credentials we used to login under Authorization parameter as follow:



This is because datas sent on channel are not encroyted, resulting in them being easily sniffed.

Create a Self-Signed SSL Certificate

Since we have a ufw firewall set up, we need to open up the http and https ports:



Step 1 – Enabling mod_ssl

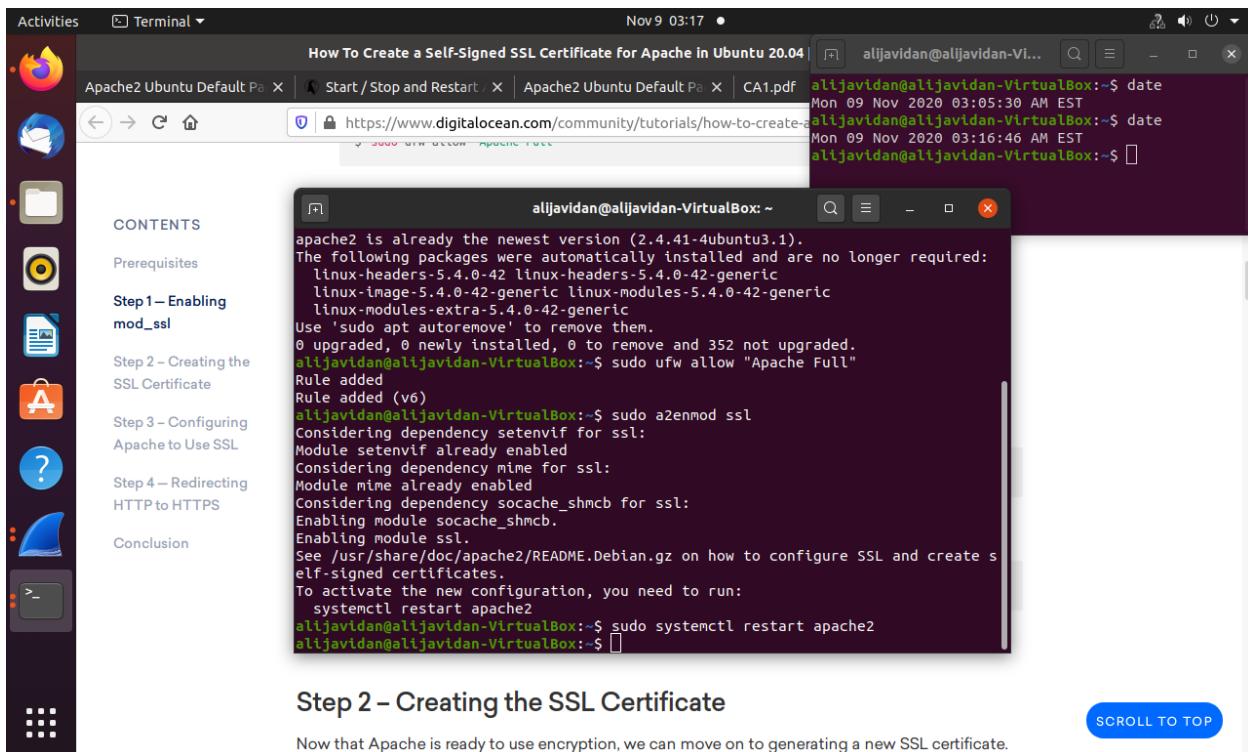
[SCROLL TO TOP](#)

2. Set up a HTTPS server with user authentication

Step 1 — Enabling mod_ssl

Before we can use any SSL certificates, we first have to enable mod_ssl, an Apache module that provides support for SSL encryption.

We Restart Apache to activate the module.



Step 2 – Creating the SSL Certificate

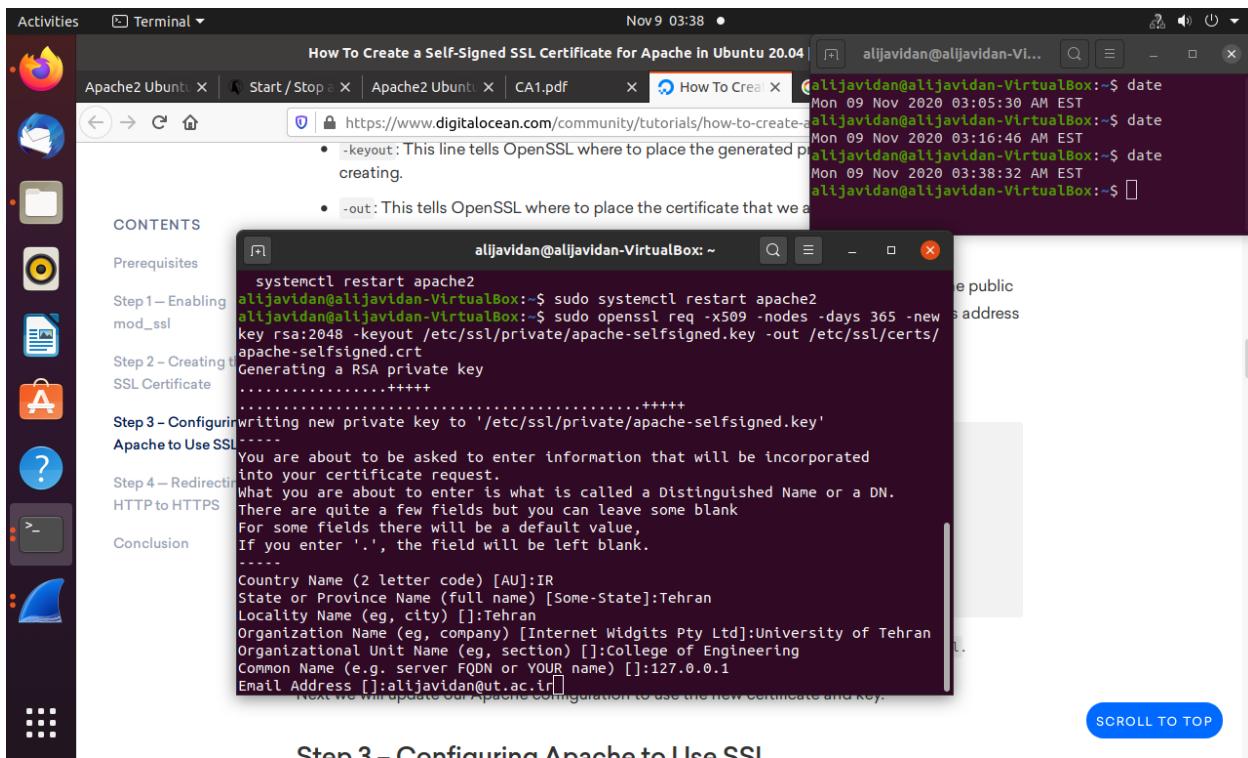
Now that Apache is ready to use encryption, we can move on to generating a new SSL certificate.

[SCROLL TO TOP](#)

Step 2 – Creating the SSL Certificate

Now that Apache is ready to use encryption, we can move on to generating a new SSL certificate. The certificate will store some basic information about our site, and will be accompanied by a key file that allows the server to securely handle encrypted data.

After we enter the command, we will be taken to a prompt where we can enter information about our website.



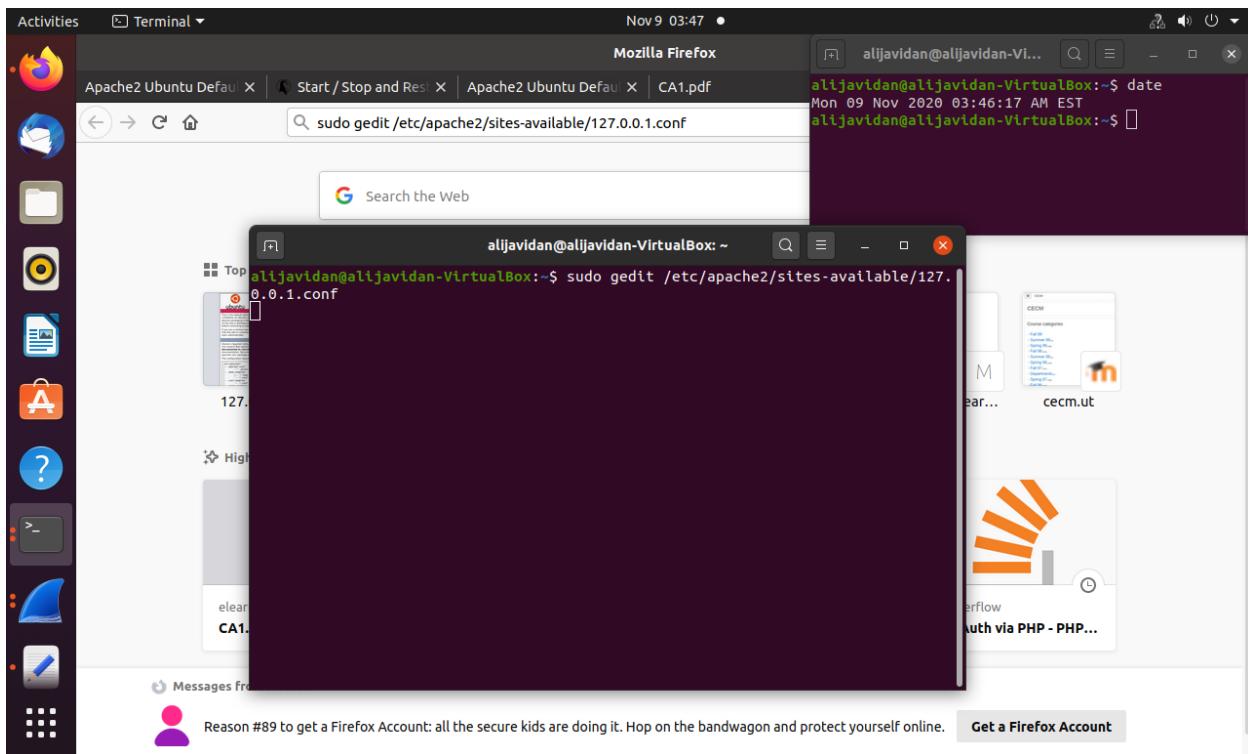
Both of the files we created will be placed in the appropriate subdirectories under /etc/ssl.

Step 3 – Configuring Apache to Use SSL

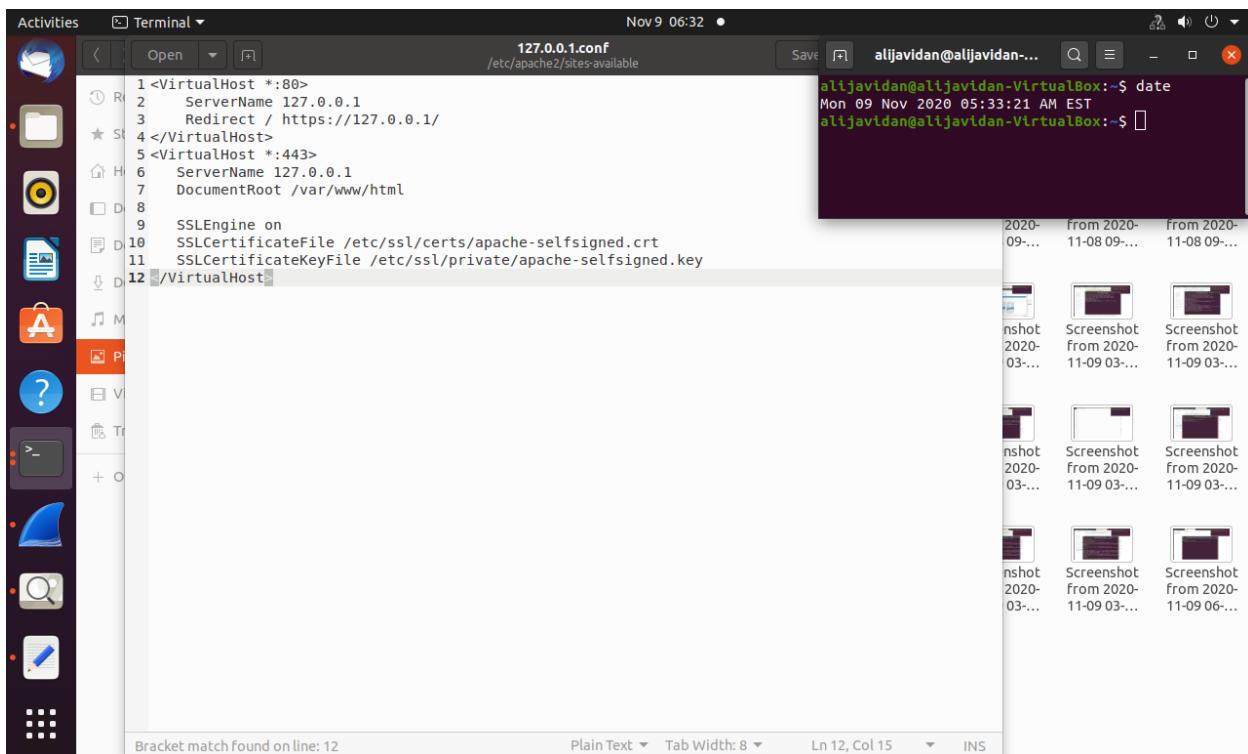
Now that we have our self-signed certificate and key available, we need to update our Apache configuration to use them. On Ubuntu, we can place new Apache configuration files (they must end in .conf) into /etc/apache2/sites-available/and they will be loaded the next time the Apache process is reloaded or restarted.

We will create a new minimal configuration file.

We open a new file in the /etc/apache2/sites-available directory,



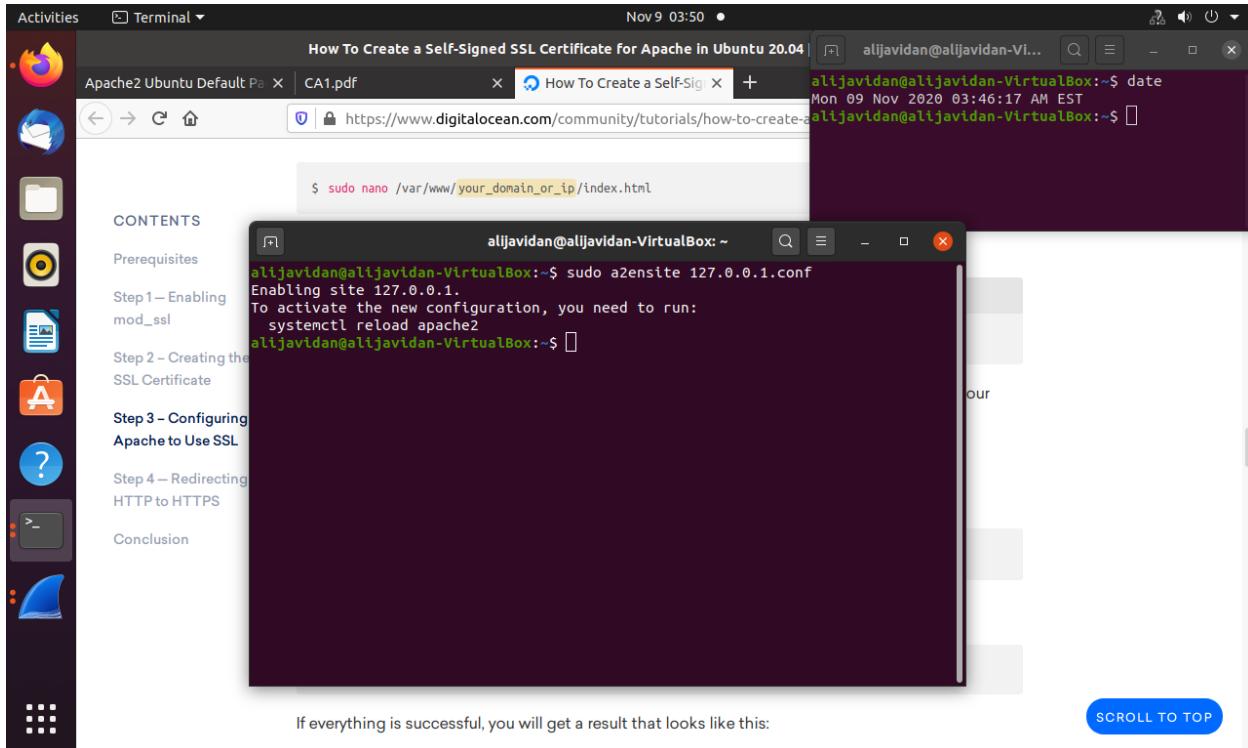
and paste in the following minimal VirtualHost configuration:



We should make sure ServerName matches the Common Name we chose when making the certificate.

The remaining lines specify a DocumentRoot directory to serve files from, and the SSL options needed to point Apache to our newly-created certificate and key.

Next, let's test for configuration errors:



If everything is successful, you will get a result that looks like this:

A screenshot of a Linux desktop environment. On the left is a vertical dock with icons for various applications like a browser, file manager, and terminal. The main window shows a terminal session titled "How To Create a Self-Signed SSL Certificate for Apache in Ubuntu 20.04". The terminal output is as follows:

```
alijavidan@alijavidan-VirtualBox:~$ sudo a2ensite 127.0.0.1.conf
Enabling site 127.0.0.1.
To activate the new configuration, you need to run:
    systemctl reload apache2
alijavidan@alijavidan-VirtualBox:~$ sudo apache2ctl configtest
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
Syntax OK
alijavidan@alijavidan-VirtualBox:~$
```

The terminal has a scroll bar on the right. Below the terminal window, there is a small "Output" section containing the same error message from the configtest command. A blue "SCROLL TO TOP" button is located at the bottom right of the terminal window.

Now that output has Syntax OK in it, our configuration file has no syntax errors. We can safely reload Apache to implement our changes.

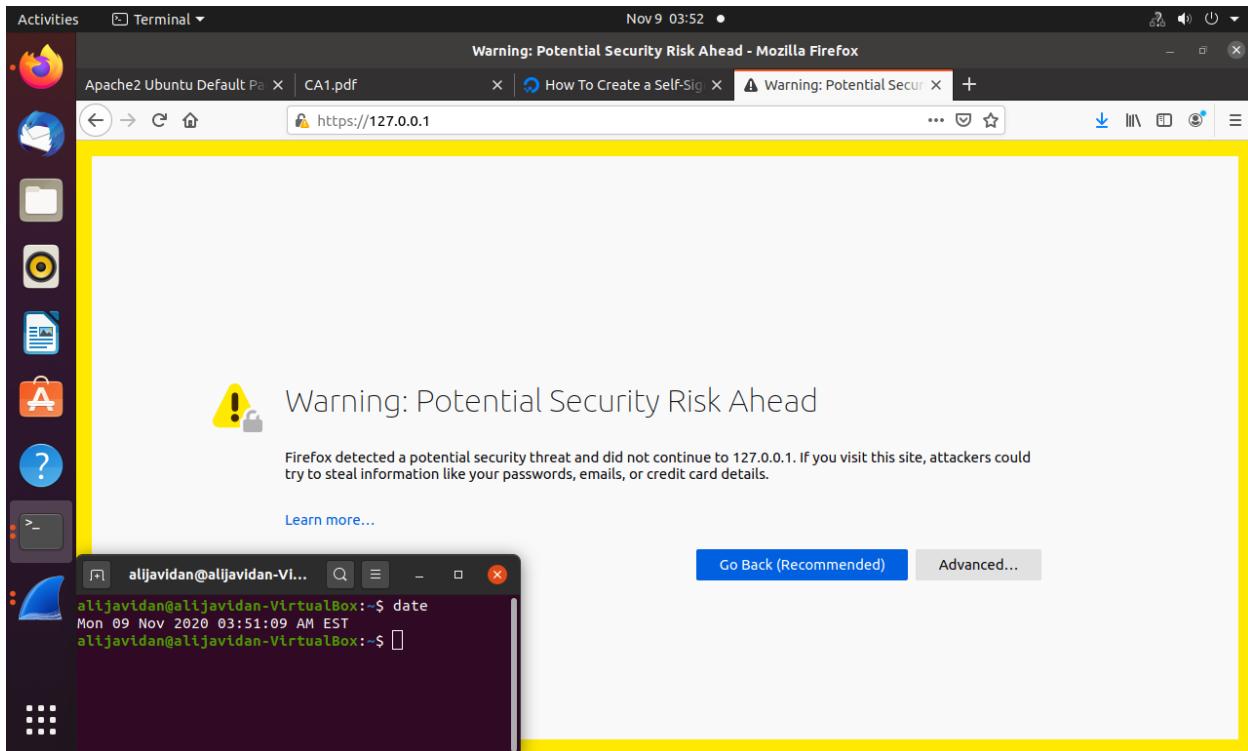
A screenshot of a Linux desktop environment, similar to the one above. The terminal window shows the following output:

```
alijavidan@alijavidan-VirtualBox:~$ sudo a2ensite 127.0.0.1.conf
Enabling site 127.0.0.1.
To activate the new configuration, you need to run:
    systemctl reload apache2
alijavidan@alijavidan-VirtualBox:~$ sudo apache2ctl configtest
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
Syntax OK
alijavidan@alijavidan-VirtualBox:~$ sudo systemctl reload apache2
alijavidan@alijavidan-VirtualBox:~$
```

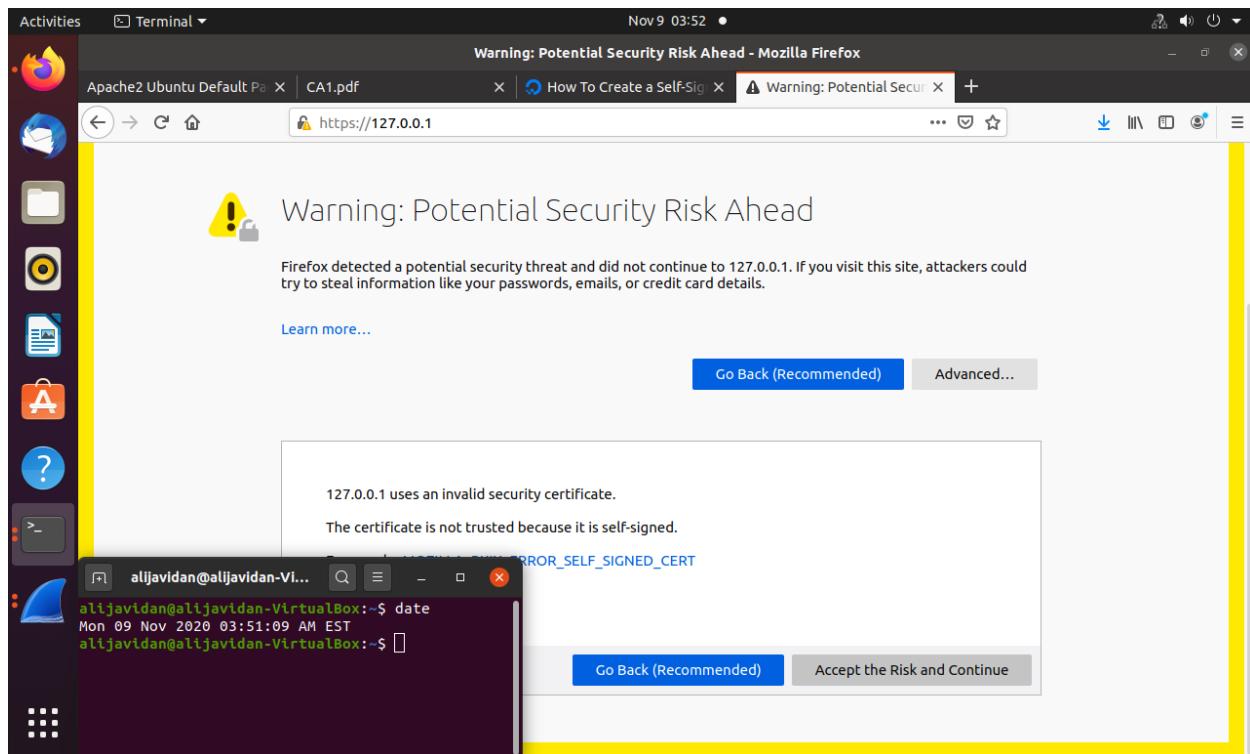
At the bottom of the terminal window, there is a message: "After you do so, your browser will load the `it worked!` message." A blue "SCROLL TO TOP" button is located at the bottom right of the terminal window.

Now we load our site in a browser.

We should see an error. This is normal for a self-signed certificate! The browser is warning us that it can't verify the identity of the server, because our certificate is not signed by any of its known certificate authorities. For testing purposes and personal use this can be fine.

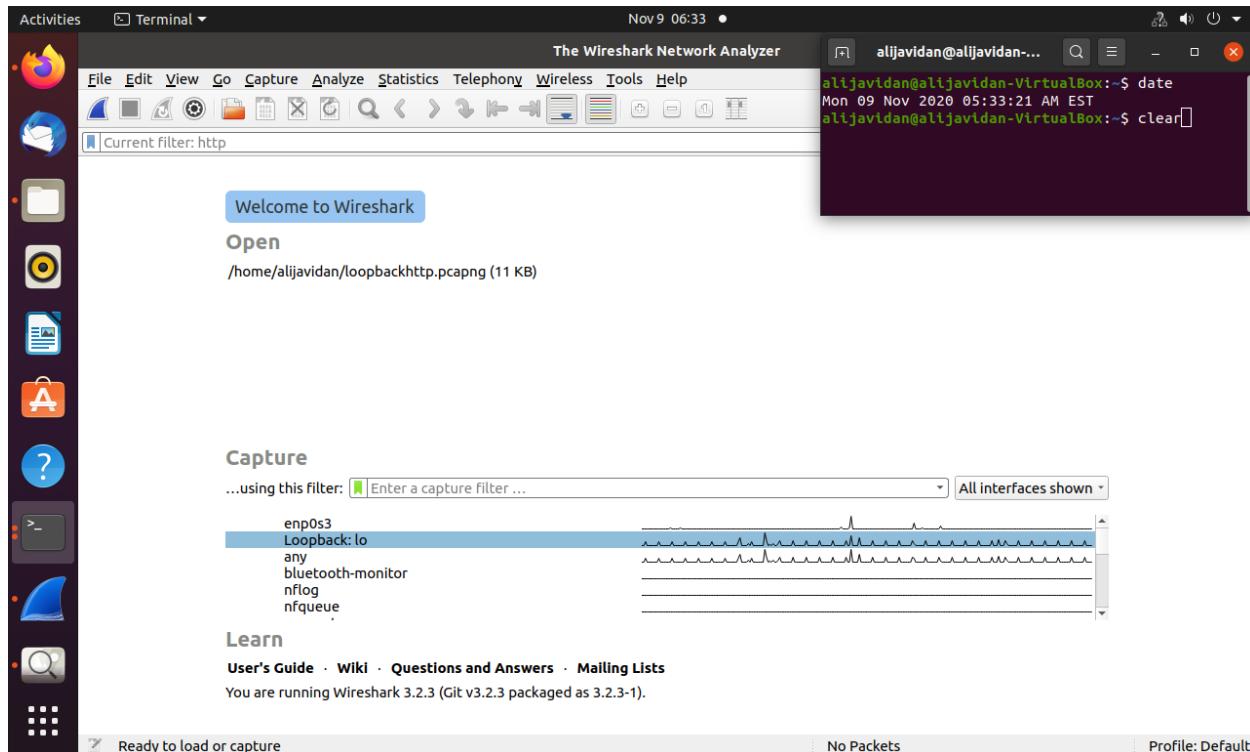


We should be able to click through to advanced or more information and choose to proceed.

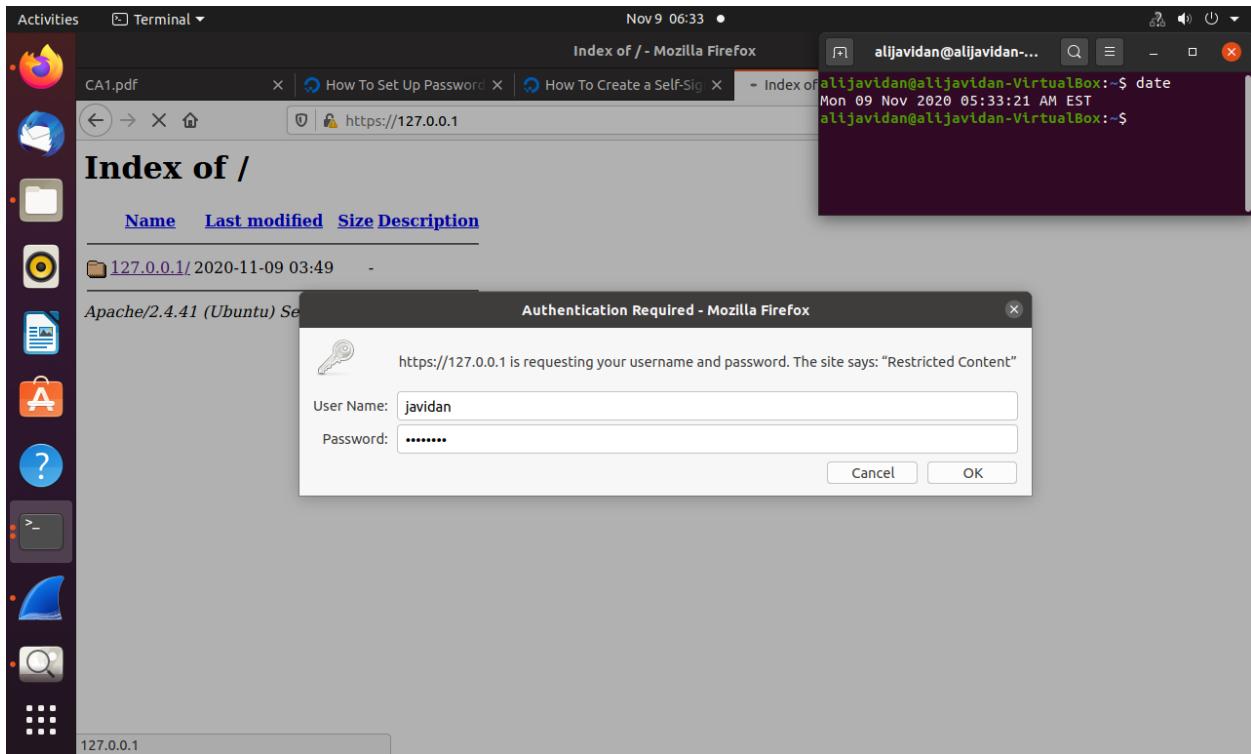


Wireshark

We start Wireshark capture,



and login to Apache homepage with our creditioanlal (javidan as User Name) through authentication procedure we have previously built up.



Now we stop Wireshark capture,

but this time we could no longer extarc any creditionals from captured packets, *since in https they are encrypted and not sent as plain-text like what was like in http before.*

