**5. Installation and Configuration of Apache server in Ubantu.**

**Introduction**

The Apache HTTP server is the most widely-used web server in the world. It provides many powerful features including dynamically loadable modules, robust media support, and extensive integration with other popular software.

## Step 1 — Installing Apache

Apache is available within Ubuntu’s default software repositories, making it possible to install it using conventional package management tools. Let’s begin by updating the local package index to reflect the latest upstream changes:

**$sudo apt update**

Then, install the apache2 package:

**$sudo apt install apache2**

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

## Step 2 — Adjusting the Firewall

Before testing Apache, it’s necessary to modify the firewall settings to allow outside access to the default web ports. 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.

List the ufw application profiles by typing:

**$sudo ufw app list**

You will receive a list of the application profiles: As indicated by the output, there are three profiles available for Apache:

* **Apache**: This profile opens only port 80 (normal, unencrypted web traffic)
* **Apache Full**: This profile opens both port 80 (normal, unencrypted web traffic) and port 443 (TLS/SSL encrypted traffic)
* **Apache Secure**: This profile opens only port 443 (TLS/SSL encrypted traffic)

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

**$sudo ufw allow ‘apache’**

You can verify the change by typing:

**$sudo ufw status**

The output will provide a list of allowed HTTP traffic:

**Output:**

Status: active

To Action From

-- ------ ----

OpenSSH ALLOW Anywhere

Apache ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

Apache (v6) ALLOW Anywhere (v6)

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

## Step 3 — Checking your Web Server

At the end of the installation process, Ubuntu 20.04 starts Apache. The web server should already be up and running. Check with the systemctl init system to make sure the service is running by typing:

**$sudo systemctl status apache2**

As confirmed by this output, the service has started successfully. However, the best way to test this is to request a page from Apache. You can access the default Apache landing page to confirm that the software is running properly through your IP address or Domain name. If you do not know your server’s IP address, you can get it a few different ways from the command line.

Try typing this at your server’s command prompt:

**$hostname -I**

When you have your server’s IP address, enter it into your browser’s address bar:

[**http://your\_server\_ip**](http://your_server_ip)

**or**

[**http://domain\_name**](http://domain_name)

**Example:** [**http://192.168.1.147**](http://192.168.1.147) **(Note: this is IP address of my system)**

The default Ubuntu 20.04 Apache web page indicates that Apache is working correctly. It also includes some basic information about important Apache files and directory locations.

## Step 4 — Managing the Apache Process

Now that you have your web server up and running, let’s go over some basic management commands using systemctl.

To stop your web server, type:

**$sudo systemctl stop apache2**

To start the web server when it is stopped, type:

**$sudo systemctl start apache2**

To stop and then start the service again, type:

**$sudo systemctl restart apache2**

If you are simply making configuration changes, Apache can often reload without dropping connections. To do this, use this command:

**$sudo systemctl reload apache2**

By default, Apache is configured to start automatically when the server boots. If this is not what you want, disable this behavior by typing:

**$sudo systemctl disable apache2**

To re-enable the service to start up at boot, type:

**$sudo systemctl enable apache2**

Apache should now start automatically when the server boots again.

## Step 5 — Setting Up Virtual Hosts (Recommended)

When using the Apache web server, you can use **virtual hosts** to encapsulate configuration details and host more than one domain from a single server. We will set up a domain called **your\_domain**, but you should **replace this with your own domain name**.

Apache on Ubuntu 20.04 has one server block enabled by default that is configured to serve documents from the **/var/www/html** directory. While this works well for a single site, it can become unwieldy if you are hosting multiple sites. Instead of modifying /var/www/html, let’s create a directory structure within /var/www for a **your\_domain** site, leaving /var/www/html in place as the default directory to be served if a client request doesn’t match any other sites.

Create the directory for **your\_domain** as follows:

**$sudo mkdir /var/www/your\_domain**

**Example: $sudo mkdir /var/www/jssasma**

Next, assign ownership of the directory with the $USER environment variable:

**$sudo chown –R $USER:$USER /var/www/your\_domain**

**Example: $sudo chown –R $USER:$USER /var/www/jssasma**

To ensure that your permissions are correct and allow the owner to read, write, and execute the files while granting only read and execute permissions to groups and others, you can input the following command:

**$sudo chmod –R 755 /var/www/your\_domain**

**Example: $sudo chmod –R 755 /var/www/jssasma**

Next, create a sample index.html page using nano or your favorite editor:

**$sudo vi /var/www/your\_domain/index.html**

**Example: $sudo vi /var/www/jssasma/index.html**

**Add the following sample HTML inside /var/www/jssasma/index.html**

**<html>**

**<head>**

**<title>Welcome to Your\_domain!</title>**

**</head>**

**<body>**

**<h1>Success! The your\_domain virtual host is working!</h1>**

**</body>**

**</html>**

**Note: change your\_domain with jssasma in this example.**

Save and close the file when you are finished.

In order for Apache to serve this content, it’s necessary to create a virtual host file with the correct directives. Instead of modifying the default configuration file located at **/etc/apache2/sites-available/000-default.conf** directly, let’s make a new one at **/etc/apache2/sites-available/your\_domain.conf:**

**$sudo vi /etc/apache2/sites-available/your\_domain.conf**

**Example: $sudo vi /etc/apache2/sites-available/jssasma.conf**

Paste in the following configuration block, which is similar to the default, but updated for our new directory and domain name: **/etc/apache2/sites-available/jssasma.conf**

<VirtualHost \*:80>

ServerAdmin **asma.ron786@gmail.com**

ServerName **jssasma**

ServerAlias **www.jssasma**

DocumentRoot **/var/www/jssasma**

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

Save and close the file when you are finished.

Notice that we’ve updated the **DocumentRoot**to our new directory and **ServerAdmin** to an email that the **your\_domain** site administrator can access. We have also added two directives: **ServerName**. This establishes the base domain that should match for this virtual host definition. And **ServerAlias**, which defines further names that should match as if they were base names.

Let’s enable the file with the a2ensite tool:

**$sudo a2ensite your\_domain.conf**

**Example: $sudo a2ensite jssasma.conf**

Disable the default site defined in 000-default.conf:

**$sudo a2dissite 000-default.conf**

Next, let’s test for configuration errors:

**$sudo apache2ctl configtest**

You should receive the following output:

**Syntax OK**

Restart Apache to implement your changes:

**$sudo systemctl restart apache2**

Apache should now be serving your domain name. You can test this by navigating to [**http://your\_domain**](http://your_domain)from your browser.

**Example: http://** **192.168.1.147**

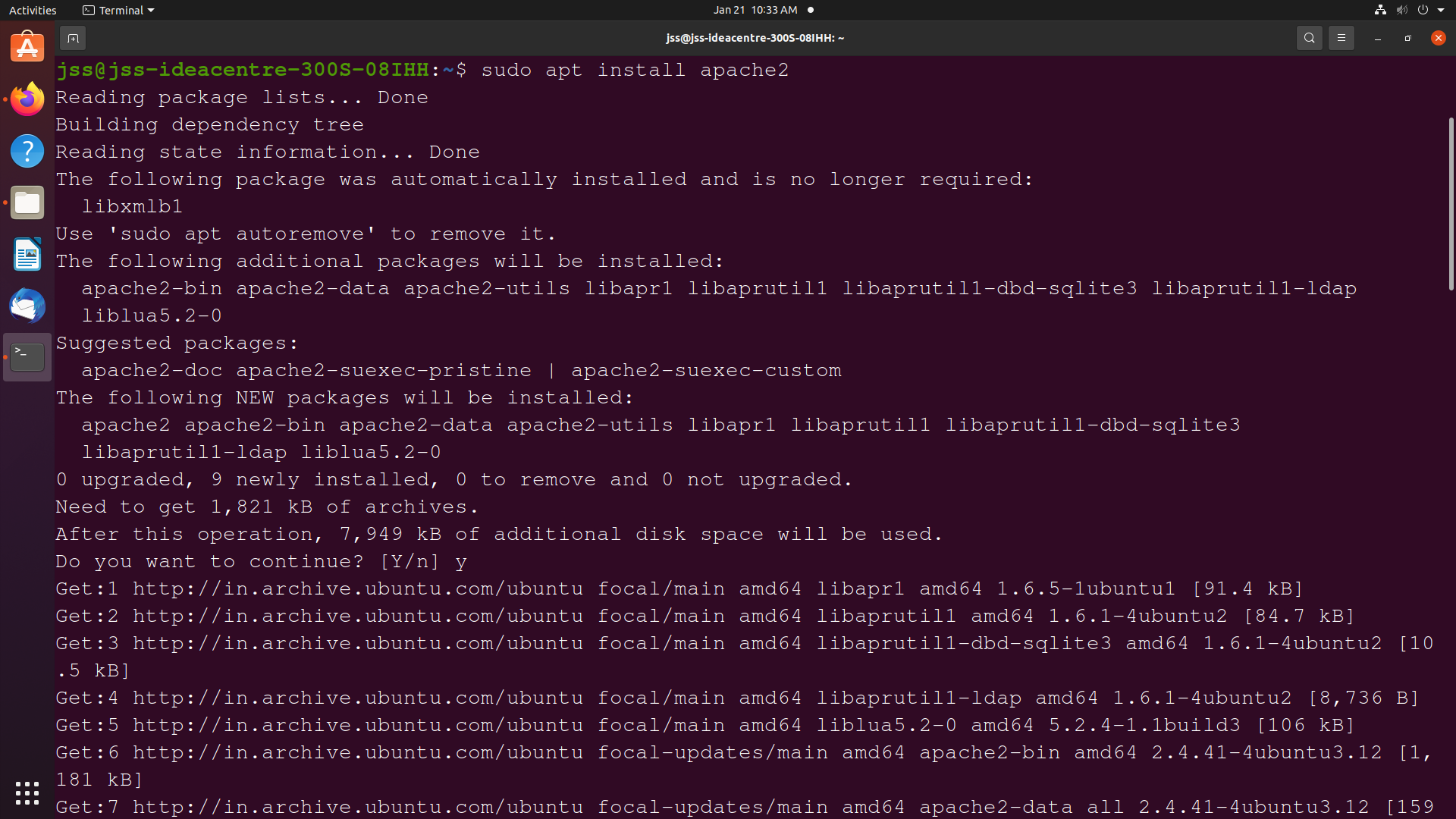
Or you can use your domain name instead of IP address. To set your domain type:

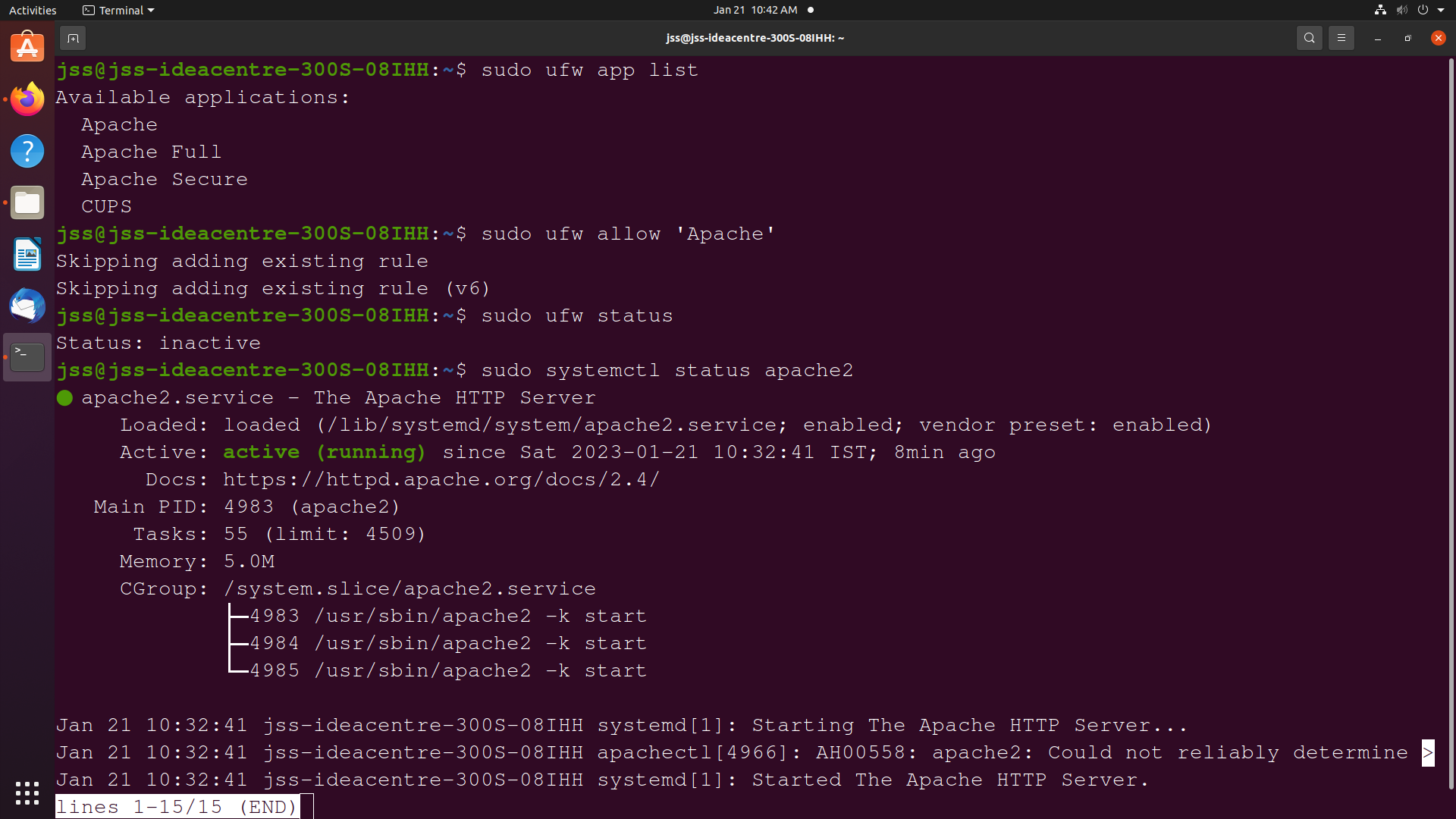
**$sudo hostname jssasma**

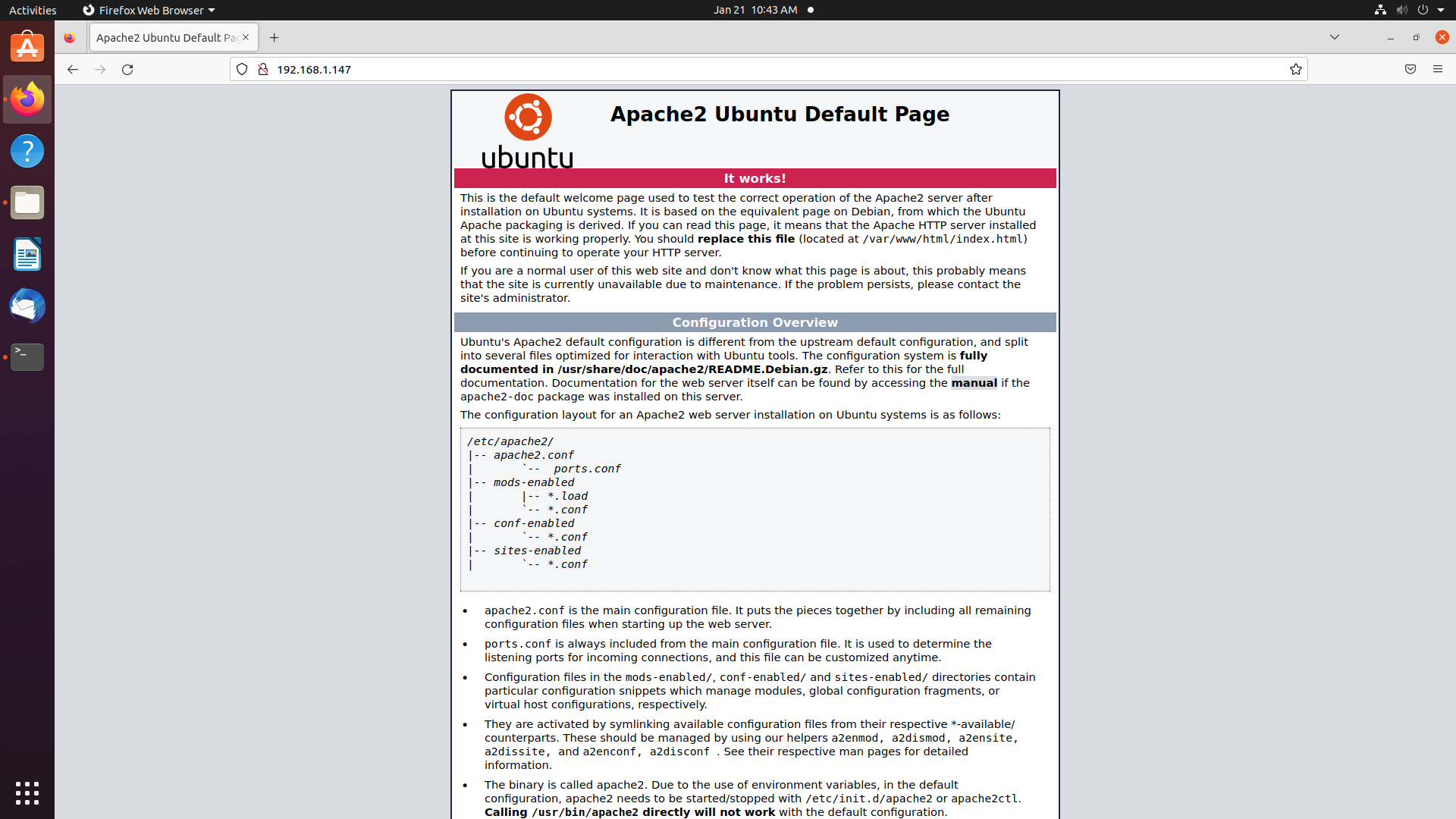
Then type the following in browser’s URL.

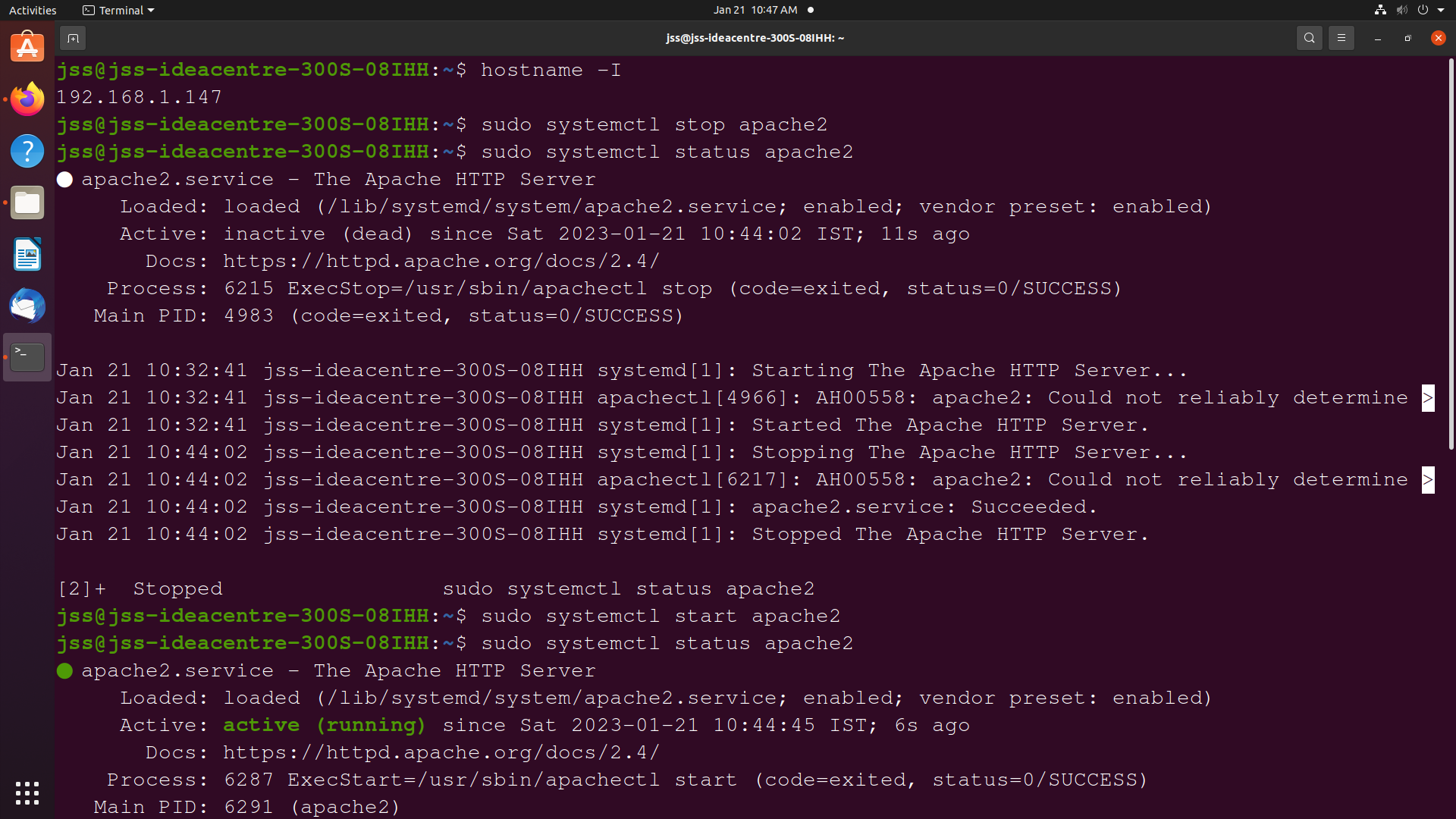
**Example:http://jssasma**

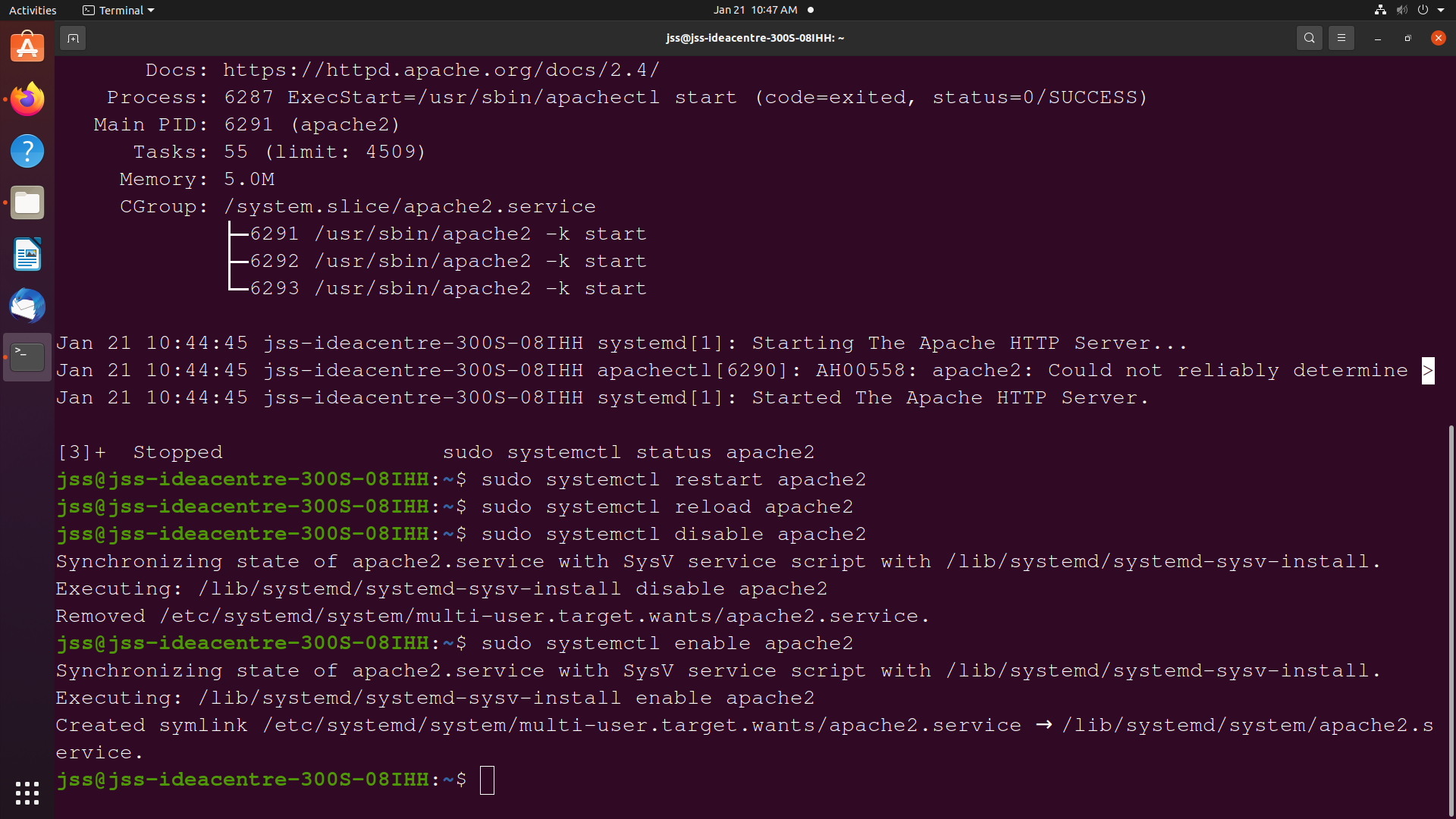
**Following screenshots demonstrate the output of each command.**

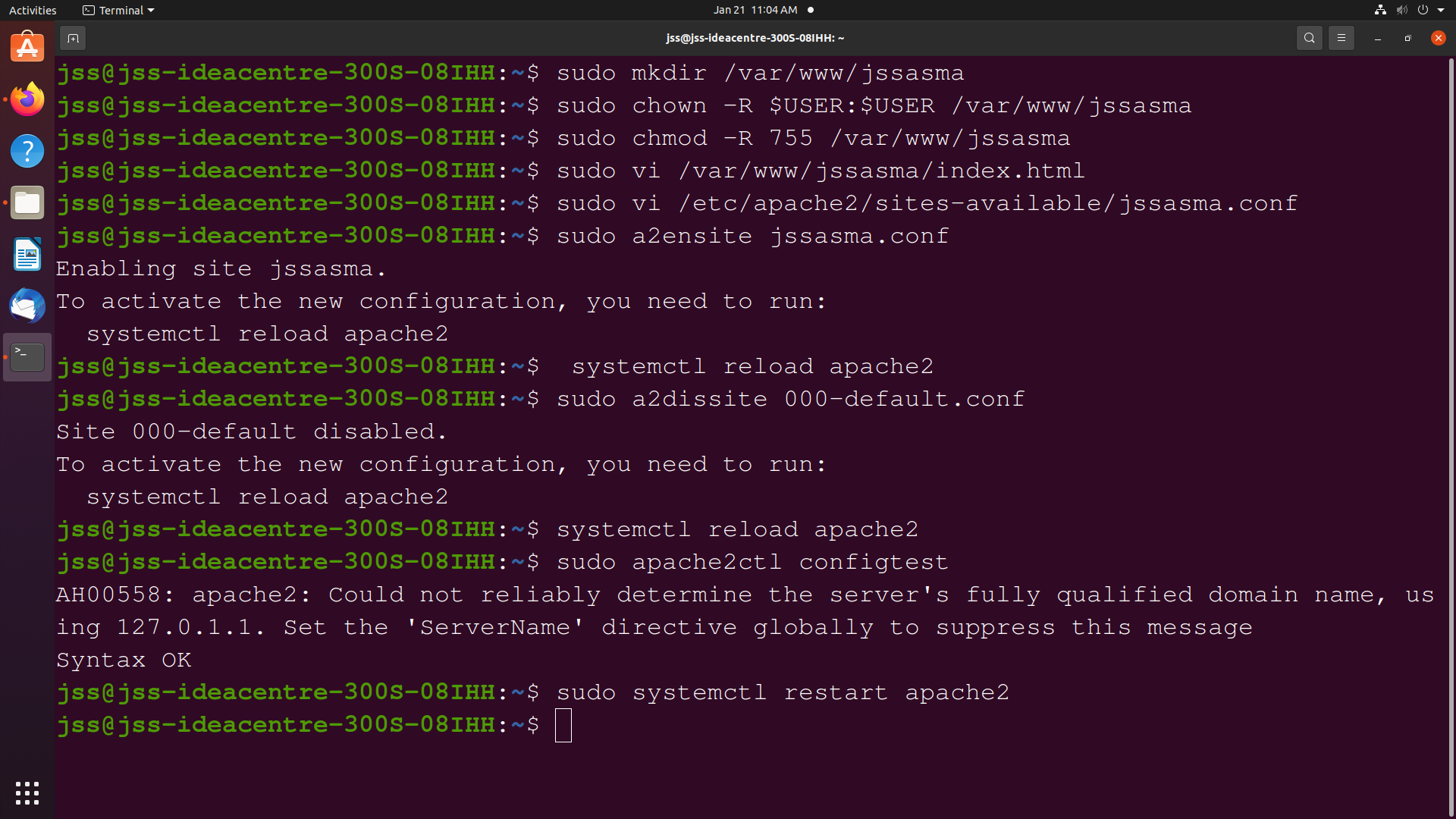
****

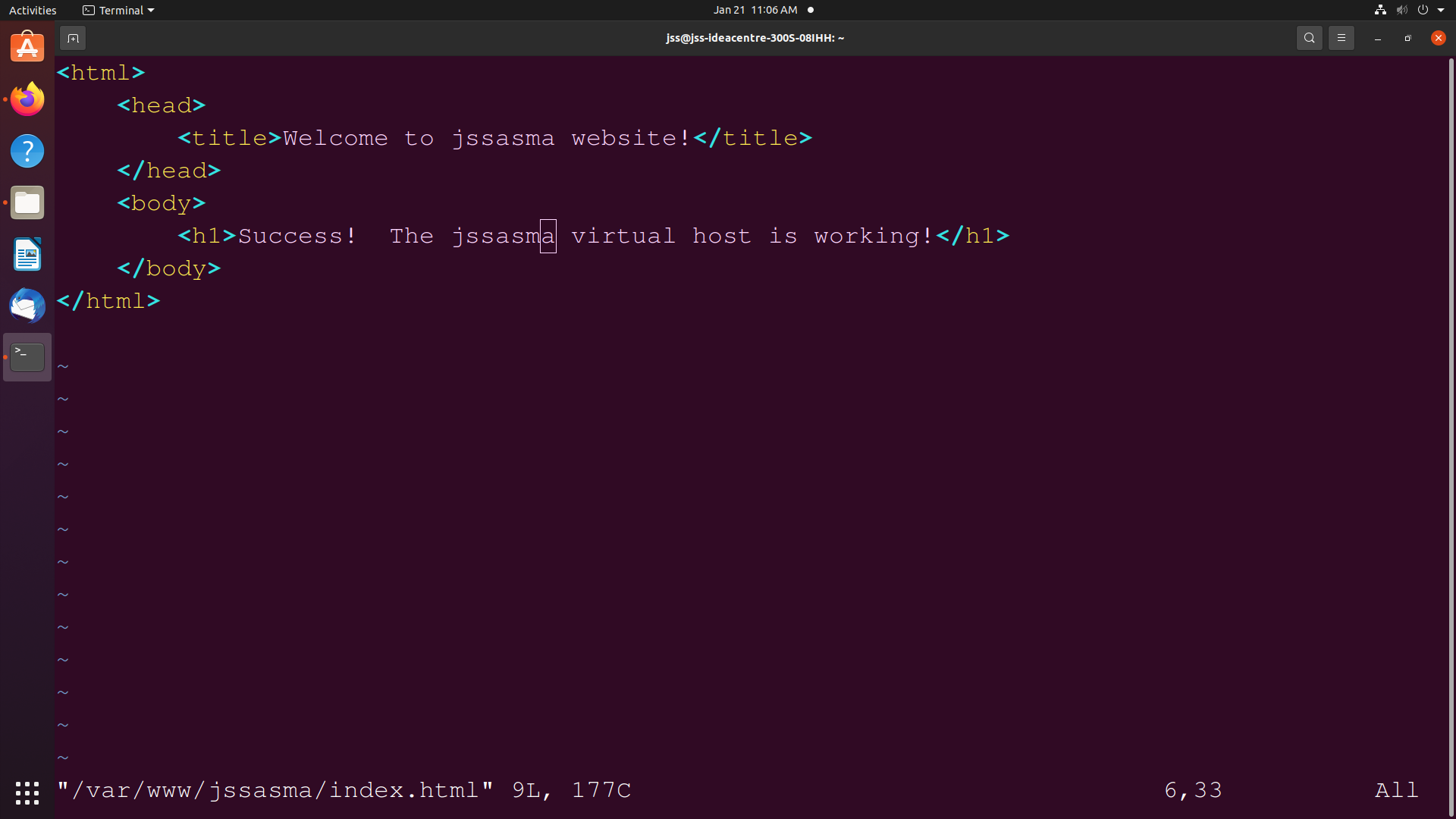
****

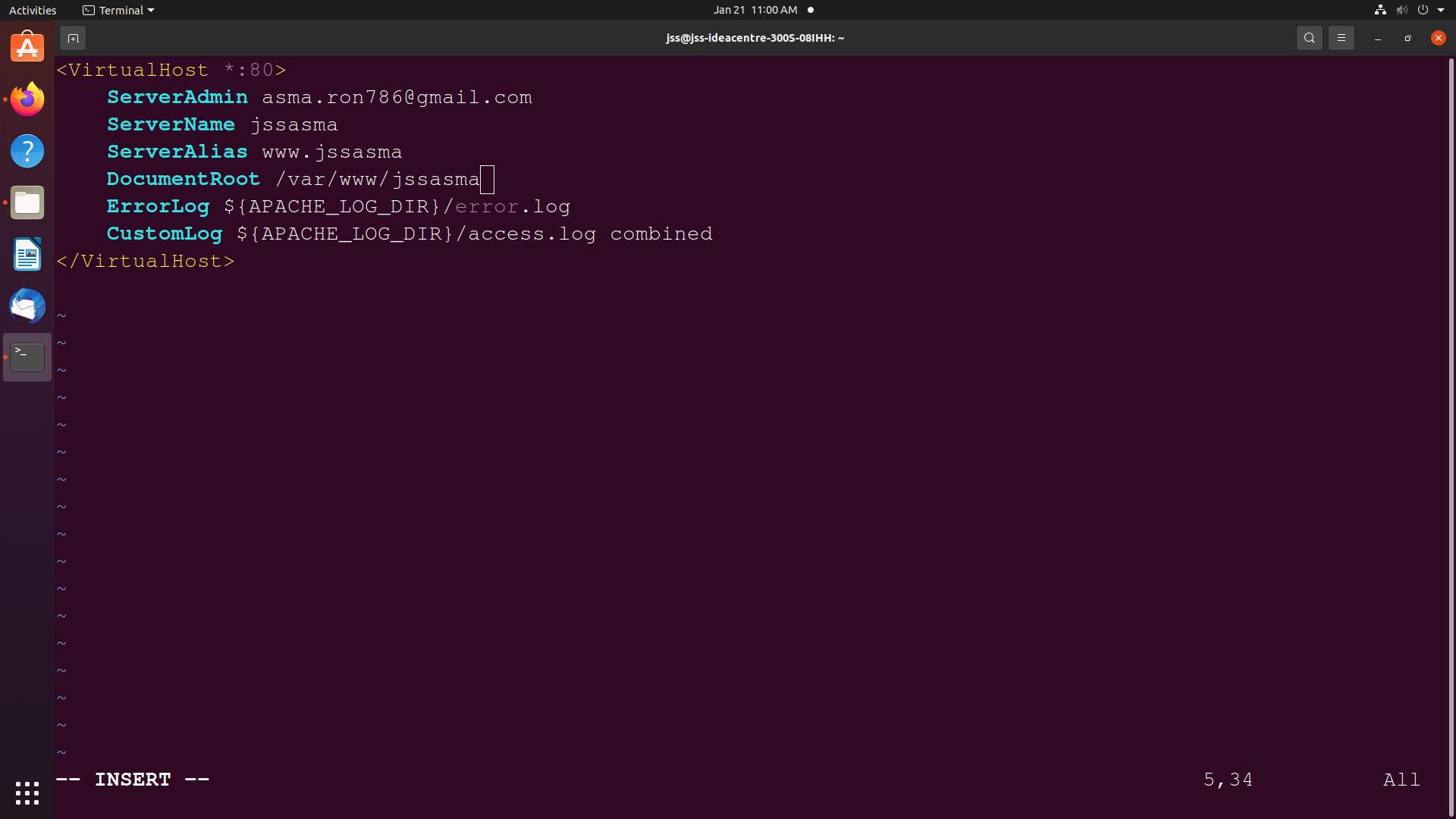
****

****

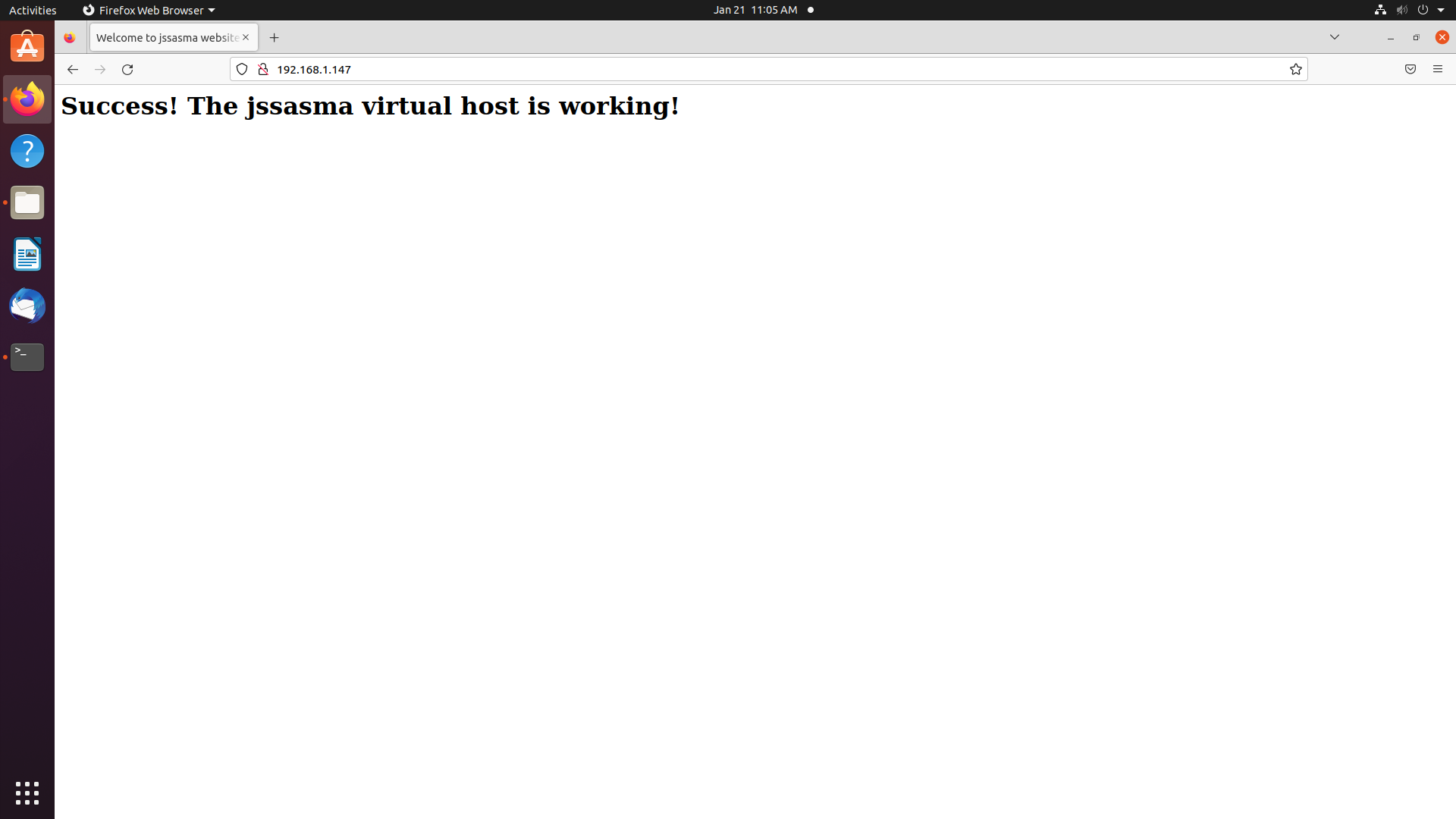
****

****

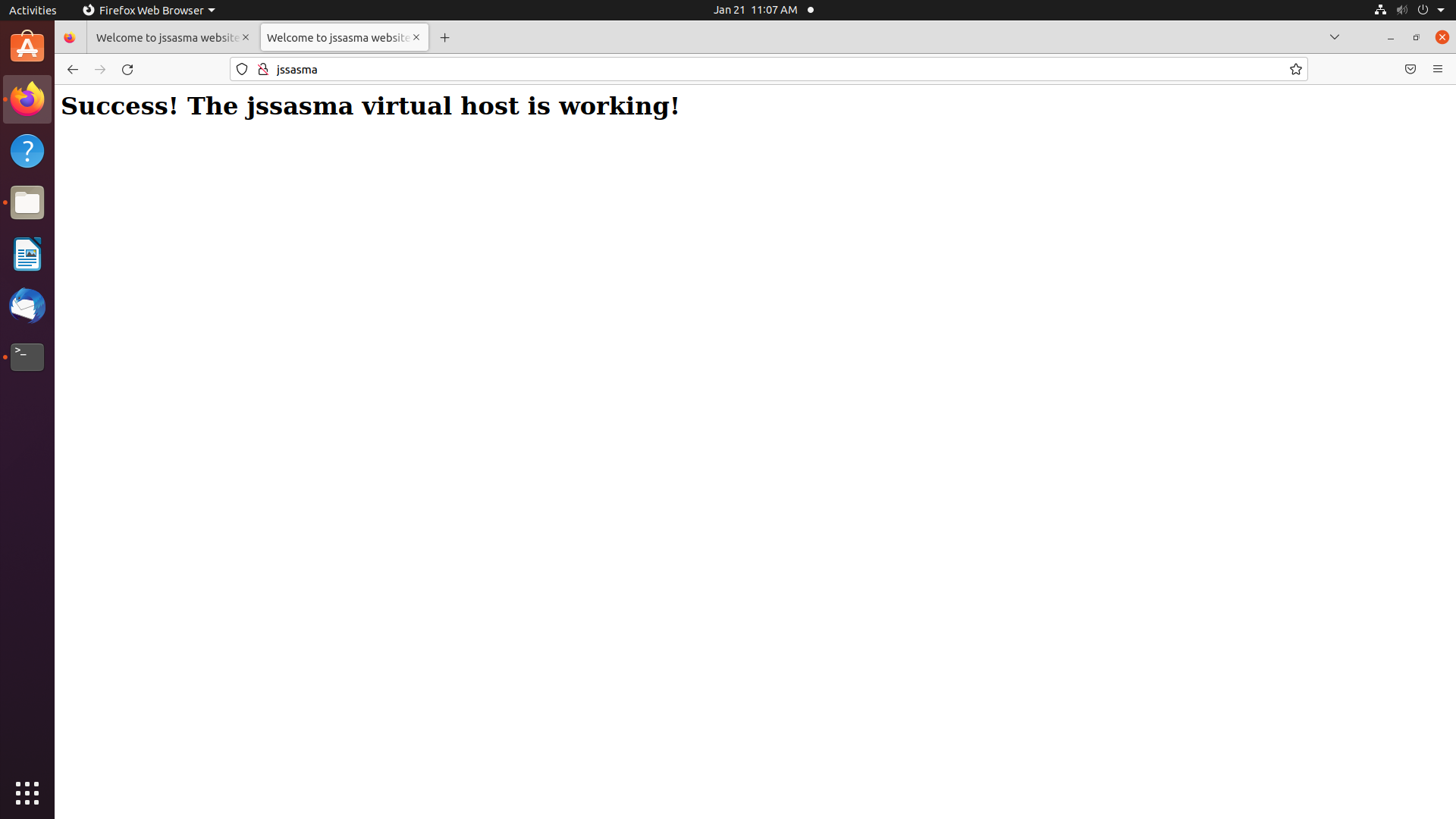
****

****

**Execution of Virtual Host using IP Address:**

****

**Execution of Virtual Host using Domain name “jssasma”:**

****