Writing the shell script to host the website

```
#!/bin/bash

# Define website directory and file path
WEB_DIR="/var/www/html"
INDEX_FILE="$WEB_DIR/index.html"

# Update package lists
echo "Updating package lists..."
sudo apt-get update -y

# Install Apache if not installed
echo "Installing Apache web server..."
sudo apt-get install apache2 -y

# Create an HTML file as the website's homepage
echo "Setting up the website files..."
sudo bash -c "cat > $INDEX_FILE <<EOF
<!DOCTYPE html>
<html lang='en'>
<head>
   <meta charset='UTF-8'>
   <meta name='viewport' content='width=device-width, initial-scale=1.0'>
   <title>My Website</title>
</head>
<body>
   <h1>Welcome to My Website!</h1>
   <p>This is a basic web page hosted with Apache on Linux.</p>
</body>
</html>
EOF"

# Change ownership of the web directory (optional)
echo "Adjusting permissions for $WEB_DIR..."
sudo chown -R www-data:www-data $WEB_DIR

# Start the Apache server
echo "Starting Apache server..."
sudo systemctl start apache2

# Enable Apache to start on boot
echo "Enabling Apache to start on boot..."
sudo systemctl enable apache2

# Allow HTTP traffic through the firewall (optional)
echo "Allowing HTTP traffic through the firewall..."
sudo ufw allow in "Apache"

# Display server status and IP address
echo "Apache server status:"
sudo systemctl status apache2 | grep Active

echo "Website hosted! Access it at http://$(hostname -I | awk '{print $1}')"
```

Explanation of the Script

1. **Update the Package Lists**: `sudo apt-get update` updates the package index to ensure we're installing the latest version of Apache.
2. **Install Apache**: `sudo apt-get install apache2 -y` installs the Apache web server.
3. **Set Up the HTML File**: This step creates an `index.html` file in Apache's root directory (`/var/www/html`). This HTML file will be the website's homepage.
4. **Change Permissions**: (Optional) This step changes ownership of the website directory to Apache's default user (`www-data`) to ensure it has full access.
5. **Start and Enable Apache**: The script starts the Apache server and enables it to start automatically on system boot.
6. **Allow HTTP Traffic**: Opens the firewall for HTTP traffic to allow external access to the website.
7. **Output the IP Address**: The last line outputs the server's IP address so you know where to access the site.

sudo bash -c

The command `sudo bash -c` is used to execute a command with elevated privileges (as the root user) within a new Bash shell. Here's a breakdown of what each part of the command does:

Breakdown

`sudo`:

1. `sudo` stands for "superuser do" and allows the execution of commands with superuser (root) privileges. This is necessary when performing administrative tasks, like installing software or modifying system files.

`bash`:

1. `bash` is the Bourne Again Shell, which is a command-line interpreter for executing shell commands.
2. When used in `sudo bash -c`, it opens a new shell to execute the command that follows the `-c` option.

`-c`:

1. The `-c` option tells Bash to execute the command that follows it as a string.
2. This allows you to run complex or multiple commands within a single string, often when you need elevated privileges for multiple commands or operations.

Command to be executed:

1. The command that follows the `-c` is the actual command or script to be executed within the new Bash shell with superuser privileges.

sudo chown -R www-data:www-data $WEB_DIR

The command `sudo chown -R www-data:www-data $WEB_DIR` is used to change the ownership of a directory and its contents. Here's a breakdown of each part of the command:

Breakdown

`sudo`:

1. `sudo` stands for "superuser do" and is used to run commands with elevated privileges (root user permissions). It's necessary here because changing file ownership is an administrative task.

`chown`:

1. `chown` stands for "change ownership". It's a command used to change the owner and group of a file or directory.

`-R` (Recursive option):

1. The `-R` option stands for "recursive". This means that the command will not only change the ownership of the specified directory (`$WEB_DIR`), but also all the files and subdirectories inside it.

`www-data:www-data`:

1. This specifies the new owner and group for the directory.
    1. `www-data` is the default user and group that Apache (and other web servers) run under.
    2. `www-data:www-data` means you're setting both the owner and the group to `www-data`. This ensures that Apache has the necessary permissions to read/write files in the directory.

`$WEB_DIR`:

1. This is a variable representing the path to the web directory, typically `/var/www/html`. It points to the directory where your website files are located.
2. `$WEB_DIR` is replaced with the actual path when the script runs, so the command will change the ownership of that directory.

$(hostname -I | awk '{print $1}')"

The command $(hostname -I | awk '{print $1}') is used in a shell script to retrieve the primary IP address of the machine.\

Breakdown

`hostname -I`:

1. The `hostname` command with the `-I` option outputs all IP addresses assigned to the machine's network interfaces, separated by spaces.
2. Example output could be something like: `192.168.1.10 10.0.0.5`
2.

`|` **(Pipe)**:

1. The pipe (`|`) sends the output of the `hostname -I` command to the `awk` command as input.

`awk '{print $1}'`:

1. `awk` is a powerful text-processing command. Here, it's used to select and print the first item (`$1`) in the input (the primary IP address).
2. In our example, it would output only `192.168.1.10`, the first IP address.

`$()` **(Command Substitution)**:

1. `$(...)` is command substitution, which means the output of the command inside `$(...)` is treated as a value. This value can then be used directly in the script.