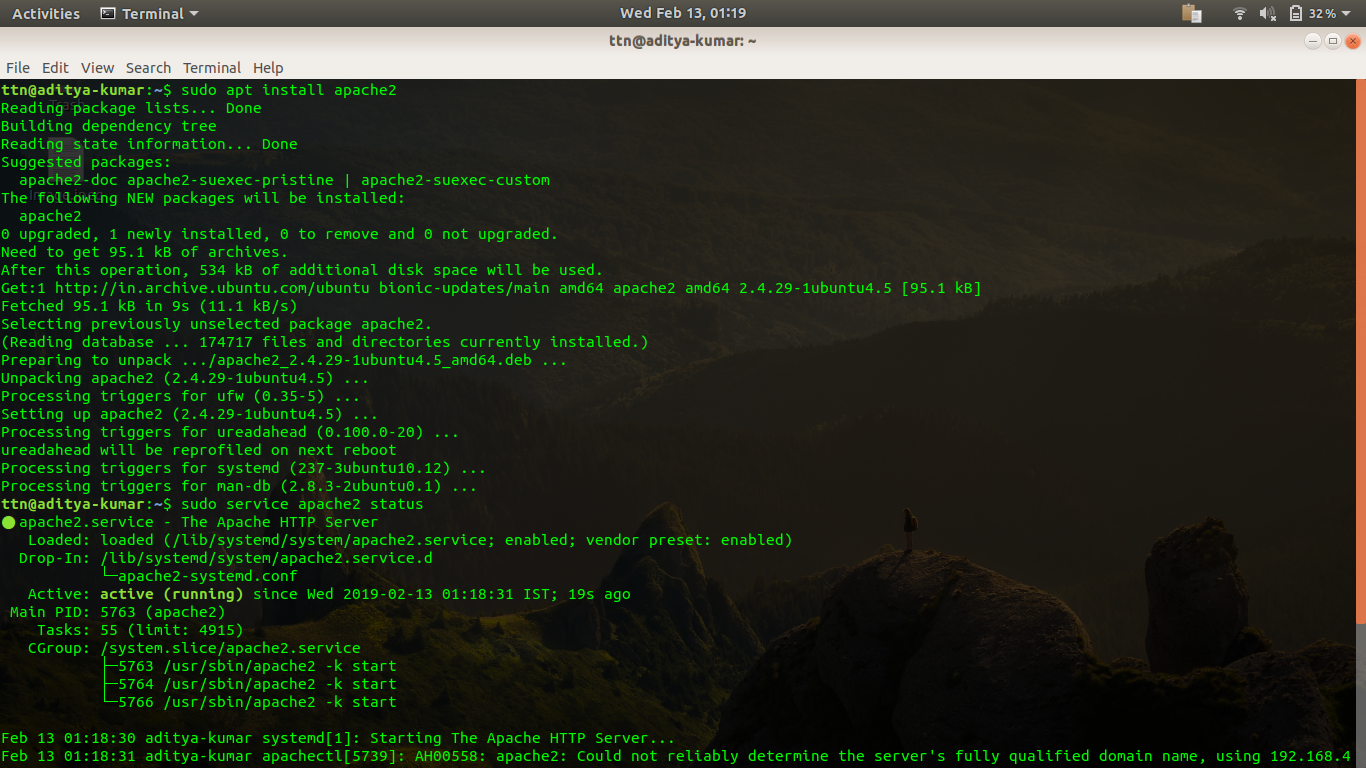
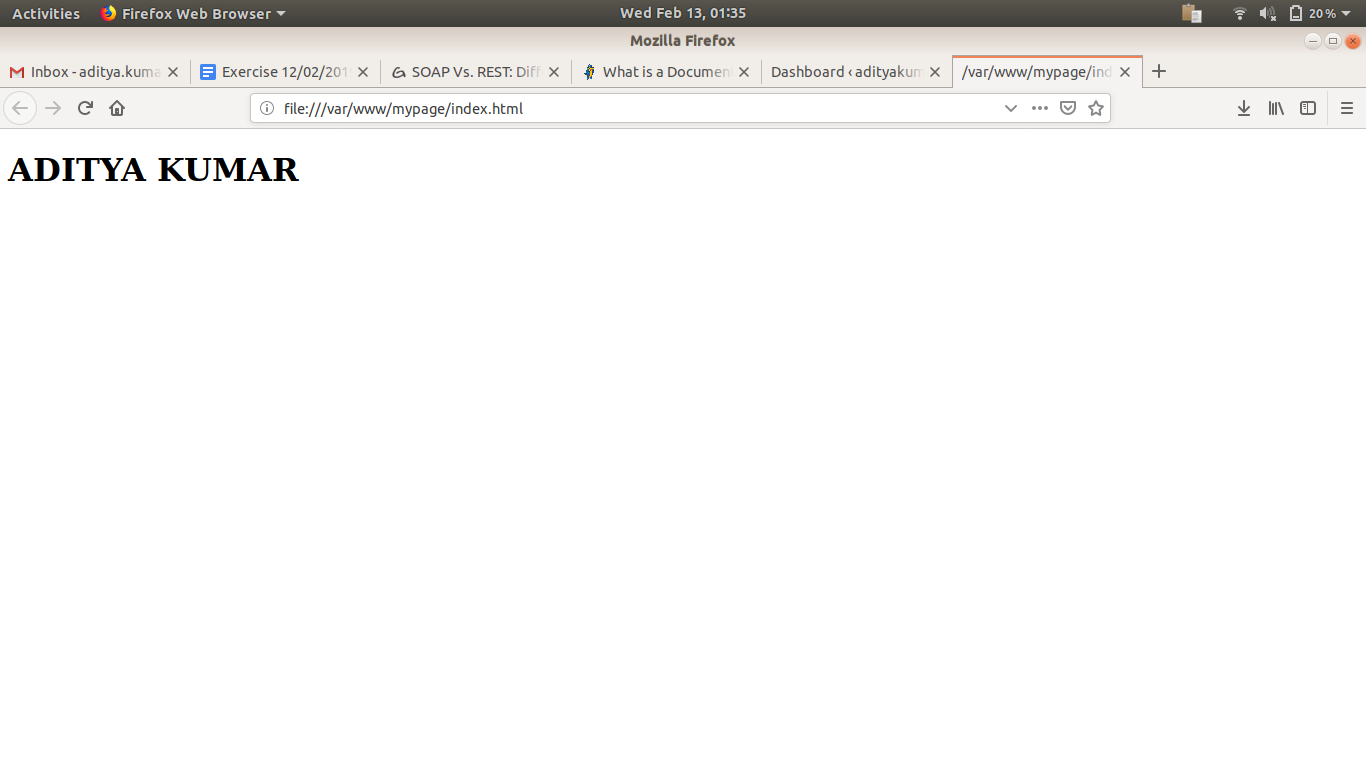


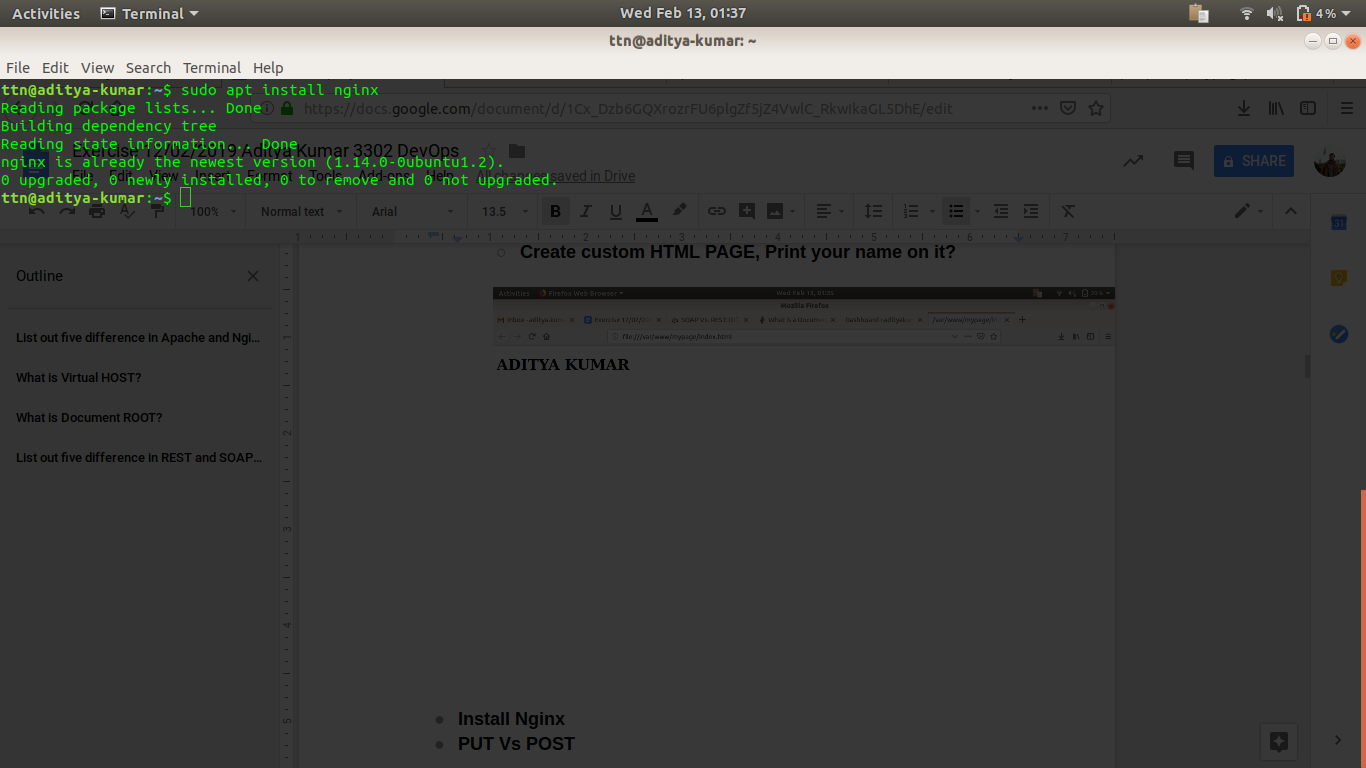
* **Install Apache**

****

* + **Create custom HTML PAGE, Print your name on it?**

****

* **Install Nginx**

****

* **PUT Vs POST**

|  |  |
| --- | --- |
| PUT | POST |
| RFC-2616 clearly mention that PUT method requests for the enclosed entity be stored under the supplied [Request-URI](https://restfulapi.net/resource-naming/). If the Request-URI refers to an already existing resource – an update operation will happen, otherwise create operation should happen if Request-URI is a valid resource URI (assuming client is allowed to determine resource identifier).  PUT /questions/{question-id} | The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. It essentially means that POSTrequest-URI should be of a collection URI.  POST /questions |
| PUT method is [idempotent](https://restfulapi.net/idempotent-rest-apis/). So if you send retry a request multiple times, that should be equivalent to single request modification. | POST is NOT idempotent. So if you retry the request N times, you will end up having N resources with N different URIs created on server. |
| Use PUT when you want to modify a singular resource which is already a part of resources collection. PUT replaces the resource in its entirety. Use PATCH if request updates part of the resource. | Use POST when you want to add a child resource under resources collection. |
| PUT is idempotent, so you can cache the response. | Responses to this method are not [cacheable](https://restfulapi.net/caching/), unless the response includes appropriate Cache-Control or Expires header fields. However, the 303 (See Other) response can be used to direct the user agent to retrieve a cacheable resource. |
| Generally, in practice, always use PUT for UPDATE operations. | Always use POST for CREATE operations. |

* **How Does SSL work?**

**Ans:**

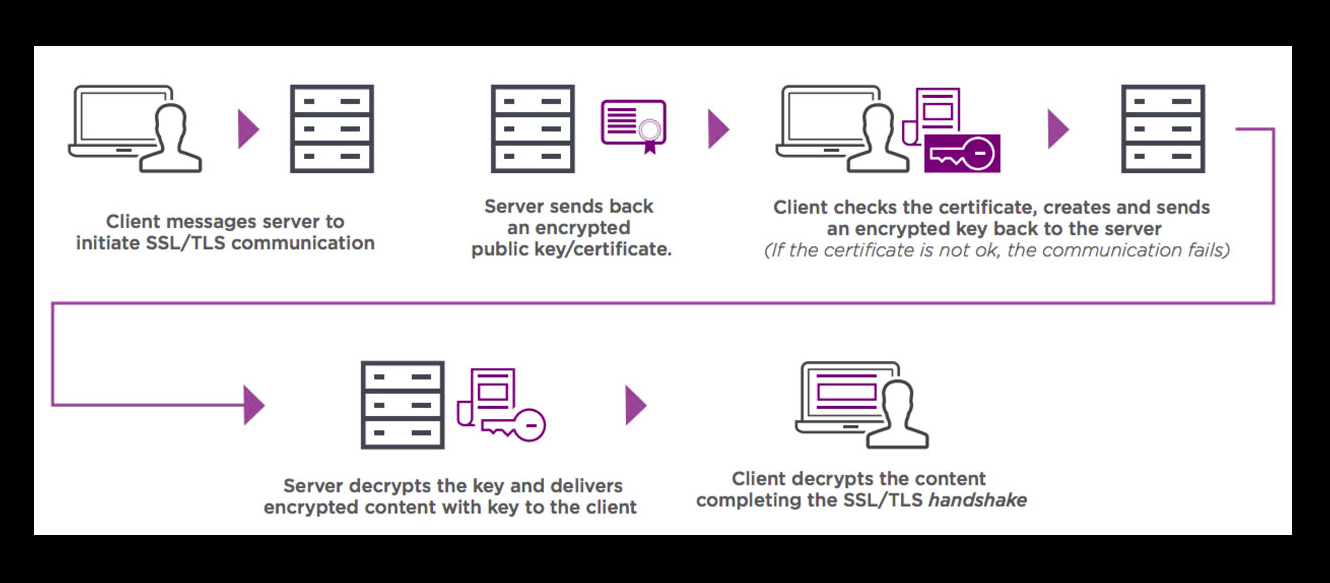
**A browser or server attempts to connect to a website (i.e. a web server) secured with SSL. The browser/server requests that the web server identify itself.**

**The web server sends the browser/server a copy of its SSL certificate.**

**The browser/server checks to see whether or not it trusts the SSL certificate. If so, it sends a message to the web server.**

**The web server sends back a digitally signed acknowledgement to start an SSL encrypted session.**

**Encrypted data is shared between the browser/server and the web server.**

****

* **List out five difference in Apache and Nginx**

|  |  |
| --- | --- |
| APACHE | NGINX |
| Apache follows multi-threaded approach to process client requests. | Nginx uses an event-driven approach to serve client requests. |
| It handles dynamic content within the web server itself. | It cannot process dynamic content natively. |
| It cannot process multiple requests concurrently with heavy web traffic. | It can process multiple client requests concurrently and efficiently with limited hardware resources. |
| Modules are dynamically loaded or unloaded making it more flexible. | The modules cannot be loaded dynamically. They must be compiled within the core software itself. |
| Apache is designed to be a web server. | Nginx is both a web server and a proxy server. |

* **What is Virtual HOST?**

Virtual hosting is a method for hosting multiple domain names on a single server. This allows one server to share its resources, such as memory and processor cycles, without requiring all services provided to use the same host name.

* **What is Document ROOT?**

The document root is the folder where the website files for a domain name are stored. Since cPanel allows for multiple domain names (addon domains and subdomains), you need to have a unique folder for each domain.

Your primary domain is rooted in the *public\_html* folder. Your addon and subdomain names will be rooted to their own folders inside *public\_html*.

* **List out five difference in REST and SOAP web service?**

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| **SOAP stands for Simple Object Access Protocol** | **REST stands for Representational State Transfer** |
| **SOAP is a protocol. SOAP was designed with a specification. It includes a WSDL file which has the required information on what the web service does in addition to the location of the web service.** | **REST is an Architectural style in which a web service can only be treated as a RESTful service if it follows the constraints of being**   * 1. **Client Server**   2. **Stateless**   3. **Cacheable**   4. **Layered System**   5. **Uniform Interface** |
| **SOAP cannot make use of REST since SOAP is a protocol and REST is an architectural pattern.** | **REST can make use of SOAP as the underlying protocol for web services, because in the end it is just an architectural pattern.** |
| **SOAP uses service interfaces to expose its functionality to client applications. In SOAP, the WSDL file provides the client with the necessary information which can be used to understand what services the web service can offer.** | **REST use Uniform Service locators to access to the components on the hardware device. For example, if there is an object which represents the data of an employee hosted on a URL as http://demo.guru99 , the below are some of URI that can exist to access them** |
| **EXAMPLE:-**  **<?xml version="1.0"?> <SOAP-ENV:Envelope  xmlns:SOAP-ENV ="http://www.w3.org/2001/12/soap-envelope"  SOAP-ENV:encodingStyle =" http://www.w3.org/2001/12/soap-encoding"> <soap:Body>  <Demo.guru99WebService  xmlns="http://tempuri.org/">  <EmployeeID>int</EmployeeID>  </Demo.guru99WebService>  </soap:Body> </SOAP-ENV:Envelope>** | **EXAMPLE:-**  **{"city":"Mumbai","state":"Maharastra"}** |

* **Explain all below mentioned HTTP status code in your words with examples?**
  + **HTTP STATUS CODE: 200, 300, 301, 302, 304, 307, 400, 401, 403, 404, 500, 503**

\***HTTP STATUS CODE 200** :-Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request, the response will contain an entity describing or containing the result of the action

\***HTTP STATUS CODE 300**:-Indicates multiple options for the resource from which the client may choose (via agent-driven content negotiation). For example, this code could be used to present multiple video format options, to list files with different filename extensions, or to suggest word-sense disambiguation

\***HTTP STATUS CODE 301**:-This and all future requests should be directed to the given URI

\***HTTP STATUS CODE 302**:- Tells the client to look at (browse to) another URL. 302 has been superseded by 303 and 307. This is an example of industry practice contradicting the standard. The HTTP/1.0 specification (RFC 1945) required the client to perform a temporary redirect (the original describing phrase was "Moved Temporarily"), but popular browsers implemented 302 with the functionality of a 303 See Other. Therefore, HTTP/1.1 added status codes 303 and 307 to distinguish between the two behaviours. However, some Web applications and frameworks use the 302 status code as if it were the 303.

\***HTTP STATUS CODE 304**:-Indicates that the resource has not been modified since the version specified by the request headers If-Modified-Since or If-None-Match. In such case, there is no need to retransmit the resource since the client still has a previously-downloaded copy

\***HTTP STATUS CODE 307**:-In this case, the request should be repeated with another URI; however, future requests should still use the original URI. In contrast to how 302 was historically implemented, the request method is not allowed to be changed when reissuing the original request. For example, a POST request should be repeated using another POST request

**\*HTTP STATUS CODE 400**:-The server cannot or will not process the request due to an apparent client error (e.g., malformed request syntax, size too large, invalid request message framing, or deceptive request routing)

\***HTTP STATUS CODE 401**:-Similar to *403 Forbidden*, but specifically for use when authentication is required and has failed or has not yet been provided. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource. See Basic access authentication and Digest access authentication.401 semantically means "unauthenticated",i.e. the user does not have the necessary credentials.

Note: Some sites incorrectly issue HTTP 401 when an IP address is banned from the website (usually the website domain) and that specific address is refused permission to access a website.

**\*HTTP STATUS CODE 403**:-The request was valid, but the server is refusing action. The user might not have the necessary permissions for a resource, or may need an account of some sort.

\***HTTP STATUS CODE 404**:-The requested resource could not be found but may be available in the future. Subsequent requests by the client are permissible.

**HTTP STATUS CODE 500**:-A generic error message, given when an unexpected condition was encountered and no more specific message is suitable.

**HTTP STATUS CODE** **503**:-The server is currently unavailable (because it is overloaded or down for maintenance). Generally, this is a temporary state