1. **Open the EC2 Console**:
   * Go to the [EC2 console](https://console.aws.amazon.com/ec2/).
2. **Edit Security Groups**:
   * Select the security group associated with your instance.
   * Ensure that:
     + **Port 80 (HTTP)** is allowed only from specific IP ranges (for initial setup).
     + **Port 443 (HTTPS)** is allowed from all IP addresses or specific ranges as needed.
     + **Port 22 (SSH)** is restricted to your IP address or range.

**2. Update Ansible Playbook for HTTPS and Self-Signed Certificates**

Here's an updated Ansible playbook that:

* Installs Apache HTTP Server.
* Configures SSL with a self-signed certificate.
* Redirects HTTP traffic to HTTPS.

1. **Generate a Self-Signed SSL Certificate** (this step can also be automated with Ansible but will be described here for completeness):

mkdir -p /ssl

openssl req -newkey rsa:2048 -nodes -keyout ~/ssl/server.key -x509 -days 365 -out ~/ssl/server.crt

This will create a self-signed certificate and a private key.

1. **Create Ansible Playbook** which we have uploaded in git webserver.yaml and run it using below command

ansible-playbook -i inventory.ini playbookname(webserver).yml

Verify Configuration:

1. **Check HTTP to HTTPS Redirection**:
   * Open a browser and navigate to http://domainname, Ensure it redirects to https://test-example.com
2. **Check HTTPS**:
   * Ensure that https://test-example.com displays the "Hello World!" page correctly and that the browser shows a warning about the self-signed certificate (expected behavior).
3. To ensure the correctness of the server configuration, including the correct setup of HTTPS redirection and SSL certificates, you can implement automated tests

2nd challenge:

Python code for validating credit card Numbers

**Explanation:**

1. **Regular Expression (pattern)**:

* ^[456]: Ensures the card number starts with 4, 5, or 6.
* \d{3}: Ensures the next 3 characters are digits.
* (-?\d{4}){3}$: Ensures the card has exactly 16 digits in total, either as 4 groups separated by hyphens or in one continuous sequence.

1. **Consecutive Repeating Digits Check**:

* card\_number\_clean = card\_number.replace("-", ""): Removes hyphens for checking consecutive repeated digits.
* re.search(r"(\d)\1{3,}", card\_number\_clean): This regex checks for any digit repeated 4 or more times consecutively.

**Input Example:**

4

4253625879615786

4424424424442444

5122-2368-7954-3214

42536258796157867

**Output Example:**

Copy code

Valid

Invalid

Valid

Invalid

This script validates each card number and prints either "Valid" or "Invalid" based on the given conditions.

**Second\_challenge:**

Data Type Handling:

Strings (S): Converted to strings, handling RFC3339 formatted dates.

Numbers (N): Converted to integers or floats, removing leading zeros.

Booleans (BOOL): Converted to true/false based on allowed values.

Nulls (NULL): Represented as null or omitted based on the value.

Lists (L): Transformed lists are sanitized and converted according to the supported data types.

Nested Map Support: Recursive transformation of maps within maps is handled.

Omissions: Empty keys, empty strings, and invalid fields are omitted.

number\_1: Converted to 1.5 as it’s a valid float number.

string\_1: Sanitized to "784498", removing trailing whitespace.

string\_2: Converted from RFC3339 date "2014-07-16T20:55:46Z" to Unix Epoch time 1405544146.

map\_1:

list\_1: The list contains:

"011": Converted to 11 by stripping the leading zero.

"f": Converted to false.

null\_1: Transformed to null.

Other invalid or empty fields like list\_2, list\_3, and empty keys are omitted as per the rules.