## 

**SED Challenge**

**Infrastructure**

For this project, please think about how you would architect a scalable and secure static web application in AWS.

* Create and deploy a running instance of a web server using a configuration management tool of your choice. The web server should serve one page with the following content.

<html>

<head>

<title>Hello World</title>

</head>

<body>

<h1>Hello World!</h1>

</body>

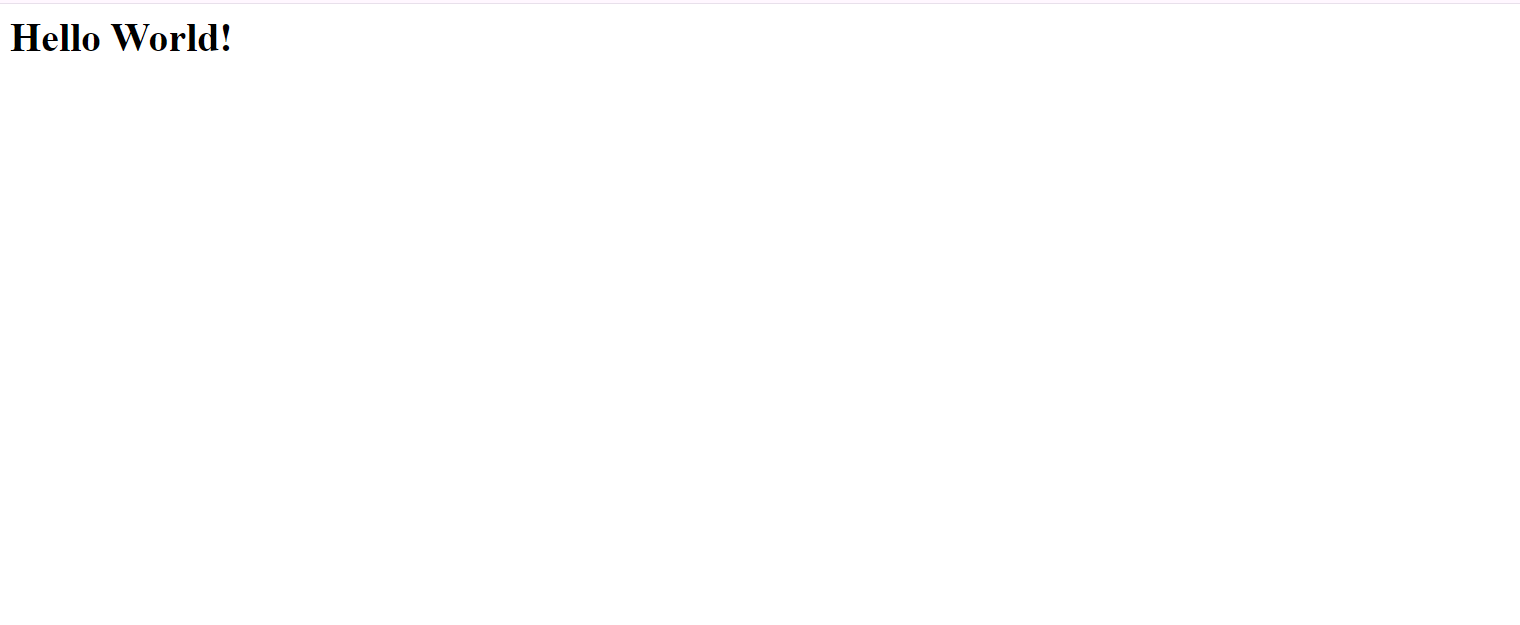
</html>

* Secure this application and host such that only appropriate ports are publicly exposed and any http requests are redirected to https. This should be automated using a configuration management tool of your choice and you should feel free to use a self-signed certificate for the web server.
* Develop and apply automated tests to validate the correctness of the server configuration.
* Express everything in code.
* Provide your code in a Git repository named <FIRSTNAME>\_Challenge on GitHub.com

Be prepared to walk though your code, discuss your thought process, and talk through how you might monitor and scale this application. You should also be able to demo a running instance of the host.  
  
  
  
**Solution:**  
  
  
  
  
  
To address the requirements, I would use AWS services such as EC2 (Elastic Compute Cloud) for hosting the web server, Route 53 for DNS management, and ACM (AWS Certificate Manager) for SSL certificates. I will use Ansible as the configuration management tool to automate the setup and deployment process.  
  
  
  
  
  
  
Here's a general outline of the steps involved:

* **1.Set up an EC2 instance:** Provision an EC2 instance with Amazon Linux 2 or any other suitable Linux distribution.
* **2.Install and configure the web server:** Install Nginx or Apache web server and configure it to serve the static content provided.
* **3.Obtain SSL certificate:** Generate a self-signed SSL certificate or obtain a free certificate from ACM.
* **4.Configure HTTPS redirection:** Set up Nginx or Apache to redirect HTTP traffic to HTTPS.
* **5.Automate the setup with Ansible:** Write Ansible playbooks to automate the provisioning and configuration of the EC2 instance, web server, SSL certificate, and HTTPS redirection.
* **6.Write automated tests:** Develop automated tests using tools like Ansible's ansible-lint for linting Ansible code and tools like serverspec or pytest for testing the deployed infrastructure.
* **7.Deploy the application:** Use Ansible to deploy the application to the EC2 instance.
* **8.Monitor and scale the application:** Set up monitoring using AWS CloudWatch for monitoring resource utilization and application performance. Use AWS Auto Scaling to automatically adjust the number of EC2 instances based on demand.  
    
    
    
    
  **. Install Ansible:**
* You can install Ansible using pip, the Python package manager. Run the following command in your terminal:
* pip install ansible
* **2. Configure AWS Credentials:**
* You need to configure your AWS credentials on your local machine. You can do this by setting environment variables or by creating an AWS credentials file.

#### Using Environment Variables:

* Set the following environment variables with your AWS Access Key ID and Secret Access Key:
* export AWS\_ACCESS\_KEY\_ID=your\_access\_key\_id
* export AWS\_SECRET\_ACCESS\_KEY=your\_secret\_access\_key
* Using AWS Credentials File:
* Create a file named ~/.aws/credentials on your local machine and add your AWS Access Key ID and Secret Access Key:
* [default]
* aws\_access\_key\_id = your\_access\_key\_id
* aws\_secret\_access\_key = your\_secret\_access\_key
* **4. Modify Variables:**
* In your Ansible playbooks and roles, you'll have variables such as ec2\_key\_name, ec2\_instance\_type, ec2\_image, ec2\_region, and ec2\_subnet\_id. Modify these variables according to your AWS setup.
* **5. Create Your Playbooks:**
* Write your Ansible playbooks according to your requirements. You may have a main playbook (main.yml) that orchestrates the execution of other playbooks and tasks.
* **6. Run Your Playbooks:**
* Execute your Ansible playbooks using the ansible-playbook command:
* ansible-playbook playbooks/main.yml  
    
    
  

**Coding**

Please solve the below problem in python or go. You don't need to do the submission on the web site. Include the program in the repo above.

1. <https://www.hackerrank.com/challenges/validating-credit-card-number/problem>

**Code Solution:**  
  
  
  
def ok(s):

import re

if not s[0] in "456": return False

if not re.match("[0-9-]+", s): return False

if sum([1 if "0" <= \_ and \_ <= "9" else 0 for \_ in s]) != 16: return False

if not (len(s) == 16 or len(s) == 19 and s[4] == "-" and s[9] == "-" and s[14] == "-"): return False

s = s.replace("-", "")

for i in range(len(s)-3):

if s[i] == s[i+1] and s[i] == s[i+2] and s[i] == s[i+3]: return False

return True

import sys

stdin = sys.stdin

t = int(stdin.readline())

import re

for z in range(t):

line = stdin.readline().rstrip()

if(ok(line)):

print("Valid")

else:

print("Invalid")  
  
  
  
