Scenario

Your company is creating an Instagram clone called Udagram. Developers pushed the latest version of their code in a zip file located in a public S3 Bucket.
You have been tasked with automated deploying the application, along with the necessary supporting software into its matching infrastructure.
This needs to be done in an fashion so that the infrastructure can be discarded as soon as the testing team finishes their tests and gathers their results.

Server specs

You'll need to create a **Launch Configuration** for your application servers in order to deploy **four servers**, **two located in each of your private subnets**. The launch configuration will be used by an **auto-scaling group**.
You'll need **two vCPUs and at least 4GB of RAM**. The Operating System to be used is **Ubuntu 18**. So, choose an Instance size and Machine Image (AMI) that best fits this spec.
Be sure to **allocate at least 10GB of disk space** so that you don't run into issues.

Security Groups and Roles

Since you will be downloading the application archive from an **S3 Bucket**, you'll need to create an **IAM Role** that allows your instances to use the S3 Service.
Udagram communicates on the default `HTTP Port: 80`, so your servers will need this inbound port open since you will use it with the **Load Balancer** and the **Load Balancer Health Check**. As for outbound, the servers will need unrestricted internet access to be able to download and update their software.
The load balancer should allow all public traffic `(0.0.0.0/0)` on `port 80` inbound, which is the default `HTTP port`. Outbound, it will only be using `port 80` to reach the internal servers.
The application needs to be deployed into private subnets with a **Load Balancer** located in a public subnet.
One of the output exports of the **CloudFormation** script should be the public URL of the **LoadBalancer**. **Bonus points** if you add `http://` in front of the load balancer **DNS Name** in the output, for convenience.

Other Considerations

You can deploy your servers with an **SSH Key** into Public subnets while you are creating the script. This helps with troubleshooting. Once done, move them to your private subnets and remove the **SSH Key** from your **Launch Configuration**.
It also helps to test directly, without the load balancer. Once you are confident that your server is behaving correctly, increase the instance count and add the load balancer to your script.

While your instances are in public subnets, you'll also need the `SSH port open (port 22)` for your access, in case you need to troubleshoot your instances.
Log information for UserData scripts is located in this file: `cloud-init-output.log` under the folder: `/var/log`.
You should be able to destroy the entire infrastructure and build it back up without any manual steps required, other than running the **CloudFormation** script.
The provided UserData script should help you install all the required dependencies. Bear in mind that this process takes several minutes to complete. Also, the application takes a few seconds to load. This information is crucial for the settings of your load balancer health check.
It's up to you to decide which values should be parameters and which you will hard-code in your script. See the provided supporting code for help and more clues.
If you want to go the extra mile, set up a bastion host (jump box) to allow you to SSH into your private subnet servers. This bastion host would be on a Public Subnet with `port 22` open only to your home `IP address`, and it would need to have the private key that you use to access the other servers.

Rubric checks

Basics	
Parameters	Using 11
Resources	• Load Balancer: ✓ • Launch Configuration with health check: ✓ • Auto scaling: ✓ • Security groups: ✓ • Listener: ✓ • Target group with health check: ✓
Outputs	• Amount: 12 • Load balancer URL with http: ✓
Working Test	http://final-webap-11j5ikbjgj75u-889281164.us-east-2.elb.amazonaws.com/
Load Balancer	
Target Group	• *auto-scaling group needs to have a property that associates it with a target group*: **It does, Line 353, "!Ref WebAppTargetGroup"** • *The Load Balancer will have a Listener rule associated with the same target group*: **It does, Line 336, "TargetGroupArn: !Ref 'WebAppTargetGroup'"**
Health Check and Listener	• *Port 80 should be used in Security groups, health checks and listeners associated with the load balancer*: ✓
Auto-Scaling	
Subnets	• *Students should be using PRIV-NET (private subnets) for their auto-scaling instances*: ✓
Machine Specs	• *The machine should have 10 GB or more of disk and should be a t3.small or better*: ✓

SSH Key	• *There shouldn't be a 'keyname' property in the launch config*: ✓ (Howewer there is an ssh key assigned to the optional bastion host I launched)
Bonus	
Outputs	• Any values in the output section are a bonus: ✓
Bastion Host	• Any resource of type AWS::EC2::Instance, optional, but nice to have: ✓ Resource called BastionHost