Maria Pouliou

09/05/2020

Challenge 1

For the 3 Tier environment, I have created 3 Tiers using Security Groups.

1) WebTier_SG that accepts web traffic from anywhere on the Internet. Ports 80 and 443. Outbound to ApplicationTier_SG all traffic.

2)ApplicationTier_SG that accepts traffic from WebTier_SG (Protocol TCP and UDP Ports: 0 - 65535

and ICMP Ports: All). Outbound to DatabaseTier_SG all traffic.

3)DatabaseTier_SG that accepts traffic from ApplicationTier_SG (Protocol TCP, UDP Ports: 0 - 65535 and ICML Ports: All)

There has also been a rule for all Tiers to allow administration remotely from my network over SSH port 22.

Then, an instance has been created that accepts as security group the WebTier_SG.

Public DNS (IPv4): ec2-18-132-52-95.eu-west-2.compute.amazonaws.com

IPv4 Public IP: 18.132.52.95

All the Security Groups:

Security Groups (4) Info

Actions | Create security group

Filter security groups

1

Security group name	VPC ID	Description
default	vpc-1f9ae277	default VPC security group
ApplicationTier_SG	vpc-1f9ae277	Allows access from the WebTier_SG and your IP
DatabaseTier_SQ	vpc-1f9ae277	Allows access from ApplicationTier_SQ and your IP
WebTier_SQ	vpc-1f9ae277	Accepts Web Traffic from anywhere on the Internet

WebTier_SG:

Security group name	Security group ID	Description	VPC ID
WebTier_SQ	sg-008ac4772438e7303	Accepts Web Traffic from anywhere on the Internet	vpc-1f9ae277

Owner	Inbound rules count	Outbound rules count
431612133416	6 Permission entries	1 Permission entry

Inbound rules | Outbound rules | Tags

Outbound rules

Edit outbound rules

Type	Protocol	Port range	Destination	Description - optional
Custom TCP	TCP	0	sg-0fa9089c72bae7f53 (ApplicationTier_SG)	-

Inbound rules | Outbound rules | Tags

Inbound rules

Edit inbound rules

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	0.0.0.0/0	Allow inbound HTTP access
HTTP	TCP	80	::/0	Allow inbound HTTP access
SSH	TCP	22	0.0.0.0/0	Allow inbound SSH access to Linux instances from your network (over the Internet gateway).
SSH	TCP	22	::/0	Allow inbound SSH access to Linux instances from your network (over the Internet gateway).
HTTPS	TCP	443	0.0.0.0/0	Allows HTTPS access
HTTPS	TCP	443	::/0	Allows HTTPS access

ApplicationTier_SG:

Security group name	Security group ID	Description	VPC ID
ApplicationTier_SG	sg-0fa9089c72bae7f53	Allows access from the WebTier_SG and your IP	vpc-1f9ae277
Owner	**Inbound rules count**	**Outbound rules count**	
431612133416	4 Permission entries	1 Permission entry	

Inbound rules | Outbound rules | Tags

Outbound rules

Edit outbound rules

Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	sg-08a3f11fbf586aa84 (DatabaseTier_SQ)	-

Inbound rules | Outbound rules | Tags

Inbound rules

Edit inbound rules

Type	Protocol	Port range	Source	Description - optional
All TCP	TCP	0 - 65535	sg-008ac4772438e7303 (WebTier_SQ)	TCP inbound from the WebTier_SecurityGroup
SSH	TCP	22	158.180.192.10/32	Allow inbound SSH access to Linux instances from your network (over the Internet gateway).
All UDP	UDP	0 - 65535	sg-008ac4772438e7303 (WebTier_SQ)	UDP inbound access from the WebTier_SecurityGroup
All ICMP - IPv4	ICMP	All	sg-008ac4772438e7303 (WebTier_SQ)	ICMP. An error-reporting protocol when network problems prevent delivery of IP packets

DatabaseTier_SG:

Security group name	Security group ID	Description	VPC ID
DatabaseTier_SQ	sg-08a3f11fbf586aa84	Allows access from ApplicationTier_SQ and your IP	vpc-1f9ae277

Owner	Inbound rules count	Outbound rules count
431612133416	5 Permission entries	0 Permission entries

Inbound rules | Outbound rules | Tags

Outbound rules

Edit outbound rules

Type	Protocol	Port range	Destination	Description - optional
No rules found				

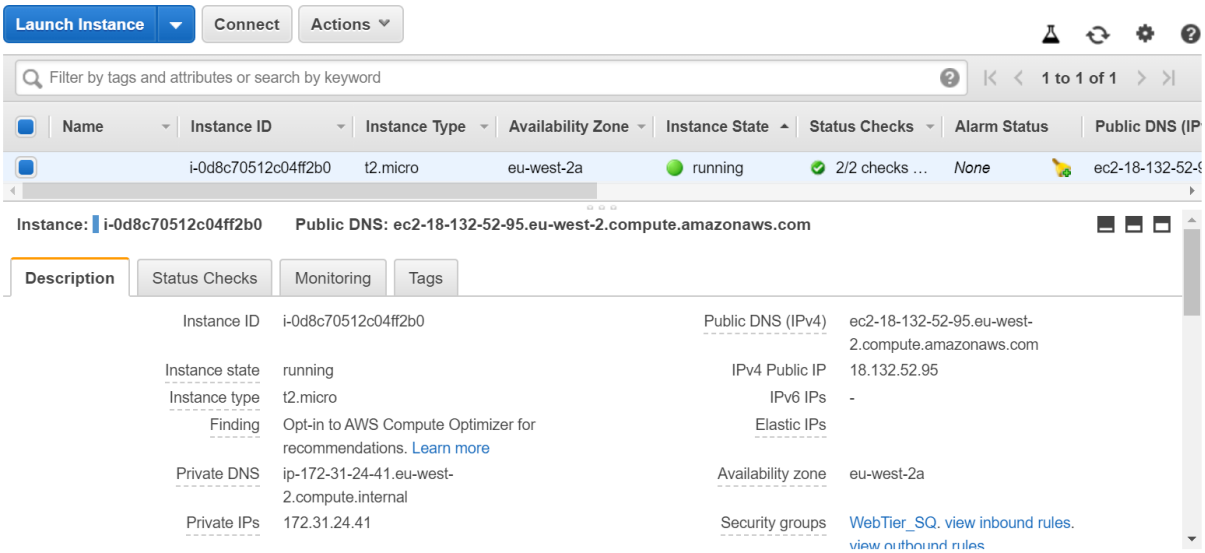
Inbound rules | Outbound rules | Tags

Inbound rules

Edit inbound rules

Type	Protocol	Port range	Source	Description - optional
All TCP	TCP	0 - 65535	sg-0fa9089c72bae7f53 (ApplicationTier_SG)	Allows access from the ApplicationTier_SecurityGroup
SSH	TCP	22	158.180.192.10/32	Allows access from your IP address
Custom ICMP - IPv6	IPv6 ICMP	All	sg-0fa9089c72bae7f53 (ApplicationTier_SG)	-
All UDP	UDP	0 - 65535	sg-0fa9089c72bae7f53 (ApplicationTier_SG)	Allows access from the ApplicationTier_SecurityGroup
All ICMP - IPv4	ICMP	All	sg-0fa9089c72bae7f53 (ApplicationTier_SG)	Allows access from the ApplicationTier_SecurityGroup

The EC2 Instance:



Challenge 2

Having been connected to the AWS EC2 instance and running the following command, we get all the possible meta-data keys.

ec2-user@ip-172-31-24-41:~

```
[ec2-user@ip-172-31-24-41 ~]$ curl http://169.254.169.254/latest/meta-data/
ami-id
ami-launch-index
ami-manifest-path
block-device-mapping/
events/
hibernation/
hostname
identity-credentials/
instance-action
instance-id
instance-type
local-hostname
local-ipv4
mac
metrics/
network/
placement/
profile
public-hostname
public-ipv4
public-keys/
reservation-id
security-groups
services/[ec2-user@ip-172-31-24-41 ~]$
```

I have created a function in python that individually queries the meta-data of the instance and provides a json formatted output.

```python
import requests
import json

#code should run within AWS EC2 instance

def metadata_Query(metadata_Name):
    try:
        url = "http://169.254.169.254/latest/meta-data/"+metadata_Name #extract metadata value
        response = requests.post(url)
        json_Response= json.dumps({metadata_Name: response.text}) #parse response to JSON format
        print(json_Response)
    except:
        print("Please check again metadata key provided") #in case metadata key is not valid
```

Tests:

```
>>> metadata_Query("public-ipv4")
{"public-ipv4": "18.132.52.95"}
>>>
>>>
>>> metadata_Query("ami-id")
{"ami-id": "ami-01a6e31ac994bbc09"}
>>>
>>>
>>> metadata_Query("instance-id")
{"instance-id": "i-0d8c70512c04ff2b0"}
>>>
```

Challenge 3:

This piece of python code, get a json object and a key, and returns the value. It also catches the invalid json or key input.

```
import json

def extract_Value(json_Object, key):
    try:
        json_Load = json.loads(json_Object) #loads the json Object
        value = json_Load[key] #extracts the value
        print(key+" = "+str(value))
    except:
    #in case json object is not valid or key is not found in the json object
        print("Please check again json object and key provided.")
```

Tests:

```
>>> extract_Value('{"name":"George", "age": 37, "job":"developer"}', "age")
age = 37
>>>
>>>
>>> extract_Value('{"name":"Chris", "age": 31, "job":"teacher", "city": "Leeds"}', "
job")
job = teacher
>>>
>>>
>>>
>>> extract_Value('{"name":"Chris", "age": 31, "job":"teacher", "city": "Leeds"}', "
address")
Please check again json object and key provided.
>>>
```

Thanks for your time and consideration!