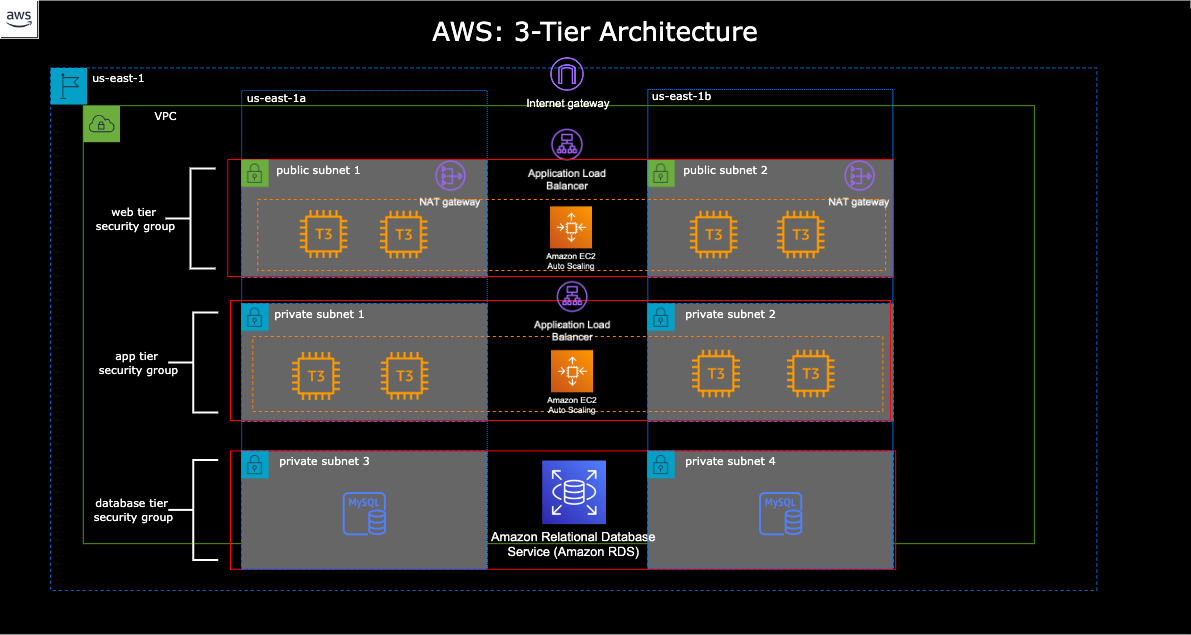
**Challenge #1**

A 3 tier environment is a common setup. Use a tool of your choosing/familiarity create these resources. Please remember we will not be judged on the outcome but more focusing on the approach, style and reproducibility.

**Solution Approach:**

1. Assumption:

Assuming the below diagram for the sample 3-tier architecture in AWS :



For implementing the scenario I used the below tools :

1)Terraform

2) AWS CLI

3)AWS Account

4)Any IDE( Used VS)

**Solution :**

Kindly find the code repo at git hub link : <https://github.com/pattanaiksrimani/Challanges_UK/tree/main/Challange-1> ,Kindly go through the read me for the execution details

**Explanation:**

The terraform code will create the below resources in aws :

* Terraform will create below resources

1. **VPC**
2. **Application Load Balancer**
3. **Public & Private Subnets**
4. **EC2 Instances**
5. **Route Table**
6. **Internet Gateway**
7. **RDS Instance**
8. **Security Groups for Web & RDS instances**

And used variables and count in order to re-use the code as its not static the values we need to define in the variables.tf

**Testing and Proof of the execution:**

1)After your infrastructure has been created there should be an Output displayed on your terminal for the Application Load Balancer DNS Name.

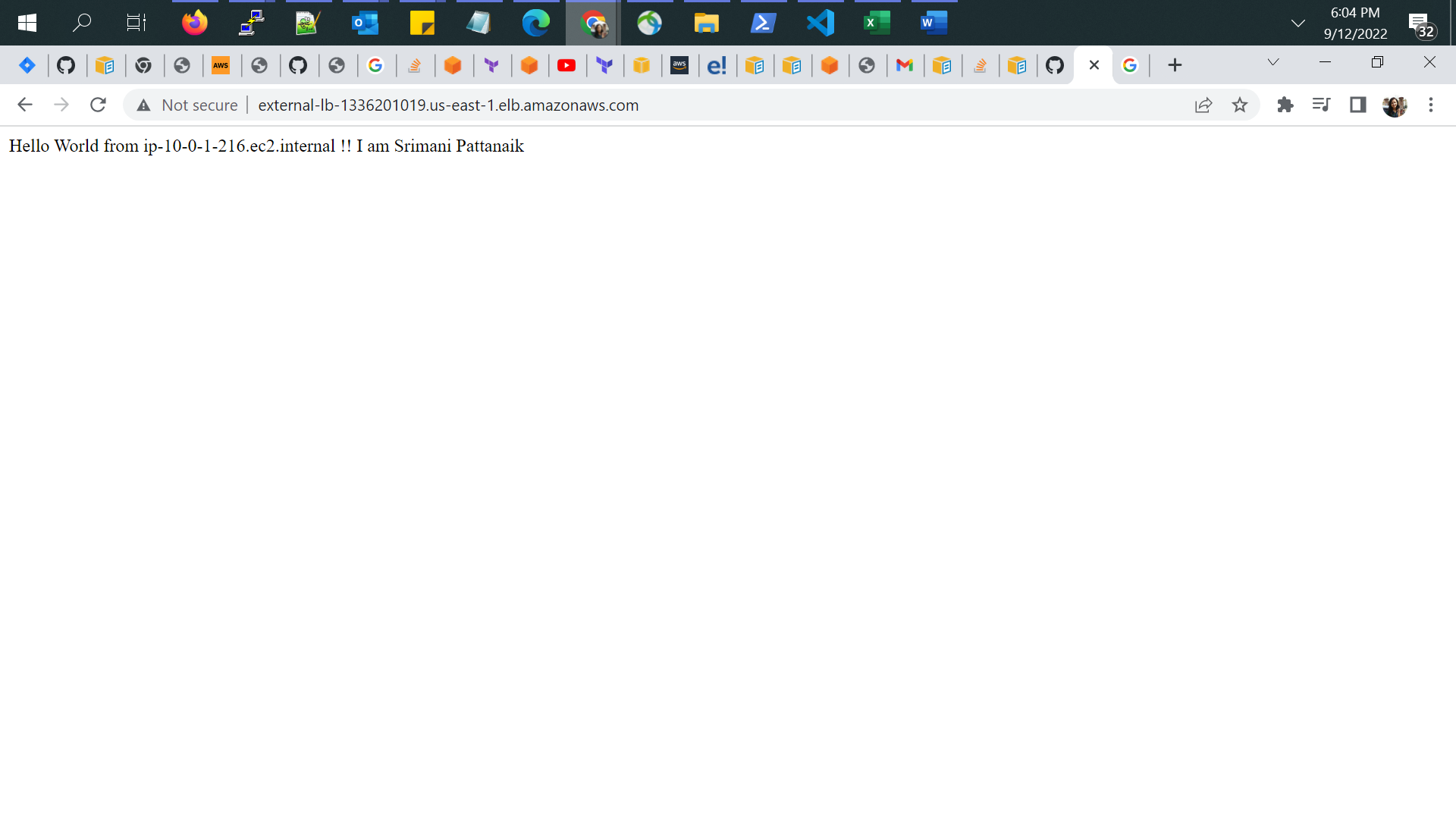
output "lb\_dns\_name" {

description = "The DNS name of the load balancer"

value = aws\_lb.external-elb.dns\_name

}

2)Copy and paste (without quotations) into a new browser tab. Refresh the page to see the load balancer switch between the two instances. For me I got the below screen



Reference Used :

<https://registry.terraform.io/providers/hashicorp/aws/latest/docs>

**Challenge #2**

**Summary**

We need to write code that will query the meta data of an instance within aws and provide a json formatted output. The choice of language and implementation is up to you.

**Bonus Points**

The code allows for a particular data key to be retrieved individually

**Hints**

·       [Aws Documentation](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html)

·       [Azure Documentation](https://docs.microsoft.com/en-us/azure/virtual-machines/windows/instance-metadata-service)

·       [Google Documentation](https://cloud.google.com/compute/docs/storing-retrieving-metadata)

**Solution Approach:**

* Pre-Requisite for the solution :
* Create EC2 Instance (<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html> )
* Install Python and git
* Install pipenv
* Install project dependencies using pipenv install
* SSH to the ECS instance use the link for details :

<https://www.youtube.com/watch?v=jv-dgOfFN4o>

* Once you are connected to the EC2 instance copy the script to EC2 instance from git :

<https://github.com/pattanaiksrimani/Challanges_UK/blob/main/Challange-2.py>

* Copy the script to EC2 instance and run it :

Sample OutPut :

[ec2-user@ip-172-31-22-154 aws\_metadata\_in\_JSON]$ ls -latr

total 12

-rw-rw-r--. 1 ec2-user ec2-user 1913 Sep 12 20:30 Pipfile.lock

-rw-rw-r--. 1 ec2-user ec2-user 153 Sep 12 20:30 Pipfile

drwx------. 6 ec2-user ec2-user 173 Sep 13 09:27 ..

-rw-rw-r--. 1 ec2-user ec2-user 1652 Sep 13 12:49 metadata\_to\_json.py

drwxrwxr-x. 2 ec2-user ec2-user 68 Sep 13 12:55 .

[ec2-user@ip-172-31-22-154 aws\_metadata\_in\_JSON]$ python3 metadata\_to\_json.py

{

"meta-data": {

"ami-id": "ami-06640050dc3f556bb",

"ami-launch-index": 0,

"ami-manifest-path": "(unknown)",

"block-device-mapping": {

"ami": "/dev/sda1",

"root": "/dev/sda1"

},

"events": {

"maintenance": {

"history": [],

"scheduled": []

}

},

"hostname": "ip-172-31-22-154.ec2.internal",

"identity-credentials": {

"ec2": {

"info": {

"AccountId": "104529772138",

"Code": "Success",

"LastUpdated": "2022-09-13T12:43:58Z"

},

"security-credentials": {

"ec2-instance": {

"AccessKeyId": "ASIARQVTVXZVA3GMML4C",

"Code": "Success",

"Expiration": "2022-09-13T19:10:41Z",

"LastUpdated": "2022-09-13T12:43:21Z",

"SecretAccessKey": "6wmNOepLKEp4OLrM25cExSC3liw97jRTntySiR4P",

"Token": "",

"Type": "AWS-HMAC"

}

}

}

},

"instance-action": "none",

"instance-id": "i-0928e6d2c01fc8862",

"instance-life-cycle": "on-demand",

"instance-type": "t2.micro",

"local-hostname": "ip-172-31-22-154.ec2.internal",

"local-ipv4": "172.31.22.154",

"mac": "0a:94:46:7e:22:4f",

"metrics": {

"vhostmd": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>"

},

"network": {

"interfaces": {

"macs": {

"0a:94:46:7e:22:4f": {

"device-number": 0,

"interface-id": "eni-0b03a02dd31c153c5",

"ipv4-associations": {

"18.209.51.214": "172.31.22.154"

},

"local-hostname": "ip-172-31-22-154.ec2.internal",

"local-ipv4s": "172.31.22.154",

"mac": "0a:94:46:7e:22:4f",

"owner-id": 104529772138,

"public-hostname": "ec2-18-209-51-214.compute-1.amazonaws.com",

"public-ipv4s": "18.209.51.214",

"security-group-ids": "sg-0acba04aed3c4b5de",

"security-groups": "launch-wizard-1",

"subnet-id": "subnet-b58485f8",

"subnet-ipv4-cidr-block": "172.31.16.0/20",

"vpc-id": "vpc-3a62ef47",

"vpc-ipv4-cidr-block": "172.31.0.0/16",

"vpc-ipv4-cidr-blocks": "172.31.0.0/16"

}

}

}

},

"placement": {

"availability-zone": "us-east-1a",

"availability-zone-id": "use1-az4",

"region": "us-east-1"

},

"profile": "default-hvm",

"public-hostname": "ec2-18-209-51-214.compute-1.amazonaws.com",

"public-ipv4": "18.209.51.214",

"public-keys": {

"0=myserver07": "<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>\n<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Transitional//EN\"\n\t\"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">\n<html xmlns=\"http://www.w3.org/1999/xhtml\" xml:lang=\"en\" lang=\"en\">\n <head>\n <title>404 - Not Found</title>\n </head>\n <body>\n <h1>404 - Not Found</h1>\n </body>\n</html>\n"

},

"reservation-id": "r-053175f06eefaa07b",

"security-groups": "launch-wizard-1",

"services": {

"domain": "amazonaws.com",

"partition": "aws"

}

}

}

What key details you like to view ?

local-ipv4s

['172.31.22.154']

[ec2-user@ip-172-31-22-154 aws\_metadata\_in\_JSON]$

3) **Challenge #3**

We have a nested object, we would like a function that you pass in the object and a key and get back the value. How this is implemented is up to you.

**Example Inputs**

object = {“a”:{“b”:{“c”:”d”}}}

key = a/b/c

object = {“x”:{“y”:{“z”:”a”}}}

key = x/y/z

value = a

**Hints**

We would like to see some tests. [A quick read to help you along the way](https://urldefense.com/v3/__https:/hexdocs.pm/elixir/master/Kernel.html*get_in/2__;Iw!!E1R1dd1bLLODlQ4!EhFkqaf_KPFdfZoStIEbUsQEnou6XZVB43WdxA9qawLdOCI7O9kqfEhYnQG2MCaBV20YzAXTgGJ37W-YajkJT76jQ_6LOL4$)

We would expect it in any other language apart from elixir.

**Solution Approach:**

Used python to code this

Used links : <https://stackoverflow.com/questions/9807634/find-all-occurrences-of-a-key-in-nested-dictionaries-and-lists>

<https://stackoverflow.com/questions/1305532/how-to-convert-a-nested-python-dict-to-object>

The example is :

>>> d = {'a': 1, 'b': {'c': 2}, 'd': ["hi", {'foo': "bar"}]}

Should be accessible in this way:

>>> x = dict2obj(d)

>>> x.a

1

>>> x.b.c

2

>>> x.d[1].foo

bar

This is a nested python dict to object .

Uploaded the code to <https://github.com/pattanaiksrimani/Challanges_UK/blob/main/Challange-3.py>

Testing :

Copy the code to any instance where python is installed :

**Output/testing of the script :**

[ec2-user@ip-172-31-22-154 ~]$ cat challange3.py

##

# Example Inputs

# object = {“a”:{“b”:{“c”:”d”}}}

# key = a/b/c

##

# object = {“x”:{“y”:{“z”:”a”}}}

# key = x/y/z

# value = a

def getKeys(obj: dict):

keys = list(obj)

if len(keys) != 1:

raise Exception('Multiple or Empty Doctionary found ')

else:

return keys[0]

def getNestedObjValue(obj: dict, key: str, isFound = False):

# print(obj, key, isFound)

if type(obj) is not dict and not isFound:

#return None

print('Not a nested Object')

if (isFound or (key in obj.keys())) :

if type(obj[key]) is dict:

return getNestedObjValue(obj[key], getKeys(obj[key]), True)

else:

return obj[getKeys(obj)]

else:

nestedKey = getKeys(obj)

return getNestedObjValue(obj[nestedKey], key, False)

if \_\_name\_\_ == '\_\_main\_\_':

obj = {'a': {'b': {'c': 'd'}}}

value = getNestedObjValue(obj, 'a')

value1 = getNestedObjValue(obj, 'b')

value2 = getNestedObjValue(obj, 'c')

print(value,value1,value2)

[ec2-user@ip-172-31-22-154 ~]$ python3 challange3.py

d d d

[ec2-user@ip-172-31-22-154 ~]$