# Overview

This function will monitor the health of an Instance (Paloaltonetworks Firewall) vi and move an elastic IP from the primary firewall instance to the secondary firewall instance. Intance health is determined by querying the status of the instance l. A “Running” status is assumed to be healthy, upon detection that the instance status is no longer healthy the lambda function will detach the Elastic IP associated with the interface and move it to the secondary firewall.

The use case presented here is for VPN failover where a customer site does not support multiple VPN tunnels to provide failover via dynamic routing. This use case is often found where customers are migrating from a traditional Active / Passive firewall cluster in a DC, to a VPC in AWS.

The solution requires that Tags are applied to ec2 instances and the elastic IP address that is to be moved between the firewalls. Tag Keys and Tag Values together with the firewall interfaces that the EIP should be bound to can be passed to the lambda function via environment variables.

Figure 1 – Test Network



**Availability Zone**

**Availability Zone**



Primary IPSEC Tunnel

Site Prefix 10/8 via

eBGP MED 100

Secondary IPSEC Tunnel

Site Prefix 10/8 via

eBGP MED 150



**Lambda**

Instance monitoring and Elastic IP Shift on Primary Instance Failure

Customer site

10.0.0.0/8

AS65300



**Availability Zone**

**Availability Zone**

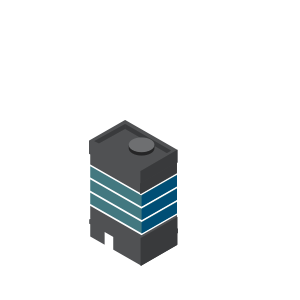
**VPN gateway**



**VPN connection**



**VPN connection**



Transit VPC

AS65301

eBGP AS-PATH Prepended

10/8 via FW2

AS-PATH 65301 65301 65300

eBGP

10/8 via FW2

AS-PATH 65301 65300

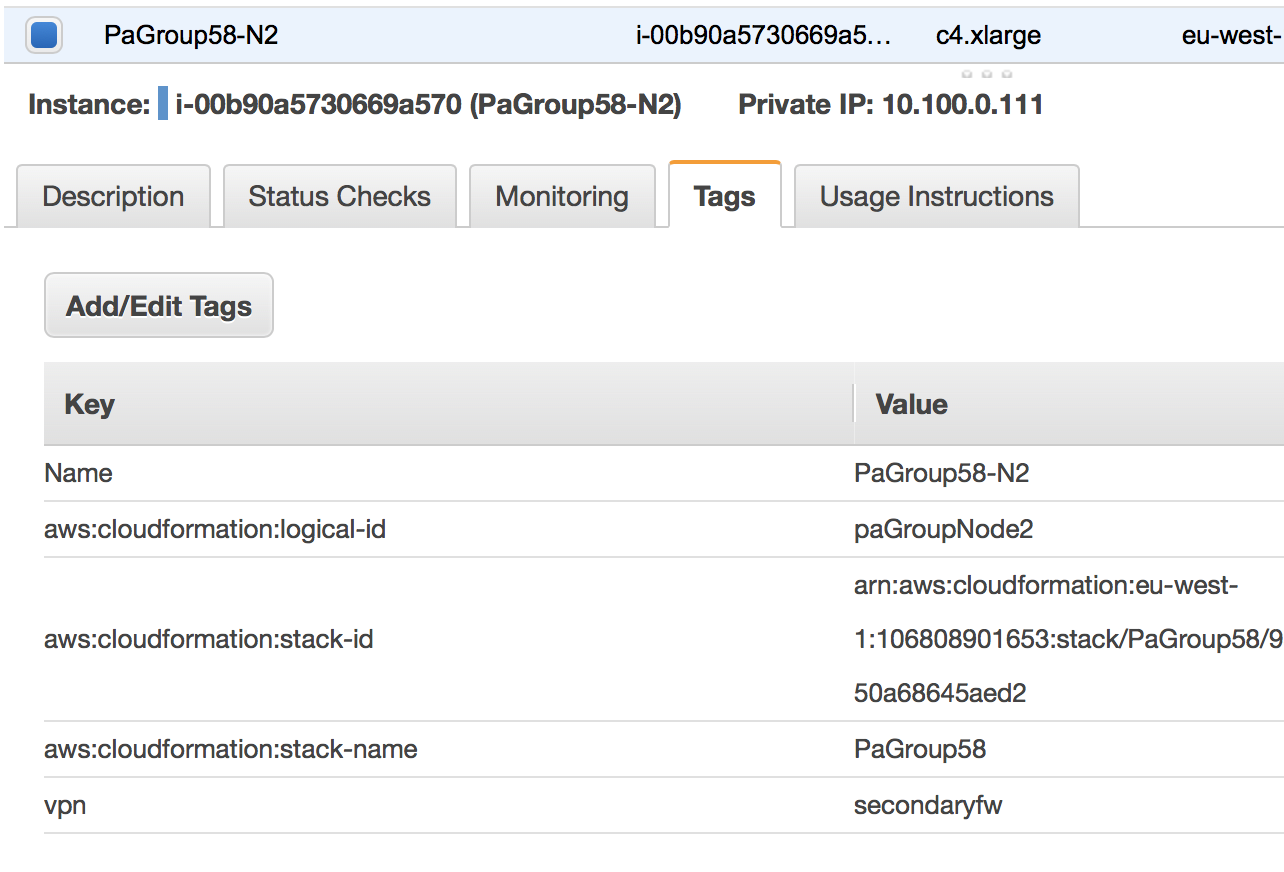
# Setup

1. Add Tags to Required Objects

The solution assumes that “Tags” will be applied to the VPC components to simplify the process of identifying instances to be monitored and the Elastic IP that will be moved.

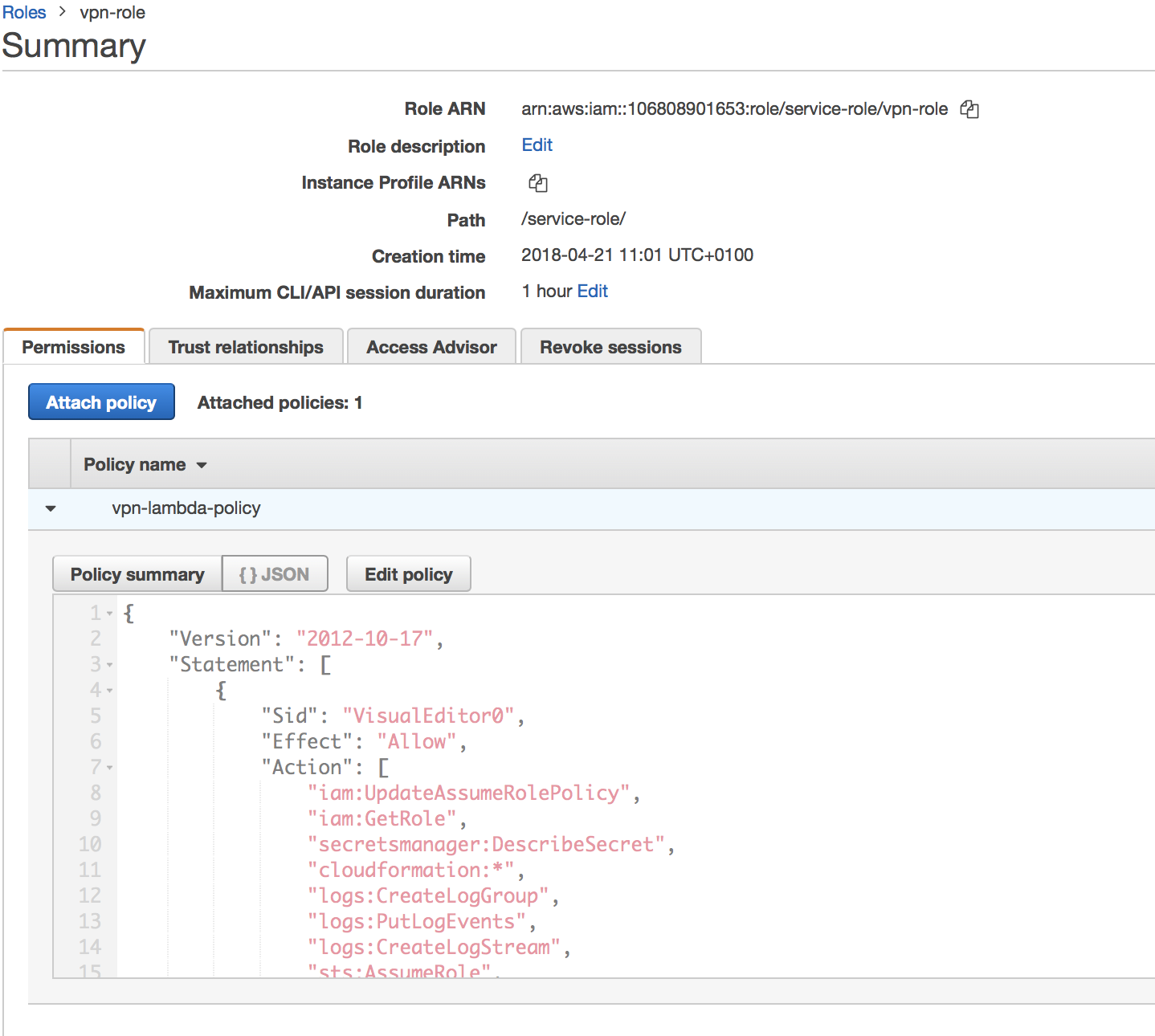
In this case the Tags that have been applied are as follows

|  |  |  |
| --- | --- | --- |
| **Instance** | **Tag-Key** | **Tag-Value** |
| Primary Firewall | vpn | prifirewall |
| Secondary Firewall | vpn | secfirewall |
| Elastic IP | vpn | floatip |



1. Create an IAM Policy and Role for the lambda function

In this example we have used the following policy



{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Action": [

"iam:UpdateAssumeRolePolicy",

"iam:GetRole",

"secretsmanager:DescribeSecret",

"cloudformation:\*",

"logs:CreateLogGroup",

"logs:PutLogEvents",

"logs:CreateLogStream",

"sts:AssumeRole",

"iam:PassRole",

"secretsmanager:GetSecretValue",

"sns:Publish",

"lambda:\*",

"ec2:\*",

"secretsmanager:ListSecrets"

],

"Resource": "\*"

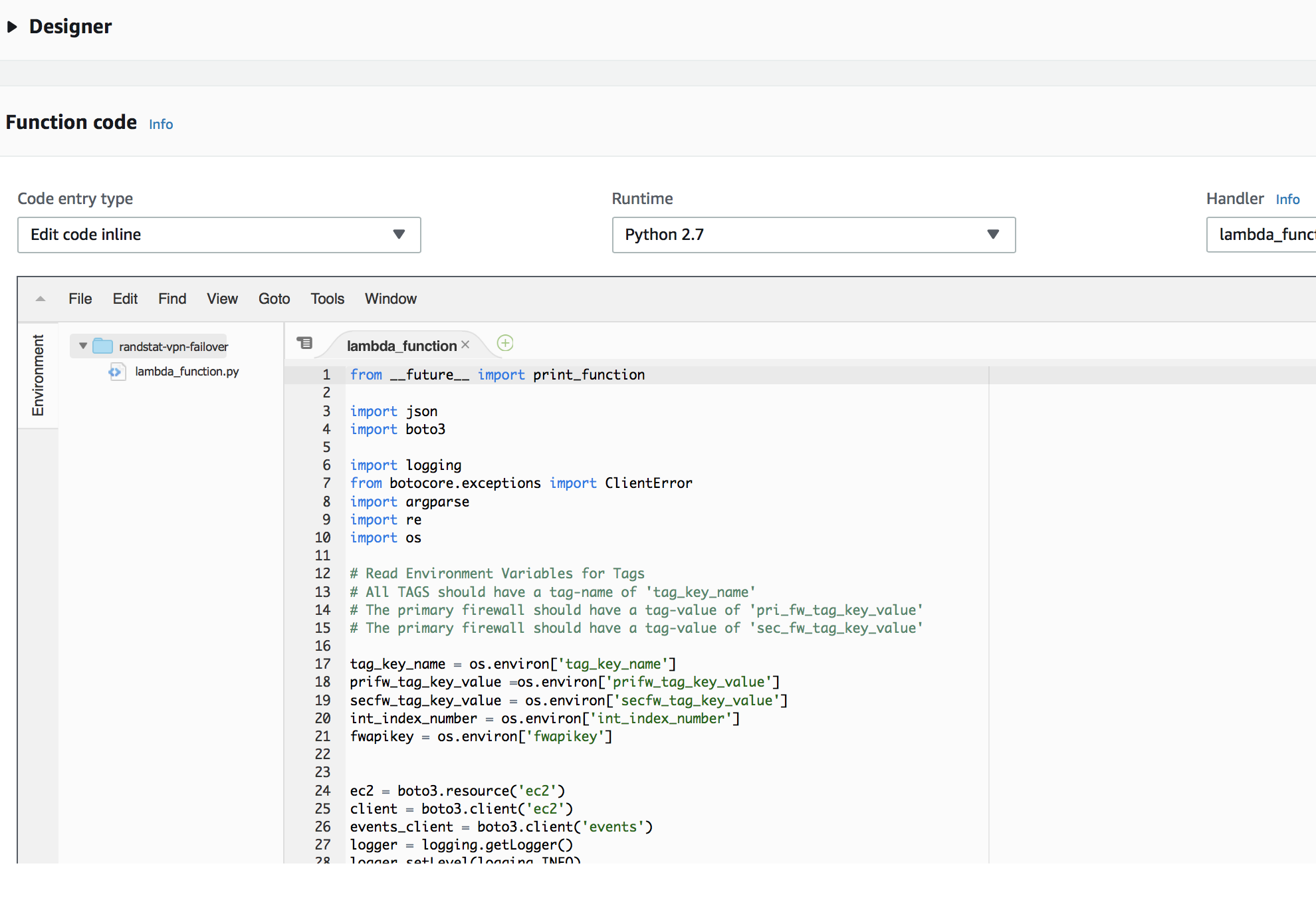
}

]

}

1. Create the Lambda Function

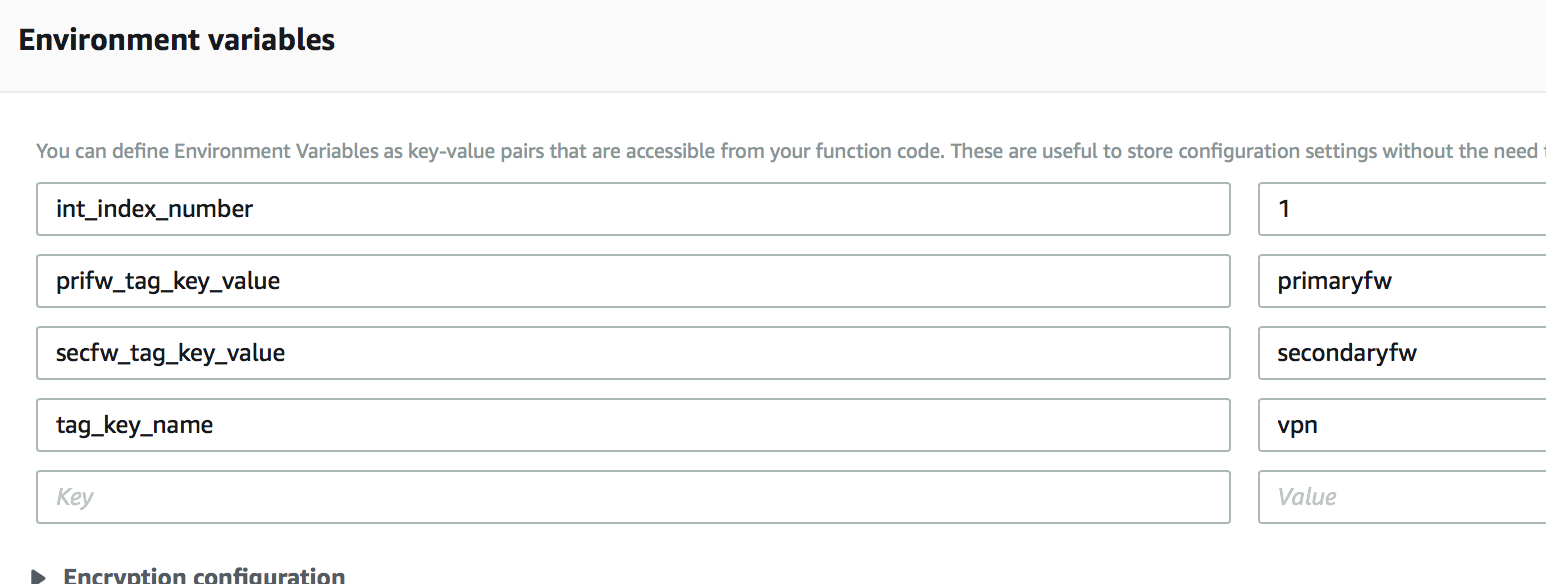
Create the lambda Function and Paste in the lambda from the Github repository



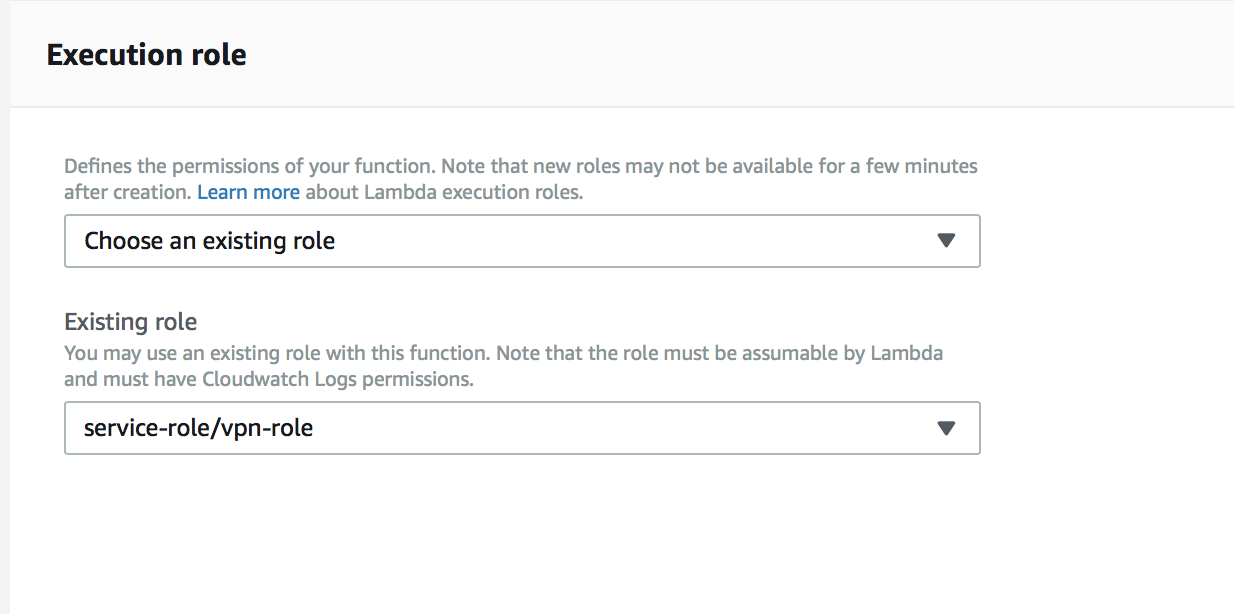
Next we will need to define the environment variables that we will pass into the lambda function

In this example

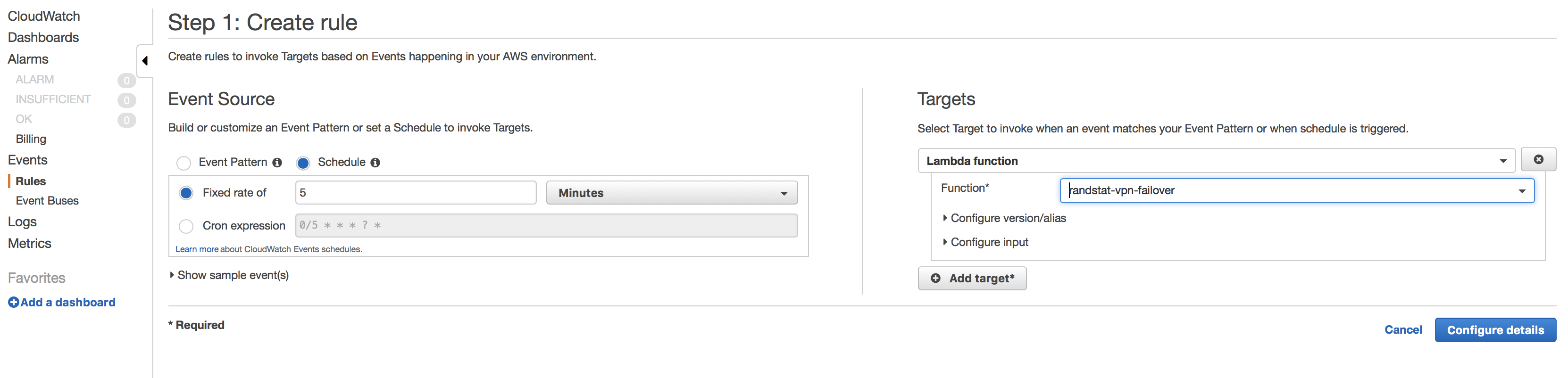
|  |  |
| --- | --- |
| Int\_index\_number | 1  (This is the index number associated with the firewall in this example we used ethernet1/1  eth1/1 ==1 ,eth1/2 ==2 ,eth1/3 ==3) |
| pri\_fw\_tag\_key\_value | primaryfw |
| sec\_fw\_tag\_key\_value | secondaryfw |
| tag\_key\_name | vpn |



Next associate the IAM role created earlier with the execution role



Define a trigger for the lambda function. The lambda is triggered via a Cloudwatch scheduled event. First create the event. In this case we have used a 5 minute trigger.



Create a cloudwatch event and create a trigger based on the rule we defined earlier.

