# Using the Web Interface

The Web Interface has been created to allow the manual creation of VPC vpn connections to Paloaltonetworks firewalls using modified lambda scripts originally provided in github

<https://github.com/PaloAltoNetworks/aws-transit-vpc> .

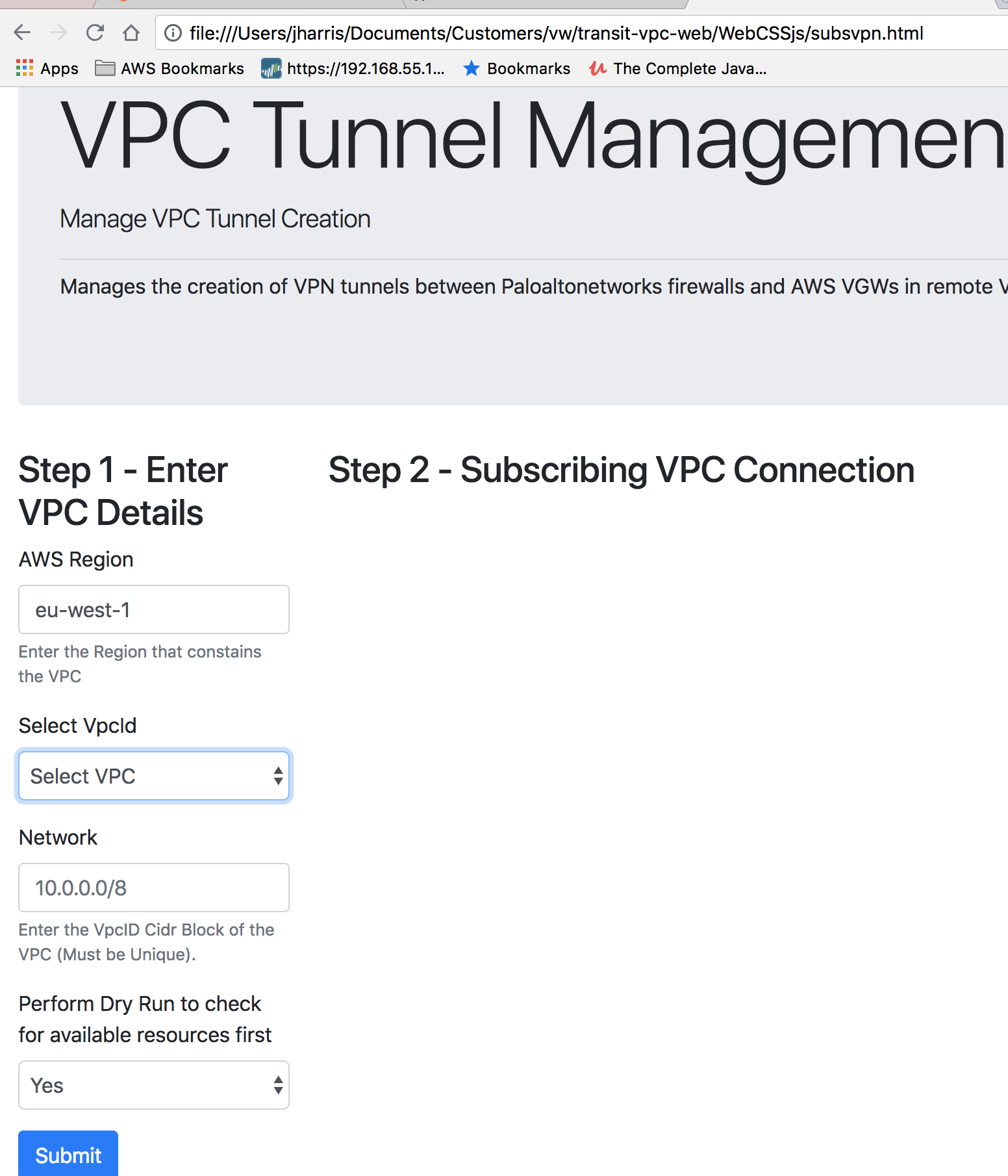
The lambda functions use the existing DynamoDB tables to allocate resources and track VPN connections.

Configuration parameters are submitted to the lambda functions via an AWS API gateway and the code has been written to allow the html pages to be hosted in S3.

VPN Creation Process.

The VPN creation process is as follows

1. Users loads the subsvpn.html page from S3



On document load a javascript function will send a query to the API gateway URL

<https://xxxxx/Production//getvpcs>

The lambda function returns a json object with list of VPCs VPC

[

{

"VpcId": "vpc-36cd8950",

"VpcCidr": "10.100.0.0/24"

},

{

"VpcId": "vpc-c6d591a0",

"VpcCidr": "11.220.0.0/24"

}

]

Javascript Function

*var dropdown = $('#VpcId');*

*dropdown.empty();*

*dropdown.append('<option selected="true" disabled>Select VPC</option>');*

*dropdown.prop('selectedIndex', 0);*

*// Populate dropdown with list of VpcId*

*$.ajax({*

*type: 'GET',*

*dataType: 'json',*

*async: true,*

*cache: true,*

*statusCode: {*

*502: function () {*

*alert('Got unknown error check Cloudwatch logs!');*

*}*

*},*

*url: "https://xxxxxx.execute-api.eu-west-1.amazonaws.com/Production/getvpcs",*

*success: function (response) {*

*vpcdata = JSON.parse(response.body);*

*$.each(vpcdata, function (key, entry) {*

*dropdown.append($('<option></option>').attr('value', entry.VpcId).text(*

*entry.VpcId));*

*})*

*}*

*});*

1. Test VPC Tunnel Creation

Select Yes ] perform a “dry run”. A dry run will check that there are no errors preventing tunnel creation. Typical errors include overlapping Network address ranges for subscribing VPCs and lack of available firewalls for VPN termination.

If successful the response page will show that the VPN connection can be created.

Javascript runs in the browser and submits the form data to the api gateway

<https://8n50iw32bg.execute-api.eu-west-1.amazonaws.com/Production/>

Javascript Function

*$("#subsbutton").click(function (event) {*

*event.preventDefault();*

*var formdata = $("form").serialize();*

*formdata = formdata + '&VpcId=' + $("#VpcId option:selected").val();*

*//console.log(formdata);*

*subscriberurl = 'https://8n50iw32bg.execute-api.eu-west-1.amazonaws.com/Production/';*

*$.ajax({*

*type: 'GET',*

*dataType: 'json',*

*async: true,*

*cache: true,*

*statusCode: {*

*502: function () {*

*alert('Got unknown error check Cloudwatch logs!');*

*}*

*},*

*data: formdata,*

*url: subscriberurl,*

*success: function (response) {*

*//alert((response)); //get the url used in the ajax query*

*console.log(response);*

*if (document.getElementById("myform").elements["dryrun"].value ===*

*'No') {*

*console.log(document.getElementById("myform").elements["dryrun"]*

*.value);*

*document.getElementById("SubsOutputTxtId1").style.visibility =*

*'visible';*

*if (response["Result"] === "Success") {*

*transitparams = $.param(response);*

*document.getElementById("subsbutton").style.display =*

*'none'*

*document.getElementById("transitsubmit").style.display =*

*'block';*

*}*

*}*

*var keys = Object.keys(response);*

*var txt = ""*

*txt += "<table class=\"table\"><tbody>";*

*for (j in response) {;*

*//console.log(j);*

*//console.log(response[j]);*

*txt += "<tr><td>" + j + "</td><td>" + response[j] +*

*"</td></tr>";*

*}*

*txt += "</tbody></table>";*

*document.getElementById("SubsOutputTxtId").style.visibility =*

*'visible';*

*console.log(txt);*

*//*

*// If response is received enable submit button to config Transit Account VPN connections*

*//*

*var divContainer = document.getElementById("SubsResultsId");*

*divContainer.innerHTML = txt;*

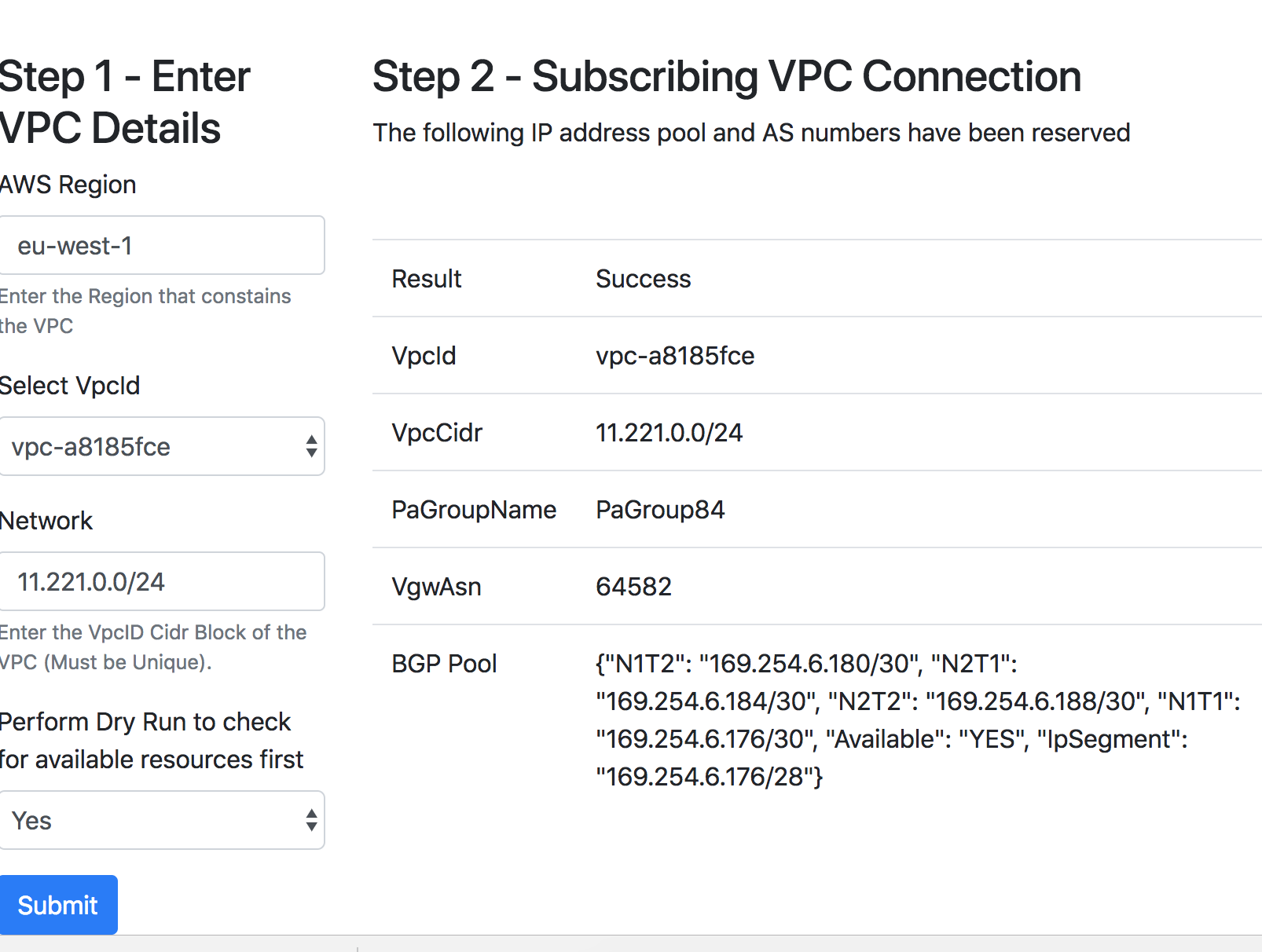
*divContainer.style.visibility = 'visible';*

*}*

*});*

*});*

*});*



3) Test VPC Tunnel Creation

Once the Dry run has succeeded change the selection box to “No”.

On Click Submit

If the VGW creation is initiated by the lambda function it returns an http 200 response to the browser. Javascript will enable a new submit button that will allow the creation of the transit VPN

*if (response["Result"] === "Success") {*

*transitparams = $.param(response);*

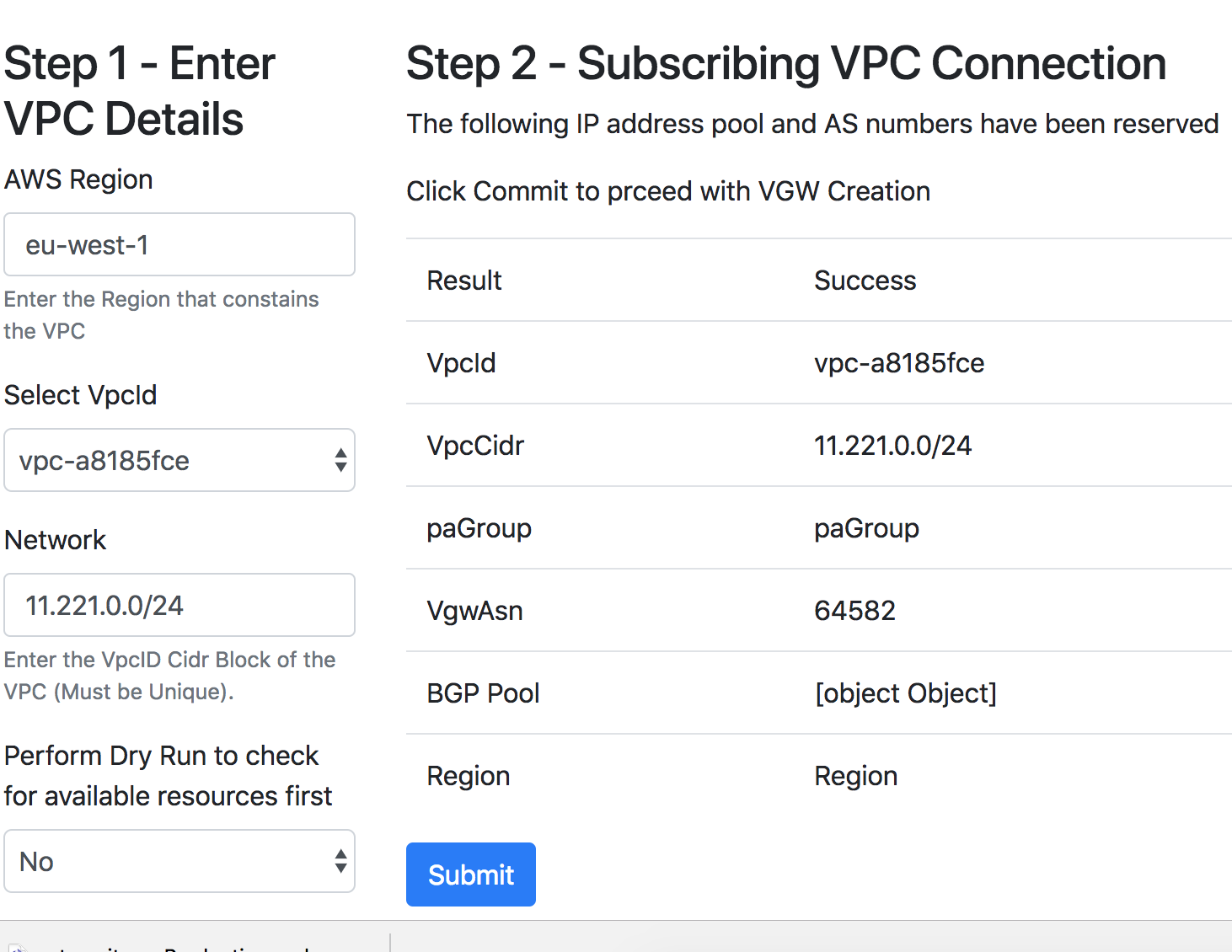
*document.getElementById("subsbutton").style.display =*

*'none'*

*document.getElementById("transitsubmit").style.display =*

*'block';*

*}*



The VGW creation process requires two lambda functions.

* apiAsyncCreateVGW.py
* apiCreateVgw.py

The api gateway invokes apiCreateVgw.py which allocates resources for the VGW and then returns a http 200 response to the browser and publishes an SNS message to a topic that apiAsyncCreateVGW.py subscribes to. We create this asynchronous function as the API gateway has a maximum timeout to requests of 29seconds and the VGW creation has been found in testing to take between 10 and 180 seconds to complete.

The payload of the SNS message is

*Input {  
 "Result": 'Success',  
 "VpcId": event['queryStringParameters']['VpcId'],  
 "VpcCidr": event['queryStringParameters']['VpcCidr'],  
 "PaGroupName": paGroup['PaGroupName'],  
 "vgwAsn": str(vgwAsnNumber),  
 "N1Eip": paGroup['N1Eip'],  
 "N2Eip": paGroup['N2Eip'],  
 "N1Asn": paGroup['N1Asn'],  
 "N2Asn": paGroup['N2Asn'],  
 "N1T1": bgpIpPool['N1T1'],  
 "N1T2": bgpIpPool['N1T2'],  
 "N2T1": bgpIpPool['N2T1'],  
 "N2T2": bgpIpPool['N2T2'],  
 "IpSegment": bgpIpPool['IpSegment'],  
 "Region": region  
}*

1. Create Transit VPC VPN

Upon completion of the VGW creation the Transit VPN configuration can be performed.

# Swagger Definition of API Gateway

---

swagger: "2.0"

info:

version: "2018-07-30T01:09:26Z"

title: "transitvpc"

host: "8n50iw32bg.execute-api.eu-west-1.amazonaws.com"

basePath: "/Production"

schemes:

- "https"

paths:

/:

get:

produces:

- "application/json"

responses:

200:

description: "200 response"

schema:

$ref: "#/definitions/Empty"

headers:

Access-Control-Allow-Origin:

type: "string"

options:

consumes:

- "application/json"

produces:

- "application/json"

responses:

200:

description: "200 response"

schema:

$ref: "#/definitions/Empty"

headers:

Access-Control-Allow-Origin:

type: "string"

Access-Control-Allow-Methods:

type: "string"

Access-Control-Allow-Headers:

type: "string"

/checkvgwstatus:

get:

produces:

- "application/json"

responses:

200:

description: "200 response"

schema:

$ref: "#/definitions/Empty"

options:

consumes:

- "application/json"

produces:

- "application/json"

responses:

200:

description: "200 response"

schema:

$ref: "#/definitions/Empty"

headers:

Access-Control-Allow-Origin:

type: "string"

Access-Control-Allow-Methods:

type: "string"

Access-Control-Allow-Headers:

type: "string"

/getvpcs:

get:

produces:

- "application/json"

parameters:

- name: "VpcId"

in: "query"

required: false

type: "string"

- name: "Region"

in: "query"

required: false

type: "string"

responses:

200:

description: "200 response"

schema:

$ref: "#/definitions/Empty"

headers:

Access-Control-Allow-Origin:

type: "string"

options:

consumes:

- "application/json"

produces:

- "application/json"

responses:

200:

description: "200 response"

schema:

$ref: "#/definitions/Empty"

headers:

Access-Control-Allow-Origin:

type: "string"

Access-Control-Allow-Methods:

type: "string"

Access-Control-Allow-Headers:

type: "string"

/transitvpn:

get:

produces:

- "application/json"

responses:

200:

description: "200 response"

schema:

$ref: "#/definitions/Empty"

headers:

Access-Control-Allow-Origin:

type: "string"

options:

consumes:

- "application/json"

produces:

- "application/json"

responses:

200:

description: "200 response"

schema:

$ref: "#/definitions/Empty"

headers:

Access-Control-Allow-Origin:

type: "string"

Access-Control-Allow-Methods:

type: "string"

Access-Control-Allow-Headers:

type: "string"

definitions:

Empty:

type: "object"

title: "Empty Schema"