FortiGate Python-Based Auto Scaling with Hybrid Licensing for AWS

Table of Contents

AWS Auto Scale Overview

It is typical for businesses to experience varying levels of demand on their digital infrastructure. The AWS public cloud platform provides several benefits through demand elasticity combined with cost optimization. Chief among these is Auto Scaling. Auto Scaling monitors compute resources and, based on utilization, will dynamically scale out the environment to support the additional demand. Conversely, when peak utilization has passed, low-watermark thresholds trigger the environment to contract, or scale in. Typical reason to deploy auto scaling include seasonal variation, failover or disaster recovery, and general cost optimization.

When considering if auto scaling is a fit for your needs, consider the variability in traffic patterns relative to a baseline. While auto scaling is great for business continuity and agility, right sizing the environment for the minimum amount of traffic will help to minimize the amount of scale-in and scale-out events to ensure the most stable environment.

A typical baseline starts with a traffic volume assessment. When deploying a security network virtual appliance (NVA), such as the FortiGate NGFW, it is important to consider processing requirements for the traffic volume being handled on a normal basis. Processing variance depends on the types of inspection being done. For example, stateful firewall and logging will consume far less resources than full threat protection and encryption functions (IPS and VPN).

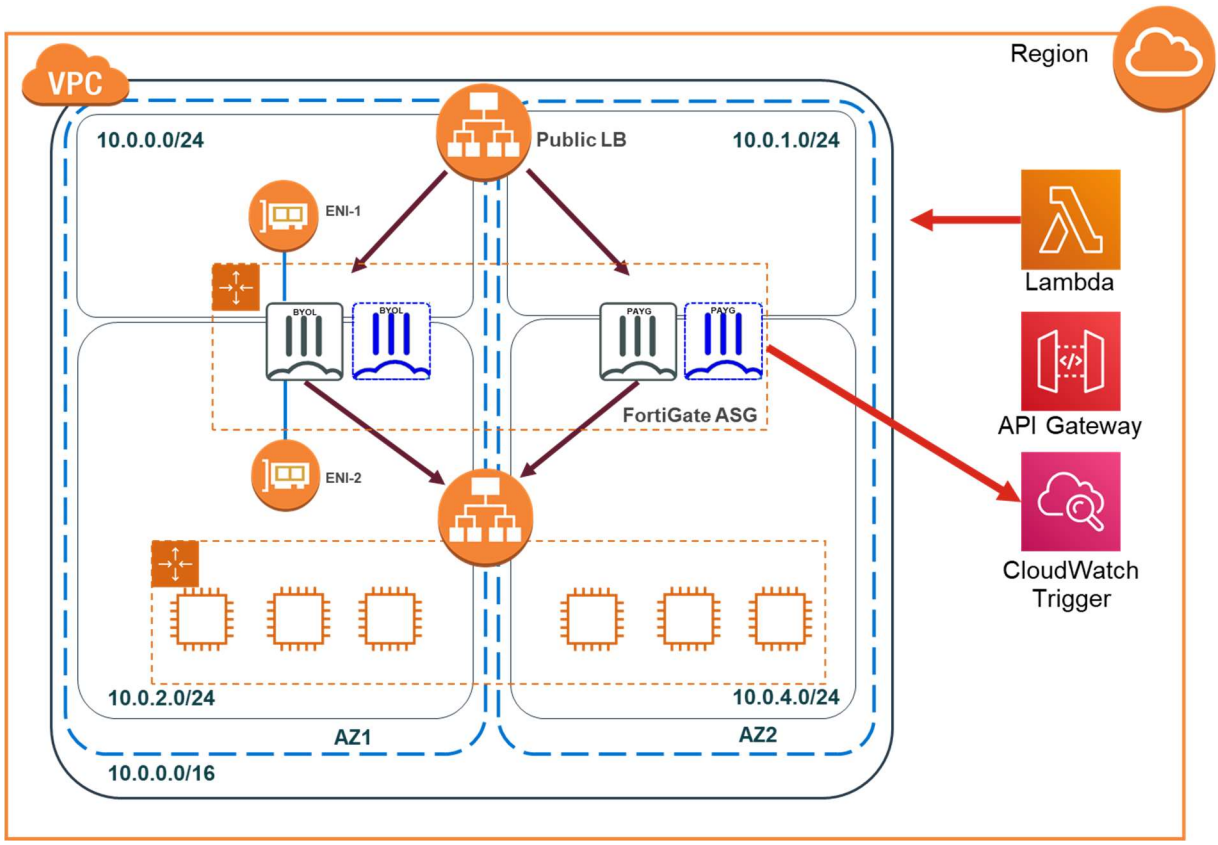
FortiGate Auto Scale Overview

Note that there are multiple versions of Fortinet's Auto Scale solution that are purposeful to different use cases and deployment scenarios. For example, Transit VPCs with ECMP VPNs. These different flavors of auto scale achieve the goal of scaling services though their frameworks are slightly different. For clarity, this document will refer to the template described for this particular deployment case and not make reference to any alternative auto scale solution.

Fortinet's auto scale solution provides many features that align with the intent of auto scaling. Hybrid licensing provides the best performance to cost by matching persistent bring-your-own-license (BYOL) instances with scale out pay-as-you-go (PAYG) instances. Both ingress and egress use cases are supported without the use of extra instances or NAT gateways.

FortiGate Auto Scale Architecture

Figure 1. Auto Scale Architecture

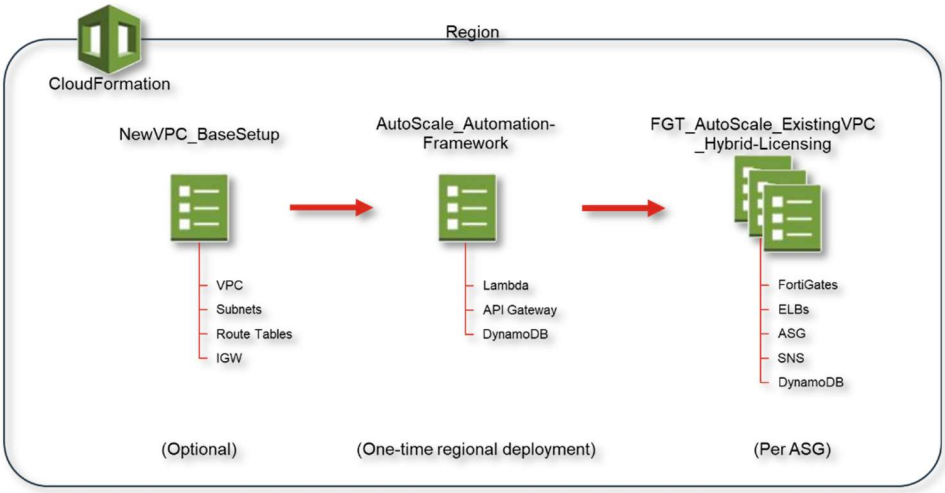


There are three templates used during the phases of auto scale deployment, supporting both new and existing VPCs.

- **NewVPC_BaseSetup** – This template is for the convenience of setting up any new VPC. Output of this stack is the VPC, Subnets, and route tables required to support the Auto Scale deployment. This an optional stack which is not necessary for existing VPC deployments.
- **AutoScale_Automation-Framework** – Deploys the Lambda service and supporting API gateway and DynamoDB table that will track all ASG groups deployed in the region.
- **FGT_AutoScale_ExistingVPC_Hybrid-Licensing** – Deploys the Auto Scale Group (ASG) and supporting AWS services such as the ELBs, target groups, instances, and SNS.

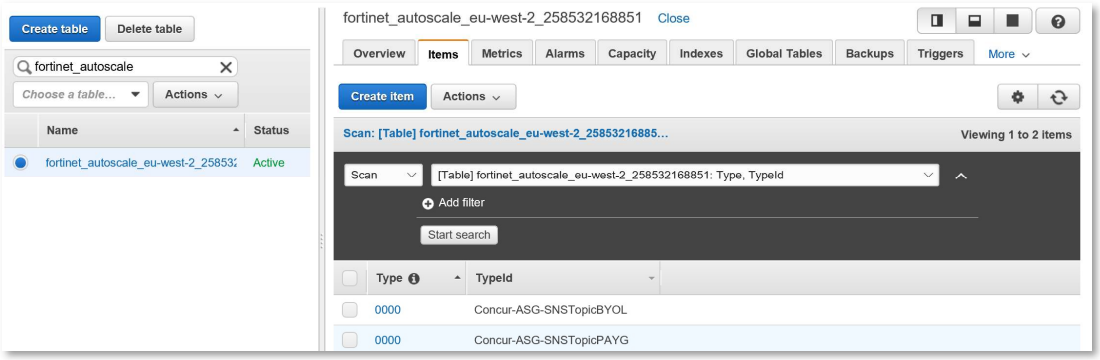
Figure 1 shows the relationship between the templates. Use the **NewVPC_BaseSetup** only when a new VPC is required. The outputs of this VPC can be fed into the **AutoScale_Automation-Framework** CFT. The AutoScale_Automation-Framework CFT will install the region's Lambda function which will manage all ASGs in the region.

Figure 2. Relationship between CloudFormation Templates for Auto Scale



The Lambda function provides a registration framework by creating DynamoDB table entry for each ASG deployed in the region. In this way, the AutoScale_Automation-Framework template only needs to be deployed once per region.

Figure 3: DynamoDB table tracking ASG deployments in region



FGT_AutoScale_ExistingVPC_Hybrid-Licensing is the core ASG CFT that will deploy each ASG instance. Outputs from NewVPC_BaseSetup and AutoScale_Automation-Framework are used as inputs for the ASG.

Service Interactions

Figure 4. provides an overview of the different services deployed and how they interact. For the CloudFormation-based mode of deployment, the services created from the previously described CFTs interact primarily through the Simple Notification Service (SNS) and the API Gateway. Auto Scaling is deployed as two ASGs – one for BYOL and a second for PAYG. These ASGs make up an Auto Scale Deployment (ASD). Each of these ASGs within the ASD can be configured independently to provide flexibility in the CAPEX or OPEX interests of the business. For example, if the business requires a very small baseline of a single FortiGate with no high availability (HA), a single BYOL instance can be instantiated with a minimum ASG size of 1. The PAYG ASG can be set with a minimum of 0 and a maximum desired of 5, as an example. Such a case may also be preferable in the scoping phase of the implementation where a solid baseline has not yet been set. As the needs of the business change, the ASG minimum and maximum values for both BYOL and PAYG can be adjusted.

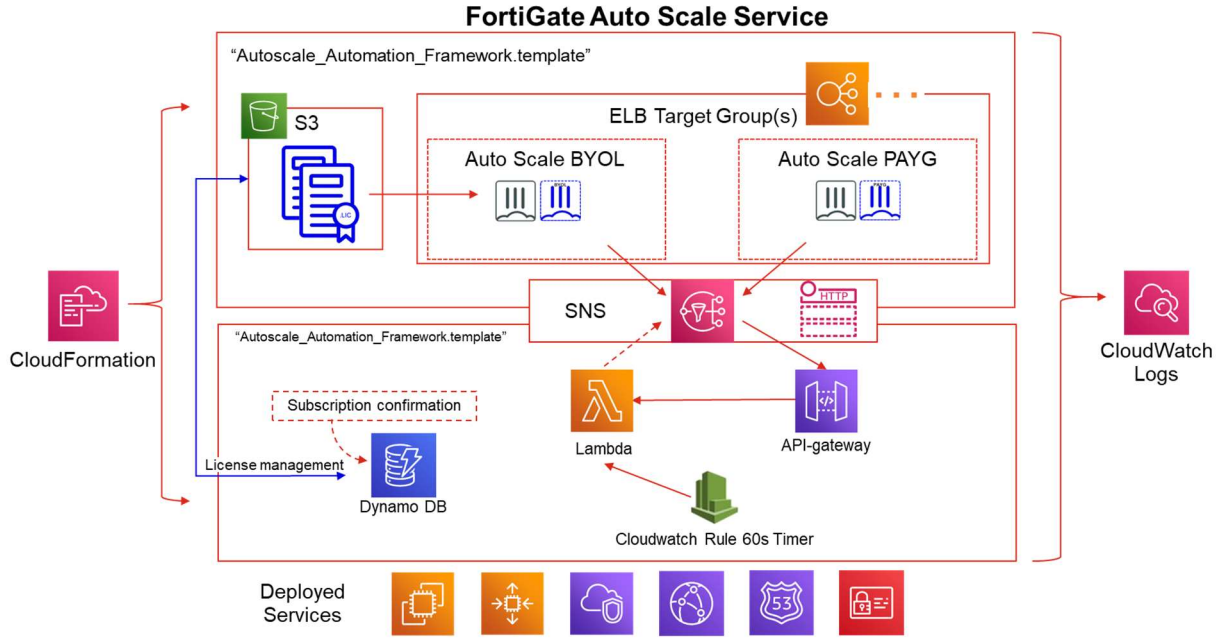
When a new ASD is deployed, auto scale events are posted to SNS which then notifies the Lambda function via the API gateway of scaling events and EC2 instance service statuses. SNS uses HTTPS for protected communication. Further reading on this topic can be found at the following URLs.

Auto Scaling Lifecycle:

https://docs.aws.amazon.com/autoscaling/ec2/userguide/lifecycle-hooks.html

https://docs.aws.amazon.com/autoscaling/ec2/userguide/AutoScalingGroupLifecycle.html

Figure 4: Auto Scale Service Interactions



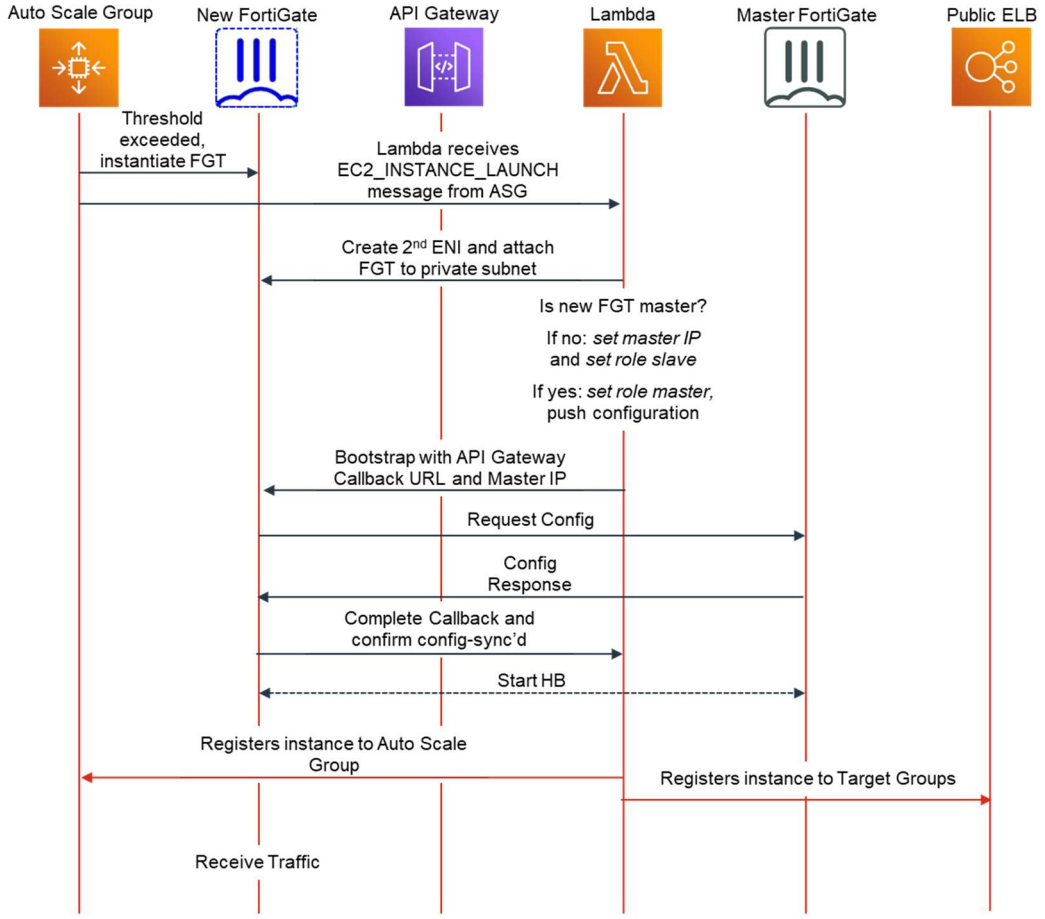
On instance instantiation, the newly launched FortiGate will transition through the “Pending:Wait”, “Pending:Proceed”, and finally to “InService”. When an auto-scaled FortiGate comes to InService status, it will bootstrap with a basic configuration which includes a basic auto scale configuration for it to communicate with the API Gateway.

Example 1: config sys auto-scale

```
FGVM04TM19000016 (auto-scale) # show
config system auto-scale
    set status enable
    set sync-interface "port3"
    set callback-url "https://someurl.apigateway.aws.com"
end
```

The FortiGates support auto scaling natively in the FortiOS (FOS) starting with version 6.0. In this way, the slave FortiGates can source their configuration from a master FortiGate. As each ASD member comes online, Lambda will configure the FortiGate’s initial, base configuration based on it’s designated role as master or slave. Lambda’s actions here are triggered by lifecycle hook message EC2_INSTANCE_LAUNCH. The FortiGates in the ASD will exchange a heartbeat message that will ensure that each device is still available and that it has the latest configuration. Should the master configuration change, the slave FortiGates will be updated immediately. Consequently, the user will only ever need to set the configuration on the master. The following summarizes the configuration steps.

Figure 5. New FortiGate configuration



BYOL License Management

The ASD supports hybrid, or mixed-mode, licensing options. When using bring-your-own-licensing (BYOL) instances, the FortiGate license files must be provided supplied to a user-defined S3 bucket. This parameter is defined when launching the "FGT_AutoScale_ExistingVPC_Hybrid-Licensing" template. Note that the licenses should be supplied prior deploying this CFT if the minimum instance count is set to >0. When the FortiGate turns up and reaches out to the Lambda service for configuration instructions, Lambda will recognize the instance as BYOL and will push a license to the new FortiGate. These licenses are then updated in a DynamoDB table for tracking the license service status and the FortiGate to which the license belongs. Should an instance be terminated, Lambda will remove the license record from the database and will make that license available in the license pool for subsequent FortiGates.

Master FortiGate Selection

Within the Auto Scale Deployment (ASD), which may be comprised of multiple ASGs, there is one master from which all other slave FortiGates source their configuration. Configurations need only occur on the master which then sync's to all other FortiGates in the ASD. Configuration can be provided by using cloud-init, a configuration file upload, via API, CLI, web GUI, or FortiManager. The configuration of the master FortiGate is backed up to S3 by Lambda. Note that no FortiGate is bootstrapped with a status of master; Lambda sets this status based on its selection process. The status of all FortiGates are tracked in DynamoDB.

Master FortiGate High Availability

Master FortiGate failover is managed by Lambda. Should the master FortiGate fail (not respond to a health check), Lambda will elect a new master determined to be the FortiGate with the longest uptime. There is currently no manual intervention to deterministically set the master. To change the master assignment, reboot the member FortiGates who's uptime is greater than the desired master. The status of the master FortiGate is tracked in DynamoDB as shown in *Figure 6*.

Figure 6. DynamoDB entry with master role assignment

Uptime.

Deploying an Auto Scale Group

Note: This assumes that the **AutoScale_Automation-Framework** has been deployed in the region.

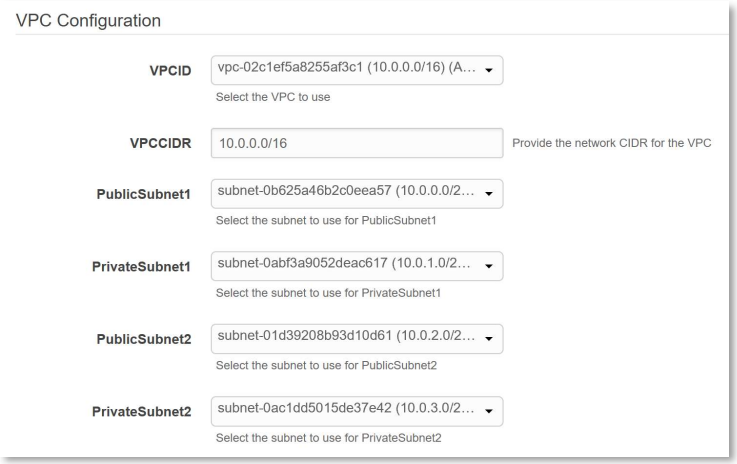
Note: The template locations and instructions may change with iterative versions of the solution. Please check the Readme file at https://github.com/aidanwalden/FortiGate_ASG_CFT_Lambda_ExistingVPC for updates.

Step 1: If not already downloaded, download athe FGT_AutoScale_ExistingVPC_Hybrid-Licensing.template.json from Github at https://github.com/aidanwalden/FortiGate_ASG_CFT_Lambda_ExistingVPC

Step: 2: Create the Stack

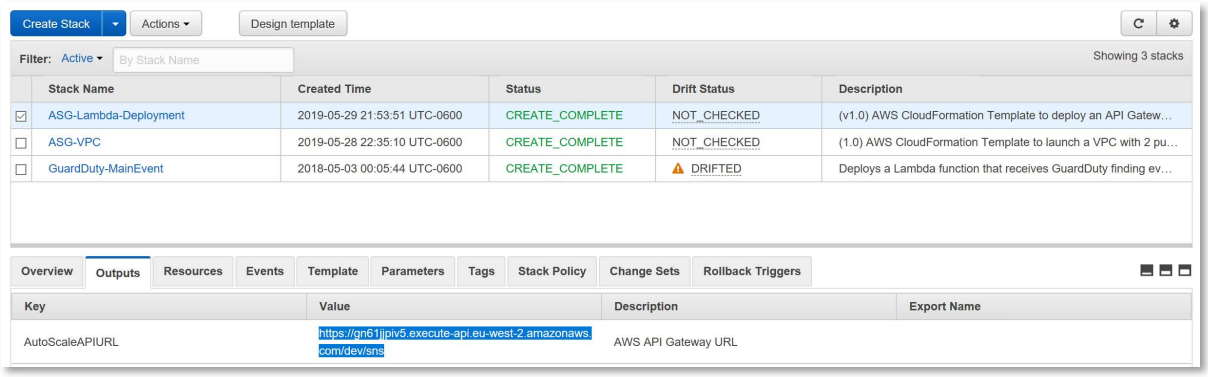
a. In the AWS console, navigate to the CloudFormation Dashboard and select "Create Stack"
b. Select to upload the FGT_AutoScale_ExistingVPC_Hybrid-Licensing template from either S3 or a local store
c. Enter the CFT parameters the VPC into which the ASG will be deployed. Public and Private subnet details are required for each Availability Zone used.

Figure 3: Existing VPC details for ASG



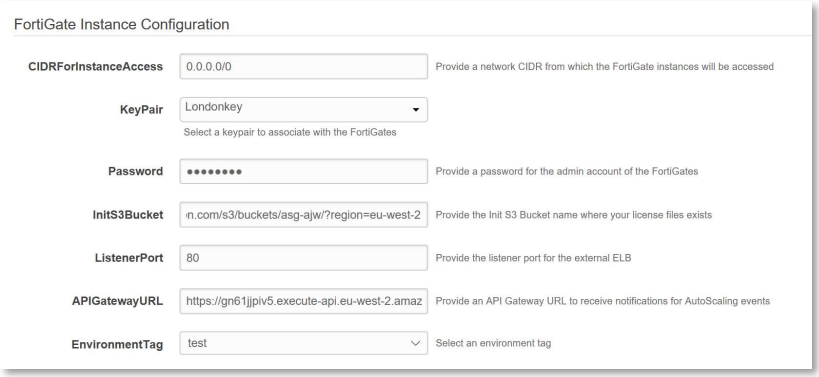
d. Locate the APIGatewayURL which can be found from the output of the AutoScale_Automation-Framework

Figure 4: APIGatewayURL from AutoScale_Automation-Framework



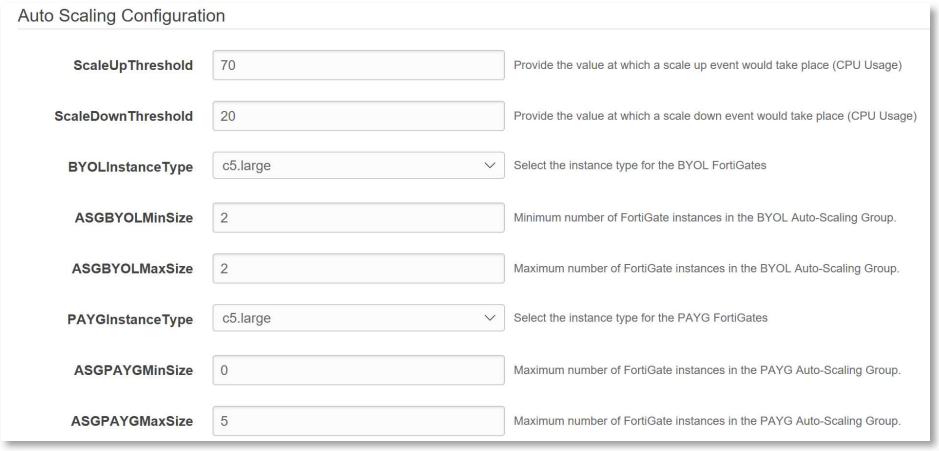
e. Enter the FortiGate Instance Configuration parameters. This includes keypair, S3 bucket for license management, etc.

Figure 5: Sample inputs for FortiGate Instance Configuration



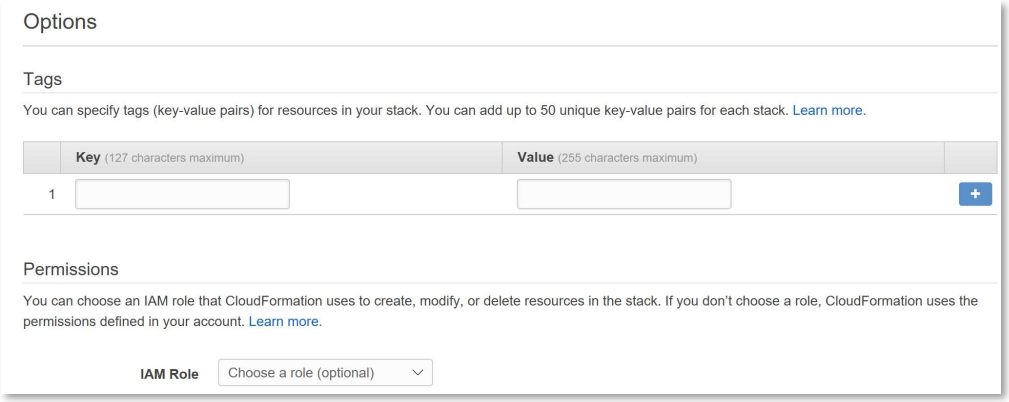
f. Enter the AutoScale parameter details.
   i. For production environments, the **BYOLInstanceType** should be sized based on the proper baseline requirements as estimated
   ii. The **ASGBYOLMinSize** is the minimum number of BYOL instances that will be in service at any time. Ensure that the minimum number of licenses allocated are available in the InitS3Bucket designated

Figure 6: Auto Scaling Configuration Parameters



g. Apply Tags as needed to identify this stack
h. If an existing IAM role is to be used, it can be chosen. Otherwise, an IAM role will be created.

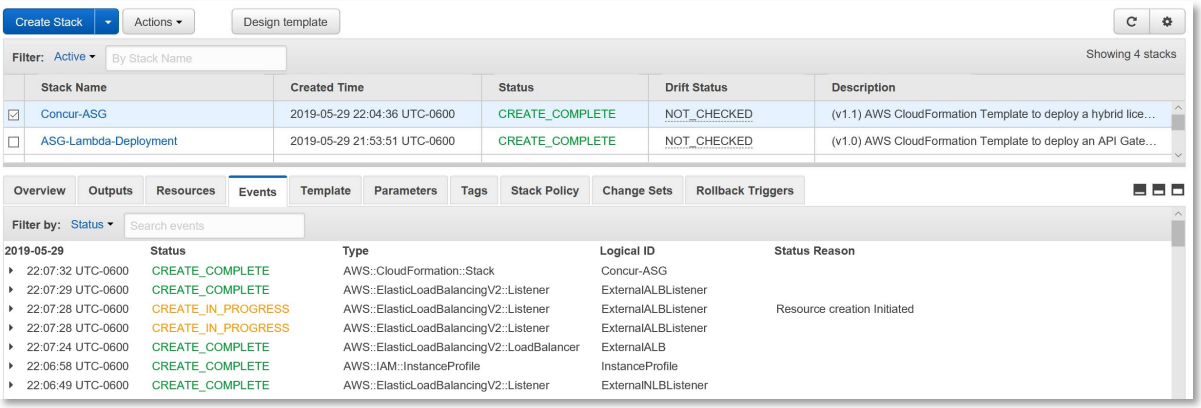
Figure 7: Tags and IAM roles



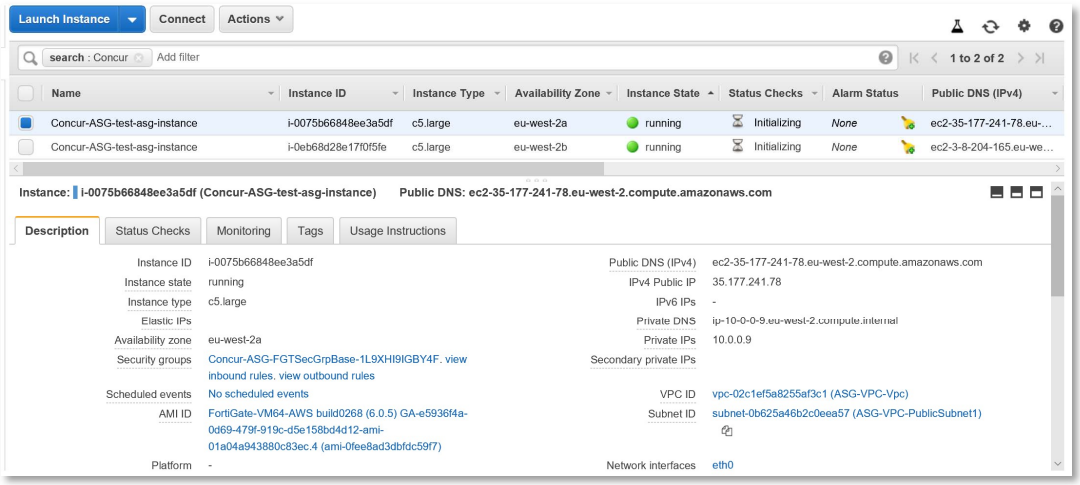
i. Select your rollback behavior and click Next to Review
j. If the parameter details are correctly entered, click “Launch Stack”

Step 3: Validate FGT_AutoScale_ExistingVPC_Hybrid-Licensing Stack Deployment

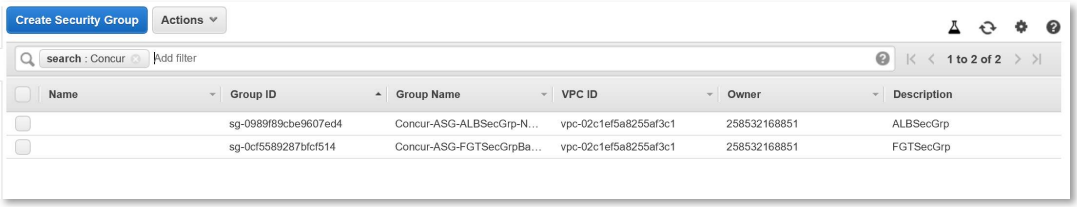
a. In the CloudFormation dashboard, validate that the ASG has a status of “CREATE_COMPLETE”



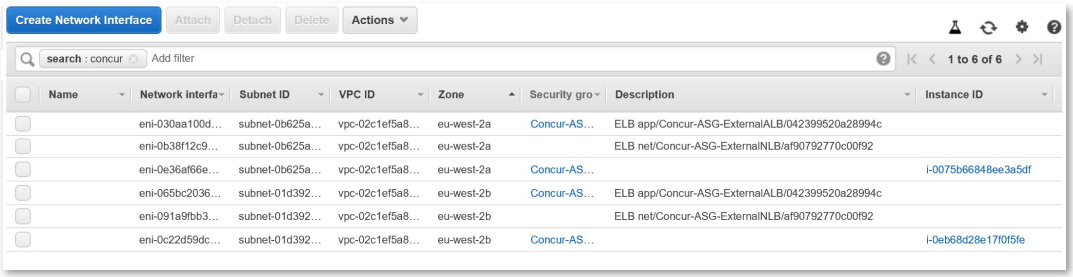
b. Navigate to the EC2 dashboard and verify that the minimum number of instances of BYOL, and if selected, PAYG instances are instantiated. Validate that the FortiGates have 2 ENIs attached and that they are attached to the proper subnets.



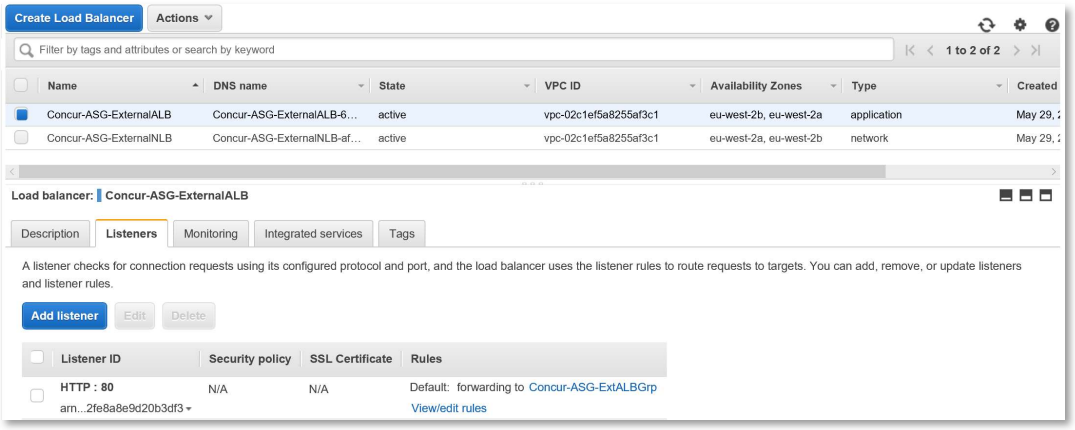
c. Validate that the Security Groups are created for the FortiGates and the ELBs



d. Validate that the ENIs are created and attached to the FortiGates and the ELBs

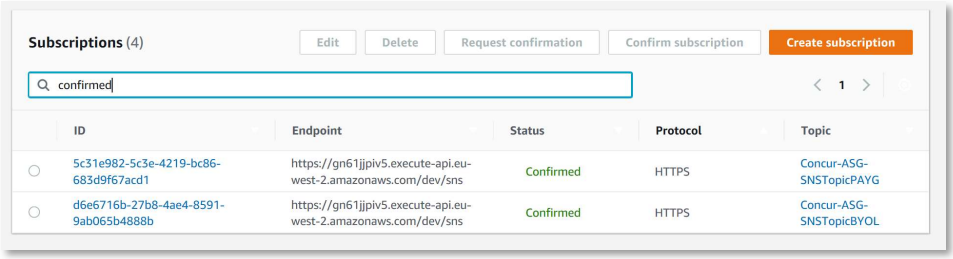


e. Validate LBs are deployed with the appropriate listeners (FortiGate policy will have to allow the ports configured on the ELBs)



f. Validate that the ASG has deployed with 2 ASGs – one each for BYOL and PAYG

g. Validate that the SNS topics has valid subscriptions that are in a “Confirmed” Status



API Callback and SNS subscription
Modifying the Auto Scale Deployment

1. Adding load balancers
2. Dependencies