Solution for BIG-IP Autoscale in AWS using BYOL and Utility Licensing

License

See LICENSE.TXT in top-level directory.

Introduction

This document discusses the code examples in this repository, which show how BIG-IP can be used to implement an auto-scaled web-application-firewall tier in AWS. In these code examples, BYOL and utility instances are deployed in parallel to meet base and variable traffic demands, respectively.

Prerequisites

The following solution requires two features introduced in version 12.0 of BIG-IP:

- Integration with EC2 Autoscale servers for pool member management
- Ability to use CloudInit for configuration of BIG-IP at start-up.

Access to both utility and BYOL instances in the AWS marketplace is required.

Concepts

High-level overview/topology

- BYOL instances make up baseline throughput and peak throughput is met using utility BIG-IP instances, which are scaled using EC2 Autoscaling.
- BYOL and Utility versions of BIG-IP are placed in the same ElasticLoadBalancer group, so the total throughput is the sum of utility + BYOL instances.
- To auto-scale the utility set of BIG-IPs, all BIG-IP instances push metrics to CloudWatch. These metrics are aggregated and used within CloudWatch alarms, which trigger Autoscale policies.
- This solution assumes that application servers are also managed via an EC2 Autoscale group. Each BIG-IP independently queries AWS endpoints to determine the active set of pool members.
- The solution is deployed within a single availability zone, but could be updated to meet availability SLAs by deploying across availability zones.
- In deployments with multiple availability zones, BIG-IP should be deployed in a routed network topology (rather than directly connected), to ensure that pool members are reachable by BIG-IPs in adjacent subnets.

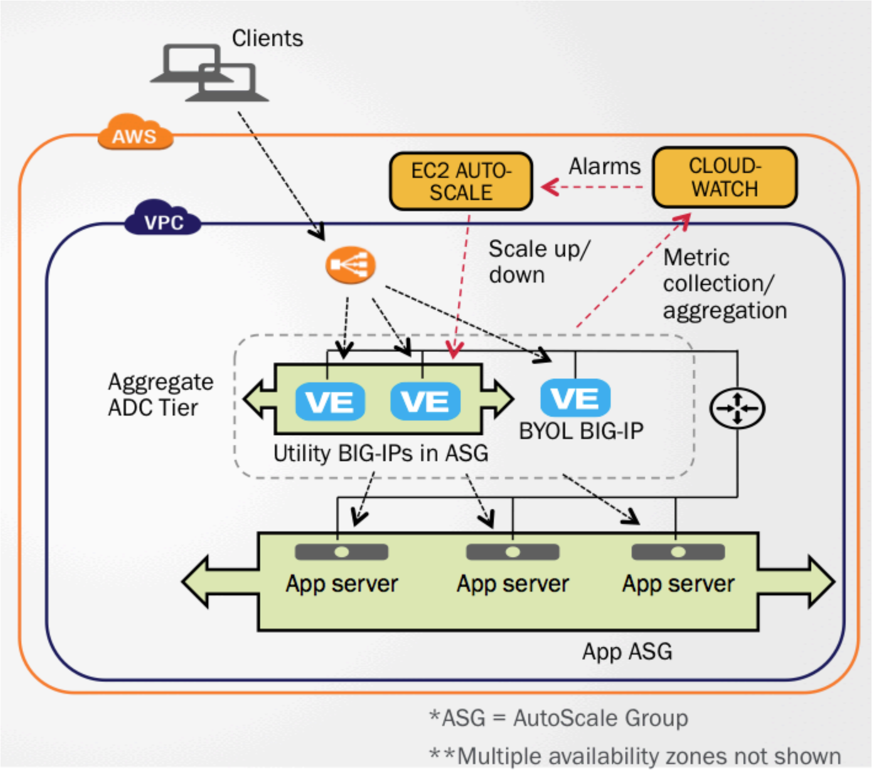
- 

Figure 1: Solution architecture in AWS

Licensing and Throughput

- All BIG-IPs (utility and BYOL) should be the have the same throughput capacity. They will receive traffic from ELB, which does not implement priority-based load balancing.

Configuration Management

- CloudInit can be used to provision the BIG-IP configuration at time of launch. Because CloudInit has a16kb limit, large files (i.e. iApps or ASM policies) will need to be hosted via some external service. For example, this could be accomplished using 'curl' to download an ASM policy stored in S3.

- Device Service Clustering is not used in this solution.

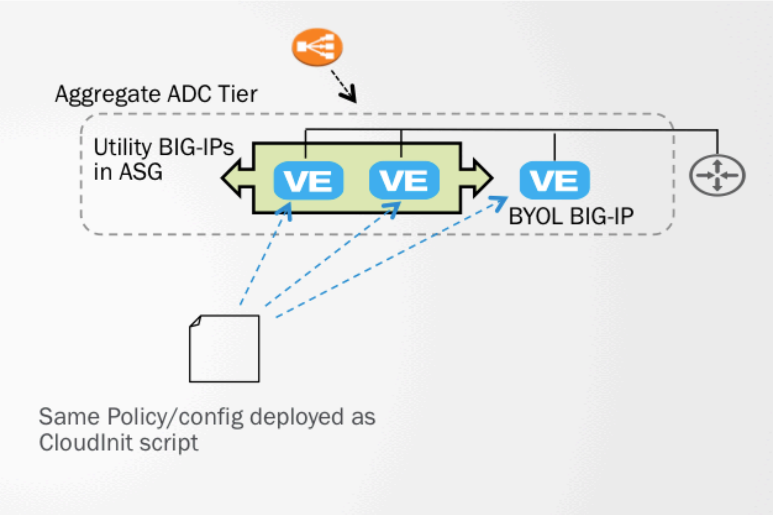


Figure 2: CloudInit is used to setup BIG-IP at time of launch

High-availability, Cross-Availability Zone Deployments, and Persistence

- High-availability may be provided by striping the deployment across multiple availability zones. If one zone dies, ELB would redistribute traffic to the remaining BIG-IPs.
- Cookie persistence will ensure that connections make it back to the same backend pool member, regardless of which BIG-IP processes traffic.

CloudWatch Alarms, Scaling Policies, and Custom metrics

- BIG-IP pushes custom metrics to CloudWatch in a CloudWatch "Namespace". This can be configured on any 12.0 BIG-IP instance launched from the AWS Marketplace, even if that instance is not managed by the EC2 Autoscale service. By creating a custom Namespace, we can aggregate these custom metrics for all BIG-IPs deployed.
- The time required to scale up a new utility instance averages 21 minutes, depending on configuration steps in the CloudInit script defined within the BIG-IP Launch Configuration.
- In advance of anticipated, high-throughput events, it would be possible "pre-warm" the BIG-IP autoscale pool by specifying the number of instances in the BIG-IP Autoscale group.

Integration with EC2 for Application Server Autoscaling

- BIG-IP will reference the EC2 Autoscale group to obtain a list of active pool members. Each BIG-IP will independently poll AWS endpoints for this list.
- See official F5 documentation referenced at the end of this document for more information on EC2 pool member autoscaling

Example code

Contents

The sample code in this repository includes the following:

- Four CloudFormation Templates (CFTs)
    - These CFTs deploy BYOL and utility version 12.0 BIG-IPs within the same ELB group, along with another EC2 Autoscale group for application servers. These templates are:
        - common.json - Deploys common EC2/VPC resources which will be used by the other templates. In particular, this template creates a VPC, subnets, a routing table, common security groups, and an ElasticLoadBalancer group to which all BIG-IPs will be added.
        - application.json - Deploys all components to support the application servers. An Autoscale group for the application is created, but we leave the creation of CloudWatch alarms and scaling policies as an exercise for the future.

    - autoscale-bigip.json - Deploys an Autoscale group for utility instances BIG-IP. Example scaling policies and CloudWatch alarms are associated with the Autoscale group.
    - byol-bigip.json - Deploys a single BYOL BIG-IP instance and leverages CloudInit to configure the instance.
  - These templates show use of:
    - CloudInit for BIG-IP instance configuration
    - BIG-IP integration with EC2 to track pool members in an Autoscale group
    - Auto-scaling of BIG-IP using CloudWatch and EC2 Autoscale
    - Example breakdown of deployment uses CFTs
- A python script to allow easy deployment of these CloudFormation scripts for ease-of-use purposes (deploy_stack.py)
- A JMeter script which can be used to test scale out/in for the resources deployed

Usage

See README.md in the code examples. The README discusses how to deploy the CFTs, and how to use JMeter to force a scale out event.

Improving these examples

The following elements might be improved in these examples to create a more robust solution:

- Adding additional subnets for various resources in the application stack in order to better leverage security constructs in EC2 Virtual Private Clouds such as security groups, network ACLs, and routing.
- Updating the deployment to span multiple availability zones.
- Reorganized the provided CFTs for better composability in larger deployments.

Further documentation

The 12.0 documentation for BIG-IP VE in AWS covers the use of CloudInit and pool member auto-scaling.

https://support.f5.com/kb/en-us/products/big-ip_ltm/manuals/product/bigip-ve-setup-amazon-ec2-12-0-0.html

<dev central link…>