F5 and AWS: Advanced Application Delivery Services in the Cloud

White Paper

Introduction

Amazon Web Services (AWS) has become the largest and most prevalent provider of public cloud Infrastructure-as-a-Service (IaaS). Organizations can now build, test, and deploy entire application stacks without purchasing or reconfiguring on-premises infrastructure. Production applications can benefit from advanced application delivery services such as a web application firewall (WAF), SSL acceleration, content-based routing, load balancing, DDoS mitigation, identity and access management, and protocol agility—which make these key applications go faster, smarter, and safer. For cloud-native AWS deployments, F5 solutions enable simple and easy delivery of these application services to AWS cloud environments.

What is Amazon Web Services?

As a provider of IaaS, Amazon Web Services (AWS) offers a rented compute infrastructure sufficient to develop applications. Without spending any money on capital expenses or upkeep, a complete application suite can be developed and deployed. Besides providing compute infrastructure like those found on premises, AWS delivers application services, automatically scales infrastructure, and hosts applications in many locations. Because of these additional capabilities, best practices for AWS are different than best practices for a customary data center. To better appreciate the differences between AWS and a traditional data center, it's important to understand how AWS delivers its services.

Infrastructure services

To gain the most benefits from AWS, let's look at several important attributes of its IaaS offering, such as geographical distribution, computing instances, networking, and storage concepts.

Geography

Understanding AWS begins with a physical description of how resources are distributed. At the highest level, AWS has several regions, each of which covers a large geographic area, such as the eastern part of the U.S. Each geographic region contains several availability zones, which are so named because they are designed to be isolated from failures with each other. A failure in one availability zone (AZ) will not cascade to another AZ within a region. While each AZ is isolated, AZs within a region are connected via low-latency network links. For availability purposes, design patterns exist for redundant capability across multiple AZs within a region.

Compute

Compute instances begin as a software disk image, known as an Amazon Machine Image (AMI). Each AMI is immutable, meaning that it is read-only by the computer using it. Booting a compute instance against an AMI creates a running virtual machine in the cloud. The virtual hardware characteristics of the compute instance can vary, such as the number of CPUs and RAM available to the instance.

Networking

Each compute instance exists within a Virtual Private Cloud (VPC), which equates roughly to a LAN in the physical world. Each VPC is logically separated. A VPC can consist of multiple subnets with implicit routing between subnets. Public subnets are routable from the Internet, while private subnets are available only internally. To deploy a complete application, one or more compute instances are deployed within a VPC. Communication across VPCs can take place when a compute instance or load balancer has an assigned DNS entry.

Storage

As noted above, each AMI is immutable from the instance using it. For persistent storage, AWS offers many solutions, primarily Simple Storage Service (S3) and Elastic Block Storage (EBS). S3 is an object storage service where an object can be stored and accessed by name. The object can range in size from zero bytes to 5 TB. In fact, each AMI is implicitly an S3 object. Objects cannot be modified, only created and deleted, making them ideal for relatively static content, such as photographs or virtual machine images.

EBS provides storage more akin to traditional storage. A compute instance attached to EBS sees the EBS as a traditional hard disk. Only one running instance at a time may attach to an EBS volume, so EBS cannot be used for shared storage.

Scaling services

Besides delivering key infrastructure components, AWS provides additional services that enable application scaling. Because AWS can quickly provision infrastructure resources, Amazon has developed solutions that allow for auto scaling and load balancing of those resources.

Load balancing

AWS provides an Elastic Load Balancing (ELB) service. Initially, ELB referred to a simple load balancer operating at layer 4 that spread traffic across multiple healthy nodes in a pool. The pool could span multiple availability zones, creating automatic redundancy in case of failure in one availability zone. While this load balancer provides some basic network layer 7 capabilities, it primarily operates at layer 4, simply routing TCP traffic in a round-robin fashion to the nodes in the pool. Health checks determine which nodes are available and therefore are candidates for traffic. AWS now refers to this initial load balancer as the Classic Load Balancer to differentiate it from the new Application Load Balancer (ALB).

AWS introduced the ALB to provide additional layer 7 capabilities such as content-based routing; support for HTTP/2, containers, and WebSockets; and more robust health checks, including HTTP return codes. The two AWS load balancers operate at different layers in the network stack and therefore provide complementary capabilities.

Auto scaling

Since AWS has standby capacity available, it can provide the option to scale nodes within a pool. The AWS CloudWatch service monitors the performance metrics of each node within a pool. When predefined thresholds—such as CPU utilization exceeding 60% for 5 minutes—are crossed, another node is provisioned. Inversely, when a lower threshold has been crossed, a node is removed. The designer can determine the maximum and minimum number of nodes in a pool and the thresholds that trigger node instantiation or node removal. Using auto scaling behind a load balancer enables the node pool to expand or contract as needed based on load.

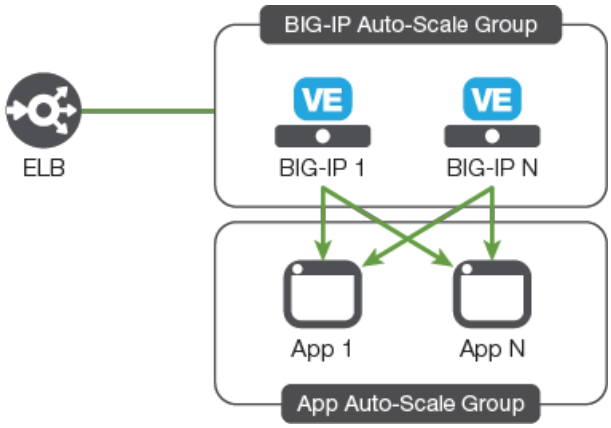

Figure 1: AWS auto scaling

F5 application delivery services

While applications handle business rules and logic, they often lack the hardening required for at-scale production deployment, management, and operation. F5 solutions enable applications to go faster, smarter, and safer by providing the advanced application delivery services detailed in the table below.

F5 Advanced Application Delivery Services

Faster	*Network and Transport Optimization* • A configurable TCP stack that can be optimized to deliver across WAN and cellular networks • An HTTP/2 gateway that brings the advantages of additional compression and request multiplexing without altering the back-end infrastructure

	Application and Data Optimization • Selective image on-the-fly optimization dependent on detected network or client characteristics • WAN acceleration over SSL encrypted tunnels with adaptive compression and TCP optimization
Smarter	*Data Path Programmability* • Complete programmatic control of application traffic, including the ability to read, write, and inspect all aspects of application data • Event-driven and comprehensive language *Control Plane Programmability* • The ability to modify configuration in response to events such as changes to server load, application behavior, or infrastructure • Fully autonomous or external API-driven triggers *Application-Level Monitoring* • Advanced application health checks (using a multiple-step monitor) • Multi-level health checks (such as checking that both the database and application are available) • Non-HTTP health checks (such as SIP, Microsoft Windows SQL Server, and FTP) • Advanced algorithms to better distribute traffic to the servers functioning best *Global Availability* • Application availability across a heterogeneous mix of different cloud providers or data centers • Integration with BIG-IP advanced monitors • DNSSec support
Safer	*Advanced Network Firewall Services* • Decisions about traffic control using criteria beyond simply IP:port:protocol, such as geographical location or endpoint reputation • HTTP protocol validation • Day and time schedules *Web Application Firewall Services* • Comprehensive tools to identify web application threats and block malicious traffic • Outbound data loss prevention (DLP) services *Access and Identity Services* • Advanced authentication services such as two-factor tokens, CAPTCHA, or geographical restrictions • Client certificate checking and endpoint inspection • SAML service provider (SP) and identity provider (IdP) services *Denial-of-Service (DoS) Mitigation* • Proactive bot defense • Layer 7 DoS detection and mitigation *SSL and Encryption* • SSL decryption, traffic inspection, and re-encryption • Offloading of SSL workloads from compute node resources

Advanced application delivery services enable applications to perform at a higher level, while being available and more secure. These services can exist at a strategic point of control independent of each application. Decoupling the services from the application business logic allows the applications to meet business needs without burdening development teams with infrastructure, management, and performance concerns. A strategic point of control also allows issues of governance to be handled uniformly and independently of each application.

Putting it all together: F5 and AWS

By providing cloud-native integration with AWS infrastructure, F5 allows organizations to get the most of their AWS deployments, empowering their applications with better performance, higher availability, and stronger security. In the following section, we'll examine how AWS and F5 work together.

Bootstrapping

When a server instance boots from a generic image, it often makes sense to change parameters or set configurations, such as the hostname and IP address. On client machines, these parameters are set via DHCP, but setting server parameters via DHCP can often cause problems. Beyond network settings, sometimes a particular instance requires specific software packages or certain software configurations.

In the case of a BIG-IP deployment, an administrator might want to automate the configuration of each of the modules, such as the virtual servers and pool configurations for BIG-IP Local Traffic Manager (LTM), specific WAF configurations for BIG-IP Application Security Manager, or perhaps firewall settings for BIG-IP Advanced Firewall Manager. The same issues face anyone installing a server instance: the base image needs additional configuration to function correctly.

AWS offers two primary approaches to configuring an instance: creating a new AMI, or using cloud-init to modify a server during the boot process. Creating a new AMI is more appropriate for changes common among multiple instances that are less likely to be updated often. In contrast, cloud-init is more appropriate for changes that impact fewer instances and have a shorter life expectancy.

AMI

For changes that are expected to persist for a longer period of time—and for changes common to multiple instances—a good approach is to create a new AMI by booting a machine from an AMI similar to the desired configuration. After the administrator has made the changes necessary to the running instance, the instance is stopped and a new AMI is generated and registered with AWS. All future instances booting from this AMI will have the changes already applied. Since this approach makes changes effectively permanent—and since generating the new AMI can consume time—changes baked into the AMI are generally those that will last for a long time and are usable across multiple instances. Another reason for using AMI is that it enables faster boot times, since some cloud-init processes can be time-intensive.

Cloud-init

Necessary changes that are not a good fit for incorporating into a new AMI are good candidates for cloud-init, which essentially enables a startup script whenever the instance boots. Using cloud-init allows for simple and instance-specific changes to be embedded into the instance.

Disadvantages of cloud-init include that the configuration changes, such as package installations, must be run at boot time, causing the boot to take longer. A long boot time has real impact in auto-scaling scenarios where an elongated boot time could make auto scaling ineffective. For these reasons, changes that take a lot of time should be included in a new AMI instead of making the changes via cloud-init.

Managing configuration can also be cumbersome when a change can be used across several, but not all instances. For example, suppose that a particular BIG-IP deployment is used in an auto-scale group with a specific virtual server configuration. A single AMI could serve for those machines and a different AMI could serve for other BIG-IP machines in another auto-scale group. Using a single AMI for each auto-scale group ensures that only changes specific to each host are necessary within the cloud-init process. Any changes common to the group can be embedded into the AMI. The disadvantage of this approach is that it requires an update to the AMI for each change common to all machines.

Auto scaling

Applications deliver a capability, generally to multiple users simultaneously. As the application becomes more successful, it can exceed the capacity of the computer on which it runs. Once the application needs exceed those of its computer, options for increasing capacity need to be evaluated. There are three generic approaches to scaling: scaling up, pipelining, and scaling out.

Scaling up

Scaling up is the simplest approach to increasing capacity because it merely involves replacing the existing computer with a faster one. By installing a faster computer, all aspects of the application, and any other services on the computer, become faster with no changes necessary to the application or infrastructure. The disadvantage of scaling up is that costs tend to increase exponentially with performance increases, leading to a point of diminishing returns. Once a threshold is crossed, scaling up becomes cost-prohibitive.

Pipelining

Pipelining is the result of decomposing the workload into sequential steps, similar to an assembly line. When different computers can each work independently on each step, the overall throughput can be increased. However, pipelining only increases throughput, and it often does it at the expense of latency. Put another way, pipelining can increase overall performance but can decrease performance for a single user or a single request. The other disadvantage of pipelining is that it requires a deep understanding of the decomposable workflow, and for the infrastructure to match that understanding. It tightly couples the infrastructure decisions to the business logic, which is the exact opposite of what many organizations are trying to do.

Scaling out

Scaling out involves leaving the application and the computer alone, and instead choosing to spread requests evenly across a pool of servers. Since applications generally process several independent requests simultaneously, requests can safely be spread out across a pool of identical application servers. Scaling out has the added benefit of redundancy in that a failure of any pool member will not cause an outage of the entire application. The disadvantage of scaling out is that it requires complex orchestration external to the application in order to ensure that the requests are balanced across the nodes in the pool.

AWS auto scale

AWS auto scale uses a scale-out approach to increasing capacity for applications that need it. The CloudWatch service monitors nodes in a pool. When nodes cross predefined thresholds, CloudWatch will automatically start up new nodes or shut down nodes in the pool as appropriate. With the BIG-IP platform, this process can take place in one of two ways: by altering the number of BIG-IP instances or by altering the number of nodes in a pool behind a single BIG-IP instance. The difference between the two approaches is a function of what is scaled: either the BIG-IP instance or a pool.

In the first scenario, a BIG-IP pool sits between a pair of ELB devices. The first ELB device controls instantiating and terminating BIG-IP members, while the second ELB device is the sole entry in a server pool for each of the BIG-IP instances. This approach makes sense when the BIG-IP instance is providing advanced application delivery services, such as SSL termination or acting as a web application firewall. The first ELB device performs the load balancing while also growing or shrinking the pool as appropriate.

In the second scenario, the number of back-end pool members grows and shrinks via CloudWatch, but the BIG-IP instance performs the load balancing. The BIG-IP instance communicates with AWS to discover nodes being added or removed from the pool. This approach makes sense when using advanced load balancing features, such as the iRules scripting language, or directing requests based on URL or content. In these cases, a single BIG-IP instance is sufficient to manage the load of servers in the back-end pool.

Security and IAM

The BIG-IP instance must interact with the AWS infrastructure in at least two scenarios. First, a multiple-zone AWS deployment requires altering the IP address behind an AWS elastic IP. Second, a BIG-IP instance needs visibility into pool members added and removed by the AWS CloudWatch service, which scales servers up and down within a pool. Each interaction with the infrastructure takes place via API calls, and just like any other software making API calls, the BIG-IP instance must authenticate to AWS. Generally, there are two approaches to authenticating to AWS: through credentials or IAM roles.

Credentials

The simplest approach to authenticating is by including the appropriate credentials with the API call. AWS credentials consist of an access key and a secret key, which roughly correspond to a username and password. The administrator generates the credentials, which the developer then embeds within the application. This gives the application access to the appropriate API calls.

While simple, embedding credentials into an application carries security risks. Unless the developer secures the credentials in the application, other people or software could recover them and use them in malicious ways. This approach also makes it difficult to alter the credentials without also altering the software. While using credentials is a reasonable approach for quick testing, a production solution should use another approach to authentication. This is why AWS best practices recommend against using stored credentials in an application.

Identity access management

A more secure approach to authenticating for API calls is the use of IAM roles. AWS Identity and Access Management (IAM) enables users to manage access to the AWS infrastructure. Any compute instance, such as a BIG-IP machine, can be assigned an IAM role that authorizes a specific set of capabilities. When the instance starts, IAM generates a temporary set of credentials for the instance. Those credentials last while the instance is functioning and enable only the API capabilities specified. When configured with an IAM role, the BIG-IP instance does not store credentials, but instead has access only to the infrastructure APIs necessary, thus providing more security than credential-based authentication.

Multi-zone availability

As mentioned earlier, AWS data centers exist in geographical regions, each of which can exist in an availability zone (AZ). Each AZ within a region shares nothing with other AZs: no shared power, networking, or buildings. In fact, each AZ is geographically separated from the others within a region. Because of the separation between zones, AWS subscribers can be confident that an event impacting one AZ will not impact another AZ. In other words, as a rule, at most one AZ within a region should be unavailable at any moment in time. This means that any service deployed across two or more availability zones should be continuously available.

The BIG-IP platform supports high-availability across AWS AZs using an AWS elastic IP, which is an IP address not intrinsically associated with a compute instance. Instead, the IP address can be dynamically forwarded to a private IP address of a running compute instance. To enable multi-zone high availability, identical sets of BIG-IP instances and application servers are each placed in their own AZ. Initially, the elastic IP is assigned to one of the BIG-IP instances. Connections are established from each client to the elastic IP which in turn forwards them to the private IP address on one of the BIG-IP instances. Should a failure occur, the other BIG-IP instance will claim the responsibilities by placing an API call to AWS, requesting that the elastic IP address be forwarded to it.

Elastic Load Balancer

By integrating with the ELB, the BIG-IP platform can provide application services that integrate seamlessly with AWS capabilities such as multiple AZs and auto scaling BIG-IP nodes.

Placing the ELB in front of a BIG-IP instance simplifies deployment across multiple AZs, because the ELB can seamlessly balance traffic to the individual application stacks within each AZ where a BIG-IP instance is providing application services. This approach simplifies load balancing across multiple AZs.

When elasticity of BIG-IP instances is needed, an ELB with auto scale can automatically scale up and down a pool of BIG-IP virtual appliances, providing application services such as a web application firewall, identity management, or SSL termination. Using an ELB sandwich, traffic is routed to the first ELB which balances and auto scales traffic to a pool of BIG-IP instances. To simplify configuration across the BIG-IP pool, each BIG-IP instance has a single ELB address in the server pool. The second ELB then routes traffic to the downstream server pool.

Various combinations of ELB and BIG-IP topologies provide auto scaling, availability, and application services that are unavailable to either alone. By exploiting the advantages of both ELB and the BIG-IP platform, the architect can provide the level of services needed for a particular deployment.

Cloud Formation Templates

To enable repeatable and scripted deployments, AWS provides Cloud Formation Templates (CFTs), which simplify both deployment and ongoing management. After the creation of a CFT for the desired service or application architecture, AWS can use it to provision an application quickly and reliably. CFTs are particularly useful in DevOps environments, allowing teams to easily create repeatable processes for testing and production deployment.

F5 not only supports using CFTs to deploy BIG-IP instances, but provides several reference CFT files for typical BIG-IP deployments.

Adjusting the parameters in the reference CFT files enables scripted deployments of BIG-IP solutions for different scenarios, including automatically scaling BIG-IP instances or back-end servers behind BIG-IP instances, as well as more complicated scenarios. By automating repeatable deployments within AWS using CFTs and F5 solutions, complex application environments can be deployed quickly and with little work.

Documentation

Of course, technology is of little use if it cannot be leveraged fully. To that end, F5 provides extensive documentation. Documentation is available for the BIG-IP platform in general, and for the specifics of a BIG-IP instance within AWS. A good starting point for any question is at Ask F5.

The documentation tab provides information about specific BIG-IP modules as well as an entire section on AWS integration. The AWS portal provides a searchable interface for documentation, articles, community, and resources from getting started to complex deployment scenarios.

For those questions not answered by documentation, the F5 DevCentral community is ready to provide answers and assistance.

Conclusion

The march toward public cloud adoption is no longer a fad, but an enduring trend in IT. Amazon Web Services, as the world's largest and most comprehensive provider of public cloud services, gives organizations the ability to build, test, and deploy applications without any on-premises equipment. F5 has made its advanced application delivery services available as part of the AWS ecosystem—and has configured them to help apps go faster, smarter, and safer in AWS cloud environments.

F5 Networks, Inc.
401 Elliott Avenue West, Seattle, WA 98119
888-882-4447 f5.com

Americas
info@f5.com

Asia-Pacific
apacinfo@f5.com

Europe/Middle-East/Africa
emeainfo@f5.com

Japan
f5j-info@f5.com