HA & Scaling

Regional and Global AWS Architecture

Global Components - Global Svc Location/Discovery, Content Delivery+optimization, global health checks/failover Regional Components - Regional entry point, scaling/resilience, app svc's/components

Evolution of Elastic Load Balancer (ELB)

Currently 3 types of ELBs available in AWS split between v1 (avoid this) and v2 (use this). "ELB" refers to all 3.

- Version 1: Classic Load Balancer (CLB), 2009. Not really layer 7, only one SSL per CLB
- Version 2:
    - v2-1. Application Load Balancer (ALB): HTTP/HTTPS/WebSocket. Layer 7 device
    - v2-2. Network Load Balancer (NLB): TCP, TLS, UDP. For apps that don't use HTTP/HTTPS, like email or SSH servers
- Version 2 is faster, cheaper, supports target groups and rules

Elastic Load Balancer (ELB)

Job of a Load Balancer is to accept connections from a user and distribute that connection to the backend of the app

- "dual stack" means using both IPv4/6
- config'd to run in 2+ AZ's. 1+ nodes are placed into a subnet in each AZ and scale with load
- Each ELB confg'd with an A record DNS name which resolves to the ELB nodes. Requests distributed to nodes, nodes scale within AZ EXAM: ELB can be internet-facing (given public and private IP Addresses) or internal (given only private IP addresses) EXAM: Internet-Facing LB can still access both public AND private EC2 instances EXAM: /27 or larger subnet is minimum for load balancer. /28 is 2nd most correct answer

ELB - Cross-Zone Load Balancing

- Load Balancer Nodes can send requests across an AZ to any other registered instance in other AZs for further balancing
- Comes enabled as default

ELB EXAM

EXAM: ELB is a DNS A Record pointing at 1+ Nodes per AZ EXAM: Nodes can scale per subnet EXAM: Internet-facing nodes have public IPv4 IPs EXAM: Internal-facing nodes have only Private IPs EXAM: EC2 instance does NOT need to be public to work with an internet-facing LB EXAM: Listener Config controls WHAT the LB does EXAM: ELBs required 8+ free IPs per subnet, and at least /27 subnet to allow scaling

Application Load balancing (ALB) vs Network Load Balancing (NLB)

When to pick ALB vs NLB.

- Remember, always avoid Version 1 Classic Load Balancer -- does not scale.

Version 2, APPLICATION Load Balancers:

- Layer 7 LB's. Listens on either HTTP and/or HTTPS
- Doesn't listen to any other Layer 7 protocols (SMTP, SSH, Gaming, etc)
- Can't listen to TCP/TLS/UDP
- Can't do end-to-end unbroken SSL encryption as the request is terminated on ALB and new connection is made from ALB -> App NOTE: if you need unbroken encryption, you need to use a Network Load Balancer
- ALBs using HTTPS must have an SSL cert on that LB
- ALBs are slower than NLB as they're more complex
- As they are Layer 7, they can do health checks at Layer 7
- "Rules" - Direct connections which arrive at a listener, rules processed in priority order, there is a final Default Rule as a catch-all
- "Rule Conditions"
- "Actions" - follows rules NOTE: if you need unbroken encryption, you need to use a Network Load Balancer

Version 2: NETWORK Load Balancers

- Layer 4 LBs: TCP, TLS, UDP, TCP_UDP
- No visibility or understanding of HTTP/S; can't see headers, cookies, session stickiness
- NLBs are super fast, about 25% of ALB latency
- Good for SMTP, SSH, game servers which don't use web protocols, financial apps which don't use web protocols
- health checks not app aware, they just check ICMP/TCP handshakes
- can have static IPs which is useful for whitelisting
- can forward TCP to instances for UNBROKEN ENCRYPTION
- used for PRIVATE LINK to provide services to other VPCs

EXAM: Need unbroken encryption? Network Load Balancer EXAM: Need highest performance LB? NLB
EXAM: Need Private Link? NLB EXAM: Static IP? NLB

EXAM--

ALB VS NLB

- Need unbroken encryption? Network Load Balancer
- Static IP for whitelisting? NLB
- Best performance? NLB
- Operate on non-HTTP/S? NLB
- Private Link? NLB
- Otherwise... ALB

EC2 Launch Configuration and Templates

Both alllow you to define config of an EC2 instance in advance

- Including.... -- AMI, instance type, storage & keypair, networking and SGs, userdata and IAM role
- Configs are NOT editable; defined one. But, Launch Templates have versions
- LT came after LC so it has newr features like T2/T3 unlimited, placement ggroups, capacity reserveations, elastic graphics NOTE: AWS recommends using Launch Templates over LCs
- LC's have one use: only used as part of Auto Scaling groups

Auto Scaling Groups

Auto Scaling group contains a collection of Amazon EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management. An Auto Scaling group also enables you to use Amazon EC2 Auto Scaling features such as health check replacements and scaling policies. Both maintaining the number of instances in an Auto Scaling group and automatic scaling are the core functionality of the Amazon EC2 Auto Scaling service.

- "sefl healing" for EC2
- uses launch templates or configs to launch all scaled instances
- Auto-scaling group has three important values: Minimum size, desired capacity, maximum size (eg 1:2:4), called min/desired/max or x/y/z
- Auto-scaling keeps running instances at the DESIRED capacity by provisioning/terminating instances
- Scaling Policies indrease or decrease automatically the desired capacity based on metrics like CPU load

Auto-Scaling Groups - Scaling Policies

Scaling policies are rules. Three ways you can scale auto-scaling groups:

1. Manual Scaling. Manually adjust min/desired/max
2. Scheduled Scaling. Time-based adjustment. Eg. 9am, increase to 10 instances, 5pm, decrease to 2 instancesß
3. Dynamic Scaling (3 subtypes). Rules which react to something and change

- a. Simple Scaling. Often a pair of rules: "CPU above 50%, +1. CPU below 50%, -1."
- b. Stepped Scaling. Bigger increase of decrease of number of instances based on difference (step): generally use Stepped scaling over Simple unless simplicity is the priority
- c. Target Tracking: Let's you define an ideal amount of something: Eg.. Desired aggregate CPU to stay at 40% Cooldown Periods: How long to wait at the end of a scaling action before another can start. Prevents rapid scaling up and down and saves costs.

ASG - Health

ASG performs healths checks using EC2 status checks

- Self healing: If ASG detects downed instance, it will replace the downed instance with a newly provisioned one

ASG - ASG + Load Balancers

- Uses Target Groups
- Can use Load Balancer health checks instead of EC2 health checks; application aware health checks (which ec2 health checks are not)

ASG - Scaling Processes

- LAUNCH and TERMINATE (can SUSPEND / RESUME) any process, like suspending the Launch process to not scale out if event takes place
- AddToLoadBalancer - Add to LB on launch
- AlarmNotification - Accept notification from CloudWatch
- AZRebalance - Balances instances evenly across all AZs
- HealthCheck - health checks on/off
- ReplaceUnhealthy
- ScheduledActions
- Standby or InService

ASG - Final Points

EXAM: Autoscaling groups are free; only the resources created are billed (use cooldowns to avoid rapid scaling and increased costs) EXAM: Think about using more smaller instances for granularity - higher control and more cost-effective EXAM: Use with Application Load Balancers for elasticity; abstraction of the instances because the LB is connected to the ASG and not the actual EC2 instances EXAM: ASG defines WHEN/WHERE, Launch Templates/Configs define WHAT

ASG - Scaling Policies

Scaling policies are NOT REQUIRED on ASGs

- Manual - When you set Min / Desired / Max. Best for Testing or when urgent
- Step / Simple scaling: you choose scaling metrics and threshold values for the CloudWatch alarms that trigger the scaling process. You also define how your Auto Scaling group should be scaled when a threshold is in breach for a specified number of evaluation periods
    - Step / Simple scaling require you to create CloudWatch alarms for the scaling policies. Both require you to specify the high and low thresholds for the alarms
    - Step / Simple scaling require you to define whether to add or remove instances, and how many, or set the group to an exact size
- Scaling based on SQS - ApproximateNumerOfMessagesVisible

ASG - Lifecycle Hooks

Lifecycle hooks enable you to perform custom actions by pausing instances as an Auto Scaling group launches or terminates them. When an instance is paused, it remains in a WAIT state either until you

1. complete the lifecycle action using the complete-lifecycle-action command or 2. the CompleteLifecycleAction operation or 3. until the timeout period ends (one hour by default) then CONTINUE or ABANDON.

- Configure Custom Actions on instances during ASG actions (launch or terminate transitions)
- Can be config'd with EventBridge or SNS Notifications

ASG - Health Check Comparison: EC2 VS ELB

Amazon EC2 Auto Scaling can determine the health status of an instance using one or more of the following:

- Amazon EC2. To identify hardware and software issues that may impair an instance. This is the Default for ASG
- Elastic Load Balancing (ELB). Disabled by default. ELB health checks can be application aware (Layer 7)
- Custom Health Checks. Instances marked Healthy/Unhealthy by an external system

EXAM: Health Check Grace Period (default 300s); a delay before starting health checks which first allows system to launch, bootstrap, application start

Elastic Load Balancer - SSL Offload & Session Stickiness

Three ways a load balancer handles secure connections: Bridge, Pass Through, Offloading

- Bridging (default). SSL encrypt/decrypt happens at ELB, but instances still need SSL certs and compute required for crpyto operations
- Pass Through. No encypt/decrpyt happens at ELB, instead each instance has SSL Cert installed
- Offload. SSL decrypted at ELB and then only plain text HTTP goes through to instance. Instance doesn't need SSL cert or crypto compute capability

ELB - Connection Stickiness

- With no stickiness, connections are distributed across all in-service backend instances
- With stickiness:
    - a cookie is generated (AWSALB) which locks the device to a single backend instance for a set duration: 1 second to 7 days
    - all the requests from that device go to the same backend instance
    - PROBLEM This might cause an offbalance of the load on the backend servers
    - SOLUTION: Instead of worrying about this, just make sure servers are stateless and state (session) is stored elsewhere like the DB

DEMO - EMB - Seeing Session Stickiness in Action

How session stickiness works with Application Load Balancers Video: https://learn.cantrill.io/courses/1820301/lectures/43242687

ADVANCED DEMO - Architecture Evolution

Stage 1 - Setup the environment and manually build wordpress - https://learn.cantrill.io/courses/1820301/lectures/41301448 Stage 2 - Automate the build using a Launch Template Stage 3 - Split out the DB into RDS and Update the LT Stage 4 - Split out the WP filesystem into EFS and Update the LT Stage 5 - Enable elasticity via a ASG & ALB and fix wordpress (hardcoded WPHOME) Stage 6 - Cleanup

Gateway Load Balancer (GWLB)

Gateway Load Balancers enable you to deploy, scale, and manage virtual appliances, such as firewalls, intrusion detection and prevention systems, and deep packet inspection systems. It combines a transparent

network gateway (that is, a single entry and exit point for all traffic) and distributes traffic while scaling your virtual appliances with the demand.

- At high level, GWLBs have two main components:

1. Endpoints. Traffic enters/leaves via these endpoints
2. The GWLB itself; balances packets across multiple backend instances

- Traffic and metadata is tunneled using GENEVE protocol
- GWLB = network security at scale