EE4221 Cloud Computing Systems

Cloud Scaling and Load Balancing

Eric Wong
City University of Hong Kong

Business Continuity

Outlines

- Business Continuity
- RPO and RTO
- Disaster Recovery (DR) Patterns
- High Availability System
- Load Balancing
    - DNS-based Load Balancing
    - Load balancing algorithms
    - Layer 4 vs Layer 7 Load Balancing
    - SSL Load Balancer
    - Session Persistence
- Elastic Load Balancer and Amazon Route 53
- EC2 Auto Scaling

Business Continuity (BC)

- In production environment, it is important to be able to recover from an unexpected event to maintain business continuity (BC).
  - BC is an integrated and enterprise-wide processes and/or procedures for ensuring continued business operations
  - BC solutions address unavailability and degraded application performance

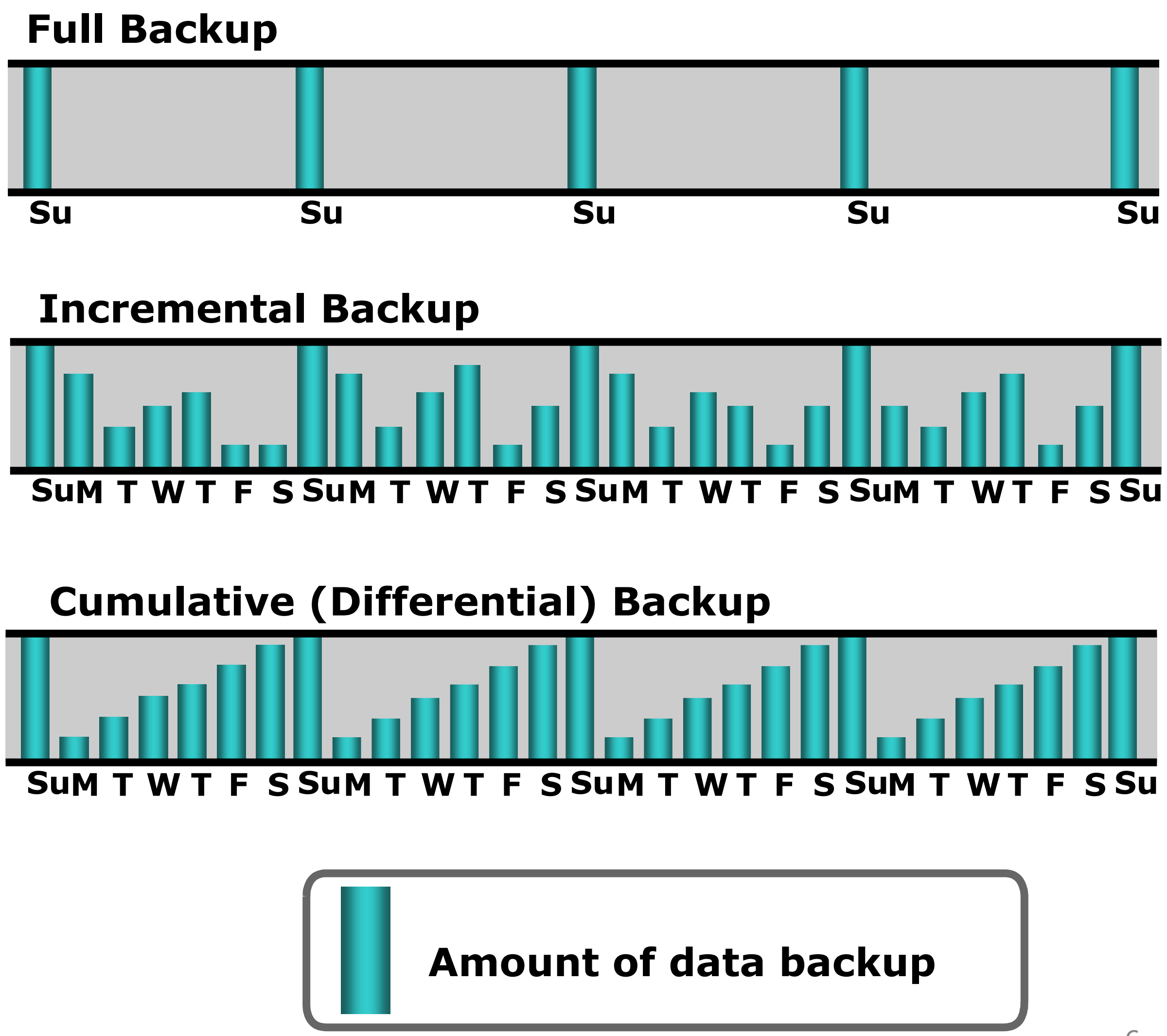
BC Terminologies	Description
Disaster Recovery (DR)	• Coordinated process of restoring systems, data, and infrastructure required to support ongoing business operations in the event of a disaster
Hot Site	• A site where an enterprise's operations can be moved in the event of disaster • DR site infrastructure is up and running all the times
Cold Site	• A site has The IT infrastructure required to support DR but is not activated
Cluster	• A group of servers and other necessary resources, coupled to operate as a single system • Typically, one server runs an application and updates the data, and the other is kept as a standby to take over completely, when required.

Backup and Recovery

- Backup is critical to protecting data and ensuring business continuity.
  - An additional copy of data that can be used for restoration and recovery purposes when the primary copy is lost or corrupted
  - A backup copy can be created by simply copying data (one or more copies) or mirroring data
- Backups are performed for three primary purposes.
  - Disaster Recovery – restore IT infrastructure in the event of a major disaster.
  - Operational Restores – restore data loss or logical corruptions that may occur during routine processes, e.g. emails accidentally deleted
  - Archival - preserve transaction records, email, and other business work products for regulatory compliance
- Main challenge:
  - Data generation rate is much faster than the growth of storage density and durability, e.g. user-created content, logs, emails, etc.

Backup Granularity

- Full backup is a backup of the complete data on the production volumes at a certain point in time. A full backup copy is created by copying the data on the production volumes to a secondary storage device.
- Incremental backup copies the data that has changed since the last full or incremental backup, whichever has occurred more recently. This is much faster (because the volume of data backed up is restricted to changed data), but takes longer to restore.
- Cumulative (or differential) backup copies the data that has changed since the last full backup. This method takes longer than incremental backup, but is faster to restore.

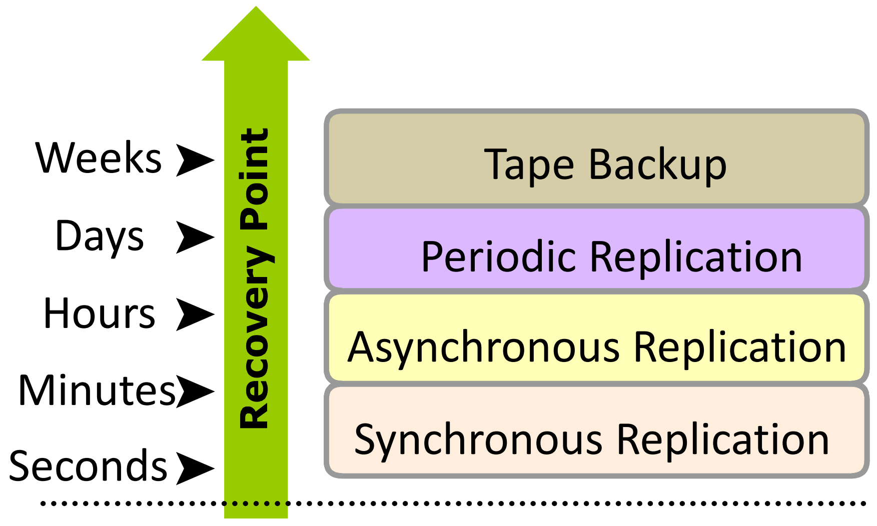


RPO and RTO

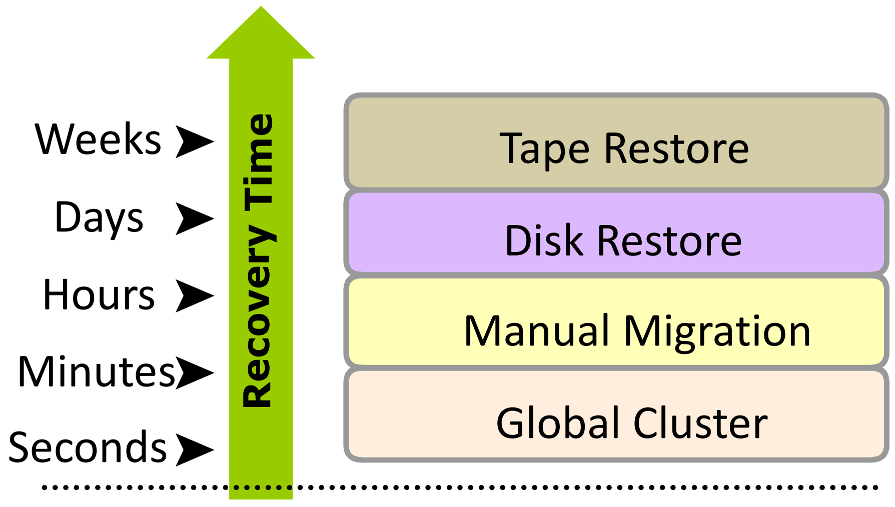
RPO and RTO

- Recovery Point Objective (RPO)
- Point in time to which systems and data must be recovered after an outage
- Amount of data loss that a business can endure
- E.g. an RPO of 1 hour means that you could lose up to 1 hour's worth of data when a disaster occurs.

- Recovery Time Objective (RTO)
- Time within which systems, applications, or functions must be recovered to a working state after an outage
- Amount of downtime that a business can endure and survive



Recovery-point objective



Recovery-time objective

Disaster Recovery (DR) Patterns

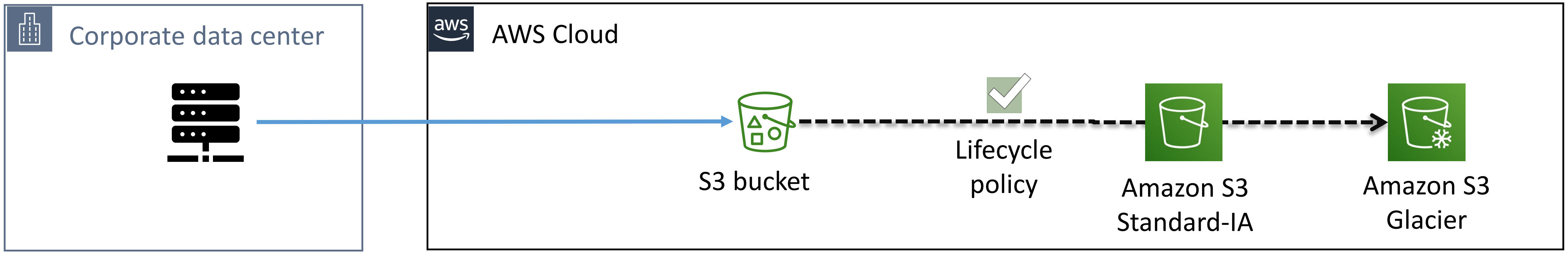
Disaster Recovery Patterns

- Based on RPO and RTO requirements, organizations use different backup patterns/strategies for disaster recovery.
- Organizations often use these four common disaster recovery patterns* in AWS:
    - Backup and restore
    - Pilot light
    - Warm standby
    - Multi-site

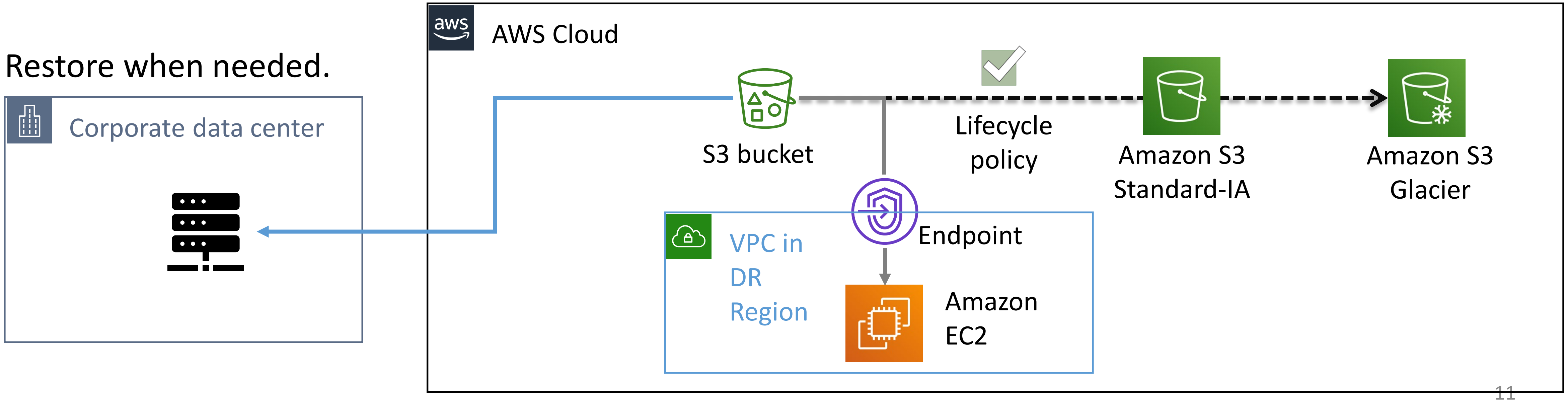
* Some of the patterns offer better cost-effectiveness while others provide a faster RPO and faster RTO but cost more to maintain.

Backup and Restore Pattern

Back up configuration and store data to S3. Implement lifecycle policy to save on cost.



Restore when needed.



Backup and Restore: Checklist

Preparation phase

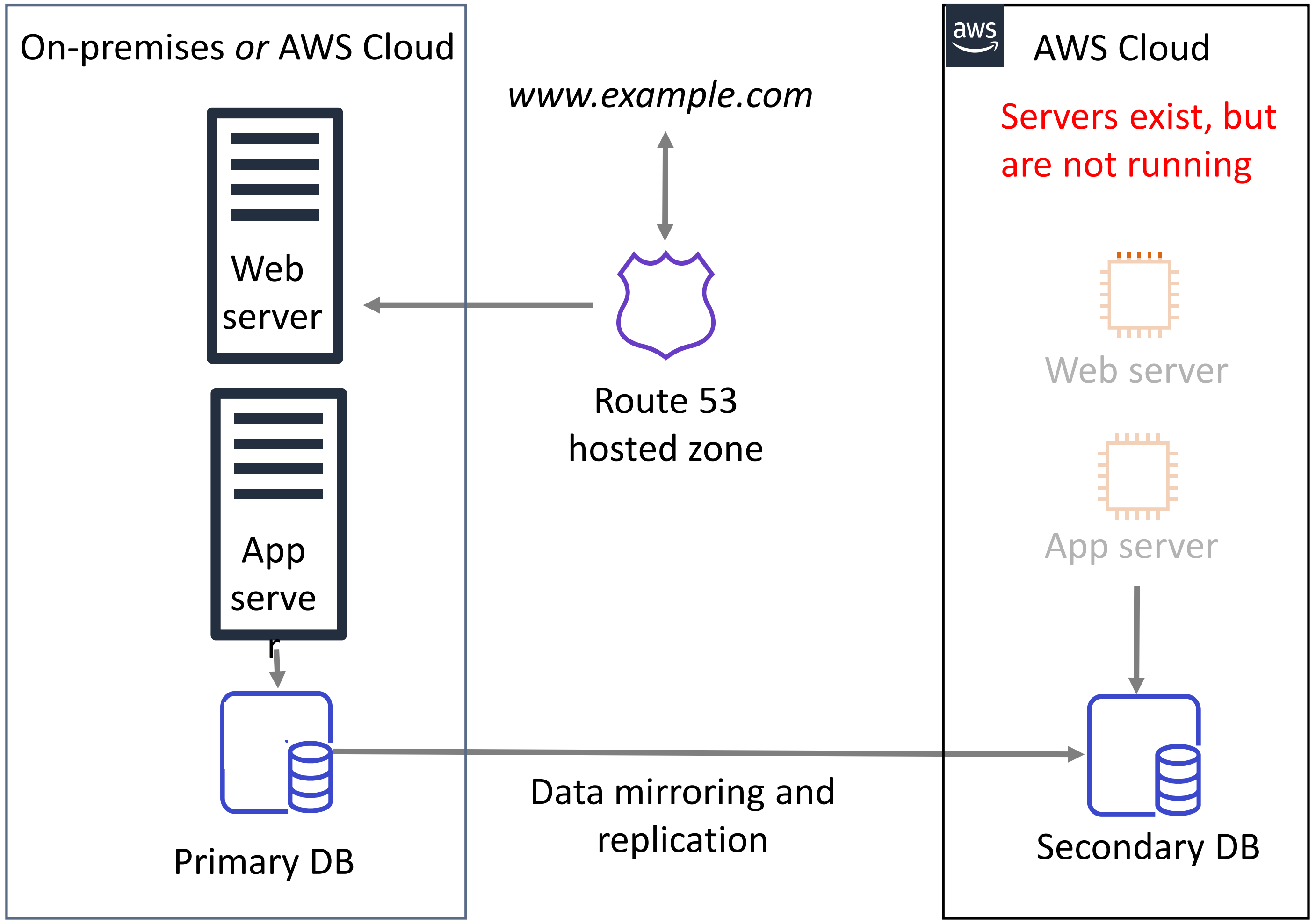
- Create backups of current systems
- Store backups in Amazon S3
- Document procedure to restore from backups
- Know:
  - Which AMI to use, and build as needed
  - How to restore system from backups
  - How to route traffic to the new system
  - How to configure the deployment

In case of disaster

- Retrieve backups from Amazon S3
- Restore required infrastructure
  - EC2 instances from prepared AMIs
  - Elastic Load Balancing load balancers
  - AWS resources created by an AWS CloudFormation stack – automated deployment to restore or duplicate the environment
- Restore system from backup
- Route traffic to the new system
  - Adjust Domain Name System (DNS) records accordingly

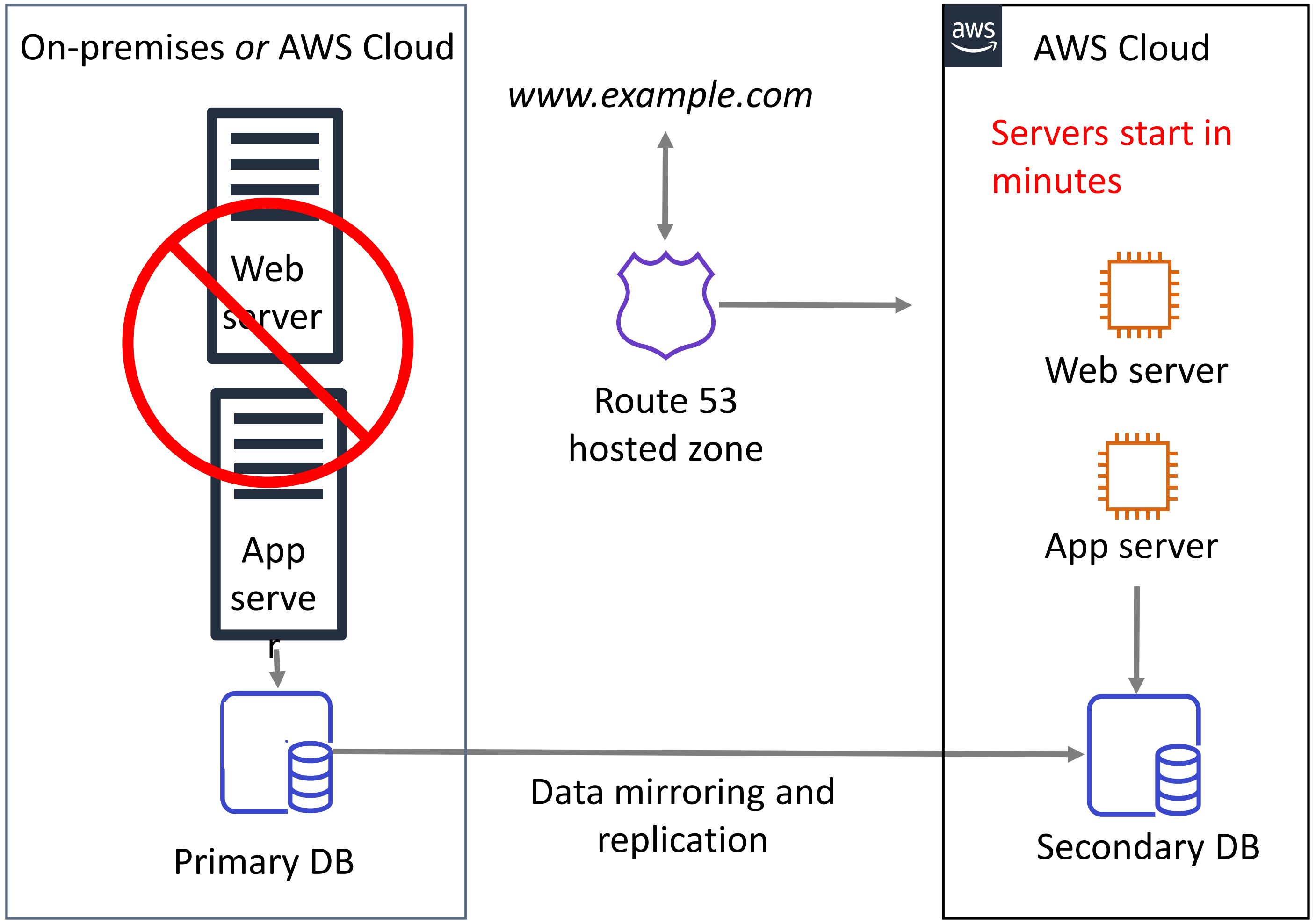
Pilot Light Pattern: Preparation phase

- Replicate the core pieces of the system
- Ensure that all supporting custom software packages are available on AWS
- Create and maintain AMIs of key servers where fast recovery is needed
- Servers exist, but are not running
- Regularly run these servers, test them, and apply any software updates and configuration changes
- Consider automating the provisioning of AWS resources



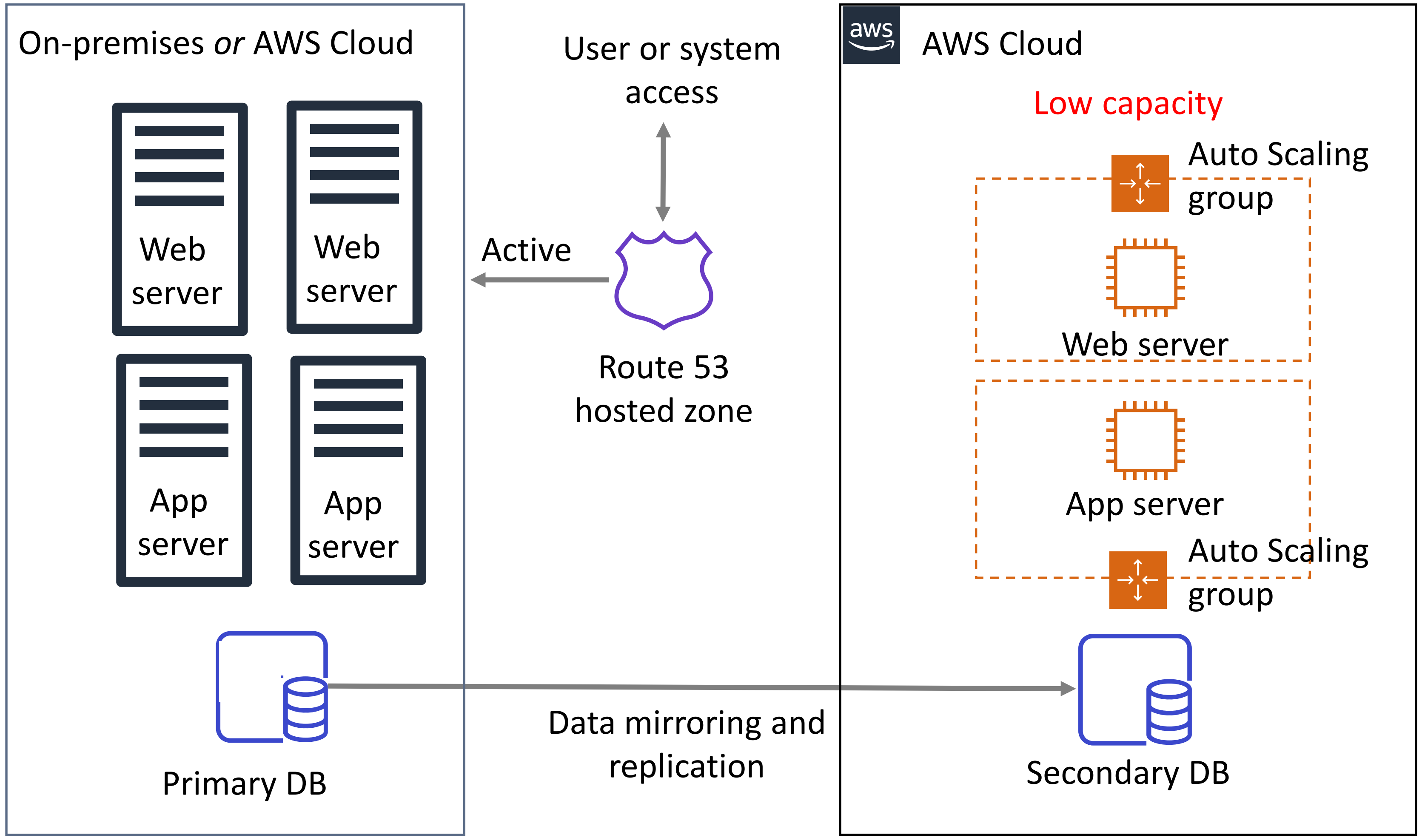
Pilot Light Pattern: In case of disaster

- Automatically bring up resources around the replicated core dataset
- Scale the system as needed to handle current production traffic
- Switch over to the new system in AWS Cloud
- Adjust DNS records to point to new system



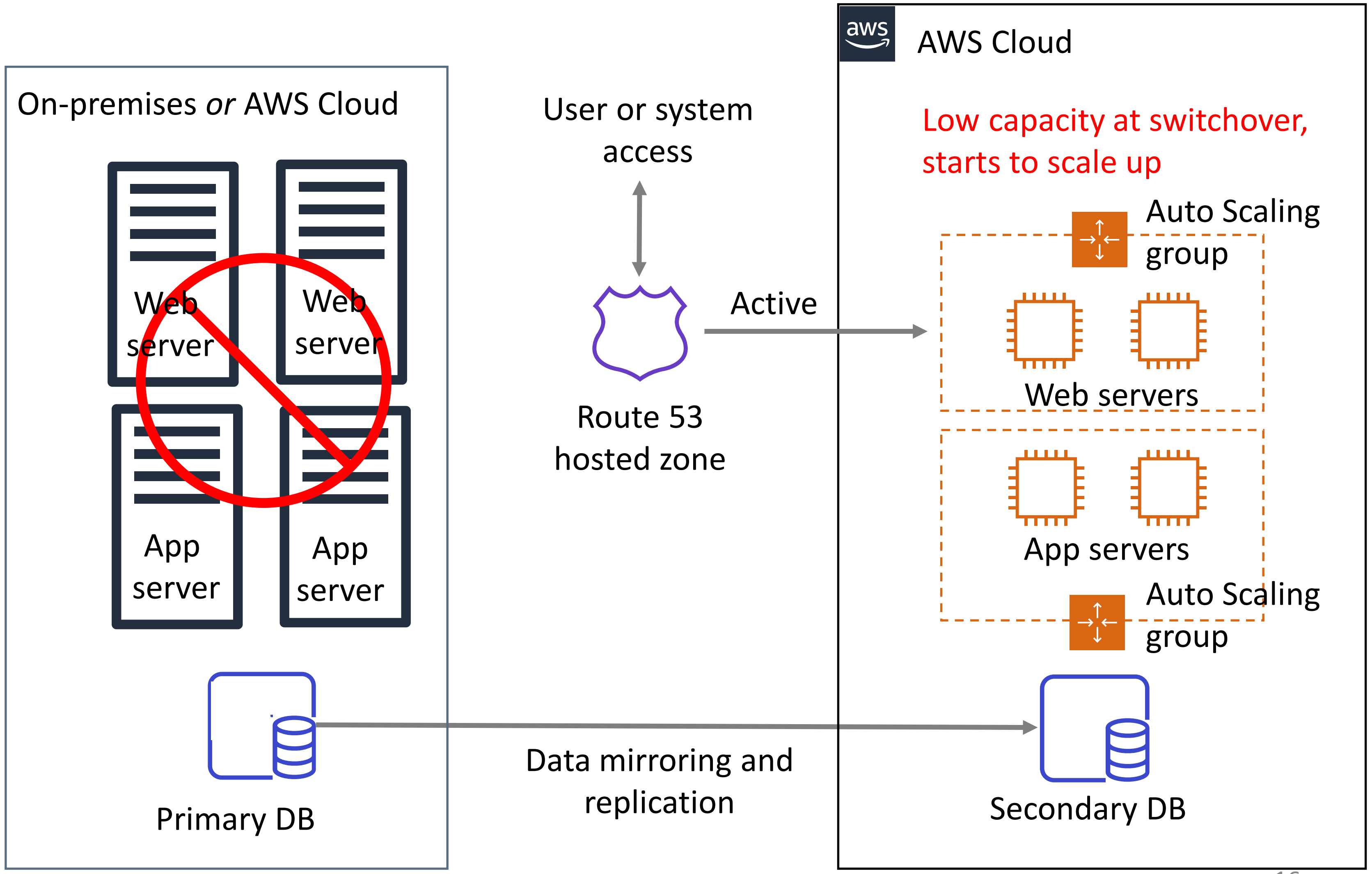
Warm Standby Pattern: Preparation phase

- Similar to pilot light
- All necessary components running 24/7, but not scaled for production traffic
- Best practice: Continuous testing
- Use the DR site for non-production work, such as testing, quality assurance, and internal use



Warm Standby Pattern: In case of disaster

- Immediately fail over most critical production load
- Adjust DNS records to point to new system in AWS
- (Automatically) Scale the system further to handle all production load



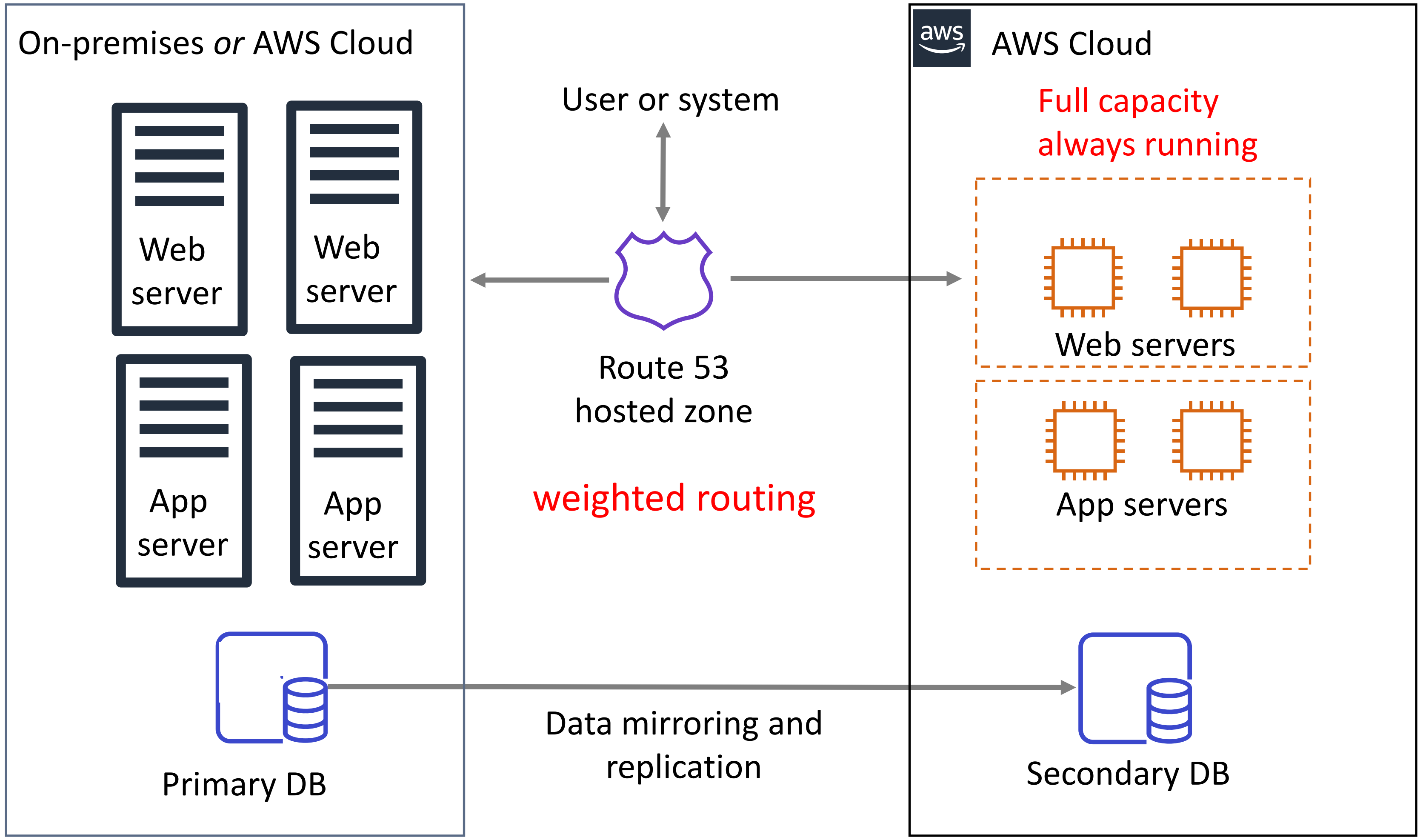
Multi-Site Pattern

Preparation:

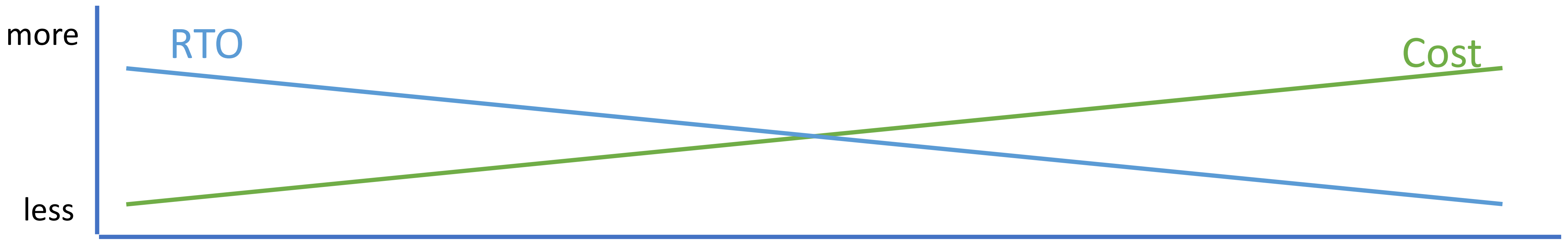
- Similar to warm standby
- Configured for full scaling in or scaling out for production load

In case of disaster:

- Immediately fail over all production load



Summary of Common DR Patterns



Backup and restore — Pilot light — Warm standby — Multi-site

Backup and restore

- Lower-priority use cases

Pilot light

- Meeting lower RTO and RPO requirements
- Core services
- Scale AWS resources in response to a DR event

Warm standby

- Solutions that require RTO and RPO in minutes
- Business-critical services

Multi-site

- Automatic failover of your environment in AWS to a running duplicate

High Availability System

High Availability System

- High availability (HA) is a characteristic of a system which aims to ensure an agreed level of operational performance for a higher than normal period.
    - Have minimized downtime
    - Can withstand some measure of degradation while remaining available
    - Recover from failure or roll over to secondary source in an acceptable amount of degraded performance time
    - Require minimal human intervention

Key Factors for High Availability

- **Fault Tolerance:**
  - The property that enables a system to continue operating properly in the event of the failure. Typical methods include:
    - **Replication** - running multiple identical systems/subsystems
    - **Redundancy** – failover to redundant (spare) components when one failed
    - **Diversity** – have multiple implementations to cope with errors in specific implementations
- **Scalability:**
  - The ability of an application to accommodate growth without changing design.
- **Recoverability:**
  - The process, policies, and procedures related to restoring service after a catastrophic event such as natural disasters.

High Availability (HA) Models

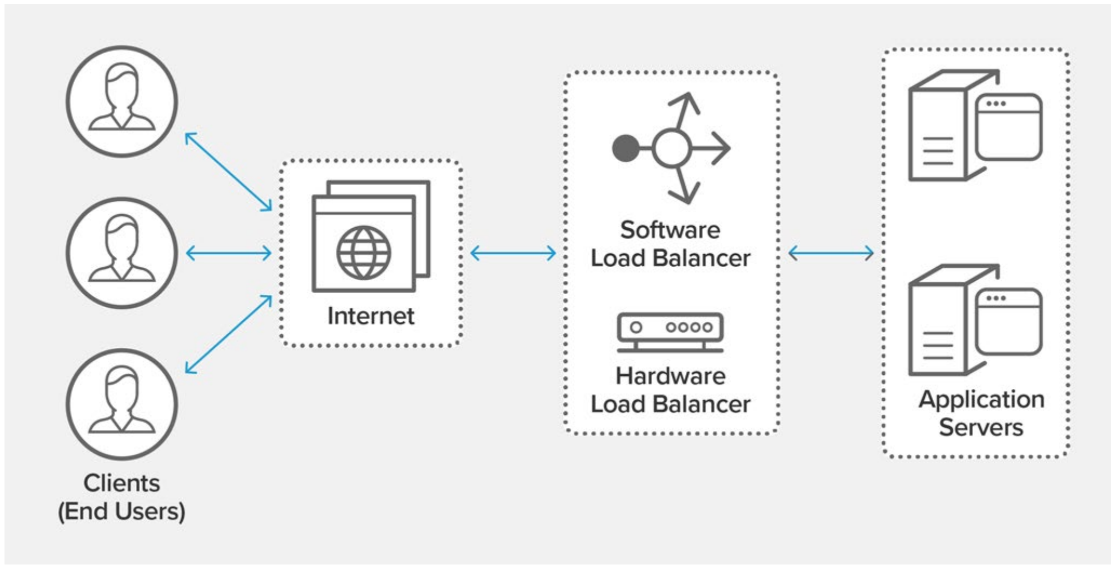
High availability model	Secondary node behavior	Data protection	Failover time
Load-balanced	Both the primary node and the secondary node are active and they process system requests in parallel.	Data replication is bidirectional and is performed based on the software capabilities.	Zero failover time
Hot standby	The software component is installed and available on both the primary node and the secondary node. The secondary system is up and running, but it does not process data until the primary node fails.	Data is replicated and both systems contain identical data. Data replication is performed based on the software capabilities.	A few seconds
Warm standby	The software component is installed and available on the secondary server that is up and running. If a failure occurs on the primary node, the software components are started on the secondary node. This process is automated by using a cluster manager.	Data is regularly replicated to the secondary system or stored on a shared disk.	A few minutes
Cold standby	A secondary node acts as the backup for an identical primary system. The secondary node is installed and configured only when the primary node breaks down for the first time. Later, in the event of a primary node failure, the secondary node is powered on and the data is restored while the failed component is restarted.	Data from a primary system is backed up on a storage system on schedule, less frequent than warm standby.	A few hours

Load Balancing

Load Balancing

- Load balancing is the practice of distributing traffic across more than one server to improve performance and availability.
- A load balancer performs the following functions:
  - Distributes client requests or network load efficiently across multiple servers
  - Ensures high availability by sending requests only to servers that are online (health check)
  - Provides the flexibility to add or subtract servers as demand dictates

Modern high-traffic websites typically serve millions of concurrent requests from users or clients.

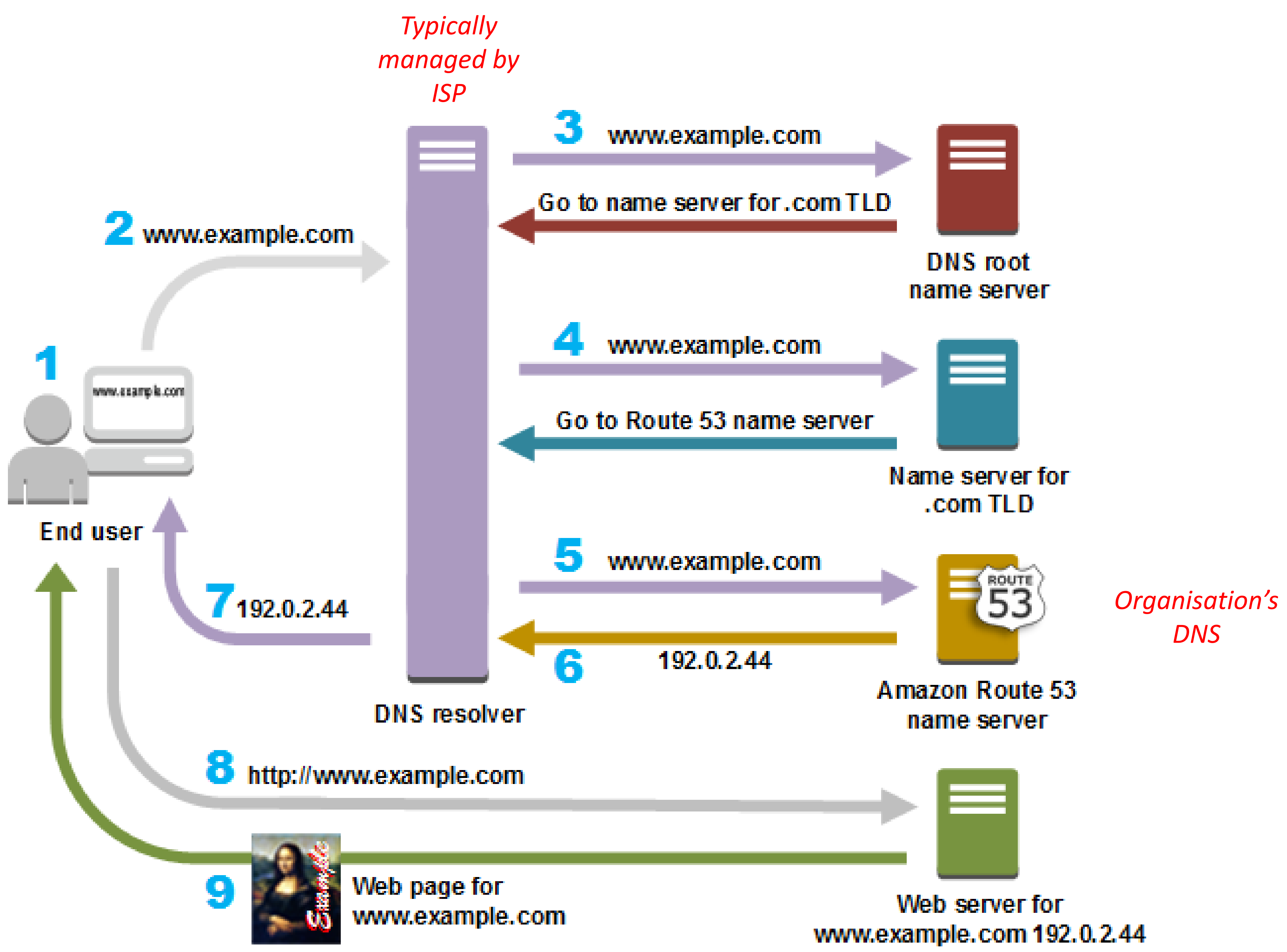


Benefits of Using Load Balancers

- To provide high availability and fault tolerance with the ability to distribute traffic across multiple servers.
- To efficiently increase elasticity and scalability with minimal overhead.
- To reduce the attack surface and provide secure access to your web servers through a single exposed point of access.
  - Centralized management of SSL certificates in the load balancer
- To optimize system architecture by decoupling your environment by using both public facing and internal load balancers.

DNS-based Load Balancing

- DNS-based load balancing is a specific type of load balancing that uses the DNS to distribute traffic across several servers. It does this by providing different IP addresses in response to DNS queries.
- Round-robin DNS
  - Hold multiple DNS records called A records, which contain a domain's name and its matching IP address.
  - Rotate IP addresses in a round-robin fashion when answering queries, spreading the requests across the associated servers
- Pros
  - fast (UDP - connectionless) and simple to implement
- Cons
  - Not consider other conditions such as server load and network congestion.


Typically managed by ISP
3 www.example.com
Go to name server for .com TLD
DNS root name server
2 www.example.com
4 www.example.com
Go to Route 53 name server
Name server for .com TLD
1
www.example.com
End user
5 www.example.com
Organisation's DNS
ROUTE 53
7 192.0.2.44
6 192.0.2.44
Amazon Route 53 name server
DNS resolver
8 http://www.example.com
9 Web page for www.example.com
Web server for www.example.com 192.0.2.44

Load Balancing Algorithms

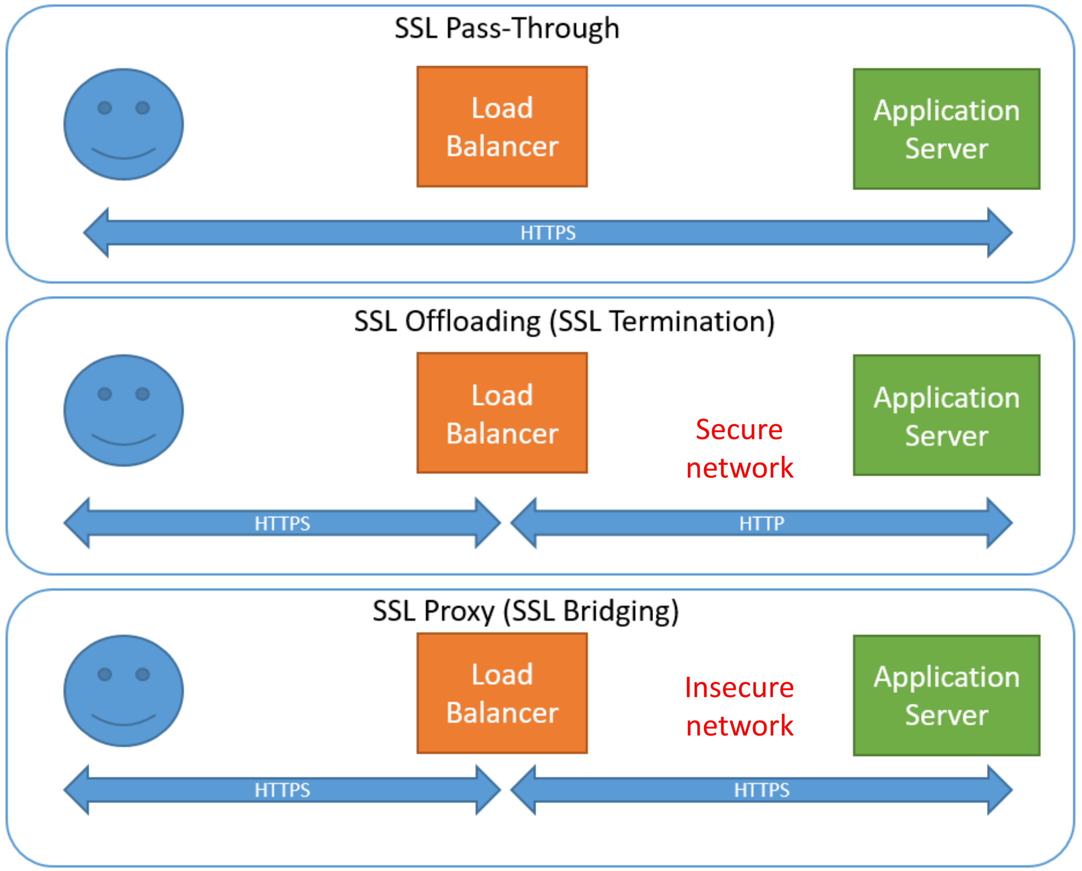
- Specialized software-based or hardware-based load balancers support different load balancing algorithms:
  - Round Robin – Requests are distributed across the group of servers sequentially.
  - Least Connections – A new request is sent to the server with the fewest current connections to clients. The relative computing capacity of each server is factored into determining which one has the least connections.
  - Least Time – Sends requests to the server selected by a formula that combines the fastest response time and fewest active connections.
  - Hash – Distributes requests based on a key you define, such as the client IP address or the request URL.
  - Random with Two Choices – Picks two servers at random and sends the request to the one that is selected by applying the Least Connections algorithm.

Layer 4 and Layer 7 Load Balancing

- Layer 4 LB (Transport Layer)
  - Forward network packets to and from the upstream server without inspecting the content of the packets.
  - Make limited routing decisions by inspecting the first few packets in the TCP stream.
  - Perform Network Address Translation (NAT) on the requests and responses
- Layer 7 LB (Application Layer)
  - Terminate the network traffic and reads the message within, then make a new TCP connection to the selected upstream server.
  - Make a load-balancing decision based on the content of the message (the URL, HTTP Header or cookie, for example).
  - More CPU-intensive than packet-based Layer 4 load balancing.
  - Apply optimizations and changes to the content such as compression and encryption.

SSL Load Balancer

- SSL load balancer is a load balancer that also performs encryption and decryption of data transported via HTTPS, which uses the Secure Sockets Layer (SSL) protocol or the Transport Layer Security (TLS) protocol to secure HTTP data as it crosses the network.
- Act as the server-side SSL endpoint for connections with clients.
  - SSL Pass-Through – not intercept requests
  - SSL Offloading – decrypt requests
  - SSL Proxy – decrypt and re-encrypt requests
- Offload computationally intensive decryption and encryption process.
- Reduce administrative overhead - only need to install and manage the SSL certificates on the load balancer instead of every web and application server.



Elastic Load Balancer and Amazon Route 53

Elastic Load Balancing

Elastic Load Balancing

- A managed load balancing service that distributes incoming application traffic across multiple EC2 instances, containers, IP addresses, and Lambda functions.
  - Distributes incoming traffic across multiple targets in multiple Availability Zones
  - Can be external-facing or internal-facing
  - Each load balancer receives a DNS name
  - Recognizes and responds to unhealthy instances
    - A registered EC2 instance must respond to the target of the health check with an HTTP status code 200 to be considered healthy by your load balancer

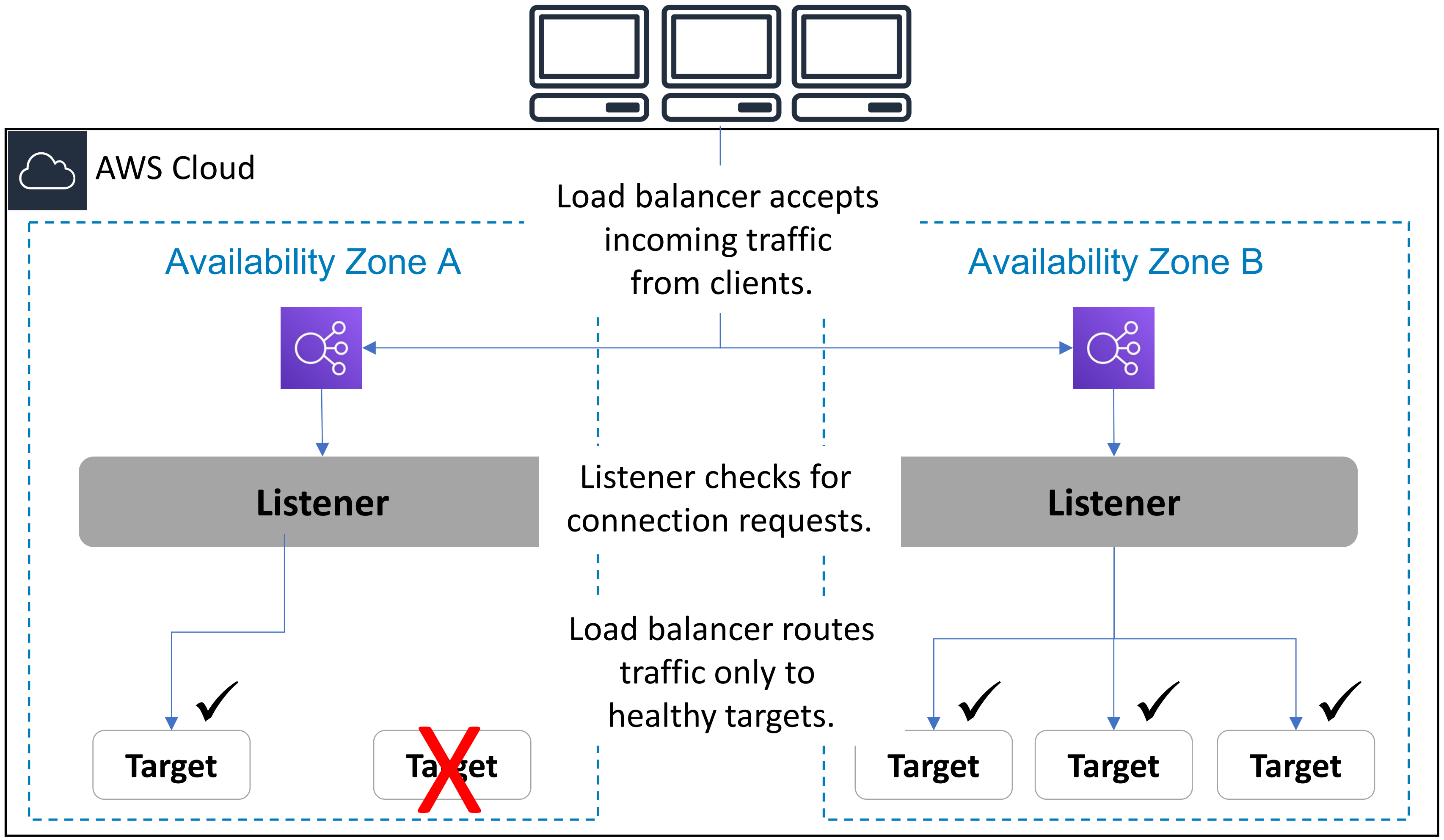
Types of Elastic Load Balancers

Application Load Balancer (ALB)	Network Load Balancer (NLB)
• Flexible application management • Advanced load balancing of HTTP and HTTPS traffic • Operates at the request level (Layer 7) • Content-based routing • Support routing to applications in containers • Support websocket	• Extreme performance and static IP for your application • Load balancing of TCP, UDP, and TLS traffic • Operates at the connection level (Layer 4) • Support tens of millions of requests per second with ultra-low latencies

How Elastic Load Balancing Works

- With Application Load Balancers and Network Load Balancers, you register targets in target groups, and route traffic to the target groups.
- With Classic Load Balancers, you register instances with the load balancer.

The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

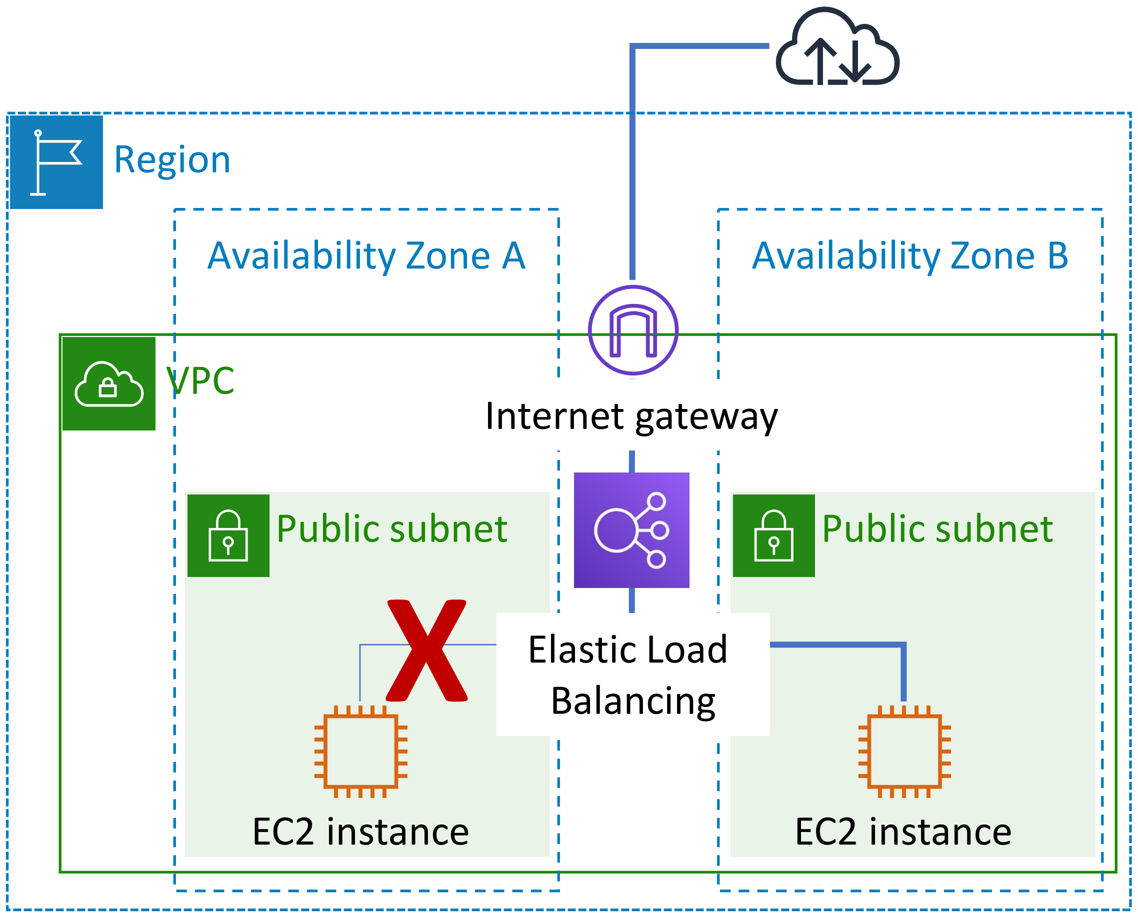

AWS Cloud
Load balancer accepts incoming traffic from clients.
Availability Zone A
Availability Zone B
Listener
Listener checks for connection requests.
Listener
Load balancer routes traffic only to healthy targets.
Target
Target
Target
Target
Target

Load balancer performs health checks to monitor health of registered targets.

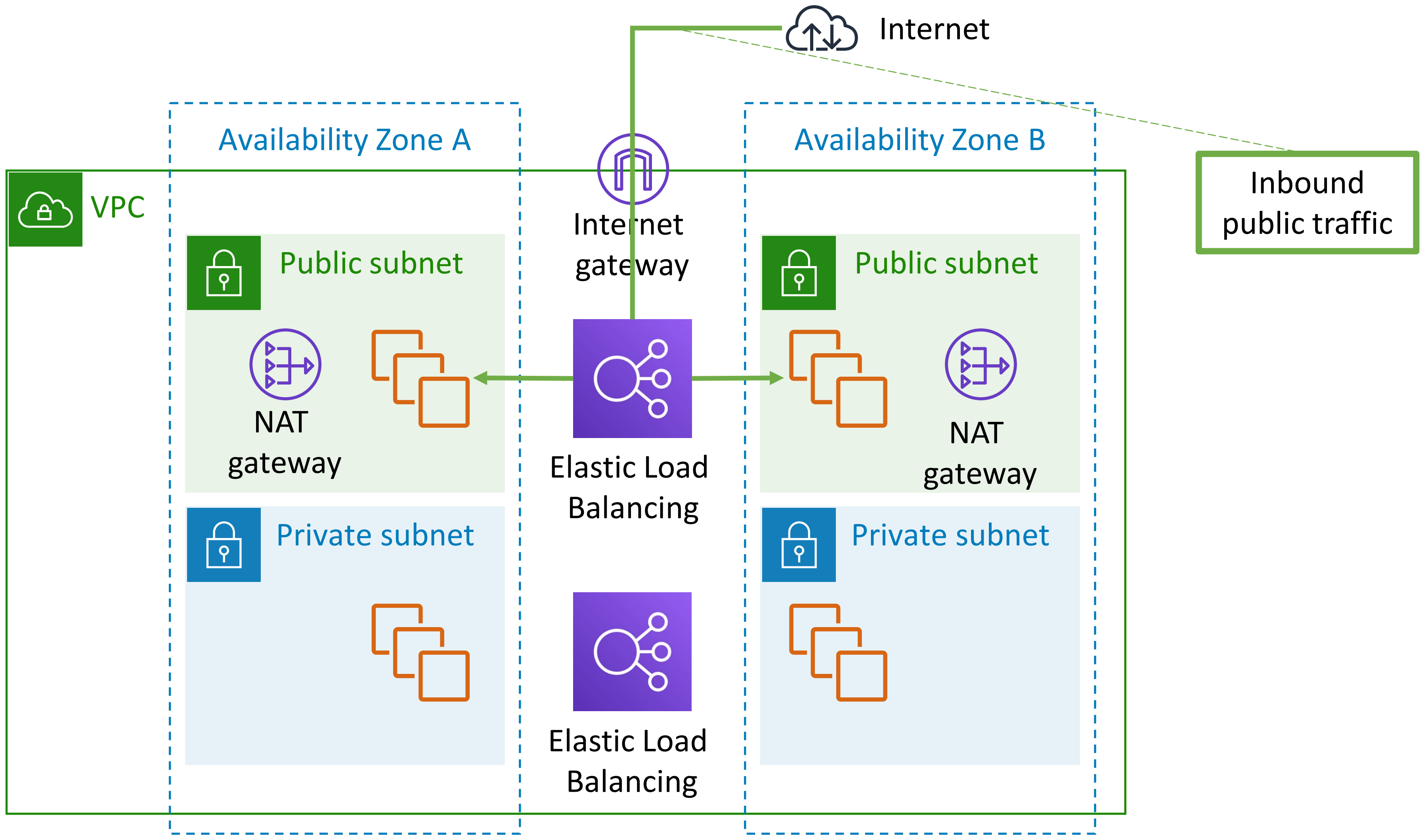
Implementing High Availability

Start with two Availability Zones per AWS Region.

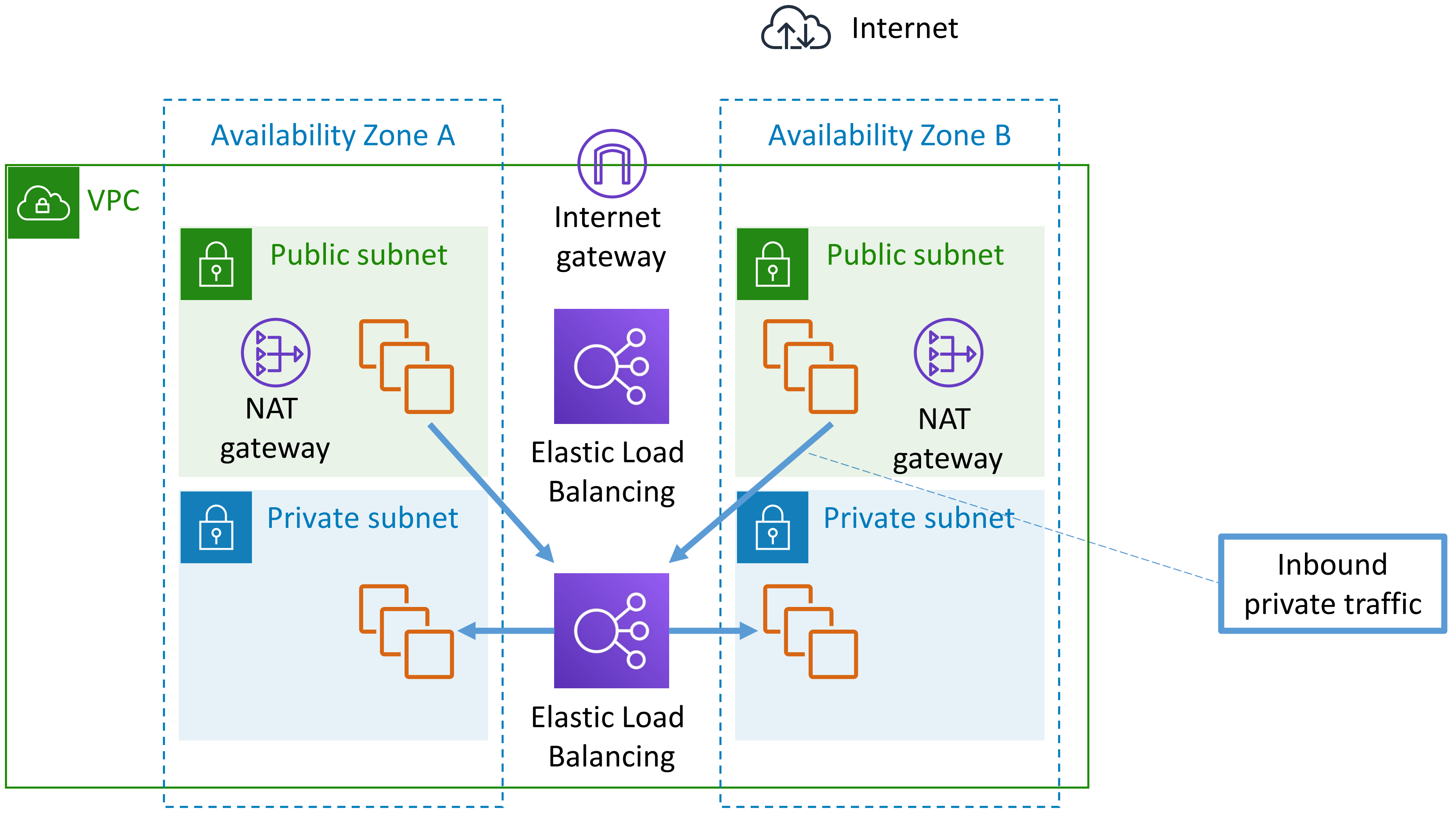
If resources in one Availability Zone are unreachable, your application shouldn't fail.



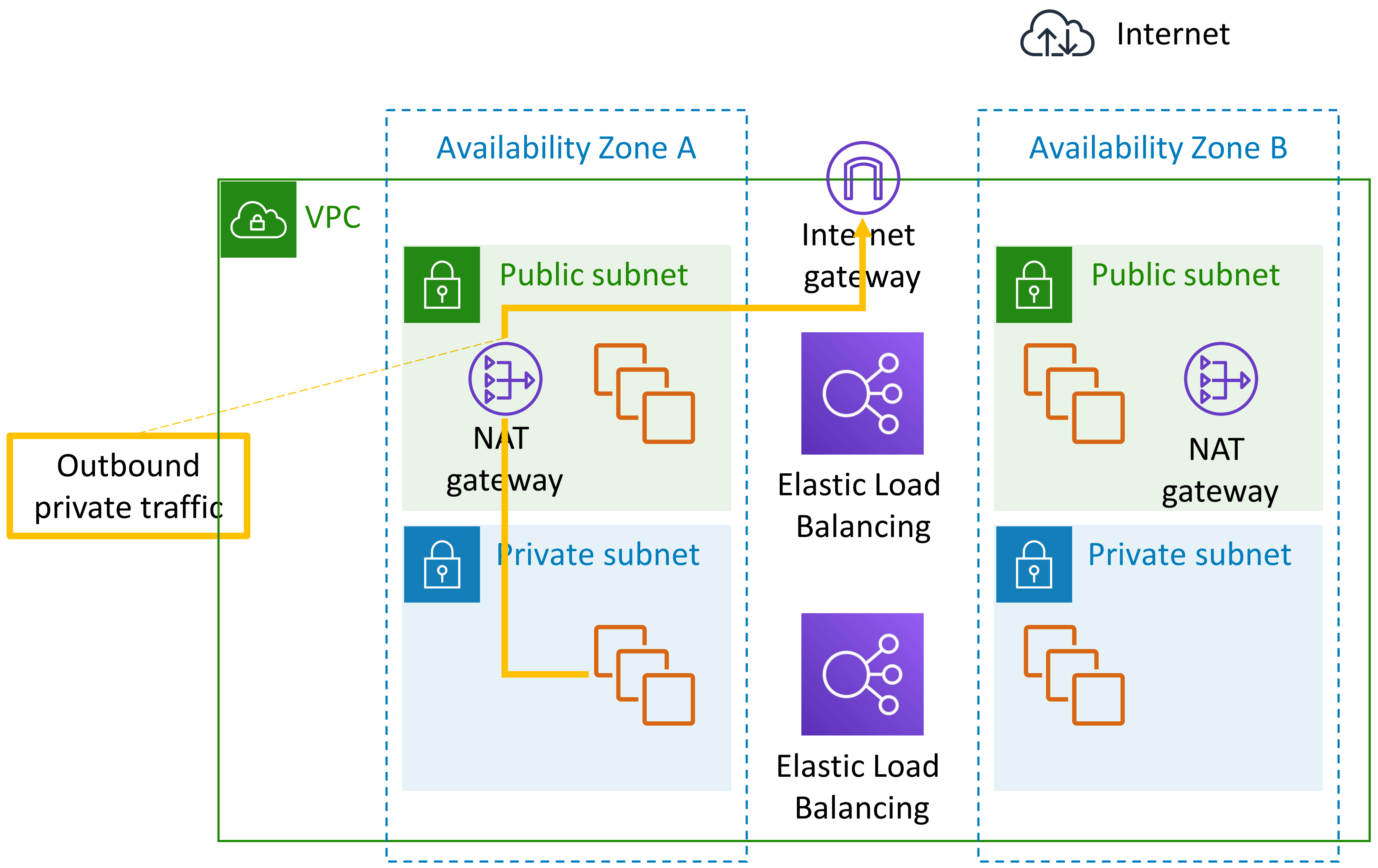
Example of a HA Architecture (1 of 3)



Example of a HA Architecture (2 of 3)



Example of a HA Architecture (3 of 3)



Multiple NAT gateways in different AZs

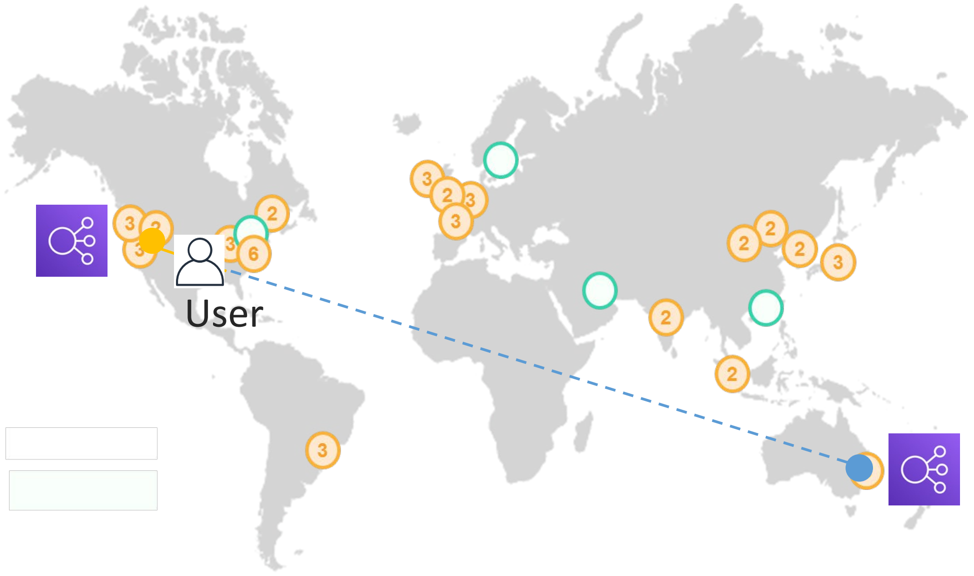
Amazon Route 53

Amazon Route 53

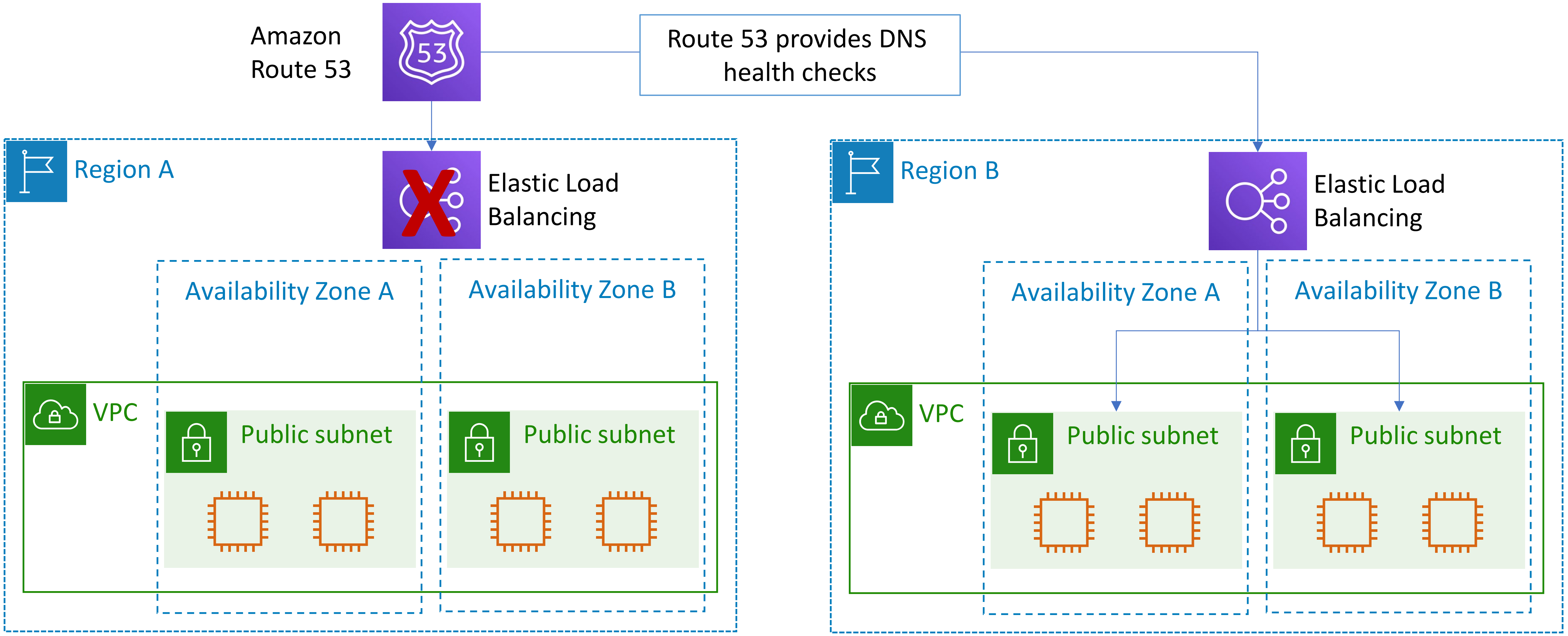
- Amazon Route 53 is a highly available and scalable cloud DNS service.
  - SLA promises 100% availability
- Translates domain names into IP addresses
- Connects user requests to infrastructure that runs inside and outside of AWS
- Can be configured to route traffic to healthy endpoints, or to monitor the health of your application and its endpoints
- Offers registration for domain names
- Has multiple routing options

Amazon Route 53 Routing Policy

- Simple routing
  - Returns all resolved values in random order
- Weighted round robin routing
  - Assign a weight (1-255) to records
- Latency-based routing
  - Based on latency data between users and AWS regions
- Geolocation routing
  - Based on the geographic location of your users
  - Good for localizing content to where you have distribution rights
- Geoproximity routing
  - Based on the physical distance between your users and your resources
- Failover routing
  - Route 53 health-checking agents monitor each location or endpoint of your application to determine its availability
- Multivalue answer routing
  - Return multiple values for healthy resources (up to 8 records), such as IP addresses for your web servers



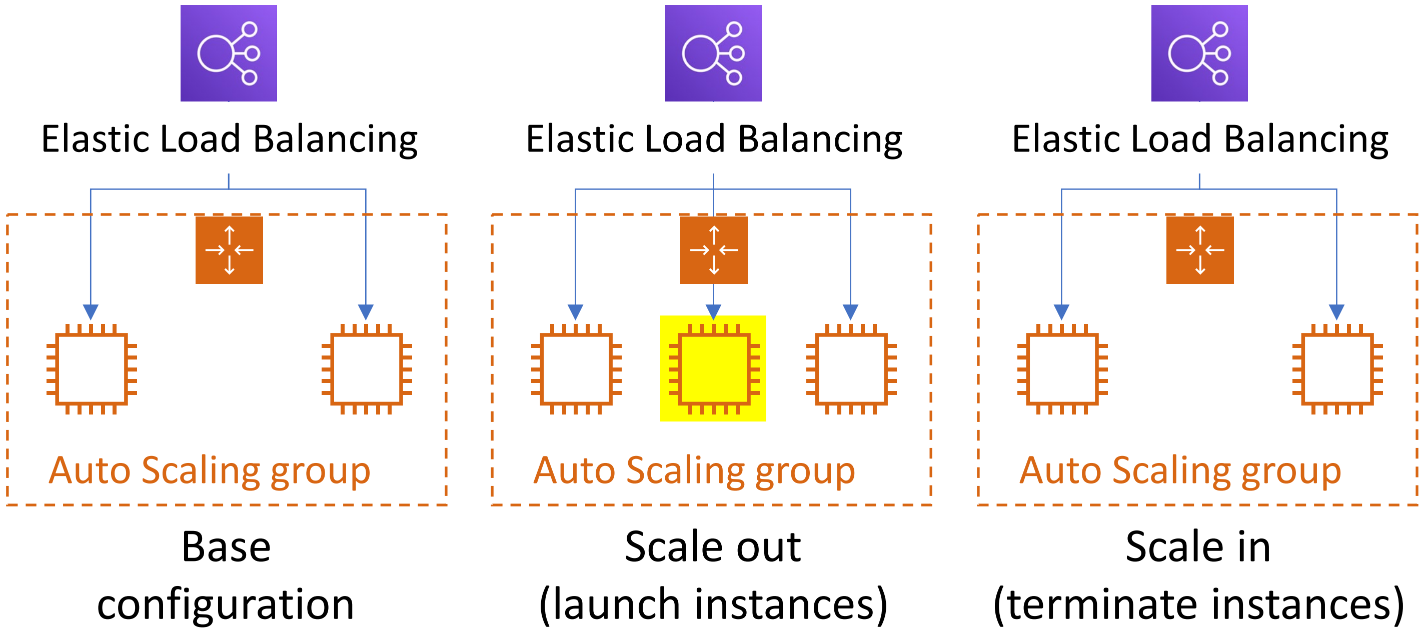
Multi-Region HA and DNS



EC2 Auto Scaling

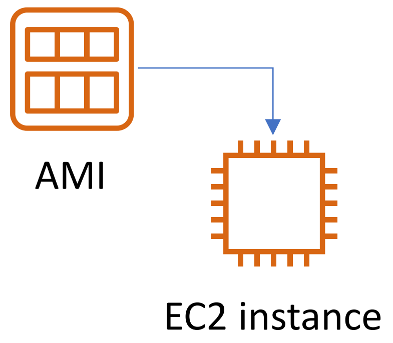
Amazon EC2 Auto Scaling

- Elasticity - an elastic infrastructure can expand and contract as capacity needs change.
- Launches or terminates instances based on specified conditions
- Automatically registers new instances with load balancers when specified
- Can launch across Availability Zones



How Amazon EC2 Auto Scaling Works

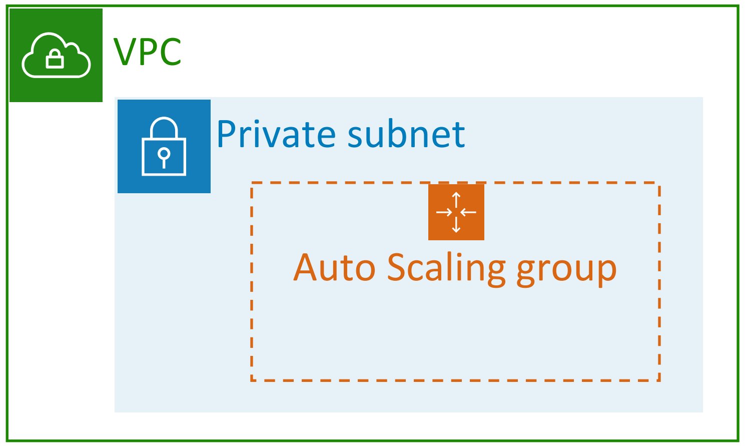
What



Launch configuration

- AMI
- Instance type
- IAM role
- Security groups
- EBS volumes

Where



Auto Scaling group

- VPC and subnets
- Load balancer

You can attach one or more load balancers to an existing Auto Scaling group, it automatically registers the instances in the group and distributes incoming traffic across the instances.

When

Maintain current number

- Health checks

Manual scaling

- Min, max, desired capacity

Scheduled scaling

- Scheduled actions

Dynamic scaling

- Scaling policies

Predictive scaling

- AWS Auto Scaling

Scaling Options

- **Manual scaling** – Specify the change in the maximum, minimum, or desired capacity of your Auto Scaling group.
  - When no auto scaling policy attached, the auto scaling group maintains the desired number of running instances at all times. Performs a periodic health check on running instances and replace unhealthy instances.
- **Scheduled scaling** –Perform scaling automatically as a function of date and time. This is useful for predictable workloads when you know exactly when to increase or decrease the number of instances in your group.
  - For example, say that every week, the traffic to your web application starts to increase on Wednesday, remains high on Thursday, and starts to decrease on Friday. You can plan your scaling actions based on the predictable traffic patterns of your web application.

Scaling Options

- **Dynamic scaling** – A more advanced way to scale your resources by tracking specific CloudWatch metrics (e.g. CPU utilization) that triggering CloudWatch alarm.
  - For example, you have a web application that currently runs on two instances and you want the CPU utilization of the Auto Scaling group to stay close to 50 percent when the load on the application changes. This option is useful for scaling in response to changing conditions, when you don't know when those conditions will change.

- **Predictive scaling** – Uses well-trained machine learning models to predict your expected traffic and EC2 usage, including daily and weekly patterns. The model needs at least 1 day of historical data to start making predictions. It is re-evaluated every 24 hours to create a forecast for the next 48 hours. The prediction process produces a scaling plan that can drive one or more groups of automatically scaled EC2 instances.

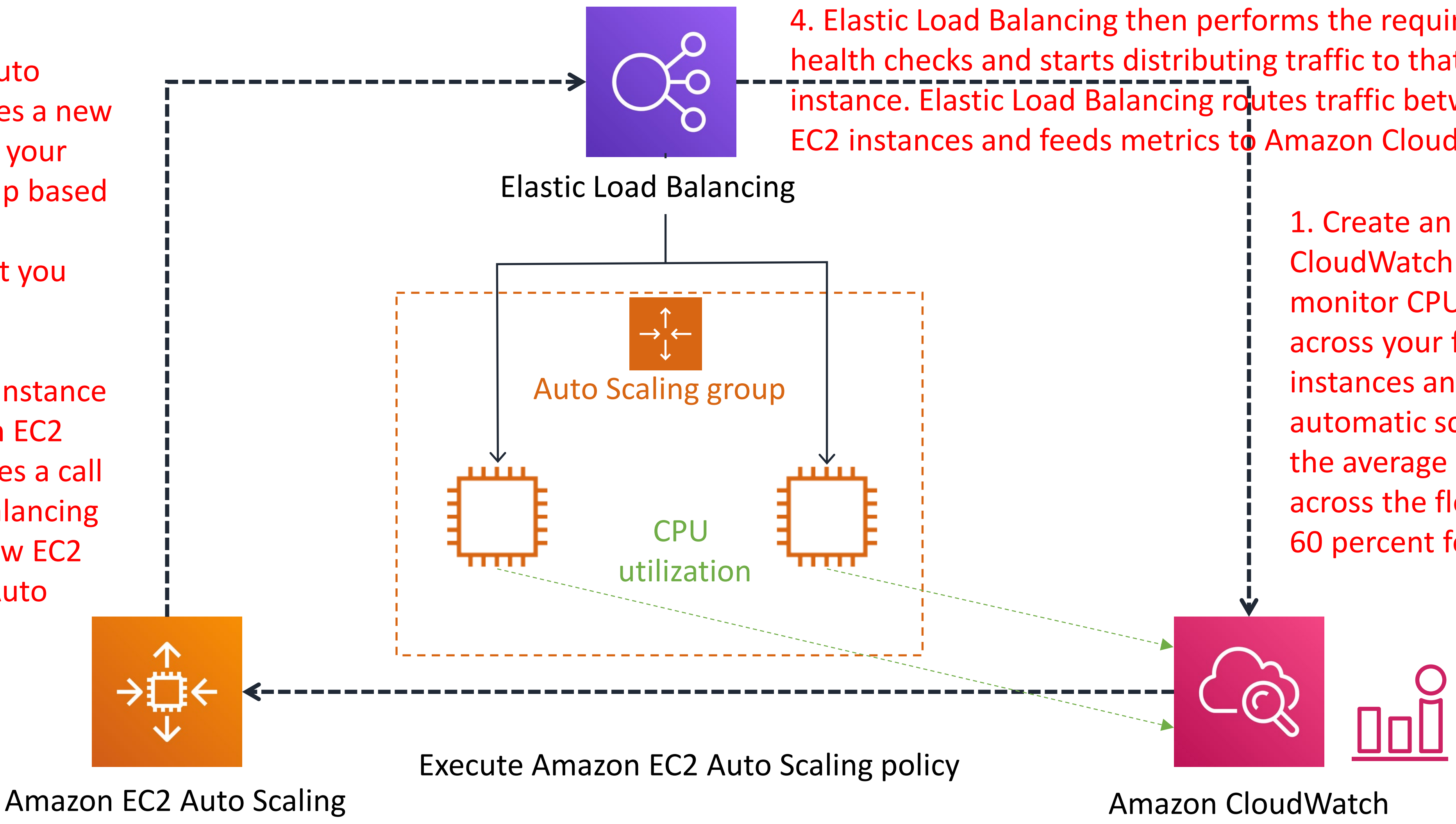
Implementing Dynamic Scaling

2. Amazon EC2 Auto Scaling instantiates a new EC2 instance into your Auto Scaling group based on the launch configuration that you create.

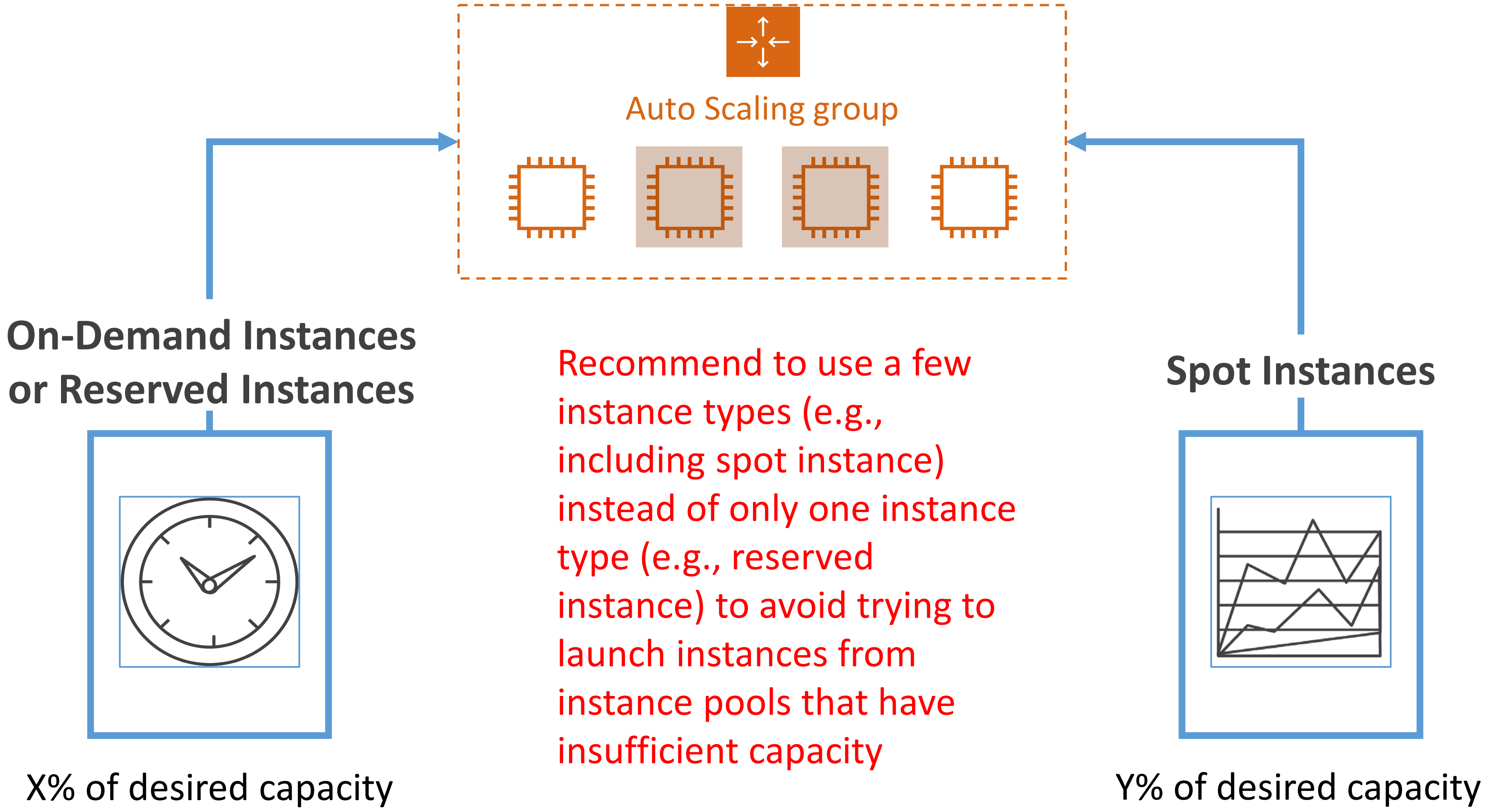
3. After the new instance is added, Amazon EC2 Auto Scaling makes a call to Elastic Load Balancing to register the new EC2 instance in that Auto Scaling group.

4. Elastic Load Balancing then performs the required health checks and starts distributing traffic to that instance. Elastic Load Balancing routes traffic between EC2 instances and feeds metrics to Amazon CloudWatch.

1. Create an Amazon CloudWatch alarm to monitor CPU utilization across your fleet of EC2 instances and execute automatic scaling policies if the average CPU utilization across the fleet goes above 60 percent for 5 minutes.



Auto Scaling Groups with Multiple Instance Types and Purchase Options



Recommend to use a few instance types (e.g., including spot instance) instead of only one instance type (e.g., reserved instance) to avoid trying to launch instances from instance pools that have insufficient capacity