PRJ-SAP-013: Multi-Region Active-Active Web Application

Certification: AWS Certified Solutions Architect – Professional
Domain: Designing for High Availability & Disaster Recovery

---

1. Business Context and GRC Mapping

This project demonstrates the design and deployment of a highly available, fault-tolerant, and low-latency web application operating in an active-active configuration across multiple AWS regions. By distributing traffic across two or more live production regions, this architecture provides superior performance for a global user base and significantly improves the application's resilience against regional outages.

Standardized GRC Mapping

GRC Category	Control/Objective	Implementation Details
Governance	Business Continuity Planning (BCP)	Multi-region active-active deployment ensures RTO (Recovery Time Objective) and RPO (Recovery Point Objective) are minimized to near zero during a regional failure.
Risk Management	Single Point of Failure (SPOF) Mitigation	Distributing workloads across isolated AWS Regions (us-east-1 and eu-west-1) mitigates the risk of a complete application outage due to localized infrastructure failures.
Compliance	Data Availability and Durability	Utilizing DynamoDB Global Tables ensures multi-master replication, meeting stringent data availability requirements common in financial and healthcare regulations.

---

2. Architecture Diagram


Architecture Diagram

The architecture consists of two identical, independent stacks deployed in different AWS regions, with Route 53 managing traffic distribution and DynamoDB handling data replication.

---

3. Comprehensive Deployment Guide

Prerequisites and IAM Permissions

Before beginning the deployment, ensure you have the following prerequisites configured:

1. **AWS Account:** An active AWS account with administrative access.
2. **AWS CLI:** The AWS Command Line Interface installed and configured with appropriate credentials.
3. **Domain Name:** A registered domain name managed by Amazon Route 53 or another DNS provider where you can create records pointing to AWS resources.
4. **IAM Permissions:** The IAM user or role executing this deployment must have permissions to create and manage the following resources:
    - `ec2:*` (VPC, Subnets, Security Groups, Instances, AMIs)
    - `elasticloadbalancing:*` (ALB, Target Groups, Listeners)
    - `autoscaling:*` (Auto Scaling Groups, Launch Configurations/Templates)
    - `dynamodb:*` (Tables, Global Tables, Replication)
    - `route53:*` (Hosted Zones, Record Sets, Health Checks)
    - `iam:*` (Roles, Instance Profiles, Policies)
    - `cloudformation:*` (Stacks, Change Sets)

Step 3.1: Create the DynamoDB Global Table

The global table acts as the multi-master database, synchronizing data across regions with sub-second latency.

1. Navigate to the **DynamoDB Console** in your primary region (e.g., `us-east-1`).
2. Select **Create table**.
3. Enter `Global-Web-App-Data` for the **Table name**.
4. Enter `ItemID` (String) for the **Partition key**.
5. Select **Customize settings** and ensure **DynamoDB Standard** is selected.
6. Click **Create table**.
7. Once the table is active, select it and navigate to the **Global Tables** tab.
8. Click **Create replica**.
9. Select your secondary region (e.g., `eu-west-1`) from the dropdown and click **Create replica**. DynamoDB will initialize the replication process.

Step 3.2: Prepare the CloudFormation Template

Create a file named `active-active-stack.yml` with the following content. This template provisions the regional application stack, including the ALB, ASG, and necessary IAM roles.

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Description: >
  PRJ-SAP-013 - Deploys a regional stack for the multi-region active-active
  application. Uses AWS::EC2::LaunchTemplate (replaces deprecated
  AWS::AutoScaling::LaunchConfiguration).

Parameters:
  VpcId:
    Type: AWS::EC2::VPC::Id
    Description: The VPC to deploy the application in.

  SubnetIds:
    Type: List<AWS::EC2::Subnet::Id>
    Description: A list of at least two public subnets for the ALB and ASG.

  InstanceType:
    Type: String
    Default: t3.micro
    Description: EC2 instance type (t3.micro recommended; t2.micro is
legacy).

  AmiId:
    Type: AWS::EC2::Image::Id
    Description: >
      The Amazon Linux 2 AMI ID for the target region.
      Find the latest ID via: aws ssm get-parameter
      --name /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
      --region <your-region>

Resources:

  # ------------------------------------------------------------
  # Security Groups
  # ------------------------------------------------------------
  ALBSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: "Allow HTTP from the internet to the ALB"
      VpcId: !Ref VpcId
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
```

```yaml
AppSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: "Allow HTTP only from the ALB Security Group"
    VpcId: !Ref VpcId
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        SourceSecurityGroupId: !Ref ALBSecurityGroup

# ------------------------------------------------------------
# Application Load Balancer
# ------------------------------------------------------------
AppLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Name: !Sub "${AWS::StackName}-ALB"
    Subnets: !Ref SubnetIds
    SecurityGroups:
      - !Ref ALBSecurityGroup
    Scheme: internet-facing
    Type: application

ALBListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    LoadBalancerArn: !Ref AppLoadBalancer
    Port: 80
    Protocol: HTTP
    DefaultActions:
      - Type: forward
        TargetGroupArn: !Ref AppTargetGroup

AppTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    Name: !Sub "${AWS::StackName}-TG"
    VpcId: !Ref VpcId
    Port: 80
    Protocol: HTTP
    TargetType: instance
    HealthCheckPath: /
    HealthCheckIntervalSeconds: 30
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 3
```

```
  # ------------------------------------------------------------
  # IAM Role & Instance Profile
  # ------------------------------------------------------------
  AppInstanceRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Sub "${AWS::StackName}-EC2Role"
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: ec2.amazonaws.com
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
      Policies:
        - PolicyName: DynamoDBAccess
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: Allow
                Action: dynamodb:*
                Resource: !Sub >-

arn:aws:dynamodb:${AWS::Region}:${AWS::AccountId}:table/Global-Web-App-Data

  AppInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Roles:
        - !Ref AppInstanceRole

  # ------------------------------------------------------------
  # Launch Template  (replaces deprecated LaunchConfiguration)
  # ------------------------------------------------------------
  AppLaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: !Sub "${AWS::StackName}-LT"
      LaunchTemplateData:
        ImageId: !Ref AmiId
        InstanceType: !Ref InstanceType
        IamInstanceProfile:
          Arn: !GetAtt AppInstanceProfile.Arn
```

```
        SecurityGroupIds:
          - !Ref AppSecurityGroup
        UserData:
          Fn::Base64: !Sub |
            #!/bin/bash
            yum update -y
            yum install -y httpd
            echo "<h1>Welcome! You are being served from ${AWS::Region}
</h1>" \
              > /var/www/html/index.html
            systemctl start httpd
            systemctl enable httpd

  # ------------------------------------------------------------
  # Auto Scaling Group  (references LaunchTemplate, not LaunchConfiguration)
  # ------------------------------------------------------------
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      AutoScalingGroupName: !Sub "${AWS::StackName}-ASG"
      MinSize: '2'
      MaxSize: '4'
      DesiredCapacity: '2'
      VPCZoneIdentifier: !Ref SubnetIds
      TargetGroupARNs:
        - !Ref AppTargetGroup
      LaunchTemplate:
        LaunchTemplateId: !Ref AppLaunchTemplate
        Version: !GetAtt AppLaunchTemplate.LatestVersionNumber
      HealthCheckType: ELB
      HealthCheckGracePeriod: 60
      Tags:
        - Key: Name
          Value: !Sub "${AWS::StackName}-Instance"
          PropagateAtLaunch: true

# ------------------------------------------------------------
# Outputs
# ------------------------------------------------------------
Outputs:
  ALBDNSName:
    Description: The DNS name of the Application Load Balancer
    Value: !GetAtt AppLoadBalancer.DNSName
    Export:
      Name: !Sub "${AWS::StackName}-ALBDNSName"
```

```
  ALBArn:
    Description: ARN of the Application Load Balancer
    Value: !Ref AppLoadBalancer

  AutoScalingGroupName:
    Description: Name of the Auto Scaling Group
    Value: !Ref AutoScalingGroup0.
```

Step 3.3: Deploy the Regional Stacks

You must deploy the CloudFormation stack in both the primary and secondary regions.

1. **Deploy in Region 1 (`us-east-1`):**
    - Navigate to the **CloudFormation Console** in `us-east-1`.
    - Select **Create stack** > **With new resources (standard)**.
    - Upload the `active-active-stack.yml` template.
    - Name the stack `WebApp-USE1`.
    - Provide the required parameters, ensuring you select a valid Amazon Linux 2 AMI ID for `us-east-1` (e.g., `ami-0c55b159cbfafe1f0`).
    - Acknowledge IAM resource creation and deploy.
    - Once complete, copy the `ALBDNSName` from the **Outputs** tab.
2. **Deploy in Region 2 (`eu-west-1`):**
    - Switch your AWS console to `eu-west-1`.
    - Repeat the deployment process, naming the stack `WebApp-EUW1`.
    - **Crucial:** Provide a valid Amazon Linux 2 AMI ID for `eu-west-1` in the parameters.
    - Copy the `ALBDNSName` for this region.

Step 3.4: Configure Route 53 Latency-Based Routing

1. Navigate to the **Route 53 Console** and select **Hosted zones**.
2. Click on your domain name.
3. Select **Create record**.

4. Enter `app` for the **Record name** (e.g., `app.yourdomain.com`).
5. Set the **Record type** to `A`.
6. Enable **Alias** and route traffic to **Alias to Application and Classic Load Balancer**.
7. Select the region of your first ALB (`us-east-1`) and choose the corresponding ALB.
8. Set the **Routing policy** to **Latency**.
9. Select the region (`us-east-1`) and enter a **Record ID** (e.g., `us-east-1-stack`).
10. Create a new health check pointing to the ALB's DNS name to enable automatic failover.
11. Click **Add another record** and repeat steps 4-10 for your second region (`eu-west-1`), selecting the appropriate ALB and region.
12. Click **Create records**.

Step 3.5: Validation and Testing

1. **Latency Routing:** Access your application URL (`http://app.yourdomain.com`). The page will display the region serving your request based on lowest latency. Use a VPN to simulate traffic from different global locations to verify routing changes.
2. **Failover:** In the `us-east-1` Auto Scaling console, set the Desired, Min, and Max capacity to `0`. Once instances terminate and health checks fail, verify that all traffic is automatically routed to `eu-west-1`.
3. **Recovery:** Restore the ASG capacity in `us-east-1`. Once instances are healthy, Route 53 will resume latency-based routing across both regions.

---

4. LinkedIn Post

🚀 New Project Deployed: Multi-Region Active-Active Architecture on AWS (PRJ-SAP-013) 🌍

In today's global digital economy, downtime is not an option. I recently designed and deployed a highly available, fault-tolerant web application spanning multiple AWS regions in an active-active configuration.

By leveraging Amazon Route 53 for latency-based routing, Application Load Balancers, Auto Scaling Groups, and DynamoDB Global Tables, this architecture ensures sub-second data replication and seamless failover capabilities.

Key Technical Highlights: 🔹 **Zero Downtime Failover:** Automated health checks instantly redirect traffic if a regional outage occurs. 🔹 **Global Performance:** Users are dynamically routed to the AWS region offering the lowest network latency. 🔹 **Multi-Master Database:** DynamoDB Global Tables provide fully replicated, multi-region data consistency. 🔹 **Infrastructure as Code:** Fully provisioned using AWS CloudFormation for rapid, repeatable deployments.

This project aligns with the core domains of the AWS Certified Solutions Architect – Professional certification, specifically focusing on Designing for High Availability & Disaster Recovery.

Check out the full architecture diagram below! Let me know your thoughts on multi-region deployments in the comments. 👇

#AWS #CloudComputing #Architecture #HighAvailability #DisasterRecovery #DynamoDB #Route53 #DevOps #TechPortfolio

---

5. YouTube Script: PRJ-SAP-013 Multi-Region Active-Active Web Application

[0:00 - 0:30] Introduction “Welcome back to the channel! Today, we are diving into a highly requested topic for the AWS Solutions Architect Professional exam: Designing a Multi-Region Active-Active Web Application. This is project PRJ-SAP-013. We are going to build an architecture that not only provides incredible performance for global users but can also survive the complete failure of an entire AWS region with zero downtime. Let’s look at the architecture.”

[0:30 - 1:45] Architecture Overview *(Show Architecture Diagram on screen)* “Here is what we are building. We have users distributed globally. They access our application via Amazon Route 53. Route 53 is configured with latency-based routing, meaning it dynamically calculates the fastest network path and sends the user to the closest AWS region—in our case, us-east-1 or eu-west-1.

Inside each region, we have an identical, stateless application stack: an Application Load Balancer distributing traffic to an Auto Scaling Group of EC2 instances. Because the application is stateless, any instance can handle any request.

But what about the data? That's where DynamoDB Global Tables come in. We have a multi-master database setup. When a user in Europe writes data, it hits the eu-west-1 table and is replicated to us-east-1 in under a second. If a region goes down, Route 53 detects the health check failure and routes 100% of traffic to the healthy region. Let's get to the console and build it."

[1:45 - 3:00] Prerequisites and DynamoDB Setup "Before we start, ensure you have an AWS account with admin permissions, the AWS CLI installed, and a registered domain name.

Step one is setting up our multi-master database. We head over to the DynamoDB console in our primary region, us-east-1. We create a standard table named 'Global-Web-App-Data' with 'ItemID' as the partition key. Once active, we navigate to the Global Tables tab and add a replica in our secondary region, eu-west-1. AWS handles the complex replication logic for us automatically."

[3:00 - 5:00] CloudFormation Deployment "Next, we deploy our regional application stacks using Infrastructure as Code. I have a CloudFormation template prepared that provisions the Security Groups, ALB, Target Group, and the Auto Scaling Group.

We deploy this stack first in us-east-1. We provide our VPC, select public subnets, and ensure we use the correct Amazon Linux 2 AMI ID for the N. Virginia region. Once deployed, we copy the Load Balancer DNS name.

Then, we switch our console to eu-west-1 and deploy the exact same template. The only difference is we must update the AMI ID parameter to match the Ireland region. Now we have two independent, running applications."

[5:00 - 6:30] Route 53 Configuration "Now for the magic: connecting them globally. We go to Route 53 and create a new 'A' record for our domain. We choose 'Alias' and route traffic to our Application Load Balancer in us-east-1.

Crucially, under Routing Policy, we select 'Latency'. We associate this with the us-east-1 region and create a health check. We then add another record for the exact same domain name, but this time pointing to the eu-west-1 ALB, again using Latency routing and creating a health check. Route 53 is now managing our global traffic."

[6:30 - 8:00] Testing and Validation “Let’s test it. If I hit the URL from my current location, the web page tells me I am being served from us-east-1 because it’s closest to me. If I turn on a VPN and connect through Europe, refreshing the page shows I am now routed to eu-west-1. Latency routing works!

Now, let’s simulate a disaster. I’ll go into the Auto Scaling Group in us-east-1 and set the capacity to zero, effectively destroying the application in that region. Route 53 health checks will fail. Within minutes, regardless of my physical location, all my traffic is seamlessly routed to the surviving region in Europe. Our active-active failover is successful.”

[8:00 - 8:30] Outro “That wraps up PRJ-SAP-013. We’ve successfully built a highly available, multi-region architecture using Route 53, ALB, ASG, and DynamoDB Global Tables. If you found this tutorial helpful for your AWS SAP studies, please hit that like button and subscribe for more complex AWS projects. The full deployment guide and CloudFormation code are linked in the description below. See you in the next video!”

---

6. YouTube Description

Title: Build a Multi-Region Active-Active AWS Architecture | Route 53 & DynamoDB Global Tables (PRJ-SAP-013)

Description: In this advanced AWS tutorial, we design and deploy a highly available, fault-tolerant, multi-region active-active web application. This architecture is a core concept for the AWS Certified Solutions Architect – Professional (SAP-C02) exam, focusing on Designing for High Availability & Disaster Recovery.

We utilize Amazon Route 53 latency-based routing to direct global users to the fastest AWS region, Application Load Balancers and Auto Scaling Groups for compute, and Amazon DynamoDB Global Tables for a sub-second, multi-master replicated database. Finally, we demonstrate a live regional failover to prove zero-downtime capabilities.

📌 **What you will learn:** 0:00 - Introduction to Active-Active Architecture 0:30 - Architecture Diagram Breakdown 2:10 - Configuring DynamoDB Global Tables 3:00 - Deploying Infrastructure via CloudFormation 5:00 - Setting up Route 53 Latency Routing & Health Checks 6:30 - Testing Global Latency Routing 7:15 - Simulating a Regional Outage (Failover Testing) 8:00 - Conclusion & Outro

🔗 **Resources & Code:** Full Deployment Guide & CloudFormation Template: [Insert Link to GitHub/Blog] AWS DynamoDB Global Tables Documentation: https://aws.amazon.com/dynamodb/global-tables/

💡 **Business Context & GRC:** This project demonstrates robust Business Continuity Planning (BCP) by mitigating Single Points of Failure (SPOF) across isolated AWS Regions, ensuring high data availability and durability compliant with strict enterprise regulations.

Don’t forget to LIKE, COMMENT, and SUBSCRIBE for more professional-grade AWS tutorials!

#AWS #SolutionsArchitect #CloudComputing #DisasterRecovery #Route53 #DynamoDB #AWSArchitecture #DevOps #TechTutorial