Disaster recovery plan

Considerations and Assumptions:

- This is a global 3-tier application. Application server customers around the globe and uptime is very important
- Uptime is very critical and the application needs to be available at mostly all times. So RTO < 6 hours
- Data is also critical but uptime is more important. The RPO < 2 hours. So, Data should be replicated/backed up at least every 2 hours

Based on these factors. A **warm stand-by DR** plan is recommended

Warm-stand-by: Full system is up and running, but at minimum size

Aurora MYSQL Database

1. Ensure snapshots of the database are taken **at least** every 2 hours. It's highly recommended this process is automated so as to avoid any more loss of data
2. Set up Cross-Region Replication for the Aurora Database: An Amazon Aurora MySQL DB cluster will be created as a Read Replica in a different AWS Region than the source DB cluster. (This is an excellent approach for quickly deploying applications to other regions). For a global application, this is the recommended approach
3. Set up Aurora Global Database
4. Implement high availability for all Aurora resources. This ensures the following
    1. Data: is written across all availability zones which protects against a single point of failure ensuring data redundancy
    2. Database: Setting up aurora read replicas so in the case the primary instance fails, a quick failover is in place to promote a read replica(in a different AZ) to the primary instance

ECS

Prerequisites

- Ensure all task definitions are versioned and preferably stored in repository for additional version control
- Use cloud formation templates to set up the Cluster, service,targets, tasks and capacity providers. Store these in a version control system like GITHUB
- Proper health checks are set up to alert when service is down

- For automation, utilize CICD pipeline to deploy tasks to cluster in both primary and secondary regions
- Identify region the ECS needs to be deployed

1. Deploy application to another region, running the application with minimal capacity as possible. If the above prerequisites are fulfilled then the ECS application and its resources should be easily deployed to the back up region
2. Set up a route 53 failover routing policy to route traffic to the back up region where the ECS is hosted in the event the primary region is down. The health check implementation and switch over can be implemented by cloudwatch events and lambda function
3. With the failover, ensure that all resource constraints are applied in the secondary region just as in the primary region

ECR

It’s also important to ensure a recovery plan is in place for your ECR images. AWS provides **Cross region replication** for ECR which replicates container images to different regions.

ALB

1. Set up an Application Load Balancer (ALB) to distribute incoming traffic across multiple ECS instances. ALB provides high availability and fault tolerance by automatically detecting unhealthy instances and routing traffic to healthy ones.
2. Set up ALB to have multiple AZs attached. Additionally, deploy ALB to secondary region and leverage AWS Global Accelerator with the ALB to distribute traffic across regions in the instance of failure in the primary region
3. Set up AWS route 53 failover routing policy to move traffic over to the secondary region when the health checks detect failure/downtime in the primary region’s services.

S3:

Enable S3 - Cross region replication to always replicate data stored in an S3 bucket within a specific region to other regions.

Other considerations (Heavily recommended)

- Ensure CICD pipelines are in place to quickly deploy applications
- Enforce deployment strategies for ECS applications (rolling, blue/green)
- Use Cloudformation/CDK to easily deploy resources(VPC, Gateway, NAT gateway) into AWS regions. Store these templates in a version control system