Table of Contents

[Prerequisites 1](#_Toc209997600)

[Create of MySQL DB 1](#_Toc209997601)

[Application load balancer & target group creation 2](#_Toc209997602)

[ECS Cluster Creation 4](#_Toc209997603)

[Task definition 4](#_Toc209997604)

[Service creation 6](#_Toc209997605)

[CloudWatch alarm setup 6](#_Toc209997606)

# **Prerequisites**

1. **IAM Permissions** Ensure proper permission to be added to role and user like fetch credential from secret manager, image push to ECR
2. **DB Subnet Group** Create a DB subnet group from the RDS interface (left-hand pane, near the bottom).
3. **Security Group** Create a dedicated security group for the database with appropriate inbound rules.

# **Create of MySQL DB**

1. Sign in to AWS Console using your credentials.
2. Select the Region where the database should be created.
3. Search for RDS in the top search bar and select **Aurora and RDS**.
4. From the navigation window, select **Databases**.
5. Choose Standard Create for full configuration control.
6. Select **MySQL** as the database engine.
7. Choose the desired engine version from the dropdown.
8. Select a **template** (e.g., Production, Dev/Test) to apply best-practice by defaults.
9. In the Settings section, provide a DB Cluster Identifier (database name).
10. **Credential Settings**:
    * Provide a username and password.
    * Choose between:
      + **Self-managed**: Manual password entry, no encryption.
      + **AWS Secrets Manager**: Auto-generated, encrypted, lifecycle-managed via KMS.
11. **Instance Configuration**:
    * Select instance type based on workload requirements.
12. **Storage Configuration**:
    * Choose Provisioned IOPS SSD for high performance.
    * Enable storage autoscaling and set thresholds.
    * Enable Dedicated Log Volume for transaction logs.
13. **Availability & Durability**:
    * Enable **Multi-AZ Deployment** for automatic failover.
14. **Connectivity Settings**:
    * **Compute Resource**: Select EC2 instance if needed.
    * **Network Type**: IPv4
    * **VPC**: Choose the target VPC.
    * **DB Subnet Group**: Select the pre-created group.
    * **Public Access**: Set to **NO**
    * **Security Group**: Attach the pre-created DB SG.
15. **Database Authentication**:
    * Choose Password Authentication (or other methods as needed).
16. **Monitoring**:
    * Select Standard or Enhanced Monitoring.
    * Encryption is enabled by default via AWS KMS.
17. **Additional Monitoring Settings**:
    * Configure:
      + Metric collection frequency
      + Log types to send to CloudWatch (enable audit, error, general, slow query)
18. **Additional Configuration**:
    * + Create an initial database inside the DB instance
      + Configure backups and retention
      + Enable cross-region replication
      + Set maintenance window

# **Application load balancer & target group creation**

1. Sign in to the AWS Console using your credentials.
2. Select the region where you want to create the ALB.
3. In the console’s top left search bar, search for Load Balancer and it.
4. Click Create Load Balancer, then select Application Load Balancer.
5. **Under** **Basic Configuration**:
   * Provide a name for your ALB.
   * **Choose the Scheme:**
     + Internet-facing (for public access) or
     + Internal (private).
   * Select IP address type as IPv4.
6. **Under** **Networking**:
   * Select the appropriate VPC.
   * Select at least two public subnets for high availability.
7. Choose the Security Group that allows HTTP and HTTPS traffic from clients.
8. **Add** **Listeners**:
   * Ports **80** (HTTP) and **443** (HTTPS).
9. If HTTPS listener is added:
   * Provide an SSL certificate from AWS Certificate Manager (ACM) to enable secure communication.
10. **Routing Configuration**:
    * Select **Forward to Target Group** as the routing action.
11. **Create Target Group** (if one doesn’t already exist):
    * From the ALB creation interface or EC2 console, choose to create a new target group.
    * Select **Target Type** as **IP address**.
    * Provide a name for the target group.
    * Define the **protocol** and **port** on which traffic will be routed.
    * Select the **VPC** where your application is deployed.
    * Configure health checks and click **Next**.
    * Review the settings and create the target group.
12. Finally, create the **Load Balancer**.

# **ECS Cluster Creation**

1. Provide a **cluster name**.
2. Select **Amazon EC2** as the infrastructure type.
3. Choose **Create new Auto Scaling Group (ASG)** from the dropdown with **on-demand** provisioning model.
4. Select **Amazon Linux** as the cluster AMI.
5. Choose instance type according to workload requirements.
6. Select **ecsInstanceRole** if already created; otherwise, choose **create new**, which will auto-provision the role.
7. Specify **desired capacity**, along with minimum and maximum instance counts.
8. Create an **SSH key pair** for EC2 instances.
9. Provide **volume size** for instances.
10. Under **Network settings** for EC2 instances:
    * Select the appropriate **VPC**.
    * Choose private subnets with NAT gateway attached.
11. Select the **security group** that allows traffic from the Application Load Balancer (ALB) only.
12. Turn **off** Auto-assign Public IP.
13. In the **Monitoring** section, select the recommended level of observability.
14. Enable **ECS exec logging and encryption** if needed:
    * Exec enables running commands inside containers.
    * All commands are logged and encrypted.
15. Select **encryption at storage level**, typically using the default AWS KMS key.
16. Click **Create cluster**.

## **Task definition**

1. Select **Task Definitions** from the left navigation.
2. Click **Create new task definition**.
3. Provide a **name** for the task definition.
4. Under **Infrastructure requirements**:
   * Select **EC2** launch type.
   * OS type: **Linux**.
   * Network mode: **awsvpc** (required for IP address-based target groups).
   * Specify CPU and memory resources.
5. Attach a **task role** that grants permissions such as:
   * Pulling images from ECR.
   * Accessing AWS Secrets Manager.
6. Select or create a **task execution role** (AWS can auto-create or manual creation possible).
7. Under **Container** section:
   * Provide container **name**.
   * Set **essential** to “Yes” if required.
   * Provide **image URI** from ECR or browse.
   * Enable **private registry authentication** and provide the Secrets Manager ARN if third party registry is used.
8. Add **Secrets Manager environment variables** to inject DB username and password into the container.
9. Use application code or libraries (e.g., Python’s os library) to fetch these secrets.
10. Configure **logging**:
    * Enable **log collection**.
    * Select **CloudWatch Logs**.
    * Set:
      + awslogs-group = value type = path where log will send to CloudWatch /ecs/nginx
      + awslogs-region = value type = select region
      + awslogs-stream-prefix = value type = assessment, it will store container log inside log group and path will start with prefix which assessment.
11. Mount volumes if required.
12. Under **Monitoring**, enable **metric collection** and select CloudWatch to gather task-level numeric metrics.

## **Service creation**

1. Select the **cluster** and click **Launch**.
2. Choose **Service**.
3. Select the **task definition** under service details.
4. Provide a **name** for the service.
5. Select compute option using **Capacity Provider Strategy** with the cluster default.
6. Enable **ECS Exec** to allow command execution inside containers.
7. Define the **desired number of replicas**.
8. Enable **Availability Zone Rebalancing**.
9. Select **Load Balancing**:
   * Choose existing ALB.
   * Select the listener and target group.
10. Click **Create**.
11. Tasks will deploy and start running successfully.

## **CloudWatch alarm setup**

1. Navigate to the **Alarms** section in CloudWatch.
2. Click **Create alarm**.
3. Select **Metric**.
4. Choose **ECS/ContainerInsights** namespace.
5. Select your **clusterName** and **TaskDefinitionFamily**.
6. Pick the specific Task Definition Family (e.g., nginx-test that I have created).
7. Choose the metric **CPUUtilized**.
8. Click **Next** to go to the **Condition** section.
9. Set **Threshold type** to **Static**.
10. Choose the condition **Greater than threshold** for CPU utilization.
11. Enter a threshold value (e.g., 70), meaning the alarm triggers if CPU usage exceeds 70%.
12. Optionally, configure **datapoints to alarm** to reduce false positives.
13. Click **Next**.
14. Set **Alarm state trigger** to **In Alarm**.
15. Select an existing or create a new **SNS topic** for notifications.
    * Ensure subscribers (email or SMS) have confirmed subscription to receive alerts.
16. We can also configure **Auto Scaling actions** to respond automatically when the alarm triggers.
17. Provide a **name** for the alarm.
18. Add a **description** for clarity.
19. Review the configuration in the preview.
20. Click **Create alarm**.

## **CloudWatch Alarm Logging**

1. Go to the AWS Console and navigate to CloudWatch.
2. n the left-hand pane, select Log groups.
3. Locate and select the log group name /ecs/nginx-test.
4. Inside the log group, click on the relevant log stream.

Review the events and logs captured from your ECS tasks