PRJ-DOP-024: Blue/Green Deployment for Stateful Database (RDS)

Certification: AWS Certified DevOps Engineer – Professional
Domain: Database Deployment Strategies

---

1. Project Overview

This project tackles one of the most challenging aspects of continuous deployment: updating a stateful, relational database with minimal downtime. While stateless applications can be easily deployed using strategies like blue/green or canary, databases require careful handling to avoid data loss or inconsistency. This project demonstrates how to perform a **blue/green deployment for an Amazon RDS database**, a strategy that allows for safe, controlled, and reversible database upgrades.

We will use the native **Amazon RDS Blue/Green Deployments** feature. This feature creates a fully synchronized, separate, and identical staging environment (the “green” environment) that mirrors the production environment (the “blue” environment). We can make changes to the green environment, such as upgrading the database engine version or modifying the schema, and then promote it to become the new production environment with near-zero downtime.

Key Objectives

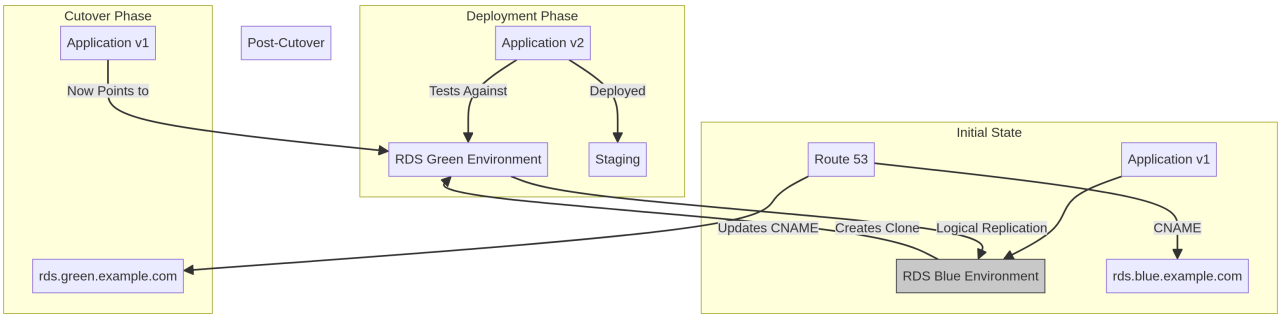
- Understand the challenges of deploying changes to stateful database systems.
- Create an RDS Blue/Green Deployment for an existing RDS database.
- Apply changes (like a minor version upgrade) to the green environment.
- Test the application against the updated green environment.
- Perform a safe, controlled switchover, promoting the green environment to become the new blue.

- Understand the rollback capabilities and safety guardrails of the feature.

---

2. Architecture

The architecture involves creating a complete, synchronized copy of the production database stack, which is then promoted to production after changes are made and verified.



Deployment Flow:

1. **Initial State (Blue Environment):**
    - The production application points to the production RDS database (the “blue” environment).
    - Best practice is for the application to connect via a CNAME record in Route 53 (e.g., `db.prod.example.com`) that points to the actual RDS endpoint. This decouples the application from the physical database endpoint.
2. **Create Blue/Green Deployment:**
    - From the RDS console, we select the blue database and choose “Create Blue/Green Deployment”.
    - RDS automatically creates a complete, separate copy of the blue database, including any read replicas. This new environment is the “green” environment.
    - RDS uses the database engine’s native **logical replication** to continuously synchronize data from the blue environment to the green environment. This ensures the green database is always up-to-date with production data.
3. **Update & Test the Green Environment:**

- The green environment is now a fully independent staging environment. We can perform risky changes on it without any impact on production.
- **Example Change:** We can perform a minor version upgrade on the green database.
- We can deploy a new version of our application to a staging environment and point it to the green database to run integration tests, performance tests, and schema validation.

4. **Switchover (Promote Green to Blue):**
    - Once we are confident that the green environment is stable and correct, we initiate the switchover.
    - RDS performs a series of steps to ensure a safe transition: a. It blocks writes on both the blue and green databases to prevent data loss. b. It waits for the green database to fully catch up with the last transactions from the blue database. c. It **renames the database endpoints**. The original blue endpoint is renamed, and the green endpoint is renamed to take the original blue endpoint's name. d. It switches traffic to the newly promoted green environment.
    - This entire process typically completes in **under a minute**, even for large databases.
5. **Post-Switchover:**
    - The application, which was always pointing to the original RDS endpoint name, is now seamlessly connected to the new, upgraded database.
    - The old blue environment is kept around. It is **not** automatically deleted. This provides a safety net; if a major issue is discovered post-switchover, you could potentially use the old environment to recover.

---

3. Prerequisites

- An AWS account with administrative permissions.
- An existing Amazon RDS for MySQL or PostgreSQL database. The blue/green feature has specific version requirements, so ensure your DB is compatible.

- The database must have **automated backups enabled**.
- A sample application that connects to the database.

---

4. Step-by-Step Implementation Guide

Step 4.1: Prepare the Production (Blue) Environment

1. If you don't have one, create an Amazon RDS for MySQL (e.g., version 8.0.28) or PostgreSQL database. This will be your blue environment.
2. Ensure **automated backups** are enabled for the instance.
3. Create a CNAME record in Route 53 (e.g., `database.prod.yourdomain.com`) that points to the RDS instance's endpoint.
4. Configure your application to connect to the database using this CNAME record.

Step 4.2: Create the Blue/Green Deployment

1. Go to the **Amazon RDS console** and select your production database.
2. From the **Actions** menu, choose **Create Blue/Green Deployment**.
3. **Blue/Green deployment identifier:** Give it a unique name (e.g., `db-upgrade-v-next`).
4. **Green database configuration:** You can specify changes here, but for this example, we will keep the configuration the same initially.
5. Click **Create**. RDS will now provision the green environment. This can take some time, depending on the size of your database.
6. Once the status is **Available**, you will see both the blue and green environments in the console. Note that the green environment has its own unique RDS endpoint.

Step 4.3: Modify and Test the Green Environment

1. **Perform an Upgrade:**
    - Select the green database instance in the RDS console.

- Click **Modify**.
- Change the **DB engine version** to a newer minor version that is supported for upgrades.
- Apply the changes immediately.
- RDS will now upgrade the green database instance. This has **zero impact** on the blue production database.

2. **Run Tests:**
    - Deploy a staging version of your application.
    - Configure this staging application to point to the **green database's endpoint**.
    - Run your full suite of regression tests, performance tests, and any other validation checks to ensure the application works correctly with the upgraded database.

Step 4.4: Perform the Switchover

1. Go back to the **Blue/Green Deployments** page in the RDS console.
2. Select the deployment you created.
3. From the **Actions** menu, choose **Switch over**.
4. **Review the confirmation:** The console will warn you that this will cause a brief outage (typically under a minute). It will also show you a list of checks it performs, such as ensuring the replication lag is minimal.
5. **Timeout:** You can specify a timeout for the switchover. If the process takes longer than this timeout, it will be aborted. This is a critical safety feature.
6. Type `Switch over` to confirm and click the **Switch over** button.

Step 4.5: Monitor and Verify

1. **Monitor the process:** The status of the blue/green deployment will change to **Switching over**.
2. **Verify the application:** Your production application, which was never reconfigured, should continue to function. After the brief switchover period, it will be connected to the newly upgraded database.

3. **Check the endpoints:** In the RDS console, you will notice that the instance that was originally green now has the original production endpoint name. The old blue instance has been renamed.

---

5. Post-Switchover and Cleanup

- The old blue environment is kept for as long as you need it. It is a best practice to keep it around for a period (e.g., a day or a week) until you are completely confident in the new environment.
- Once you are satisfied, you can delete the old blue environment.
- Go to the **Blue/Green Deployments** page, select the deployment, and from the **Actions** menu, choose **Delete**. This will terminate the old blue database instance.

---

6. Best Practices & Considerations

- **Workload:** It is best to perform switchovers during a period of low database traffic to minimize any potential impact.
- **Replication Lag:** Before switching over, always check the replication lag in the CloudWatch metrics for the green database to ensure it is near zero.
- **Schema Changes:** This feature is ideal for engine upgrades, parameter changes, and some types of schema changes. However, complex schema changes that are not backward-compatible may still require a more traditional migration approach (e.g., using AWS DMS).
- **Testing:** The value of a blue/green deployment is directly proportional to the quality of the testing performed on the green environment before switchover.

Business Context

The Problem

Development teams face slow deployment cycles, manual testing bottlenecks, and security vulnerabilities introduced during the CI/CD process. Traditional deployment

methods lack security scanning, leading to vulnerabilities reaching production. Manual processes create inconsistency and human error.

The Solution

Secure CI/CD pipeline with integrated security scanning, automated testing, and infrastructure as code. Implements shift-left security with SAST, DAST, and dependency scanning. Automates deployments while enforcing security gates and compliance checks at every stage.

Business Value

- **Faster Time to Market:** Automated pipelines reduce deployment time from weeks to hours
- **Improved Security Posture:** Catches vulnerabilities before production deployment
- **Reduced Manual Effort:** Eliminates 80%+ of manual deployment tasks
- **Audit Trail:** Complete deployment history and security scan results for compliance

Risk Mitigation

Prevents deployment of vulnerable code, hardcoded secrets, misconfigured infrastructure, and non-compliant resources to production environments.

GRC Mapping

Compliance Frameworks

- **NIST CSF:** PR.IP-1 (Baseline configuration), PR.DS-6 (Integrity checking), DE.CM-4 (Code analysis)
- **ISO 27001:** A.12.1 (Operational procedures), A.14.2 (Security in development), A.12.4 (Logging)
- **CIS Controls:** Control 16 (Application Software Security), Control 11 (Secure Configuration)

- **OWASP DevSecOps:** Security testing integration, Secret management, Dependency checking

Security Controls Implemented

- Static Application Security Testing (SAST)
- Dynamic Application Security Testing (DAST)
- Software Composition Analysis (SCA)
- Secrets scanning and prevention
- Infrastructure as Code security validation

Audit Evidence

- Pipeline execution logs with security scan results
- Code commit and approval records
- Security test reports (SAST/DAST/SCA)
- Deployment approval and rollback records

Regulatory Alignment

- **SOX:** Section 404 (Change management controls)
- **PCI DSS:** Requirement 6.3 (Secure development), Requirement 6.5 (Common vulnerabilities)
- **GDPR:** Article 25 (Data protection by design), Article 32 (Security of processing)
- **SOC 2:** CC8.1 (Change management), CC7.2 (System monitoring)