**Blue-Green Deployment Document - Modular Design**

Document Version: 1.7

Date: June 17, 2025

Author: [Your Name/Team]

**1. Introduction**

This document outlines a modular approach to Blue-Green deployment, allowing for the independent deployment and management of individual components, as well as their orchestration within a complete Blue-Green pipeline. This strategy enhances reusability, reduces dependencies, and provides greater flexibility for deploying specific infrastructure or application elements outside of a full pipeline run.

**2. Modular Deployment Strategy Overview**

The Blue-Green deployment process is broken down into a series of independent, callable modules. Each module encapsulates a specific logical step or resource creation, with defined inputs and outputs. These modules can be executed individually to deploy or manage specific components, or chained together to form the complete Blue-Green pipeline for an application, as depicted in the accompanying flowcharts. This document presents two distinct end-to-end pipeline paths, defined by the method chosen for populating the "Green" database: either via an **InRegion Read Replica** (where the "Green" database instance is *created directly from* the "Blue" database as a replica) or through a traditional **Export/Import** process (where a new, empty "Green" database instance is first provisioned). A preliminary step involves ensuring the KMS key for the Green environment is appropriately managed, which may include "decoupling" if the existing KMS key is tightly integrated with the Blue database's application code.

**3. Module Design Principles**

* **Independence:** Each module should ideally manage a single, well-defined piece of infrastructure or a specific action.
* **Idempotency:** Running a module multiple times with the same inputs should result in the same state without adverse side effects.
* **Parameterization:** Modules should accept parameters (inputs) to allow for customization and reuse across different environments or contexts.
* **Clear Outputs:** Modules should clearly define their outputs, which can then serve as inputs for subsequent modules in a pipeline.
* **Error Handling:** Each module should incorporate robust error handling and logging.
* **Version Control:** Each module's code/configuration should be version-controlled (e.g., Git).

**4. Core Deployment Modules**

Below is a breakdown of the key modules derived from the Blue-Green deployment flowchart. For each module, we'll describe its purpose, typical inputs, and expected outputs.

**4.1. Module: KMS Key Management for Green Environment**

* **Purpose:** To establish the appropriate Key Management Service (KMS) key configuration for the new "Green" database environment. This module addresses the need for an independent KMS key, especially crucial when the existing KMS key used by the "Blue" environment is tightly integrated with the database's application code. It handles determining if an existing suitable key can be reused, if a new key needs to be created (acting as a "decoupling" process from the tightly integrated source key), or if a new key needs to be provisioned as a standard practice.
* **Inputs:**
  + kms\_action\_type: (DECOUPLE\_CREATE\_NEW\_KEY, REUSE\_EXISTING\_KEY, CREATE\_NEW\_KEY\_STANDARD).
  + source\_db\_kms\_key\_id\_or\_arn: (Optional, but useful if kms\_action\_type is DECOUPLE\_CREATE\_NEW\_KEY) Identifier or ARN of the KMS key associated with the Blue database, indicating a tightly coupled scenario.
  + existing\_kms\_key\_id\_or\_arn\_to\_reuse: (Required if kms\_action\_type is REUSE\_EXISTING\_KEY) Identifier or ARN of an existing KMS key that is suitable for the Green environment.
  + new\_key\_alias: (Required if kms\_action\_type is DECOUPLE\_CREATE\_NEW\_KEY or CREATE\_NEW\_KEY\_STANDARD) Alias for a new KMS key if one is to be created.
  + key\_description: (Optional) Description for a new KMS key.
  + target\_region: AWS region where the KMS key will be managed.
  + tags: Key-value tags for the KMS key (if created).
* **Outputs:**
  + green\_db\_kms\_key\_arn: The ARN of the KMS key that will be used for the Green database.
  + kms\_key\_management\_status: Status of the KMS operation (e.g., DECOUPLED, REUSED, CREATED).
* **Reusability:** Highly reusable module for managing KMS keys for any encrypted AWS resource, adapting to different integration scenarios.

**4.2. Module: Network Load Balancer (NLB) Configuration**

* **Purpose:** To provision and configure Network Load Balancers and associated target groups for routing application traffic. This module prepares the entry point for the "Green" environment.
* **Inputs:**
  + nlb\_name: Name for the NLB.
  + vpc\_id: VPC ID where the NLB will reside.
  + subnet\_ids: List of subnet IDs for the NLB.
  + target\_group\_name: Name for the associated target group.
  + target\_group\_protocol: Protocol for the target group (e.g., TCP, TLS).
  + target\_group\_port: Port for the target group.
  + health\_check\_path: Path for health checks (if applicable).
* **Outputs:**
  + nlb\_arn: ARN of the created NLB.
  + target\_group\_arn: ARN of the created target group.
* **Reusability:** Essential for any service requiring load balancing; can be used for deploying new services, updating existing ones, or during DR scenarios.

**4.3. Module: RDS Oracle Green Environment Provisioning (Empty Instance)**

* **Purpose:** To provision a *new, empty* Amazon RDS Oracle database instance. This module is specifically used as the initial step in the **Export/Import-Based Green Environment Creation path** to set up the target database for data loading.
* **Inputs:**
  + db\_instance\_identifier: Unique identifier for the DB instance.
  + db\_instance\_class: Instance type (e.g., db.m5.large).
  + engine\_version: Oracle engine version (e.g., 19.0.0.0.ru-2023-01.rur-2023-01.r1).
  + allocated\_storage: Storage in GB.
  + vpc\_security\_group\_ids: List of security group IDs.
  + db\_subnet\_group\_name: DB Subnet Group name.
  + master\_username: Master database username.
  + master\_password: Master database password (preferably retrieved securely).
  + kms\_key\_arn: ARN of the KMS key for encryption (if applicable).
  + multi\_az\_enabled: Boolean to enable Multi-AZ (can be enabled later post-promotion).
  + parameter\_group\_name: Name of the DB parameter group.
  + option\_group\_name: Name of the DB option group.
* **Outputs:**
  + db\_instance\_arn: ARN of the provisioned RDS instance.
  + db\_endpoint: Endpoint address of the RDS instance.
* **Reusability:** Can be used to provision any new, empty RDS Oracle database instance.

**4.4. Module: S3 Bucket Association for RDS**

* **Purpose:** To associate an S3 bucket with an RDS instance, enabling features like Data Pump import/export, audit logs, or backup transfers.
* **Inputs:**
  + db\_instance\_identifier: Identifier of the RDS instance.
  + s3\_bucket\_name: Name of the S3 bucket to associate.
  + iam\_role\_arn: ARN of the IAM role with S3 access permissions.
* **Outputs:** (Confirmation of association)
* **Reusability:** Useful whenever an RDS instance needs to interact with S3.

**4.5. Module: RDS Policies Application**

* **Purpose:** To apply necessary IAM policies to the RDS instance or associated roles, granting it required permissions for operations (e.g., connecting to other services, fine-grained access control).
* **Inputs:**
  + db\_instance\_identifier: Identifier of the RDS instance.
  + policy\_arns: List of IAM policy ARNs to attach.
* **Outputs:** (Confirmation of policy application)
* **Reusability:** General utility module for managing IAM policies for RDS.

**4.6. Module: Database Schema & Access Creation (Green)**

* **Purpose:** To establish the necessary logical database structure and access permissions within the "Green" Oracle database. This includes creating application schemas, users, roles, sysgrants, and configuring proxy access. This module prepares the database with its required logical components regardless of how the data itself is populated (via import or replication).
* **Implementation Note:** While the orchestration of this step can be managed by IaC tools (e.g., triggering scripts), the actual creation of schemas, users, and grants typically involves executing **SQL scripts** directly against the database or using database-specific client tools. This is generally *not* a direct Terraform resource creation for the schema objects themselves but rather an execution step.
* **Inputs:**
  + db\_endpoint: Endpoint of the Green RDS instance.
  + master\_username: Master database username.
  + master\_password: Master database password.
  + schemas\_to\_create: List of schema names and their properties.
  + users\_to\_create: List of user details (username, password, default tablespace).
  + grants\_to\_apply: List of grants (e.g., CONNECT, RESOURCE, CREATE SESSION) to apply to users/roles.
  + proxy\_users\_to\_create: List of proxy user configurations.
  + sql\_scripts\_path: (Optional) Path to SQL scripts for schema/user creation.
* **Outputs:** (Confirmation of schema/user creation, logs of script execution)
* **Reusability:** Highly reusable for initializing the logical structure and access control of any new Oracle database.

**4.7. Module: Export Data from Source DB (Blue)**

* **Purpose:** To perform a data export (e.g., Oracle Data Pump expdp) from the "Blue" (source) database. This is the first step when using the Export/Import path.
* **Inputs:**
  + source\_db\_endpoint: Endpoint of the Blue RDS instance.
  + source\_db\_username: Database username with export privileges.
  + source\_db\_password: Database password.
  + export\_schemas: List of schemas to export.
  + s3\_bucket\_name: S3 bucket for dump files.
  + export\_directory\_name: Directory object name in DB.
* **Outputs:**
  + dump\_file\_names: List of generated dump file names in S3.
  + export\_scn: The SCN at which the export was started.
* **Reusability:** General utility for backing up or extracting data from an Oracle database.

**4.8. Module: Transfer Dump Files to S3**

* **Purpose:** To manage the transfer of database dump files (e.g., generated by expdp) to an S3 bucket. While expdp can directly write to S3 in RDS, this module can handle external transfers or validation.
* **Inputs:**
  + local\_dump\_paths: Paths to dump files on a transient host (if not direct S3 export).
  + s3\_bucket\_name: Target S3 bucket.
  + s3\_prefix: Optional S3 prefix for organization.
* **Outputs:**
  + s3\_dump\_paths: S3 URIs of the uploaded dump files.
* **Reusability:** Generic file transfer module to S3.

**4.9. Module: Import Data to Green DB**

* **Purpose:** To import data (e.g., Oracle Data Pump impdp) from S3 dump files into the "Green" RDS Oracle instance. This populates the "Green" database when using the Export/Import path.
* **Inputs:**
  + green\_db\_endpoint: Endpoint of the Green RDS instance.
  + green\_db\_username: Database username with import privileges.
  + green\_db\_password: Database password.
  + s3\_dump\_paths: S3 URIs of the dump files to import.
  + import\_schemas: List of schemas to import (can be remapped).
  + import\_directory\_name: Directory object name in DB.
* **Outputs:** (Confirmation of successful import)
* **Reusability:** General utility for restoring or populating an Oracle database from dump files.

**4.10. Module: Database Validation & Missing Objects Creation**

* **Purpose:** To perform post-import validation (e.g., row counts, checksums) and identify/create any database objects (indexes, constraints, triggers) that might not have been included or correctly created during the bulk import. This ensures the "Green" database is fully consistent.
* **Inputs:**
  + source\_db\_endpoint: Endpoint of the Blue DB.
  + green\_db\_endpoint: Endpoint of the Green DB.
  + validation\_scripts: SQL scripts for validation queries.
  + missing\_objects\_scripts: SQL scripts to create missing objects.
* **Outputs:** (Validation report, confirmation of object creation)
* **Reusability:** Critical for ensuring data integrity and completeness after database migrations or copies.

**4.11. Module: Create InRegion Read Replica (IRR)**

* **Purpose:** To provision an InRegion Read Replica *from* the "Blue" RDS Oracle instance. **This module is the core action for creating the "Green" RDS database instance itself when choosing the Read Replica path.** The replica continuously replicates data, providing a near real-time copy.
* **Inputs:**
  + source\_db\_instance\_identifier: Identifier of the Blue RDS instance.
  + replica\_instance\_identifier: Identifier for the new Read Replica (this will become the "Green" DB).
  + db\_instance\_class: Instance type for the replica.
  + allocated\_storage: Storage for the replica.
  + vpc\_security\_group\_ids: Security group IDs for the replica.
  + db\_subnet\_group\_name: DB Subnet Group name for the replica.
  + kms\_key\_arn: KMS key for encryption (if applicable).
* **Outputs:**
  + read\_replica\_arn: ARN of the created Read Replica.
  + read\_replica\_endpoint: Endpoint of the Read Replica.
* **Reusability:** Can be used to create any RDS Read Replica for scaling reads or for DR purposes.

**4.12. Module: Promote InRegion Read Replica (IRR)**

* **Purpose:** To promote a read replica to a standalone, writable RDS instance. This is a critical step when using the IRR path to establish the "Green" primary database.
* **Inputs:**
  + read\_replica\_identifier: Identifier of the Read Replica to promote.
* **Outputs:**
  + promoted\_db\_instance\_arn: ARN of the promoted instance.
  + promoted\_db\_endpoint: Endpoint of the promoted instance.
  + promotion\_scn: The SCN at which the promotion occurred (derived from CloudWatch/DB logs).
* **Reusability:** Specific to the promotion of RDS read replicas.

**4.13. Module: GoldenGate Hub Primary Creation**

* **Purpose:** To provision and configure the GoldenGate Hub Primary instance (e.g., on an EC2 instance), which will host the Extract process for the "Blue" environment, capturing ongoing changes.
* **Inputs:**
  + instance\_type: EC2 instance type.
  + ami\_id: AMI ID for the GoldenGate server.
  + security\_group\_ids: Security group IDs.
  + subnet\_id: Subnet ID.
  + key\_pair\_name: EC2 key pair for access.
  + db\_endpoint: Endpoint of the source DB (Blue).
  + db\_username: DB username for GG connection.
  + db\_password: DB password.
  + initial\_scn: The starting SCN for the extract (from export or source DB's current SCN).
* **Outputs:**
  + gg\_primary\_instance\_id: EC2 Instance ID.
  + gg\_primary\_ip: Private IP of the instance.
  + gg\_primary\_extract\_status: Status of the extract process.
* **Reusability:** Part of any GoldenGate setup, can be reused for other replication scenarios.

**4.14. Module: GoldenGate Hub Secondary Creation**

* **Purpose:** To provision and configure the GoldenGate Hub Secondary instance, which will host the Replicat process for the "Green" environment, applying changes captured from "Blue".
* **Inputs:** (Similar to Primary, but configured for the Green environment)
  + green\_db\_endpoint: Endpoint of the Green DB.
  + green\_db\_username: Green DB username for GG connection.
  + green\_db\_password: Green DB password.
  + primary\_hub\_ip: IP of the Primary GG Hub for distribution path.
* **Outputs:**
  + gg\_secondary\_instance\_id: EC2 Instance ID.
  + gg\_secondary\_ip: Private IP of the instance.
* **Reusability:** Part of any GoldenGate setup.

**4.15. Module: GoldenGate Distribution Path Management**

* **Purpose:** To create, start, or manage the GoldenGate Distribution Path between two GoldenGate instances (e.g., Primary Hub to Secondary Hub).
* **Inputs:**
  + source\_gg\_hub\_endpoint: Endpoint of the source GG Hub.
  + target\_gg\_hub\_endpoint: Endpoint of the target GG Hub.
  + extract\_name: Name of the extract process on the source.
  + path\_name: Name for the distribution path.
* **Outputs:** (Status of the distribution path)
* **Reusability:** Core GoldenGate configuration.

**4.16. Module: GoldenGate Replicat Management**

* **Purpose:** To create, start, or manage GoldenGate Replicat processes on a target GoldenGate Hub, applying changes to the destination database.
* **Inputs:**
  + target\_gg\_hub\_endpoint: Endpoint of the target GG Hub.
  + target\_db\_endpoint: Endpoint of the target database.
  + replicat\_name: Name for the replicat process.
  + trail\_file\_name: Name of the trail file to process.
* **Outputs:** (Status of the replicat process)
* **Reusability:** Core GoldenGate configuration.

**4.17. Module: GoldenGate Lag Monitoring**

* **Purpose:** To continuously monitor the replication lag between GoldenGate Extract and Replicat processes, ensuring data synchronization is within acceptable limits before cutover.
* **Inputs:**
  + gg\_primary\_endpoint: Primary GG Hub endpoint.
  + gg\_secondary\_endpoint: Secondary GG Hub endpoint.
  + threshold\_seconds: Acceptable lag threshold.
* **Outputs:** current\_lag\_seconds, is\_synced (boolean)
* **Reusability:** Essential for any GoldenGate-based replication.

**4.18. Module: Application Traffic Switchover**

* **Purpose:** To redirect application traffic from the "Blue" environment to the "Green" environment. This typically involves updating DNS records or modifying NLB/ALB target group weights to point to the new, validated environment.
* **Inputs:**
  + nlb\_arn: ARN of the NLB.
  + blue\_target\_group\_arn: ARN of the "Blue" target group.
  + green\_target\_group\_arn: ARN of the "Green" target group.
  + primary\_target\_group\_arn: The target group to set to 100% weight.
  + secondary\_target\_group\_arn: The target group to set to 0% weight.
  + dns\_record\_name: (Optional) DNS record to update.
  + dns\_new\_alias: (Optional) New alias target.
* **Outputs:** (Confirmation of traffic switch)
* **Reusability:** Fundamental for any blue-green or canary deployment.

**4.19. Module: Secrets Regeneration**

* **Purpose:** To regenerate or update secrets (e.g., database credentials, API keys) in a secrets management service (e.g., AWS Secrets Manager) for the "Green" environment, ensuring proper security for the new environment.
* **Inputs:**
  + secret\_id: Identifier of the secret to update.
  + new\_secret\_value: The new secret value.
  + rotation\_lambda\_arn: (Optional) ARN of a Lambda for secret rotation.
* **Outputs:** (Confirmation of secret update)
* **Reusability:** General security module for managing application secrets.

**4.20. Module: Convert RDS to Multi-AZ**

* **Purpose:** To modify an existing RDS instance to enable Multi-Availability Zone (Multi-AZ) deployment for high availability, typically applied to the new "Green" primary.
* **Inputs:**
  + db\_instance\_identifier: Identifier of the RDS instance.
  + apply\_immediately: Boolean to apply changes immediately or during maintenance window.
* **Outputs:** (Confirmation of Multi-AZ status)
* **Reusability:** Useful for any RDS instance needing enhanced availability.

**4.21. Module: Environment Decommissioning**

* **Purpose:** To safely tear down and delete resources associated with an old "Blue" environment after a successful cutover and stabilization period. This is vital for cost management and resource hygiene.
* **Inputs:**
  + resources\_to\_delete: List of resource ARNs or identifiers (e.g., old RDS, old NLB, old EC2 instances, old GG Hubs).
  + force\_delete: Boolean to allow immediate deletion.
* **Outputs:** (Confirmation of resource deletion)
* **Reusability:** Crucial for cost management and resource hygiene.

**5. Orchestration of Modules (Complete Blue-Green Pipeline)**

These independent modules can be orchestrated using a CI/CD pipeline tool like **Spinnaker**. Spinnaker's pipeline stages can directly invoke these modules (e.g., through Jenkins jobs, custom scripts, or directly interacting with cloud provider APIs), defining the sequence of module execution, passing outputs from one module as inputs to the next, and handling decision points and error recovery. Its robust deployment strategies (like blue/green) are a natural fit for this modular approach.

**5.1. Orchestration Path A: Read Replica-Based Green Environment Creation**

This path is typically preferred for in-place database upgrades or rapid cloning within the same database engine/version, leveraging native replication for initial data seeding.

**Spinnaker Pipeline Stages for Path A:**

1. **Start Deployment Stage**
2. **KMS Key Setup Stage:**
   * **Action:** Invoke "KMS Key Management for Green Environment" module (e.g., kms\_action\_type could be REUSE\_EXISTING\_KEY or CREATE\_NEW\_KEY\_STANDARD if decoupling isn't needed, or DECOUPLE\_CREATE\_NEW\_KEY if the Blue DB's key is tightly integrated with its code).
3. **NLB Configuration Stage** (for Green environment)
4. **Create InRegion Read Replica (IRR) Stage:**
   * **Action:** Invoke "Create InRegion Read Replica (IRR)" module. **This stage directly creates the Green database instance as a replica of the Blue database.**
5. **S3 Bucket Association for RDS Stage** (Associate S3 with the newly created Green Read Replica)
6. **RDS Policies Application Stage** (Apply policies to the Green Read Replica/future primary)
7. **Promote InRegion Read Replica (IRR) Stage:**
   * **Action:** Invoke "Promote InRegion Read Replica (IRR)" module to make the replicated Green DB writable.
8. **Database Schema & Access Creation (Green) Stage:** (Apply any post-promotion schema/user changes not handled by replication)
9. **GoldenGate Hub Primary Creation Stage** (Configured to extract from Blue DB)
10. **GoldenGate Hub Secondary Creation Stage** (Configured to replicate to Green DB)
11. **GoldenGate Distribution Path Management Stage** (Blue to Green)
12. **GoldenGate Replicat Management Stage** (for Green environment)
13. **GoldenGate Lag Monitoring Stage** (Looping/polling stage until lag is within acceptable thresholds)
14. **Application Traffic Switchover Stage** (Manual or Automated Cutover to Green)
15. **GoldenGate Extract Management Stage** (for Green to Blue replication, for rollback/future cycles)
16. **GoldenGate Replicat Management Stage** (for Blue environment, for rollback/future cycles)
17. **Secrets Regeneration Stage** (for Green environment)
18. **Convert RDS to Multi-AZ Stage** (for the new Green primary DB)
19. **Environment Decommissioning Stage** (for the old Blue environment, after stabilization period)

**5.2. Orchestration Path B: Export/Import-Based Green Environment Creation**

This path provides more control over the initial data transfer and is suitable for cross-version database migrations, selective data transfers, or scenarios where native replication is not feasible.

**Spinnaker Pipeline Stages for Path B:**

1. **Start Deployment Stage**
2. **KMS Key Setup Stage:**
   * **Action:** Invoke "KMS Key Management for Green Environment" module (e.g., kms\_action\_type could be DECOUPLE\_CREATE\_NEW\_KEY if the Blue DB's key is tightly integrated with its code, or CREATE\_NEW\_KEY\_STANDARD otherwise).
3. **NLB Configuration Stage** (for Green environment)
4. **RDS Oracle Green Environment Provisioning (Empty Instance) Stage:**
   * **Action:** Invoke "RDS Oracle Green Environment Provisioning (Empty Instance)" module. **This stage creates a new, empty Green database instance to serve as the target for the import.**
5. **S3 Bucket Association for RDS Stage** (Associate S3 with the newly provisioned Green DB)
6. **RDS Policies Application Stage** (Apply policies to the Green DB)
7. **Database Schema & Access Creation (Green) Stage:** (Initial schemas, users, and grants for the new, empty Green DB)
8. **Export Data from Source DB (Blue) Stage:**
   * **Action:** Invoke "Export Data from Source DB (Blue)" module to dump data.
9. **Transfer Dump Files to S3 Stage:**
   * **Action:** Invoke "Transfer Dump Files to S3" module.
10. **Import Data to Green DB Stage:**
    * **Action:** Invoke "Import Data to Green DB" module to load data into the Green DB.
11. **Database Validation & Missing Objects Creation Stage:**
    * **Action:** Invoke "Database Validation & Missing Objects Creation" module.
12. **GoldenGate Hub Primary Creation Stage** (Configured to extract from Blue DB, picking up from export SCN)
13. **GoldenGate Hub Secondary Creation Stage** (Configured to replicate to Green DB)
14. **GoldenGate Distribution Path Management Stage** (Blue to Green)
15. **GoldenGate Replicat Management Stage** (for Green environment)
16. **GoldenGate Lag Monitoring Stage** (Looping/polling stage until lag is within acceptable thresholds)
17. **Application Traffic Switchover Stage** (Manual or Automated Cutover to Green)
18. **GoldenGate Extract Management Stage** (for Green to Blue replication, for rollback/future cycles)
19. **GoldenGate Replicat Management Stage** (for Blue environment, for rollback/future cycles)
20. **Secrets Regeneration Stage** (for Green environment)
21. **Convert RDS to Multi-AZ Stage** (for the new Green primary DB)
22. **Environment Decommissioning Stage** (for the old Blue environment, after stabilization period)

**6. Implementation Tools**

* **Infrastructure as Code (IaC):** Terraform
* **Scripting:** Python, Bash (for custom logic within Spinnaker stages).
* **CI/CD Pipelines:** **Spinnaker** (for orchestration, pipeline definition, and automated execution).
* **Secrets Management:** AWS Secrets Manager
* **Monitoring & Logging:** AWS CloudWatch,Custom code.

**Appendix A: Flowchart Legend (Based on image content)**

* **Green Boxes:** Represent database-related actions or states.
* **Blue Boxes:** Represent infrastructure or general deployment actions.
* **Yellow Boxes:** Represent GoldenGate specific actions.
* **Diamonds:** Represent decision points.
* **NLB:** Network Load Balancer
* **KMS:** Key Management Service
* **RDS:** Relational Database Service
* **S3:** Simple Storage Service
* **GG:** GoldenGate
* **SCN:** System Change Number
* **IRR:** InRegion Read Replica
* **Multi A/Z:** Multi-Availability Zone