1. AWS Provider Configuration (providers.tf)

Purpose:
Authenticates Terraform with AWS and specifies the region where resources will be deployed.

Why It's Needed:

- Without this, Terraform cannot interact with your AWS account.
- The region variable allows environment-specific deployments (dev/prod in different regions).

---

2. S3 Backend for Remote State (backend.tf)

Purpose:
Stores Terraform state remotely in S3 and locks it using DynamoDB to prevent conflicts.
Key Code:

Why It's Needed:

- **Collaboration**: Teams can share the same state file.
- **Safety**: State is encrypted, and DynamoDB prevents concurrent edits.

---

3. IAM Role for Lambda (iam.tf)

Purpose:
Grants the Lambda function permissions to interact with AWS services (SQS, DynamoDB, CloudWatch, X-Ray).

Why It's Needed:

- Lambda needs explicit permissions to access SQS, DynamoDB, and publish logs/traces.
- **Least Privilege**: Restricts Lambda to only required actions
  (e.g., sqs:ReceiveMessage, dynamodb:PutItem).

---

4. Lambda Function (lambda.tf)

Purpose:
Processes messages from SQS and writes data to DynamoDB.

Why It's Needed:

- Acts as the **event-driven bridge** between SQS and DynamoDB.
- Uses environment variables to avoid hardcoding resource names.

5. SQS Queues (Main + DLQ) (sqs.tf)

Purpose:
Decouples message producers and consumers, with a dead-letter queue (DLQ) for failed messages.

Why It's Needed:

- **Resilience**: DLQ captures messages that fail processing after retries.
- **Observability**: Isolate failures for debugging.

---

6. DynamoDB Table (dynamodb.tf)

Purpose:
Stores data from the Lambda function with infinite scaling and TTL for automatic cleanup.

Why It's Needed:

- **Scalability**: Handles high-throughput workloads.
- **Cost Efficiency**: Pay-per-request pricing.
- **TTL**: Automatically deletes stale data.

---

7. Environment Variables (environments/*.tfvars)

Purpose:
Separates configuration for development (dev) and production (prod) environments.

Why It's Needed:

- Avoids hardcoding values like region or resource names.
- Enables safe promotion of code from dev to prod.

Workflow Summary

1. **SQS Trigger**: Messages arrive in the main queue.
2. **Lambda Execution**: Processes messages and writes to DynamoDB.
3. **Retry Logic**: Failed messages move to the DLQ after 3 attempts.
4. **State Management**: Terraform state is stored securely in S3.

Key Security Practices

1. **Least Privilege**: IAM roles restrict Lambda to only necessary actions.

2. **Encryption**: S3 state files are encrypted.
3. **Locking**: DynamoDB prevents concurrent state modifications.