Multi-Agent Compliance Analysis Stack

This CDK stack deploys a multi-agent compliance analysis system that uses Amazon Bedrock AgentCore to perform regulatory compliance assessments. The system processes compliance analysis jobs through a Fargate-based workflow that coordinates multiple specialized agents (lawyer, writer, and auditor) to generate comprehensive compliance reports.

Overview

The stack creates:

- **ECS Fargate Service**: Processes compliance analysis jobs from an SQS queue
- **VPC with Endpoints**: Secure networking with DynamoDB and S3 VPC endpoints
- **Multi-Agent Workflow**: Coordinates lawyer, writer, and auditor agents via Bedrock AgentCore
- **Parameter Store Integration**: Securely manages agent ARNs and resource names
- **CloudWatch Logging**: Comprehensive logging for monitoring and debugging

Prerequisites

Before deploying this stack, ensure you have:

- **AWS CLI** configured with appropriate permissions
- **Docker** installed and running
- **Python 3.12+** with pip
- **A CDK bootstrapped** AWS account
- **A Python virtual environment** with the dependencies installed
- **A Bedrock AgentCore execution role ARN** stored in the environment variable $AGENT_CORE_ROLE_ARN
- **Agent's runtime ARN** stored in the environment variables $LAWYER_AGENT_RUNTIME_ARN, $AUDITOR_AGENT_RUNTIME_ARN, $WRITER_AGENT_RUNTIME_ARN

Required AWS Resources

This stack depends on external resources that must be created first:

1. **AgentCore Agents**: Deploy the following agents using the AgentCore IAM role stack:
   - Lawyer Agent ARN
   - Writer Agent ARN
   - Auditor Agent ARN
2. **Knowledge ingestion stack**: The knowledge ingestion stack should have been previously deployed

Deployment

Deploy the stack by providing the required parameters:

```
cd cdk

# Deploy with all required parameters
cdk deploy MultiAgentComplianceAnalysis \
  --parameters lawyerAgentARN=$LAWYER_AGENT_RUNTIME_ARN \
  --parameters auditorAgentARN=$AUDITOR_AGENT_RUNTIME_ARN \
  --parameters writerAgentARN=$WRITER_AGENT_RUNTIME_ARN \
  --require-approval=never
```

Scaling Configuration

The Fargate service is configured with:

- **Memory**: 1024 MiB
- **Max Scaling Capacity**: 5 tasks
- **Min Healthy Percent**: 70%
- **Log Retention**: 30 days

Monitoring and Troubleshooting

CloudWatch Logs

Monitor the application through CloudWatch log groups:

- **Fargate Logs**: `/aws/ecs/FargateComplianceAnalysisWorkflowLogs`
- **VPC Flow Logs**: Auto-generated log group for network traffic

Common Issues

1. **Container fails to start**:
    - Check CloudWatch logs for Python import errors
    - Verify all required parameters are provided
    - Ensure Docker image builds successfully
2. **Permission errors**:
    - Verify agent ARNs are correct and accessible
    - Check IAM roles have necessary Bedrock AgentCore permissions
    - Ensure VPC endpoints are properly configured
3. **SQS message processing fails**:
    - Check DynamoDB table permissions
    - Verify S3 bucket write permissions
    - Review KMS key decrypt permissions

Useful Commands

```
# View stack outputs
aws cloudformation describe-stacks --stack-name
MultiAgentComplianceAnalysis --query 'Stacks[0].Outputs'

# Check Fargate service status
aws ecs describe-services --cluster <cluster-name> --services <service-
name>

# View recent logs
aws logs tail /aws/ecs/FargateComplianceAnalysisWorkflowLogs --follow
```

Security Considerations

- **VPC Isolation**: All resources run in a private VPC
- **VPC Endpoints**: Secure access to AWS services without internet gateway
- **KMS Encryption**: SQS messages encrypted with customer-managed keys
- **IAM Least Privilege**: Minimal permissions for each component
- **Container Security**: Non-root container execution
- **Network Security**: VPC Flow Logs enabled for monitoring

Cleanup

To remove all resources:

```
cd cdk
cdk destroy MultiAgentComplianceAnalysis
```

Note: This will delete all resources created by the stack, but external dependencies (DynamoDB table, S3 bucket, etc.) will remain unless their stack is deleted.