SAM Deployment Debugging Guide

Component Relationships Overview

1. AWS CodeBuild (CI/CD Orchestrator)

- **Purpose**: Executes the entire build and deployment pipeline
- **Dependencies**: buildspec.yaml, Service Role, Docker Engine
- **Controls**: Build environment, resource allocation, permissions
- **Issues**: Docker unavailable, insufficient IAM permissions

2. buildspec.yaml (Build Instructions)

- **Purpose**: Defines build phases and commands
- **Dependencies**: SAM CLI, Python runtime, Docker
- **Controls**: Build sequence, environment setup, deployment commands
- **Current State**: ✅ Correctly configured

3. SAM (Serverless Application Model)

- **Purpose**: Abstracts serverless infrastructure management
- **Dependencies**: template.yaml, Docker, CloudFormation, ECR
- **Controls**: Resource provisioning, container building, deployment
- **Issues**: Cannot build containers without Docker

4. template.yaml (Infrastructure Definition)

- **Purpose**: Defines AWS resources and their configurations
- **Dependencies**: SAM Transform, CloudFormation
- **Controls**: Lambda function specs, IAM policies, resource relationships
- **Current State**: ✅ Correctly configured

5. Docker (Container Runtime)

- **Purpose**: Builds container images for Lambda functions
- **Dependencies**: Dockerfile, privileged mode, Docker daemon
- **Controls**: Image creation, layer management, registry operations
- **Issues**: ❌ Not available in CodeBuild environment

6. IAM (Identity and Access Management)

- **Purpose**: Controls access to AWS resources

- **Dependencies**: Service roles, policies, trust relationships
- **Controls**: CodeBuild permissions, Lambda execution rights
- **Issues**: ❌ Missing CloudFormation permissions

7. Lambda (Serverless Function)

- **Purpose**: Executes your application code
- **Dependencies**: Container image, execution role, runtime environment
- **Controls**: Function configuration, event processing, scaling
- **Status**: 🔄 Pending creation

8. ECR (Elastic Container Registry)

- **Purpose**: Stores container images
- **Dependencies**: Docker images, IAM permissions
- **Controls**: Image versioning, access control, lifecycle policies
- **Status**: 🔄 Pending creation

9. EKS (Elastic Kubernetes Service)

- **Purpose**: Alternative container orchestration (not used in current setup)
- **Relationship**: Could use same container images as Lambda
- **Dependencies**: Kubernetes cluster, worker nodes
- **Status**: ❌ Not applicable to current SAM deployment

Debug Flow Analysis

Issue 1: Docker Unavailability

```
Build Phase → SAM Build → Docker Build → ❌ FAIL
```

Root Cause Chain:

```
CodeBuild Environment → Docker Engine → Container Build → ECR Push
     ↓ (Missing)          ↓ (Failed)      ↓ (Blocked)      ↓ (Blocked)
Non-privileged mode → No Docker Access → Build Failure → Deploy Blocked
```

Solution Path:

1. Enable privileged mode in CodeBuild project
2. Ensure Docker daemon is accessible
3. Add ECR authentication commands

4. Verify container build process

Issue 2: IAM Permissions

```
Deploy Phase → CloudFormation → Create ChangeSet → ❌ FAIL
```

Root Cause Chain:

```
CodeBuild Role → IAM Policies → CloudFormation Access → Resource Creation
     ↓              ↓              ↓ (Missing)           ↓ (Blocked)
Service Role → Limited Policies → Access Denied → Deploy Failure
```

Solution Path:

1. Identify CodeBuild service role
2. Attach CloudFormation permissions
3. Add ECR, Lambda, S3, IAM permissions
4. Verify cross-service access

Debugging Checklist

✅ Currently Working

- [x] Source code structure
- [x] buildspec.yaml syntax
- [x] template.yaml configuration
- [x] SAM CLI installation
- [x] Python runtime setup
- [x] Template parsing

❌ Issues to Fix

- [ ] **Docker Engine Access**
- [ ] Enable privileged mode in CodeBuild
- [ ] Verify Docker daemon availability
- [ ] Add ECR login commands
- [ ] **IAM Permissions**
- [ ] CloudFormation permissions
- [ ] ECR access permissions
- [ ] Lambda management permissions
- [ ] S3 bucket permissions

🔄 Pending Validation

- [ ] Container image build
- [ ] ECR repository creation
- [ ] Lambda function deployment
- [ ] IAM role creation
- [ ] Function configuration

Component Interaction Flow

```
Developer Code → CodeBuild → Docker → SAM → CloudFormation → AWS Resources
   ↓       ↓     ↓    ↓      ↓       ↓
Source Control → CI/CD → Container → IaC → Resource Manager → Lambda/ECR/IAM
```

Alternative Approaches

If Docker Issues Persist:

1. **Use ZIP deployment** instead of container images
2. **Switch to AWS Lambda Layers** for dependencies
3. **Use CodeDeploy** with pre-built images

If IAM Issues Persist:

1. **Use CloudFormation directly** instead of SAM
2. **Pre-create required resources** manually
3. **Use AWS CDK** for infrastructure deployment

Monitoring and Validation

Success Indicators:

- Docker build completes successfully
- Container image pushed to ECR
- CloudFormation stack created
- Lambda function deployed and configured
- Function can be invoked successfully

Failure Points to Monitor:

- Docker daemon connectivity
- ECR authentication
- CloudFormation changeset creation
- IAM role assumptions

- Lambda function initialization