Configuration Drift in AWS CloudFormation

Configuration drift occurs when the actual runtime configuration of an environment diverges from its expected configuration as defined in code or templates. In AWS CloudFormation, this happens when manual changes are made to the resources that a CloudFormation stack manages, without updating the stack's template. This can lead to inconsistencies, making further automation and management challenging, as the environment no longer matches its template.

---

1. Understanding Configuration Drift

a. What Causes Configuration Drift?

Configuration drift in AWS CloudFormation can occur due to:

- **Manual Changes**: Direct modifications to resources through the AWS Management Console, CLI, or SDKs, bypassing CloudFormation.
- **Automated Processes**: External scripts or services modifying the environment outside of the CloudFormation process.
- **Resource Evolution**: Certain AWS resources might change due to internal AWS processes, or updates that are not captured in the CloudFormation template.

b. Impacts of Configuration Drift

- **Inconsistencies**: The stack's behavior might become unpredictable, especially during updates or when replicating environments.
- **Deployment Failures**: Updates via CloudFormation might fail if the actual configuration doesn’t match the expected state defined in the template.
- **Security and Compliance Risks**: Untracked changes might violate compliance policies or introduce security weaknesses.

---

2. Detecting Configuration Drift in AWS CloudFormation

AWS CloudFormation provides a feature called **Drift Detection** that helps you identify resources in your stack that have drifted from their expected configuration.

a. How Drift Detection Works

- CloudFormation compares the current configuration of resources within a stack to the expected configurations defined in the stack's template and parameters.
- The drift status of each resource is reported as:
    - **IN_SYNC**: The resource's current configuration matches the template.
    - **MODIFIED**: The resource has been changed outside of CloudFormation.
    - **DELETED**: The resource was deleted outside of CloudFormation.

b. Performing Drift Detection

To perform drift detection, you can use the AWS Management Console, AWS CLI, or AWS SDKs. Here's how you can initiate drift detection using the AWS CLI:

```
aws cloudformation detect-stack-drift --stack-name example-stack
```

This command returns a `StackDriftDetectionId` that you can use to check the status of the drift detection operation:

```
aws cloudformation describe-stack-drift-detection-status
--stack-drift-detection-id <detection-id>
```

---

3. Handling Configuration Drift

a. Resolving Drift

Once drift is detected, you can take the following steps to resolve it:

- **Manual Reconciliation**: Manually update the resource to match its expected configuration as per the CloudFormation template.
- **Update Stack**: If the changes are necessary, update the CloudFormation template to incorporate the changes and re-deploy the stack.
- **Resource Import**: If resources were created or modified outside of CloudFormation, use the import functionality to bring them under CloudFormation management.

b. Preventive Measures

- **Restricted Access**: Limit permissions to modify resources only through CloudFormation by using AWS Identity and Access Management (IAM) policies.

- **Automation Practices**: Encourage the use of automated pipelines that include infrastructure as code, ensuring all changes are made through CloudFormation.
- **Regular Audits**: Conduct regular drift detection audits to identify and address drift before it impacts operations.

---

4. Best Practices for Managing Configuration Drift

- **Educate Teams**: Ensure that all team members understand the importance of making changes through CloudFormation to maintain stack integrity.
- **Monitoring and Alerts**: Set up monitoring and alerts for unauthorized changes to resources using AWS CloudTrail and AWS Config.
- **Version Control**: Keep your CloudFormation templates in version-controlled repositories and review changes through pull requests.