1) Why Real-Time Drift Detection is Not Possible in AWS:

AWS CloudFormation Drift Detection is a manual or scheduled process—it is not an event-driven service. AWS does not provide real-time drift monitoring natively.

CloudFormation does not automatically detect drift when a resource is modified outside the stack. The only way to check for drift is by running detect_stack_drift on demand or scheduling it. We are using this on our script so we only need to schedule it using Event bridge and which is why I assume going with smaller set of stacks would be better than going with larger sets despite the client requirements.

link: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/detect-drift-stack.html

Limitation: AWS CloudFormation drift detection is a scheduled process, not real-time. AWS does not provide automatic detection or remediation of drift, so we need to use a combination of scheduled checks and CloudTrail logs.

2) Why AWS Does Not Provide Event-Based Drift Detection:

AWS Drift Detection is a pull-based operation, meaning it requires an explicit request to check for changes. AWS does not trigger drift detection automatically because it is a resource-intensive operation.

- AWS designed drift detection to be an API-driven process.
- It does not generate real-time drift events when an unauthorized change is made.
- Running drift detection continuously in real-time would exceed API rate limits and cause performance issues.

Example: If we tried running detect_stack_drift in real-time for 400+ stacks, AWS would throttle our requests due to API rate limits.

Link: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cloudformation-limits.html

3) Running Drift Detection on 400+ Stacks Will Exceed AWS Limits:

AWS enforces API rate limits on CloudFormation operations. Running drift detection on 400+ stacks at once will likely exceed these limits and cause throttling issues.

- AWS limits detect_stack_drift API calls to one request per stack at a time.
- If we attempt to scan 400+ stacks in parallel, AWS will throttle the API requests.
- This can delay detection and lead to incomplete results.

Link: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cloudformation-limits.html

4) Large-Scale Drift Detection Is Costly & Resource-Intensive:

Drift detection is a resource-heavy operation. Scanning 400+ stacks frequently would increase AWS costs and may affect overall account performance.

Drift detection is not lightweight— it scans every resource in a stack. Running it across hundreds of stacks frequently could increase AWS costs. Other AWS workloads may be affected due to high resource utilization.

5) Dealing with CloudTrail Event Retention: as of now if someone updated pipeline, we can see the user in output for like 5 to 6 minutes, if we re-test after like 10 minutes then we get unknown user in output.

The lookup-events API retrieves recent CloudTrail events, but AWS does not guarantee how long they stay immediately queryable.

- lookup-events only searches the last few minutes to hours of logs in memory.
- If an event is not available in memory, lookup-events won't return it, even though it exists in CloudTrail.
- After a few minutes, AWS moves logs to S3 or other storage (if enabled).

5.1) Why were events found for few minutes?

- CloudTrail keeps recent events in memory for quick lookup.
- After ~6 minutes, AWS likely moved the logs to long-term storage, making them unavailable via lookup-events.

5.2) Why does AWS say they keep CloudTrail events or 90 days?

AWS stores logs for 90 days in Event History, but after a short time (few minutes), they are not searchable via lookup-events.

The logs still exist in S3 (if CloudTrail logging is enabled), but lookup-events no longer finds them. (which is why we get unknown user after we retest the drifted stack after like 10 minutes or later or so.

Closest solution for user or log retention issue:

1) We can store detected drift events and responsible users in a DynamoDB table: ensuring long-term tracking even if CloudTrail logs expire through lookup-events. Every time drift is detected, store the drift timestamp, stack name, and responsible user in DynamoDB. This way, we always know who caused drift, even after CloudTrail deletes logs or the retention time is over.

2) we can query logs from S3 using AWS Athena instead:

Instead of storing in dynamodb we can query the logs from s3

link: https://docs.aws.amazon.com/athena/latest/ug/cloudtrail-logs.html

3) Alternative Fix of Fix (1): Store Detected Drift Data in DynamoDB:

Since CloudTrail logs disappear from lookup-events after a few minutes, another approach is to store drift events immediately when detected. To avoid losing modification data, we can store detected drift events in DynamoDB at the time of detection.

- Every time drift is detected, immediately store the user and event details in DynamoDB.
- The next time the script runs, check DynamoDB first before querying CloudTrail.
- Prevents losing drift details after CloudTrail events disappear from lookup-events.