ArgoCD Configs - RETRY Options in ArgoCD

RETRY Options in ArgoCD

The "Retry" option in ArgoCD controls whether failed sync operations are automatically retried. Let's break this down and also address your questions about the behavior when retries are turned off and the default sync interval.

Retry Option

- **OFF**: No automatic retries. If a sync operation fails, ArgoCD will not retry it automatically.
- **Behavior**: When a sync operation fails, ArgoCD will not attempt to retry the operation until the next scheduled sync or a manual sync is triggered.
- **ON**: Automatically retry on failure. If a sync operation fails, ArgoCD will attempt to retry the operation until it succeeds or reaches a retry limit.
- **Behavior**: When a sync operation fails, ArgoCD will automatically retry the operation based on configured retry settings (like retry count and interval).

Sync Intervals and Retries

ArgoCD can operate in two primary modes regarding syncing: automated sync with intervals or manual syncs. Here's how it works:

Automated Sync

- **Sync Interval**: If you have automated sync enabled, ArgoCD will check for changes in your Git repository at a regular interval.
- **Default Sync Interval**: The default sync interval in ArgoCD is 3 minutes. This means ArgoCD will check the repository for changes every 3 minutes and attempt to sync if there are any differences between the desired state in Git and the actual state in the cluster.
- **Behavior with Retry OFF**:
- If a sync operation fails, ArgoCD will not retry the failed operation automatically.
- The next sync attempt will happen at the next sync interval (every 3 minutes by default) unless you trigger a manual sync.

Manual Sync

- **Manual Trigger**: If you are using manual sync mode or if you want to re-sync before the next interval, you need to manually trigger a sync operation.
- **Behavior with Retry OFF**: After a failed sync, you would need to manually trigger another sync operation or wait for the next automated sync interval.

Example Scenario

Let's consider an example to make this clear:

1. **Automated Sync with Retry OFF**:
2. **Time 0**: A sync operation starts and fails due to some error.
3. **Time 0 to 3 Minutes**: No retry happens automatically because the retry option is OFF.
4. **Time 3 Minutes**: The next automated sync interval occurs, and ArgoCD attempts to sync again.
5. **Automated Sync with Retry ON**:
6. **Time 0**: A sync operation starts and fails due to some error.
7. **Time 0 to (retry interval)**: ArgoCD automatically retries the sync operation according to the configured retry settings until it succeeds or reaches the retry limit.

Configuring Retry Behavior

When retries are turned ON, you can configure how ArgoCD handles retries:

1. yamlCopy codeapiVersion: argoproj.io/v1alpha1
2. kind: Application
3. metadata:
4. name: my-application
5. spec:
6. project: default

```
  source:
    repoURL: https://github.com/my-repo.git
    path: manifests
  destination:
    server: https://kubernetes.default.svc
    namespace: default
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
    - Retry=true
    retry:
      limit: 5 # Maximum number of retries
      backoff:
        duration: 5s # Initial retry interval
        factor: 2 # Multiplication factor for the backoff
        maxDuration: 1m # Maximum retry interval
```

where,

1. **limit**: Maximum number of retries. In this case, it is set to 5.
2. **backoff**:
3. **duration**: Initial retry interval. In this case, it is set to 5 seconds.
4. **factor**: Multiplication factor for the backoff. In this case, it is set to 2.
5. **maxDuration**: Maximum retry interval. In this case, it is set to 1 minute (60 seconds).

Diving into the factor Parameter

The factor determines how the retry interval increases exponentially after each failed retry attempt. The backoff duration is multiplied by this factor for each subsequent retry until the maximum duration is reached.

Example Scenario

Let's go through a detailed example using the provided configuration:

1. **Initial Retry**:
2. **Initial Interval**: 5 seconds
3. **First Retry (Attempt 1)**:
4. Wait 5 seconds before retrying.
5. **Next Interval**: 5s * 2 = 10 seconds
6. **Second Retry (Attempt 2)**:
7. Wait 10 seconds before retrying.
8. **Next Interval**: 10s * 2 = 20 seconds
9. **Third Retry (Attempt 3)**:
10. Wait 20 seconds before retrying.
11. **Next Interval**: 20s * 2 = 40 seconds
12. **Fourth Retry (Attempt 4)**:
13. Wait 40 seconds before retrying.
14. **Next Interval**: 40s * 2 = 80 seconds (but limited to maxDuration)
15. **Fifth Retry (Attempt 5)**:
16. Wait the maximum duration, which is 60 seconds before retrying (since 80 seconds exceeds the maxDuration).

Summary of Retry Intervals

Here’s a summary of the retry intervals for each attempt based on the given configuration:

1. **Retry 1**: 5 seconds

2. **Retry 2**: 10 seconds
3. **Retry 3**: 20 seconds
4. **Retry 4**: 40 seconds
5. **Retry 5**: 60 seconds (maxDuration)

Here's a simple ASCII diagram to visualize the retry intervals:

1. Retry Attempts:
2. |
3. |-- Attempt 1: Wait 5 seconds
4. |
5. |-- Attempt 2: Wait 10 seconds
6. |
7. |-- Attempt 3: Wait 20 seconds
8. |
9. |-- Attempt 4: Wait 40 seconds
10. |
11. |-- Attempt 5: Wait 60 seconds (capped by maxDuration)

The factor parameter in the retry configuration allows for exponential backoff of retry intervals. Each subsequent retry interval is calculated by multiplying the previous interval by the factor, up to the maxDuration. This approach helps in gradually increasing the wait time between retries, reducing the load on the system and allowing transient issues to resolve themselves.

Conclusion

- **With Retry OFF**: ArgoCD will not retry failed sync operations automatically. It will wait until the next sync interval (default 3 minutes) or a manual sync is triggered.
- **With Retry ON**: ArgoCD will automatically retry failed sync operations based on the configured retry settings.

By understanding these options, you can configure ArgoCD to handle sync failures in a way that suits your operational needs, whether prioritizing quick resolution of failures with retries or relying on periodic sync intervals.