Member-only story

ArgoCD 3.2: The Latest Stable Release Is Here

Rui Coelho Follow 8 min read · 4 days ago

A deep dive into the newest features, improvements, and what you need to know before upgrading

The GitOps community just received a significant update. ArgoCD 3.2.0 was released as a stable version on November 5th, 2025. If you're running ArgoCD in production, this release deserves your immediate attention — especially if you're still on version 2.14, which officially reached End of Life on November 4th, 2025.

In this article, we'll explore what's new, what's changing, and how to prepare your team for the upgrade.

The Context: Why ArgoCD 3.2 Matters

Before diving into the features, let's understand where ArgoCD 3.2 fits in the bigger picture.

The Evolution of ArgoCD 3.x

ArgoCD 3.0, released in early 2025, was a foundational release that introduced significant architectural improvements without being a risky upgrade. It refined RBAC controls, improved resource exclusions, and updated secrets management guidance.

Version 3.1, launched in August 2025, brought game-changing features like native OCI registry support, CLI plugins, and enhanced Source Hydrator functionality. These additions positioned ArgoCD as a more versatile GitOps tool for enterprise adoption.

The Critical Timeline

Here's what you need to know: **ArgoCD 2.14 reached End of Life (EOL) on November 4th, 2025.** According to ArgoCD's support policy, only the three most recent minor versions receive security updates and bug fixes. This means:

- Currently supported: 3.2, 3.1, and 3.0
- No longer supported: 2.14 and earlier versions
- No more security patches or bug fixes for 2.14

If you're still on 2.14 or earlier, you're running an unsupported version. Your upgrade should be a top priority.

What's New in ArgoCD 3.2

Let's break down the key features and improvements coming in this release.

1. Enhanced ApplicationSet Progressive Sync

Progressive Sync for ApplicationSets has received significant improvements in 3.2. This feature allows you to roll out changes gradually across multiple applications, which is crucial for risk management in production environments.

What's improved:

- **Better UI visibility:** The ApplicationSet UI now properly displays Progressive Sync status, resolving the "Unknown" state issue that plagued previous versions
- **Status cleanup:** When Progressive Sync is disabled, the ApplicationSet now correctly clears the `applicationStatus` field, preventing stale data
- **Resource count tracking:** A new `status.resourcesCount` field provides visibility into the number of resources managed by each ApplicationSet

This last point is particularly important. Large ApplicationSets could previously cause performance issues due to unbounded resource tracking. The new resource count limit helps prevent this:

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: my-appset
spec:
  # ... your spec
status:
  resourcesCount: 150  # New field
  resources:
    - # Limited list of resources
```

2. Memory Optimization for Webhook Handlers

If you're operating large monorepos or high-traffic environments, you'll appreciate this: ArgoCD 3.2 optimizes webhook handlers to use informers instead of direct API calls. This change significantly reduces memory consumption during webhook processing.

Why it matters: In environments with frequent Git commits or large repositories, webhook processing could cause memory spikes. The new implementation is more efficient and stable.

Important note: Users with very large monorepos may still encounter repo-server lock contention requiring pod restarts. The ArgoCD team has acknowledged this issue, and a fix is planned for the next patch release (3.2.1).

3. Updated Health Checks

Health assessment is a critical part of ArgoCD's functionality. Version 3.2 includes several health check updates:

- **Crossplane V2 support:** Health checks now support Crossplane V2 resources, reflecting the evolution of the Crossplane ecosystem
- **External Secrets Operator:** The ExternalSecret discovery script now includes the `refreshPolicy` field for more accurate health assessment
- **PromotionStrategy corrections:** Fixed typos in the Promotion health checks that could cause false negatives

4. OCI Registry Improvements

Building on the OCI support introduced in 3.1, version 3.2 loosens layer restrictions, making it easier to use OCI registries for storing Kubernetes configurations. This is part of ArgoCD's broader strategy to treat configuration artifacts with the same maturity as container images.

5. CLI and Notifications Fixes

Several quality-of-life improvements for CLI users:

- The notifications CLI now properly initializes the `argocdService`, fixing initialization issues
- Webhook payload handlers now gracefully recover from panics instead of crashing
- Various documentation improvements and bug fixes

Breaking Changes and Migration Considerations

ArgoCD 3.2 maintains the low-risk upgrade philosophy of the 3.x series. However, there are some considerations:

Coming from ArgoCD 2.14

If you're upgrading from 2.14, you'll need to account for all the breaking changes introduced in 3.0 and 3.1:

Major changes from 3.0:

- Fine-grained RBAC no longer applies to sub-resources by default

- Health status persistence changes (now disabled by default)
- Default resource exclusions for high-churn resources
- Dex authentication claim changes (uses `federated_claims.user_id` instead of `sub`)

Major changes from 3.1:

- OCI registry support enabled
- CLI plugins architecture
- Source Hydrator enhancements

Recommended Upgrade Path

1. **Read the docs first:** Review the official upgrade guide for your version
2. **Test in non-production:** Deploy the RC in a staging environment
3. **Backup your state:** Ensure you have backups of your ApplicationSet and Application resources
4. **Plan for RBAC:** If coming from 2.x, audit your RBAC policies
5. **Monitor after upgrade:** Watch for memory usage patterns and webhook processing

Real-World Impact: Who Benefits Most?

Platform Engineering Teams

If you're building internal developer platforms, the ApplicationSet improvements will help you manage hundreds of applications more efficiently. The resource count tracking and memory optimizations mean you can scale further without hitting performance walls.

Large Monorepo Users

The webhook handler optimizations directly address pain points for teams managing large repositories. Less memory pressure means more stable ArgoCD instances during high-commit periods.

Multi-Tenant Environments

The continued refinement of RBAC and the stability improvements make ArgoCD 3.2 more suitable for multi-tenant setups where different teams share the same ArgoCD instance but need strict isolation.

Crossplane Users

If you're adopting Crossplane for infrastructure management alongside ArgoCD for application deployment, the updated health checks for Crossplane V2 ensure better visibility into your control plane resources.

Release Timeline and What's Next

ArgoCD 3.2.0 was released as a stable version on November 5th, 2025. Here's what to expect moving forward:

- **Current stable version**: 3.2.0
- **Supported versions**: 3.2, 3.1, and 3.0

- **Next release**: 3.3 is expected in approximately 3 months (following the quarterly release cadence)
- **Patch releases**: Bug fixes and security updates will be released as 3.2.x versions as needed

The first patch release (3.2.1) is expected soon to address the large monorepo lock contention issue.

Hands-On: Installing ArgoCD 3.2

Ready to try the latest stable release? Here's how to deploy ArgoCD 3.2.0 in your cluster:

Installation via Helm (Recommended)

Helm is the recommended way to install ArgoCD in production environments as it provides better configuration management and easier upgrades.

Add the ArgoCD Helm repository:

```
helm repo add argo https://argoproj.github.io/argo-helm
helm repo update
```

Install ArgoCD 3.2.0:

```
# Create namespace
kubectl create namespace argocd
```

```
# Install with default configuration
helm install argocd argo/argo-cd \
  --namespace argocd \
  --version 9.1.0

# Or install with custom values
helm install argocd argo/argo-cd \
  --namespace argocd \
  --version 9.1.0 \
  --values values.yaml
```

Example values.yaml for production:

```
# Enable HA mode
redis-ha:
  enabled: true

controller:
  replicas: 2

server:
  replicas: 2
  ingress:
    enabled: true
    ingressClassName: nginx
    hosts:
      - argocd.yourdomain.com
    tls:
      - secretName: argocd-tls
        hosts:
          - argocd.yourdomain.com

repoServer:
  replicas: 2

applicationSet:
  replicas: 2

# Enable notifications
notifications:
  enabled: true
```

```
  # Configure resource limits
  configs:
    params:
      server.insecure: false
```

Upgrade from previous version:

```
helm upgrade argocd argo/argo-cd \
  --namespace argocd \
  --version 9.1.0 \
  --values values.yaml
```

Installation via kubectl (Quick Start)

For testing or non-production environments, you can use kubectl directly:

Non-HA Installation:

```
kubectl create namespace argocd
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v3
```

High Availability Installation:

```
kubectl create namespace argocd
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v3
```

Verifying the Installation

Check ArgoCD pods status:

```
kubectl get pods -n argocd

# Expected output (HA installation):
# NAME                                                READY   STATUS    RESTARTS
# argocd-application-controller-0                     1/1     Running   0
# argocd-applicationset-controller-xxx                1/1     Running   0
# argocd-dex-server-xxx                               1/1     Running   0
# argocd-notifications-controller-xxx                 1/1     Running   0
# argocd-redis-ha-haproxy-xxx                         1/1     Running   0
# argocd-redis-ha-server-0                            2/2     Running   0
# argocd-repo-server-xxx                              1/1     Running   0
# argocd-server-xxx                                   1/1     Running   0
```

Verify the ArgoCD version:

bash

```
kubectl get pods -n argocd -l app.kubernetes.io/name=argocd-server -o jsonpath='

# Should return: quay.io/argoproj/argocd:v3.2.0
```

Access the ArgoCD UI:

```
# Port-forward (for quick access)
kubectl port-forward svc/argocd-server -n argocd 8080:443
```

```
# Then access: https://localhost:8080
```

Get the initial admin password:

bash

```
# For Helm installations
kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.pas

# For kubectl installations (same command)
kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.pas
```

Login via CLI:

```
# Install ArgoCD CLI
brew install argocd  # macOS

# or
curl -sSL -o argocd https://github.com/argoproj/argo-cd/releases/download/v3.2.0
chmod +x argocd
sudo mv argocd /usr/local/bin/

# Login
argocd login localhost:8080 --username admin --password <password> --insecure
# Verify version
argocd version
```

What to Test

Focus your testing on:

1. **ApplicationSet Progressive Sync**: Create an ApplicationSet with progressive sync enabled and verify the UI shows proper status
2. **Memory usage**: Monitor memory consumption during webhook processing
3. **Health checks**: If you use Crossplane, verify V2 resources show correct health status
4. **RBAC**: Validate your existing policies work as expected

Looking Ahead: The Future of ArgoCD

ArgoCD 3.2 continues the project's trajectory toward becoming the definitive GitOps tool for Kubernetes. Some trends worth watching:

OCI-Native Configuration

The push toward OCI registries suggests a future where Kubernetes configurations are treated as first-class artifacts, with the same supply chain security guarantees as container images.

Progressive Delivery Integration

The improvements to Progressive Sync hint at deeper integration with progressive delivery patterns. We may see more sophisticated rollout strategies in future versions.

Platform Engineering Enablement

With better scalability and multi-tenancy support, ArgoCD is positioning itself as a critical component of internal developer platforms, not just a deployment tool.

Action Items: Upgrading to ArgoCD 3.2

Here's your checklist:

Immediate (if on 2.14 or earlier — you're on EOL!):

- **Upgrade immediately** — you're no longer receiving security patches
- Review the 3.0, 3.1, and 3.2 release notes
- Audit your RBAC policies for 3.0 compatibility
- Set up a test environment with ArgoCD 3.2
- Test your critical ApplicationSets and Applications

Before production upgrade:

- Document your current ArgoCD configuration
- Backup your ArgoCD state
- Create a rollback plan
- Schedule a maintenance window for production upgrade
- Review the known issues (large monorepo lock contention)

Post-upgrade:

- Monitor memory usage patterns
- Validate all Applications sync correctly
- Check webhook processing performance
- Review ApplicationSet statuses
- Watch for 3.2.1 patch release if you have large monorepos

Conclusion

ArgoCD 3.2 is now stable and production-ready. This release represents a measured evolution of the platform — not revolutionary, but a refinement that makes ArgoCD more stable, performant, and user-friendly. The ApplicationSet improvements, memory optimizations, and updated health checks address real pain points that operators face in production.

If you're on ArgoCD 2.14, you're running an unsupported version. Upgrade immediately. If you're already on 3.0 or 3.1, upgrading to 3.2 should be low-risk, but still test thoroughly, especially if you operate large monorepos.

The GitOps ecosystem continues to mature, and ArgoCD remains at the forefront. Version 3.2 is another solid step in that journey, and with the EOL of 2.14, there's no better time to upgrade than now.

Additional Resources

- ArgoCD Official Documentation

- ArgoCD GitHub Releases
- ArgoCD 3.2.0 Release Notes
- Upgrading to ArgoCD 3.x Guide
- ArgoCD 2.14 to 3.0 Migration Guide

Have you upgraded to ArgoCD 3.2 yet? What's your experience been like? Share your thoughts in the comments below!

Kubernetes Argocd Gitops Sre DevOps

Written by Rui Coelho

23 followers · 3 following

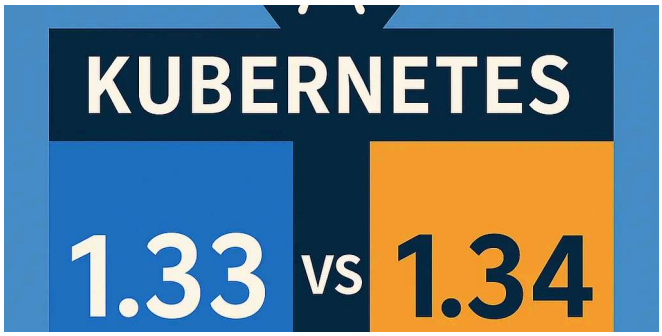
Follow

No responses yet

David B Chase

What are your thoughts?

More from Rui Coelho

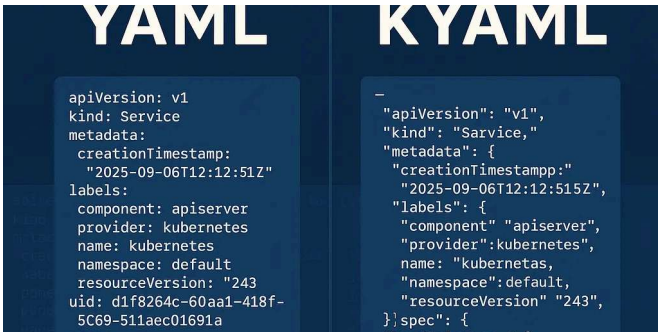


Rui Coelho

Kubernetes 1.33 vs 1.34: What's New, What Changed, and Why It...

Introduction

Sep 15 4

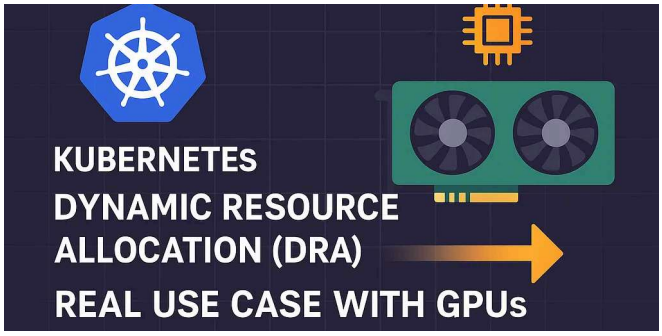


Rui Coelho

KYAML in Kubernetes v1.34: A Safer, Leaner Alternative to YAML...

Introduction

Sep 6 1



Rui Coelho

How to Use Kubernetes Dynamic Resource Allocation (DRA)—Real...

Introduction

Rui Coelho

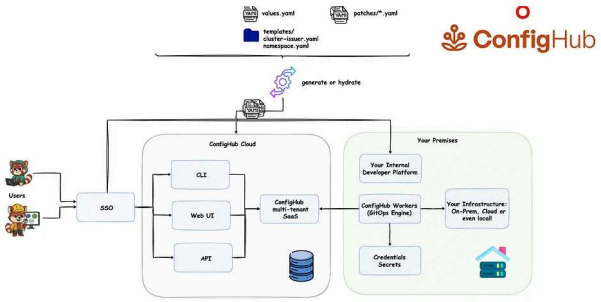
Control Xiaomi Yeelight Bulbs with Python

I'm an enthusiast user of Smart Home stuff, a year ago I bought a Xiaomi Yeelight Bulb and...

Sep 17 1 Mar 23, 2021 2

See all from Rui Coelho

Recommended from Medium

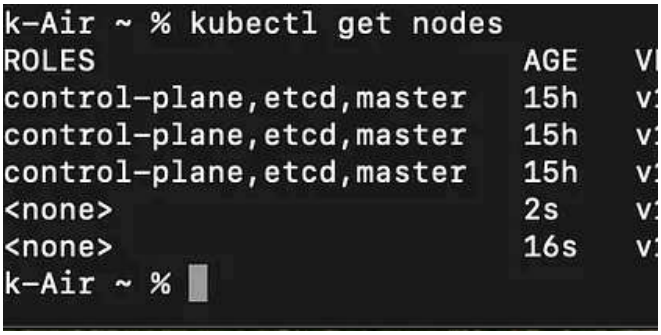


In ITNEXT by Artem Lajko

ConfigHub: Why Your Internal Developer Platform Needs It

See why GitOps often feels like a sprawl of configs, discover how to manage...

5d ago 64 1

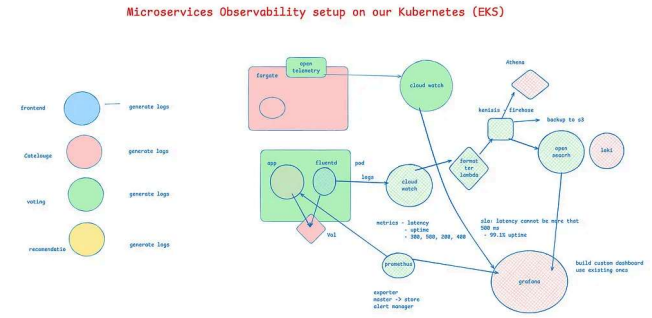


Mark Southworth

Build Your Own High-Availability Kubernetes Home Lab for Less...

A 2025 Guide to Budget Kubernetes Home Labs.

Oct 16 14

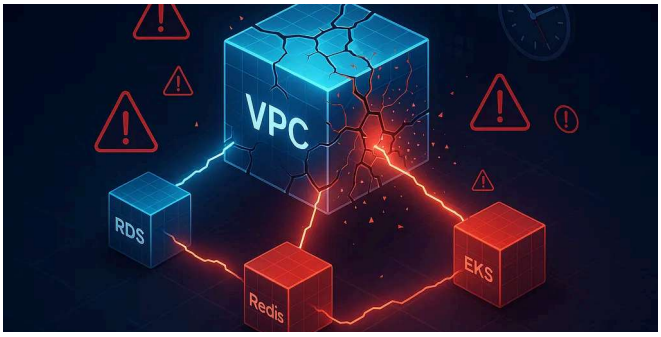


In Towards AWS by Akhilesh Mishra

You're Not Ready for Kubernetes Observability Until You Understan...

The Real Story of Kubernetes Observability (That Actually Works in Production)

3d ago 101 1



Heinancabouly

How Crossplane's Auto-Upgrades Took Down Our Hospital...

When a broken patch release, dangerous defaults, and auto-upgrades collided with...

5d ago 6 1

In loveholidays tech by Alina Frolova

One Tool to Rule Them All: Migrating to Grafana Alloy

It's hard to overestimate the importance of observability for modern technology system...

5d ago 502

Bhavyansh

The CI/CD Setup That Cut Our Deployment Time from 40 Minute...

How we optimized our deployment pipeline and got back 32 minutes of our lives (per...

Oct 27 65 3

See more recommendations