Kubernetes Backup & Disaster

The Essential 20%

Why Kubernetes Backup Matters

In Kubernetes environments, data lives across distributed components and persistent volumes.

Without proper backup strategies:

- Data loss in container environments is a significant concern, with many organizations reporting incidents
- Kubernetes itself doesn't provide comprehensive built-in backup capabilities
- Configuration errors, failed updates, and infrastructure failures can cause significant downtime

What Needs Protection in Kubernetes

The critical resources requiring backup:

- Cluster State: etcd database (core configuration)
- API Objects: Deployments, StatefulSets, ConfigMaps, Secrets
- Persistent Volumes: Application data stored outside ephemeral containers
- Custom Resources: CRDs and associated configurations
- RBAC Settings: Role-based access control configurations

The "Stateful" Challenge

Kubernetes was designed for stateless applications, making backup complex:

- Persistent data resides in volumes provisioned through StorageClasses
- Pods are ephemeral but the data must persist
- Application-consistent backups require coordination
- Different storage providers have varying snapshot capabilities
- Database workloads need special consideration for consistency

The 3-2-1 Backup Rule for Kubernetes

Industry-standard backup strategy applied to Kubernetes:

- 3 copies of your data (production + 2 backups)
- Stored on 2 different types of media
- 1 copy stored offsite/different failure domain
- Crucial for protecting against infrastructure-wide failures
- Mitigates risks from regional cloud provider outages when implemented correctly

Recovery Point Objective & Recovery Time Objective

Business-critical metrics for your backup strategy:

- **RPO**: Maximum acceptable data loss measured in time
- **RTO**: Maximum acceptable downtime until service restoration
- Kubernetes environments aim for low RPOs, but achieving minutes may depend on tools and complexity
- Automation is key to achieving low RTOs in complex environments
- Higher availability requirements = higher costs

Backup Approaches

Three primary approaches to Kubernetes backup:

- **Native Snapshots**: Using CSI volume snapshots for PVs
- **Application-Aware**: Leveraging application hooks for consistency
- **Cluster-Wide Tools**: Purpose-built K8s backup solutions

Each offers different levels of consistency, complexity, and recovery capabilities.

The Etcd Factor

The etcd database is Kubernetes' brain:

- Stores all cluster configuration and state
- Regular etcd snapshots provide disaster recovery capabilities
- Corruption or loss means complete cluster failure
- **Recommendation**: Automate regular etcd backups (at least daily)
- Consider dedicated backup tools that handle etcd correctly

Disaster Recovery Strategies

Key DR approaches for Kubernetes:

- **Backup & Restore**: Rebuild from backups (highest RPO/RTO)
- **Pilot Light**: Minimal standby environment with core components
- **Warm Standby**: Fully functional replica with reduced resources
- **Hot Standby**: Full replica ready for immediate failover
- **Multi-Cluster**: Active-active across regions/providers

Velero - A Popular Option

Velero (formerly Heptio Ark) is a widely used K8s backup solution:

- Open-source, community-supported
- Backs up cluster resources and persistent volumes
- Supports major cloud providers
- Handles scheduled backups
- Provides namespace filtering and hook capabilities
- Not the only option, other tools like Kasten K10 and TrilioVault are also prominent alternatives

Testing is Non-Negotiable

Backup validation is essential:

- Many recoveries fail due to untested backups, underscoring the importance of validation
- Schedule regular disaster recovery drills
- Automate backup verification when possible
- Test full cluster restoration to validate processes
- Document recovery procedures for team accessibility

Namespace Granularity

Effective backup planning requires namespace organization:

- Group related resources in namespaces for easier backup/restore
- Consider backup schedules based on namespace criticality
- Enables partial restores without impacting the entire cluster
- Aligns with multi-team environments and responsibilities
- Simplifies testing of restore procedures

Beyond Backup: GitOps Approach

Complement traditional backups with GitOps practices:

- Store all declarative configurations in version control
- Enable infrastructure-as-code recovery approaches alongside traditional backups for runtime data recovery
- Maintain deployment pipelines for rapid reconstruction
- Reduces reliance on full cluster state backups for configuration recovery

Building enterprise-grade cloud solutions?

Follow for architecture & implementation insights.