Gmail

Sandhya Babu <sandhya.bca7@gmail.com>

Day21 Challenge: K8s Upgrade Mastery: From Minikube to EKS

Sagar Utekar <getfitwithsagar2366@gmail.com> Mon, Dec 23, 2024 at 8:00 AM
Bcc: sandhya.bca7@gmail.com

Hello Learners,

Welcome back to another thrilling episode of the **DevOps SRE Daily Challenge!** 🎉

In the world of Kubernetes, keeping your clusters up-to-date isn't just a best practice—it's essential for ensuring security, stability, and access to the latest features. For DevOps engineers, SREs, and Kubernetes certification candidates, **mastering cluster upgrades** across diverse environments is a critical skill.

Whether you're managing clusters in local development environments, production-grade cloud infrastructures, or anything in between, understanding the nuances of different upgrade strategies will make you a Kubernetes pro!

Why Cluster Upgrades Matter for Your Exam

Real-World Relevance:

Upgrading clusters is a routine task in production environments to patch vulnerabilities, improve performance, and enable new features. Understanding these processes prepares you for managing real-world Kubernetes workloads.

Certification Preparation:

Tasks related to version upgrades are commonly seen in certification exams like CKA. Being familiar with these workflows will save you valuable time during the test.

Managed vs. Self-Hosted Clusters:

Understanding the differences between upgrading managed Kubernetes services like EKS and self-hosted clusters is essential. Managed services often abstract some complexities but still require careful planning to ensure compatibility and performance. Self-hosted clusters offer more control but demand a deeper understanding of the upgrade process.

Best Practices:

- Always test upgrades in staging or non-production environments first.
- Backup your cluster and critical data before initiating any upgrade.
- Read the release notes carefully for deprecations and breaking changes.
- Automate where possible to ensure consistency and reduce human errors.

Challenge Tasks

Theory Challenge: Understanding Cluster Upgrades

1. ### Why Upgrade Clusters?
   - Explain the importance of upgrading Kubernetes clusters in real-world scenarios.
   - Discuss the risks of outdated cluster versions.
2. ### Upgrade Strategies:
   - What are the typical upgrade approaches for Kubernetes clusters (e.g., rolling upgrades, blue-green deployments)?
   - When should you choose one strategy over another?
3. ### Managed vs. Self-Hosted Clusters:
   - Highlight the key differences in upgrade processes for managed services like EKS and self-hosted clusters using tools like Kubeadm.
   - Discuss the unique challenges and best practices for each.

Practical Challenge: Upgrading Clusters in Different Environments

1. Minikube Cluster Upgrade

- Upgrade your existing Minikube setup to the latest version.
- Verify that the Kubernetes version in the cluster matches the latest stable release.

2. Kind Cluster Upgrade

- Create a Kind cluster using an older Kubernetes version (e.g., v1.30.8).
- Upgrade kind cluster to latest version
- Confirm the version upgrade with `kubectl version`.

3. Kubeadm Cluster Upgrade

- Create a multi-node Kubernetes cluster using Kubeadm.
- Upgrade the Control Plane and Nodes to a newer Kubernetes version.
- Verify that all components (kube-apiserver, kube-controller-manager, kube-scheduler, etc.) are updated.

Resources:

- Kubeadm Upgrade Documentation
- Kubeadm Cluster Upgrade Guide
- YouTube Video on Kubeadm Upgrades

4. EKS Cluster Upgrade

- Upgrade the EKS Control Plane to a newer version using AWS CLI.
- Update the node groups to match the Control Plane version.
- Confirm the upgrade with `kubectl version`.

Note: EKS also provides an **auto mode** for node groups, which simplifies the process of keeping your nodes up-to-date. While this reduces manual intervention, it's important to monitor the process and validate workloads after the upgrade to ensure seamless transitions.

Best Practices:

- Monitor your workloads during the upgrade to ensure there are no disruptions.
- Use a phased approach to minimize risks in production.
- Leverage EKS version release notes and upgrade tools to plan your upgrades effectively.

Submission Guidelines

1. Your answers to the theory questions.
2. Screenshots of:
    - The pre-upgrade and post-upgrade versions for each cluster.
    - Key commands executed during the upgrade process.
3. Document your observations and challenges during the upgrades.
4. Post your progress with the hashtags: #KubernetesUpgrades, #DevOpsForAll, #ckawithsagar

Resources to Help You

- **Minikube Upgrade Documentation**
- **Kind Cluster Guide**

- **Kubeadm Upgrade Guide**
- **YouTube: Kubeadm Upgrades**
- **EKS Cluster Upgrade Documentation**

If you missed any previous challenges, you can catch up by reviewing the problem statements on GitHub.

Best regards,
Sagar Utekar