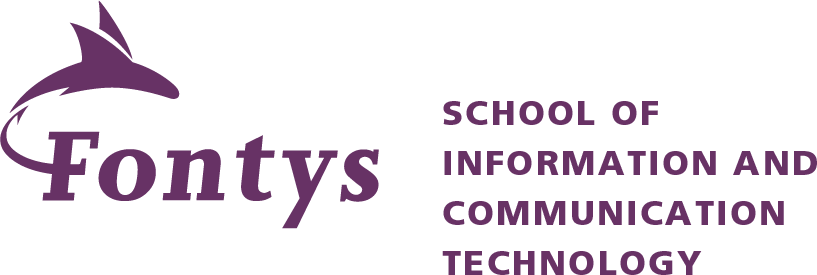
|  |  |
| --- | --- |
| Name : | Pierre-Antoine Ridderstap  Vadym Tkachenko  Yanina Petrova  Gabriel Rafael |
| Date : | 12/06/2023 |



Kubernetes Cluster Upgrade Without Disruption

Analysis Document

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Revision Table | Description | Date | Author | Checked by | Approved by |
| V0.1 | First version | 06-06-2023 | Students | Haverkort, Frank | Haverkort, Frank |

Table of Contents

[Introduction 3](#_Toc136865683)

[Upgrade strategy 4](#_Toc136865684)

[Comparison 4](#_Toc136865685)

[Availability 4](#_Toc136865686)

[Rollback 4](#_Toc136865687)

[Resource Efficiency 4](#_Toc136865688)

[Complexity 5](#_Toc136865689)

[Scalability 5](#_Toc136865690)

[Upgrade Duration 5](#_Toc136865691)

[Score 6](#_Toc136865692)

[Conclusion 6](#_Toc136865693)

[Tools 7](#_Toc136865694)

[Comparison 7](#_Toc136865695)

[Support 7](#_Toc136865696)

[Documentation 7](#_Toc136865697)

[Flexibility 7](#_Toc136865698)

[Cloud Provider Support 8](#_Toc136865699)

[Complexity 8](#_Toc136865700)

[Score 9](#_Toc136865701)

[Conclusion 9](#_Toc136865702)

[Overall Conclusion 10](#_Toc136865703)

# Introduction

In this document, we will be discussing the upgrade strategy and tools for our Kubernetes cluster version upgrade project. We have identified three upgrade strategies: Rolling Upgrade, Blue-Green, and Canary in our *Research Document*. Our goal is to select the best strategy based on a comparison of their key aspects such as availability, rollback capability, resource efficiency, complexity, scalability, and upgrade duration. By evaluating these categories, we can determine the strategy that aligns most effectively with our project requirements.

Additionally, we have identified three tools: Kubeadm, Kubespray, and Kops, which are suitable for managing Kubernetes clusters. We will compare these tools based on their support, documentation, flexibility, cloud provider support, and complexity. By assessing these factors, we can select the most appropriate tool that meets our project requirements.

Through the comparing and scoring of the strategies and tools, we aim to provide a clear recommendation for the upgrade strategy and the preferred tool for upgrading our Kubernetes cluster version.

# Upgrade strategy

In the research document we talked about 3 strategies that we found best suit our project. To select one strategy, we need to compare these strategies to the requirements we have for the project. For this we will be comparing them on the following subjects:

* Availability
* Rollback
* Resource Efficiency
* Complexity
* Scalability
* Upgrade Duration

## Comparison

### Availability

**Rolling Upgrade:**

The rolling upgrade strategy ensures that the Kubernetes cluster remains available during the upgrade process. It updates one node at a time while the rest of the cluster continues to function normally.

**Blue-green:**

The blue-green strategy involves deploying a new cluster alongside the existing one, allowing for a seamless switch between the two. This ensures high availability during the upgrade.

**Canary:**

The canary strategy gradually directs a small portion of the traffic to the new cluster version while keeping most of the workload on the existing cluster, maintaining availability throughout the upgrade process.

### Rollback

**Rolling Upgrade:**

Rolling upgrade strategy allows easy rollback by simply reverting the upgrade on individual nodes.

**Blue-green:**

Blue-green strategy provides an easy rollback by redirecting traffic back to the original cluster.

**Canary:**

Canary strategy allows for a easy rollback by reducing the traffic to the new version and directing it back to the original cluster.

### Resource Efficiency

**Rolling Upgrade:**

Rolling upgrade strategy is resource efficient as it updates nodes one by one, requiring minimal additional resources during the upgrade process.

**Blue-green:**

Blue-green strategy may require additional resources as it involves running two clusters simultaneously until the switch is made.

**Canary:**

Canary strategy works the same as Blue-green as it involves running two clusters simultaneously until the switch is made.

### Complexity

**Rolling Upgrade:**

Rolling upgrade strategy is relatively simple and has a lower complexity compared to other strategies.

**Blue-green:**

Blue-green strategy can be more complex as it involves managing two separate clusters and ensuring smooth traffic redirection.

**Canary:**

Canary strategy can be more complex as it requires traffic management during the upgrade process.

### Scalability

**Rolling Upgrade:**

Rolling upgrade strategy can handle both small and large environments, as it upgrades one node at a time.

**Blue-green:**

Blue-green strategy is less scalable as it involves running two separate clusters, which requires managing two identical size environments (in case of a large environment it may become complex)

**Canary:**

Canary strategy works the same as Blue-green as it involves running two separate clusters, which requires managing two identical size environments (in case of a large environment it may become complex)

### Upgrade Duration

**Rolling Upgrade:**

The upgrade duration for rolling upgrade strategy depends on the number of nodes in the cluster but can be longer compared to other strategies as each node is upgraded individually.

**Blue-green:**

Blue-green strategy can have a shorter upgrade duration as the switch between clusters can be made quickly.

**Canary:**

Canary strategy allows for a controlled and gradual upgrade, which may extend the overall upgrade duration compared to Blue-green, but is shorter compared to Rolling Upgrade.

## Score

Based on the requirement comparison, we can give each strategy a score out of 5. This will help us find the best strategy out of the three. The following is a table with the overall score for each strategy.

|  |  |  |  |
| --- | --- | --- | --- |
| Requirements | Rolling Upgrade | Blue-Green | Canary |
| Availability | 5 | 5 | 5 |
| Rollback | 4 | 5 | 5 |
| Resource Efficiency | 5 | 3 | 3 |
| Complexity | 4 | 3 | 3 |
| Scalability | 5 | 3 | 3 |
| Upgrade Duration | 3 | 5 | 5 |
| Total: | 26 | 24 | 24 |

## Conclusion

Based on the comparison of the three strategies, Rolling Upgrade outperforms Blue-Green and Canary in several key areas. The key differences where:

* Resource Efficiency:
  + Rolling Upgrade is the most resource efficient out of the three, this is because it doesn’t require any extra resources.
* Complexity:
  + Because Rolling Upgrade, upgrades the cluster a node at a time, makes it less complex compared to the other two strategies. As in the case of the other two strategies, the larger the environment the more complex it becomes to recreate.
* Scalability:
  + Because Rolling Upgrade, upgrades the cluster a node at a time, it doesn’t matter how large the cluster is. In the case of the other two strategies, the larger the environment, the more work needs to be done to create them.

Considering these factors, Rolling Upgrade stands out as our recommended strategy for upgrading Kubernetes cluster version in this project.

# Tools

In the research document we talked about 3 tools that we found best suit our project. To select one tool, we need to compare these tools to the requirements we have for the project. For this we will be comparing them on the following subjects:

* Support
* Documentation
* Flexibility
* Cloud provider support
* Complexity

## Comparison

### Support

**Kubeadm:**

Kubeadm has a good community support and is an official tool for Kubernetes cluster management. It has a large user base that can help ensure support and continuous improvement. Users can use the community forums, and online resources to seek support and share knowledge.

**Kubespray:**

Kubespray has a dedicated community and is in active development. However, compared to Kubeadm it is not an official Kubernetes cluster management tool. It is an open-source project that has dedicated users that contributes to the project's maintenance and improvements. Users can participate in discussions, report issues, and contribute to the project's further development.

**Kops:**

Same as Kubespray, Kops is an open-source project and has a user’s that participate in discussions, report issues, and contribute to the project's further development.

### Documentation

**Kubeadm:**

Kubeadm documentation is provided by its own community. The documentation includes installation guides, configuration examples, and troubleshooting tips, making it easier for users to manage their clusters. Additionally, Kubeadm documentation is frequently updated, ensuring accurate and up-to-date information for users.

**Kubespray:**

Same as Kubeadm, Kubespray documentation is provided by the community and includes installation guides, configuration examples, and troubleshooting tips. The documentation is frequently updated, ensuring accurate and up-to-date information for users.

**Kops:**

Same goes for Kops, the documentation is provided by the community and includes installation guides, configuration examples, and troubleshooting tips. The documentation is frequently updated, ensuring accurate and up-to-date information for users.

### Flexibility

**Kubeadm:**

Kubeadm provides a solid foundation for cluster management and upgrades. It focuses on simplicity and follows Kubernetes best practices. While it has some configuration options, it is designed to be straightforward and user-friendly.

**Kubespray:**

Kubespray is more high level, flexible, and has more customization options. It allows users to change cluster configuration parameters, including network settings, authentication mechanisms, and storage options.

**Kops:**

Kops is the same as Kubespray as it is more high level, flexible, and has more customization options. It allows users to change cluster configuration parameters, including network settings, authentication mechanisms, and storage options.

### Cloud Provider Support

**Kubeadm:**

Kubeadm only communicates with e kubernetes nodes and does not manage or require any resource from a cloud provider itself. Making it cloud agnostic.

**Kubespray:**

Kubespray does manage and require resources from cloud providers. However, Kubespray supports multiple cloud providers. This broad cloud provider support allows users to use their preferred cloud services and take advantage of specific features offered by different providers.

**Kops:**

Kops is primarily focuses on managing Kubernetes clusters on AWS. While Kops supports other cloud providers to some extent (such as GCP, and Azure), its capabilities and level of support are nowhere near sufficient.

### Complexity

**Kubeadm:**

Kubeadm is simple to use, it is automating it where it becomes complex.

**Kubespray:**

Kubespray is more complex compared to Kubeadm, as it has more advanced configuration options. It requires additional setup and configuration effort, including defining inventory files, specifying configuration parameters, and managing Ansible playbooks. However, this increased complexity provides users with a higher degree of control and flexibility over their cluster deployments.

**Kops:**

In terms of complexity, Kops is at the same level as Kubespray. Although Kubespray requires more additional steps when setting up, Kops has the same level of customization that makes it more complex.

## Score

Based on the requirement comparison, we can give each tool a score out of 5. This will help us find the best tool out of the three. The following is a table with the overall score for each tool.

|  |  |  |  |
| --- | --- | --- | --- |
| Requirements | Kubeadm | Kubespray | Kops |
| Support | 5 | 4 | 4 |
| Documentation | 5 | 5 | 5 |
| Flexibility | 2 | 5 | 3 |
| Cloud Provider Support | 5 | 4 | 2 |
| Complexity | 4 | 4 | 4 |
| Total: | 21 | 22 | 18 |

## Conclusion

Based on the comparison of the three tools, Kubespray outperforms Kubeadm and Kops in several key areas. The key differences where:

* Flexibility:
  + Kubespray is the most flexible among the three tools. Because Kubespray is like a library of different automation scripts, it allows users to make adjustments to these scripts where needed.
* Cloud Provider Support:
  + Although Kubespray came in second in this category, the fact that it supports multiple cloud providers (such as AWS, GCP, and Azure) and had other providers on the way, made it a good choice for this project.
* Complexity:
  + The combination of flexibility and complexity that Kubespray has made it a great choice. The advanced configuration options, which make Kubespray more flexible, also make it more complex. However, because something is more complex doesn’t mean it is always a bad thing. The ability to make complex changes is a good thing in this case.

Considering these factors, Kubespray stands out as our recommended tool for upgrading Kubernetes cluster version in this project.

# Overall Conclusion

### Upgrade strategies

Based on the requirements for this project we have concluded that a rolling upgrade strategy would be best suited for this use case. It facilitates all of the important aspects of the upgrade process like high availability, low cost and complexity while also allowing for rollback if necessary. This conclusion is based on exploring existing literature on the matter and comparing it to the use case requirements.

### Tools

Similarly to the update strategy we based our choice on the use case requirements and tests performed. The choice of tool which would best suit this project would be Kubespray. Despite its complexity Kubespray is highly flexible and allows for further customization which would help accommodate different use cases. It is possible to use it with the major cloud providers like AWS, Azure and GCP and on premise. The only downside to this tool would be the lack of on-demand support which combined with the complexity of the tool can potentially be a point of concern.