Top 30 Kustomize Interview Questions and Answers

by FOSS TechNix

In this article we are going to cover Kustomize Interview Questions and Answers, Kustomize commands Interview Questions and answers and scenario based kustomize interview questions and answers.

Table of Contents

Kustomize Interview Questions and Answers

What is Kustomize, and how does it differ from other configuration management tools in the Kubernetes ecosystem?

Kustomize is a tool for customizing Kubernetes deployments. It allows you to build, customize, and manage Kubernetes deployments without having to write or edit YAML files.

Kustomize is a configuration management tool for Kubernetes that allows you to customize Kubernetes manifests without directly modifying them. It uses a declarative approach, enabling users to define desired states without having to write complex YAML configurations. Kustomize differs from tools like Helm because it focuses on patching and transforming existing manifests rather than templating them.

Kustomize uses several key concepts, including:

- **Bases:** A base is a collection of Kubernetes resources that represent a reusable component of an application.
- **Patches:** A patch is a set of instructions that modifies a base.
- **Resources:** Resources are the individual Kubernetes resources that make up an application.

What are the benefits of using Kustomize?

There are several benefits to using Kustomize, including:

- **Ease of use:** Kustomize makes it easy to build, customize, and manage Kubernetes deployments.

- **Consistency:** Kustomize ensures that deployments are consistent across environments.
- **Reusability:** Kustomize allows you to reuse bases and patches to build new deployments.
- **Declarative:** Kustomize uses a declarative syntax, which makes it easier to understand and maintain deployments.

What are the key components of Kustomize?

The key components of Kustomize are:

- **Kustomization files:** Kustomization files are YAML files that define bases and patches.
- **ConfigMapGenerator:** The ConfigMapGenerator is a tool that generates Kubernetes ConfigMaps from Kustomization files.
- **KubeBuilder:** KubeBuilder is a tool that generates Kubernetes resources from Kustomization files.

What is a Kustomization file?

A Kustomization file is a YAML file that defines a base or patch. It contains information about the resources in the base or patch, as well as instructions for customizing the resources.

What is a base?

A base is a collection of Kubernetes resources that represent a reusable component of an application. Bases are typically stored in a separate directory from the application that they are part of.

What is a patch?

A patch is a set of instructions that modifies a base. Patches are typically used to add or remove resources from a base, or to modify the configuration of existing resources.

What is a resource?

A resource is an individual Kubernetes resource, such as a Pod, Deployment, or Service.

Explain the concept of overlays in Kustomize.

Answer: Overlays in Kustomize are a way to layer configurations on top of a base set of resources. Instead of modifying the original resource files directly, overlays provide a mechanism to apply changes in a structured and modular way. Overlays can be used to manage different environments (dev, staging, prod) or to apply variations based on different factors.

How does Kustomize handle environment-specific configurations?

Answer: Kustomize allows you to use overlays to manage environment-specific configurations. You can create overlays for each environment and apply them selectively. This enables you to keep a common set of base configurations and apply environment-specific changes without duplicating the entire set of manifests.

What are Kustomize patches, and how do they work?

Answer: Kustomize patches are used to modify or add to existing resources defined in base manifests. Patches are applied in a specific order and can target resources based on labels, names, or other criteria. Patches are typically written in JSON or YAML format and are used to customize resources without modifying the original manifests directly.

Explain the difference between a Kustomization file and a Kustomize patch.

Answer: The Kustomization file is the configuration file used by Kustomize to define the customization options for a set of resources. It can include information about the base resources, overlays, and patches. On the other hand, a Kustomize patch is a separate file that contains specific modifications or additions to the base resources. The Kustomization file references these patches to apply the desired changes.

Kustomize Commands Interview Questions and Answers

What is the command to build a Kustomize resource?

The command to build a Kustomize resource is `kustomize build <path_to_kustomization_file>`. This command will generate the Kubernetes resources for the base or patch.

What is the command to apply a Kustomize resource?

The command to apply a Kustomize resource is `kubectl apply -k <path_to_kustomization_file>`. This command will deploy the Kubernetes resources for the base or patch to the Kubernetes cluster.

What is the command to diff a Kustomize resource?

The command to diff a Kustomize resource is `kubectl diff -k <path_to_kustomization_file>`. This command will show the changes that will be made to the Kubernetes resources when the resource is applied.

Please Explain kustomize Commands

Kustomize comes with a set of commands that you can use to customize and manage your Kubernetes manifests. Here are some of the commonly used commands:

```
kustomize build
```

Description: This command is used to generate customized Kubernetes manifests by applying Kustomize configurations.

Example:bash

```
kustomize build path/to/kustomization/directory
```

```
kustomize create
```

- **Description:** Creates a new Kustomization file in the current directory or the specified path.
- **Example:**bash

- `kustomize create --resources path/to/base/resources`

```
kustomize edit
```

- **Description:** Opens the Kustomization file in the default editor for modification.
- **Example:**bash

- `kustomize edit path/to/kustomization/directory`

```
kustomize build --output
```

- **Description:** Generates customized manifests and writes them to the specified file.
- **Example:**bash

- `kustomize build path/to/kustomization/directory --output=path/to/output.yaml`

```
kustomize validate
```

- **Description:** Validates the Kustomization file for correctness.
- **Example:**bash

- `kustomize validate path/to/kustomization/directory`

```
kustomize version
```

- **Description:** Displays information about the installed Kustomize version.
- **Example:**bash

- `kustomize version`

```
kustomize create secret
```

- **Description:** Creates a new secret generator in the Kustomization file.

- **Example:**bash

- `kustomize create secret generic mysecret --from-literal=username=admin --from-literal=password=secretpassword`

`kustomize create configmap`

- **Description:** Creates a new ConfigMap generator in the Kustomization file.
- **Example:**bash

- `kustomize create configmap myconfigmap --from-file=path/to/config/file`

`kustomize edit add resource`

- **Description:** Adds a resource to the Kustomization file.
- **Example:**bash

- `kustomize edit add resource path/to/resource.yaml`

`kustomize edit set image`

- **Description:** Sets or modifies the image for a container in the Kustomization file.
- **Example:**bash

- `kustomize edit set image my-container=myregistry/myimage:mytag`

`kustomize edit add label`

- **Description:** Adds a label to resources in the Kustomization file.
- **Example:**bash

- `kustomize edit add label app=myapp`

These commands are just a subset of the functionalities provided by Kustomize. You can explore additional options and parameters for each command by referring to the official Kustomize documentation.

Scenario based Kustomize Interview Questions and Answers

How do you manage environments with Kustomize?

Kustomize can be used to manage environments by using different bases and patches for each environment. For example, you could have a base for the development environment, a patch for the staging environment, and a patch for the production environment.

How do you integrate Kustomize into a CI/CD pipeline?

Kustomize can be integrated into a CI/CD pipeline by using the `kustomize build` and `kubectl apply` commands in your pipeline steps. This will automate the deployment of your applications to Kubernetes.

What are some advanced Kustomize features?

Kustomize has several advanced features, including:

- **ConfigMap generators:** ConfigMap generators allow you to generate Kubernetes ConfigMaps from Kustomization files.
- **Environments:** Kustomize supports environments, which allow you to manage different configurations for different environments.
- **Overlays:** Kustomize supports overlays, which allow you to layer patches on top of each other.

How can you integrate Kustomize into a CI/CD pipeline for Kubernetes deployments?

Answer: In a CI/CD pipeline, you can use Kustomize to customize your Kubernetes manifests based on the deployment environment. This involves running the Kustomize build command to generate the final manifests before applying them to the cluster. The Kustomization file, overlays, and patches are typically stored in version control, allowing the pipeline to pull the necessary configurations based on the deployment stage.

What are the potential challenges or limitations of using Kustomize?

Answer: Some challenges of using Kustomize may include a learning curve for newcomers, especially those accustomed to other configuration management tools like Helm. Additionally, complex customizations might require a deep understanding of Kustomize features. Users should also be aware of potential compatibility issues with specific Kubernetes versions.

Can you give an example of a scenario where Kustomize would be particularly beneficial over other configuration management tools?

Answer: Kustomize is particularly beneficial when managing configurations across different environments or when applying customizations to existing manifests. For example, if you have a set of base manifests for a microservices architecture and need to deploy variations of these services to multiple environments, Kustomize's overlay and patching system makes it easy to manage these differences without duplicating or modifying the entire set of manifests.

How does Kustomize help in achieving a GitOps workflow?

Answer: Kustomize supports the GitOps model by allowing you to version control your base manifests, overlays, and patches. This enables teams to manage changes to Kubernetes configurations through pull requests and promotes a declarative approach to infrastructure management. The Kustomization files and associated configurations become part of the version-controlled repository, making it easier to track changes and rollbacks.

Can you explain the process of using Kustomize to manage secrets in Kubernetes?

Answer: While Kustomize itself doesn't handle secrets directly, it can work in conjunction with tools like Kubernetes Secrets, external secret management systems, or environment variables. You can reference secrets in your base manifests and use Kustomize overlays to provide different values for secrets in each environment. This allows you to keep sensitive information separate from your base configurations and manage them securely in your CI/CD pipeline.

Scenario 1:

Question: You have a microservices-based application with a common set of resources and configurations. However, each microservice requires a slight variation in its configuration. How would you use Kustomize to manage this scenario efficiently?

Answer: In this scenario, I would create a base set of manifests for the common resources using a Kustomization file. Then, I would use overlays to customize each microservice's configuration. Each overlay would contain the specific changes needed for that microservice. By applying these overlays selectively, I can generate customized manifests for each microservice without duplicating the common configurations.

Scenario 2:

Question: Your team is working on a project with multiple environments: development, staging, and production. How would you structure your Kustomize setup to handle configurations for each environment?

Answer: I would organize my project with a directory structure that includes a base directory for common configurations and separate overlay directories for each environment. The Kustomization file in each overlay directory would reference the base resources and include environment-specific changes. This approach allows us to manage environment-specific configurations in a modular and maintainable way.

Scenario 3:

Question: You have a Kubernetes cluster with applications that require secrets for sensitive information. How would you manage these secrets using Kustomize?

Answer: I would use Kustomize to create secret generators in the Kustomization file. This allows me to define the secrets in a secure and declarative manner. Additionally, I can use overlays to customize the secret values for different environments. By keeping the secret definitions separate from the base resources, I ensure that sensitive information is managed independently and securely.

Scenario 4:

Question: You are working on a project where some Kubernetes resources need to be customized based on the team responsible for the application. How would you approach this using Kustomize?

Answer: I would use Kustomize patches to customize resources based on the team responsible for each application. Each team would have its own patch file containing the necessary modifications. The Kustomization file would reference these patches, and by applying the appropriate patches for each team, I can customize the resources accordingly.

Scenario 5:

Question: Your team practices a GitOps workflow, and you want to integrate Kustomize into your continuous integration pipeline. How would you set up this pipeline to ensure smooth deployments?

Answer: In the CI pipeline, I would use the `kustomize build` command to generate the customized manifests based on the environment. These manifests would then be applied to the Kubernetes cluster using `kubectl apply` or a similar command. The Kustomization files, overlays, and patches would be version-controlled, and the pipeline would trigger deployments based on changes to the configuration repository, ensuring a GitOps-driven approach.

Scenario 6:

Question: You have a Kubernetes application with multiple microservices, and you need to update the image for one of the containers in all services. How would you achieve this efficiently with Kustomize?

Answer: I would use the `kustomize edit set image` command to update the image for the specific container in the Kustomization file. This change would then be applied across all microservices when generating the manifests. By centralizing the image update in the Kustomization file, I can ensure consistency and efficiency in updating container images.

Conclusion:

We have covered Kustomize Interview Questions and Answers, Kustomize commands Interview Questions and answers and scenario based kustomize interview questions and answers.