Documentation of dgb-kube-runner

This document is generated by AI. Learn more in the feature documentation.

The objective of this integration is to manage the deployment of Kubernetes (K8s) resources through a pipeline. The process starts with the receipt of an event triggering the deployment action. The event contains information about the deployment, such as the deployment name, namespace, and other relevant details.

The first step in the integration is to add the Kubernetes deployment information to the pipeline session, ensuring that it is available for subsequent steps. This step is crucial as it sets the context for the deployment process and ensures that the relevant information is accessible throughout the pipeline.

Following this, the integration involves updating the status of the Kubernetes deployment in an object store. This step is important as it provides a mechanism to track the status of the deployment, including recording the start of execution, timestamp, and setting the status to "stand-by". This allows for effective monitoring and management of the deployment process.

Subsequently, the integration retrieves the Kubernetes deployment information from the pipeline session. This step is essential as it ensures that the deployment details are accurately captured and available for further processing within the pipeline.

The integration then involves making a choice based on the action to be performed, such as creating or deleting a Kubernetes deployment. This decision-making step is critical as it determines the subsequent actions to be taken based on the received event and the desired operation.

Depending on the choice made, the integration either creates a Kubernetes deployment based on a YAML configuration or deletes an existing deployment. These actions are fundamental as they directly impact the Kubernetes infrastructure, facilitating the deployment or removal of resources based on the specified criteria.

Throughout the process, error handling and logging are incorporated to capture and manage any potential issues that may arise during the deployment process. This ensures that any errors or exceptions are appropriately recorded and managed, contributing to the overall reliability and robustness of the integration.

In conclusion, the integration orchestrates the deployment and management of Kubernetes resources based on the received event, utilizing a series of steps to handle the deployment process, track the status, and manage any potential errors that may occur.

External Systems Involved

- No external systems found in the pipeline

Events

- No events found in the pipeline

Globals

- {{global.ext-host-kube-port}}
- {{global.ext-host-kube-hostname}}
- {{global.ext-callback-kube-url}}
- {{global.ext-host-kube-username}}
- {{global.ext-callback-kube-apikey}}

Global	Step Name
{{global.ext-host-kube-port}}, {{global.ext-host-kube-hostname}}, {{global.ext-callback-kube-url}}, {{global.ext-host-kube-username}}, {{global.ext-callback-kube-apikey}}	Create a K8s Deploy Based on a YAML with Dyn Name & Callback
{{global.ext-host-kube-port}}, {{global.ext-host-kube-hostname}}, {{global.ext-callback-kube-url}}, {{global.ext-host-kube-username}}, {{global.ext-callback-kube-apikey}}	Delete a K8s Deploy Based on a YAML with Dyn Name & Callback

Accounts

- kuberunner-private-key

Account	Step Name
kuberunner-private-key	Create a K8s Deploy Based on a YAML with Dyn Name & Callback
kuberunner-private-key	Delete a K8s Deploy Based on a YAML with Dyn Name & Callback