**Stakater Reloader Use Case**

Background

Stakater Reloader is a Kubernetes controller designed to automatically trigger a pod restart when a ConfigMap or Secret is updated. It is primarily used to ensure that your application always uses the most recent configuration values or secrets without requiring manual intervention. This is crucial for dynamic environments where secrets or configurations might change frequently, and you want these changes to be reflected in your applications seamlessly.

The main goal of Reloader is to automate the process of applying updates to running pods that are dependent on specific ConfigMaps or Secrets. When a change occurs in a monitored ConfigMap or Secret, Reloader will automatically trigger a rolling restart of the associated pods, ensuring that they pick up the latest version of the data.

In our environment today, we are using Reloader in combination with External Secrets Operator (ESO). ESO is used to automatically update EKS secrets with values from AWS Secrets manager, and ESO watches AWS Secrets manager for any changes/updates to the values. ESO then updates local secrets hence the need for Reloader to monitor local secrets for updates so that it can trigger rolling restarts of application pods.

Analysis

Reloader was initially deployed via the helm chart available at the vendors website (<https://stakater.github.io/stakater-charts>). By default, reloader monitors all namespaces upon initial deployment but can be restricted to specific namespaces if need be. Reloader watches for ConfigMaps and Secrets that are annotations in the application pods manifest label section.

This helm chart was installed via a helm install command after adding the external repo.

Desired Result

The desired goal is to have reloader deployed as part of the entire suite of pipeline components, via packaging it as a dependency into the master helm chart which deploys all pipeline components currently. This will simplify pipeline deployment as a whole and package the required add-ons needed for our solution to function. This will also allow any necessary customizations to reloader to be implemented/managed via the master helm chart, as well as proper version control. The problem that may arise is the order of chart applications deployment can possibly cause reloader to fail. In-depth testing will need to occur to validate this.

Next Steps

The next steps are to test deploying Reloader as a helm dependency in a local helm chart and installing that, and depending on those results then migrate the dependency chart to the main pipeline application helm chart and deploying locally via minikube and adding results to this document.