Deploying applications in Kubernetes using Argo CD

Pavan Kumar
Sep 1, 2020 · 5 min read

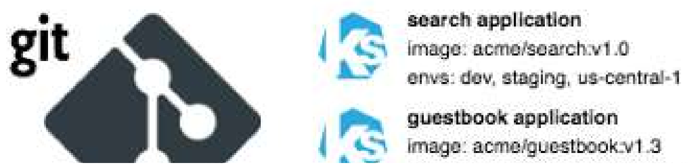
What is Argo CD?

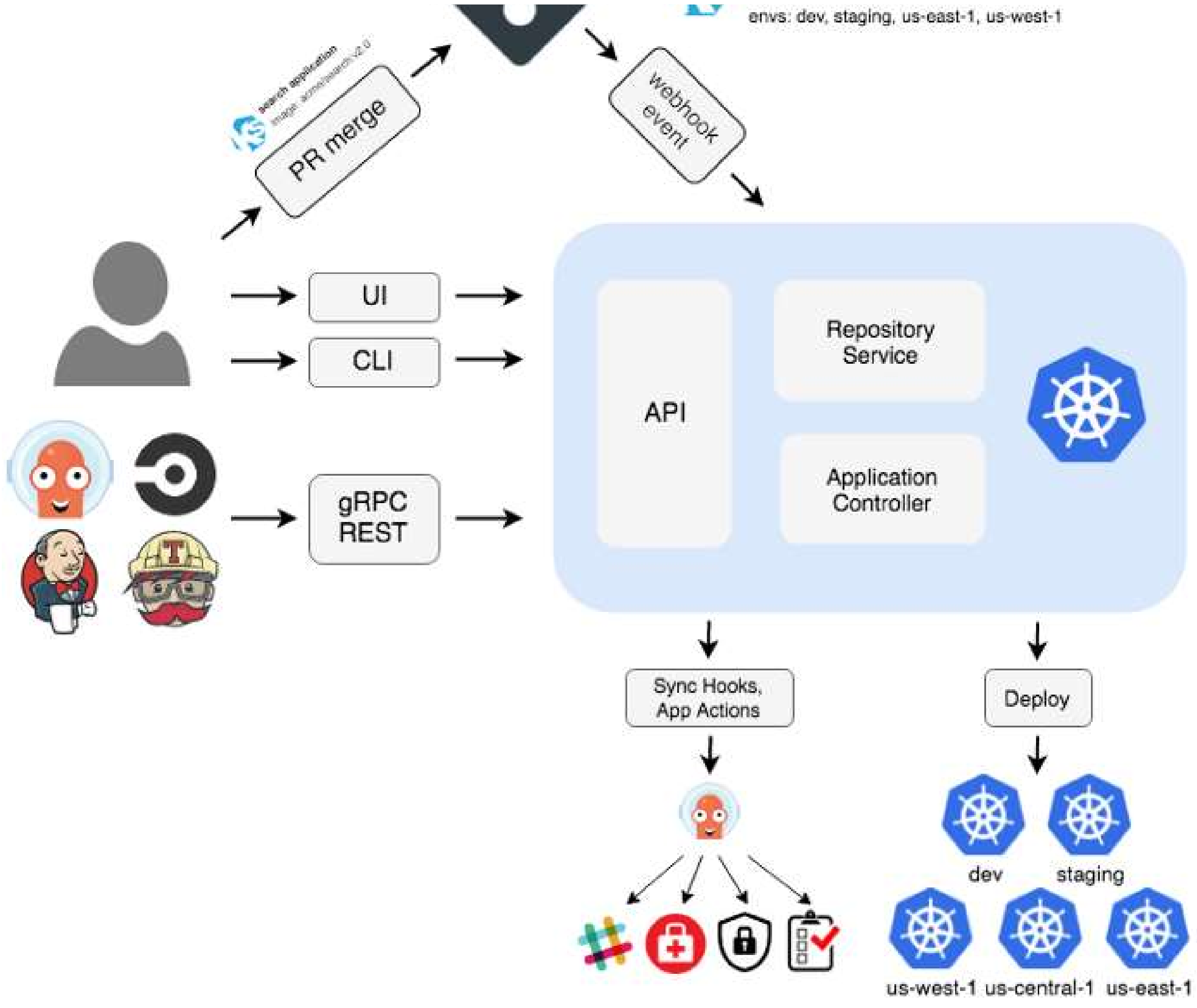
Argo CD is a declarative, GitOps continuous delivery tool for Kubernetes. So what is GitOps? Is it a new tool in the market? GitOps provides a way for developers to manage operational workflow for using Kubernetes using Git. It is all about using a version-controlled system for deployment of applications in Kubernetes. So Developers can directly push the code into the production from the version-controlled system like Git. And moreover, any changes made can be easily tracked and reverted back in case of any chaos. There are multiple tools in the market to run GitOps. Today in this article we would be experimenting on a tool called Argo CD.

Argo CD is a declarative, GitOps continuous delivery tool for Kubernetes. It supports Kubernetes manifests specified in

1. kustomize
2. helm charts
3. YAML Manifests
4. ksonnet applications
5. jsonnet files

Let us look into the architecture of Argo CD





Argo CD Architecture

Now since we are now familiar with the concepts of Argo CD, let us install Argo CD in our Kubernetes Cluster.

Follow the steps specified in the ***official documentation of Argo CD*** to Install and configure Argo CD in your Kubernetes Cluster.

In the first step let us configure the private key associated with our private Github repository. This step makes all our manifest files in our Private Github repository accessible by our Argo CD server. This can be done by issuing a simple argocd command

> ***argocd repo add git@github.com:pavan-kumar-99/lamp-k8-s-application.git — ssh-private-key-path id_rsa ~/.ssh/argo_priavate_key***

Let us also verify the same from the Argo CD console

Yasss!! The connection to our Private repository is now successful. Let us go ahead and deploy our applications into our Kubernetes cluster.

In the UI create a new application by clicking on the new app icon.

Argo CD UI

In the above snapshot, you can see various parameters to configure your application.

1. Application Name: The name of your application in Argo CD
2. Project: The Project in which you want to deploy your application
3. Sync Policy: Whether the sync of manifests from GitHub should be automatic or manual. (Let's put that as manual as of now as we are going to explore History and Rollback features in Argo CD)
4. Repository URL: The URL of your GitHub Repository
5. Revision: The Branch to be used in the GitHub Repository.

Selection of Cluster

6. Cluster URL: The Kubernetes cluster to where you want to deploy your application.

7. Namespace: The Namespace to which the application has to be deployed.

The agro-cd folder in my GitHub repo has a simple Nginx Deployment file with the Image version to be **1.18.** Now let us sync our application manually. Once the sync is successfully you would see something like this in your Argo CD console

Argo CD Console

You can find all the components defines in your manifests file here. The Synced output says that the application is synced with the GitHub repository. Now let us try changing the version of the Nginx image to **1.19** in the GitHub. Since the AutoSync feature is disabled the changes are not replicated instantly. Else as soon as the change is changed in the GitHub and merged to the branch specified in the project the changes are instantly replicated in the Kubernetes cluster.

Once you make the change you see that the Sync Status is **out of sync**. Let's review what has been changed in the Argo CD console by clicking the App Diff Section.

As it is clearly seen that the Image has been changed. Let us sync this change with our cluster. Now you see that the Sync status is back to Synced with the GitHub. Now everything is going good and suddenly your containers start to fail. After you identify that your code is having an issue with version 1.19 and you want to rollback to the previous version of your application. Click on the History and Rollback option in the Console and choose the revision that you want to roll back to. Now you see that your application gets rolled back to the older version and the sync status changes to out of sync.

The status is set to Out of Sync because the state of the Argo CD is not aligned with the manifests in the GitRepo. Making the changes in the Git Repo would change the status in Argo CD UI.

There are a few other scenarios, which I would be covering in my next article.

Recommended

Introduction to Istio Service Mesh

What is Istio Service Mesh?

medium.com

Securing your secrets using vault-k8s in Kubernetes — Part 1

Kubernetes secrets let you store and manage sensitive data such as passwords, ssh keys, Tls certificates, etc. However...

medium.com

Argo Cd Kubernetes Rollbacks Docker K8s