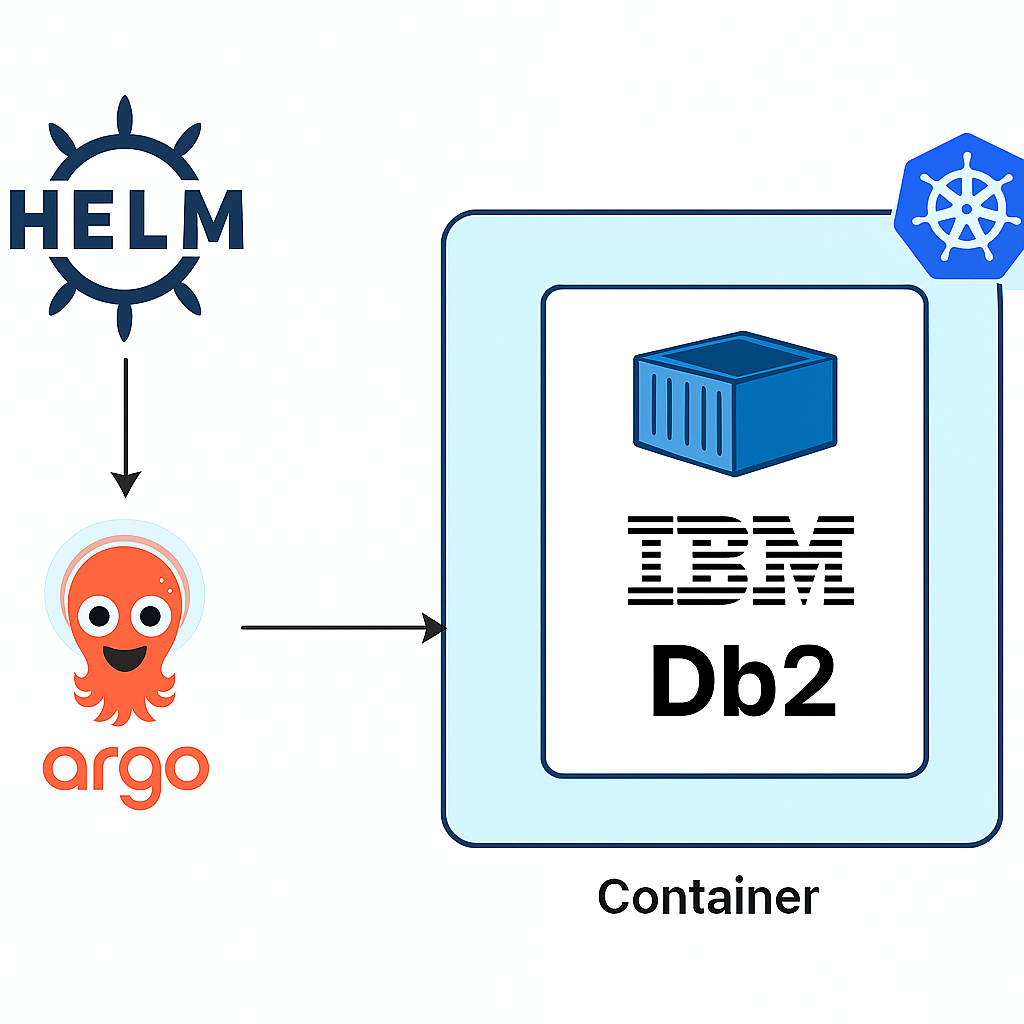
Next-Gen IBM Db2 on Kubernetes: A GitOps Journey with Argo CD and Helm



# Introduction: -

GitOps with Argo CD uses Git as the single source of truth for managing Kubernetes deployments, enabling automated, declarative, and version-controlled operations. This approach ensures that infrastructure and application states are always aligned with what's defined in Git, reducing manual intervention and configuration drift.

Argo CD continuously monitors Git repositories for changes and syncs them to the OpenShift and Kubernetes cluster. It enforces a desired state model, meaning any deviation from the Git-defined configuration is automatically corrected or flagged. This not only improves deployment consistency and security but also simplifies rollbacks and auditing. By combining GitOps principles with Argo CD, teams gain a powerful workflow for scalable, reliable, and transparent cloud-native application delivery.

If you are new to Db2 containerization, check out [**this**](https://www.idug.org/news/modernizing-db2-containerization-footprint-with-db2u) IDUG blog which gives a comprehensive overview of the current gen Db2U operator and the underpinning Kubernetes concepts.

# Enhancing Db2 Deployments with Argo CD

Deploying IBM Db2 in a containerized Kubernetes platform becomes significantly more efficient and transparent when powered by Argo CD’s GitOps principles.

## Real-Time Visibility into Kubernetes Resources

* 1. Argo CD provides a live dashboard that visualizes all Kubernetes resources associated with your Db2 deployment, such as deployments, pods, db2uengine, services, ConfigMaps, Secrets, and PersistentVolumeClaims.
  2. You can inspect the resource tree for your Db2 application and see how each component is connected and deployed.

## Monitoring Db2 Operator and Application Pods

* 1. Argo CD shows the status of the Db2 Operator pod, which orchestrates the lifecycle of Db2 instances.
  2. You can drill down into specific Db2 pods to:
     1. View health status and readiness probes.
     2. Access live pod logs directly from the UI for troubleshooting and performance monitoring.
     3. Restart or delete pods if needed, all within the GitOps workflow.

# Prerequisites for Next-Gen GitOps Deployment

To successfully deploy IBM Db2 using GitOps principles with Argo CD and Helm, you’ll need a solid foundation. Here are the key prerequisites:

1. Kubernetes Version (v1.31 or higher)
2. Helm (a widely adopted Kubernetes package manager)
3. Argo CD
4. Access to a Github repository with a personal access token for the git user

# Installing Helm and setting up Argo CD for Next-Gen GitOps Deployment

In this blog, we’ll walk through the installation of Helm and Argo CD, two essential tools for enabling GitOps driven deployments of IBM Db2 on Kubernetes.

## Installing Helm

To install Helm, follow the instructions at <https://github.com/helm/helm?tab=readme-ov-file#install>

## Install the Argo CD CLI

You need to install the Argo CD Command Line Interface (CLI) to use the argocd commands

The installation method depends on your operating system. Here are common ways to install it:

1. Using a Package Manager (Recommended for macOS/Linux)

macOS (using Homebrew):

brew install argocd

Linux (using a package manager like apt or yum): Installation varies. The most universal approach on Linux is often the direct download method below.

1. Direct Download and Manual Installation (Universal)

Download the latest release:

curl -sSL -o /usr/local/bin/argocd <https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64>

(Note: Replace argocd-linux-amd64 with the appropriate binary for your system, e.g., argocd-darwin-amd64 for macOS.)

1. Make it executable:

chmod +x /usr/local/bin/argocd

1. Verification

After installation, verify that the command is available:

argocd version

## Setting Up Argo CD for GitOps-Driven Db2 Deployments

With Helm installed and your OpenShift/Kubernetes cluster ready, the next step in building a GitOps pipeline for IBM Db2 is installing Argo CD. Argo CD acts as the GitOps controller, continuously syncing your Git repository with your OpenShift/Kubernetes cluster and ensuring that your Db2 deployments, like db2oltp or db2whmpp, stay in their desired state.

In this section, we’ll walk through the installation of Argo CD using Helm. This method is clean, repeatable, and aligns perfectly with the declarative infrastructure-as-code philosophy.

### Create a Dedicated Namespace

To keep things organized, we’ll start by creating a separate namespace for Argo CD:

kubectl create ns argocd

This isolates Argo CD components from other workloads in your cluster.

You can set the default namespace to argocd

kubectl config set-context --current --namespace=argocd

### Add the Argo CD Helm Repository

Next, add the official Argo CD Helm chart repository

helm repo add argo-cd https://argoproj.github.io/argo-helm

Verify the repo has been added properly

helm repo list

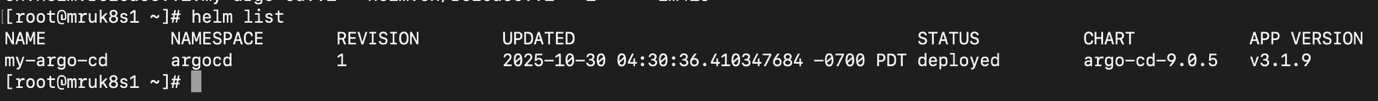
This gives you access to the latest stable charts maintained by the Argo project.

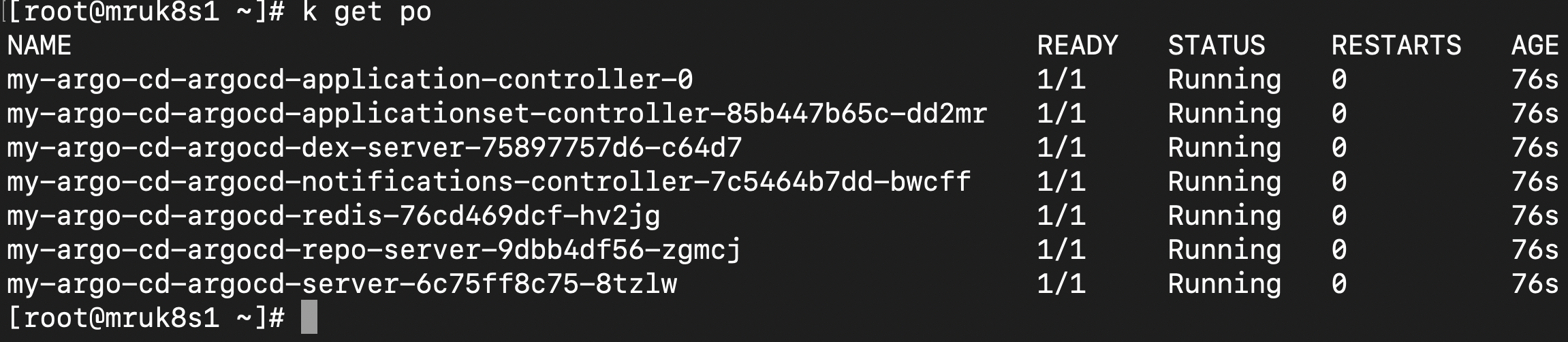
### Install Argo CD Using Helm

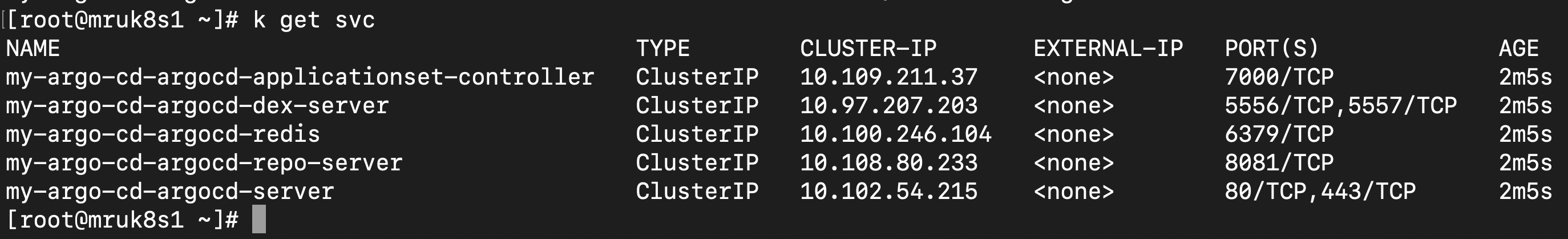
Now deploy Argo CD into your cluster:

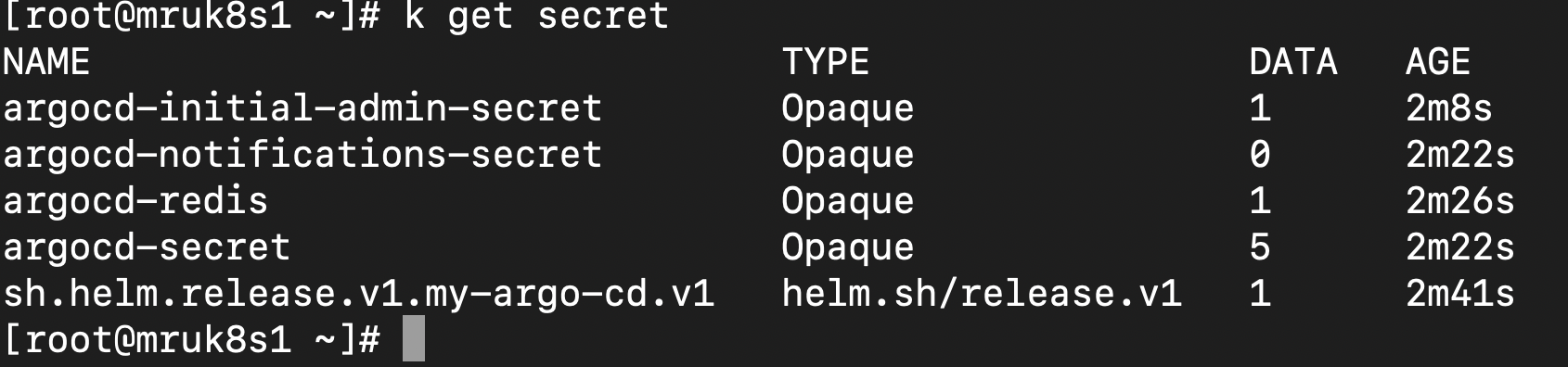
helm install my-argo-cd argo-cd/argo-cd

Verify the deployment









This installs Argo CD under the release name my-argo-cd in the argocd namespace.

Reference: -

For advanced configuration options, visit: <https://github.com/argoproj/argo-helm/tree/main/charts/argo-cd>

### Install Route to host Argo CD UI (OpenShift cluster Only)

If you're deploying Argo CD on an OpenShift cluster, exposing the Argo CD UI requires creating an Open Shift Route resource. This allows you to access the Argo CD dashboard externally via a secure URL.

Run the following command to create the Route:

oc create route passthrough route-argocd-passthrough --service=my-argo-cd-argocd-server --port=https

Once the route is created, you can retrieve the URL with:

You can read more about OpenShift routes at the blog [here](https://www.redhat.com/en/blog/encryption-secure-routes-openshift#:~:text=The%20resulting%20route%20resource%20should,in%20a%20PEM%2Dencoded%20file).

### Access the Argo CD UI via Route Host/Post

Once the Route is created, OpenShift automatically assigns a hostname that exposes the Argo CD UI externally. You can retrieve this hostname using the following command:

oc get route -n argocd

In this example, the Argo CD UI is accessible at: https://route-argocd-passthrough-argocd.apps.deb2.cp.fyre.ibm.com

Paste this URL into your browser to open the Argo CD dashboard.



### Log in to the Argo CD UI

Once you've reached the Argo CD login page via your OpenShift route or LoadBalancer host (for cloud providers), use the following credentials to sign in:

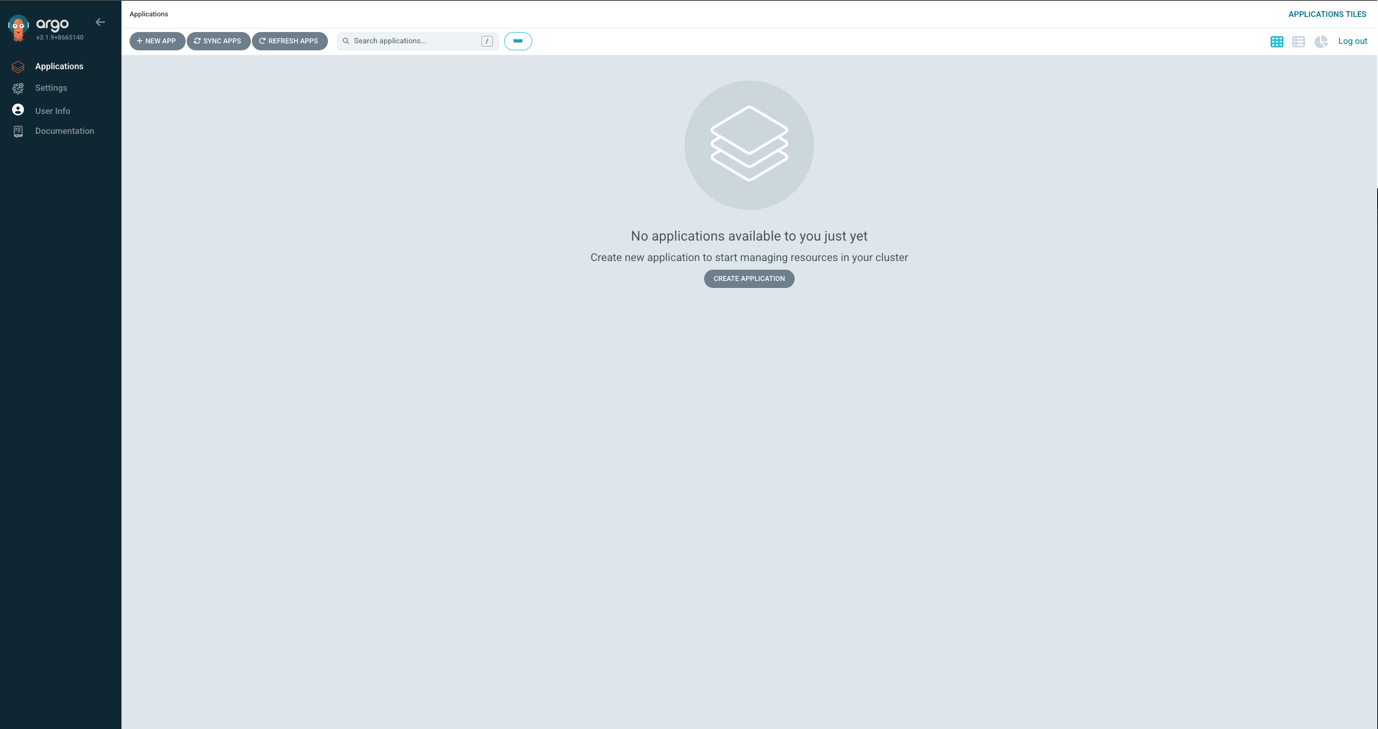
Username: - admin

Password: - echo "$(kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d)"

Note: - This command fetches the secret from the argocd namespace and decodes the password.

Once logged in, you’ll land on the Argo CD dashboard where you can start managing your GitOps workflows.

Once you login to the Argo Cd UI you will see something like:



### (Cloud Providers): Expose Argo CD UI via Load Balancer

If you're using a managed Kubernetes service such as AWS EKS, Azure AKS, or Google GKE, you don't need to install or configure HAProxy. These platforms provide built-in support for external service exposure using the LoadBalancer type.

Change Argo CD Service Type

To expose the Argo CD UI, simply patch the service to use a cloud-native load balancer:

kubectl patch svc my-argo-cd-argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'

After a few moments, check the service details

kubectl get svc my-argo-cd-argocd-server -n argocd

You should see an external IP or hostname assigned:



Access the UI. Open a browser and visit:

https://<LoadBalancer-Hostname>

For example: https://a1b2c3d4e5f6.elb.amazonaws.com

This will take you to the Argo CD login screen.

### Log in to Argo CD CLI from the Cluster

Once the Argo CD UI is exposed—either via route (OpenShift cluster) or LoadBalancer (for cloud providers)—you can also interact with Argo CD using its CLI tool directly from within the cluster.

CLI Login Command

Run the following command from a node or pod that has the Argo CD CLI installed:

argocd login route-argocd-passthrough-argocd.apps.deb2.cp.fyre.ibm.com --insecure --grpc-web

Replace the hostname with your actual route or LoadBalancer IP/hostname.

Provide Credentials

When prompted, enter:

Username: admin

Password: Use the same password retrieved earlier:

You should see output like:



### Connect a GitHub Repository to Argo CD

With Argo CD now accessible via UI and CLI, the next step is to connect your GitOps repository, typically hosted on GitHub for Enterprise, so Argo CD can start tracking and deploying your Kubernetes manifests.

We can either go for 11.1 or 11.2.

1. Add Repository via CLI

Use the Argo CD CLI to register your GitHub repository:

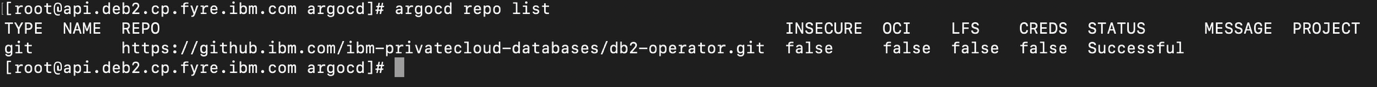
argocd repo add https://github.acme.com/db2-operator.git (This is an example repo where ACME is a fictional company using GitHub enterprise)

--username <your-github-username> \

--password <your-personal-access-token>

You should see confirmation like:

argocd repo list

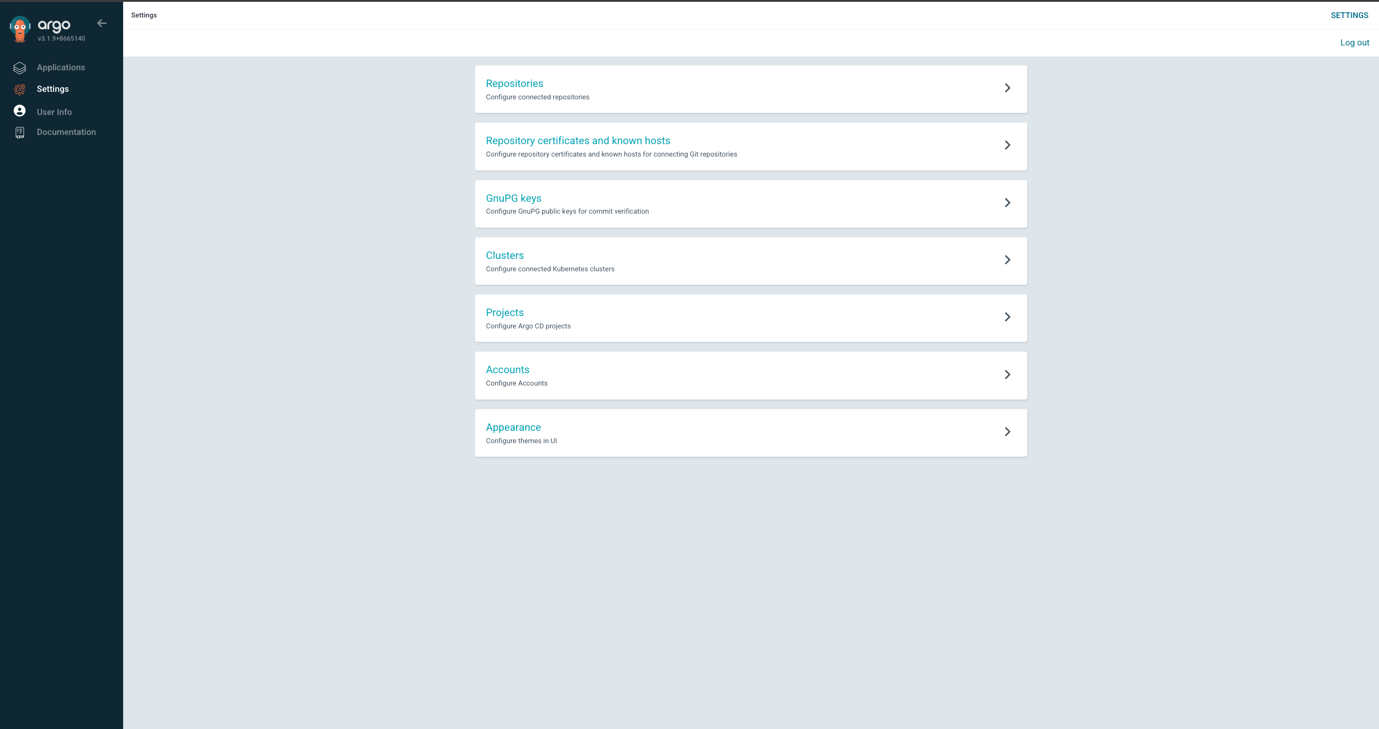


1. Add a GitHub Repository via Argo CD UI

Once you're logged into the Argo CD dashboard, you can connect your GitOps repository directly through the web interface. This allows Argo CD to monitor and deploy Kubernetes manifests stored in your GitHub repo.

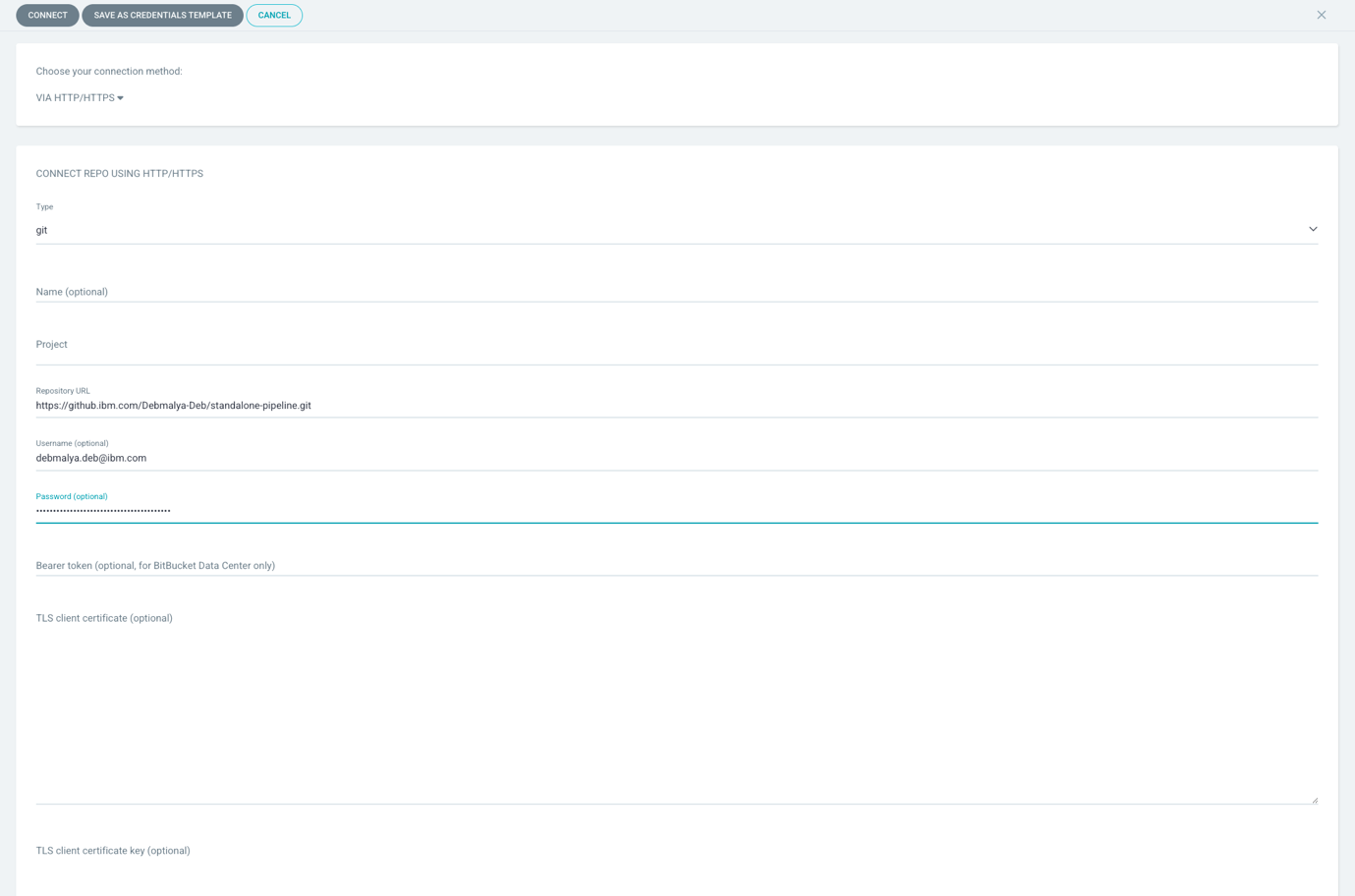
Navigate to the Repositories Page

1. Log in to the Argo CD UI (e.g., https://<oc get route>/ or your Load Balancer URL).
2. In the left-hand sidebar, click on Settings (gear icon).
3. Select Repositories from the dropdown menu.



Add a New Repository

1. Click the + New Repository button.
2. Fill in the following details:
3. Type: Git
4. Repository URL: https://github.acme.com/db2-operator.git (This is an example of a GitHub enterprise repo where ACME is a fictional company)
5. Username: Your GitHub username
6. Password: Your GitHub Personal Access Token (PAT)
7. Click Connect.



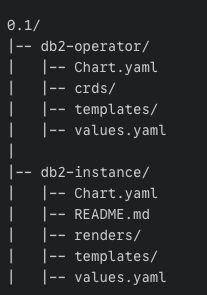
# Deploy DB2 Operator and DB2uInstance Applications

This step involves creating Argo CD applications that will deploy:

1. The DB2 Operator
2. Two DB2uInstance workloads:
   1. One for OLTP (Online Transaction Processing)
   2. One for DB2WH (Data Warehouse)

Prerequisites

Ensure your GitHub repository contains the following structure:



Each folder should contain the necessary manifests (template, charts, deployments, etc.) for the respective components.

## Installing Db2U Operator and Db2uInstance Helm Charts

With Argo CD and Helm in place, you're now ready to bring in the core components of your Db2 deployment: the Db2-Operator and the Db2uInstance. These are provided in a Helm chart tarball that can be requested from IBM Technical Support.

Key Resources

#### IBM Db2U Next-Gen Overview

<https://www.ibm.com/support/pages/db2u-next-generation> This page provides comprehensive documentation on architecture, features, and deployment guidance.

#### Official Helm Chart Tarball

db2-operator-0.1.tar.gz This tarball contains the Helm chart for the Db2-Operator and Db2uInstance (for both db2oltp and db2whmpp) and must be requested via IBM Technical Support.

#### Untar the Db2-Operator Tarball and Add to Enterprise GitHub Repository

Once you've obtained the official Helm chart tarball, the next step is to extract its contents and commit them to your GitHub repository. This prepares the chart for GitOps-based deployment via Argo CD.

Example Instructions Step-by-Step

1. Untar the Helm Chart

tar -zxvf db2-operator-0.1.tar.gz

cd 0.1

cd db2-operator

This will extract the chart into a directory containing chart.yaml, values.yaml, and the templates/ folder.

1. Initialize a Git Repository (if needed)

git init

1. Add the Chart to Git

git remote add origin https://github.acme.com/db2-operator.git

git add .

git commit -m "Initial commit of Db2-Operator Helm chart"

git push -u origin main

Replace acme with your enterprise domain. This pushes the chart to your GitHub repository, making it accessible to Argo CD for GitOps deployment.

## Deploying the Db2 Operator via Argo CD UI

With your Helm chart committed to your GitHub repository, the next step is to deploy the Db2 Operator using Argo CD’s UI. This process leverages Helm templating and GitOps automation to declaratively install the operator into your Kubernetes or OpenShift cluster.

1. Steps to Apply the Db2 Operator YAML
   1. Log in to the Argo CD UI

Open your browser and navigate to your Argo CD dashboard. This could be a LoadBalancer URL or a Route in OpenShift, for example:

* 1. Navigate to Applications

In the left-hand sidebar, click on Applications.

* 1. Create a New Application

Click the + New App button at the top.

* 1. Switch to YAML Mode

In the dialog that opens, switch to the YAML tab (top right corner).

* 1. Paste the Argo CD Application YAML

Use the following example YAML to define your Db2 Operator deployment:

An Example Argocd yaml for db2-operator yaml:-

apiVersion: argoproj.io/v1alpha1

kind: Application

metadata:

name: db2-operator

namespace: argocd

spec:

project: default

source:

repoURL: 'https://github.acme.com/db2-operator.git'

path: 'charts/db2-operator'

targetRevision: v0.1

helm:

values: |

namespace: <db2-operator\_namespace>

# Please add scc.create: false if it is a Kubernetes environment

platform: OCP # Use K8S for Kubernetes

image:

repository: icr.io/db2u/db2-operator

tag: 0.1

watchNamespace:

- <db2-operator\_namespace>

destination:

server: 'https://kubernetes.default.svc'

namespace: <db2-operator\_namespace>

syncPolicy:

automated:

prune: true

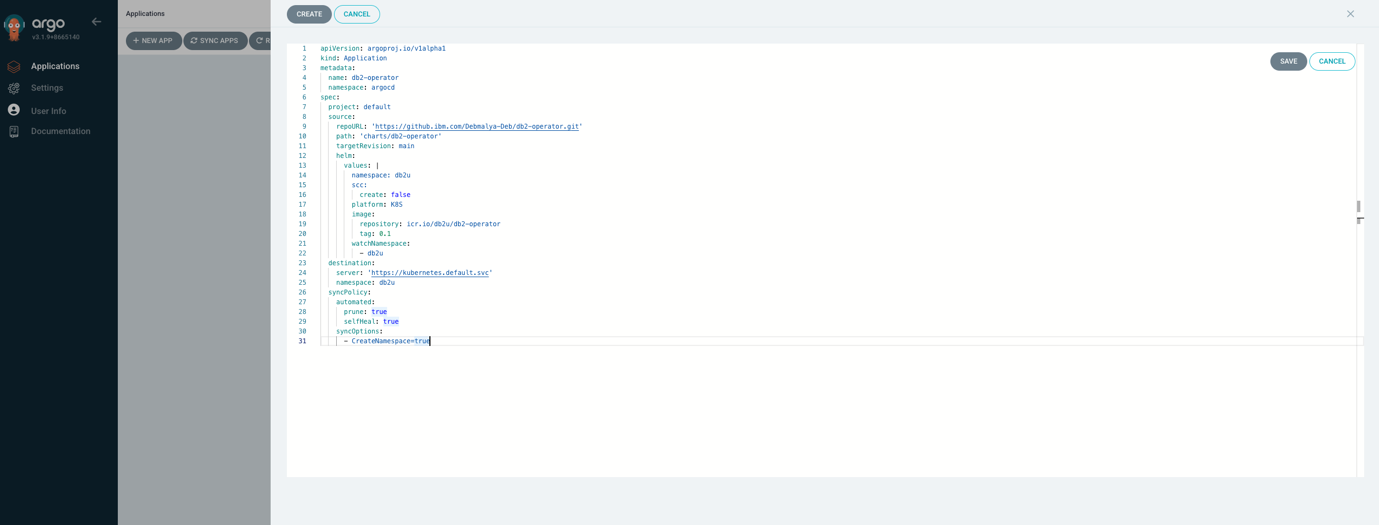
selfHeal: true

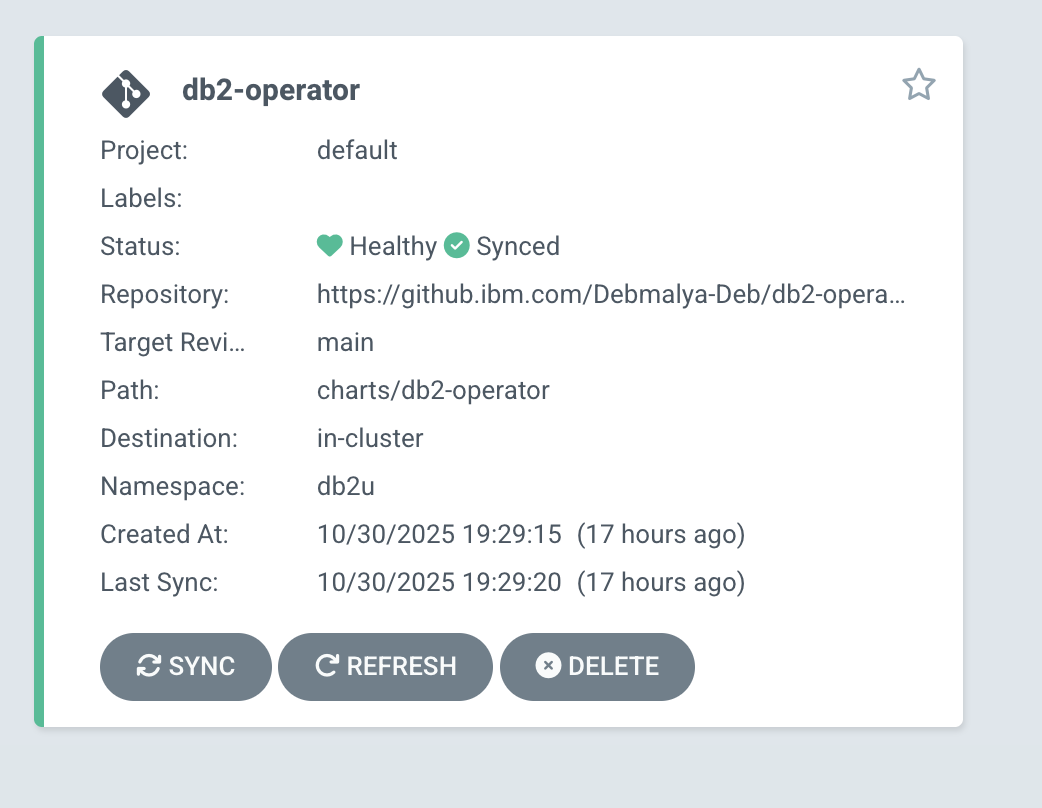
syncOptions:

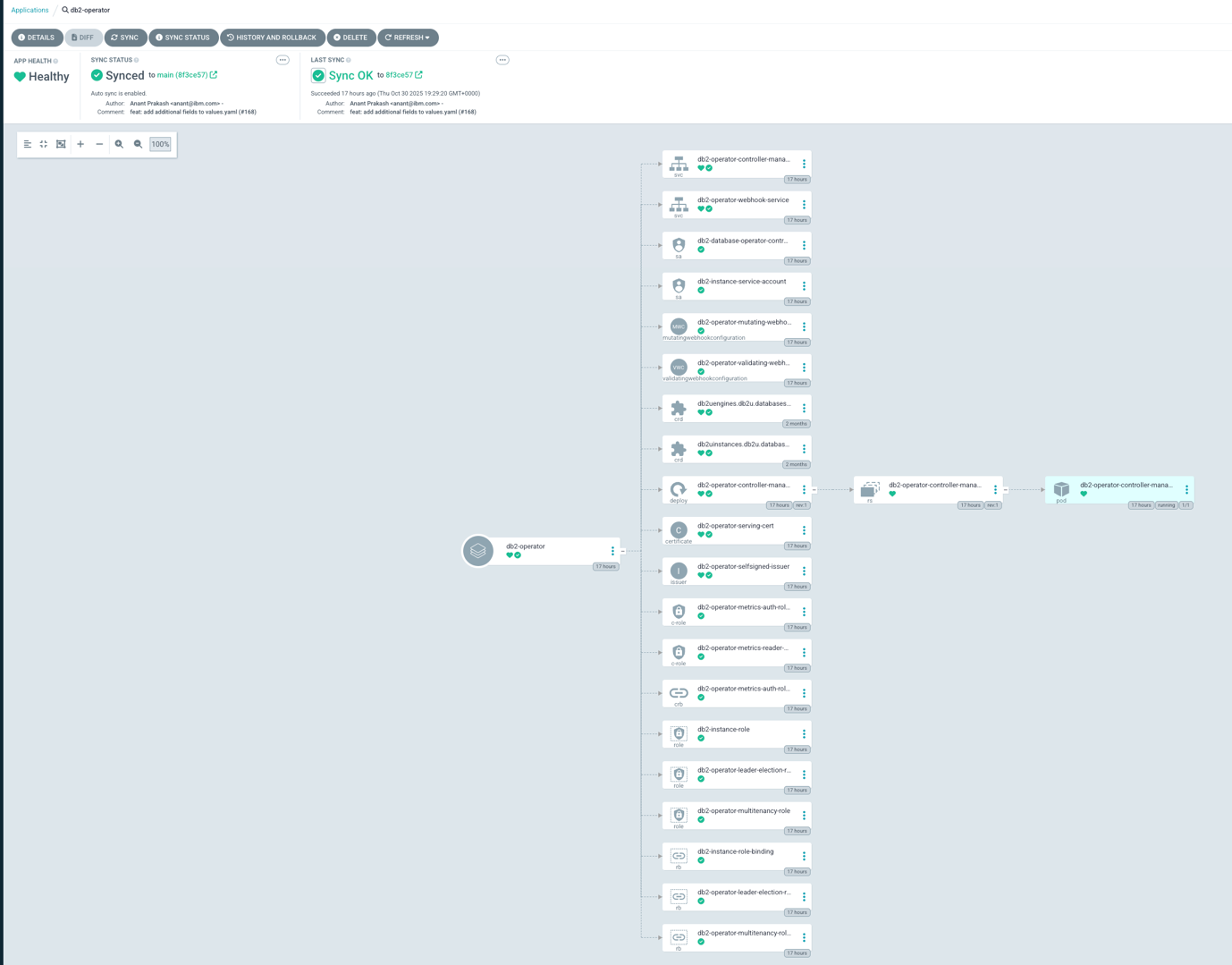
- CreateNamespace=true

<Notes:->

* This Argo CD Application uses Helm templating, meaning it will render and apply the templates defined in the chart located at charts/db2-operator.
* The values block overrides the chart’s default settings to tailor the deployment to your environment.
* Replace <db2-operator\_namespace> with the actual namespace you’ve created for the operator.
* Set platform appropriately based on whether you're using OpenShift (OCP) or vanilla Kubernetes (K8s).
* Once applied, Argo CD will sync the chart from your Git repository and deploy the Db2 Operator into your cluster.
  1. Here how it looks like in Argo CD UI



* 1. This yaml can be applied from the k8s/OCP cluster itself as well: kubectl/oc apply -f argo\_cd\_prod\_installOperator.yaml
  2. The deployment of db2-operator will look like this
  3. 



### Deploying DB2OLTP Db2uInstance Argo CD Applications

With the Db2 Operator deployed and synced via Argo CD, the next step is to deploy your Db2uInstance for transactional workloads (db2oltp).

Steps to Deploy Db2uInstance via Argo CD UI

1. Log in to the Argo CD UI Navigating to your Argo CD dashboard using the LoadBalancer or Route URL.
2. Go to Applications
3. In the sidebar, click on Applications.
4. Create a New Application
5. Click the + New App button.
6. Switch to YAML Mode
7. Paste the Argo CD Application YAML

Use the following example YAML to deploy a Db2uInstance for OLTP workloads:

apiVersion: argoproj.io/v1alpha1

kind: Application

metadata:

name: <db2uinstance\_name>

namespace: argocd

spec:

project: default

source:

repoURL: 'https://github.<ACME>.com/db2-operator.git'

path: 'charts/db2-instance'

targetRevision: v0.1

helm:

values: |

name: <db2uinstance\_name>

deployments:

- db2oltp

storageclassname:

rwx: <storageClass\_reference>

rwo: <storageClass\_reference>

image:

repository: icr.io/db2u

tag: 12.1.3.0

destination:

server: 'https://kubernetes.default.svc'

namespace: <db2uinstance\_namespace>

syncPolicy:

automated:

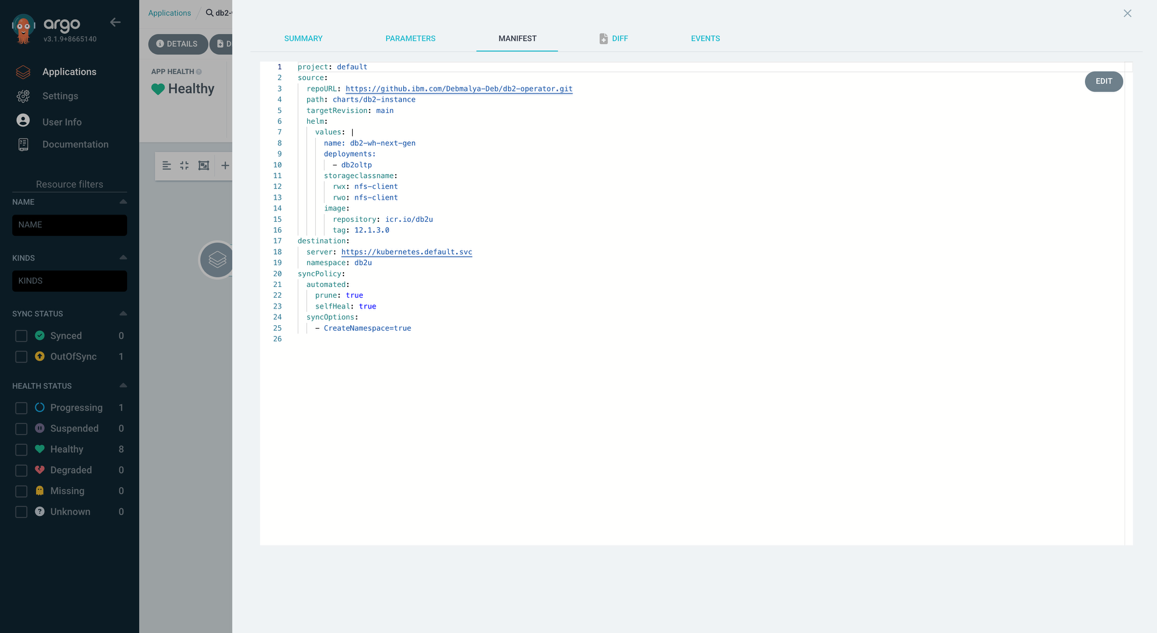
prune: true

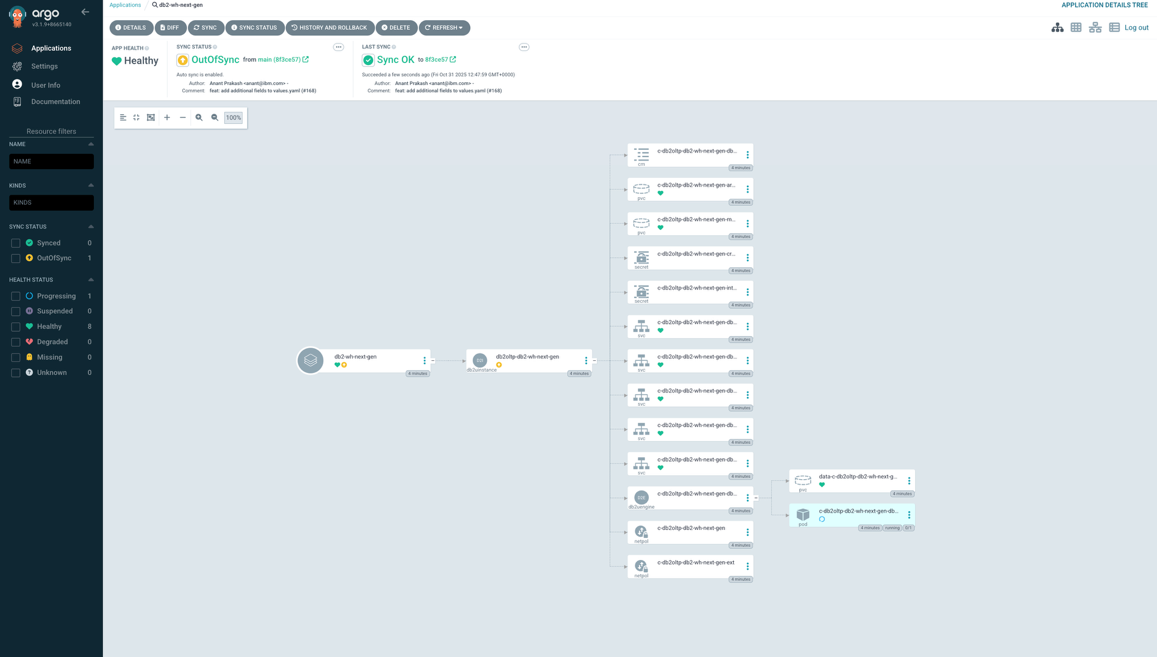
selfHeal: true

syncOptions:

- CreateNamespace=true

1. An example Argo CD UI view.

.



Notes

* This Argo CD Application uses Helm templating, rendering the templates defined in the chart at charts/db2-instance.
* The values block overrides the chart’s default configuration to tailor the deployment to your environment.
* Replace <db2uinstance\_name>, <db2uinstance\_namespace> and <storageClass\_reference> with values specific to your setup.
* To deploy Db2OLTP, please add the deployments value from db2oltp
* Once deployed, Argo CD will sync the chart and instantiate the Db2uInstance in your cluster, ready for use.

### Deploying DBWHMPP Db2uInstance Argo CD Applications

With the Db2 Operator deployed and synced via Argo CD, the next step is to deploy your Db2uInstance for transactional workloads (db2wh).

Steps to Deploy Db2uInstance via Argo CD UI

1. Log in to the Argo CD UI Navigating to your Argo CD dashboard using the LoadBalancer or Route URL.
2. Go to Applications
3. In the sidebar, click on Applications.
4. Create a New Application
5. Click the + New App button.
6. Switch to YAML Mode
7. Paste the Argo CD Application YAML

Use the following example YAML to deploy a Db2uInstance for DB2WHMPP workloads:

apiVersion: argoproj.io/v1alpha1

kind: Application

metadata:

name: <db2uinstance\_name>

namespace: argocd

spec:

project: default

source:

repoURL: 'https://github.acme.com/db2-operator.git'

path: 'charts/db2-instance'

targetRevision: v0.1

helm:

values: |

name: <db2uinstance\_name>

deployments:

- db2wh

storageclassname:

rwx: <storageClass\_reference>

rwo: <storageClass\_reference>

image:

repository: icr.io/db2u

tag: 12.1.3.0

destination:

server: 'https://kubernetes.default.svc'

namespace: <db2uinstance\_namespace>

syncPolicy:

automated:

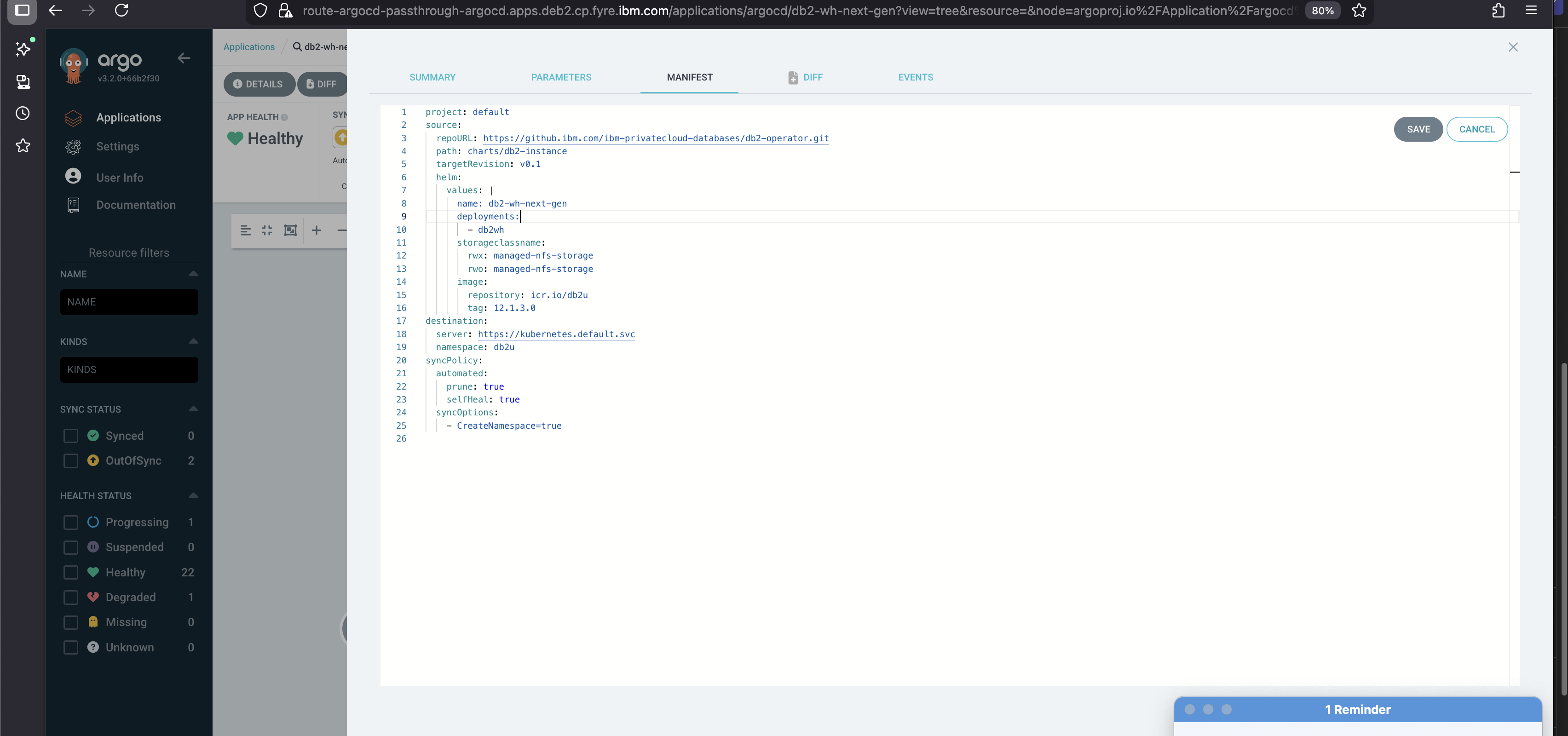
prune: true

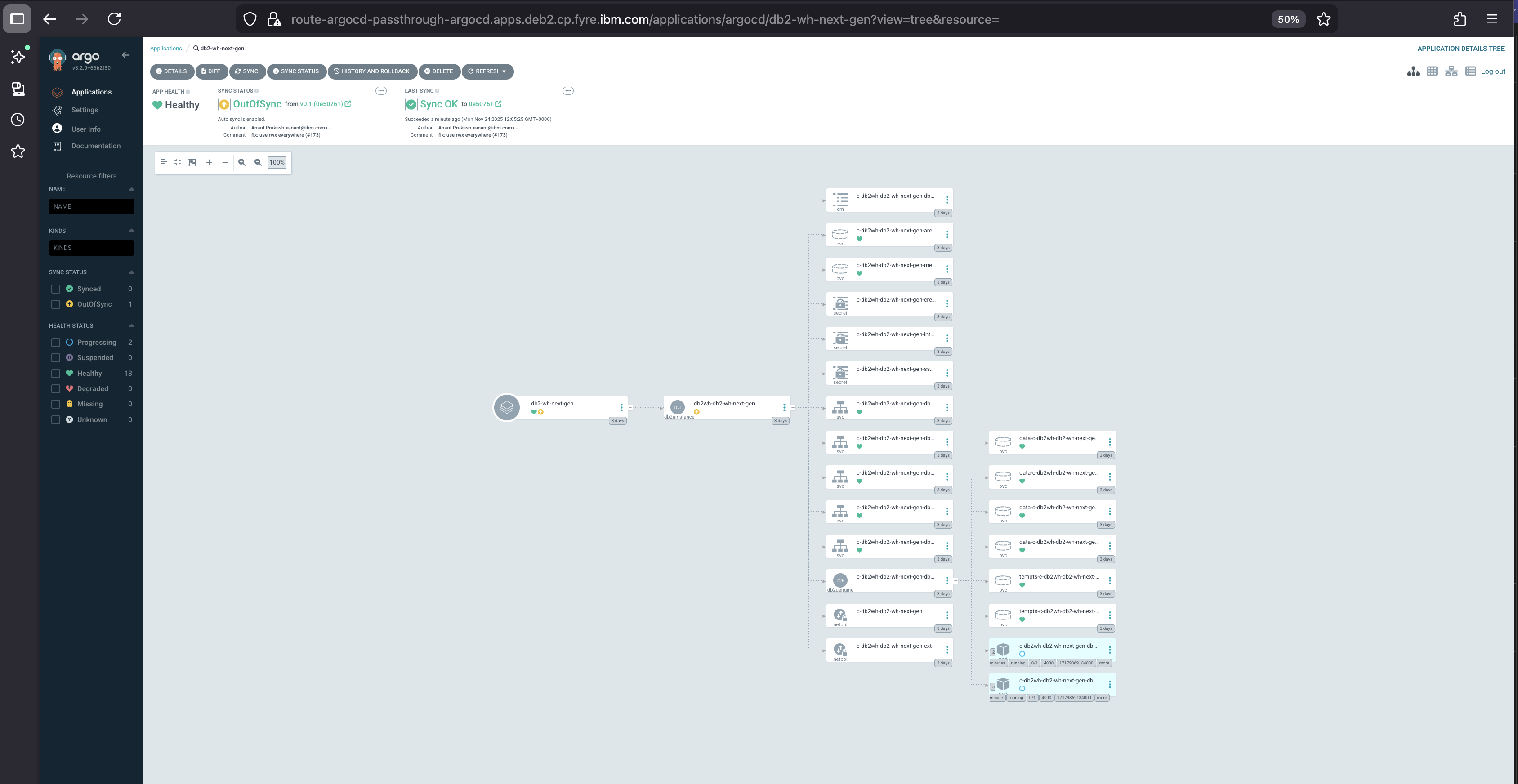
selfHeal: true

syncOptions:

- CreateNamespace=true

1. An example Argo CD UI view.





Notes

* This Argo CD Application uses Helm templating, rendering the templates defined in the chart at charts/db2-instance.
* The values block overrides the chart’s default configuration to tailor the deployment to your environment.
* Replace <db2uinstance\_name>, <db2uinstance\_namespace> and <storageClass\_reference> with values specific to your setup.
* To deploy Db2WHMPP, please add the deployments value from db2wh
* Once deployed, Argo CD will sync the chart and instantiate the Db2uInstance in your cluster, ready for use.

# Benefits of Argo CD Helm-Based Deployment for DB2 (Operator & Instances)

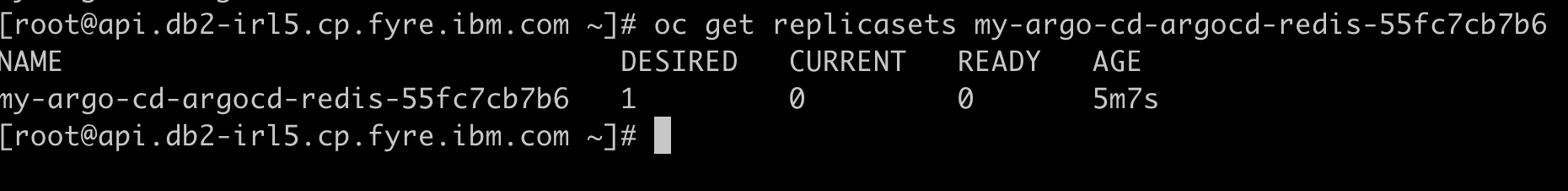
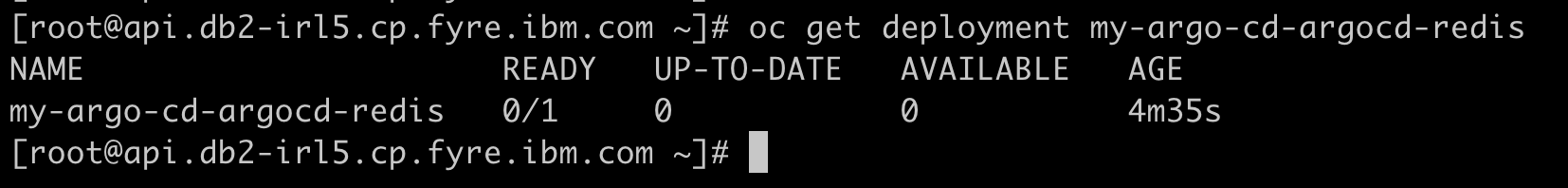
Using Argo CD to deploy the DB2 Operator and DB2uInstance via Helm charts offers powerful advantages for both visibility and control:

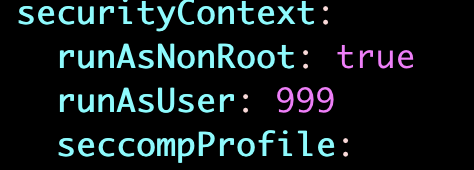
1. Real-Time Visibility: Argo CD provides a live dashboard of all Kubernetes resources created by your Helm charts—Deployments, Db2uengine, Services, PVCs, CRDs, and more. You can instantly see what’s running, what’s failing, and what’s out of sync.
2. Immediate Actionability: If something goes wrong like a pod crash or a failed sync you can take corrective action directly from the UI. Argo CD lets you resync, rollback, or even delete resources with a few clicks.
3. Automated Sync & Self-Healing: With automated sync and self-heal enabled, Argo CD continuously monitors your DB2 workloads and ensures they stay aligned with your Git source of truth.
4. Modular Helm Values: Helm makes it easy to customize DB2 deployments (e.g., platform type, image tags, namespaces) without editing raw manifests. Argo CD handles these values cleanly and tracks them as part of your GitOps flow.
5. Git-Driven Audit Trail: Every change to your DB2 infrastructure is version-controlled. You can trace deployments back to specific commits, making audits and rollbacks simple and reliable.

# FAQ: -

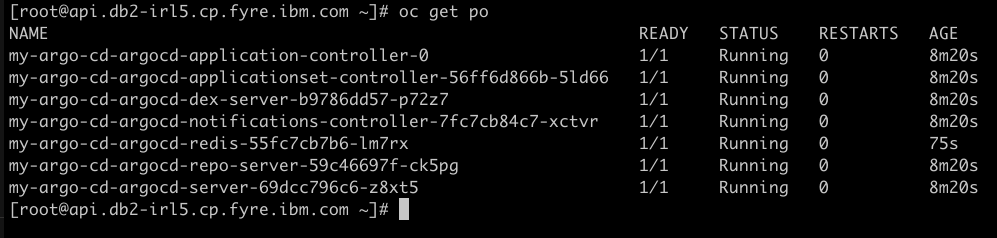
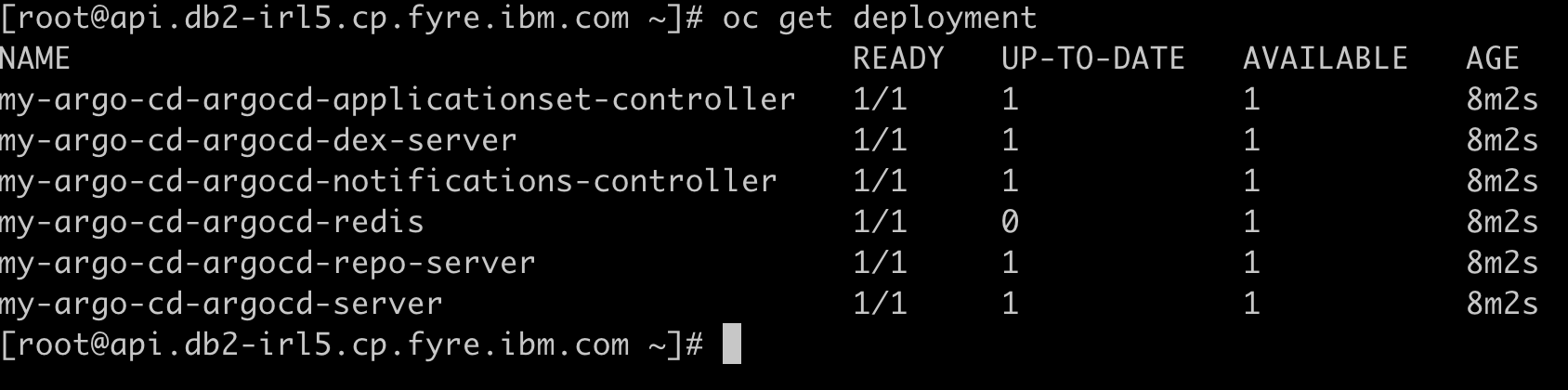
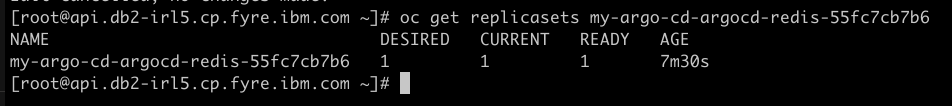
Why cannot I see Argo CD UI coming up on a OCP platform?

After argo cd helm installation command, you may see my-argo-cd-argocd-redis deployment is not available.

Please edit the my-argo-cd-argocd-redis replicaset and delete runAsUser: 999.



You will see then



# About the Authors

**Debmalya Deb** is a software engineer with in the Db2 containerization offering based in the Ireland lab. Deb has 4 years of working experience in a Software development team as a DevOps engineer with a proven track record in providing technical solutions, designing, implementing, and managing robust DevOps pipelines, infrastructure and cloud solutions. He can be reached at [Debmalya.Deb@ibm.com](mailto:Debmalya.Deb@ibm.com)

**Anant Prakash** a software engineer with in the Db2 containerization offering based in the Toronto lab. He is experienced in Software Development with a demonstrated history of working in the information technology and services industry. He is a strong business development professional with a Bachelor of Applied Science - BASc focused in Computer Science from McMaster University. He can be reached at [anant@ibm.com](mailto:anant@ibm.com)

**Janpreet Singh Chandhok** is a software engineer working in the Db2 containerization offering with special focus on hybrid cloud technologies. He can be reached at [janpreet.chandhok@ibm.com](mailto:janpreet.chandhok@ibm.com)

**Austin Clifford** is a Senior Technical Staff Member in Hybrid Data Management based in the Ireland Lab. He has worked with database, data lake and warehousing technologies for more than two decades. In 2012, Austin led the team to achieve a Guinness World Record for the Largest Data Warehouse, a record that IBM held for two years. Austin has authored numerous papers and patents, advises clients on data warehouse, analytics and containerization best practices and is a regular speaker at technical conferences. He can be reached at [acliffor@ie.ibm.com](mailto:acliffor@ie.ibm.com)