

# Migrating microservice oriented application to Kubernetes

With focus on best industry practises in CI/CD



# Agenda

- Introduction
  - example project
  - current CI/CD process
- Improving current CI/CD
- Migrating app to new CI/CD environment
- Demo with improved CI/CD

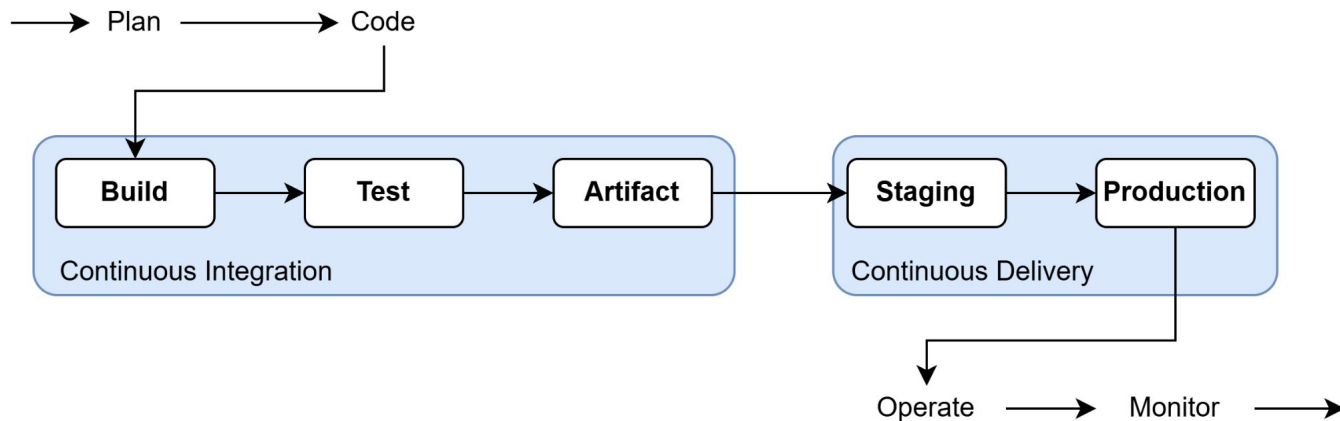


# NGINX

# Improve software development

- “CI/CD [is] a set of practices and tools designed to improve the software development process.” [1]

software development lifecycle:



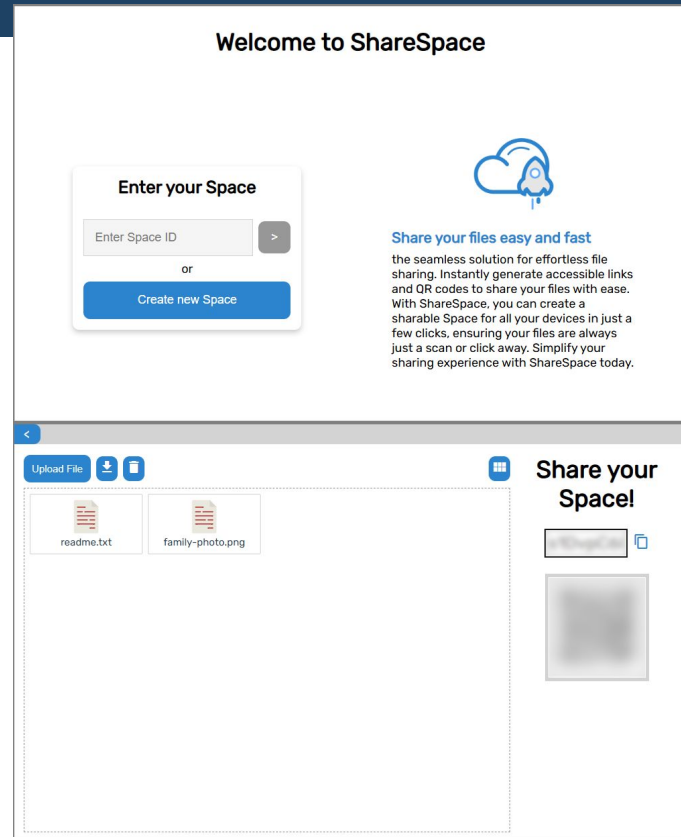
[1] <https://github.com/resources/articles/ci-cd>, accessed Nov 2025

Diagram inspired by GitHub's CI/CD documentation [1]

# Example Project: Fileshare

“Cloud-based application for easy file sharing”

- create and share spaces
- upload and manage files
- download files on any device
- easily share your space via QR code



# Fileshare app architecture

## 3 microservices

### Angular frontend

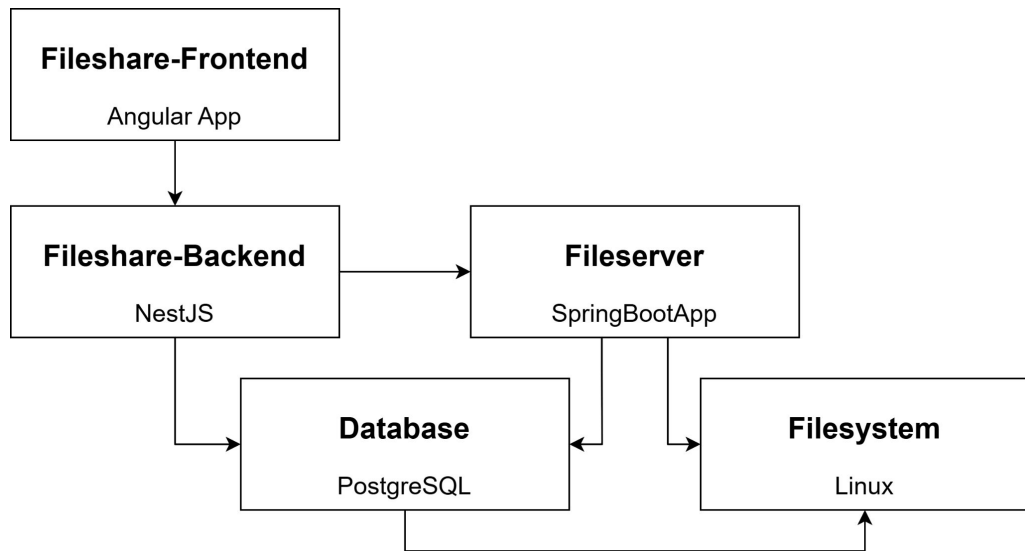
- end-user application

### NestJS backend

- manages spaces and what files are in which spaces

### Spring Boot fileserver

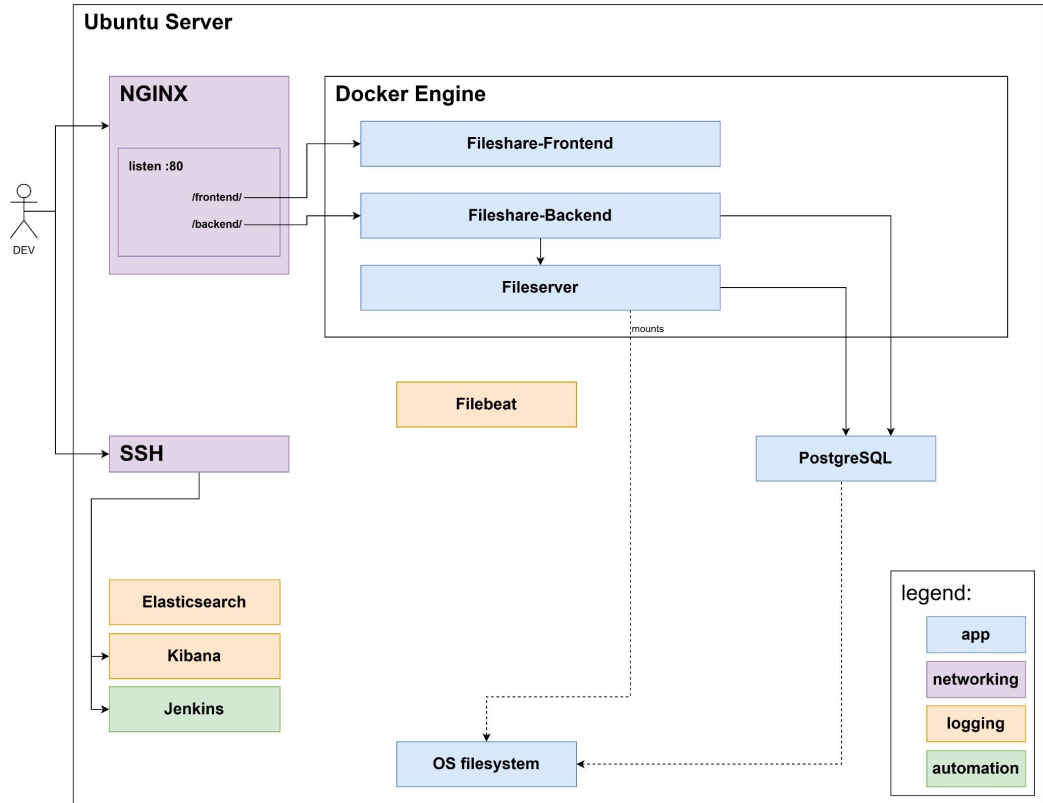
- serves as repository for files



# Complete Fileshare architecture

Architecture →

- microservices are containerized
- 1 git repository and 1 Jenkins pipeline for every microservice
- Jenkins deploys microservices in docker
- NGINX, PostgreSQL, Elastic Stack run natively



# Fileshare CI/CD process

1. new app feature is planned
2. write code
3. test changes locally
4. check into git and create pull request
5. merging pull request starts CI/CD pipeline
  - build docker image for microservice
  - stop running docker container
  - create and run new docker container
  - run docker networks for server-local communication
6. manually use and test deployed app

# Challenges in CI/CD

Addressing pain points and introducing industry solutions



# Rollout strategy problem

Pain point scenario:

- “We release a new feature. In Prod we notice it’s breaking our service. We need to roll back.”

Root cause:

- faulty test / misconfiguration

The problem:

- downtimes in production
- not safe rollout strategy
- no rollback functionality

# Rollout strategy solution

## Solution:

- reliable rollout process
- automated rollout with easy rollback

## Technology:

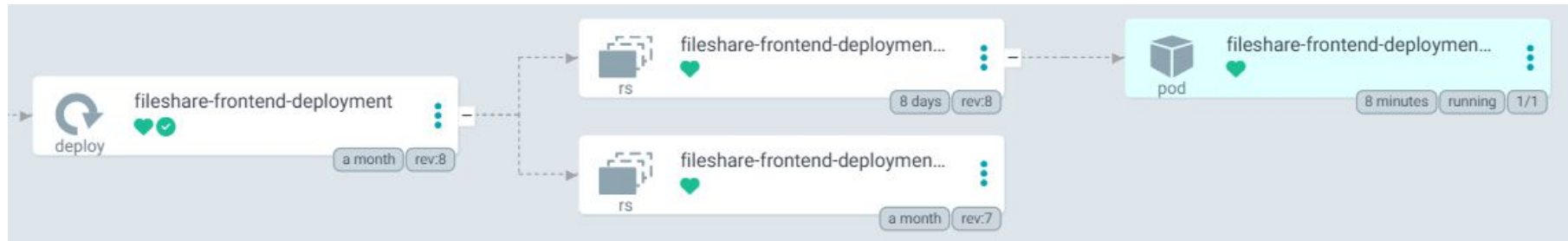


- kubernetes
- “Kubernetes is a [...] extensible [...] platform for managing containerized workloads and services” [2]
- use kubernetes deployments for app microservices
- let kubernetes deploy them

# Rollout strategy solution



- kubernetes resource type deployment
- safe rollout strategy
  - only deletes old pods when new ones are running and healthy
- deployment keeps old replica sets for easy rollbacks



# Spinning Up Environments problem

Pain point scenario:

- “For additional testing you need a new environment. A new deployment costs you one day of work.”

Root cause:

- deployment process is too complex
- difficult to see desired state and live state

The problem:

- the deployment process involves many manual steps
- troubleshooting infrastructure problems is difficult

# Spinning Up Environments solution

## Solution:

- declarative Configuration & orchestration
- Infrastructure as Code (IaC)

## Technology:



- Kubernetes
- define deployment with kubernetes yaml files
- live state is held in the kubernetes yaml files
- “IaC is managing and provisioning [...] infrastructure through code instead of manual processes.” [3]

# Configuration drift problem

Pain point scenario:

- “The memory limit of a service is 2gb but in the configuration file it is 512mb. Is it supposed to be 2gb or 512mb? You don’t know when or why it changed.”

Root cause:

- manual, undocumented changes to deployment

The problem:

- no single source of truth for desired state
- difficult to keep track of changes

# Configuration drift solution

## Solution: GitOps

- git repository to store configuration files, serves as desired state
- changes with git commits, git history serves as audit log
- automatically update live state to reach git state

## Technology:



- Argo CD
- “a declarative, GitOps continuous delivery tool for Kubernetes” [4]
- continuously compares live state to git state
  - marks resources out of sync
  - can automatically create, update, delete resources

# Migrating to an improved CI/CD

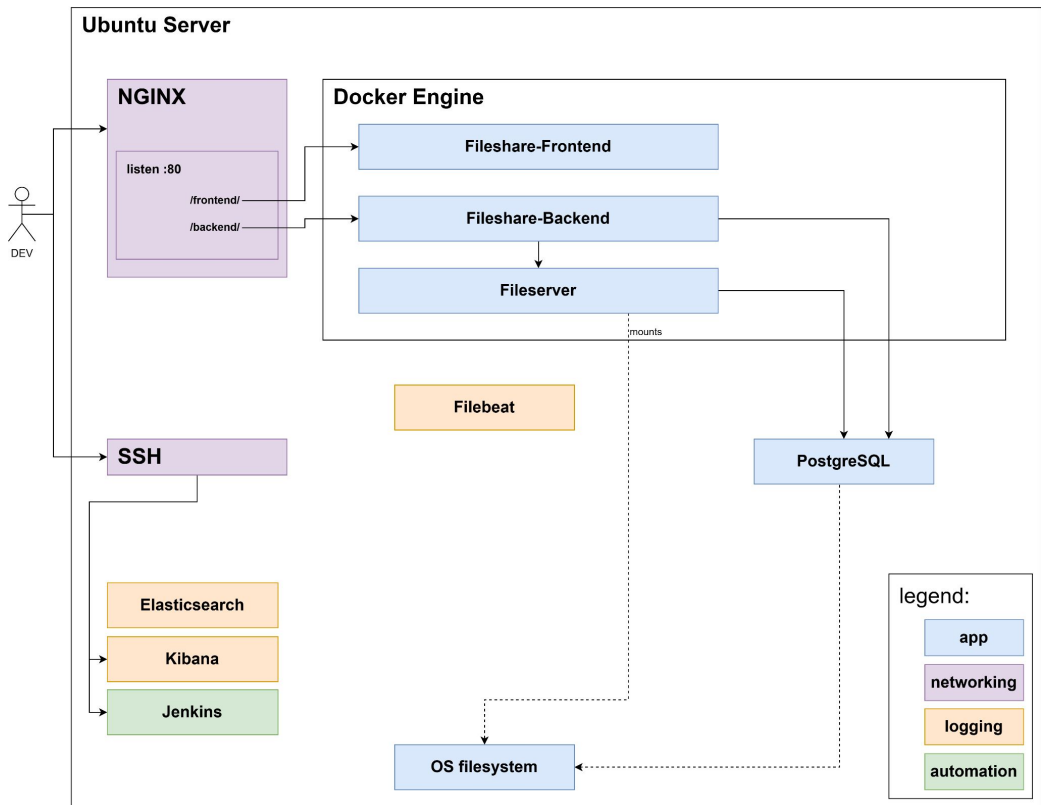
Preparing infrastructure and applying CI/CD industry practises



# Deployment before

Architecture →

- microservices are containerized
- 1 git repository and 1 Jenkins pipeline for every microservice
- Jenkins deploys microservices in docker
- NGINX, PostgreSQL, Elastic Stack run natively



# Migration plan

1. Deploy microservices in Kubernetes
  - ensures automated and reliable rollout
  - enables easy automated spin up of microservices
2. Implement GitOps with Argo CD
  - single source of truth
  - clear auditable change history
3. Add Secret Management with Sealed Secrets
  - add layer of security on our credentials
4. Adapt ELK to new CI/CD
  - restore logging functionality

# Migration plan

## 1. Deploy microservices in Kubernetes

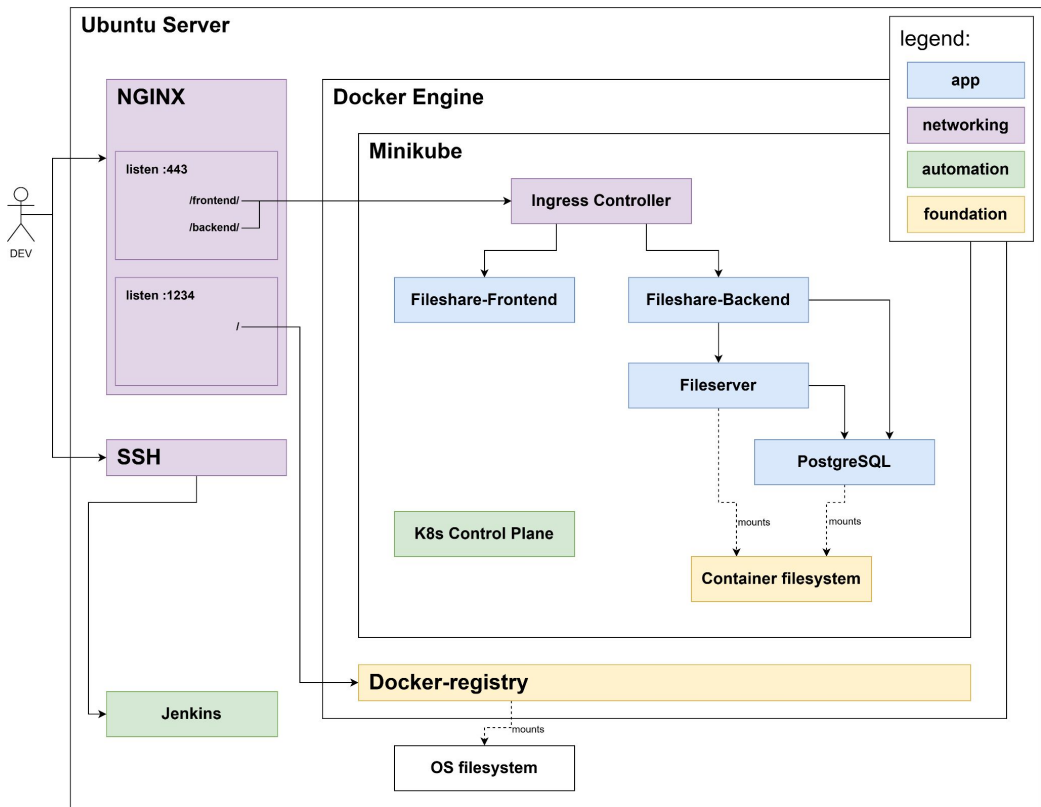
- ensures automated and reliable rollout
- enables easy automated spin up of microservices
- set up kubernetes environment
  - provision a kubernetes cluster
  - create kubernetes resources
- adapt ci/cd process
  - install a private docker registry
  - modify jenkins pipelines

# Deployment after step 1

Architecture →

differences:

- microservices deployed in kubernetes cluster
- traffic routed through ingress controller to microservices
- docker-registry providing app images



# Migration plan

## 2. Implement GitOps with Argo CD

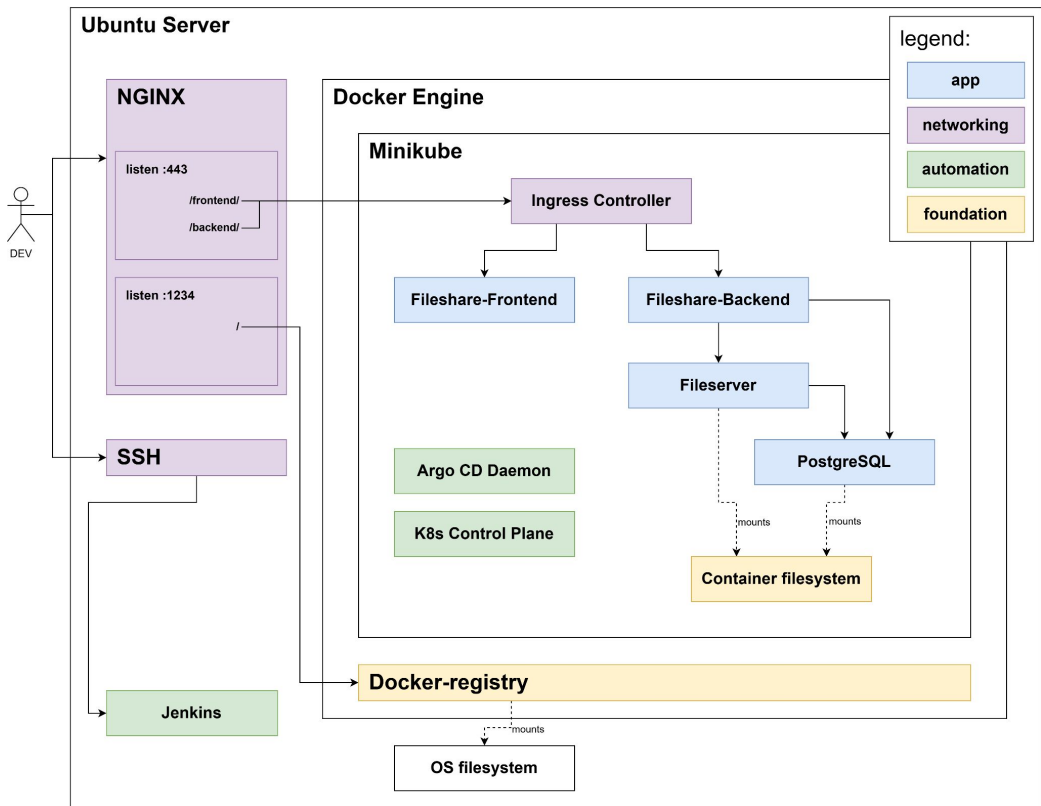
- single source of truth
- clear auditable change history
- set up Argo CD
  - deploy Argo CD in kubernetes cluster
  - create Argo CD resources
    - hold sync policies for other resources

# Deployment after step 2

Architecture →

differences:

- Argo CD deployed in kubernetes cluster

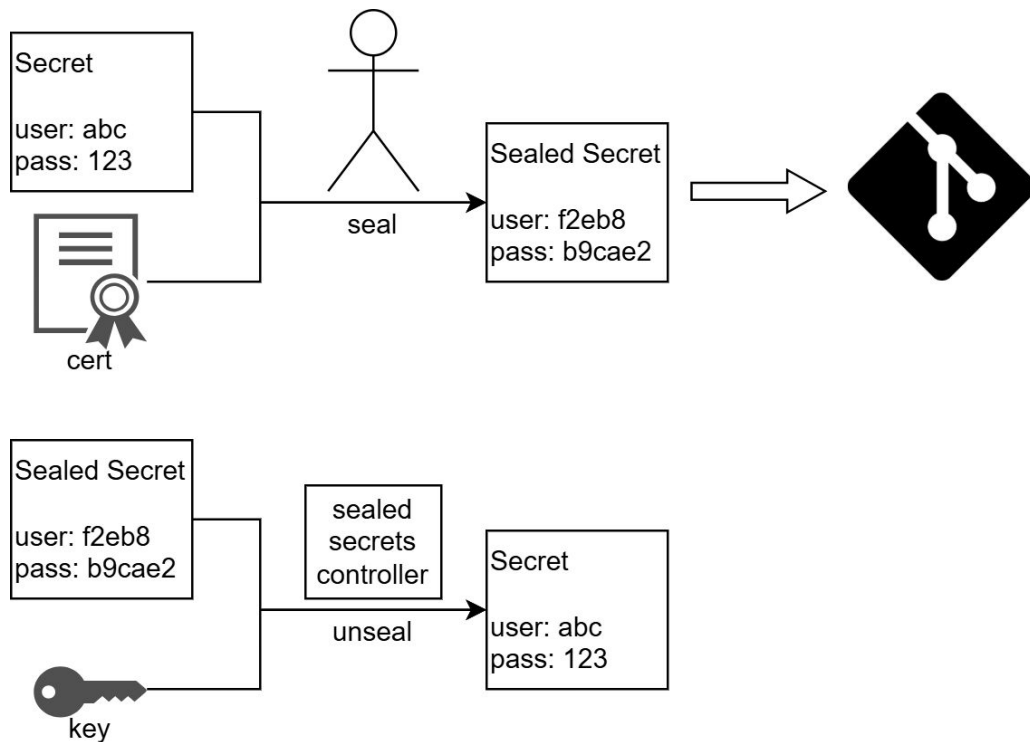


# Migration plan

## 3. Add Secret Management with Sealed Secrets

- add layer of security on our credentials
- core concept:
  - secrets are encrypted at rest
  - sealed secrets controller decrypts during runtime
- seal secrets and put in git repository
- deploy sealed secrets controller

# Sealed Secrets



core concept:

- secrets are encrypted at rest
- sealed secrets controller decrypts during runtime
- key needs to be handled manually

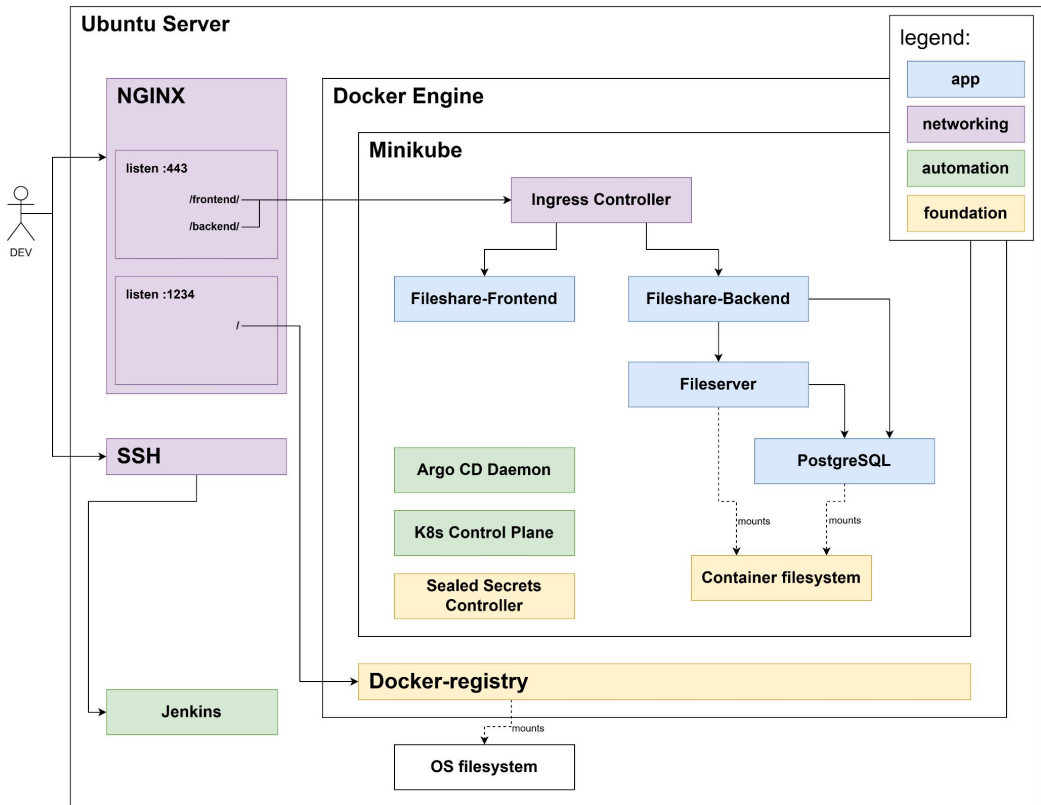


# Deployment after step 3

Architecture →

differences:

- Sealed Secrets Controller deployed in kubernetes cluster



# Migration plan

## 4. Adapt ELK to new CI/CD

- restore logging functionality
- add kubernetes metadata
  - deploy Filebeat inside kubernetes
  - adapt configuration

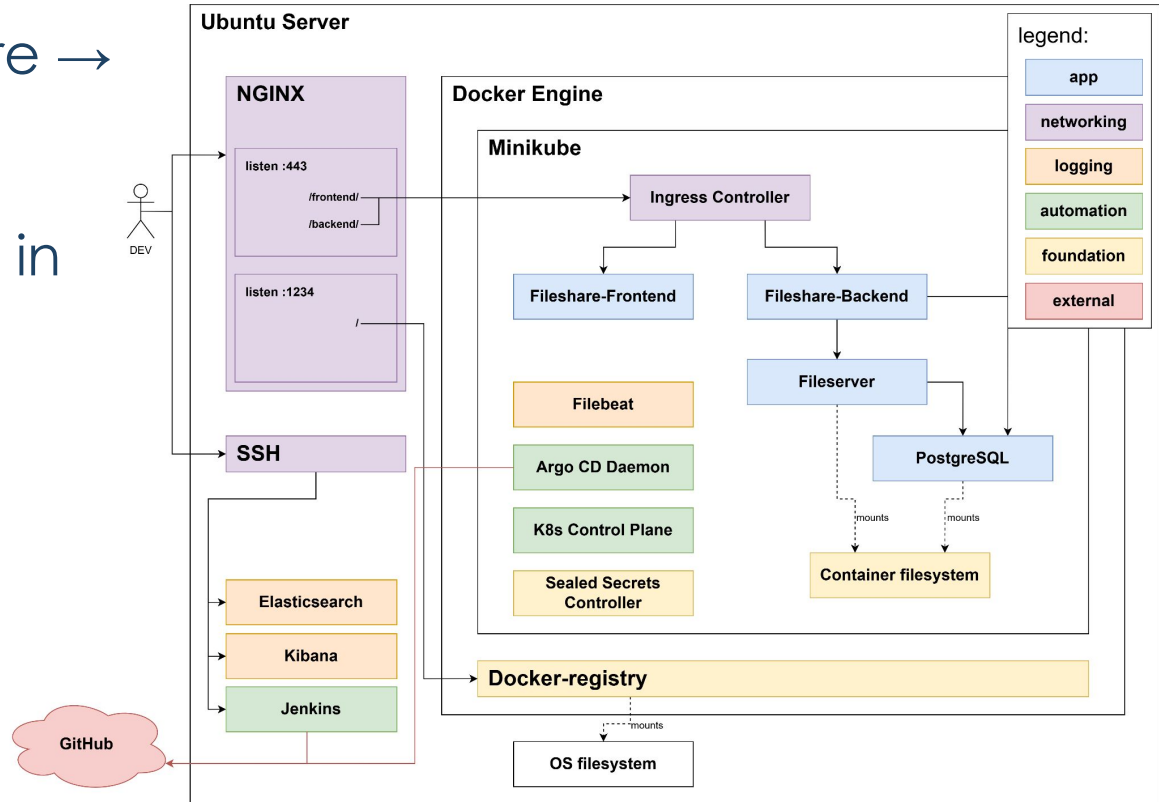


# Deployment after

Architecture →

Main differences:

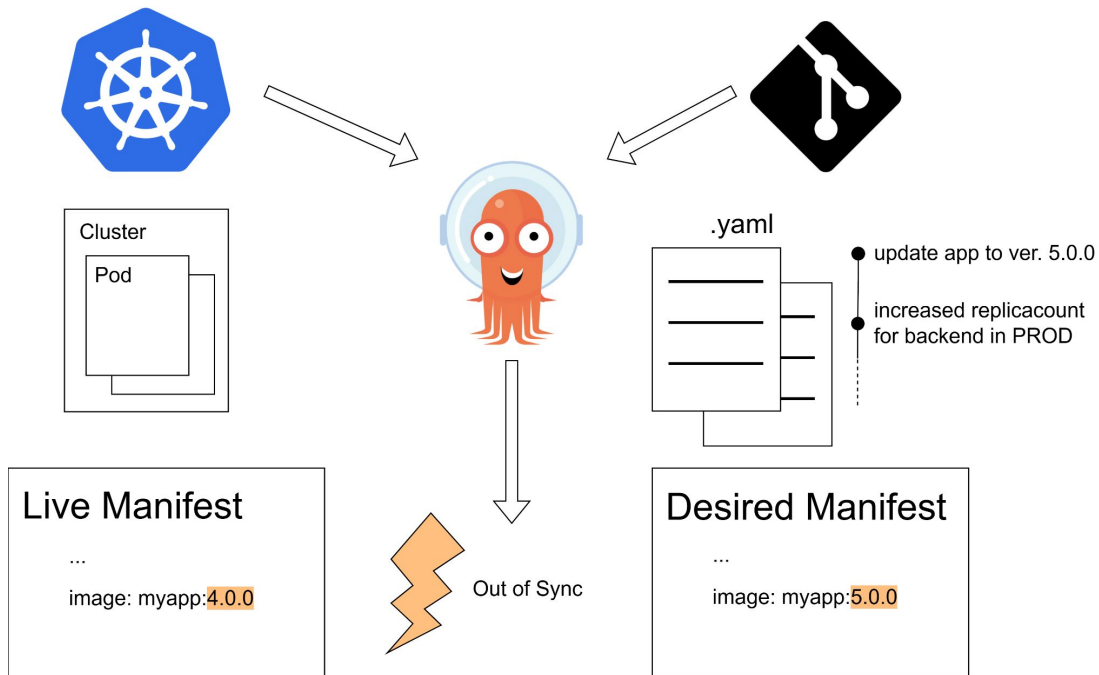
- microservices run in Kubernetes
- Argo CD deploys microservices
- PostgreSQL and Filebeat run in kubernetes



# Argo CD

## Sequence of events:

- cluster runs app image version 4.0.0
- commit with image tag update to version 5.0.0 pushed
- Argo CD compares git state to live state
- Argo CD marks resource as 'Out of Sync'
- (configurable) Argo CD updates live manifest to desired state



# Argo CD

## fileserver-application.yaml

```
1  apiVersion: argoproj.io/v1alpha1
2  kind: Application
3  metadata:
4    name: fileserver-argo-application
5    namespace: argocd
6  spec:
7    project: default
8
9    source:
10     repoURL: [REDACTED]
11     targetRevision: HEAD
12     path: base/apps/fileserver
13
14    destination:
15     server: https://kubernetes.default.svc
16     namespace: fileshare
17
18    syncPolicy:
19     syncOptions:
20       - CreateNamespace=true
21     automated: null
```

Argo is not allowed to sync automatically

## directory structure:

```
✓ GIT-ROOT
  ✓ base \ apps \ fileserver
    ✓ templates
      ≡ deployment.yaml
      ≡ pvc.yaml
      ! kustomization.yaml
```

## Argo application

- has a source, a collection of files containing kubernetes resources
- has a destination, a cluster and namespace

## Dashboard:

→

Applications / Q fileserver-argo-application

APP DETAILS

APP DIFF

SYNC

SYNC STATUS

HISTORY AND ROLLBACK

DELETE

REFRESH

APP HEALTH

Healthy

SYNC STATUS

OutOfSync

from HEAD (eed0e64)

Auto sync is not enabled.

Author: Lennart Ruck <lennart.ruck@powmio.com> -

Comment: Revert "Upgraded App version"

LAST SYNC

Sync OK

to 3939c1f

Succeeded a month ago (Wed Oct 15 2025 17:02:11 GMT+0200)

Author: Lennart Ruck <lennart.ruck@powmio.com> -

Comment: Expanded documentation

APPLICATION DETAILS TREE

Log out

fileserver-argo-application

2 months

fileserver-volume-claim

pvc

2 months

fileserver-service

svc

2 months

fileserver-deployment

deploy

2 months

rev:4

fileserver-service

ep

2 months

fileserver-service-4nwjv

endpointslice

2 months

fileserver-deployment-bfb6c6...

rs

a few seconds

rev:4

fileserver-deployment-bfb6c6...

pod

a few seconds

running

1/1

Screenshot of Argo CD UI for demonstration purposes.

© PowMio GmbH

31

# CI/CD process after

1. a new feature is planned
2. developer writes code
3. developer tests changes locally
4. check into source code repo and create pull request
5. merging pull request starts CI/CD pipeline
  - builds docker image for microservice
  - pushes docker image to docker registry
6. update image version in kubernetes manifest in GitOps repo
7. check changes into GitOps repo and create pull request
8. merging pull request makes Argo CD deploy updated version
9. developer verifies correct deployment and uses App



# Hands-on with improved CI/CD

Steps:

1. Change background color of Angular App
2. Prepare commit, push and merge to master
3. Write the new image tag into gitOps repository
4. Prepare commit, push and merge to master
5. Sync Angular deployment in Argo CD Web UI

# Fileshare before

## Welcome to ShareSpace

### Enter your Space

>

or

Create new Space



### Share your files easy and fast

the seamless solution for effortless file sharing. Instantly generate accessible links and QR codes to share your files with ease. With ShareSpace, you can create a sharable Space for all your devices in just a few clicks, ensuring your files are always just a scan or click away. Simplify your sharing experience with ShareSpace today.

# Fileshare Frontend before

```
# styles.css > body
src > # styles.css > body
1 /* You can use global styles in this file, and also import other style files */
2 @import url('https://fonts.googleapis.com/css2?family=Rubik:wght@400;500&family=Open+Sans:wght@400;500&display=block');
3
4 {
5   margin: 0;
6   padding: 0;
7   border: 0;
8 }
9
10 body {
11   font-family: 'Rubik', 'Open Sans', sans-serif;
12   font-size: var(--font-normal);
13   height: 100vh;
14   background-color: #rgb(255, 255, 255);
15 }
16
17 :root {
18   --color-primary: #000000;
19   --color-primary-highlight: #000000;
20   --color-primary-highlight-strong: #000000;
21   --color-on-primary: #ffffff;
22
23   --color-primary-variant: #000000;
24   --color-primary-variant-highlight: #000000;
25   --color-primary-variant-highlight-strong: #000000;
26   --color-on-primary-variant: #ffffff;
27
28   --color-secondary: #ff0000;
29   --color-on-secondary: #ffffff;
30
31   --color-background: #ffffff;
32   --color-on-background-strong: #000000;
33   --color-on-background-soft: #000000;
34
35   --color-background-variant: #rgb(248, 248, 248);
36
37   --color-surface: #ffffff;
38   --color-surface-highlight: #ffffff;
39   --color-on-surface-strong: #000000;
40   --color-on-surface-soft: #rgb(24, 24, 24);
41 }
```

# Fileshare Frontend after

```
# styles.css X
src > # styles.css > body
1 /* You can use global styles to this file, and also import other style files */
2 @import url('https://fonts.googleapis.com/css2?family=Rubik:wght@400;700&family=Open+Sans:wght@400;700&display=block');
3
4 {
5   margin: 0;
6   padding: 0;
7   border: 0;
8 }
9
10 body {
11   font-family: 'Rubik', 'Open Sans', sans-serif;
12   font-size: var(--font-normal);
13   height: 100vh;
14   background-color: rgb(255, 0, 0);
15 }
16
17 :root {
18   --color-primary: #000000;
19   --color-primary-highlight: #000000;
20   --color-primary-highlight-strong: #000000;
21   --color-on-primary: #ffffff;
22
23   --color-primary-variant: #000000;
24   --color-primary-variant-highlight: #000000;
25   --color-primary-variant-highlight-strong: #000000;
26   --color-on-primary-variant: #ffffff;
27
28   --color-secondary: #ff0000;
29   --color-on-secondary: #ffffff;
30
31   --color-background: #ffffff;
32   --color-on-background-strong: #000000;
33   --color-on-background-soft: #000000;
34
35   --color-background-variant: #fff(240, 240, 240);
36
37   --color-surface: #lightgray;
38   --color-surface-highlight: #white;
39   --color-on-surface-strong: #black;
40   --color-on-surface-soft: #rgb(38, 38, 38);
41 }
```

# Committed Frontend changes

The screenshot shows the Sourcetree application interface. The top menu bar includes File, Edit, View, Repository, Actions, Tools, and Help. Below the menu is a toolbar with icons for Commit, Pull, Push, Fetch, Branch, Merge, Stash, Discard, and Tag. The left sidebar contains sections for WORKSPACE (File Status, History, Search), BRANCHES (kubernetes-migration-experimenta), TAGS, REMOTES, and STASHES. The main area displays a commit history table with columns for Graph, Description, Date, Author Name, and Commit. The selected commit is c1cd005, titled "Changed background color to red (for demo purposes)", by Lennart Ruck on 24 Nov 2025. Below the table, the commit details are shown, including the commit hash, author, date, and committer. The diff view for src/styles.css is displayed, showing changes to the background-color property. The diff highlights the change from rgb(255, 255, 255) to rgb(255, 0, 0).

Graph	Description	Date	Author Name	Commit
•	Changed background color to red (for demo purposes)	24 Nov 2025 15:13	Lennart Ruck <lennart.ruck@powmio.com>	c1cd005

**Commit:** c1cd005ea252eba07189d8a767cba31e82d5f4e [c1cd005]  
**Author:** Lennart Ruck <lennart.ruck@powmio.com>  
**Date:** Monday, 24 November 2025 15:13:15  
**Committer:** Lennart Ruck

Changed background color to red (for demo purposes)

src/styles.css

```
11 11 .....font-family: 'Rubik', 'Open Sans', sans-serif;  
12 12 .....font-size: var(--font-normal);  
13 13 .....height: 100px;  
14 14 .....background-color: rgb(255, 255, 255);  
15 15 .....background-color: rgb(255, 0, 0);  
16 16 .....  
17 17 .....root {
```

# Jenkins running Frontend pipeline

**Jenkins**  🔔 🛡️ 👤 jenkins\_user 🚪 Abmelden

Dashboard > PowMio\_Fileshare\_Frontend\_Kubernetes > #14 > Pipeline Console

✅ < Build #14 ▶ Rebuild 🔗 Overview ⚙️ Configure ⋮

Erfolgreich 6 Minuten 34 Sekunden ago in 4 Minuten 38 Sekunden

- ✅ Checkout SCM
- ✅ Build Docker Image
- ✅ Push Docker Image

**Stage 'Push Docker Image'**

🕒 Started 1 Minute 56 Sekunden ago  
⌚ Queued 0 ms  
⏱️ Took 0.9 Sekunden  
✅ Success  
[View as plain text](#)

- ✅ **Checks if running on a Unix-like node** 6 ms 🔗 🛡️ ⌵
- ✅ **docker tag "\$JD\_ID" "\$JD\_TAGGED\_IMAGE\_NAME"** 0.26 Sekunden 🔗 🛡️ ⌵  
Shell Script
- ✅ **Checks if running on a Unix-like node** 5 ms 🔗 🛡️ ⌵
- ✅ **docker push "\$JD\_TAGGED\_IMAGE\_NAME"** 0.26 Sekunden 🔗 🛡️ ⬆  
Shell Script

```

0  + docker push [REDACTED]:c1cd00655ea252eba07189d8a767cba31e82d5f4e
1  The push refers to repository [REDACTED]
2  a8a97fd1e9d: Preparing
3  b70075175591: Preparing
4  38d44e06fd01: Preparing
5  388bb4cadb9e: Preparing
6  5f0d4d15245b: Preparing
7  fe0771a36433: Preparing
8  1e79db1a7c1e: Preparing
  
```

Jenkins 2.462.2

```

$ curl -u $(cat creds.txt) https://[REDACTED]/tags/list
{"name": "[REDACTED]", "tags": ["1.0.1", "33f32e1305ab32ff510babe47f78397354a29945", "1aea8bf
aa32deb060c159b6c344c0068ff6ed28ac", "fa118d38a5f7cc3718f156edaf79b6cd30261d21", "d625ac429fec632895f9ae2b79a3dc5e7505e
b67", "c1cd00655ea252eba07189d8a767cba31e82d5f4e", "1.0.0", "a5f1fe7937fb96c4a8757b0ac2210271ece8977e", "9cebd8ecb61176a
f3d8271d9b93ce8da390b72e", "8fe035267ecb49b23e527bfed4d15a7d5b0ac9d1", "f42646a400e5a5b7ede58a56a798c9161e27131a", "1.0
.1.prod", "de9cb504cf82d42edabd1b79353152a932930998"]}
  
```

# Updating App version in Deployment

Before:

```
! kustomization.yaml X
overlays > prod > apps > fileshare-frontend > ! kustomization.yaml > {} 0 > newTag
kustomization.yaml - Kubernetes native configuration management (kustomization.json)
1  apiVersion: kustomize.config.k8s.io/v1beta1
2  kind: Kustomization
3  resources:
4  - ../../../../base/apps/fileshare-frontend
5
6  images:
7  - name:
8    newTag: 8fe035267ecb49623e527bfd4d15a7d5b0ac9d1
9
10 patches:
11 - path: appconfig-configmap.yaml
12   target:
13     kind: ConfigMap
14     name: appconfig-json
15
```

After:

```
! kustomization.yaml M X
overlays > prod > apps > fileshare-frontend > ! kustomization.yaml > {} 0 > newTag
kustomization.yaml - Kubernetes native configuration management (kustomization.json)
1  apiVersion: kustomize.config.k8s.io/v1beta1
2  kind: Kustomization
3  resources:
4  - ../../../../base/apps/fileshare-frontend
5
6  images:
7  - name:
8    newTag: c1cd0055ea252eba07189d8a767cbe31e82d5f4e
9
10 patches:
11 - path: appconfig-configmap.yaml
12   target:
13     kind: ConfigMap
14     name: appconfig-json
15
```

# Argo CD shows out of sync resource

The screenshot displays the Argo CD web interface for an application named 'fileshare-frontend-argo-application'. The top navigation bar includes tabs for APP DETAILS, APP DIFF, SYNC, SYNC STATUS, HISTORY AND ROLLBACK, DELETE, and REFRESH. The 'SYNC STATUS' tab is active, showing a red 'OutOfSync' status with a warning icon and the message 'from HEAD (4830cd1)'. Below this, it states 'Auto sync is not enabled.' and provides author and comment information. The 'LAST SYNC' tab shows a green 'Sync OK' status with a checkmark icon, indicating a successful sync 4 days ago. The main area displays a dependency graph of the application's resources. The graph starts with 'fileshare-frontend-argo-app' (2 months) and branches into 'appconfig-json' (2 months), 'fileshare-frontend-service' (2 months), 'fileshare-frontend-deployment' (2 months, rev:11), and 'fileshare-frontend-ingress' (2 months). 'fileshare-frontend-service' further branches into 'fileshare-frontend-service' (2 months) and 'fileshare-frontend-service-85c...' (2 months). 'fileshare-frontend-deployment' branches into 'fileshare-frontend-deployment...' (2 months, rev:11) and 'fileshare-frontend-deployment...' (4 days, rev:10). The final resource in the graph is 'fileshare-frontend-deployment...' (4 days, running, 1/1), which is highlighted with a red border, indicating it is the resource that is out of sync.



# Argo CD 'App Diff'

The screenshot displays the Argo CD 'App Diff' interface. On the left, a sidebar shows the application 'fileshare-frontend-argo-app' with a 'Healthy' status. The main panel is titled 'apps/Deployment/fileshare/fileshare-frontend-deployment' and shows a diff between two versions of the deployment manifest. The diff is presented in a compact format, with line numbers 147 through 151 visible. The left side of the diff shows the previous version with a red background, and the right side shows the current version with a green background. The diff highlights changes in the 'image' field, where the previous version used 'fe035267ecb49623e527bfed4d15a7d5b0ac9d1' and the current version uses '1cd0055ea252eba07189d8a767cba31e82d5f4e'. The 'imagePullPolicy' and 'name' fields are identical in both versions.

Applications / Q fileshare-frontend-argo-app

APP DETAILS APP DIFF

APP HEALTH **Healthy**

SYNC STATUS **OutOfSync**

Auto sync is not enable  
Author: Lennart  
Comment: Upgrade

100%

fileshare-frontend-argo-app

SUMMARY PARAMETERS MANIFEST **DIFF** EVENTS

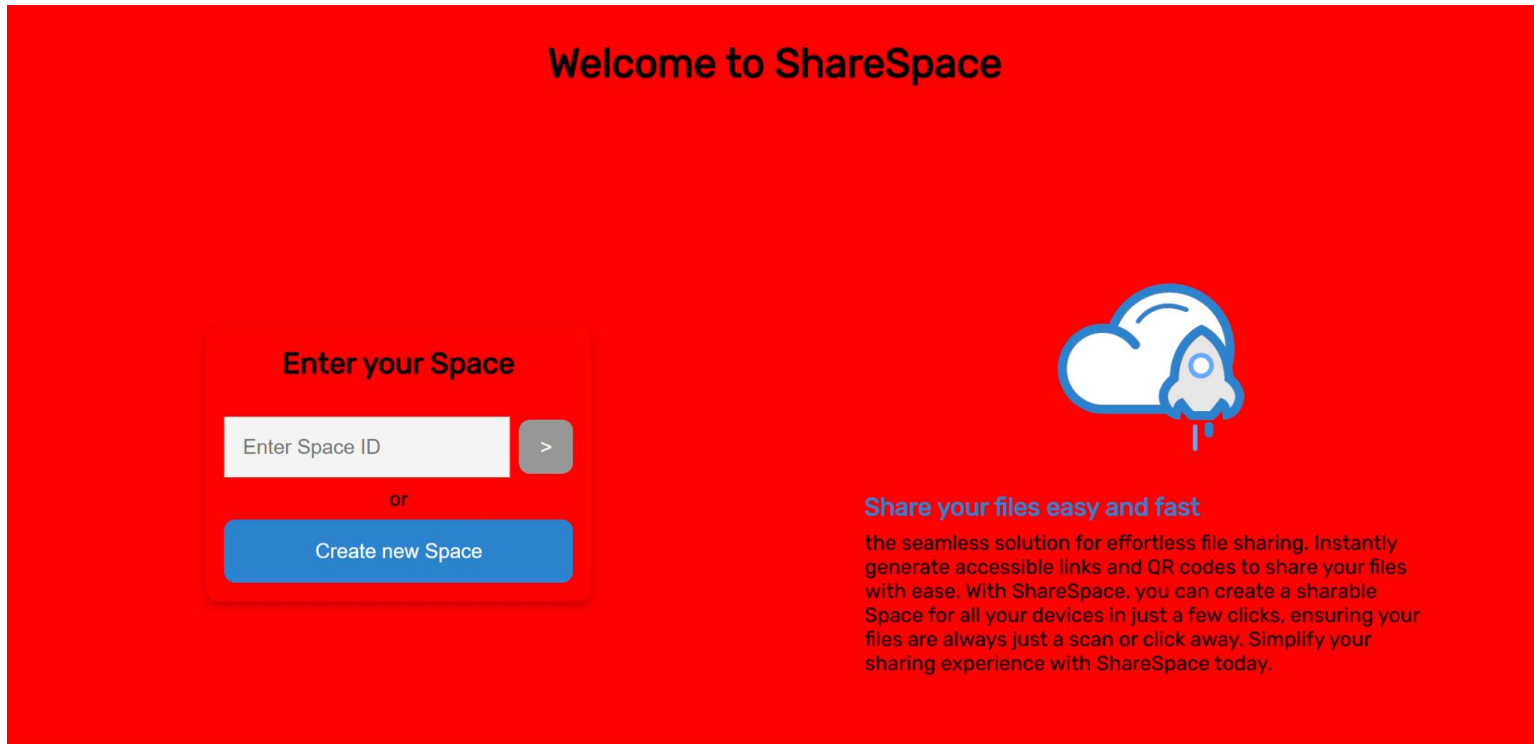
☒ Compact diff ☐ Inline diff

apps/Deployment/fileshare/fileshare-frontend-deployment

```
147     containers:
148     - image: >-
149       fe035267ecb49623e527bfed4d15a7d5b0ac9d1
150     imagePullPolicy: IfNotPresent
151     name: fileshare-frontend
```

```
147     containers:
148     - image: >-
149       1cd0055ea252eba07189d8a767cba31e82d5f4e
150     imagePullPolicy: IfNotPresent
151     name: fileshare-frontend
```

# Fileshare after pressing Sync in Argo CD



# Thank you!

PowMio

Alter Schlachthof 33  
76131 Karlsruhe

[www.powmio.com](http://www.powmio.com)



[contact@powmio.com](mailto:contact@powmio.com)