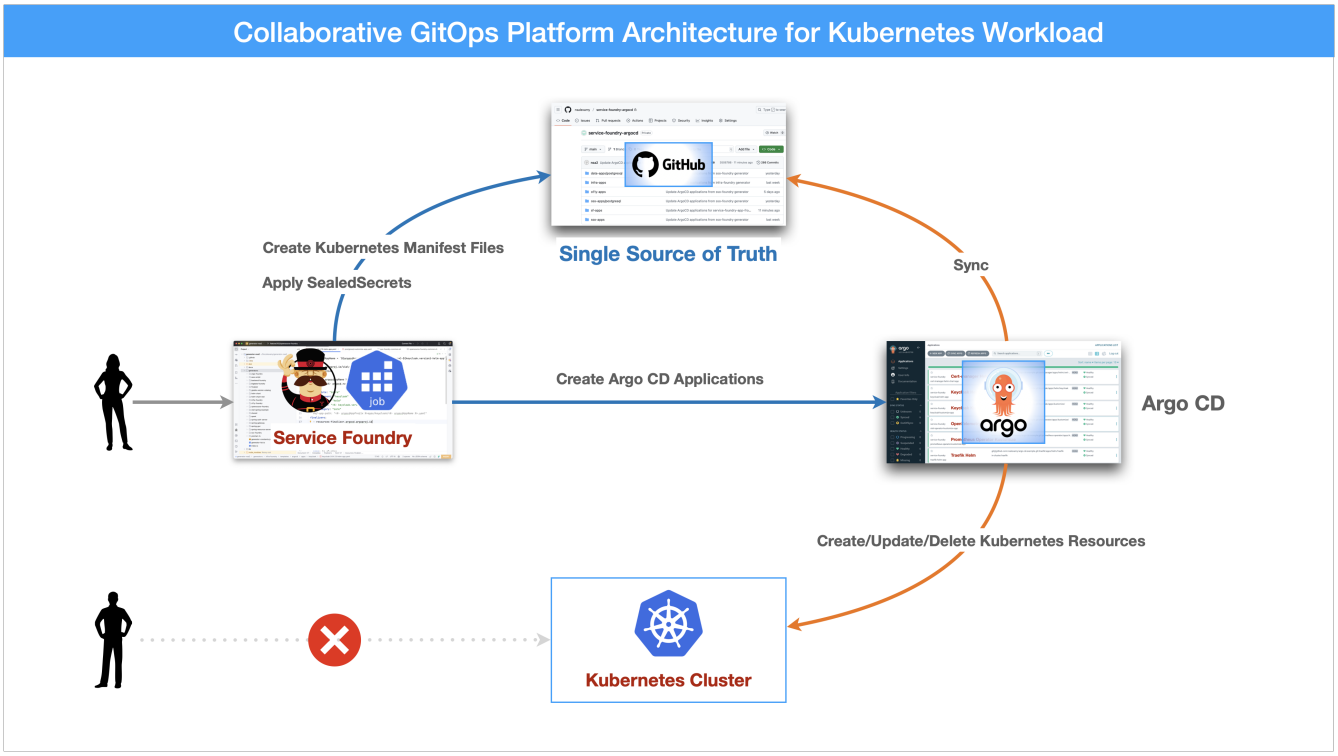
Service Foundry: User Guide

Young Gyu Kim <credemol@gmail.com>



> Helping you manage your Kubernetes apps with ease - no deep tech knowledge needed!

> End-to-End GitOps Orchestration for Kubernetes Workloads

> A Modular GitOps Framework for Kubernetes-Native Platform Engineering

What Is Service Foundry?

Service Foundry is a Kubernetes-native platform engineering framework designed to help teams **provision, deploy, secure, and observe workloads** using GitOps workflows.

It integrates popular open-source tools into a cohesive automation system, enabling you to:

- Use standardized GitOps patterns to manage infrastructure and workloads
- Collaborate through version-controlled configuration
- Deploy complete application stacks (e.g., observability, identity, data pipelines)
- Secure resources using Single Sign-On (SSO) and encrypted secrets

It bridges the gap between developers and operations by providing a **UI-driven GitOps console**, while preserving the **auditable**, **declarative** nature of Git-based infrastructure as code.

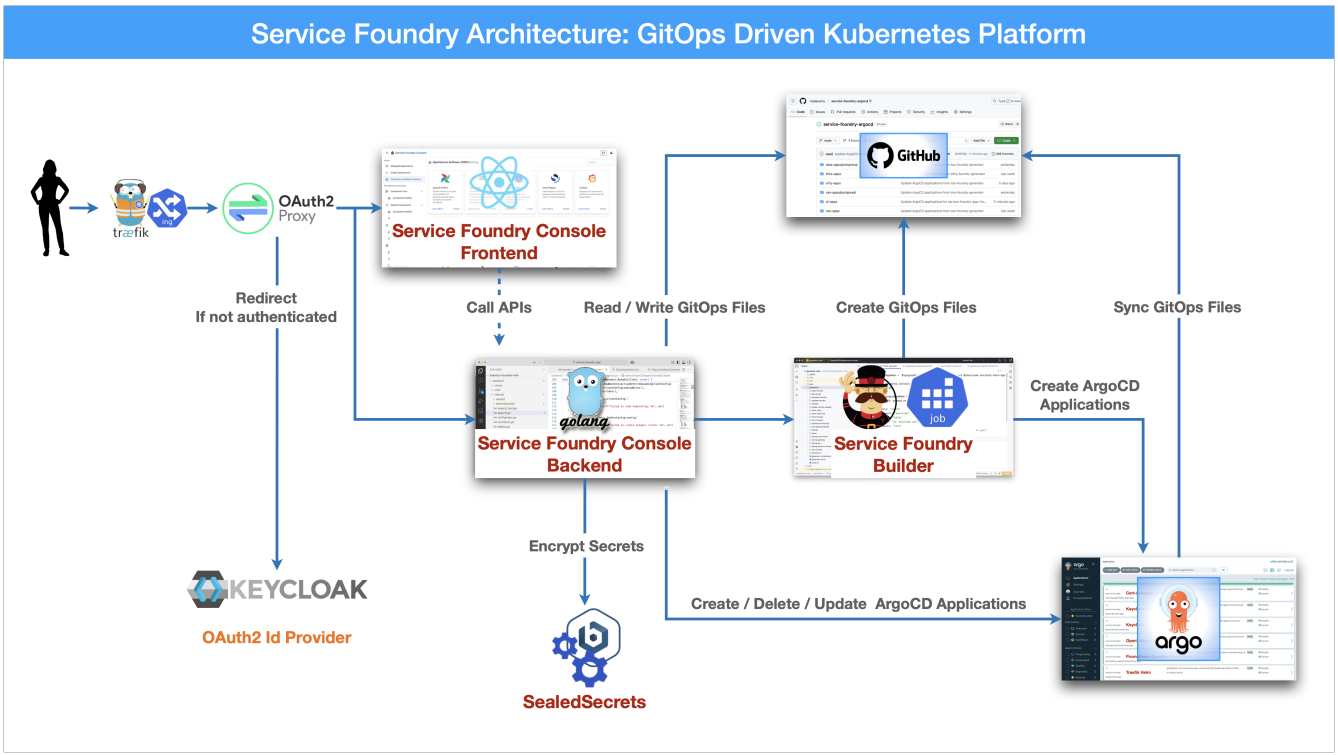


Figure 1. GitDps-Driven Kubernetes Platform

GitOps-Driven Platform Management

Service Foundry adopts a GitOps-first approach, meaning:

- All Kubernetes resources (infra modules, apps, secrets, etc.) are declared in YAML and stored in Git
- Changes are reviewed via Pull Requests and automatically applied by Argo CD
- The entire state of the cluster is version-controlled, auditable, and reproducible

This enables:

- Predictable deployments — Git is the single source of truth
- Safe rollbacks — Easily revert to previous versions
- Transparent ops — Operational changes are traceable via Git history

Core Components of Service Foundry

Table 1. Service Foundry Components

Component	Purpose
Argo CD	Reconciles Kubernetes state with Git
SealedSecrets	Encrypts secrets for safe Git storage
Keycloak	Acts as the OAuth2 identity provider (SSO)
Traefik	Kubernetes-native ingress and routing
Oauth2 Proxy	Authentication middleware for UIs
Service Foundry Console	Web UI to generate, review, and manage GitOps manifests
Service Foundry Builder	CLI/Backend tool to generate manifests and bootstrap environments
Source Generator	Generates Kubernetes manifests from templates

Why Use Service Foundry?

- **Declarative + Automated**: All infrastructure and app configurations are versioned and applied automatically
- **Built-in Security**: Seamless OAuth2-based login, secret encryption, and role-based access
- **Production-Grade Observability**: OpenTelemetry, Prometheus, Grafana, and Loki are pre-integrated
- **Unified Management Interface**: Web console for all configuration and lifecycle tasks
- **Modular Architecture**: Deploy only the modules you need — SSO, Observability, Big Data, etc.
- **Developer & Platform Collaboration**: Developers install services; operators manage infrastructure — all within GitOps

Rapid Bootstrap with a Single Command

Service Foundry provides a CLI tool and Helm charts to **bootstrap your platform in one step**:

- Installs Argo CD, Keycloak, Traefik, SealedSecrets, and Service Foundry Console
- Sets up initial Git repository structure
- Deploys infrastructure modules and app scaffolds using GitOps
- Secures access with SSO

Installation Command using Helm

```
$ helm install service-foundry-builder \
  service-foundry/service-foundry-builder \
  --set command=bootstrap \
  -n service-foundry --create-namespace \
  --version $SF_BUILDER_CHART_VERSION
```

After setup, platform users can log in to the console and provision applications from a form-based UI — with YAML manifests automatically committed to Git.

Service Foundry Console

The Service Foundry Console provides a self-service web interface for platform users to provision and manage workloads. By default, it is accessible at:

http://sfapp.{your-root-domain}

NOTE Ensure that your DNS is configured to point to the Traefik load balancer.

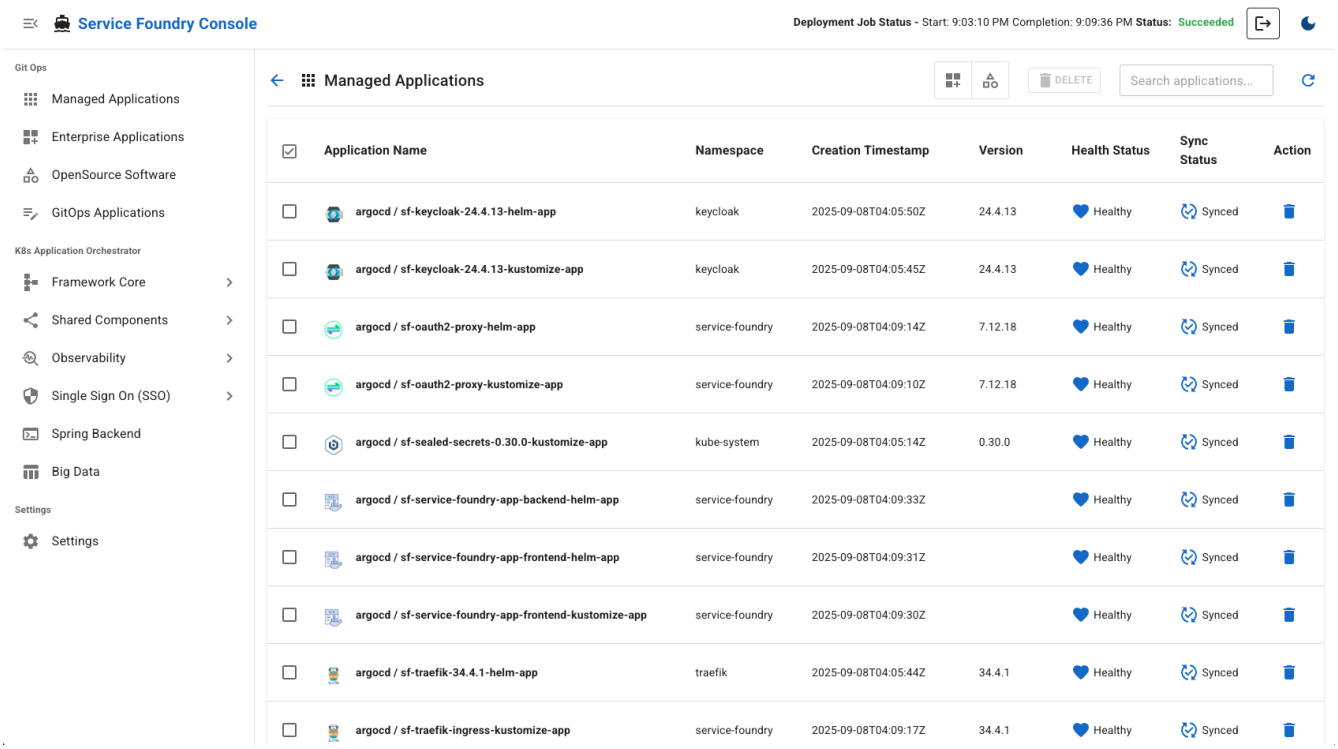


Figure 2. Service Foundry Console

All components, including the console itself, are deployed as **Argo CD Applications**. Their manifests are stored and versioned in the GitOps repository, ensuring full traceability and reproducibility. Users can operate either via the console or directly using Git workflows.

Git Repository

The Git repository initialized during bootstrap stores all Kubernetes manifest files managed by

Service Foundry — including infrastructure modules, Argo CD Applications, Helm values files, SealedSecrets and Kubernetes Resources.

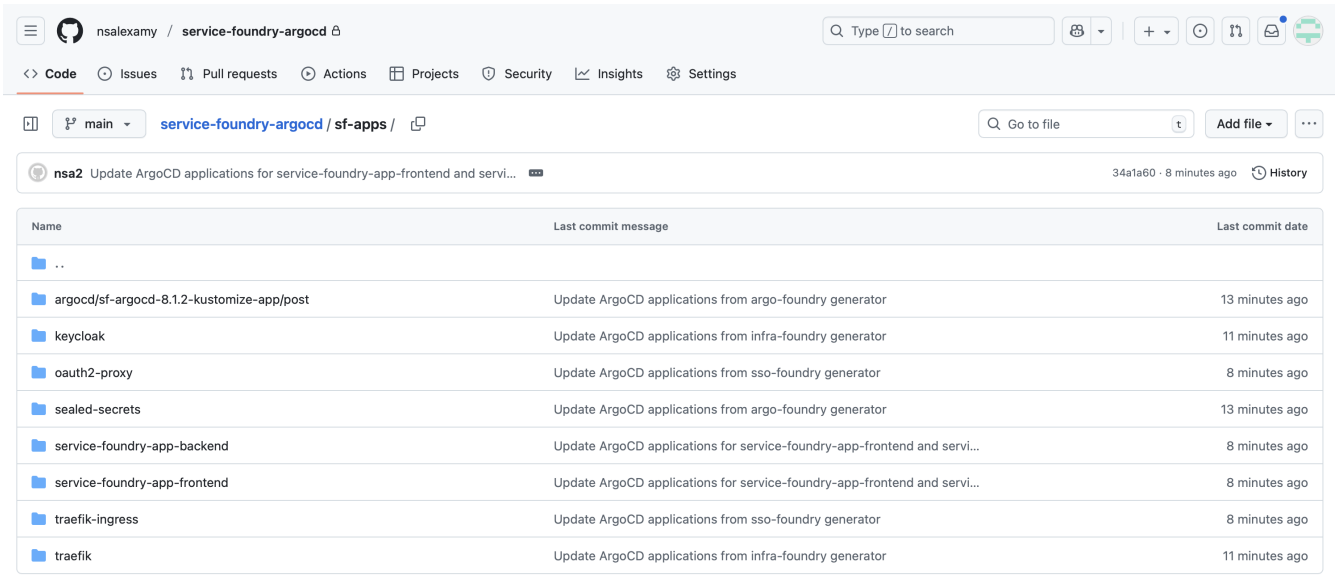


Figure 3. GitOps Repository with Initial Structure

This centralized structure supports both manual Git operations (clone, commit, push, etc.) and visual editing via the Service Foundry Console.

SealedSecrets for Secure Configuration

Sensitive data (e.g., passwords, tokens, API keys) are encrypted using Bitnami SealedSecrets before being committed to Git. During bootstrap, Service Foundry automatically converts Kubernetes Secret manifests into SealedSecret manifests.

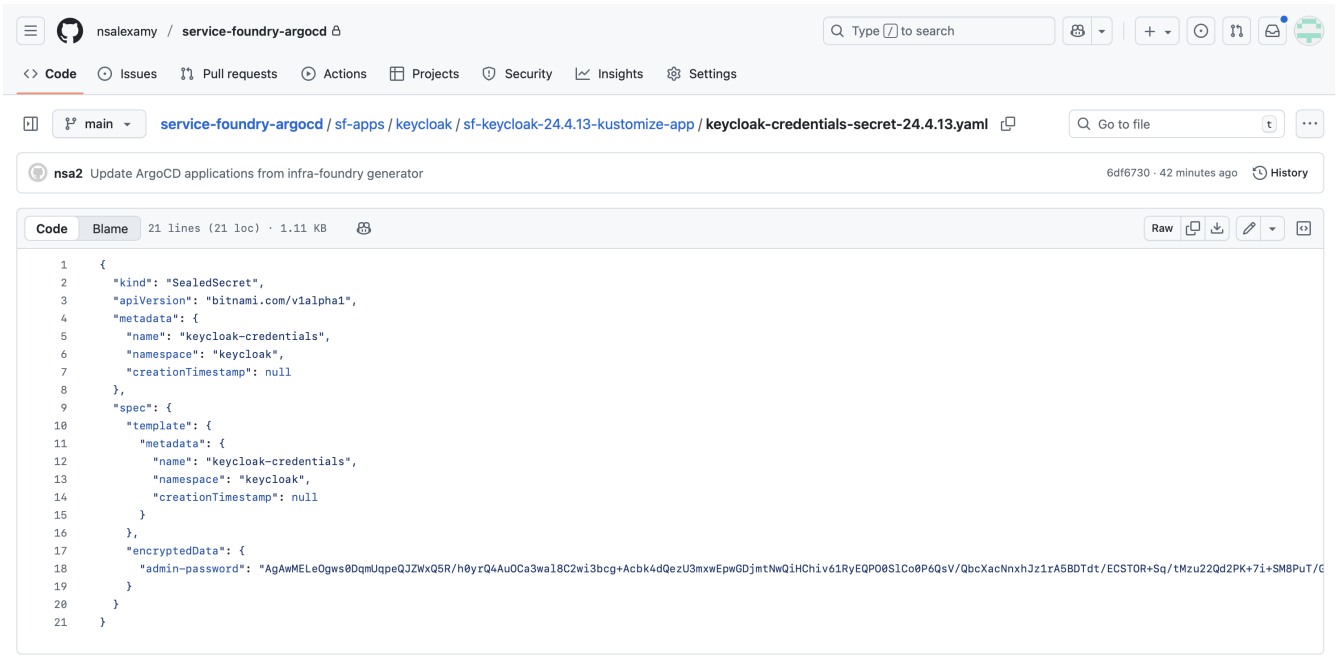


Figure 4. SealedSecrets Applied

Argo CD Integration

All components of the Service Foundry platform, Open Source Software applications, and user-defined applications are managed as Argo CD Applications. This ensures that the desired state defined in Git is continuously reconciled with the actual state in the Kubernetes cluster.

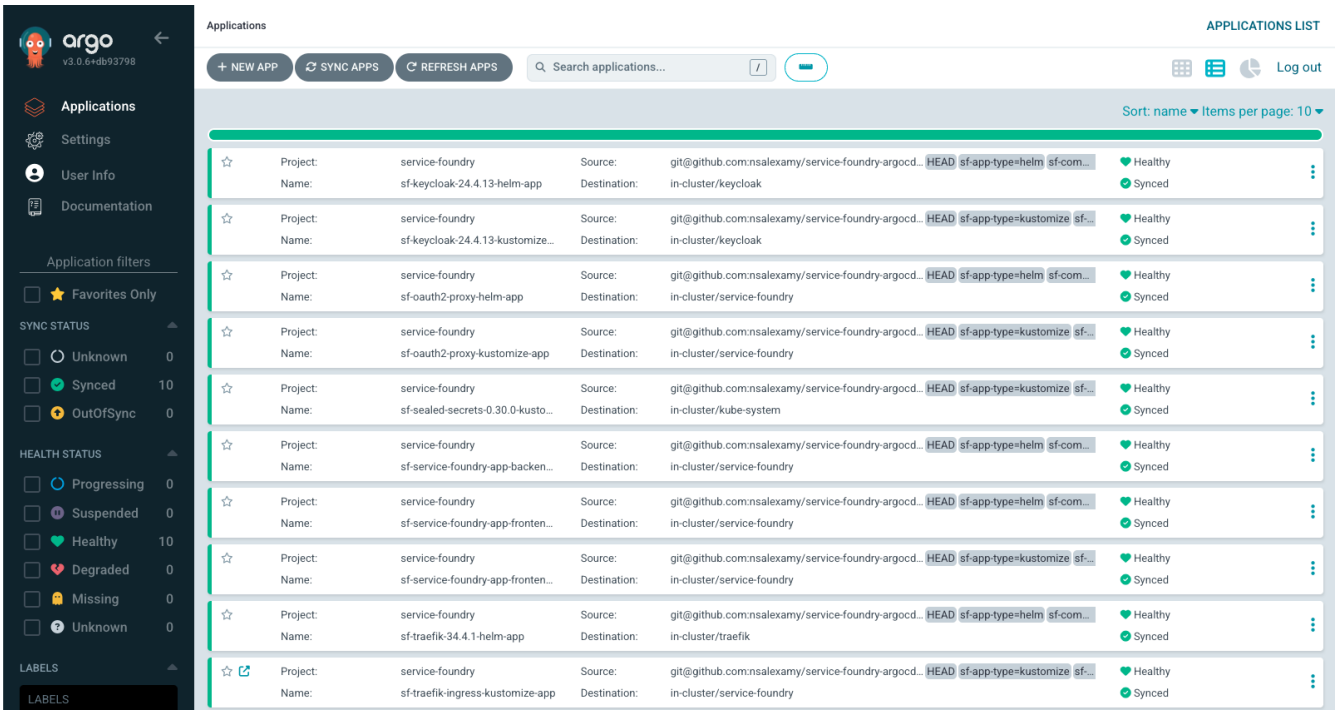


Figure 5. Argo CD Applications

The GitOps Repository is configured in Argo CD Settings as a repository, allowing Argo CD to sync the applications defined in the repository to the Kubernetes cluster.

Repository Configuration

Service Foundry automatically registers your GitOps repository in Argo CD during bootstrap, allowing continuous sync of application states.

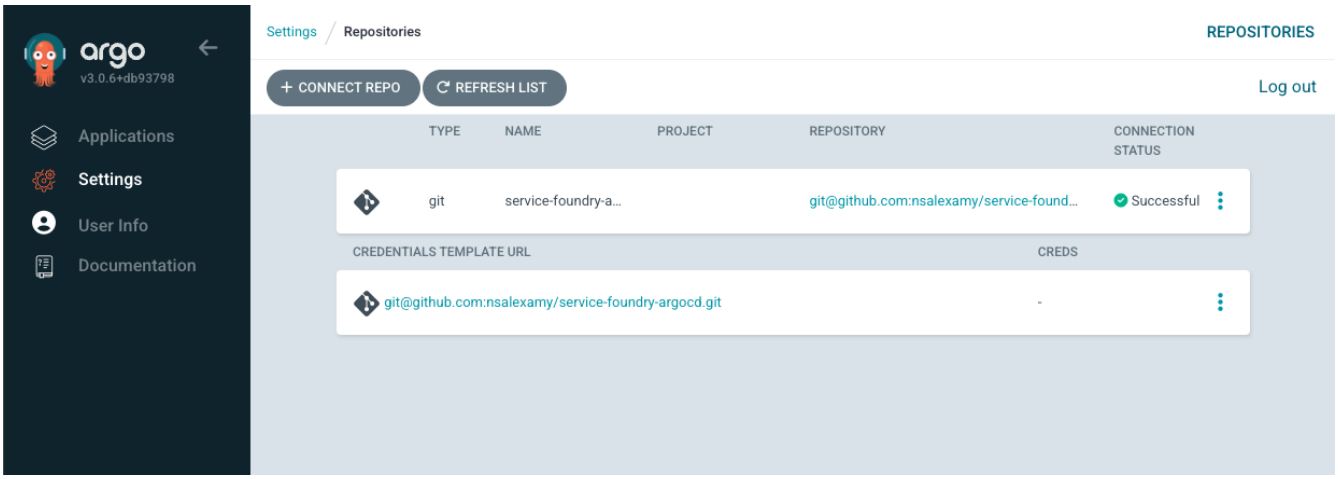


Figure 6. Argo CD Repositories

Argo CD Project Scope

A dedicated Argo CD Project named service-foundry is created by default to scope all applications deployed via Service Foundry.

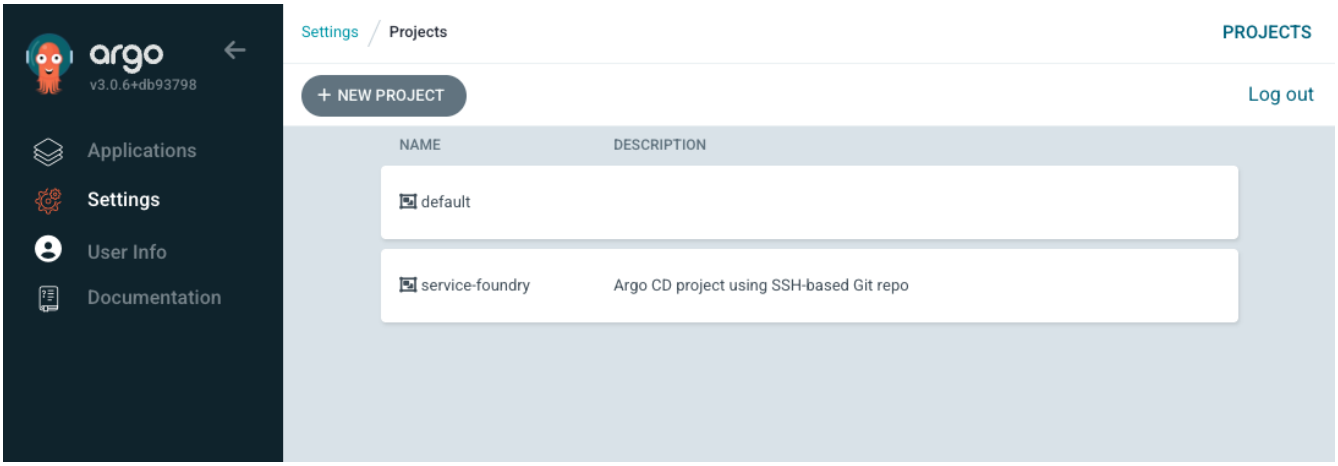


Figure 7. Argo CD Projects

Service Foundry Console Features

The **Service Foundry Console** provides a visual control plane for managing, observing, and operating applications and platform components deployed across your Kubernetes clusters. It offers a simplified interface for interacting with GitOps-managed resources and platform services.

GitOps-Centric Application Management

- **Managed Applications**: View the full list of Argo CD–managed applications, monitor their sync and health status, and perform lifecycle operations such as uninstall or update.
- **Enterprise Applications**: Deploy and manage proprietary or internal software packages, typically sourced from private Helm registries.
- **Open Source Software**: Browse and install curated open-source packages (e.g., Redis, Postgres, Kafka) with Helm chart–based GitOps workflows.
- **Raw GitOps Applications**: Directly edit or remove raw Argo CD Applications stored in the GitOps repository.

Kubernetes Stack Orchestration

- **Framework Core**: Inspect and manage foundational components installed during platform bootstrap (e.g., Argo CD, Traefik, SealedSecrets).
- **Shared Components**: Control cluster-wide operators and services that support multiple stacks — such as Prometheus Operator, OpenTelemetry Operator, Spark Operator.
- **Observability Stack**: Manage observability tools like Grafana, Prometheus, Loki, Tempo, and their provisioning.
- **Single Sign-On (SSO)**: Configure identity providers and manage user access via Keycloak and

OAuth2 Proxy.

- **Big Data Stack**: Deploy scalable analytics infrastructure — including Apache Spark, Airflow, Neo4j, and OpenSearch.
- **Spring Backend Stack**: Easily deploy and operate Spring Boot–based microservices using standard templates and Helm charts.

Console Settings

- **Global Configuration**: Manage environment-wide settings related to GitOps, authentication, UI preferences, and more.

Managed Applications

The Managed Applications section provides a centralized dashboard for all Argo CD applications deployed in the Kubernetes cluster. From here, users can monitor, modify, and manage applications through a GitOps-driven workflow.

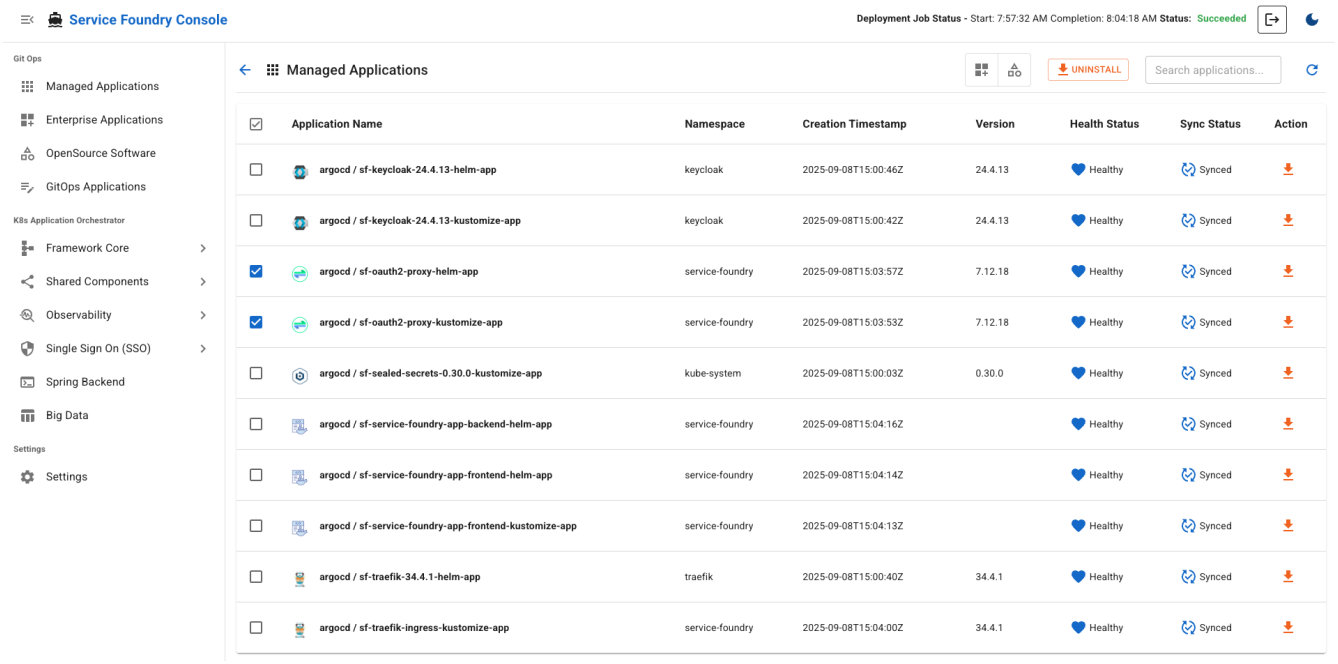


Figure 8. Managed Applications

Platform users can:

- View the sync and health status of all deployed Argo CD applications
- Uninstall selected applications directly from the UI
- Filter between Open Source and Enterprise applications
- Navigate to application-specific views for further inspection

Application Filters and Actions

Table 2. Buttons on the header section:

Button	Action
	Toggle to display only **Enterprise Applications**.
	Toggle to display only **Open Source Software applications**.
Uninstall	Click to **uninstall the selected applications** from the cluster and remove it from the Git repository.

Table 3. Buttons on each application row:

Button	Action
	Uninstall the application in a row.

Clicking on any application name opens the detailed application view with the following tabs:

Application Files

Users can inspect and modify manifest files directly in the browser-based editor. The Console supports a Git-aware editing workflow:

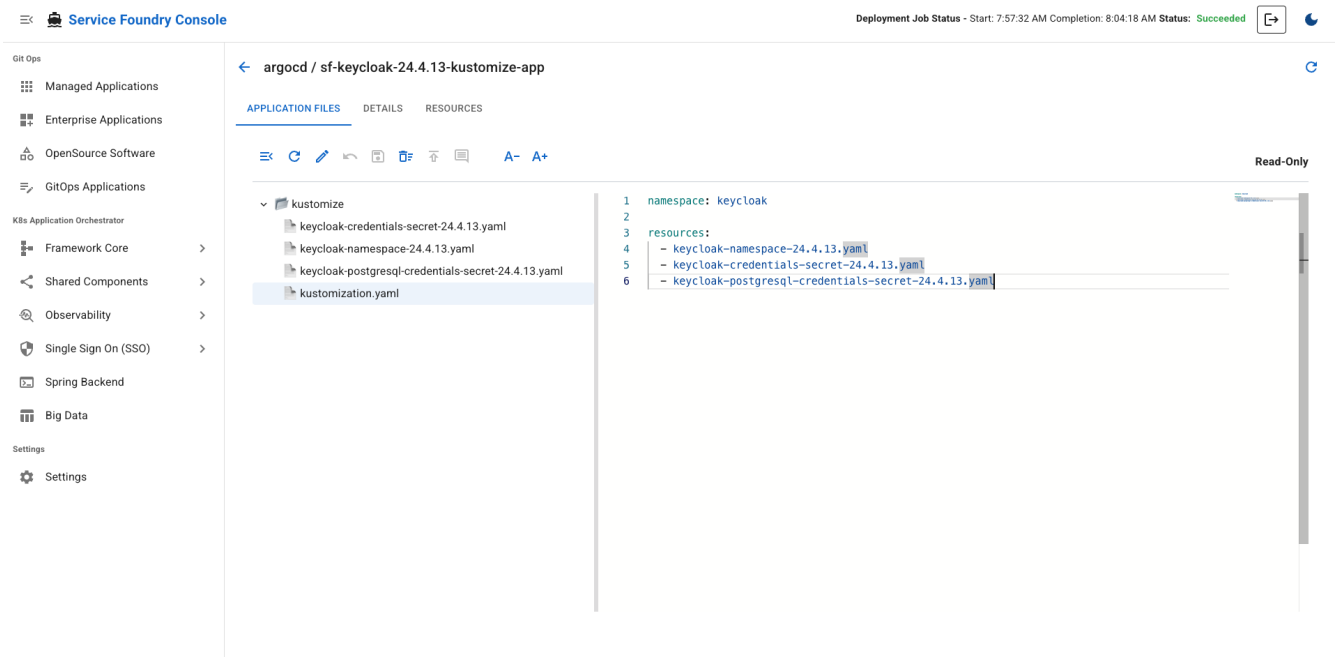


Figure 9. Application File Editor Tab

Table 4. Buttons on the editor header section:

Button	Action
	Hide File Tree.
	Show File Tree.
	Refresh the file tree to reflect the current state from Git.
	Enable editing mode for the selected manifest file.
	Discard edits and revert to the last committed version.
	Save changes made to the manifest files. This will stage the changes for commit.

Button	Action
	Discard all unsaved changes across files.
	Commit and push changes to the Git repository (triggers Argo CD sync).
	Add a Git commit message before pushing.
A−	Decrease font size in the file editor.
A+	Increase font size in the file editor.

Application Details Tab

The **Details** tab shows metadata for the selected Argo CD application, including:

- Application name and namespace
- Argo CD project
- Sync status (Synced / OutOfSync)
- Health status (Healthy / Degraded / Missing)
- Source repository path and revision

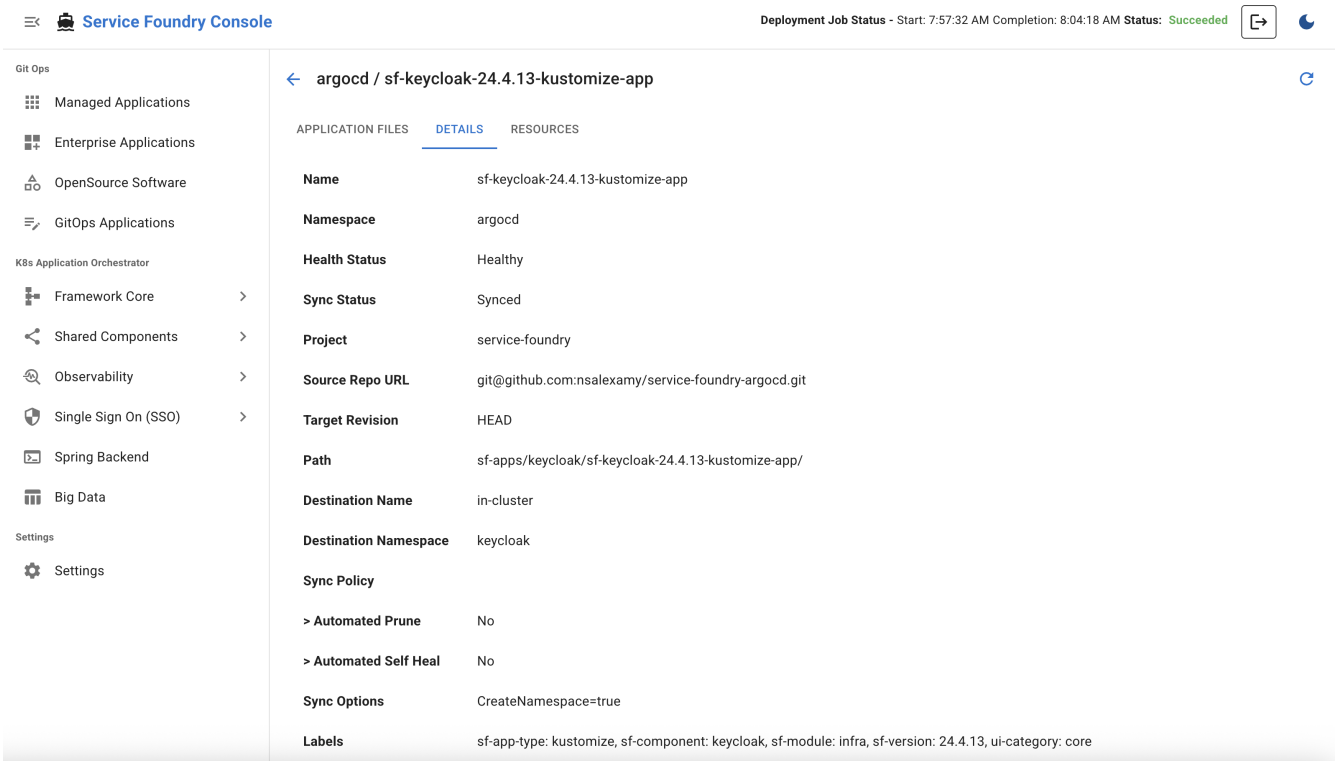


Figure 10. Application Details Tab

Kubernetes Resources Tab

The Resources tab lists all Kubernetes resources associated with the application. It allows users to:

- View resource type, name, namespace, and status
- Drill down into specific workloads (e.g., Deployments, Services, Secrets)

- Monitor resource state and lifecycle

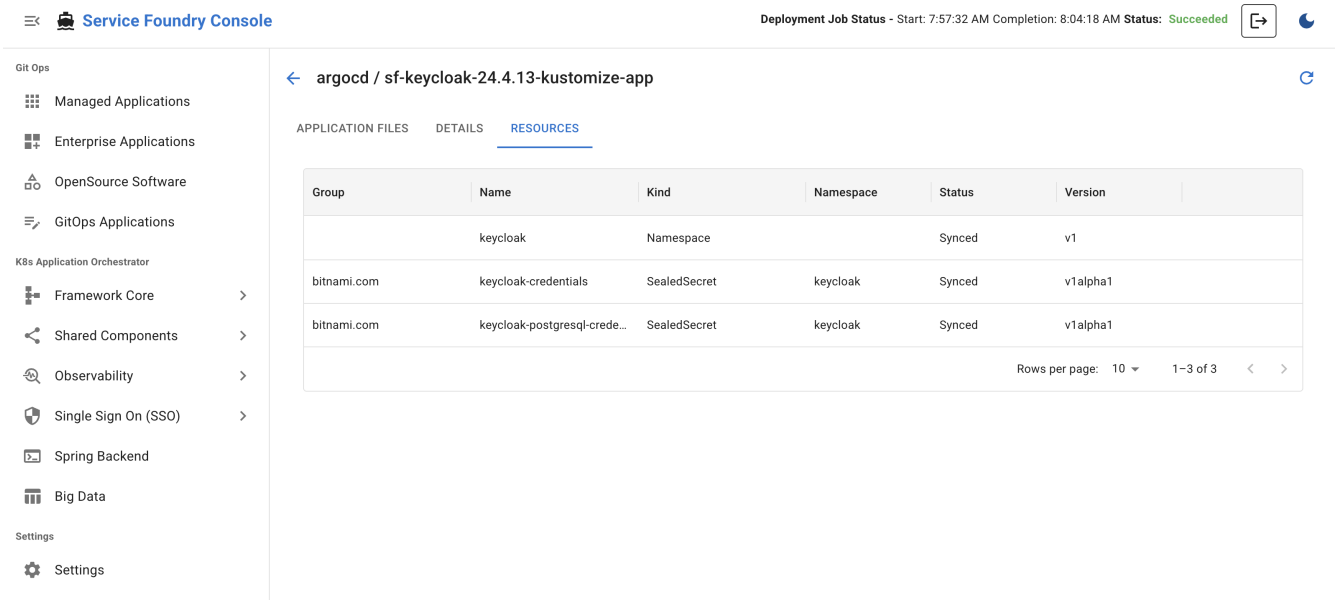


Figure 11. Kubernetes Resources Tab

Enterprise Applications

The Enterprise Applications section enables teams to define, deploy, and manage internal business applications sourced from private container registries. Service Foundry supports both Helm-based and Kustomize-based GitOps workflows, allowing users to scaffold applications using reusable templates and manage deployments through Argo CD.

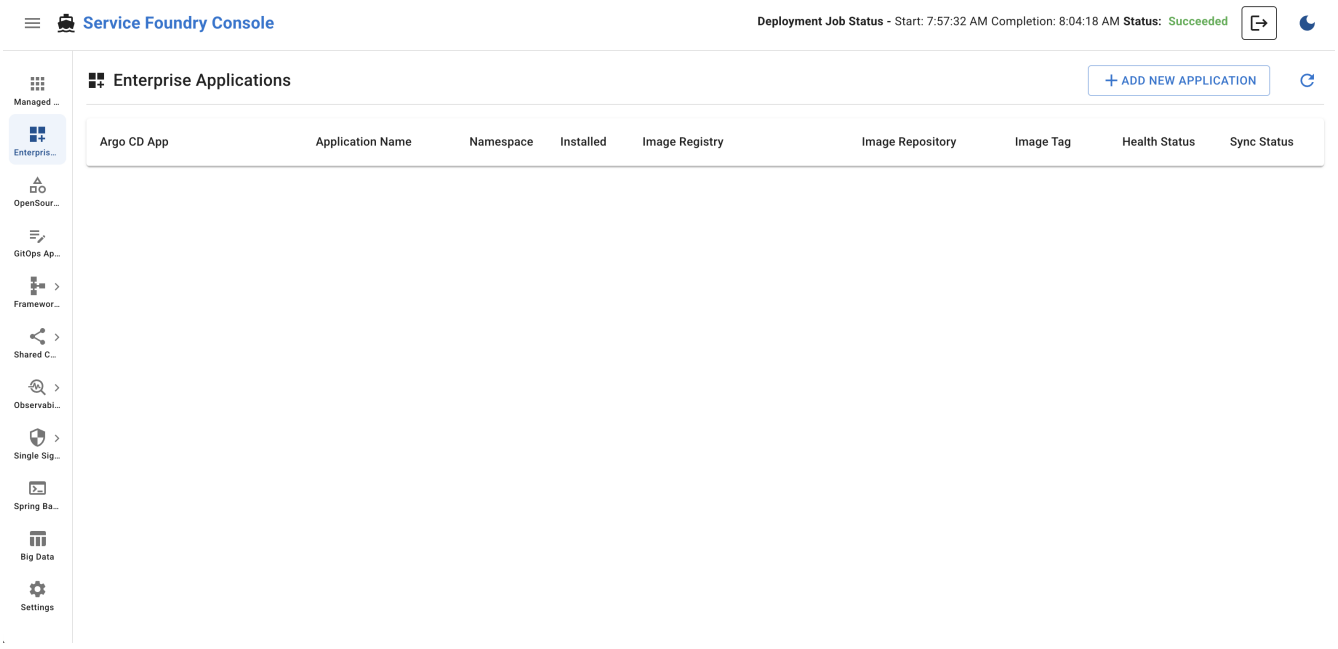


Figure 12. Enterprise Applications

To create a new application, click “Add New Application”, then select either a Helm or Kustomize deployment model depending on your application packaging format.

Create Enterprise Application

Users can scaffold enterprise applications by filling in a guided form. Service Foundry generates the required manifest files and Argo CD Application resources, commits them to the GitOps repository, and deploys them automatically.

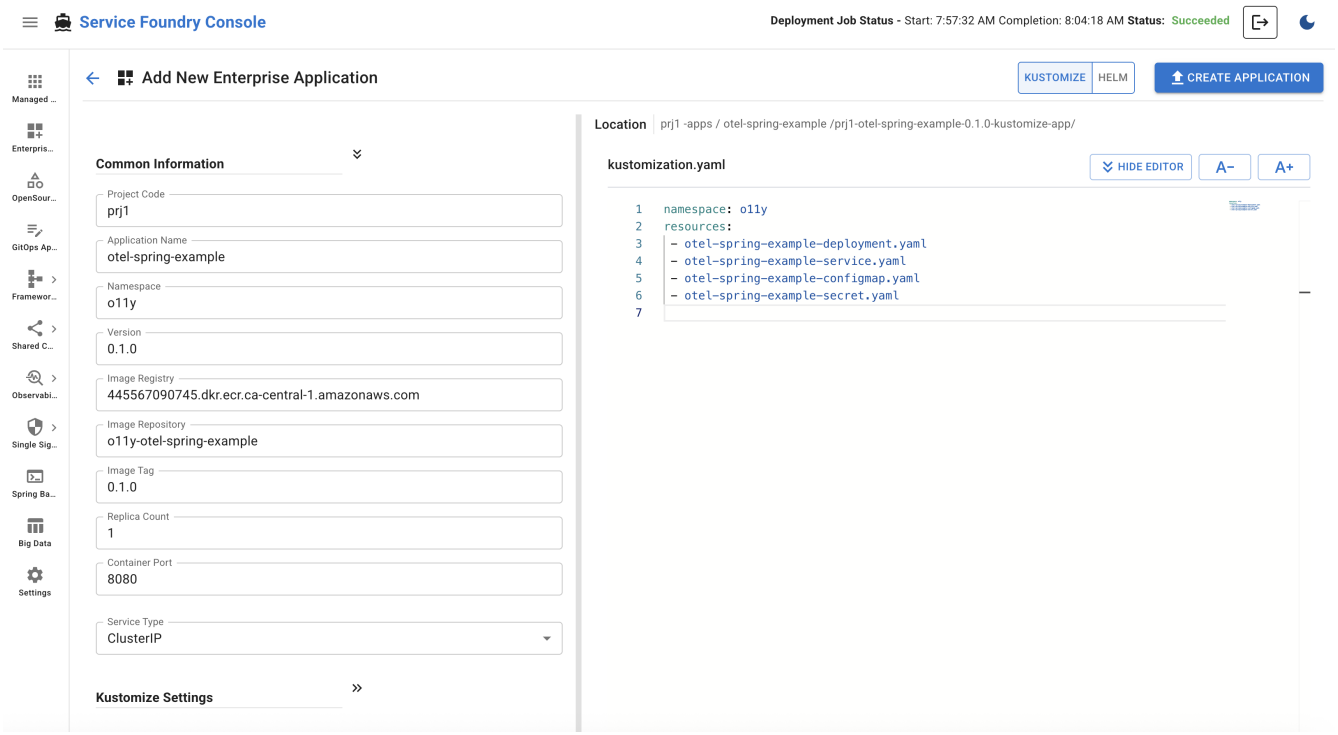


Figure 13. Create Enterprise Application

Common Fields

These fields are shared between Helm and Kustomize application types and are used to populate Kubernetes manifests and Argo CD configuration files.

Table 5. Common Fields Form

Field name	Description	example
Project Code	Logical identifier used as a prefix for Kubernetes resources.	prj1
Application Name	Application-specific identifier used in resource naming.	myapp
Namespace	Target Kubernetes namespace for deployment.	prj1
Version	Chart version (for Helm) or application version tag.	0.1.0
Image Registry	Hostname of the Docker image registry.	ghcr.io
Image Repository	Repository path to the container image.	o11y-otel-spring-example
Image Tag	Docker image tag to be deployed.	0.1.0
Replica Count	Number of application pods to deploy.	2
Container Port	Port the application container listens on.	8080

Field name	Description	example
Service Type	Kubernetes Service type. Options: ClusterIP, NodePort, LoadBalancer.	ClusterIP

Kustomize-Based Application

For applications defined using raw Kubernetes manifests, Service Foundry provides a visual resource composer for generating Kustomize-based applications. Users can dynamically add/remove/rename Kubernetes resource templates (e.g., Deployment, Service, ConfigMap, Secret, Ingress, etc.).

Table 6. Manage Kustomize Resources

Button	Action
Add Resource	Insert a new resource (e.g., Deployment, Service) into the manifest set.
Remove	Delete the selected resource from the Kustomize structure.
Rename	Rename an existing resource before creation.

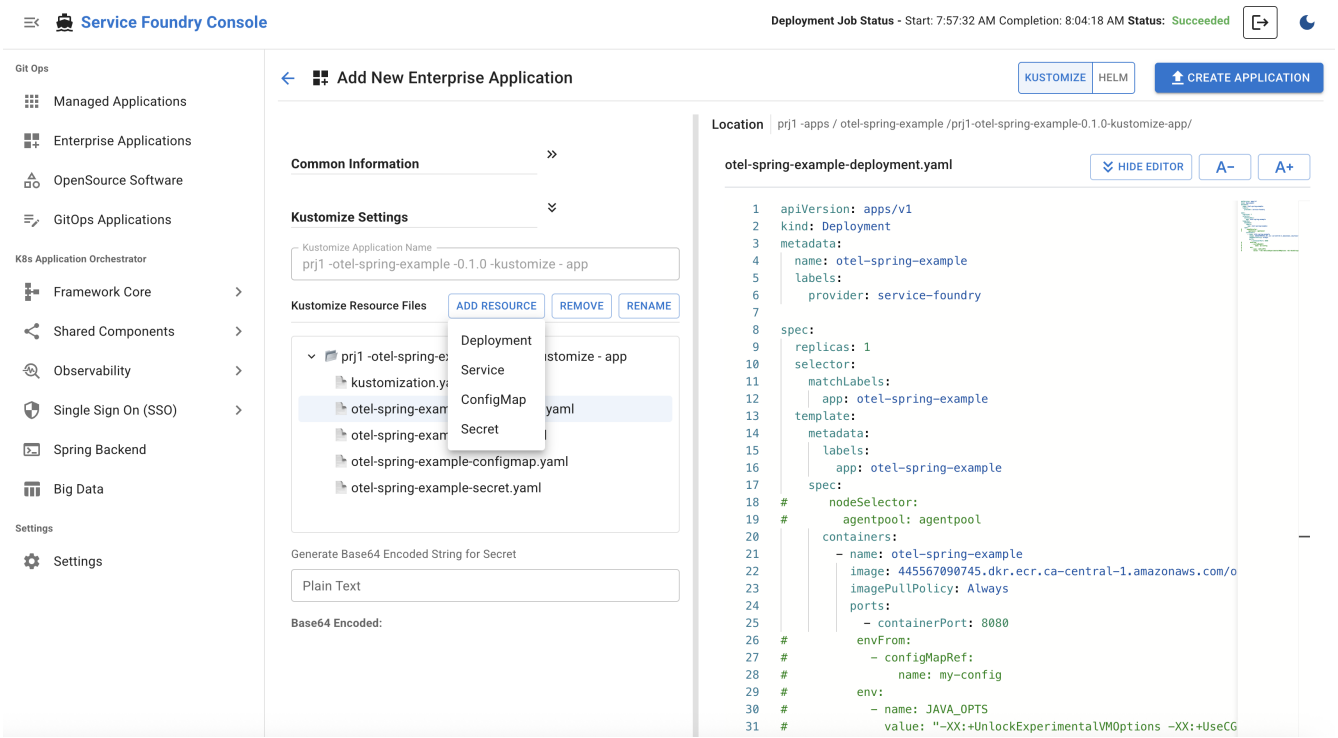


Figure 14. Create Kustomize Application - Kustomize

Once resources are configured, users can review and edit the generated YAML files in the built-in editor.

Deploy via GitOps

Click “**Create Application**” to generate the necessary manifest and Argo CD application files. These files are automatically committed to the GitOps repository and Argo CD will detect and deploy the new application to the Kubernetes cluster.

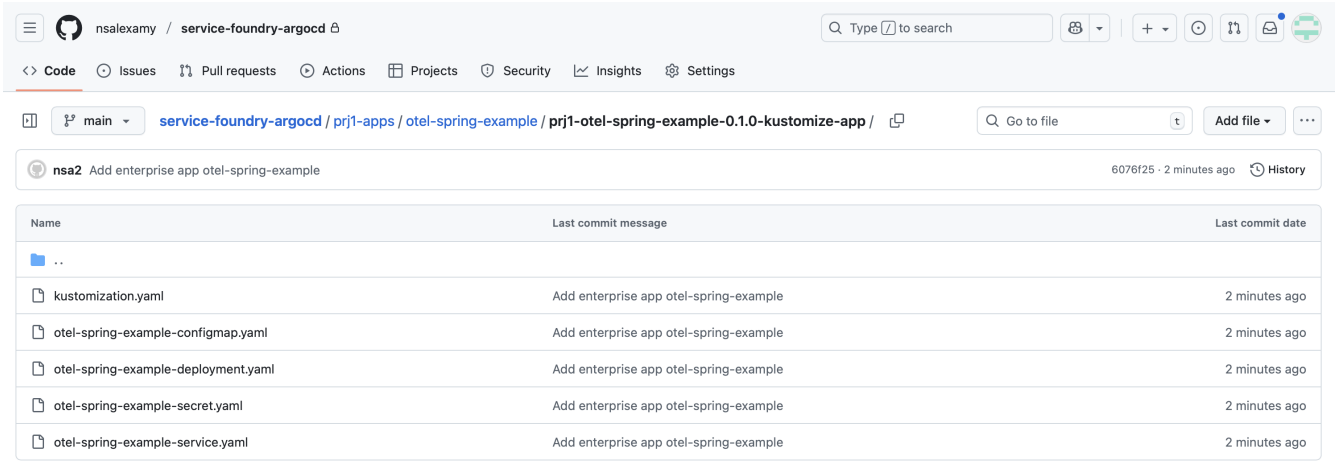


Figure 15. GitOps Repository after Creating Kustomize Application

The newly created application will also appear in the **Managed Applications** section of the Console.

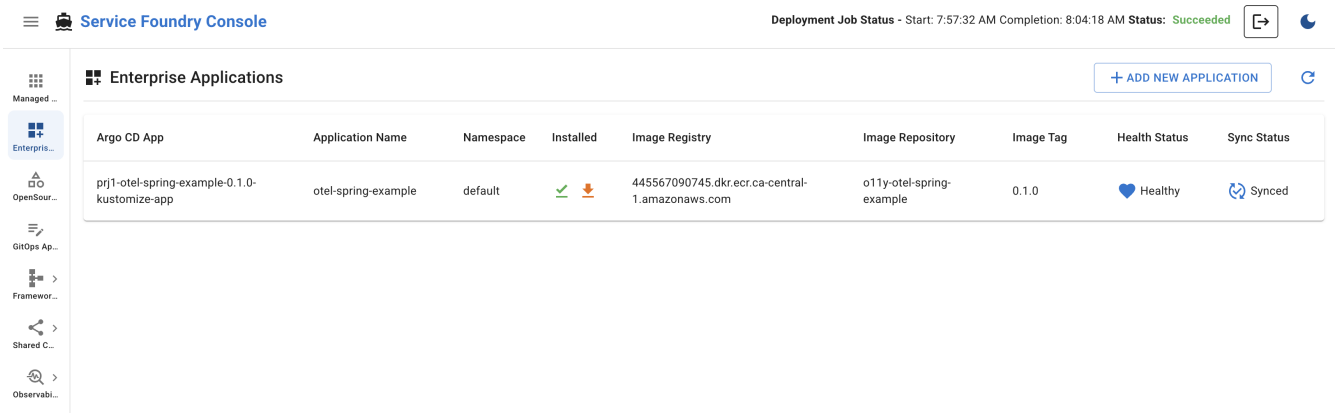


Figure 16. Enterprise Application Created

You can click on the application name to:

- Inspect deployment manifests
- View sync and health status
- Access related Kubernetes resources

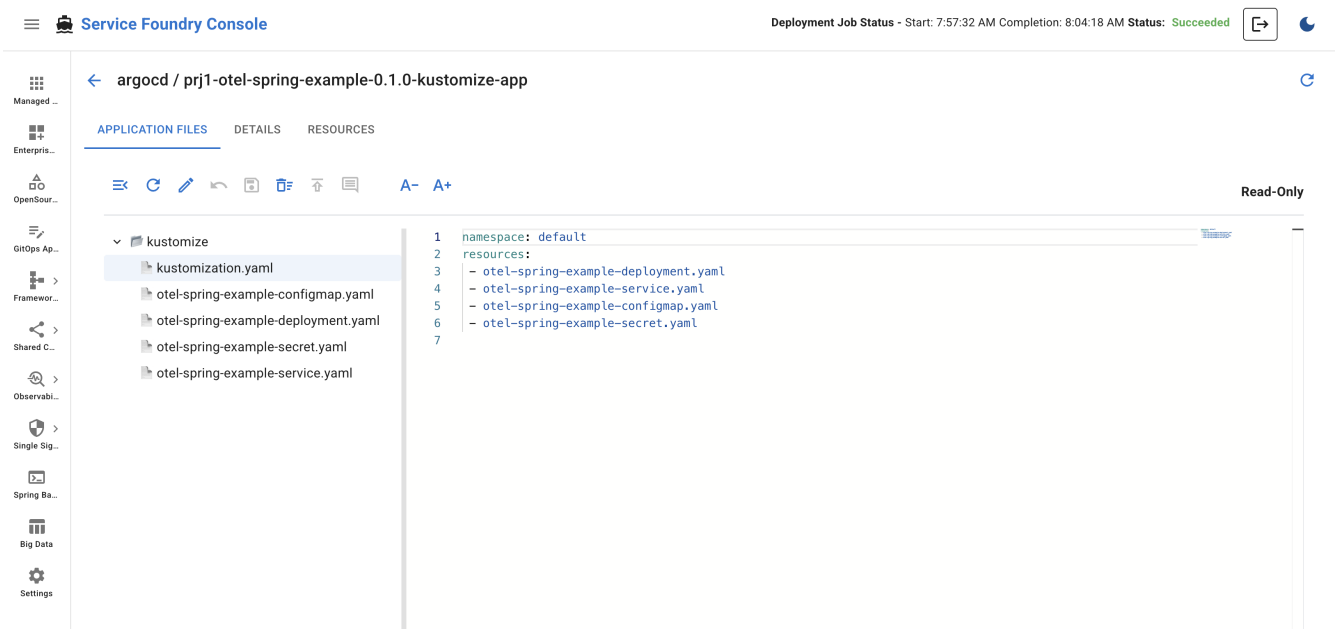


Figure 17. Enterprise Application Details

All generated manifest files remain fully editable from the Console. Once modified, changes can be committed and pushed directly to the Git repository, triggering Argo CD to re-sync the application and apply updates to the cluster.

Open Source Software

Service Foundry provides a curated catalog of popular open-source applications that can be deployed seamlessly using Helm charts from public registries. These applications can be provisioned with just a few clicks and are fully integrated into the GitOps workflow.

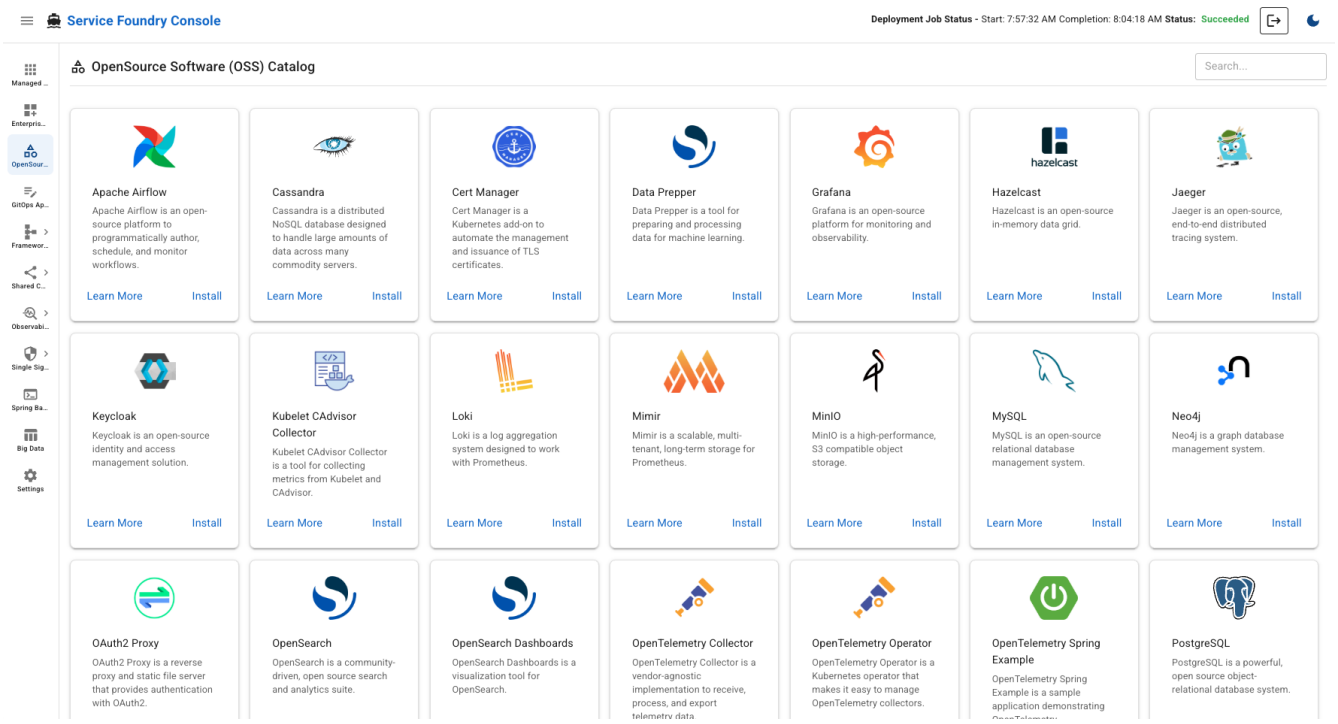


Figure 18. Open Source Software Catalog

To install an application, select it from the catalog and click “Install”. Service Foundry guides you

through a streamlined setup process using Helm-based configuration templates.

Example: Installing PostgreSQL

When installing PostgreSQL, users are prompted to configure essential parameters such as:

- Database username and password
- Initial database name
- Helm chart version

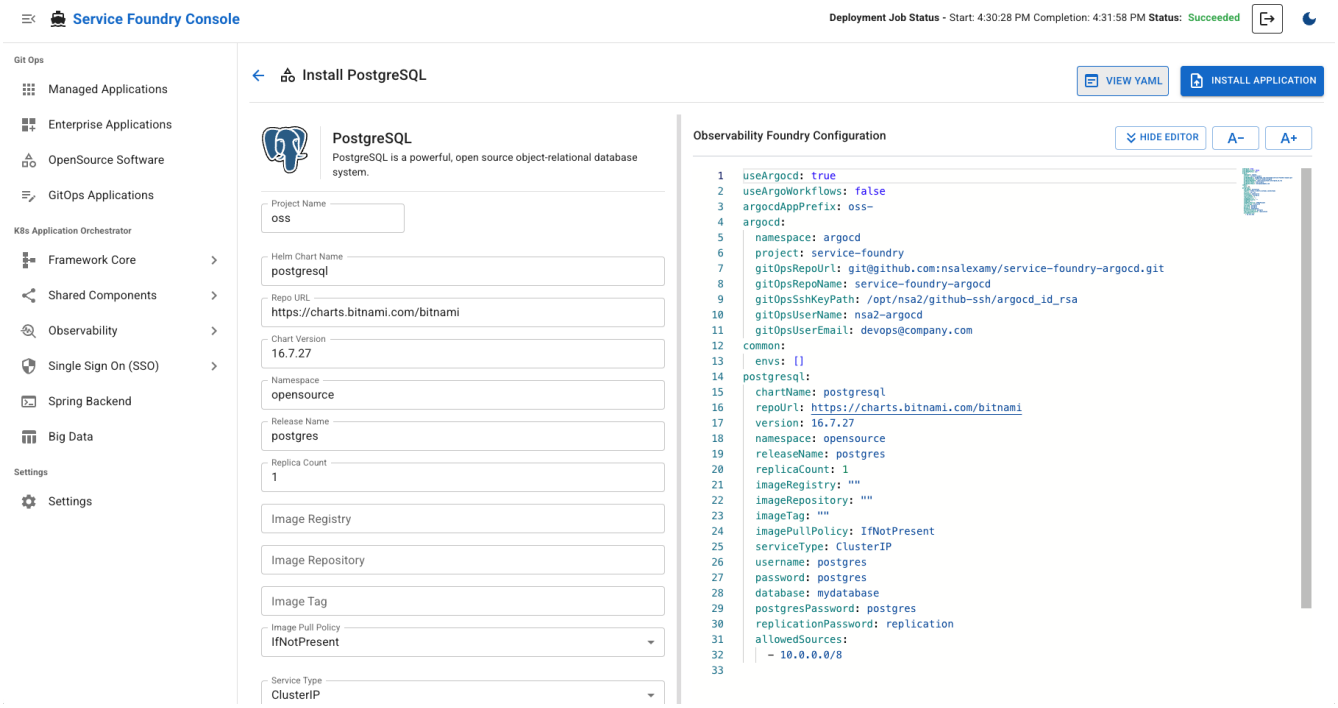


Figure 19. Install Open Source Software Application

After customization, clicking “Install Application” will:

- Generate Helm-based Kubernetes manifests
- Commit them to the GitOps repository
- Create a corresponding Argo CD application to manage the deployment

During installation, the Job Status is displayed in the header area to track progress in real time.

Figure 20. Job Status - In Progress

GitOps Repository Integration

Once installation is complete, all manifests and configuration files are stored in the GitOps repository under a versioned path.

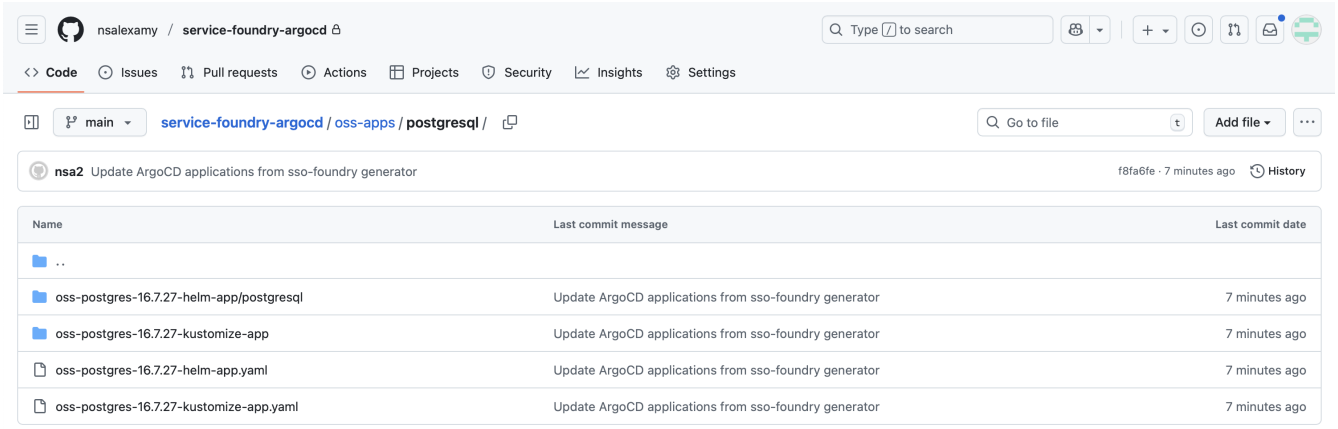


Figure 21. GitOps Repository after Installing Open Source Software Application

The deployed application will also appear in the Managed Applications section of the Console, alongside other enterprise or custom workloads.

Application Details & Management

Click on any deployed open-source application to view detailed information such as:

- Kubernetes manifests
- Helm release values
- Resource status and health
- Argo CD sync history

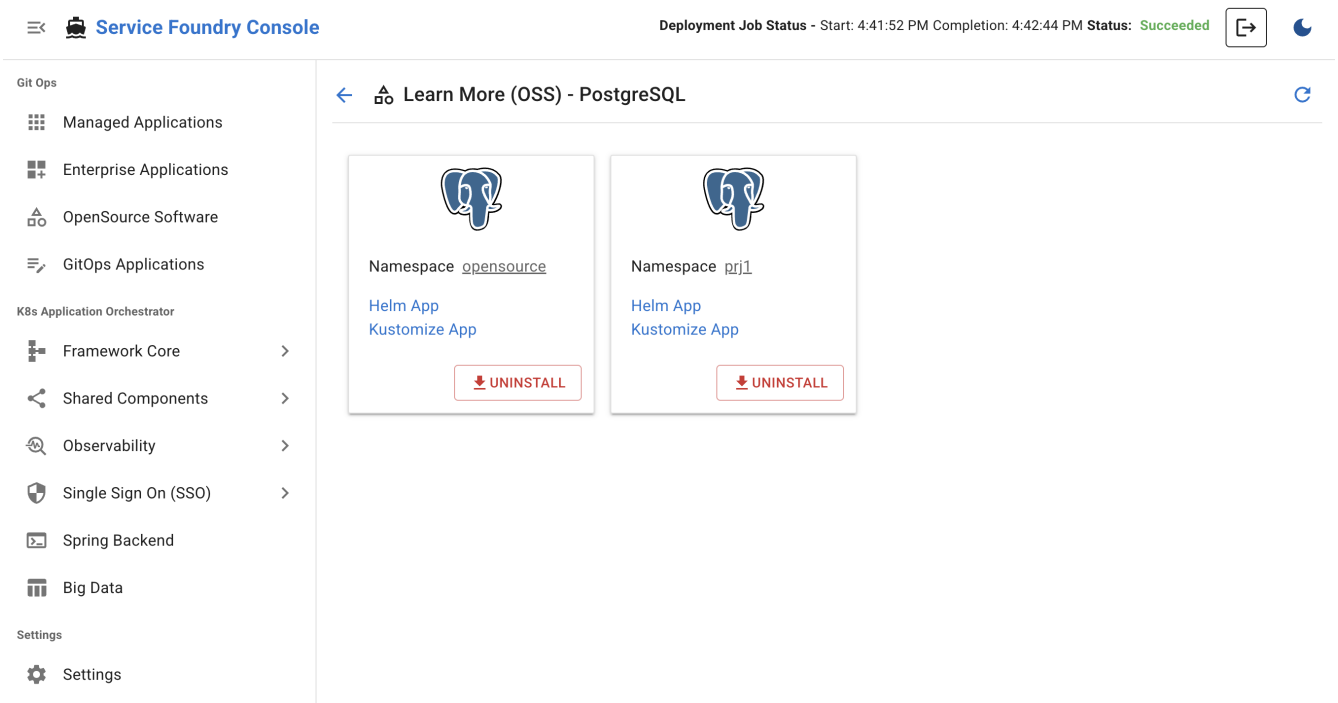


Figure 22. Open Source Software Application Details

Applications can be deployed into different namespaces, and each instance is managed as an isolated Argo CD application with its own configuration and lifecycle.

Application Actions

Buttons to manage the application:

Table 7. Application Actions

Button	Action
Helm App	View Helm-specific values and manifests associated with the deployment.
Kustomize App	View Kustomize manifests if the application was scaffolded using Kustomize.
UNINSTALL	Remove the application from the Kubernetes cluster and delete the associated Argo CD application. Note: Manifest files will remain in the Git repository for audit purposes.

GitOps Applications

The GitOps Applications section enables full lifecycle management of raw Kubernetes manifests stored in the Git repository. Users can create, edit, deploy, and remove applications using a Git-centric workflow—without needing to leave the console.

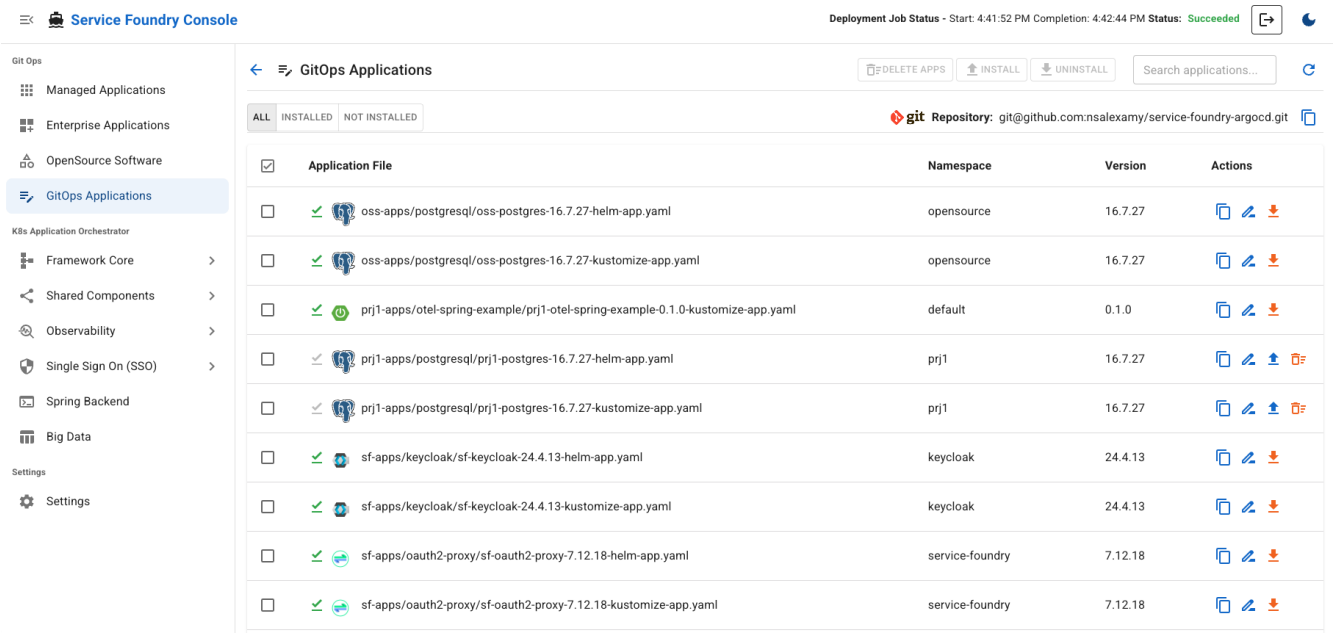


Figure 23. GitOps Applications

Service Foundry allows you to reuse existing Kubernetes manifests in your GitOps repository to scaffold new Argo CD applications.

Application List — Actions

Each application entry supports quick access to common actions:

Table 8. Action Buttons in the application list

Button	Action
	Copy the full path to the Argo CD application file for reference or external tooling.
	Edit the Argo CD application manifest directly in the built-in file editor.
	Deploy the application to the cluster by creating the corresponding Argo CD application. Available only if the application is not yet installed.
	Uninstall the application from the cluster and remove all associated Kubernetes resources. Available only if already installed.
	Permanently delete the application manifest from the Git repository. This action cannot be undone.

Batch Operations — Header Actions

The header also provides multi-application controls for bulk operations:

Table 9. Header Buttons for Batch Operations

Button	Description
 DELETE APPS	Delete selected GitOps application manifests from the repository. Irreversible.
 INSTALL APPS	Create Argo CD applications for selected manifests and deploy them to the cluster.
 UNINSTALL APPS	Uninstall selected applications and remove their Kubernetes resources from the cluster.

Click an application name to view detailed information and edit files in place.

View GitOps Application

Users can drill down into each GitOps application to inspect and manage its manifests. This includes editing files, creating Argo CD apps, and managing installation status.

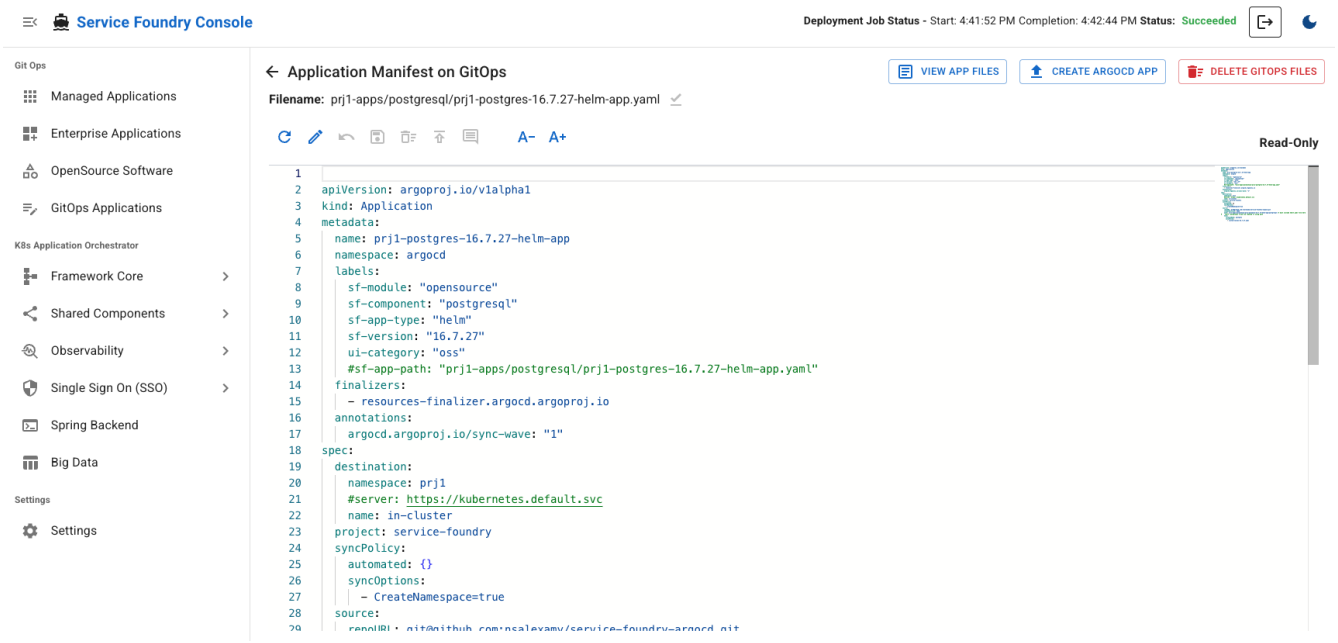


Figure 24. GitOps Application Details

Following buttons are available in the header for application-specific actions:

Table 10. Header Buttons for Application Management

Button	Action
View App Files	Open the file tree to browse and edit application manifests.
Create ArgoCD App	Create a new Argo CD application to deploy the manifest to the Kubernetes cluster.
Uninstall ArgoCD App	Remove the Argo CD application and associated resources from the cluster.
DELETE APP	Permanently delete the application manifest from the Git repository. This action is irreversible.

Any edits to the manifest files can be committed directly from the console, triggering Argo CD to sync changes and apply them to the cluster automatically.

Kubernetes Stack Orchestration

Service Foundry supports modular, stack-based orchestration for managing complex Kubernetes workloads. Each stack bundles a set of components into a cohesive deployment unit, enabling teams to layer functionalities in a controlled and scalable manner.

Users can selectively deploy the stacks they need, in any order, while respecting inter-stack dependencies. For example, the Observability Stack can be layered on top of the Framework Core and Shared Components.

Available Stacks:

- Framework Core

- Shared Components
- Observability Stack
- Single Sign-On (SSO) Stack
- Spring Backend Stack (*Work In Progress*)
- Big Data Stack (*Work In Progress*)

Framework Core

The **Framework Core** is the foundation of Service Foundry. It includes critical services required for the platform to operate reliably from initial setup onward.

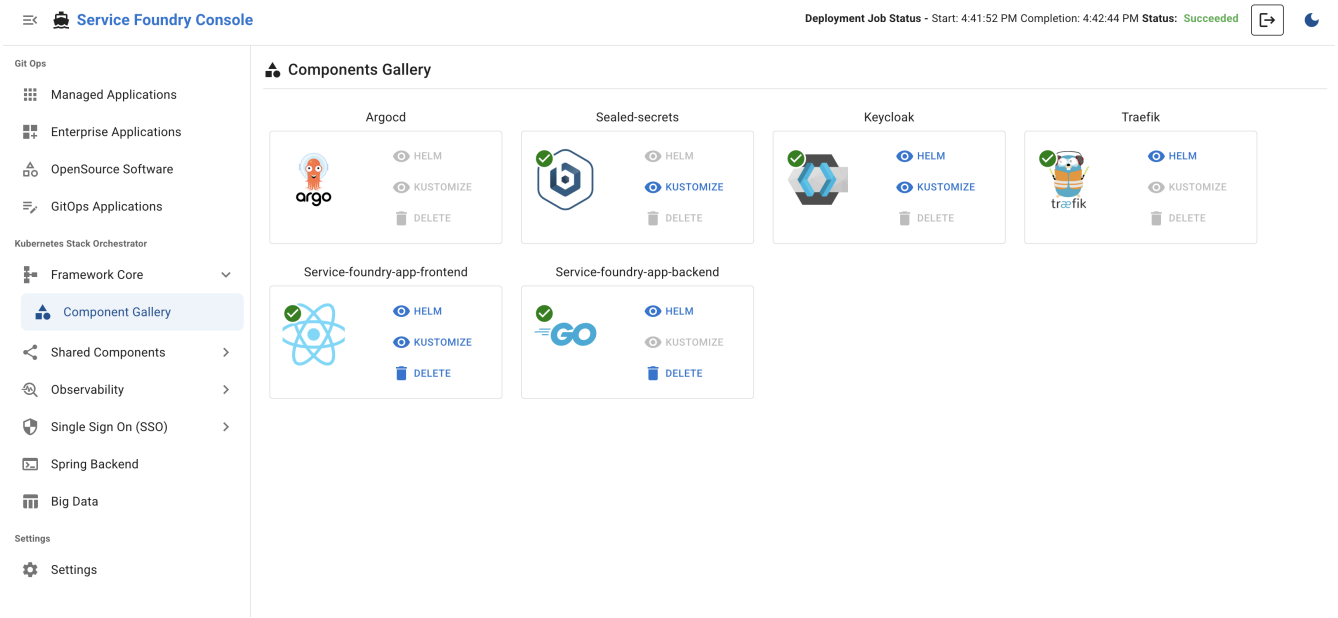


Figure 25. Framework Core Components

Each component in this stack is preconfigured for seamless integration. Users may review configuration details and adjust settings as needed—but it is not recommended to uninstall any components from this stack, as they are essential for overall platform stability and functionality.

Shared Components

The Shared Components stack includes reusable services and Kubernetes Operators that provide cross-cutting functionality across multiple application domains and stacks.

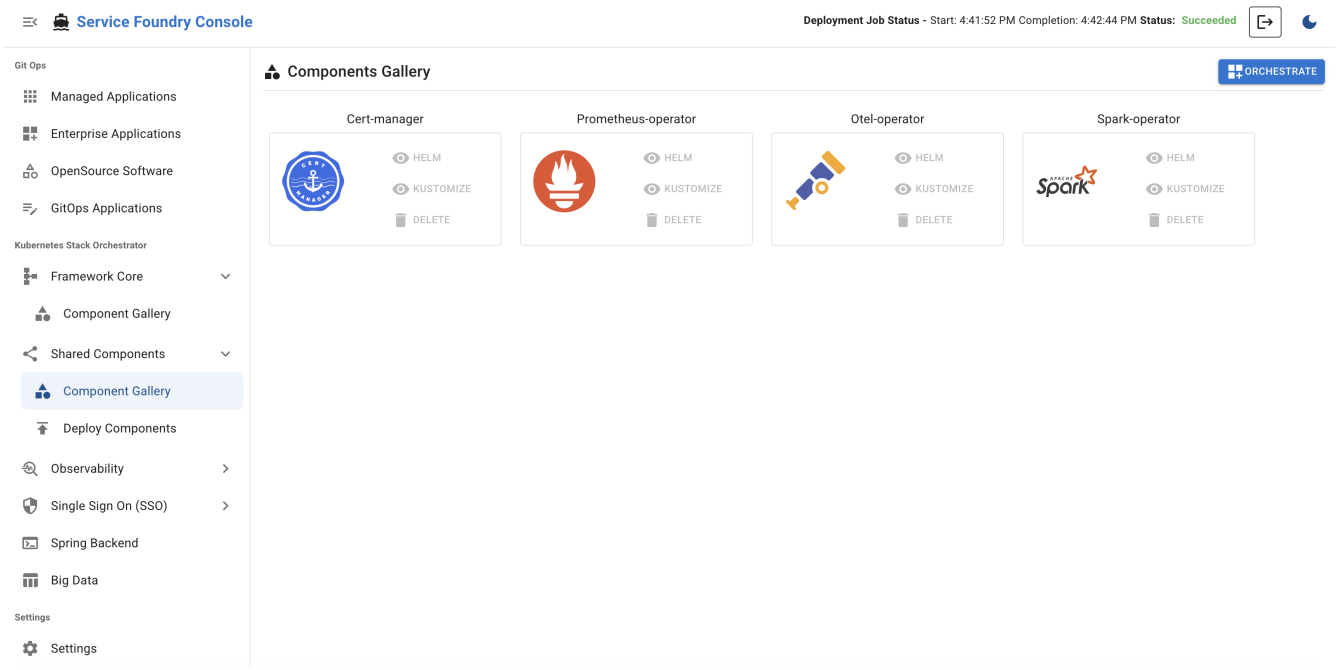


Figure 26. Shared Components

Operators such as prometheus-operator, opentelemetry-operator, and spark-operator are part of this stack. These are not mandatory, and users can choose to install only the components relevant to their use case.

For example, if you're deploying the Observability Stack, it's recommended to install the prometheus-operator and opentelemetry-operator beforehand.

Click **Orchestrate** to generate Kubernetes manifests, commit them to the GitOps repository, and deploy them via Argo CD.

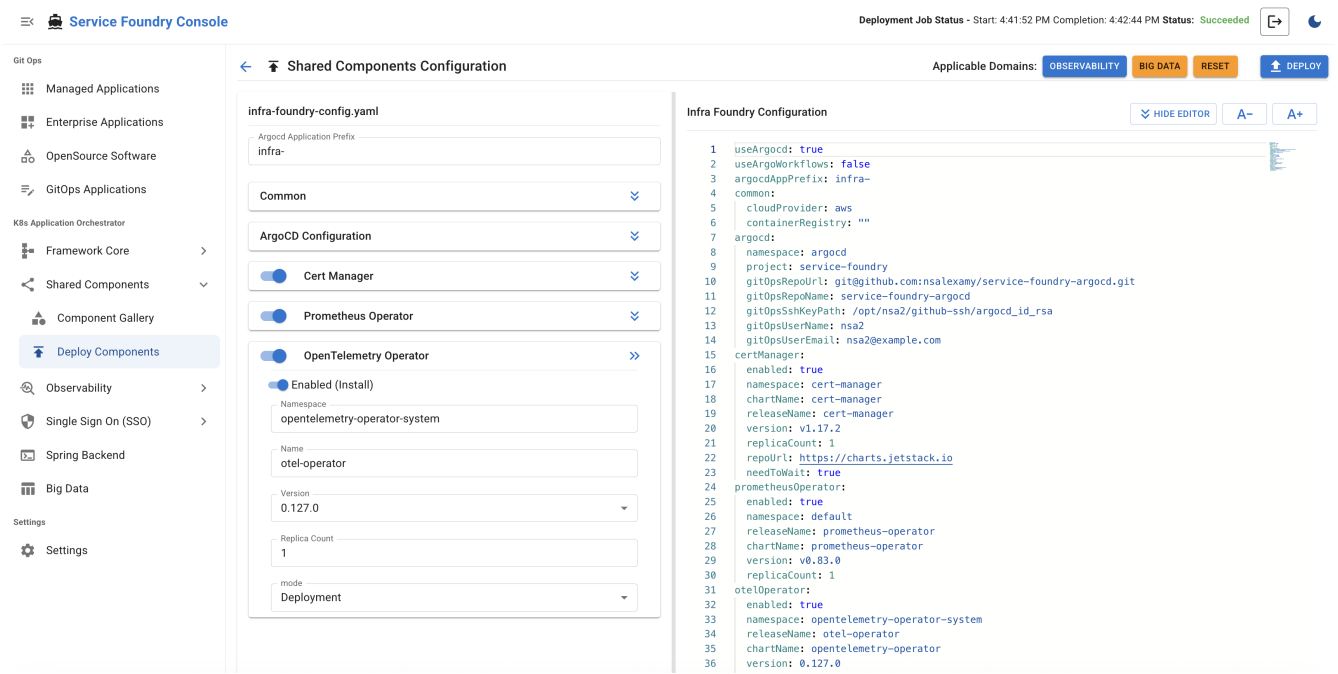


Figure 27. Deploy Shared Components

Applicable Domain

Table 11. Supported Domains and Required Components

Domain	Required Components
Observability	Cert-manager, Prometheus-operator, Opentelemetry-operator
Big Data	Cert-manager, Spark-operator

Upon deployment, manifests are organized under the infra-apps directory in the GitOps repository:

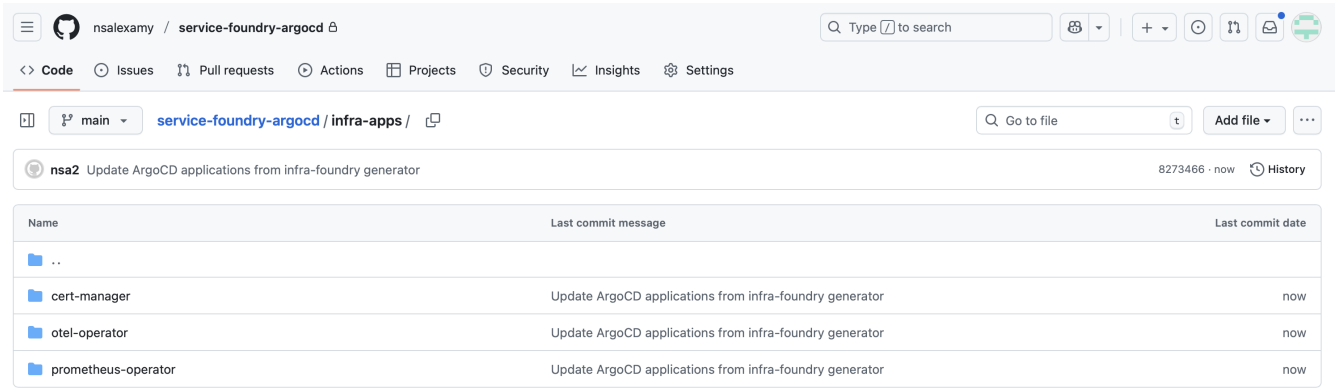


Figure 28. GitOps Repository - Shared Components

The stack will also appear in the **Managed Applications** section of the console:

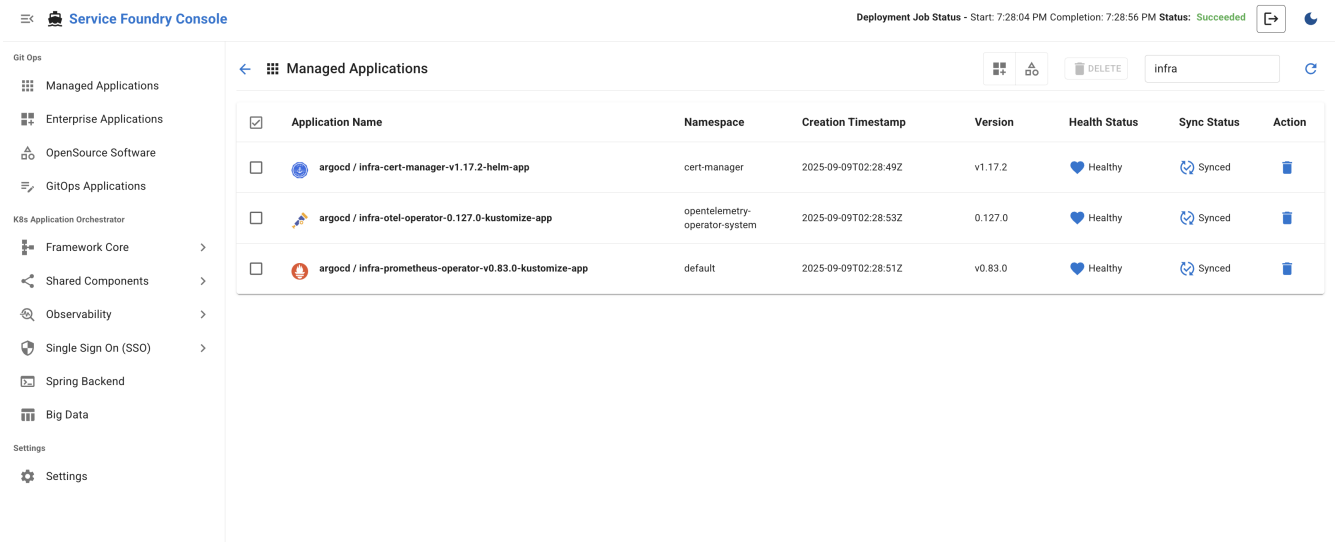


Figure 29. Managed Applications - Shared Components

Observability Stack

The **Observability Stack** provides full support for logs, metrics, and traces. It is designed to adapt to different environments—**Development**, **Staging**, and **Production**—by offering tailored profiles for each.

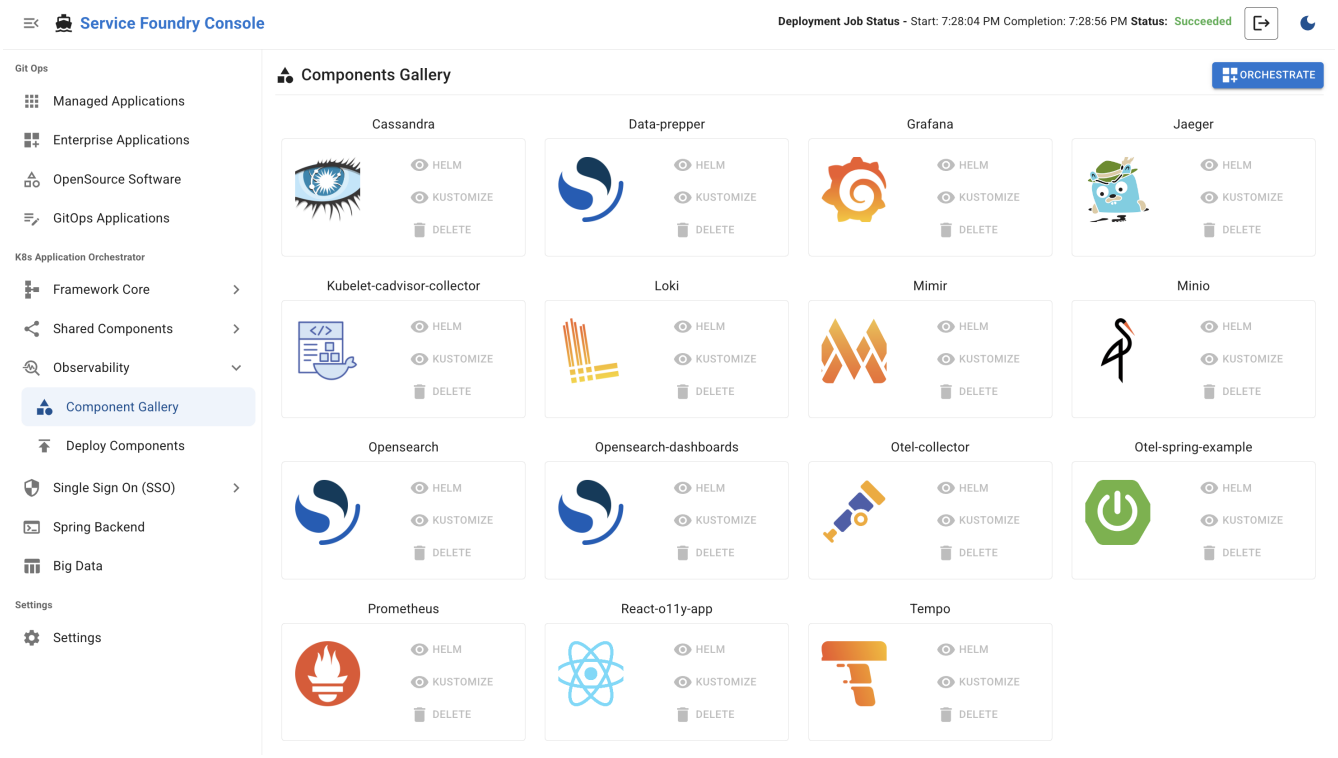


Figure 30. Observability Stack

Component dependencies are considered during orchestration. For example, the Otel Collector may require services like Loki, Tempo, and Prometheus to be deployed together.

Click **Orchestrate** to start the deployment process.

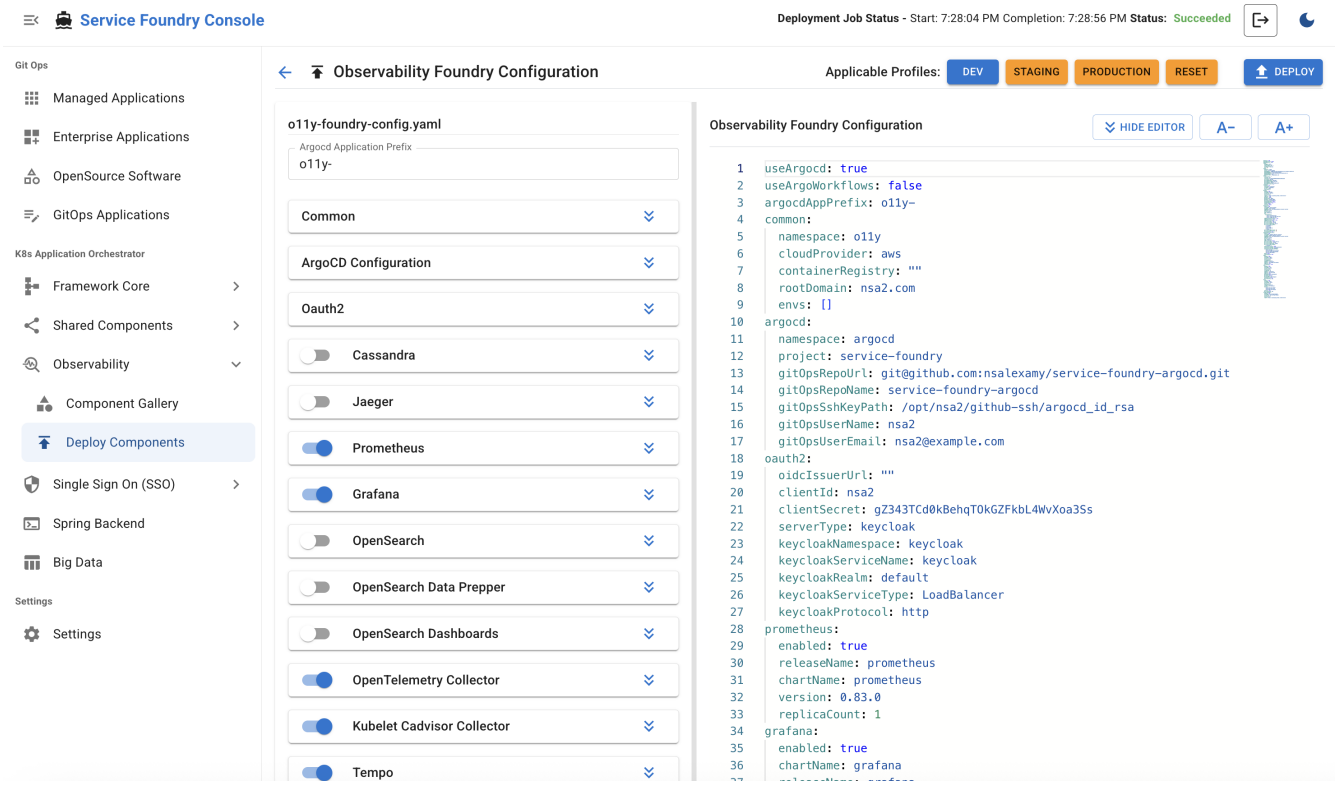


Figure 31. Deploy Observability Stack

Available Profiles

Table 12. Observability Stack Profiles

Profile	Description
Dev Profile	A lightweight configuration including Prometheus, Grafana, Loki, and Otel Collector—ideal for development and local testing.
Staging Profile	Adds components like OpenSearch and S3-compatible object storage to support staging environments and persistent data retention.
Production Profile	A comprehensive observability suite including Jaeger, Cassandra, and high-availability configurations suitable for production workloads.

Each profile triggers manifest generation and GitOps deployment, with resources organized under the observability-apps directory:

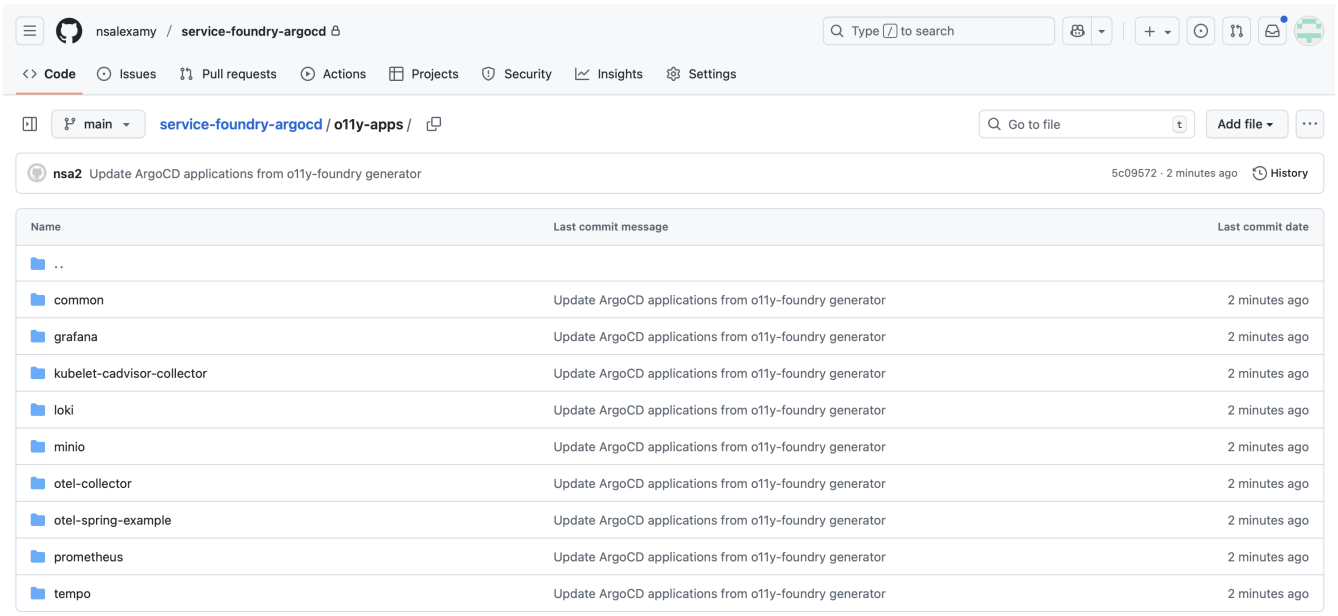


Figure 32. GitOps Repository - Observability Stack

Once deployed, the application appears in the Managed Applications section:

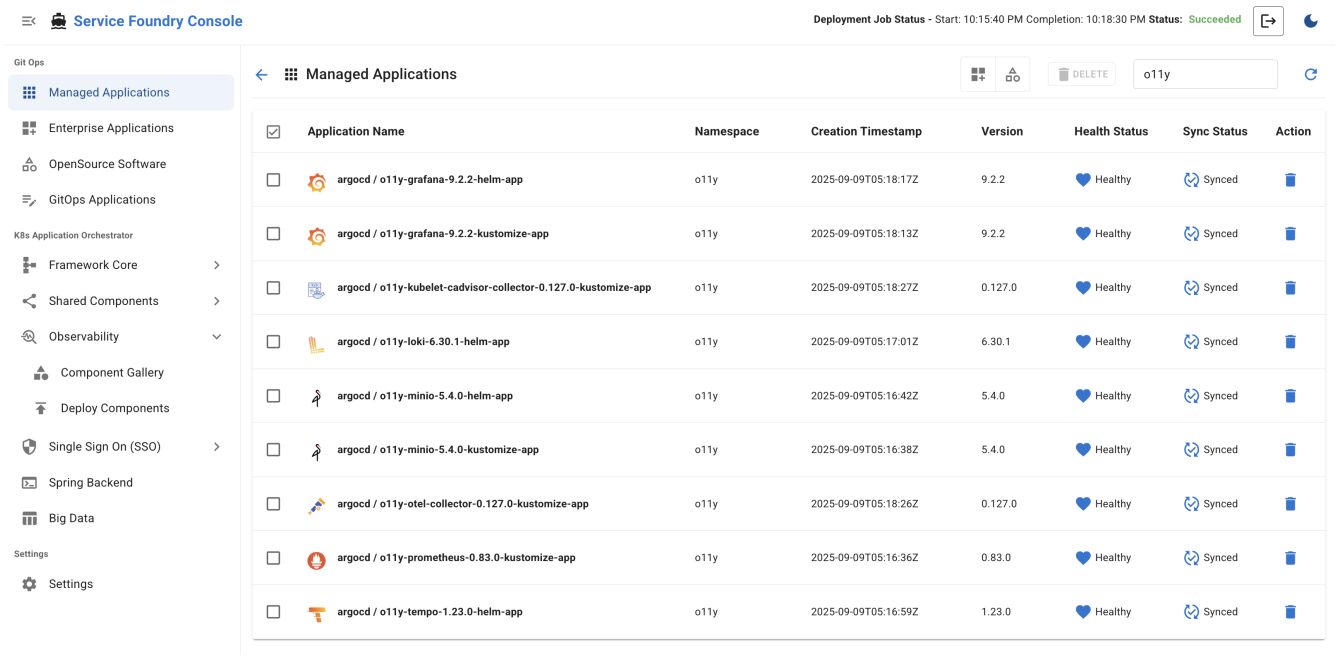


Figure 33. Managed Applications - Observability Stack

Single Sign-On (SSO) Stack

The **SSO Stack** enables authentication and secure access management across the platform. It integrates Keycloak, OAuth2 Proxy, and Traefik to provide seamless SSO for internal and third-party applications.

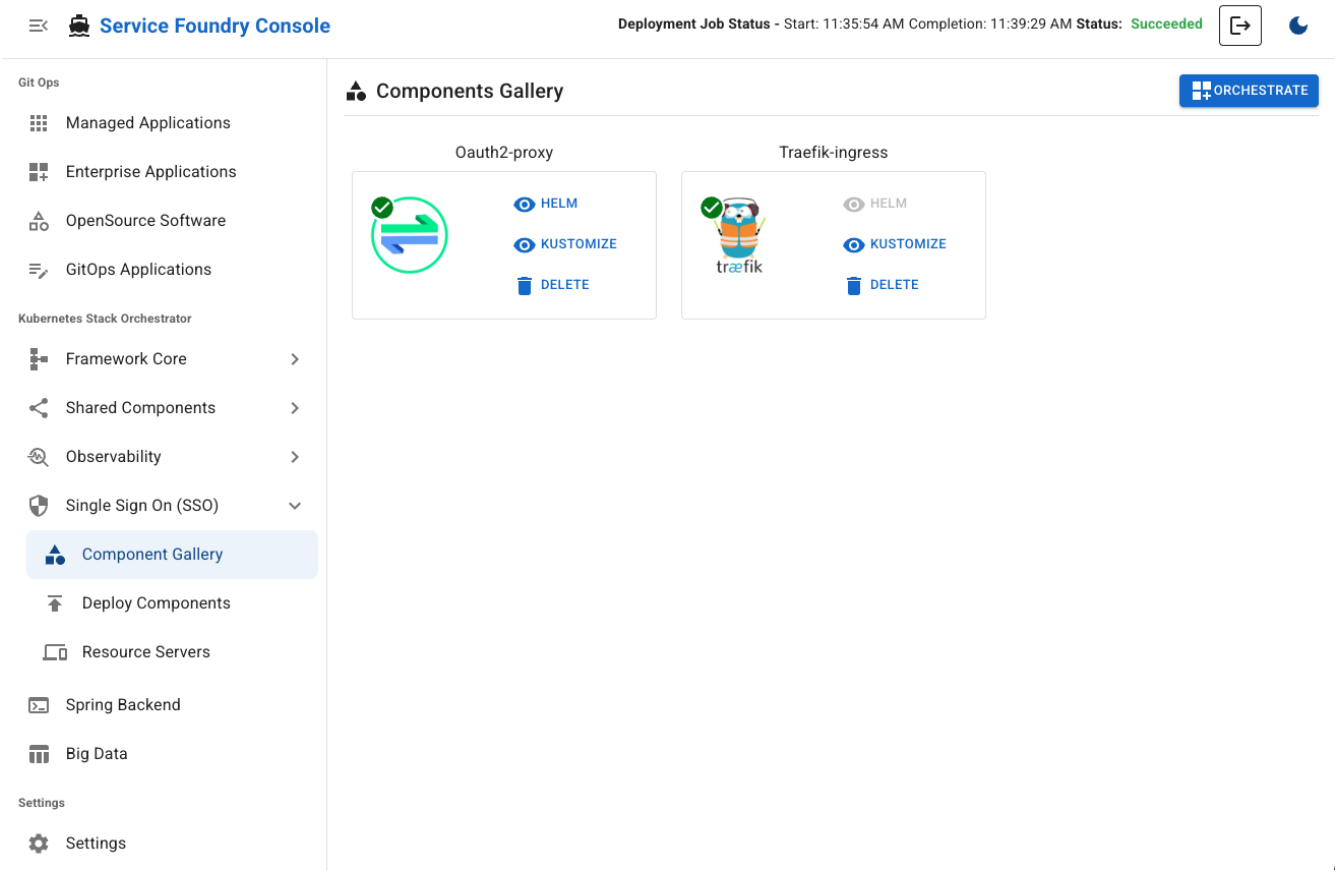


Figure 34. Single Sign-On (SSO) Stack

Traefik IngressRoute and OAuth2 Proxy are preconfigured to secure access to UIs such as Grafana

and the Service Foundry Console.

Deploying the SSO Stack

Click **Orchestrate** to deploy the SSO stack. This creates manifest files, commits them to Git, and provisions the stack via Argo CD.

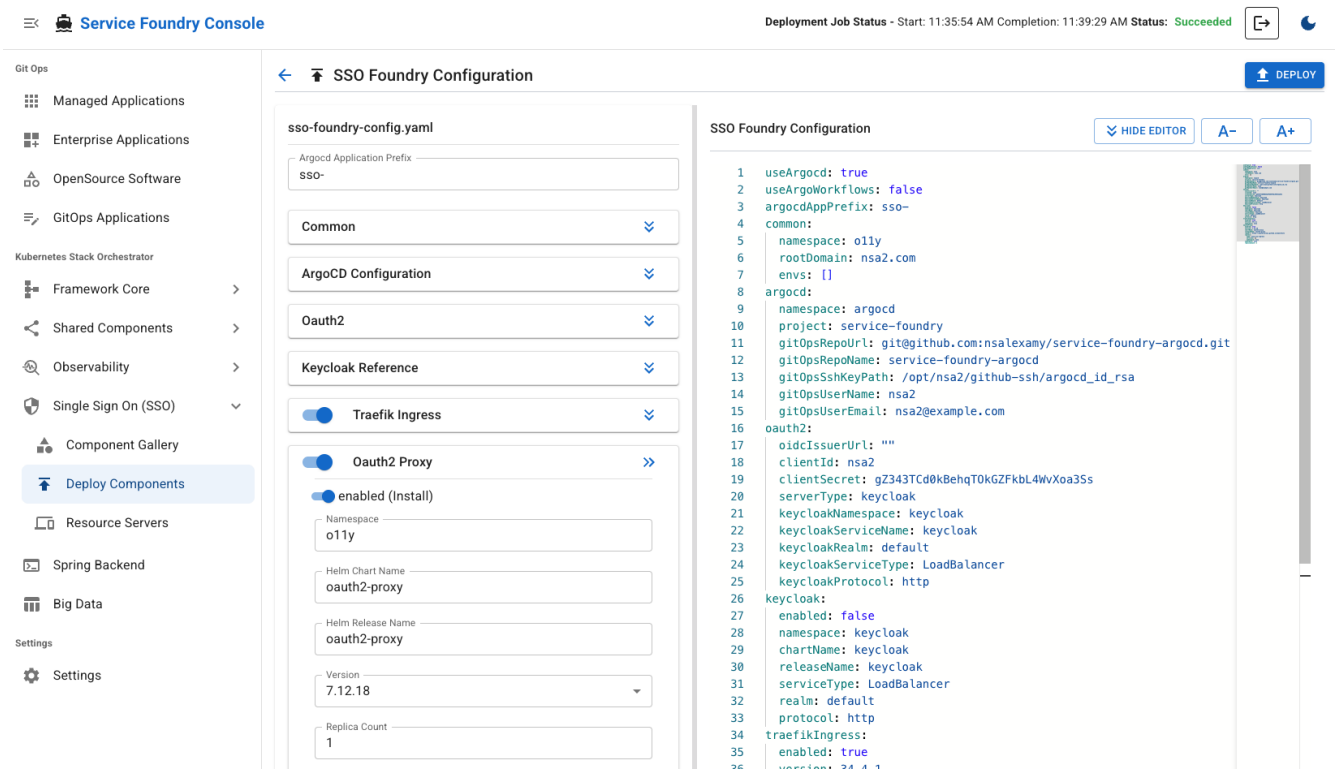


Figure 35. Deploy SSO Stack

SSO Configuration

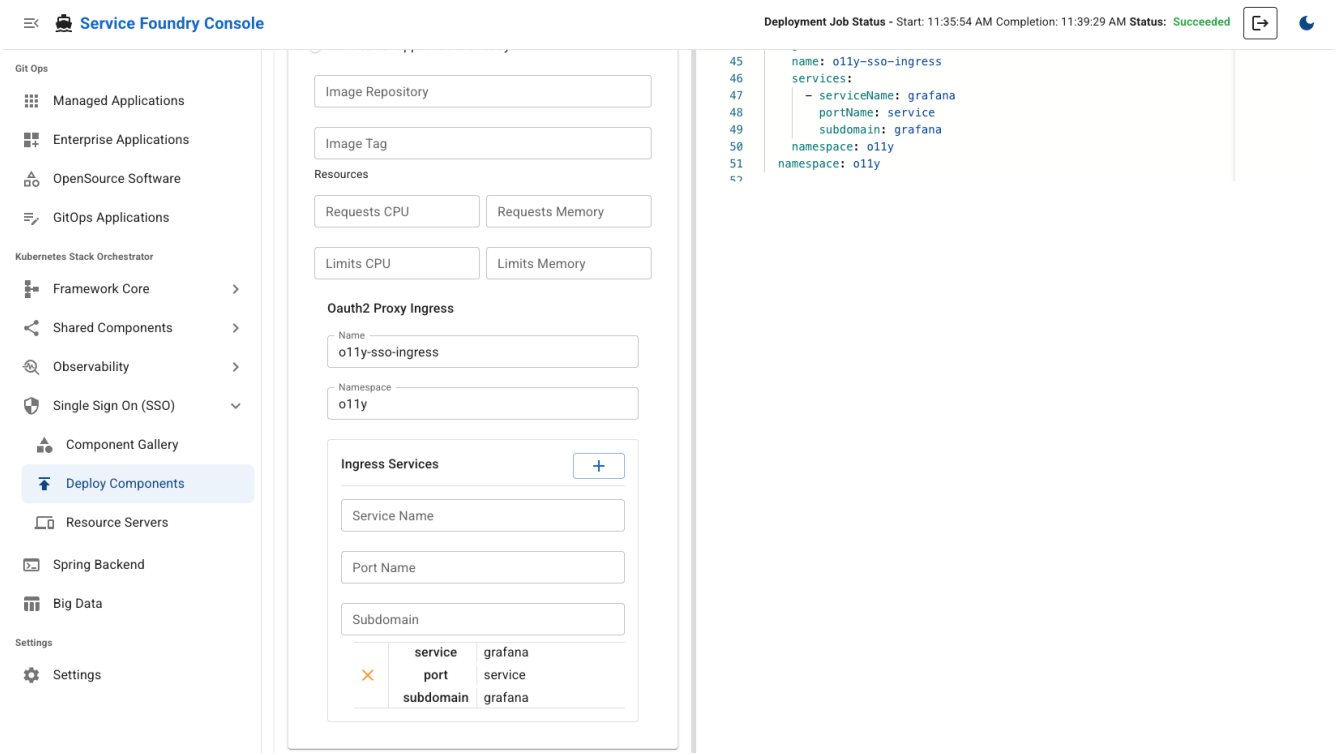


Figure 36. SSO Configuration

Oauth2 Proxy Ingress Form

The IngressRoute configuration form allows you to define routing rules for SSO-protected applications. Key fields include:

Table 13. IngressRoute Configuration Fields

Field name	Description	Example
Name	Unique name for the Ingress resource	o11y-sso-ingress
Namespace	Kubernetes namespace	o11y
Service Name	Kubernetes service to route to	grafana
Port Name	Target port name on the service	service
Subdomain	Subdomain for routing	grafana

A subdomain like grafana with root domain nsa2.com will create a route: http://grafana.nsa2.com.

You can verify the deployed route:

```
$ kubectl -n o11y get ingressroutes o11y-sso-ingress-route -o yaml | yq '.spec'
```

Sample IngressRoute manifest

```
entryPoints:
  - web
routes:
  - kind: Rule
```

```
    match: Host(`grafana.nsa2.com`)
    middlewares:
      - name: cors-headers
      - name: forward-auth
    services:
      - name: grafana
        port: service
```

Once deployed, the SSO application is visible in both the GitOps repository and the Service Foundry Console:

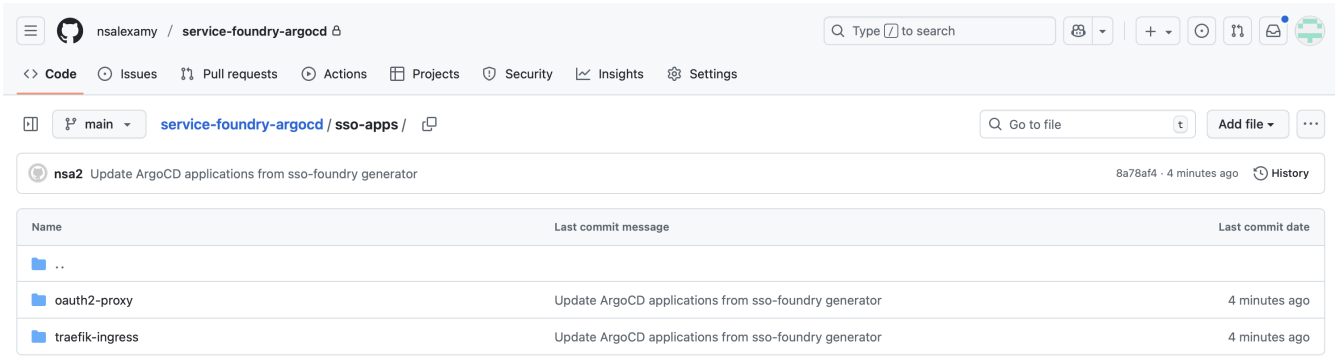


Figure 37. GitOps Repository - SSO Stack

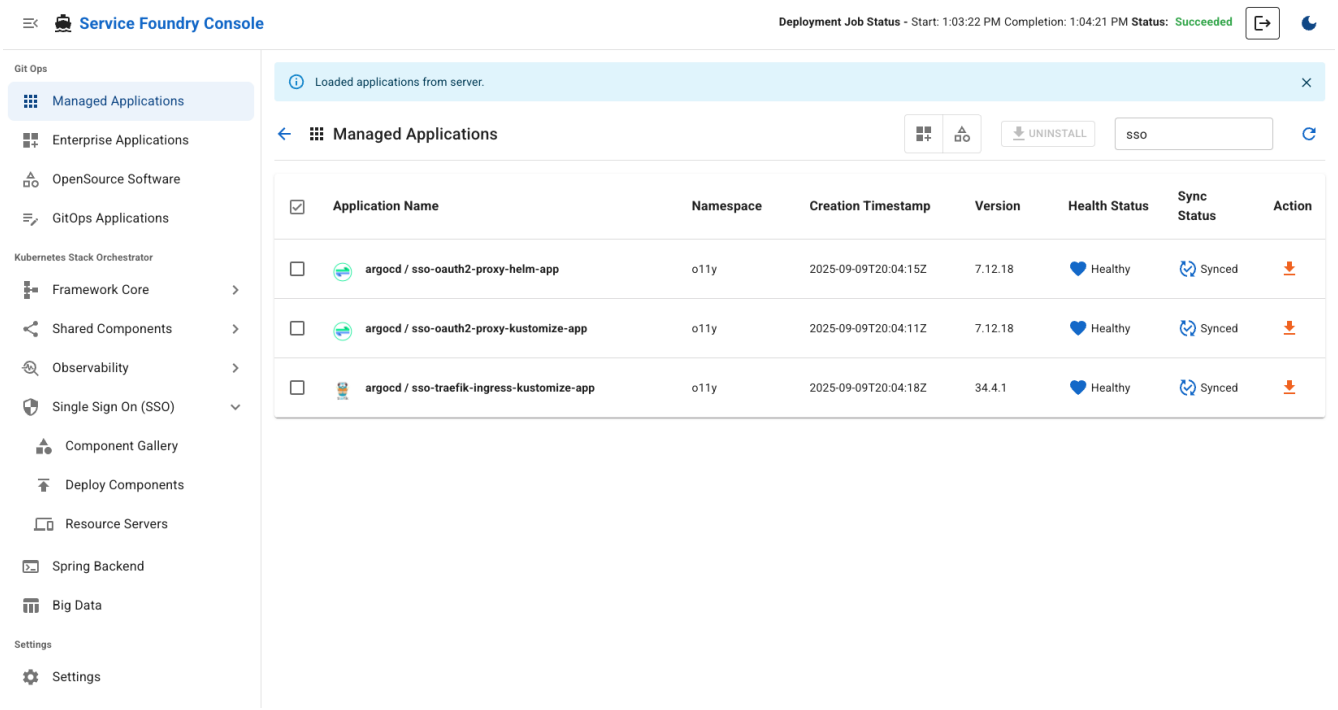


Figure 38. Managed Applications - SSO Stack

Resource Servers

SSO-protected applications, such as Grafana, are listed under Resource Servers. These services inherit the same credentials and session state for unified access control.

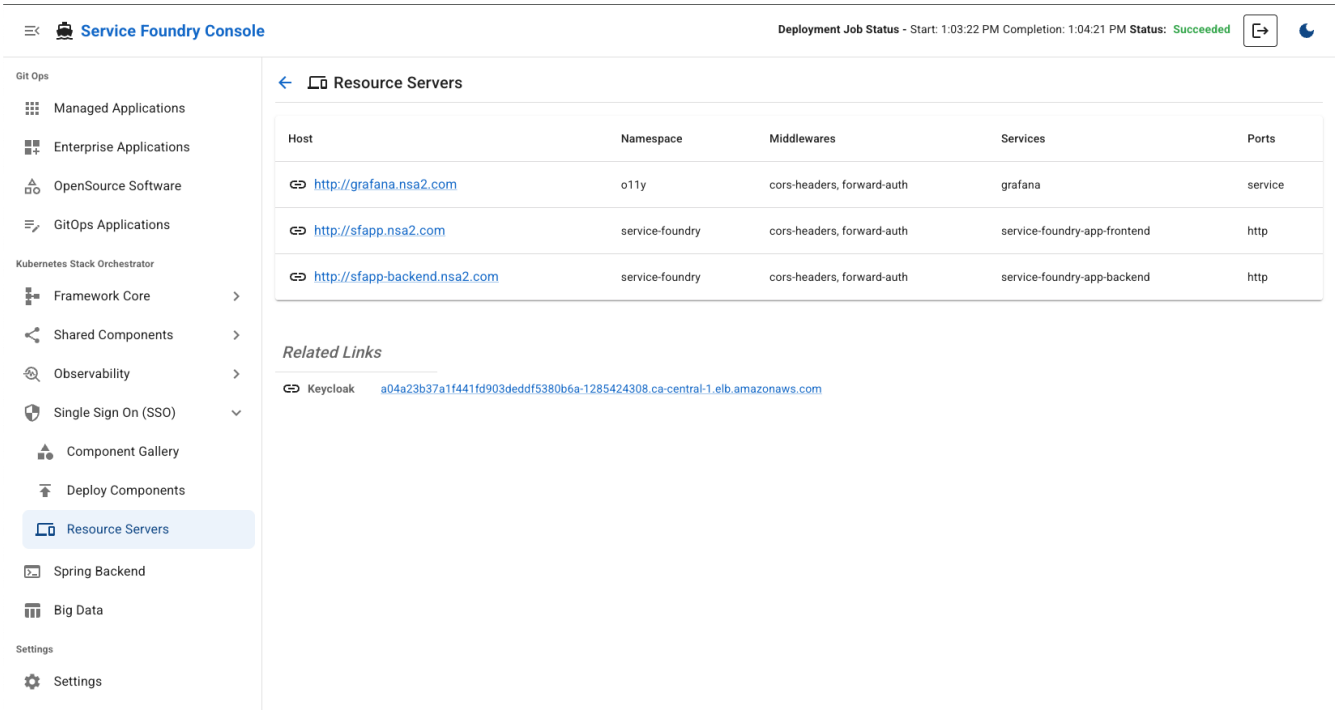


Figure 39. SSO Resource Servers

When accessing Grafana, users are redirected to the Keycloak login page for authentication. After successful login, they are granted access to Grafana without needing to re-enter credentials.

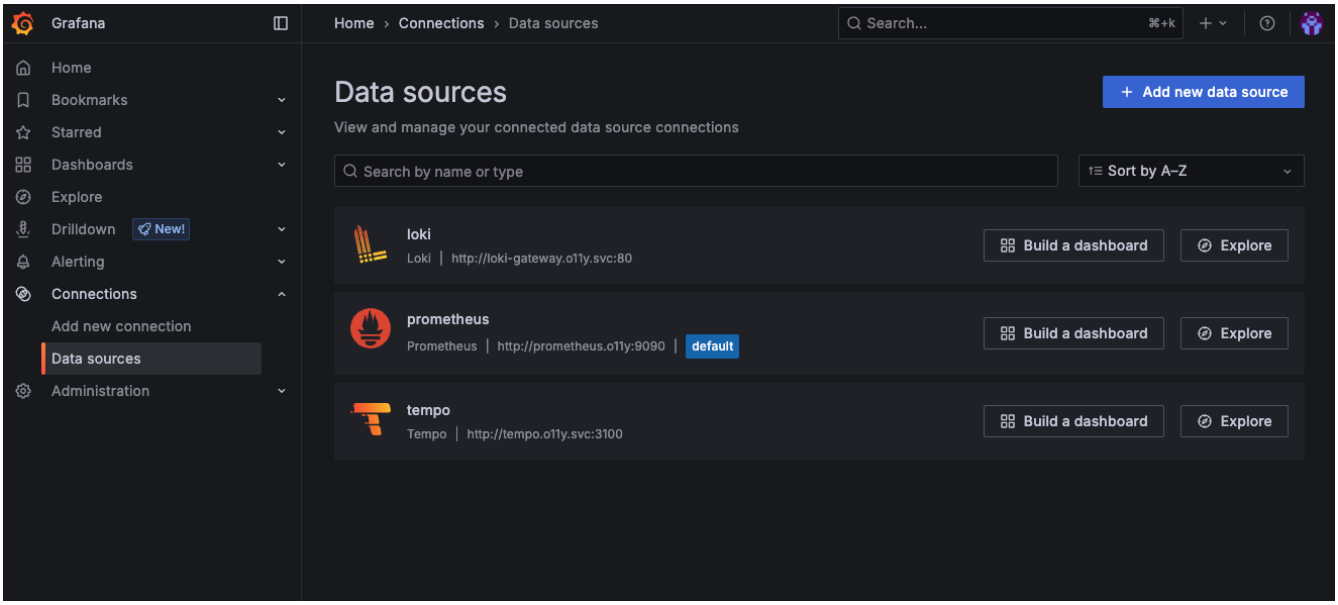


Figure 40. Accessing Grafana with SSO

Spring Backend Stack (Work In Progress)

The upcoming **Spring Backend Stack** is designed to support enterprise-grade Java applications using Spring Boot. It will include runtime dependencies such as PostgreSQL, Redis, RabbitMQ, and configuration tools tailored for microservice development and deployment.

Big Data Stack (Work In Progress)

The **Big Data Stack** will enable scalable data processing using popular open-source technologies. It includes support for Apache Spark, Airflow, OpenSearch, Neo4j, MinIO, and dbt. This stack is intended for teams building data pipelines, graph analytics, or large-scale ETL workflows.