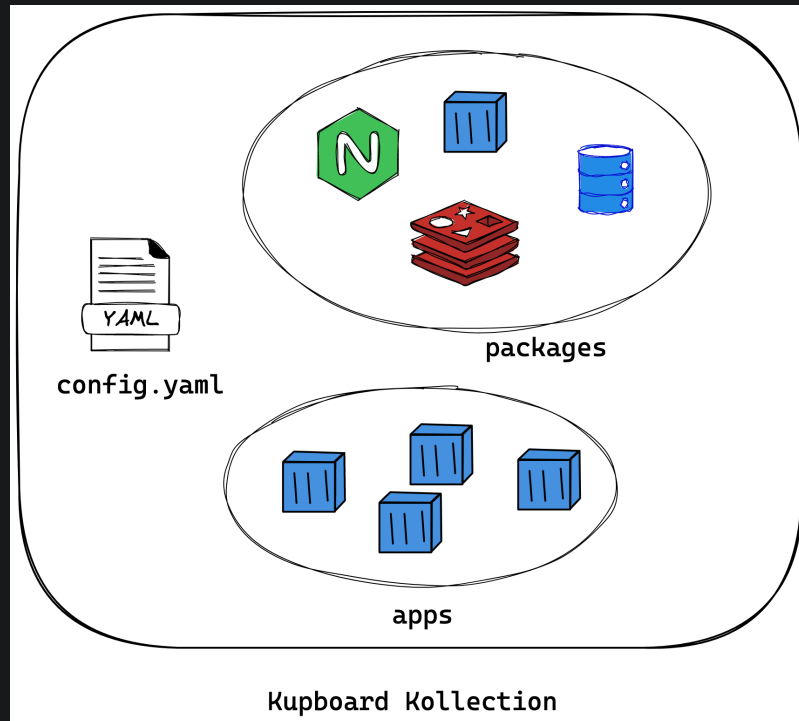
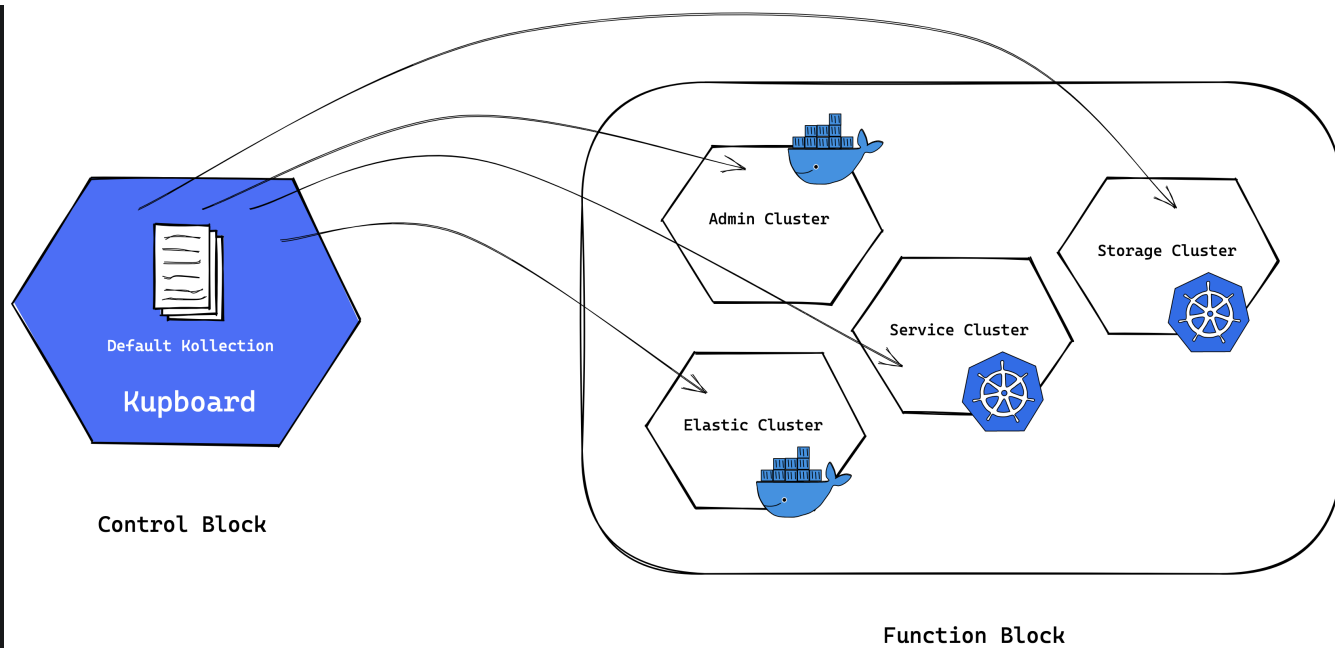


Introduction

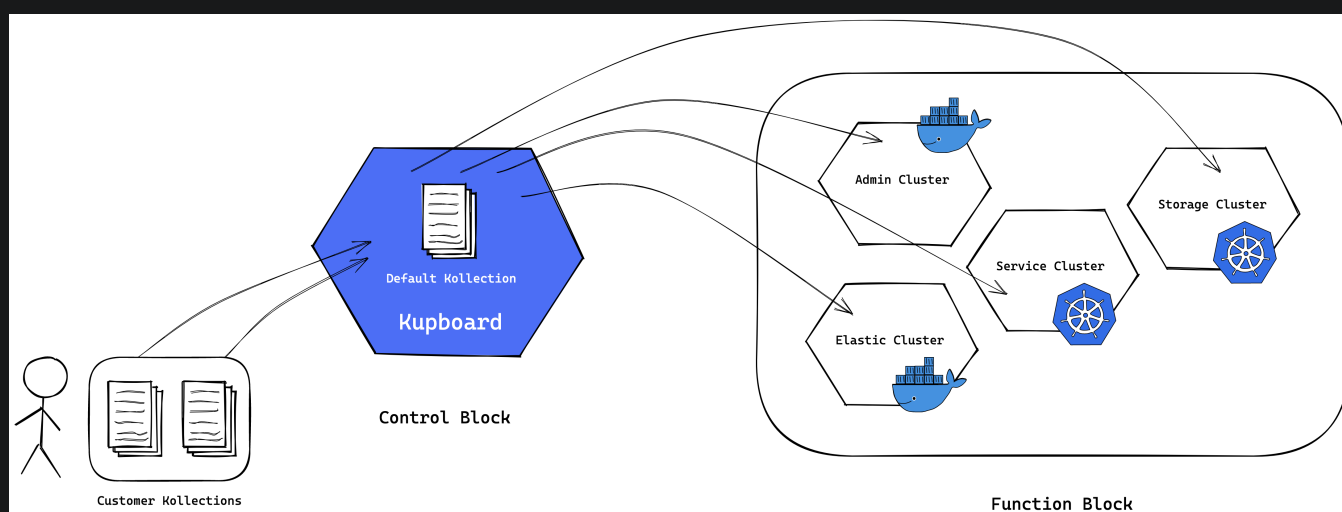
kollection of kupboard is a bundle of software packages and applications to install on the platform. For example, you can include open sources such as Nginx or Prometheus, and you can include applications in a container format that you want to deploy.



Kollection consists of package, application, and config.yaml, and Kupboard contains the default kollection. You can install any package on a given cluster using cli commands. Basic kollection information is available in [Default Package](#).



Users can create their own kollections and deploy them through the kupboard. This is very useful if you need to manage the platform separately by project or service. The packages included in the kollection can be commonly used open source solutions and can also be user-written software. All you need to do is to create a simple Ansible Playbook for installing the package so that Kupboard can read it. For applications, you only need to provide the information you need to build a docker container, regardless of whether the source file is included or not.




A kollection has only one `config.yaml` which defines the information of packages and applications in the kollection. The user can define many actions to manage packages and applications, as well as the parameters required when each action is performed. `config` are automatically loaded by the kupboard, and users can deploy the packages and

applications they want using the actions defined in the config. Collections must be in `data/kollections` with non-duplicate names.



The kollection type can be an archive like `kollection-name.tar.gz`.

 [Edit this page](#)

Config

`config` contains information about the applications and packages included in the kollection.

NOTE

Before `kupboard-0.9.3`, the configuration filename of kollection was `mugset.yaml`. If you have a kollection that includes `mugset.yaml`, simply renaming it to `config.yaml` will work.

Application

Applications included in the kollection are often used when users want to deploy their applications to the service cluster. (For this reason, there are no applications in a default kollection of kupboard.) Include the source directory containing `Dockerfile` in the kollection and define the build and deploy options in the `config`. The kupboard builds the application based on the specifications defined in the config and deploys it to the service cluster.

```
apps:
  - name: myapp
    version: 1.0
    app_port: 80
    service_port: 8080
    exposed_port: 32001
    replica: 2
    envs:
      - name: ENV1
        value: value_of_ENV1
      - name: ENV2
        value: value_of_ENV2
```

- `name` application name
- `version` application version (container tag)
- `app_port` container port on kubernetes

- `service_port` service port on kubernetes
- `exposed_port` service port(node port) on kubernetes. `exposed_port` should set to expose a service using a port between 32000~32767.
- `replica` pod replica
- `envs` env variables

Packages

If you have your own solution or open source package that you need for a service or operation, you can install it using kubernetes. In addition you can define a variety of `action` and `parameter` to handle many situations. Each package must contain at least one Ansible playbook, and can be executed in a variety of ways using the `action` defined in the config.

```
packages:
  - name: mypackage
    vars:
      target_group: gateway
    playbook: nginx/nginx.yaml
    actions:
      - name: deploy
        default: true
        vars:
          state: present
        params:
          - "name=value"
```

- `name` package name
- `vars` playbook variables (for package)
- `actions` action list
 - `name` action name
 - `default` default or not. If `action` not given, the action with `default` true will be executed.
 - `vars` playbook variable (for action)
 - `params` playbook parameter (used for `ansible-playbook`)



Application

Depending on how you compose `Dockerfile`, you can build container images for your application in a variety of ways. You can change only some files you need from the base image, or you can build a container image with just one `Dockerfile`. Also you can use environment variables to create images for different environments.

```
mykollection
├── apps
│   ├── app1
│   │   └── Dockerfile
│   └── app2
│       ├── src
│       └── Dockerfile
```

Build

Applications defined in config can be built using the commands below, and the container image built in this way is automatically pushed to the Harbor registry defined in `kupboard.yaml` (or custom configuration file).

Build customer application

```
$ kupboard kollection app -c <kollection-name> -n <app-name>
```

Build all customer applications

```
$ kupboard kollection app -c <kollection-name>
```


Build all customer applications from all customer kollections

```
$ kupboard kollection app
```

Deployment

If a container image is pushed properly to the Harbor registry, you can use the `deploy` command to deploy the container to the service cluster.

```
$ kupboard deploy -s <application-name>
```

 [Edit this page](#)

Package

The kollection package of the kupboard uses Ansible Playbook. Ansible is a great open source tool and is already used widely for a variety of purposes in many solutions. There are many good written playbooks on the Internet. You can use those playbooks very easily and quickly by defining only the `action` in the config. In the kupboard, we tried to avoid the functional limitations of creating a package's playbook. Users can also use playbooks and actions to create their own great kollections.

```
mykollection
├── packages
│   ├── package1
│   │   └── playbook.yaml
│   └── package2
│       ├── data
│       └── playbook.yaml
```

This is an example to run a kollection command. The `-c <kollection-name>` option allows you to use packages in a specific kollection.

```
$ kupboard kollection package -n <package-name> [-a <action-name>]
```

kupboard-kollection-nginx

Let's check an example [kupboard-kollection-nginx](#).

In `config.yaml` a package named `nginx` is defined and there are two actions, `deploy` and `delete`. Only one playbook named `nginx.yaml` is used and the variable `target_group` can be used in the playbook. And the variable `state` is used when actions are executed, and its value can be `started` and `absent`.

```
version: 1.0
metadata:
  name: Nginx
packages:
  - name: nginx
```

```

playbook: nginx/nginx.yaml
vars:
  target_group: gateway
actions:
  - name: deploy
    default: true
    vars:
      state: started
    params:
  - name: delete
    vars:
      state: absent
    params:

```

The Nginx package uses the 'nginx:1.19.10' container. It uses `target_group` and `state` defined in the config, and the `state` value is used for container deployment.

```

- name: Nginx Package
  hosts: "{{ target_group }}"
  become: no
  vars:
    release_state: "{{ state | default('started') }}"
  tasks:
    - name: "{% if release_state == 'started' %} Deploy Nginx {% else
      %} Remove Nginx {% endif %}"
      docker_container:
        name: nginx
        image: nginx:1.19.10
        state: "{{ release_state }}"
        restart_policy: always
        container_default_behavior: compatibility
        restart: yes
        ports:
          - 80:80

```

Now you can see that you need to use the `-a deploy` option to install nginx in the gateway cluster when using Nginx Package, and you can delete nginx installed in the gateway cluster with the `-a delete` option.

```


$ kupboard kollection package -c kupboard-kollection-nginx -n nginx -
a <deploy|delete>

```

Default Packages

The default kollection of kupboard includes many packages for various open source.

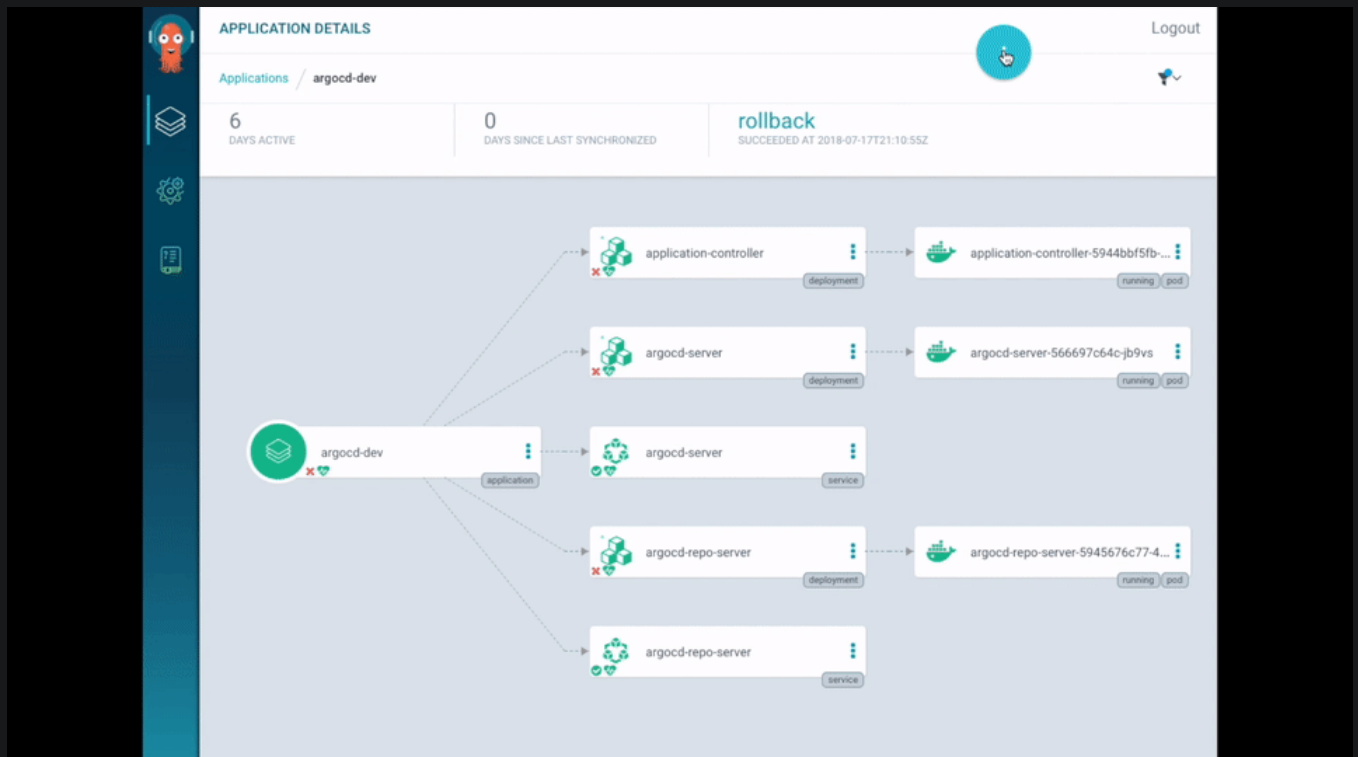
- [argocd](#)
- [dashboard](#)
- [elastic](#)
- [fluentd](#)
- [harbor](#)
- [influxdb](#)
- [istio](#)
- [kafka](#)
- [keycloak](#)
- [minio](#)
- [mongodb](#)
- [mysql](#)
- [nginx](#)
- [prometheus](#)
- [redis](#)

 [Edit this page](#)

ArgoCD

ArgoCD is a GitOps CD(Continuous Delivery) tool for applications deployed on Kubernetes. It provides various functionality for deploying and managing applications based on manifests or Helm charts stored in github repos.

Reference: <https://argoproj.github.io/argo-cd>



Package Deployment

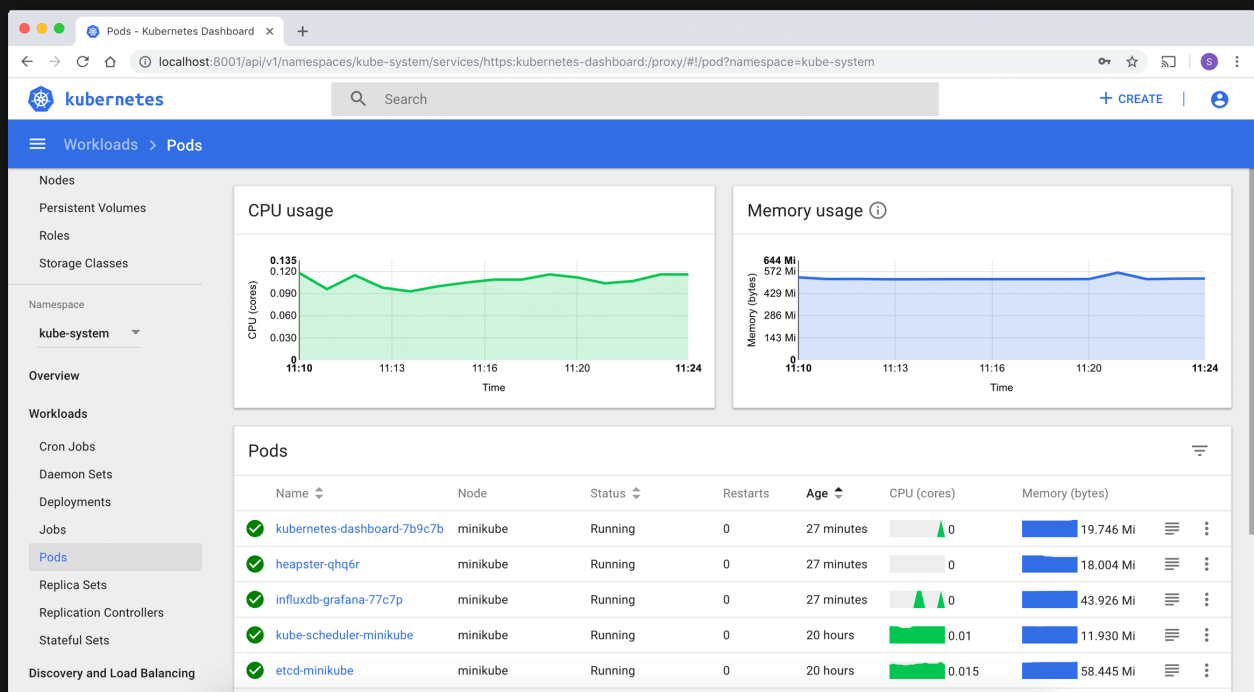
```
$ kubectl kubernetes package -n argocd -a <action>
```

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| argocd | service | deploy | true |
| | | delete | |

Dashboard (k8s)

Dashboard is a web-based Kubernetes user interface. You can use Dashboard to deploy containerized applications to a Kubernetes cluster, troubleshoot your containerized application, and manage the cluster resources. You can use Dashboard to get an overview of applications running on your cluster, as well as for creating or modifying individual Kubernetes resources (such as Deployments, Jobs, DaemonSets, etc). For example, you can scale a Deployment, initiate a rolling update, restart a pod or deploy new applications using a deploy wizard.

Reference: <https://kubernetes.io/ko/docs/tasks/access-application-cluster/web-ui-dashboard>




Package Deployment

```
$ kubectl kubernetes package -n dashboard -a <action>
```

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
|--------------|---------|--------|---------|

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| dashboard | service | deploy | true |
| | | delete | |

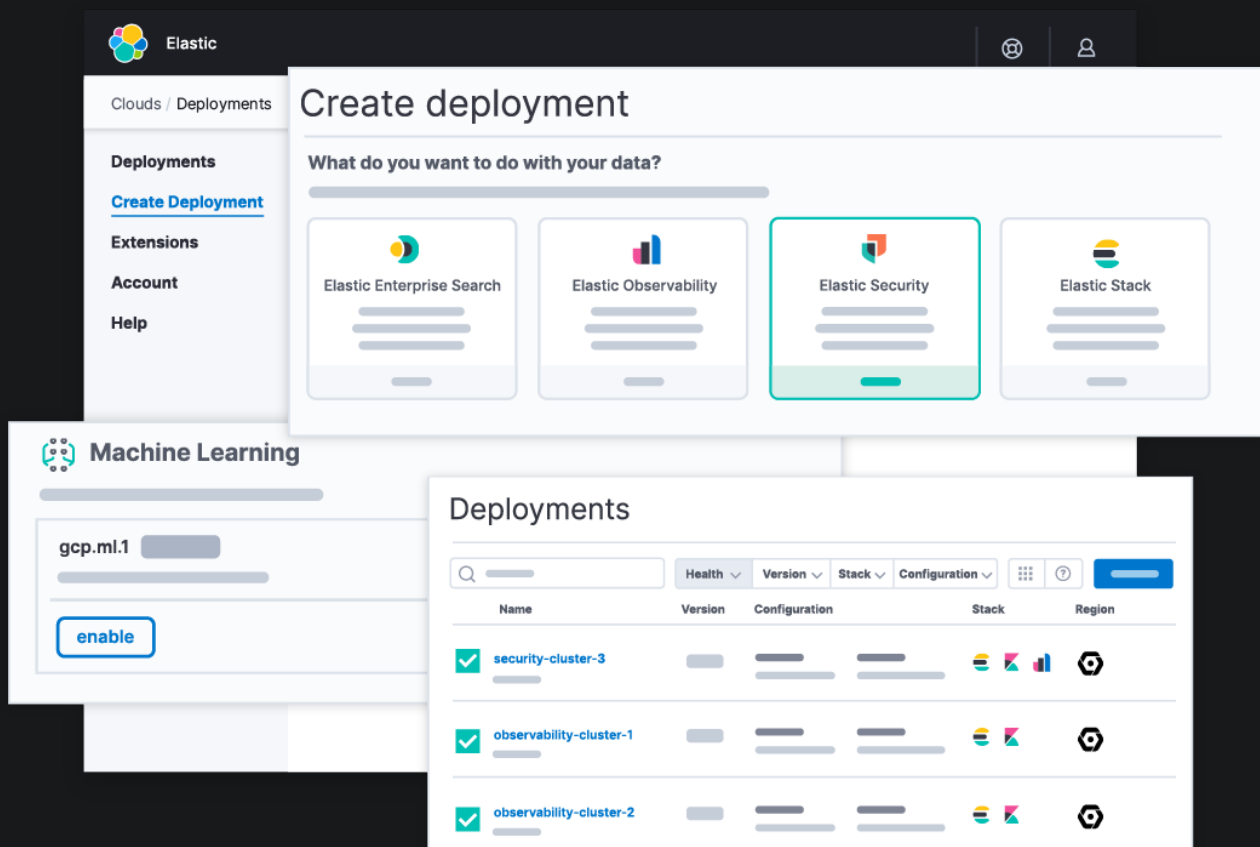
 [Edit this page](#)

Elasticsearch

Reference: <https://www.elastic.co/>

Elasticsearch

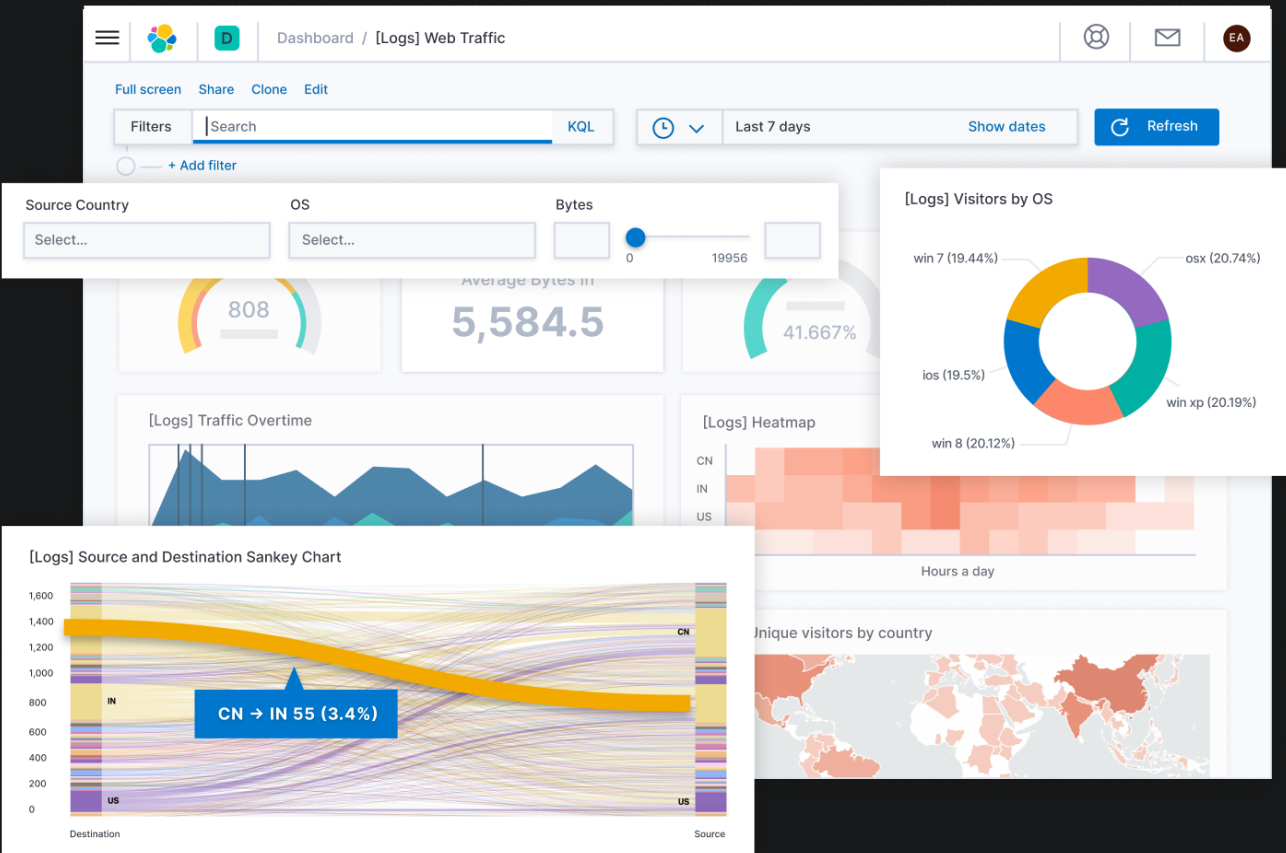
Whether you're looking for actions from a specific IP address, analyzing a spike in transaction requests, or hunting for a taco spot in a one-mile radius, the problems we're all trying to solve with data boil down to search. Elasticsearch lets you store, search, and analyze with ease at scale.



Kibana

Start exploring your data with stunning visualizations in Kibana, from waffle charts and heatmaps to time series analysis and beyond. Use preconfigured dashboards for your

diverse data sources, create live presentations to highlight KPIs, and manage your deployment in a single UI.



Package Deployment

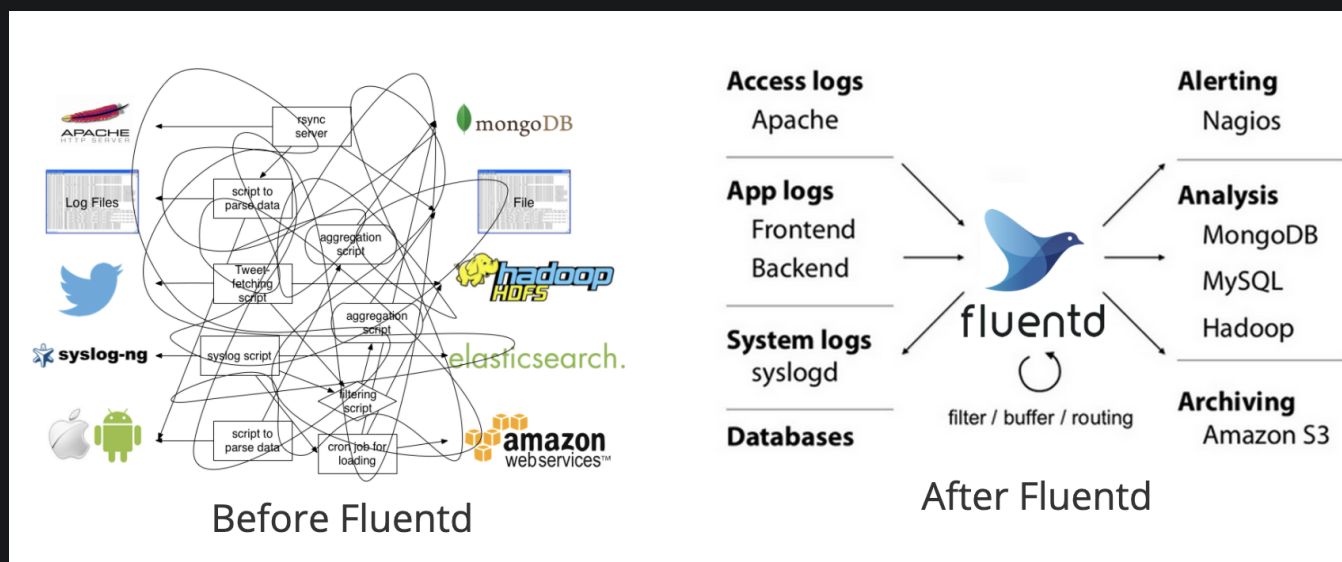
```
$ kupboard kollection package -n elastic -a <action>
```

| Pakage Name | Cluster | Action | Default |
|-------------|---------|--------|---------|
| elastic | elastic | deploy | true |
| | | delete | |

Fluentd

Fluentd is an open source data collector, which lets you unify the data kollection and consumption for a better use and understanding of data.

Reference: <https://www.fluentd.org/>



Package Deployment

```
$ kupboard kollection package -n fluentd -a <action>
```

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| fluentd | service | deploy | true |
| | | delete | |

[Edit this page](#)

Harbor

Harbor is an open source registry that secures artifacts with policies and role-based access control, ensures images are scanned and free from vulnerabilities, and signs images as trusted. Harbor, a CNCF Graduated project, delivers compliance, performance, and interoperability to help you consistently and securely manage artifacts across cloud native compute platforms like Kubernetes and Docker.

Reference: <https://goharbor.io/>

Install Harbor

To deploy your application to the service cluster, you must install Docker Contains Registry. The kubernetes uses **Harbor** as the default registry.

Kubernetes provides a kubernetes package to install the harbor registry.

```
$ kubernetes kubernetes package -n harbor
```

NOTE

Before install the harbor registry, ssl certification should be prepared in **data/certs** and **harbor.mycompany.com** is updated in your DNS management system to point to **admin-node1**.

Package Deployment

```
$ kubernetes kubernetes package -n harbor -a <action>
```

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| harbor | admin | deploy | true |

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| | | delete | |

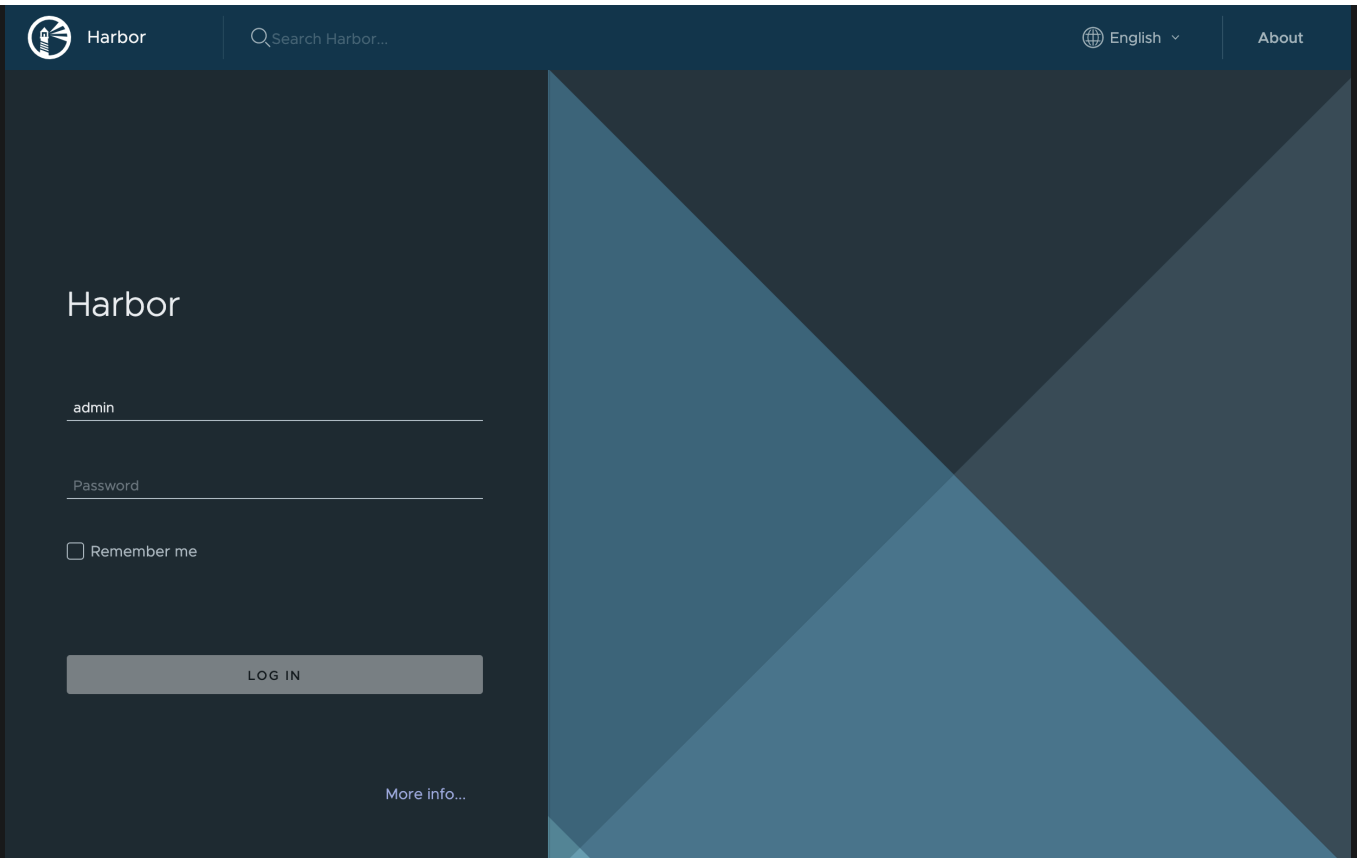
Setting

First, you need to access `https://harbor.mycompany.com` to create an user and project.

```
## Project Information
##
kube_registry_username: kupboard
kube_registry_password: Kupboard1234
kube_registry_email: username@email.com
kube_registry_domain: harbor.mycompany.com
kube_registry_url: harbor.mycompany.com/myproject

## Secrets
##
harbor_admin_password: kupboard
```

Login with the admin account. Its default username is `admin` and password is the value of `harbor_admin_password`.



The image shows the Harbor login page. At the top, there is a dark blue header with the Harbor logo, a search bar, and links for 'English' and 'About'. The main content area has a dark background with a large blue abstract graphic on the right. On the left, there is a login form with the title 'Harbor'. The form includes a username field with 'admin' entered, a password field, a 'Remember me' checkbox, and a 'LOG IN' button. A 'More info...' link is located below the login button.

Harbor

admin

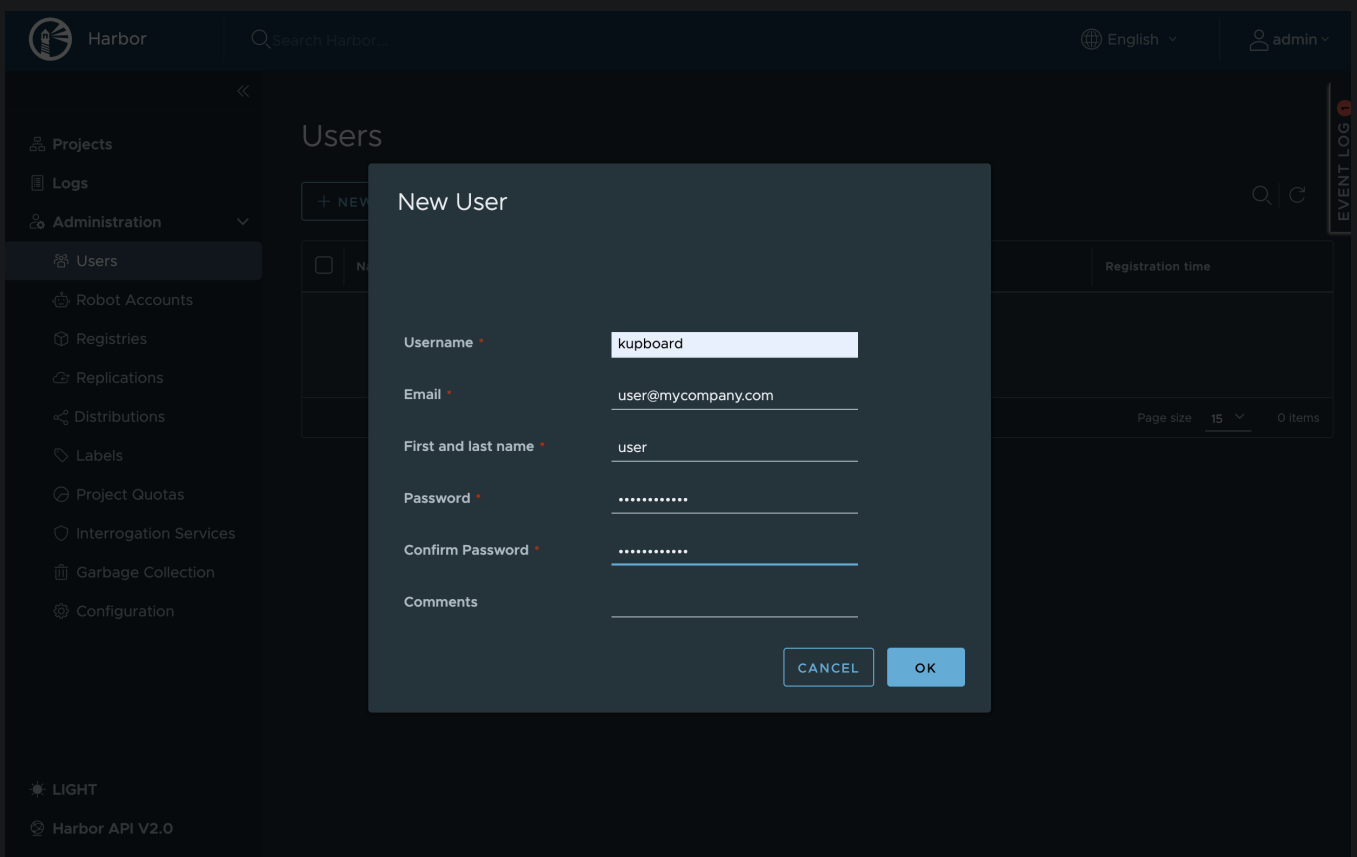
Password

☐ Remember me

LOG IN

More info...

Then create a new user with `kube_registry_username` and `kube_registry_password`.



The image shows the Harbor 'Users' management page. A 'New User' modal is open in the center. The modal contains fields for Username, Email, First and last name, Password, Confirm Password, and Comments. The Username field is filled with 'kupboard', the Email with 'user@mycompany.com', and the First and last name with 'user'. The Password and Confirm Password fields are masked with dots. The modal has 'CANCEL' and 'OK' buttons at the bottom right. In the background, the 'Users' page is visible, showing a table with columns for Username, Email, and Registration time. The left sidebar contains navigation links for Projects, Logs, Administration, and Users. The bottom of the page shows a 'LIGHT' theme indicator and 'Harbor API V2.0'.

Harbor

Search Harbor...

English

admin

Users

+ NEW

Registration time

Page size 15 0 items

NEW USER

Username *

kupboard

Email *

user@mycompany.com

First and last name *

user

Password *

Confirm Password *

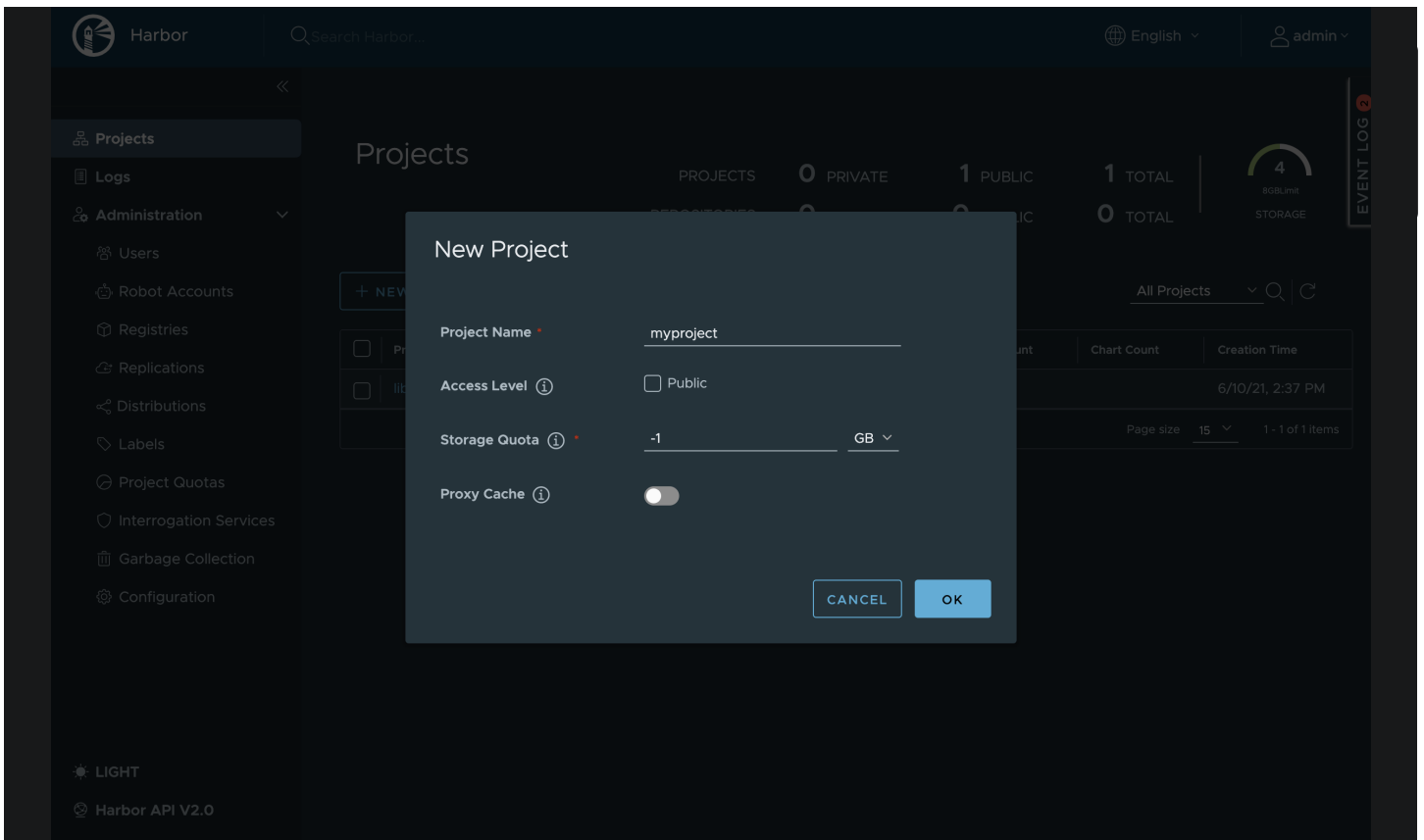
Comments

CANCEL OK

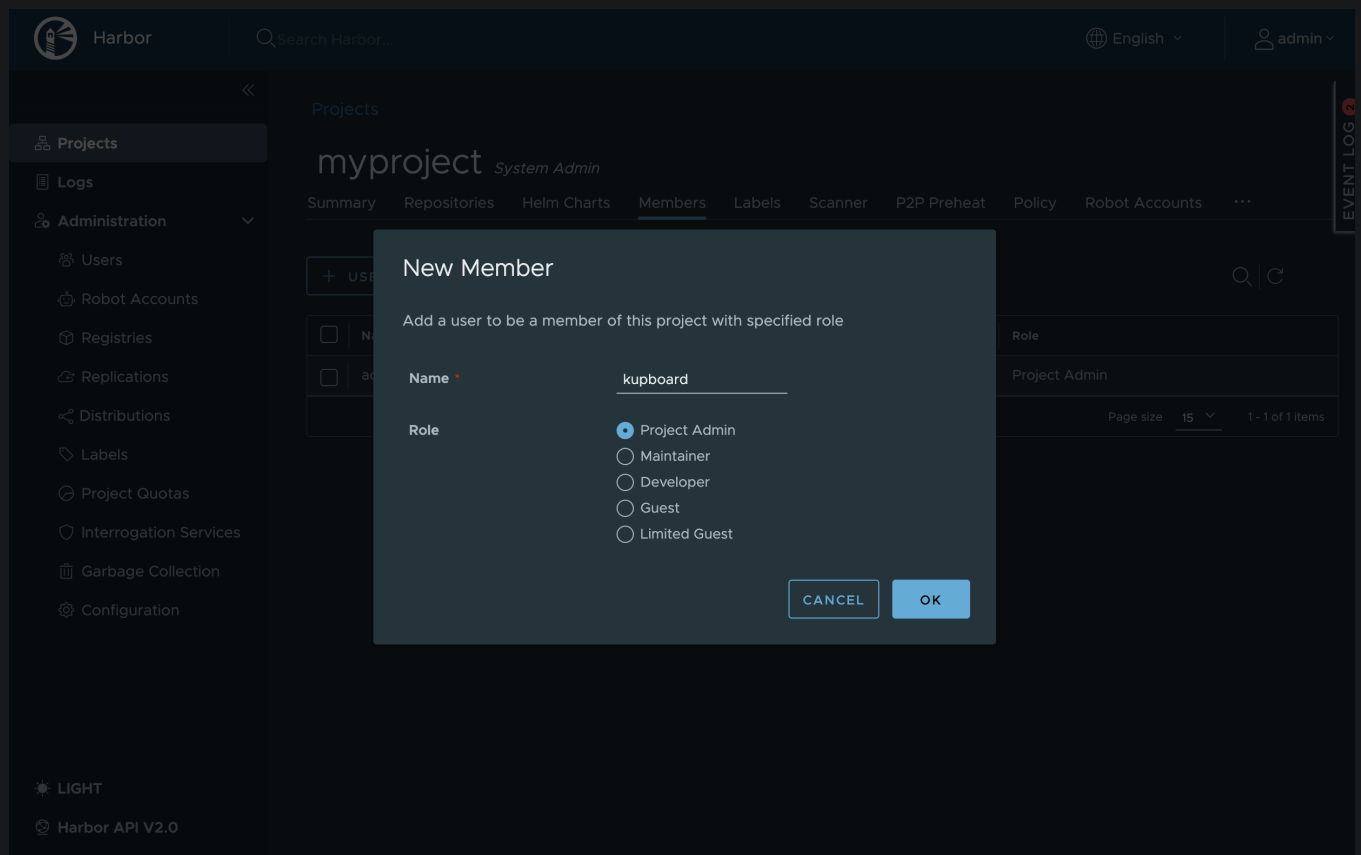
LIGHT

Harbor API V2.0

Create a new project. In this example, the project name should be `myproject`.



Add a new user to the project.



Influxdb


InfluxDB is a time series platform InfluxDB empowers developers to build IoT, analytics and monitoring software. It is purpose-built to handle the massive volumes and countless sources of time-stamped data produced by sensors, applications and infrastructure.

Reference: <https://www.influxdata.com/>

Package Deployment

```
$ kupboard kollection package -n influxdb -a <action>
```

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| influxdb | service | deploy | true |
| | | delete | |

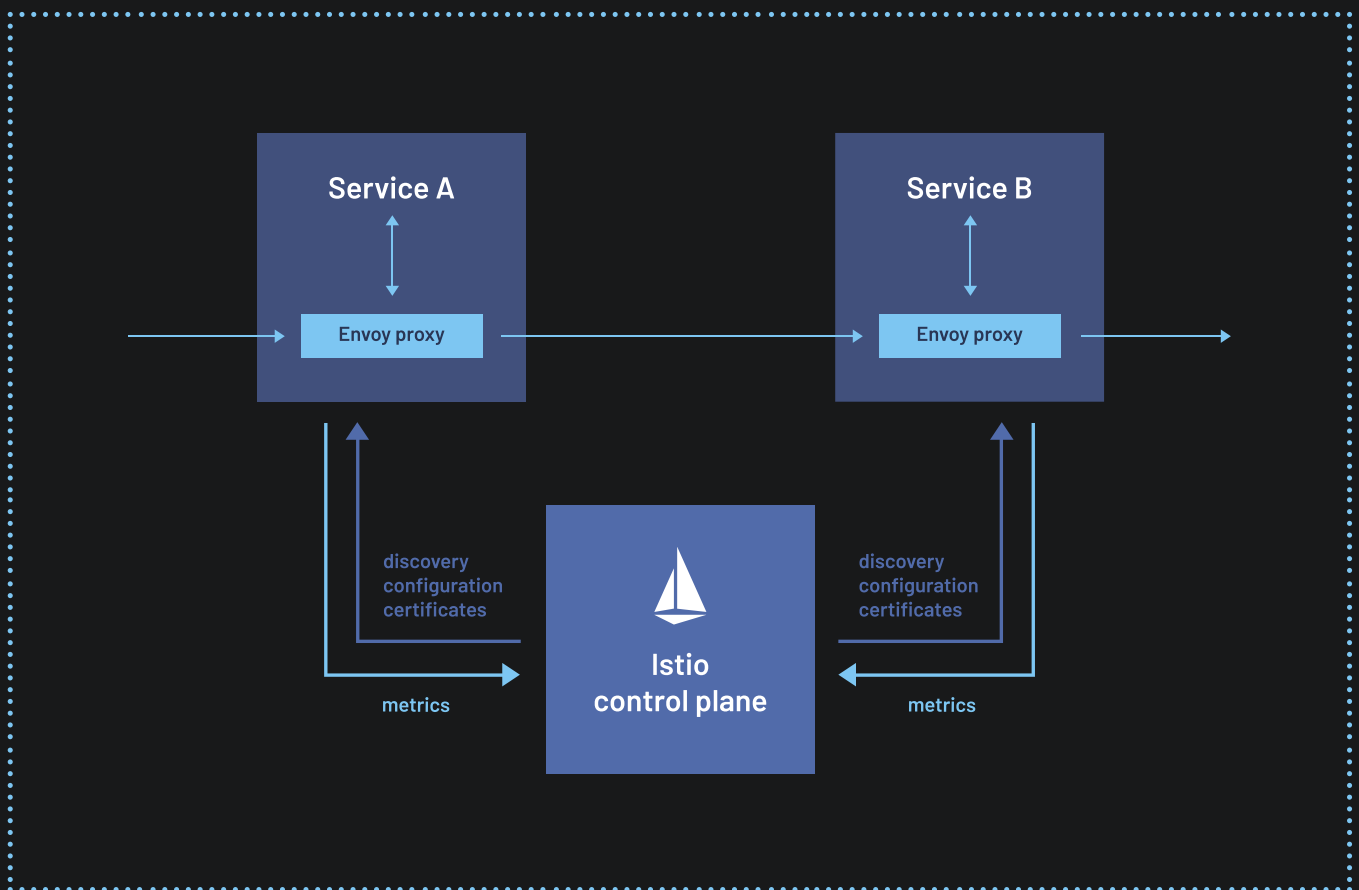
 [Edit this page](#)

Istio

Istio is an open source service mesh that layers transparently onto existing distributed applications. Istio's powerful features provide a uniform and more efficient way to secure, connect, and monitor services. Istio is the path to load balancing, service-to-service authentication, and monitoring – with few or no service code changes. Its powerful control plane brings vital features, including:

- Secure service-to-service communication in a cluster with TLS encryption, strong identity-based authentication and authorization
- Automatic load balancing for HTTP, gRPC, WebSocket, and TCP traffic
- Fine-grained control of traffic behavior with rich routing rules, retries, failovers, and fault injection
- A pluggable policy layer and configuration API supporting access controls, rate limits and quotas
- Automatic metrics, logs, and traces for all traffic within a cluster, including cluster ingress and egress


Reference: <https://istio.io/>



Package Deployment

```
$ kupboard kollection package -n istio -a <action>
```

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| istio | service | deploy | true |
| | | delete | |

 [Edit this page](#)

Kafka

Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

Reference: <https://kafka.apache.org/>

AKHQ (Kafka GUI)

Kafka GUI for Apache Kafka to manage topics, topics data, consumers group, schema registry, connect and more...

Reference: <https://akhq.io/>

Package Deployment

Kafka

```
$ kupboard kollection package -n kafka -a <action>
```


| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| kafka | service | deploy | true |
| | | delete | |

AKHQ

```
$ kupboard kollection package -n kafka-akhq -a <action>
```

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
|--------------|---------|--------|---------|

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| kafka-akhq | service | deploy | true |
| | | delete | |

 [Edit this page](#)

Keycloak

Keycloak is an Open Source Identity and Access Management solution for modern Applications and Services.

Keycloak provides the flexibility to export and import configurations easily, using a single view to manage everything. Together, these technologies let you integrate front-end, mobile, and monolithic applications into a microservice architecture. In this article, we discuss the core concepts and features of Keycloak and its application integration mechanisms. You will find links to implementation details near the end.

Reference: <https://www.keycloak.org/>

The screenshot shows the Keycloak Master console interface. On the left is a dark sidebar with a navigation menu. The top of the sidebar has a 'Master' dropdown and a 'Configure' section. Under 'Configure', 'Realm Settings' is highlighted with a blue bar. Below it are 'Clients', 'Client Scopes', 'Roles', 'Identity Providers', 'User Federation', and 'Authentication'. The 'Manage' section includes 'Groups', 'Users', and 'Sessions'. The main content area is titled 'Master' with a trash icon. It has tabs for 'General' (selected), 'Login', 'Keys', 'Email', 'Themes', 'Cache', 'Tokens', 'Client Registration', and 'Security Defenses'. The 'General' tab contains the following settings: 'Name' (required, value 'master'), 'Display name' (value 'master'), 'HTML Display name' (value 'master realm'), 'Frontend URL' (empty), 'Enabled' (toggle set to 'ON'), 'User-Managed Access' (toggle set to 'OFF'), and 'Endpoints' (a list containing 'OpenID Endpoint Configuration' and 'SAML 2.0 Identity Provider Metadata'). At the bottom are 'Save' and 'Cancel' buttons.

Package Deployment

```
$ kupboard kollection package -n keycloak -a <action>
```

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| keycloak | service | deploy | true |
| | | delete | |

MinIO


MinIO is a High Performance Object Storage released under GNU Affero General Public License v3.0. It is API compatible with Amazon S3 cloud storage service. Use MinIO to build high performance infrastructure for machine learning, analytics and application data workloads.

Reference: <https://min.io/>

Package Deployment

```
$ kupboard kollection package -n minio -a <action>
```

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| minio | service | deploy | true |
| | | delete | |

 [Edit this page](#)

MongoDB


MongoDB is a document database, which means it stores data in JSON-like documents. We believe this is the most natural way to think about data, and is much more expressive and powerful than the traditional row/column model.

Reference: <https://www.mongodb.com/>

Package Deployment

```
$ kupboard kollection package -n mongodb -a <action>
```

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| mongodb | service | deploy | true |
| | | delete | |

 [Edit this page](#)

MySQL


MySQL is the world's most popular open source database. Whether you are a fast growing web property, technology ISV or large enterprise, MySQL can cost-effectively help you deliver high performance, scalable database applications.

Reference: <https://www.mysql.com/>

Package Deployment

```
$ kupboard kollection package -n mysql -a <action>
```

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| mysql | service | deploy | true |
| | | delete | |

 [Edit this page](#)

Nginx


NGINX accelerates content and application delivery, improves security, facilitates availability and scalability for the busiest web sites on the Internet.

Reference: <https://www.nginx.com/>

Package Deployment

```
$ kupboard kollection package -n nginx -a <action>
```

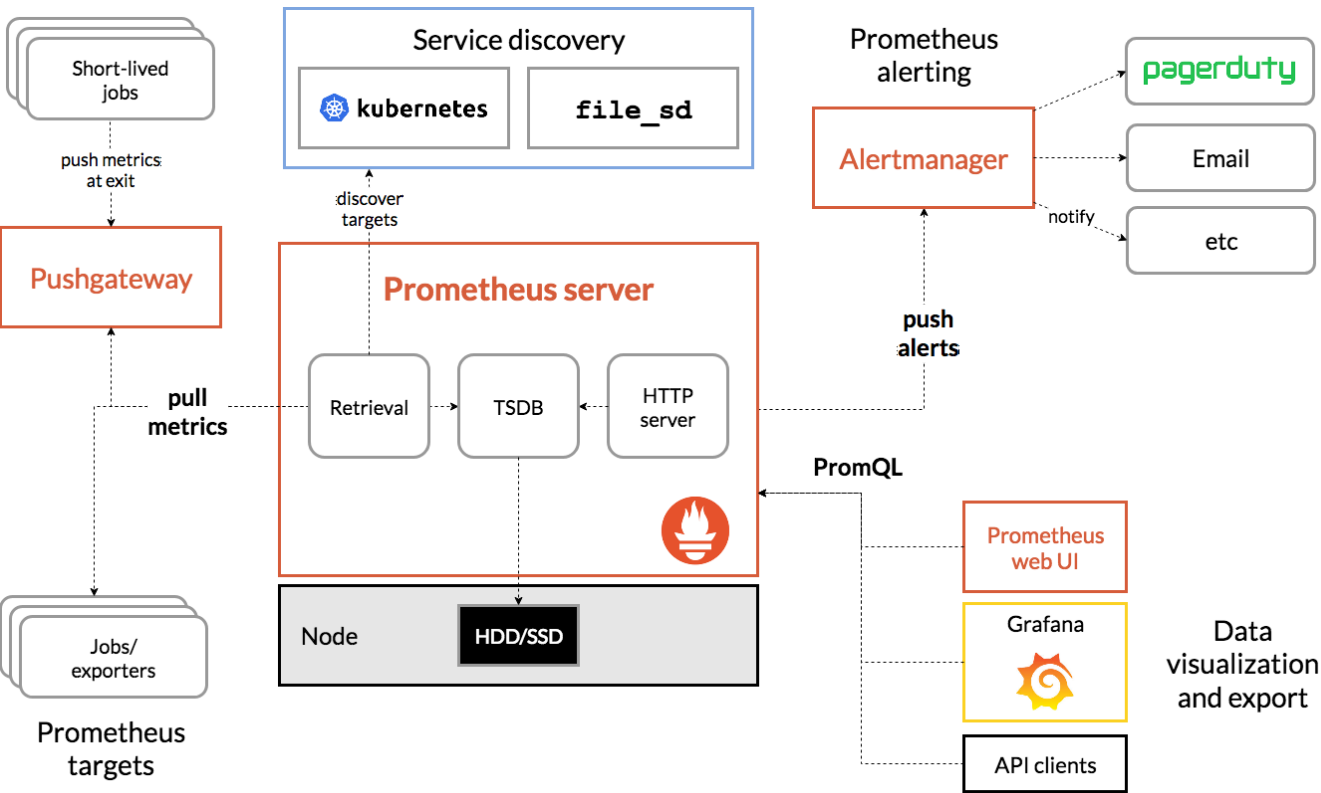
| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| nginx | gateway | deploy | true |
| | | delete | |

 [Edit this page](#)

Prometheus

Prometheus is an open-source systems monitoring and alerting toolkit originally built at SoundCloud. Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has a very active developer and user community. It is now a standalone open source project and maintained independently of any company. To emphasize this, and to clarify the project's governance structure, Prometheus joined the Cloud Native Computing Foundation in 2016 as the second hosted project, after Kubernetes.

Reference: <https://prometheus.io/>



Package Deployment

```
$ kubectl kubernetes package -n prometheus -a <action>
```

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| prometheus | service | deploy | true |

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| | | delete | |

 [Edit this page](#)

Redis


Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker. Redis provides data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes, and streams. Redis has built-in replication, Lua scripting, LRU eviction, transactions, and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster

Reference: <https://redis.io/>

Package Deployment

```
$ kupboard kollection package -n redis -a <action>
```

| Pakcage Name | Cluster | Action | Default |
|--------------|---------|--------|---------|
| redis | service | deploy | true |
| | | delete | |

 [Edit this page](#)