ELO Cloud on Kubernetes

Getting started

Table of contents

Overview

Core concepts

work in progress

Checklist

Install Kubectl, Docker, Helm

1. Kubctl
2. Docker
3. Helm

Namespace

Create a namespace for the operator

1. Open Terminal
2. Create namespace

   ```
   kubectl create namespace <elo-system>
   ```

   texts within the <> are freely selectable
3. Set the right namespace context

   ```
   kubectl config set-context --current --namespace=<elo-system>
   ```
4. Namespace created and correct context is set.

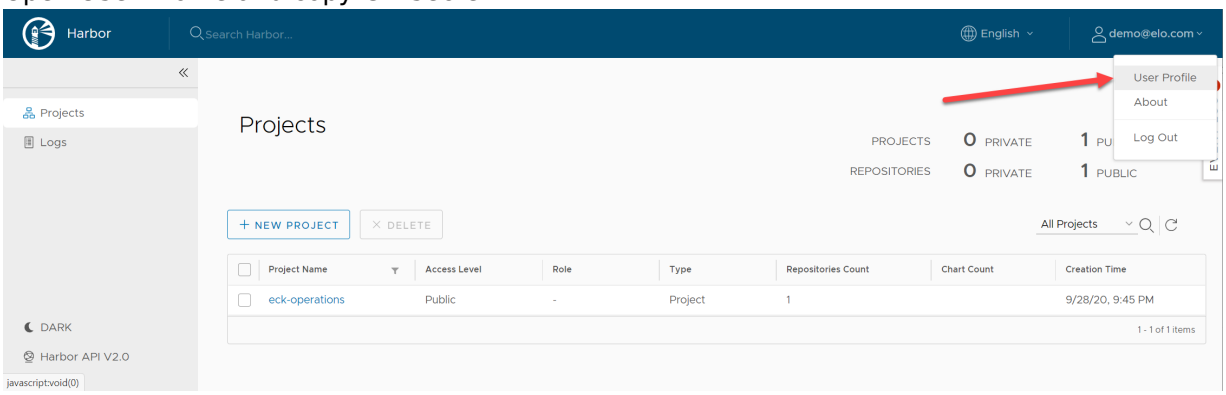
Pull secret

As the ELO services are stored in a private docker registry, therefore authentication is required.

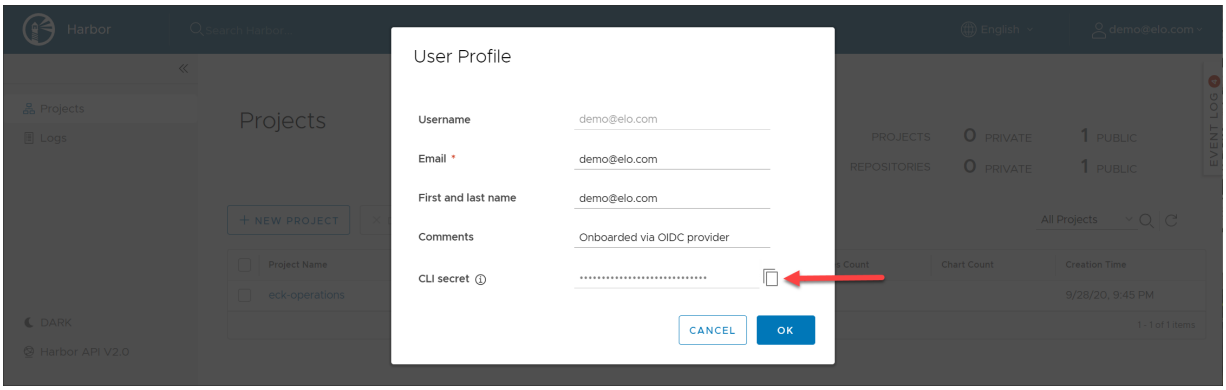
1. Open registry.elo.com and login using OIDC

2.

Open User Profile and copy CLI secret



3. Note username, email, and CLI secret



4. Open the terminal and enter

```
docker login registry.elo.com -u <username> -p <cli-secret>
```

5. Generate Pull secret `regcred-elo-com`

```
kubectl create secret docker-registry regcred-elo-com \
-n elo-system \
--docker-server=registry.elo.com \
--docker-username=<username> \
--docker-password=<cli-secret> \
--docker-email=<email>
```

6. Tweak Kubernetes Nodes Settings

   ELO-iSearch requires a minimum number of memory map areas. Use `DeamonSet` to automatically run a pod on each node that sets the `vm.max_map_count` of every node.

```
cat <<EOF | kubectl apply -f -
apiVersion: apps/v1
kind: DaemonSet
```

```
metadata:
  labels:
    k8s-app: sysctl-conf
  name: sysctl-conf
spec:
  selector:
    matchLabels:
        k8s-app: sysctl-conf
  template:
    metadata:
      labels:
        k8s-app: sysctl-conf
    spec:
      containers:
      - command:
        - sh
        - -c
        - sysctl -w vm.max_map_count=262166 && while true; do sleep 86400; done
        image: busybox:1.26.2
        name: sysctl-conf
        resources:
          limits:
            cpu: 10m
            memory: 50Mi
          requests:
            cpu: 10m
            memory: 50Mi
        securityContext:
          privileged: true
      terminationGracePeriodSeconds: 1
EOF
```

Required for each cluster

Refer to Virtual Memory Settings on worker nodes for more details.

Install the Operator

1. Install the CRDs

   See this page for more details

2. Add the chart repository to the local helm repository

```
helm repo add elo-operator https://registry.elo.com/chartrepo/eck-operations
```

3. Update information about available charts locally from chart repositories

```
helm repo update
```

4. Install the operator

```
helm install <release-name> --namespace <elo-system> --version 1.0.0-nightly elo-opera
```

Release name can be freely selected

Example:

```
helm install helmoperator --namespace mysystem --version 1.0.0-nightly elo-operator/e
```

Deploying the repository

1. Deploy an elo-lic secret

```
kubectl create secret generic elo-lic --from-file=license=/path/to/license
```

> It is important to have correct line endings for Linux environments applied to your license. Therefore you have to make sure, that instead of `\r\n` (Windows/DOS line ending) only `\n` (Unix/Linux line ending) is used as the line ending.
>
> You also have to make sure that your ELO license matches the major version of the ELO version you want to deploy. An ELO 21 license won't work in an ELO 23 deployment.

1. Check for usable versions in your operator environment

```
kubectl get elo-version -n <elo-system>
```

2. Check if regcred-elo-com secret is set in the `elo-operator-config` configMap & if the `docker_secret_deploy` has been set to "true"

```
kubectl get configMap elo-operator-config -n elo-system -o jsonpath='{"docker_secret:
```

If this is not the case, deploy your repository with your image pull secret information:

```
cat <<EOF | kubectl apply -f -
apiVersion: cloud.elo.com/v1beta1
kind: Repository
metadata:
  name: demo
  labels:
    cloud.elo.com/controller-id: <elo-system>
spec:
  eloLicenseSecretName: elo-lic
  version: v21-4.0
  kubernetes:
    pull:
      deploy: true
      secret: regcred-elo-com
EOF
```

3. Monitor cluster health and creation progress

```
kubectl get elo-repositories
```

4. Watch all resources that are being deployed by executing

```
kubectl get all
```

5. Get a list of all secrets

```
kubectl get secrets
```

6. Get username

```
kubectl get secrets/demo-auth-admin --template={{.data.username}} | base64 -d
```

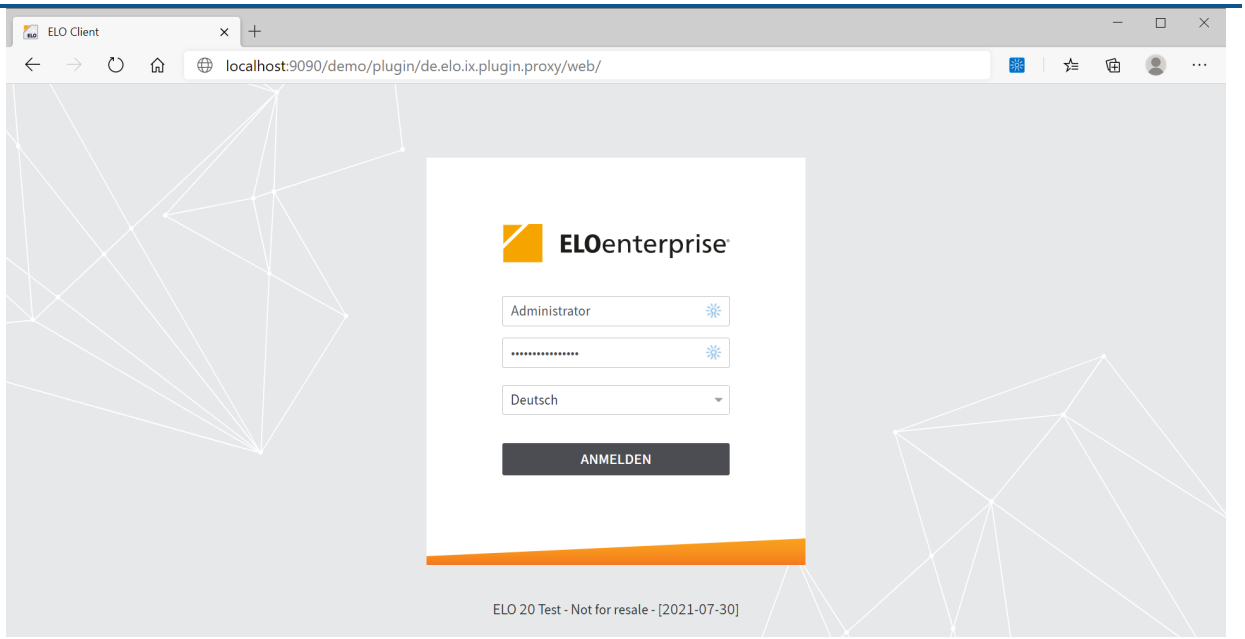
7. Get password

```
kubectl get secrets/demo-auth-admin --template={{.data.password}} | base64 -d
```

8. Access repository

```
kubectl port-forward service/demo-server 9090
```

9. Open the browser and access localhost:9090/demo/ and get redirected to the Web Client.


ELO Client
localhost:9090/demo/plugin/de.elo.ix.plugin.proxy/web/
ELOenterprise
Administrator
Deutsch
ANMELDEN
ELO 20 Test - Not for resale - [2021-07-30]

Verify Container Images

Signatures

Starting with ELO Image Releases after v23-0.0 and all new Operator Images starting with v1.0.0 will be signed using Cosign.

To verify the signature you may use Cosign as well. Follow these steps and use our public key.

Image Contents

In the near future we are also planning to attach "software bill of materials" (SBOM) to our Images.

Unfortunately is this not yet considered being production ready and therefore may take a bit.

Enforce using signed Images in Kubernetes

To make sure that you are only using signed images, from an authorized source, when deploying ELO and the Operator to Kubernetes one of the following tools may become helpful:

- https://sse-secure-systems.github.io/connaisseur/
- https://kyverno.io/
- https://www.openpolicyagent.org/

Getting started

Introduction

In single tenant mode one ELO instance is represented by one repository. All repositories get their own database and don't share access managers.

Quickstart

This quickstart guide shows how to

1. Setup prerequisites for running ELO Cloud on Kubernetes via Helm
2. Install and uninstall the ELO Cloud on Kubernetes Operator Helm Chart

Supported versions

- kubectl
- Kubernetes 1.24+ or OpenShift 4+
- helm 3

Known Issues

We are collecting known issues at this page. If you are encountering any errors, make sure to check this page first.

Prerequisites

You need docker and kubectl

Create a namespace for the operator

ECK supports multiple Operator instances running in one kubernetes cluster. Each operator must have its own namespace.

Create a namespace `elo-system`.

```
$ kubectl create namespace elo-system
$ kubectl config set-context --current --namespace=elo-system
```

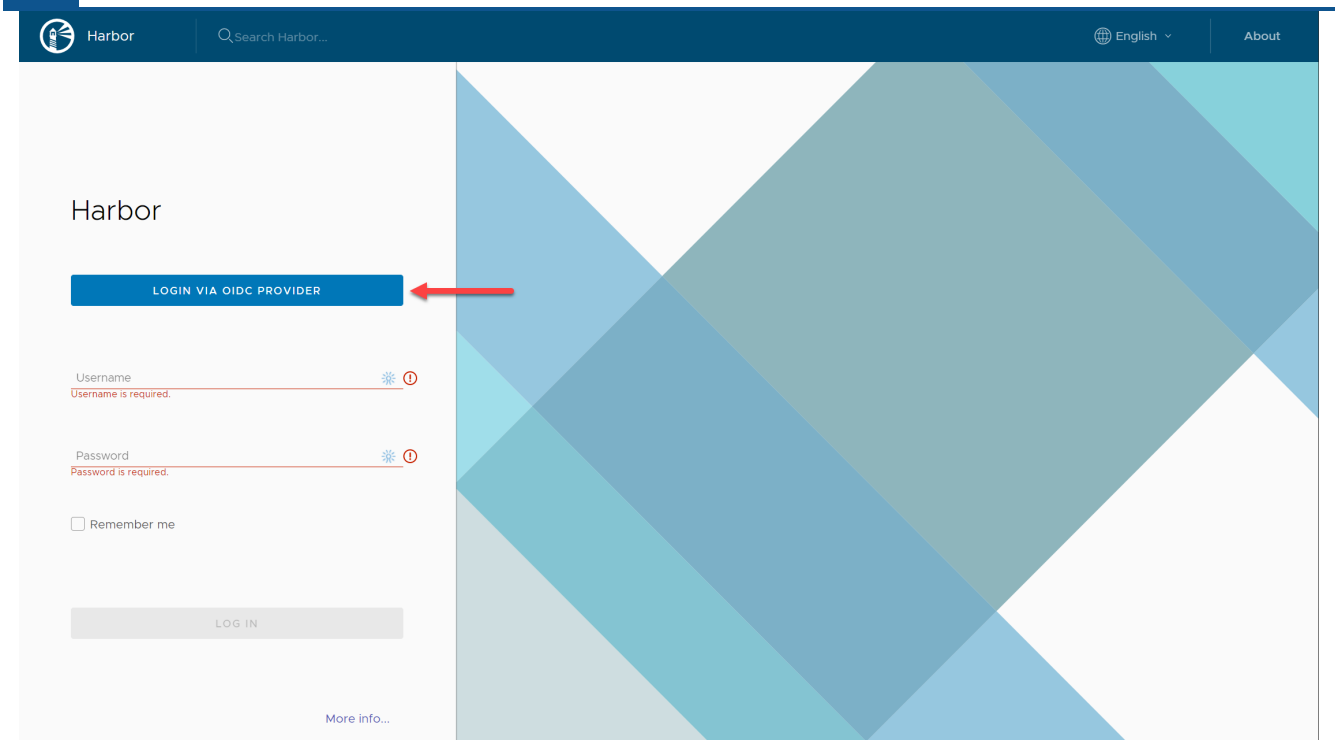
> Setting the default namespace context using `kubectl config set-context` after creating a new namespace ensures commands are executed and applied in the right namespace.
>
> In order to make handling namespaces and multiple clusters more easily there's a tips and tricks section in this documentation.

Create pull secret for registry.elo.com

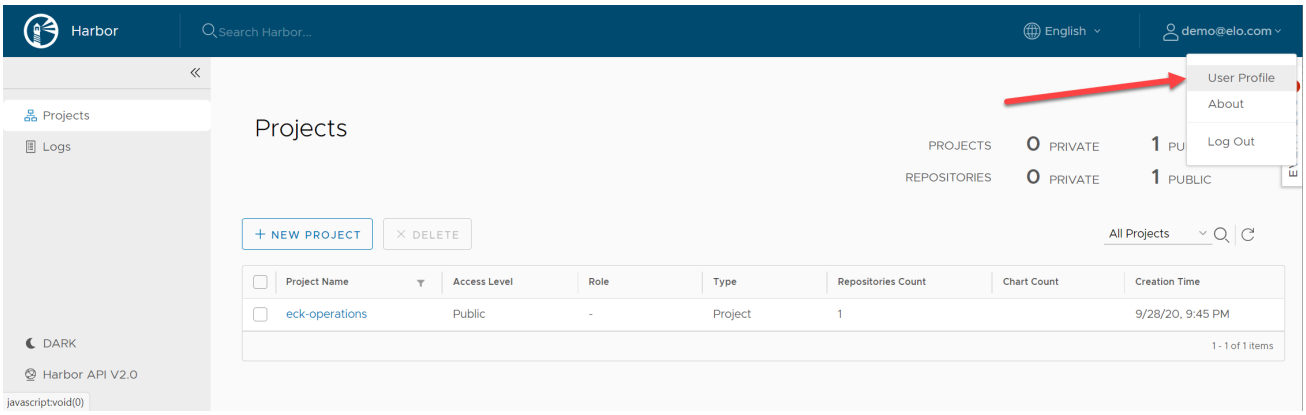
Images for ELO services are stored in a private docker registry hosted on registry.elo.com. Before pulling images from this registry an authentication is required. The following steps guide you through the process of generating a token that allows accessing private images from our registry.

Retrieve a CLI secret for pulling images

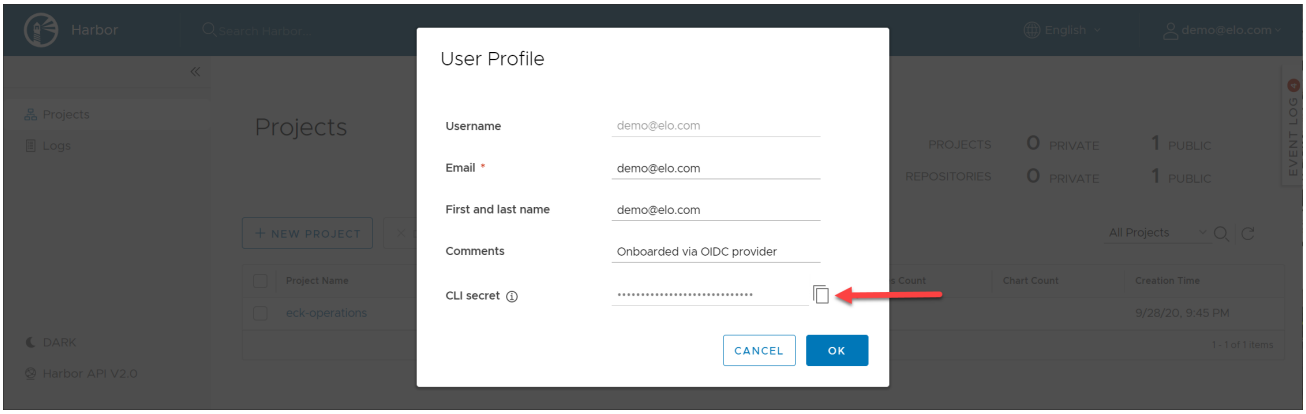
Open https://registry.elo.com in a web browser and login using OIDC provider.



The access token/ cli secret that can be used for pulling images can be retrieved in the user profile.



Copy the CLI secret from the user profile.



Remember username and cli secret for the next step.

```
username: demo@elo.com
email: demo@elo.com
cli-secret: y8mTMQ6uYy8mTMQ6uYy8mTMQ6uY
```

Authenticate and create kubernetes pull secret

Test credentials and connection with `docker login`. This allows pulling images for docker and docker-compose as well.

```
$ docker login registry.elo.com -u <username> -p <cli-secret>
```

Use the following command to generate the Pull Secret that can later be referenced by `regcred-elo-com`:

```
$ kubectl create secret docker-registry regcred-elo-com \
  -n elo-system \
  --docker-server=registry.elo.com \
  --docker-username=<username> \
  --docker-password=<cli-secret> \
  --docker-email=<email>
```

Tweak Kubernetes Nodes Settings

The ELO-iSearch requires a minimum number of memory map areas (262144). Deploy a DeamonSet which will automatically run a pod on each node that sets the `vm.max_map_count` of every node.

```
cat <<EOF | kubectl apply -f -
apiVersion: apps/v1
kind: DaemonSet
metadata:
  labels:
    k8s-app: sysctl-conf
  name: sysctl-conf
spec:
  selector:
    matchLabels:
        k8s-app: sysctl-conf
  template:
    metadata:
      labels:
```

```
        k8s-app: sysctl-conf
    spec:
      containers:
      - command:
        - sh
        - -c
        - sysctl -w vm.max_map_count=262166 && while true; do sleep 86400; done
        image: busybox:1.26.2
        name: sysctl-conf
        resources:
          limits:
            cpu: 10m
            memory: 50Mi
          requests:
            cpu: 10m
            memory: 50Mi
        securityContext:
          privileged: true
      terminationGracePeriodSeconds: 1
EOF
```

Refer to Virtual Memory Settings on worker nodes for more details.

Deploying the Operator

eck-operator

ELO Cloud Operator based on the OperatorSDK

Maintainers

Name	Email	Url
ELO DevOps	DevOps@elo.com	https://www.elo.com

Installing the Chart

> The Operator requires the CRDs to be present in the Cluster before being started. See CRD Versions for available options.

To install the chart with the release name `elo-operator`:

```
$ helm repo add elo-operator https://registry.elo.com/chartrepo/eck-operations
$ helm repo update
$ helm install elo-operator elo-operator/eck-operator
```

Values file Operator Helm Chart

The Values file provides access to values passed into the chart. The default values file can be changed with passing a value file into `helm install` or `helm upgrade` with the `-f` flag

```
$ helm install -f myvals.yaml elo-operator/eck-operator
```

Or with individual parameters passed with `--set` such as

```
$ helm install --set <name>=<value> elo-operator/eck-operator
```

For example the `pullPolicy` can be changed with:

```
$ helm install --set image.pullPolicy=Always elo-operator/eck-operator
```

You can pass multiple individual parameters with `--set`

```
$ helm install --set <name1>=<value1> --set <name2>=<value2> elo-operator/eck-operato
```

Additional information to `helm install` you can find here

Deploy the operator from template

Use `helm template` to render helm chart templates locally and display the output.

```
$ helm template demo --namespace elo-system --version 1.0.0 elo-operator/eck-operator > cat
```

This will output all k8s Ressources with the release name `demo` and the namespace value `elo-system` of the ECK Operator version `1.0.0`.

The All-in-one YAML manifest can be deployed with:

```
$ kubectl apply -f helm-template-aio.yaml
```

The annotation `helm.sh/resource-policy: keep` is set to the CRDs by default in order to prevent the CRDs from being deleted when the chart is uninstalled.

```yaml
metadata:
  annotations:
    helm.sh/resource-policy: keep
```

Parameters

Key	Type	Default
commonLabels	object	{}
config.debugMode	string bool	"false"
config.fsgroupDeploy	string bool	"true"
config.images.as	string	"eck-services/as"
config.images.busybox	string	"eck-services/busybox"
config.images.database	string	"eck-services/postgresql"
config.images.database_metrics	string	"eck-services/postgresql-exporter"
config.images.flows	string	"eck-services/flows"
config.images.flows-registry	string	"eck-services/flows-registry"

Key	Type	Default
config.images.flows-worker	string	`"eck-services/flows-worker"`
config.images.ix	string	`"eck-services/ix"`
config.images.ix-import	string	`"eck-services/ix-import"`
config.images.ix-setup	string	`"eck-services/ix-setup"`
config.images.search	string	`"eck-services/search"`
config.images.server	string	`"eck-services/server"`
config.images.tr2	string	`"eck-services/tr2"`
config.images.ui	string	`"eck-services/ui"`
config.servicesLogLevel	string	`"info"`
config.setRepositoryPVCOwnerReference	string bool	`"false"`
config.tenantHostnameBase	string	`"elo.dev"`
config.tenantHostnameTemplate	string	`"{tenant.name}-{operator.namespace}.{tenant.ho`
config.tlsName	string	`""`
controller.prometheus.enabled	bool	`false`
controller.prometheus.service	string	`{"annotations":{"prometheus.io/path":"/metrics scrape":"true"}}`
controller.prometheus.serviceMonitor	object	`{"additionalLabels": {},"enabled":false,"honorLabels":false,"interv [],"namespace":"","scrapeTimeout":"","selector`
controller.prometheus.serviceMonitor.additionalLabels	object	`{}`
controller.prometheus.serviceMonitor.enabled	bool	`false`
controller.prometheus.serviceMonitor.honorLabels	bool	`false`
controller.prometheus.serviceMonitor.interval	string	`""`
controller.prometheus.serviceMonitor.jobLabel	string	`""`
controller.prometheus.serviceMonitor.metricRelabelings	list	`[]`
controller.prometheus.serviceMonitor.namespace	string	`""`
controller.prometheus.serviceMonitor.scrapeTimeout	string	`""`
controller.prometheus.serviceMonitor.selector	object	`{}`
controller.singleNamespace	bool	`false`
controller.verbosity	string int	`"2"`
fullnameOverride	string	`""`
image	object	`{"pullPolicy":"IfNotPresent","repository":"eck`
image.pullPolicy	string	IfNotPresent

Key	Type	Default
image.repository	string	`"eck-operations/operator"`
image.tag	string	`""`
imagePullSecret	string	`"regcred-elo-com"`
imagePullSecretDeploy	bool string	`"true"`
nameOverride	string	`""`

operator	object	`{"crds":{"install":false,"pullPolicy":"Always"`
		`crd","serviceAccount":"crd-rollout-sa","tag":"`
		`{"development":false,"encoder":"json","level":`

operator.crds.install	bool	false
operator.crds.pullPolicy	string	Always
operator.crds.repository	string	`"eck-operations/operator-crd"`
operator.crds.serviceAccount	string	`"crd-rollout-sa"`
operator.crds.tag	string	`""`
operator.log.development	bool	false
operator.log.encoder	string	json
operator.log.level	string	info
operator.log.stacktraceLevel	string	error
operator.log.timeEncoding	string	iso8601
rbac	object	`{"create":true}`
rbac.create	bool	`true`
registry	string	`"registry.elo.com"`
resources	object	`{"limits":{"cpu":"500m","memory":"128Mi"},"req`
resources.limits	object	`{"cpu":"500m","memory":"128Mi"}`
resources.limits.cpu	string	`"500m"`
resources.limits.memory	string	`"128Mi"`
resources.requests	object	`{"cpu":"10m","memory":"64Mi"}`
resources.requests.cpu	string	`"10m"`

Key	Type	Default
resources.requests.memory	string	`"64Mi"`
service.Port	int	8181
serviceAccount	object	`{"create":true,"name":"eck-operator-sa"}`
serviceAccount.create	bool	`true`
serviceAccount.name	string	`"eck-operator-sa"`

Default values.yaml

```
# -- Image for the Operator
image:
  # -- The repository from where the Operator image should be pulled
  repository: eck-operations/operator
  # -- (string) Specify the image tag
  tag: ""
  # -- Specify the image pull policy
  # @default -- IfNotPresent
  pullPolicy: IfNotPresent

# -- BindFlags allows to pass flags to the Operator
operator:
  log:
    # -- (string) The log level for the Operator, Can be one of 'debug', 'info', 'error', o
    # @default -- info
    level: info
    # -- (bool) Whether to enable the development mode for the Operator. If false = (encode
    # @default -- false
    development: false
    # -- (string) The log encoder for the Operator.  Zap log encoding (one of 'json' or 'co
    # @default -- json
    encoder: json
    # -- (string) The log stacktrace level for the Operator. Zap Level at and above which s
    # @default -- error
    stacktraceLevel: error
    # -- (string) The log time encoding for the Operator. Zap time encoding (one of 'epoch'
    # @default -- iso8601
    timeEncoding: iso8601
  crds:
    # -- (bool) Whether to install the crds with a job or not
    # @default -- false
    install: false
    # -- (string) The name of the service account to use for the crds job
    serviceAccount: crd-rollout-sa
    # -- (string) The name of the repository for the operator crd job image
```

```
    repository: eck-operations/operator-crd
    # -- (string) The tag of the operator crd job image
    tag: ""
    # -- (string) The pull policy for the operator crd job image
    # @default -- Always
    pullPolicy: Always

# -- Whether to create the rbac ressources or not
rbac:
  # -- (bool) Whether to create the rbac ressources or not
  create: true

# -- Whether to create the ServiceAccount or not
serviceAccount:
  # -- (bool) Whether to create the ServiceAccount or not
  create: true
  # -- (string) The name of the ServiceAccount to use
  name: eck-operator-sa

service:
  # -- (int) Test Connection service port
  Port: 8181

# -- The registry from where the Operator and the ELO images should be pulled from
registry: registry.elo.com

# -- (bool string) Whether to add the image pull secret to deployed workloads or not
imagePullSecretDeploy: "true"

# -- The secret for pulling the images from the registry mentioned before
imagePullSecret: regcred-elo-com

# -- (string) String to partially override eck-operator.fullname template (will maintain th
nameOverride: ""

# -- (string) String to fully override eck-operator.fullname
fullnameOverride: ""

# -- (object) Labels to add to all deployed objects
commonLabels: {}

# -- (object) Resources allows overwriting the default resources
resources:
  # -- The resources limits for the Operator
  limits:
    # -- (string) The CPU limit for the Operator
```

```
    cpu: 500m
    # -- (string) The memory limit for the Operator
    memory: 128Mi
  # -- The resources requests for the Operator
  requests:
    # -- (string) The CPU request for the Operator
    cpu: 10m
    # -- (string) The memory request for the Operator
    memory: 64Mi

controller:
  # -- (bool) Limit the controller to only watch resources in its own namespace
  singleNamespace: false

  # -- (string int) Controls the verbosity of the Ansible Operator while executing
  verbosity: "2"

  prometheus:
    # -- (bool) Enable the export of Prometheus metrics
    enabled: false

    # -- (string) The port on which the metrics are exposed
    service:
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "8080"
        prometheus.io/path: "/metrics"

    # -- Prometheus Operator service monitors
    serviceMonitor:
      # -- (bool) if `true`, creates a Prometheus Operator ServiceMonitor (also requires `p
      enabled: false
      # -- (string) Namespace in which Prometheus is running
      namespace: ""
      # -- (string) Interval at which metrics should be scraped.
      # -- ref: https://github.com/coreos/prometheus-operator/blob/master/Documentation/api
      # -- e.g: 15s
      interval: ""
      # -- (string) Timeout after which the scrape is ended
      # -- ref: https://github.com/coreos/prometheus-operator/blob/master/Documentation/api
      # -- e.g: 10s
      scrapeTimeout: ""
      # -- (object) Prometheus instance selector labels
      # -- ref: https://github.com/prometheus-operator/prometheus-operator/blob/master/Docu
      # -- e.g: selector.prometheus: my-prometheus
      selector: {}
```

```
      # -- (list) Specify Metric Relabellings to add to the scrape endpoint
      # -- ref: https://github.com/prometheus-operator/prometheus-operator/blob/master/Docu
      metricRelabelings: []
      # -- (bool) Labels to honor to add to the scrape endpoint
      honorLabels: false
      # -- (object) custom labels for the ServiceMonitor
      additionalLabels: {}
      # -- (string) The name of the label on the target service to use as the job name in p
      jobLabel: ""

config:
  # -- (string bool) Whether to enable the debug mode or not
  debugMode: "false"
  # -- (string) The name of the tls secret to use for the Operator
  tlsName: ""
  # -- (string bool) Whether to enable the fsGroup for the Operator or not
  fsgroupDeploy: "true"

  # -- (string) Log level for the Operator
  servicesLogLevel: "info"
  # -- (string) The tenant hostname for the Operator
  tenantHostnameBase: "elo.dev"
  # -- (string) The tenant hostname template for the Operator
  tenantHostnameTemplate: "{tenant.name}-{operator.namespace}.{tenant.hostname.base}"
  # -- (string bool) Whether to set repository pvc owner reference or not
  setRepositoryPVCOwnerReference: "false"

  images:
    # -- (string) The image tag for the as component
    as: eck-services/as
    # -- (string) The image tag for the database component
    database: eck-services/postgresql
    # -- (string) The image tag for the database_metrics component
    database_metrics: eck-services/postgresql-exporter
    # -- (string) The image tag for the indexserver component
    ix: eck-services/ix
    # -- (string) The image tag for the ix-setup component
    ix-setup: eck-services/ix-setup
    # -- (string) The image tag for the ix-import component
    ix-import: eck-services/ix-import
    # -- (string) The image tag for the search component
    search: eck-services/search
    # -- (string) The image tag for the ui component
    ui: eck-services/ui
    # -- (string) The image tag for the busybox image
    busybox: eck-services/busybox
```

```
    # -- (string) The image tag for the flows component
    flows: eck-services/flows
    # -- (string) The image tag for the flows-registry component
    flows-registry: eck-services/flows-registry
    # -- (string) The image tag for the flows-worker component
    flows-worker: eck-services/flows-worker
    # -- (string) The image tag for the tr2 component
    tr2: eck-services/tr2
    # -- (string) The image tag for the server component
    server: eck-services/server
```

CRD versions

The Custom Resources Definitions (CRDs) can be downloaded from the documentation of the ECK Operator. The CRDs must be installed before the operator can be deployed. Installing the CRDs with the Helm chart can be done by setting the value `operator.crds.install` to `true` in the values file. The default value is `false`.

```yaml
operator:
  crds:
    install: true
```

Or by using the `--set` flag with `helm install` or `helm upgrade`.

```
$ helm install --set operator.crds.install=true elo-operator/eck-operator
```

This method installs the CRDs using a Job and ServiceAccount, which are deleted after the CRDs are installed. This job will also create the latest versions of the Custom Resource for the operator.

> This is the recommended way to install the CRDs.

In addition to this method the CRDs can be downloaded and installed manually. The following sections describe how to do this

Downloading CRDs and install from Path

One of the matching CRD versions has to be downloaded first:

- 1.1.0
- 1.0.0

The following command be used to install the CRDs from a local path:

```
kubectl apply --server-side -f path/to/downloaded/crds.yaml
```

The CRDs must be applied to the cluster with the kubectl command flag `--server-side`.

Deploying a repository

Apply a simple ELO Repository resource.

> It is recommended that kubernetes worker nodes have at least 4 GiB of memory or at least 2 GiB of free memory. Otherwise pods might stuck in pending state.

Make sure that you deploy a ELO license secret `elo-lic` with a valid ELO license in the same namespace in which your repository should be deployed. The `license` key in your ELO license secret `elo-lic` has to contain your license.

To deploy an `elo-lic` secret you can apply:

```
kubectl create secret generic elo-lic --from-file=license=/path/to/license
```

> It is important to have correct line endings for Linux environments applied to your license. Therefore you have to make sure, that instead of `\r\n` (Windows/DOS line ending) only `\n` (Unix/Linux line ending) is used as the line ending.
>
> You also have to make sure that your ELO license matches the major version of the ELO version you want to deploy. An ELO 21 license won't work in an ELO 23 deployment.

The Operator will start deploying the repository as soon as the CR is applied.

```
$ cat <<EOF | kubectl apply -f -
  apiVersion: cloud.elo.com/v1beta1
  kind: Repository
  metadata:
    name: demo
  spec:
    version: v21-4.0
EOF
```

> In the previous statement, the Version v21-4.0 was used. There might be a newer Version available. To check for usable Versions in your Operator environment use the following statement:
>
> ```
> $ kubectl get elo-version -n <operator-namespace>
>
> NAME      STACK     AS        FLOWS     FLOWSW    IX        SEARCH    UI
> ```

```
v21-4.0   v21-4.0   v21-4.0   v21-4.0   v21-4.1   v21-4.2   v21-4.1   v21-4.1
v23-0.0   v21-4.0   v23-0.0   v23-0.0   v23-0.0   v23-0.0   v23-0.0   v23-0.0
```

If you want to apply a different Version, just replace the previously mentioned `version: v21-4.0` with `version: <STACK>`.

If you are pulling the images directly from registry.elo.com, make sure to have a valid imagePullSecret created in the namespace of the `Repository`.

More information regarding imagePullSecrets and the handling within the Operator can be found here.

The operator automatically creates and manages Kubernetes resources to achieve the desired state of the ELO Repository. It may take up a couple of minutes until all the resources are created and the Repository is ready for use.

Monitor cluster health and creation progress

Get an overview of the current Repositories deployed in the cluster. `Running reconciliation` means that the Operator is currently processing the repository configuration and will generate kubernetes resources.

```
kubectl get elo-repositories

NAME   VERSION   STATUS    MESSAGE
demo   v21-4.0   Running   Running reconciliation
```

You can watch all resources being deployed by executing

```
kubectl get all
```

Getting generated secrets

During initialization the Operator will automatically create a couple of secrets for the repository. This includes access tokens like passwords or if the embedded PostgreSQL database is used database credentials. These are stored as kubernetes secrets.

In order to login passwords must be extracted from the generated secrets.

Get a list of all secrets first.

```
kubectl get secrets

NAME                                    TYPE                                  DATA   AGE
```

```
demo-aio-node-0                  kubernetes.io/tls                 2      9m32
demo-auth-admin                  Opaque                            2      9m34
demo-auth-database               Opaque                            5      9m33
demo-auth-keystore               Opaque                            1      9m34
demo-auth-service                Opaque                            4      9m34
demo-auth-tomcat                 Opaque                            2      9m34
demo-search-ca                   kubernetes.io/tls                 2      9m33
demo-server-search               kubernetes.io/tls                 2      9m30
demo-sg-admin                    kubernetes.io/tls                 2      9m33
```

`{repo-name}-auth-admin` and `{repo-name}-auth-service` contain the generated Administrator and ELO Service credentials.

Please note that secrets are base64 encoded.

```
kubectl get secrets/demo-auth-admin --template={{.data.username}} | base64 -d

Administrator
```

```
kubectl get secrets/demo-auth-admin --template={{.data.password}} | base64 -d

e.9nZq.1rPHAQ1Yh
```

Access the repository

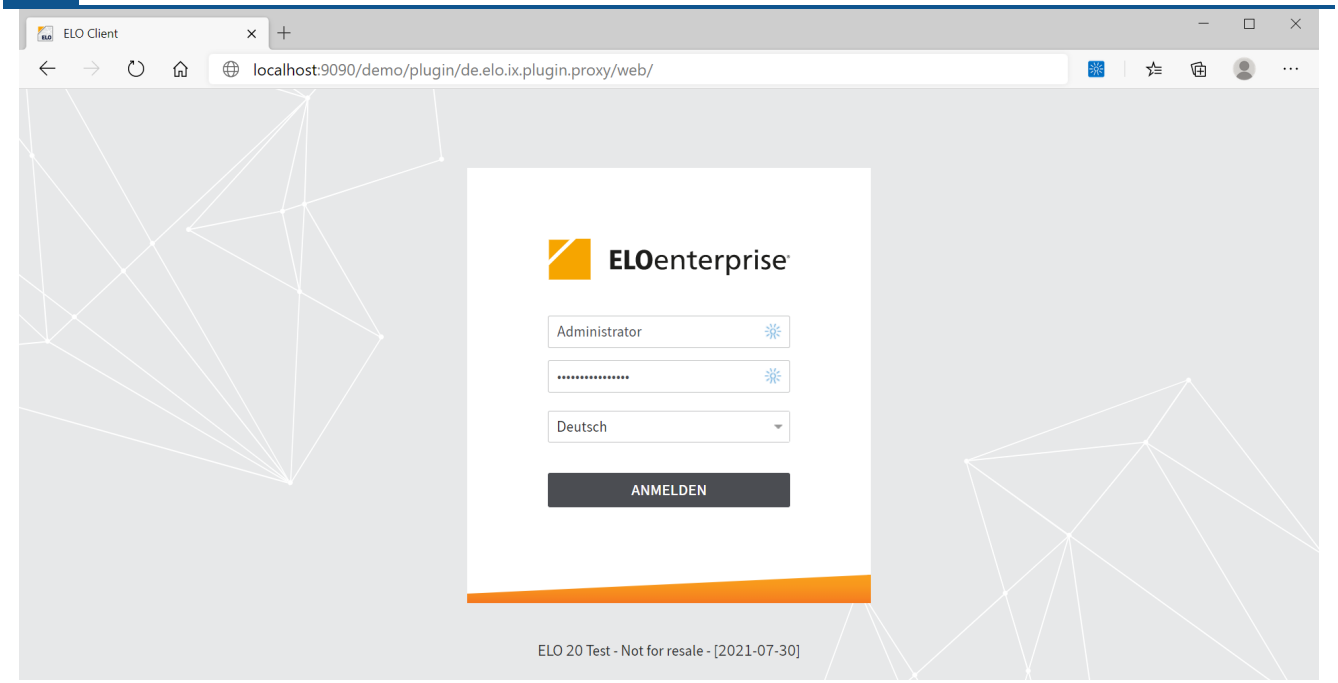
> This getting started guide focuses on getting services running in Kubernetes.
>
> Repositories allow defining host names which will internally create Ingress ressources in kubernetes. Refer to Administration/Hostname for more information.

Kubectl provides a proxy that allows connecting to local services.

```
kubectl port-forward service/demo-ix 9090
```

The Index Server acts as a frontend proxy for user centric services. Repositories are mapped using their name in the URL. e.g. `localhost:9090/{repository-name}/` points to the ELO Index Server. `localhost:9090/{repository-name}/ix` will be the ix API Endpoint used by ELO Clients.

Open a web browser with the following URL localhost:9090/demo/. The Index Server will redirect to the Web Client.


ELO Client
localhost:9090/demo/plugin/de.elo.ix.plugin.proxy/web/
ELOenterprise
Administrator
Deutsch
ANMELDEN
ELO 20 Test - Not for resale - [2021-07-30]

Getting started - Multitenancy

Introduction

The ELO Cloud on Kubernetes Operator has the capability of handling multiple Tenants.

To be able to achieve this, you have to create each Tenant, and its related Repositories in the same Namespace. You also have to make sure that a maximum of one Tenant is being placed within a single Namespace.

Quick start

In addition to the Getting started guide this quick start shows how to

1. Deploy the first tenant
2. Deploy the first repository for a tenant

Deploy a tenant

> Do not attempt to create multiple Tenants in the same namespace. Always put each Tenant in a different Namespace.

Apply a new resource for a tenant

```
cat <<EOF | kubectl apply -f -
apiVersion: cloud.elo.com/v1alpha1
kind: Tenant
metadata:
  name: contelo
  labels:
    cloud.elo.com/controller-id: elo-operator
spec:
  version: latest #<-- Change this
EOF
```

The operator automatically creates and manages Kubernetes resources to achieve the desired state of the ELO Tenant.

Monitor tenant health and creation progress

Get an overview of all tenants deployed in the cluster.

```
$ kubectl get elo-tenants

NAME         VERSION   STATUS    MESSAGE
contelo      latest    Running   Running reconciliation
```

All tenants will get their own database instance including required authentication secrets. These are created by the operator in the tenants namespace. Get an overview of all resources deployed for a tenant.

```
$ kubectl get all --namespace <tenant-namespace>

NAME                            READY   STATUS    RESTARTS   AGE
pod/database-74c88dbf4f-n2dxr   1/1     Running   0          9s

NAME               TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)             AGE
service/database   ClusterIP   10.43.197.37   <none>        5433/TCP,5432/TCP   8s
```

```
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/database      1/1     1            1           9s

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/database-74c88dbf4f     1         1         1       9s
```

Getting generated secrets

Authentication secrets are generated in the tenants namespace and are shared across repositories. This allows all repositories sharing one access manager. Therefore secret names don't prefix the repository name.

Get a list of all secrets.

```
$ kubectl get secrets --namespace <tenant-namespace>

NAME                          TYPE                                  DATA   AGE
auth-admin                    Opaque                                2      34s
auth-database                 Opaque                                5      32s
auth-keystore                 Opaque                                1      33s
auth-service                  Opaque                                2      35s
```

Get the `Administrator` password.

```
$ kubectl --namespace <tenant-namespace> get secrets/auth-admin --template={{.data.password

e.9nZq.1rPHAQ1Yh
```

Deploy a repository for the tenant

Repositories require the information for what Tenant they should be deployed for. The Tenant reference has to be defined in the label `cloud.elo.com/tenant` using `metadata.name` of the Tenant CR.

To be able to link an ELO Repository to a Tenant you also have to place the Repository CR within the Namespace of the Tenant.

```
$ cat <<EOF | kubectl apply -f -
apiVersion: cloud.elo.com/v1beta1
kind: Repository
metadata:
  name: contelo-demo
  labels:
    cloud.elo.com/tenant: contelo
EOF
```

> Please note that repository version should be set in the Tenant resource. The Tenants version definition is inherited to the Repository and can be overridden by `specs.version`.
>
> This allows keeping repository services that access shared databases (e.g. access manager database) in line.

Monitor cluster health and creation progress

Repository resources are deployed in the Tenant's namespace.

```
$ kubectl get all --namespace <tenant-namespace>

pod/database-74c88dbf4f-n2dxr               1/1     Running   0          4m11s
pod/contelo-demo-as-54796cf848-mwhr5        1/1     Running   0          3m56s
pod/contelo-demo-ix-0                       1/1     Running   0          3m59s
pod/contelo-demo-search-55cfff6d7f-64p22    1/1     Running   0          3m54s
pod/contelo-demo-ui-554d5d5486-29s5c        1/1     Running   0          3m57s

NAME                             TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)
service/database                 ClusterIP   10.43.197.37   <none>        5433/TCP,5432/TCP
service/contelo-demo-as          ClusterIP   10.43.5.164    <none>        9090/TCP
service/contelo-demo-ix          ClusterIP   10.43.46.163   <none>        9090/TCP
service/contelo-demo-search      ClusterIP   10.43.97.151   <none>        9204/TCP,9200/TCP,93
service/contelo-demo-ui          ClusterIP   10.43.210.16   <none>        9090/TCP

NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
```

```
deployment.apps/database              1/1     1            1           4m11s
deployment.apps/contelo-demo-as       1/1     1            1           3m56s
deployment.apps/contelo-demo-search   1/1     1            1           3m54s
deployment.apps/contelo-demo-ui       1/1     1            1           3m57s

NAME                                             DESIRED   CURRENT   READY   AGE
replicaset.apps/database-74c88dbf4f              1         1         1       4m11s
replicaset.apps/contelo-demo-as-54796cf848       1         1         1       3m56s
replicaset.apps/contelo-demo-search-55cfff6d7f   1         1         1       3m54s
replicaset.apps/contelo-demo-ui-554d5d5486       1         1         1       3m57s
```

Access the repository

> This getting started guide focuses on getting services running in Kubernetes. If operators run in multi tenant mode host names can be generated based on a subdomain. e.g. `https://contelo.elo.cloud/contelo-demo/`
>
> Refer to Administration/Hostname for more information.

Kubectl provides a proxy that allows connecting to local services.

```
$ kubectl port-forward service/contelo-demo-ix 9090 --namespace elo-system-contelo
```

The Index Server acts as a frontend proxy for user-centric services. Repositories are mapped using their name in the URL. e.g. `localhost:9090/{repository-name}/` points to the ELO Index Server. `localhost:9090/{repository-name}/ix` will be the ix API Endpoint used by ELO Clients.

Open a web browser with the following URL localhost:9090/contelo-demo/. The Index Server will redirect to the Web Client.