Magnolia PaaS:

Deployment Guide

1.0, 2021-06-16: initial release

Table of Contents

Copyright . . . 1
Revision History . . . 2
1. Deploy Magnolia PaaS . . . 3
  1.1. Deploy Magnolia PaaS . . . 3
  1.2. Configure Helm values . . . 5
  1.3. Create activation key . . . 8
  1.4. Use Makefile for deployment . . . 9
2. Add Cluster to GitLab project . . . 12
  2.1. Prerequisites . . . 12
  2.2. Instructions . . . 12
3. Webapp deployment . . . 18
  3.1. The `.gitlab-ci.yml` file . . . 18
  3.2. The `values.yml` file . . . 21
4. Light Module deployment . . . 24
  4.1. The `.gitlab-ci.yml` file . . . 24
Appendix A: Helm Values reference . . . 26

Copyright

Revision History

Revision	Date	Comments
`1.0`	2021-06-16	initial release

Chapter 1. Deploy Magnolia PaaS

1.1. Deploy Magnolia PaaS

To deploy Magnolia PaaS, you need to make sure you meet certain prerequisites as well as follow the instructions provided here. There are two sub-sections in this guide:

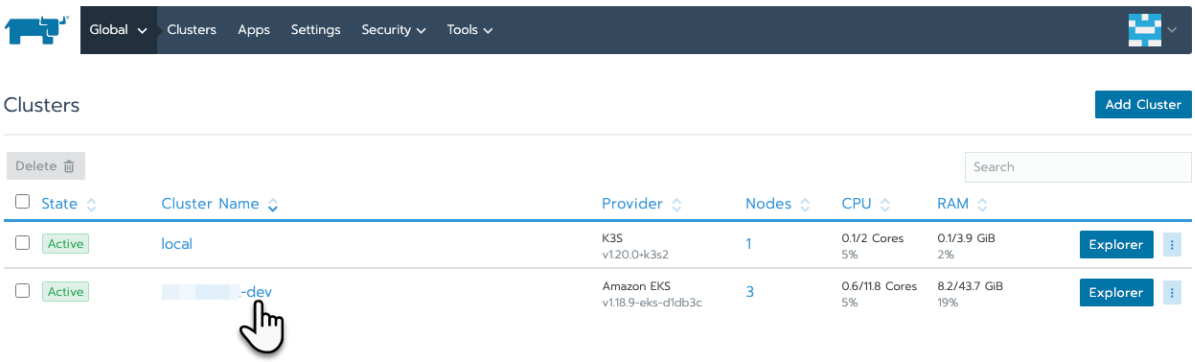
- Prerequisites
- Set up cluster
- Configure Helm values
- Use Makefile for deployment

1.1.1. Prerequisites

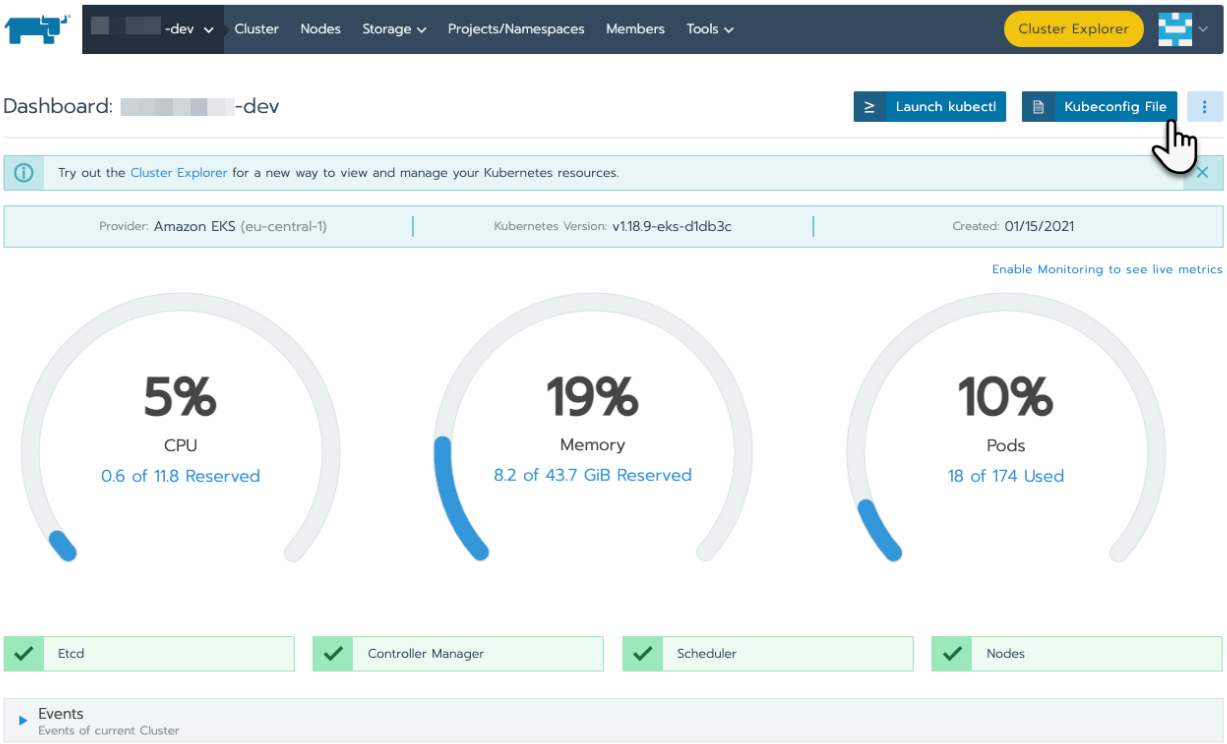
- You must have `kubectl` installed.
- You should be familiar with at least basic `kubectl` commands.
- You must have Helm installed.
- You must have been provided Rancher credentials by a member of the Magnolia PaaS team.

1.1.2. Set up cluster

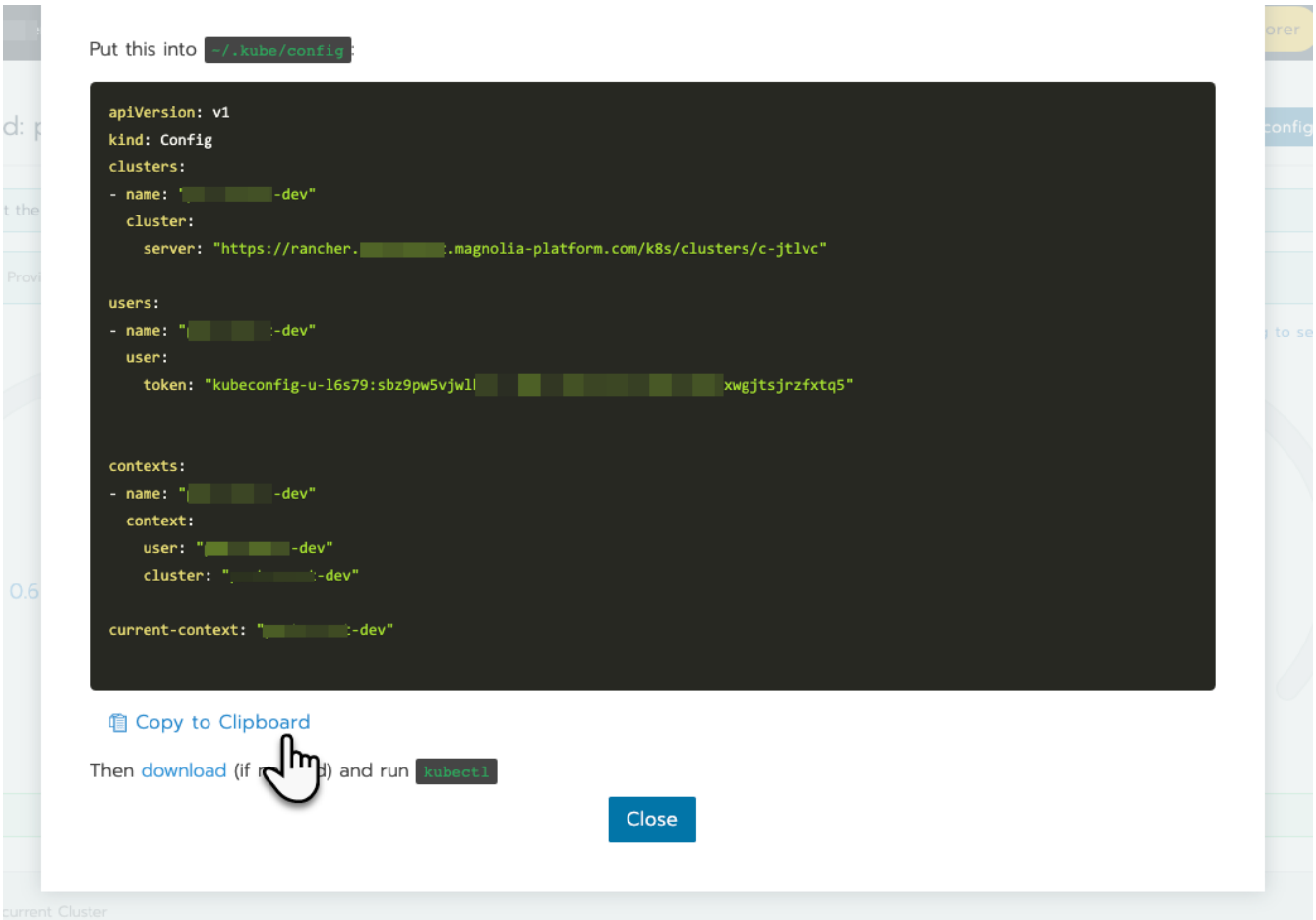
1. First, log in to your Rancher platform with your provided credentials. *This is where you see the clusters we have provisioned for you.*
2. Click the cluster that you want to connect to. *this takes you to the cluster*



3. Click **Kubeconfig File** in the top right above your cluster details.



4. In the pop-up window, copy the configuration settings shown to your local `~/.kube/config` file.



5. Check your configuration locally with the following command:

```
kubectl config get-contexts
```

You can switch to the context with `kubectl config use-context <your-context>`.

6. Add the repo that hosts the **magnolia helm chart** with the following commands:

```
helm repo add mironet https://charts.mirohost.ch;
helm repo update
```

1.2. Configure Helm values

Helm Values files are built-in Helm objects provide access to values passed to a chart and can be used inside of templates.

This section provides a template `values.yml` file to get your started. The table below describes the differnet parameters in the file.

For more details on values, see our Magnolia PaaS Helm Values reference page.

figure 1. values.yml

```
ingress:
  enabled: true
  annotations:
    nginx.ingress.kubernetes.io/proxy-body-size: 512m
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
  hosts:
    - host: <env>.<project>.magnolia-platform.com ① ②
      paths:
        - path: /
          instance: public
        - path: /author
          instance: author
  tls:
    - hosts:
      - <env>.<project>.magnolia-platform.com ① ②
      secretName: <env>-cert ①

image:
  pullSecrets: ③
    - name: docker-registry
  pullPolicy: Always

magnoliaAuthor:
  restartPolicy: Always
  redeploy: true ④
  contextPath: /author
  webarchive:
    repository: magnoliahamburg/custom-cloud
```

```
    tag: 6.2.6.1
  bootstrap:
    password: "<password>" ⑤
  activation:
    useExistingSecret: True ⑥
    secret:
      name: activation-key
      key: activation-secret
  env:
    - name: magnolia.superuser.enabled
      value: "true"
    - name: magnolia.superuser.password
      value: "<password>" ⑤
    - name: magnolia.bootstrap.license.owner
      value: "<license owner>"
    - name: magnolia.bootstrap.license.key
      value: "<license key>"
  setenv:
    memory:
      maxPercentage: 80 ⑦
  resources:
    requests:
      memory: 4Gi ⑧
    limits:
      memory: 4Gi ⑧
  livenessProbe:
    enabled: true

magnoliaPublic:
  restartPolicy: Always
  contextPath: /
  webarchive:
    repository: magnoliahamburg/custom-cloud
    tag: 6.2.6.1
  bootstrap:
    password: "<password>" ⑤
  activation:
    useExistingSecret: True ⑥
    secret:
      name: activation-key
      key: activation-secret
  env:
    - name: magnolia.superuser.enabled
      value: "true"
    - name: magnolia.superuser.password
      value: "<password>" ⑤
    - name: magnolia.bootstrap.license.owner
      value: "<license owner>"
    - name: magnolia.bootstrap.license.key
      value: "<license key>"
  setenv:
```

```
      memory:
        maxPercentage: 80 ⑦
    resources:
      requests:
        memory: 4Gi ⑧
      limits:
        memory: 4Gi
    livenessProbe:
      enabled: true
```

Item	Notes
1	The `<env>` value should be the same as used in the *RELEASE* value in the Helm call. A wildcard DNS entry for `<project>.magnolia-platform.com` `would support any valid value for` `<env>` and create a corresponding SSL certificate.
2	The `<project>` name is generated by Magnolia when we create your cluster. This is typically your company name.
3	Specify the `pullSecrets`. if the webarchive image is located in a private registry, see Use Docker secret.
4	Boolean specifying if there is an automatic redeployment triggered by changes. *defaults* • `magnoliaAuthor` = `false` • `magnoliaPublic` = `true` If set to *true* the instance is restarted even if the tag of the webarchive was not changed between deployments
5	Specify the environment `password`. This must be the same in all places for the bootstrapping container to work correctly.
6	The activation key is handled by the bootstrapper container. This keeps `magnoliaAuthor` and `magnoliaPublic` in sync.

Item	Notes
7	Specify the maximum percentage memory that is reserved for the tomcat container. **Default** = `60`
8	Sets your memory for requests and limits. `4GB` is typically sufficient for Magnolia CMS.

1.3. Create activation key

Create an activation secret for each of your environments using the following shell script below.

Set the `<env>` **(1)** to the same value (*production, integration, etc.*) set in the Helm call. This ensures the secret can be used properly by your deployment.

figure 2. create-activation-secret.sh

```
TEMPDIR=$(cat /dev/urandom | tr -dc 'a-z' | fold -w 10 | head -n 1)
NAMESPACE=<env> ①
mkdir $TEMPDIR
openssl genrsa -out $TEMPDIR/key.pem 1024
openssl rsa -in $TEMPDIR/key.pem -pubout -outform DER -out $TEMPDIR/pubkey.der
openssl pkcs8 -topk8 -in $TEMPDIR/key.pem -nocrypt -outform DER -out $TEMPDIR/key.der
echo key.public=$(xxd -p $TEMPDIR/key.der | tr -d '\n') > $TEMPDIR/secret.yml
echo key.private=$(xxd -p $TEMPDIR/pubkey.der | tr -d '\n') >> $TEMPDIR/secret.yml
kubectl create secret generic activation-key --from-file=activation
-secret=$TEMPDIR/secret.yml -n $NAMESPACE
rm $TEMPDIR/key.pem
rm $TEMPDIR/pubkey.der
rm $TEMPDIR/key.der
rm $TEMPDIR/secret.yml
rmdir $TEMPDIR
```

1.3.1. Use Docker secret

If you store your webarchive image in a private registry and *not* in the Gitlab registry, you must create a **docker secret** which is referenced in the Helm values file.

Use the command below as a template for creating the docker secret:

```
kubectl create secret docker-registry docker-registry --docker
-server=https://index.docker.io/v1/ --docker-username=<username> --docker
-password='<password>' -n <env>
```

The `<env>` value must be set to the same value as in the Helm call.

1.4. Use Makefile for deployment

You can use a local `Makefile` to test your deployment before officially implementing the pipeline.

Your generated Docker images should ideally be handled via a CI/CD pipeline.

figure 3. Makefile

```
RELEASE=<env> ①
VALUES_FILE=values.yml ②
CHART_NAME=mironet/magnolia-helm
CHART_VERSION=v1.4.3 ③

# HELP
# This will output the help for each task
# thanks to https://marmelab.com/blog/2016/02/29/auto-documented-makefile.html
.PHONY: help

help:
@grep -E '^[a-zA-Z_-]+:.*?## .*$$' $(MAKEFILE_LIST) | sort | awk 'BEGIN {FS = ":.*?##
"}; {printf "\033[36m%-30s\033[0m %s\n", $$1, $$2}'

.DEFAULT_GOAL := help

values: ## Show generated yaml resources and values.
helm install --dry-run --debug -f $(VALUES_FILE) --version $(CHART_VERSION) --generate
-name $(CHART_NAME) -n $(RELEASE)

clean: ## Clean up environment.
helm del $(RELEASE) -n $(RELEASE)

clean-pvc: ## Clean disks (PVCs) too.
kubectl get persistentvolumeclaims -n $(RELEASE) -l 'release=$(RELEASE)' -o json |
kubectl delete -f -

install: ## Install helm chart on k8s.
helm upgrade --install $(RELEASE) $(CHART_NAME) --version $(CHART_VERSION) --create
-namespace -n $(RELEASE) -f $(VALUES_FILE)

upgrade: ## Upgrade locally deployed release.
helm upgrade $(RELEASE) $(CHART_NAME) --version $(CHART_VERSION) -n $(RELEASE) --reuse
-values -f $(VALUES_FILE)

test: ## Start helm tests.
helm test --logs $(RELEASE) -n $(RELEASE)

template: ## Template out, do not send to k8s.
helm template -f $(VALUES_FILE) $(CHART_NAME) --version $(CHART_VERSION) -n $(RELEASE)
```

Item	Notes
1	The release value is used for the Helm relesae name as well as the namespace in which the release was created. It should state the name of the environment of the release such as: • `dev` • `test` • `integration` Another cluster is installed for **production**.
2	Specifies the `values.yml` file that contains important configuration. See [Helm values] for more details.
3	Checks the latest chart version with the command: `helm search repo mironet/magnolia-helm`

Chapter 2. Add Cluster to GitLab project

In order for the project artifact to be deployed on your cluster, you must make sure that your cluster is connected to your GitLab repository correctly.

2.1. Prerequisites

- You must have a remote GitLab repository for your custom light module.
- You must have administrator privileges on your custom light module repository.
- You must have a Magnolia PaaS subscription.
- You must have an existing Rancher account and cluster.

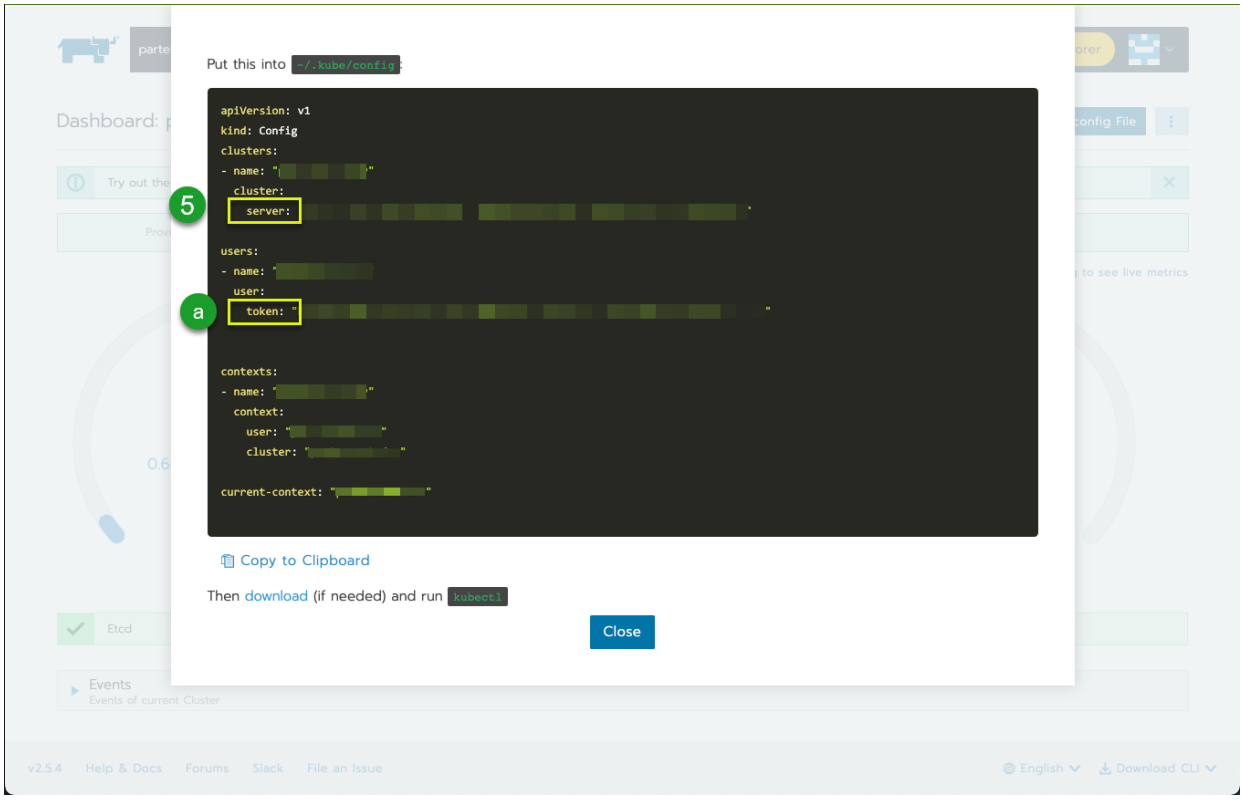
2.2. Instructions

To complete these instructions, you need to have both Rancher and Gitlab open.

> You should also have a text editor to hand.

2.2.1. Rancher steps

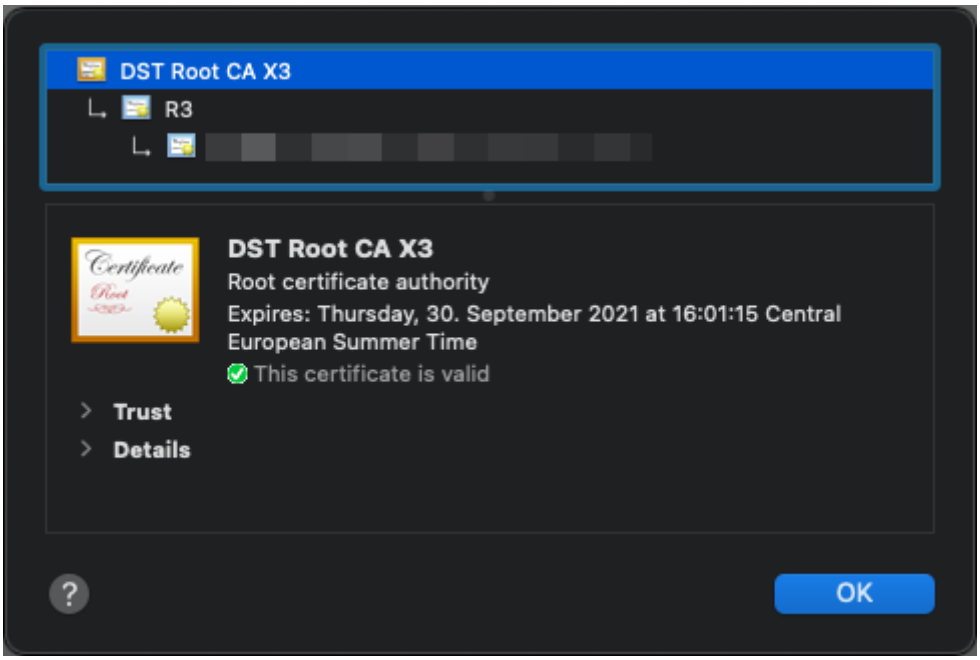
1. First thing's first, have your text editor open.
2. Log in to your Rancher account.
3. From the Global view, open the cluster you want to connect with your light module repo.
4. Open `Kubeconfig File`.
5. In the pop-up, copy the `server` URL. *Paste it into your text editor*.
    a. Also, copy the `token`. *Paste it into your text editor*.



Keep this handy for the Gitlab steps.

6. Now, in your browser, click the 🔒 next to the Rancher URL.

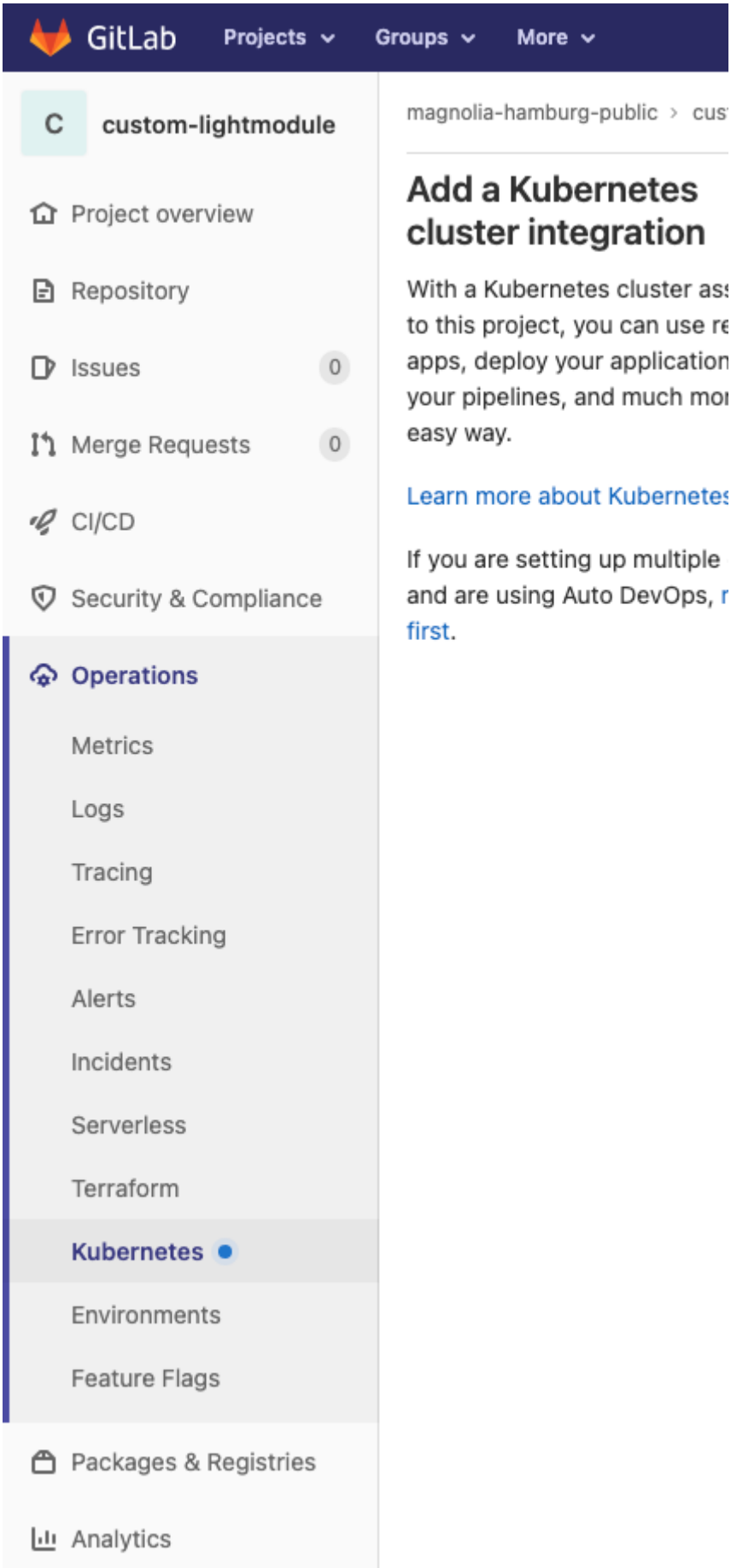
7. Select the **Root Certificate** and drag the certificate icon while holding `option` (MacOS) or `alt` (Windows) into your text editor.



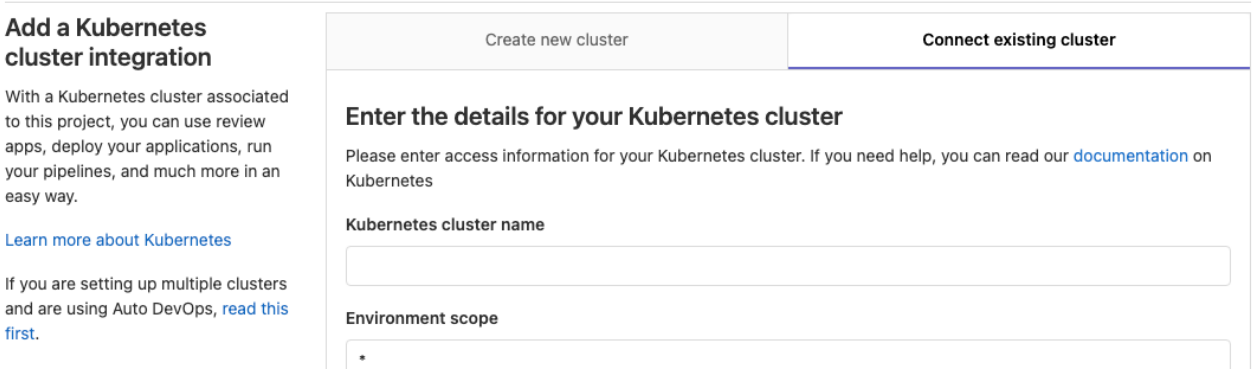
Keep this handy for the Gitlab steps.

2.2.2. Gitlab steps

1. Log in to your Gitlab account.
2. Navigate to your custom light module repository.
3. Go to **Operations** > **Kubernetes**.



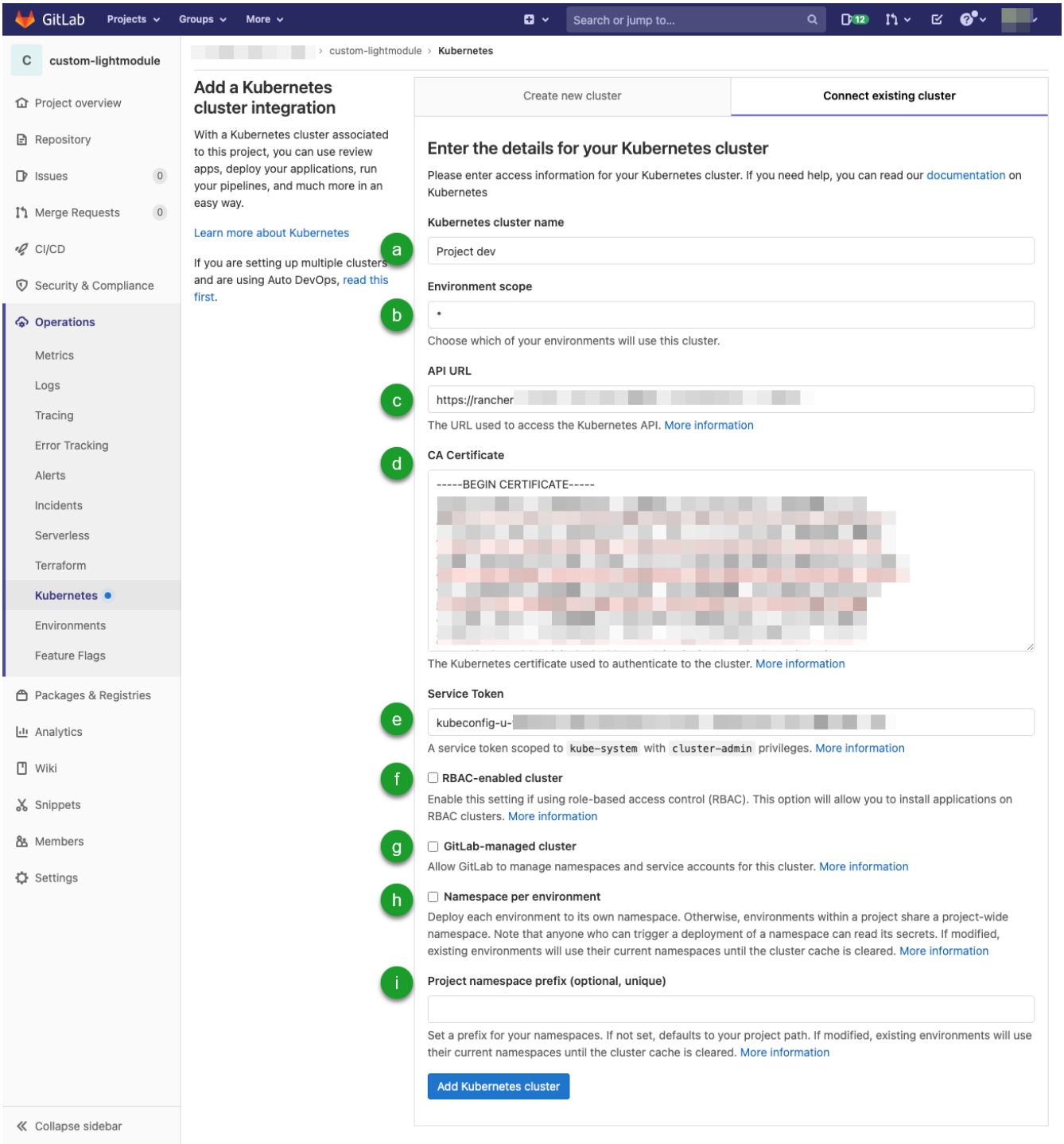
4. Choose the **Connect existing cluster** tab.



5. Fill in the details: (see the image below for help if needed)
   a. **Kubernetes cluster name**

   > Choose any name, but note that it should correspond to your Rancher cluster name.

   b. **Environment Scope** *Leave the asterisk.*
   c. **API URL** *Paste the* `server` *URL that you copied from Rancher.*
   d. **CA Certificate** *Paste the CA Certificate you took from the Rancher steps above.*
   e. **Service Token** *Paste the* `token` *you took from the Rancher steps above.*
   f. Uncheck **RBAC-enabled cluster**
   g. Uncheck **GitLab-managed cluster**
   h. Uncheck **Namespace per environment**
   i. Leave the **Project namespace prefix (optional unique)** empty.
6. Click **Add Kubernetes cluster**.



🚀 Ready for launch

The cluster is now ready for deployments.

Chapter 3. Webapp deployment

A Java Web Application (webapp) is a collection of servlets, other Java classes, static resources (*such as HTML pages*), other resources, and meta information that describes the webapp bundled together. The Java webapp in Magnolia PaaS has a typical structure.

figure 4. Webapp structure example

```
custom
 ├─── .gitignore
 ├─── .gitlab-ci.yml
 ├─── .m2
 │    └─── settings.xml
 ├─── README.md
 ├─── custom-module
 │    ├─── pom.xml
 │    ├─── src
 │    └─── target
 ├─── custom-webapp
 │    ├─── Dockerfile
 │    ├─── pom.xml
 │    ├─── src
 │    └─── target
 ├─── pom.xml
 └─── values.yml
```

3.1. The `.gitlab-ci.yml` file

It's important that you configure the `.gitlab-ci.yml` file correctly so that your development changes are picked up and deployed.

Magnolia automatically picks up the changes when using this approach.

Table 1. .gitlab-ci.yml descriptors

Item	Notes
1	The environment variables are set automatically by GitLab if the GitLab registry is used for the project. We recommend that you use GitLab. *env variables* • `$CI_REGISTRY_USER` • `$CI_REGISTRY_PASSWORD` • `$CI_REGISTRY`

Item	Notes
2	The `GIT_TAG` is used to set the tag for the created Docker image.
3	The first part of the URL ('integration' in this case) should match the DEPLOYMENT environment variable. The ingress creates the appropriate routes and since a wildcard DNS is used, the URLs are immediately accessible and secured by a certificate.
4	The `project-name` will be assigned by Magnolia at the start project.
5	The `deployment` must be triggered manually.

figure 5. .gitlab-ci.yml

```
# Use the latest Maven version

stages:
  - build
  - test
  - push
  - deploy

variables:
  MAVEN_OPTS: "-Dhttps.protocols=TLSv1.2
-Dmaven.repo.local=$CI_PROJECT_DIR/.m2/repository
-Dorg.slf4j.simpleLogger.log.org.apache.maven.cli.transfer.Slf4jMavenTransferListener=
WARN -Dorg.slf4j.simpleLogger.showDateTime=true -Djava.awt.headless=true"
  MAVEN_CLI_OPTS: "-s .m2/settings.xml --batch-mode --errors --fail-at-end --show
-version -DinstallAtEnd=true -DdeployAtEnd=true"


# Build the Maven project.
build-magnolia:
  image: maven:3.6-jdk-11-slim
  stage: build
  cache:
    key: "$CI_JOB_NAME"
    paths:
      - $CI_PROJECT_DIR/.m2/repository
  before_script:
    - mkdir -p $CI_PROJECT_DIR/.m2
  script:
    - mvn $MAVEN_CLI_OPTS package
    - ls -Fahl demo-cluster-webapp/target
  artifacts:
    expire_in: 30 days
    paths:
      - demo-cluster-webapp/target/*.war
```

```
# Execute the unit tests.
test:
  image: maven:3.6-jdk-11-slim
  stage: test
  cache:
    key: "$CI_JOB_NAME"
    paths:
      - $CI_PROJECT_DIR/.m2/repository
  before_script:
    - mkdir -p $CI_PROJECT_DIR/.m2
  script:
    - mvn $MAVEN_CLI_OPTS test

# Build docker images based on artifacts from the build stage.
push-docker-image:
  image: docker:19.03.12
  stage: push
  dependencies:
    - build-magnolia
  before_script:
    - apk add --no-cache git
    - git --version
    # for debugging
    #- export
    - docker login -u "$CI_REGISTRY_USER" -p "$CI_REGISTRY_PASSWORD" $CI_REGISTRY ①
    - export GIT_TAG=$(git describe --always) ②
    - export WEBAPP_IMAGE=${CI_REGISTRY_IMAGE}/magnolia-webapp
  script:
    - cd demo-cluster-webapp
    - docker build --pull -t "$WEBAPP_IMAGE:$GIT_TAG" .
    - docker push "$WEBAPP_IMAGE:$GIT_TAG"

.deploy:
  image: registry.gitlab.com/mironet/helm-kubectl-gomplate:v0.0.3
  stage: deploy
  before_script:
    - apk add --no-cache git
    - git --version
    - export GIT_TAG=$(git describe --always) ②
    - apk add --no-cache openssl
    - helm repo add mironet https://charts.mirohost.ch/

deploy-integration:
  extends: .deploy
  script:
    - export LE_ENVIRONMENT=letsencrypt-prod
    - export DEPLOYMENT=integration
    - export HELM_CHART_VERSION=1.4.3
    - cat values.yml | gomplate > ${DEPLOYMENT}.yml
    - cat ${DEPLOYMENT}.yml
    - helm upgrade --install ${DEPLOYMENT} mironet/magnolia-helm --version
```

```
${HELM_CHART_VERSION} --namespace ${DEPLOYMENT} --create-namespace -f
${DEPLOYMENT}.yml
  environment:
    name: integration
    url: https://integration.[project-name].magnolia-platform.com ③ ④
  when: manual ⑤

stop-integration:
  stage: deploy
  image: registry.gitlab.com/mironet/helm-kubectl-gomplate:v0.0.3
  script:
    - export DEPLOYMENT=integration
    - helm uninstall ${DEPLOYMENT} mironet/magnolia-helm --namespace ${DEPLOYMENT}
  when: manual ⑤
```

3.2. The `values.yml` file

The values.yml file will hold the configuration used by the Magnolia Helm Chart in the process of deploying the application to the cluster. Properties like `project-name` must be the same as in the .gitlab-ci.yml.

Table 2. values.yml descriptors

Item	Notes
1	For the development cluster the crawling of the public pages should by disallowed
2	The `project-name` will be assigned by Magnolia at the start project.
3	The password `secret-pw` must be the same in all 4 occurences.
4	The `license-email` and `license-key` will be provided by Magnolia.

figure 6. values.yml

```
# Values for Magnolia Helm chart...
ingress:
  enabled: true
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/proxy-body-size: 512m
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
    nginx.ingress.kubernetes.io/configuration-snippet: |
      more_set_headers "X-Robots-Tag: noindex, nofollow"; ①
  hosts:
    - host: {{ .Env.DEPLOYMENT }}.[project-name].magnolia-platform.com ②
      paths:
        - path: /
          instance: public
```

```
        - path: /author
          instance: author
  tls:
    - hosts:
        - {{ .Env.DEPLOYMENT }}.[project-name].magnolia-platform.com ②
      secretName: {{ .Env.DEPLOYMENT }}.[project-name].magnolia-platform.com ②
image:
  pullSecrets:
    - name: gitlab
  pullPolicy: Always
magnoliaAuthor:
  restartPolicy: Always
  redeploy: true
  contextPath: /author
  webarchive:
    repository: gitlab.[project-name].magnolia-platform.com/magnolia/base/magnolia-
webapp ②
    tag: {{ .Env.GIT_TAG | quote }}
  bootstrap:
    password: "[secret-pw]" ③
  activation:
    useExistingSecret: True
    secret:
      name: activation-key
      key: activation-secret
  env:
    - name: magnolia.superuser.enabled
      value: "true"
    - name: magnolia.superuser.password
      value: "[secret-pw]" ③
    - name: magnolia.bootstrap.license.owner
      value: [license-email] ④
    - name: magnolia.bootstrap.license.key
      value: [license-key] ④
  setenv:
    memory:
      maxPercentage: 80
  resources:
    requests:
      memory: 2Gi
    limits:
      memory: 2Gi
  livenessProbe:
    enabled: true
magnoliaPublic:
  restartPolicy: Always
  contextPath: /
  webarchive:
    repository: gitlab.[project-name].magnolia-platform.com/magnolia/base/magnolia-
webapp ②
    tag: {{ .Env.GIT_TAG | quote }}
```

```
  bootstrap:
    password: "[secret-pw]" ③
  activation:
    useExistingSecret: True
    secret:
      name: activation-key
      key: activation-secret
  env:
    - name: magnolia.superuser.enabled
      value: "true"
    - name: magnolia.superuser.password
      value: "[secret-pw]" ③
    - name: magnolia.bootstrap.license.owner
      value: [license-email] ④
    - name: magnolia.bootstrap.license.key
      value: [license-key] ④
  setenv:
    memory:
      maxPercentage: 80
  resources:
    requests:
      memory: 2Gi
    limits:
      memory: 2Gi
  livenessProbe:
    enabled: true
```

Chapter 4. Light Module deployment

Your light module project must contain a `light-modules` folder containing one or more light modules.

> The `.gitlab-ci.yml` will break if there are no light modules in the light module project.

> The example `.gitlab-ci.yml` does not contain the JavaScript build commands.

figure 7. Light module structure example

```
custom-lightmodule
|-- README.md
|-- .gitlab-ci.yml
|-- light-modules
    |-- custom-lightmodule
        |-- README.md
        |-- decorations
        |-- dialogs
        |   |-- components
        |   |-- pages
        |       |-- custom-page.yaml
        |-- i18n
        |   |-- custom-lightmodule-messages_en.properties
        |-- includes
        |   |-- README.txt
        |-- templates
        |   |-- components
        |   |-- pages
        |       |-- custom-page.ftl
        |       |-- custom-page.yaml
        |-- webresources
```

4.1. The `.gitlab-ci.yml` file

It's important that you configure the `.gitlab-ci.yml` file correctly so that your light development changes are picked up and deployed.

> Magnolia automatically picks up the changes when using this approach.

Table 3. gitlab-ci.yml descriptors

Item	Notes
1	The `deploy stage` first collects all pods running author or public instances and uses the devspace tool to rsync light module files to the running pods.

Item	Notes
2	The `environment name` must correspond to the environment name you used in your Helm deployment, the `namespace` is derived from the environment name.
3	The `deployment` must be triggered manually.

figure 8. gitlab-ci.yml

```
stages:
  - deploy ①

deploy:integration:sync:
  stage: deploy
  image: devspacesh/devspace:5
  cache:
    # We need this for syncing light modules into multiple pods.
    key: podlist-$CI_COMMIT_REF_SLUG
    paths:
      - pods.txt
    policy: pull
  environment:
    name: dev ②
  before_script:
    - export KUBECTL_NAMESPACE=${CI_ENVIRONMENT_SLUG} ②
    - export LIGHT_MODULES_CONTAINER_PATH=/mgnl-home/modules
  script:
    - kubectl -n $KUBECTL_NAMESPACE get pods -l "release=$KUBECTL_NAMESPACE,tier=app"
-o name | sed 's/^pod\///' > pods.txt
    - >
      for pod in `cat pods.txt`; do
        devspace sync -n $KUBECTL_NAMESPACE --local-path light-modules/ --pod $pod -c
magnolia-helm --container-path=$LIGHT_MODULES_CONTAINER_PATH --initial-sync
mirrorLocal --no-watch --upload-only
      done
  when: manual ③
```

Appendix A: Helm Values reference

This page contains a reference table for Magnolia helm values.

Key	Type	Default	Description
`bootstrap.enabled`	bool	`true`	Do enable bootstrapping via REST.
`deploy.directory`	string	`"/usr/local/tomcat/webapps"`	Deploy into this directory innside the app server container.
`deploy.securityContext.fsGroup`	int	`1000`	Fixup file permissions for volumes mounted to the Magnolia pod.
`deploy.securityContext.runAsGroup`	int	`1000`	Group ID.
`deploy.securityContext.runAsUser`	int	`1000`	Run application pod under this user ID. Do not use a privileged user here.
`deploy.tempDir`	string	`"/usr/local/tomcat/temp"`	
`fullnameOverride`	string	`""`	
`image.pullSecrets`	list	`[]`	
`image.tomcat.pullPolicy`	string	`"IfNotPresent"`	Tomcat repo pull policy.
`image.tomcat.repository`	string	`"tomcat"`	The tomcat image we're going to use.
`image.tomcat.tag`	string	`"9-jre11-slim"`	Tomcat repo tag.
`ingress.annotations`	object	\{}	Additional annotations for the ingress object.
`ingress.enabled`	bool	`false`	Enable/disable ingress configuration.
`ingress.hosts`	list	`[]`	Specify hosts here as an array.
`ingress.tls`	list	`[]`	
`jars[0].env[0].name`	string	`"INIT_DEST"`	

Key	Type	Default	Description
`jars[0].env[0].value`	string	`"/app/magnolia/WEB-INF/lib"`	
`jars[0].initScript`	string	`"/init.sh"`	Where to find the init script which copies .jar files into tomcat/lib.
`jars[0].name`	string	`"postgres-jdbc"`	
`jars[0].repository`	string	`"registry.gitlab.com/mironet/magnolia-jar/postgres-42.2.8"`	Example of additional jar, here the Postgres JDBC driver.
`jars[0].tag`	string	`"v0.0.1"`	
`magnoliaAuthor`	object	See values below ...	This is the author's configuration. It should not use H2 data base (the default).
`magnoliaAuthor.activation.useExistingSecret`	bool	`false`	Set this to `true` in case you want to use an existing activation key stored as a secret and provide its name.
`magnoliaAuthor.bootstrap.instructions`	string	`""`	Verbatim content of the instructions for this instance. If empty use a default. This is intended to be used with the –set-file flag of ``helm install''.
`magnoliaAuthor.catalinaExtraEnv`	object	`{}`	These key/value pairs will be added to CATALINA_OPTS.
`magnoliaAuthor.contextPath`	string	`"/author"`	The context path of this Magnolia instance. Always use a leading slash.
`magnoliaAuthor.db.backup.enabled`	bool	`false`	Enable db backup sidecar.
`magnoliaAuthor.db.persistence.mountPath`	string	`"/db"`	Mount point is /db, PGDATA=/db/data
`magnoliaAuthor.db.persistence.subPath`	string	`"data"`	Mount point is /db, PGDATA=/db/data

Key	Type	Default	Description
`magnoliaAuthor.extraContainers`	list	`[]`	Extra sidecar containers added to the Magnolia pod.
`magnoliaAuthor.extraInitContainers`	list	`[]`	Extra init containers added to the Magnolia pod.
`magnoliaAuthor.jndiResources`	list	`[]`	Additional JDNI resources to be added in tomcat's `server.xml`. The key/value pairs will be mapped to xml.
`magnoliaAuthor.persistence.enabled`	bool	`true`	Enable persistence for indexes, cache, tmp files. If this is enabled the MGNL_HOME_DIR env var will be set and a volume will be mounted to the default location unless it's specified here as mountPath.
`magnoliaAuthor.persistence.existingClaim`	string	`nil`	Existing volumes can be mounted into the container. If not specified, helm will create a new PVC.
`magnoliaAuthor.persistence.size`	string	`"10Gi"`	In case of local-path provisioners this is not enforced.
`magnoliaAuthor.persistence.storageClassName`	string	`""`	Empty string means: Use the default storage class.
`magnoliaAuthor.podAnnotations`	object	`{}`	Custom annotations added to pod.
`magnoliaAuthor.redeploy`	bool	`false`	If true, redeploy on ``helm upgrade/install'' even if no changes were made.
`magnoliaAuthor.rescueMode`	bool	`false`	Enable Groovy rescue console.
`magnoliaAuthor.resources.limits.memory`	string	`"512Mi"`	Maximum amount of memory this pod is allowed to use. This is not the heap size, the heap size is smaller, see `setenv.memory` for details.
`magnoliaAuthor.resources.requests.memory`	string	`"512Mi"`	Minimum amount of memory this pod requests.

Key	Type	Default	Description
`magnoliaAuthor.setenv.memory.maxPercentage`	int	`60`	Maximum amount allocated to heap as a percentage of the pod's resources.
`magnoliaAuthor.setenv.memory.minPercentage`	int	`25`	Minimum amount allocated to heap as a percentage of the pod's resources.
`magnoliaAuthor.setenv.update.auto`	string	`"true"`	Auto-update Magnolia if repositories are empty (usually on the first run).
`magnoliaAuthor.strategy.type`	string	`"Recreate"`	Kubernetes rollout strategy on `helm upgrade …`.
`magnoliaAuthor.webarchive.repository`	string	`"registry.gitlab.com/mironet/magnolia-demo"`	The docker image where to fetch compiled Magnolia libs from.
`magnoliaAuthor.webarchive.tag`	string	`"latest"`	Do not use `latest' in production.
`magnoliaPublic`	object	See values below …	This is the public instance.
`magnoliaPublic.activation.useExistingSecret`	bool	`false`	Set this to `true` in case you want to use an existing activation key stored as a secret and provide its name.
`magnoliaPublic.bootstrap.instructions`	string	`""`	Verbatim content of the instructions for this instance. If empty use a default. This is intended to be used with the –set-file flag of ``helm install''.
`magnoliaPublic.catalinaExtraEnv`	object	`{}`	These key/value pairs will be added to CATALINA_OPTS.
`magnoliaPublic.contextPath`	string	`"/"`	The context path of this Magnolia instance. Always use a leading slash.
`magnoliaPublic.db.backup.enabled`	bool	`false`	Enable db backup sidecar.
`magnoliaPublic.db.contentsync.address`	string	`":9998"`	TLS port of the backup sidecar.

Key	Type	Default	Description
`magnoliaPublic.db.contentsync.enabled`	bool	`true`	Enable content sync on public instances. Depends on the backup being enabled and configured correctly for pg_wal log shipping.
`magnoliaPublic.db.persistence.mountPath`	string	`"/db"`	Mount point is /db, PGDATA=/db/data
`magnoliaPublic.db.persistence.subPath`	string	`"data"`	Mount point is /db, PGDATA=/db/data
`magnoliaPublic.extraContainers`	list	`[]`	Extra sidecar containers added to the Magnolia pod.
`magnoliaPublic.extraInitContainers`	list	`[]`	Extra init containers added to the Magnolia pod.
`magnoliaPublic.jndiResources`	list	`[]`	Additional JDNI resources to be added in tomcat's `server.xml`. The key/value pairs will be mapped to xml.
`magnoliaPublic.persistence.enabled`	bool	`true`	Enable persistence for indexes, cache, tmp files. If this is enabled the MGNL_HOME_DIR env var will be set and a volume will be mounted to the default location unless it's specified here as mountPath.
`magnoliaPublic.persistence.existingClaim`	string	`nil`	Existing volumes can be mounted into the container. If not specified, helm will create a new PVC.
`magnoliaPublic.persistence.size`	string	`"10Gi"`	In case of local-path provisioners this is not enforced.
`magnoliaPublic.persistence.storageClassName`	string	`""`	Empty string means: Use the default storage class.
`magnoliaPublic.podAnnotations`	object	`{}`	Custom annotations added to pods.
`magnoliaPublic.redeploy`	bool	`true`	If true, redeploy on ``helm upgrade/install'' even if no changes were made.
`magnoliaPublic.replicas`	int	`1`	How many public instances to deploy.

Key	Type	Default	Description
`magnoliaPublic.rescueMode`	bool	`false`	Enable Groovy rescue console.
`magnoliaPublic.resources.limits.memory`	string	`"512Mi"`	Maximum amount of memory this pod is allowed to use. This is not the heap size, the heap size is smaller, see `setenv.memory` for details.
`magnoliaPublic.resources.requests.memory`	string	`"512Mi"`	Minimum amount of memory this pod requests.
`magnoliaPublic.setenv.memory.maxPercentage`	int	`60`	Maximum amount allocated to heap as a percentage of the pod's resources.
`magnoliaPublic.setenv.memory.minPercentage`	int	`25`	Minimum amount allocated to heap as a percentage of the pod's resources.
`magnoliaPublic.setenv.update.auto`	string	`"true"`	Auto-update Magnolia if repositories are empty (usually on the first run).
`magnoliaPublic.strategy.type`	string	`"Recreate"`	Kubernetes rollout strategy on `helm upgrade …`.
`magnoliaPublic.webarchive.repository`	string	`"registry.gitlab.com/mironet/magnolia-demo"`	The docker image where to fetch compiled Magnolia libs from.
`magnoliaPublic.webarchive.tag`	string	`"latest"`	Do not use `latest' in production.
`nameOverride`	string	`""`	
`postjob.image`	string	`"registry.gitlab.com/mironet/magnolia-bootstrap"`	Where to get the bootstrapper from. This should not be changed under normal circumstances.
`postjob.imagePullPolicy`	string	`"IfNotPresent"`	
`postjob.tag`	string	`"v0.2.2"`	
`postjob.waitFor`	string	`"10m"`	
`service.annotations`	object	`{}`	
`service.clusterIP`	string	`"None"`	

Key	Type	Default	Description
`service.ports[0].name`	string	`"http"`	
`service.ports[0].port`	int	`80`	
`service.ports[0].protocol`	string	`"TCP"`	
`service.ports[0].targetPort`	int	`8080`	
`service.type`	string	`"ClusterIP"`	
`sharedDb`	object	See values below ...	Shared database (jackrabbit ``clustering'').
`sharedDb.db.persistence.subPath`	string	`"data"`	Mount point is /db, PGDATA=/db/data
`sharedDb.enabled`	bool	`false`	Enable shared db
`timezone`	string	`"Europe/Zurich"`	Timezone for Magnolia.

magnolia®