Container platform installation

V 1.5, 21.06.2022

Content

1 Provided Repositories

This document describes possibilities for running Series M/ products on Container platforms.

We provide set of images which can be used to deploy our applications into container environment. Images are available in Docker Registry https://registry.kwsoft.cloud under Project kwsoft. You could create your own account on this page under "Sign up for an account". Please create only one technical user for the whole company. When the sign up procedure is completed, you could request K&W hotline for access rights in Harbor (registry) for the project kwsoft.

Under the Project you will find several Repositories with Docker Images, which can be combined at the installation of the product.

1.1 kwsoft/seriem613

The most important Images are in seriem repository. The tag of the image is the last part of the M/TEXT version number. For example, if the M/TEXT version is 6.13.0.231-hotfix5, there will be a seriem613 image with the tag **231-hotfix5**. Every seriem image contains all resources which are bound to one product version. There is a seriem.ear with standard modules (M/OMS, M/DO, M/TEXT, ContentHub), as well clients.zip file with M/Workbench, M/TEXT Daemon Client, etc. The Image is used as initial container. Its job is the preparation of resources for other Containers and copying of necessary files on a Volume. It is possible to influence this process and for example add custom jars into the ear by copying jars int mtext-init directory on volume. Details of volume mapping and available content are described in Appendix 2 - volumeMounts kwsoft/seriem612.

1.2 kwsoft/wildfly26_init

This container is used for initialization of volumes for Wildfly. When the configuration folder is empty, the default setting files are copied.

1.3 kwsoft/wildfly26

This container contains installation of Wildfly 26, which is ready to deploy our application. Wildfly uses the default port mapping, so http is on port 8080, https 8443 and wildfly console on 9990.

There are following four database modules installed in Wildfly:

- **PostgreSQL**, postgresql-42.3.4 - Supports PostgreSQL 8.2 or newer and requires Java 8 or newer.
- **Oracle**, ojdbc11 – Supports Oracle 21c, 19c, 18c, and 12.2
- **DB2**, jcc-11.5.7.0
- **MSSQL**, mssql-jdbc-10.2.0.jre8.jar

The container can be configured by environment variables. It is for example possible to configure CONNECTION_URL, USER_NAME, PASSWORD of external database and wildfly will use this database for data sources. For more details see Appendix 1 - Environment Variables.

1.4 kwsoft/postgres12

This is the standard PostgreSQL Docker image without any data. When the container is started and there is some content on the volume in folder /postgres-init, ddl scripts located in that folder will be executed. to create the schema mtext and all tables, which are needed for the deployment of the product.

1.5 kwsoft/svnserver

Contains the Subversion server with preconfigured Authorization user=mtext, password=mtext. You can create a new user by executing the following command in container shell:

```
htpasswd /etc/subversion/passwd <username>
```

1.6 kwsoft/gitserver

Contains the GIT server with preconfigured Authorization user=mtext, password=mtext.

```
htpasswd /var/lib/git/passwd <username>
```

There are many possibilities how Docker images can be used to build a running environment. We will describe most common scenarios in the next chapters.

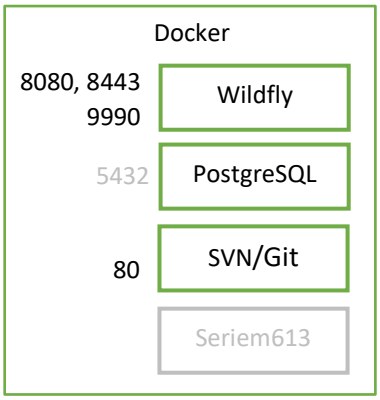
2 Infrastructure architecture

Before we start with the installation, we should decide, which existing parts of the infrastructure will be used, and which parts should be provided by the installation.

2.1 Server topology

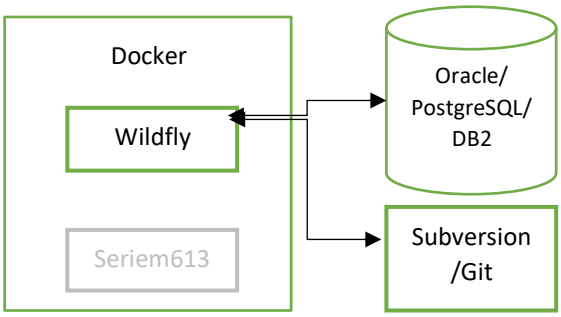
2.1.1 Standalone server

Standalone server is all in one environment, which uses its own application server (kwsoft/wildfly26), database (kwsoft/postgres) and version control system (kwsoft/svnserver, kwsoft/gitserver). Database and all tables will be in this scenario created automatically from scripts, which are mapped from folder postgres-init on volume (see kwsoft/postgres).



2.1.2 Integrated server

In many cases it is reasonable to integrate the server into some existing infrastructure like Database or version control system, which will be used instead of the provided docker image. For version control nothing special needs to be done, we just need network access. The Database is configured by environment variables CONNECTION_URL, DRIVER_CLASS, USER_NAME , PASSWORD and DB_SCHEMA. In this case you have to take care about creating of database and tables. Database scripts are located in wildfly container under /home/seriem/dbchecker/DDL.



3 Installation on target platform

Docker images can be used many ways. We describe most usual usage scenarios.

3.1 Install on Kubernetes using Helm Chart

Helm is the package manager for Kubernetes and is recommended for installation on this container platform.

Current versions of helm charts are available at this address:
https://registry.kwsoft.cloud/harbor/projects/3/helm-charts/seriem/versionshttps://registry.k/harbor/projects/3/helm-charts/seriem/versions

3.1.1 Prerequisities

Kubernetes Cluster, for example https://aws.amazon.com/de/eks/, tested version 1.21

kubectl, https://kubernetes.io/docs/tasks/tools/, tested with version 1.21

helm, https://helm.sh/docs/intro/install/ tested version 3.7.1

3.1.2 Installation procedure, standalone server

Add a new chart repository. You must enter the credentials for registry.kwsoft.cloud as mentioned in chapter 1.

```
helm repo add kwsoft https://registry.kwsoft.cloud/chartrepo/kwsoft \
--username xxx
```

As result you should see:

```
"kwsoft" has been added to your repositories
```

Start standalone server by calling upgrade/install of the chart (you must re-enter credentials for the registry in `imageCredentials`).

Remark: You should use unique deployment name instead of myserver, own namespace instead of mynamespace and real dns domain, where the server will be available instead of mydomain.com.

```
helm upgrade --install myserver kwsoft/seriem --namespace mynamespace \
--create-namespace --set general.hostDomain=mydomain.com \
--set imageCredentials.username=xxx --set imageCredentials.password=xxx
```

Check if the Chart was successfully installed by calling:

```
helm list -n mynamespace
NAME    NAMESPACE REVISION UPDATED       STATUS   CHART        APP VERSION
myserver  mynamespace    1        2021-11-22…   deployed seriem-0.3.1
6.12.404-hotfix3
```

Then a pod is automatically created in the Kubernetes cluster. You can check it by calling:

```
kubectl get pods -n mynamespace
NAME                          READY   STATUS    RESTARTS   AGE
dmemyserver-8cb44c5f9-f2qml   3/3     Running   0          2m29s
```

If your server is running, you should be able to open welcome page as described in Section 3.5. The URL where your server should be available is constructed from deployment name and

general.hostDomain value as for example http://myserver.mydomain.com. It is of course required to have corresponding dns entry to be able to access the url in internet. It is recommended to put into dns wildcart record like *.mydomain.com, which will be routed to your loadballancer or created ingress instance. You can list created ingress instance by calling

```
kubectl get ingress -n mynamespace
NAME          CLASS    HOSTS                   ADDRESS   PORTS   AGE
dmemyserver   <none>   myserver.mydomain.com   1.2.3.4   80      18d
```

3.1.3 Installation of Integrated server

Example of starting the server with external database and external team support storage:

```
helm upgrade --install myserver kwsoft/seriem --namespace mynamespace --
create-namespace --set general.hosDomain=mydomain.com --set
imageCredentials.username=myuser --set imageCredentials.password=mypassword --
set imageCredentials.email=myemail --set general.createPostgresPod="" --set
general.createSVNPod="" --set
wildfly.database.connectionUrl=jdbc:postgresql://mydbserver:5432/mydatabase --
set wildfly.database.userName=mydbuser --set
wildfly.database.password=mydbpassword
```

3.1.4 Installation as cluster

We provide also helm chart for dynamically scalable cluster called seriem-cluster. This kind of deployment separates M/TEXT and M/OMS functionality into different pod types. The separation is not strict, but long running processes like Stack runs are located only on M/OMS pod type. Cluster uses Horizontal Pod Autoscaling, which you could setup with Values like `mtext.minReplicas`, `mtext.maxReplicas`.

For M/TEXT pod type, there are teoreticly no restrictions for scaling. Maximal reasonable sizing is limited mainly by database and is arround maxReplicas=10.

It is not recommended to scale M/OMS type dynamicly, as there should not be long running processes in the time of scaling. So it is recommended to setup same Values to moms.minReplicas and moms.maxReplicas.

If you plan to scale the cluster over more server nodes (typical usage) you have to use persistent volume shared between several nodes. This means, you should setup storage class (Value `general.storageClass`), which can provide such type of Persistent Volumes for example EFS or NFS.

Example of starting the server with external Postgres database and external team support storage as cluster with separately scalable pods for M/TEXT and M/OMS. Example is runnable also on OpenShift with restricted SCP:

```
helm upgrade --install myname kwsoft/seriem-cluster --namespace mynamespace --
create-namespace --set general.hosDomain=mydomain.com --set
imageCredentials.username=myuser --set imageCredentials.password=mypassword --
set imageCredentials.email=myemail --set general.createPostgresPod="" --set
general.createSVNPod="" --set
wildfly.database.connectionUrl=jdbc:postgresql://mydbserver:5432/mydatabase --
set wildfly.database.userName=mydbuser --set
wildfly.database.password=mydbpassword --set mtext.maxReplicas=2 --set
moms.maxReplicas=1 --set-string general.fsGroup="" –set
general.storageClass=efs-sc
```

3.1.5 Customizations

As you can see in examples above, we use the set parameter (**--set**) to overwrite helm values as needed. You can use own values.yaml as well, if you set large amount of values. The complete listing of values can be found in chapter *9 Appendix Helm Values* or in Harbor under tab Values.

3.2 Docker compose

Probably the easiest way to run a Docker based local environment is by usage of docker-compose. It is available for Windows as well as for Linux. This kind of installation is mainly suitable for smaller installations and testing purposes.

You need to define services in a Compose file called docker-compose.yml in your project directory. We provide a docker-compose.yaml file with preconfigured settings for standalone server topology. YAML files are called definition files since they define the service that you are deploying.

The user may adjust some settings in yaml to the target environment like port number or version number.

Example:

```
wildfly:
     image: registry.kwsoft.cloud/kwsoft/wildfly26:4
     ports:
     - "8080:8080"
     environment:
     - EXTERNAL_HOST_NAME=localhost
     - EXTERNAL_HOST_PORT=8080
     volumes:
     - "./volumes/clients:/clients:rw"
     - "./volumes/deployments:/deployments:rw"
     - "./volumes/seriem-home:/seriem:rw"


seriem:
     image: registry.kwsoft.cloud/kwsoft/seriem613:256
     environment:
     - EXTERNAL_HOST_NAME=localhost
     - EXTERNAL_HOST_PORT=8080
     volumes:
     - "./volumes/clients:/clients:rw"
     - "./volumes/deployments:/deployments:rw"
     - "./volumes/seriem-home:/seriem:rw"
```

For the complete set of Environment Variables see Appendix 1 and a full docker-compose.yaml file in Appendix 3.

As a first step, create a project folder for the application and save the docker-compose.yaml file in it.

To ensure the access to the docker registry for downloading the images enter the command:

`docker login https://registry.kwsoft.cloud/harbor`

Enter username and password to complete the login.

Create a folder called volumes, where docker volumes will be placed. It is on the same level as the docker-compose.yml file.

From your project directory, start up your application by running :

`docker-compose up`

Compose pulls an image, builds an image for your code, and starts the services you defined. In this case, the code is statically copied into the image at build time.

When containers are started successfully, the welcome page is reachable from the root of the external hostname. For example: *http://localhost:8080*.

3.2.1 Linux

On linux operating systems, it is necessary to create a folder called volumes, where docker volumes will be placed, before running initial docker compose up, so that the folders are not automatically created under root user. The volumes folder is located at the docker-compose.yaml level. You can use this script:

create_docker_compose.sh

```
#!/bin/sh
mkdir -p volumes/postgres-data
mkdir -p volumes/postgres-init
mkdir -p volumes/deployments
mkdir -p volumes/seriem-home
mkdir -p volumes/jboss-config
mkdir -p volumes/clients
mkdir -p volumes/svn-data
mkdir -p volumes/svn-config
mkdir -p volumes/mtext-init
docker-compose up
```

3.3 Kubernetes

If you don't want to use helm in combination with kubernetes, you can create directly a yaml that describes kubernetes objects. Find here the main parts of the YAML file:

```
initContainers:
        - name: seriem
        image: registry.kwsoft.cloud/kwsoft/seriem612:311-hotfix2
        env:
```

```
        - name: EXTERNAL_HOST_NAME
          value: example.url
        - name: EXTERNAL_HOST_PORT
          value: "443"
        volumeMounts:
          - mountPath: /home/clients"
            name: ebsvolume
            subPath: dmeRelease/clients
          - mountPath: /opt/jboss/"
            name: ebsvolume
            subPath: dmeRelease/
          - mountPath: /"
            name: ebsvolume
            subPath: dmeRelease/
          - "./volumes/deployments:/deployments:rw"
          - "./volumes/seriem-home:/seriem:rw"

containers:
        - name: wildfly
        image: registry.kwsoft.cloud/kwsoft/wildfly16:4
        env:
        - name: EXTERNAL_HOST_NAME
          value: example.url
        - name: EXTERNAL_HOST_PORT
          value: "443"

volumes:
- name: ebsvolume
  emptyDir: {}
```

Full YAML is showed in chapter 8. Appendix 4 – Full example of Kubernetes pod yaml definition.

After assembling the yaml, the pod can be started by this command:

```
kubectl apply -f example.yaml
```

When containers are started successfully, the welcome page is reachable from the root of the external hostname. For example: *https:*//example.url

3.3.1 Secured connection using HTTPS

For security reasons we recommend using of HTTPS for accessing of server. This can be done on different places.

3.3.1.1 Terminating HTTPS (TLS) connections on LoadBalancer

Usual way is terminating of HTTPS on Company or Cloud provider LoadBallancer. Between LB and server could be already used HTTP connection, because this is understood as internal DMZ communication. On following link is an example how to configure such LB on AWS.

https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-create-https-ssl-load-balancer.html

3.3.1.2 Terminating HTTPS on Ingress

Another way to terminating of HTTPS is on Ingress. The certificate can be created manually using a yaml configuration of secret that contains a TLS private key and certificate. Private key and certificate must be created for the server domain and encoded with base64.

Yaml file example:

```
apiVersion: v1
       data:
              tls.crt: "base64 encoded cert"
```

```
             tls.key: "base64 encoded key"
kind: Secret
metadata:
       name: secretName
       namespace: namespaceName
type: kubernetes.io/tls
```

Created secret can be used in helm value ingress.tls.secretNameTls.

CertManager (https://cert-manager.io/docs/) can be used to automatically create and renew certificates.

3.3.1.3 Wildfly elytron security

If you want to secure complete communication channel, it is possible set HTTPS certificate in Widfly Container by elytron security.

https://docs.wildfly.org/18/WildFly_Elytron_Security.html#configure-ssltls

3.4 OpenShift

OpenShift is a commercial distribution of Kubernetes from RedHat, that contains additional features which are not available from the open-source project. The configuration is the same as for Kubernetes and installation using helm is possible. However you should be aware of problems with Security Context Policy, because OpenShift default SCP is restricted and do not support usage of root UID, fsGroup etc.

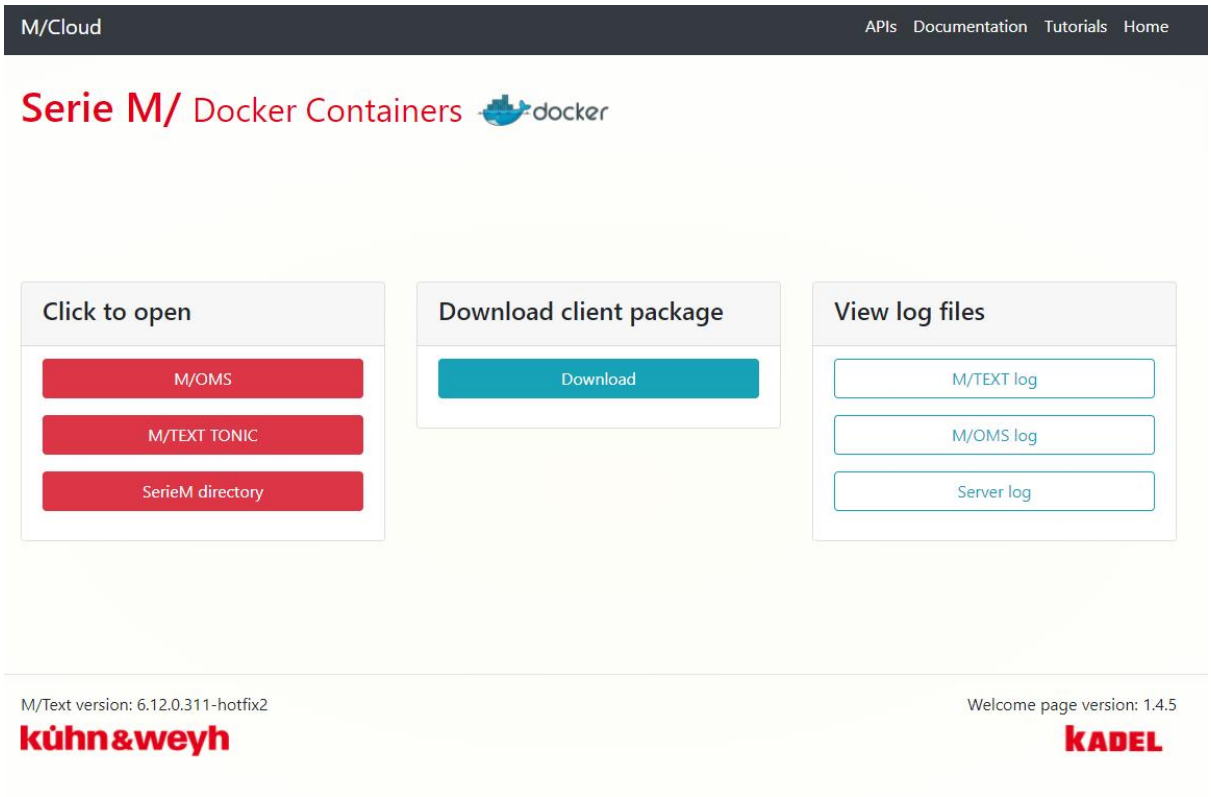
Example of large scale production environment in cluster mode using integrated infrastracture scenario you can find under section 3.1.4 Installation as cluster.

We do not recommend to use standalone server topology on OpenShift. Installation including postgres and svn image (standalone) would request root UID. You would need to configure less restrictive policy to be able to start pods. More info regarding SCC: https://cloud.redhat.com/blog/managing-sccs-in-openshift

3.5 Welcome page

The Welcome page is automatically deployed after the start of the server instance. This page serves as a crossroad and is available in the root of the domain.

For example: *http://localhost:8080/*.



In the left part are links to running services M/OMS Cockpit, M/TEXT TONIC, directory listing on server, SVN or GIT.

In the middle of the page the download of the client package. It contains client-side tools and application. Such as M/Workbench, JEDI client, client jars and the tool for user synchronization.

And in the right part are the three available main log files: M/Text log, M/OMS log and server log. They are usually located on the server at: /home/seriem/server/log

4 Appendix 1 - Environment Variables

4.1 kwsoft/wildfly26

ENV variable	Default value	Description (options)
DRIVER_CLASS	postgresql	DB type (postgresql, oracle, db2, mssql)

4.2 kwsoft/wildfly26

ENV variable	Default value	Description (options)
EXTERNAL_HOST_NAME	127.0.0.1	JMS connection endpoint (docker host HOSTNAME)
EXTERNAL_HOST_PORT	8080	JMS connection endpoint port (docker host PORT)
JBOSS_CONFIG	standalone-full.xml	Wildfly server configuration (standalone-full.xml, standalone-full-ha.xml)
DRIVER_CLASS	postgresql	DB type (postgresql, oracle, db2, mssql)
CONNECTION_URL	jdbc:postgresql://db:5432/mtext	DB connection URL
USER_NAME	postgres	DB username
PASSWORD	postgres	DB password
DB_SCHEMA	mtext	DB schema used in server.ini
MTEXT_HOME	/home/seriem	Seriem home folder / CSHome
SERVER_INI	/home/seriem/ini/server.ini	Seriem configuration file / server.ini
ANOTHER_JAVA_OPTS	-	other JAVA OPTS if needed - agentlib:jdwp=transport=dt_socket,address=8787,server=y,suspend=y
WAIT_TIME	0	delay wildfly start if needed on slow machines
WS_HOST_NAME	localhost	DNS name of the server (example.eu)
WS_HOST_PORT	8080	Port to reach the deployment from outside the container
WS_HTTP_PROTOCOL	http	http / https
WS_HTTPS_PORT	443	Port for HTTPS protocols

CLUSTER_LABELS		Kubeping - labels for cluster members selection
CLUSTER_NAMESPACE		Kubeping - Namespace of the cluster
WEB_ACCESSED_DIR	/home/seriem/server	folder accessible under myserver/output
SHARED_DEPLOYMENTS_FOLDER	/opt/jboss/wildfly/standalone/deployments-shared/	EAR folder for cluster for cluster deployment

4.3 kwsoft/seriem613

ENV variable	Default value	Description (options)
ADMIN_TOOLS	false	delete admin and DDL
DAEMON_CLIENT	true	prepare daemon client
EXTERNAL_HOST_NAME	localhost	server name (docker host HOSTNAME)
EXTERNAL_HOST_PORT	8080	server port (docker host PORT)
WS_ONLY	true	Webservice only
MTEXT_ONLY	false	Only M/Text will be deployed
ONLY_DB_TOOLS	false	Only DbTools will be deployed

4.4 kwsoft/postgres12

ENV variable	Default value	Description (options)
SERVER_NAME	-	domain of the new server
SERVER_DESC	-	description of the new server
REPOSITORY_URL	-	url of repository

5 Appendix 2 - volumeMounts

5.1 kwsoft/seriem613

mountPath	subPath	description
/deployments	<volume-name>/deployments	Deployment's folder of Wildfly.
/clients	<volume-name>/clients	Contains clients.zip with M/Workbench, Client jars, etc..
/postgres-init	<volume-name>/postgres-init	Contains database ddl scripts, which will be executed at postgres image start.
/seriem	<volume-name>/seriem-home	Contains files like server.ini and M/Spell dictionaries.
/mtext-init	<volume-name>/mtext-init	This folder provides the possibility to inject customer specific parts into the product. The content of client (resp. ear) subfolder will be included in clients.zip (resp. seriem.ear). For example, if you need to deploy moms extension jar, it is enough to put the jar into /mtext-init/ear/lib folder. The content of the subfolder deployment will be copied to wildfly/standalone/deployments directory "as it is".

5.2 kwsoft/wildfly26_init

mountPath	subPath	description
/opt/jboss/iwldfly/standalone/configuration	<volume-name>/jboss-config	Saves all configuration files of jboss installation. In a Linux standard installation is this volume mapped to /opt/jboss/wildfly/standalone/configuration.

5.3 kwsoft/wildfly26

mountPath	subPath	description
/opt/jboss/iwldfly/standalone/configuration	<volume-name>/jboss-config	Saves all configuration files of jboss installation. In a Linux standard installation is this volume mapped to /opt/jboss/wildfly/standalone/configuration.
/home/seriem	<volume-name>/seriem-home	Contains files like server.ini and M/Spell dictionaries.
/opt/jboss/wildfly/standalone/deployments	<volume-name>/deployments	Deployments folder of Wildfly
/home/clients	<volume-name>/clients	Contains clients.zip with M/Workbench, JEDI client, client jars and tool for user synchronization
/opt/jboss/wildfly/standalone/configuration	<volume-name>/jboss-config	Wildfly configuration files
/opt/jboss/wildfly/standalone/log	<volume-name>/jboss-log	Wildfly log files

/home/mtext-init	<volume-name>/mtext-init	This folder provides the possibility to inject customer specific parts into product. The content of client (resp. ear) subfolder will be included in clients.zip (resp. seriem.ear). For example, if you need to deploy moms extension jar, it is enough to put the jar into /mtext-init/ear/lib folder. The content of the subfolder deployment will be copied to wildfly/standalone/deployments directory "as it is".

5.4 kwsoft/svn

mountPath	subPath	description
/var/opt/svn/	<volume-name>/svn-data	SVN data folder
/etc/subversion	<volume-name>/svn-config	SVN configuration folder

5.5 kwsoft/git

mountPath	subPath	description
/var/lib/git	<volume-name>/git-data	GIT data folder

5.6 kwsoft/postgres12

mountPath	subPath	description
/var/lib/postgresql/data	<volume-name>/postgres-data	This location keeps a database data. In a Linux installation this volume is mapped to */var/lib/postgresql/data/*
/docker-entrypoint-initdb.d	<volume-name>/postgres-init	Contains SQL scripts for the initialization or updating of the database.

6 Appendix 3 – Full example of docker-compose.yaml

Example configuration of docker-compose.yaml. It contains:

- Wildlfy with product, exposed on port 8080
- Postgres database exposed on port 5432
- SVN exposed on port 80

```
# Docker M/Express - Docker Compose
#
# Volumes folder is created next to this file, which contains mapped
# dockervolumes
# Welcome page is reachable at http://localhost:8080/
#
#  user rights must be changed manually on linux machines
#
#  cd volumes
#  chown -R 1000:0 clients/ deployments/ jboss-config/ mtext-init/ seriem-
home/
#  docker-compose down and up
version: '3'
services:
    seriem:  # Container is used for volumes initialization only - exits immed
iately.
        image: registry.kwsoft.cloud/kwsoft/seriem613:256
        entrypoint: /home/volumes_init.sh
        environment:
        - EXTERNAL_HOST_NAME=localhost
        - EXTERNAL_HOST_PORT=8080
        - DEFAULT_WORKSPACE=false
        - WS_ONLY=false
        volumes:
        - "./volumes/clients:/clients:rw"
        - "./volumes/deployments:/deployments:rw"
        - "./volumes/seriem-home:/seriem:rw"
        - "postgres-init:/home/postgres_init:rw"
        - "./volumes/mtext-init:/mtext-init:rw"
    wildfly-
init: # Container is used for initialization of wildfly configuration - exits
immediately.
        image: registry.kwsoft.cloud/kwsoft/wildfly26_init:1
        environment:
        - DRIVER_CLASS=postgresql
        volumes:
        - "./volumes/jboss-config:/configuration"
    postgres: # Database container
        image: registry.kwsoft.cloud/kwsoft/postgres
        ports:
        - "5432"
```

```
        volumes:
        - "postgres-init:/docker-entrypoint-initdb.d"
        - "./volumes/postgres-data:/var/lib/postgresql/data/"
        depends_on:
        - seriem
        environment:
        - SERVER_NAME="localhost"
        - SERVER_DESC="SerieM docker-compose server"
        - REPOSITORY_URL="http://svn/svn/seriem"
    svn: # SVN container
        image: registry.kwsoft.cloud/kwsoft/svnserver:1
        ports:
        - "80:80"
        volumes:
        - "./volumes/svn-data:/var/opt/svn/"
        - "./volumes/svn-config:/etc/subversion/"
        depends_on:
        - postgres
    wildfly: # Application server container
        image: registry.kwsoft.cloud/kwsoft/wildfly26:1
        ports:
        - "8080:8080"
        - "8787:8787"
        - "9990"
        environment:
        - EXTERNAL_HOST_NAME=localhost
        - EXTERNAL_HOST_PORT=8080
        - DB_SCHEMA=mtext
        - DRIVER_CLASS=postgresql
        links:
        - postgres:db
        - svn:svn
        volumes:
        - "./volumes/deployments:/opt/jboss/wildfly/standalone/deployments/"
        - "./volumes/seriem-home:/home/seriem"
        - "./volumes/jboss-
config:/opt/jboss/wildfly/standalone/configuration/"
        - "./volumes/clients:/home/clients/"
        - "./volumes/mtext-init:/home/mtext-init"
        depends_on:
        - postgres
        - seriem
        - wildfly-init
volumes:
    postgres-data: # Database data
    postgres-init: # Initialization SQL scripts for database
    deployments: # Deployments folder of an application server
    seriem-home: # Serie/M home folder
```

```
jboss-config: # JBoss folder configuration
clients: # Contains archive with preconfigurated client environment
svn-data: # SVN data folder
svn-config: # SVN configuration folder
mtext-init: # Folder for custom changes of deployment and client archive
```

7 Appendix 4 – Full example of Kubernetes pod yaml definition

Example configuration of Kubernetes resources. It contains:

- Wildlfy with product, exposed on https://example.eu/
- Postgres database exposed on port 5432 only for internal purposes (wildfly access)
- SVN exposed on port https://example.eu/svn

```
# Docker M/Express - Kubernetes
#
# Volumes are defines as Amazon EBS volumes.
# Welcome page is reachable at https://example.eu/
kind: Service
apiVersion: v1
metadata:
  name: dmeexample
  namespace: seriem
  labels:
    app.kubernetes.io/name: dme
    app.kubernetes.io/instance: example
    app.kubernetes.io/createdby: k8sadmin
spec:
  selector:
    app.kubernetes.io/name: dme
    app.kubernetes.io/instance: example
  ports:
  - name: http
    protocol: TCP
    port: 8080
    targetPort: http
  - name: jboss
    protocol: TCP
    port: 9990
    targetPort: jboss
  - name: svn
    protocol: TCP
    port: 80
    targetPort: svnhttp
  - name: debuging
    protocol: TCP
    port: 8787
    targetPort: debuging
  type: NodePort
---
kind: Ingress
apiVersion: extensions/v1beta1
metadata:
  name: dmeexample
  namespace: seriem
  labels:
```

```
    app.kubernetes.io/name: dme
    app.kubernetes.io/instance: example
    app.kubernetes.io/createdby: k8sadmin
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/force-ssl-redirect: "true"
    nginx.ingress.kubernetes.io/proxy-body-size: "200m"
    nginx.ingress.kubernetes.io/whitelist-source-
range: 0.0.0.0/0 #Any source IP address is allowed
    cert-manager.io/cluster-issuer: letsencrypt
spec:
  tls:
  - hosts:
    - example.eu
    secretName: dmeexample.eu
  rules:
  - host: example.eu
    http:
      paths:
      - path: /svn
        backend:
          serviceName: dmeexample
          servicePort: 80
      - path: /
        backend:
          serviceName: dmeexample
          servicePort: 8080
      - path: /console
        backend:
          serviceName: dmeexample
          servicePort: 9990
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: dmeexample
  namespace: seriem
  metadata:
  annotations:
    MTEXT_VERSION: MTEXTCS_6.12
  creationTimestamp: null
  labels:
    app.kubernetes.io/name: dme
    app.kubernetes.io/instance: example
    app.kubernetes.io/createdby: k8sadmin
spec:
  replicas: 1
  strategy:
```

```
    type: Recreate
  selector:
    matchLabels:
      app.kubernetes.io/name: dme
      app.kubernetes.io/instance: example
  template:
    metadata:
      creationTimestamp: null
      labels:
        app.kubernetes.io/name: dme
        app.kubernetes.io/instance: example
    spec:
      imagePullSecrets:
        - name: registry-omscloud-credentials
      initContainers:
      - name: wildfly-
init # Container is used for initialization of wildfly configuration - exits i
mmediately.
        image: registry.kwsoft.cloud/kwsoft/wildfly26_init:1
        imagePullPolicy: IfNotPresent
        command: ["/bin/sh"]
        args: ["-c", "/home/init.sh"]
        env:
        - name: DRIVER_CLASS
          value: postgresql
        volumeMounts:
        - mountPath: /configuration
          name: ebsvolume
          subPath: dmeexample/jboss-config
      - name: seriem
        image: registry.kwsoft.cloud/kwsoft/seriem613:256 # Container is used
for volumes initialization only - exits immediately.
        imagePullPolicy: IfNotPresent
        command: ["/bin/sh"]
        args: ["-c", "/home/volumes_init.sh"]
        env:
        - name: EXTERNAL_HOST_NAME
          value: example.eu
        - name: EXTERNAL_HOST_PORT
          value: "443"
        - name: DEFAULT_WORKSPACE
          value: "false"
        - name: WS_ONLY
          value: "true"
        volumeMounts:
        - mountPath: /deployments
          name: ebsvolume
          subPath: dmeexample/deployments
```

```
        - mountPath: /clients
          name: ebsvolume
          subPath: dmeexample/clients
        - mountPath: /postgres-init
          name: ebsvolume
          subPath: dmeexample/postgres-init
        - mountPath: /seriem
          name: ebsvolume
          subPath: dmeexample/seriem-home
        - mountPath: /mtext-init
          name: ebsvolume
          subPath: dmeexample/mtext-init
      containers:
      - name: postgres # Database container
        image: registry.kwsoft.cloud/kwsoft/postgres12:3
        imagePullPolicy: IfNotPresent
        env:
        - name: SERVER_NAME
          value: example.eu
        - name: SERVER_DESC
          value: example.eu
        - name: REPOSITORY_URL
          value: svn://localhost/seriem
        readinessProbe:
          tcpSocket:
            port: 5432
          initialDelaySeconds: 30
          periodSeconds: 30
        resources:
          requests:
            memory: "50Mi"
          limits:
            memory: "1000Mi"
        ports:
        - containerPort: 5432
          name: postgres
          protocol: TCP
        volumeMounts:
        - mountPath: /var/lib/postgresql/data
          name: ebsvolume
          subPath: dmeexample/postgres-data
        - mountPath: /docker-entrypoint-initdb.d
          name: ebsvolume
          subPath: dmeexample/postgres-init
      - name: svn # SVN container
        image: registry.kwsoft.cloud/kwsoft/svnserver:1
        imagePullPolicy: IfNotPresent
        readinessProbe:
```

```
          httpGet:
            path: /svn/seriem
            port: 80
            httpHeaders: # Probe for checking health of SVN service. Http head
er contains Basic authentication token composed of name and password of SVN us
er(mtext:mtext)
              - name: Authorization
                value: Basic bXRleHQ6bXRleHQ=
          initialDelaySeconds: 15
          periodSeconds: 30
          timeoutSeconds: 4
        resources:
          requests:
            memory: "50Mi"
          limits:
            memory: "500Mi"
        ports:
        - containerPort: 3690
          name: svn
          protocol: TCP
        - containerPort: 80
          name: svnhttp
          protocol: TCP
        volumeMounts:
        - mountPath: /var/opt/svn/
          name: ebsvolume
          subPath: dmeexample/svn-data
        - mountPath: /etc/subversion/
          name: ebsvolume
          subPath: dmeexample/svn-config
      - name: wildfly # Application server container
        image: registry.kwsoft.cloud/kwsoft/wildfly16:4
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8080
          name: http
          protocol: TCP
        - containerPort: 8443
          name: https
          protocol: TCP
        - containerPort: 9990
          name: jboss
          protocol: TCP
        - containerPort: 8787
          name: debuging
          protocol: TCP
        env:
        - name: EXTERNAL_HOST_NAME
```

```
          value: example.eu
        - name: EXTERNAL_HOST_PORT
          value: "443"
        - name: DB_SCHEMA
          value: "mtext"
        - name: DRIVER_CLASS
          value: postgresql
        - name: WS_HOST_NAME
          value: example.eu
        - name: WS_HTTP_PROTOCOL
          value: "https"
        - name: JBOSS_CONFIG
          value: "standalone-full.xml"
        - name: WEB_ACCESSED_DIR
          value: "/home/seriem/server/"
        - name: CONTENT_HUB_URL
          value: "http://localhost/svn/seriem"
        - name: ANOTHER_JAVA_OPTS
          value: "-XX:MaxMetaspaceSize=512m -Xmx3072m -
XX:NativeMemoryTracking=summary -XX:+PrintFlagsFinal -XX:+PrintFlagsFinal -
XX:MaxRAM=$(($(cat /sys/fs/cgroup/memory/memory.limit_in_bytes)*95/100)) -
agentlib:jdwp=transport=dt_socket,address=8787,server=y,suspend=n -
Djava.net.preferIPv4Stack=true"
        readinessProbe:
          httpGet:
            path: /mtext-integration-adapter/version-info
            port: 8080
          initialDelaySeconds: 20
          periodSeconds: 30
          timeoutSeconds: 4
        resources:
          requests:
            memory: "2048Mi"
          limits:
            memory: "4096Mi"
```

8 Appendix 5 – Helm Values

Default values for the seriem helm chart are listed in following table. Actual version you can download on Helm chart page in Values tab.

https://registry.kwsoft.cloud/harbor/projects/3/helm-charts/seriem/versions

```
general:
  createGitPod: "false"
  createIngress: "true"
  createPostgresPod: "true"
  createSVNPod: "true"
  createSecrets: "true"
  databaseSecretName: postgres-default-credentials
  externalHostPort: 443
  fsGroup: "1000"
  hostDomain: mydomain.com
  onlyDbTools: "false"
  registrySecretName: registry-omscloud-credentials
  storageClass: null
  volumeSize: 10Gi
git:
  limits:
    cpu: 1000m
    memory: 1000Mi
  requests:
    cpu: 100m
    memory: 50Mi
imageCredentials:
  email: null
  password: null
  registry: registry.kwsoft.cloud
  username: null
images:
  git: registry.kwsoft.cloud/kwsoft/gitserver:1
  imagePullPolicy: IfNotPresent
  postgres: registry.kwsoft.cloud/kwsoft/postgres
  seriem: registry.kwsoft.cloud/kwsoft/seriem612:554
  svn: registry.kwsoft.cloud/kwsoft/svnserver:2
  wildfly: registry.kwsoft.cloud/kwsoft/wildfly16:11
```

```
  wildflyInit: registry.kwsoft.cloud/kwsoft/wildfly16_init:7
ingress:
  annotations: |-
    # route.openshift.io/termination: edge
    # nginx.ingress.kubernetes.io/whitelist-source-range: 0.0.0.0/0
    # nginx.ingress.kubernetes.io/force-ssl-redirect: "true"
    # nginx.ingress.kubernetes.io/proxy-body-size: "200m"
  tls:
    enableTls: "false"
    secretNameTls: null
postgres:
  limits:
    cpu: 4000m
    memory: 1000Mi
  requests:
    cpu: 100m
    memory: 50Mi
svn:
  limits:
    cpu: 1000m
    memory: 1000Mi
  requests:
    cpu: 100m
    memory: 50Mi
wildfly:
  WSProtocol: https
  database:
    connectionUrl: ""
    driverClass: postgresql
    password: postgres
    schema: mtext
    userName: postgres
  jbossConfig: standalone-full.xml
  jvm:
    opts: -XX:MaxMetaspaceSize=512m -Xmx3072m -XX:NativeMemoryTracking=summary
-XX:+PrintFlagsFinal
      -XX:+PrintFlagsFinal -XX:MaxRAM=$(($(cat
/sys/fs/cgroup/memory/memory.limit_in_bytes)*95/100))
  limits:
```

```
    cpu: 8000m
    memory: 4096Mi
  repositoryUrl: http://localhost/svn/seriem
  requests:
    cpu: 1000m
    memory: 2048Mi
```