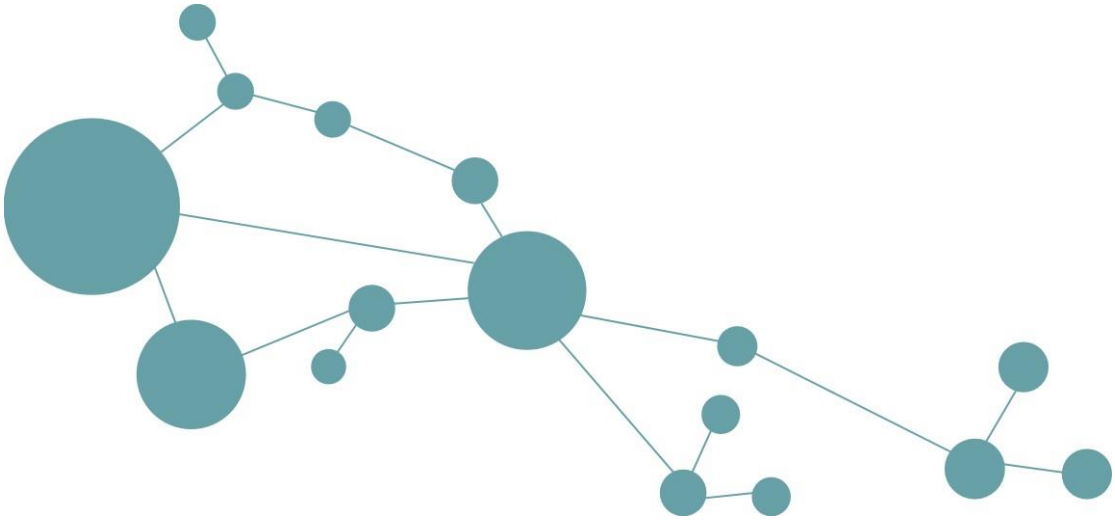

Manuelle Installation von i-views

i·views

Contents

1 General information **3**

2 Operating Systems **5**

2.1 Microsoft Windows 5

2.2 Unix-ish 7

2.2.1 Debian/Ubuntu (x86/amd64) 7

2.2.2 RedHat/Fedora/CentOS (x64/amd64) 9

2.2.3 Mac OS X 10.4+ (amd64) 10

2.2.4 SUSE Linux Enterprise (SLES-12) 11

2.3 Not officially supported operating systems 12

2.3.1 FreeBSD 13

3 Service configuration **13**

3.1 SystemV init 13

3.2 systemd 14

3.3 docker-compose 16

3.4 kubernetes 16

4 Typical Requirements **16**

4.1 Load-Balancing of multiple REST services 16

4.1.1 Nginx 17

4.1.2 Apache HTTPD 17

4.1.3 haproxy 18

4.2 Firewall configuration 18

4.2.1 firewalld 18

5 Appendix **19**

5.1 init.d scripts 19

5.2 docker-compose configuration 33

5.3 kubernetes configuration 34

1 General information

For the operation of i-views components, it is not relevant if processes run in physical hardware (servers, workstation, notebook, ...) or in virtual environments (VMWare, VirtualBox, docker, ...).

The operation of i-views does not require any special access rights or account permissions of the underlying operating system. Processes can run as any operating system account considered reasonable. Nonetheless, running the processes as system accounts is possible; i-views will only work in configured data areas and its product parts only communicate via TCP/IP.

Without further configuration, i-views processes read and write data and log-files in or below the working directory assigned at process startup. That means that the account the process is running as requires write access to the areas the process is running in.

There are also no special requirements for install locations. i-views can be installed in any directory matching your operating guidelines.
Operating any process in a non-locally attached directory (e.g. a network share) is not recommended. Next to reduced I/O performance this can lead to problems in specific situations.

All i-views product parts are run as a combination of virtual machine (VM) and so called image. The image files contain the code of the respective product part and are independent of the operating system, therefore these are interchangeable between operating systems. The distributed virtual machines perform the abstraction of the underlying operating system. For Windows, i-views offers executables which are a pre-packaged combination of VM and image.

Above mentioned virtual machines are available as 32-bit and 64-bit variants. The images "bitness" by default can be recognized by the (missing) "-64" ending in the file name. The "bitness" of image and virtual machine must match.

Currently we recommend - whenever possible - to use the 32-bit version of the software. Operating systems in 64-bit mode sometimes require the installation of additional variants of software packages (see below). The 64-bit images should only be used, if the 2GB address space of 32-bit processes will not be sufficient to perform the tasks of the respective process (usually even large datasets can be handled by 32-bit processes). Mixing of 32-bit and 64-bit processes in a single installation is possible and supported.

All product parts communicate via TCP/IP, so in principle they can be distributed across networked machines. The distribution is not described in this document, please refer to the individual tools documentation for that. If the selected TCP/IP ports are to be used from outside the local machine, the respective operating systems firewall mechanisms might need to be configured to allow such communication.

The configuration of product parts depending on the local installation is performed in ini-files. Some products parts even require such an ini-file and will issue error messages when started without one. When on the processes command line no explicit ini-file is given, the product part searches for a default ini-file in its working directory. The file name of this ini-file depends of the image's role, e.g. a mediator image will search for a mediator.ini file. The filename of the image file is not relevant for its role, so renaming an image file will not make it look for a differently named configuration file.

All product parts can be started via command line or from file manager in foreground. Closing a window or pressing ctrl-c/cmd-c will end the processes. Also, the processes react to standard OS signals. All headless processes can also be started as services, for Windows they support this by built-in command line parameters.

Component / Feature	Description	Required for ...
Core 32 Bit	32-bit-environment	required for Core
Core (de-)compression	Tools for compression	required for Core
Core with graphical interface	i-views-tools with graphical interface	required for Core with UI
Mediator-checksums	Accelerates the calculation of checksums for graph contents	optional for mediator
Image scaling	Accelerated scaling of pixel-based images	optional for Core with UI
Graph-editor alpha-channel	Prettier display of icons with transparent areas in the graph editor	optional for Core with UI
JNI	Java Native Interface to connect to Java based software (e.g. Lucene)	required for the feature
Encryption	Cryptographically protected connections (e.g. HTTPS)	required for the feature
Enhanced text display	Display texts with better anti-aliasing and kerning in UI tools	optional for Core with UI
MySQL-connection	Connect to MySQL databases	required for the feature
ODBC-connection	Connect to ODBC data sources	required for the feature
Oracle-connection	Connect to Oracle databases	required for the feature
SQLite-connection	Connection to SQLite databases	required for the feature and BlobService/Mediator
Extended regular expressions	Extension of the available regular expressions	required for the feature

Keep in mind, that the above mentioned "bitness" of the running i-views virtual machine and image require external libraries of matching "bitness". A library of non-matching bitness will not be activated. This is also the case if one the library's prerequisite libraries is not present with the same "bitness".

2 Operating Systems

i-views unterstützt verschiedene Betriebssysteme. Dieses Kapitel beschreibt die Installation unter voll unterstützten Betriebssystemen. Am Ende finden sich Hinweise zur Installation unter weiteren möglichen Betriebssystemen, die aber nicht offiziell unterstützt werden.

Bei Bedarf können alle in i-views verwalteten Daten auf eine andere Systemumgebung (Hardware, Betriebssystem, ...) ohne Konvertierung migriert werden. Falls die Produktteile über mehrere Rechner verteilt betrieben werden, dürfen auch diese jeweils mit unterschiedlichen Betriebssystemen laufen (z.B. ein Mix von Windows und Linux). Auch für den Zugriff von Clients auf die Server gibt es keine Beschränkungen bei den verwendeten Betriebssystemen.

2.1 Microsoft Windows

i-views runs on all Windows versions with NT Kernel (starting with Windows 2000). Running on brand new versions usually works but becomes officially supported a while after that.

For Windows the product parts usually are delivered as EXE-files. These combine internally a virtual machine and image. If required, instead of EXE-files also separated virtual machines and images can be used.

All product parts, that support the operation as a service, can be installed as a service by issuing the following command line in an administrative shell:

```
PRODUCTPART.exe -installAsService SERVICENAME [SERVICE1 ...] [SERVICEPARAMETER ...]
```

This invocation only creates the service configuration in the Windows environment and stops after that. If a service should depend on the state different services SERVICE1 to SERVICEn, this can be added directly on the command line. Parameters, which should be passed to the process started by the Windows service manager can be added at the end of the command line.

In the Windows service configuration (usually invoked by services.msc) the service then shows up in the list of services with the name SERVICENAME. Here it also can also be modified, e.g. to change the account, the start-type or description of the service. Usual command line tools (like sc.exe) can also be used to modify the service.

To deregister the service, you can use Windows tools like "sc delete SERVICENAME" or:

```
PRODUCTPART.exe -deinstallService SERVICENAME
```

Alternatively, the installation as service can also be performed by Windows service control utilities.

Feature	Library	To install
Core 32 Bit	-	Integrated into Windows
Core (de-)compression	-	Integrated into Windows

Core with graphical interface	-	Integrated into Windows
Mediator-checksums	coastbinary	coastbinary.dll: supplied, next to mediator.exe
Image scaling	GDIPlus	Integrated into Windows
Graph-editor alpha-channel	Cairo	libcairo-2.dll from http://www.gtk.org/download/windows.php
JNI	Java Virtual Machine	libjvm.dll is installed with a Java Runtime (jre) or Java Development Kit (jdk), e.g. from java.com
Encryption	Windows-native OpenSSL	bcrypt.dll: Integrated into Windows, libeay32.dll: via https://wiki.openssl.org/index.php/Binaries (sometimes required because of bcrypt problems)
Enhanced text display	-	Integrated into Windows
MySQL-connection	MySQL client libraries	libmysql.dll: via https://dev.mysql.com/downloads/connector/c/
ODBC-connection	ODBC	Integrated into Windows or installable as Windows feature
Oracle-connection	Windows Advanced Services Oracle client libraries	advapi32.dll: Windows Feature Advanced Services, usually available. oci.dll: from http://www.oracle.com/ in the version matching the target database servers version
SQLite-connection	SQLite	sqlite3.dll from https://www.sqlite.org/download.html
Extended regular expressions	Boost-Regex	boost_regex.dll, msvcp140.dll, vcruntime140.dll from http://www.boost.org/users/download/

When running on Windows, sometimes the libraries of the Microsoft Visual C++ Runtime

need to be installed. This depends on version and type of the OS installation. If they are not present in the current setup, they need to be installed.

The recommended way to do this, is to use the redistributables offered from Microsofts official page of the Visual C++ Redistributable. Please select the correct runtime release, the language matching the operating systems language and the "bitness" matching the used i-views software.
Using this approach not only Microsoft's update mechanisms provide new security fixes and patches for the libraries but also there is only a single copy of the libraries present on your system.

Alternatively you can copy the required msvcpXXX.dll and/or vcruntimeXXX.dll files to the same directory as the executables requiring them.

2.2 Unix-ish

i-views läuft unter verschiedenen Unix-artigen Betriebssystemen, wobei die Mehrzahl der Installationen unter diversen Varianten von Linux laufen.

Für i-views werden (noch) keine Repositories zur Installation und automatischem Upgrade der Software angeboten. Die Software wird aktuell als Archiv geliefert, das an den bevorzugten Ort kopiert werden kann.

Die virtuellen Maschinen benötigen zum Start einige wenige Bibliotheken. Diese dürften auf den meisten Systemen out-of-the-box installiert sein, auf Minimalinstallationen können diese fehlen. Für bestimmte Features verwenden die i-views-Komponenten externe Bibliotheken, die dann aber erst bei Verwendung des Features angesprochen und nachgeladen werden. Falls die notwendigen Bibliotheken nicht gefunden werden, wird dieser Umstand in den Log-Dateien genannt. Hierbei kann es vorkommen, dass in den Log-Dateien die Haupt-Bibliothek als nicht verfügbar genannt wird, diese aber vorhanden ist, jedoch abhängige Bibliotheken fehlen. In diesem Fall hilft meist ein "ldd BIBLIOTHEKSDATEI" um vom System nicht gefundene abhängige Bibliotheken zu identifizieren.

Auf den meisten heute üblichen Servern wird per Default eine 64-Bit-Version eingesetzt. Um i-views als 32-Bit-Prozesse zu starten, ist es dann nötig, die 32-Bit-Laufzeitumgebung parallel zu installieren, wenn diese noch nicht vorhanden ist.

2.2.1 Debian/Ubuntu (x86/amd64)

Debian/Ubuntu verwendet apt als Paketmanager. Es wird empfohlen, wo immer möglich Standard-Paket-Quellen zu verwenden, die aktuelle Patches liefern. Die Pakete werden mit "apt-get install PAKETNAME" installiert.

In der folgenden Tabelle sind die Paketnamen der Standard-Architektur erwähnt. Wird 32-Bit-i-views auf einer 64-Bit-Umgebung betrieben, muss bei der Installation das Suffix ":i386" an den Paketnamen angehängt werden.

Feature	**OS Biblio-thek**	**Paketname / Bibliothek**
Core 32 Bit	GLibC 32 Bit	libc:i386

Core (De-)Kompression	Kompressionsutilities	zlib1g
Core mit grafischer Oberfläche	X11 Umgebung	libx11-6
Mediator Prüfsummen	coastbinary.so	coastbinary.so mitgeliefert, im Verzeichnis der VM
Bildskalierung	FreeImage	libfreeimage3
Graph-Editor Alpha-Kanal	Cairo	libcairo2
JNI	Java Virtual Machine	libjvm.so wird bei der Installation einer Java Runtime (jre) oder eines Development Kits (jdk) mitgeliefert, z.B. unter java.com
Verschlüsselung	OpenSSL	libssl1.0.0
Verbesserte Text Darstellung	XFT	libxft2
MySQL-Anbindung	MySQL oder MariaDB	libmysqlclient20 oder libmariadb-client-lgpl-dev-compat
ODBC-Anbindung	iODBC	libiodbc2
Oracle-Anbindung	Oracle	libclntsh.so Von http://www.oracle.com/ die zur DB passende Version installieren.
SQLite-Anbindung	SQLite	libsqlite3-0
Erweiterte Reguläre Ausdrücke	Boost-Regex	libboost-regex1.58.0

Unter Debian kann man Pakete, die ein bestimmtes Feature enthalten, mit "apt-cache search 'NAME'" finden.

Ob ein Paket wirklich die gewünschten Dateien enthält, kann man mit "apt-file list 'PAKET'" (aus dem Paket apt-file) überprüfen.

Alternativ kann eine fehlende Datei über die Webseiten der jeweiligen Distribution gefunden werden, also z.B. https://www.debian.org/distrib/packages#search_contents bzw. http://packages.ubuntu.com/ gesucht werden.

2.2.2 RedHat/Fedora/CentOS (x64/amd64)

RedHat-basierte Distributionen verwenden meist yum als Paketmanager. Es wird empfohlen, wo immer möglich Standard-Paket-Quellen zu verwenden, die aktuelle Patches liefern. Die Pakete werden mit "yum install PAKETNAME" installiert.

In der folgenden Tabelle sind die Paketnamen der Standard-Architektur erwähnt. Wird 32-Bit-i-views auf einer 64-Bit-Umgebung betrieben, muss bei der Installation das Suffix ".i686" an den Paketnamen angehängt werden.

Feature	**OS Bibliothek**	**Paketname / Bilbiothek**
Core 32 Bit	GLibC 32 Bit	glibc.i686
Core (De-)Kompression	LibZ	zlib
Core mit grafischer Oberfläche	X11 Umgebung	libX11
Mediator Prüfsummen	coastbinary.so	coastbinary.so mitgeliefert, im Verzeichnis der VM
Bildskalierung	FreeImage	freeimage
Graph-Editor Alpha-Kanal	Cairo	cairo
JNI	Java Virtual Machine	libjvm.so wird bei der Installation einer Java Runtime (jre) oder eines Development Kits (jdk) mitgeliefert, z.B. unter java.com
Verschlüsselung	OpenSSL	openssl

Verbesserte Text Darstellung	XFT	libXft
MySQL-Anbindung	MariaDB	mariadb-libs
ODBC-Anbindung	iODBC	libiodbc
Oracle-Anbindung	Oracle	libclntsh.so Von http://www.oracle.com/ die zur DB passende Version installieren.
SQLite-Anbindung	SQLite	sqlite
Erweiterte Reguläre Ausdrücke	Boost-Regex	boost-regex

In RPM-basierten Distributionen kann man Pakete, die ein bestimmtes Feature enthalten-mit "yum search 'NAME' finden. Ob ein Paket wirklich die gewünschten Dateien enthält kann man mit "repoquery -l 'PAKET'" (aus dem Paket yum-utils) überprüfen.

Alternativ kann eine fehlende Datei über https://rpmfind.net/linux/rpm2html/search.php gesucht werden.

2.2.3 Mac OS X 10.4+ (amd64)

Some components are already included in a typical Max OS X, other can be added from the standard code source, via MacPorts or from the respective project websites.

Feature	**OS part**	**Filename**
Core 32 Bit	LibC 32 Bit	libc.dylib, included in Mac OS X as libSystem.dylib
Core (de-)compression	LibZ	Included in Mac OS X
Core with graphical interface	X11 environment	Included in Mac OS X
Mediator-checksums	coastbinary	not available for Mac OS X, checksums are calculated without library support

TLS connections	tlsPlugin	tlsPlugins.dylib: supplied with the client app
Image scaling	FreeImage	Available at the FreeImage Project
Graph-editor alpha-channel	Cairo	libcairo.2.dylib: via Cairo Project or MacPorts
JNI	Java Virtual Machine	
Encryption	OpenSSL	libcrypto.1.0.dylib
Enhanced text display	-	Included in Mac OS X
MySQL-connection	MySQL	libmysqlclient.dylib: from the MySQL website
ODBC-connection	diverse, z.B. iODBC	libiodbc.dylib: e.g. from iodbc.org
Oracle-connection	Oracle	libclntsh.dylib: from Oracles website in the version matching the target database server's version
SQLite-connection	SQLite	libsqlite3.dylib: via sqlite.org
Extended regular expressions	Boost-Regex	libboost_regex.dylib: via boost.org

2.2.4 SUSE Linux Enterprise (SLES-12)

SUSE Distributionen verwenden meist zypper als Paketmanager. Es wird empfohlen, wo immer möglich Standard-Paket-Quellen zu verwenden, die aktuelle Patches liefern. Die Pakete werden mit "zypper install PAKETNAME" installiert.

In der folgenden Tabelle sind die Paketnamen der Standard-Architektur erwähnt. Wird 32-Bit-i-views auf einer 64-Bit-Umgebung betrieben, wird bei der Installation meist das Suffix "-32bit" an den Paketnamen angehängt.

Feature	**OS Bibliothek**	**Paketname / Bilbiothek**
Core 32 Bit	GLibC 32 Bit	glibc-32bit

Core (De-)Kompression	LibZ	libz1
Core mit grafischer Oberfläche	X11 Umgebung	libX11-6
Mediator Prüfsummen	coastbinary.so	coastbinary.so mitgeliefert, im Verzeichnis der VM
Bildskalierung	FreeImage	libfreeimage3
Graph-Editor Alpha-Kanal	Cairo	libcairo2
JNI	Java Virtual Machine	libjvm.so wird bei der Installation einer Java Runtime (jre) oder eines Development Kits (jdk) mitgeliefert, z.B. unter java.com
Verschlüsselung	OpenSSL	libopenssl1_0_0
Verbesserte Text Darstellung	XFT	libXft2
MySQL-Anbindung	MariaDB	libmysqlclient18
ODBC-Anbindung	iODBC	
Oracle-Anbindung	Oracle	libclntsh.so Von http://www.oracle.com/ die zur DB passende Version installieren.
SQLite-Anbindung	SQLite	libsqlite3-0
Erweiterte Reguläre Ausdrücke	Boost-Regex	libboost_regex1_54_0

2.3 Not officially supported operating systems

Für jedes Betriebssystem wird eine spezifische virtuelle Maschine benötigt. Neben den oben beschriebenen Betriebssystemen gibt es VMs für:

- Mac OS X (PowerPC)
- Linux (SPARC), Linux (PPC)
- HPUX (PA RISC)
- AIX
- Solaris (SPARC), Solaris (x86+x86-64)

Diese VMs können auf Anfrage geliefert werden, es liegen aber wenige bis keine Erfahrungen zum Betrieb von i-views auf diesen Plattformen vor.

2.3.1 FreeBSD

FreeBSD bietet einen „Linux-binärkompatiblen Modus“ an. Details zu diesem Modus finden sich in der FreeBSD Dokumentation „10.2. Configuring Linux® Binary Compatibility“ unter https://www.freebsd.org/doc/handbook/linuxemu-lbc-install.html.

Nach Aktivierung des Modus kann man den Anweisungen auf der Webseite zu Linux folgen. Dabei sind zur Installation von möglicherweise noch nicht vorhandenen, aber benötigten Bibliotheken die FreeBSD-spezifischen Befehle zu verwenden.

3 Service configuration

Running of i-views product parts depends on the distributions "init"-system. Here we currently offer sample files for two styles of init- systems:

- systemd-based like CentOS (from V7 on) und Ubuntu (from 16.04 on)
- System-V-init.d scripts like SUSE-Enterprise-Linux-11

Other init-systems like e.g. upstart (CentOS 6.5) or LaunchDaemon (Mac OS X) are currently not offered here, as we do only have incomplete or not-generic files for those. Since i-views does not depend on any special operating system support, running i-views on those platforms should be easily possible.

For container based environments the build system creates standard container images, which can be easily used. Example configurations provided are:

- docker-compose
- kubernetes

Other container runtime variants like docker, podman or openshift can be derived from these templates.

3.1 SystemV init

The presented init.d scripts

- assume a bash-shell executing the scipt
- use /opt/iviews/ as a base path for all installed i-views components
- try to control all processes with an account named "iviews".

The scripts themselves can execute without any external resources, which also implies that environmental parameters need to be adapted in the scripts for all product parts. The script files contain comments, which should explain each parameters meaning and default value.

Please refer to the appendix for the script files.

3.2 systemd

For regular operation in distributions using systemd as init-system, the following configuration files can be used as a blueprint:

/etc/systemd/system/iviews-mediator.service:

```
[Unit]
Description=iviews mediator
After=network.target nss-lookup.target

[Service]
User=iviews
WorkingDirectory=/opt/iviews/mediator
ExecStart=/opt/iviews/vm/vwlinux86 -noherald -=/opt/iviews/mediator/mediator.im
Restart=on-failure

[Install]
WantedBy=default.target
```

/etc/systemd/system/iviews-bridge-rest.service:

```
[Unit]
Description=iviews bridge rest
After=network.target nss-lookup.target iviews-mediator.service
Wants=iviews-mediator.service

[Service]
User=iviews
WorkingDirectory=/opt/iviews/bridge
ExecStart=/opt/iviews/vm/vwlinux86 -noherald -=/opt/iviews/bridge/bridge.im -ini bridge-rest.ini
Restart=on-failure

[Install]
WantedBy=default.target
```

/etc/systemd/system/iviews-jobclient.service:

```
[Unit]
Description=iviews jobclient
After=network.target nss-lookup.target iviews-mediator.service
Wants=iviews-mediator.service

[Service]
User=iviews
```

```
WorkingDirectory=/opt/iviews/jobclient
ExecStart=/opt/iviews/vm/vwlinux86 -noherald -=/opt/iviews/jobclient/jobclient.im
Restart=on-failure

[Install]
WantedBy=default.target
```

If services are to be combined to a single installation, e.g. if you want to run multiple projects in parallel on the same machine and want to be able to control whole projects with single command lines, it might be useful to create a project-specific systemd-target:

/etc/systemd/system/iviews.target:

```
[Unit]
Description=iviews
After=network.target nss-lookup.target
Wants=iviews-mediator.service iviews-bridge-rest.service iviews-jobclient.service

[Install]
WantedBy=default.target
```

In this case you can remove the [Install] sections from the individual service files and add an entry to each [Unit] section:

```
PartOf=iviews.target
```

When starting or stopping a iviews.target unit, all parts will be started or stopped. You also can control individual services without affecting the iviews.target.

Names of the services and paths need to be adapted to your environment. The above service definitions can be used as a blueprint for services with further i-views product parts. After modification of these files, systemd needs to be informed about the changed services via systemctl.

If required, dependencies of services can be expressed differently. Please see the systemd homepage or your distributions man-pages.

Short cheat-sheet for controlling systemd services:

Tasks	**Command**
Register a service	systemctl enable SERVICENAME
Deregister a service	systemctl disable SERVICENAME
Start a service	systemctl start SERVICENAME
Stop a service	systemctl stop SERVICENAME
Status of a service	systemctl status SERVICENAME
Edit a service file	systemctl edit –full SERVICENAME

After manual changes to a service file	systemctl daemon-reload

3.3 docker-compose

The presented example provides a configuration file for docker-compose starting a mediator bridge and jobclient for a project. There is also an example env-file containing the secrets required in the installation. The compose-file can safely be stored in a versioning system like git. The env-file in the repository should not contain actual secrets.

In the files values enclosed in angular brackets (like "<knowledge-graph-name>") are placeholders to be filled with your projects values. Identical placeholders indicate the same value to be inserted. These are intentionally not part of the env-file in this configuration variant to enable only having a single env-file and being able to keep all relevant configuration information in a repository.

Please note that the installations does not use any INI-files, but relies on IV_ prefixed variables. The i-views tools assist in converting from ini to environment variables by using the commandline parameter -iniToEnv.

Please refer to the appendix for the configuration files.

3.4 kubernetes

Running i-views in kubernetes environments is possible. For the mediator a reliable storage driver is required, since it only supports filesystem storage.

In the files values in angular brackets (like "<knowledge-graph-name>") are placeholders to be filled with your projects values. Identical placeholder names in the files indicate the same value to be inserted. The single-file configuration below can be separated into multiple yaml files if you prefer to manage the setup in one file per resource.

Please note:

- Nearly all values mentioned here are depending on the project, the kubernetes environment and policies relevant to the environment.
- Providing storage via NFS shares is strongly discouraged due to its history of failures.
- Also avoid transfering process with permanent storage (especially the mediator) to different nodes (see PodDisruptionBudget, maxUnavailable: 0).

Please refer to the appendix for the configuration file.

4 Typical Requirements

4.1 Load-Balancing of multiple REST services

Bei REST-Services bietet es sich an, durch mehrere Diensterbringer einen höheren Durchsatz an parallelen Requests zu erreichen. i-views selbst ermöglicht mit einer Bridge mit

LoadBalancer-Konfiguration das Starten und Überwachen mehrerer REST-Bridges. In der Konfigurationsdatei der Bridge werden dann z.B. folgende Einstellungen vorgenommen:

```
[KLoadBalancer]
hostname=localhost
port=5001
vm=/opt/kinfinity/vm/vwlinux86
directory=.
image=bridge.im
configNames=REST
autoRestart=true

[REST]
bridgeClientClassName=KWeb.KHTTPRestBridge
inifile=bridge_rest.ini
bridgeLogfile=bridge_rest_<id>.log
ports=5002-5005
```

Diese Konfiguration startet vier REST-Bridges auf den Ports 5001 bis 5005.

Allerdings unterstützt diese LoadBalancer-Bridge nicht selbst das LoadBalancing. Hierfür werden üblicherweise externe Pakete wie Nginx oder Apache-HTTPD eingesetzt.

4.1.1 Nginx

Nginx kann mit der proxy_pass Direktive Anfragen an einen bestimmten Pfad intern an verschiedene Backends weiterleiten. Dazu erweitert man die Konfiguration von nginx um folgende Zeilen:

```
upstream kinf-rest-bridges {
   server 127.0.0.1:5002;
   server 127.0.0.1:5003;
   server 127.0.0.1:5004;
   server 127.0.0.1:5005;
}

server {
   ...

   location ^~ /myRestPath {
      proxy_pass http://kinf-rest-bridges/myRestPath;
   }
}
```

4.1.2 Apache HTTPD

Apache HTTPD enthält typischweise das Modul Proxy, welches u.a. Load Balancing anbietet. Eine Konfiguration für obige Bridge könnte folgendermaßen aussehen:

```
<Proxy balancer://kinfinity>
   BalancerMember http://127.0.0.1:5002/ disablereuse=On
   BalancerMember http://127.0.0.1:5003/ disablereuse=On
   BalancerMember http://127.0.0.1:5004/ disablereuse=On
   BalancerMember http://127.0.0.1:5005/ disablereuse=On
</Proxy>
ProxyPass  /myRestPath  balancer://kinfinity  lbmethod=bybusyness
```

4.1.3 haproxy

haproxy ist im Gegensatz zu nginx oder apache nur ein reiner Proxy-Dienst. haproxy wird typischerweise konfiguriert in

/etc/haproxy/haproxy.cfg:

```
frontend         main  *:5000
default_backend  rest_bridge

backend rest_bridge
  balance  first
  server   rest_bridge1 127.0.0.1:5002 check maxconn 1 weight 100
  server   rest_bridge2 127.0.0.1:5003 check maxconn 1 weight 99
  server   rest_bridge3 127.0.0.1:5004 check maxconn 1 weight 98
  server   rest_bridge4 127.0.0.1:5005 check maxconn 1 weight 97
```

4.2 Firewall configuration

i-views-Prozesse kommunizieren untereinander ausschließlich über TCP/IP. Die konkret verwendeten Ports hängen von der eingesetzten Software-Version und natürlich der lokalen Konfiguration ab.

Je nach eingesetztem OS und Distribution findet die Konfiguration unterschiedlich statt, daher kann hier keine komplette Dokumentation aller Firewall-Utilities vorgestellt werden. Z.B. unter Linux gibt es diverse Frontends und Konfigurationsweisen, welche aber alle die unterliegende iptables- bzw. nftables-Implementierung im Kernel steuern.

4.2.1 firewalld

firewalld, wie es z.B. unter CentOS7 eingesetzt wird, verwendet zur Konfiguration Zonen und Dienste.

Firewalld verwendet XML-Konfigurationsdateien, in denen man z.B. für eine Installation die komplette Konfiguration der Firewall-Regeln in einer Datei eintragen kann. Die Dateien werden in /etc/firewalld/services/NAME.xml abgelegt.

/etc/firewalld/services/kinfinity.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>iviews-example</short>
  <description>i-views example installation</description>
```

```
  <!- mediator ->
  <port protocol="tcp" port="30061"/>
  <!- rest-bridge ->
  <port protocol="tcp" port="8815"/>
</service>
```

Eine neue Konfiguration kann man dann dauerhaft (–permanent) mit folgenden Befehlen für die öffentliche Zone (public) aktivieren:

```
firewall-cmd -permanent -zone=public -add-service=kinfinity
firewall-cmd -reload
```

Alternativ kann man auch die entsprechende Zonendatei manuell erweitern.

/etc/firewalld/zones/public.xml:

```
  <service name="kinfinity"/>
```

5 Appendix

5.1 init.d scripts

/etc/init.d/mediator:

```
#!/bin/bash
#
# Mediator Control-Script
#
#   chkconfig: 2345 20  80
#   description: Startup/shutdown script for Mediator

#=========================================================================
#== change the following parameters up to the second double line
#== to fit to the local installation

#-----------------------------------
#- environment specific parameters

# path to external programs used in this script

# sudo, to support running as different user
SUDO=/usr/bin/sudo

# id to test current user (supporting options -nu)
ID=/usr/bin/id

# rm to delete pid files
RM=/bin/rm
```

```
# sh for execution of sudo commands in a subshell
SH=/bin/bash

#-----------------------------------
#- mediator specific parameters

# directory, where the mediator will run in
# volumes will be expected in the 'volumes' subdirectory, some logfiles are dropped there too
#   (format: path, default: ".")
DIRECTORY="/opt/iviews/mediator"

# what image should be used for the mediator
#   (format: path+file, default: "$DIRECTORY/mediator.im")
IMAGE=""

# OPTIONAL: extra mediator parameters
#   (format: "(-OPTION)*", default: none)
PARAMS=""

# OPTIONAL: stop method: KILL|SHUTDOWN
# see documentation for details on SHUTDOWN
#   (format: "KILL|SHUTDOWN", default: "SHUTDOWN")
STOPMETHOD=""

# OPTIONAL: user to run as
#   (format: username, default: none)
RUNASUSER="iviews"

#-----------------------------------
#- mediator parameters, possibly defined in the ini-file

# port, where the mediator should run on
#   (format: number, default: none)
PORT=""

# OPTIONAL: mediator logfile location
#   (format: path+file, default: "$DIRECTORY/mediator.log")
LOGFILE=""

#-----------------------------------
#- script specific parameters

# OPTIONAL: pidfile location (unused for stop, only used by kill)
#   (format: path+file, default: "$DIRECTORY/mediator.pid")
PIDFILE=""

#-----------------------------------
#- visualworks specific parameters

# where is the vm for that image located
#   (format: path+file, default: "$DIRECTORY/vwlinux86")
VM=""
# note: on some platforms it might be necessary to prefix the vm
```

```
#       command by "setarch i386 -L " or similar, when normal starts
#       directly lead to a segmentation fault

# vm parameters ("-noherald" is usually a good idea for services)
#   (format: "(-OPTION)*", default: none)
VM_PARAMS="-noherald"

#=========================================================================
#== no need to change anything below

PRODUCT="mediator"

#-----------------------------------
#- parameter checking and defaulting

if test -z "$DIRECTORY"; then DIRECTORY='.'; fi
if test ! -d "$DIRECTORY" -o ! -w "$DIRECTORY"; then
   echo "directory '$DIRECTORY' not accessible, abort"; exit 1
fi

if test -z "$VM"; then VM="$DIRECTORY/vwlinux86"; fi
if test ! -f "$VM" -o ! -x "$VM"; then
   echo "virtual machine '$VM' not accessible, abort"; exit 1
fi

if test -z "$IMAGE"; then IMAGE="$DIRECTORY/$PRODUCT.im"; fi
if test ! -f "$IMAGE" -o ! -r "$IMAGE"; then
   echo "image '$IMAGE' not accessible, abort"; exit 1
fi

if test -n "$PORT"; then PARAMS="$PARAMS -port $PORT"; fi

if test -n "$LOGFILE"; then PARAMS="$PARAMS -log $LOGFILE"; fi

if test -z "$PIDFILE"; then PIDFILE="$DIRECTORY/$PRODUCT.pid"; fi

#-----------------------------------
#- actions

startproc () {
  if test $# -lt 0; then return; fi
  local CMD=$1
  local BGFG=$2
  local PIDFILE=$3
  if test -z "$CMD"; then return; fi

  CMD="$CMD <&- >&- 2>&-"
  if test "$BGFG" = bg; then
     CMD="$CMD & disown"
     if test -n "$PIDFILE"; then
        CMD="$CMD; echo \$! > '$PIDFILE'"
     fi
  fi
```

```
  if test -z "$RUNASUSER" -o "$RUNASUSER" = $($ID -nu); then
     eval "$CMD"
  else
     $SUDO -u "$RUNASUSER" $SH -c "$CMD"
  fi
}

case "$1" in
  start)
    echo -n "Starting k-infinity $PRODUCT in $DIRECTORY: "
    cwd=$(pwd); cd $DIRECTORY
    startproc "'$VM' $VM_PARAMS '$IMAGE' $PARAMS" bg "$PIDFILE"
    cd "$cwd"
    echo "done"
    ;;
  stop)
    echo -n "Stopping k-infinity $PRODUCT in $DIRECTORY: "
    cwd=$(pwd); cd $DIRECTORY
    startproc "'$VM' $VM_PARAMS '$IMAGE' $PARAMS -stop" fg
    startproc "$RM -f '$PIDFILE'" fg
    cd "$cwd"
    echo "done"
    ;;
  kill)
    if test -n "$PIDFILE"; then
       echo -n "Killing k-infinity $PRODUCT in $DIRECTORY: "
       if test ! -f "$PIDFILE" -o ! -r "$PIDFILE";
          then echo "missing pidfile '$PIDFILE', abort"; exit 1
          else
            startproc "kill -KILL \$(< '$PIDFILE')" fg
            startproc "$RM -f '$PIDFILE'" fg
       fi
       echo 'done'
       echo 'Do not forget to clean up all volume directories'
    else echo 'No PIDFILE parameter specified, NOT killed!'
    fi
    ;;
  restart)
    $0 stop; $0 start;
    ;;
  *)
    echo "Usage: $0 {start|stop|kill|restart}"
    exit 1
    ;;
esac

#=====================================================================
#== the end

exit 0
```

/etc/init.d/bridge:

```
#!/bin/bash
#
# Bridge Control-Script
#
#   chkconfig: 2345 30  70
#   description: Startup/shutdown script for Bridge

#=====================================================================
#== change the following parameters up to the second double line
#== to fit to the local installation

#-----------------------------------
#- environment specific parameters

# path to external programs used in this script

# sudo, to support running as different user
SUDO=/usr/bin/sudo

# id to test current user (supporting options -nu)
ID=/usr/bin/id

# rm to delete pid files
RM=/bin/rm

# sh for execution of sudo commands in a subshell
SH=/bin/bash

#-----------------------------------
#- bridge specific parameters

# directory, where the bridge will run in
#   (format: path, default: ".")
DIRECTORY="/opt/iviews/bridge"

# what image should be used for the bridge in
#   (format: path+file, default: "$DIRECTORY/bridge.im")
IMAGE=""

# OPTIONAL: ini-file location
#   (format: path+file, default: none)
INIFILE=""

# OPTIONAL: extra parameters for the bridge
#   (format: "(-OPTION)*", default: none)
PARAMS=""

# OPTIONAL: stop method: KILL|SHUTDOWN
# see documentation for details on SHUTDOWN
#   (format: "KILL|SHUTDOWN", default: "SHUTDOWN")
STOPMETHOD=""
```

```
# OPTIONAL: user to run as
#   (format: username, default: none)
RUNASUSER="iviews"

#-----------------------------------
#- bridge parameters, usually better defined in the ini-file

# OPTIONAL: the mediator location, where to connect to
#   (format: "HOST[:PORT]", default: none)
MEDIATOR=""

# OPTIONAL: local port to start on
#   (format: "PORT", default: none)
LOCALPORT=""

#-----------------------------------
#- script specific parameters

# pidfile location. used for stop with KILL method
#   (format: path+file, default: "$DIRECTORY/bridge.pid")
PIDFILE=""

# The maximum time in seconds that a regular service shutdown may take
# (time from the execution of the shutdown command to the termination of the
# process). If the process is still running after the given time, it will
# be terminated by a kill command. Setting the timeout to a zero or negative
# value will disable that mechanism. The timeout parameter will only be
# evaluated if STOPMETHOD is set to "SHUTDOWN" and if PIDFILE is set to
# a valid path.
TIMEOUT=30

#-----------------------------------
#- visualworks specific parameters

# where is the vm for that image located
#   (format: path+file, default: "$DIRECTORY/vwlinux86")
VM=""
# note: on some platforms it might be necessary to prefix the vm
#       command by "setarch i386 -L " or similar, when normal starts
#       directly lead to a segmentation fault

# vm parameters ("-noherald" is usually a good idea for services)
#   (format: "(-OPTION)*", default: none)
VM_PARAMS="-noherald"

#===============================================================================
#== no need to change anything below

PRODUCT="bridge"

#-----------------------------------
#- parameter checking and defaulting
```

```
if test -z "$DIRECTORY"; then DIRECTORY='.'; fi
if test ! -d "$DIRECTORY" -o ! -w "$DIRECTORY"; then
   echo "directory '$DIRECTORY' not accessible, abort"; exit 1
fi

if test -z "$VM"; then VM="$DIRECTORY/vwlinux86"; fi
if test ! -f "$VM" -o ! -x "$VM"; then
   echo "virtual machine '$VM' not accessible, abort"; exit 1
fi

if test -z "$IMAGE"; then IMAGE="$DIRECTORY/$PRODUCT.im"; fi
if test ! -f "$IMAGE" -o ! -r "$IMAGE"; then
   echo "image '$IMAGE' not accessible, abort"; exit 1
fi

if test -z "$PIDFILE"; then PIDFILE="$DIRECTORY/$PRODUCT.pid"; fi

if test -n "$INIFILE"; then PARAMS="$PARAMS -ini $INIFILE"; fi

if test -n "$MEDIATOR"; then PARAMS="$PARAMS -host $MEDIATOR"; fi

if test -n "$LOCALPORT"; then PARAMS="$PARAMS -port $LOCALPORT"; fi

#----------------------------------
#- actions

startproc () {
  if test $# -lt 0; then return; fi
  local CMD=$1
  local BGFG=$2
  local PIDFILE=$3
  if test -z "$CMD"; then return; fi

  CMD="$CMD <&- >&- 2>&-"
  if test "$BGFG" = bg; then
     CMD="$CMD & disown"
     if test -n "$PIDFILE"; then
        CMD="$CMD; echo \$! > '$PIDFILE'"
     fi
  fi

  if test -z "$RUNASUSER" -o "$RUNASUSER" = $($ID -nu); then
     eval "$CMD"
  else
     $SUDO -u "$RUNASUSER" $SH -c "$CMD"
  fi
}

case "$1" in
  start)
    echo -n "Starting k-infinity $PRODUCT in $DIRECTORY: "
    cwd=$(pwd); cd $DIRECTORY
    startproc "'$VM' $VM_PARAMS '$IMAGE' $PARAMS" bg "$PIDFILE"
```

```
  cd "$cwd"
  echo "done"
  ;;
stop)
  echo -n "Stopping k-infinity $PRODUCT in $DIRECTORY: "
  if test "$STOPMETHOD" = KILL; then
     if test ! -f "$PIDFILE" -o ! -r "$PIDFILE";
        then echo "missing pidfile '$PIDFILE', abort"; exit 1
        else
          startproc "kill -KILL \$(< '$PIDFILE')" fg
          startproc "$RM -f '$PIDFILE'" fg
     fi
  else
    cwd=$(pwd); cd $DIRECTORY
    startproc "'$VM' $VM_PARAMS '$IMAGE' $PARAMS -stop localhost" fg
    if test -f "$PIDFILE" ; then
       if test "$TIMEOUT" -gt 0 ; then
          PID=`cat $PIDFILE`
          for (( i=0 ; i < $TIMEOUT ; i++ )) do
              if ps -p $PID /dev/null 2>&1 ; then
                 sleep 1s
                 echo -n "."
              else
                 startproc "$RM -f '$PIDFILE'" fg
                 i=$TIMEOUT;
              fi
          done
          if [ -f "$PIDFILE" ]; then
             echo -n " regular shutdown failed - killing process! "
             startproc "kill -KILL \$(< '$PIDFILE')" fg
             startproc "$RM -f '$PIDFILE'" fg
          fi
       else
          startproc "$RM -f '$PIDFILE'" fg
       fi
    fi
    cd "$cwd"
  fi
  echo "done"
  ;;
kill)
  echo -n "Killing k-infinity $PRODUCT in $DIRECTORY: "
  if test ! -f "$PIDFILE" -o ! -r "$PIDFILE";
     then echo "missing pidfile '$PIDFILE', abort"; exit 1
     else
       startproc "kill -KILL \$(< '$PIDFILE')" fg
       startproc "$RM -f '$PIDFILE'" fg
  fi
  echo "done"
  ;;
restart)
  $0 stop; $0 start;
  ;;
```

```
  *)
    echo "Usage: $0 {start|stop|kill|restart}"
    exit 1
    ;;
esac

#=====================================================================
#== the end

exit 0
```

/etc/init.d/blobservice:

```
#!/bin/bash
#
# Blobservice Control-Script
#
#   chkconfig: 2345 30  70
#   description: Startup/shutdown script for Blobservice

#=====================================================================
#== change the following parameters up to the second double line
#== to fit to the local installation

#-----------------------------------
#- environment specific parameters

# path to external programs used in this script

# sudo, to support running as different user
SUDO=/usr/bin/sudo

# id to test current user (supporting options -nu)
ID=/usr/bin/id

# rm to delete pid files
RM=/bin/rm

# sh for execution of sudo commands in a subshell
SH=/bin/bash

#-----------------------------------
#- blobservice specific parameters

# directory, where the blobservice will run in
#   (format: path, default: ".")
DIRECTORY="/opt/iviews/blobservice"

# what image should be used for the blobservice in
#   (format: path+file, default: "$DIRECTORY/blobservice.im")
IMAGE=""

# OPTIONAL: ini-file location
```

```
#   (format: path+file, default: none)
INIFILE=""

# OPTIONAL: extra parameters for the blobservice
#   (format: "(-OPTION)*", default: none)
PARAMS=""

# OPTIONAL: user to run as
# OPTIONAL: user to run as
#   (format: username, default: none)
RUNASUSER="iviews"

#----------------------------------
#- blobservice parameters, usually better defined in the ini-file

# OPTIONAL: the bridge location, where to connect to
#   (format: "HOST[:PORT]", default: none)
MEDIATOR=""

# OPTIONAL: local port to start on
#   (format: "PORT", default: none)
LOCALPORT=""

#----------------------------------
#- script specific parameters

# pidfile location. used for stop with KILL method
#   (format: path+file, default: "$DIRECTORY/blobservice.pid")
PIDFILE=""

#----------------------------------
#- visualworks specific parameters

# where is the vm for that image located
#   (format: path+file, default: "$DIRECTORY/vwlinux86")
VM=""
# note: on some platforms it might be necessary to prefix the vm
#       command by "setarch i386 -L " or similar, when normal starts
#       directly lead to a segmentation fault

# vm parameters ("-noherald" is usually a good idea for services)
#   (format: "(-OPTION)*", default: none)
VM_PARAMS="-noherald"

#==============================================================================
#== no need to change anything below

PRODUCT="blobservice"

#----------------------------------
#- parameter checking and defaulting

if test -z "$DIRECTORY"; then DIRECTORY='.'; fi
```

```
if test ! -d "$DIRECTORY" -o ! -w "$DIRECTORY"; then
   echo "directory '$DIRECTORY' not accessible, abort"; exit 1
fi

if test -z "$VM"; then VM="$DIRECTORY/vwlinux86"; fi
if test ! -f "$VM" -o ! -x "$VM"; then
   echo "virtual machine '$VM' not accessible, abort"; exit 1
fi

if test -z "$IMAGE"; then IMAGE="$DIRECTORY/$PRODUCT.im"; fi
if test ! -f "$IMAGE" -o ! -r "$IMAGE"; then
   echo "image '$IMAGE' not accessible, abort"; exit 1
fi

if test -z "$PIDFILE"; then PIDFILE="$DIRECTORY/$PRODUCT.pid"; fi

if test -n "$INIFILE"; then PARAMS="$PARAMS -ini $INIFILE"; fi

if test -n "$MEDIATOR"; then PARAMS="$PARAMS -host $MEDIATOR"; fi

if test -n "$LOCALPORT"; then PARAMS="$PARAMS -port $LOCALPORT"; fi

#-----------------------------------
#- actions

startproc () {
  if test $# -lt 0; then return; fi
  local CMD=$1
  local BGFG=$2
  local PIDFILE=$3
  if test -z "$CMD"; then return; fi

  CMD="$CMD <&- >&- 2>&-"
  if test "$BGFG" = bg; then
     CMD="$CMD & disown"
     if test -n "$PIDFILE"; then
        CMD="$CMD; echo \$! > '$PIDFILE'"
     fi
  fi

  if test -z "$RUNASUSER" -o "$RUNASUSER" = $($ID -nu); then
     eval "$CMD"
  else
     $SUDO -u "$RUNASUSER" $SH -c "$CMD"
  fi
}

case "$1" in
  start)
    echo -n "Starting k-infinity $PRODUCT in $DIRECTORY: "
    cwd=$(pwd); cd $DIRECTORY
    startproc "'$VM' $VM_PARAMS '$IMAGE' $PARAMS" bg "$PIDFILE"
    cd "$cwd"
```

```
    echo "done"
    ;;
  stop)
    echo -n "Stopping k-infinity $PRODUCT in $DIRECTORY: "
    if test ! -f "$PIDFILE" -o ! -r "$PIDFILE";
       then echo "missing pidfile '$PIDFILE', abort"; exit 1
       else
         startproc "kill -KILL \$(< '$PIDFILE')" fg
         startproc "$RM -f '$PIDFILE'" fg
    fi
    echo "done"
    ;;
  restart)
    $0 stop; $0 start;
    ;;
  *)
    echo "Usage: $0 {start|stop|restart}"
    exit 1
    ;;
esac

#=====================================================================
#== the end

exit 0
```

/etc/init.d/jobclient:

```
#!/bin/bash
#
# JobClient Control-Script
#
#   chkconfig: 2345 30  70
#   description: Startup/shutdown script for JobClient

#=====================================================================
#== change the following parameters up to the second double line
#== to fit to the local installation

#-----------------------------------
#- environment specific parameters

# path to external programs used in this script

# sudo, to support running as different user
SUDO=/usr/bin/sudo

# id to test current user (supporting options -nu)
ID=/usr/bin/id

# rm to delete pid files
RM=/bin/rm
```

```
# sh for execution of sudo commands in a subshell
SH=/bin/bash

#-----------------------------------
#- jobclient specific parameters

# directory, where the jobclient will run in
#   (format: path, default: ".")
DIRECTORY="/opt/iviews/jobclient"

# what image should be used for the jobclient in
#   (format: path+file, default: "$DIRECTORY/jobclient.im")
IMAGE=""

# MANDATORY: ini-file location
#   (format: path+file, default: "$DIRECTORY/jobclient.ini")
INIFILE=""

# OPTIONAL: extra parameters for the jobclient
#   (format: "(-OPTION)*", default: none)
PARAMS=""

# OPTIONAL: user to run as
#   (format: username, default: none)
RUNASUSER="iviews"

#-----------------------------------
#- script specific parameters

# pidfile location. used for stop
#   (format: path+file, default: "$DIRECTORY/jobclient.pid")
PIDFILE=""

#-----------------------------------
#- visualworks specific parameters

# where is the vm for that image located
#   (format: path+file, default: "$DIRECTORY/vwlinux86")
VM=""
# note: on some platforms it might be necessary to prefix the vm
#       command by "setarch i386 -L " or similar, when normal starts
#       directly lead to a segmentation fault

# vm parameters ("-noherald" is usually a good idea for services)
#   (format: "(-OPTION)*", default: none)
VM_PARAMS="-noherald"

#============================================================================
#== no need to change anything below

PRODUCT="jobclient"

#-----------------------------------
```

```
#- parameter checking and defaulting

if test -z "$DIRECTORY"; then DIRECTORY='.'; fi
if test ! -d "$DIRECTORY" -o ! -w "$DIRECTORY"; then
   echo "directory '$DIRECTORY' not accessible, abort"; exit 1
fi

if test -z "$VM"; then VM="$DIRECTORY/vwlinux86"; fi
if test ! -f "$VM" -o ! -x "$VM"; then
   echo "virtual machine '$VM' not accessible, abort"; exit 1
fi

if test -z "$IMAGE"; then IMAGE="$DIRECTORY/$PRODUCT.im"; fi
if test ! -f "$IMAGE" -o ! -r "$IMAGE"; then
   echo "image '$IMAGE' not accessible, abort"; exit 1
fi

if test -z "$PIDFILE"; then PIDFILE="$DIRECTORY/$PRODUCT.pid"; fi

if test -n "$INIFILE"; then PARAMS="$PARAMS -ini $INIFILE"; fi

#-----------------------------------
#- actions

startproc () {
  if test $# -lt 0; then return; fi
  local CMD=$1
  local BGFG=$2
  local PIDFILE=$3
  if test -z "$CMD"; then return; fi

  CMD="$CMD <&- >&- 2>&-"
  if test "$BGFG" = bg; then
     CMD="$CMD & disown"
     if test -n "$PIDFILE"; then
        CMD="$CMD; echo \$! > '$PIDFILE'"
     fi
  fi

  if test -z "$RUNASUSER" -o "$RUNASUSER" = $($ID -nu); then
     eval "$CMD"
  else
     $SUDO -u "$RUNASUSER" $SH -c "$CMD"
  fi
}

case "$1" in
  start)
    echo -n "Starting k-infinity $PRODUCT in $DIRECTORY: "
    cwd=$(pwd); cd $DIRECTORY
    startproc "'$VM' $VM_PARAMS '$IMAGE' $PARAMS" bg "$PIDFILE"
    cd "$cwd"
    echo "done"
```

```
    ;;
  stop)
    echo -n "Stopping k-infinity $PRODUCT in $DIRECTORY: "
    if test ! -f "$PIDFILE" -o ! -r "$PIDFILE";
       then echo "missing pidfile '$PIDFILE', abort"; exit 1
       else
         startproc "kill -KILL \$(< '$PIDFILE')" fg
         startproc "$RM -f '$PIDFILE'" fg
    fi
    echo "done"
    ;;
  restart)
    $0 stop; $0 start;
    ;;
  *)
    echo "Usage: $0 {start|stop|restart}"
    exit 1
    ;;
esac

#=================================================================
#== the end

exit 0
```

5.2 docker-compose configuration

.env

```
MEDIATOR_PASSWORD=<mediator-password>
AUTH_TOKEN=<system-account>
```

docker-compose.yaml

```
version: '3'
services:
  mediator:
    image: container-registry.example.com/iviews-mediator:<image-tag-1>
    environment:
      IV_PASSWORD: "${MEDIATOR_PASSWORD:?MEDIATOR_PASSWORD not configured}"
      IV_SCHEDULED_JOBS: "job-bu,job-gc"
      IV_JOB_BU_VOLUME_PATTERN: "<knowledge-graph-name>"
      IV_JOB_BU_BACKUP_INTERVAL: "1"
      IV_JOB_BU_BACKUP_TIME: "22:22"
      IV_JOB_BU_BACKUPS_TO_KEEP: "5"
      IV_JOB_GC_VOLUME_PATTERN: "<knowledge-graph-name>"
      IV_JOB_GC_GARBAGE_COLLECT_INTERVAL: "1"
      IV_JOB_GC_GARBAGE_COLLECT_TIME: "22:33"
    volumes:
```

```
      - "mediator-volumes:/mediator/volumes"
      - "mediator-backup:/mediator/backup"
    ports:
      - "30000:30000"
      - "30001:30001"
    restart: unless-stopped
  jobclient:
    image: container-registry.example.com/iviews-jobclient:<image-tag-2>
    environment:
      IV_HOST: "mediator:30001"
      IV_VOLUME: "<knowledge-graph-name"
      IV_AUTHENTICATION: "${AUTH_TOKEN:?AUTH_TOKEN not configured}"
      IV_JOB_POOLS: "script,index"
    depends_on:
      - mediator
    deploy:
      replicas: 1
    restart: unless-stopped
  bridge:
    image: container-registry.example.com/iviews-bridge:<image-tag-3>
    environment:
      IV_HOST: "mediator:30001"
      IV_VOLUME: "<knowledge-graph-name>"
      IV_AUTHENTICATION: "${AUTH_TOKEN:?AUTH_TOKEN not configured}"
    depends_on:
      - mediator
      - jobclient
    ports:
      - "8815:8815"
    deploy:
      replicas: 1
    restart: unless-stopped
volumes:
  mediator-volumes:
  mediator-backup:
```

5.3 kubernetes configuration

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mediator
  labels:
    app: mediator
spec:
  selector:
    matchLabels:
      app: mediator
  strategy:
    type: Recreate
  template:
```

```
metadata:
  labels:
    app: mediator
spec:
  containers:
  - name: mediator
    image: container-registry.example.com/iviews-mediator:<image-tag-1>
    ports:
    - containerPort: 30000
    - containerPort: 30001
    env:
    - name: IV_PASSWORD
      valueFrom:
        secretKeyRef:
          name: iviews-secret
          key: MEDIATOR_PASSWORD
    - name: IV_INTERFACES
      value: "http://0.0.0.0:30000,cnp://0.0.0.0:30001"
    - name: IV_SCHEDULED_JOBS
      value: "job-bu,job-gc"
    - name: IV_JOB_BU_VOLUME_PATTERN
      valueFrom:
        configMapKeyRef:
          name: iviews-config
          key: VOLUME
    - name: IV_JOB_BU_BACKUP_INTERVAL
      value: "1"
    - name: IV_JOB_BU_BACKUP_TIME
      value: "22:22"
    - name: IV_JOB_BU_BACKUPS_TO_KEEP
      value: "1"
    - name: IV_JOB_GC_VOLUME_PATTERN
      valueFrom:
        configMapKeyRef:
          name: iviews-config
          key: VOLUME
    - name: IV_JOB_GC_GARBAGE_COLLECT_INTERVAL
      value: "1"
    - name: IV_JOB_GC_GARBAGE_COLLECT_TIME
      value: "22:33"
    volumeMounts:
    - mountPath: /mediator/volumes
      name: mediator-volumes
    - mountPath: /mediator/backup
      name: mediator-backup
  volumes:
  - name: mediator-volumes
    persistentVolumeClaim:
      claimName: mediator-volumes
  - name: mediator-backup
    persistentVolumeClaim:
      claimName: mediator-backup
  securityContext:
```

```
        runAsUser: 10000
        runAsGroup: 10000
        fsGroup: 10000
--
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mediator-volumes
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
--
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mediator-backup
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
--
apiVersion: v1
kind: Service
metadata:
  name: mediator
  labels:
    app: mediator
spec:
  selector:
    app: mediator
  ports:
  - name: mediator-http
    port: 30000
  - name: mediator-cnp
    port: 30001
--
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: mediator
spec:
  policyTypes:
  - Ingress
  podSelector:
    matchLabels:
      app: mediator
  ingress:
  - from:
```

```
    - podSelector:
        matchLabels:
          app: jobclient
    - podSelector:
        matchLabels:
          app: bridge
--
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jobclient
  labels:
    app: jobclient
spec:
  selector:
    matchLabels:
      app: jobclient
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: jobclient
    spec:
      containers:
      - name: jobclient
        image: container-registry.example.com/iviews-mediator:<image-tag-2>
        imagePullPolicy: IfNotPresent
        env:
        - name: IV_HOST
          value: "mediator:30001"
        - name: IV_VOLUME
          valueFrom:
            configMapKeyRef:
              name: iviews-config
              key: VOLUME
        - name: IV_AUTHENTICATION
          valueFrom:
            secretKeyRef:
              name: iviews-secret
              key: AUTHENTICATION
        - name: IV_JOB_POOLS
          value: script,index,query
      securityContext:
        runAsUser: 10000
        runAsGroup: 10000
--
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bridge
  labels:
    app: bridge
```

```
spec:
  selector:
    matchLabels:
      app: bridge
  strategy:
    type: Recreate
  replicas: 1
  template:
    metadata:
      labels:
        app: bridge
    spec:
      containers:
      - name: bridge
        image: container-registry.example.com/iviews-bridge:<image-tag-3>
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8815
        env:
        - name: IV_HOST
          value: "mediator:30001"
        - name: IV_VOLUME
          valueFrom:
            configMapKeyRef:
              name: iviews-config
              key: VOLUME
        - name: IV_AUTHENTICATION
          valueFrom:
            secretKeyRef:
              name: iviews-secret
              key: AUTHENTICATION
      securityContext:
        runAsUser: 10000
        runAsGroup: 10000
--
apiVersion: v1
kind: Service
metadata:
  name: bridge
  labels:
    app: bridge
spec:
  type: ClusterIP
  selector:
    app: bridge
  ports:
  - name: bridge
    port: 8815
--
apiVersion: v1
kind: Secret
metadata:
  name: iviews-secret
```

```
type: Opaque
data:
  MEDIATOR_PASSWORD: <encoded-mediator-password>
  AUTHENTICATION: <encoded-system-account>
--
apiVersion: v1
kind: ConfigMap
metadata:
  name: iviews-config
data:
  VOLUME: <knowledge-graph-name>
```