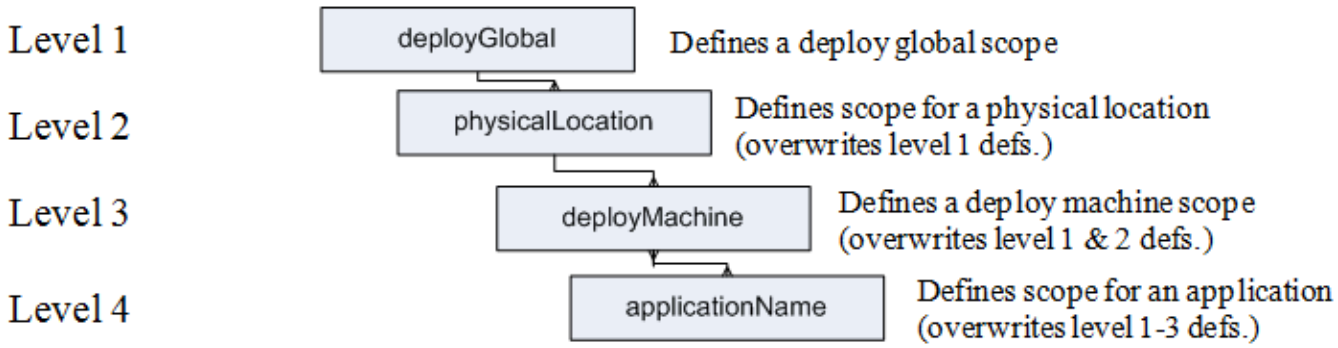
The Deploy Configuration File

Contents

- Settings scope
- Deploy scope
  - Parameters
  - Application Instance Id
- Limitations and Requirements
- Configuration example
  - Deploy Global
  - Physical Locations
  - Machine
  - Application
  - BitarchiveApplication
  - HarvestControllerApplication
  - IndexServerApplication and ViewerProxyApplication=
  - BitarchiveMonitorApplication

The deploy configuration file contains the definitions for the installation and distribution of !NetarchiveSuite. This involves the scopes for the levels in the figure below, and their settings.

This figure also shows the pattern of inheritance of the settings (`physicalLocation` inherits settings and parameters from `deployGlobal`, `deployMachine` inherits from `physicalLocation`, etc.).



These levels can have several instances of the levels below them.

Settings scope

The settings scope is described in the Configuration Manual for NetarchiveSuite. It is no longer required that every variable within the settings scope is explicitly defined for an application, since the undefined variables are replaced by the default settings, when the application is run.

Each level (in the figure at the beginning of this section) inherits the settings from the level above it (until deployGlobal), though only the variables which is not explicitly defined at the current level. The content of the settings scope at the application level (level 4) is printed into an application specific settings file, which is used for running the application.

Some parts within the settings scope is used by deploy, and they will be described in the following section.

Deploy scope

The levels in the figure can have an instance of the settings scope defined. These settings are inherited through the hierarchy.

The scope levels of Deploy:

- <deployGlobal>
  . Defines a deploy global 1. level scope where settings can be set to overwrite setting defaults.
- <thisPhysicalLocation name="...">
  . Defines the 2. level scope for a physical location. The settings for this scope will overwrite the settings for the 1. level scope (deployGlobal). The attribute 'name' for thisPhysicalLocation overwrites settings.common.thisPhysicalLocation.
- <deployMachine name="..." os="...">
  . Defines a deploy machine 3. level scope where common settings for the machine and the applications running in the machine can be set. These settings will overwrite 1. and 2. level settings. The attribute 'name' for the machine is the network name the machine, and will be used for communicating with the machine. The attribute 'os' is optional and defines the operating system on the machine. If 'os' is not set or has value different from 'windows' (not case sensitive), then the default 'Linux/Unix' is used.
- <applicationName name="...">
  . Defines the 4. level scope where the application specific settings are placed. These settings will overwrite the inherited 1., 2. and 3. level

settings. The attribute 'name' for applicationName is used for calling the application. Only the last part of the name is used for all purposes (except running the application) and it overwrites settings.common.applicationName
(e.g. the application `dk.netarkivet.archive.bitarchive.BitarchiveApplication` will have the name BitarchiveApplication).
If the application has an specific applicationInstanceId, it is specified under settings.
One level can have several instances of a lower level (e.g. a deployMachine can have several applicationName, and not vice versa).

This will look like the following:

```
<deployGlobal>
    <thisPhysicalLocation name="myPhysicalLocation">
        <deployMachine name="myMachine" os="linux">
            <applicationName name="myApplication">
            </applicationName>
            <applicationName name="myOtherApplication">
            </applicationName>
        </deployMachine>
        <deployMachine name="myOtherMachine" os="windows">
            <applicationName name="myApplication">
            </applicationName>
        </deployMachine>
    </thisPhysicalLocation>
</deployGlobal>
```

This configuration has one physical location with two machines, one with Linux/Unix and one with Windows. The Linux/Unix machine has two applications, 'myApplication' and 'myOtherApplication', while the Windows machine has only one application, 'myApplication'.

Parameters

Each of the above scopes can have several of the following parameters defined. These parameters can be applied to each of the above scopes, and they are inherited from the parent scope in the same way as settings.

The parameter scopes the levels can have:

- <deployClassPath>
  . Defines a class path to be added for running an application. Note: several additional class paths can be specified within a scope, but new definitions in inner scopes will overwrite outer scopes.
- <deployJavaOpt>
  . Defines a Java option for an application. Note: several additional java options can be specified within a scope, but new definitions in inner scopes will overwrite all outer scopes.
- <deployInstallDir>
  . Defines the installation directory for a deployMachine, can only handle one deployInstallDir. Note: only one install directory is supported (if several, a warning is placed in the log and the first install directory is used).
- <deployMachineUserName>
  . Defines the user name for a deployMachine. This is used when communicating with the machine. Note: only one machine user name is supported (if several, a warning is placed in the log and the first machine user name is used).
- <deployDatabaseDir>
  . Defines the directory for the database to unzipped. This directory can be full path or path relative to install directory. It is an optional parameter for defining where a machine should have the database unpacked, and if the machine does not include this parameter it will not have the database unpacked. Also it requires the settings.common.database.url set. Note: This must be set on the machines where the database are to be unpacked. Only one database directory is supported (if several, a warning is placed in the log and the first database directory is used).
- <deployBitpreservationDatabaseDir>
  . Defines the directory for the bitpreservation database to be unzipped. This directory can be full path or path relative to the installation directory. It is an optional parameter for defining where a machine should have the bitpreservation database unpacked, and if a machine does not have this parameter it will not have the database unpacked.
  An example of how this works is given below.

```
<deployGlobal>
    <deployClassPath>lib/dk.netarkivet.common.jar</deployClassPath>
    <deployClassPath>lib/dk.netarkivet.archive.jar</deployClassPath>
    <deployJavaOpt>-Xmx1536m</deployJavaOpt>
    <thisPhysicalLocation name="myPhysicalLocation">
        <deployMachineUserName>myUserName</deployMachineUserName>
        <deployMachine name="myLinuxMachine">

<deployInstallDir>/home/myUserName/myInstallationDirectory</deployInstallD
ir>
            <deployDatabaseDir>myDatabaseDir</deployDatabaseDir>
            <settings>
                <common>
                    <database>
                         <url>jdbc:derby:myDatabaseDir/fullhddb</url>
                    </database>
                </common>
            </settings>
            <applicationName name="myLinuxApplication">
            </applicationName>
        </deployMachine>
        <deployMachine name="myWindowsMachine" os="windows">
            <deployInstallDir>C:\myInstallationDirectory</deployInstallDir>
            <deployJavaOpt>-Xmx1150m</deployJavaOpt>
            <applicationName name="myWindowsApplication">

<deployClassPath>lib/dk.netarkivet.common.jar</deployClassPath>

<deployClassPath>lib/dk.netarkivet.harvester.jar</deployClassPath>

<deployClassPath>lib/dk.netarkivet.viewerproxy.jar</deployClassPath>
            </applicationName>
        </deployMachine>
    </thisPhysicalLocation>
<deployGlobal>
```

This defines two different machines each with a single application. These machines have different operating systems (one with windows and one with linux), and therefore they have different installation directories and Java options.

The Linux machine inherits the Java option `-Xmx1536m` from the physical location, which inherits it from deployGlobal. The Windows machine has a Java option specified and does therefore not inherit deployGlobal Java option.

The deployDatabaseDir is only specified on the Linux machine, and the database will therefore be unpacked only on this machine. It is specified in settings.common.database.url what type the database is, and where the it is found after it is unpacked. If a specific database is not given as parameter when calling deploy the default Derby database 'fullhddb.jar' is used.

The application myLinuxApplication on the Linux machine does not have any class paths specified, and does therefore inherit the `lib/dk.netarkivet.common.jar` and `lib/dk.netarkivet.archive.jar` all the way from deployGlobal (through thisPhysicalLocation and deployMachine).

On the other hand does myWindowsApplication on the Windows machine not inherit these libraries, since it has its own class paths specified. It has the libraries `lib/dk.netarkivet.common.jar`, `lib/dk.netarkivet.harvester.jar` and `lib/dk.netarkivet.viewerproxy.jar` in the class path, and does therefore not have the `lib/dk.netarkivet.archive.jar` since it is neither specified nor inherited.

The myLinuxApplication will be called with the following command:

```
java -Xmx1536m -cp
lib/dk.netarkivet.common.jar:lib/dk.netarkivet.archive.jar
myLinuxApplication
```

The myWindowsApplication will be called with the following command:

```
java -Xmx1150m -cp
lib/dk.netarkivet.common.jar;lib/dk.netarkivet.harvester.jar;lib/dk.netark
ivet.viewerproxy.jar myWindowsApplication
```

The class paths are separated with ':' on Linux/Unix and with ';' on Windows.

Application Instance Id

The scope settings.common.applicationInstanceId defines identification of a single application instance (e.g. suffix for application specific scripts, suffix for directory to place files etc.). This is needed in cases where there are more instances of the same application are placed on the same machine (e.g. BitarchiveMonitors)

An example of two identical applications with different application instance id on the same machine is given below:

```
<deployGlobal>
    <thisPhysicalLocation name="myPhysicalLocation">
        <deployMachine name="myMachine">
            <applicationName
name="dk.netarkivet.archive.bitarchive.BitarchiveApplication">
                <settings>
                    <common>

<applicationInstanceId>myFirstInstance</applicationInstanceId>
                    </common>
                </settings>
            </applicationName>
            <applicationName
name="dk.netarkivet.archive.bitarchive.BitarchiveApplication">
                <settings>
                    <common>

<applicationInstanceId>mySecondInstance</applicationInstanceId>
                    </common>
                </settings>
            </applicationName>
        </deployMachine>
    </thisPhysicalLocation>
</deployGlobal>
```

These application will be called !BitarchiveApplication_myFirstInstance and !BitarchiveApplication_mySecondInstance respectivly.

Limitations and Requirements

And deploy has the following requirements:

- The environmentName (settings.common.environmentName) has to be set in settings on the global level.
- The environmentName (settings.common.environmentName) must be a combination of digits (0-9) and the letters (a-z, lower or upper case). Deploy fails if the environmentName contains other characters.
- Different environmentNames between physical location level, machine level and application level is not supported (or meaningful).
- Databases are not supported on Windows.
- The GUIApplication and the !ArcRepositoryApplication must be placed on the same machine.
- The install directory on Windows must be "C:\Documents and Settings\user\", where user is the username on the machine. Except Windows Vista (or equivalent server os), where the directory must be `C:\Users\user`, where user is the username on the machine.
- All applications on the same machine with jmx login for monitor must have identical login.
- All applications on the same machine with jmx login for heritrix must have identical login.
- When creating a test instance, the arguments 'http-port' and 'offset' is only supported as 4 digit numbers.
- Every physical location, machine and application must have the name attribute defined.
- Deploy does not handle network connection permissions. E.g. if there is a firewall, it has to be setup to allow the applications in NetarchiveSuite to communicate with each other.
- Permission to create the wanted directories is required.
- The unzip command (or program) has to be accessible through 'ssh'.
- Two instances of the same application on the same machine must have different applicationInstanceIds.
- Several instances of the same setting cannot extend one setting. E.g. a physical location with several instances of the remoteFile defined need to have each remoteFile setting completely defined, since they are not extended by a single remoteFile in the global settings.

The deploy configuration has the following limitations in comparison to the manual installation.

- Only embedded Derby databases have been tested with the new Deploy, and other databases have to be installed manually.
  The limitations and requirements for the configuration of the applications can be found in the Configuration Manual. Specific for deploy are the following:

- Every application must have a jmx-port and rmi-port, and they must be unique for the machine where the application is running.
- dk.netarkivet.harvester.harvesting.!HarvestControllerApplication does not run on Windows machines.
- A dk.netarkivet.archive.bitarchive.!BitarchiveApplication must have at least one settings.archive.bitarchive.baseFileDir defined.
- Only the dk.netarkivet.archive.bitarchive.!BitarchiveApplication is properly tested on the Windows platform. Some of the other applications should work, though they have not been tested enough to say for certain.
- If a machine has several instances of dk.netarkivet.archive.bitarchive.!BitarchiveApplication, then each application must have a unique temporary file directory defined (settings.common.tempDir).

Configuration example

Here is an example of a configuration file for deploy.

Example of deploy configuration file

The following part of this section describes how to change this configuration file template to fit your specific system.
This describes how to make the changes, scope for scope, to fit a system with the same structure,
and it describes how to expand the scopes with new machines and applications.

Deploy Global

The `deployGlobal` scope contains two parts: the parameters and the settings.

Just leave the `<deployClassPath` parameters, since they will be overwritten for the applications which need other libraries.
The `<deployJavaOpt>-Xmx1536m</deployJavaOpt>` parameter just sets the maximum heap size to 1.5 GB (1536 MB).
This value should not be larger than the amount of accessible memory on a machine.

Within the settings scope of `deployGlobal` the following needs to be done.

The environment name is not required to be changed for the system to work, though it is usually a good idea to change this to a more appropriately name for the installation or system.
This is the settings at 'settings.common.environmentName'.

```
<settings>
    <common>
        <environmentName>test</environmentName>
    <common>
<settings>
```

The replicas should be changed to fit the system.
A replica will generally be connected to a specific physical location, though a physical location can have several replicas.
These settings can be found under 'settings.common.replicas'.

```
<settings>
    <common>
        <replicas>
            <replica>
                <replicaId>A</replicaId>
                <replicaName>ReplicaA</replicaName>
                <replicaType>bitArchive</replicaType>
            </replica>
            <replica>
                <replicaId>B</replicaId>
                <replicaName>ReplicaB</replicaName>
                <replicaType>bitArchive</replicaType>
            </replica>
        </replicas>
    <common>
<settings>
```

The JMS-broker is defined at the global level, and it should be set to the administation machine, e.g. the machine with the `dk.netarkivet.common.webinterface.GUIApplication`, the `dk.netarkivet.archive.arcrepository.ArcRepositoryApplication` and the instances of `dk.netarkivet.archive.bitarchive.BitarchiveMonitorApplication` should be run.
This is defined in the settings: 'settings.common.jms.broker'.

```
    <settings>
        <common>
            <broker>kb-test-adm-001.kb.dk</broker>
        <common>
    <settings>
```

If more replicas are wanted, they have to be defined in the settings at the `deployGlobal` level.
Each replica needs a unique `replicaId` and `replicaName`, and it also needs the following applications: `dk.netarkivet.archive.bitarchive.BitarchiveApplication`, and `dk.netarkivet.archive.bitarchive.BitarchiveMonitorApplication`.

Physical Locations

The configuration example file has two physical locations: EAST and WEST.
Every physical location need to have a unique name.

```
<thisPhysicalLocation name="EAST">
    ...
</thisPhysicalLocation>
<thisPhysicalLocation name="WEST">
    ...
</thisPhysicalLocation>
```

For the settings of a physical location the following need to be done.
A physical location needs to know which replica it uses.
This replicaId has to be amongst the replicas defined in the `deployGlobal` scope.
It has the path: 'settings.common.useReplicaId'.

```
<settings>
    <common>
        <useReplicaId>A</useReplicaId>
    </common>
</settings>
```

If using FTPRemoteFile, it is necessary to specify a machine on which an ftp server is running, together with valid login credentials, for example

```
<remoteFile>
    <serverName>kb-test-har-001.kb.dk</serverName>
    <userName>ftptestuser</userName>
    <userPassword>ftptestpasswd</userPassword>
</remoteFile>
```

The notifications settings should be setup to tell where mails should be sent.
The receiver should be changed to the mail of the administrator of the system.

```
<notifications>
    <sender>example@netarkivet.dk</sender>
    <receiver>example@netarkivet.dk</receiver>
</notifications>
```

It is currently not possible to have more than two physical locations, but this problem will be dealt with, and it will be possible in a future release.

Machine

The name of a machine has to be change to the network ID, e.g. either network name or IP address.
The 'os' attribute should only be set for the windows machines, which can only run applications of the instance `dk.netarkivet.archive.bitarchive.BitarchiveApplication`.

```
<deployMachine os="windows" name="kb-dev-bar-011.bitarkiv.kb.dk">
```

Change the following parameters to fit to the machine definition:
A machine needs to have the following parameters defined (they can also be defined at the physicalLocation level, and then just be inherited).

```
<deployMachineUserName>test</deployMachineUserName>
<deployInstallDir>/home/test</deployInstallDir>
```

There are no specific settings required at the machine level, which is not inherited by the outer scopes.
And therefore no settings to change to fit to your system.

A new machine has to be created within a physical location scope.
It requires the name attribute, and the parameters `deployMachineUserName` and `deployInstallDir` has to be defined or inherited.
The parameter `deployDatabaseDir` is required, if the machine runs an application which requires a database.

Application

All applications need the following settings defined under `settings.common.jmx`:

```
<port>8100</port>
<rmiPort>8300</rmiPort>
```

These port values must be unique for the machine, where the application should run.

A new application needs the name attribute to be defined as the name in the classpath for the application. E.g:

```
<applicationName
name="dk.netarkivet.common.webinterface.GUIApplication">
```

It is important to notify that when a new application is added to a machine, which already has an application of the same instance, these applications must have the `settings.common.applicationInstanceId` defined with different values.

Some of the applications require some specific settings to be defined.
This is described in the following specifically

BitarchiveApplication

The `dk.netarkivet.archive.bitarchive.BitarchiveApplication` requires the settings `settings.archive.bitarchive.baseFileDir` to be defined.
This path should be changed, and it has to be changed if the drive/partition in the path does not exist on the machine.

HarvestControllerApplication

For the `dk.netarkivet.harvester.harvesting.HarvestControllerApplication` the following settings defined under `settings.harvester.harvesting.heritrix` should be changed to fit your system: `guiPort` and `jmxPort`.

A new instance of the `dk.netarkivet.harvester.harvesting.HarvestControllerApplication` requires the settings `settings.harvester.harvesting.queuePriority` to be defined to either `LOWPRIORITY` or `HIGHPRIORITY`.
A system requires at least one !HarvestControllerApplication with each priority.

IndexServerApplication and ViewerProxyApplication=

Both the `dk.netarkivet.archive.indexserver.IndexServerApplication` and `dk.netarkivet.viewerproxy.ViewerProxyApplication` should have the `settings.common.http.port` and the `settings.viewerproxy.baseDir'` changed to fit your system.

BitarchiveMonitorApplication

All the instances of `dk.netarkivet.archive.bitarchive.BitarchiveMonitorApplication` should be placed on the same machine as the `dk.netarkivet.common.webinterface.GUIApplication`.
These applications monitors the BitarchiveApplications at a given replica, though they do not have to be on the same physical location.
They should therefore have the `settings.common.useReplicaId` defined.