Krestfield PKCloud

Installation and Configuration Guide

version 2.0

Overview

PKCloud runs as a web application requiring a servlet container to run. The Apache Tomcat container is shipped with the installations and is used by default, although other suitable containers can be used if preferred

The application may be installed using the provided installers, where all required components are also provided (e.g. java and tomcat) or a more bespoke setup can be configured by deploying the stand-alone war (Web Archive) file onto existing infrastructure

Note: When the server is first started, you may logon with the following credentials that have administrator privileges

Username: **admin**
Password: **password**

You will immediately be prompted to change the password

Installation

PKCloud is supported on the following operating systems:

- Windows Server (2012 R2, 2016, 2019)
- Solaris (10, 11.x)
- Linux (Amazon Linux, CentOS, Debian, Red Hat, Ubuntu)

The steps below detail the installation steps for Windows, Solaris and Linux

Run the **pkcloud.msi** installer. You may edit or retain the default locations then click **Next** to complete the installation

If the defaults are accepted the installation will be located on the system drive at this location:

```
\Program Files\Krestfield\PKCloud
```

The application may then be started and stopped via scripts contained within the `.\bin` directory beneath this location

e.g. `C:\Program Files\Krestfield\PKCloud\bin`

The user or group that will be starting and stopping the application must have write permissions on the installation folder (e.g. `C:\Program Files\Krestfield\PKCloud`) folder. To set this, perform the following steps:

1. Right click the **Installation folder** (e.g. `C:\Program Files\Krestfield\PKCloud`), choose **Properties** then select the **Security** tab
2. Click the **Edit** button
3. Locate the user/group (or add using the **Add…** button) and select the **Allow** for **Write** permission. Click **OK** to close the dialog

Service Installation

The Tomcat server may be installed as a Service (rather than as a stand-alone application which requires the `startpkcloud.bat` and `stoppkcloud.bat` scripts to be run)

The Service requires the following environment variables to be set:

CATALINA_HOME

```
CATALINA_HOME = [PKCloud Installation]\apache-tomcat-[version]
```
e.g.
```
CATALINA_HOME = C:\Program Files\Krestfield\PKCloud\apache-tomcat-9.0.34
```

JAVA_HOME

```
JAVA_HOME = [PKCloud Installation]\openjdk-[version]
```
e.g.
```
JAVA_HOME = C:\Program Files\Krestfield\PKCloud\openjdk-11.0.1
```

JRE_HOME

```
JRE_HOME = %JAVA_HOME%
```

These variables may be set by performing the following steps:

1. From the start menu, type **Control Panel**

2. From the Control Panel, click **System**

3. From the left-hand pane, click **Advanced system settings**

4. From the System Properties dialog, select the **Advanced** tab and click the **Environment Variables** button

5. Under System Variables, click **New…**

6. For the Variable Name, enter **CATALINA_HOME**

7. For the Variable Value, enter the full path to the tomcat directory in the PKCloud installation e.g. `C:\Program Files\Krestfield\PKCloud\apache-tomcat-9.0.34`

8. Click **OK**

9. Repeat for **JAVA_HOME** and **JRE_HOME**

Once the variables have been set correctly, the service may be installed by performing these steps:

1. Open a command prompt, as Administrator

2. Navigate to the **tomcat bin** directory in the PKCloud installation e.g. **C:\Program Files\Krestfield\PKCloud\apache-tomcat-9.0.34\bin**

3. At the prompt, type: **service.bat install** and press **Enter**. This will install the Service

4. To check that the Service has installed, from the Start menu, type **Services** and press **Enter**

5. Ensure that the Apache Tomcat service is now present. Right click the **Apache Tomcat Service** and set the Start-up type to be **Automatic** (leave at Manual if this service should not auto-start on reboot)

6. Click **OK**

Solaris

The Solaris installer does not come packaged with a java runtime

If Java is not already installed on the system, install ensuring that it meets the minimum requirements (Java 8 or above) and, if required (Java 8), that the Unlimited Strength Jurisdiction Policy Files have also been installed

Navigate to the directory to install PKCloud (e.g. `/opt/apps`) and unzip and untar the **pkcloud.tar.gz** file as follows:

```
$ tar xpvzf pkcloud.tar.gz
```

This will create the `./pkcloud` directory e.g. `/opt/apps/pkcloud`

Navigate to the bin directory e.g. `/opt/apps/pkcloud/bin` and edit the **envs.sh** file as follows:

1. Edit the **PKCLOUDHOME** setting to point to the full path of the **./pkcloud** directory that was created above e.g.

   ```
   PKCLOUDHOME=/opt/apps/pkcloud
   ```

2. Edit the **JRE_HOME** export setting to point to the java installation e.g. /opt/java/jdk-11.0.1 e.g.

   ```
   export JRE_HOME=/opt/java/jdk-11.0.1
   ```

3. Save **envs.sh**

Linux

Navigate to the directory to install PKCloud (e.g. `/opt/apps`) and unzip and untar the **pkcloud.tar.gz** file as follows:

```
$ tar xpvzf pkcloud.tar.gz
```

This will create the `./pkcloud` directory e.g. `/opt/apps/pkcloud`

Navigate to the bin directory e.g. `/opt/apps/pkcloud/bin` and edit the **envs.sh** file as follows:

4. Edit the **PKCLOUDHOME** Setting to point to the full path of the **./pkcloud** directory that was created above e.g.

   ```
   PKCLOUDHOME=/opt/apps/pkcloud
   ```

5. Save **envs.sh**

Start-up and Shutdown

Linux/Solaris

If the default Tomcat container is used, navigate to the `/bin` directory under the installation. The following scripts can be used to start and stop the Tomcat server:

```
startpkcloud.sh
stoppkcloud.sh
```

Windows

If the server is being started manually, navigate to the .\bin directory under the installation e.g. `C:\Program Files\Krestfield\PKCloud\bin`. The following scripts will start and stop the Tomcat server:

```
startpkcloud.bat
stoppkcloud.bat
```

The following script will open a browser pointing to the PKCloud application (if the default port of 8080 is used):

```
gotopkcloud.bat
```

If the Service has been installed, starting and stopping is controlled via the Tomcat Windows service

Configuration

Tomcat Listening Port

By default Tomcat listens on port 8080. To change this, open the `./conf/server.xml` file and locate the following section.

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

Update the port value (highlighted above) to that required and restart the application. Note: in UNIX/Linux systems port values below 1024 require higher privileges (i.e. port 80 may require Tomcat to be run as root). Port redirection (redirecting from port 80 to a higher port) or the use of a reverse proxy may be put in place to continue running the server on higher ports

Tomcat SSL Setup

There are several options when configuring TLS within Apache Tomcat and the Tomcat documentation will contain the most up to date information. But the following commands show how to configure a JKS (Java Key Store) file with a TLS certificate (and configure the tomcat settings to use this file)

1. Navigate to the Tomcat `conf` directory e.g.

   Windows:
   ```
   C:\Program Files\Krestfield\PKCloud\apache-tomcat-9.0.34\conf
   ```

   Linux/Unix:
   ```
   /opt/pkcloud/apache-tomcat-9.0.34/conf
   ```

2. Run the following command to generate a key-pair. This example creates a **2048** bit key pair called **testkeys** in a keystore file called **keystore.jks**. Update the highlighted settings to your requirements:

   ```
   keytool -genkey -alias testkeys -keyalg RSA -keysize 2048 -keystore keystore.jks
   ```

   You will be prompted for the following:

   ```
   Enter keystore password: YourPassword
   Re-enter new password: YourPassword
   What is your first and last name?
     [Unknown]:  server.com
   What is the name of your organizational unit?
     [Unknown]:  Dev
   What is the name of your organization?
     [Unknown]:  Company
   ```

```
What is the name of your City or Locality?
  [Unknown]:  Paddington
What is the name of your State or Province?
  [Unknown]:  London
What is the two-letter country code for this unit?
  [Unknown]:  GB
Is CN=server.com, OU=Dev, O=Company, L=Paddington, ST=London, C=GB correct?
  [no]:  yes
```

Complete the details (example values are given above), noting:

- `YourPassword` is a password of your choosing. This is used to protect the keystore
- `server.com` should be the DNS name of the server e.g. if you will be accessing the application as follows: `https://signingserver.corp.com/pkcloud` this value should be set to `signingserver.corp.com`

The other settings should be set as per your requirements (company name, location etc.)

3. The following command will create a CSR (certificate signing request), saving to a file named **test.csr**:

```
keytool -certreq -alias testkeys -keyalg RSA -file test.csr -keystore keystore.jks
```

Ensure you use the same values for the alias (`testkeys`) and keystore (`keystore.jks`) as used above

You will be prompted to re-enter the password (`YourPassword`) used above

4. Take the csr file generated (e.g. `test.csr`) and process at a CA to obtain a TLS certificate. Note that the certificate must have a *DNS Subject Alternative Names* extension, an entry in which must match that of the server name (e.g. server.com in the example above)

   Also obtain the root and any intermediate certificates

   Once all certificates have been received, you can import them into the keystore as follows…

5. The following command will import the root certificate into the keystore:

```
keytool -import -alias root -keystore keystore.jks -trustcacerts -file root.cer
```

6. If there are intermediate certificates in the chain, they can be imported as follows:

```
keytool -import -alias int -keystore keystore.jks -file int.cer
```

Note: The aliases for the root and intermediate certificates can be anything of your choosing and do not have to be `root` and `int` as shown above

7. Finally, the following command will import the issued TLS certificate (e.g. ee.cer)

```
keytool -import -alias testkeys -keystore keystore.jks -file ee.cer
```

   Note: In this case the alias (`testkeys`) must match what has been used in steps 2 and 3 above

   Ensure the following message is seen:

```
Certificate reply was installed in keystore
```

The keystore configuration is now complete and the `keystore.jks` file will contain the keys and TLS certificate

To configure the Tomcat configuration to now make use of this keystore, the following can be performed:

1. In the same directory (`conf`) edit the `server.xml` file. Beneath the comment starting:
   `<!-- Define a SSL/TLS HTTP/1.1 Connector on port 8443`
   add the following

   ```
    <Connector
           protocol="org.apache.coyote.http11.Http11NioProtocol"
           port="443" maxThreads="200"
           scheme="https" secure="true" SSLEnabled="true"
           keystoreFile="conf/keystore.jks" keystorePass="YourPassword"
           clientAuth="false" sslProtocol="TLS"/>
   ```

2. Restart the Tomcat Server and navigate to your server via https e.g. `https://server.com`. The certificate just processed should now be protecting the link

REST API SSL Setup

The REST API can be secured with TLS by checking the **Use TLS** option in the Server configuration. This requires the configuration of the JKS (Java Key Store) named pkrest.jkslocated within the ./webapps/ezsign/rest folder of the installation e.g.

Windows:

```
C:\Program Files\Krestfield\PKCloud\apache-tomcat-9.0.34\webapps\pkcloud\ezsign\rest\
pkrest.jks
```

Linux/Unix:

```
/opt/pkcloud/apache-tomcat-9.0.34/webapps/pkcloud/ezsign/rest/pkrest.jks
```

The exact location for your installation will be displayed when selecting the **Use TLS** option on the console

This file may be created using a utility such as keytool but any tool that can manipulate JKS files can be used. The following instructions are for the java provided keytool utility

First, note the DNS name of the end point. For example, if your REST API calls should target `https://api.myserver.com` then the DNS name will be `api.myserver.com`

1. Start by creating a key pair by running the following command:

```
keytool -genkey -alias pkcloudrest -keyalg RSA -keysize 2048 -keystore pkrest.jks
```

Note: the `keysize`, `keyalg` and `alias` settings can be changed to other values if desired, but the alias must be consistent throughout the following commands

You will be prompted for the following:

```
Enter keystore password: YourPassword
Re-enter new password: YourPassword
What is your first and last name?
  [Unknown]:  api.myserver.com
What is the name of your organizational unit?
  [Unknown]:  Dev
What is the name of your organization?
  [Unknown]:  Company
What is the name of your City or Locality?
  [Unknown]:  Paddington
What is the name of your State or Province?
  [Unknown]:  London
What is the two-letter country code for this unit?
  [Unknown]:  GB
Is CN=server.com, OU=Dev, O=Company, L=Paddington, ST=London, C=GB correct?
  [no]:  yes
```

Complete the details (example values are given above but can be adjusted as required), ensuring that the given first and last name is the server DNS name

2. Create a certificate request, saving to a file named `rest.csr`:

```
keytool -certreq -alias pkcloudrest -keyalg RSA -file rest.csr -keystore pkrest.jks
```

   You will be prompted to re-enter the password

3. Take the csr file generated and process at a CA to obtain a TLS certificate. Note that the certificate must have a *DNS Subject Alternative Names* extension which has a DNS entry that must match that of the server name (e.g. api.myserver.com in the example above). Most public PKI services will automatically populate this for you

   Also obtain the root and any intermediate certificates associated with the CA

4. Now import the root certificate. The following command will import the file named `root.cer` as a trusted root

```
keytool -import -alias root -keystore pkrest.jks -trustcacerts -file root.cer
```

5. Import any intermediate certificate(s) (if required)

```
keytool -import -alias int -keystore pkrest.jks -file int.cer
```

6. Finally, import the end-entity TLS certificate (e.g. in this example `pkrest.cer` – which is the certificate returned from the CSR generated above)

```
keytool -import -alias pkcloudrest -keystore pkrest.jks -file pkrest.cer
```

7. Copy the `pkrest.jks` file to the location mentioned above (`../webapps/pkcloud/ezsign/rest/`)

8. Back at the console, for the server configuration, enter the **keystore password** entered in step 1 for the JKS File Password

When the server is restarted the REST API will now be protected via TLS and the certificate created above

Environment Variables

The following environment variables can be set

Master Password

Requiring that the master password is entered when the main server is started offers the best security as the master password then does not need to be present on the system. However, this requires that the master password is entered each time the system is restarted.

For convenience the master password may be set within the following environment variable:

```
pkcloud_masterpass
```

e.g.

```
pkcloud_masterpass=Mast3rP4ssw0rd
```

If this variable is set the server will not require that the master password is entered upon starting

Ensure this is not set as a system wide environment variable where other processes or users may be able to read the master password

Another option is to store the master password within a text file. The file must only contain the password text and must be accessible by the PKCloud process. The filename may be provided in the following environment variable:

```
pkcloud_masterpass_file
```

e.g.

```
pkcloud_masterpass_file=/opt/pkcloud/masterpass.txt
```

Ensure that only the PKCloud process (i.e. the user running this process) has access to this file to prevent other processes or users accessing it

Note: If both variables are set the `pkcloud_masterpass` variable will be used

Logging

To increase the level of logging in the pkcloud.log file, set the following environment variable:

```
pkcloud_channel_loglevel
```

to a value between 0 and 4 e.g. `pkcloud_channel_loglevel=3`

This value defaults to 0 which outputs the lowest logging output level. Level 1 will report more errors, level 2 more errors and warnings, level 3 more errors, warnings and events and level 4 is the debug level, outputting the maximum log detail

Heap Sizes

When servers are started they run with a maximum heap size set to 256Mb. This is normally sufficient but can be altered. For example, installations with a minimal amount of memory may wish to reduce this so that the java virtual machine allocates less memory. In systems that may be utilising a large number of channels or threads it may be necessary to increase this value

To increase this value for the server, set the following environment variable:

```
pkcloud_ezsign_maxheapmb
```

To the max heap size in Mb e.g. `pkcloud_ezsign_maxheapmb=128` indicates that 128Mb will be allocated

The REST API runs as a separate java process and its maximum heap size can also be controlled by setting this environment variable:

```
pkcloud_rest_maxheapmb
```

E.g. `pkcloud_rest_maxheapmb=128` - indicates a maximum heap allocation of 128mb. If no value is set this will default to 64mb

Auto Startup

By default, when the container (e.g. apache tomcat) is started, any configured EzSign and Restful API instances will not automatically start

To enable this, so that they are automatically started, set the following system property

```
pkcloud_servers_startonstartup
```

to **true** e.g.

```
pkcloud_servers_startonstartup=true
```

Note: That for the servers to start successfully the master password must also be set via the environment variable mentioned above