How to do Conan Packaging and Deploy to QA Instance

Last edited by **Mohanasamy, Sabapathi (S.)** 4 months ago

Pre-Requisites:

What is conan and why it is used?

- Conan is a dependency and package manager for C and C++ languages.
- It is a free and open-source, and works on all platforms.
- It also integrates with all build systems like CMake, Visual Studio (MSBuild), Makefiles, SCons, etc., including proprietary ones.
- It is specifically designed and optimized for accelerating the development and Continuous Integration of C and C++ projects.

▼ **Conan Installation**

- To install particular version of Conan, use the below step.

```
python3 -m pip install conan==*.*.*
```

Example: Below is the example for installing conan version 1.59.0

```
python3 -m pip install conan==1.59.0
```

To find the version of conan installed use the following command.

```
conan --version
```

▼ **Access to Artifactory repository**

- Users are allowed to upload the packages into Bangalore QA and Sofia Production Beta repository Instance.
- We request users to raise a VESS request to get access to conan package. You can find the details here

Steps to create a conan package

1. Configuration File Preparation

Copy the binaries into a particular location and run the below command to create a new conanfile.py file with required modules.

In this example, we have the binaries available under `/data1/Conanpackage/openssl/opensslbinaries` location. Click here to download the binaries

```
conan new <packagename>/<version>@<organisation_name>/<package_type> --bare
```

❗ **Note:** For the `package_type` we are allowing only “beta” or “stable” in the Production and Beta repository and “testing” in only Beta Repository. Other than these we are not allowing any other names in the Artifactory.

Example:

```
cd /data1/Conanpackage/openssl/opensslbinaries

conan new openssl/3.0.0beta1@visteon/beta --bare
```

- A new conanfile.py file will be created.In this file, update the required data for classes, variables and methods.

Class name and variables:

Class name should be unique for each package. For other variables, refer the below details.

- name - Name of the package (openssl)
- version - Version of the package (3.0.0beta1)
- platform - For which platform the package is created for.
- settings - OS and build types are to be updated. Supported OS and build types
  - OS = Linux/Windows
  - Build Types = Release/debug/relwithdebinfo/minsizerel

```
class opensslforConan(ConanFile):
    name = "packagename"
```

```
    version = "version"
    platform="Linux"
    settings = {"os": ["Linux"],"build_type": ["Release"]}
```

Binaries Location - package module

When user is packaging, User has to update the binaries location under `src=""`. So that, from that location, binaries are fetched for packaging.

```
def package(self):
        if self.settings.os=="Linux":
            self.copy("*", dst="", src="/data1/Conanpackage/openssl/opensslbinaries")
```

Build settings Method :

With self.info.include_build_settings(), Conan will generate different packages when you change the os_build

```
def package_id(self):
        self.info.include_build_settings()
```

Environment variables - Package info

- Variable that need to be add to the “PATH” should be add using `self.env_info.path.append()` function
- Custom environment variable can be added using `self.env_info`. Example : To add “OPENSSL_HOME” use “**self.env_info.OPENSSL_HOME = **”

```
def package_info(self):
        self.env_info.path.append(os.path.join(str(self.package_folder),"bin"))
        self.env_info.LD_LIBRARY_PATH.append(os.path.join(str(self.package_folder),"lib"))
        self.env_info.LD_LIBRARY_PATH.append(os.path.join(str(self.package_folder),"include"))
        self.env_info.path.append(str(self.package_folder))
        self.env_info.OPENSSL_HOME=str(self.package_folder)
```

APT dependency

To install the dependency of APT package use the below

```
from conan.tools.system.package_manager import Apt

def requirements(self):
    apt_pkgs = ["libncurses5"]
    installer = Apt(self)
    installer.install(apt_pkgs, check=True)
```

install(packages: Any, update: bool = False, check: bool = False, recommends: bool = False)

check=True will check whether the package is installed or not, This help in avoiding sudo prompt while the package already installed.

2. Create a conan Package

Create a package with the below command. You can change the OS and Build Type based on the packaging requirement.

```
conan export-pkg . <packagename>/<version>@<organisation_name>/<package_type> -s os="Linux" -s build_type="Release"
```

Example:

```
conan export-pkg . openssl/3.0.0beta1@visteon/beta -s os="Linux" -s build_type="Release"
```

conan export-pkg . -s os="Linux" -s build_type="Release" -s openssl/3.0.0beta1@visteon/beta

- Supported OS and build types
  - OS = Linux/Windows
  - Build Types = Release/debug/relwithdebinfo/minsizerel

You can find more details on packaging: here

3. Upload the package into Artifactory

To upload a conan package into a repository, It is required to add the repository details in the remotes.json file.

- You can check the Conan remotes by execute the below command. It lists the connection configured in the machine.

```
conan remote list
```

- If the required repository is not available, open the remotes.json file available under the given location and then added as given in example.

`remotes.json` files is available under `~/.conan/` or `%USERPROFILE%/.conan` or `$CONAN_USER_HOME/.conan/` in the user machine.

Repository entry should be as below in the remotes.json file.

```
{
   "name": "bangaloreqa",
   "url": " https://jfrog.bangalore.qa.visteon.com/artifactory/api/conan/conan ",
   "verify_ssl": false
},

{
   "name": "sofia",
   "url": "https://jfrog.sofia.visteon.com/artifactory/api/conan/devNext-conan-beta",
   "verify_ssl": false
  }
```

Now you can upload the Conan package into a repository in Artifactory using the below command.

```
conan upload -r <Remote_Name> <packagename>/<version>@<organisation_name>/<package_type> --all
```

Example

```
conan upload -r bangaloreqa openssl/3.0.0beta1@visteon/beta --all
```

4. Installing and testing the Conan package from artifactory

a. Installing the Package :

- To Install the Package in the user machine for testing it is required to clear the package data available in the machine to avoid overwrite issues.
- Execute the below command to check the package is existing in the machine or not.

```
conan inspect <packagename>/<version>@<organisation_name>/<package_type>
```

Example:

```
conan inspect openssl/3.0.0beta1@visteon/beta
```

- Execute the below command to remove if the conan package is already available. This command will remove the reference of the package in user machine.

```
conan remove <packagename>/<version>@<organisation_name>/<package_type>
```

Example:

```
conan remove openssl/3.0.0beta1@visteon/beta
```

- Create a file as `conanfile.txt` and add the below details to fetch the data into the user machine.

```
[requires]
openssl/3.0.0beta1@visteon/beta
```

```
[generators]
virtualenv
```

- Execute the below command: ****, this will install the package in the machine and package files will be available under `/.conan/data`.

```
conan install .
```

b. Testing the Package:

- To perform testing, Execute the below command in the command prompt from the location where conanfile.txt is created.

```
source ./activate.sh
```

❗ **Note:** Once the above command is executed, it will prompt for test commands. Input the test commands for the respective package accordingly.

```
**Example: **

- echo $OPENSSL_HOME   (For Environment Variable validation)
```

- Execute `source ./deactivate.sh` to exit.
- ❓ **Support:** If you've any Queries about conan packaging, please reach out to 📧 artifactory@visteon.com and for any Queries w.r.t devNext 📧 devNext.support@visteon.com