Debugging Windows with Remote Desktop

By default, you can connect CircleCI Windows machines only with SSH. However, in some cases, you may want a full Remote Desktop session so you can edit files, view the stack trace, etc. This wiki page tells you how to connect to these instances via remote desktop. Credit to @peterjc123 for these instructions.

1. Use port forwarding in SSH

   By default, CircleCI will provides us with a line of ssh command for remote debugging. e.g. `ssh -p 54782 35.237.241.189 -- cmd.exe`. So you may pass an another argument `-L [remote port]:localhost:[local port]` so that the command will look like `ssh -p 54782 35.237.241.189 -L 8000:localhost:8000 -- cmd.exe`. Please note that the local port and the remote port can be different.

2. Log on to the machine

   Type in the last command in your shell (e.g CMD, Powershell on Windows or Bash on Unix) and press enter.

3. Add a rule for local port forwarding on the remote VM

   Since it is impossible to listen to the RDP port 3389 directly, we need to set up a rule for port forwarding locally. That is `netsh interface portproxy add v4tov4 listenport=8000 connectaddress=127.0.0.1 connectport=3389`. Please note that the `listenport` here and the `[remote port]` in the previous command should be the same.

4. Change user password on the remote VM

   Actually, the RDP connection is ready in the previous step. However, you don't know the password so you cannot login. So you may just change the password, since you have admin rights. `net user circleci [new password]`. Please note that Windows Server requires a very complex password, which should include an uppercase letter, a digit and a special symbol and at least 12 letters long.

5. Test connection locally

   Open `mstsc` or any clients that supports RDP connection. Since we've setup port forwarding, connect to `localhost:[local port]` (if you were copy-pasting the instructions above, that is `localhost:8000`. Login with `circleci` and the password you set. And you should be all set.

What can you do with RDP?

1. Edit files

   Vim is already installed. You can open a file by right clicking on it and then open it with Vim. Alternatively, if you want to use Visual Studio Code, then

right click on Windows logo from the bottom-left corner of your desktop and select "Windows PowerShell (Admin)". In the newly opened windows, type in `choco install -y vscode`. After the installation completes, you can right click and open it with VSCode.

2. Restore build/test environment in PyTorch jobs

   We have written down script for your to restore the build/test environment when needed. Usually, there should be a shortcut called `Restore PyTorch Environment` on the Desktop. Double click on it to open a Command Prompt with admin rights. If the shortcut is not generated, you may activate the environment manually. Right click on Windows logo from the bottom-left corner of your desktop and select "Windows PowerShell (Admin)". In the newly opened windows, use `start cmd` to start a new Command Prompt window and then type in the following commands to restore the environment.

   ```
   # for build / test jobs
   call C:\Users\circleci\project\build\win_tmp\ci_scripts\pytorch_env_restore.bat
   ```

3. Trigger a rebuild in PyTorch jobs

   Based on the previous step, you can retry the build using any one set of the commands below.

   - ```
     set REBUILD=1
     sh .jenkins/pytorch/win-build.sh
     ```
   - ```
     python setup.py install --cmake
     ```

Debugging Windows with CDB

`cdb.exe` is a command-line debugger on Windows that is installed as a part of **Debugging Tools for Windows**. `cdb.exe` is quite handy for * getting stacktraces for crashes * looking at the assembly code `$pc`.

Unfortunately, given that our windows are running release builds of PyTorch, you won't be able to inspect C++ symbols and view the source code. Alternatively you could try rebuilding PyTorch (see **3. Trigger a rebuild in PyTorch jobs** above)

Installation:

Check if `windbg` is already installed in `C:\Program Files (x86)\Windows Kits\10\Debuggers\x64`. If you need to install it, run `choco install windbg`

Debugging:

1. Call `pytorch_env_restore.bat` to get a conda environment with PyTorch already installed

```
    # for build / test jobs
    call C:\Users\circleci\project\build\win_tmp\ci_scripts\pytorch_env_restore.bat
```

2. Change a directory into `project`

```
cd project
```

3. Run a crashing test with `cdb -o <program> <args>`

```
"C:\Program Files (x86)\Windows Kits\10\Debuggers\x64\cdb.exe" -o
python test/test_jit.py
```

4. Type `g` (the equivalent of `gdb`'s `r(un)`) and then `k` to display a stacktrace when PyTorch crashes

Most basic commands:

command	description
g	the same as gdb's run
k	show stacktrace
dv	displays a local variable if symbols are available

Links:

https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/debugging-using-cdb-and-ntsd