

---

# **ns-3 Waf build system**

ns-3 Annual Meeting  
June 2017

# Software introduction

---

- Download the latest release

- `wget http://www.nsnam.org/releases/ns-allinone-3.26.tar.bz2`
- `tar xjf ns-allinone-3.26.tar.bz2`

- Clone the latest development code

- `hg clone http://code.nsnam.org/ns-3-allinone`

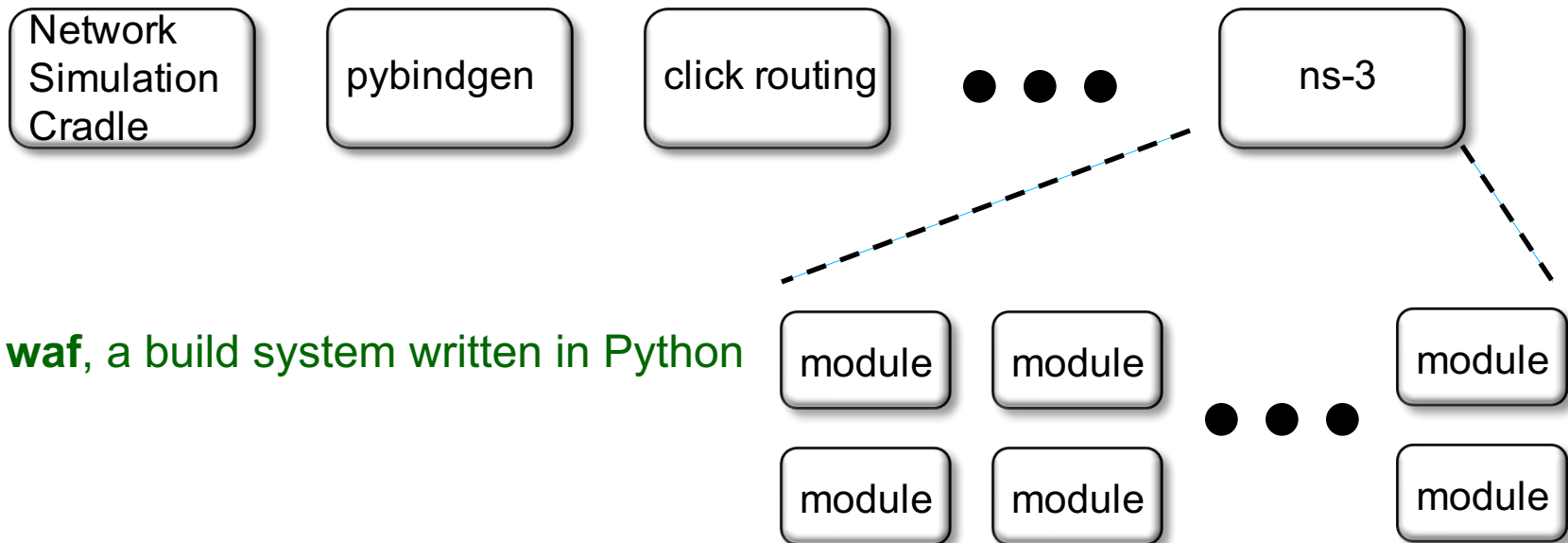
Q. What is "**hg clone**"?

A. Mercurial (<http://www.selenic.com>) is our source code control tool.

# Software building

- Two levels of ns-3 build

1) **bake** (a Python-based build system to control an ordered build of ns-3 and its libraries)



2) **waf**, a build system written in Python

3) **build.py** (a custom Python build script to control an ordered build of ns-3 and its libraries) <--- **may eventually be deprecated**

# ns-3 uses the 'waf' build system

---

- Waf is a Python-based framework for configuring, compiling and installing applications.
  - It is a replacement for other tools such as Autotools, Scons, CMake or Ant
  - <http://code.google.com/p/waf/>
- For those familiar with autotools:
  - `configure`       $\longrightarrow$  `./waf configure`
  - `make`             $\longrightarrow$  `./waf build`

# waf configuration

---

- Key waf configuration examples

```
./waf configure  
--enable-examples  
--enable-tests  
--disable-python  
--enable-modules
```

- Whenever build scripts change, need to reconfigure

Demo: `./waf --help`  
`./waf configure --enable-examples --enable-tests --enable-modules='core'`

Look at: `build/c4che/_cache.py`

# wscript example

---

```
## -*- Mode: python; py-indent-offset: 4; indent-tabs-mode: nil; coding: utf-8; -*-

def build(bld):
    obj = bld.create_ns3_module('csma', ['network', 'applications'])
    obj.source = [
        'model/backoff.cc',
        'model/csma-net-device.cc',
        'model/csma-channel.cc',
        'helper/csma-helper.cc',
    ]
    headers = bld.new_task_gen(features=['ns3header'])
    headers.module = 'csma'
    headers.source = [
        'model/backoff.h',
        'model/csma-net-device.h',
        'model/csma-channel.h',
        'helper/csma-helper.h',
    ]

    if bld.env['ENABLE_EXAMPLES']:
        bld.add_subdirs('examples')

    bld.ns3_python_bindings()
```

# waf build

---

- Once project is configured, can build via `./waf build` or `./waf`
- waf will build in parallel on multiple cores
- waf displays modules built at end of build

Demo: `./waf build`

Look at: `build/` libraries and executables

# Running programs

---

- `./waf shell` provides a special shell for running programs
  - Sets key environment variables

```
./waf --run sample-simulator
```

```
./waf --pyrun src/core/examples/sample-simulator.py
```



# Build variations

---

- Configuring a build type is done at waf configuration time
- debug build (default): all asserts and debugging code enabled

```
./waf -d debug configure
```

- optimized

```
./waf -d optimized configure
```

- static libraries

```
./waf --enable-static configure
```

# Controlling the modular build

---

- One way to disable modules:
  - `./waf configure --enable-modules='a','b','c'`
- The `.ns3rc` file (found in `utils/` directory) can be used to control the modules built
- Precedence in controlling build
  - 1) command line arguments
  - 2) `.ns3rc` in ns-3 top level directory
  - 3) `.ns3rc` in user's home directory

Demo how `.ns3rc` works

# Building without wscript

---

- The scratch/ directory can be used to build programs without wscripts

Demo how programs can be built without wscripts