MOCI Infrastructure Builds

Harry Shepherd

July 20, 2022

Contents

1 Introduction

Coupled production models contain several components, developed at several different institutions. It is important that these components can each be built in a consistant and repeatable way, for example knowing which repository revisions and compiler versions are used. A key part of this process is having a suite to build the libraries they depend on as an automated process, and to deploy those libraries as modules with a particular suite revision. This suite contains a series of scripts to perform those processes, and the scripts can be run as stand alone entities if required.

2 Suite Design

The suite is contained within the `infrastructure_suite` directory. All changes, be it to the build scripts or suite paramaters are carried out in this one location. The suite contains a full set of metadata. Paramaters can be changed by running the rose-edit command and different paramaters can be found in the following locations under **suite conf**

> **Top Level Control** Change the host, and also any preloaded modules before the build process. If different compilers were to be used these would be specified by the preloaded modules, and these are encapsulated

in the programming environment created. By checking the XIOS Only box, XIOS will be built without OASIS3-MCT, along with a corresponding programming environment module.

OASIS Extraction and build of OASIS3-MCT from the nitrox Git repository. Takes in a Git hash in OASIS Branch to allow for repeatability of builds.

XIOS XIOS branch/revision to allow for repeatability of builds.

Modules The modules to be created by the suite (Names, version numbers etc.)

Any compiler flags may be changed by editing the scripts (in the `bin/` directory) `oasis_build.sh`, `xios_build.sh`. These generate `make.inc` and `.arch files` respectively.

2.1 Overall Design

A list of directories and a summary of their contents

`app/` Rose applications to perform the tasks on various platforms.

`bin/` Executable shell scripts. These are described further below in section 2.2.

`file/` Met Office specific files for the Oasis3-MCT tutorial tests

`meta/` Metadata for the rose suite

`src/` Fortran 90 source files, MO specific for the OASIS3-MCT tests.

2.2 The Scripts

The scripts are found in the `bin/` directory.

`extract_xios.sh` Run on the HPC, extracts XIOS from the `fcm:xios.xm` repository path.

`extract_oasis.sh` Runs on Spice. Clones the OASIS3-MCT Git repository, then checks out a branch, either using a name or a hash. Transfers the files to the `$CYLC_SUITE_RUN_DIR/share` directory on the HPC. (This is already created by the `prepare_hpc_host` app defined in suite.rc.

`xios_build.sh` Runs on the HPC shared queue. Using cat, the arch files for XIOS are created, and then the code is compiled on the RAM-disk. To change compiler flags the definitions in this script can be edited directly.

`oasis_build.sh` Run on the HPC serial queue. Creates the make.inc file, then builds OASIS3-MCT using `make -f TopMakefileOasis3`. Then builds the tutorial using the models located in the `src/` directory.

`run_tutorial.sh` Run on HPC parallel queue. Runs the OASIS3-MCT tutorial tests

`create_oasis_module.sh`, `create_xios_module.sh`, `create_prgenv_module.sh`. Create the OASIS3-MCT, XIOS, and PrgEnv modules respectively. The module definitions are contained within the scripts themselves. Any specified environment variables will be automatically included. Includes OASIS3-MCT branch name/hash, or XIOS revision in the path. If the suite is run with `./launch_suite.sh` it will also include the suite revision.

3 Deployment of Modules

If the suite is run using the `rose suite-run` command, then the modules will be created in the `$CYLC_SUITE_RUN_DIR/share/modules` directory, with the corresponding packages in `$CYLC_SUITE_RUN_DIR/share/packages`. In this scenario the modules *will not* contain full revision numbers and are not suitiable for production work. To deploy the modules, a wrapper script `launch_suite.sh` is avaliable. This is run with two compulsory arguments, the HPC host on which the script is to be run, and the location to which the modules are to be deployed. For example:

```
$ ./launch_suite.sh xcslr0 /data/d00/moci/modules
```

will run the build on the XCS and will place the `modules` and `packages` directories in `/data/d00/moci/modules/`. The suite *will not* run using this method unless it is checked in, and there are no changes in the working copy. This is to ensure the versioning of the modules created is correct, and the modules can be rebuilt if required. This script overrides the enviroment variables `DEPLOYMENT_HOST` and `MODULE_BASE`, and passes in the suite url and suite revision as the environment variables `SUITE_URL` and `SUITE_REVISION` respectively.

There are three modules created, one for OASIS3-MCT, one for XIOS and one for the programming environments. They are versioned[1] and named using environment variables within the suite, which can be modified via `rose-edit`.

The OASIS3-MCT module:

```
$OASIS_MOD_NAME/$OASIS_MOD_VERSION/$SUITE_REVISION/$OASIS_BRANCH
```

The XIOS module:

```
$XIOS_MOD_NAME/$XIOS_MOD_VERSION/$SUITE_REVISION/$XIOS_REV
```

The Programming Environment module (which loads both above modules):

```
$GC_PRG_ENV_NAME/$GC_PRG_ENV_VERSION/$SUITE_REVISION/
```

[1]If the suite is not run using the launch script, then ‘undefined’ will appear instead of the suite revision

4 Building the documentation

The source for this documentation can be found in `doc/README.tex`. It can be compiled using LaTeXby running the script `doc/build_doc.sh`, which will then overwrite the pdf file in the top level directory.