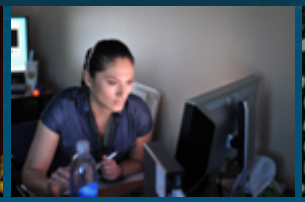
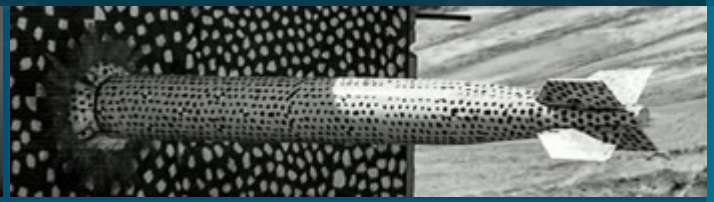
Sandia National Laboratories

SAND2021-6163C

LoadEnv

Consistently Loading Supported Environments Across Machines

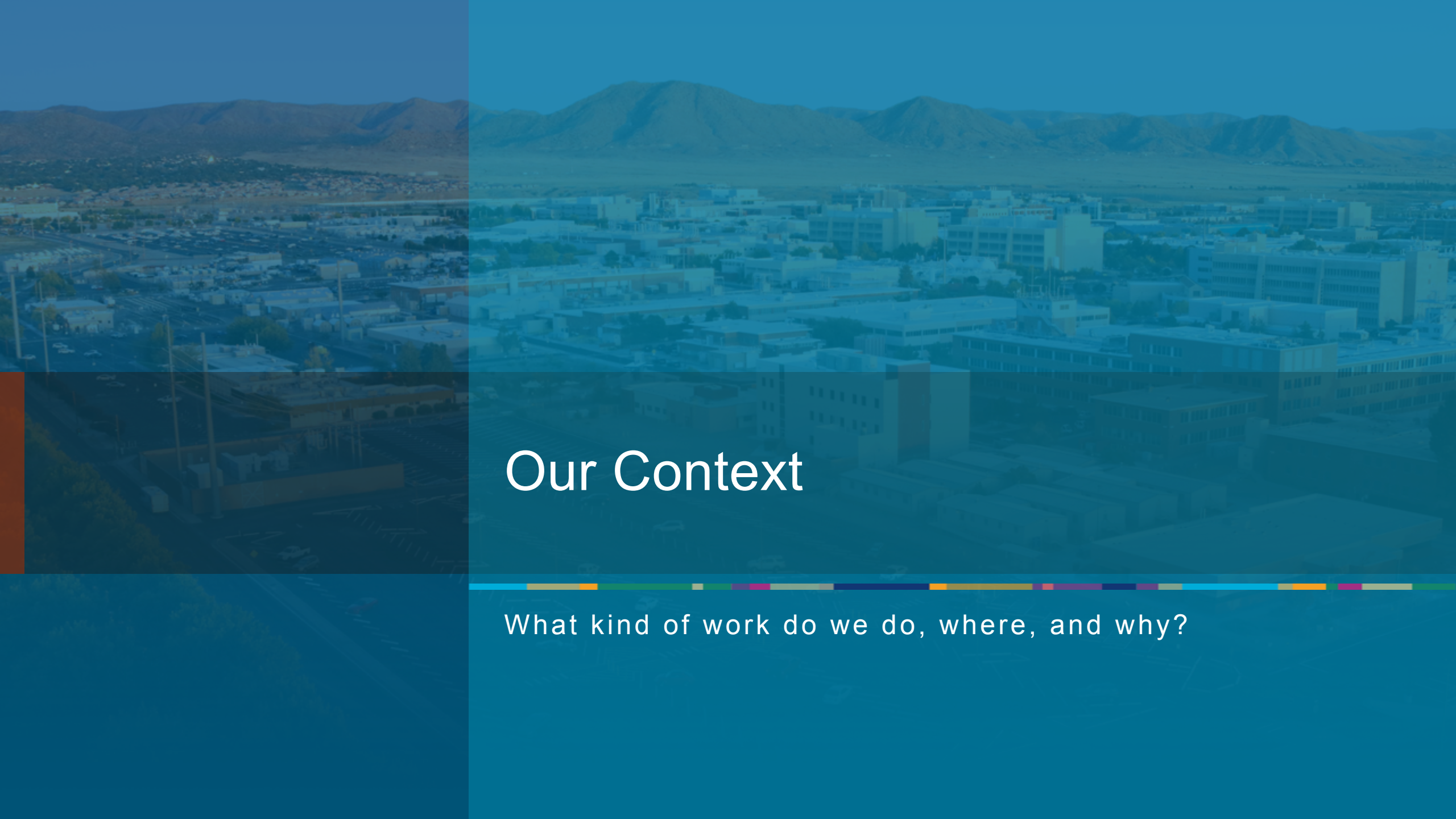




PRESENTED BY

Jason M. Gates, William Mclendon, Josh Braun, Evan Harvey

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.



Our Context

What kind of work do we do, where, and why?

Our Context

- Supporting ~50 compiler + TPL stacks
- Across ~10 machines (RHEL7, HPCs, next-gen testbeds)
- For codes with dozens of configuration knobs to turn
- Need a unified solution for users, developers, and CI services
- Developers:
  - Need to quickly switch between environments in the midst of developing/testing/debugging
  - Spack and containerization are not complete solutions here

Our Context: The Six Layers

What are they?

1. Operating System: base state of the machine
2. Programming Environment: compilers and TPLs
3. Loading the Environment: consistently loading particular environments
4. Configuring the Code: consistently setting configuration options
5. User/Application Build Scripts: a nice user interface to the layers below
6. Orchestration Services: running scripts on various platforms

Our Context: The Six Layers

What are we trying to do?

1. Operating System: assume this is set up
2. Programming Environment: assume we'll have access to all necessary modules
3. Loading the Environment: develop a general purpose tool; usable by any team building on top of environment modules; confi… each team
4. Configuring the Code: N/A
5. User/Application Build Scripts: N/A
6. Orchestration Services: N/A

Today's Talk

Improving Reproducibility

How do we ensure everyone does the same thing in the same way?

Improving Reproducibility

- One tool with one user interface used by everyone
- The user/developer/CI service:

```
$ source load-env.sh Build_develop_cuda_opt_Volta70_static_rdc
```

- The argument is just an example Jenkins job name
- It can be any string
- The only part that's relevant to loading the environment in this case is `cuda`
- The user is given the appropriate CUDA toolchain on the system they're on

Improving Reproducibility

```
[jmgate@machine-name]$ source load-env.sh --list-envs

Using system 'rhel7' based on matching hostname 'machine-name'.

+=============================================================================+
|   INFO:  Please select one of the following.
|
|   - Supported Environments for 'rhel7':
|     - clang-9.0.1-openmpi-4.0.3-serial
|       * Aliases:
|         - clang-9
|         - clang-9-serial
|     - cuda-10.1.243-gnu-7.2.0-openmpi-4.0.3
|       * Aliases:
|         - cuda
|         - cuda-10
|     - gnu-7.2.0-openmpi-4.0.3-serial
|       * Aliases:
|         - gnu
|         - gnu-serial
|     - intel-19.0.3-intelmpi-2018.4-serial
|       * Aliases:
|         - intel
|         - intel-19
|         - intel-19-serial
|     [abbreviated for the slide]
|
|   See /path/to/configuration/file for details.
+=============================================================================+
```

Improving Reproducibility

```
[jmgate@machine-name]$ source load-env.sh cuda

Using system 'rhel7' based on matching hostname 'machine-name'.
NOTICE:  Matched alias 'cuda' in build name 'cuda' to environment name 'cuda-10.1.243-gnu
-7.2.0-openmpi-4.0.3'.
Environment 'rhel7_cuda-10.1.243-gnu-7.2.0-openmpi-4.0.3' validated.
Environment loaded successfully.

[jmgate@machine-name]$ module list -t

Currently Loaded Modulefiles:
foo-env
foo-cmake/3.19.1
foo-ninja_fortran/1.10.0
bar-tools/python/3.7.9
bar-tools/aerotools/3
bar-tools/taos/2020.09.23
bar-cmake/3.19.4
bar-git/2.19.1
bar-dev/cuda-10.1.243_gcc-7.2.0_openmpi-4.0.3
```

Improving Reproducibility

```
[jmgate@machine-name]$ source load-env.sh clang-9 --output load_matching_env.sh

Using system 'rhel7' based on matching hostname 'machine-name'.
NOTICE:  Matched alias 'clang-9' in build name 'clang-9' to environment name 'clang-9.0.1-openmpi-
4.0.3-serial'.
Environment 'rhel7_clang-9.0.1-openmpi-4.0.3-serial' validated.
Environment loaded successfully.

[jmgate@machine-name]$ tail -14 load_matching_env.sh

module purge
module use /some/path
module load foo-env
module load foo-cmake/3.19.1
module load foo-ninja_fortran/1.10.0
module load bar-dev/clang-9.0.1_openmpi-4.0.3
envvar_op find_in_path MPICC mpicc              # These functions are defined further up
envvar_op find_in_path MPICXX mpicxx            # in load_matching_env.sh.
envvar_op find_in_path MPIF90 mpif90            #
envvar_op find_in_path OMPI_CXX clang++         #
envvar_op find_in_path OMPI_CC clang            #
envvar_op find_in_path OMPI_FC gfortran         #
```

Improving Reproducibility

- CI services can archive `load_matching_env.sh`
- Developers can exactly reproduce the environment
- One developer can hand another a `load_matching_env.sh` to debug something together
- A user can include `load_matching_env.sh` when filing a bug report
- All users/developers/CI services are doing the exact same thing

Improving Collaboration

How can teams communicate effectively about environments?

Improving Collaboration

- Historically trying to collaborate on environments has involved
  - Emailing scripts back and forth
  - “How did you set up your environment?”
  - “What’s the output of `env`?”
- Now everything needed to communicate within the team on environments is contained in three configuration files committed to the repository
  - Supported systems
  - Supported environments
  - Environment specifications
- Configuration files are clear and succinct, getting straight to the “what” and abstracting away the “how” of loading environments across systems

Improving Collaboration

```
# Supported systems configuration file
[system-name]
foo.*          # regular expressions
bar.*          # to match hostname
baz.*          # against
bif.*          #
```

```
# Supported environments configuration file
[system-name]
intel-18.0.2-openmpi-4.0.1-openmp  # env name
intel-18.0.2-openmpi-4.0.1-serial
intel-18.0.2-openmpi-4.0.3-openmp: # env name
    intel-18-openmp                # followed by
    intel-18                       # optional
    intel-openmp                   # aliases
    intel                          #
    default-env                    #
intel-18.0.2-openmpi-4.0.3-serial:
    intel-18-serial
    intel-serial
intel-19.0.4-openmpi-4.0.3-openmp:
    intel-19-openmp
    intel-19
intel-19.0.4-openmpi-4.0.3-serial:
    intel-19-serial
```

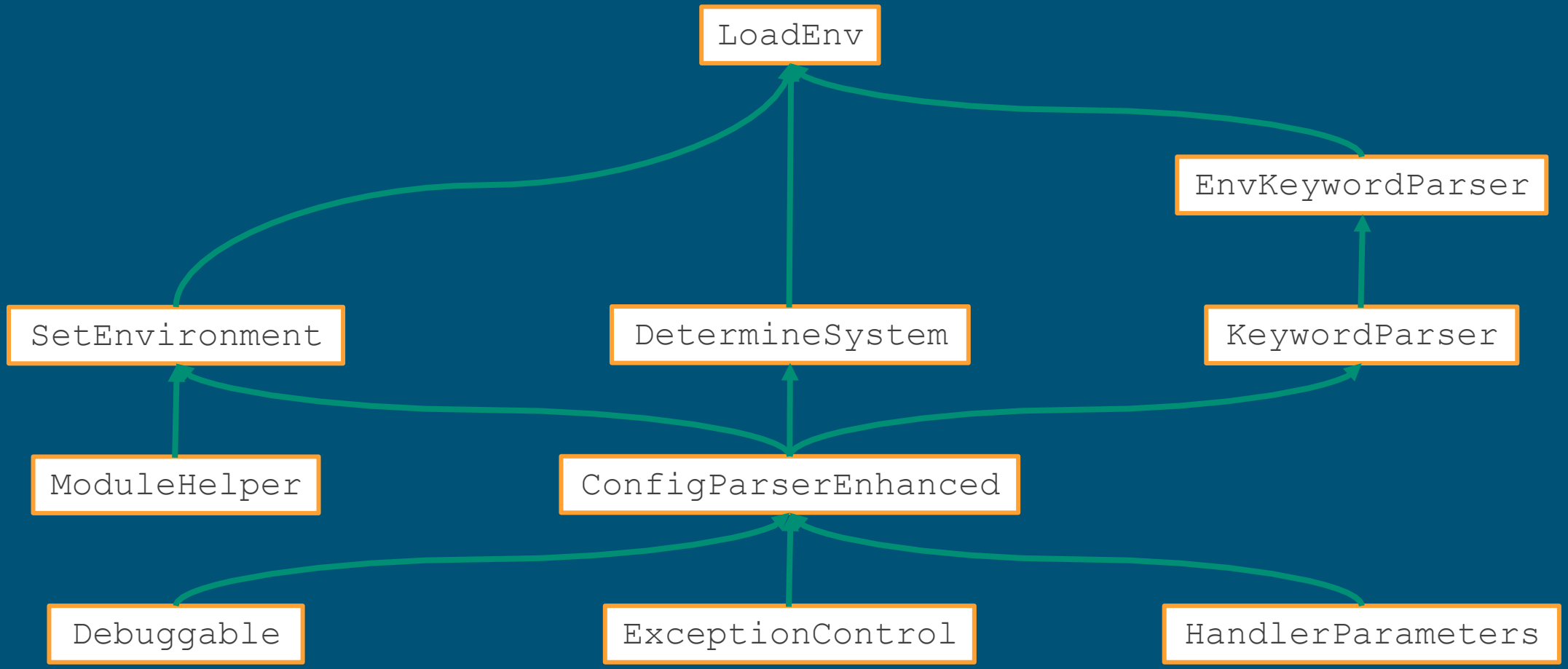
```
# Environment specifications configuration file
# Partial envs intended to be `use`d in others.
[SYSTEM-NAME_PRE]
use MODULE-PURGE
[SYSTEM-NAME_POST]
use MPI-COMPILER-VARS
use OMPI-GNU-VARS
envvar-unset OMPI_CXX
[SYSTEM-NAME_OPENMP]
envvar-set OMP_NUM_THREADS : 2
[SYSTEM-NAME_GNU-7.2.0-OPENMPI-2.1.2]
use SYSTEM-NAME_PRE
module-load devpack  :
20180521/openmpi/2.1.2/gcc/7.2.0/cuda/9.2.88
module-swap openblas : netlib/3.8.0/gcc/7.2.0
use SYSTEM-NAME_POST
envvar-find-in-path OMPI_CXX : g++
# Full environments intended to be loaded.
[system-name_gnu-7.2.0-openmpi-2.1.2-openmp]
use SYSTEM-NAME_GNU-7.2.0-OPENMPI-2.1.2
use SYSTEM-NAME_OPENMP
[system-name_gnu-7.2.0-openmpi-2.1.2-serial]
use SYSTEM-NAME_GNU-7.2.0-OPENMPI-2.1.2
```

Focusing on Long-Term Sustainability

How can ensure this solution is easy to maintain for years to come?

Focusing on Long-Term Sustainability

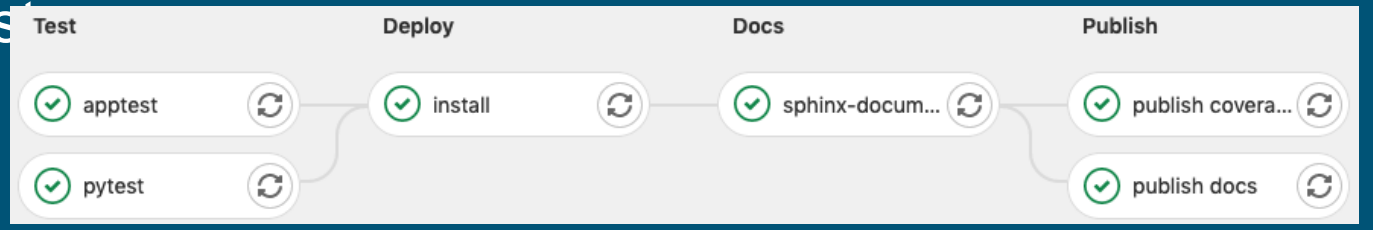
- Modular design + reusable components



Focusing on Long-Term Sustainability

- GitLab CI + pytest + Sphinx + GitLab Pages

  - All changes via merge reques
  - Test suite must pass
  - Coverage cannot degrade
  - Sphinx must build clean with warnings as errors



Focusing on Long-Term Sustainability

- Code vs documentation vs tests
  - `ConfigParserEnhanced`
    - Documentation-to-code ratio: 1.45
    - Tests-to-code ratio: 2.66

Piece	Executable Lines of Code	Lines of Comments
Library	779	1133
Unit Tests	2073	671

  - `SetEnvironment`
    - Documentation-to-code ratio: 1.66
    - Tests-to-code ratio: 2.84

Piece	Executable Lines of Code	Lines of Comments
Library	779	1133
Unit Tests	2073	671

Future Plans

What's on our horizon?

Future Plans

- Prepare `LoadEnv` for 1.0.0 release
- Finalize deployment plans
- Open-sourcing
- Develop tool for generating configuration options (Layer 4)