

# IES

Integrated Energy Systems

## NEAMS Workbench and IES

Summary

5 April 2023

Rob Lefebvre, Paul Talbot

# Purpose of the Work Package

- Context:
  - IES modeling and simulation tools are constantly evolving to support the demonstration of integrated technologies along every step of the technology maturation
  - The problem inputs and workflows are varied and can be complex
  - Modeling and simulation can help scale-up experimental systems, but only if the tools are usable
  - How do we increase usability and deployability?
- Key goals:
  - Improve problem input preparation, execution, and results visualization
  - Simplify problem input and workflow
  - Support an extended user base to increase the toolset's impact

# Methodology

- Leverage the **NEAMS Workbench** and its **integrated development environment (IDE)** to streamline user interactions and Framework for Optimization of ResourCes and Economics (FORCE) workflows
- Provide wizards and an active input assistant to accelerate input operations
  - Leverage DOE Nuclear Energy Advanced Modeling and Simulation (NEAMS) campaign development efforts to help automate the creation of more familiar input wizards
  - Use the NEAMS Workbench language processors to provide simplified, case-specific problem input for new users
- Enhance FORCE tools with advanced language capabilities
  - Enable FORCE to communicate, on demand, with the NEAMS Workbench (or other IDEs) information to assist user interaction
  - Important flexibility for increasing adoption of research software

# NEAMS Workbench Mission Statement

Provide a cross-platform graphical user interface (GUI) designed to facilitate problem creation, modification, navigation, validation, and visualization, as well as output and data file interaction as needed by new and experienced users.



Latest release available at <https://code.ornl.gov/neams-workbench/downloads>  
For assistance, email [nwb-help@ornl.gov](mailto:nwb-help@ornl.gov)

# NEAMS Workbench Input Editor

**Syntax Highlights**

```
199 # Define auxiliary kernels for each of the aux variables
200 [./fast_neutron_flux]
201   type = FastNeutronFluxAux
202   variable = fast_neutron_flux
203   block = clad
204   rod_ave_lin_pow = power_history
205   axial_power_profile = axial_peaking_factors
206   factor = 3e13
207   execute_on = timestep_begin
208 [../]
209
210 [./fast_neutron_fluence]
211   type = FastNeutronFluenceAux
212   variable = fast_neutron_fluence
213   block = clad
214   fast_neutron_flux = fast_neutron_flux
215   execute_on = timestep_begin
216 [../]
217
218 [./grain_radius]
219   type = GrainRadiusAux
220   block = pellet_type_1
221   variable = grain_radius
222   temp = temp
223   execute_on = linear
224
225 [./AuxKernels/GrainRadiusAux_type
226   active - If specified only the blocks named will be visited and made active
227   boundary - The list of boundary IDs from the mesh where this boundary conditio...
228   burnup - Burnup
229   control - Labels for accessing object parameters via contro...
230   error - Status of the MooseObject.
231   identifier - Identifiers matching these identifiers will be skipped.
232   model - The Grain Radius Model
233   is - This is a MooseObjectAction.
234   seed - The seed for the master random number generator
235   use_displaced_mesh - Whether or not this object should use the displaced mesh for comput...
236 [./stress_yy]
237   type = MaterialTensorAux
238   tensor = stress
239   variable = stress_yy
240   index = 1
241   execute_on = timestep_end
242 [../]
243 [./stress_zz]
244   type = MaterialTensorAux
245   tensor = stress
246   variable = stress_zz
247   index = 2
248   execute_on = timestep_end
249 [../]
```

**Context Aware Input Autocompletion**

line:182 column:18 - Validation Error: num\_radial value "-80" is less than the allowed minimum inclusive value of -70

line:183 column:17 - Validation Error: num\_axial value "-11" is less than the allowed minimum inclusive value of -10

line:268 column:16 - Validation Error: quantity value "wrong" is not one of the allowed values: [ ... "radial" "se

line:224, Col: 5 /AuxKernels/GrainRadiusAux\_type

**Input Block Highlights**

```
1 [GlobalParams]
2 # Set initial fuel density, other global parameters
3 density = 10431.0
4 disp_x = disp_x
5 disp_y = disp_y
6 disp_z = disp_z
7 order = SECOND
8 family = LAGRANGE
9 energy_per_fission = 3.2e-11 # J/fission
10 []
11
12 [Mesh]
13 file = smearedTest3.e
14 displacements = 'disp_x disp_y disp_z'
15 patch_size = 5 #40
16 patch_update_strategy = auto
17 partitioner = centroid
18 centroid_partitioner direction = v
```

**Customizable Application Run Configuration**

Mon Feb 19 07:18:21 2018

```
21 exactly when Open MPI kills them.
22 -----
23
24 Process finished with 0 return code; ran in 0 secs, finished at M
```

**Execution Messages**

Line: 1, Col: 2 /GlobalParams/decl

**Document Type and Quick Navigation**

```
1 [GlobalParams]
2 # Set initial fuel density, other global parameters
3 density = 10431.0
4 disp_x = disp_x
5 disp_y = disp_y
6 disp_z = disp_z
7 order = SECOND
8 family = LAGRANGE
9 energy_per_fission = 3.2e-11 # J/fission
10 []
11
12 [Mesh]
13 file = smearedTest3.e
14 displacements = 'disp_x disp_y disp_z'
15 patch_size = 5 #40
16 patch_update_strategy = auto
17 partitioner = centroid
18 centroid_partitioner direction = v
19 []
20
21 [Problem]
22 type = ReferenceReferenceProblem
23 solution_variables = 'disp_x disp_y disp_z temp'
24 reference_residual_variables = 'saved x saved y saved z saved t'
```

**Cursor Context**

Line: 15, Col: 16 /Mesh/patch\_size/value

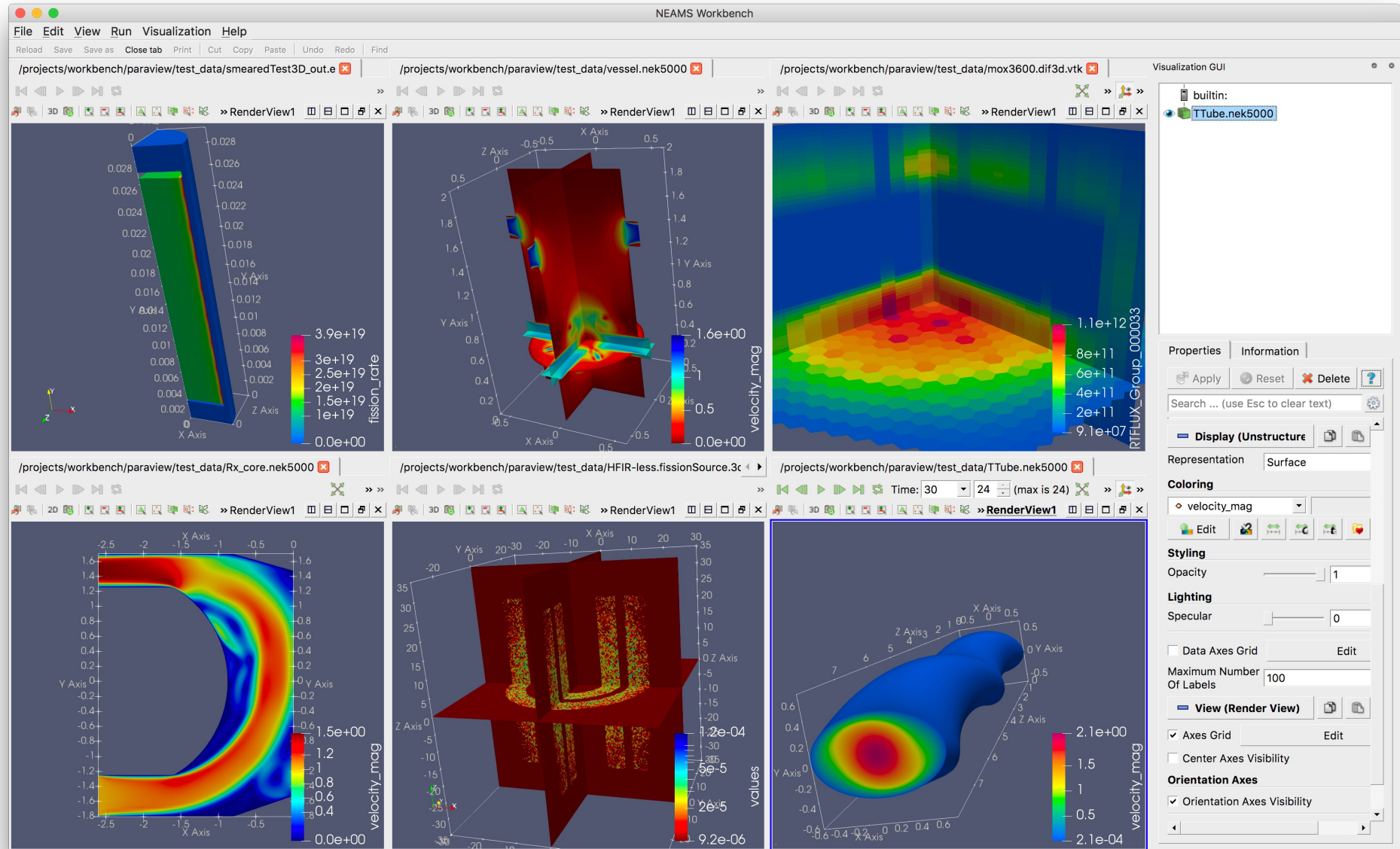
# NEAMS Workbench 2D Plots

- Extensible data processor
- Export plot to image and PDF (supports SVG)



# NEAMS Workbench ParaView Visualization

- Supports all ParaView 5.6 default data types
- ParaView is deployed within the NEAMS Workbench



# Workbench Analysis Sequence Processor (WASP)

- Provides input editor components to the NEAMS Workbench
- Includes reusable input processors accessible via C++ and Python API
  - Standard Object Notation (SON) used by PyARC and Nek5000 integration
  - Definition Driven Interpreter (DDI) used by Dakota and CTF integration
  - MOOSE Application Input Syntax (HIT)
  - VERA Input Interpreter (EDDI) used by VERA-CS and CTFFuel integration
  - Language Server (LS) and client interface used by MCNP ® integration
    - **Latest efforts involve incorporation of LS into the MOOSE framework to enable native diagnostics in Workbench**
- Utilities (input validation, retrieval, and template engines) and Python
- Open source at <https://code.ornl.gov/neams-workbench/wasp>

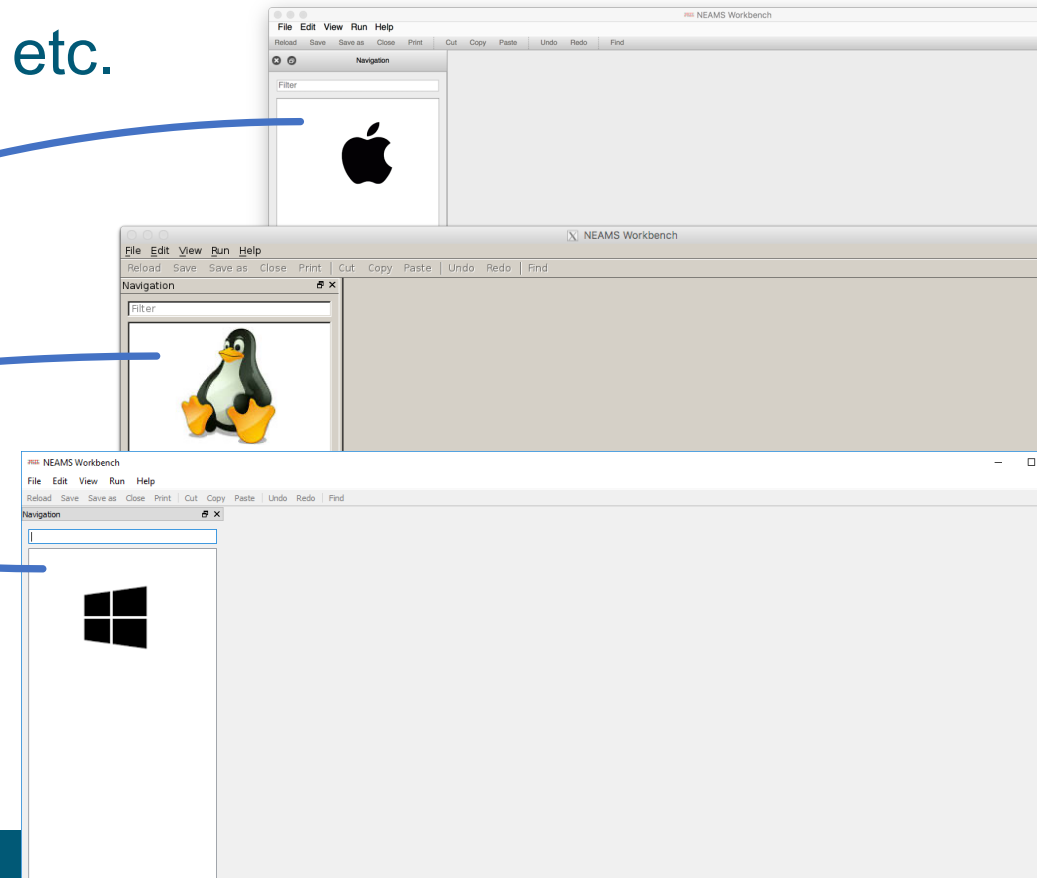
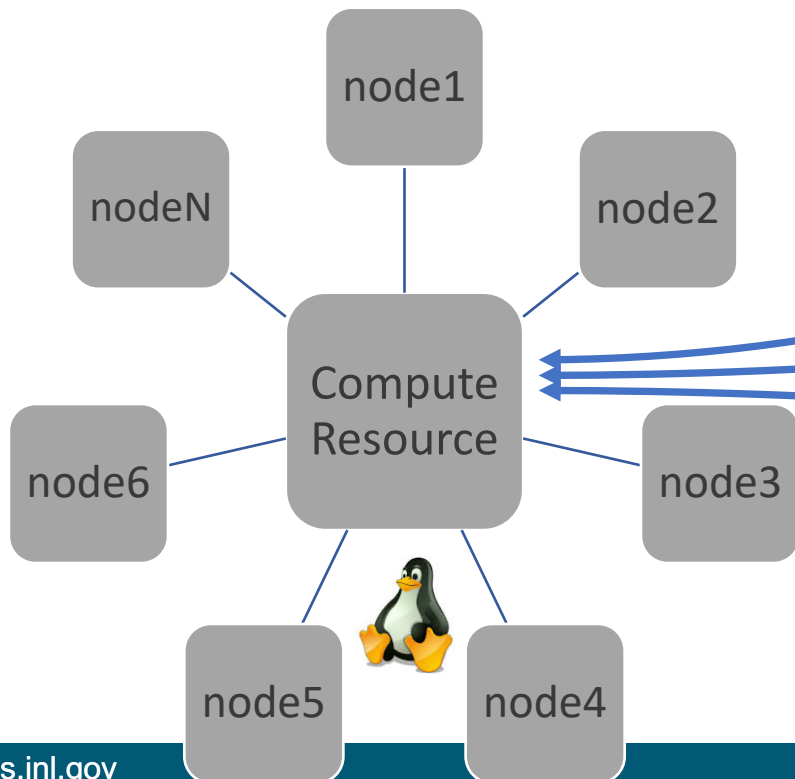


# NEAMS Workbench Application Runtime Layer

- All applications integrated into the NEAMS Workbench have a runtime layer written in Python (3)
  - A configuration-controlled *offline* Python environment is included with Workbench (Workbench/rte/entry.sh or Workbench/rte/entry.bat)
- Enables consistent job launch interactions between all integrated codes
  - Some applications lack any level of runtime (e.g., requiring users always name inputs 'input'), this layer normalizes application job launches with options available to users in Workbench
- Enables remote job launches across networks

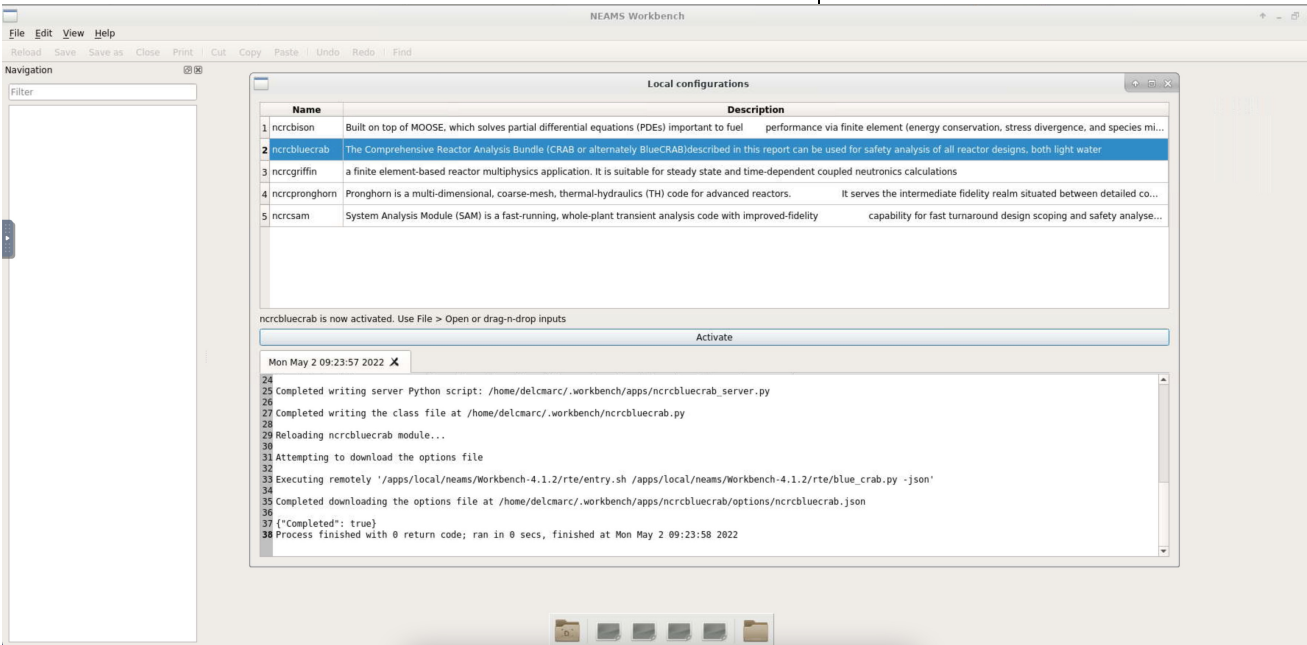
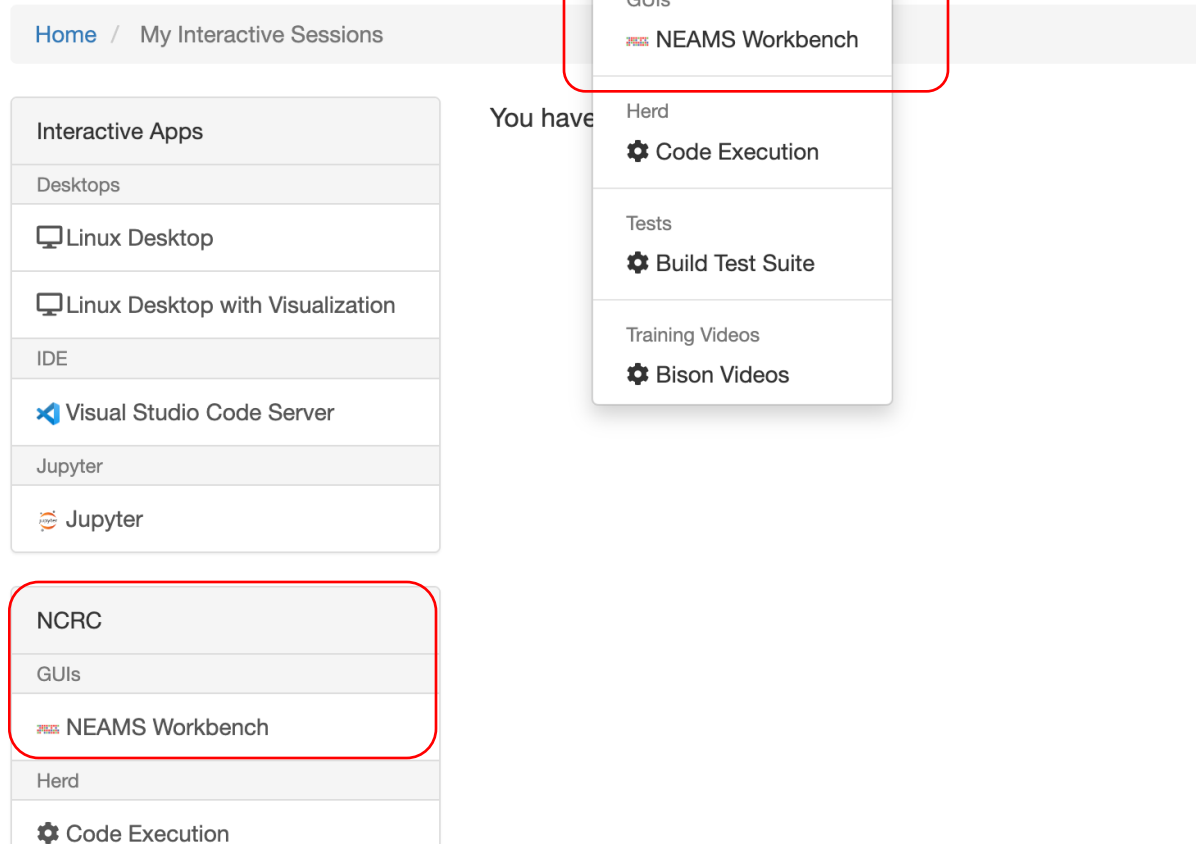
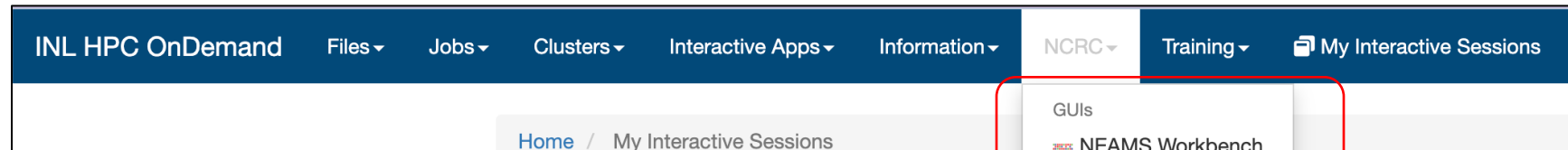
# Job Scheduling

- Under development
  - Support Windows, Mac, Linux, and **Open OnDemand** clients
  - Only supports Linux compute resources
- Support for popular schedulers
  - No-scheduler, PBS-based, IBM LSF, SLURM, etc.



# Open OnDemand and INL's NCRC Instance

- NEAMS Workbench is available on INL's Nuclear Computation Resource Center (NCRC)
- 3-click app activation



# Main Results and Desired Outcomes

- Accelerate user adoption and onboarding by easing problem definitions – let users focus on data not syntax
- Streamline, improve, and standardize presentation of results
- Improve workflow – especially for multi-level parallel analysis

# Conclusions and Next Steps

- Integration has just started
- Initial focus is on improving native capabilities toward a stable foundation in FORCE leading to enhanced usability, advanced areas of research, and successful adoption of the GUI
- Extent of programmatically available metadata can be improved to enable re-use (do not want to recreate software)
- Next steps:
  - Demonstrate lightweight user input syntax (more natural than XML)
  - Increase programmatic accessibility of Holistic Energy Resource Optimization Network (HERON)'s native input
  - Identify highest priority items to aid production analysis

# Preview | Lightweight UI

The screenshot displays the NEAMS Workbench interface with three main panels:

- Navigation Panel (Left):** Shows a tree view of the project structure. The 'Component - steamer' is selected under the 'heron\_input.xml.heron' component.
- Code Editor (Middle):** Displays the XML code for the 'steamer' component. The selected component is highlighted in yellow. The code includes parameters like 'lifetime', 'initial\_stored', and 'strategy'.
- Code Editor (Right):** Displays the XML code for the 'steam\_storage' component. The selected component is highlighted in yellow. The code includes parameters like 'lifetime', 'initial\_stored', and 'strategy'.

```
<Component name="steamer">
  <produces resource="steam" dispatch="fixed">
    <capacity resource="steam">
      <sweep_values>1, 100</sweep_values>
    </capacity>
  </produces>
  <economics>
    <lifetime>27</lifetime>
  </economics>
</Component>

<Component name="steam_storage">
  <stores resource="steam" dispatch="independent">
    <capacity resource="steam">
      <fixed_value>100</fixed_value>
    </capacity>
    <initial_stored>
      <fixed_value>1</fixed_value>
    </initial_stored>
    <strategy>
      <Function method="tiered">storage_control</Function>
    </strategy>
  </stores>
  <economics>
    <lifetime>10</lifetime>
  </economics>
</Component>
```

# Questions?

- Working lunch session will provide demonstration and hands on opportunity

