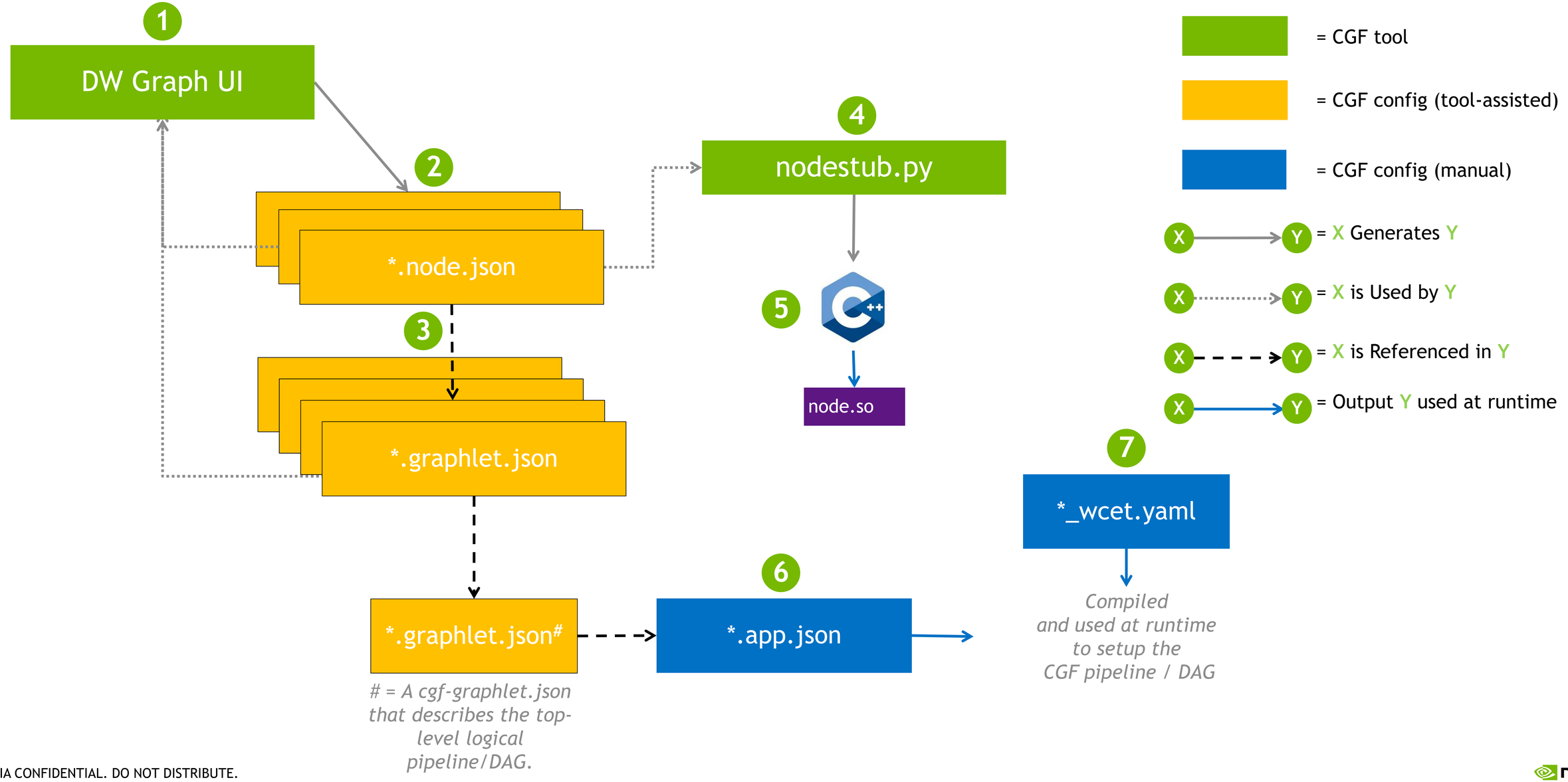
COMPUTE GRAPH FRAMEWORK DEMO

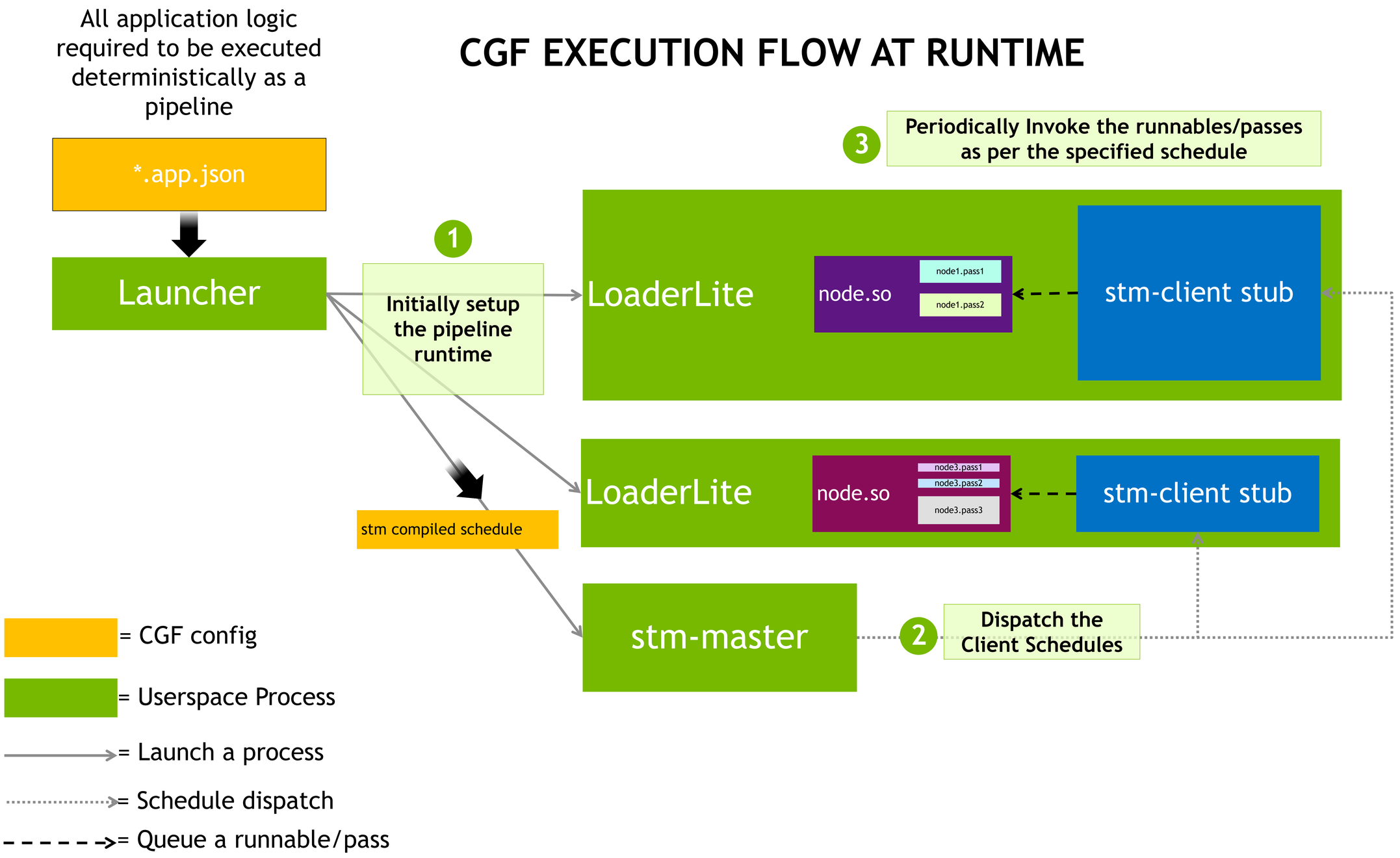
AGENDA

- CGF workflow at design and development time
- CGF execution flow at runtime
- What is a node
- DW Graph UI
- Nodestub - Autogenerating code for custom nodes
- Adding custom logic within autogenerated code
- Hello world Sample
- Custom node integration in CGF Demo

CGF WORKFLOW AT DESIGN / DEVELOPMENT TIME

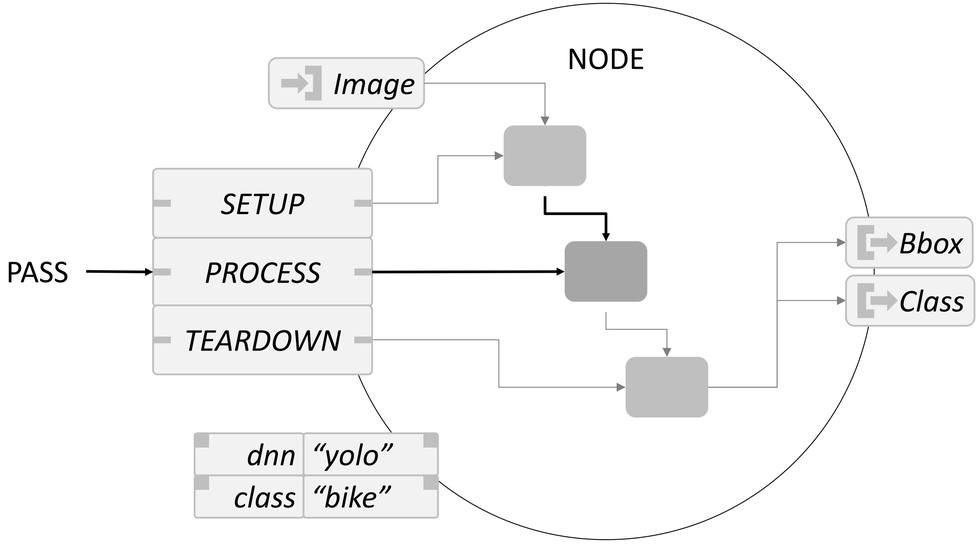


CGF EXECUTION FLOW AT RUNTIME



WHAT IS A CGF NODE

Key Components and Implementation



Node
Inheriting from base Node

Constructor receives parameters

Create function registers node

Ports
Inputs/Output ports specifying data type and unique port name

Passes
Passes to be executed in order, on specified compute engine

Parameters
Parameters for the constructor

Ports
Initialize ports, prepare output data containers

Passes
Register methods used for passes defined in public interface

Implementation of the passes

Public Node Interface MyNode.hpp

```
class MyNode : public dw::framework::ExceptionSafeProcessNode
{
   MyNode(const MyNodeParams& params, ...);

   dw::framework::create<Node>(ParameterProvider& provider);

   static constexpr auto describeInputPorts(){...};
   static constexpr auto describeOutputPorts(){...};

   static constexpr auto describePasses(){
      return dw::framework::describePassCollection(
         dw::framework::describePass("SETUP", DW_PROCESSOR_TYPE_CPU),...)};

   static constexpr auto describeParameters(){...};

}
```

Implementation MyNode.cpp

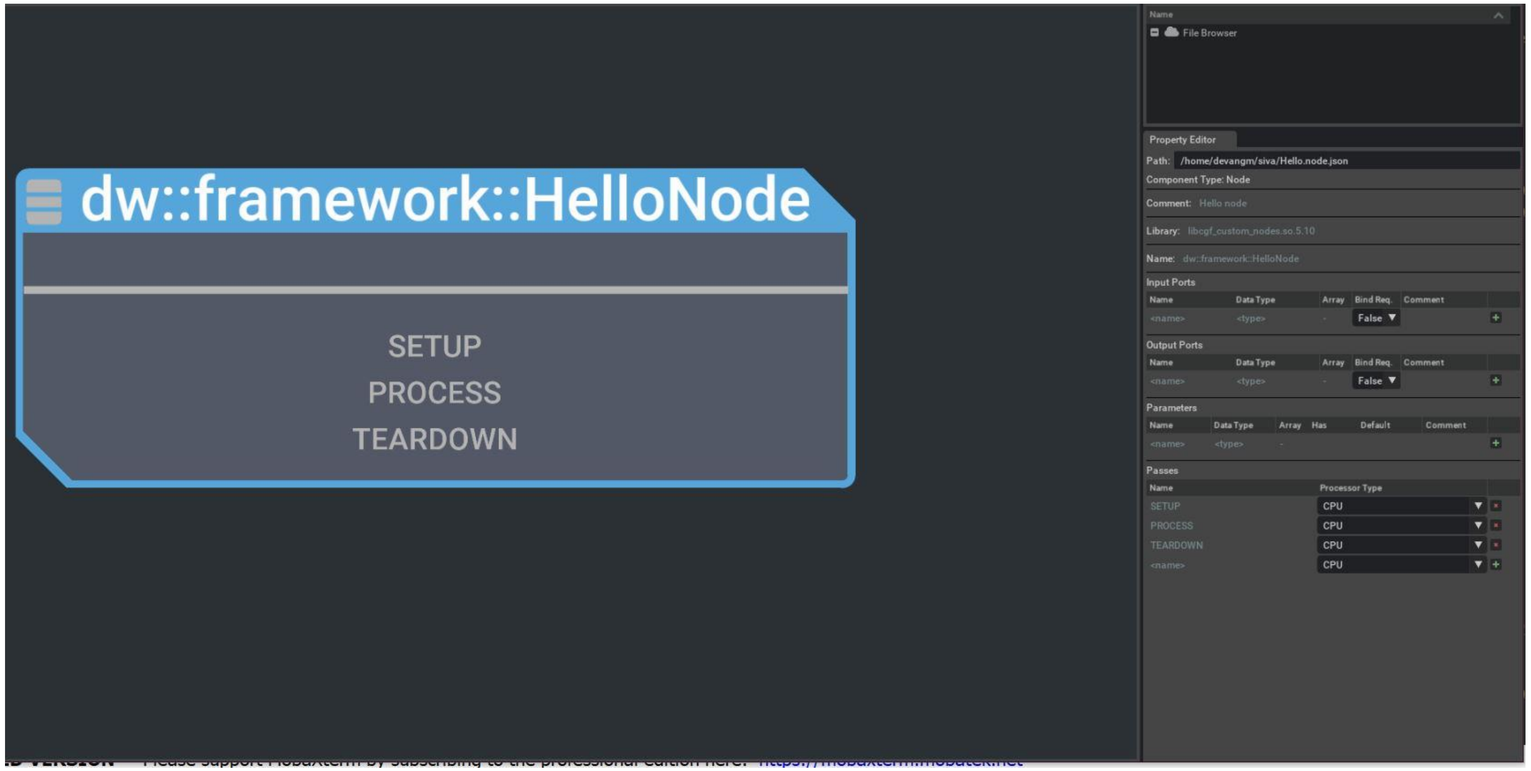
```
void MyNodeImpl:initPorts(){
   NODE_INIT_INPUT_PORT("IMAGE"_sv);
   ...
}

void MyNodeImpl::initPasses(){
   NODE_REGISTER_PASS("PROCESS"_sv, [this]() { return process(); });...}

dwStatus MyNodeImpl::process(){...}
```

DW GRAPH UI

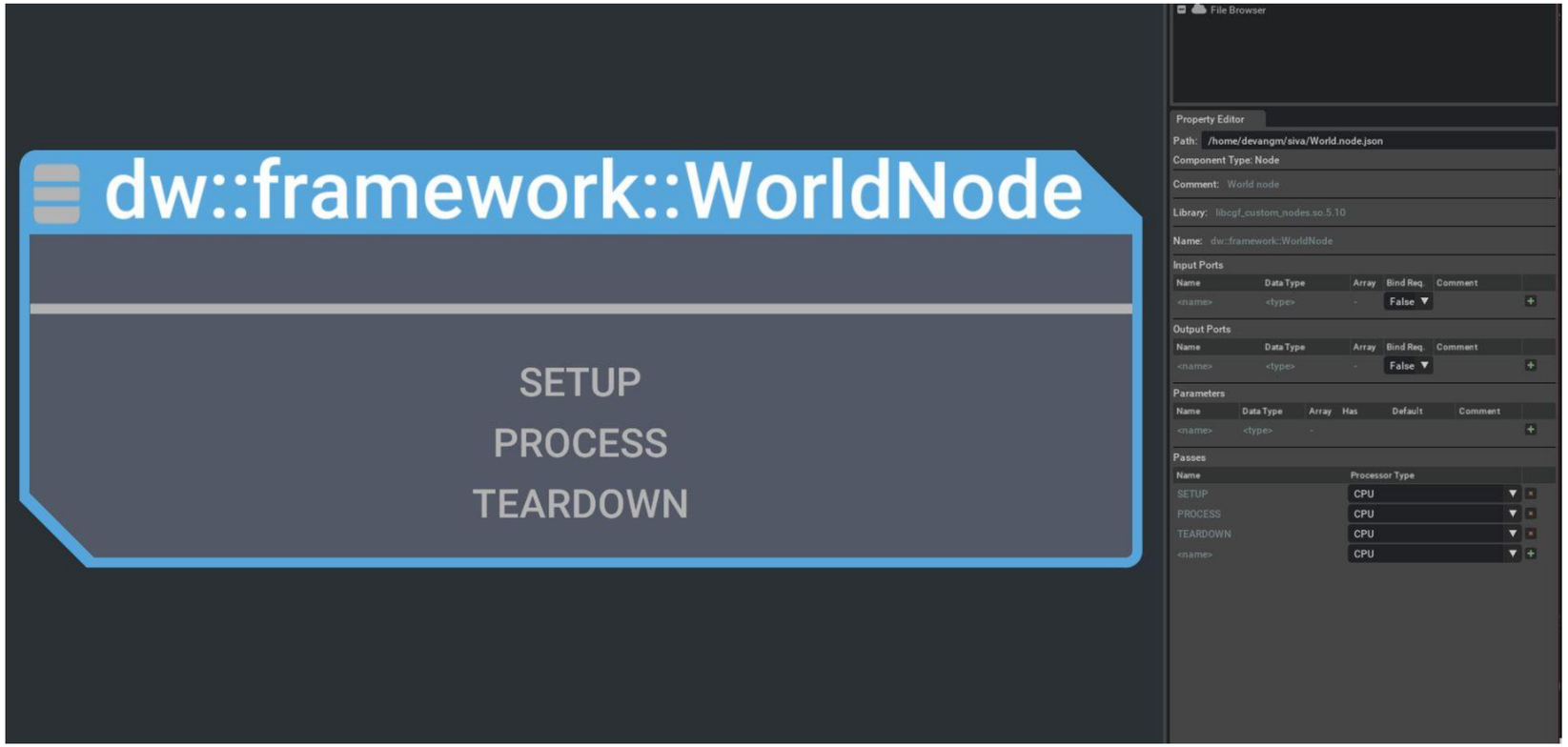
Create Hello Node




```
"comment": "Hello node",
"generated": false,
"library": "libcgf_custom_nodes.so.5.10",
"name": "dw::framework::HelloNode",
"inputPorts": {},
"outputPorts": {},
"parameters": {},
"passes": [
    {
        "name": "SETUP",
        "processorTypes": [
            "CPU"
        ]
    },
    {
        "name": "PROCESS",
        "processorTypes": [
            "CPU"
        ]
    },
    {
        "name": "TEARDOWN",
        "processorTypes": [
            "CPU"
        ]
    }
]
```

DW GRAPH UI

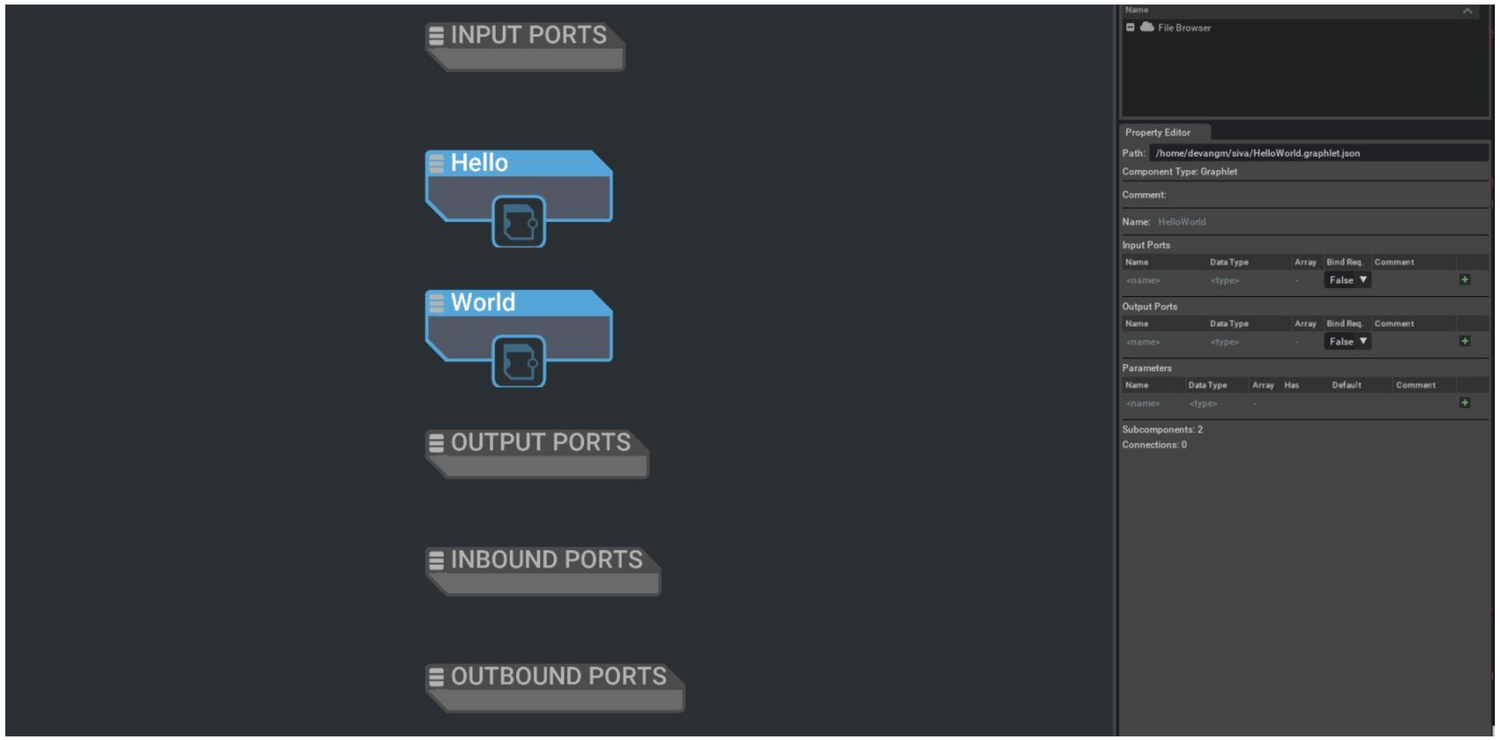
Create World Node




```
"comment": "World node",
"generated": false,
"library": "libcgf_custom_nodes.so.5.10",
"name": "dw::framework::WorldNode",
"inputPorts": {},
"outputPorts": {},
"parameters": { },
"passes": [
    {
        "name": "SETUP",
        "processorTypes": [
            "CPU"
        ]
    },
    {
        "name": "PROCESS",
        "processorTypes": [
            "CPU"
        ]
    },
    {
        "name": "TEARDOWN",
        "processorTypes": [
            "CPU"
        ]
    }
]
```

DW GRAPH UI

Create Hello World Graphlet




```
"name": "HelloWorld",
"parameters": {
},
"subcomponents": {
    "Hello": {
        "componentType": "./Hello.node.json",
        "parameters": {
                        }
    },
    "World": {
        "componentType": "./World.node.json",
        "parameters": {

        }
    }
},
"inputPorts": {},
"outputPorts": {},
"connections": []
```

NODESTUB - AUTOGENERATING CODE FOR CUSTOM NODES

```
tse@tse-ubuntu:/usr/local/driveworks-5.10/tools/nodestub$ ./nodestub.py --output-
path=/home/tse/siva/Helloworld/ /usr/local/driveworks/src/cgf/nodes/Hello.node.json
dw::framework::ExceptionSafeProcessNode --overwrite-existing-files
Generating files in: /home/tse/siva/Helloworld
* Node.thpp -> /home/tse/siva/Helloworld/HelloNode.hpp
Unknown data type 'int', additional #include directives might be needed in the node header
* Node.tcpp -> /home/tse/siva/Helloworld/HelloNode.cpp
* NodeImpl.thpp -> /home/tse/siva/Helloworld/HelloNodeImpl.hpp
* NodeImpl.tcpp -> /home/tse/siva/Helloworld/HelloNodeImpl.cpp
```

```
tse@tse-ubuntu:/usr/local/driveworks-5.10/tools/nodestub$ ./nodestub.py --output-
path=/home/tse/siva/Helloworld/ /usr/local/driveworks/src/cgf/nodes/World.node.json
dw::framework::ExceptionSafeProcessNode --overwrite-existing-files
Generating files in: /home/tse/siva/Helloworld
* Node.thpp -> /home/tse/siva/Helloworld/WorldNode.hpp
Unknown data type 'int', additional #include directives might be needed in the node header
* Node.tcpp -> /home/tse/siva/Helloworld/WorldNode.cpp
* NodeImpl.thpp -> /home/tse/siva/Helloworld/WorldNodeImpl.hpp
* NodeImpl.tcpp -> /home/tse/siva/Helloworld/WorldNodeImpl.cpp
```

ADDING CUSTOM LOGIC WITHIN AUTOGENERATED CODE

```
70
71 dwStatus WorldNodeImpl::processPass()
72 {
73     |
74     return DW_SUCCESS;
75 }
76
```

Add logic in the auto-generated stub code

- `HelloNodeImpl.cpp`
  - Added Print inside constructor, destructor, processPass()
  - Used DW_LOGD to route messages to process log.
- `WorldNodeImpl.cpp`
  - Added Print inside constructor, destructor, processPass()
  - Used DW_LOGD to route messages to process log.

GENERATE NODE SHARED LIBRARY AND STM BINARY

Steps to generate share library and stm binary for target

- Copy the node source files to */usr/local/driveworks/samples/src/cgf_nodes*
- Cross compile DW

```
cmake -B ~/DW-cross    -DCMAKE_TOOLCHAIN_FILE=/usr/local/driveworks/samples/cmake/Toolchain-V5L.cmake    -DVIBRANTE_PDK=/cmake
-B /home/tse/DW-cross    -DCMAKE_TOOLCHAIN_FILE=/usr/local/driveworks/samples/cmake/Toolchain-V5L.cmake    -
DVIBRANTE_PDK=/home/tse/nvidia/nvidia_sdk/DRIVE_OS_6.0.6_SDK_Linux_DRIVE_AGX_ORIN_DEVKITS/DRIVEOS/drive-linux -S
/usr/local/driveworks/samples
home/tse/nvidia/nvidia_sdk/DRIVE_OS_6.0.6_SDK_Linux_DRIVE_AGX_ORIN_DEVKITS/DRIVEOS/drive-linux -S /usr/local/driveworks/samples
```

- Generate STM binary

```
cd /usr/local/driveworks/tools/descriptionScheduleYamlGenerator
./descriptionScheduleYamlGenerator.py --app ../../src/cgf/graphs/descriptions/systems/HelloWorld.app.json --output
~/HelloWorld__standardSchedule.yaml
./stmcompiler -i ~/HelloWorld__standardSchedule.yaml -o ~/HelloWorld__standardSchedule.stm
```

HELLO WORLD SAMPLE

Steps to run Hello world sample on target

- Copy below files to target
  - Node Json files at /usr/local/driveworks/src/cgf/nodes
  - Graphlet Json file at /usr/local/driveworks/src/cgf/graphs/descriptions/graphlets
  - App Json file at /usr/local/driveworks/src/cgf/graphs/descriptions/systems
- Copy custom node shared library
  - Shared library *libcgf_custom_nodes.so* at */usr/local/driveworks/lib/libcgf_custom_nodes.so.5.10*
- Copy STM binary
  - Stm files at /usr/local/driveworks/src/cgf/graphs
- Launch CGF APP

```
sudo /usr/local/driveworks-5.10/bin/launcher --binPath=/usr/local/driveworks-5.10/bin \
--spec=/usr/local/driveworks-5.10/src/cgf/graphs/descriptions/systems/HelloWorld.app.json \
 --logPath=/home/nvidia/siva/CGF-Launch/LogFolder  --path=/usr/local/driveworks-5.10/bin \
  --datapath=/usr/local/driveworks-5.10/data/samples/cgf/trafficlightturning-hyperion8 \
  --dwdatapath=/usr/local/driveworks-5.10/data --vdcpath=/usr/local/driveworks-5.10/bin \
  --schedule=/usr/local/driveworks-5.10/src/cgf/graphs/HelloWorld__standardSchedule.stm \
  --start_timestamp=0 --mapPath=maps/sample/sanjose_loop --loglevel=DW_LOG_VERBOSE \
  --fullscreen=0 --winSizeW=1920 --winSizeH=1200 --virtual=1 --disableStmControlLogger=1 --gdb_debug=0 \
  --app_parameter= > /home/nvidia/siva/CGF-Launch/LogFolder/launcher.log 2>&1
```

HELLO WORLD SAMPLE

Helloworld_process0.log

```
[TopExecutor.hpp:2652][TopExecutor] helloworld_process0: SchedulerClient
[TopExecutor.hpp:2941][TopExecutor] Register runnable helloworld_process0_ssm_pass_0
[TopExecutor.hpp:3147][TopExecutor] Register runnable helloworld_World_pass_0
[TopExecutor.hpp:3147][TopExecutor] Register runnable helloworld_World_pass_1
[TopExecutor.hpp:3147][TopExecutor] Register runnable helloworld_World_pass_2
[TopExecutor.hpp:3147][TopExecutor] Register runnable helloworld_Hello_pass_0
[TopExecutor.hpp:3147][TopExecutor] Register runnable helloworld_Hello_pass_1
[TopExecutor.hpp:3147][TopExecutor] Register runnable helloworld_Hello_pass_2
```

```
[DEBUG][tid:71][HelloNodeImpl.cpp:74][HelloNode] HelloNode: processPass executed
[DEBUG][tid:71][WorldNodeImpl.cpp:74][WorldNode] WorldNode: processPass executed
[DEBUG][tid:71][HelloNodeImpl.cpp:74][HelloNode] HelloNode: processPass executed
[DEBUG][tid:71][WorldNodeImpl.cpp:74][WorldNode] WorldNode: processPass executed
[DEBUG][tid:71][HelloNodeImpl.cpp:74][HelloNode] HelloNode: processPass executed
```

CUSTOM NODE INTEGRATION IN CGF DEMO

HelloWorld Node and SumNode integration

- **Host and Target**
  - Copy *HelloWorld.json* and *SumNode.json* at `/usr/local/driveworks/src/cgf/nodes`
  - Update HelloWorld and Sum nodes details in *CGFDemo.graphlet.json*
    - Under the top level key subcomponents add below snippet

```
"helloworld": {
    "componentType": "../../../nodes/HelloWorldNode.node.json"
    "parameters": { "name": "$name" }
},
"sum": {
    "componentType": "../../../nodes/SumNode.node.json"
}
```

    - Under the top level key parameters, add HelloWorld name parameter

```
"name": { "type": "dw::core::FixedString<64>", "default": "Demo" }
```

    - Under the top level key connections:

```
{ "src": "helloworld.VALUE_0", "dests": {"sum.VALUE_0": {}} },
{ "src": "helloworld.VALUE_1", "dests": {"sum.VALUE_1": {}} },
```

CUSTOM NODE INTEGRATION IN CGF DEMO

HelloWorld Node and SumNode integration

- **Host and target**
  - In CGFDemo.app.json, add HelloWorld and Sum nodes in these sections:
    - Under both, the standardSchedule and slowSchedule schedule, add demo nodes to the renderEpoch passes:

```
"renderEpoch": {
    "passes": [
        "cgfDemo.arender",
        "cgfDemo.helloworld",
        "cgfDemo.sum"
    ]
}
```

    - Under the camera_pipeline0 process, add demo nodes to the list of subcomponents:

```
"subcomponents": [
    "cgfDemo.cameraPipelineFront0",
    "cgfDemo.arender",
    "cgfDemo.helloworld",
    "cgfDemo.sum"
],
```

CUSTOM NODE INTEGRATION IN CGF DEMO

HelloWorld Node and SumNode integration

- **Host**
  - Set up DW cross compile environment on host and cross compile the samples.
  - libcgf_custom_nodes.so will then be generated under <build_dir>/src/cgf_node directory
  - Create STM binary

```
./descriptionScheduleYamlGenerator.py --app CGFDemo.app.json --output CGFDemo__standardSchedule.yaml
CGFDemo__slowSchedule.yaml
./stmcompiler -i CGFDemo__standardSchedule.yaml -o CGFDemo__standardSchedule.stm
./stmcompiler -i CGFDemo__slowSchedule.yaml -o CGFDemo__slowSchedule.stm
```

  - grep "helloworld" CGFDemo__standardSchedule.stm -grep "helloworld" CGFDemo__slowSchedule.stm

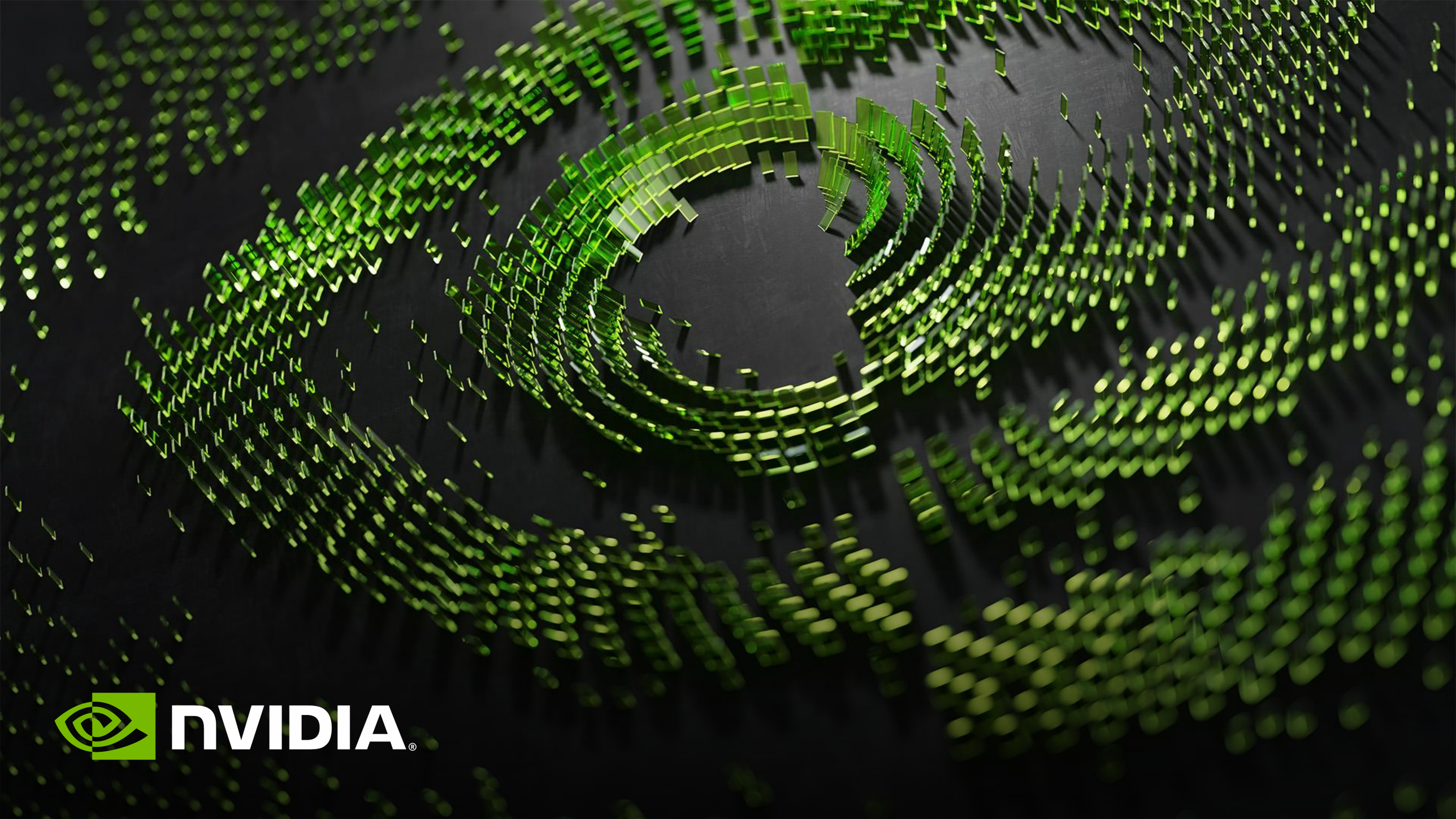
CUSTOM NODE INTEGRATION IN CGF DEMO

HelloWorld Node and SumNode integration

- **Target**
  - Copy cross compiled libcgf_custom_nodes.so to /usr/local/driveworks/lib/
  - Replace with new CGFDemo__standardSchedule.stm and CGFDemo__slowSchedule.stm in /usr/local/driveworks/src/cgf/graphs folder
  - Run CGF Demo with new custom nodes

```
cd /usr/local/driveworks/bin/config
sudo run_cgf.sh
```

  - Check Helloworld and Sum Node prints in camera_pipeline0.0.log


NVIDIA