H31W

Open Source Tools for Configuring the Next Generation Water Resources Modeling Framework

Austin Raney[1,2], Nels Frazier[1,2], Keith Jennings[1,2], Luciana L. Kindl da Cunha[2,3], Ahmad Jan[1,2]

[1]Lynker Technologies, Boulder, CO, United States, [2]NOAA Office of Water Prediction, National Water Center, Tuscaloosa, AL, United States
[3]WEST Consultants, Inc, Folsom, CA, United States

OWP | OFFICE OF WATER PREDICTION

Problem & Solution

A Next Generation Water Resources Modeling Framework (NextGen) CONUS simulation requires **880,000+** configuration files (1 file per Basic Model Interface (BMI) model per *catchment*). It is infeasible to manually create the configuration files necessary for such large simulations. This work presents **tools that make it possible to generate NextGen configuration files at any scale**, including CONUS.

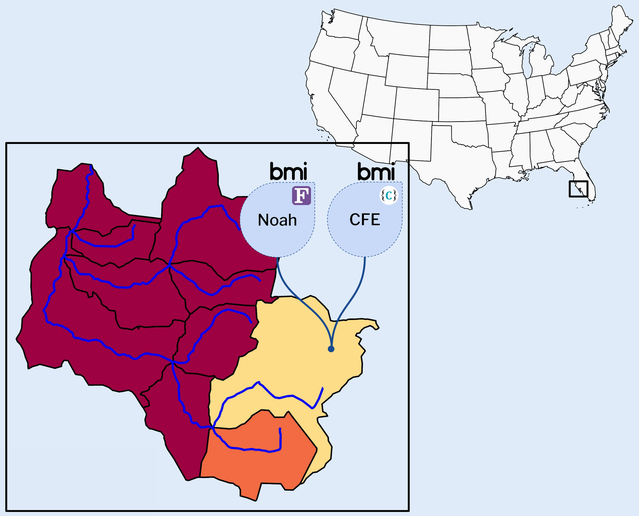
- **Automate** the generation of NextGen configuration files at any scale
- The tools are fast, **flexible**, and designed to fit into your workflow
- Generate configuration files for **any combination** of OWP maintained BMI models or **your own model**!

Background

NextGen is a flexible software for conducting simulations over a set of *catchments* each consisting of a single or composite multi-model *formulation*. Streamflow outputs from each *catchment* are routed using a modular, multi-scheme, tree based routing framework, T-Route, over a standardized topology of rivers, lakes, and catchments, known as a *hydrofabric*. NextGen inspects, sets, and executes modules using the BMI. Each module that runs at the catchment scale is allowed a *configuration file*. However, **with this flexibility comes complexity in the form of configuration**. As the number of modeled catchments or BMI models increases, so does the number of configuration files.

Automatically Generate 880,000+ NextGen Configuration Files

- Tools are capable of generating model configuration files for **any combination** of OWP maintained BMI models
- The ~**16 lines of code** (*shown right*), generate **1.7+ million configuration files** required to run a NextGen *formulation* consisting of the Noah land surface model coupled with the conceptual equivalent (CFE) NWM over
- Data used to create configuation files is primarily sources from *hydrofabric** (1) and *hydrofabric linked data products** (2)

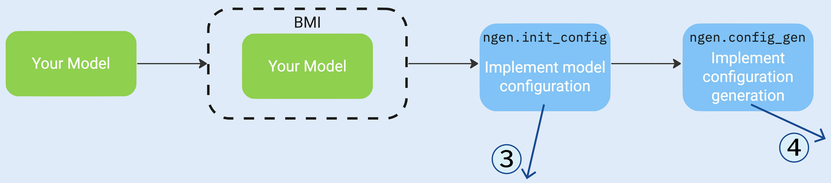



```python
import geopandas as gpd
import pandas as pd

from ngen.config_gen.hook_providers import DefaultHookProvider
from ngen.config_gen.file_writer import DefaultFileWriter
from ngen.config_gen.generate import generate_configs

from ngen.config_gen.models import Cfe, NoahOWP

hf_uri = "https://lynker-spatial.s3.amazonaws.com/v20.1/conus.gpkg"                    (1)
hf_lnk_data_uri = "https://lynker-spatial.s3.amazonaws.com/v20.1/conus_net.parquet"    (2)

hf = gpd.read_file(hf_uri, layer="divides")
hf_lnk_data = pd.read_parquet(hf_lnk_data_uri)

hook_provider = DefaultHookProvider(hf=hf, hf_lnk_data=hf_lnk_data)
file_writer = DefaultFileWriter("./config/")

generate_configs(
    hook_providers=hook_provider,
    hook_objects=[Cfe, NoahOWP],   (5)
    file_writer=file_writer,
)
```

** See H33C-07 Wednesday afternoon for an update on hydrofabrics products*

Bring Your Model into the NextGen Ecosystem

New **models are easily added and integrated into existing workflows** and effectively **join the NextGen ecosystem**



After wrapping *Your Model* (e.g. LSTM) in BMI, generate its configuration files by:

- Defining the fields and data types of LSTM's configuration file (3)
- Specifying the data sources to build an LSTM configuration file (4)
- Adding LSTMHooks to the list of models in hook_objects at (5)

```python
from ngen.init_config import serializer_deserializer as serde
from pathlib import Path

class LSTM(serde.YamlSerializerDeserializer):
    train_cfg_file: Path
    verbose: bool = False
    lat: float
    lon: float
    slope_mean: float

config = LSTM.from_yaml("config.yaml")
config.to_yaml("another_config.yaml")
```

```python
from lstm import LSTM

class LSTMHooks:
    def __init__(self):
        self.data = {}

    def hydrofabric_linked_data_hook(
        self, version: str, divide_id: str, data: Dict[str, Any]
    ) -> None:
        self.data["lat"] = data["Y"]
        self.data["lon"] = data["X"]
        self.data["slope_mean"] = data["slope"]

    def visit(self, hook_provider: "HookProvider") -> None:
        hook_provider.provide_hydrofabric_linked_data(self)
        self.data["train_cfg_file"] = pathlib.Path("weights.pt")
        self.data["verbose"] = True

    def build(self) -> pydantic.BaseModel:
        return LSTM(**self.data)
```

Code & Examples on GitHub

ACKNOWLEDGEMENTS:

lynker-spatial.com
lynker-spatial.com/copyright.html

REFERENCES:

Johnson, J. M. (2022). National Hydrologic Geospatial Fabric (hydrofabric) for the Next Generation (NextGen) Hydrologic Modeling Framework, HydroShare http://www.hydroshare.org/resource/129787b468aa4d55ace7b124ed27dbde

CONTACT

Website: *https://water.noaa.gov*
Email: *nws.nwc@noaa.gov*

View my poster and other AGU materials