# CESM
# Engineering Innovation on EPIC

## June 14, 2023

# Agenda

- Partners/EPIC Overview
- Community Infrastructure
    a. Repeatable to SRW, RRFS, LandDA, HAFS, and future applications
- CI/CD
    a. Complete for SRW
    b. Repeatable for LandDA, HAFS, and future applications
- Tutorials and training
- Need for Testing

# Partners

# Community Collaborators/Partners

**Acknowledgement**

- NOAA OAR: WPO, GSL, PSL, NSSL, CSL, AOML, GFDL
- NOAA Open Data Dissemination (NODD) Program
- NWS: EMC, OSTI
- DTC
- UCAR: CGD, JCSDA
- Academia: George Mason University, Oklahoma University, University of Michigan
- CSPs: AWS, Azure, and Google Cloud
- Cooperative Institutes: CIRES, CIMSS

**DRAFT & PREDICISIONAL**

# EPIC Overview

# PARTNERING WITH THE COMMUNITY FOR THE BENEFIT OF THE NATION

## Vision

To enable the most accurate and reliable operational numerical forecast model in the world.

## Mission

To be the catalyst for community research and modeling system advances that continually inform and accelerate advances in our nation's operational forecast modeling systems.



The Art of the Possible
Community Modeling, Earth Prediction and Innovation

Students    Government    Academia    Industry

# Simplifying NOAA's Operational Forecast Suite

## Transitioning 21 of NOAA's Operational Forecast Systems into Eight Applications

### 21 Systems in NOAA's Forecast Suite

Global Weather, Waves & Global Analysis - GFS/ GDAS
Global Weather and Wave Ensembles, Aerosols - GEFS
Short-Range Regional Ensembles - SREF
Global Ocean & Sea-Ice - RTOFS
Global Ocean Analysis - GODAS
Seasonal Climate - CDAS/ CFS
Regional Hurricane 1 - HWRF
Regional Hurricane 2 - HMON
Regional HiRes CAM 1 - HiRes Window
Regional HiRes CAM 2 - NAM nests/ Fire Wx
Regional HiRes CAM 3 - RAPv5/ HRRR
Regional HiRes CAM Ensemble - HREF
Regional Mesoscale Weather - NAM
Regional Air Quality - AQM
Regional Surface Weather Analysis - RTMA/ URMA
Atmospheric Transport & Dispersion - HySPLIT
Coastal & Regional Waves - NWPS
Great Lakes - GLWU
Regional Hydrology - NWM
Space Weather 1 - WAM/IPE
Space Weather 2 - ENLIL

*NOAA's weather to climate prediction capability developed over the last two decades*

### Transition Over Time

Private

COMMUNITY MODELING
*CO-DEVELOPMENT*

NOAA

Academia
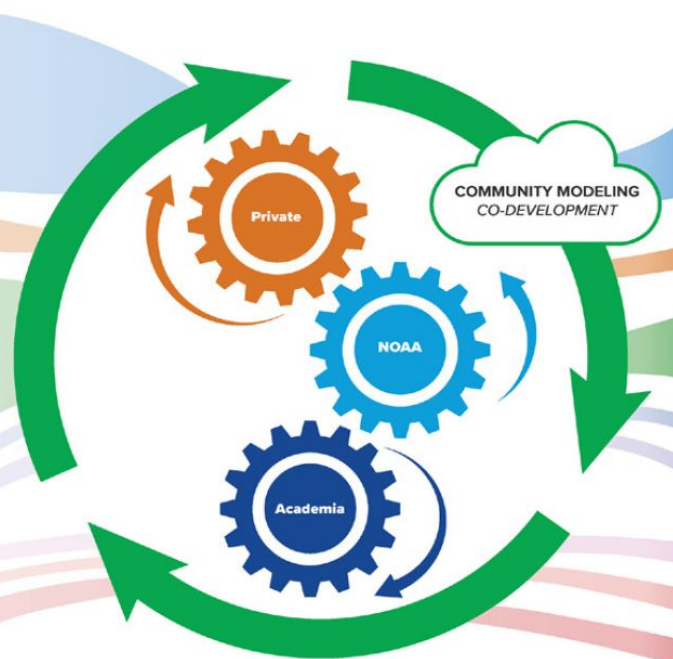
### UFS Applications

**Medium Range & Subseasonal**

**Marine & Cryosphere**

**Seasonal**

**Hurricane**

**Short-Range Regional HiRes CAM & Regional Air Quality**

**Air Quality & Dispersion**

**Coastal**

**Lakes**

**Hydrology**

**Space Weather**

*Uccellini et al. (BAMS, 2022)*

# EPIC

EARTH PREDICTION INNOVATION CENTER

**Building Open and Dynamic Collaboration within the Earth Sciences Community**
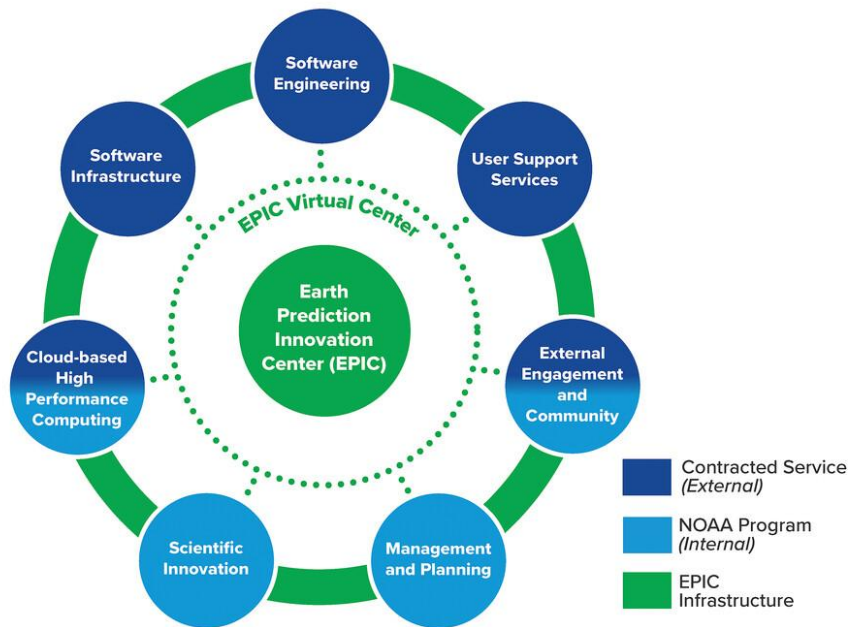
Future Plans:
- CI/CD
- Repeatable Application runs
- Fail Quickly
- Enhanced testing frameworks
- Advanced User Support
- Configuration Management
- Cloud configuration scripts
- Community Tools
- Unified Workflow
- Community Events

**DRAFT & PREDICISIONAL**

# Community Infrastructure

# EPIC Cloud Architecture

| | |
|---|---|
| **AWS RTX Sandbox** | An AWS account that sits outside of the NOAA firewall to allow for non-CAC users. |
| **AWS ACIO Sandbox** | An AWS account that sits outside of the NOAA firewall to allow for non-CAC users. Login.gov access. |
| **Azure RTX and ACIO Sandbox** | An Azure account that sits outside of the NOAA firewall to allow for non-CAC users. |
| **GCP ACIO Sandbox** | A GCP account that sits outside of the NOAA firewall to allow for non-CAC users. |
| **AWS ACIO Dev** | An AWS account that sits inside the NOAA firewall containing our application code that is in active development under ACIO. |
| **AWS ACIO Prod** | An AWS account that sits inside the NOAA firewall containing our application code that is live in the production account under ACIO. |
| **Parallel Works** | A third-party HPC provider inside of the NOAA firewall that provides us virtual machines to test UFS applications on all 3 cloud service providers. |

aws

Azure

Google Cloud

# New Tutorial - Infrastructure

- Common Infrastructure using Packer - able to deploy on any CSP
- https://github.com/NOAA-EPIC/packer-srwcluster
    - 12 lines of code to build out and run SRW
    - Tutorials - Earth Prediction Innovation Center (noaa.gov)


- Video 2: Creating a Head Node in AWS
    - Can be any of the CSP's


- Video 3: Running any application
    - Starting with SRW v2.1
    - Next: LandDA



HashiCorp
Packer

# CI/CD Pipeline

# Pipeline Gates

- Average Build time
- Average time per gate
- Average build time per platform
- Code Coverage
- Forecast Skill

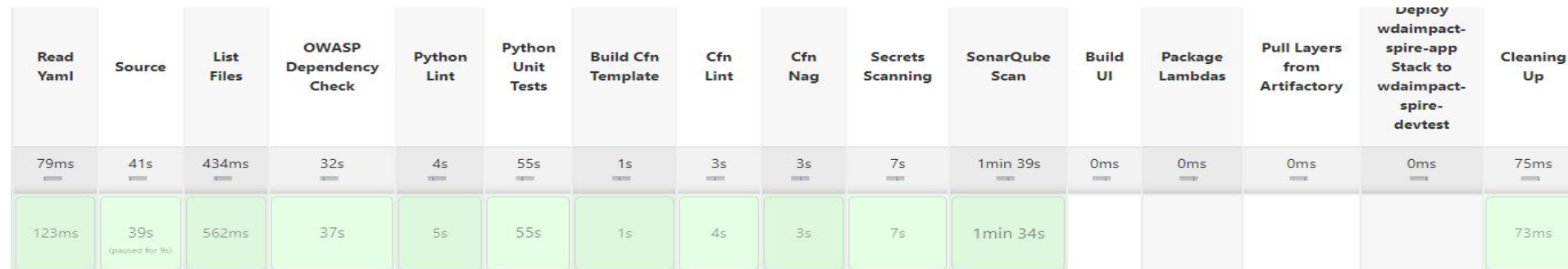| | |
|---|---|
| Checkout Source Code | Pull source code from GitHub and stage the data for analysis before deploying the code. |
| Unit Testing | Run available unit tests for projects and ensure that the tests run as expected. Collect code coverage metrics for the available baselines. |
| Lint (Flake 8) | Perform static code analysis that enforces style consistencies across programming languages. |
| Dependency Check | Scan third-party libraries and modules for current vulnerabilities. |
| Build the Cloud Stack | Terraform/Cloudformation scripts will create a repeatable process for deploying applications. |
| Lint Cloud Stack | Examine the cloud stack template and return various suggestions. |
| Nag Cloud Stack | Pinpoint security vulnerabilities in cloud stack templates. |
| Scan Secrets | Scan for any improper use of security passwords or credentials. |
| Static Code Analysis | Scan code in all programming languages using SonarQube to determine current vulnerabilities, maintenance issues, and defects. Note: SonarQube also has the ability to utilize architectural metrics such as cyclomatic complexity and maintainability metrics. Cyclomatic complexity as the example infers is a value that tells the ability that a new engineer will be able to come in and maintain the baseline. If the number is high, then you have an application that is tough to upkeep, so tracking this number over time will make sure that your application is easy to maintain, which in turn reduces technical debt costs. |
| Package/Pull Artifacts/Deploy | This gate sequence will package up the artifacts and the application and deploy the application as needed after completing all quality gate checks. |
| Run Regression Tests | Run a list of regression tests to test the overall end-to-end functionality. |

# CI/CD Pipeline

- Master Pipeline:

**Stage View**

| | Build and Test | Matrix - SRW_PLATFORM = 'cheyenne', SRW_COMPILER = 'gnu' | Matrix - SRW_PLATFORM = 'cheyenne', SRW_COMPILER = 'intel' | Matrix - SRW_PLATFORM = 'gaea', SRW_COMPILER = 'intel' | Matrix - SRW_PLATFORM = 'hera', SRW_COMPILER = 'intel' | Matrix - SRW_PLATFORM = 'jet', SRW_COMPILER = 'intel' | Matrix - SRW_PLATFORM = 'orion', SRW_COMPILER = 'intel' | Initialize | Initialize | Initialize | Initialize | Initialize | Initialize | Build | Build | Build | Build | Build | Build | Test | Test | Test | Test | Test | Test |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Average stage times: (Average full run time: ~26min 23s) | 1s | 2s | 2s | 2s | 2s | 2s | 2s | 3min 6s | 0ms | 0ms | 0ms | 0ms | 0ms | 12min 38s | 0ms | 0ms | 0ms | 0ms | 0ms | 1s | 0ms | 0ms | 0ms | 0ms | 0ms |
| #40 Sep 25 09:34  1 commit | 1s | 2s | 2s | 2s | 2s | 2s | 2s | 1min 47s | 1min 36s | 5min 9s | 4min 19s | 2min 47s | 3min 2s | 15min 35s | 9min 14s | 23min 26s | 8min 57s | 8min 18s | 10min 20s | 1s | 1s | 1s | 1s | 1s | 2s |

- No-Integration Pipeline:

| Read Yaml | Source | List Files | OWASP Dependency Check | Python Lint | Python Unit Tests | Build Cfn Template | Cfn Lint | Cfn Nag | Secrets Scanning | SonarQube Scan | Build UI | Package Lambdas | Pull Layers from Artifactory | Deploy wdaimpact-spire-app Stack to wdaimpact-spire-devtest | Cleaning Up |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 79ms | 41s | 434ms | 32s | 4s | 55s | 1s | 3s | 3s | 7s | 1min 39s | 0ms | 0ms | 0ms | 0ms | 75ms |
| 123ms | 39s (paused for 9s) | 562ms | 37s | 5s | 55s | 1s | 4s | 3s | 7s | 1min 34s | | | | | 73ms |

14

# EPIC Dashboard - Selenium

**DRAFT & PREDICISIONAL**

# EPIC Dashboard - GitHub

**DRAFT & PREDICISIONAL**

# EPIC Dashboard - GitHub Traffic



## UFS_UTILS

UFS_UTILS GitHub repository

Data from: 2023-05-21 to 2023-06-11

| Name | Email | Commits |
|---|---|---|
| GeorgeGayno-NOAA | 52789452+GeorgeGayno-NOAA@users.noreply.github.com | 1 |

## ufs-weather-model

ufs-weather-model GitHub repository

Data from: 2023-05-21 to 2023-06-11

| Name | Email | Commits |
|---|---|---|
| Sadegh Sadeghi Tabas | 31417680+SadeghTabas-NOAA@users.noreply.github.com | 1 |
| jiandewang | jiande.wang@noaa.gov | 1 |
| Gillian Petro | 96886803+gspetro-NOAA@users.noreply.github.com | 1 |
| Dustin Swales | dustin.swales@noaa.gov | 1 |
| dkokron | dkokron@users.noreply.github.com | 1 |

Rows per page: 100 ▾    1–6 of 6    ‹  ›

**DRAFT & PREDICISIONAL**

# EPIC Dashboard - Pipeline

**EPIC CI Build Status – ufs-srweather-app**

**Last updated: Sun Mar 12 22:42:01 PDT 2023**

| ufs-srweather-app/job/pipeline/view/change-requests | | | | | | |
|---|---|---|---|---|---|---|
| timestamp | PR-build | inProgress | duration (min) | result | WE2E-tests | S3-artifacts |
| 2023-03-10 15:29:36 | ufs-srweather-app/job/pipeline/job/PR-667/1/ | true | | | | ▼ |
| 2023-03-10 17:13:46 | ufs-srweather-app/job/pipeline/job/PR-663/1/ | false | 309.8 | FAILURE | cheyenne-intel gaea-intel jet-intel orion-intel | srw_build-cheyenne-gnu.log ▼ |
| 2023-03-08 17:06:26 | ufs-srweather-app/job/pipeline/job/PR-657/3/ | false | 518.3 | FAILURE | | srw_build-cheyenne-gnu.log ▼ |
| 2023-03-08 17:00:05 | ufs-srweather-app/job/pipeline/job/PR-657/2/ | false | 1 | FAILURE | | ▼ |
| 2023-03-08 16:53:00 | ufs-srweather-app/job/pipeline/job/PR-657/1/ | false | 0 | FAILURE | | ▼ |
| 2023-03-10 15:08:26 | ufs-srweather-app/job/pipeline/job/PR-656/1/ | false | 274.9 | SUCCESS | cheyenne-gnu cheyenne-intel gaea-intel jet-intel orion-intel | srw_build-cheyenne-gnu.log ▼ |
| 2023-03-08 19:16:35 | ufs-srweather-app/job/pipeline/job/PR-650/1/ | false | 438.7 | FAILURE | cheyenne-gnu cheyenne-intel gaea-intel jet-intel | srw_build-cheyenne-gnu.log ▼ |
| 2023-03-09 01:43:47 | ufs-srweather-app/job/pipeline/job/PR-637/2/ | false | 117 | FAILURE | cheyenne-gnu cheyenne-intel gaea-intel jet-intel | srw_build-cheyenne-gnu.log ▼ |
| 2023-03-08 16:29:15 | ufs-srweather-app/job/pipeline/job/PR-637/1/ | false | 554.5 | FAILURE | | srw_build-cheyenne-gnu.log ▼ |
| 2023-03-06 16:44:27 | ufs-srweather-app/job/pipeline/job/PR-632/1/ | false | 167.3 | FAILURE | cheyenne-gnu cheyenne-intel gaea-intel jet-intel | srw_build-cheyenne-gnu.log ▼ |
| 2023-02-24 18:37:40 | ufs-srweather-app/job/pipeline/job/PR-628/1/ | false | 218.5 | SUCCESS | cheyenne-gnu cheyenne-intel gaea-intel jet-intel orion-intel | srw_build-cheyenne-gnu.log ▼ |
| 2023-03-03 18:51:37 | ufs-srweather-app/job/pipeline/job/PR-627/1/ | false | 432.1 | FAILURE | cheyenne-gnu cheyenne-intel gaea-intel jet-intel | srw_build-cheyenne-gnu.log ▼ |
| 2023-02-23 16:50:45 | ufs-srweather-app/job/pipeline/job/PR-626/1/ | false | 140.6 | ABORTED | cheyenne-gnu cheyenne-intel gaea-intel jet-intel | srw_build-cheyenne-gnu.log ▼ |

# Implemented Process

**UFS-SRW Application - Example**

- *Infrastructure* – Update the CICD pipeline of the SRW to include the driver for forecast verification
- *Scientific Hypothesis* – Evaluate the impact on severe winter weather with the relevant UFS case to validate the hypothesis, i.e., Indianapolis case
- *Objective Verification* – Calculate skill score index based on weighted average of a combination of metrics (RMSE), variables (wind speed, dew point temperature, temperature, and pressure at the lowest level in the atmosphere), and lead time
- *Output* – Every source code update has a performance indicator; i.e., aiming for higher than 1.0.

**Next Steps**

- Expansion of the infrastructure to all the UFS repositories, already in the Land DA
- Well-established problems with focused research and development
- Significant increase to HPC resources dedicated to the testing, i.e., currently the SRW testing is not triggered due to the lack of resources
- Support for incremental change in development mindset, there are already great examples

# Tutorials and Training

# Community Engagement Activities

**Community Portal and Resources**
- Regular Updates, FAQs
- Detailed descriptions of products and Services
- Feedback Pages / Incorporating Feedback

**Social Media Campaigns**
- Twitter
- Facebook
- Instagram

**Webinars and Workshops**
- Host webinars and workshops for EPIC community
- Topics related to EPIC, model dev and data analysis

**Community Events**
- Application Training
- CodeFest
- UIFCW

**Publications and Newsletters**
- Publish latest developments
- Articles, impacts and contributions
- Guides and technical documents for users

**Outreach and Marketing**
- Increase awareness of EPIC and community
- Collaborate with external partners and stakeholders
- Targeted messaging and communications strategies

# Upcoming Events, Projects & Promotions

- Quarterly CodeFests & Application Trainings
  - Short-Range Weather CodeFest 2023: Unit Testing Framework for UFS - April 3-7, 2023
  - Short-Range Weather Application Training 2023: Running V2.1 Containers in AWS - April 7, 2023
  - EPIC CodeFest June 2023: Unit Testing Framework for the UFS - June 19-23, 2023
  - EPIC Application Training June 2023: Land Data Assimilation (DA) System - June 23, 2023
- Plan/Host UIFCW, Summer 2023
- Quarterly Video Tutorials
- UPP webpage on ECP (support transitioned from DTC to EPIC)
- Explore combining EPIC-UFS Communications Strategy
- Conferences (AGU & AMS)
- Launch an EPIC-UFS Newsletter
- Develop an enhanced metrics dashboard for the ECP
- Identify potential areas for improvement and engagement, discover new ways to incentivize external participation

# UIFCW

Notable Information
- Event: Monday, July 24, 2023 – Friday, July 28, 2023
- Location: Boulder, CO & Online
- Final Registration Day: Friday, July 17, 2023 (no on-site registration)
- Abstract Submissions closed: Wednesday, May 31, 2023
- Training Demos
  - Contributing to UFS/EPIC GitHub Repositories
  - Land Data Assimilation (DA) System
  - Live Demo Running the UFS SRW Application in the Cloud

# Need for testing and innovative processes
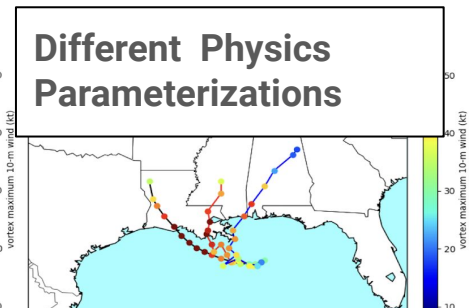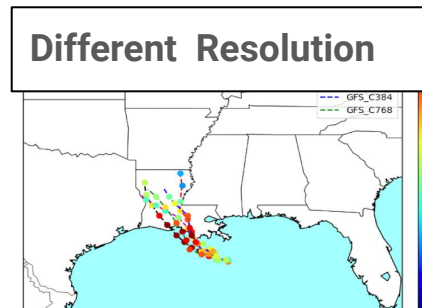
# Testing Framework

Objective: To quantify the impact of any code update, in terms of forecast accuracy and computational performance.

- Homogenize testing infrastructure
- Optimized testing (Reduction of cost)
- Simplification of Code Management
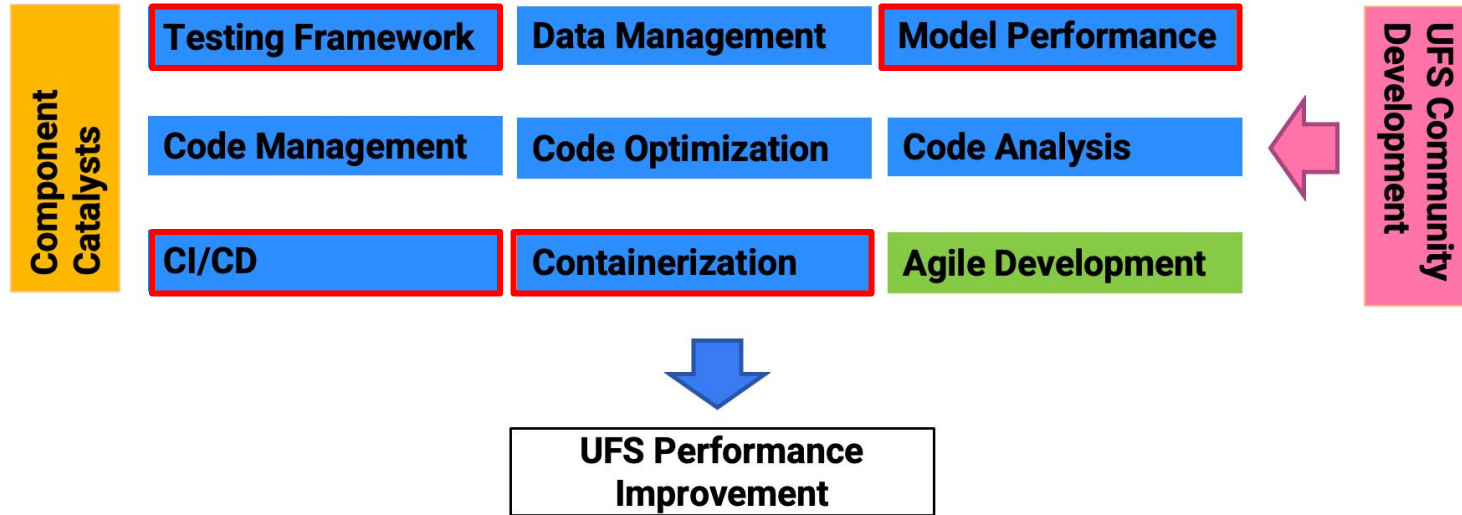- User-friendly
- Multi-level testing



```
     Start 1: ocn_unit_test
1/4 Test #1: ocn_unit_test ...................     Passed     0.75 sec
     Start 2: ocn_scm_test
2/4 Test #2: ocn_scm_test ...................     Passed    11.09 sec
     Start 3: datm_ocn_test
3/4 Test #3: datm_ocn_test ...................     Passed   248.82 sec
     Start 4: ufs_case_test
4/4 Test #4: ufs_case_test ...................     Passed   370.78 sec
100% tests passed, 0 tests failed out of 4
```

Increasing Complexity



Different Resolution

Different Physics Parameterizations

# Infrastructure as an Innovation's Catalyst

Work in Progress by EPIC contract with the UFS Community



| Component Catalysts | Testing Framework | Data Management | Model Performance | UFS Community Development |
|---|---|---|---|---|
| | Code Management | Code Optimization | Code Analysis | |
| | CI/CD | Containerization | Agile Development | |

↓

**UFS Performance Improvement**

*Creation of a prosperous environment for rapid innovation!*

**DRAFT & PREDICISIONAL**