# OpenMFC Evaluation Infrastructure

**Lukas Diduch**[*+]

Chocot Baptiste [+], Ghorbanian Bajgiran Ilia [+],
Haiying Guan[+] and Yooyoung Lee[+]

[*](CTR) Dakota Consulting, Inc.
[+] Multimodal Information Group (MIG) , Information Access Division (IAD), NIST

OpenMFC2022 Workshop : Day 1, Tuesday, Dec. 6, 2022

NATIONAL INSTITUTE OF
STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

INFORMATION
TECHNOLOGY
LABORATORY

# Disclaimer

- Certain commercial equipment, instruments, software, or materials are identified in this article in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the equipment, instruments, software or materials are necessarily the best available for the purpose.

- The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

- All images, graphs, and charts are original works created for DARPA MediFor as well as other NIST Programs.

# Acknowledgement

- NIST contributors
  - Peter Fontana
  - Timothee Kheyrkhah
  - Jesse G. Zhang
  - Edmond Golden
  - Noah Dove

# Talk Overview

- Evaluation Driven Research Cycle

- OpenMFC Evaluation Infrastructure Components
  - Overview
  - Public-Facing Infrastructure
    - Website
    - Leaderboards
  - Internal Infrastructure
    - Indus Framework
    - Scoring Code

# Motivation

- Evaluations across various domains are conducted at MIG each year

- Conditions and requirements vary across domains:
    - Tracks/Challenges/Phases, Sites/Teams, Licensing, Transfer-Modes
    - Constraints: Data release, Timing, Reporting, Leaderboard

- Need for a tailored but rapidly adaptable Evaluation Management Platform (EMP) while addressing core requirements such as availability, security and usability

- Platform must play along with internal scoring microservices as well as internal and external cloud services

# Evaluation-Driven Research
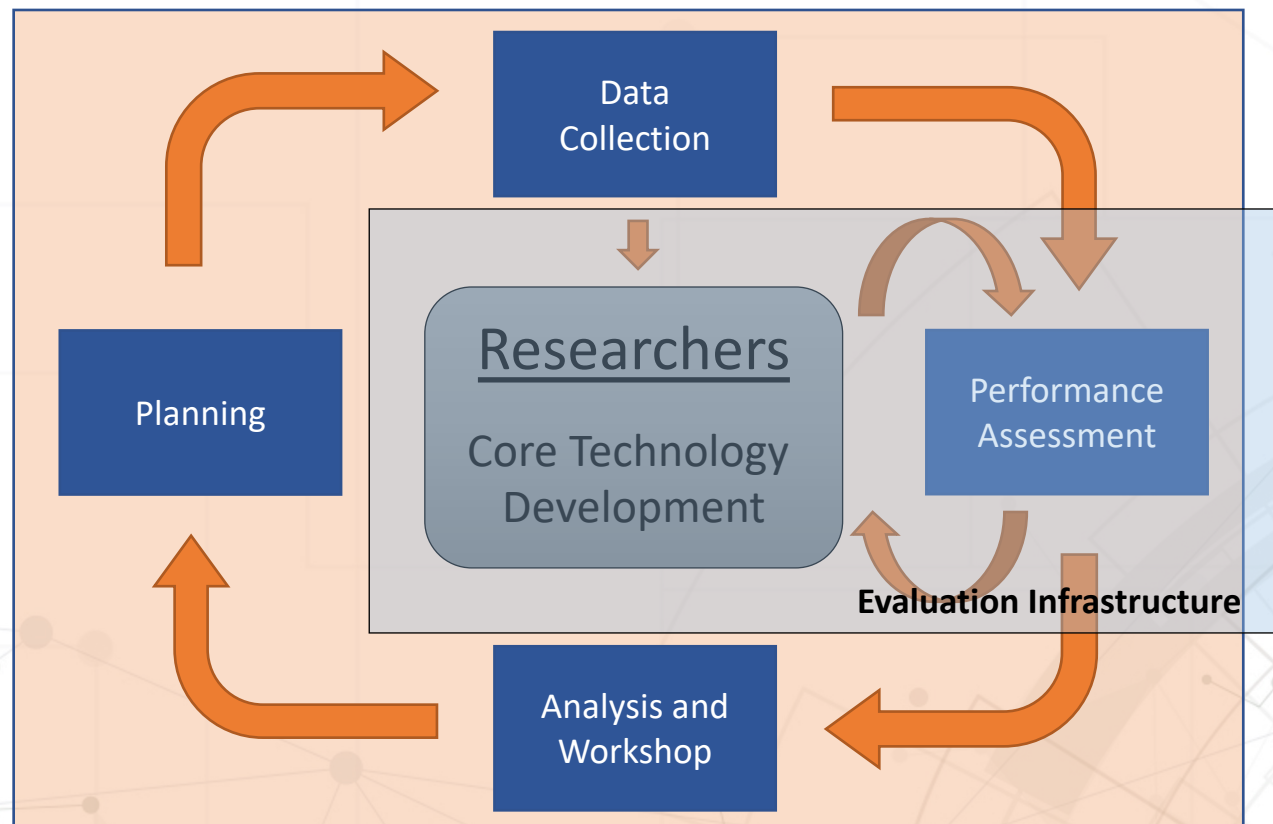
# Evaluation Infrastructure Overview

## Network Partitioning & Systems Architecture of the Evaluation system
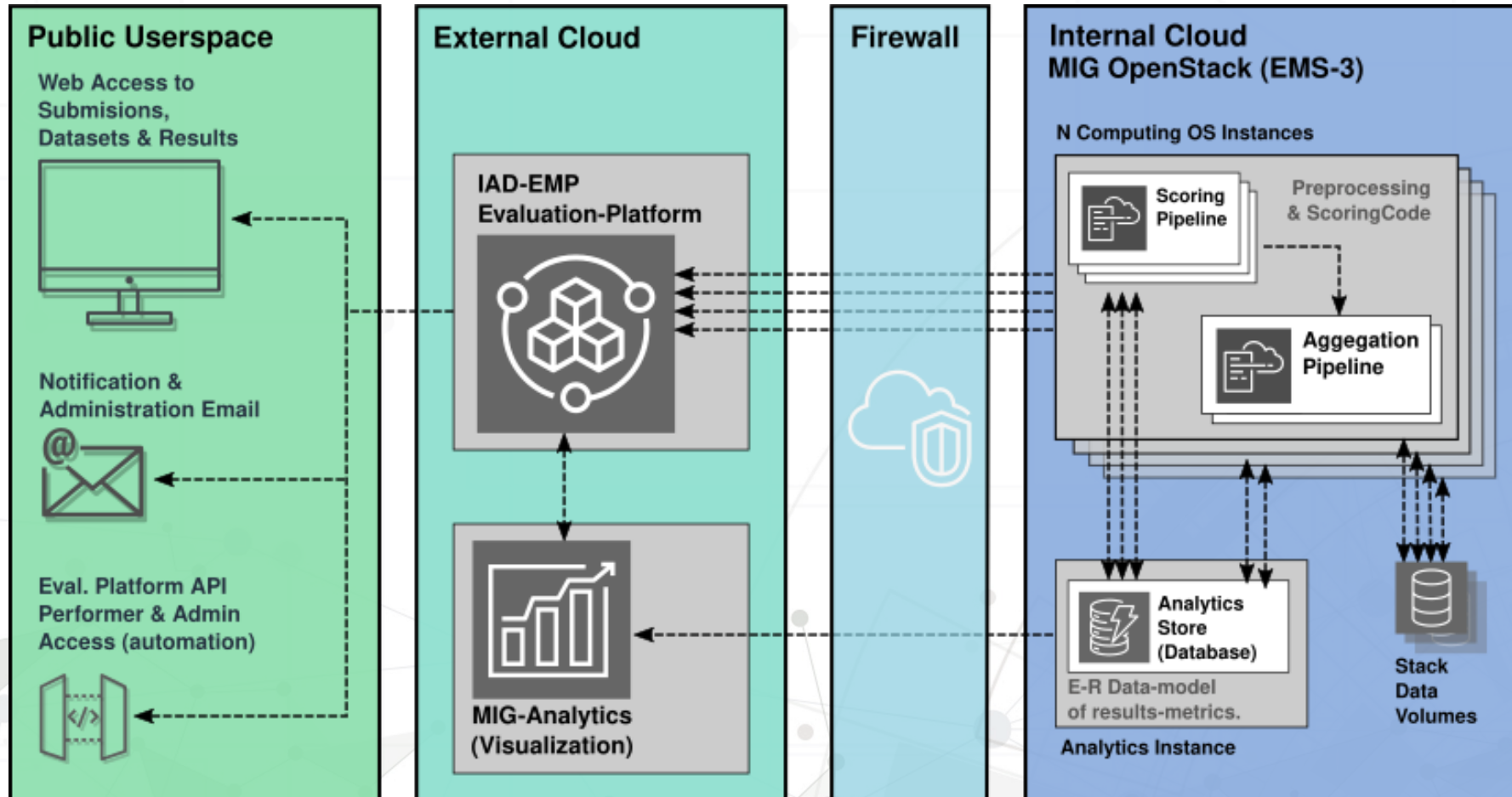
# Public-facing Infrastructure



Content Management System (CMS)

Evaluation Admin UI

Performer UI

Teams, Licensing System Output Uploads

Results
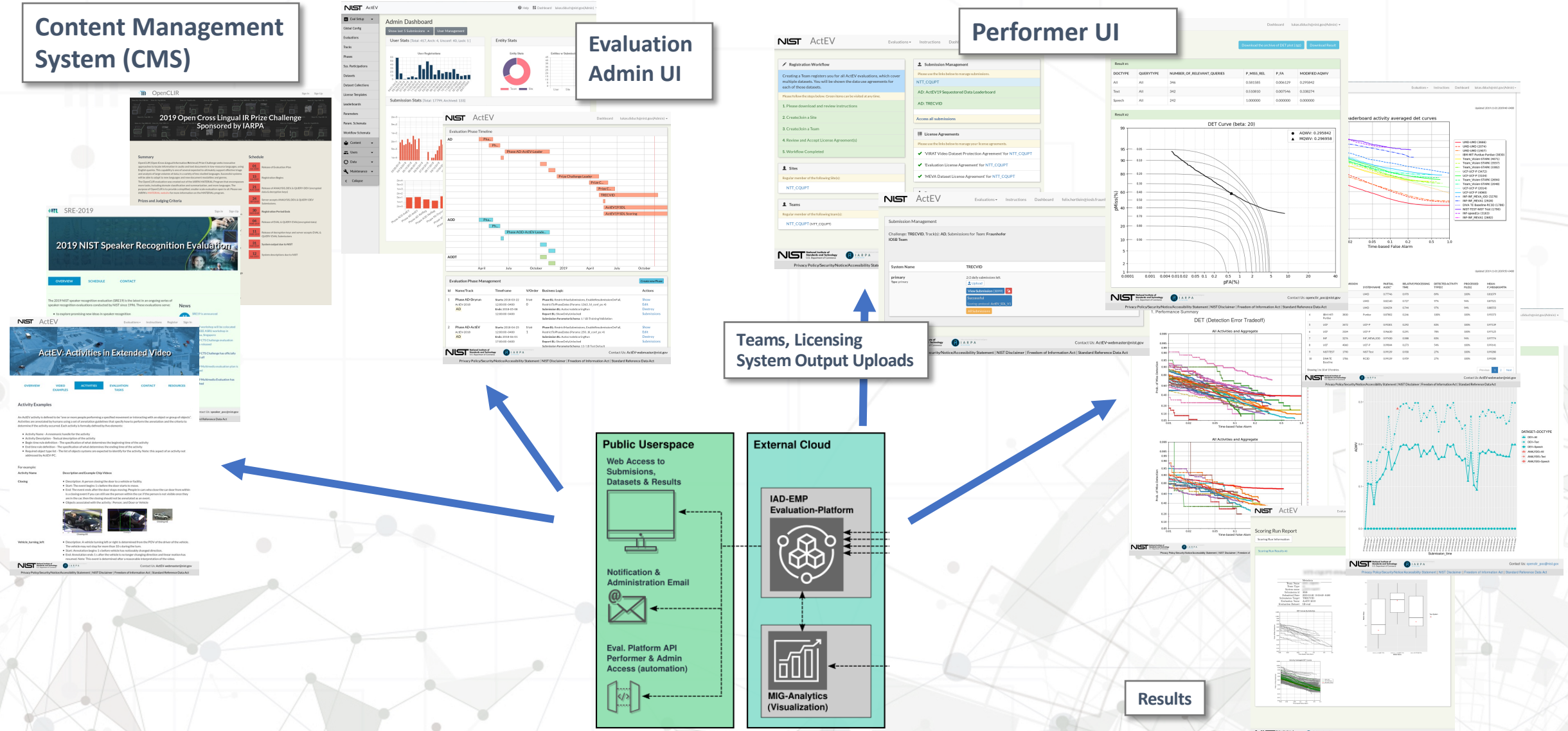
# Content and Access Management

# Performer Dashboard

**General Participant Interface: Home Dashboard**

1. Evaluation Workflow

2. Sites & Team Managements

3. Evaluation Tasks & Submission management

4. License agreements

5. Datasets Access

# Phases, Systems, Submission Management

- **Tracks** represent an evaluation task
- **Phases** represent stages of a Track across a time-period
- **Systems** represent different system instances/ implementations (e.g. training sets or system parameters etc.). Systems and Phases form a Matrix
- **Submissions**: System Output to be scored against sequestered test dataset

# Submissions

- **UI Generation : Submission Form example**

- Customized evaluation parameters stored as Json-schemas

- Parsed & Validated in ruby to generate the HTML & JavaScript interface

Form Features:

- Form elements (text input, drop-down list, multiple selections, etc.)

- Conditional forms (mutual exclusions between form inputs)

**New Submission**

Select submission parameters if applicable. Use the 'Browse' button to select a file from your computer and upload it to the server by clicking on 'Submit'.

**System:** test
**Evaluation:** OpenMFC 2020
**Track:** Image Manipulation Detection and Localization
**Phase:** MFC20
**Allowed file formats:** `.url`

Dataset

OpenMFC20 Image

Input condition
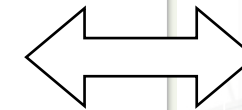
Image only

Evaluation protocol

Detection
Localization

Specify Submission URL

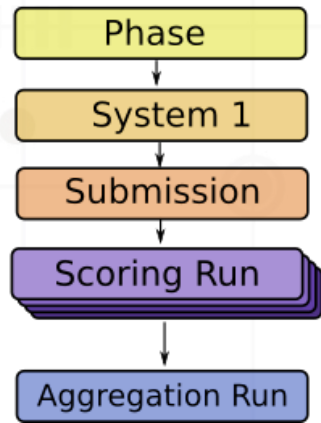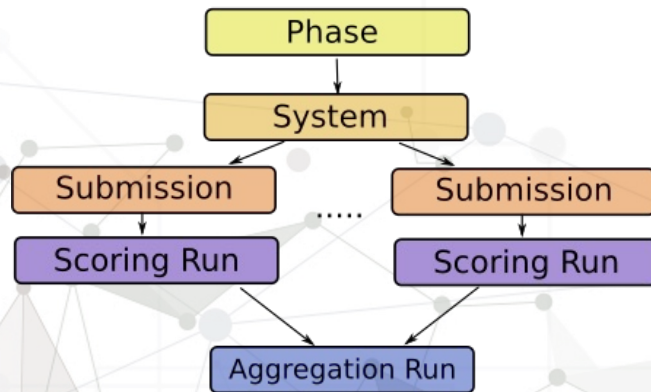Enter URL to fetch ressource from here.

Submit

Back

```json
{
  "type":"object",
  "required":[
    "Dataset",
    "Evaluation Protocol",
    "Input Condition"
  ],
  "properties":{
    "Dataset":{
      "type":"string",
      "enum":"nil"
    },
    "Evaluation Protocol":{
      "type":"array",
      "items":{
        "enum":[
          "Detection",
          "Localization"
        ]
      }
    },
    "Input Condition":{
      "type":"string",
      "enum":"nil"
    }
  }
}
```
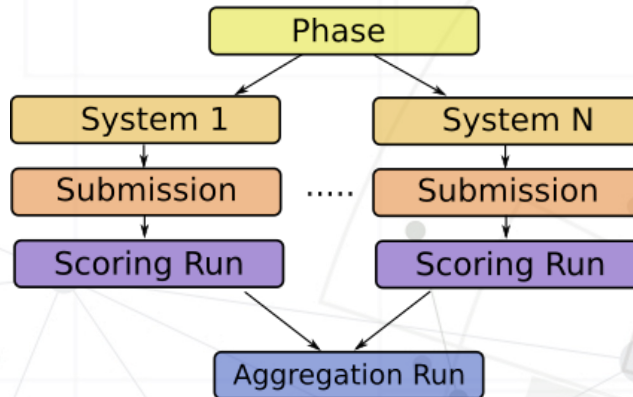
# Submissions, Scoring Runs, Aggregation Runs



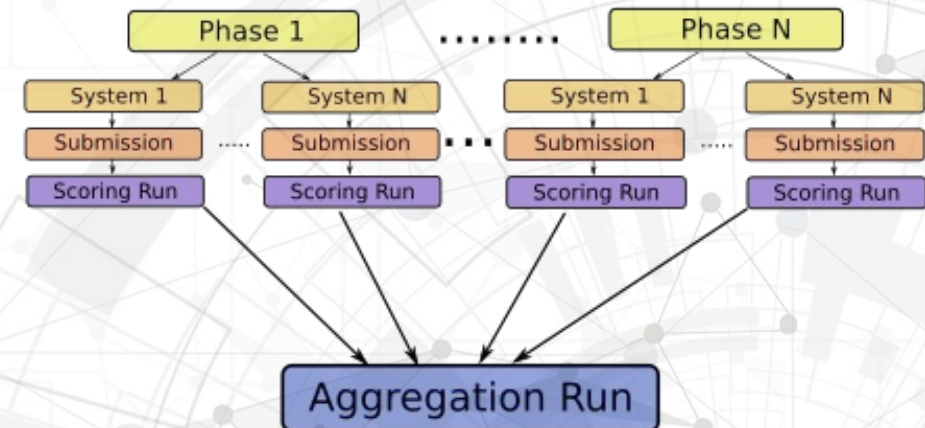Across Scoring-Runs
(one System, one Submission)

- System Output is considered a **Submission**
- Multiple **Scoring Runs** can be run against a submission
- **Aggregation Runs** can be run against a set of submissions or scoring-runs, or aggregation-runs and merge them together for comparison

Across Submissions
(one System)

Across Submissions
(multiple Systems)

Across Phases
(multiple Systems, multiple Phases)

# Scoring-Run Report

Few minutes after teams make a submission they can see scores / error-logs associated for their submission.

# Leaderboards



- Manipulation Detection (MD)
  - Image_MD (IMD)
  - **ImageSplice_MD (ISMD)**
  - Video_MD (VMD)
- Deepfake Detection (DD)
  - Image_DD (IDD)
  - Video_DD (VDD)
- **Steganography Image Detection (StegD)**

# IndusR Backend



## Software overview

- IndusR is a ruby *command line tool* for running pipeline jobs in concurrent and distributed environments

- Each pipeline is defined by the user's **environment**, **config file** and associated set of **scripts**

- The config file is used to fully describe the pipeline **parameters**, which includes server setup, sequence of steps with their respective scripts and associated hooks and hook parameters
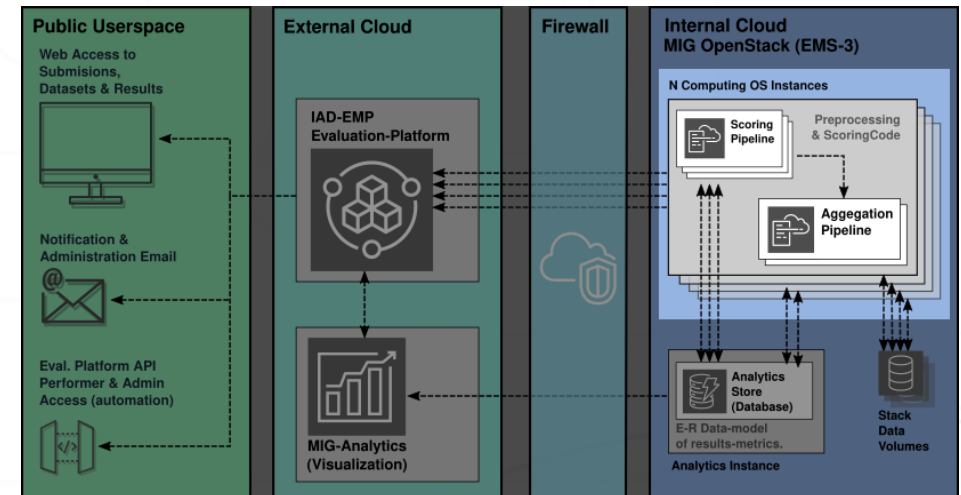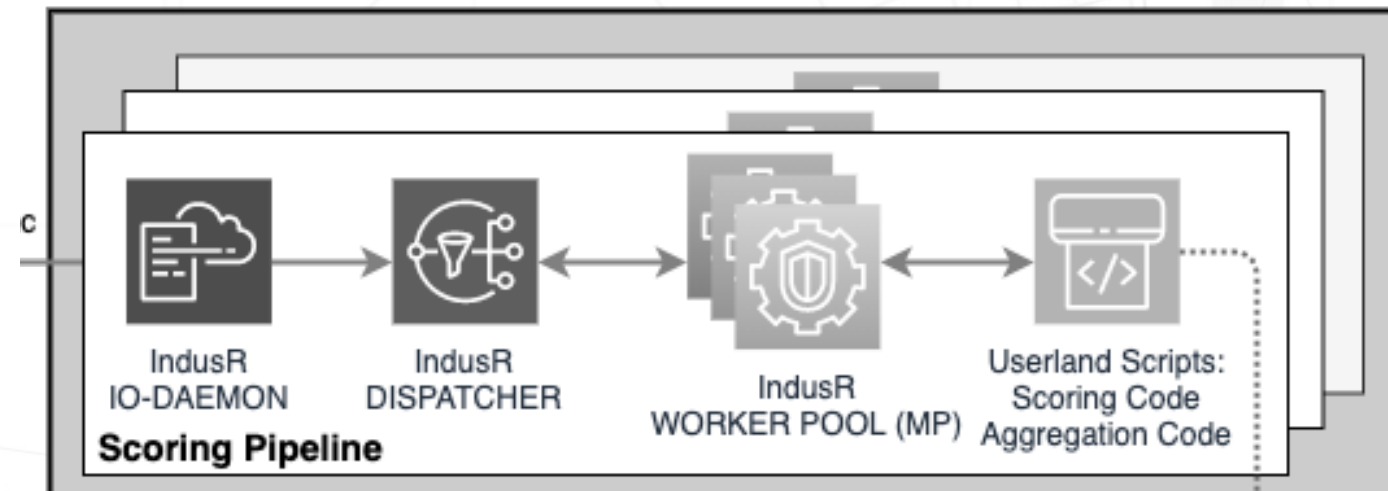
# IndusR: Configuration Driven

**Pipeline Config File :**

```
evalid: MFCBackend
redis: "redis://localhost:6379"
dispatch_type: scoring_run
eval_root_dir: !ENV INDUS_BE_EVAL_ROOT_DIR
io_interval: 45 # seconds
pipeline:
        setup: step_setup.sh
        download: step_download.sh
        uncompress: step_uncompress.sh
        validate: step_validate.sh
        detscore: step_detscore.sh
        locscore: step_locscore.sh
        updatedb: step_updatedb.sh

hooks: […]
[paths…]
```

- **Additional Scoring Pipeline Hooks :**

- IO Hooks

  - RSYNC/SSH: push, pull

  - REST API: push, pull + complex query for pull

- POST step success/fail Hooks

  - bind a script to execute based on step condition

- Configurable Log-scrubbing

  - Define scripts filtering out sensitive information and excessive detail

# IndusR Components
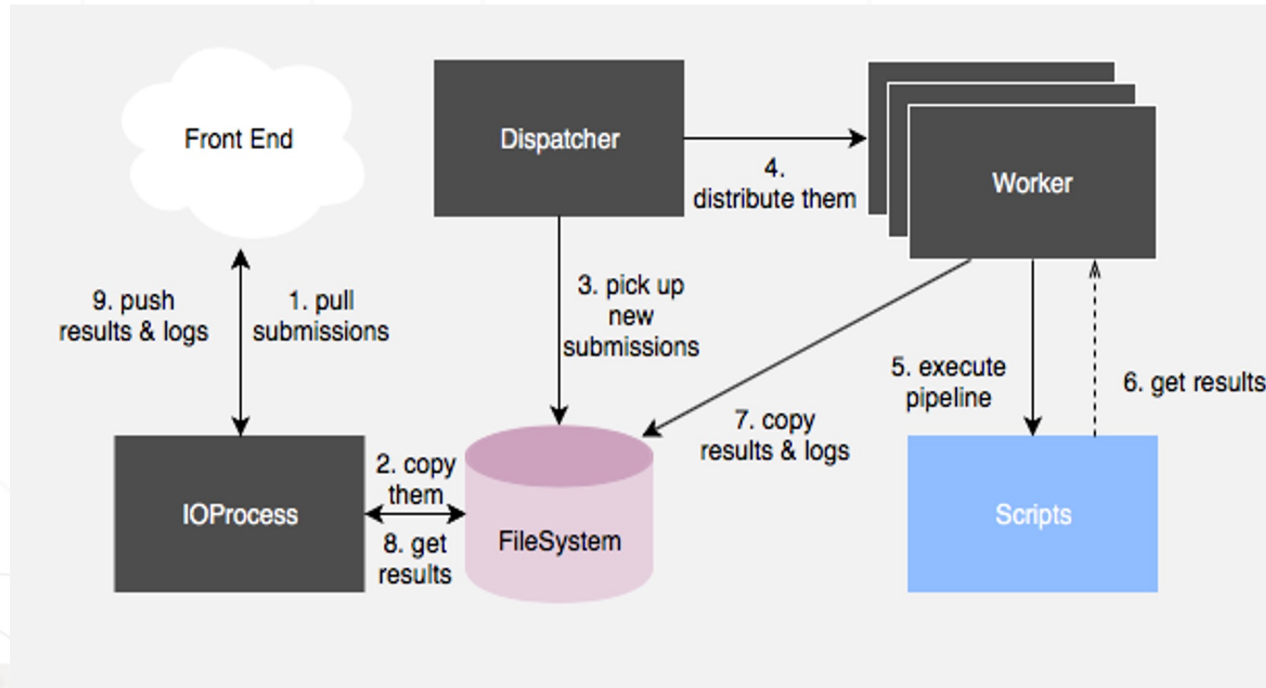


- **I/O process**: automatically retrieve and upload submissions/scoring- and aggregation-runs/status/results between WebUI and Scoring Cluster either via SSL or REST API

- **Dispatcher process**: queue incoming runs

- **Worker processes**: Execute single step of a step-sequence expressed as a scoring- or aggregation pipeline running against the performers submission. Workers can be bound to individual steps

- **Redis**: Key-value store to synchronize application state across all distributed/parallel processes

- **Scripts**: Userland scripts written by evaluators

# OpenMFC Implementation

**Scoring Pipeline Implementation:**

- One scoring pipeline handles all eval Tracks

- One aggregation pipeline handles all Leaderboards

**Configuration Driven mainly through Database:**

- Database stores
  - **Track configuration**
  - Track scores
- Leaderboards can use DB for aggregation or as a source for complex analysis online and offline
- Advantages:
  - Adding new tasks and datasets quickly to the evaluation logic is straight-forward
  - Database w/ Scores can be hosted anywhere
- Disadvantages
  - Advanced setup
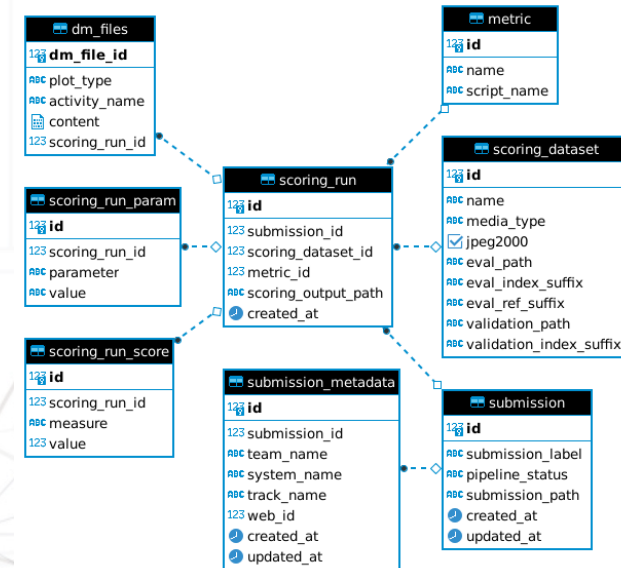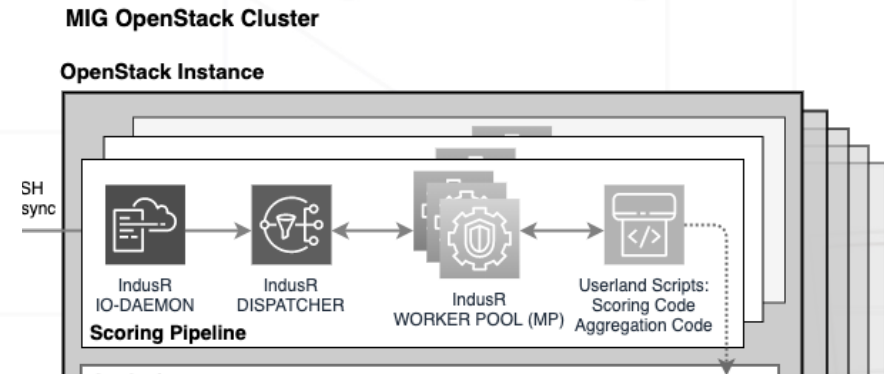
# Conclusion

- As an **independent Scoring Entity** we are providing
  - Fair **comparison of systems performance** across community peers
  - **Standardized metrics** against **sequestered** Dataset(s)

- **Our Evaluation Infrastructure** is
  - Providing all **essential evaluation resources** online
  - Facilitating and **managing scoring process** within eval-constraints
  - Providing **tracking of progress** across different modalities
  - Providing scoring computation resources (for open evaluations)

- Leaderboard based **evaluation cycle** enables **rapid R&D**

# Questions?

OpenMFC team: mfc_poc@nist.gov

# Thank You!

NIST OpenMFC 2022