Title

* Hello
  + Thank you to the committee
  + For the invitation to speak briefly to you today
* My name is Andy Gallo – software architect at GE Research
  + represent a cross-functional physical-digital team – physicists, SE
  + collaborating for several years now on workflow tooling for design systems
  + with utility at GE Research, GE Power, Aviation
* Presenting a refactoring of workflow tooling from our industrial experience
  + Refactor – restructure, smooth out design rough edges over time, rollup themes
  + Includes experience with many national laboratory software projects & facilities
  + OLCF
  + ECP, E4S

Architectural themes for this refactoring of the workflow platform

* 1) Complexity in the hardware & software systems – increasingly heterogeneous
  + Hardware - CPU, GPU, etc.
  + Software libs, too many to count, at the bleeding edge, high barriers to use
* Approach: address the complexity with loose coupling of logical components, shield from details, front with APIs / interfaces
* 2) Actionable insights at all stages of design process requires digital thread 🡪 metadata
  + 5Ws
  + Data provenance – where did the data come from, where did it go, what decisions were made using it
* 3) Software Defined Everything – allocation of compute, network, workflows
  + Repeatable
  + Auditable
* 4) Goal: drive trust, virtual test in lieu of physical
  + Sufficient fidelity
  + Model validation, UQ, decision provenance
* **View of the world is not app centric – process centric** 
  + “the workflow is the app”
  + When does an HPC app get run without pre-/post-processing? Almost never.
  + Intend to capture metadata about the process and all the objects (people, apps, workflows, computing resources, data) used in it

Refactoring #1: Four Pillars for engineering design workflow platform

* An enterprise system which enables our design engineering workflows must provide four main functions or subsystems each with a small set of action verbs:
  + Auth – authenticate & authorize use within tenancies
    - Verbs: login, “am I authorized?”
    - Tenancy = a group of users with a similar interest & data classification
  + Run
    - async job exec & status reporting – verbs: run job, check status
    - job chaining – triggering on job events – control flow & data flow
  + Repo
    - Put & get data under management by tenancy
    - Distributed storage, medium agnostic – S3, bare FS, HDFS, HTTP
    - “MetaRepo” index
    - System & arbitrary user metadata (name=value)
    - Find by metadata – leveraging FAIR principles
  + Spin
    - Provision / deprovision compute resources
      * e.g., cloud – “node with 4 CPUs and 32GB RAM”, “a turnkey HPC”
      * e.g., with specific applications installed – a “compute type”
    - Or other devices / resources…
    - Laboratory devices – human-in-the-loop
    - Details hidden behind spin up / down API verbs
* Upon these four pillars are stacked a secure REST API, and upon that, native language interfaces (Python) and GUIs
* Workflows are authored in Python using the API
* Due to the programming model and triggering mechanisms, workflows may create and alter their own futures
  + Very flexible, but adds debugging complexity

Refactoring #3: Three Types of Workflows

* Type 1) Intra-job / in-situ
  + E.g., extraction of interim results, simulation steerage
* Type 2) Inter-job / intra-site
  + The site provides the four pillars – Auth, Run, Repo, Spin
  + Jobs run & trigger other downstream jobs
  + E.g., HPC data driving ML training
* Type 3) Inter-site
  + Each site implements the four pillars with their minimal verb set
  + Incl. normalization of status reporting – “COMPLETED” == “DONE”
  + Workflows can span sites, collect status and trigger jobs on different sites
  + Façade tested on GE’s internal design system, NERSC Superfacility API
  + E.g., MxN CI/CD
* This is a system for workflow interoperability, not reusability – there is no assumption of “write once deploy anywhere”
  + E.g., Site.Run.runJob() allows arbitrary site-specific arguments

Workflow Examples

Clockwise from top left:

* In-situ / intra-site – type 1 & 2 hybrid
  + In-situ analysis results in call on DT4D AP
  + Posts job status incl. app/workflow-specific declarative GUI
  + Human interacts, message returned to the running sim
  + 🡪 Interactive in-situ steerage
* MxN CI/CD – inter-site
  + Maintain readiness of N applications on M platforms
  + Build, test, rollup results
  + Reduce time to onboard to allocation, lower barriers, democratize
* Spin – intra-site
  + Spin up, use, store data & metadata, spin down, control cost function
* Intra-site GE design workflow for CFD application driving ML training of surrogate

Type 3 Workflow Tooling

* Exposes the Four Pillars – Auth, Run, Repo, Spin – with their minimal (but extensible) set of verbs
* Hides the complexity of the site’s own scheduler, persistence, authentication… all the legacy details
* Could also front devices… a particle accelerator, a test stand – e.g.,
  + 1) Provision the resource
  + 2) Put data “to the device”
  + 3) Run some “job”
  + 4) Get data “from the device”
* Type 3 layer = on the order of hundreds of lines of Python per collaborating site
  + Prototyped on GE’s internal design system, NERSC Superfacility API 🡪 others?

Next Steps for Collaboration

* Further refactoring of the workflow types & subsystems model; unification of tooling
  + Messaging – between workflow types
  + Resource Types
  + Standard APIs
  + Zero trust security
  + Formalisms for metadata using FAIR principles
* Plug-and-play MxN CI/CD
* Multi-order modeling, model quality assess & notate
* Workflow visualization & navigation – past, present, future state of the workflow
* Spin

Thank you for your time.

Questions?