2017 Achievements

- Many backends matured implementing the CWT-API
  - LLNL: CDAT
    - subset/aggregate/regrid/min/max (including curvilinear)
  - NASA/NCCS: EDAS
    - *13 available: emul, ediff, min, emin, max, emax, sum, esum, avg, eavg, rms, erms, ediv*
  - CMCC: Ophidia
    - subsetting along any dimension (space and time), maximum & minimum along a specific dimension
- All compatible via End User API
- COG Integrated Front End
- Ready to be considered as part of installation
- Started work on a common test suite (https://github.com/Ouranosinc/CWT-API-TestSuite)
- Abstract workflow descriptions
  - CWT-API workflow description
  - JSON schema defined to describe workflows based on wps requests (https://ouranosinc.github.io/pavics-sdi/en/workflows/vocabulary.html)

2018 and Beyond Roadmap

- 1 -Once Abstract Operator Grammar defined
  - Run operator thru standard dataset
  - Then get ESGF certified
  - Can define new “operator”
  - Use namespace to identify the operation’s provider (e.g. nasa.averager)
- 1- Documenting Services
  - For users
  - For discoverability
- 1- Complete common test suite
  - Expose test results to users
  - Make enough information (input files, outputs) available for users to be able to compare their results with those of the CWT implementations.
- 2- Full support for OAuth
  - Log on one site, run on many
  - Openid group?

- 2- Integrated in ESGF release cycle
    - Vetting system for official stack
    - Ways to add to your local node
    - 3.0 installer
- 2- Exception Handling
- 2- Fully distributed
    - NASA and ESGF interoperability
      (call NASA and run part at LLNL)
    - Scalability
    - Discovery
- 2- Helping other teams' work to be compatible with end-user API
    - Ouranos/Pavics
- 3- Abstract workflow descriptions
    - Define Grammar for describing Operator
    - need ability to articulate a workflow at the highest level
    - create a system to instantiate such workflows
        - utilizing best map to existing services and frameworks
        - requires an introspection API to query available services/operators
- 3- Workflows finalized
    - Combining processes into a workflow, mixing backends, processing at many nodes
    - Take a chunk of an entire workflow and run that
- 3- Provenance Handling
    - Which Data
    - Which Process
    - Which Server
- 3- More Operators
    - EDAS modes
    - Handling of irregular grids (CMIP6?)
    - etc...
- 3- Analytics
    - To help get idea about data size and compute time.
    - Eventually helps optimize the workflows.
- 4- More Advanced Caching
    - From least used
    - Time expiration (30days)
- 4- Resources Management (New Operator)
    - User permission
    - Storage
    - Cpu use
    - Dry run, basic checks for data availability and permissions