Testbed Working Group
Draft Design and Recommendations for a Testbed Environment

Version 1.3

Introduction

EarthCube Integration and Test Environment, ECITE, is an infrastructure for test and integration of current and future EarthCube components. In addition to these functions, there is a need to have a repository for storage and preservation of software and documentation (included as artifacts in this document) developed under EarthCube and elsewhere that support the EarthCube mission and capabilities. In the near term, ECITE will focus on:

1. Serving as an infrastructure for test and evaluation of software and EarthCube components, capabilities, and services
2. Fostering Integration and interoperability across funded projects, data facilities, as well as with other important community infrastructures
3. Delivery and preservation of funded projects
4. Offering open and easily accessible computational resource to projects for I&T
5. Creating and archiving EarthCube technical documentation

This document builds off of the current initial requirements (see https://docs.google.com/document/d/13mUy48gD2tC5OoD78wKv7fbkUYjYnJZ-WSDmBjUftJw/edit?usp=sharing) to capture conceptual ideas for the design of ECITE. The requirements are based on the synthesis of responses to a previous EarthCube (EC) Technology Architecture Committee (TAC) survey of the current EC building block and research collaboration network projects, plus input from the TAC Testbed Working Group (TWG) participants, and feedback from the recent Testbed session at the 2015 EC All Hands meeting.

Summary of Funded Projects Responses to Testbed Questions

TWG analysis of TAC survey responses focused on the following survey questions:
Q9 What are the key technologies being used or developed in your project?
Q14 What types of interfaces (or APIs) are used?
Q15 Where is your software hosted?
Q17 How are you addressing sustainability of your project outcomes?
Q28 Do you have a testbed environment within your project that you use for testing and demonstration?
Q29 Would you like access to a testbed environment, for example something EarthCube wide?
Q30 What interfaces with an EarthCube-wide testbed would you recommend for your project?

Q31 Could your software be hosted in an EarthCube testbed?
Q32 What are the technologies required to host and test your products?
Q33 What are your minimum security needs for a shared test platform? (e.g. basic login, shared keys, etc.)

Through a synthesis of the TAC Survey, the TWG was encouraged to find that over ⅔ of the survey participants indicated that an EarthCube Testbed would be useful, and there were indications that this number might have been higher if a clearer description of what an EarthCube Testbed would involve had already been available. The responses were evenly split on whether the projects currently have an existing testing environment available, and a similar split on whether their current components could be hosted on a common Testbed. Three quarters of the responses indicated that they are currently using cloud-based resources, suggesting that a cloud-hosted Testbed would likely be a generally accepted and appropriate environment for the current projects. Responses for the level of security suggested keeping the system reasonably simple with single sign-on across the EarthCube Testbed resources. Respondents provided an extensive list of required technologies for their current projects, all of which seemed consistent with planned capabilities of ECITE.

ECITE Vision (or roadmap)

Core Concepts

ECITE should be designed to be a virtual environment in that there will not be a single system in a single facility that constitutes the environment. Rather, a collection of geographically distributed hybrid (both public and private cloud) computational and storage resources will be managed as a federated testbed, providing a common set of capabilities so users will have a consistent interface to ECITE regardless of the actual type and location of federated resources being employed for a particular test case. The test bed will draw on all of this to provide the potential resources to users that would be best suited for a particular test scenario, in terms of functional capabilities, as well as geographic proximity for optimal utilization and response.

The ECITE should provide common capabilities such as single sign-on, resource scheduling, provisioning, discovery and access and ensure compatibility across federated ECITE resources and facilities. A federated cloud environment creating ECITE that spans various private and public infrastructures will require advanced and secure connectivity as well as broad interoperability across the data and infrastructure. ECITE needs to address challenges and issues dealing with multiple domains, system configurations, security requirements and interaction conventions. One approach would be an employment of predefined and preconfigured common cloud stack along with an automated deployment, allowing easy instantiation and installation of a common software stack that enables a common configuration across ECITE infrastructure, and ease of use by EC projects. These preconfigured software stacks should be the culmination of input and requirements from EC projects.

Management is a critical component for successfully leveraging a federated cloud strategy for EarthCube. ECITE use cases should investigate ways to extend existing centralized management applications in order to develop and design the components needed to manage the hybrid ECITE infrastructure.

Several EC-funded Building Block projects will be wrapping up their initial developments so the need for an integration and testing (I&T) environment is immediate. Initial ECITE efforts should focus on addressing specific building block use cases, with limited testbed resources supporting a basic platform and software stack to begin demonstrating capabilities. These initial use cases should be developed using a quick turnaround agile approach by working closely with selected projects to be sure the ECITE can address their technology and I&T requirements, while at the same time enabling and demonstrating a prototype approach to utilizing hybrid cloud computing resources. Results and lessons learned from these use cases will feed directly into development of an operational ECITE. In addition, these efforts can provide insight in usage and performance and can serve as a base of cost estimation to better plan for sustainability of ECITE.

Additionally, ECITE will provide software and document version control to enable the preservation of EC software, data, user documentation, best practices documentation, and other project artifacts. The approach will also facilitate delivery and capture of EC project deliverables. The GitHub account provided by the EC office should be considered for versioning resources for ECITE activities.

Computational Resources

ECITE should provide seamless access to a system of hybrid computational resources, both public and private. Public resources will include commercial services such as Amazon Web Services. Private resources will include capabilities available at federally funded sites (by NSF, and other agency) and educational facilities that can be available to ECITE. This mix of resources will provide larger opportunities for EarthCube projects to make use of optimal capabilities, as well and demonstrate the flexibility and interoperability of running project software on multiple systems. The composition of the ECITE hybrid system will evolve over time with resources being added or withdrawn as available. Some initial private resources that have been offered include capabilities at George Mason University, the Jet Propulsion Laboratory, the NSF-funded XSEDE system (including the new Jetstream cloud capability), and capabilities at National Center for Supercomputing Applications (NCSA). Other resources will be identified and negotiated as ECITE evolves.

Computational resources include the storage capabilities necessary for the processing and operation of EarthCube project software and functionality as necessary for the integration and testing of project capabilities during the duration of their funded efforts, as determined by

EarthCube. Projects should not expect long term storage of data and results, beyond the funded activities and integration.

ECITE administrators will work with EarthCube projects to ensure support for all the required software dependencies for funded projects/systems, as well as any other functional needs such as operating systems, programming language support, interactive development environments (IDEs), databases, GIS tools, virtual hosting capabilities (e.g. docker, etc.) and other stated requirements. Software licenses will be addressed on an ad-hoc basis to ensure compliance and accessibility.

ECITE will provide tools to monitor and notify administrators about usage of the computational resources to guard against runaway processes that might degrade the performance and waste CPU time (money) where unnecessary.

As listed in the Core section, ECITE will consider utilizing the EC GitHub project to capture software and a variety of possible documentation products.

Recommendations for Staged Implementation of ECITE

A test environment that addresses the needs of EarthCube members is necessarily diverse. The capability needs range from temporary resource access to persistent, high compute load, from large datasets to small, and everything in between, not to mention administrative functions such as user authentication, usage monitoring, process monitoring, etc. The funding needs of course could vary by orders of magnitude depending on the particular choice of capabilities to be implemented. An incremental implementation approach will best address such diversity.
Suggested phases include:
Phase 1: Rapid prototype
Phase 2: Validation
Phase 3: Evolution

Phase 1: Rapid Prototype

Initial phase to implement ECITE is to rapidly prototype a limited scope of ECITE based on immediate needs and requirements developed from assessment (result of funded project questionnaire). This entails first to identify and collect a limited set of artifacts, then provide a test environment to preserve the artifacts for the EarthCube specified duration, and perform verification/integration on a use case by use case basis. As lessons learned from individual testbed operations accumulate and are documented and available to the EC community, they will be helpful to evolve the testbed into a more general, permanent solution.

1. Identify artifacts

One of the first things that will be useful in setting up a test environment is to know; "What resources/artifacts are required and available (including discovery and access) for testing?" Gathering this type of information and then making them more broadly available could be facilitated by storing metadata in a registry and then using a broker middleware for access and, if desired, transformation. For example, CINERGI BB provides one or more registries for EarthCube resources. The current registries, if considered too general, might be enhanced to provide test specific information. It may be as simple as adding new metadata "ResourceTypes" and/or new attributes to an existing CINERGI table. The BCube Broker could then use the metadata registry information (such as defined endpoints and data formats to access and transform data. The Broker could also support other functions of ECITE such a single point signon.

Recommendation:
Investigate use of CINERGI to store metadata describing resource and artifacts of past/current projects.
Investigate the use of the BCube broker for data access, transformation and single source sign-on.
Examples of resources and artifacts for which metadata can be created:

- documentation on test activities
  - identify processes used
  - use cases fulfilled
  - outcomes (i.e. test results)
  - lessons learned
  - software middleware, monitoring, useful metrics, etc.
- setup scripts for servers, services, virtual environments, etc
- source code
- physical resources (e.g. virtual machines, data files) and services for sharing
  - access requirements
  - availability
  - constraints, etc.
- consultation support for test definition, setup, etc.

2. Set up basic ECITE test environment based on identified use cases

Identify a limited set of use cases that an ECITE prototype could satisfy the deliverables of. The next step would be to create a minimum platform and software stack that can be used to:

- serve as the foundation and basic infrastructure and services of ECITE
- demonstrate feasibility of hybrid cloud computing for ECITE

- initially, avoid trying to create one test environment for all use cases, build individual test environments aligned with specific objects until pattern of use emerges
- meet a minimum set of identified requirements

Recommendation:
Work with representative existing building block projects to select and demonstrate test resources. Leverage GMU, JPL and AWS compute and storage resources.

3. Set up basic ECITE environment to Preserve Artifacts

This is a complementary, but independent design choice for an ECITE prototype. Saving time, repeating test results, and increasing testing capability can be greatly enhanced by taking advantage of past work. This section discusses storage of artifacts of digital form, such as virtual machines, sample data sets, spreadsheets, etc.

Two approaches are suggested here: a federated approach when it is possible to expose existing information via a simple web service or linked data and; a dedicated repository approach for those projects that do not or will not have permanent storage ability.

Establishing a repository would provide for long term preservation of artifacts that might otherwise be lost when projects end. The cost and level of effort can be adjusted depending on the size of the store and amount of curation provided. A repository could range from a GitHub style document archive to a place to store virtual machines and possibly data sets for validating test procedures.

Recommendation:
Establish test artifact access either with new repository or standardized access to existing data store facilities.

- setup a simple interface definition to store/retrieve artifacts
    - Investigate use of other EarthCube resources such as GeoDataspace, GeoSoft, GeoLink, etc.
- place the respective endpoints (i.e. locator URLs) in a registry (e.g. CINERGI as mentioned above)
- using artifacts at hand, place respective keywords, brief descriptions, names, etc. in a registry.
- provide brokering or simple web services as needed to deliver respective artifacts
- start with documents and scripts, then incrementally add virtual machines, relevant test data, etc. as needs and funding allow.

Phase 2: ECITE Validation

The ECITE must demonstrate that it provides value-added integration and testing capabilities. Validation of ECITE needs to cover 4 key aspects: functionality, usability, affordability and technology. The infrastructure built of hybrid computing resources along with its management system must be verified. A suggested approach is to identify a use case that requires ECITE-provided computational resources and capabilities and demonstrates one or more EarthCube integration principles. In addition usability testing needs to be performed to determine if ECITE is easy to use and its user interface is functioning properly.

Recommendation:
Work with relevant EarthCube BBs to build a prototype ECITE that can a host a representative implementation. Identify the required resources as well as potential CyberInfrastructure (CI) artifacts which might be used by other EarthCube projects. For example, artifacts might include virtual machines and setup scripts for web service endpoints, a GIS database, and data input mechanisms, respectively. EarthCube integration examples might include catalog content for CINERGI BB and data products for BCube.

Phase 3: ECITE Evolution

As the test needs for geoscience are diverse, so are the potential test environment solutions! As such, it is unlikely that any one physical configuration will satisfy a significant percentage of testing needs. It may be possible however to establish a few baseline configurations that are useful to a majority of test needs.

Recommendations for test environment analysis:
- establish test information and artifacts store, as mentioned above
- use test requirements to define needs, then search and retrieve relevant artifacts
- capture general test environment configuration after patterns emerge
- work with EC governance and office to establish process and resources (e.g. administrative) to ensure alignment with the EC architecture roadmap, sustainability and evolution of ECITE

Appendix A - Acronym List and Glossary

Terms and Acronyms

Artifacts	Any project related materials that would be worthwhile to preserve for future use, to include software, virtual machines, documentation, data, reports, etc.
AWS	Amazon Web Services
BCube	EarthCube building block project providing brokering services for data and services
CHORDS	Cloud-Hosted Real-time Data Services for the Geosciences
CI	CyberInfrastructure
CINERGI	EarthCube building block project to provide a comprehensive catalog of data and services
EC	EarthCube
ECITE	EarthCube Integration and Test Environment
Federated	Approach of using middleware management software to provide seamless interface with multiple cloud resources
GeoWS	EarthCube building block project providing Geo Web Services
GIS	Geographical Information System
GMU	George Mason University
Hybrid	Involving both public and private cloud resources
I&T	Integration and Testing
IDE	Interactive Development Environment
JPL	Jet Propulsion Laboratory
NCSA	National Center for Supercomputing Applications
NSF	National Science Foundation
TAC	Technology and Architecture Committee
TWG	Testbed Working Group
XSEDE	Extreme Science and Engineering Discovery Environment

Appendix B - Contributors To This Document

Emily Law
Don Middleton
Chris MacDermaid
Mike Stults
Doug Fils
Jay Pearlman
Ken Keiser
Chaowei (Phil) Yang
Danie Kinkade