A Project Management Framework for Cloud and HPC Resource Providers[+]

Jefferson Ridgeway[2], Ifeanyi Rowland Onyenweaku[3], Gregor von Laszewski[1*], Fugang Wang[1]

[+] Authors are sorted in alphabetical order

[1] School of Computing and Informatics, Bloomington IN, 47405, U.S.A., laszewski@gmail.com, kevinwangfg@gmail.com

[2] Elizabeth City State University, Elizabeth City NC, 27909, U.S.A., jdridgeway4@gmail.com

[3] Mississippi Valley State University, Itta Bena MS, 38941, U.S.A., rowlandifeanyi17@gmail.com

[*] Corresponding author E-mail: laszewski@gmail.com

ABSTRACT

Cloudmesh is a project that allows the management of virtual machines in a federated fashion. It can be run in two modes. One is a standalone mode where the users run cloudmesh on the local machines. The second mode is a hosted mode where multiple users share a web server through which the virtual machines are managed. One of the important functions for cloudmesh is to provide a sophisticated user management. This user management is currently conducted in drupal through the FutureGrid portal via an integration to the FutureGrid LDAP server. However, as the rest of cloudmesh is developed in python, the user management in FutureGrid has had some limitations. It is important to identify a python based solution with more advanced features in order to re-implement the user management functionality in cloudmesh to address long term sustainability of the user management component.

This research will explore how to design a data model in python via mongoengine to represent users and user created projects to identify how we can leverage either django or flask to manage such created projects. To address the later, we will first evaluate if we can get a django web application started and identify how to use it. If this turns out to be too difficult, we will fall back to flask. As part of the management, we need to implement a queue in which users are queued for approval, and a project queue whereby projects are queued and approved by a committee. A simple backend system written in python will support this task and provide an abstraction that is outside the web interface.

Categories and Subject Descriptors

D.3.3 [**Programming Languages**]: Language Constructs and Features – *Python Programming Language*

General Terms

Algorithms, Design, Documentation, Management, Performance, Reliability, Verification, Experimentation

Keywords

Cloudmesh., Project Management, User Management, Project Review Management, FutureGrid, MongoDB, Django,

1. INTRODUCTION

Ever since the inception of clouds and their functionality in maintaining data, the field of cloud computing has grown immensely. To provide users with a testbed, Indiana University offers FutureGrid for users of clouds, HPC, and Grids FutureGrid. FutureGrid enables researchers to experiment research challenges that are related to the applicability of grids and clouds [1]. The test-bed aids virtual machine based environments, and native operating systems for experiments aimed at minimizing overhead and maximizing performance [1]. Furthermore this test-bed has been the motivating driver for Cloudmesh. Cloudmesh allows for (a) federated resource management of virtual machines , (b) bare metal provisioning, and (c) access to a rich set of interfaces including REST, shell, and a python API.. The goal of cloudmesh is to provide Software Defined Distributed System (SDDSaas)[2]. Currently, Cloudmesh uses flask as web development framework. While there is no issue with using flask as the main web development framework, the cloud computing community uses django as web development framework. Django operates in a similar fashion as flask, such as displaying views, using certain templates, and other components, but mainly it is more widely used and accepted within the community.

The goals of our activities include to develop a role based project management framework with users, project, and project comittees to vet the projects and to evaluate if django can be used instead of flask as the web development framework for accessing Cloudmesh databases and much of the logic in Cloudmesh. As django has a very high learning curve we assist new developers by providing a number of selected examples addressing issues such as themes, forms, views, models, and mongodb integration. This also includes creating proper and appropriate documentation on how to install and manage a Django server. [4]

2. REQUIREMENTS

Cloudmesh requires a management framework dealing with users, projects and project review committees. These requirements are listed and discussed in details next.

2.1 Users

Cloudmesh allows user-based authentication. A user needs to have several attributes in order to assist in the identification of users in case of an emergency or security breach. This includes not only an e-mail address, but also a concrete postal address with

phone number and possibly also a FAX number. In case of a security breach a user must be able to be disabled immediately in order to prevent damage to the infrastructure. Naturally, we also need to store activities that relate to users activation, and modification in case of a security audit. The information about this is restricted and can only be modified by the user. Personal information is not exposed to other users. Administrators of cloudmesh have full access to the user data. Table 1 shows a summary of the attributes collected and managed as part of the user object. Additional objects can be added at runtime as well. Important to note is that additional objects can be added at runtime. Originally we considered to use attributes as defined in LDAP's inetorg person object [ref], but we found we do not need a precise definition of an address, as a single text field is sufficient. However, we need additional information that is not provided in the inetorg definition to for example record expertise, and user agreements, as well as recording a log that identifies when a user modifies the user data for security auditing purposes. Hence we have chosen a very simple approach on managing the attributes for the user.

2.1.1 User and Committee Roles

The management framework is using user roles as part of the authorization framework. We distinguish roles such as regular users, administrators, and project committee members that approve projects. Within the projects users can be project leads, project managers, alumni, and members. A project committee approves each project. From practical experience with FutureGrid it was sufficient to only have a single project committee for all projects. However in this framework we generalize this approach to allow for individual and different project committees based on the projects. We also plan to allow changes on the project committee over time and must have record of past and current constitution of the committee members with timestamps when potential changes took place. In addition to the various forms of membership in the project, we need to make sure need to be able to activate, deactivate and block users. For committees we also establish a membership for committee members, an alumni role, and a committee lead and managers. Managers do have the same privileges as the lead. This allows the lead to delegate some of the administrative tasks to a manager. Hence we have the same role names for projects and committees, but they naturally have slightly different functionality.

2.2 Projects

Users in cloudmesh must be part of an active project to use resources registered as part of the ploudmesh provider framework [ref]. The project leads and managers decide and manage the project mebers and alumnis. The project itself is reviewed and approved by a project committee that is assigned to each project after a project application is submitted to cloudmesh. The information provided with the project will assit the committee to judge appropriateness and scope of the project. It includes a justification about why the project is important and how the project is executed. Important attributes that are required include title, abstract, intellectual merit, broader impact and many more as listed in Table 2. Obviously the data model for this framework must be able to support the process of application and review. Once a project is submitted a committee reviews the project abd decides if the project needs to be approved, denied, or the application needs to be improved to work towards approval. A project is in pending state as long as it has not been assigned to the other categories. A project that needs to be improved maintaines in pending toll the desiccion of approval or decline has been cast. The actions of the vetting process are clearly logged as part of the project in order to provide an activity log in case of a project specific audit. This is helpful to identify also metrics such as average approval time or time it takes between a project and actual usage of resources while associating such data with other log information from resource providers. An important part of the project application is also a clear indication of which resources and services are used which provides valuable information to those managing the infrastructure.

2.3 User and Project Approval

Users and project information must be verified before they can be activated. The user is verified by validation of the information entered. Include the username, email, institution, country, and much more as listed in table 1. Part of the user validation process is that a registered user cannot create another user account. This is as the email of the about-to-be registered user is compared to all the email addresses in the database.

In accordance, the project undergoes several screenings too as a way of its approval. The project approval works in a way that no instance of project object is successfully created without its' information verified. Only after this verification can the project be approved and queued up for review.

3. Related Activities

Objects

3.1.1 Users

The users object is made up of essential attributes listed in Table 1. In contrary to Web based social portals every user is vetted before they can apply for a project. This has been identified to be necessary in order to avoid spam project applications. We will reconsider this logic and identify automatic mechanisms for identity providers such as the integration of InCommon or the allowance of certain domains as identified as part of the e-mail address of the user. Once a user is approved they can apply for projects. Even before a project is approved, users may already add project members and identify project members. Administrators can deactivate users immediately and any activities of such users on registered resources is disabled.

3.1.2 Projects

By users being registered, the building of an efficient approval framework for the validation of projects can be accomplished. The variables with their definitions can be seen in details in Table 2.

3.1.3 Roles

Roles are components of both the user and project objects. However, the definition of these roles differ in both objects. In the user objects, the roles are active, inactive and blocked, while, in the project object, the roles are admin, lead, manager, member and alumni. These roles determine permissions granted to the users.

3.1.4 Project Committee

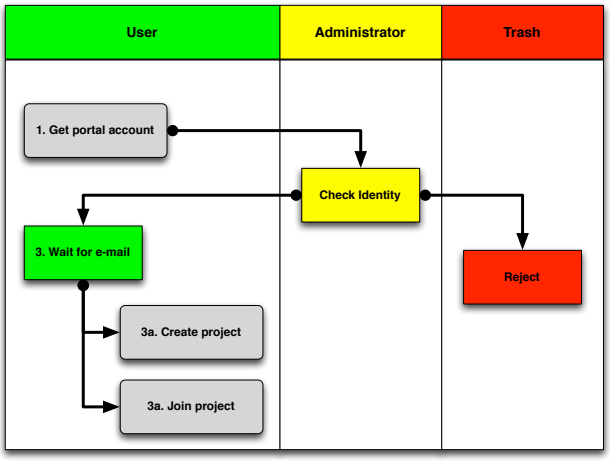
The committee object is defined to add users to a projects committee by default. If needed, reviewers can be added or removed from the committee of a particular project. Projects are queued up for their respective committees. The project committee object also notifies members, alumni and project management of any information regarding to the research they are working on. This object is also responsible for enabling and disenabling reviews.

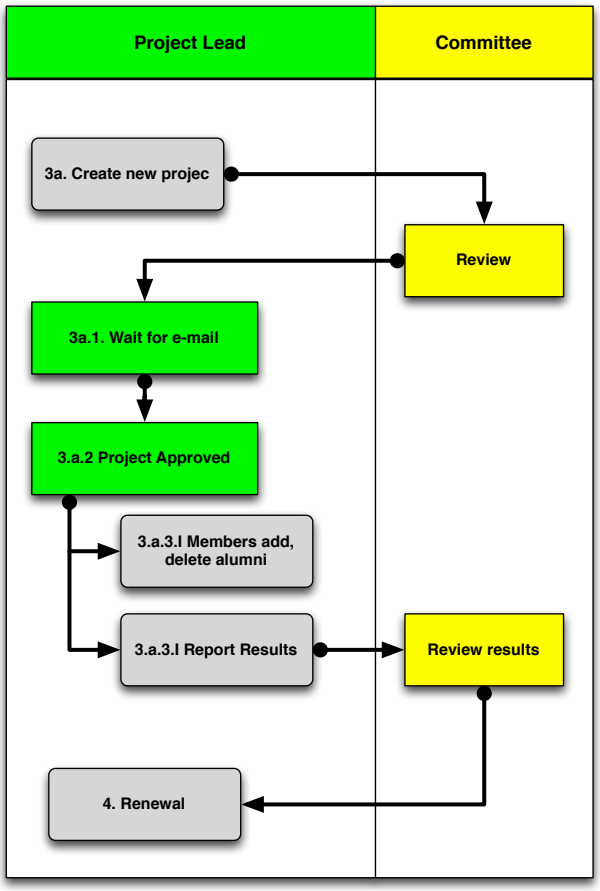
3.2 Workflows

This section shows the different connections between the objects and their different functions.

3.2.1 User Management

At the registration of a user, the user object deals with how to add a user, block a user, deactivate a user, and set the activation and deactivation date of the user. The user can either be active, inactive, blocked or just viewable. An admin is assigned to manage the responsibilities of other users and oversees other user activity.





3.2.1.1 User Application

Before one can become a registered user, one must fill in the required fields on the user application page. These fields can be seen in figure 1. These fields have different instructions in which the registered user must heed to before being registered.

3.2.1.2 User Approval and Vetting

After filling in the required fields, the information inputted is verified by the functions created within the user object. These functions verify that all usernames are unique and all emails are unique. If there is any fault found by the program, the user is not approved and an error statement is returned.

3.2.1.3 User Disabling

Even after a user has been registered or approved, it is still possible for the user to be disabled. This could occur due to the user admin, the date to be deactivated has been reached, or as a request from the user. If it is by the request of the user, the user is first reviewed by the admin to see if relevant contributions have been made by the user. If so, then the user would not completely be disabled but allowed the permission to view the resources.

3.2.2 Project Management

The project object has built-in functions that perform some certain actions. Actions include but not limited to the ability to add or delete a project, a project lead or project members. The functions within projects are also used to find projects depending on the search information given. More information on the management of this object is explained in the next sets of sections.

3.2.2.1 Project Application

A registered user applies for a project, in accordance a list of fields is required for the user to fill in. These fields are what determine the legibility of the project. More information on these fields can be seen in figure 2.

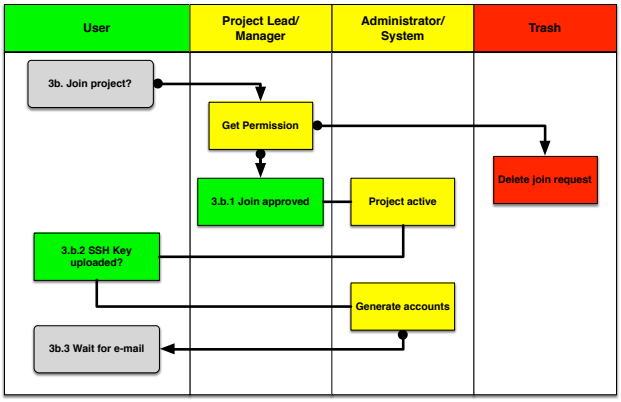
3.2.2.2 Project Approval

After a project has been applied by the user, meaning that all the required fields have been filled, then the project is being queued up for approval and this takes place within the committee. As the committee has the permission to review all information about the applied project and verify the project, the project is on queue and its status is "pending". If approved, then its status is "approved", but if denied the status is "disapproved."

3.2.2.3 Project Completion

When the project has been approved, the project is then worked on by its members. After the completion of the project, the project status is "completed", and is now documented in the database as a past project of its members.

3.2.2.4 *Joining a project*



3.2.3 *User key management*

See fg documentation

3.2.4 *Committee Management*

The committee in details is the object that manages the approval system of the applied projects. This part of the model is capable of setting up the committee of each project, adding new committee members, storing reviews, and notifying all the members of a project about the status or messages about the project. This component is also capable of listing projects according to their status.

3.3 Interfaces

3.3.1 *Command line and shell*

The Command Line and shell Interface was used in this project to read data from mongoengine. This data included Users, Groups, Approvals/Disapprovals, and other data within the database. This was important because if this data from mongoengine with mongoDB as a backend can be displayed in the Command Line and shell then, the data should be able to be displayed using the Django Framework. This is a sign of the possibility of Django working because all of the programming has been done in the Command Line and transferring the data into Django is only a matter of changing the syntax so that Django can read and understand it, so that it can be displayed on the web interface.

3.3.2 *REST*

Django REST framework is a Web Application Programming Interface (API) that will be used to connect to the mongoDB database using mongoengine. What makes this ability work is the fact that the REST API has a program called Serializers that "allow complex data such as querysets and model instances to be converted to native Python datatypes that can then be easily rendered into JSON, XML or other content types" [7]. JSON is JavaScript Object Notation and "is a way to store information in an organized, easy-to-access manner" [8]. MongoDB usese JSON documents for storage of the data. This relationship will be implemented in this research.

Table 2: Django Management commands

a)	django-admin.py startproject project_name *Creates new Django Project*
b)	python manage.py startapp app_name * *Creates Django app that holds models and views*
c)	python manage.py syncdb * *Syncs backend of Django to the database displayed*
d)	python manage.py collectstatic * *Collects the static files (css and js files) from the static olderin the django project*
e)	python manage.py runserver * *Runs development Django server*

3.3.3 *Browser*

The Browser will be the final interface that the user will see that will have the Django Bootstrap theme template for all of its HTML web pages with the Django REST framework using mongoDB as a backend.

3.4 Integration

3.4.1 *Keystone*

3.4.2 *LDAP*

3.4.3 *OAUTH2*

3.4.4 *OpenID*

4. Implementation

4.1 Drupal

TBD

4.2 Django

Django is a web development framework that is used to create and maintain websites and web applications that are deal with the construct and management of databases. With maintaining databases and their components, Django uses programs such as models, views, and templates within each project that is created by the client. These programs come collectively to help the user have an easier time with navigating the database and search for their and/or others projects via the Django framework application. Models, templates, and views have been made using the Django framework along with documentation to serve other developers in creating and establishing their own Django framework as well. When the django framework is installed properly, there is an automated admin interface as well. Documentation for models, views, and templates and other components of Django that were used for the Project Management can be seen in Figure 2.

Models

In this component of the Django framework, the models represent the layer and structure for modifying the data in the database [4]. Within both models and fields, classes can be set. Classes in models however are python classes that subclasses "django.db.models.Model" [4]. Each of the variables that are created within the class, serve as fields in the database along with their attributes as well. The fields have many options and types that can be utilized in the database.

Creating a Model

In order to create the desired models, a creation of a website and an app have to be made. Referring to Figure 1 and running commands a) and b), the model.py file will be located in the newly created Django application. After the creation of the fields to make sure that the model can be viewed when the server runs, app needs to be installed in the "INSTALLED_APPS" parameter in the settings.py file that is located within the website folder.

4.3 Views

A view in Django is "a Python function that takes a Web request and returns a Web response" [5]. The user will be able to see the contents of the web request made in the view via web pages that are controlled by the urls.py file in the website folder. The urls.py file dictates which functions or classes are going to be displayed from the views.py file in the django apps folder. Within a view however, after each function is defined, there is either a Response that is returned at the end of the function or a rendering of the function.

4.4 Templates

A template in Django is "a text file" [6]. In this template file, there contains variables created in the Django application. Besides the variables there are block tags as well that "control the logic of the template" [6]. Normally in a template, there is an extends tag at the beginning of it to let the text file know to go to the base html file that will cover and design the webpage to look a certain way so that the user is not looking at bare skeleton html page. In order to have both a template and a base html file, there has to be a "templates" folder created in the django website directory and then a "static" folder to hold css and js files of the base html. Our full documentation and the description for each sample use case can be seen in the following table in Table 1. We describe each use case in more detail next.

4.4.1 Mongo

Mongo is " an open-source document database, and the leading NoSQL database" [7]. MongoDB is equipped with High Performance, Easy Scalability, Advanced Operations, High Availability, and a Document Database. All of these components come together to have the availability to make the Users, Projects, Roles, Committees, and Approvals/Disapprovals for our Project Management.

4.5 Flask

This research focused solely on the Django Web Framework for development of the Management frameworkand not flask. Flask has already and is currently being implemented as the general web framework for Cloudmesh.

5. Status

The objects have currently been identified within MongoDB and Django Web Framework tutorials are available to the general public and successfully connection to mongoengine with mongoDB as a backend.

6. Availability

The code is available in github at https://github.com/cloudmesh/management

7. ACKNOWLEDGMENTS

We like to thank Dr. Geoffrey Fox for his support, We also would like to thank the School of Informatics at Indiana University Bloomington and the IU-SROC director Dr. Lamara Warren. This material is based upon work supported in part by the National Science Foundation under Grant No. 0910812.

8. REFERENCES

1. Bolte J., Diaz J., Kulshrestha A., Laszewski G., Wang F., & et al. FutureGrid: Project, *FutureGrid*. Retrieved June 28, 2014, from Indiana University, Bloomington, 2013: https://portal.futuregrid.org/about

2. Laszewski Von G., Cloudmesh:Overiew, *Cloudmesh*. Retrieved June 28, 2014, from Indiana University, Bloomington, 2013: http://cloudmesh.futuregrid.org/cloudmesh/about.html

3. MongoDB Inc., Introduction to MongoDB, *MongoDB*. Retrieved June 28, 2014: http://mongodb.org/about/introduction/

4. Bennett, J., Graham, T., Hurley, G., and Tamlyn, M., Django Documentation, *Django*. Retrieved July 10, 2014: https://docs.djangoproject.com/en/1.6/

5. Bennett, J., Graham, T., Hurley, G., and Tamlyn, M., Django Documentation: Writing Views. *Django*. Retrieved July 10, 2014: https://docs.djangoproject.com/en/1.6/topics/http/views/

6. Bennett, J., Graham, T., Hurley, G., and Tamlyn, M., Django Documentation: The Django Template Language. *Django*. Retrieved July 10, 2014: https://docs.djangoproject.com/en/1.6/topics/templates/

7. Christie T., Django REST framework: Serializers. *Django REST framework*. Retrieved July 11, 2014: http://www.django-rest-framework.org/api-guide/serializers

8. Lengstrof J., JSON: What it is, How it works, & How to Use it. *Copter Labs*. Retrieved July 11, 2014: http://www.copterlabs.com/blog/json-what-it-is-how-it-works-how-to-use-it/

Table 1: User Object

Attribute	Type	Description
Email	String	The user email which must be from one's university or organization. Gmail, hotmail and others alike are not used. This email is not made public.
Password	String	Password provided by the user for the account. A login box is provided, in which if checked by the user would have to be forced to change his or her password on the first login.
Title	String	User's desired title, such as Mr., Dr., Prof., etc.
firstname	String	The user's first name.
lastname	String	The user's last name.
phone	String	This is the user's phone number. The content of this field is kept private and will not be shown publicly.
department	String	This is the user's institution name, department, or similar; Examples are Computer Science Department, Mathematics and Computer Science Division.
institution	String	The name of the user's University, Government Organization, or Company. Examples are Indiana University, Argonne National Laboratory, Google, Open Science Grid. Abbreviations are prohibited.
adviser contact	String	In this field, students are expected to put their adviser's contact information, which includes full name, department, phone number, email, URL, address, and much more. The content of this field is kept private and will not be shown publicly.
institution address	String	The user's institution address.
institution country	String	The user's institution country.
URL	URL	This field, if applicable is the URL of the user's personal profile page in his or her organization's website.
citizenship	String	The user's citizenship, however this field is kept hidden from the public.
bio	String	A short bio about the user.
sign up code	String	This is the user's sign up code, if the user as attained one.

Table 2: Project Object

Attribute	Type	Description
Project title	String	project title.
Category	List of Strings	project categories.
Keywords	List of Strings	useful keywords related to the project, separated by comma (", ").
Lead	List of Users	The person that initiates the project and is responsible for its execution as well as the completion of reporting results to FG.
Institutional Role	Institutional Role	institutional role that best identifies you in your organization. The content of this field is kept private and will not be shown publicly.
Manager	List of users	A person that works with the project lead to interact with FG. If specified, we assume we will contact this person in addition to the project lead when asking for results.
Contact	String	primary contact address for the project. This could be different from the Project Lead and Manager.
Members	List of users	Members of the project. The list includes the username and is not a reference to the user
Alumni	List of users	users that were part of the project but have since left.
Grant Organization	String	Organization of the sponsor. Examples, NSF, DOE, DoD, NIH,
Grant Number	String	Grant Number associated with your experiment, if any.

Grant url	URL	URL to the Grant Abstract on the Grant sponsering web site associated with your experiment, if any.
Results	String	results reports of your project and include pointers as urls. Please also add all references that use FG resources.
Agreement	Boolean	This field documents that you agree to the project
Slide Collection Aggreement	String	This field gives you a choice if you agree that you will provide FutureGrid with Electronic copies of slides from talks that reference your work done with FutureGrid or which mention FutureGrid (.pdfs or other 'not easily reusable' format o.k.; we will ask you for your permission to post slides publicly and will not post them publicly without your permission).
Other	String	Additional comments that did not fit in any of the above fields is added here.
Project Join Button	String	This allows or disallows users, depending on the option chosen, to request to join a project through the portal.
Join Notification	String	This field, depends if checked, indicates if the project lead would like to be notified via email when a user requests to join the project or not.
Orientation	String	Orientation of the project which could either research, education, industry or government.
Primary Discipline	List of strings	Primary subdiscipline as defined by the NSF.
Abstract	String	a short abstract of the proposed research or educational activity using FutureGrid.
Intellectual Merit	String	In reference to NSF merit review criteria, a brief description of the intellectual merit of the proposed research or educational activity.
Broader Impact	List of strings	In reference to NSF merit review criteria, a brief description of the broader impact of the proposed research or educational activity.
Software Contribution	Boolean	This field has a Yes or No option determines if the project will generate software that can be used by other FutureGrid users or not.
Documentation Contribution	Boolean	This field has a Yes or No option determines if the project lead will be able to generate documentation for the project and software created.
Support Software	Boolean	This field has a Yes or No option determines if the project lead will be able to provide support for the software developed.
Hardware Resources	List of strings	Multi-choice selections of hardware resources, which could either be alomo, foxtrot, hotel, india, sierra, xray, bravo, delta or, and other choices.
Provision Type	String	This field contains different options which has different set rules and boundaries on the use of VM's across FutureGrid.
Base environment	Base Environments	Multi-choice selection of Base environment relevant to the research. This could be High Performance Computing Environment, Eucalyptus, Nimbus, OpenStack, OpenNebula or, and other choices.
Services	List of strings	Multi-choice selection of services for the research which could be Genesis II, gLite, Hadoop, MapReduce, Twister, Unicore 6, OpenNebula or, and other choices.
Comment	String	Any information whose field is not specified above.
Use of FG	String	Intent on use of FutureGrid on the proposed research or educational activity.
Scale of use	String	Brief description of the scale of resources expected to be needed in the research.

Table 3: Project Committee Review

Attribute	Label	Description
Projects	List of strings	This is a list of all the projects in a queue to be reviewed by the committee

Reviewers	List of user references	This is a list of all the committee members that reviews the projects
Default Reviewers	List of user references	This is a default list of reviewers that are automatically added to a project's committee when a project application is submitted.
Reviews	String	This is the comment or thought of the reviewers on a particular project.
Default	Boolean	Defines if a project already has default reviewers or not.

NOTE THAT THIS DOES NOT WORK AS EACH PROJECT HAS ITS OWN LIST OF REVIEWERS

Appendix Project Form

This appendix contains : a sample project Information from FutureGrid (does not include the questions about services and resources)

Title: Course: Example Course On Advanced Cloud Computing

Project Keywords:
Course, Cloud, OpenStack, Eucalyptus

Project Lead:
Gregor von Laszewski (portalname)

Project Manager:
Gregor von Laszewski (portalname)

Project Members:
Fugang Wang (use portalname)
Albert Elfstein (use portalname)

Project Alumni:

Project Orientation: *
- [ ] Research
- [x] Education
- [ ] Industry
- [ ] Government

Primary Discipline: *
Computer Science

Abstract: *
==========

Note: this is an example project and is not a real project, although the contents presented in this material is available.

This course will introduce the students at Indiana University as part of the Summer Semester 2012 into the essentials of Cloud Computing and HPC. We will start the course by teaching the students python within one week. As cloud computing framework we ave chosen OpenStack, as it has become one of the ubiquitous IaaS frameworks and is available on FutureGrid (Sierra). Additionally we will teach the students how to program a simple MPI application so that they can further develop the virtual cluster code available from github

(https://github.com/futuregrid/virtual-cluster). We will compare the performance between the virtualized and non virtualized environment as develop with the help of our cloud metrics system a scheduler that enables us to use bare metal provisioned clusters and virtualized clusters on-demand based on resource requirements and specifications. We are aware that the FutureGrid team is developing such an environment, and would like to join the efforts throughout our course with the contributions conducted by the students.

Course Dates:

This class will be taught in 10 weeks as part of the Indiana University CS curriculum. The following dates are important

Start: July 13, 2013
End: Sept 23, 2013
Extension: 1 month for students with programming in-completes.

Course Outline (tentative):

1. Introduction and Overview
2. Essential Python for the Cloud
3. Introduction to OpenStack
4. Programming OpenStack
5. Programming a HPC Cluster
6. Creating a Virtual Cluster
7. Performance Comparison
8. Cloud Metrics
9. Cloudmesh
10. Joining FutureGrid Software Developments

Grading Policies:

Class participation and contribution: 5%
Homework assignments, reading summary, and paper presentation: 50%
Programming assignments: (30%)
Reading Summaries: (10%)
Paper Presentation: (10%)
Course Project: 50%
Proposal: (10%)
Midterm Presentation: (10%)
Final Presentation and Demo: (15%)
Final Report: (15%)

Note:
Homework and programming assignments are due by 11:59pm Thursdays (unless announced in class otherwise). Late homework (non-programming) will NOT be accepted. Late program penalty is 10% per day, according to the timestamp of your online submission. Only when verifiable extenuating circumstances can be demonstrated will extended assignment due dates be considered. Verifiable extenuating circumstances must be reasons beyond control of the students, such as illness or

accidental injury. Poor performance in class is not an extenuating circumstance. Inform your instructor of the verifiable extenuating circumstances in advance or as soon as possible. In such situations, the date and nature of the extended due dates for the assignments will be decided by the instructor.

Please note that FutureGrid does not approve accounts on the weekends. Regular support hours are Mo-Fri 9am - 5pm. Please note that answering support questions does take time. Do not start the night before the homework is due. Plan your programming assignments to be done early.

Intellectual Merit: *
==========

The course will be introducing the students to cloud computing and will also be used to derive new class materiel that we will be using in subsequent lessons.

Broader Impact: *
==========

This class will be educating a number of students in cloud computing programming. Cloud computing is an important factor in job availability after graduation of students, thus this course will be useful to increase marketability of the students. In addition we have in the past also been able to increase participation of minority students. In the past we had 10 minority students and 9 female students taking this class. We intend to work together with Gregor von Laszewski and improve the FutureGrid manual and to make our course material available via FutureGrid through its github and community portal pages.

Scale of use: *
==========

We anticipate the course will have 30-35 students. The course will be using OpenStack and HPC compute resources and requires for selected students access to bare metal provisioning. The course will not require to run computationally intense applications. However, we require that students be able to run up to 30 VMs at a time. We know that this may in peak hours be beyond the capabilities of FutureGrid and are advising our students to kill machines if they are not used. The maximum duration of a single VM will typically be less than 5 minutes.

Results:
========

Columns on Last Page Should Be Made As Close As Possible to Equal Length