# Cloudmesh Resource Reservation

Natiele Bohn, Oliver Lewis, Gregor von Laszewski, Fugang Wang
Indiana University
Dillard University

## Abstract

We present a general resource reservation framework for cloud computing frameworks while also focusing on bare metal resource provisioning. This work is conducted as part of Cloudmesh. It is important that resource reservation is supported in a multi-user, multi-project environment. The team has developed a reservation system in which users will have access to resources based on time slices. The interface to the reservation framework allows access through commandline tool, Web interface, REST, as well as a programming API in python. Through these interfaces the administrator or user can interact and create access control, query the reservations, and apply for new ones, but limits will be put in place so that users do not reserve too many resources and block the system while not allowing others to use it. The system will have an abstract plugin framework that allows the integration of a real bare metal provisioning. Convenient tables and views are developed as part of the Web service development based on previous activities in Cloudmesh.

## Introduction

Cloud computing technology provides by commercial companies provide access to seemingly an unlimited number of resources with enough financial backing. However for cloud computers in an academic environment we do have to work with a limited number of resources due to cost efficiency and availability. By integrating resource reservation into Cloudmesh, it is possible to organize the aaccess to resources based on time slicing. Once our framework runs in the production environment, users will be allowed to create, manage and delete their own reservations and also verify whether there are resources available. The administrator will control what is happening in each cloud computer and better manage the use of these resources. For the users, the main goal is that they know when and which resource can be accessed in a determinate time and also what is the next time available for that resource. Reservations can be scheduled for projects or for individuals. This also allows access in an exclusive mode that is important to conduct reproducible experiments.

## Design

The design features of the reservation framework include a number of convenient interfaces. They include the following:

- a REST interface is provided to enable easy accessibility of the functionality through a variety of frameworks and different programming languages,
- a commandline interface provides integration of the reservation framework into scripting environments and allows access from a traditional compute terminal,
- A command shell is provided to allow the integration with Cloudmesh so that the reservation framework not only from a commandline, but also within a custom shell environment that manages state. The shell is also able to use scripts to assist in execution of repeatable experiments the would otherwise be difficult while just using a commandline interface,
- a Web service framework is provided to allow easy access via a portal framework. The service can either be hosted in a shared environment or run stand alone on a user's computer.

## Screenshots

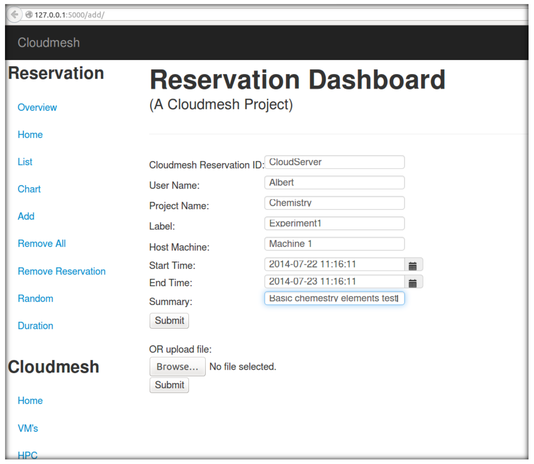


Figure 1: Web interface to add new reservations.

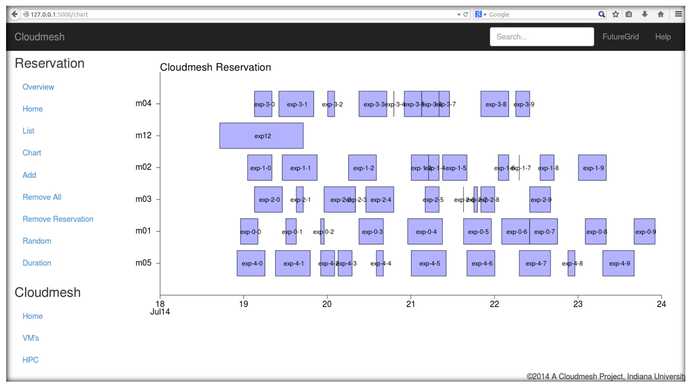


Figure 2: Display of the reservations for various resources.

## Requirements

Cloudmesh Resource Reservation requires a framework to deal with users and reservations. This framework is being developed with Flask, Bootstrap, a REST interface, and templates in HTML. Not every user is allowed to perform resource reservations on all available resources. Administrators must be able to restrict the access.

- Users: as Cloudmesh Resource Reservation is part of FutureGrid, the user authentication must use the same username and password that allows access to FutureGrid.
- User Roles: each user can be linked with one or more projects. To be sure that each user is linked with the right project some security verification will be necessary. A user can be administrator of one project and be just a regular user in another. In case the user tries to access some function that are not allowed, the system should pop up a message and not allow the access to that function.

## Implementation

Using Python as a programming language, a commandline interface was created so the usage of those functions could be tested. To simplify and force the documentation of commandline and command shell functions we have chosen Docopts. In addition we could derive from the API on which the command tools are based also a GUI. We found the GUI is needed in order to interface with the insertion, deletion and other management functions. The GUI is based on Flask a python Web development framework. The creation of HTML templates in Flask were used to generate the web pages implementing the functions for reservations management as well as testing the API. Besides the main functions already listed we also implemented:

- A search function to filter reservations and display them in tables
- a chart to see the reservations (Figure 2)
- a template that calls a function to generate random reservations to make the process of testing possible
- another template to search for the duration of a reservation
- an index page where it's possible to see all the functionalities that the system offer.

To allow display also on modern portable devices such as cell phones and tablets, we have used the Twitter Bootstrap theme throughout our templates.

With the intention of minimize the duplication of code we implemented REST interface. The usage of REST wasn't just to avoid the duplication of code, but also, with the implementation of REST, we are able to integrate more easily with other frameworks and programming languages.

## Acknowledgments

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

## References

1. von Laszewski, G.; Fox, G. C.; Wang, F.; Younge, A. J.; Kulshrestha; Pike, G. G.; Smith, W.; Voeckler, J.; Figueiredo, R. J.; Fortes, J.; Keahey, K., Deelman, E., Design of the FutureGrid Experiment Management Framework, *Proceedings of Gateway Computing Environments 2010 (GCE2010) at SC10, IEEE,* **2010**

2. von Laszewski, G.; Wang, F.; Lee, H.; Chen, H. & Fox, G. C., Accessing Multiple Clouds with Cloudmesh, *Proceedings of the 2014 ACM International Workshop on Software-defined Ecosystems, ACM,* **2014**, 21-28

## Primary Contact

Gregor von Laszewski, Indiana University, laszewski@gmail.com