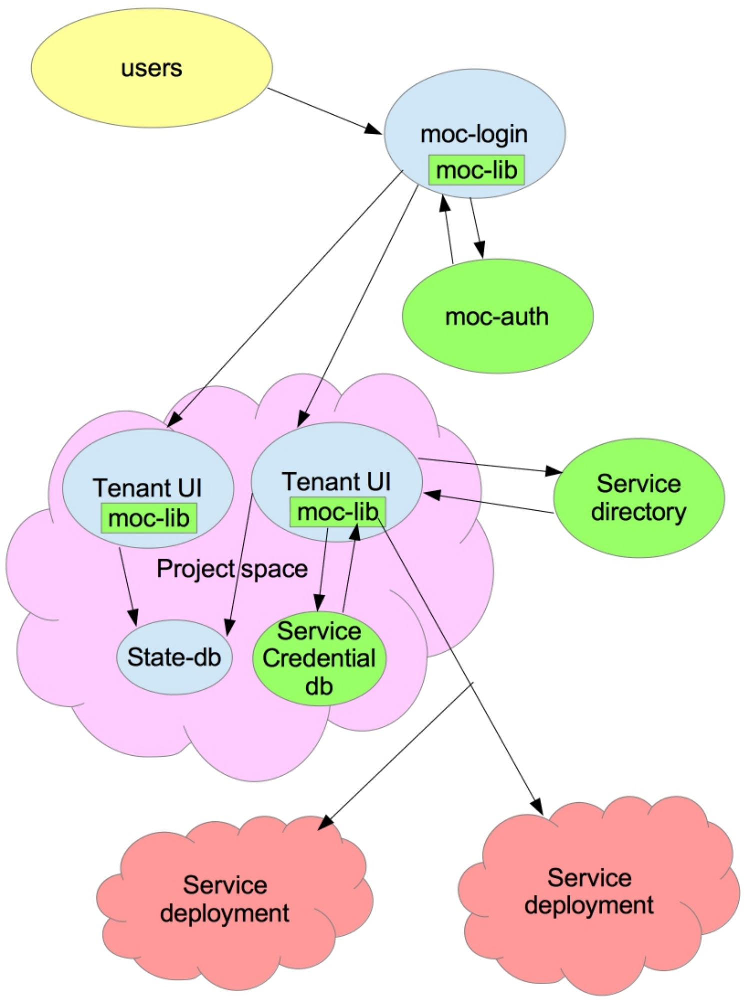
Massachusetts Open Cloud Marketplace by Logan Bernard and Shiran Sukumar

Proposal Content

1. High-Level Goals

I. Provide an interface to allow multiple providers to offer services without needing to trust each other
II. Open a marketplace to show the high and low level customizable offerings of multiple providers
III. Develop a simple user experience

2. Architecture



MOCi architecture

The MOCi is an interface for the MOC Marketplace. User requests with the MOC come from the MOCi and are fulfilled by the MOC Library. The MOC Library is an API that gives users access to using the MOC. To use the graphical user interface of the MOC Library, users sign into MOCi. User credentials are authorized using the MOC Library to check a database of projects and user's associated with the projects. The data is available through the moc-auth machine which is hosted on the MOC. Once authenticated the interface becomes the Tenant UI. For each project, there is a VM reserved to maintain the project state. The Project Space VM acts as a web server for each project with a database to keep track of the project state and a separate database for the service credentials available to the project. The virtualized space is spawned when one user accesses the project, and continues to run while users connect and leave. When the last user leaves, the Project Space terminates. The state maintains the sum of all user activity. Primarily, the Project Space contains a list of what resources have been allocated for the project and the corresponding services. The Project Space does not manage the actual services or resources. Only when accessing a service with the Tenant UI can a user manage resources. To interact with a service, the Tenant UI uses the MOC Library to get the associated service credential from the credentials database within the Project Space. The credentials are passed through the Library to the lower level services of the MOC. The MOC Library provides communication with a central MOC authority. The authority, built on the MOC, will be a Keystone node that verifies the user's credentials. Service Providers can trust any request made within the MOC, because all user requests are verified by the central authority. The Service Provider will interact with the MOC Library to initiate the changes requested by the user. The Service Directory is the mediator between the Tenant UI and the Service Providers. The Service Directory allows Providers to list their resources and availability to Tenant UI. Providers will have to store their resource availability by using the MOC Library. The Tenant UI will be able to access the available services by making requests to the MOC Library. The Tenant UI requests resources for a project through the MOC Library which uses the Service Directory to procure the resources from the Service Provider for the project. Services are exposed through the MOC Library which are visualized by the MOCi.

Scope

The MOC requires tools to facilitate interactions between Providers and Users. A well defined environment must be constructed to provide the MOC with a system that can process user requests. The scope of this project is to layout the foundation for a MOC Marketplace and UI that will ultimately be used by users and providers. In this environment, users will want to take advantage of a well designed marketplace with multiple providers' resources who do not need to trust each other to offer services. The MOC is not a homogenous cloud provider, imagine different storage from EMC and HP, coupled with compute from Harvard. Multiple providers need to trust each other in order to pool and share resources. The security risks involved with the environment mandate users interactions with resources must be authorized. However, providers should not have to handshake with every other provider to offer services. The MOC should have an authorization protocol that maintains the service credentials of each project. The Project Space is a virtual space for multi-tenant environments to maintain project state. Users should not be burdened with overhead to configure or instantiate their projects. The user does not need to maintain their credentials for each resource, or service within a project. Users will have one interface that maintains their authorization. The single interface, MOCi, can provide access to resources through communication with a central MOC authority. Service Providers can trust any request made within the MOC, because all user requests are verified by a central authority using the MOC Library. The message path described by the architecture connects users with the underlying services. The MOC Library serves as the tool used to fulfill user requests. The MOC Library will verify user requests to manage projects and their resources by passing messages through the MOC. Actions executed by the MOC Library and available for user interaction through MOCi. The MOC uses a Library and UI to fulfill user requests by maintaining a secure and personal environment.

Requirements

To have a functional MOCi, the project will be dependent upon the MOC for the implementation of the MOC Library, which will be housed in the MOC and utilize a Service Directory for information on available resources. The completed MOC Library will have the ability to spin up a Project Space VM after a user logs into the MOC. Each user's state will be stored locally in their Tenant UI. The focus of the project will be to create the user experience for the Tenant UI. The MOC Library will also allow the user to manipulate their cloud by adding and removing services exposed by the Service Directory. The Service Directory will be implemented by the MOC and display services available to users based off current providers' exposed resources.

While laying the framework for the components of the MOC Library is in the scope of the project, the project will not handle the implementation. Instead, it will utilize a pseudo MOC Library which returns expected values (e.g. click login, return "getUser"). Similarly, there is a dependency on a pseudo Service Directory, with static data on available resources. Although this project is dependent on the MOC for the implementation and attachment of a real MOC Library and Service Directory, the model of the MOCi will be built with the potential for future integration in mind.

MOCi is dependent on several credential databases within the MOC. The MOC authority and service credentials database will use models, but not be built within this project. However, a sample Project Space will be constructed to show the environment the MOC Library will interact with.

Related Work

Through determining how MOCi will compete in the already existing market, research was required to evaluate the design. The evaluation criteria examined the aesthetic, experience, and the practicality.

AWS

Examining the Amazon Web Service, it is a basic interface that did not seem very user friendly. The AWS is the most widely used interface on the market and does provide a baseline for what one can expect to accomplish. The design was very simple and did not offer any components that could be incorporated into the design. Although AWS is not intuitive, it can be used to build example projects quickly.

JUJU

The most advanced, customizable, and intuitive user experience that provides the MOCi with a great model for presenting a unique user interface for setting up services. JuJu uses a GUI that allows users to drag and drop nodes, select their configuration and deploy. The ease of setting up OpenStack in their demo made a great case for adopting a similar interface. The model of being able to drag-and-drop or click and change components of a cloud infrastructure is very practical and user friendly. Ultimately, to provide the most customizable experience and be able to set up any configuration on the MOC, MOCi needs to be open to possibilities. Rather than limit users to ready-to-deploy packages, we aim to present choices and rather than preventing deployments if there are configuration conflicts. JuJu is a model that MOCi incorporates into its design, because in the future more people will likely want a visualization tool to see their deployments.

MOC-UI

The MOC-UI represents a standard for what the MOC needs. While examining MOC-UI the aim was to find what made the MOC-UI desirable and useful for the MOC. The ease of explanation and intuitive experience stood out as major factors. While JuJu is extremely powerful, it may be hard for some users to quickly use. MOC-UI has a universal design that almost anyone should be able to find exactly what they are looking for and purchase. Most of the design requirements were derived from understanding the MOC-UI.

Horizon

OpenStack’s UI, Horizon, is a barebones interface to deploy projects. The UI is simple, and clean, but is very bland. Too bland for the MOC’s needs. However, the usefulness with Horizon is the wide familiarity with the system. While MOCi will not look like Horizon, the pop up options will try to remain consistent. For example, there is a horizontal bar that represents a VM and the edit buttons should contain the same information that Horizon has in its implementation. The goal with evolving Horizon is to provide a richer experience without over complicating the pre existing user interface.

Milestones and Timeline

Mid-to-End October

Goals: Final Proposal; Final Design; UI Implementation; Demo

Final Proposal and Visualization

Oct 16 - Oct 23

- We will create a final proposal and visualization of our design.
- The finalized proposal
- The finalized design with focus on a market-first approach

Basic Django Implementation of Design

Oct 24 - Oct 31

- Begin the implementation using Django.
- Build out wireframe design
- Third party user will be able to maneuver through the interface

Mid November

Goals: UI Implementation; MOC Library Integration with MOCi; Demo

MOC Library Integration

Nov 3 - Nov 14

- Focus on the underpinnings that will be necessary to make the MOCi function.
- Lay out the API calls necessary for: user authentication, retrieval of available resources, and service manipulation (attachment to User Cloud, billing, etc).
- Integrate a model of the MOC Library (which will be developed by the MOC)

End of November

Goals: Enhanced and Full User Experience

Visual Enhancement and Finalized User Stories

Nov 17 - Nov 25

- Click through the MOCi and understand the intended functionality
- Expose the MOC Library interaction
- Visual elements will be refined

December

Goals: Final Project Report and Presentation

Project Fixes: Odds and Ends

Dec 1 - Dec 9

- Buffer time to pick up any slack

Final Presentation

Dec 9

- Able to give a full walk-through of the MOCi