# **Cloud Resource Library Solution Proposal**

## 1. Vision and Goals of the Project:

The goal of the project is to build a solution for web application development and deploy it to the Massachusetts Open Cloud (MOC).

The solution would comprise of systems with software stacks that would enable development of Python web applications on familiar servers such as Apache, Nginx etc.

Even though the solution is intended for MOC, it could be deployed on any other available cloud service. The solution would be initially developed on the AWS platform and finally deployed on the MOC.

### **Users/Personas of the Project**

All web app developers who have access to the MOC.

## 2. Scope and Features of the Project:

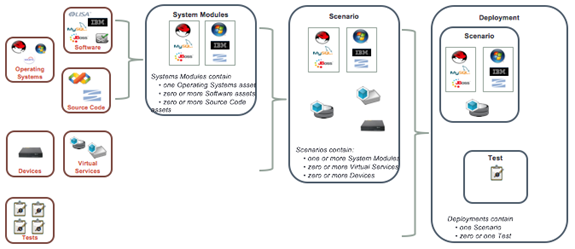
At a high level the solution would comprise of the following systems:

* System 1: A server with Windows/Python/Django/Nginx software stack.
* System 2: A server with Windows/JavApa/eclipse/ache software stack.
* System 3: A server with Linux/Python/Apache software stack.
* System 4: A server with Red Hat/Oracle/MySQl/PostgreSQL Database software stack

## 3. Solution Concept

This section provides a high-level outline of the solution.

Global Architectural Structure of the Project: The key concept of cons3rt is “ASSET”, everything in cons3rt is an asset. The combination of assets is also an asset.



**Figure 1 A Solution asset in cons3rt**

Figure 1 presents the relation of assets. As seen in the figure an asset could be a single component including an operating system, a hardware device or a software. To deploy the configuration to virtual machines, only an operating system is needed, but based on need additional software could be installed. Each solution comprises of Systems which are also assets. In each system module, we can add zero or more virtual services to get a scenario. A Scenario is a complete solution for a specific task. The final step is to deploy our scenario with the desired configurations using the Cons3rt system.

Cons3rt is a platform where we can design our system by using existing component or creating our custom components. The goal of our project is to construct an integrated environment for python web development, we divide our tasks as below:

### **3.1 Development environment stack**

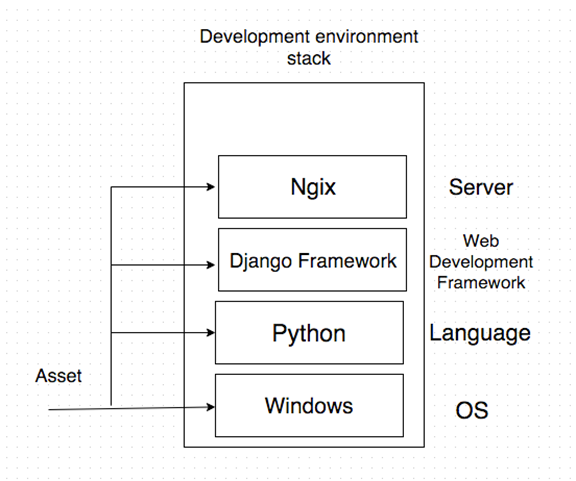
**Window Server**: Use existing asset to install windows server as the operating system.

**Python**: This asset can be deployed on windows machine and install python on windows. Install scripts need to be written to create this asset.

**Django Framework Asset**: Provide Django development kit.

**Ngix server Asset**: Create asset for installing and configuring Ngix server on windows

machine.



**(a)**



**(b)**

**Figure 2 (a) Structure of development environment stack on Windows**

**(b) Structure of development environment stack on Linux**

Figure 2 shows the structure of our development environment stack. Windows/Linux, python, Django frameworks, Ngix/Apache server, these four components make up our development environment. We put the four components in a package and we can view all the component as an asset that we can reuse in our future deployment.

**3.2 Test software stack** – There is the server for developing test applications.

**Windows**: Use the exist resource.

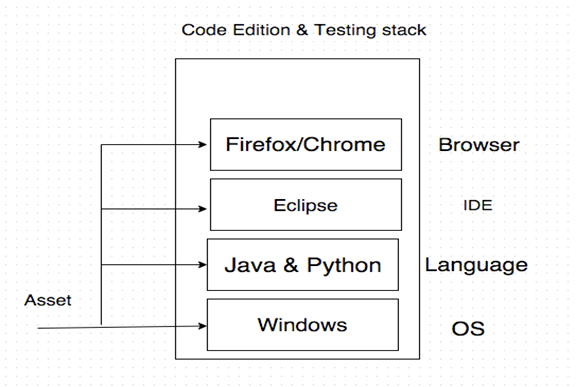
**Java Development Kit:** It is available as an existing asset. This is optional since we plan to install Eclipse in further development.

**IDE**: We will create assets for installing some IDEs like eclipse.

**Browser**:

**Firefox**: Provide the asset for installing firefox. Enable users to debug their code and watch what is going on with their code

**Chrome**: Another environment for observing their code performance.

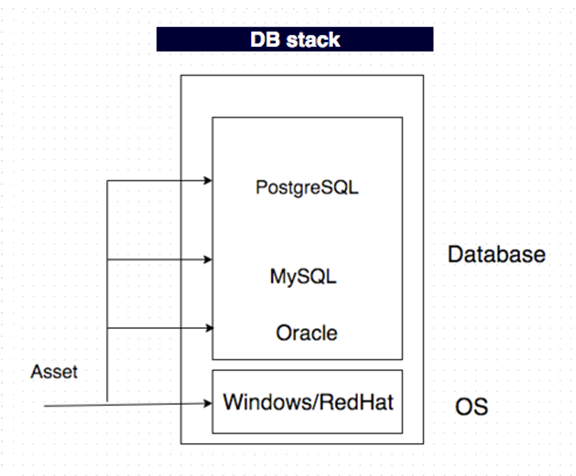


**Figure 3 Structure of Code Edition & Testing stack**

**3.3 Database support software stack** – The server where database is deployed.

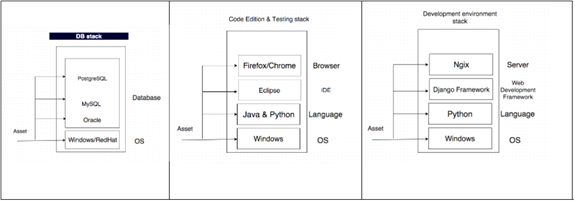
Provide the assets for MySQL, PostgreSQL, Oracle(Optimal) on Windows.

## 



**Figure 4 Database support asset**

**3.4 Overall design**



**Figure 5 Overall Design**

All these three assets consist of our final solution for users who want to do python web development. Using this scenario, user can easily deploy their workspace and focus on developing. If time is enough, we can also replicate the development system on a linux machine. That is easy because we need only replace the development environment stack. The combination takes only minutes for us.

Design Implications and Discussion:

**Principle: Assets are building blocks.**

**Separate functions as much as possible:** On cons3rt, everything should be separated, peers should be alternative. In this way, user can get the exact functions that they want. For example, if user only want a windows machine with python in it, they can choose the asset python and windows. The two components are enough for them. So in our design, if two components are not alternative, they should not appear in the same box. However, when we view peer components in a lower level, they are in different boxes because they actually are different assets.

**Building system is like playing building block**: When we develop our assets, we must consider its compatibility. If components are compatible other users should be able to reuse our component in their project without any problems. This is the principle in our development. In our project, we will create some new assets for the community and we also make use of some existing assets in the community. No matter where the asset comes from, they should be compatible. For example, until now there is not much work in the community about installing python on windows. Once we develop the asset for this task, other users would be able to use our solution as an asset for building their own solution.

## Acceptance Criteria

Here is the list of assets that we will use in our project, each name represents one asset on cons3rt. For each asset, if there is no existing asset available, it is our duty to create a new one. Our first step is to create all the assets that not exist in the community. Next, we will assemble them to provide an integrated python web development environment.

Successful development and integration of the following components will satisfy our acceptance criteria for the solution:

**Operating System Level**: Windows, Linux.

**Programming Languages**: Python for Windows, Java for Windows.

**Database**: MySQL, PostgreSQL, Oracle.

**IDE:** Eclipse.

**Server**: Ngix, Apache

**Browser**: Firefox, Chrome.

## Release Planning :

## Sprint 1 (Week 1 and Week 2) :

## Tasks :

* Setup cons3rt and slack accounts
* Schedule walkthrough session of cons3rt training document with Joe(or any other mentor)
* Familirize with Cons3rt environment and start creating sample assets.
* Start creating systems with required os by reusing existing OS assets :
  + Dev/test systems with Windows OS
  + Dev system with linux OS
  + Database system with Red hat OS
* Write install script for installing Python on Dev system

## 

## Sprint 2 (Week 3 and Week 4) :

* Work on first sprint feedback and re-evaluate sprint goals
* Write install script for installing Java on test system
* Write install script for installing Python on test System
* Write install scripts for intsalling Python on second Dev System
* Write install script for MySQL on Database System
* Write install script for Oracle on Database System(Optimal)

## Sprint 3 (Week 5 and Week 6) :

* Write install script for installing Eclipse IDE on test system
* Write install script for installing Django framework on test System
* Write install script for PostgreSQL on Database System
* Write install script for installing Nginx server on Dev system

## Sprint 4 (Week 7 and Week 8) :

* Write install scripts for installing Apache on the second Dev system
* Write install scripts for installing Apache on the test system
* Write install script for installing Browsers on test System
* Identify and add additional vertical components for each systems to enhance usefullness

## Sprint 5 (Week 9 and Week 10) :

* Package all systems into a single solution
* Deploy the solution to AWS
* Deploy the solution to MOC(Depending on availability of MOC connector)
* Prepare documentation