Response

To

US General Services Administration

Federal Acquisition Service, Integrated Technology Service

National IT Commodity Program

401 W. Peachtree Street Suite 820

Atlanta, GA 30308

FOR

Solicitation Number: 4QTFHS150004

GSA eBuy - RFQ993471

Multiple Award Blanket Purchase Agreements (BPAs)

for

**Agile Delivery Services (ADS I)**

**Technical Approach**

Submitted By



[**www.e-sci.net**](http://www.e-sci.net)

**Submission Date: July 7, 2015 1:00pm**

|  |  |  |
| --- | --- | --- |
| **Submitted To:** | **Submitted By: e-Sci Corporation** | |
| **Corporate Address** | **Point of Contact** |
| |  |  |  | | --- | --- | --- | | Hassan Harris  Senior Contracting Officer   |  | | --- | | National IT Commodity Program | | 401 W. Peachtree Street Suite 820  Atlanta, GA 30308 | | | 404-215-8705 / hassan.harris@gsa.gov | | e-Sci_Final_Full_Logo  5406 Roosevelt Street  Bethesda, MD 20817 | Sushma Kulkarni  Email: [srk@e-sci.net](mailto:srk@e-sci.net)  Tel: 301-257-0426 |

**Table of Contents**

[a. Single Leader Assignment 3](#_Toc424003073)

[b. Multidisciplinary and Collaborative Team 3](#_Toc424003074)

[c. Five Modern and Open-Source Technologies 3](#_Toc424003075)

[d. Deployed the Prototype on a PaaS Provider 4](#_Toc424003076)

[e. Unit Tests for Code 5](#_Toc424003077)

[f. Set Up Continuous Integration System 6](#_Toc424003078)

[g. Set Up Configuration Management 7](#_Toc424003079)

[h. Set Up Continuous Monitoring 8](#_Toc424003080)

[i. Deploy Software in a Container 8](#_Toc424003081)

[j. Iterative Approach 9](#_Toc424003082)

[k. Installation Documentation 9](#_Toc424003083)

[l. Openly Licensed and Free of Charge Platforms 9](#_Toc424003084)

**e-Sci Technical Approach to Agile Delivery Services**

**Overview**

The General Services Administration is seeking proposals and prototypes to establish BPAs for tasks related to Agile Delivery Services. e-Sci Corporation is pleased to submit the proposal and the prototype for **Pool Two (Development)** of the RFQ for Agile Delivery Services.

This document addresses the technical approach adopted by e-Sci team and explains how our agile methodology follows the prescribed guidelines of the U.S. Digital Services Playbook. The following sections refer to the criteria specified in Section “24. EVALUATION AND BASIS FOR AWARD / A. NON-PRICE FACTORS: / Pool Two: Development Pool” of the RFQ and explain how e-Sci team meets those criteria.

1. Single Leader Assignment

e-Sci president Sushma Kulkarni was chosen as a single person of authority and responsibility to make sure that the entire project was executed smoothly and thereby provide full accountability. Ms. Kulkarni oversaw the process of agile methodology, ensure that the relevant artifacts were generated and appropriate tasks were assigned to team members.

1. Multidisciplinary and Collaborative Team

For the design and development of the prototype FDA Drug, Device, and Food datasets, we used a compressed Agile Sprint approach. We put together an agile team for the roles of a Product Manager (Category 1), Technical Architect (Category 2), Frontend Web Developer (Category 6), Backend Web Developer (Category 7), and DevOps Engineer (Category 8).

Ms. Sushma Kulkarni took on the role of Product Manager, and became the single point of contact for the progress of the project.

e-Sci has a very capable Technical Architect in Ravi Kulkarni who has more than 30 years of experience in software and systems engineering, and is a veteran technologist. Ravi provided the guidance in choosing the technology infrastructure and various technologies that we used in development of the prototype.

Will Liu is a very savvy and experienced software engineer and ably performed the tasks of Frontend Developer, Backend Developer and also the DevOps Engineer.

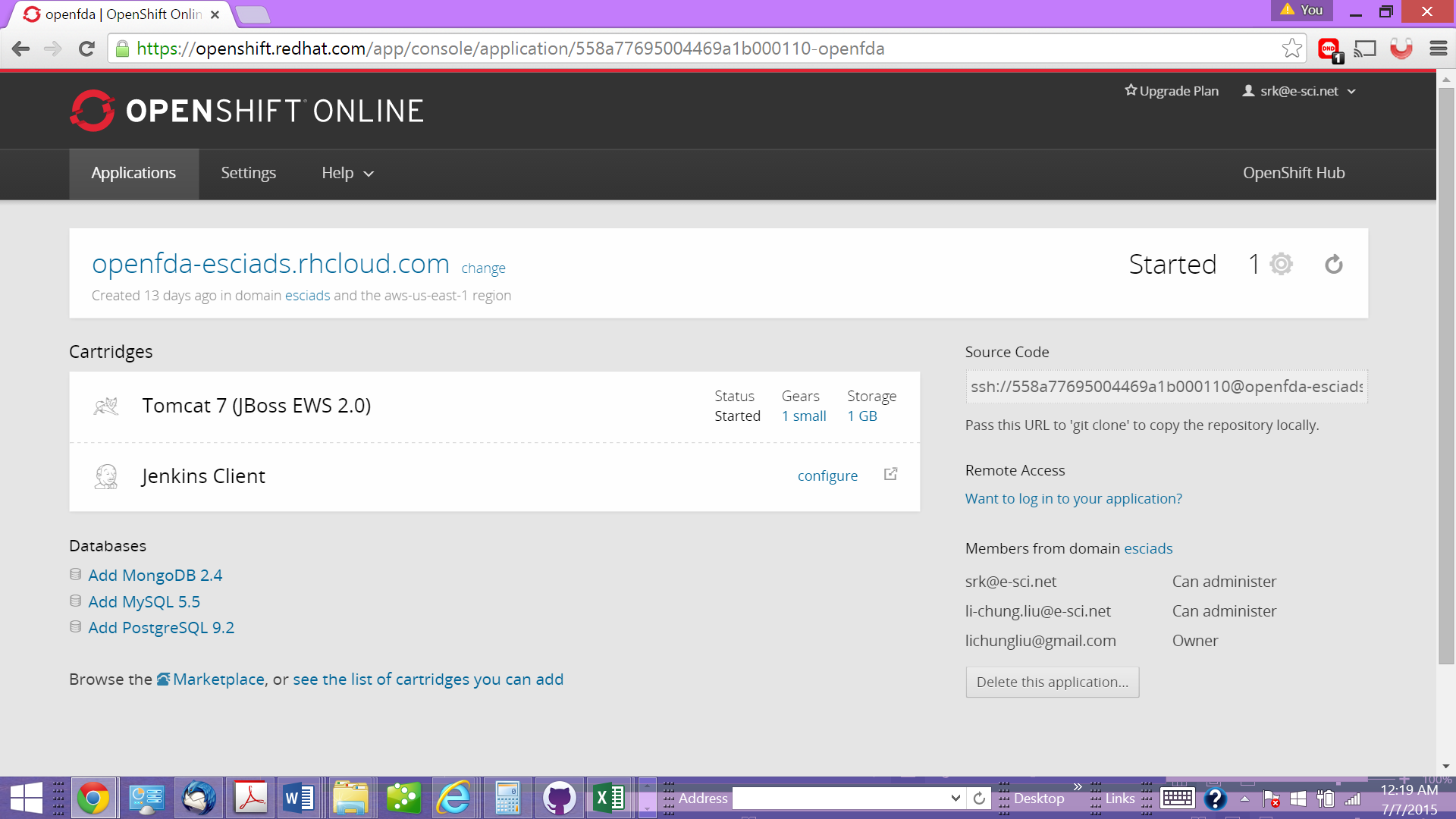
1. Five Modern and Open-Source Technologies

e-Sci team has been working on cutting edge technologies while performing on current government contracts and has up-to-date knowledge of the latest trends in technology. Below is a partial list of the modern and open-source technologies that we used while developing the Agile Delivery Services prototype:

* 1. The Hyper Text Markup Language (HTML5) - to mark-up text for web pages
  2. JQuery - Cross-platform JavaScript library with features to provide HTML manipulation and event handling.
  3. JavaScript - to generate the array containing query results which is then passed to Google charts API for further processing.
  4. Google Charts API - for displaying results obtained from queries to the FDA datasets, in the form of line charts and bar charts.
  5. Apache Tomcat - Open source J2EE container for deploying a web application.
  6. Red Hat OpenShift - Open source Platform-as-a-Service for hosting a web application in a cloud environment.

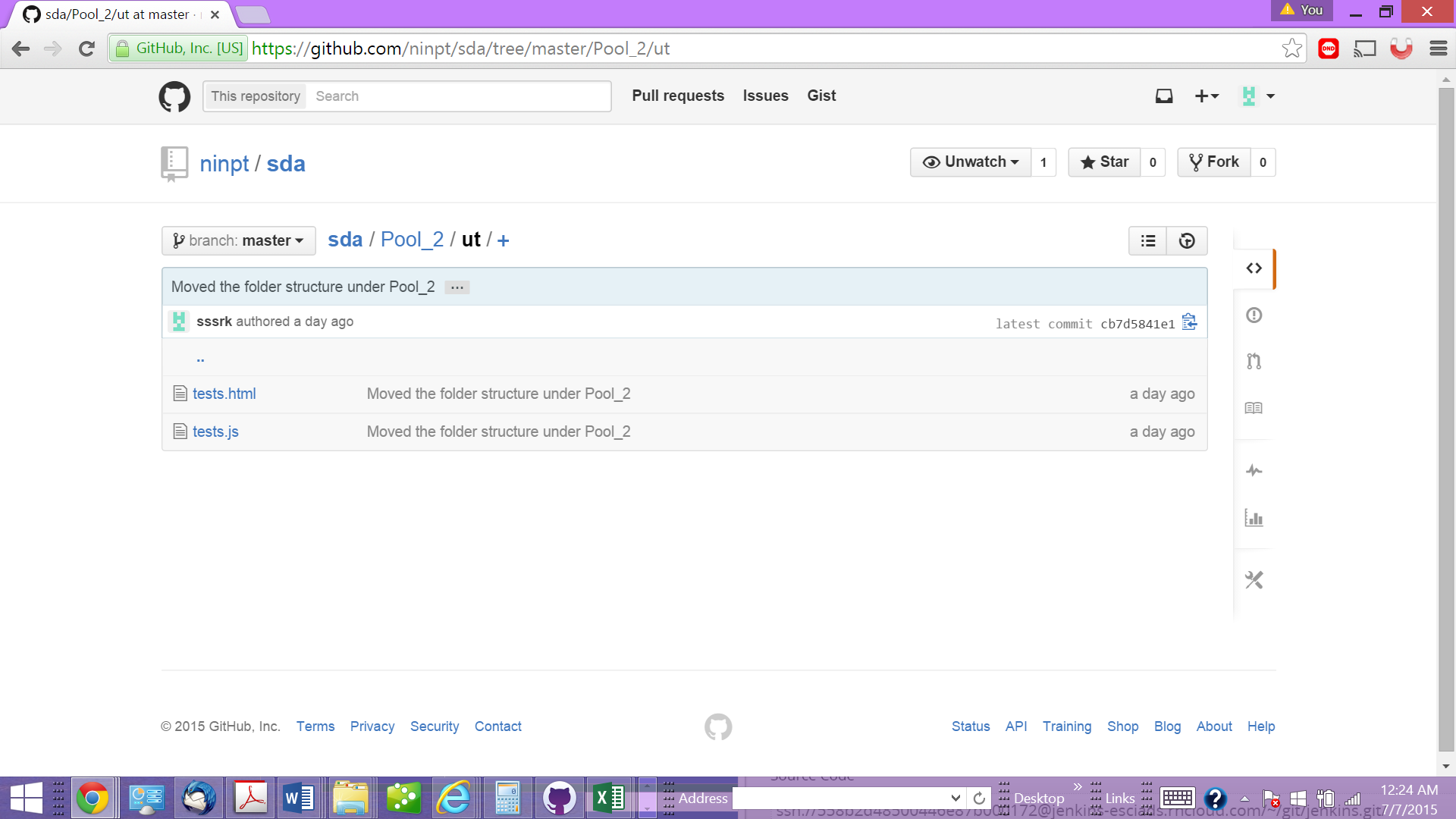
1. Deployed the Prototype on a PaaS Provider

e-Sci technical architect chose the Red Hat OpenShift Online Public PaaS service for deployment of the prototype web application. OpenShift allows users to host applications on a public platform while providing services such as automated provisioning, management and scaling of applications. It also works well with the GitHub online repository which facilitates deployment of applications to the OpenShift platform.



1. Unit Tests for Code

e-Sci team has had a vast experience and knows first-hand the importance of writing unit tests. Our software engineer used QUnit framework which is meant for unit testing JavaScript code. Since we used the public web services API provided by the FDA, we did not have to write complex Java code to build the prototype. The code was written using JavaScript, and QUnit was an appropriate framework for performing unit tests.



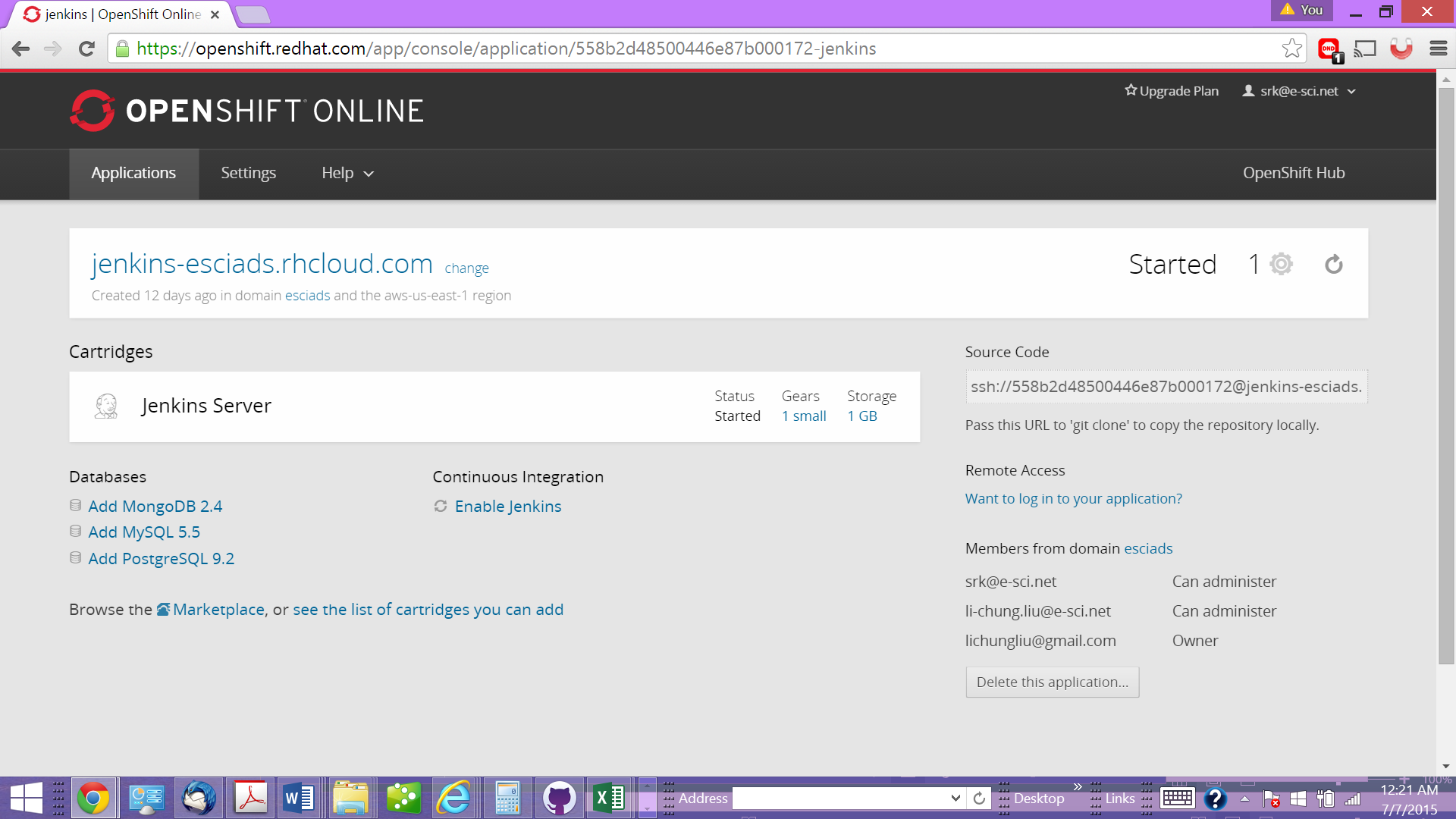
1. Set Up Continuous Integration System

One of the main reasons for choosing Red Hat OpenShift PaaS service was the various robust features provided by OpenShift. Red at OpenShift provides a plug-in for a full featured continuous integration (CI) server, Jenkins.

The setup for Jenkins involves the following steps:

1. Create Jenkins
2. Create/add an application with embedded Jenkins
3. Commit and push new code to your repository. Tis step triggers the build/test/deploy sequence in Jenkins.
4. Jenkins waits for this commit, runs a full series of tests.
5. With OpenShift, if the tests and build are successful, the new code gets deployed. If it fails, the old code continues to run with no downtime related to the push.
6. Users can review the persistent build history maintained by Jenkins.

Jenkins includes a web user interface that provides the ability to trigger builds, customize builds, manage resources, and manage plugins.



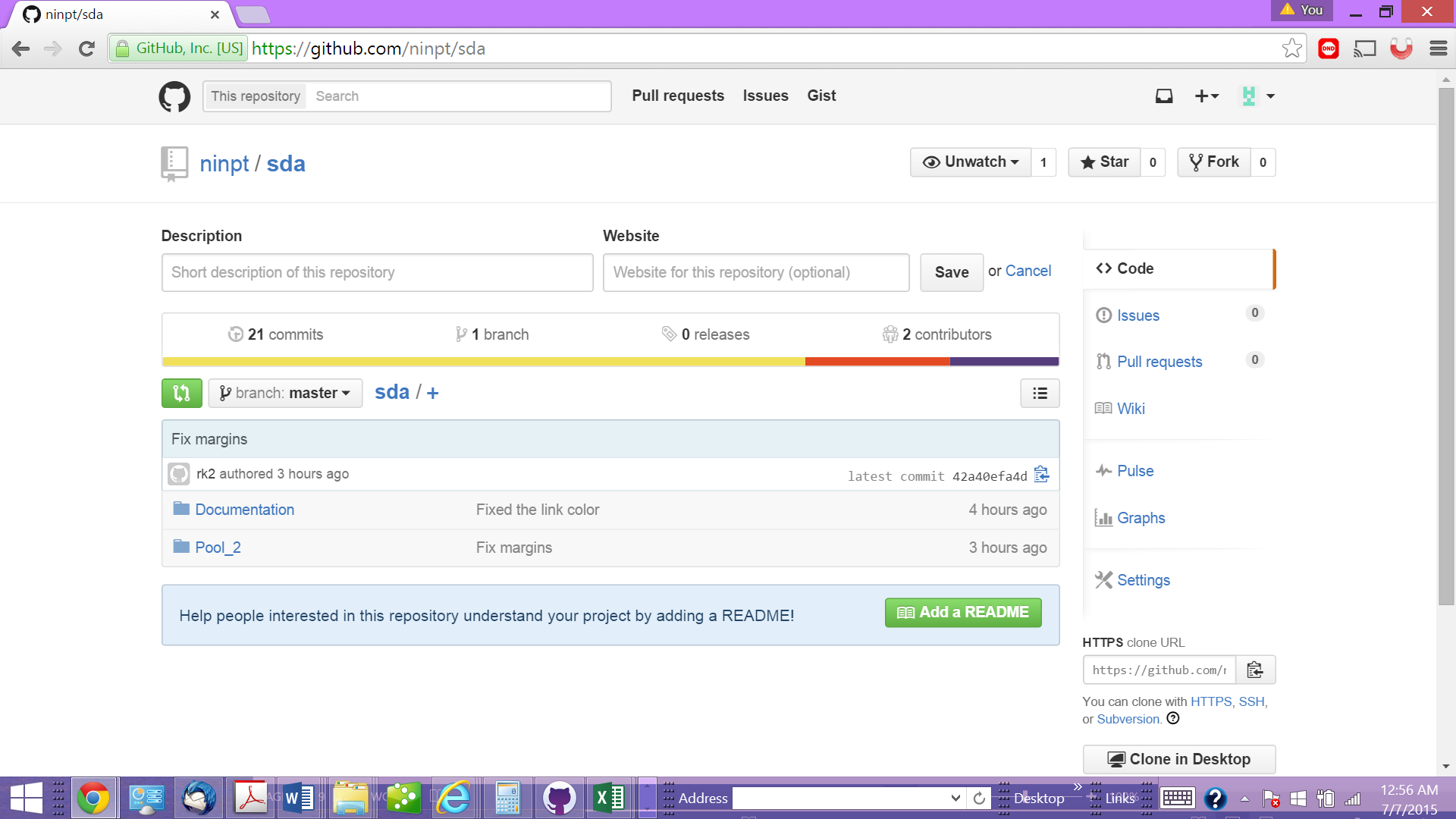
1. Set Up Configuration Management

e-Sci team has used various configuration management systems such as CVS, Rational ClearCase and Subversion during the performance of software engineering projects.

For GSA Agile Delivery Services prototype, e-Sci team set up a public GitHub repository associated with an organizational account for e-Sci. We then created a team for the GSA-ADS prototype. This structure allowed e-Sci team members to push and pull code and also deploy it to Red Hat OpenShift remotely.

The URL for e-Sci repository is: <https://github.com/ninpt/sda>





1. Set Up Continuous Monitoring

The Red Hat OpenShift infrastructure integrates with Zabbix, which is an availability and performance monitoring solution. Zabbix has an open-source version available via Extra Packages for Enterprise Linux (EPEL). Zabbix offers agent based monitoring, items, triggers, and action driven events. It provides system specific metrics such as CPU utilization, memory utilization, file system size, and system uptime.

Zabbix provides the following node monitoring features:

**Number of running apps -** This metric reports the number of applications running on the node. This is useful for capacity reporting and identifying over-utilized or under-utilized nodes.

**Node active capacity** - This metric reports how many gears are currently active out of the total possible number of gears that could possibly run on that node.

**Number of idled apps** - This metric reports back how many applications are idle on the node. This is helpful when trying to decide how much the node can and should be over-committed. **Number of new apps** - This metric shows the number of new applications on the node since last count. Setting a threshold at “X” number lets the administrator know when the node is close to exceeding capacity.

**Number of not started apps** - This metric reports the number of applications on the node that are not idled and not started.

**Number of total apps** - This metric reports the number of applications on the node that are either not started, started but idled, and running.

The stages of monitoring with Zabbix include the following:

1. Monitor some metric with a Zabbix agent and return some value to the Zabbix server.
2. Use triggers to define operational requirements as they relate to changes in the value.
3. If the value exceeds some given threshold, cause an event and take a defined action. This could be a self-healing mechanism like restarting a service or sending an email to an on-call administration.

To deploy this monitoring infrastructure follow these steps:

1. Deploy Zabbix Server
2. Configure Zabbix server
   1. Notifications
   2. Host Groups
3. Configure Zabbix templates
   1. Broker
   2. Node
   3. Broker Support Nodes
4. Deploy Zabbix Agent on Broker Support Nodes, Brokers, and Nodes.
5. Add hosts to Zabbix web based user interface and assign templates to hosts
6. Configure events

The configuration is discussed in detail at <http://www.redhat.com/en/files/resources/en-osen-openshift-enterprise-monitoring-paas-infrastructure.pdf>

1. Deploy Software in a Container

e-Sci team has deployed its GSA-ADS prototype on Red Hat OpenShift public PaaS service. Linux containers rely on Linux operating system-level isolation technology for running multiple isolated processes or “containers” on a single controller host. In case of OpenShift, these Linux Containers are called “Gears”, and are leveraged to deploy and run applications in a secure platform.

For deploying application source code, OpenShift integrates with Git. Developers push their application code to a Git repository running on their application “Gears”. Gears allow users to deploy multiple applications on a single Red Hat Enterprise Linux host “Node”. A typical OpenShift deployment may have tens or hundreds of Nodes, hosting thousands of Gears or more.

1. Iterative Approach

e-Sci team followed a human centered design process for the development of GSA-ADS prototype. Our design started with a discussion of the end users, their environment and the context (in this case the FDA datasets).

The users are involved in the analysis of requirements, design and presentation of wireframes and navigation. A short sprint of tasks and results produce a tangible product for user to see, evaluate and provide feedback on. The process is repeated in the following sprint with corrections and new features.

We used the human centered design tools such as “Persona”, “Scenario” and “Use Case”.

For this prototype, e-Sci team came up with a **persona**, “Jon”, who is the primary stakeholder or the end user of the prototype.

The scenario addresses the needs of the end user “Jon”. We had three **scenarios**:

1. Jon is a heart patient and has been receiving medication for hypertension. E is anxious to know the adverse effects of various drugs that are prescribed to treat hypertension.
2. Jon may have to go through surgery to have a pacemaker installed. He wants to know how many adverse reports for medical devices were filed in the last fifteen years, and how many were reported for Pacemakers.
3. Jon also wants to know the frequency of adverse reports for medical devices, which resulted in serious conditions, such as injury or death.

Our **use cases** were based on the above tree scenarios. Each of the scenarios led us to the queries that we would use to get the desired results from the FDA datasets.

Our agile approach is discussed in detail in the document titled “e-Sci\_AgileProcess.docx” in the location: <https://github.com/ninpt/sda/tree/master/Documentation>

1. Installation Documentation

For installation documentation, please refer to the file "ADS-Installation\_Deployment.docx" in the location:

<https://github.com/ninpt/sda/tree/master/Documentation/>

1. Openly Licensed and Free of Charge Platforms

Below is a list of technologies used by the e-Sci team in the development of GSA-ADS prototype.

* jQuery (MIT License - https://jquery.org/license/)
* Google Chart (Free for use - https://developers.google.com/chart/?csw=1)
* QUnit (JQuery Foundation License - https://jquery.org/license/)
* Apache Tomcat (Apache License 2.0 - <http://tomcat.apache.org/legal.html>)
* Red Hat OpenShift (https://www.openshift.com/legal/services\_agreement)

All of the above technologies are open source and free of charge.