AN EMPIRICAL STUDY OF USING CLOUD-BASED SERVICES IN CAPSTONE PROJECT DEVELOPMENT

Zhiguang Xu

Department of Computer Science, Valdosta State University, Valdosta, GA, USA
zxu@valdosta.edu

ABSTRACT

Cloud computing is gaining prominence and popularity in three important forms: Software as a Service, Platform as a Service, and Infrastructure as a Service. In this paper, we will present an empirical study of how these cloud-based services were used in an undergraduate Computer Science capstone class to enable agile and effective development, testing, and deployment of sophisticated software systems, facilitate team collaborations among students, and ease the project assessment and grading tasks for teachers. Especially, in this class, students and teachers could leverage time, talent, and resources collaboratively and distributedly on his/her own schedule, from his/her convenient location, and using heterogeneous programming platforms thanks to such a completely All-In-Cloud environment, which eliminated the necessity of spending valuable development time on local setup, configuration, and maintenance, streamlined version control and group management, and greatly increased the collective productivity of student groups. Despite of the relatively steep learning curve in the beginning of the semester, all nine groups of students benefitted tremendously from such an All-In-Cloud experience and eight of them completed their substantial software projects successfully. This paper is concluded with a vision on expanding and standardizing the adoption of the Cloud ecosystem in other Computer Science classes in the future.

KEYWORDS

SPI, Cloud Computing, Software Development, Capstone Project, Computer Science Education

1. INTRODUCTION

CS 4900, Senior Seminar, is a project-driven course designed to provide senior capstone experiences for graduating Computer Science (CS) majors at Valdosta State University (VSU). While oftentimes, students produce impressive applications, much of their efforts centre only on the task of coding such applications itself with rare, if any, concerns about how it fits into a much larger enterprise-level picture, how to factor in the real-world software production parameters such as bottom-line economics, control of data, security, compatibility, etc., and ultimately, the strategy to respond to the rapid migration toward cloud computing as the framework for most modern applications in today's industry [3]. Without a well-architected exposure to the cloud and an easy-to-follow procedure to take advantage of the services provided by it, such a disconnection between the academia and industry results in a great deal of students' valuable time being spent only on their local computers performing tedious installation and configuration of the software development environment and the quality test of their software products. Moreover, as the complexity of the applications they build scales up, it is a challenging undertaking for the students to conduct efficient and effective team collaborations when they work distributedly and for the instructor to help the students during the semester and evaluate individual student's performance at the end of it.

In spring of 2016, we addressed this problem head on in Senior Seminar. Thirty students in this class formed nine groups to build a full-fledged Web server application completely in cloud. The application was called FriendsNextDoor, a private social network service that allows users to connect with people who live in their neighbourhood, and it was mainly written in Ruby on Rails. CS faculty at VSU including the instructor of the class attended the project demonstrations at the end of the semester. They concluded that eight out of nine student groups completed their projects with an "excellent" overall quality. Homepage of one of such projects, LiveTogether as they named it, is show in Figure 1 below. In addition, the end-of-semester survey indicated that 86% of students strongly agreed that "the all-in-cloud programming ecosystem used in the capstone project gave them a great exposure to what it means to work in a professional context".

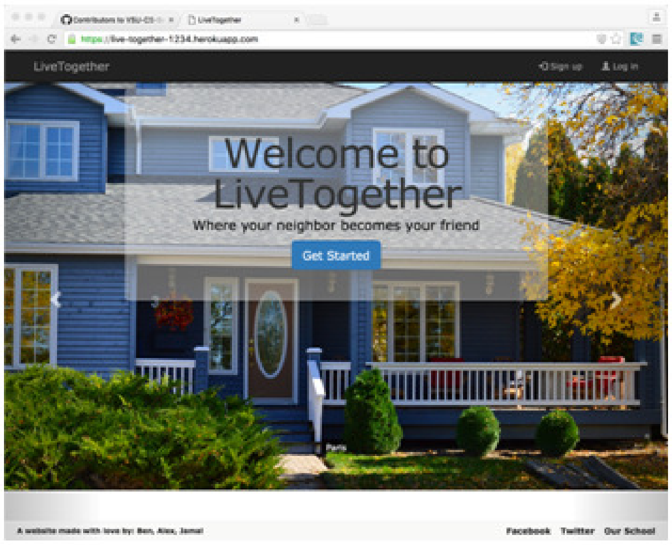

LiveTogether
https://live-together-1234.herokuapp.com
LiveTogether
Sign up
Log in
Welcome to LiveTogether
Where your neighbor becomes your friend
Get Started
A website made with love by: Ben, Alex, Jamal
Facebook
Twitter
Our School

Figure 1. Homepage of a Student Capstone Project

SPI is an acronym for three popular IT paradigms under the umbrella notion of cloud computing: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). In Section 2, we briefly present how SPI has evolved in the recent years, in particular, the opportunities and challenges they brought to the Computer Science education. Then, in section 3, we outline the requirements of the capstone project that students were asked to develop in the Senior Seminar class. Then we present a big picture along with one detailed configuration example of how the SPI ecosystem was actually exploited to fulfil such requirements with an emphasis on the All-In-Cloud (AIC) programming settings. This is followed by section 4, where seven specific SPI services/products are selected to elaborate the procedures of integrating them into our capstone projects in technological details. In Section 5, we evaluate the final products of the capstone projects based on the assessments from both the students' and CS faculty's perspectives. Finally, in Section 6, we summarize the paper with conclusions and a vision on improving the AIC-SPI model and the expansion of adopting it in other CS classes in the future.

2. COMPARING TRADITIONAL IT AND CLOUD-BASED SPI

National Institute of Standards and Technology (NIST) defines Cloud Computing as Internet-based technology, which offers computational resources such as large storage capacity, high network bandwidth, and vast processing power via a computer network and delivers flexible, scalable, and on-demand services to the end-users [1]. In comparison to traditional IT, such resources and services are classified into three "SPI" layers as shown in Figure 2 below, each serving different purposes and tasks. Infrastructure as a Service (IaaS) delivers physical or (more often) virtual machines and other resources such as servers, storage, and networking connectivity, for example, Amazon Simple Storage Service (S3). The Platform as a Service (PaaS) layer acts as a container enabling the users to modify and develop their platform and deploy their applications. Typical example at this layer is Heroku. The most abstract layer is Software as a Service (SaaS), which allows users to run hosted applications on the Cloud and use them through a web browser remotely, e.g. Cloud9, an online IDE with full Ubuntu workspace. Besides others, all three examples mentioned above were heavily used in our Senior Seminar capstone projects.

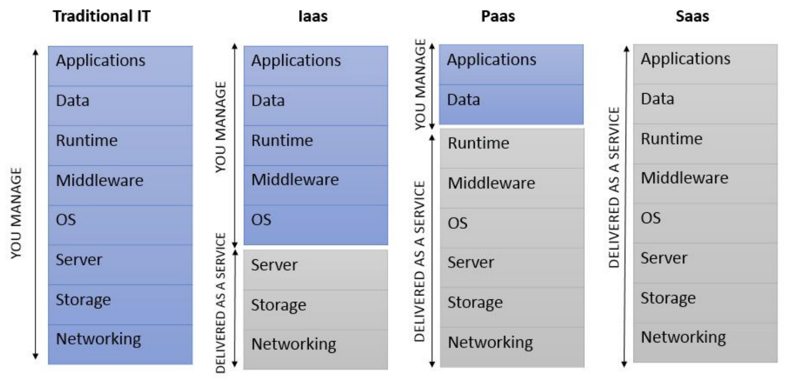


Figure 2. Traditional IT vs. SPI

Cloud computing in less than a decade has gone from utility computing with nearly unlimited on-demand scalability at very attractive prices, to the industrialization of IT that has been pioneered by born-to-the-cloud companies like Amazon, Google, and Salesforce, to the total transformation to digital business [2] as nicely summarized in a recently released report State of the Market: Enterprise Cloud 2016 from Verizon Communications Inc. – "As cloud increasingly becomes the norm … it is not enough to think cloud first. You need to think cloud only."

In the context of CS education that is inherently and tremendously impacted by the Cloud, technologically speaking, the advantages of SaaS, PaaS, and IaaS solutions are simplicity of integration (most of the time, students need only a web browser), reduced cost (the code/data center resides within the cloud rather than local), and scalability (resources are dynamically and transparently allocated in response to the fluctuation in data and computation sizes). Among others, there are two major challenges of adopting SPI – data security and "lock-in" to the products of a particular provider [3, 4]. During the development of the capstone projects in senior seminar, we encountered both of such issues, which will be discussed more in sections 4 and 5.

3. USE CLOUD-BASED SPI IN CAPSTONE PROJECT DEVELOPMENT

3.1. Capstone Project Requirements

Consider FriendsNextDoor a modern, more attractive, more secure, and more versatile version of a community email list service or Yahoo Groups, the popular message board, where regular homeowners, community leads, Home Owner Associations, local businesses, public agencies, and other neighbourhood constituencies can post neighbourhood news, offer items for sale, seek/provide babysitting opportunities, lend/borrow yard tools, ask for help finding lost pets, or organize a block party, just to name a few. Although it was an open ended project, the following list of features were required:

- For any potential user, signing up to the web site should be subject to some sort of verifications, e.g. location/address, invitation, approval by the community leads, or a combination of them.
- Users are of different roles with different levels of privileges/responsibilities. Such roles at least include regular users, community leads, local businesses, public agencies, and system admin.
- Users sharing common interests can form, join, and leave groups.
- Two types of message/post boards: public board and group-specific private board. And optionally, a real-time chat system.
- Events section where event organizers and participants can share information (e.g. through calendar, map, photo album, etc.)
- Community leads should be elected through a point system. They cannot be simply appointed.
- Optionally, a RSS feed publish/subscribe system to share information among multiple neighborhoods.

3.2. A High-level View

In order to provide a high-level view and a global picture of how the AIC-SPI ecosystem fits in our capstone projects, a visualization of the most popularly used SPI services in today's software ecosystem that is custom-rendered on https://modeanalytics.com/ is displayed in Figure 3 below.

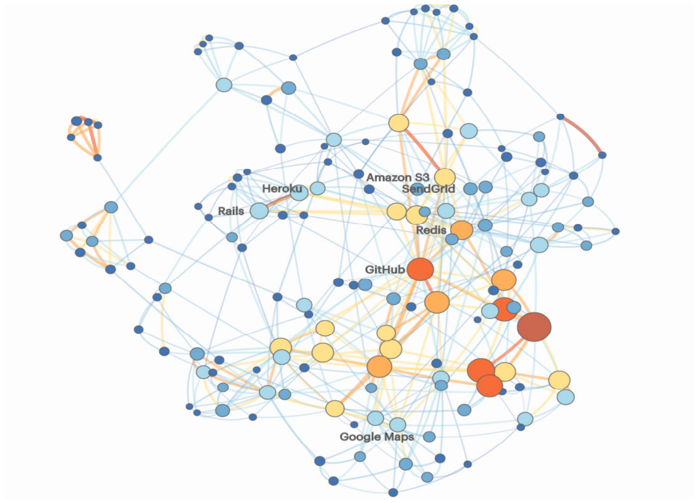


Figure 3. The SPI Ecosystem

Dots represent software services/products, and the lines between them represent companies using both services they connect. The bigger and redder the dot, the more companies use the service. The thicker and redder the line, the more companies use two services together. Particularly, those labeled dots in Figure 3 were the SPI services/products that were actually chosen to be used in the FriendsNextDoor projects by our Senior Seminar students themselves. Such services belonging to different layers in the AIC-SPI model as shown in Figure 4 were integrated into FriendsNextDoor through respective Rails "gems" (i.e. third-party libraries). As an example, one of such gems named fog for Amazon S3 is going to be discussed in the next section to show how such integration was configured at some technically low level.

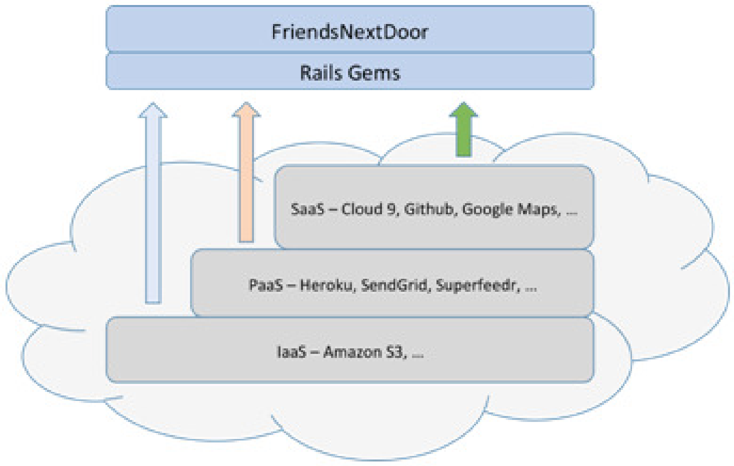

FriendsNextDoor
Rails Gems
SaaS – Cloud 9, Github, Google Maps, ...
PaaS – Heroku, SendGrid, Superfeedr, ...
IaaS – Amazon S3, ...

Figure 4. The SPI Layers

3.3. An Example of SPI Configuration at Low-level

Rails leverages a wide set of third-party libraries, most of which are released in the form of a "gem". A few selected Rails gems commonly used in our FriendsNextDoor application along with some brief descriptions are listed in Table 1 below.

Table 1. Selected Gems used in FriendsNextDoor

Gem Name	**Description**
fog	A powerful cloud services gem
jquery-rails	A JavaScript library
bcrypt	Securing password
bootstrap-sass	A sass-powered version of Bootstrap
devise	User authentication
pundit	User authorization
sqlite3	DBMS for SQLite
pg	DBMS for PostgreSQL
carrierwave	Uploading files
gmaps4rails	Google Maps solution for Rails
simple_calendar	A calendar render
figaro	A Heroku-friendly Rails app configuration tool
websocket-rails	Websocket support in Rails
forem	A Rails forum engine

In particular, the first gem on the list, fog, was used to switch the file uploader in our project (built upon another gem carrierwave) from using regular file storage that suffers from suboptimal efficiency to using Amazon S3, one of the most popular cloud storage providers. Sensitive information such as access keys and passwords can be configured in an automatic and secure fashion via yet another gem of figaro.

Specifically, thanks to the fact that in the current version of fog, all fog providers are getting separated into meta-sub-gems (e.g. for-aws, fog-google, and fog-vsphere, etc.) to lower the load time and dependency count, the only fog gem that we actually needed to install was fog-aws. The configuration of the fog gem turned out to be very simple and straightforward. It only involved two files `config/initializers/carrierwave.rb`:

```
if Rails.env.production?
  CarrierWave.configure do |config|
    config.fog_credentials = {
      # Configuration for Amazon S3
      :provider              => 'AWS',
      :aws_access_key_id     => ENV['S3_ACCESS_KEY'],
      :aws_secret_access_key => ENV['S3_SECRET_KEY']
    }
    config.fog_directory     =  ENV['S3_BUCKET']
  end
end
```

and `app/uploaders/picture_uploader.rb`:

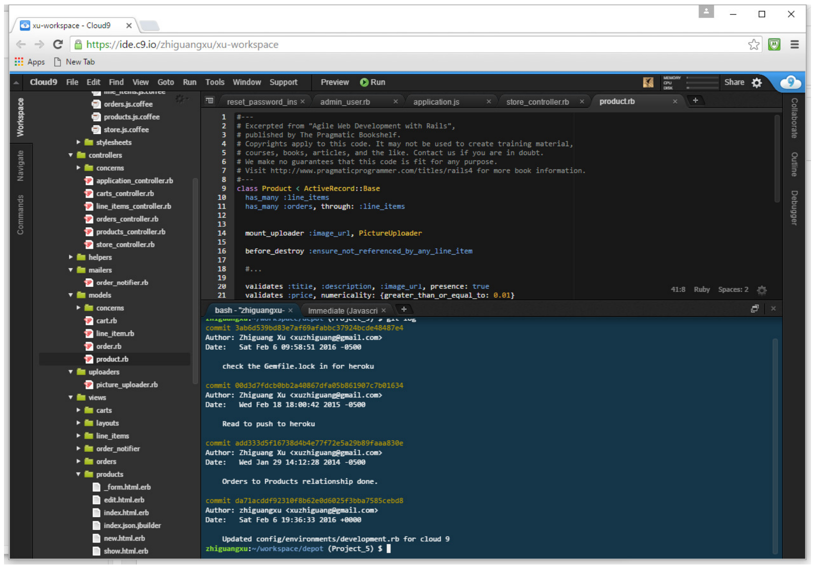
```
# Choose what kind of storage to use:
if Rails.env.production?
    storage :fog
else
    storage :file
end
```

Other SPI services/products were integrated into the FriendsNextDoor application in a similar fashion.

4. AIC-SPI IN ACTION

4.1. Cloud9 – Online IDE with Full Ubuntu Workspace

Cloud9 as shown in Figure 5 provides a hosted development environment to which you get access from a web browser. A cloud based IDE has been on everyone's mind since the concept of asynchronous web applications started taking a hold among developers as well as CS educators. However, it is Cloud9 as a key component in our AIC-SPI ecosystem that makes it truly "all in cloud". Unlike Linux virtual private servers from sites such as Linode, the Cloud9 service is free and the system is maintained for us so we will spend more time as a developer and less time as a sysadmin.



```
#---
# Excerpted from "Agile Web Development with Rails",
# published by The Pragmatic Bookshelf.
# Copyrights apply to this code. It may not be used to create training material,
# courses, books, articles, and the like. Contact us if you are in doubt.
# We make no guarantees that this code is fit for any purpose.
# Visit http://www.pragmaticprogrammer.com/titles/rails4 for more book information.
#---
class Product < ActiveRecord::Base
  has_many :line_items
  has_many :orders, through: :line_items


  mount_uploader :image_url, PictureUploader

  before_destroy :ensure_not_referenced_by_any_line_item

  #...

  validates :title, :description, :image_url, presence: true
  validates :price, numericality: {greater_than_or_equal_to: 0.01}
```

```
commit 3ab6d539bd83e7af69afabbc37924bcde48487e4
Author: Zhiguang Xu <xuzhiguang@gmail.com>
Date:   Sat Feb 6 09:58:51 2016 -0500

    check the Gemfile.lock in for heroku

commit 00d3d7fdcb0bb2a40867dfa05b861907c7b01634
Author: Zhiguang Xu <xuzhiguang@gmail.com>
Date:   Wed Feb 18 18:00:42 2015 -0500

    Read to push to heroku

commit add333d5f16738d4b4e77f72e5a29b89faaa830e
Author: Zhiguang Xu <xuzhiguang@gmail.com>
Date:   Wed Jan 29 14:12:28 2014 -0500

    Orders to Products relationship done.

commit da71acddf92310f8b62e0d6025f3bba7585cebd8
Author: zhiguangxu <xuzhiguang@gmail.com>
Date:   Sat Feb 6 19:36:33 2016 +0000

    Updated config/environments/development.rb for cloud 9
zhiguangxu:~/workspace/depot (Project_5) $
```

Figure 5. Cloud 9 IDE

In senior seminar, the following features that Cloud9 provides are highlighted:

- Cloud9 offers complete and seamless Git integration with services like Github and Bitbucket. Students were able to login to Cloud9 using their Github accounts, pull and push code in between, deploy applications to online host such as Heroku, and collaborate with teammates through live coding.

- The virtualized backends allowed students to spin up instances of servers (which is always up to date and compatible with the latest Rails) when load increased.

- Cloud9 was a lifesaver (as students in senior seminar called it) for PC users since Rails and most of the related SPI services/products are not necessarily Windows-friendly.

- The Ubuntu terminal access to command line functionalities was particularly appropriate for some students who wanted to learn UNIX but might be overwhelmed by the effort required to install and configure Ubuntu Linux.

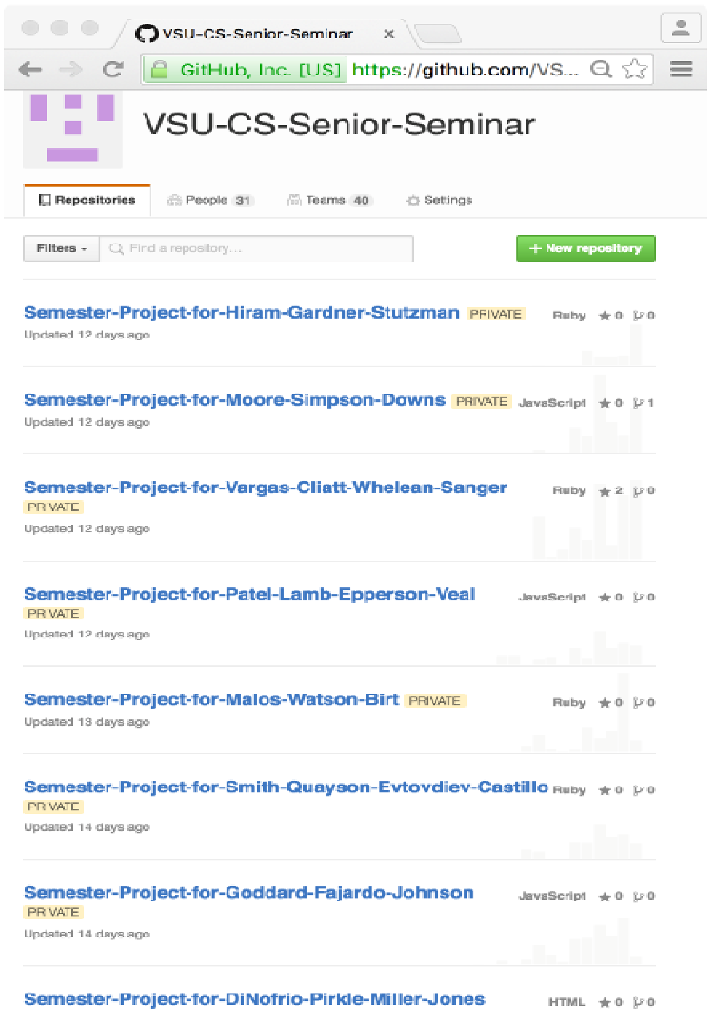
4.2. Github – Git Repository Hosting Service

Github was chosen as the online Git repository-hosting site for senior seminar due to the following reasons:

- Free private repositories, thanks to Github's educational program that allow students and teacher to securely access their projects from anywhere at any time [5]. We created an Organization "VSU-CS-Senior-Seminar" on Github that has 31 members (30 students and 1 teacher) and 40 private repositories (30 for student individual projects, 9 for group projects, and 1 for the teacher, see Figure 6).

- Secure source code backup in the Cloud (The proved important when one student's laptop crashed in the middle of the semester and it was his backups on Github that saved his project).

- Seamless integration with Cloud9.



VSU-CS-Senior-Seminar

Semester-Project-for-Hiram-Gardner-Stutzman PRIVATE Ruby ★ 0 0
Updated 12 days ago

Semester-Project-for-Moore-Simpson-Downs PRIVATE JavaScript ★ 0 1
Updated 12 days ago

Semester-Project-for-Vargas-Cliatt-Whelean-Sanger PRIVATE Ruby ★ 2 0
Updated 12 days ago

Semester-Project-for-Patel-Lamb-Epperson-Veal PRIVATE JavaScript ★ 0 0
Updated 12 days ago

Semester-Project-for-Malos-Watson-Birt PRIVATE Ruby ★ 0 0
Updated 13 days ago

Semester-Project-for-Smith-Quayson-Evtovdiev-Castillo PRIVATE Ruby ★ 0 0
Updated 14 days ago

Semester-Project-for-Goddard-Fajardo-Johnson PRIVATE JavaScript ★ 0 0
Updated 14 days ago

Semester-Project-for-DiNofrio-Pirkle-Miller-Jones HTML ★ 0 0

Figure 6. Repositories on Github

- Clean and fast submission and grading of projects, especially when their sizes went beyond megabytes.

- Rich tools for administrating student groups, visualizing students' contributions to their group projects, archiving projects for future course assessments, and much more [5]. Grading individual student based on his/her contributions to a group programming project like the ones we had in senior seminar has always been a challenge for the CS teachers. Thanks to the graphs provided by Github where each group member's commits are chronically tracked as shown in Figure 7, along with other traditional project deliverables such as group report, workloads could be distributed more balanced among group members, "free riders" could be identified more early-on, and grades could be assigned more fair and reasonable.

- Source code and tutorials of most of the third-party libraries (i.e. Rails gems) are housed on Github.

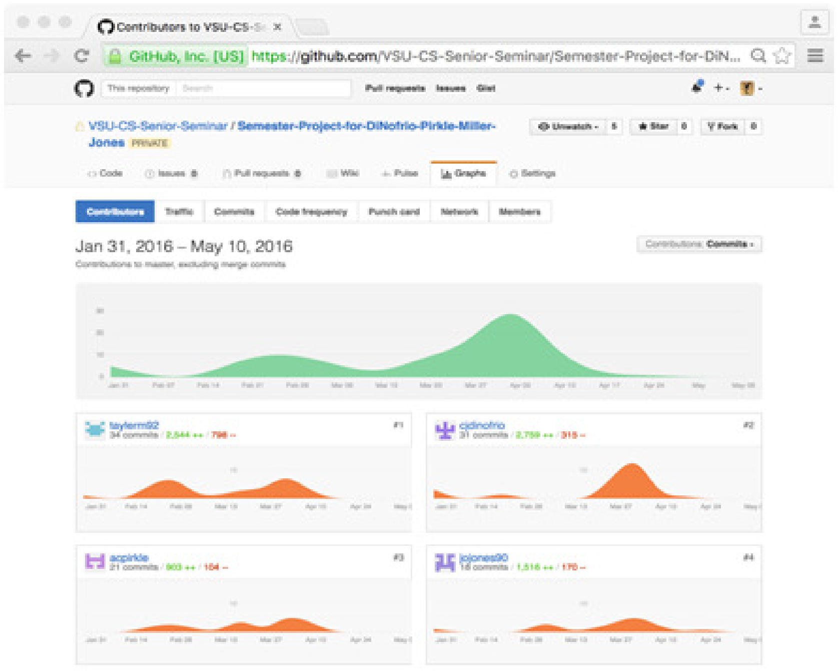

Contributors to VSU-CS-Se
GitHub, Inc. [US] https://github.com/VSU-CS-Senior-Seminar/Semester-Project-for-DiN...
VSU-CS-Senior-Seminar / Semester-Project-for-DiNofrio-Pirkle-Miller-Jones PRIVATE
Code Issues Pull requests Wiki Pulse Graphs Settings
Contributors Traffic Commits Code frequency Punch card Network Members
Jan 31, 2016 – May 10, 2016
Contributions to master, excluding merge commits
taylerm92
cjdinofrio
acpirkle
jojones90

Figure 7. Commits Graph from Individual Students

4.3. AWS S3 – Cloud Storage

The file/image uploader via a third-party gem named “carrierwave” is good enough for preliminary development and testing on Cloud9, however, it uses the local file system (public/uploads) for storing the files/images, which isn’t a good and efficient practice and produces lots of headaches in production on Heroku. Therefore, we used one of the most popular and well-supported cloud storage, Amazon Simple Storage Service (S3) to store files/images separately from our application server. Details on the integration of Amazon S3 into our applications were discussed in section 3.1 above.

4.4. Sendgrid – Email Delivery and Management System

Instead of using a private email service provider such as Microsoft Outlook and Gmail which would cause problems of being locked in to these specific products, we outsourced the email delivery and management to SendGrid, an in-cloud transactional email service provider, to send potentially mass emails. In addition to better throughput when handling large volume of emails, other benefits of using SendGrid include better deliverability, professional level reliability, and transparent performance analytics.

SendGrid makes the email configuration fairly easy. The file below (`config/environments/production.rb`) needs to be updated for running the project on Heroku. Similar to what was discussed for Amazon S3, environment variables SENDGRID_USERNAME and SENDGRID_PASSWORD can be encrypted using the Figaro gem for better security purposes.

```
 config.action_mailer.raise_delivery_errors = true
config.action_mailer.delivery_method = :smtp
host = 'your-project-name.herokuapp.com'
config.action_mailer.default_url_options = { host: host }
ActionMailer::Base.smtp_settings = {
  :address        => 'smtp.sendgrid.net',
  :port           => '587',
  :authentication => :plain,
  :user_name      => ENV['SENDGRID_USERNAME'],
  :password       => ENV['SENDGRID_PASSWORD'],
  :domain         => 'heroku.com',
  :enable_starttls_auto => true
}
```

4.5. Google Maps – Location Based Services from Google

Several groups implemented a location based user verification system via Google Maps API by showing a visual verification of a user's neighbourhood as a superimposition of the neighbourhood's outline on the map (Figure 8).

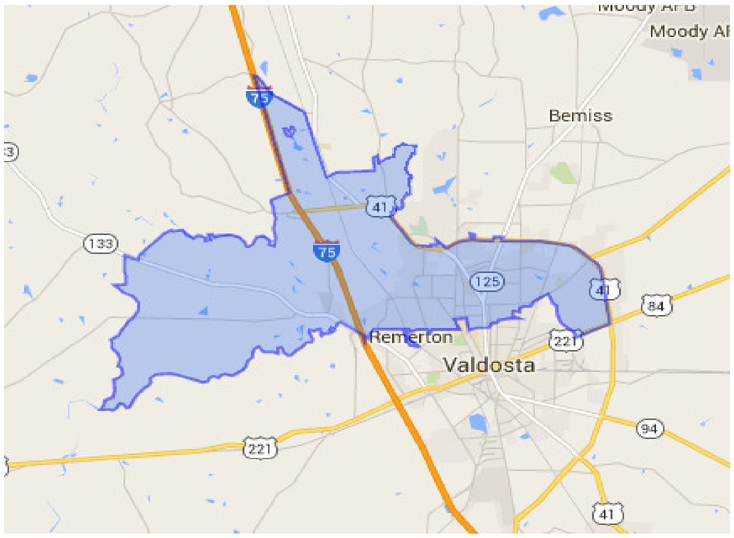


Figure 8. Visual Verification of Neighborhood

Each outline was stored in a kml file that stored Google Maps compliant polygons in xml syntax. The complexity came when accommodating every zip code in the United States. These approximately 30,000 kml files were parsed from a single large file (totalling 1.4 million lines of code) using a small Java program. Each kml file was then independently hosted providing their FriendsNextDoor application with embedded Google Maps based services without inducing a prohibitively long delay in load or draw times (Figure 9).

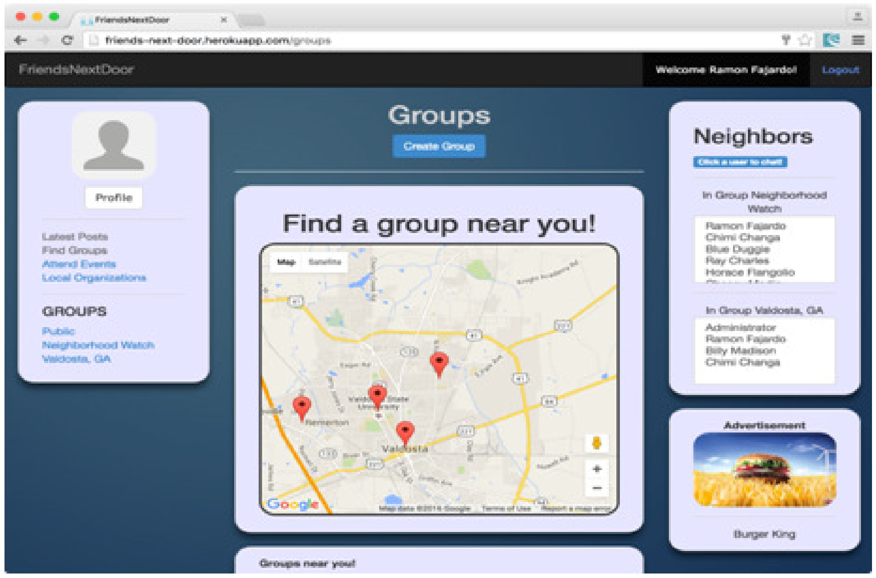


Figure 9. Location Based Service in a Student Project

4.6. Superfeedr – Hub for RSS Feed Publication and Subscription

In order to facilitate cross-site information sharing and dissemination, a Rich Site Summary (RSS) feed publication/subscription system was built in the FriendsNextDoor site developed by one group of three students as their honours option project.

Pubsubhubbub (PuSH) is an open protocol for distributing content from RSS publishers to subscribers that improves upon the RSS specification by adding an intermediate server to serve as a “hub” server at https://superfeedr.com which assists both the subscriber and the publisher with RSS communications.

Specifically, the hub eases the load of the publisher by establishing a subscription system and handing all subscription requests to the publisher’s feed, reducing the number of calls to access the publisher’s feed to only the occasional calls by the hub to fetch the RSS page check it for updates. The hub also eases the burdens of the subscribers by not requiring them to have to frequently request for any potential updates. Instead, subscribers can subscribe to a publisher’s feed by sending an HTTP POST request to the hub containing the publisher’s feed URL and a URL for the subscriber’s webhook – a URL that routes to a callback function running on the subscriber’s server, telling it to process that HTTP request body in a specific way. The webhook will “listen” for updates from the hub and store any updates it receives in the subscriber’s database. In this way, when a publisher makes changes to their RSS feed, the hub will “push” notifications containing the new RSS page to all of the subscribers, rather than having them constantly probe for new information (Figure 10).

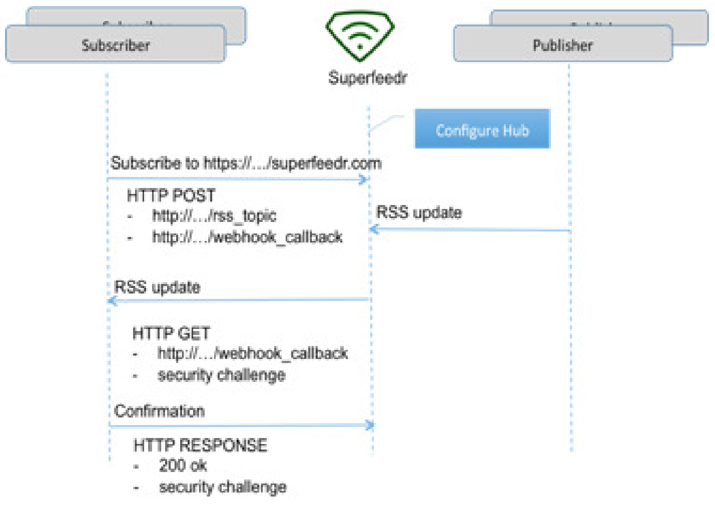


Figure 10. PuSH and the Hub Server

4.7. Heroku – Cloud PaaS

After developing a Rails application locally on Cloud9, the next step is to deploy it online. Heroku is a popular PaaS hosting service that is free to start using. Typically, in a Ubuntu terminal that is built into Cloud9, the deployment process involves running a sequence of command-line commands and scripts such as git-push the source code of the application to Heroku, install relevant gems and their dependencies, create and pre-seed database tables with certain data, configure and encrypt environmental variables, and fire up separate servers for various purposes. In Figure 11, three of such servers in addition to the Rails server are shown: a Solr server for site-wise searches, a Faye server for live chattings, and a Redis server for real-time RSS feed updates.

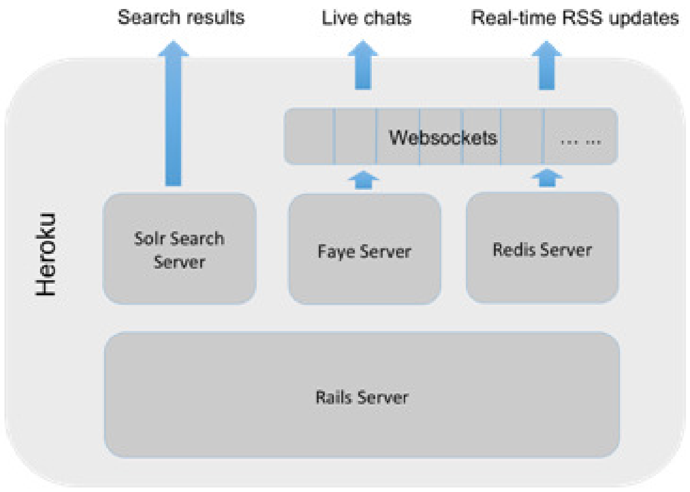


Figure 11. Rails and Other Servers Running on Heroku

Specifically, as a continuation discussion of the RSS feed feature presented in section 4.6 above, a major issues faced in the implementation of that feature was live pushing of updated feed data once it was received by the subscriber, parsed, and stored in the subscriber's database. Although each subscriber could receive updates from the publisher's feed, each browser client could not see any changes in their web page until they refreshed the page. This issue was solved using Web Sockets, which provide a connection for communication between the subscriber server and the browser clients currently viewing a page from the subscriber server, allowing the subscriber to send updated data that it received from the hub to each browser client. However, as the number of clients accessing the server increases, the amount of processing needed to be done by the Web Socket increases, slowing down access times substantially. To aid in this process, we used Redis, an in-memory hash, to store subscriber-client connection information, significantly reducing the read/write times for the Web Socket and allowing it to continue functioning quickly for a larger number of clients.

5. PROJECT ASSESSMENT

5.1. Faculty Assessment

After attending the final project demonstrations given by the students in senior seminar, CS faculty at VSU concluded that eight out of nine (or 89%) student groups completed their capstone projects with an "excellent" overall quality (see the detailed faculty assessments in Table 2). These projects not only met all the requirements listed in section 3.1 with elegant and professionally looking user interfaces, but also provided quite a few additional features such as live chatting, full-fledged searching, RSS based cross-site information sharing, and location based services, etc.

Table 2. Faculty Assessment

Assessment Item	**Agree**
Students have designed, implemented, and evaluated a computer-based system, process, component, or program to meet desired needs	87%
Students have demonstrated ability to use current techniques, skills, and tools necessary for computer practice	91%
Students have applied mathematical foundations, algorithmic principles, and computer science theory in the modelling and design of computer-based systems in a way that demonstrates comprehension of the trade-offs involved in design choices	90%

5.2. Student Assessment

The end-of-semester survey completed by all students in senior seminar indicated that 86% of them strongly agreed that "the all-in-cloud programming ecosystem used in the capstone project gave them a great exposure to what it means to work in a professional context" and all of them strongly agreed or agreed that "overall speaking, working in the capstone project better prepared them for a career in software development". Additionally, the survey asked students to vote for two services/products provided in the SPI ecosystem that benefited them the most. The result is shown in Figure 12 below where the number of student votes for a specific service/product is marked as its "popularity".

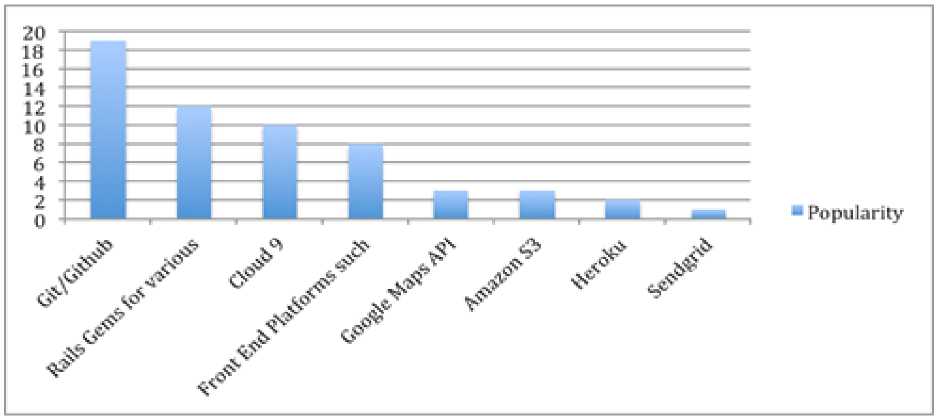


Figure 12. Cloud-based Services/Products that benefited the students the most

5.3. Issues

There were two major issues that were brought to our attention while we appreciated the agility and effectiveness of the project development in senior seminar thanks to the AIC-SPI ecosystem – security and inadequate testing.

Data and code security cloud computing in general is one of the top issues that everyone concerns. We are not exceptional. Two months after the instructor created buckets on Amazon S3 for storing images for his own Rails projects, hackers comprised his Amazon access keys. Fortunately, Amazon caught unauthorized activities and resolved the issue. Further investigations indicated that these access keys were leaked on Github where accidentally, they were made accessible to the public for a short period of time. As an alarm, the vulnerability of the AIC-SPI ecosystem was emphasized to the students in senior seminar before it had further widely spread.

A widely cited 2002 study prepared for NIST reported that 50 percent of software development budgets go to testing [6]. However, flaws in software still cost the U.S. economy $59.5 billion annually. Unfortunately, due to time constraints, in senior seminar, we didn't spend as much time as we should on software testing including unit testing, although it is such an important built-in component in Rails.

6. CONCLUSION AND FUTURE WORK

Our experience in developing capstone application with the AIC-SPI ecosystem has been very positive. We have seen senior students in the capstone class voluntarily and comfortably use such an ecosystem in other projects. SPI and cloud computing in general give them unique opportunities and exposures to collaborative, distributed, and real-world practices that are prevalent in today's software development industry and community. The experience and competitive skills gained in CS 4900 will scale with students and enable them to collaborate with their peers, contribute to open source software projects, and eventually transfer their new knowledge to the work environment in the future. It also streamlines the teacher's tasks of grading student projects and giving lectures.

The proposed AIC-SPI ecosystem takes advantage of the intimate relationship that exists between the cloud technologies and CS courses [1]. Due to this unique attribute, it can penetrate into all layers of the Cloud and provide meaningful assistance for students in a wider range of CS classes.

Therefore, future works include expanding the adoption of the AIC-SPI ecosystem in other Computer Science classes where students' programming skills are emphasized.

As instructors in computer science departments we are preparing people to develop software. If testing is 50% of the effort, we are not properly preparing our students if we do not include software testing in the curriculum. Therefore, another area to work on in the future is to invest more time in teaching students software testing.

REFERENCES

[1] H. Rajaei and A. Aldakheel, "Cloud Computing in Computer Science and Engineering Education," in 2012 American Society for Engineering Education Annual Conference, San Antonio, TX, June 17-20, 2012.

[2] P. Kumar, S. Kommareddy, and N. Rani, "Effective Ways Cloud Computing Can Contribute to Education Success," Advanced Computing: An International Journal (ACIJ), Vol.4, No. 4, July 2013.

[3] R. Roggio, "Cloud Computing for Capstone Software Development Courses," 2011 Information Systems Educators Conference (ISECON) Proceedings v28-n1692, Wilmington, North Carolina, Nov 2011.

[4] O. Akin, F. Matthew, and D. Comfort, "The Impact and Challenges of Cloud Computing Adoption on Public Universities in Southwestern Nigeria," International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 5, No. 8, 2014.

[5] Z. Xu, "Distributed Student Software Project Management with Git," the 7th International Multi-Conference on Computing in the Global Information Technology (ICCGI 2012), p.p. 159-164, ISBN 978-1-61208-202-8, Venice, Italy, June 24-29, 2012.

[6] G. Tassey, "The Economic Impacts of Inadequate Infrastracture for Software Testing", NIST Planning Report 02-3, May 2002.