Agenda

- Refresher
- Why Cloud First?
- What’s Happenin’?
- What’s Next?
- UAccess Research Case Study
What is it?
Why should you want one?

- Increased worker productivity
  - Agility
  - Flexibility
  - Speed of deployment
- Pay only for what you used
- Eliminate capital costs
- Reduced operating costs
- No data center to manage
Why Cloud First?

If you want to increase innovation, you have to lower the cost of failure. – Joi Ito

MIT
Why Cloud First?

- Demand for IT services continues to increase.
- Discover new ways to deliver IT services.
- Funding and staffing needs remain constant.

UA GOALS:
- Responsive IT
- Reduce risk
- Reduce cost
- Create capacity

“If you want to increase innovation you have to lower the cost of failure.”
-Terry Hanold, Vice President, Amazon Web Services.
Why Cloud First?

How Will We Move to the Cloud?

Infrastructure as a Service (IaaS):
- Pay only for what we use and only when we use it.
- Eliminate dependence on data centers/closets.
- Add capacity when we need it and remove it when we don’t.

Software as a Service (SaaS) & Platform as a Service (PaaS):
- Leverage commodity solutions whenever possible.
- Spend our time on differentiating activities.

On-Premises:
We will only rely upon or build on-premises solutions when close proximity to campus is required.

Agility
Cost
Resiliency
Scalability
Sustainability
Why Cloud First?
Cloud Happenin’s

- UAR
- Board Metrics site
- AWS Educate
- CCP

- Student Portal
- Common Help Desk
- Acquia
- Student Email

- Amazon Web Services
- Box
What’s Next?

PeopleSoft
Email
DNS
Compliant Box
Desktop Backup
Call Center?

Azure
Google Compute & App Engine
Voice
AWS Educate
[Your Application Here]

WE NEED YOU!
UAccess Research Case Study

Mark Fischer
UAccess Research in AWS

- UAccess Research launched in AWS in April of 2015
UAccess Research in AWS

- Application Stack

- Kuali Research App
  - Tomcat
  - Java

- Oracle Database
Amazon Tools

- CloudFormation
- JSON format
Amazon Tools – CloudFormation

- CloudFormation
- Allows you to define every aspect of your application stack
- CloudFormation Templates stored in git repository
When we started, CloudFormation template JSON was all made by hand.

We wrote a custom CloudFormation template generator called Stratiform.
Amazon Tools – CloudFormation

- Now they have this awesome web-based design tool!
Amazon Tools – OpsWorks

- OpsWorks
- “DevOps” buzzword!
- Allows for automated, repeatable deployments
- Configuration as Code
Amazon Tools – OpsWorks

- Amazon Linux Base AMI
- Install Java
- Install Tomcat
- Configure Application
Amazon Tools – OpsWorks

- When changes need to be made to environments, the Chef scripts are updated, and Application Instances are re-deployed

- Routinely Cycle through Instances
Instances

An instance represents a server. It can belong to one or more layers, that define the instance's settings, resources, installed packages, profiles and security groups. When you start the instance, OpsWorks uses the associated layer’s blueprint to create and configure a corresponding EC2 instance. Learn more.

Instances

<table>
<thead>
<tr>
<th>Status</th>
<th>Type</th>
<th>AZ</th>
<th>Public IP</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>online</td>
<td>24/7</td>
<td>us-west-2a</td>
<td></td>
<td>stop ssh</td>
</tr>
<tr>
<td>stopped</td>
<td>24/7</td>
<td>us-west-2b</td>
<td></td>
<td>start delete</td>
</tr>
<tr>
<td>online</td>
<td>24/7</td>
<td>us-west-2b</td>
<td></td>
<td>stop ssh</td>
</tr>
<tr>
<td>stopped</td>
<td>24/7</td>
<td>us-west-2a</td>
<td></td>
<td>start delete</td>
</tr>
</tbody>
</table>
Each time you launch a new instance via OpsWorks your deployment workflow is run

- All your Chef scripts are checked out from version control
  
- Allows for documented, repeatable deployments
Amazon Tools – RDS

- RDS is Amazon’s Relational Database Service
- Running Oracle EE 11i
- Allows for High-Availability
  - Your database is replicated across multiple availability zones in your region
Amazon Tools – RDS

- Fill out a form – have a database!
Amazon Tools – API

- AWS has a very complete and well documented API

- The entire AWS Web Interface is built upon the API – Anything you can do with the Web Interface you can do with the API
The AWS API allows us to automate a wide variety of tasks:

- Copy Data from one DB to another
- Create / Backup / Restore DB snapshots
- Deploy updated code
- Stop / Start Instances on a schedule
  - Production up 24/7
  - Support / Dev / Test environments only up during business hours
Other Tools – Jenkins

- Jenkins is a very popular build automation tool
- Used for application builds
- Used to run custom scripts against the AWS API
Other Tools – Jenkins

- Jenkins is a very popular build automation tool.
- Used for application builds.
- Used to run custom scripts against the AWS API.
Other Tools – Jenkins

- Jenkins allows us to push business tasks up to non-developers

- Business Analysts can now perform day to day tasks that used to require a developer or system administrator
Flexibility Flexibility Flexibility Flexibility

On premises, we didn’t know how much hardware we needed. We ended up over-buying by several magnitudes ‘to be safe’

In AWS we also didn’t know, so we launched with large instances

After two months of monitoring, we were able to reduce the instance size to match observed load
Amazon Tools – The Awesome Bits

- Flexibility Flexibility Flexibility

- When our RDS Database filled up, we were able to expand its storage in a few minutes

- No over-buying of disk, or waiting weeks/months for new disk to be purchased and installed
Amazon Tools – The Not-So-Fun-Parts

- VPC (Virtual Private Cloud) setup can be tricky

- Keeping track of Subnets, Security Groups, Route Tables and Access Control Lists is finicky work

- Once the settings are figured out, codify them in the CloudFormation template so you don’t have to figure it out again
Amazon Tools – The Not-So-Fun-Parts

- Identity Access Management Roles are powerful but exacting
  - Thinking about access policy ahead of time pays off
 UITC AWS Next Steps

- Docker!
- Pre-Build application containers instead of using Chef to build out each instance