software development tools 2014

Mark McCahill - mccahill@duke.edu
Liz Wendland - liz@duke.edu
how do you roll?

languages
libraries
software ecosystems
data access
naming
scaling up and migrating your applications
tools from OIT

self-service sandboxes + infrastructure for coursework

research projects

Innovation CoLab (http://colab.duke.edu)
need a server?

VM-Manage: http://vm-manage.oit.duke.edu

- semester-long reservations
- linux VMs with public IP addresses
- pre-built app stacks
- root access
- freedom + responsibility + exit strategy
VM-Manage servers’ naming convention:

`colab-sbx-XYZ.oit.duke.edu`

Vanity DNS CName registrations available here:

[https://vm-manage.oit.duke.edu/cnames](https://vm-manage.oit.duke.edu/cnames)
Bitnami is your friend

PROTIP: don’t start from scratch

a great source of pre-built app and development stacks: http://bitnami.com/
versioning + backups

git @ Duke: http://gitorious.oit.duke.edu

PROTIP: check your code into git then clone onto a VM-Manage sandbox server

git cheat sheet: http://gitolite.com/gcs/index.html
Redmine bug tracker

http://redmine.oit.duke.edu/

track bugs, tasks, feature requests

generate timelines, task lists

Redmine server is backed up daily
OIT has an Apple Enterprise iOS developer license.

We can distribute iOS apps for use by Duke faculty, staff, students *without* going through the Apple Store (and Apple approval process).

**what about mobile?**
documentation & data?

- work in-progress documentation for CoLab developers: http://dev.colab.duke.edu
- streamer: https://streamer.oit.duke.edu
Please don’t solicit usernames+passwords from users to log in on their behalf to Duke services.

use Shibboleth for netID authentication.

opt-in access to some Duke data via OAuth.
OAuth

individualized authorization service
the problem

Blanket approvals for data access are hard

Individuals have varying comfort levels with use of their non-public data

My willingness to grant access to non-public data depends on how it will be used
OAuth

Allows individuals to opt-in to permit access to data on a per-app basis

informed consent which can be revoked at will

apps are granted limited access to data for limited periods of time

OAuth used by Google, Facebook, Twitter, ...
example scenarios

• flex spending account use for an app that helps students manage expenses
• course calendar access to forming study groups or K-ville tenting scheduling
• access to directory photo for social or online forum/discussion apps
OAuth architecture

User → App → token broker

App checks authorized app?

token broker sends authorization code

App → API to authorized resource / data

API to data
OAuth architecture

- User
- App
- Token broker
- API

User authentication

- Does you grant permission?
- Authorized app?
- Authorization code
- API access token for this user?
- Access + refresh tokens

API to protected resource/data

Data
OAuth architecture

User authentication

App

Token broker

API to protected resource / data

User netID + scope

Access token OK?

Access token + API call

data

API access token for this user?

Access + refresh tokens

Authorized app?

Authorization code

Does you grant permission?

Yes

access token + API call

data

authorized app?

access token OK?
**OAuth architecture**

- **IDM (self-service portal)**: manages app access, remove permission.
- **User**: shibboleth authentication, revoke access permission.
- **App**: API to protected resource / data, access tokens, access token + API call.
- **Token broker**: authorized app?, access tokens, access token?, token OK?, netID + scope.
- **API to protected resource / data**: data

Flow:
- User authenticates through shibboleth.
- IDM handles app access permissions.
- User requests access tokens from the token broker.
- Access tokens are used for API calls to protected resources.
- Permission checks determine if access is granted or revoked.
Source code

OAuth-enabled client (node.js)
https://gitorious.oit.duke.edu/oauth-node-examples

IODocs API explorer (node.js)
https://gitorious.oit.duke.edu/iodocs-duke/iodocs-duke

OAuth-enabled LDAP proxy (ruby on rails)
https://gitorious.oit.duke.edu/oauth_ldapproxy/public_sample
coming attractions...
Docker.io

Containerized Linux environments

very lightweight virtualization

10-20x more efficient than traditional VMs

OS and libraries packaged with app

super portable

use a build script to create container
Docker Containers vs. VMs

- **Docker Containers**
  - Containers share network stack & operating system services

- **Virtual Machines**
  - VMs do not share operating system
Docker build script

# mccahill/r-studio
#
# VERSION               0.1
!
FROM   ubuntu:12.04
MAINTAINER Mark McCahill "mark.mccahill@duke.edu"
!
RUN apt-get update && \
    apt-get install

#Utilities
RUN DEBIAN_FRONTEND=noninteractive apt-get install -y vim less net-tools inetutils-ping curl git telnet nmap socat python-software-properties

# need wget and the curl dev libraries to install and run R-Studio and associated packages
RUN DEBIAN_FRONTEND=noninteractive apt-get install -y wget sudo libcurl4-openssl-dev

# install TeXLive 2014 using the installer found here: https://github.com/scottkosty/install-tl-ubuntu
# we do this instead of using the stock Ubuntu version because Ubuntu is way out of date and does not
# include tlmgr (TeXLive's package manager)
#RUN wget https://github.com/scottkosty/install-tl-ubuntu/raw/master/install-tl-ubuntu && chmod +x ./install-tl-ubuntu
#RUN ./install-tl-ubuntu
#RUN rm -rf install-tl*
#RUN echo 'export PATH=/opt/texbin:$PATH' >> /etc/environment
#RUN echo 'export PATH=/opt/texbin:$PATH' >> ~/.profile
#RUN echo 'export PATH=/opt/texbin:$PATH' >> ~guest/.profile
...build script...

```bash
# we need TeX for the rmarkdown package in RStudio - this backport seems to work
RUN apt-add-repository ppa:texlive-backports/ppa
RUN apt-get update
RUN DEBIAN_FRONTEND=noninteractive apt-get install -y texlive texlive-base
RUN DEBIAN_FRONTEND=noninteractive apt-get install -y texlive-latex-extra texlive-pstricks

# get R from the CRAN archive at http://cran.cnr.Berkeley.edu
RUN DEBIAN_FRONTEND=noninteractive apt-key adv --keyserver keyserver.ubuntu.com --recv-keys E084DAB9
RUN echo "deb http://cran.cnr.Berkeley.edu/bin/linux/ubuntu precise/" >> /etc/apt/sources.list
RUN apt-get update
RUN DEBIAN_FRONTEND=noninteractive apt-get install -y r-base r-base-dev

# R-Studio
RUN DEBIAN_FRONTEND=noninteractive apt-get install -y gdebi-core
RUN DEBIAN_FRONTEND=noninteractive apt-get install -y libapparmor1
RUN DEBIAN_FRONTEND=noninteractive wget http://download2.rstudio.org/rstudio-server-0.98.1028-amd64.deb
RUN DEBIAN_FRONTEND=noninteractive gdebi -n rstudio-server-0.98.1028-amd64.deb
RUN rm rstudio-server-0.98.1028-amd64.deb

# update the R packages we will need for knitr
RUN DEBIAN_FRONTEND=noninteractive wget http://cran.r-project.org/src/contrib/knitr_1.6.tar.gz
RUN DEBIAN_FRONTEND=noninteractive wget http://cran.r-project.org/src/contrib/yaml_2.1.13.tar.gz
RUN DEBIAN_FRONTEND=noninteractive wget http://cran.r-project.org/src/contrib/htmltools_0.2.6.tar.gz
```
...build script

#Supervisord
RUN DEBIAN_FRONTEND=noninteractive apt-get install -y supervisor && \\
    mkdir -p /var/log/supervisor
CMD ["/usr/bin/supervisord", "]-n"]

#SSHD
RUN DEBIAN_FRONTEND=noninteractive apt-get install -y openssh-server && \\
    mkdir /var/run/sshd && \\
    echo 'root:CHANGETHISpw' |chpasswd

#Config files
RUN cd /r-studio && \\
    cp supervisord-RStudio.conf /etc/supervisor/conf.d/supervisord-RStudio.conf
RUN rm /r-studio/*

# add a non-root user so we can log into R studio as that user
RUN (adduser --disabled-password --gecos "" guest && echo "guest:CHANGETHISpw"|chpasswd)

# set the locale so RStudio doesn't complain about UTF-8
RUN locale-gen en_US en_US.UTF-8
RUN DEBIAN_FRONTEND=noninteractive dpkg-reconfigure locales

EXPOSE 8787

CMD ["/usr/bin/supervisord"]
R-Studio in Docker

intro statistics course wants RStudio

300+ students

monolithic architecture = FAIL

individual VMs = too resource intensive
Architecture

- **User**
  - Shibboleth Authentication
  - VM-manage
    - Map user to their assigned Docker container
    - Redirect them to the container host/port

- **RStudio-Host-1**
  - Port 49100 + homedir100
  - Port 49101 + homedir101
  - Port 49102 + homedir102
  - ...etc...

- **RStudio-Host-2**
  - Port 49200 + homedir200
  - Port 49201 + homedir201
  - Port 49202 + homedir202
  - ...etc...

- **RStudio-Host-3**
  - Port 49300 + homedir300
  - Port 49301 + homedir301
  - Port 49302 + homedir302
  - ...etc...

- **Backup Mule**
  - Rsync homedirs

- **TSM Backups**

---

Within the diagram:
- The user flows through the Shibboleth authentication box.
- The user then connects to RStudio-Hosts through their assigned port and homedir.
- Rsync is used to copy the homedirs to the backup mule.
- TSM is used for backups.
Architecture: phase II

- **user**
  - shibboleth authentication
  - VM-manage
    - map user to their assigned Docker container
    - redirect them to the container host/port

- RStudio-Host-1
  - port 49300 + homedir300
  - port 49301 + homedir301
  - port 49302 + homedir302
  - ...etc...

- RStudio-Host-2
  - port 49200 + homedir200
  - port 49201 + homedir201
  - port 49202 + homedir202
  - ...etc...

- RStudio-Host-3
  - port 49100 + homedir100
  - port 49101 + homedir101
  - port 49102 + homedir102
  - ...etc...

- Google-Host-1
  - port 49300 + homedir300
  - port 49301 + homedir301
  - port 49302 + homedir302
  - ...etc...

- Google-Host-2
  - port 49200 + homedir200
  - port 49201 + homedir201
  - port 49202 + homedir202
  - ...etc...

- Google-Host-3
  - port 49100 + homedir100
  - port 49101 + homedir101
  - port 49102 + homedir102
  - ...etc...

- CoreOS

- Ubuntu

- backup mule

- TSM backups

- rsync homedirs
Architecture: phase III

- User
  - Shibboleth authentication
  - VM-manage
    - Map user to their assigned Docker container
    - Redirect them to the container host/port

- RStudio-Host-1
  - Port 49300 + homedir300
  - Port 49301 + homedir301
  - Port 49302 + homedir302
  - ...etc...

- RStudio-Host-2
  - Port 49300 + homedir300
  - Port 49301 + homedir301
  - Port 49302 + homedir302
  - ...etc...

- RStudio-Host-3
  - Port 49300 + homedir300
  - Port 49301 + homedir301
  - Port 49302 + homedir302
  - ...etc...

- Google-Host-1
  - Port 49300 + homedir300
  - Port 49301 + homedir301
  - Port 49302 + homedir302
  - ...etc...

- Google-Host-2
  - Port 49300 + homedir300
  - Port 49301 + homedir301
  - Port 49302 + homedir302
  - ...etc...

- Google-Host-3
  - Port 49300 + homedir300
  - Port 49301 + homedir301
  - Port 49302 + homedir302
  - ...etc...

- CoreOS + fleet + etcd

- Rsync homedirs

- Backup mule

- TSM backups
case studies

Switchboard SDN config app
RStudio in Docker
Box provisioning app
You now have access to the same tools and infrastructure the pros use — don’t re-invent the wheel, and have fun!
questions?