Engineering Robust Server Software

Introduction

Duke
UNIVERSITY

Welcome To ERSS!

- Welcome to Engineering Robust Server Software (ERSS)
  - An almost brand new class [second time offered]
    - Feedback/suggestions welcomed
- Introductions:
  - I'm Drew Hilton—-call me Drew
    - Many of you know me from 551 (but not all)
  - Introduce yourselves to everyone
- TAs: Yuxiang, Shangxin, Amber

Assumptions Going Into This Class

- I assume you want to be a software development professional
- I assume you are taking 650 (or have equivalent preparation)
  - You are competent C programmer (Mastery of 551 material)
  - You know basic systems concepts: caching, instructions, etc… (550)
  - If not in 650, you know or are learning:
    - Programming with pthreads
    - Networking
- I assume you are eager to learn this material, and write a bunch of code
- I assume you can consult documentation, try things out, etc.

What is this class about?

- Engineering Robust Server Software
  - **Software**: This class is all about software
    - Hardware may come up in regards to how it affects sw performance
  - **Engineering**: Designing and building systems
    - This is an engineering class, so expect to build a lot of software
    - Focus on useful things in real world
  - **Robust**: Stands up in the face of adversity
    - Badly formed user inputs, many requests at once, evil users…
  - **Server**: handles requests from clients
    - Different constraints from most programs you have written

Server Software

- Servers come in a wide range of "flavors"
- We are going to consider two major ones
  - UNIX daemons: sshd, httpd, …
    - C/C++, systems programming…
  - Web-sites: writing the server side logic for a website
    - Django, databases
- Three major themes
  - Security
  - Resilience
  - Scalability

Duke UNIVERSITY

Five Major Parts To Semester

- [1] Intro (now—~2/13)
  - Requirements/constraints/differences from other software
  - Protocols
  - Unix Daemons
  - DJango/website/AJAX basics
  - Guest lecture (Drew Stinnett [OIT]): Broad Systems Picture
  - Containers
  - Guest lecture (Melissa [IBM]): Continuous Integration/Deployment

Five Major Parts To Semester

- [2] Resilience (~2/13,2/7-3/1)
  - Error handling, exception models/safety
  - High-availability/disaster recovery (Tyler)

Five Major Parts To Semester

- [3] Security (~2/15—3/8)
  - Cryptography basics
  - Common attacks/vulnerability types
    - (e.g., SQL injection, privilege escalation, …)
  - Famous vulnerabilities: Heartbleed, Dirty COW, Apple goto
  - Defense in Depth

Interlude

- Spring break (No class 3/13 or 3/15)
  - I will be in China, limited email
- Then midterm exam Tuesday 3/20

Five Major Parts To Semester

- [4] Performance/Scalability (~3/22—4/5)
  - Non-blocking IO
  - C++ atomics, memory model
  - Serialization bottlenecks
    - Locking granularity
    - "hidden" locks
  - Load balancing
  - Load testing
  - IO Scalability (Tyler)

Five Major Parts To Semester

- [5] "Topics" Guest Lectures (~4/5–4/19)
  - Jim Posen: Coinbase
  - Ravi Soundararajan: VMWare
  - Vlad Petric: Hedge Fund
  - Ken Edwards: Lyft
  - Salman Azhar: previously @ various startups

What Will You Do?

- 4 Homeworks:
  - Pair programming (different partner each homework)
  - Thinking about and write down "dangers"
    - Revisit as semester progresses
- Simple Website (Django)
- Caching Http Proxy (Unix Daemon in C)
- Exploit programs (Attack programs I give you)
- Exchange Matching (Pair "Buy" with "Sell" orders)

"Danger" Log

- Critical programming skill: "spidey sense"
  - As you write, internal mental warning of danger
    - "What if the user …"
    - "What if we run out of memory…"
    - "What if this fails…"
    - "What if…"
- As you code, think of these, write them down
  - Submit a text file with your thoughts
  - Particular focus on class themes (security, resilience, scalability)

"Danger" Log 2.0

- As you learn new things, revisit old assignments
  - Look at code:
    - What should you have worried about?
  - Look at danger logs:
    - What could you have done about these dangers?
- Update log with new thoughts ~weekly.

Pair Programming

- Highly recommended development model: **pair programming**
  - Not just "doing assignment with a partner"
- Partners work on code at same time
  - One is "driver"
  - The other "navigator"
  - Switch roles frequently/as needed
- Driver: writes code
- Navigator: watches
  - Looks for errors, danger, thinks about bigger picture..

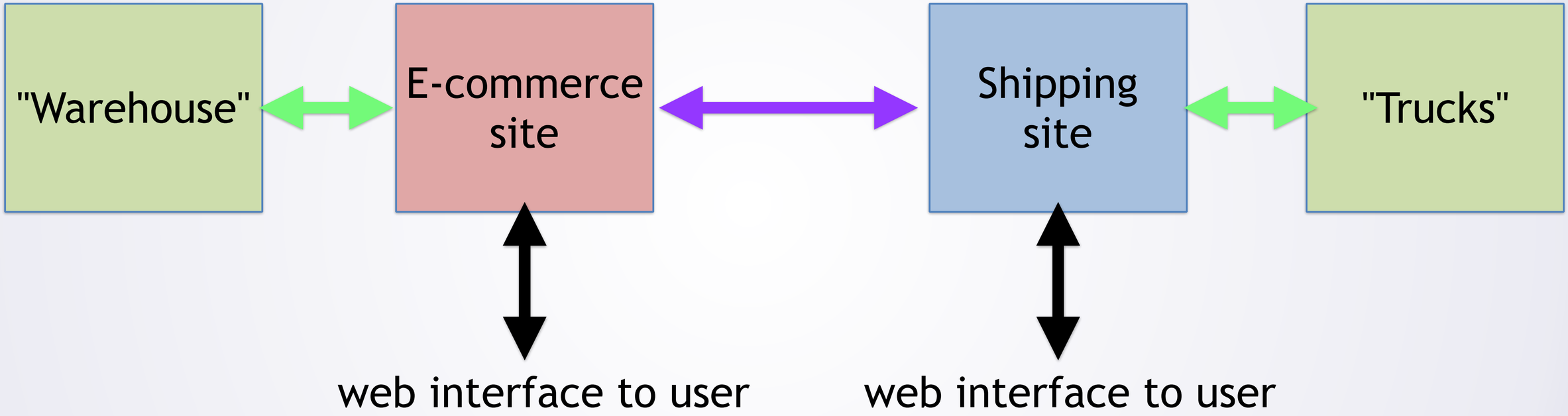
Pair Programming

- Useful tool: screen (or tmux)
  - Multiplex terminal session
  - Can have two terminals connected to one logical terminal
    - Both of you can look at, edit code from your own laptops
  - Facilitates switching driver/navigator
- Either in same room, or on voice chat of some sort
  - Typing too slow

What Will You Do? (cont'd)

- 1 Midterm (Tuesday 3/20)
- 1 Final (Registrar exam schedule)
- 1 Project (Due Fri: 4/27)
  - Do in pairs (may select partner from prior hwk)
  - Half class: e-commerce site ("Amazon")
  - Half class: shipping site ("UPS")
  - Systems have to interact

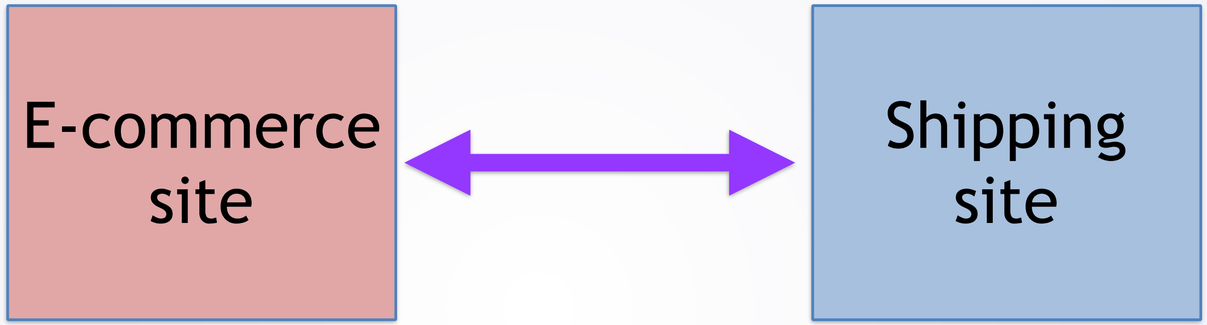
Project: High-level View



Project: High-level View

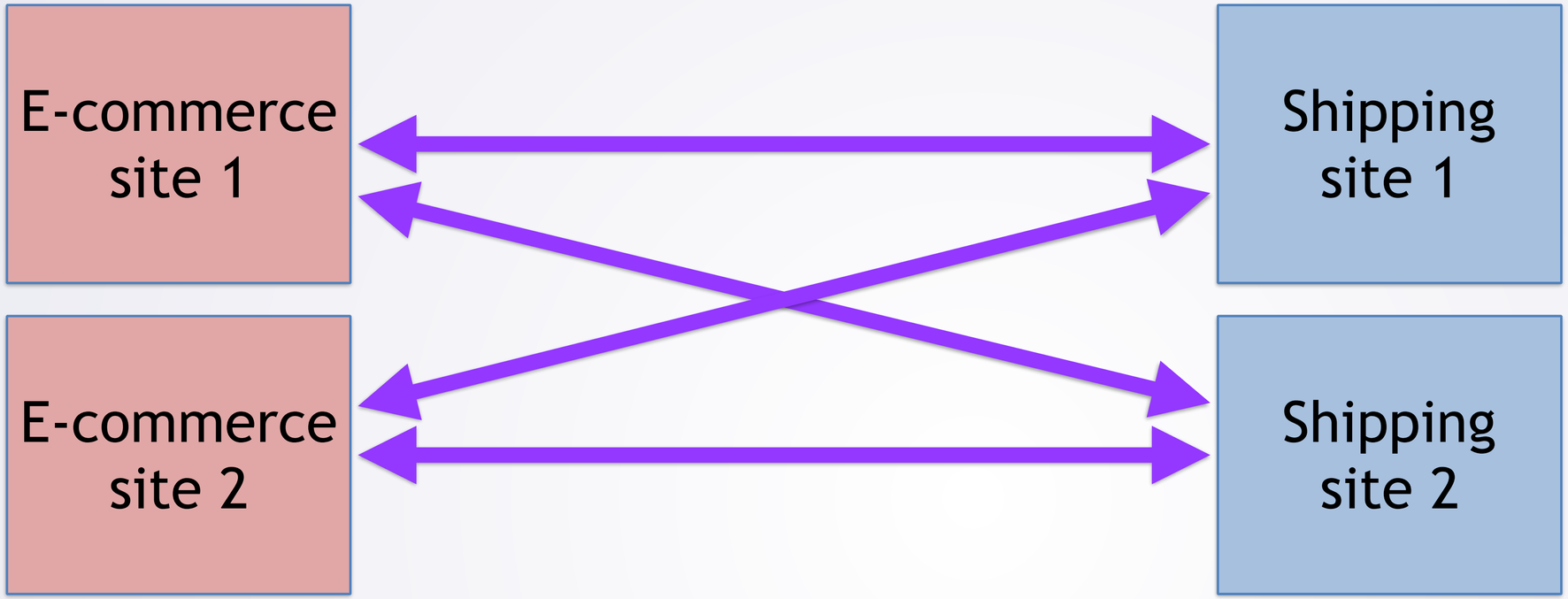
- I will define these protocols/implement these parts…
  - I'll give you a protocol spec
  - …but you should be resilient to **anything**
    - After all, that is a goal of this class

Project: High-level View



- You will do either the red (e-commerce) or the blue (shipping)
  - Protocol between them? Defined by your **interoperability** group

Project: High-level View

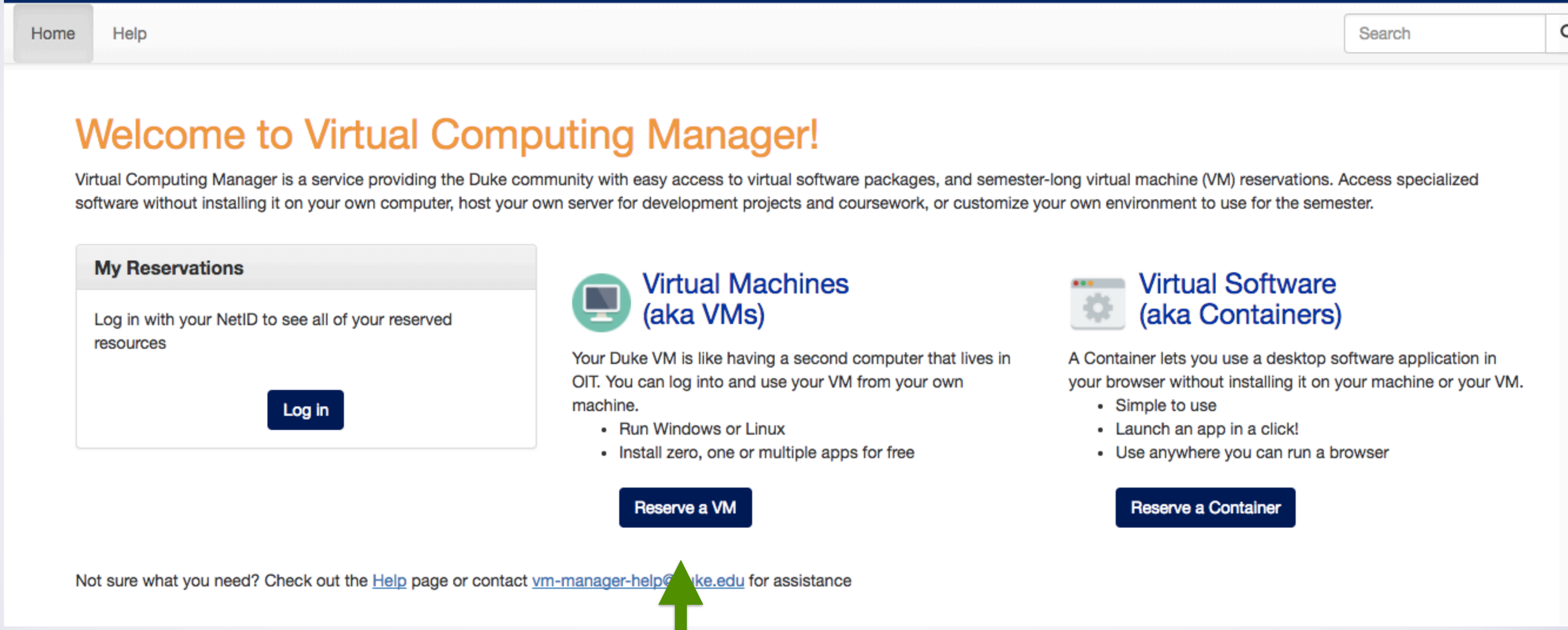


- 4 groups (8 people) = 1 interoperability group
  - Both e-commerce sites must work with both/either shipping site.
  - 8 of you define protocol

Where will you do it?

- You will each have your own server
  - You get root on it, you administer it
- OIT VMs
  - Go to https://vcm.duke.edu/
  - Login with netid
  - Special setup with https certs pre-installed
- Deployment to Azure
  - Details coming later

Choose Reserve a VM in Lower Middle

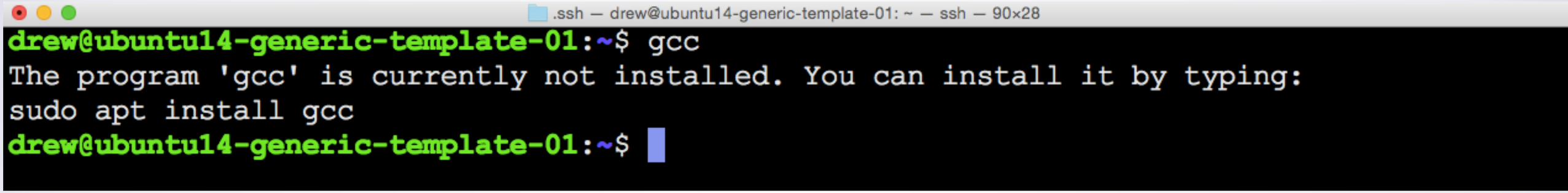


- OIT needs a few more days to setup: so can't do right now

Next Steps

- Login to your server
  - Username vcm
  - Password (provided on confirmation screen)
- Setup a user account w/ sudo
  - sudo adduser *name*
  - sudo adduser *name* sudo
- Now you can ssh in as *name*
- Recommended: setup ssh key pair

Install Software!



- Your server: fresh image, not much software installed
  - sudo apt get install *package*

Packages you probably want to install

- For C development: **gcc g++ make valgrind**
- For editing: **emacs screen**
  - Recommended .screenrc: escape ^oo
- For source control: **git**
- Database: **postgresql-9.5**
- For Django: **python python3-pip**
  - Then do: **sudo pip3 install django psycopg2**
  - Then **django-admin --version** should give 1.10.4
- Libraries: **libssl-dev libxerces-c-dev libpqxx-dev**
- Documentation: **manpages-posix-dev**

Recommended Server Setup [Optional]

- Set up your "dot files"
  - ~/.emacs : emacs configuration
  - ~/.profile : commands read on login

```
export EDITOR='emacs -nw'
export VISUAL='emacs -nw'
```

- Setup ssh key pair(s)
  - Login without password: private key authenticates
- Pick somewhere to backup your work
  - Keep a git remote on **another** computer

Grading

- Grade Breakdown:
  - Homeworks: 25%
  - Project: 25%
  - Midterm: 20%
  - Final: 30%
- Letter grade:

A- [90,93)	A [93,97)	A+ [97,∞)
B- [80,83)	B [83,87)	B+ [87,90)
C- [70,73)	C [73,77)	C+ [77,80)
	F (-∞,70)	

RFCs

- Many standards are in the form of RFCs
- You SHOULD spend some time reading RFCs this semester
  - …and may effectively write one during your project
- Start with this one (describes MUST/MAY/SHOULD etc in RFCs)
  - https://tools.ietf.org/html/rfc2119

Next Time..

- Wrap up for this time:
  - Questions?
  - Find partners for homework 1
- Next time:
  - Start talking about server software