ResBaz
RESEARCH BAZAAR

ResGuides
Training for digital research

Table of Contents

Readme

This a the material for a "Train the Trainers" event.

The trainers are preparing to deliver the NeCTAR cloud lessons.

Thus they must be familiar with the basics of the NeCTAR cloud. These are the topics they should be comfortable with:

- Security groups
- Key pairs
- Launching an instance
- Ephemeral drives
- Snapshots
- ssh
- The Linux command line
- The object store

Attendees prerequisites for this course

1. [ ] A laptop
2. [ ] Internet access
3. [ ] A GitHub account
4. [ ] Git locally installed
5. [ ] An active account on the NeCTAR dashboard

Git

If you check this repository out be aware that it uses Git submodules to manage the reveal.js dependency. To also clone reveal.js, you will have to either:

```
# fetch it all in one hit
git clone --recursive https://github.com/MartinPaulo/ResOsTrainTheTrainer.git
```

Or:

```
# take it step by step
git clone https://github.com/MartinPaulo/ResOsTrainTheTrainer.git
git submodule init
git submodule update
```

To regenerate the slides

The SlideExtractor.jar in the root directory will re-create the slides if needed.

To run it ensure that the java version installed is java 8:

```
java -version
```

should return something along the lines of `java version "1.8.0_65"`.

If it doesn't then install java 8 from here:
http://www.oracle.com/technetwork/java/javase/overview/java8-2100321.html

Then in a command prompt in the root directory simply issue:

```
java -jar SlideExtractor.jar
```

You should see something like the following fly by:

```
Working on: ./Booklet/Booklet.md
Writing to: ./Presentation/Booklet.html
Writing to: ./Presentation/index.html
```

Instructor's notes

Check that the etherpad exists. If not, recreate it using the downloaded version
Remember to always walk through your slides when you regenerate them!

Preparation for delivering the actual course.

To give the lessons, you need to

1. Clone the repository
2. Update the repository submodules
3. **Optional:** Checkout the tag you want to deliver
4. **Optional:** Update the introductory slide with your name
5. **Optional:** Regenerate the slides
6. Open the lesson slides in your browser!

So, you, the presenter, will need to:

1. [ ] Have an AAF account
2. [ ] Have checked the training material out of GitHub
3. [ ] Have a terminal of some kind that you can ssh into instances with

We suggest getting attendees to log onto the NeCTAR cloud, and to install any needed software before the day.

So before the day, send out an email message to attendees advising them of the prerequisites.

Find out about the venue. Does it have enough:

1. [ ] Seats
2. [ ] Tables
3. [ ] Power points

Does the venue have:

1. [ ] Network connectivity (wifi)
2. [ ] Internet access
3. [ ] A big display you can use
4. [ ] A whiteboard (or paper) that you can draw on
5. [ ] Pens (and cleaner for the whiteboard)

What kind of connectors do you need to bring along for your devices?

Do you have:

1. [ ] Network connectivity (eg: are you banking on having EduRoam work for you, and have you tested that it is available and does indeed work for you...)
2. [ ] Enough sticky notes, in two different colours, to hand out to attendees at the start and after each break
3. [ ] A set of answer cards, enough for each learner, lettered 'A', 'B', 'C', 'D' and 'E'

respectively.
4. [ ] A set of three envelopes, at least one of which fits inside the other two, to describe ssh and man in the middle attacks.
5. [ ] Any other props you might like to use?
6. [ ] A set of spare pens for people to write on the sticky notes with (It would appear in this digital age a lot of people travel penless)?

The image to be used:

1. [ ] Is the image named `res_os_drupal7` available and public on the NeCTAR cloud?
2. [ ] If so, check that it works as expected. Make a snapshot so that you have a fall back image.
3. [ ] If it's not there, or doesn't work as expected, make a new image to use.

Have you checked the slides are all as you expect?

On the day:

1. [ ] Each learner will need a laptop with wifi access
2. [ ] Do you need to deal with Chromebooks? Crosh Or Chrome Shell might be needed
3. [ ] Each learner on the course must have an AAF logon.
4. [ ] Each learner must have an allocation on the Research Cloud that they can use.

On the day for those that have expired trial projects you can:

1. [ ] get them to pair up with others
2. [ ] have a special tenancy for the lesson, and then add them to it on the fly. This may not a great solution as people in the tenancy will step on each others toes. But is has worked for us.
3. [ ] have someone on hand to extend their trial tenancies on the spot?

If you are going to stream people, devise and send out a pre-assessment questionnaire. We'd love a copy!

Afterwards...

- Collate the feedback. Is any of it actionable?
- Remove users from a shared tenancy (if you had to add any)

Some links

- The software carpentry store: http://www.cafepress.com/swcarpentry
- Log shell output to Etherpad: https://github.com/c-martinez/shellther
- http://sciencecases.lib.buffalo.edu/cs/pdfs/Clicker%20Cases-XXXVI-2.pdf

To Resolve

1. [ ] As in Software Carpentry, should trainers who submit a GitHub correction get a certificate?
2. [ ] What about a code of conduct: http://software-carpentry.org/conduct/
3. [ ] Add a backing etherpad?
4. [ ] Send out reminder email about the post day survey...

Introduction

-- *Slide* --

Program

- Introduction
- Software Carpentry: an overview
- The format of the course material
- An Overview of the course material
- Git - and our lessons
- Practical
- Feedback
- Preparing to present
- How learning works
- How to contribute changes back to the course

-- *Slide End* --

-- *Slide* --

Your guides today:

- Martin Paulo (martin.paulo@unimelb.edu.au)
- Jared Winton (jwinton@unimelb.edu.au)

-- *Slide End* --

-- *Slide* --

Etherpad for today:

- https://etherpad.wikimedia.org/p/ResOsTtT

-- *Slide End* --

-- *Slide* --

In the beginning was ResBaz

ResBaz (the Research Bazaar) is a campaign, a community and conference.

-- *Slide End* --

- A campaign: to provide free peer to peer training on digital skills to researchers.
- A community: to provide support to researchers through collaboration.
- A conference: an annual 3 day multi city event to kick start the campaign and the community every year.

-- *Slide* --

We noticed

That an increasing number of researchers were turning to the cloud

-- *Slide End* --

-- *Slide* --

We realized

There was a training gap

-- *Slide End* --

-- *Slide* --

Hence

We developed a course

-- *Slide End* --

We delivered an introduction to the NeCTAR cloud at ResBaz 2015 in Melbourne.

As ResBaz makes heavy use of Software Carpentry, we decided to emulate its style and practices.

What is ResOS?

We also came up with a conceit. A handle we call:

ResOS == Researcher Operating System

A handle we have used to abstract away the mechanics of the cloud from researchers.

-- *Slide* --

ResOS: what is it?

1. Researcher Open Source
2. Researcher Onion Storage
3. Researcher Open Storage
4. Researcher Operating System
5. A conceit

-- *Slide End* --

Answer: D & E I have such mixed feelings about this term. But it's one that some people have taken to heart. So I'm putting it here just in case you talk to someone who uses it. Now you don't have to interrupt them to find out what on earth they are talking about!

Subsequent to that initial ResBaz event we were commissioned by NeCTAR to expand and enhance that material into a full day hands on course.

So some of our foundations precede the online NeCTAR training material created by Intersect. We had to aligned the two by changing our material. Which we were only able to do after Intersect published their training.

We've piloted all the material by giving lessons to small numbers of researchers, and solicited their feedback.

We are now using it in training at the University of Melbourne. And we will be changing it and incorporating what we learn from delivering it as we go forward.

Software Carpentry

-- *Slide* --

Software Carpentry: an overview

NB ResOS training is *not* a Software Carpentry course.

-- *Slide End* --

History

Given that we based our training on we learnt from Software Carpentry, you should know a bit about Software Carpentry, and the ideas behind it.

Software carpentry grew from the realization that researchers need to install and maintain software: not only that - they need to occasionally write, debug and test it as well. Yet most researchers are not taught to do this!

A course to tackle this problem was developed and run by Greg Wilson and Brent Gorda in 1998.

From the delivery of this course they learnt that:

-- *Slide* --

Lesson 1

- Week long courses frazzle the attendees brains
- Textbook SLDC[1] is not useful to researchers

1: Software Development Life Cycle

-- *Slide End* --

Discussion Try to have a discussion about the SLDC, and why this might be.

Subsequent to this the course materials were put on line. Other iterations followed.

During this period it was realized that:

-- *Slide* --

Lesson 2

- Most computer scientists don't have an interest in training researchers
- The target market is new postgraduates

-- *Slide End* --

MOOC style courses were also attempted. From this came the realization:

-- *Slide* --

Lesson 3: The MOOC[1]

- Video's are a huge amount of effort
- Most people don't complete the courses

1: Massive Open Online Courses

-- *Slide End* --

-- *Slide* --

Who's done a MOOC?

- G = Yes!
- R = No: why would I?

-- *Slide End* --

-- *Slide* --

Who's finished all the MOOC's they have started?

- G = Yes!
- R = Life. Just keeps on interfering :(

-- *Slide End* --

After years of effort, review, and development, Software Carpentry now has the following format:

-- *Slide* --

Format

- A two day workshop
- Covering a small set of tools
- That introduce high level concepts
- With the goal of teaching computational competence

-- *Slide End* --

The following guidelines are in place

-- *Slide* --

Guidelines

- No more than 40 people in a workshop
- If possible, parallel sessions, with attendees streamed by ability

If parallel sessions are run, then:

- A pre-assessment questionnaire is used to create the groups
- People are not swapped from group to group

-- *Slide End* --

The workshops aim to have:

-- *Slide* --

Staffing

- An instructor up front
- At least 1 roaming helper per 8 attendees

-- *Slide End* --

Group sign up leads to better workshops and better attendance by under represented groups

-- *Slide* --

Feedback loops

- Sticky notes

- Minute cards
- Summary feedback
- Post workshop assessments

-- *Slide End* --

Extensive use is made of feedback loops. So:

Sticky notes are handed out to learners:

- two different colours (typically, Red and Green)
- used for true/false answers
- to show pain if not not coping
- and to signal that activities are completed

Minute cards:

Before each break the learners are asked to write one thing:

- that they've learnt on their green sticky note
- that they found confusing on their red sticky note

The instructor then collates these and incorporates the feedback into the next iteration.

Summary feedback at the end of the day:

Every attendee is asked to stand and to give one good point about the day, and one negative point, without repeating what has already been said. The instructor makes notes and incorporates the feedback into the next iteration.

Post workshop assessments:

After the workshop an assessment is sent to the attendees. These don't have high return rates.

Instructors are also requested to provide feedback in debriefing sessions.

-- *Slide* --

We really like

- Multiple choice questions

-- *Slide End* --

We find that not only do they help us with seeing if the students understand the material, they also act as punctuation marks in the lessons that engage the students.

We have found it really hard to design good questions, though...

-- *Slide* --

Delivery

- Live coding
- Two devices for the instructor
- An Etherpad session
- Learners work on their own machines
- Pair programming (so dinner table style rooms over lecture theaters)

-- *Slide End* --

We find it really good if the instructor leaves the podium and wanders around during the exercises. This stops the students from focusing on the front of the class. There seems to be some kind of weird leadership thing in play?

-- *Slide* --

SWC: biggest challenges

- Are they truly helping researchers?
- Diversity of learners - 20% bored, 20% lost :(
- People don't contribute back...

-- *Slide End* --

So

Software carpentry has been under continuous iterative development for many years now. Along the way the material has been reworked to take into account the latest in educational research and feedback from the actual delivery.

BTW, Software Carpentry material is licenced under a creative commons licence. Anyone is free to use it.

NB But you are not free to use their name and logo! There are strict conditions applied to this. They want to maintain their good name and reputation!

We feel that if you haven't done it yet, that you should sign up for their trainers course!

-- *Slide* --

Subscribe!

If you are going to be doing a lot of SW Carpentry style training consider subscribing...

- http://software-carpentry.org/join/

-- *Slide End* --

I'm going to intersperse today with some fun things we've learnt from Software Carpentry. So:

-- *Slide* --

True or False: People have different learning styles

- G = True.
- R = False.

-- *Slide End* --

A False - it's true that we have preferred learning styles. But our preferred learning style is probably not be our best learning style...

Further, learning styles are typically given a granular nature and labelled ("Visual!", "Auditory"), but us humans actually form a continuum. Preparing lessons for a label short changes the continuum.

The format of the course material

We have elected to write our course material in Markdown.

-- *Slide* --

Do you know what markdown is?

- G = Yes.
- R = No.

-- *Slide End* --

NB If there are any "No's" you have to explain markdown...

The following may help:

HTML is simply text with tags in it.

Demo:

```
<h1>A Heading</h1>
<p>Paragraph text
```

Your browser parses the text and then renders it appropriately. But all those tags are a pain to type. Hence markdown: the lazy way of generating HTML. You write:

```
#A heading

Paragraph text
```

And feed it into an engine, and out comes beautifully formed HTML.

-- *Slide* --

Open the following sites in tabs:

- http://tinyurl.com/ResOsLd
- http://tinyurl.com/ResOsMd

- G = I'm ready to proceed.

- R = What: there is no such site!

-- *Slide End* --

Pair up!

And see if you can produce the following well turned out html in the live demo site...

-- *Slide* --

This is a heading

Followed by a **paragraph**

and a subheading

- with a bullet
- list

```bash
# this is pretty printed bash code
ls -al
```

-- *Slide End* --

Hold up a green card when you are done

And a red card if you want help...

Demonstrate Show that you can type raw html into the markdown by adding

```html
<a href="http://www.nectar.org.au">Nectar</a>
```

Tiny URL

There are URL's in our material. We do put them into the course Etherpads, but find that our learners don't engage with the Etherpads. They kind of recoil in horror.

-- *Slide* --

An Etherpad from one of our courses...

https://public.etherpad-mozilla.org/p/NeCTAR_Cloud_1.0.1

-- *Slide End* --

To work around this, and make it simple for learners to enter a url from the slide we decided to use a URL shortner.

There are a lot of url shortners out there.

We settled on TinyURL.com

For one simple reason: it was the first we found in which we could set the URL to be generated.

I believe, with no research behind me, that well selected human readable strings are easier to work with than randomly generated characters.

-- *Slide* --

Our url format

[TinyURL](The original url being shortened)

-- *Slide End* --

We try to keep to this format for URL's in the material.

-- *Slide* --

Why do we use this format?

[TinyURL](The original url being shortened)

1. Markdown mandates this format for url's
2. Developers like clarity
3. TinyURL might vanish!

-- *Slide End* --

Answer: C Although it's been around for quite a while, we aren't sure that TinyURl will continue to be around. It also helps us when reading the source material to see where the shortened url is actually pointing.

-- *Slide* --

Can you

Find a long link, and shorten it with http://tinyurl.com ?

Once done, enter it into the live demo site

- http://tinyurl.com/ResOsLd

In our preferred format:

- [TinyURL](The original url being shortened)

Then get your neighbour to check it for correctness.

- G = Done...
- R = Please help us!

-- *Slide End* --

And that's about the level of markdown you need to know to work on and understand the training material.

-- *Slide* --

True or False: Physical activity helps learning

- G = True.
- R = False.

-- *Slide End* --

A True - Activity makes for a healthy brain. Especially true for children!

An overview of the course material

-- *Slide* --

The material can be found at:

- https://github.com/resbaz/nectar-cloud-lessons

Can you open this in your browser?

- G = Yes.
- R = No.

-- *Slide End* --

How we developed the course.

- We did a concept map on paper.
- Then we created a template for each lesson plan.
- We used the lesson plan to develop each lesson.
- We gave trial deliveries of each lesson to guinea pigs.
- We modified the lessons to match what we learnt from the trial delivery.

The lesson plans are in the repository, under the "Planning" folder. They give a high level overview of each of the lessons.

-- *Slide* --

Do you think we have kept the lesson plans up to date?

- G = Yes.
- R = No.

-- *Slide End* --

Answer: We have tried. But like so much documentation in the software world, we struggle :(

But in getting to grips with a lesson, reading the lesson plan and ensuring that you understand the concepts the lesson is attempting to convey is key.

-- *Slide* --

Can you find and open the lesson plans?

- G = Yes.
- R = No.

-- *Slide End* --

Follow along in the lesson plan as we discuss the lessons.

Lesson 1

We do a high level drive through of the dashboard: and keep the learners engaged by playing a game of follow my leader: and by asking questions as we go. In the end the instructor launches an instance: but it's not intended for the class to follow suite...

-- *Slide* --

What's an ephemeral disk?

1. A quantum disk
2. A disk that moves from VM to VM
3. A disk that only exists for the life of its VM
4. A made up word set
5. Something victorians put their tea cups on

-- *Slide End* --

Answer: C. A disk that lives only for the lifetime of its associated VM.

-- *Slide* --

What's a NeCTAR image?

1. A picture at an exhibition
2. A representation of NeCTAR
3. The general impression that NeCTAR presents to the world
4. A file that contains the contents of a hard drive
5. A fine piece in the NeCTAR art collection

-- *Slide End* --

Answer: D. A file that contains the contents of a hard drive.

-- *Slide* --

No one really does anything in this lesson!

Is that a problem?

- G = Yes.
- R = No.

-- *Slide End* --

Discuss that we've tried mixing lesson 1 and two up: and that we've had best success by introducing the concepts and then the hands on work.

Lesson 2

We get the students to create their own security group, keypair and then launch their own instance.

We scaffold this by giving them checklists to follow.

-- *Slide* --

Jared's lesson

Don't print out the checklists!

-- *Slide End* --

BTW: this is how we communicate key pairs (give a run through of Pem the greek merchant banker...)

-- *Slide* --

Our keypair analogy

Is it a problem?

- G = Yes.

- R = No.

-- *Slide End* --

If it's a problem, we'd like to have a better one!

Lesson 3

Our most ambitious lesson so far!

We introduce the command line, ssh, sudo, package management and show security groups in action.

A secret...

The command line is in here because of our historic baggage.

Outside of our focus groups, we've never done the command line play and the command line itself.

This is because of where and when these lessons have been given.

I want to move it out into its own lesson.

-- *Slide* --

Discuss!

What to do with the command line?

-- *Slide End* --

-- *Slide* --

Our biggest problem...

The amazing teleporting terminal!

-- *Slide End* --

Hence our increasingly laboured explanations around terminals...

-- *Slide* --

Our ssh analogy

Is it a problem?

- G = Yes.
- R = No.

-- *Slide End* --

Can anyone think of a better one?

-- *Slide* --

Our eve in the middle analogy

Is it a problem?

- G = Yes.
- R = No.

-- *Slide End* --

We are finding it more fun to do a play with one of the helpers playing out the roll of the eavesdropper in the middle: and no envelopes!

Also, we put this in because when we first started delivering this course the NeCTAR scheduler was aggressively reusing IP numbers and users would run into this issue often. Now that Neutron is in the mix this no longer the case. Perhaps we could dress down the warning?

Lesson 4

We get our learners to use scp and CyberDuck to move data to and from their remote machines.

-- *Slide* --

Moving data: our top researcher request!

Hence it's worth spending time on...

-- *Slide End* --

Lesson 5

We launch and run graphical applications via XWindows over ssh.

-- *Slide* --

Graphical applications: our top promoter request!

Of interest...

-- *Slide End* --

-- *Slide* --

X-Windows

Where's the XServer?

1. On the users workstation
2. On the remote VM
3. Does it matter?

-- *Slide End* --

Answer: A, in our case. We don't really dive into this, just because of the amount of confusion it can engender.

Lesson 6

We dive deeper into snapshots and scaling.

-- *Slide* --

Why are NeCTAR snapshots broken?

1. They are just too easy to do: suspicious enough!
2. The magic nostrums aren't up to the task!
3. They are not broken! How dare you!
4. The state of the machine is not captured
5. Cosmic rays can flip their bits.

-- *Slide End* --

Answer: D. Whilst E is a topic for a good debate, the answer is: D. The state of the machine is not captured

This seems to be a little known fact :(

We teach a work around that gives faultless snapshots.

-- *Slide* --

Anyone care to guess our technique?

-- *Slide End* --

Answer: Yes, we shut the VM down. Note: Not terminate!!

-- *Slide* --

Do you know the differences between:

1. A soft reboot of a VM
2. A terminated VM
3. A paused VM
4. A stopped VM
5. A hard reboot of a VM

?

-- *Slide End* --

A paused VM is simply halted in memory, ready to be started again. A stopped VM has its state written to the host machines disk, ready to be resumed from there. A soft reboot tries to do a graceful reboot (you ask the OS to reboot). A hard reboot just reboots the machine! (you press the button)

Lesson 7

We have a very clumsy display that transient storage is not to be relied on.

And also do some vertical scaling and public sharing of snapshots.

-- *Slide* --

Vertical scaling: an often featured support request.

Strangely, people want to keep their IP numbers...

Q: is it worth adding info about this to the course?

Or is it too much information too soon?

-- *Slide End* --

Lesson 8

We dive into the Object Store

It's a very simple lesson.

We introduce containers, objects, and get them upload, download and make public an image.

We also, if time permits, get them to work through connecting CyberDuck to the object store using Intersects training material. Which needs fixing...

-- *Slide* --

Object Store

Is it really this simple?

-- *Slide End* --

-- *Slide* --

True or False: Knowledge is as perishable as a fresh fish!

- G = True.
- R = False.

-- *Slide End* --

A False - Is the knowledge that you have, somehow, in some strange quantum way, flipping from correct to wrong with the passage of time?

Or is it simply that the knowledge available is exploding? Sure, some of the knowledge we may hold dear may turn out to be incorrect (Carbohydrates are good for you?), but the majority still holds true. Pythagoras still Rulez!

Lesson 9

We dive into Security.

We explain the shared security model, and give some basic tips on staying safe in the cloud.

This the messiest and least explored lesson, as you'll be able to tell by the lesson plan and the lesson.

-- *Slide* --

True or False: Discovery trumps explanation

- G = True.
- R = False.

-- *Slide End* --

A True and False - Both sides have a point. If you can learn quantum physics by discovery: well, why are you in this room? Research has shown that pure discovery learning doesn't work: but guided investigative learning can be very effective in certain settings.

Git - and our lessons

As you have seen, the course material is kept under version control.

-- *Slide* --

Do you know what git is?

- G = Yes.
- R = No.

-- *Slide End* --

NB If there are any "No's" you have to explain git...

The following may help you:

- https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control
- https://git-scm.com/book/en/v2/Getting-Started-Git-Basics
- http://tom.preston-werner.com/2009/05/19/the-git-parable.html
- https://matthew-brett.github.io/pydagogue/curious_git.html

-- *Slide* --

Which of these is false:

Git:

1. Is a version control system
2. Is a distributed version control system
3. Is able to run offline
4. Has three states for files: modified, staged, committed
5. Can do your head in...

-- *Slide End* --

Discuss each of these options further before asking for the answer. For Option E you can use this:

"*Git has a very simple and powerful underlying model. Atop this model is piled an immense trashheap of confusing, overlapping, inconsistent commands. If you try to just learn what commands to run in what order, your life will be miserable, because none of the commands make sense. Learning the underlying model has a much better payoff because it is much easier to understand what is really going on underneath than to try to infer it, Sherlock-Holmes style, from the top.*" - Mark Dominus

Answer I believe them all to be true...

Any confusion here should require further discussion...

Not only is the course material kept under version control: that version control is accessible via an online repository, so you can revisit it at any time, use it as a stand alone course if you want to, and even, hopefully, contributes changes and extensions back!

To check the course out onto our local machine we do the following:

```
git clone https://github.com/resbaz/nectar-cloud-lessons.git
```

Question Can you check the lessons out?

Hold up a green card when you are done And a red card if you need help.

The exciting thing about our repository is that we actually include a reference to another repository.

So to get a copy of the child repository we need to tell git to update the local configuration files with information about any referenced repositories.

```
git submodule init
```

Question Can you init your submodules?

Hold up a green card when you are done And a red card if you need help.

Once we've done that we tell git to fetch any referenced repositories.

```
git submodule update
```

Question Can you update your submodules?

Hold up a green card when you are done And a red card if you need help.

What is this referenced repository, I hear you ask?

Well, to quote someone

-- *Slide* --

"I've got a plan so cunning, you could put a tail on it and call it a weasel[1]"

AKA: REVEAL.JS

1: Blackadder

-- *Slide End* --

Reveal.js is a Javascript presentation framework. It's really cool

We have written a tool that that chunders through our lecture notes...

-- *Slide* --

SlideExtractor.jar

`-- *Slide* --`

In the lessons any text between lines that have only the above marker

And finish with only the below marker

`-- *Slide End* --`

Will extracted into a lesson slide.

-- *Slide End* --

Demonstrate on the command line

```
ls  # show the jar and the properties file
java -jar SlideExtractor.jar # need to have Java. Talk about language choices  And
 VM choices!
open ./Presentation/Lesson_I.html  # show in browser
```

The reason for doing this was because it means that our slides and lecture notes all live in the same file. Hence:

- As we are working through them we can see what will be coming up next.
- We don't have to maintain separate presentation material.
- It should help people who use screen readers track what's happening.

Also we/I am toying with the idea of splitting out the questions into a separate file: and of writing an app to allow us to get users to actually select these from within a browser window. So that we can gather metrics about the responses.

-- *Slide* --

Discuss!

Are clickers a good idea?

-- *Slide End* --

The downside of our slide extractor is that we run the risk of sinking into the PowerPoint trap

We've divided the material into one file per lesson.

And there's no need for you to generate the slides: we've checked the generated slides into the repository.

-- *Slide* --

To give the lessons (basic)

1. [ ] Clone the repository
2. [ ] Update the repository submodules
3. [ ] **Optional:** Checkout the tag you want to deliver
4. [ ] **Optional:** Update the introductory slide with your name
5. [ ] **Optional:** Regenerate the slides
6. [ ] Open the lesson slides in your browser!

-- *Slide End* --

BTW, I just print the lesson plans off of GitHub: which does a beautiful job of rendering the markdown.

-- *Slide* --

Can you open, say Lesson_II.html in your browser?

And step through some of the slides?

- G = Yes.

- R = Houston, we've had a problem here!

-- *Slide End* --

We've settled on the following way of managing the material:

There is a version number at the top of the README.md. That version number corresponds to the last tagged copy. If we are going to deliver the course and there are lots of changes since the tagged copy given by the version number, we create a new version and tag it.

This means that you should always be able to go back to a particular version in time.

-- *Slide* --

What's the current version number?

1. 1.0.0
2. 1.0.1
3. 1.1.0
4. 1.1.1
5. 1.2.0

-- *Slide End* --

To fetch all the tags from the remote origin:

```
git pull --tags
```

Then to see a list of the tags

```
git tag
```

-- *Slide* --

Can you list the tags?

- G = Yes.
- R = No.

-- *Slide End* --

GitHub shows you, but if you want see the commits between the last tag and now locally

```
git log --pretty=oneline master...1.0.0
```

You might have to explain the above command on a board...

-- *Slide* --

Are there are any commits since the last tag?

- G = Yes.
- R = No.

-- *Slide End* --

It's good to check to see if there are any changes since the last tag: and if you want to include them in your lessons or not.

If you have a set of notes for a particular version of the course, and that's what you want to deliver you can checkout that particular tag and use it.

How to checkout a specific tag as a local branch:

```
$ git checkout tags/<tag_name> -b <tag_name>
```

-- *Slide* --

Can you checkout version 1.0.0?

- G = Yes.
- R = No.

-- *Slide End* --

-- *Slide* --

True or False: Google makes learning obsolete!

- G = True.
- R = False.

-- *Slide End* --

A False - Would you let a brain surgeon near you who going to Google every step of your operation along the way? I wouldn't...

We need context and a mental structure to make sense of the facts we find on the Internet. Context and structure we get from learning... In fact, given the amount of information available on the Internet we might need more learning to make sense of it all!

Practical

Personas: an introduction

Ok, break up into groups of 3-4 people.

The first exercise that I want you to do as a group is to develop a persona.

"What's a persona?" I think I hear you ask.

A persona is the representation of the goals and behaviour of a group of hypothetical users.

The theory is that by understanding the needs and abilities of that persona, you can better satisfy the needs and abilities of the larger group.

A good persona should represent a skill set and a behaviour pattern - not a job description.

Software Carpentry make use of persona's: they call them "Learner Profiles".

By example, here's a persona that we developed:

-- *Slide* -- **Name:** Associate Professor Glenn Bording

Job title: Senior researcher and lecturer, Radio Astronomy group

Demographics:

- 40 years old
- Programming experience in C and Fortran
- Basic HPC experience
- Prefers to work with physical hardware

-- *Slide End* --

-- *Slide* --

Background:

- While preparing a new course for undergraduates, developed a computing challenge to support research using real data
- Realised the BYOD for most students were not powerful enough

- Labs not suitable for task
- Teaching parallel programming and getting accounts for students would be difficult
- Heard about the research cloud

Goal:

- Now keen to learn about ResOS to see if it solves this problem.

-- *Slide End* --

Once you, as a group, have agreed on your persona, hold up a green sticky note. But you'd better be quick, because I'm giving you only 5 minutes!

For more on personas, there is the wikipedia entry and of course, the book The Inmates are Running the Asylum. Also, the blog posting Perfecting Your Persona's And dont' forget: Software Carpentry's Personas.

Ok: for our purposes today, the persona's that you have developed represent the people that you will be delivering our ResOS training course to.

So as we go through the rest of today I want each group to ask themselves just what it would take to communicate the concepts or activities we are covering to this persona.

If you find you can't do it, then make a note, and we will all discuss it after the next section and see if we can come up with a better approach.

Ask for one or two persona's to be read out

So as you have seen, we have developed a fairly formal course intended to be delivered by a person standing in front of a class.

And we've now done a fairly high level walk through of the lesson plans and their formatting: and of the tool we've developed to support our delivery.

We are now going to split up into groups: each group will choose a lesson to cover. Then the group will work through the lesson plan and associated lesson. We'll take, say, 45 minutes to do this.

Then we will meet up again and try presenting our material for a short period.

As we work through this, I want you all to note any errors or changes that need to be done to the course material.

By raising issues!

-- *Slide* --

Do you know how to raise an issue on GitHub?

- G = Yes.
- R = No.

-- *Slide End* --

If there are any No's: well, they need to be shown!

-- *Slide* --

True or False: Memory records exactly what we experience...

- G = True.
- R = False.

-- *Slide End* --

A False - Sadly it's been proved that not only is our memory of what we experience highly unreliable, it's even possible to plant false remembered experiences through clever questions.

-- *Slide* --

BTW - who has seen this:

-- *Slide End* --

Once you've worked through your chosen lesson and think you are ready to present hold up a Green sticky note!

Remember, the easy thing about software carpentry is that you can present with the lesson right up in front of you. So you don't have to memorize anything.

I've seen people go as far as reading from the lesson.

I try not to: I try to keep the notes to one side, and to glance at them from time to time. I do scribble highlights of important stuff on them so I don't miss it.

The more familiar I am with material the less I need to look at the material.

But I plead guilty to putting extra slides up just to keep me on track...

```
Take break while people go through the lessons
```

Improving our teaching

Before we actually try presenting lets talk about improving our teaching.

We all have unconscious nervous habits. I used to rock backward and forwards.

When that was pointed out to me I stopped doing it. Then I started wringing my hands instead.

That was pointed out to. I stopped doing it. Then I started to rock from side to side.

That was pointed out to me...

What I'm trying to say here is that I need constant input on what I'm doing, and how I can improve on it.

When I was first told about my rocking from side to side, I was horrified. I took it personally.

"What did people think of me? I must have looked odd!"

But the more feedback I've got, the more I've learnt to not take it personally: to rather find it interesting.

In a scientific way. I've found that rather than wondering what people think of me, by trying to understand why I'm doing these oddities, the easier it is to tackle them.

The reason I'm giving you this viewpoint is that you should solicit feedback about your teaching if you want to improve. And that it's just once person's point of view. Try not to take it personally. Understand it. If it has merit try to fix it. If it doesn't, note it. It's just how you improve!

And remember, just have we have points for improvement, we also have things we do really well.

So I ask for feedback in order to improve: but I always ask for one point of improvement, and for one thing that I've done really well. That way I keep my equilibrium.

Never teach alone if you can help it! And solicit feedback from your fellow teachers!

So now we are going to take turns in teaching a short extract from each of our chosen lessons.

```
Work out a suitable time for each person to teach for
```

At the end of each timed session we the learners will give the teacher a heads up on one thing we thought they can improve on, and the one thing that they do really well. Just for practice.

```
If there are a large number of teachers, do a random sampling
```

Finally, before we launch into this practice:

-- *Slide* --

Nerves!

-- *Slide End* --

I get stage fright: how do I handle it?

There's a very simple technique.

-- *Slide* --

"*You're Excited, Not Nervous. You Just Keep Telling Yourself That.*"

-- *Slide End* --

You look forward to the event: its an opportunity. Don't fear it!

Ok: let's present!

```
Now present and do the feedback...
```

Feedback

Now that we've all had turns presenting and getting feedback, let's turn our attention to our getting feedback for the course on the day.

We have found the Software Carpentry technique of using sticky notes to be very handy. And the multi-choice questions also give a really good idea as to who needs more explanation. They also allow us to break up the proceedings with some humour.

The multi choice question answer cards are a real pain to manage and distribute. Having each answer a different colour works really well in allowing us to see who has answered what without having to read the cards.

We've also found that the use of minute cards is brilliant:

To recap: at the end of a session, learners write one thing

- that they've learnt on their green sticky note
- that they found confusing on their red sticky note

-- *Slide* --

Sample feedback from a course

Voyeurs!

http://tinyurl.com/ResOsTnT

-- *Slide End* --

Instructor feedback is also something we practice.

Although I suspect that this might be largely self affirming. In which case do we get value from it?

The post workshop assessment is another practice that the University follows.

To do this we use the Net Promoter Score metric.

We send out an email customized to the session, along the lines of:

-- *Slide* --

Thank you so much for attending the basic high performance computing with NeCTAR course yesterday.

If you haven't done so already, please take two minutes to fill out this anonymous two-question survey on how you felt the training went:

https://docs.google.com/forms/d/18-FVw2R3svRyh6AE4ropHcors0wc50tqoBX5R5dUOOoo/viewform

For those of you who asked for more links to material on the cloud side:

We covered the basic concepts:

- http://training.nectar.org.au/package01/sections/all.html
- http://training.nectar.org.au/package05/sections/all.html

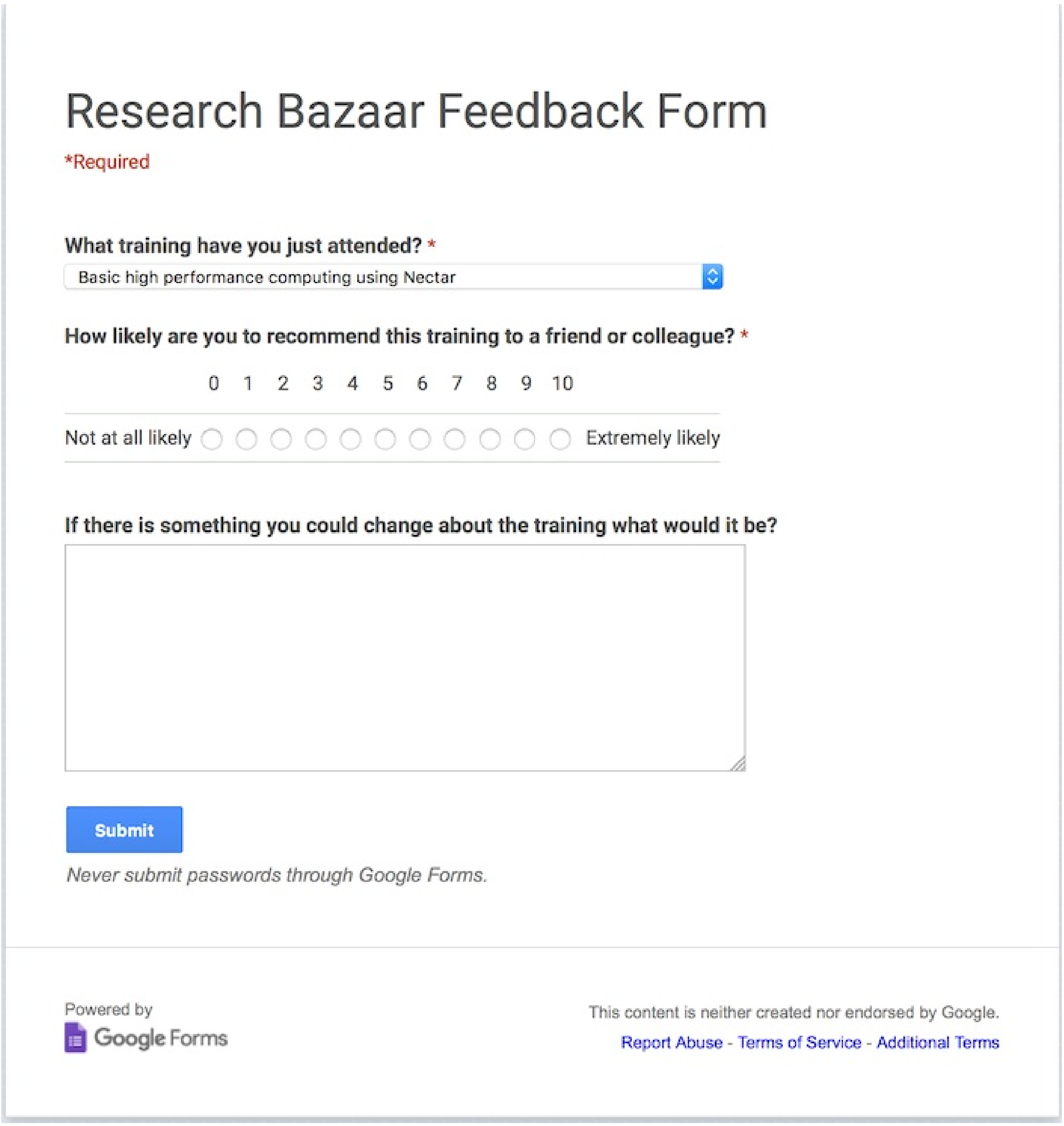
...

-- *Slide End* --

When the learners click on the link they see the following Google form:

-- *Slide* --

Feedback form



Research Bazaar Feedback Form

*Required

What training have you just attended? *

Basic high performance computing using Nectar

How likely are you to recommend this training to a friend or colleague? *

	0	1	2	3	4	5	6	7	8	9	10	
Not at all likely	○	○	○	○	○	○	○	○	○	○	○	Extremely likely

If there is something you could change about the training what would it be?

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google.

Report Abuse - Terms of Service - Additional Terms

-- *Slide End* --

The form asks for one question on a numeric scale to be answered. Those who answer

- 9 - 10 are seen as promoters
- 7 -8 are seen as passive
- Everything below this is seen as being detractors.

-- *Slide* --

Net Promoter Score

For more on this:

http://tinyurl.com/ResOsNps

-- *Slide End* --

In line with Software Carpentry we get very poor returns: at the moment we are at about 30%. But on the upside we've had no response below 7, and the comments have all been most usefull.

-- *Slide* --

Summary feedback

We haven't trialed summary feedback yet :(

-- *Slide End* --

-- *Slide* --

True or False: We only use 10% of our brain...

- G = True.
- R = False.

-- *Slide End* --

A False: If true, we should be able to survive the loss of 90% of our brain quite happily...

Also, modern scanning techniques have shown that this is false. Simple activities, such as walking into a room full of people, or playing a musical instrument, cause all of your brain to light up!

Preparing to present

The course itself has a set of instructions on its home page.

But we have an extended "checklist" on the readme of this course.

-- *Slide* --

Have we missed anything?

http://tinyurl.com/ResOsPn

- G = True.
- R = False.

-- *Slide End* --

How learning works

Software Carpentry attempts to build on the findings of current research.

So:

-- *Slide* --

Analogies

Students learn new ideas by building on what they know.

-- *Slide End* --

This can be both a help and a hindrance.

A help if their existing knowledge is good. A hindrance otherwise.

So our material must:

- Use analogies that students are likely to understand
- Introduce concepts in sequence, building on what has gone before

-- *Slide* --

Memory

From Working -> Long Term

-- *Slide End* --

Working memory has very limited capacity.

So our material:

- Can use scaffolding to reduce cognitive overload
- But must gradually remove scaffolding
- Should use multiple modalities (but don't split students attention)
- Must be well paced.

-- *Slide* --

Where did I put my keys?

What goes into memory often vanishes almost straight away

-- *Slide End* --

We can address this by

- Asking questions
- Telling stories
- Introducing mnemonics

-- *Slide* --

Practice is good

But not all practice is equivalent.

-- *Slide End* --

- Spaced practice is good
- Quizzes help
- Interleaving practice is more effective

-- *Slide* --

Feedback is needed

And it should be:

- Specific
- Clear
- Focused on the task
- Explanatory

-- *Slide End* --

All of this is the inverse of http://www.barbaraoakley.com/pdf/10rulesofstudying.pdf, really.

How to contribute changes back to the course

Why are we telling you all of this?

Well, we'd like you to help us improve the material in the course.

And this stuff helps with course preparation.

But in contributing back, we dont' want to as painful as the tier 0 documentation!

-- *Slide* --

Has anyone edited tier 0 documentation?

- G = Yes
- R = No

-- *Slide End* --

So either:

- submit a bug report

Or we are thinking:

- fork the repository, make your fix and submit a pull request.

If following the latter course, it would be good to discuss your fixes first if they make a lot of changes.

https://help.github.com/articles/fork-a-repo/

DISCUSS thoughts about this, and also about a support group vs. mailing list.

-- *Slide* --

Thank you

- We'd like your feedback
- http://tinyurl.com/ResOsFb

-- *Slide End* --