

AP/ITEC 2210 3.0 A: System Administration Fall 2023

Instructor: Jamon Camisso

ITEC 2210 Chat: mattermost.itec2210.ca

Email: jamon@yorku.ca

Website: <https://eclass.yorku.ca/>

Date/Time: Wednesday, 19:00-22:00

Location: Zoom / ACE 003

Office hours: Via Mattermost any time

Lecture 2 - Troubleshooting cont.

- Troubleshooting anecdote:
- Had an issue with launchpadlibrarian.net - our build farm storage cluster that tracks metadata like build logs, mirror probes etc. ~180TB of space with 3X data replication.
- Users were reporting intermittent timeouts, with no actual build errors
- Alerts showed it had been happening for a month or so

Lecture 2 - Troubleshooting cont.

- Troubleshooting anecdote:
- My colleagues, manager, even the Director of IS Operations had all tried to figure it out, to no avail
- This happened first semester I taught this course after the troubleshooting lecture - I decided to solve the problem
- 1. Started with a catch up discussion with a colleague who had spent a day on the issue, which he inherited from another colleague (rubber duck session, but with a person)

Lecture 2 - Troubleshooting cont.

- Troubleshooting anecdote:
- They had been looking at every system involved, and when I was brought in, the task they were on was tcpdumping and looking at core routers between datacentres.
- Sounds familiar from last week? Happens all the time when the issue is vague and you're fumbling around

Lecture 2 - Troubleshooting cont.

- Troubleshooting anecdote:
- Colleagues had all been bisecting, looking at various components and trying to prove or disprove their theories about which was misbehaving
- I decided to start at the beginning with HTTP errors per user reports, eliminate, and work backwards

Lecture 2 - Troubleshooting cont.

- Troubleshooting anecdote:

- Architecture looks like this:

- Apache web front-ends -> Haproxy load-balancers
- Haproxy load-balancers -> Librarian App processes
- Librarian App processes -> Swift storage proxy
- Swift storage proxy -> Backend storage nodes
- Backend storage nodes -> individual /dev/sd* disks

Lecture 2 - Troubleshooting cont.

- Troubleshooting anecdote:

- Started at the beginning:

- Apache logs we showing generic timeout messages, but importantly, no errors
- Haproxy status page shows backend App processes were up and healthy, with a 5% or so error rate
- Backend App processes showed application logic was fine, but incoming files were timing out being written

Lecture 2 - Troubleshooting cont.

- Troubleshooting anecdote:
- At this point, all HTTP traffic was flowing as expected, so it could be eliminated
- Swift proxy, and all swift backend servers were responding normally, no alerts, performance looked fine. Start eliminating again and look at logs

Lecture 2 - Troubleshooting cont.

- Troubleshooting anecdote:
- Swift proxy logs showed something interesting:
- Jan 11 09:13:53 uwan proxy-server: ERROR with Object server 10.34.2.7:6000/sdc re: Trying to HEAD /v1/AUTH_4e55663613384335a47336ffe0dd9727/librarian_6485/3242856883: ConnectionTimeout (0.5s) (txn: tx6fceb7689fb1443cb059b-005c385e50) (client_ip: 91.189.89.136)

Lecture 2 - Troubleshooting cont.

- Troubleshooting anecdote:
- First promising lead showing the `ConnectionTimeout` error, which maps precisely to HTTP timeouts users were seeing
- Also helpful: a 'txn' that can be used to look for corresponding requests across servers (grep `tx6fceb7689fb1443cb059b-005c385e50` through logs)
- Better still, an IP address of a backend swift storage node

Lecture 2 - Troubleshooting cont.

- Troubleshooting anecdote:
- Said storage node had recently (as of Dec. 15) had a drive replaced.. See where this is going? All logs showed that IP.
- Jan 11 18:44:57 takam object-replicator: rsync:
get_xattr_data:
lgetxattr("/srv/node/sdb/objects/138882/fffb/87a09fc44e6d
8108ab480b4397f81fffb/1448600942.51777.data","user.swif
t.metadata",0) failed: Permission denied (13)

Lecture 2 - Troubleshooting cont.

- Troubleshooting anecdote:

- On that host, takam, /srv/node/sdb was indeed the drive that had been replaced! Permission denied is damning:

- jamon@takam:/srv/node/sdb\$ ls -alh /srv/node/sdb

drwxr-xr-x	4	nagios	nagios	31 Mar 17	2015	accounts
drwxr-xr-x	97	nagios	nagios	4.0K Aug 12	2017	async_pending
drwxr-xr-x	28	nagios	nagios	330 Apr 29	2016	containers
drwxr-xr-x	18851	nagios	nagios	404K Jun 22	2018	objects
drwxr-xr-x	2	nagios	nagios	6 Aug 12	2017	tmp

Lecture 2 - Troubleshooting cont.

- Troubleshooting anecdote:
- Knowing about swift, and looking at say, /srv/node/sdc (a working drive) owner and group were swift:swift
- jamon@takam:~\$ sudo ls -alh /srv/node/sdc
 - drwxr-xr-x 10 swift swift 109 May 11 2015 accounts
 - drwxr-xr-x 2 swift swift 6 Jan 16 10:43 async_pending
 - drwxr-xr-x 914 swift swift 20K Dec 21 09:05 containers
 - drwxr-xr-x 11918 swift swift 492K Jan 16 14:02 objects
 - drwxr-xr-x 3 swift swift 20 Sep 9 2015 quarantined

Lecture 2 - Troubleshooting cont.

- Troubleshooting anecdote:
- I changed user/group using 'chown -R swift:swift' and once that completed (took 2 days) the errors went back to normal
- Anyone could have done the same rubber duck debugging, elimination, successive refinement, and looking for logs
- Point is, you have to be methodical, aware of ongoing work in your team, and have proper logging to save days of work

Lecture 2 - Troubleshooting cont.

- Troubleshooting anecdote:
- We've since built monitoring to check for incorrect permissions
- Implemented a +1 process to swift specifically (get a peer to review work before committing changes)
- Correlated past alerts since Dec. 15 and sent out an incident report (internally) to ensure we hold ourselves accountable

Lecture 2 - Launching a New Service

- Readings:

- PSNA chapter 19 - Service Launch: Fundamentals

- New service launch can range in scope

- Could be something small like a connectivity checker
 - Could be large like iTunes store, serving millions of new devices the first day of a new product launch

Lecture 2 - Launching a New Service

– First priority: plan for problems

“A clever person solves a problem. A wise person avoids it.”

-- Albert Einstein

Lecture 2 - Launching a New Service

- **First priority: plan for problems**
- Like everything else SA, the process and knowledge you gain accumulates over time
- Planning around problems means when they do arise, you and everyone else involved will know exactly what to do
- Defining and planning for problems is one of the best ways to refine processes - if everything always went right and worked, there'd be no incentive to improve processes

Lecture 2 - Launching a New Service

- First priority: plan for problems
- NASA example of Apollo 7, 8, 12 launches is a great one:
- Even in a space ship with thousands of subsystems, each one had well thought out failure scenarios and playbooks
- Those launches had problems, but they were planned for
- When things went wrong with Apollo 13, they were wrong because the cause of the problem was unknown

Lecture 2 - Launching (an Apollo mission)



Lecture 2 - Launching a New Service

- Hopefully your SA roles won't be a matter of life and death, but planning for problems (and there were many with Apollo missions) worked
- Listen to every team checking in & confirming their launch checklist at ~2m:30s. Then listen to troubleshooting an unplanned for problem.
- 6:11 “Everybody, think of the kinds of things we'd be venting”
- 7:32 “Let's not make it any worse by guessing”
- 8:18 “The thing that concerns me is we had a problem, we don't know the cause of the problem - I don't know why we've lost them”

Lecture 2 - Launching a New Service

- **The Six Step Launch Process:**

- Two main components: **ready list**, and **iteration**

- Ready list is a living document, and changes as requirements and features are added, bugs and regressions are found, etc.

- Launch itself is iterative, with phases contributing back to the overall ready list, until everything is live

Lecture 2 - Launching a New Service

– The Six Step Launch Process:

- 1. **Define** the ready list
- 2. **Work** the ready list
- 3. Launch the **beta** service
- 4. Launch the **production** service
- 5. **Capture** lessons learned
- 6. **Repeat**

Lecture 2 - Launching a New Service

- **Define the ready list:**

- A list of tasks or assertions, that when completed means things are ready for launch
- It is the main measure of progress in preparing for a launch
- Usually has four categories of items:

Lecture 2 - Launching a New Service

- Define the ready list:
- 1. **Must have** features
- 2. **Would be nice** features
- 3. **Bugs and regressions**
- 4. **Assertions and approvals**

Lecture 2 - Launching a New Service

- Define the ready list:

- 1. **Must have** features

- Any must have feature is an agreed upon commitment to deliver a specific feature, by a given time within a release process
- Use this list as a contract between you (the SA and your IT organization) and the customer to keep everyone informed and set expectations

Lecture 2 - Launching a New Service

- Define the ready list:

- 2. Would be nice features

- Crucial to keep track of these, and separate from must have features, even if you know they won't be included
- Keeps everyone on the same page in terms of expectations, and keeps you from being distracted from the must have list (cf. one big list of everything)

Lecture 2 - Launching a New Service

- Define the ready list:

- 3. Bugs and regressions

- Bugs happen, and sadly so do regressions. Track and triage them so that they're all documented.
- Tracking also reduces duplication of effort, since everyone can see who is working on a bug

Lecture 2 - Launching a New Service

- Define the ready list:

- 4. Assertions and Approvals

- Everything that must be 'true' before launch
- Technical tasks, and organizational ones
 - Legal, communications, and marketing approval
 - CEO has signed off
 - Infrastructure capacity is confirmed
 - Connectivity is established and reliable etc. etc.

Lecture 2 - Launching a New Service

– List attributes

– Each item should have at least:

- Title
- Priority
- Owner
- History of the task
- Current status

Lecture 2 - Launching a New Service

– List attributes

– Priority is worth looking at more closely:

- Standardize across projects. In many organizations, people move between groups. A standard ensures continuity, and that everyone has the same understanding of what 'critical' or 'low priority'
- Not required priority is a nice way of making things visible, and not being dismissive

Lecture 2 - Launching a New Service

– S.M.A.R.T Goals and Objectives

- **Specific:** The goal addresses a specific area for improvement
- **Measurable:** The goal's success (or progress) can be quantified
- **Achievable:** It can be realistically achieved given available resources
- **Relevant:** It is worthwhile, and in line with broader goals
- **Time-bound:** The goal has a deadline

Lecture 2 - Launching a New Service

- S.M.A.R.T Goals and Objectives: example
- Title: “Deploy VRRP on production load balancers”
 - **Specific:** the task is to do one thing - could have a sub-task to ‘test VRRP...’ with same list attributes
 - **Measurable:** Either it is or isn’t deployed
 - **Achievable:** Less important than ‘Write a full CMS’
 - **Relevant:** Not important if there’s no need for it
 - **Time-bound:** Next week, next month, or next year?

Lecture 2 - Launching a New Service

- **Ready list visibility**

- A list is no good if only you can see it, so share it widely
- Sharing ensures everyone buys in, and makes it a central point of reference, instead of having silos each doing their own thing
- If you have external stakeholders, you may only share a subset of the list, or even subset of attributes of each item

Lecture 2 - Launching a New Service

- **Ready list visibility**

- **Storage and distribution methods:**

- **Bug or ticket system with tags**
- **Bug or ticket system with watchlists**
- **Kanban board**
- **Online/shared spreadsheet**
- **Offline coordinator** (e.g. project manager, release coordinator)

Lecture 2 - Launching a New Service

- Do it!

- Set clear expectations (again, beginning with title)

- Make sure there's some kind of deadline

- Try to let the **delegate** set deadline
- Most of the time it will be sooner than expected
- Doing it this way lets people have agency
- Obviously, bear in mind overall schedule constraints

Lecture 2 - Launching a New Service

- Do it!

- Monitoring progress

- Some groups use a weekly status meeting to go over ready list, re-triage features, bugs, timelines

- Many teams will use a 'stand up' meeting each day where everyone stands up (to keep meetings short)

- What was done, what you're doing, what's blocking

Lecture 2 - Launching a New Service

- **Launching a beta service**

- A service or a process can be in beta, just as much as a piece of software
- Use a **staging area** to deploy pre-release version
- You'll want a few different areas to use in staging

Lecture 2 - Launching a New Service

– Launching a beta service: staging environments

- **Dev**, also known as sandbox
- **QA**, lands developer releases, accepts or rejects based on tests, and generates a list of bugs or regressions
- **UAT**, User Acceptance Testing, for external customers to **approve** a release
- **Beta**, a subset of customers or users get early access
- **Production**, live users of a service

Lecture 2 - Launching a New Service

– Launching a beta service: staging environments

- Regardless of Agile, XP, waterfall, there need to be well-defined criteria to promote from one environment to another
- Dev -> QA push can be automated, but QA can entail manual tests that must pass, thus they **gate** the release
- **Assertions** and **approvals** are also gating criteria

Lecture 2 - Launching a New Service

– Launching a production service

- Do the launch once everyone has agreed the ready list up to the production roll out is complete
- Launch is just doing technical tasks that remain, while being sure to communicate them to everyone involved
- Always try to do gradual roll-outs, e.g. 10% of users, then 20%, then 50%, then 100%

Lecture 2 - Launching a New Service

– Capture lessons learned: launch retrospective

- Agree about needed improvements
- Educate everyone about issues visible to only a few
- Explain problems to management, particularly ones large enough that their resolution will require additional resources
- Share what you learned to the entire organization so all can benefit.

Lecture 2 - Launching a New Service

– Capture lessons learned: launch retrospective

- Schedule a live meeting for the retrospective
- Include remote participants, or record for later review
- Design a retrospective document
- Useful to share issues, discuss shortcomings, and keep others from repeating mistakes, or duplicating effort

Lecture 2 - Launching a New Service

– Capture lessons learned: launch retrospective

- Document should contain:

- Executive summary - what happened, when, good and bad, what to do next time. KISS principle
- Timeline - start, finish, significant milestones
- Successes - Yay
- Failures - blameless! Stick to facts
- Short-term changes - what to do better next time
- Long-term changes - what to fix in the future

Lecture 2 - Launching a New Service

– Capture lessons learned: launch retrospective

- Important to feed any short term and long term changes back into the bug/ticketing system
- Ensures that subsequent releases track issues, and that nothing is forgotten if it wasn't addressed in the launch of the service
- In the Apollo 13 example, you can be sure failures were tracked, new temp. sensors were used in 14-17

Lecture 2 - Launching a New Service

– Repeat

- Most software spends most of its life running and needs maintenance, new releases, or upgrades
- Same process for a new service can be used, just on a smaller scale or time frame

Lecture 2 - Launching a New Service

– Launch Readiness Criteria

- **Launch criteria** are not the same as launch procedures
- **Criteria** are assertions, or non-technical steps that need to be completed, e.g. load testing is done, scaling is planned
- **Procedures** are the checklist of steps to complete a launch, like DNS updates, firewall changes etc.

Lecture 2 - Launching a New Service

– Launch Readiness Criteria

- Criteria lists grow over time, but a basic list will need:
 - Monitoring and escalation policies in place
 - Backups and restores tested and working
 - Authentication, authorization, access control
 - SLA (service level agreement with customer)
 - Support/service request tool and process in place
 - Documentation is complete (how to use this?)
 - Ops Documentation is complete (how to run this?)
 - User training is done. Otherwise, what's the point?
 - Load testing is complete (does it work under load?)

Lecture 2 - Launching a New Service

– Launch Readiness Criteria

- Criteria lists grow over time, and become unwieldy
- Be cautious about adding new items, and diligent about removing obsolete items
- Each item means work for someone. Gradually, people can become fearful of the LRC list, work around it, delay because of it, or just ignore it altogether

Lecture 2 - Launching a New Service

– Launch Readiness Criteria

- Add items based on actual experiences, not hypotheticals
- Update and rewrite items, and remove what you can
 - Point about being liable for removing is important
 - Have everyone agree to remove an item
- Use automation to remove items! E.g. sandbox setup, database pushes from prod -> beta -> UAT -> QA -> Dev

Lecture 2 - Launching a New Service

– Launch Calendar

- Have an organization-wide launch calendar for future launches
- Lets other schedule around, or with your launch
- Keeps everyone accountable by broadcasting a commitment to a date, even if it will be missed
- In non-Agile environments, define repeat windows

Lecture 2 - Launching a New Service

– Common Problems - List?



Lecture 2 - Launching a New Service

– Common Problems

- Failure in production
- How many times have I heard this:
- “It worked on localhost”
- Stems from different teams owning different infrastructure, or drift over time
- Use the same processes in every environment

Lecture 2 - Launching a New Service

– Common Problems

- **Unexpected Access Methods**
- Testing from an office connection on 1gbit fibre is different from home over a VPN, which is different from a 3G network etc.
- Timeouts are the norm, so test for them from the start

Lecture 2 - Launching a New Service

– Common Problems

- Not enough resources
- Do capacity planning at the beginning, order hardware as soon as is reasonably possible
- Having SAs join stand-ups, or weekly meetings will help catch assumptions and set expectations about capacity early on

Lecture 2 - Launching a New Service

– Common Problems

- New technology failures
- Sometimes new technologies are decided upon without those involved communicating how difficult they are to set up, monitor, scale
- Best approach is to set up, destroy, and repeat until the process is completely automated, before going into production

Lecture 2 - Launching a New Service

– Common Problems

- Lack of User training
- Users get used to things very quickly, and many are averse to change
- Overhauls and new services can be very disruptive
- Having a dedicated staging area for training is helpful

Lecture 2 - Launching a New Service

– Common Problems

- No backups
 - Happens everywhere: “we’ll do backups at the end”
 - Then you have file servers that are out of sync
 - Or database snapshots that take too long
 - Worse than no backups, are untested backups. They’re a waste of people’s time designing them, computing resources, and give a false sense of security