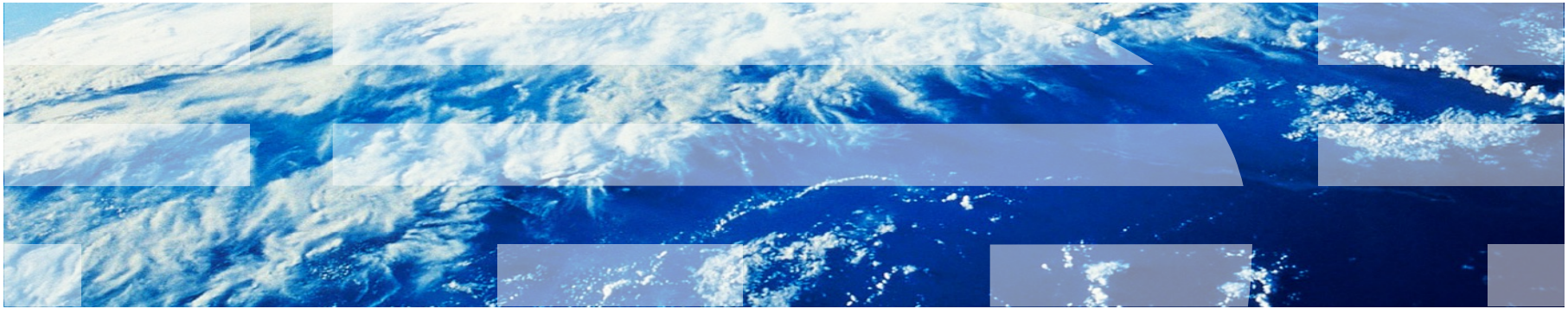
WebSphere Application Server

Liberty Performance Tuning Hands-On Lab



Agenda

- Lab Overview and Key Performance Tuning Principles (15 minutes)
- Lab (2 hours, 15 minutes)

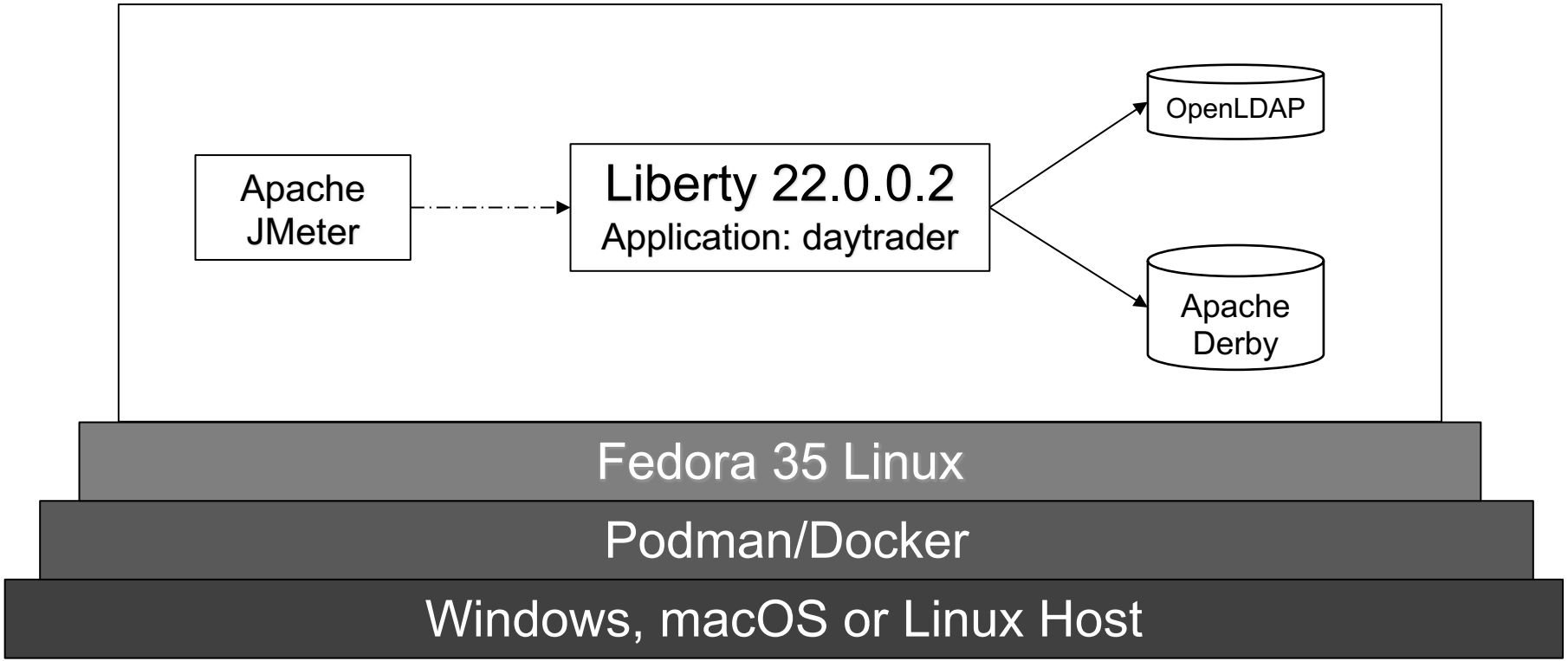
Lab Timeline

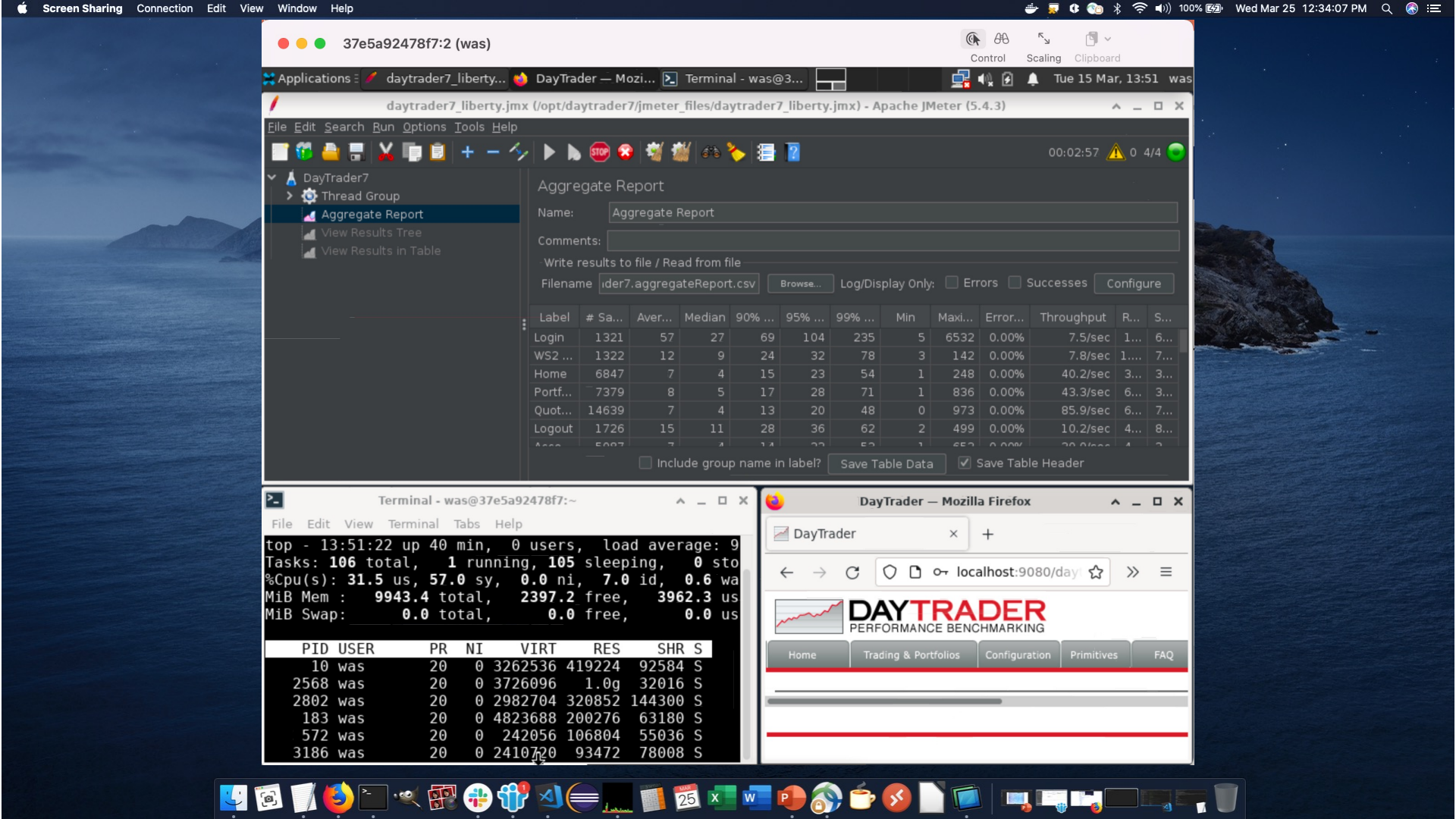
- 00:00 – 00:15: Outline, Lab Overview, and Key Performance Tuning Principles
- 00:15 – 00:25: Running the lab and the performance test
- 00:25 – 00:45: Using thread dumps
- 00:45 – 01:15: Analyzing garbage collection
- 01:15 – 01:20: Break
- 01:20 – 01:50: Using a sampling profiler
- 01:50 – 02:00: Presentation: Top 10 Tuning Tips
- 02:00 – 02:10: Liberty Request Timing
- 02:10 – 02:20: Liberty HTTP Access Log
- 02:20 – 02:30: General Q&A

Lab Overview

- Self-paced, free, publicly downloadable Liberty performance tuning lab based on containers
- Over 100 pages of exercises which can be done in sequence or a la carte
- If you can't install Podman or Docker Desktop, or you can't download the 20GB lab now, the instructor will be running the lab and you can watch.
- For those running the lab, you can mute the instructor and come back at preset times
- Today, we'll cover the most common areas of Liberty performance tuning:
  - Thread dumps
  - Garbage collection
  - Profiling
  - Top 10 tuning tips
  - Request timing and HTTP access logs

Lab Overview




Screen Sharing Connection Edit View Window Help
100% Wed Mar 25 12:34:07 PM
37e5a92478f7:2 (was)
Control Scaling Clipboard
Applications daytrader7_liberty... DayTrader — Mozi... Terminal - was@3... Tue 15 Mar, 13:51 was
daytrader7_liberty.jmx (/opt/daytrader7/jmeter_files/daytrader7_liberty.jmx) - Apache JMeter (5.4.3)
File Edit Search Run Options Tools Help
00:02:57 0 4/4
DayTrader7
Thread Group
Aggregate Report
View Results Tree
View Results in Table
Aggregate Report
Name: Aggregate Report
Comments:
Write results to file / Read from file
Filename ider7.aggregateReport.csv Browse... Log/Display Only: Errors Successes Configure
Label # Sa... Aver... Median 90% ... 95% ... 99% ... Min Maxi... Error... Throughput R... S...
Login 1321 57 27 69 104 235 5 6532 0.00% 7.5/sec 1... 6...
WS2 ... 1322 12 9 24 32 78 3 142 0.00% 7.8/sec 1.... 7...
Home 6847 7 4 15 23 54 1 248 0.00% 40.2/sec 3... 3...
Portf... 7379 8 5 17 28 71 1 836 0.00% 43.3/sec 6... 3...
Quot... 14639 7 4 13 20 48 0 973 0.00% 85.9/sec 6... 7...
Logout 1726 15 11 28 36 62 2 499 0.00% 10.2/sec 4... 8...
Include group name in label? Save Table Data Save Table Header
Terminal - was@37e5a92478f7:~
File Edit View Terminal Tabs Help
top - 13:51:22 up 40 min, 0 users, load average: 9
Tasks: 106 total, 1 running, 105 sleeping, 0 sto
%Cpu(s): 31.5 us, 57.0 sy, 0.0 ni, 7.0 id, 0.6 wa
MiB Mem : 9943.4 total, 2397.2 free, 3962.3 us
MiB Swap: 0.0 total, 0.0 free, 0.0 us
PID USER PR NI VIRT RES SHR S
10 was 20 0 3262536 419224 92584 S
2568 was 20 0 3726096 1.0g 32016 S
2802 was 20 0 2982704 320852 144300 S
183 was 20 0 4823688 200276 63180 S
572 was 20 0 242056 106804 55036 S
3186 was 20 0 2410720 93472 78008 S
DayTrader — Mozilla Firefox
DayTrader
localhost:9080/day
DAYTRADER
PERFORMANCE BENCHMARKING
Home Trading & Portfolios Configuration Primitives FAQ

How to run it?

1. Install podman or Docker Desktop: https://ibm.biz/liberty_performance_lab_install
2. Run the container from Command Prompt or Terminal:
   - podman/docker run --cap-add SYS_PTRACE --cap-add NET_ADMIN --ulimit core=-1 --ulimit memlock=-1 --ulimit stack=-1 --shm-size="256m" --rm -p 9080:9080 -p 9443:9443 -p 9043:9043 -p 9081:9081 -p 9444:9444 -p 5901:5901 -p 5902:5902 -p 3390:3389 -p 9082:9082 -p 9083:9083 -p 9445:9445 -p 8080:8080 -p 8081:8081 -p 8082:8082 -p 12000:12000 -p 12005:12005 -it quay.io/kgibm/fedorawasdebug
3. Wait 2 minutes until you see:

```
=========
= READY =
=========
```

How to run it?

4. Remote into the container:
    1. VNC to localhost:5902
        1. From the Terminal in macOS:
           open vnc://localhost:5902
        2. Linux Terminal:
           vncviewer localhost:5902
        3. Windows 3rd party VNC viewers
           OR
           Windows Remote Desktop: Requires configuration; see lab appendix
    2. Password = websphere
5. Perform the step-by-step lab: https://ibm.biz/liberty_performance_lab_start

Key Performance Tuning Principles

- Most performance gains are found in a handful of areas:
  - Tuning the Java garbage collector
  - Tuning various pools (JDBC connections, authentication cache, etc.)
  - Tuning the operating system
- Beyond those, the main thing to do is to be data-driven:
  - Find issues by gathering and analyzing thread dumps, verbose garbage collection, sampling profiler data, request timing, access logs, etc.
  - Use a realistic performance test environment with repeatable tests and basic statistics

Major Recommended Tools

Tool	Analyze	Purpose
Thread and Monitor Dump Analyzer (TMDA)	Thread dumps	What is Liberty doing?
Garbage Collection and Memory Visualizer (GCMV)	Verbose garbage collection	Garbage collector overhead
IBM Java Health Center	Sampling profiler	CPU Deep Dive
Liberty Request Timing	Liberty logs	Slow HTTP responses
Liberty HTTP Access Log	Liberty logs	HTTP response statistics

References

- Start the lab: https://ibm.biz/liberty_performance_lab_start

Demo

Thread Dumps Lab

- Lab link: https://github.com/kgibm/dockerdebug/blob/master/fedorawasdebug/Liberty_Perf_Lab.md#ibm-java-and-openj9-thread-dumps
- Thread dumps tell you what is happening inside Liberty

Thread Dumps Lab Summary

- In general, thread dumps are non-destructive, cheap, and easy to get
- If you're having any problem, it's always a good idea to get thread dumps
- Use the WebSphere Support MustGather (e.g. linperf.sh) to get thread dumps and OS stats
- Use the free IBM Thread and Monitor Dump Analyzer tool to analyze them
- Review lock contention
- Review thread stacks, sort by stack depth descending, and look for patterns
- Use the Compare threads function to analyze multiple thread dumps over time

Garbage Collection Lab

- Lab link: https://github.com/kgibm/dockerdebug/blob/master/fedorawasdebug/Liberty_Perf_Lab.md#garbage-collection
- Healthy garbage collection should be less than ~5-10% of process time

Garbage Collection Lab Summary

- In general, verbose garbage collection should always be enabled, even in production
- For performance issues, always review verbose GC
- Use the free IBM Garbage Collection and Memory Visualizer tool
- Crop to the time period of interest, click Report, and review “Proportion of time spent in garbage collection pauses (%)”
- In general, healthy GC is less than ~5-10%

5 minute break

Java Profiler Lab

- Lab link: https://github.com/kgibm/dockerdebug/blob/master/fedorawasdebug/Liberty_Perf_Lab.md#health-center
- Analyze CPU hot spots

Java Profiler Lab Summary

- A sampling profiler is kind of like taking hundreds or thousands of thread dumps at a very high frequency
- Generally, Health Center is production ready with an overhead of < ~2%
- Use the free IBM Health Center client to review the data
- Zoom to a time period of interest
- Review hot Self methods consuming > ~2% of samples
- Review Tree CPU % breakdown by large outgoing call percentage drops
- Review lock contention
- Export thread dumps and load in TMDA to review non-CPU-using activity

Liberty Tuning Top 10 Tips

1. Ensure your operating system CPU, RAM, disk, and network aren't saturated
2. Ensure time in Java garbage collection is less than ~5-10%; tune -Xmx and -Xmn primarily
3. Liberty's main thread pool auto-tunes for throughput and generally should not be tuned
4. Gather and review thread dumps and/or a sampling profiler to find areas to tune
5. If using databases, tune the maximum connection pool size
6. If using JMS MDBs, tune the maxConcurrency
7. If using security, tune the authentication cache size
8. Consider enabling request timing to watch for slow HTTP requests
9. Consider enabling the HTTP access log to understand and tune HTTP activity
10. Consider enabling HTTP response compression and/or caching

For more, see the WebSphere Performance Cookbook

Liberty Request Timing Lab

- Lab link: https://github.com/kgibm/dockerdebug/blob/master/fedorawasdebug/Liberty_Perf_Lab.md#request-timing
- Request timing watches for long-running HTTP requests

Liberty Request Timing Lab Summary

- In general, request timing is recommended to be enabled, even in production
- Set the threshold to your largest expected HTTP response time plus 20%
- Tune sampleRate if needed to bring the overhead down
- Add monitoring to watch for the request timing warning
- Review the stack and tree breakdown to understand what caused the slowdown

Liberty HTTP Access Log Lab

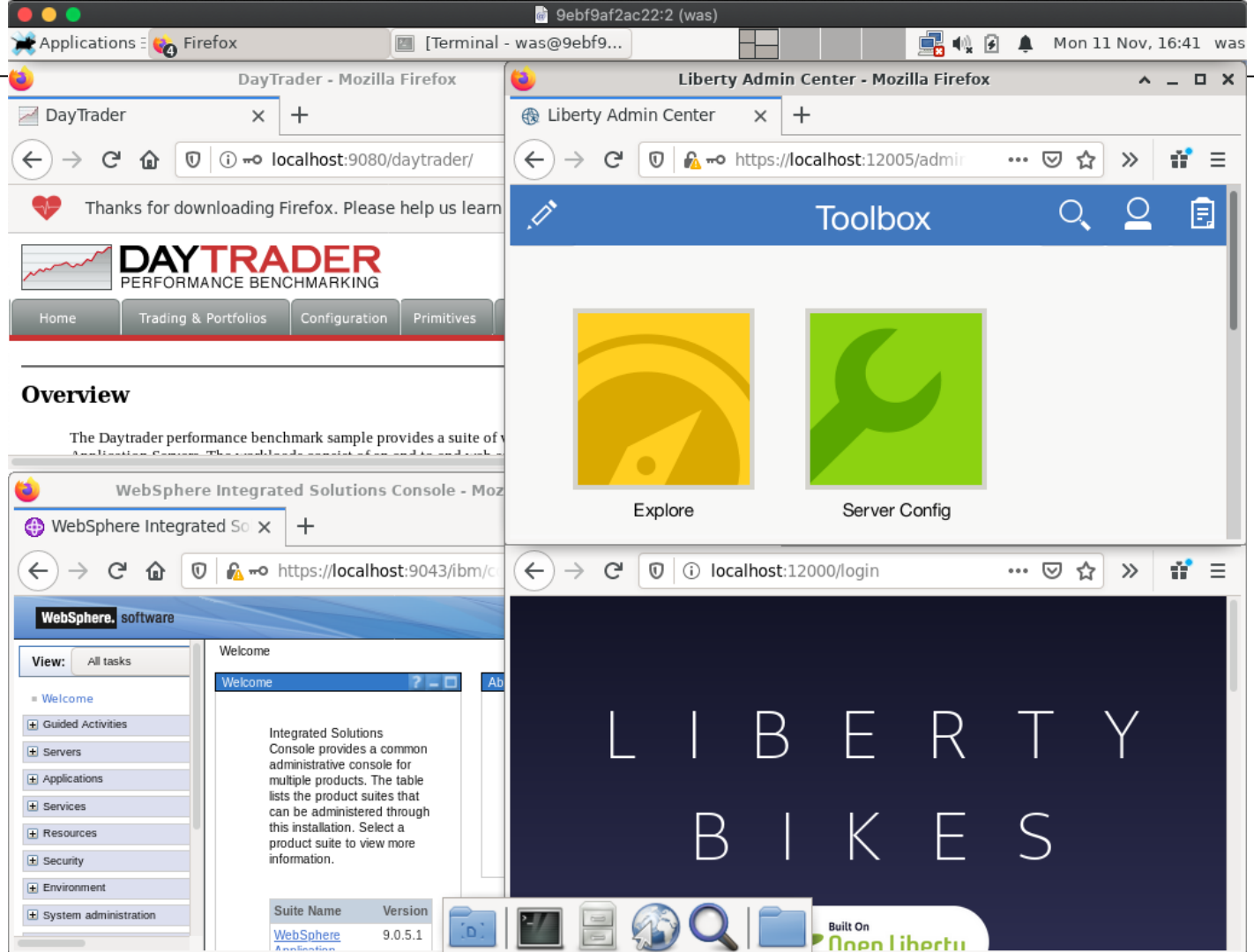
- Lab link: https://github.com/kgibm/dockerdebug/blob/master/fedorawasdebug/Liberty_Perf_Lab.md#http-ncsa-access-log
- Tracks information about every completed HTTP request

Liberty HTTP Access Log Lab Summary

- In general, consider always enabling the HTTP access log if the overhead is acceptable (~2%)
- Use post-processing scripts or tools to understand HTTP activity, errors, etc. over time

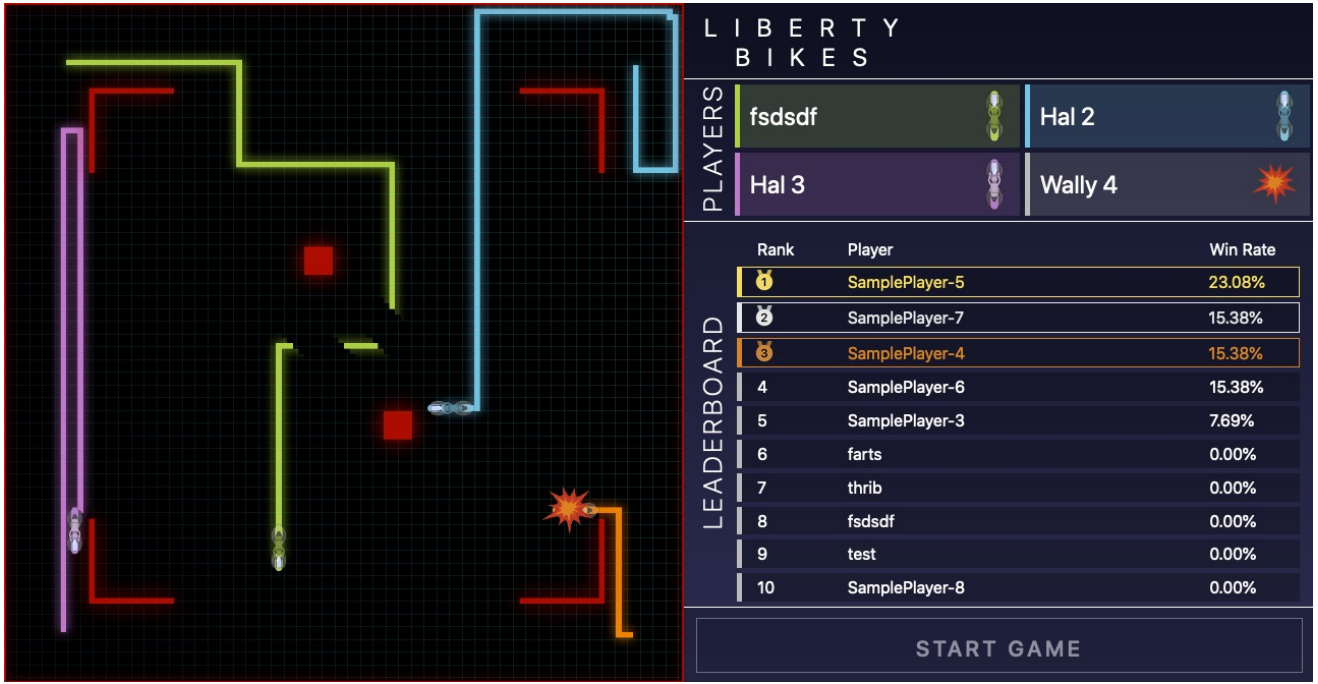
Thank you.
Questions?

Appendix

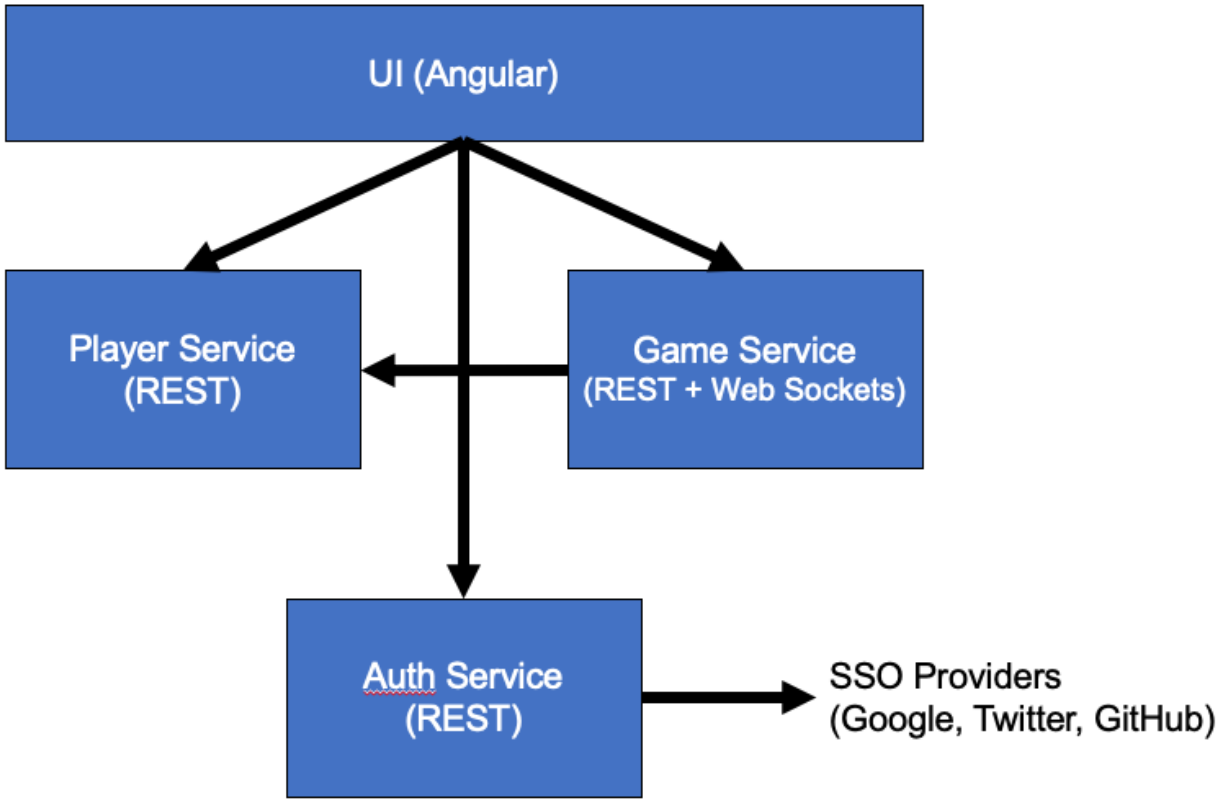


What's in it?

- Liberty Bikes to show off MicroServices: https://github.com/OpenLiberty/liberty-bikes



Liberty Bikes



Tips

- You can share files at **/host/** to use the container to analyze diagnostics:
  - Windows:
    - docker run ... **-v //c/:/host/** -it quay.io/kgibm/fedorawasdebug
  - Linux/macOS:
    - docker run ... **-v /:/host/** -it quay.io/kgibm/fedorawasdebug
- For port conflict with something else running on the host, stop that thing or change the redirect, e.g., if 9080 is already used, change localhost:9081 to point to container:9080:
  - -p 9081:9080
- The lab appendix shows how you can save/restore a container filesystem if needed.
- You don’t need to expose ports at all if you just access everything within the remote desktop.