

TCSS 562:
SOFTWARE ENGINEERING
FOR CLOUD COMPUTING

Cloud Enabling Technology III
&
Containerization

Wes J. Lloyd
School of Engineering and Technology
University of Washington – Tacoma
TR 5:50-7:50 PM



1

OFFICE HOURS – FALL 2022

THIS WEEK

Tuesday:

4:30 to 5:30 pm - CP 229 and Zoom

Thursday*

4:30 to 5:30 pm - CP 229 and Zoom

Or email for appointment

* Rescheduled due to Veteran's Day holiday – Nov 11th

> Office Hours set based on Student Demographics survey feedback

November 10, 2022

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington – Tacoma

L13.2

2

OBJECTIVES – 11/10

Questions from 11/8

Tutorials Questions

Class Presentations:
Cloud Technology or Research Paper Review

Quiz 1

Ch. 5: Cloud Enabling Technology

Containerization

November 10, 2022

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington – Tacoma

L13.3

3

ONLINE DAILY FEEDBACK SURVEY

Daily Feedback Quiz in Canvas – Take After Each Class

Extra Credit
for completing

Announcements

Assignments

Discussions

Zoom

Grades

People

Pages

Files

Quizzes

Collaborations

UW Libraries

UW Resources

Upcoming Assignments

Class Activity 1 – Implicit vs. Explicit Parallelism

Available until Oct 13 at 11:59pm | Due Oct 7 at 7:59pm | -10 pts

Tutorial 1 - Linux

Available until Oct 19 at 11:59pm | Due Oct 13 at 11:59pm | -20 pts

Past Assignments

TCSS 562 - Online Daily Feedback Survey - 10/5

Available until Oct 18 at 11:59pm | Due Oct 6 at 8:59pm | -15 pts

TCSS 562 - Online Daily Feedback Survey - 9/30

Available until Oct 18 at 11:59pm | Due Oct 4 at 8:59pm | -15 pts

November 10, 2022

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington – Tacoma

L13.4

4

TCSS 562 - Online Daily Feedback Survey - 10/5

Started: Oct 7 at 1:13am

Quiz Instructions

Question 1

0.5 pts

On a scale of 1 to 10, please classify your perspective on material covered in today's class:

1

2

3

4

5

6

7

8

9

10

Mostly Review To Me

Equal New and Review

Mostly New To Me

Question 2

0.5 pts

Please rate the pace of today's class:

1

2

3

4

5

6

7

8

9

10

Slow

Just Right

Fast

November 10, 2022

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington – Tacoma

L13.5

5

MATERIAL / PACE

Please classify your perspective on material covered in today's class (39 respondents):

1-mostly review, 5-equal new/review, 10-mostly new

Average – 6.38 (↓ - previous 6.52)

Please rate the pace of today's class:

1-slow, 5-just right, 10-fast

Average – 5.38 (↓ - previous 5.41)

Response rates:

TCSS 462: 21/33 – 63.63%

TCSS 562: 18/26 – 69.23%

November 10, 2022

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington – Tacoma

L13.6

6

Slides by Wes J. Lloyd

L13.1

FEEDBACK FROM 11/8

- From tutorial 4 I don't understand?
PART 8 - QUESTION 4: Report the number of runs with `newcontainer=0`
(these are recycled runtime environments)
QUESTION 5: Report the number of runs with `newcontainer=1`
(these are newly created runtime environments)
- AWS Lambda functions run in special runtime environments hosted using Firecracker microVMs
 - These are mini-VMs with a smaller footprint than full Linux VMs such as those created with `ec2` (IaaS cloud)
 - The lifetime of these VMs can be < 1 hour
 - They are created and destroyed on demand solely for the purpose of executing Lambda function securely in isolation

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.7

7

FEEDBACK - 2

- Does Lambda use Firecracker?**
 - Yes – Firecracker was introduced Nov 2018, and by Fall 2020 it was rolled out globally across all Regions / Availability Zones
 - <https://aws.amazon.com/blogs/aws/firecracker-lightweight-virtualization-for-serverless-computing/>
- What are the disadvantages of the Firecracker?**
 - Firecracker is a microVM
 - It would be inappropriate as a replacement for standard VMs that need to run a full operating system instance
 - For example, VMs that need to run a GUI, or multiple applications simultaneously
- What kind of application we should consider to use Firecracker over the other VMs?**
 - Firecracker VMs are great for hosting serverless functions, microservices (e.g. small bits of code), or a single application with small(er) footprints requiring less memory and where the microVM has a shorter uptime

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.8

8

FEEDBACK - 3

- Not related to the lecture, but 3 days after completing assignment 5, I received an email stating that I had used 85% of my S3 put capacity.
- Could it be a mistake I made or did we have to delete the event trigger from CloudTrail/EventBridge?

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.9

9

FEEDBACK - 4

- AWS accounts only allow 2,000 S3 PUT actions per month for free
- After the free tier, each additional 1,000 PUTS costs \$0.005 (half a penny)
- Cloud Trails created in the management console by default publish event logs to S3 buckets in every region globally !!
- See the GUI column 'Multi-region trail' - it is Yes by default

Name	Home region	Multi-region trail
<input type="radio"/> f22-new	US East (Ohio)	No

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.10

10

FEEDBACK - 5

- There are about 28 AWS regions globally
- I found over 8,000 objects in my "global" cloud trail bucket from some testing I had performed earlier in October
- Each log event may put an object in every region using a separate S3 PUT action
- My bill shows 10 cents for S3 PUT, COPY, POST, and LIST actions (about 20,000 S3 actions)
- The Billing Dashboard does not breakout the specific actions
- To make the CloudTrail specific to one region using AWS CLI:

```
aws cloudtrail update-trail
--name (name-of-your-cloudtrail)
--no-is-multi-region-trail
```

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.11

11

AWS CLOUD CREDITS

- IAM User Accounts Create – please let me know of any issues with these accounts
- If you did not provide your AWS account number on the AWS CLOUD CREDITS SURVEY to request AWS cloud credits and you would like credits this quarter, please contact the professor

October 11, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L4.12


12

Don't Forget to Terminate (Shutdown)
all EC2 Instances for Tutorial 3

Spot Instances:
c5d.large instance @ ~2 cents / hour

\$0.48 / day
\$3.36 / week
\$14.60 / month
\$175.20 / year

AWS CREDITS →→→→→→→→



13

OBJECTIVES – 11/10

- Questions from 11/8
- Tutorials Questions**
- Class Presentations:
Cloud Technology or Research Paper Review
- Quiz 1
- Ch. 5: Cloud Enabling Technology
- Containerization

November 10, 2022

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.14

14

TUTORIAL 0

- Getting Started with AWS
- http://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2022_tutorial_0.pdf
- Create an account
- Create account credentials for working with the CLI
- Install awsconfig package
- Setup awsconfig for working with the AWS CLI

November 10, 2022

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.15

15

TUTORIAL 4 – NOV 6

- Introduction to AWS Lambda with the Serverless Application Analytics Framework (SAAF)
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2022_tutorial_4.pdf
- Obtaining a Java development environment
- Introduction to Maven build files for Java
- Create and Deploy "hello" Java AWS Lambda Function
 - Creation of API Gateway REST endpoint
- Sequential testing of "hello" AWS Lambda Function
 - API Gateway endpoint
 - AWS CLI Function Invocation
- Observing SAAF profiling output
- Parallel testing of "hello" AWS Lambda Function with faas_runner
- Performance analysis using faas_runner reports
- Two function pipeline development task

November 10, 2022

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.16

16

IAM USERS – TUTORIAL 4

- Students completing tutorial 4 with an IAM user account may encounter permission issues
- Please contact the instructor if encountering any issues

November 10, 2022

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.17

17

TUTORIAL 4 - RESUBMISSION

- For tutorial 4 submissions, several submission indicate **Thread.sleep(10000)** was added but the results for the question 6 do not confirm this.
- It is possible that:
 - The provided results from the SAAF Report Generator were from a test run before the **Thread.Sleep()** statement was added to the code
 - OR -
 - The **Thread.Sleep()** statement was added in the incorrect location of the code
 - OR -
 - When opening the CSV output from the Report Generator, the file separator characters were set incorrectly.
- The only separator for a CSV file is the comma " , ". Be sure to correctly open the CSV file in the spreadsheet. Columns can be offset resulting in the wrong answers being provided for Question 6.

November 10, 2022

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.18

18

TUTORIAL 4 – RESUBMISSION - 2

- The sleep statement must go between the START FUNCTION and END FUNCTION comments in the `handleRequest()` method specified as the AWS Lambda function's handler under runtime settings in the AWS Lambda GUI.

```

//*****START FUNCTION IMPLEMENTATION*****
try
{
    Thread.sleep(10000);
}
catch (InterruptedException ie)
{
    System.out.println("Interruption occurred while sleeping.");
}
//*****END FUNCTION IMPLEMENTATION*****

```

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.19

19

TUTORIAL 4 – RESUBMISSION - 3

- **SANITY CHECK:** consider that adding 10 seconds of sleep to your AWS Lambda function will cause the function to run for at least 10 seconds. This will impact the outputs requested for Question 6:

- **avg_runtime** is the server-side (cloud) runtime of the function
- This is the time it takes for the function to run on AWS Lambda (cloud)
- Adding sleep of 10 seconds should increase a function's **avg_runtime**
- **avg_roundTripTime** is the total time for a request from a client (laptop?) to travel to the server (cloud), make the function call, and return.
- If trying to make 50 calls at once on a laptop with a small # of CPU cores this time may be slow
- Adding sleep of 10 seconds should increase a function's **avg_roundTripTime**

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.20

20

TUTORIAL 4 – RESUBMISSION - 4

- **avg_cpuidleDelta** time is the amount of time the Lambda function's Firecracker vCPUs are idle during the function call on the server measured in centiseconds:

100 centiseconds = 1 second
100 centiseconds = 1000 milliseconds

- By default, AWS Lambda functions with 512 MB run in a runtime environment with access to two vCPU cores
- This is the total vCPU idle time for both cores (it is doubled)
- Adding sleep of 10 seconds should increase your function's **avg_cpuidleDelta**
- **How much should avg_cpuidleDelta increase?**

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.21

21

TUTORIAL 5 – NOV 13

- Introduction to Lambda II: Working with Files in S3 and CloudWatch Events
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2022_tutorial_5.pdf
- Customize the Request object (add getters/setters)
 - Why do this instead of HashMap?
- Import dependencies (jar files) into project for AWS S3
- Create an S3 Bucket
- Give your Lambda function(s) permission to work with S3
- Write to the CloudWatch logs
- Use of CloudTrail to generate S3 events
- Creating CloudWatch rule to capture events from CloudTrail
- Have the CloudWatch rule trigger a target Lambda function with a static JSON input object (hard-coded filename)
- **Optional:** for the S3 PutObject event, dynamically extract the name of the file put to the S3 bucket for processing

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.22

22

TUTORIAL 6 – NOV 21

- Introduction to Lambda III: Serverless Databases
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2022_tutorial_6.pdf
- Create and use Sqlite databases using `sqlite3` tool
- Deploy Lambda function with `Sqlite3` database under `/tmp`
- Compare in-memory vs. file-based `Sqlite` DBs on Lambda
- Create an Amazon Aurora "Serverless" v2 MySQL database
- Using an `ec2` instance in the same VPC (Region + availability zone) connect and interact with the database using the `mysql` CLI app
- Deploy an AWS Lambda function that uses the MySQL "serverless" database

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.23

23

OBJECTIVES – 11/10

- Questions from 11/8
- Tutorials Questions
- Class Presentations:
Cloud Technology or Research Paper Review
- Quiz 1
- Ch. 5: Cloud Enabling Technology
- Containerization

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.24

24

GROUP PRESENTATION

- TWO OPTIONS:
- Cloud technology presentation
- Cloud research paper presentation
 - Recent & suggested papers will be posted at:
<http://faculty.washington.edu/wlloyd/courses/tcss562/papers/>

- Submit presentation type and topics (paper or technology) with desired dates of presentation via Canvas by:
TODAY: Wednesday November 16th @ 11:59pm

- Presentation dates:
- Tuesday November 22, Tuesday November 29
- Tuesday December 6, Thursday December 8

November 10, 2022

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.25

25

OBJECTIVES – 11/10

- Questions from 11/8
- Tutorials Questions
- Class Presentations:
Cloud Technology or Research Paper Review
- Quiz 1
- Ch. 5: Cloud Enabling Technology
- Containerization

November 10, 2022

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.26

26

QUIZ 1

- Opens Monday Nov 14 at 8:00 am
- Closes Friday November 18 at 11:59 am
- Individual work only
- Please answer every question
- Book, notes, slides, calculator, and internet are allowed

Grading:

- The Canvas autograder produces a preliminary score, not the final score.
- The instructor will manually review all quizzes and add partial credit
- A curve adjustment will also be applied as appropriate
- These updates may not occur until several days after the quiz closes
- Please report suspected grading problems to the instructor

Attempts:

- 1 quiz attempt, 120 minute limit, 20 questions.
- Coverage is inclusive of Lectures ~1-8
- Please plan accordingly. Once started, there will be 2 hours to complete

November 10, 2022

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.27

27

OBJECTIVES – 11/10

- Questions from 11/8
- Tutorials Questions
- Class Presentations:
Cloud Technology or Research Paper Review
- Quiz 1
- Ch. 5: Cloud Enabling Technology
- Containerization

November 10, 2022

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.28

28

CLOUD ENABLING TECHNOLOGY



29

CLOUD ENABLING TECHNOLOGY

- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology

November 10, 2022

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.30

30

KEY VIRTUALIZATION TRADEOFF

Tradeoff space:

What is the “right” level of abstraction in the cloud for sharing resources with users?

Degree of Hardware Abstraction



Abstraction Concerns:

- Overhead
- Performance
- Isolation
- Security

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.31

31

TYPES OF ABSTRACTION IN THE CLOUD

- **Virtual Machines** – original IaaS cloud abstraction
- **OS and Application Containers** – seen with CaaS
 - **OS Container** – replacement for VM, mimics full OS instance, heavier
 - OS containers run 100s of processes just like a VM
 - **App Container** – Docker: packages dependencies to easily transport and run an application anywhere
 - Application containers run only a few processes
- **Micro VMs** – FaaS / CaaS
 - Lighter weight alternative to full VM (KVM, XEN, VirtualBox)
 - Firecracker
- **Unikernel Operating Systems** – research mostly
 - Single process, multi-thread operating system
 - Designed for cloud, objective to reduce overhead of running too many OS instances

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

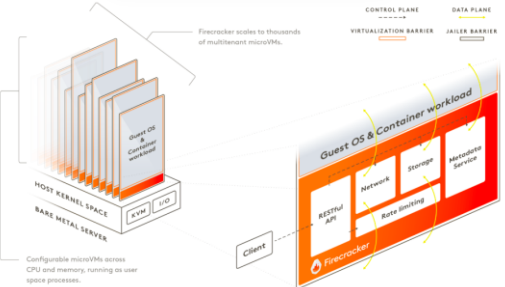
L13.32

32

FIRECRACKER MICRO VM

From <https://firecracker-microvm.github.io/>

The following diagram depicts an example host running Firecracker microVMs.



33

FIRECRACKER MICRO VM

- Provides a virtual machine monitor (VMM) (i.e. hypervisor) using KVM to create and manage microVMs
- Has a minimalist design with goals to improve security, decreases the startup time, and increases hardware utilization
- Excludes unnecessary devices and guest functionality to reduce memory footprint and attack surface area of each microVM
- Supports boot time of <125ms, <5 MiB memory footprint
- Can run 100s of microVMs on a host, launching up to 150/sec
- Is available on 64-bit Intel, AMD, and Arm CPUs
- Used to host AWS Lambda and AWS Fargate
- Has been open sourced under the Apache 2.0 license

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.34

34

FIRECRACKER - 2

- **Minimalistic**
- MicroVMs run as separate processes on the host
- Only 5 emulated devices are available: virtio-net, virtio-block, virtio-vsock, serial console, and a minimal keyboard controller used only to stop the microVM
- Rate limiters can be created and configured to provision resources to support bursts or specific bandwidth/operation limitations
- **Configuration**
- A RESTful API enables common actions such as configuring the number of vCPUs or launching microVMs
- A metadata service between the host and guest provides configuration information

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.35

35

FIRECRACKER - 2

- **Security**
- Runs in user space (**not the root user**) on top of the Linux Kernel-based Virtual Machine (KVM) hypervisor to create microVMs
- Lambda functions, Fargate containers, or container groups can be encapsulated using Firecracker through KVM, enabling workloads from different customers to run on the same machine, without sacrificing security or efficiency
- MicroVMs are further isolated with common Linux user-space security barriers using a companion program called “jailer” which provides a second line of defense if KVM is compromised

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.36

36

TYPES OF ABSTRACTION IN THE CLOUD

- **Virtual Machines** – original IaaS cloud abstraction
- **OS and Application Containers** – seen with CaaS
 - **OS Container** – replacement for VM, mimics full OS instance, heavier
 - OS containers run 100s of processes just like a VM
 - **App Container** – Docker: packages dependencies to easily transport and run an application anywhere
 - Application containers run only a few processes
- **Micro VMs** – FaaS / CaaS
 - Lighter weight alternative to full VM (KVM, XEN, VirtualBox)
 - Firecracker
- **Unikernel Operating Systems** – research mostly
 - Single process, multi-thread operating system
 - Designed for cloud, objective to reduce overhead of running too many OS instances

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

L13.37

37

UNIKERNELS

- Lightweight alternative to containers and VMs
 - Custom Cloud Operating System
 - Single process, multiple threads, runs one program
 - Launch separately atop of hypervisor (XEN/KVM)
 - Reduce overhead, duplication of heavy weight OS
- OSv is most well known unikernel
- Several others exist has research projects
- More information at: <http://unikernel.org/>
- Google Trends OSv →

November 10, 2022

TCSS462/562
School of Eng



38

WE WILL RETURN AT
~7:15 PM



39

VIRTUALIZATION MANAGEMENT

- Virtual infrastructure management (VIM) tools
- Tools that manage pools of virtual machines, resources, etc.
- Private cloud software systems can be considered as a VIM
- Considerations:
 - Performance overhead
 - Paravirtualization: custom OS kernels, I/O passed directly to HW w/ special drivers
 - Hardware compatibility for virtualization
 - Portability: virtual resources tend to be difficult to migrate cross-clouds

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

L13.40

40

VIRTUAL INFRASTRUCTURE MANAGEMENT (VIM)

- Middleware to manage virtual machines and infrastructure of IaaS "clouds"
- Examples
 - OpenNebula
 - Nimbus
 - Eucalyptus
 - OpenStack

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

L13.41

41

VIM FEATURES

- Create/destroy VM instances
- Image repository
 - Create/Destroy/Update images
 - Image persistence
- Contextualization of VMs
 - Networking address assignment
 - DHCP / Static IPs
 - Manage SSH keys

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

L13.42

42

VIM FEATURES - 2

- Virtual network configuration/management
 - Public/Private IP address assignment
 - Virtual firewall management
 - Configure/support isolated VLANs (private clusters)
- Support common virtual machine managers (VMMs)
 - XEN, KVM, VMware
 - Support via libvirt library

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.43

43

VIM FEATURES - 3

- Shared "Elastic" block storage
 - Facility to create/update/delete VM disk volumes
 - Amazon EBS
 - Eucalyptus SC
 - OpenStack Volume Controller

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.44

44

CONTAINER ORCHESTRATION FRAMEWORKS

- Middleware to manage Docker application container deployments across virtual clusters of Docker hosts (VMs)
- Considered Infrastructure-as-a-Service
- Opensource
 - Kubernetes framework
 - Docker swarm
 - Apache Mesos/Marathon
- Proprietary
 - Amazon Elastic Container Service

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.45

45

CONTAINER SERVICES

- Public cloud container cluster services
 - Azure Kubernetes Service (AKS)
 - Amazon Elastic Container Service for Kubernetes (EKS)
 - Google Kubernetes Engine (GKE)
- Container-as-a-Service
 - Azure Container Instances (ACI - April 2018)
 - AWS Fargate (November 2017)
 - Google Kubernetes Engine Serverless Add-on (alpha-July 2018)

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.46

46

CLOUD ENABLING TECHNOLOGY

- Adapted from Ch. 5 from Cloud Computing Concepts, Technology & Architecture
- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.47

47

4. MULTITENANT APPLICATIONS

- Each tenant (like in an apartment) has their own view of the application
- Tenants are unaware of their neighbors
- Tenants can only access their data, no access to data and configuration that is not their own
- Customizable features
 - UI, business process, data model, access control
- Application architecture
 - User isolation, data security, recovery/backup by tenant, scalability for a tenant, for tenants, metered usage, data tier isolation

November 10, 2022

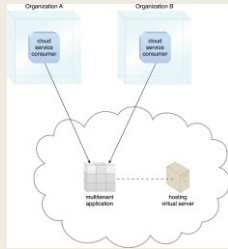
TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.48

48

MULTITENANT APPS - 2

- Forms the basis for SaaS (applications)



49

CLOUD ENABLING TECHNOLOGY

- Adapted from Ch. 5 from *Cloud Computing Concepts, Technology & Architecture*
- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology

50

5. WEB SERVICES/WEB

- Web services technology is a key foundation of cloud computing's "as-a-service" cloud delivery model
- SOAP – "Simple" object access protocol
 - First generation web services
 - WSDL – web services description language
 - UDDI – universal description discovery and integration
 - SOAP services have their own unique interfaces
- REST – instead of defining a custom technical interface REST services are built on the use of HTTP protocol
- HTTP GET, PUT, POST, DELETE

51

HYPERTEXT TRANSPORT PROTOCOL (HTTP)

- An ASCII-based request/reply protocol for transferring information on the web
- HTTP request includes:
 - request method (GET, POST, etc.)
 - Uniform Resource Identifier (URI)
 - HTTP protocol version understood by the client
 - headers—extra info regarding transfer request
- HTTP response from server
 - Protocol version & status code →
 - Response headers
 - Response body

HTTP status codes:	
2xx	— all is well
3xx	— resource moved
4xx	— access problem
5xx	— server error

52

REST: REPRESENTATIONAL STATE TRANSFER

- Web services protocol
- Supersedes SOAP – Simple Object Access Protocol
- Access and manipulate web resources with a predefined set of stateless operations (known as web services)
- Requests are made to a URI
- Responses are most often in JSON, but can also be HTML, ASCII text, XML, no real limits as long as text-based
- HTTP verbs: GET, POST, PUT, DELETE, ...

53

```
// SOAP REQUEST

POST /InStock HTTP/1.1
Host: www.bookshop.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.bookshop.org/prices">
    <m:GetBookPrice>
      <m:BookName>The Fleamarket</m:BookName>
    </m:GetBookPrice>
  </soap:Body>
</soap:Envelope>
```

54

```
// SOAP RESPONSE
POST /InStock HTTP/1.1
Host: www.bookshop.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.bookshop.org/prices">
    <m:GetBookPriceResponse>
      <m:Price>10.95</m:Price>
    </m:GetBookPriceResponse>
  </soap:Body>
</soap:Envelope>
```

November 10, 2022

TCSS462/562 (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.55

55

```
// WSDL Service Definition
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="DayOfWeek"
  targetNamespace="http://www.cogswave.com/soapwsc/examples/DayOfWeek.wsdl"
  xmlns:tns="http://www.cogswave.com/soapwsc/examples/DayOfWeek.wsdl"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:tns1="http://schemas.xmlsoap.org/wsdl/">
  <port name="data" type="tns1:data"/>
  <message name="DayOfWeekInput">
    <part name="data" type="tns1:data"/>
  </message>
  <message name="DayOfWeekResponse">
    <part name="DayOfWeek" type="xsd:string"/>
  </message>
  <portType name="DayOfWeekPortType">
    <operation name="GetDayOfWeek">
      <input message="tns:DayOfWeekInput"/>
      <output message="tns:DayOfWeekResponse"/>
    </operation>
  </portType>
  <binding name="DayOfWeekBinding" type="tns:DayOfWeekPortType">
    <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetDayOfWeek">
      <soap:operation soapAction="getDayOfWeek"/>
      <input>
        <soap:body use="unencoded"
          namespace="http://www.cogswave.com/soapwsc/examples"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
      </input>
      <output>
        <soap:body use="unencoded"
          namespace="http://www.cogswave.com/soapwsc/examples"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
      </output>
    </operation>
  </binding>
  <service name="DayOfWeekService">
    <documentation>
      Returns the day-of-week name for a given date
    </documentation>
    <port name="DayOfWeekPort" binding="tns:DayOfWeekBinding">
      <soap:address location="http://localhost:8090/dayOfWeek/DayOfWeek"/>
    </port>
  </service>
</definitions>
```

November 10, 2022

TCSS462/562 (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.56

56

REST CLIMATE SERVICES EXAMPLE

- **USDA Lat/Long Climate Service Demo**
- Just provide a Lat/Long

```
// REST/JSON
// Request climate data for Washington
{
  "parameter": [
    {
      "name": "latitude",
      "value": 47.2529
    },
    {
      "name": "longitude",
      "value": -122.4443
    }
  ]
}
```

November 10, 2022

TCSS462/562 (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.57

57

REST - 2

- App manipulates one or more types of resources.
- Everything the app does can be characterized as some kind of operation on one or more resources.
- Frequently services are CRUD operations (create/read/update/delete)
 - Create a new resource
 - Read resource(s) matching criterion
 - Update data associated with some resource
 - Destroy a particular a resource
- Resources are often implemented as objects in OO languages

November 10, 2022

TCSS462/562 (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.58

58

REST ARCHITECTURAL ADVANTAGES

- **Performance:** component interactions can be the dominant factor in user-perceived performance and network efficiency
- **Scalability:** to support large numbers of services and interactions among them
- **Simplicity:** of the Uniform Interface
- **Modifiability:** of services to meet changing needs (even while the application is running)
- **Visibility:** of communication between services
- **Portability:** of services by redeployment
- **Reliability:** resists failure at the system level as redundancy of infrastructure is easy to ensure


November 10, 2022

TCSS462/562 (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.59

59

CONTAINERIZATION



November 10, 2022

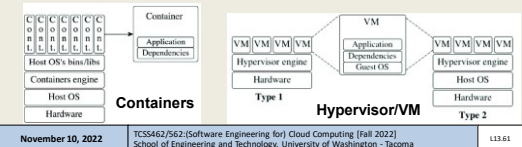
TCSS462/562 (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.60

60

MOTIVATION FOR CONTAINERIZATION

- Containers provide "light-weight" alternative to full OS virtualization provided by a hypervisor
- Containers do not provide a full "machine"
- Instead use operating system constructs to provide "sand boxes" for execution
 - Linux cgroups, namespaces, etc.
- Containers can run on bare metal, or atop of VMs



61

CONTAINER PERFORMANCE - LU FACTORIZATION PERFORMANCE

- Solve linear equations - matrix algebra

Performance data from IC2E 2015: Hypervisors vs. Lightweight Virtualization: A Performance Comparison

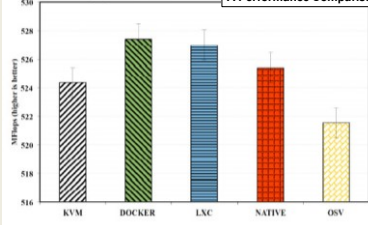
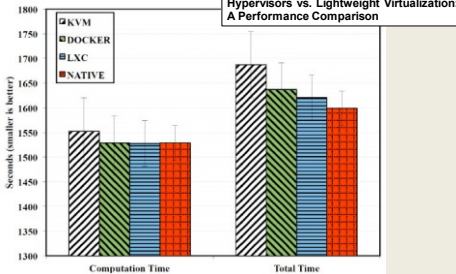


Fig. 4. The value of Linpack results on each platform over 15 runs. This is the particular case of N=1000.

62

CONTAINER PERFORMANCE - Y-CRUNCHER: PI CALCULATOR

Performance data from IC2E 2015: Hypervisors vs. Lightweight Virtualization: A Performance Comparison



63

CONTAINER PERFORMANCE - BONNIE++

Performance data from IC2E 2015: Hypervisors vs. Lightweight Virtualization: A Performance Comparison

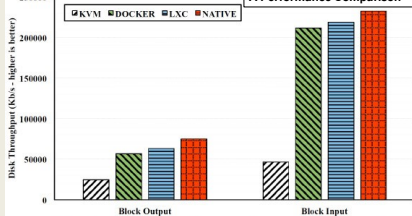


Fig. 6. Disk Throughput achieved by running Bonnie++ (test file of 25 GiB). Results for sequential writes and sequential read are shown.

64

WHAT IS A CONTAINER?

According to NIST (National Institute of Standards Technology)

- Virtualization:** the simulation of the software and/or hardware upon which other software runs. (800-125)
- System Virtual Machine:** A System Virtual Machine (VM) is a software implementation of a complete system platform that supports the execution of a complete operating system and corresponding applications in a cloud. (800-180 draft)
- Operating System Virtualization (aka OS Container):** Provide multiple virtualized OSes above a single shared kernel (800-190). E.g., Solaris Zone, FreeBSD Jails, LXC
- Application Virtualization (aka Application Containers):** Same shared kernel is exposed to multiple discrete instances (800-180 draft). E.g., Docker (containerd), rkt

65

OPERATING SYSTEM CONTAINERS

- Virtual environments: share the host kernel
- Provide user space isolation
- Replacement for VMs: run multiple processes, services
- Mix different Linux distros on same host

- Examples: LXC, OpenVZ, Linux Vserver, BSD Jails, Solaris zones



66

APPLICATION CONTAINERS

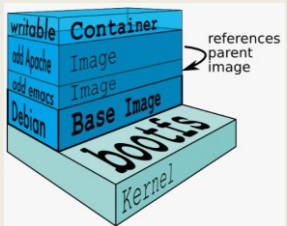
- Designed to package and run a single service
- All containers share host kernel
- Subtle differences from operating system containers
- Examples: Docker, Rocket
- Docker: runs a single process on creation
- OS containers: run many OS services, for an entire OS
- Create application containers for each component of an app
- Supports a micro-services architecture
- DevOPS: developers can package their own components in application containers
- Supports horizontal and vertical scaling

November 10, 2022 TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma L13.67

67

APPLICATION CONTAINERS - 2

- Container images are "layered"
- Base image: common for all components
- Add layers that are specific for components, services as needed
- Layering promotes reuse
- Reduces duplication of data across images

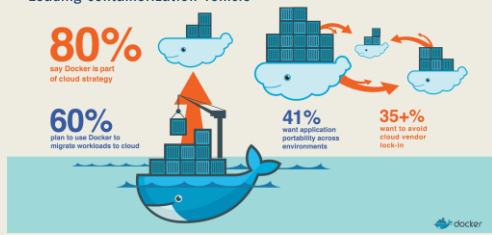


November 10, 2022 TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma L13.68

68

2016 DOCKER SURVEY

- Docker application containers
- Leading containerization vehicle

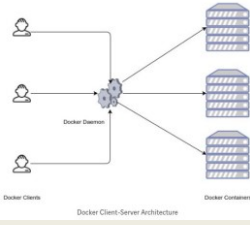


November 10, 2022 TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma L13.69

69

DOCKER

- Docker daemon "dockerd"
- Implements docker engine that interprets CLI requests and creates/manages containers using backend layered Docker architecture
- Starting in 2017 version numbering switches from 1.x to YR.x
- 2017 releases: 17.03 - 17.12
- 2018 releases: 18.01 - 18.09
- 2019 releases: 19.03.0 - 19.03.13

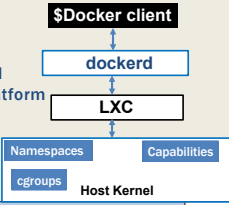


November 10, 2022 TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma L13.70

70

ORIGINAL DOCKER ENGINE IMPLEMENTATION

- (1) Original Docker engine relied on LXC
- LXC itself is a containerization tool predating Docker
- Original Docker API just called it
- LXC originally provided access to Linux kernel features: namespaces and cgroups
- LXC was Linux specific - caused issues if wanting to be multi-platform
- Docker implemented their own replacement for LXC

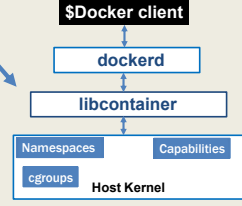


November 10, 2022 TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma L13.71

71

INTRODUCTION OF LIBCONTAINER

- Docker v0.9: **libcontainer** introduced (~2014) to replace LXC as the default Docker daemon



November 10, 2022 TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma L13.72

72

OPEN CONTAINER INITIATIVE (OCI)

- OCI created container standards for:
 - Image specification
 - Container runtime specification
- Docker 1.1 (2016): Docker refactored the docker engine to be compliant with OCI standards
 - Essentially this introduced abstraction layers (i.e. generic interfaces that map to the implementation) so that Docker's design conformed to the OCI standard
- Runc** was added to implement the OCI container runtime spec
 - Provides small, lightweight wrapper for libcontainer
 - Can build and run OCI compliant containers directly using runc provided in Docker, but it is "bare bones" and low-level.
 - The Docker API is much more user friendly
- Support for OCI compliant images was added to **Containerd**

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.73

73

CREATING A CONTAINER

```
$ docker run -it --rm tc5558client sh
```

- Docker CLI posts request to **Docker daemon**
- Daemon calls **containerd**
- Containerd** passes request to **runc**
 - Containerd** converts docker image into OCI compliant bundle
 - This step would allow any OCI compliant container to be plugged into the back-end
- Runc** interfaces with the Linux kernel (namespaces, cgroups, etc.) to create container
- Shim**: once a container is created, runc exits
 - Shim remains as a daemonless stub to implement the container
 - Allows Docker to be upgraded w/o stopping the container !!!

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.74

74

CREATING A CONTAINER - 2

Containerd Integration Architecture

- Docker CLI: interfaces with **dockerd** daemon
- Docker engine: **dockerd** daemon, interfaces with **containerd**
- Containerd**: simple daemon, interfaces with **runc** to manage containers; CRUD interface for containers, images, volumes, networks, builds; HTTP API → Google RPC (gRPC) interface;
- runc**: lightweight command-line tool for running containers; Interfaces with Linux cgroups, namespaces; Runs an OCI container

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.75

75

SUPPORT FOR ALTERNATE CONTAINER RUNTIMES

- Modularity of Docker implementation supports "execution drivers concept":
- Enables docker to support many alternate container backends
- OpenVZ, system-nspawn, libvirt-lxc, libvirt-sandbox, qemu/kvm, BSD Jails, Solaris Zones, and chroot

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.76

76

LINUX KERNEL NAMESPACES

- Partitions kernel resources
- Processes see only their set of resources
- Provides isolation
- Namespaces are hierarchical
- Parent processes can see down the hierarchy
- 7 namespaces in Linux (cgroups not shown)
- Each process can only see resources associated with the namespace, and descendent namespaces

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.77

77

NAMESPACES - 2

- Provides isolation of OS entities for containers
- mnt**: separate filesystems
- pid**: independent PIDs; first process in container is PID 1
- ipc**: prevents processes in different IPC namespaces from being able to establish shared memory. Enables processes in different containers to reuse the same identifiers without conflict. ... provides expected VM like isolation...
- user**: user identification and privilege isolation among separate containers
- net**: network stack virtualization. Multiple loopbacks (lo)
- UTS (UNIX time sharing)**: provides separate host and domain

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.78

78

CONTROL GROUPS (CGROUPS)

- Collection of Linux processes
- Group-level resource allocation: CPU, memory, disk I/O, network I/O
- Resource limiting
 - Memory, disk cache
- Prioritization
 - CPU share
 - Disk I/O throughput
- Accounting
 - Track resource utilization
 - For resource management and/or billing purposes
- Control
 - Pause/resume processes
 - Checkpointing → Checkpoint/Restore in Userspace (CRIU)
 - https://criu.org

November 10, 2022

TCSS462/562 (Software Engineering for) Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

L13.79

79

CGROUPS - 2

- Control groups are hierarchical
- Groups inherit limits from parent groups
- Linux has multiple cgroup controllers (subsystems)
- ls /proc/cgroups
- "memory" controller limits memory use
- "cpuacct" controller accounts for CPU usage
- cgroup filesystem:
 - /sys/fs/cgroup
- Can browse resource utilization of containers...

subsys_name	hierarchy	num_cgroups	enabled
cpuset	3	2	1
cpu	5	97	1
cpuacct	5	97	1
blkio	8	97	1
memory	9	218	1
devices	6	97	1
freezer	4	2	1
net_cls	2	2	1
perf_event	10	2	1
net_prio	2	2	1
hugetlb	7	2	1
pids	11	98	1

November 10, 2022

TCSS462/562 (Software Engineering for) Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

L13.80

80

OVERLAY FILE SYSTEMS

- Docker leverages overlay filesystems
- 1st: AUFS - Advanced multi-layered unification filesystem
- Now: overlay2
- Union mount file system: combine multiple directories into one that appears to contain combined contents
- Idea: Docker uses layered file systems
- Only the top layer is writeable
- Other layers are read-only
- Layers are merged to present the notion of a real file system
- Copy-on-write-implicit sharing
 - Implement duplicate copy
- https://medium.com/@nagarwal/docker-containers-filesystem-demystified-b6ed8112a04a
- https://www.slideshare.net/jpetazzo/scale11x-ixc-talk-1/

November 10, 2022

TCSS462/562 (Software Engineering for) Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

L13.81

81

LAYERED FS: BUILDING A CONTAINER

Dockerfile:

FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py

Python /app/app.py → 99e54dfb1179 0 B

Run make /app → d74508fb6632 1.895 KB

Copy . /app → c22013c84729 194.5 KB

Ubuntu base image → d5a1f33e8a5a 188.1 MB

ubuntu:18.04

Container

Thin R/W layer

Container layer

Image layers (R/O)

November 10, 2022

TCSS462/562 (Software Engineering for) Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

L13.82

82

THREE-TIER ARCHITECTURE

Node.js
Postgres
Nginx

OS containers

- Meant to be used as an OS - run multiple services
- No layered filesystems by default
- Built on cgroups, namespaces, native process resource isolation
- Examples - LXC, OpenVZ, Linux VServer, BSD Jails, Solaris Zones

Node.js
Postgres
Nginx

App containers

- Meant to run for a single service
- Layered filesystems
- Built on top of OS container technologies
- Examples - Docker, Rocket

November 10, 2022

TCSS462/562 (Software Engineering for) Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

L13.83

83

CONTAINER ISOLATION

- Is the host isolated from application containers?
- Are application containers isolated from each other?

Application containers

App App

Bin/ldso Bin/ldso

Container runtime

VM kernel

Host kernel

Application containers

App App

Bin/ldso Bin/ldso

Container runtime

Host kernel

November 10, 2022

TCSS462/562 (Software Engineering for) Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

L13.84

84

LXC (LINUX CONTAINERS)

- Operating system level virtualization
- Run multiple isolated Linux systems on a host using a single Linux kernel
- Control groups(cgroups)
 - Including in Linux kernels => 2.6.24
 - Limit and prioritize sharing of CPU, memory, block/network I/O
- Linux namespaces
- Docker initially based on LXC

November 10, 2022

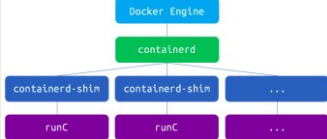
TCSS462/562: Software Engineering for Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

L13.85

85

OTHER DOCKER TOOLS

- **Docker Machine:** automatically provision and manage sets of docker hosts to form a cluster



- **Docker Swarm:** Clusters multiple docker hosts together to manage as a cluster.
- **Docker Compose:** Config file (YAML) for multi-container application; Describes how to deploy and configure multiple containers

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

L13.86

86

CONTAINER ORCHESTRATION FRAMEWORKS

- Framework(s) to deploy multiple containers
- Provide container clusters using cloud VMs
- Similar to "private clusters"
- Reduce VM idle CPU time in public clouds
- Better leverage "sunk cost" resources
- Compact multiple apps onto shared public cloud infrastructure
- Generate to cost savings
- Reduce vendor lock-in

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

L13.87

87

KEY ORCHESTRATION FEATURES

- Management of container hosts
- Launching set of containers
- Rescheduling failed containers
- Linking containers to support workflows
- Providing connectivity to clients outside the container cluster
- Firewall: control network/port accessibility
- Dynamic scaling of containers: horizontal scaling
 - Scale in/out, add/remove containers
- Load balancing over groups of containers
- Rolling upgrades of containers for application

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

L13.88

88

CONTAINER ORCHESTRATION FRAMEWORKS - 2

- Docker swarm
- Apache mesos/marathon
- Kubernetes
 - Many public cloud provides moving to offer Kubernetes-as-a-service
- Amazon elastic container service (ECS)
- Apache aurora
- Container-as-a-Service
 - Serverless containers without managing clusters
 - Azure Container Instances, AWS Fargate...

November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

L13.89

89

TUTORIAL #7 DOCKER, CGROUPS, RESOURCE ISOLATION



November 10, 2022

TCSS462/562: Software Engineering for Cloud Computing (Fall 2022)
School of Engineering and Technology, University of Washington - Tacoma

Fall 2022
L13.90

90

TUTORIAL COVERAGE

- Docker CLI → Docker Enginer (dockerd) → containerd → runc
- Concepts:
- Docker installation
- Working with docker files
- Docker run – create a container
- Docker ps – list containers
- Docker exec –it – run a process in an existing container
- Docker stop –stop container

November 10, 2022

TCSS462/562 (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.91

91

attach
build
commit
cp
create
deploy
diff
events
exec
export
history
images
import
info
inspect
kill
load
login
logout
logs
pause
port
ps
pull
push
rename
restart
rm
rmi
run
save
search
start
stats
stop
tag
top
unpause
update
version
wait

Attach local standard input, output, and error streams to a running container
Build an image from a Dockerfile
Create a new image from a container's changes
Copy files/folders between a container and the local filesystem
Create a new container
Deploy a new stack or update an existing stack
Inspect changes to files or directories on a container's filesystem
Get real time events from the server
Run a command in a running container
Export a container's filesystem as a tar archive
Show the history of an image
List images
Import the contents from a tarball to create a filesystem image
Display system-wide information
Return low-level information on Docker objects
Kill one or more running containers
Load an image from a tar archive or STDIN
Log in to a Docker registry
Log out from a Docker registry
Fetch the logs of a container
Pause all processes within one or more containers
List port mappings or a specific mapping for the container
List containers
Pull an image or a repository from a registry
Push an image or a repository to a registry
Rename a container
Restart one or more containers
Remove one or more containers
Remove one or more images
Run a command in a new container
Save one or more images to a tar archive (streamed to STDOUT by default)
Search the Docker Hub for images
Start one or more stopped containers
Display a live stream of container(s) resource usage statistics
Stop one or more running containers
Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
Display the running processes of a container
Unpause all processes within one or more containers
Update configuration of one or more containers
Show the Docker version information
Block until one or more containers stop, then print their exit codes

Docker CLI

92

TUTORIAL 7

- Linux performance benchmarks
- stress-ng
- 100s of CPU, memory, disk, network stress tests
- Sysbench
- Used in tutorial for memory stress test


November 10, 2022

TCSS462/562 (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.93

93

QUESTIONS



November 10, 2022

TCSS462/562 (Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L13.94

94