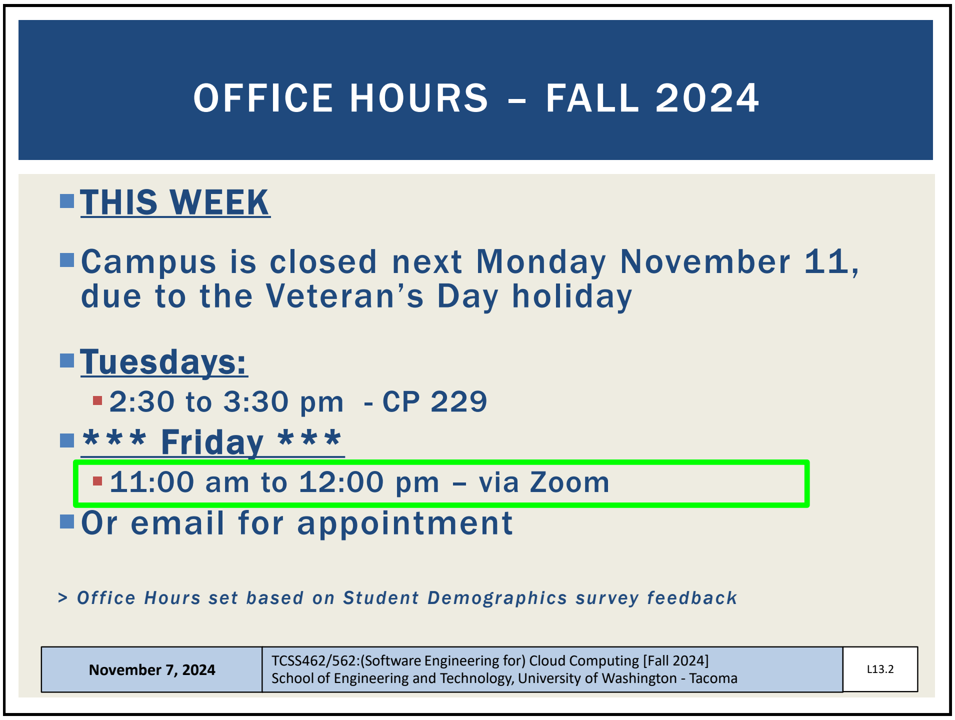
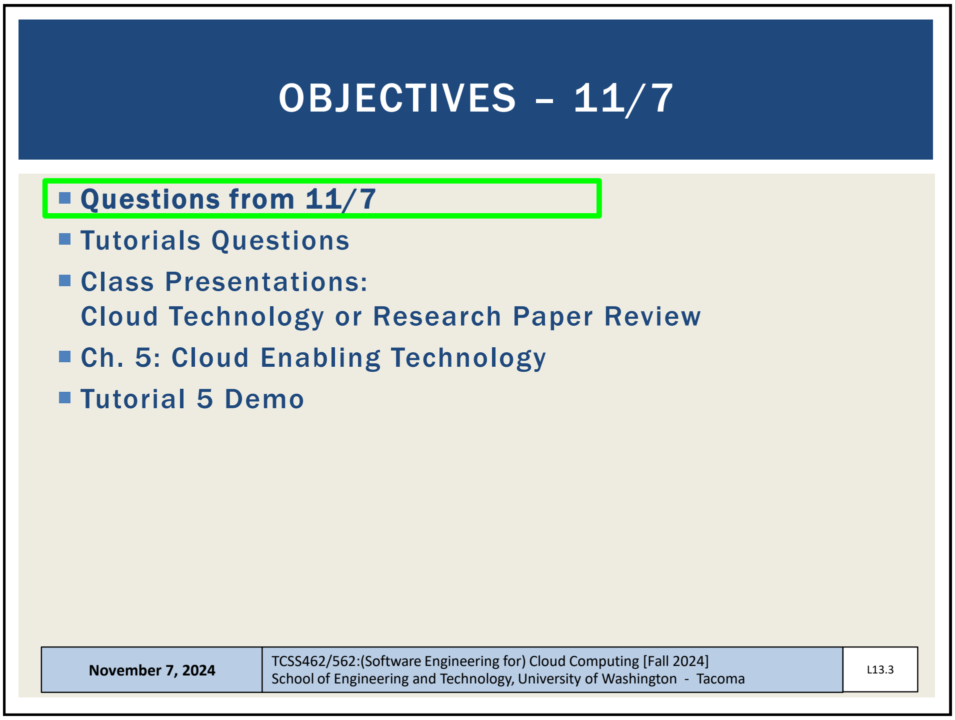


OFFICE HOURS – FALL 2024

- **THIS WEEK**
- Campus is closed next Monday November 11, due to the Veteran's Day holiday
- **Tuesdays:**
  - 2:30 to 3:30 pm - CP 229
- ***** Friday *****
  - 11:00 am to 12:00 pm – via Zoom
- Or email for appointment

> Office Hours set based on Student Demographics survey feedback

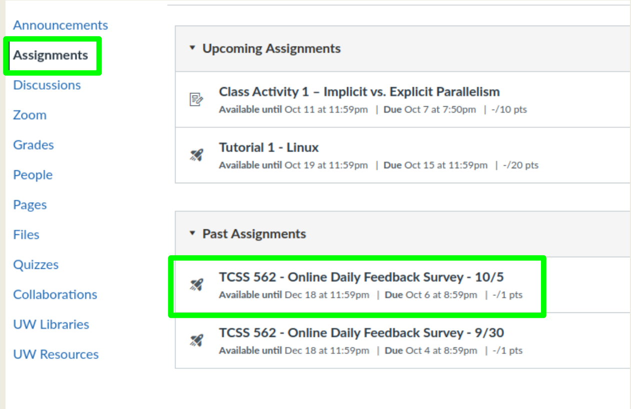


OBJECTIVES – 11/7

- Questions from 11/7
- Tutorials Questions
- Class Presentations:
  Cloud Technology or Research Paper Review
- Ch. 5: Cloud Enabling Technology
- Tutorial 5 Demo

ONLINE DAILY FEEDBACK SURVEY

- Daily Feedback Quiz in Canvas – Take After Each Class
- Extra Credit for completing



▾ Upcoming Assignments

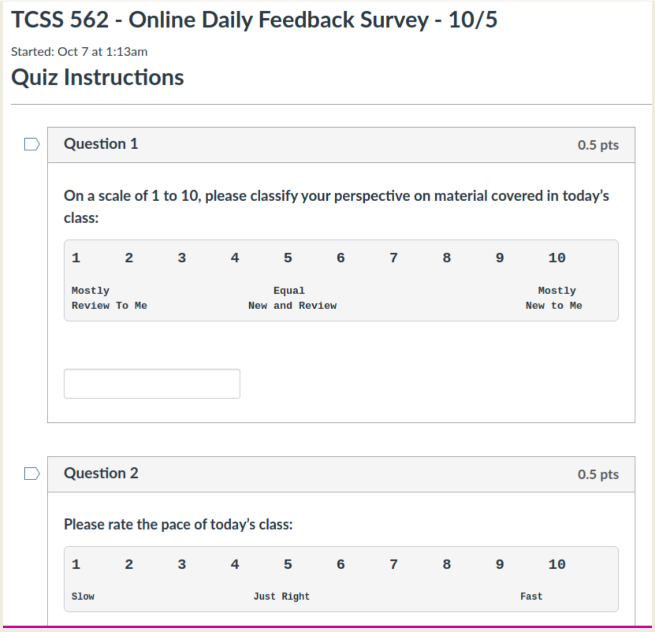
Class Activity 1 – Implicit vs. Explicit Parallelism
Available until Oct 11 at 11:59pm | Due Oct 7 at 7:50pm | -/10 pts

Tutorial 1 - Linux
Available until Oct 19 at 11:59pm | Due Oct 15 at 11:59pm | -/20 pts

▾ Past Assignments

TCSS 562 - Online Daily Feedback Survey - 10/5
Available until Dec 18 at 11:59pm | Due Oct 6 at 8:59pm | -/1 pts

TCSS 562 - Online Daily Feedback Survey - 9/30
Available until Dec 18 at 11:59pm | Due Oct 4 at 8:59pm | -/1 pts



TCSS 562 - Online Daily Feedback Survey - 10/5

Started: Oct 7 at 1:13am

Quiz Instructions

Question 1 0.5 pts

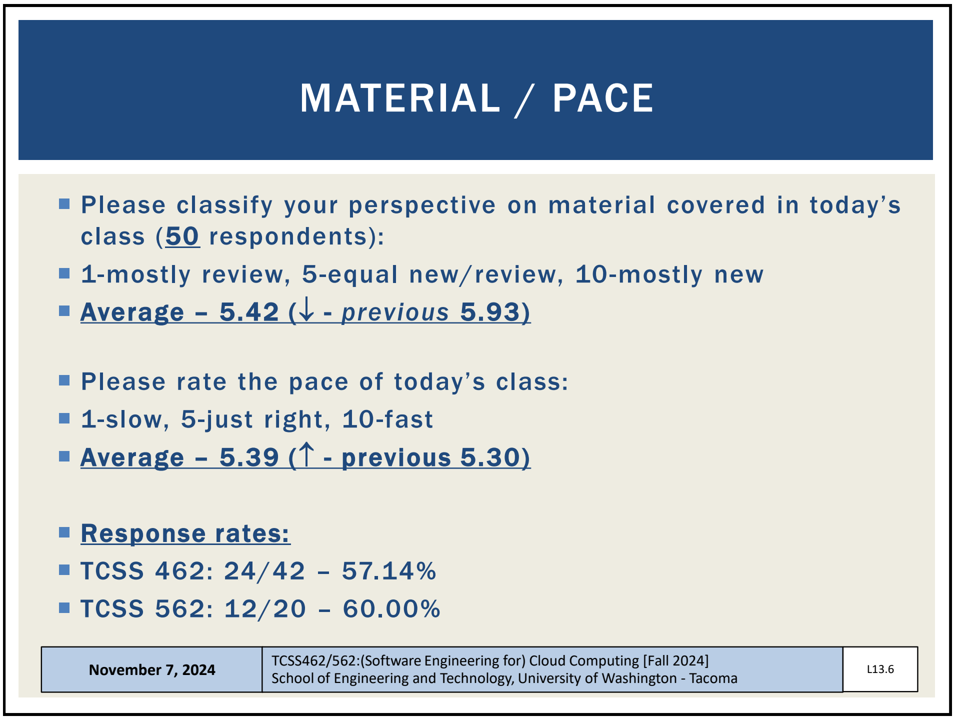
On a scale of 1 to 10, please classify your perspective on material covered in today's class:

```
1     2     3     4     5     6     7     8     9     10

Mostly                Equal                      Mostly
Review To Me          New and Review             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
```



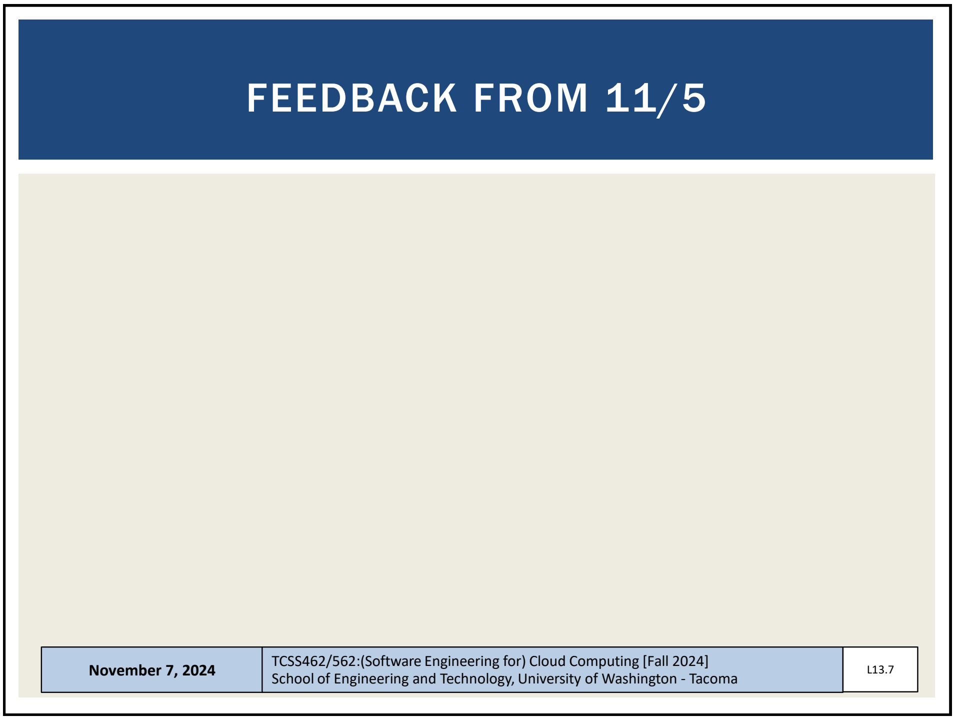
MATERIAL / PACE

- Please classify your perspective on material covered in today's class (**50** respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 5.42 (↓ - *previous* 5.93)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.39 (↑ - previous 5.30)**

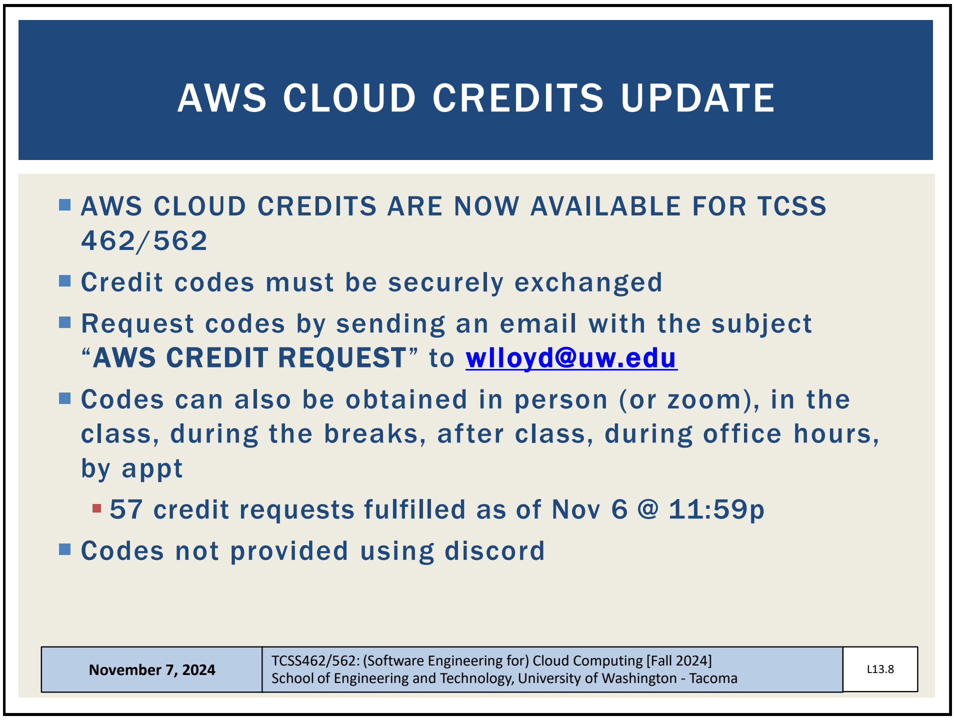
- **Response rates:**
- TCSS 462: 24/42 – 57.14%
- TCSS 562: 12/20 – 60.00%

6



FEEDBACK FROM 11/5

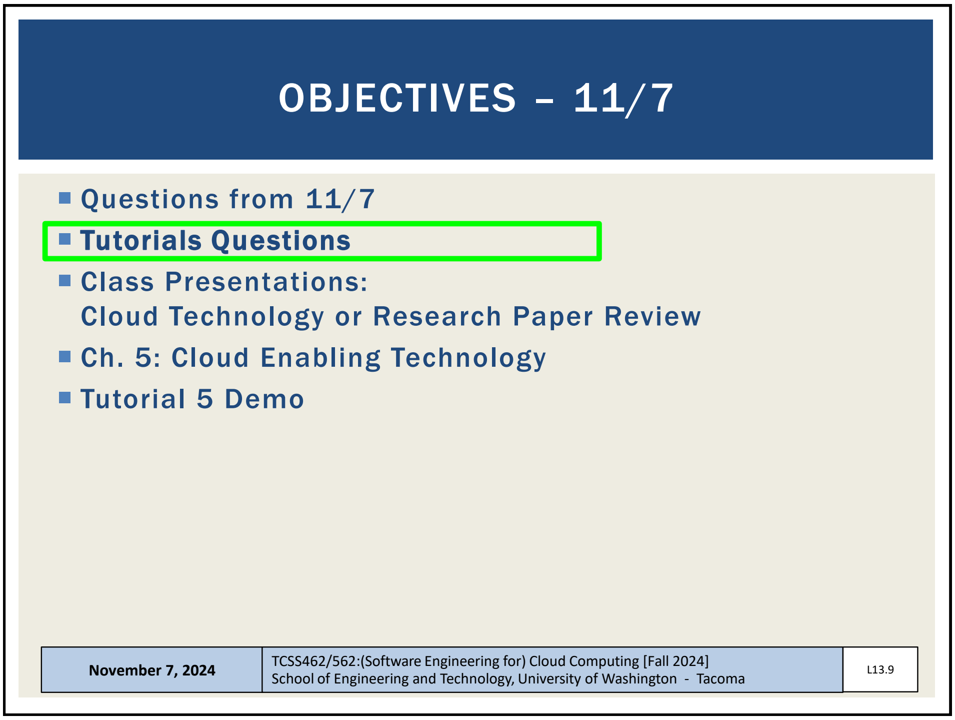
7



AWS CLOUD CREDITS UPDATE

- AWS CLOUD CREDITS ARE NOW AVAILABLE FOR TCSS 462/562
- Credit codes must be securely exchanged
- Request codes by sending an email with the subject "**AWS CREDIT REQUEST**" to wlloyd@uw.edu
- Codes can also be obtained in person (or zoom), in the class, during the breaks, after class, during office hours, by appt
  - 57 credit requests fulfilled as of Nov 6 @ 11:59p
- Codes not provided using discord

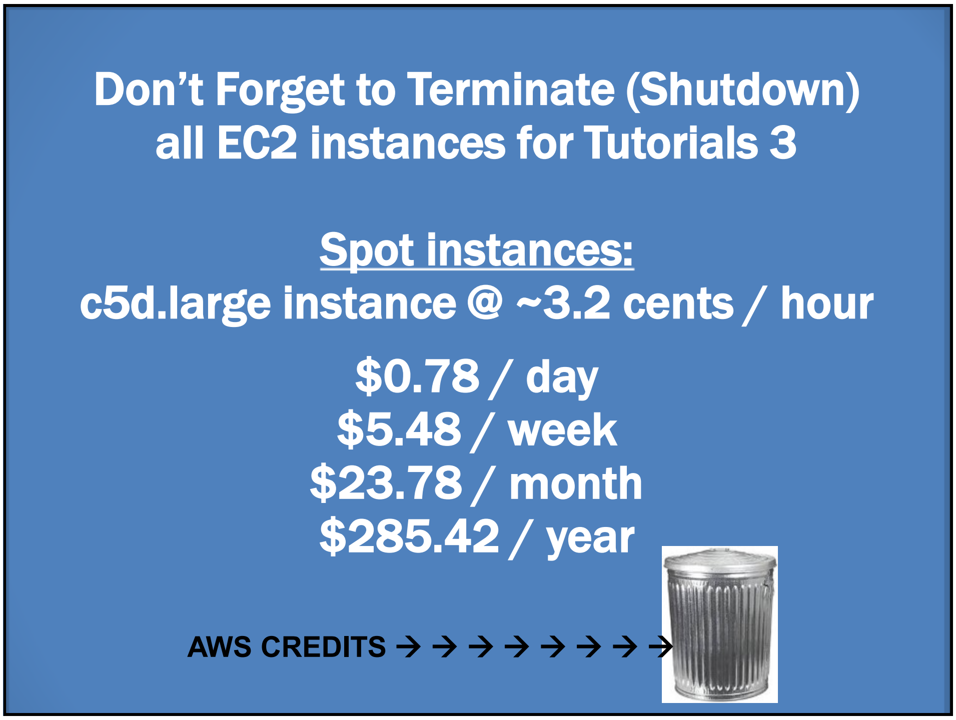
8



OBJECTIVES – 11/7

- Questions from 11/7
- **Tutorials Questions**
- Class Presentations:
  Cloud Technology or Research Paper Review
- Ch. 5: Cloud Enabling Technology
- Tutorial 5 Demo

9



Don’t Forget to Terminate (Shutdown) all EC2 instances for Tutorials 3

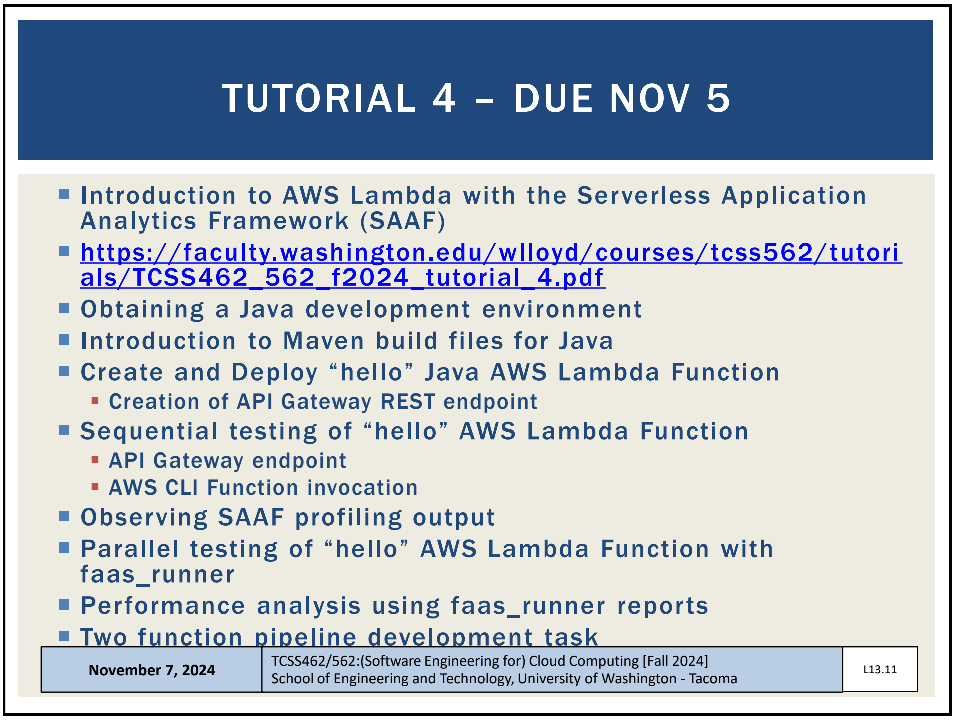
Spot instances:

c5d.large instance @ ~3.2 cents / hour

$0.78 / day
$5.48 / week
$23.78 / month
$285.42 / year

AWS CREDITS → → → → → → → →

10

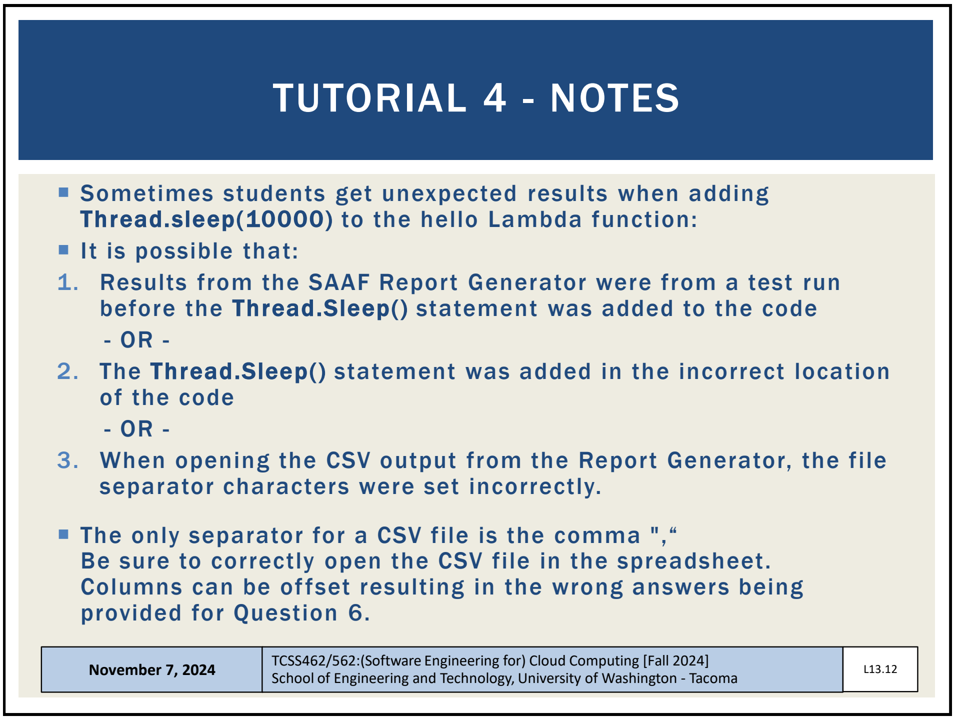


TUTORIAL 4 – DUE NOV 5

- Introduction to AWS Lambda with the Serverless Application Analytics Framework (SAAF)
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2024_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 7, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L13.11

11



TUTORIAL 4 - NOTES

- Sometimes students get unexpected results when adding **Thread.sleep(10000)** to the hello Lambda function:
- It is possible that:

1. Results from the SAAF Report Generator were from a test run before the **Thread.Sleep()** statement was added to the code

   - OR -

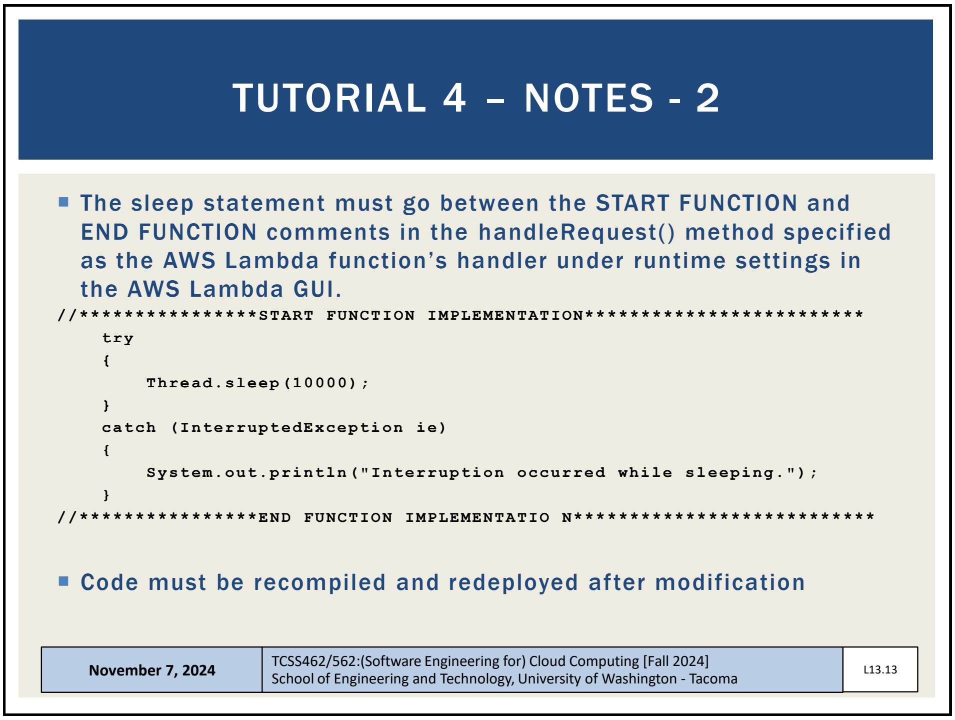
2. The **Thread.Sleep()** statement was added in the incorrect location of the code

   - OR -

3. 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.

12



TUTORIAL 4 – NOTES - 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 IMPLEMENTATIO N***************************
```

- Code must be recompiled and redeployed after modification

13

TUTORIAL 4 – NOTES - 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**

TUTORIAL 4 – NOTES - 4

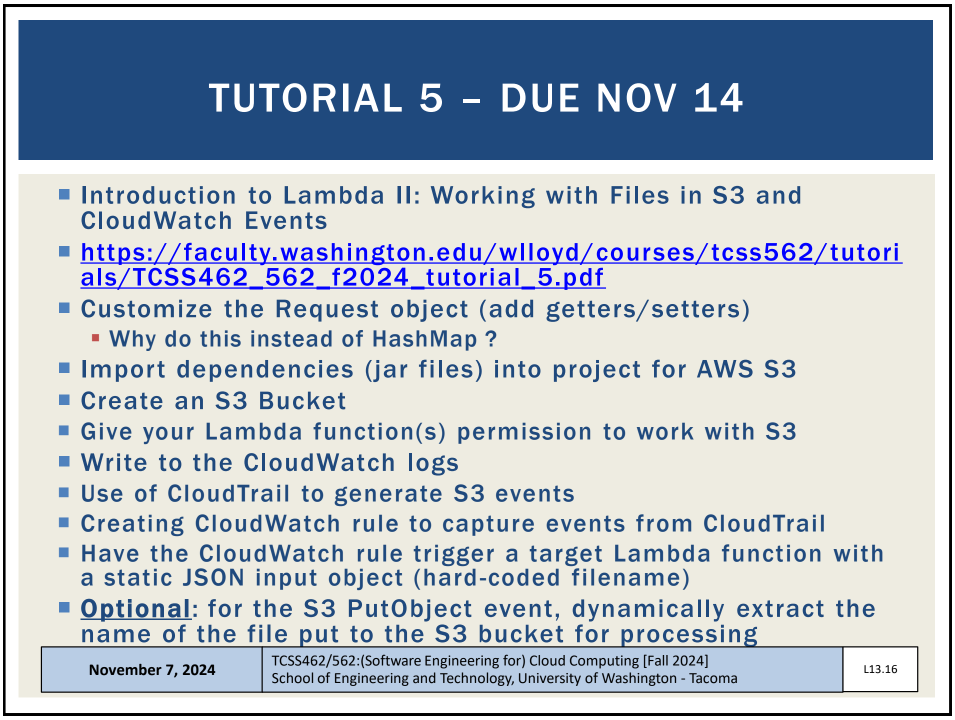
- **avg_cpuIdleDelta** time is the amount of time the Lambda function's Firecracker two vCPUs are idle during the function call on the server measured in centiseconds:

100 centiseconds = 1 second

100 centiseconds = 1000 milliseconds

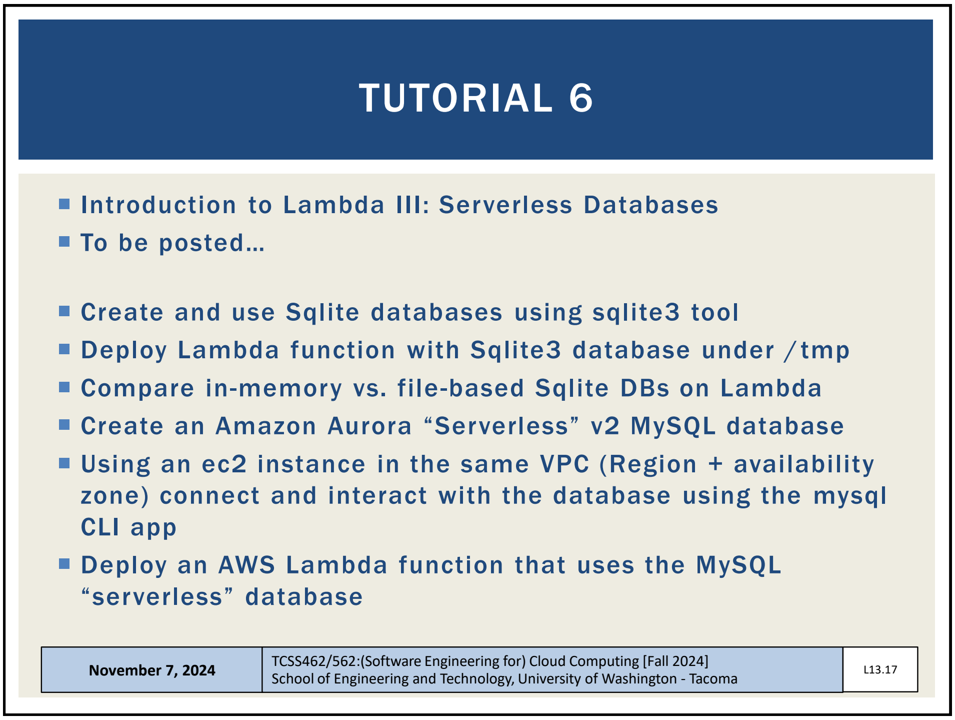
- By default, AWS Lambda functions with 512 MB run in a runtime environment with access to two vCPU cores
- This is the total vCPU idle time for both cores (it is doubled)
- Adding sleep of 10 seconds should increase your function's avg_cpuIdleDelta

- **How much should avg_cpuIdleDelta increase ?**



TUTORIAL 5 – DUE NOV 14

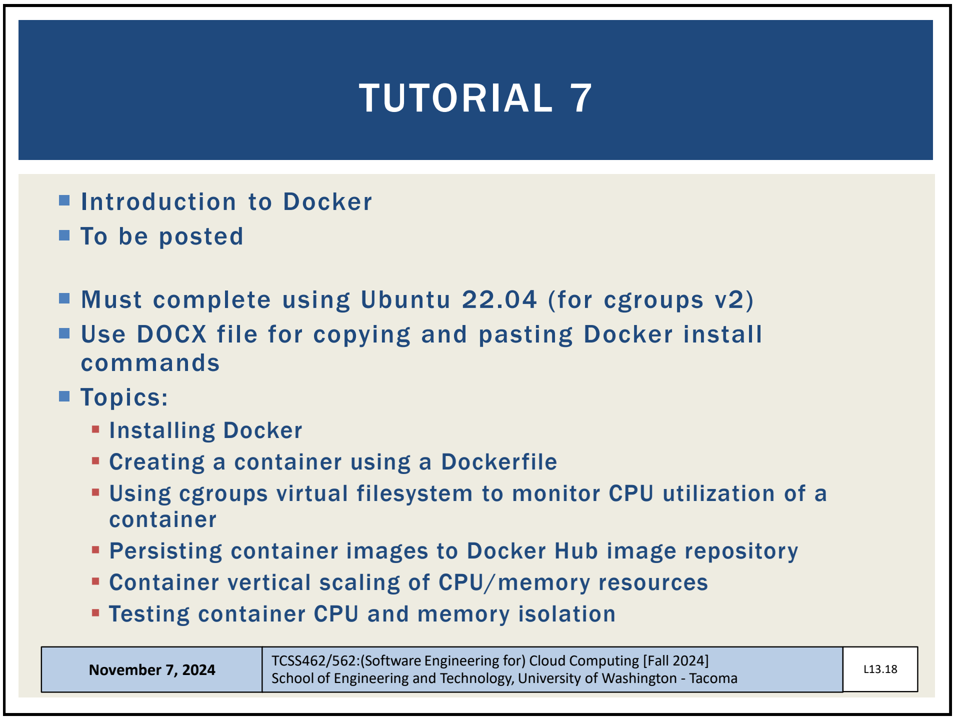
- Introduction to Lambda II: Working with Files in S3 and CloudWatch Events
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2024_tutorial_5.pdf
- Customize the Request object (add getters/setters)
  - Why do this instead of HashMap ?
- Import dependencies (jar files) into 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



TUTORIAL 6

- Introduction to Lambda III: Serverless Databases
- To be posted…

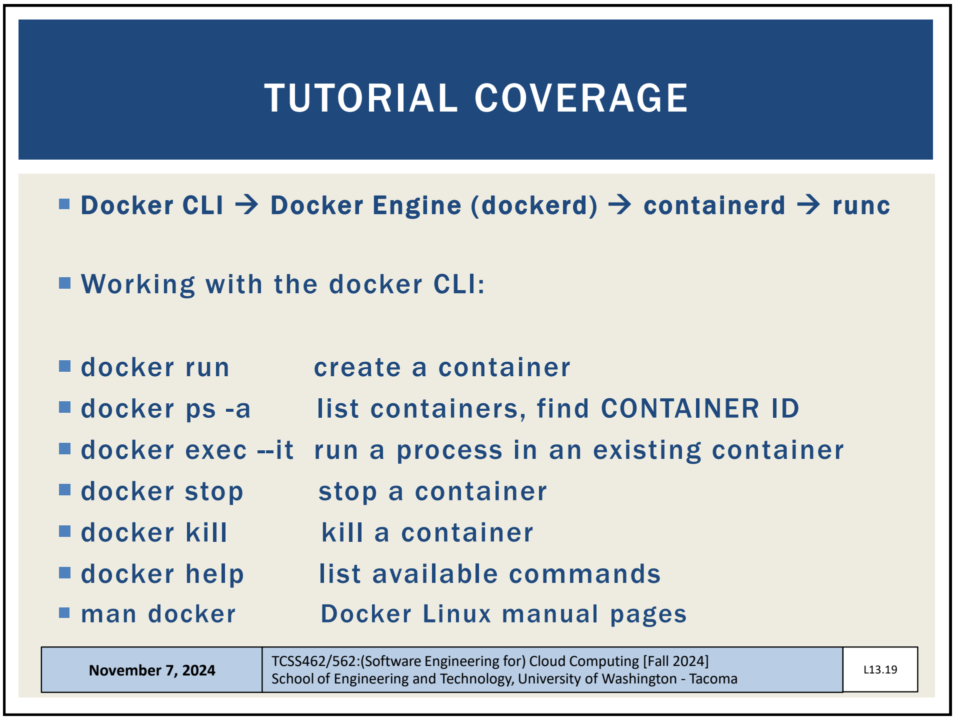
- Create and use Sqlite databases using sqlite3 tool
- Deploy Lambda function with Sqlite3 database under /tmp
- Compare in-memory vs. file-based Sqlite DBs on Lambda
- Create an Amazon Aurora “Serverless” v2 MySQL database
- Using an ec2 instance in the same VPC (Region + availability zone) connect and interact with the database using the mysql CLI app
- Deploy an AWS Lambda function that uses the MySQL “serverless” database



TUTORIAL 7

- Introduction to Docker
- To be posted

- Must complete using Ubuntu 22.04 (for cgroups v2)
- Use DOCX file for copying and pasting Docker install commands
- Topics:
  - Installing Docker
  - Creating a container using a Dockerfile
  - Using cgroups virtual filesystem to monitor CPU utilization of a container
  - Persisting container images to Docker Hub image repository
  - Container vertical scaling of CPU/memory resources
  - Testing container CPU and memory isolation

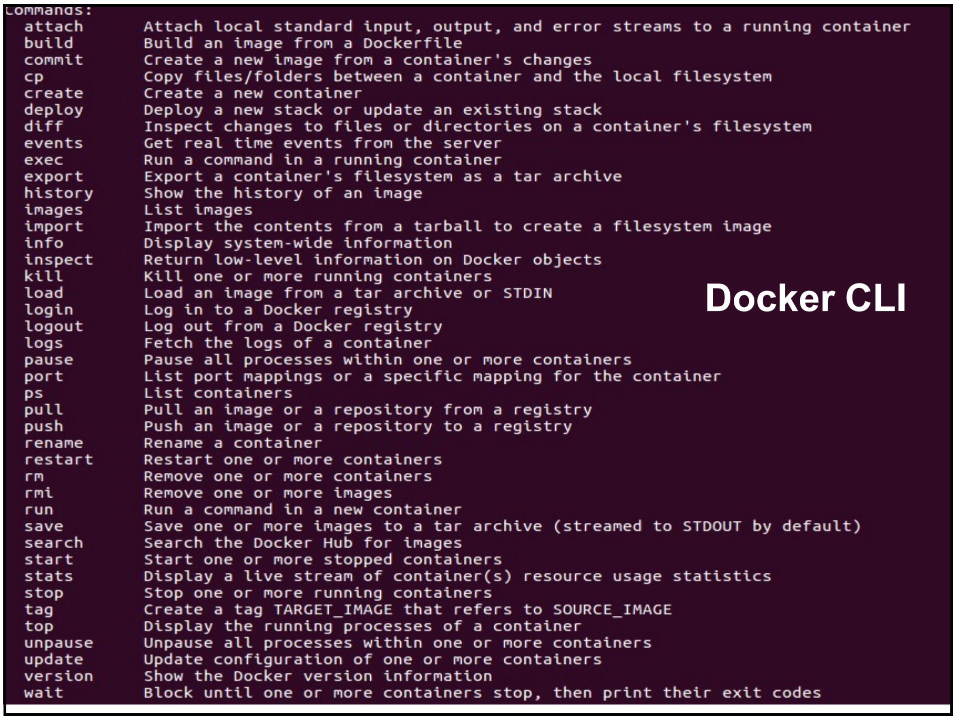


TUTORIAL COVERAGE

- **Docker CLI → Docker Engine (dockerd) → containerd → runc**

- Working with the docker CLI:

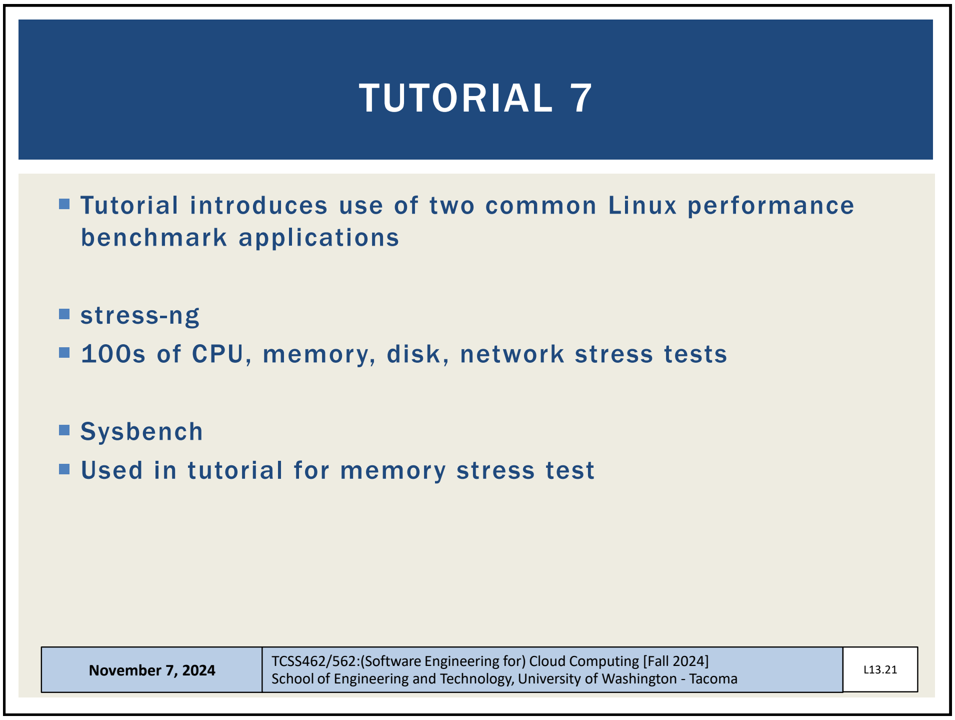
- docker run   create a container
- docker ps -a   list containers, find CONTAINER ID
- docker exec --it run a process in an existing container
- docker stop   stop a container
- docker kill   kill a container
- docker help   list available commands
- man docker   Docker Linux manual pages



```
Commands:
  attach      Attach local standard input, output, and error streams to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  deploy      Deploy a new stack or update an existing stack
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  run         Run a command in a new container
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  search      Search the Docker Hub for images
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  version     Show the Docker version information
  wait        Block until one or more containers stop, then print their exit codes
```

Docker CLI

20



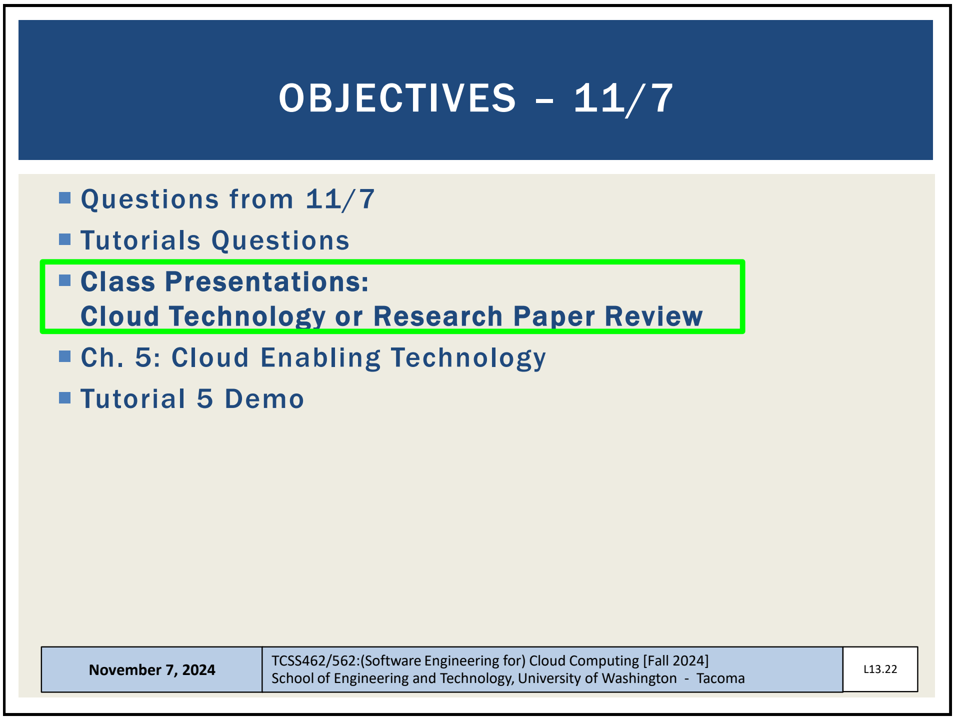
TUTORIAL 7

- Tutorial introduces use of two common Linux performance benchmark applications

- stress-ng
- 100s of CPU, memory, disk, network stress tests

- Sysbench
- Used in tutorial for memory stress test

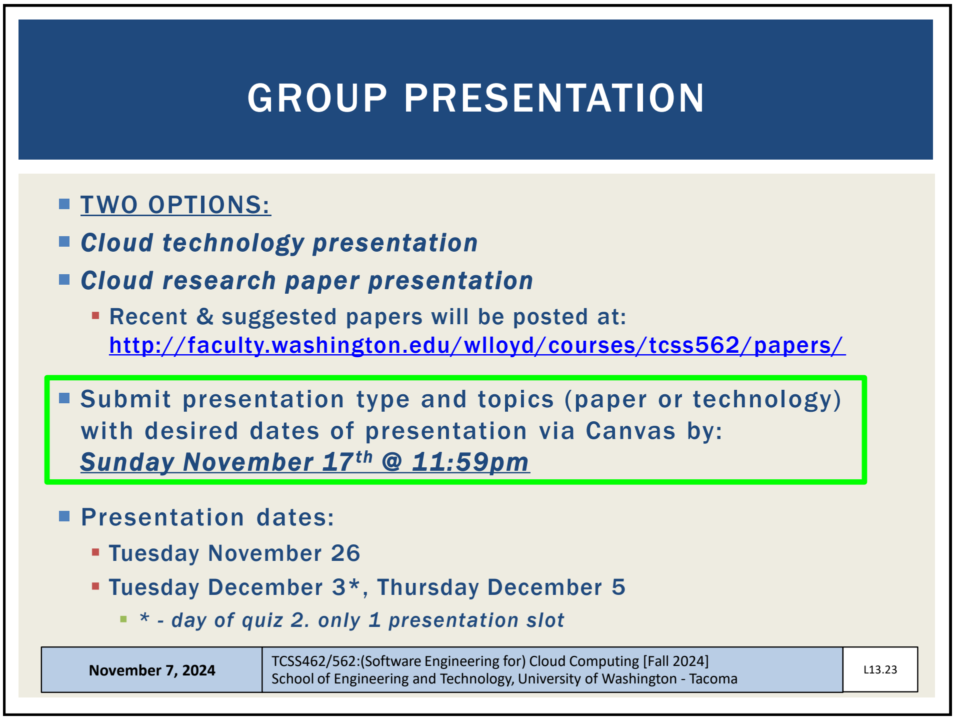
21



OBJECTIVES – 11/7

- Questions from 11/7
- Tutorials Questions
- **Class Presentations:
  Cloud Technology or Research Paper Review**
- Ch. 5: Cloud Enabling Technology
- Tutorial 5 Demo

22



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: ***Sunday November 17th @ 11:59pm***
- Presentation dates:
  - Tuesday November 26
  - Tuesday December 3*, Thursday December 5
    - ** - day of quiz 2. only 1 presentation slot*

23

AWS OVERVIEW AND DEMO

24

INSPECTING INSTANCE INFORMATION

- EC2 VMs run a local metadata service
- Can query instance metadata to self discover cloud config attributes
- **Version 2 (default) of the metadata service requires a token**
- Get Token:

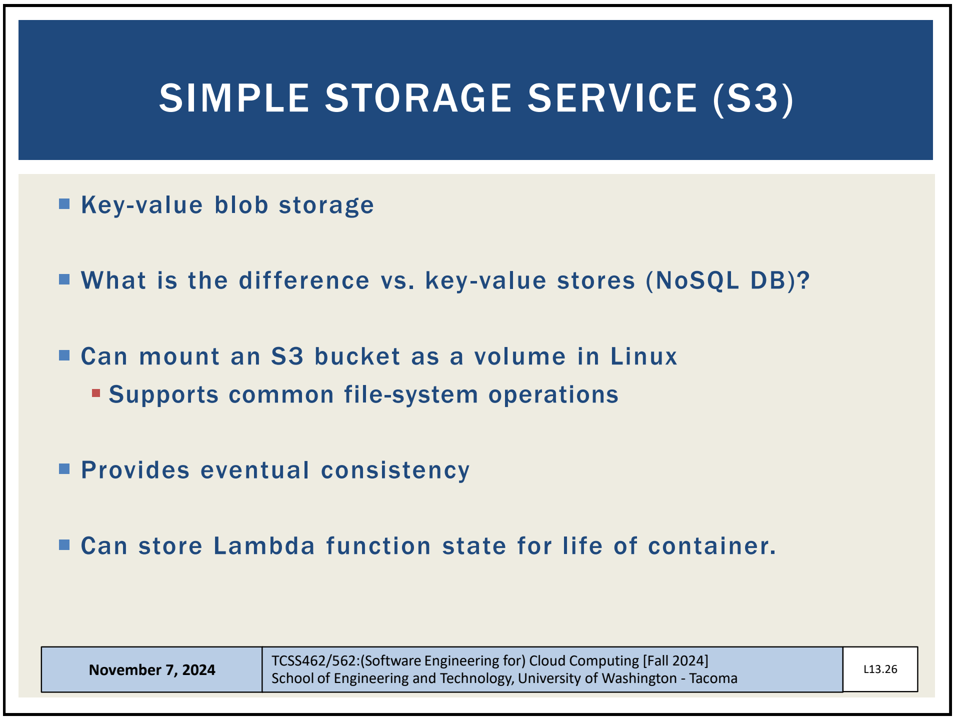
```
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api /token" -H
"X-aws-ec2-metadata-token-ttl-seconds: 21600"`
```

- Find your instance ID:

```
curl -H "X-aws-ec2-metadata-token: $TOKEN“ http://169.254.169.254/

curl -H "X-aws-ec2-metadata-token: $TOKEN“
http://169.254.169.254/latest/

curl -H "X-aws-ec2-metadata-token: $TOKEN“
http://169.254.169.254/latest/meta-data/

curl -H "X-aws-ec2-metadata-token: $TOKEN“
http://169.254.169.254/latest/meta-data/instance-id ; echo
```

See: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/configuring-instance-metadata-service.html#instance-metadata-retrieval-examples

25

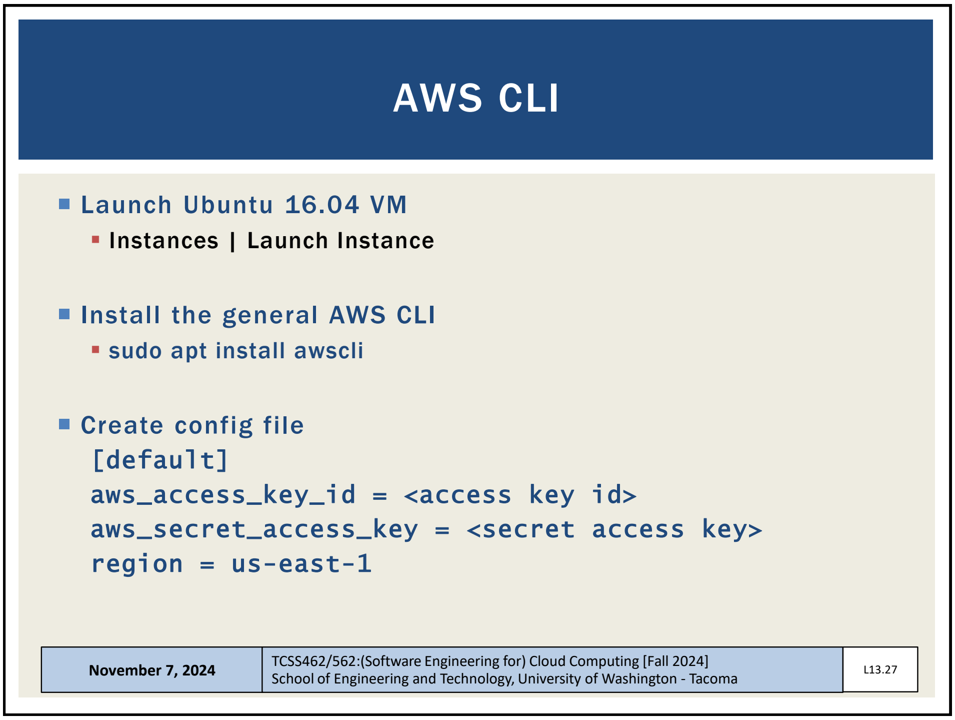


SIMPLE STORAGE SERVICE (S3)

- Key-value blob storage
- What is the difference vs. key-value stores (NoSQL DB)?
- Can mount an S3 bucket as a volume in Linux
  - Supports common file-system operations
- Provides eventual consistency
- Can store Lambda function state for life of container.

November 7, 2024 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]
School of Engineering and Technology, University of Washington - Tacoma | L13.26

26



AWS CLI

- Launch Ubuntu 16.04 VM
  - **Instances | Launch Instance**
- Install the general AWS CLI
  - sudo apt install awscli
- Create config file

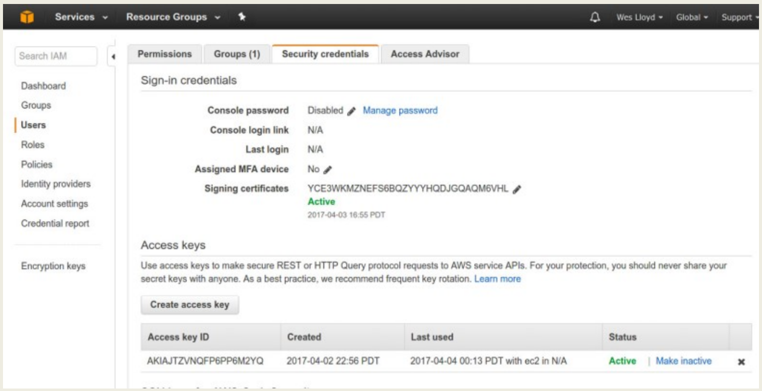
```
[default]
aws_access_key_id = <access key id>
aws_secret_access_key = <secret access key>
region = us-east-1
```

November 7, 2024 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]
School of Engineering and Technology, University of Washington - Tacoma | L13.27

27

AWS CLI - 2

- **Creating access keys:** IAM | Users | Security Credentials | Access Keys | Create Access Keys



Sign-in credentials

Console password	Disabled Manage password
Console login link	N/A
Last login	N/A
Assigned MFA device	No
Signing certificates	YCE3WKMZNEFS6BQZYYYHQDJGQAQM6VHL Active 2017-04-03 16:55 PDT

Access keys

Use access keys to make secure REST or HTTP Query protocol requests to AWS service APIs. For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation. Learn more

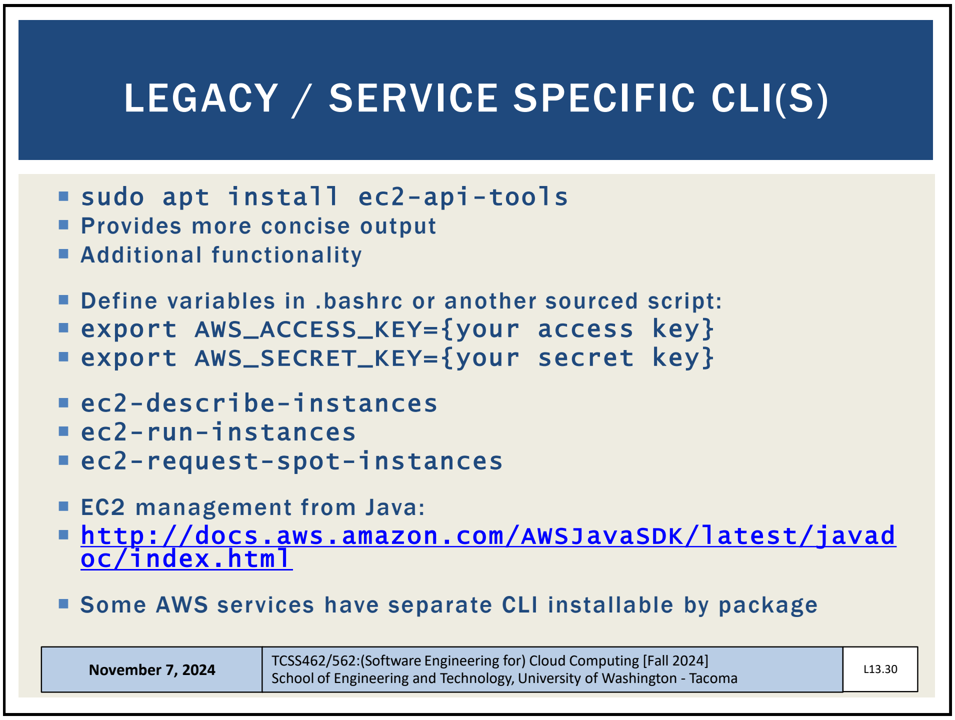
Access key ID	Created	Last used	Status	
AKIAJTZVNQFP6PP6M2YQ	2017-04-02 22:56 PDT	2017-04-04 00:13 PDT with ec2 in N/A	Active | Make inactive	✖

AWS CLI - 3

- Export the config file
  - Add to /home/ubuntu/.bashrc

```
export AWS_CONFIG_FILE=$HOME/.aws/config
```

- Try some commands:
  - `aws help`
  - `aws command help`
  - `aws ec2 help`
  - `aws ec2 describes-instances --output text`
  - `aws ec2 describe-instances --output json`
  - `aws s3 ls`
  - `aws s3 ls vmscaleruw`



LEGACY / SERVICE SPECIFIC CLI(S)

- sudo apt install ec2-api-tools
- Provides more concise output
- Additional functionality

- Define variables in .bashrc or another sourced script:
- export AWS_ACCESS_KEY={your access key}
- export AWS_SECRET_KEY={your secret key}

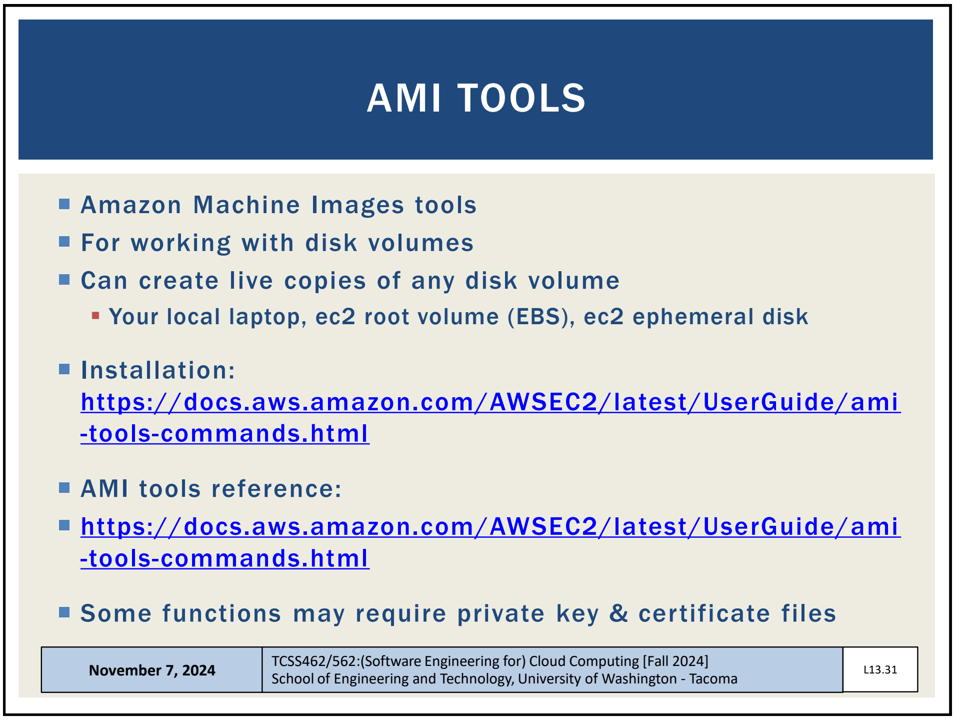
- ec2-describe-instances
- ec2-run-instances
- ec2-request-spot-instances

- EC2 management from Java:
- http://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html

- Some AWS services have separate CLI installable by package

November 7, 2024 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma | L13.30

30



AMI TOOLS

- Amazon Machine Images tools
- For working with disk volumes
- Can create live copies of any disk volume
  - Your local laptop, ec2 root volume (EBS), ec2 ephemeral disk

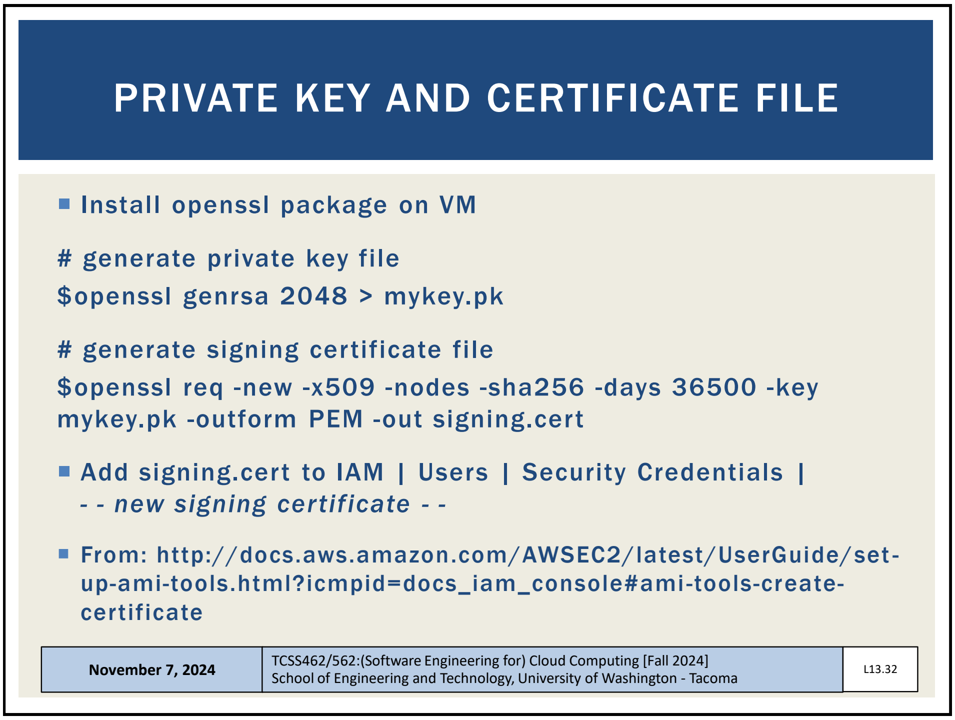
- Installation: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ami-tools-commands.html

- AMI tools reference:
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ami-tools-commands.html

- Some functions may require private key & certificate files

November 7, 2024 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma | L13.31

31

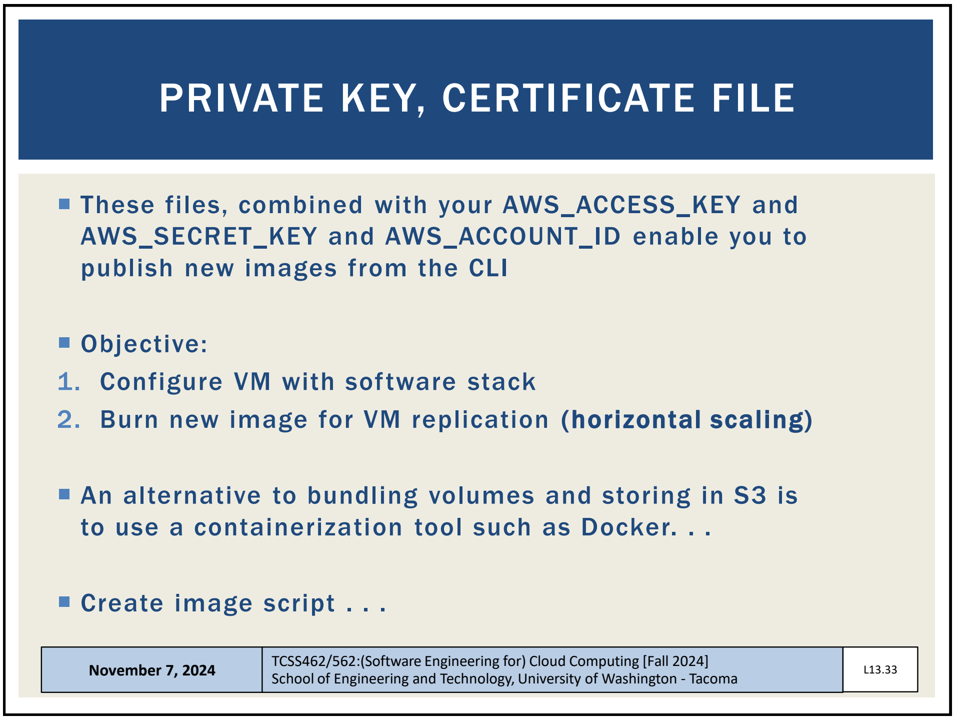


PRIVATE KEY AND CERTIFICATE FILE

- Install openssl package on VM

```
# generate private key file
$openssl genrsa 2048 > mykey.pk

# generate signing certificate file
$openssl req -new -x509 -nodes -sha256 -days 36500 -key
mykey.pk -outform PEM -out signing.cert
```

- Add signing.cert to IAM | Users | Security Credentials | - - *new signing certificate* - -
- From: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/set-up-ami-tools.html?icmpid=docs_iam_console#ami-tools-create-certificate



PRIVATE KEY, CERTIFICATE FILE

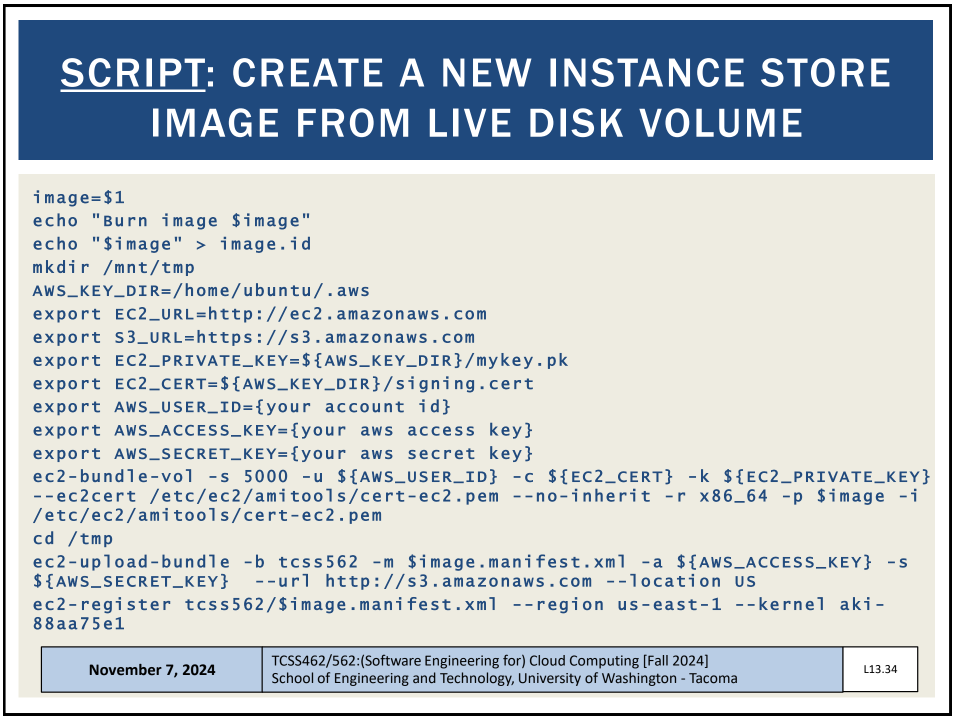
- These files, combined with your AWS_ACCESS_KEY and AWS_SECRET_KEY and AWS_ACCOUNT_ID enable you to publish new images from the CLI

- Objective:

1. Configure VM with software stack
2. Burn new image for VM replication (**horizontal scaling)**

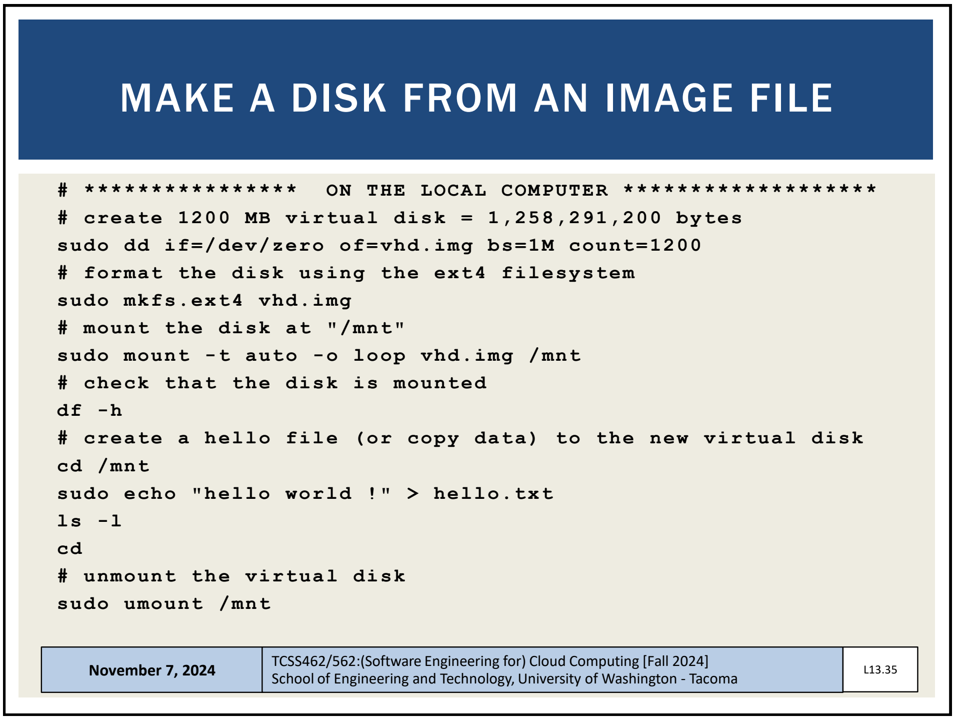
- An alternative to bundling volumes and storing in S3 is to use a containerization tool such as Docker. . .

- Create image script . . .



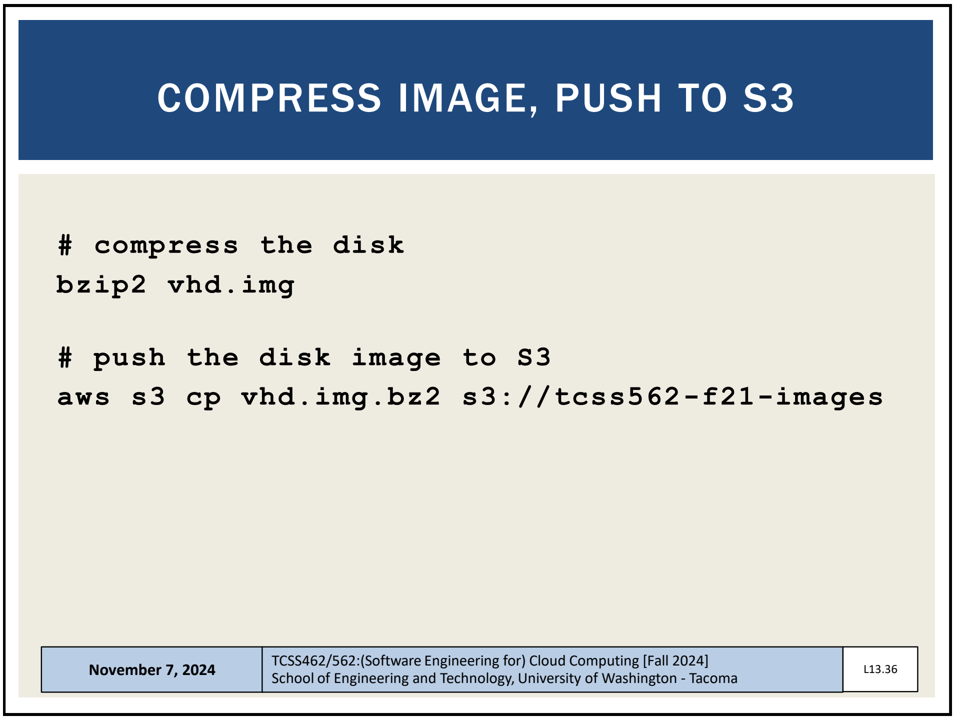
SCRIPT: CREATE A NEW INSTANCE STORE IMAGE FROM LIVE DISK VOLUME

```
image=$1
echo "Burn image $image"
echo "$image" > image.id
mkdir /mnt/tmp
AWS_KEY_DIR=/home/ubuntu/.aws
export EC2_URL=http://ec2.amazonaws.com
export S3_URL=https://s3.amazonaws.com
export EC2_PRIVATE_KEY=${AWS_KEY_DIR}/mykey.pk
export EC2_CERT=${AWS_KEY_DIR}/signing.cert
export AWS_USER_ID={your account id}
export AWS_ACCESS_KEY={your aws access key}
export AWS_SECRET_KEY={your aws secret key}
ec2-bundle-vol -s 5000 -u ${AWS_USER_ID} -c ${EC2_CERT} -k ${EC2_PRIVATE_KEY}
--ec2cert /etc/ec2/amitools/cert-ec2.pem --no-inherit -r x86_64 -p $image -i
/etc/ec2/amitools/cert-ec2.pem
cd /tmp
ec2-upload-bundle -b tcss562 -m $image.manifest.xml -a ${AWS_ACCESS_KEY} -s
${AWS_SECRET_KEY}  --url http://s3.amazonaws.com --location US
ec2-register tcss562/$image.manifest.xml --region us-east-1 --kernel aki-
88aa75e1
```



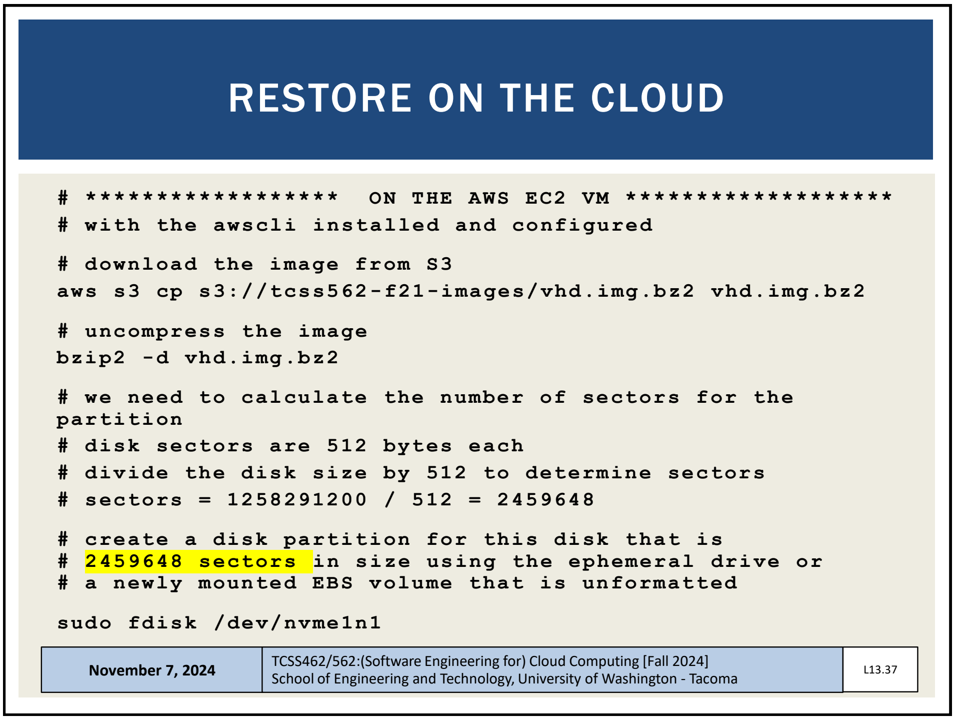
MAKE A DISK FROM AN IMAGE FILE

```
# ****************  ON THE LOCAL COMPUTER *******************
# create 1200 MB virtual disk = 1,258,291,200 bytes
sudo dd if=/dev/zero of=vhd.img bs=1M count=1200
# format the disk using the ext4 filesystem
sudo mkfs.ext4 vhd.img
# mount the disk at "/mnt"
sudo mount -t auto -o loop vhd.img /mnt
# check that the disk is mounted
df -h
# create a hello file (or copy data) to the new virtual disk
cd /mnt
sudo echo "hello world !" > hello.txt
ls -l
cd
# unmount the virtual disk
sudo umount /mnt
```



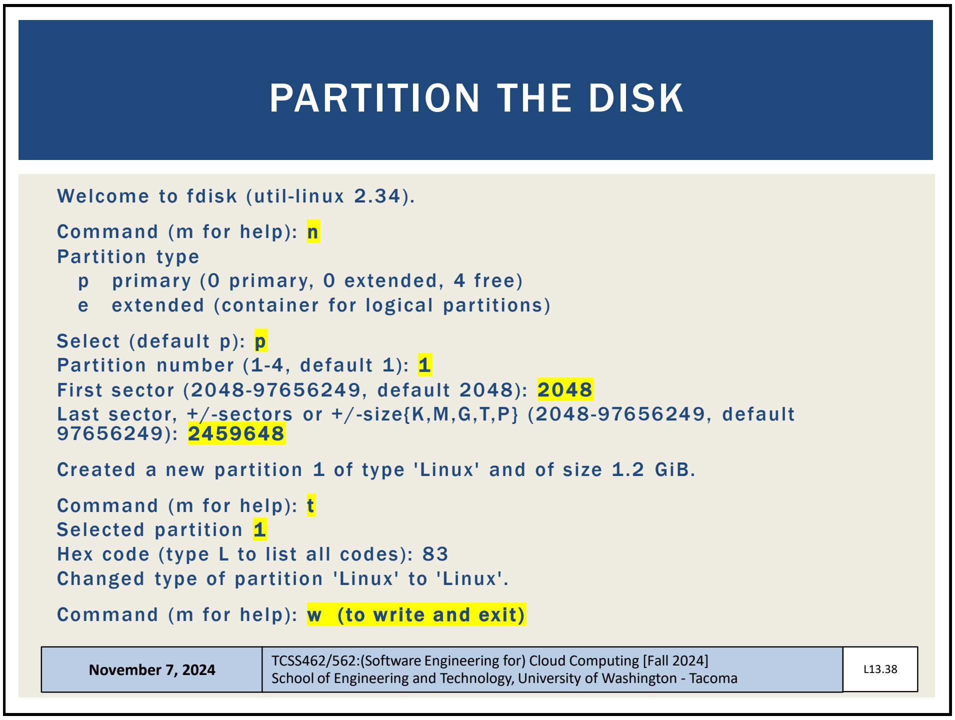
COMPRESS IMAGE, PUSH TO S3

```
# compress the disk
bzip2 vhd.img

# push the disk image to S3
aws s3 cp vhd.img.bz2 s3://tcss562-f21-images
```



RESTORE ON THE CLOUD

```
# *****************  ON THE AWS EC2 VM ******************
# with the awscli installed and configured

# download the image from S3
aws s3 cp s3://tcss562-f21-images/vhd.img.bz2 vhd.img.bz2

# uncompress the image
bzip2 -d vhd.img.bz2

# we need to calculate the number of sectors for the
partition
# disk sectors are 512 bytes each
# divide the disk size by 512 to determine sectors
# sectors = 1258291200 / 512 = 2459648

# create a disk partition for this disk that is
# 2459648 sectors in size using the ephemeral drive or
# a newly mounted EBS volume that is unformatted

sudo fdisk /dev/nvme1n1
```

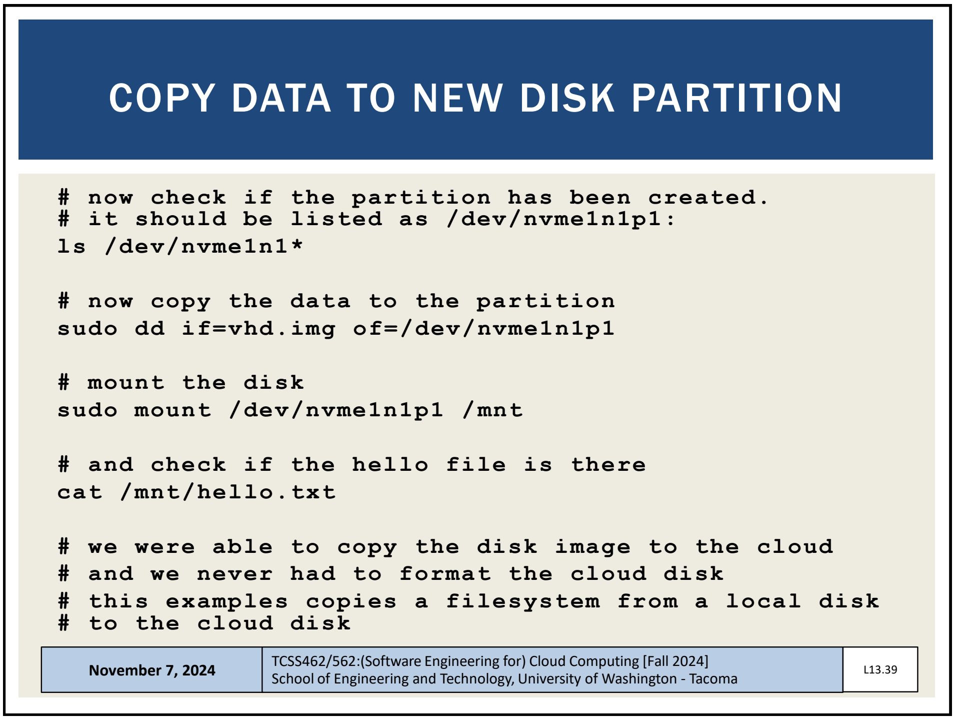


PARTITION THE DISK

```
Welcome to fdisk (util-linux 2.34).

Command (m for help): n
Partition type
  p   primary (0 primary, 0 extended, 4 free)
  e   extended (container for logical partitions)

Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-97656249, default 2048): 2048
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-97656249, default
97656249): 2459648

Created a new partition 1 of type 'Linux' and of size 1.2 GiB.

Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): 83
Changed type of partition 'Linux' to 'Linux'.

Command (m for help): w  (to write and exit)
```

November 7, 2024 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]
School of Engineering and Technology, University of Washington - Tacoma | L13.38

38

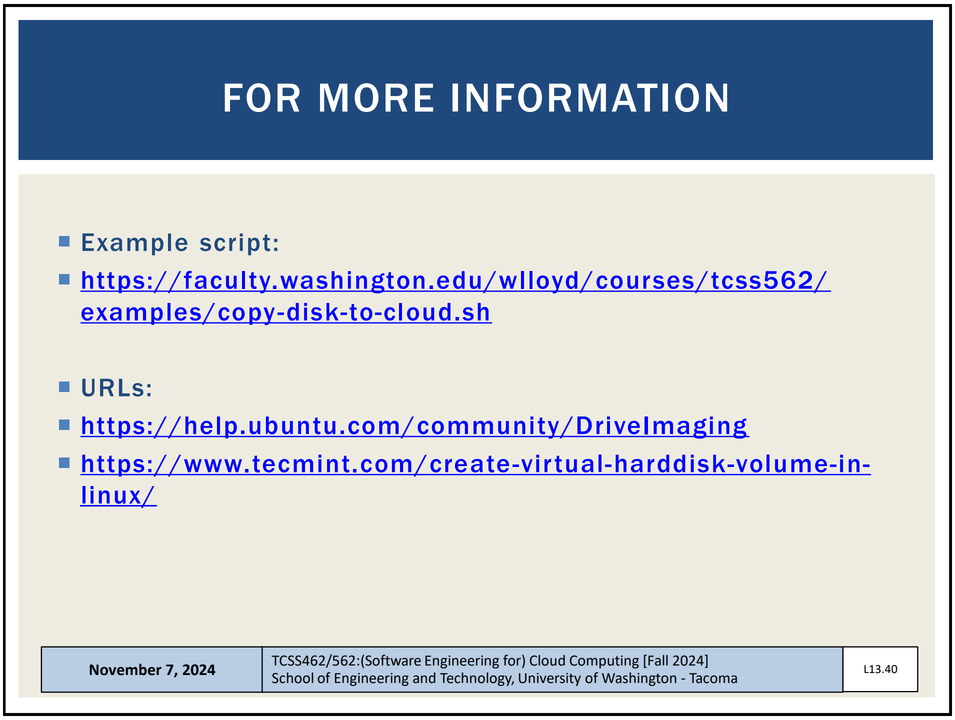


COPY DATA TO NEW DISK PARTITION

```
# now check if the partition has been created.
# it should be listed as /dev/nvme1n1p1:
ls /dev/nvme1n1*

# now copy the data to the partition
sudo dd if=vhd.img of=/dev/nvme1n1p1

# mount the disk
sudo mount /dev/nvme1n1p1 /mnt

# and check if the hello file is there
cat /mnt/hello.txt

# we were able to copy the disk image to the cloud
# and we never had to format the cloud disk
# this examples copies a filesystem from a local disk
# to the cloud disk
```

November 7, 2024 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]
School of Engineering and Technology, University of Washington - Tacoma | L13.39

39



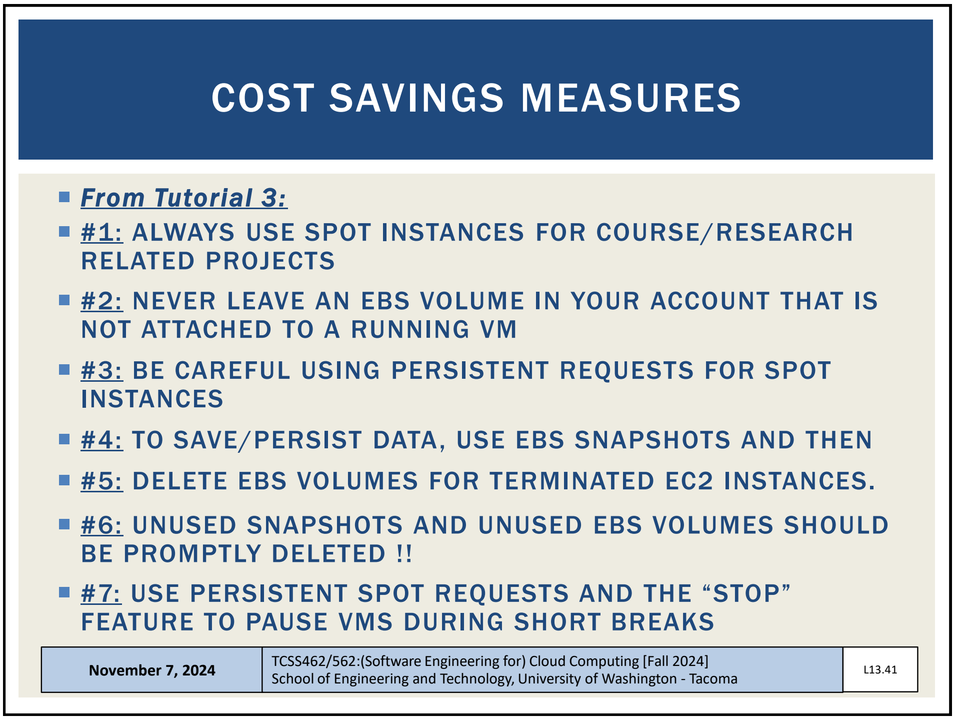
FOR MORE INFORMATION

- Example script:
- https://faculty.washington.edu/wlloyd/courses/tcss562/examples/copy-disk-to-cloud.sh

- URLs:
- https://help.ubuntu.com/community/DriveImaging
- https://www.tecmint.com/create-virtual-harddisk-volume-in-linux/

November 7, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L13.40

40

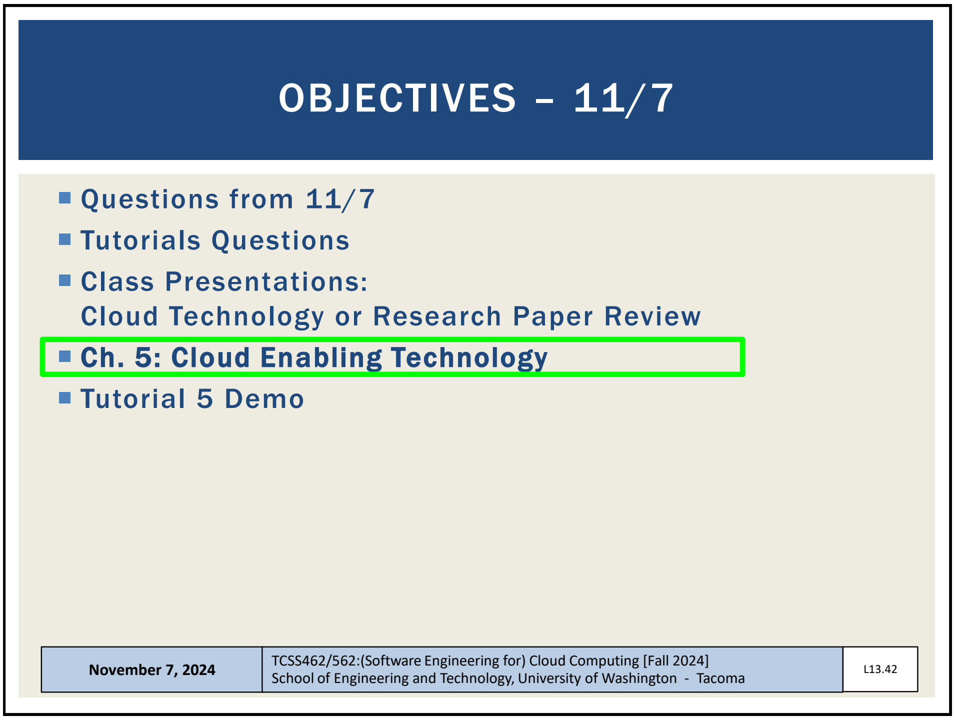


COST SAVINGS MEASURES

- ***From Tutorial 3:***
- #1: ALWAYS USE SPOT INSTANCES FOR COURSE/RESEARCH RELATED PROJECTS
- #2: NEVER LEAVE AN EBS VOLUME IN YOUR ACCOUNT THAT IS NOT ATTACHED TO A RUNNING VM
- #3: BE CAREFUL USING PERSISTENT REQUESTS FOR SPOT INSTANCES
- #4: TO SAVE/PERSIST DATA, USE EBS SNAPSHOTS AND THEN
- #5: DELETE EBS VOLUMES FOR TERMINATED EC2 INSTANCES.
- #6: UNUSED SNAPSHOTS AND UNUSED EBS VOLUMES SHOULD BE PROMPTLY DELETED !!
- #7: USE PERSISTENT SPOT REQUESTS AND THE “STOP” FEATURE TO PAUSE VMS DURING SHORT BREAKS

November 7, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L13.41

41



OBJECTIVES – 11/7

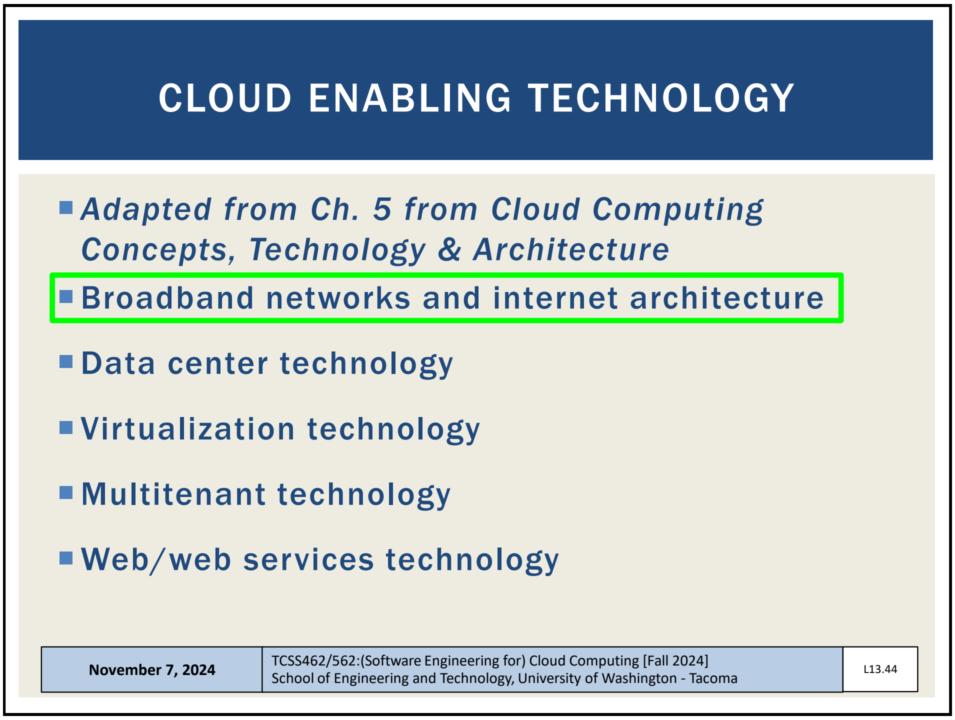
- Questions from 11/7
- Tutorials Questions
- Class Presentations:
  Cloud Technology or Research Paper Review
- **Ch. 5: Cloud Enabling Technology**
- Tutorial 5 Demo

November 7, 2024 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]
School of Engineering and Technology, University of Washington - Tacoma | L13.42

42

CLOUD ENABLING TECHNOLOGY

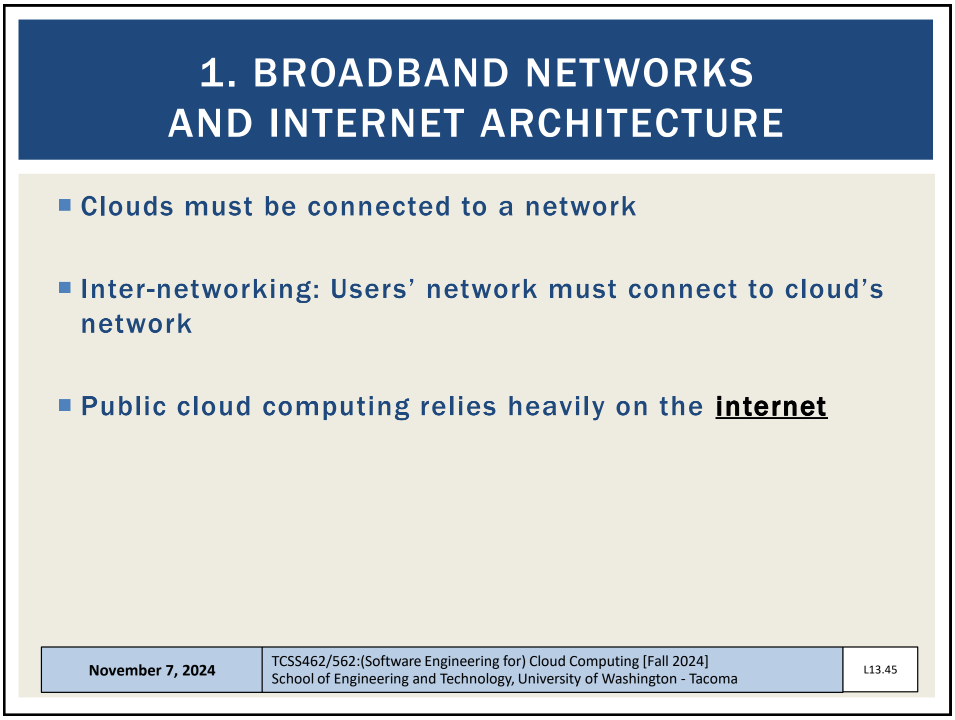
43



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

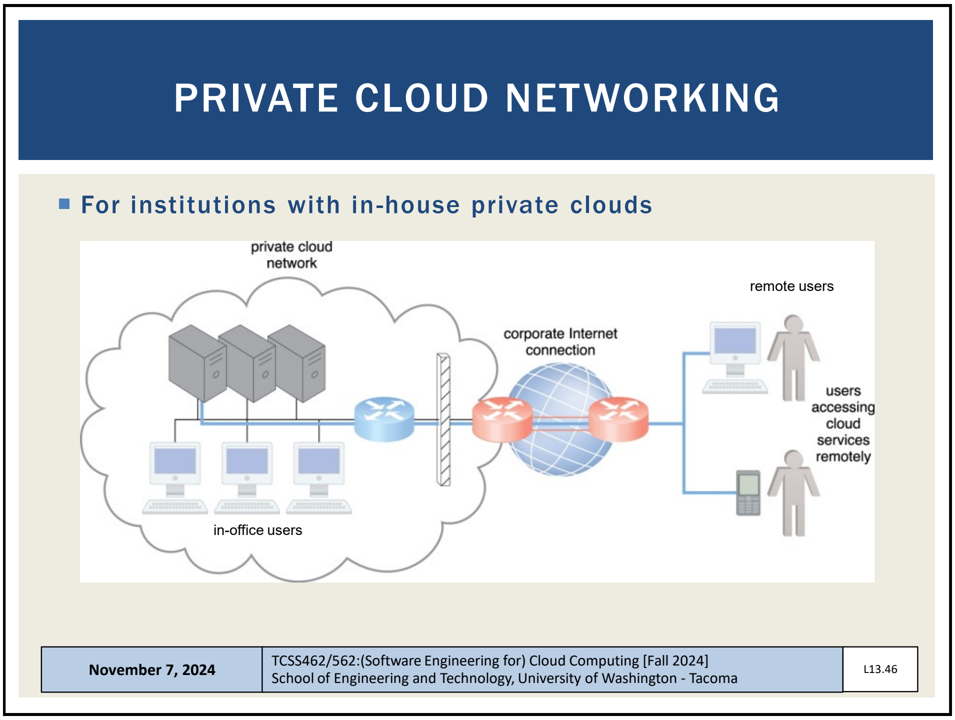
44



1. BROADBAND NETWORKS AND INTERNET ARCHITECTURE

- Clouds must be connected to a network
- Inter-networking: Users' network must connect to cloud's network
- Public cloud computing relies heavily on the **internet**

45



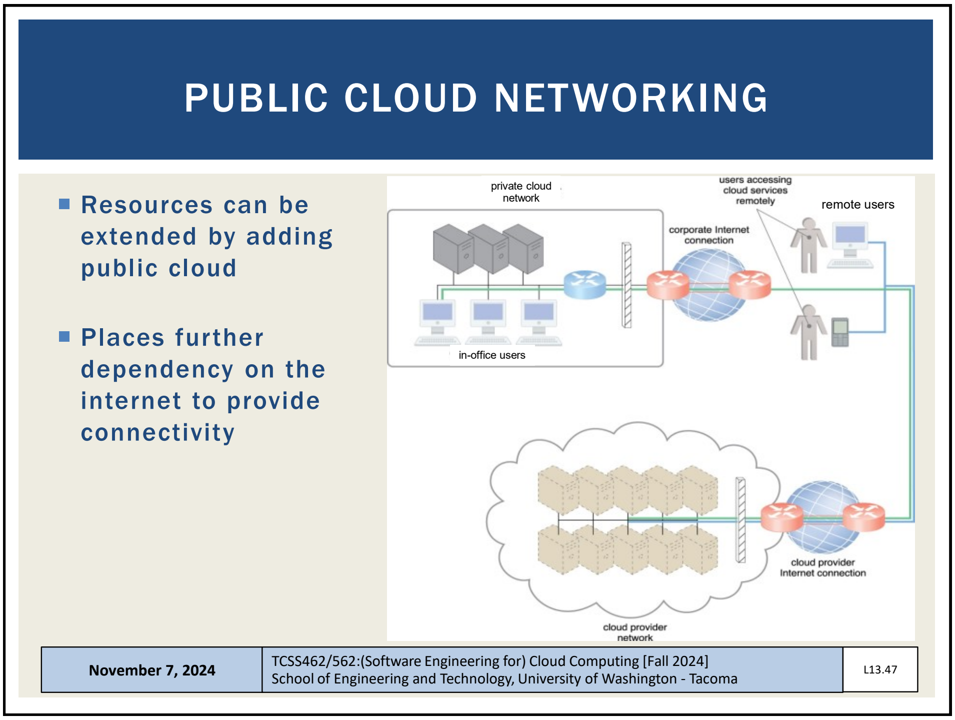
PRIVATE CLOUD NETWORKING

- For institutions with in-house private clouds

private cloud network
remote users
corporate Internet connection
users accessing cloud services remotely
in-office users

November 7, 2024 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma | L13.46

46



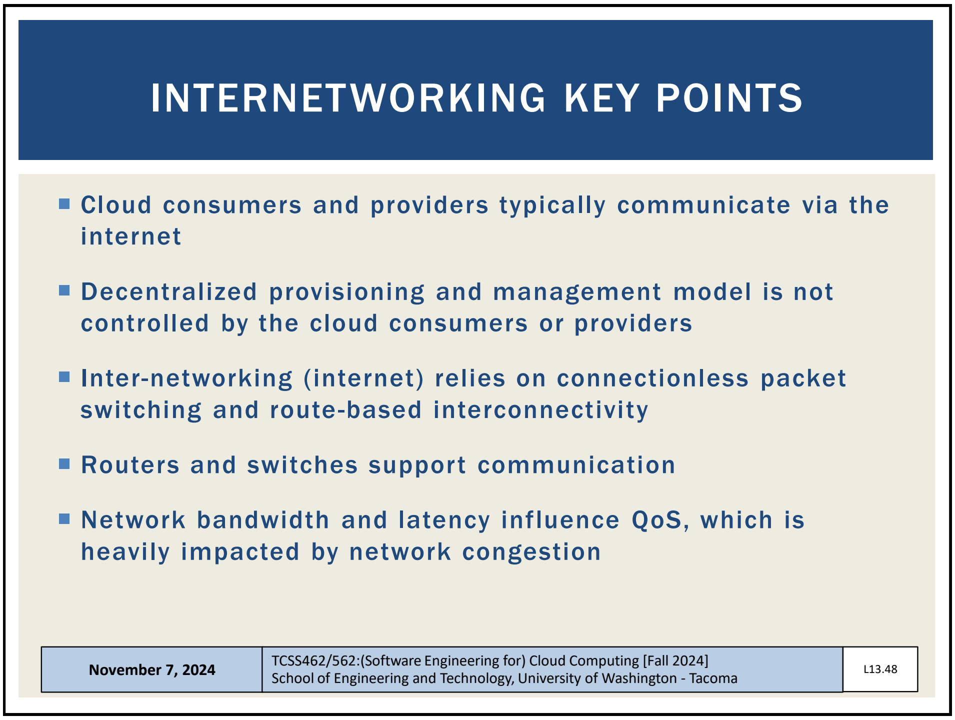
PUBLIC CLOUD NETWORKING

- Resources can be extended by adding public cloud
- Places further dependency on the internet to provide connectivity

private cloud network
users accessing cloud services remotely
remote users
corporate Internet connection
in-office users
cloud provider Internet connection
cloud provider network

November 7, 2024 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma | L13.47

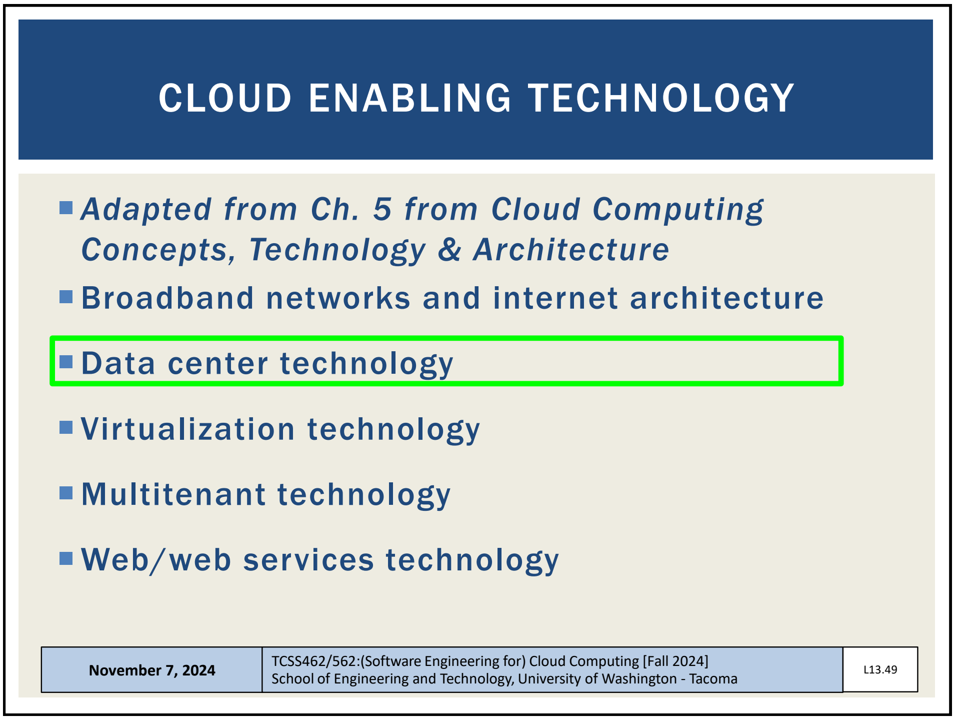
47



INTERNETWORKING KEY POINTS

- Cloud consumers and providers typically communicate via the internet
- Decentralized provisioning and management model is not controlled by the cloud consumers or providers
- Inter-networking (internet) relies on connectionless packet switching and route-based interconnectivity
- Routers and switches support communication
- Network bandwidth and latency influence QoS, which is heavily impacted by network congestion

48



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

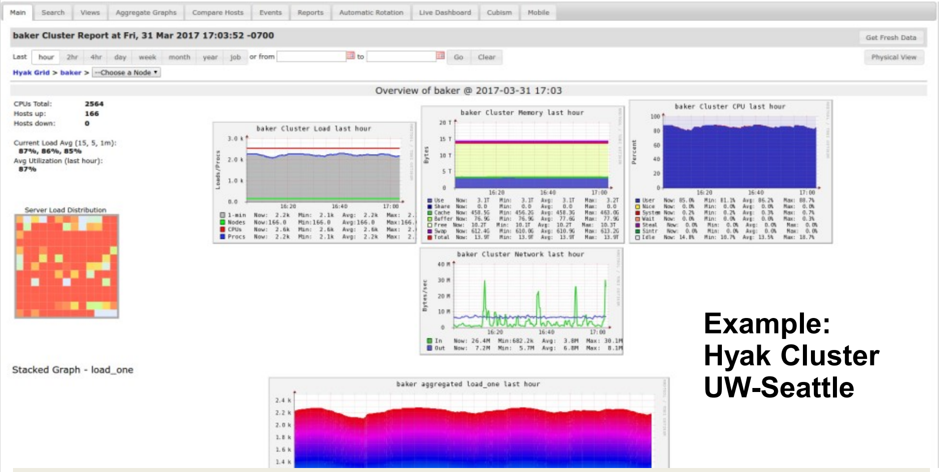
49

2. DATA CENTER TECHNOLOGY

- Grouping servers together (clusters):
- Enables power sharing
- Higher efficiency in shared IT resource usage (less duplication of effort)
- Improved accessibility and organization

- Key components:
  - Virtualized and physical server resources
  - Standardized, modular hardware
  - Automation support: enable server provisioning, configuration, patching, monitoring without supervision… ***tool/API support is desirable***

CLUSTER MANAGEMENT TOOLS



baker Cluster Report at Fri, 31 Mar 2017 17:03:52 -0700

Last hour 2hr 4hr day week month year job or from to Go Clear

Hyak Grid > baker > --Choose a Node ▾

Overview of baker @ 2017-03-31 17:03

CPUs Total: 2564
Hosts up: 166
Hosts down: 0

Current Load Avg (15, 5, 1m):
87%, 86%, 85%
Avg Utilization (last hour):
87%

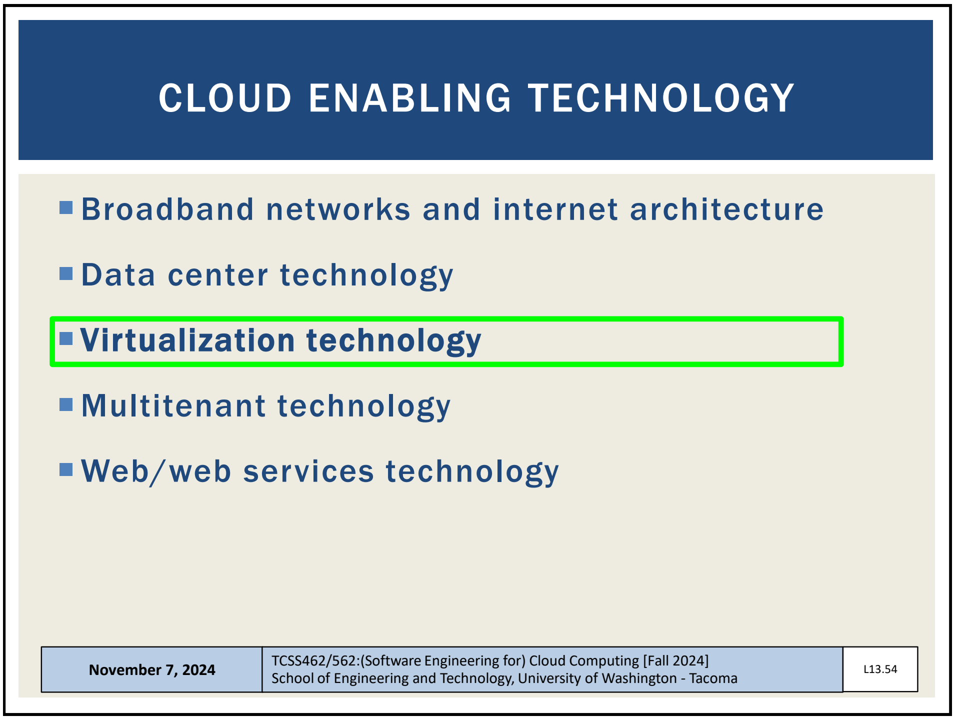
Stacked Graph - load_one

Example:
Hyak Cluster
UW-Seattle

DATA CENTER TECHNOLOGY – KEY COMPONENTS

- Remote operation / management
- **High availability support**: **redundant everything** Includes: power supplies, cabling, environmental control systems, communication links, duplicate warm replica HW
- **Secure design**: physical and logical access control
- **Servers**: rackmount, etc.
- **Storage**: hard disk arrays (RAID)
- storage area network (SAN): disk array w/ multiple servers (individual nodes w/ disks) and a dedicated network
- network attached storage (NAS): inexpensive single node with collection of disks, provides shared filesystems, for NFS, etc.
- **Network hardware**: backbone routers (WAN to LAN connectivity), firewalls, VPN gateways, managed switches/routers

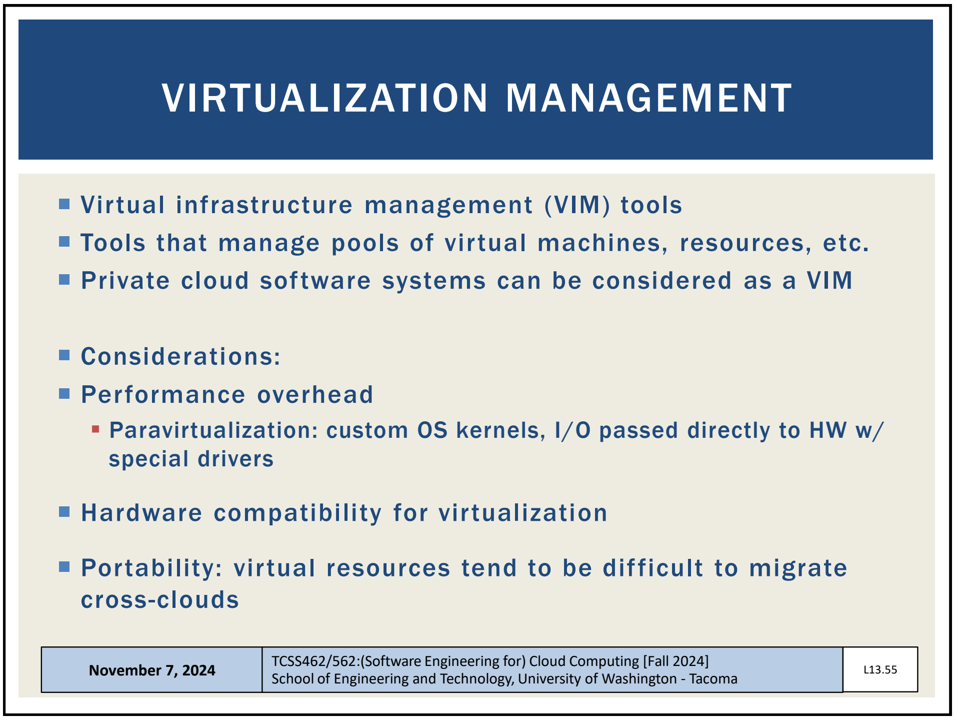
CLOUD ENABLING TECHNOLOGY



CLOUD ENABLING TECHNOLOGY

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

54

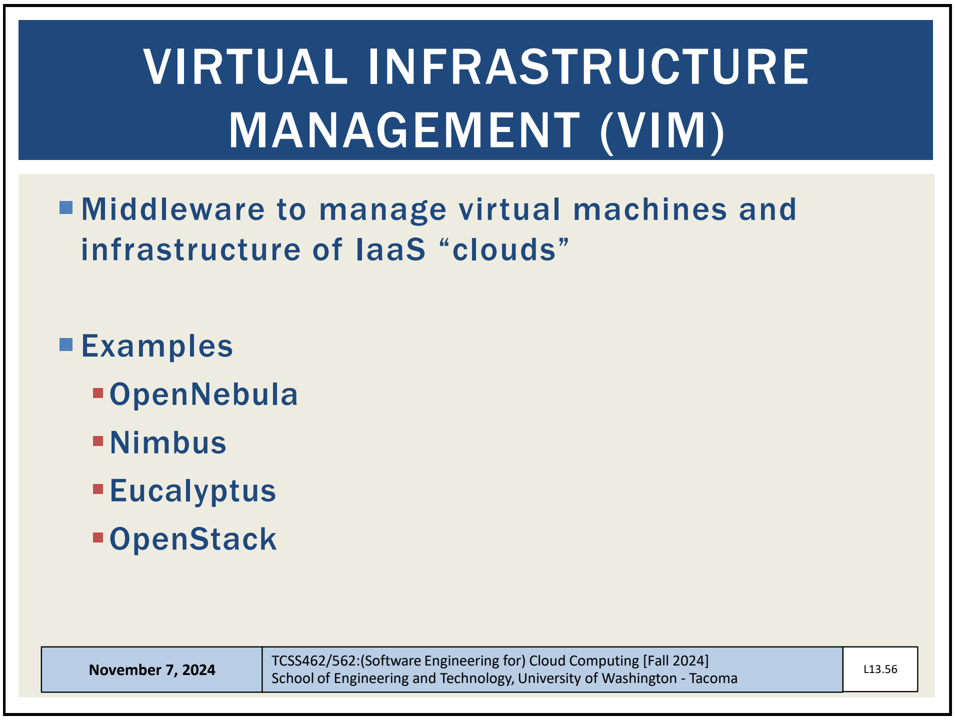


VIRTUALIZATION MANAGEMENT

- 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

55

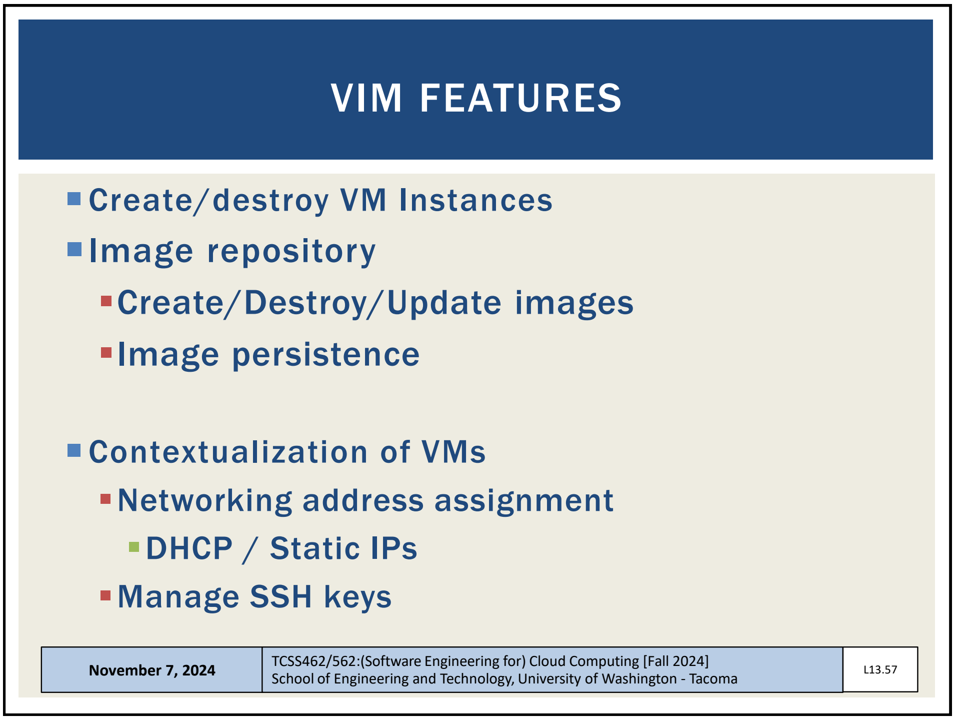


VIRTUAL INFRASTRUCTURE MANAGEMENT (VIM)

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

November 7, 2024 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma | L13.56

56

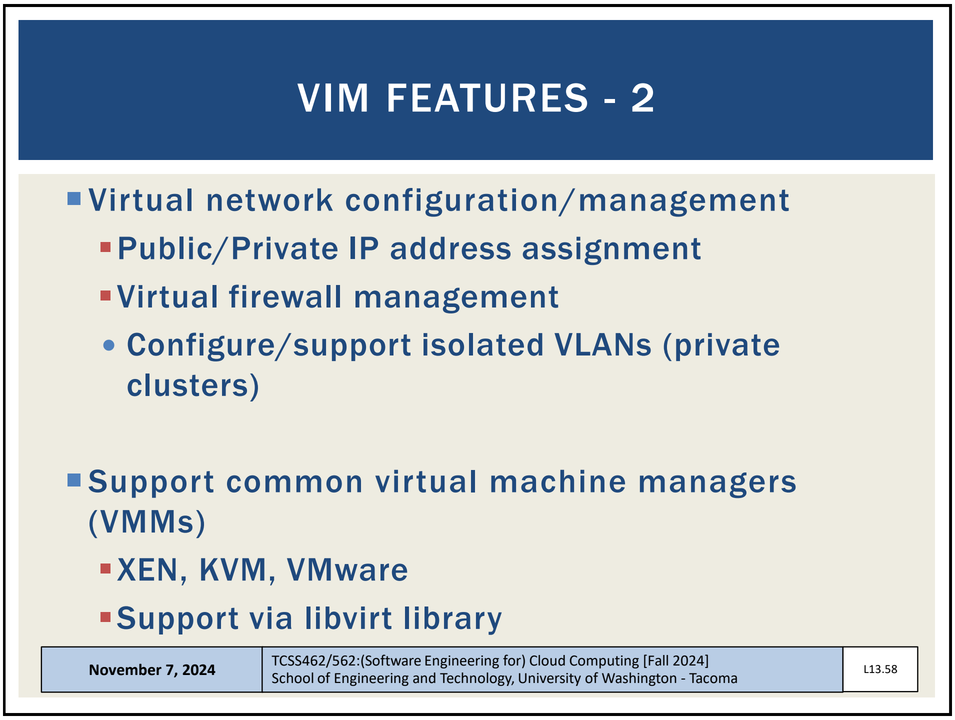


VIM FEATURES

- 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 7, 2024 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma | L13.57

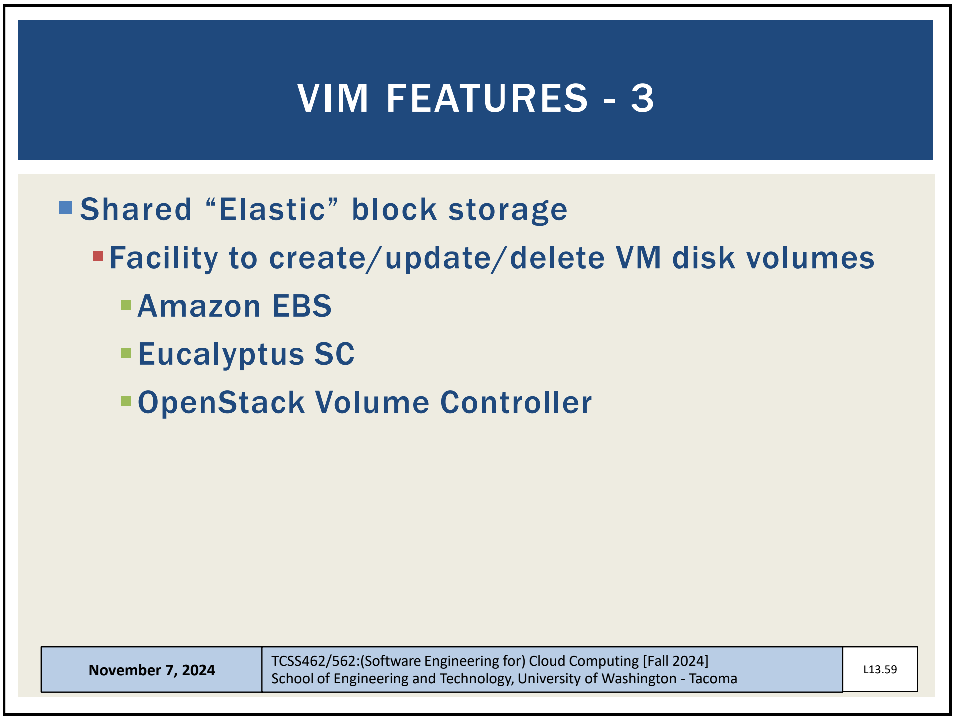
57



VIM FEATURES - 2

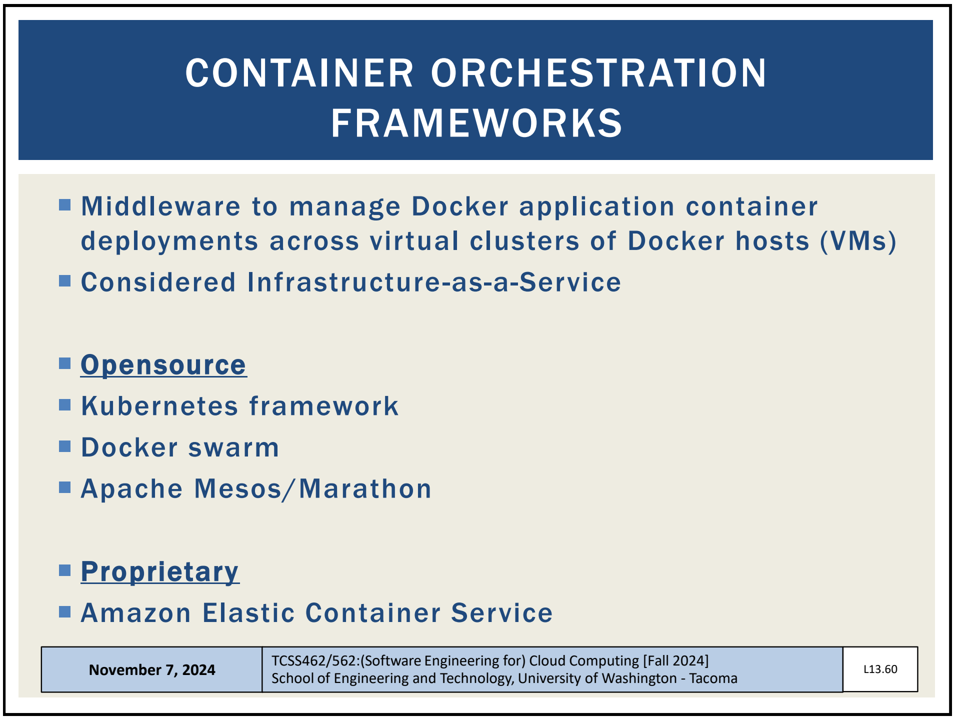
- Virtual network configuration/management
  - Public/Private IP address assignment
  - Virtual firewall management
  - Configure/support isolated VLANs (private clusters)

- Support common virtual machine managers (VMMs)
  - XEN, KVM, VMware
  - Support via libvirt library



VIM FEATURES - 3

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



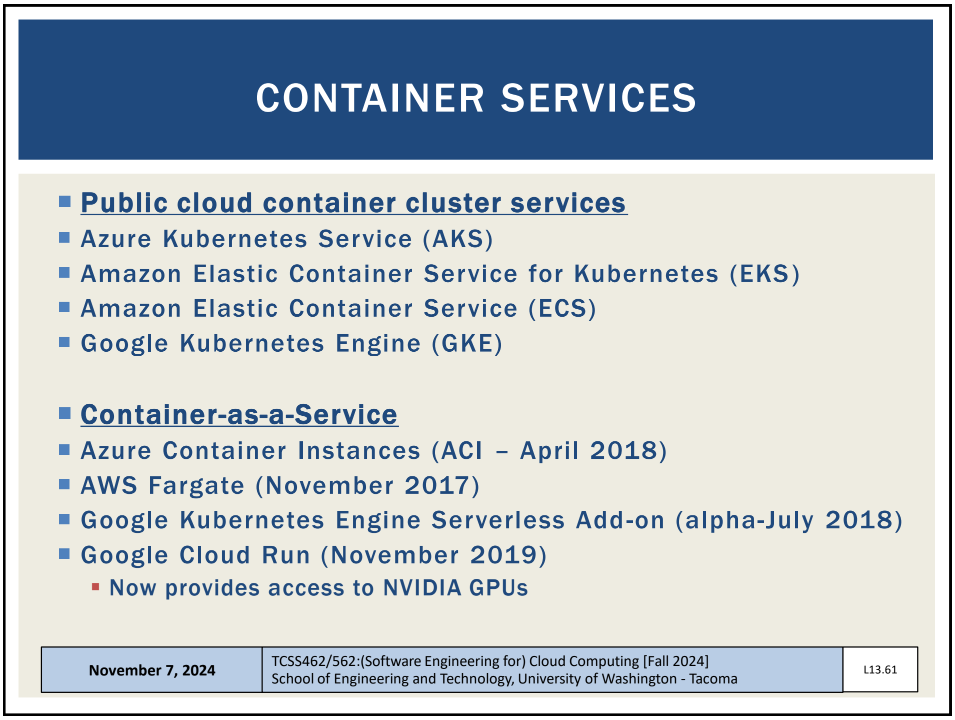
CONTAINER ORCHESTRATION FRAMEWORKS

- 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

60

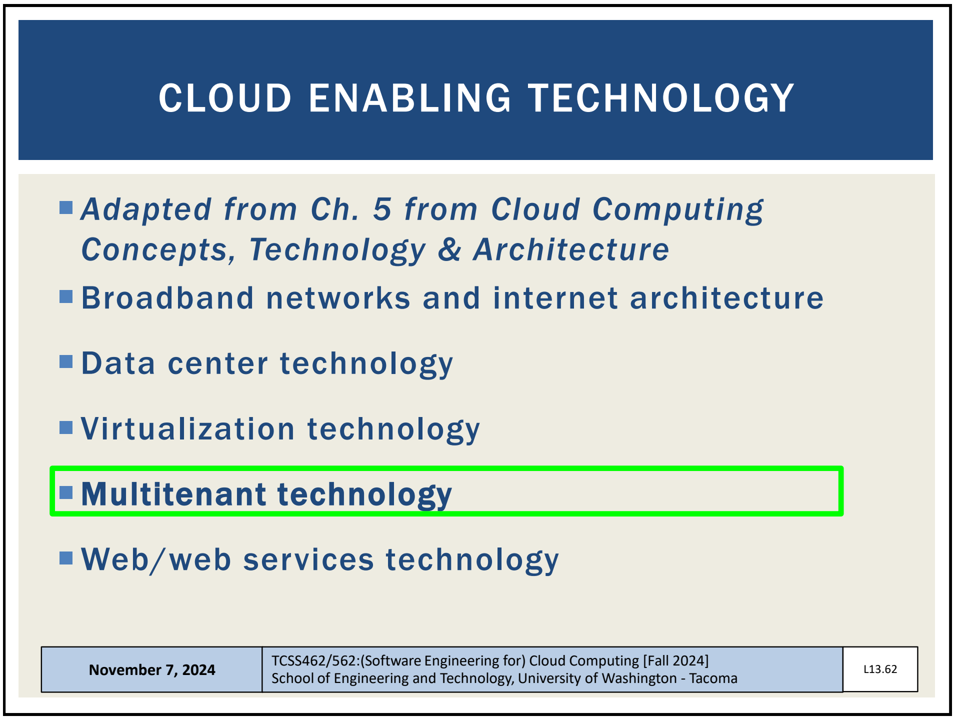


CONTAINER SERVICES

- **Public cloud container cluster services**
- Azure Kubernetes Service (AKS)
- Amazon Elastic Container Service for Kubernetes (EKS)
- Amazon Elastic Container Service (ECS)
- Google Kubernetes Engine (GKE)

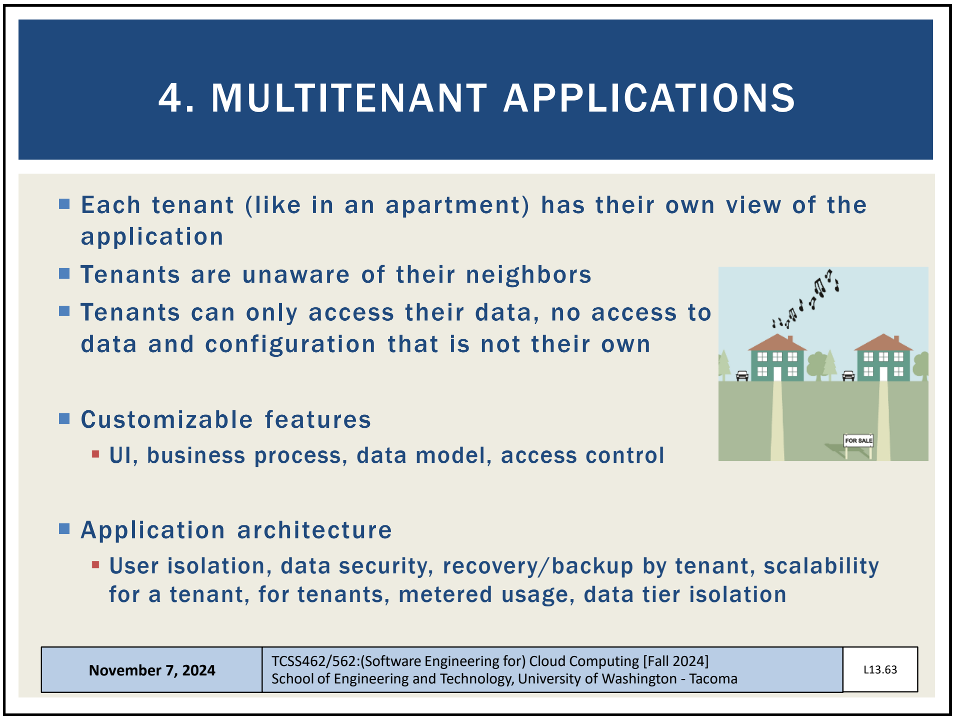
- **Container-as-a-Service**
- Azure Container Instances (ACI – April 2018)
- AWS Fargate (November 2017)
- Google Kubernetes Engine Serverless Add-on (alpha-July 2018)
- Google Cloud Run (November 2019)
  - Now provides access to NVIDIA GPUs

61



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

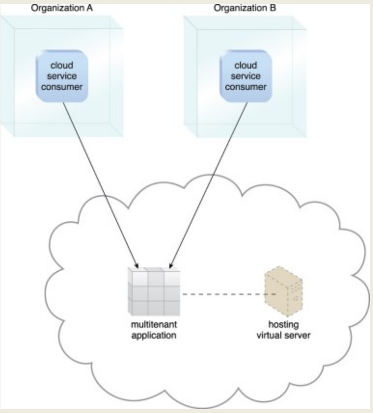


4. MULTITENANT APPLICATIONS

- 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

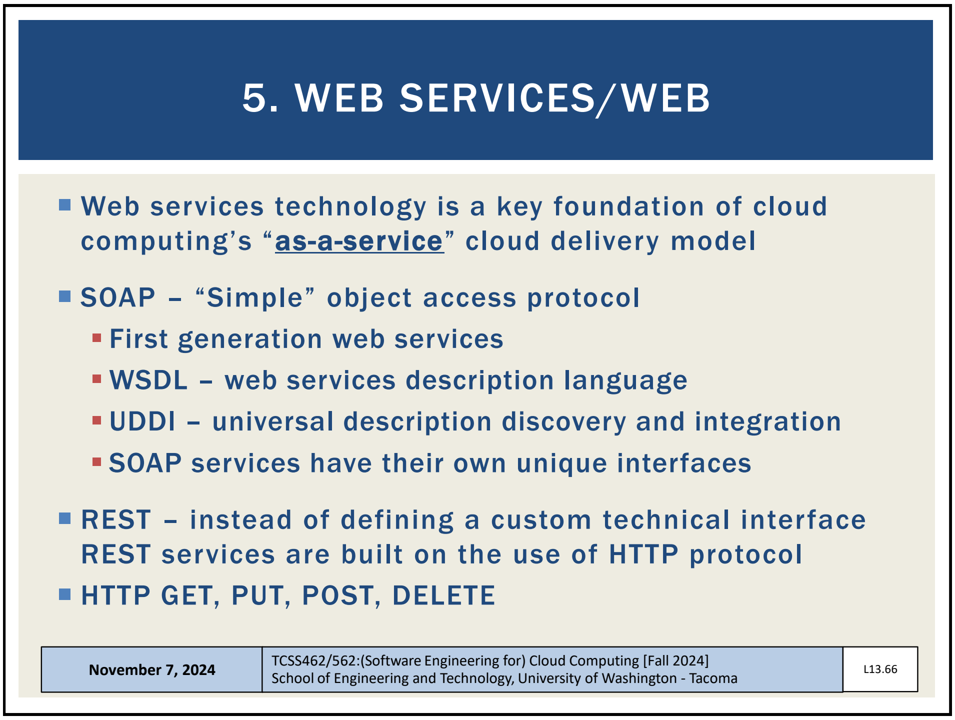
MULTITENANT APPS - 2

- Forms the basis for SaaS (applications)



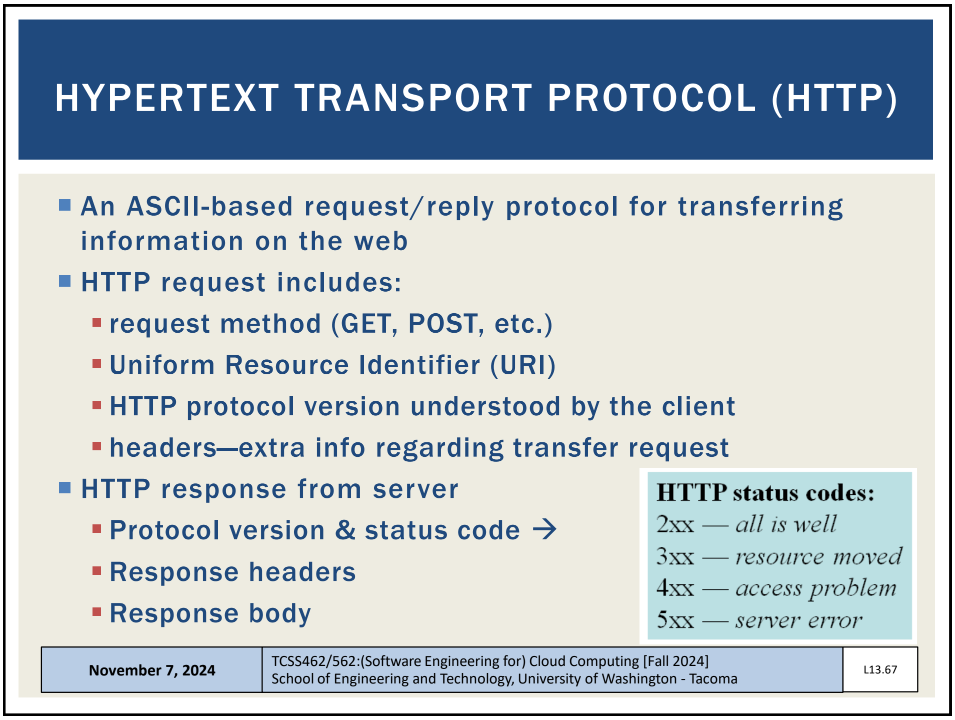
CLOUD ENABLING TECHNOLOGY

- *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**



5. WEB SERVICES/WEB

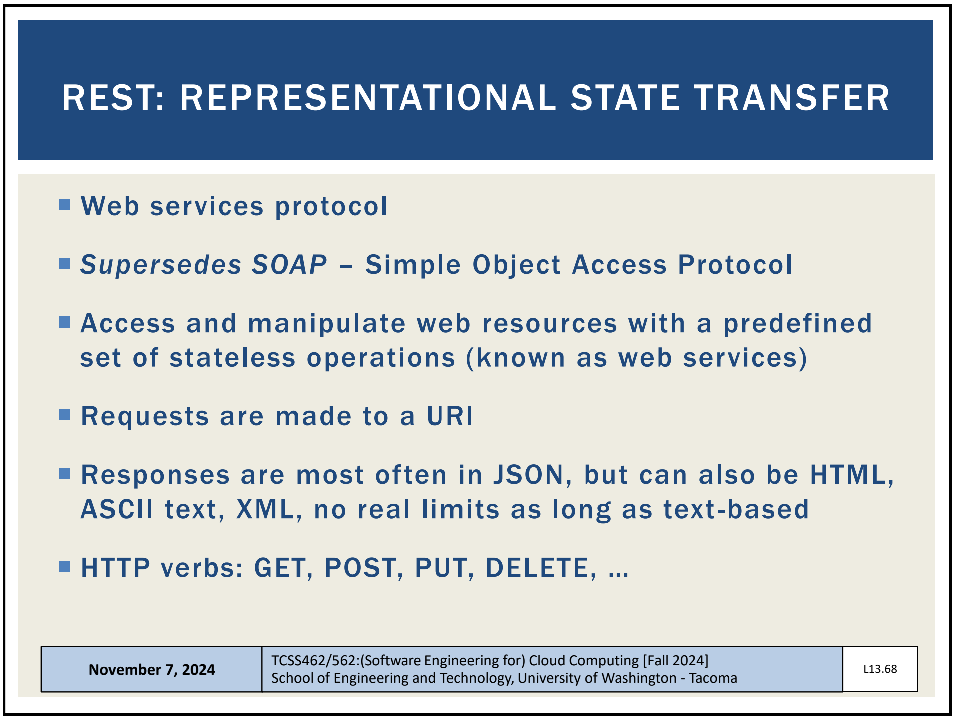
- Web services technology is a key foundation of cloud computing's "**as-a-service**" cloud delivery model
- SOAP – "Simple" object access protocol
  - First generation web services
  - WSDL – web services description language
  - UDDI – universal description discovery and integration
  - SOAP services 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



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

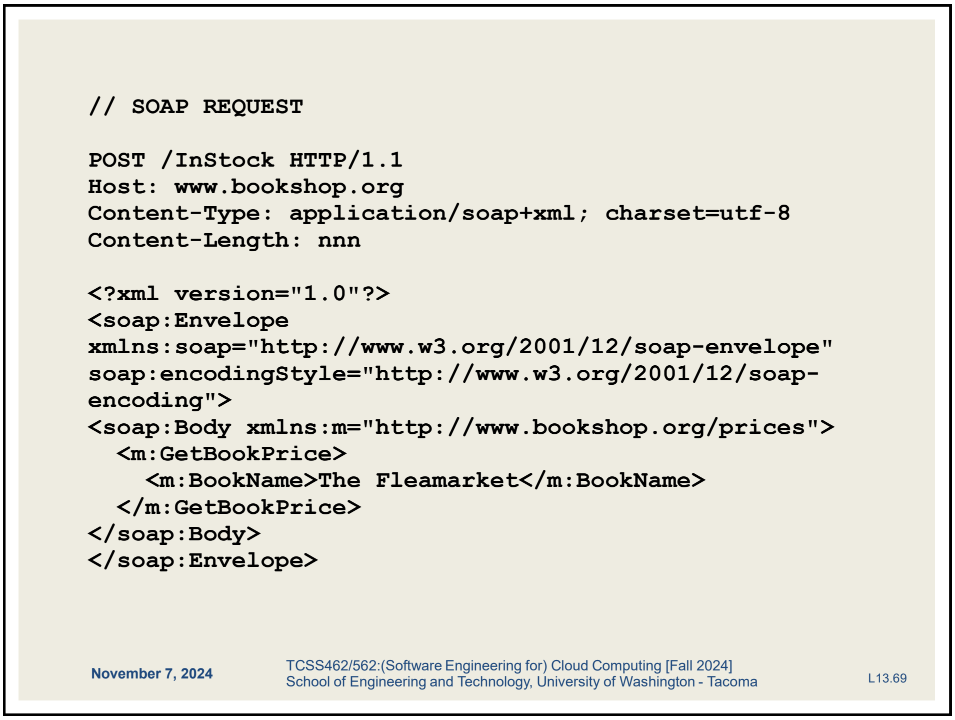
HTTP status codes:
2xx — *all is well*
3xx — *resource moved*
4xx — *access problem*
5xx — *server error*



REST: REPRESENTATIONAL STATE TRANSFER

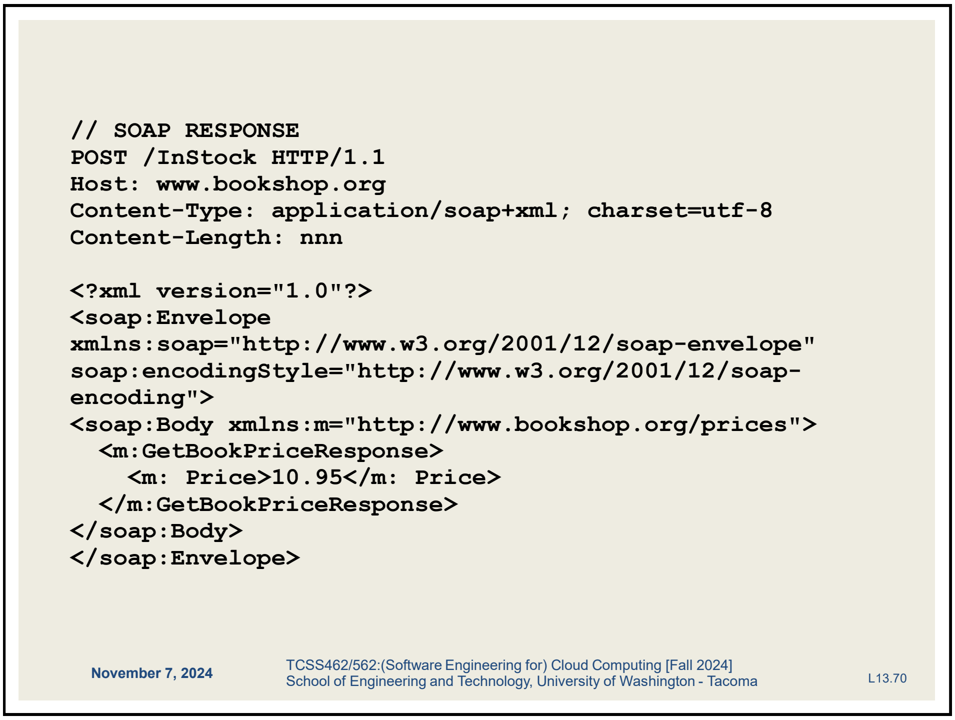
- Web services protocol
- *Supersedes SOAP* – Simple Object Access Protocol
- Access and manipulate web resources with a predefined set of stateless operations (known as web services)
- Requests are made to a URI
- Responses are most often in JSON, but can also be HTML, ASCII text, XML, no real limits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

68



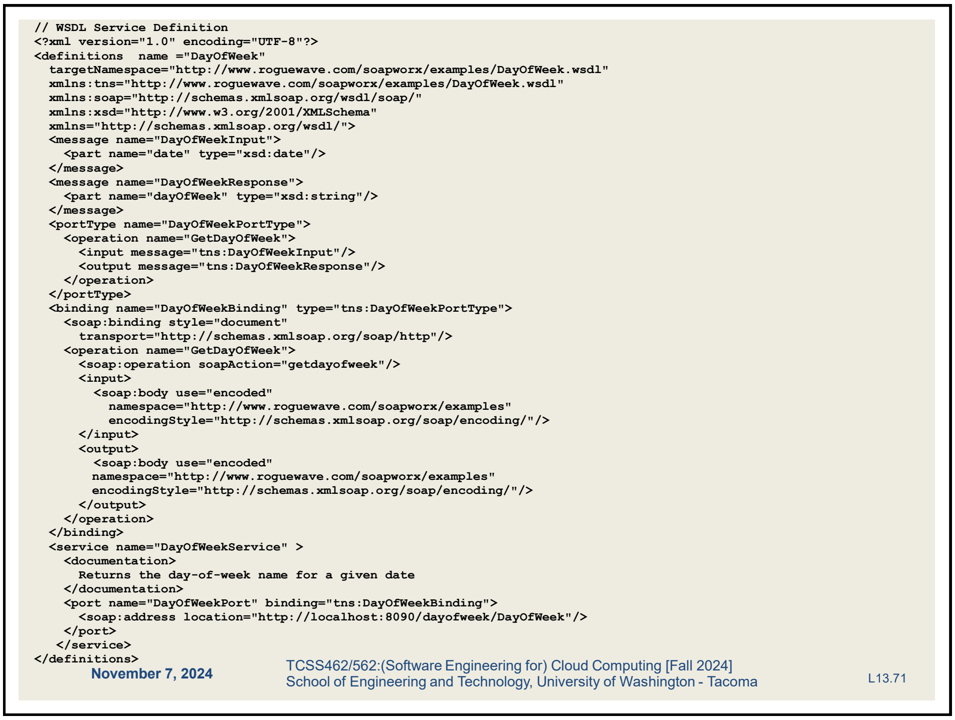
```
// SOAP REQUEST

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>
```

69



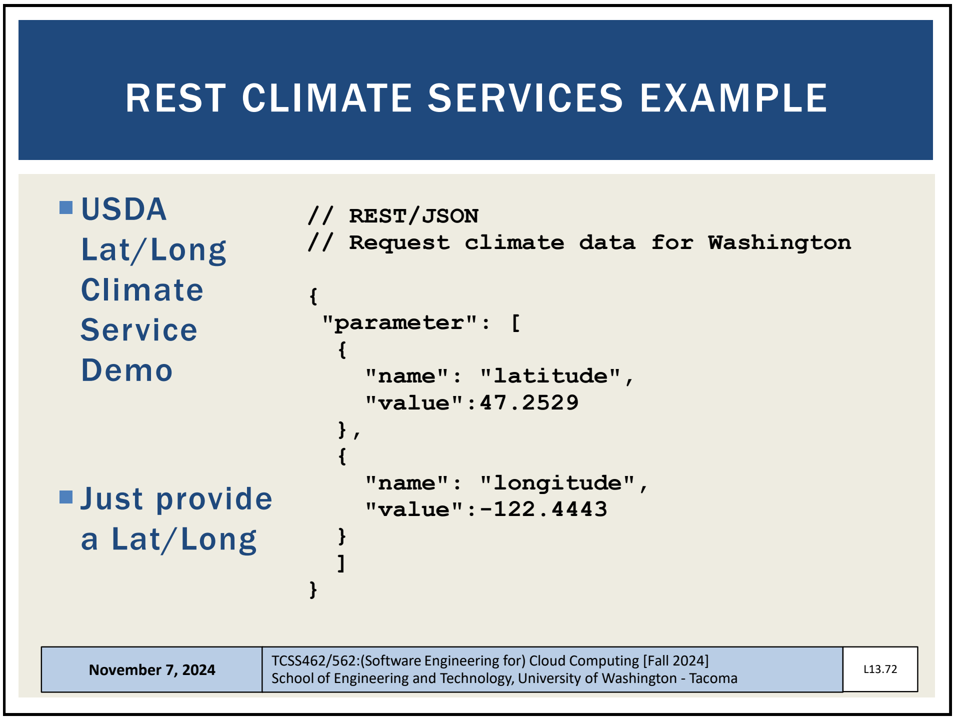
```
// SOAP RESPONSE
POST /InStock HTTP/1.1
Host: www.bookshop.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?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>
```

70



```
// WSDL Service Definition
<?xml version="1.0" encoding="UTF-8"?>
<definitions  name ="DayOfWeek"
  targetNamespace="http://www.roguewave.com/soapworx/examples/DayOfWeek.wsdl"
  xmlns:tns="http://www.roguewave.com/soapworx/examples/DayOfWeek.wsdl"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="DayOfWeekInput">
    <part name="date" type="xsd:date"/>
  </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="encoded"
          namespace="http://www.roguewave.com/soapworx/examples"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded"
        namespace="http://www.roguewave.com/soapworx/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>
```

71

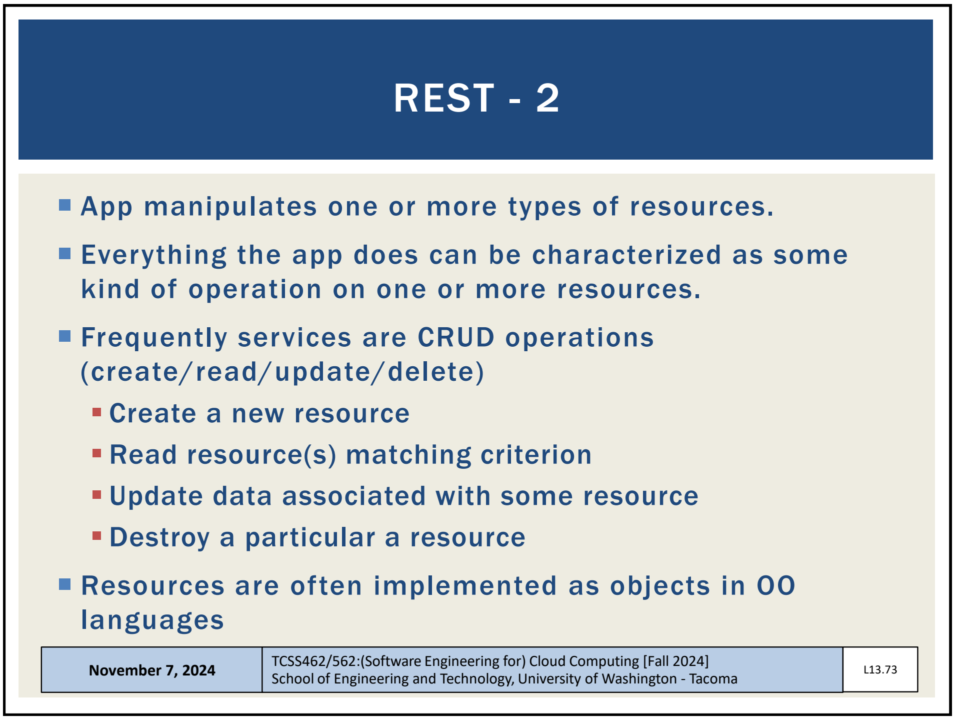


REST CLIMATE SERVICES EXAMPLE

- 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
  }
  ]
}
```

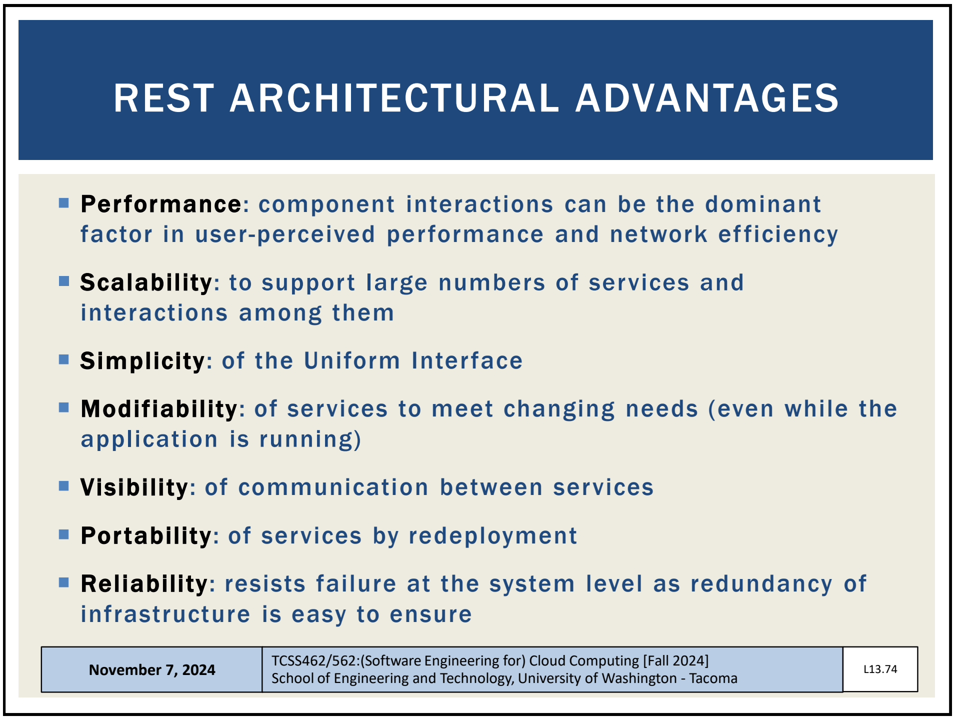
72



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

73



REST ARCHITECTURAL ADVANTAGES

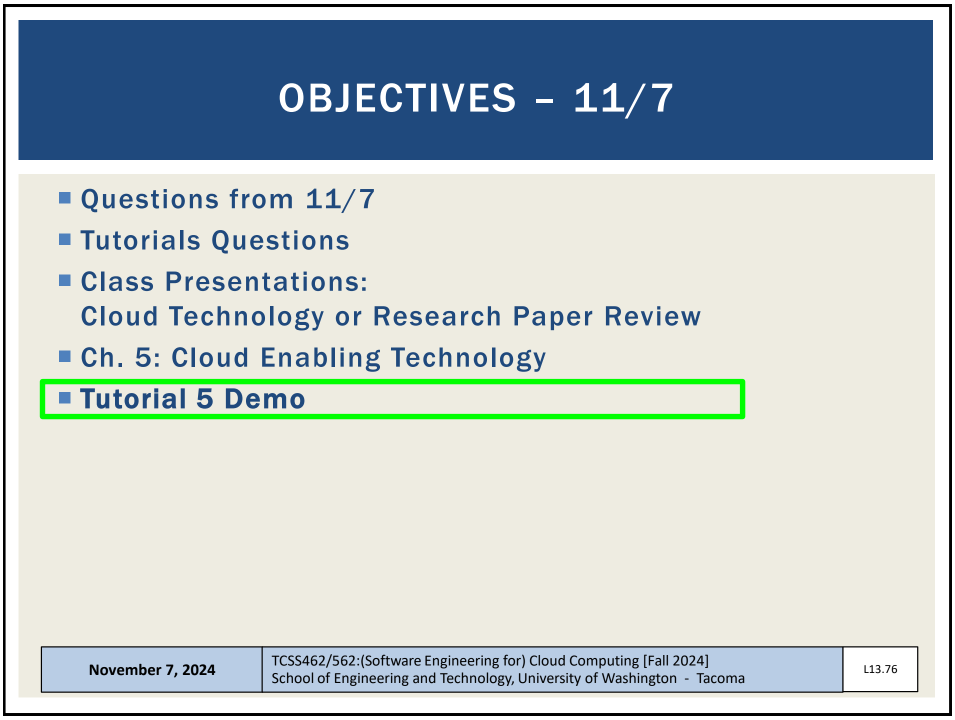
- **Performance**: component interactions can be the dominant factor in user-perceived performance and network efficiency
- **Scalability**: to support large numbers of services and interactions among them
- **Simplicity**: of the Uniform Interface
- **Modifiability**: of services to meet changing needs (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 7, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L13.74

74

WE WILL RETURN AT ~4:50 PM

75

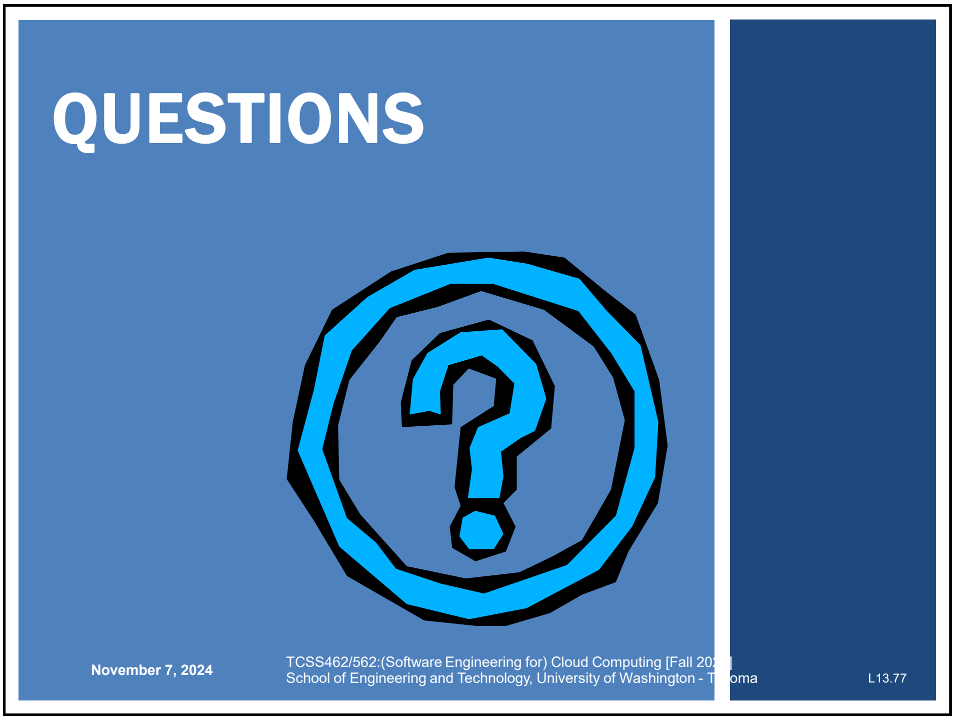


OBJECTIVES – 11/7

- Questions from 11/7
- Tutorials Questions
- Class Presentations:
  Cloud Technology or Research Paper Review
- Ch. 5: Cloud Enabling Technology
- **Tutorial 5 Demo**

November 7, 2024 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]
School of Engineering and Technology, University of Washington - Tacoma | L13.76

76



QUESTIONS

November 7, 2024 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 20]
School of Engineering and Technology, University of Washington - T oma | L13.77

77