TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

AWS Demo, Cloud Enabling Technology

Wes J. Lloyd
School of Engineering and Technology
University of Washington – Tacoma

MW 5:50-7:50 PM

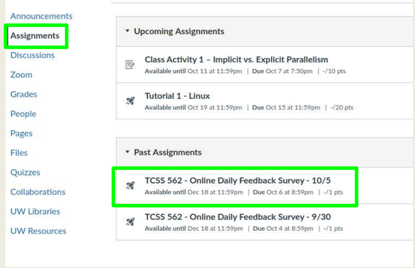
OBJECTIVES – 11/2

- Questions from 10/28
- Term Project Proposals
- Quiz 1 Review
- AWS overview and demonstration

- 2nd hour:
- Tutorial #5
- AWS overview and demonstration
- Cloud Enabling Technology
- Tutorial questions
- Team planning

ONLINE DAILY FEEDBACK SURVEY

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



▾ Upcoming Assignments

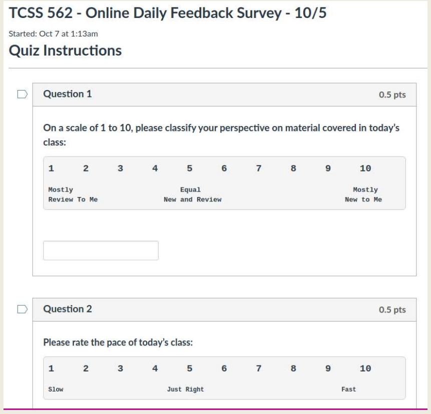
Class Activity 1 – Implicit vs. Explicit Parallelism
Available until Oct 11 at 11:59pm | Due Oct 7 at 7:50pm | -/10 pts

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
```

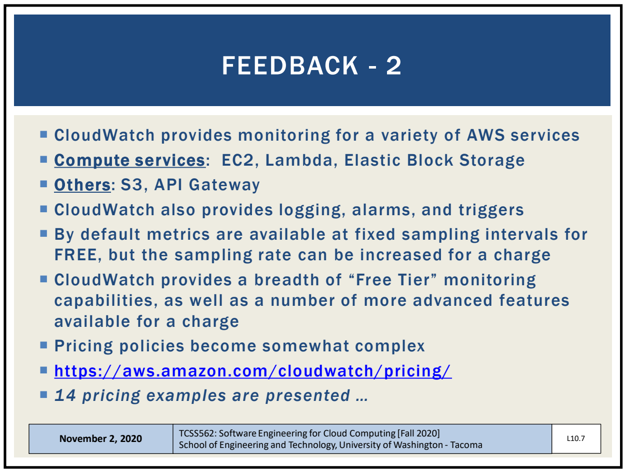
MATERIAL / PACE

- Please classify your perspective on material covered in today's class (25 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.60** (↓ - *previous 6.95*)

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.52** (↓ - *previous 6.14*)

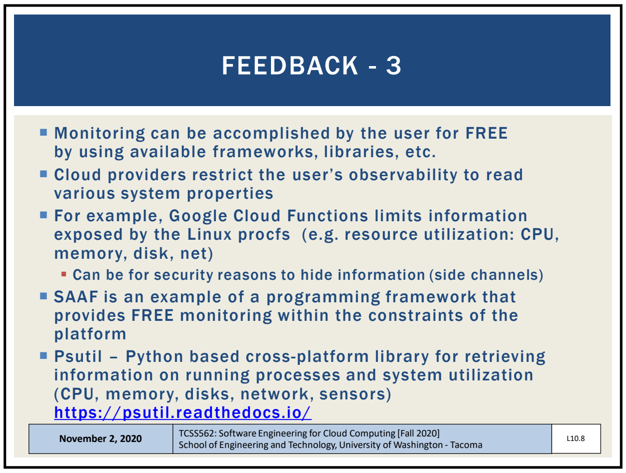
FEEDBACK FROM 10/28

- ***Can you please explain the difference between Metrics Server API and AWS Cloud Watch.***
- ***As per my understanding CloudWatch is provided by AWS to monitor EC2 instances. Metrics Server API can also be used to monitor CPU Utilization, Network and disk I/O statistics but has to be manually installed.***
- ***Can you please explain if there are more differences?***
- The Metrics Server API is a monitoring tool for Kubernetes
- On AWS, the Elastic Kubernetes Service (EKS) can be used to automate the setup and management of Kubernetes container clusters
- The Metrics Server API is then installed onto the cluster:
- https://docs.aws.amazon.com/eks/latest/userguide/metrics-server.html



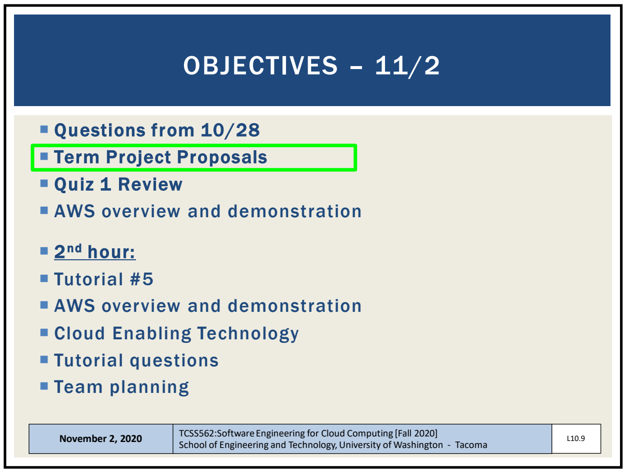
FEEDBACK - 2

- CloudWatch provides monitoring for a variety of AWS services
- **Compute services**: EC2, Lambda, Elastic Block Storage
- **Others**: S3, API Gateway
- CloudWatch also provides logging, alarms, and triggers
- By default metrics are available at fixed sampling intervals for FREE, but the sampling rate can be increased for a charge
- CloudWatch provides a breadth of "Free Tier" monitoring capabilities, as well as a number of more advanced features available for a charge
- Pricing policies become somewhat complex
- https://aws.amazon.com/cloudwatch/pricing/
- *14 pricing examples are presented …*



FEEDBACK - 3

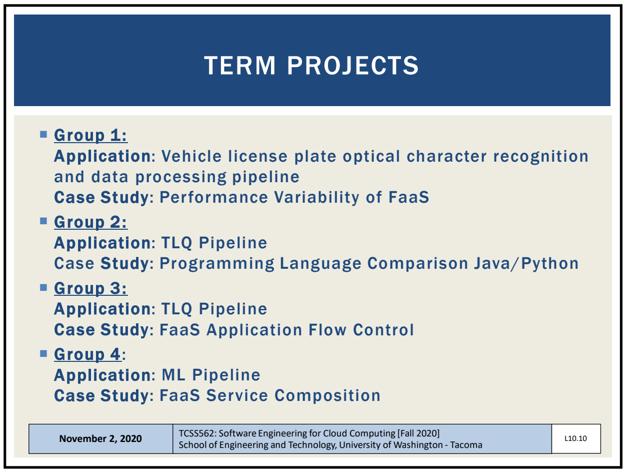
- Monitoring can be accomplished by the user for FREE by using available frameworks, libraries, etc.
- Cloud providers restrict the user's observability to read various system properties
- For example, Google Cloud Functions limits information exposed by the Linux procfs (e.g. resource utilization: CPU, memory, disk, net)
  - Can be for security reasons to hide information (side channels)
- SAAF is an example of a programming framework that provides FREE monitoring within the constraints of the platform
- Psutil – Python based cross-platform library for retrieving information on running processes and system utilization (CPU, memory, disks, network, sensors) https://psutil.readthedocs.io/



OBJECTIVES – 11/2

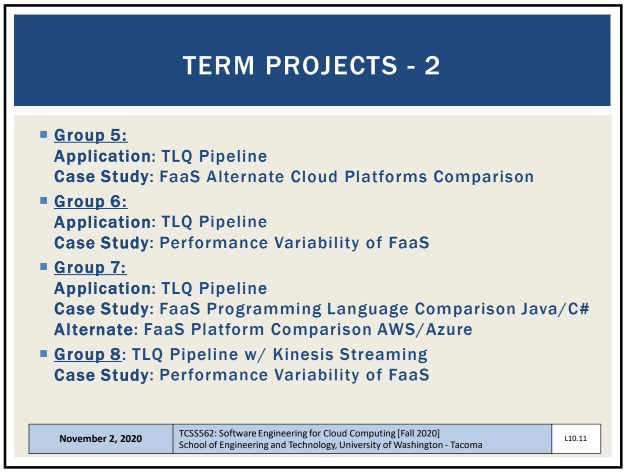
- **Questions from 10/28**
- **Term Project Proposals**
- **Quiz 1 Review**
- AWS overview and demonstration

- **2nd hour:**
- Tutorial #5
- AWS overview and demonstration
- Cloud Enabling Technology
- Tutorial questions
- Team planning



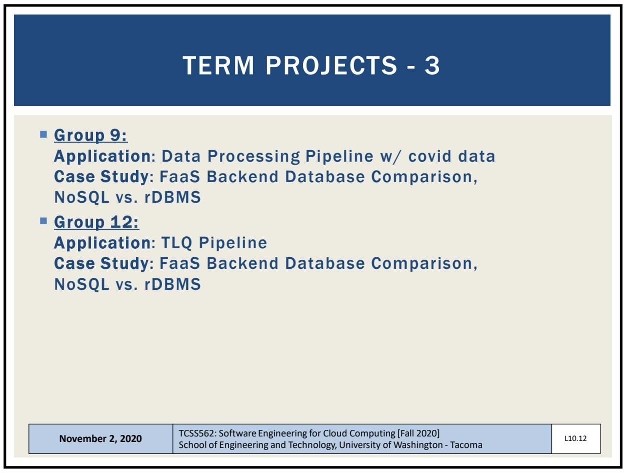
TERM PROJECTS

- **Group 1:**
  Application: Vehicle license plate optical character recognition and data processing pipeline
  Case Study: Performance Variability of FaaS
- **Group 2:**
  Application: TLQ Pipeline
  Case Study: Programming Language Comparison Java/Python
- **Group 3:**
  Application: TLQ Pipeline
  Case Study: FaaS Application Flow Control
- **Group 4**:
  Application: ML Pipeline
  Case Study: FaaS Service Composition



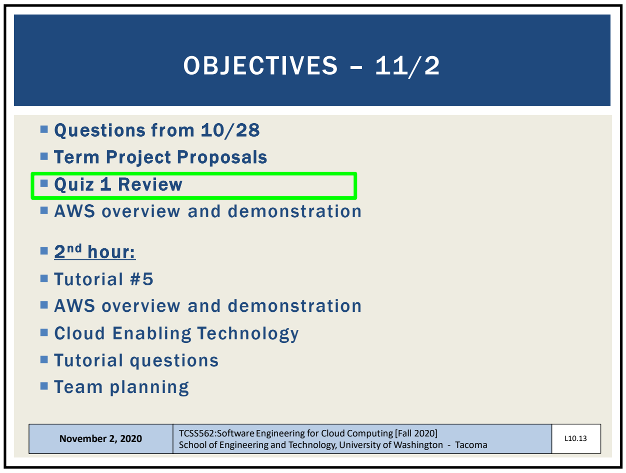
TERM PROJECTS - 2

- **Group 5:**
  Application: TLQ Pipeline
  Case Study: FaaS Alternate Cloud Platforms Comparison
- **Group 6:**
  Application: TLQ Pipeline
  Case Study: Performance Variability of FaaS
- **Group 7:**
  Application: TLQ Pipeline
  Case Study: FaaS Programming Language Comparison Java/C#
  Alternate: FaaS Platform Comparison AWS/Azure
- **Group 8**: TLQ Pipeline w/ Kinesis Streaming
  Case Study: Performance Variability of FaaS



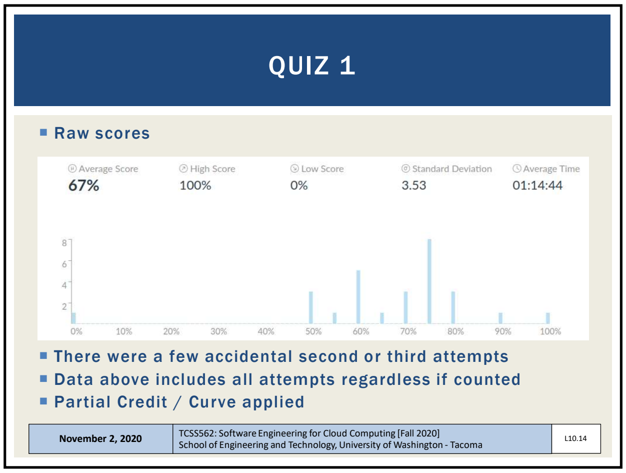
TERM PROJECTS - 3

- **Group 9:**
  Application: Data Processing Pipeline w/ covid data
  Case Study: FaaS Backend Database Comparison, NoSQL vs. rDBMS
- **Group 12:**
  Application: TLQ Pipeline
  Case Study: FaaS Backend Database Comparison, NoSQL vs. rDBMS



OBJECTIVES – 11/2

- Questions from 10/28
- Term Project Proposals
- Quiz 1 Review
- AWS overview and demonstration
- 2nd hour:
- Tutorial #5
- AWS overview and demonstration
- Cloud Enabling Technology
- Tutorial questions
- Team planning

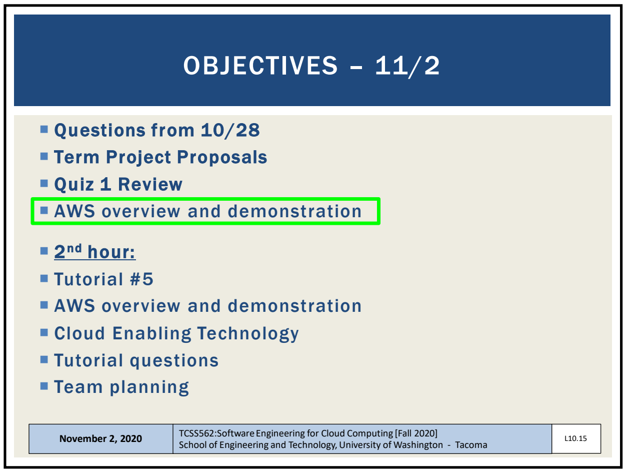


QUIZ 1

- Raw scores

Average Score 67%
High Score 100%
Low Score 0%
Standard Deviation 3.53
Average Time 01:14:44

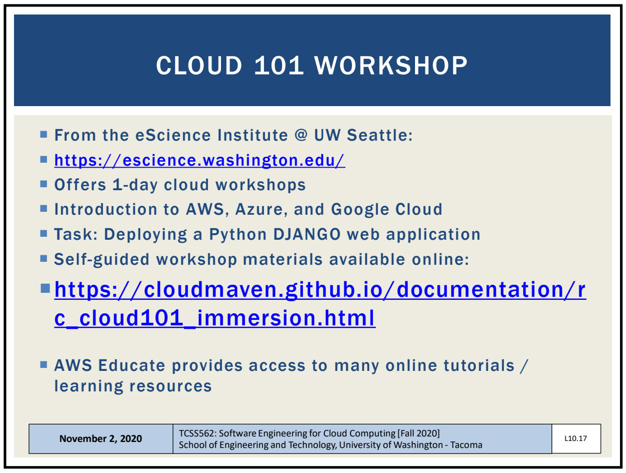
- There were a few accidental second or third attempts
- Data above includes all attempts regardless if counted
- Partial Credit / Curve applied



OBJECTIVES – 11/2

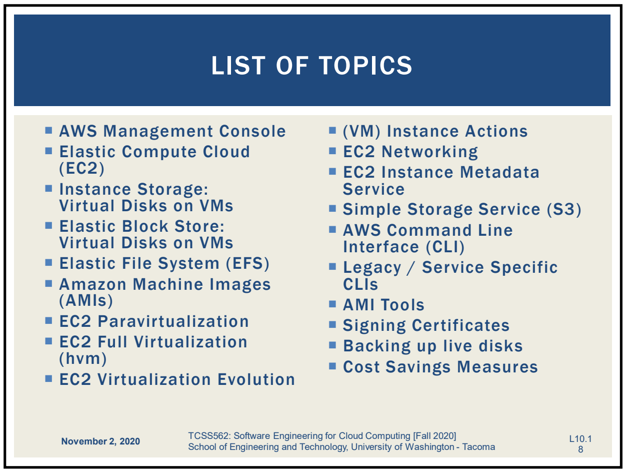
- Questions from 10/28
- Term Project Proposals
- Quiz 1 Review
- AWS overview and demonstration
- 2nd hour:
- Tutorial #5
- AWS overview and demonstration
- Cloud Enabling Technology
- Tutorial questions
- Team planning

AWS DEMO



CLOUD 101 WORKSHOP

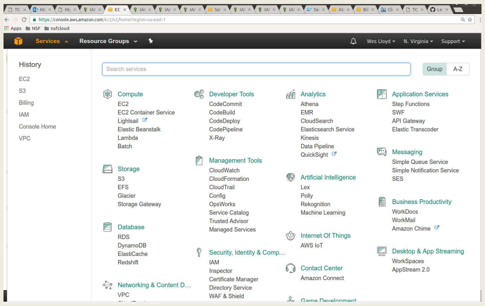
- From the eScience Institute @ UW Seattle:
- https://escience.washington.edu/
- Offers 1-day cloud workshops
- Introduction to AWS, Azure, and Google Cloud
- Task: Deploying a Python DJANGO web application
- Self-guided workshop materials available online:
- https://cloudmaven.github.io/documentation/rc_cloud101_immersion.html
- AWS Educate provides access to many online tutorials / learning resources



LIST OF TOPICS

- AWS Management Console
- Elastic Compute Cloud (EC2)
- Instance Storage: Virtual Disks on VMs
- Elastic Block Store: Virtual Disks on VMs
- Elastic File System (EFS)
- Amazon Machine Images (AMIs)
- EC2 Paravirtualization
- EC2 Full Virtualization (hvm)
- EC2 Virtualization Evolution
- (VM) Instance Actions
- EC2 Networking
- EC2 Instance Metadata Service
- Simple Storage Service (S3)
- AWS Command Line Interface (CLI)
- Legacy / Service Specific CLIs
- AMI Tools
- Signing Certificates
- Backing up live disks
- Cost Savings Measures

AWS MANAGEMENT CONSOLE



AWS EC2

- Elastic Compute Cloud
- Instance types: https://ec2instances.info
  - On demand instance – full price
  - Reserved instance – contract based
  - Spot instance – auction based, terminates with 2 minute warning
  - Dedicated/reserved host – reserved HW
  - Reserved host
  - Instance families:
    General, compute-optimized, memory-optimized, GPU, etc.
- Storage types
  - Instance storage - ephemeral storage
  - EBS - Elastic block store
  - EFS - Elastic file system

INSTANCE STORAGE

- Also called ephemeral storage
- Persisted using images saved to S3 (simple storage service)
  - ~2.3¢ per GB/month on S3
  - 5GB of free tier storage space on S3
- Requires "burning" an image
- Mutli-step process:
  - Create image files
  - Upload chunks to S3
  - Register image
- Launching a VM
  - Requires downloading image components from S3, reassembling them… is potentially slow
- VMs with instance store backed root volumes not pause-able
- Historically root volume limited to 10-GB max– ***faster imaging…***

ELASTIC BLOCK STORE

- EBS cost model is different than instance storage (uses S3)
  - ~10¢ per GB/month
  - 30GB of free tier storage space
- EBS provides "live" mountable volumes
  - Listed under volumes
  - **Data volumes**: can be mounted/unmounted to any VM, dynamically at any time
  - **Root volumes**: hosts OS files and acts as a boot device for VM
  - In Linux drives are linked to a mount point "directory"
- Snapshots back up EBS volume data to S3
  - Enables replication (required for horizontal scaling)
  - EBS volumes not actively used should be snapshotted, and deleted to save EBS costs…

EBS VOLUME TYPES - 2

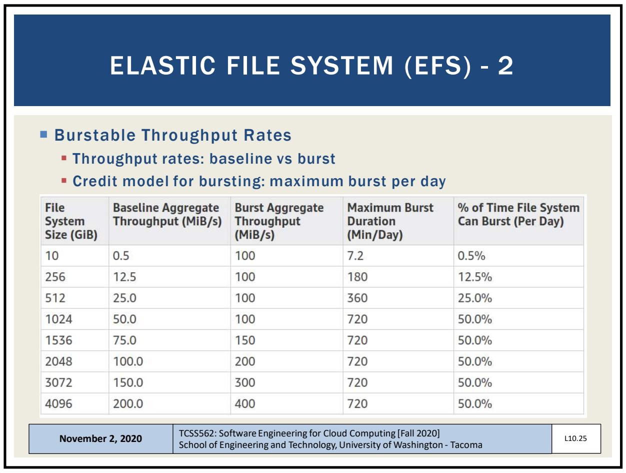
- Metric: I/O Operations per Second (IOPS)
- General Purpose 2 (GP2)
  - 3 IOPS per GB, Max 10,000 IOPS, 160MB/sec per volume
- Provisioned IOPS (IO1)
  - 32,000 IOPS, and 500 MB/sec throughput per volume
- Throughput Optimized HDD (ST1)
  - Up to 500 MB/sec throughput
  - 4.5 ₵ per GB/month
- Cold HDD (SC1)
  - Up to 250 MB/sec throughput
  - 2.5 ₵ per GB/month
- Magnetic
  - Up to 800 MB/sec throughput
  - 5 ₵ per GB/month

ELASTIC FILE SYSTEM (EFS)

- Network file system (based on NFSv4 protocol)
- Shared file system for EC2 instances
- Enables mounting (sharing) the same disk "volume" for R/W access across multiple instances at the same time
- Different performance and limitations vs. EBS/Instance store

- Implementation uses abstracted EC2 instances
- ~ 30 ₵ per GB/month storage – ***default burstable throughput***
- **Throughput modes:**
- Can modify modes only once every 24 hours

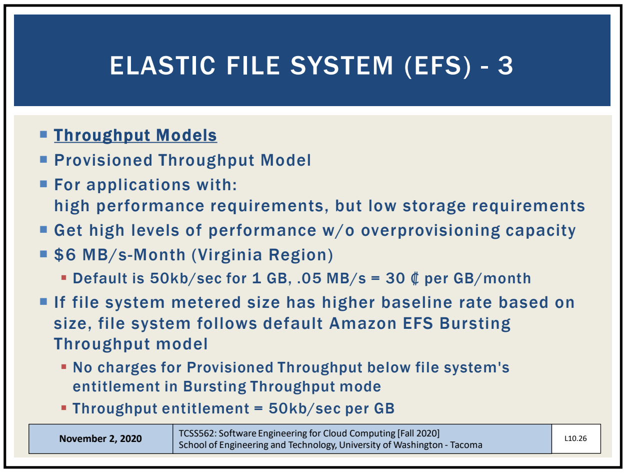
- **Burstable Throughput Model:**
  - Baseline – 50kb/sec per GB
  - Burst – 100MB/sec pet GB (for volumes sized 10GB to 1024 GB)
  - Credits - .72 minutes/day per GB



ELASTIC FILE SYSTEM (EFS) - 2

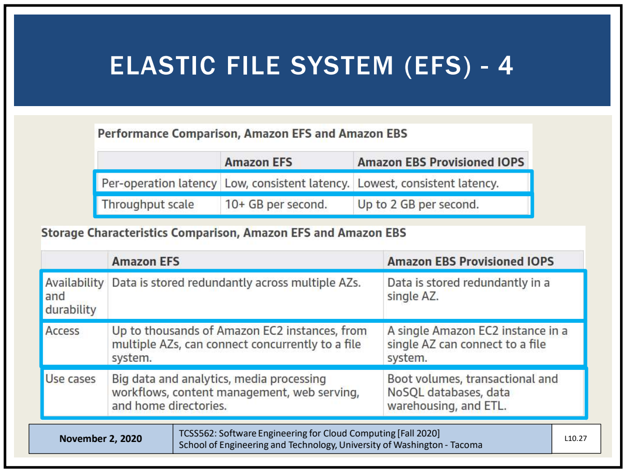
- Burstable Throughput Rates
  - Throughput rates: baseline vs burst
  - Credit model for bursting: maximum burst per day

File System Size (GiB)	Baseline Aggregate Throughput (MiB/s)	Burst Aggregate Throughput (MiB/s)	Maximum Burst Duration (Min/Day)	% of Time File System Can Burst (Per Day)
10	0.5	100	7.2	0.5%
256	12.5	100	180	12.5%
512	25.0	100	360	25.0%
1024	50.0	100	720	50.0%
1536	75.0	150	720	50.0%
2048	100.0	200	720	50.0%
3072	150.0	300	720	50.0%
4096	200.0	400	720	50.0%



ELASTIC FILE SYSTEM (EFS) - 3

- **Throughput Models**
- Provisioned Throughput Model
- For applications with:
  high performance requirements, but low storage requirements
- Get high levels of performance w/o overprovisioning capacity
- $6 MB/s-Month (Virginia Region)
  - Default is 50kb/sec for 1 GB, .05 MB/s = 30 ₵ per GB/month
- If file system metered size has higher baseline rate based on size, file system follows default Amazon EFS Bursting Throughput model
  - No charges for Provisioned Throughput below file system's entitlement in Bursting Throughput mode
  - Throughput entitlement = 50kb/sec per GB



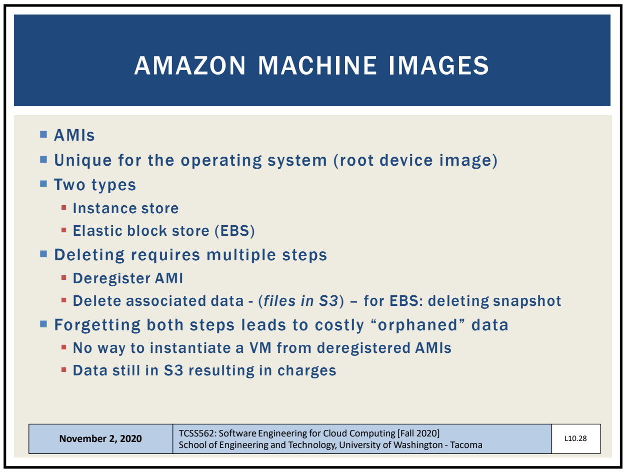
ELASTIC FILE SYSTEM (EFS) - 4

Performance Comparison, Amazon EFS and Amazon EBS

	Amazon EFS	Amazon EBS Provisioned IOPS
Per-operation latency	Low, consistent latency.	Lowest, consistent latency.
Throughput scale	10+ GB per second.	Up to 2 GB per second.

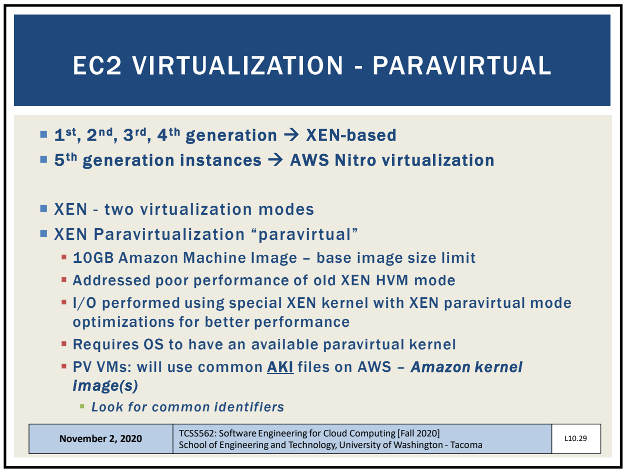
Storage Characteristics Comparison, Amazon EFS and Amazon EBS

	Amazon EFS	Amazon EBS Provisioned IOPS
Availability and durability	Data is stored redundantly across multiple AZs.	Data is stored redundantly in a single AZ.
Access	Up to thousands of Amazon EC2 instances, from multiple AZs, can connect concurrently to a file system.	A single Amazon EC2 instance in a single AZ can connect to a file system.
Use cases	Big data and analytics, media processing workflows, content management, web serving, and home directories.	Boot volumes, transactional and NoSQL databases, data warehousing, and ETL.



AMAZON MACHINE IMAGES

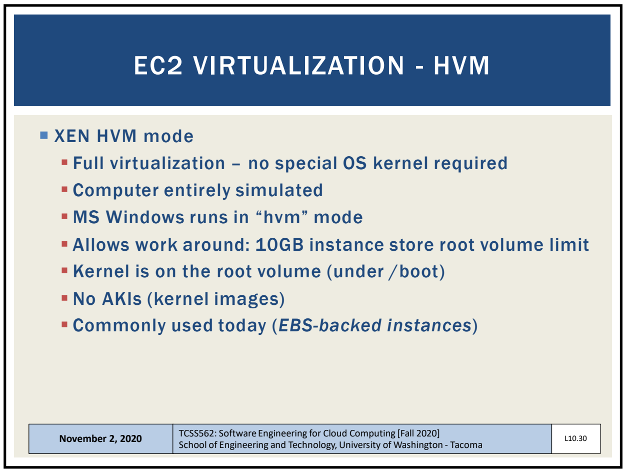
- AMIs
- Unique for the operating system (root device image)
- Two types
  - Instance store
  - Elastic block store (EBS)
- Deleting requires multiple steps
  - Deregister AMI
  - Delete associated data - (*files in S3*) – for EBS: deleting snapshot
- Forgetting both steps leads to costly "orphaned" data
  - No way to instantiate a VM from deregistered AMIs
  - Data still in S3 resulting in charges



EC2 VIRTUALIZATION - PARAVIRTUAL

- **1st, 2nd, 3rd, 4th generation → XEN-based**
- **5th generation instances → AWS Nitro virtualization**

- XEN - two virtualization modes
- XEN Paravirtualization "paravirtual"
  - 10GB Amazon Machine Image – base image size limit
  - Addressed poor performance of old XEN HVM mode
  - I/O performed using special XEN kernel with XEN paravirtual mode optimizations for better performance
  - Requires OS to have an available paravirtual kernel
  - PV VMs: will use common **AKI** files on AWS – ***Amazon kernel image(s)***
    - *Look for common identifiers*



EC2 VIRTUALIZATION - HVM

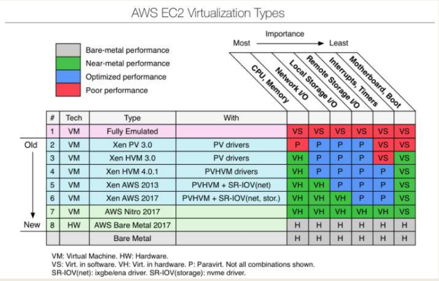
- XEN HVM mode
  - Full virtualization – no special OS kernel required
  - Computer entirely simulated
  - MS Windows runs in "hvm" mode
  - Allows work around: 10GB instance store root volume limit
  - Kernel is on the root volume (under /boot)
  - No AKIs (kernel images)
  - Commonly used today (*EBS-backed instances*)

EC2 VIRTUALIZATION - NITRO

- Nitro based on Kernel-based-virtual-machines
  - Stripped down version of Linux KVM hypervisor
  - Uses KVM core kernel module
  - I/O access has a direct path to the device
- Goal: provide indistinguishable performance from bare metal

EVOLUTION OF AWS VIRTUALIZATION

- From: http://www.brendangregg.com/blog/2017-11-29/aws-ec2-virtualization-2017.html



AWS EC2 Virtualization Types

⬜ Bare-metal performance
🟩 Near-metal performance
🟦 Optimized performance
🟥 Poor performance

Importance: Most → Least

	#	Tech	Type	With	CPU, Memory	Network I/O	Local Storage I/O	Remote Storage I/O	Interrupts, Timers	Motherboard, Boot
	1	VM	Fully Emulated		VS	VS	VS	VS	VS	VS
Old	2	VM	Xen PV 3.0	PV drivers	P	P	P	P	VS	VS
	3	VM	Xen HVM 3.0	PV drivers	VH	P	P	P	VS	VS
	4	VM	Xen HVM 4.0.1	PVHVM drivers	VH	P	P	P	P	VS
	5	VM	Xen AWS 2013	PVHVM + SR-IOV(net)	VH	VH	P	P	P	VS
	6	VM	Xen AWS 2017	PVHVM + SR-IOV(net, stor.)	VH	VH	VH	P	P	VS
	7	VM	AWS Nitro 2017		VH	VH	VH	VH	VH	VS
New	8	HW	AWS Bare Metal 2017		H	H	H	H	H	H
			Bare Metal		H	H	H	H	H	H

VM: Virtual Machine. HW: Hardware.
VS: Virt. in software. VH: Virt. in hardware. P: Paravirt. Not all combinations shown.
SR-IOV(net): ixgbe/ena driver. SR-IOV(storage): nvme driver.

INSTANCE ACTIONS

- Stop
  - Costs of “pausing” an instance
- Terminate
- Reboot

- Image management
- Creating an image
  - EBS (snapshot)
- Bundle image
  - Instance-store

EC2 INSTANCE: NETWORK ACCESS

- Public IP address
- Elastic IPs
  - Costs: in-use FREE, not in-use ~12 ₵/day
  - Not in-use (e.g. “paused” EBS-backed instances)

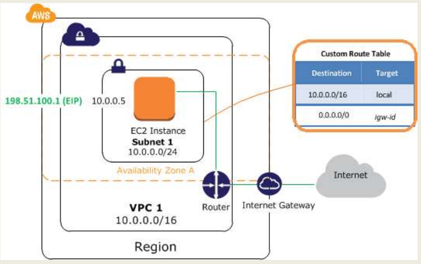
- Security groups
  - E.g. firewall

- Identity access management (IAM)
  - AWS accounts, groups

- VPC / Subnet / Internet Gateway / Router
- NAT-Gateway

SIMPLE VPC

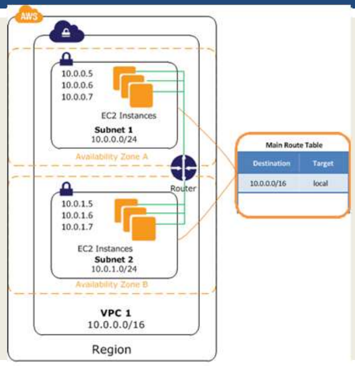
- Recommended when using Amazon EC2



Custom Route Table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	igw-id

VPC SPANNING AVAILABILITY ZONES



Main Route Table

Destination	Target
10.0.0.0/16	local

INSPECTING INSTANCE INFORMATION

- EC2 VMs run a local metadata service
- Can query instance metadata to self discover cloud configuration attributes

- Find your instance ID:

```
curl http://169.254.169.254/
curl http://169.254.169.254/latest/
curl http://169.254.169.254/latest/meta-data/
curl http://169.254.169.254/latest/meta-data/instance-id
; echo
```

- `ec2-get-info` command
- Python API that provides easy/formatted access to metadata

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.

AWS CLI

- Launch Ubuntu 16.04 VM
  - Instances | Launch Instance

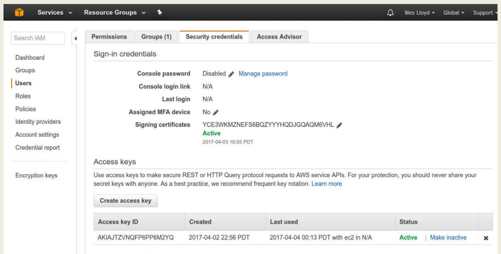
- Install the general AWS CLI
  - sudo apt install awscli

- Create config file

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

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

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

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 . . .

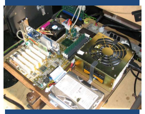
SCRIPT: CREATE A NEW INSTANCE STORE IMAGE FROM LIVE DISK VOLUME

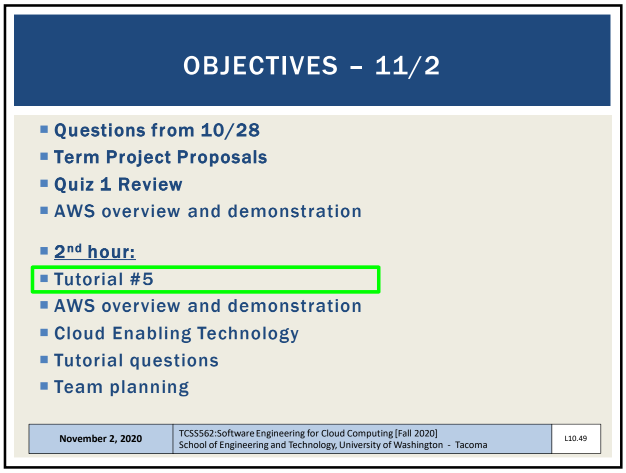
```
image=$1
echo "Burn image $image"
echo "$image" > image.id
mkdir /mnt/tmp
AWS_KEY_DIR=/home/ubuntu/.aws
export EC2_URL=http://ec2.amazonaws.com
export S3_URL=https://s3.amazonaws.com
export EC2_PRIVATE_KEY=${AWS_KEY_DIR}/mykey.pk
export EC2_CERT=${AWS_KEY_DIR}/signing.cert
export AWS_USER_ID={your 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
```

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

WE WILL RETURN AT ~7:10PM

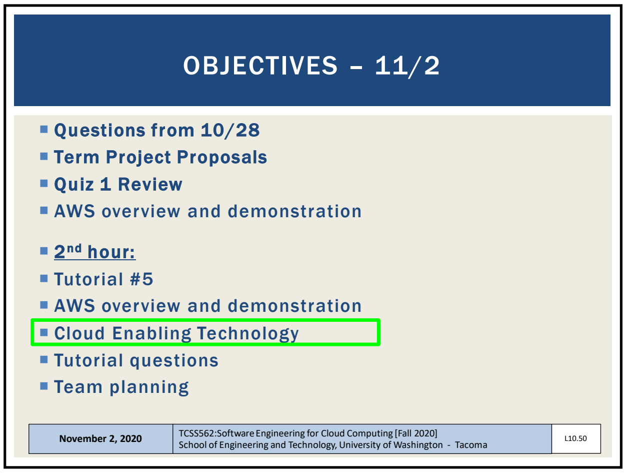




OBJECTIVES – 11/2

- **Questions from 10/28**
- **Term Project Proposals**
- **Quiz 1 Review**
- AWS overview and demonstration

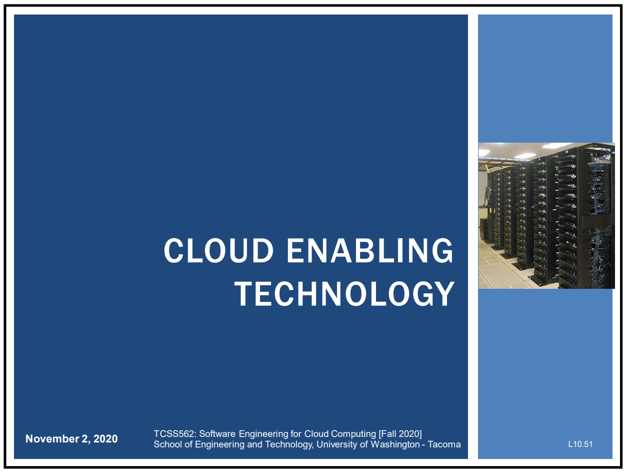
- **2nd hour:**
- Tutorial #5
- AWS overview and demonstration
- Cloud Enabling Technology
- Tutorial questions
- Team planning



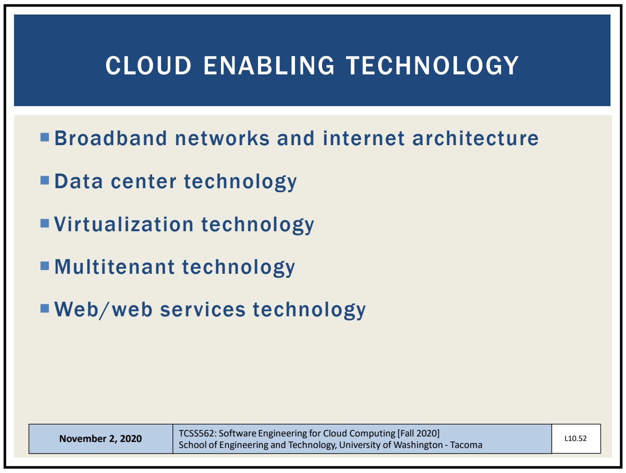
OBJECTIVES – 11/2

- **Questions from 10/28**
- **Term Project Proposals**
- **Quiz 1 Review**
- AWS overview and demonstration

- **2nd hour:**
- Tutorial #5
- AWS overview and demonstration
- Cloud Enabling Technology
- Tutorial questions
- Team planning

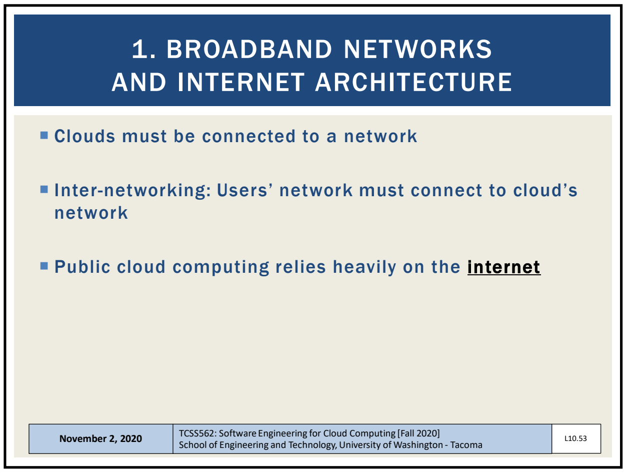


CLOUD ENABLING TECHNOLOGY



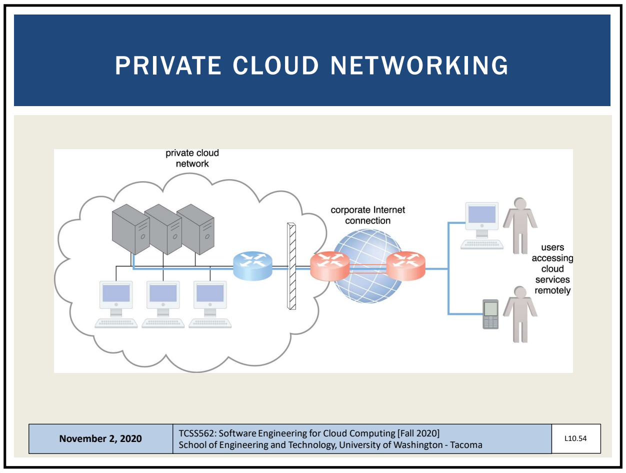
CLOUD ENABLING TECHNOLOGY

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



1. BROADBAND NETWORKS AND INTERNET ARCHITECTURE

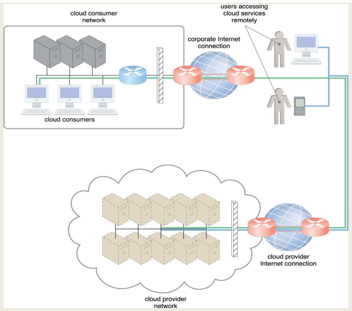
- Clouds must be connected to a network
- Inter-networking: Users' network must connect to cloud's network
- Public cloud computing relies heavily on the **internet**



PRIVATE CLOUD NETWORKING

private cloud network
corporate Internet connection
users accessing cloud services remotely

PUBLIC CLOUD NETWORKING


cloud consumer network
users accessing cloud services remotely
corporate Internet connection
cloud consumers
cloud provider Internet connection
cloud provider network

INTERNETWORKING KEY POINTS

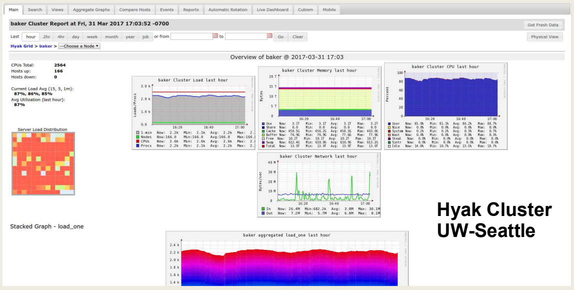
- Cloud consumers and providers typically communicate via the internet
- Decentralized provisioning and management model is not controlled by the cloud consumers or providers
- Inter-networking (internet) relies on connectionless packet switching and route-based interconnectivity
- Routers and switches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

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: ease server provisioning, configuration, patching, monitoring without supervision… ***tools are desirable***

CLUSTER MANAGEMENT TOOLS


Main
Search
Views
Aggregate Graphs
Compare Hosts
Events
Reports
Automatic Rotation
Live Dashboard
Cubism
Mobile
Get Fresh Data
Last hour 2hr 4hr day week month year job or from to Go Clear
Physical View
Hyak Grid > baker > --Choose a Node
Server Load Distribution
baker Cluster Load last hour
baker Cluster Memory last hour
baker Cluster CPU last hour
baker Cluster Network last hour
baker aggregated load_one last hour

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

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

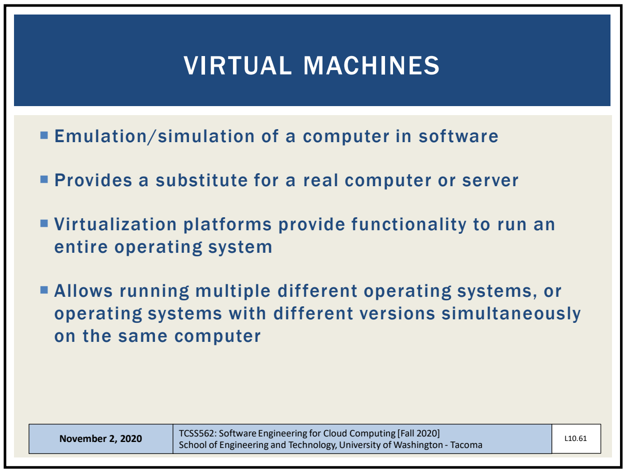
Hyak Cluster
UW-Seattle

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 hardware
- **Secure design**: physical and logical access control
- **Servers**: rackmount, etc.
- **Storage**: hard disk arrays (RAID), storage area network (SAN): disk array with dedicated network, network attached storage (NAS): disk array on network for NFS, etc.
- **Network hardware**: backbone routers (WAN to LAN connectivity), firewalls, VPN gateways, managed switches/routers

3. VIRTUALIZATION TECHNOLOGY

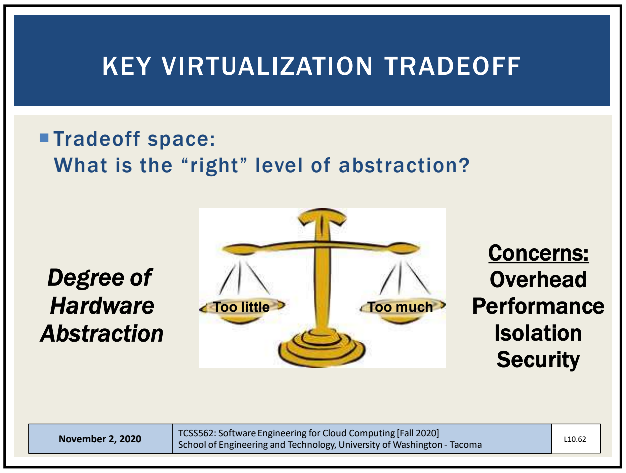
- Convert a physical IT resource into a virtual IT resource
- Servers, storage, network, power (virtual UPSs)
- Virtualization supports:
  - Hardware independence
  - Server consolidation
  - Resource replication
  - Resource pooling
  - Elastic scalability
- Virtual servers
  - Operating-system based virtualization
  - Hardware-based virtualization



VIRTUAL MACHINES

- Emulation/simulation of a computer in software
- Provides a substitute for a real computer or server
- Virtualization platforms provide functionality to run an entire operating system
- Allows running multiple different operating systems, or operating systems with different versions simultaneously on the same computer

November 2, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L10.61



KEY VIRTUALIZATION TRADEOFF

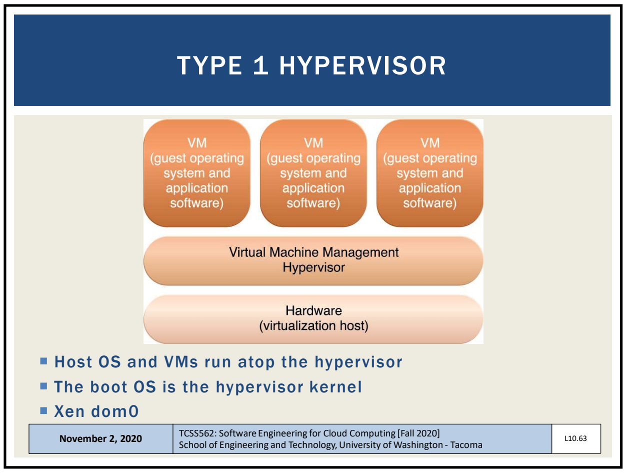
- Tradeoff space:
  What is the “right” level of abstraction?

Degree of Hardware Abstraction

Too little | Too much

<u>Concerns:</u>
Overhead
Performance
Isolation
Security

November 2, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L10.62



TYPE 1 HYPERVISOR

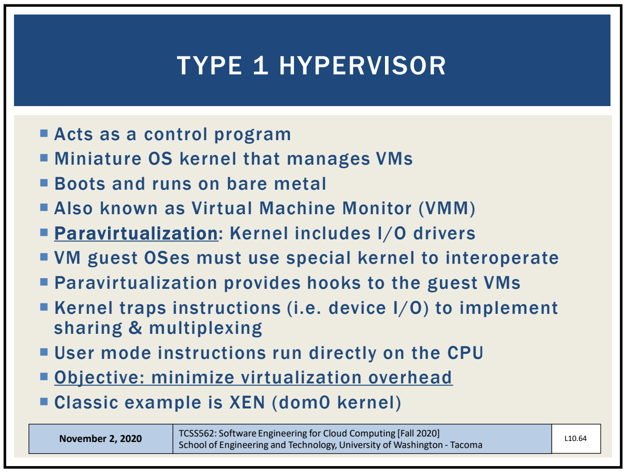
VM (guest operating system and application software) | VM (guest operating system and application software) | VM (guest operating system and application software)

Virtual Machine Management
Hypervisor

Hardware
(virtualization host)

- Host OS and VMs run atop the hypervisor
- The boot OS is the hypervisor kernel
- Xen dom0

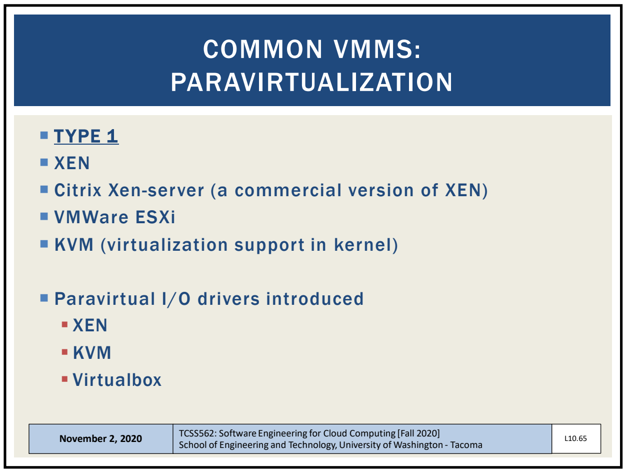
November 2, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L10.63



TYPE 1 HYPERVISOR

- Acts as a control program
- Miniature OS kernel that manages VMs
- Boots and runs on bare metal
- Also known as Virtual Machine Monitor (VMM)
- **<u>Paravirtualization</u>**: Kernel includes I/O drivers
- VM guest OSes must use special kernel to interoperate
- Paravirtualization provides hooks to the guest VMs
- Kernel traps instructions (i.e. device I/O) to implement sharing & multiplexing
- User mode instructions run directly on the CPU
- <u>Objective: minimize virtualization overhead</u>
- Classic example is XEN (dom0 kernel)

November 2, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L10.64

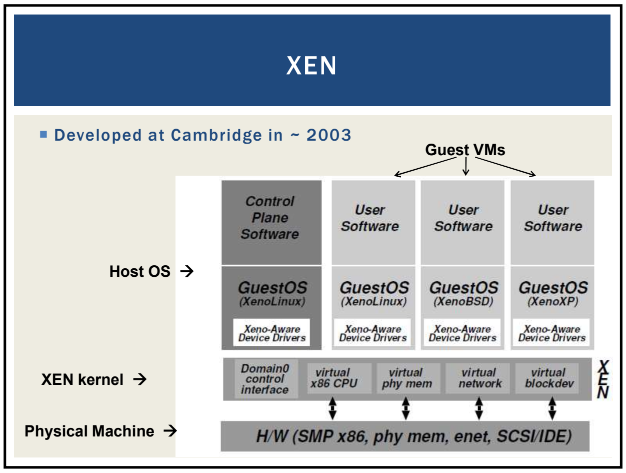


COMMON VMMS: PARAVIRTUALIZATION

- **<u>TYPE 1</u>**
- XEN
- Citrix Xen-server (a commercial version of XEN)
- VMWare ESXi
- KVM (virtualization support in kernel)

- Paravirtual I/O drivers introduced
  - XEN
  - KVM
  - Virtualbox

November 2, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L10.65



XEN

- Developed at Cambridge in ~ 2003

Guest VMs

Host OS →

Control Plane Software | User Software | User Software | User Software

GuestOS (XenoLinux) | GuestOS (XenoLinux) | GuestOS (XenoBSD) | GuestOS (XenoXP)

Xeno-Aware Device Drivers | Xeno-Aware Device Drivers | Xeno-Aware Device Drivers | Xeno-Aware Device Drivers

XEN kernel →

Domain0 control interface | virtual x86 CPU | virtual phy mem | virtual network | virtual blockdev

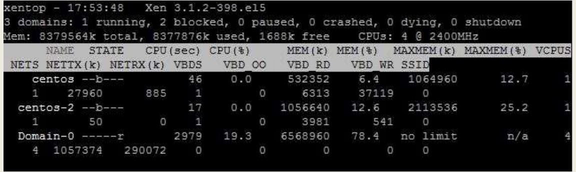
XEN

Physical Machine →

H/W (SMP x86, phy mem, enet, SCSI/IDE)

XEN - 2

- VMs managed as “domains”
- Domain 0 is the hypervisor domain
  - Host OS is installed to run on bare-metal, but doesn’t directly facilitate virtualization (*unlike KVM*)
- Domains 1..n are guests (VMs) – not bare-metal



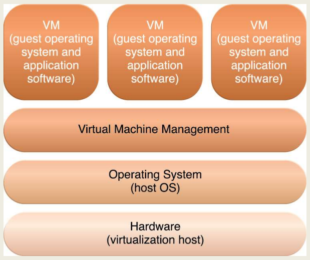
```
xentop - 17:53:48   Xen 3.1.2-398.el5
3 domains: 1 running, 2 blocked, 0 paused, 0 crashed, 0 dying, 0 shutdown
Mem: 8379564k total, 8377876k used, 1688k free    CPUs: 4 @ 2400MHz
      NAME  STATE   CPU(sec) CPU(%)     MEM(k) MEM(%)  MAXMEM(k) MAXMEM(%) VCPUS
 NETS NETTX(k) NETRX(k) VBDS   VBD_OO   VBD_RD   VBD_WR SSID
    centos --b---         46    0.0     532352    6.4    1064960      12.7     1
    1    27960      885    1        0     6313    37119    0
  centos-2 --b---         17    0.0    1056640   12.6    2113536      25.2     1
    1       50        0    1        0     3981      541    0
  Domain-0 -----r       2979   19.3    6568960   78.4   no limit       n/a     4
    4  1057374   290072    0        0        0        0    0
```

XEN - 3

- Physical machine boots special XEN kernel
- Kernel provides paravirtual API to manage CPU & device multiplexing
- Guests require modified XEN-aware kernels
- Xen supports full-virtualization for unmodified OS guests in hvm mode
- Amazon EC2 largely based on modified version of XEN hypervisor (EC2 gens 1-4)
- XEN provides its own CPU schedulers, I/O scheduling

TYPE 2 HYPERVISOR

- Adds additional layer



VM (guest operating system and application software)

VM (guest operating system and application software)

VM (guest operating system and application software)

Virtual Machine Management

Operating System (host OS)

Hardware (virtualization host)

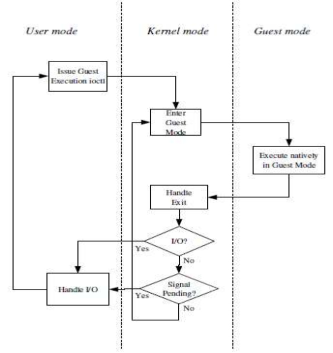
TYPE 2 HYPERVISOR

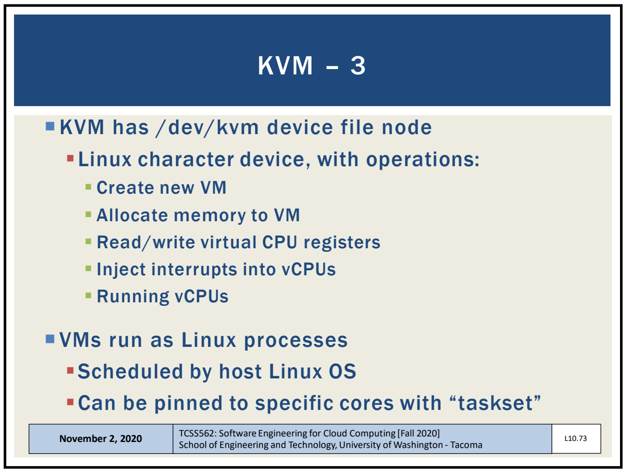
- Problem: Original x86 CPUs could not trap special instructions
- Instructions not specially marked
- Solution: Use Full Virtualization
- Trap ALL instructions
- “Fully” simulate entire computer
- Tradeoff: Higher Overhead
- Benefit: Can virtualize any operating system without modification

KERNEL BASED VIRTUAL MACHINES (KVM)

- x86 HW notoriously difficult to virtualize
- Extensions added to 64-bit Intel/AMD CPUs
  - Provides hardware assisted virtualization
  - New “guest” operating mode
  - Hardware state switch
  - Exit reason reporting
  - Intel/AMD implementations different
    - Linux uses vendor specific kernel modules

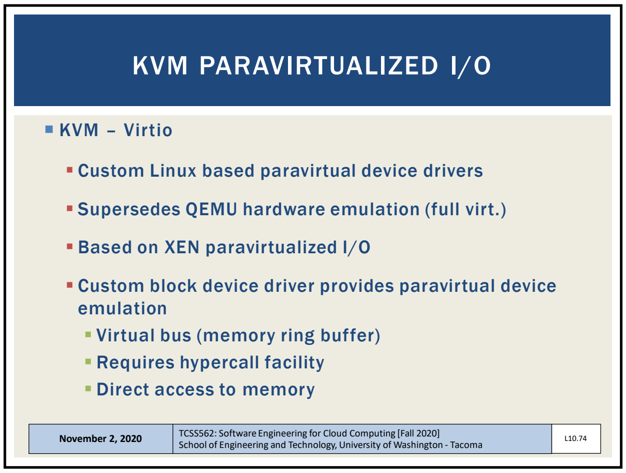
KVM – 2





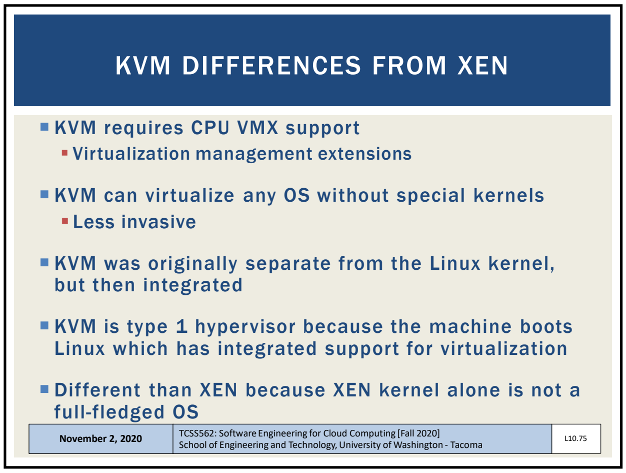
KVM – 3

- KVM has /dev/kvm device file node
  - Linux character device, with operations:
    - Create new VM
    - Allocate memory to VM
    - Read/write virtual CPU registers
    - Inject interrupts into vCPUs
    - Running vCPUs
- VMs run as Linux processes
  - Scheduled by host Linux OS
  - Can be pinned to specific cores with “taskset”



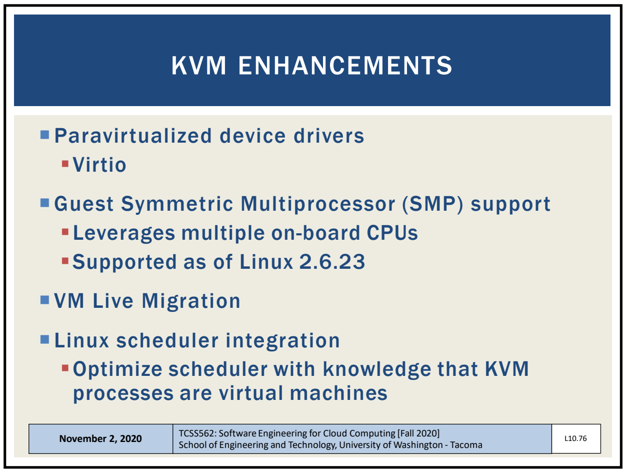
KVM PARAVIRTUALIZED I/O

- KVM – Virtio
  - Custom Linux based paravirtual device drivers
  - Supersedes QEMU hardware emulation (full virt.)
  - Based on XEN paravirtualized I/O
  - Custom block device driver provides paravirtual device emulation
    - Virtual bus (memory ring buffer)
    - Requires hypercall facility
    - Direct access to memory



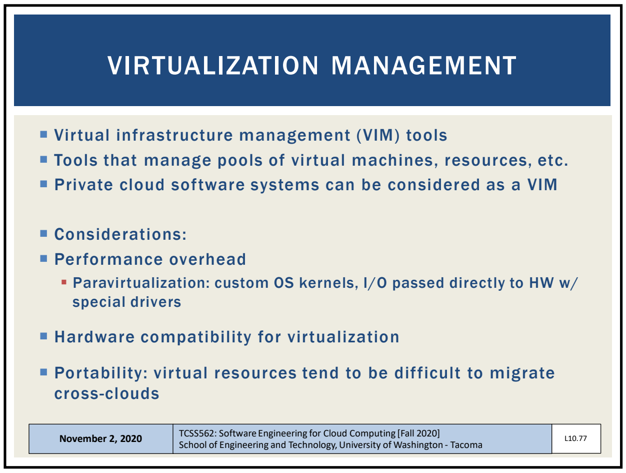
KVM DIFFERENCES FROM XEN

- KVM requires CPU VMX support
  - Virtualization management extensions
- KVM can virtualize any OS without special kernels
  - Less invasive
- KVM was originally separate from the Linux kernel, but then integrated
- KVM is type 1 hypervisor because the machine boots Linux which has integrated support for virtualization
- Different than XEN because XEN kernel alone is not a full-fledged OS



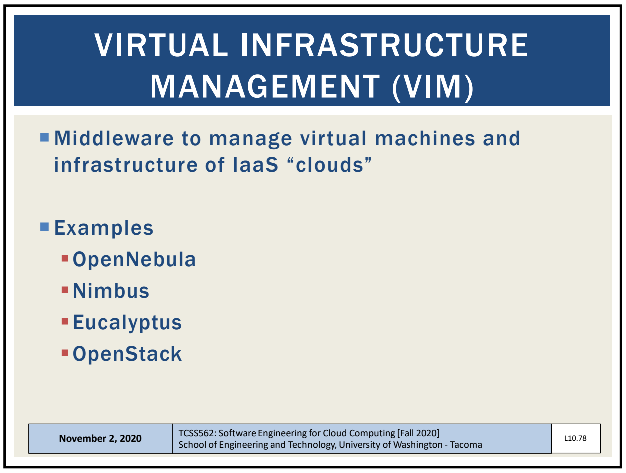
KVM ENHANCEMENTS

- Paravirtualized device drivers
  - Virtio
- Guest Symmetric Multiprocessor (SMP) support
  - Leverages multiple on-board CPUs
  - Supported as of Linux 2.6.23
- VM Live Migration
- Linux scheduler integration
  - Optimize scheduler with knowledge that KVM processes are virtual machines



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



VIRTUAL INFRASTRUCTURE MANAGEMENT (VIM)

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

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

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

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)

4. MULTITENANT APPLICATIONS

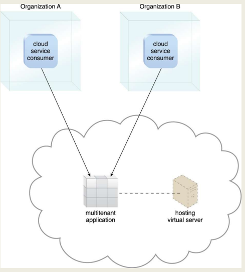
- Each tenant (like in an apartment) has their own view of the application
- Tenants are unaware of their neighbors
- Tenants can only access their data, no access to data and configuration that is not their own

- 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)



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, …

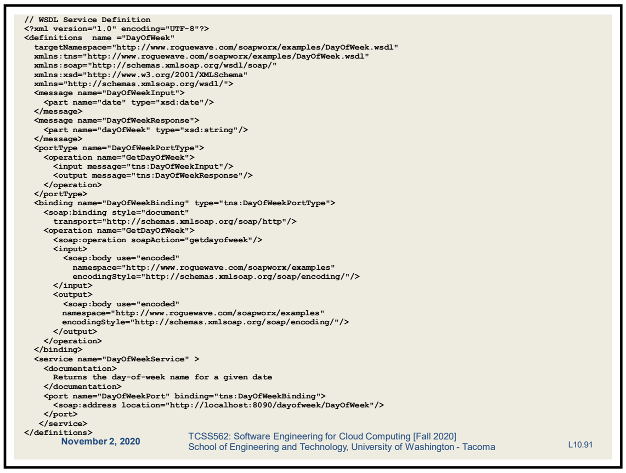
```
// SOAP REQUEST

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>
```

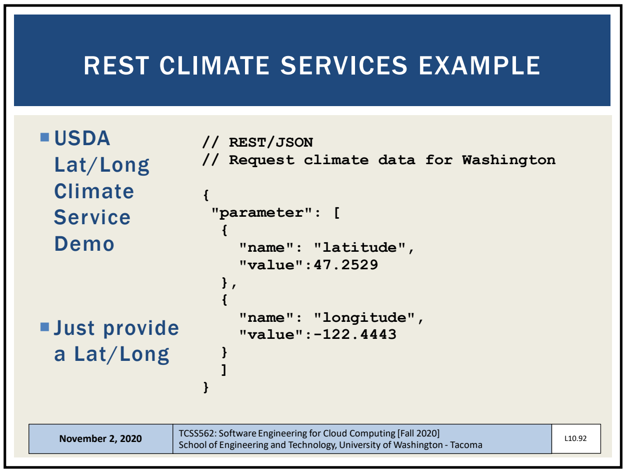
```
// SOAP RESPONSE
POST /InStock HTTP/1.1
Host: www.bookshop.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?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>
```



```
// 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>
```

November 2, 2020
TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma
L10.91



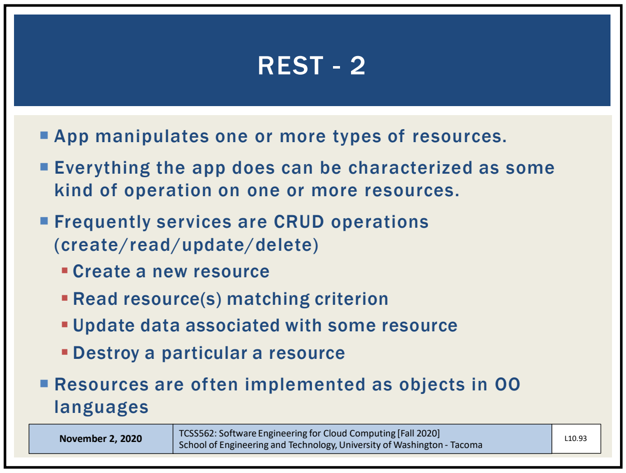
REST CLIMATE SERVICES EXAMPLE

- 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
  }
  ]
}
```

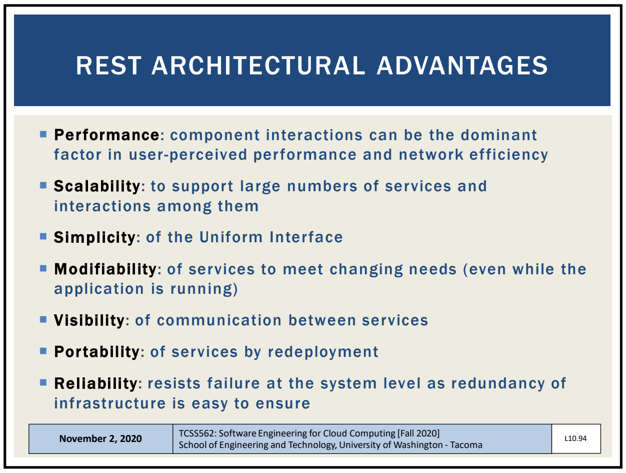
November 2, 2020
TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma
L10.92



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

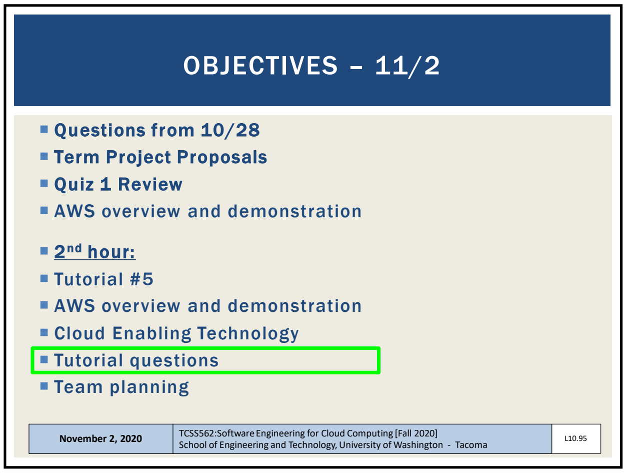
November 2, 2020
TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma
L10.93



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

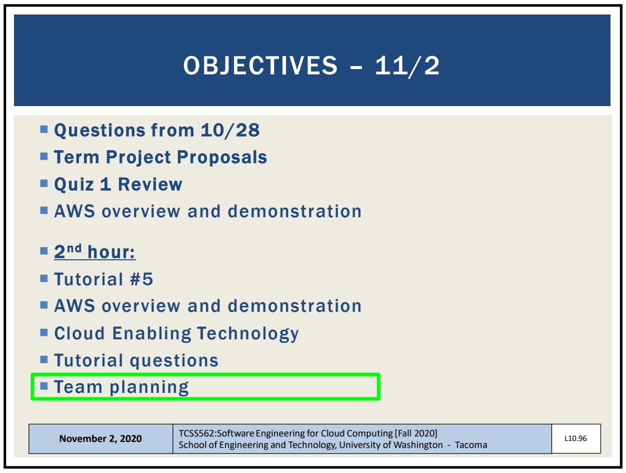
November 2, 2020
TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma
L10.94



OBJECTIVES – 11/2

- **Questions from 10/28**
- **Term Project Proposals**
- **Quiz 1 Review**
- AWS overview and demonstration
- **2nd hour:**
- Tutorial #5
- AWS overview and demonstration
- Cloud Enabling Technology
- Tutorial questions
- Team planning

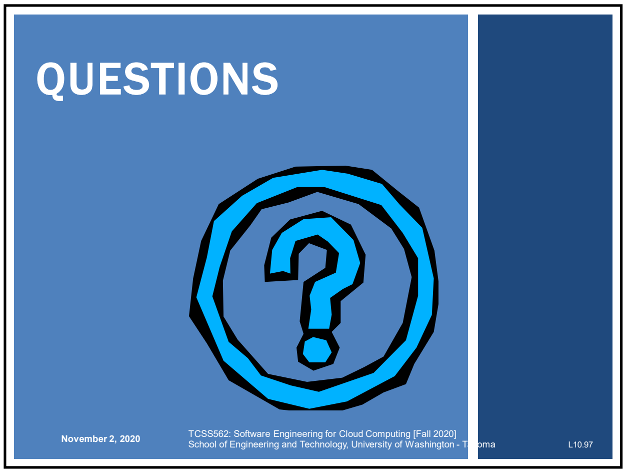
November 2, 2020
TCSS562:Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma
L10.95



OBJECTIVES – 11/2

- **Questions from 10/28**
- **Term Project Proposals**
- **Quiz 1 Review**
- AWS overview and demonstration
- **2nd hour:**
- Tutorial #5
- AWS overview and demonstration
- Cloud Enabling Technology
- Tutorial questions
- Team planning

November 2, 2020
TCSS562:Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma
L10.96

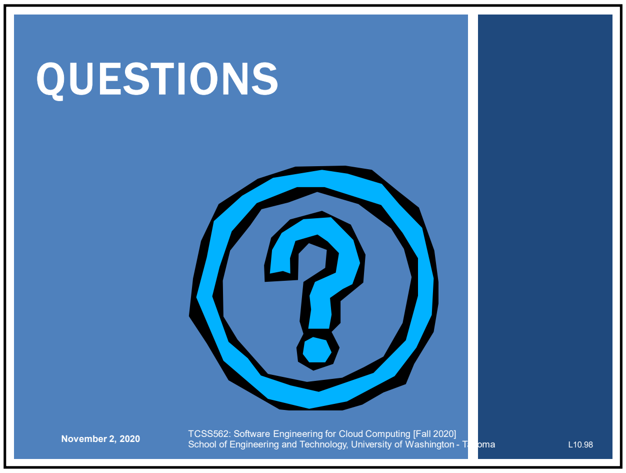


QUESTIONS

November 2, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L10.97



QUESTIONS

November 2, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L10.98