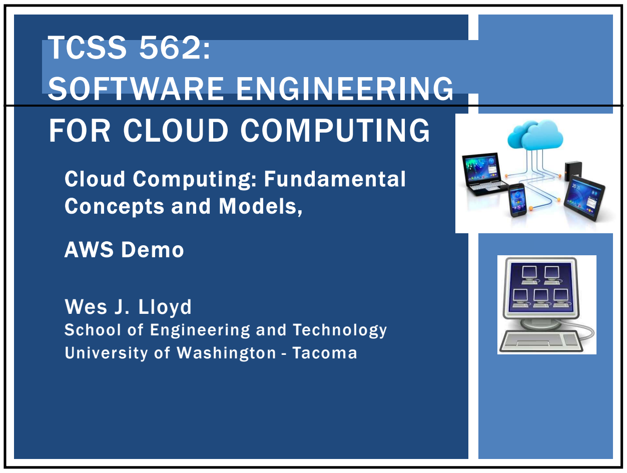


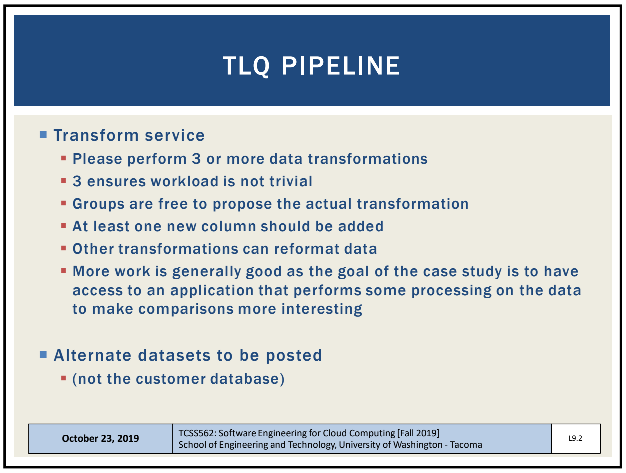
TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

Cloud Computing: Fundamental Concepts and Models,

AWS Demo

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

1

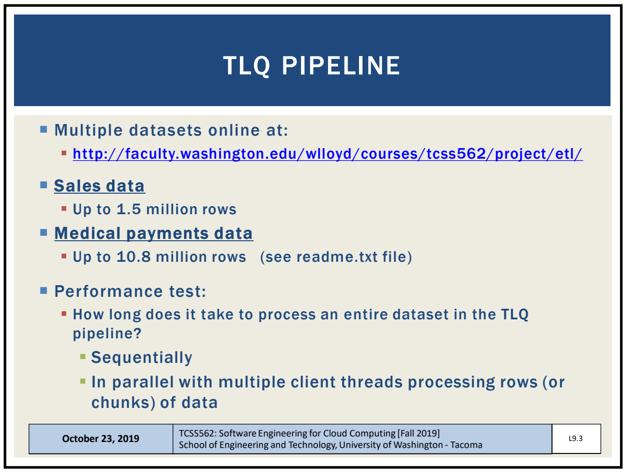


TLQ PIPELINE

- Transform service
  - Please perform 3 or more data transformations
  - 3 ensures workload is not trivial
  - Groups are free to propose the actual transformation
  - At least one new column should be added
  - Other transformations can reformat data
  - More work is generally good as the goal of the case study is to have access to an application that performs some processing on the data to make comparisons more interesting
- Alternate datasets to be posted
  - (not the customer database)

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.2

2

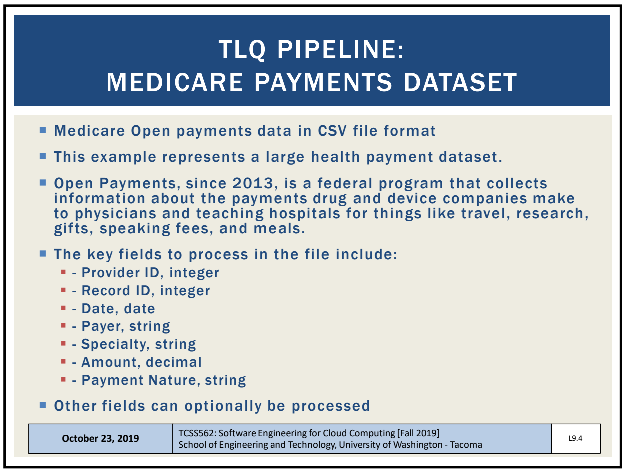


TLQ PIPELINE

- Multiple datasets online at:
  - http://faculty.washington.edu/wlloyd/courses/tcss562/project/etl/
- **Sales data**
  - Up to 1.5 million rows
- **Medical payments data**
  - Up to 10.8 million rows (see readme.txt file)
- Performance test:
  - How long does it take to process an entire dataset in the TLQ pipeline?
    - Sequentially
    - In parallel with multiple client threads processing rows (or chunks) of data

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.3

3

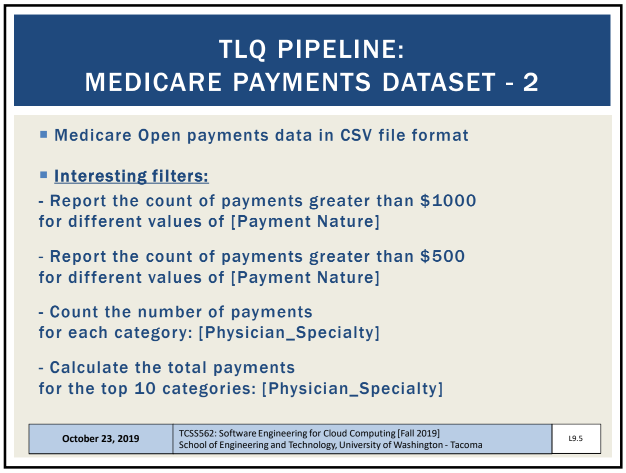


TLQ PIPELINE: MEDICARE PAYMENTS DATASET

- Medicare Open payments data in CSV file format
- This example represents a large health payment dataset.
- Open Payments, since 2013, is a federal program that collects information about the payments drug and device companies make to physicians and teaching hospitals for things like travel, research, gifts, speaking fees, and meals.
- The key fields to process in the file include:
  - - Provider ID, integer
  - - Record ID, integer
  - - Date, date
  - - Payer, string
  - - Specialty, string
  - - Amount, decimal
  - - Payment Nature, string
- Other fields can optionally be processed

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.4

4



TLQ PIPELINE: MEDICARE PAYMENTS DATASET - 2

- Medicare Open payments data in CSV file format
- **Interesting filters:**

- Report the count of payments greater than $1000 for different values of [Payment Nature]

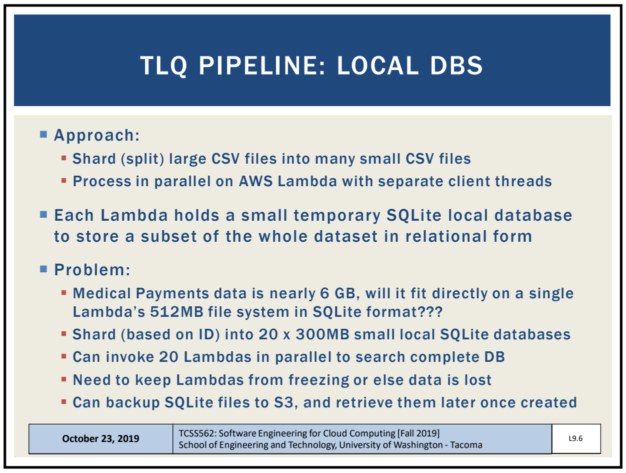
- Report the count of payments greater than $500 for different values of [Payment Nature]

- Count the number of payments for each category: [Physician_Specialty]

- Calculate the total payments for the top 10 categories: [Physician_Specialty]

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.5

5



TLQ PIPELINE: LOCAL DBS

- Approach:
  - Shard (split) large CSV files into many small CSV files
  - Process in parallel on AWS Lambda with separate client threads
- Each Lambda holds a small temporary SQLite local database to store a subset of the whole dataset in relational form
- Problem:
  - Medical Payments data is nearly 6 GB, will it fit directly on a single Lambda's 512MB file system in SQLite format???
  - Shard (based on ID) into 20 x 300MB small local SQLite databases
  - Can invoke 20 Lambdas in parallel to search complete DB
  - Need to keep Lambdas from freezing or else data is lost
  - Can backup SQLite files to S3, and retrieve them later once created

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.6

6

TLQ PIPELINE: CENTRALIZED DB

- Can load data to centralized database
- Amazon Aurora Serverless
  - Provides MySQL (cheaper), and PostgreSQL (more expensive) options
  - Aurora Serverless is an alternative to hosting a DB with an always-on VM - - ***but is it cheaper???***
  - Storage is 10¢/GB/month
  - Size of Aurora instance is scalable
  - Amazon Aurora Serverless charges based on reserved or dynamic "**Aurora Capacity** Units"
  - 1 ACU = 2GB memory, 1 vCPU, with corresponding networking
  - Single database instance becomes a processing bottleneck
  - ***How long will it take to load 10 million rows on a 1 vCPU, 2GB DB??***
  - ***How many parallel clients can this DB support?***

7

FEEDBACK FROM 10/21

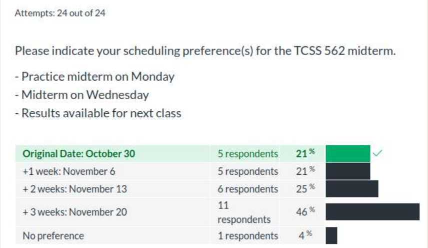
- Perspective on material: 7 (→ *mostly new to me*)
- Pace: 5.149 (~ just right)
- 18 respondents

- **Tutorial 3:**
  What images should be attached to the PDF file?
  - **Looking for output from bonnie++ (html file) or (txt format)**
  - **Can import html into text editor (Google docs or Word) to create PDF**
  - **Can also take a screen capture (*picture*) of the HTML output and include in document as an image**
  - **To produce HTML from CSV:** `bon_csv2html < bonnie.csv`
  - **To produce text from CSV:** `bon_csv2txt < bonnie.csv`

8

MIDTERM EXAM SURVEY RESULTS

- When is the midterm exam?



Attempts: 24 out of 24

Please indicate your scheduling preference(s) for the TCSS 562 midterm.

- Practice midterm on Monday
- Midterm on Wednesday
- Results available for next class

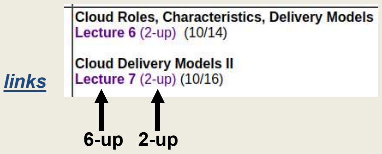
Original Date: October 30	5 respondents	21 %	🟩 ✓
+1 week: November 6	5 respondents	21 %	■
+ 2 weeks: November 13	6 respondents	25 %	■
+ 3 weeks: November 20	11 respondents	46 %	■
No preference	1 respondents	4 %	■

24 respondents, consensus: +2 weeks

9

SLIDES ONLINE

- **Can you please upload PDF slides with 1 slide on 1 page (rather than 4)?**
- Currently there are 2-up and 6-up options
- 1-up would require many many pages
  - Some slide sets nearly ~100 slides = 100 pages

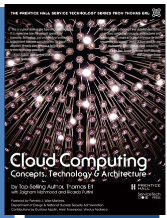


links

Cloud Roles, Characteristics, Delivery Models
Lecture 6 (2-up) (10/14)

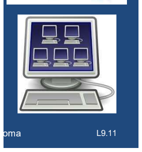
Cloud Delivery Models II
Lecture 7 (2-up) (10/16)

10



THE PRENTICE HALL SERVICE TECHNOLOGY SERIES FROM THOMAS ERL
Cloud Computing
Concepts, Technology & Architecture

CHAPTER 4: FUNDAMENTAL CONCEPTS AND MODELS

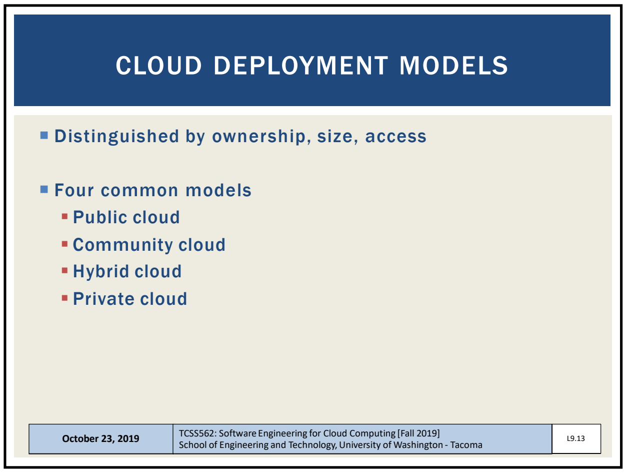


11

OBJECTIVES

- Cloud Computing Concepts and Models
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - **Cloud deployment models**

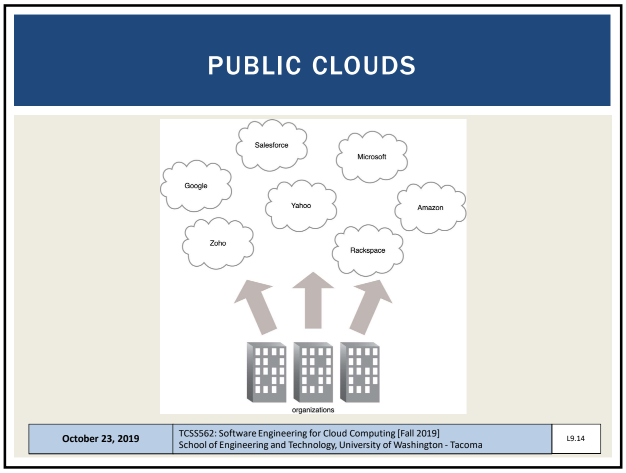
12



CLOUD DEPLOYMENT MODELS

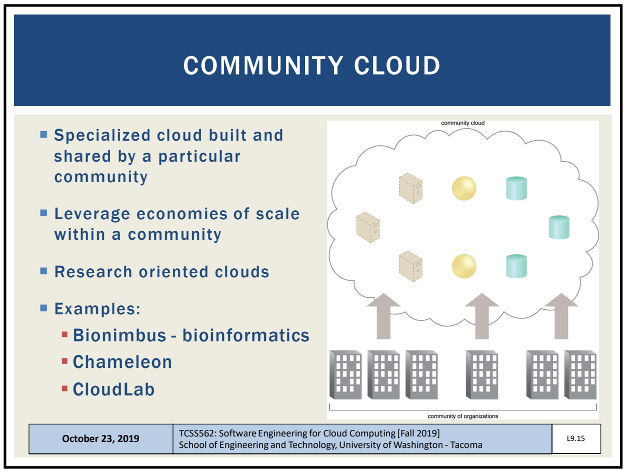
- Distinguished by ownership, size, access
- Four common models
  - Public cloud
  - Community cloud
  - Hybrid cloud
  - Private cloud

13



PUBLIC CLOUDS

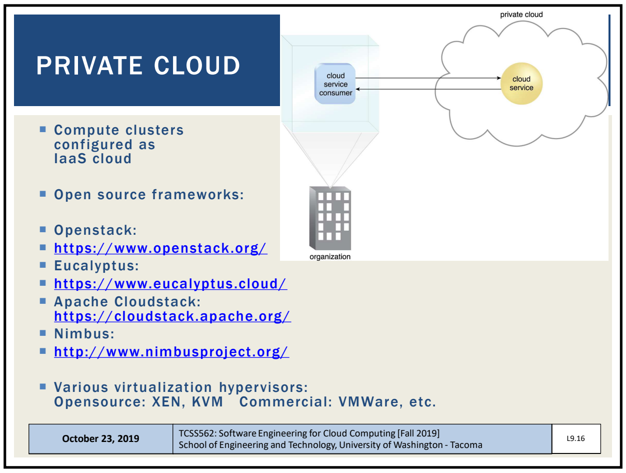
14



COMMUNITY CLOUD

- Specialized cloud built and shared by a particular community
- Leverage economies of scale within a community
- Research oriented clouds
- Examples:
  - Bionimbus - bioinformatics
  - Chameleon
  - CloudLab

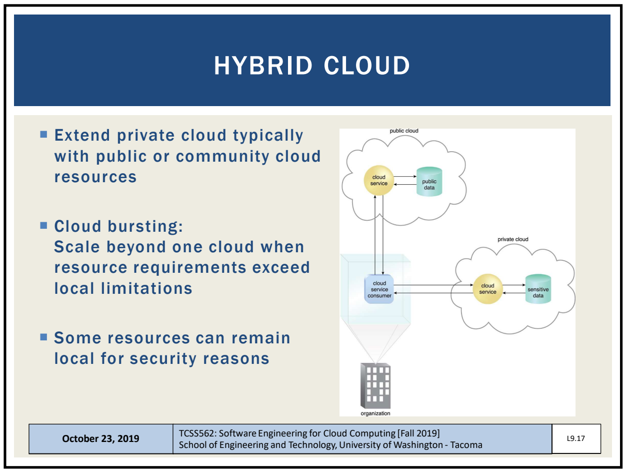
15



PRIVATE CLOUD

- Compute clusters configured as IaaS cloud
- Open source frameworks:
- Openstack:
- https://www.openstack.org/
- Eucalyptus:
- https://www.eucalyptus.cloud/
- Apache Cloudstack: https://cloudstack.apache.org/
- Nimbus:
- http://www.nimbusproject.org/
- Various virtualization hypervisors: Opensource: XEN, KVM Commercial: VMWare, etc.

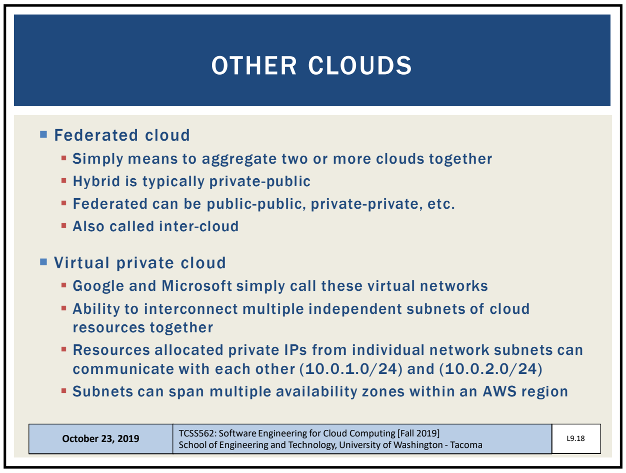
16



HYBRID CLOUD

- Extend private cloud typically with public or community cloud resources
- Cloud bursting: Scale beyond one cloud when resource requirements exceed local limitations
- Some resources can remain local for security reasons

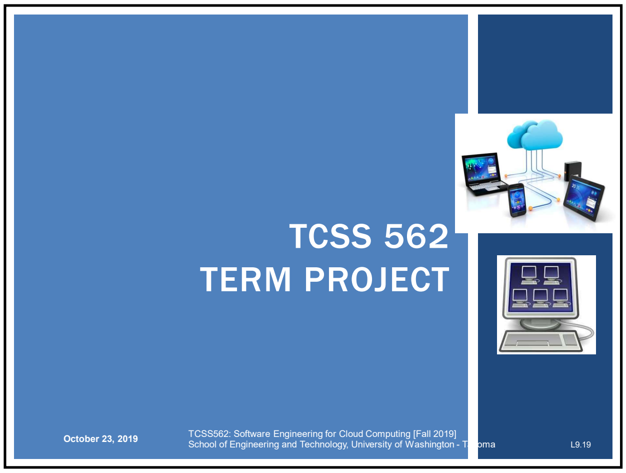
17



OTHER CLOUDS

- Federated cloud
  - Simply means to aggregate two or more clouds together
  - Hybrid is typically private-public
  - Federated can be public-public, private-private, etc.
  - Also called inter-cloud
- Virtual private cloud
  - Google and Microsoft simply call these virtual networks
  - Ability to interconnect multiple independent subnets of cloud resources together
  - Resources allocated private IPs from individual network subnets can communicate with each other (10.0.1.0/24) and (10.0.2.0/24)
  - Subnets can span multiple availability zones within an AWS region

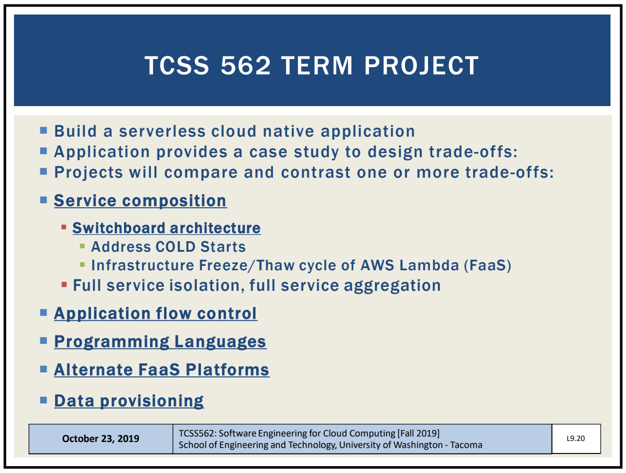
18



TCSS 562 TERM PROJECT

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.19

19

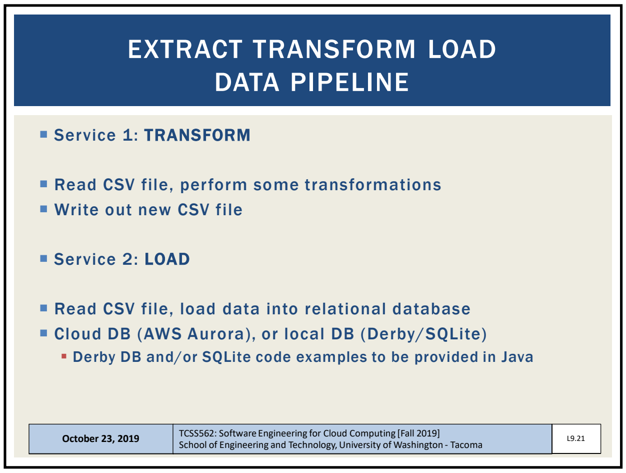


TCSS 562 TERM PROJECT

- Build a serverless cloud native application
- Application provides a case study to design trade-offs:
- Projects will compare and contrast one or more trade-offs:
- Service composition
  - Switchboard architecture
    - Address COLD Starts
    - Infrastructure Freeze/Thaw cycle of AWS Lambda (FaaS)
  - Full service isolation, full service aggregation
- Application flow control
- Programming Languages
- Alternate FaaS Platforms
- Data provisioning

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.20

20

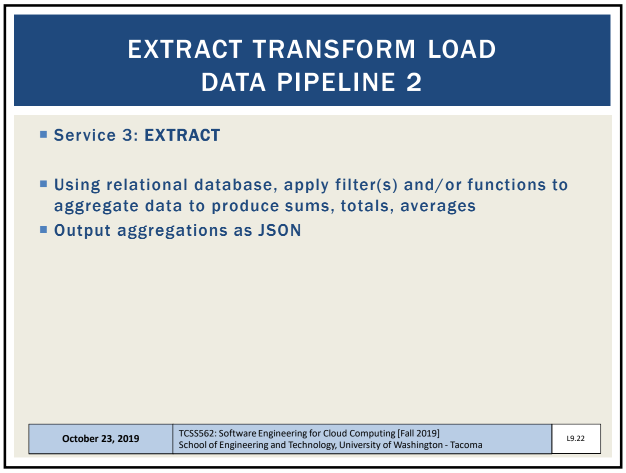


EXTRACT TRANSFORM LOAD DATA PIPELINE

- Service 1: **TRANSFORM**
- Read CSV file, perform some transformations
- Write out new CSV file
- Service 2: **LOAD**
- Read CSV file, load data into relational database
- Cloud DB (AWS Aurora), or local DB (Derby/SQLite)
  - Derby DB and/or SQLite code examples to be provided in Java

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.21

21

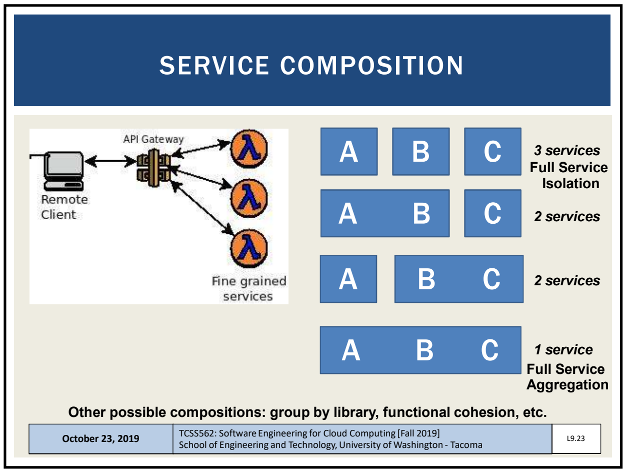


EXTRACT TRANSFORM LOAD DATA PIPELINE 2

- Service 3: **EXTRACT**
- Using relational database, apply filter(s) and/or functions to aggregate data to produce sums, totals, averages
- Output aggregations as JSON

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.22

22

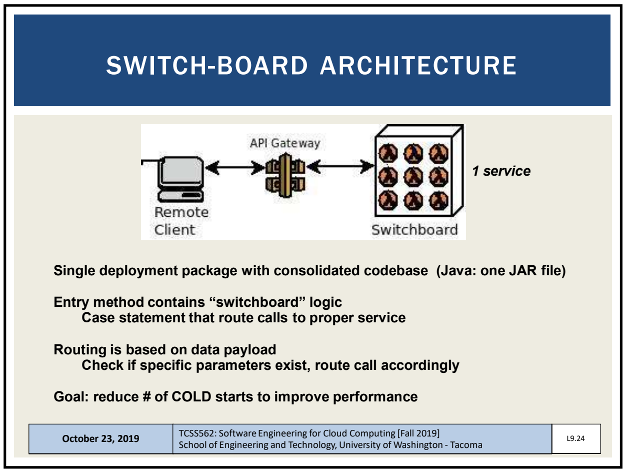


SERVICE COMPOSITION

Other possible compositions: group by library, functional cohesion, etc.

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.23

23



SWITCH-BOARD ARCHITECTURE

Single deployment package with consolidated codebase (Java: one JAR file)

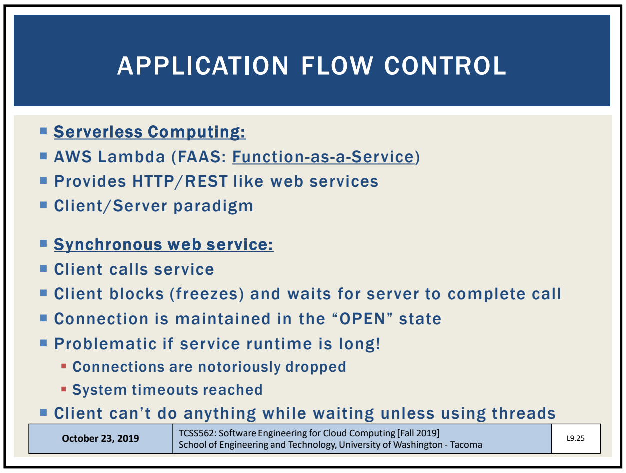
Entry method contains "switchboard" logic
Case statement that route calls to proper service

Routing is based on data payload
Check if specific parameters exist, route call accordingly

Goal: reduce # of COLD starts to improve performance

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.24

24

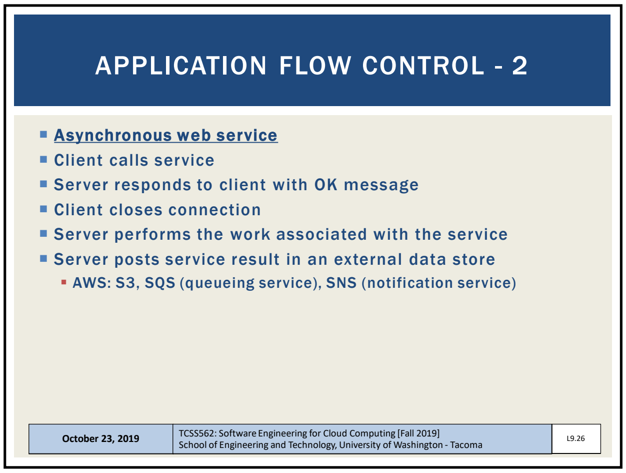


APPLICATION FLOW CONTROL

- **Serverless Computing:**
- AWS Lambda (FAAS: Function-as-a-Service)
- Provides HTTP/REST like web services
- Client/Server paradigm

- **Synchronous web service:**
- Client calls service
- Client blocks (freezes) and waits for server to complete call
- Connection is maintained in the "OPEN" state
- Problematic if service runtime is long!
  - Connections are notoriously dropped
  - System timeouts reached
- Client can't do anything while waiting unless using threads

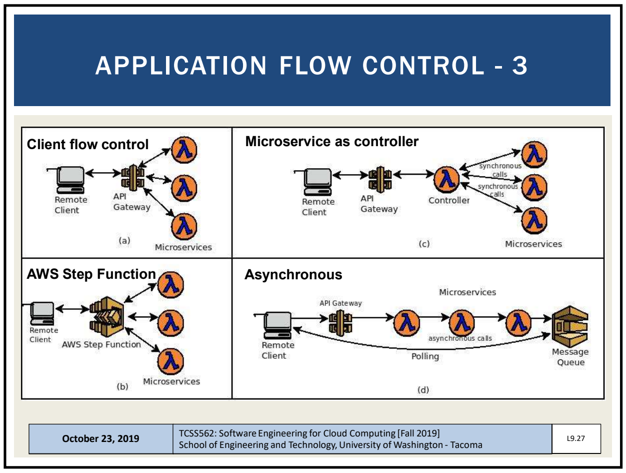
25



APPLICATION FLOW CONTROL - 2

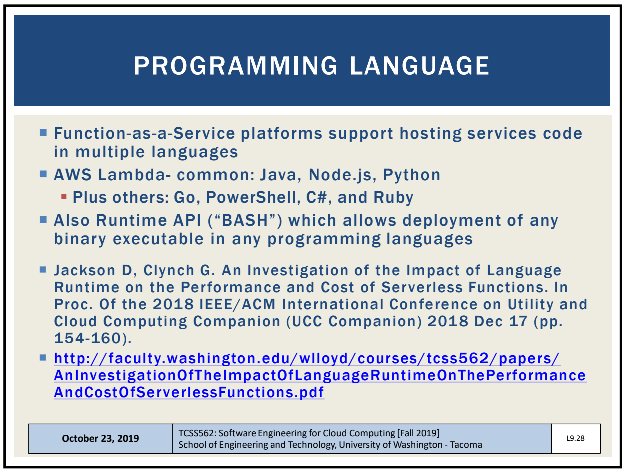
- **Asynchronous web service**
- Client calls service
- Server responds to client with OK message
- Client closes connection
- Server performs the work associated with the service
- Server posts service result in an external data store
  - AWS: S3, SQS (queueing service), SNS (notification service)

26



APPLICATION FLOW CONTROL - 3

27

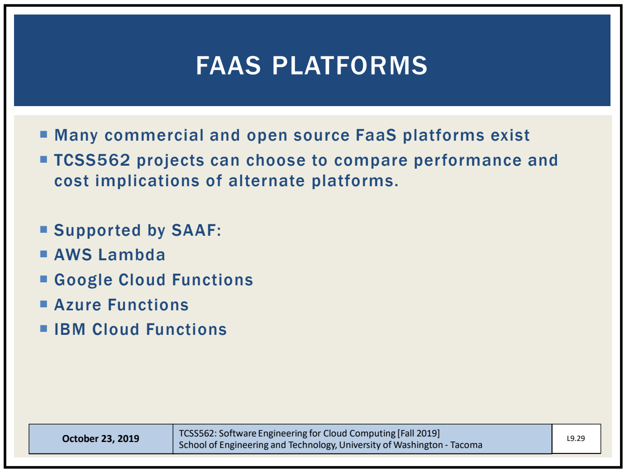


PROGRAMMING LANGUAGE

- Function-as-a-Service platforms support hosting services code in multiple languages
- AWS Lambda- common: Java, Node.js, Python
  - Plus others: Go, PowerShell, C#, and Ruby
- Also Runtime API ("BASH") which allows deployment of any binary executable in any programming languages

- Jackson D, Clynch G. An Investigation of the Impact of Language Runtime on the Performance and Cost of Serverless Functions. In Proc. Of the 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion) 2018 Dec 17 (pp. 154-160).
- http://faculty.washington.edu/wlloyd/courses/tcss562/papers/AnInvestigationOfTheImpactOfLanguageRuntimeOnThePerformanceAndCostOfServerlessFunctions.pdf

28

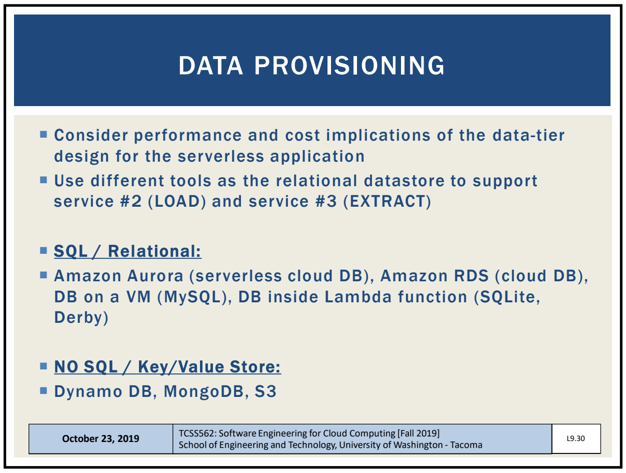


FAAS PLATFORMS

- Many commercial and open source FaaS platforms exist
- TCSS562 projects can choose to compare performance and cost implications of alternate platforms.

- Supported by SAAF:
- AWS Lambda
- Google Cloud Functions
- Azure Functions
- IBM Cloud Functions

29



DATA PROVISIONING

- Consider performance and cost implications of the data-tier design for the serverless application
- Use different tools as the relational datastore to support service #2 (LOAD) and service #3 (EXTRACT)

- **SQL / Relational:**
- Amazon Aurora (serverless cloud DB), Amazon RDS (cloud DB), DB on a VM (MySQL), DB inside Lambda function (SQLite, Derby)

- **NO SQL / Key/Value Store:**
- Dynamo DB, MongoDB, S3

30

AWS DEMO

31

CLOUD 101 WORKSHOP

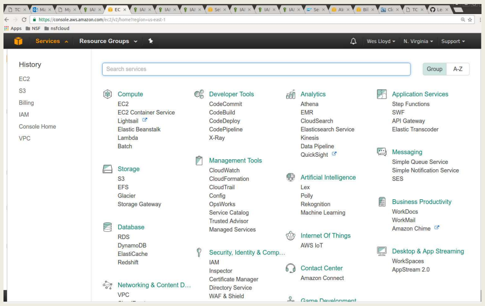
- From the eScience Institute @ UW Seattle:
- https://escience.washington.edu/
- Offers 1-day cloud workshops

- Introduction to AWS, Azure, and Google Cloud
- Task: Deploying a Python DJANGO web application
- Workshop materials available online:

- https://cloudmaven.github.io/documentation/rc_cloud101_immersion.html

32

AWS MANAGEMENT CONSOLE



33

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

34

INSTANCE STORAGE

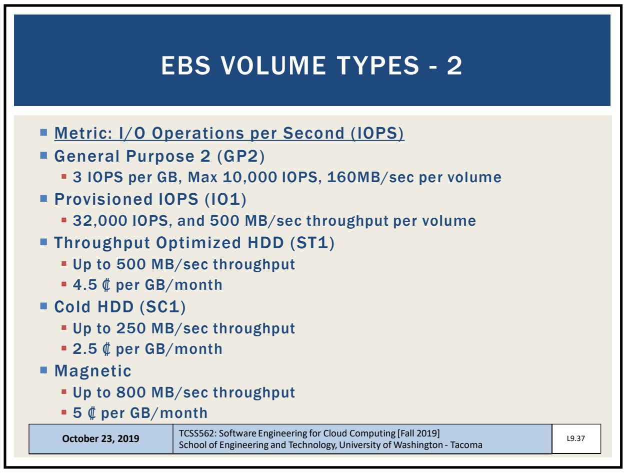
- Also called ephemeral storage
- Persisted using images saved to S3 (simple storage service)
  - ~2.3¢ per GB/month on S3
  - 5GB of free tier storage space on S3
- Requires “burning” an image
- Mutli-step process:
  - Create image files
  - Upload chunks to S3
  - Register image
- Launching a VM
  - 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…***

35

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…

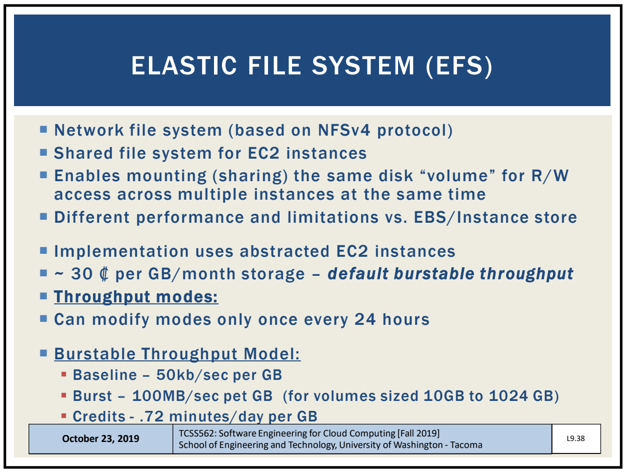
36



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

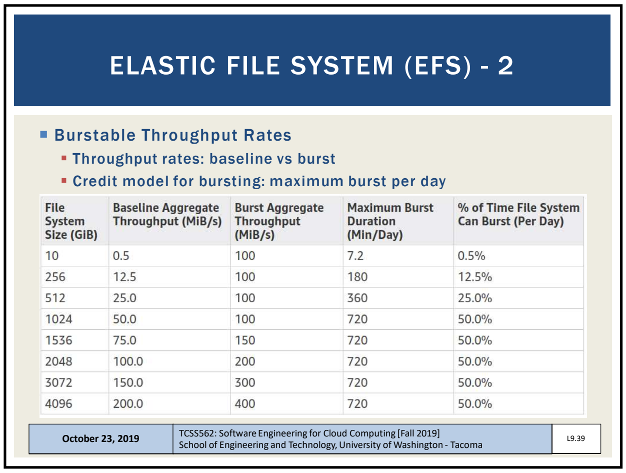
37



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

38

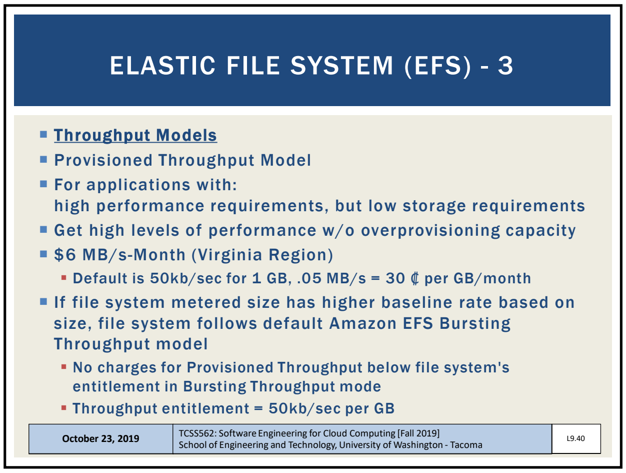


ELASTIC FILE SYSTEM (EFS) - 2

- 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%

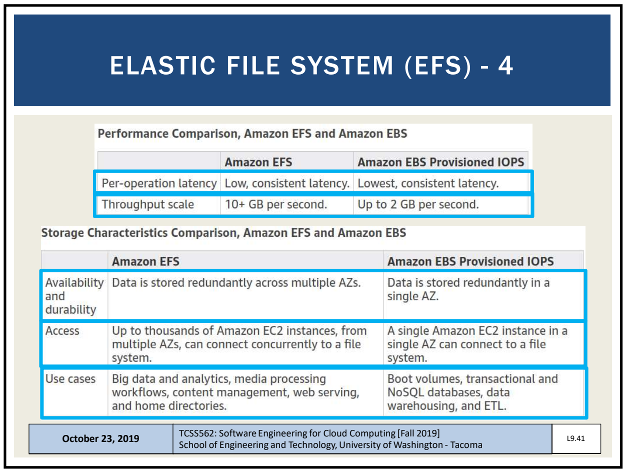
39



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

40



ELASTIC FILE SYSTEM (EFS) - 4

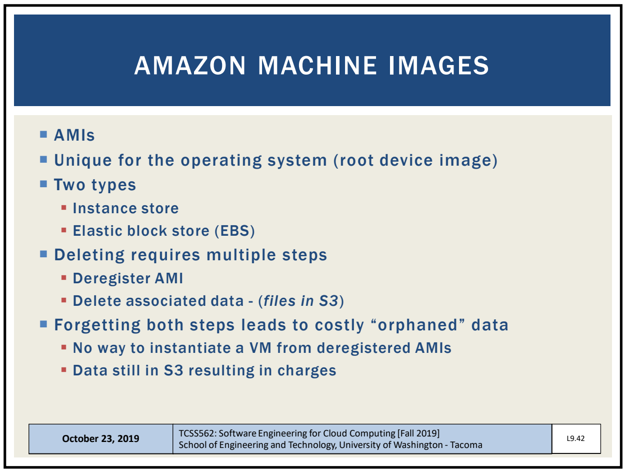
Performance Comparison, Amazon EFS and Amazon EBS

	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.

41



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*)
- Forgetting both steps leads to costly "orphaned" data
  - No way to instantiate a VM from deregistered AMIs
  - Data still in S3 resulting in charges

42

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

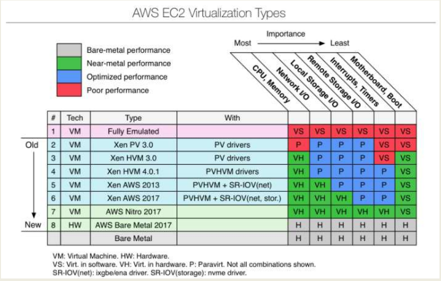
- XEN HVM mode
  - Full virtualization – no special OS kernel required
  - Computer entirely simulated
  - MS Windows runs in "hvm" mode
  - Allows work around: 10GB instance store root volume limit
  - Kernel is on the root volume (under /boot)
  - No AKIs (kernel images)
  - Commonly used today (*EBS-backed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

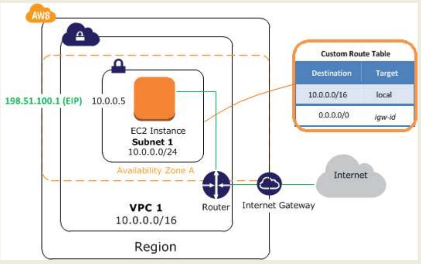
- Security groups
  - E.g. firewall

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

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

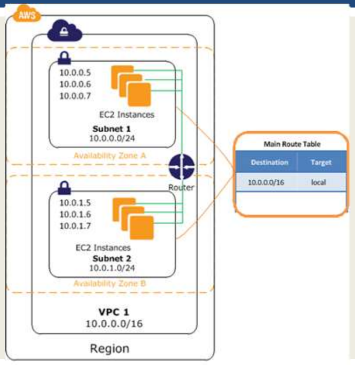
SIMPLE VPC

- Recommended when using Amazon EC2



49

VPC SPANNING AVAILABILITY ZONES



50

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.

51

AWS CLI

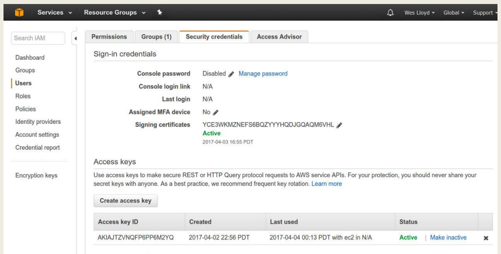
- Launch Ubuntu 16.04 VM
  - Instances | Launch Instance
- Install the general AWS CLI
  - sudo apt install awscli
- Create config file

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

52

AWS CLI - 2

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



Sign-in credentials

Console password Disabled
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.

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

53

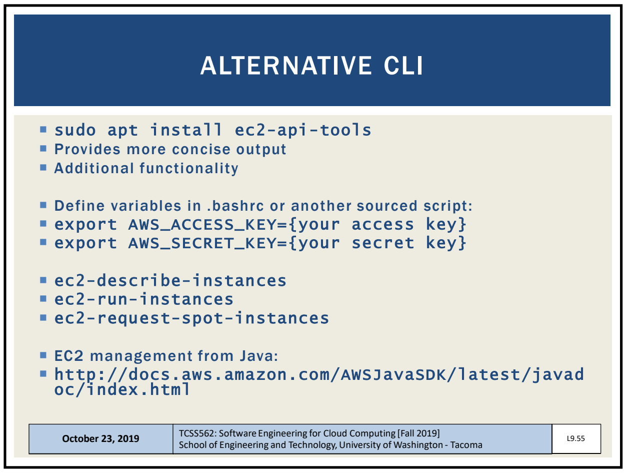
AWS CLI - 3

- 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`

54



ALTERNATIVE CLI

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

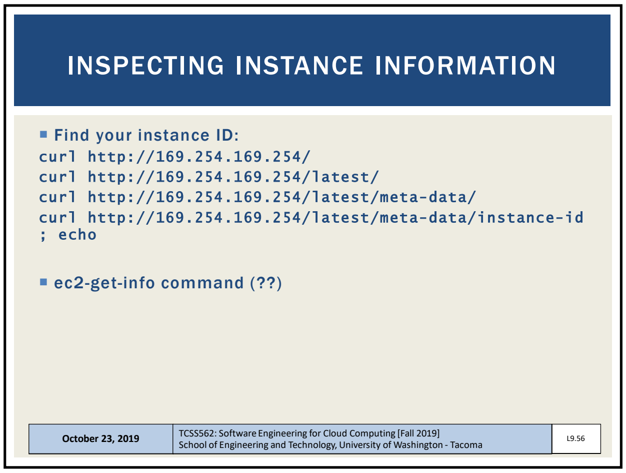
- Define variables in .bashrc or another sourced script:
- `export AWS_ACCESS_KEY={your access key}`
- `export AWS_SECRET_KEY={your secret key}`

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

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

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.55

55



INSPECTING INSTANCE INFORMATION

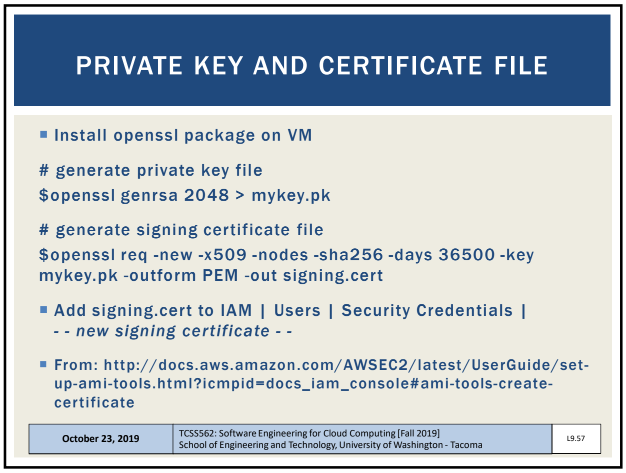
- Find your instance ID:

```
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 (??)

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.56

56



PRIVATE KEY AND CERTIFICATE FILE

- Install openssl package on VM

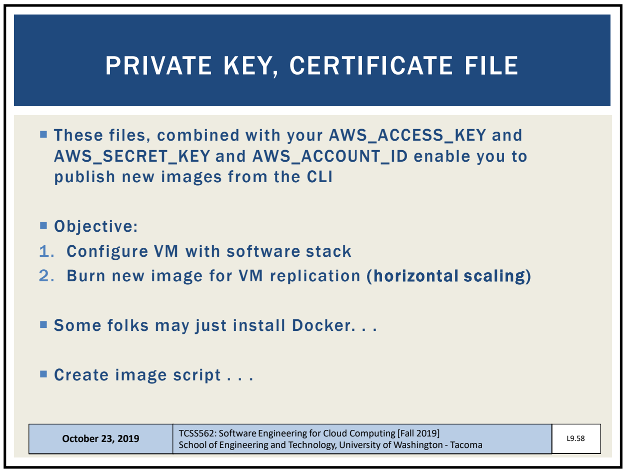
```
# generate private key file
$openssl genrsa 2048 > mykey.pk

# 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

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.57

57



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:

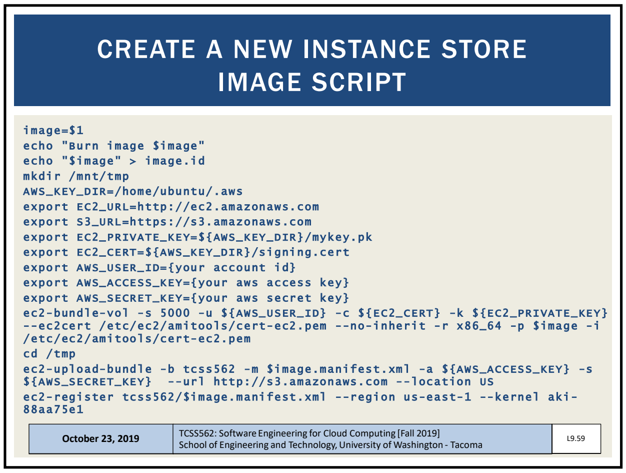
1. Configure VM with software stack
2. Burn new image for VM replication (**horizontal scaling**)

- Some folks may just install Docker. . .

- Create image script . . .

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.58

58

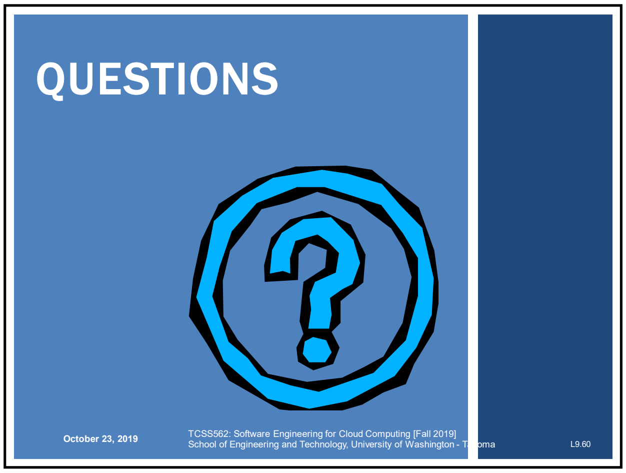


CREATE A NEW INSTANCE STORE IMAGE SCRIPT

```
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
```

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.59

59



QUESTIONS

October 23, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L9.60

60