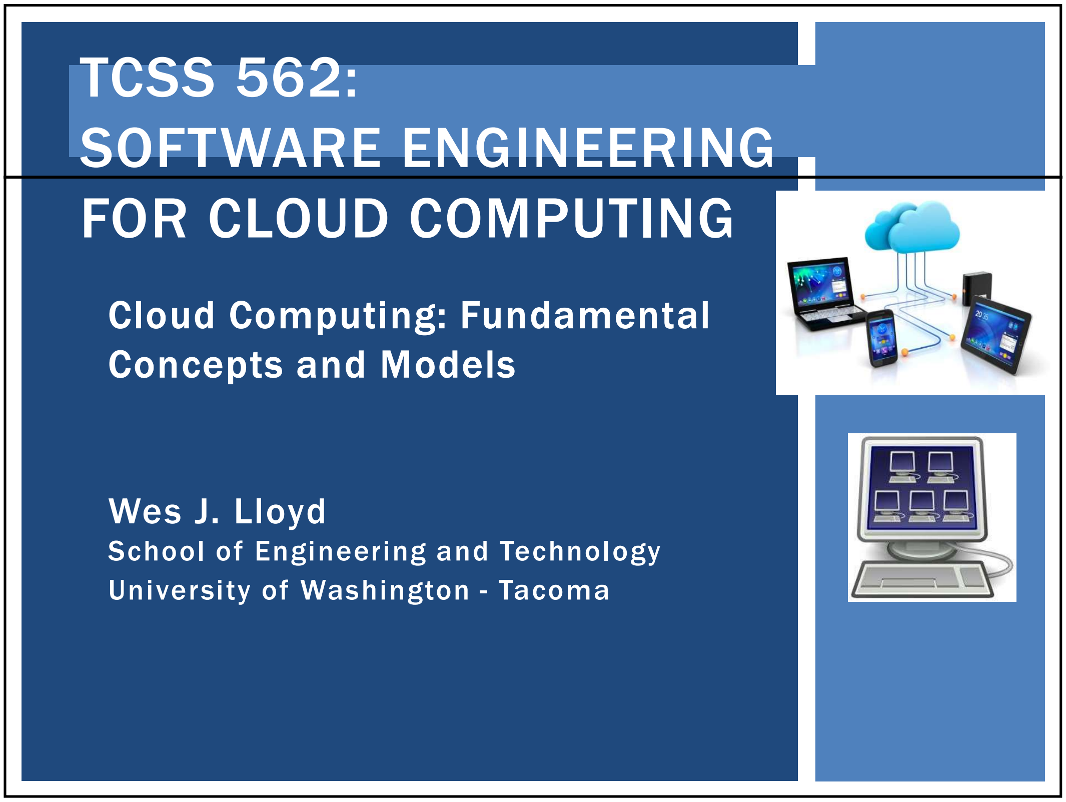
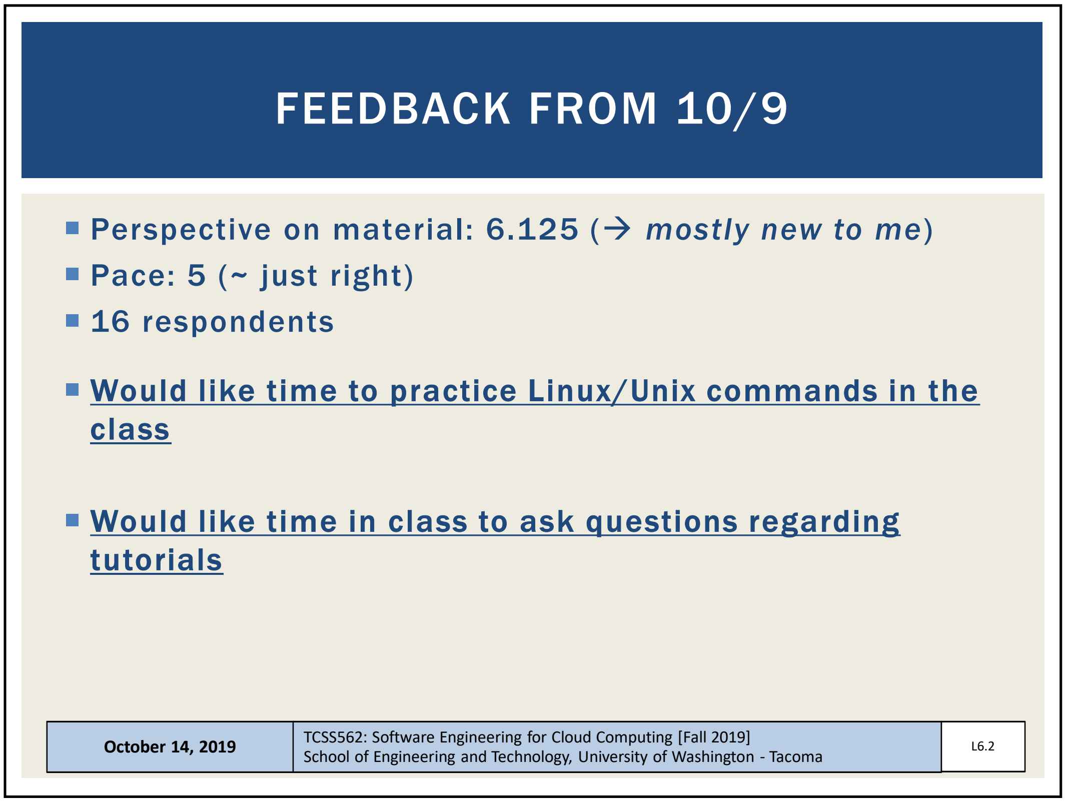


TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

Cloud Computing: Fundamental Concepts and Models

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

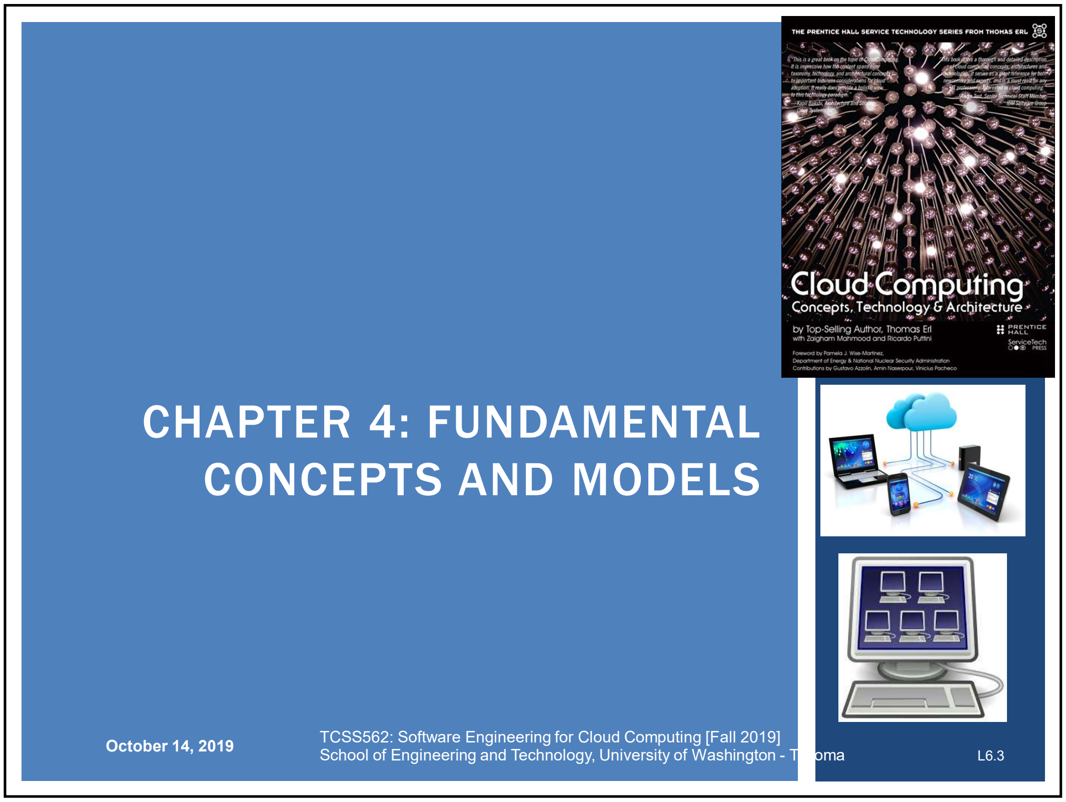
1



FEEDBACK FROM 10/9

- Perspective on material: 6.125 (→ *mostly new to me*)
- Pace: 5 (~ just right)
- 16 respondents
- Would like time to practice Linux/Unix commands in the class
- Would like time in class to ask questions regarding tutorials

2


THE PRENTICE HALL SERVICE TECHNOLOGY SERIES FROM THOMAS ERL
Cloud Computing
Concepts, Technology & Architecture
by Top-Selling Author, Thomas Erl
with Zaigham Mahmood and Ricardo Puttini
PRENTICE HALL
ServiceTech PRESS

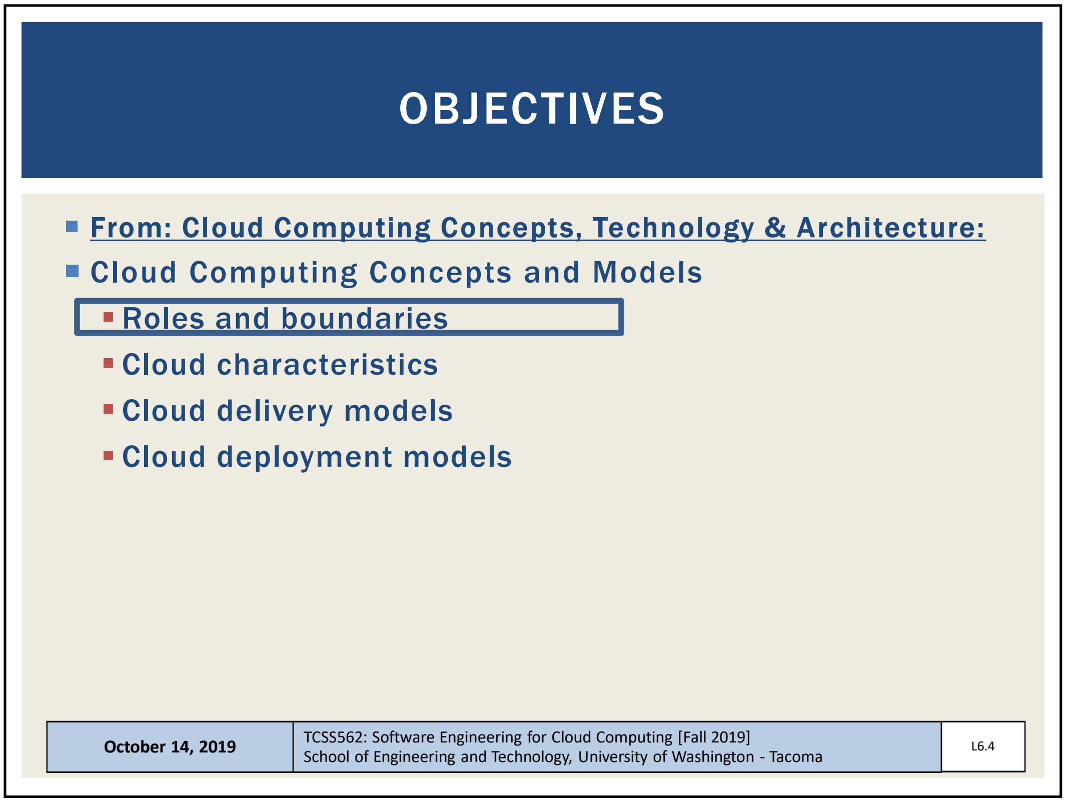
CHAPTER 4: FUNDAMENTAL CONCEPTS AND MODELS

October 14, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - T oma

L6.3

3



OBJECTIVES

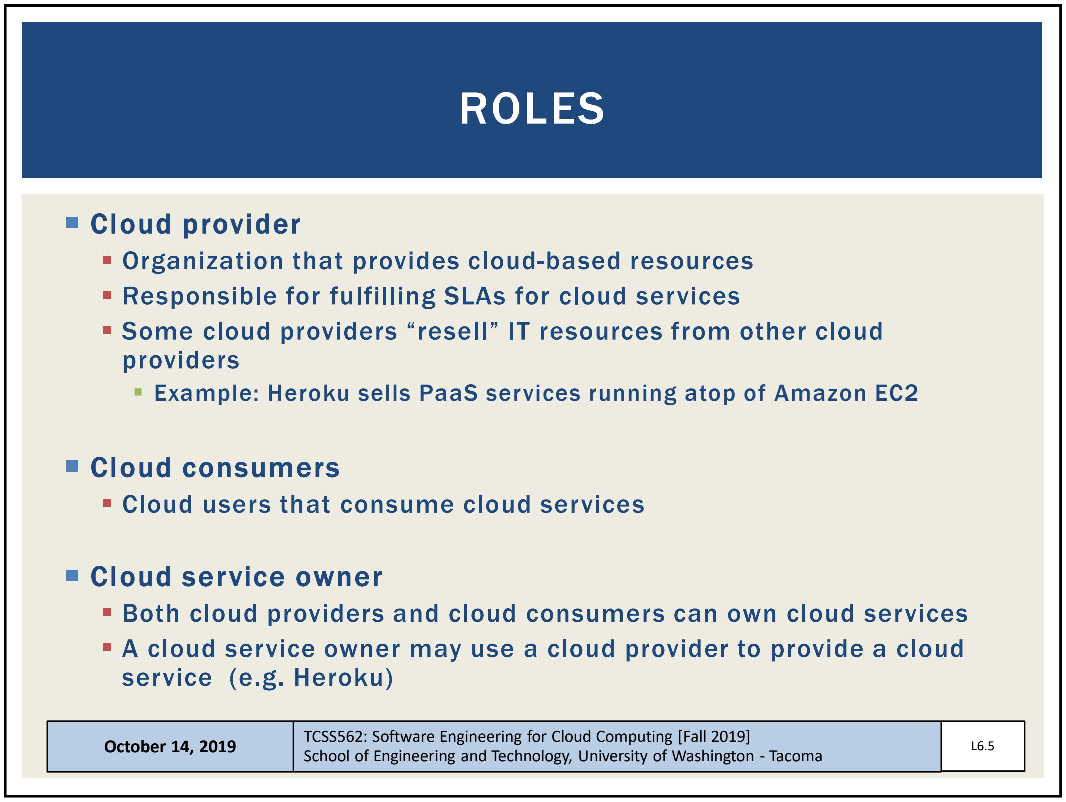
- From: Cloud Computing Concepts, Technology & Architecture:
- Cloud Computing Concepts and Models
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models

October 14, 2019

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

L6.4

4

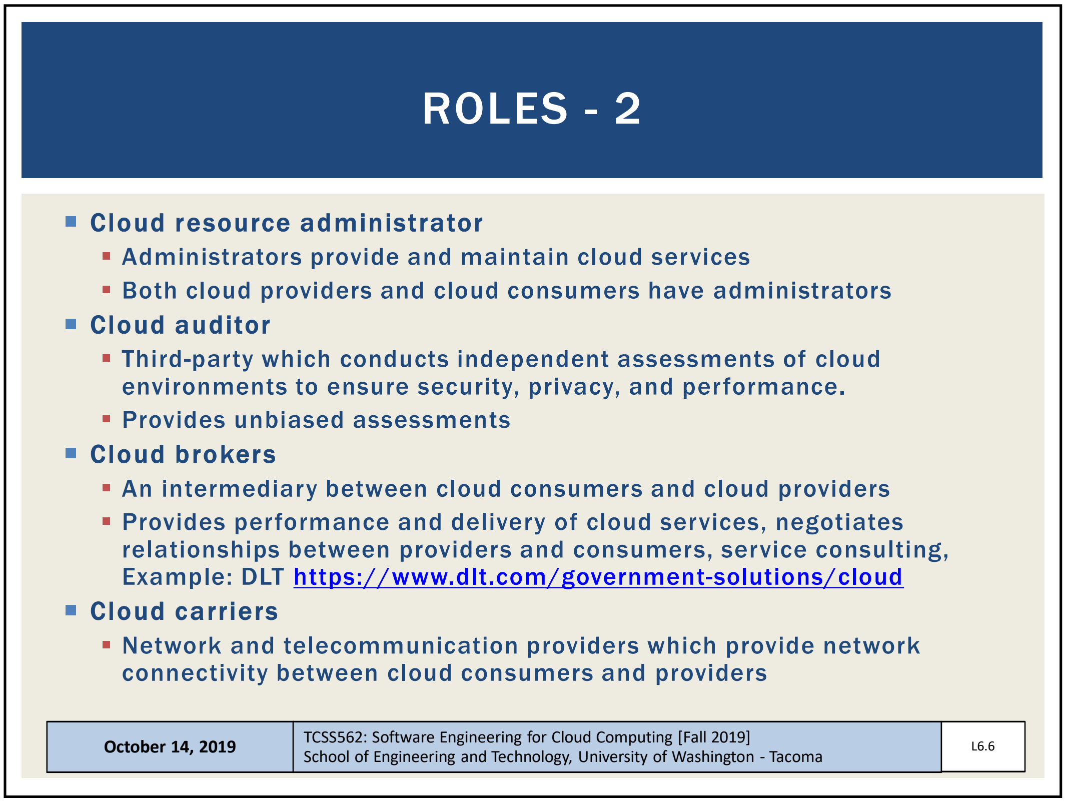


ROLES

- Cloud provider
  - Organization that provides cloud-based resources
  - Responsible for fulfilling SLAs for cloud services
  - Some cloud providers “resell” IT resources from other cloud providers
    - Example: Heroku sells PaaS services running atop of Amazon EC2
- Cloud consumers
  - Cloud users that consume cloud services
- Cloud service owner
  - Both cloud providers and cloud consumers can own cloud services
  - A cloud service owner may use a cloud provider to provide a cloud service (e.g. Heroku)

October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L6.5

5

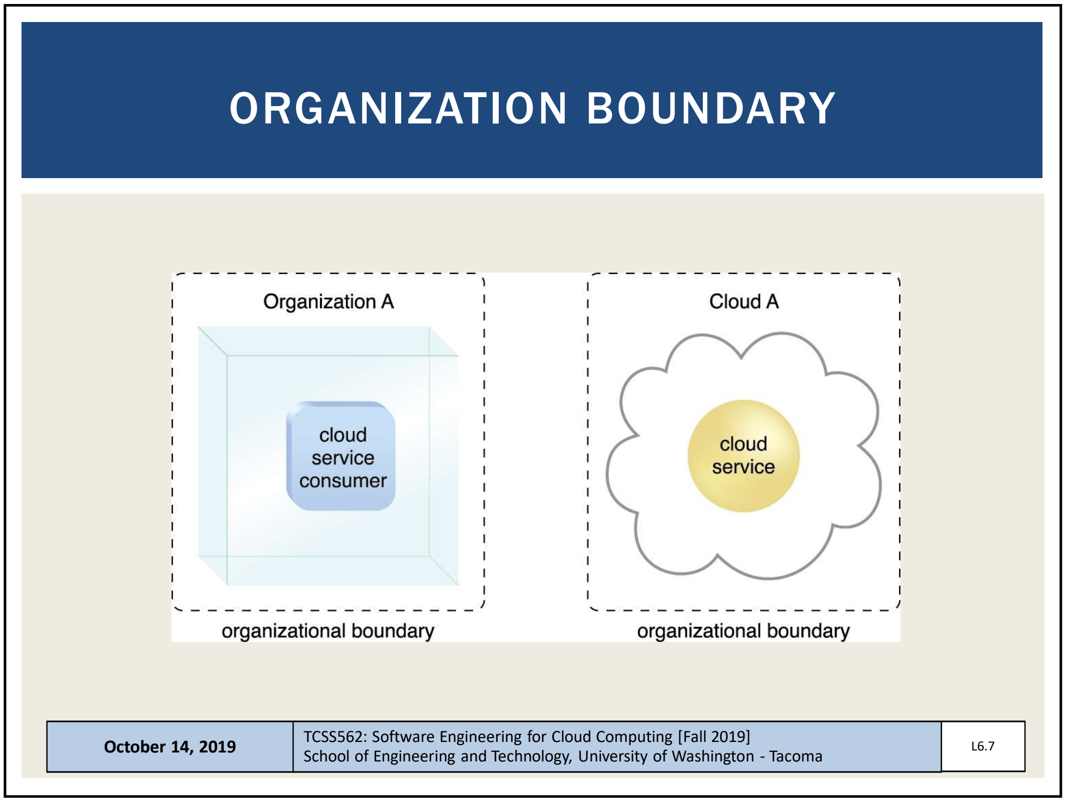


ROLES - 2

- Cloud resource administrator
  - Administrators provide and maintain cloud services
  - Both cloud providers and cloud consumers have administrators
- Cloud auditor
  - Third-party which conducts independent assessments of cloud environments to ensure security, privacy, and performance.
  - Provides unbiased assessments
- Cloud brokers
  - An intermediary between cloud consumers and cloud providers
  - Provides performance and delivery of cloud services, negotiates relationships between providers and consumers, service consulting, Example: DLT https://www.dlt.com/government-solutions/cloud
- Cloud carriers
  - Network and telecommunication providers which provide network connectivity between cloud consumers and providers

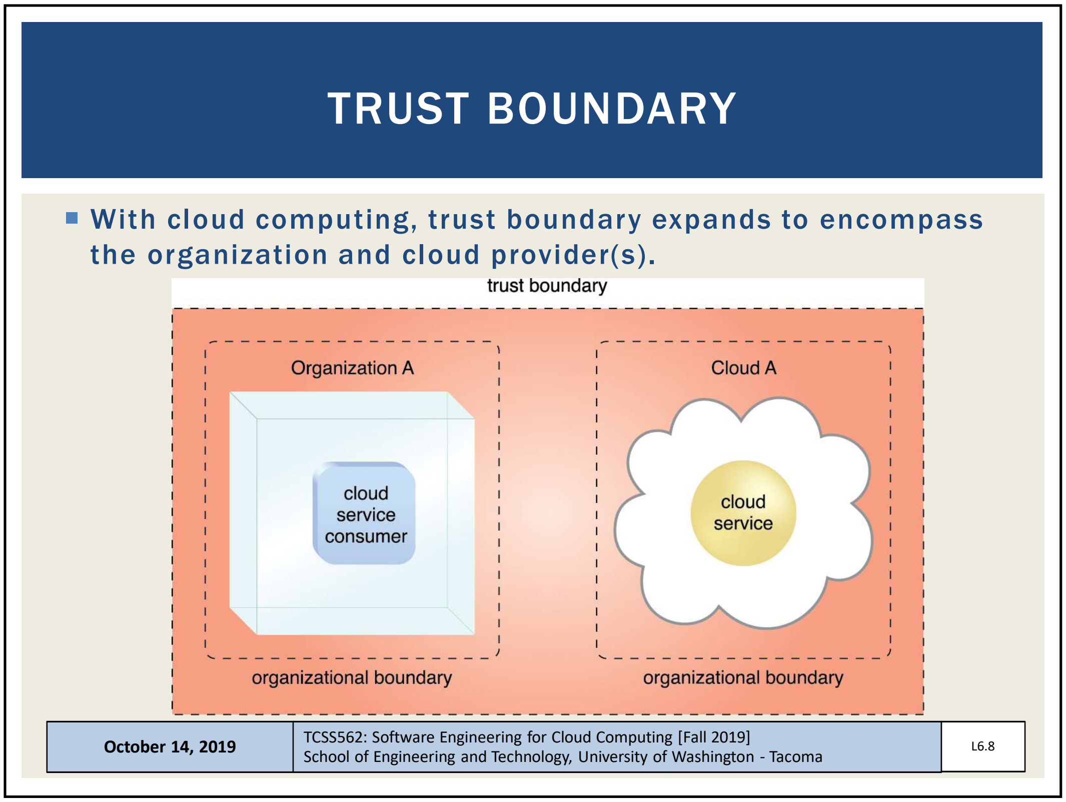
October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L6.6

6



ORGANIZATION BOUNDARY

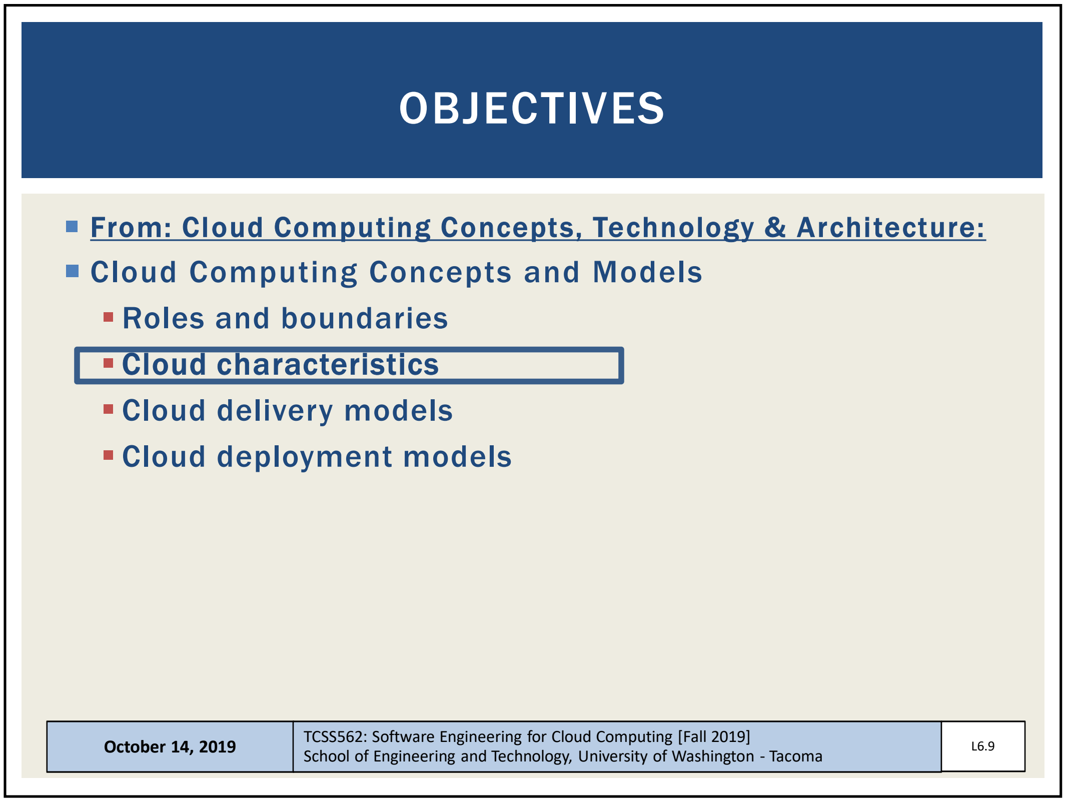
7



TRUST BOUNDARY

- With cloud computing, trust boundary expands to encompass the organization and cloud provider(s).

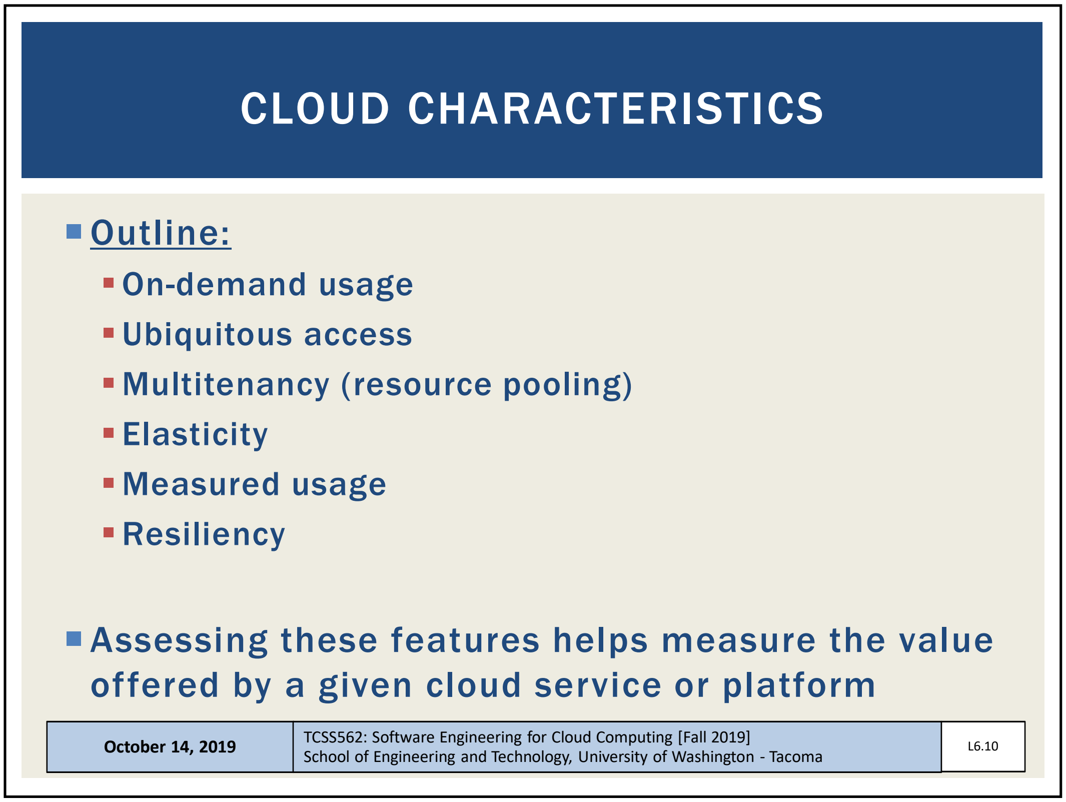
8



OBJECTIVES

- **From: Cloud Computing Concepts, Technology & Architecture:**
- Cloud Computing Concepts and Models
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models

9



CLOUD CHARACTERISTICS

- **Outline:**
  - On-demand usage
  - Ubiquitous access
  - Multitenancy (resource pooling)
  - Elasticity
  - Measured usage
  - Resiliency

- Assessing these features helps measure the value offered by a given cloud service or platform

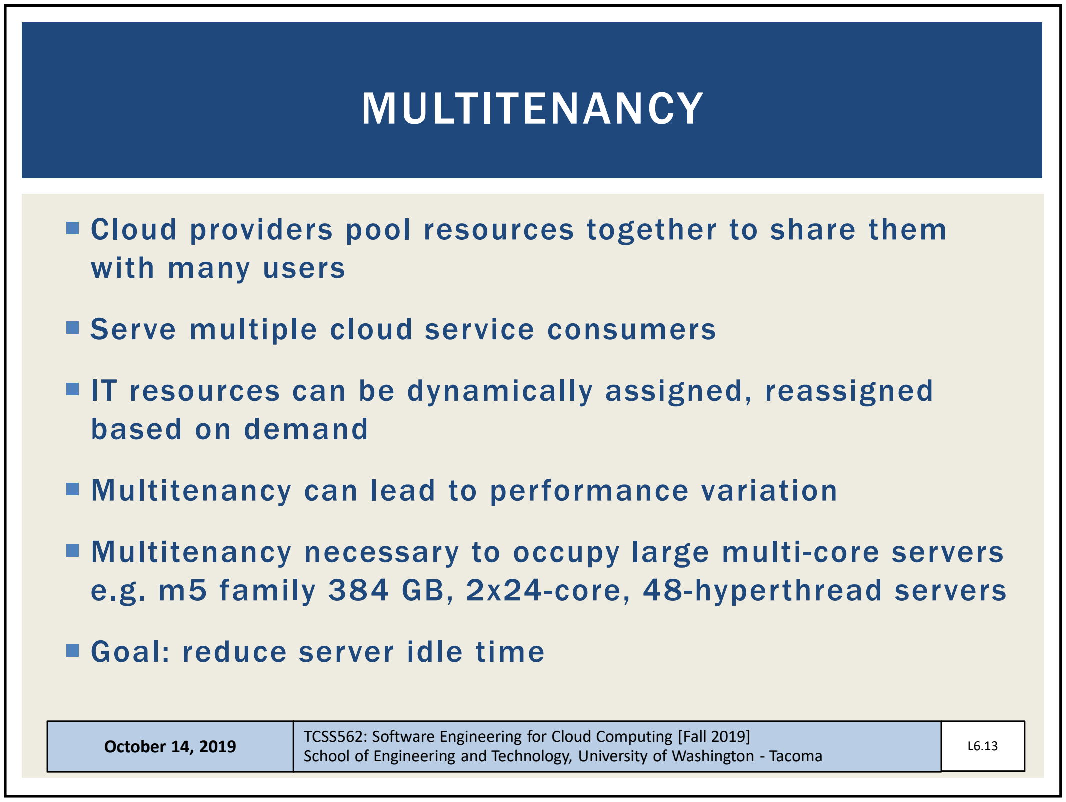
10

ON-DEMAND USAGE

- The freedom to self-provision IT resources
- Generally with automated support
- Automated support requires no human involvement
- Automation through software services interface

UBIQUITOUS ACCESS

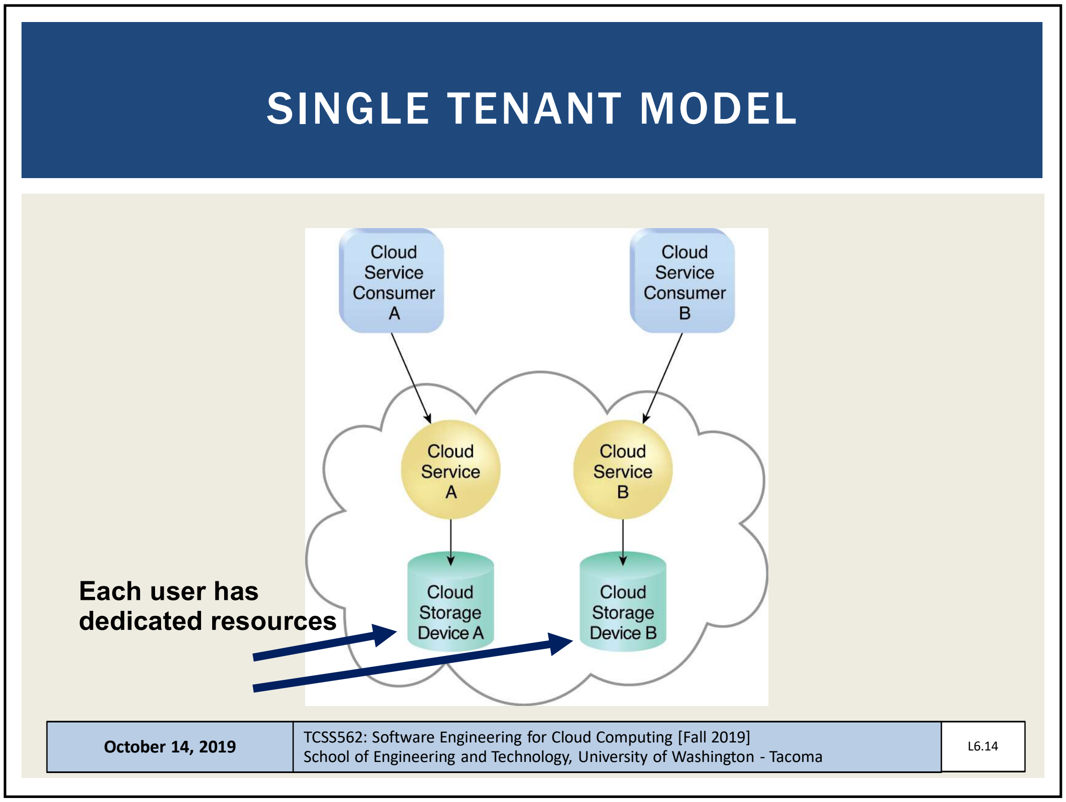
- Cloud services are widely accessible
- Public cloud: internet accessible
- Private cloud: throughout segments of a company’s intranet
- 24/7 availability



MULTITENANCY

- Cloud providers pool resources together to share them with many users
- Serve multiple cloud service consumers
- IT resources can be dynamically assigned, reassigned based on demand
- Multitenancy can lead to performance variation
- Multitenancy necessary to occupy large multi-core servers e.g. m5 family 384 GB, 2x24-core, 48-hyperthread servers
- Goal: reduce server idle time

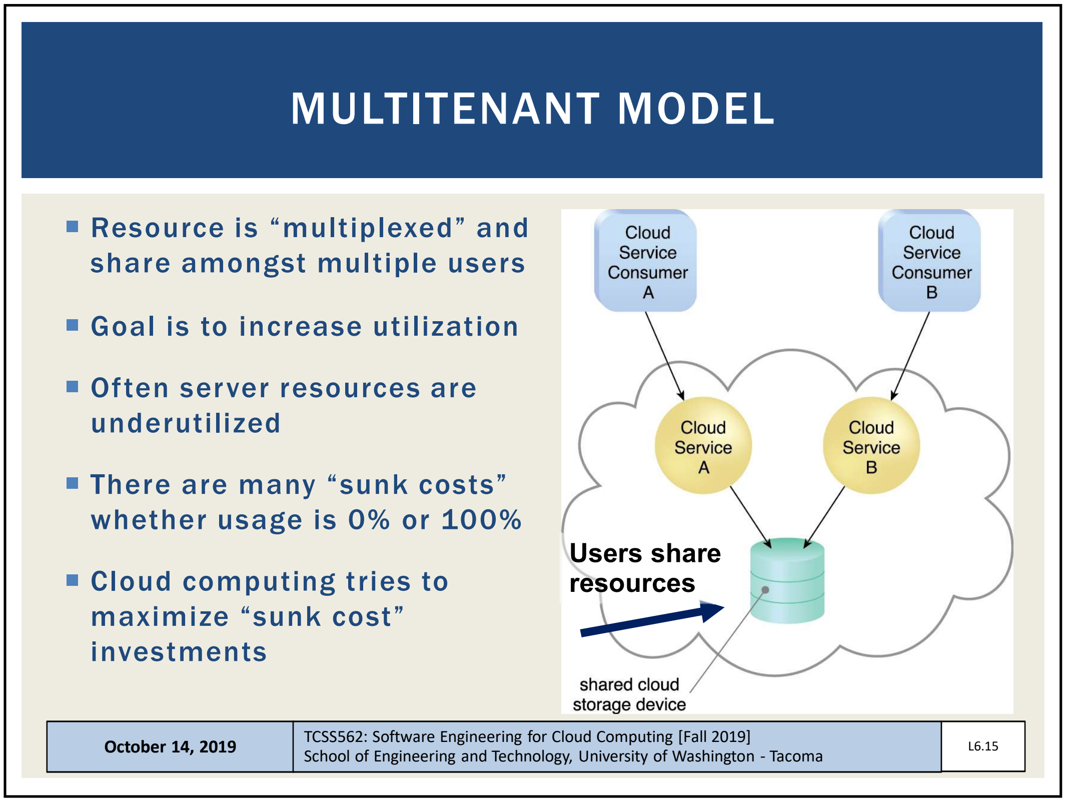
13



SINGLE TENANT MODEL

Each user has dedicated resources

14

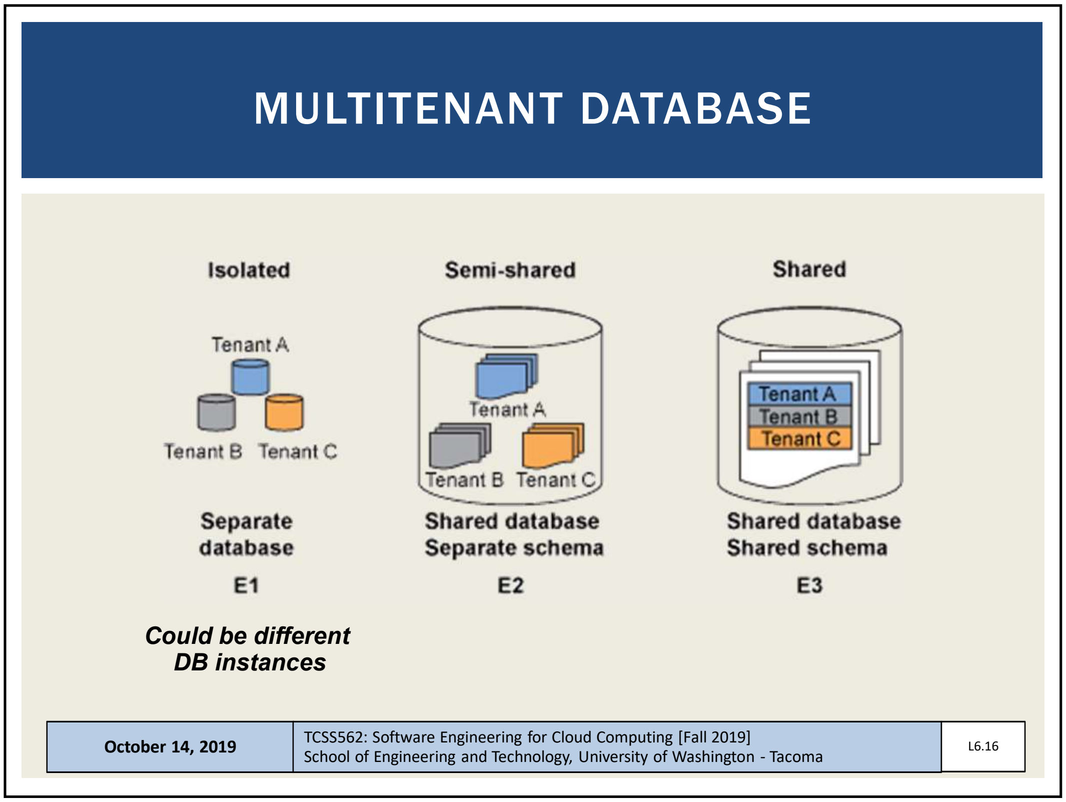


MULTITENANT MODEL

- Resource is "multiplexed" and share amongst multiple users
- Goal is to increase utilization
- Often server resources are underutilized
- There are many "sunk costs" whether usage is 0% or 100%
- Cloud computing tries to maximize "sunk cost" investments

Users share resources

15



MULTITENANT DATABASE

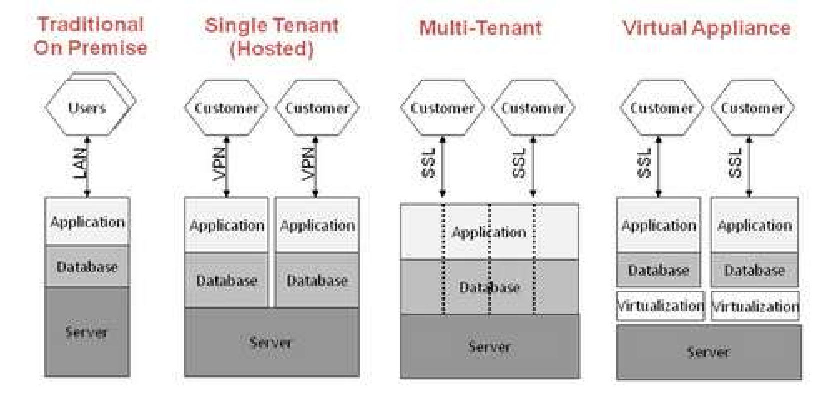
Separate database	Shared database Separate schema	Shared database Shared schema
E1	E2	E3

Could be different DB instances

16

MULTITENANCY OF RESOURCES

- Where is the multitenancy?



MEASUREMENT STUDY OF SERVER UTILIZATION IN PUBLIC CLOUDS

H. Liu, A Measurement Study of Server Utilization in Public Clouds, Proc. 9th IEEE International Conference on Cloud and Green Computing (CAG'11), Sydney, Australia, Dec 2011, pp.435-442.

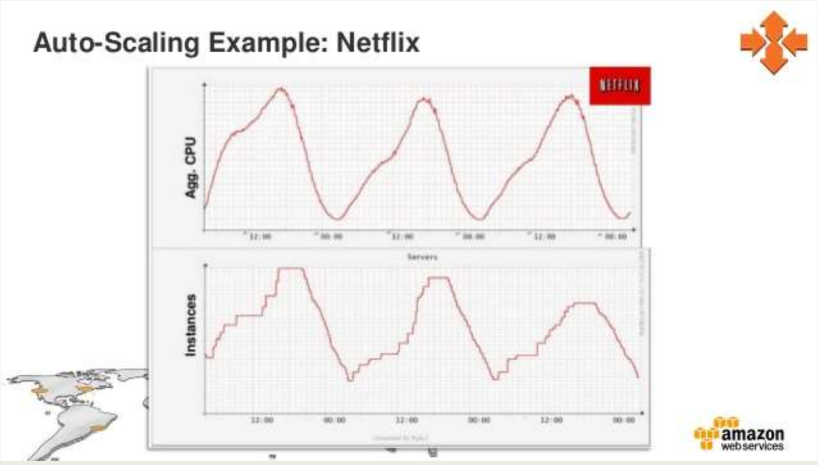
- H. Liu characterized CPU utilization across a public cloud by analyzing CPU temperature
- Liu's approach averages thermal measurements of the CPU from small VMs which are context switched across the physical host's CPU cores for extended periods to approximate CPU die temperature
- Local tests on private cluster established correlation between CPU die temperature and CPU utilization
- Using this approach Liu observed CPU utilization using 20 m1.small VMs on Amazon EC2 in 2011 for 1 week and estimated average CPU utilization of the physical hosts to be around 7.3%

ELASTICITY

- Automated ability of cloud to transparently scale resources
- Scaling based on runtime conditions or pre-determined by cloud consumer or cloud provider
- Threshold based scaling
  - Application agnostic:
    - `CPU-utilization > threshold_A`
  - Application specific:
    - `Response_time > 100ms`
  - Why might an application agnostic threshold be non-ideal?
- Load prediction
  - Historical models *(historical data)*
  - Real-time trends *(live observations)*

PREDICTABLE DEMAND

- Example:

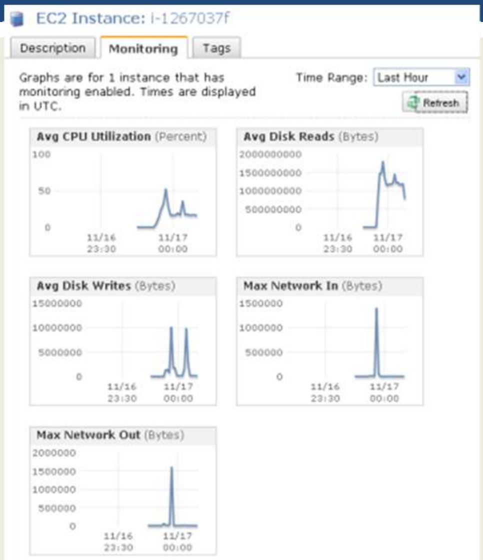

Auto-Scaling Example: Netflix

MEASURED USAGE

- Cloud platform tracks usage of IT resources
- For billing purposes
- Enables charging only for IT resources actually used
- Can be time-based (minute, hour, day)
- Can be throughput-based (MB, GB)

- Not all measurements are for billing
- Some measurements can support auto-scaling
- For example CPU utilization

EC2 CLOUDWATCH METRICS

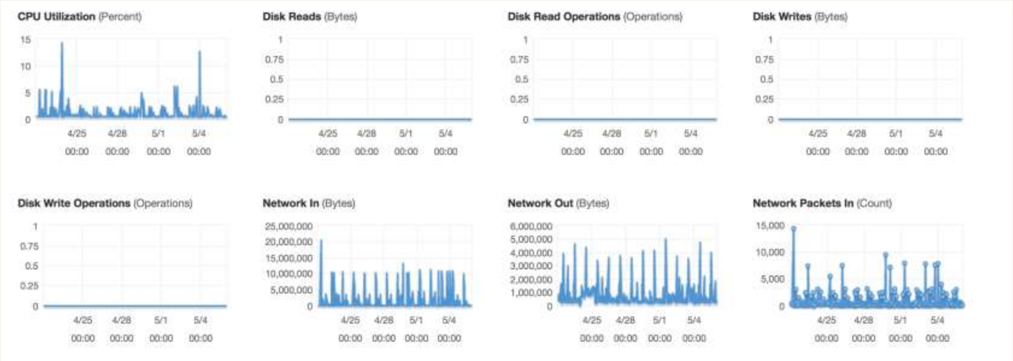


EC2 Instance: i-1267037f

Graphs are for 1 instance that has monitoring enabled. Times are displayed in UTC.

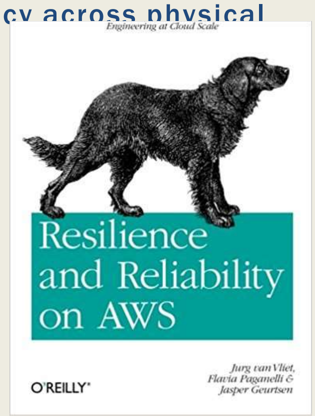
Time Range: Last Hour

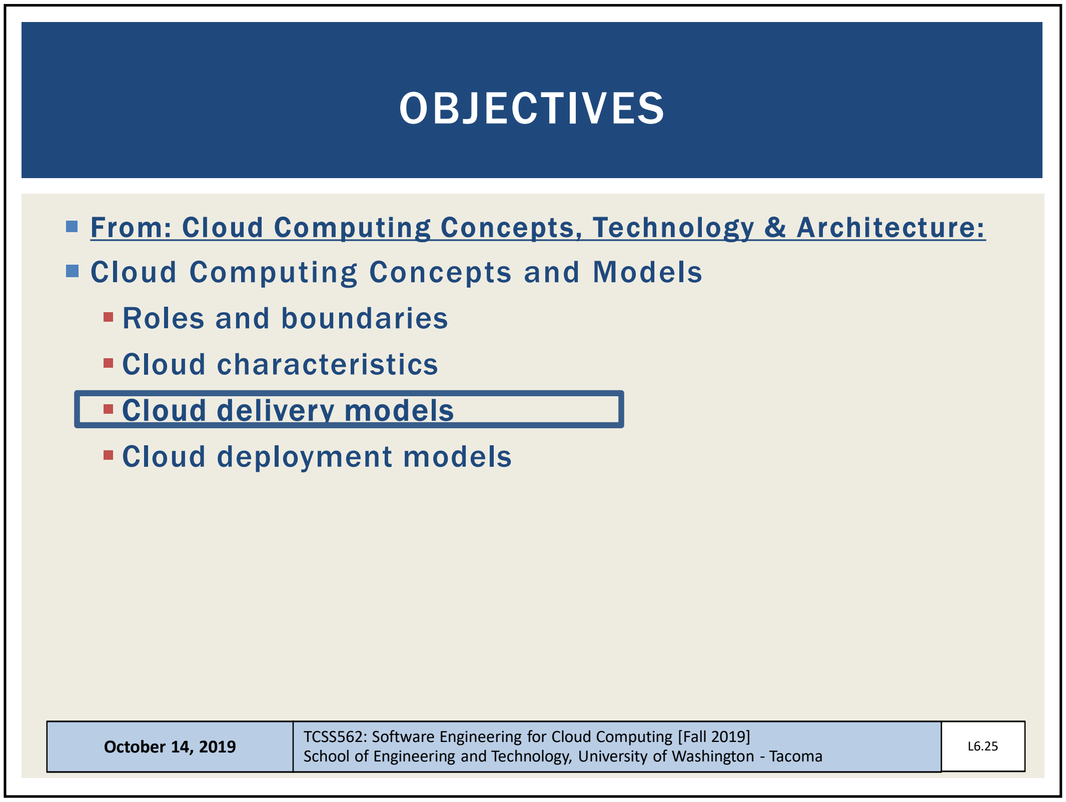
EC2 CLOUDWATCH METRICS



RESILIENCY

- Ability to withstand a major disruption within acceptable degradation parameters and recover within an acceptable time.
- Cloud achieves resiliency through redundancy across physical locations *(regions, availability zones)*
- Redundancy used to improve reliability and availability of cloud-hosted applications
- Very much an engineering problem
- No “resiliency-as-a-service” for user deployed apps
- Unique characteristics of user applications make a one-size fits all service solution challenging


Engineering at Cloud Scale
Resilience and Reliability on AWS
O'REILLY
Jurg van Vliet, Flavia Paganelli & Jasper Geurtsen

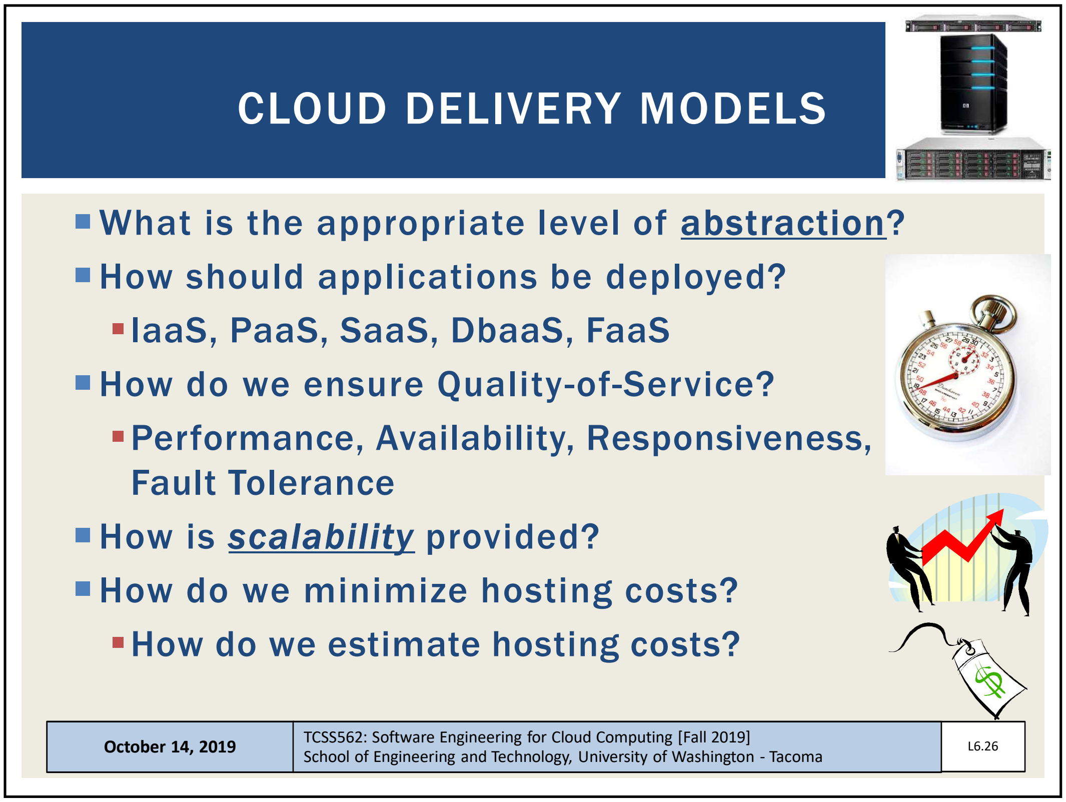


OBJECTIVES

- From: Cloud Computing Concepts, Technology & Architecture:
- Cloud Computing Concepts and Models
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models

October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L6.25

25

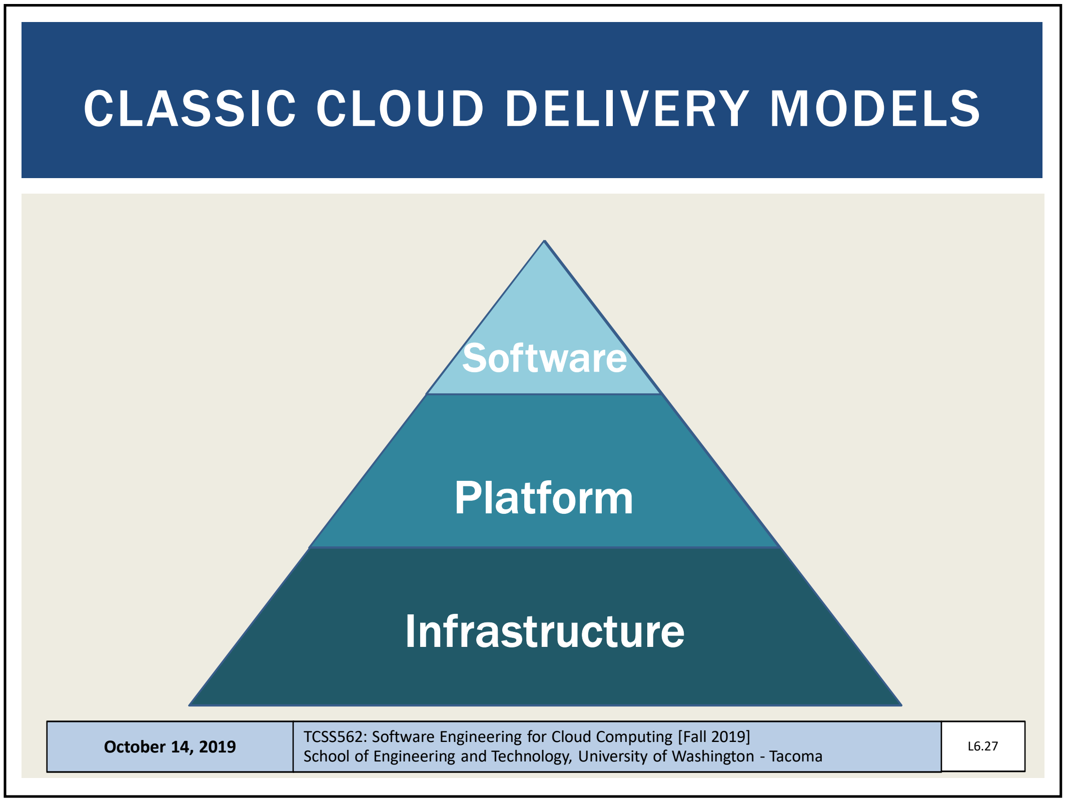


CLOUD DELIVERY MODELS

- What is the appropriate level of abstraction?
- How should applications be deployed?
  - IaaS, PaaS, SaaS, DbaaS, FaaS
- How do we ensure Quality-of-Service?
  - Performance, Availability, Responsiveness, Fault Tolerance
- How is *scalability* provided?
- How do we minimize hosting costs?
  - How do we estimate hosting costs?

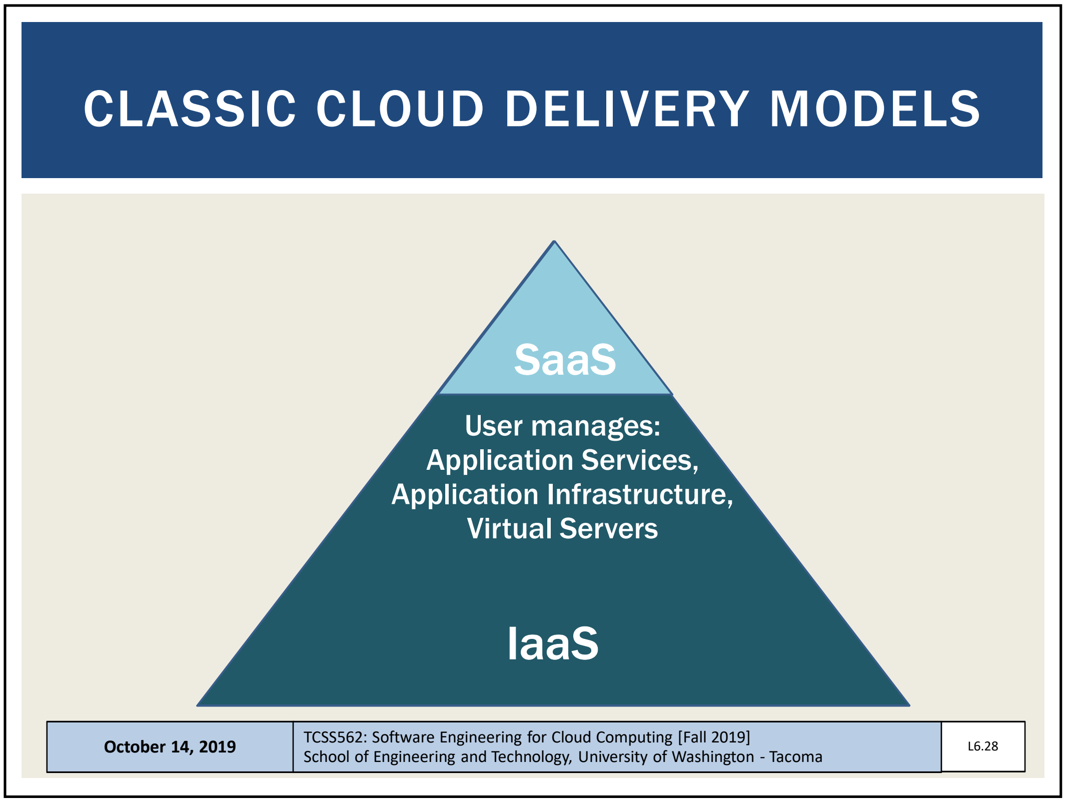
October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L6.26

26



CLASSIC CLOUD DELIVERY MODELS

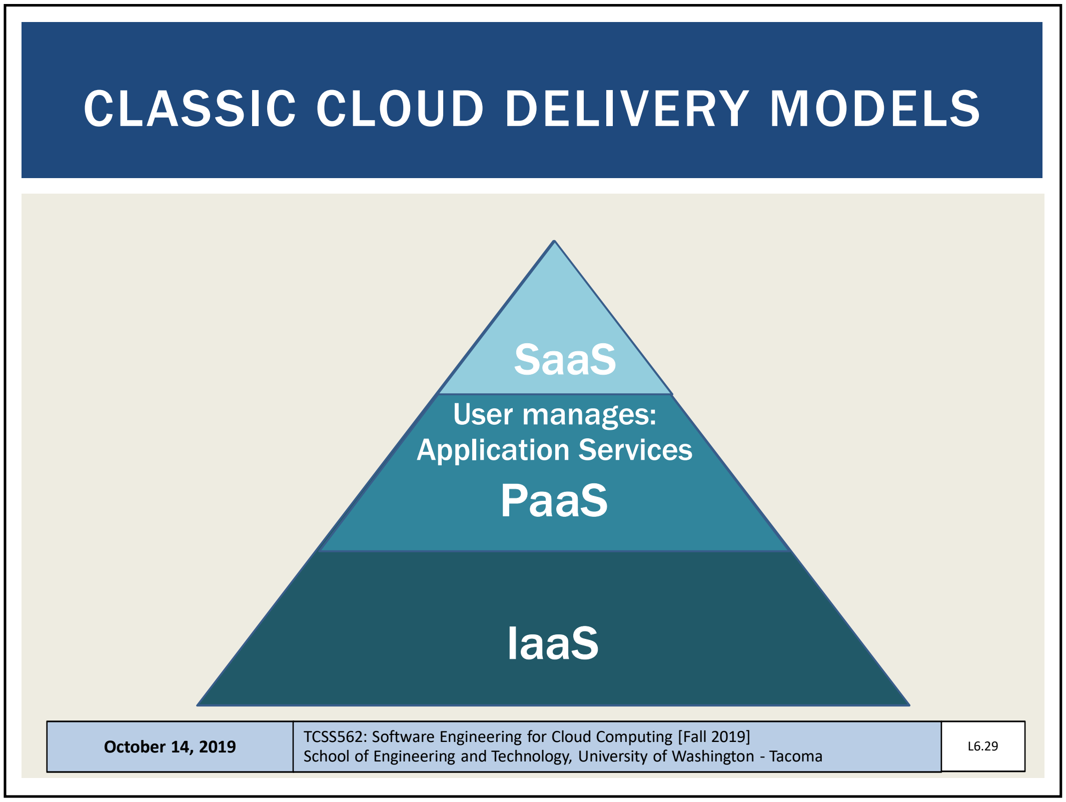
27



CLASSIC CLOUD DELIVERY MODELS

User manages:
Application Services,
Application Infrastructure,
Virtual Servers

28



CLASSIC CLOUD DELIVERY MODELS

SaaS

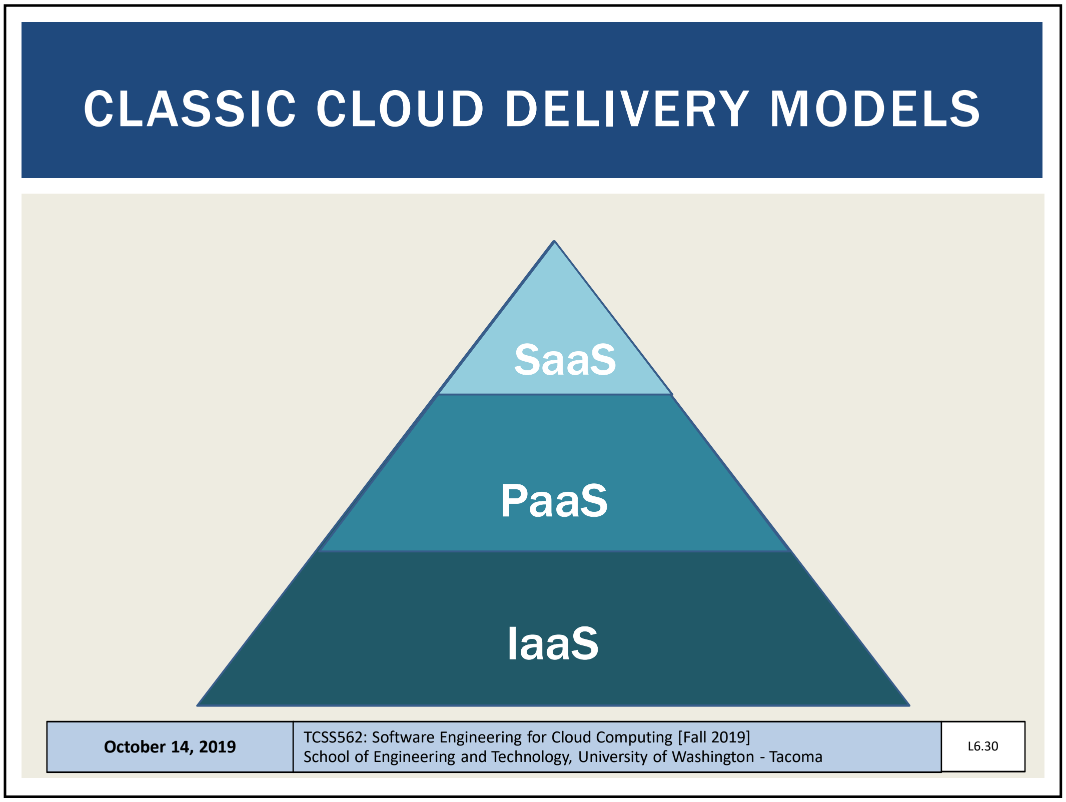
User manages:
Application Services

PaaS

IaaS

October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma | L6.29

29



CLASSIC CLOUD DELIVERY MODELS

SaaS

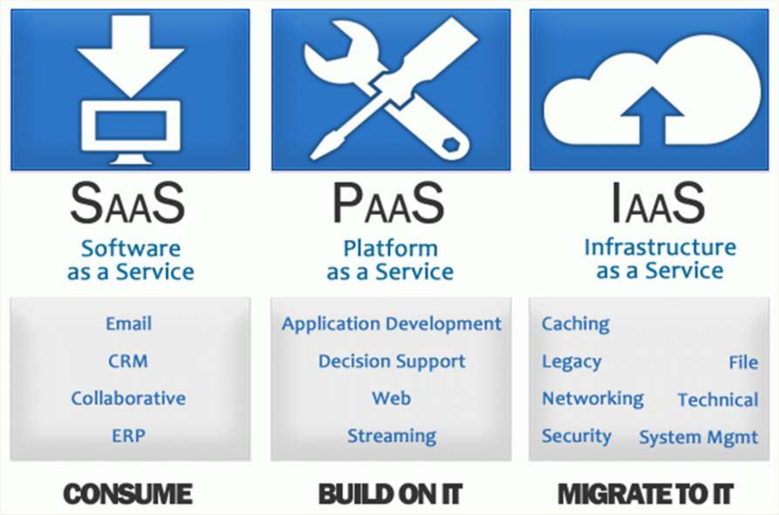
PaaS

IaaS

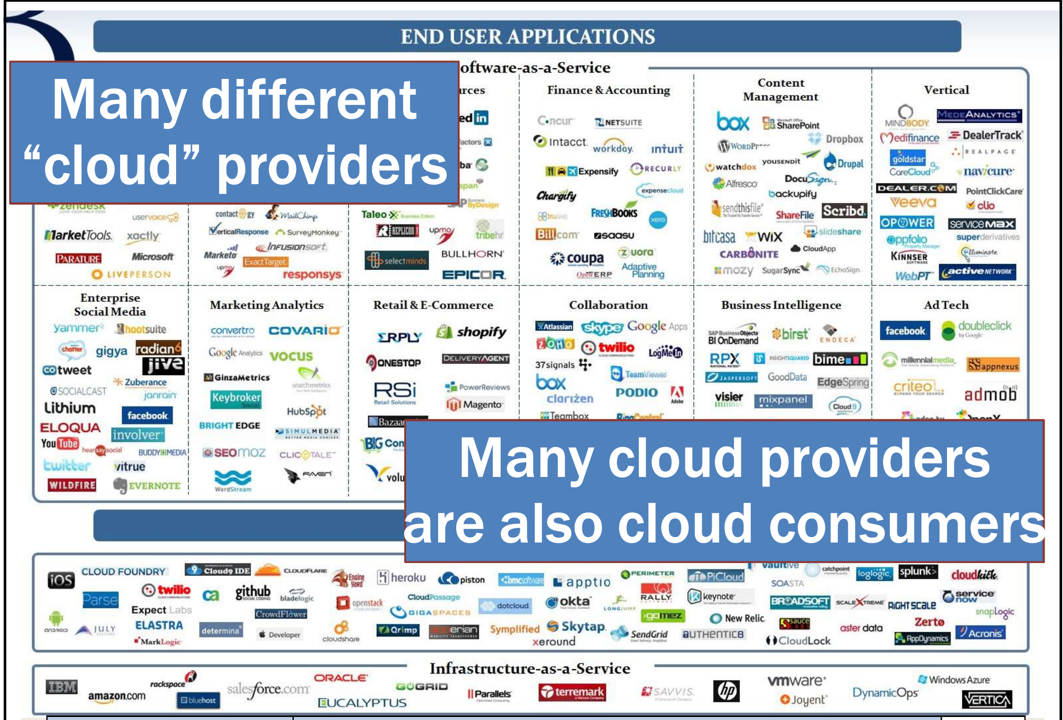
October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma | L6.30

30

EXAMPLE CLOUD SERVICES



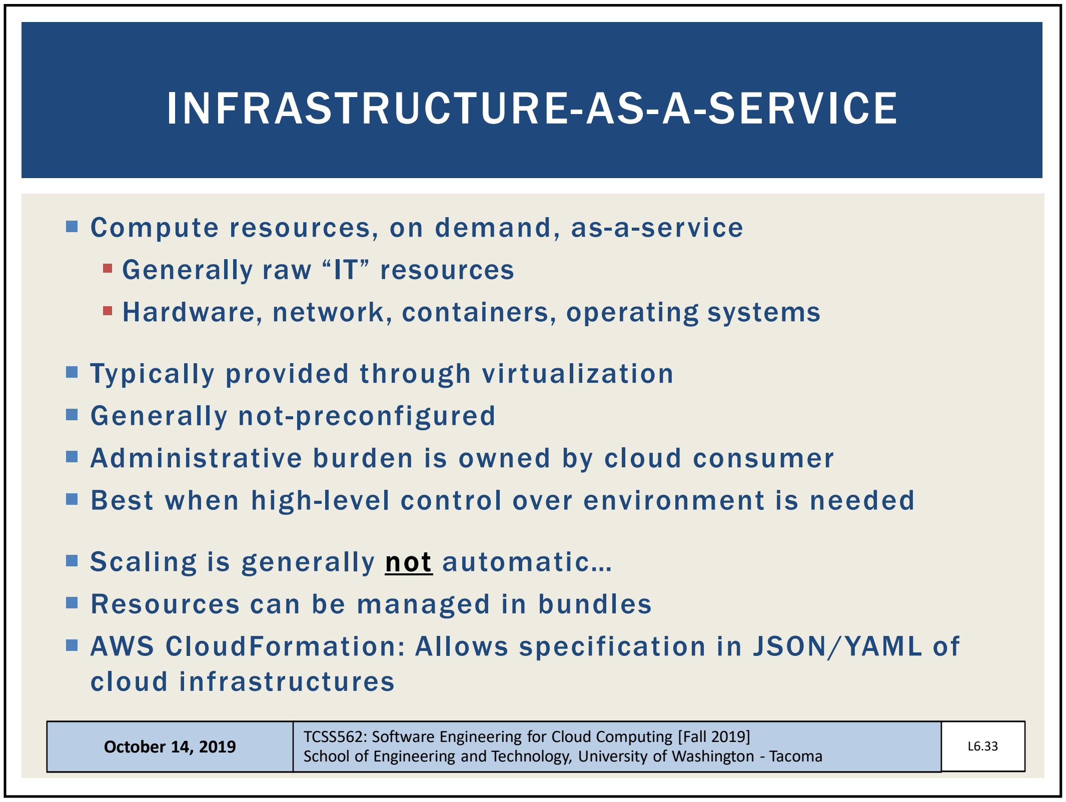
31



Many different “cloud” providers

Many cloud providers are also cloud consumers

32

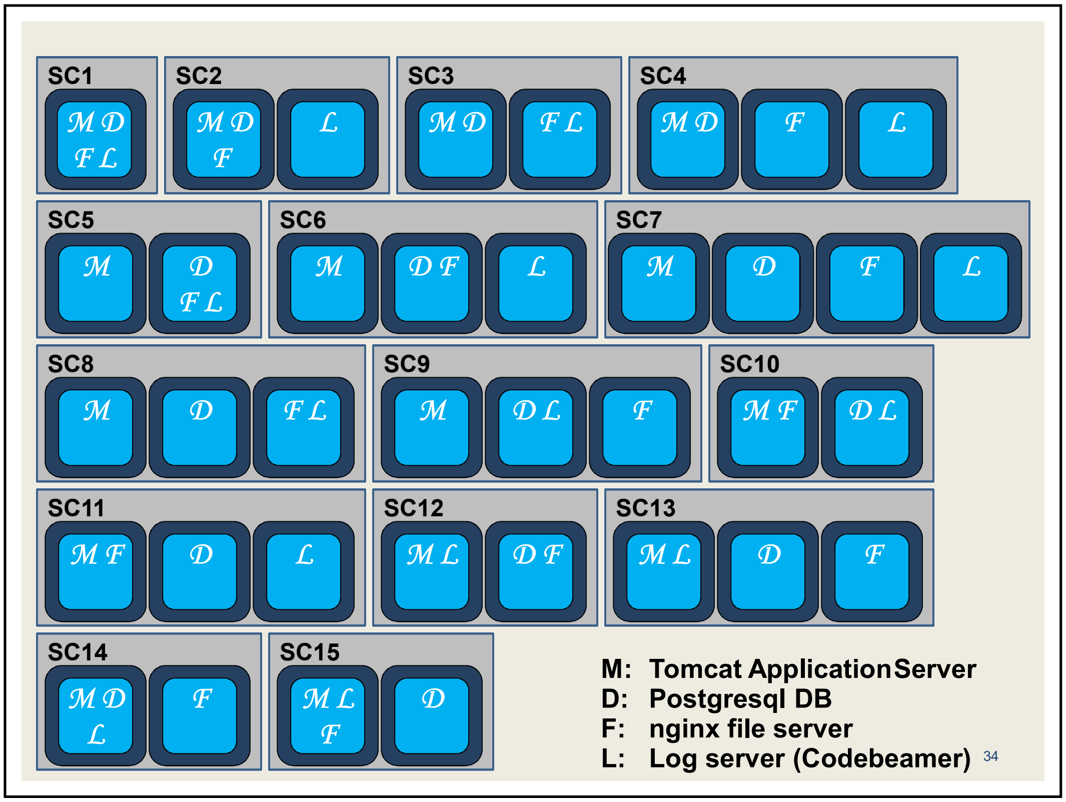


INFRASTRUCTURE-AS-A-SERVICE

- Compute resources, on demand, as-a-service
  - Generally raw "IT" resources
  - Hardware, network, containers, operating systems
- Typically provided through virtualization
- Generally not-preconfigured
- Administrative burden is owned by cloud consumer
- Best when high-level control over environment is needed
- Scaling is generally **not** automatic…
- Resources can be managed in bundles
- AWS CloudFormation: Allows specification in JSON/YAML of cloud infrastructures

October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L6.33

33



SC1
M D F L

SC2
M D F
L

SC3
M D
F L

SC4
M D
F
L

SC5
M
D F L

SC6
M
D F
L

SC7
M
D
F
L

SC8
M
D
F L

SC9
M
D L
F

SC10
M F
D L

SC11
M F
D
L

SC12
M L
D F

SC13
M L
D
F

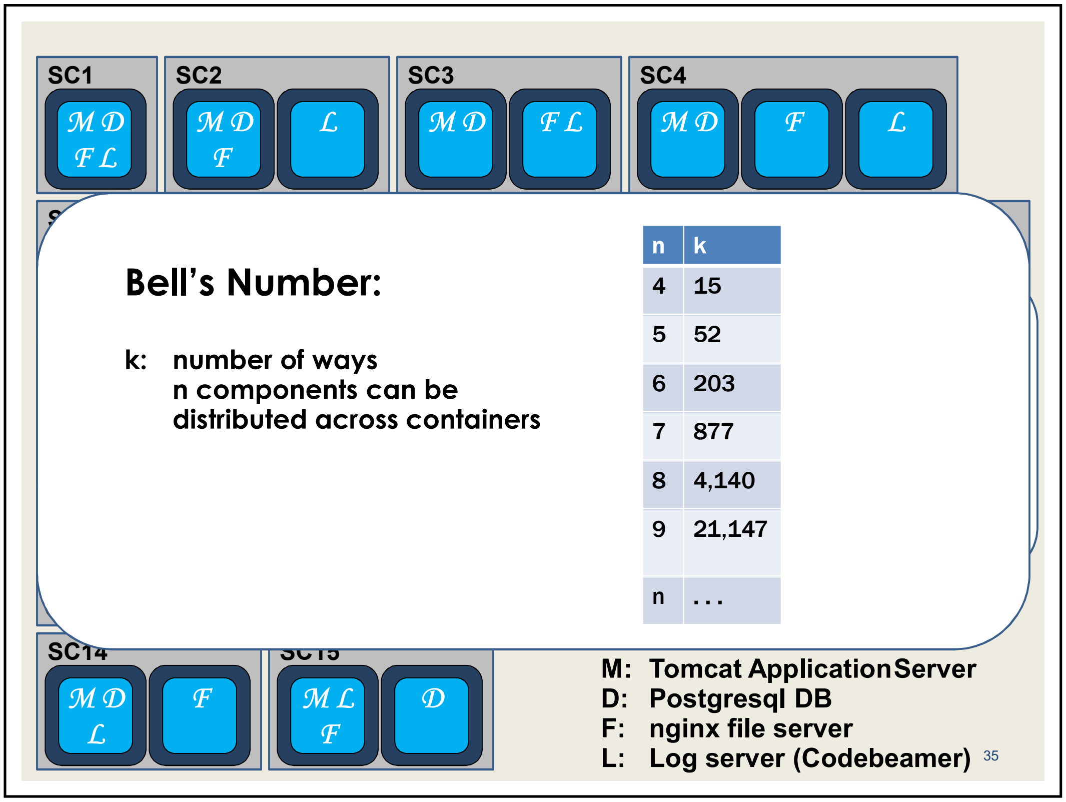
SC14
M D L
F

SC15
M L F
D

M: Tomcat Application Server
D: Postgresql DB
F: nginx file server
L: Log server (Codebeamer)

34

34



Bell's Number:

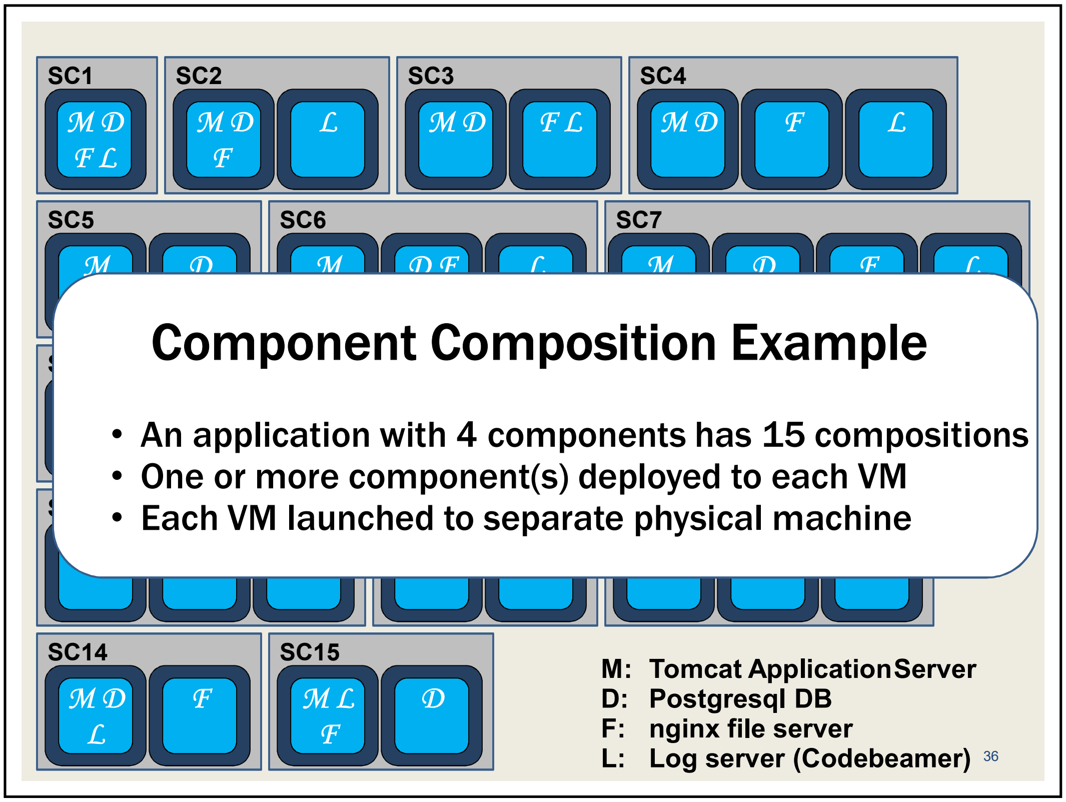
k: number of ways n components can be distributed across containers

n	k
4	15
5	52
6	203
7	877
8	4,140
9	21,147
n	. . .

M: Tomcat Application Server
D: Postgresql DB
F: nginx file server
L: Log server (Codebeamer)

35

35



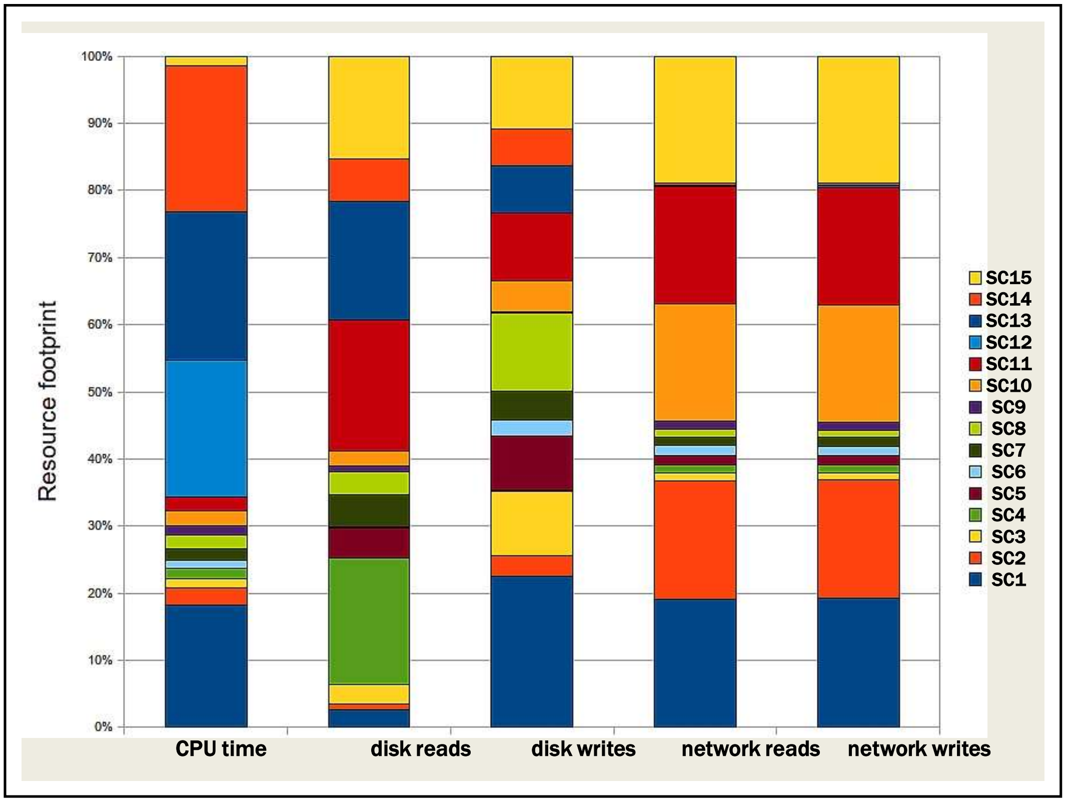
Component Composition Example

- An application with 4 components has 15 compositions
- One or more component(s) deployed to each VM
- Each VM launched to separate physical machine

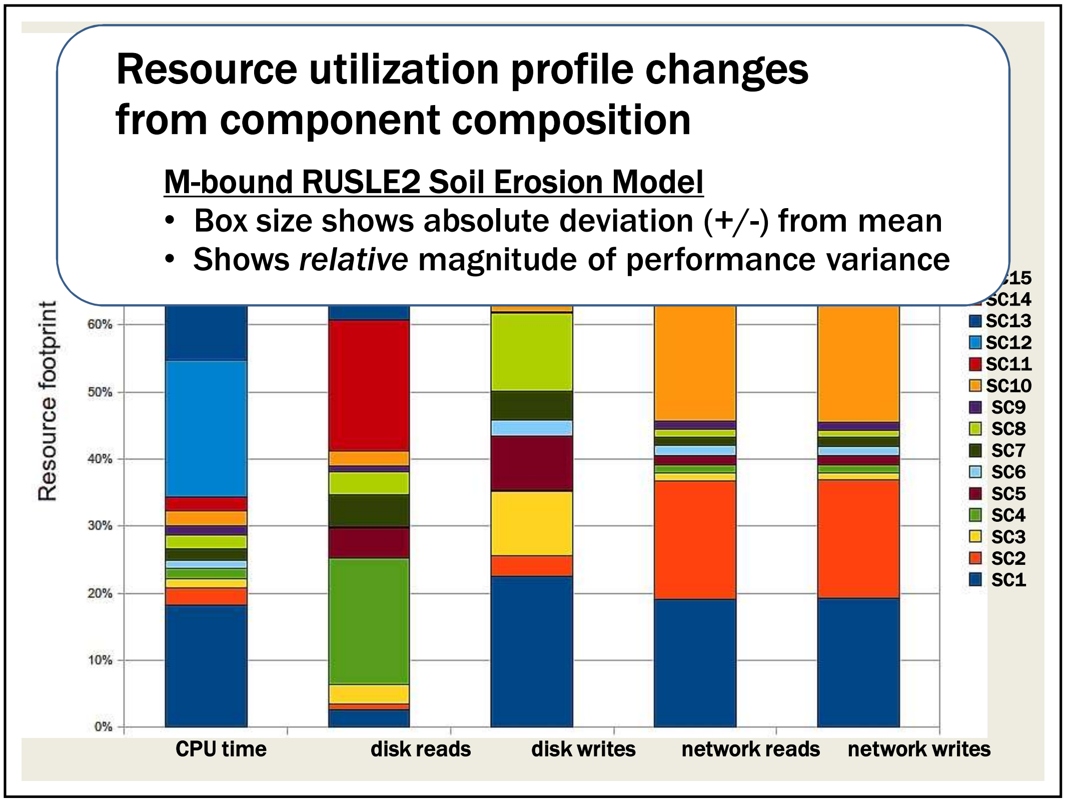
M: Tomcat Application Server
D: Postgresql DB
F: nginx file server
L: Log server (Codebeamer)

36

36



37

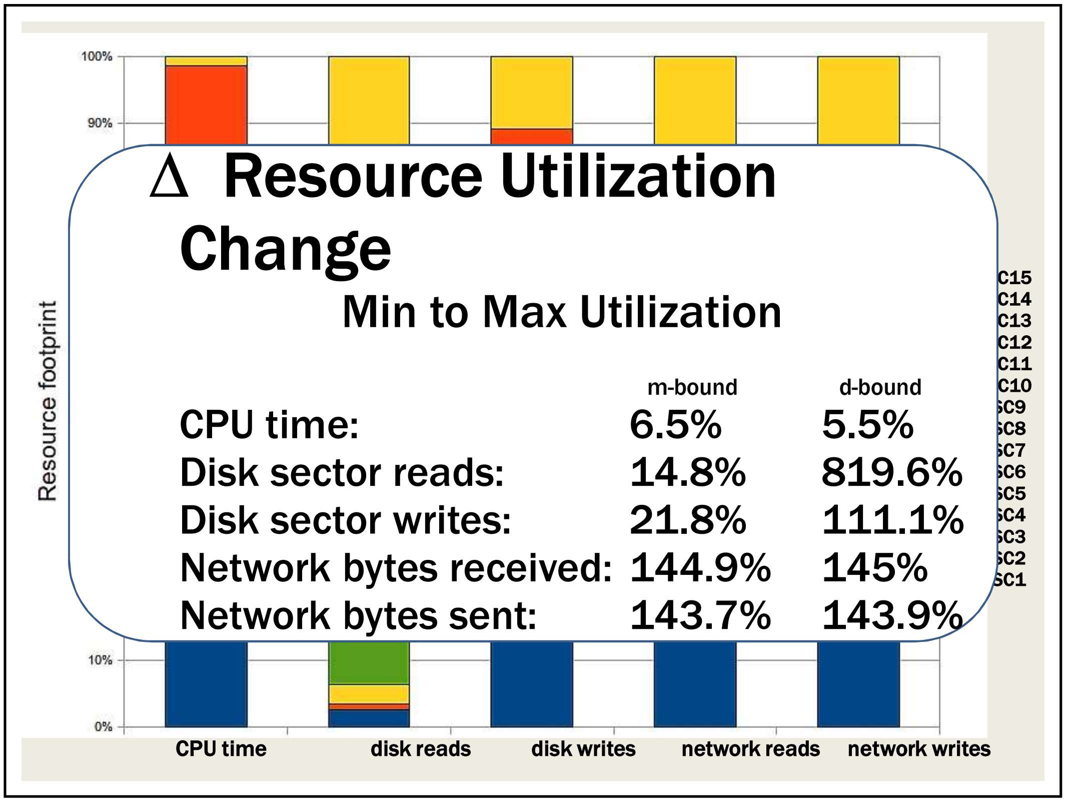


Resource utilization profile changes from component composition

M-bound RUSLE2 Soil Erosion Model

- Box size shows absolute deviation (+/-) from mean
- Shows *relative* magnitude of performance variance

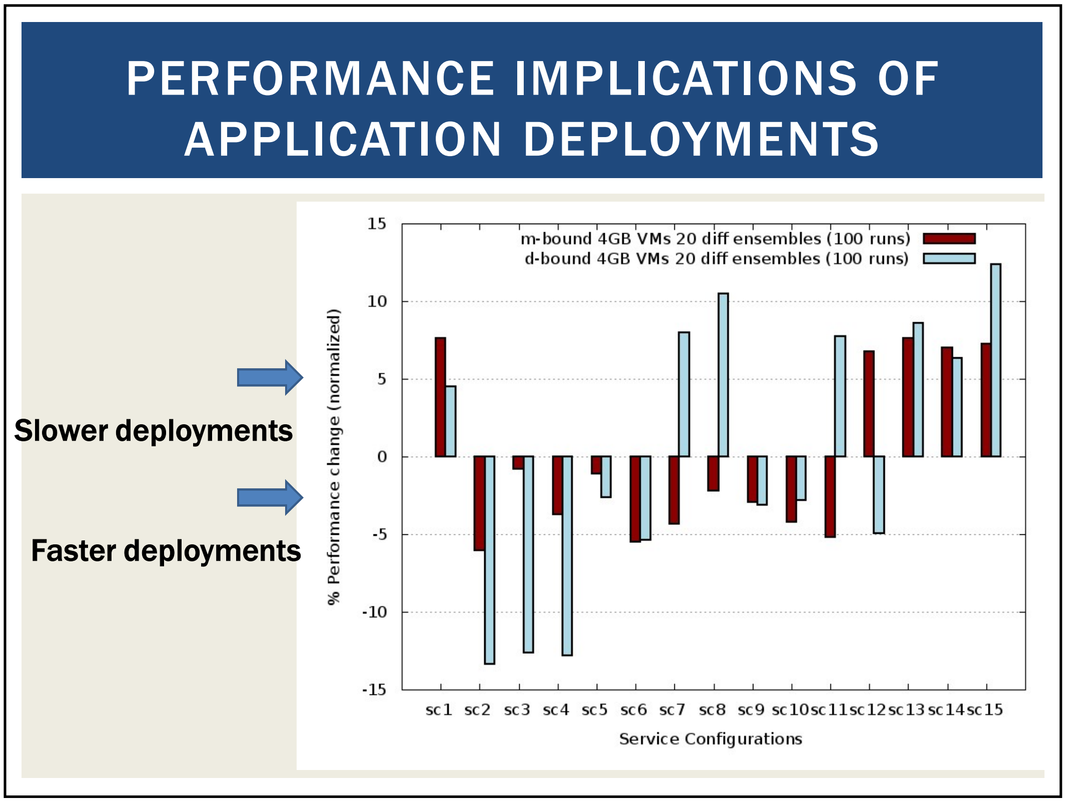
38



∆ Resource Utilization Change

Min to Max Utilization

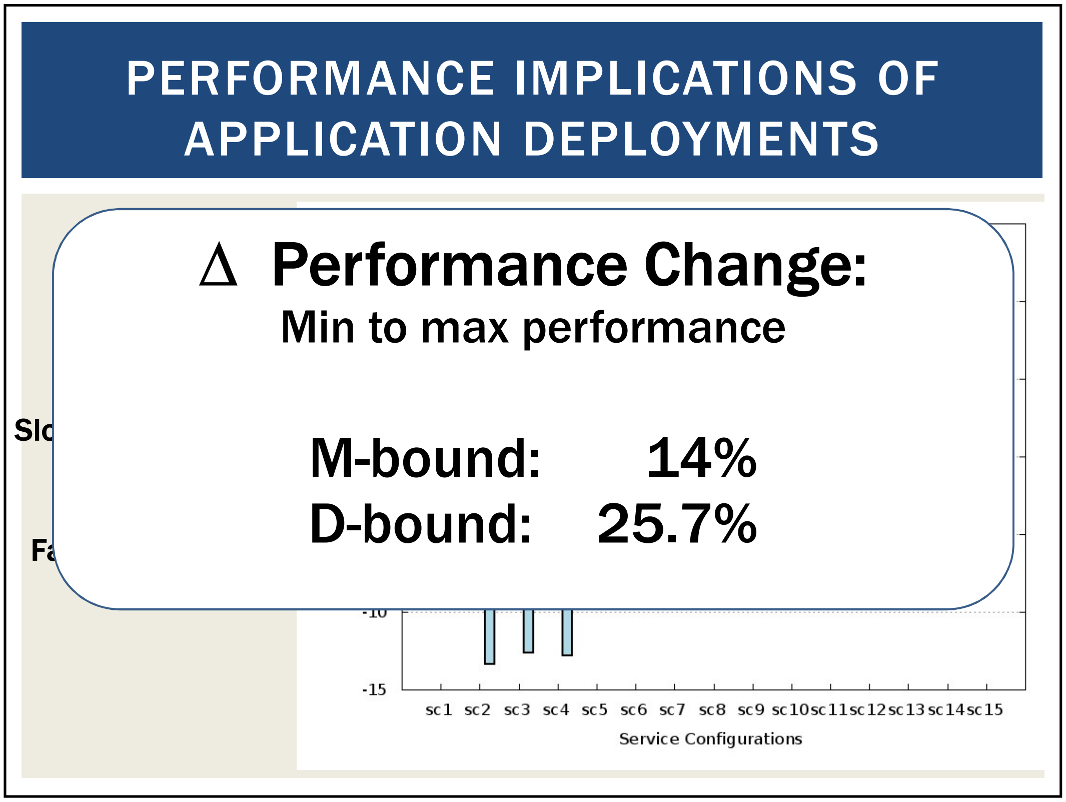
	m-bound	d-bound
CPU time:	6.5%	5.5%
Disk sector reads:	14.8%	819.6%
Disk sector writes:	21.8%	111.1%
Network bytes received:	144.9%	145%
Network bytes sent:	143.7%	143.9%



PERFORMANCE IMPLICATIONS OF APPLICATION DEPLOYMENTS

Slower deployments

Faster deployments



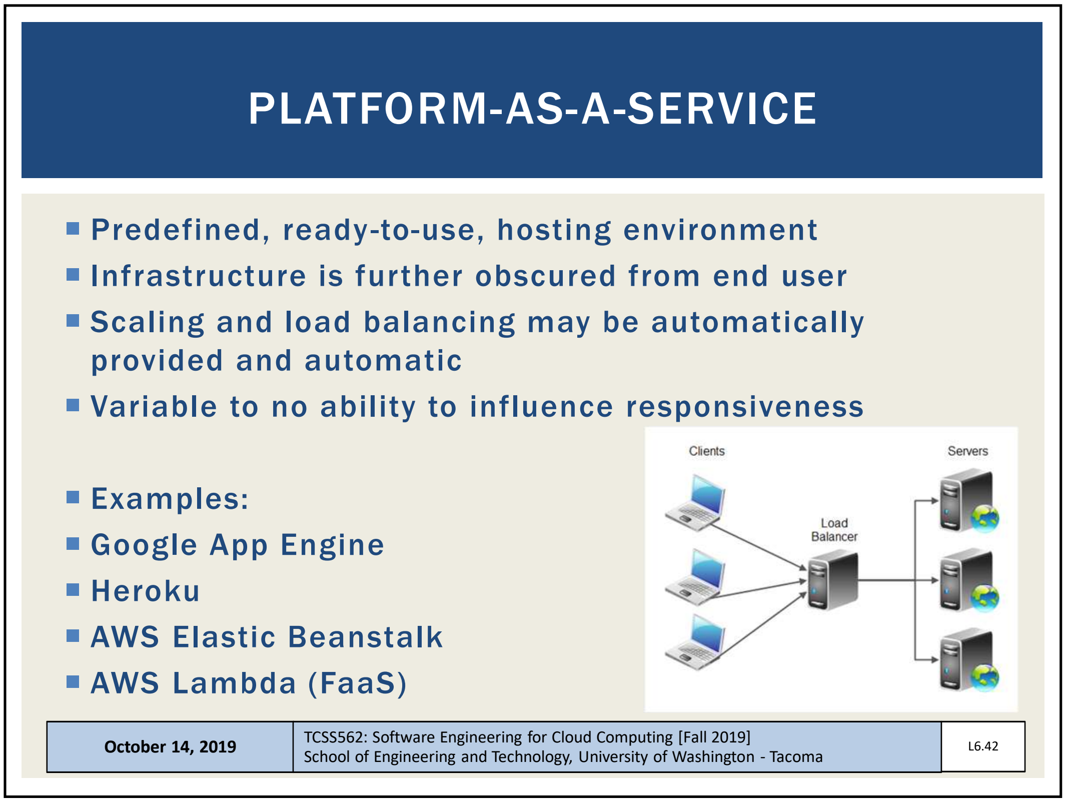
PERFORMANCE IMPLICATIONS OF APPLICATION DEPLOYMENTS

Δ Performance Change:

Min to max performance

M-bound:	14%
D-bound:	25.7%

41

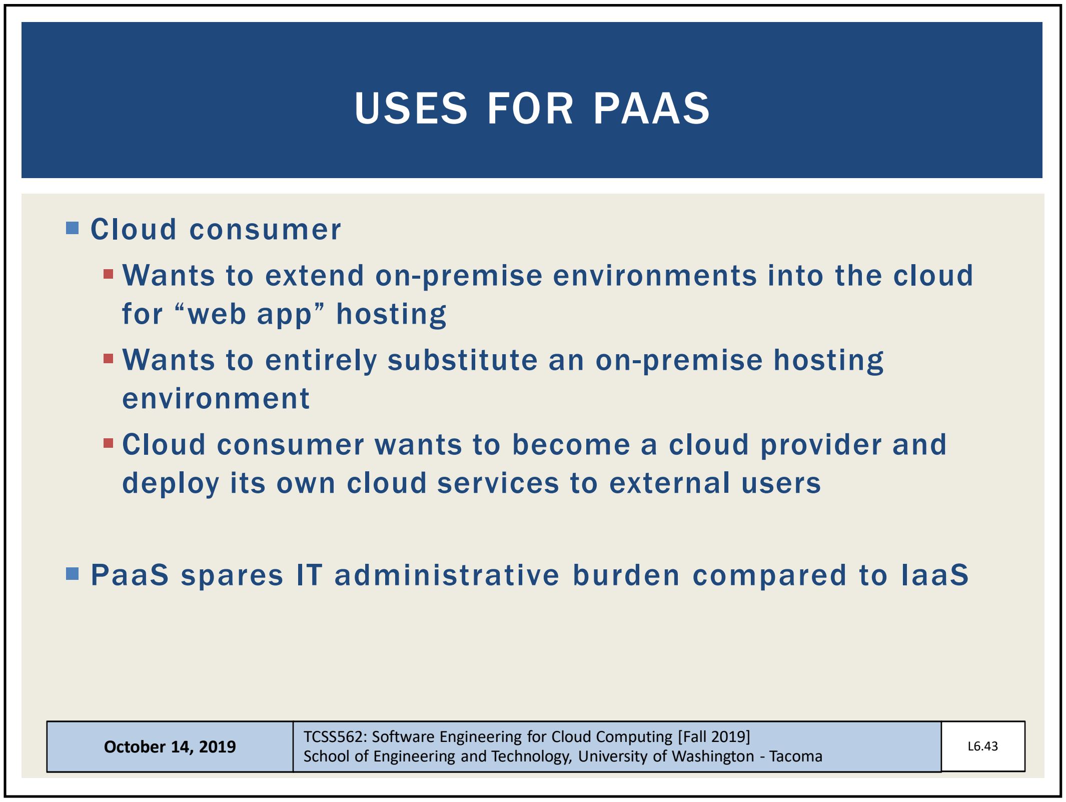


PLATFORM-AS-A-SERVICE

- Predefined, ready-to-use, hosting environment
- Infrastructure is further obscured from end user
- Scaling and load balancing may be automatically provided and automatic
- Variable to no ability to influence responsiveness

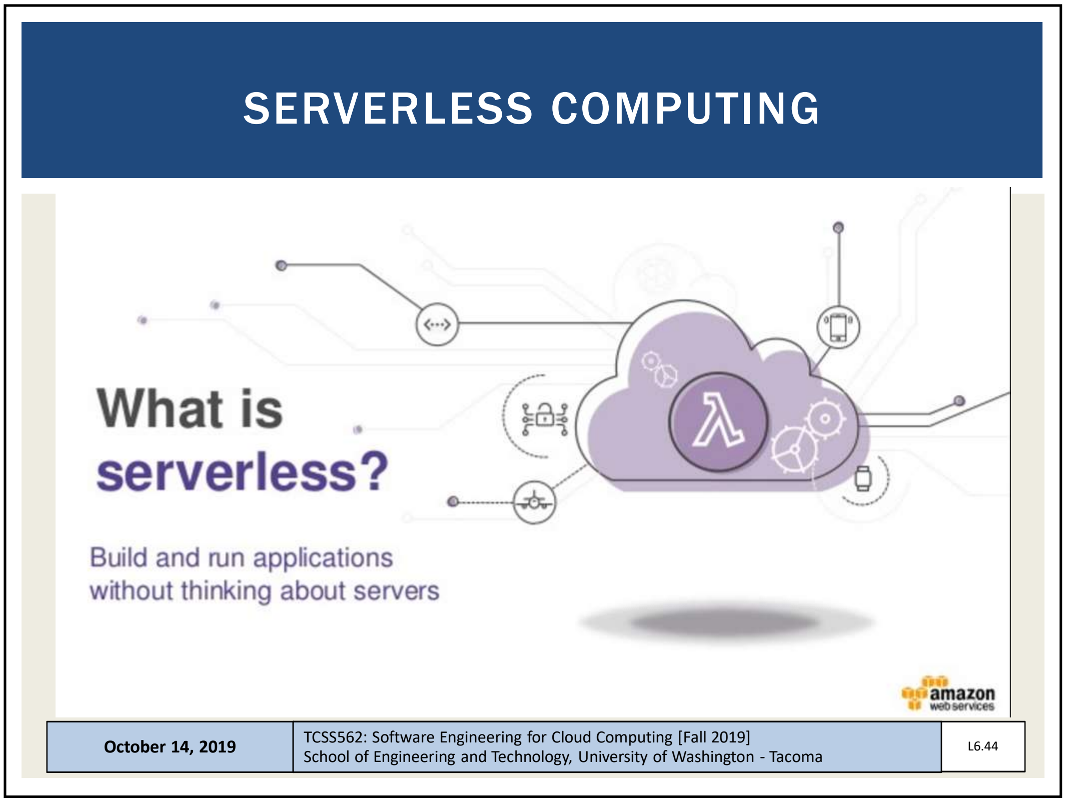
- Examples:
- Google App Engine
- Heroku
- AWS Elastic Beanstalk
- AWS Lambda (FaaS)

42



USES FOR PAAS

- Cloud consumer
  - Wants to extend on-premise environments into the cloud for “web app” hosting
  - Wants to entirely substitute an on-premise hosting environment
  - Cloud consumer wants to become a cloud provider and deploy its own cloud services to external users
- PaaS spares IT administrative burden compared to IaaS

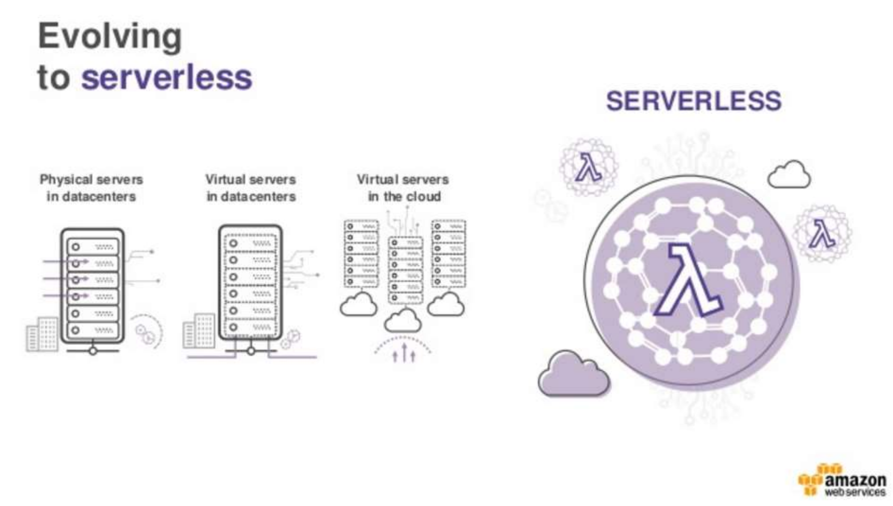


SERVERLESS COMPUTING

What is serverless?

Build and run applications without thinking about servers

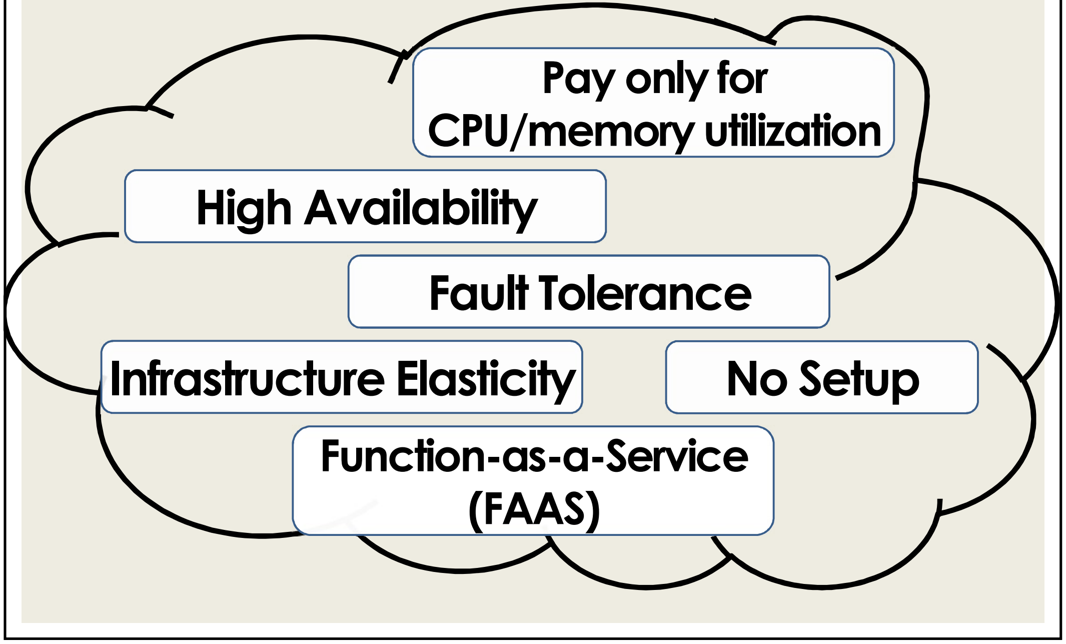
SERVERLESS COMPUTING - 2



October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma | L6.45

45

SERVERLESS COMPUTING



Pay only for CPU/memory utilization

High Availability

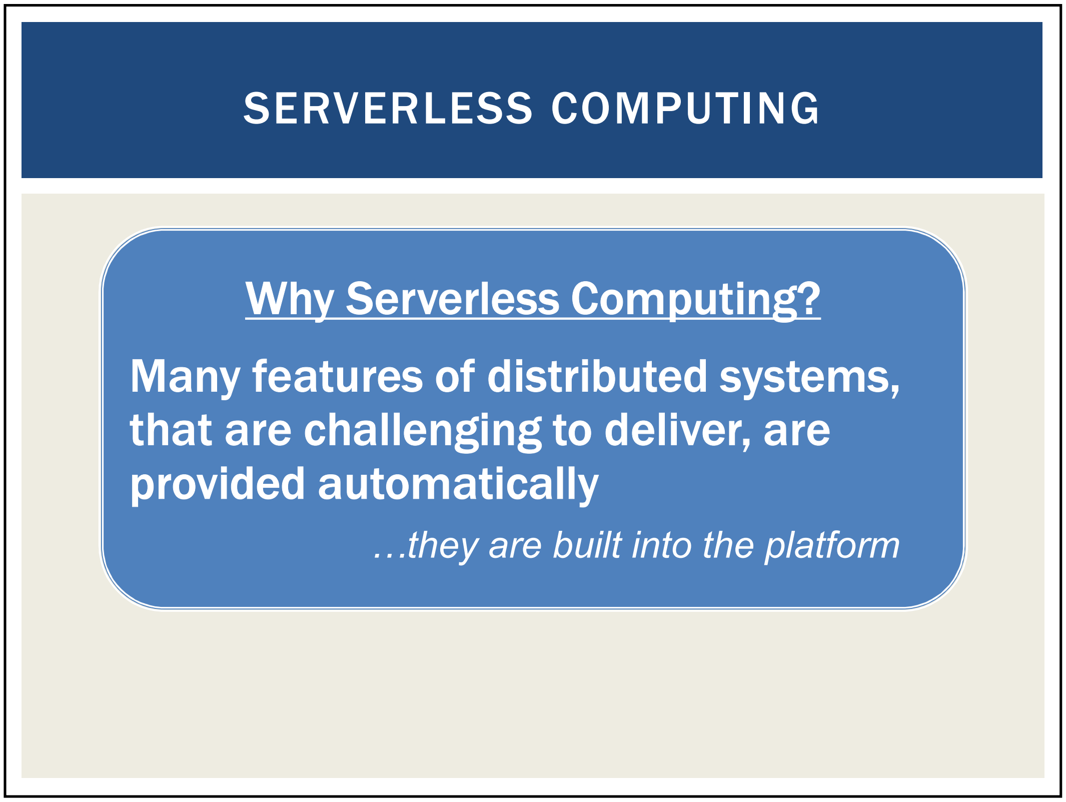
Fault Tolerance

Infrastructure Elasticity

No Setup

Function-as-a-Service (FAAS)

46



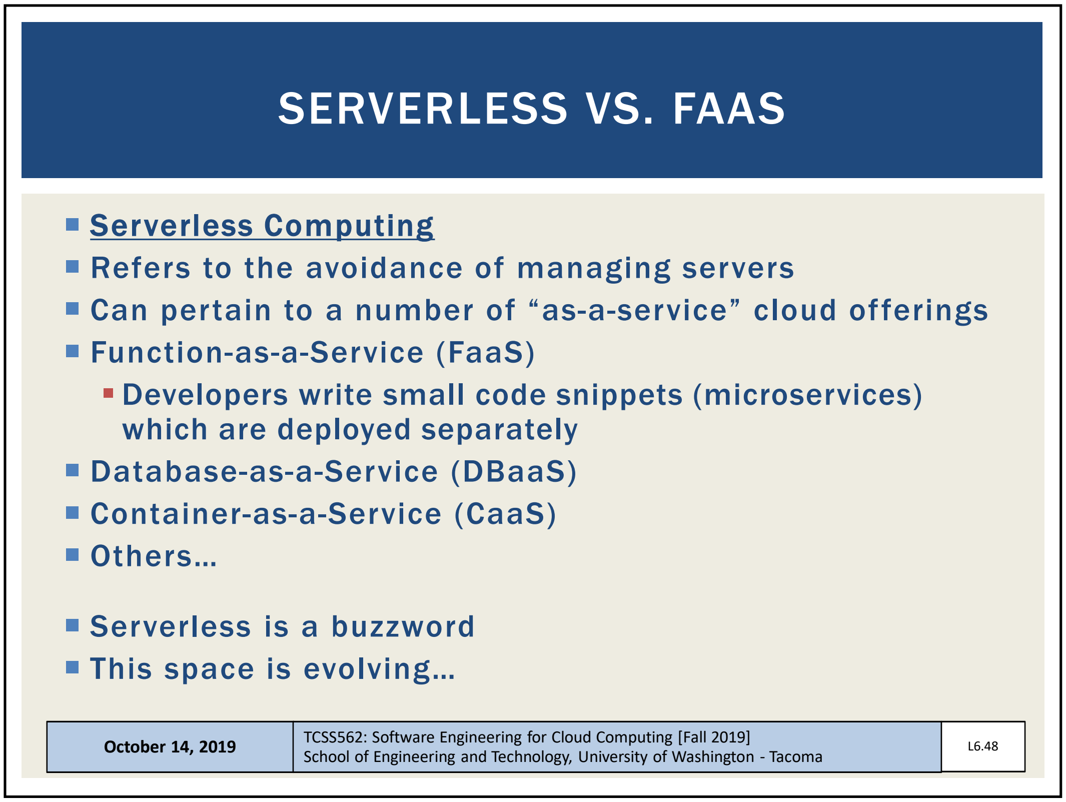
SERVERLESS COMPUTING

Why Serverless Computing?

Many features of distributed systems, that are challenging to deliver, are provided automatically

…they are built into the platform

47



SERVERLESS VS. FAAS

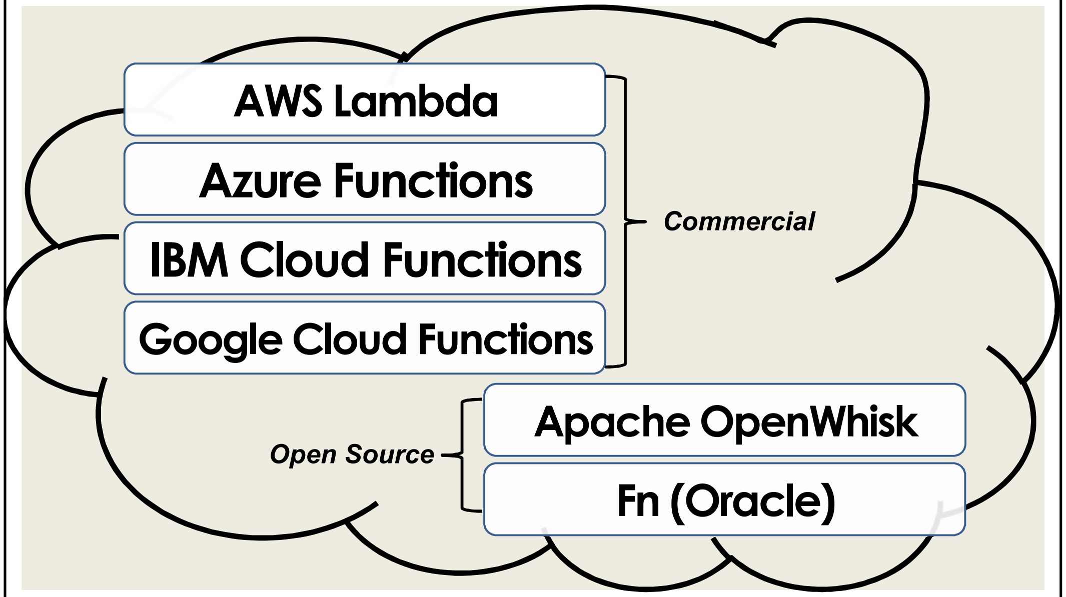
- **Serverless Computing**
- Refers to the avoidance of managing servers
- Can pertain to a number of "as-a-service" cloud offerings
- Function-as-a-Service (FaaS)
  - Developers write small code snippets (microservices) which are deployed separately
- Database-as-a-Service (DBaaS)
- Container-as-a-Service (CaaS)
- Others…

- Serverless is a buzzword
- This space is evolving…

October 14, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L6.48

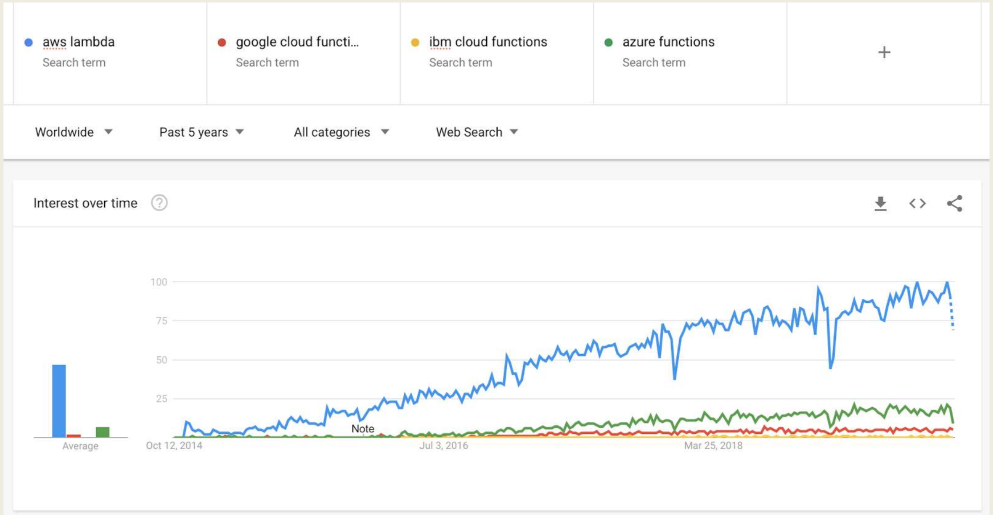
48

FAAS PLATFORMS



49

GOOGLE TRENDS: FAAS PLATFORMS



● aws lambda — Search term
● google cloud functi... — Search term
● ibm cloud functions — Search term
● azure functions — Search term
+

Worldwide ▾ Past 5 years ▾ All categories ▾ Web Search ▾

Interest over time

October 14, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L6.50

50

OPEN SOURCE FAAS FRAMEWORKS

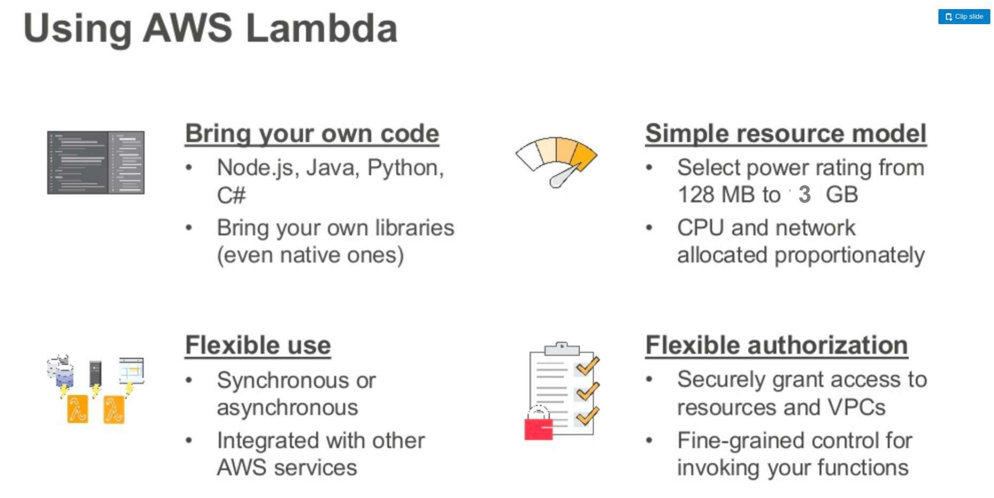
Nuclio graphic

- Deployable to Docker container(s) or a Kubernetes cluster
- Fission: https://fission.io/
- Kubeless: https://kubeless.io/
- Nuclio: https://nuclio.io/
- OpenFaaS: https://www.openfaas.com/

- Supports cloud native development principles
- Building a cloud application by adopting a “deploy it yourself” framework avoids vendor lock-in
- Requires common medium of Kubernetes

51

AWS LAMBDA



Using AWS Lambda

Bring your own code
- Node.js, Java, Python, C#
- Bring your own libraries (even native ones)

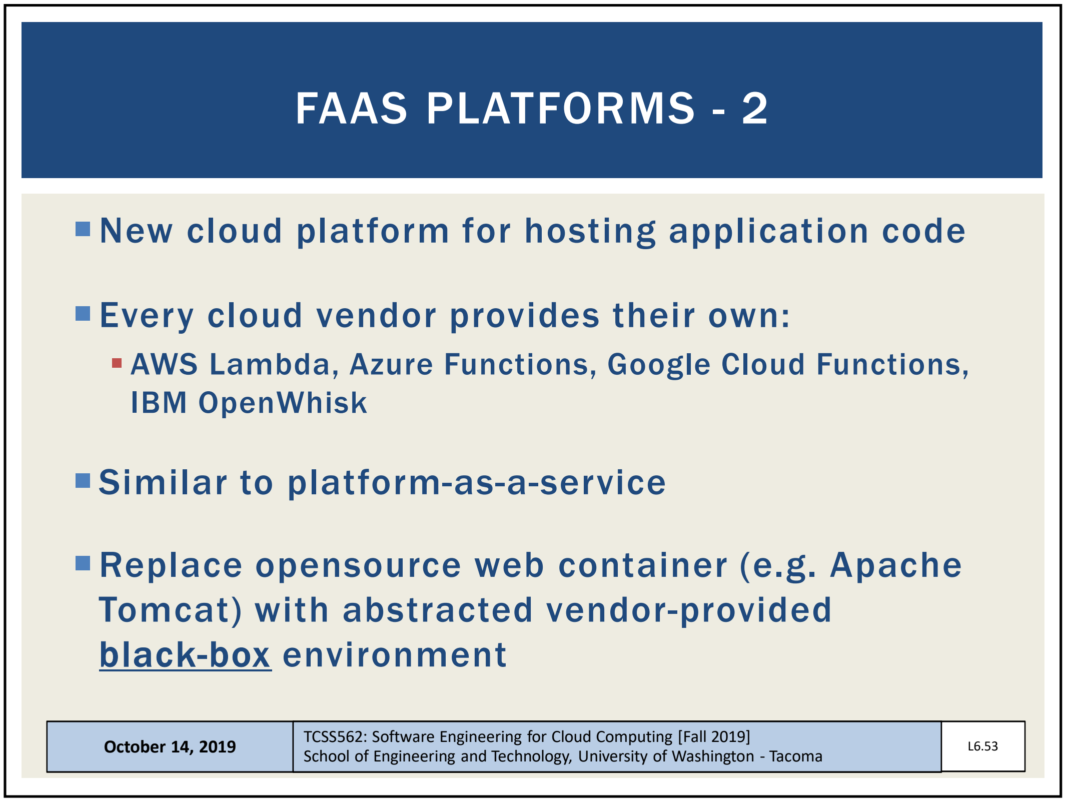
Simple resource model
- Select power rating from 128 MB to 3 GB
- CPU and network allocated proportionately

Flexible use
- Synchronous or asynchronous
- Integrated with other AWS services

Flexible authorization
- Securely grant access to resources and VPCs
- Fine-grained control for invoking your functions

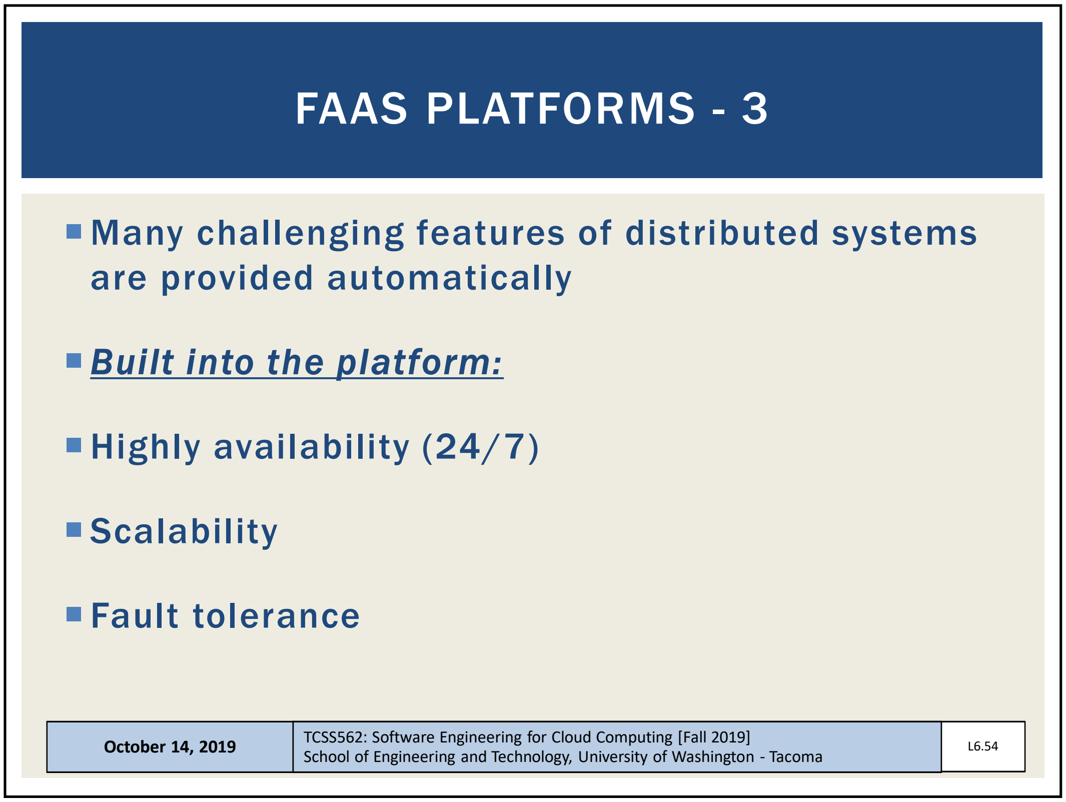
Images credit: aws.amazon.com

52



FAAS PLATFORMS - 2

- New cloud platform for hosting application code
- Every cloud vendor provides their own:
  - AWS Lambda, Azure Functions, Google Cloud Functions, IBM OpenWhisk
- Similar to platform-as-a-service
- Replace opensource web container (e.g. Apache Tomcat) with abstracted vendor-provided <u>black-box</u> environment

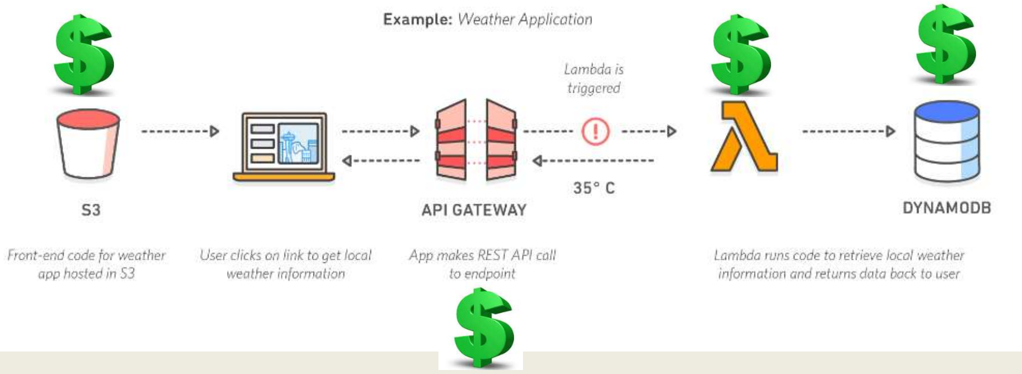


FAAS PLATFORMS - 3

- Many challenging features of distributed systems are provided automatically
- <u>*Built into the platform:*</u>
- Highly availability (24/7)
- Scalability
- Fault tolerance

CLOUD NATIVE SOFTWARE ARCHITECTURE

- Every service with a different pricing model

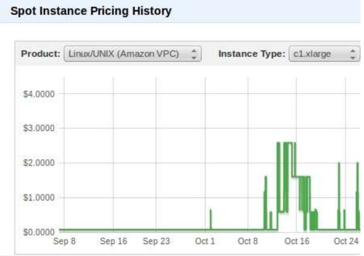

Example: Weather Application
$
S3
Front-end code for weather app hosted in S3
User clicks on link to get local weather information
API GATEWAY
App makes REST API call to endpoint
$
Lambda is triggered
35° C
$
Lambda runs code to retrieve local weather information and returns data back to user
$
DYNAMODB

IAAS BILLING MODELS

- Virtual machines as-a-service at ¢ per hour
- No premium to scale:

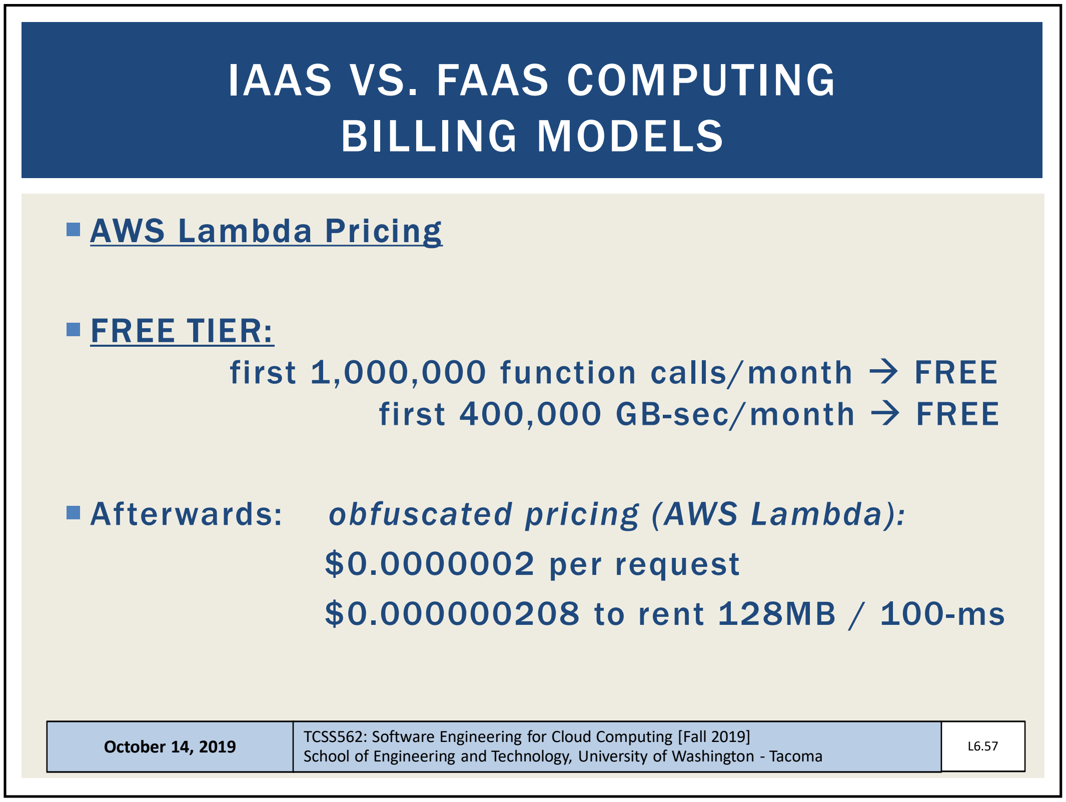
```
        1000 computers   @     1 hour
=          1 computer    @  1000 hours
```

- Illusion of infinite scalability to cloud user
- As many computers as you can afford
- Billing models are becoming increasingly granular
  - By the minute, second, 1/10th sec
- Auction-based instances: Spot instances →


Spot Instance Pricing History

Product: Linux/UNIX (Amazon VPC)

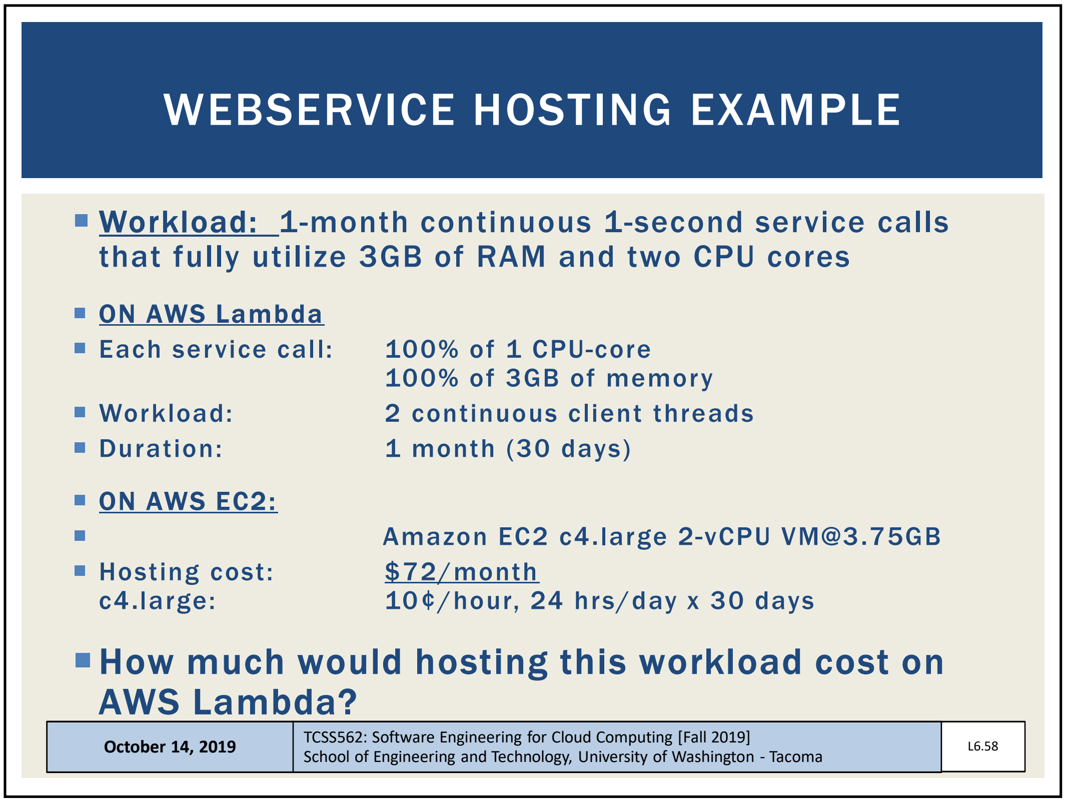
Instance Type: c1.xlarge



IAAS VS. FAAS COMPUTING BILLING MODELS

- **AWS Lambda Pricing**
- **FREE TIER:**
  first 1,000,000 function calls/month → FREE
  first 400,000 GB-sec/month → FREE
- Afterwards: *obfuscated pricing (AWS Lambda):*
  $0.0000002 per request
  $0.000000208 to rent 128MB / 100-ms

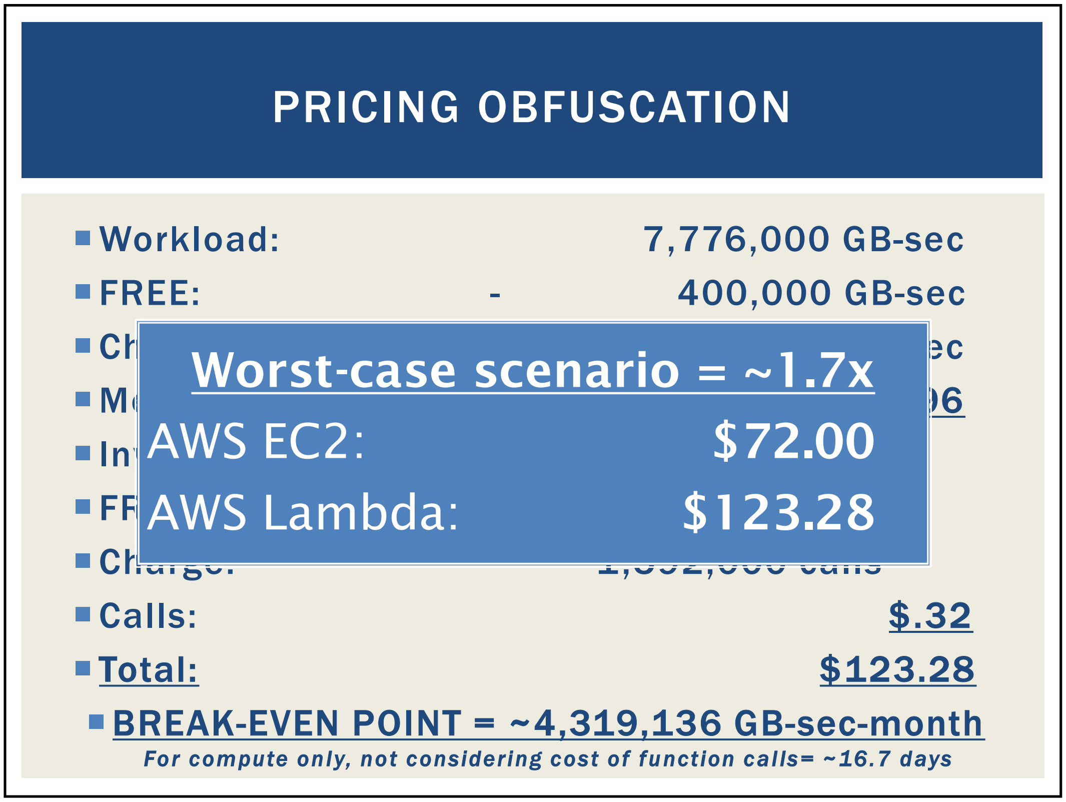
57



WEBSERVICE HOSTING EXAMPLE

- **Workload:** 1-month continuous 1-second service calls that fully utilize 3GB of RAM and two CPU cores
- **ON AWS Lambda**
- Each service call: 100% of 1 CPU-core
  100% of 3GB of memory
- Workload: 2 continuous client threads
- Duration: 1 month (30 days)
- **ON AWS EC2:**
- Amazon EC2 c4.large 2-vCPU VM@3.75GB
- Hosting cost: $72/month
  c4.large: 10¢/hour, 24 hrs/day x 30 days
- **How much would hosting this workload cost on AWS Lambda?**

58



PRICING OBFUSCATION

- Workload: 7,776,000 GB-sec
- FREE: - 400,000 GB-sec
- Ch ec
- M 96
- In
- FR
- Charge: 1,392,000 calls
- Calls: $.32
- **Total:** **$123.28**
  - **BREAK-EVEN POINT = ~4,319,136 GB-sec-month**

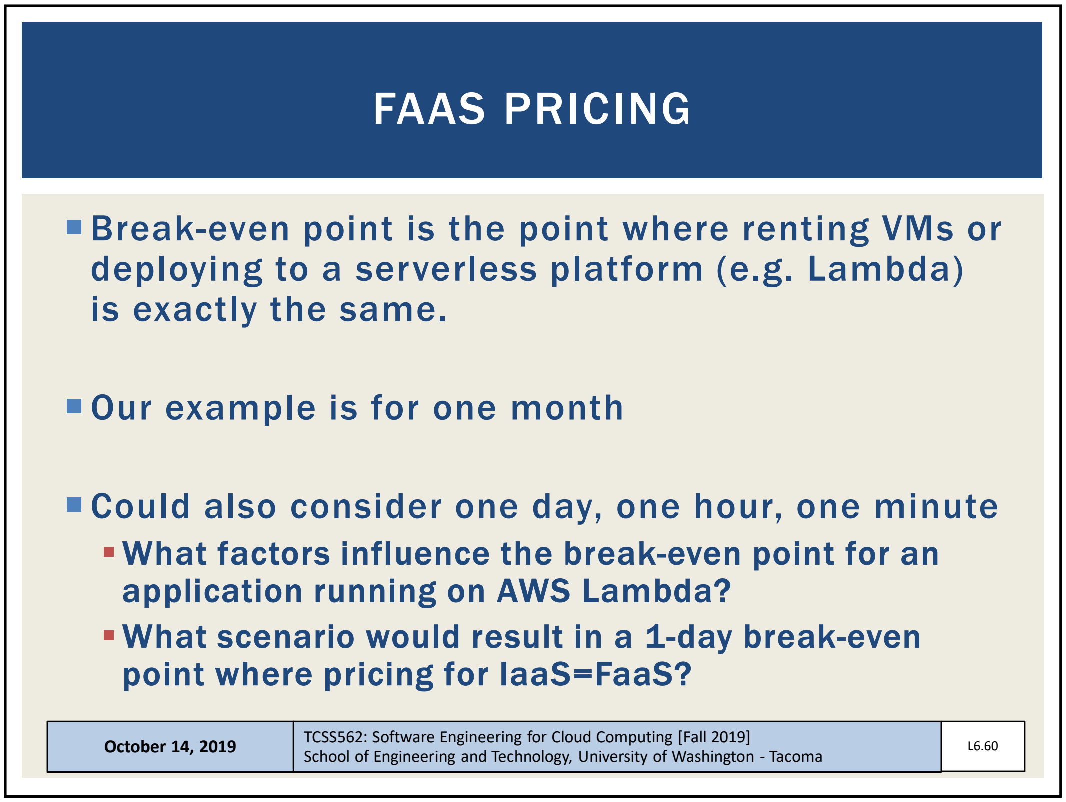
    For compute only, not considering cost of function calls= ~16.7 days

Worst-case scenario = ~1.7x

AWS EC2: **$72.00**

AWS Lambda: **$123.28**

59

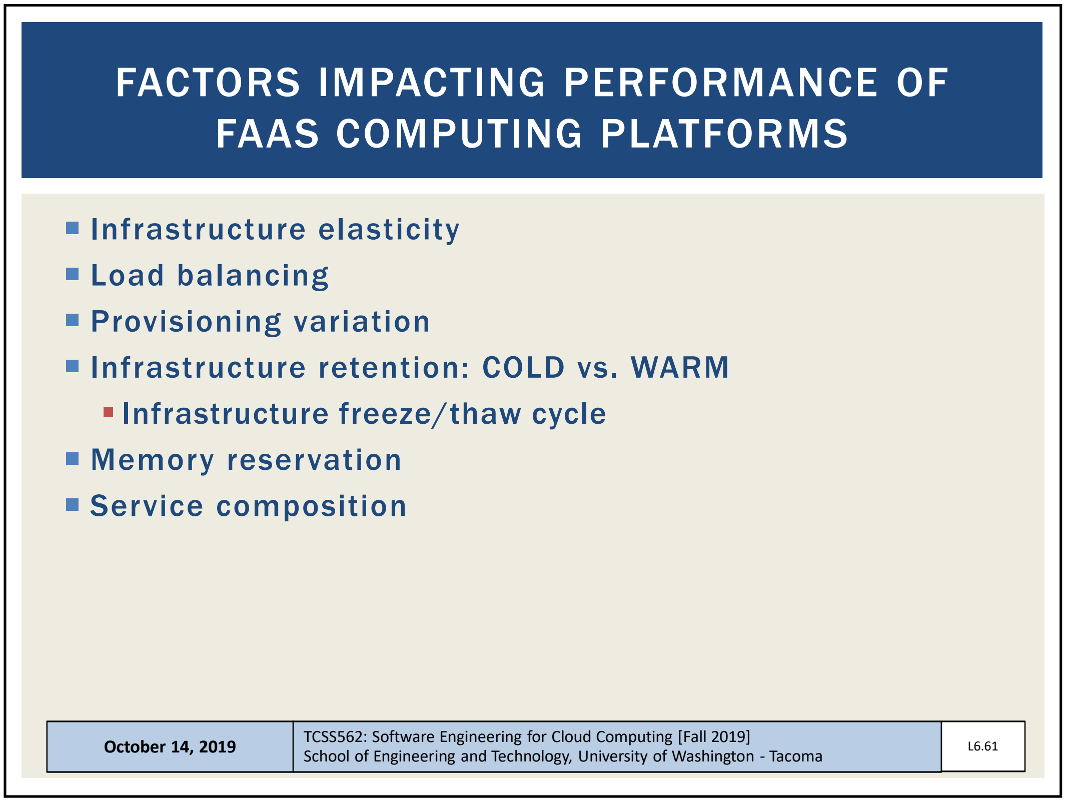


FAAS PRICING

- Break-even point is the point where renting VMs or deploying to a serverless platform (e.g. Lambda) is exactly the same.
- Our example is for one month
- Could also consider one day, one hour, one minute
  - What factors influence the break-even point for an application running on AWS Lambda?
  - What scenario would result in a 1-day break-even point where pricing for IaaS=FaaS?

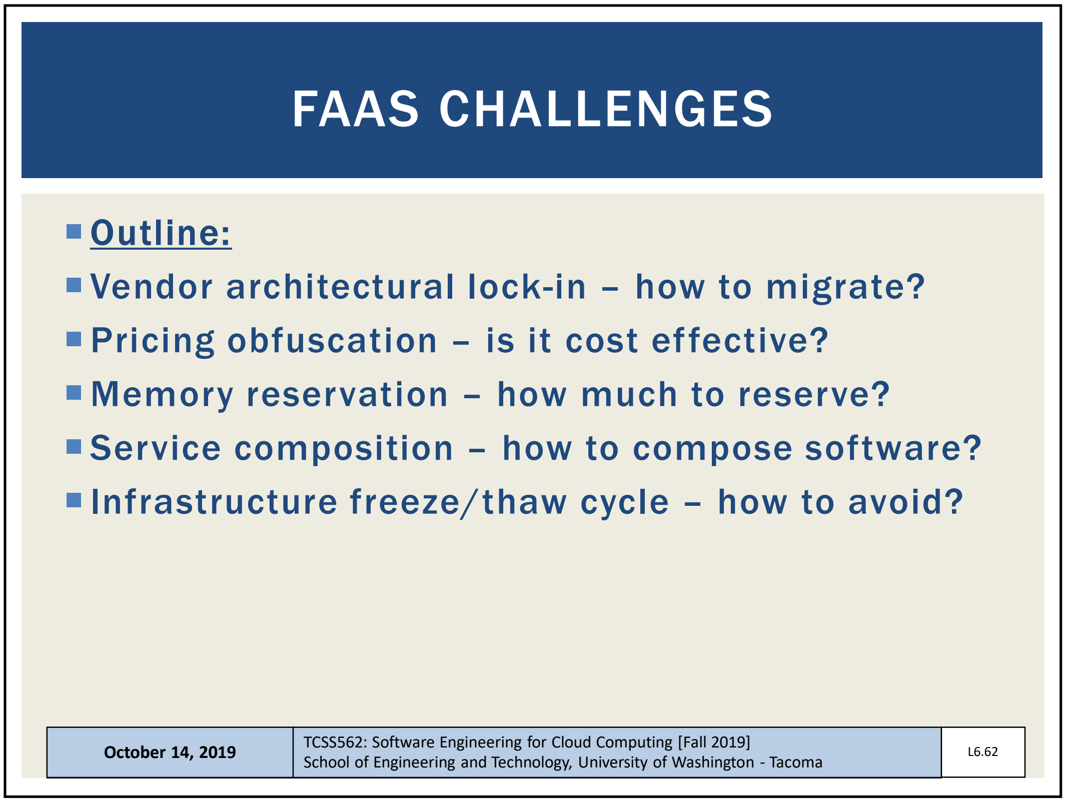
October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L6.60

60



FACTORS IMPACTING PERFORMANCE OF FAAS COMPUTING PLATFORMS

- Infrastructure elasticity
- Load balancing
- Provisioning variation
- Infrastructure retention: COLD vs. WARM
  - Infrastructure freeze/thaw cycle
- Memory reservation
- Service composition

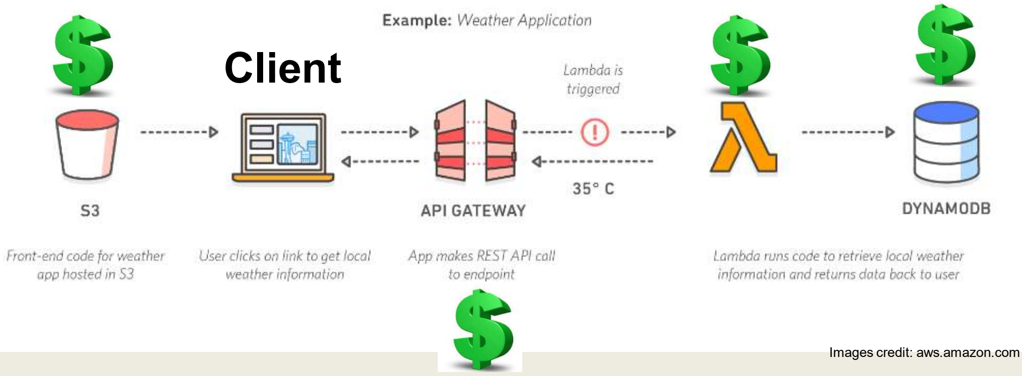


FAAS CHALLENGES

- <u>Outline:</u>
- Vendor architectural lock-in – how to migrate?
- Pricing obfuscation – is it cost effective?
- Memory reservation – how much to reserve?
- Service composition – how to compose software?
- Infrastructure freeze/thaw cycle – how to avoid?

VENDOR ARCHITECTURAL LOCK-IN

- Cloud native (FaaS) software architecture requires external services/components


Example: Weather Application
Client
Lambda is triggered
35° C
S3
API GATEWAY
DYNAMODB
Front-end code for weather app hosted in S3
User clicks on link to get local weather information
App makes REST API call to endpoint
Lambda runs code to retrieve local weather information and returns data back to user

Images credit: aws.amazon.com

- Increased dependencies → increased hosting costs

63

PRICING OBFUSCATION

- **VM pricing:** hourly rental pricing, billed to nearest second is intuitive…
- **FaaS pricing:**

AWS Lambda Pricing

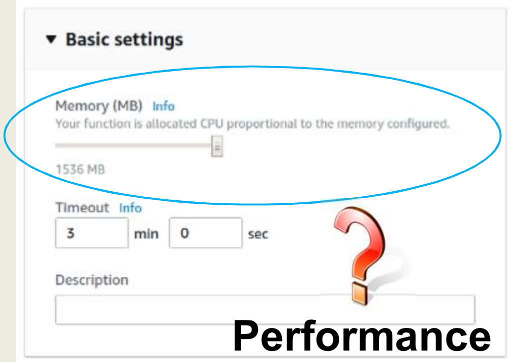
FREE TIER: first 1,000,000 function calls/month → FREE
first 400 GB-sec/month → FREE

- Afterwards: $0.0000002 per request
$0.000000208 to rent 128MB / 100-ms

64

MEMORY RESERVATION QUEST

- Lambda memory reserved for functions
- UI provides “slider bar” to set function’s memory allocation
- Resource capacity (CPU, disk, network) coupled to slider bar: *“every doubling of memory, doubles CPU…”*



▾ Basic settings

Memory (MB) Info
Your function is allocated CPU proportional to the memory configured.
1536 MB

Timeout Info
3 min 0 sec

Description

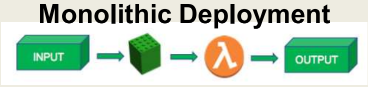
Performance

- **But how much memory do model services require?**

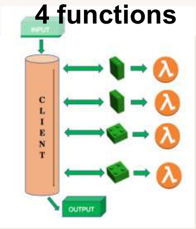
SERVICE COMPOSITION

- **How should application code be composed for deployment to serverless computing platforms?**

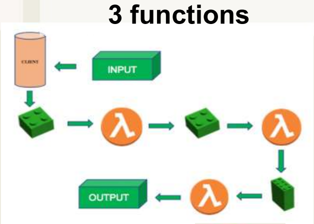
Monolithic Deployment



Client flow control, 4 functions



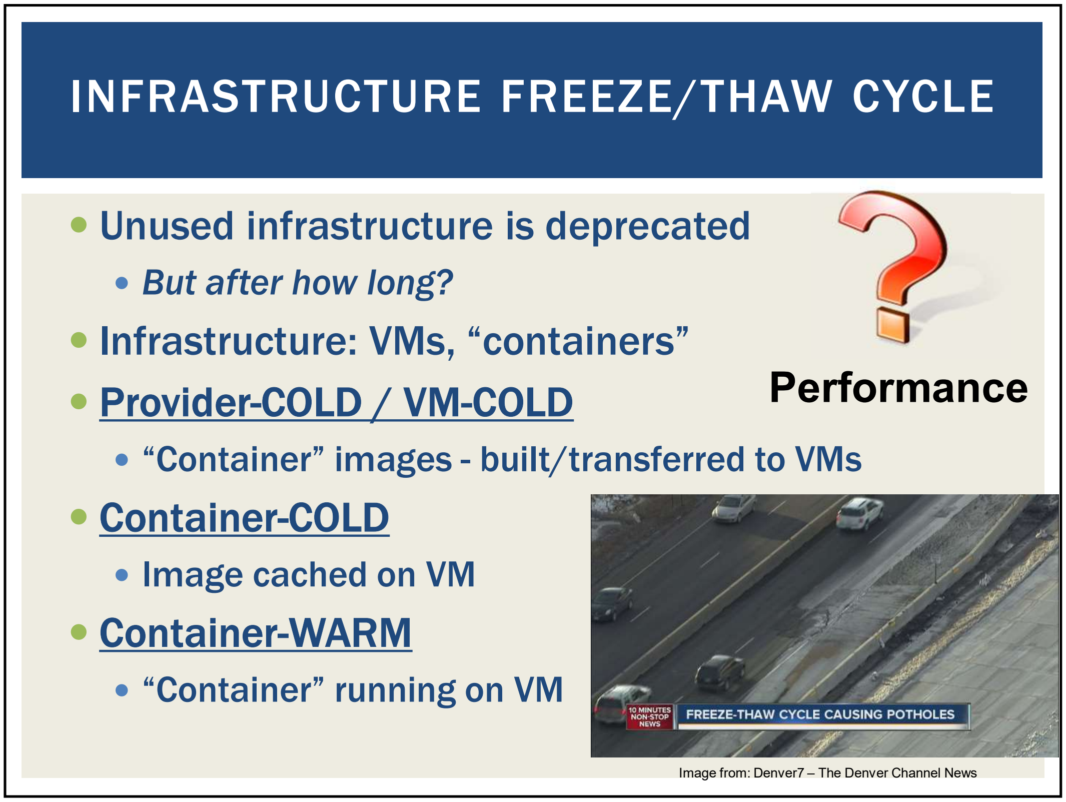
Server flow control, 3 functions



- Recommended practice: Decompose into many microservices
- Platform limits: code + libraries ~250MB

Performance

- How does composition impact the number of function invocations, and memory utilization?



INFRASTRUCTURE FREEZE/THAW CYCLE

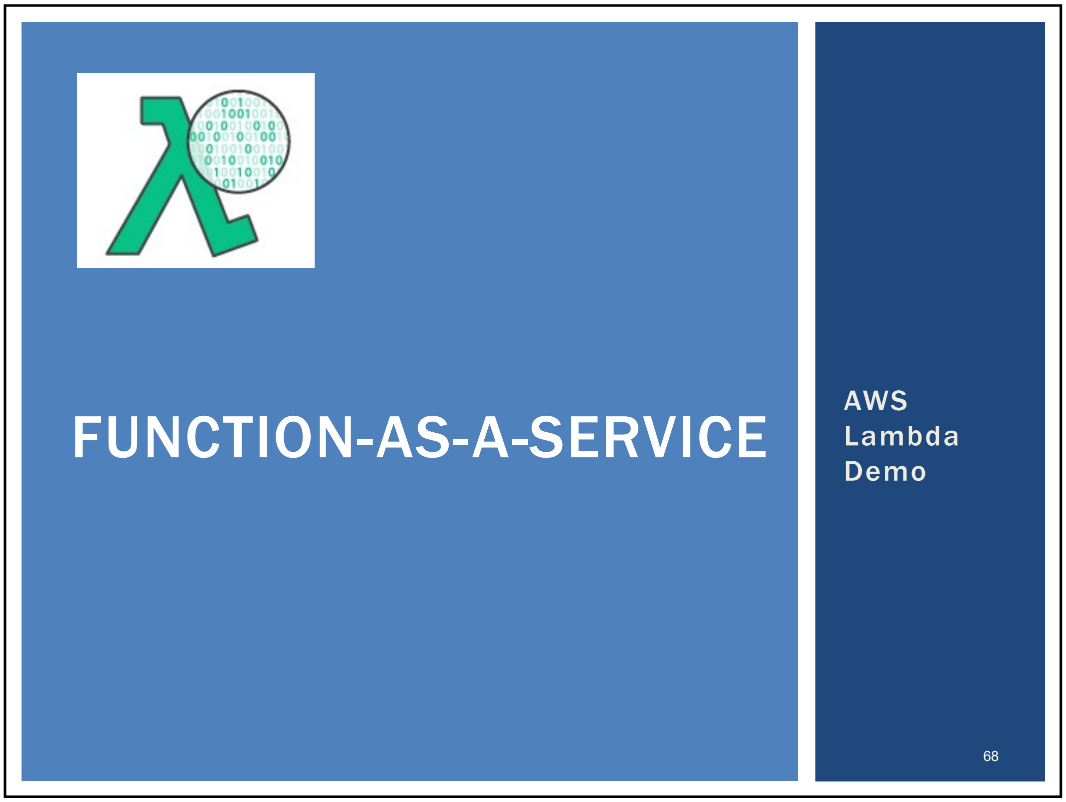
- Unused infrastructure is deprecated
  - *But after how long?*
- Infrastructure: VMs, “containers”
- Provider-COLD / VM-COLD
  - “Container” images - built/transferred to VMs
- Container-COLD
  - Image cached on VM
- Container-WARM
  - “Container” running on VM

Performance

10 MINUTES NON-STOP NEWS
FREEZE-THAW CYCLE CAUSING POTHOLES

Image from: Denver7 – The Denver Channel News

67

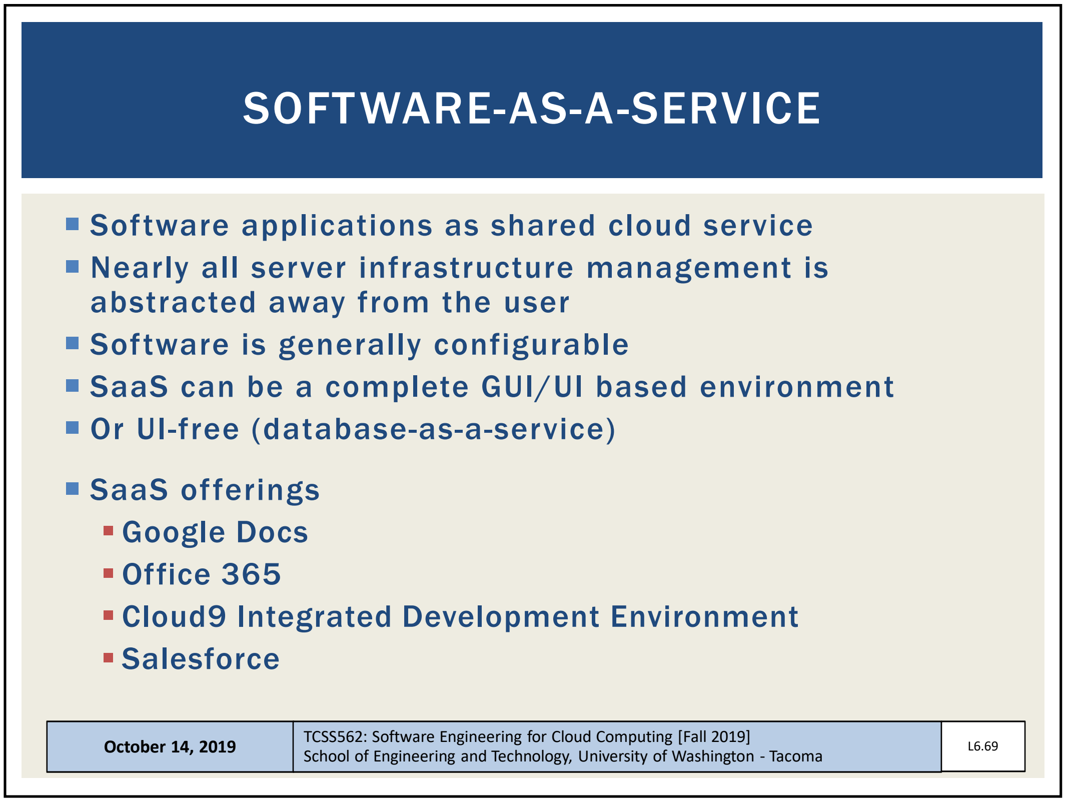


FUNCTION-AS-A-SERVICE

AWS
Lambda
Demo

68

68

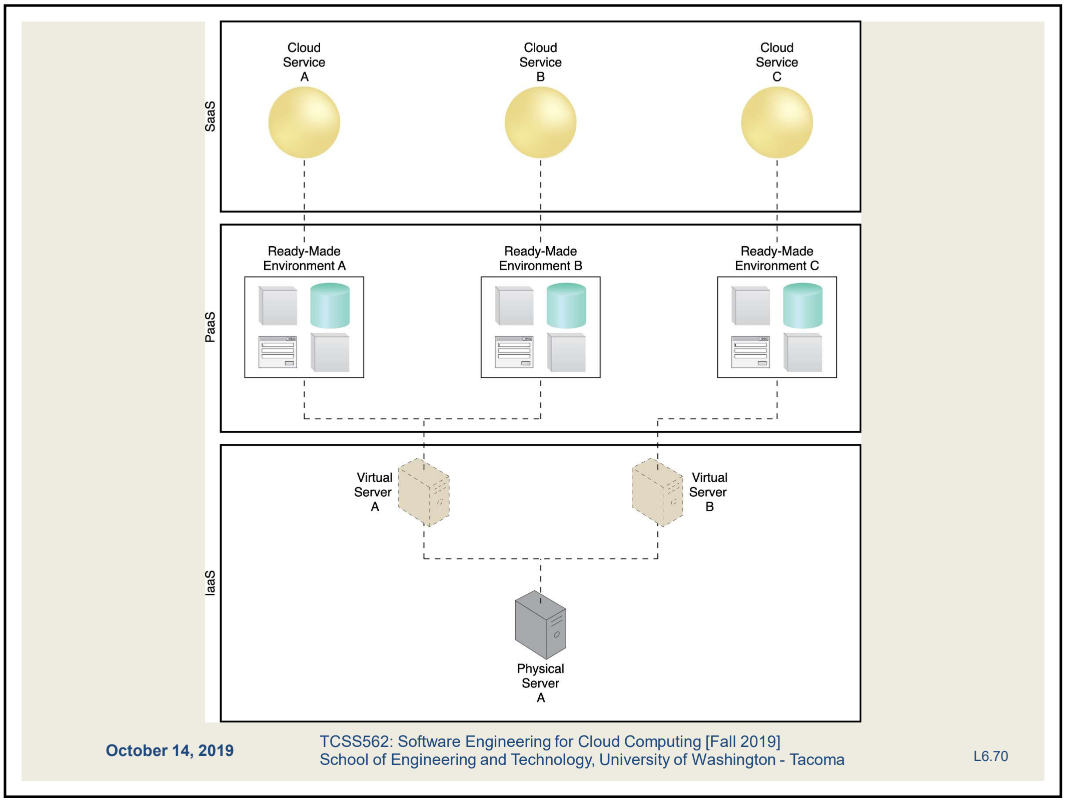


SOFTWARE-AS-A-SERVICE

- Software applications as shared cloud service
- Nearly all server infrastructure management is abstracted away from the user
- Software is generally configurable
- SaaS can be a complete GUI/UI based environment
- Or UI-free (database-as-a-service)
- SaaS offerings
  - Google Docs
  - Office 365
  - Cloud9 Integrated Development Environment
  - Salesforce

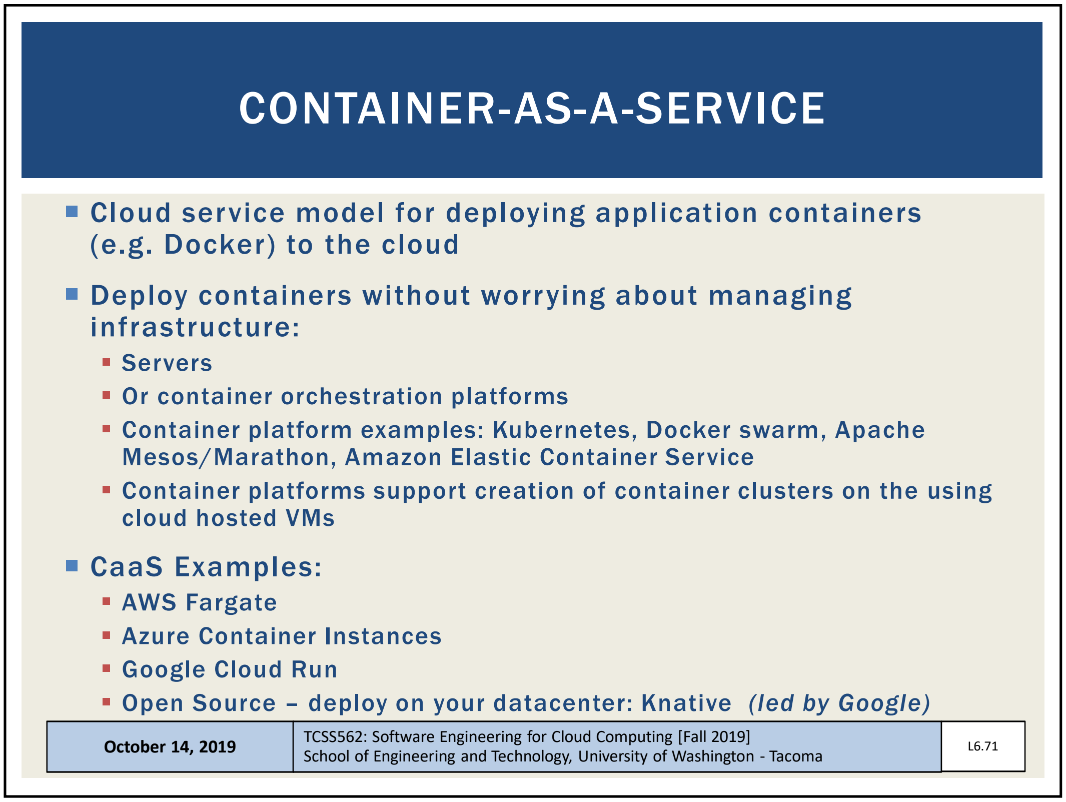
October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma | L6.69

69



October 14, 2019
TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma
L6.70

70

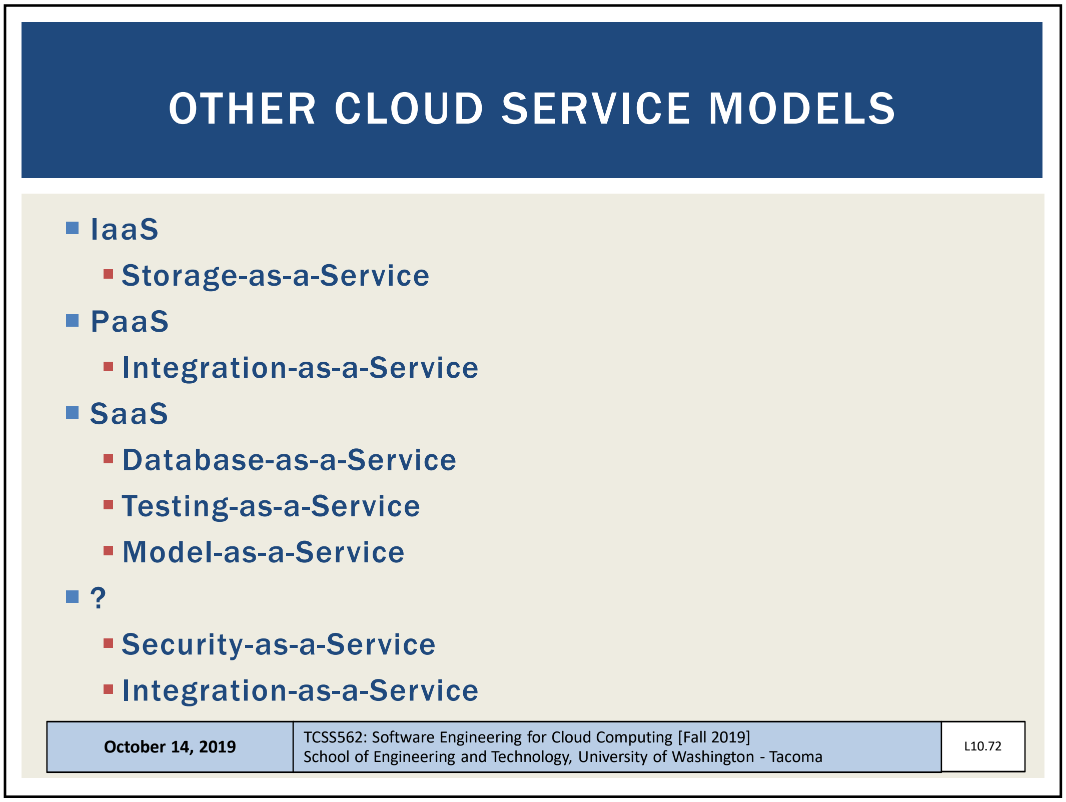


CONTAINER-AS-A-SERVICE

- Cloud service model for deploying application containers (e.g. Docker) to the cloud
- Deploy containers without worrying about managing infrastructure:
  - Servers
  - Or container orchestration platforms
  - Container platform examples: Kubernetes, Docker swarm, Apache Mesos/Marathon, Amazon Elastic Container Service
  - Container platforms support creation of container clusters on the using cloud hosted VMs
- CaaS Examples:
  - AWS Fargate
  - Azure Container Instances
  - Google Cloud Run
  - Open Source – deploy on your datacenter: Knative *(led by Google)*

October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma | L6.71

71

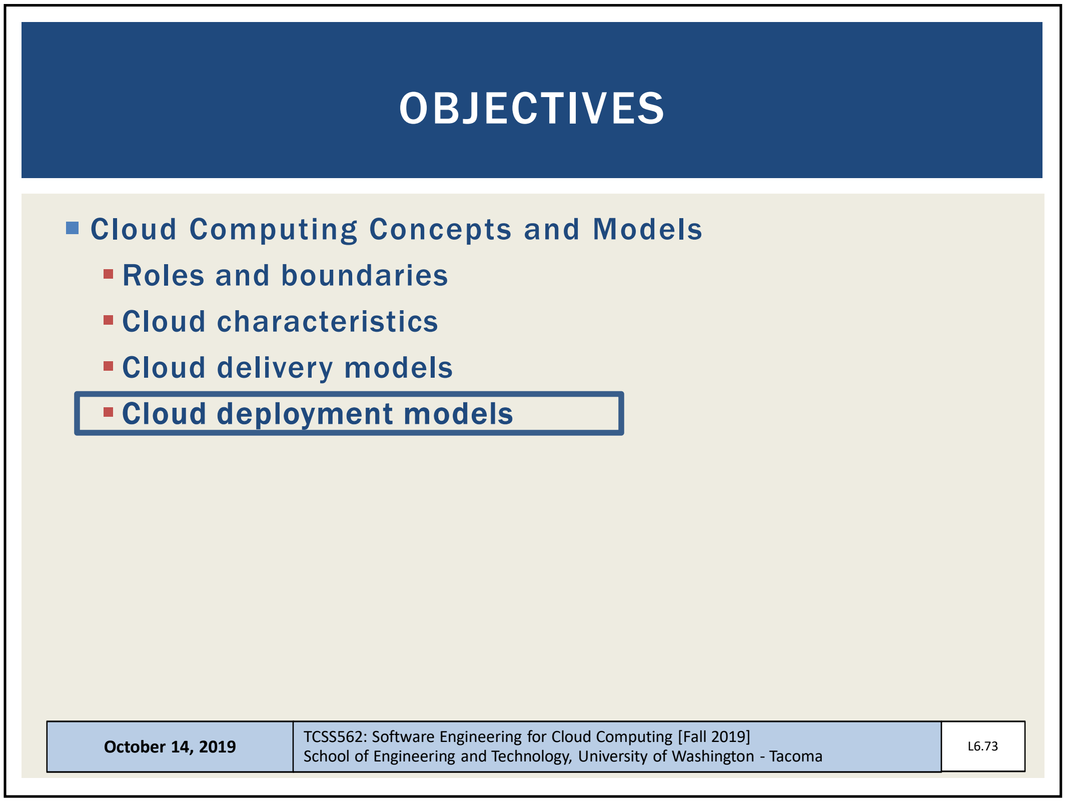


OTHER CLOUD SERVICE MODELS

- IaaS
  - Storage-as-a-Service
- PaaS
  - Integration-as-a-Service
- SaaS
  - Database-as-a-Service
  - Testing-as-a-Service
  - Model-as-a-Service
- ?
  - Security-as-a-Service
  - Integration-as-a-Service

October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma | L10.72

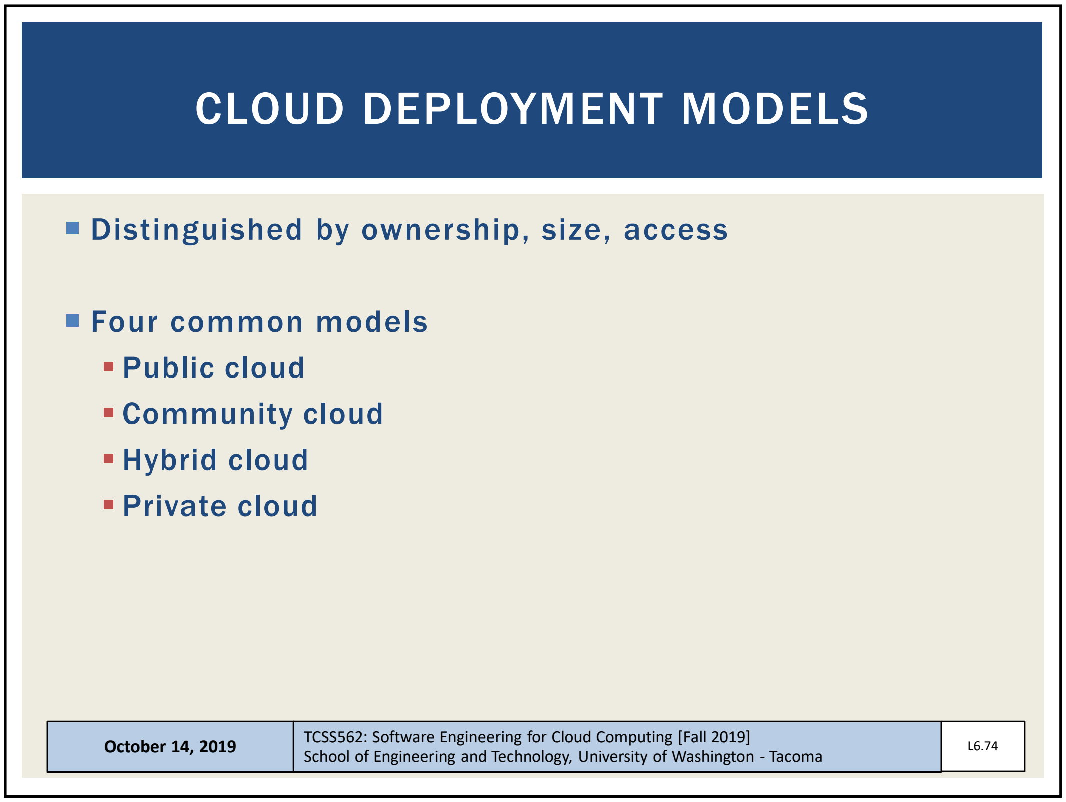
72



OBJECTIVES

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

73

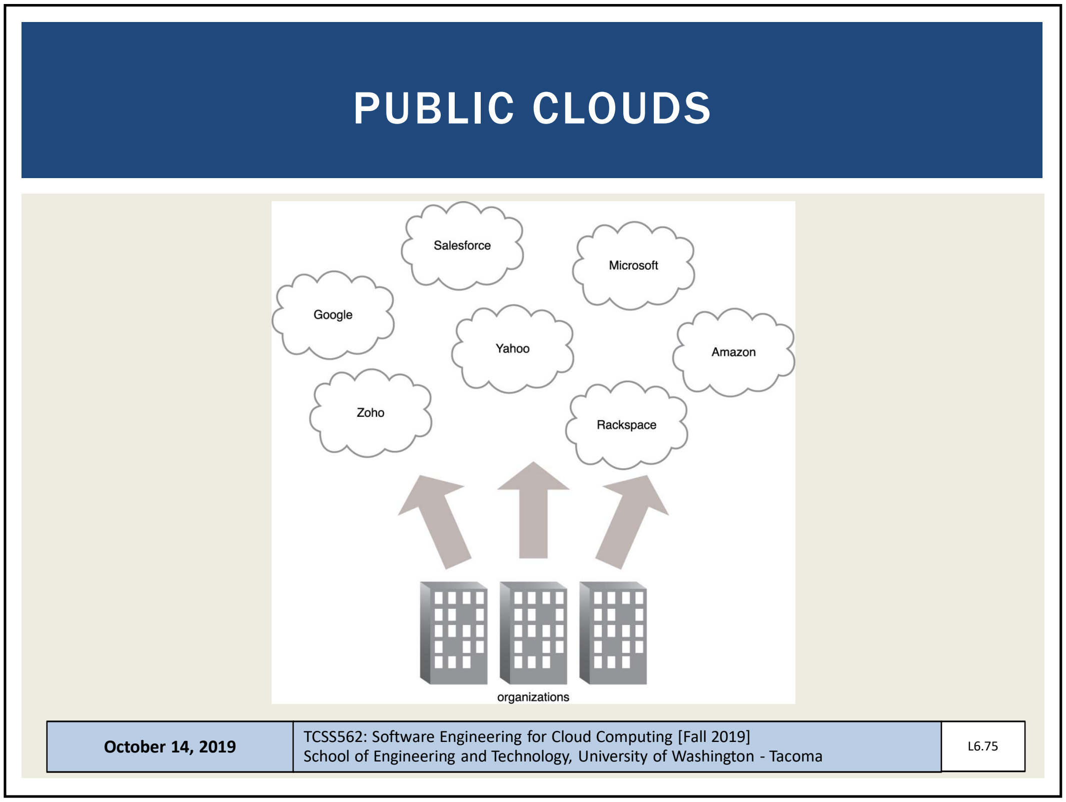


CLOUD DEPLOYMENT MODELS

- Distinguished by ownership, size, access

- Four common models
  - Public cloud
  - Community cloud
  - Hybrid cloud
  - Private cloud

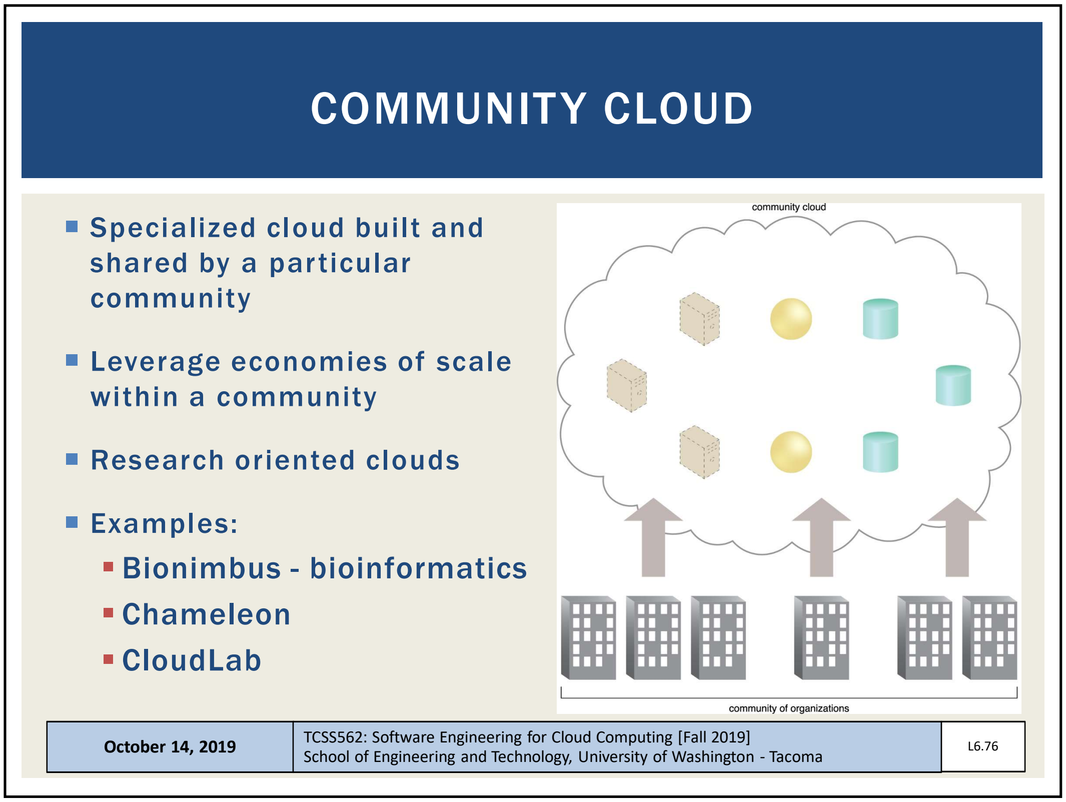
74



PUBLIC CLOUDS

October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L6.75

75

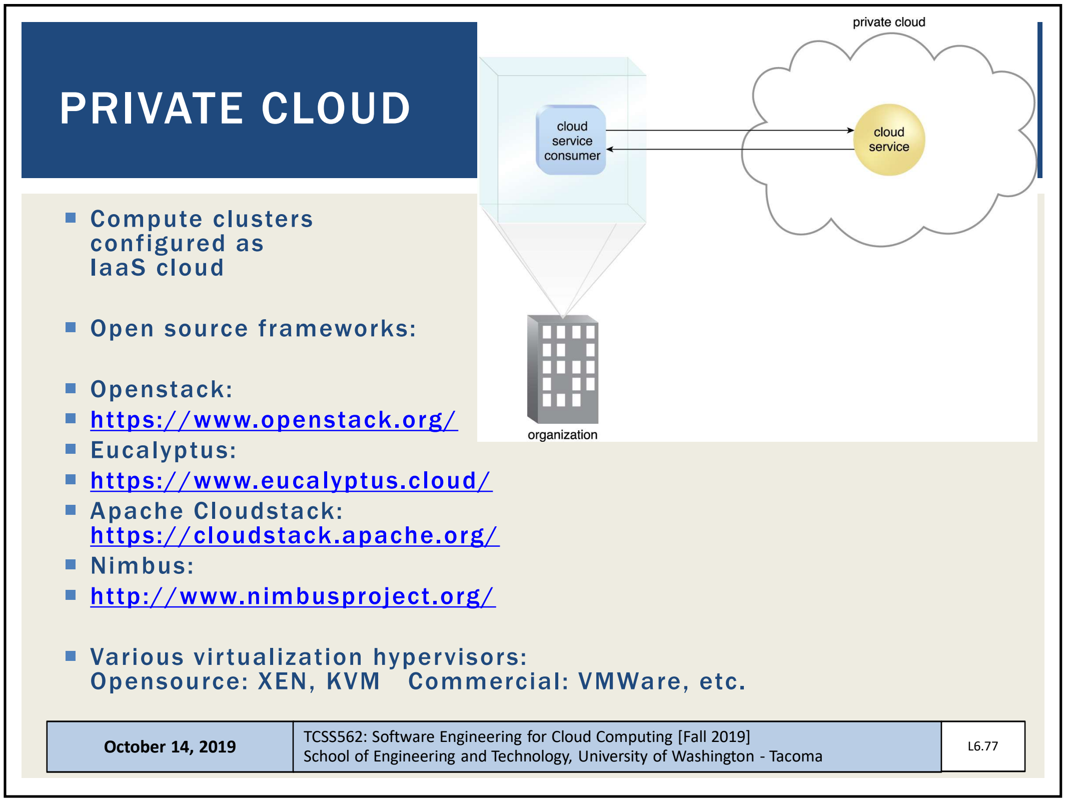


COMMUNITY CLOUD

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

October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L6.76

76

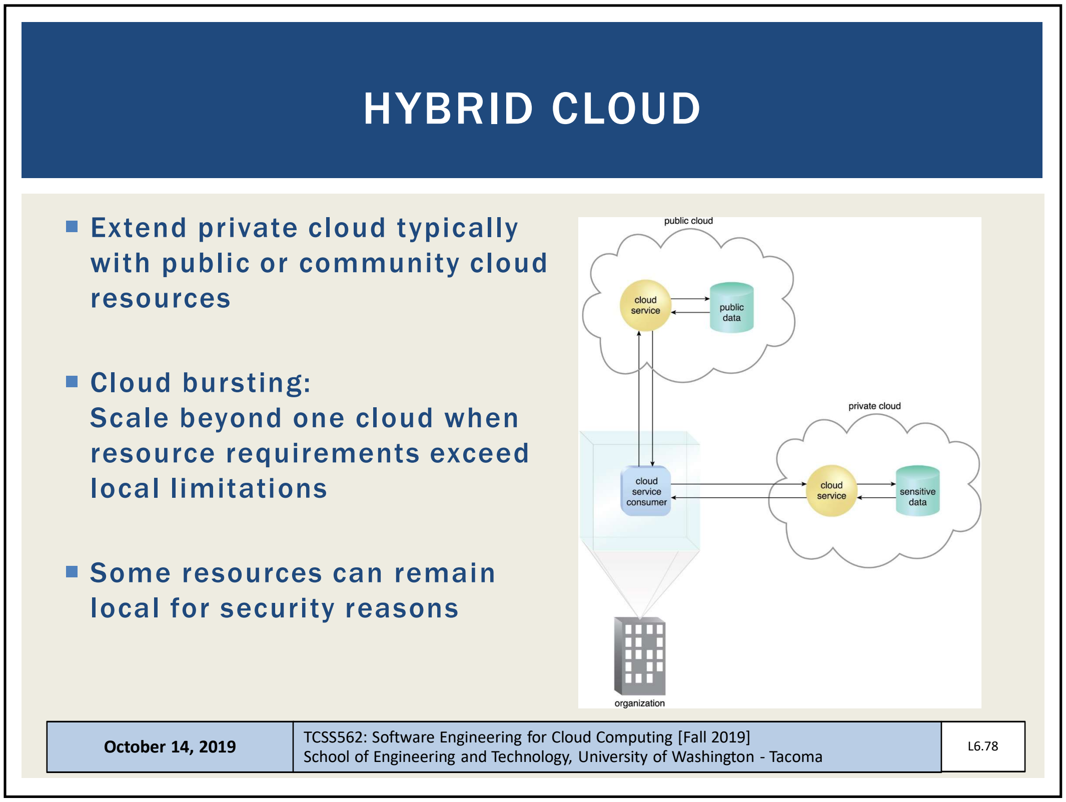


PRIVATE CLOUD

- 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.

October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L6.77

77

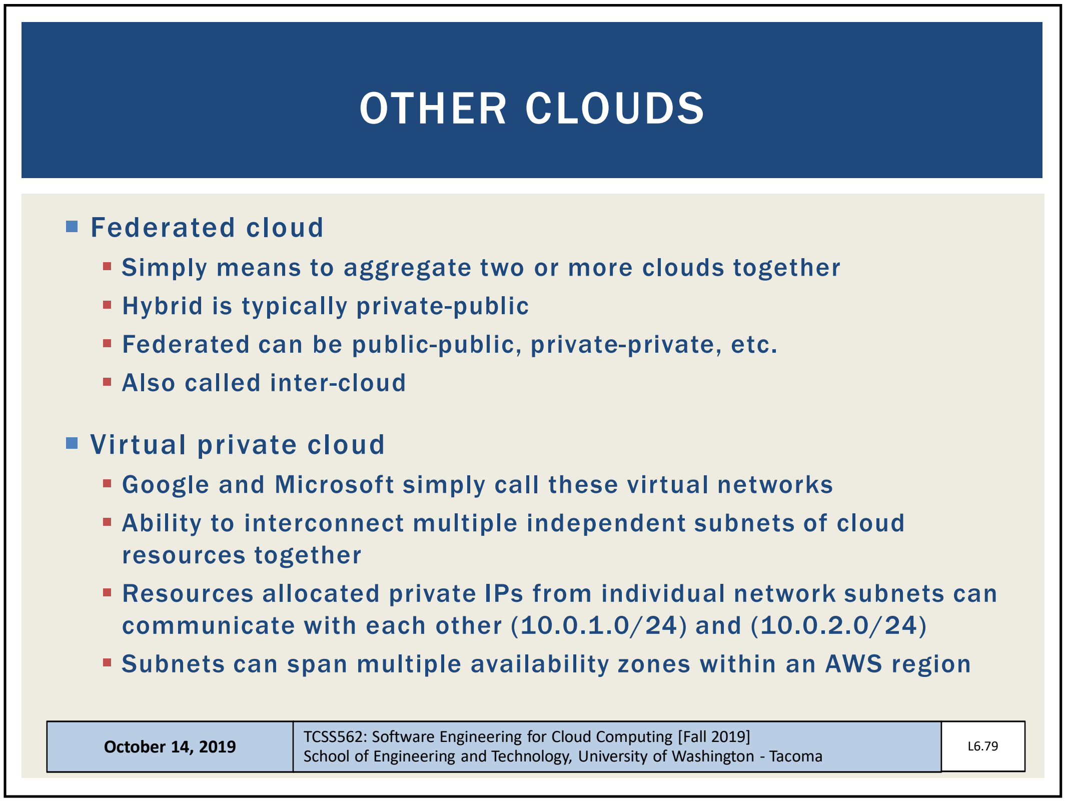


HYBRID CLOUD

- 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

October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L6.78

78

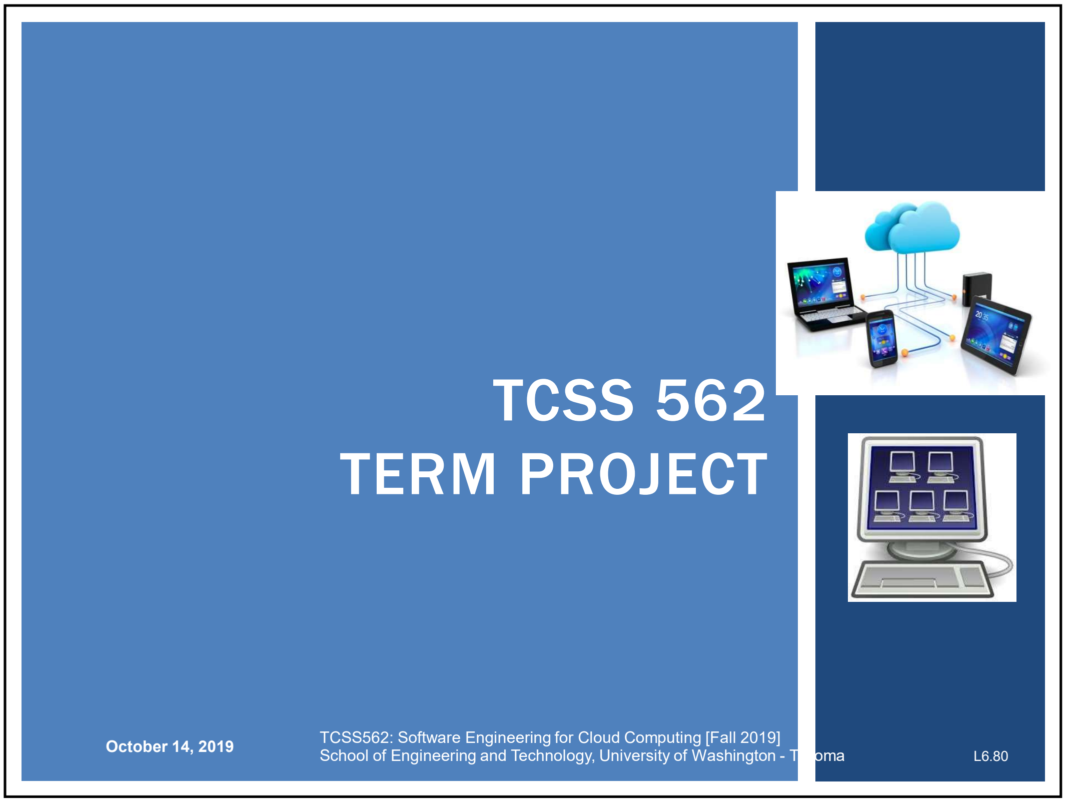


OTHER CLOUDS

- 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

October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L6.79

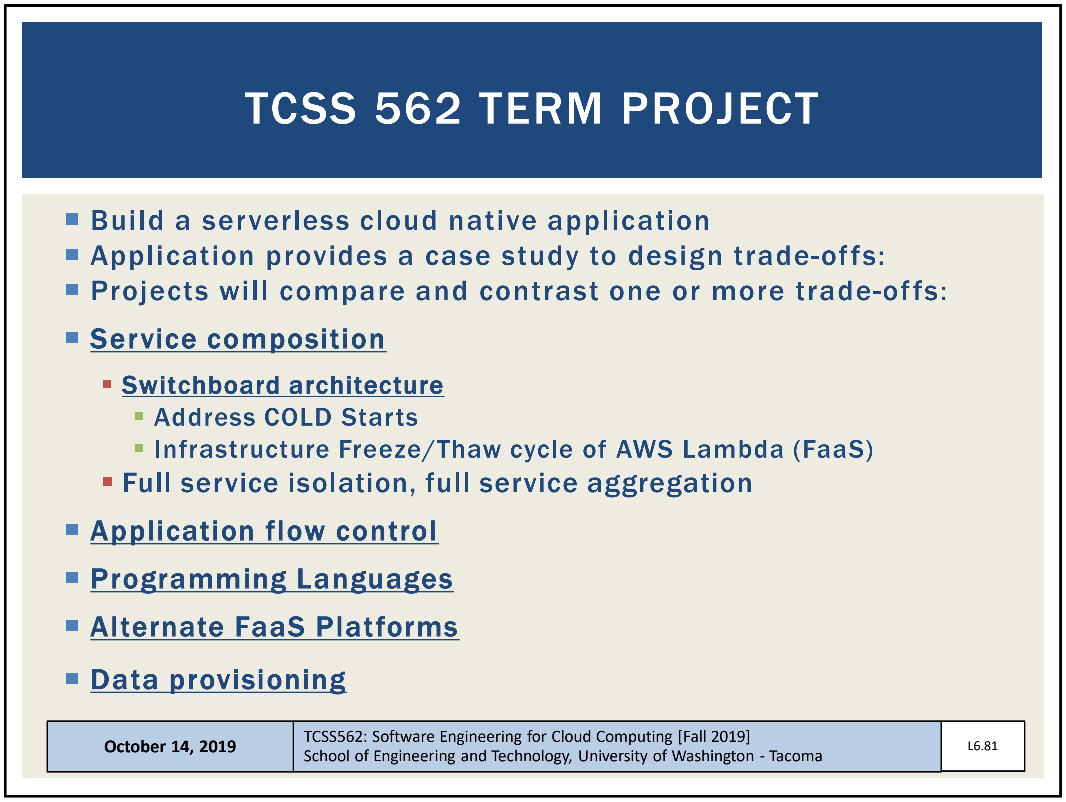
79



TCSS 562
TERM PROJECT

October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - T oma | L6.80

80

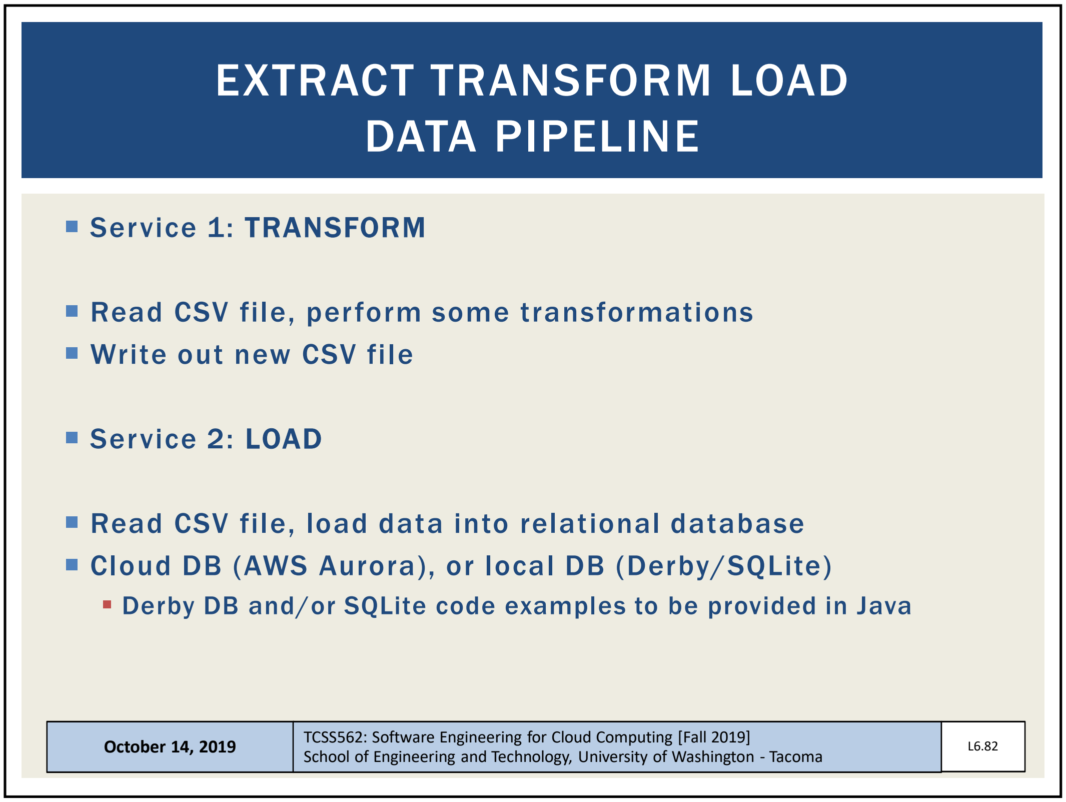


TCSS 562 TERM PROJECT

- 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 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma | L6.81

81



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

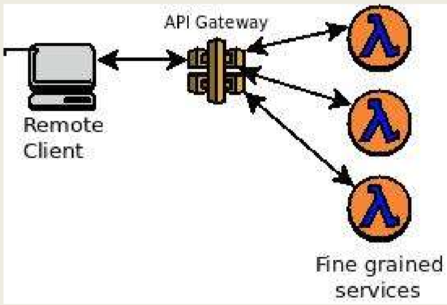
October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma | L6.82

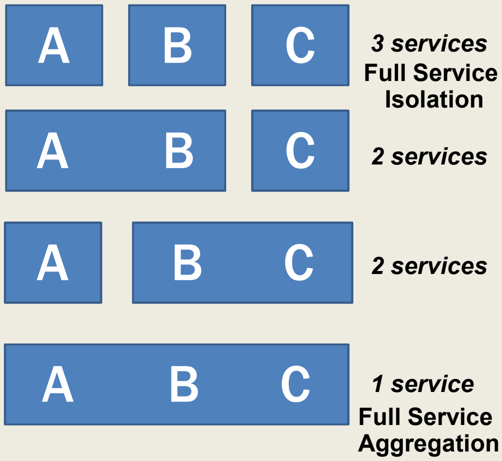
82

EXTRACT TRANSFORM LOAD DATA PIPELINE 2

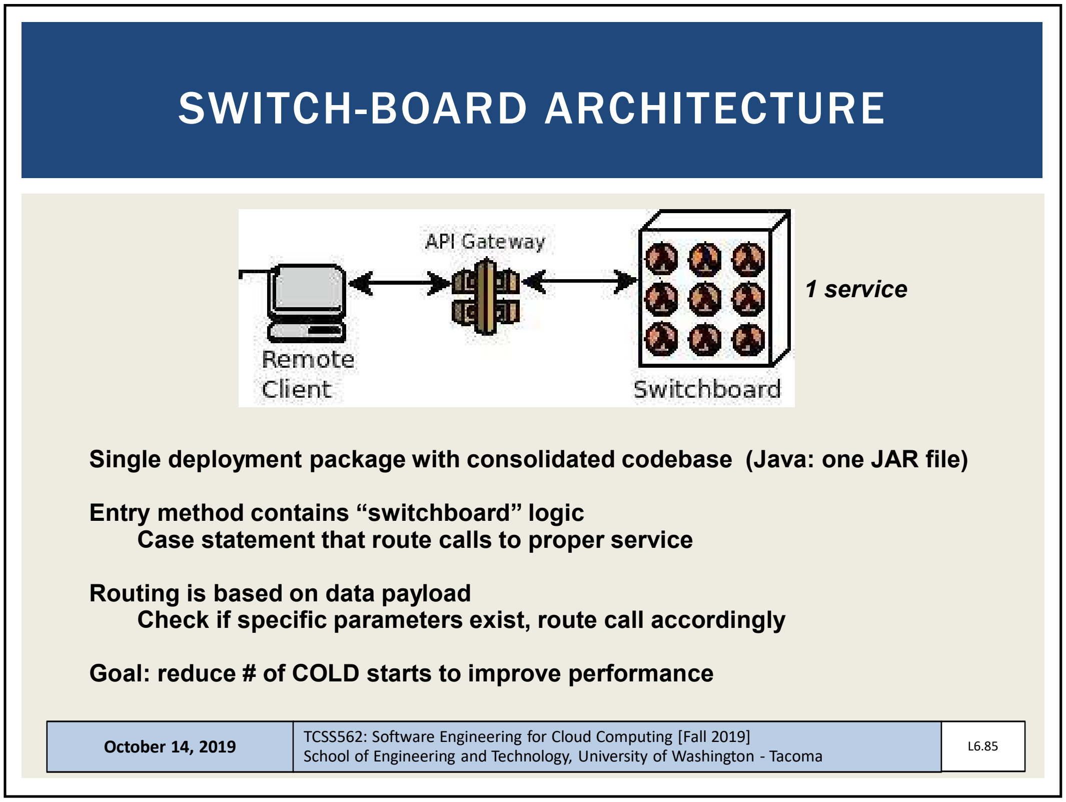
- Service 3: EXTRACT
- Using relational database, apply filter(s) and/or functions to aggregate data to produce sums, totals, averages
- Output aggregations as JSON

SERVICE COMPOSITION





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



SWITCH-BOARD ARCHITECTURE

API Gateway
Remote Client
Switchboard
1 service

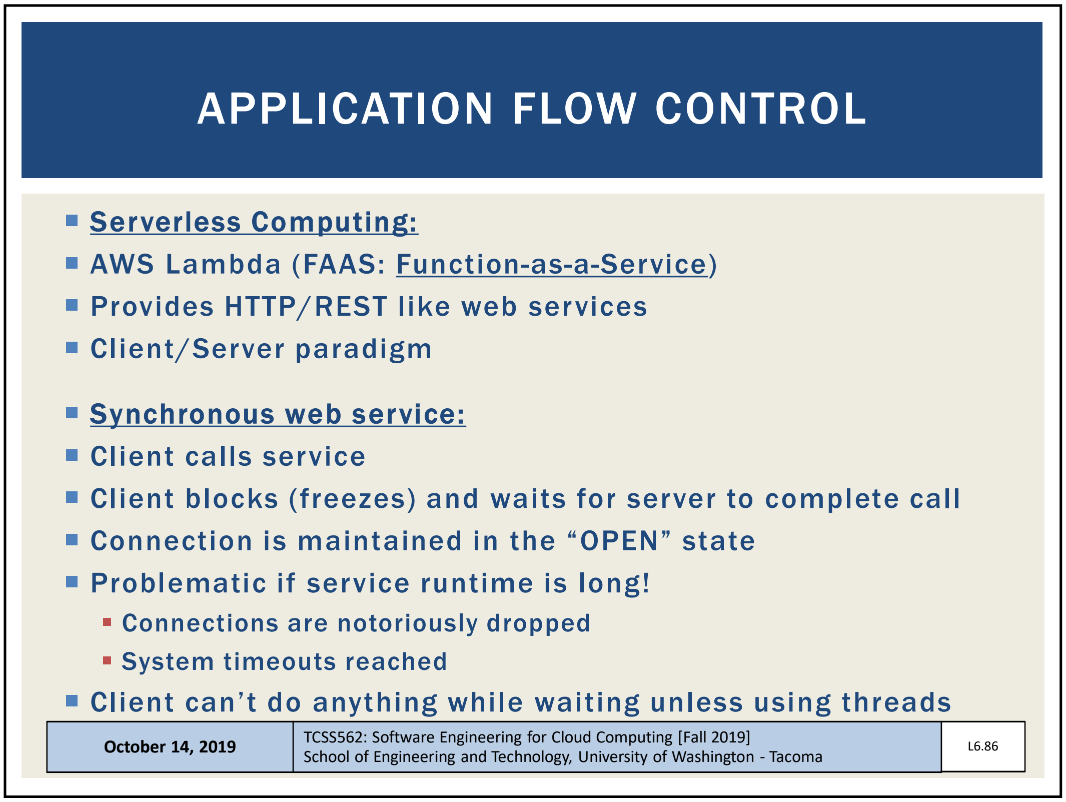
Single deployment package with consolidated codebase (Java: one JAR file)

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

85

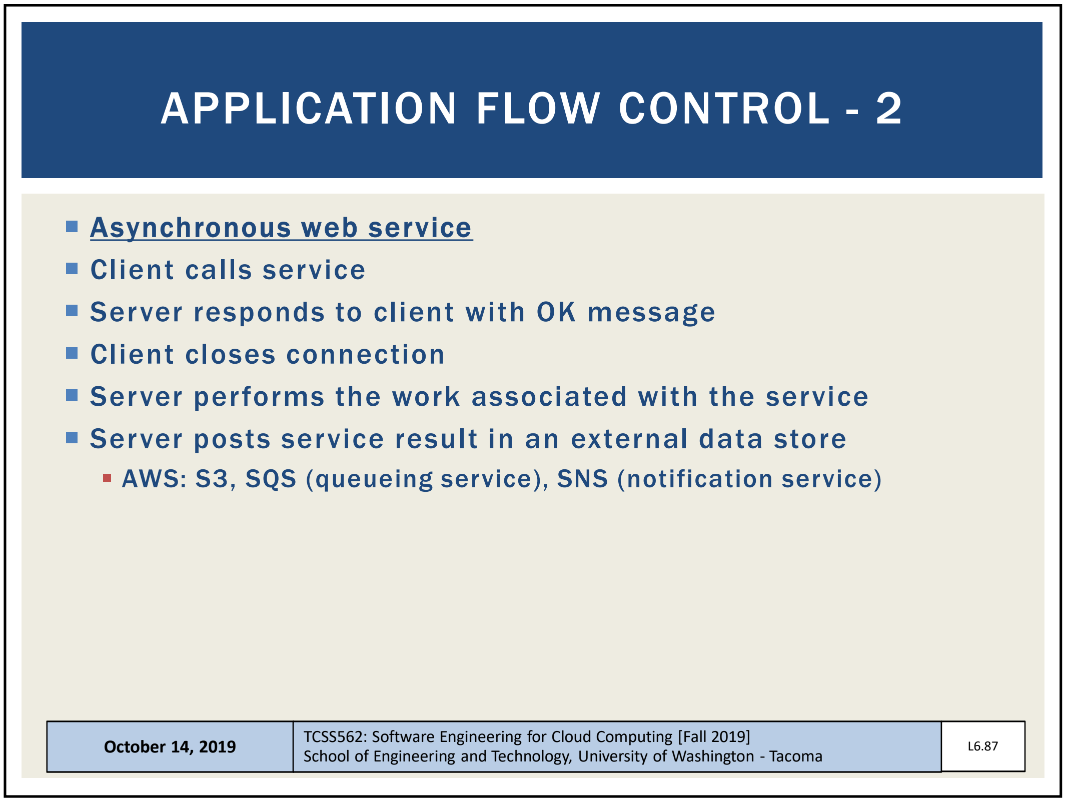


APPLICATION FLOW CONTROL

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

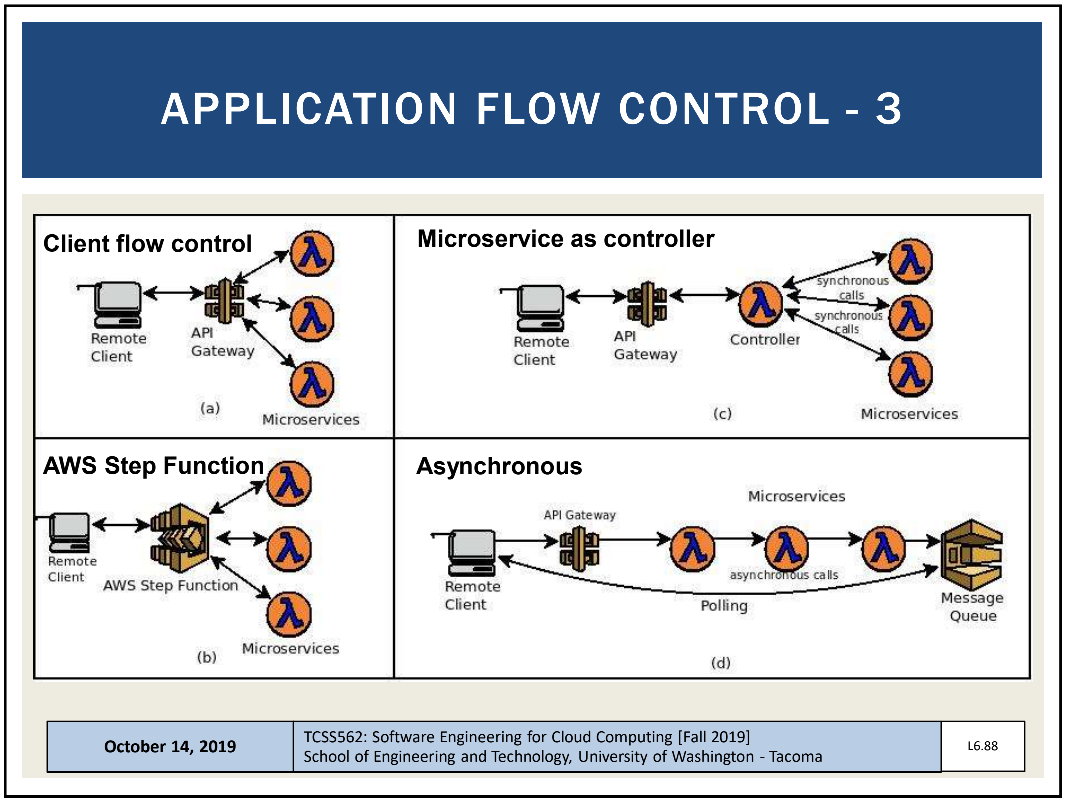
- Synchronous web service:
- Client calls service
- Client blocks (freezes) and waits for server to complete call
- Connection is maintained in the “OPEN” state
- Problematic if service runtime is long!
  - Connections are notoriously dropped
  - System timeouts reached
- Client can’t do anything while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

86

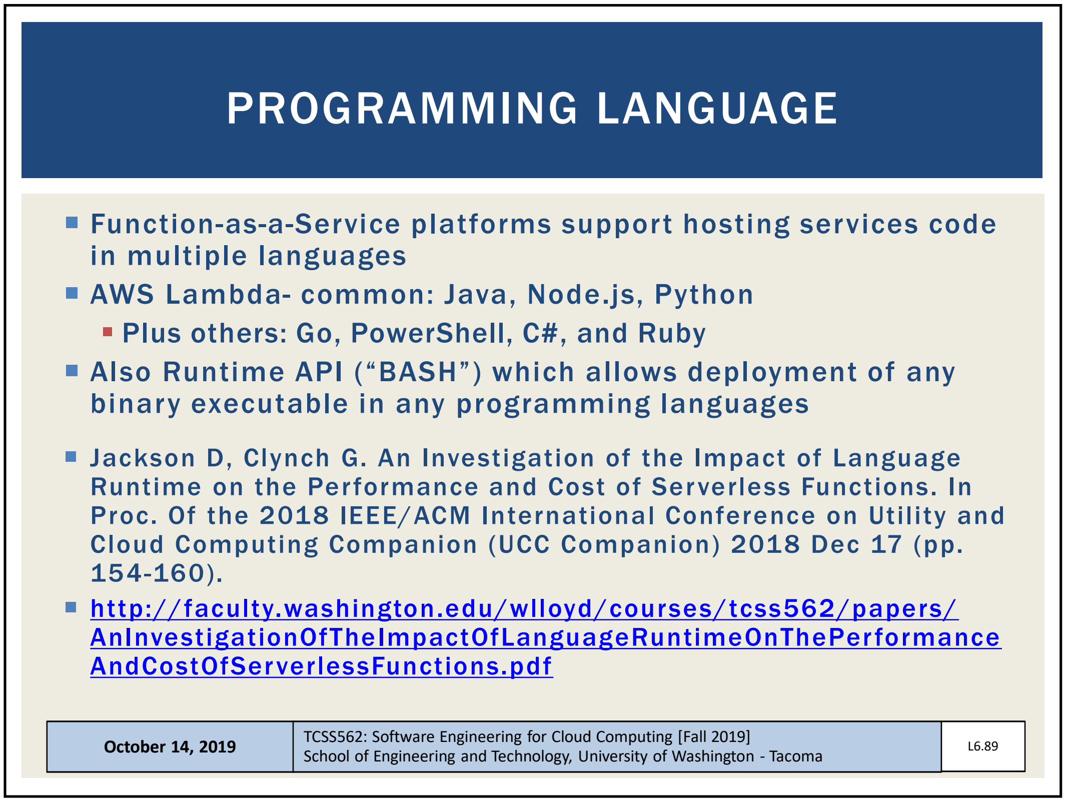


APPLICATION FLOW CONTROL - 2

- 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)

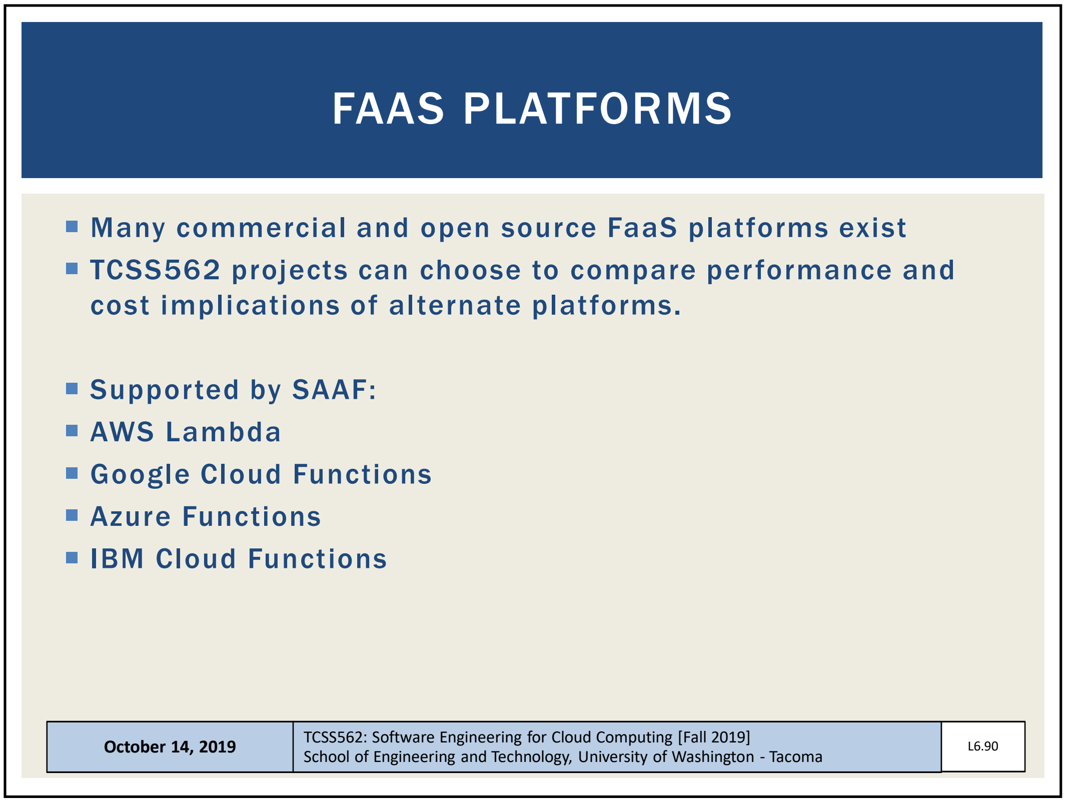


APPLICATION FLOW CONTROL - 3



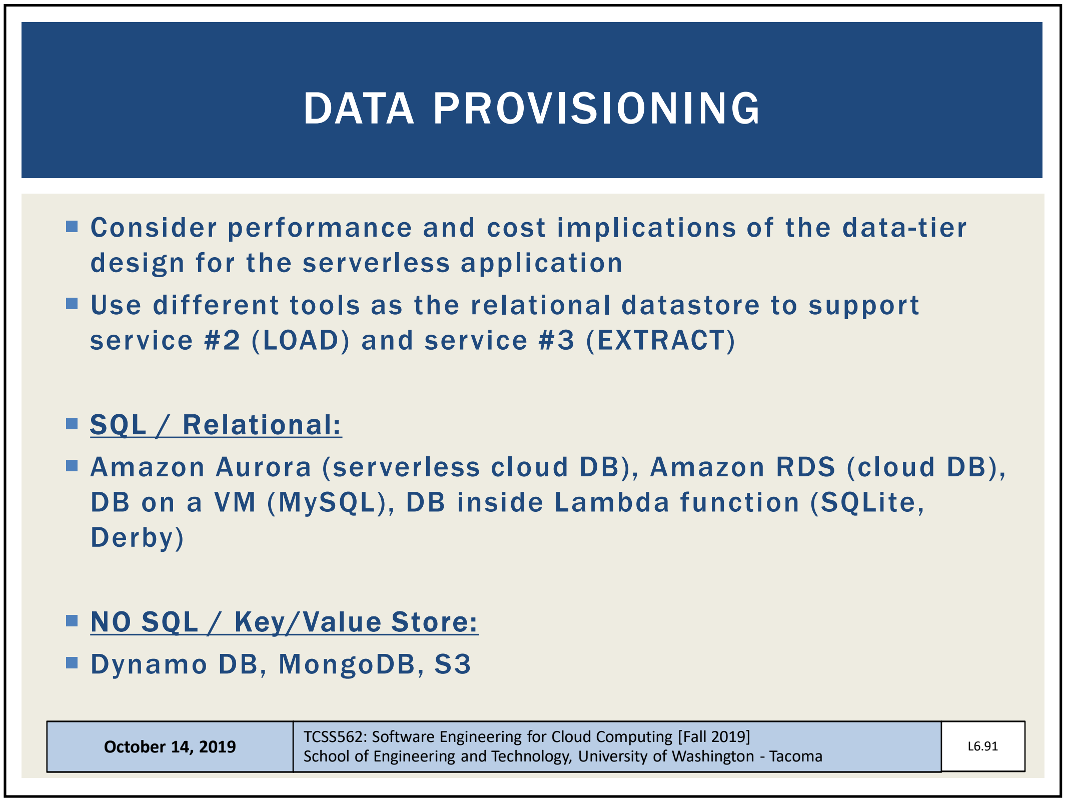
PROGRAMMING LANGUAGE

- 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



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



DATA PROVISIONING

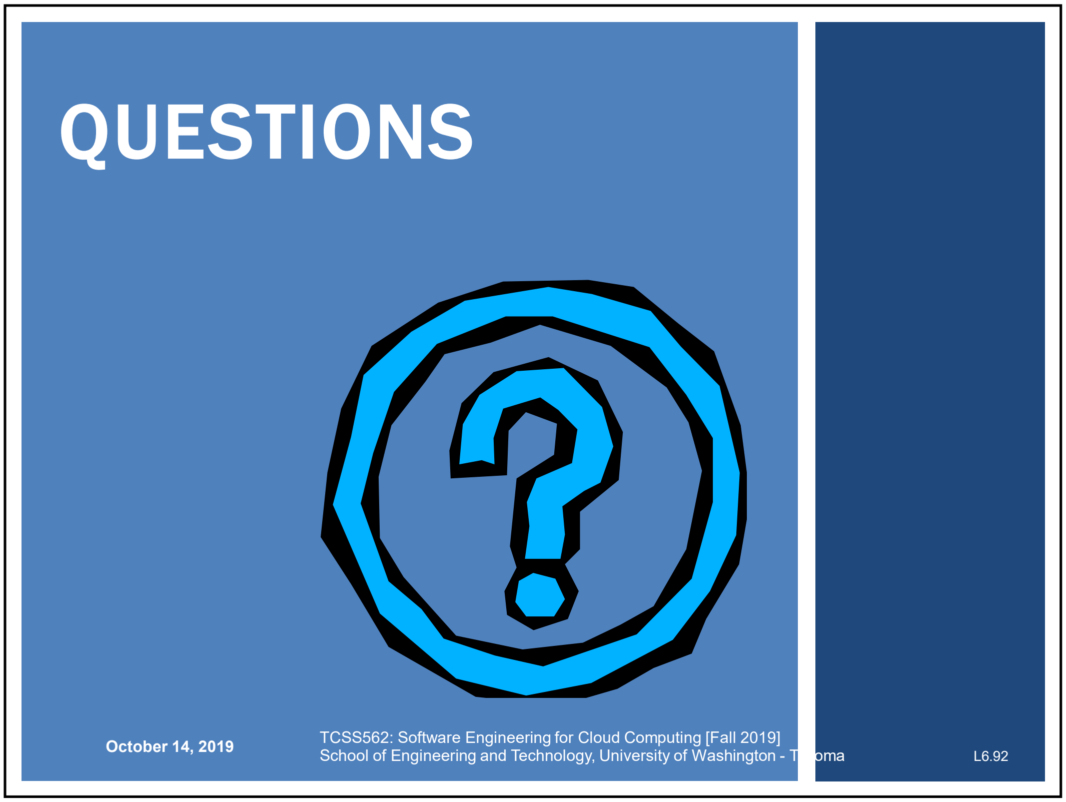
- Consider performance and cost implications of the data-tier design for the serverless application
- Use different tools as the relational datastore to support service #2 (LOAD) and service #3 (EXTRACT)

- **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

October 14, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L6.91

91



QUESTIONS

October 14, 2019 TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - T oma L6.92

92