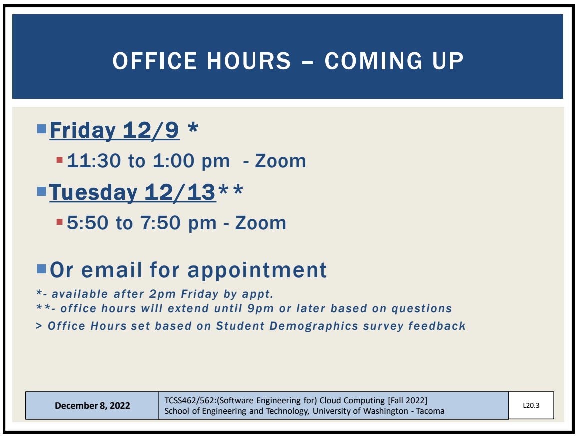
TCSS 462/562: (SOFTWARE ENGINEERING FOR) CLOUD COMPUTING

Group Presentations IV

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

TR 5:50-7:50 PM

2



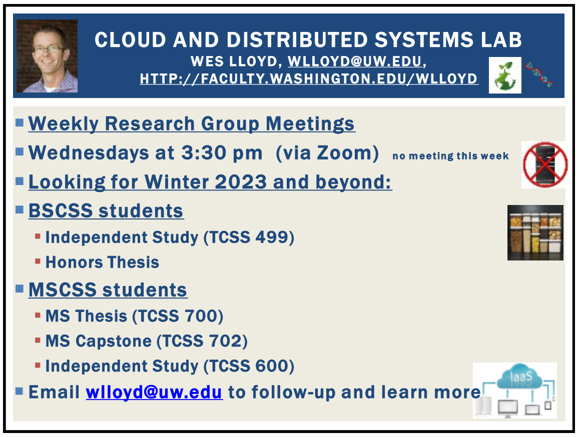
OFFICE HOURS – COMING UP

- **Friday 12/9** *
  - 11:30 to 1:00 pm - Zoom
- **Tuesday 12/13****
  - 5:50 to 7:50 pm - Zoom
- Or email for appointment

**- available after 2pm Friday by appt.*
***- office hours will extend until 9pm or later based on questions*
> Office Hours set based on Student Demographics survey feedback

December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.3

3

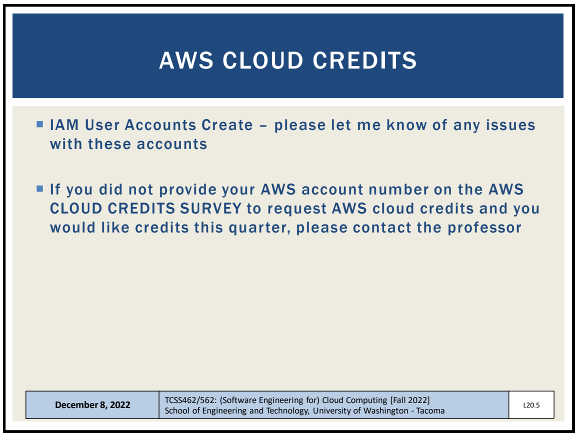


CLOUD AND DISTRIBUTED SYSTEMS LAB

WES LLOYD, WLLOYD@UW.EDU, HTTP://FACULTY.WASHINGTON.EDU/WLLOYD

- **Weekly Research Group Meetings**
- **Wednesdays at 3:30 pm (via Zoom)** no meeting this week
- **Looking for Winter 2023 and beyond:**
- **BSCSS students**
  - Independent Study (TCSS 499)
  - Honors Thesis
- **MSCSS students**
  - MS Thesis (TCSS 700)
  - MS Capstone (TCSS 702)
  - Independent Study (TCSS 600)
- Email wlloyd@uw.edu to follow-up and learn more

4

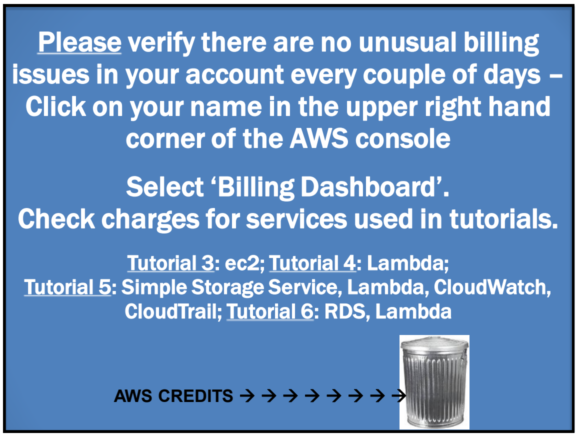


AWS CLOUD CREDITS

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

December 8, 2022 | TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.5

5



Please verify there are no unusual billing issues in your account every couple of days – Click on your name in the upper right hand corner of the AWS console

Select 'Billing Dashboard'. Check charges for services used in tutorials.

Tutorial 3: ec2; Tutorial 4: Lambda; Tutorial 5: Simple Storage Service, Lambda, CloudWatch, CloudTrail; Tutorial 6: RDS, Lambda

AWS CREDITS → → → → → → → →

6

Don't be sorry. Check your AWS bill and credits early and often

AWS CREDITS → → → → → → → →

7

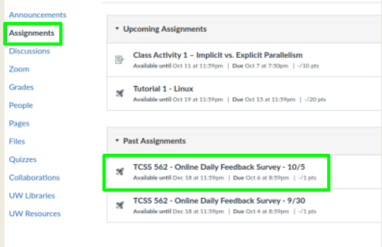
OBJECTIVES – 12/8

- **Questions from 12/6**
- Tutorials Questions
- Tutorial 8: Addressing Serverless Computing Vendor Lock-In through Cloud Service Abstraction (UW Research Study)
- Quiz 2 is posted, available until Tues Dec 13 @ 11:59p
- Optional Tutorials
  - Tutorial 9: AWS Step Functions, SQS
  - Tutorial 10: Intro to FaaSRunner & Pipeline profiling
  - Tutorial 11: Asynchronous Function Profiling w/ SAAF
- A2: Term Project Report
- A3: Term Project Slides (no presentations)
- Class Presentations – 12/8

8

ONLINE DAILY FEEDBACK SURVEY

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



Announcements
Assignments
Discussions
Zoom
Grades
People
Pages
Files
Quizzes
Collaborations
UW Libraries
UW Resources

▾ Upcoming Assignments

Class Activity 1 – Implicit vs. Explicit Parallelism
Available until Oct 11 at 11:59pm | Due Oct 7 at 7:30pm | -/10 pts

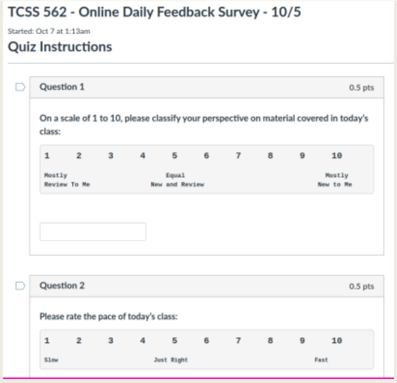
Tutorial 1 - Linux
Available until Oct 19 at 11:59pm | Due Oct 15 at 11:59pm | -/20 pts

▾ 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

9



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
```

10

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (37 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 5.97 (↑ - *previous 5.89*)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.14 (↑ - previous 5.08)**

- **Response rates:**
- TCSS 462: 20/32 – 62.50%
- TCSS 562: 19/26 – 65.38%

11

FEEDBACK FROM 12/6

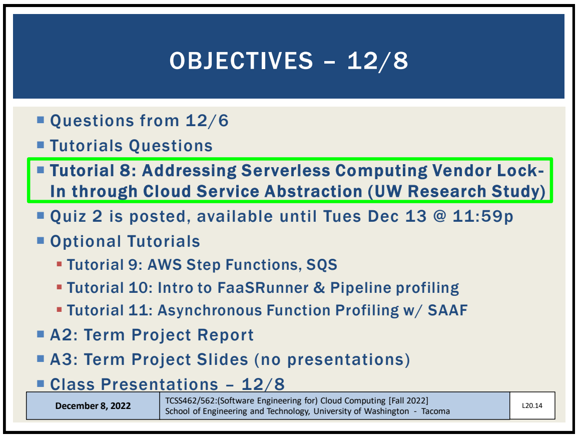
- ..

12

OBJECTIVES – 12/8

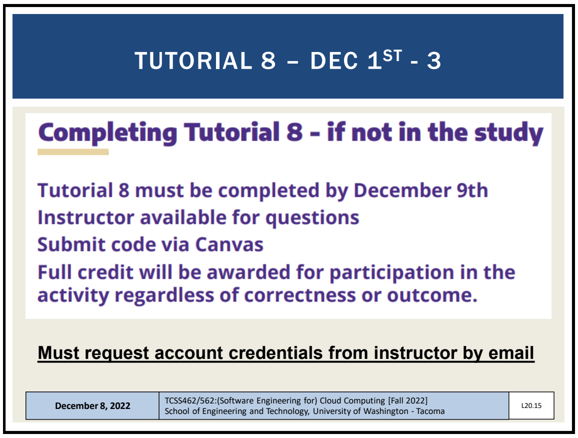
- Questions from 12/6
- **Tutorials Questions**
- Tutorial 8: Addressing Serverless Computing Vendor Lock-In through Cloud Service Abstraction (UW Research Study)
- Quiz 2 is posted, available until Tues Dec 13 @ 11:59p
- Optional Tutorials
  - Tutorial 9: AWS Step Functions, SQS
  - Tutorial 10: Intro to FaaSRunner & Pipeline profiling
  - Tutorial 11: Asynchronous Function Profiling w/ SAAF
- A2: Term Project Report
- A3: Term Project Slides (no presentations)
- Class Presentations – 12/8

13



OBJECTIVES – 12/8

- Questions from 12/6
- Tutorials Questions
- **Tutorial 8: Addressing Serverless Computing Vendor Lock-In through Cloud Service Abstraction (UW Research Study)**
- Quiz 2 is posted, available until Tues Dec 13 @ 11:59p
- Optional Tutorials
  - Tutorial 9: AWS Step Functions, SQS
  - Tutorial 10: Intro to FaaSRunner & Pipeline profiling
  - Tutorial 11: Asynchronous Function Profiling w/ SAAF
- A2: Term Project Report
- A3: Term Project Slides (no presentations)
- Class Presentations – 12/8

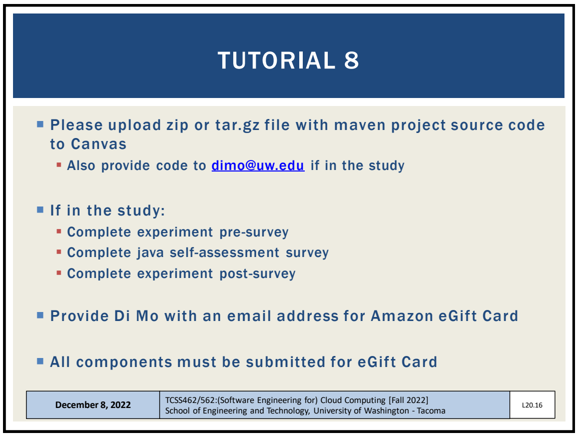


TUTORIAL 8 – DEC 1ST - 3

Completing Tutorial 8 - if not in the study

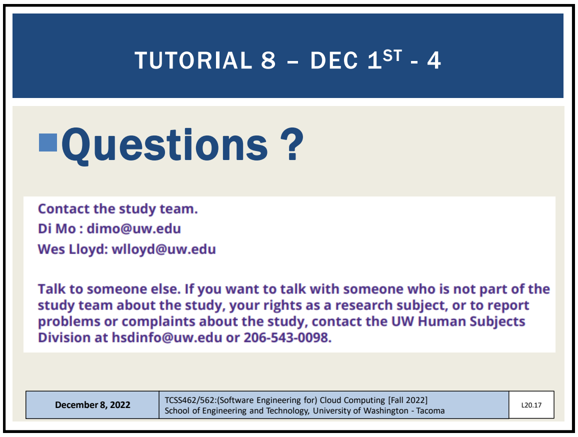
Tutorial 8 must be completed by December 9th
Instructor available for questions
Submit code via Canvas
Full credit will be awarded for participation in the activity regardless of correctness or outcome.

Must request account credentials from instructor by email



TUTORIAL 8

- Please upload zip or tar.gz file with maven project source code to Canvas
  - Also provide code to dimo@uw.edu if in the study
- If in the study:
  - Complete experiment pre-survey
  - Complete java self-assessment survey
  - Complete experiment post-survey
- Provide Di Mo with an email address for Amazon eGift Card
- All components must be submitted for eGift Card

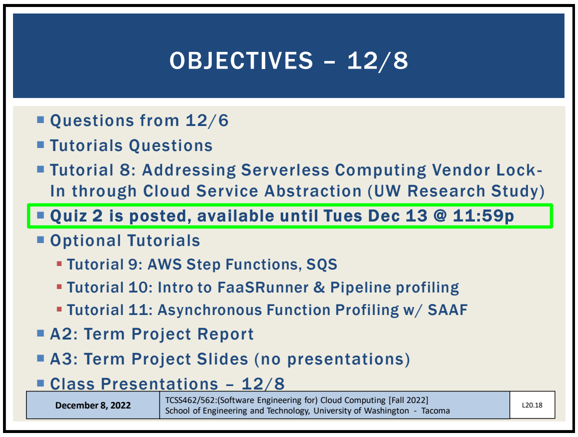


TUTORIAL 8 – DEC 1ST - 4

- **Questions ?**

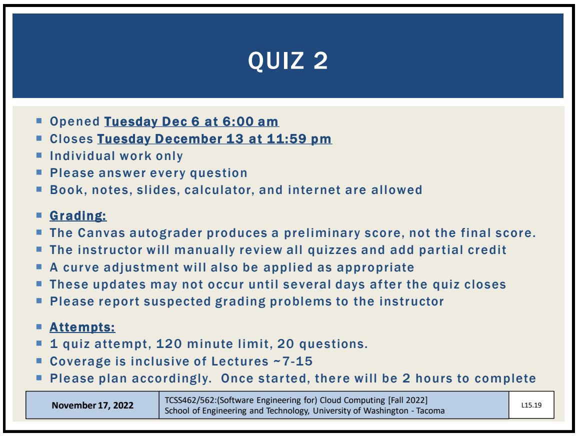
Contact the study team.
Di Mo : dimo@uw.edu
Wes Lloyd: wlloyd@uw.edu

Talk to someone else. If you want to talk with someone who is not part of the study team about the study, your rights as a research subject, or to report problems or complaints about the study, contact the UW Human Subjects Division at hsdinfo@uw.edu or 206-543-0098.



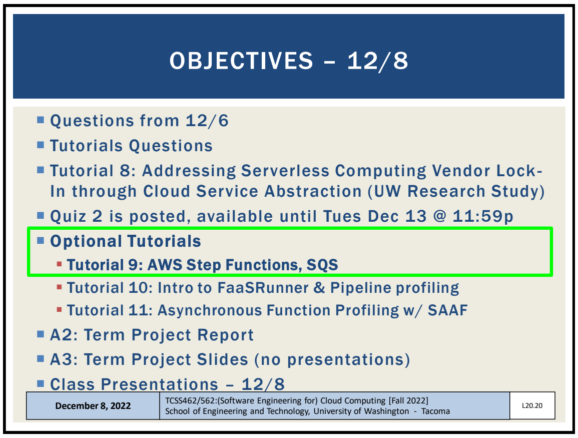
OBJECTIVES – 12/8

- Questions from 12/6
- Tutorials Questions
- Tutorial 8: Addressing Serverless Computing Vendor Lock-In through Cloud Service Abstraction (UW Research Study)
- **Quiz 2 is posted, available until Tues Dec 13 @ 11:59p**
- Optional Tutorials
  - Tutorial 9: AWS Step Functions, SQS
  - Tutorial 10: Intro to FaaSRunner & Pipeline profiling
  - Tutorial 11: Asynchronous Function Profiling w/ SAAF
- A2: Term Project Report
- A3: Term Project Slides (no presentations)
- Class Presentations – 12/8



QUIZ 2

- Opened **Tuesday Dec 6 at 6:00 am**
- Closes **Tuesday December 13 at 11:59 pm**
- Individual work only
- Please answer every question
- Book, notes, slides, calculator, and internet are allowed
- **Grading:**
- The Canvas autograder produces a preliminary score, not the final score.
- The instructor will manually review all quizzes and add partial credit
- A curve adjustment will also be applied as appropriate
- These updates may not occur until several days after the quiz closes
- Please report suspected grading problems to the instructor
- **Attempts:**
- 1 quiz attempt, 120 minute limit, 20 questions.
- Coverage is inclusive of Lectures ~7-15
- Please plan accordingly. Once started, there will be 2 hours to complete

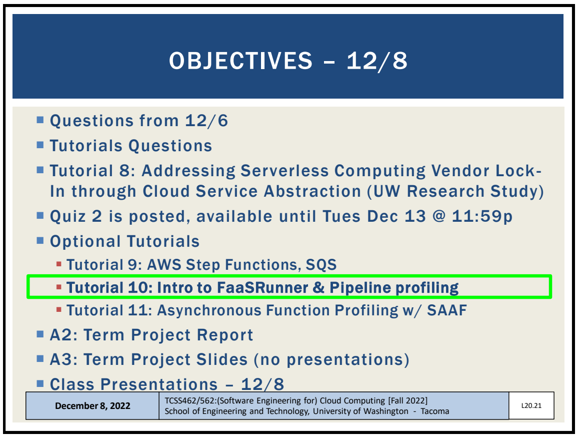


OBJECTIVES – 12/8

- Questions from 12/6
- Tutorials Questions
- Tutorial 8: Addressing Serverless Computing Vendor Lock-In through Cloud Service Abstraction (UW Research Study)
- Quiz 2 is posted, available until Tues Dec 13 @ 11:59p
- **Optional Tutorials**
  - **Tutorial 9: AWS Step Functions, SQS**
  - Tutorial 10: Intro to FaaSRunner & Pipeline profiling
  - Tutorial 11: Asynchronous Function Profiling w/ SAAF
- A2: Term Project Report
- A3: Term Project Slides (no presentations)
- Class Presentations – 12/8

December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.20

20

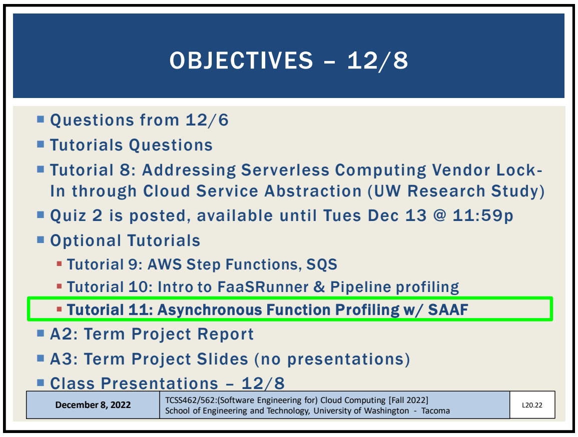


OBJECTIVES – 12/8

- Questions from 12/6
- Tutorials Questions
- Tutorial 8: Addressing Serverless Computing Vendor Lock-In through Cloud Service Abstraction (UW Research Study)
- Quiz 2 is posted, available until Tues Dec 13 @ 11:59p
- Optional Tutorials
  - Tutorial 9: AWS Step Functions, SQS
  - **Tutorial 10: Intro to FaaSRunner & Pipeline profiling**
  - Tutorial 11: Asynchronous Function Profiling w/ SAAF
- A2: Term Project Report
- A3: Term Project Slides (no presentations)
- Class Presentations – 12/8

December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.21

21

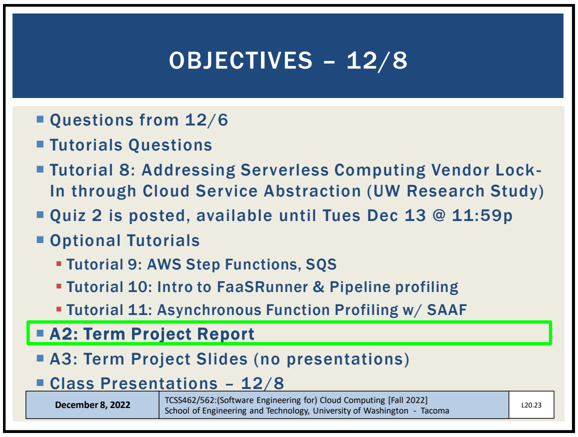


OBJECTIVES – 12/8

- Questions from 12/6
- Tutorials Questions
- Tutorial 8: Addressing Serverless Computing Vendor Lock-In through Cloud Service Abstraction (UW Research Study)
- Quiz 2 is posted, available until Tues Dec 13 @ 11:59p
- Optional Tutorials
  - Tutorial 9: AWS Step Functions, SQS
  - Tutorial 10: Intro to FaaSRunner & Pipeline profiling
  - **Tutorial 11: Asynchronous Function Profiling w/ SAAF**
- A2: Term Project Report
- A3: Term Project Slides (no presentations)
- Class Presentations – 12/8

December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.22

22

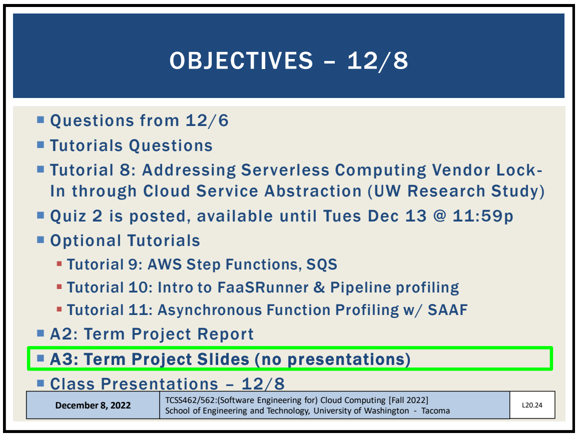


OBJECTIVES – 12/8

- Questions from 12/6
- Tutorials Questions
- Tutorial 8: Addressing Serverless Computing Vendor Lock-In through Cloud Service Abstraction (UW Research Study)
- Quiz 2 is posted, available until Tues Dec 13 @ 11:59p
- Optional Tutorials
  - Tutorial 9: AWS Step Functions, SQS
  - Tutorial 10: Intro to FaaSRunner & Pipeline profiling
  - Tutorial 11: Asynchronous Function Profiling w/ SAAF
- **A2: Term Project Report**
- A3: Term Project Slides (no presentations)
- Class Presentations – 12/8

December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.23

23

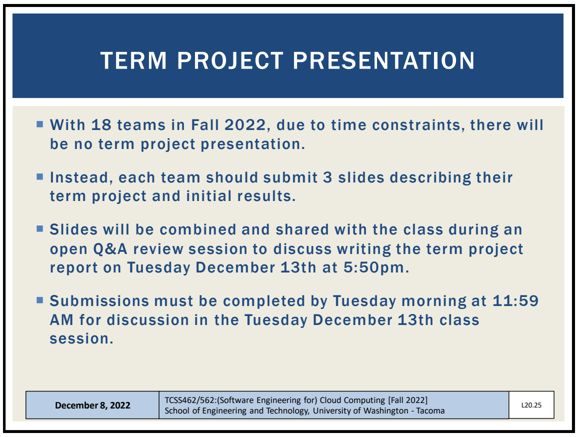


OBJECTIVES – 12/8

- Questions from 12/6
- Tutorials Questions
- Tutorial 8: Addressing Serverless Computing Vendor Lock-In through Cloud Service Abstraction (UW Research Study)
- Quiz 2 is posted, available until Tues Dec 13 @ 11:59p
- Optional Tutorials
  - Tutorial 9: AWS Step Functions, SQS
  - Tutorial 10: Intro to FaaSRunner & Pipeline profiling
  - Tutorial 11: Asynchronous Function Profiling w/ SAAF
- A2: Term Project Report
- **A3: Term Project Slides (no presentations)**
- Class Presentations – 12/8

December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.24

24



TERM PROJECT PRESENTATION

- With 18 teams in Fall 2022, due to time constraints, there will be no term project presentation.
- Instead, each team should submit 3 slides describing their term project and initial results.
- Slides will be combined and shared with the class during an open Q&A review session to discuss writing the term project report on Tuesday December 13th at 5:50pm.
- Submissions must be completed by Tuesday morning at 11:59 AM for discussion in the Tuesday December 13th class session.

December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.25

25

OBJECTIVES – 12/8

- Questions from 12/6
- Tutorials Questions
- Tutorial 8: Addressing Serverless Computing Vendor Lock-In through Cloud Service Abstraction (UW Research Study)
- Quiz 2 is posted, available until Tues Dec 13 @ 11:59p
- Optional Tutorials
  - Tutorial 9: AWS Step Functions, SQS
  - Tutorial 10: Intro to FaaSRunner & Pipeline profiling
  - Tutorial 11: Asynchronous Function Profiling w/ SAAF
- A2: Term Project Report
- A3: Term Project Slides (no presentations)
- **Class Presentations – 12/8**

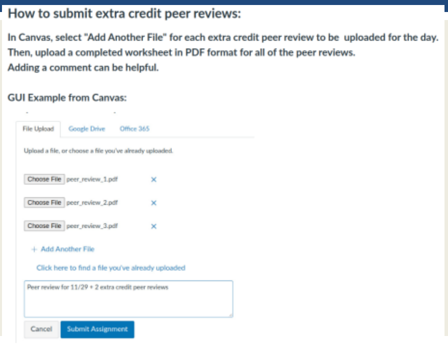
26

GROUP PRESENTATIONS

- *Cloud technology presentation*
- *Cloud research paper presentation*
- *** PLEASE UPLOAD FINAL VERSION OF SLIDES ***
  - If final slides are not uploaded, then the draft version will be used for grading
  - Final slides are due by Friday Dec 9th at 11:59pm *(late submission until Tuesday Dec 13th at 11:39pm)*
- **Peer Reviews**
  - Word DOCX form is posted, fill out, submit PDFs on Canvas
  - Feedback shared with groups
  - TCSS 462: 1 review/day required, additional are extra credit
  - TCSS 562: same as 462, but no peer review req'd on day of your talk

27

SUBMITTING EXTRA CREDIT PEER REVIEWS



How to submit extra credit peer reviews:

In Canvas, select "Add Another File" for each extra credit peer review to be uploaded for the day. Then, upload a completed worksheet in PDF format for all of the peer reviews. Adding a comment can be helpful.

GUI Example from Canvas:

Upload a file, or choose a file you've already uploaded.

Choose File peer_review_1.pdf

Choose File peer_review_2.pdf

Choose File peer_review_3.pdf

Peer review for 11/29 + 2 extra credit peer reviews

28

PRESENTATIONS – 12/8

Cloud Research Paper Presentation: Towards Federated Learning using FaaS Fabric (Team 2)
Mohammed Alshayeb

Cloud Technology: Azure ML (team 8)
Nicole Guobadia, Andrew Moreno-Escareno

Cloud Technology: AWS Simple Notification Service (SNS) (team 12)
RamaSoumya Naraparaju, Sathwika Suddala, Chhavi Gupta

Cloud Research Paper Presentation: Cypress: Input size – Sensitive Container Provisioning and Request Scheduling for Serverless (Team 5)
Yafei Li, Sue Yang

29

WE WILL RETURN AT ~7:00 PM

30

PRESENTATIONS – 12/8

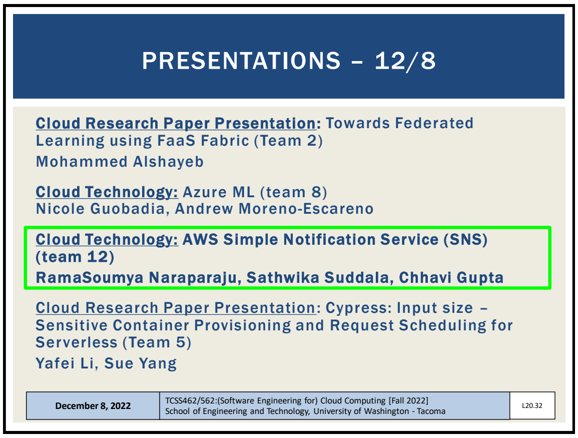
Cloud Research Paper Presentation: Towards Federated Learning using FaaS Fabric (Team 2)
Mohammed Alshayeb

Cloud Technology: Azure ML (team 8)
Nicole Guobadia, Andrew Moreno-Escareno

Cloud Technology: AWS Simple Notification Service (SNS) (team 12)
RamaSoumya Naraparaju, Sathwika Suddala, Chhavi Gupta

Cloud Research Paper Presentation: Cypress: Input size – Sensitive Container Provisioning and Request Scheduling for Serverless (Team 5)
Yafei Li, Sue Yang

31



PRESENTATIONS – 12/8

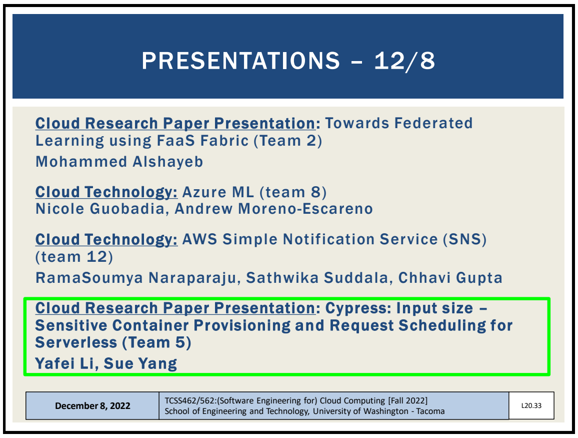
Cloud Research Paper Presentation: Towards Federated Learning using FaaS Fabric (Team 2)
Mohammed Alshayeb

Cloud Technology: Azure ML (team 8)
Nicole Guobadia, Andrew Moreno-Escareno

Cloud Technology: AWS Simple Notification Service (SNS) (team 12)
RamaSoumya Naraparaju, Sathwika Suddala, Chhavi Gupta

Cloud Research Paper Presentation: Cypress: Input size – Sensitive Container Provisioning and Request Scheduling for Serverless (Team 5)
Yafei Li, Sue Yang

32



PRESENTATIONS – 12/8

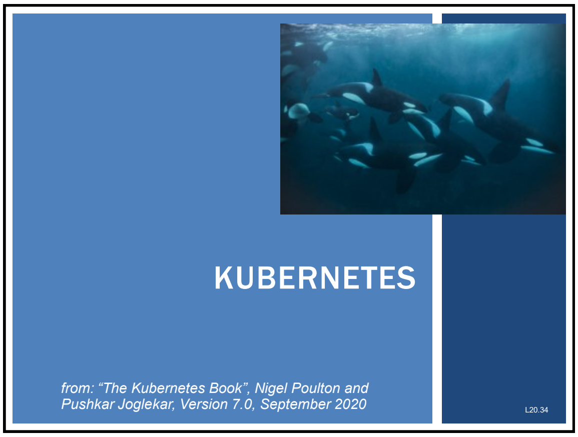
Cloud Research Paper Presentation: Towards Federated Learning using FaaS Fabric (Team 2)
Mohammed Alshayeb

Cloud Technology: Azure ML (team 8)
Nicole Guobadia, Andrew Moreno-Escareno

Cloud Technology: AWS Simple Notification Service (SNS) (team 12)
RamaSoumya Naraparaju, Sathwika Suddala, Chhavi Gupta

Cloud Research Paper Presentation: Cypress: Input size – Sensitive Container Provisioning and Request Scheduling for Serverless (Team 5)
Yafei Li, Sue Yang

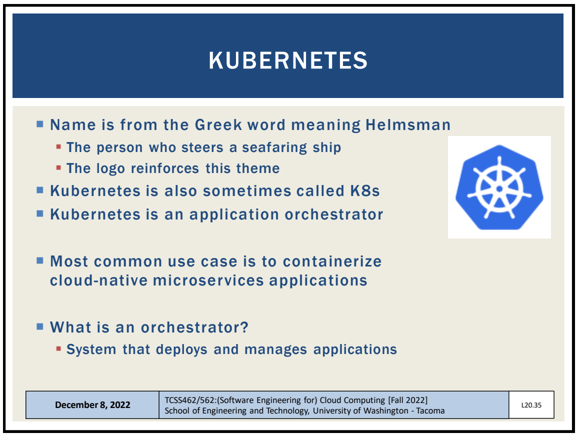
33



KUBERNETES

from: "The Kubernetes Book", Nigel Poulton and Pushkar Joglekar, Version 7.0, September 2020

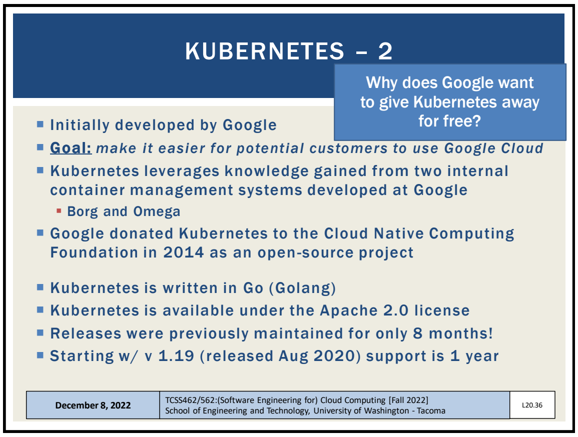
34



KUBERNETES

- Name is from the Greek word meaning Helmsman
  - The person who steers a seafaring ship
  - The logo reinforces this theme
- Kubernetes is also sometimes called K8s
- Kubernetes is an application orchestrator
- Most common use case is to containerize cloud-native microservices applications
- What is an orchestrator?
  - System that deploys and manages applications

35

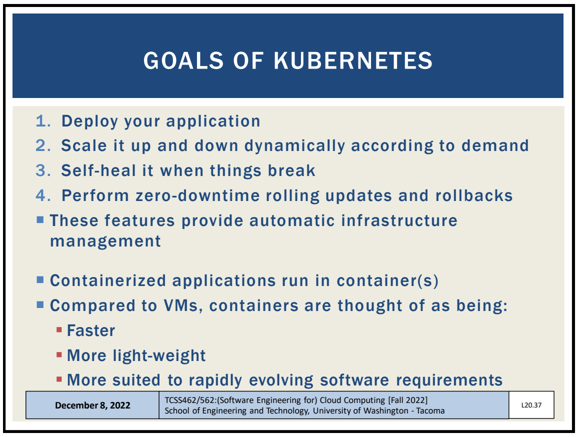


KUBERNETES – 2

Why does Google want to give Kubernetes away for free?

- Initially developed by Google
- **Goal:** *make it easier for potential customers to use Google Cloud*
- Kubernetes leverages knowledge gained from two internal container management systems developed at Google
  - Borg and Omega
- Google donated Kubernetes to the Cloud Native Computing Foundation in 2014 as an open-source project
- Kubernetes is written in Go (Golang)
- Kubernetes is available under the Apache 2.0 license
- Releases were previously maintained for only 8 months!
- Starting w/ v 1.19 (released Aug 2020) support is 1 year

36

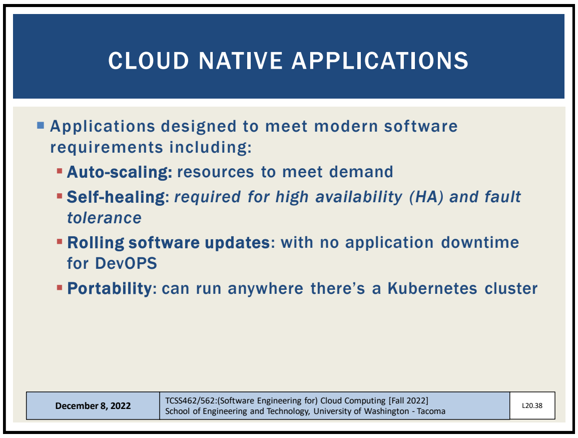


GOALS OF KUBERNETES

1. Deploy your application
2. Scale it up and down dynamically according to demand
3. Self-heal it when things break
4. Perform zero-downtime rolling updates and rollbacks

- These features provide automatic infrastructure management
- Containerized applications run in container(s)
- Compared to VMs, containers are thought of as being:
  - Faster
  - More light-weight
  - More suited to rapidly evolving software requirements

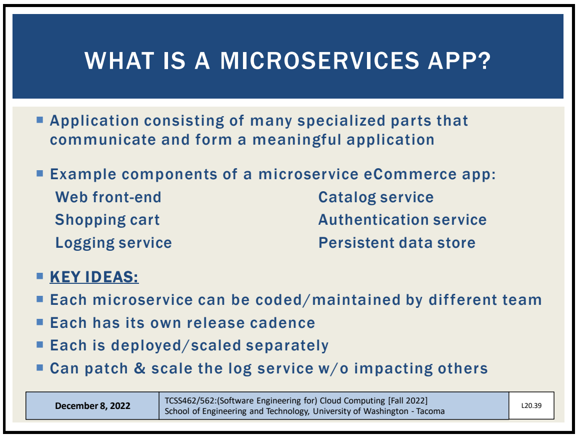
37



CLOUD NATIVE APPLICATIONS

- Applications designed to meet modern software requirements including:
  - **Auto-scaling:** resources to meet demand
  - **Self-healing**: *required for high availability (HA) and fault tolerance*
  - **Rolling software updates**: with no application downtime for DevOPS
  - **Portability**: can run anywhere there's a Kubernetes cluster

38



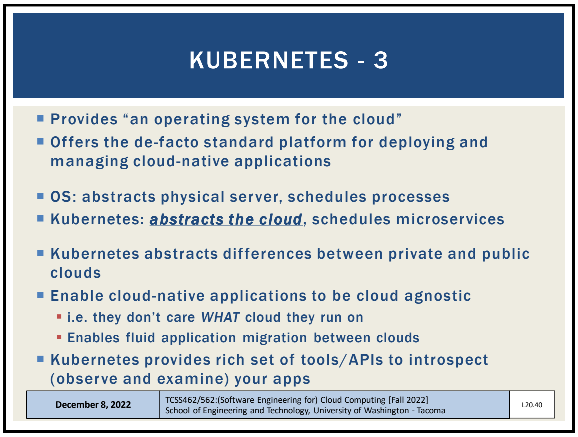
WHAT IS A MICROSERVICES APP?

- Application consisting of many specialized parts that communicate and form a meaningful application
- Example components of a microservice eCommerce app:

Web front-end	Catalog service
Shopping cart	Authentication service
Logging service	Persistent data store

- **KEY IDEAS:**
- Each microservice can be coded/maintained by different team
- Each has its own release cadence
- Each is deployed/scaled separately
- Can patch & scale the log service w/o impacting others

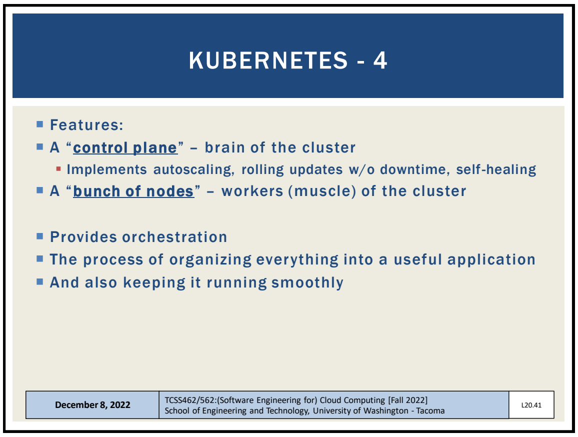
39



KUBERNETES - 3

- Provides "an operating system for the cloud"
- Offers the de-facto standard platform for deploying and managing cloud-native applications
- OS: abstracts physical server, schedules processes
- Kubernetes: ***abstracts the cloud***, schedules microservices
- Kubernetes abstracts differences between private and public clouds
- Enable cloud-native applications to be cloud agnostic
  - i.e. they don't care *WHAT* cloud they run on
  - Enables fluid application migration between clouds
- Kubernetes provides rich set of tools/APIs to introspect (observe and examine) your apps

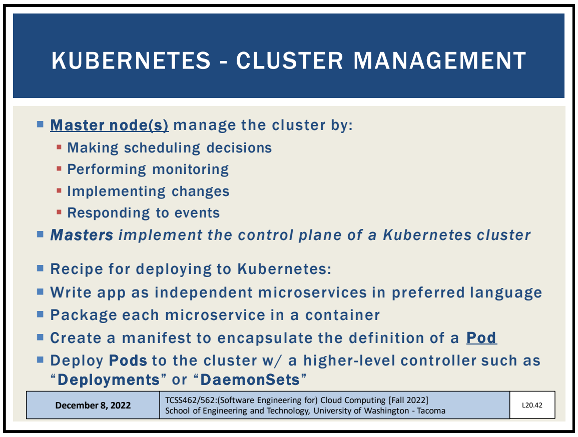
40



KUBERNETES - 4

- Features:
- A "**control plane**" – brain of the cluster
  - Implements autoscaling, rolling updates w/o downtime, self-healing
- A "**bunch of nodes**" – workers (muscle) of the cluster
- Provides orchestration
- The process of organizing everything into a useful application
- And also keeping it running smoothly

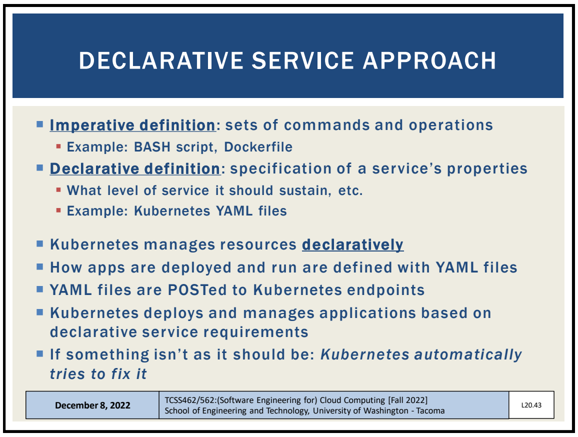
41



KUBERNETES - CLUSTER MANAGEMENT

- **Master node(s)** manage the cluster by:
  - Making scheduling decisions
  - Performing monitoring
  - Implementing changes
  - Responding to events
- ***Masters** implement the control plane of a Kubernetes cluster*
- Recipe for deploying to Kubernetes:
- Write app as independent microservices in preferred language
- Package each microservice in a container
- Create a manifest to encapsulate the definition of a **Pod**
- Deploy **Pods** to the cluster w/ a higher-level controller such as "**Deployments**" or "**DaemonSets**"

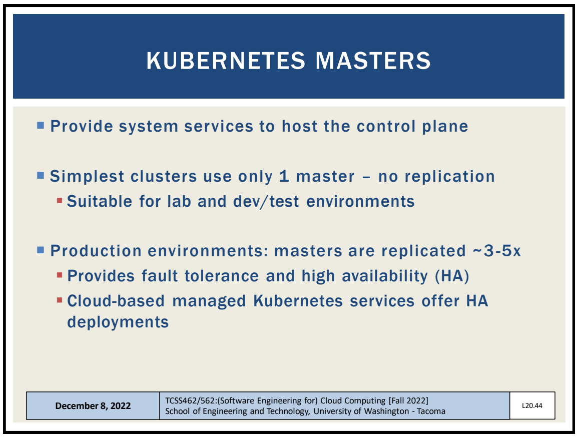
42



DECLARATIVE SERVICE APPROACH

- **Imperative definition**: sets of commands and operations
  - Example: BASH script, Dockerfile
- **Declarative definition**: specification of a service's properties
  - What level of service it should sustain, etc.
  - Example: Kubernetes YAML files
- Kubernetes manages resources **declaratively**
- How apps are deployed and run are defined with YAML files
- YAML files are POSTed to Kubernetes endpoints
- Kubernetes deploys and manages applications based on declarative service requirements
- If something isn't as it should be: *Kubernetes automatically tries to fix it*

43

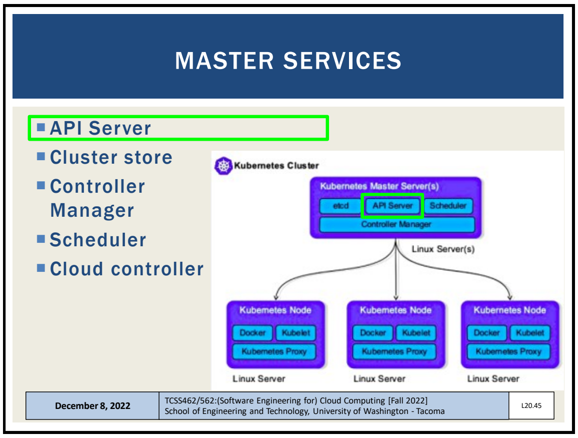


KUBERNETES MASTERS

- Provide system services to host the control plane
- Simplest clusters use only 1 master – no replication
  - Suitable for lab and dev/test environments
- Production environments: masters are replicated ~3-5x
  - Provides fault tolerance and high availability (HA)
  - Cloud-based managed Kubernetes services offer HA deployments

December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.44

44

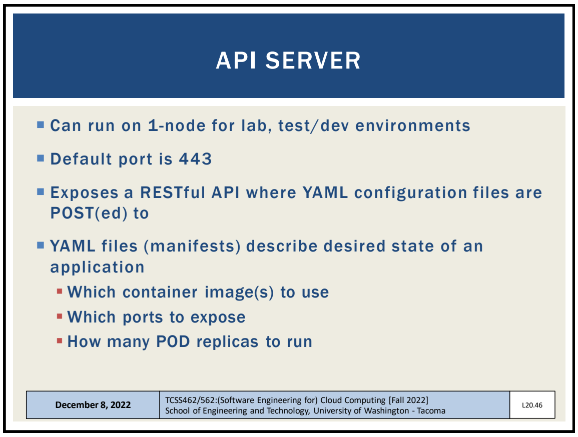


MASTER SERVICES

- API Server
- Cluster store
- Controller Manager
- Scheduler
- Cloud controller

December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.45

45

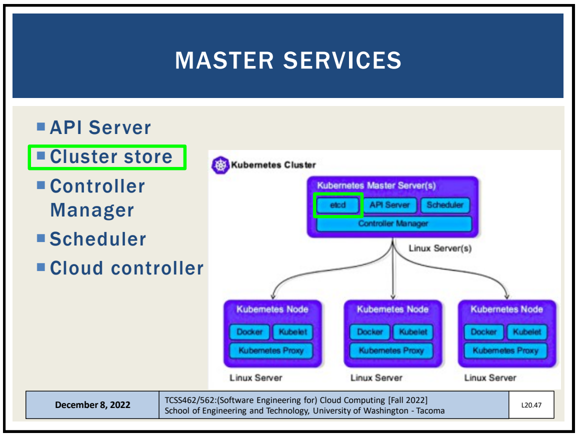


API SERVER

- Can run on 1-node for lab, test/dev environments
- Default port is 443
- Exposes a RESTful API where YAML configuration files are POST(ed) to
- YAML files (manifests) describe desired state of an application
  - Which container image(s) to use
  - Which ports to expose
  - How many POD replicas to run

December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.46

46

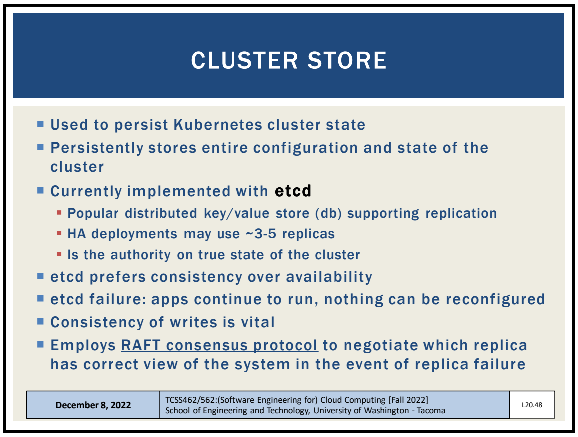


MASTER SERVICES

- API Server
- Cluster store
- Controller Manager
- Scheduler
- Cloud controller

December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.47

47

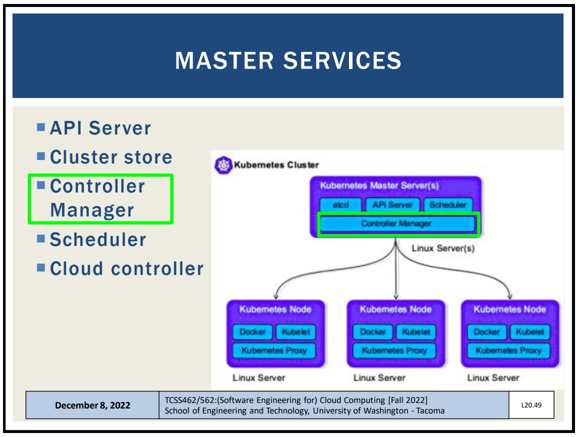


CLUSTER STORE

- Used to persist Kubernetes cluster state
- Persistently stores entire configuration and state of the cluster
- Currently implemented with **etcd**
  - Popular distributed key/value store (db) supporting replication
  - HA deployments may use ~3-5 replicas
  - Is the authority on true state of the cluster
- etcd prefers consistency over availability
- etcd failure: apps continue to run, nothing can be reconfigured
- Consistency of writes is vital
- Employs RAFT consensus protocol to negotiate which replica has correct view of the system in the event of replica failure

December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.48

48

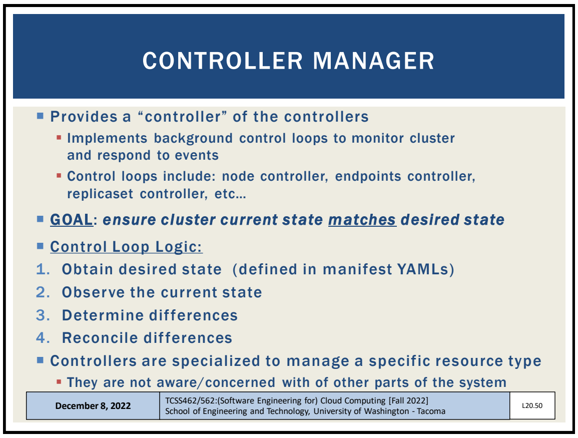


MASTER SERVICES

- API Server
- Cluster store
- Controller Manager
- Scheduler
- Cloud controller

December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.49

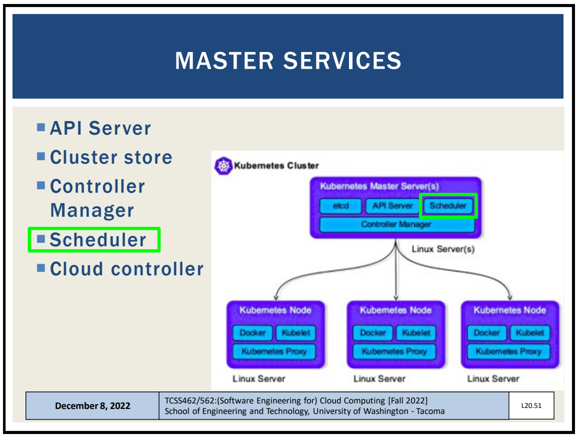
49



CONTROLLER MANAGER

- Provides a “controller” of the controllers
  - Implements background control loops to monitor cluster and respond to events
  - Control loops include: node controller, endpoints controller, replicaset controller, etc…
- **GOAL**: ***ensure cluster current state matches desired state***
- Control Loop Logic:
1. Obtain desired state (defined in manifest YAMLs)
2. Observe the current state
3. Determine differences
4. Reconcile differences
- Controllers are specialized to manage a specific resource type
  - They are not aware/concerned with of other parts of the system

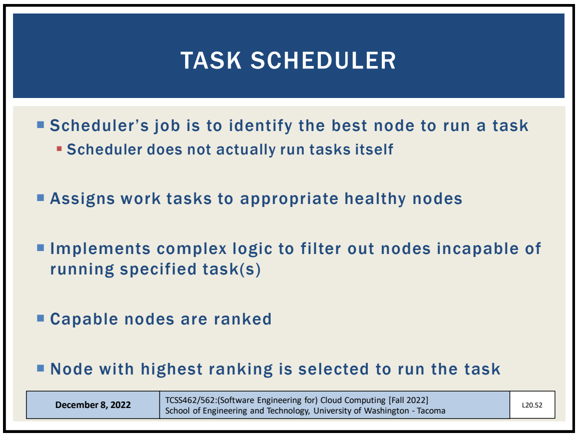
50



MASTER SERVICES

- API Server
- Cluster store
- Controller Manager
- Scheduler
- Cloud controller

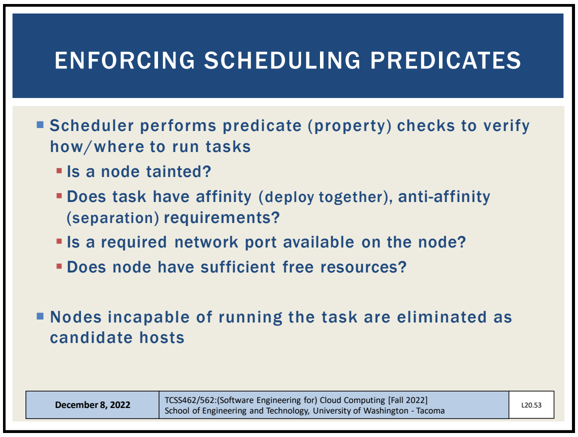
51



TASK SCHEDULER

- Scheduler’s job is to identify the best node to run a task
  - Scheduler does not actually run tasks itself
- Assigns work tasks to appropriate healthy nodes
- Implements complex logic to filter out nodes incapable of running specified task(s)
- Capable nodes are ranked
- Node with highest ranking is selected to run the task

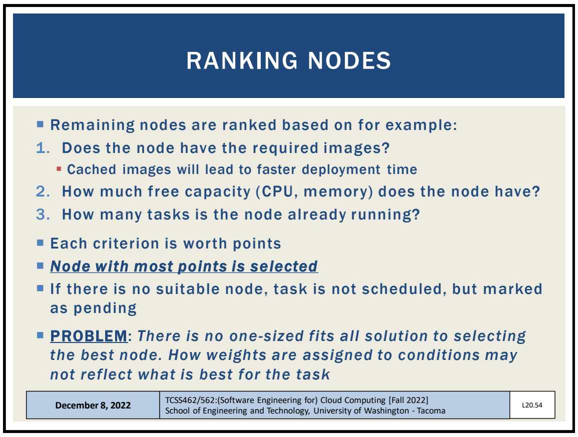
52



ENFORCING SCHEDULING PREDICATES

- Scheduler performs predicate (property) checks to verify how/where to run tasks
  - Is a node tainted?
  - Does task have affinity (deploy together), anti-affinity (separation) requirements?
  - Is a required network port available on the node?
  - Does node have sufficient free resources?
- Nodes incapable of running the task are eliminated as candidate hosts

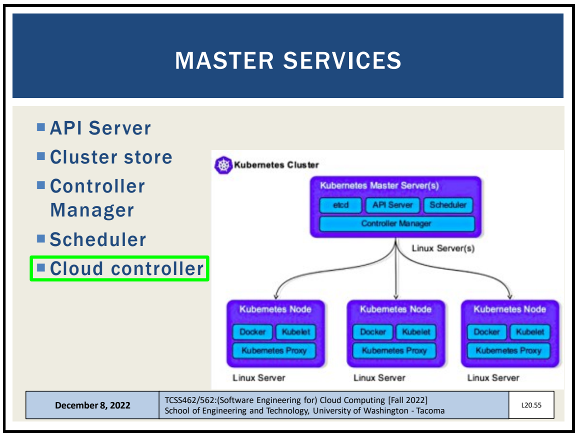
53



RANKING NODES

- Remaining nodes are ranked based on for example:
1. Does the node have the required images?
   - Cached images will lead to faster deployment time
2. How much free capacity (CPU, memory) does the node have?
3. How many tasks is the node already running?
- Each criterion is worth points
- ***Node with most points is selected***
- If there is no suitable node, task is not scheduled, but marked as pending
- **PROBLEM**: *There is no one-sized fits all solution to selecting the best node. How weights are assigned to conditions may not reflect what is best for the task*

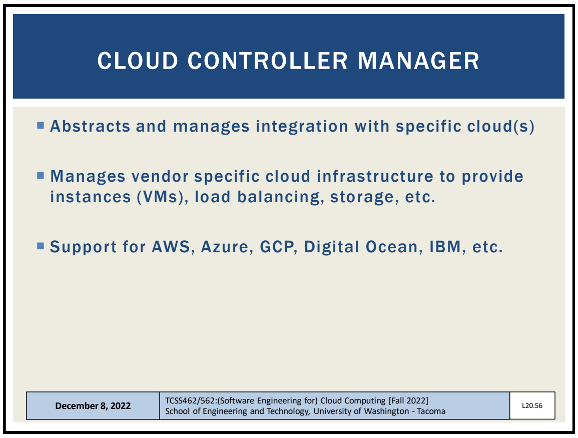
54



MASTER SERVICES

- API Server
- Cluster store
- Controller Manager
- Scheduler
- Cloud controller

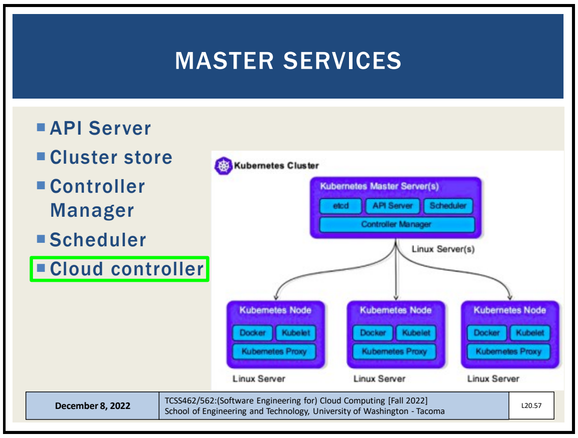
55



CLOUD CONTROLLER MANAGER

- Abstracts and manages integration with specific cloud(s)
- Manages vendor specific cloud infrastructure to provide instances (VMs), load balancing, storage, etc.
- Support for AWS, Azure, GCP, Digital Ocean, IBM, etc.

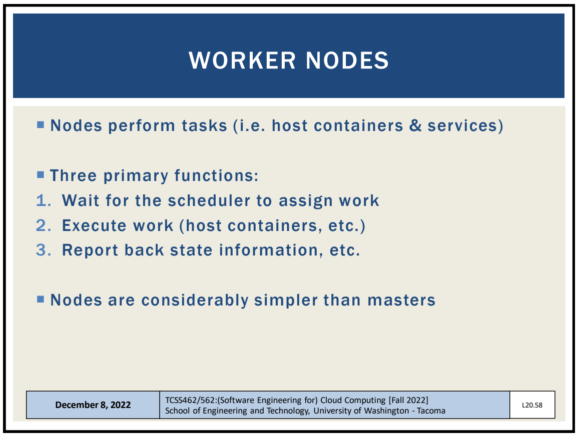
56



MASTER SERVICES

- API Server
- Cluster store
- Controller Manager
- Scheduler
- Cloud controller

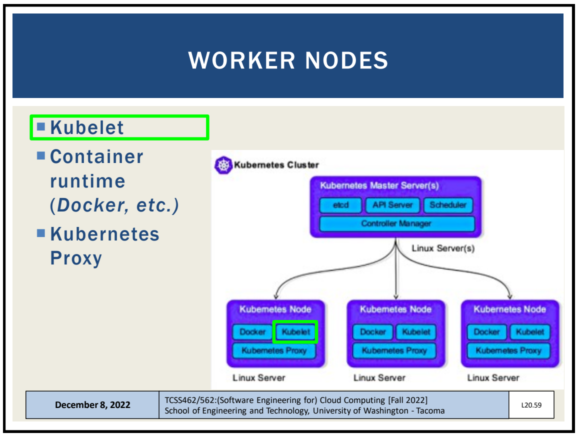
57



WORKER NODES

- Nodes perform tasks (i.e. host containers & services)
- Three primary functions:
  1. Wait for the scheduler to assign work
  2. Execute work (host containers, etc.)
  3. Report back state information, etc.
- Nodes are considerably simpler than masters

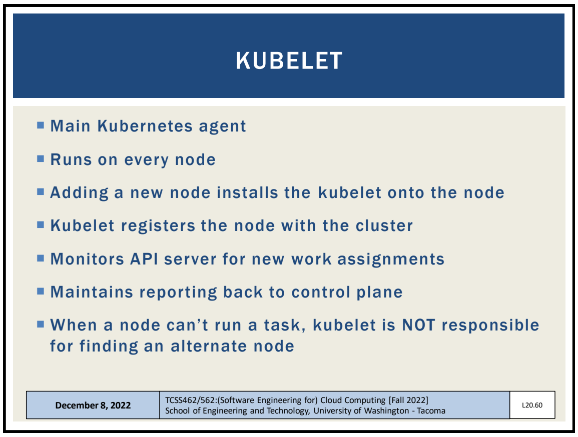
58



WORKER NODES

- Kubelet
- Container runtime *(Docker, etc.)*
- Kubernetes Proxy

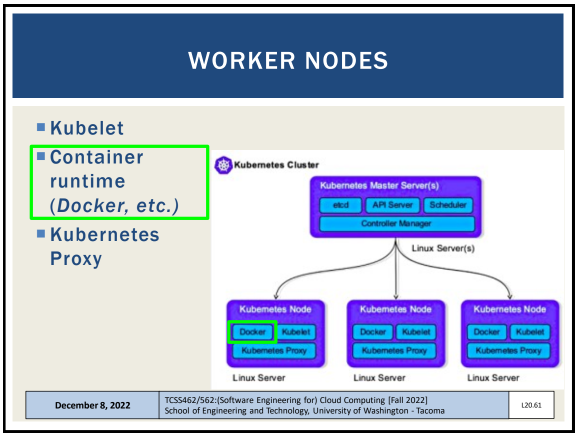
59



KUBELET

- Main Kubernetes agent
- Runs on every node
- Adding a new node installs the kubelet onto the node
- Kubelet registers the node with the cluster
- Monitors API server for new work assignments
- Maintains reporting back to control plane
- When a node can't run a task, kubelet is NOT responsible for finding an alternate node

60



WORKER NODES

- Kubelet
- Container runtime *(Docker, etc.)*
- Kubernetes Proxy

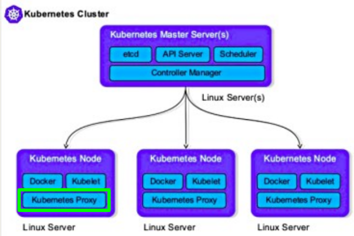
61

CONTAINER RUNTIME(S)

- Each node requires a container runtime to run containers
- Early versions had custom support for a limited number of container types, e.g. Docker
- Kubernetes now provides a standard Container Runtime Interface (CRI)
- CRI exposes a clean interface for 3rd party container runtimes to plug-in to
- Popular container runtimes: Docker, containerd, Kata

WORKER NODES

- Kubelet
- Container runtime (*Docker, etc.)*
- Kubernetes Proxy



KUBE-PROXY

- Runs on every node in the cluster
- Responsible for managing the cluster's networking
- Ensures each node obtains a unique IP address
- Implemented local IPTABLES and IPVS rules to route and load-balance traffic
- IPTABLES (ipv4) – enables configuration of IP packet filtering rules of the Linux kernel firewall
- IPVS – IP Virtual Server: provides transport-layer (layer 4) load balancing as part of the Linux kernel; Configured using ipvsadm tool in Linux

CORE KUBERNETES COMPONENTS

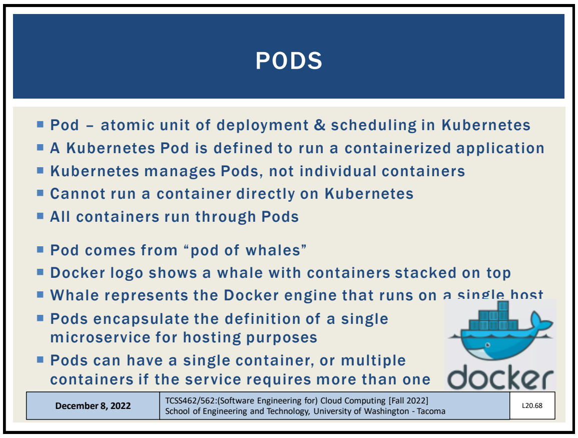
- Kubernetes DNS
- Pods
- Services

KUBERNETES DNS

- Every Kubernetes cluster has an internal DNS service
- Accessed with a static IP
- Hard-coded so that every container can find it
- Every service is registered with the DNS so that all components can find every Service on the cluster by **NAME**
- Is based on CoreDNS (https://coredns.io)

CORE KUBERNETES COMPONENTS

- Kubernetes DNS
- Pods
- Services

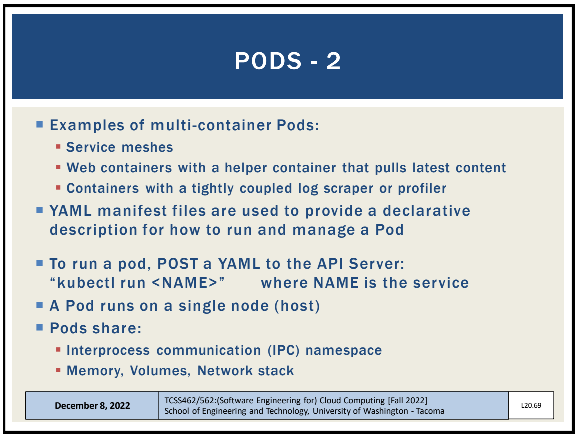


PODS

- Pod – atomic unit of deployment & scheduling in Kubernetes
- A Kubernetes Pod is defined to run a containerized application
- Kubernetes manages Pods, not individual containers
- Cannot run a container directly on Kubernetes
- All containers run through Pods

- Pod comes from "pod of whales"
- Docker logo shows a whale with containers stacked on top
- Whale represents the Docker engine that runs on a single host
- Pods encapsulate the definition of a single microservice for hosting purposes
- Pods can have a single container, or multiple containers if the service requires more than one

68

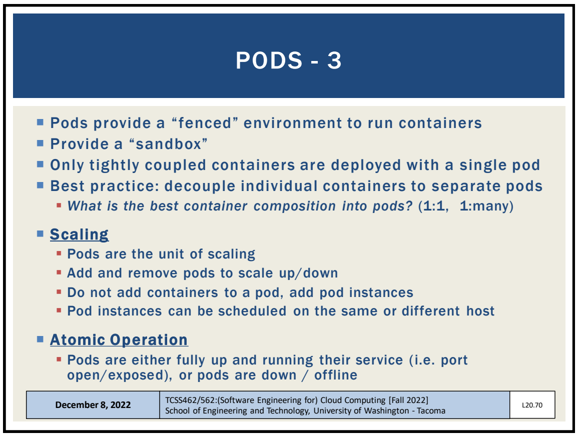


PODS - 2

- Examples of multi-container Pods:
  - Service meshes
  - Web containers with a helper container that pulls latest content
  - Containers with a tightly coupled log scraper or profiler
- YAML manifest files are used to provide a declarative description for how to run and manage a Pod

- To run a pod, POST a YAML to the API Server: "kubectl run <NAME>" where NAME is the service
- A Pod runs on a single node (host)
- Pods share:
  - Interprocess communication (IPC) namespace
  - Memory, Volumes, Network stack

69



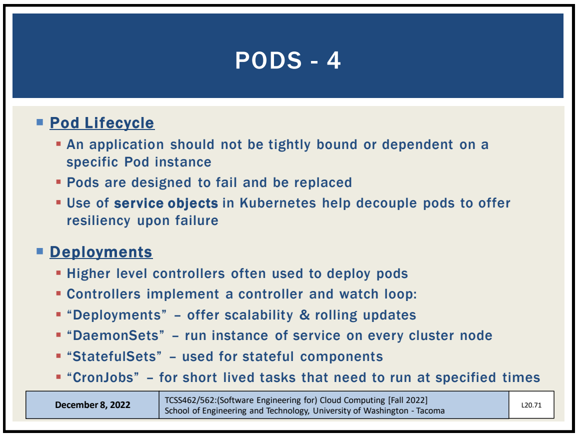
PODS - 3

- Pods provide a "fenced" environment to run containers
- Provide a "sandbox"
- Only tightly coupled containers are deployed with a single pod
- Best practice: decouple individual containers to separate pods
  - *What is the best container composition into pods?* (1:1, 1:many)

- **Scaling**
  - Pods are the unit of scaling
  - Add and remove pods to scale up/down
  - Do not add containers to a pod, add pod instances
  - Pod instances can be scheduled on the same or different host

- **Atomic Operation**
  - Pods are either fully up and running their service (i.e. port open/exposed), or pods are down / offline

70

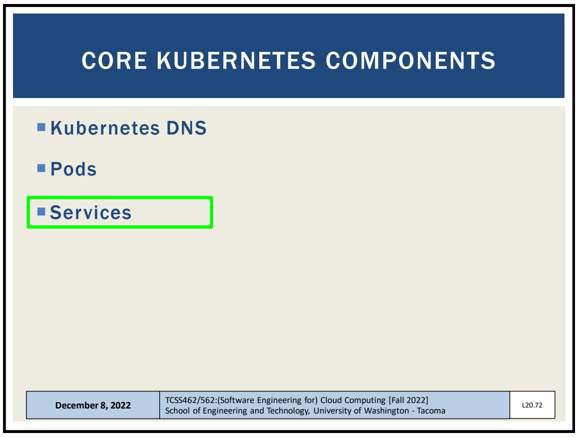


PODS - 4

- **Pod Lifecycle**
  - An application should not be tightly bound or dependent on a specific Pod instance
  - Pods are designed to fail and be replaced
  - Use of **service objects** in Kubernetes help decouple pods to offer resiliency upon failure

- **Deployments**
  - Higher level controllers often used to deploy pods
  - Controllers implement a controller and watch loop:
  - "Deployments" – offer scalability & rolling updates
  - "DaemonSets" – run instance of service on every cluster node
  - "StatefulSets" – used for stateful components
  - "CronJobs" – for short lived tasks that need to run at specified times

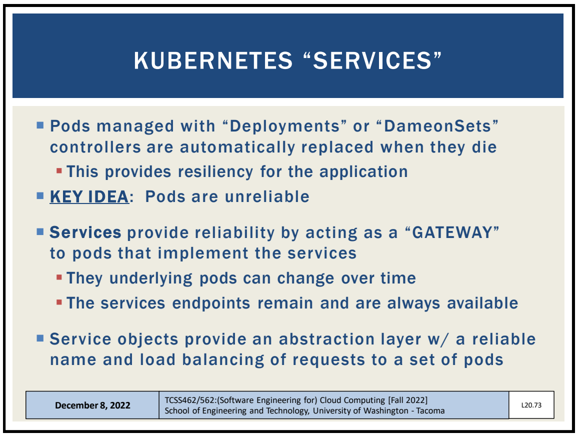
71



CORE KUBERNETES COMPONENTS

- Kubernetes DNS
- Pods
- Services

72



KUBERNETES "SERVICES"

- Pods managed with "Deployments" or "DameonSets" controllers are automatically replaced when they die
  - This provides resiliency for the application
- **KEY IDEA**: Pods are unreliable

- **Services** provide reliability by acting as a "GATEWAY" to pods that implement the services
  - They underlying pods can change over time
  - The services endpoints remain and are always available

- Service objects provide an abstraction layer w/ a reliable name and load balancing of requests to a set of pods

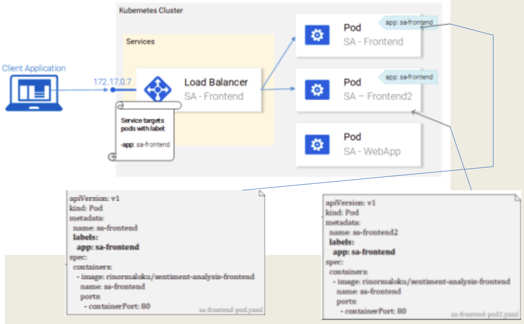
73

SERVICES

- Provide reliable front-end with:
  - Stable DNS name
  - IP Address
  - Port
- Services do not posses application intelligence
- No support for application-layer host and path routing
- Services have a “label selector” which is a set of lables
- Requests/traffic is only sent to Pods with matching labels
- Services only send traffic to healthy Pods
- **KEY IDEA**: Services bring stable IP addresses and DNS names to unstable Pods

December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.74

74



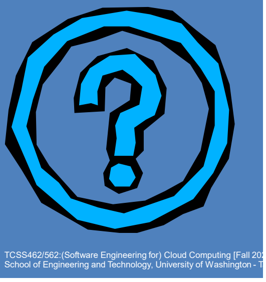
```
apiVersion: v1
kind: Pod
metadata:
  name: sa-frontend
  labels:
    app: sa-frontend
spec:
  containers:
    - image: rinormaloku/sentiment-analysis-frontend
      name: sa-frontend
      ports:
        - containerPort: 80
```

```
apiVersion: v1
kind: Pod
metadata:
  name: sa-frontend2
  labels:
    app: sa-frontend
spec:
  containers:
    - image: rinormaloku/sentiment-analysis-frontend
      name: sa-frontend
      ports:
        - containerPort: 80
```

December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.75

75

QUESTIONS



December 8, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L20.76

76