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

Class Presentations, Day 2

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

1

OFFICE HOURS – THIS WEEK

- **THIS WEEK**
- **Tuesday: (After Quiz)**
  - 6:30 to 7:30 pm - CP 229 & Zoom
- **Thursday (After Class) :**
  - 6:00 pm to 7:00 pm – CP 229 & Zoom
- Or email for appointment

> *Office Hours set based on Student Demographics survey feedback*

2

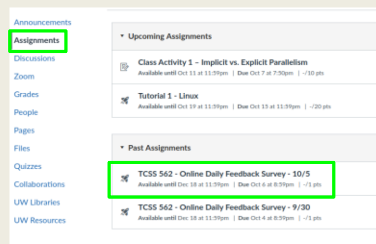
OBJECTIVES – 12/2

- **Questions from 11/25**
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 9: AWS Step Functions, AWS SQS
- Tutorial 8: Serverless Beyond Java, Container-Based Functions
- Kubernetes

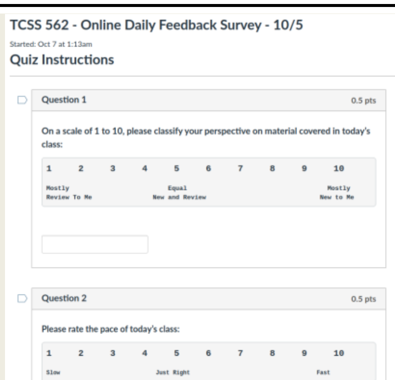
3

ONLINE DAILY FEEDBACK SURVEY

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



4



5

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (42 respondents, 23 in-person, 19 online):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.17 (↓ - *previous 6.47*)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.12 (↓ - previous 5.16)**

6

FEEDBACK FROM 12/2

- ***Is term project due on final week?***
- The term project is due on Friday December 12th AOE
- Which is Saturday December 13th at 4:59 am
- Canvas will close on Saturday December 13th at 11:59 am

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.7

7

FEEDBACK - 2

- ***Is LLM use allowed on term project?***
- Yes. One of the term project recommended case studies is to:
  - #1: Implement a data processing pipeline in multiple programming languages (e.g. Java, Python) on AWS Lambda, and compare the performance, where LLMs help convert/write code from Java to Python (or other languages)
  - #2: Implement a data processing pipeline in one programming language (Java), but implement the pipeline multiple times using different LLMs to determine which LLMs generate better code
    - The recommendation is to write detailed prompts to minimize conversations with the LLM. Refactor missing details to create long prompts which can be reused to compare alternate LLMs
    - Evaluate code syntactical correctness, functional correctness, serverless performance

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.8

8

SPEEDING UP DATA INSERTS WITH MYSQL

- Executing each INSERT query separately is prohibitively slow if looking to load 100,000+ rows
- AWS Lambda functions can time out when loading Large CSV files (1,000,000)
- GOAL: Improve the performance of your LOAD function to enable ingestion of 1+ million rows of data
- Techniques:

1. **SQL Batch Queries**
2. **Prepared Statements**
3. **Stored Procedures**

- ChatGPT recommends combining the use of SQL Batch Queries with Prepared Statements for maximum speed-up

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.9

9

OBJECTIVES – 12/2

- Questions from 11/25
- **Tutorials Questions**
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 9: AWS Step Functions, AWS SQS
- Tutorial 9: Serverless Beyond Java,Container-Based Functions
- Kubernetes

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.10

10

TUTORIAL 6 – CLOSED

- Introduction to Lambda III: Serverless Databases
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2025_tutorial_6.pdf
- Create and use Sqlite databases using sqlite3
- Deploy Lambda function with Sqlite3 database under /tmp
- Compare in-memory vs. file-based Sqlite DBs on Lambda
- Create an Amazon Aurora "Serverless" v2 MySQL database
- Using the AWS CloudShell in the same VPC (Region + availability zone) connect and interact your Aurora serverless database using the mysql CLI app
- Deploy an AWS Lambda function that uses the MySQL "serverless" database
- **'FREE PLAN': okay to use db.t3.micro, db.t4g.micro RDS MySQL VM – must indicate use of RDS VM on tutorial PDF**

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.11

11

TUTORIAL 7 – DEC 4

- Introduction to Docker
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2025_tutorial_7.pdf
- Must complete using c7i-flex.large ec2 instance & Ubuntu 24.04 (for cgroups v2)
- Use DOCX file for copying and pasting Docker install commands
- Topics:
  - Installing Docker
  - Creating a container using a Dockerfile
  - Using cgroups virtual filesystem to monitor CPU utilization of a container
  - Persisting container images to Docker Hub image repository
  - Container vertical scaling of CPU/memory resources
  - Testing container CPU and memory isolation

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.12

12

OBJECTIVES – 12/2

- Questions from 11/25
- Tutorials Questions
- **Class Presentations Schedule - Cloud Technology or Research Paper Review**
- Tutorial 9: AWS Step Functions, AWS SQS
- Tutorial 8: Serverless Beyond Java, Container-Based Functions
- Kubernetes

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.13

13

GROUP PRESENTATIONS

- TWO OPTIONS:
- ***Cloud technology presentation***
- ***Cloud research paper presentation***
  - Recent & suggested papers will be posted at: http://faculty.washington.edu/wlloyd/courses/tcss562/papers/
- Presentation dates:
  - Tuesday November 25
  - Tuesday December 2, Thursday December 4
- Peer Reviews
  - Word DOCX review form posted, fill out, submit PDF on Canvas
  - Feedback shared with groups
  - TCSS 462: submit 4 total peer reviews in lieu of a group presentation

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.14

14

GROUP PRESENTATIONS

- 7 Presentation Teams
- 3 Cloud Technology Talks
- 4 Cloud Research Paper Presentations
- 1 one-person teams
- 2 two-person teams
- 4 three-person teams

- Thank you for the submissions

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.15

15

PRESENTATION SCHEDULE

- <Tuesday November 25>

1. Team 4: Xiaoling Wei, Bohan Xiong, Xu Zhu
Research paper: ***Serverless Replication of Object Storage across Multi-Vendor Clouds and Regions***
2. Team 1: William Hay
Cloud Technology: ***Amazon Athena***
3. *Robert Cordingly – Original Research Paper:* ***Sky Computing for Serverless: Infrastructure Assessment to Support Performance Enhancement*** *(IEEE/ACM UCC 2025 Practice Talk)*

- <Tuesday December 2>

1. Team 5: Sparsha Jha, Chris Biju
Cloud Technology: ***Intelligent Optimization of Distributed Pipeline Execution in Serverless Platforms: A Predictive Model Approach***

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.16

16

PRESENTATION SCHEDULE - 2

- <Thursday December 4>

1. Team 3: Jiameng Li, Naomi Nottingham, Headley Brissett
Research paper: ***A Perfect Fit? – Towards Containers on Microkernels***
2. Team 2: Ruby Plangphatthanaphanit, Junjia Li, Ari Yin
Cloud Technology: ***CI/CD in the Cloud (GitHub Actions + Cloud Deploy)***
3. Team 8: Aamena Suzzane, Dhruva Bhat
Research paper: ***CoFaaS: Automatic Transformation-based Consolidation of Serverless Functions***
4. Team 6: Han Zhang, Sahil Bhatt, Pengcheng Cao
Cloud Technology: ***AWS Amplify***

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.17

17

OBJECTIVES – 12/2

- Questions from 11/25
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- **Tutorial 9: AWS Step Functions, AWS SQS**
- Tutorial 8: Serverless Beyond Java, Container-Based Functions
- Kubernetes

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.18

18

TUTORIAL 9 – TO BE POSTED

- Introduction to AWS Step Functions and Amazon Simple Queue Service (SQS)
- **Not Required**, available for **EXTRA CREDIT** (scored out of 0)
  - adds points to overall tutorials score
- Tasks
  - Adapt Caesar Cipher Lambda functions for use with AWS Step Functions
  - Create AWS Step Functions State Machine
  - Create a BASH client to invoke the AWS Step Function
  - Create Simple Queue Service Queue for messages
  - Add message to SQS queue from AWS Lambda function
  - Modify AWS Step Function Bash client script to retrieve AWS Step Function result from SQS queue

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.19

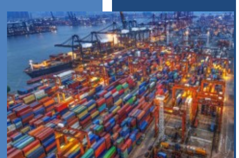
19

OBJECTIVES – 12/2

- Questions from 11/25
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 9: AWS Step Functions, AWS SQS
- **Tutorial 8: Serverless Beyond Java, Container-Based Functions**
- Kubernetes

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.20

20



TUTORIAL 8: SERVERLESS BEYOND JAVA, CONTAINER-BASED FUNCTIONS

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.21

21

TUTORIAL 9

- **Python Based AWS Lambda Functions w/ SAAF, and Container-Based AWS Lambda Functions**
- **Not Required**, available for **EXTRA CREDIT** (scored out of 0)
  - adds points to overall tutorials score
  - 10 pts for Python Functions / 15 pts for Container Based Function
- Tasks
  - Build/Deploy/Test Python-based Lambda Functions
  - Deploy and Test Container Based AWS Lambda Function
    - Requires Docker Engine installation on local VM
  - Create role to use CLI/publish script
  - Use a config file to specify container-based function details
  - Update bash script to deploy hello function
  - Build Docker container locally, Publish to Elastic Container Registry
  - Create new 'hello' Lambda Function based on Container image
  - Test Container-based 'hello' AWS Lambda Function
  - Adapt your function to run sysbench prime number generation
  - Test prime number generation performance on AWS Lambda vs. memory

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.22

22

WE WILL RETURN AT ~4:55 PM

23

OBJECTIVES – 12/2

- Questions from 11/25
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 9: AWS Step Functions, AWS SQS
- Tutorial 9: Serverless Beyond Java,Container-Based Functions
- **Kubernetes**

December 2, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L17.24

24

KUBERNETES

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

L17.25

25

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

26

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

27

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 represent 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

28

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

29

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

30

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

31

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 the goal of keeping it running smoothly

32

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"

33

LOOK AHEAD: 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**

docker

34

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*

35

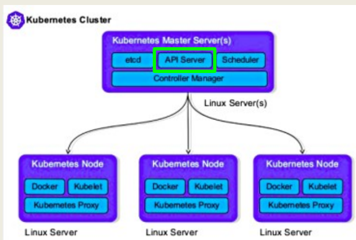
KUBERNETES MASTERS

- Provide system services to host the control plane
- Simplest clusters use only 1 master – (i.e. 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

36

MASTER SERVICES

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



37

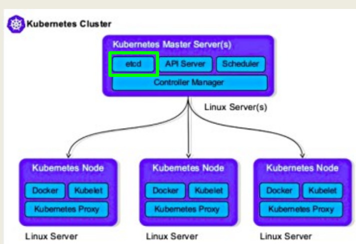
API SERVER

- 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

38

MASTER SERVICES

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



39

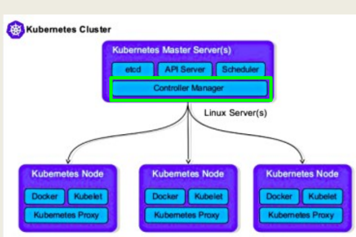
CLUSTER STORE

- Used to persist Kubernetes cluster state information
- 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

40

MASTER SERVICES

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



41

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:

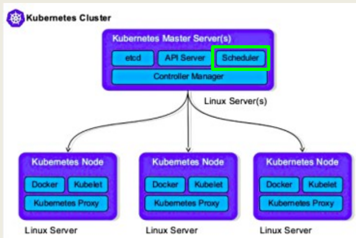
1. Obtain desired state (defined in manifest YAMLs)
2. Observe the current state
3. Determine differences
4. Reconcile differences

- Controllers are specialized to manage a specific resource type
  - They are not aware/concerned with of other parts of the system

42

MASTER SERVICES

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



43

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

44

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

45

RANKING NODES

- Remaining nodes are ranked based on for example:

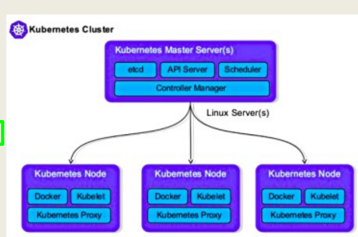
1. Does the node have the required images?
   - Cached images will lead to faster deployment time
2. How much free capacity (CPU, memory) does the node have?
3. How many tasks is the node already running?

- 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*

46

MASTER SERVICES

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



47

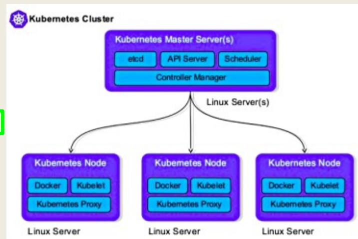
CLOUD CONTROLLER MANAGER

- 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.

48

MASTER SERVICES

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



49

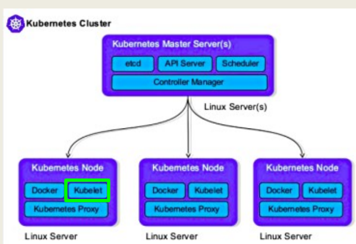
WORKER NODES

- 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

50

WORKER NODES

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



51

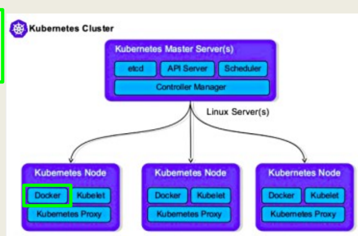
KUBELET

- 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

52

WORKER NODES

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



53

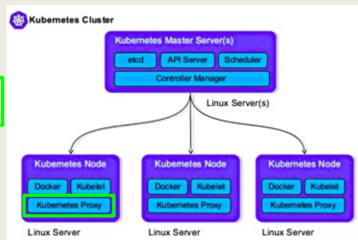
CONTAINER RUNTIME(S)

- 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

54

WORKER NODES

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



55

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

56

CORE KUBERNETES COMPONENTS

- Kubernetes DNS
- Pods
- Services

57

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)

58

CORE KUBERNETES COMPONENTS

- Kubernetes DNS
- Pods
- Services

59

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

60

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

61

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

62

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

63

CORE KUBERNETES COMPONENTS

- Kubernetes DNS
- Pods
- Services

64

KUBERNETES "SERVICES"

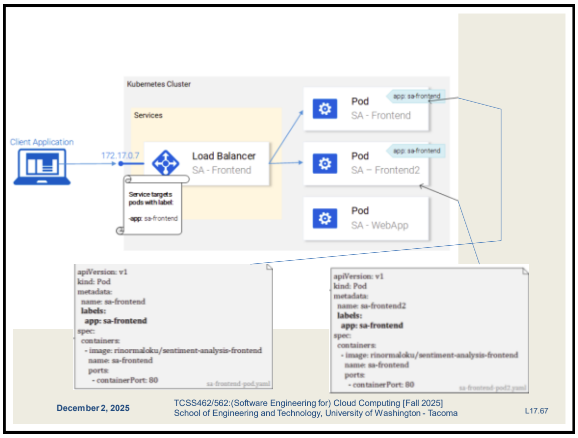
- Pods managed with "Deployments" or "DameonSets" controllers are automatically replaced when they die
  - This provides resiliency for the application
- **KEY IDEA**: Pods are unreliable
- **Services** provide reliability by acting as a "GATEWAY" to pods that implement the services
  - They underlying 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

65

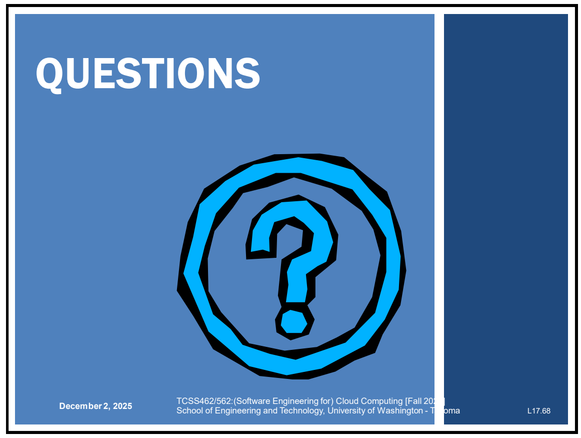
SERVICES

- 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

66


Kubernetes Cluster
Services
Client Application
172.17.0.7
Load Balancer
SA - Frontend
Service targets pods with label:
-app: sa-frontend
Pod
SA - Frontend
app: sa-frontend
Pod
SA – Frontend2
app: sa-frontend
Pod
SA - WebApp
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
sa-frontend-pod.yaml
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
sa-frontend-pod2.yaml
December 2, 2025
TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025]
School of Engineering and Technology, University of Washington - Tacoma
L17.67

67


QUESTIONS
December 2, 2025
L17.68

68