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

Containerization II & Kubernetes

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

1

OFFICE HOURS – FALL 2023

- **THIS WEEK**
- **Tuesdays:**
  - 2:30 to 3:30 pm - CP 229
- Or email for appointment

> Office Hours set based on Student Demographics survey feedback

2

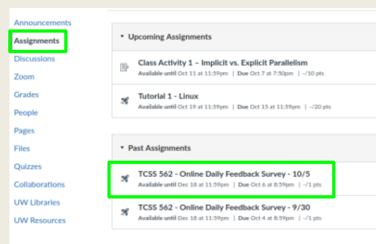
OBJECTIVES – 11/21

- Questions from 11/16
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 8: AWS Step Functions, AWS SQS
- Containerization
- Kubernetes

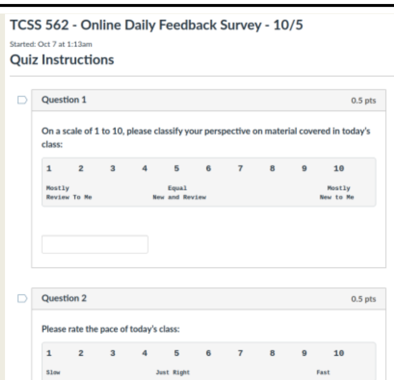
3

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 (56 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.38 (↑ - *previous 5.45*)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.48 (↑ - previous 5.33)**

- **Response rates:**
- TCSS 462: 34/44 – 77.3%
- TCSS 562: 22/25 – 88.0%

6

FEEDBACK FROM 11/16

- ***Why is it advantageous for containers to be run on top of VMs?***
- ***Why is it advantageous for containers to be run on top of bare metal?***

7

FEEDBACK - 2

- ***Since services like Fargate also use VM as management system, what is advantage of it compared to running container on a EC2 instance?***
- AWS Fargate is fully managed and comes with its own GUI, CLI, and API
- There is no need to install Docker onto an ec2 instance
- There is no need to create and manage ec2 instances and Docker installations
- You simply run a container image, Fargate provides the requested infrastructure on request
- There is no VM or cluster to manage

8

AWS CLOUD CREDITS UPDATE

- AWS CLOUD CREDITS ARE NOW AVAILABLE FOR TCSS 462/562
- Credits provided on request with expiry of Sept 30, 2024
- Credit codes must be securely exchanged
- Request codes by sending an email with the subject "**AWS CREDIT REQUEST**" to **wlloyd@uw.edu**
- Codes can also be obtained in person (or zoom), in the class, during the breaks, after class, during office hours, by appt
  - 61 credit requests fulfilled as of Nov 13 @ 11:59p
- Codes not provided using discord

9

Don't Forget to Terminate (Shutdown) all EC2 instances for Tutorials 3 & 7

Spot instances:
c5d.large instance @ ~3c cents / hour

$0.72 / day
$5.04 / week
$21.88 / month
$262.80 / year

AWS CREDITS → → → → → → → → →

10

OBJECTIVES – 11/21

- Questions from 11/16
- **Tutorials Questions**
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 8: AWS Step Functions, AWS SQS
- Containerization
- Kubernetes

11

TUTORIAL 6 – ~~NOV 21~~ NOV 22

- Introduction to Lambda III: Serverless Databases
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2023_tutorial_6.pdf
- Create and use Sqlite databases using sqlite3 tool
- 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 an ec2 instance in the same VPC (Region + availability zone) connect and interact with the database using the mysql CLI app
- Deploy an AWS Lambda function that uses the MySQL "serverless" database

12

TUTORIAL 7 – DEC 1

- Introduction to Docker
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2023_tutorial_7.pdf
- Complete tutorial using Ubuntu 22.04 (for cgroups v2)
- Complete using **c5.large ec2 instance** (for consistency)
- 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

13

OBJECTIVES – 11/21

- Questions from 11/16
- Tutorials Questions
- **Class Presentations Schedule - Cloud Technology or Research Paper Review**
- Tutorial 8: AWS Step Functions, AWS SQS
- Containerization
- Kubernetes

14

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 22, Tuesday November 29
  - Tuesday December 6, Thursday December 8
- **Peer Reviews**
  - Word DOCX form will be provided, fill out, submit PDF 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

15

GROUP PRESENTATIONS

- 9 Presentation Teams
- 4 Cloud Technology Talks
- 5 Cloud Research Paper Presentations
- 2 two-person teams
- 7 three-person teams
- Thank you for the submissions

16

PRESENTATION SCHEDULE

- **Tuesday November 28**

1. Lucas Lu, Yexuan Gao, Christopher Henderson (team 3)
Research paper: Research Paper: The Gap between Serverless Research and Real-world Systems
2. Daniil Filienko, Xuchong (Nicolas) Du, Preethika Pradeep (team 1)
Cloud Technology: Amazon Sagemaker (ML)

- **Thursday November 30**

1. Vishnu Priya Rajendran, Malavika Suresh, Alekhya Parisha (team 5)
Cloud Technology: Amazon DynamoDB
2. Heyuan Wang, Baiqiang Wang, Lynn Yang (team 2)
Cloud Technology: Amazon Elastic Kubernetes Service (EKS)

17

PRESENTATION SCHEDULE - 2

- **Tuesday December 5**

1. Kewei Liu, Sherry Liu (team 15)
Research paper: AWSomePy : A Dataset and Characterization of Serverless Applications
2. Sanjay Vuppugandla, Sai Prateek Atluri, Ankit Kadian (team 9*)
Research paper: Lukewarm Serverless Functions: Characterization and Optimization *(* - team 9 can swap with team 6, 7, or 8 if agreed)*

- **Thursday December 7**

1. Cynthia Pang, Lifan Cao (team 6)
Research paper: Evicting for the Greater Good: The Case for Reactive Check Pointing in Serverless Computing
2. Srishty, Angela C Farin, Tomoki Kusunoki (team 7)
Cloud Technology: Amazon Redshift
3. Xiaoqing Zhou, Mary Yang, Micaela Nomakchteinsky (team 8)
Research paper: Research paper: Rendezvous - Where Serverless Functions Find Consistency

18

OBJECTIVES – 11/21

- Questions from 11/16
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- **Tutorial 8: AWS Step Functions, AWS SQS**
- Containerization
- Kubernetes

19

TUTORIAL 8 – TO BE POSTED

- Introduction to AWS Step Functions and Amazon Simple Queue Service (SQS)
- Not Required, available for extra credit
  - adds points to overall tutorials score
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2023_tutorial_8.pdf
- 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

20

OBJECTIVES – 11/21

- Questions from 11/16
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 8: AWS Step Functions, AWS SQS
- **Containerization**
- Kubernetes

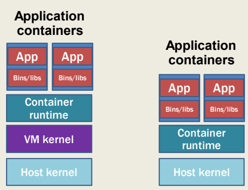
21

CONTAINERIZATION

22

CONTAINER ISOLATION

- Is the host isolated from application containers?
- Are application containers isolated from each other?



23

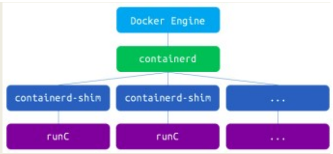
LXC (LINUX CONTAINERS)

- Operating system level virtualization
- Run multiple isolated Linux systems on a host using a single Linux kernel
- Control groups(cgroups)
  - Including in Linux kernels => 2.6.24
  - Limit and prioritize sharing of CPU, memory, block/network I/O
- Linux namespaces
- Docker initially based on LXC

24

OTHER DOCKER TOOLS

- **Docker Machine:** automatically provision and manage sets of docker hosts to form a cluster



- **Docker Swarm**: Clusters multiple docker hosts together to manage as a cluster.
- **Docker Compose**: Config file (YAML) for multi-container application; Describes how to deploy and configure multiple containers

25

CONTAINER ORCHESTRATION FRAMEWORKS

- Framework(s) to deploy multiple containers
- Provide container clusters using cloud VMs
- Similar to "private clusters"
- Reduce VM idle CPU time in public clouds
- Better leverage "sunk cost" resources
- Compact multiple apps onto shared public cloud infrastructure
- Generate to cost savings
- Reduce vendor lock-in

26

KEY ORCHESTRATION FEATURES

- Management of container hosts
- Launching set of containers
- Rescheduling failed containers
- Linking containers to support workflows
- Providing connectivity to clients outside the container cluster
- Firewall: control network/port accessibility
- Dynamic scaling of containers: horizontal scaling
  - Scale in/out, add/remove containers
- Load balancing over groups of containers
- Rolling upgrades of containers for application

27

CONTAINER ORCHESTRATION FRAMEWORKS - 2

- Docker swarm
- Apache mesos/marathon
- Kubernetes
  - Many public clouds now offer managed services to host Kubernetes clusters
  - Amazon Elastic Kubernetes Service (EKS), Azure Kubernetes Service (AKS), Google Kubernetes Engine (GKE)
- Amazon elastic container service (ECS)
- Apache aurora (retired project based on Mesos)

- Container-as-a-Service
  - Serverles containers without managing clusters
  - Azure Container Instances, AWS Fargate…

28

OBJECTIVES – 11/21

- Questions from 11/16
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 8: AWS Step Functions, AWS SQS
- Containerization
- **Kubernetes**

29

KUBERNETES

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

30

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

31

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

32

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

33

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

34

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

35

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

36

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

37

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"

38

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*

39

WE WILL RETURN AT ~4:50 PM

40

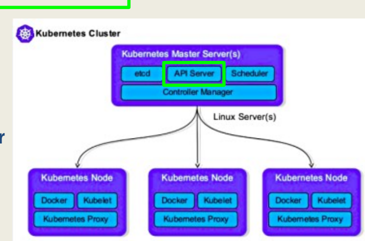
KUBERNETES MASTERS

- 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

41

MASTER SERVICES

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



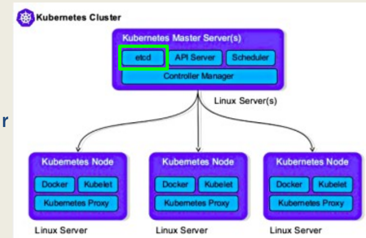
42

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

MASTER SERVICES

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

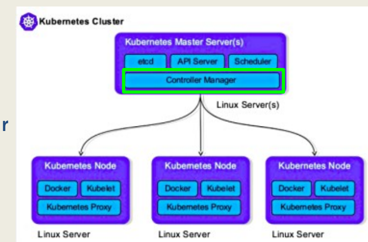


CLUSTER STORE

- 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

MASTER SERVICES

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



CONTROLLER MANAGER

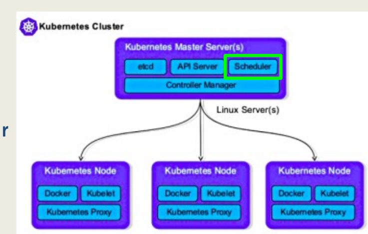
- Provides a "controller" of the controllers
  - Implements background control loops to monitor cluster and respond to events
  - Control loops include: node controller, endpoints controller, replicaset controller, etc…
- **GOAL**: ***ensure cluster current state matches desired state***
- Control Loop 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

MASTER SERVICES

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



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

49

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

50

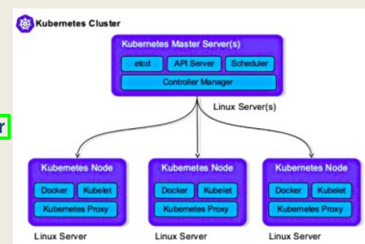
RANKING NODES

- 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*

51

MASTER SERVICES

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



52

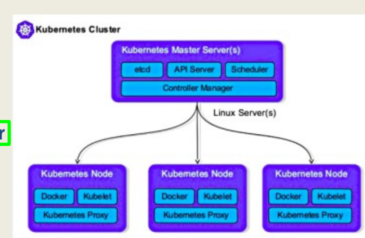
CLOUD CONTROLLER MANAGER

- 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.

53

MASTER SERVICES

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



54

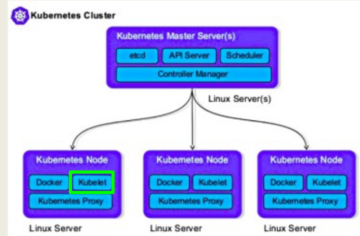
WORKER NODES

- 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

55

WORKER NODES

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



56

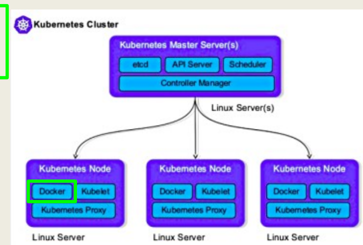
KUBELET

- 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

57

WORKER NODES

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



58

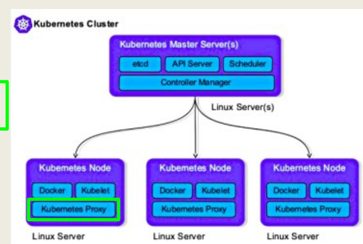
CONTAINER RUNTIME(S)

- 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 3^{rd} party container runtimes to plug-in to
- Popular container runtimes: Docker, containerd, Kata

59

WORKER NODES

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



60

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

61

CORE KUBERNETES COMPONENTS

- Kubernetes DNS
- Pods
- Services

62

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)

63

CORE KUBERNETES COMPONENTS

- Kubernetes DNS
- Pods
- Services

64

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

65

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

66

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

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

CORE KUBERNETES COMPONENTS

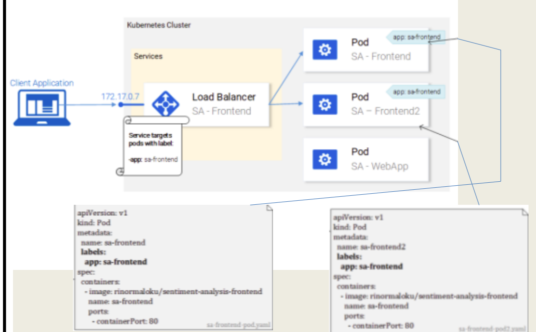
- Kubernetes DNS
- Pods
- Services

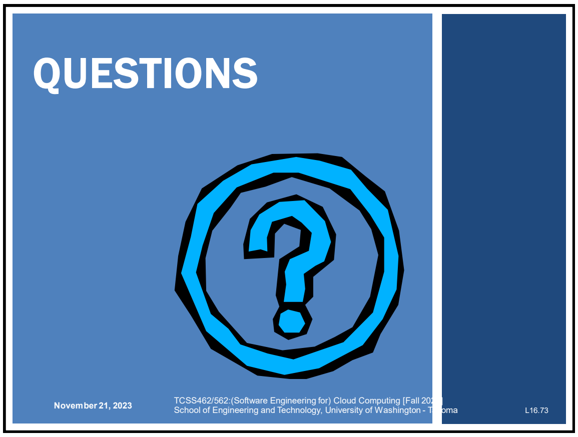
KUBERNETES "SERVICES"

- 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

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





73