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

Tutorial 9: Serverless Beyond Java, Containter-Based Functions

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

1

OFFICE HOURS – FALL 2024

- **THIS WEEK**
- **Tuesday:**
  - 2:30 to 3:30 pm - CP 229
- **Friday *:**
  - 1:30 pm to 2:30 pm –via Zoom*
- Or email for appointment

> Office Hours set based on Student Demographics survey feedback

2

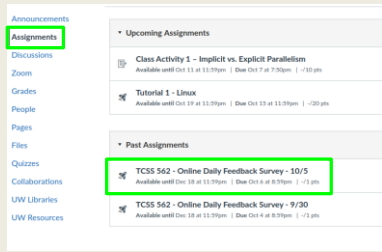
OBJECTIVES – 11/21

- **Questions from 11/19**
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 8: AWS Step Functions, AWS SQS
- Tutorial 9: Serverless Beyond Java, Container-Based Functions
- Kubernetes

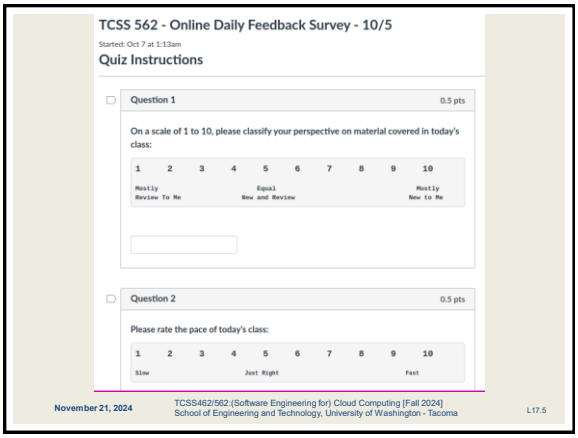
3

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 (**36** respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.36 (↑ - *previous 5.31*)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.83 (↑ - previous 5.10)**

- **Response rates:**
- TCSS 462: 25/42 – 59.2%
- TCSS 562: 11/20 – 55.0%

6

FEEDBACK FROM 11/19

- ***Could you use Docker to run legacy software on a personal computer?***
- Yes, legacy software can be containerized to provide legacy library dependencies from earlier versions of Linux, tc.
- ***What makes Docker different from an emulator?***
- Emulators use hardware and software to allow a computer system to behave like another computer system to enable running applications and services designed for the foreign computer
- Example: Rosetta on M1 ARM Mac for running x86_64 apps
- Docker does not mimic any computer system. It divides and shares the Linux kernel to enable sandboxes with different root filesystems that can have different versions of Linux tools, shells, compilers, interpreters (Python), etc.

7

FEEDBACK - 2

- **How is it determined what is layered and what is cached in the Dockerfile ?**
- *FROM statements are used in Dockerfiles to specify layers*
  - *There can be multiple 'FROM' statements*
- *Base layers are downloaded once and cached*
- *Everything, once built, is cached until there is a change, then layer(s) are repulled, cached, and the container image rebuilt*
- **For ther term project, do we submit any of the code used for the project?**
- *Yes, in addition to the presentation or paper, groups submit ALL project source code in a tar.gz or zip file to Canvas*

8

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

Tutorial 3 spot instance:
c5d.large instance @ ~3.2 cents / hour

$0.78 / day
$5.48 / week
$23.78 / month
$285.42 / year

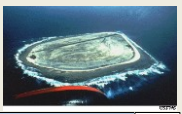
AWS CREDITS → → → → → → → →

9

TUTORIAL SUBMISSION TIME

- Tutorials can now be submitted on the due date until the very last minute of the day **Anywhere-on-Earth (AOE)**
  - **Equivalent to 4:59 AM Pacific Standard Time (PST)**
- Anywhere-on-Earth timezone: **Baker Island, Pacific Ocean**
- https://www.timeanddate.com/time/zones/aoe
- Uninhabited island in Pacific Ocean

Coordinates	0°11′45″N 176°28′45″W
Area	2.1 km2 (0.81 sq mi)
Length	1.81 km (1.125 mi)
Width	1.13 km (0.702 mi)
Coastline	4.8 km (2.98 mi)
Highest elevation	8 m (26 ft)
Population	0 (2000)



10

OBJECTIVES – 11/21

- Questions from 11/19
- **Tutorials Questions**
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 8: AWS Step Functions, AWS SQS
- Tutorial 9: Serverless Beyond Java,Container-Based Functions
- Kubernetes

11

TUTORIAL 6 – NOV 23

- Introduction to Lambda III: Serverless Databases
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2024_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_f2024_tutorial_7.pdf
- Complete tutorial using Ubuntu 24.04 (for cgroups v2)
- Complete using **c6i.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 11/19
- Tutorials Questions
- **Class Presentations Schedule - Cloud Technology or Research Paper Review**
- Tutorial 8: AWS Step Functions, AWS SQS
- Tutorial 9: Serverless Beyond Java, Container-Based Functions
- 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 26
  - Tuesday December 3, Thursday December 5
- **Peer Reviews**
  - Word DOCX form will be provided, fill out, submit PDF on Canvas
  - Feedback shared with groups
  - TCSS 462: submit 4 total peer reviews in lieu of a group presentation

15

GROUP PRESENTATIONS

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

- Thank you for the submissions

16

PRESENTATION SCHEDULE

- **<Tuesday November 26>**

1. Soumith Kondubhotla, Siva Srinivasa Aditya, Sri Mylavarapu
Research paper: ***Sandboxing Functions for Efficient and Secure Multi-tenant Serverless Deployments***
2. Mingzhi Ma, Derry Cheng, Aaron Chen
Research paper: ***Serverless? RISC more!***
3. Ishwarya Narayana Subramanian, Thanvi Yadav Sirla
Cloud Technology: ***Azure Kubernetes Service***
4. Steven Golob
Research paper: ***Tiny Autoscalers for Tiny Workloads: Dynamic CPU Allocation for Serverless Functions***

- **<Tuesday December 3>**

1. Andrew Nguyen, Pavel Braginskiy
Cloud Technology: ***AWS Amplify***

17

PRESENTATION SCHEDULE - 2

- **<Thursday December 5>**

1. Viktoria Dolojan and Carla Peterson
Research paper: ***FootPrinter: Quantifying Data Center Carbon Footprint***
2. Andrew Jang, Shrey Srivastava, Naga
Cloud Technology: ***SageMaker: training configurations***
3. Roark Zhang
Research paper: ***Process-as-a-Service: Unifying Elastic and Stateful Clouds with Serverless Processes***
4. Sanya Sinha, Jackson Davis
Research paper: ***Goldfish: Serverless Actors with Short-Term Memory State for the Edge-Cloud Continuum***

18

OBJECTIVES – 11/21

- Questions from 11/19
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- **Tutorial 8: AWS Step Functions, AWS SQS**
- Tutorial 9: Serverless Beyond Java, Container-Based Functions
- Kubernetes

19

TUTORIAL 8 – TO BE POSTED

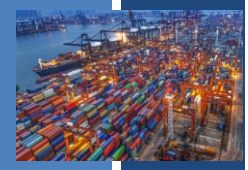
- Introduction to AWS Step Functions and Amazon Simple Queue Service (SQS)
- Not Required, available for extra credit (scored out of 0)
  - adds points to overall tutorials score
- Tasks
  - Adapt Caesar Cipher Lambda functions for use with AWS Step Functions
  - Create AWS Step Functions State Machine
  - 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

20

OBJECTIVES – 11/21

- Questions from 11/19
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 8: AWS Step Functions, AWS SQS
- **Tutorial 9: Serverless Beyond Java, Container-Based Functions**
- Kubernetes

21



TUTORIAL 9: SERVERLESS BEYOND JAVA, CONTAINER-BASED FUNCTIONS

22

TUTORIAL 9

- Available for up to 25 pts extra credit – tutorial category
  - Up to 2.3% boost to overall course grade
- One or two-person teams
- Immediate grading via demo in class, after class, office hours
- Asynchronous submission by Canvas
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2024_tutorial_9.pdf
- Profiling Python AWS Lambda functions with SAAF
- Profiling Container-based AWS Lambda functions with code or programs in any language with SAAF

23

WE WILL RETURN AT ~4:55 PM

24

OBJECTIVES – 11/21

- Questions from 11/19
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 8: AWS Step Functions, AWS SQS
- Tutorial 9: Serverless Beyond Java,Container-Based Functions
- **Kubernetes**

25

KUBERNETES

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

26

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

27

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

28

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

29

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

30

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
  - Shopping cart
  - Logging service
  - Catalog service
  - Authentication 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

31

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

32

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

33

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"

34

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**

35

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*

36

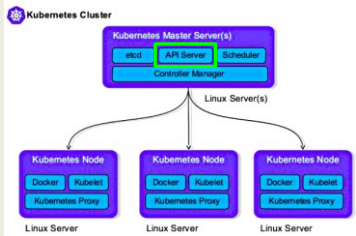
KUBERNETES MASTERS

- 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

37

MASTER SERVICES

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



38

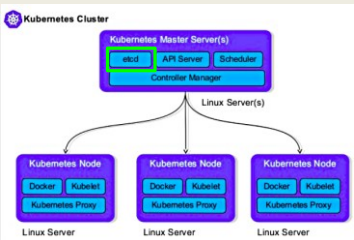
API SERVER

- 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

39

MASTER SERVICES

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



40

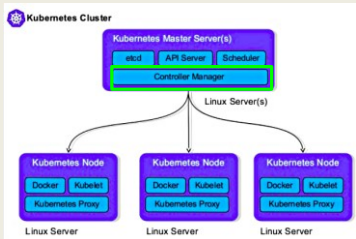
CLUSTER STORE

- 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

41

MASTER SERVICES

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



42

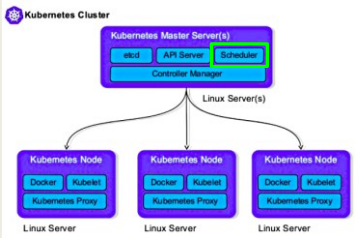
CONTROLLER MANAGER

- 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

43

MASTER SERVICES

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



44

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

45

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

46

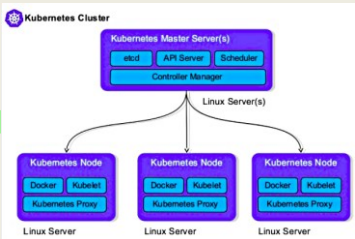
RANKING NODES

- 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*

47

MASTER SERVICES

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



48

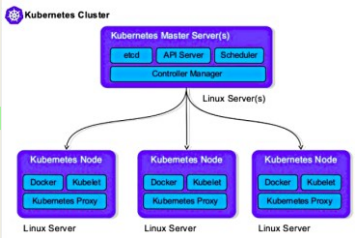
CLOUD CONTROLLER MANAGER

- 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.

49

MASTER SERVICES

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



50

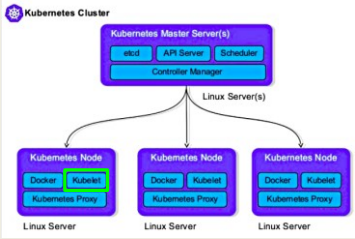
WORKER NODES

- 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

51

WORKER NODES

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



52

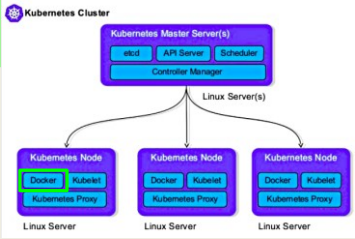
KUBELET

- 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

53

WORKER NODES

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



54

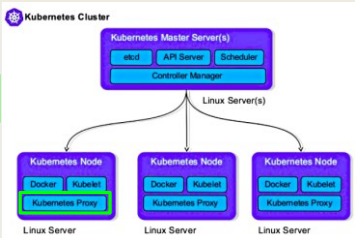
CONTAINER RUNTIME(S)

- 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

55

WORKER NODES

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



56

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

57

CORE KUBERNETES COMPONENTS

- Kubernetes DNS
- Pods
- Services

58

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)

59

CORE KUBERNETES COMPONENTS

- Kubernetes DNS
- Pods
- Services

60

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

61

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

62

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

63

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

64

CORE KUBERNETES COMPONENTS

- Kubernetes DNS
- Pods
- Services

65

KUBERNETES "SERVICES"

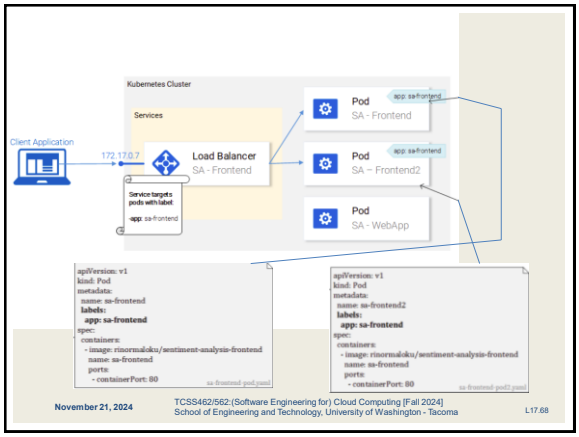
- Pods managed with "Deployments" or "DameonSets" controllers are automatically replaced when they die
  - This provides resiliency for the application
- **KEY IDEA**: Pods are unreliable
- **Services** provide reliability by acting as a "GATEWAY" to pods that implement the services
  - They underlying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

66

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

67



68

QUESTIONS

69