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

Team 3 & Team 1 Presentations, Kubernetes II

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

1

OFFICE HOURS – FALL 2023

- **THIS WEEK**
- **Tuesday:**
  - 2:30 to 3:30 pm - CP 229
- **Friday:**
  - 11:00 am to 12:00 pm – ONLINE via Zoom
- Or email for appointment

> Office Hours set based on Student Demographics survey feedback

2

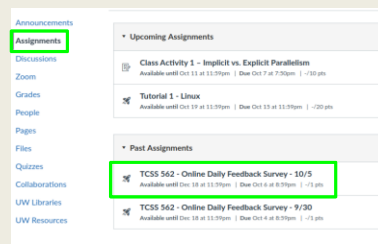
OBJECTIVES – 11/28

- Questions from 11/21
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 8: AWS Step Functions, AWS SQS
- Team 3: Research paper: Research Paper: The Gap between Serverless Research and Real-world Systems
- Team 1: Cloud Technology: Amazon Sagemaker (ML)
- Kubernetes

6

ONLINE DAILY FEEDBACK SURVEY

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



7

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 Review To Me — Equal New and Review — Mostly 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

8

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (52 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.19 (↑ - *previous 6.38*)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.69 (↑ - previous 5.48)**

- **Response rates:**
- TCSS 462: 33/44 – 75.0%
- TCSS 562: 19/25 – 76.0%

9

FEEDBACK FROM 11/21

- ***Could you reiterate what exactly is the different between a set of containers and traditional serverless services?***
- Serverless is a characteristic of cloud services which refers to not having to manage servers
  - On AWS we will not see VM types associated with creating "serverless" cloud services . . . There is no c5.large etc...
- Serverless != Function-as-a-Service (FaaS)
- FaaS is a cloud computing delivery model for hosting user functions which are similar to microservices using "serverless" infrastructure managed by the cloud provider
  - Microservices are small, independent deployable services that communicate with each other over a network

November 28, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L17.10

10

FEEDBACK - 2

- Containers are infrastructure abstractions for sharing a physical server which are lighter-weight than full virtual machines
- Containers can be used to encapsulate and host microservices
  - Encapsulate means capture all of the software dependencies together in one place for easy deployment
- Containers could also encapsulate and run a Bash script or an entire application

November 28, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L17.11

11

FEEDBACK - 3

- ***When would we prefer deploying a service as a set of docker containers managed by k8s? and when would it be more appropriate to host it on the cloud, managed by the provider?***
- By deploying as a set of docker containers on k8s, do you mean the service will be hosted on a private cluster?
- Note that k8s clusters can be created on public clouds for service hosting as well
- Hosting a service on the cloud for be done using a variety of "compute" centric cloud computing delivery models
- Service/microservices can be hosted with any of the following:
  - Infrastructure-as-a-Service: EC2
  - Container-as-a-Service: Fargate, ECS, EKS
  - Function-as-a-Service: AWS Lambda
- ***Or is it not exclusive, but perhaps we could utilize both?***
  - It is possible to combine compute models (IaaS, CaaS, FaaS) for service hosting, but less typical that using just one delivery model

November 28, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L17.12

12

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
  - 62 credit requests fulfilled as of Nov 27 @ 11:59p
- Codes not provided using discord

November 28, 2023 | TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L17.13

13

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 → → → → → → → →

14

OBJECTIVES – 11/28

- Questions from 11/21
- **Tutorials Questions**
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 8: AWS Step Functions, AWS SQS
- Team 3: Research paper: Research Paper: The Gap between Serverless Research and Real-world Systems
- Team 1: Cloud Technology: Amazon Sagemaker (ML)
- Kubernetes

November 28, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L17.15

15

TUTORIAL 6 – ~~NOV 21~~ NOV 28

- 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

16

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

17

OBJECTIVES – 11/28

- Questions from 11/21
- Tutorials Questions
- **Class Presentations Schedule - Cloud Technology or Research Paper Review**
- Tutorial 8: AWS Step Functions, AWS SQS
- Team 3: Research paper: Research Paper: The Gap between Serverless Research and Real-world Systems
- Team 1: Cloud Technology: Amazon Sagemaker (ML)
- Kubernetes

18

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 28, Tuesday November 30
  - Tuesday December 5, Thursday December 7
- **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

19

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

20

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)
3. Robert Cordingly: IEEE CloudCom Conference Paper - Practice Presentation: Addressing Serverless Computing Vendor Lock-In through Cloud Service Abstraction

21

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: Rendezvous - Where Serverless Functions Find Consistency

22

OBJECTIVES – 11/28

- Questions from 11/21
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- **Tutorial 8: AWS Step Functions, AWS SQS**
- Team 3: Research paper: Research Paper: The Gap between Serverless Research and Real-world Systems
- Team 1: Cloud Technology: Amazon Sagemaker (ML)
- Kubernetes

23

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

24

OBJECTIVES – 11/28

- Questions from 11/21
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 8: AWS Step Functions, AWS SQS
- **Team 3: Research paper: Research Paper: The Gap between Serverless Research and Real-world Systems**
- Team 1: Cloud Technology: Amazon Sagemaker (ML)
- Kubernetes

25

OBJECTIVES – 11/28

- Questions from 11/21
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 8: AWS Step Functions, AWS SQS
- Team 3: Research paper: Research Paper: The Gap between Serverless Research and Real-world Systems
- **Team 1: Cloud Technology: Amazon Sagemaker (ML)**
- Kubernetes

26

WE WILL RETURN AT ~4:50 PM

27

OBJECTIVES – 11/28

- Questions from 11/21
- Tutorials Questions
- Class Presentations Schedule - Cloud Technology or Research Paper Review
- Tutorial 8: AWS Step Functions, AWS SQS
- Team 3: Research paper: Research Paper: The Gap between Serverless Research and Real-world Systems
- Team 1: Cloud Technology: Amazon Sagemaker (ML)
- Kubernetes

28

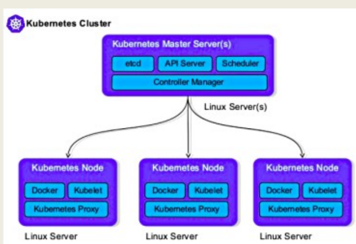
KUBERNETES

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

29

MASTER SERVICES

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



30

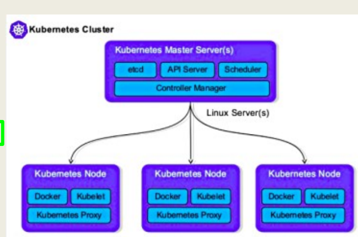
CLOUD CONTROLLER MANAGER

- 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.

31

MASTER SERVICES

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



32

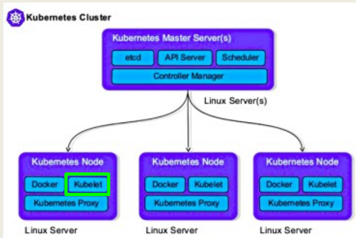
WORKER NODES

- 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

33

WORKER NODES

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



34

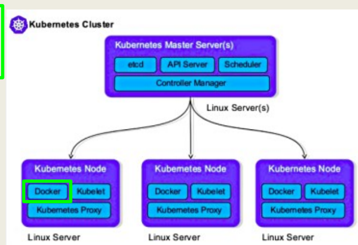
KUBELET

- 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

35

WORKER NODES

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



36

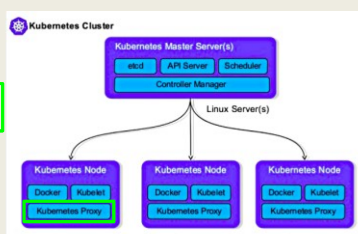
CONTAINER RUNTIME(S)

- 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

37

WORKER NODES

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



38

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

39

CORE KUBERNETES COMPONENTS

- Kubernetes DNS
- Pods
- Services

40

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)

41

CORE KUBERNETES COMPONENTS

- Kubernetes DNS
- Pods
- Services

42

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

43

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

44

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

45

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

46

CORE KUBERNETES COMPONENTS

- Kubernetes DNS
- Pods
- Services

47

KUBERNETES "SERVICES"

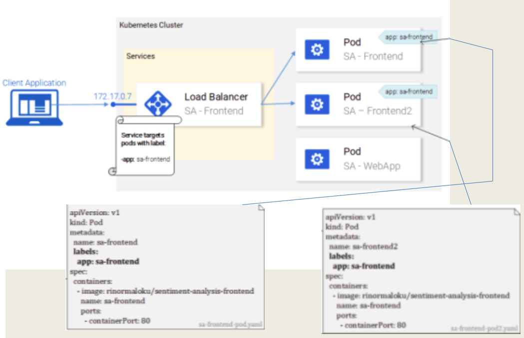
- Pods managed with "Deployments" or "DameonSets" controllers are automatically replaced when they die
  - This provides resiliency for the application
- **KEY IDEA**: Pods are unreliable
- **Services** provide reliability by acting as a "GATEWAY" to pods that implement the services
  - They underlying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

48

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

49



50

QUESTIONS

51