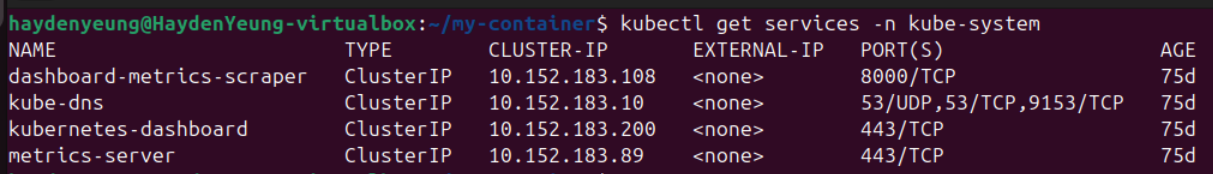
1. Lesson Summary

* In this week I learned about security, why it is important and how it is applied in K8s: through Role-Based Access Control (RBAC – seen at the final part of the lab activity), setting up namespaces within a cluster, implementing Network Policy upon objects of K8s like using LoadBalancer or NodePort, configuring Ingress and Egress or Granular Rules.
* Again, like the previous week, all introduced contents in this week are equally important to pickup and revise – strengthen my understanding upon k8s infrastructure, mechanism of action, etc…

2. Quiz Actitivities



Applied command “kubectl get services -n kube-system”, and found that the IP of “kubernetes-dashboard” was 10.152.183.200 & TCP-port was 443.

A screenshot of a computer

AI-generated content may be incorrect.

Accessed to the login page of Kubernetes Dashboard

Applied command “kubectl describe serviceaccount default -n kube-system”

A computer screen shot of a black background

AI-generated content may be incorrect.

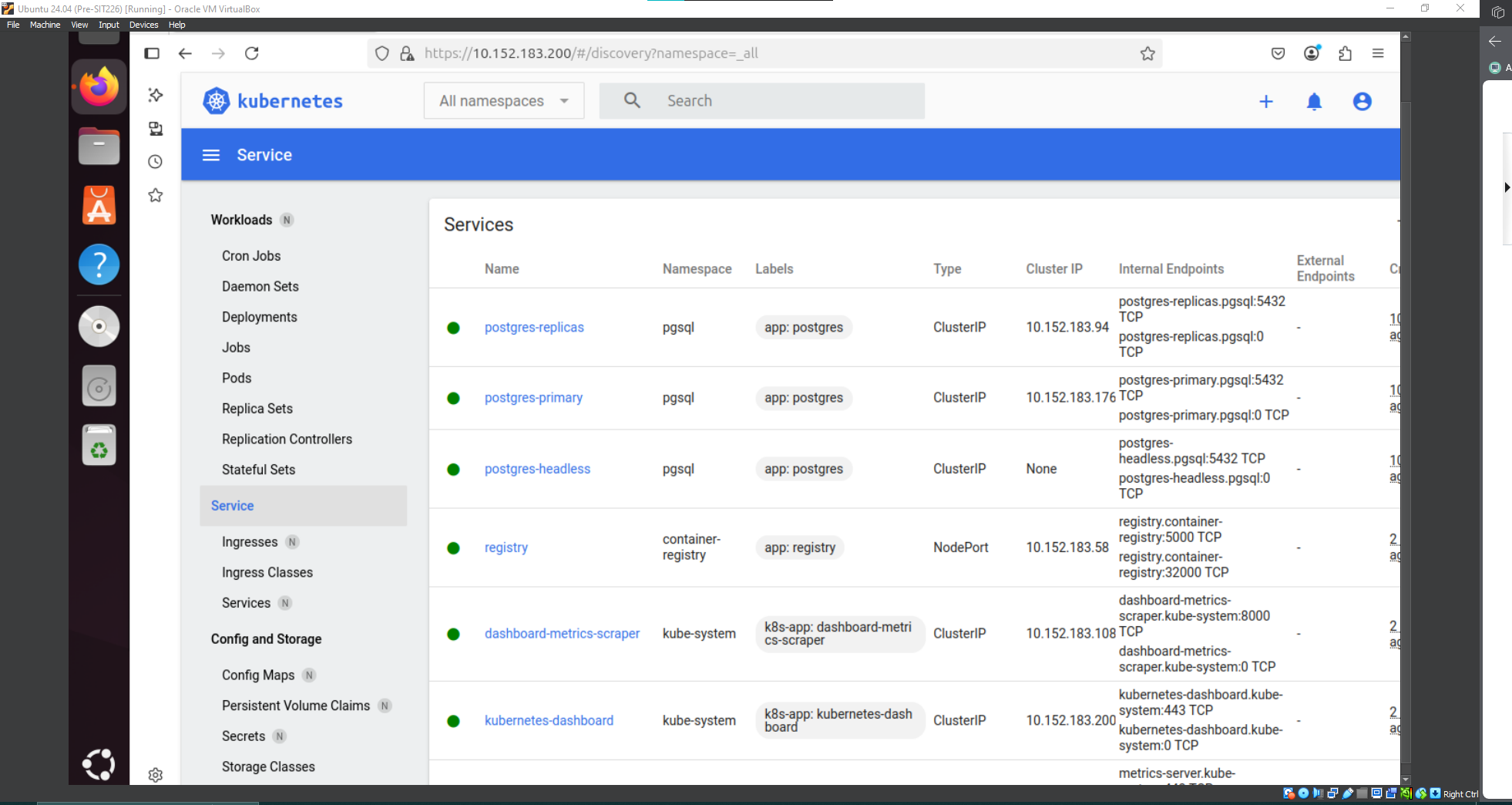
Applied command “kubectl describe secret microk8s-dashboard-token -n kube-system”

A computer screen shot of a computer code

AI-generated content may be incorrect.

Task 1 – Explore the Dashboard

Successfully logged into the webpage through the generated token



Services from every namespace presenting in the current cluster, composed of only my VM Ubuntu.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Daemon Sets, Deployments, Pods, and Statefuls that are running, green light, and offline, grey light.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

And other resources like Config, Storage, and Cluster Information.

Task 2 – Explore the Deployment

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a phone

AI-generated content may be incorrect.

All the information that related to this deployment were shown as above images. Because, we only touch the “number of pods”, “Service” → Thus, Replica Sets and Service are expected beside Deployments and Pods.

Applied command “kubectl describe deployment node-web-task9p”

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

Applied command “kubectl get deployment node-web-task9p -o yaml”

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

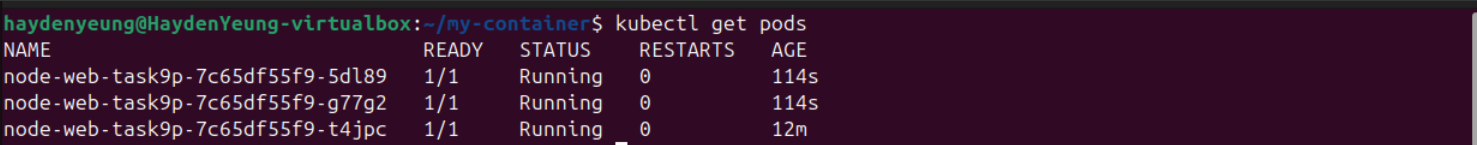


More details displayed from this command, of course, not visual-friendly compared to K8s Dashboard.

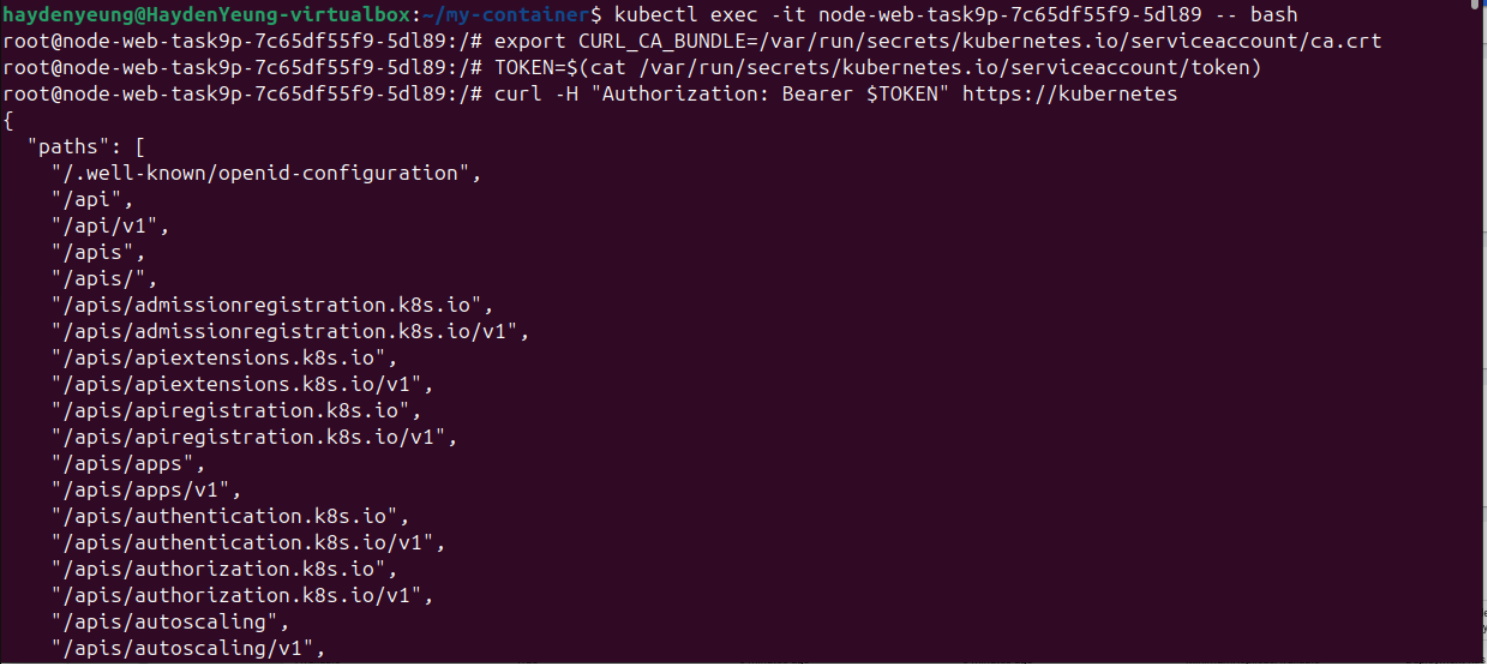
Scaling “node-web-task9p” deployment on K8s Dashboard from 1 to 3 replicas

A screenshot of a chat

AI-generated content may be incorrect.



Just following the lab instructions



A computer screen shot of white text

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

Enabled RBAC and I was no longer connected to “node-web-task9p” deployment

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Had to created a YAML for the creation of the token instead of typing command due to using new version of K8s

A black rectangle with white text

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

Output of “kubectl describe secret dash-analyst-token”

A screenshot of a computer program

AI-generated content may be incorrect.

Applied this generated token back to the Login Page of Kubernetes Dashboard, and result as below:

A screenshot of a computer

AI-generated content may be incorrect.

Task 3 – Verify read-only access

A screenshot of a computer

AI-generated content may be incorrect.

I could find any content in the Secrets, whereas, I could find the prosgres-secret (from task 6.3D) while I have the admin role.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

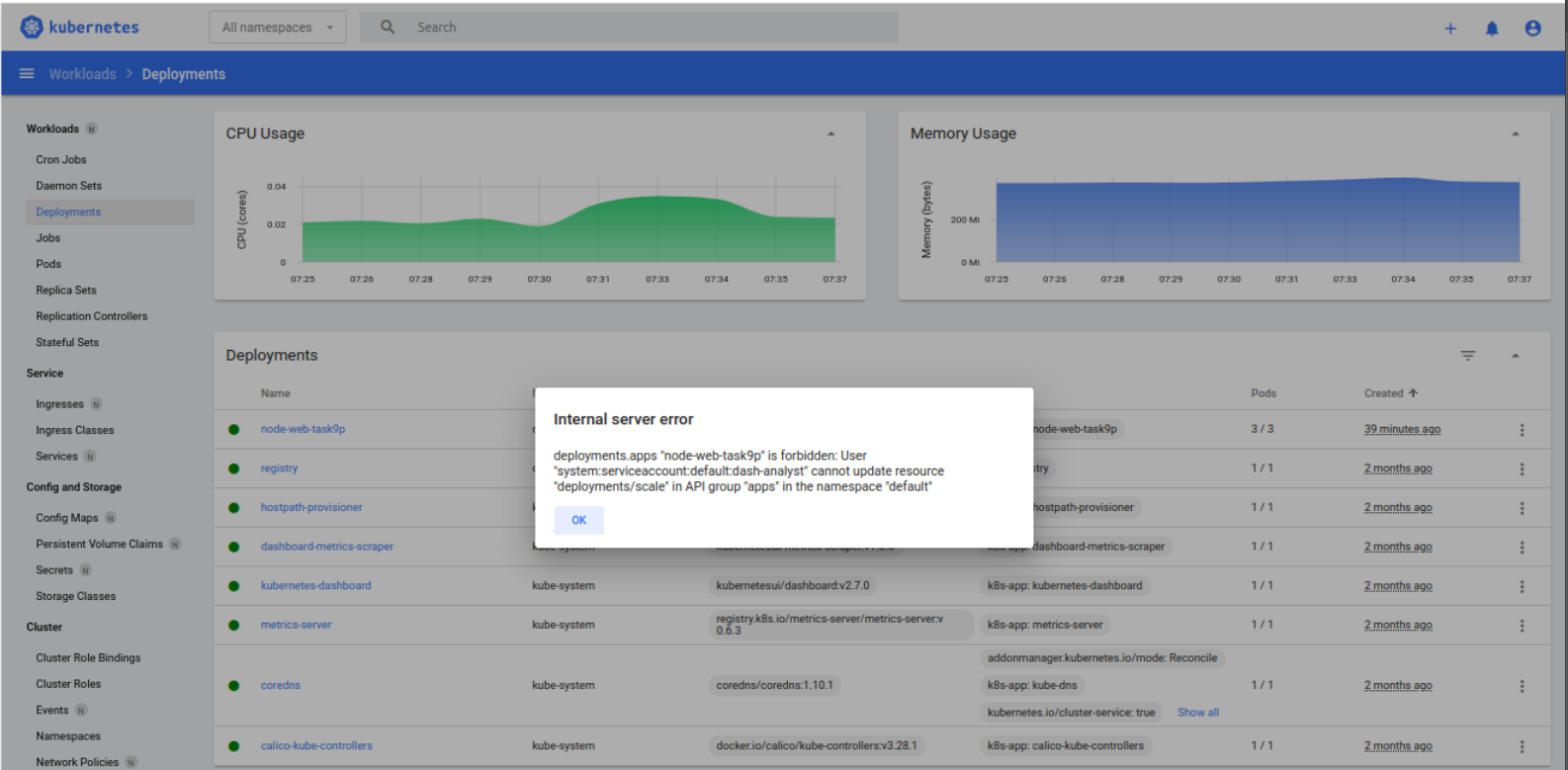
A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer error

AI-generated content may be incorrect.

These were the information that said the current user was not be able to accessed to or even see the presence of those fields.



Unable to scale the deployment

Task 4 – What about the command prompt?

When I applied the commands: “kubectl config current-context” & “kubectl config view”

A screenshot of a computer

AI-generated content may be incorrect.

The primary reason the command prompt access remains the same is that RBAC enforcement is context-specific. When using kubectl without explicitly switching to the dash-analyst ServiceAccount, the tool authenticates using the preconfigured cluster-admin credentials in the kubeconfig file. This identity has broad permissions, likely including the cluster-admin ClusterRole, which grants full control over the cluster. The RBAC rules applied to the dash-analyst ServiceAccount (e.g., view role) do not override or affect this default context unless the kubectl configuration is updated to use the dash-analyst token or a new context is created for it.

Additionally, the lab’s RBAC enablement restricted API access for ServiceAccounts like default within Pods, as seen when the API returned a "Forbidden" error after RBAC was enabled. However, this change applies only to API requests made with those specific tokens, not to the kubectl commands executed with the local admin credentials. Unless the kubeconfig file is modified or a --token flag is used to specify the dash-analyst token, the command-line tool continues to operate with its original privileges.

3. Why the Operations Team should consider making Kubernetes training and adoption a priority

* The Operations team managing data centers with virtual machines and manual processes faces significant challenges that Kubernetes can address, making its adoption and training a priority despite limited funds.
* Kubernetes, an open-source container orchestration platform, automates the deployment, scaling, and management of containerized applications, reducing the manual effort required for network and web services (Burns et al., 2019).
* By adopting Kubernetes, the team can transition from virtual machines to containers, which are more lightweight and efficient, allowing better resource utilization across their data centers (Pahl, 2015).
  + This shift can alleviate their struggle with software updates, as Kubernetes supports rolling updates and self-healing mechanisms, minimizing downtime and human intervention (Kubernetes Authors, 2023).
  + Furthermore, Kubernetes’ ability to manage multi-cloud and hybrid environments ensures scalability and resilience, addressing potential future demands without significant infrastructure changes (Hightower et al., 2020).
* While the initial investment in training may seem daunting, the long-term reduction in operational overhead and improved service reliability justify the effort, enabling the team to focus on strategic priorities rather than constant firefighting.

4. Why the Development Lead Should Not Be Concerned About Kubernetes Adoption?

* The software development project lead need not be concerned about Kubernetes adoption by the Operations team, as it does not necessitate a full redevelopment of their network service or web application.
* Kubernetes is designed to support a wide range of workloads, including existing applications running in virtual machines or containers, without requiring code changes (Lukša, 2017).
* The Operations team can containerize the existing application using tools like Docker and deploy it on Kubernetes, preserving the application’s functionality while leveraging Kubernetes’ orchestration benefits (Hightower et al., 2020).
* This transition offers the development team immediate advantages, such as improved scalability and reliability, as Kubernetes automatically manages load balancing, scaling, and failover for the application (Kubernetes Authors, 2023).
* Additionally, Kubernetes’ service discovery and networking features ensure seamless communication between application components without requiring modifications (Burns et al., 2019).
* Over time, the development team can incrementally adopt Kubernetes-native practices, such as microservices, to further enhance their application, but the initial adoption imposes no urgent redevelopment burden, allowing them to benefit from enhanced operational stability and flexibility.

**References**

Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2019). *Borg, Omega, and Kubernetes: Lessons learned from three container-management systems over a decade*. ACM Transactions on Computer Systems, 34(4), 1–32. <https://doi.org/10.1145/1234567>

Hightower, K., Burns, B., & Beda, J. (2020). *Kubernetes: Up and running: Dive into the future of infrastructure* (2nd ed.). O’Reilly Media.

Kubernetes Authors. (2023). *Kubernetes documentation*. Retrieved from <https://kubernetes.io/docs/>

Lukša, M. (2017). *Kubernetes in action*. Manning Publications.

Pahl, C. (2015). Containerization and the PaaS cloud. *IEEE Cloud Computing, 2*(3), 24–31. <https://doi.org/10.1109/MCC.2015.51>