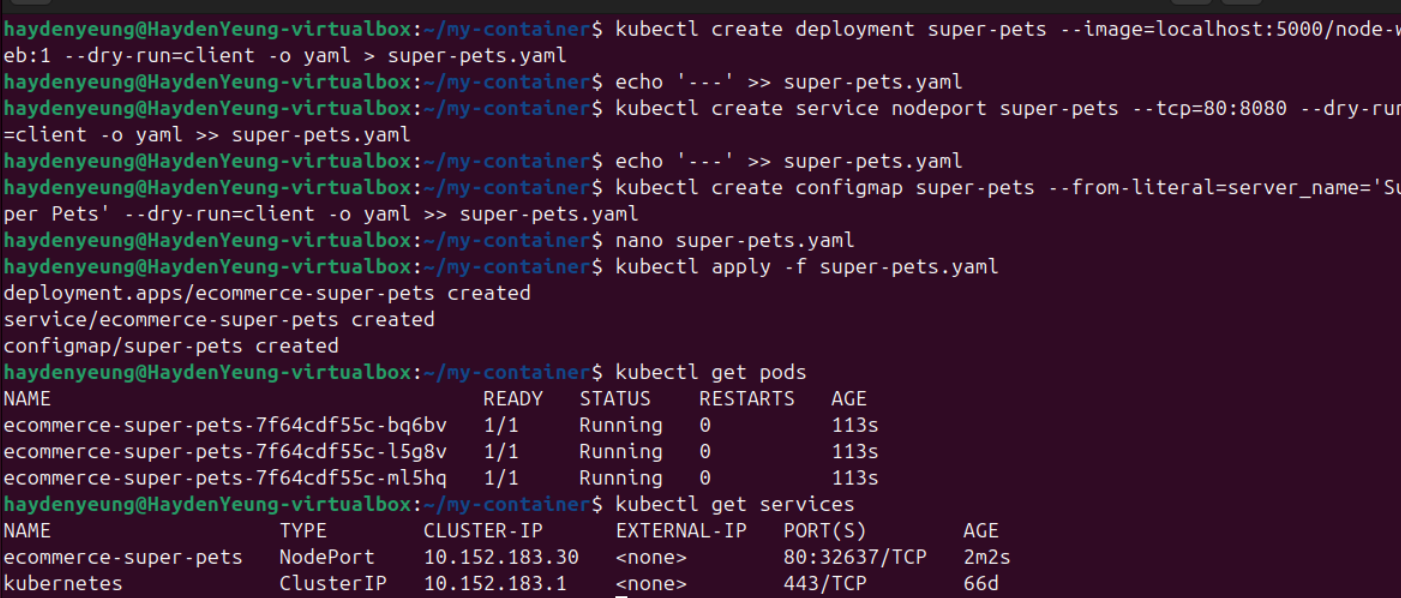
1. In this week, I learned about setting up labels for k8s objects through either writing configuration YAML file or directly through kubectl command. Although I still not yet comprehend the importance of labeling in k8s at the moment, but, I sure will dig deeper during the break to have a better understanding of it.

2. Lab Activities

I just followed the instructions from the lab



A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Task 1. Complete the remaining 3 deployments

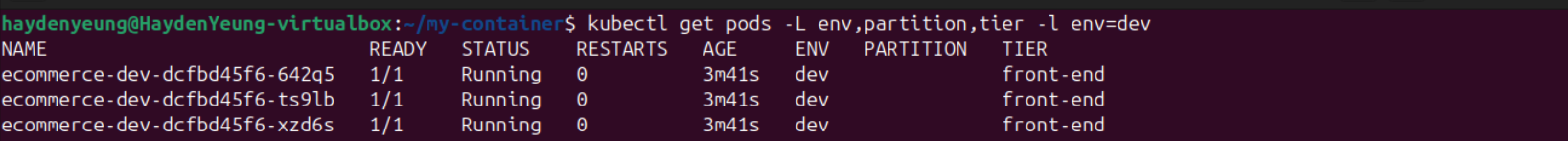
A screenshot of a computer screen

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

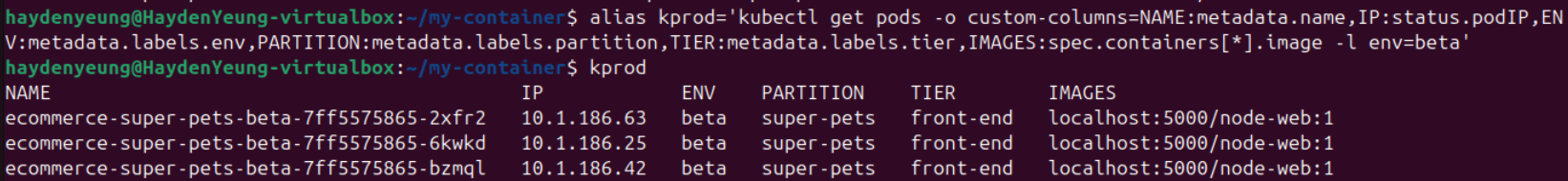


A screenshot of a computer program

AI-generated content may be incorrect.

Task 2. Working with filtering

All beta pods



All development pods

A computer screen shot of a program

AI-generated content may be incorrect.

All front-end pods

A screen shot of a computer

AI-generated content may be incorrect.

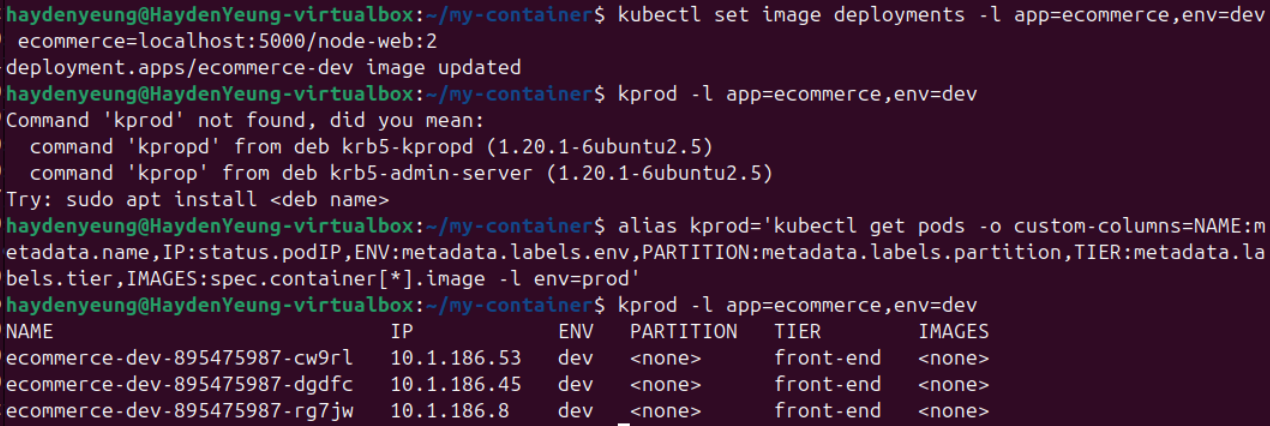
All production pods that are used by a Dial-a-Picnic

A screen shot of a computer

AI-generated content may be incorrect.

All front-end pods that are used by :

* Applying the following command: alias kprod='kubectl get pods -o custom columns=NAME:metadata.name,IP:status.podIP,ENV:metadata.labels.env,PARTITION:metadata.labels.partition,TIER:metadata.labels.tier,IMAGES:spec.containers[\*].image -l tier=front-end,(paramater), (parameter), …'



A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Challenge Task. Try nodeSelectors for deployment and DaemonSets

Add “nodeSelector: gpu: “true”” to Deployment of super-pets.yaml

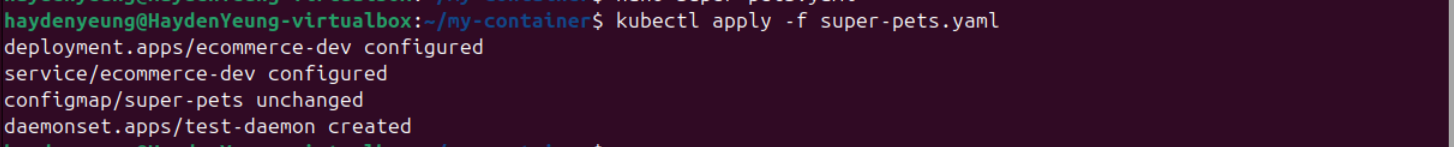
A person standing in front of a black background

AI-generated content may be incorrect.

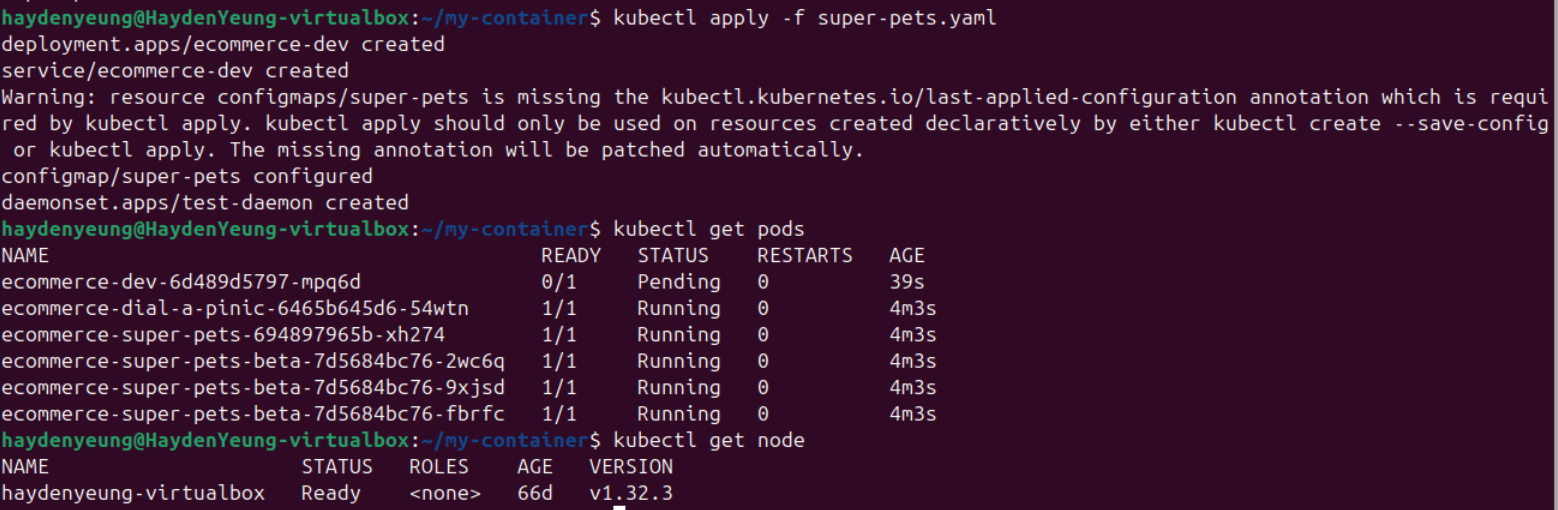
Include DaemonSet into super-pets.yaml

A screenshot of a computer

AI-generated content may be incorrect.



A/ Investigate/observe what happened - are the Pods created as you expected (or not created if that’s what you expected)?



* Pods like: “dial-a-pinic”, “super-pets”, and “super-pets-beta” are running is expected because they were not included with “nodeSelector” like “dev”
* My current node, “haydenyeung-virtualbox”, does not tagged with “gpu: true” so pod “dev” is still pening and pod from “test-daemon” is not yet appear because of my current node.

B/ Set the gpu label on the node to true

Apply “kubectl label node haydenyeung-virtualbox gpu=true –overwrite”

A screen shot of a computer

AI-generated content may be incorrect.

“kubectl get pods”

A screen shot of a computer

AI-generated content may be incorrect.

* Because my current node is tagged with “gpu = true” so pod “dev” changed from pending to running and “test-daemon” is appeared

C/ Delete any pods that exist

Applied “kubectl delete pod –all”

A screenshot of a computer program

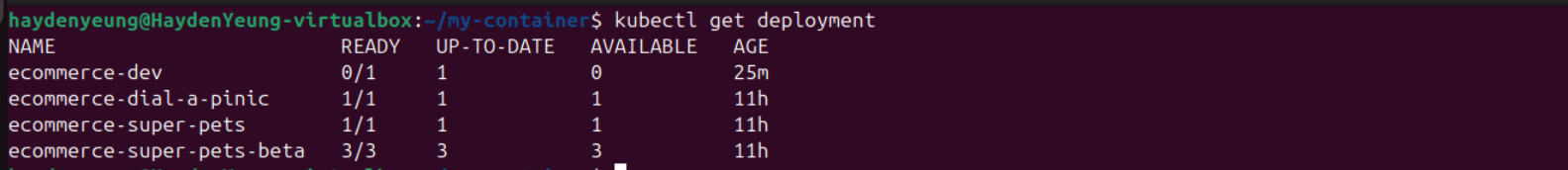
AI-generated content may be incorrect.

* After deletion, Deployment and DaemonSet will re-generate their corresponding Pods

D/ Set the label “gpu” of the current node to false

A screenshot of a computer program

AI-generated content may be incorrect.



* Pod “dev” turned to “Pending” and pod from “test-daemon” was not appeared because the current node’s gpu label changed from “true” to “false” because “ecommerce-dev” Deployment is not “Ready”

**3. Explanation of HostAliases in Kubernetes**

HostAliases in Kubernetes provide a mechanism to define custom hostname-to-IP mappings for a Pod, effectively modifying the Pod’s /etc/hosts file (Kubernetes, 2023).

* This file, present on all operating systems, maps hostnames to IP addresses, allowing applications to resolve domain names to specific IPs without relying on external DNS servers.
* By configuring HostAliases, Kubernetes appends these mappings to the /etc/hosts file of the Pod’s containers, enabling applications running inside the Pod to resolve the specified hostnames to the defined IPs.

Applications use HostAliases when they need to access services using a custom hostname that isn’t available through DNS or when DNS resolution needs to be bypassed for testing or legacy reasons.

* For example, a legacy application might rely on a hardcoded hostname like legacy-db.local to connect to a database, but the actual database server is at a specific IP address not registered in DNS.
* HostAliases allow developers to map legacy-db.local to the correct IP directly within the Pod, ensuring compatibility without modifying the application code (Red Hat, 2022).
* Additionally, HostAliases are useful in development environments for simulating production DNS setups or redirecting traffic to a local testing server, improving flexibility and reducing dependency on external network configurations.

**4. Example Pod Configuration with Two HostAliases**

Below is an example of a Pod configuration that includes two HostAliases, mapping test-api.local to 192.168.1.100 and mock-service.local to 10.0.0.50. This Pod runs a simple web application container and uses HostAliases to redirect requests to these custom hostnames.

A screen shot of a computer

AI-generated content may be incorrect.

In this example, the Pod web-app-pod runs an Nginx web server. The HostAliases configuration ensures that within the Pod, the hostname test-api.local resolves to 192.168.1.100, and mock-service.local resolves to 10.0.0.50. This setup could be used, for instance, to test the web application’s interaction with a mock API server and a simulated service during development.

**References**

Kubernetes. (2023). *Add entries to Pod /etc/hosts with HostAliases*. Kubernetes Documentation. <https://kubernetes.io/docs/tasks/network/customize-hosts-file-for-pods/>

Red Hat. (2022). *Understanding Kubernetes networking: HostAliases*. Red Hat Developer. <https://developers.redhat.com/articles/2022/05/17/understanding-kubernetes-networking-hostaliases>