




Kubernetes Metacog

Owner	 Ryan Fontaine
Tags	Kubernetes MetaCog
Created time	@July 13, 2023 8:28 AM

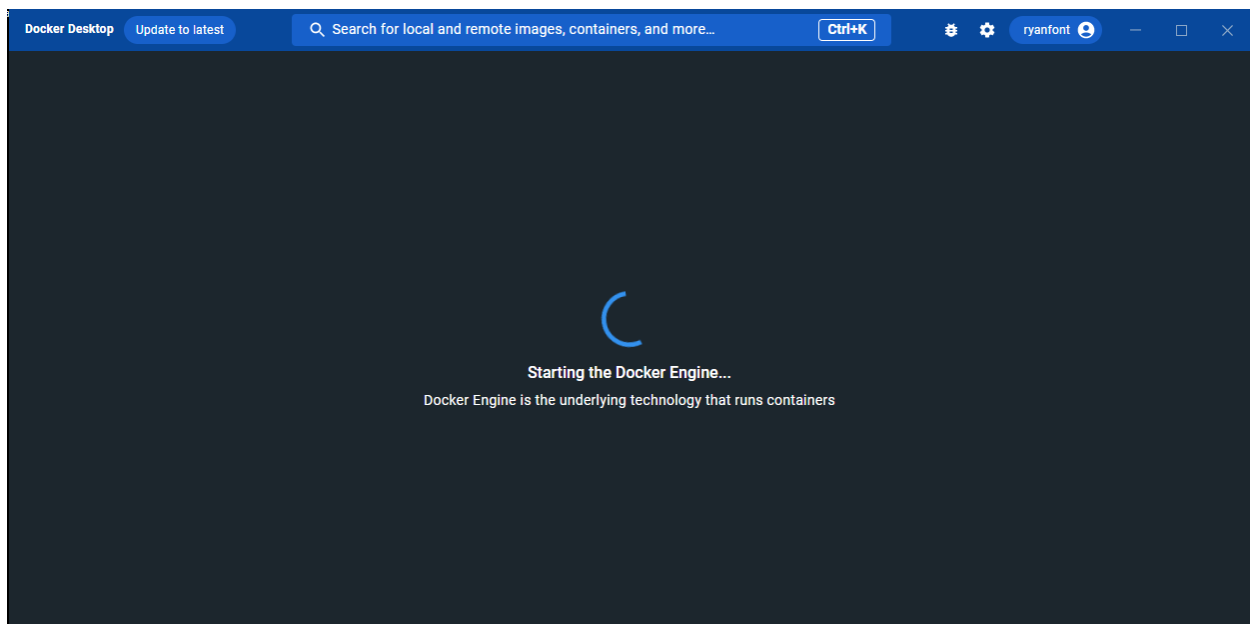
[System Setup](#)

[Code Overview](#)

[System Output](#)

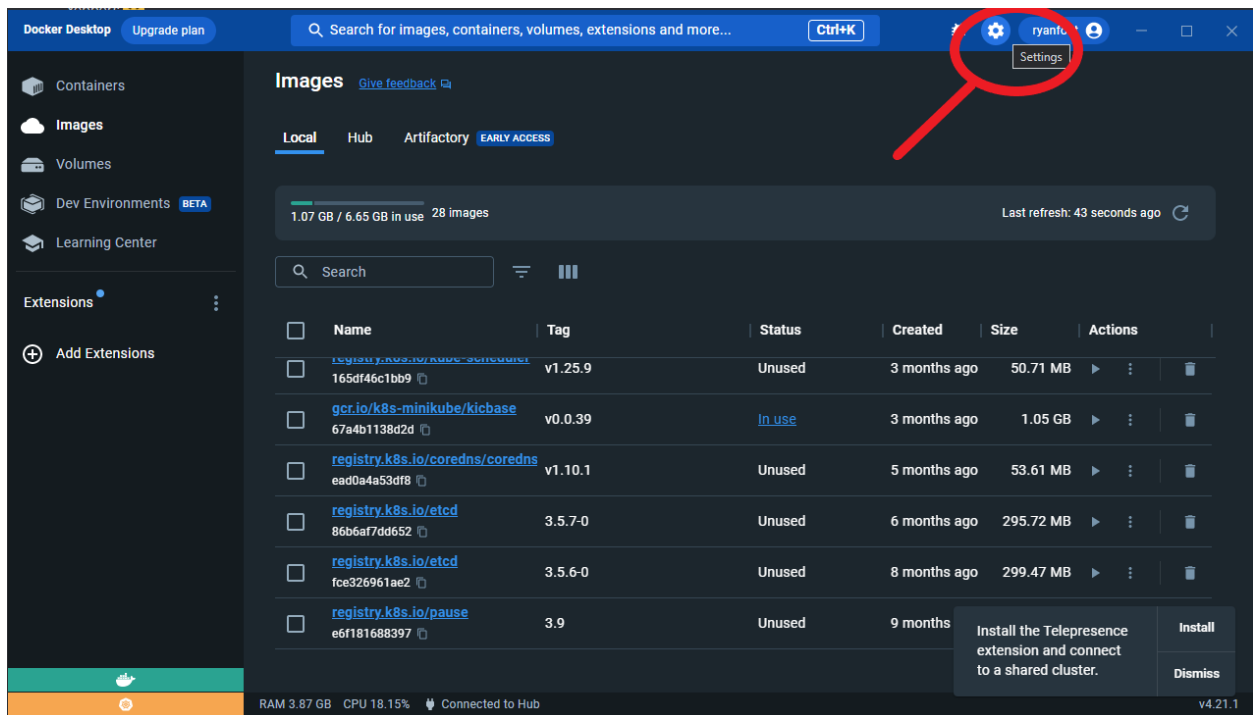
System Setup

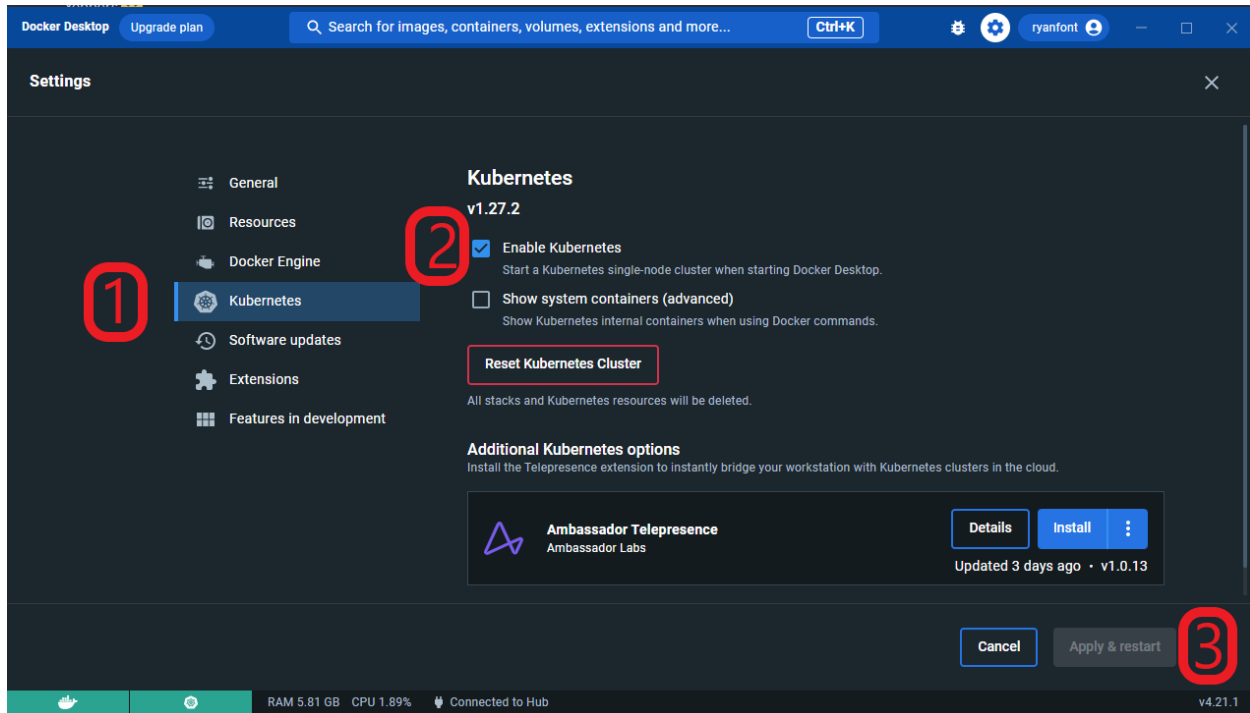
- First, install Docker Desktop if you have not already: <https://www.docker.com/>



- Open Docker Desktop to start the Docker Engine
- Download the Dynamic-CFAR repository from GitHub

- Then, build the Metacog Docker Images in the /Dynamic-CFAR/metacogtainer folder if you have not already, these commands can be found in the README_system_guide.txt
 - You may need to run the following command in the terminal: pip install docker
- Install minikube and kubectl: <https://minikube.sigs.k8s.io/docs/start/>





- Navigate to the Settings menu in Docker Desktop
- Then activate Kubernetes by:
 - Clicking into the Kubernetes Tab on the left side
 - Clicking the option to 'Enable Kubernetes'
 - Clicking Apply & Restart in the bottom right
- Once you have kubectl and the code installed, navigate to the following folder:
 - Dynamic-CFAR\metacogtainer\kubernetes-implementation

```
(base) PS C:\Users\Ryan\Documents\GitHub\Dynamic-CFAR\metacogtainer\kubernetes-implementation\debug> kubectl create -f .\metacog-orchestra.yaml
```

- To activate Metacog in Kubernetes run the following command:
 - kubectl create -f .\metacog-orchestra.yaml
 - This will create a Kubernetes “Job” that runs the Modular Metacog Docker Images

- Currently it creates one instance of this in a “Pod”, which this number can be scaled to create multiple of these and load balance the processing

```
(base) PS C:\Users\Ryan\Documents\GitHub\Dynamic-CFAR\metacogtainer\kubernetes-implementation\debug>
kubectl get jobs
NAME      COMPLETIONS   DURATION   AGE
metacog   0/1            102s       102s
(base) PS C:\Users\Ryan\Documents\GitHub\Dynamic-CFAR\metacogtainer\kubernetes-implementation\debug>
kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
metacog-dgkrz  0/1     Init:5/10  0           115s
```

- To view the active Kubernetes Job run the following command:
 - kubectl get jobs
- To view the active Kubernetes Pods run the following command:
 - kubectl get pods

```
(base) PS C:\Users\Ryan\Documents\GitHub\Dynamic-CFAR\metacogtainer\kubernetes-implementation\debug>
kubectl get jobs
NAME      COMPLETIONS   DURATION   AGE
metacog   1/1            3m55s      4m50s
```

- Once the Job / Pod is complete it will display as 1/1 Completions
 - Duration is the runtime; here it took about four minutes to complete
 - Age is how long since it was created

```
(base) PS C:\Users\Ryan\Documents\GitHub\Dynamic-CFAR\metacogtainer\kubernetes-implementation\debug>
kubectl logs metacog-dgkrz metacog-results
CD: 0.0 / GLRT: 0.0 / Ideal: 2.0
```

- To view the output, view the logs of the pod
 - The pod name will have a unique suffix attached, that can be found using 'kubectl get jobs'
 - metacog-results is the Docker Container name that generates the results
 - So, viewing the logs of the results container will give us the output

```
(base) PS C:\Users\Ryan\Documents\GitHub\Dynamic-CFAR\metacogtainer\kubernetes-implementation\debug>
kubectl delete jobs metacog
job.batch "metacog" deleted
```

- To delete the job run the following command:
 - `kubectl delete jobs metacog`
 - This will also subsequently delete all pods created by the Job as well

Code Overview

The code for the Kubernetes Job can be found in the following file: `metacog-orchestra.yaml`

Overall, the Job creates a template that runs the Docker Images in an instanced pod.

```
restartPolicy: OnFailure # Restarts a container when it fails
initContainers: # These initContainers are run prior to the main container, this can be used to run the containers sequentially
- name: base
  image: metacog-base
  imagePullPolicy: Never # This means it will not look for the images on Docker Hub (The online one), and instead use local Images
volumeMounts:
- name: metacog-volume # Uses the metacog-volume file storage
  mountPath: /app/docker_bind # The folder in the volume where files are saved and loaded from
```

- The 'initContainers' are run before the main 'containers', there are 10 in total
 - This allows the code to be run in the correct order
 - Base → Discriminator → Evaluator → Models → Results
 - `restartPolicy : OnFailure`
 - This specifies that Images should be restarted if they fail
 - So, containers are only run once and are resilient to random error
 - `imagePullPolicy : Never`
 - This specifies that Images should be found locally and not on the online Docker Hub
 - `volumeMounts`

- This specifies the file system to share output between containers

```
containers: # These are run after the init containers
- name: metacog-results
  image: metacog-results
  imagePullPolicy: Never
  volumeMounts:
  - name: metacog-volume
    mountPath: /app/docker_bind
```

- The main 'containers' are run after the 'initContainers' have finished
 - Here, the metacog-results container needs the output of all the previous containers
 - So, it is only non-init container and runs once the rest of the system has finished

```
volumes: # Specifies a volume that can be mounted to containers to share files
- name: metacog-volume
  emptyDir: {}
```

- The last section in the code simply specifies the volumes used by the containers
 - Here, the containers are mounted to the same metacog-volume to share data

System Output

- Currently the metacog-results container generates the output of the system
 - This is both printed to the terminal (with Kubernetes stored in the container's logs) and also to saved to the file : metacog-results.csv
 - This csv is stored in the volume, and can be accessed by other containers