Docker kubernetes tutorial pdf

Approximate reading time: 3 minutes docker desktop includes standalone Kubernetes server and client, as well as Docker CLI integration. The Kubernete server is running locally on your Docker instance, is non-configurable, and is a single node cluster. The Kubernetes server is running in a docker container on your local system and is for local testing only. When Kubernetes support is enabled, you can place your workload in parallel, on Kubernetes, Swarm, and as standalone containers. Enabling or disabling the kubernete server does not affect other workloads. See Docker Desktop for Windows > Getting Started to provide Kubernetes and start testing the placement of your workload on Kubernetes. Using docker commands in Kubernetes can accommodate a stack with a docker stack, a docker-compose.yml file, and a stack name. docker stack deploy - compose-file/path/to/docker-compose.yml mystack docker stack services mystack You can see the service deployed with kubectl get services command. Specify the default namespace for the default namespace. You can specify a namespace with a --namespace flag. Docker stack deploy - namespace my-app - compose-file/path/to/docker-compose.yml mystack Run kubectl get services-n my-app see only services hosted in my-app namespace. To override the default orchestrator when testing Kubernetes, you may want to place some workloads in bar mode. Use DOCKER_STACK_ORCHESTRATOR to override the default orchestrator for a specific terminal session or one Docker command. This variable can be unset (default, in which case Kubernetes is an orchestrator) or set to a swarm or kubernetes. This command ignores the orchestrator in one deployment by setting the variable before the command is run. set DOCKER_STACK_ORCHESTRATOR =swarm docker stack deploy --compose-file /path/to/docker-compose.yml mystack Alternatively , - orchestrator flag can be set on bars or kubernetes when deploying override default orchestra placement. Docker stack deploy - orchestrator bars - compose-file/path/to/docker-compose.yml mystack Note Deploying the same app in Kubernetes and Swarm mode can cause conflicts with ports and service names. Use the kubectl command in Windows Cuber integration to provide the Cubernetes CLI command C:\>Program Files\Docker\Docker\Resources\bin\kubectl.exe. This location may not be your shell in the ROAD variable, so you may need to elect a full command path or add a path. For more information about kubectl, see the official kubectl documentation. The command can be checked by listing the available nodes: kubectl get nodes NAME STATUS ROLES AGE VERSION Docker-desktop Ready master 3h v1.8.2 Example app Docker has created the following demo app that you can deploy in swarm mode or Kubernetes using docker stack deploy command. version: 3.8 services: web: figure: ports: - - words: image: dockersamples/k8s-wordsmith-api deploy: copies: 5 endpoint_mode: dnsrr resources: limits: memory: 50M reservation: memory: 50M db: image: dockersamples/k8s-wordsmith-db If you already have a Kubernetes YAML file, you can deploy it using the kubectl command. windows, edges, cubers, kubectl, orchestration Pod is the lowest unit of application kubernetes. Now, before we move on, we need to get one thing straight – and this is a pod not equal to the container docker world. Pod can consist of several containers. If you have come from a pure Docker background, it can be difficult to wrap your head around. If the pod can have more than one container, how does it work? There are some limitations that we need to be aware of. The pod is as follows: One IP address Share localhost Shared IPC space shared network port range Shared volumes Containers pod talk to each other through the local host, while pod-to-pod communication is done through the services. ReplicaSets Now pod section, we found that pods are mortal, and if they die, that's the end of them. What if you want to have three versions of the same pod running accessibility? Enter the replication controller. The primary responsibility for the replication controller is to prevent a failure, and it is located above the pod resource type and controls it. Let's look at the example. I want to place 4 of my pod x, this time I could create a replica set. The replica set has a certain number of pods to run, in this case 4. If one of the pods fails or dies, the replication controller will start a new pod for me and again, I will have 4 pod x running. So, this functionality looks at the question we mentioned earlier about the pods being mortal. Services If we want to have a connection to our pods, we will need to create a service. In Kubernetes, the service is for network extraction over a set of pods. This allows you to balance traffic for failures. The service allows Kubernetes to detect single DNS record pods. As we mentioned earlier, each pod has a separate IP address. The deployment deployment resource type is located above the replica set and can manipulate them. Why do we want to manipulate the replica kit? Duplicate sets are all or nothing. If you need to upgrade, you must replace the replica set. This action will cause downtime for your application. One of the main advantages of Kubernete is high availability. Placements give us functionality to upgrade without downtime. As you do in the replica set, specify how many pots you want to run. When you activate the update the deployment will do a rolling upgrade to the pods, all while making sure that the upgrade is successful on the pod before moving on to the next one. For more information about working with pods, services, and deployments, you can visit When to use Kubernetes deployments, pods, and services Remember: deployment control replica sets and replica set control pods; means that when you use the deployment resource type, you cannot forget that you still need a service to access it. When you start business with Kubernetes and deploy the cluster locally, installing the kubectl Kubernetes command-line tool, kubectl, is used to manage the cluster and applications running inside it. Here's how to install it on Windows, Linux, and Mac: Windows Curl-LO (curl-s /bin/windows/amd64/kubectl.exe curl Linux -LO (curl-s /bin/linux/amd64/kubectl chmod +x./kubectl sud LO /bin/linux/amd64/kubectl chmod +x./kub etl sudo mv ./kubectl /usr/local/bin/kubectl Mac curl -LO curl -s o mv./kubectl/usr/local/bin / kubectl Mac Curl-LO /bin/darwin/amd64/kubectl chmod +x ./kubectl sudo mv ./kubectl /usr/local/bin/kubectl Verifying your setup kubectl version -output=yaml Installing Minikube Minikube supports several virtualization technologies. We will use VirtualBox because it is the only virtualization supported by all operating systems. Please note that virtualbox or HyperV to run when virtualization is enabled in the BIOS. Most laptops must be enabled by default. Windows Finally, you won't get a command if you're a Windows user. Instead, download the latest release from the minicubue-windows-amd64.exe file, rename it to minikube.exe and add it to your path. Linux Curl-Lo Minicubi & chmod + x minicupe & sudo mv minikube /usr/local/bin/mac brew butt install minicubi Creating a local set of minicubi Minikube makes creating a cluster as easy as it can get to. All you have to do is execute one command. The minicucucub will start the virtual machine on site and place the necessary Kubernete components in it. VM will get configured with Docker and Kubernetes using a single binary called localkube. minicude start - VM-driver = VirtualBox When we executed the Minikube start command, it created a new VM based on minikube image. This picture contains some binaries. It has both Docker and RKT container engines, as well as a localcubue library. rkt is an application container engine developed in today's cloud native environment. The Localkube library includes all the components required for the operation of Kubernete. Now the important thing is that the localcubis provide everything we need to run Kubernetes Locally. This tutorial shows you how to run a sample app for Kubernetes using minicupe and Katacoda. Katacoda provides a free, in-browser Kubernetes environment. Note: You can also follow this tutorial if you have installed the minicubi locally. See minicubi start installation instructions. TargetRele away sample application minicucub. Start the program. View View Journals. Before you startThis tutorial provides a container image that uses NGINX to echo back all requests. Create a minicubi clusterClick Launch for TerminalNote: If you installed the minicubi locally, run the minicuc start. Open the Kubernete dashboard in the Browser-Only:Katacoda environment: Click the plus sign at the top of the Terminal pane, and then click Select Port to view only host 1.Katacoda environment: type 30,000, and then click Display Port.Create a DeploymentA Kubernetes Pod is a group of one or more containers associated together for administration and networking. Pod in this tutorial there is only one Container. Kubernetes placement checks the health of your Pod and restarts the Pod container if it expires. Deployment is the recommended way to manage the creation and scaling of Pods.Use kubectl to create a command to create a deployment that manages the pod. Pod running container based on provided Docker image.kubectl create deployment hello-node - image = k8s.gcr.io/echoserver: 1.4 View deployment: output is similar: NAME READY UP-TO-DATE AVAILABLE AGE hello-node 1/1 1 1 1 1m See Pod : Output is similar: NAME READY STATUS restarts AGE hello-node-5f76cf6ccf-br9b5 1/1 Running 0 1m View cluster events: View kubectl configuration: Create ServiceBy default, The pod is only available with its internal IP address in the Kubernetes cluster. To have the hello-node Container available outside the Kubernetes virtual network, you must expose the Pod as kubernetes Service.Expose Pod to the public internet using kubectl expose command: kubectl expose placement hello-node - type = LoadBalancer - port = 8080 --type=LoadBalancer flag indicates that you want to expose your service outside the cluster. The application code inside the image k8s.gcr.io/echoserver only listens to TCP port 8080. If you used a kubectl to expose a different port, the clients could not connect to that other port. View newly created service: Output is similar:NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE hello-node LoadBalancer 10.108.144.78 <pending>8080:30369/TCP 21s Kubernetes ClusterIP 10.96.0.1 <none>443/TCP 23m Cloud Service Providers That Support Load Balancers, the external IP address will be provided, to access the service. Minicubi, LoadBalancer type makes the Service available through the minicude service command. Run the following command:minicupe service hello to the Katacoda node only: Click the plus sign, and then click Select port to view host in 1.Katacoda environment only: Note the 5-digit port number that is displayed opposite the 8080 service output. This port number is generated randomly and may be different. Type your number in the port number text box, and then click Show Port. Enable Add-insMinikub tool includes</none> </pending>var iespējot, atspējot un atvērt vietējā Kubernetes vidē. List the currently supported addons:The output is similar to:addon-manager: enabled dashboard: enabled default-storageclass: enabled efk: disabled freshpod: disabled gvisor: disabled helm-tiller: disabled ingress: disabled ingress-dns: disabled logviewer: disabled metrics-server: disabled nvidia-driver-installer: disabled nvidia-gpu-device-plugin: disabled registry: disabled registry-creds: disabled storage-provisioner: enabled storage-provisioner-gluster: disabled Enable an addon, for example, metrics-server:minikube addons enable metrics-server The output is similar to:metrics-server was successfully enabled View the Pod and Service you just created:kubectl get pod,svc -n kube-system The output is similar to:NAME READY STATUS RESTARTS AGE pod/coredns-5644d7b6d9-mh9ll 1/1 Running 0 34m pod/coredns-5644d7b6d9-pqd2t 1/1 Running 0 34m pod/metrics-server-67fb648c5 1/1 Running 0 26s pod/etcd-minikube 1/1 Running 0 34m pod/influxdb-grafana-b29w8 2/2 Running 0 26s pod/kube-addon-manager-minikube 1/1 Running 0 34m pod/kube-apiserver-minikube 1/1 Running 0 34m pod/kube-controller-manager-minikube 1/1 Running 0 34m pod/kube-proxy-rnlps 1/1 Running 0 34m pod/kube-scheduler-minikube 1/1 Running 0 34m pod/storage-provisioner 1/1 Running 0 34m NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE service/metrics-server ClusterIP 10.96.241.45 <none>80/TCP 26s service/kube-dns ClusterIP 10.96.0.10 53/1 service/kube-dns ClusterIP <none>10.96.0.10 53/1UDP, 53/TCP 34m pakalpojumu/monitoringa grafana NodePort 10.99.24.54 <none>80:30002/TCP 26s service/monitoring-influxdb Cluster IP 10.111.169.94 <none>8083/TCP, 8086/TCP 26s Atspējot metrikas-server: minikube addons atspējot metrikas serveri Izvade ir līdzīga :metrikas serveris tika veiksmīgi atspējots Clean upNow varat tīrīt resursus, kurus izveidojāt jūsu klasterī: kubectl dzēst pakalpojumu hello-node kubectl dzēst izvietošanu hello-node Pēc izvēles, apturēt Minikube virtuālo mašīnu (VM): Pēc izvēles, dzēst Minikube VM: Kas nextLast modificēts oktobris 22, 2020 pie 3:27 PM PST: Fix saites konsultācijas sadaļā (774594bf1) (774594bf1)</none> </none> </none> </none>