*Kubernetes*

**Kubernetes for the Absolute Beginner – Hands On**

(Tutorial – Channel Name: Udemy)

Videos - #1 - #

**First Published On: 16th Jun 2021**

**Last Updated On: 16th Jun 2021**

**Kubernetes**

|  |  |
| --- | --- |
| Setting up Kubernetes                                            **Pod**        **Demo - POD**    **YAML**              **Pods with YAML (Udemy Video #18)**    **Demo**        **Tricks and Tips – Installing YAML extension to VsCode**        **Replication Controller & Replica Set**    **Replica Controller**    **Replica Set**        **Demo – Replica Sets**                **Editing existing Replica Set**      **Another way to edit replica set using command**    **Deployment**        **Deployment Demo –**              **Networking in Kubernetes (Udemy Video #37)**      **Services**                **Create a Service Object - Demo**      **Cluster IP Service**      **LoadBalancer Service**      **Microservices Applications on k8s (Udemy Video #46)**    **Important: Watch Demo – Deploying Microservice Application on K8s (Udemy Video #47,48)**  **Important: Watch Demo – K8s on AWS (Udemy Video #53)**  **Kubeadm** | * Kubernetes is container management technology * Process of automatically deploying and managing containers is called as Container orchestration * Any node or container needs to be managed, e.g. if 1 node fails other node should be available * So k8s sets up multiple replicas of Nodes with k8s installed * A master node to monitor all these nodes * API Server – acts as the frontend for kubernetes, users, cli all talk to API server * etcd – key value store used to store all data,      * These are the components installed on Master and Worker Node * Container Runtime can be any runtime like Docker, rkt or CRI-O * K8s can be set up on local server using Minikube, MicroK8s or Kubadmn * As well the can be set up on cloud platforms as shown * Normally, the setup is as shown, but using Minikube this can be automated * Minikube packages all the components together and forms an iso package for download * This can be downloaded in installed on Hyper-V or Virtual box * Also you need to have kubectl and Minikube.exe installed * First install kubectl * We will install on Linux * You can also use windows to install minikube * Also, you can install directly on host system, however if anything gets messed up its difficult to rollback whereas in Virtualbox, we can directly delete the image and install again * Verify if virtualization is supported on Linux * Enter the command and the output should be as shown * Install minikube * We will install it on VirtualBox * Enter the command as shown for installing package * Add minikube to path * Enter following command for downloading the VM image * Check Virtual box , the image minikube should be in running state * Check the status by typing this command * If the output is as shown, you have the Cluster Set up successfully * Lets do some deployments * Get nodes to see the current node i.e. minikube in Ready state * Create a deployment * This way you have created a simple deployment “hello-minikube” * Now expose this deployment as a service * Get the URL of the service by this command * Paste the URL into the browser to observe the results * Delete everything * A container is encapsulated in a single kubernetes object known as Pod * A Pod is the smallest object that can be created in k8s * Pods usually have 1-1 relationship to containers running an application * However, if the application containers are of different kind, they can be installed in a single POD * They share same network and storage space * *kubectl run nginx –image nginx* : means pull image nginx * in this case, we also need to specify the image name specifically unlike docker. * Execute 1st command as shown to run an nginx container * 2nd command to get list of pods * 3rd to get details of the pod * The parameters in the .yml files are the core and important ones to be used * First create a yaml file * Use *kubectl apply* or *kubectl create* command to create a pod with container specified. * Replication controller is responsible for ensuring the HA of Pods, if one goes down it is transferred to another * Even if there is one Pod, replication control brings up new pod if existing fails. * The YAML file will be different as shown * In spec section, under template section, all the data will be same that we put in pod yaml files. * Execute commands as shown * Replica Controller is old version * Replica Set is new and recommended version. * Observe the difference highlighted. * The role of Replica Set is to monitor the Pods and if any of them fails, deploy a new one. * There may be 100s of PODs running, how does Replica Set know which of them to monitor? * In definition yml file, we define matchLabels section and same should be available in labels section. This ties the Pods with Replica Sets * Create a yml file as shown * Run the command to execute replica set and create pods according to number of replicas specified * Get the status of the replicaset * Get the status of the pods created * Now, lets delete 1 Pod to check if the replica set creates new one to maintain 3 Pods as per the yml file * Observe that a new Pod has been automatically generated 15s ago. * Also, if you try to create a new pod i.e. 4th pod with same name, observe that the 4th will automatically be terminated as we have only 3 replicas mentioned for the same name in replica set. * Let’s update existing Replica Set * This opens the replica set managed by k8s * Scroll down and change the replicas number as shown * Save the file and run *kubectl get pods* * Observe the new pod is now created automatically. * Until this point, the deployment file and replica files are similar, and works similar * Only difference is in file, the kind is Deployment * And command is different * Create a deployment using the yaml file * If you immediately run this command, you can see that the Pod is getting up one at a time * This is called rollout * Use --record to record the revision history * There is another way to edit the image name instead of using edit command * *nginx=nginx:1.18-perl* means replace nginx with nginx:1.18-perl * Here, observe that, in rollout history, we see the changes were made 3 times and history was recorded * To rollback, you can use this command highlighted to the state you want. * An IP Address is assigned to Pod * In this case, external client, is 192.168.1.0 and want to connect to 10.244.0.0, this is not possible as both are in different network * To solve this challenge, Service object is used to map incoming external request with the Pod ip address * This is called NodePort Service * NodePort Service, creates a service object which has its own ip and port * To map the Service to the Pod, use the selector section * Copy the labels from pods and paste it inside the selector to tie both together * If there are multiple Pods, Service considers the selector name and finds that there are multiple pods for that service * The service then acts as an internal load balancer to rout the requests to the pods randomly * The problem with NodePort is that there is a Service with port mapped for incoming traffic, so external client can use any ip of pod to access the application * But this is not suitable if we want to share only single ip to the Clients * Here you can use LoadBalancer Service |