*DevOps - Kubernetes*

**DevOps**

(Tutorial – Channel Name: Technical Guftgu)

**First Published On: 24th Jul 2021**

**Last Updated On: 9th Aug 2021**

**DevOps**

|  |
| --- |
| **Agenda** |

|  |  |
| --- | --- |
| Kubernetes    **Monolithic Architecture**    **Microservices**    **Containers**  **Kubernetes**        **Architecture**              **Commands and Implementation**    **Commands for installation**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*COMMANDS FOR INSTALLATION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **sudo su**  **apt-get update**  **apt-get install apt-transport-https** (required for securely communicate between master and nodes)  **apt install docker.io -y** (install docker)  **docker --version**  **systemctl start docker**  **systemctl enable docker**  **sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add** (key for intra communication with master and nodes)  **nano /etc/apt/sources.list.d/kubernetes.list**  **deb http://apt.kubernetes.io/ kubernetes-xenial main**  **apt-get update**  **apt-get install -y kubelet kubeadm kubectl kubernetes-cni**  **BOOTSTRAPPING THE MASTER NODE (IN MASTER)** – all commands in **blue** color  **kubeadm init**    **COPY THE COMMAND TO RUN IN NODES & SAVE IN NOTEPAD**  **mkdir -p $HOME/.kube**  **cp -i /etc/kubernetes/admin.conf $HOME/.kube/config**  **chown $(id -u):$(id -g) $HOME/.kube/config**  **kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml**  **kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/k8s-manifests/kube-flannel-rbac.yml**  **CONFIGURE WORKER NODES (IN NODES)** – all commands in **purple** color  **COPY LONG CODE PROVIDED MY MASTER IN NODE NOW LIKE CODE GIVEN BELOW**  **e.g- kubeadm join 172.31.6.165:6443 --token kl9fhu.co2n90v3rxtqllrs --discovery-token-ca-cert-hash sha256:b0f8003d23dbf445e0132a53d7aa1922bdef8d553d9eca06e65c928322b3e7c0**  **GO TO MASTER AND RUN THIS COMMAND**  **kubectl get nodes**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***        **YAML (manifest) File**    **Commands for Installing and Executing USING MINIKUBE:** (Video: <https://www.youtube.com/watch?v=hV8zi3vdQqk&list=PLBGx66SQNZ8aPsFDwb79JrS2KQBTIZo10&index=49>)  **Instance: UBUNTU AMI**  *sudo su*  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INSTALL DOCKER \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  *sudo apt update && apt -y install docker.io*  *OR*  *curl -fsSL https://get.docker.com -o get-docker.sh*  *sudo sh get-docker.sh*  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INSTALL KUBECTL \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  *curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl && chmod +x ./kubectl && sudo mv ./kubectl /usr/local/bin/kubectl*  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INSTALL MINIKUBE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  *curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 && chmod +x minikube && sudo mv minikube /usr/local/bin/*  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*START MINIKUBE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  *apt install conntrack*  *minikube start –vm-driver=none*  *minikube status*  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*YAML MANIFEST\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  kind: Pod  apiVersion: v1  metadata:  name: testpod  spec:  containers:  - name: c00  image: ubuntu  command: ["/bin/bash", "-c", "while true; do echo Hello-Bhupinder; sleep 5 ; done"]  restartPolicy: Never # Defaults to Always  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*** **MULTI CONTAINER POD ENVIRONMENT** **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  kind: Pod  apiVersion: v1  metadata:  name: testpod3  spec:  containers:  - name: c00  image: ubuntu  command: ["/bin/bash", "-c", "while true; do echo Technical-Guftgu; sleep 5 ; done"]  - name: c01  image: ubuntu  command: ["/bin/bash", "-c", "while true; do echo Hello-Bhupinder; sleep 5 ; done"]  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RUN MANIFEST FILE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  kubectl apply -f pod1.yml  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*COMMON COMMANDS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  kubectl get pods *(get the running pods details)*  kubectl delete pod <<podname>> *(delete pod specified)*  kubectl describe pod <<podname>> *(get details of pod specified)*  kubectl logs -f <<podname>> -c <<containername>> *(get details of specified container)* | Kubernetes   * This logo shows a steering wheel of a ship * There are 7 wheels on the logo, because when google launched, the first name was ***project 7*** * So the meaning, Kubernetes is used to drive (orchestrate) the ship (containers) * IT is also called K8s * In 1980s a new concept was invented called numeronym – Greek word * K *ubernete* S = K8s * I *nternationalizatio* n = i18n * **Monolithic Architecture** * Previously 2 decades back, the applications were designed and developed at once * E.g. Facebook, everything was developed at once, like login, news feed, posts, likes, story etc. * So, if anything needs to be changed, it would impact the complete code. * If any module goes down, complete application would go down.   **Microservices**   * After Monolithic architecture, came Microservices. * These are separate modules for each functionality or feature which has their own database * So if anything needs to be changed, only that module can be brought down and changed * **But, there is a problem**, what if all Servers have 64GB ram and later, some service require only 16GB whereas other service require more than 64GB, then the resources could not be shared. * Thus to resolve this, Containers came into existence.   **Containers**   * Containers has many advantages, * You can even run multiple containers each of separate services (login, news feed, posts, etc) on a single server. * The containers can be packaged and shared among other teams * Makes the processes faster as there is no OS and saves license costs * **But, there is a problem**, what if we want to generate 1000 containers? We will not type docker run command 1000 times   **Kubernetes**   * Using kubernetes, you can manage and create/delete chunk of containers automatically. * E.g. in Hotstar, when there is IPL final match, number of users increases exponentially, so, if by default a container handles 500 users load, and there are only 5 containers, there will be heavy traffic and apps may crash. To resolve this kubernetes is smart enough to determine the load and automatically duplicate containers as much there is load. Once the match is over, users are reduced, so kubernetes again will scale down the containers. * Responsible for AutoScaling, LoadBalancing, etc * Can run on any server – Virtual, Physical or Cloud * Can manage any Container Runtime services not only docker. * Master Node architecture * 1 Master to Many Nodes, Many to Many or 1 to 1 * A pod is the smallest unit of K8s * Hierarchy:   Cluster 🡪 Node 🡪 POD 🡪 Container 🡪 Apps/microservices**.**  **Architecture**  **Master (Control Plane)**   * **Controller manager**- This unit manages the desirable amount of containers to be created or destroyed. * **Kube Scheduler**- This unit actually performs appropriate actions passed by Controller manager. * **etcd Cluster**- This unit is the database responsible for keeping information on all the containers, pods etc. This is an outside component and not a part of K8s, but this component is required to run K8s. * **API Server**- all the communication happens through this API server * **POD**- Smallest unit of k8s, responsible for storing the containers (Docker, Rocket, etc) * **Kubelet**- This is responsible to control the pods demands and requirement 🡪 kubelet sends this requests to API server 🡪 API server updates this to etcd Cluster 🡪 API server then sends this demands to Controller Manager 🡪 Controller Manager sends this information to Kube Scheduler 🡪 Kube Scheduler performs the actual actions. * **Kube-proxy**- Responsible for networking, assigning IP to POD, communication between PODs * **State of Object –** means the current state. * **YAML –** Yet Another Markup Language. It is also called manifest file   **Below are two methods for creating objects**   * **Imperative Commands –** Executing commands directly from command line * **Declarative Object Configuration –** Declaring a manifest file with instructions and can be used by others to do the same operations * **Kubectl –** command line tool. Any command should start with kubectl * If any POD fails, same POD cannot be recreated, kube scheduler will create a new identical one with new ID (UID) * **All-in-One Single node installation –** used for practice and experiment. We use Minikube which generates all the components in one instance.   **YAML (manifest) File**   * Extension- .yml or. yaml * Eg. 1   StudentName:  Student 1:  Student 2:  Student 3:  Age: 12   * Eg. 2 (array) * Student1 details:   Name: Allen  Age: 12  Class: 5   * Student2 details:   Name: Bob  Age: 12  Class: 5 |
| **Play with K8s** *(YouTube - https://www.youtube.com/watch?v=sGNqFq9oMrE)* | **Play with K8s**   * To get access to a ready environment from Docker for working on Kubernetes, we have an easy method. * Go to <https://labs.play-with-k8s.com/> * Click +ADD NEW INSTANCE * Copy and Past the 1st Command as shown   *kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10.5.0.0/16*   * This will initialize the master node for the current instance * Run Second command on the same instance to initialize cluster networking   *kubectl apply -f https://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml*   * Now create another instance for Node * Now, Copy the command generated dynamically and paste it to the another instance (Node) * Now your Master Node setup is completed. * Create a container in a Pod * Run below command on Master   *kubectl run nginx --image=nginx*   * Run below command to know the status of Pods   *kubectl get pods* |
| **Labels and Selectors**        **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*YAML Files used in demo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **EXAMPLE OF LABELS**  kind: Pod  apiVersion: v1  metadata:  name: delhipod  labels:  env: development  class: pods  spec:  containers:  - name: c00  image: ubuntu  command: ["/bin/bash", "-c", "while true; do echo Hello-Bhupinder; sleep 5 ; done"]  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*NODE SELECTOR EXAMPLE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  kind: Pod  apiVersion: v1  metadata:  name: nodelabels  labels:  env: development  spec:  containers:  - name: c00  image: ubuntu  command: ["/bin/bash", "-c", "while true; do echo Hello-Bhupinder; sleep 5 ; done"]  nodeSelector:  hardware: t2-medium  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*EXAMPLE OF REPLICATION CONTROLLER\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  kind: ReplicationController  apiVersion: v1  metadata:  name: myreplica  spec:  replicas: 2  selector:  myname: bhupinder  template:  metadata:  name: testpod6  labels:  myname: bhupinder  spec:  containers:  - name: c00  image: ubuntu  command: ["/bin/bash", "-c", "while true; do echo Hello-Bhupinder; sleep 5 ; done"]  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*EXAMPLE OF REPLICA SET\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  kind: ReplicaSet  apiVersion: apps/v1  metadata:  name: myrs  spec:  replicas: 2  selector:  matchExpressions: # these must match the labels  - {key: myname, operator: In, values: [Bhupinder, Bupinder, Bhopendra]}  - {key: env, operator: NotIn, values: [production]}  template:  metadata:  name: testpod7  labels:  myname: Bhupinder  spec:  containers:  - name: c00  image: ubuntu  command: ["/bin/bash", "-c", "while true; do echo Technical-Guftgu; sleep 5 ; done"]  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **Labeling Pods using imperative method**    **Node Selector**    **Scaling and Replication**    **Replication Controller**      **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*EXAMPLE OF REPLICATION CONTROLLER\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  kind: ReplicationController  apiVersion: v1  metadata:  name: myreplica  spec:  replicas: 2  selector:  myname: bhupinder  template:  metadata:  name: testpod6  labels:  myname: bhupinder  spec:  containers:  - name: c00  image: ubuntu  command: ["/bin/bash", "-c", "while true; do echo Hello-Bhupinder; sleep 5 ; done"]          **Scaling the Pods**    **Replica Set**    **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*EXAMPLE OF REPLICA SET\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  kind: ReplicaSet  apiVersion: apps/v1  metadata:  name: myrs  spec:  replicas: 2  selector:  matchExpressions: # these must match the labels  - {key: myname, operator: In, values: [Bhupinder, Bupinder, Bhopendra]}  - {key: env, operator: NotIn, values: [production]}  template:  metadata:  name: testpod7  labels:  myname: Bhupinder  spec:  containers:  - name: c00  image: ubuntu  command: ["/bin/bash", "-c", "while true; do echo Technical-Guftgu; sleep 5 ; done"]    **Scale Replica Set** | * Labels are used to assign a tag to the Pod similar as we do in AWS * Using Selectors, you can get the selected Pod back   **Commands**   * **To Apply the manifest yaml file and run the container**   kubectl apply -f <<manifest\_file\_name>>  *kubectl apply -f pod5.yml*   * **To get list of pods with all labels**   *kubectl get pods -show-labels*   * **To label any pod adhoc**   kubectl label pods <<podname>> <<label=value>>  *kubectl label pods delhipod myname=xyz*   * **To label any Node adhoc**   kubectl label nodes <<podname>> <<label=value>>  *kubectl label nodes ip-172-31-23-34 myname=xyz*   * **To get list of pods matching to label**   kubectl get pods -l <<label=value>>  *kubectl get pods -l env=development*   * **To get list of pods excluding the specified label**   kubectl get pods -l <<label!=value>>  *kubectl get pods -l env!=development*   * **Delete pod using label**   kubectl delete pod -l <<label=value>>  *kubectl delete pod -l env=development*   * **To get list of pods with multiple matches**   kubectl get pods -l ‘env in (<<label1, label2>>)’  *kubectl get pods -l ‘env in (development, testing)’*   * **To get list of pods excluding matches**   kubectl get pods -l ‘env not in (<<label1, label2>>)’  *kubectl get pods -l ‘env not in (development, testing)’*  **Labeling Pods using imperative method**   * There are two ways of labeling  1. Declarative – define labels in manifest files itself 2. Imperative – define labels on runtime using commands   **Node Selector**   * Node Selector is used when you want to create and run the pod to the specified Node only   **Scaling and Replication**  **Replication Controller**   * Replica will always make sure that your desired matches to current state * If it doesn’t match, it will add or remove the instances accordingly * To make desired and current equal, you will have to run replication controller object * Replication controller object can be invoked simply adding Kind: * Your pods will be automatically in High Availability if you have created Pods using Replication Controller * Kind: ReplicationController will make the K8s understand that this is Replication Controller and will start automatically maintaining High Availability * Replicas: number of pods to be created. * Selector: this means that when any pod goes down, create a similar one that has myname:bhupinder. * Run the file and observe the results * Now, if you delete a pod, and run get pods command again, you will observe that a new pod is again created. * This is how Replication Controller controls the number of pods defined.   **Scaling the Pods**   * You can scale the desired number pods adhoc using command line.   kubectl scale –replicas=<<desired numberr>> rc –l <<label=name>>  *kubectl scale -replicas=8 rc -l myname=bhupinder*    **Replica Set**   * In this case, the changes are   **kind: ReplicaSet**  **apiVersion: apps/v1**  **matchExpressions**   * Replica Set is new version of Replica Controller that has capability to use selector as match expression set * Use the above file for yml file of Replica Set and run to observe the results.   **Scale Replica Set** |