* Demonized containers : run in background
* Interactive containers:
* Ctl +pq –for disconnecting to a running container without killing the docker.
* Dhclient ---for getting ip when interphase was connected in host mode.
* <https://github.com/amitvashist7/kubernetes-project/tree/master/setup> training stuff
* download <https://storage.googleapis.com/minikube/releases/latest/minikube-windows-amd64.exe> or minikube for Windows and install from command line….rename minikube.exe.
* .\minikube.exe start --- will download the image on windows.
* <https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-with-chocolatey-on-windows>

Above link has Windows curl to download kubectl for windows. (**curl -LO** [**https://storage.googleapis.com/kubernetes-release/release/v1.10.3/bin/windows/amd64/kubectl.exe**](https://storage.googleapis.com/kubernetes-release/release/v1.10.3/bin/windows/amd64/kubectl.exe))

* Vagrant init ---with vagrant file in current locatin.
* Vagrant status
* Vagrant ssh kuber-master
* Echo ‘export $KUBECONFIG=/etc/kubernetes/admin.conf’ >> ~/.bash.rc
* Kubectl create –f <file/dir>
* Kubectl get pods –o wide
* Kubectl describe pod <pod name>
* Kubectl delete pod <pod>
* Kubectl delete service <pod>
* Kubectl get rc
* Kubectl edit rc hellowordo-controller
* Kubectl scale –replicas=1 rc helloworld-controller
* Kubectl delete rc helloworld-controller
* Git repo got to deployment
* Kubectl create –f helloworld.yml
* Kubectl get deployments
* Kubectl get pods
* Kubectl get pods –show-labels
* Kubectl rollout status deployments/helloworld-deployments
* Kubectl expose deployment hellowrodl-deployment –type=NodePort
* Kubectl get services.
* Kubectl describe service helloworld-deployment
* Kubectl get deployment
* Kubectl set image deployment helloworld-deployment k8s-demo=amitvashist7/k8s-tiny-web:2
* Kubectl rollout status deployment helloworld-deployment
* Kubectl rollout undo deployment helloworld-deployment
* Kubectl set image deployment helloworld-deployment k8s-demo=amitvashist7/k8s-tiny-web:1

* Kubectl rollout history deployment helloworld-deployment deployments “helloworold-deployment”
* Kubectl rollout history deployment helloworld-deployment –revision=1
* Kubectl rollout undo deployment helloworld-deployment –to-revision=1

**Service :**

* Kubectl expose –nodeport/clusterip/load balance.
* Ingress traffic will come in if we use load balance.
* DNS name can be used using service discovery.
* Port range 30000-320767 –random port.
* Kubectl create –f helloworld.yml
* Kubectl create –f helloworld-nodeport-service.yml
* Kubectl describe services helloworld-service
* Kubectl expose pod helloworld
* Kubectl expose pod helloworld –type=NodePort

**Labels**

* Labe can be applied to pods/nodes (nodeSelector).
* Kubectl get nodes –show-labels
* Kubectl label node worker01 hardware=high
* Kubectl label node worker02 hardware=low
* Kubectl get nodes –show-labels
* kubectl create –f hellowrod-nodeselector.yml
* kubectl get deployments

**Health check**

* URL Health check
* Command status check
* Kubectl create –f helloworld-healthcheck.yml.
* Kubectl logs <node>/<pod>
* Kubectl get events

**Secrets**

* Username/passwd/keys/secret data etc.
* Environment variable.
* Stores in file
* .env file to store secrets
* Echo –n > “root” ./username.txt
* Echo –n>”passwd” ./password.txt
* Kubectl create secret generic db-user-pass –from-file=./username.txt –from-file=./password.txt
* Echo –n “root”|base64 ---create hashes
* Kubectl create –f deployments/helloworld-secrets.yml
* Kubectl get secrets
* Kubectl create –f deployment/helloworld-secret-volume.yml
* Kubectl exec –i –t <pod> /bin/sh
* kubectl describe pod/wordpress-deployment-6f47769b85-xn5f8 -n default
* kubectl creat -f wordpress-service.yml
* kubectl create -f wordpress-service.yml
* kubectl create -f wordpress-single-deployment-no-volumes.yml
* kubectl get pods
* kubectl exec -i -t wordpress-deployment-6f47769b85-xn5f8 /bin/sh
* kubectl get services
* kubectl get pods -o wide
* kubectl describe pods wordpress-deployment-6f47769b85-xn5f8
* kubectl exec -i -t wordpress-deployment-6f47769b85-xn5f8 sh -c wordpress --------------to login to Docker in a pod.
* Kubectl run –i --tty busybox --image=busybox –-restart=Never -- sh
* Kubectl get pod –all-namespaces.
* Kubectl exec database –i –t – mysql –u root –p <admin>

**Ingress**

* Alternate to load balance and node port.
* Default is available we can customize or create new.
* Nginex configuration
* Kubectl create –f ingress.yml
* Kubectl get ingress
* Kubectl create –f helloworld-v1.yml v2.yml
* Kubectl create –f echoservice.yml

**Volume**

* Persistence storage information for state full application**.**
  + - **NFS Setup**
* ufw allow from 172.30.0.0/24 to any port nfs
* apt-get install nfs-kernel-server (nfs-common for client)
* cat /etc/exports
* /exports (rw,sync,no\_subtree\_check)
* systemctl restart nfs-mountd
* mount -t nfs kube-master:/exports /mnt

**Volume in K8s**

* kubectl get pv
* kubectl create –f pv/busy box

**Container background :** [**http://delivery.acm.org/10.1145/2900000/2898444/p10-burns.pdf?ip=183.82.116.191&id=2898444&acc=OPEN&key=4D4702B0C3E38B35%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35%2E6D218144511F3437&\_\_acm\_\_=1529504863\_be8f3e7e7832ecb5c7a8f1ffded4fa97**](http://delivery.acm.org/10.1145/2900000/2898444/p10-burns.pdf?ip=183.82.116.191&id=2898444&acc=OPEN&key=4D4702B0C3E38B35%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35%2E6D218144511F3437&__acm__=1529504863_be8f3e7e7832ecb5c7a8f1ffded4fa97)

**Horizontal pod expansion**

* kubectl create –f hpe-example.yml
* kubectl get hpa
* kubectl run -i --tty load-genrater --image=busybox /bin/sh
* kubectl get all

**K8s Master Services**

**Quota : Namespace quota and cluster level** quota.

Kubectl create namespace myname

Kubectl get namespace

kubectl get rs

Configmaps

kubectl get pods --v=7 ----let you know the API’s

kubectl get limits –namespace=myspace

kubectl get deployment -n myspace

kubectl get pods –namespace=myspace

kubectl describe deployment –n myspace

**uht global------Training in 4 hours online training by amit.**

**Authorization**

**RBAC: Access.**

* –authorization-mode=rbac
* Kops and kubeadm uses rbac.
* Kubectl to grant permission
* Role is namespace
* Cluster role: cluster admin
* Rolebinding: single name space.
* Role can like see pods secrets ,list,watch,
* Assigned the role to user.
* Rolebinding –
* To spine up on all name spaces need to reader at cluster wide.
* User creation :
* sudo apt install openssl
* openssl genrsa -out amit.pem 2048
* openssl req -new -key amit.pem -out amit-csr.pem -subj "/CN=amit/O=myteam/"
* openssl x509 -req -in amit-csr.pem -CA ca.crt -CAkey ca.key -CAcreateserial -out amit.crt -days 10000
* ```
* ## add new context
* ```
* kubectl config set-credentials amit --client-certificate=amit.crt --client-key=amit.pem
* kubectl config set-context amit --cluster=kubernetes.newtech.academy --user amit
* ```
* Kubectl config get-contexts
* Kubectl config use-context kubernetes-admin@kubernetes.
* Kubectl config use-context amit
* Kubectl create –f admin-user.yml
* Kubectl delete –f admin-user.yml
* Kubectl config use-context amit

**Configmap**

* Global configuration for group of applicaions.
* NGINEX variables
* Env
* Command line arguments
* Volume etc
* Kubectl create configmap nginx-config –from-file=reverseproxy.conf.
* Kubectl get configmap.
* Kubectl describe configmap nginx-config.
* Kubectl