**initContainers:**

Create a pod with application Nginx

The nginx application should start after 100 seconds of Pod creation.

Create init container with custom command to sleep for 100 sec (sleep 100)

**Static Pods:**

Create a static pod with Nginx on worker1

On a worker Node:

1. Append the below text at the end of the last line in the file “/etc/systemd/system/kubelet.service.d/10-kubeadm.conf”:
   1. --pod-manifest-path=/etc/kubelet.d/
2. Make dir /etc/kubelet.d/
3. Reload the configuration file
   1. sudo systemctl daemon-reload
4. Restart kubelet service
   1. sudo kubelet restart
5. Dump a pod manifest file in the above directory
6. Check the pods by running kubectl commands on master node

| **apiVersion**: v1  **kind**: Pod  **metadata**:  **name**: static-pod  **spec**:  **containers**:  **- image**: nginx  **name**: nginx |
| --- |

**Ingress**

**Install Ingress controller**

| kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/baremetal/deploy.yaml> |
| --- |

Get nginx controller endpoint

kubectl get svc -n ingress-nginx ingress-nginx-controller

| Output  NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  ingress-nginx-controller NodePort 10.101.148.251 <none> 80:**32043**/TCP,443:30676/TCP 24m |
| --- |

**Define Ingress Rules**

Create backend services

| kubectl create deployment dep1 --image nginx --port 80  kubectl expose deployment dep1 --name svc1 --target-port 80  kubectl create deployment dep2 --image httpd --port 80  kubectl expose deployment dep2 --name svc2 --target-port 80 |
| --- |

Create ingress Rules

| **apiVersion**: networking.k8s.io/v1  **kind**: Ingress  **metadata**:  **name**: ingres-demo  **spec**:  **rules**:  **- http**:  **paths**:  **- path**: /account  **pathType**: Prefix  **backend**:  **service**:  **name**: svc1  **port**:  **number**: 80  **- path**: /cart  **pathType**: Prefix  **backend**:  **service**:  **name**: svc2  **port**:  **number**: 80 |
| --- |

Explore the created ingress

kubectl describe ingress ingres-demo

Browse the application on any node on the cluster

[http://localhost:32043/account](http://localhopst:32043/account)

[http://localhost:32043/cart](http://localhopst:32043/cart)

**Manage Node’s workloads**

**Cordon**

1. Cordon off worker2 node
   1. kubectl cordon <nodename>
2. Observe the pods
   1. kubectl get pods -o wide
3. Check node status
   1. kubectl get nodes
4. Scale up a deployment
   1. kubectl scale deployment <deployment name> --replicas 10
5. Observe the pods
   1. kubectl get pods -o wide

**Drain**

1. Drain worker2 Node
   1. kubectl drain worker2 --force --ignore-daemonsets
2. Observe the pods
   1. kubectl get pods -o wide
3. Check node status
   1. kubectl get nodes
4. Uncordon the node
   1. kubectl uncordon worker2

**Commands dump**

| vim initC.yaml  kubectl delete svc mydbsvc  kubectl apply -f initC.yaml  kubectl logs -f pod-with-initc -c container-to-check-db-service  kubectl expose deployment mydep --name mydbsvc --port 80  kubectl get pods  kubectl logs -f pod-with-initc -c container-to-check-db-service  kubectl get pods  cat initC.yaml  kubectl get pods -A  kubectl get deployments.apps -A  kubectl get pods  kubectl get pods -o wide  kubectl delete pod static-web-worker1  kubectl get pods -o wide  kubectl get pods -A  cd /etc/kubernetes/manifests/  ll  service kubelet status  cd ..  ll  cat kubelet.conf  vim /var/lib/kubelet/config.yaml  ll  kubectl get pods  kubectl get pod nginx-7848d4b86f-g226t -o yaml  kubectl apply https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/baremetal/deploy.yaml  kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/baremetal/deploy.yaml  kubectl get ns  kubectl get all -n ingress-nginx  kubecl get svc -A  kubectl get svc -A  cd  ll  vim ingress-rules.yaml  kubectl create deployment --name dep1 --image nginx  kubectl create deployment dep1 --image nginx  kubectl expose deployment dep1 --name svc1 --target-port 80  kubectl delete deployments.apps dep1  kubectl create deployment dep1 --image nginx --port 80  kubectl expose deployment dep1 --name svc1 --target-port 80  kubectl get svc  kubectl create deployment dep2 --image httpd --port 80  kubectl expose deployment dep2 --name svc2 --target-port 80  kubectl get svc  vim ingress-rules.yaml  kubectl apply -f ingress-rules.yaml  vim ingress-rules.yaml  kubectl apply -f ingress-rules.yaml  kubectl get ingress  kubectl logs dep1-bfd6cd757-crq5g nginx  kubectl logs -f dep1-bfd6cd757-crq5g nginx  kubectl logs -f dep2-7467887f6b-qw9dr httpd  kubectl get ingress  kubectl get nodes -o wide  cat ingress-rules.yaml  history  kubectl get svc -A  kubectl get svc -n ingress-nginx ingress-nginx-controller  kubectl describe ingress ingres-demo  kubectl version  kubeadm upgrade -h  kubeadm upgrade plan  kubeadm --version  kubeadm version  kubeadm upgrade plan -h  kubeadm upgrade plan 1.22.0  kubeadm upgrade  kubeadm upgrade -h  kubeadm upgrade apply 1.22.0 --dry-run  kubeadm upgrade diff 1.22.0  kubectl get pods -o wide  kubectl drain worker1  kubectl cordon worker1  kubectl uncordon worker1  kubectl cordon worker1  kubectl get nodes  kubectl get pods  kubectl get pods -o wide  kubectl get dep  kubectl get deployments.apps  kubectl scale deployment dep1 --replicas=10  kubectl get pods -o wide  kubectl uncordon worker1  kubectl scale deployment dep2 --replicas=10  kubectl get pods -o wide  kubectl drain worker1  kubectl drain worker1 --force --ignore-daemonsets  kubectl get pods -o wide  kubectl get nodes  kubectl drain -h  kubectl uncordon worker |
| --- |