**Kubernetes Cluster Upgrade: Manual Commands and Flow**

**Pre-Upgrade Preparation**

**1. Check Current State**

kubectl version --short

kubectl get nodes -o wide

kubectl get pods --all-namespaces -o wide

**2. Review Pod Disruption Budgets (PDBs)**

# List all existing PDBs

kubectl get pdb --all-namespaces

# Check PDB details

kubectl describe pdb <pdb-name> -n <namespace>

# Create PDBs for critical applications if missing

kubectl create pdb <app-pdb> --namespace=<namespace> --selector=app=<label> --min-available=1

# OR

kubectl create pdb <app-pdb> --namespace=<namespace> --selector=app=<label> --max-unavailable=1

**3. Backup etcd**

ETCDCTL\_API=3 etcdctl snapshot save /backup/etcd-backup.db \

--endpoints=https://127.0.0.1:2379 \

--cacert=/etc/kubernetes/pki/etcd/ca.crt \

--cert=/etc/kubernetes/pki/etcd/server.crt \

--key=/etc/kubernetes/pki/etcd/server.key

# Verify backup

ETCDCTL\_API=3 etcdctl snapshot status /backup/etcd-backup.db

**4. Check Available Versions**

apt update

apt-cache madison kubeadm | head -10

**Upgrade Flow - Control Plane First**

**MASTER NODE 1 (Primary Control Plane)**

**Step 1: Drain Master Node**

kubectl drain master-1 --ignore-daemonsets

kubectl get nodes

**Step 2: Upgrade kubeadm**

apt-mark unhold kubeadm

apt-get update

apt-get install -y kubeadm=1.29.0-00

apt-mark hold kubeadm

kubeadm version

**Step 3: Check Upgrade Plan**

sudo kubeadm upgrade plan

**Step 4: Apply Upgrade**

sudo kubeadm upgrade apply v1.29.0

**Step 5: Upgrade kubelet and kubectl**

apt-mark unhold kubelet kubectl

apt-get update

apt-get install -y kubelet=1.29.0-00 kubectl=1.29.0-00

apt-mark hold kubelet kubectl

**Step 6: Restart Services**

sudo systemctl daemon-reload

sudo systemctl restart kubelet

sudo systemctl status kubelet

**Step 7: Uncordon Node**

kubectl uncordon master-1

kubectl get nodes

**ADDITIONAL MASTER NODES (if HA setup)**

**Step 1: Drain Node**

kubectl drain master-2 --ignore-daemonsets

**Step 2: Upgrade kubeadm**

apt-mark unhold kubeadm

apt-get update

apt-get install -y kubeadm=1.29.0-00

apt-mark hold kubeadm

**Step 3: Upgrade Node Config**

sudo kubeadm upgrade node

**Step 4: Upgrade kubelet and kubectl**

apt-mark unhold kubelet kubectl

apt-get update

apt-get install -y kubelet=1.29.0-00 kubectl=1.29.0-00

apt-mark hold kubelet kubectl

**Step 5: Restart Services**

sudo systemctl daemon-reload

sudo systemctl restart kubelet

**Step 6: Uncordon Node**

kubectl uncordon master-2

kubectl get nodes

**Upgrade Worker Nodes (One by One)**

**For Each Worker Node:**

**Step 1: Check PDB Status Before Draining**

# Check which pods are on this node

kubectl get pods --all-namespaces -o wide | grep worker-1

# Check if PDBs will be violated

kubectl get pdb --all-namespaces

**Step 2: Drain Worker Node**

kubectl drain worker-1 --ignore-daemonsets --delete-emptydir-data

kubectl get nodes

**Step 3: SSH to Worker and Upgrade kubeadm**

ssh worker-1

apt-mark unhold kubeadm

apt-get update

apt-get install -y kubeadm=1.29.0-00

apt-mark hold kubeadm

**Step 4: Upgrade Node Configuration**

sudo kubeadm upgrade node

**Step 5: Upgrade kubelet and kubectl**

apt-mark unhold kubelet kubectl

apt-get update

apt-get install -y kubelet=1.29.0-00 kubectl=1.29.0-00

apt-mark hold kubelet kubectl

**Step 6: Restart kubelet**

sudo systemctl daemon-reload

sudo systemctl restart kubelet

sudo systemctl status kubelet

exit

**Step 7: Uncordon Node**

kubectl uncordon worker-1

kubectl get nodes

**Step 8: Verify Pods are Rescheduled**

kubectl get pods --all-namespaces -o wide | grep worker-1

**Verification Commands After Each Node**

# Check node status

kubectl get nodes

# Check system pods

kubectl get pods -n kube-system

# Check PDB status

kubectl get pdb --all-namespaces

# Check events for issues

kubectl get events --sort-by='.lastTimestamp' -A | head -20

**Final Verification After All Nodes**

# Verify all nodes on new version

kubectl get nodes -o jsonpath='{range .items[\*]}{.metadata.name}{"\t"}{.status.nodeInfo.kubeletVersion}{"\n"}{end}'

# Check all system components

kubectl get pods -n kube-system

kubectl get cs

# Verify API versions

kubectl api-versions

# Check for any pending or failed pods

kubectl get pods --all-namespaces | grep -v Running | grep -v Completed

# Verify PDBs are satisfied

kubectl get pdb --all-namespaces

**Important Notes on PDBs During Upgrade**

1. **Before draining any node:**

kubectl get pdb --all-namespaces -o wide

1. **Check if drain would violate PDB:**

kubectl drain <node-name> --dry-run=client --ignore-daemonsets

1. **If PDB blocks drain, check details:**

kubectl describe pdb <pdb-name> -n <namespace>

kubectl get deployment <deployment-name> -n <namespace>

1. **Temporarily adjust PDB if necessary:**

# Edit PDB to allow drain

kubectl edit pdb <pdb-name> -n <namespace>

# Change minAvailable or maxUnavailable temporarily

1. **Restore PDB after node upgrade:**

kubectl edit pdb <pdb-name> -n <namespace>

# Restore original values

**Rollback (if needed)**

# On master node

sudo kubeadm upgrade apply <previous-version> --force

# Restore etcd if necessary

ETCDCTL\_API=3 etcdctl snapshot restore /backup/etcd-backup.db \

--data-dir=/var/lib/etcd-backup

# Downgrade packages

apt-get install -y kubeadm=<previous-version> kubelet=<previous-version> kubectl=<previous-version>

apt-mark hold kubeadm kubelet kubectl