**Kubernetes Cluster Deployment with kops**

**Table of Contents**

1. Prerequisites Setup
   * DNS Name Configuration
   * S3 Bucket Creation
   * EC2 Instance Setup
   * IAM Role Configuration
   * SSH Key Generation
2. Tool Installation
   * kops Installation
   * kubectl Installation
   * Environment Configuration
3. Cluster Deployment
   * Cluster Configuration
   * Cluster Creation
   * Cluster Validation
4. Cleanup
5. Real-World Use Case

**Prerequisites Setup**

**1. DNS Name Configuration**

* Purchase a domain (e.g., cloudvishwakarma.in) from a registrar like GoDaddy
* Configure DNS settings to point to AWS Route 53 nameservers
* Required for cluster component communication and Ingress controllers

**2. S3 Bucket Creation**

bash

aws s3api create-bucket --bucket cloudvishwakarma.in --region us-east-1

* Stores cluster state and configuration
* Must be globally unique name
* Enable versioning for disaster recovery

**3. EC2 Instance Setup**

* Launch EC2 instance:
  + Type: t2.medium (minimum recommended)
  + OS: Ubuntu 20.04/22.04 LTS
  + IAM role: kops-role (created in next step)
  + Security group: Allow SSH (port 22) from your IP

**4. IAM Role Configuration**

1. Create IAM policy with these permissions:
   * AmazonEC2FullAccess
   * AmazonRoute53FullAccess
   * AmazonS3FullAccess
   * IAMFullAccess
   * AmazonVPCFullAccess
2. Create IAM role:

bash

aws iam create-role --role-name kops-role --assume-role-policy-document <file://trust-policy.json>

1. Attach policy to role:

bash

aws iam attach-role-policy --role-name kops-role --policy-arn arn:aws:iam::aws:policy/AdministratorAccess

1. Attach role to EC2 instance

**5. SSH Key Generation**

bash

ssh-keygen -t ed25519 -f ~/.ssh/id\_ed25519

* Keys will be used by kops to access all cluster nodes
* Public key will be distributed to all nodes automatically

**Tool Installation**

**1. kops Installation**

bash

wget https://github.com/kubernetes/kops/releases/download/v1.28.0/kops-linux-amd64

mv kops-linux-amd64 /usr/local/bin/kops

chmod +x /usr/local/bin/kops

* Version 1.28.0 used here (check for latest stable version)
* Installs to /usr/local/bin for system-wide access

**2. kubectl Installation**

bash

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

mv kubectl /usr/local/bin/

chmod +x /usr/local/bin/kubectl

* Kubernetes CLI tool for cluster management

**3. Environment Configuration**

Edit ~/.bashrc:

bash

export NAME=cloudvishwakarma.in

export KOPS\_STATE\_STORE=s3://cloudvishwakarma.in

export AWS\_REGION=us-east-1

export CLUSTER\_NAME=cloudvishwakarma.in

export EDITOR='/usr/bin/nano'

alias ku=kubectl

Apply changes:

bash

source ~/.bashrc

* Sets up required environment variables
* Creates handy alias for kubectl

**Cluster Deployment**

**1. Cluster Configuration**

Generate cluster manifest:

bash

kops create cluster --name=cloudvishwakarma.in \

--state=s3://cloudvishwakarma.in --zones=us-east-1a,us-east-1b \

--node-count=2 --control-plane-count=1 --node-size=t3.medium \

--control-plane-size=t3.medium --control-plane-zones=us-east-1a \

--control-plane-volume-size 10 --node-volume-size 10 \

--ssh-public-key ~/.ssh/id\_ed25519.pub \

--dns-zone=cloudvishwakarma.in --dry-run --output yaml > cluster.yml

Configuration details:

* **name**: Cluster name matching DNS
* **state**: S3 bucket for cluster state
* **zones**: AWS availability zones
* **node-count**: Number of worker nodes
* **node-size**: EC2 instance type for workers
* **control-plane-count**: Master nodes (1 for dev, 3 for production)
* \***-volume-size**: Storage size in GB
* **ssh-public-key**: For node access

**2. Cluster Creation**

bash

kops create -f cluster.yml

kops update cluster --name cloudvishwakarma.in --yes --admin

* First command creates cluster configuration
* Second command provisions actual AWS resources
* Takes 5-10 minutes to complete

**3. Cluster Validation**

bash

kops validate cluster --wait 10m

Checks:

* All nodes are Ready
* Control plane components are healthy
* DNS records are properly configured

Verify with:

bash

kubectl get nodes

kubectl -n kube-system get pods

**Cleanup**

bash

kops delete -f cluster.yml --yes

aws s3 rb s3://cloudvishwakarma.in --force

* First command destroys all cluster resources
* Second command removes the state bucket
* Wait for all resources to be deleted before terminating management node

**Real-World Use Case**

**Scenario**: E-commerce platform deployment

**Challenge**:

* Handle traffic spikes during sales
* Zero-downtime deployments
* Cost-effective scaling

**Solution**:

1. Deploy 6-node cluster (3 masters, 3 workers) using the commented configuration
2. Set up Horizontal Pod Autoscaler for frontend pods
3. Use StatefulSets for databases
4. Implement rolling updates for application deployments

**Result**:

* 300% traffic increase handled automatically
* Deployment downtime reduced from 30 minutes to zero
* Infrastructure costs optimized by scaling down during off-peak

**Maintenance Tip**: Use kops rolling update to upgrade nodes without downtime:

bash

kops rolling-update cluster --name cloudvishwakarma.in --yes

**Key Components Explained Simply**

**1. Cluster State (S3 Bucket)**

* **What it is**: Kubernetes' memory bank
* **Contains**:
  + Node configurations
  + Network settings
  + Security policies
* **Example Command**:

bash

aws s3 ls s3://your-cluster-state

**2. Control Plane (Master Nodes)**

* **Functions**:
  + API Server (front desk)
  + Scheduler (assignment manager)
  + Controller Manager (health monitor)
* **Typical Setup**:

bash

--control-plane-count=1 --control-plane-size=t3.medium

**3. Worker Nodes**

* **Responsibilities**:
  + Run application pods
  + Report status to control plane
* **Configuration Example**:

bash

--node-count=2 --node-size=t3.small

**4. Pods**

* **Basic Unit**:
  + Smallest deployable object
  + Contains 1+ containers
* **Lifecycle**:

bash

kubectl get pods -w *# Watch pod states*

**Detailed Command Reference**

**Cluster Creation**

bash

kops create cluster \

--name=cluster.yourdomain.com \

--state=s3://your-state-bucket \

--zones=us-east-1a,us-east-1b \

--node-count=2 \

--node-size=t3.small \

--ssh-public-key ~/.ssh/id\_ed25519.pub \

--dry-run --output yaml

**Daily Operations**

| **Task** | **Command** |
| --- | --- |
| Check nodes | kubectl get nodes -o wide |
| View pods | kubectl get pods -A |
| Check services | kubectl get svc |
| View cluster info | kops get cluster |
| Validate cluster | kops validate cluster |

**Troubleshooting**

bash

*# Get detailed node information*

kubectl describe node <node-name>

*# View pod logs*

kubectl logs <pod-name>

*# SSH into node*

kops ssh --name=cluster.yourdomain.com <node-name>

**Visual Cluster Overview**

[ Control Plane ]

├─ API Server (kube-apiserver)

├─ Scheduler (kube-scheduler)

└─ Controller Manager (kube-controller-manager)

[ Worker Nodes ]

├─ Node 1

│ ├─ Pod A (app + sidecar)

│ └─ Pod B (database)

└─ Node 2

├─ Pod C (app)

└─ Pod D (monitoring)

**Important Notes**

1. **Cluster State** is persistent - survives node failures
2. **Master Nodes** should be odd-numbered (1,3,5) for HA
3. **Worker Nodes** can be added/removed dynamically
4. **All changes** should go through kops for consistency

This structure provides both a comprehensive reference and beginner-friendly explanations. The table of contents allows easy navigation to specific topics while maintaining logical flow from setup to operations.