1) Base OS prerequisites

Install base tools

apt-get install -y apt-transport-https ca-certificates curl gnupg lsb-release software-properties-common

Kernel modules and sysctls for Kubernetes networking

modprobe overlay

modprobe br_netfilter

tee /etc/sysctl.d/99-kubernetes-cri.conf >/dev/null <<'EOF'

net.bridge.bridge-nf-call-iptables = 1

net.bridge.bridge-nf-call-ip6tables = 1

net.ipv4.ip_forward   = 1

EOF

sysctl --system

---

2) Install containerd (recommended for kubeadm)

Install containerd from Ubuntu repos (simple and stable)

apt-get install -y containerd

Generate default config and switch to systemd cgroups (important!)

mkdir -p /etc/containerd

containerd config default | tee /etc/containerd/config.toml >/dev/null

sed -i 's/SystemdCgroup = false/SystemdCgroup = true/' /etc/containerd/config.toml

Use systemd-resolved for DNS if needed (optional but helps pull issues on some hosts):

sed -i 's|sandbox_image = ".*"|sandbox_image = "registry.k8s.io/pause:3.9"|' /etc/containerd/config.toml

systemctl daemon-reload

systemctl enable --now containerd

Note: We're not installing Docker Engine since Kubernetes uses containerd directly. If you still want Docker CLI for local builds, install it **after** Kubernetes is up (Section 8).

---

3) Install Kubernetes (kubeadm / kubelet / kubectl)

Add Google Cloud apt repo for Kubernetes

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' \

  | tee /etc/apt/sources.list.d/kubernetes.list

apt-get update

Install a consistent version set (example: 1.30.x)

apt-get install -y kubelet kubeadm kubectl

apt-mark hold kubelet kubeadm kubectl

Ensure swap is off (again)

swapoff -a

---

4) Create the single-node cluster (kubeadm + Flannel)

We'll give the cluster a pod CIDR that Flannel expects: **10.244.0.0/16**.

Pull control-plane images (optional, kubeadm will do it anyway)

kubeadm config images pull

Initialize the cluster

kubeadm init --pod-network-cidr=10.244.0.0/16

Configure kubectl for your user

mkdir -p $HOME/.kube

cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

chown $(id -u):$(id -g) $HOME/.kube/config

Allow scheduling on the control-plane node (single-node setup)

kubectl taint nodes --all node-role.kubernetes.io/control-plane- || true

kubectl taint nodes --all node-role.kubernetes.io/master- || true

Install **Flannel** CNI (avoids your previous Tigera/Calico CRD annotation bloat):

kubectl apply -f https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation/kube-flannel.yml

Wait for node to become Ready

kubectl get nodes -w

If the node stays NotReady, check:

kubectl -n kube-flannel get pods -o wide

kubectl describe node $(hostname)

journalctl -u kubelet -f

Common fixes: ensure sysctls in Section 1 are applied, containerd is running, and DNS resolves.

---

5) Install Helm (cleanly)

Using apt (or use official script if you prefer)

curl https://baltocdn.com/helm/signing.asc | gpg --dearmor -o /usr/share/keyrings/helm.gpg

echo "deb [signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all main" \

  | tee /etc/apt/sources.list.d/helm-stable-debian.list

apt-get update

apt-get install -y helm

helm version

---

6) (Optional) Metrics & sanity checks

Metrics Server (optional, nice for HPA & kubectl top)

kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

Watch core components

kubectl get pods -A -w

---

7) Install Camunda 8 with Helm (single-node friendly)

We'll put Camunda in its own namespace and use a **values.yaml** tuned for a single node. Key points that address today's pain:

- Don't patch Elasticsearch anti-affinity live; configure it via Helm values.
- Keep replicas at 1.
- Keep resources light; disable Optimize if low on RAM.

helm repo add camunda https://helm.camunda.io

helm repo update

kubectl create namespace camunda

Create camunda-values.yaml:

cat > camunda-values.yaml <<'EOF'

Minimal single-node friendly Camunda 8 (Camunda Platform)

Disables strict anti-affinity so ES/Zeebe can schedule on the single node.

global:

  image:

    # You can pin versions if needed, otherwise defaults to chart defaults

```
    # tag: 8.6.0
  ingress:
    enabled: false  # we'll use port-forward or NodePort

zeebe:
  enabled: true
  clusterSize: 1
  partitionCount: 1
  replicationFactor: 1
  resources:
    requests:
      cpu: "200m"
      memory: "512Mi"
    limits:
      cpu: "1"
      memory: "2Gi"

elasticsearch:
  enabled: true
  master:
    replicas: 1
    persistence:
      enabled: false
    nodeSelector: {}
    tolerations: []
    affinity: {} # disables requiredDuring... anti-affinity that blocked you before
  # To reduce memory footprint in dev:
  clusterHealthCheckParams: "wait_for_status=yellow&timeout=1s"
```

```
operate:
  enabled: true
  resources:
    requests:
      cpu: "100m"
      memory: "256Mi"
    limits:
      cpu: "500m"
      memory: "512Mi"

tasklist:
  enabled: true
  resources:
    requests:
      cpu: "100m"
      memory: "256Mi"
    limits:
      cpu: "500m"
      memory: "512Mi"

identity:
  enabled: true
  keycloak:
    enabled: true
    resources:
      requests:
        cpu: "100m"
```

```
      memory: "256Mi"
    limits:
      cpu: "500m"
      memory: "512Mi"

optimize:
  enabled: false  # disable to save memory in a small VM
EOF
```

Install:

```
helm install camunda camunda/camunda-platform -n camunda -f camunda-values.yaml

# Watch pods come up (first pull can take a bit)
kubectl -n camunda get pods -w
```

When all are Running/Ready, you can port-forward to access the apps:

```
# Operate (workflow management UI)
kubectl -n camunda port-forward svc/camunda-operate 8081:80

# Tasklist
kubectl -n camunda port-forward svc/camunda-tasklist 8082:80

# Identity/Keycloak login (Camunda web entry)
kubectl -n camunda port-forward svc/camunda-identity 8083:80
```

Open:

- http://localhost:8083 (Identity/Keycloak)
  Default credentials vary by chart; typically:
- **Keycloak** admin: admin / admin (or as set by chart)
- **Camunda** default users/clients are created by the chart; after logging into Identity, you can reach Operate and Tasklist via their URLs.

If you prefer NodePorts instead of port-forwards, set service.type: NodePort for each component in camunda-values.yaml.

---

8) (Optional) Install Docker Engine CLI (for local image builds)

Kubernetes already uses containerd. If you also want Docker CLI:

```
# Docker official repo
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
  gpg --dearmor -o /etc/apt/keyrings/docker.gpg
chmod a+r /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \
  https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo $VERSION_CODENAME) stable" \
  | tee /etc/apt/sources.list.d/docker.list > /dev/null
apt-get update
apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

# Add your user to docker group (log out/in after)
usermod -aG docker $SUDO_USER 2>/dev/null || usermod -aG docker $USER
```

---

9) Health checks & common fixes (covers today's errors)

Node NotReady:

```
kubectl get nodes
kubectl describe node $(hostname)
journalctl -u kubelet -n 200 --no-pager
```

- Ensure containerd is running, swap is disabled, Section 1 sysctls applied, Flannel pods are healthy.

- If DNS issues cause image pull failures, verify /etc/resolv.conf points to systemd-resolved or a valid upstream; restart containerd after changes.

Image pull errors (ErrImagePull / ImagePullBackOff):

kubectl -n camunda describe pod <pod-name> | sed -n '/Events/,$p'

ctr -n k8s.io images list | head

- Retry: kubectl delete pod <pod-name> (Deployment will re-create).
- If a private registry is involved, create an imagePullSecret and reference it in values.

Elasticsearch anti-affinity / scheduling failures:

- We disabled strict anti-affinity in camunda-values.yaml. Don't live-patch the StatefulSet; update via Helm values and helm upgrade.

Calico/Tigera CRD annotation bloat error:

- We've avoided Calico entirely by using Flannel. If you must use Calico later, **install only the Tigera operator + a matching Calico manifest version** and avoid layering multiple kustomize bases that inflate annotations.

Helm chart upgrades / reconfig:

Edit values then:

helm upgrade camunda camunda/camunda-platform -n camunda -f camunda-values.yaml

If things get weird:

helm -n camunda get values camunda

helm -n camunda get manifest camunda | less

Clean uninstall of Camunda (namespace-scoped):

helm -n camunda uninstall camunda

kubectl delete ns camunda

---

10) Quick end-to-end verification

1) Cluster Ready

kubectl get nodes

2) CNI Ready

kubectl -n kube-flannel get pods

3) Camunda services exposed

kubectl -n camunda get svc

4) Pods healthy

kubectl -n camunda get pods

5) Login via Identity and reach Operate/Tasklist (via port-forward or NodePort)
