

Ridgeserve DevOps Journey

Abhishek Tiwari

26 August 2026

CONTENTS

Ridgeserve DevOps Journey	
Ye document kya hai	
Server ka parichay	
Shuruat: ek galti jisne ek ghanta khaya	
Debugging kaise ki	
Recovery: boot volume rescue	
Asli wajah	
Isse kya seekha	
Phase 0 — Foundation	
Kya banaya	
Kyun	
Commands	
Jo install hua	
Phase 1 — Containers	
Kya banaya	
Kyun	
Commands	
Dhyan dene wali baat	
Phase 2 — Git server, CI/CD, aur asli HTTPS	
Kya banaya	
Kyun	
Domain (DuckDNS)	
Gitea	
nginx + Let's Encrypt	
CI runner	
Pehli pipeline	
Phase 3 — Kubernetes	
Kya banaya	
Kyun	

Ek asli decision

Commands

Ek galti jo hui

Manifests

Phase 4 — GitOps (ArgoCD)

Kya banaya

Kyun

Install

Application

Dono behaviours test kiye

Ek chhoti trick

Phase 5 — Observability

Kya banaya

Kyun

Install

Verify kaise kiya

Phase 6 — Infrastructure as Code

Kya banaya

Kyun

OpenTofu

State file

Ansible

Phase 7 — Security

Kya banaya

Vault kyun

Install

Seal aur unseal

Secret store karna

Trivy

CI mein security gate

Alag se: PostgreSQL for ecomersehub

Access: tunnel, public port nahi

Cheat Sheet

Roz ka

Containers

Kubernetes

ArgoCD

Vault

nginx aur certificates

IaC

Security

Debugging

Gotchas ki list

1. Port kholne ke do layers hote hain
2. ssh.socket se port mat badalna
3. kubectl apply directory ko alphabetically padhta hai
4. k3s ko Docker ki images nahi dikhtin
5. Boot volume ka AVAILABLE ka matlab detached nahi
6. OCIDs 60 characters ke hote hain (.. ke baad)
7. Distroless containers mein debugging tools nahi hote
8. Gitea ko INSTALL_LOCK chahiye
9. act_runner ko network batana padta hai
10. Bade CRDs ko server-side apply chahiye

Aaj ka final state

Live services

Numbers

Git history

Ek zaroori deadline

Aage kya

Ridgeserve DevOps Journey

Ye document kya hai

26 August 2026 ko humne ek khaali Oracle Cloud server liya aur usko ek poora DevOps lab bana diya. Ye document batata hai ki **kya kiya, kyun kiya, aur kaise kiya** — har step ki asli command ke saath.

Ye tutorial nahi hai jo internet se copy kiya gaya ho. Ye wahi hai jo sach mein is server pe chala, jismein wo galtiyan bhi shamil hain jo raaste mein hui. Galtiyan isliye rakhi hain kyunki asli seekh wahin se milti hai.

Padhne ka tareeka: har phase mein pehle “kya banaya” hai, phir “kyun”, phir commands. Agar sirf command chahiye to seedha aakhir mein **Cheat Sheet** dekh lena.

Server ka parichay

Cheez	Value
Console naam	abhishekdevelopment
IP	144.24.122.201
Domain	ridgeserve.duckdns.org
CPU	4 OCPU, arm64 (Ampere A1)
RAM	24 GB
Disk	193 GB
OS	Ubuntu 24.04.4 LTS
Region	India West (Mumbai)

Ghusne ka tareeka:

```
ssh oracle
```

Bas. Key, IP, username sab `~/.ssh/config` mein set hai. `ssh oracle-big` bhi wahi server hai.

arm64 ka matlab — ye ARM processor hai, x86 nahi. Zyadatar modern software arm64 support karta hai, par kabhi-kabhi koi Docker image ya binary sirf amd64 ka milta hai. Tab dhoondhna padta hai. Ye asli production mein bhi hota hai, isliye ye seekhne layak problem hai, dikkat nahi.

Shuruat: ek galti jisne ek ghanta khaya

Sabse pehla kaam SSH ko secure karna tha. Maine SSH port 22 se 2222 pe shift karne ki koshish ki. Ubuntu 24.04 mein sshd **socket-activated** hota hai, matlab port `sshd_config` mein nahi, systemd socket unit mein set hota hai. Maine ye likha:

```
# /etc/systemd/system/ssh.socket.d/override.conf
[Socket]
ListenStream=
ListenStream=22
ListenStream=2222
```

Restart kiya. `ss -tlnp` ne dikhaya ki dono ports listen kar rahe hain. Sab theek lag raha tha. **Phir SSH पूरी तरह मर गया.**

Debugging kaise ki

Andar ghus nahi sakte the, to bahar se hi evidence jama karna pada. Alag-alag ports probe kiye:

```
for p in 22 2222 80 9999; do
  printf "port %s: " "$p"
  timeout 12 ssh -o ConnectTimeout=10 -p $p ubuntu@144.24.122.201 true 2>&1 | head -1
done
```

Result:

```
port 22: Connection refused
port 2222: Connection refused
port 80: Connection timed out
port 9999: Connection timed out
```

Ye difference sona hai. Yaad rakhna:

- **Timeout** = packet host tak pahuncha hi nahi. Firewall (yahan OCI Security List) ne chup-chaap drop kar diya.
- **Connection refused** = packet host tak pahuncha, aur host ne active reject kiya. Matlab machine zinda hai, bas us port pe koi listen nahi kar raha.

To pata chala: server chal raha hai, sirf sshd sun nahi raha.

Phir maine check kiya ki kahin mera IP hi to ban nahi hua (fail2ban se). Uske liye **purane server se** probe kiya, jo alag IP tha:

```
ssh oracle "timeout 8 bash -c 'echo > /dev/tcp/144.24.122.201/22'"
# bash: connect: Connection refused
```

Wahan se bhi refused. Matlab ban nahi tha, sach mein listener hi nahi tha.

Recovery: boot volume rescue

Andar jaane ka koi raasta nahi bacha tha. Serial Console bhi kaam nahi karta kyunki OCI ki Ubuntu image mein kisi user ka password hi set nahi hota.

Jo kiya:

1. Broken instance **Stop** kiya
2. Uska **boot volume detach** kiya
3. Wahi volume **doosre instance pe data disk ki tarah attach** kiya
4. Wahan mount karke file delete ki
5. Wapas boot volume attach karke Start kiya

Console ka "Attach block volume" button greyed out tha, to Cloud Shell se CLI use ki:

```
oci compute volume-attachment attach \
  --type paravirtualized \
  --instance-id <doosre-instance-ka-ocid> \
  --volume-id <broken-boot-volume-ka-ocid>
```

Disk `/dev/sdb` ban ke aa gaya. Mount karke:

```
sudo mkdir -p /mnt/rescue
sudo mount /dev/sdb1 /mnt/rescue
cat /mnt/rescue/etc/hostname      # confirm sahi disk hai
```

Asli wajah

Sabse zaroori step: **us disk pe purane boot ke journal logs padhe.**

```
sudo journalctl --directory=/mnt/rescue/var/log/journal \
  --no-pager -b -1 -u ssh.socket -u ssh.service
```

Aur jawab mil gaya. Mere change se **pehle**:

```
sshd: Server listening on 0.0.0.0 port 22.    <- IPv4
sshd: Server listening on :: port 22.         <- IPv6
```

Mere change ke **baad**:

```
sshd: Server listening on :: port 22.         <- sirf IPv6
sshd: Server listening on :: port 2222.       <- sirf IPv6
```

Default `ssh.socket` alag-alag IPv4 aur IPv6 sockets banata hai. Maine `ListenStream=` se sab reset karke sirf port numbers likhe, to systemd ne har port ka **sirf IPv6** socket banaya. Hum IPv4 se connect karte hain, wahan koi listener nahi tha, kernel ne RST bhej diya.

Fix ek line ka tha:

```
sudo rm /mnt/rescue/etc/systemd/system/ssh.socket.d/override.conf  
sudo umount /mnt/rescue
```

Isse kya seekha

`ss` ka output verification nahi hota. `ss` ne “sab theek” dikhaya, `sshd -t` bhi pass hua, par asli IPv4 connection fail thi. Local check kabhi remote reality prove nahi karta.

Jab remote access ka raasta hi badal rahe ho, purana raasta khula rakho. Naya port kholo, use **alag session se** test karo, tabhi purana band karo. Maine ulta kiya aur phas gaya.

Recovery ka raasta pehle se bana ke rakho. Isliye ab server pe `abhishek` aur `root` ke passwords set hain — taaki agar dobara SSH mare to OCI Serial Console se ghus sakein, boot volume ka drama na karna pade.

Phase 0 — Foundation

Kya banaya

Languages, basic tooling, aur sabse important — **ek git repo jisme saara config rehta hai.**

Kyun

DevOps ka pehla usool: **kuch bhi haath se configure mat karo.** Agar aapne server pe koi file haath se badli aur 3 mahine baad server mar gaya, to aapko yaad nahi rahega ki kya-kya kiya tha. Agar sab git mein hai, to server dobara banana ek script chalane jaisa hai.

Isliye pehle din se `~/devops/` repo banaya aur har phase ka config wahin gaya.

Commands

```
# Base tooling
sudo apt-get update
sudo apt-get install -y btop jq tree ncd u fzf ripgrep bat make \
  build-essential unzip zip openjdk-21-jdk python3-pip python3-venv

# Go (official tarball, arm64)
GOVER=$(curl -fsSL "https://go.dev/VERSION?m=text" | head -1)
curl -fsSL "https://go.dev/dl/${GOVER}.linux-arm64.tar.gz" -o /tmp/go.tgz
sudo rm -rf /usr/local/go
sudo tar -C /usr/local -xzf /tmp/go.tgz
echo 'export PATH=$PATH:/usr/local/go/bin:$HOME/go/bin' | sudo tee /etc/profile.d/go.sh

# Node 22 LTS
curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -
sudo apt-get install -y nodejs

# yq aur uv
sudo curl -fsSL https://github.com/mikefarah/yq/releases/latest/download/yq_linux_arm64 \
  -o /usr/local/bin/yq && sudo chmod +x /usr/local/bin/yq
curl -LsSf https://astral.sh/uv/install.sh | sh

# Repo
mkdir -p ~/devops/{compose,k8s,ansible,terraform,scripts,docs,nginx}
cd ~/devops && git init
```

`.gitignore` shuru se hi likha, taaki secret kabhi galti se commit na ho:

```
*.tfstate
*.tfstate.*
.terraform/
.env
secrets/
.pg-*.creds
.vault-init.creds
```

Jo install hua

Tool	Version	Kyun
Go	1.27.0	Docker, Kubernetes, Terraform, Prometheus sab Go mein likhe hain
Node.js	22.23.2	Web apps aur CI
Python	3.12.3	Automation, Ansible
Java	21.0.12	Jenkins, Spring
ripgrep, jq, yq, btop, fzf	—	Roz ka kaam

Phase 1 — Containers

Kya banaya

Docker CE, Compose, buildx, lazydocker, aur ek 4-service practice stack.

Kyun

Container ek app ko uske saare dependencies ke saath ek box mein band kar deta hai. “Mere laptop pe to chalta tha” wali problem yahin khatam hoti hai. Aur aage ka poora stack (Kubernetes, CI runners, monitoring) containers pe hi khada hai.

Commands

```
# Docker ka official repo
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
  sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo "deb [arch=arm64 signed-by=/etc/apt/keyrings/docker.gpg] \
  https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io \
  docker-buildx-plugin docker-compose-plugin

# Apne user ko docker group mein (naya login chahiye iske baad)
sudo usermod -aG docker ubuntu
```

Practice stack `~/devops/compose/hello/compose.yaml`:

```

services:
  web:
    image: nginx:alpine
    ports: ["8080:80"]
    volumes:
      - ./site:/usr/share/nginx/html:ro
    depends_on: [api]

  api:
    image: python:3.12-alpine
    working_dir: /app
    command: python -m http.server 8000
    expose: ["8000"]
    depends_on: [db, cache]

  db:
    image: postgres:16-alpine
    environment:
      POSTGRES_PASSWORD: devpass
      POSTGRES_USER: dev
      POSTGRES_DB: devdb
    volumes:
      - db-data:/var/lib/postgresql/data

  cache:
    image: redis:alpine
    volumes:
      - cache-data:/data

volumes:
  db-data:
  cache-data:

```

Chalane ke commands:

```

cd ~/devops/compose/hello
docker compose up -d          # chalu karo
docker compose ps             # kya chal raha hai
docker compose logs -f web    # logs dekho
docker compose down -v        # band karo, volumes bhi hata do
lazydocker                    # TUI - sabse achha seekhne ka tareeka

```

Dhyan dene wali baat

`docker` **group ka matlab root hai**. Jo user docker group mein hai wo container se host ka `/` mount karke kuch bhi kar sakta hai. Ye standard practice hai, par jaan lena zaroori hai.

Phase 2 — Git server, CI/CD, aur asli HTTPS

Kya banaya

- **Gitea** — apna khud ka GitHub
- **Gitea Actions** + `act_runner` — apni CI/CD
- **nginx** reverse proxy + **Let's Encrypt** certificate
- **DuckDNS** se free domain

Kyun

Ye phase sabse important hai kyunki iske baad ka sab kuch isi pipeline se deploy hota hai. Aur Gitea Actions ka syntax **GitHub Actions ka hi hai** — jo yahan seekhoge wo GitHub pe seedha kaam aayega.

Domain (DuckDNS)

Certificate ke liye ek asli hostname chahiye. Domain khareedne ki zaroorat nahi: duckdns.org pe free subdomain milta hai. `ridgeserve` naam liya, IP set kiya, ho gaya.

Ek bonus: DuckDNS **wildcard subdomains** support karta hai, matlab `git.ridgeserve.duckdns.org`, `k8s.ridgeserve.duckdns.org` sab apne aap resolve hote hain. Isse har service ko apna subdomain mil gaya.

Gitea

```
~/devops/compose/gitea/compose.yaml:
```

```

services:
  gitea:
    image: gitea/gitea:latest
    restart: unless-stopped
    environment:
      GITEA__database__DB_TYPE: postgres
      GITEA__database__HOST: giteadb:5432
      GITEA__database__NAME: gitea
      GITEA__database__USER: gitea
      GITEA__database__PASSWORD: ${GITEA_DB_PASSWORD}
      GITEA__server__ROOT_URL: https://git.ridgeserve.duckdns.org/
      GITEA__actions__ENABLED: "true"
      GITEA__service__DISABLE_REGISTRATION: "true"
      GITEA__security__INSTALL_LOCK: "true"
    volumes:
      - /srv/gitea/data:/data
    ports:
      - "127.0.0.1:3000:3000"
    depends_on:
      giteadb:
        condition: service_healthy

  giteadb:
    image: postgres:16-alpine
    environment:
      POSTGRES_USER: gitea
      POSTGRES_PASSWORD: ${GITEA_DB_PASSWORD}
      POSTGRES_DB: gitea
    volumes:
      - gitea-db:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U gitea"]
      interval: 5s
      retries: 10

volumes:
  gitea-db:

```

Do cheezein yahan important hain:

INSTALL_LOCK: "true" — iske bina Gitea pehli baar web installer maangta hai aur `app.ini` banti hi nahi. Uske bina admin user CLI se nahi ban sakti. Ye maine galti se seekha: pehli koshish mein error aaya `Unable to load config file for a installed Gitea instance`.

Port 127.0.0.1:3000 — Gitea seedha internet pe nahi hai. nginx uske aage khada hai. Isse Gitea ko TLS ki chinta nahi karni padti.

Admin user:

```

docker compose exec -u git gitea gitea admin user create \
  --admin --username abhishek --password '<password>' --email <email>

```

nginx + Let's Encrypt

Reverse proxy config repo mein rehta hai, `/etc` mein nahi. Ek script use deploy karti hai — yahi Ansible ka chhota bhai hai:

```
# ~/devops/scripts/deploy-nginx.sh
for f in "$REPO"/nginx/sites/*.conf; do
  name=$(basename "$f")
  sudo cp "$f" "/etc/nginx/sites-available/$name"
  sudo ln -sf "/etc/nginx/sites-available/$name" "/etc/nginx/sites-enabled/$name"
done
sudo nginx -t && sudo systemctl reload nginx
```

Certificate:

```
sudo certbot --nginx \
  -d ridgeserve.duckdns.org \
  -d git.ridgeserve.duckdns.org \
  --non-interactive --agree-tos -m <email> --redirect
```

`--redirect` se HTTP apne aap HTTPS pe chala jata hai. Auto-renewal `certbot.timer` se hota hai, kuch karna nahi padta:

```
systemctl is-active certbot.timer      # active
sudo certbot certificates               # kab expire ho raha hai
```

Jab naya subdomain add karna ho to `--expand` use karo, saare domains ek saath likh ke:

```
sudo certbot --nginx --expand \
  -d ridgeserve.duckdns.org -d git.ridgeserve.duckdns.org \
  -d k8s.ridgeserve.duckdns.org -d argocd.ridgeserve.duckdns.org \
  -d grafana.ridgeserve.duckdns.org \
  --non-interactive --agree-tos -m <email> --redirect
```

CI runner

```
# Gitea se registration token
docker compose exec -u git gitea gitea actions generate-runner-token
```

Runner compose mein:

```
runner:
  image: gitea/act_runner:latest
  environment:
    GITEA_INSTANCE_URL: http://gitea:3000
    GITEA_RUNNER_REGISTRATION_TOKEN: ${GITEA_RUNNER_TOKEN}
    GITEA_RUNNER_LABELS: "ubuntu-latest:docker://catthehacker/ubuntu:act-latest"
    CONFIG_FILE: /config.yaml
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
    - ./runner-config.yaml:/config.yaml:ro
```

`runner-config.yaml` mein ek line bahut zaroori hai:

```
container:
  network: gitea_default    # job containers ko "gitea" resolve karna hai
```

Iske bina jobs Gitea tak pahunch hi nahi paate.

Pehli pipeline

`.gitea/workflows/ci.yaml` :

```
name: ci
on: [push]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: run tests
        run: python3 -c "assert 2 + 2 == 4; print('test passed')"

      - name: build image
        run: docker build -t ci-demo:${GITHUB_SHA::7} .

      - name: run it
        run: docker run --rm ci-demo:${GITHUB_SHA::7}
```

Push kiya, pipeline chali, image bani. Verify:

```
docker images | grep ci-demo
# ci-demo  98d3e6d  13.5MB

docker run --rm ci-demo:98d3e6d
# built on aarch64
```

Phase 3 — Kubernetes

Kya banaya

k3s (single node), kubectl, helm, k9s, aur ek 2-replica app jo internet pe live hai.

Kyun

Kubernetes containers ko **manage** karta hai: kitni copies chalni chahiye, ek mar jaye to nayi banao, config alag rakho, traffic route karo. Compose ek machine ke liye theek hai; k8s tab chahiye jab ye sab automatic chahiye.

k3s isliye chuna kyunki wo halka hai aur arm64 pe first-class support hai.

Ek asli decision

k3s default mein **Traefik** ingress laata hai, jo ports 80 aur 443 maangta hai. Wo ports pehle se **nginx** ke paas the (aur usi pe Gitea aur TLS cert chal rahe the).

Do raaste the:

- **A:** Traefik disable, nginx front door rahe, k8s services NodePort pe, nginx unke aage proxy kare
- **B:** nginx band, Traefik front door bane

A chuna, kyunki chalta hua setup nahi todna tha. Seekhne mein koi kami nahi hui — Ingress objects phir bhi banaye ja sakte hain.

Commands

```
# k3s, traefik ke bina
curl -sfl https://get.k3s.io | \
  INSTALL_K3S_EXEC="--disable traefik --write-kubeconfig-mode 644" sh -

# kubeconfig apne user ke liye
mkdir -p ~/.kube
sudo cp /etc/rancher/k3s/k3s.yaml ~/.kube/config
sudo chown $USER:$USER ~/.kube/config
echo 'export KUBECONFIG=$HOME/.kube/config' >> ~/.bashrc

# helm
curl -fsSL https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | sudo bash

# k9s (arm64)
K9SVER=$(curl -s https://api.github.com/repos/derailed/k9s/releases/latest \
  | grep -o '"tag_name": *"[^"]*"' | cut -d'"' -f4)
curl -fsSL "https://github.com/derailed/k9s/releases/download/${K9SVER}/k9s_Linux_arm64.tar.gz" \
  | tar -xz -C /tmp k9s
sudo install -m 755 /tmp/k9s /usr/local/bin/k9s
```

Ek galti jo hui

Pehli baar `kubect1 apply -f .` chalaya to ye error aaya:

```
Error from server (NotFound): namespaces "demo" not found
```

Wajah: `kubectl apply -f <directory>` files ko **alphabetical order** mein lagata hai. `configmap.yaml` aur `deployment.yaml`, `namespace.yaml` se pehle aate hain. To namespace bane bina hi uske andar cheezein banane ki koshish hui.

Fix: files ko number kar do taaki alphabetical order hi dependency order ban jaye:

```
00-namespace.yaml
10-configmap.yaml
20-deployment.yaml
30-service.yaml
```

Ye standard practice hai. ArgoCD isi problem ko "sync waves" se solve karta hai.

Manifests

```
# 20-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo
  namespace: demo
spec:
  replicas: 2
  selector:
    matchLabels: { app: demo }
  template:
    metadata:
      labels: { app: demo }
    spec:
      containers:
        - name: web
          image: nginx:alpine
          ports: [{ containerPort: 80 }]
          volumeMounts:
            - { name: page, mountPath: /usr/share/nginx/html }
          readinessProbe:
            httpGet: { path: /, port: 80 }
          resources:
            requests: { cpu: 10m, memory: 16Mi }
            limits: { cpu: 200m, memory: 128Mi }
      volumes:
        - name: page
          configMap: { name: demo-page }
```

Isme teen cheezein dhyan se dekho:

- `replicas: 2` — do copies. Ek mar jaye to k8s nayi bana dega.
- `readinessProbe` — pod tabhi traffic lega jab wo sach mein ready ho.
- `resources` — `requests` scheduling ke liye, `limits` cap ke liye. Iske bina ek app poori machine kha sakta hai.

Deploy aur expose:

```
kubectl apply -f ~/devops/k8s/demo
kubectl -n demo rollout status deploy/demo
kubectl -n demo get pods -o wide
```

nginx site jo NodePort ke aage lagti hai:

```
server {
    listen 80;
    server_name k8s.ridgeserve.duckdns.org;
    location / {
        proxy_pass http://127.0.0.1:30080;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Phase 4 — GitOps (ArgoCD)

Kya banaya

ArgoCD, aur ek git repo jise wo watch karta hai. Ab cluster **git se** chalta hai.

Kyun

Ab tak humne `kubectl apply` haath se chalaya. Problem: kisi ko nahi pata cluster mein sach mein kya hai. Kisi ne raat ko kuch badal diya to kahin record nahi.

GitOps ka usool: **git hi sach hai**. Cluster ko git se match karna hai. Aap git badalte ho, ArgoCD cluster ko follow karata hai. Koi manually kuch badle to ArgoCD use wapas theek kar deta hai.

Install

```
kubectl create namespace argocd
kubectl apply -n argocd \
  -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
```

Ek error aayega:

```
The CustomResourceDefinition "applicationsets.argoproj.io" is invalid:
metadata.annotations: Too long: may not be more than 262144 bytes
```

Ye known issue hai. `kubectl apply` poori config ko annotation mein rakhta hai, aur ye CRD bahut bada hai. Fix — server-side apply:

```
kubectl apply --server-side --force-conflicts -n argocd \
  -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
```

nginx ke peeche chalane ke liye ArgoCD ka apna TLS band karo:

```
kubectl -n argocd patch configmap argocd-cmd-params-cm --type merge \
  -p '{"data":{"server.insecure":"true"}}'
kubectl -n argocd rollout restart deploy/argocd-server
```

Admin password:

```
kubectl -n argocd get secret argocd-initial-admin-secret \
  -o jsonpath='{.data.password}' | base64 -d
```

Application

```

apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: hello
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://git.ridgeserve.duckdns.org/abhishek/gitops.git
    targetRevision: main
    path: apps/hello
  destination:
    server: https://kubernetes.default.svc
  syncPolicy:
    automated:
      prune: true      # git se hataya to cluster se bhi hatao
      selfHeal: true   # manual change ko wapas theek karo
    syncOptions:
      - CreateNamespace=true

```

`prune` aur `selfHeal` hi asli GitOps banate hain. Inke bina ArgoCD sirf ek deploy tool hai.

Dono behaviours test kiye

Test 1 — git badlo, cluster follow kare:

```

sed -i 's|v1|v2|' apps/hello/10-configmap.yaml
sed -i 's|replicas: 2|replicas: 3|' apps/hello/20-deployment.yaml
git commit -am "bump to v2, scale to 3" && git push

```

Bina kuch kiye pods 3 ho gaye aur page v2 dikhane laga.

Test 2 — haath se todo, ArgoCD theek kare:

```
kubectl -n hello scale deploy hello --replicas=1
```

Kuch second baad:

```

spec.replicas=1
spec.replicas=3    <- argocd ne wapas kar diya

```

Yahi `selfHeal` hai. Cluster mein manual change tik nahi sakta.

Ek chhoti trick

ArgoCD default mein har 3 minute mein git check karta hai. Turant sync chahiye to:

```

kubectl -n argocd patch app hello --type merge \
  -p '{"metadata":{"annotations":{"argocd.argoproj.io/refresh":"hard"}}}'

```


Phase 5 — Observability

Kya banaya

Prometheus (metrics), Grafana (dashboards), Loki (logs), Promtail (log shipper), Alertmanager.

Kyun

Ab tak humne cheezein banayi. Ab sawal ye hai: **chal kya rahi hai?** CPU kitna use ho raha hai, kaunsa pod restart ho raha hai, kal raat 3 baje kya hua tha.

Do alag cheezein hain:

- **Metrics** (Prometheus) — numbers, samay ke saath. “CPU 40% tha.”
- **Logs** (Loki) — text lines. “Us waqt ye error aaya tha.”

Dono chahiye. Metrics batate hain ki kuch galat hai, logs batate hain ki kya.

Install

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo add grafana https://grafana.github.io/helm-charts
helm repo update

helm upgrade --install monitoring prometheus-community/kube-prometheus-stack \
  -n monitoring --create-namespace \
  -f ~/devops/k8s/monitoring/values.yaml --timeout 15m --wait
```

Values file mein chhote node ke liye tuning:

```
grafana:
  adminPassword: "<password>"
  service:
    type: NodePort
    nodePort: 30083
  grafana.ini:
    server:
      root_url: "https://grafana.ridgeserve.duckdns.org"

prometheus:
  prometheusSpec:
    retention: 7d          # lab ko mahinon ka data nahi chahiye
    resources:
      requests: { cpu: 100m, memory: 512Mi }
      limits:   { cpu: 1000m, memory: 2Gi }
    storageSpec:
      volumeClaimTemplate:
        spec:
          accessModes: ["ReadWriteOnce"]
          resources:
            requests:
              storage: 8Gi
```

Loki single-binary mode mein (ek chhote node ke liye yahi sahi shape hai):

```
deploymentMode: SingleBinary
loki:
  auth_enabled: false
  commonConfig:
    replication_factor: 1
  storage:
    type: filesystem
  limits_config:
    retention_period: 168h
singleBinary:
  replicas: 1
  persistence: { enabled: true, size: 5Gi }
read: { replicas: 0 }
write: { replicas: 0 }
backend: { replicas: 0 }
```

Grafana mein Loki ko datasource banane ka saaf tareeka — ek labelled ConfigMap:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: loki-datasource
  namespace: monitoring
  labels:
    grafana_datasource: "1"      # grafana ka sidecar is label ko dhundhta hai
data:
  loki.yaml: |
    apiVersion: 1
    datasources:
      - name: Loki
        type: loki
        url: http://loki:3100
```

Grafana ko restart nahi karna padta, sidecar khud utha leta hai.

Verify kaise kiya

Sirf “pods Running hain” kaafi nahi. Asli data check kiya:

```
kubectl -n monitoring run probe --rm -i --restart=Never \
  --image=curlimages/curl:latest -- sh -c '
  curl -s "http://monitoring-kube-prometheus-prometheus:9090/api/v1/targets?state=active" \
  | grep -o "\"health\": \"[a-z]*\"" | sort | uniq -c
  curl -s "http://loki:3100/loki/api/v1/label/namespace/values"
```

Result:

```
13 "health": "up"
{"data": ["argocd", "demo", "hello", "kube-system", "monitoring"]}
```

13 targets scrape ho rahe hain, aur Loki paanch namespaces ke logs le raha hai.

Note: Prometheus ka container distroless hai, usme `wget` / `curl` nahi hai. Isliye alag se curl pod chalana pada. Ye aaj kal aam hai — chhota image = kam attack surface.

Grafana mein 29 dashboards pehle se aa jaate hain (Kubernetes compute, node exporter, CoreDNS, etcd, API server).

Phase 6 — Infrastructure as Code

Kya banaya

- **OpenTofu** — asli OCI cloud resources code se
- **Ansible** — server ki configuration code se

Kyun

Ab tak humne server ke **andar** ki cheezein code mein rakhi. Par server khud? VCN, firewall rules, volumes? Wo abhi bhi console se click karke bane the.

IaC ka matlab: cloud infrastructure bhi code mein likho, review karo, version karo.

OpenTofu

```
curl -fsSL https://get.opentofu.org/install-opentofu.sh -o /tmp/t.sh
chmod +x /tmp/t.sh && sudo /tmp/t.sh --install-method deb
```

Provider wahi API key use karta hai jo pehle se `~/.oci/config` mein hai:

```
terraform {
  required_providers {
    oci = { source = "oracle/oci", version = "~> 7.0" }
  }
}

provider "oci" {
  config_file_profile = "DEFAULT"
}
```

Do tarah ki cheezein:

```
# data source = sirf padhta hai, kabhi kuch badalta nahi
data "oci_core_instance" "server" {
  instance_id = var.instance_ocid
}

# resource = manage karta hai
resource "oci_objectstorage_bucket" "backups" {
  compartment_id = var.tenancy_ocid
  namespace      = data.oci_objectstorage_namespace.ns.namespace
  name           = "ridgeserve-backups"
  access_type    = "NoPublicAccess"
  versioning     = "Enabled"

  freeform_tags = {
    managed_by = "opentofu"
  }
}
```

Workflow:

```
tofu init      # provider download
tofu plan      # kya badlega - APPLY SE PEHLE HAMESHA DEKHO
tofu apply     # kar do
```

`plan` ka output asli account se aaya:

```
server_name = "abhishekdevelopment"
server_shape = "VM.Standard.A1.Flex"
server_state = "RUNNING"
Plan: 1 to add, 0 to change, 0 to destroy.
```

Apply ke baad bucket sach mein ban gaya, aur OCI CLI se independently verify kiya:

```
oci os bucket list --compartment-id <tenancy> --output table
# ridgeserve-backups | 2026-08-26T13:58:26
```

Idempotence test — dobara chalao, kuch nahi hona chahiye:

```
tofu apply -auto-approve
# No changes. Your infrastructure matches the configuration.
```

Yahi IaC ka poora point hai. Jitni baar chalao, result same.

State file

`terraform.tfstate` ek bahut important file hai — OpenTofu isi mein yaad rakhta hai ki usne kya-kya banaya. **Isme secrets bhi aa jaate hain**, isliye ye `.gitignore` mein hai. Team mein ye remote backend (S3, OCI Object Storage) mein rakhi jaati hai.

Ansible

```
sudo apt-get install -y ansible
```

Playbook server ko **describe** karta hai, commands nahi likhta:

```

- name: Configure the lab server
  hosts: lab
  become: true
  vars:
    base_packages: [btop, jq, tree, ncdu, ripgrep, git, tmux, unzip]

  tasks:
    - name: Base packages are installed
      ansible.builtin.apt:
        name: "{{ base_packages }}"
        state: present
        update_cache: true
        cache_valid_time: 3600

    - name: Swap file exists
      ansible.builtin.command: "fallocate -l 4G /swapfile"
      args:
        creates: /swapfile          # already ho to skip

    - name: Services that must be running
      ansible.builtin.systemd:
        name: "{{ item }}"
        state: started
        enabled: true
      loop: [nginx, docker, postgresql, fail2ban, k3s]

```

Task ke naam dhyan se dekho: **"Base packages are installed"**, "install packages" nahi. Ansible mein aap **haalat** likhte ho, kaam nahi. Wo khud dekhta hai kya karna hai.

```

cd ~/devops/ansible
ansible-playbook site.yml

```

Dono baar:

```
localhost : ok=9  changed=0  unreachable=0  failed=0
```

`changed=0` matlab server pehle se sahi haalat mein hai. Ab wo haalat code mein likhi hai — server dobara banana pade to ye playbook chala do.

Phase 7 — Security

Kya banaya

- **Vault** — secrets ka proper ghar
- **Trivy** — image vulnerability scanner, CI pipeline mein gate ki tarah

Vault kyun

Abhi tak passwords `.env` files mein hain jo gitignored hain. Kaam chal jata hai, par do problem hain: repo se server rebuild nahi ho sakta (secrets missing rahenge), aur kisi ko access dena ho to poori file deni padti hai.

Vault secrets ko encrypted rakhta hai, access control deta hai, aur har access ka log rakhta hai.

Install

```
helm repo add hashicorp https://helm.releases.hashicorp.com
helm upgrade --install vault hashicorp/vault -n vault --create-namespace \
-f ~/devops/k8s/vault/values.yaml
```

Values:

```
injector:
  enabled: false
server:
  standalone:
    enabled: true      # dev mode nahi - asli seal/unseal lifecycle
  dataStorage:
    enabled: true
    size: 1Gi
ui:
  enabled: true
```

`standalone` **chuna**, `dev` **mode nahi**. Dev mode automatically unsealed hota hai aur restart pe data chala jata hai. Standalone mein asli lifecycle milta hai, jo seekhne ke liye zaroori hai.

Seal aur unseal

Install ke baad pod `0/1 Running` dikhega. **Ye galti nahi hai**. Vault **sealed** hota hai — data encrypted hai aur bina unseal key ke Vault khud bhi use nahi padh sakta.

```
# Initialise - unseal key aur root token yahin milte hain, dobara kabhi nahi
kubectl -n vault exec vault-0 -- vault operator init \
  -key-shares=1 -key-threshold=1 -format=json > init.json

# Unseal
kubectl -n vault exec vault-0 -- vault operator unseal <unseal-key>
```

```
Initialized      true
Sealed           false
```

Ab pod `1/1 Ready` hai.

`-key-shares=1` **sirf lab ke liye hai**. Production mein key 5 logon mein baanti jaati hai aur unseal ke liye kam se kam 3 chahiye hote hain, taaki koi akela Vault na khol sake. Isko Shamir's Secret Sharing kehte hain.

Yaad rakho: Vault restart hone pe **dobara sealed** ho jata hai aur manually unseal karna padta hai. Production mein iske liye auto-unseal (cloud KMS) use hota hai.

Secret store karna

```
export VAULT_TOKEN=<root-token>
vault secrets enable -path=lab kv-v2
vault kv put lab/postgres user=ecomersehub host=127.0.0.1
vault kv get lab/postgres
```

`kv-v2` versioned hai — purani values bhi rehti hain, galti se overwrite ho jaye to wapas la sakte ho.

Trivy

```
curl -sfl https://raw.githubusercontent.com/aquasecurity/trivy/main/contrib/install.sh \
| sudo sh -s -- -b /usr/local/bin

trivy image --scanners vuln --severity HIGH,CRITICAL nginx:alpine
```

Purani image pe chalao to farak dikh jata hai:

```
node:18-bullseye Total: 1839 (HIGH: 1622, CRITICAL: 217)
```

CI mein security gate

Pipeline mein ek step add kiya jo vulnerable image ko rok deta hai:

```
- name: scan image for vulnerabilities
  run: |
    docker run --rm \
      -v /var/run/docker.sock:/var/run/docker.sock \
      aquasec/trivy:latest image \
      --quiet --scanners vuln --severity HIGH,CRITICAL \
      --exit-code 1 \
      ci-demo:${GITHUB_SHA::7}
```

`--exit-code 1` hi asli gate hai. Vulnerability mili to Trivy 1 return karta hai, step fail hota hai, build ruk jati hai.

Aur wo sach mein ruki. Push karte hi pipeline fail ho gayi, kyunki `alpine:latest` mein ek HIGH CVE tha (`CVE-2026-14456`, openssl).

Fix ye **nahi** tha ki severity threshold neeche kar do. Fix ye tha ki base image patch karo:

```
FROM alpine:latest
RUN apk --no-cache upgrade      # yahi fix hai
RUN echo "built on $(uname -m)" > /where
```

Poori kahani:

```
run 1: success   koi gate nahi tha
run 2: failure   gate laga, vulnerable image block hui
run 3: success   base image patch kiya
```

Yahi wo loop hai jo asli production mein chalta hai.

Alag se: PostgreSQL for ecomersehub

Ek doosre project ko database chahiye tha. Local `postgresql-x64-18` har baar elevated shell maang raha tha, to server wala use karne ka faisla hua.

Database aur ek **non-superuser** role banaya:

```
CREATE ROLE ecomersehub LOGIN PASSWORD '<random>';
CREATE DATABASE ecomersehub OWNER ecomersehub;
GRANT ALL ON SCHEMA public TO ecomersehub;
```

Verify:

```
select rolname, rolsuper from pg_roles where rolname='ecomersehub';
-- ecomersehub | f      <- superuser NAHI
```

Access: tunnel, public port nahi

Postgres ko internet pe kholne ka option tha, par **SSH tunnel chuna**:

```
ssh -L 5432:127.0.0.1:5432 oracle
```

Kyun: internet pe khula 5432 automated scanners ka pehla target hai. Tunnel already encrypted hai aur port band rehta hai.

Tunnel har baar khud kholna na pade, isliye Windows pe permanent bana diya:

- `~/.ssh/pg-tunnel.ps1` — reconnect loop, log rakhta hai
- `Startup\pg-tunnel.vbs` — login pe hidden window mein chalu

```
while ($true) {
    ssh -N -o ExitOnForwardFailure=yes -o ServerAliveInterval=30 `
        -L 5432:127.0.0.1:5432 oracle
    Start-Sleep -Seconds 5
}
```

Connection string:

```
postgresql://ecomersehub:<password>@127.0.0.1:5432/ecomersehub
```

`sslmode=disable` sahi hai kyunki tunnel khud encrypted hai.

Check karne ke commands:

```
Get-NetTCPConnection -LocalPort 5432 -State Listen
Get-Content ~\.ssh\pg-tunnel.log -Tail 20
```

Agar wo app kabhi hosted jagah pe deploy hua (Vercel, koi server), tab tunnel kaam nahi karega. Tab port public karna padega, TLS ke saath.

Cheat Sheet

Roz ka

```
ssh oracle          # server
k9s                 # cluster browse karo
lazydocker          # containers browse karo
bttop               # system resources
```

Containers

```
docker compose up -d
docker compose ps
docker compose logs -f <service>
docker compose down -v
docker images
docker system prune -a          # jagah khali karo
```

Kubernetes

```
kubectl get pods -A          # sab kuch
kubectl -n <ns> get pods -o wide
kubectl -n <ns> logs -f <pod>
kubectl -n <ns> describe pod <pod>    # kyun nahi chal raha
kubectl -n <ns> rollout status deploy/<name>
kubectl -n <ns> rollout restart deploy/<name>
kubectl apply -f ~/devops/k8s/demo
kubectl top nodes
kubectl top pods -A
```

ArgoCD

```
kubectl -n argocd get apps
kubectl -n argocd get app hello -o jsonpath='{.status.sync.status}'
kubectl -n argocd patch app hello --type merge \
  -p '{"metadata":{"annotations":{"argocd.argoproj.io/refresh":"hard"}}}'
```

Vault

```
kubectl -n vault exec -it vault-0 -- sh
export VAULT_TOKEN=<root-token>
vault status
vault kv put lab/<name> key=value
vault kv get lab/<name>
# restart ke baad:
kubectl -n vault exec vault-0 -- vault operator unseal <key>
```

nginx aur certificates

```
~/devops/scripts/deploy-nginx.sh      # repo se deploy
sudo nginx -t                         # config check
sudo certbot certificates              # kab expire
sudo certbot renew --dry-run          # renewal test
systemctl is-active certbot.timer
```

IaC

```
cd ~/devops/terraform
tofu plan                            # apply se pehle hamesha
tofu apply
tofu destroy                          # sab hata do (dhyan se)

cd ~/devops/ansible
ansible-playbook site.yml --check     # dry run
ansible-playbook site.yml
```

Security

```
trivy image --severity HIGH,CRITICAL <image>
sudo fail2ban-client status sshd
sudo iptables -L INPUT -n --line-numbers
```

Debugging

```
sudo journalctl -u <service> -f      # live logs
sudo journalctl -u <service> -b -1   # pichle boot ke logs
sudo ss -tlnp                        # kaun kaunsa port sun raha hai
curl -s -o /dev/null -w '%{http_code}\n' <url>
```

Gotchas ki list

Ye wo cheezein hain jo yaad rakhne layak hain. Har ek humein sach mein mili.

1. Port kholne ke do layers hote hain

OCI mein port kholna do jagah karna padta hai:

- **OCI Security List** — cloud level
- **iptables** — host level

Ek bhi chhoot gaya to traffic gayab. Pehchanne ka tareeka:

Symptom	Kahan problem hai
Timeout	Security List block kar rahi hai
Connection refused	Host tak pahuncha, par koi listen nahi kar raha

2. ssh.socket se port mat badalna

`ListenStream` override karne se IPv4 listener gayab ho jata hai, aur `ss` phir bhi theek dikhata hai. Ek ghanta aur boot volume rescue laga tha.

Port badalna hi ho to `ssh.socket` ke bajaye plain `ssh.service` use karo, jahan `sshd_config` ka `Port` normal kaam karta hai.

3. kubectl apply directory ko alphabetically padhta hai

Files number karo (`00-`, `10-`, `20-`) taaki namespace pehle bane.

4. k3s ko Docker ki images nahi dikhtin

Docker aur containerd ke image stores alag hain. Jo tag `docker run` se chalta hai, wo pod mein pull fail karega. Registry chahiye jise dono dekh sakein.

5. Boot volume ka `AVAILABLE` ka matlab detached nahi

Attachment alag resource hai. Sahi check:

```
oci compute boot-volume-attachment list \
  --compartment-id <c> --availability-domain <ad>
```

6. OCIDs 60 characters ke hote hain (`..` ke baad)

Screenshot se copy karte waqt `a` ki ginti galat ho jaati hai. Length se verify karo.

7. Distroless containers mein debugging tools nahi hote

Prometheus jaise pods mein `curl` / `wget` nahi milega. Alag se curl pod chalao:

```
kubectl run probe --rm -i --restart=Never --image=curlimages/curl:latest -- sh -c '...'
```

8. Gitea ko INSTALL_LOCK chahiye

Bina `GITEA__security__INSTALL_LOCK: "true"` ke `app.ini` banti hi nahi aur admin CLI kaam nahi karti.

9. act_runner ko network batana padta hai

`runner-config.yaml` mein `container.network: gitea_default` chahiye, warna job containers Gitea ko resolve nahi kar paate.

10. Bade CRDs ko server-side apply chahiye

`kubectl apply` config ko annotation mein rakhta hai, jiski limit 256 KB hai. ArgoCD ka ApplicationSet CRD usse bada hai. `--server-side` use karo.

Aaj ka final state

Live services

Service	URL
Lab dashboard	https://ridgeserve.duckdns.orgSMARTPANTS_PRESERVED_1039SMARTPANTS_PRESERVED_1040
Gitea	https://git.ridgeserve.duckdns.orgSMARTPANTS_PRESERVED_1047SMARTPANTS_PRESERVED_1048
Kubernetes demo	https://k8s.ridgeserve.duckdns.orgSMARTPANTS_PRESERVED_1055SMARTPANTS_PRESERVED_1056
ArgoCD	https://argocd.ridgeserve.duckdns.orgSMARTPANTS_PRESERVED_1063SMARTPANTS_PRESERVED_1064
Grafana	https://grafana.ridgeserve.duckdns.orgSMARTPANTS_PRESERVED_1071SMARTPANTS_PRESERVED_1072

Sab pe asli Let's Encrypt certificate, auto-renewal ke saath.

Numbers

```

Kubernetes pods    24 / 24 running
Docker containers  7
RAM                3.6 GB / 23 GB used
Disk               23 GB / 193 GB used
CPU                8%
Git commits        7 (ek per phase)
Secrets in git     0

```

Git history

```

d7001d6 phase 7: vault with real seal lifecycle, trivy gate in ci
d91b76a phase 6: opentofu managing real OCI, ansible describing the server
db5a618 phase 5: prometheus, grafana, loki, promtail
84c469d phase 4: argocd gitops, verified sync + self-heal
3ef830d phase 3: k3s cluster, demo workload, tls subdomain
26d1d6f phase 2 complete: gitea actions runner + verified pipeline
db32baa phase 0+1: foundation, languages, containers

```

Ek zaroori deadline

Oracle ka Free Trial **23 August 2026** ko shuru hua tha, 30 din ka hai.

Kab	Kya hoga
~22 September	Trial khatam, 30 din ka grace period shuru
~22 October	Grace khatam, free limit se upar ka sab delete

Aur ek problem: **July 2026 se Always Free ka Ampere allowance aadha ho gaya hai.**

Account	Free Ampere
Always Free (bina card)	2 OCPU / 12 GB
Paid / Pay-As-You-Go	4 OCPU / 24 GB

Ye server 4 OCPU / 24 GB ka hai — Always Free se **do guna**. Matlab ye “paid resource” gina jayega aur grace period ke baad delete ho jayega.

Card apne aap charge nahi hoga. Oracle tabhi bill karta hai jab aap khud “Upgrade” dabao. Risk paisa nahi, server khona hai.

Do raaste:

1. **Pay-As-You-Go pe upgrade** — 4 OCPU / 24 GB free hi rahega (paid tier ka free allowance itna hi hai), bill 0 rahega. Budget Alert lagana zaroori hai.
2. **2 OCPU / 12 GB pe chhota karo** — sach mein hamesha free, par machine aadhi.

22 October se pehle faisla karna hai.

Aage kya

Jo abhi bhi kar sakte ho:

Registry pipeline पूरी करो. Abhi CI image banati hai par k8s use pull nahi kar sakta (gotcha #4). Gitea ki built-in container registry use karke: CI push kare, k8s `imagePullSecret` se pull kare. Tab poora loop banega — code push se lekar cluster mein chalne tak.

Alerts lagao. Prometheus aur Alertmanager already hain, par abhi koi alert rule nahi hai. Ek achha pehla rule: pod restart loop mein ho to batao.

Backup. OpenTofu se `ridgeserve-backups` bucket ban chuka hai par khaali hai. Ek cron banao jo Gitea ka data aur Postgres dumps wahan bheje.

Vault ko sach mein use karo. Abhi secrets `.env` files mein hain. Unhe Vault mein daalo aur External Secrets Operator se k8s tak pahunchao.

Ingress controller. Traefik disable kiya tha, isliye Ingress objects abhi kaam nahi karte. `ingress-nginx` cluster mein daal ke uske aage host nginx rakho — tab Ingress bhi seekh jaoge.

Ye document 26 August 2026 ke kaam ka record hai. Sab kuch `~/devops` repo mein hai server pe.