Christian Dussol

The complete

Kubernetes command reference

Here are the kubectl commands you actually use daily

- Organized by role:

  - 👨‍💻 Developer
  - ⚙️ Administrator
  - 🔒 Security
  - 🔧 Troubleshooting
  - 📊 Monitoring

Developer Essentials

1. Quick pod creation

```
kubectl run nginx --image=nginx
```

2. Generate YAML template

```
export do="--dry-run=client -o yaml"
kubectl run nginx --image=nginx $do > pod.yaml
```

3. View logs in real-time

```
kubectl logs my-pod -f
```

4. Scale deployment

```
kubectl scale deployment nginx --replicas=3
```

5. Port-forward for testing

```
kubectl port-forward svc/nginx 8080:80
```

Admin Power Commands

1. Drain node for maintenance

```
kubectl drain node-1 --ignore-daemonsets
```

2. Create RBAC role

```
kubectl create role pod-reader \
--verb=get,list --resource=pods
```

3. Set resource quota

```
kubectl create quota prod-quota \
--hard=cpu=10,memory=20Gi
```

4. Switch namespace context

```
kubectl config set-context --current \
--namespace=production
```

5. View cluster information

```
kubectl cluster-info
kubectl top nodes
```

🔒 Security Must-Haves

1. Apply Pod Security Standards

```
kubectl label namespace production \
  pod-security.kubernetes.io/enforce=restricted
```

2. Create secure secret

```
kubectl create secret generic db-pass \
  --from-literal=password=$(openssl rand -base64 32)
```

3. Decode secret value

```
kubectl get secret db-pass \
  -o jsonpath="{.data.password}" | base64 -d
```

4. Deny-all network policy

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-all
spec:
  podSelector: {}
  policyTypes: [Ingress, Egress]
```

Troubleshooting Survival Kit

1. Check detailed status

```
kubectl describe pod my-pod
```

2. View current and previous logs

```
kubectl logs my-pod
kubectl logs my-pod --previous
```

3. Check recent events

```
kubectl get events --sort-by='.lastTimestamp'
```

4. Launch temporary debug pod

```
kubectl run debug --rm -it --image=busybox -- sh
```

💡 Pro tip: Always check events first!
They often reveal the root cause instantly.

📊 Monitoring Essentials

1. Check resource usage

```
kubectl top pods --sort-by=memory
kubectl top nodes
```

2. Follow logs with label selector

```
kubectl logs -l app=nginx -f
# Note: Use 'stern' for multiple pods
```

3. Check health probes status

```
kubectl describe pod my-pod | grep -A 5 "Readiness"
```

4. Find pods without resource limits

```
kubectl get pods -o json | \
  jq '.items[] | select(.spec.containers[].resources.limits == null)'
```

⚠️ Pods without limits = potential runaway resources

Pro Tips that save hours

1. Essential aliases

```
alias k=kubectl
alias kgp='kubectl get pods'
alias kdes='kubectl describe'
export do="--dry-run=client -o yaml"
```

2. Enable autocompletion (must-have!)

```
source <(kubectl completion bash)
complete -F __start_kubectl k
```

3. View current context

```
kubectl config view --minify
```

4. Use explain as built-in doc

```
kubectl explain pod.spec.containers
kubectl explain deployment.spec.strategy
```

Key Takeaways

🎯Organize commands by workflow, not alphabetically

⚡Use imperative commands + dry-run for maximum speed

🔐 Automate security enforcement with Kyverno policies

📊 Always monitor resource usage before issues happen

🛠️ Practice troubleshooting in dev, not production

💡 The goal is not memorization, it's having the right command at the right moment

🚀 Speed in Kubernetes = Organization

Get the Complete Cheat Sheet

This carousel = Top 30 commands

Full cheat sheet includes:

✅ 100+ production-ready commands
✅ All sections with detailed examples
✅ Security best practices
✅ Troubleshooting workflows
✅ Kyverno policy templates
✅ Complete monitoring guide

💻 GitHub Markdown: https://bit.ly/46EQ87s