CDW Documentation

kubectl Cheat Sheet

kubectl Cheat Sheet

CLUSTER LIFECYCLE COMMANDS

- **minikube start**
  - Starts a local Kubernetes cluster
  - Automatically configures kubectl to point to the cluster
- **minikube stop**
  - Stops the cluster without deleting it
- **minikube delete**
  - Completely deletes the Kubernetes cluster
  - Removes all namespaces, system pods, and workloads
  - This is the correct way to fully reset Kubernetes

BASIC CLUSTER VERIFICATION

- **kubectl get nodes**
  - Confirms kubectl can connect to the cluster
  - Verifies node readiness
- **kubectl get pods**
  - Lists pods in the current namespace
- **kubectl get pods -A**
  - Lists pods across all namespaces
  - Includes system namespaces like kube-system

RESOURCE INSPECTION

- **kubectl describe pod**
  - Shows detailed pod configuration and runtime state
  - Displays events (very important for debugging)
  - First command to use when something is broken

WINDOWS CMD SHORTCUT

- **doskey k=kubectl $***
  - Creates a temporary shortcut so “k” can be used instead of “kubectl”
  - Only lasts for the current Command Prompt session
  - Windows CMD does not support Linux-style aliases

OUTPUT FORMATTING

- **kubectl get pods**
  - Default, human-readable summary output
- **kubectl get pods -o wide**
  - Adds pod IP and node name
- **kubectl get pod -o yaml**
  - Shows the full Kubernetes object definition
  - Useful for learning resource structure
- **kubectl get pods -o name**
  - Outputs only resource names
  - Useful for scripting
- **kubectl get pods -o custom-columns=NAME:.metadata.name,PHASE:.status.phase**
  - Displays selected fields in custom columns
- **kubectl get pods -o jsonpath="{.items[*].metadata.name}"**
  - Extracts specific values from Kubernetes API output
  - Foundation for automation and scripting

IMPERATIVE VS DECLARATIVE

- Imperative commands
  - Tell Kubernetes what to do immediately
  - Fast but not repeatable or version-controlled
- Examples:
  - Creating deployments
  - Scaling replicas directly
- Declarative workflow
  - Define desired state in YAML
  - Kubernetes reconciles actual state to match
- Workflow:
  - Export YAML from a resource
  - Edit YAML using Notepad or VS Code
  - Apply YAML with kubectl apply
- Declarative is preferred in production environments.

DEBUGGING PODS AND CONTAINERS

Standard debugging flow:
1. **kubectl get pods** → Identify failing or pending pods
2. **kubectl describe pod** → Look at events and status
3. **kubectl logs** → View application logs
4. **kubectl logs** –previous → View logs from a crashed container
5. **kubectl exec -it – sh** → Access a running container shell
6. **kubectl debug -it –image=busybox –target=[PODNAME]** → Attach an ephemeral debug container
Important distinction:

- Pod name and container name are different
- Container names are found via describe or JSONPath

NAMESPACES AND CONTEXTS

- **kubectl create namespace sandbox**
  - Creates an isolated workspace
- **kubectl get pods -n sandbox**
  - Lists resources in a specific namespace
- **kubectl config set-context –current –namespace=sandbox**
  - Sets default namespace for the current context
- **kubectl config current-context**
  - Shows which cluster/context kubectl is using
- Namespaces are the safest unit for testing and cleanup.

CLEANUP COMMANDS

- **kubectl delete deployment demo**
  - Removes user-created workloads
- **kubectl delete namespace sandbox**
  - Deletes everything inside that namespace
- **minikube delete**
  - Fully resets Kubernetes
  - Only supported way to remove kube-system indirectly

KUBE-SYSTEM NAMESPACE

- Created automatically by Kubernetes
- Contains control plane and core system components
- Cannot and should not be deleted
- A clean cluster always includes kube-system pods
- If you want kube-system gone, delete the entire cluster instead.

WINDOWS-SPECIFIC NOTES

- Windows CMD does not support alias; use doskey
- YAML editing is done with external editors (Notepad, VS Code)
- JSONPath requires double quotes in CMD
- Many Kubernetes tutorials assume Linux shells and need adjustment

COMMON MISTAKES AND GOTCHAS

- Thinking kubectl installs or runs Kubernetes
  - kubectl is only a client

- Expecting an empty cluster after cleanup
  - System pods in kube-system always exist
- Trying to delete kube-system
  - This is intentionally forbidden
- Forgetting namespaces when resources seem missing
  - Default namespace may be empty
- Confusing pod names with container names
  - Multi-container pods require explicit container targeting
- Using imperative commands for long-term management
  - Declarative YAML is the production standard
- Assuming Linux shell commands work on Windows CMD
  - Alias, quoting, and editors differ
- Editing live resources instead of source YAML
  - kubectl edit is convenient but not Git-friendly

MENTAL MODELS

- kube-system = operating system processes
- Your namespaces = application folders
- kubectl delete deployment = uninstalling an app
- minikube delete = reinstalling the OS

AI Home