DevOps Commands Cheat Sheet

Git • Docker • Kubernetes • Terraform • GitHub Actions • Linux — part of the Techwider DevOps Starter Kit

TECHWIDER
techwider.com

Git

COMMAND	WHAT IT DOES
`git init / git clone <url>`	Start a repo / copy one
`git status`	What changed?
`git add -p`	Stage changes piece by piece
`git commit -m "msg"`	Save a snapshot
`git switch -c feature/x`	Create and move to a branch
`git pull --rebase`	Get latest, keep history clean
`git push -u origin feature/x`	Publish a new branch
`git log --oneline --graph`	Compact history
`git diff --staged`	Review what you'll commit
`git stash / git stash pop`	Park work / bring it back
`git rebase -i HEAD~3`	Tidy the last 3 commits
`git restore <file>`	Discard local changes
`git revert <sha>`	Undo a pushed commit safely
`git reset --soft HEAD~1`	Undo last commit, keep changes
`git cherry-pick <sha>`	Copy one commit here
`git tag v1.0.0 && git push --tags`	Mark a release

Docker

COMMAND	WHAT IT DOES
`docker build -t app:1.0 .`	Build an image
`docker run -d -p 8080:8080 app:1.0`	Run in background, map port
`docker ps / docker ps -a`	Running / all containers
`docker logs -f <name>`	Follow logs
`docker exec -it <name> sh`	Shell inside a container
`docker stop / rm <name>`	Stop / remove container
`docker images / rmi <img>`	List / delete images
`docker tag app:1.0 user/app:1.0`	Name for a registry
`docker push user/app:1.0`	Upload to registry
`docker system prune -a`	Free disk space (careful)
`docker inspect <name>`	Full config as JSON
`docker stats`	Live CPU / memory

Docker Compose

COMMAND	WHAT IT DOES
`docker compose up -d --build`	Build and start everything
`docker compose ps`	Service status
`docker compose logs -f web`	Logs for one service
`docker compose exec web sh`	Shell in a service
`docker compose down -v`	Stop and delete volumes

Kubernetes (kubectl)

COMMAND	WHAT IT DOES
`kubectl config get-contexts`	Which clusters can I use?
`kubectl config use-context <ctx>`	Switch cluster
`kubectl get pods -A`	Pods in every namespace
`kubectl get all -n <ns>`	Everything in a namespace
`kubectl apply -f k8s/`	Create / update from files
`kubectl describe pod <pod>`	Events — why is it failing?
`kubectl logs -f <pod> [-c box]`	Follow pod logs
`kubectl logs <pod> --previous`	Logs from the crashed run
`kubectl exec -it <pod> -- sh`	Shell in a pod
`kubectl port-forward svc/app 8080:80`	Reach a service locally
`kubectl rollout status deploy/app`	Watch a deployment
`kubectl rollout undo deploy/app`	Roll back
`kubectl scale deploy/app --replicas=3`	Change replica count
`kubectl set image deploy/app web=img:2`	Update image
`kubectl top pods`	CPU / memory (metrics-server)
`kubectl get events --sort-by=.lastTimestamp`	Recent cluster events
`kubectl create secret generic s --from-literal=K=V`	Create a secret
`kubectl delete -f k8s/`	Remove what you applied

DevOps Commands Cheat Sheet

Git • Docker • Kubernetes • Terraform • GitHub Actions • Linux — part of the Techwider DevOps Starter Kit

TECHWIDER
techwider.com

Pod troubleshooting

COMMAND	WHAT IT DOES
`CrashLoopBackOff`	App exits — check logs --previous
`ImagePullBackOff`	Wrong image name / registry auth
`Pending`	No node fits — check requests, describe
`OOMKilled`	Raise memory limit or fix leak
`0/1 Ready`	Readiness probe failing — check path/port

Terraform

COMMAND	WHAT IT DOES
`terraform init`	Download providers, set up backend
`terraform fmt -recursive`	Format all files
`terraform validate`	Check syntax
`terraform plan -out tf.plan`	Preview changes
`terraform apply tf.plan`	Apply the saved plan
`terraform destroy`	Delete everything (careful)
`terraform output`	Show outputs
`terraform state list`	Resources being tracked
`terraform import <addr> <id>`	Adopt an existing resource
`terraform workspace new dev`	Separate state per env
`terraform apply -var-file=prod.tfvars`	Use a variables file

GitHub Actions — key syntax

COMMAND	WHAT IT DOES
`on: [push, pull_request]`	When the workflow runs
`on: workflow_dispatch`	Add a manual Run button
`runs-on: ubuntu-latest`	Runner machine
`needs: test`	Run after another job
`if: github.ref == 'refs/heads/main'`	Only on main
`uses: actions/checkout@v4`	Get the code
`${{ secrets.MY_KEY }}`	Use a repo secret
`strategy: matrix: python: [3.11, 3.12]`	Test several versions
`concurrency: ci-${{ github.ref }}`	Cancel duplicate runs

Linux essentials

COMMAND	WHAT IT DOES
`ls -lah / cd / pwd`	List, move, where am I
`tail -f /var/log/syslog`	Follow a log file
`grep -rn "error" .`	Search text in files
`df -h / du -sh *`	Disk free / folder sizes
`top / htop / free -h`	CPU and memory
`ss -tulpn`	Which ports are open
`systemctl status nginx`	Service status
`journalctl -u nginx -f`	Service logs
`chmod +x run.sh / chown`	Permissions / owner
`curl -I https://site.com`	Check a URL's headers
`ssh -i key.pem user@host`	Connect to a server
`scp file user@host:/path`	Copy a file to a server

DevOps golden rules

1. Everything in Git — code, config, infrastructure.
2. Never commit secrets. Use a secret manager or CI secrets.
3. Small, frequent changes with automated tests.
4. Same image from dev to prod; only config changes.
5. Health checks, resource limits and logs on every service.
6. Always **plan** before **apply**, and destroy practice infra.