
DevOps Shack

500 Essential DevOps Commands

Linux Commands

1. `uname -a` – Show system information
2. `uptime` – Show system uptime
3. `hostname` – Show hostname
4. `whoami` – Show current user
5. `ls -lah` – List directory contents with permissions
6. `pwd` – Show current directory
7. `cd /path` – Change directory
8. `mkdir -p /path` – Create a directory (with parent if needed)
9. `rm -rf /path` – Remove directory recursively
10. `cp -r source target` – Copy files/directories recursively



-
11. `mv oldname newname` – Move/Rename file
 12. `ps aux` – Show running processes
 13. `top` – Show real-time system performance
 14. `kill -9 PID` – Kill a process forcefully
 15. `pkill -f process_name` – Kill process by name
 16. `df -h` – Show disk space usage
 17. `du -sh /path` – Show directory size
 18. `netstat -tulnp` – Show listening ports
 19. `ss -tulnp` – Show listening ports (alternative)
 20. `who` – Show logged-in users
 21. `adduser username` – Create a new user
 22. `passwd username` – Change user password
 23. `usermod -aG group user` – Add user to a group
 24. `grep "text" file.txt` – Search for a string in a file
 25. `find /path -name "*.log"` – Find files matching pattern

-
26. `chmod 755 file.sh` – Change file permissions
 27. `chown user:group file` – Change file owner
 28. `tar -czf archive.tar.gz directory/` – Compress files
 29. `tar -xzf archive.tar.gz` – Extract compressed file
 30. `cat file.txt` – View file contents
 31. `tail -f /var/log/syslog` – View real-time log updates
 32. `echo "text" >> file.txt` – Append text to file
 33. `history` – Show command history
 34. `clear` – Clear terminal screen
 35. `alias ll='ls -lah'` – Create an alias
 36. `df -Th` – Show file system disk space
 37. `wc -l file.txt` – Count number of lines in a file
 38. `cut -d':' -f1 /etc/passwd` – Extract first column from file
 39. `awk '{print $1}' file.txt` – Print first column of

Text

40. `sed -i 's/old/new/g' file.txt` - Replace text in a file
41. `crontab -e` - Edit cron jobs
42. `cron -l` - List scheduled cron jobs
43. `rsync -av source/ target/` - Sync files
44. `scp user@host:/path/to/file .` - Secure copy file
45. `ssh user@host` - SSH into a server
46. `exit` - Logout from shell
47. `uptime` - Show system uptime
48. `free -m` - Show memory usage
49. `top` - Show system performance
50. `vmstat` - Show CPU/memory stats



Git Commands

- 51. `git init` - Initialize a git repository
- 52. `git clone repo_url` - Clone a repository
- 53. `git status` - Show changes
- 54. `git add .` - Stage all changes
- 55. `git commit -m "msg"` - Commit changes
- 56. `git push origin branch` - Push changes
- 57. `git pull origin branch` - Pull latest changes
- 58. `git checkout branch` - Switch branch
- 59. `git branch -a` - List all branches
- 60. `git merge branch` - Merge branch into current branch
- 61. `git log --oneline` - Show commit history
- 62. `git diff` - Show changes
- 63. `git stash` - Stash changes
- 64. `git stash pop` - Apply stashed changes

-
65. `git reset --hard HEAD~1` – Reset last commit
 66. `git revert commit_id` – Revert commit
 67. `git tag -a v1.0 -m "Version 1.0"` – Create tag
 68. `git push --tags` – Push tags
 69. `git rm file` – Remove a file from git
 70. `git clean -fd` – Remove untracked files
 71. `git config --global user.name "Your Name"` – Set global username
 72. `git config --global user.email "you@example.com"`
– Set global email
 73. `git show commit_id` – Show commit details
 74. `git blame file.txt` – Show changes by line
 75. `git remote -v` – Show remote repositories
 76. `git remote add origin URL` – Add remote repo
 77. `git rebase branch` – Rebase branch
 78. `git cherry-pick commit_id` – Pick specific commit

-
79. `git fetch` – Fetch remote changes
 80. `git log --graph` – Show graphical commit log
 81. `git branch -d branch_name` – Delete branch
 82. `git pull --rebase` – Rebase while pulling
 83. `git push --force` – Force push changes
 84. `git reflog` – Show reflog
 85. `git diff HEAD~1` – Show last commit changes
 86. `git apply patch.diff` – Apply patch
 87. `git reset --soft HEAD~1` – Soft reset
 88. `git archive --format=tar.gz HEAD > archive.tar.gz`
– Archive repo
 89. `git grep "search_term"` – Search for text
 90. `git bisect start` – Start binary search
 91. `git ls-files` – List tracked files
 92. `git update-index --assume-unchanged file` – Ignore
file temporarily

-
93. `git push origin --delete branch_name` - Delete remote branch
 94. `git pull --all` - Fetch all branches
 95. `git submodule add URL path` - Add a submodule
 96. `git submodule update --init --recursive` - Update submodules
 97. `git commit --amend -m "Updated commit message"` - Amend commit
 98. `git fetch --prune` - Remove deleted remote branches
 99. `git push --mirror destination_url` - Mirror repository
 100. `git worktree add ../branch_name branch_name` - Work with multiple branches

Docker Commands

101. `docker --version` - Check Docker version
102. `docker ps` - List running containers
103. `docker ps -a` - List all containers
104. `docker images` - List all images



-
105. `docker run -d -p 8080:80 image_name` – Run a container
 106. `docker exec -it container_id bash` – Access a running container
 107. `docker stop container_id` – Stop a container
 108. `docker rm container_id` – Remove a container
 109. `docker rmi image_id` – Remove an image
 110. `docker-compose up -d` – Start containers using Docker Compose
 111. `docker-compose down` – Stop and remove all containers
 112. `docker build -t my_image .` – Build an image from a Dockerfile
 113. `docker pull image_name` – Download an image from Docker Hub
 114. `docker push myrepo/image_name` – Push image to registry
 115. `docker logs container_id` – View container logs

-
- 116. **docker inspect container_id** – Get container details
 - 117. **docker network ls** – List Docker networks
 - 118. **docker network inspect network_name** – Inspect network details
 - 119. **docker volume ls** – List Docker volumes
 - 120. **docker volume inspect volume_name** – Inspect volume details
 - 121. **docker system prune -a** – Clean up unused containers, images, and networks
 - 122. **docker login** – Authenticate with Docker Hub
 - 123. **docker logout** – Logout from Docker Hub
 - 124. **docker stats** – Show live container resource usage
 - 125. **docker top container_id** – Show running processes inside a container
 - 126. **docker tag image_id new_repo:new_tag** – Tag an image
 - 127. **docker restart container_id** – Restart a container

128. `docker update --restart=always container_id` – Set restart policy

129. `docker rename old_name new_name` – Rename a container

130. `docker wait container_id` – Wait for a container to exit

131. `docker events` – Show real-time events

132. `docker history image_name` – Show image history

133. `docker cp container_id:/path/to/file .` – Copy file from container

134. `docker diff container_id` – Show container file changes

135. `docker pause container_id` – Pause a running container

136. `docker unpause container_id` – Resume a paused container

137. `docker info` – Show Docker system information

138. `docker search image_name` – Search for an image on Docker Hub



-
- 139. **docker run --rm image_name** – Run a container and remove after exit
 - 140. **docker network create my_network** – Create a custom Docker network
 - 141. **docker network connect my_network container_id** – Connect container to a network
 - 142. **docker network disconnect my_network container_id** – Disconnect container from a network
 - 143. **docker-compose build** – Build services defined in **docker-compose.yml**
 - 144. **docker-compose ps** – List services in a Docker Compose setup
 - 145. **docker-compose restart** – Restart all services in Compose
 - 146. **docker-compose logs -f** – Tail logs for all services
 - 147. **docker-compose exec service_name bash** – Run command inside a service container
 - 148. **docker-compose scale service=3** – Scale a service to 3 instances

149. **docker-compose stop** – Stop all services without removing containers

150. **docker-compose config** – Validate Docker Compose config

Kubernetes (kubectl)

151. **kubectl get nodes** – List nodes in the cluster

152. **kubectl get pods** – List pods

153. **kubectl describe pod pod_name** – Show pod details

154. **kubectl logs pod_name** – Show pod logs

155. **kubectl exec -it pod_name -- bash** – Access a running pod

156. **kubectl apply -f file.yaml** – Apply a YAML file

157. **kubectl delete pod pod_name** – Delete a pod

158. **kubectl get svc** – List services

159. **kubectl get deployments** – List deployments

160. **kubectl scale deployment my-deploy --replicas=5** – Scale deployment

-
161. `kubectl rollout status deployment my-deploy` -
Check rollout status
 162. `kubectl get configmaps` - List ConfigMaps
 163. `kubectl get secrets` - List Secrets
 164. `kubectl create configmap my-config`
`--from-file=config.txt` - Create ConfigMap
 165. `kubectl create secret generic my-secret`
`--from-literal=key=value` - Create Secret
 166. `kubectl port-forward pod/my-pod 8080:80` - Forward
port
 167. `kubectl delete svc my-service` - Delete a service
 168. `kubectl expose deployment my-deploy`
`--type=NodePort --port=80` - Expose deployment
 169. `kubectl get ingress` - List Ingress resources
 170. `kubectl describe ingress my-ingress` - Show
Ingress details
 171. `kubectl get namespaces` - List namespaces
 172. `kubectl create namespace my-namespace` - Create a
namespace

173. **kubectl delete namespace my-namespace** – Delete a namespace

174. **kubectl config view** – View kubeconfig settings

175. **kubectl get events** – Show cluster events

176. **kubectl drain node_name** – Drain a node for maintenance

177. **kubectl uncordon node_name** – Mark node as schedulable

178. **kubectl taint nodes node_name
key=value:NoSchedule** – Taint a node

179. **kubectl edit deployment my-deploy** – Edit deployment config

180. **kubectl rollout undo deployment my-deploy** – Rollback deployment

181. **kubectl set image deployment/my-deploy
container-name=myimage:v2** – Update container image

182. **kubectl autoscale deployment my-deploy --min=2
--max=5 --cpu-percent=80** – Autoscale deployment

183. **kubectl get hpa** – Show Horizontal Pod Autoscaler



-
184. `kubect1 cordon node_name` – Mark node as unschedulable
 185. `kubect1 delete -f resource.yaml` – Delete a resource using YAML
 186. `kubect1 top nodes` – Show node resource usage
 187. `kubect1 top pods` – Show pod resource usage
 188. `kubect1 get pv` – List PersistentVolumes
 189. `kubect1 get pvc` – List PersistentVolumeClaims
 190. `kubect1 logs -f pod_name` – Stream logs from a pod
 191. `kubect1 exec -it pod_name -- sh` – Access a pod using `sh` shell
 192. `kubect1 delete pod --grace-period=0 --force pod_name` – Force delete pod
 193. `kubect1 get all` – List all resources
 194. `kubect1 get roles` – List roles
 195. `kubect1 get rolebindings` – List role bindings
 196. `kubect1 get clusterroles` – List cluster roles

197. **kubectl get clusterrolebindings** – List cluster role bindings

198. **kubectl get networkpolicy** – List network policies

199. **kubectl delete node node_name** – Delete a node

200. **kubectl run nginx --image=nginx --restart=Never** – Run a single pod

Terraform Commands

201. **terraform --version** – Check Terraform version

202. **terraform init** – Initialize Terraform working directory

203. **terraform plan** – Show execution plan

204. **terraform apply -auto-approve** – Apply configuration

205. **terraform destroy -auto-approve** – Destroy infrastructure

206. **terraform show** – Show Terraform state

207. **terraform validate** – Validate Terraform configuration



-
- 208. `terraform fmt` – Format Terraform code
 - 209. `terraform providers` – List providers used
 - 210. `terraform state list` – Show managed resources
 - 211. `terraform state show resource_name` – Show specific resource details
 - 212. `terraform state rm resource_name` – Remove resource from state
 - 213. `terraform taint resource_name` – Mark resource for recreation
 - 214. `terraform untaint resource_name` – Remove taint from resource
 - 215. `terraform import resource_type.name id` – Import existing infrastructure
 - 216. `terraform refresh` – Refresh state file
 - 217. `terraform graph` – Generate dependency graph
 - 218. `terraform workspace list` – List available workspaces
 - 219. `terraform workspace select workspace_name` –

Switch workspace

220. **terraform workspace new workspace_name** – Create new workspace

221. **terraform state mv old_resource new_resource** – Rename resource

222. **terraform output** – Show Terraform outputs

223. **terraform output output_name** – Get specific output value

224. **terraform apply -target=resource_name** – Apply specific resource

225. **terraform console** – Open interactive Terraform shell

226. **terraform debug** – Enable debugging logs

227. **terraform version** – Show installed Terraform version

228. **terraform plan -var="instance_type=t2.micro"** – Pass variable via CLI

229. **terraform apply -var-file="vars.tfvars"** – Apply using variable file

-
- 230. **terraform apply -auto-approve** – Skip manual approval
 - 231. **terraform workspace delete workspace_name** – Delete workspace
 - 232. **terraform plan -detailed-exitcode** – Get exit codes for changes
 - 233. **terraform force-unlock LOCK_ID** – Unlock Terraform state
 - 234. **terraform state push** – Manually push state file
 - 235. **terraform state pull** – Download current state file
 - 236. **terraform output -json** – Get output in JSON format
 - 237. **terraform fmt -recursive** – Format Terraform in subdirectories
 - 238. **terraform destroy -target=resource_name** – Destroy specific resource
 - 239. **terraform show -json** – Show JSON output
 - 240. **terraform validate -no-color** – Validate without

Colors

- 241. `terraform workspace select default` – Switch to default workspace
- 242. `terraform plan -var 'region=us-east-1'` – Pass a variable inline
- 243. `terraform apply -refresh-only` – Only refresh state, do not modify
- 244. `terraform state replace-provider old new` – Replace provider
- 245. `terraform plan -out=tfplan` – Save plan output
- 246. `terraform apply tfplan` – Apply saved plan
- 247. `terraform version -json` – Output Terraform version in JSON
- 248. `terraform workspace show` – Show current workspace
- 249. `terraform plan -var-file=config.tfvars` – Use a variable file
- 250. `terraform output -state=terraform.tfstate` – Show output from a specific state file

Jenkins CLI Commands

- 251. `java -jar jenkins-cli.jar -s http://localhost:8080/ help` - Show CLI help
- 252. `java -jar jenkins-cli.jar -s http://localhost:8080/ list-jobs` - List all jobs
- 253. `java -jar jenkins-cli.jar -s http://localhost:8080/ build job_name` - Trigger a job
- 254. `java -jar jenkins-cli.jar -s http://localhost:8080/ console job_name` - View job console output
- 255. `java -jar jenkins-cli.jar -s http://localhost:8080/ delete-job job_name` - Delete a job
- 256. `java -jar jenkins-cli.jar -s http://localhost:8080/ disable-job job_name` - Disable a job
- 257. `java -jar jenkins-cli.jar -s http://localhost:8080/ enable-job job_name` - Enable a job
- 258. `java -jar jenkins-cli.jar -s`

```
http://localhost:8080/ safe-restart - Restart  
Jenkins safely
```

```
259. java -jar jenkins-cli.jar -s  
http://localhost:8080/ set-build-description  
job_name 1 "Updated Description" - Update build  
description
```

```
260. java -jar jenkins-cli.jar -s  
http://localhost:8080/ version - Show Jenkins  
version
```

```
261. java -jar jenkins-cli.jar -s  
http://localhost:8080/ who-am-i - Check  
authenticated user
```

```
262. java -jar jenkins-cli.jar -s  
http://localhost:8080/ get-job job_name - Get job  
config
```

```
263. java -jar jenkins-cli.jar -s  
http://localhost:8080/ set-job job_name <  
config.xml - Set job config
```

```
264. java -jar jenkins-cli.jar -s  
http://localhost:8080/ shutdown - Shutdown Jenkins
```

```
265. java -jar jenkins-cli.jar -s
```



```
http://localhost:8080/ reload-configuration -  
Reload Jenkins configuration
```

```
266. java -jar jenkins-cli.jar -s  
http://localhost:8080/ install-plugin plugin-name -  
Install a plugin
```

```
267. java -jar jenkins-cli.jar -s  
http://localhost:8080/ list-plugins - List  
installed plugins
```

```
268. java -jar jenkins-cli.jar -s  
http://localhost:8080/ delete-plugin plugin-name -  
Delete a plugin
```

```
269. java -jar jenkins-cli.jar -s  
http://localhost:8080/ update-job job_name <  
config.xml - Update job configuration
```

```
270. java -jar jenkins-cli.jar -s  
http://localhost:8080/ build job_name -p  
PARAM=value - Trigger job with parameters
```

```
271. java -jar jenkins-cli.jar -s  
http://localhost:8080/ safe-exit - Gracefully stop  
Jenkins
```

```
272. java -jar jenkins-cli.jar -s  
http://localhost:8080/ disconnect-node node_name -  
Disconnect a node
```

273. `java -jar jenkins-cli.jar -s
http://localhost:8080/ install-plugin plugin.hpi -`
Install plugin from file

274. `java -jar jenkins-cli.jar -s
http://localhost:8080/ get-view view_name -` Get
view configuration

275. `java -jar jenkins-cli.jar -s
http://localhost:8080/ create-view view_name <
config.xml -` Create a new view

Ansible Commands

276. `ansible --version` - Check Ansible version

277. `ansible all -m ping` - Ping all hosts in the
inventory

278. `ansible-playbook playbook.yml` - Run a playbook

279. `ansible-inventory --list` - Show inventory

280. `ansible all -a "uptime"` - Run a command on all
hosts

281. `ansible-playbook -i inventory.ini playbook.yml -`
Specify inventory file

282. `ansible-playbook -e "variable=value" playbook.yml`



– Pass extra variables

283. **ansible-playbook --syntax-check playbook.yml** –
Check for syntax errors

284. **ansible-vault encrypt file.yml** – Encrypt a file

285. **ansible-vault decrypt file.yml** – Decrypt a file

286. **ansible-vault create secret.yml** – Create a new
encrypted file

287. **ansible all -m setup** – Gather system facts

288. **ansible all -m shell -a "df -h"** – Run shell
commands

289. **ansible all -m service -a "name=nginx
state=started"** – Manage services

290. **ansible all -m yum -a "name=httpd state=present"**
– Install packages

291. **ansible all -m copy -a "src=file.txt
dest=/tmp/file.txt"** – Copy files

292. **ansible all -m file -a "path=/tmp/test mode=0755
state=directory"** – Create a directory

293. **ansible-playbook -i inventory.yml site.yml --tags**



web – Run specific tags

294. **ansible-galaxy install role_name** – Install a role

295. **ansible-galaxy list** – List installed roles

296. **ansible-playbook -i inventory playbook.yml
--check** – Run in dry-run mode

297. **ansible-doc -l** – List all available Ansible modules

298. **ansible-playbook -i inventory playbook.yml
--start-at-task="task_name"** – Start playbook from a specific task

299. **ansible -i inventory -m ping all** – Ping all hosts

300. **ansible-galaxy init myrole** – Create a new Ansible role

AWS CLI Commands

301. **aws configure** – Configure AWS CLI credentials

302. **aws s3 ls** – List S3 buckets

303. **aws s3 cp file.txt s3://mybucket/** – Copy file to S3



304. `aws s3 sync . s3://mybucket/` - Sync local files to S3

305. `aws s3 rm s3://mybucket/file.txt` - Delete file from S3

306. `aws ec2 describe-instances` - List EC2 instances

307. `aws ec2 start-instances --instance-ids i-1234567890` - Start EC2 instance

308. `aws ec2 stop-instances --instance-ids i-1234567890` - Stop EC2 instance

309. `aws ec2 terminate-instances --instance-ids i-1234567890` - Terminate EC2 instance

310. `aws ec2 create-key-pair --key-name MyKeyPair` - Create a new key pair

311. `aws ec2 describe-volumes` - List EBS volumes

312. `aws ec2 create-volume --size 10 --region us-east-1` - Create an EBS volume

313. `aws ec2 attach-volume --volume-id vol-12345 --instance-id i-12345 --device /dev/sdf` - Attach volume

-
- 314. `aws ec2 describe-security-groups` - List security groups
 - 315. `aws lambda list-functions` - List Lambda functions
 - 316. `aws lambda invoke --function-name my_lambda output.json` - Invoke a Lambda function
 - 317. `aws eks update-kubeconfig --name cluster_name` - Configure kubectl for EKS
 - 318. `aws rds describe-db-instances` - List RDS instances
 - 319. `aws rds stop-db-instance --db-instance-identifier mydb` - Stop RDS instance
 - 320. `aws rds start-db-instance --db-instance-identifier mydb` - Start RDS instance
 - 321. `aws ec2 create-snapshot --volume-id vol-12345` - Create EBS snapshot
 - 322. `aws ec2 describe-snapshots` - List EBS snapshots
 - 323. `aws cloudwatch describe-alarms` - List CloudWatch alarms
 - 324. `aws ssm send-command --document-name`

```
AWS-RunShellScript --targets  
"Key=instanceIds,Values=i-12345" --parameters  
commands="df -h" - Run command on EC2
```

```
325. aws iam list-users - List IAM users
```

```
326. aws iam list-roles - List IAM roles
```

```
327. aws route53 list-hosted-zones - List Route 53  
hosted zones
```

```
328. aws route53 create-hosted-zone --name example.com  
- Create a Route 53 hosted zone
```

```
329. aws cloudformation list-stacks - List  
CloudFormation stacks
```

```
330. aws cloudformation create-stack --stack-name  
my-stack --template-body file://template.json -  
Create a stack
```

```
331. aws cloudformation delete-stack --stack-name  
my-stack - Delete a stack
```

```
332. aws ecr get-login-password --region us-east-1 |  
docker login --username AWS --password-stdin  
account_id.dkr.ecr.us-east-1.amazonaws.com - Login  
to ECR
```

-
333. **aws ecr create-repository --repository-name myrepo** – Create an ECR repository
334. **aws ecr list-repositories** – List ECR repositories
335. **aws dynamodb list-tables** – List DynamoDB tables
336. **aws dynamodb describe-table --table-name mytable**
– Describe a DynamoDB table
337. **aws dynamodb put-item --table-name mytable --item '{"id": {"S": "123"}, "name": {"S": "Test"}}'** –
Insert item into DynamoDB
338. **aws sns list-topics** – List SNS topics
339. **aws sns publish --topic-arn arn:aws:sns:us-east-1:123456789012:MyTopic --message "Hello"** – Send SNS message
340. **aws sqs list-queues** – List SQS queues
341. **aws sqs send-message --queue-url https://sqs.us-east-1.amazonaws.com/123456789012/MyQueue --message-body "Test message"** – Send message to SQS
342. **aws sqs receive-message --queue-url https://sqs.us-east-1.amazonaws.com/123456789012/MyQueue** – Receive SQS message

343. `aws kms list-keys` – List KMS keys

344. `aws kms encrypt --key-id key-id --plaintext "Hello"` – Encrypt text using KMS

345. `aws kms decrypt --ciphertext-blob fileb://encrypted.txt` – Decrypt KMS ciphertext

346. `aws ec2 describe-key-pairs` – List EC2 key pairs

347. `aws ec2 delete-key-pair --key-name MyKeyPair` – Delete a key pair

348. `aws s3 mb s3://my-new-bucket` – Create a new S3 bucket

349. `aws s3 rb s3://my-new-bucket --force` – Delete an S3 bucket

350. `aws cloudfront list-distributions` – List CloudFront distributions

Helm Commands

351. `helm version` – Check Helm version

352. `helm repo add stable https://charts.helm.sh/stable` – Add Helm repository

353. `helm repo update` – Update Helm repositories



354. **helm search repo nginx** – Search for a Helm chart

355. **helm install myapp stable/nginx** – Install a Helm chart

356. **helm upgrade myapp stable/nginx** – Upgrade a Helm release

357. **helm rollback myapp 1** – Rollback release to a previous version

358. **helm uninstall myapp** – Uninstall a Helm release

359. **helm list** – List all installed Helm releases

360. **helm status myapp** – Show the status of a Helm release

361. **helm get values myapp** – Show values used in a Helm release

362. **helm get manifest myapp** – Show Kubernetes manifests of a release

363. **helm template myapp stable/nginx** – Render chart templates locally

364. **helm lint mychart/** – Validate Helm chart syntax

365. **helm package mychart/** – Package a Helm chart

-
- 366. **helm repo remove stable** – Remove a Helm repository
 - 367. **helm dependency update mychart/** – Update chart dependencies
 - 368. **helm dependency build mychart/** – Build chart dependencies
 - 369. **helm history myapp** – Show the history of a Helm release
 - 370. **helm create mychart** – Create a new Helm chart
 - 371. **helm pull stable/nginx** – Download a chart
 - 372. **helm plugin list** – List installed Helm plugins
 - 373. **helm env** – Show Helm environment variables
 - 374. **helm show values stable/nginx** – Show default values of a chart
 - 375. **helm upgrade --install myapp stable/nginx** – Install or upgrade release

Prometheus Commands

376. **prometheus --version** – Check Prometheus version

377. **promtool check config prometheus.yml** – Validate Prometheus configuration

378. **prometheus --config.file=prometheus.yml** – Start Prometheus

379. **prometheus --storage.tsdb.path=data/** – Define storage path

380. **prometheus --web.listen-address=:9090** – Start Prometheus on a specific port

381. **curl http://localhost:9090/api/v1/targets** – Check active Prometheus targets

382. **curl http://localhost:9090/api/v1/status/config** – Fetch Prometheus config

383. **curl http://localhost:9090/api/v1/query?query=up**
– Run an instant query

384. **prometheus --query.lookback-delta=5m** – Adjust query range

385. **prometheus --storage.tsdb.retention.time=15d** –

Set data retention

386. **prometheus --web.enable-lifecycle** – Enable reload API

387. **curl -X POST http://localhost:9090/-/reload** – Reload configuration

388. **systemctl restart prometheus** – Restart Prometheus service

389. **journalctl -u prometheus --no-pager** – Check Prometheus logs

390. **promtool check rules rules.yml** – Validate recording rules

391. **prometheus --storage.tsdb.max-block-duration=2h** – Adjust block duration

392. **prometheus --storage.tsdb.no-lockfile** – Disable lockfile

393. **prometheus --enable-feature=remote-write-receiver** – Enable remote write

394. **prometheus --web.enable-admin-api** – Enable admin API

395. `promtool query instant http://localhost:9090 "up"`
- Query using `promtool`

396. `promtool tsdb analyze data/` - Analyze TSDB database

397. `curl http://localhost:9090/api/v1/rules` - Show alerting rules

398. `prometheus --log.level=debug` - Start Prometheus with debug logs

399. `promtool check-metrics metrics.prom` - Validate metrics file

400. `prometheus --config.file=prometheus.yml`
`--web.console.templates=console/` - Load custom web templates

Grafana Commands

401. `grafana-server --version` - Check Grafana version

402. `systemctl start grafana-server` - Start Grafana service

403. `systemctl stop grafana-server` - Stop Grafana service

404. `systemctl restart grafana-server` - Restart



Grafana

405. `systemctl status grafana-server` – Check Grafana status

406. `grafana-cli plugins list` – List installed plugins

407. `grafana-cli plugins install grafana-clock-panel` – Install a Grafana plugin

408. `grafana-cli plugins update-all` – Update all plugins

409. `grafana-cli admin reset-admin-password newpassword` – Reset admin password

410. `curl http://localhost:3000/api/health` – Check Grafana health

411. `curl -u admin:admin http://localhost:3000/api/dashboards/home` – Get home dashboard

412. `grafana-cli plugins remove plugin-name` – Remove a plugin

413. `grafana-cli admin reset-admin-password newpass` – Reset admin password



414. **grafana-server -config /etc/grafana/grafana.ini -**
Start with custom config

415. **curl -X GET http://localhost:3000/api/dashboards**
- List dashboards

416. **curl -X POST**
http://localhost:3000/api/dashboards/db -d
'@dashboard.json' - Create dashboard

417. **journalctl -u grafana-server --no-pager - Check**
logs

418. **grafana-cli plugins install**
grafana-simple-json-datasource - Install JSON data
source

419. **grafana-cli plugins update - Update Grafana**
plugins

420. **systemctl enable grafana-server - Enable Grafana**
on startup

421. **grafana-cli plugins update-all - Update all**
plugins

422. **grafana-cli server restart - Restart Grafana**

423. **grafana-cli admin reset-admin-password**

`mynewpassword` – Reset password

424. `curl http://localhost:3000/api/org` – Get organization details

425. `grafana-server --homepath /var/lib/grafana` – Run Grafana with a specific path

ArgoCD Commands

426. `argocd version` – Show ArgoCD CLI version

427. `argocd login my-argocd.com` – Login to ArgoCD

428. `argocd app list` – List all ArgoCD applications

429. `argocd app get my-app` – Show details of an app

430. `argocd app sync my-app` – Sync an application

431. `argocd app delete my-app` – Delete an application

432. `argocd app rollback my-app 1` – Rollback an application

433. `argocd app history my-app` – Show deployment history

434. `argocd app create my-app --repo`



```
https://github.com/user/repo.git --path app -  
Create an application
```

```
435. argocd app update my-app --sync-policy automated  
- Enable auto-sync
```

```
436. argocd app set my-app --dest-server  
https://k8s-cluster - Set destination server
```

```
437. argocd app unset my-app --dest-server - Unset  
destination server
```

```
438. argocd repo list - List connected repositories
```

```
439. argocd repo add https://github.com/user/repo.git  
- Add a repository
```

```
440. argocd cluster list - List managed clusters
```

```
441. argocd cluster add my-cluster - Add a new cluster
```

```
442. argocd login my-argocd.com --username admin  
--password pass - Login with credentials
```

```
443. argocd app diff my-app - Show differences in  
application
```

```
444. argocd proj list - List ArgoCD projects
```

```
445. argocd proj create my-project - Create a project
```



446. `argocd proj delete my-project` – Delete a project

447. `argocd settings list` – Show ArgoCD settings

448. `argocd account list` – List user accounts

449. `argocd logout` – Logout from ArgoCD

450. `argocd help` – Show help for ArgoCD CLI

GitHub Actions

451. `gh workflow list` – List all GitHub Actions workflows

452. `gh workflow run my-workflow.yml` – Run a specific workflow

453. `gh run list` – List all workflow runs

454. `gh run view <run_id>` – View details of a workflow run

455. `gh run cancel <run_id>` – Cancel a running workflow

456. `gh run rerun <run_id>` – Rerun a failed workflow

457. `gh secret list` – List GitHub repository secrets



458. **gh secret set MY_SECRET --body "value"** – Set a secret in GitHub Actions

459. **gh variable list** – List environment variables in GitHub Actions

460. **gh variable set MY_VAR --body "value"** – Set an environment variable

GitLab CI/CD

461. **gitlab-runner register** – Register a new GitLab runner

462. **gitlab-runner start** – Start GitLab Runner

463. **gitlab-runner stop** – Stop GitLab Runner

464. **gitlab-runner verify** – Verify if runners are properly registered

465. **gitlab-runner list** – List all registered runners

466. **gitlab-runner status** – Show GitLab Runner status

467. **gitlab-runner logs** – View logs for GitLab Runner

CircleCI

468. **circleci config validate** – Validate CircleCI



Configuration

469. `circleci local execute -c .circleci/config.yml` – Run pipeline locally

470. `circleci project list` – List all projects in CircleCI

471. `circleci setup` – Configure CircleCI for the first time

472. `circleci build` – Run a build locally

473. `circleci config process .circleci/config.yml` – Process CircleCI config

474. `circleci rerun <build_number>` – Rerun a specific build

475. `circleci admin namespace list` – List CircleCI namespaces

Security & Compliance (Trivy, Vault, Falco)

Trivy (Container Security Scanning)

476. `trivy image nginx:latest` – Scan a container image

477. `trivy filesystem /path/to/dir` – Scan a directory



478. `trivy repo https://github.com/user/repo` – Scan a Git repository

479. `trivy k8s cluster` – Scan a Kubernetes cluster

480. `trivy config .` – Scan IaC configurations

Vault (Secrets Management)

481. `vault server -dev` – Start Vault in development mode

482. `vault kv put secret/mysecret key=value` – Store a secret

483. `vault kv get secret/mysecret` – Retrieve a secret

484. `vault kv delete secret/mysecret` – Delete a secret

485. `vault login <token>` – Authenticate with Vault

Monitoring & Logging (Loki, ELK, Fluentd)

Loki (Log Aggregation)

486. `loki -config.file=loki-config.yml` – Start Loki with config file

487. `curl -X GET`



```
http://localhost:3100/loki/api/v1/labels - Get  
available labels
```

```
488. curl -X GET  
'http://localhost:3100/loki/api/v1/query?query={app  
="nginx"}' - Query logs
```

ELK Stack (Elasticsearch, Logstash, Kibana)

```
489. curl -X GET "localhost:9200/_cat/indices?v" -  
List Elasticsearch indices
```

```
490. curl -X POST "localhost:9200/logs/_doc/" -H  
"Content-Type: application/json" -d '{"message":  
"hello world"}' - Insert a log entry
```

```
491. curl -X GET "localhost:9200/logs/_search" -  
Search logs
```

```
492. systemctl start logstash - Start Logstash
```

```
493. systemctl restart kibana - Restart Kibana
```

Fluentd (Log Forwarding)

```
494. fluentd --config fluentd.conf - Start Fluentd  
with a specific config
```

```
495. fluentd --dry-run --config fluentd.conf -  
Validate Fluentd config
```

Final Set of Essential Commands

496. **htop** – Interactive process monitoring

497. **iftop** – Monitor network bandwidth usage

498. **iostat** – Monitor disk I/O usage

499. **nc -zv host 80** – Check if a port is open

500. **curl -I http://example.com** – Fetch HTTP headers