

1. Jenkins / GitHub Actions / GitLab CI

Best Practices:

- Use pipeline-as-code (e.g., Jenkinsfile, .github/workflows/)
- Keep pipelines modular: break into stages (build, test, deploy)
- Implement approval gates before production deployments
- Use secrets management (not hard-coded credentials)
- Integrate automated tests (unit, integration)
- Parallelize jobs to speed up pipeline execution
- Limit plugin use in Jenkins to reduce complexity/security risk

2. Docker

Best Practices:

- Use small base images (like Alpine) for minimal attack surface
- Avoid using latest tag in production
- Multi-stage builds to reduce image size
- Don't run containers as root
- Scan images for vulnerabilities using tools like Trivy or Clair
- Use .dockerignore to avoid copying unnecessary files

3. Kubernetes (K8s/EKS)

Best Practices:

- Use Helm charts for repeatable and templated deployments
- Apply resource requests and limits on all pods
- Enable liveness and readiness probes
- Use RBAC and Namespaces for multi-team access control
- Externalize secrets with AWS Secrets Manager, Vault, or Sealed Secrets
- Avoid hardcoding config; use ConfigMaps and environment variables

- Set up cluster autoscaler and HPA for cost efficiency

4. Terraform

Best Practices:

- Use modules to reuse and standardize code
- Remote state with locking (e.g., S3 + DynamoDB)
- Run terraform plan and terraform validate in CI/CD
- Don't store secrets in .tf files or state files
- Version control all Terraform code
- Tag resources for cost tracking and ownership
- Use Workspace for multi environment deployment

6. CloudWatch / Prometheus / Grafana

Best Practices:

- Set up proactive alerts (CPU, memory, latency, etc.)
- Use structured logging (JSON format)
- Create dashboards per service or team
- Monitor both application and infrastructure metrics
- Integrate alerting with Slack, PagerDuty, or OpsGenie

7. Splunk / ELK Stack / AppDynamics

Best Practices:

- Centralize logs from all services
- Tag logs with metadata (e.g., environment, pod name, app)
- Use log retention and archiving policies
- Secure log access with RBAC
- Set thresholds and anomaly alerts for early detection

8. AWS Secrets Manager / HashiCorp Vault

Best Practices:

- Never hardcode secrets in source code or Git
- Enable automatic rotation of secrets
- Use IAM roles for access control
- Audit access logs regularly
- Limit secret scope per environment/service

9. ArgoCD / FluxCD

Best Practices:

- Enable automated sync with manual approvals for prod
- Use ApplicationSets for multi-tenant or multi-cluster deployments
- Store all manifests/Helm charts in Git
- Track and alert on drift detection
- Don't store secrets in Git – use integrations with sealed-secrets or external secrets

General DevOps Best Practices (Cross-Tool)

- Shift-left testing and security: run tests and scans early in the pipeline
- Implement Role-Based Access Control (RBAC) across all tools
- Backup important configs and states (e.g., Jenkins configs, Terraform state)
- Use tagging and naming conventions for traceability
- Keep tool versions and plugins up to date
- Monitor costs and optimize idle resources
- Automate repetitive tasks and integrate tools (e.g., Slack notifications for builds)