50 DevOps Interview Questions & Answers

Complete Technical Interview Guide for DevOps Engineers

About This Guide

DevOps bridges development and operations, emphasizing automation, collaboration, and continuous improvement. This guide covers essential DevOps concepts, tools, and practices frequently tested in technical interviews.
Last Updated: December 2024 | **Coverage:** CI/CD, Containers, Orchestration, Cloud, Monitoring, Security

CI/CD Docker Kubernetes AWS Prometheus Terraform
Jenkins GitLab CI GitHub Actions Ansible Linux Networking

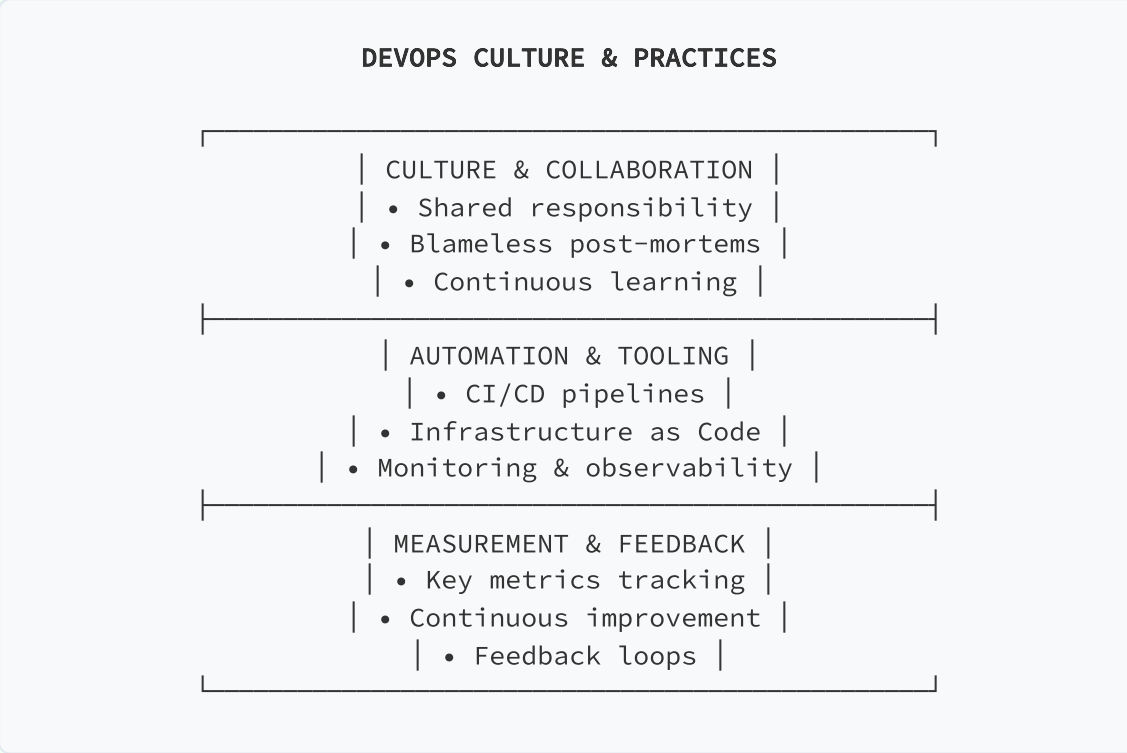
Table of Contents

> 💡 **Pro Tip:** DevOps interviews often focus on practical scenarios. Be ready to design solutions, troubleshoot issues, and explain trade-offs.

1. DevOps Fundamentals

1 What is DevOps and its key principles?

DevOps is a cultural and technical movement that bridges development and operations teams to improve collaboration, automation, and continuous delivery.



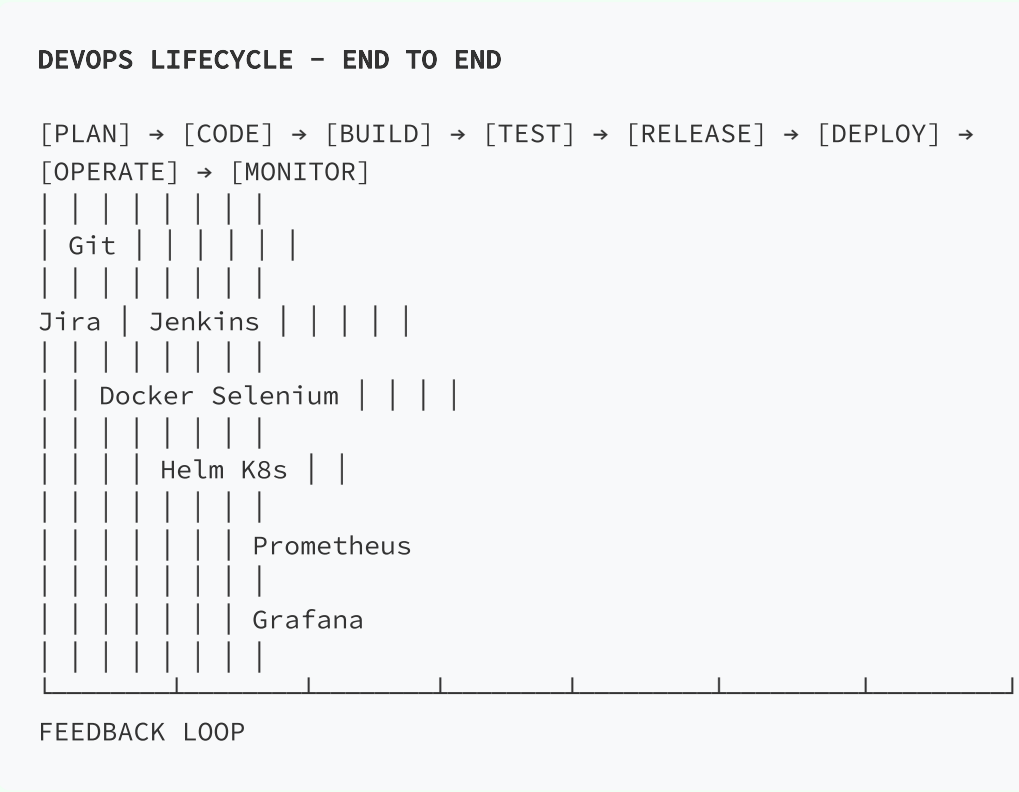
Key Principles (CALMS):

- **C**ulture - Collaboration between teams
- **A**utomation - Reduce manual intervention
- **L**ean - Eliminate waste, optimize flow
- **M**easurement - Data-driven decisions
- **S**haring - Knowledge sharing, transparency

Benefits:

- Faster time to market
- Improved deployment frequency
- Lower failure rate of new releases
- Faster mean time to recovery
- Better resource utilization

2 Explain the DevOps Lifecycle (CI/CD Pipeline)



Stages Explained:

1. **Plan:** Requirements, backlog, sprint planning
2. **Code:** Version control, peer reviews, branching strategy
3. **Build:** Compilation, dependency resolution, artifact creation
4. **Test:** Unit, integration, functional, performance testing
5. **Release:** Versioning, change management, approval gates
6. **Deploy:** Infrastructure provisioning, application deployment
7. **Operate:** Configuration management, scaling, maintenance
8. **Monitor:** Performance tracking, log analysis, alerting

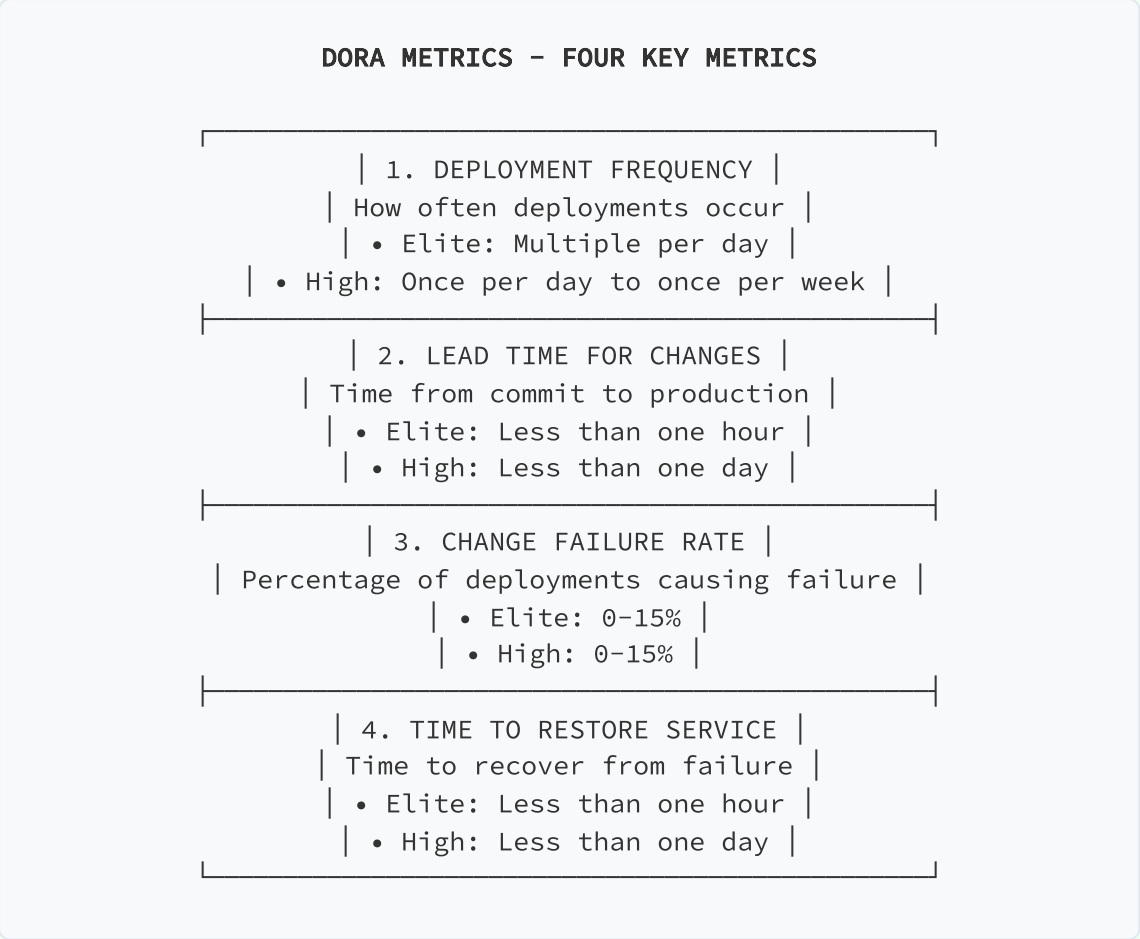
3 Difference between DevOps, SRE, and Platform Engineering

Aspect	DevOps	SRE (Site Reliability Engineering)	Platform Engineering
Focus	Culture, collaboration, automation	Reliability, scalability, performance	Internal developer platforms, tooling
Primary Goal	Faster delivery, collaboration	System reliability, uptime	Developer productivity, standardization
Key Metrics	Deployment frequency, lead time	SLOs, SLIs, error budgets	Platform adoption, time to deploy
Tools	CI/CD, configuration management	Monitoring, alerting, chaos engineering	Internal platforms, self-service portals
Team Structure	Cross-functional teams	Specialized SRE teams	Platform teams

Relationship: SRE implements DevOps principles with a focus on reliability. Platform Engineering builds the foundation that enables DevOps practices at scale.

4 What are the key DevOps metrics (DORA Metrics)?

DORA Metrics (DevOps Research and Assessment) are four key metrics that measure DevOps performance:



Additional Important Metrics:

- **Mean Time Between Failures (MTBF):** Average time between system failures
- **Mean Time To Recovery (MTTR):** Average time to restore service
- **Availability:** Uptime percentage (e.g., 99.9%, 99.99%)
- **Deployment Success Rate:** Percentage of successful deployments
- **Resource Utilization:** CPU, memory, storage usage

5 Explain Infrastructure as Code (IaC) principles

Infrastructure as Code (IaC) is managing and provisioning infrastructure through machine-readable definition files rather than physical hardware configuration or interactive configuration tools.

```
 "comment">"comment"># Terraform Example - AWS Infrastructure
"devops-keyword">provider "aws" {
  region = "us-east-">1"
}

"devops-keyword">resource "aws_vpc" "main" {
  cidr_block = """>10.">0.">0.">0/">16"

  tags = {
    Name = "main-vpc"
    Environment = "production"
  }
}

"devops-keyword">resource "aws_subnet" "public" {
  vpc_id     = aws_vpc.main.id
  cidr_block = """>10.">0.">1.">0/">24"

  tags = {
    Name = "public-subnet"
  }
}

"devops-keyword">resource "aws_instance" "web" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.public.id

  tags = {
    Name = "web-server"
  }
}
```

IaC Principles:

- **Declarative vs Imperative:** Define WHAT you want vs HOW to get it
- **Idempotency:** Same configuration applied multiple times results in same state
- **Version Control:** Store IaC files in Git for history and collaboration
- **Modularity:** Reusable modules/components
- **Testing:** Test infrastructure changes like application code
- **Documentation:** Self-documenting code

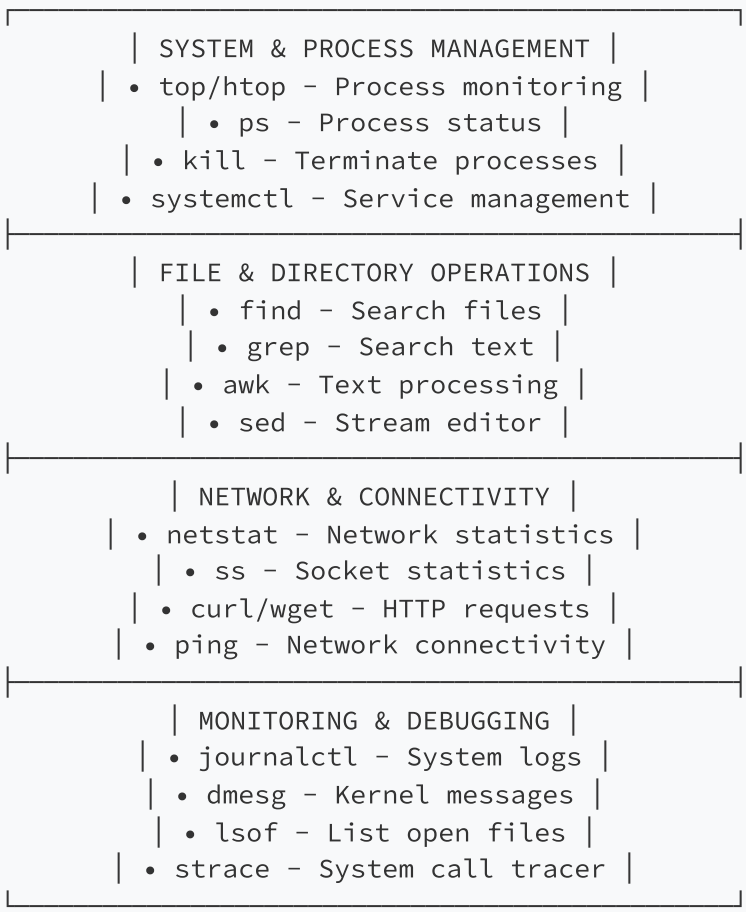
Popular IaC Tools:

Terraform AWS CloudFormation Azure ARM

Google Deployment Manager Pulumi Ansible Chef Puppet

2. Linux & Shell Scripting

6 Essential Linux Commands for DevOps



Practical Examples:

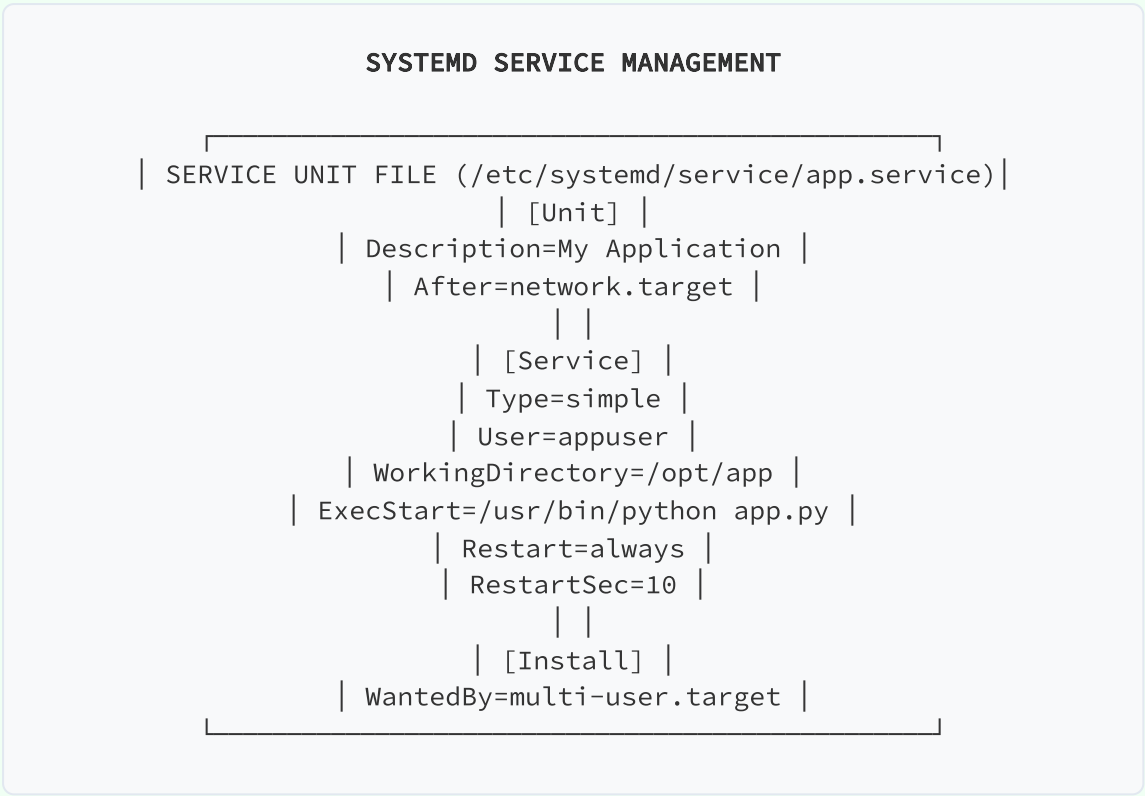
```
 "comment">"comment"># Find and kill a process
"command">ps aux | "command">grep nginx
"command">kill -"devops-keyword">9 "devops-keyword"><PID>

"comment">"comment"># Check disk usage
"command">df -h
"command">du -sh /var/log/*

"comment">"comment"># Monitor real-time system metrics
"command">htop
"command">iotop
"command">iftop

"comment">"comment"># Check listening "devops-keyword">ports
"command">netstat -tulpn
"command">ss -tulpn

"comment">"comment"># Analyze log files
"command">tail -f /var/log/nginx/access.log
"command">grep "ERROR" /var/log/app.log | "command">head -"devops-keyw
```

7 Shell Scripting for Automation

```
 "comment">"comment">#!/bin/bash
"comment">"comment"># ============================================
"comment">"comment"># DevOps Automation Script - Backup & Cleanup
"comment">"comment"># ============================================

"comment">"comment"># Configuration
"devops-keyword">BACKUP_DIR="tool-">name">"/backups"
"devops-keyword">LOG_FILE="tool-">name">"/var/log/backup_$(date +%Y%m%
"devops-keyword">RETENTION_DAYS="devops-keyword">"devops-keyword">7
"devops-keyword">APP_DIR="tool-">name">"/var/www/app"
"devops-keyword">DB_NAME="tool-">name">"app_db"

"comment">"comment"># Logging function
"devops-keyword">log_message() {
    "command">echo "tool-">name">"$(date '+%Y-%m-%d %H:%M:%S') - $">1"
}

"comment">"comment"># Error handling function
"devops-keyword">handle_error() {
    "devops-keyword">log_message "tool-">name">"ERROR: $">1"
    "command">exit "devops-keyword">"devops-keyword">1
}

"comment">"comment"># Check prerequisites
"command">check_prerequisites() {
    "command">log_message "tool-">name">"Checking prerequisites..."

    "comment">"comment"># Check if running as root
    "command">if [[ "devops-keyword">$EUID -ne "devops-keyword">"devop
        "devops-keyword">handle_error "tool-">name">"This script must
    "command">fi

    "comment">"comment"># Check required commands
    "devops-keyword">for cmd "devops-keyword">in mysqldump tar gzip fi
        "command">command -v "devops-keyword">$cmd >/dev/"devops-keywo
    "devops-keyword">done
}

"comment">"comment"># Create database backup
"devops-keyword">backup_database() {
    "command">log_message "tool-">name">"Starting database backup..."

    "devops-keyword">local backup_file="tool-">name">"${BACKUP_DIR}/db

    "command">if mysqldump --single-transaction "devops-keyword">$DB_N
        "command">log_message "tool-">name">"Database backup completed
    "command">else
```

```
        "devops-keyword">handle_error "tool-">name">"Database backup f
    "command">fi
}

"comment">"comment"># Create application backup
"devops-keyword">backup_application() {
    "command">log_message "tool-">name">"Starting application backup..

    "devops-keyword">local backup_file="tool-">name">"${BACKUP_DIR}/ap

    "command">if tar -czf "devops-keyword">$backup_file -C / "devops-k
        "command">log_message "tool-">name">"Application backup comple
    "command">else
        "devops-keyword">handle_error "tool-">name">"Application backu
    "command">fi
}

"comment">"comment"># Cleanup old backups
"devops-keyword">cleanup_old_backups() {
    "command">log_message "tool-">name">"Cleaning up backups older tha

    "devops-keyword">local files_deleted=$(find "devops-keyword">$BACK

    "command">log_message "tool-">name">"Deleted $files_deleted old ba
}

"comment">"comment"># Main execution
"devops-keyword">main() {
    "command">check_prerequisites
    "command">backup_database
    "command">backup_application
    "command">cleanup_old_backups
    "command">log_message "tool-">name">"Backup process completed succ
}

"comment">"comment"># Run main function
"devops-keyword">main "devops-keyword">"$@"
```

8 Linux Process Management & Systemd



Systemd Commands:

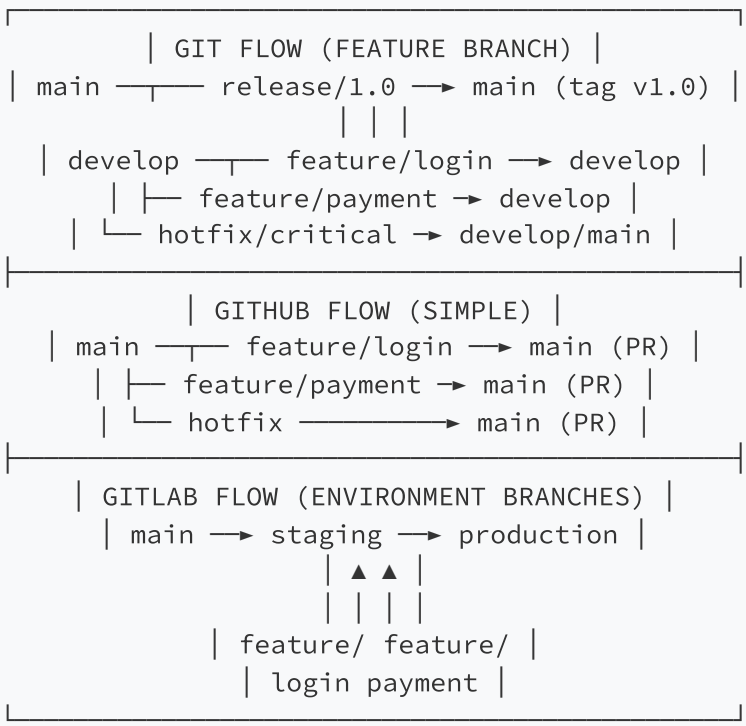
```
 "comment">"comment"># Service Management
"command">systemctl start nginx
"command">systemctl stop nginx
"command">systemctl restart nginx
"command">systemctl reload nginx
"command">systemctl status nginx

"comment">"comment"># Enable/Disable auto-start
"command">systemctl enable nginx
"command">systemctl disable nginx

"comment">"comment"># View logs
"command">journalctl -u nginx -f
"command">journalctl --since """>1 hour ago"

"comment">"comment"># Process Management
"command">ps aux | grep nginx
"command">pstree -p
"command">top -p $(pgrep nginx)

"comment">"comment"># Kill processes
"command">kill -"devops-keyword">15 PID        "comment">"comment"># C
"command">kill -"devops-keyword">9 PID         "comment">"comment"># F
"command">pkill nginx         "comment">"comment"># Kill by "devops-ke
"command">killall nginx       "comment">"comment"># Kill all instances
```

Process States:

- **R (Running):** Currently executing
- **S (Sleeping):** Waiting for event
- **D (Uninterruptible Sleep):** Waiting for I/O
- **Z (Zombie):** Terminated but parent hasn't waited
- **T (Stopped):** Stopped by job control signal

3. Version Control & Git

9 Git Workflow Strategies



Common Git Commands:

```
 "comment">"comment"># Branch Management
"command">git branch feature/login
"command">git checkout feature/login
"command">git checkout -b feature/login  "comment">"comment"># Create

"comment">"comment"># Commit Workflow
"command">git add .
"command">git commit -m "feat: add user authentication"
"command">git push origin feature/login

"comment">"comment"># Merge Strategies
"command">git merge feature/login  "comment">"comment"># Fast-forward
"command">git rebase main          "comment">"comment"># Reapply commi
"command">git cherry-pick abc123   "comment">"comment"># Pick specific

"comment">"comment"># Undo Changes
"command">git reset --soft HEAD~"devops-keyword">1   "comment">"commer
"command">git reset --hard HEAD~"devops-keyword">1   "comment">"commer
"command">git revert abc123         "comment">"comment"># Create undo

"comment">"comment"># Stashing
"command">git stash                 "comment">"comment"># Save uncommi
"command">git stash pop             "comment">"comment"># Restore stas
"command">git stash list            "comment">"comment"># View stashes
```

10 Git Hooks for DevOps Automation

```
 "comment">"comment">#!/bin/bash
"comment">"comment"># .git/hooks/pre-commit - Example pre-commit hook

"comment">"comment"># Colors for output
"devops-keyword">RED="tool-">name">'\033[0;31m'
"devops-keyword">GREEN="tool-">name">'\033[0;32m'
"devops-keyword">YELLOW="tool-">name">'\033[1;33m'
"devops-keyword">NC="tool-">name">'\033[0m' "comment">"comment"># No C

"devops-keyword">echo -e "tool-">name">"${YELLOW}Running pre-commit ch

"comment">"comment"># Check for debug statements
"devops-keyword">DEBUG_FOUND="devops-keyword">"devops-keyword">false
"devops-keyword">for file "devops-keyword">in $(git diff --cached --"d
    "command">if [[ -f "devops-keyword">$file ]]; "command">then
        "command">if grep -n "tool-">name">"console\.log\|debugger\|TO
            "devops-keyword">DEBUG_FOUND="devops-keyword">"devops-keyw
            "devops-keyword">echo -e "tool-">name">"${RED}Found debug
        "command">fi
    "command">fi
"devops-keyword">done

"comment">"comment"># Run linter
"command">if command -v eslint >/dev/"devops-keyword">null "devops-key
    "devops-keyword">echo -e "tool-">name">"${YELLOW}Running ESLint...
    npx eslint --fix $(git diff --cached --"devops-keyword">name-only
    "command">if [[ "devops-keyword">$? -ne "devops-keyword">"devops-k
        "devops-keyword">echo -e "tool-">name">"${RED}ESLint checks fa
        "command">exit "devops-keyword">"devops-keyword">1
    "command">fi
"command">fi

"comment">"comment"># Run tests
"command">if [[ -f package.json ]]; "command">then
    "devops-keyword">echo -e "tool-">name">"${YELLOW}Running unit test
    npm test -- --passWithNoTests
    "command">if [[ "devops-keyword">$? -ne "devops-keyword">"devops-k
        "devops-keyword">echo -e "tool-">name">"${RED}Tests failed${NC
        "command">exit "devops-keyword">"devops-keyword">1
    "command">fi
"command">fi

"comment">"comment"># Check commit message format
"devops-keyword">COMMIT_MSG_FILE="tool-">name">$"devops-keyword">1
"devops-keyword">COMMIT_MSG=$(cat "devops-keyword">$COMMIT_MSG_FILE)

"command">if ! "command">echo "tool-">name">"$COMMIT_MSG" | grep -qE '
```

```
    "devops-keyword">echo -e "tool-">name">"${RED}Invalid commit messa
    "devops-keyword">echo -e "tool-">name">"Expected: ">type(scope): c
    "devops-keyword">echo -e "tool-">name">"Example: feat(auth): add l
    "command">exit "devops-keyword">"devops-keyword">1
"command">fi

"devops-keyword">echo -e "tool-">name">"${GREEN}All pre-commit checks
"command">exit "devops-keyword">"devops-keyword">0
```

Useful Git Hooks for DevOps:

- **pre-commit:** Run linters, tests, security scans
- **prepare-commit-msg:** Enforce commit message format
- **post-commit:** Trigger notifications, update tickets
- **pre-push:** Run integration tests, build verification
- **post-checkout:** Update dependencies, environment setup
- **post-merge:** Update packages, database migrations

4. CI/CD Pipeline

11 Jenkins Pipeline as Code

```
 "comment">// Jenkinsfile - Declarative Pipeline
pipeline {
    agent {
        docker {
            image "tool-">name">'maven:3.8.4-openjdk-11'
            args "tool-">name">'-v $HOME/.m2:/root/.m2'
        }
    }

    environment {
        "devops-keyword">DOCKER_REGISTRY = "tool-">name">'registry.exa
        "devops-keyword">APP_NAME = "tool-">name">'myapp'
        "devops-keyword">VERSION = "devops-keyword">${"devops-keyword'
    }

    options {
        timeout(time: "devops-keyword">1, unit: "tool-">name">'HOURS')
        buildDiscarder(logRotator(numToKeepStr: "tool-">name">'10'))
        disableConcurrentBuilds()
    }

    stages {
        stage("tool-">name">'Checkout') {
            steps {
                checkout scm
                sh "tool-">name">'git log -1 --oneline'
            }
        }

        stage("tool-">name">'Build & Test') {
            steps {
                sh "tool-">name">'mvn clean compile'
                sh "tool-">name">'mvn test'
                junit "tool-">name">'target/surefire-reports/*.xml'
            }
            post {
                always {
                    junit "tool-">name">'target/surefire-reports/*.xml
                }
            }
        }

        stage("tool-">name">'Static Analysis') {
            steps {
                sh "tool-">name">'mvn sonar:sonar'
            }
        }
```

```
        stage("tool-">name">'Build Docker Image') {
            steps {
                script {
                    dockerImage = docker.build("tool-">name">"${DOCKER
                }
            }
        }

        stage("tool-">name">'Security Scan') {
            steps {
                sh "tool-">name">'trivy image ${DOCKER_REGISTRY}/${APP
            }
        }

        stage("tool-">name">'Push to Registry') {
            steps {
                script {
                    docker.withRegistry("tool-">name">'https://${DOCKE
                        dockerImage.push()
                        dockerImage.push("tool-">name">'latest')
                    }
                }
            }
        }

        stage("tool-">name">'Deploy to Staging') {
            when {
                branch "tool-">name">'develop'
            }
            steps {
                sh "tool-">name">'''
                    kubectl set image deployment/${APP_NAME} \
                    ${APP_NAME}=${DOCKER_REGISTRY}/${APP_NAME}:${VERSI
                    -n staging
                '''
            }
        }

        stage(name">'Deploy to Production') {
            when {
                branch name">'main'
            }
            steps {
                input name">message: 'Deploy to production?', ok: 'Dep
                sh name">'''
                    kubectl set image deployment/${APP_NAME} \
                    ${APP_NAME}=${DOCKER_REGISTRY}/${APP_NAME}:${VERSI
                    -n production
```

```
                '''
            }
        }
    }

    post {
        success {
            emailext (
                subject: name">"SUCCESS: Job '${"devops-keyword">env.J
                body: name">'''${SCRIPT, "devops-keyword">template="gr
                to: name">'team@example.com'
            )
        }
        failure {
            emailext (
                subject: name">"FAILED: Job '${"devops-keyword">env.JO
                body: name">'''${SCRIPT, "devops-keyword">template="gr
                to: name">'team@example.com'
            )
            sh name">'kubectl rollout undo deployment/${APP_NAME} -n p
        }
        always {
            cleanWs()
        }
    }
}
```

12 GitHub Actions CI/CD Pipeline

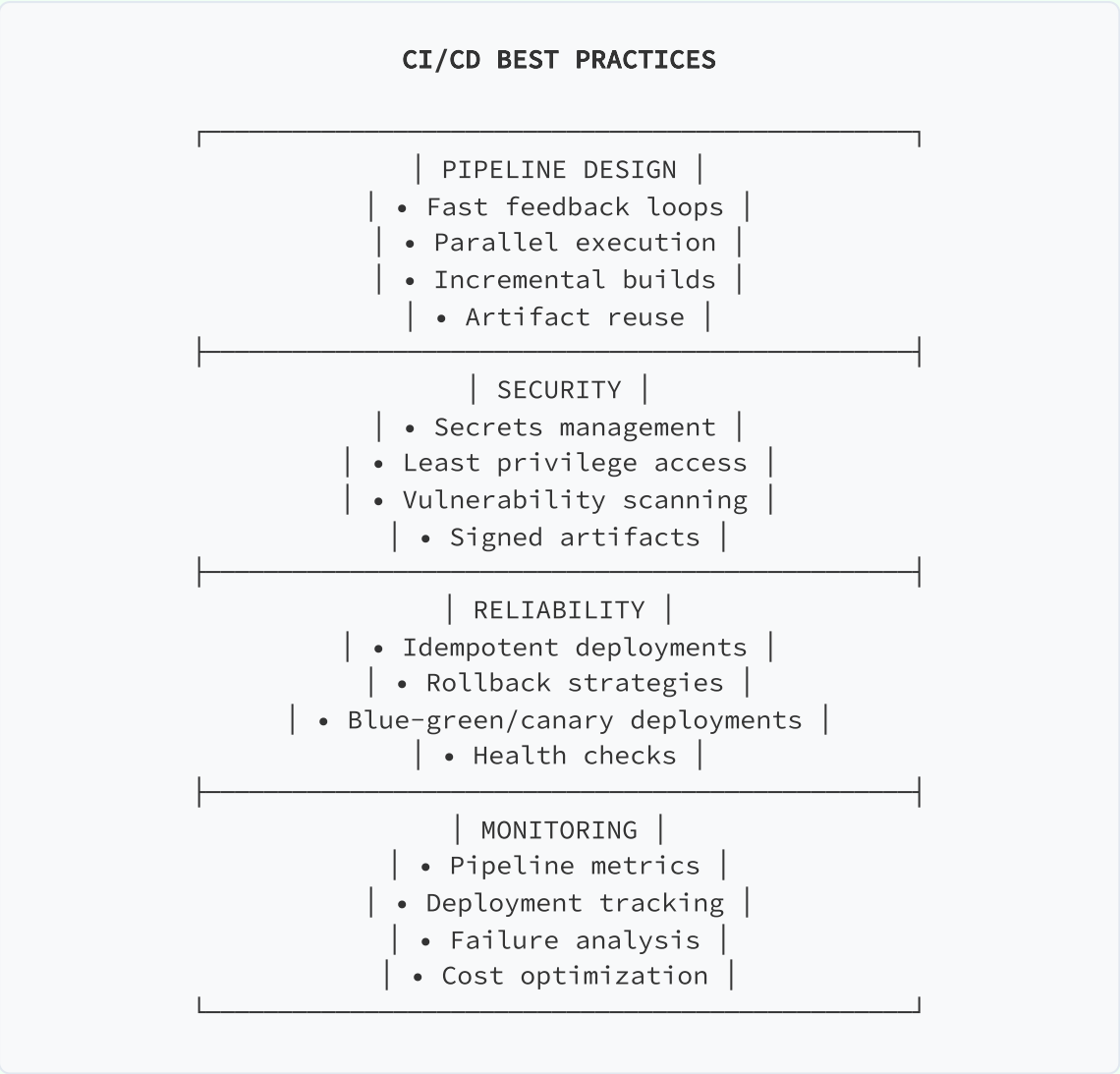
```
 "comment">"comment"># .github/workflows/ci-cd.yml
"devops-keyword">"devops-keyword">name: CI/CD Pipeline

"devops-keyword">on:
  push:
    branches: [ main, develop ]
  pull_request:
    branches: [ main ]
  schedule:
    - cron: "tool-">name">'0 2 * * *'  "comment">"comment"># Daily at

"devops-keyword">"devops-keyword">env:
  REGISTRY: ghcr.io
  IMAGE_NAME: ${{ github.repository }}
  VERSION: ${{ github.sha }}

"devops-keyword">jobs:
  test:
    runs-on: ubuntu-latest
    "devops-keyword">strategy:
      matrix:
        node-"devops-keyword">version: ["tool-">name">'16', "tool-">na

    steps:
    - uses: actions/checkout@v3

    - "devops-keyword">name: Setup Node.js
      uses: actions/setup-node@v3
      with:
        node-"devops-keyword">version: ${{ matrix.node-"devops-keyword
        cache: "tool-">name">'npm'

    - "devops-keyword">name: Install dependencies
      run: npm ci

    - "devops-keyword">name: Run linting
      run: npm run lint

    - "devops-keyword">name: Run tests
      run: npm test
      "devops-keyword">env:
        CI: "devops-keyword">true

    - "devops-keyword">name: Upload coverage
      uses: codecov/codecov-action@v3

  build:
```

```
    needs: test
    runs-on: ubuntu-latest
    if: github.event_name == 'push' && github.ref == 'refs/heads/main'

    steps:
    - uses: actions/checkout@v3

    - "devops-keyword">name: Set up Docker Buildx
      uses: docker/setup-buildx-action@v2

    - "devops-keyword">name: Log in to Container Registry
      uses: docker/login-action@v2
      with:
        registry: ${{ "devops-keyword">env.REGISTRY }}
        username: ${{ github.actor }}
        password: ${{ secrets.GITHUB_TOKEN }}

    - "devops-keyword">name: Extract "devops-keyword">metadata
      id: meta
      uses: docker/"devops-keyword">metadata-action@v4
      with:
        images: ${{ "devops-keyword">env.REGISTRY }}/${{ "devops-keywo
        tags: |
          "devops-keyword">type=sha,prefix={{branch}}-
          "devops-keyword">type=ref,event=branch
          "devops-keyword">type=ref,event=tag
          "devops-keyword">type=raw,value=latest

    - "devops-keyword">name: Build and push
      uses: docker/build-push-action@v4
      with:
        context: .
        push: "devops-keyword">true
        tags: ${{ steps.meta.outputs.tags }}
        "devops-keyword">labels: ${{ steps.meta.outputs."devops-keywor
        cache-from: "devops-keyword">type=gha
        cache-to: "devops-keyword">type=gha,mode=max

  security-scan:
    needs: build
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v3

    - "devops-keyword">name: Run Trivy vulnerability scanner
      uses: aquasecurity/trivy-action@master
      with:
        image-ref: "tool-">name">'${{ env.REGISTRY }}/${{ env.IMAGE_NA
```

```
        format: "tool-">name">'table'
        exit-code: "tool-">name">'1'
        severity: "tool-">name">'CRITICAL,HIGH'

    - "devops-keyword">name: Run Snyk security scan
      uses: snyk/actions/node@master
      "devops-keyword">env:
        SNYK_TOKEN: ${{ secrets.SNYK_TOKEN }}
      with:
        args: --severity-threshold=high

  deploy-staging:
    needs: [test, security-scan]
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/develop'
    environment:
      "devops-keyword">name: staging
      url: https://staging.example.com

    steps:
    - uses: actions/checkout@v3

    - "devops-keyword">name: Configure AWS credentials
      uses: aws-actions/configure-aws-credentials@v1
      with:
        aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
        aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
        aws-region: us-east-"devops-keyword">1

    - "devops-keyword">name: Deploy to EKS
      run: |
        aws eks update-kubeconfig --"devops-keyword">name my-cluster -
        kubectl set image deployment/myapp \
          myapp=${{ "devops-keyword">env.REGISTRY }}/${{ "devops-keywo
          -n staging
        kubectl rollout status deployment/myapp -n staging

  deploy-production:
    needs: [test, security-scan, deploy-staging]
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/main'
    environment:
      "devops-keyword">name: production
      url: https://example.com

    steps:
    - uses: actions/checkout@v3

    - "devops-keyword">name: Manual approval
```

```
      uses: trstringer/manual-approval@v1
      with:
        secret: ${{ github.TOKEN }}
        approvers: ${{ secrets.APPROVERS }}
        minimum-approvals: "devops-keyword">1

    - "devops-keyword">name: Configure AWS credentials
      uses: aws-actions/configure-aws-credentials@v1
      with:
        aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
        aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
        aws-region: us-east-"devops-keyword">1

    - "devops-keyword">name: Deploy to EKS
      run: |
        aws eks update-kubeconfig --"devops-keyword">name my-cluster -
        kubectl set image deployment/myapp \
          myapp=${{ "devops-keyword">env.REGISTRY }}/${{ "devops-keywo
          -n production
        kubectl rollout status deployment/myapp -n production

    - "devops-keyword">name: Smoke test
      run: |
        curl -f https://example.com/health
        sleep "devops-keyword">30
        curl -f https://example.com/api/status

  notify:
    needs: [deploy-staging, deploy-production]
    if: always()
    runs-on: ubuntu-latest

    steps:
    - "devops-keyword">name: Send Slack notification
      uses: 8398a7/action-slack@v3
      with:
        status: ${{ job.status }}
        author_name: GitHub Actions
        fields: repo,message,commit,author,action,eventName,ref,workfl
      "devops-keyword">env:
        SLACK_WEBHOOK_URL: ${{ secrets.SLACK_WEBHOOK_URL }}
```

13 CI/CD Best Practices



Key Practices:

1. **Keep Builds Fast:**
   - Use incremental builds
   - Implement caching (dependencies, Docker layers)
   - Run tests in parallel
   - Use build farms for large projects
2. **Security First:**
   - Scan for vulnerabilities in dependencies and containers
   - Use secret management (HashiCorp Vault, AWS Secrets Manager)
   - Implement SAST (Static Application Security Testing)
   - Enforce least privilege for pipeline agents
3. **Reliable Deployments:**
   - Implement rollback strategies
   - Use canary or blue-green deployments
   - Perform health checks post-deployment
   - Monitor key metrics during deployment
4. **Infrastructure as Code:**
   - Version control pipeline definitions

- Use parameterized pipelines
- Implement pipeline templates
- Automate environment provisioning

5. **Monitoring & Observability:**
   - Track pipeline success/failure rates
   - Monitor build times and resource usage
   - Set up alerts for pipeline failures
   - Implement distributed tracing

5. Containerization - Docker

14 Dockerfile Best Practices

```
 "comment">"comment"># Multi-stage Dockerfile for Node.js application
"comment">"comment"># Stage "devops-keyword">1: Build
"tool-">name">FROM node:"devops-keyword">18-alpine AS builder
"tool-">name">WORKDIR /app

"comment">"comment"># Install dependencies separately for caching
"tool-">name">COPY package*.json ./
"tool-">name">RUN npm ci --only=production

"comment">"comment"># Copy source and build
"tool-">name">COPY . .
"tool-">name">RUN npm run build

"comment">"comment"># Stage "devops-keyword">2: Production
"tool-">name">FROM node:"devops-keyword">18-alpine AS production
"tool-">name">WORKDIR /app

"comment">"comment"># Create non-root user
"tool-">name">RUN addgroup -g "devops-keyword">1001 -S nodejs \
    && adduser -S nodejs -u "devops-keyword">1001

"comment">"comment"># Copy built artifacts from builder
"tool-">name">COPY --from=builder --chown=nodejs:nodejs /app/package*.
"tool-">name">COPY --from=builder --chown=nodejs:nodejs /app/node_modu
"tool-">name">COPY --from=builder --chown=nodejs:nodejs /app/dist ./di

"comment">"comment"># Switch to non-root user
"tool-">name">USER nodejs

"comment">"comment"># Health check
"tool-">name">HEALTHCHECK --interval=30s --timeout=3s --start-period=5
    CMD node -e "require('http').get('http://localhost:3000/health', (
    {if(r.statusCode!=="devops-keyword">200)throw new Error()})"

">"># Environment variables
"devops-keyword">name">ENV NODE_ENV=production \
    PORT="devops-keyword">3000

"comment">"comment"># Expose port
"tool-">name">EXPOSE "devops-keyword">3000

"comment">"comment"># Run application
"tool-">name">CMD ["node", "dist/index.js"]

"comment">"comment"># Labels for "devops-keyword">metadata
"tool-">name">LABEL maintainer="devops@example.com" \
```

```
        "devops-keyword">version=""">1."">0."">0" \
        description="Node.js application"
```

Dockerfile Best Practices:

1. **Use Official Base Images:** Alpine variants for smaller size
2. **Multi-stage Builds:** Separate build and runtime environments
3. **Minimize Layers:** Combine RUN commands, clean up in same layer
4. **Non-root User:** Run containers as non-root for security
5. **Health Checks:** Implement container health monitoring
6. **.dockerignore:** Exclude unnecessary files from build context
7. **Tag Versioning:** Use specific version tags, not just "latest"
8. **Environment Variables:** Use ENV for configuration
9. **Resource Limits:** Set memory and CPU limits in deployment

.dockerignore Example:

```
 "comment">"comment"># .dockerignore
"tool-">name">node_modules
npm-debug.log
.DS_Store
.git
.gitignore
."devops-keyword">env
.dockerignore
README.md
.vscode
.idea
coverage
dist
*.log
*.tmp
*.swp
Dockerfile*
docker-compose*.yml
build/
test/
tests/
```

15 Docker Compose for Local Development

```
 "comment">"comment"># docker-compose.yml - Multi-service application
"devops-keyword">"devops-keyword">version: "tool-">name">'3.8'

"devops-keyword">services:
  "devops-keyword">app:
    "devops-keyword">build:
      context: .
      dockerfile: Dockerfile.dev
    "devops-keyword">container_name: "tool-">name">webapp
    "devops-keyword">"devops-keyword">ports:
      - "tool-">name">"">3000:">3000"
    "devops-keyword">environment:
      - "tool-">name">NODE_ENV=development
      - "tool-">name">DATABASE_URL=postgresql://user:pass@db:"devops-k
      - "tool-">name">REDIS_URL=redis://redis:"devops-keyword">6379
    "devops-keyword">"devops-keyword">volumes:
      - "tool-">name">.:/app
      - "tool-">name">/app/node_modules
    "devops-keyword">depends_on:
      - "tool-">name">db
      - "tool-">name">redis
    "devops-keyword">networks:
      - "tool-">name">app-network
    "devops-keyword">healthcheck:
      test: ["tool-">name">"CMD", "tool-">name">"curl", "tool-">name">
      interval: 30s
      timeout: 10s
      retries: "devops-keyword">3
      start_period: 40s

  "devops-keyword">db:
    "devops-keyword">image: "tool-">name">postgres:"devops-keyword">15
    "devops-keyword">container_name: "tool-">name">postgres-db
    "devops-keyword">environment:
      - "tool-">name">POSTGRES_USER=user
      - "tool-">name">POSTGRES_PASSWORD=pass
      - "tool-">name">POSTGRES_DB=appdb
    "devops-keyword">"devops-keyword">ports:
      - "tool-">name">"">5432:">5432"
    "devops-keyword">"devops-keyword">volumes:
      - "tool-">name">postgres_data:/var/lib/postgresql/"devops-keywor
      - "tool-">name">./init.sql:/docker-entrypoint-initdb.d/init.sql
    "devops-keyword">networks:
      - "tool-">name">app-network
    "devops-keyword">healthcheck:
      test: ["tool-">name">"CMD-SHELL", "tool-">name">"pg_isready -U u
      interval: 10s
```

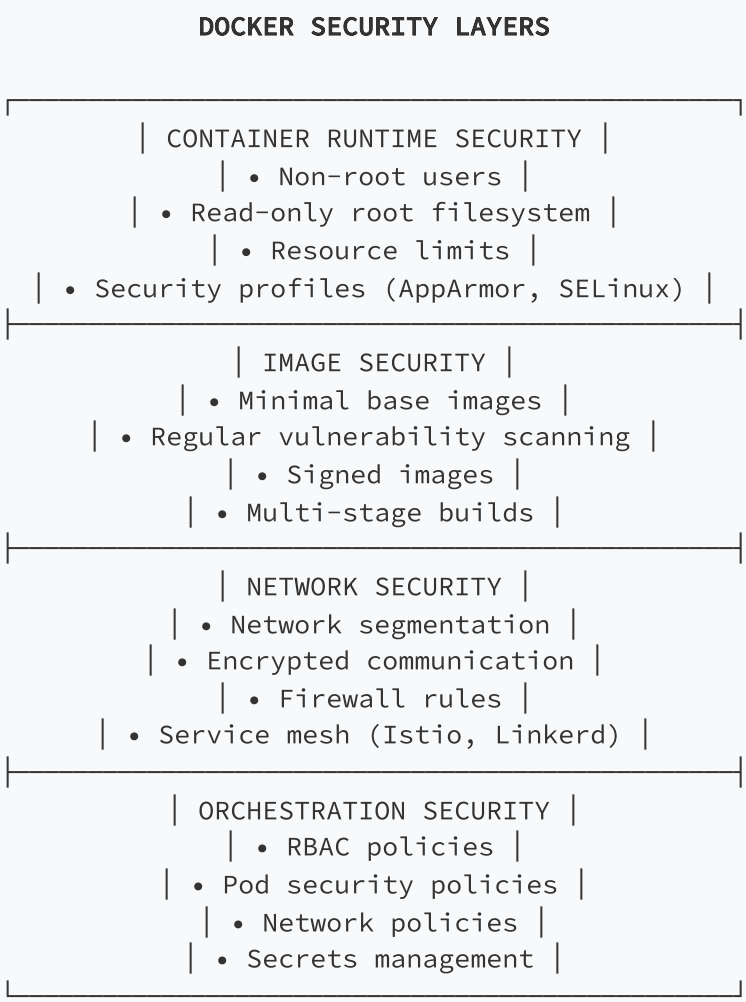
```
      timeout: 5s
      retries: "devops-keyword">5

  "devops-keyword">redis:
    "devops-keyword">image: "tool-">name">redis:"devops-keyword">7-alp
    "devops-keyword">container_name: "tool-">name">redis-cache
    "devops-keyword">"devops-keyword">ports:
      - "tool-">name">""">6379:">6379"
    "devops-keyword">command: redis-server --appendonly yes
    "devops-keyword">"devops-keyword">volumes:
      - "tool-">name">redis_data:/"devops-keyword">data
    "devops-keyword">networks:
      - "tool-">name">app-network
    "devops-keyword">healthcheck:
      test: ["tool-">name">"CMD", "tool-">name">"redis-cli", "tool-">r
      interval: 10s
      timeout: 5s
      retries: "devops-keyword">5

  "devops-keyword">nginx:
    "devops-keyword">image: "tool-">name">nginx:alpine
    "devops-keyword">container_name: "tool-">name">nginx-proxy
    "devops-keyword">"devops-keyword">ports:
      - "tool-">name">""">80:">80"
      - "tool-">name">""">443:">443"
    "devops-keyword">"devops-keyword">volumes:
      - "tool-">name">./nginx.conf:/etc/nginx/nginx.conf:ro
      - "tool-">name">./ssl:/etc/nginx/ssl:ro
    "devops-keyword">depends_on:
      - "tool-">name">app
    "devops-keyword">networks:
      - "tool-">name">app-network

  "devops-keyword">monitoring:
    "devops-keyword">image: "tool-">name">grafana/grafana:latest
    "devops-keyword">container_name: "tool-">name">grafana
    "devops-keyword">"devops-keyword">ports:
      - "tool-">name">""">3001:">3000"
    "devops-keyword">environment:
      - "tool-">name">GF_SECURITY_ADMIN_PASSWORD=admin
    "devops-keyword">"devops-keyword">volumes:
      - "tool-">name">grafana_data:/var/lib/grafana
    "devops-keyword">networks:
      - "tool-">name">app-network

"devops-keyword">"devops-keyword">volumes:
  "devops-keyword">postgres_data:
  "devops-keyword">redis_data:
  "devops-keyword">grafana_data:
```

```
"devops-keyword">networks:
  "devops-keyword">app-network:
    "devops-keyword">driver: "tool-">name">bridge
```

Docker Compose Commands:

```
 "comment">"comment"># Start services
"command">docker-compose up -d
"command">docker-compose up --build  "comment">"comment"># Rebuild ima

"comment">"comment"># View logs
"command">docker-compose logs -f
"command">docker-compose logs -f app

"comment">"comment"># Stop services
"command">docker-compose down
"command">docker-compose down -v  "comment">"comment"># Remove "devops

"comment">"comment"># Service management
"command">docker-compose ps
"command">docker-compose restart app
"command">docker-compose exec app bash

"comment">"comment"># Scale services
"command">docker-compose up --scale app="devops-keyword">3 -d

"comment">"comment"># Environment-specific files
"command">docker-compose -f docker-compose.yml -f docker-compose.prod.
```

16 Docker Security Best Practices



Security Practices:

```
"comment">"comment"># Docker run with security options
"command">docker run \
  --user "devops-keyword">1001:"devops-keyword">1001 \
  --read-only \                          "comment">"comment"># Read-onl
  --tmpfs /tmp \                         "comment">"comment"># Writable
  --memory 512m \                        "comment">"comment"># Memory l
  --cpus "devops-keyword">1 \                                "comment">"co
  --security-opt no-new-privileges \     "comment">"comment"># No privi
  --cap-drop ALL \                       "comment">"comment"># Drop all
  --cap-add NET_BIND_SERVICE \           "comment">"comment"># Add only
  nginx:alpine

"comment">"comment"># Docker Compose with security
"devops-keyword">services:
  "devops-keyword">app:
    "devops-keyword">security_opt:
      - no-new-privileges:"devops-keyword">true
    "devops-keyword">cap_drop:
      - ALL
    "devops-keyword">cap_add:
      - NET_BIND_SERVICE
    "devops-keyword">read_only: "devops-keyword">true
    "devops-keyword">tmpfs:
      - /tmp
    "devops-keyword">user: "tool-">name">"">1001:">1001"
```

Security Tools:

- **Image Scanning:** Trivy, Clair, Docker Scout, Snyk
- **Runtime Security:** Falco, Aqua Security, Sysdig
- **Secret Scanning:** GitLeaks, TruffleHog
- **Compliance:** CIS Docker Benchmarks, OpenSCAP

> **Security Checklist:** 1. Use minimal base images (Alpine, Distroless)
> 2. Run as non-root user
> 3. Scan images for vulnerabilities
> 4. Sign and verify images
> 5. Implement network policies
> 6. Use secrets management
> 7. Regular updates and patching
> 8. Monitor runtime behavior

6. Orchestration - Kubernetes

17 Kubernetes Architecture Components

KUBERNETES ARCHITECTURE

Control Plane (Master Node):

- **API Server:** Kubernetes frontend, handles REST operations
- **etcd:** Distributed key-value store for cluster data
- **Scheduler:** Assigns pods to nodes based on resources
- **Controller Manager:** Runs controller processes
- **Cloud Controller Manager:** Cloud-specific control logic

Worker Nodes:

- **Kubelet:** Agent that runs on each node
- **Kube-proxy:** Network proxy maintaining network rules
- **Container Runtime:** Docker, containerd, CRI-O

Addons:

- **DNS:** CoreDNS for service discovery
- **Dashboard:** Web-based UI
- **Ingress Controller:** NGINX, Traefik, HAProxy
- **Network Plugin:** Calico, Flannel, Weave Net

Kubernetes Objects:

- **Pod:** Smallest deployable unit, one or more containers
- **Deployment:** Manages ReplicaSets, enables rolling updates
- **Service:** Abstract way to expose applications
- **ConfigMap:** Configuration data in key-value pairs
- **Secret:** Sensitive configuration data
- **Volume:** Storage attached to pods
- **Namespace:** Virtual cluster within a physical cluster
- **Ingress:** Manages external access to services

18 Kubernetes Deployment YAML Examples

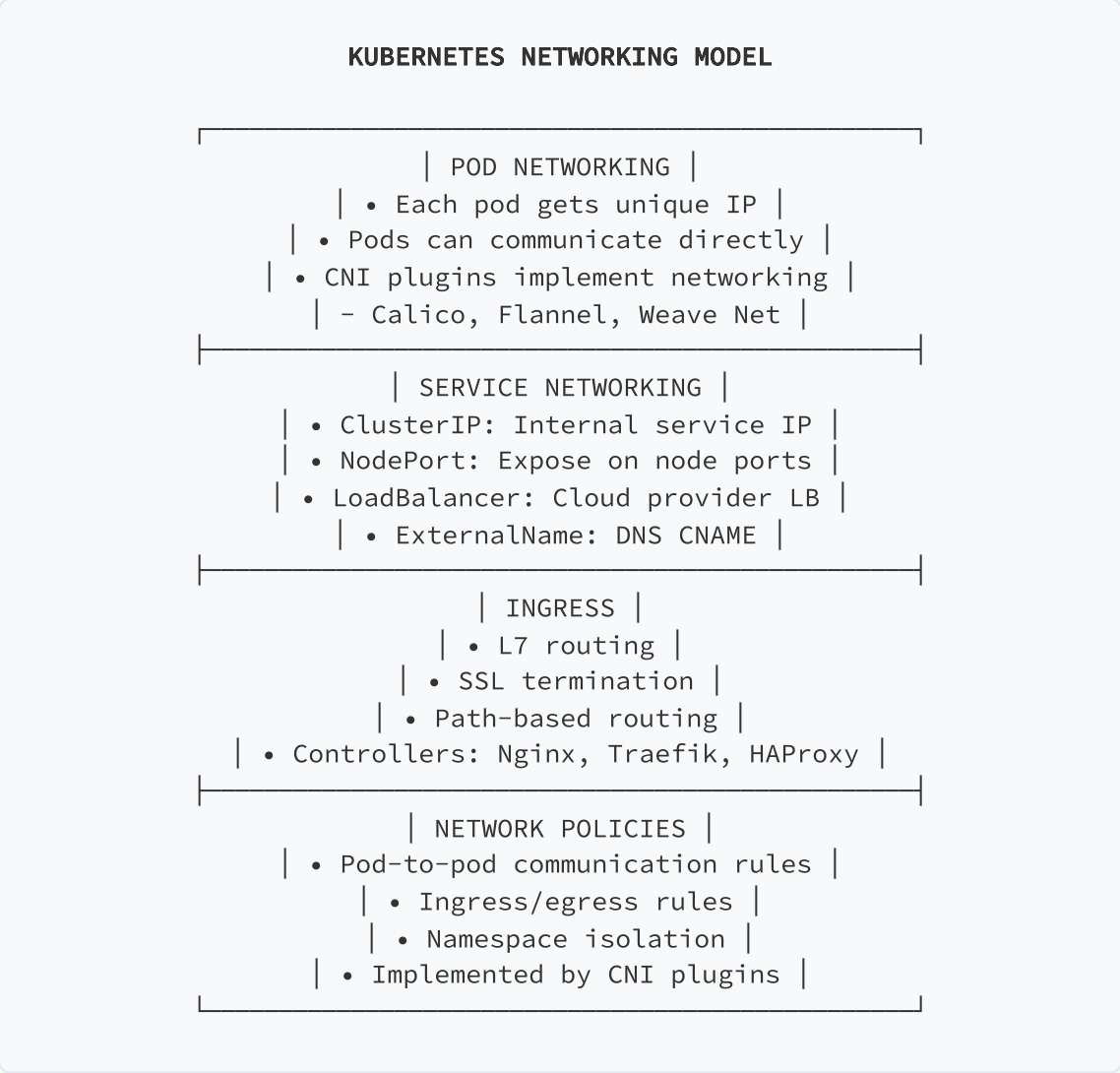
```
 "comment">"comment"># deployment.yaml - Web application deployment
"devops-keyword">"devops-keyword">apiVersion: apps/v1
"devops-keyword">"devops-keyword">kind: Deployment
"devops-keyword">"devops-keyword">metadata:
  "devops-keyword">"devops-keyword">name: webapp
  "devops-keyword">"devops-keyword">namespace: production
  "devops-keyword">"devops-keyword">labels:
    app: webapp
    environment: production
"devops-keyword">"devops-keyword">spec:
  "devops-keyword">"devops-keyword">replicas: "devops-keyword">3
  "devops-keyword">"devops-keyword">strategy:
    "devops-keyword">"devops-keyword">type: RollingUpdate
    "devops-keyword">rollingUpdate:
      maxSurge: "devops-keyword">1
      maxUnavailable: "devops-keyword">0
  "devops-keyword">"devops-keyword">selector:
    "devops-keyword">"devops-keyword">matchLabels:
      app: webapp
  "devops-keyword">"devops-keyword">template:
    "devops-keyword">"devops-keyword">metadata:
      "devops-keyword">"devops-keyword">labels:
        app: webapp
        "devops-keyword">version: v1."devops-keyword">2."devops-keywor
    "devops-keyword">"devops-keyword">spec:
      "devops-keyword">serviceAccountName: webapp-sa
      "devops-keyword">securityContext:
        runAsUser: "devops-keyword">1000
        runAsGroup: "devops-keyword">1000
        fsGroup: "devops-keyword">1000
      "devops-keyword">"devops-keyword">containers:
      - "devops-keyword">"devops-keyword">name: webapp
        "devops-keyword">image: registry.example.com/webapp:v1."devops
        "devops-keyword">imagePullPolicy: IfNotPresent
        "devops-keyword">"devops-keyword">ports:
        - "devops-keyword">containerPort: "devops-keyword">8080
          "devops-keyword">"devops-keyword">name: http
        "devops-keyword">"devops-keyword">env:
        - "devops-keyword">"devops-keyword">name: NODE_ENV
          "devops-keyword">value: production
        - "devops-keyword">"devops-keyword">name: DATABASE_URL
          "devops-keyword">valueFrom:
            "devops-keyword">secretKeyRef:
              "devops-keyword">name: db-credentials
              key: url
        "devops-keyword">"devops-keyword">resources:
          "devops-keyword">requests:
```

```
            memory: "256Mi"
            cpu: "250m"
          "devops-keyword">limits:
            memory: "512Mi"
            cpu: "500m"
        "devops-keyword">livenessProbe:
          httpGet:
            path: /health
            port: "devops-keyword">8080
          initialDelaySeconds: "devops-keyword">30
          periodSeconds: "devops-keyword">10
          timeoutSeconds: "devops-keyword">5
          failureThreshold: "devops-keyword">3
        "devops-keyword">readinessProbe:
          httpGet:
            path: /ready
            port: "devops-keyword">8080
          initialDelaySeconds: "devops-keyword">5
          periodSeconds: "devops-keyword">5
          timeoutSeconds: "devops-keyword">3
          failureThreshold: "devops-keyword">1
        "devops-keyword">volumeMounts:
        - "devops-keyword">name: config-volume
          mountPath: /app/config
      "devops-keyword">"devops-keyword">volumes:
      - "devops-keyword">name: config-volume
        configMap:
          "devops-keyword">name: app-config
      "devops-keyword">"devops-keyword">affinity:
        "devops-keyword">podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: "devops-keyword">100
            podAffinityTerm:
              labelSelector:
                "devops-keyword">matchExpressions:
                - key: app
                  operator: In
                  values:
                  - webapp
              topologyKey: kubernetes.io/hostname
      "devops-keyword">"devops-keyword">tolerations:
      - key: "node-">type"
        operator: "Equal"
        value: "spot"
        effect: "NoSchedule"
---
"comment">"comment"># service.yaml - Expose the deployment
"devops-keyword">"devops-keyword">apiVersion: v1
"devops-keyword">"devops-keyword">kind: Service
```

```
"devops-keyword">"devops-keyword">metadata:
  "devops-keyword">"devops-keyword">name: webapp-service
  "devops-keyword">"devops-keyword">namespace: production
"devops-keyword">"devops-keyword">spec:
  "devops-keyword">"devops-keyword">selector:
    app: webapp
  "devops-keyword">"devops-keyword">ports:
  - "devops-keyword">port: "devops-keyword">80
    "devops-keyword">targetPort: "devops-keyword">8080
    "devops-keyword">"devops-keyword">name: http
  "devops-keyword">"devops-keyword">type: ClusterIP
---
"comment">"comment"># ingress.yaml - External access
"devops-keyword">"devops-keyword">apiVersion: networking.k8s.io/v1
"devops-keyword">"devops-keyword">kind: Ingress
"devops-keyword">"devops-keyword">metadata:
  "devops-keyword">"devops-keyword">name: webapp-ingress
  "devops-keyword">"devops-keyword">namespace: production
  "devops-keyword">annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    cert-manager.io/cluster-issuer: letsencrypt-prod
"devops-keyword">"devops-keyword">spec:
  "devops-keyword">tls:
  - "devops-keyword">hosts:
    - app.example.com
    "devops-keyword">secretName: webapp-tls
  "devops-keyword">rules:
  - "devops-keyword">host: app.example.com
    "devops-keyword">http:
      "devops-keyword">paths:
      - "devops-keyword">path: /
        "devops-keyword">pathType: Prefix
        "devops-keyword">backend:
          "devops-keyword">service:
            "devops-keyword">"devops-keyword">name: webapp-service
            "devops-keyword">port:
              number: "devops-keyword">80
---
"comment">"comment"># hpa.yaml - Horizontal Pod Autoscaler
"devops-keyword">"devops-keyword">apiVersion: autoscaling/v2
"devops-keyword">"devops-keyword">kind: HorizontalPodAutoscaler
"devops-keyword">"devops-keyword">metadata:
  "devops-keyword">"devops-keyword">name: webapp-hpa
  "devops-keyword">"devops-keyword">namespace: production
"devops-keyword">"devops-keyword">spec:
  "devops-keyword">scaleTargetRef:
    "devops-keyword">"devops-keyword">apiVersion: apps/v1
    "devops-keyword">"devops-keyword">kind: Deployment
    "devops-keyword">"devops-keyword">name: webapp
```

```
  "devops-keyword">minReplicas: "devops-keyword">2
  "devops-keyword">maxReplicas: "devops-keyword">10
  "devops-keyword">metrics:
  - "devops-keyword">"devops-keyword">type: Resource
    "devops-keyword">resource:
      "devops-keyword">"devops-keyword">name: cpu
      "devops-keyword">target:
        "devops-keyword">"devops-keyword">type: Utilization
        "devops-keyword">averageUtilization: "devops-keyword">70
  - "devops-keyword">"devops-keyword">type: Resource
    "devops-keyword">resource:
      "devops-keyword">"devops-keyword">name: memory
      "devops-keyword">target:
        "devops-keyword">"devops-keyword">type: Utilization
        "devops-keyword">averageUtilization: "devops-keyword">80
```

19 Kubernetes Networking Concepts



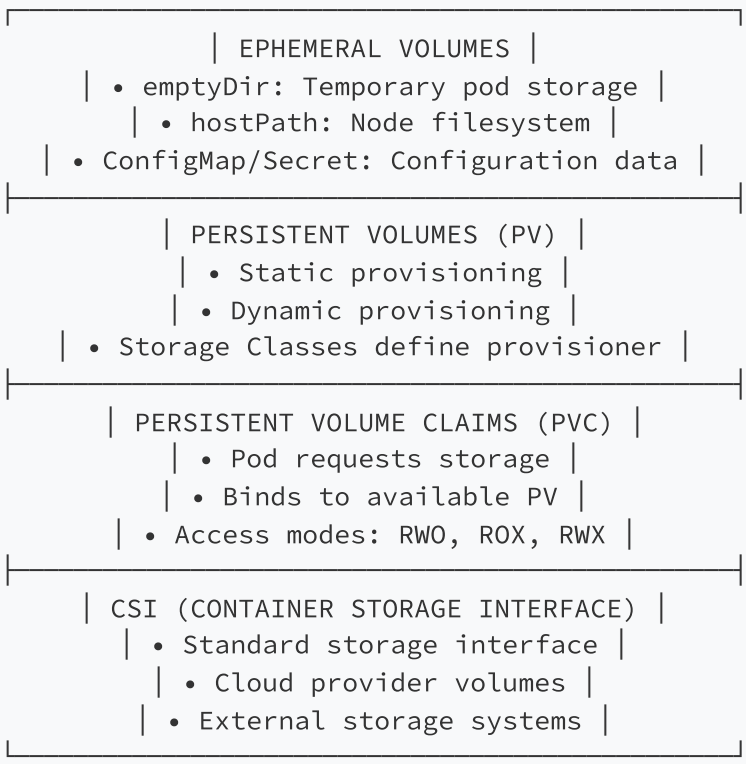
Service Types:

1. **ClusterIP:** Internal IP, accessible within cluster
2. **NodePort:** Exposes service on each node's IP at static port
3. **LoadBalancer:** Creates external load balancer in cloud
4. **ExternalName:** Maps service to DNS name

Network Policy Example:

```
 "devops-keyword">"devops-keyword">apiVersion: networking.k8s.io/v1
"devops-keyword">"devops-keyword">kind: NetworkPolicy
"devops-keyword">"devops-keyword">metadata:
  "devops-keyword">"devops-keyword">name: webapp-network-policy
  "devops-keyword">"devops-keyword">namespace: production
"devops-keyword">"devops-keyword">spec:
  "devops-keyword">podSelector:
    "devops-keyword">"devops-keyword">matchLabels:
      app: webapp
  "devops-keyword">policyTypes:
  - Ingress
  - Egress
  "devops-keyword">ingress:
  - "devops-keyword">from:
    - "devops-keyword">podSelector:
        "devops-keyword">"devops-keyword">matchLabels:
          app: ingress-controller
    - "devops-keyword">namespaceSelector:
        "devops-keyword">"devops-keyword">matchLabels:
          "devops-keyword">name: monitoring
    "devops-keyword">"devops-keyword">ports:
    - "devops-keyword">protocol: TCP
      "devops-keyword">port: "devops-keyword">8080
  "devops-keyword">egress:
  - "devops-keyword">to:
    - "devops-keyword">podSelector:
        "devops-keyword">"devops-keyword">matchLabels:
          app: database
    "devops-keyword">"devops-keyword">ports:
    - "devops-keyword">protocol: TCP
      "devops-keyword">port: "devops-keyword">5432
```

20 Kubernetes Storage & Volumes



Storage Examples:

```
 "comment">"comment"># Storage Class for AWS EBS
"devops-keyword">"devops-keyword">apiVersion: storage.k8s.io/v1
"devops-keyword">"devops-keyword">kind: StorageClass
"devops-keyword">"devops-keyword">metadata:
  "devops-keyword">"devops-keyword">name: fast-ebs
"devops-keyword">provisioner: kubernetes.io/aws-ebs
"devops-keyword">parameters:
  "devops-keyword">type: gp3
  fsType: ext4
"devops-keyword">reclaimPolicy: Retain
"devops-keyword">allowVolumeExpansion: "devops-keyword">true
"devops-keyword">volumeBindingMode: WaitForFirstConsumer
---
"comment">"comment"># Persistent Volume Claim
"devops-keyword">"devops-keyword">apiVersion: v1
"devops-keyword">"devops-keyword">kind: PersistentVolumeClaim
"devops-keyword">"devops-keyword">metadata:
  "devops-keyword">"devops-keyword">name: app-"devops-keyword">data-pv
  "devops-keyword">"devops-keyword">namespace: production
"devops-keyword">"devops-keyword">spec:
  "devops-keyword">accessModes:
    - ReadWriteOnce
  "devops-keyword">storageClassName: fast-ebs
  "devops-keyword">"devops-keyword">resources:
    "devops-keyword">requests:
      storage: 10Gi
---
"comment">"comment"># Pod using PVC
"devops-keyword">"devops-keyword">apiVersion: v1
"devops-keyword">"devops-keyword">kind: Pod
"devops-keyword">"devops-keyword">metadata:
  "devops-keyword">"devops-keyword">name: app-with-storage
"devops-keyword">"devops-keyword">spec:
  "devops-keyword">"devops-keyword">containers:
  - "devops-keyword">"devops-keyword">name: app
    "devops-keyword">image: nginx
    "devops-keyword">volumeMounts:
    - "devops-keyword">"devops-keyword">name: app-storage
      "devops-keyword">mountPath: /var/www/html
  "devops-keyword">"devops-keyword">volumes:
  - "devops-keyword">"devops-keyword">name: app-storage
    "devops-keyword">persistentVolumeClaim:
      "devops-keyword">claimName: app-"devops-keyword">data-pvc
```

7. Infrastructure as Code

21 Terraform Best Practices & Modules

TERRAFORM PROJECT STRUCTURE

```
terraform-project/
├── modules/
│   ├── networking/
│   │   ├── main.tf
│   │   ├── variables.tf
│   │   └── outputs.tf
│   ├── compute/
│   └── database/
├── environments/
│   ├── dev/
│   │   ├── main.tf
│   │   ├── terraform.tfvars
│   │   └── backend.tf
│   ├── staging/
│   └── production/
├── scripts/
├── .terraform-version
└── README.md
```

Terraform Module Example:

```
 "comment">"comment"># modules/ec2-instance/main.tf
"devops-keyword">resource "aws_instance" "this" {
  count = var.instance_count

  ami                    = var.ami_id
  instance_type          = var.instance_type
  subnet_id              = var.subnet_id
  vpc_security_group_ids = var.security_group_ids

  key_name = var.key_name

  root_block_device {
    volume_size = var.root_volume_size
    volume_type = var.root_volume_type
    encrypted   = "devops-keyword">true
  }

  tags = merge(
    var.tags,
    {
      Name = format("%s-%02d", var.name_prefix, count.index + "devops-
    }
  )

  lifecycle {
    create_before_destroy = "devops-keyword">true
    ignore_changes = [ami]
  }
}

"devops-keyword">resource "aws_eip" "this" {
  count = var.assign_eip ? var.instance_count : "devops-keyword">0

  instance = aws_instance.this[count.index].id
  vpc      = "devops-keyword">true

  tags = merge(
    var.tags,
    {
      Name = format("eip-%s-%02d", var.name_prefix, count.index + "dev
    }
  )
}

"comment">"comment"># modules/ec2-instance/variables.tf
"devops-keyword">variable "instance_count" {
  description = "Number of instances to create"
  "devops-keyword">type        = number
```

```
  default     = "devops-keyword">1
}

"devops-keyword">variable "ami_id" {
  description = "AMI ID to use for instances"
  "devops-keyword">type        = string
}

"devops-keyword">variable "instance_type" {
  description = "Instance ">type"
  "devops-keyword">type        = string
  default     = "t3.micro"
}

"devops-keyword">variable "name_prefix" {
  description = "Prefix for resource names"
  "devops-keyword">type        = string
}

"comment">"comment"># modules/ec2-instance/outputs.tf
"devops-keyword">output "instance_ids" {
  description = "List of instance IDs"
  value       = aws_instance.this[*].id
}

"devops-keyword">output "private_ips" {
  description = "List of private IP addresses"
  value       = aws_instance.this[*].private_ip
}

"devops-keyword">output "public_ips" {
  description = "List of public IP addresses"
  value       = aws_eip.this[*].public_ip
}
```

Terraform Commands:

```
 "comment">"comment"># Initialize Terraform
"command">terraform init
"command">terraform init -upgrade

"comment">"comment"># Validate configuration
"command">terraform validate

"comment">"comment"># Plan changes
"command">terraform plan -out=tfplan
"command">terraform plan -var-file=production.tfvars

"comment">"comment"># Apply changes
"command">terraform apply tfplan
"command">terraform apply -auto-approve

"comment">"comment"># Destroy infrastructure
"command">terraform destroy

"comment">"comment"># State management
"command">terraform state list
"command">terraform state show aws_instance.web
"command">terraform state mv aws_instance.old aws_instance.new

"comment">"comment"># Workspace management
"command">terraform workspace new dev
"command">terraform workspace select prod
"command">terraform workspace list
```

22 Configuration Management: Ansible vs Puppet vs Chef

Feature	Ansible	Puppet	Chef
Architecture	Agentless, push-based	Agent-based, pull-based	Agent-based, pull-based
Language	YAML (declarative)	Domain-specific language	Ruby DSL
Learning Curve	Low	Medium	Medium-High
Master Required	No	Yes	Yes
Idempotency	Yes	Yes	Yes
Use Case	Orchestration, config management	Complex infrastructures	Complex infrastructures

Ansible Playbook Example:

```
 "comment">"comment"># webserver.yml - Configure Nginx web server
---
- "devops-keyword">"devops-keyword">name: Configure Nginx web server
  "devops-keyword">hosts: webservers
  "devops-keyword">become: yes
  "devops-keyword">vars:
    nginx_version: "devops-keyword">1."devops-keyword">20."devops-keyw
    app_name: myapp
    domain: example.com

  "devops-keyword">tasks:
  - "devops-keyword">"devops-keyword">name: Install Nginx
    "devops-keyword">apt:
      "devops-keyword">name: nginx
      state: present
      update_cache: yes

  - "devops-keyword">"devops-keyword">name: Create web directory
    "devops-keyword">file:
      path: "/var/www/{{ app_name }}"
      state: directory
      owner: www-"devops-keyword">data
      group: www-"devops-keyword">data
      mode: '0755'

  - "devops-keyword">"devops-keyword">name: Deploy application files
    "devops-keyword">copy:
      src: "{{ item }}"
      dest: "/var/www/{{ app_name }}/"
      owner: www-"devops-keyword">data
      group: www-"devops-keyword">data
    loop:
      - index.html
      - style.css
      - app.js

  - "devops-keyword">"devops-keyword">name: Configure Nginx site
    "devops-keyword">"devops-keyword">template:
      src: nginx.conf.j2
      dest: "/etc/nginx/sites-available/{{ app_name }}"
      owner: root
      group: root
      mode: '0644'
    notify: Restart Nginx

  - "devops-keyword">"devops-keyword">name: Enable site
    "devops-keyword">file:
      src: "/etc/nginx/sites-available/{{ app_name }}"
```

```
      dest: "/etc/nginx/sites-enabled/{{ app_name }}"
      state: link

  - "devops-keyword">"devops-keyword">name: Remove default site
    "devops-keyword">file:
      path: /etc/nginx/sites-enabled/default
      state: absent

  - "devops-keyword">"devops-keyword">name: Install SSL certificates
    "devops-keyword">acme_certificate:
      acme_directory: https://acme-v02.api.letsencrypt.org/directory
      account_email: admin@example.com
      certificate_name: "{{ domain }}"
      challenge: http-"devops-keyword">01

  "devops-keyword">handlers:
  - "devops-keyword">"devops-keyword">name: Restart Nginx
    "devops-keyword">service:
      "devops-keyword">name: nginx
      state: restarted
      enabled: yes
```

8. Monitoring & Logging

23 Prometheus & Grafana Monitoring Stack

PROMETHEUS MONITORING ARCHITECTURE

Components:

- **Prometheus Server:** Time-series database, data collection
- **Exporters:** Node Exporter, Blackbox Exporter, custom exporters
- **Alertmanager:** Handles alerts from Prometheus
- **Pushgateway:** For short-lived jobs
- **Grafana:** Visualization and dashboards

Key Concepts:

- **Metrics:** Time-series data with labels
- **Scrape Interval:** How often to collect metrics
- **PromQL:** Query language for metrics
- **Service Discovery:** Auto-discovery of targets

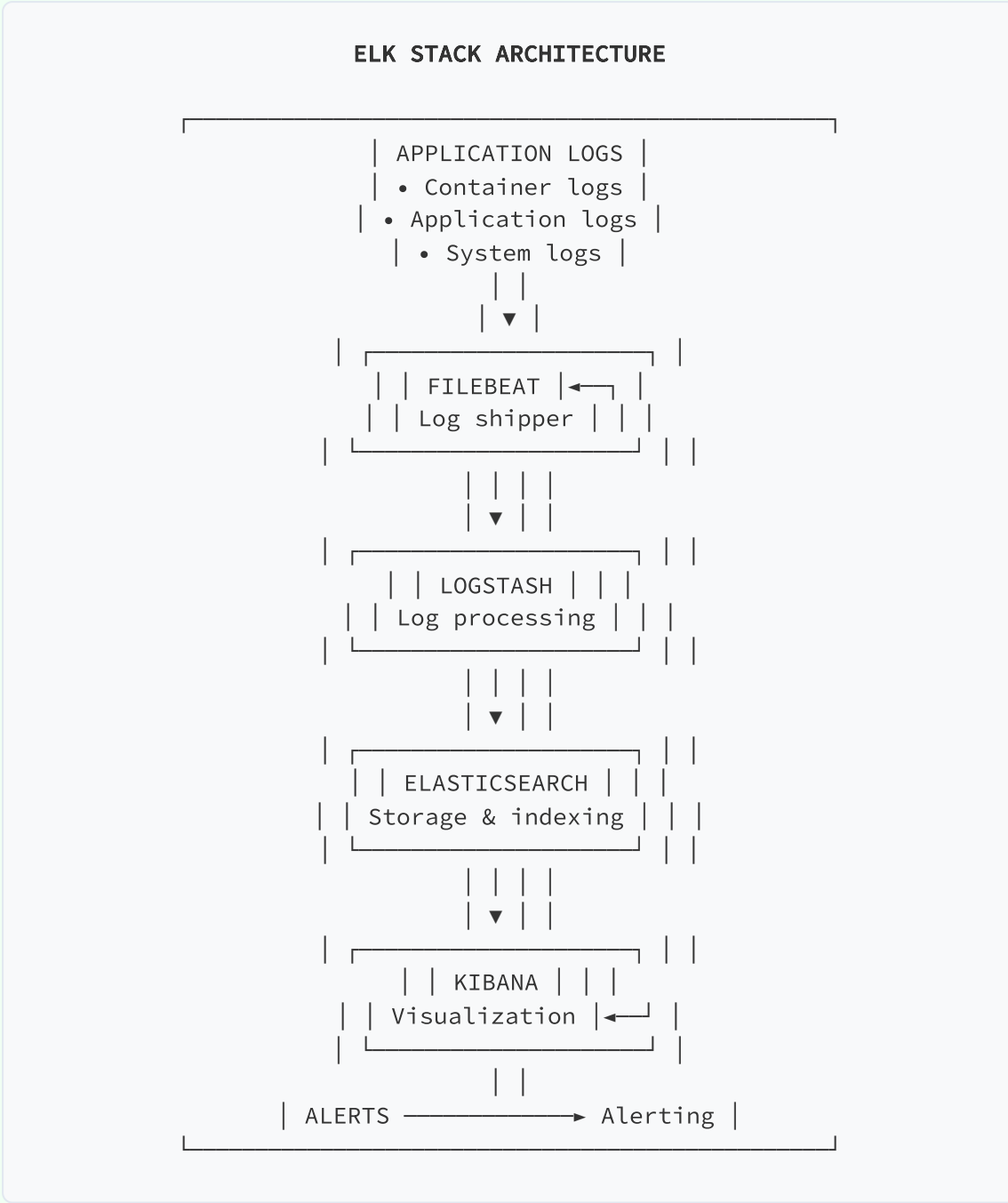
Prometheus Configuration:

```
 "comment">"comment"># prometheus.yml
"devops-keyword">global:
  "devops-keyword">scrape_interval: 15s
  "devops-keyword">evaluation_interval: 15s

"devops-keyword">rule_files:
  - "alerts/*.yml"

"devops-keyword">scrape_configs:
  - "devops-keyword">job_name: 'prometheus'
    "devops-keyword">static_configs:
      - "devops-keyword">targets: ['localhost:9090']

  - "devops-keyword">job_name: 'node'
    "devops-keyword">kubernetes_sd_configs:
      - "devops-keyword">role: node
    "devops-keyword">relabel_configs:
      - "devops-keyword">source_labels: [__address__]
        "devops-keyword">regex: '(.*):10250'
        "devops-keyword">replacement: '${1}:9100'
        "devops-keyword">target_label: __address__

  - "devops-keyword">job_name: 'kubernetes-pods'
    "devops-keyword">kubernetes_sd_configs:
      - "devops-keyword">role: pod
    "devops-keyword">relabel_configs:
      - "devops-keyword">source_labels: [__meta_kubernetes_pod_annotat
        "devops-keyword">action: keep
        "devops-keyword">regex: "devops-keyword">true
      - "devops-keyword">source_labels: [__meta_kubernetes_pod_annotat
        "devops-keyword">action: replace
        "devops-keyword">target_label: __metrics_path__
        "devops-keyword">regex: (.+)
      - "devops-keyword">source_labels: [__address__, __meta_kubernete
        "devops-keyword">action: replace
        "devops-keyword">regex: ([^:]+)(?::\d+)?;(\d+)
        "devops-keyword">replacement: $"devops-keyword">1:$"devops-key
        "devops-keyword">target_label: __address__

"devops-keyword">alerting:
  "devops-keyword">alertmanagers:
    - "devops-keyword">static_configs:
        - "devops-keyword">targets: ['alertmanager:9093']
```

Alert Rules Example:

```
 "comment">"comment"># alerts/node_alerts.yml
"devops-keyword">groups:
  - "devops-keyword">"devops-keyword">name: node_alerts
    "devops-keyword">rules:
      - "devops-keyword">alert: HighNodeCPU
        "devops-keyword">expr: "devops-keyword">100 - (avg by(instance
        "devops-keyword">for: 5m
        "devops-keyword">"devops-keyword">labels:
          severity: warning
        "devops-keyword">annotations:
          summary: "High CPU usage on {{ $">labels.instance }}"
          description: "CPU usage is above ">80% for ">5 minutes"

      - "devops-keyword">alert: HighNodeMemory
        "devops-keyword">expr: (node_memory_MemTotal_bytes - node_memo
        "devops-keyword">for: 5m
        "devops-keyword">"devops-keyword">labels:
          severity: warning
        "devops-keyword">annotations:
          summary: "High memory usage on {{ $">labels.instance }}"
          description: "Memory usage is above ">85% for ">5 minutes"
```

24 ELK Stack for Logging



Filebeat Configuration:

```
 "comment">"comment"># filebeat.yml
"devops-keyword">filebeat.inputs:
- "devops-keyword">"devops-keyword">type: container
  "devops-keyword">paths:
    - '/var/lib/docker/containers/*/*.log'
  "devops-keyword">json.keys_under_root: "devops-keyword">true
  "devops-keyword">json.add_error_key: "devops-keyword">true
  "devops-keyword">processors:
    - add_docker_metadata:
        host: "unix:///var/run/docker.sock"

- "devops-keyword">"devops-keyword">type: log
  "devops-keyword">enabled: "devops-keyword">true
  "devops-keyword">paths:
    - '/var/log/*.log'
    - '/var/log/syslog'
    - '/var/log/auth.log'

"devops-keyword">processors:
- add_cloud_metadata: ~
- add_host_metadata: ~

"devops-keyword">setup.kibana:
  "devops-keyword">host: "kibana:">5601"

"devops-keyword">output.logstash:
  "devops-keyword">hosts: ["logstash:">5044"]

"comment">"comment"># Or directly to Elasticsearch
"comment">"comment"># output.elasticsearch:
"comment">"comment"">#   hosts: ["elasticsearch:">9200"]
"comment">"comment"">#   username: "elastic"
"comment">"comment"">#   password: "changeme"
```

Logstash Configuration:

```
 "comment">"comment"># logstash.conf
"devops-keyword">input {
  beats {
    port => "devops-keyword">5044
  }
}

"devops-keyword">filter {
  grok {
    match => { "message" => "%{COMBINEDAPACHELOG}" }
  }

  date {
    match => [ "timestamp", "dd/MMM/yyyy:HH:mm:ss Z" ]
  }

  geoip {
    source => "clientip"
  }

  useragent {
    source => "agent"
    target => "useragent"
  }

  mutate {
    remove_field => [ "message", "timestamp" ]
  }
}

"devops-keyword">output {
  elasticsearch {
    hosts => ["elasticsearch:">9200"]
    index => "logs-%{+YYYY.MM.dd}"
    user => "elastic"
    password => "changeme"
  }

  stdout {
    codec => rubydebug
  }
}
```

25 Distributed Tracing with Jaeger

DISTRIBUTED TRACING CONCEPTS

Key Components:

- **Trace:** Entire request journey through services
- **Span:** Individual operation within a trace
- **Context Propagation:** Passing trace context between services
- **Sampling:** Deciding which traces to collect

Jaeger Architecture:

- **Jaeger Client:** Instrumentation libraries
- **Jaeger Agent:** Receives spans from clients
- **Jaeger Collector:** Receives spans from agents
- **Jaeger Query:** Query service for traces
- **Storage Backend:** Elasticsearch, Cassandra

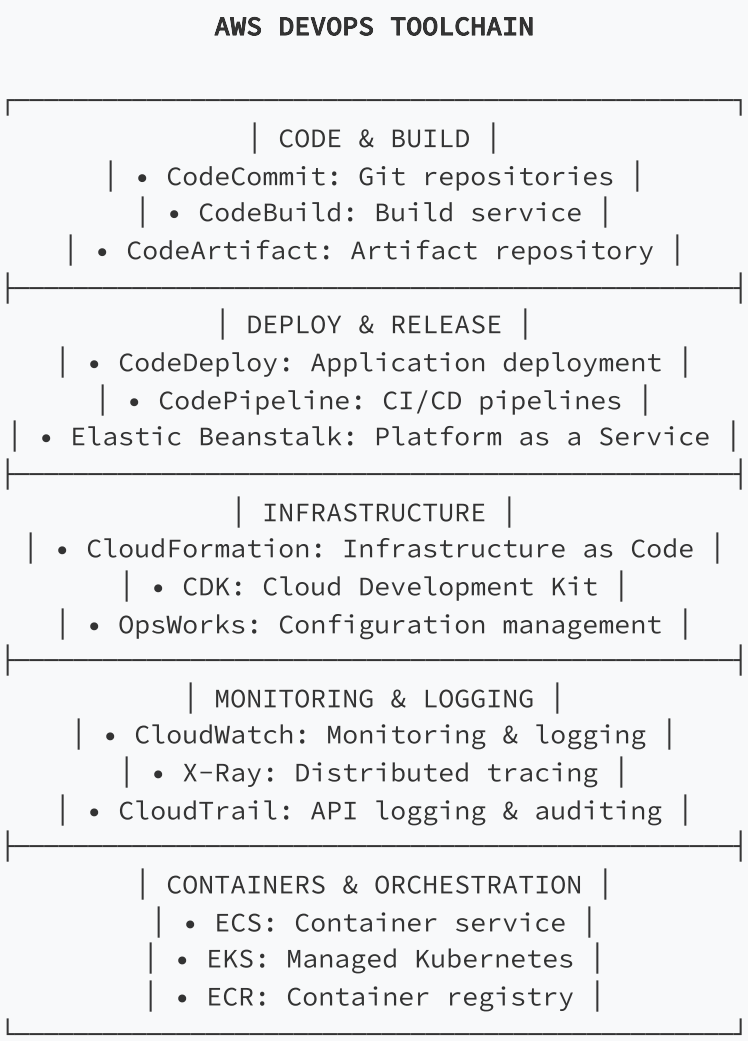
OpenTelemetry Instrumentation:

```
 "comment">"comment"># Node.js Application with OpenTelemetry
"devops-keyword">const opentelemetry = require("tool-">name">'@opentel
"devops-keyword">const { NodeTracerProvider } = require("tool-">name">
"devops-keyword">const { SimpleSpanProcessor } = require("tool-">name'
"devops-keyword">const { JaegerExporter } = require("tool-">name">'@op
"devops-keyword">const { Resource } = require("tool-">name">'@opentele
"devops-keyword">const { SemanticResourceAttributes } = require("tool-

"comment">// Create tracer provider
"devops-keyword">const provider = new NodeTracerProvider({
  resource: new Resource({
    [SemanticResourceAttributes.SERVICE_NAME]: 'api-service',
    [SemanticResourceAttributes.DEPLOYMENT_ENVIRONMENT]: 'production',
  }),
});

"comment">// Configure Jaeger exporter
"devops-keyword">const jaegerExporter = new JaegerExporter({
  endpoint: 'http://jaeger-collector:14268/api/traces',
});

"comment">// Add exporter to provider
provider.addSpanProcessor(new SimpleSpanProcessor(jaegerExporter));

"comment">// Register the provider
provider.register();

"comment">// Create tracer
"devops-keyword">const tracer = opentelemetry.trace.getTracer('api-tra

"comment">// Instrumented function
"devops-keyword">async "devops-keyword">function processOrder(orderId)
  "comment">// Create span for this operation
  "devops-keyword">const span = tracer.startSpan('process-order');

  try {
    "comment">// Add attributes to span
    span.setAttribute('order.id', orderId);
    span.setAttribute('customer.id', 'cust-123');

    "comment">// Add event to span
    span.addEvent('order-processing-started', {
      timestamp: Date.now(),
    });

    "comment">// Business logic
    "devops-keyword">const order = await validateOrder(orderId);
    "devops-keyword">const payment = await processPayment(order);
```

```
    const shipment = await createShipment(order);

    // Add more events
    span.addEvent('order-processing-completed');

    return { order, payment, shipment };
  } catch (error) {
    // Record error in span
    span.recordException(error);
    span.setStatus({
      code: opentelemetry.SpanStatusCode.ERROR,
      message: error.message,
    });
    throw error;
  } finally {
    // End the span
    span.end();
  }
}
```

9. Cloud & Security

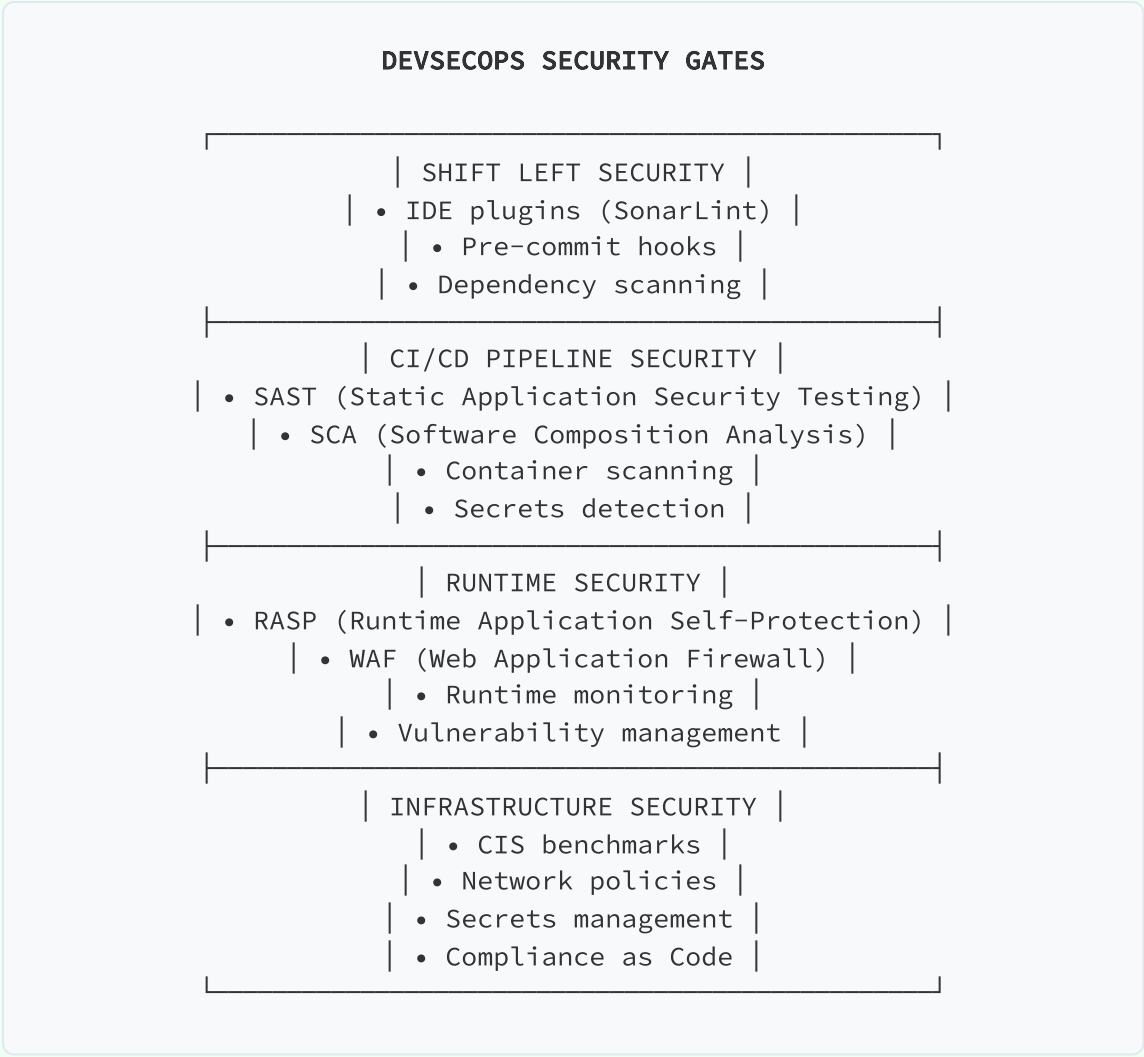
26 AWS DevOps Services Overview



AWS CodePipeline Example:

```
"comment">"comment"># buildspec.yml for AWS CodeBuild
"devops-keyword">"devops-keyword">version: "devops-keyword">0."devops-

"devops-keyword">phases:
  "devops-keyword">install:
    "devops-keyword">runtime-versions:
      nodejs: "devops-keyword">18
    "devops-keyword">commands:
      - npm ci

  "devops-keyword">pre_build:
    "devops-keyword">commands:
      - npm run lint
      - npm test

  "devops-keyword">build:
    "devops-keyword">commands:
      - npm run build
      - docker build -t $REPOSITORY_URI:latest .
      - docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG

  "devops-keyword">post_build:
    "devops-keyword">commands:
      - "command">echo Build completed on `date`
      - "command">echo Pushing the Docker images...
      - docker push $REPOSITORY_URI:latest
      - docker push $REPOSITORY_URI:$IMAGE_TAG
      - printf '[{"name":"webapp","imageUri":"%s"}]' $REPOSITORY_URI:$

"devops-keyword">artifacts:
  "devops-keyword">files:
    - imagedefinitions.json
    - appspec.yml
  "devops-keyword">discard-paths: yes

"devops-keyword">cache:
  "devops-keyword">paths:
    - node_modules/**/*
```

27 DevOps Security (DevSecOps)



Security Tools in CI/CD Pipeline:

```
 "comment">"comment"># GitLab CI/CD with security scanning
"devops-keyword">stages:
  - build
  - test
  - security
  - deploy

"devops-keyword">variables:
  DOCKER_DRIVER: overlay2

"devops-keyword">sast:
  stage: security
  image: registry.gitlab.com/gitlab-org/security-products/sast:latest
  variables:
    SAST_EXCLUDED_PATHS: "devops-keyword">spec, test, tests, tmp
  artifacts:
    reports:
      sast: gl-sast-report.json

"devops-keyword">dependency_scanning:
  stage: security
  image: registry.gitlab.com/gitlab-org/security-products/dependency-s
  artifacts:
    reports:
      dependency_scanning: gl-dependency-scanning-report.json

"devops-keyword">container_scanning:
  stage: security
  image: registry.gitlab.com/gitlab-org/security-products/container-sc
  variables:
    DOCKER_IMAGE: $CI_REGISTRY_IMAGE:$CI_COMMIT_SHA
    DOCKER_USER: $CI_REGISTRY_USER
    DOCKER_PASSWORD: $CI_REGISTRY_PASSWORD
  artifacts:
    reports:
      container_scanning: gl-container-scanning-report.json

"devops-keyword">secret_detection:
  stage: security
  image: registry.gitlab.com/gitlab-org/security-products/secret-detec
  artifacts:
    reports:
      secret_detection: gl-secret-detection-report.json

"devops-keyword">dast:
  stage: security
  image: registry.gitlab.com/gitlab-org/security-products/dast:latest
  variables:
```

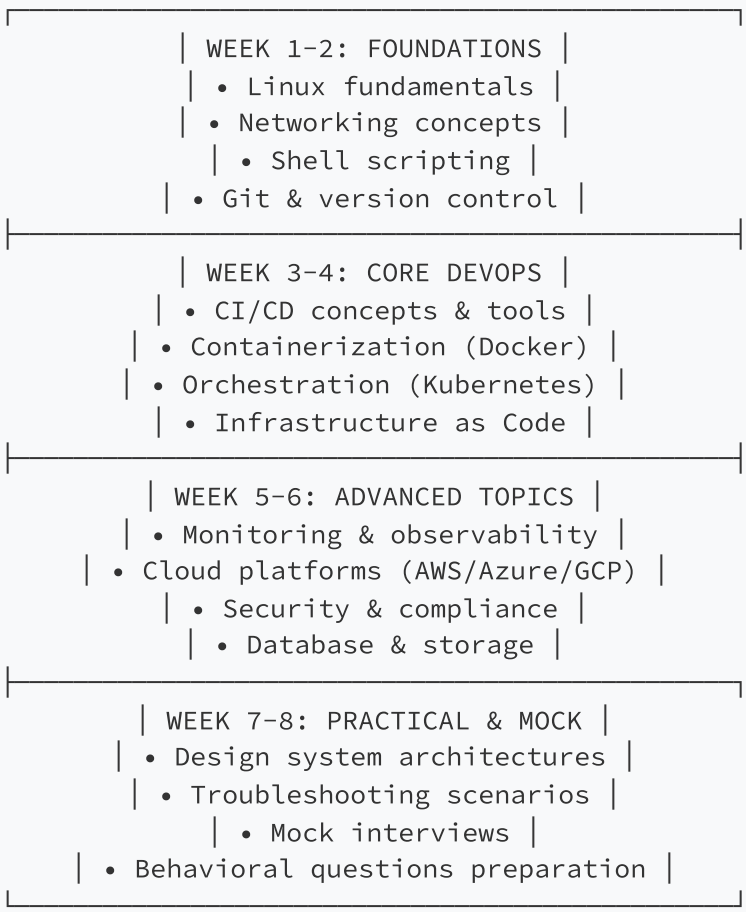
```
    DAST_WEBSITE: https://staging.example.com
    DAST_AUTH_URL: https://staging.example.com/login
    DAST_USERNAME: testuser
    DAST_PASSWORD: $DAST_PASSWORD
  artifacts:
    reports:
      dast: gl-dast-report.json
```

Security Best Practices:

1. **Secrets Management:**
   - Never commit secrets to version control
   - Use secret management tools (Vault, AWS Secrets Manager)
   - Rotate secrets regularly
   - Use short-lived credentials
2. **Infrastructure Security:**
   - Principle of least privilege
   - Network segmentation
   - Regular vulnerability scanning
   - Immutable infrastructure
3. **Compliance as Code:**
   - Automate compliance checks
   - Policy as Code (Open Policy Agent)
   - Continuous compliance monitoring
   - Audit trail for all changes

10. Interview Preparation Strategy

28 How to Prepare for DevOps Interviews?



Common Interview Question Categories:

1. **Conceptual Questions:**
   - Explain DevOps principles and benefits
   - Difference between CI and CD
   - Blue-Green vs Canary deployments
   - Monolithic vs Microservices architecture
2. **Technical Deep Dives:**
   - Design a CI/CD pipeline for given scenario
   - Kubernetes architecture and components
   - Terraform modules and state management
   - Docker networking and storage
3. **Troubleshooting Scenarios:**
   - Debug a failing Kubernetes pod
   - Investigate high CPU usage
   - Troubleshoot network connectivity
   - Debug slow application performance
4. **Hands-on Coding:**
   - Write Dockerfile for given application

- Create Kubernetes deployment YAML
- Write shell script to automate task
- Create Terraform module

29 Real-World DevOps Scenarios

Scenario 1: Migrating Monolith to Microservices

MIGRATION STRATEGY

Phase 1: Preparation

- Analyze current architecture and dependencies
- Identify bounded contexts for microservices
- Set up CI/CD pipeline for new services
- Implement service discovery and API gateway

Phase 2: Strangler Pattern

- Create new microservice for one functionality
- Route traffic through API gateway
- Gradually migrate features from monolith
- Implement feature flags for controlled rollout

Phase 3: Data Migration

- Implement database per service pattern
- Use event sourcing for data consistency
- Set up monitoring and observability
- Implement circuit breakers and retry logic

Scenario 2: Multi-Region Deployment Strategy

MULTI-REGION ARCHITECTURE

Active-Active Configuration:

- Deploy application in multiple regions
- Use Global Load Balancer (AWS Global Accelerator, Cloudflare)
- Implement database replication across regions
- Use CDN for static assets

Disaster Recovery:

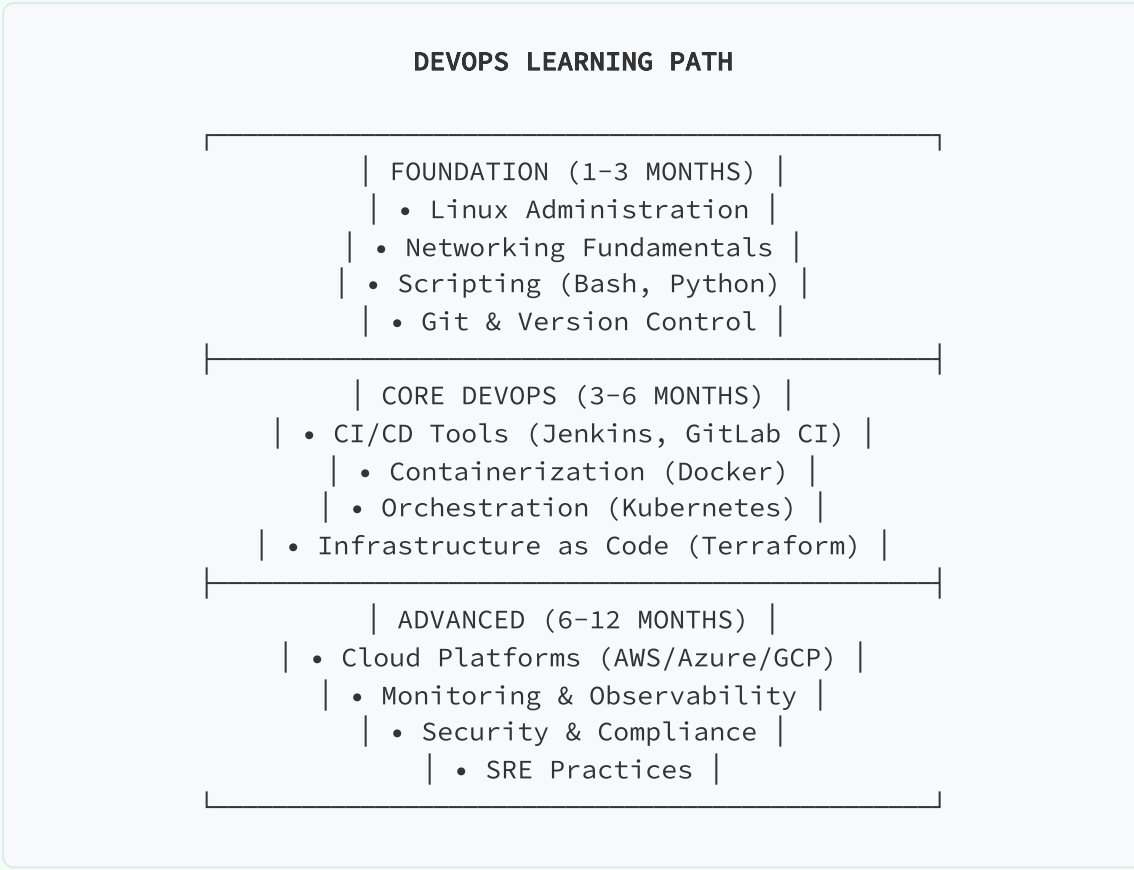
- Automated failover with Route 53 health checks
- Regular backup and restore testing
- Chaos engineering for resilience testing
- Documented runbooks for incident response

Data Consistency:

- Multi-master database with conflict resolution
- Eventual consistency model
- Caching strategy with Redis Cluster
- Data synchronization mechanisms

11. Additional Resources & Next Steps

Recommended Learning Path:



Certification Paths:

- **AWS:** DevOps Engineer Professional, Solutions Architect
- **Azure:** DevOps Engineer Expert, Solutions Architect
- **Google Cloud:** Professional Cloud DevOps Engineer
- **Kubernetes:** Certified Kubernetes Administrator (CKA)
- **Docker:** Docker Certified Associate
- **Terraform:** HashiCorp Certified Terraform Associate

Hands-on Projects to Build:

1. **Complete CI/CD Pipeline:** Jenkins/GitLab CI with Docker and K8s
2. **Infrastructure Automation:** Terraform modules for cloud resources
3. **Monitoring Stack:** Prometheus, Grafana, Alertmanager setup
4. **Logging Solution:** ELK/EFK stack implementation
5. **GitOps Workflow:** ArgoCD/Flux for GitOps deployments
6. **Security Pipeline:** SAST, DAST, container scanning integration

Communities & Resources:

- **Platforms:**
  - DevOps Subreddit (r/devops)
  - Kubernetes Slack Community
  - Docker Community Forums
  - CNCF (Cloud Native Computing Foundation)
- **Blogs & Newsletters:**
  - DevOps.com
  - InfoQ DevOps
  - The New Stack
  - Last Week in AWS
- **Books:**
  - "The Phoenix Project" - Gene Kim
  - "The DevOps Handbook" - Gene Kim et al.
  - "Site Reliability Engineering" - Google
  - "Kubernetes Up & Running" - Kelsey Hightower
- **Online Courses:**
  - Linux Foundation Training
  - A Cloud Guru
  - Udemy DevOps Courses
  - Coursera DevOps Specializations

Final Checklist Before Interview

- ✓ Review all 50 questions in this guide
- ✓ Practice hands-on with Docker and Kubernetes
- ✓ Build a complete CI/CD pipeline project
- ✓ Prepare 3-5 real project experiences to discuss
- ✓ Research the company's tech stack and challenges
- ✓ Practice explaining complex concepts simply
- ✓ Prepare troubleshooting scenarios
- ✓ Review system design principles
- ✓ Prepare questions for the interviewer
- ✓ Test your environment if interview is remote

💡 **Interview Day Tips:**

1. Explain your thought process clearly
2. Ask clarifying questions before solving problems
3. Admit when you don't know something
4. Show enthusiasm for learning and problem-solving

5. Discuss trade-offs in your solutions
6. Be prepared for both technical and behavioral questions

⚠️ **Common Pitfalls to Avoid:**

- Not understanding basic Linux commands
- Confusing Docker and Kubernetes concepts
- Unable to explain CI/CD pipeline stages
- Not considering security implications
- Forgetting about monitoring and observability
- Not asking questions about the role/team

Remember: DevOps is as much about culture and collaboration as it is about tools and technology. Show your ability to bridge development and operations, automate processes, and drive continuous improvement.

```
THE THREE WAYS OF DEVOPS (Gene Kim)

FIRST WAY: FLOW
Make work visible, reduce batch sizes, reduce intervals

SECOND WAY: FEEDBACK
Create fast feedback loops, amplify signals

THIRD WAY: CONTINUOUS LEARNING
Foster experimentation, learn from failures, improve continuously
```