

Deployment Tools - Exam Questions

Objective Questions (MCQ/MSQ) - 20 Questions

1. Which Markdown syntax creates a level 2 heading?

- A) `#` Heading
- B) `##` Heading
- C) `###` Heading
- D) `####` Heading

Answer: B

2. What does GitHub Pages primarily host?

- A) Dynamic web applications
- B) Static websites
- C) Database applications
- D) Mobile apps

Answer: B

3. Which Docker command builds an image from a Dockerfile?

- A) `docker create`
- B) `docker build`
- C) `docker make`
- D) `docker compile`

Answer: B

4. In GitHub Actions, what file defines workflows?

- A) `.github/workflows/*.yaml`
- B) `.github/actions/*.yaml`
- C) `.workflows/*.yaml`
- D) `.actions/*.yaml`

Answer: A

5. Which Google Colab command installs Python packages?

- A) `!install package`
- B) `!pip install package`
- C) `!conda install package`
- D) `!apt install package`

Answer: B

6. What does CORS stand for?

- A) Cross-Origin Resource Sharing
- B) Cross-Origin Request Security
- C) Cross-Origin Response Sharing
- D) Cross-Origin Resource Security

Answer: A

7. Which FastAPI decorator defines a GET endpoint?

- A) `@app.get()`
- B) `@app.route()`
- C) `@app.endpoint()`
- D) `@app.api()`

Answer: A

8. In Docker, which command runs a container?

- A) `docker start`
- B) `docker run`

- C) `docker execute`
- D) `docker launch`

Answer: B

9. Which ngrok command exposes a local port 8000?

- A) `ngrok http 8000`
- B) `ngrok expose 8000`
- C) `ngrok tunnel 8000`
- D) `ngrok port 8000`

Answer: A

10. What is the default port for Ollama API?

- A) `8000`
- B) `8080`
- C) `11434`
- D) `3000`

Answer: C

11. Which GitHub Codespaces file configures the development environment?

- A) `.devcontainer/devcontainer.json`
- B) `.codespaces/config.json`
- C) `.github/codespaces.yml`
- D) `.vscode/settings.json`

Answer: A

12. In Vercel, which file configures deployment settings?

- A) `vercel.json`
- B) `deploy.json`

- C) `config.json`
- D) `settings.json`

Answer: A

13. Which HTTP status code indicates successful resource creation?

- A) `200`
- B) `201`
- C) `202`
- D) `204`

Answer: B

14. What does the `--detach` flag do in Docker?

- A) Removes the container
- B) Runs container in background
- C) Stops the container
- D) Attaches to container

Answer: B

15. Which Google Auth scope provides read-only access to user profile?

- A) `profile`
- B) `email`
- C) `openid`
- D) `userinfo.profile`

Answer: A

16. In FastAPI, which import provides automatic API documentation?

- A) `from fastapi import docs`
- B) `from fastapi import swagger`
- C) Built-in with FastAPI
- D) `from fastapi import openapi`

Answer: C

17. Which Docker Compose command starts services?

- A) `docker-compose start`
- B) `docker-compose up`
- C) `docker-compose run`
- D) `docker-compose launch`

Answer: B

18. What does the `--watch` flag do in development servers?

- A) Monitors file changes and restarts
- B) Watches network traffic
- C) Monitors memory usage
- D) Watches user activity

Answer: A

19. Which GitHub Actions event triggers on pull requests?

- A) `on: pull_request`
- B) `on: pr`
- C) `on: merge_request`
- D) `on: pull`

Answer: A

20. In REST APIs, which HTTP method is idempotent?

- A) `POST`
- B) `PUT`

- C) PATCH
- D) All of the above

Answer: B

Subjective/Scenario Questions - 20 Questions

1. Static Site Deployment Strategy

Design a complete deployment pipeline for a data science portfolio website using GitHub Pages. Include content management, automated builds, custom domain setup, and performance optimization.

Answer: Use **Jekyll** or **Hugo** for static site generation, **GitHub Actions** for automated builds on content changes, **custom domain** with HTTPS via GitHub Pages, **CDN** for performance, **SEO optimization**, and **analytics integration**. Implement content versioning, automated testing, and deployment previews for pull requests.

2. Containerization Architecture

Design a Docker-based deployment for a machine learning API that handles model inference, data preprocessing, and result caching. What containers and orchestration would you use?

Answer: Create separate containers for **API service** (FastAPI), **model inference** (Python/GPU), **Redis cache**, and **database**. Use **Docker Compose** for local development, **Kubernetes** for production orchestration, **health checks**, **resource limits**, **secrets management**, and **horizontal scaling** based on load.

3. CI/CD Pipeline Design

Create a comprehensive CI/CD pipeline for a data science project that includes testing, model validation, deployment, and monitoring. What stages and tools would you include?

Answer: Stages: **code quality** (linting, testing), **data validation**, **model training/testing**, **security scanning**, **staging deployment**, **integration testing**, **production deployment**, **monitoring setup**. Use **GitHub Actions**, **Docker**, **automated rollbacks**, **blue-green deployment**, and **performance monitoring**.

4. Multi-Environment Deployment

Design a deployment strategy that supports development, staging, and production environments with different configurations and data sources. How would you manage environment-specific settings?

Answer: Use **environment variables** for configuration, **separate Docker images** or **config overlays**, **infrastructure as code** (Terraform), **environment-specific secrets**, **database migrations**, **feature flags**, and **automated promotion** between environments with approval gates.

5. API Gateway Architecture

Design an API gateway solution for multiple microservices including authentication, rate limiting, logging, and load balancing. What components would you include?

Answer: Implement **reverse proxy** (Nginx/Traefik), **authentication middleware**, **rate limiting**, **request/response logging**, **load balancing**, **circuit breakers**, **API versioning**, **CORS handling**, **metrics collection**, and **health checks**. Use **Kong**, **AWS API Gateway**, or **custom FastAPI** middleware.

6. Serverless Deployment Strategy

Compare serverless vs. traditional deployment for a data processing pipeline. Design both approaches and recommend when to use each.

Answer: Serverless: Use **AWS Lambda/Azure Functions** for event-driven processing, **API Gateway** for HTTP endpoints, **managed databases**, **auto-scaling**, **pay-per-use**. **Traditional:** Use **containers** with **orchestration**, **persistent storage**, **predictable costs**. Choose serverless for sporadic workloads, traditional for consistent high-volume processing.

7. Security Implementation

Implement comprehensive security for a deployed data science application including authentication, authorization, data encryption, and vulnerability management.

Answer: Implement **OAuth 2.0/JWT** authentication, **RBAC** authorization, **HTTPS/TLS** encryption, **secrets management**, **input validation**, **SQL injection prevention**, **dependency scanning**, **container security**, **network segmentation**, **audit logging**, and **regular security assessments**.

8. Monitoring and Observability

Design a monitoring solution for deployed data science applications that tracks performance, errors, business metrics, and user behavior.

Answer: Implement **application metrics** (Prometheus), **logging** (ELK stack), **distributed tracing** (Jaeger), **error tracking** (Sentry), **uptime monitoring**, **business KPIs**, **alerting** (PagerDuty), **dashboards** (Grafana), and **automated incident response**.

9. Scalability Architecture

Design a scalable architecture for a machine learning inference service that can handle variable load patterns and maintain low latency.

Answer: Use **horizontal pod autoscaling**, **load balancers**, **caching layers** (Redis), **async processing** (Celery), **database read replicas**, **CDN** for static assets, **connection pooling**, **circuit breakers**, **graceful degradation**, and **performance monitoring** with auto-scaling triggers.

10. Disaster Recovery Planning

Create a disaster recovery plan for a critical data science application including backup strategies, failover procedures, and recovery testing.

Answer: Implement **automated backups**, **multi-region deployment**, **database replication**, **infrastructure as code** for quick recovery, **runbook documentation**,

regular DR drills, RTO/RPO definitions, monitoring and alerting, communication plans, and post-incident reviews.

11. Cost Optimization Strategy

Optimize deployment costs for a data science platform while maintaining performance and reliability. What strategies would you implement?

Answer: Use **right-sizing** instances, **spot instances** for batch processing, **auto-scaling** to match demand, **reserved instances** for predictable workloads, **storage tiering**, **CDN** for static content, **resource monitoring**, **cost alerts**, **regular cost reviews**, and **efficiency metrics**.

12. Compliance and Governance

Implement deployment practices that comply with regulatory requirements (SOC 2, HIPAA, GDPR). What controls and processes would you establish?

Answer: Implement **access controls**, **audit logging**, **data encryption**, **change management**, **vulnerability management**, **incident response**, **regular assessments**, **documentation**, **training programs**, **third-party risk management**, and **compliance monitoring**.

13. Development Workflow Integration

Design deployment workflows that integrate seamlessly with data science development practices including notebook development, model experimentation, and collaborative work.

Answer: Use **GitOps** workflows, **notebook-to-production** pipelines, **model registries**, **experiment tracking**, **automated testing**, **staging environments**, **feature branches**, **code reviews**, **documentation generation**, and **collaboration tools** integration.

14. Performance Optimization

Optimize the performance of deployed data science applications including response times, throughput, and resource utilization.

Answer: Implement **caching strategies, database optimization, code profiling, async processing, connection pooling, CDN usage, image optimization, lazy loading, resource monitoring, performance testing, and continuous optimization.**

15. Multi-Cloud Strategy

Design a multi-cloud deployment strategy that avoids vendor lock-in while maintaining operational efficiency. What challenges would you address?

Answer: Use **containerization** for portability, **infrastructure as code, cloud-agnostic tools, data synchronization, network connectivity, cost management, compliance consistency, skill requirements, operational complexity, and disaster recovery** across clouds.

16. Edge Computing Deployment

Design a deployment strategy for machine learning models that need to run at the edge with limited connectivity and resources.

Answer: Use **model optimization** (quantization, pruning), **edge computing platforms, offline capabilities, data synchronization, remote management, security measures, resource constraints, update mechanisms, monitoring solutions, and failover strategies.**

17. API Versioning Strategy

Implement API versioning for a data science service that needs to maintain backward compatibility while introducing new features.

Answer: Use **semantic versioning, URL versioning (/v1/, /v2/), header versioning, deprecation policies, migration guides, automated testing, documentation, client SDKs, monitoring usage, and sunset timelines** for old versions.

18. Database Migration Strategy

Plan and execute database migrations for a production data science application with minimal downtime and data loss risk.

Answer: Use **blue-green deployment, database versioning, migration scripts, rollback procedures, data validation, performance testing, backup strategies, monitoring, staged rollouts, and communication plans** for stakeholders.

19. Automated Testing in Production

Implement automated testing strategies for deployed data science applications including integration tests, performance tests, and chaos engineering.

Answer: Implement **health checks, integration tests, load testing, chaos engineering, canary deployments, A/B testing, synthetic monitoring, automated rollbacks, test data management, and continuous validation** of model performance.

20. Vendor Management

Manage relationships with cloud providers and third-party services for data science deployments. What governance and risk management practices would you implement?

Answer: Establish **vendor evaluation criteria, SLA management, cost monitoring, security assessments, compliance verification, contract management, exit strategies, performance monitoring, relationship management, and risk mitigation** plans.