☑ 11. CD — 50 Preguntas Técnicas (Respuestas)

---

1. ¿Qué es Continuous Delivery y en qué se diferencia de Continuous Deployment?

- **Continuous Delivery (CD):** el código está *siempre listo* para desplegar, pero el despliegue a producción requiere un **paso manual**.
- **Continuous Deployment:** cada cambio que pasa CI se **despliega automáticamente** sin intervención humana.

---

2. Beneficios de CD en proyectos de datos

- Versionado y despliegue seguro de pipelines, modelos y DBT
- Reducción de errores humanos
- Menos tiempo entre cambios y valor en producción
- Control de versiones de esquemas y transformaciones
- Integración con data quality gates

---

3. Herramientas de CD usadas

GitLab CI/CD, GitHub Actions, Jenkins, ArgoCD, Spinnaker, FluxCD, Harness.

---

4. Etapas típicas de un pipeline de CD

- Build
- Test
- Generate artifacts
- Provision environments
- Deploy (blue-green, rolling, canary)
- Validate (smoke tests)
- Monitoring & rollback

---

5. Blue-green deployment

Dos entornos paralelos:

- **Blue:** producción actual
- **Green:** nueva versión
  Se redirige el tráfico al entorno green cuando se valida.

---

6. Canary deployment

Despliegue gradual hacia un porcentaje pequeño de usuarios antes de extenderlo a todos.

---

7. Rolling deployment

Actualización por lotes/instancias sin downtime.

---

8. Feature flag

Activar/desactivar funcionalidades sin redeploy; útil para experimentos, canary y rollbacks rápidos.

---

9. Integración de CD con control de versiones

Cada commit/PR dispara el pipeline; el despliegue se basa en tags, branches o releases.

---

10. Artifact repository

Lugar centralizado para almacenar artifacts: paquetes, contenedores, wheels, JARs.

---

11. Promotion of artifacts

Paso de artifacts aprobados por diferentes ambientes (dev → test → prod) sin volverlos a build.

---

12. Environment provisioning

Creación automática de entornos (infra, bases, buckets, redes) usando IaC.

---

13. Infrastructure as Code (IaC)

Definir infraestructura en archivos versionados (Terraform, CloudFormation).

---

14. Terraform vs CloudFormation vs Deployment Manager

- **Terraform:** multi-cloud, modular, más flexible.
- **CloudFormation:** solo AWS.
- **Deployment Manager:** solo GCP (ya deprecated).

---

15. Manejo de secretos en CD

- Vault
- Secret Manager
- Variables protegidas
- KMS

---

16. Deployment approval

Gate manual para autorizar despliegues antes de producción.

---

17. Métricas importantes en CD

- Deployment frequency
- Lead time
- Change failure rate
- MTTR (Mean Time to Recovery)
- Éxito/fallo por release

---

18. Staging vs production

- **Staging:** entorno casi idéntico para pruebas.
- **Production:** tráfico real, datos reales.

---

19. Rollback y cómo implementarlo

Volver a una versión previa mediante:

- Cambiar tag de contenedor
- Feature flags
- Deshacer migraciones
- Restaurar snapshots

---

20. Idempotencia en deployments

Ejecutar el mismo despliegue varias veces debe producir el mismo resultado.

---

21. Test-driven deployment

Solo se despliega si los tests pasan; incluye pruebas de regresión y datos.

---

22. Integrar pruebas automáticas antes de producción

Jobs `pre-deploy` con:

- Unit tests
- Integration tests

- dbt tests
- Data quality checks
- Contract testing

---

23. Smoke test

Prueba rápida para validar si la aplicación/pipeline arranca correctamente.

---

24. End-to-end test

Prueba completa que cubre el flujo total (origen → transformación → destino).

---

25. Manejo de dependencias entre servicios

- Versionado semántico
- Tests contractuales
- Despliegues independientes
- Backward compatibility

---

26. Container registry

Repositorio para imágenes Docker (ECR, GAR, ACR, DockerHub).

---

27. Docker image vs Helm chart

- **Docker image:** empaqueta una aplicación.
- **Helm chart:** define cómo desplegarla en Kubernetes.

---

28. Continuous configuration

Actualización continua de parámetros sin redeploy (ConfigMaps, SSM, etc.).

29. Observability en CD

Monitoreo de logs, métricas y trazas para validar despliegues.

---

30. Métricas para monitorear deployments

- Latencia
- Errores
- Saturación
- Tasa de éxito por release
- Tiempo de rollback

---

31. Deployment drift

Diferencias entre el estado deseado (IaC) y el real.
Se evita con **GitOps**.

---

32. Chaos testing

Introducir fallas intencionales para validar resiliencia del sistema.

---

33. Pipeline promotion strategy

Promover versiones a entornos superiores solo si pasan criterios definidos.

---

34. Immutable infrastructure

No se modifica; en vez de actualizar, se **reemplaza** por completo.

---

35. CD para aplicaciones vs datos

Aspecto	Apps	Datos
Rollback	Fácil	Difícil (datos ya transformados)
Testing	Unit/Integration	Data quality, esquemas
Artifacts	Contenedores	Pipelines, SQL, DBT, DAGs
Riesgos	Downtime	Corrupción de datos

---

36. A/B deployment

Dos versiones activas reciben tráfico dividido para experimentación.

---

37. Rollback en bases de datos

- Migraciones reversibles
- Versionado de schema
- Snapshots
- Tablas shadow

---

38. Versionar modelos de datos en CD

- dbt versions
- Semantic layer
- Cambios controlados en schema
- Contract testing

---

39. Semantic versioning en CD

MAJOR.MINOR.PATCH para controlar compatibilidad entre releases.

---

40. Deployment gate

Condición obligatoria antes de pasar a producción:

- QA OK
- Seguridad
- Data quality gate
- Approval manual

---

41. Documentación de pipelines de CD

En repositorios: README, diagramas, scripts, flujos, versiones, endpoints.

---

42. Problemas comunes en CD y mitigación

- Drifts → usar GitOps
- Secretos → usar vault
- Deployments lentos → paralelización
- Rollbacks difíciles → migraciones reversibles

---

43. Seguridad en pipelines de CD

- Escaneo de imágenes
- Validación de dependencias
- Secretos cifrados
- Roles mínimos

---

44. Canary analysis automatizada

Evaluar métricas del canary con herramientas como Argo Rollouts.

---

45. Multi-region deployments

- Replicación
- Failover automático
- IaC multi-región
- Versionado consistente

---

46. Progressive delivery

Despliegue gradual con telemetría en tiempo real.

---

47. Pipeline as code aplicado a CD

Todo el proceso de despliegue está en código versionado (YAML, Helm, Terraform).

---

48. Push deployment vs pull deployment

- **Push:** el CI/CD empuja a los servidores.
- **Pull (GitOps):** los entornos leen del repositorio y actualizan su estado.

---

49. Consideraciones cloud vs on-prem

- Cloud: autoscaling, servidores gestionados, seguridad integrada
- On-prem: control total, más mantenimiento

---

50. Buenas prácticas CD para datos/BI

- Validar esquemas antes de despliegue
- Data quality gates
- Deploys con IAAC + dbt
- Canary para pipelines críticos
- Rollbacks seguros de datos
- Métricas de latencia y frescura